IBM Page Printer Formatting Aid: User's Guide

S544-5284-05

IBM Page Printer Formatting Aid: User's Guide

S544-5284-05

Note

Before using this information and the product it supports, be sure to read the general information in “About This Publication” on page xi and “Notices” on page 399.

Sixth Edition (December 2000)

This edition applies to Version 3 Release 7 Modification Level 1 (V3R7M1) of Page Printer Formatting Aid (PPFA) for OS/400, Version 1 Release 1 of PPFA for System/390, Version 3 Release 2 of PPFA for Infoprint Manager for AIX, Version 1 Release 1 of PPFA for Infoprint Manager for Windows NT and Windows 2000 , and to all subsequent releases of this product until otherwise indicated in new releases or technical newsletters, and replaces the following publications: *IBM Page Printer Formatting Aid/370: User’s Guide and Reference* (S544-3700-02), *IBM Page Printer Formatting Aid/6000: User’s Guide* (S544-3918-01), and *IBM Page Printer Formatting Aid: User’s Guide* (S544-5284-04).

See the Summary of Changes for the changes made to this publication. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

The IBM Printing Systems Division welcomes your comments. A form for reader’s comments is provided at the back of this publication. If the form has been removed, you may send your comments to the following address:

INFORMATION DEVELOPMENT
THE IBM PRINTING SYSTEMS DIVISION
DEPARTMENT H7FE, BUILDING 003G
PO BOX 1900
BOULDER, CO 80301–9191

If you prefer to send comments electronically, use one of the following methods:

- Internet: printpub@us.ibm.com
- Fax: 1-800-524-1519 or 1-303-924-6873

Internet

Visit our home pages at http://www.ibm.com/printers

Contents

Figures

Tables

About This Publication

This publication describes how to use the Page Printer Formatting Aid (PPFA) to create and compile page definitions and form definitions for printing or viewing files with Advanced Function Presentation products, such as IBM Print Services Facility.

Who Should Use This Publication?

This publication is for anyone who wants to use PPFA to create form definitions and page definitions (traditional and record format). This publication has been written assuming that you are one of the following:

- A first-time user

  You are using PPFA for the first time to create form definitions and page definitions. You are familiar with system commands, but you are not familiar with Print Services Facility (PSF) concepts and Page Printer Formatting Aid parameters. You should read all of the information contained in this publication, and then use it as a reference.

  For more information about Advanced Function Presentation concepts, refer to *Guide to Advanced Function Presentation*.

  For more information about AIX concepts, refer to *IBM Print Services Facility for AIX: AIX for Users of Print Services Facility*.

  For more information about OS/400 concepts, refer to OS/400 User's Guide.

  For more information about VSE, MVS, VM, or OS/400, refer to the Application Programming Guide for the platform you are using.

- An intermediate user

  You are familiar with print server concepts and with Page Printer Formatting Aid parameters and you know the difference between a logical page and a physical page. You already know how to create and use form definitions and page definitions. Use this publication as a reference to learn more about PPFA commands and syntax. Refer to the examples for useful information.

- An advanced user

  You understand print server concepts and have used PPFA to create form definitions and page definitions. You understand the use of data stream processing. You will use this publication mostly as a reference. Chapter 5. Creating Complex Printouts might be especially helpful.

Note: Not all of the functions provided by PPFA are supported in all print server licensed programs. Refer to the information for the print server licensed program that you are using to determine which functions are supported. For more information about a specific environment, see Appendix A. System Dependencies for PPFA for the steps required to process page definitions and form definitions.

How This Publication Is Organized

You can use this publication both as a guide and as a reference to help you learn about the following:

- Chapter 1. Introducing Page Printer Formatting Aid summarizes PPFA and describes the purpose of form definitions and page definitions. Key PPFA concepts and terms are defined in this section.
- Chapter 2. Using Form Definition Commands shows examples illustrating the use of basic form-definition controls for traditional line data.
- Chapter 3. Using Page Definition Commands for Traditional Line Data shows examples illustrating the use of basic page-definition controls.
- Chapter 4. Using Page Definition Commands for Record Format Line Data shows examples illustrating the use of basic form-definition controls for record format line data.
- Chapter 5. Creating Complex Printouts shows examples of print jobs that require advanced use of form-definition and page-definition controls.
- Chapter 6. Conditional Processing shows examples of conditional processing used in formatting complex printing applications.
- Chapter 7. N_UP Printing describes how you can use N_UP printing.
- Chapter 8. PPFA Command Syntax defines the rules and the syntax for writing a set of PPFA commands.
- Chapter 9. Form Definition Command Reference defines all the PPFA form-definition commands, their subcommands, and their parameters.
- Chapter 10. Page Definition Command Reference (Traditional) defines all the PPFA traditional page-definition commands, their subcommands, and their parameters.
- Chapter 11. Page Definition Command Reference (Record Formatting) defines all the PPFA record format page-definition commands, their subcommands, and their parameters.
- Appendix A. System Dependencies for PPFA shows the steps needed to create and use form definitions and page definitions in VSE, MVS, VM, AIX, and OS/400 systems.
- Appendix B. More about Direction expands on the direction information and includes a lookup table.
- Appendix C. Differences in Measurements and REPEATs with AFP Utilities describes the differences in printing with measurements and REPEATs between PPFA, OGL, and PMF.
- Appendix D. More About Bar Code Parameters contains supplemental information about bar codes.
- Appendix E. PPFA Keywords contains lists of PPFA symbols and keywords.
- Appendix F. PPFA Media Names contains a list of media names, types, and component identifiers.
- Appendix G. Fill Patterns for Drawgraphic Commands contains examples of fill patterns for the Drawgraphic commands.
- Appendix H. PPFA Messages and Codes lists all diagnostic messages generated by PPFA and suggests a cause and solution for each.

Notices to include trademarks, a glossary of terms, a bibliography, and an index are included at the back of the publication.

Reading Syntax Diagrams

The syntax for PPFA commands is shown using graphic notation. To read the diagrams, move from left to right and top to bottom, following the main path line.

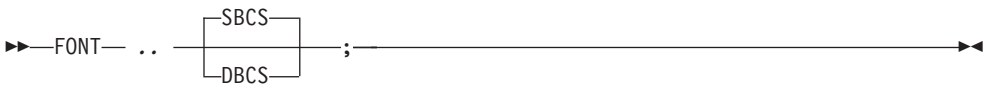
Style Rules:

Syntax diagrams use the following style rules to show how to enter commands and parameters:

- A word in uppercase must be spelled exactly as shown, but may be coded in any case. For example in coding, FORMDEF or FormDef or formdef are equivalant.
- A word in all italic, lowercase letters shows a parameter that you can replace. For example:

  name

  shows that you replace *name* with a resource name that is retained in the library.
- A parameter above the line shows the default parameter. For example, SBCS is the default parameter in the syntax diagram for the FONT command:



Symbols:

Syntax diagrams use symbols to help you follow the flow of information they communicate:

- Statements begin with:

  ▶▶─
- and end with:

  ─▶◀
- Statements longer than one line contine to a second line with:

  ─▶
- Where they resume with:

  ▶─

Required Parameters:

A parameter that you must include is displayed on the main path line. For example, the syntax diagram for the SEGMENT command:

shows that you must follow SEGMENT with its required parameter.

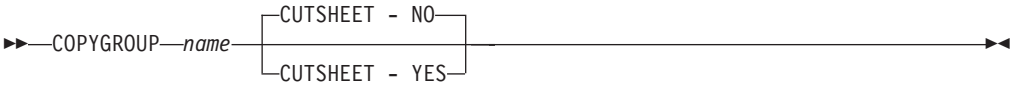
If there are two or more required parameters from which to choose, the parameters are shown with the first choice on the main path line and the other choices on branch lines under it. For example, the partial syntax diagram for the DIRECTION Command:

shows that you must type the command in any of the following ways:

- DIRECTION ACROSS
- DIRECTION DOWN
- DIRECTION BACK
- DIRECTION UP

Optional Parameters:

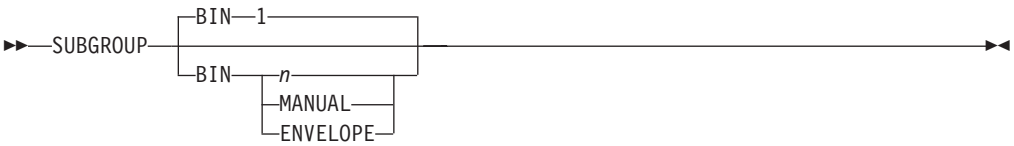
Parameters that you can include with a command are displayed on the branch line below the main path line. For example, the partial syntax diagram for the COPYGROUP command:



shows you can type the command in one of these ways:

- COPYGROUP *name1* CUTSHEET YES ;
- COPYGROUP *name1* CUTSHEET NO ;
- COPYGROUP *name1*;

Branch lines can include branch lines of their own. An example of this is the partial syntax diagram for the SUBGROUP command with the optional BIN parameter:



Repeating Parameters:

An arrow on a line above a parameter means that you can either repeat the parameter or enter more than one of the listed parameters. An example of this is the partial syntax diagram for the SUPPRESSION subcommand in the SUBGROUP Command:

The arrow above *name* means you can include one or more field name parameters with the SUPPRESSION command.

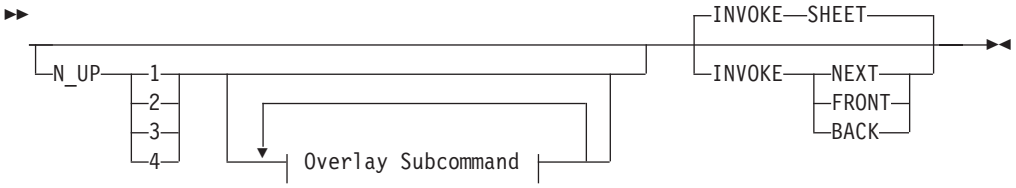
Fragment Elements

A syntax diagram can contain a section that either has too many items or groups to fit in the diagram or is used more than once. This section can be presented as a ″fragment″, and given a label that corresponds to the section within the main

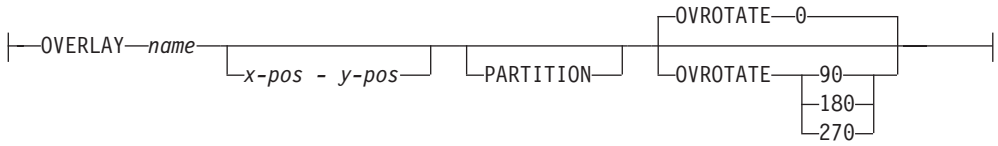
diagram. An example of this is the syntax diagram for the FORMDEF subcommand with its fragmented Overlay subcommand shown below:

Note: This FORMDEF diagram example also displays examples of some of the parameters mentioned above.

FORMDEF



Overlay Subcommand:



Related Information

See the Bibliography section for lists of publications, by category, that may be helpful to you as you use PPFA.

Publications that are referred to in this document or that contain additional information about Advanced Function Presentation (AFP), the MVS operating system, print server, and related products and systems are listed in the Bibliography.

The following is a listing of WEB sites that may be helpful to you as you use PPFA:

- RS6000 and AIX = http://www.rs6000.ibm.com/support/
- Guide to IBM BookManager BookServer = http://rs3bt.cae.de.ibm.com/

Summary Of Changes

Current changes (December 2000) between this edition and the previous edition are marked by a "|" in the left margin.

This publication contains revised formatting commands to include the new record formatting, eight new media names, two new bar codes, definition of color models to be used for graphics, text and rules, and an example of fill patterns that can be used with drawgraphic commands.

This publication also includes the Windows NT and Windows 2000 environment information and provides the necessary **ppfa** commands that create form definitions and page definitions on the Windows NT and Windows 2000 operating systems. After they are created, you can transfer the form definitions and page definitions to other operating systems (such as OS/390, VM, or VSE) to use as AFP resources. Contact your IBM representative to order PPFA for Infoprint Management for Windows NT and Windows 2000.

Part 1. What is PPFA?

Chapter 1. Introducing Page Printer Formatting Aid

Page Printer Formatting Aid (PPFA) is an IBM licensed program that enables users of IBM's Advanced Function Presentation (AFP) products to create their own formatting resources, called form definitions and page definitions. The form definitions and page definitions are stored in libraries [1] as AFP resources. Using AFP resources requires IBM print servers Facility, a licensed program or feature, which merges resources with user data files. This merging creates a data stream for printing or viewing.

Using a form definition or a page definition created by PPFA requires you to perform three steps:

1. Write a set of PPFA commands that define how to position the data or handle the physical sheets.
2. Run PPFA to build the specified page definition or form definition and store the output as resources in a library.
3. Submit the print file using print server , specifying the page definition and form definition needed to accomplish the desired results.

Figure 1 on page 4 shows how form definition and page definition relate to print servers Facility. In Figure 1 on page 4, the area inside the broken line represents steps 1 and 2. The area outside of the broken line shows how print server merges resources with the specified print job to form a single print stream and sends it to a page printer.

1. For purposes of this book, the term "library" includes AIX directories as well as OS/390, VSE, and OS/400 libraries and VM files.

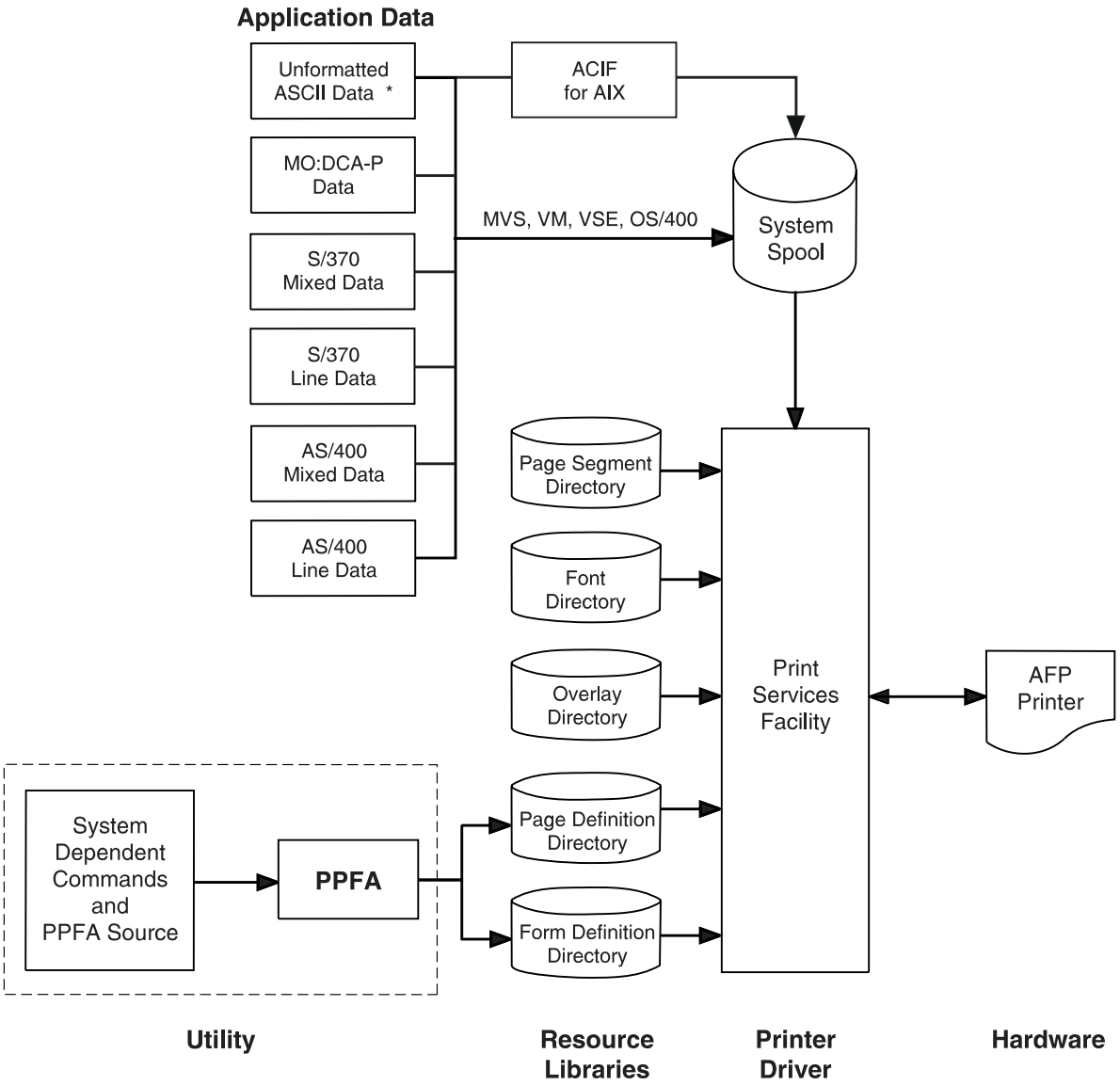


Figure 1. Form Definition and Page Definition Environment

Note: Figure 1 is a general representation for operating systems that use PPFA. However, print server users in the OS/400 environment do not use unformatted ASCII data. Also, print server users in the OS/400, VSE, OS/390, and VM environments should substitute the word "Directory" for the system-specific file organization (for example, OS/390 library).

Summary of a Form Definition

A PPFA command stream can contain form-definition commands. A *form definition* specifies how the printer controls the processing of the physical sheets of paper. In a form definition, you can specify modifications that distinguish formatting one print job from another when both are derived from the same data. Form definitions are used for all print server print files regardless of data type.

Form definitions can specify the following functions:

- Position of a logical page on a physical page
- Duplex printing
- Inclusion of overlays, which substitute for preprinted forms
- Flash (the use of a forms flash—only on 3800 printers)
- Selection of the number of copies for any page of data
- Suppression (the exclusion of selected fields of data in one printed version of a page of data but not in another)
- Jog (the offset stacking of cut-sheet output or copy marking on continuous-forms output)
- Selection among paper sources in a cut-sheet printer
- Adjustment of the horizontal position of the print area on the sheet (only on 3800 printers)
- Quality (selection among print quality levels)
- Constant (allows front or back printing of a page without variable data)
- Printing one, two, three, or four logical pages on a single side of a page
- Postprocessing controls, such as:
  - Selecting functions

    Selecting device-dependent functions defined by the postprocessing device
- Perforating
- Cutting

Summary of a Page Definition

A *page definition* specifies how you want data positioned on the logical page. A page definition can control the following functions:

- Dimensions of the logical page
- Print direction of the logical page
- Print direction of text lines and fields relative to the logical page
- Conditional processing (different formats on different pages, based on content of data)
- Text line spacing (number of lines per inch)
- Location of individual text lines and fields
- Number of text lines per page
- Page segments for inclusion in printed output
- Overlays for inclusion in printed output (positioned anywhere on the page)
- Page-ejection points
- Fonts and font rotation used on a page
- Multiple-up printing (placing more than one subpage on one side of a single sheet)
- Colors to be used (on printers that support this function)

Formatting Output of Different Data File Types

The four basic types of data printed on print server printers are:

Line-data files
Mixed-data files
MO:DCA-P data files (called AFPDS in OS/400)
Unformatted ASCII files (typically AIX)

Line-data files, mixed-data files, and unformatted ASCII require a page definition and a form definition. MO:DCA-P data files require only a form definition.

Line-Data Files

Line data is EBCDIC data that is arranged for printing on line printers. These records may contain line-printer control characters such as carriage control characters (CC or FCFC), table-reference characters (TRC), or only data. To compose pages for the page printer from line data, print servers Facility separates the incoming print records into pages according to specifications in a page definition. A page definition is always required for printing line data with print server . You can create your own page definition or use a page definition provided with print server . There are two types of line data: traditional and record format.

The line data input to print server can consist of records that are fully formatted; it can consist of records that contain only the fields of data to be printed; or it can consist of records of both types. You can use the page definition resource to format fields of line data outside of the application program. Refer to *Print Servers Facility for OS/390 User's Guide, Version 3, Release 2.0* for additional information.

The following example shows two types of line data. The first type shows data arranged as it prints out and the second shows data that requires field processing.

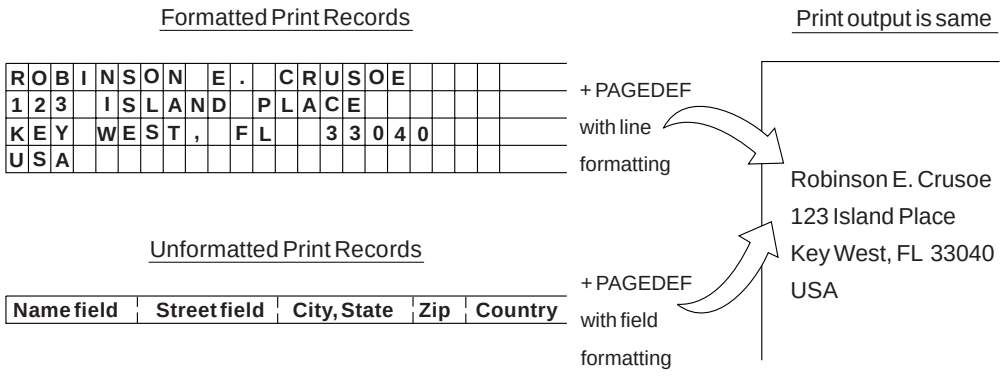


Figure 2. Formatted / Unformatted Print Records

The technique of mapping the unformatted data to locations on the output pages is known as field processing or record processing and is available through use of page-definition controls. Field processing is explained in detail in "Processing Fields" on page 44.

Traditional Line Data

Traditional line data is data formatted for printing on a line printer. Fully formatted line data can be printed on a line printer without a page definition, however all line data needs a page definition to be printed on a page printer.

A traditional line data record can contain a 1-byte carriage control character and a 1-byte table reference character followed by the data to be printed. (With a line printer, the maximum number of data bytes in a single input record is 208. With a page printer, the maximum number is 32,768 bytes). Refer to "Chapter 3. Using Page Definition Commands for Traditional Line Data" on page 33 for additional information on using traditional line data.

Record Format Line Data

The *record formatting function* allows an application to specify a format identifier (record id) with each set of data fields (data record). The format identifier references a specific layout format in a Page Definition (Pagedef). At print time, each layout format (referenced by a record id in a data record) is retrieved from the Pagedef and used to position and format the associated data records/fields on the output page. The Pagedef can contain any number of layout formats. The application can use a Pagedef layout format to either insert an end of page when a specified last line point is exceeded on the output page or to force an end of page. Refer to "Chapter 4. Using Page Definition Commands for Record Format Line Data" on page 57 on using record format line data.

```
statmid    Chubby Checker     123 Redlight Lane  Twistnshout   MA 02345
ckheader
ckdata     352          01/04/90  $ 321.50   WalMart
ckdata     353          01/05/90  $ 100.00   Sears
ckdata     354          01/10/90  $ 122.30   Boulder Bookstore
ckdata     355          01/11/90  $  59.95   Kathy's Pretty Things
ckdata     356          01/15/90  $ 852.33   Pirie Racing Enterprises
ckdata     357          01/30/90  $ 500.35   Rockley's Music Center
Ckend
```

Figure 3. Example of Record Format Line Data

Mixed-Data Files

Mixed-data files consist of MO:DCA-P data and line data or unformatted ASCII data. Such files may or may not specify the beginning and ending of pages and may or may not contain page addresses and data controls for page printing. The line-data portion of such files must be formatted for page printers by page-definition controls.

MO:DCA-P Data Files

MO:DCA-P data files are formed into pages before print server receives them. These files already contain the imbedded controls for printing on page printers. They contain such things as page addresses and data controls for page printing functions.

Note: Refer to *Mixed Object Document Content Architecture Reference* (SC31-6802) and *Advanced Function Presentation Programming Guide and Line Data Reference* (S544-3884) for more information about MO:DCA-P data. User application programs can also generate MO:DCA-P data. In OS/400, MO:DCA-P print files are created automatically when DEVTYPE=AFPDS in the Printer File.

Unformatted ASCII Files

Unformatted ASCII files consist of ASCII data with no formatting controls (escape sequences) in the data.

The technique of mapping the unformatted ASCII data to locations on the output pages is known as field processing or record processing and is available through use of page-definition controls. Field processing is explained in detail in "Processing Fields" on page 44.

Unformatted ASCII data differs from unformatted EBCDIC data in that ASCII data is what is generally created on a personal computer or workstation, while EBCDIC data is what is generally created on a mainframe host, such as OS/390, VM, or VSE, or on OS/400.

PPFA Concepts

The concepts of physical page, logical page, and subpage are basic to understanding form-definition and page-definition controls.

Physical Page

A *physical page* is the sheet of paper or other medium (a sheet of labels, for instance) that moves through the printer.

Logical Page

A *logical page* is the area you define in a PPFA command stream as the space on the physical page where data is printed. The logical page is positioned in relation to the *media origin*. For more information about the media origin of your printer, refer to your printer documentation or the *Advanced Function Presentation: Printer Information*. The positioning of the logical page on the sheet of paper is described in "Positioning a Logical Page on a Sheet" on page 19.

An N_UP command enables you to place one, two, three, or four logical pages on a single sheet. This is in contrast to multiple up, which enables you to place subpages on one logical page.

Subpage

A *subpage* is a part of a logical page on which line data may be placed. Subpages are used only with conditional processing. Multiple-up printing can be done with or without subpages being defined. In the page definition, multiple subpages can be placed on the physical page based on changes in the print data. A good example of this is the use of *multiple-up* printing, which is printing two or four pages on a single side of a sheet. For more information, see "Subpage Description and Processing" on page 107.

PPFA Basic Terms

The following terms have meanings that are special to PPFA:

- Printline
- Layout
- Direction
- Rotation
- Presentation
- N_UP partitions
- Modifications

Printline

Printline is a single line of text, and is the traditional command that is synonymous with the record formatting Layout command. In the formatting of line data and unformatted ASCII, a printline is normally the output generated by one record in the print file. However, printlines and print records are not the same.

PRINTLINE commands in the PPFA page definition define the number and position of printlines on a page. Each record in the print file is written to a single

printline on a page. Usually, one print record is written to each printline. However, control information in the print data can specify two or more print records be written to the same printline, providing overprinting. Controls also can specify that print records skip printlines. For example, a print record may skip the remaining printlines on a page and print instead on the first printline of a new page.

Layout

Layout specifies a single line of text, and is the record formatting command that is synonymous with the traditional Printline command. In the formatting of line data and unformatted ASCII, a layout is normally the output generated by one record in the print file. However, layouts and print records are not the same.

LAYOUT commands in the PPFA page definition define the number and position of layouts on a page. Each record in the print file is written to a single layout on a page. Usually, one print record is written to each layout. However, control information in the print data can specify two or more print records be written to the same layout, providing overprinting. Controls also can specify that print records skip layouts. For example, a print record may skip the remaining layouts on a page and print instead on the first layout of a new page.

Direction

Text can be printed in four print directions. A print direction is a combination of both inline and baseline directions. For each of the directions, characters can be printed in four rotations.

The line direction is the direction in which successive characters are added to a line of text. The four line directions are:

ACROSS = Text characters are placed in a line from left to right across the page.
DOWN = Text characters are placed in a line from top to bottom down the page.
BACK = Text characters are placed in a line from right to left across the page.
UP = Text characters are placed in a line from bottom to top up the page.

The baseline direction is the direction in which successive lines of text are added to a page. The four character rotations, measured clockwise around each inline direction, for each line direction are:

0°
90°
180°
270°

For example, the text in this paragraph is printed ACROSS the page, and its rotation is 0°.

Figure 4 on page 10 shows the four possible directions. For information about the combinations supported by the printer you are using, refer to *Advanced Function Presentation: Printer Information*.

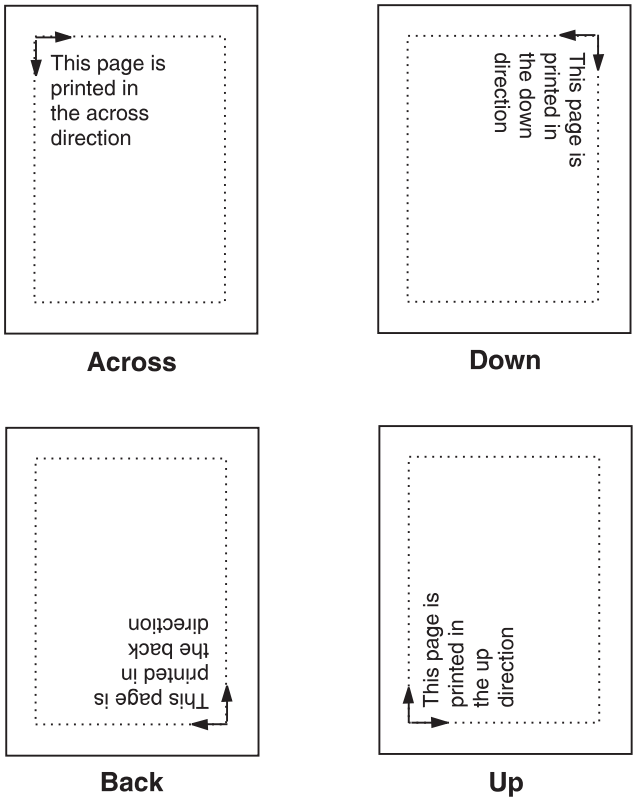


Figure 4. Baseline Direction and Inline Direction

Rotation

Individual characters can be *rotated*. Character rotation can be 0°, 90°, 180°, or 270° relative to the inline direction of the printline or field.

Note: On the 3800 printers only, character rotation differs between *bounded-box fonts* and *unbounded-box fonts*. Bounded-box fonts rotate the fonts; unbounded-box fonts are rotated by selecting the correct font.

Presentation

Presentation describes the shape of the page as it is viewed by the reader. Figure 5 on page 11 shows an example of how text is presented (positioned) on the page. There are two page presentations—*portrait* and *landscape*.

Portrait

Is designed to be viewed with the short side at the top of the page.

Landscape

Is designed to be viewed with the long side at the top of the page.

Document A - Portrait Presentation

Document B - Landscape Presentation

Figure 5. Portrait and Landscape Presentations

N_UP Partitions

Some printers allow the physical sheet of paper to be divided into equal-sized partitions. For two or three partitions, each sheet is divided along one or two lines equally spaced along the longer side of the sheet. The printer will position a logical page of print data in each partition. This enables printing multiple logical pages with different formats and modifications on a single sheet of paper.

The size and arrangement of the partitions on the sheet depends on the number of partitions and the shape and size of the paper. For two or three partitions, each sheet is divided at two or three points equally spaced along the longer side of the sheet. For four partitions, each sheet is equally divided both vertically and horizontally. See “Chapter 7. N_UP Printing” on page 127 for more information.

Modifications

Modifications are sets of form definition controls that apply to one page of a data file. With these controls, you can:

- Define the type of duplex printing to be done
- Define one, two, three, or four partitions for N_UP
- Select an overlay
- Suppress the appearance of a field
- Select the forms flash option (only for the 3800 printer)
- Specify the number of copies for a set of modifications
- Specify post-printing processing options

You can specify different sets of modifications for the same page of data in one form definition, and therefore in one print job, by a series of SUBGROUP commands. For example, a form definition with two SUBGROUP commands is said to have two sets of modifications. The same page of data is printed for each set of modifications, resulting in a slightly different output for each printing.

Definitions of Command, Subcommand, and Parameter

Commands, subcommands, and parameters are terms used throughout this publication to refer to the contents of PPFA control statements. Chapter 9. Form Definition Command Reference and Chapter 10. Page Definition Command Reference (Traditional) describe these commands with all their applicable subcommands.

Commands

Commands are the major controls composing form definitions and page definitions.

Subcommands

Subcommands are used to further define commands. The absence of subcommands means that the default values specified with those subcommands are used. Three command terms also appear as subcommand terms—FONT, OVERLAY, and SUPPRESSION. These subcommand terms further define other commands.

Parameters

You can specify *parameters* with subcommands or accept the defaults; valid entries and their defaults are shown in the command reference chapters.

Basic Controls in Traditional Line Data

The following line-printer controls may be included in a line data or unformatted ASCII file and can be used by a page definition to enable AFP functions:

- Carriage control characters
- Table-reference characters
- Record Ids

Carriage Control Characters (CC)

Carriage control characters, which control line skipping, line spacing, and page ejection on line printers, are fields within line-data and unformatted-ASCII records. They are compatible with page printers when page definitions format the printed data. In page definitions, you can specify CHANNEL subcommands that correspond to carriage control characters corresponding to channels 1 through 12 in the data. When you do so, the carriage control characters operate just as they do in a line-printer environment.

Note: ASCII ANSI, ANSI, and EBCDIC (machine) handle carriage control characters differently. See the **SPACE_THEN_PRINT** subcommand listed in “Subcommands” on page 197 for more information.

Table-Reference Characters (TRC)

Table-reference characters (TRCs) control font selection in line-data and unformatted-ASCII output. Page definitions can be used to map table-reference characters to AFP fonts for use with page printers.

Record Id

Record ids are only used with the record formatting function.

Basic Controls in Record Format Line Data

Record format line data is a new form of line data that is supported by print server and formatted by a page definition. With this format, each data record contains a 10-byte record identifier that selects the record descriptor (RCD) in a record format page definition used to format the line data. This RCD might contain a carriage control (CC) byte.

- Carriage control characters
- Table-reference characters (not applicable in record format)

Carriage Control Characters (CC)

The CC byte is required when record format data is mixed with MO:DCA-P data, but is ignored. The CC byte is optional for record format line data at all other times, however if you enter it, you must inform print server that it is there.

Many functions used in the line descriptor (LND) to format traditional line data are used in RCD to format record format line data. Others, such as header and trailer processing, are unique to RCDs.

Traditional line data is similar to record format line data in that neither is formatted into pages. However, traditional line data can be printed on line printers while record format line data cannot. For more information, refer to “Chapter 4. Using Page Definition Commands for Record Format Line Data” on page 57.

Note: ASCII ANSI, ANSI, and EBCDIC (machine) handle carriage control characters differently. See the **SPACE_THEN_PRINT** subcommand listed in “Subcommands” on page 197 for more information.

Table-Reference Characters (TRC)

Table-reference characters (TRCs) cannot be used in record formatted line data.

Record Id

Record ids are only used with the record formatting function. They reside in the first 10 characters of each line data record, and control the layout type that is selected for each given record. These 10 characters are reserved for record ids and are not included as part of a defined field or conditional area.

Structured Fields in Line Data

To make use of the full function of page definitions and form definitions, MO:DCA-P structured fields may be required in the users data. The following MO:DCA-P structured fields can be included in a line-data or unformatted ASCII file (typically AIX) to activate AFP functions:

- Invoke Data Map
- Invoke Medium Map
- Include Page Segment
- Include Page Overlay
- Include Object
- Include Presentation Text (PTX)
- Include No Operation (NOP)

Invoke Data Map

Add the Invoke Data Map structured field to the line-data or unformatted ASCII file at a point that requires switching from one page format to another. The term

"data map" is the name used for the term "page format" in print server publications and print server terminology.

Invoke Medium Map

Add the Invoke Medium Map structured field to the line-data or unformatted-ASCII file at a point that requires switching from one copy group to another. The term "medium map" is the name used for the term "copy group" in print server publications and print server terminology.

Include Page Segment

Position the Include Page Segment structured field within the line or unformatted ASCII data for placing the page segment on the page.

Include Page Overlay

Position the Include Page Overlay structured field within the line or unformatted ASCII data for placing the overlay anywhere on the page.

Include Object

Position the Include Object structured field for placing an object containing other object types (for example, IOCA or BCOCA) for placing the object anywhere on the page.

Presentation Text

A presentation text object can be included in line data using the Presentation Text (PTX) structured field which is a self contained object consisting of line spacing, page margin, data position and font settings. Refer to the *AFP Programming Guide and Line Data Manual* (S544–3864) and the *Presentation Text Object Content Architecture Reference* (SC31–6803) for additional information.

No Operation

A No Operation (NOP) structured field can be placed in the line data stream. This can be used to insert information, such as a comment, into the data stream.

Normal Duplex and Tumble Duplex

Some page printers can print on both sides of a sheet, which is called *duplex* printing. Duplex printing can be done in four ways:

Normal duplex
Tumble duplex
Rotated normal duplex
Rotated tumble duplex

In normal duplex, both sides have the same orientation, as in most books. In tumble duplex, the back of each page is upside down with respect to the front of the page: the top of one side of the sheet is at the same edge as the bottom of the other side. These two types of duplex allow you to specify top binding or side binding of the printed pages.

Duplex also involves the commands RNORMAL (rotated normal) and RTUMBLE (rotated tumble), which are used with landscape-presentation pages to specify the type of duplex printing. See Figure 13 on page 26 and Figure 14 on page 27 for illustrations of duplex printing.

Part 2. Examples of Using PPFA

Chapter 2. Using Form Definition Commands

A form definition is a resource, used by print server, that specifies how the printer controls the processing of the sheets of paper. With form definitions, you can perform the tasks listed in Table 1.

Table 1. Form Definition Tasks

Tasks	Location of Example
Creating a form definition	"Commands Required to Create a Form Definition" on page 18
Positioning a logical page	"Positioning a Logical Page on a Sheet" on page 19
Specifying landscape presentation	"OFFSET Subcommand with Rotated Print Direction" on page 20
Specifying copies and electronic overlays	"Specifying Copies and Electronic Overlays" on page 20
Printing constant forms	"Printing Constant Forms" on page 22
Duplex printing in two orientations	"Duplex Printing" on page 23
Printing portrait and landscape	"Duplex Printing in Portrait and Landscape Presentations" on page 25
Specifying the page presentation on continuous-forms printers	"Specifying Page Presentation on Continuous-Forms Printers" on page 27
Migrating from 3800 printers to other IPDS printers	"3800 Coexistence and Migration" on page 30

Copy Groups and Subgroups

A single form definition can contain several subsets of page controls, called *copy groups*. Copy groups define each physical page in the file. When you are printing jobs in duplex, the copy group defines both sides of the physical paper. Copy groups, in turn, can contain up to 127 *subgroups*, each of which creates a different set of modifications for the same page of data.

A series of copy groups can be used where either the data or the printing requirements call for a variety of page control schemes. Part of the file can be printed from one (bin) paper source and part from another. Part can be printed duplex; part can be printed simplex. Duplex commands can be specified for a printer that does not support this function. This command treats the two adjacent pages as duplexed. A variety of controls can be contained in one form definition having several copy groups.

You can control the following options within a copy group:

- Position of the logical page on a sheet of paper
- Duplex printing
- Type of cut-sheet paper to be printed on (by choosing between paper input sources in page printers that have more than one paper source)
- Offset stacking or copy marking of parts of a print job in the output stacker
- Printing one, two, three, or four logical pages on a single side of a sheet

- Vendor-attached devices for post-processing functions to be performed on the sheet
- Print-quality level

To access a new copy group within a form definition you can:

- Add to your data file an Invoke Medium Map structured field immediately before the page of data that requires the new copy group.
- Use a page definition that specifies conditional processing. When you access a new copy group, printing begins on the next physical sheet of paper.

For more information on the Invoke Medium Map structured field, refer to *Mixed Object Document Content Architecture Reference.*

Subgroups allow the same page of data within a file to be printed more than once, using different sets of modifications each time the page is printed. One example is the printing of an invoice and a packing list from the same records in a data file.

The following modifications to the page of data can be specified in a subgroup:

- Selection of suppressed fields for the page
- Selection of overlays used with the page
- Selection of forms flash with the page (only on the 3800 printer)
- Selection of the modification for front, back, or both sides of a sheet
- Selection of the number of copies of the subgroup to print
- Selection of the input bin

Commands Required to Create a Form Definition

The following simplified command stream shows the proper nesting of commands and the sequence in which the commands must be entered when you are creating a form definition:

```
[SETUNITS ]
FORMDEF
[SUPPRESSION ...]
[COPYGROUP ]
  [OVERLAY ...]
  [SUBGROUP ...]
[COPYGROUP ]
  [OVERLAY ...]
  [SUBGROUP ...]
```

Notes:

1. If the form definition has only one copy group, the COPYGROUP command can be omitted. The OVERLAY command then follows any SUPPRESSION command.
2. Indentations are used to improve readability.
3. Complete definitions of commands are in "Chapter 9. Form Definition Command Reference" on page 155.

Command Nesting Rules

1. SUPPRESSION commands must be specified immediately after FORMDEF commands.
2. SUBGROUP commands are specified under their associated COPYGROUP command or under the FORMDEF command.
3. OVERLAY commands are specified immediately after COPYGROUP commands.

4. The first COPYGROUP command can be omitted in a form definition if the form definition has only one copy group, and if it contains no OVERLAY commands.
5. A SETUNITS command can be placed anywhere in the PPFA command stream and is in effect until another SETUNITS command is encountered.
6. More than one of each command can appear under one form definition.

Positioning a Logical Page on a Sheet

The example in this section shows how the OFFSET subcommand is used to position the logical page on the physical sheet. A logical page is the area on a sheet of paper where all printing occurs. You establish the *logical page origin*, the point nearest the media origin, with the OFFSET subcommand. The OFFSET subcommand requires two coordinates and may have four. The first *x* and *y* coordinate defines the position on the front of the sheet, and the second *x* and *y* coordinate defines the position on the back of the sheet. A sample form definition that specifies the logical page position for a simplex sheet is:

```
FORMDEF ABCD
        OFFSET 1 IN 1 IN ;
```

Note: The 1 IN 1 IN is an abbreviation for 1 INCH 1 INCH. PPFA supports a number of different units of measurement formats. See "Units of Measurement" on page 152 for all the different formats.

The example places the logical page origin one inch to the right of and one inch down from the media origin.

Figure 6 shows the meaning of the *x* and *y* coordinates. In writing an OFFSET subcommand, the first parameter specifies *x*; the second parameter specifies *y*. If the *x* and *y* are repeated for the offset of the back side of the physical page, the same applies. The *x* defines the horizontal offset; the *y* defines the vertical offset. In this example, the logical page direction is ACROSS. The arrows within the logical page indicate the inline direction for text on the page. The lines of text are added according to the baseline direction.

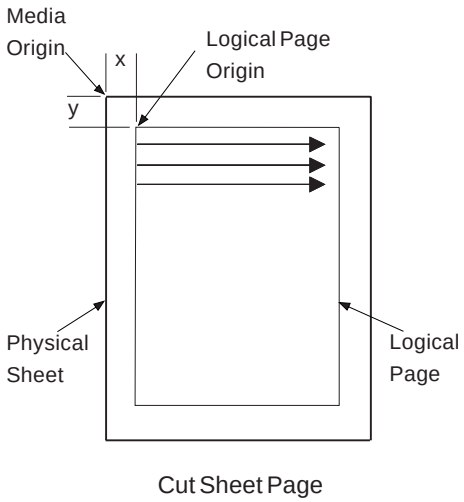


Figure 6. Origin of Logical Page

Figure 7 shows the meaning of *x* and *y* in a logical page specification for a 3900 sheet. The 3900 sheet does not have an unprintable area, but FORMDEFs supplied with print server have a 1/6 inch offset.

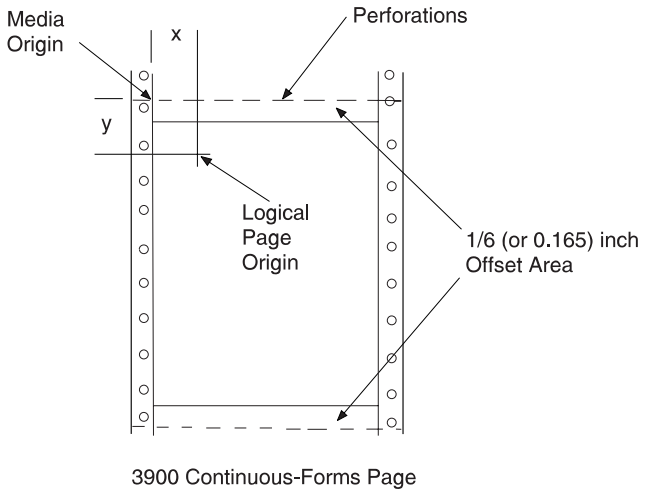


Figure 7. Origin of a Logical Page on a 3900 Sheet

OFFSET Subcommand with Rotated Print Direction

Figure 8 shows that the media origins and logical page origins do not change when the print direction of the page changes, although the way you view the page does change. The arrows within the logical page show the DOWN print direction—producing landscape page presentation.

Be careful to coordinate form definitions and page definitions when you change between portrait and landscape presentations.

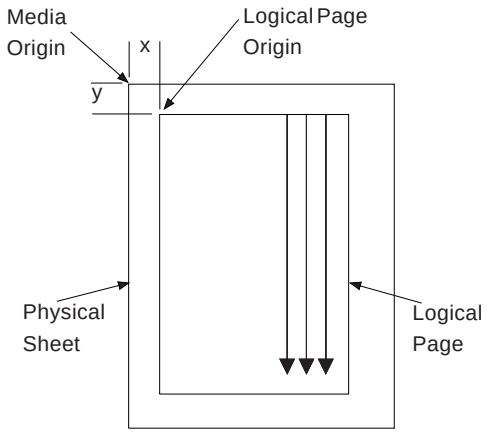


Figure 8. The Meaning of OFFSET Parameters within a Landscape Page

Specifying Copies and Electronic Overlays

This example shows how to specify different electronic overlays in different subgroups. The electronic overlays you specify are created separately, using a program such as IBM Overlay Generation Language/370 or the AFP Utilities/400, and are stored as resources in the overlay library. No positioning controls are needed in the form definition with an overlay; the overlays are merely named. The

overlay contains its own positioning data relative to the physical sheet. A form definition containing two overlays might look like this:

```
FORMDEF SLSCOM ;
  COPYGROUP SLSCOM ;
    OVERLAY SLSRPT  M1001 ;   /*LOCAL NAME AND USER-ACCESS NAME*/
    OVERLAY M1002 ;           /*USER-ACCESS NAME ONLY          */
    SUBGROUP COPIES 2
             OVERLAY SLSRPT ;
    SUBGROUP COPIES 3
             OVERLAY M1002 ;
```

The steps to write this form definition are:

1. Create a copy group.
   a. Write a COPYGROUP command.
   b. Write an OVERLAY command for each overlay.
2. Create two subgroups by writing two SUBGROUP commands. Each subgroup contains an OVERLAY subcommand naming one of the selected overlays.

   Note: The overlays must be named in each copy group.

Overlay Names

To identify overlays by name, you must be aware of the three possible names for an overlay: a local name (SLSRPT) and two system names (M1001, O1M1001). The *local name* is used only within the PPFA command stream; its use is optional. An example of this is SLSRPT in the first OVERLAY command of the previous sample command stream.

The *system name* identifies an overlay in the library. It has two forms: the *user-access name* (M1001 in the sample set of commands) and the *library-resource name*. Of these, you use only the user-access name. PPFA automatically adds the O1 overlay prefix to the user-access name, which identifies the resource in the library. An overlay referenced through a form definition built with PPFA, therefore, must begin with the O1 prefix. An example of the result is O1M1001, the library-resource name.

You can make up your own local name for an overlay. However, the local name must be used in the OVERLAY subcommand in the subgroup if it is used in an OVERLAY command for the copy group. If it is not, the subgroup must specify the user-access name, as has been done for overlay M1002 in the example.

This example, specifying copies and electronic overlays, also specifies the number of copies of each subgroup. More than one copy of printed output can be requested by placing the COPIES subcommand and the number of copies of the subgroup desired in the SUBGROUP command. This example specifies that two copies of the first subgroup and three copies of the second subgroup are to be printed. See Figure 9 on page 22, which shows the result of printing a job that includes overlays as specified in the sample command stream at the beginning of this example.

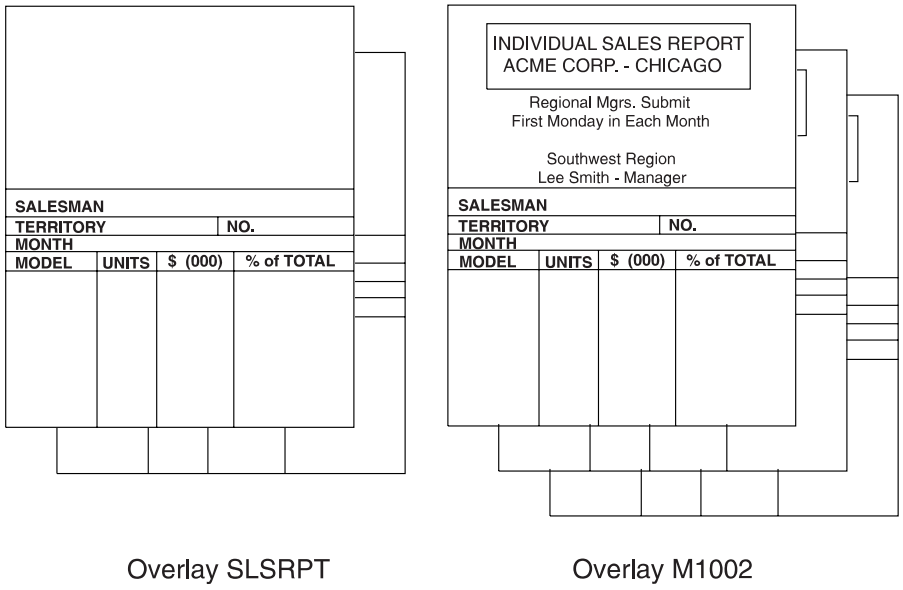


Figure 9. Two Electronic Overlays Incorporated into Two Subgroups

Printing Constant Forms

This example shows how to specify the constant-forms function using the CONSTANT command. The constant-forms function allows you to print overlays or a forms flash on blank pages without adding blank pages to your print job. Instead, the CONSTANT command generates blank pages on which to print the requested overlays and forms flash. These pages are called *constant forms* because no variable data from the print file is printed on the pages.

You specify the CONSTANT command for an entire copy group; you identify the overlays and forms flash in the subgroups of the copy groups.

The sample form definition XMPXXX shown below specifies that overlay XMP be printed on the back of each sheet with no variable data from the print job. The data from the print file is printed only on the front side of each sheet.

```
FORMDEF XMPXXX
        REPLACE YES
        DUPLEX NORMAL ;
  COPYGROUP XMPXXY
            CONSTANT BACK ;
    OVERLAY XMP;
    SUBGROUP  FRONT ;
    SUBGROUP  BACK
            OVERLAY XMP;

PAGEDEF XMPXXX
        REPLACE YES ;
  FONT  NORMALFONT  GT10 ;
  PAGEFORMAT XMPXXX  ;
    PRINTLINE CHANNEL 1 REPEAT 20
            POSITION 1 1 ;
```

The steps to write this form definition are:

1. Create a copy group.

a. Specify duplex printing.
b. Specify printing of a constant form as the back side of each sheet.
c. Write an OVERLAY command.

2. Create two subgroups by writing two SUBGROUP commands. The subgroup for the back side specifies the overlay to be printed.

Note: If you do not specify an overlay in the subgroup for the back, the back side of each sheet will be blank.

Duplex Printing

Printing on both sides of a sheet (duplex printing) can be done in two ways: by the use of the FRONT and BACK subcommand combination or by the use of the BOTH subcommand. If FRONT and BACK are chosen, the number of copies requested for each must be the same.

To demonstrate some of the functions available for duplex printing, assume you want to print a six-page data file (a simplified version is shown in Figure 10).

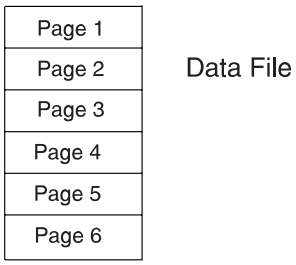


Figure 10. Six-Page Formatted Data File

Assume, too, that the file is already composed and formatted, so only a form definition is needed. The first form definition follows:

```
FORMDEF ABCD
        DUPLEX NORMAL ;
  OVERLAY AB ;
  SUBGROUP FRONT
           OVERLAY AB ;
  SUBGROUP BACK ;
```

In this command stream, form definition ABCD contains two subgroups, one specified with a FRONT subcommand and the other with a BACK subcommand.

By including a pair of FRONT and BACK subcommands within the copy group, you can specify that the front and back of printed sheets are to be controlled by different subgroups. The purpose of this is to allow modifications (overlays or suppressions, for example) to be separately specified for the front and back of sheets. Figure 11 on page 24 shows the result of using this control where the front sheets have a header (OVERLAY AB) that the backs do not have.

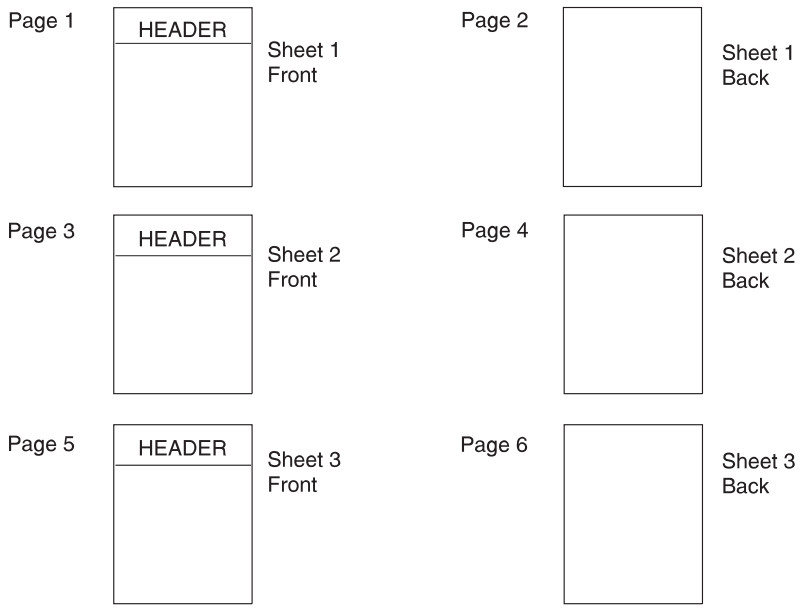


Figure 11. Result of Using a Pair of FRONT and BACK Subgroups

The rules of the FRONT and BACK subcommands are:

- FRONT and BACK subgroups must be specified in pairs.
- Subgroups specifying FRONT must always immediately precede subgroups specifying BACK.
- FRONT and BACK subgroups must agree in the number of copies.

The BOTH subcommand also can be used with a form definition or a copy group that specifies duplex printing. An example of this type of form definition is:

```
FORMDEF EFGH
        DUPLEX NORMAL ;
  SUBGROUP BOTH
           COPIES 2 ;
```

The form definition EFGH contains only one SUBGROUP command.

Notes:

1. The copy group actually contains the subgroup, but if a form definition contains only one copy group, the copy group need not be specified.
2. With the BOTH subcommand, you specify only one subgroup: both sides of all sheets have the same modifications.
3. The above form definition does *not* put the same data on the front and back of the same sheet. Internally to PPFA, a single BOTH subgroup actually produces two subgroups. As a result, two pages of data (one for each internal subgroup) are processed before copy number 2 is made. For more information about this topic, see “SUBGROUP Command” on page 190.

Figure 12 on page 25 shows a sample print resulting from using the FORMDEF EFGH specifying BOTH to control the printing of the six-page (2 copies) data file.

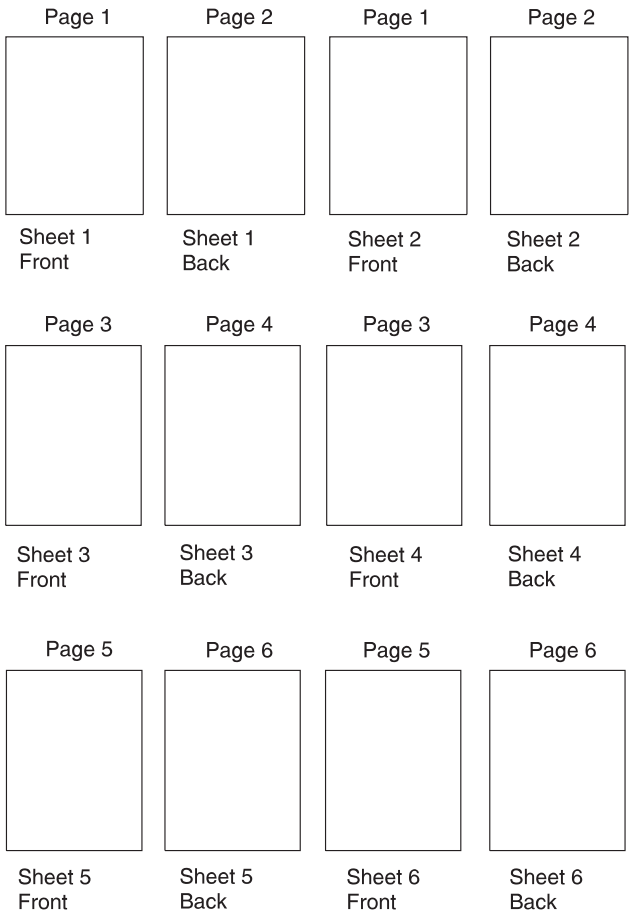


Figure 12. Form Definition EFGH Using DUPLEX with BOTH

Duplex Printing in Portrait and Landscape Presentations

Duplex printing with PPFA and print server printers offers several other options. This example shows the combination of portrait and landscape presentations with normal and tumble duplex printing.

Note: The terms normal, tumble, portrait, and landscape are used in this example. They are explained in this chapter and in the Glossary.

NORMAL and TUMBLE are parameters of a DUPLEX subcommand. For example, a form definition specifying DUPLEX NORMAL could be written this way:

```
FORMDEF ABCD ;
  COPYGROUP ABCD
            DUPLEX NORMAL ;
    SUBGROUP BOTH
             COPIES 1 ;
```

Document A in Figure 13 on page 26 shows the result of a DUPLEX NORMAL specification in the portrait presentation. Document D shows the result of the same form definition when a landscape presentation is specified. The printout in landscape presentation is really in a tumble-duplex format, having the tops (of the front side) and the bottoms (of the back side) of the logical pages toward the same edge of the sheet.

Although tumble duplex can be specified in this manner for landscape pages, another parameter, RTUMBLE (rotated tumble), exists to make the form definition look more sensible for use in landscape print jobs. It also produces the results shown in Figure 13, depending on whether the form definition called for portrait or landscape presentation. For landscape, the form definition should be written as follows:

```
FORMDEF ABCD
   PRESENT LANDSCAPE ;
  COPYGROUP ABCD
            DUPLEX RTUMBLE ;
    SUBGROUP BOTH
            COPIES 1 ;
```

Note: The example presented is for continuous printers. You must use N_UP for cut-sheet printers. In "Chapter 9. Form Definition Command Reference" on page 155, see the PRESENT subcommand of COPYGROUP.

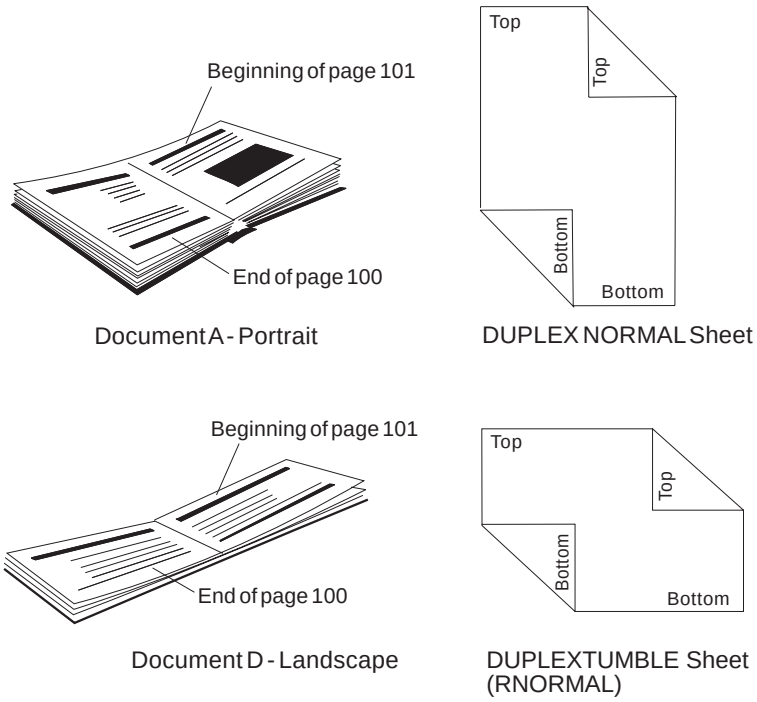


Figure 13. DUPLEX NORMAL: Portrait and Landscape Presentation

The DUPLEX NORMAL and DUPLEX RTUMBLE controls actually produce the same result on the physical page. RTUMBLE is used to maintain an association between duplex specifications and logical page print direction. The same relationship exists between the RNORMAL and the TUMBLE parameters as exists between the NORMAL and the RTUMBLE parameters; that is, within the two sets the terms are interchangeable.

For example, you could write a form definition using DUPLEX TUMBLE as follows:

```
FORMDEF DEFG ;
  COPYGROUP DEFG
            DUPLEX TUMBLE ;
    SUBGROUP BOTH
            COPIES 1 ;
```

Documents C and B in Figure 14 are the results, depending on how page definition direction is specified to achieve either a portrait page or a landscape page.

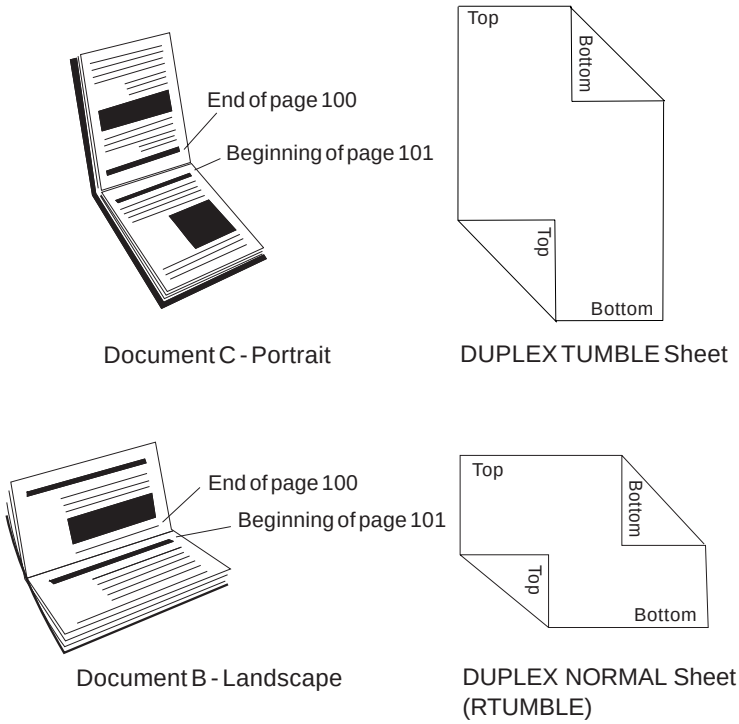


Figure 14. Result When Either TUMBLE or RNORMAL Is Specified

To help you remember, use Table 2.

Table 2. Duplex Specifications

If the form definition duplex specification is . . .	**and if the page definition direction is . . .**	**then, the duplex printing result is . . .**
DUPLEX NORMAL	ACROSS or BACK	normal duplex - portrait
DUPLEX RTUMBLE	DOWN or UP	tumble duplex - landscape
DUPLEX TUMBLE	ACROSS or BACK	tumble duplex - portrait
DUPLEX RNORMAL	DOWN or UP	normal duplex - landscape
Note: Other control combinations are not recommended.		

Specifying Page Presentation on Continuous-Forms Printers

This example shows how to specify the page presentation (portrait or landscape) on printers that use continuous-forms paper. The page presentation is specified in the form definition using the PRESENT subcommand in conjunction with the DIRECTION subcommand.

The PRESENT subcommand specifies how your pages will be presented when they are printed and has two valid values: PORTRAIT and LANDSCAPE.

The DIRECTION subcommand specifies the inline direction in which your pages have been formatted by the page definition (see “FIELD Command (Traditional)” on page 206

on page 206) or by the program formatting the data. The DIRECTION subcommand has two valid values: ACROSS and DOWN.

The conditions in which you should use these subcommands and some conditions in which they are not required are described below. For more information about how these subcommands work with data sent to specific printers, refer to the appropriate printer documentation.

In order to understand the description that follows, you must be aware of the difference between the two types of continuous forms: *narrow* and *wide.* Narrow forms are forms that have perforations on the shorter edge of the paper and tractor holes on the longer edge. Wide forms are forms that have perforations on the longer edge of the paper and tractor holes on the shorter edge. The two types of forms are illustrated in Figure 15.

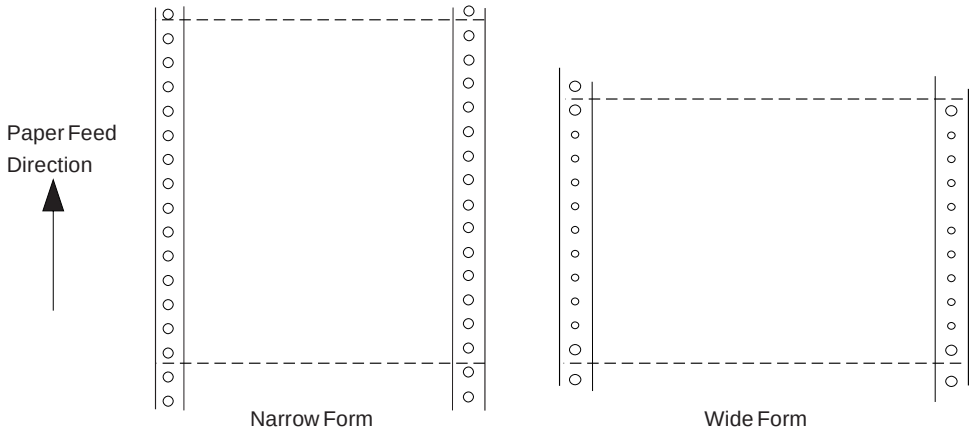


Figure 15. Narrow and Wide Continuous Forms

When to Use the PRESENT and DIRECTION Subcommands

You should use the PRESENT and DIRECTION subcommands if you are building a form definition that will be used:

- With wide forms on an IBM 3835 or 3900 Page Printer when the print data has been formatted in the DOWN print direction (see "The DOWN Direction for the 3835 or 3900 Printer" on page 29)
- When you do not know which type of form (narrow or wide) will be used on a 3835 or 3900 printer (see "The DOWN Direction for the 3835 or 3900 Printer" on page 29)
- To print data formatted for a 3800 printer on a 3835 or 3900 printer (see "3800 Coexistence and Migration" on page 30)
- To migrate data previously printed on a 3800 printer to a 3835 or 3900 printer (see "3800 Coexistence and Migration" on page 30)

Note: References to the IBM 3835 Page Printer point of origin also apply to all print server continuous-forms printers except the 3800.

When the PRESENT and DIRECTION Subcommands Are Not Required

You do not need to use the PRESENT and DIRECTION subcommands if you are building a form definition that will be used:

- With cut-sheet printers only

- With narrow forms only
- With the 3800 printer only
- With print data that has been formatted in the BACK direction by the page definition or the program formatting the data

The DOWN Direction for the 3835 or 3900 Printer

If your data has been formatted in the DOWN print direction for landscape page presentation and is to be printed on wide forms on a 3835 or 3900 printer, you must specify LANDSCAPE on the PRESENT subcommand to produce readable output.

If PRESENT LANDSCAPE and DIRECTION DOWN are not specified on the FORMDEF command, the data will be printed in the landscape presentation; however, the data will be upside down, as shown in Figure 16. The data is upside down in this case because the media origin for the 3835 or 3900 printer is located on the same corner of the form, regardless of whether a narrow or wide form is being used (see Figure 16).

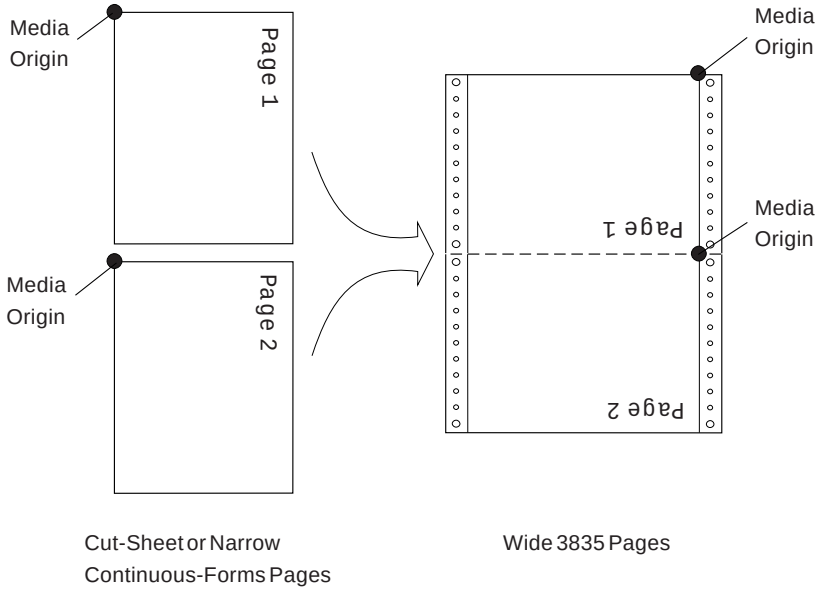


Figure 16. The Results of Not Specifying PRESENT LANDSCAPE and DIRECTION DOWN on a 3835 or 3900 Printer

If PRESENT LANDSCAPE and DIRECTION DOWN are specified on the FORMDEF command, the data will be printed as shown in Figure 17. In this example, line data is formatted using a page definition.

PRESENT LANDSCAPE and DIRECTION DOWN can also be specified for data formatted in the DOWN print direction that will be printed on narrow forms. Although PRESENT LANDSCAPE and DIRECTION DOWN do not need to be specified in this case in order to produce readable output, specifying them will enable you to use the same form definition regardless of whether the data will be printed on wide forms or narrow forms.

Note: If you are building a form definition that can be used with both wide and narrow forms, remember that the left margin as viewed by the reader will become the top margin from the printer's perspective (and vice versa). Because many printers have an unprintable area at the margins, you should

position the logical page using the OFFSET subcommand in the form definition, so data will not be placed in the unprintable area on either wide or narrow forms.

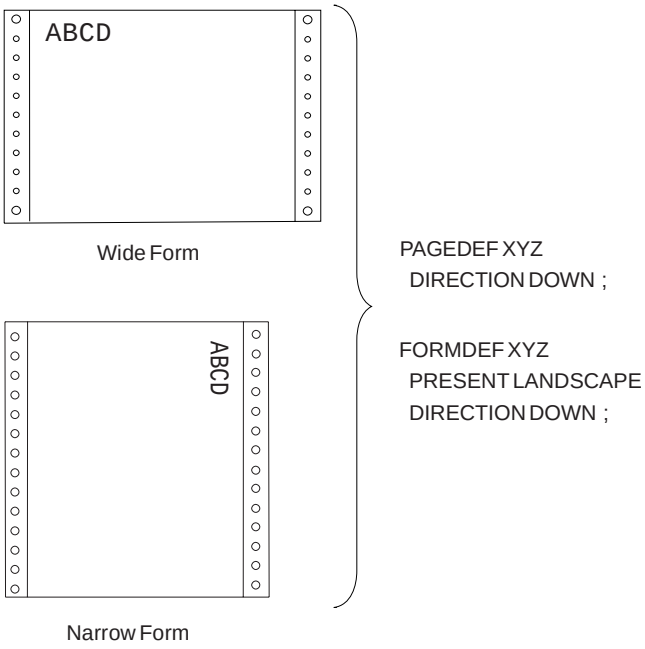


Figure 17. The Results of Specifying PRESENT LANDSCAPE and DIRECTION DOWN on a 3835 or 3900 Printer

3800 Coexistence and Migration

The PRESENT and DIRECTION subcommands should be used if you are doing either of the following:

- Building a form definition that will be used to print data formatted for the 3800 on the 3800, 3835, and the 3900 printers
- Migrating data formatted for the 3800 printer to the 3835 or 3900 printer
- Migrating data formatted for an impact printer

If the PRESENT and DIRECTION subcommands are not specified, the print data may exceed the valid printable area on the 3835 or 3900 printer, as shown in Figure 18 on page 31. The data exceeds the valid printable area in this case because the media origin of the 3835 or 3900 printer is different from the media origin of the 3800 printer.

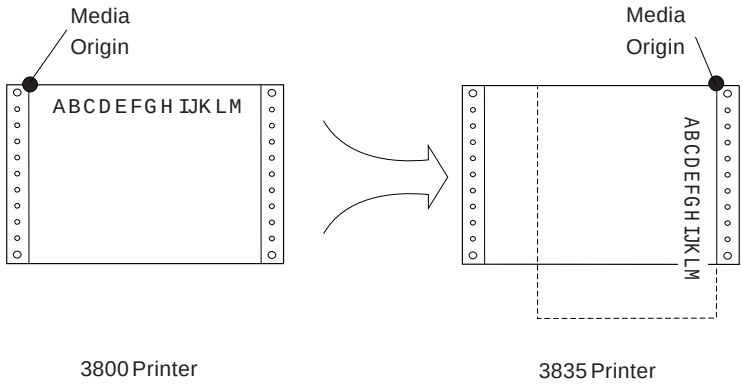


Figure 18. The Results of Not Specifying PRESENT and DIRECTION When Migrating from a 3800 to a 3835 or 3900 Printer

When you are building a form definition for migration or coexistence from the 3800 to the 3835 or 3900, code the PRESENT and DIRECTION subcommands as follows:

1. If you have a 3800 wide-forms application formatted in the ACROSS or UP direction that you want to print on wide or narrow forms on a 3835 or 3900, specify PRESENT LANDSCAPE and DIRECTION ACROSS.
2. If you have a 3800 wide-forms application formatted in the DOWN direction that you want to print on wide or narrow forms on a 3835 or 3900, specify PRESENT PORTRAIT and DIRECTION DOWN.
3. If you have a 3800 narrow-forms application formatted in the ACROSS direction that you want to print on narrow or wide forms on a 3835 or 3900, specify PRESENT PORTRAIT and DIRECTION ACROSS.
4. If you have a 3800 narrow-forms application formatted in the DOWN direction that you want to print on narrow or wide forms on a 3835 or 3900, specify PRESENT LANDSCAPE and DIRECTION DOWN.

Note: You can only specify ACROSS and DOWN on the DIRECTION subcommand when you are building a form definition for 3800 to 3835 or 3900 migration or coexistence.

Figure 19 on page 32 illustrates the output on 3800, 3835, or 3900 printers with all four combinations of PRESENT and DIRECTION. These illustrations assume that the type of forms were *not* changed from one printer type to another.

The PRESENT and DIRECTION subcommands should also be used if you want to print data formatted for the 3800 printer on a different type of forms (narrow versus wide) on the 3835 or 3900 printer. The combinations of PRESENT and DIRECTION to use when performing this type of migration are included in Line Item 2, as previously listed.

Note: When you migrate an application from one type of forms to another, remember that the top and left margins from the printer's perspective will change places. Because many printers have an unprintable area at the margins, you should position the logical page using the OFFSET subcommand in the form definition, so data will not be placed in the unprintable area on the forms you are migrating to.

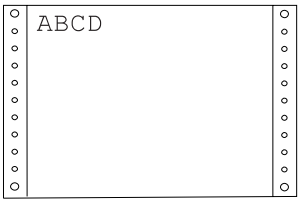
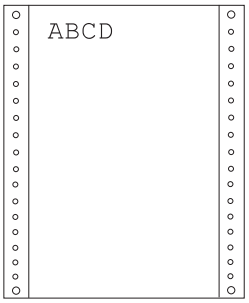
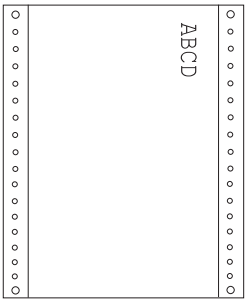
If you want output formatted for the 3800 to look like this on the 3835 or 3900:	Specify:	Or use this IBM supplied FORMDEF:
	FORMDEF ABCD PRESENT LANDSCAPE DIRECTION ACROSS ;	F10101LA
	FORMDEF ABCD PRESENT PORTRAIT DIRECTION DOWN ;	F10101PD
	FORMDEF ABCD PRESENT PORTRAIT DIRECTION ACROSS ;	F10101PA
	FORMDEF ABCD PRESENT LANDSCAPE DIRECTION DOWN ;	F10101LD

Figure 19. PRESENT/DIRECTION Combinations When Using the Same Forms Type on 3800, 3835, and 3900 Printers

Print Quality Control

If your printer has more than one print-quality selection, you can specify different levels of print quality. For more information refer to the manual for your printer.

Chapter 3. Using Page Definition Commands for Traditional Line Data

A *page definition* specifies how you want data positioned on the logical page.

A page definition is a resource used by print server that defines the rules of transforming line data and unformatted ASCII into composed pages and text controls for printing. With page definitions, you can perform the tasks listed in Table 3.

Table 3. Page Definition Tasks

Tasks	**Location of an Example**
Creating a page definition	"Page Definition Command Nesting" on page 34
Defining logical page size	"Defining Logical Page Size" on page 34
Positioning data on a logical page	"Positioning the First Line of Data" on page 35
Changing the print direction	"Changing Logical Page Print Direction" on page 37
Printing line data	"Printing Line Data on a Print Server Printer" on page 39
Processing fields	"Processing Fields" on page 44
Changing fonts	"Varying Fonts on a Page" on page 47
Printing in different directions	"Printing Lines in Two Directions on a Page" on page 50
Printing fields in two directions	"Printing Fields in Two Directions on the Same Page" on page 51
Rotating fonts	"Rotating Fonts" on page 52
Printing kanji	"Using Traditional Kanji Formatting" on page 54
Printing multiple up	"Printing Multiple-Up Pages" on page 54

Page Formats within Page Definitions

Just as form definitions can include more than one copy group, page definitions can include several *page formats*. Page formats use the same subcommands (except REPLACE) as page definitions, and if a subcommand is specified in a page format, it overrides the value specified in the page definition for the page format. A single page definition may contain multiple page formats. If pages in a file are to be formatted differently, specify more than one page format in your page definition. Within a page definition, page formats are generated in the order in which they are specified.

Using more than one page format to control different pages requires one of the following:

- Adding the Invoke Data Map structured field to the data file each time you want to change page formats
- Using conditional processing.

Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the Invoke Data Map structured field.

Page Definition Command Nesting

The following simplified command stream shows the proper nesting of commands and the order in which they must be entered when you create a page definition:

```
[SETUNITS]
PAGEDEF
  [FONT]
  [OBJECT]
  [PAGEFORMAT]
    [TRCREF]
    [OBJECT]
    [SEGMENT]
    [OVERLAY]
     PRINTLINE
      [FIELD]
      [CONDITION]
    [ENDSUBPAGE]
[SETUNITS]
```

Notes:

1. Brackets enclosing a command mean the command is optional.
2. A command and its subcommands end with a semicolon.
3. Indentations are used to improve readability.
4. Complete definitions of all commands are included in "Chapter 10. Page Definition Command Reference (Traditional)" on page 195.

Command Nesting Rules

1. FONT commands must be specified immediately after PAGEDEF commands.
2. A SETUNITS command can be specified anywhere in the PPFA command stream and is in effect until another SETUNITS command is specified.
3. OBJECT commands may appear after the FONT command, before any PAGEFORMAT command (global objects) or after a specific PAGEFORMAT command. A global object is defined for all page formats in the page definition. Otherwise the object is just defined for the pageformat in which it is specified.
4. TRCREF, SEGMENT, and OVERLAY commands must be specified under their associated PAGEFORMAT command.
5. The first PAGEFORMAT command can be omitted in a page definition, if the page definition has only one page format.
6. At least one PRINTLINE command is required.

Defining Logical Page Size

"Positioning a Logical Page on a Sheet" on page 19 shows how to establish the origin point of a logical page, relative to the media origin on a sheet of paper, using the OFFSET subcommand. The following example shows you how to establish the width and height of the logical page relative to this origin point. This example illustrates how the dimensions of a logical page are determined by form definitions and page definitions.

```
FORMDEF ABCD
        OFFSET (1)(2) ;
PAGEDEF ABCD
        WIDTH  (3)
        HEIGHT (4)  ;
        PRINTLINE  ;
```

Note: The parenthetical numbers represent dimensions. Figure 20 shows how these dimensions relate to the logical page.

Normally, all parameters consist of a number and a unit of measurement, for example, 6 IN. (See “Units of Measurement” on page 152 for information on units that are available.) Numbers can be specified with up to three decimal places. The PRINTLINE command is included because at least one is required for all page definitions; see “PRINTLINE Command (Traditional)” on page 231 for more information.

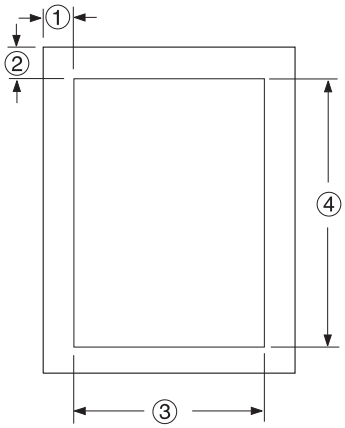


Figure 20. Logical Page Dimensions

The OFFSET subcommand (1) (2) in the sample form definition establishes the corner or origin of the logical page relative to the physical sheet. The WIDTH and HEIGHT subcommands, (3) and (4), specify the dimensions of the logical page relative to the logical page origin.

Note: Be careful not to define a logical page larger than the physical sheet. PPFA does not check the size of the physical sheet.

“Positioning the First Line of Data” shows you two ways to position the first line of data on the page.

Positioning the First Line of Data

The previous section showed you how to define the size of a logical page. The next two examples show you how to position the first line of data inside the logical page, using the LINEONE subcommand. This subcommand position is relative to the logical page origin, as shown in Figure 21 on page 36. The two coordinates, (1) and (2), of the LINEONE parameter define the starting point for the first line of text.

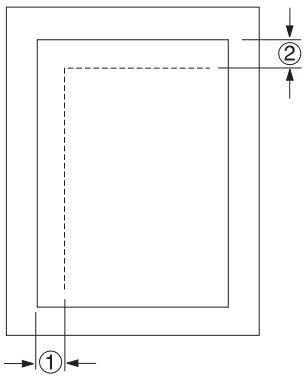


Figure 21. LINEONE Coordinates

This starting point works with the POSITION, MARGIN, and TOP subcommands (of the PRINTLINE command) to position lines of print on a page.

The defaults for LINEONE are:

x = 0,
y = 80% of one line space from the top of the logical page:
80% of 1/6 inch if lines per inch (lpi) = 6,
80% of 1/8 inch if lpi = 8,
and so on.

These defaults leave room for the character ascenders in the first line of text.

Note: PPFA subtracts one logical unit (L-unit) from the *y* value to compensate for the fact that the printer counts L-units beginning with the number 0. Therefore, if you specify the offsets to the first line in L-units (PELS is the measurement command for L-units) using the LINEONE subcommand, you must remember to subtract one L-unit from the *y* offset value. This is necessary to prevent descenders on the last printed line from dropping off the bottom of the logical page.

The following examples illustrate two methods for positioning the first line of text:

1. The position of the first line of data defaults by specifying the SETUNITS command prior to the PAGEDEF command, like this:

```
SETUNITS  1 IN 1 IN
          LINESP 8 LPI;
FORMDEF   ABCD
          OFFSET 0 .5;
PAGEDEF   ABCD
          WIDTH 7.5
          HEIGHT 10
          DIRECTION ACROSS;
  FONT GS12 GS12;
  PRINTLINE REPEAT 60
          FONT GS12
          POSITION  0 TOP;
```

   Note: It is important that the LINESP subcommand (of the SETUNITS command) must precede the PAGEDEF commands.

If the LINESP subcommand *follows* the PAGEDEF command, PPFA will then use the default LINESP value to calculate the y offset value, which is used to position the first line of print.

The default for the LINESP subcommand of the SETUNITS command is 6 lpi. If LINEONE is allowed to default, based upon the LINESP default, the LINEONE value will be 31 L-units:

LINEONE = ((240 L-units / 6 lpi) x 80%) − 1 L-unit= 31 L-units.

This value will be the vertical (y) position of the printline because TOP is specified in a later POSITION subcommand. However, this value may cause the data to exceed the bottom boundary of the logical page if the LINESP value is changed later.

2. Another way you can specify the starting position for the first print line is to specify LINEONE explicitly, like this:

```
FORMDEF   ABCD
          OFFSET 0 .5;
PAGEDEF   ABCD
          WIDTH 7.5
          HEIGHT 10
          LINEONE 0 PELS 23 PELS
          DIRECTION ACROSS;
SETUNITS  1 IN 1 IN
          LINESP 8 LPI;
  FONT GS12 GS12;
  PRINTLINE REPEAT 60
          FONT GS12
          POSITION  0 TOP;
```

In this example, the LINESP subcommand following the PAGEDEF command will not cause a data placement problem because the LINEONE command determines explicitly where the first line of text is positioned, and no default LINESP value is used:

LINEONE = [(240 L-units / 8 lpi) x 80%] − 1 L-unit= 23 L-units

If you use the LINEONE command to specify an absolute starting position for the first line, in L-units, you must remember to subtract one L-unit from that value.

Changing Logical Page Print Direction

Logical pages can have four different print directions: ACROSS, DOWN, BACK, and UP. This example shows that all four directions can be specified in relation to one offset specification:

```
FORMDEF ABCD
        OFFSET (1) (2) ;
PAGEDEF DEFG  ;
  PAGEFORMAT DEFG1
             WIDTH (3)
             HEIGHT (4)
             DIRECTION ACROSS ;
    PRINTLINE  ;
  PAGEFORMAT DEFG2
             WIDTH (3)
             HEIGHT (4)
             DIRECTION DOWN ;
    PRINTLINE  ;
```

```
PAGEFORMAT DEFG3
           WIDTH (3)
           HEIGHT (4)
           DIRECTION BACK ;
  PRINTLINE  ;
PAGEFORMAT DEFG4
           WIDTH (3)
           HEIGHT (4)
           DIRECTION UP ;
  PRINTLINE  ;
```

One page definition is used to simplify the example, yet four logical pages are specified. The PAGEFORMAT commands create subsets of page definitions for each logical page.

Note: The page formats in this example require an Invoke Data Map structured field at the place in the data file where you want to change page formats. The PRINTLINE commands are required but are not relevant in the example.

The DIRECTION subcommand with one of its four direction parameters, ACROSS, DOWN, UP, or BACK, specifies the print direction of the logical page.

Figure 22 on page 39 shows the format of each of the logical pages specified in the page definition with the direction specification of each. The pages with the ACROSS and BACK directions are in portrait presentation. The pages with the DOWN and UP directions are in landscape presentation.

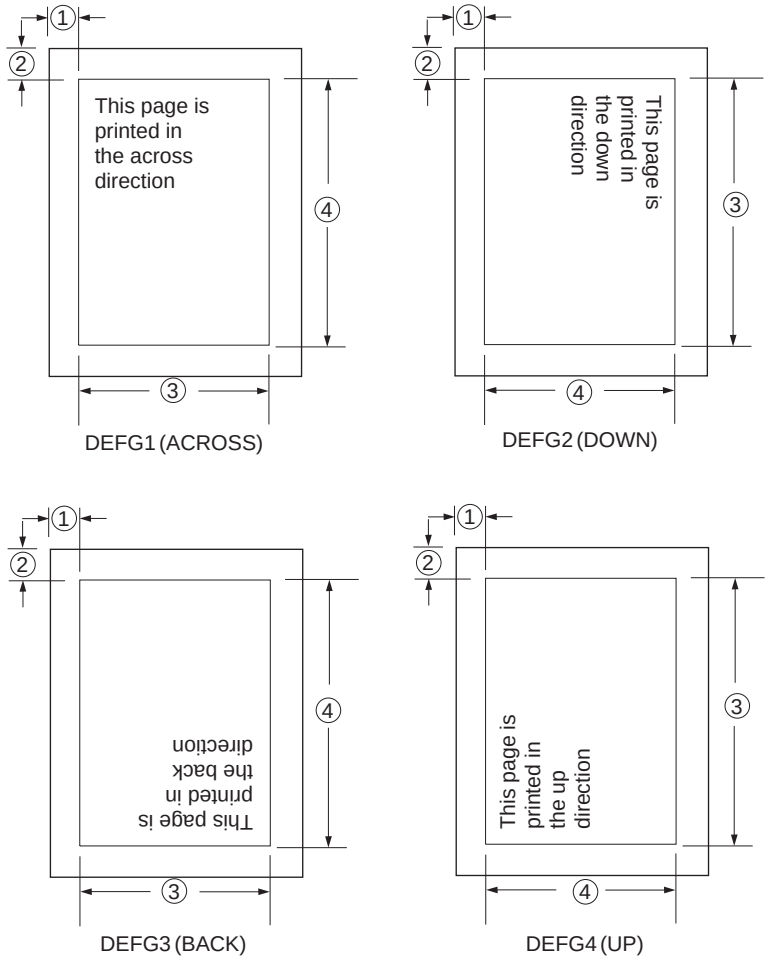


Figure 22. Logical Page Print Directions in Relation to Origin

The media origins and logical page origins do not change with the presentation of the data on the page. The OFFSET subcommand of the form definition need not change. However, the width and height dimensions do change; that is, the WIDTH subcommand always governs the horizontal (inline) dimension as you view the page, and the HEIGHT subcommand always governs the vertical (baseline) dimension whether the page is in portrait or in landscape presentation. Ensure that these specifications do not cause the logical page to cross the edge of the physical page.

However, if the DOWN direction is specified for use with the IBM 3835 or 3900 Page Printer, the PRESENT and DIRECTION subcommands may need to be specified in the form definition. See “Specifying Page Presentation on Continuous-Forms Printers” on page 27 for more information.

Printing Line Data on a Print Server Printer

This example shows how you can print a data file developed for a line printer on a page printer without altering the data. The example compares the effects of line printer controls with the corresponding controls in the PPFA commands and subcommands. PRINTLINE, LINESP, POSITION, CHANNEL, and REPEAT are

page definition controls related to the lines of text in your printout. Line printer controls examined are the forms control buffer (FCB) and carriage control characters.

As shown in Figure 23, a file consisting of 13 records is to be printed. Several different printouts of this data are formatted in the following examples. In the first two printouts, records 1–6 are printed on page 1, records 7–9 on page 2, and records 10–13 on page 3.

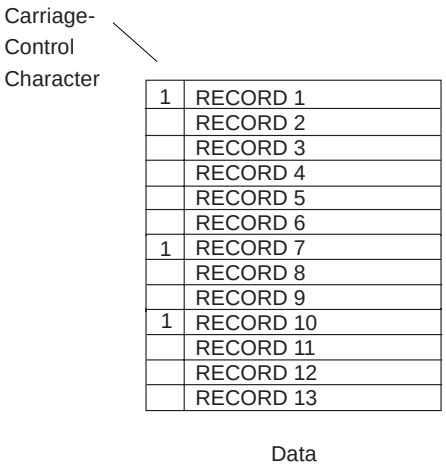


Figure 23. Line-Data File

Figure 24 on page 41 shows the formatting process used when the file is printed on a line printer. For many line printers, an FCB is used to format the output in the S/370 (OS/390, VM, VSE) environment. The sample FCB represented in Figure 24 on page 41 determines that no printed page contain more than eight lines. A page can have exactly eight lines without using carriage control characters in the data. A page may contain any number of lines fewer than eight; this is effected by placing fewer than eight records between the carriage control characters in the data. In the data file in Figure 23, fewer than eight records are, in all cases, placed between channel 1 carriage control characters. A ninth record, if encountered before a carriage control character, would cause a page eject and a return to the beginning of the FCB. The printout shown in Figure 24 on page 41 results from the data being formatted by this FCB.

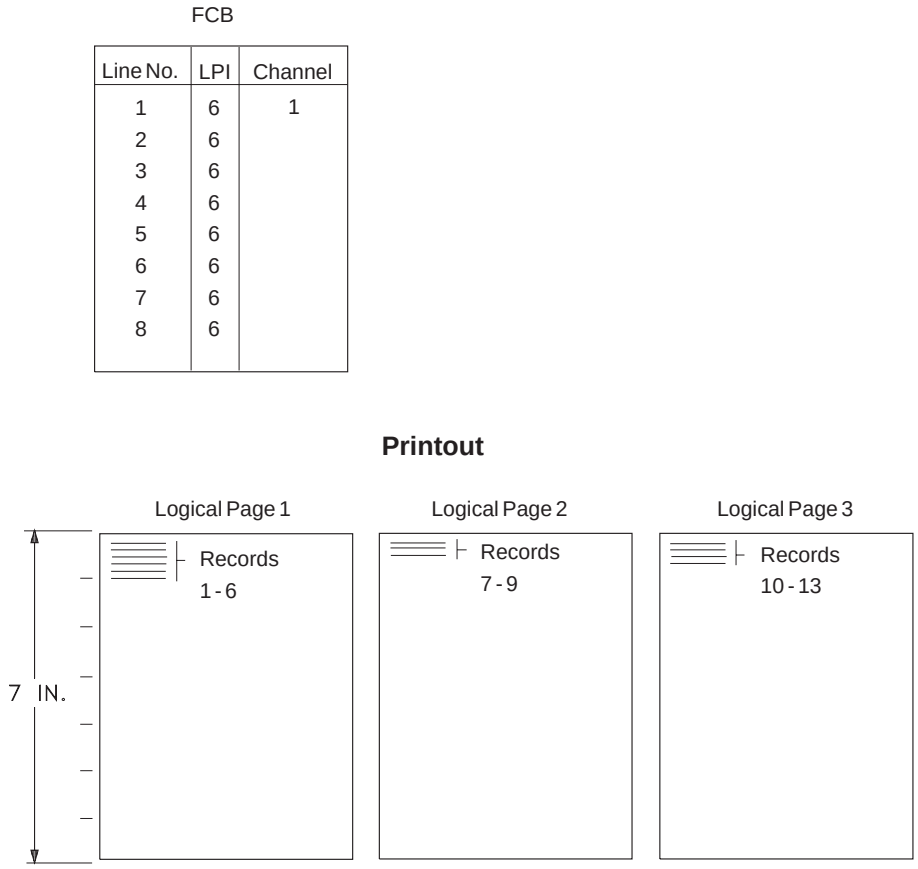


Figure 24. Data File Printed on a Line Printer

A page definition can work exactly the same way. Consider the following example:

```
SETUNITS 1 IN 1 IN
         LINESP 6 LPI  ;
PAGEDEF ABCD
        WIDTH 5
        HEIGHT 7
        LINEONE .5 .5  ;
  PRINTLINE CHANNEL 1
            POSITION MARGIN TOP
            REPEAT 8 ;
```

This command stream contains one new command (PRINTLINE) and four new subcommands (LINESP, CHANNEL, POSITION, and REPEAT) related to controlling individual lines.

- The subcommand has the same function as the LPI specifications in the FCB or in a Printer File; it defines the line density iLINESPn lines per inch.
- The PRINTLINE command contains the controls for one or more lines.
- The CHANNEL subcommand has the same function as the channel 1 control character in the FCB, causing a page eject at each channel 1 control character encountered in the data records.
- The POSITION subcommand establishes the location of the first line relative to the upper-left corner of the logical page. This example uses the MARGIN and TOP parameters; however, numeric parameters similar to those used with the OFFSET subcommand can also be used. Those values are also relative to the logical page.

- The REPEAT subcommand is a commonly used control in PPFA text formatting. It is the way you specify the total number of PRINTLINEs in a logical page.

  Note: The constraints in specifying a REPEAT value and, thereby, the number of lines per page are: the lines-per-inch specification, the height of the logical page, and the font selection. The REPEAT variable "8" is chosen to equal the maximum number of records to be printed per page. As in the line printer version, if a ninth record were encountered before a channel 1 carriage control character, a page eject would occur and the line would be printed as the first line at the top of the next page.

The result of this page definition is represented in Figure 25.

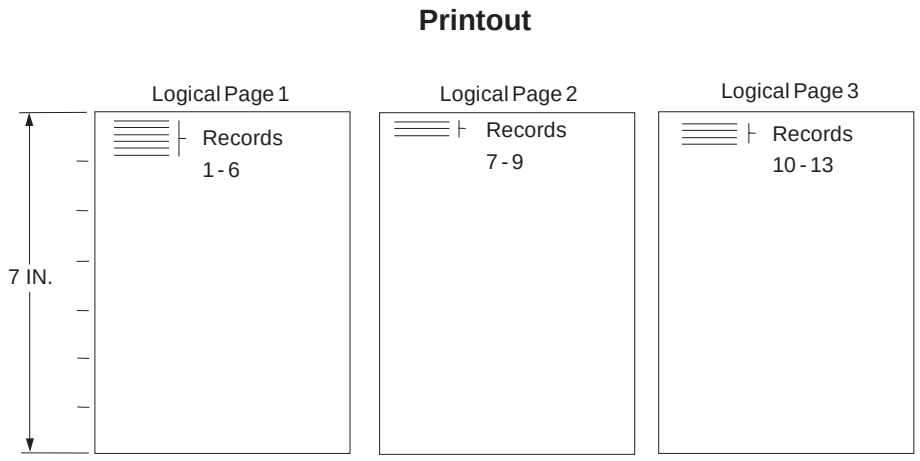


Figure 25. Printout Examples Specifying POSITION MARGIN TOP

Changing line printing specifications for the following example is shown in Figure 26.

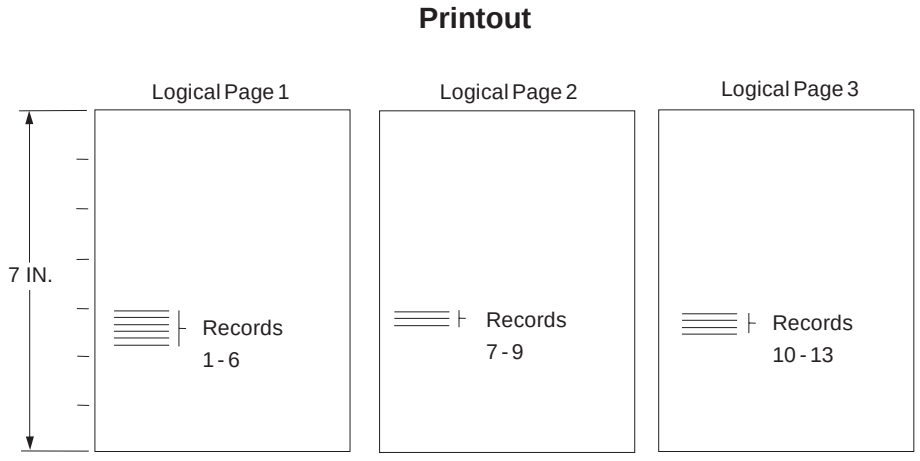


Figure 26. Printout Example Specifying POSITION MARGIN 4.1

```
SETUNITS 1 IN 1 IN
         LINESP 6 LPI ;
PAGEDEF ABCD
        WIDTH 5
```

```
       HEIGHT 7
       LINEONE .1 .1  ;
  PRINTLINE CHANNEL 1
            POSITION MARGIN 4.1
            REPEAT 8 ;
```

Observe that the second parameter of POSITION is no longer TOP; instead it is 4.1, which places the first line of text 4.1 inches down the page rather than at the top (Figure 26 on page 42).

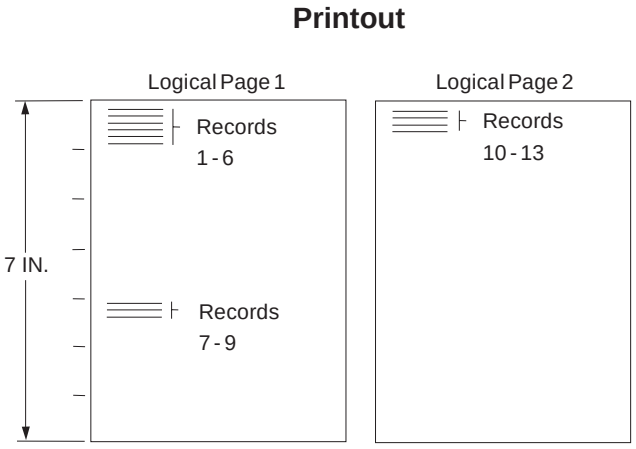


Figure 27. Printout Example Specifying POSITION MARGIN TOP and POSITION MARGIN 4.1

The following example and Figure 27 show a third version of the possible formats for the data represented in Figure 24 on page 41.

```
SETUNITS 1 IN 1 IN
         LINESP 6 LPI ;
PAGEDEF ABCD
        WIDTH 5
        HEIGHT 7
        LINEONE .1 .1 ;
  PRINTLINE CHANNEL 1
            POSITION MARGIN TOP
            REPEAT 8 ;
  PRINTLINE CHANNEL 1
            POSITION MARGIN 4.1
            REPEAT 8 ;
```

You also can skip over space using carriage control characters. This example shows how to do this by using a second PRINTLINE command to create a second starting position on the page (as shown in Figure 27). The second starting position is vertically 4.1 inches down from the top of the page; see the second POSITION subcommand. The two CHANNEL 1 subcommands take turns mapping the records governed by the successive channel 1 carriage control characters in the data to their specified positions on the page. In this case, the carriage control 1 characters cause printing to alternate between the TOP position (0.1 inch down the page) and 4.1 inches down the page.

The OS/400 Environment

This example shows how you can print a data file developed for a line printer on a page printer without altering the data, within the OS/400 environment.

If, in DDS, the following example were used:

```
Page length=66, LPI=6, OVRFLW=60 (10 inches)
  Print 36 lines at 6 LPI (6 inches)
  Print 16 lines at 4 LPI (4 inches)
```

you could get the same formatting in PPFA by coding:

```
PAGEDEF MIXLPI
    WIDTH 8 IN
    HEIGHT 10 IN
    LINEONE x Y
    SETUNITS LINESP 6 LPI;
  PRINTLINE POSITION MARGIN TOP
    REPEAT 36;
  SETUNITS 4 LPI;
  PRINTLINE REPEAT 16;
```

Processing Fields

This section describes the mapping of individual fields to the printed sheets. The technique allows you to print unformatted data according to precise specifications, and these specifications can change without affecting the data file.

The rules for field processing of data files are:

- Each record in your file must correspond to a separate PRINTLINE command because each record is mapped separately. When processing identical fields, you can define a single printline and use the REPEAT subcommand.
- Each FIELD command must follow its associated PRINTLINE command, and more than one FIELD command can be specified for a single PRINTLINE command.

For this field-processing example, the data file shown in Figure 28 on page 45 is used. Figure 29 on page 45 represents an output format that could be used to place data on a form, such as an invoice or an order. The page definition commands to print Figure 29 on page 45 are as follows:

```
PAGEDEF ABCD
        WIDTH 7 IN
        HEIGHT 8 IN ;
 PRINTLINE POSITION 1 IN 1 IN ; /*PROCESSING FOR R1              */
  FIELD START 1 LENGTH 4 ;      /*THE PRINTLINE POSITION IS      */
                                /*THE DEFAULT FOR THE FIRST FIELD*/
  FIELD START 11 LENGTH 4
        POSITION 4 IN 0 IN ;
 PRINTLINE POSITION 3 IN 4 IN ; /*PROCESSING FOR R2              */
  FIELD START 1 LENGTH 4 ;      /*DEFAULT POSITION               */
  FIELD START 6 LENGTH 4
        POSITION 0 IN 1 IN ;
  FIELD START 13  LENGTH 3
        POSITION 2 IN 3 IN ;
 PRINTLINE POSITION 1 IN 2 IN ; /*PROCESSING FOR R3              */
  FIELD START 1 LENGTH 4 ;      /*DEFAULT POSITION               */
  FIELD START 11 LENGTH 4
        POSITION 4 IN 0 IN ;
```

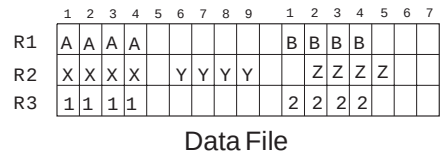


Figure 28. Unformatted Print Data File

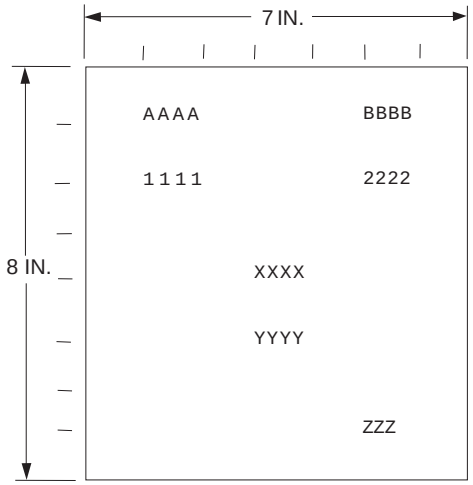


Figure 29. Data Arranged on the Printed Page

Position Subcommand as Used in this Example

The POSITION subcommand of each PRINTLINE command specifies the printline position relative to the logical page origin. The POSITION subcommands below FIELD commands specify a field position relative to the governing printline position. Following POSITION subcommands come the horizontal (*x*) then the vertical (*y*) offsets from the reference point. They are parallel in structure to the OFFSET subcommand of the form definition.

For example, the final POSITION subcommand places the final field 1 + 4 inches to the right of the left edge of the logical page, combining the *x* value of 1 in the PRINTLINE command, and the *x* value of 4 in the nested FIELD command. The 0 in the FIELD command specifies no change to the *y* value in the PRINTLINE command. Thus, the position of the final field is 5 IN (*x*), 2 IN (*y*).

Note: The first FIELD command within each PRINTLINE has no position specification, because the PRINTLINE POSITION value is the default for the first FIELD command nested under it.

Alternate controls for the *x* and *y* values of a POSITION subcommand are available. See the description of the POSITION subcommand in “FIELD Command (Traditional)” on page 206 and “PRINTLINE Command (Traditional)” on page 231.

FIELD Command as Used in this Example

In the FIELD command, the START and LENGTH parameters specify the location of the field in the record to be processed. START indicates the starting byte position, and LENGTH specifies the number of bytes in the field.

Because a field can be located independently within the data and on the printed page, more than one page definition or page format can be created for the same data file, each specifying different mapping of the data to the output pages.

Color on the IBM InfoPrint HiLite Color Post Processor

This section provides an example of the use of Highlight color. Figure 31 on page 47 shows where the text is placed on the page. The CALIBRATION setup is as follows:

Start pel	Color	Width in pels
200	high 1	288
488	high 2	288
776	high 3	192

The page definition commands to print Figure 31 on page 47 are as follows:

```
PAGEDEF XMP1
        WIDTH 7 IN
        HEIGHT 8 IN
        REPLACE YES;

 PRINTLINE
  POSITION .83 IN 1 IN ;
   FIELD START 1 LENGTH 4
    HIGHLIGHT 1;                    /*  'AAAA'                      */

  FIELD START 11 LENGTH 4
    POSITION 2.4 IN 0 IN
      HIGHLIGHT 3;                  /*  'BBBB'                      */

 PRINTLINE POSITION 2.03 IN 4 IN ;
  FIELD START 1 LENGTH 4
   HIGHLIGHT 2;                     /*  'XXXX'                      */

  FIELD START 6 LENGTH 4
   POSITION 0 IN 1 IN
    HIGHLIGHT 2;                    /*  'YYYY'                      */

  FIELD START 13  LENGTH 3
   POSITION 1.2 IN 3 IN
    HIGHLIGHT 3;                    /*  'ZZZ'                     */

 PRINTLINE POSITION .83 IN 2 IN ;
  FIELD START 1 LENGTH 4
    HIGHLIGHT 1;                    /*  '1111'                      */

  FIELD START 11 LENGTH 4
   POSITION 2.4 IN 0 IN
    HIGHLIGHT 3;                    /*  '2222'                      */
```

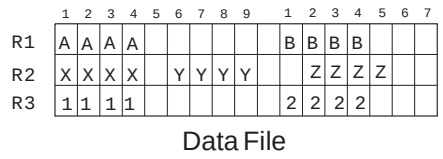


Figure 30. Unformatted Print Data File

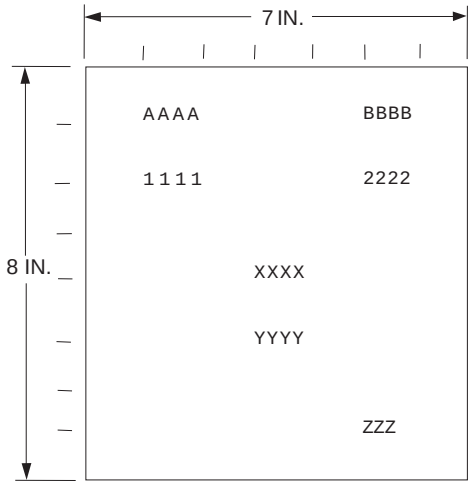


Figure 31. Data Arranged on the Printed Page with Color

The data ‘AAAA’ and ‘1111’ are printed in highlight color 1. The data ‘XXXX’ and ‘YYYY’ are printed in highlight color 2. The data ‘BBBB’, ‘ZZZ’, and ‘2222’ are printed in highlight color 3.

Setup Verification

You can use the VFYSETUP subcommand to put the name of the printer color setup into the form definition. At print time, print server compares the setup name in the form definition to verify that the setup was activated in the printer. See the VFYSETUP subcommand in “FORMDEF Command” on page 169 for more information.

Varying Fonts on a Page

This example illustrates a simple font variation within a printout. The task is to print a line-data file having the first line of each page in bold-faced type and the rest in standard type. This requires controls for two fonts in the page definition.

The commands to select a single font for the page, as shown in Figure 32 on page 48, are as follows:

The FONT command contains two names: the local (STANDARD) name and the user-access (M101) name for the selected font.

```
PAGEDEF ABCD ;
  FONT STANDARD M101 ;
  PRINTLINE ;
```

Note: Fonts cannot be an FGID. Also, all page definitions require a PRINTLINE command.

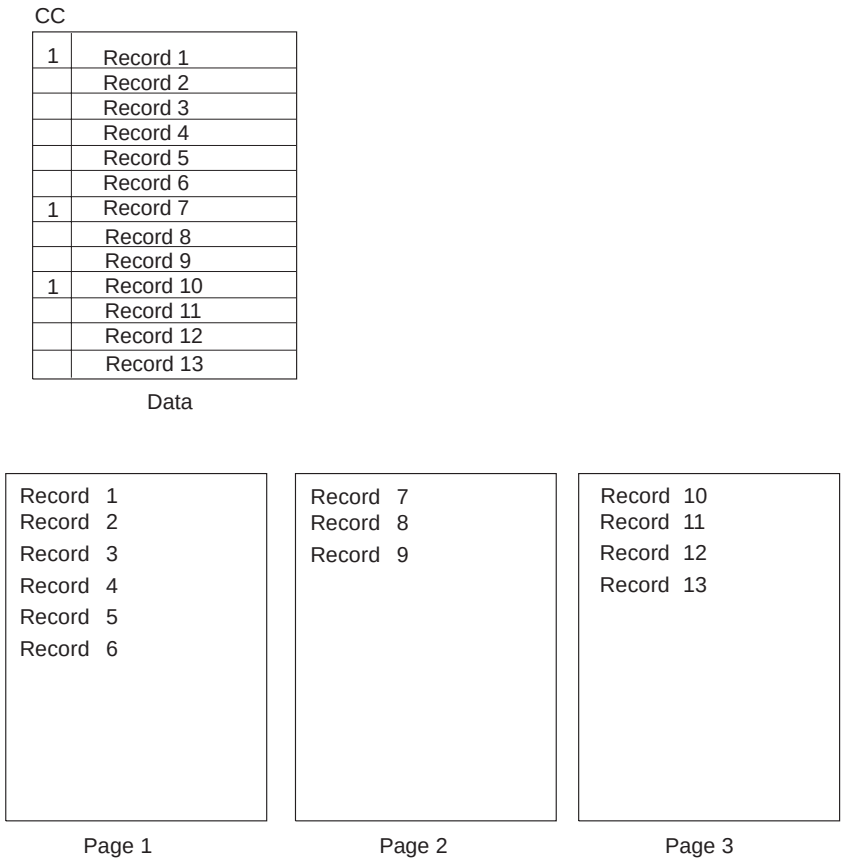


Figure 32. Data File Printed Using a Single Font

The next command stream changes the font by incorporating a TRCREF command. Assume the data file to be formatted incorporates table reference characters (TRCs) as shown in Figure 33 on page 49.

```
PAGEDEF ABCD ;
  FONT STANDARD M101 ;     /*CREATING LOCAL FONT NAMES  */
  FONT BOLDFACE M102 ;
  PAGEFORMAT ABCD ;
    TRCREF 0                /*DEFINING THE TRC VALUES    */
           FONT STANDARD ;
    TRCREF 1
           FONT BOLDFACE ;
  PRINTLINE CHANNEL 1
            POSITION 1 IN 1 IN
            REPEAT 8 ;
```

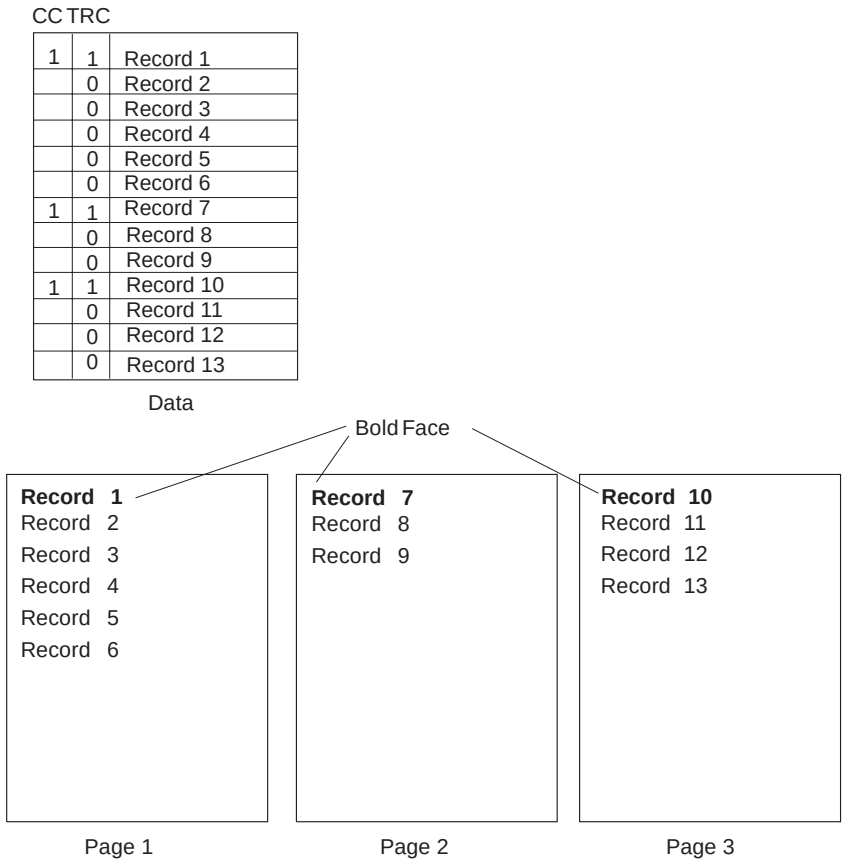


Figure 33. Font Change Using TRCREF Command

The TRCs in the data cause the font switch to be made. The TRCREF command equates a TRC in the data file with the local name of a font specified in the FONT command. The FONT command also contains the user-access name for the font. See Table 7 on page 151 for information on local names and user-access names. Because of the relationship among the user-access name, the local name, and the TRC number that is established in the page definition, the TRCs in the data can cause a font switch automatically.

You can specify fonts within a PRINTLINE command when the data file contains no TRCs. For example:

```
PAGEDEF ABCD ;
  FONT M101 ;
  FONT BOLDFACE M102 ;
    PRINTLINE CHANNEL 1              /*BOLDFACE LINE      */
              POSITION MARGIN  TOP
              FONT BOLDFACE ;
    PRINTLINE POSITION MARGIN NEXT   /*STANDARD-TYPE LINE */
              FONT M101
              REPEAT 7 ;
```

assume the data file represented in the sample print in Figure 34 on page 50 is to be formatted by this page definition.

This command stream, based on a data file without TRCs, works on the principle that each line of output whose font you want to change from the font in the previous line must be controlled by a separate PRINTLINE command. The FONT subcommand of the PRINTLINE command names the font desired for that line. In

this example, two PRINTLINE commands are used because one font change and two fonts are intended for the output. The user-access font names appear in the two FONT commands immediately below the PAGEDEF command and, optionally, a local name. M101 and M102 in the example are user-access names; BOLDFACE is a local name. Use the local name in the FONT subcommand of PRINTLINE if it is included in the corresponding FONT command, as is done for the first PRINTLINE command.

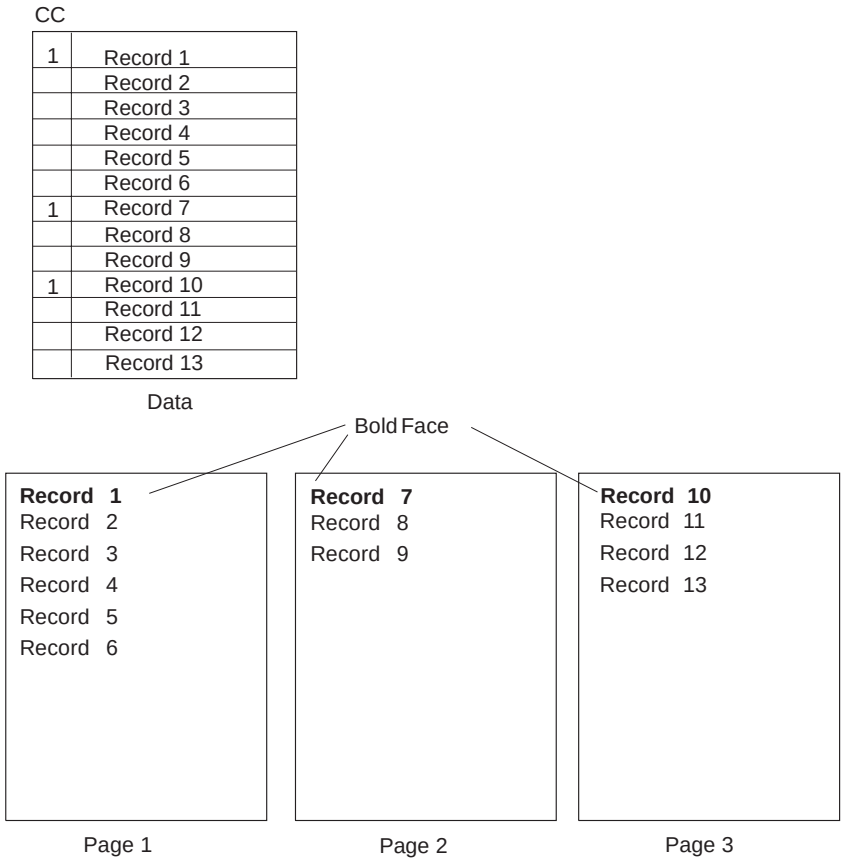


Figure 34. Font Change Using FONT Commands and Subcommands

Changing fonts field by field is similar to changing them in printlines. You map each field individually with a FIELD command; include a FONT subcommand in the FIELD command. If a font change is desired for a field, as with the FONT subcommand of a PRINTLINE command, the font must be previously named in a FONT command.

Two possible defaults apply in case you do not specify a font within a field. If the governing printline has a FONT subcommand, it will contain the font default for the field. If the governing printline has no font specification, print server will assign a font according to its default rules.

Printing Lines in Two Directions on a Page

Lines can be printed in any of four directions, depending on the type of printer being used. Refer to *Advanced Function Presentation: Printer Information* for the print directions supported by your printer.

The four parameters for line direction are ACROSS, DOWN, BACK, and UP. The PPFA commands used to format a line-data file with lines printed in more than one direction (as shown in Figure 35) are stated in the following page definition:

```
PAGEDEF ATOG
       DIRECTION ACROSS ;
  PRINTLINE  POSITION 1 IN 1 IN       /*LINES A-E   */
             REPEAT 5 ;
  PRINTLINE  POSITION .5 IN 6 IN      /*LINE  F     */
             DIRECTION UP ;
  PRINTLINE  POSITION 1 IN 6 IN ;     /*LINE  G     */
```

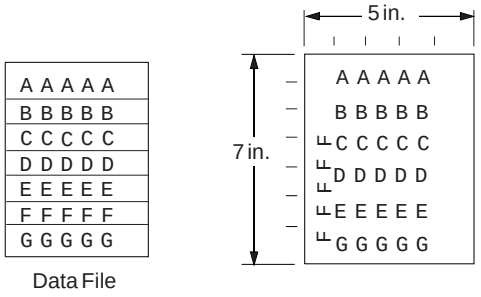


Figure 35. A Printout with More Than One Line Direction

In this page definition, the logical page direction ACROSS is specified. This is actually the default, but its inclusion clarifies that no direction control is needed for lines A–E. The default direction of a printline is the direction specification of the logical page of which it is part. The PRINTLINE command for the record F has a DIRECTION subcommand because the direction specification changes from that of the previous line. Record G is to be printed in the ACROSS direction again. A direction is not specified, however, because the ACROSS direction is the default for all lines in this page definition.

Note: If you are building the page definition for use with the 3800 printer, and if the input data contains table reference characters, you can use the DIRECTION subcommand of the TRCREF command to specify a font that will print UP on the page, as in line F. For more information, see "TRCREF Command (Traditional)" on page 248.

Printing Fields in Two Directions on the Same Page

This example is similar to Printing Lines in Two Directions on a Page, except that you learn how to control direction field by field. This method creates a field-processing page definition and places direction controls in the FIELD commands. This command stream contains a portion of the page definition controls, showing only the PRINTLINE commands:

```
PRINTLINE POSITION MARGIN TOP ;
  FIELD START 1 LENGTH 4 ;
PRINTLINE POSITION 2 IN 4 IN ;
  FIELD START 7 LENGTH 4
       DIRECTION UP ;
```

As expected in field processing, FIELD commands are nested within PRINTLINE commands. Figure 36 on page 52 shows a simplified portion of an unformatted file and two pages of the printout formatted by the page definition, part of which is shown in the command stream. Two printlines are specified because, as Figure 36 on page 52

on page 52 shows, the data file contains two input record formats (1 and 3 are alike; 2 and 4 are alike) and because the fields are mapped to two different positions in the output. The assumption of this sample is that the data file is actually much longer than the portion shown. If, however, the records in the file alternate in format as the first four do, the two PRINTLINEs of this page definition will format as many records as are presented, two to a page, on pages 1 through *n*.

If more than two mappings are required by the print job, more than two PRINTLINE commands are required in the page definition.

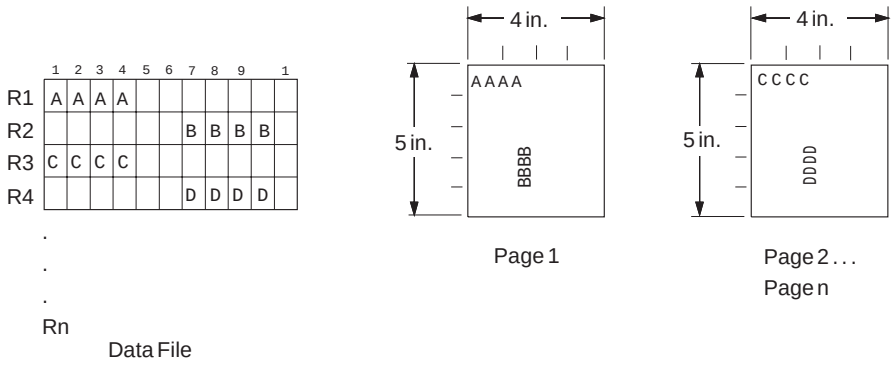


Figure 36. Field Direction

Rotating Fonts

Fonts rotate relative to the inline direction of lines (or fields).

This example focuses on a single letter A from FONTA. With PPFA, a single font specified in a page definition can produce letters in any of four rotations. This is accomplished by a FONT command that specifies rotation. If, as in this example, you want to vary the rotation of a font twice within a page, you use two FONT commands, one for each rotation. You also use two PRINTLINE commands to map the data to the printout, using the two rotations of the font. In a field processing application, FIELD commands can be used in the same way. These PRINTLINE commands name the rotated font in a FONT subcommand.

Figure 37 breaks down the elements required for the FONT commands and subcommands. Distinct local names and rotation specifications for each font are placed in a FONT command. These identify a font as rotated within a page definition. The rotation of a character is relative to the inline direction of a printline or field. The characters and rotations shown here assume an inline direction of ACROSS. See “PPFA Basic Terms” on page 8.

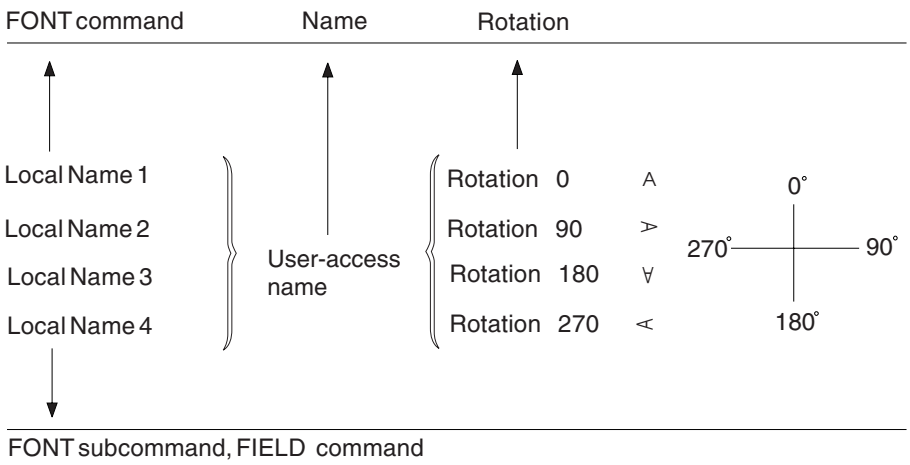


Figure 37. Character Rotation

You can use up to 16 possible combinations of logical page direction and font rotation for page printers other than the 3800.

The FONT subcommands within PRINTLINE or FIELD commands that name the rotated font in that page definition use only the local name. The following command stream shows the proper specification and nesting of FONT commands and subcommands for rotation.

```
PAGEDEF ABCD ;
  FONT FONTA M103 ;            /*NO ROTATION, LOCAL AND          */
                               /*USER-ACCESS NAMES.              */
  FONT FONTARTD180 M103        /*ROTATED FONT, LOCAL, USER-ACCESS*/
       ROTATION 180 ;          /*NAMES PLUS ROTATION SUBCOMMAND  */
                               /*AND PARAMETER.                  */
  PRINTLINE  FONT FONTA        /*LOCAL NAME                      */
             REPEAT 3 ;
  PRINTLINE  FONT FONTARTD180  /*LOCAL NAME                      */
             REPEAT 2 ;
```

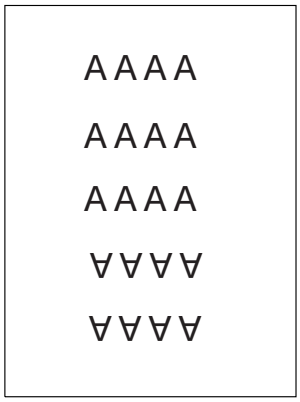


Figure 38. Example of Assumed Data File and Rotation Specifications

FONTA, identified in the first FONT command, requires no rotation parameter because it is printed in the default position (or 0° rotation) for font M103. For the rotated font, the second FONT command identifies FONTARTD180 (the local name) as M103 rotated 180°.

Using Traditional Kanji Formatting

Traditional kanji print presentation, called *tate*, is possible with print server printers, using a combination of font rotation and logical page direction. A logical page in the DOWN direction and a 270° font rotation provide the right combination to present kanji in tate format on an print server printer.

```
FORMDEF TATE
        OFFSET 1 IN 1 IN ;
PAGEDEF TATE
        HEIGHT 5 IN
        WIDTH 6 IN
        DIRECTION DOWN ;
  FONT  KANJIRTD  M104
        ROTATION 270 ;
  PRINTLINE FONT KANJIRTD
            REPEAT 3 ;
```

Figure 39 shows the result of formatting with the above page definition. The characters are added to lines down the page. Lines are added right to left.

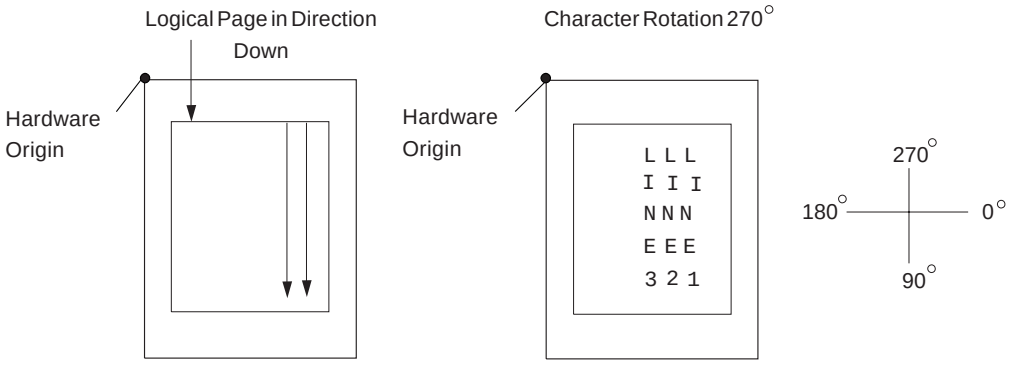


Figure 39. 3820 Tate Presentation

Printing Multiple-Up Pages

Multiple up is a printer's term for printing two or more pages of data on one side of a sheet, which is possible with print server printers and PPFA formatting. The steps used in this example are:

1. Change the print direction of the logical page to one of the landscape presentations.
2. Conceptually divide the sheet of paper into parts, one for each multiple-up page (subpage).
3. Create a printline position at the top of each multiple-up page.

This example assumes the existence of a line-data file with carriage control 1 characters after records 4, 7, and 11. Each carriage control 1 character begins a new page. Because there are really four pages on the sheet, a skip-to-channel 1 must be used four times. The fifth channel 1 character causes a page eject and the beginning of a new physical sheet. The PPFA commands that follow are for one

version of a multiple-up page. This set of commands creates a page layout like the one shown in Figure 40 (the physical sheet is not shown).

```
FORMDEF MULTUP
        OFFSET 1 IN .5 IN ;
SETUNITS LINESP 4 LPI ;
PAGEDEF  MULTUP1
         WIDTH 10 IN
         HEIGHT 8 IN
         DIRECTION DOWN           /*FOR LANDSCAPE PRESENTATION  */
  PRINTLINE CHANNEL 1             /*PAGE 1                      */
            POSITION 1 IN 1.5 IN
            REPEAT 6 ;
            ENDSUBPAGE ;
  PRINTLINE CHANNEL 1             /*PAGE 2                      */
            POSITION 1 IN 5.5 IN
            REPEAT 6 ;
            ENDSUBPAGE ;
  PRINTLINE CHANNEL 1             /*PAGE 3                      */
            POSITION 6 IN 1.5 IN
            REPEAT 6 ;
            ENDSUBPAGE ;
  PRINTLINE CHANNEL 1             /*PAGE 4                      */
            POSITION  6 IN 5.5 IN
            REPEAT 6 ;
```

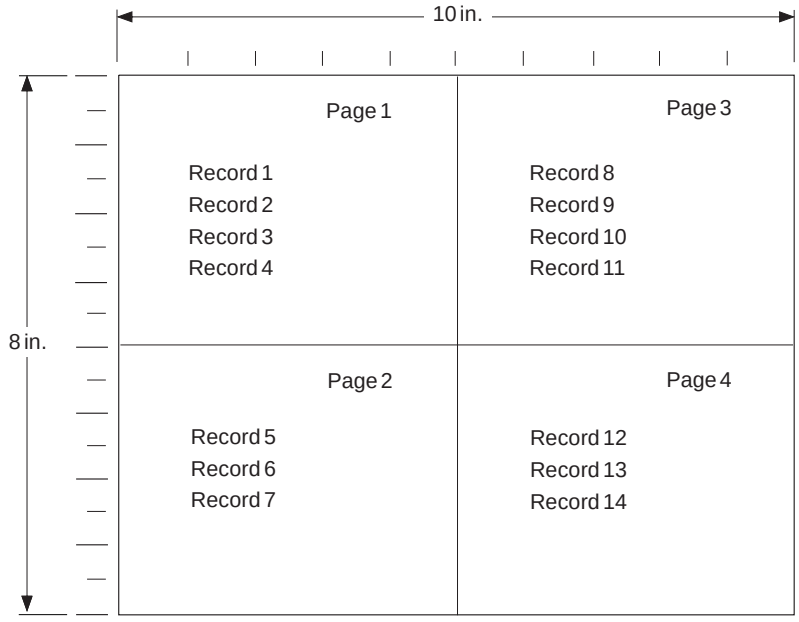


Figure 40. Multiple-Up Page Layout

The DOWN printline direction creates a page with a landscape presentation typical of multiple-up printing. Individual printlines are specified for the initial lines of the four pages. Ensure that the lines of each page fit in the space designated by the use of a small font.

Note: In this example, no font is specified for the page definition; therefore, the default font for the page printer is used. If you want a different font, write a FONT command naming it.

The next set of commands alters the sequence of pages.

```
FORMDEF MULTUP
  OFFSET 1 IN .5 IN ;
SETUNITS LINESP 4 LPI ;
PAGEDEF MULTUP2
        WIDTH 10 IN
        HEIGHT 8 IN
        DIRECTION DOWN ;
  PRINTLINE CHANNEL 1             /*  PAGE 1   */
            POSITION 1 IN 1.5 IN
            REPEAT 4 ;
            ENDSUBPAGE ;
  PRINTLINE CHANNEL 1             /*  PAGE 2   */
            POSITION 6 IN 1.5 IN
            REPEAT 4 ;
            ENDSUBPAGE ;
  PRINTLINE CHANNEL 1             /*  PAGE 3   */
            POSITION 1 IN 5.5 IN
            REPEAT 4 ;
            ENDSUBPAGE ;
  PRINTLINE CHANNEL 1             /*  PAGE 4   */
            POSITION  6 IN 5.5 IN
            REPEAT 4 ;
```

Here, the upper-right and lower-left pages have been reversed by reversing the position controls for the second and third printlines.

Figure 41 shows the changed printout resulting from the page definition command changes. Once you have set up your basic page definition, changes such as this become easy.

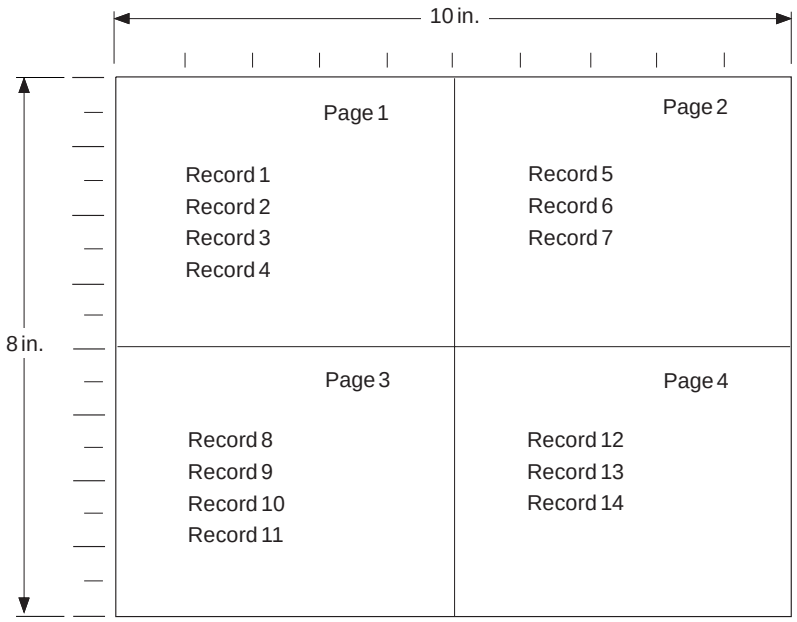


Figure 41. Multiple-Up Page Layout after Page Definition Modification

Note: The ENDSUBPAGE command can be used to mark the boundaries between subpages. Without it, the page definition is no different from any other sequence of PRINTLINEs with POSITION commands. Boundaries do not have to be marked unless conditional processing is being performed. The examples given here will print identically with and without ENDSUBPAGE commands. (See "Subpage Description and Processing" on page 107 for more information.)

Chapter 4. Using Page Definition Commands for Record Format Line Data

Record Formatting Function

The *record formatting function* allows an application to specify a format identifier (Record ID) with each set of output data fields (Data Record). The format identifier references a specific layout format in a Page Definition (Pagedef). At print time, each layout format (referenced by a Record ID in a Data Record) is retrieved from the Pagedef and used to position and format the associated Data Records/fields on the output page.

The purpose of the record formatting capabilities is to move more of the output formatting function into the Pagedef and allow for greater flexibility in creating and changing output pages without changing the base application. Rather than the application generating page headers, page trailers and group headers for each page (and thereby fixing the page endings), the page headers, page trailers and group headers can be generated by a Pagedef layout, allowing the page endings to change as font sizes or data layouts change.

In order to visualize how the record formatting function can be used, review the first six pages of “Record Formatting Examples” on page 79. These examples show the output of an application before and after it is formatted with Pagedef using the record formatting functions.

These functions are provided by several new PPFA commands (LAYOUT, DEFINE COLOR, DRAWGRAPHIC and ENDGRAPHIC), and modifications to the PAGEDEF, PAGEFORMAT, FONT, CONDITION, and FIELD commands. This chapter provides an explanation of the record formatting functions with examples of their use. For details on the syntax of these commands, see “Chapter 11. Page Definition Command Reference (Record Formatting)” on page 251.

Some of the functions that can be accomplished in a layout format with the record formatting commands include:

- Selecting different formatting for different types of Data Records/fields based on the Record ID. The output formatting can change mid-page independent of where the output occurs on a page.
- Defining page headers and trailers to be automatically printed on subsequent pages. The headers and trailers can incorporate data from the associated Data Record.
- Numbering the output pages.
- Inserting page ejects can be automatic when text reaches the bottom margin.
- Creating group headings to be printed at the beginning of a group of data. For example, you can create group headings (including column headings) to be repeated each time a different account type is formatted on a banking statement. An active group heading is automatically repeated on subsequent pages until the data group ends.
- Forcing page ejects to occur in the output.
- Creating boxes with or without black and white or color shading. A set of boxes for a table can be started in a group header and automatically ended and restarted on subsequent pages until the table completes.

- Creating graphical objects such as circles, ellipses, lines, graphs, etc. in color or black and white output.
- Formatting database records created with field delimiters (rather than fixed length fields).
- Aligning field output to the left or right.

Record Format Page Definition

A *record format page definition* specifies how you want data positioned on the logical page.

A record format page definition is a resource used by print server that defines the rules of transforming line data and unformatted ASCII into composed pages and text controls for printing. With record format page definitions, you can perform the tasks listed in Table 4.

Table 4. Record Format Page Definition Tasks

Tasks	**Location of an Example**
Creating a page definition	"Page Definition Command Nesting" on page 59
Record ID	"Record ID Data Format" on page 60
Layout Command	"LAYOUT Command" on page 60
Body Records	"Body Records" on page 61
Fields	"Field Command" on page 62
Defining logical page size	"Defining Logical Page Size" on page 66
Positioning data on a logical page	"Positioning the Data" on page 67
Changing the print direction	"Changing Logical Page Print Direction" on page 67
Processing fields	"Processing Fields" on page 70
Changing fonts	"Varying Fonts on a Page" on page 74
Printing in different directions	"Printing Lines in Two Directions on a Page" on page 73
Printing fields in two directions	"Printing Fields in Two Directions on the Same Page" on page 74
Rotating fonts	"Rotating Fonts" on page 76
Printing kanji	"Using Traditional Kanji Formatting" on page 78
Example formats and commands	"Record Formatting Examples" on page 79

Page Formats within Page Definitions

Just as form definitions can include more than one copy group, page definitions can include several *page formats*. Page formats use basically the same subcommands as page definitions, and if a subcommand is specified in a page format, it overrides the value specified in the page definition for the page format. A single page definition may contain multiple page formats. If pages in a file are to be formatted differently, specify more than one page format in your page definition. Within a page definition, page formats are generated in the order in which they are specified.

Using more than one page format to control different pages requires one of the following:

- Adding the Invoke Data Map structured field to the data file each time you want to change page formats.
- Using conditional processing.

Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the Invoke Data Map structured field.

Page Definition Command Nesting

The following simplified command stream shows the proper nesting of commands and the order in which they must be entered when you create a page definition:

```
[SETUNITS]
PAGEDEF
FONT
[OBJECT]
[DEFINE COLOR]
[PAGEFORMAT]
  [SEGMENT]
  [OVERLAY]
  [LAYOUT]
    [CONDITION]
    [FIELD]
    [DRAWGRAPHIC]
    [ENDGRAPHIC]
[PAGEFORMAT]
  [SEGMENT]
  [OVERLAY]
  [LAYOUT]
    [CONDITION]
    [FIELD]
    [DRAWGRAPHIC]
    [ENDGRAPHIC]
```

Notes:

1. Brackets enclosing a command mean the command is optional.
2. Indentations are used to improve readability.
3. Complete definitions of all commands are included in "Chapter 11. Page Definition Command Reference (Record Formatting)" on page 251.

Command Nesting Rules

1. Record format LAYOUT commands and traditional PRINTLINE commands cannot be used within the same PAGEDEF. At least one LAYOUT command is required per page format for a record formatting page definition.
2. A SETUNITS command can be placed before any other PPFA command. The values set are in effect until the next SETUNITS command.
3. SEGMENT and OVERLAY commands must be specified under their associated PAGEFORMAT command.
4. The first PAGEFORMAT command can be omitted in a page definition, if the page definition contains only one page format. If the PAGEFORMAT command is omitted, the PAGEDEF command parameters are used to define the page format.
5. One file can contain multiple sets of page definitions.

Record ID Data Format

In order to allow different formats for different groups (or tables) of data, each of which have an unpredictable number of entries, a Record ID is assigned to each output record to identify the type of record and control layout formatting. An application can group data fields that are to be formatted together as an entity into Data Records with a specific Record ID. For example, in a bank statement, the data fields for a check transaction might be grouped together with a Record ID identifying that record as a check transaction. The Pagedef would then define a special layout format for a check transaction with a matching Record ID (see "Record Formatting Examples" on page 79 for detailed examples).

Record formatting in PPFA is achieved by identifying each input record in the data file with a 10 byte ID, similar to an expanded carriage control (CC) (see "Basic Controls in Record Format Line Data" on page 13 for additional information). Each record in the data file must contain a Record ID if record formatting is used. The Record ID must be the first 10 bytes in **every** print record in the data file.

Even though the Record ID is specified as a character string, the Record ID is treated as a hexidecimal string, not a character string. This means there is no translation from ASCII to EBCDIC or vice versa when the Record ID is processed. The Record ID in the input data must match exactly the string specified for the LAYOUT Record ID in the page definition in order for correct processing to occur.

When a record is read from the data file at print time, the print server will use the 10 byte Record ID to determine which LAYOUT command in the page definition should be used to format the record.

TRCs (Table Reference Characters) cannot be used with record format data. If you have TRCs in the data and tell print server that TRCs are present at print time, the print server will use the TRC byte as the first byte of the Record ID, and the Record ID will not be recognized as such.

Data files can contain both carriage controls and Record IDs. If your data file is mixed mode (line data plus MO:DCA structured fields), then you **must** have a CC byte in the data. The CC byte will not be counted as part of the 10 byte Record ID. If your file is plain line data, then CCs are allowed but not required. (See "Basic Controls in Record Format Line Data" on page 13 for additional information.)

LAYOUT Command

When record formatting, the LAYOUT command is used instead of traditional PRINTLINE commands in the page definition. You cannot mix record format LAYOUT and traditional PRINTLINE commands in the page definition. With LAYOUT (see "LAYOUT Command (Record Format)" on page 289), you can identify four types of Data Records:

- Body Records
- Page Headers
- Page Trailers
- Group Headers

Each of the record types is discussed in the following sections. No matter which type of record you are formatting, you can control the positioning, font, color, and direction for the print record.

The POSITION keyword on the LAYOUT command is used to set the initial print position for subsequent text and graphics placed with the FIELD and DRAWGRAPHIC commands.

- The **horizontal position** can be specified as LEFTMARGIN, at the same position as the previous layout, or at an absolute or relative location given in inches, millimeters, centimeters, points, or pels (see "PAGEDEF Command (Record Format)" on page 304).
- The **vertical position** can be specified as TOPMARGIN, at the same position as the previous layout, at the next vertical position (using current LINESP value), or at an absolute or relative location given in inches, millimeters, centimeters, points, or pels (see "PAGEDEF Command (Record Format)" on page 304).

Body Records

The BODY layout type is used for the majority of data in the user's input file. That is, any record that is not used for special processing as a page header, page trailer, or group header, will contain data to be formatted and placed on the page.

Body records are positioned initially with the LAYOUT command. The default *x* (horizontal) position for each body record is to be at the same horizontal position as the previous LAYOUT. If this is the first LAYOUT on a logical page, the default horizontal position is 0.

The default *y* (vertical) position is to place the layout record down one line (as defined in the LINESP subcommand of the last SETUNITS command) from the previous field. If this is the first LAYOUT on a logical page, the default vertical position is one line down from the top margin of the logical page. See "PAGEDEF Command (Record Format)" on page 304 for details about margins.

You can specify the rotation of data with the DIRECTION keyword on LAYOUT. All of the fields defined for this record layout will use the same direction unless it is overridden on the FIELD command. On relative LAYOUTs and their fields, the rotation must be ACROSS, so that they will have the same net rotation as the page format.

You can also specify fonts and color to be used for the text formatted with this layout record. Double-byte fonts can additionally be requested if you have double byte characters in your data. The color of the text and graphic borders is specified with the COLOR keyword. See "DEFINE COLOR Command (Record Format)" on page 258 and "FONT Command (Record Format)" on page 286 for details.

Page segments, overlays and objects can be included with keywords on the LAYOUT command. This processing is the same as the traditional PRINTLINE command.

Body records can also be identified as belonging to a group. When the GROUP keyword is used on the body LAYOUT, the group header that is in effect at the time will be repeated on subsequent pages as long as the input records use Record ID's that select body LAYOUT and use the GROUP keyword. The group is ended as soon as a Record ID in the input selects a LAYOUT that does not use the GROUP keyword.

Page Headers and Trailers

Page headers and trailers are printed automatically on each new page. Default headers and trailers can be created, which are automatically invoked on each new

page without requiring or allowing any input data. No input record data is allowed in a default header or trailer because they are triggered automatically by page ejects and are not associated with any records in the input data file. See "LAYOUT Command (Record Format)" on page 289 for additional details.

Rather than using the defaults, you can create headers and trailers that are invoked by a Data Record containing the header or trailer Record ID. These headers and trailers can use input record data in their layout, however it is not required.

The following example creates a page header and trailer. See "PAGEDEF Command (Record Format)" on page 304 for additional details.

```
LAYOUT C'statmid'
  SEGMENT ibmlog  1.15 in 1.35 in
   PAGEHEADER NEWPAGE
   POSITION SAME ABSOLUTE NEXT;

LAYOUT C'pgenum' PAGETRAILER
   POSITION SAME ABSOLUTE 10.7 in;
```

Figure 42. Sample Page Header and Trailer

Group Headers

A Group Header layout consists of text, graphics, and other data that is to be printed at the beginning of a group of user records. For example, if you are creating a banking statement, you might define a Group Header for checking, one for savings, etc.

The group header is defined with a special LAYOUT GRPHEADER command, and stays in effect until a BODY layout is encountered that specifies NOGROUP. See "LAYOUT Command (Record Format)" on page 289 for additional details on the GRPHEADER subcommand.

If a logical page eject occurs before the group is ended, the header is printed after the top margin on each new page until the group ends.

Field Command

The FIELD command is used to identify a field in a Data Record to be formatted and placed on the page. FIELD must follow the LAYOUT command, and parameters that are not specified on FIELD will be inherited from the previous LAYOUT. This section describes the new keywords on FIELD that are used with record formatting.

Page numbering can be accomplished by specifying FIELD with the PAGENUM parameter. Most often, you will just specify FIELD PAGENUM with other formatting information such as position and alignment, which will cause the current page number to print at the specified position. The current page number is calculated based on the specification of the PAGECOUNT parameter on the previous PAGEDEF or PAGEFORMAT command. You can override the page number to a specific value using the RESET parameter on the FIELD command. For details, see "Page Numbering" on page 64.

You can retrieve the value of the Record ID for printing using the RECID keyword on FIELD. RECID also has START and LENGTH subparameters to allow only

portions of the Record ID to be printed. Normally, you will only use the RECID parameter for debugging your application by tracing which Record IDs are being processed, although it can be used for anything that makes sense for your application.

You can also specify the POSITION, COLOR, DIRECTION, and ALIGN keywords with the PAGENUM or RECID parameters on FIELD. The BARCODE and SUPPRESSION keywords are not allowed with PAGENUM or RECID, but can be used with other text fields from the Data Record.

ALIGN is a keyword that is allowed with the START/LENGTH or TEXT forms of the FIELD command, but only if you are doing record formatting. ALIGN lets you specify whether the field text should be LEFT or RIGHT aligned at the given horizontal position.

If your Data Records are stored in a database, the fields may be separated with ″field delimiters″ instead of just being positional within the record. The DELIMITER keyword on the preceding LAYOUT command is used to specify the one- or two-byte value that is used to separate fields in the Data Records.

If your data uses field delimiters, you can also specify the FLDNUM parameter on the FIELD command to indicate the number of the field within the record to be extracted, rather than the START position. Fields are numbered from left to right beginning with ″1″. You can also use the starting position (START) and LENGTH keywords with the FLDNUM to indicate that only part of the field is to be formatted. An example of a typical command is:

```
COMMANDS
 .
 .
 .
 LAYOUT 'abc' DELIMITER '*';
  FIELD  FLDNUM 1  START 2  LENGTH 8   ALIGN RIGHT
    POSITION  5.6 in  CURRENT
    FONT varb ; /* Variable text - Amount  */
  FIELD  FLDNUM 2  ALIGN LEFT
    POSITION  1.1 in  .9 in
    FONT varb ; /*variable - customer name */


DATA
 abc       *Here is some data*more data*

FIELDS used
 1st field  'ere is s'
 2nd field  'more data'
```

Figure 43. Sample commands and data with delimiters.

Controlling Page Formatting

Parameters on the PAGEDEF and PAGEFORMAT commands let you specify the margins of the page. The TOPMARGIN and BOTMARGIN keywords are used to reserve space at the top and bottom of the page. The page headers and trailers are normally placed into this reserved space.

Note: No other text or objects should be written into the margins - only page header and trailer data.

The bottom margin is also used for two other purposes:

- a BODY or GRPHEADER Data Record that would cause the baseline position to move into the bottom margin area will cause a logical page eject
- any graphic that has been started with the DRAWGRAPHIC command, but not explicitly ended, will automatically be ended at print time before it extends into the bottom margin area.

You can force a new logical page in the output with the NEWPAGE keyword on a LAYOUT command (see "LAYOUT Command (Record Format)" on page 289). When an input record is encountered whose Record ID matches that LAYOUT name, a page eject will be completed before the record data is processed. If this is a header or trailer layout, the page eject is performed before the header or trailer becomes active.

The ENDSPACE keyword can also be used to control where page ejects are performed. If ENDSPACE is coded on a LAYOUT, and a Data Record with the matching Record ID is encountered, a page eject will be performed before the data is processed - if the remaining space on the page (before the bottom margin) is less than the ENDSPACE value.

The ENDSPACE keyword can be used to ensure that a Table Heading (Group Heading) does not print at the end of a page without allowing space for additional Data Records (body records), or to ensure that a table entry does not print at the bottom of a page without allowing space for a totals record.

The following example shows the use of page margins and the NEWPAGE and ENDSPACE keywords:

```
PAGEFORMAT  chub1   TOPMARGIN 2 in BOTMARGIN  2 in;
/*********************************************/
/** statmid  BODY                            **/
/*********************************************/
LAYOUT  C'statmid'  PAGEHEADER NEWPAGE  ENDSPACE  .5 in
                POSITION .6 in ABSOLUTE .55 in;
         FIELD  TEXT C'Big Brother Bank'  ALIGN LEFT
                FONT comp ;   /* default to LAYOUT positioning*/
```

Figure 44. Sample Page Formatting

Page Numbering

Page numbers can be placed with the PAGENUM keywords on the FIELD command. PAGENUM lets you specify whether the page number should print or not, and whether you want it reset to a specific value rather than using the current value (page count).

The page number prints as an integer (e.g. 1, 2, 3 ...) and has a valid range of 1 to four billion (four unsigned bytes of data). If the specified or defaulted font used for printing the page number is other than an EBCDIC font, you must specify it using the TYPE subcommand on the FONT command.

The page number prints using the font specified on the FIELD command. You can also select a POSITION, COLOR, and DIRECTION for the page number using existing FIELD keywords.

The ALIGN parameter on FIELD can also be used to specify whether you want the page number LEFT or RIGHT aligned at the given position.

The PAGECOUNT keyword is allowed with the PAGEDEF and PAGEFORMAT commands that allows you to specify how page numbering is to be handled when switching between page formats. Page numbering can be stopped, reset, resumed for a certain point or continued from a certain point. For a detailed description on how to specify these options, see Pagedef Command on "PAGEDEF Command (Record Format)" on page 304.

Graphical Objects

When creating output with record formatting, you can use the DRAWGRAPHIC commands to create boxes, lines, circles, and ellipses relative to the data printed with the LAYOUT command. DRAWGRAPHIC can be used with DEFINE COLOR to shade an object with a percentage of black or other colors, however DRAWGRAPHIC is not allowed if you are formatting with the traditional PRINTLINE.

Conditional Processing Considerations

Conditional processing works much the same in record formatting as when using the traditional PRINTLINE processing. The only difference is the ability to process based upon a field that is defined by delimiters instead of just a fixed start position and length.

Logical Page Eject Processing

A logical page eject can be caused by the following:

- Any Record ID that references a layout format with a specification of New Page.
- A relative baseline overflow (a Body or Group Header layout format that when processed against the current input record causes an overflow of the current print position into the bottom margin). If processing of the input record would cause a relative baseline overflow, the page eject is processed before any part of the input record is printed.
- A Data Map change or Medium Map change, or, in Mixed-Mode, a Begin Document or Begin Page structured field.

Page Header, Page Trailer, and Group Header Data Records used with page ejects are activated in the following manner:

- If a Data Record specifies the Record ID of a Pagedef Page Header layout format, that Data Record is not printed on receipt but is saved as the active page header record (for that Pageformat). It is saved for the duration of the job or until a subsequent Data Record specifies a Page Header (for that Pageformat).
- If a Data Record specifies the Record ID of a Pagedef Page Trailer layout format, that Data Record is not printed on receipt but is saved as the active page trailer record (for that Pageformat). It is saved for the duration of the job or until a subsequent Data Record specifies a Page Trailer (for that Pageformat).
- If a Data Record specifies the Record ID of a Pagedef Group Header layout format, that Data Record is not printed on receipt but is saved as the active group header record. The Pagedef Group Header is printed when the next Data Record specifies a Body layout with a GROUP specification and on subsequent page ejects. The Group Header and its associated Data Record is kept active until a subsequent Data Record specifies a Body layout with a NOGROUP specification.

When a logical page eject occurs, the following actions are taken in the following order.

- For the current page:
  1. If this is the start of a line data document (no previous page ejects, group header records or body records have been processed with this Pagedef), current page items 1 through 3 are skipped.
  2. If an active page header record was in effect prior to this layout format, that record is presented on the current page using the matching layout. Otherwise, if the active Pageformat contains a default Page Header layout, that layout is used to present a page header.
  3. If an active page trailer record was in effect prior to this layout format, that record is presented on the current page using the matching layout. Otherwise, if the active Pageformat contains a default Page Trailer layout, that layout is used to present a page trailer.
- For the new page:
  1. The current print position is moved to the top of the new page and offset from the top of the new page by the top margin. If the Pageformat is changed, the new Data Map's Margin Definition and layouts are used.
  2. If an active group header record exists for this Pageformat, that record is presented on the new page using the matching Record layout. Note that the group header is not actually printed and causes no action until a Body layout with Group Indicator is processed for the page. If the layout specifies relative positioning, the baseline position of the layout is offset from the top of the page by the top margin plus one line.
  3. If the page eject was caused by a Body layout, the input record causing the page eject is presented on the new page using the layout referenced by the record. If the layout specifies relative positioning and is preceded on the page by a group header, the baseline position is relative to the last printed line of the group header. If the layout specifies relative positioning and is not preceded on the page by a group header, the baseline position of the layout is offset from the top of the page by the top margin plus one line.

Note: The actual locations of 'top of page' and 'top margin' are affected by the text orientation. See "Using Margins in Record Formatting" on page 69 for additional information.

Defining Color Models

Record formatting provides you with the ability to predefine a color with your own name and then use that name anytime this color is needed. It works in much the same way as a FONT command where you define the FONT with an internal name and then use that name when you place text on the page.

Defining Logical Page Size

"Positioning a Logical Page on a Sheet" on page 19 shows how to establish the origin point of a logical page, relative to the media origin on a sheet of paper, using the OFFSET subcommand. The following example shows you how to establish the width and height of the logical page relative to this origin point. This example illustrates how the dimensions of a logical page are determined by form definitions and page definitions.

```
SETUNITS  1 IN 1 IN
           LINESP 8 LPI;
FORMDEF   ABCD
           OFFSET 0 .5;
```

```
PAGEDEF   ABCD
           WIDTH 7.5
           HEIGHT 10
           DIRECTION ACROSS;
   FONT GS12 GS12;
   LAYOUT 'abc'
           FONT GS12
           POSITION  0 TOP;
```

Normally, all parameters consist of a number and a unit of measurement, for example, 6 IN. (See “Units of Measurement” on page 152 for information on units that are available.) Numbers can be specified with up to three decimal places. The LAYOUT command is included because at least one is required for all page definitions; see “LAYOUT Command (Record Format)” on page 289 for more information.

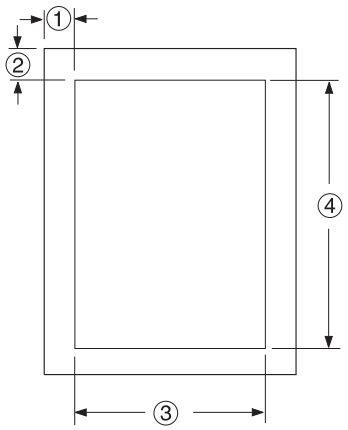


Figure 45. Logical Page Dimensions

The OFFSET subcommand (0) (.5) in the sample form definition establishes the corner or origin of the logical page relative to the physical sheet. The WIDTH and HEIGHT subcommands, (7.5) and (10), specify the dimensions of the logical page relative to the logical page origin.

Note: Be careful not to define a logical page larger than the physical sheet. PPFA does not check the size of the physical sheet.

Positioning the Data

The previous section showed you how to define the size of a logical page. The next examples show you how to position data inside the logical page.

Changing Logical Page Print Direction

Logical pages can have four different print directions: ACROSS, DOWN, BACK, and UP. This example shows that all four directions can be specified in relation to one offset specification:

```
FORMDEF ABCD
         OFFSET (1) (2) ;
PAGEDEF DEFG  ;
FONT GS12 GS12;
   PAGEFORMAT DEFG1
              WIDTH (3)
              HEIGHT (4)
              DIRECTION ACROSS ;
```

```
  LAYOUT 'abc'  ;
PAGEFORMAT DEFG2
           WIDTH (3)
           HEIGHT (4)
           DIRECTION DOWN ;
  LAYOUT 'def'  ;
PAGEFORMAT DEFG3
           WIDTH (3)
           HEIGHT (4)
           DIRECTION BACK ;
  LAYOUT 'ghi'  ;
PAGEFORMAT DEFG4
           WIDTH (3)
           HEIGHT (4)
           DIRECTION UP ;
  LAYOUT 'jki'  ;
```

Note: The parenthetical numbers represent dimensions. Figure 45 on page 67 shows how these dimensions relate to the logical page.

One page definition is used to simplify the example, yet four logical pages are specified. The PAGEFORMAT commands create subsets of page definitions for each logical page.

Note: The page formats in this example require an Invoke Data Map structured field at the place in the data file where you want to change page formats. The LAYOUT commands are required but are not relevant in the example.

The DIRECTION subcommand with one of its four direction parameters, ACROSS, DOWN, UP, or BACK, specifies the print direction of the logical page.

Figure 46 on page 69 shows the format of each of the logical pages specified in the page definition with the direction specification of each. The pages with the ACROSS and BACK directions are in portrait presentation. The pages with the DOWN and UP directions are in landscape presentation.

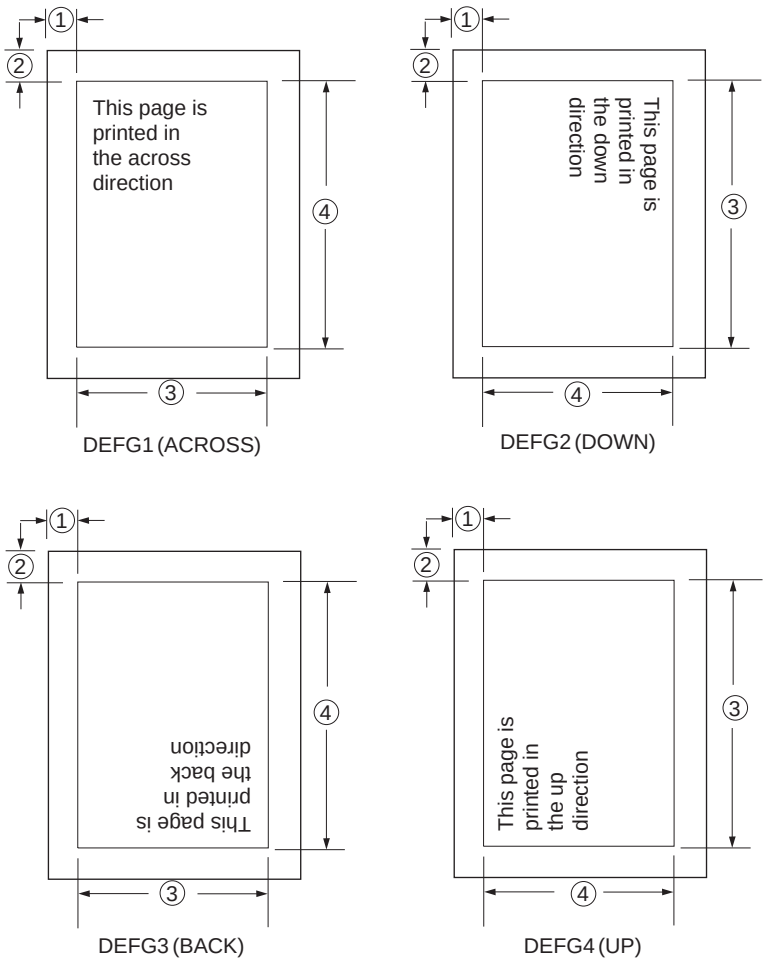


Figure 46. Logical Page Print Directions in Relation to Origin

The media origins and logical page origins do not change with the presentation of the data on the page. The OFFSET subcommand of the form definition need not change. However, the width and height dimensions do change; that is, the WIDTH subcommand always governs the horizontal (inline) dimension as you view the page, and the HEIGHT subcommand always governs the vertical (baseline) dimension whether the page is in portrait or in landscape presentation. Ensure that these specifications do not cause the logical page to cross the edge of the physical page.

However, if the DOWN direction is specified for use with the IBM 3835 or 3900 Page Printer, the PRESENT and DIRECTION subcommands may need to be specified in the form definition. See “Specifying Page Presentation on Continuous-Forms Printers” on page 27 for more information.

Using Margins in Record Formatting

Margins follow the inline direction of the page. For example, if the text orientation is Across, the top-left diagram in Figure 47 shows the left, top, right, and bottom margins, respectively. Once specified, these margins define a bounding box for the Pageformat as indicated by the dotted lines.

Note that if the text orientation is changed, the same bounding box applies to the new orientation, but the name of the margins change in the new orientation. For

example, if the new text orientation is Down, as shown in the top-right diagram of this same figure, the top margin in the new orientation is now defined on the long side of the page, and so on.

- **Left Margin** = Specifies the offset of the left margin along the i axis from the left edge of the page. The left edge of the page is the zero position on the i axis.
- **Top Margin** = Specifies the offset of the top margin along the b axis from the top edge of the page. The top edge of the page is the zero position on the b axis.
- **Right Margin** = Specifies the offset of the right margin along the i axis from the right edge of the page.
- **Bottom Margin** = Specifies the offset of the bottom margin along the b axis from the bottom edge of the page.

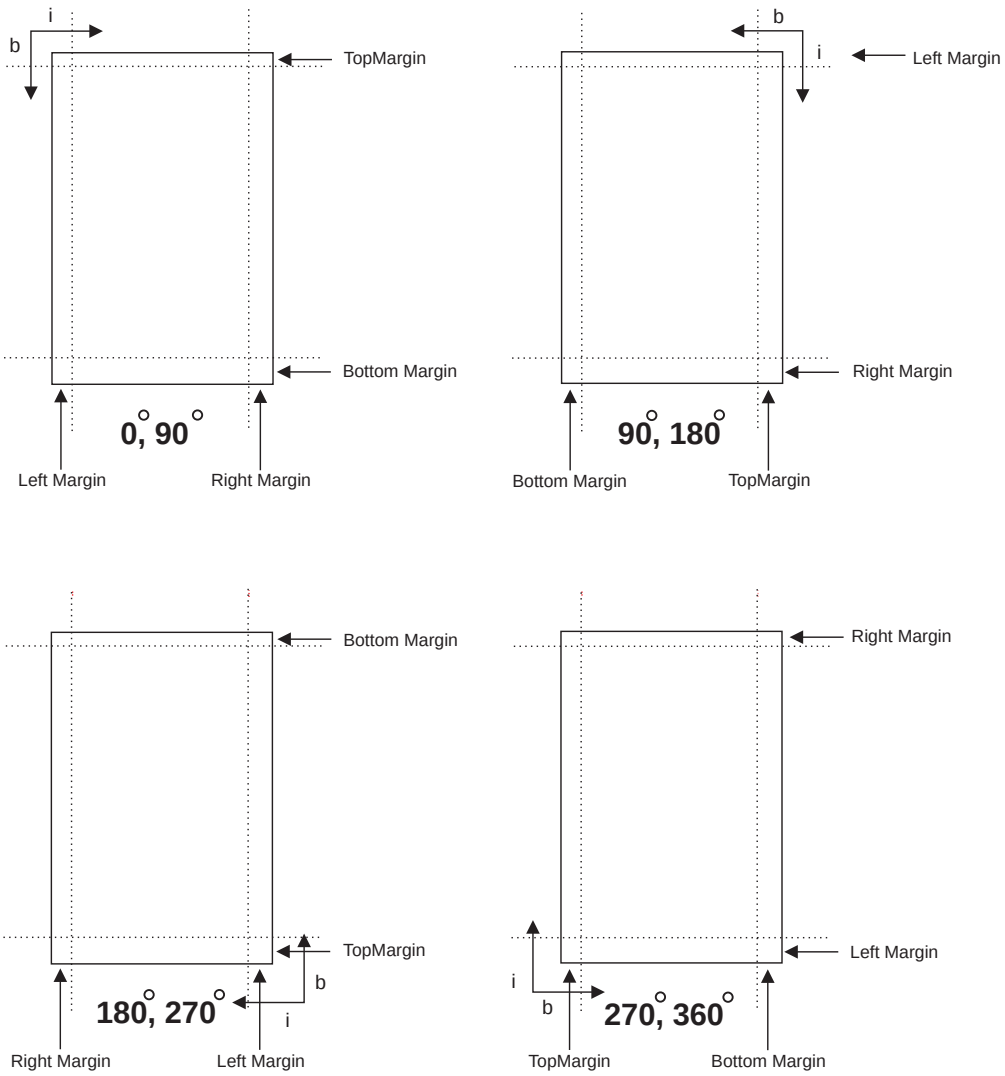


Figure 47. Relationship of Margin Definition to Text Orientation

Processing Fields

This section describes the mapping of individual fields to the printed sheets. The technique allows you to print unformatted data according to precise specifications, and these specifications can change without affecting the data file.

The rule for field processing of data files is: Each FIELD command must follow its associated LAYOUT command, and more than one FIELD command can be specified for a single LAYOUT command.

For this field-processing example, the data file shown in Figure 48 is used. Figure 49 represents an output format that could be used to place data on a form, such as an invoice or an order. The page definition commands to print Figure 49 are as follows:

```
PAGEDEF ABCD
        WIDTH 7 IN
        HEIGHT 8 IN ;
FONT GS12 GS12;
LAYOUT 'abc' POSITION 1 IN ABSOLUTE 1 IN ; /*PROCESSING FOR R1   */
  FIELD START 1 LENGTH 4 ;      /*THE LAYOUT POSITION IS         */
                                /*THE DEFAULT FOR THE FIRST FIELD*/
  FIELD START 11 LENGTH 4
        POSITION 4 IN 0 IN ;
 LAYOUT 'def' POSITION 3 IN ABSOLUTE 4 IN ; /*PROCESSING FOR R2  */
  FIELD START 1 LENGTH 4 ;      /*DEFAULT POSITION               */
  FIELD START 6 LENGTH 4
        POSITION 0 IN 1 IN ;
  FIELD START 13  LENGTH 3
        POSITION 2 IN 3 IN ;
 LAYOUT 'ghi' POSITION 1 IN ABSOLUTE 2 IN ; /*PROCESSING FOR R3  */
  FIELD START 1 LENGTH 4 ;      /*DEFAULT POSITION               */
  FIELD START 11 LENGTH 4
        POSITION 4 IN 0 IN ;
```

Note: The data area of this example does not show the Record ID.

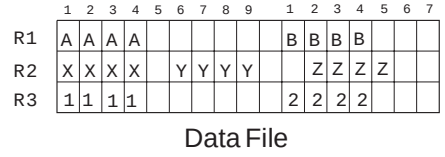


Figure 48. Unformatted Print Data File

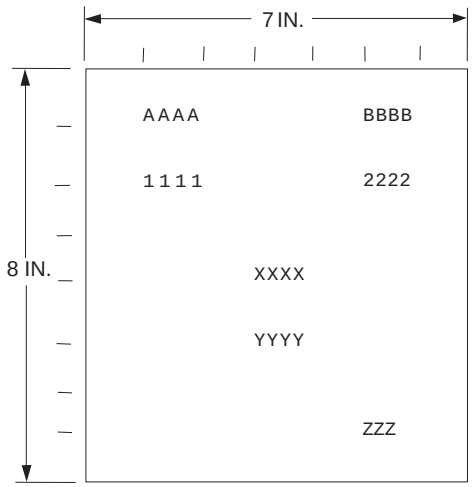


Figure 49. Data Arranged on the Printed Page

Position Subcommand

The POSITION subcommand of each LAYOUT command specifies the layout position relative to either the logical page origin or the previous LAYOUT position. The POSITION subcommands below FIELD commands specify a field position relative to the governing LAYOUT position.

This is for use in positioning text, objects and graphics. If RELATIVE is specified or POSITION is not specified, the baseline of the Position is relative to the previous LAYOUT position.

1. For PageHeader LAYOUT: The baseline position can be anywhere on a logical page.
2. For PageTrailer, GroupHeader and Body LAYOUT: The baseline position can be anywhere on a logical page and can be specified as Relative.

Following POSITION subcommands come the horizontal (*x*) then the vertical (*y*) offsets from the reference point.

- *x* = Specifies the horizontal offset from the left side of the logical page.
- *y* = Specifies the vertical offset from the top side of the logical page.

They are parallel in structure to the OFFSET subcommand of the form definition.

For example, the final POSITION subcommand on the previous example places the final field 1 + 4 inches to the right of the left edge of the logical page, combining the *x* value of 1 in the LAYOUT command, and the *x* value of 4 in the nested FIELD command. The 0 in the FIELD command specifies no change to the *y* value in the LAYOUT command. Thus, the position of the final field is 5 IN (*x*), 2 IN (*y*).

Note: The first FIELD command within each LAYOUT has no position specification, because the LAYOUT POSITION value is the default for the first FIELD command nested under it.

Alternate controls for the *x* and *y* values of a POSITION subcommand are available. See the description of the POSITION subcommand in FIELD Command (Record Format).

FIELD Command as Used in this Example

In the FIELD command, the START and LENGTH parameters specify the location of the field in the record to be processed. START indicates the starting byte position, and LENGTH specifies the number of bytes in the field.

```
setunits linesp 6 lpi;
PAGEDEF rel9  replace yes
  direction across width 8.5 in height 11.0 in;
FONT GS12 GS12;
LAYOUT 'abc' position 0 IN 1.0 IN;

/* The fields will be placed at +120 pels, +24 pels (next) */
/* and +48 pels (.20 IN) from lines previously placed on page */

setunits linesp 10 lpi;
LAYOUT 'def' position 0 relative next;
  FIELD START 1 LENGTH 3 position 0 IN .5 IN;
  FIELD START 4 LENGTH 3 position 0 IN next;
  FIELD START 7 LENGTH 3 position current .20 IN;
```

Printing Lines in Two Directions on a Page

Lines can be printed in any of four directions, depending on the type of printer being used. Refer to *Advanced Function Presentation: Printer Information* for the print directions supported by your printer.

The four parameters for line direction are ACROSS, DOWN, BACK, and UP. The PPFA commands used to format a line-data file with lines printed in more than one direction (as shown in Figure 50) are stated in the following page definition:

```
PAGEDEF ATOG
        DIRECTION ACROSS ;
  FONT GS12 GS12;
  LAYOUT 'abc'  POSITION 1 IN  ABSOLUTE 1 IN ; /*LINES A-E   */
  LAYOUT 'def'  POSITION .5 IN ABSOLUTE 6 IN   /*LINE  F     */
          DIRECTION UP ;
  LAYOUT 'ghi'  POSITION 1 IN ABSOLUTE  6 IN ; /*LINE  G     */
```

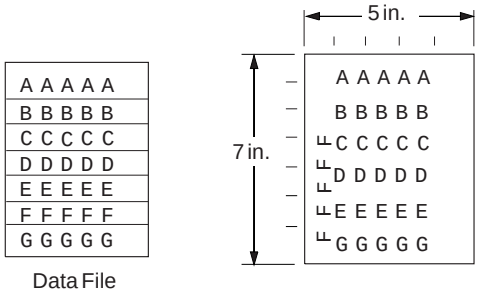


Figure 50. A Printout with More Than One Line Direction

Note: The data area of this example does not show the Record ID.

In this page definition, the logical page direction ACROSS is specified. This is actually the default, but its inclusion clarifies that no direction control is needed for lines A–E. The default direction of a layout is the direction specification of the logical page of which it is part. The LAYOUT command for the record F has a DIRECTION subcommand because the direction specification changes from that of

the previous line. Record G is to be printed in the ACROSS direction again. A direction is not specified, however, because the ACROSS direction is the default for all lines in this page definition.

Printing Fields in Two Directions on the Same Page

This example is similar to Printing Lines in Two Directions on a Page, except that you learn how to control direction field by field. This method creates a field-processing page definition and places direction controls in the FIELD commands. This command stream contains a portion of the page definition controls, showing only the LAYOUT commands:

```
LAYOUT 'abc' POSITION LEFTMARGIN TOPMARGIN NEWPAGE;
  FIELD START 1 LENGTH 4 ;
LAYOUT 'def' POSITION 2 IN ABSOLUTE 4 IN ;
  FIELD START 7 LENGTH 4
        DIRECTION UP ;
```

As expected in field processing, FIELD commands are nested within LAYOUT commands. Figure 51 shows a simplified portion of an unformatted file and two pages of the printout formatted by the page definition, part of which is shown in the command stream. Two layouts are specified because the data file contains two input record formats (1 and 3 are alike; 2 and 4 are alike) and because the fields are mapped to two different positions in the output. The assumption of this sample is that the data file is actually much longer than the portion shown. If, however, the records in the file alternate in format as the first four do, the two LAYOUTs of this page definition will format as many records as are presented, two to a page, on pages 1 through *n*.

If more than two mappings are required by the print job, more than two LAYOUT commands are required in the page definition.

Note: The data area of this example does not show the Record ID.

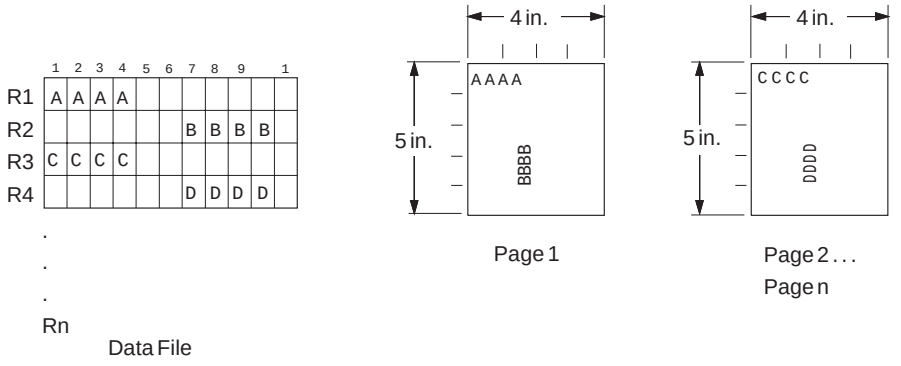


Figure 51. Field Direction

Varying Fonts on a Page

This example illustrates a simple font variation within a printout. The task is to print a line-data file having the first line of each page in bold-faced type and the rest in standard type. This requires controls for two fonts in the page definition.

The commands to select a single font for the page, as shown in Figure 53 on page 75, are as follows:

The FONT command contains two names: the local (STANDARD) name and the user-access (M101) name for the selected font.

```
PAGEDEF ABCD ;
  FONT STANDARD M101;
  FONT BOLDFACE M102;
  LAYOUT 'abc' FONT BOLDFACE NEWPAGE;
  LAYOUT 'def' FONT STANDARD NEWPAGE;
  LAYOUT 'ghi' FONT STANDARD;
```

Note: Fonts cannot be an FGID (Font Typeface Global Identifier). (See page 407 of the Glossary for additional explanation.) Also, all page definitions require a LAYOUT command.

The following example shows line data using a single font:

```
def       Record 1
ghi       Record 2
ghi       Record 3
ghi       Record 4
ghi       Record 5
ghi       Record 6
def       Record 7
ghi       Record 8
ghi       Record 9
def       Record 10
ghi       Record 11
ghi       Record 12
ghi       Record 13
```

Figure 52. Line Data for Single Font Example

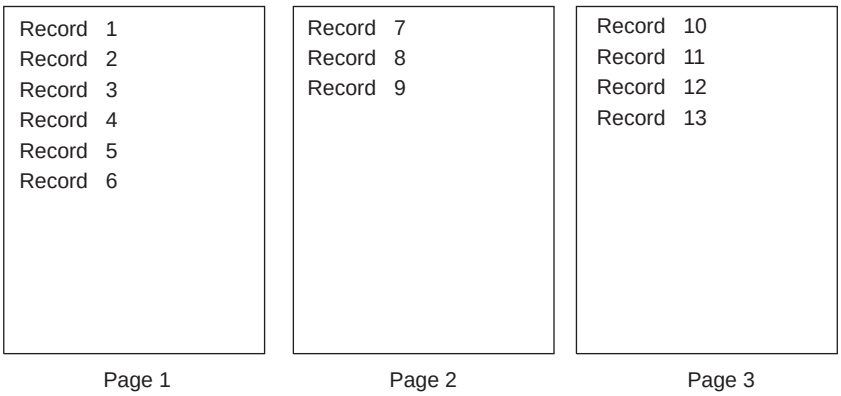


Figure 53. Data File Printed Using a Single Font

This command stream works on the principle that each line of output whose font you want to change from the font in the previous line must be controlled by a separate LAYOUT command. The FONT subcommand of the LAYOUT command names the font desired for that line. In this example, two LAYOUT commands are used because one font change and two fonts are intended for the output. The user-access font names appear in the two FONT commands immediately below the PAGEDEF command and, optionally, a local name. M101 and M102 in the example are user-access names; BOLDFACE is a local name. Use the local name in the FONT subcommand of LAYOUT if it is included in the corresponding FONT command, as is done for the first LAYOUT command.

```
abc       Record 1
ghi       Record 2
ghi       Record 3
ghi       Record 4
ghi       Record 5
ghi       Record 6
abc       Record 7
ghi       Record 8
ghi       Record 9
abc       Record 10
ghi       Record 11
ghi       Record 12
ghi       Record 13
```

Figure 54. Line Data for Two Font Example

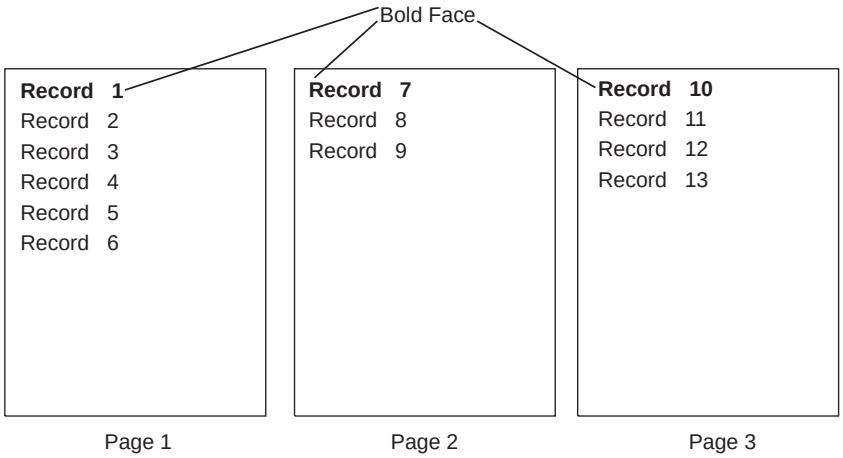


Figure 55. Font Change Using FONT Commands and Subcommands

Changing fonts field by field is similar to changing them in layouts. You map each field individually with a FIELD command; include a FONT subcommand in the FIELD command. If a font change is desired for a field, as with the FONT subcommand of a LAYOUT command, the font must be previously named in a FONT command.

Rotating Fonts

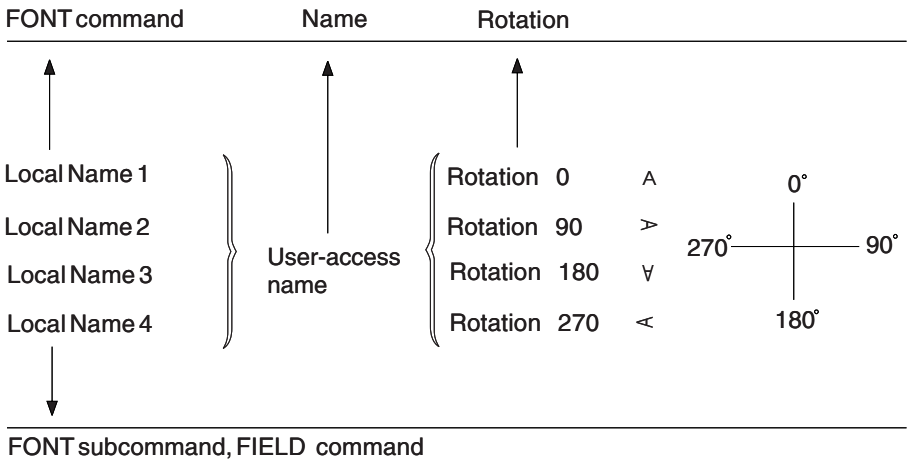
Fonts rotate relative to the inline direction of lines (or fields).

This example focuses on a single letter A from FONTA. With PPFA, a single font specified in a page definition can produce letters in any of four rotations. This is accomplished by a FONT command that specifies rotation. If, as in this example, you want to vary the rotation of a font twice within a page, you use two FONT commands, one for each rotation. You also use two LAYOUT commands to map the data to the printout, using the two rotations of the font. In a field processing application, FIELD commands can be used in the same way. These LAYOUT commands name the rotated font in a FONT subcommand.

Figure 56 breaks down the elements required for the FONT commands and subcommands. Distinct local names and rotation specifications for each font are placed in a FONT command. These identify a font as rotated within a page definition. The rotation of a character is relative to the inline direction of a field or

LAYOUT. The characters and rotations shown here assume an inline direction of ACROSS.

Figure 56. Character Rotation



You can use up to 16 possible combinations of logical page direction and font rotation for page printers other than the 3800.

The FONT subcommands within LAYOUT or FIELD commands that name the rotated font in that page definition use only the local name. The following command stream shows the proper specification and nesting of FONT commands and subcommands for rotation.

```
PAGEDEF ABCD ;
  FONT FONTA M103 ;               /*NO ROTATION, LOCAL AND          */
                                  /*USER-ACCESS NAMES.              */
  FONT FONTARTD180 M103           /*ROTATED FONT, LOCAL, USER-ACCESS*/
       ROTATION 180 ;             /*NAMES PLUS ROTATION SUBCOMMAND  */
                                  /*AND PARAMETER.                  */
  LAYOUT 'abc' FONT FONTA ;       /*LOCAL NAME                      */
  LAYOUT 'def' FONT FONTARTD180 ; /*LOCAL NAME                     */
```

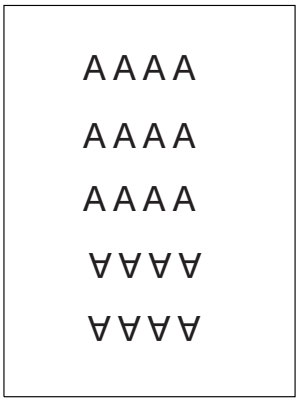


Figure 57. Example of Assumed Data File and Rotation Specifications

FONTA, identified in the first FONT command, requires no rotation parameter because it is printed in the default position (or 0° rotation) for font M103. For the rotated font, the second FONT command identifies FONTARTD180 (the local name) as M103 rotated 180°.

Using Traditional Kanji Formatting

Traditional kanji print presentation, called *tate*, is possible with print server printers, using a combination of font rotation and logical page direction. A logical page in the DOWN direction and a 270° font rotation provide the right combination to present kanji in tate format on an print server printer.

```
FORMDEF TATE
        OFFSET 1 IN 1 IN ;
PAGEDEF TATE
        HEIGHT 5 IN
        WIDTH 6 IN
        DIRECTION DOWN ;
  FONT  KANJIRTD  M104
        ROTATION 270 ;
  LAYOUT 'tate' FONT KANJIRTD;
```

Figure 58 shows the result of formatting with the above page definition. The characters are added to lines down the page. Lines are added right to left.

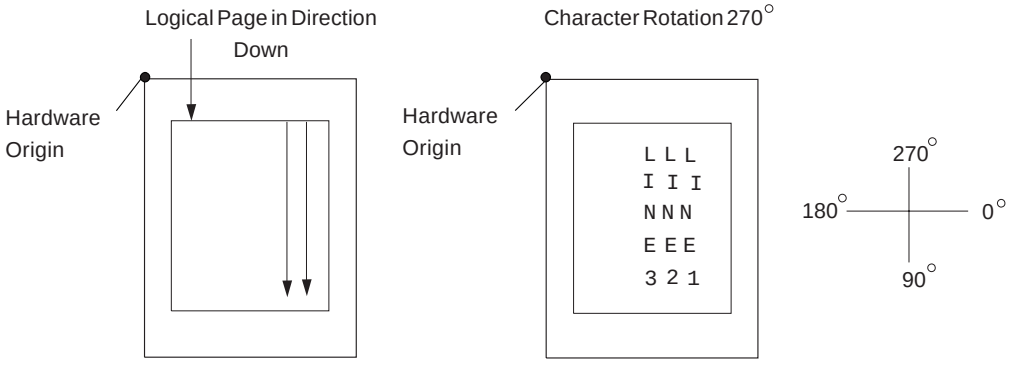


Figure 58. 3820 Tate Presentation

Record Formatting Examples

In order to allow different formats for different groups (or tables) of data, each of which have an unpredictable number of entries, a Record ID is assigned to each output record to identify the type of record and control layout formatting. An application can group data fields that are to be formatted together as an entity into Data Records with a specific Record ID. For example, in a bank statement, the data fields for a check transaction might be grouped together with a Record ID identifying that record as a check transaction. The Pagedef would then define a special layout format for a check transaction with a matching Record ID.

The same thing could be done for a deposit transaction, customer account information, deposit totals, check totals, etc. If the customer account information is going to be used in a page header on each page, the Pagedef can define a special layout format for a customer information record that automatically generates a page header for each page.

This section shows two complete examples using the record formatting process. Each is divided into three parts - the desired output (after PAGEDEF processing), the application output (before PAGEDEF processing), and the PPFA commands.

Note: The source and data for these examples can be located on WEB site http://www.ibm.com/printers.R5PSC.NSF/web/ppfa. Once the WEB site has been displayed, then select the option titled ″PPFA Record Formatting Description″.

Example 1 Desired Output (after PAGEDEF Processing)

The example user data along with the PPFA commands are meant to create this printed output. (The following page has been resized to fit the format of this User's Guide.)

Big Brother Bank

"We watch over you"
P.O. Box 1573
Beantown, MA 02116

Account Number: 026-257311
Statement Begin Date: JAN 02, 1990
Statement End Date: FEB 01, 1990

Chubby Checker
123 Redlight Lane
Twistnshout MA 02345

Super Checking Account Activity

Beginning Balance	Credits	Debits	Service Charge	Ending Balance
$2591.24	$1946.93	$1956.43	$0.00	$2581.72

Credits

Description	Date	Amount
DEPOSIT	01/05/90	$ 26.90
AUTO DEPOSIT	01/15/90	$ 954.27
AUTO DEPOSIT	01/30/90	$ 954.27
INTEREST	01/31/90	$ 11.49
Total Credits		$1946.93

Checks

Check No.	Date	Amount	Check No.	Date	Amount
352	01/04/90	$ 321.50	353	01/05/90	$ 100.00
354	01/10/90	$ 122.30	355	01/11/90	$ 59.95
356	01/15/90	$ 852.33	357	01/30/90	$ 500.35
358	01/15/90	$ 852.33	359	01/30/90	$ 500.35
360	01/15/90	$ 852.33	361	01/30/90	$ 500.35
362	01/15/90	$ 852.33	363	01/30/90	$ 500.35
364	01/15/90	$ 852.33	365	01/30/90	$ 500.35
366	01/15/90	$ 852.33	367	01/30/90	$ 500.35
368	01/15/90	$ 852.33	369	01/30/90	$ 500.35
370	01/15/90	$ 852.33	371	01/30/90	$ 500.35
372	01/15/90	$ 852.33	373	01/30/90	$ 500.35
374	01/15/90	$ 852.33	375	01/30/90	$ 500.35
376	01/15/90	$ 852.33	377	01/30/90	$ 500.35
378	01/15/90	$ 852.33	379	01/30/90	$ 500.35
380	01/15/90	$ 852.33	381	01/30/90	$ 500.35
382	01/15/90	$ 852.33	383	01/30/90	$ 500.35
384	01/15/90	$ 852.33	385	01/30/90	$ 500.35
386	01/15/90	$ 852.33	387	01/30/90	$ 500.35
388	01/15/90	$ 852.33	389	01/30/90	$ 500.35
390	01/15/90	$ 852.33	391	01/30/90	$ 500.35
392	01/15/90	$ 852.33	393	01/30/90	$ 500.35
394	01/15/90	$ 852.33	395	01/30/90	$ 500.35

Page 1

Figure 59. Part one of Sample Graphic Created by the Following User Data and PPFA Commands.

Big Brother Bank
"We watch over you"
P.O. Box 1573
Beantown, MA 02116

Account Number: 026-257311
Statement Begin Date: JAN 02, 1990
Statement End Date: FEB 01, 1990

Chubby Checker
123 Redlight Lane
Twistnshout MA 02345

Checks

Check No.	Date	Amount	Check No.	Date	Amount
396	01/15/90	$ 852.33	397	01/30/90	$ 500.35
398	01/15/90	$ 852.33	399	01/30/90	$ 500.35
400	01/15/90	$ 852.33	401	01/30/90	$ 500.35
402	01/15/90	$ 852.33	403	01/30/90	$ 500.35
404	01/15/90	$ 852.33	405	01/30/90	$ 500.35
406	01/15/90	$ 852.33	407	01/30/90	$ 500.35
408	01/15/90	$ 852.33	409	01/30/90	$ 500.35
410	01/15/90	$ 852.33	411	01/30/90	$ 500.35
412	01/15/90	$ 852.33	413	01/30/90	$ 500.35
414	01/15/90	$ 852.33	415	01/30/90	$ 500.35
416	01/15/90	$ 852.33	417	01/30/90	$ 500.35
418	01/15/90	$ 852.33	419	01/30/90	$ 500.35

Total Checks $1956.43

Daily Balances

Date	Balance	Date	Balance
01/04/90	$2269.74	01/05/90	$2196.64
01/10/90	$2074.34	01/11/90	$2014.39
01/15/90	$2116.33	01/30/90	$2570.25

Final Balance $2581.74

Interest Rate As of 01/04 * * * 5.321%

Page 2

Figure 60. Part two of Sample Graphic Created by the Following User Data and PPFA Commands.

Example 1 Application Output (before PAGEDEF Processing)

Each layout record contains all information for a given layout. Because of lack of space, only the first 80 bytes are shown here. The first 10 characters must contain the layout id.

```
         1111111111222222222233333333334444444444555555555566666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890
statmid   026-257311Chubby Checker  123 Redlight Lane  Twistnshout   MA    02345
statsum   $2591.24 $1946.93  $1956.43  $0.00  $2581.72
pgenum
crheader
crdata    DEPOSIT      01/05/90  $  26.90
crdata    AUTO DEPOSIT 01/15/90  $ 954.27
crdata    AUTO DEPOSIT 01/30/90  $ 954.27
crdata    INTEREST     01/31/90  $  11.49
crtotal                          $1946.93
ckheader
ckdatal    352         01/04/90  $ 321.50
ckdatar    353         01/05/90  $ 100.00
ckdatal    354         01/10/90  $ 122.30
ckdatar    355         01/11/90  $  59.95
ckdatal    356         01/15/90  $ 852.33
ckdatar    357         01/30/90  $ 500.35
ckdatal    358         01/15/90  $ 852.33
ckdatar    359         01/30/90  $ 500.35
ckdatal    360         01/15/90  $ 852.33
ckdatar    361         01/30/90  $ 500.35
ckdatal    362         01/15/90  $ 852.33
ckdatar    363         01/30/90  $ 500.35
ckdatal    364         01/15/90  $ 852.33
ckdatar    365         01/30/90  $ 500.35
ckdatal    366         01/15/90  $ 852.33
ckdatar    367         01/30/90  $ 500.35
ckdatal    368         01/15/90  $ 852.33
ckdatar    369         01/30/90  $ 500.35
ckdatal    370         01/15/90  $ 852.33
ckdatar    371         01/30/90  $ 500.35
ckdatal    372         01/15/90  $ 852.33
ckdatar    373         01/30/90  $ 500.35
ckdatal    374         01/15/90  $ 852.33
ckdatar    375         01/30/90  $ 500.35
ckdatal    376         01/15/90  $ 852.33
ckdatar    377         01/30/90  $ 500.35
ckdatal    378         01/15/90  $ 852.33
ckdatar    379         01/30/90  $ 500.35
ckdatal    380         01/15/90  $ 852.33
ckdatar    381         01/30/90  $ 500.35
ckdatal    382         01/15/90  $ 852.33
ckdatar    383         01/30/90  $ 500.35
ckdatal    384         01/15/90  $ 852.33
ckdatar    385         01/30/90  $ 500.35
ckdatal    386         01/15/90  $ 852.33
ckdatar    387         01/30/90  $ 500.35
ckdatal    388         01/15/90  $ 852.33
ckdatar    389         01/30/90  $ 500.35
ckdatal    390         01/15/90  $ 852.33
ckdatar    391         01/30/90  $ 500.35
ckdatal    392         01/15/90  $ 852.33
ckdatar    393         01/30/90  $ 500.35
ckdatal    394         01/15/90  $ 852.33
ckdatar    395         01/30/90  $ 500.35
ckdatal    396         01/15/90  $ 852.33
ckdatar    397         01/30/90  $ 500.35
ckdatal    398         01/15/90  $ 852.33
ckdatar    399         01/30/90  $ 500.35
ckdatal    400         01/15/90  $ 852.33
```

```
ckdatar    401         01/30/90  $ 500.35
ckdatal    402         01/15/90  $ 852.33
ckdatar    403         01/30/90  $ 500.35
ckdatal    404         01/15/90  $ 852.33
ckdatar    405         01/30/90  $ 500.35
ckdatal    406         01/15/90  $ 852.33
ckdatar    407         01/30/90  $ 500.35
ckdatal    408         01/15/90  $ 852.33
ckdatar    409         01/30/90  $ 500.35
ckdatal    410         01/15/90  $ 852.33
ckdatar    411         01/30/90  $ 500.35
ckdatal    412         01/15/90  $ 852.33
ckdatar    413         01/30/90  $ 500.35
ckdatal    414         01/15/90  $ 852.33
ckdatar    415         01/30/90  $ 500.35
ckdatal    416         01/15/90  $ 852.33
ckdatar    417         01/30/90  $ 500.35
ckdatal    418         01/15/90  $ 852.33
ckdatar    419         01/30/90  $ 500.35
cktotal                          $1956.43
balhead
baldatal               01/04/90  $2269.74
baldatar               01/11/90  $2014.39
baldatal               01/05/90  $2196.64
baldatar               01/15/90  $2116.33
baldatal               01/10/90  $2074.34
baldatar               01/30/90  $2570.25
baltotal                         $2581.74
statrail
```

Example 1 PPFA Commands

```
PAGEDEF chubby replace yes
          WIDTH 8.5 in
          HEIGHT 11.0 in;
    FONT  comp  a075nc ;    /*Big Brother Bank font  */
    FONT  ital  a175dc ;    /*Italic theme           */
    FONT  addr  a075dc ;    /*Big Brother address    */
    FONT  varb  gt10   ;    /*Variable data          */
    FONT  super a075dc ;    /*Super Checking Account */
    FONT  head   a055ac;    /*Headings               */
    FONT  bhead  a075ac;    /*Bold Headings          */

PAGEFORMAT  chub1   TOPMARGIN 2 in BOTMARGIN  2 in;
/*********************************************/
/** statmid  BODY                           **/
/*********************************************/
LAYOUT  C'statmid'  PAGEHEADER NEWPAGE
POSITION .6 in ABSOLUTE .55 in;
  FIELD  TEXT C'Big Brother Bank'  ALIGN LEFT
FONT comp ;   /* default to LAYOUT positioning*/
FIELD  TEXT C'"We watch over you"' ALIGN LEFT
POSITION   0 NEXT
        FONT ital ; /*default to next line          */
FIELD  TEXT C'P.O. Box 1573' ALIGN LEFT
POSITION   0 NEXT
       FONT addr ; /*default to next line          */
FIELD  TEXT C'Beantown, MA  02116' ALIGN LEFT
POSITION   0 NEXT
FONT addr ; /*default to next line          */
   FIELD  TEXT C'Account Number:' ALIGN LEFT
POSITION   4.3 in .2 in
FONT head ; /*New area on right             */
FIELD  TEXT C'Statement Begin Date:' ALIGN LEFT
         POSITION  4.3 in  NEXT
FONT head ; /*New area on right             */
FIELD  TEXT C'Statement End Date:' ALIGN LEFT
POSITION  4.3 in  NEXT
FONT head ; /*New area on right             */
FIELD  START  1 LENGTH 10 ALIGN RIGHT
POSITION  7.5 in .2 in
              FONT varb ; /*variable - account number*/
FIELD  START 75 LENGTH 12
POSITION  7.5 in  NEXT
ALIGN RIGHT /* data is missing from example */
FONT varb ; /*variable - begin date         */
FIELD  START 88 LENGTH 12
POSITION  7.5 in  NEXT
    ALIGN RIGHT /* data is missing from example */
FONT varb ; /*variable - end date           */
FIELD  START 11 LENGTH 19 ALIGN LEFT
POSITION  1.1 in  .9 in
              FONT varb ; /*variable - customer name */
   FIELD  START 30 LENGTH 19          ALIGN LEFT
POSITION  1.1 in  NEXT
FONT varb ; /*variable - customer address   */
FIELD  START 49 LENGTH 22          ALIGN LEFT
         POSITION  1.1 in  NEXT
FONT varb ; /*variable - customer city, st. */

/*********************************************/
/** statsum  BODY                           **/
/*********************************************/
LAYOUT C'statsum'  BODY
POSITION .6 in .5 in;
FIELD  TEXT C'Super Checking Account Activity'
FONT super ; /* Static text - Super Checking */
```

```
DRAWGRAPHIC LINE  ACROSS 7.5 IN LINEWT BOLD
POSITION  0  .15 in
copy   down 2 spaced 1 mm;
FIELD  TEXT C'Beginning Balance'
POSITION  .3 in .4 in
FONT head  ; /* Static text - first header   */
FIELD  TEXT C'Credits'
POSITION 2.4 in CURRENT
FONT head  ; /* Static text - first header   */
FIELD  TEXT C'Debits'
POSITION 3.6 in CURRENT
   FONT head  ; /* Static text - first header   */
   FIELD  TEXT C'Service Charge'
POSITION 4.8 in CURRENT
       FONT head  ; /* Static text - first header   */
   FIELD  TEXT C'Ending Balance'
POSITION 6.3 in CURRENT
    FONT head  ; /* Static text - first header   */
   FIELD  START  1  LENGTH  8
POSITION  .6 in .6 in
       FONT varb  ; /* Variable text - Beg balance  */
  FIELD  START 10  LENGTH  8
POSITION 2.2 in CURRENT
     FONT varb  ; /* Variable text - Credits      */
   FIELD  START 20  LENGTH  8
POSITION 3.4 in CURRENT
       FONT varb  ; /* Variable text - Debits       */
   FIELD  START 30  LENGTH  5
POSITION 5.0 in CURRENT
      FONT varb  ; /* Variable text - Service Chrg */
   FIELD  START 40  LENGTH  8
POSITION 6.5 in CURRENT
       FONT varb  ; /* Variable text - End Balance  */
   DRAWGRAPHIC LINE  ACROSS 7.5 IN LINEWT BOLD
POSITION  0 .7 in;

/*********************************************/
/** crheader GROUPHEADER                     **/
/*********************************************/
LAYOUT C'crheader'  GRPHEADER XSPACE .2 in
POSITION SAME .9 in;
FIELD  TEXT C'Credits'
FONT bhead ; /* Static text - Credits        */
FIELD  TEXT C'Description'
         POSITION  1.3 in   CURRENT
FONT head  ; /* Stat text - Deposit Descr.   */
FIELD  TEXT C'Date'
    POSITION  3.2 in   CURRENT
FONT  head ; /* Static text - Date           */
FIELD  TEXT C'Amount'
        POSITION  5.0 in   CURRENT
FONT  head ; /* Stat text - Amount of deposit*/
DRAWGRAPHIC LINE  ACROSS 6.2 IN LINEWT BOLD
POSITION 1.3 in next;

/*********************************************/
/** crdata   BODY                            **/
/*********************************************/
LAYOUT C'crdata'   BODY  GROUP;
FIELD  START  1 LENGTH 13
      POSITION  1.3 in  CURRENT
FONT  varb ; /* Variable text - Description  */
FIELD  START 14 LENGTH 8
       POSITION  3 in  CURRENT
FONT  varb ; /* Variable text - Date         */
FIELD  START 24 LENGTH 8     ALIGN RIGHT
POSITION  5.6 in  CURRENT
```

```
   FONT  varb  ; /* Variable text - Amount       */

/***********************************************/
/** crtotal  BODY                              **/
/***********************************************/
LAYOUT C'crtotal'  BODY  GROUP;
FIELD  TEXT C'Total Credits'
       POSITION  1.5 in   .2 in
FONT bhead ; /* Stat text - Total credits    */
FIELD  START 24 LENGTH 8   ALIGN RIGHT
POSITION  7.3 in  CURRENT
   FONT  varb  ; /* Variable text - Amount      */
DRAWGRAPHIC LINE ACROSS 7.5 IN LINEWT BOLD
POSITION   0  next;

/***********************************************/
/**  ckheader  GROUPHEADER                    **/
/***********************************************/
LAYOUT C'ckheader' GRPHEADER XSPACE .2 in
POSITION  SAME .6 in;
FIELD  TEXT C'Checks'
FONT bhead ; /* Static text - Checks         */
FIELD  TEXT C'Check No.'
         POSITION  1.4 in   CURRENT
FONT  head ; /* Stat text - Check number     */
FIELD  TEXT C'Date'
       POSITION  2.5 in   CURRENT
FONT  head  ;/* Stat text - Date of check    */
FIELD  TEXT C'Amount'
       POSITION  3.5 in   CURRENT
FONT  head  ;/* Static text - Amount of check*/
FIELD  TEXT C'Check No.'
        POSITION  4.6 in   CURRENT
FONT  head ; /* Stat text - Check number     */
FIELD  TEXT C'Date'
        POSITION  5.6 in   CURRENT
FONT  head  ;/* Stat text - Date of check    */
FIELD  TEXT C'Amount'
        POSITION  6.8 in   CURRENT
FONT  head  ;/* Static text - Amount of check*/
DRAWGRAPHIC LINE  ACROSS 6.2 IN LINEWT BOLD
POSITION 1.3 in next;
DRAWGRAPHIC LINE  DOWN LINETYPE shortdash
POSITION 4.5 in CPOS;

/***********************************************/
/**  ckdatal BODY left side                   **/
/***********************************************/
LAYOUT C'ckdatal'  BODY  GROUP
POSITION SAME NEXT;
FIELD  START  2 LENGTH 3
POSITION 1.4 in  CURRENT
        FONT  varb  ; /* Variable text - Check number */
   FIELD  START 14 LENGTH 8
POSITION  2.4 in  CURRENT
        FONT  varb  ; /* Variable text - Date         */
   FIELD  START 24 LENGTH 8  ALIGN RIGHT
POSITION  4.4 in   CURRENT
FONT  varb  ; /* Variable text - Amount */

/***********************************************/
/**  ckdatar BODY right side                  **/
/***********************************************/
LAYOUT C'ckdatar'  BODY  GROUP
POSITION SAME SAME;
   FIELD  START  2  LENGTH 3
POSITION  4.6 in  CURRENT
```

```
      FONT  varb  ; /* Variable text - Check number */
   FIELD  START 14 LENGTH 8
POSITION  5.6 in  CURRENT
       FONT  varb  ; /* Variable text - Date         */
   FIELD  START 24 LENGTH 8   ALIGN RIGHT
POSITION 7.5 in  CURRENT
FONT  varb  ; /* Variable text - Amount       */

/*********************************************/
/** cktotal  BODY                            **/
/*********************************************/
LAYOUT C'cktotal'  BODY  GROUP;
ENDGRAPHIC LPOS;  /*ends dashed line between checks */
FIELD  TEXT C'Total Checks'
        POSITION  1.5 in   .2 in
FONT bhead ; /* Stat text - Total checks      */
FIELD  START 24 LENGTH 8  ALIGN RIGHT
POSITION  7.3 in  CURRENT
   FONT  varb  ; /* Variable text - Amount       */
DRAWGRAPHIC LINE ACROSS 7.5 IN LINEWT BOLD
POSITION   0   next;

/*********************************************/
/** balhead  GROUPHEADER                     **/
/*********************************************/
LAYOUT C'balhead'   GRPHEADER XSPACE .2 in
POSITION SAME .6 in;
FIELD  TEXT C'Daily'
FONT bhead ; /* Static text - Daily Balance  */
FIELD  TEXT C'Date'
        POSITION  1.3 in   CURRENT
FONT  head  ;/* Stat text - Date of balance  */
FIELD  TEXT C'Balance'
       POSITION  2.8 in   CURRENT
FONT  head  ;/* Static text - Balance        */
FIELD  TEXT C'Date'
         POSITION  4.3 in   CURRENT
FONT  head  ; / Stat text - Date of balance  */
FIELD  TEXT C'Balance'
       POSITION  5.8 in   CURRENT
FONT  head  ; /*Static text - Balance        */
FIELD  TEXT C'Balances'
       POSITION  0  NEXT
FONT bhead  ; /*Static text - Daily Balance  */
DRAWGRAPHIC LINE  ACROSS 6.2 IN LINEWT BOLD
       POSITION 1.3 in CPOS;

/*********************************************/
/** baldatal BODY  left side                 **/
/*********************************************/
LAYOUT C'baldatal'  BODY  GROUP
POSITION SAME  NEXT;
FIELD  START 14 LENGTH 8
POSITION 1.3 in  CURRENT
       FONT  varb  ; /* Variable text - Date        */
   FIELD  START 24 LENGTH 8    ALIGN RIGHT
POSITION 3.6 in  CURRENT
FONT  varb  ; /* Variable text - Amount      */

/*********************************************/
/** baldatar BODY  right side                **/
/*********************************************/
LAYOUT C'baldatar'  BODY  GROUP
POSITION SAME  SAME;
  FIELD  START 14 LENGTH 8
POSITION 4.3 in  CURRENT
         FONT  varb  ; /* Variable text - Date        */
```

```
   FIELD  START 24 LENGTH 8   ALIGN RIGHT
POSITION 6.6 in  CURRENT
FONT  varb  ; /* Variable text - Amount      */

/*********************************************/
/** baltotal BODY                            **/
/*********************************************/
LAYOUT C'baltotal'  BODY  GROUP;
FIELD  TEXT C'Final Balance'
        POSITION  1.5 in   .2 in
FONT bhead ; /* Stat text - Final balance     */
FIELD  START 24 LENGTH 8    ALIGN RIGHT
POSITION  7.3 IN CURRENT
  FONT  varb  ; /* Variable text - Amount       */

/*********************************************/
/** statrail BODY                            **/
/*********************************************/
LAYOUT C'statrail'  BODY
POSITION SAME .4 in;
DRAWGRAPHIC LINE  ACROSS 7.5 IN LINEWT BOLD
POSITION 0 CPOS;
FIELD TEXT C'Interest Rate '
POSITION  2.0 in NEXT
        FONT bhead ; /* Static text - Interest rate   */
FIELD  TEXT C'As of 01/04  * * *  5.321%'
POSITION  CURRENT CURRENT
        FONT varb ;  /* Static text                 */
DRAWGRAPHIC LINE  ACROSS 7.5 IN LINEWT BOLD
POSITION 0  NEXT
 copy down 2 spaced 1 mm;

/*********************************************/
/** pgenum   PAGE NUMBER                     **/
/*********************************************/
LAYOUT C'pgenum'  PAGETRAILER
POSITION SAME ABSOLUTE 10.7 in;
FIELD  TEXT C 'Page '
POSITION 6.5 in CURRENT
        FONT  varb;/* placement of page number      */
   FIELD  PAGENUM PRINT RESET 1 /* request page numbering*/
      FONT  varb /* placement of page number      */
POSITION CURRENT CURRENT;
```

Example 2 Using Repeated and Unended Boxes

This example shows how to use the repeated box option, a single circle and some un-ended boxes. (The following example has been resized to fit the format of this User's Guide.)

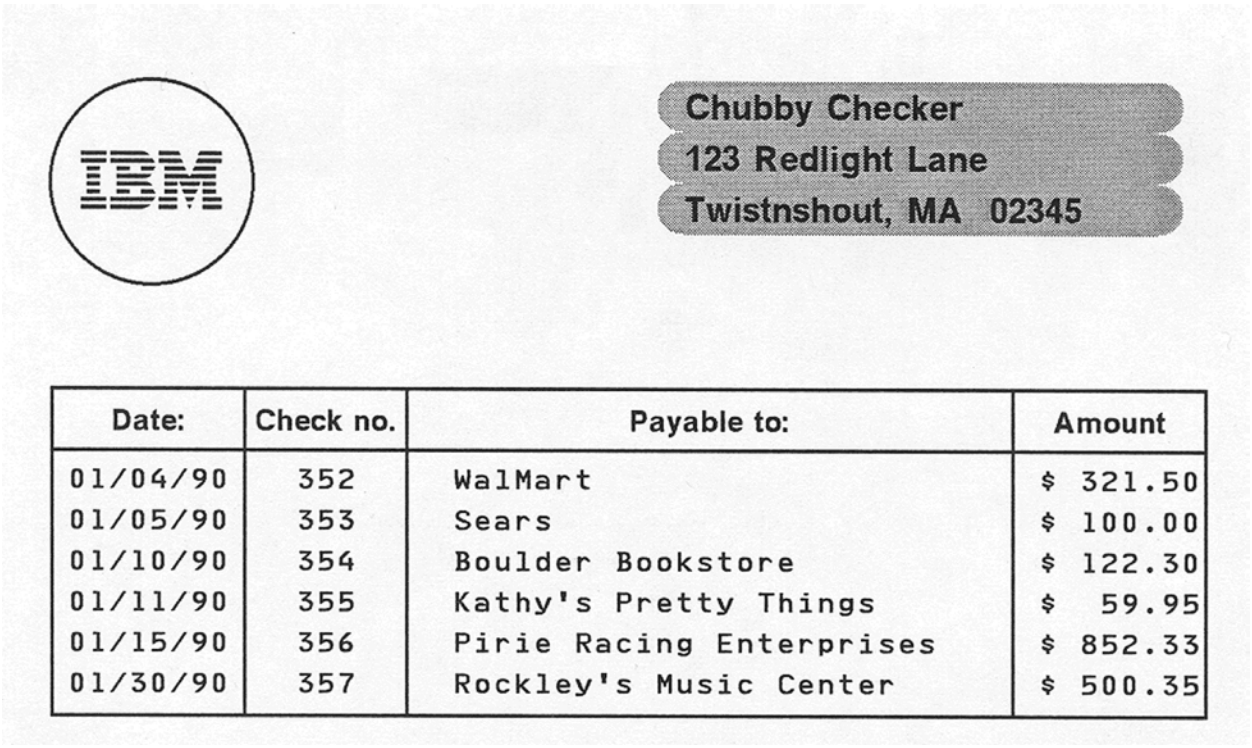


Figure 61. Example Showing How to Use the Repeating Box Option

Example 2 Application Output (before PAGEDEF Processing)

```
         1111111111222222222233333333334444444444555555555566666666677
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901

statmid    Chubby Checker     123 Redlight Lane  Twistnshout   MA 02345
ckheader
ckdata     352         01/04/90  $ 321.50   WalMart
ckdata     353         01/05/90  $ 100.00   Sears
ckdata     354         01/10/90  $ 122.30   Boulder Bookstore
ckdata     355         01/11/90  $  59.95   Kathy's Pretty Things
ckdata     356         01/15/90  $ 852.33   Pirie Racing Enterprises
ckdata     357         01/30/90  $ 500.35   Rockley's Music Center
ckend
```

PPFA Input for Repeated Boxes Example 2

```
PAGEDEF rept1   replace yes;
    FONT  addr  a075dc ;          /*customer address       */
    FONT  varb  gt10   ;          /*Variable data          */
    FONT  bhead  a075ac;          /*Bold Headings          */
    SETUNITS  LINESP .25 in ;     /* Line spacing          */

    PAGEFORMAT  rept1   BOTMARGIN  2 in;
    SEGMENT  ibmlog;              /*IBM logo               */
   /*********************************************/
   /** statmid  PAGEHEADER                     **/
   /*********************************************/
    LAYOUT C'statmid'
```

```
        SEGMENT ibmlog  1.15 in 1.35 in
        PAGEHEADER NEWPAGE
         POSITION SAME ABSOLUTE NEXT;
   DRAWGRAPHIC  CIRCLE RADIUS .5 in         /* 1 inch circle    */
        POSITION 1.5 in  1.5 in;
   DRAWGRAPHIC  BOX  BOXSIZE 2.6 IN .25 IN ROUNDED LARGE
        LINEWT 0                            /* invisible border */
        POSITION 4 IN 1 IN
        COPY   DOWN 2 SPACED 0
        FILL ALL  DOT02;
   FIELD  START  2 LENGTH 19  ALIGN LEFT
        POSITION  4.2 in 1.2 in
        FONT addr ;    /*variable - customer name      */
   FIELD  START 21 LENGTH 19  ALIGN LEFT
        POSITION  4.2 in NEXT
        FONT addr ;    /*variable - customer address   */
   FIELD  START 40 LENGTH 22  ALIGN LEFT
        POSITION  4.2 in NEXT
         FONT addr ;    /*variable - customer city, st. */
/***********************************************/
/**  ckheader  GROUPHEADER                     **/
/***********************************************/
 LAYOUT C'ckheader'  GRPHEADER XSPACE .25 in
        POSITION 1 in ABSOLUTE 2.5 in;       /* set position   */
   DRAWGRAPHIC  BOX BOXSIZE .95 IN  .3 IN
        POSITION 0 0;
   DRAWGRAPHIC  BOX BOXSIZE .95 IN        /* box started for data */
        POSITION 0   .3 in;               /* no vertical size */
   FIELD  TEXT C'Date'
        POSITION   .3 in   .2 in
      FONT  bhead  ; /* Stat text - Date of check    */
   DRAWGRAPHIC  BOX  BOXSIZE  .8 IN  .3 IN
        POSITION  .95 IN 0;
   DRAWGRAPHIC  BOX  BOXSIZE .8 IN        /* box started for data */
        POSITION .95 in .3 in;            /* no vertical size     */
   FIELD  TEXT C'Check No.'
        POSITION 1 in   .2 in
        FONT  bhead  ; /* Stat text - Check number     */
   DRAWGRAPHIC  BOX  BOXSIZE  3 IN  .3 IN
        POSITION 1.75 IN  0;
   DRAWGRAPHIC  BOX  BOXSIZE 3 IN         /* box started for data */
        POSITION 1.75 in  .3 in;          /* no vertical size     */
   FIELD  TEXT C'Payable to:'
       POSITION  2.9 in    .2 in
        FONT  bhead  ; /* Static text - Payable to:    */
   DRAWGRAPHIC  BOX BOXSIZE .95 IN  .3 IN
        POSITION 4.75 IN   0 in;
   DRAWGRAPHIC  BOX  BOXSIZE .95 in       /* box started for data */
        POSITION 4.75 in  .3 in;          /* no vertical size     */
   FIELD  TEXT C'Amount'
        POSITION  5 in    .2 in
        FONT  bhead  ; /* Stat text - Amount of check  */
/***********************************************/
/**  ckdata  BODY  w/ un-ended boxes           **/
/***********************************************/
 LAYOUT C'ckdata'  BODY  GROUP;
   FIELD  START  2 LENGTH 3  ALIGN LEFT
       POSITION 1.2 in  CURRENT
        FONT  varb  ; /* Variable text - Check number */
   FIELD  START 14 LENGTH 8  ALIGN LEFT
       POSITION  .1 in  CURRENT
       FONT  varb  ; /* Variable text - Date         */
   FIELD  START 35 LENGTH 25 ALIGN LEFT
        POSITION  2.0 in   CURRENT
         FONT  varb  ; /* Variable text - Payable to:  */
   FIELD  START 24 LENGTH 8  ALIGN RIGHT
        POSITION  5.6 in   CURRENT
```

```
        FONT  varb  ; /* Variable text - Amount       */
/*********************************************/
/**  ckend   BODY  to end boxes               **/
/*********************************************/
 LAYOUT C'ckend' BODY  GROUP;  /* If this layout and command are  */
   ENDGRAPHIC LPOS;            /* not issued, the boxes should be */
                               /* closed anyway. But if there was */
                               /* a trailer, they may not end in  */
                               /* the right place.                */
```

Chapter 5. Creating Complex Printouts

You are now ready to learn about some formatting tasks that might apply to more complex printouts. The basic form definition and page definition elements have been covered. This chapter describes how these elements are combined to create complete print jobs.

The advanced techniques covered in this section are illustrated in the following examples:

Table 5. Form Definitions and Page Definition Tasks

Tasks	**Example location**
Field Processing with Overlay	"Combining Field Processing and an Electronic Overlay" on page 93
Suppressing Data	"Using Suppressions to Vary Data Presentation" on page 95
Including Fixed Text	"Incorporating Fixed Text into a Page Definition" on page 96
Combining Two Reports	"Combining Two Reports into One Printout" on page 99

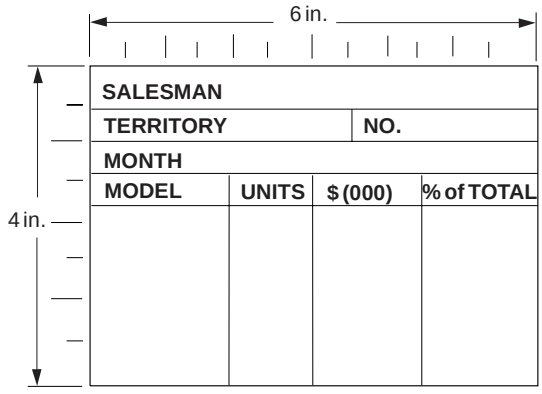
The examples in this chapter build on a single sales application, showing different sales reports being formatted by form definitions and page definitions.

Combining Field Processing and an Electronic Overlay

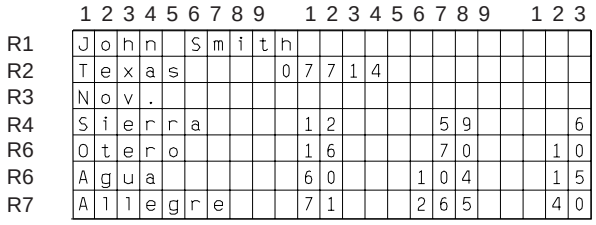
This example involves printing a monthly individual sales report for a specified distribution. The following items are needed to generate the sales report:

- A pre-designed electronic overlay for the sales report
- An unformatted print data file with periodic sales statistics

An example of these is shown in Figure 62 on page 94.



Overlay



Data File

Figure 62. Electronic Overlay and Data File for a Sales Report

The code example that follows contains a form definition and a page definition. The page definition maps the file to the overlay.

In Figure 62 the 0,0 point is the upper-left corner of the overlay. This means that the logical page origin must coincide with the overlay origin in this example. POSITION subcommands are relative to the logical page origin. The overlay origin point that positions the overlay is specified in the Overlay Generation Language/370 that creates the overlay, but can be modified in the page definition. In mapping to an overlay, you should check the input to the overlay creation program so you can coordinate its origin with the logical page origin. You can reposition the overlay through the PRINTLINE command.

```
01 FORMDEF SLSRPT OFFSET 0 0 ;
02   OVERLAY SLSRPT ;
03     SUBGROUP OVERLAY SLSRPT ;
04
05 PAGEDEF SLSRPT ;
06   PRINTLINE  POSITION 2 IN 1.3 IN ;   /* RECORD 1           */
07    FIELD START 1 LENGTH 23  ;
08   PRINTLINE  POSITION 2 IN 1.70 IN ;  /* RECORD 2           */
09     FIELD START 1 LENGTH 9 ;          /* DEFAULT POSITION   */
10     FIELD START 10 LENGTH 5
11           POSITION 4.3 IN  * ;        /* THE ASTERISK MEANS */
12                                       /* CURRENT LINE       */
13   PRINTLINE POSITION 1.5 IN 6 IN ;    /* RECORD 3           */
14     FIELD START 1 LENGTH 4 ;
15  SETUNITS LINESP 4 LPI ;
16   PRINTLINE  REPEAT 4                 /* RECORDS 4-7        */
17              POSITION 1.5 IN  3.6 IN ;
18   FIELD START 1 LENGTH 7 ;            /* DEFAULT POSITION   */
19   FIELD START 10 LENGTH 3
```

```
20         POSITION 1.5 IN * ;
21   FIELD START 16 LENGTH 3
22         POSITION 2.5 IN * ;
23   FIELD START 21 LENGTH 3
24         POSITION 3.5 IN * ;
```

A time-saving device used in the above example is the REPEAT subcommand (line 16), which maps a single printline with its field subsets to records 4 through 7 with all model names and sales statistics. The length values in the repeated fields are 7, 3, 3, and 3—sufficient to accommodate the largest model name, unit value, $(000), and percentage fields mapped by this FIELD command.

Figure 63 shows the report formatted by the resources generated in the command stream of this example.

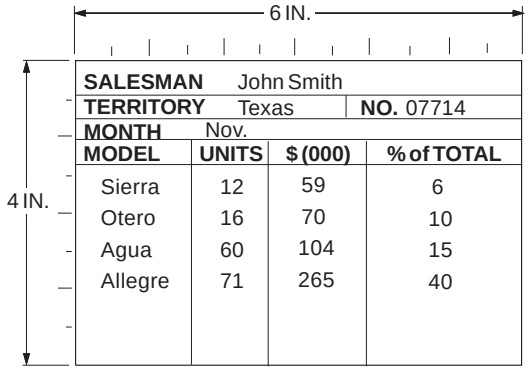


Figure 63. Sales Report

Using Suppressions to Vary Data Presentation

PPFA and the print server printers enable you to produce variations of the same report in a single job. The essential function for this capability is called *suppression*. Suppression involves the coordinated specification of elements in both the page definition and the form definition. You create a suppression in the page definition and turn it on or off in a subgroup within a form definition.

This example shows how to alter the controls in the previous example (“Combining Field Processing and an Electronic Overlay” on page 93) in order to generate a second report along with the one already created.

First, change the page definition by adding a SUPPRESSION subcommand to the third field in the repeated printline—the printline that mapped the models and sales figures in “Combining Field Processing and an Electronic Overlay” on page 93. The suppression is, in effect, created by the SUPPRESSION subcommand in the FIELD command. The following example shows the addition at line 23.

```
18   FIELD START 1 LENGTH 7 ;
19   FIELD START 10 LENGTH 3
20         POSITION 1.5 IN * ;
21   FIELD START 16 LENGTH 3
22         POSITION 2.5 IN  *
23         SUPPRESSION SALES ;       /*ADDED LINE     */
24   FIELD START 21 LENGTH 3
25         POSITION 3.5 IN * ;
```

The SUPPRESSION subcommand creates the potential for selective suppression of the data in the “$(000)” field of the report.

Then, rewrite the form definition, creating two subgroups within the copy group. Next, write a SUPPRESSION command immediately after the FORMDEF command. Finally, place a SUPPRESSION subcommand in the subgroup in which you want the data suppressed. This names the suppression. The resulting form definition command stream is as follows:

```
FORMDEF SECRPT ;
  SUPPRESSION SALES ;             /*NAMING THE SUPPRESSION      */
  COPYGROUP SECRPT ;
    OVERLAY SLSRPT ;              /*NAMING THE OVERLAY          */
    SUBGROUP COPIES 1
             OVERLAY SLSRPT  ;
    SUBGROUP COPIES 1
             OVERLAY SLSRPT
             SUPPRESSION SALES ;  /*TURNING ON THE SUPPRESSION  */
```

The result is shown in Figure 64. The second subgroup creates the second output page of the same data with a second set of modifications; in this case, *modifications* means a suppression that is not in the first subgroup.

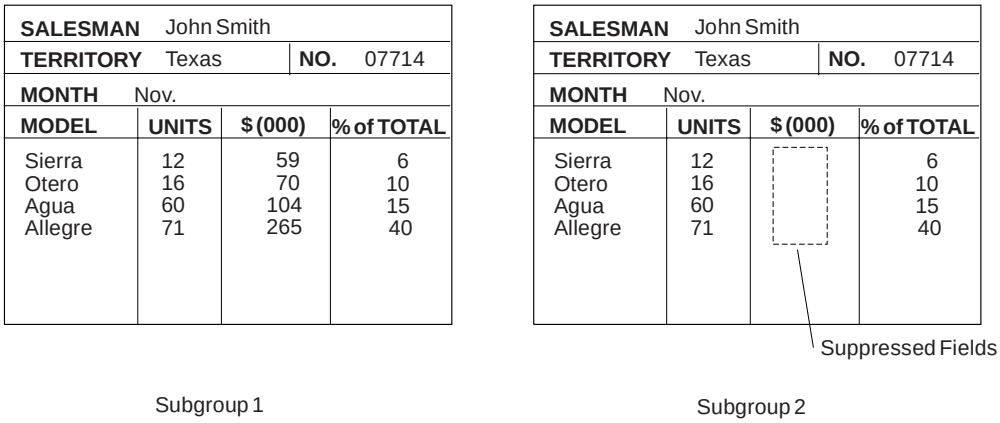


Figure 64. Selective Suppression

Review the steps in this example. To suppress a field, identify the field as *suppressible* in the page definition under the FIELD command in question. Then create a subgroup, activating this suppression with a SUPPRESSION subcommand in the form definition.

The first subgroup produces an output identical to the report in "Combining Field Processing and an Electronic Overlay" on page 93. It contains no suppression.

Note: This example can only be printed simplex.

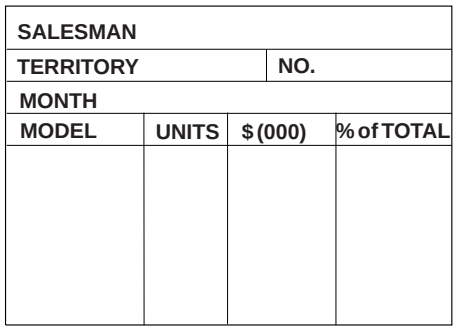
Incorporating Fixed Text into a Page Definition

Fixed text can be incorporated into an electronic overlay through the use of programs, such as Overlay Generation Language/370. Having another place (the page definition) to incorporate fixed text permits you to format documents more efficiently.

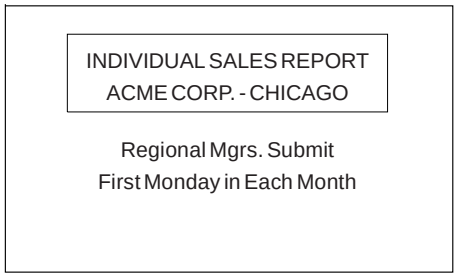
In "Combining Field Processing and an Electronic Overlay" on page 93, a territory sales report for salesman John Smith is created. Here, the territory sales report is incorporated into a larger format going to ACME's corporate headquarters in Chicago. Therefore, the identification for the region needs to appear on the report form. An overlay is used as a header for the composite report. This means that two

overlays appear in the command stream: one carries over from “Combining Field Processing and an Electronic Overlay” on page 93 and the other is the header.

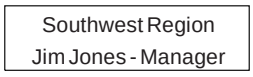
So, as shown in Figure 65, three fixed inputs generate the final report: overlay SLSRPT, overlay HDR, and the fixed regional identification text. (It is the second item that is worked into the page definition in this example.)



Overlay SLSRPT



Overlay HDR



Fixed Text

Figure 65. Input for the Corporate Version of an Individual Sales Report

The data file used to generate this report is the same as the one shown in Figure 62 on page 94.

```
FORMDEF CORP ;
  OVERLAY SLSRPT ;
  OVERLAY HDR ;
  SUBGROUP OVERLAY SLSRPT HDR ;
PAGEDEF CORP
        WIDTH 6 IN
        HEIGHT 7 IN ;
 PRINTLINE POSITION 1.9 IN 2.5 IN ;         /*RECORD 1             */
  FIELD TEXT C 'SOUTHWEST REGION' ;         /*DEFAULT FIELD TEXT   */
                                            /*POSITION             */
  FIELD POSITION -.2 IN NEXT                /*NOTE NEGATIVE VALUE  */
        TEXT 1 C 'JIM JONES - MANAGER' ;
  FIELD START 1 LENGTH 23
        POSITION .1 IN .8 IN ;
 PRINTLINE POSITION 2 IN 3.7 IN ;           /*RECORD 2             */
  FIELD START 1 LENGTH 9 ;                  /*DEFAULT FIELD        */
                                            /*POSITION             */
  FIELD START 10 LENGTH 5
```

```
       POSITION 2.5  IN *  ;
 PRINTLINE POSITION 1.5 IN  4 IN ;          /*RECORD 3             */
  FIELD START 1 LENGTH 4 ;
SETUNITS LINESP 4 LPI ;
 PRINTLINE REPEAT 4                         /*RECORDS 4-7          */
          POSITION .4 IN  4.7 IN ;
  FIELD START 1 LENGTH 7 ;                  /*DEFAULT FIELD        */
                                            /*POSITION             */
  FIELD START 10 LENGTH 3
        POSITION 1.6 IN * ;
  FIELD START 16 LENGTH 3
        POSITION 2.9 IN * ;
  FIELD START 21 LENGTH 3
        POSITION 4.3 IN * ;
```

In the above command stream, the same basic commands from “Combining Field Processing and an Electronic Overlay” on page 93 are used, although the positions of fields have been changed to accommodate the new layout.

New FIELD commands with TEXT subcommands have been inserted in the first PRINTLINE command to produce the regional text, which is positioned at the bottom of the header form. The 1 is a duplication parameter indicating how many times the fixed text is to be repeated. The C can precede single-byte characters such as those used, for example, to write English or German. Both 1 and C are the default values for a TEXT subcommand. The text you want inserted appears between single quotation marks. Observe how the POSITION subcommands change to accommodate both fixed text and record-1 text.

Note: Each PRINTLINE command in your PPFA command stream should have a corresponding record in the input data file. If you specify a fixed-text data field and an input data field under the same PRINTLINE command, they will both be associated with the same input data file record. However, if all the FIELD commands under a PRINTLINE command specify fixed text, the corresponding input record will simply be discarded. In that case, you should insert a blank record into the input data file to preserve the correct relationship between records and PRINTLINE commands.

Figure 66 on page 99 shows how the finished output looks.

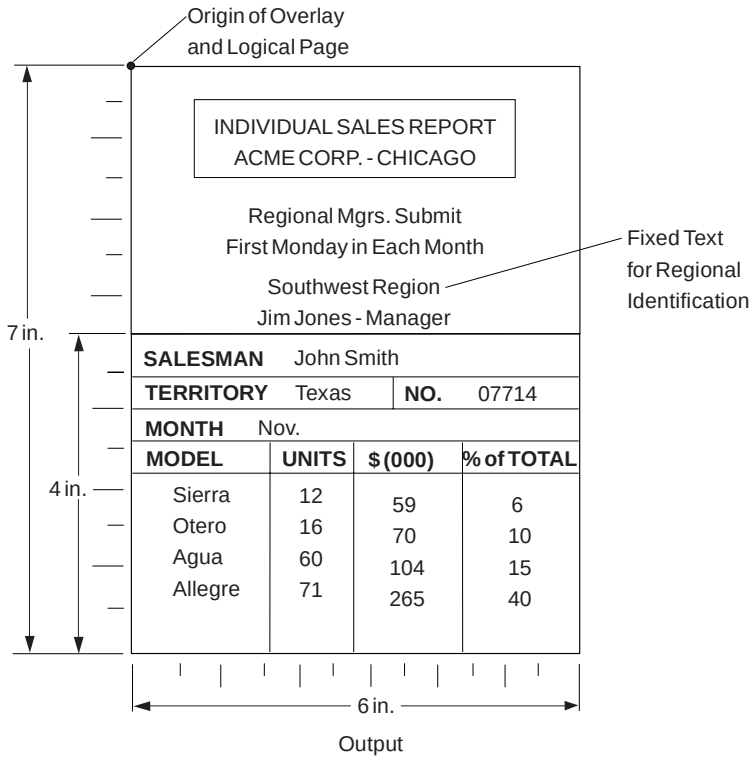


Figure 66. The Corporate Version of the Sales Report with Fixed Text

Combining Two Reports into One Printout

This example combines two data files and two page layouts into one printout, also building on “Combining Field Processing and an Electronic Overlay” on page 93.

Figure 67 on page 100 shows the new data and a new overlay.

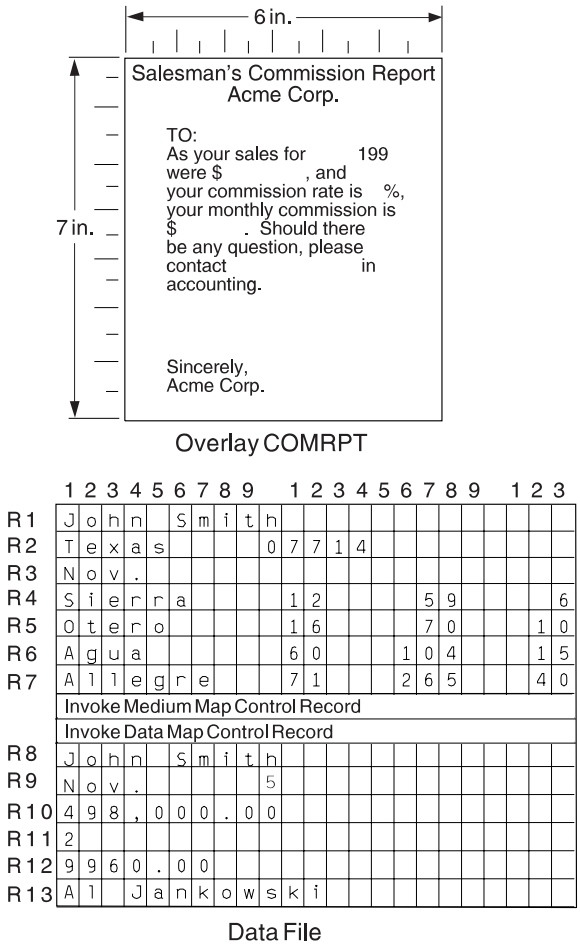


Figure 67. Input for a New Report Produced from the Combined Data Files

Here is the command stream needed to generate both pages of the preceding report:

```
FORMDEF SLSCOM ;
  COPYGROUP SLSRPT ;
    OVERLAY SLSRPT ;
    SUBGROUP OVERLAY SLSRPT ;
  COPYGROUP COMRPT ;
    OVERLAY COMRPT ;
    SUBGROUP OVERLAY COMRPT ;
PAGEDEF SLSCOM ;
  FONT M104 ;
  FONT M105 ;
  PAGEFORMAT SLSRPT ;                    /*SALES REPORT*/
    PRINTLINE  FONT M104
               POSITION 2 IN .5 IN ;
      FIELD START 1 LENGTH 23 ;
    PRINTLINE  POSITION 2 IN .75 IN ;
      FIELD START 1 LENGTH 9 ;           /*DEFAULT FIELD POSITION*/
      FIELD START 10 LENGTH 5
            POSITION 2.3 IN * ;
    PRINTLINE POSITION 1.5 IN 1 IN ;
      FIELD START 1 LENGTH 4 ;
    PRINTLINE  REPEAT 4
               POSITION .3 IN  1.8 IN ;
      FIELD START 1 LENGTH 7 ;           /*DEFAULT FIELD POSITION */
      FIELD START 11 LENGTH 3
            POSITION 1.5 IN * ;
```

```
    FIELD START 16 LENGTH 3
          POSITION 3 IN * ;
    FIELD START 21 LENGTH 3
          POSITION 4.3 IN * ;
 PAGEFORMAT  COMRPT ;                  /*COMMISSION REPORT      */
   PRINTLINE  FONT M105                /*RECORD 8               */
              POSITION 1.3 IN  1.7 IN ;
    FIELD START 1 LENGTH 18 ;
   PRINTLINE POSITION 3.3 IN 2.2 IN ;  /*RECORD 9               */
    FIELD START 1 LENGTH 4 ;           /*DEFAULT FIELD POSITION */
FIELD START 10 LENGTH 1
          POSITION 1.7 IN * ;
 PRINTLINE POSITION 1.9 IN 2.6 IN ;    /*RECORD 10              */
   FIELD START 1 LENGTH 10 ;
 PRINTLINE POSITION 4.2 IN 2.9 IN ;    /*RECORD 11              */
   FIELD START 1 LENGTH 1 ;
 PRINTLINE POSITION 1 IN 3.7 IN ;      /*RECORD 12              */
   FIELD  START 1 LENGTH 7 ;
 PRINTLINE POSITION 1.7 IN 4.2 IN ;    /*RECORD 13              */
   FIELD START 1 LENGTH  15 ;
```

Although requiring a complex series of commands, the following commission report is handled much like any other field processing problem: the data must be carefully mapped into the overlay exactly where it is wanted. If, as in this example, you change copy groups and page formats, both the Invoke Medium Map structured field and the Invoke Data Map structured field must be inserted into the data file where the changes are desired. Here they occur together.

Figure 68 shows both the commission report and the sales report. With page printers and with careful data positioning, such reports look like they were individually prepared with no differences in the presentation of the fixed data.

SALESMAN John Smith

TERRITORY Texas **NO.** 07714

MONTH Nov.

MODEL	UNITS	$ (000)	% of TOTAL
Sierra	12	59	6
Otero	16	70	10
Agua	60	104	15
Allegre	71	265	40

Salesman's Commission Report

Acme Corp.

To: John Smith

As your sales for Nov. 1995 were $498,000.00, and your commission rate is 2 %, your monthly commission is $9960.00. Should there be any question, please contact Al Jankowski in accounting.

Sincerely,
Acme Corp.

Figure 68. The Sales and the Commission Reports

Chapter 6. Conditional Processing

Conditional processing allows you to test fields within an input line data record (for example, a customer number). Based on the results of the test, you can specify the action to be taken such as to change copy groups or page formats. This section provides:

- An explanation of how conditional processing works
- A detailed list of rules, restrictions, and considerations
- Examples showing how conditional processing can be used to perform some commonly-requested functions

General Description

Conditional processing allows you to:

- Test the input data using the CONDITION command.
- Choose the copy group and page format to be used when printing the data.
- Change to a different copy group or page format after the data has been read. You can specify that the new copy group or page format is to be used:
  - Before printing the current subpage
  - Before printing the current line
  - After printing the current line
  - After printing the current subpage

Table 6 shows the tasks you may perform with conditional processing.

Table 6. Conditional Processing Tasks

Tasks	**Location of the Example**
Stack offset from previous jobs	"Jog Output Example" on page 116
Use different print directions for front and back sides of a sheet	"Duplex Output with Different Front and Back Print Directions" on page 116
Record reprocessing example	"Record Reprocessing Example" on page 117
Select different paper sources	"Selecting Paper from an Alternate Bin Example" on page 118
Multiple CONDITION commands	"Multiple CONDITION Commands" on page 119
Repeat PRINTLINE commands	"Field Processing When PRINTLINEs Are Repeated" on page 122

Using Conditional Processing versus Normal Line Data Processing

Normal line-data processing consists of:

- Setting up the physical page environment by defining a copy group
- Setting up the logical page environment by defining a page format

Input records correspond to PRINTLINE commands that determine such things as where the input records are to be printed, which font to use and what print direction to use. Only one copy group and page format can be used for processing each input record.

Conditional processing acts as a preprocessor by allowing you to test the input data before deciding which copy group and page format to use. Furthermore, you can change these specifications based on changes in the input data. Except for record reprocessing (explained on page 107), once the copy group and page-format specifications have been made, conditional processing operates the same as normal line-data processing.

Note: The copy group and page format can also be changed by placing Advanced Function Presentation data stream (AFP data stream) Invoke Medium Map (IMM) and Invoke Data Map (IDM) structured fields in the input data. Use of these structured fields within the input print file causes results that differ from what is described in this section. Refer to *Mixed Object Document Content Architecture Reference* for information about these structured fields.

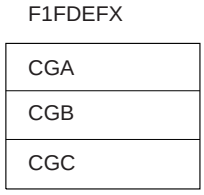
Using Conditional Processing to Set Up the Environment

Setting up the environment consists of selecting a copy group and a page format.

Selecting a Copy Group

Conditional processing can be used to *select* a copy group; it does not *process* the copy group.

As described in "Chapter 2. Using Form Definition Commands" on page 17, a form definition contains the controls that govern the physical page on which the print file is to be printed. A form definition can contain one or more copy groups as shown in the following diagram.

PPFA Commands	Resulting Form Definition
FORMDEF FDEFX . . COPYGROUP CGA . . OVERLAY ... SUBGROUP ... . COPYGROUP CGB . . OVERLAY ... SUBGROUP ... COPYGROUP CGC . . OVERLAY ... SUBGROUP	

The first copy group within a form definition is always active when processing of a print file begins. To select a different copy group, use the CONDITION command.

Note: By using the BEFORE SUBPAGE and BEFORE LINE parameters with conditional processing, you can change to a different active copy group before any lines have actually been formatted.

Using the previous diagram as a reference, assume copy group CGB is active. The copy-group selections that can be made from a CONDITION command are:

condname
which starts the named copy group

CURRENT
which *re*starts copy group CGB

= which *re*starts copy group CGB (alternate for CURRENT)

NEXT which starts copy group CGC

FIRST which starts copy group CGA

NULL which does *not* make any change to the current copy group processing

/ which does *not* make any change to the current copy group processing (alternate for NULL)

See "Using the CONDITION Command to Select a Copy Group and a Page Format" on page 114 for more information on each of these options.

Selecting a Page Format

Conditional processing can be used to *select* an active page format. Selecting the page format does not change the basic rules for processing a page format:

- PRINTLINE commands are processed sequentially unless skip-to-channel or spacing commands are used.
- When the end of the page format is reached, processing returns to the first PRINTLINE command in the same page format. Processing does *not* continue with the next page format (if any) in the page definition.

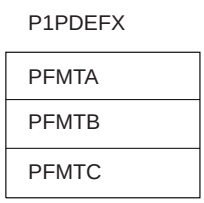
However, conditional processing does involve some additional considerations:

- Subpages

  A page format consists of one or more subpages. A subpage is defined by a group of PRINTLINE commands followed by an ENDSUBPAGE command. If an ENDSUBPAGE command is not defined, then the entire page format is one subpage. See "Subpage Description and Processing" on page 107 for more information.
- Record reprocessing

  Record reprocessing is used when input records are processed according to one set of copy-group and page-format specifications, and then new specifications are invoked for the same input records. See "Record Reprocessing Description and Processing" on page 107 for more information.

As described in "Chapter 3. Using Page Definition Commands for Traditional Line Data" on page 33, a page definition is a set of controls for formatting line-data and unformatted ASCII files (typically AIX) for printing on a logical page. A page definition can contain one or more page formats as shown in the following diagram.

PPFA Commands	Resulting Page Definition
PAGEDEF PDEFX . . PAGEFORMAT PFMTA . . PRINTLINE ... PRINTLINE ... . PAGEFORMAT PFMTB . . PRINTLINE ... PRINTLINE ... . PAGEFORMAT PFMTC . . PRINTLINE ... PRINTLINE	

The first page format in the page definition is always active when processing of the print file begins. To invoke a new page format, use the CONDITION command.

Note: By using the BEFORE SUBPAGE and BEFORE LINE parameters, it is possible to change to a different active page format before any lines have actually been formatted.

Using the previous diagram as a reference, assume page format PFMTB is active. The page-format selections that can be made from a CONDITION command are:

condname
which starts the named page format

CURRENT
which *re*starts page format PFMTB

= which *re*starts page format PFMTB (alternate for CURRENT)

NEXT which starts page format PFMTC

FIRST which starts page format PFMTA

NULL which does *not* make any change to the current page format processing

/ which does *not* make any change to the current page format processing (alternate for NULL)

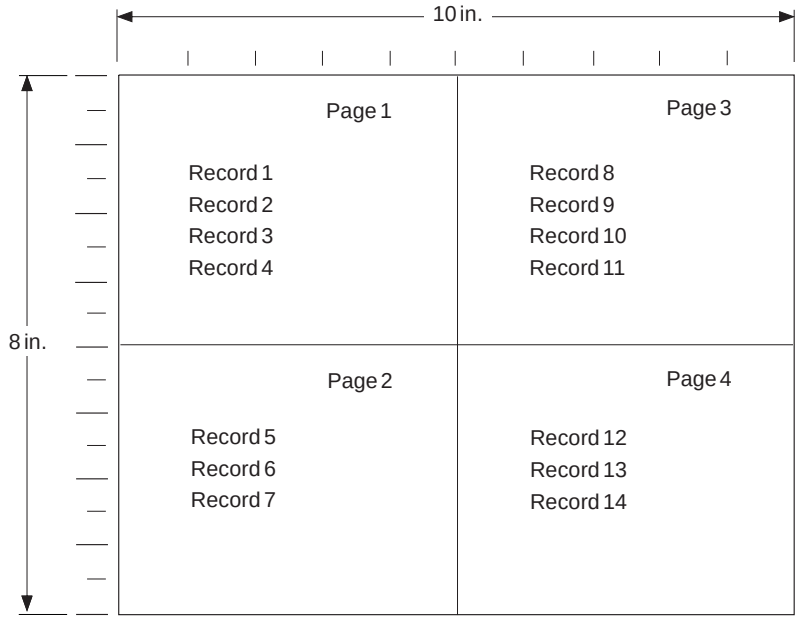
See “Using the CONDITION Command to Select a Copy Group and a Page Format” on page 114 for more information on each of these options.

Subpage Description and Processing

A page format consists of one or more subpages. A subpage is defined by a group of PRINTLINE commands followed by an ENDSUBPAGE command. If an ENDSUBPAGE command is not defined, then the entire page format is one subpage. The following considerations apply to subpages:

- Subpages are necessary only with conditional processing.

  Multiple-up printing can be done with or without subpages being defined, but to change the page format or copy group at the level of one of the multiple-up pages, the multiple-up pages must be defined as subpages. In the following diagram, pages 1 through 4 can be defined as four separate subpages within one page format, or all defined within one subpage. However, in order to present the data on page 3 (for example) in a format different from that used for pages 1 and 2, the four pages must be defined as subpages.



- A subpage is processed sequentially starting from the beginning of the page format. Moving from one subpage to the next subpage is done by processing all the PRINTLINE commands for a given subpage, or by skipping (by means of the CHANNEL subcommand) or spacing to a PRINTLINE command in a different subpage.

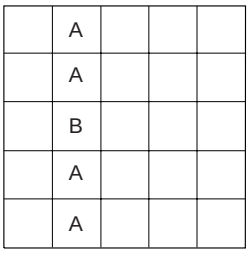
  Note: Conditional processing cannot be used to select a subpage except by default. When a page format is started (or the *current* one is restarted), processing begins with the first PRINTLINE command of the page format. The effect is to select the first subpage in the page format.

Record Reprocessing Description and Processing

Record reprocessing is used when input records are processed according to one set of copy group and page format specifications, and then new specifications are invoked for the same input records. If the new specifications are to be applied using either the BEFORE SUBPAGE or the BEFORE LINE parameter, then the input records must be processed again using the new specifications instead of the original ones.

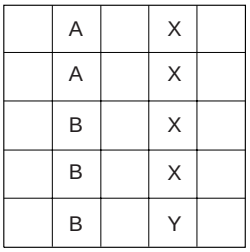
Note: Input records are not printed twice; record reprocessing just changes the specifications used when formatting the records.

The process is shown in the following diagram.

PPFA Commands	Input Records
PAGEFORMAT PFMTA ; PRINTLINE POSITION 1 IN 1 IN DIRECTION ACROSS REPEAT 5 ; CONDITION cond1 START 2 LENGTH 1 WHEN EQ 'B' BEFORE SUBPAGE NULL PAGEFORMAT PFMTB ; PAGEFORMAT PFMTB ; PRINTLINE POSITION 7 IN 1 IN DIRECTION DOWN REPEAT 5 ; CONDITION cond2 START 4 LENGTH 1 WHEN EQ 'Y' BEFORE SUBPAGE NULL PAGEFORMAT PFMTA ;	

Assume page format PFMTA is active. Under normal processing the first input record would print in the ACROSS direction, starting at a horizontal offset of 1 inch and a vertical offset of 1 inch. However, the third record satisfies the CONDITION statement and causes a new page format (PFMTB) to be started. Since CONDITION cond1 specifies BEFORE SUBPAGE, the first two records must be *reprocessed* using page format PFMTB. As a result, all of the records will be printed in a DOWN direction, starting at a horizontal offset of 7 inches and a vertical offset of 1 inch.

If allowed to operate without restrictions, record reprocessing could force PSF into an infinite loop. For example:

PPFA Commands	Input Records
PAGEFORMAT PFMTA ; PRINTLINE POSITION 1 IN 1 IN DIRECTION ACROSS REPEAT 5 ; CONDITION cond1 START 2 LENGTH 1 WHEN EQ 'B' BEFORE SUBPAGE NULL PAGEFORMAT PFMTB ; PAGEFORMAT PFMTB ; PRINTLINE POSITION 7 IN 1 IN DIRECTION DOWN REPEAT 5 ; CONDITION cond2 START 4 LENGTH 1 WHEN EQ 'Y' BEFORE SUBPAGE NULL PAGEFORMAT PFMTA ;	

As in the previous example, page format PFMTA is initially active, and input record 3 results in the selection of page format PFMTB. However, page format PFMTB has a condition that checks position four for the character 'Y', which is satisfied by input record 5. Therefore, if there were no restrictions, page format PFMTA would again be selected, the input data would be reprocessed (starting with input record 1), leading to an infinite loop.

To prevent this situation, after a BEFORE condition has been satisfied, all other BEFORE conditions are ignored until data has actually been formatted. See "Record Reprocessing" on page 110 for detailed information on this restriction.

Conditional Processing Rules, Restrictions, and Considerations

Multiple Conditions

Conditional processing supports:

- Multiple PRINTLINE commands in each subpage
- Multiple CONDITION commands on one PRINTLINE command
- Multiple WHEN statements on one CONDITION command

Rule

For *all these situations*, the rule is the same; the *first* true condition is the one processed, and any following true conditions are ignored.

Considerations

Conditions are evaluated when they are encountered. For example, if a true condition has not been detected when an OTHERWISE statement is encountered, the OTHERWISE statement always results in a true condition. (An exception to this is explained in "Interaction Between the CONDITION Command and the CHANNEL Subcommand" on page 111.)

See “Multiple CONDITION Commands” on page 119 for an example of multiple CONDITION commands.

Record Reprocessing

Restrictions

To prevent an infinite program loop, be aware that the following restrictions apply:

1. When the conditional action is to take place *before* the current subpage:
   a. Actions specified as taking place before the current subpage are shut off until the current subpage end.
   b. Actions specified as taking place before the current line are shut off for one line (the first line processed in the subpage).
2. When the conditional action is to take place before the current line, actions specified as taking place before the current subpage or before the current line are shut off for one line.

Considerations

- If a *before subpage* condition is true and causes a switch to a new page format, all *before subpage* conditions in the new page format are *ignored*.
- If a *before line* condition is true and causes a switch to a new page format, all *before subpage* and *before line* conditions in the new page format are ignored until one line has been processed.

The consequence of this is that, after a true condition, at least one line must be processed before the next *before* condition will be considered. This can be confusing because a condition that would otherwise yield a true result can be ignored.

See “Record Reprocessing Example” on page 117 for an example of record reprocessing.

Interaction Between a CONDITION Command and a REPEAT Subcommand

See “Interaction Between the CONDITION Command and the CHANNEL Subcommand” on page 111 for what can appear to be an exception to the following rules.

Rule for a CONDITION Command and a REPEAT Subcommand

The REPEAT subcommand is used with the PRINTLINE command to specify the number of printlines (usually greater than one) that are to be constructed with the same specifications (font, direction, and so on). The CONDITION command is used to invoke conditional processing based on the data in a particular line. When the REPEAT and CONDITION commands are both specified for the same PRINTLINE command, *every line* described by the PRINTLINE command is checked for the given condition until either the condition is satisfied or there are no more lines described by the PRINTLINE command.

Note: This is different from the way in which the CHANNEL and POSITION subcommands interact with the PRINTLINE command. These two subcommands apply only to the *first line* described by the PRINTLINE command.

Rule for a CONDITION Command With an OTHERWISE Subcommand

The REPEAT subcommand is used with the PRINTLINE command to specify the number of printlines (usually greater than one) that are to be constructed with the same specifications (font, direction, and so on). The CONDITION command is used to invoke conditional processing based on the data in a particular line. The CONDITION command includes one or more WHEN subcommands and may include an OTHERWISE subcommand. If an OTHERWISE is coded, and none of the preceding WHEN conditions are true, the OTHERWISE condition *is always true*. If an OTHERWISE command is not coded, it is treated as a null.

Considerations

For the situation where REPEAT and CONDITION with OTHERWISE are coded for the same PRINTLINE command, the *first* input line determines the processing to be performed. This happens because either one of the WHEN conditions or the OTHERWISE condition is always true for the very first line.

Interaction Between the CONDITION Command and the CHANNEL Subcommand

Rule

A condition is checked if its associated PRINTLINE command is *actually processed*.

Note: ANSI carriage controls and machine (EBCDIC) carriage controls are processed differently. See the **SPACE_THEN_PRINT** subcommand on page "Subcommands" on page 197 for more information.
ANSI A skip or space occurs before printing the line.
Machine
The line is printed and then skipping or spacing is done.

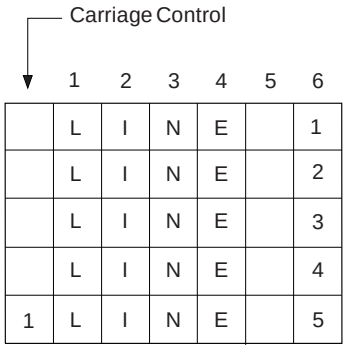
For a CONDITION to be checked, it must be associated with the PRINTLINE command that is actually used for printing.

ANSI Skipping Consideration

The PRINTLINE command is not processed if a skip-to-channel-n character in the carriage control field causes the given PRINTLINE command not to be processed.

If a data record contains a character '1' (for example) in the carriage control field, and a PRINTLINE command has been specified with CHANNEL 1 subcommand, the data record is processed under the "new" PRINTLINE command (the one that specified CHANNEL 1). Any CONDITION associated with the "old" PRINTLINE command is ignored (never even checked). See the following diagram for an example of this.

The character '1' in the carriage-control field of the fifth input record causes a page end before condition *cond1* is ever checked. Thus, the fifth input record is processed using the first PRINTLINE command of the current page format.

PPFA Commands	Input Records
PAGEFORMAT PFMTA ; PRINTLINE CHANNEL 1 ; PRINTLINE ; PRINTLINE ; PRINTLINE ; PRINTLINE ; CONDITION cond1 START 6 LENGTH 1 WHEN EQ '5' AFTER SUBPAGE CURRENT NULL;	

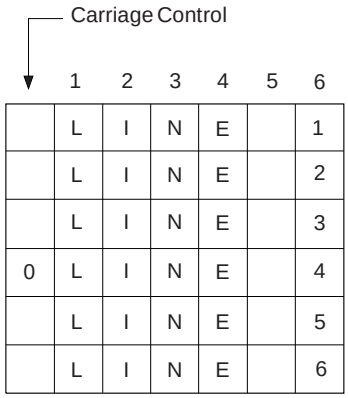
Considerations

The PRINTLINE command is not processed if the PRINTLINE command is spaced over, for example, when multiple line spacing causes certain PRINTLINE commands to be bypassed.

If the input-record carriage-control field specifies a double space before print (for example), and a CONDITION command is specified for the spaced line, the CONDITION is ignored (never checked). Because the OTHERWISE subcommand is part of a CONDITION command, the OTHERWISE subcommand is also ignored.

This can be confusing. You might expect an OTHERWISE condition to be true if all other conditions have failed. In fact, the OTHERWISE condition can be true if it is associated with a PRINTLINE command that is actually processed. See the following diagram for an example of this. This assumes ANSI carriage controls have been specified for this print file. ANSI carriage control ‘0’ means space two lines before printing.

The fifth input record contains data (character ‘5’ in the sixth position) that would normally satisfy the condition specified on the fifth PRINTLINE command. However, the character ‘0’ in the carriage control field of input record 4 causes the fifth PRINTLINE command to be ignored. The fifth input record is processed by the sixth PRINTLINE command; therefore, the condition is not satisfied.

PPFA Commands	Input Records
PAGEFORMAT PFMTA ; PRINTLINE CHANNEL 1 ; PRINTLINE ; PRINTLINE ; PRINTLINE ; PRINTLINE ; CONDITION cond1 START 6 LENGTH 1 WHEN EQ '5' AFTER SUBPAGE CURRENT NULL; PRINTLINE ;	

WHEN CHANGE is Always False at Start of a Page Format

Rule

The WHEN CHANGE process compares the contents of a given field with the contents of the same field in the last record that was processed *with the current page format* and *current condition*. Whenever a page format is started (either by a condition that *changes* page formats or when processing of the *data file begins*), a WHEN CHANGE condition is always false because the previous record was not processed with the *current* page format.

Note: The following meanings apply to the previous statement.

changes
switching to a page format that has a different name

data file begins
if conditional processing invokes the CURRENT data map, CHANGE information is retained

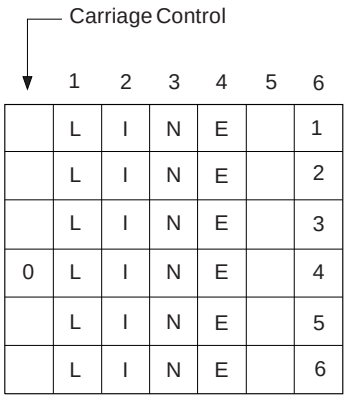
Considerations

Ensure that the WHEN CHANGE statement is processed before the switch to a new page format has been performed. See "Multiple CONDITION Commands" on page 119 for an example of how a combination of WHEN CHANGE BEFORE SUBPAGE and WHEN CHANGE AFTER SUBPAGE can lead to unexpected results.

Relationship of CC and TRC fields to the START Subcommand

Rule

The position specified by the START subcommand of the CONDITION command is in reference to the start of the *data record*. The first one or two bytes of an input record may contain either both a carriage-control character (CC) or a table-reference character (TRC). However, these characters are not considered part of the data record and are not to be counted when determining the START subcommand value. In the following example, the field being checked is actually the seventh character of the input record, but is the sixth character of the data record.

PPFA Commands	Input Records
PAGEFORMAT PFMTA ; PRINTLINE CHANNEL 1 ; PRINTLINE ; PRINTLINE ; PRINTLINE ; PRINTLINE ; CONDITION cond1 START 6 LENGTH 1 WHEN EQ '5' AFTER SUBPAGE CURRENT NULL; PRINTLINE ;	

Using the CONDITION Command to Select a Copy Group and a Page Format

Rules

1. Within the CONDITION command, a copy group and a page format can be specified by using either a specific name or a parameter (CURRENT or =, FIRST, NEXT) or NULL or / can be specified. The use of the NULL or / parameters differs from the use of the others:

   Others
   When any parameter other than NULL or / is specified, the specifications for the copy group or page format selected replace the current specifications. When the current specifications are replaced, the action is referred to as starting or restarting the copy group or page format. In AFP terminology, an Invoke Medium Map (IMM) command is generated for a copy group and an Invoke Data Map (IDM) command is generated for a page format.

   NULL or /
   When NULL or / is specified, no IMM or IDM is generated and processing continues as if no condition check was present.

2. The COPYGROUP and the PAGEFORMAT parameters are positional. If both parameters are specified, the COPYGROUP parameter must be first. If you want only to specify the copy group, the PAGEFORMAT parameter can be omitted, or specified as NULL or /. However, if you want only to specify the page format, the COPYGROUP parameter must be specified as NULL or /.

Considerations

Starting or Restarting a Copy Group: When a copy group is started (or restarted), the remaining input data is forced to the start on the next *sheet*. Therefore, if duplex output was expected, but the copy group is restarted while processing the front side of a sheet, the remaining data will start on the front side of the *next* sheet rather than on the back side of the current sheet.

See “Duplex Output with Different Front and Back Print Directions” on page 116 for an example.

Furthermore, observe that any copy group action except NULL restarts the page format (see the following item).

Starting or Restarting a Page Format: When a page format is started (or restarted), the remaining input data is forced to the start on the next side. Furthermore, that data is processed starting with the first PRINTLINE command in the specified page format. This is true even if CURRENT is specified as the page format parameter.

Not Restarting a Copy Group: If the copy group is not to be restarted, specify NULL or /. Do *not* specify COPYGROUP NULL or COPYGROUP /.

The following example illustrates this point. The command sequence on the left invokes a copy group named NULL. The command sequence on the right leaves the current copy group active.

Incorrect Format	Correct Format
CONDITION condname START ... . . . WHEN ... COPYGROUP NULL . .	CONDITION condname START ... . . . WHEN ... NULL . .

Not Restarting a Page Format: If the page format is not to be restarted, specify NULL or / (or simply omit the specification). Do *not* specify PAGEFORMAT NULL or PAGEFORMAT /.

The following example illustrates this point. The command sequence on the left invokes a page format named NULL. The command sequence on the right will leave the current page format active.

Incorrect Format	Correct Format
CONDITION condname START ... . . . WHEN ... COPYGROUP CGA PAGEFORMAT NULL . .	CONDITION condname START ... . . . WHEN ... COPYGROUP CGA NULL . .

Variable Length Records and the CONDITION Command

Considerations

The CONDITION command inspects a field that starts at a particular position and extends for a certain length. If the *entire* field is not available within the input record, the condition is always false. If the input file contains variable-length records, the record may not extend the full length specified by the START and LENGTH subcommands. In this way, a condition which seems as if it should be satisfied can actually fail.

Truncation of Blanks and the CONDITION Command

Considerations

Truncation occurs when blank characters are removed from the end of records on the spool. If blank truncation is in effect, the result can be the same as if the input file contained variable-length records.

Blank truncation is a consideration at the time the input records are passed to print server. In the JES2 environment, blank truncation occurs unless the BLNKTRNC=NO parameter is specified. In the JES3 environment, blank truncation

occurs unless the TRUNC=NO parameter is specified as part of either the BUFFER or SYSOUT initialization statements. Blank truncation can affect conditional processing since a field could “disappear” by being truncated causing no WHEN/OTHERWISE clause to be executed.

Conditional Processing Examples

This section provides conditional processing examples. The examples are grouped into functionally similar applications and are increasingly complex. The examples provided are:

- Jog output based on a change in the input data
- Duplex output with different front and back print directions
- Record reprocessing
- Select paper from an alternate bin
- Multiple CONDITION commands

Jog Output Example

This example shows how to jog the printed output, based on a change in the input data.

Copy group CGJOG specifies *JOG YES*. Page format PFJOG contains a CONDITION command that checks for any change in positions 8 through 10. If a change is detected, copy group CGJOG is restarted. Observe that the only result is to start printing on a new sheet and to jog that sheet.

Jog Output Example

```
FORMDEF TJOG;
  COPYGROUP CGJOG JOG YES;

PAGEDEF TJOG;
  PAGEFORMAT PFJOG WIDTH 11 IN HEIGHT 8.5;
    PRINTLINE REPEAT 50
              CHANNEL 1;
    CONDITION NUPAGE START 8 LENGTH 3
      WHEN CHANGE BEFORE SUBPAGE
      COPYGROUP CGJOG;
```

Duplex Output with Different Front and Back Print Directions

This example shows how to establish one print direction on the front side and a different print direction on the back side of a duplex sheet.

The page definition in this example contains two page formats, each of which has a CONDITION statement that always returns a true value. The value is true because the character in position 1 will always have a value greater than or equal to hexadecimal zero. Therefore, every time a page change occurs (front to back, or back to next front) a different page format will be started. The different DIRECTION statements in the two page formats change the layout of the text on the page.

Observe that the COPYGROUP parameter is specified as NULL. If a parameter other than NULL or / is specified for COPYGROUP, the copy group restarts every time a page change occurs. Because restarting a copy group forces data to a new sheet, duplex printing *does not occur*.

Duplex Output

```
FORMDEF XMPDUP
        DUPLEX NORMAL;

PAGEDEF XMPDUP WIDTH 8.5  HEIGHT 11.0;
  PAGEFORMAT  P2FRONT DIRECTION ACROSS;
    PRINTLINE CHANNEL 1 POSITION   0.75   TOP;
      CONDITION GOTOBACK START 1 LENGTH 1
        WHEN GE X'00'  AFTER SUBPAGE NULL  PAGEFORMAT P2BACK;
    PRINTLINE REPEAT 59;

  PAGEFORMAT  P2BACK DIRECTION UP;
    PRINTLINE CHANNEL 1 POSITION   0.25   TOP;
      CONDITION GOTOFRNT START 1 LENGTH 1
        WHEN GE X'00'  AFTER SUBPAGE NULL PAGEFORMAT P2FRONT;
    PRINTLINE REPEAT 59;
```

Record Reprocessing Example

This example uses the BEFORE SUBPAGE function with record reprocessing because the copy group and page format cannot be determined until input record 3 for each subpage has been read.

Notes:

1. This example includes two subpages.
2. The CONDITION command specifies that the action to be performed is NEWFORM. Therefore, if the condition is satisfied, the data in the current subpage is forced to start on the next form. If the data is already at the start of a new form, no action is performed. In other words, a blank page is not generated.

Record Reprocessing Example

```
/* Page definition for 2-up printing              */
/* Test field in line 3 of each subpage           */
/* Eject to new sheet if the field changes.       */

PAGEDEF REPROC
        WIDTH  10.6 HEIGHT 8.3 DIRECTION DOWN;

  PAGEFORMAT  PFREPROC;

    /* Definition of first subpage                */
    PRINTLINE CHANNEL 1
              REPEAT 2
              POSITION MARGIN TOP;

    PRINTLINE REPEAT 1
              POSITION MARGIN NEXT;
              CONDITION EJECT
                START 5 LENGTH 5
                WHEN CHANGE BEFORE SUBPAGE
                NEWFORM;

    PRINTLINE REPEAT 40
              POSITION MARGIN NEXT;
    ENDSUBPAGE;

    /* Definition of second subpage              */
    PRINTLINE CHANNEL 1
              REPEAT 2
              POSITION 5.3 TOP;

    PRINTLINE REPEAT 1
              POSITION 5.3 NEXT;
              CONDITION EJECT;

    PRINTLINE REPEAT 40
              POSITION 5.3 NEXT;
    ENDSUBPAGE;
```

Selecting Paper from an Alternate Bin Example

This example selects the first sheet from the alternate bin and all other pages from the primary bin. This function is useful when special paper (such as one having the company logo) is to be used for the first page of a document.

Note: Bin selection will be overridden by the printer should the form defined to each bin be the same form number. Only the primary bin is selected.

Alternate Bin Example

```
/* The form definition contains two copy groups --   */
/*   ALTBIN - for the first page                     */
/*   PRIBIN - for all other pages                    */

FORMDEF BINEX
        DUPLEX NO;
  COPYGROUP ALTBIN BIN 2;
  COPYGROUP PRIBIN BIN 1;

PAGEDEF BINEX
        WIDTH 8.3  HEIGHT 10.6;

  /* Pageformat for first page - bin 2               */

  PAGEFORMAT FIRST;
    PRINTLINE CHANNEL 1
              POSITION MARGIN TOP;
    CONDITION GOTOPRIM  START 1 LENGTH 1
      WHEN GE X'00'  AFTER SUBPAGE
      COPYGROUP PRIBIN PAGEFORMAT REST;
    PRINTLINE REPEAT 59;

  /* Pageformat for all other pages - bin 1          */

   PAGEFORMAT REST;
     PRINTLINE CHANNEL 1
               POSITION MARGIN TOP
               REPEAT 60;
```

Multiple CONDITION Commands

Two examples are shown here. The first example shows how two CONDITION commands can interact to give unintended results. The second example shows how to use the two CONDITION commands to achieve the correct results.

Example 1 Multiple CONDITION Command—Incorrect Solution

The example on page 120 demonstrates how two CONDITION commands can interact to give unintended results. Specifically, one CONDITION command causes a change of page format and then a second CONDITION command inspects a field with a WHEN CHANGE subcommand.

The purpose of condition:

NEWREP
: Starts a new report on a new sheet of paper whenever the specified field changes and jogs the output so the report can be easily located.

SHIFTB and SHIFTF
: Handles the situation where all four subpages of the front (or back) contain data.

  In this situation, the objective is to change the print direction of the text on the page.

In the situation where both conditions *seem* to be true at the same time, the results may be unexpected.

Note: Condition SHIFTB (or SHIFTF) takes effect *after* the current subpage and therefore precedes the *before subpage* processing defined by condition

NEWREP. Because condition SHIFTB results in starting a new page format, condition NEWREP returns a false value, and the expected new report processing is not performed.

Example 2 Multiple CONDITION Command—Correct Solution

The example on page 121 differs from the one on page 120 in two significant ways:

- Because the page format for the back side is the first one defined in the page definition, it is the one that is initially active
- Both CONDITION commands (NEWREP and SHIFTIT) specify that the action should happen before the current subpage has been processed

When processing begins, condition NEWREP fails because this is a WHEN CHANGE condition and the page format has just been started. However, condition SHIFTIT will return a true result, and the NEXT page format (PFFRONT) will be started. No lines have been formatted, so condition SHIFTIT has the effect of moving to the page format for the front side.

INCORRECT Solution Example

```
FORMDEF XMPICO OFFSET 0 0 DUPLEX RTUMBLE JOG YES REPLACE YES;
  COPYGROUP CG1;
        OVERLAY OVLY1;
        OVERLAY OVLY2;
        SUBGROUP       OVERLAY OVLY1 FRONT;
        SUBGROUP       OVERLAY OVLY2 BACK ;

PAGEDEF XMPICO REPLACE YES;
        FONT GT24;
        FONT GT12;

        /* Definition of pageformat for front side */
        PAGEFORMAT PFFRONT WIDTH 11 IN HEIGHT 8.5 IN DIRECTION UP;
SETUNITS 1 PELS 1 PELS LINESP 16 LPI;
          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 75 188;
          CONDITION NEWREP START 8 LENGTH 3
            WHEN CHANGE BEFORE SUBPAGE COPYGROUP CG1 PAGEFORMAT PFFRONT;
          PRINTLINE REPEAT 40 FONT GT24 POSITION 75 NEXT;
          ENDSUBPAGE;

          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 1377 188;
          CONDITION NEWREP START 8;
          PRINTLINE REPEAT 40 FONT GT24 POSITION 1377 NEXT;
          ENDSUBPAGE;

          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 75 1102;
          CONDITION NEWREP START 8;
          PRINTLINE REPEAT 40 FONT GT24 POSITION 75 NEXT;
          ENDSUBPAGE;

          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 1377 1102;
          CONDITION NEWREP START 8;
          CONDITION SHIFTB START 1 LENGTH 1
            WHEN GE X'00' AFTER SUBPAGE NULL PAGEFORMAT PFBACK;
          PRINTLINE REPEAT 40 FONT GT24 POSITION 1377 NEXT;
          ENDSUBPAGE;

        /* Definition of pageformat for back side    */
        PAGEFORMAT PFBACK WIDTH 8.5 IN HEIGHT 11 IN DIRECTION ACROSS;
        SETUNITS 1 PELS 1 PELS LINESP 8 LPI;
          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT12 POSITION 75 61;
          CONDITION NEWREP START 8;
          PRINTLINE REPEAT 40 FONT GT12 POSITION 75 NEXT;
          ENDSUBPAGE;

          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT12 POSITION 75 1335;
          CONDITION NEWREP START 8;
          CONDITION SHIFTF START 1 LENGTH 1
            WHEN GE X'00' AFTER SUBPAGE NULL PAGEFORMAT PFFRONT;
          PRINTLINE REPEAT 40 FONT GT12 POSITION 75 NEXT;
          ENDSUBPAGE;
```

CORRECT Solution Example

```
FORMDEF XMPCOR OFFSET 0 0 DUPLEX RTUMBLE JOG YES REPLACE YES;
  COPYGROUP CG1;
        OVERLAY OVLY1;
        OVERLAY OVLY2;
        SUBGROUP       OVERLAY OVLY1 FRONT;
        SUBGROUP       OVERLAY OVLY2 BACK ;

PAGEDEF XMPCOR REPLACE YES;
        FONT GT24;
        FONT GT12;
        /* The pageformat for the back side of the form is  */
        /* the first pageformat in the PAGEDEF.  Therefore, */
        /* it will initially be the active pageformat       */
        PAGEFORMAT PFBACK WIDTH 8.5 IN HEIGHT 11 IN DIRECTION ACROSS;
SETUNITS 1 PELS 1 PELS LINESP 8 LPI;
          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT12 POSITION 75 61;
          CONDITION NEWREP START 8 LENGTH 3
            WHEN CHANGE BEFORE SUBPAGE COPYGROUP CG1 PAGEFORMAT
            PFFRONT;
          CONDITION SHIFTIT START 1 LENGTH 1
            WHEN GE X'00' BEFORE SUBPAGE NULL NEXT;
          PRINTLINE REPEAT 40 FONT GT12 POSITION 75 NEXT;
          ENDSUBPAGE;

          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT12 POSITION 75 1335;
          CONDITION NEWREP START 8;
          PRINTLINE REPEAT 40 FONT GT12 POSITION 75 NEXT;
          ENDSUBPAGE;

        /* This is the pageformat for the front side of the form. */
        PAGEFORMAT PFFRONT WIDTH 11 IN HEIGHT 8.5 IN DIRECTION UP;
SETUNITS 1 PELS 1 PELS LINESP 16 LPI;
          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT23 POSITION 75 188;
          CONDITION NEWREP START 8;
          CONDITION SHIFTIT START 1;
          PRINTLINE REPEAT 40 FONT GT24 POSITION 75 NEXT;
          ENDSUBPAGE;

          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 1377 188;
          CONDITION NEWREP START 8;
          PRINTLINE REPEAT 40 FONT GT24 POSITION 1377 NEXT;
          ENDSUBPAGE;

          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 75 1102;
          CONDITION NEWREP START 8;
          PRINTLINE REPEAT 40 FONT GT24 POSITION 75 NEXT;
          ENDSUBPAGE;

          PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 1377 1102;
          CONDITION NEWREP START 8;
          PRINTLINE REPEAT 40 FONT GT24 POSITION 1377 NEXT;
          ENDSUBPAGE;
```

Field Processing When PRINTLINEs Are Repeated

The following examples show the effect of the [LINE | FIELD] parameter on **REPEAT** *n*.

The first PRINTLINE example uses FIELD type repetition. The second PRINTLINE example shows LINE type repetition.

Note: When LINE type repetition is used, SETUNITS LINESP may need to be set to a higher value to avoid over printing.

REPEAT n type FIELD Example

```
PAGEDEF  rept01  WIDTH      8.0 IN
                 HEIGHT    10.5 IN
                 LINEONE    0.2 IN 0.2 IN
                 DIRECTION ACROSS
                 REPLACE   YES;

   FONT  normal  CR10  SBCS  ROTATION 0;
   FONT  italic  CI10  SBCS  ROTATION 0;
   FONT  bold    CB10  SBCS  ROTATION 0;
   .
   .
   .
   SETUNITS LINESP 6 LPI;

   PRINTLINE  POSITION 1.0 IN 1.0 IN
              DIRECTION ACROSS
              FONT bold
              REPEAT 3 FIELD;
      FIELD   POSITION 0.0 IN 0.0 IN
              DIRECTION ACROSS
              FONT normal
              START * LENGTH 20;
      FIELD   POSITION 2.5 IN 0.0 IN
              DIRECTION DOWN
              FONT normal
              START * LENGTH 20;
      FIELD   POSITION 2.5 IN 2.5 IN
              DIRECTION BACK
              FONT normal
              START * LENGTH 20;
      FIELD   POSITION 0.0 IN 2.5 IN
              DIRECTION UP
              FONT normal
              START * LENGTH 20;
```

REPEAT n type LINE Example

```
:
SETUNITS LINESP 3.0 IN;

PRINTLINE  POSITION 5.0 IN 1.0 IN
           DIRECTION ACROSS
           FONT bold
           REPEAT 3 LINE;
   FIELD   POSITION 0.0 IN 0.0 IN
           DIRECTION ACROSS
           FONT normal
           START * LENGTH 20;
   FIELD   POSITION 2.0 IN 0.0 IN
           DIRECTION DOWN
           FONT normal
           START * LENGTH 20;
   FIELD   POSITION 2.0 IN 2.0 IN
           DIRECTION BACK
           FONT normal
           START * LENGTH 20;
   FIELD   POSITION 0.0 IN 2.0 IN
           DIRECTION UP
           FONT normal
           START * LENGTH 20;
```

The next example shows Input Line Data.

(Input) Line Data:

```
Field Type Repeat   Field Type Repeat  Field Type Repeat  Field Type Repeat
Field Type Repeat   Field Type Repeat  Field Type Repeat  Field Type Repeat
Field Type Repeat   Field Type Repeat  Field Type Repeat  Field Type Repeat
Line Type Repeat    Line Type Repeat   Line Type Repeat   Line Type Repeat
Line Type Repeat    Line Type Repeat   Line Type Repeat   Line Type Repeat
Line Type Repeat    Line Type Repeat   Line Type Repeat   Line Type Repeat
Field Type Repeat   Notice that the  fields are repeated based on the prior
instance of the same field, and not the  printline. This  has advantages if
special effects are desired.
Line Type Repeat    is based on the     printline.  Good  for sales tickets.
Generally, this type of repeat needs a   SETUNITS LINESP     command...
 ...so that lines    won't overlap!      This is             SETUNITS    LINESP 3 IN
```

Sample Output

When the previous example is processed by print server, the following output will be printed.

```
Field Type Repeat
Field Type Repeat
Field Type Repeat
```

```
Field Type Repeat
Field Type Repeat
Field Type Repeat
```

```
Field Type Repeat
Field Type Repeat
Field Type Repeat
```

```
Field Type Repeat
Field Type Repeat
Field Type Repeat
```

```
Line Type Repeat
Line Type Repeat
Line Type Repeat
```

```
Line Type Repeat
Line Type Repeat
Line Type Repeat
```

```
Line Type Repeat
Line Type Repeat
Line Type Repeat
```

```
Line Type Repeat
Line Type Repeat
Line Type Repeat
```

Line Type Repeat

is based on the

printline. Good

for sales tickets.

Generally, this type of repeat needs a

SETUNITS LINESP

command...

This is

won't overlap!

SETUNITS LINESP 3 IN ...so that lines

Field Type Repeat

Notice that the
field, and not the
desired.

fields are repeated
printline. This

instance of the same
special effects are

based on the prior
has advantages if

Chapter 7. N_UP Printing

With N_UP printing, which is defined in the form definition, you can print up to four pages on a sheet of paper in simplex mode and up to eight pages in duplex mode. Each of these pages are independent, allowing use of different page formats and copy groups for each page. This provides significantly more flexibility and function than the traditional multiple-up capability which is defined in the page definition. Refer to "N_UP Compared to Multiple-up" on page 145 for more differences between N_UP printing and multiple-up printing.

There are two levels of N_UP: [2]

- basic N_UP supported by older AFP printers: 3825, 3827, 3828, 3829, 3835, and 3900-001.
- enhanced N_UP supported by printers with the new Advanced Function Common Control Unit (AFCCU): IBM 3900 Wide and Duplex printers, 3935, 3130, 3160, and follow-on AFCCU printers.

N_UP Partitions and Partition Arrangement

A key concept in N_UP printing is the *partition*. In both basic and enhanced N_UP, each sheet of paper is divided into equal size areas called partitions. Pages are placed in these partitions in sequential order in basic N_UP. Pages are placed in relation to one or more of these partitions in enhanced N_UP. Knowing the partition arrangement is critical to designing applications using N_UP.

Note: If you are using basic N_UP printing with print server set to **DATACK=BLOCK**, data must fall within the boundary of the partition. Any data placed outside the edge of the partition boundary will not be printed, and no error message will be generated. However, enhanced N_UP printing allows pages to overlap partitions. The only limits are that the pages must not extend beyond the boundaries of the physical sheet, and the pages must not exceed the total number of N_UP partitions specified for the sheet.

The number, size, and position of partitions are determined by three things:

- the N_UP value (1, 2, 3, or 4)
- the size and shape of the sheet of paper
- the form definition presentation options, PRESENT and DIRECTION

When printing in duplex mode, the same number of partitions is also defined for the back of the sheet. For normal duplex, back partitions are placed as if the sheet were flipped around its right side or y-axis. For tumble duplex, they are placed as if the sheet were flipped around its top edge, or x-axis. See Figure 70 on page 129 and Figure 71 on page 130 for illustrations of duplex partitions.

Figure 69 on page 128 through Figure 72 on page 131 show the partition arrangement that results from every combination of N_UP value, paper size, and presentation option.

2. You must have the correct level of PPFA to generate basic or enhanced N_UP commands and the correct level of Print Servers Facility for your operating system to drive the printer in the basic or enhanced N_UP mode.

Use these figures to determine how your N_UP application will be formatted by the printer. In the figures, each equal-sized partition has a number indicating its default presentation sequence. The origin for each partition is in the same relative position as the origin point shown for the medium. This point serves as the top left corner for a page printed in the ACROSS (or 0°) printing direction. Figure 69 through Figure 72 on page 131 also shows the way data formatted in the DOWN printing direction is printed when you use a DOWN copy group.

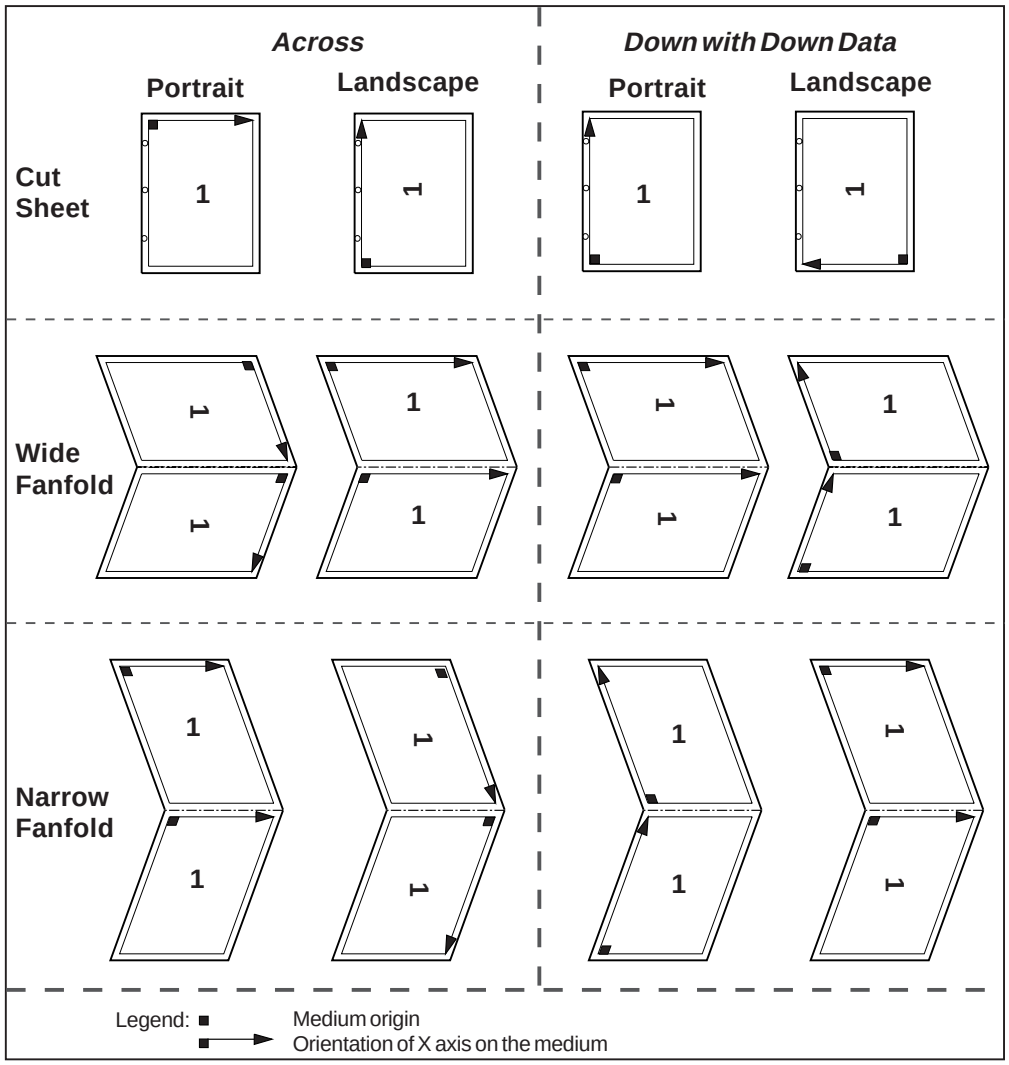


Figure 69. N_UP 1 Partition Arrangement

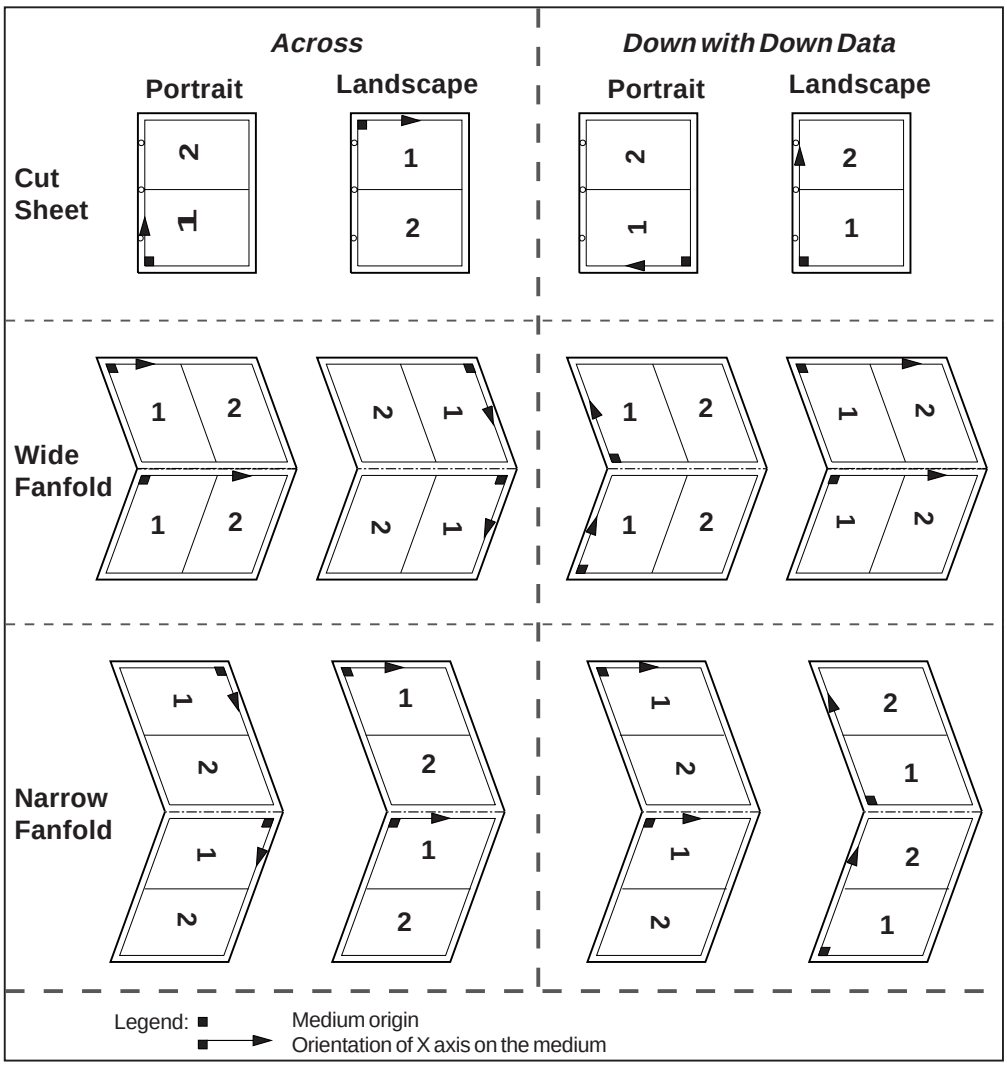


Figure 70. N_UP 2 Partition Arrangement

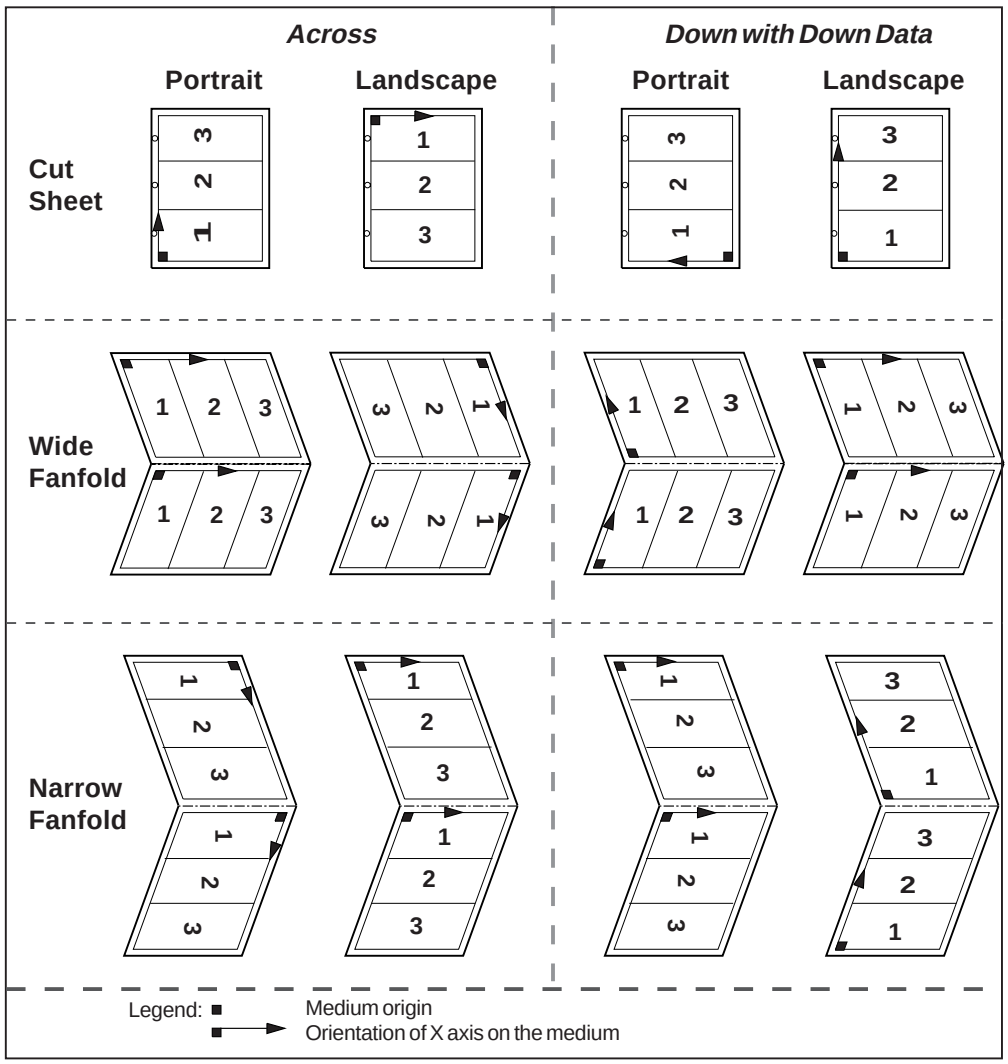


Figure 71. N_UP 3 Partition Arrangement

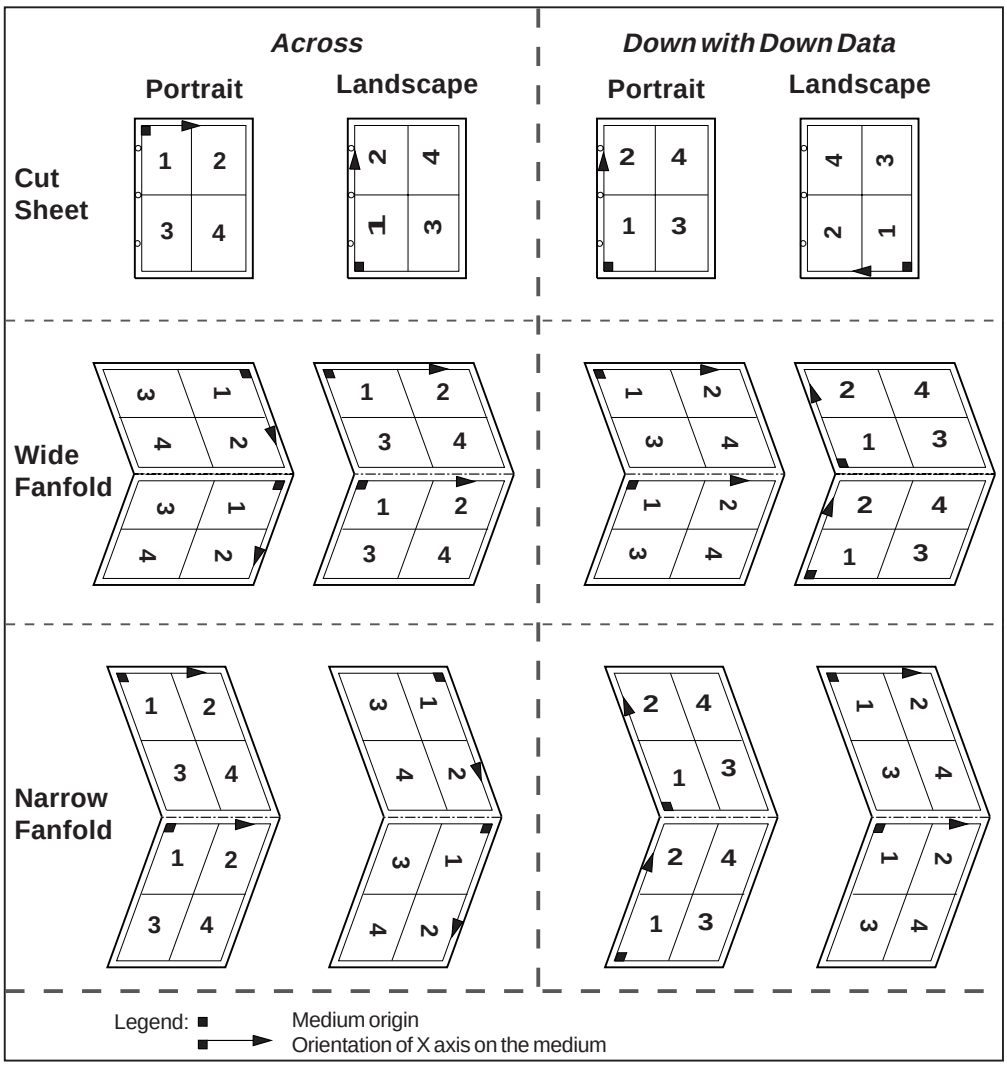
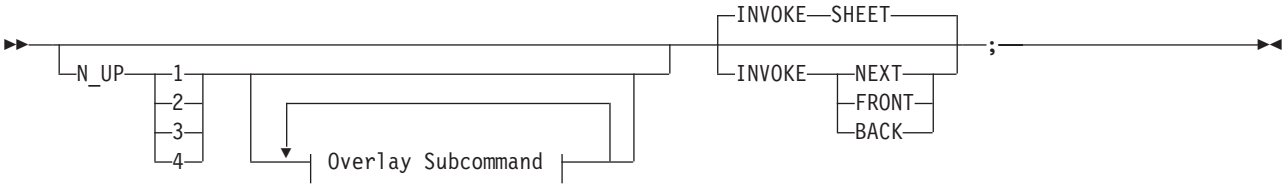


Figure 72. N_UP 4 Partition Arrangement

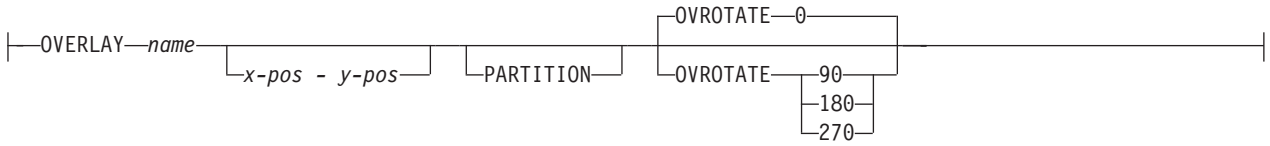
Basic N_UP Printing

You can specify the N_UP subcommand on either the FORMDEF or COPYGROUP commands in the form definition. Figure 73 shows the subcommands and parameters enabled with basic N_UP printing.

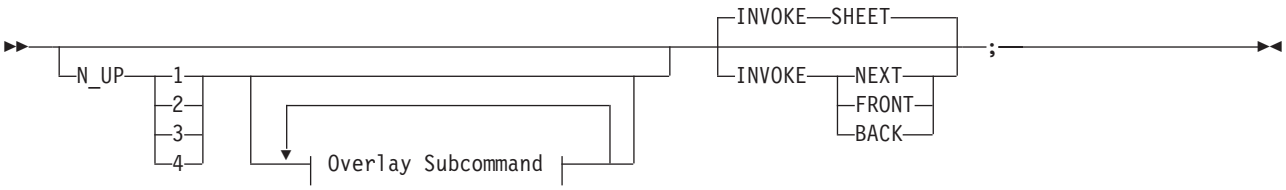
FORMDEF Subcommand



Overlay Subcommand:



COPYGROUP Subcommand



Overlay Subcommand:

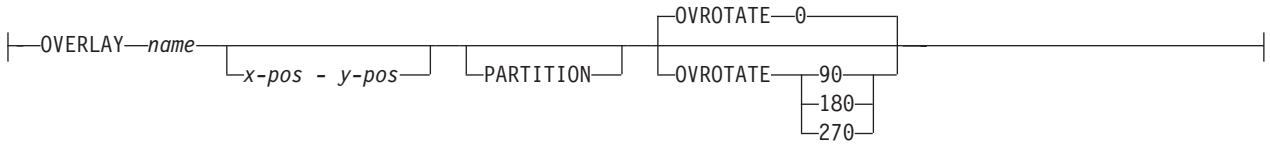


Figure 73. Subcommands for Basic N_UP Printing

The N_UP subcommand divides the medium into one, two, three, or four partitions, as described in "N_UP Partitions and Partition Arrangement" on page 127. The OVERLAY subcommand prints a page overlay in each partition at a specified offset from the page origin or the partition origin. For more information about page overlays, see "Medium Overlays and Page Overlays" on page 144.

The INVOKE subcommand controls the action that occurs if you invoke a new copy group. You can invoke copy groups using conditional processing in the page definition or by including an Invoke Medium Map (IMM) structured field in the print data. The default action is to eject to a new sheet. By specifying an INVOKE subcommand on a COPYGROUP command, you can instead eject to a new N_UP

partition, which may be on the same sheet. If printing in duplex mode, you can specify whether to eject to a partition on the front or back side of the sheet.

You must use page overlays instead of medium overlays if you want to change overlays while ejecting to a new partition. print server honors the NEXT, FRONT, and BACK values of the INVOKE subcommand only if the new copy group has the same medium modifications as the previous copy group. Medium modifications include duplexing, bin, page offset, N_UP values, presentation, direction, and medium overlays. If any of these modifications differ, print server ejects to a new sheet when the copy group is invoked.

By combining INVOKE with the N_UP OVERLAY subcommand, you can place different page overlays in different partitions when you change copy groups. This is illustrated in “Basic N_UP Example 1: Using INVOKE and OVERLAY”.

The following examples show the use of basic N_UP. Because each example builds on the previous one, read them in sequential order to better understand basic N_UP. All the pages used in the examples are formatted in the ACROSS printing direction. Their orientation on the media is the result of using the available PRESENT and DIRECTION combinations in the FORMDEF command.

Basic N_UP Example 1: Using INVOKE and OVERLAY

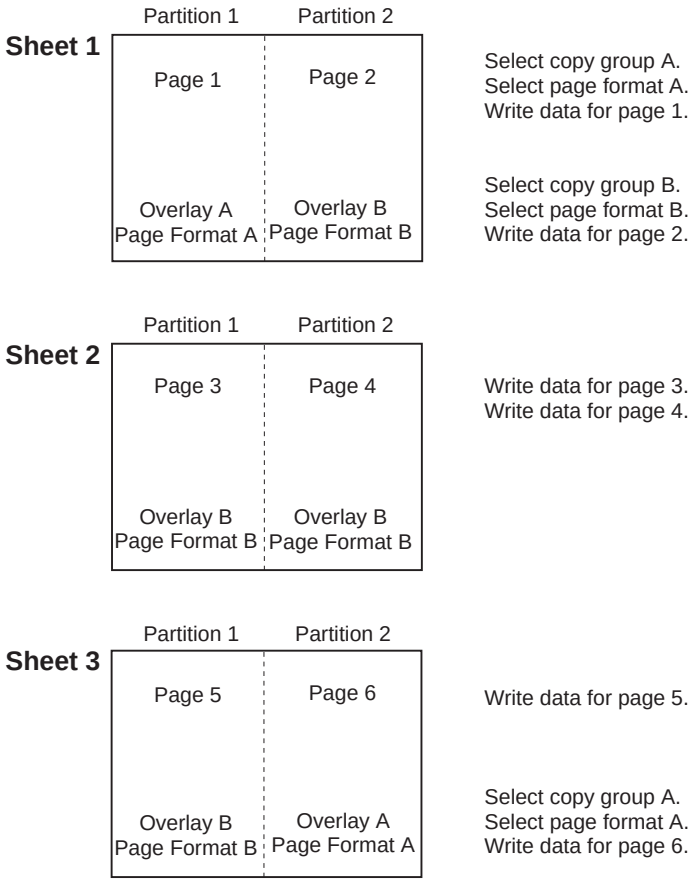


Figure 74. Basic N_UP Example 1: Using INVOKE and OVERLAY

```
FORMDEF TWOUPS ;

COPYGROUP A
   N_UP 2
     OVERLAY A
   INVOKE NEXT ;

COPYGROUP B
   N_UP 2
     OVERLAY B
   INVOKE NEXT ;
```

Figure 75. Form Definition for Basic N_UP Example 1

Figure 74 on page 133 shows the INVOKE and OVERLAY functions of basic N_UP printing. Specifying INVOKE NEXT on the COPYGROUP command ensures that when the copy group is invoked by an Invoke Medium Map (IMM) structured field with conditional processing, the next page will be placed in the next partition of the N_UP form.

The OVERLAY subcommand specifies a *page overlay*, which can be positioned relative to the page origin or relative to the partition origin. In basic N_UP, the OVERLAY subcommand prints the overlay with the page data in every partition on the sheet. However, as you will see in this example, using INVOKE NEXT allows the application to use different overlays in different partitions.

Example 1 has been defined as N_UP 2 simplex with the default PORTRAIT ACROSS presentation, which results in the partitions illustrated in Figure 74 on page 133. The application uses different page formats on different application pages. With N_UP, changing page formats ejects to the next partition, just as it ejects to a new page in applications without N_UP.

The application also needs different overlays on different pages. Because the overlays are specified on N_UP in the COPYGROUP subcommand, the application accomplishes this by changing copy groups. Without the INVOKE subcommand, changing the copy group forces an eject to a new physical sheet. However, because INVOKE NEXT is specified, the eject will be to the next partition. Changing to copy group B after page 1 is written places page 2 in partition 2 of the same physical sheet. If the change is made after a page is placed in partition 2, the eject to the next partition will be to partition 1 of the next sheet. The page is printed with the overlay specified in the new copy group.

Notes:

1. The pages in this example are line-format print data, formatted using a page definition. The example would be the same for MO:DCA data, except that page formats would not be used.
2. You can select the copy groups and page formats by including IMM and IDM structured fields in the print data or by using conditional processing in the page formats.
3. Overlays can be defined as *page overlays* in the page definition or in the form definition N_UP or PLACE subcommands. Overlays can also be defined as *medium overlays* in the form definition SUBGROUP command. If you want to change overlays when ejecting to a new partition, use *page overlays* instead of medium overlays. See “Medium Overlays and Page Overlays” on page 144 for information about page and medium overlays.

Basic N_UP Example 2: Normal Duplex

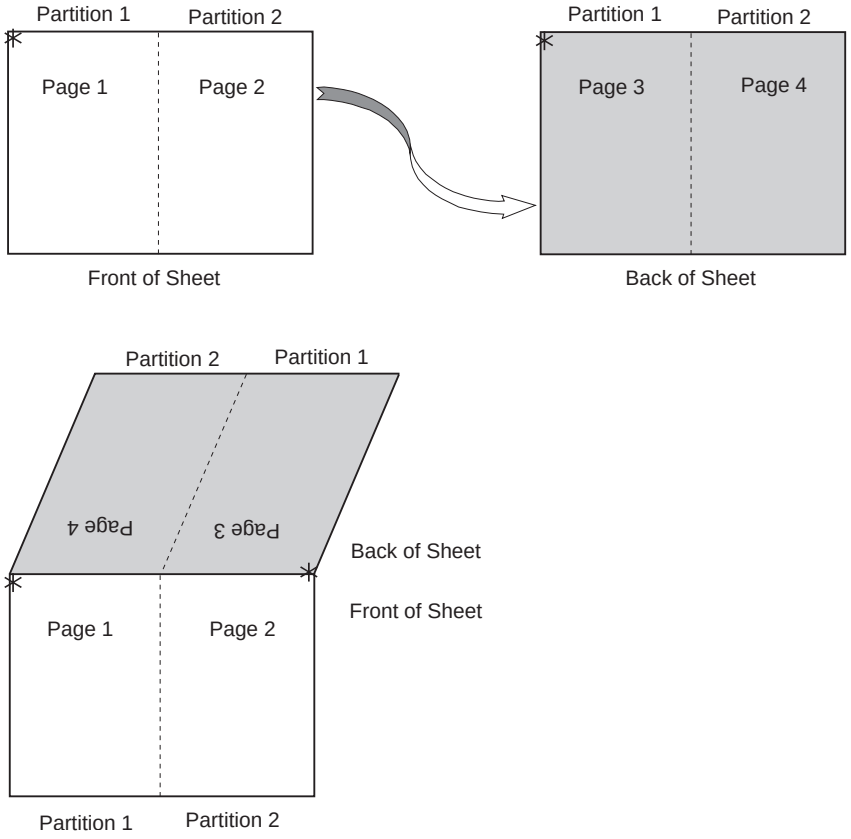


Figure 76. Basic N_UP Example 2: Normal Duplex

```
FORMDEF NUPDUP
N_UP 2
PRESENT PORTRAIT
DIRECTION ACROSS
DUPLEX NORMAL ;
```

Figure 77. Form Definition for Basic N_UP Example 2: Normal Duplex

Figure 76 shows the partition order for duplexed pages. This figure also shows the partitions into which the sheet is divided for N_UP 2 with PORTRAIT presentation and ACROSS direction. With normal duplex, the sheet is rotated around its "y-axis", which is the right edge of the sheet. The result is that partition 2 for the back side of the sheet is on the back of partition 1 for the front side, and page 4 is on the back of page 1. The tops of pages 3 and 4 are aligned with the tops of pages 1 and 2.

Basic N_UP Example 3: Tumble Duplex

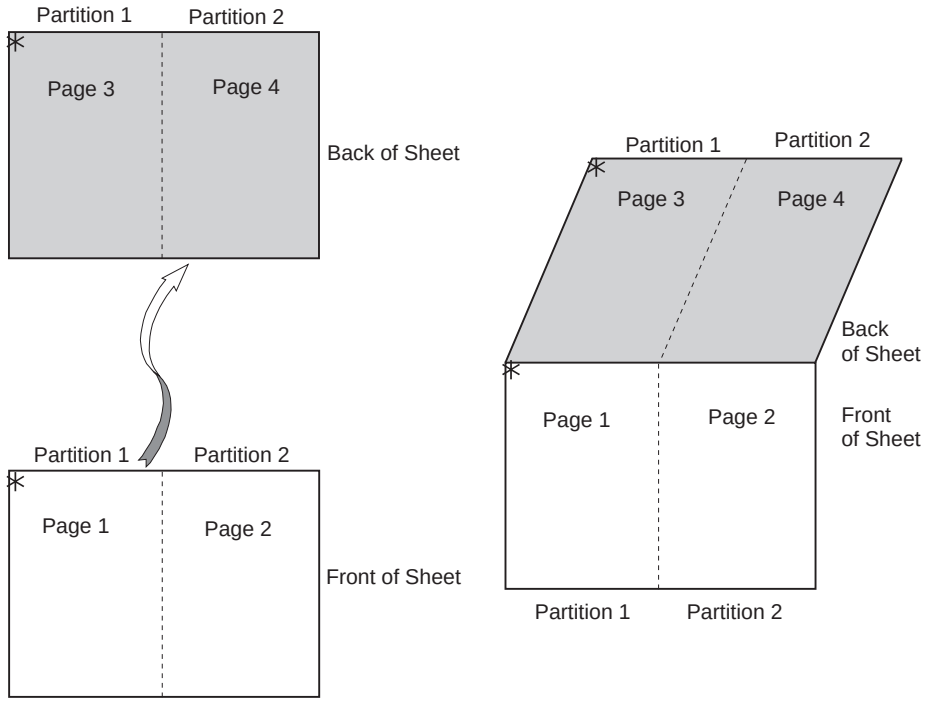


Figure 78. Basic N_UP Example 3: Tumble Duplex

```
FORMDEF NUPTUM
N_UP 2
PRESENT PORTRAIT
DIRECTION ACROSS
DUPLEX TUMBLE ;
```

Figure 79. Form Definition for Basic N_UP Example 3: Tumble Duplex

Figure 78 shows the partition order for tumble duplex pages. This figure also shows the partitions into which the sheet is divided for N_UP 2 with PORTRAIT presentation and ACROSS direction. With tumble duplex, the sheet is rotated around its *x-axis*, which is the top of the sheet. The result is that partition 1 of the back of the sheet falls on the back of partition 1 for the front, and page 3 falls on the back of page 1. The tops of pages 3 and 4 are aligned with the bottoms of pages 1 and 2. For more information about normal and tumble duplex printing, refer to “Normal Duplex and Tumble Duplex” on page 14.

Enhanced N_UP Printing

Enhanced N_UP is supported on the newest family of AFP printers, which includes the 3900-0W1 Wide, the 3900 Duplex, the 3935, 3130, and 3160. In addition to all the function of basic N_UP, enhanced N_UP includes the powerful PLACE subcommand.

Using the PLACE subcommand, you can place pages in the partitions in any sequence, specify unique overlays for each page, and rotate both the page and the overlays in the partitions. You can place multiple pages in the same partition and no pages in other partitions, and you can extend pages across partition boundaries. In short, you can place pages of any size at any location on the front or back of the sheet, in any orientation. The only limits are that the pages must not extend outside the boundaries of the physical sheet, and the pages must not exceed the total number of N_UP partitions specified for the sheet.

You use a single PLACE command to place each page of data on the sheet. You must specify the same number of PLACE commands as the number of N_UP partitions for the sheet. This is required for error recovery and restart integrity. If you do not want to place as many pages as partitions, you can specify CONSTANT on a PLACE command to indicate that no data is to be placed in the partition. You can specify the OVERLAY subcommand with the CONSTANT subcommand to place overlays without user's data. The syntax diagrams in Figures 80 and 81 show the subcommands and parameters enabled with enhanced N_UP printing.

For most applications, place constant overlays **before** placing data on the sheet. This is because the overlay is not actually placed until the next page of data is placed. If your application changes copy groups or runs out of pages on the sheet before reaching the constant overlay PLACE subcommand, the constant overlay will not be printed. However, if you do not want the overlays to print in these cases, place the constant overlay after placing the page data.

FORMDEF

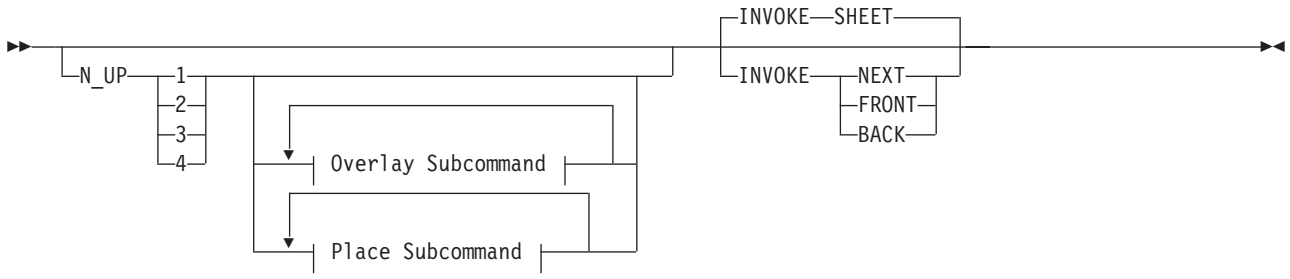
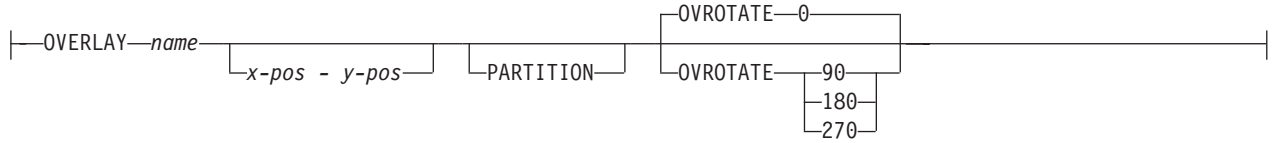
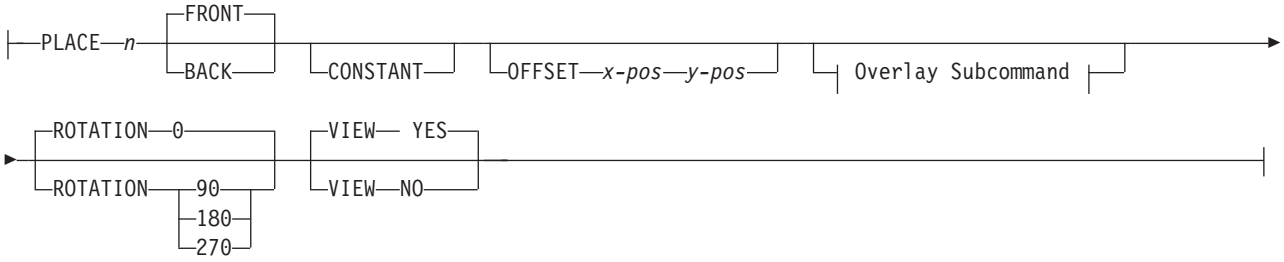


Figure 80. FORMDEF Subcommand for Enhanced N_UP Printing

Overlay Subcommand:



*** Place Subcommand:**



*** The use of the PLACE subcommand indicates enhanced N_UP printing.**

COPYGROUP

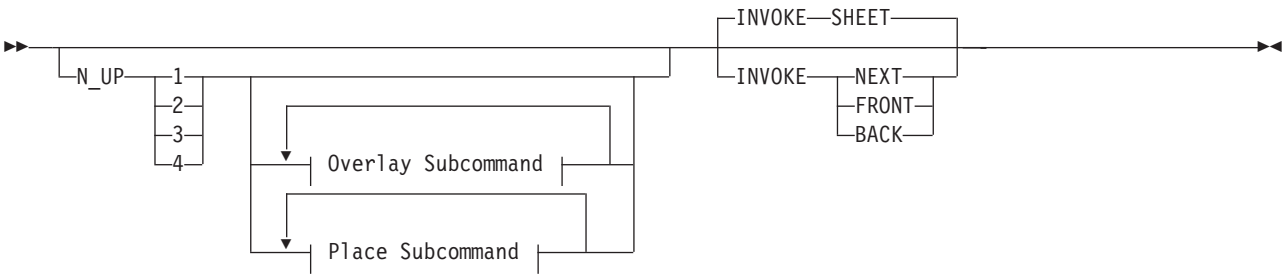
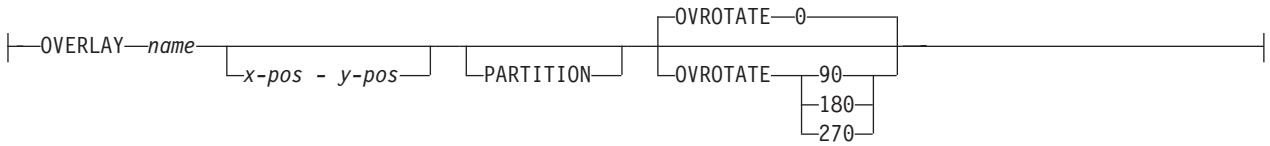
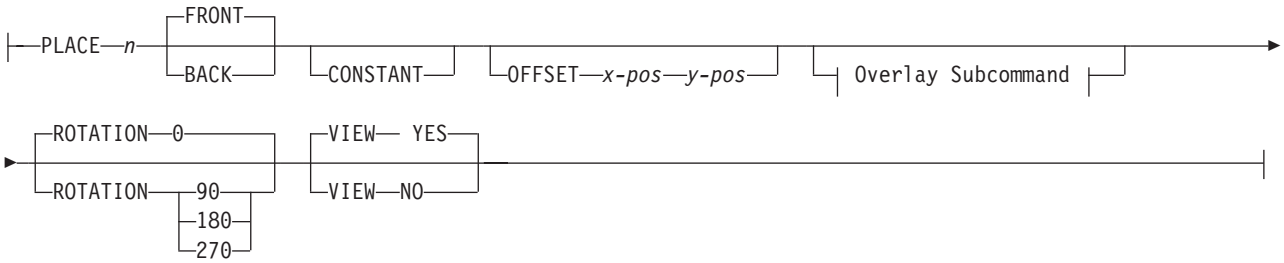


Figure 81. COPYGROUP Subcommand for Enhanced N_UP Printing

Overlay Subcommand:



*** Place Subcommand:**



*** The use of the PLACE subcommand indicates enhanced N_UP printing.**

The following examples show enhanced N_UP printing. Read these examples in sequence to better understand enhanced N_UP printing.

Enhanced N_UP Example 1: Using PLACE

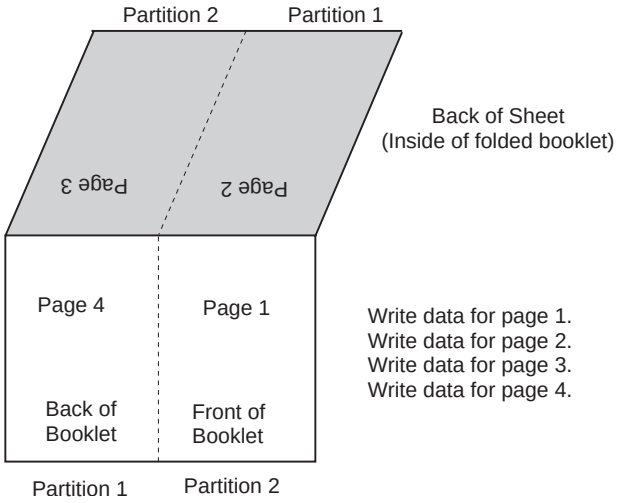


Figure 82. Enhanced N_UP Example 1: Using PLACE

```
   FORMDEF BOOKLT DUPLEX NORMAL
       N_UP 2
 /* Page 1 */   PLACE 2 FRONT
 /* page 2 */   PLACE 1 BACK
 /* Page 3 */   PLACE 2 BACK
 /* Page 4 */   PLACE 1 FRONT ;
```

Figure 83. Form Definition for Enhanced N_UP Example 1

Figure 82 shows the function of the PLACE subcommand in specifying the sequence of partitions into which pages will be placed. This example is N_UP 2 duplex. The default partition sequence is from left to right. Notice that when printing in normal duplex, partition 1 on the back of the sheet aligns with partition 2 on the front of the sheet. See “Basic N_UP Example 2: Normal Duplex” on page 135 and “Basic N_UP Example 3: Tumble Duplex” on page 136 for information on N_UP duplex partitions.

For this booklet, you do not want to print pages in the default order: partitions 1 and 2 on the front, followed by partitions 1 and 2 on the back. Instead, print the pages so that when the sheet is folded, you will have a booklet, with page 1 on the front outside of the booklet, pages 2 and 3 inside the folded booklet, and page 4 on the back outside of the booklet. The form definition shown in Figure 83 uses the PLACE subcommand of enhanced N_UP to place pages in the partitions in the order needed to accomplish this. The application writes the pages in order, page 1 through page 4, and the N_UP form definition provides the correct sequencing in the partitions.

Enhanced N_UP Example 2: Using CONSTANT and OVERLAY

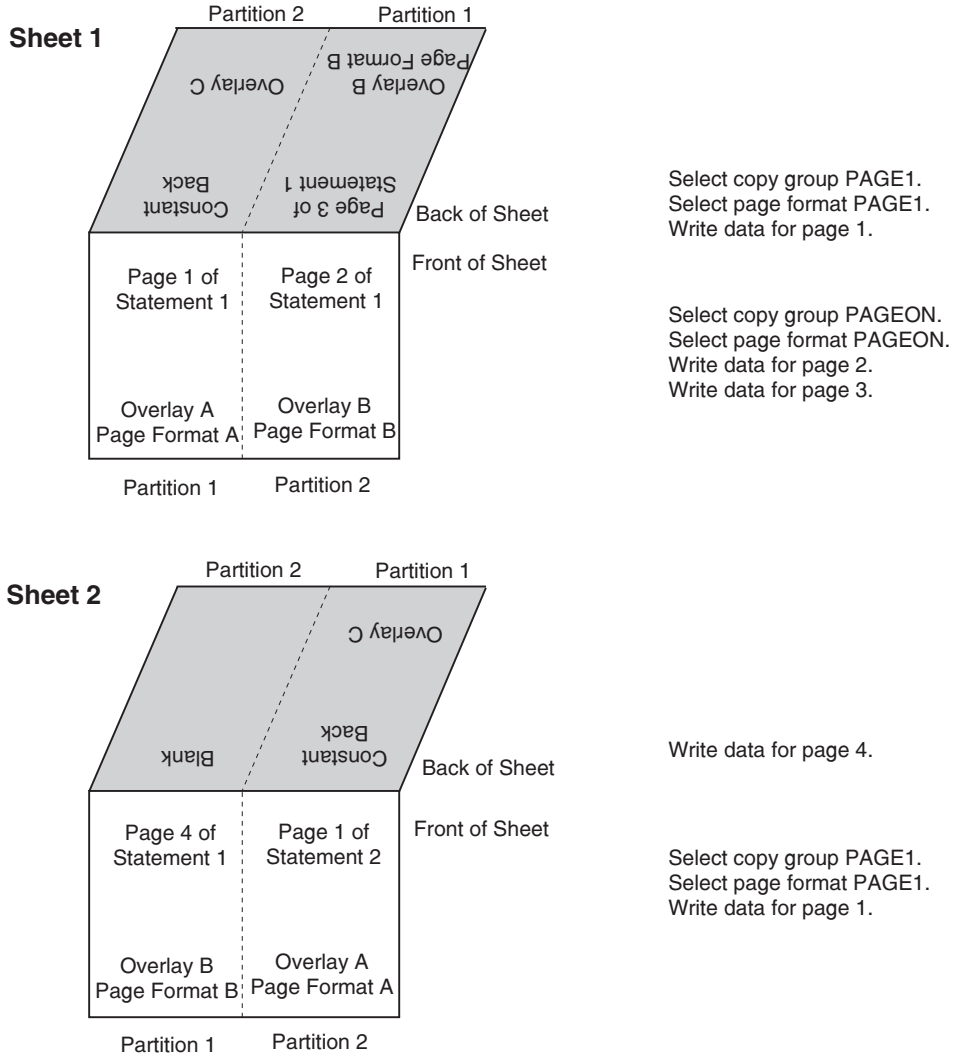


Figure 84. Enhanced N_UP Example 2: Using CONSTANT and OVERLAY

```
FORMDEF STATMT DUPLEX NORMAL ;

COPYGROUP PAGE1
  INVOKE BACK
  N_UP 2
  PLACE 2 BACK      CONSTANT OVERLAY C
  PLACE 1 FRONT     OVERLAY A
  PLACE 1 BACK      CONSTANT OVERLAY C
  PLACE 2 FRONT     OVERLAY A  ;

COPYGROUP PAGON
  INVOKE NEXT
  N_UP 2
  PLACE 1 FRONT     OVERLAY B
  PLACE 2 BACK      OVERLAY B
  PLACE 2 FRONT     OVERLAY B
  PLACE 1 BACK      OVERLAY B  ;
```

Figure 85. Form Definition for Enhanced N_UP Example 2

Figure 84 on page 140, introduces the CONSTANT subcommand of enhanced N_UP and shows the functions of the PLACE subcommand, which was described in "Enhanced N_UP Example 1: Using PLACE" on page 139 and the INVOKE subcommand, which was described in "Basic N_UP Example 1: Using INVOKE and OVERLAY" on page 133. This figure represents a user application that is printing customer statements using the values N_UP 2 duplex. The PLACE subcommand places the pages in the correct order for post-processing equipment to cut the sheets into individual pages and interleave them to produce sequential pages. The INVOKE subcommand guarantees that one customer's statement will never be printed on the back side of another customer's statement. The N_UP 2 subcommand, combined with the default PORTRAIT ACROSS presentation, divides the sheet into the two partitions illustrated in Figure 84 on page 140.

In Figure 84 on page 140, page 1 of each customer's statement is printed with overlay A. The back side of page 1 is a constant overlay, with no user's data. The remaining pages of each customer's statement are printed with overlay B.

The copy groups place the required overlays on both the right and left halves of the sheet, so that a new customer statement can begin on either half of the sheet. COPYGROUP PAGON assigns overlay B to all partitions on the sheet. COPYGROUP PAGE1 assigns overlay A to all front partitions and overlay C to all back partitions. The CONSTANT parameter used with OVERLAY C means that no user's data will be printed in the partition with the overlay. To guarantee that the constant overlay prints whenever page 1 is printed, the PLACE subcommand for the constant overlay is specified before the PLACE subcommand for page 1 print data. The INVOKE subcommand specifies BACK to ensure that the overlay is printed on the back of the partition.

In the application shown in Figure 84 on page 140, the copy group is changed to PAGON after page 1 is printed. Because the constant overlay and page 1 were printed with the first two PLACE commands of copy group PAGE1, the third PLACE command in new copy group will be used for the next page. Page 2 of statement 1 will be placed in partition 2 front, as specified in the third PLACE subcommand of copy group PAGON.

After the fourth and last page of statement 1, the copy group is changed back to PAGE1 to print page 1 of statement 2. Page 4 of statement 1 printed in front

partition 1 using the first PLACE subcommand of copy group PAGON. N_UP selects the second PLACE subcommand of copy group PAGE1: `PLACE 1 FRONT`. But the INVOKE subcommand for copy group PAGE1 specifies BACK. N_UP continues sequentially through the PLACE subcommands of copy group PAGE1 until it finds a BACK partition. This is the third PLACE subcommand: `PLACE 1 BACK CONSTANT OVERLAY C`. The constant overlay is placed in partition 1 on the back of the sheet, then page 1 of the new customer's statement is printed using the next PLACE subcommand: `PLACE 2 FRONT` on the front side of the constant overlay.

Note: You can use NEXT, FRONT, or BACK on the INVOKE subcommand only when switching between copy groups that have identical medium modifications. This includes identical N_UP values and an identical number of PLACE subcommands. If the copy groups have different values, the INVOKE command causes an eject to a new physical sheet.

Enhanced N_UP Example 3: Asymmetric Pages

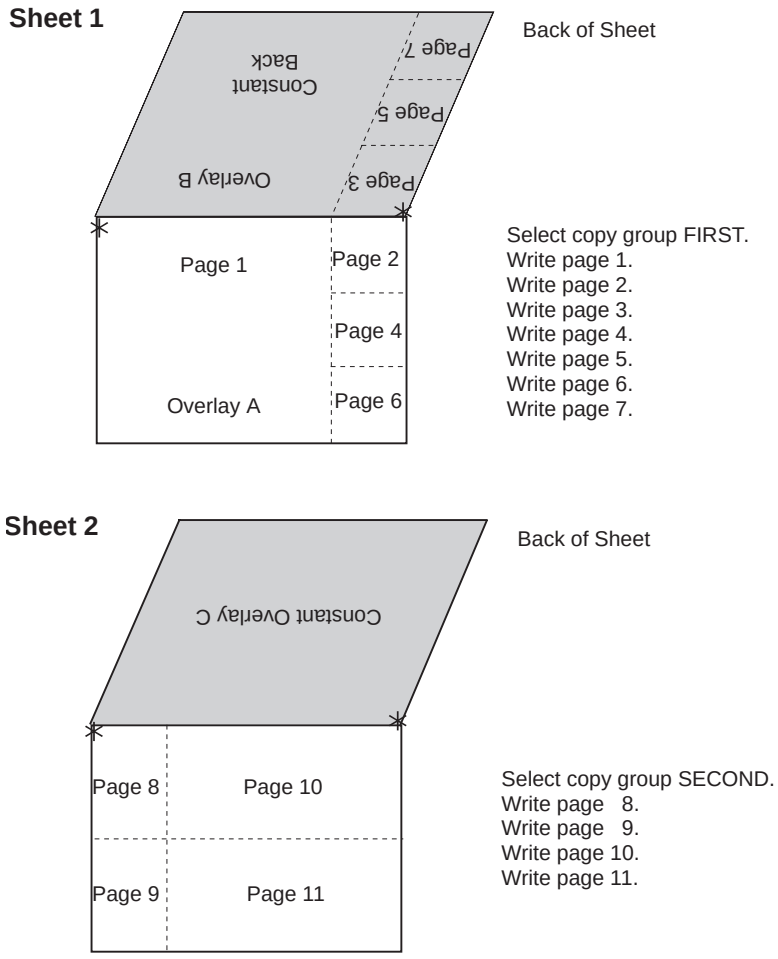


Figure 86. Enhanced N_UP Example 3: Asymmetric Pages

Figure 86 shows the flexibility and power of enhanced N_UP printing. With enhanced N_UP printing, you can place pages relative to any partition on the sheet, front or back, in any sequence. Pages are not limited by partition boundaries. The only limitations are that pages must not print outside the physical form boundaries, and you cannot place more pages on a sheet than the number specified in the N_UP subcommand. For an N_UP 4 duplex page, the limit is eight

pages total on front and back sides combined. For N_UP 3 duplex, the limit is six pages on the front and back combined.

```
FORMDEF ASYMET DUPLEX NORMAL ;

COPYGROUP FIRST
  PRESENT LANDSCAPE DIRECTION ACROSS
  N_UP 4
/* Constant*/   PLACE 1 BACK  OFFSET  4  0  CONSTANT OVERLAY B
/* Page  1 */   PLACE 1 FRONT OFFSET  0  0  Overlay A
/* Page  2 */   PLACE 1 FRONT OFFSET 12  0
/* Page  3 */   PLACE 1 BACK  OFFSET  0  0
/* Page  4 */   PLACE 1 FRONT OFFSET 12  4
/* Page  5 */   PLACE 1 BACK  OFFSET  0  4
/* Page  6 */   PLACE 1 FRONT OFFSET 12  8
/* Page  7 */   PLACE 1 BACK  OFFSET  0  8 ;

COPYGROUP SECOND
  PRESENT PORTRAIT DIRECTION ACROSS
  N_UP 3
/* Constant*/   PLACE 1 BACK  OFFSET  0  0  CONSTANT OVERLAY C
/* Page  8 */   PLACE 1 FRONT OFFSET  0  0
/* Page  9 */   PLACE 1 FRONT OFFSET  0  4
/* Page 10 */   PLACE 1 FRONT OFFSET  6  0
/* Page 11 */   PLACE 1 FRONT OFFSET  6  4
/* 6th place */ PLACE 1 BACK  OFFSET  0  0  CONSTANT ;
```

Figure 87. Form Definition for Enhanced N_UP Example 3

To achieve the asymmetrical page placement shown in this example, place all the pages relative to the origin of partition 1 on the front or the back side of the sheet. You can place the pages relative to the origin of any of the partitions, but using partition 1 simplifies the calculations for page positions.

With N_UP 4, the default PORTRAIT presentation and ACROSS direction place the origin at the top right of the partition on wide, continuous-form paper. In this example, specifying LANDSCAPE ACROSS sets the origin at the top-left corner, to achieve the correct page arrangement.

The coding of the form definition for example 3 is shown in Figure 87. Copy group FIRST specifies N_UP 4, which requires eight PLACE subcommands for the duplex page. Observe that the constant overlay B on the back of the sheet represents one of the eight PLACE subcommands. COPYGROUP SECOND used for the second sheet specifies N_UP 3. You must use six PLACE subcommands. Four pages are placed on the front side, and a constant overlay is placed on the back, using five of the six PLACE subcommands. A CONSTANT page is specified without an overlay to fill the sixth PLACE subcommand. Nothing will be printed with this PLACE subcommand, but it is required to ensure a correct internal page count for recovery and restart.

Note: In each copy group, the PLACE subcommand for the constant overlay is placed in front of all the PLACE subcommands for page data. This placement ensures that the constant overlay prints if any pages are printed on the sheet. Otherwise, if you change copy groups or run out of pages before the PLACE command for the constant overlay, the overlay will not print.

Additional N_UP Considerations

N_UP can affect the scope of other PPFA commands that operate on a page or a medium.

COPIES
The COPIES subcommand in the SUBGROUP of the form definition operates on the physical medium. When you specify five copies using N_UP 2, you will get five sheets of the N_UP 2 data.

SUPPRESSION
The SUPPRESSION subcommand in the SUBGROUP of the form definition operates on the physical medium. The suppression names in the SUBGROUP operate on all N_UP pages on the sheet.

OVERLAY
You can specify an OVERLAY subcommand in multiple places in the form definition and can also specify an overlay in the page definition. The result will be either a *page overlay* or a *medium overlay*. See "Medium Overlays and Page Overlays" for a description of the differences between these commands and the uses of these overlays.

PRESENT DIRECTION
You use the PRESENT and DIRECTION subcommands of the form definition with the N_UP subcommand to determine partition arrangement. These commands, which are described in this update guide, now affect all N_UP printers, including cut-sheet printers.

CONDITION
You can use the CONDITION command of the page definition with N_UP just as you use it with non N_UP jobs. However, the NEWSIDE and NEWFORM parameters may operate differently than you expect. NEWSIDE, which is equivalent to invoking a new page format, will eject to the next partition, which may not be on a new side of an N_UP sheet. NEWFORM, which is equivalent to invoking a new copy group, will eject to a new sheet with basic N_UP. The effect with enhanced N_UP depends on the coding of the INVOKE subcommand.

Medium Overlays and Page Overlays

An AFP overlay can be used as a *page overlay* or as a *medium overlay*. Different actions are performed on these two different types of overlays. Page overlays apply to the page and are placed relative to the page origin. Medium overlays always apply to the entire medium and are placed at the medium origin. When used with N_UP, the medium overlay still applies to the entire sheet of paper, not to the individual partitions.

The same overlay can be either a page overlay or a medium overlay, depending on the method used to invoke it for printing. An overlay invoked by a page definition or by an Include Page Overlay (IPO) structured field is always a page overlay. An overlay invoked by a form definition without N_UP is always a medium overlay. When N_UP is specified in the form definition, you can specify commands to invoke a page overlay. The examples below show the ways in which overlays can be invoked.

```
PAGEDEF EXMPL1 ;
PAGEFORMAT P2EXMPL1;
OVERLAY EXMPL1;        /* Allows this page overlay to be     */
                       /* invoked by an IPO structured field */
PRINTLINE REPEAT 60;   /* coded in the print data            */
```

Figure 88. Page Overlay Invoked by an IPO Structured Field

```
PAGEDEF EXMPL2 ;
PAGEFORMAT P2EXMPL2;
OVERLAY EXMPL2;        /* Optional. Stores overlay for reuse */
PRINTLINE REPEAT 1
  POSITION 1 IN 1 IN
  OVERLAY EXMPL2      /* Prints overlay if data prints on printline */
     -1 IN -1 IN ;    /* Positions overlay relative to printline    */
PRINTLINE REPEAT 50;
```

Figure 89. Page Overlay Invoked by a PRINTLINE Command

```
FORMDEF EXMPL3 ;
COPYGROUP F2EXMPL3
 DUPLEX NORMAL ;
 OVERLAY XMPL3F;       /* Allows SUBGROUP to invoke overlay   */
 OVERLAY XMPL3B;       /* Allows SUBGROUP to invoke overlay   */
 SUBGROUP FRONT
 OVERLAY XMPL3F;       /* Prints overlay on front of every form */
 SUBGROUP BACK
 OVERLAY XMPL3B;       /* Prints overlay on back of every form */
```

Figure 90. Medium Overlay Invoked by a Form Definition

```
FORMDEF EXMPL4 ;
COPYGROUP F2EXMPL4
 N_UP 2
 OVERLAY EXMPL4        /* Prints overlay with page in every  */
   0 0  ;              /* Partition at the page origin (0,0) */
```

Figure 91. Page Overlay in a Simple N_UP Form Definition

```
FORMDEF EXMPL5 ;
COPYGROUP F2EXMPL5
 N_UP 2
  PLACE 1
   OVERLAY XMPL51        /* Prints overlay in Partition 1     */
    0 0 PARTITION        /* Places it relative to Partition   */
  PLACE 2
   OVERLAY XMPL52        /* Prints overlay in Partition 2     */
    0 0 PARTITION ;      /* Places it relative to Partition   */
```

Figure 92. Page Overlay in an Enhanced N_UP Form Definition

N_UP Compared to Multiple-up

With the addition of the N_UP capability, AFP now provides two methods to format multiple application pages on a single sheet:

- N_UP as defined in a form definition
- Multiple-up as defined in a page definition

The multiple-up function has long been available for line-format data printed on AFP printers. Multiple-up achieves the *appearance* of multiple pages on a sheet by

formatting multiple groups of print lines as a single AFP page. The output is still a single AFP page on a side of a sheet, and the entire output is formatted by a single page format. If the application pages within that sheet require different print layouts, you must design a different page format for all possible arrangements of data. For example, if one side of a 2-up sheet has ten different print layouts, you need 100 different page formats to cover all the possible combinations.

In contrast, N_UP enables you, for the first time in AFP, to place *multiple AFP pages* on a side of a sheet. This means that each of the N_UP pages can be formatted using a different page format. You can change page formats between each N_UP page without ejecting to a new side of the sheet. For the same example with N_UP, you need only ten page formats for a 2-up sheet with ten different print layouts.

N_UP also means you can place multiple pages of fully-composed AFP data (or MO:DCA data) on a single sheet. This was not possible using the multiple-up function defined in the page definition, because AFP data does not use a page definition.

Part 3. PPFA Commands and Syntax

Chapter 8. PPFA Command Syntax

PPFA controls are made up of four elements: commands, subcommands, parameters, and literals.

- *Commands* are controls representing the major functions of PPFA and are separated from other commands by semicolons. Each command has its own entry in "Chapter 9. Form Definition Command Reference" on page 155 and in "Chapter 10. Page Definition Command Reference (Traditional)" on page 195.
- *Subcommands* fall within commands and specify the function of that command.
- *Parameters* specify the values for one subcommand.
- *Literals* consist of fixed text included in a field definition or as constant data for comparison in a conditional processing definition.

Rules for Creating a PPFA Command Stream

When you create a PPFA command stream, follow these rules:

- You cannot intermix uppercase and lowercase characters to write commands, subcommands, and literals. Before processing the commands, PPFA converts lowercase characters into uppercase characters, except those in literals. Thus, it does not discriminate between uppercase and lowercase characters. For example, **OVERLAY abc** and **overlay ABC** produce the same results because both **overlay** and **abc** are converted to uppercase.
- Commands and subcommands can be abbreviated to the first five characters, which are always unique. For example, PRINTLINE can be abbreviated to PRINT, FORMDEF to FORMD, CHANNEL to CHANN, and so forth.
- User names for form definitions and page definitions must not be the same as PPFA command names and subcommand names. These are reserved words. For a list of the reserved words, see "Appendix E. PPFA Keywords" on page 373. For example, REPEAT or CHANNEL must not be form-definition names.
- The subcommands governed by a command can be entered in any order; however, the name of a font or form definition, for example, must come immediately after the object being named. Parameters defined in a subcommand must be entered immediately after the subcommand.
- Commands must end with a semicolon.
- A command or subcommand can start in any column and can continue on the next line without a continuation indicator.
- More than one form definition and page definition can be specified in a job stream.
- PPFA neither checks nor sets default values for items that depend on printer hardware.

Token Rules

Tokens are character strings, within a set of PPFA commands, that PPFA recognizes as units. Tokens include:

- Both local names and user-access names for fonts, form definitions, page definitions, overlays, and suppressions
- Commands
- Subcommands

- Parameters
- Literals
- Special characters

The only PPFA element that is not a token is a blank. A token cannot be split between two lines.

To create a token, you must separate a string from the previous token by either a special character or a blank. See the list of special characters in “Character Set”. Thus, A+B is the same as A + B, because + is a special character. But AB is not the same as A B. The blank in A B creates two tokens.

Character Set

The four types of characters are alphabetic, numeric, blank, and special. Characters of each type are as follows:

- The following are PPFA alphabetic characters:

  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  a b c d e f g h i j k l m n o p q r s t u v w x y z
  # @ $

- The following are PPFA numeric characters:

  0 1 2 3 4 5 6 7 8 9

- The blank character has a character code of X'20' in ASCII (which is the data stream used for creating the form definition or page definition

  Note: In EBCDIC data, the blank character has a character code of X'40'.

- The following are PPFA special characters:

  . (+ *) – % ’ = ; / &

- The following are EBCDIC shift-out and shift-in codes:
  X'0E', the shift-out (SO) code
  X'0F', the shift-in (SI) code

Other character codes are also allowed within comments and literals. See “Comments” on page 151 and “Literals” on page 152 for details of what can be included.

Command Delimiters

A command always ends with a semicolon. One command can extend over several lines and does not end until a semicolon appears.

Blanks and Blank Lines

Blanks and blank lines can occur anywhere and have no effect on the processing of PPFA. The “;” is the command delimiter.

Names

The maximum number of alphanumeric characters in a PPFA name varies. Table 7 on page 151 shows the number of characters allowed in the PPFA names.

Table 7. Character Length for PPFA Names

Type of Name	Number of Characters Allowed
Form Definition	
COPYGROUP	1–8
FORMDEF	1–6
OVERLAY (local name)	1–16
OVERLAY (user-access name)	1–6
SUPPRESSION	1–8
Page Definition	
BARCODE	1–8
CONDITION	1–8
DEFINE COLOR	1–10
FONT (local name)	1–16
FONT (user-access name)	1–6
OBJECT (internal-name)	1–16
OBJECT	1–8
OVERLAY	1–6
PAGEDEF	1–6
PAGEFORMAT	1–8
SEGMENT	1–6
Note: The *name* can consist of the characters shown under the "Alphabetic Characters" and the "Numeric Characters" in "Character Set" on page 150. A *local name* identifies a font or an overlay only within a set of PPFA commands. A *user-access* name is the name the operating system uses to find the data. PPFA adds the appropriate prefix to the user-access name (for example, F1 for form definitions, P1 for page definitions, and O1 for overlays) to match the library resource name.	

Two PPFA commands can be used to equate the local name and the user-access name.

OVERLAY Within a form definition
FONT Within a page definition

Comments

Programmer comments used to document PPFA command streams are allowed anywhere within the command stream. Comments must be enclosed with the delimiters /* and */ . A comment is allowed anywhere a blank is allowed and can continue for any number of lines.

Note: For VSE, however, a comment must not start at the beginning of the line. A /* specified as the first two bytes of a record in PPFA running under VSE is interpreted as the end of system input.

The following example shows the available variations in comment formats:

```
FIELD /* comment */ FONT GT10 /* comment,
  multiline comment,
  more comment */ START * + 10  LENGTH 5 ;
FIELD LENGTH 10 ; FIELD START * + 10  LENGTH 15 ;
```

Notes:

1. A comment must end with the closing delimiter (*/) .
2. Double-byte character codes in comments must be enclosed within SO (X'0E') and SI (X'0F') on EBCDIC platforms.

Literals

A literal is any material specified in single quotation marks. Literals can be used within a:

- TEXT subcommand to create fixed text for a page definition
- WHEN subcommand to define constant text for comparison

Literals can contain any characters in any position, except those that have special syntactic meanings. Single quotation marks may be used within a literal only if they are entered in pairs ('). PPFA translates a pair of single quotation marks into one quotation mark. For example, 'JOAN''S' yields JOAN'S.

A literal can continue for any number of lines. For example:

```
TEXT 'THIS IS ' 'A LITERAL' /* The four separated        */
     'THE TEXT SPANS'       /* text elements will produce*/
     'THREE LINES'  ;       /* one sequence of text      */

TEXT X'0101'                /* Hexadecimal literals      */
     X'ABAB'                /* spanning three lines      */
     X'BBBB'        ;

TEXT K'100,200'             /* kanji numbers             */
     K'321,400'     ;       /* specified sequentially    */
```

Invalid:

```
TEXT 'THIS IS'
    K'100,200'      ;       /* Mixing single-byte and
                               double-byte characters in one
                               field is not allowed */
```

A double-byte literal must be enclosed within apostrophe shift-out (X'7D0E') and shift-in apostrophe (X'0F7D').

Numeric Values

Numeric variables are specified as decimal numbers; up to three decimal places can be specified.

Units of Measurement

Numbers used to specify dimensions in form definitions and page definitions can be in any of five units of measurement. They are specified in a command stream as follows:

- IN (inches)
- MM (millimeters)
- CM (centimeters)
- POINTS — Points are a common measurement in printing used to measure character height, as in 20-point type. A point is approximately 1/72 inch.
- PELS (equates to L-units)

  The number of pels per inch is a user-specified parameter. The default is 240 pels per inch.

Two additional measurement units can be used in the SETUNITS command; the measurement units are:
- LPI (lines per inch)
- CPI (characters per inch)

The parameters in PPFA that define a measurement can include any of the first five units of measurement shown in the previous list. For example:

```
POSITION  1 IN  1 IN ;
        or
POSITION  1 MM  1 MM ;
```

However, PPFA converts all measurements to logical units (L-units) as the common measurement. (Normally, one inch equals 240 L-units, but this number can be changed by the user.) If a fraction exists, the first decimal point is truncated. A SETUNITS command defines a unit of measurement that is to be used as the default for any parameter that does not specify a given dimension. This default is in effect until another SETUNITS command is encountered. This example:

```
SETUNITS  1 IN 1 IN ;
 .
 .
 .
POSITION (or OFFSET or LINEONE) 1 1 ;
```

shows part of a PPFA command stream in which a SETUNITS command sets the units of measurement to one inch for a subsequent POSITION (or OFFSET or LINEONE) subcommand.

SETUNITS can be used as a multiplier:

```
SETUNITS 2 IN 2 IN ;
 .
 .
 .
POSITION 2 2 ;
```

In this example, the SETUNITS command sets two-inch *x* and *y* default values. The POSITION subcommand values are multiplied by the default values creating a position four inches horizontally and four inches vertically from a given reference point. See “SETUNITS Command” on page 188 for a more detailed explanation.

Diagram Shorthand

These terms are used in the command definitions:

x-pos A horizontal position using a numeric number followed optionally by a unit. For the available units, see “Units of Measurement” on page 152.

y-pos A vertical position using a numeric number followed optionally by a unit. For the available units, see “Units of Measurement” on page 152.

Chapter 9. Form Definition Command Reference

This section includes:
- Sequence of commands for form definitions
- Form definition commands listed alphabetically
- Detailed information on each command
- Descriptions of the applicable subcommands and parameters for each command

Sequence of Commands for Form Definitions

```
[SETUNITS ]
FORMDEF
[SUPPRESSION ...]
[COPYGROUP ]
  [OVERLAY ...]
  [SUBGROUP ...]
[COPYGROUP ]
  [OVERLAY ...]
  [SUBGROUP ...]
```

1. SUPPRESSION commands must be specified immediately after FORMDEF commands. The exception is the SETUNITS command (see item 5).
2. One file can contain multiple sets of form definitions.
3. OVERLAY and SUBGROUP commands must be specified under their associated COPYGROUP command. The OVERLAY commands must be specified immediately after a COPYGROUP command.
   - The OVERLAY command is required only to designate an overlay that is to be kept in the 3800 printer as raster data, or to specify a local name for referencing an overlay in a SUBGROUP command. If you do not code the OVERLAY command, you can still specify an overlay in a SUBGROUP command using its user-access name.
   - Overlays also may be specified using the N_UP subcommand of the FORMDEF or COPYGROUP command, or using the PRINTLINE command in the page definition. If the overlay is specified in one of these ways, it should also not be coded on the OVERLAY or SUBGROUP commands shown here. For more information, see “Medium Overlays and Page Overlays” on page 144.

   Notes:

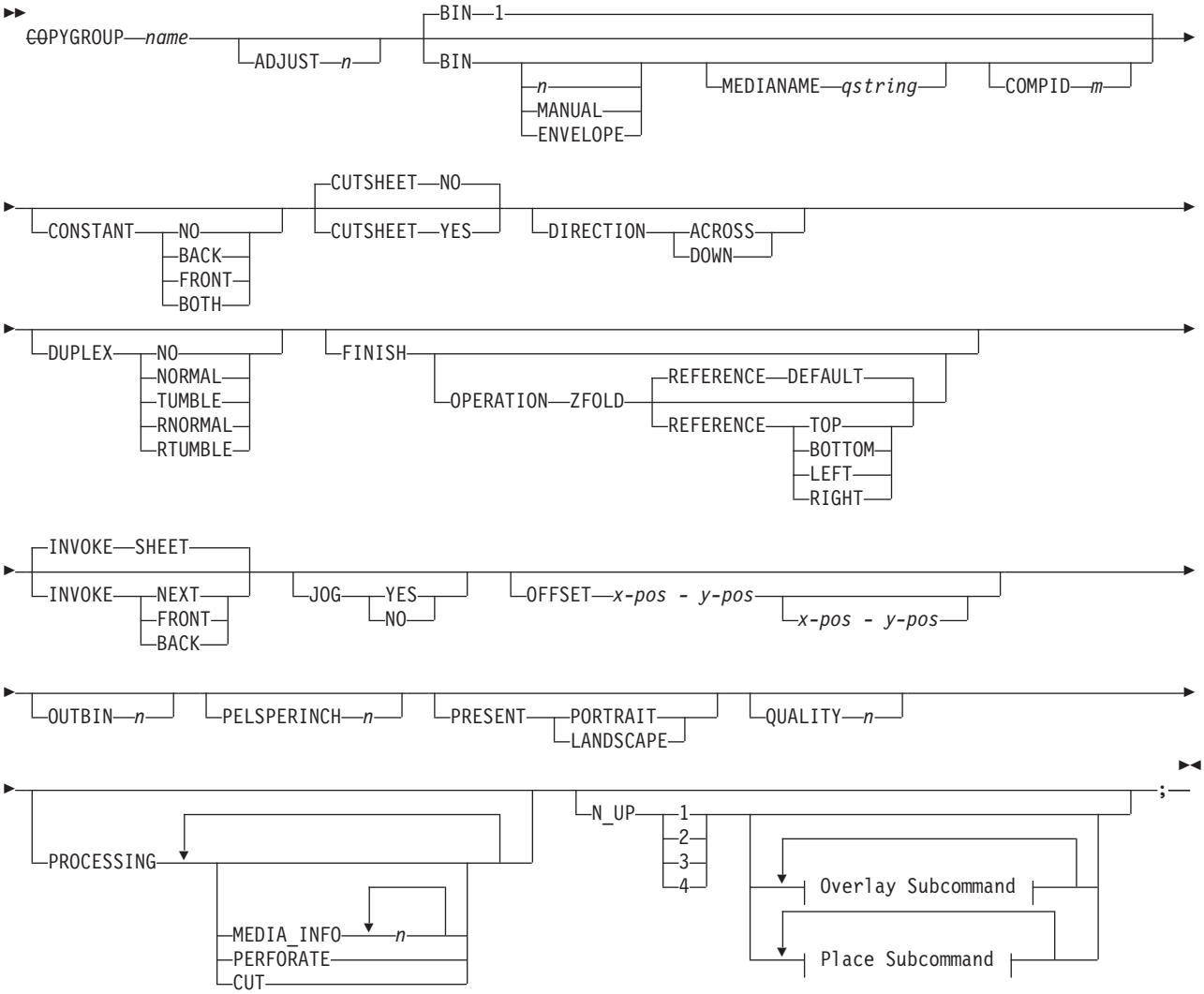
   a. If the form definition has only one copy group, the COPYGROUP command can be omitted. The OVERLAY command then follows any SUPPRESSION command.

   b. The appearance of a misplaced OVERLAY command prior to the first COPYGROUP command will cause a default copygroup to be generated as the first copygroup.
4. The first COPYGROUP command can be omitted in a form definition if it contains only one copy group and no OVERLAY commands. If it is omitted, the FORMDEF command parameters are used to define the copy group.
5. A SETUNITS command can be placed before any PPFA command. The values set are in effect until the next SETUNITS command.
6. Each command can appear more than once under one FORMDEF command.

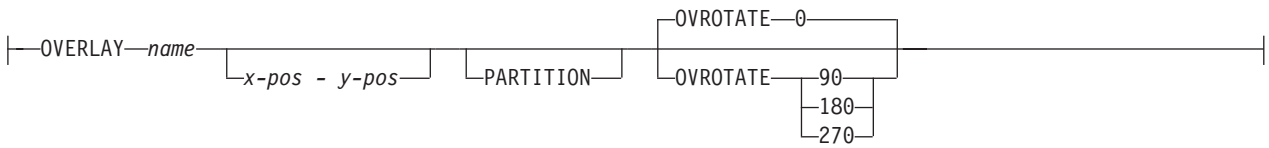
7. To do an INSERT finishing task, select a COPYGROUP that specifies the dedicated INSERT bin number[3] from which the pages are to be inserted and apply (usually dummy) print data to that page. Observe that nothing will be printed on the inserted page.

3. The INSERT bin number is printer specific. See the documentation for the specific printer being used.

COPYGROUP Command

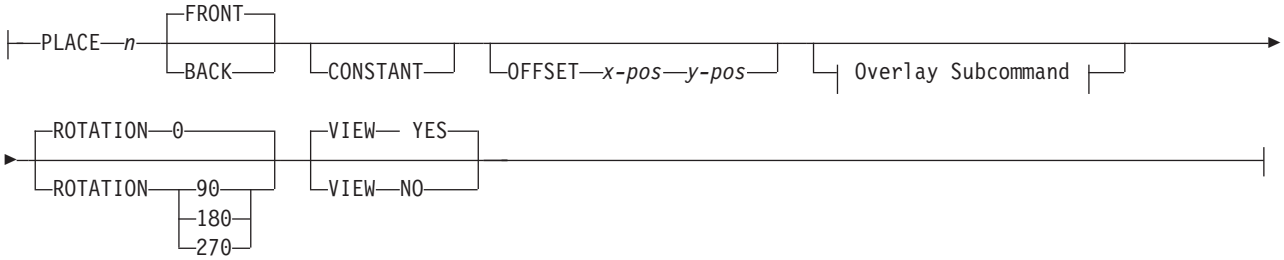


Overlay Subcommand:



COPYGROUP

*** Place Subcommand:**



*** The use of the PLACE subcommand indicates enhanced N_UP printing.**

Copy groups are subsets of a form definition. A form definition can contain one or several copy groups. Copy groups are nested within a form definition following any SUPPRESSION command. COPYGROUP subcommands have no fixed defaults; if any subcommand is omitted, its value is selected from the corresponding subcommand in the FORMDEF command.

Notes:

1. Subsets of copy groups are called subgroups.
2. If you specified DUPLEX NO anywhere in the copy group, output will be simplex regardless of any other DUPLEX subcommand within the same copy group.
3. If a form definition has only one copy group, the COPYGROUP command can be omitted. If omitted, a name is automatically assigned by PPFA to the copy group, using the form definition resource name, including the F1 prefix. All values for the copy group are given the values from the FORMDEF command and subcommands. You need to know this name should you use conditional processing and need to invoke this copy group by name. Copy groups are placed within the form definition in the order in which they are generated.

To change copy groups during formatting, use conditional processing.

Note: Another way to change copy groups after the resource is stored is to insert an Invoke Medium Map structured field into your print data file (copy groups are known to print server as medium maps). If no Invoke Medium Map structured field is found and no conditional processing is being performed, the first copy group in the form definition is used for the job.

COPYGROUP *name*
Defines an alphanumeric name of 1–8 characters. This name must be unique in a single form definition. If any names are duplicated, PPFA issues an error message and does not create the form definition.

Subcommands

ADJUST *n* **(3800 Printers Only)**
Establishes the range of horizontal adjustment for the print area on the sheet.

n The adjustment range can be set from 0 to 20 L-units. After a value is set, it is the maximum amount available in both directions, plus and minus.

Note: If you specify ADJUST, the maximum logical page size (in the horizontal direction) is reduced by the amount you specified here.

BIN parameter
Specifies the paper source. This subcommand should be used only for printers that have more than one paper source.

1 Selects the primary paper source.

2–255 Selects another paper source. If the specified bin does not exist on your printer, the default paper source for that printer will be used. For more information about paper sources on your printer, refer to your printer publications.

MANUAL
Selects manual feed as a paper source on those printers that support manual feed. For more information, refer to your printer publications.

ENVELOPE

Selects an envelope paper source on those printers that support this function. For more information, refer to your printer publications.

MEDIANAME

Selects a media source by specifying an agreed-upon name for the bin.

qstring

Up to 12 characters within single quotes, specifying the media source name. On some printers, this name is pre-set into the printer; on other printers, it can also be entered into the printer by the user. For a current list of the valid media names, see "Appendix F. PPFA Media Names" on page 375. Refer to your printer publications for further information.

Notes:

1. BIN selection is overridden by the printer if the form defined to each bin is the same form number. Only the primary bin is selected.
2. The primary source usually contains either letter-size (U.S.) or A4 (I.S.O.) paper. Other paper sources are used for less common paper sizes (such as legal-size) and for special paper (such as colored stock or pre-printed letterhead on heavy bond).
3. If duplexing is requested and you select from the front side from one bin and the back side from another bin, a warning message will be issued and the printer will take the paper from the bin specified on the front side.

COMPID
Selects a bin based on the component id.

m For a current list of component ids, see "Appendix F. PPFA Media Names" on page 375. Component ids from 12,288 to 268,435,455 are reserved for the user.

CONSTANT
Specifies whether the constant-forms function is on or off and whether constant form is to be printed on the front or back sides of a sheet.

NO Specifies that the constant forms function is off.

BACK Specifies that a constant form is to be printed on the back side without variable data.

FRONT
Specifies that a constant form is to be printed on the front side without variable data.

BOTH Specifies that a constant form is to be printed on both sides without variable data.

CUTSHEET
If you are using a cut-sheet printer, this subcommand specifies whether the medium orientation information, using the DIRECTION and/or PRESENT subcommands, is to be passed to the printer. The default value is NO.

YES Specifies the rotation data is to be passed.

NO Specifies the rotation data is not to be passed unless N_UP is coded.

Notes:

1. If you have a continuous form printer, the medium orientation information is passed. If you have a cut-sheet printer and N_UP is coded, the orientation information is passed.
2. If you have a cut-sheet printer and CUTSHEET YES is coded, the orientation information will be passed, providing you also have a level of print server that supports that feature.
3. You must have a printer that allows its media origin to be changed in order to use this subcommand.

Example:

In the following example, the CUTSHEET subcommand is coded on the form definition to give copygroups c1 and c2 ″CUTSHEET YES″ behavior and copygroup c3 ″CUTSHEET NO″ behavior. The copygroup c1 inherits its behavior from the form definition.

```
FORMDEF cut1 REPLACE YES CUTSHEET YES;
     COPYGROUP c1 ;
     COPYGROUP c2 CUTSHEET YES ;
     COPYGROUP c3 CUTSHEET NO ;
```

DIRECTION
Determines, along with the PRESENT subcommand, how data is oriented on printers whose media origin can be changed. See the list of printers “Chapter 7. N_UP Printing” on page 127.

If you are printing line data, you usually specify the same value for the DIRECTION subcommand as is specified for the DIRECTION subcommand in the page definition.

ACROSS
Specifies that the pages are formatted in the ACROSS printing direction.

DOWN
Specifies that the pages are formatted in the DOWN printing direction.

If the DIRECTION subcommand is specified, you must specify the PRESENT subcommand. The default for DIRECTION is determined by the value specified for PRESENT.

The direction default of PORTRAIT is ACROSS; the direction default of LANDSCAPE is DOWN. If neither PRESENT nor DIRECTION is specified, the default is PRESENT PORTRAIT and DIRECTION ACROSS.

DUPLEX

Specifies whether printing is done on both sides of the sheet. This subcommand should be used only for page printers that have duplex capability.

NO Duplex printing is not performed.

NORMAL

Duplex printing is performed, with the tops of both sides printed along the same edge for side binding.

TUMBLE

Duplex printing is performed with the top of one side and the bottom of the other printed along the same edge of the sheet for top binding.

RNORMAL

Rotated normal. Duplex printing is performed with the tops of both sides printed along the same edge. Used with landscape pages, N_UP 2, and N_UP 3.

RTUMBLE

Rotated tumble. Duplex printing is performed with the top of one side printed along the same edge of the sheet as the bottom of the other. Used with landscape pages, N_UP 2, and N_UP 3.

FINISH

A finishing operation is to be performed on this COPYGROUP. This option is to be used only on a document, set of documents, or an entire print file.

OPERATION ZFOLD

OPERATION specifies the FINISH operation type. Only the **ZFOLD** attribute is allowed on the COPYGROUP command.

Note: The default for OPERATION is ZFOLD. It is necessary to code OPERATION only if REFERENCE is coded.

ZFOLD

Do a Z-FOLD operation along the finishing edge (axis). Z-Folding causes the sheet to first be folded in half inwards (the front side of the sheet is now inside the fold) along a line parallel to the reference edge. The half of the sheet originally furthest from the reference edge is again folded in half outwards along a line parallel to the reference edge. For example, when Z-Folding is applied to an 11 by 17 inch sheet with the reference edge along a short side, the result is an 8.5 by 11 inch fold-out.

REFERENCE

Select the FINISH operation reference edge. The REFERENCE subcommand is optional and, when omitted, the DEFAULT attribute is the default.

DEFAULT

Specifies that, for the FINISH operation, the device default edge determines the reference edge.

TOP
Specifies that, for the FINISH operation, the reference is positioned along the top edge.

BOTTOM
Specifies that, for the FINISH operation, the reference edge is positioned along the bottom edge.

RIGHT
Specifies that, for the FINISH operation, the reference edge is positioned along the right edge.

LEFT
Specifies that, for the FINISH operation, the reference edge is positioned along the left edge.

Notes:

1. The default OPERATION is ZFOLD, and the default REFERENCE is DEFAULT.
2. The finishing OPERATION subcommand may be specified only once on a COPYGROUP.
3. For the finishing operation, changing the orientation of the medium presentation space does not change the finishing position. For instance the finishing reference edge (corner) is not affected by DIRECTION or PRESENT values.

The following are examples of the finishing operations:

1. ZFOLD pages (for which the xyz COPYGROUP is in effect), specifying the left edge of the document as the reference edge:

```
COPYGROUP xyz
  FINISH OPERATION ZFOLD REFERENCE LEFT
    ;
```

2. Three examples of ZFOLD pages that specify the default edge of the document:

```
COPYGROUP uvw FINISH;
```

or

```
COPYGROUP uvw FINISH OPERATION ZFOLD;
```

or

```
COPYGROUP uvw FINISH OPERATION ZFOLD REFERENCE DEFAULT;
```

INVOKE
Specifies where the next page of data is placed when this copy group is activated by conditional processing or by an Invoke Medium Map structured field.

INVOKE SHEET, which is the default, places the next page of data on a new sheet. The NEXT, FRONT, and BACK parameters place the next page in a subsequent partition on the same sheet or, if no partitions are available, on the next sheet. If FRONT or BACK is specified, INVOKE selects only partitions on the front or back, respectively.

Print server honors the NEXT, FRONT, and BACK values of the INVOKE subcommand only if the new copy group has the same medium modifications as the previous copy group. Medium modifications include duplexing, bin, page offset, N_UP values, presentation, direction, and medium overlays.

If any of these modifications differ, print server ejects to a new sheet when the copy group is invoked. If you want to change overlays when ejecting to a new partition, use page overlays instead of medium overlays. See "Medium Overlays and Page Overlays" on page 144 for information about page and medium overlays.

When you use PLACE subcommands, the NEXT, FRONT, and BACK parameters place the next page using the next sequential PLACE subcommand that matches the requirement (next, front, or back). For example, if you print using the second PLACE subcommand of copy group A, and then you change to copy group B, you will start with the third PLACE subcommand of copy group B.

A CONSTANT parameter on the PLACE subcommand does not alter the selection process. The selection is complete, even though the selected PLACE subcommand does not place the data. N_UP performs the constant modification and continues until it finds a PLACE subcommand that does not specify CONSTANT. The data is placed with this subcommand. Observe that this PLACE subcommand need not match the FRONT or BACK specifications of the INVOKE subcommand.

SHEET Specifies that data be placed in the first selected partition of the sheet.

NEXT Specifies that data be placed in the next selected partition.

FRONT Specifies that data be placed in the next selected front partition.

BACK Specifies that data be placed in the next selected back partition.

JOG

Specifies whether a JOG subcommand is sent to the printer when this COPYGROUP is selected by an IMM structured field, or through conditional processing. When the JOG subcommand is sent, a printer will either offset (jog) or print copymarks. For cut-sheet printers, or for continuous-forms printers with burster-trimmer-stacker enabled, the JOG subcommand causes the first sheet controlled by this COPYGROUP to be stacked offset from the previous sheets. For continuous forms printers without a burster-trimmer-stacker, the JOG subcommand causes an increment in the copymark printed on the carrier strip. JOG subcommands also are sent to the printer at the beginning of each data set or at the beginning of each job, depending on host parameters. For more information about copymarks, see the system programming guide for your host print server.

YES Specifies that a JOG subcommand be sent to the printer. The first sheet printed is offset or the copymark is incremented.

NO Specifies that no JOG subcommand be sent to the printer. The first sheet printed is not offset; the copymark is not incremented.

OFFSET

Specifies the offset of the logical page for both the front and back pages in reference to the media origin. The media origin is printer dependent. For more information about media origin, see your printer publications or *Advanced Function Presentation: Printer Information*.

If you specify offset values for the back of the page, you must also specify the front offset values.

Note: The OFFSET subcommand does not affect the position of medium overlays.

x-pos Specifies the horizontal offset of the logical page on the front or back side of the copy group relative to the media origin. The valid options for *x-pos* are described in the SETUNITS command for the horizontal value.

If no unit is specified, a default setting is:
- Taken from the last SETUNITS command
- **IN** (inch) if no SETUNITS command has been issued

y-pos Specifies the vertical offset for the logical page for the front or back side of the page. The valid options for *y-pos* are described in the SETUNITS command for the vertical value.

Note: The vertical offset for the 3800 must be 0.5 inch or greater.

If no unit is specified, a default setting is:
- Taken from the last SETUNITS command
- **IN** (inch) if no SETUNITS command has been issued

Note: If OFFSET is *not* specified, the OFFSET default is **0.1 IN 0.1 IN**.

OUTBIN *n*

Specifies the destination bin number for any pages directed by this COPYGROUP. Subgroups in this form definition that do not specify an output bin number inherit this one.

n the output bin number

PELSPERINCH *n*

Specifies the Logical Units in pels per inch for this COPYGROUP. Use the PELSPERINCH parameter to tell PPFA the pel resolution of your printer to generate more exact object placements.

n Specifies an integer number between 1 and 3,276, which determines the Logical Units in pels per inch.

Note: If the L-Units are not specified on the copy group, they are inherited from the form definition. See Figure 94 on page 180 for more information.

PRESENT

Specifies, along with the DIRECTION subcommand, how the data is oriented on printers whose media origin can be changed.

The PRESENT and DIRECTION subcommands are only supported by cut-sheet printers when you specify the N_UP subcommand or the CUTSHEET subcommand with the YES parameter. See Figure 69 on page 128 through Figure 72 on page 131 to determine the effect of the PRESENT and DIRECTION subcommands when you use them with the N_UP subcommand.

PORTRAIT

Specifies that the pages are printed in the portrait page presentation, with their short edges at the top and bottom and their long edges at the sides.

LANDSCAPE
Specifies that the pages are printed in the landscape page presentation, with their long edges at the top and bottom and their short edges at the sides.

QUALITY *n*
Specifies the print quality. This subcommand is recognized only on printers that can produce more than one level of print quality. The default is determined by the printer model. (On some printers, the default may be set at the printer itself.) For more information, refer to your printer publications.

n You can select a level of print quality by entering any whole number from 1 to 10. Higher numbers correspond to higher levels of print quality; lower numbers correspond to lower levels. For more information, refer to your printer publications.

Print quality is determined by a numerical code in the range of 1 to 254 (hexadecimal X'01'–X'FE'). The codes corresponding to the possible QUALITY parameters are:

1 = 15 (X'0F')
2 = 40 (X'28')
3 = 65 (X'41')
4 = 90 (X'5A')
5 = 115 (X'73')
6 = 140 (X'8C')
7 = 165 (X'A5')
8 = 190 (X'BE')
9 = 215 (X'D7')
10 = 240 (X'F0')

PROCESSING
Specifies additional post processing capabilities for selected printers and attached equipment. This option can only be used on a single sheet or collection of sheets. This subcommand expects 1 to 3 of the following keywords:

MEDIA_INFO *n*
This parameter specifies the ID of fixed medium information that a printer or printer–attached device applies to a sheet. Examples include color plates logos, letter heads, and other fixed images.

The numeric values that can be included are:

0–254 These numeric values select a particular fixed medium local ID that the printer or printer–attached device applies to a sheet. One or more IDs can be specified within this range.

255 This value selects all the current fixed medium local IDs that the printer or printer–attached devices applies to a sheet.

PERFORATE
Specifies a perforation cut at one or more fixed locations on the sheet according to the printer or printer–attached device.

CUT
Specifies a separation cut at one or more fixed locations on the sheet according to the printer or printer–attached device.

N_UP { 1 | 2 | 3 | 4 }

Specifies the number (1, 2, 3, or 4) of equal-size partitions into which the sheet will be divided. See page 127 for a list of printers that support the N_UP subcommand.

If you do not specify the N_UP subcommand in the COPYGROUP command, the N_UP subcommand from the FORMDEF command is the default for the COPYGROUP command. You can mix N_UP printing and non-N_UP printing by specifying or not specifying the N_UP subcommand in each copy group and by *not* specifying N_UP in the FORMDEF command.

OVERLAY

name Specifies the user access name (up to six characters) of an overlay to be placed with every page in each of the N_UP partitions. You can specify a maximum of 254 OVERLAY subcommands in a copy group.

Note:

- The prefix 'O1' is not part of the six-character user-access name. The overlay name can be an alphanumeric.
- This name is not related to names as defined on the OVERLAY command.

x-pos y-pos

Specifies the horizontal and vertical adjustment to the position of the overlay. This is in addition to any offset values built into the overlay. The x and y values must be positive (+). You can specify them in inches (IN), millimeters (MM), centimeters (CM), POINTS, or PELS. If you do not specify a unit value, PPFA uses the unit value specified in the last SETUNITS command or uses a default unit value of inches.

Note: This OVERLAY subcommand cannot be specified if the PLACE subcommand is specified.

PARTITION

Specifies that the overlay is to be placed relative to the partition origin.

OVROTATE

Specifies the rotation of the placed overlay with respect to the x-axis of the page.

Example:

Assuming the overlay has (0,0) placement coordinates, this causes page overlay "01x2" to be placed 1.5 inches to the right and 2.7 inches below the beginning of the page and rotated 90 degrees clockwise with respect to the page.

```
Formdef xmp1
     N_UP 1   PLACE 1 FRONT
              OVERLAY x2  1.5 in  2.7 in
              OVROTATE 90;
```

PLACE

Places a page of data or a constant modification relative to a partition. Each PLACE subcommand specifies the number *n* of a partition on either the front or back side of the sheet. FRONT is the default, if you do not specify this subcommand. You must specify the same number of PLACE

subcommands as the number of partitions on the sheet. The sequence of the PLACE subcommands is the sequence in which incoming pages will be placed in the partitions.

Note: The PLACE subcommand is valid only on printers that support enhanced N_UP printing. If PLACE is not specified, pages are placed in partitions in the default partition sequence.

n Specifies the numbered partition (1–4) into which the page of data is placed. See Figure 69 on page 128 through Figure 72 on page 131 for the locale of each numbered partition.

FRONT
Specifies that this partition be placed on the front side of the sheet.

BACK Specifies that this partition be placed on the back side of the sheet.

CONSTANT
Specifies that no page data will be placed by this PLACE subcommand.

Use CONSTANT when you are placing overlays without user's data or are placing fewer data pages on the sheet than the number of partitions specified in the N_UP subcommand.

For an example of using the CONSTANT parameter with overlays and to understand how the ordering of the PLACE subcommand affects overlays, see "Enhanced N_UP Example 3: Asymmetric Pages" on page 142 .

OFFSET
Specifies a positive offset of the page horizontally (x) and vertically (y) from the partition origin.

x-pos y-pos
The default value is 0.1 inch for both x and y offsets. This OFFSET parameter overrides any other OFFSET parameters specified on the FORMDEF or COPYGROUP command. You can specify the units in inches (in), millimeters (mm), centimeters (cm), points, or pels. If you do not specify a unit value, PPFA uses the unit value specified in the last SETUNITS command or uses a default unit value of inches.

OVERLAY

name Specifies the user access name (up to six characters) of an overlay to be placed with this PLACE subcommand. The overlay is placed relative to the page origin or, if the PARTITION keyword is specified, to the partition origin. You can specify multiple OVERLAY parameters in each PLACE subcommand.

Note: This OVERLAY subcommand cannot be specified if the PLACE subcommand is specified.

x-pos y-pos
Specifies the horizontal and vertical adjustment to the position of the overlay. This is in addition to any offset values built into the overlay. The x and y values must be positive (+). You can specify them in inches (in), millimeters (mm), centimeters (cm),

points, or pels. If you do not specify a unit value, PPFA uses the unit value specified in the last SETUNITS command or uses a default value of inches.

PARTITION
Specifies that the previous offset is from the partition origin. If not present, the offset is from the page origin, which is subject to the OFFSET parameter.

OVROTATE { 0 | 90 | 180 | 270 }
Specifies the rotation of the placed overlay with respect to the x-axis of the page.

ROTATION{ 0 | 90 | 180 | 270 }
Specifies the clockwise rotation of the page and associated page overlays placed by this PLACE command.

Rotation turns the page and its associated page overlays around their fixed origin points. If you rotate the page without moving its origin point, you might rotate it off the physical medium. To prevent this, always offset the page origin to the place you want it to be for the rotated page, as shown in Figure 93.

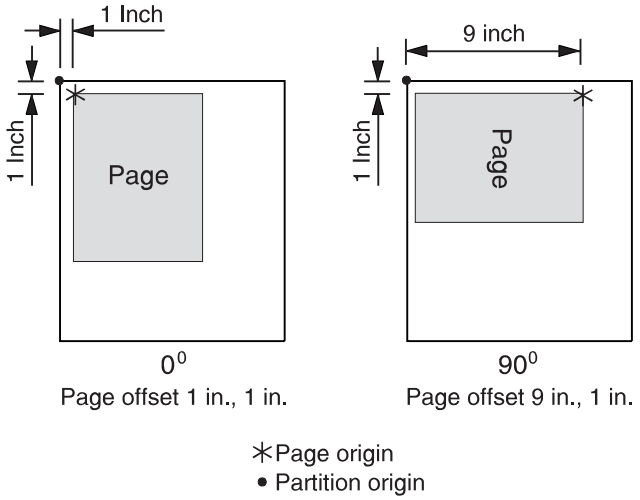


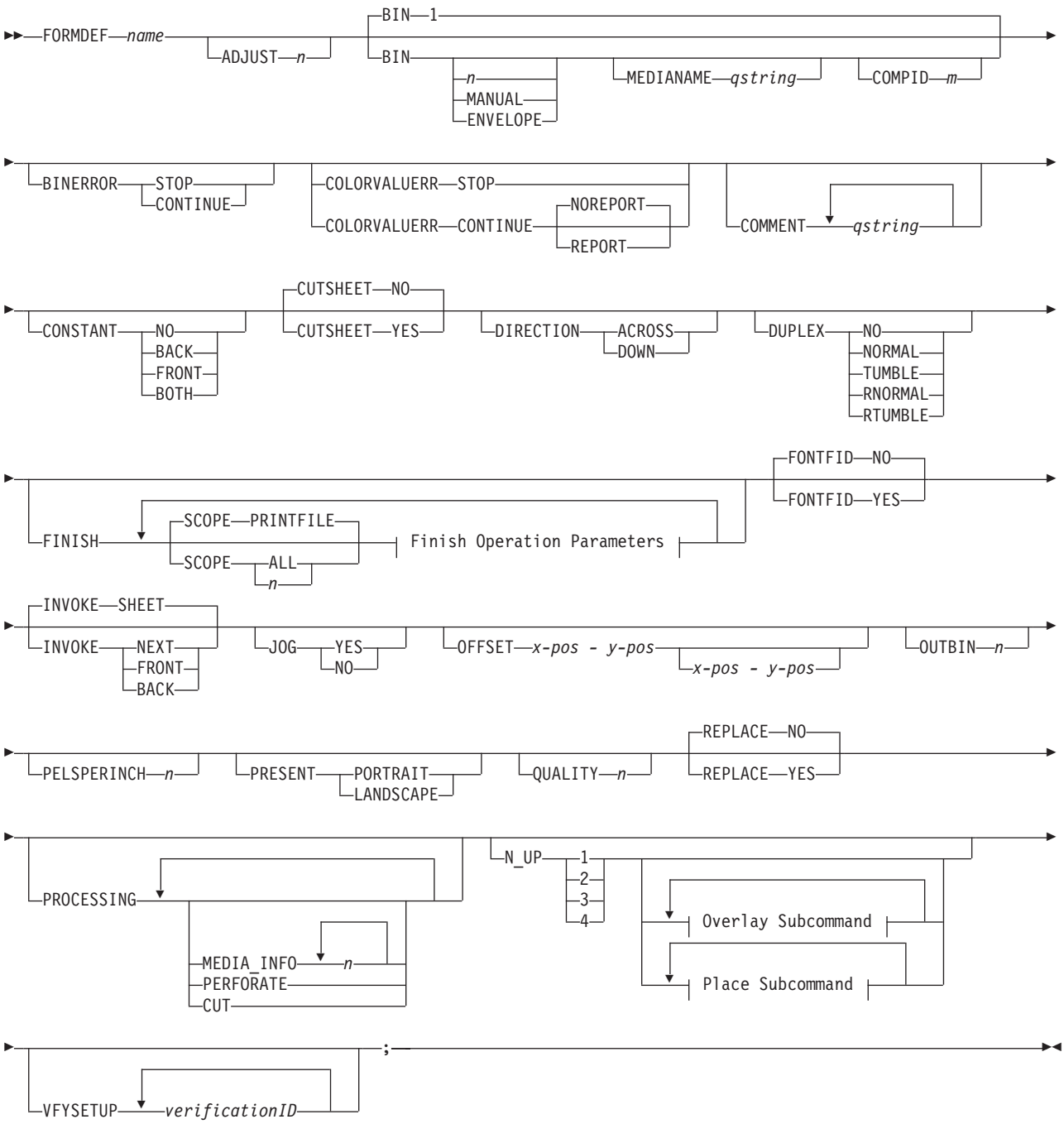
Figure 93. Offsetting the Page Origin for Rotated Pages

VIEW
Determines if this N_UP PLACE page is viewable. VIEW is relevant only when the page is being presented on a display. VIEW is ignored if the page is being printed. If VIEW is not coded, it is equivalent to specifying VIEW YES.

YES Specifies that this N_UP page is viewable and will be presented.

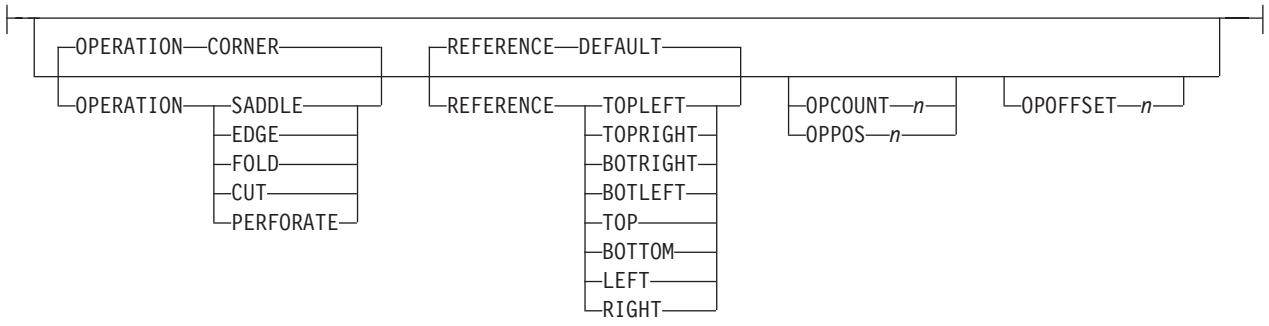
NO Specifies that this N_UP page is not to be presented.

FORMDEF Command

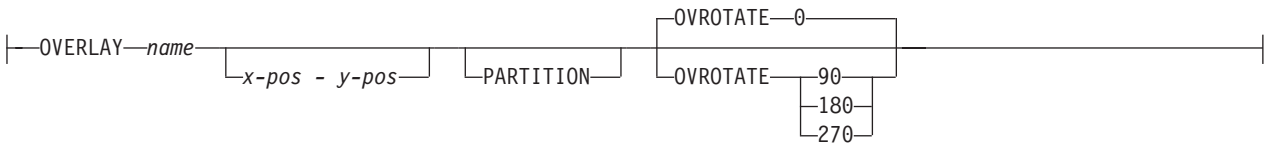


FORMDEF

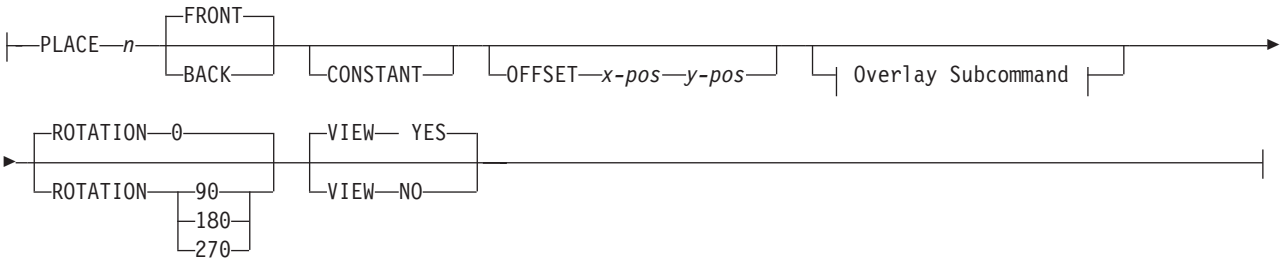
Finish Operation Parameters:



Overlay Subcommand:



*** Place Subcommand:**



*** The use of the PLACE subcommand indicates enhanced N_UP printing.**

A form definition is a resource that contains all the controls relating to the physical sheet. A FORMDEF command must be specified when you define a new form definition. When subcommands (except for the REPLACE, PRESENT, and DIRECTION subcommands) are specified, they become the defaults for all COPYGROUP commands nested within this form definition.

FORMDEF
Identifies the form definition to be used with the print job.

name Defines an alphanumeric name of 1–8 characters for the form definition. When you create a form definition, PPFA assigns a prefix of F1 to the name you specify. F1nnnnnn is the external resource name in the form-definition library.

Subcommands

ADJUST *n* **(3800 Printers Only)**
Establishes the range of horizontal adjustment for the printed area on the sheet. The default is 0.

n The adjustment range can be set from 0 to 20 L-units. After a value is set, it is the maximum amount available in both directions, plus and minus.

Note: If you specify ADJUST, the maximum logical page size (in the horizontal direction) is reduced by the amount you specified here.

BIN parameters

Specifies which paper source is to be used on printers with more than one paper source. The value range is 1–255. (This subcommand should be used only for printers that have more than one paper source.)

Note: If you specify the BIN subcommand, you must also specify at least one of the legal parameters.

n An integer number between 1 and 255 that is the Media Source Id (also known as the bin number).

1 Selects the primary paper source.

2–255 Selects another paper source. If the specified bin does not exist on your printer, the default paper source for that printer will be used. For more information about paper sources on your printer, refer to your printer publications. Using a value of *100* is the same as specifying MANUAL.

MANUAL

Selects manual feed as a paper source on those printers that support manual feed. For more information, refer to your printer publications.

ENVELOPE

Selects an envelope paper source on those printers that support this function. For more information, refer to your printer documentation.

MEDIANAME

Selects a media source by specifying an agreed upon name for the bin. For a list of the valid media names, see "Appendix F. PPFA Media Names" on page 375.

qstring Up to 12 characters within single quotes specifying the media source name. On some printers, this name is pre-set into the printer; on others, it also can be entered into the printer by the user. Refer to your printer documentation for further information.

COMPID

Selects a bin based on the component id.

m For a current list of component ids, see "Appendix F. PPFA Media Names" on page 375. Component ids from 12,288 to 268,435,455 are reserved for the user.

Notes:

1. BIN selection is overridden by the printer if the form defined to each bin is the same form number. Only the primary bin is selected.
2. The primary source usually contains either letter-size (U.S.) or A4 (I.S.O.) paper. Other paper sources are used for less common paper sizes (such as legal-size) and for special paper (such as colored stock or pre-printed letterhead on heavy bond).

3. If duplexing is requested and you select from the front side from one bin and the back side from another bin, a warning message will be issued and the printer will take the paper from the bin specified on the front side.

BINERROR
Tells the printer whether or not you wish to stop printing if the wrong media is loaded on the printer.

This subcommand is displayed only on the Formdef command, not the Copygroup or the Subgroup commands since the scope of the subcommand is throughout the Formdef. Printing control is based on the status of the media loaded as it pertains to the BIN subcommand in effect at the time.

STOP If the specified input bin cannot be found, stop the print job and hold it in a state from which it can be resubmitted.

CONTINUE
If the specified input bin cannot be found, continue printing using the printer default input bin.

COLORVALUERR

When the form definition contains color values that the printer cannot render exactly as specified, you may request that the printer substitute colors and continue job processing, or you may request the printer to stop. If you request STOP, the printer issues an error and terminates. If you request CONTINUE, you may ask for an error report.

STOP Specifies that an error should be issued by the printer and the job terminated if the printer reports a color exception. A color exception is reported if the color specification in the data stream cannot be rendered as specified. Also, a color exception is reported if the host print server supports color fidelity and the target printer does not.

CONTINUE
Specifies that an exception condition should be ignored. Also, the printer substitutes colors for any that it cannot render, and the job continues.

REPORT
Specifies that the error should be reported by the printer.

NOREPORT
Specifies that the error should not be reported by the printer. NOREPORT is the default if COLORVALUERR CONTINUE is coded and neither REPORT nor NOREPORT is coded.

Note: When the printer reports a color value exception, the following actions are taken:

- If print server and the printer both support Color Fidelity and the COLORVALUERR subcommand is coded, printing occurs as previously described.
- If print server and the printer both support Color Fidelity and the COLORVALUERR subcommand is not coded, print server instructs the printer to reset to defaults at the beginning of the job.
- Whenever print server supports Color Fidelity, but the printer does not, the following rules apply:
  - If no COLORVALUERR subcommand is issued, printing continues. However, color exception errors are reported and ignored.

  - If the COLORVALUERR subcommand is issued, you could receive print server errors or the command could be ignored, depending on the level of print server you have installed and your platform (for example, OS/390, VM, AIX, and so on). Therefore, you should not use the COLORVALUERR subcommand if you do not have a host print server that supports it.
- Whenever the printer supports Color Fidelity, but print server does not, the following rules apply:
  - If no COLORVALUERR subcommand is issued, printing continues. However, color exception errors are reported and ignored.
  - If either **COLORVALUERR STOP** or **COLORVALUERR CONTINUE NOREPORT** are coded, print server issues an error and stops printing, even if there is no color exception error.
  - If **COLORVALUERR CONTINUE REPORT** is coded, print server continues printing. However, color exception errors are reported and ignored.

COMMENT *qstring*

Specifies a string comment. Use COMMENT to mark a form definition with a user comment. The string is placed in the NOP structured field of the form definition.

qstring

Specifies a quoted set of strings up to a total of 255 characters.

Note: In PPFA, a keyword or parameter (token) cannot extend across a line. Therefore, you must break the string into several strings in order to have a comment string that is longer than what will fit on one line. Each string must be a complete token with beginning and ending quotes. For example:

```
FORMDEF replace yes
COMMENT 'first line of comment'
     'second line of comment';
```

PPFA composes the comment to be:

```
first line of comment second line of comment
```

and places it in a separate NOP structured field in the form definition.

CONSTANT

Specifies whether the constant-forms function is on or off and whether constant form is to be printed on the front or back sides of a sheet.

NO Specifies that the constant forms function is off.

BACK Specifies that a constant form is to be printed on the back side without variable data.

FRONT

Specifies that a constant form is to be printed on the front side without variable data.

BOTH Specifies that a constant form is to be printed on both sides without variable data.

CUTSHEET

If you are using a cut-sheet printer, this subcommand specifies whether the medium orientation information, which is coded using the DIRECTION

and/or PRESENT subcommands, is to be passed to that printer. Not coding the CUTSHEET subcommand is equivalent to coding **CUTSHEET NO**.

NO Specifies the rotation data is not to be passed unless, of course, N_UP is coded.

YES Specifies the rotation data is to be passed.

Note: As always: If you have a continuous form printer, the medium orientation information is passed. If you have a cut-sheet printer and N_UP is coded, the orientation information is passed. The default for a COPYGROUP for which no CUTSHEET subcommand is coded is to inherit the behavior of the FORMDEF.

New: If you have a cut-sheet printer and CUTSHEET YES is coded, the orientation information is passed if you also have a level of print server that supports the CUTSHEET feature.

In all cases: Before using this command, you must have a printer that allows its media origin to be changed.

DIRECTION

Determines, along with the PRESENT subcommand, how data is oriented on printers whose media origin can be changed. See the list of printers under the PRESENT subcommand.

If you are printing line data, you usually specify the same value for the DIRECTION subcommand as is specified for the DIRECTION subcommand in the page definition.

ACROSS
Specifies that the pages are formatted in the ACROSS printing direction.

DOWN
Specifies that the pages are formatted in the DOWN printing direction.

If the DIRECTION subcommand is specified, you must specify the PRESENT subcommand. The default for DIRECTION is determined by the value specified for PRESENT.

The direction default of PORTRAIT is ACROSS; the direction default of LANDSCAPE is DOWN. If neither PRESENT nor DIRECTION is specified, the default is PRESENT PORTRAIT and DIRECTION ACROSS.

DUPLEX

Specifies whether printing is done on both sides of the sheet. This subcommand should be used only for page printers that have duplex capability.

NO Duplex printing is not performed.

NORMAL
Duplex printing is performed, with the tops of both sides printed along the same edge for side binding.

TUMBLE
Duplex printing is performed with the top of one side and the bottom of the other printed along the same edge of the sheet for top binding.

RNORMAL
Rotated normal. Duplex printing is performed with the top of one side printed along the same edge of the sheet as the bottom of the other. Used with landscape pages, N_UP 2, and N_UP 3.

RTUMBLE
Rotated tumble. Duplex printing is performed with the tops of both sides printed along the same edge. Used with landscape pages, N_UP 2, and N_UP 3.

FINISH
Specifies where the media should be stapled, folded, cut, or perforated.

This option can only be used on a document, set of documents, or an entire print file. Finishing operations are device dependent; check your printer documentation before using the FINISH parameter.

Notes:

1. The FINISH operation is for the InfoPrint 32, 40 and 60 printers.
2. The finishing operation must be specified at least once, and may occur more than once. It specifies finishing operations to be applied to the collected media.
3. If more than one finishing operation is specified, the operations are applied in the order in which they are specified.
4. FINISH positions are not affected by DIRECTION or PRESENT values.
5. Changing the orientation of the medium presentation space does not change the finishing corners or edges.
6. For continuous forms media, the carrier strips are not considered to be part of the physical media.
7. For saddle stitch operation, the staples are placed along the center of the media, parallel to the reference edge. Any offset value is ignored. If no OPCOUNT or OPPOS values are specified, the device default count is used.
8. User-specified OPCOUNT and OPPOS values are ignored for FOLD, CUT, or PERFORATE operations.

SCOPE
Determines how the finishing operation is applied.

PRINTFILE
Determines that the specified finishing operations for the OPERATION subcommand are applied to the complete print file, excluding header pages, trailer pages, and message pages.

ALL Determines that the specified finishing operations for the OPERATION subcommand are applied individually to all documents in the print file.

n Use the *n* to apply the finishing operation to a specific document. Use a value of *1* to apply the finishing operation to the first document in a print file. Use the value *2* to apply the finishing operation to the second document in a print file, and so on. The range of values includes 1-32,767.

OPERATION
Specifies the type of finishing operation.

CORNER
Specifies that one staple is driven into the media at the reference

corner (see REFERENCE parameter). For corner staples, the offset and angle of the staple from the selected corner is device dependent.

SADDLE
Specifies that one or more staples are driven into the media along the axis of the finishing operation, which is positioned at the center of the media, parallel to the reference edge (see REFERENCE parameter).

EDGE Specifies that one or more staples are driven into the media along the axis of the finishing operation.

FOLD Specifies that the media is folded along the axis of the finishing operation.

CUT Specifies that a separation cut is applied to the media along the axis of the finishing operation.

PERFORATE
Specifies that a perforation cut is applied to the media along the axis of the finishing operation.

REFERENCE
Determines the reference corner or edge of the finishing operation.

DEFAULT
Specifies that the device default determines the reference corner or edge.

TOPLEFT
Specifies that, for the finishing operation, the reference corner is positioned at the top in the left corner. This REFERENCE parameter can be used only for CORNER operations.

TOPRIGHT
Specifies that, for the finishing operation, the reference corner is positioned at the top in the right corner. This REFERENCE parameter can be used only for CORNER operations.

BOTRIGHT
Specifies that, for the finishing operation, the reference corner is positioned at the bottom in the right corner. This REFERENCE parameter can be used only for CORNER operations.

BOTLEFT
Specifies that, for the finishing operation, the reference corner is positioned at the bottom in the left corner. This REFERENCE parameter can be used only for CORNER operations.

TOP Specifies that, for the finishing operation, the reference edge is positioned at the top.[4]

BOTTOM
Specifies that, for the finishing operation, the reference edge is positioned at the bottom.[5]

4. This REFERENCE parameter can be used only for edge type operations (for example, SADDLE, EDGE, FOLD, CUT, PERFORATE).

5. This REFERENCE parameter can be used only for edge type operations.

LEFT Specifies that, for the finishing operation, the reference edge is positioned at the left.[6]

RIGHT
Specifies that, for the finishing operation, the reference edge is positioned at the right.[7]

OPCOUNT *n* | **OPPOS** *n*
Specify either the number of finishing operations requested (see OPCOUNT), or the positions for each operation on the axis of the finishing operation (see OPPOS).

OPCOUNT

Use OPCOUNT to request a specific number of finishing operations; valid values are 1-122. Do not specify OPPOS values with OPCOUNT. If OPPOS is specified for corner staple, separation cut, perforation cut or fold, this OPCOUNT value is ignored. The printer determines the positions of the operations. The default is 0 (zero).

OPPOS

Use OPPOS to control the position of each operation on the axis of the finishing operation. The subparameter is an integer value in the range of 0-32,767, representing millimeters. Do not specify the unit of measure. Do not specify OPCOUNT when you use OPPOS. If OPPOS is specified for corner staple, fold, separation cut, or perforation cut, the OPCOUNT value is ignored.

OPOFFSET *n*
Specifies that the axis for the finishing operation is offset. The subparameter is an integer value in the range of 0-32,767, representing millimeters. Do not specify OPOFFSET for corner staple or saddle stitch; the corner staple or saddle stitch values will be ignored when specified with OPOFFSET.

The following examples show how to specify finishing operations.

To request scope as the entire print job with one corner staple in the top left corner, specify:

```
FINISH SCOPE PRINTFILE OPERATION CORNER REFERENCE TOPLEFT;
```

Sometimes a user wants to request multiple finishing operations. To request that the fifth document in the job stream be finished using top left corner staple and the ninth document be edge stitched only at the print default location, specify:

```
FINISH SCOPE 5
          OPERATION CORNER
          REFERENCE TOPLEFT
       SCOPE 9
          OPERATION EDGE;
```

FONTFID

Indicates to print server whether the form definition will honor the fidelity of the specified fonts when a raster font of a specified resolution and metric-technology cannot be found on the printer. In order to get print server to honor this command you also must specify font resolution on

6. This REFERENCE parameter can be used only for edge type operations.

7. This REFERENCE parameter can be used only for edge type operations.

either the FONT command or externally (for example, on the JCL). Not coding FONTFID is equivalent to coding **FONTFID NO**.

YES Specifies that no substitution is allowed and print server issues an error message if it cannot find the font that matches the specified resolution and metric.

NO Specifies that print server will not enforce font fidelity. print server does not check for a match of the specified resolution and metric with the font found on the system.

Notes:

1. The FONTFID subcommand is designed to be used in concert with the RESOLUTION and METRICTECHNOLOGY subcommands on the FONT command, which are used to rigorously specify the font characteristics.
2. This subcommand assists the user who has created a form definition and page definition for printing with a raster font on a printer of one resolution (for example, a 240 pel printer), and has moved that application to a printer of another resolution (for example, a 300 pel printer). When print server cannot match the raster font, it substitutes an outline font, which often causes the placed text to overflow or underflow the intended space on the page. If this happens, the user can specify the actual metric and resolution of the font being used to print the text and also specify FONTFID YES, so that print server would not substitute another font.

INVOKE

Specifies where the next page of data is placed when this copy group is activated by conditional processing or by an Invoke Medium Map structured field.

INVOKE SHEET, which is the default, places the next page of data on a new sheet. The NEXT, FRONT, and BACK parameters place the next page in a subsequent partition on the same sheet or, if no partitions are available, on the next sheet. If FRONT or BACK is specified, INVOKE selects only partitions on the front or back, respectively.

print server honors the NEXT, FRONT, and BACK values of the INVOKE subcommand only if the new copy group has the same medium modifications as the previous copy group. Medium modifications include duplexing, bin, page offset, N_UP values, presentation, direction, and medium overlays.

If any of these modifications differ, print server ejects to a new sheet when the copy group is invoked. If you want to change overlays when ejecting to a new partition, use page overlays instead of medium overlays. See “Medium Overlays and Page Overlays” on page 144 for information about page and medium overlays.

When you use PLACE subcommands, the NEXT, FRONT, and BACK parameters place the next page using the next sequential PLACE subcommand that matches the requirement (next, front, or back). For example, if you print using the second PLACE subcommand of copy group A, and then you change to copy group B, you will start with the third PLACE subcommand of copy group B.

A CONSTANT parameter on the PLACE subcommand does not alter the selection process. The selection is complete, even though the selected PLACE subcommand does not place the data. N_UP performs the constant modification and continues until it finds a PLACE subcommand that does not specify CONSTANT. The data is placed with this subcommand. Observe that this PLACE subcommand need not match the FRONT or BACK specifications of the INVOKE subcommand.

SHEET
Specifies that data be placed in the first selected partition of the sheet.

NEXT Specifies that data be placed in the next selected partition.

FRONT
Specifies that data be placed in the next selected front partition.

BACK Specifies that data be placed in the next selected back partition.

JOG

Specifies whether a JOG subcommand is sent to the printer when this FORMDEF is selected by an IMM structured field, or through conditional processing. When the JOG subcommand is sent, a printer will either offset (jog) or print copymarks. For cut-sheet printers, or for continuous-forms printers with burster-trimmer-stacker enabled, the JOG subcommand causes the first sheet controlled by this FORMDEF to be stacked offset from the previous sheets. For continuous forms printers without a burster-trimmer-stacker, the JOG subcommand causes an increment in the copymark printed on the carrier strip. JOG subcommands also are sent to the printer at the beginning of each data set or at the beginning of each job, depending on host parameters. For more information about copymarks, see the system programming guide for your host print server.

YES Specifies that a JOG subcommand be sent to the printer. The first sheet printed is offset or the copymark is incremented.

NO Specifies that no JOG subcommand be sent to the printer. The first sheet printed is not offset; the copymark is not incremented.

OFFSET

Specifies the offset of the logical page for both the front and back pages in reference to the media origin. The media origin is printer dependent. For more information about media origin, see your printer publications or *Advanced Function Presentation: Printer Information*.

If you specify offset values for the back of the page, you must also specify the front offset values.

Note: The OFFSET subcommand does not affect the position of medium overlays.

x-pos Specifies the horizontal offset of the logical page on the front or back side of the copy group relative to the media origin. The valid options for *x-pos* are described in the SETUNITS command for the horizontal value.

The default unit is:
- Taken from the last SETUNITS command
- IN (inch) if no SETUNITS command has been issued
- 0.1 IN

y-pos Specifies the vertical offset for the logical page for the front or back side of the page. The valid options for *y-pos* are described in the SETUNITS command for the vertical value.

The default unit is:
- Taken from the last SETUNITS command
- IN (inch) if no SETUNITS command has been issued
- 0.1 IN

Note: The vertical offset for the 3800 must be 0.5 inch or greater.

OUTBIN *n*

Specifies the destination bin number for any pages directed by this form definition. Copygroups and subgroups in this form definition that do not specify an output bin number inherit this bin number.

n Specifies the output bin number.

PELSPERINCH *n*

Specifies the Logical Units in pels per inch for this form definition. Use the PELSPERINCH parameter to tell PPFA the pel resolution of your printer to generate more exact object placements.

n Specifies an integer number between 1 and 3,276, which determines the Logical Units in pels per inch.

Note: If the L-Units are not specified on this form definition, they are defaulted to 240 pels per inch.

```
FORMDEF xmp01 replace yes
   PELSPERINCH 300 ;

  COPYGROUP C1
     offset 2 in  3 in;

  COPYGROUP C2
     offset 2 in  3 in
     PELSPERINCH  1200;
```

Figure 94. PELSPERINCH example

In Figure 94, the form definition xmp01 has specified L-Units as 300 pels per inch. Because the COPYGROUP C1 does not specify L-Units, it inherits 300 pels per inch. COPYGROUP C2 does specify L-Units as 1200 pels per inch.

The code in COPYGROUP C1 (“offset 2 in 3 in”) produces internal and structured field values for x and y of 600 and 900, whereas in COPYGROUP C2 the same code produces values of 2400 and 3600, because of the difference in L-Units.

PRESENT

Specifies, along with the DIRECTION subcommand, how the data will be oriented on printers whose media origin can be changed.

The PRESENT and DIRECTION subcommands are only supported by cut-sheet printers when you specify the N_UP subcommand or the CUTSHEET subcommand with the YES parameter. See Figure 69 on page 128

page 128 through Figure 72 on page 131 to determine the effect of the PRESENT and DIRECTION subcommands when you use them with the N_UP subcommand.

PORTRAIT
Specifies that the pages are printed in the portrait page presentation, with their short edges at the top and bottom and their long edges at the sides.

LANDSCAPE
Specifies that the pages are printed in the landscape page presentation, with their long edges at the top and bottom and their short edges at the sides.

QUALITY *n*
Specifies the print quality. This subcommand is recognized only on printers that can produce more than one level of print quality. The default is determined by the printer model. (On some printers, the default may be set at the printer itself.) For more information, refer to your printer publications.

n You can select a level of print quality by entering any whole number from 1 to 10. Higher numbers correspond to higher levels of print quality; lower numbers correspond to lower levels. For more information, refer to your printer publications.

Print quality is determined by a numerical code in the range of 1 to 254 (hexadecimal X'01'–X'FE'). The codes corresponding to the possible QUALITY parameters are:

1 = 15 (X'0F')
2 = 40 (X'28')
3 = 65 (X'41')
4 = 90 (X'5A')
5 = 115 (X'73')
6 = 140 (X'8C')
7 = 165 (X'A5')
8 = 190 (X'BE')
9 = 215 (X'D7')
10 = 240 (X'F0')

REPLACE
Specifies whether this form definition is to replace an existing one with the same resource name in the library.

YES Replace an existing form definition of the same name in the library if there is one. If a form definition with the same name does not exist in the library, then store this form definition.

NO Do not replace an existing form definition of the same name. If a form definition with the same name does not exist in the library, then store this form definition.

This is the default.

PROCESSING

Specifies additional post-processing capabilities for selected printers and attached equipment. This option can only be used on a single page or a set of pages. The subcommand expects one to three of the following keywords:

MEDIA_INFO *n*
This parameter specifies the ID of fixed medium information that a printer or printer–attached device applies to a page. Examples such as color plates logos, letter heads, and other fixed images.

The numeric values that can be included are:

0–254 These numeric values select a particular fixed medium local ID that the printer or printer–attached device applies to a sheet. One or more IDs can be specified within this range.

255 This value selects all the current fixed medium local IDs that the printer or printer–attached devices applies to a sheet.

PERFORATE
Specifies a perforation cut at one or more fixed locations on the sheet according to the printer or printer–attached device.

CUT
Specifies a separation cut at one or more fixed locations on the sheet according to the printer or printer–attached device.

N_UP { 1 | 2 | 3 | 4 }
Specifies the number (1, 2, 3, or 4) of equal-size partitions into which the sheet will be divided. See page 127 for a list of printers that support the N_UP subcommand.

If you do not specify the N_UP subcommand in the COPYGROUP command, the N_UP subcommand from the FORMDEF command is the default for the COPYGROUP command. You can mix N_UP printing and non-N_UP printing by specifying or not specifying the N_UP subcommand in each copy group and by *not* specifying N_UP in the FORMDEF command.

OVERLAY *name*
Specifies the name of an overlay to be placed with every page in each of the N_UP partitions. The overlay is placed relative to the page origin, or if the PARTITION parameter is specified, relative to the partition origin. You can specify a maximum of 254 OVERLAY subcommands in a copy group.

x-pos y-pos
Specifies the horizontal and vertical adjustment to the position of the overlay. This is in addition to any offset values built into the overlay. The x and y values must be positive (+). You can specify them in inches (in), millimeters (mm), centimeters (cm), points, or pels. If you do not specify a unit value, PPFA uses the unit value specified in the last SETUNITS command or uses a default unit value of inches.

Note: This OVERLAY subcommand cannot be specified if the PLACE subcommand is specified. Use the OVERLAY parameter of the PLACE subcommand instead.

OVROTATE { *0* | *90* | *180* | *270* }
Specifies the rotation of the placed overlay with respect to the x-axis of the page.

Example:

Assuming the overlay has (0,0) placement coordinates, this causes page overlay "x2" to be placed 1.5 inches to the right and 2.7 inches below the beginning of the page and rotated 90 degrees clockwise with respect to the page.

```
Formdef f1
     N_UP 1   PLACE 1 FRONT
              OVERLAY x2  1.5 in  2.7 in
              OVROTATE 90;
```

PLACE

Places a page of data or a constant modification relative to a partition. Each PLACE subcommand specifies the number *n* of a partition on either the front or back side of the sheet. FRONT is the default, if you do not specify this subcommand. You must specify the same number of PLACE subcommands as the number of partitions on the sheet. The sequence of the PLACE subcommands is the sequence in which incoming pages will be placed in the partitions.

Note: The PLACE subcommand is valid only on printers that support enhanced N_UP printing. If PLACE is not specified, pages are placed in partitions in the default partition sequence.

n Specifies the numbered partition (1–4) into which the page of data will be placed.

FRONT

Specifies that this partition be placed on the front side of the sheet.

BACK Specifies that this partition be placed on the back side of the sheet.

CONSTANT

Specifies that no page data will be placed by this PLACE subcommand.

Use CONSTANT when you are placing overlays without user's data or are placing fewer data pages on the sheet than the number of partitions specified in the N_UP subcommand.

For an example of using the CONSTANT parameter with overlays and to understand how the ordering of the PLACE subcommand affects overlays, see "Enhanced N_UP Example 3: Asymmetric Pages" on page 142.

OFFSET *x-pos y-pos*

Specifies a positive offset of the page horizontally (x) and vertically (y) from the partition origin. If OFFSET is not coded, PPFA uses the value of 0.1 inch for both the x and y offsets. This OFFSET parameter overrides any other OFFSET parameters specified on the FORMDEF or COPYGROUP command. You can specify the units in inches (in), millimeters (mm), centimeters (cm), points, or pels. If you do not specify a unit value, PPFA uses the unit value specified in the last SETUNITS command or uses a default unit value of inches.

OVERLAY *name*

Specifies the name of an overlay to be placed with this PLACE subcommand. The overlay is placed relative to the page origin or,

if the PARTITION keyword is specified, to the partition origin. You can specify multiple OVERLAY parameters in each PLACE subcommand.

x-pos y-pos
Specifies the horizontal and vertical adjustment to the position of the overlay. This is in addition to any offset values built into the overlay. The x and y values must be positive (+). You can specify them in inches (in), millimeters (mm), centimeters (cm), points, or pels. If you do not specify a unit value, PPFA uses the unit value specified in the last SETUNITS command or uses a default value of inches.

PARTITION
Specifies that the previous offset is from the partition origin. If not present, the offset is from the page origin, which is subject to the OFFSET parameter.

OVROTATE { 0 | *90* | *180* | *270* }
Specifies the rotation of the placed overlay with respect to the x-axis of the page.

ROTATION {0 | 90 | 180 | 270 }
Specifies the clockwise rotation of the page and associated page overlays placed by this PLACE command.

Rotation turns the page and its associated page overlays around their fixed origin points. If you rotate the page without moving its origin point, you might rotate it off the physical medium. To prevent this, always offset the page origin to the place you want it to be for the rotated page, as shown in Figure 95.

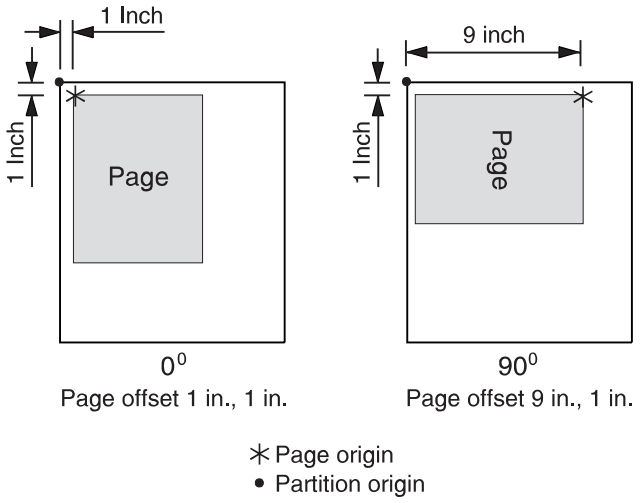


Figure 95. Offsetting the Page Origin for Rotated Pages

VIEW
Determines if this N_UP PLACE page is viewable. VIEW is relevant only when the page is being presented on a display. VIEW is ignored if the page is being printed. If VIEW is not coded, it is equivalent to specifying VIEW YES.

YES Specifies that this N_UP page is viewable and will be presented.

NO Specifies that this N_UP page is not to be presented.

VFYSETUP *verificationID* **...**

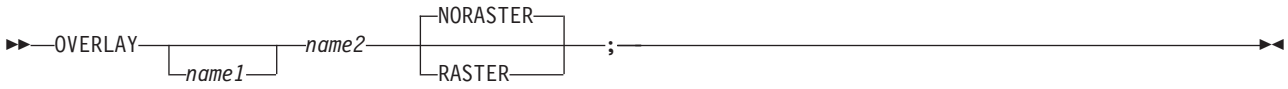
Use specifically for the IBM InfoPrint 4000 Highlight Color post processor to propagate the setup IDs to all medium maps (copygroups) in the form definition. Do not specify VFYSETUP on the COPYGROUP command. Before using the VFYSETUP subcommand, verify that your version of print server supports FORMDEF setup verification.

To use VFYSETUP, specify one or more 4-character identifier sets that match the Setup Verification IDs defined at the printer operator's console for the specific print job. For example, if the Setup Verification IDs defined at the printer were X'012F', X'0521', and X'938A', specify the following:

```
FORMDEF vfy7 REPLACE YES VFYSETUP 012F 0521 938A;
```

When print server processes the print job, it compares the setup verification IDs in the form definition to the IDS that are active in the printer. If the active IDs in the printer do not match the IDs required by the form definition, or if the printer does not support FORMDEF setup verification IDs, the job is held.

OVERLAY Command



This OVERLAY command identifies an electronic *medium overlay* to be used in one or more subgroups of a copy group, see "Medium Overlays and Page Overlays" on page 144 for additional information. When using the OVERLAY command, follow these guidelines:

- An OVERLAY command comes after the COPYGROUP command.
- A separate OVERLAY command must be specified for each electronic overlay used in a subgroup.
- A maximum of 254 OVERLAY commands can be specified for coded overlays per copy group.
- The overlay named here must be referenced in a SUBGROUP command in order to be printed (see page 192).

Notes:

1. Overlays contain their own positioning controls.
2. This does not define *page overlays*, that are placed using the N_UP subcommand. See "Medium Overlays and Page Overlays" on page 144 for additional information.

OVERLAY [*name1* **]** *name2*

Identifies an electronic overlay to be used in one or more subgroups of a copy group.

name1 Specifies an alphanumeric name of 1 to 16 characters (local name) for the overlay. It must conform to the token rules and must be unique within a copy group.

Note: If *name1* is omitted, *name2* is used as the local name and is the name used in the subgroup command.

name2 Specifies an alphanumeric name of 1 to 6 characters (user-access name) for this overlay. A prefix of *O1* is added by PPFA to identify the overlay resource.

Subcommand

RASTER or NORASTER *(3800 printers only)*

Specifies overlays as raster or not raster data.

RASTER

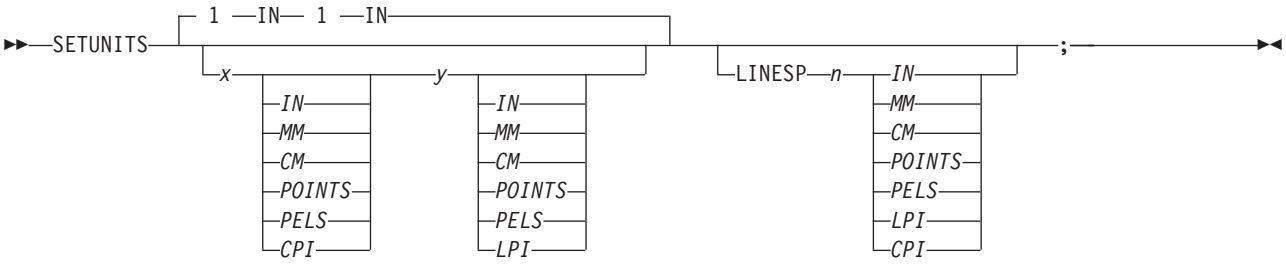
Specifies this overlay is to be kept in the printer as raster data. If this overlay is to be used several times, the printer does not need to recompile it each time.

Note: This function is ignored by print server for AIX. One raster overlay can be specified per copy group.

NORASTER

Specifies this is a coded overlay. A maximum of 254 coded overlays can be specified per copy group.

SETUNITS Command



The SETUNITS command specifies the value and the unit of measurement that will be the default for any subsequent measurement parameter in all of the commands and subcommands. These values remain the default values until another SETUNITS command is specified. The SETUNITS command should be specified as the first command in a form definition. If neither this command nor a measurement parameter is specified, the defaults identified within the following description are used.

SETUNITS

Specifies the value and the unit of measurement that will be the default for any subsequent measurement parameter in all of the commands and subcommands.

x-pos Specifies the number used for horizontal measurement. A number with up to three decimal places may be used. The default is **1**. The unit choices are IN, MM, CM, POINTS, PELS, or CPI.

Note: This value affects subsequent OFFSET subcommands.

y-pos Specifies the number used for vertical measurement. A number with up to three decimal places may be used. The default is **1**. The unit choices are IN, MM, CM, POINTS, PELS, or LPI.

Note: This value affects subsequent OFFSET subcommands.

Using CPI and LPI Units of Measurement

The CPI and LPI units of measurement make it possible to write the following command:

```
SETUNITS 10 CPI 6 LPI ;
```

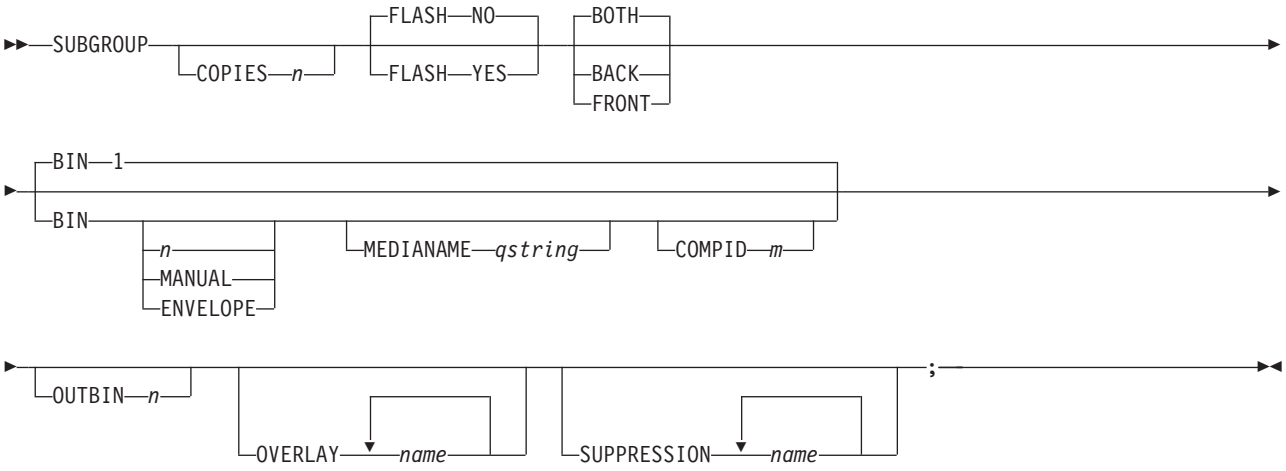
This command sets the units of measurement for horizontal and vertical spacing in terms of characters per inch and lines per inch. You can then use the OFFSET subcommand specifications to increment the spacing one character or one line at a time. The distance specified by *n* characters over and by *n* lines down is defined in the governing SETUNITS command. In this example, there are 10 characters per inch (CPI) and 6 lines per inch (LPI).

Subcommand

LINESP *n unit*

This subcommand is to be used within a page definition to set up default line spacing; it serves no purpose when used within a form definition.

SUBGROUP Command



The SUBGROUP command specifies the number of copies of a single page that are to be printed and any modifications (consisting of overlays, suppressions, type of duplexing, and forms flash) to be made to the copies. A SUBGROUP command follows a COPYGROUP command; a maximum of 127 SUBGROUP commands can be specified within each copy group.

Notes:

1. The BOTH subcommand causes two subgroups to be generated. Thus, a maximum of 63 subgroups can be specified when the BOTH subcommand is used.
2. When you specify the DUPLEX subcommand (with a parameter other than NO) in the COPYGROUP command, you must include one SUBGROUP command for each side of a sheet, or you may specify the BOTH subcommand in a single SUBGROUP command.

Subcommands

COPIES *n*

Specifies how many copies of each page are to be printed.

n Defines the number of copies (the maximum number is 255). When BACK is specified within a SUBGROUP command, the system counts the front pages printed (the actual number of sheets) not copies made (front and back). The default is **1**.

FLASH *(3800 printers only)*

Specifies whether to use forms flash.

Note: When forms flash is used, its name must be specified in the job control language for the print job. The operator must place the correct negative into the 3800 when the job is ready to print.

NO Specifies that forms flash does not occur.

YES Specifies that forms flash occurs for the number of copies designated.

{ BACK or FRONT or BOTH }

These optional subcommands specify whether the subgroup is for both sides of a sheet or for only the front or the back side.

Rules:

1. Subgroups must specify FRONT and BACK if an overlay, suppression, or forms flash appears on one side but not on the other.
2. The FRONT and BACK subgroups must have the same number of copies.

   If the number of copies differs, the COPIES parameter of the BACK subgroup is ignored, and a warning message is issued.
3. The FRONT and BACK subcommands must occur in pairs.
4. If the FRONT and BACK subcommands are specified with DUPLEX NO (in the FORMDEF or COPYGROUP commands), PPFA issues an error message and does not create the form definition.

BACK Specifies this SUBGROUP command is for the back sides of the sheets.

A subgroup with a BACK subcommand must have a FRONT subcommand in the preceding subgroup.

FRONT

Specifies this subgroup is for the front sides of the sheets.

If a DUPLEX subcommand in a FORMDEF or COPYGROUP command is specified with a parameter other than NO and the FRONT subcommand is specified in a SUBGROUP command, the next SUBGROUP command must have a BACK subcommand.

BOTH Specifies this subgroup is used for both sides of the sheet.

This is the default when DUPLEX is specified in the copy group.

If BOTH is specified with DUPLEX NO (in a FORMDEF or COPYGROUP command), PPFA issues a warning message and ignores the BOTH subcommand.

BIN parameters

Specifies the paper source. This subcommand should be used only for printers that have more than one paper source.

Note: If you specify the BIN subcommand, you must also specify at least one of the legal parameters.

n An integer number between 1 and 255 that is the Media Source Id (also known as the bin number).

1 Selects the primary paper source.

2–255 Selects another paper source. If the specified bin does not exist on your printer, the default paper source for that printer will be used. For more information about paper sources on your printer, refer to your printer publications. Using a value of *100* is the same as specifying MANUAL.

MANUAL

Selects manual feed as a paper source on those printers that support manual feed. For more information, refer to your printer publications.

ENVELOPE

Selects an envelope paper source on those printers that support this function. For more information, refer to your printer documentation.

Notes:

1. BIN selection is overridden by the printer if the form defined to each bin is the same form number. Only the primary bin is selected.
2. The primary source usually contains either letter-size (U.S.) or A4 (I.S.O.) paper. Other paper sources are used for less common paper sizes (such as legal-size) and for special paper (such as colored stock or pre-printed letterhead on heavy bond).
3. If duplexing is requested and you select from the front side from one bind and the back side from another bin, a warning message will be issued and the printer will take the paper from the bin specified on the front side.

MEDIANAME

Selects a media source by specifying an agreed upon name for the bin. For a current list of the valid media names, see "Appendix F. PPFA Media Names" on page 375.

qstring Up to 12 characters within single quotes specifying the media source name. On some printers, this name is pre-set into the printer; on others, it also can be entered into the printer by the user. Refer to your printer documentation for further information.

COMPID

Selects a bin based on the component id.

m For a current list of component ids, see "Appendix F. PPFA Media Names" on page 375. Component ids from 12,288 to 268,435,455 are reserved for the user.

OUTBIN

Specifies the destination bin number for any pages directed by this form definition. Copygroups and subgroups in this form definition that do not specify an output bin number inherit this bin number.

n Specifies the output bin number.

OVERLAY

Specifies the electronic overlay that is to be used with this subgroup.

name Specifies either the local or user-access name. A maximum of eight names can be specified within a subgroup.

Notes:

1. If the local name is used, it must be defined in an OVERLAY command before it can be referenced.
2. PPFA does not check for duplicate user-access names.

SUPPRESSION

Specifies that the named field is suppressed.

name Specifies a alphanumeric name of 1 to 8 characters (local name) of the field to be suppressed. A maximum of eight names can be specified within a subgroup.

The suppression field named here must be defined in a SUPPRESSION command following the FORMDEF command before it can be referenced. See page 193.

SUPPRESSION Command

A SUPPRESSION command, if used, must immediately follow the FORMDEF command. It names the suppression that is specified in the FIELD command of a page definition associating the form definition and the page definition.

SUPPRESSION *name*
Identifies an alphanumeric name of 1 to 8 characters (local name). The name must conform to the token rules.

You must specify the area to be suppressed in a FIELD command or a SUBGROUP command using one of the names specified within this series of SUPPRESSION commands for the suppression to be effective.

Note: A maximum of eight suppressions can be specified for one SUBGROUP command, and a maximum of 127 suppressions can be specified within one form definition.

SUPRESSION

Chapter 10. Page Definition Command Reference (Traditional)

This section is for Traditional Line Data Processing (for Record Formatting Line Data Processing, refer to "Chapter 11. Page Definition Command Reference (Record Formatting)" on page 251), and includes:

- Sequence of commands for page definitions
- Page definition commands listed alphabetically
- Detailed information on each command
- Descriptions of the applicable subcommands and parameters for each command

Sequence of Traditional Commands for Page Definitions with PRINTLINE

```
PAGEDEF [ SETUNITS ... ]
[ FONT ...]
[OBJECT ... ]
[DEFINE COLOR... ]
[ PAGEFORMAT ]
   [ TRCREF ...]
   [ SEGMENT ...]
   [ OVERLAY ...]
   PRINTLINE [ FIELD | CONDITION ...]
   [ ENDSUBPAGE ] |
   [ PRINTLINE [ FIELD | CONDITION ...] ...]
[ PAGEFORMAT ]
   [ TRCREF ...]
   [ SEGMENT ...]
   [ OVERLAY ...]
   PRINTLINE [ FIELD | CONDITION ...]
   [ ENDSUBPAGE ] |
   [ PRINTLINE [ FIELD | CONDITION ...] ...]
```

- FONT commands must be specified immediately after a PAGEDEF command. The exception is the SETUNITS command.
- OBJECT commands must be specified immediately after any FONT commands and before any PAGEFORMAT or other commands, except the SETUNITS command.
- A SETUNITS command can be placed before any other PPFA command. The values set are in effect until the next SETUNITS command.
- TRCREF, SEGMENT, and OVERLAY commands must be specified under their associated PAGEFORMAT command.
- The first PAGEFORMAT command can be omitted in a page definition, if the page definition contains only one page format. If the PAGEFORMAT command is omitted, the PAGEDEF command parameters are used to define the page format.
- At least one PRINTLINE command is required per page format for Traditional Line Data Page definition. PRINTLINE and LAYOUT commands cannot be used within the same page definition.
- An ENDSUBPAGE command can occur anywhere in a page definition that a PRINTLINE command can occur, except it can not occur between a PRINTLINE command and its associated FIELD and CONDITION commands.
- One file can contain multiple sets of page definitions.

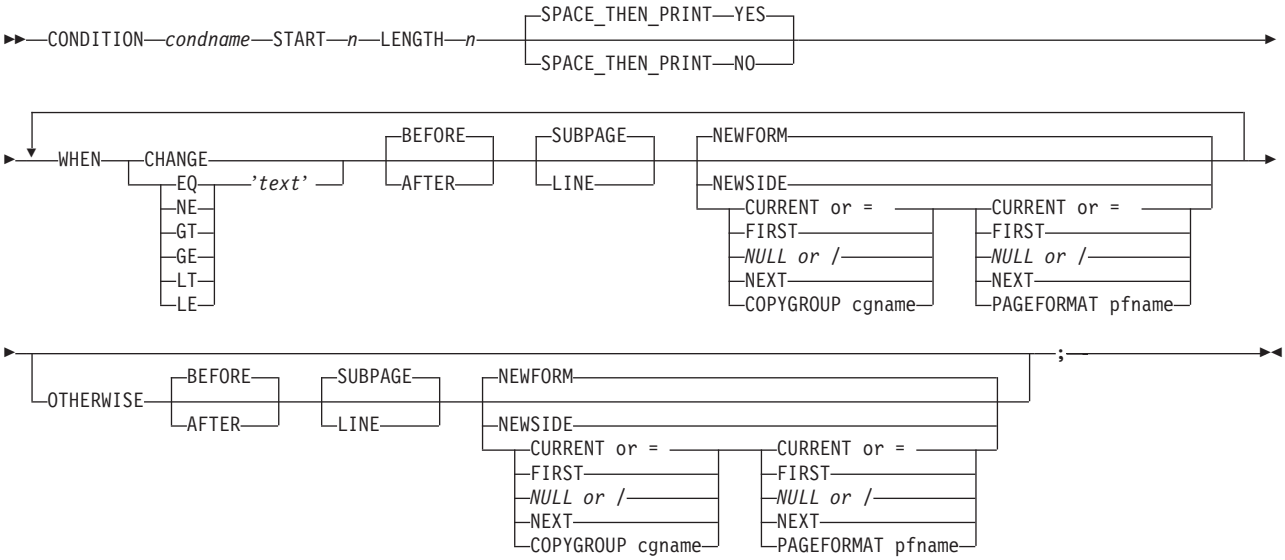
Diagram Shorthand

These terms are used in the command definitions:

x-pos A vertical position using a numeric number followed optionally by a unit. For the available units, see "Units of Measurement" on page 152.

y-pos A horizontal position using a numeric number followed optionally by a unit. For the available units, see "Units of Measurement" on page 152.

CONDITION Command (Traditional)



Short Form

CONDITION

The CONDITION command examines data in an input record and specifies actions to be taken based on the result of the examination.

- The *condname* parameter must come before any subcommands
- No WHEN subcommand can follow an OTHERWISE subcommand in the same CONDITION command

condname
Names the condition. The name must contain 1 to 8 alphanumeric characters.

PPFA allows cross-referencing to the *condname*. The cross-reference is done by using the short form of the CONDITION command (second format in the syntax table). By specifying a previously defined *condname*, PPFA uses the specifications from that command. When the condition is reused, the point where you want the comparison to begin may be at a different point in the record. By specifying the optional START subcommand, you can change the starting point of the comparison but not the field length. If the START subcommand is not specified, the starting point is the same as defined in the original CONDITION command.

Subcommands

START *n*
Specifies the starting byte of the comparison field within the data record where the comparison is to be done.

n Specifies the number of bytes from the first data byte in the record as the starting point of the comparison field. The first data byte position of an input record is 1.

Note: The carriage-control character and the table-reference character are not considered data.

LENGTH *n*

Specifies the length of the comparison field.

n Specifies the number of bytes in the data record to be compared, beginning with the position specified in START. Valid values are numbers from 1 to 8000. The length of the constant text must be the same as defined in this parameter or the results are invalid.

Comparisons are done on a byte-by-byte basis. Because the comparison field and the constant text must have the same lengths, padding is not necessary.

Note: If any part of the comparison field specified by the combination of START and LENGTH is outside the boundaries of the data record, all conditional processing is not performed. No WHEN is executed. If an OTHERWISE is present, it is not executed either.

SPACE_THEN_PRINT

Specifies whether ANSI carriage controls for spacing are enabled for the first record on the new logical page following the execution of the CONDITION command. The abbreviation of this command is SPACE.

Note: This subcommand is effective for print files that contain ANSI carriage controls. It is not used for data files containing machine carriage controls, or a mixture of ANSI and machine carriage controls.

YES Specifies that the ANSI carriage-control character in the first print record of the new page is enabled for spacing. The spacing action specified in the carriage control is performed after the eject to the new page. For example, if the carriage-control byte in the first record of the new page is a blank (skip one line before printing), then the first record skips the first line of the new page and prints at the second printline position.

NO Specifies the ANSI carriage-control character spacing action is suppressed for the first print record of the new page. If this record contains a carriage-control spacing value, such as “blank”, “0”, or “–”, the spacing is ignored and the record will print at the first printline position on the new page. Channel code values will not be ignored. If the first print record contains a valid channel code value of 1–9, or A–C, then the first record on the new page will print at the printline defined with that channel code.

WHEN *parameters*

Marks the start of the conditional comparison parameters. At least one WHEN subcommand is required.

comparisontype= { EQ | NE | GT | GE | LT | LE }

Specifies the type of comparison that is to be performed between the data in the comparison field (the portion of the record specified by START and LENGTH) and the constant text defined in the *text* parameter.

The choices are:

EQ equal to
NE not equal to
GT greater than
GE greater than or equal to
LT less than
LE less than or equal to

text

Specifies constant text for comparison with the comparison field text. The constant text length must be the same as the value on the LENGTH subcommand, with a maximum length of 8000 bytes. Examples of valid text are:

```
2C(3)'AB'
K'321,400'
X'41FE7799' 2 'CHARS'
```

Any values or parameters that are valid for the TEXT subcommand within the FIELD command may be used as text.

CHANGE

Specifies that the contents of the comparison field in this record are to be compared with the field in the record last processed by the same CONDITION command.

This parameter is an alternative to the *comparisontype* and *text* parameter combination but can be specified only once in a CONDITION command.

The results of the comparison is either TRUE or FALSE.

TRUE When the contents of the comparison field have changed from one record to the next.

FALSE

When print server processes the data, if the comparison field lies outside the boundary of the current record, which may occur with variable-length records or with truncated trailing blanks, the current record will not be used in future comparisons.

CHANGE is always false if used with the first WHEN subcommand of a series (no previous record to compare against). Whenever a new data map (one with a different name) is invoked, all the CHANGE comparisons are reset. Field values in the previous data map are not retained.

BEFORE

Specifies that the conditional action takes place before the current line or subpage is processed. This is the default.

AFTER

Specifies that the conditional action takes place after the current line or subpage is processed.

LINE

Specifies that the conditional action takes place either before or after the current line.

SUBPAGE

Specifies that the conditional action takes place either before or after the current subpage. This is the default.

For a description of subpages, see “Logical Page” on page 8.

NEWFORM
NEWFORM specifies that the only action to be taken is skipping to the front of a new form (sheet) and restarting the page format.

Note: This parameter is an alternative to using the copygroup and pageformat parameters, and is equivalent to specifying CURRENT for the copy group parameter and NULL for the pageformat parameter. CURRENT NULL are the respective defaults for copy group and pageformat parameters; therefore, NEWFORM is the default action.

NEWSIDE
Specifies that the only action to be taken is skipping to a new side (either the back of the current sheet or the front of a new sheet) and restarting the page format.

Notes:

1. This parameter is an alternative to using the copygroup and pageformat parameters, and is equivalent to specifying NULL for the copy group parameter and CURRENT for the pageformat parameter.
2. Conditional processing does not result in unnecessary blank pages.

If the line currently being processed is the first line on a side, then:

- a *copygroup* or NEWFORM action taking effect BEFORE LINE does not force an additional new form.
- a *pageformat* or NEWSIDE action taking effect BEFORE LINE does not force an additional new side.

Similarly, additional sides or forms are not forced by BEFORE SUBPAGE if the line currently being processed is in the first subpage on a side or a form.

copygroup options
Specifies a copy group to be invoked if the condition is true.

Note: Any copy group action (except NULL) restarts the page format.

{ CURRENT or = }
Invoke the current copy group again. This results in ending printing on the current sheet and resuming on the front side of a new sheet. This is the default.

The page format is restarted. This means that the first input record to go on the new page is printed using the first PRINTLINE command of the current page format, and so on. For example, data that was to be printed as subpage 4 on the sheet might be printed on subpage 1 on the new sheet.

Note: The character “=” can be used for CURRENT.

FIRST Invokes the first copy group in the current form definition.

{ NULL or / }
Retains the current copy group, taking no action. The character “/” can be used for NULL.

NEXT Invokes the next copy group in the current form definition.

Note: If NEXT is specified from the last copy group in the form definition, the first copy group in the form definition will be used.

COPYGROUP *cgname*
Uses the named copy group defined in the current form definition. The name must contain 1 to 8 alphanumeric characters.

pageformat options
Specifies a page format to be invoked if the condition is true.

{ CURRENT or = }
Invokes the current page format again. This results in ending printing on the current sheet and resuming on the front side of a new sheet.

The page format is restarted. This means that the first input record to go on the new page is printed using the first PRINTLINE command of the current page format, and so on.

The character "=" can be used for CURRENT.

FIRST Invokes the first page format in the current page definition.

{ NULL or / }
Retains the current page format, taking no action. The character "/" can be used for NULL. This is the default.

NEXT Invokes the next page format in the current page definition.

Note: If NEXT is specified from the last page format in the page definition, the first page format in the page definition will be used.

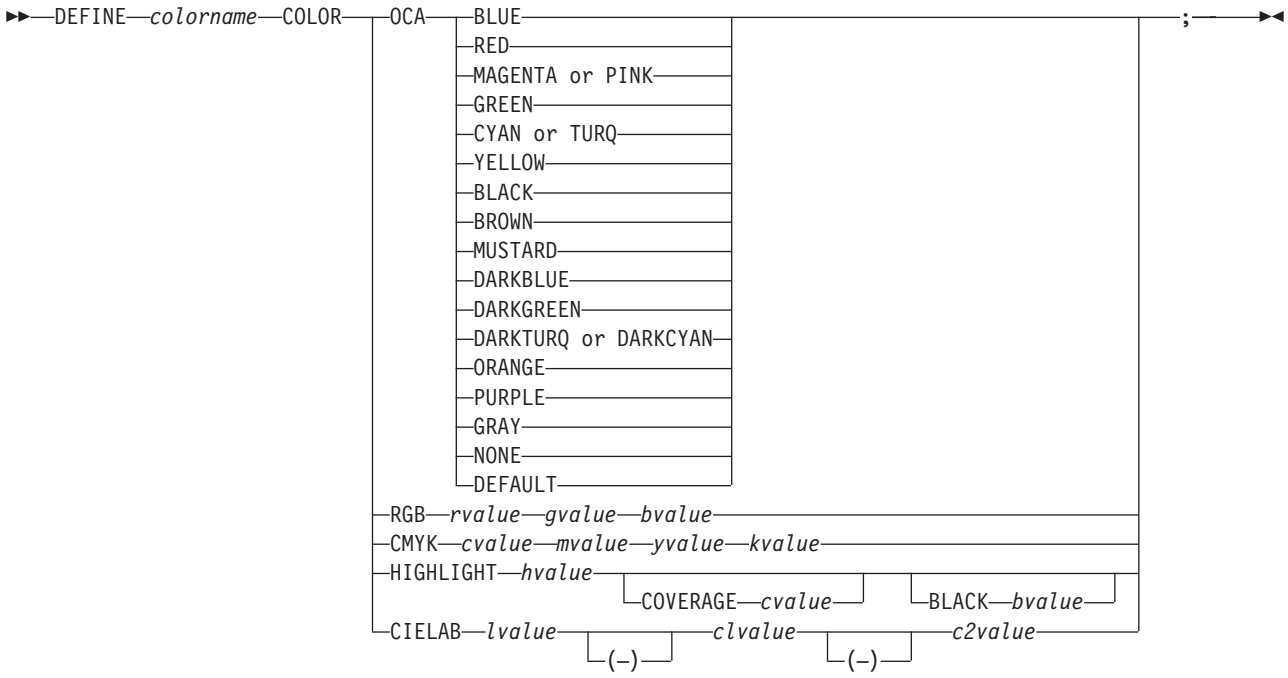
PAGEFORMAT *pfname*
Uses the named page format defined in the current page definition. The name must contain 1 to 8 alphanumeric characters.

OTHERWISE *parameters*
Marks the start of a conditional action to be taken if all preceding WHEN comparisons have proved false. The syntax is the same as the WHEN subcommand, except that the comparison parameters (*comparisontype text* or 'CHANGE') are not used. See the WHEN parameters starting with BEFORE on page 200 for a description of the parameters.

If the OTHERWISE subcommand is not used within the sequence, no action is taken. This is the same as if an OTHERWISE NULL NULL had been entered.

Note: OTHERWISE is not executed if any part of the comparison field specified by the combination of START and LENGTH is outside the boundaries of the data record.

DEFINE COLOR Command (Traditional)



colorname
This command is used to predefine a color model to be used as a *colorname* in a later command, such as the PRINTLINE or FIELD commands.

Subcommands

DEFINE COLOR
Defines a color name of a particular color model such as OCA, RGB, CMYK, HIGHLIGHT, or CIELAB. This name can be used anywhere color of that model is allowed. For example a defined color of any color model can be used as text color in the FIELD or PRINTLINE commands, but only a color defined as an OCA color can be used as an object placement area color. See the OBCOLOR subcommand in "PRINTLINE Command (Traditional)" on page 231.

colorname

Select a 1 to 10 character name. Use this name on the command to identify this color. For example:

```
DEFINE oldblue COLOR OCA brown;
PRINTLINE   COLOR oldblue;
```

Color Model
Specifies the color of print for this field supported in MO:DCA for the OCA, the Red/Green/Blue color model (RGB), the highlight color space, the Cyan/Magenta/Yellow/Black color model (CMYK), and the CIELAB color model.

OCA Chose one of the standard OCA colors from the previous syntax diagram.

Note: In some printer publications, the color turquoise (TURQ) is called “cyan”, and the color pink (PINK) is called “magenta”.

PPFA supports the following synonyms:

- CYAN for TURQ
- DARKCYAN for DARKTURQ
- DBLUE for DARKBLUE
- DCYAN for DARKTURQ
- DGREEN for DARKGREEN
- DTURQ for DARKTURQ
- MAGENTA for PINK

RGB *rvalue gvalue bvalue*

Three **RGB** integer values are used. The first (rvalue) represents a value for red, the second (gvalue) represents a value for green, and the third (bvalue) represents a value for blue. Each of the three integer values may be specified as a percentage from 0 to 100.

Note: An RGB specification of 0/0/0 is black. An RGB specification of 100/100/100 is white. Any other value is a color somewhere between black and white, depending on the output device.

HIGHLIGHT *hvalue* **COVERAGE** *cvalue* **BLACK** *bvalue*

Indicates the highlight color model. Highlight colors are device dependent.

You can use an integer within the range of 0 to 65,535 for the *hvalue*.

Note: An *hvalue* of 0 indicates that there is no default value defined; therefore, the default color of the presentation device is used.

COVERAGE indicates the amount of coverage of the highlight color to be used. You can use an integer within the range of 0 to 100 for the *cvalue*. If less than 100 percent is specified, the remaining coverage is achieved with the color of the medium.

Note: Fractional values are ignored. If **COVERAGE** is not specified, a value of 100 is used as a default.

BLACK indicates the percentage of black to be added to the highlight color. You can use an integer within the range of 0 to 100 for the *bvalue*. The amount of black shading applied depends on the **COVERAGE** percentage, which is applied first. If less than 100 percent is specified, the remaining coverage is achieved with black.

Note: If **BLACK** is not specified, a value of 0 is used as a default.

See “Color on the IBM InfoPrint HiLite Color Post Processor” on page 46 for more information.

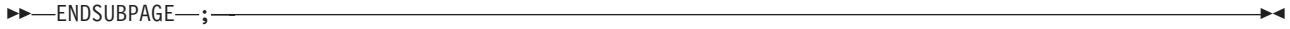
CMYK *cvalue mvalue yvalue kvalue*
Defines the cyan / magenta / yellow / black color model. *Cvalue* specifies the cyan value. *Mvalue* specifies the magenta value. *Yvalue* specifies the yellow value. *Kvalue* specifies the black value. You can use an integer percentage within the range of 0 to 100 for any of the CMYK values.

CIELAB *Lvalue* **(−)***c1value* **(−)***c2value*
Defines the CIELAB model. Use a range of 0.00 to 100.00 with *Lvalue* to specify the luminance value. Use signed integers from −127 to 127 with *c1value* and *c2value* to specify the chrominance differences.

Lvalue, c1value, c2value must be specified in this order. There are no defaults for the subvalues.

Note: Do not specify both an OCA color with the **COLOR** subparameter and an extended color model on the same **FIELD** or **PRINTLINE** command. The output is device dependent and may not be what you expect.

ENDSUBPAGE Command (Traditional)

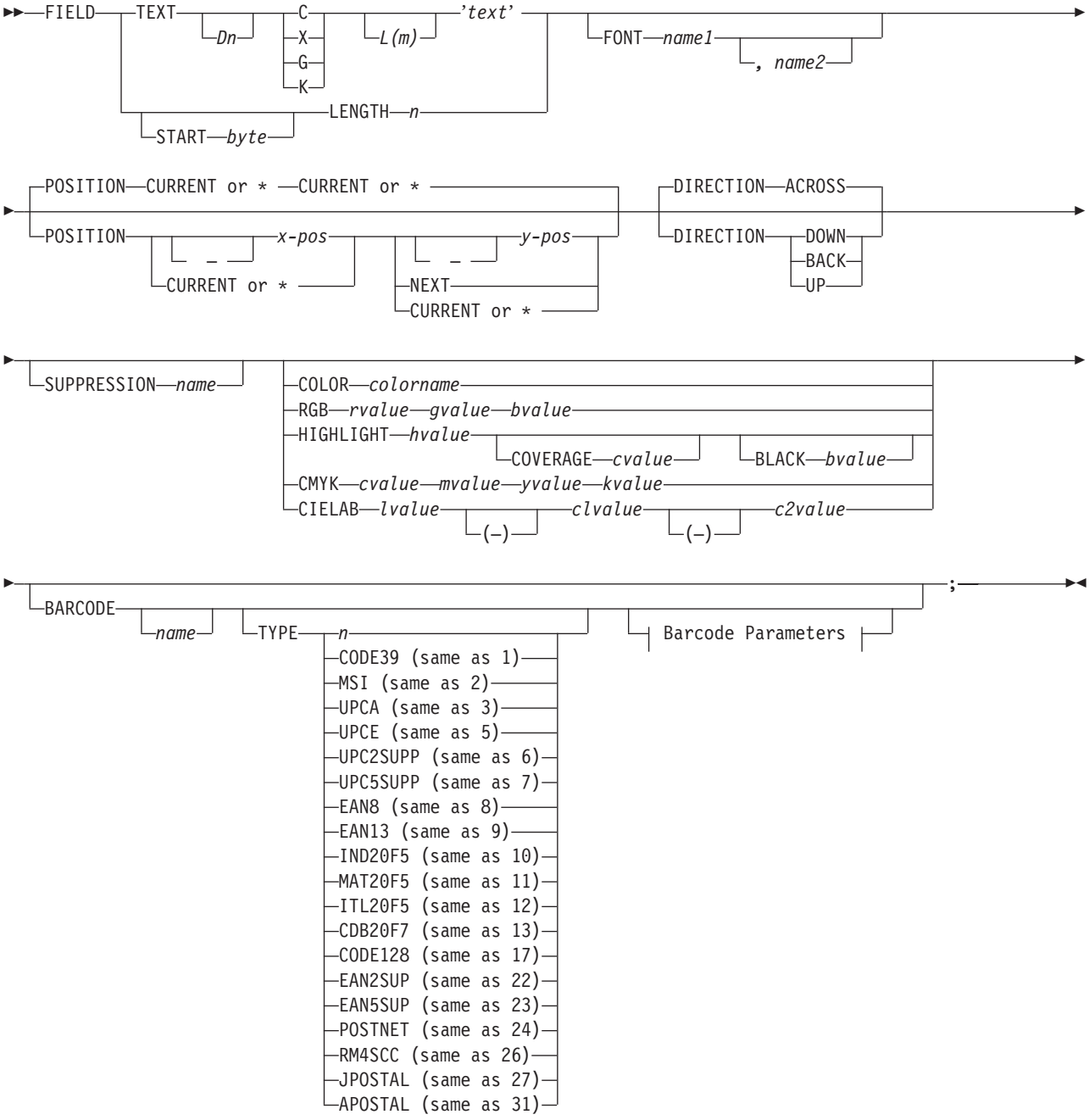


The ENDSUBPAGE command is used to identify the end of a subpage for conditional processing.

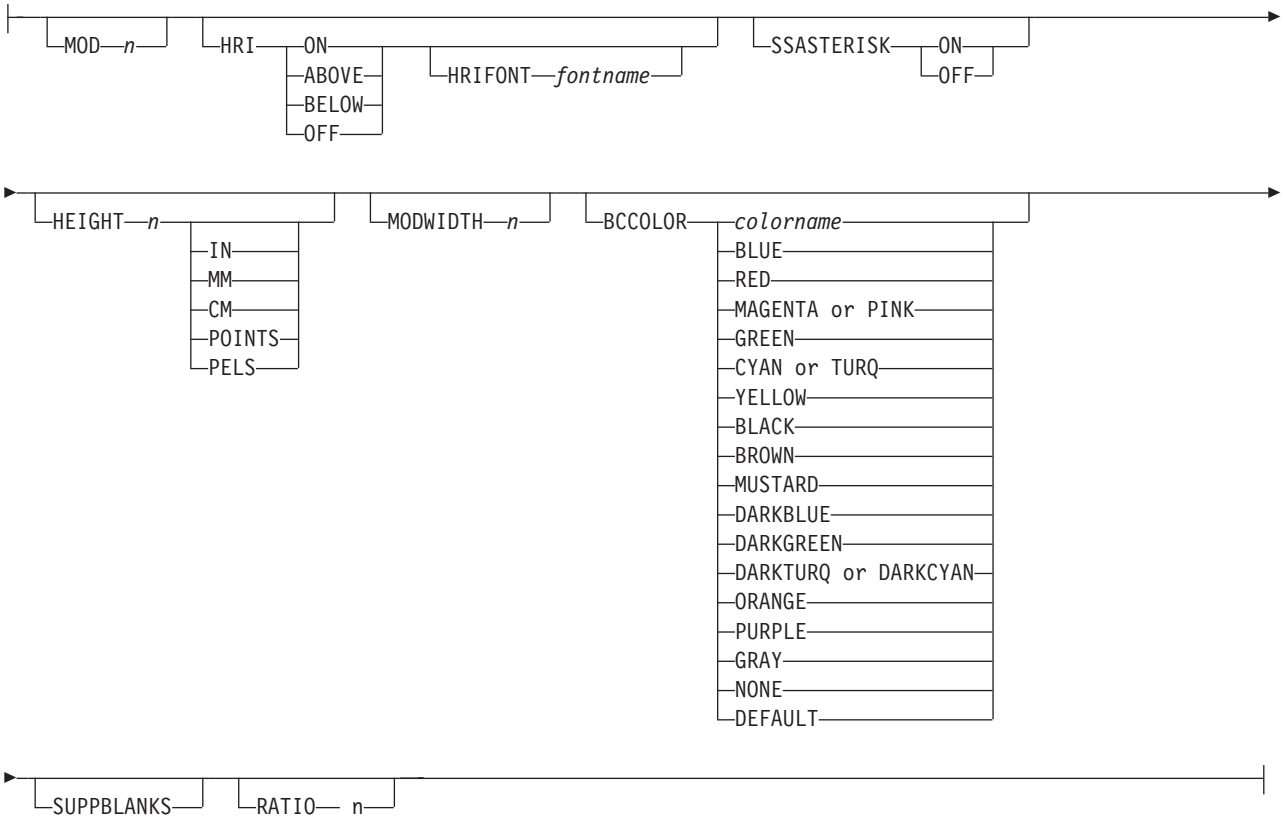
You can specify the ENDSUBPAGE command at any point in a page definition command stream where a PRINTLINE or LAYOUT command can occur. However, you must not enter the ENDSUBPAGE command between a PRINTLINE or LAYOUT command and its associated FIELD or CONDITION command.

If an ENDSUBPAGE command is not specified, the entire page format is treated as one subpage.

FIELD Command (Traditional)



Barcode Parameters:



The FIELD command identifies a field in a data record or supplies a field of constant text, and positions where the field is on the page. More than one position on the page can be specified.

FIELD commands:
- Are subordinate to a PRINTLINE command
- Must follow a PRINTLINE command
- Must contain either a LENGTH subcommand or a TEXT subcommand

The FONT, DIRECTION, and COLOR subcommands do not have fixed defaults. If any of these subcommands is omitted, the value for the omitted subcommand is obtained from corresponding subcommand in the PRINTLINE command.

Subcommands

START

Specifies the starting byte in the data record for the desired field.

n Specifies the number of bytes from the first data byte in the record to be used as the starting point of the field. The first data byte position of an input record is 1.

Note: The carriage-control character and the table-reference character are not considered data.

***** Denotes the next byte after the field identified in the previous FIELD command, excluding FIELD commands with constant TEXT.

If START * was specified in the previous FIELD command, byte 1 is assumed.

+ *n* Adds the value of n to the * byte position.

− *n* Subtracts the value of n from the * byte position.

If START is omitted and LENGTH is specified, then START * is assumed.

LENGTH
Specifies the number (*n*) of bytes to process from the data record, beginning with the position specified in START.

TEXT
Specifies the constant text that is to be printed in the output. A maximum of 65,535 bytes of text can be provided in one page format.

Note: This text is considered constant in that the same text is printed each time. In reference to the CONSTANT command within a form definition, this text is considered variable because the text prints only where variable data is allowed to print.

duplication=*Dn*
Specifies the number of times the text is to be repeated (use a decimal number). The maximum times the text is repeated varies depending on the size of the text. The default is **1**.

texttype = {C | X | G | K }
Specifies the type of text.

C Indicates that the text contains single-byte code characters, which includes all Roman alphabetic characters (for example, those used for English). Any valid character code can be specified, including blanks. This is the default.

X Indicates that the text contains hexadecimal codes (in groups of two hexadecimal codes) that specify values from X'00' through X'FE'.

G Indicates that the text contains double-byte code characters (for example, kanji characters).

Characters in type G text must start with shift-out (SO X'0E') and end with shift-in (SI X'0F') characters within opening and closing apostrophes (X'7D').

K Indicates that the text contains kanji numbers enclosed in apostrophes. Kanji numbers are separated by commas:

```
K'321,400'
```

Valid double-byte character set (DBCS) codes are from X'41' through X'FE' for each byte. Code X'4040' (blank) is the only exception.

Valid: X'4040', X'4041', X'41FE' X'FE41', X'FEFE'

Invalid: X'2040', X'413E', X'4100' X'7F00', X'FE3E'

L(*m*) Specifies the length of text (use a decimal number in parentheses). When the actual length of the text is different from *m*, the *m* specification is honored. That is, the text is either padded with blanks to the right or truncated.

'*text*' Specifies the text.

Examples:

- When TEXT 2C(3)'AB' is specified, 'AB AB ' is generated. The blanks are generated because of the (3) specification.
- TEXT 2C(1)'AB' generates 'AA', truncating the Bs.

FONT

Defines the font to be used for the field.

name1 Specifies the name of a font used to print the data. This font must have been defined in a previous FONT command in this page definition.

If Shift-Out, Shift-In (SOSI) processing is used, *name1* must be the single-byte font.

name2 Specify only when using Shift-Out, Shift-In (SOSI) processing to dynamically switch between a single-byte font and a double-byte font within the field. *name2* must be the double-byte font.

Notes:

1. If this subcommand is not specified, the font specified in the preceding PRINTLINE command is used. If neither has been specified, print server assigns a font.
2. When selecting a font in AIX, you could consider that the text will be selected in EBCDIC, not ASCII. Therefore, an EBCDIC font and code page 500 (also called International #5) should be used for *name1*.

POSITION

Specifies the starting position of the field in the printout.

x-pos

Do not mix *x-pos* specifications with CURRENT or * except in ACROSS fields.

− Specifies that the *x* value is negative.

x Specifies the horizontal offset for the starting print position relative to the *printline starting position*. The choices are IN, MM, CM, POINTS, or PELS.

The default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

The PELS measurement equals one L-unit or 1/240 of an inch, depending on whether the PELSPERINCH parameter had been specified previously.

CURRENT

Specifies that the inline offset (relative to the field's direction) is the end of the previous field. For the first field, use the PRINTLINE offset. This is the default.

Note: The meaning of CURRENT differs from the meaning of the PRINTLINE command parameter SAME.

***** Alternate for CURRENT.

y-pos

Do not mix *y-pos* specifications with CURRENT or * except in ACROSS fields.

− Specifies that the *y* value is negative.

y Specifies the vertical offset for the starting print position relative to the *printline starting position*. The choices are IN, MM, CM, POINTS, or PELS.

The default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

NEXT Specifies a field that is positioned down one line in the baseline direction (as defined in the SETUNITS command LINESP subcommand) from the previous field.

Use NEXT only in ACROSS fields.

CURRENT

Specifies that the baseline offset (relative to the field's direction) is the same as the previous field. That is, the baseline position does not change. For the first field, use the PRINTLINE offset. This is the default.

***** Alternate for CURRENT.

DIRECTION

Specifies the print direction of the field, relative to the upper-left corner as you view the logical page. If this subcommand is omitted, the direction specified in the governing PRINTLINE command is used.

Note: Not all printers can print in all directions. Refer to your printer documentation for more information.

ACROSS

The page is printed with the characters added from *left to right* on the page, and the lines are added from the top to the bottom.

DOWN

The page is printed with the characters added from *top to bottom* on the page, and the lines added are from the right to the left.

BACK The page is printed with the characters added from *right to left* on the page, and the lines are added from the bottom to the top.

UP The page is printed with the characters added from *bottom to top* on the page, and the lines are added from the left to the right.

SUPPRESSION

Specifies that this field can be suppressed.

name Specifies the name of a field to be suppressed.

Printing of this field is suppressed if this name is identified by a SUPPRESSION command within the form definition.

The same name can be used in one or more fields to suppress these fields as a group.

COLOR

Specifies an OCA or defined color for the text of this field. This subcommand

is recognized only by printers that support multiple-color printing. Refer to your printer publication for more information.

colorname

Values for *colorname* are NONE, DEFAULT, BLACK, BLUE, BROWN, GREEN, PINK, RED, TURQ (turquoise), YELLOW, ORANGE, PURPLE, MUSTARD, GRAY, DARKBLUE, DARKGREEN, DARKTURQ (dark turquoise) or a predefined color. The color choices depend on the printer. NONE is the color of the medium. DEFAULT is the printer default color.

Note: In some printer publications, the color turquoise (TURQ) is called “cyan”, and the color pink (PINK) is called “magenta.”

PPFA supports the following synonyms:

- CYAN for TURQ
- DARKCYAN for DARKTURQ
- DBLUE for DARKBLUE
- DCYAN for DARKTURQ
- DGREEN for DARKGREEN
- DTURQ for DARKTURQ
- MAGENTA for PINK

Color Model

Specifies the color of print for this field supported in MO:DCA for the Red/Green/Blue color model (RGB), the highlight color space, the Cyan/Magenta/Yellow/Black color model (CMYK), and the CIELAB color model.

RGB *rvalue gvalue bvalue*

Three **RGB** integer values are used. The first (rvalue) represents a value for red, the second (gvalue) represents a value for green, and the third (bvalue) represents a value for blue. Each of the three integer values may be specified as a percentage from 0 to 100.

Note: An RGB specification of 0/0/0 is black. An RGB specification of 100/100/100 is white. Any other value is a color somewhere between black and white, depending on the output device.

HIGHLIGHT *hvalue* **COVERAGE** *cvalue* **BLACK** *bvalue*

Indicates the highlight color model. Highlight colors are device dependent.

You can use an integer within the range of 0 to 65535 for the *hvalue*.

Note: An *hvalue* of 0 indicates that there is no default value defined; therefore, the default color of the presentation device is used.

COVERAGE indicates the amount of coverage of the highlight color to be used. You can use an integer within the range of 0 to 100 for the *cvalue*. If less than 100 percent is specified, the remaining coverage is achieved with the color of the medium.

Note: Fractional values are ignored. If **COVERAGE** is not specified, a value of 100 is used as a default.

BLACK indicates the percentage of black to be added to the highlight color. You can use an integer within the range of 0 to 100 for the *bvalue*. The amount of black shading applied depends on the **COVERAGE** percentage, which is applied first. If less than 100 percent is specified, the remaining coverage is achieved with black.

Note: If **BLACK** is not specified, a value of 0 is used as a default.

See "Color on the IBM InfoPrint HiLite Color Post Processor" on page 46 for more information.

CMYK *cvalue mvalue yvalue kvalue*
Defines the cyan / magenta / yellow / black color model. *Cvalue* specifies the cyan value. *Mvalue* specifies the magenta value. *Yvalue* specifies the yellow value. *Kvalue* specifies the black value. You can use an integer percentage within the range of 0 to 100 for any of the CMYK values.

CIELAB *Lvalue* **(−)***c1value* **(−)***c2value*
Defines the CIELAB model. Use a range of 0.00 to 100.00 with *Lvalue* to specify the luminance value. Use signed integers from −127 to 127 with *c1value* and *c2value* to specify the chrominance differences.

Lvalue, *c1value*, *c2value* must be specified in this order. There are no defaults for the subvalues.

Note: Do not specify both an OCA color with the **COLOR** subparameter and an extended color model on the same **FIELD** or **PRINTLINE** command. The output is device dependent and may not be what you expect.

```
FIELD START 1 LENGTH 5
                COLOR BLUE ;
 FIELD START 1  LENGTH 1
                RGB 10  75  30 ;
 FIELD START 1  LENGTH 1
                cmyk 80 10 10 10 ;
 FIELD START 1  LENGTH 2
                CIELAB 80 100 20 ;
 FIELD START 1  LENGTH 2
                highlight 5 ;
 FIELD START 1  LENGTH 2
                highlight 300 COVERAGE 50 BLACK 30 ;
```

Figure 96. Color Model Using the FIELD Command

BARCODE parameters
Specifies a bar code in a page definition.

The bar code name can be 1-8 characters long. Refer to your printer documentation for additional information about bar code support. Ensure that the bar code fits on the page or you will get errors at print time.

Please read your printer hardware documentation before using bar codes. The documentation will indicate which bar code types, modifiers, modwidth, element heights, and ratio values are valid for the printer.

PPFA does minimal verification of the bar code values. If you use the MOD, HEIGHT, MODWIDTH and RATIO parameters, ensure that the values you specify are valid for your printer.

For printer optimization, specify **BARCODE** *name options* in the first instance of a specific type of bar code. If this type is used again, position it as usual with **START**, **LENGTH**, and **POSITION**, but specify the barcode information using only **BARCODE** *same-name-as-previously*. The BARCODE subcommand is recognized only by printers that support BCOCA bar code printing; refer to *Advanced Function Presentation: Printer Information* (G544-3290) for more information.

Note: If you want to suppress blanks, use the SUPPBLANKS parameter.

For more information about bar codes, see "Appendix D. More About Bar Code Parameters" on page 349 and refer to *Data Stream and Object Architectures: Bar Code Object Content Architecture Reference*, S544-3766.

TYPE { *n* | *type-name* }

TYPE Specifies the type of bar code symbol to be generated.

Note: If a type indicates "(same as *n*)", you may substitute the number given for the character name.

The following bar code types are supported:

type-name
Specifies a specific bar code name to be included in a page definition.

CODE39 (same as 1)
Specifies a bar code type of Code 39 (3-of-9 code), Automatic Identification Manufacturers Uniform Symbol Specification 39.

MSI (same as 2)
Specifies a bar code type of modified Plessey code.

UPCA (same as 3)
Specifies a bar code type of Universal Product Code (United States) and the Canadian Grocery Product Code, Version A

UPCE (same as 5)
Specifies a bar code type of Universal Product Code (United States) and the Canadian Grocery Product Code, Version E

UPC2SUPP (same as 6)
Specifies a bar code type of Universal Product Code (United States) two-digit Supplemental (periodicals).

UPC5SUPP (same as 7)
Specifies a bar code type of Universal Product Code (United States) five-digit Supplemental (paperbacks).

EAN8 (same as 8)
Specifies a bar code type of European Article Numbering 8 (includes Japanese Article Numbering-short).

EAN13 (same as 9)
Specifies a bar code type of European Article Numbering 13 (includes Japanese Article Numbering-standard).

IND2OF5 (same as 10)
Specifies a bar code type of Industrial 2-of-5.

MAT2OF5 (same as 11)
Specifies a bar code type of Matrix 2-of-5.

ITL2OF5 (same as 12)
Specifies a bar code type of Interleaved 2-of-5, Automatic Identification Manufacturers Uniform Symbol Specification-I 2/5.

CDB2OF7 (same as 13)
Specifies a bar code type of Codabar, 2-of-7, Automatic Identification Manufacturers Uniform Symbol Specification-Codabar.

CODE128 (same as 17)
Specifies a bar code type of Code 128, Automatic Identification Manufacturers Uniform Symbol Specification-128.

Note: There is a subset of CODE128 called EAN128. These EAN128 bar codes can be produced with PPFA by specifying CODE128 for the bar code type in the pagedef and including the ″extra″ parts of the bar code in the data. The UCC-128 bar code format is:

```
startcode FNC1 ai nnnnnnnnnnnnnnnnnn m c stopchar
```

The string of n's represents the bar code data. The start code, stop character, and 'c' value are generated by the printer microcode for BCOCA bar codes. The FNC1 is a hexidecimal 8F character. The ″ai″ is an application identifier and needs to be defined for use by each EAN128 application. The ″m″ is a modulo 10 check digit that must be calculated by the application and included in the bar code data.

Not all IBM printers will generate the EAN128 bar codes, thus you may need to verify that the bar code produced in this manner is readable by your bar code scanner.

For more information about the EAN128 bar codes, visit the Uniform Code Council WEB site at http://www.UC-council.org.

EAN2SUP (same as 22)
Specifies a bar code type of European Article Numbering, Two-digit Supplemental.

EAN5SUB (same as 23)
Specifies a bar code type of European Article Numbering, Five-digit Supplemental.

POSTNET (same as 24)
Specifies a bar code type of POSTal Numeric Encoding Technique (United States Postal Service), and defines specific values for the BSD module width, element height, height multiplier, and wide-to-narrow ratio fields.

RM4SCC (same as 26)
Specifies a 4-state customer code defined by the Royal Mail Postal Service of England for bar coding postal code information.

JPOSTAL (same as 27)
A complete Japan Postal Bar Code symbol consists of a set of distinct bars and spaces for each character followed by a modulo 19 checksum character and enclosed by a unique start character, stop character and quiet zones.

APOSTAL (same as 31)
Specifies the barcode type as defined by the Australian Postal Service.

MOD Specifies additional processing information about the bar code symbol to be generated (for example, MOD specifies whether a check-digit [8] should be generated for the bar code symbol).

n The meaning of *n* differs between the types. For more information, see Table 18 on page 358.

If **MOD** is not specified, the MOD value defaults as follows, depending on the bar code type specified:

TYPE	MOD	TYPE	MOD
1	1	11	1
2	1	12	1
3	0	13	1
5	0	17	2
6	0	22	0
7	0	23	0
8	0	24	0
9	0	26	0
10	1	27	0
		31	1

HRI Specifies the human-readable interpretation (text characters) to be generated and placed above or below the bar code symbol, as directed.

ON Specifies that HRI should be generated at the default location for the barcode type.

ABOVE
Specifies that HRI should be placed above the bar code symbol.

8. Check digits are a method of verifying data integrity during the bar code reading process.

BELOW
Specifies that HRI should be placed below the bar code symbol.

OFF Specifies that HRI should not be generated.

Note: If **HRI** is requested, and HRI font isn't, the printer default font is used to render the HRI, instead of the font specified on the FIELD FONT subcommand.

HRIFONT *fontname*
Specifies the font to be used in printing the HRI for the barcode.

SSASTERISK
Specifies whether an asterisk is to be generated as the HRI for **CODE39** bar code start and stop characters.

Note: SSASTERISK is ignored by all bar code types except **CODE39**.

ON Specifies that start and stop characters should be generated in the HRI.

OFF Specifies that start and stop characters should not be generated in the HRI.

HEIGHT
Specifies the height of bar code element. For UPC and EAN bar codes, the total height includes the bar code and the HRI characters.

If **HEIGHT** is not specified, the printer default height is used.

Note: HEIGHT is ignored by bar code types that explicitly specify the element heights (for example, **POSTNET** or **RM4SCC**).

n Specifies the height of the bar code.

unit Specifies a unit of measurement for the *HEIGHT* parameter. The choices are IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

MODWIDTH
Specifies the width of the smallest defined bar code element, using mils (thousandths of an inch). For bar code types that explicitly specify the module width (for example, **POSTNET** and **RM4SCC**), this field is ignored. The range of values allowed is 1-254. If **MODWIDTH** is not specified, the printer default MODWIDTH is used.

n Specifies the width of each module, using thousandths of an inch (1/1000) as the unit of measurement.

BCOLOR
Specifies an OCA or defined OCA color to be used in printing the barcode and its HRI.

cname
Specifies the name of a defined color.

SUPPBLANKS
Suppress the trailing blanks in the data field used to generate the barcode.

When the page definition selects any of the EAN, UPC or Postnet bar code types and modifiers and have also requested that trailing blanks be truncated for the bar code field, print server will examine the resulting data length and choose the correct bar code type and modifier for the bar code object created.

Note: If the data length does not match any of the bar code type and modifier combinations, print server will use the original bar code type and modifier requested to build the bar code object.

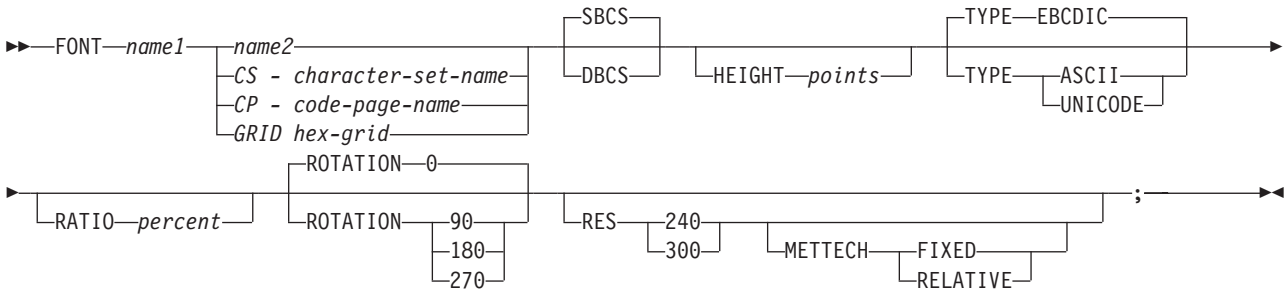
RATIO

Specifies the ratio between the width of the wide and the narrow bar code elements. The range of values allowed is 100-500, but you must specify a value appropriate for your printer and bar code type or you will get errors at print time.

If **RATIO** is not specified, the printer default ratio is used.

n The RATIO is specified as a percent value. For example, form *nnn*. For example, 200 represents a ratio of 2 to 1; 250 represents a ratio of 2.5 to 1. For most bar code symbols, the **RATIO** value should be between 200 and 300. For bar code types that explicitly specify the module width (for example, **POSTNET** and **RM4SCC**, this field is ignored. If **RATIO** is not specified, the default ratio for the bar code symbol is used.

FONT Command (Traditional)



The FONT command is used to identify the fonts that are to be specified in the PRINTLINE, FIELD, and TRCREF commands. A maximum of 127 font names for each page definition can be identified.

Note: Naming a font with the FONT command does not, by itself, affect your output. You must specify the font in a PRINTLINE, FIELD, or TRCREF command for the font to become effective. If you do not name a font, the default font will be used.

FONT commands immediately follow the PAGEDEF command. A separate FONT command is required:

For each font used within a page definition
For each rotation of the same font

Note: See the TRCREF command for the exception.

FONT *name1* **{** *name2* **|** **CS** *character-set-name* **CP** *code-page-name* **|** **GRID** *hex-grid* **}**

Identifies the fonts to be specified in the PRINTLINE, FIELD, and TRCREF commands.

name1 Specifies an alphanumeric name of 1 to 16 characters (local name) of the font to be used in this page definition. The name must conform to the token rules and must be unique within this page definition.

name1 is used in the PRINTLINE, FIELD, or TRCREF commands of a page definition.

name1 is optional if *name2* is specified.

name2 Specifies an alphanumeric name of 1 to 6 characters (user-access name) of the coded font to be used in this page definition. Specify this name without the Xn prefix.

character-set-name
Specifies an alphanumeric name of 1 to 6 characters of the character set to be used in this page definition. Specify this name without the Cn prefix.

code-page-name
Specifies an alphanumeric name of 1 to 6 characters of the code page without the T1 prefix to be used in this page definition.

hex-grid
Specifies the 16-character hexadecimal GRID.

Subcommands

SBCS or DBCS
Specifies single-byte or double-byte fonts.

SBCS Specifies that the font is a single-byte character set. This is the default.

DBCS Specifies that the font is a double-byte character set.

HEIGHT *points*
Specifies the height of the outline font.

points Each point is equal to 1/72 of one inch.

TYPE The TYPE subcommand indicates the type of Font being used.

EBCDIC
This parameter is normally used for fonts on OS390–based systems. This is the default.

ASCII This parameter is normally used for fonts on workstation–based systems.

UNICODE
This parameter is used with Unicode type fonts.

RATIO
Specifies the ratio of scaling the width relative to the height in an outline font.

percent Represents the percent of the “normal” width of the character that will be printed. For example, specifying RATIO 50 yields a font with characters half as wide as normal, and specifying RATIO 200 yields a font with characters twice as wide (200% as wide) as normal. If RATIO is specified, you must also specify the HEIGHT.

ROTATION
Specifies the rotation of characters in degrees. The specified value is relative to the inline direction of a printline or field. Valid rotations are 0°, 90°, 180°, or 270°; **0°** is the default.

RESOLUTION
Specifies the resolution and metric technology on a font. Examples of resolution command inputs are:

RES or **RESOLUTION**

```
The raster-pattern resolution units in pels per inch
     240            240 pels per inch
     300            300 pels per inch
```

METTECH or **METRICTECHNOLOGY**

```
The metric technology used for this raster font
     FIXED          Fixed-metric technology
     RELATIVE       Relative-metric technology
```

Notes:

1. The resolution and metrictechnology subcommands allow rigorous font specifications for use with font fidelity. See the font fidelity subcommand FONTFID on the FORMDEF command.

2. For a description of metric technologies, refer to:
   - *Intelligent Printer Data Stream Reference*, S544-3417
   - *Font Object Content Architecture Reference*, S544-3285
3. RESOLUTION can be abbreviated as RES; METRICTECHNOLOGY can be abbreviated as METTECH.

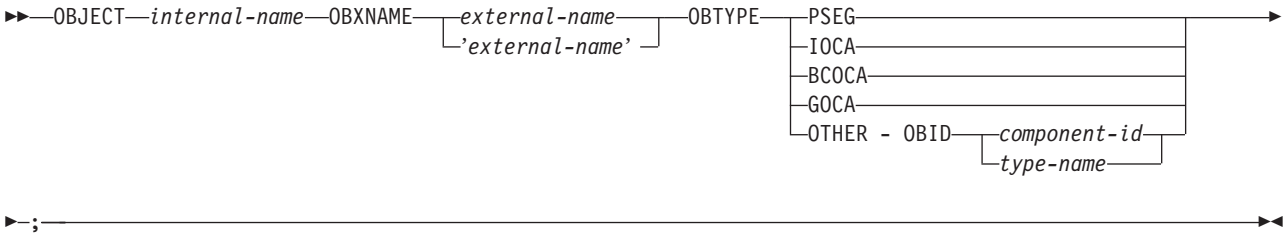
```
FORMDEF  xmp01
  FONTFID  YES ;

  PAGEDEF  xmp01    replace  yes ;
    FONT  xx2  res  240  mettech  fixed ;
    PRINTLINE  font  xx2 ;
```

Figure 97. Example of PPFA Support for Font Fidelity

In the example above, the form definition xmp01 specifies font fidelity and the page definition specifies a font that has 240 pels per inch resolution and fixed-metric technology. If a font with exactly those characteristics is not accessible by the printer, an error occurs and processing stops.

OBJECT Command (Traditional)



The OBJECT command allows you to define an external object to PPFA. Then you can use the PRINTLINE command with the OBJECT subcommand to place the defined object on a page.

You can use one PRINTLINE command to place one or many defined objects multiple times with different placement parameters on each placement. On the PRINTLINE OBJECT subcommand, enter information about the positioning, rotation, color, object size, and mapping instructions. All positioning is relative to the print line coordinate system. The *internal-name* appears on both the OBJECT command and on the PRINTLINE OBJECT subcommand, and is used similar to the way overlays and page segments are defined and placed (or printed).

Notes:

1. The *internal-name* is case insensitive but, other than that, the *internal-name* of the OBJECT command and of the PRINTLINE OBJECT subcommand must match exactly.
2. This function requires both print server and printer support. Check your print server and printer documentation.

OBJECT *internal-name*
Identifies the object and also is used to match a PRINTLINE OBJECT subcommand. The *internal-name* can be no more than 16 alphanumeric characters.

OBXNAME *external-name*
Specifies the external name of the resource object, which indicates where the object is located. For example, in OS/390, the *external-name* is the member name of the object in the object library. No prefixes are assumed on the name.

The *external-name* can be no more than 8 alphanumeric characters. If your operating system is AIX, the *external-name* is translated to EBCDIC.

Note: Items within quotation marks are not translated to uppercase or to their EBCDIC code equivalent.

OBTYPE

Used to specify the type of the object. Observe that each of the object types restricts the type of mapping option allowed in the placement of the object (**OBMAP** on the OBJECT subcommand on the PRINTLINE command).

PSEG Specifies a page segment object, as described in the *Mixed Object Document Content Architecture (MODCA) Reference*

Manual, (SC31-6802). All mapping types (**OBMAP**) are allowed by PPFA; however, print server issues an error if any of the objects contained in the page segment is not compatible with the coded OBMAP parameter.

GOCA

Specifies a graphics object, as described in the *Graphics Object Content Architecture (GOCA) Reference Manual*, (SC31-6804). GOCA allows you to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.

BCOCA

Specifies a bar code object, as described in the *Bar Code Object Content Architecture (BCOCA) Reference Manual*, (S544-3766). BCOCA allows you to specify only the LEFT parameter on the OBMAP subcommand.

IOCA Specifies a image object, as described in the *Image Object Content Architecture (IOCA) Reference Manual*, (SC31-6805). The IOCA object type allows you to specify the TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP command.

OTHER

Specifies other object data. The object data to be included is a paginated presentation object with a format that may or may not be defined by an IBM presentation architecture. When you specify OTHER, you must also specify the OBID parameter. The OTHER object type allows you to specify the TRIM, FIT, LEFT, CENTER, and FILL parameters on the OBMAP subcommand.

OBID

Specifies either a component identifier or a type name from Table 8. The OBID is translated into an Encoded OID and matched to the OID inside the object; they must match.

component-id
Specifies the component identifier.

type-name
Type-name is a name chosen by PPFA as an alternative to coding a component identifier.

Table 8. Non-OCA Objects supported by IOB.

Type-Name	Component-id	Description of OBID Object Type
EPS	13	Encapsulated PostScript
TIFF	14	Tag Image File Format
WINDIB	17	Device Dependent Bit Map [DIB], Windows Version
OS2DIB	18	Device Dependent Bit Map [DIB], PM Version
PCX	19	Paintbrush Picture File Format
GIF	22	Graphics Interchange Format
JFIF	23	JPEG file Interchange Format

Table 8. Non-OCA Objects supported by IOB. (continued)

Type-Name	Component-id	Description of OBID Object Type
PCLPO	34	PCL Page Object
PDFSPO	25	PDF Single Page Object

OVERLAY Command (Traditional)

This OVERLAY command is used to identify the overlay that will be positioned on a page at some spot other than the position defined within the overlay. This function is similar to the SEGMENT command. A separate OVERLAY command is required for each overlay. A maximum of 254 OVERLAY commands (each of the 254 names must be unique) can be specified for each page format.

The OVERLAY commands are nested within the PAGEFORMAT command.

```
PAGEFORMAT
    [ TRCREF ]
    [SEGMENT ]
    [ OVERLAY ]
    ...
    [ OVERLAY ]
```

For the overlay to be used, the line data must contain an Include Page Overlay (IPO) structured field. The same name must appear within the structured field as identified by this command, and the page origin must be stated.

To include page overlays without using the IPO structured field, see the "PRINTLINE Command (Traditional)" on page 231.

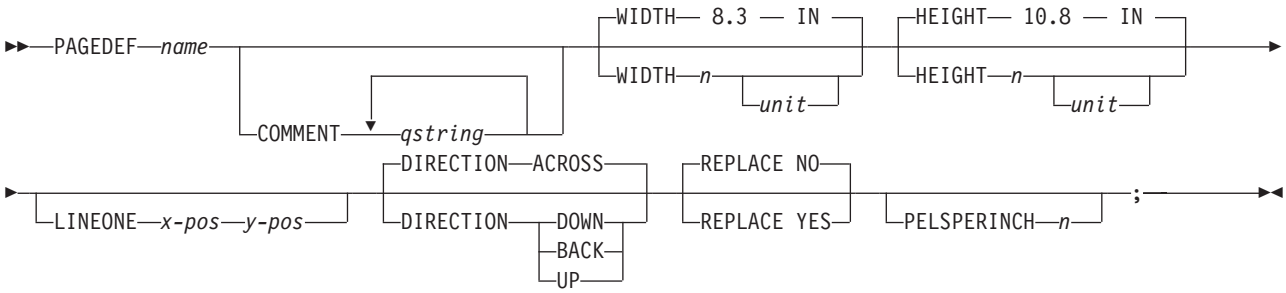
OVERLAY *name*;

Identifies the overlay that will be positioned on the page.

name Specifies the user-access name of an overlay to be used with the page definition.

Note: The prefix 'O1' is not part of the six-character user-access name. The overlay name can be an alphanumeric.

PAGEDEF Command (Traditional)



A page definition is a resource used to define how data is to be formatted on a logical page. When generated by PPFA, a page definition is stored as a resource in the page-definition library.

This command must be specified when you define a page definition. All of the PAGEDEF subcommands are optional; defaults are assumed.

Note: Values assigned within the subcommands or the default values become the values for any PAGEFORMAT subcommand not specified. REPLACE is not a PAGEFORMAT subcommand, so its default is not carried forward.

PAGEDEF
Identifies the page definition to be used with the print job.

name Defines an alphanumeric name of 1 to 6 characters for the page definition. When page definitions are generated, PPFA assigns the prefix 'P1' to this name as the external resource name.

Subcommands

COMMENT *qstring*
Specifies a user comment. This string comment is placed in the NOP structured field of the page definition.

qstring Specifies a quoted set of strings from 1 to 255 characters in total length.

WIDTH
Defines the width of the logical page.

n A number with up to three decimal places is used. The width may vary according to the type of printer being used. For more information, refer to your printer documentation. The default is 8.3 IN.

unit Specifies a unit of measurement for the WIDTH subcommand. The choices are IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

HEIGHT
Defines the height of the logical page.

n A number with up to three decimal places is used. The height may

vary according to the type of printer being used. For more information, refer to your printer documentation. The default is **10.8 IN**.

unit Specifies a unit of measurement for the HEIGHT subcommand. The choices are IN, MM, CM, POINTS, and PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

LINEONE
Specifies the values for the MARGIN and TOP parameters used in the POSITION subcommand of the PRINTLINE command.

x-pos Specifies the offset from the left edge of the logical page (margin position). The valid options for *x-pos* are described in the SETUNITS command for the horizontal value.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

y-pos Specifies the vertical offset from the top of the logical page (top line position). The valid options for *y-pos* are described in the SETUNITS command for the vertical value.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

DIRECTION
Specifies the print direction of the logical page. Not all printers can print in all print directions. For more information, refer to your printer documentation.

Note: Some printers such as the IBM 3835 Page Printer and the IBM 3900 Advanced Function Printer have a different media origin and require different direction settings than most page printers. For printing in the landscape page presentation when using wide forms, the PRESENT subcommand must be specified on the FORMDEF command to produce readable output. Alternatively, if you have existing page definitions, the UP direction can be used in the page definition without changes to the form definition to produce the same result.

ACROSS
The page is printed with the characters added *left to right* in each line, and the lines added from the top to the bottom.

DOWN
The page is printed with the characters added to the page from *top to bottom*, and the lines added from the right to the left.

BACK The page is printed with the characters added to the page from *right to left*, and the lines added from the bottom to the top.

UP The page is printed with the characters added to the page from *bottom to top*, and the lines added from the left to the right.

REPLACE
Specifies whether this page definition is to replace an existing one with the same resource name in the library.

NO This page definition does not replace one with the same resource name in the library.

If a page definition with the same resource name does not exist in the library, this page definition is stored.

YES If a page definition with the same resource name already exists in the library, this page definition replaces it.

If a page definition with the same resource name does not exist in the library, this page definition is stored.

PELSPERINCH *n*

Specifies the Logical Units in pels per inch for this page definition. Use the PELSPERINCH parameter to tell PPFA the pel resolution of your printer to generate more exact object placements.

n Specifies an integer number between 1 and 3,276, which determines the Logical Units in pels per inch.

Note:

If the L-Units are not specified on this page definition, they are defaulted to 240 pels per inch.

```
PAGEDEF xmp01 replace yes
  PELSPERINCH 300 ;

  PAGEFORMAT P1
     width  7 in
     height 3 in;
   PRINTLINE;

  PAGEFORMAT P2
     width  7 in
     height 3 in
     PELSPERINCH 1200;
   PRINTLINE;
```

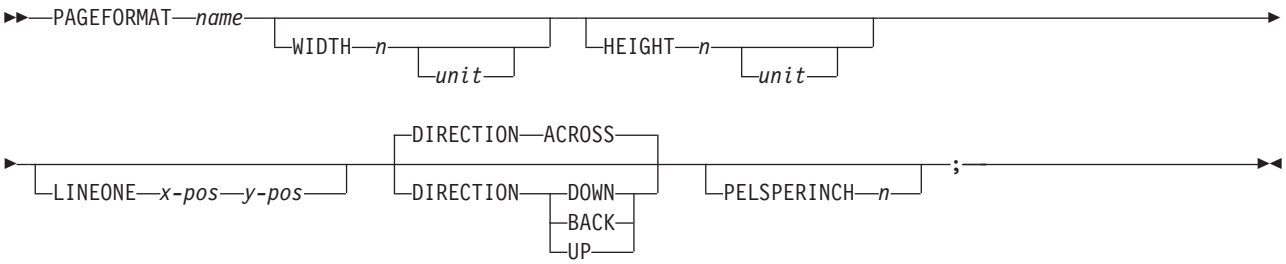
Figure 98. PELSPERINCH example

In the example above, the page definition xmp01 has specified L-Units as 300 pels per inch. Because the PAGEFORMAT P1 does not specify L-Units, it inherits 300 pels per inch. PAGEFORMAT P2 does specify L-Units as 1200 pels per inch.

The width and height in PAGEFORMAT P1 (7 in, 3 in) produces internal and structured field values of 2100 and 900, whereas in PAGEFORMAT P2 the same code produces values of 8400 and 3600, because of the difference in L-Units.

PAGEFORMAT Command (Traditional)

PAGEFORMAT Command



Page formats are subsets of page definitions. If you want to use more than one set of specifications to format a page within a single print job, you must use more than one page format. To change page formats, use conditional processing or insert an Invoke Data Map structured field in your print file. (Page formats are known to print server as data maps.) If you do not use conditional processing or if you do not insert an Invoke Data Map structured field, print server uses only the first page format in the page definition. Page formats are placed in the page definition in the order in which they are generated.

PAGEFORMAT subcommands have no fixed defaults. The entire PAGEFORMAT command and all of its subcommands can assume defaults. If any PAGEFORMAT subcommand is omitted, its value is selected from the corresponding subcommand in the governing PAGEDEF command.

This command can be omitted for the first page format in a page definition if only one page format is used. If omitted, PPFA assigns a page format name by using the page-definition name, including the 'P1' prefix.

PAGEFORMAT *name*

Specifies an alphanumeric name of 1 to 8 characters. This name must be unique within the page definition.

The following subcommands are used for each page format. They may be issued in the same way as in a page definition. Values specified in the PAGEDEF subcommands are used if any of the following subcommands are not defined within a page format.

Subcommands

WIDTH

Defines the width of the logical page.

n A number with up to three decimal places is used. The width may vary according to the type of printer being used. For more information, refer to your printer documentation.

unit Specifies a unit of measurement for the WIDTH subcommand. The choices are IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

HEIGHT
Defines the height of the logical page.

n A number with up to three decimal places is used. The height may vary according to the type of printer being used. For more information, refer to your printer documentation.

unit Specifies a unit of measurement for the HEIGHT parameter. The choices are IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

LINEONE
Specifies the values for the MARGIN and TOP parameters used in the POSITION subcommand of the PRINTLINE command.

x-pos Specifies the offset from the left edge of the logical page (margin position). The valid options for *x-pos* are described in the SETUNITS command for the horizontal value.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

y-pos Specifies the offset from the top of the logical page (top line position). The valid options for *y-pos* are described in the SETUNITS command for the vertical value.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

DIRECTION
Specifies the print direction of the logical page. Not all printers can print in all print directions. For more information, refer to your printer documentation.

Note: Some printers such as the IBM 3835 Page Printer and the IBM 3900 Advanced Function Printer have a different form origin and require different direction settings than most page printers. For printing in the landscape page presentation when using wide forms, the PRESENT subcommand must be specified on the FORMDEF command to produce readable output. Alternatively, if you have existing page definitions, the UP direction can be used in the page definition without changes to the form definition to produce the same result.

ACROSS
The page is printed with the characters added to the page from *left to right*, and the lines added from the top to the bottom.

DOWN
The page is printed with the characters added to the page from *top to bottom*, and the lines added from the right to the left.

BACK The page is printed with the characters added to the page from *right to left*, and the lines added from the bottom to the top.

UP The page is printed with the characters added to the page from *bottom to top*, and the lines added from the left to the right.

PELSPERINCH *n*

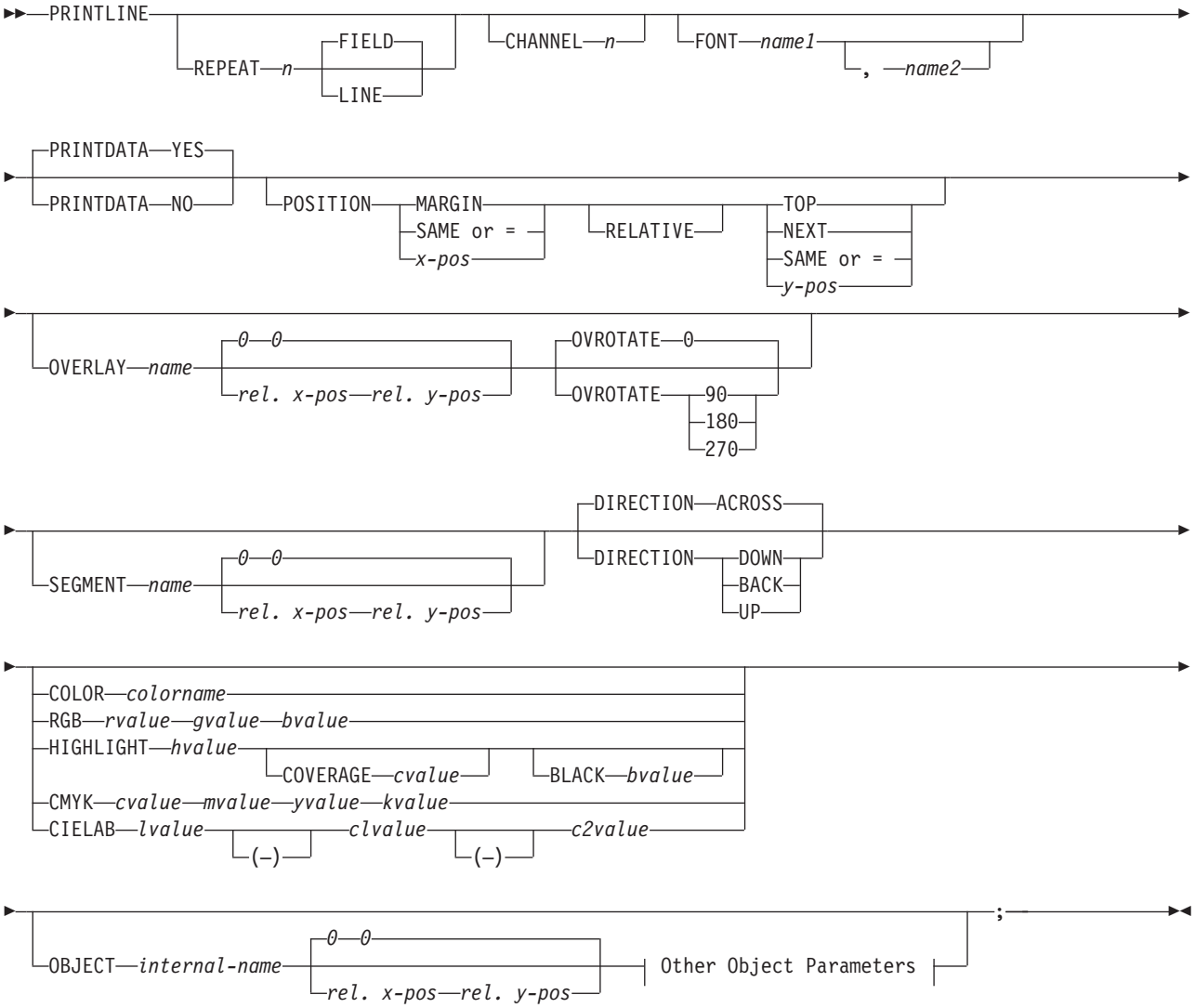
Specifies the Logical Units in pels per inch for this page format. Use the PELSPERINCH parameter to tell PPFA the pel resolution of your printer to generate more exact object placements.

n Specifies an integer number between 1 and 3,276, which determines the Logical Units in pels per inch.

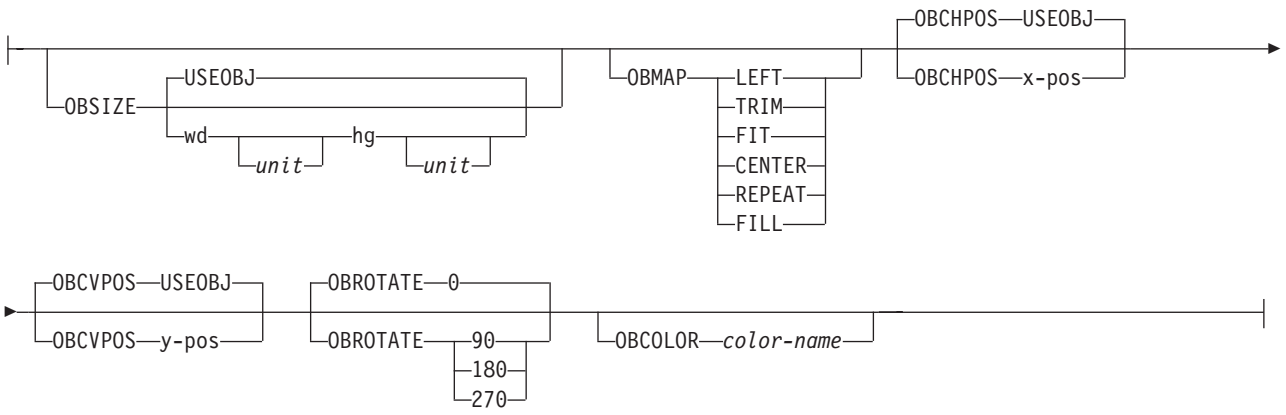
Note: If the L-Units are not specified on the page format, they are inherited from the page definition that contains this page format. See Figure 98 on page 227.

PRINTLINE Command (Traditional)

PRINTLINE (Traditional)



Other Object Parameters:



PRINTLINE

The PRINTLINE command specifies the printing of one data record on a line. If a *formatted* printline is to be printed, one or more FIELD commands

must follow the governing PRINTLINE command; at least one is required. If this is not done, field processing is not performed and the unformatted data is printed.

Subcommands

REPEAT

Specifies the number of printlines that are to be printed on a logical page. The direction and font specified within this printline applies to all lines printed. By using this command, you do not have to write specifications for each line.

Note: If the REPEAT subcommand is omitted, only one line is printed for this PRINTLINE command.

n This value specifies the number of printlines for a logical page; the maximum value is 65,535.

REPEAT 0

Not valid

REPEAT 1

Only one line is printed

If the CHANNEL or POSITION subcommands are specified within this PRINTLINE command, they apply only to the first line.

If this PRINTLINE is followed by several FIELD commands, the related field controls are also repeated.

FIELD Specifies that fields associated with repetitions of this PRINTLINE are to be positioned based on the first instance of the same field.

This parameter has no affect in fields with the same direction as the PRINTLINE of which they are a part.

This parameter specifies that the direction of repetition—for a given field—will be the direction of the first instance of this field, plus 90 °. Therefore, every field of an ACROSS PRINTLINE will be repeated down the page, *regardless of the direction of the FIELD*.

LINE Specifies that fields associated with repetitions of this printline are to be positioned based on the repetition of the PRINTLINE itself.

This parameter has no effect in fields with the same direction as the PRINTLINE of which they are a part.

This parameter specifies that the direction of repetition—for a given field—will be the direction of the associated PRINTLINE plus 90°. Therefore, every field of an ACROSS PRINTLINE will be repeated down the page, *regardless of the direction of the FIELD*.

CHANNEL *n*

Used to specify line spacing, skipping within a logical page, or page ejection (skipping to a new page). This subcommand is equivalent to the Forms Control Buffer (FCB) channel.

n The range of channels is 1 to 12. These correspond to carriage-control characters in the data. There is no default.

FONT

Defines the font to be used for the printline.

name1 Specifies the name of a font used to print the data. This font must have been defined in a previous FONT command in this page definition.

If Shift-Out, Shift-In (SOSI) processing is used, *name1* must be the single-byte font.

name2 Specify only when using Shift-Out, Shift-In (SOSI) processing to dynamically switch between a single-byte font and a double-byte font within the printline. *name2* must be the double-byte font.

Notes:

1. If this subcommand is not specified and TRC (Table Reference Character) bytes are specified in the print data, print server will use the font indicated by the TRC byte. Otherwise, print server will select a default font.
2. When selecting a font in AIX, you should consider that the text will be selected in EBCDIC, not ASCII. Therefore, an EBCDIC font and code page 500 (also called International #5) should be used for *name1*.

PRINTDATA

Specifies whether the line of data associated with the current PRINTLINE should be printed. The PRINTDATA subcommand is useful when the data stream is interspersed with lines of comments, blank lines, or lines without data that are not meant to be printed.

YES Specifies the data for the current PRINTLINE will be printed. YES is the default.

NO Specifies the data for the current PRINTLINE will not be printed.

Note: Any FIELD command that is associated with a PRINTLINE command that specifies PRINTDATA NO is ignored, and an error message is issued.

The default position for a PRINTLINE command that specifies PRINTDATA NO is position same same.

In Figure 99, the page definition xmp01 is expecting six lines of data. The

```
PAGEDEF  xmp01 ;
  SETUNITS  LINESP  1  LPI ;

  PRINTLINE ;
  PRINTLINE  PrintData  NO ;
  PRINTLINE  PrintData  yes ;
  PRINTLINE ;
  PRINTLINE  Segment X  PrintData NO  Overlay Y  Position Same Next ;
  PRINTLINE  PrintData yes ;
```

Figure 99. PRINTLINE NO example

LINESP parameter specifies that one line per inch is to be printed.

1. The first line of data will be read and printed.
2. The second line of data will be read, but not printed.
3. The third line of data will be read and printed one inch down from the first line.

4. The fourth line of data will be read and printed one inch down from the third line.
5. The fifth line of data will be read, but not printed.
   - The segment X will be printed.
   - The overlay Y will be printed.
6. The sixth line of data will be read and printed two inches down from the fourth line.

Note: The data line 2 was not printed and did not affect the positioning of the lines that followed. Line 3 was positioned as though line 2 did not exist.

POSITION

Specifies the starting position of the printline in the printout.

horizontal position

x-pos Specifies the horizontal offset from the left side of the logical page. The value is a number with up to three decimal places. The valid options for *x-pos* are described in the SETUNITS command for the horizontal value.

MARGIN

Specifies this line starts at the position specified as the horizontal (*x*) value in the previous LINEONE subcommand within this page definition.

SAME Specifies this line starts at the same horizontal offset position as the previous printline. If applied to the first printline of a logical page, the horizontal position is **0**, which is the default.

= Alternate for SAME.

RELATIVE

Specifies that the following vertical position value is to be processed as a relative value. The printline is positioned relative to the last printline placed on the page.

If a set of printlines were skipped over in the page definition because of a skip-to-channel carriage control, and the new active printline contains a relative vertical position, the output line will be positioned relative to the location of the last line printed on the page.

Note: If both TOP and RELATIVE are requested for the Y position value, the RELATIVE request is ignored.

When using RELATIVE positioning, PPFA does not flag off-the-page conditions for the position of a printline or for any overlays, segments or objects placed relative to that printline. Printlines that fall outside the bounds of the logical page are flagged by print server at run time.

When specifying RELATIVE, use the minus sign to indicate any negative values for the PRINTLINE vertical position; you may use the plus sign to indicate positive values. If no sign is used, a positive value is assumed.

The DIRECTION for a relative printline must be ACROSS. Fields associated with a relative printline must have the same DIRECTION as the printline and must match the pageformat DIRECTION.

If RELATIVE is specified with "SAME" or "=" as the "y" value, the relative value in the printline will be +0.

Relative positioning is allowed on a PRINTLINE command only if the PRINTLINE and all its associated FIELD commands are formatted to print in the same direction as the PAGEFORMAT. That is, the DIRECTION parameter in the PRINTLINE and any associated FIELD commands must specify (or default to) ACROSS. The DIRECTION in the PAGEFORMAT or PAGEDEF command may be any allowable value: ACROSS, DOWN, BACK, or UP.

The PRINTLINE command in which relative positioning is used must specify a CHANNEL parameter. The "n" value specified for the CHANNEL parameter cannot be used for any other PRINTLINE in the same PAGEFORMAT.

```
setunits linesp 6 lpi;
PAGEDEF rel9  replace yes
  direction across width 8.5 in height 11.0 in;
PRINTLINE channel 1 repeat 7 position 0 IN 1.0 IN;

/* The fields will be placed at +120 pels, +24 pels (next) */
/* and +48 pels (.20 IN) from lines previously placed on page */

setunits linesp 10 lpi;
PRINTLINE channel 2 repeat 2 position 0 relative next;
  FIELD START 1 LENGTH 3 position 0 IN .5 IN;
  FIELD START 4 LENGTH 3 position 0 IN next;
  FIELD START 7 LENGTH 3 position current .20 IN;
```

vertical position

y-pos Specifies the vertical offset from the top side of the logical page. The value options for *y-pos* are described in the SETUNITS command for the vertical value.

TOP Specifies that the printline is placed in the position specified as the vertical (*y*) value in the previous LINEONE subcommand within this page definition.

NEXT Specifies the printline is to be positioned down (on the logical page) one line (as defined in the LINESP subcommand of the last SETUNITS command) from the previous field. The LINESP subcommand of the SETUNITS command establishes the distance from one line to the next.

When NEXT is specified for the first printline of a logical page, the starting position of the line is one line down from the top of the logical page, which is the default.

Note: The "down" direction is determined by the direction of the logical page (as specified in the page format), not the printline direction. NEXT is, therefore, mainly useful in ACROSS printlines.

Note: For additional details on this area, please refer to the URL:
http://www.ibm.com/printers/R5PSC.NSF/Web/ppfaupdt

SAME Specifies this printline starts at the same vertical position as the previous printline.

= Alternate for SAME.

OVERLAY

Specifies the name of an overlay that is to be positioned relative to the location specified in the PRINTLINE command in which the OVERLAY subcommand was named. The PAGEFORMAT OVERLAY command may contain the named overlays. The maximum number of overlays specified for a PAGEFORMAT including the PRINTLINE OVERLAY subcommand is 254.

Specifies the electronic overlay that is to be used with this subgroup.

name Specifies the user-access name as defined in the Overlay command.

Notes:

1. PPFA checks for duplication of local names. If there is a duplication, the page definition is generated, but a warning message is issued.
2. PPFA does not check for duplicate user-access names.

relative_xpos relative_ypos

Specifies the number of units (inches, mm, and so on) that are added to the position of the printline to position the top-left corner of the overlay. The values for horizontal and vertical may be (+) or (-). The maximum value is + or - 32,760 L-units. For example:

- OVERLAY NAME1 2 in 1 in
- OVERLAY NAME2 5 mm 1 mm

Note: Any offset coded in the overlay itself is added to this offset.

OVROTATE { *0* **|** *90* **|** *180* **|** *270* **}**

Specifies the rotation of the placed overlay with respect to the x-axis of the page.

See “FORMDEF Command” on page 169 for an OVROTATE example, which is presented in the FORMDEF description.

SEGMENT

Specifies the name of a segment that is to be positioned relative to the location specified in the PRINTLINE command in which the SEGMENT subcommand was named. The PAGEFORMAT SEGMENT command may contain the named segments. The maximum number of segments specified for a PAGEFORMAT including the PRINTLINE SEGMENT subcommand is 127.

Specifies the page segment that is to be used with this subgroup.

name Specifies the user-access name as defined in the SEGMENT command.

Notes:

1. PPFA checks for duplication of local names. If there is a duplication, the page definition is generated, but a warning message is issued.
2. PPFA does not check for duplicate user-access names.

relative_xpos relative_ypos

Specifies the number of units (inches, mm, and so on) that are added to the position of the printline to position the top-left corner of the page segment. The values for horizontal and vertical may be (+) or (-). The maximum value is + or - 32760 L-units. For example:

- SEGMENT MYSEG1 2 in 1 in
- SEGMENT MYSEG1 5 mm 1 mm

DIRECTION

Specifies the print direction of the line relative to the upper-left corner as you view the logical page. Not all printers can print in all print directions. For more information about your printer, refer to your printer documentation.

If DIRECTION is not specified, the direction specified in the PAGEFORMAT command is used. Observe that this direction is additive to the direction specified in the PAGEFORMAT command. See 229.

ACROSS

The printline direction is rotated 0 degrees relative to the direction specified in the PAGEFORMAT (the printlines are oriented in the same direction as the page).

DOWN

The printline direction is rotated 90 degrees relative to the direction specified in the PAGEFORMAT.

BACK The printline direction is rotated 180 degrees relative to the direction specified in the PAGEFORMAT.

UP The printline direction is rotated 270 degrees relative to the direction specified in the PAGEFORMAT.

COLOR *colorname*

Specifies an OCA or defined color for the text of this field. This subcommand is recognized only by printers that support multiple-color printing. Refer to your printer publication for information about the colors that can printed.

colorname

Values for *colorname* are NONE, DEFAULT, BLACK, BLUE, BROWN, GREEN, PINK, RED, TURQ (turquoise), YELLOW, ORANGE, PURPLE, MUSTARD, GRAY, DARKBLUE, DARKGREEN, or DARKTURQ (dark turquoise). The color choices depend on the printer.

If you do not enter one of these colors, the default color for that printer is used. NONE is the color of the medium. DEFAULT is the printer default color.

Note: In some printer manuals, the color turquoise (TURQ) is called “cyan”, and the color pink (PINK) is called “magenta”.

PPFA supports the following synonyms:

- CYAN for TURQ
- DARKCYAN for DARKTURQ
- DBLUE for DARKBLUE
- DCYAN for DARKTURQ
- DGREEN for DARKGREEN
- DTURQ for DARKTURQ
- MAGENTA for PINK

Color Models

Specifies the color of print for this field supported in MO:DCA for the Red/Green/Blue color model (RGB), the highlight color space, the Cyan/Magenta/Yellow/Black color model (CMYK), and the CIELAB color model.

RGB *rvalue gvalue bvalue*

Three **RGB** integer values are used. The first (rvalue) represents a value for red, the second (gvalue) represents a value for green, and the third (bvalue) represents a value for blue. Each of the three integer values may be specified as a percentage from 0 to 100.

Note: An RGB specification of 0/0/0 is black. An RGB specification of 100/100/100 is white. Any other value is a color somewhere between black and white, depending on the output device.

HIGHLIGHT

Indicates the highlight color model. Highlight colors are device dependent, and can be specified for the IBM InfoPrint Hi-Lite Color Printer Model 4005-HCI.

You can use an integer within the range of 0 to 65535 for the *hvalue*.

Note: An *hvalue* of 0 indicates that there is no default value defined; therefore, the default color of the presentation device is used.

COVERAGE indicates the amount of coverage of the highlight color to be used. You can use an integer within the range of 0 to 100 for the *cvalue*. If less than 100 percent is specified, the remaining coverage is achieved with the color of the medium.

Note: Fractional values are ignored. If **COVERAGE** is not specified, a value of 100 is used as a default.

BLACK indicates the percentage of black to be added to the highlight color. You can use an integer within the range of 0 to 100 for the *bvalue*. The amount of black shading applied depends on the **COVERAGE** percentage, which is applied first. If less than 100 percent is specified, the remaining coverage is achieved with black.

Note: If **BLACK** is not specified, a value of 0 is used as a default.

CMYK

Defines the cyan/magenta/yellow/black color model. *Cvalue* specifies the cyan value. *Mvalue* specifies the magenta value. *Yvalue* specifies the yellow value. *Kvalue* specifies the black value. You can use an integer percentage within the range of 0 to 100 for any of the CMYK values.

CIELAB

Defines the CIELAB model. Use a range of 0.00 to 100.00 with *Lvalue* to specify the luminance value. Use signed integers from −127 to 127 with *c1value* and *c2value* to specify the chrominance differences.

Lvalue, c1value, c2value must be specified in this order. There are no defaults for the subvalues.

Note: Do not specify both an OCA color with the **COLOR** subparameter and an extended color model on the same **FIELD** or **PRINTLINE** command. The output is device dependent and may not be what you expect.

Do not specify two extended **COLOR** subcommands on the same **FIELD** or **PRINTLINE** command.

```
PRINTLINE POSITION 1 IN .5 IN
                COLOR BLUE ;
  PRINTLINE POSITION 1 IN 1 IN
                RGB 10  75  30 ;
  PRINTLINE POSITION 1 IN 1.5 IN
                cmyk 80 10 10 10 ;
  PRINTLINE POSITION 1 IN 2 IN
                CIELAB 80 100 20 ;
  PRINTLINE POSITION 1 IN 2.5 IN
                highlight 5 ;
  PRINTLINE POSITION 1 IN 2.5 IN
                highlight 300 COVERAGE 50 BLACK 30 ;
```

Figure 100. Color Model Usage

OBJECT

Note: This function requires both print server and printer support. Check your print server and printer documentation.

An included object is positioned and oriented in the following manner:

- All measurements are from the LND position established by the PRINTLINE position. Reference these measurements using the inline direction of the printline.
- Measure the"*relative-xpos* and *relative-ypos*" units from the PRINTLINE current position to determine the object area origin.
- Apply any rotation from OBROTATE to modify the PRINTLINE axis, and to create the new object area coordinate system.
- Use the OBSIZE parameter to determine the object area size within the object area coordinate system, and to define the object placement area.
- To determine the object content origin, apply the Object Content Offset from parameters OBCHPOS (OBject Content Horizontal Position) and OBCVPOS (OBject Content Vertical POSition) to the object area origin.

Note: The object content offset is used only for position (**LEFT**) and position and trim (**TRIM**) mapping options.

OBJECT *parameters*

Specifies the name of an object that is to be positioned and oriented relative to the location specified in the PRINTLINE command in which the OBJECT subcommand was named. The OBJECT, as identified by the internal-name parameter, must have been defined by an OBJECT command. You may place multiple objects on the same PRINTLINE command and you may place the same object multiple times. Each placement must have its own set of placement parameters, as follows:

internal-name

Specifies the name of an object that is up to 16 alphanumeric

characters in length. The *internal-name* is used to match the PRINTLINE OBJECT subcommand to its definition from the OBJECT command. An object must be defined with this internal name by the OBJECT command.

relative-xpos relative-ypos
Specifies the number of units (inches, mm, and so on) that are added to the position of the current printline to position the top-left corner of the object. The values for the horizontal and vertical positioning are limited by the type of printer used and the L-units specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

Each position specification can be a positive or negative number with up to three decimal places. The units specified can be one of the following: IN, MM, CM, POINTS, or PELS.

OBSIZE
Specifies the size of the object placement area. When no OBSIZE is specified, the default is the size specified in the object. If no size is specified in the object, the size of the page is used. The page width is as specified on the PAGEDEF or PAGEFORMAT commands, or it defaults to 8.3 inches by 10.8 inches.

wd Specifies the width of an object placement area as a number with up to three decimal places. The allowable width may vary with the type of printer used and the L-units specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

hg Specifies the height of the object placement area as a number with up to three decimal places. The allowable height may vary with the type of printer used and the L-units specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

unit Specifies a unit of measurement for the width parameter. The choices are: IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

USEOBJ
Specifies that the size measurements specified in the object are to be used. If no size is specified in the object, the size of the page is used, which is the length and width as specified on the PAGEDEF or PAGEFORMAT commands, or it defaults to 8.3 inches by 10.8 inches.

OBMAP
Specifies mapping options. The OBMAP parameter defines the mapping of the object to the object placement area. If OBMAP is not coded, the mapping option within the object is used. If the object does not contain a mapping option, then print server sets it to the created default for the container type.

Each object type (OBTYPE on the OBJECT command) dictates the allowable mapping options for that type. When it can, PPFA issues a message when these rules are violated. However, in the case of an

object type of page segment (OBTYPE=PSEG), PPFA does not know what types of objects are contained in it; therefore, PPFA cannot enforce the restrictions. See "OBJECT Command (Traditional)" on page 221 for a description of the restrictions.

LEFT Specifies that the object is positioned at the upper, left-hand corner of the object placement area, as defined or defaulted by the *relative-xpos*, *relative-ypos*, OBCHPOS, and OBCVPOS parameters. Any portion of the object that falls outside the object placement area as defined by the OBSIZE parameter is not trimmed and could cause an exception condition by the presentation system.

TRIM Specifies position and trim. The object is positioned at the upper, left-hand corner of the object placement area, as defined or defaulted by the *relative-xpos*, *relative-ypos*, OBCHPOS, and OBCVPOS parameters. Any portion of the object that falls outside the object placement area as defined by the OBSIZE parameter is trimmed.

FIT Specifies scale to fit; this is the default value if the OBMAP parameter is not coded. The object is to be scaled to fit within the object placement area, as defined by the OBSIZE parameter. The center of the object is placed in the center of the object placement area and the object is scaled up or down to fit the block. Scaling in the horizontal and vertical directions is symmetrical. The FIT parameter ensures that all of the data in the object is presented in the object placement area at the largest possible size. The object is not trimmed.

CENTER
Specifies that the center of the object be positioned at the center of the object placement area. Any portion of the object that falls outside the object placement area is trimmed.

REPEAT
Specifies that the origin of the data object be positioned with the origin of the object placement area. The object is then replicated in the X and Y directions. If the last replicated data does not fit in the object area, it is trimmed to fit.

FILL Specifies that the center of the data object be positioned coincident with the center of the object placement area. The data object is then scaled, so that it totally fills the object placement area in both the X and Y directions. This may require that the object be asymmetrically scaled by different scale factors in the X and Y directions.

OBCHPOS
Specifies the horizontal offset of the object contents within the object placement area.

x-pos The valid options for *x-pos* are described in the SETUNITS command for the horizontal value.

USEOBJ
Specifies that the offset value from the object is to be used. If no value is set in the object, the value defaults to 0.

OBCVPOS
Specifies the vertical offset of the object contents within the object

placement area, as defined by the OBSIZE parameter. If OBCVPOS is not specified, it defaults to USEOBJ and uses the value set in the object. If no value is set in the object, the value defaults to 0. The OBCHPOS parameter is used only in LEFT and TRIM mapping of the object into the object placement area.

y-pos Specifies a positive or negative number. The valid options for *y-pos* are described in the SETUNITS command for the vertical value.

USEOBJ
Specifies that the offset value from the object is to be used. If no value is set in the object, the value defaults to 0.

OBROTATE { 0 | 90 | 180 | 270 }
Specifies the object rotation with respect to the current LND's coordinate system.

OBCOLOR *color-name*
Specifies the color to be used as the default color or initial color for the object placement area. The OBCOLOR parameter is used only for objects of the PSEG, GOCA, BCOCA, and IOCA type. If the object type is OTHER, this parameter is ignored. Colors specified must be of the standard OCA color space.

color-name
Specifies standard OCA color space color names, which are: NONE, DEFAULT, BLACK, BLUE, BROWN, GREEN, RED, PINK (or MAGENTA), TURQ (or CYAN), YELLOW, DARKBLUE (or DBLUE), ORANGE, PURPLE, MUSTARD, GRAY, DARKGREEN (or DGREEN), and DARKTURQ (DTURQ, or DARKCYAN, or DCYAN).

In the following example, the page definition pd1 has defined an object with an external name of "PSEGxyz", of object type PSEG. The object has an internal name of "xyzintname". The internal name identifies the object for the PRINTLINE OBJECT subcommand when the object is placed. Observe that case is not significant on either the internal nor the external names.

```
PAGEDEF pd1 Replace Yes
  COMMENT 'this is my program';

  OBJECT xzZIntName
    OBXNAME PSEGxyz
    OBTYPE  PSEG ;

  PAGEFORMAT pf1;
    PRINTLINE
      OBJECT xyzintname -1.1 in 2.1 in
        OBSIZE 3 in  5 in
        OBMAP FILL
        OBCOLOR BLUE ;
```

Figure 101. Example of PPFA Support for IOB in a PAGEDEF

The PRINTLINE in PAGEFORMAT pf1 places the object on the page 1.1 inches to the left and 2.1 inches below the current printline position. It also maps the object into the object area with the FILL parameter, which centers the object in the object area and totally fills the area, possibly with different scaling factors in the X and Y

directions. It will have an area size of 3 by 5 inches, and overrides the default presentation space color to BLUE.

SEGMENT Command (Traditional)

Use the SEGMENT command only if you want page segments to be loaded to the printer before the page begins printing. If segments are used repeatedly and need to be available in the printer, this eliminates the need to load them each time. However, they do take up raster-pattern storage. If the segments are included on a page but not in the SEGMENT command, they are loaded to the printer as they are used in the print data.

A separate SEGMENT command is required for each page segment with a maximum of 127 SEGMENT commands within a single page format.

```
PAGEFORMAT
 TRCREF
 SEGMENT
 ...
 SEGMENT
```

A SEGMENT command is nested within the page format and follows the PAGEFORMAT command.

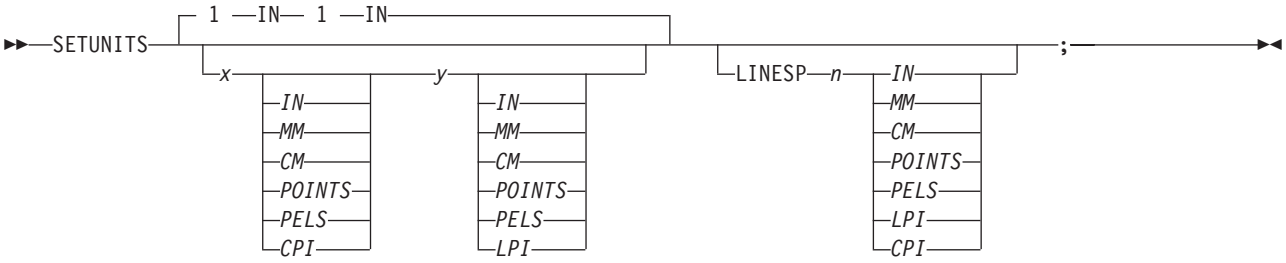
To include a page segment on a page without using an IPS structured field within the user data, see the “PRINTLINE Command (Traditional)” on page 231.

SEGMENT *name*

Specifies the alphanumeric name of 1 to 6 characters (user-access name) of the page segment. Each name must be unique within a single page format.

Note: The prefix ‘S1’ is not part of the six-character user-access name.

SETUNITS Command (Traditional)



The SETUNITS command specifies the value and the unit of measurement that will be the defaults for any subsequent measurement parameter in all of the commands and subcommands. These values remain the default values until another SETUNITS command is specified. The SETUNITS command should be specified as the first command in a page definition. If neither this command nor a measurement parameter is specified, the defaults identified within the following description are used.

SETUNITS

Specifies the value and the unit of measurement that will be the defaults for any subsequent measurement parameter in all of the commands and subcommands.

x-pos Specifies the number used for horizontal measurement. A number with up to three decimal places is used. The default is **1**. The choices are IN, MM, CM, POINTS, PELS, or LPI. The default is **IN**.

Note: This value affects subsequent OFFSET subcommands.

y-pos Specifies the number used for vertical measurement. A number with up to three decimal places is used. The default is **1**. The choices are IN, MM, CM, POINTS, PELS, or LPI. The default is **IN**.

Note: This value affects subsequent OFFSET subcommands.

Using CPI and LPI Units of Measurement

The CPI and LPI units of measurement make it possible to write the following command:

```
SETUNITS 10 CPI 6 LPI ;
```

This command sets the units of measurement for horizontal and vertical spacing in terms of characters per inch and lines per inch. You can then use the OFFSET subcommand specifications to increment the spacing one character or one line at a time. The distance specified by *n* characters over and by *n* lines down is defined in the governing SETUNITS command. In this example, there are 10 characters per inch (CPI) and 6 lines per inch (LPI).

Subcommand

LINESP *n unit*

Determines the line density or "leading" of the text. Any unit of measurement can be used. This subcommand values affects:

- The following PRINTLINE NEXT subcommand
- The vertical (y) position of the first line on a logical page when the LINEONE subcommand is not specified and the default is assumed

The default is **6 LPI**. If LINESP is allowed to default to 6 LPI, the LINEONE default is 1 L-unit less than 80% of 1/6 inch.

n The meaning is determined by the type of unit-of-measurement specified in the unit parameter.

LPI The number of lines per inch

All others
The distance between lines

unit Specifies a unit of measurement. The choices are:

IN Inch

LPI Lines-per-inch

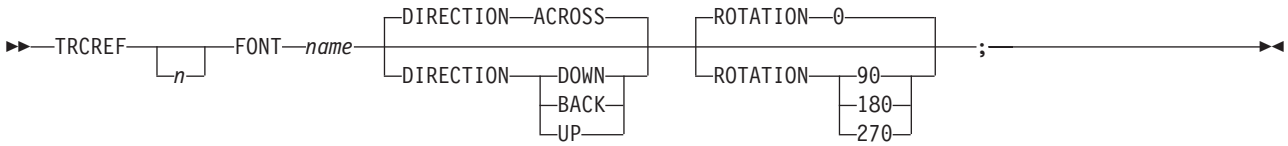
MM Millimeter

CM Centimeter

PELS L-units per inch (The number of L-units per inch can be defined by the user or can default to 240 L-units in an inch)

POINTS
Points per inch (72 points in an inch)

TRCREF Command (Traditional)



The TRCREF command specifies the relationship between a font and a table-reference character (TRC) in the data. When specified, the TRCREF command must immediately follow a PAGEFORMAT command.

PAGEFORMAT
TRCREF
SEGMENT
OVERLAY

Depending on the value specified for *n*, the TRC is interpreted by print server as being either S/370 1403 line-mode compatible or S/370 1403 line-mode incompatible: Notice that, if compatibility TRCs are to be used, no fonts should be specified in any PRINTLINE or FIELD commands within the same PAGEFORMAT.

0–3 Indicate a compatible TRC for a S/370 1403 line-mode data stream
4–126 Indicate a incompatible TRC for a S/370 1403 line-mode data stream

Also notice that any TRC number outside the range of 0-3 results in non-compatibility TRCs for the entire page definition. If compatibility TRCs are used, do not specify fonts on PRINTLINE or FIELD commands within the same PAGEFORMAT.

TRCREF *n*

Specifies the TRC numbers that can appear in print data.

n The allowable values are 0 to 126; each TRCREF command must contain a unique number within a page format.

If *n* is omitted, PPFA automatically adds one to the *n* value of the previous TRCREF command in the sequence and assigns that value.

The default for the first TRCREF command is **0**.

Note: You may have multiple TRCs pointing to the same font.

Subcommands

FONT *name*

Specifies the font that is associated with the TRC number.

name Specifies the local name of a font; the font must be one that has been named in a FONT command.

If you have used both the user-access name and the local name in the FONT command, use the local name here. If you have used only the user-access name, use it here.

DIRECTION

Specifies the print direction of the line relative to the upper-left corner as

you view the logical page. Not all printers can print in all print directions. For more information about your printer, refer to your printer documentation.

The DIRECTION on the TRCREF command must match the DIRECTION of the PRINTLINE command with which the TRC is to be used. If TRCREF DIRECTION subcommand is not specified, DIRECTION ACROSS is assumed. Observe that this direction is additive to the direction specified in the PAGEFORMAT command.

ACROSS
The page is printed with the characters added to the page from *left to right,* and the lines added from the top to the bottom.

DOWN
The page is printed with the characters added to the page from *top to bottom,* and the lines added from the right to the left.

BACK The page is printed with the characters added to the page from *right to left,* and the lines added from the bottom to the top.

UP The page is printed with the characters added to the page from *bottom to top,* and the lines added from the left to the right.

ROTATION
Specifies the rotation of characters in degrees. The specified value is relative to the inline direction of the printline.

Valid rotations are 0°, 90°, 180°, or 270°; **0°** is the default.

If the TRCREF ROTATION subcommand is not specified, the default is the rotation value specified on the FONT command.

TRCREF (Traditional)

Chapter 11. Page Definition Command Reference (Record Formatting)

This section describes Record Formatting Line Data Processing (for Traditional Line Data Processing, refer to "Chapter 10. Page Definition Command Reference (Traditional)" on page 195), and includes:

- Sequence of record formatting commands for page definitions
- Commands listed alphabetically
- Detailed information on each command
- Descriptions of the applicable subcommands and parameters for each command

Sequence of Record Formatting Commands for Page Definitions with LAYOUT

```
 [ SETUNITS ...]
PAGEDEF
 FONT
[OBJECT ... ]
[DEFINE COLOR... ]
[ PAGEFORMAT ]
   [ SEGMENT ...]
   [ OVERLAY ...]
   [ LAYOUT ...]
    [ CONDITION ...]
    [ FIELD ...]
    [ DRAWGRAPHIC ...]
    [ ENDGRAPHIC ...]
[ PAGEFORMAT ]
   [ SEGMENT ...]
   [ OVERLAY ...]
   [ LAYOUT ...]
    [ CONDITION ...]
    [ FIELD ...]
    [ DRAWGRAPHIC ...]
    [ ENDGRAPHIC ...]
```

- LAYOUT commands and PRINTLINE commands cannot be used within the same PAGEDEF. At least one LAYOUT command is required per page format for a record formatting page definition.
- A SETUNITS command can be placed before any other PPFA command. The values set are in effect until the next SETUNITS command.
- SEGMENT and OVERLAY commands must be specified under their associated PAGEFORMAT command.
- The first PAGEFORMAT command can be omitted in a page definition, if the page definition contains only one page format. If the PAGEFORMAT command is omitted, the PAGEDEF command parameters are used to define the page format.
- One file can contain multiple sets of page definitions.

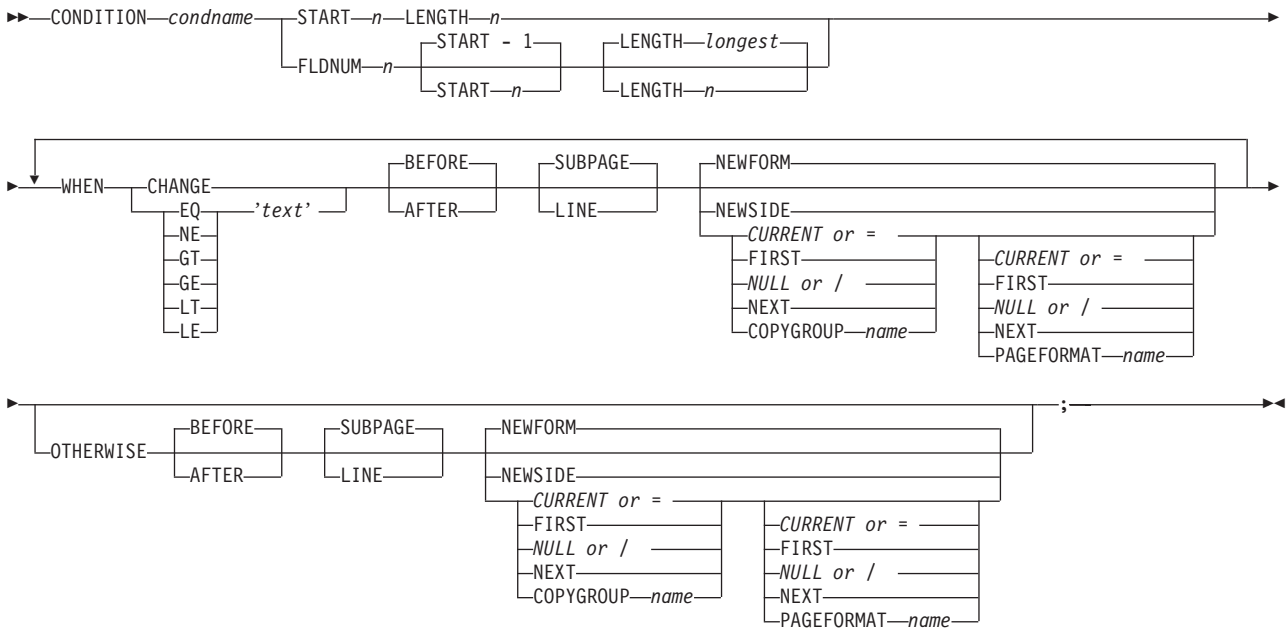
Diagram Shorthand

These terms are used in the command definitions:

x-pos A horizontal position using a numeric number followed optionally by a unit. For the available units, see “Units of Measurement” on page 152.

y-pos A vertical position using a numeric number followed optionally by a unit. For the available units, see “Units of Measurement” on page 152.

CONDITION Command (Record Format)



Short Form

CONDITION

The CONDITION command examines data in an input record and specifies actions to be taken based on the result of the examination.

- The *condname* parameter must come before any subcommands
- No WHEN subcommand can follow an OTHERWISE subcommand in the same CONDITION command

condname

Names the condition. The name must contain 1 to 8 alphanumeric characters.

PPFA allows cross-referencing to the *condname*. The cross-reference is done by using the short form of the CONDITION command (second format in the syntax table). By specifying a previously defined *condname*, PPFA uses the specifications from that command. When the condition is reused, the point where you want the comparison to begin may be at a different point in the record. By specifying the optional START subcommand, you can change the starting point of the comparison but not the field length. If the START subcommand is not specified, the starting point is the same as defined in the original CONDITION command.

Subcommands

START *n*

Specifies the starting position of characters to be compared within the data base where the comparison is to be done.

n Specifies the number of bytes from the first data byte in the record as the starting point of the comparison field. The first data byte position of an input record is 1.

Note: The carriage-control character and the record id are not considered data, therefore the first character of data starts in column eleven.

LENGTH *n*

Specifies the length of the field to be compared within the database.

n Specifies the number of bytes in the data record to be compared, beginning with the position specified in START. Valid values are numbers from 1 to 8000. The length of the constant text must be the same as defined in this parameter or the results are invalid.

Comparisons are done on a byte-by-byte basis. Because the comparison field and the constant text must have the same lengths, padding is not necessary.

Note: If any part of the comparison field specified by the combination of START and LENGTH is outside the boundaries of the data record, all conditional processing is not performed. No WHEN is executed. If an OTHERWISE is present, it is not executed either.

FLDNUM

Field number to be used in comparison. This keyword should only be used if the DELIMITER field was used in the LAYOUT command. Fields cannot be counted without delimiters being specified in the database. When counting, the first field after the record id is to be considered FLDNUM 1.

To allow for the identification of a part of a field which has been numbered, you can specify the starting position (from the delimiter) and the length of the field to be used in the WHEN condition (the default of the *longest* parameter is the length of the longest condition or when no specific condition is specified [i.e. when change] it is from the starting position to the end of the field.)

WHEN *parameters*

Marks the start of the conditional comparison parameters. At least one WHEN subcommand is required.

comparisontype= { EQ | NE | GT | GE | LT | LE }

Specifies the type of comparison that is to be performed between the data in the comparison field (the portion of the record specified by START and LENGTH) and the constant text defined in the *text* parameter.

The choices are:

EQ equal to
NE not equal to
GT greater than
GE greater than or equal to
LT less than
LE less than or equal to

text
Specifies constant text for comparison with the comparison field text. The constant text length must be the same as the value on the LENGTH subcommand, with a maximum length of 8000 bytes. Examples of valid text are:

```
2C(3)'AB'
K'321,400'
X'41FE7799' 2 'CHARS'
```

Any values or parameters that are valid for the TEXT subcommand within the FIELD command may be used as text; see the TEXT subcommand in "Subcommands" on page 275.

CHANGE
Specifies that the contents of the comparison field in this record are to be compared with the field in the record last processed by the same CONDITION command.

This parameter is an alternative to the *comparisontype* and *text* parameter combination but can be specified only once in a CONDITION command.

The results of the comparison is either TRUE or FALSE.

TRUE When the contents of the comparison field have changed from one record to the next.

FALSE
When print server processes the data, if the comparison field lies outside the boundary of the current record, which may occur with variable-length records or with truncated trailing blanks, the current record will not be used in future comparisons.

CHANGE is always false if used with the first WHEN subcommand of a series (no previous record to compare against). Whenever a new data map (one with a different name) is invoked, all the CHANGE comparisons are reset. Field values in the previous data map are not retained.

BEFORE
Specifies that the conditional action takes place before the current line or subpage is processed. This is the default.

AFTER
Specifies that the conditional action takes place after the current line or subpage is processed.

LINE
Specifies that the conditional action takes place either before or after the current line.

SUBPAGE
Specifies that the conditional action takes place either before or after the current subpage. This is the default.

NEWFORM
NEWFORM specifies that the only action to be taken is skipping to the front of a new form (sheet) and restarting the page format.

Note: This parameter is an alternative to using the copygroup and pageformat parameters, and is equivalent to specifying CURRENT for the copy group parameter and NULL for the pageformat

parameter. CURRENT NULL are the respective defaults for copy group and pageformat parameters; therefore, NEWFORM is the default action.

NEWSIDE
Specifies that the only action to be taken is skipping to a new side (either the back of the current sheet or the front of a new sheet) and restarting the page format.

Notes:

1. This parameter is an alternative to using the copygroup and pageformat parameters, and is equivalent to specifying NULL for the copy group parameter and CURRENT for the pageformat parameter.
2. Conditional processing does not result in unnecessary blank pages.

If the line currently being processed is the first line on a side, then:

- A *copygroup* or NEWFORM action taking effect BEFORE LINE does not force an additional new form.
- A *pageformat* or NEWSIDE action taking effect BEFORE LINE does not force an additional new side.

Similarly, additional sides or forms are not forced by BEFORE SUBPAGE if the line currently being processed is in the first subpage on a side or a form.

copygroup options
Specifies a copy group to be invoked if the condition is true.

Note: Any copy group action (except NULL) restarts the page format.

{ CURRENT or = }
Invoke the current copy group again. This results in ending printing on the current sheet and resuming on the front side of a new sheet. This is the default.

The page format is restarted. This means that the first input record to go on the new page is printed using the first LAYOUT command of the current page format, and so on. For example, data that was to be printed as subpage 4 on the sheet might be printed on subpage 1 on the new sheet.

FIRST Invokes the first copy group in the current form definition.

{ NULL or / }
Retains the current copy group, taking no action.

NEXT Invokes the next copy group in the current form definition.

Note: If NEXT is specified from the last copy group in the form definition, the first copy group in the form definition will be used.

COPYGROUP *cgname*
Uses the named copy group defined in the current form definition. The name must contain 1 to 8 alphanumeric characters.

pageformat options
Specifies a page format to be invoked if the condition is true.

{ CURRENT or = }
Invokes the current page format again. This results in ending printing on the current sheet and resuming on the front side of a new sheet.

The page format is restarted. This means that the first input record to go on the new page is printed using the first LAYOUT command of the current page format, and so on.

FIRST Invokes the first page format in the current page definition.

{ NULL or / }
Retains the current page format, taking no action. This is the default.

NEXT Invokes the next page format in the current page definition.

Note: If NEXT is specified from the last page format in the page definition, the first page format in the page definition will be used.

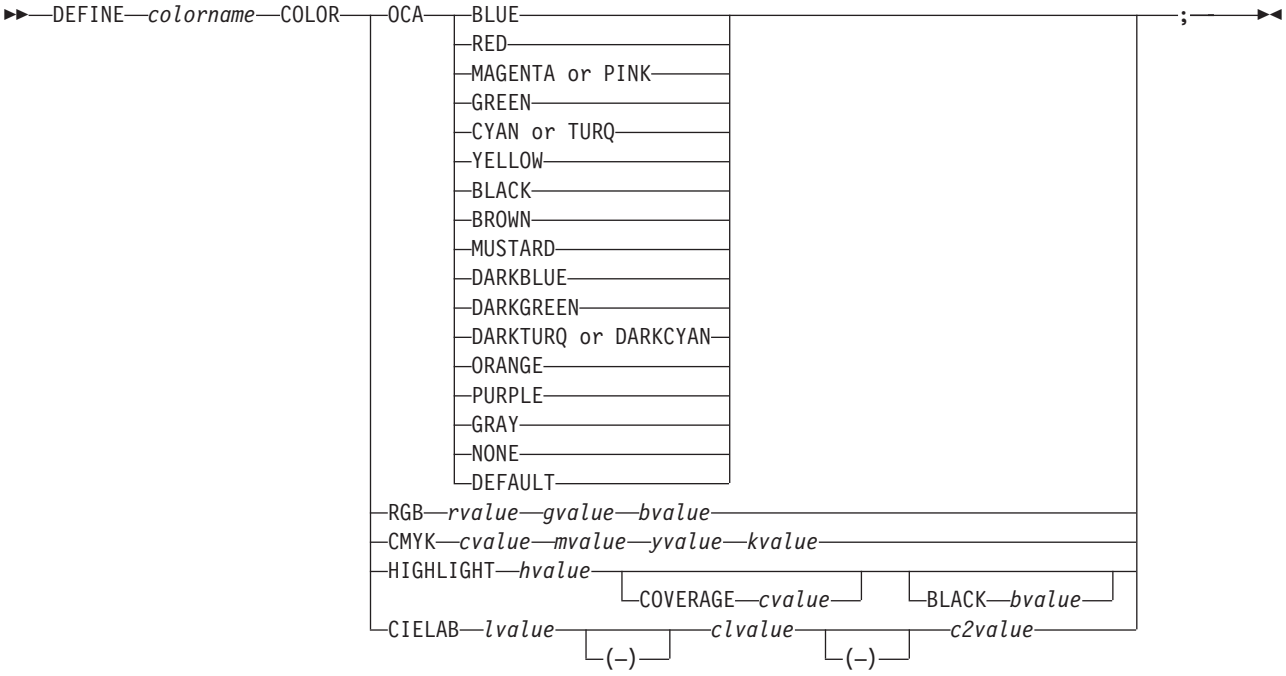
PAGEFORMAT *pfname*
Uses the named page format defined in the current page definition. The name must contain 1 to 8 alphanumeric characters.

OTHERWISE *parameters*
Marks the start of a conditional action to be taken if all preceding WHEN comparisons have proved false. The syntax is the same as the WHEN subcommand, except that the comparison parameters (*comparisontype text* or 'CHANGE') are not used. See the WHEN parameters starting with BEFORE on page 254 for a description of the parameters.

If the OTHERWISE subcommand is not used within the sequence, no action is taken. This is the same as if OTHERWISE NULL NULL had been entered.

Note: OTHERWISE is not executed if any part of the comparison field specified by the combination of START and LENGTH is outside the boundaries of the data record.

DEFINE COLOR Command (Record Format)



colorname
This command is used to predefine a color model to be used as a colorname in any later command, such as the DRAWGRAPHIC command.

Subcommands

COLOR
Defines a color name of a particular color model such as OCA, RGB, CMYK, HIGHLIGHT, or CIELAB. This name can be used anywhere color of that model is allowed. For example a defined color of any color model can be used as text color in the FIELD or PRINTLINE commands, but only a color defined as an OCA color can be used as an object placement area color. See the OBCOLOR subcommand in "LAYOUT Command (Record Format)" on page 289.

colorname

Select a 1 to 10 character name. Use this name on the command to identify this color. For example:

```
DEFINE oldblue COLOR OCA brown;
PRINTLINE   COLOR oldblue;
```

Color Model
Specifies the color of print for this field supported in MO:DCA for the OCA, the Red / Green / Blue color model (RGB), the highlight color space, the Cyan / Magenta / Yellow / Black color model (CMYK), and the CIELAB color model.

OCA Chose one of the standard OCA colors from the previous syntax diagram.

Note: In some printer publications, the color turquoise (TURQ) is called “cyan”, and the color pink (PINK) is called “magenta”.

PPFA supports the following synonyms:

- CYAN for TURQ
- DARKCYAN for DARKTURQ
- DBLUE for DARKBLUE
- DCYAN for DARKTURQ
- DGREEN for DARKGREEN
- DTURQ for DARKTURQ
- MAGENTA for PINK

RGB *rvalue gvalue bvalue*

Three **RGB** integer values are used. The first (rvalue) represents a value for red, the second (gvalue) represents a value for green, and the third (bvalue) represents a value for blue. Each of the three integer values may be specified as a percentage from 0 to 100.

Note: An RGB specification of 0/0/0 is black. An RGB specification of 100/100/100 is white. Any other value is a color somewhere between black and white, depending on the output device.

HIGHLIGHT *hvalue* **COVERAGE** *cvalue* **BLACK** *bvalue*

Indicates the highlight color model. Highlight colors are device dependent.

You can use an integer within the range of 0 to 65535 for the *hvalue*.

Note: An *hvalue* of 0 indicates that there is no default value defined; therefore, the default color of the presentation device is used.

COVERAGE indicates the amount of coverage of the highlight color to be used. You can use an integer within the range of 0 to 100 for the *cvalue*. If less than 100 percent is specified, the remaining coverage is achieved with the color of the medium.

Note: Fractional values are ignored. If **COVERAGE** is not specified, a value of 100 is used as a default.

BLACK indicates the percentage of black to be added to the highlight color. You can use an integer within the range of 0 to 100 for the *bvalue*. The amount of black shading applied depends on the **COVERAGE** percentage, which is applied first. If less than 100 percent is specified, the remaining coverage is achieved with black.

Note: If **BLACK** is not specified, a value of 0 is used as a default.

See “Color on the IBM InfoPrint HiLite Color Post Processor” on page 46 for more information.

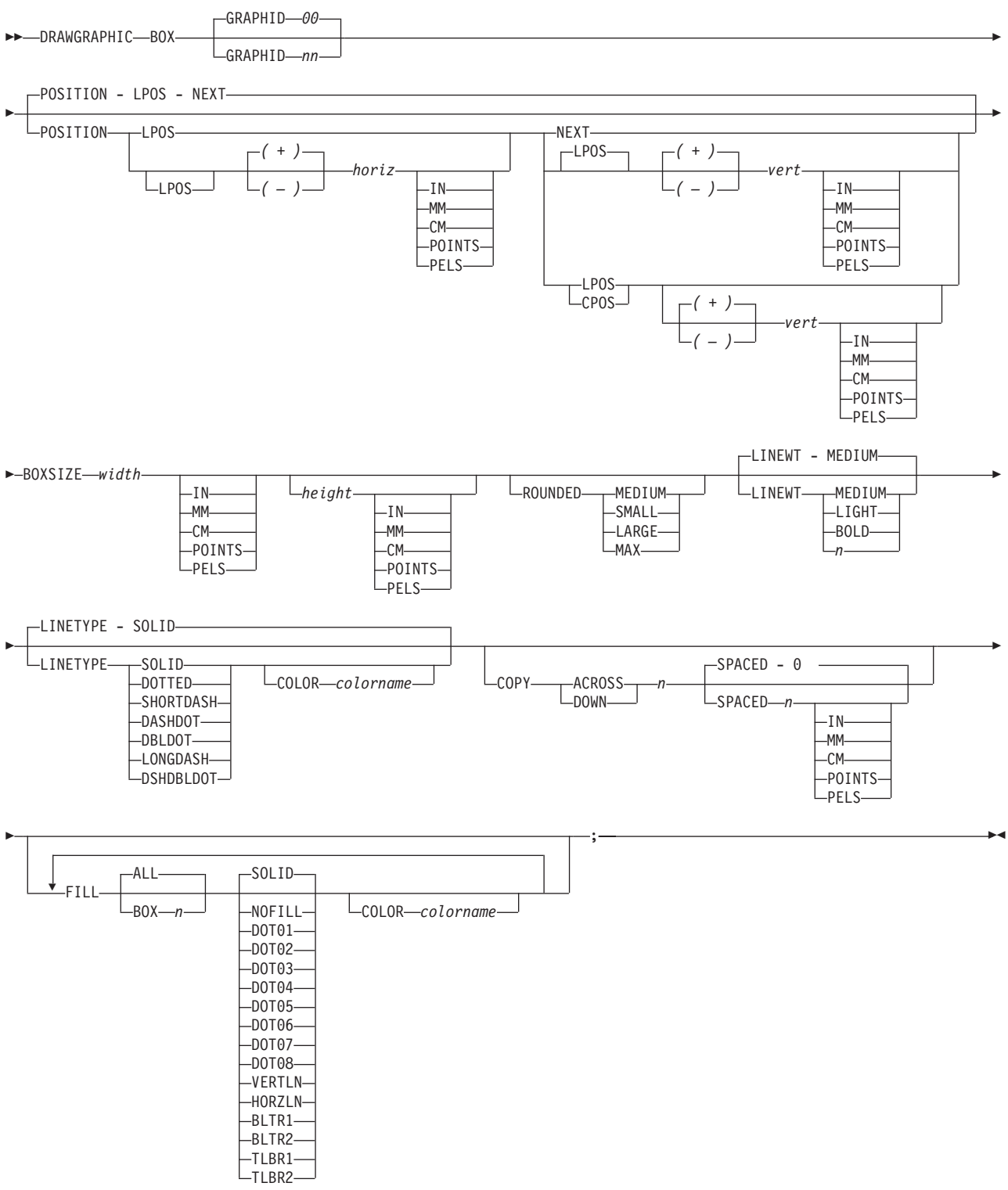
CMYK *cvalue mvalue yvalue kvalue*
Defines the cyan / magenta / yellow / black color model. *Cvalue* specifies the cyan value. *Mvalue* specifies the magenta value. *Yvalue* specifies the yellow value. *Kvalue* specifies the black value. You can use an integer percentage within the range of 0 to 100 for any of the CMYK values.

CIELAB *Lvalue* **(−)***c1value* **(−)***c2value*
Defines the CIELAB model. Use a range of 0.00 to 100.00 with *Lvalue* to specify the luminance value. Use signed integers from −127 to 127 with *c1value* and *c2value* to specify the chrominance differences.

Lvalue, c1value, c2value must be specified in this order. There are no defaults for the subvalues.

Note: Do not specify both an OCA color with the **COLOR** subparameter and an extended color model on the same **FIELD** or **LAYOUT** command. The output is device dependent and may not be what you expect.

DRAWGRAPHIC Command - Box (Record Format)



The DRAWGRAPHIC-Box command allows you to generate GOCA objects in order to draw boxes on the page.

Note: GOCA boxes require specific microcode in your printer.

This command allows you to draw a box of varying attributes and colors at either the current line position or a specified position. DRAWGRAPHIC can be used with the COLOR parameter and DEFINE COLOR to shade a box with a percentage of black or other colors.

Subcommands

GRAPHID

Specified number is used to later identify as the box or set of boxes to be closed by the ENDGRAPHIC. The default is '00'.

POSITION

Horizontal and vertical position for first box. This position is relative to the LAYOUT command's position statement or the current position.

LPOS and CPOS refer to Layout Position and Current Position respectively. If LPOS is used alone, the position is used exactly at the same position as is specified on the LAYOUT command. If it is used with a + or − value, the position moves that amount from the LAYOUT position. The same is true for Current position except that the position is taken from the previous FIELD or DRAWGRAPHIC command.

BOXSIZE

Specify the horizontal and, optionally, vertical dimensions of the box. The first parameter is required and specifies the horizontal width of the box, which is a fixed size. The second parameter is optional and if given, specifies the fixed vertical depth of the box. If the second parameter is omitted, the box will be a variable size or ″floating″ box. For a floating box, the depth of the box is determined when the box is closed with an ENDGRAPHIC command.

ROUNDED

Size of the rounded cornerlength is determined by the following parameters:

- **MEDIUM** = Medium cornerlength - equates to a radius of 20 pels at 240 pels/inch or 120 pels at 1440 pels/inch
- **SMALL** = Small cornerlength - equates to a radius of 10 pels at 240 pels/inch or 60 pels at 1440 pels/inch.
- **LARGE** = Large cornerlength - equates to a radius of 30 pels at 240 pels/inch or 180 pels at 1440 pels/inch
- **MAX** = Maximum cornerlength gives an arc with a radius that extends half the length of the shortest box side. If boxes are rounded MAX, they cannot be open-ended.

line weight

Specify either one of the following keywords or the number of lineweights to be used (1 lineweight = .01 inch). Specify 0 if you want invisible borders (type and color are then ignored).

- **LIGHT** = the same as LINEWT .01 inch.
- **MEDIUM** = the same as LINEWT .02 inch.
- **BOLD** = the same as LINEWT .03 inch.

LINETYPE

Specify one of the following keywords for the border type: **SOLID,**

DOTTED, SHORTDASH, DASHDOT, DBLDOT (double dot), LONGDASH, or DSHDBLDOT (dash double dot).

COLOR
Color to be used for the box border. The colorname must be either one of the pre-defined OCA keywords or the colorname from the DEFINE COLOR command.

COPY Repeat the same box at regular intervals either across or down the page. Total number of boxes is one more than the value specified on this parameter.

Restriction: If boxes are repeated in the DOWN direction, they cannot be open-ended.

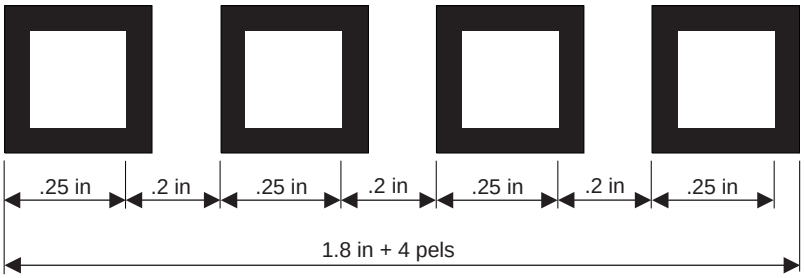


Figure 102. Spaced Boxes (not to scale).

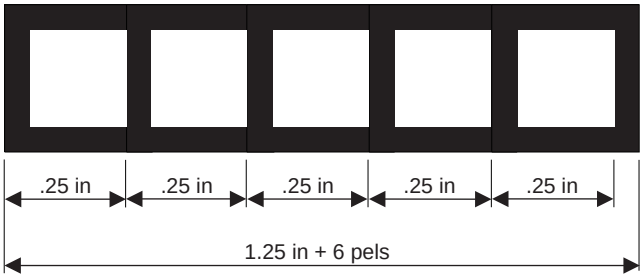


Figure 103. Boxes Spaced 0 (not to scale).

SPACED
Spacing between the boxes can be specified directly. The default is to have no space between the boxes. If there are no spaces between the boxes, the common border is shared and not duplicated.

FILL Allows the option of filling a box with a pre-defined GOCA pattern and optionally specifying a color. The numbering of the boxes is done in the order they are defined within this one command - 1,2,3,.... Filling follows the rule that the ″last fill wins″.

The Nofill keyword fills ALL boxes with one fill pattern, then specify Nofill on one box to remove that box's pattern.

For an example of the various GOCA-supported fill patterns, see Fill Patterns for Drawgraphic Commands, Figure 119 on page 379.

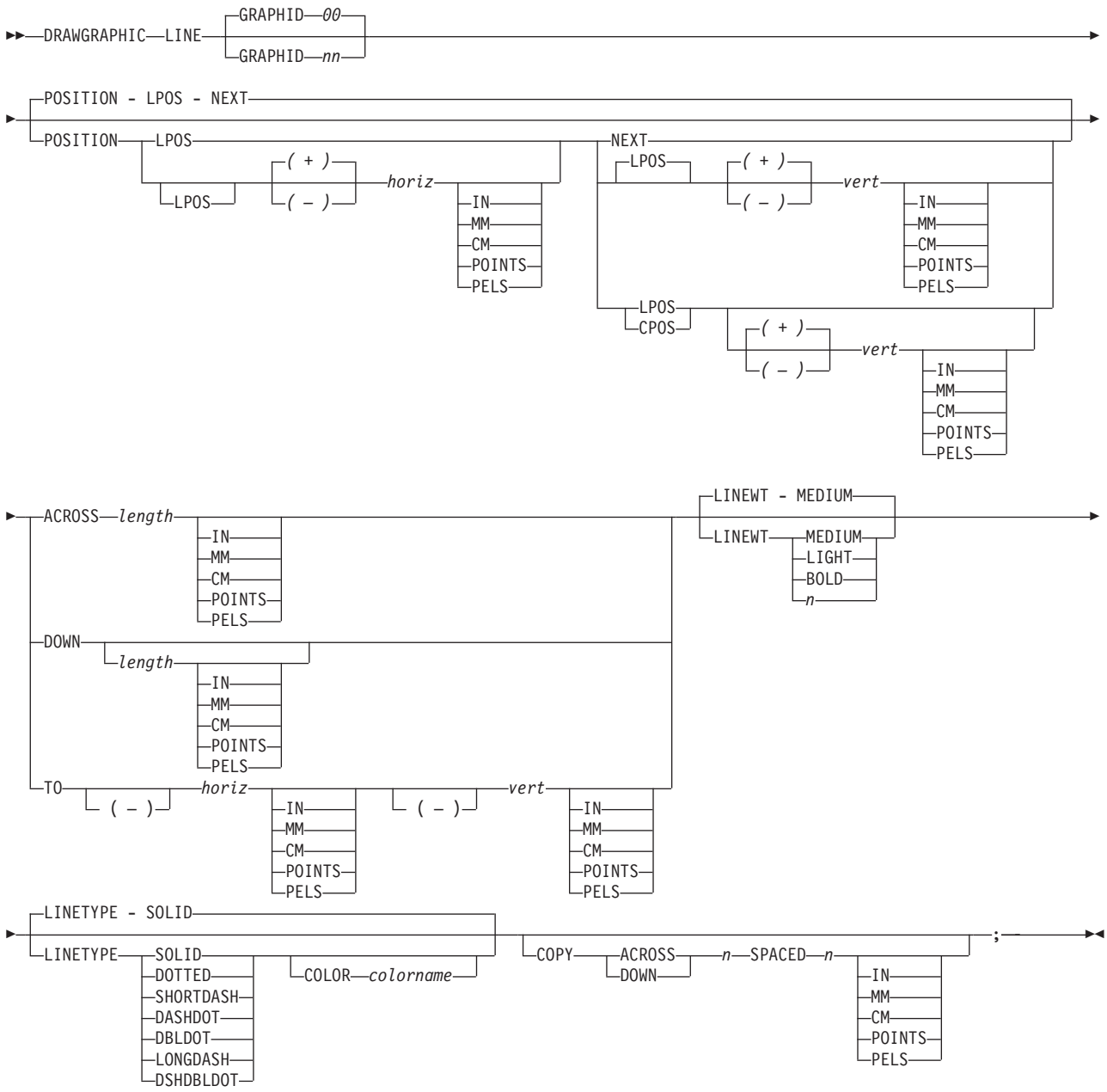
The NOFILL keyword can be used when a series of boxes has been specified as filled and one or more of them are to be left empty. In the example, boxes 1, 2, 4, and 5 will be filled with solid blue and box 3 will be empty:

```
LAYOUT ...
Drawgraphic BOX boxsize 1 in .2 in copy down 4
Linetype solid color green
FILL ALL SOLID Color Blue
FILL Box 3 NOFILL;
```

ALL All boxes are filled.

BOX *n* Boxes are numbered starting at 1 for the initial box in this command, and increasing through the use of the COPY parameter.

DRAWGRAPHIC Command - Line (Record Format)



The DRAWGRAPHIC-Line command allows you to use GOCA (Graphic Character Global Identifier) objects in order to draw lines on the page.

Note: GOCA lines require specific microcode in your printer.

The DRAWGRAPHIC-Line command allows you to create either one straight line or a series of straight lines from either the current line position or a specified position.

Subcommands

GRAPHID
Specifies a number used to later identify the graphic line to be closed by the ENDGRAPHIC. The default is '00'.

POSITION

Horizontal and vertical position for the start of the first line. This position is relative to either the Layout Position parameter or the current position.

LPOS and CPOS refer to Layout Position and Current Position respectively. If LPOS is used alone, the position is used exactly at the same position as is specified on the LAYOUT command. If it is used with a + or − value, the position moves that amount from the Layout position. The same is true for Current position except that the position is taken from the previous FIELD or DRAWGRAPHIC command.

ACROSS or DOWN
Specify the line length in either the ACROSS or DOWN directions. If ACROSS is specified, the line length must also be specified. If DOWN is specified and the *n units* value is not entered, the line will continue until either a logical page eject is executed or an ENDGRAPHIC is found.

TO Horizontal and vertical ending positions for the line. Used for lines that are point-to-point. The TO position is specified relative to the Position parameter values in this command.

LINEWT
Specify either one of the following keywords or the number of lineweights to be used (1 lineweight = .01 inch).

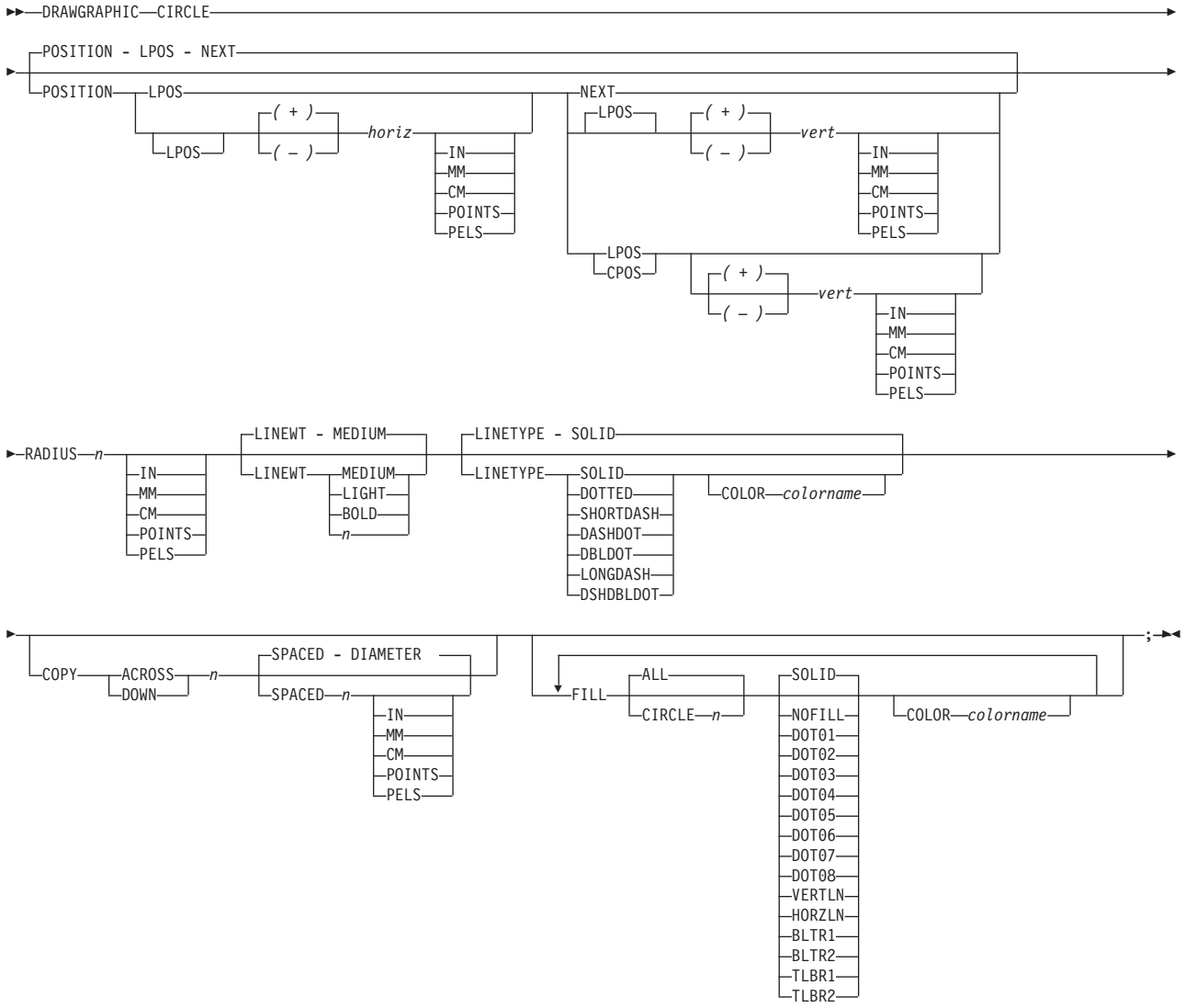
- **LIGHT** = the same as LINEWT .01 inch.
- **MEDIUM** = the same as LINEWT .02 inch.
- **BOLD** = the same as LINEWT .03 inch.

LINETYPE
Specify one of the following keywords for the line type: **SOLID, DOTTED, SHORTDASH, DASHDOT, DBLDOT (double dot), LONGDASH, or DSHDBLDOT (dash double dot)**.

COLOR
Color to be used for the line. The colorname must be either one of the pre-defined OCA keywords or the colorname from the DEFINE COLOR command.

COPY Repeat the same line at regular intervals either across or down the page. Total number of lines is one more than the value specified on this parameter.

DRAWGRAPHIC Command - Circle (Record Format)



The DRAWGRAPHIC-Circle command allows you to generate GOCA (Graphics Object Content Architecture) objects in order to draw circles on the page.

Note: GOCA circles require specific microcode in your printer.

The DRAWGRAPHIC-Circle command allows you to create a circle at either a specified radial distance from the last line printed or a specified position.

DRAWGRAPHIC can be used with the COLOR parameter and DEFINE COLOR to shade a circle with a percentage of black or other colors.

Subcommands

POSITION

Horizontal and vertical position of the center of the first circle. This position value is relative to either the Layout Position parameter or the current position.

LPOS and CPOS refer to Layout Position and Current Position respectively. If LPOS is used alone, the position is used exactly at the same position as is specified on the LAYOUT command. If it is used with a + or − value, the position moves that amount from the Layout position. The same is true for Current position except that the position is taken from the previous FIELD or DRAWGRAPHIC command.

RADIUS
Specify the circle radius. (The radius is measured from the center of the circle to the middle of the line width.)

LINEWT
Specify either one of the following keywords or the number of lineweights to be used (1 lineweight = .01 inch). Specify 0 if you want invisible borders (type and color are then ignored).

- **LIGHT** = the same as LINEWT .01 inch
- **MEDIUM** = the same as LINEWT .02 inch
- **BOLD** = the same as LINEWT .03 inch

LINETYPE
Specify one of the following keywords for the line type: **SOLID, DOTTED, SHORTDASH, DASHDOT, DBLDOT (double dot), LONGDASH, or DSHDBLDOT (dash double dot)**.

COLOR
Color to be used for the circle border. The colorname must be one of the pre-defined OCA keywords or the colorname from the DEFINE COLOR command.

COPY Repeat the same circle at regular intervals either across or down the page. Repeating ACROSS or DOWN with the DIAMETER indication means that the circles are placed to join at one point with the center positions of each being one diameter width apart. See following figures for a pictorial view of repeating circles.

Total number of circles is one more than the value specified on this parameter.

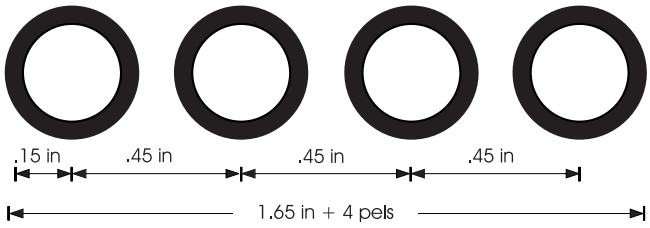


Figure 104. Repeating circles with .45 inch spacing (not to scale).

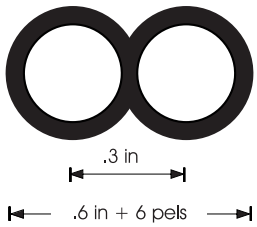


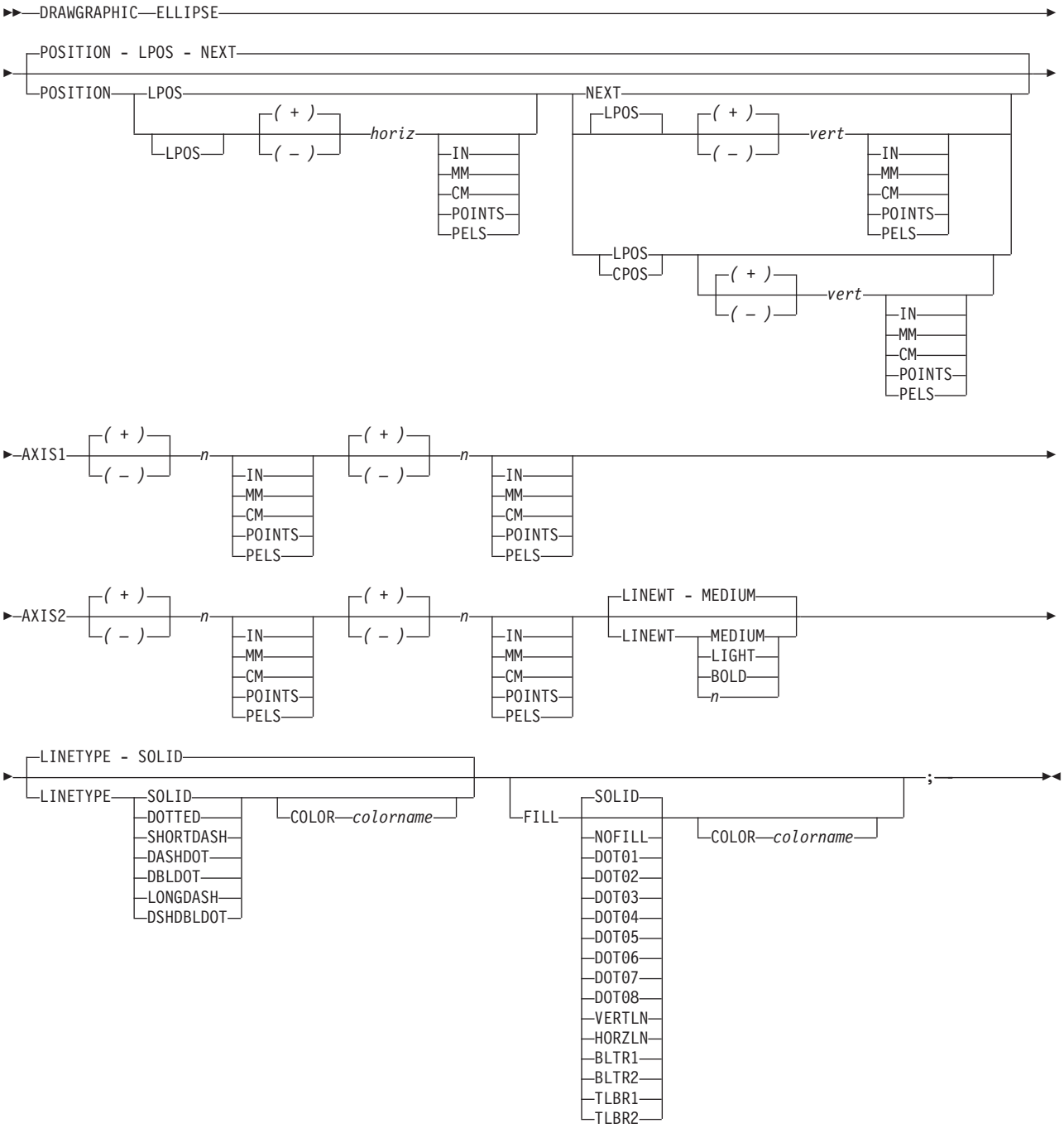
Figure 105. Repeating circles with DIAMETER spacing (not to scale).

FILL Allows the option of filling a circle with a pre-defined GOCA pattern and optionally specifying a color. Circles are numbered in the order they are defined within this command - 1,2,3,.... Filling follows the rule that the ″last fill wins″.

Using the Nofill keyword fills ALL circles with one fill pattern. Then specify Nofill on one circle to remove that circle's pattern.

For an example of the various GOCA-supported fill patterns, see Figure 119 on page 379.

DRAWGRAPHIC Command - Ellipse (Record Format)



The DRAWGRAPHIC - Ellipse command allows you to draw ellipses on the page by generating GOCA (Graphics Object Content Architecture) structure fields.

Note: GOCA lines require specific microcode in your printer.

The DRAWGRAPHIC - Ellipse command allows you to create an ellipse with a number of positions showing the major and minor axes at a specified distance from the last line printed.

The DRAWGRAPHIC can be used with the COLOR parameter and DEFINE COLOR to shade an ellipse with a percentage of black or other colors.

Subcommands

POSITION

Horizontal and vertical position of the ellipse.

LPOS and CPOS refer to Layout Position and Current Position respectively. If LPOS is used alone, the position is used exactly at the same position as is specified on the LAYOUT command. If it is used with a + or − value, the position moves that amount from the Layout position. The same is true for Current position except that the position is taken from the previous FIELD or DRAWGRAPHIC command.

AXIS1 The first pair of *n units* specifies the location of one point on the ellipse specified in relation to the POSITION parameter on this command. This location is specified as if the POSITION parameter is now at (0,0) on a coordinate system. The x and y movements are either in the positive or negative direction from the center point at (0,0). For a picture of how this is used, see Figure 106 on page 272. Point R,Q.

AXIS2 The second pair of *n units* specifies the location of second point on the ellipse specified in relation to the POSITION parameter on this command. This location is specified as if the POSITION parameter is now at (0,0) on a coordinate system. The x and y movements are either in the positive or negative direction from the center point at (0,0). For a picture of how this is used, see Figure 106 on page 272. Point P,S.

LINEWT

Specify either a keyword or the number of lineweights to be used (1 lineweight = .01 inch). Specify 0 if you want invisible borders (type and color are then ignored).

- **LIGHT** = the same as LINEWT .01 inch
- **MEDIUM** = the same as LINEWT .02 inch
- **BOLD** = the same as LINEWT .03 inch

LINETYPE

Specify one of the following keywords for the line type: **SOLID, DOTTED, SHORTDASH, DASHDOT, DBLDOT (double dot), LONGDASH, or DSHDBLDOT (dash double dot)**.

COLOR

Color to be used for the ellipse border. Specify either one of the pre-defined OCA keywords or the colorname from the DEFINE COLOR command.

FILL Allows the option of filling an ellipse with a pre-defined GOCA pattern and optionally specifying a filling color. For an example of the various GOCA-supported fill patterns, see Appendix G. ″Fill Patterns″ Figure 119 on page 379.

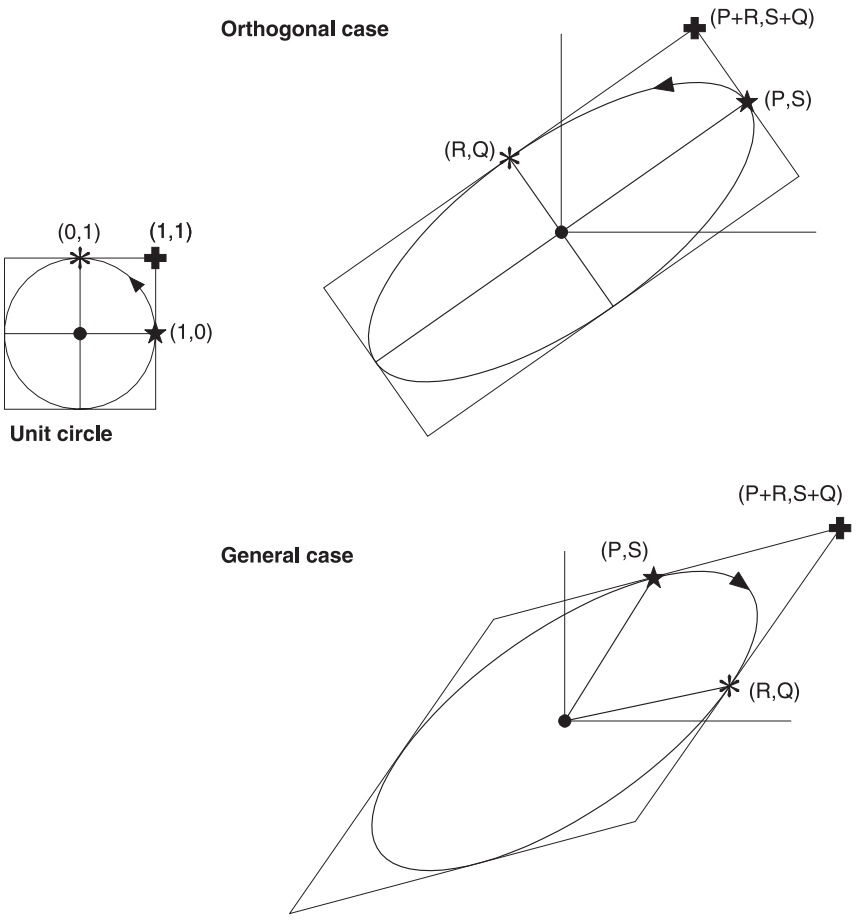
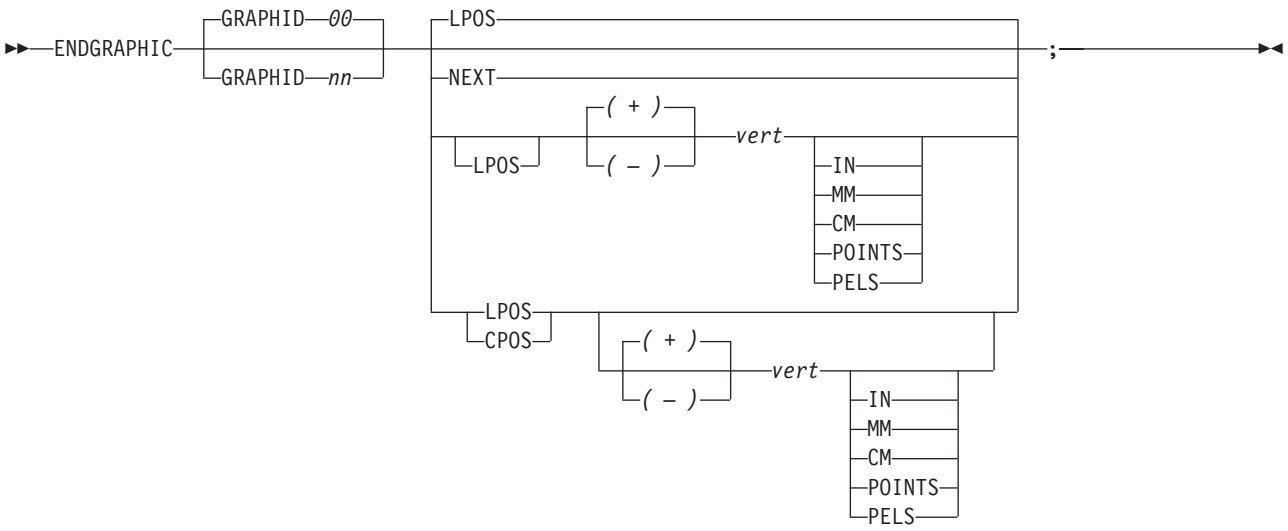


Figure 106. Ellipse parameters

The dot in the center of the ellipse shows the POSITION parameter. The asterisk shows the major axis position and star shows the minor axis position.

ENDGRAPHIC Command (Record Format)



The ENDGRAPHIC command allows you to end all active graphics with a matching graphic id. An active graphic is one that has been started but not ended, for example a vertical line or a box with no vertical size.

Subcommands

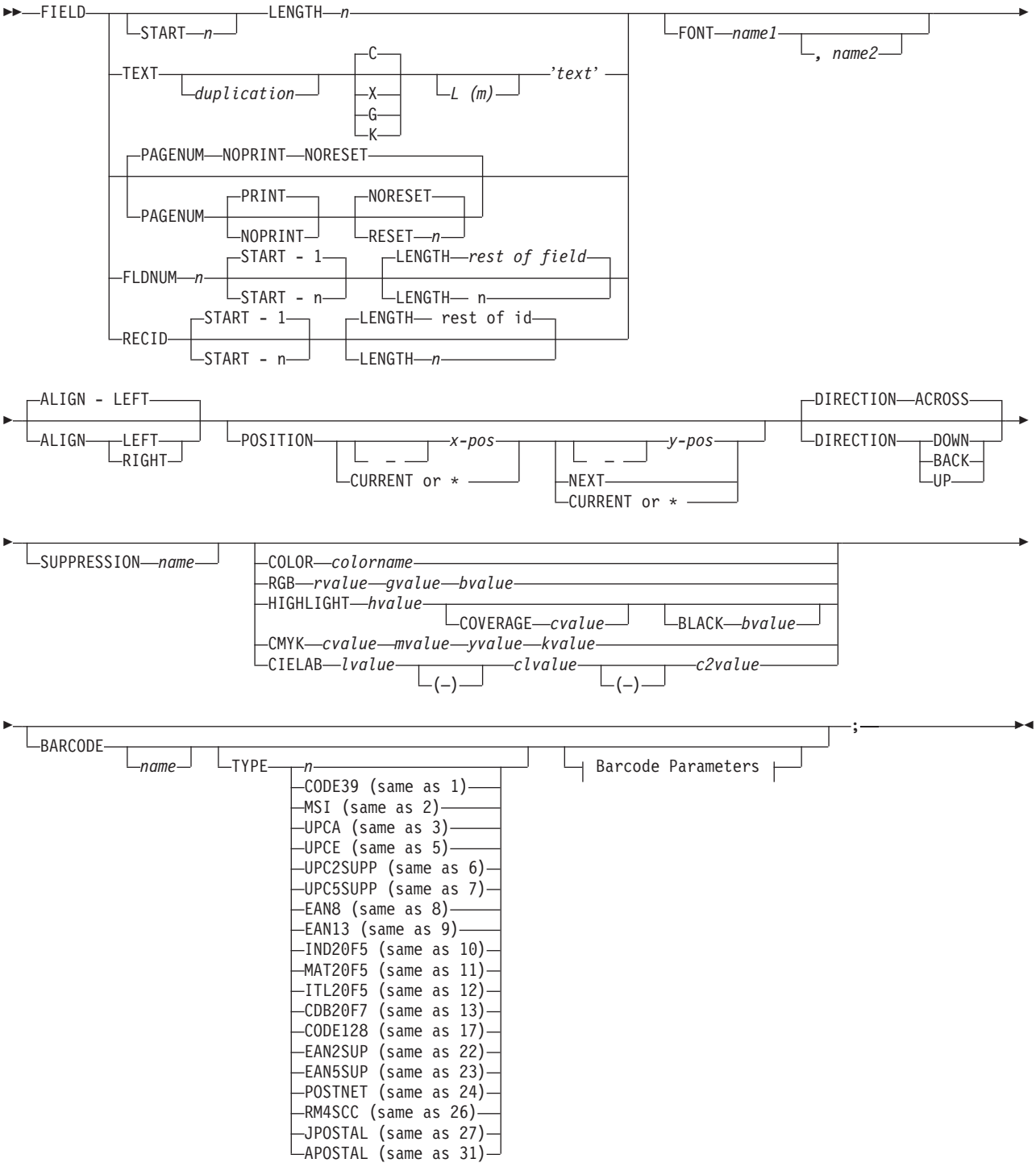
GRAPHID
Id must match one previously defined in a DRAWGRAPHIC command. If no GRAPHID is specified, all DRAWGRAPHIC commands that have no GRAPHID will be closed (i.e. GRAPHID 00).

NEXT Specifies the layout is to be positioned down (on the logical page) one line (as defined in the LINESP subcommand of the last SETUNITS command) from the previous field. The LINESP subcommand of the SETUNITS command establishes the distance from one line to the next.

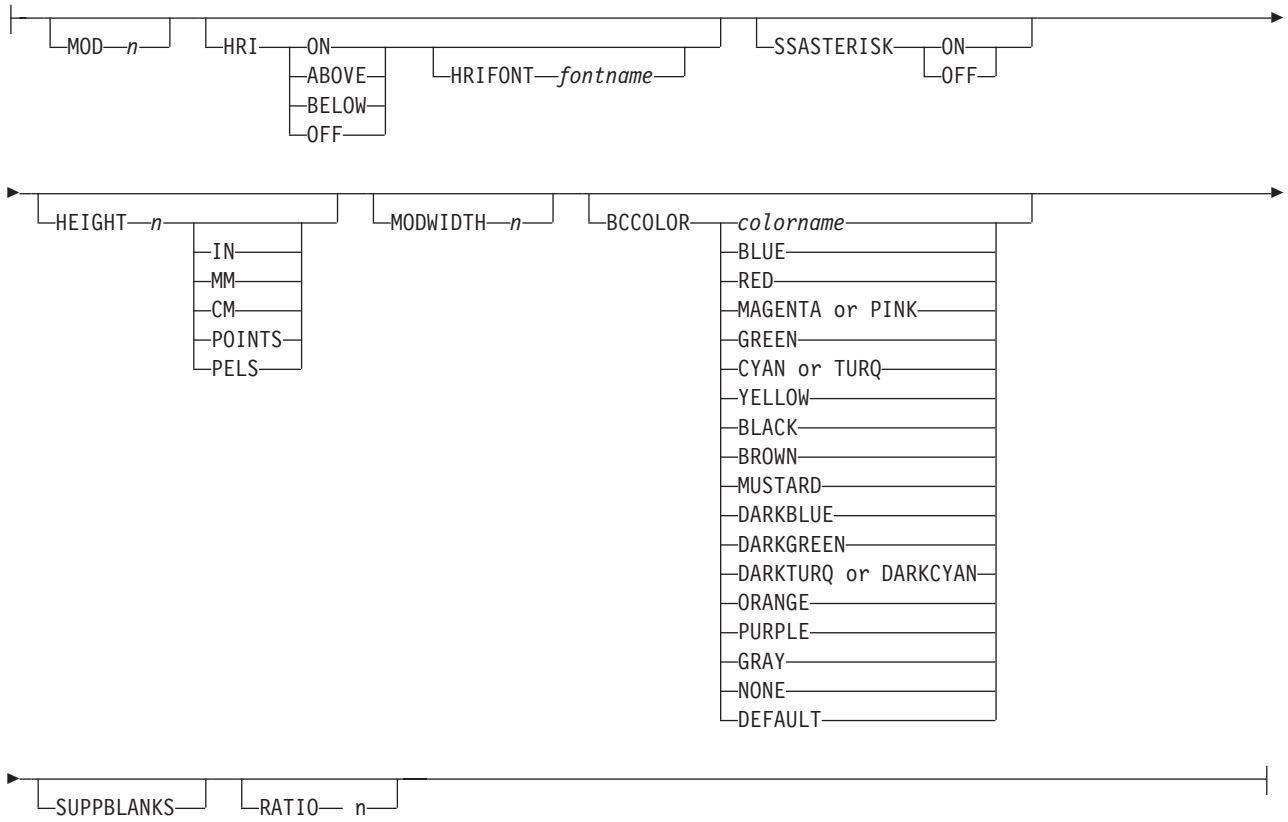
LPOS CPOS
LPOS and CPOS refer to Layout Position and Current Position respectively. If LPOS is used alone, the position is used exactly at the same position as is specified on the LAYOUT command. If it is used with a + or − value, the position moves that amount from the Layout position. The same is true for Current position except that the position is taken from the previous Field or Drawgraphic command.

vert This value is relative to the Layout position. If not specified, the graphics will be closed one line spacing from the Layout position.

FIELD Command (Record Format)



Barcode Parameters:



The FIELD command identifies a field in a data record or supplies a field of constant text, and positions where the field is on the page. More than one position on the page can be specified.

FIELD commands:
- Are subordinate to a LAYOUT command
- Must follow a LAYOUT command

The FONT, DIRECTION, and COLOR subcommands do not have fixed defaults. If any of these subcommands is omitted, the value for the omitted subcommand is obtained from corresponding subcommand in the LAYOUT command.

Subcommands

START

Specifies the starting byte in the data record for the desired field.

n Specifies the number of bytes from the first data byte in the record to be used as the starting point of the field. The first data byte position of an input record is 1.

Note: The carriage-control character is not considered data.

***** Denotes the next byte after the field identified in the previous FIELD command, excluding FIELD commands with constant TEXT.

If START * was specified in the previous FIELD command, byte 1 is assumed.

+ *n* Adds the value of n to the * byte position.

− *n* Subtracts the value of n from the * byte position.

If START is omitted and LENGTH is specified, then START * is assumed.

LENGTH
Specifies the number (*n*) of bytes to process from the data record, beginning with the position specified in START.

Once the maximum length of the field has been determined, the print server will then truncate all of the fields not containing data.

TEXT
Specifies the constant text that is to be printed in the output. A maximum of 65,535 bytes of text can be provided in one page format.

Note: This text is considered constant in that the same text is printed each time. In reference to the CONSTANT command within a form definition, this text is considered variable because the text prints only where variable data is allowed to print.

duplication = *Dn*
Specifies the number of times the text is to be repeated (use a decimal number). The maximum times the text is repeated varies depending on the size of the text. The default is **1**.

texttype = {C | X | G | K }
Specifies the type of text.

C Indicates that the text contains single-byte code characters, which includes all Roman alphabetic characters (for example, those used for English). Any valid character code can be specified, including blanks. This is the default.

X Indicates that the text contains hexadecimal codes (in groups of two hexadecimal codes) that specify values from X'00' through X'FE'.

G Indicates that the text contains double-byte code characters (for example, kanji characters).

Characters in type G text must start with shift-out (SO X'0E') and end with shift-in (SI X'0F') characters within opening and closing apostrophes (X'7D').

K Indicates that the text contains kanji numbers enclosed in apostrophes. Kanji numbers are separated by commas:

```
K'321,400'
```

Valid double-byte character set (DBCS) codes are from X'41' through X'FE' for each byte. Code X'4040' (blank) is the only exception.

Valid: X'4040', X'4041', X'41FE', X'FE41', X'FEFE'

Invalid: X'2040', X'413E', X'4100', X'7F00', X'FE3E'

L (*m***)** Specifies the length of text (use a decimal number in parentheses). When the actual length of the text is different from

m, the *m* specification is honored. That is, the text is either padded with blanks to the right or truncated.

'*text***'** Specifies the text.

Examples:

- When TEXT 2C(3)'AB' is specified, 'AB AB ' is generated. The blanks are generated because of the (3) specification.
- TEXT 2C(1)'AB' generates 'AA', truncating the Bs.

PAGENUM *n*

Although parameters are specified as optional, at least one must be specified.

Page numbers could be set at this point to start with the value specified as *n*, otherwise they will follow the specification made in the Pagedef or Pageformat command.

The POSITION parameters specified with the PAGENUM parameter will reflect the position of the page number only.

If you do not wish a page number printed, either do not use this parameter or specify NOPRINT.

The RESET parameter is only used when you wish to reset the page number that is to be used with this page.

Note: You should define a font that specifies the font type to be used for printing page numbers.

FLDNUM

This keyword should only be used if the DELIMITER field was used in the LAYOUT command. Fields cannot be counted without delimiters being specified in the database.

To allow for the identification of a part of a field which is field delimited, you can specify the starting position (from the delimiter), and optionally the length of the part of the field you want to use. The LENGTH default is to use the entire remainder of the field from the start position to the ending delimiter.

RECID

This keyword allows you to access characters in the first 10 characters of a record. This area is reserved for the record identifier, and all other field starts and lengths are calculated after this area. These starts and lengths reference only the area within the first 10 bytes.

If no record length is specified, the remaining bytes of the 10–byte field is assumed.

FONT

This keyword specifies the font to be used for printing the field contents. This parameter must have been previously defined. If this subcommand is not used, print server will assign a font.

Note: When selecting a font in AIX, you must consider that the text will be encoded in EBCDIC, not ASCII. Therefore, an EBCDIC font and code page 500 (also called International #5) must be used .

ALIGN LEFT | RIGHT
The data in this field is left or right aligned to the x position specified in the horizontal POSITION parameter.

POSITION
Specifies the starting position or the alignment position of the field in the printout.

x-pos
Do not mix *x-pos* specifications with CURRENT or * except in ACROSS fields.

– Specifies that the *x* value is negative.

x Specifies the horizontal offset for the starting print position relative to the *layout starting position*. The unit choices are IN, MM, CM, POINTS, or PELS.

The default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

The PELS measurement equals one L-unit or 1/240 of an inch, depending on whether the PELSPERINCH parameter had been specified previously.

CURRENT
Specifies that the inline offset (relative to the field's direction) is the end of the previous field. For the first field, use the LAYOUT offset. This is the default.

Note: The meaning of CURRENT differs from the meaning of the LAYOUT command parameter SAME.

***** Alternate for CURRENT.

y-pos
Do not mix *y-pos* specifications with CURRENT or * except in ACROSS fields.

– Specifies that the *y* value is negative.

y Specifies the vertical offset for the starting print position relative to the *layout starting position*. The unit choices are IN, MM, CM, POINTS, or PELS.

The default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

NEXT Specifies a field that is positioned down one line in the baseline direction (as defined in the SETUNITS command LINESP subcommand) from the previous field.

Use NEXT only in ACROSS fields.

CURRENT
Specifies that the baseline offset (relative to the field's direction) is the same as the previous field. That is, the baseline position does not change. For the first field, use the LAYOUT offset. This is the default.

***** Alternate for CURRENT.

DIRECTION
Specifies the print direction of the field, relative to the upper-left corner as you

view the logical page. If this subcommand is omitted, the direction specified in the governing LAYOUT command is used.

Note: Not all printers can print in all directions. Refer to your printer documentation for more information.

ACROSS
The page is printed with the characters added from *left to right* on the page, and the lines are added from the top to the bottom.

DOWN
The page is printed with the characters added from *top to bottom* on the page, and the lines added are from the right to the left.

BACK The page is printed with the characters added from *right to left* on the page, and the lines are added from the bottom to the top.

UP The page is printed with the characters added from *bottom to top* on the page, and the lines are added from the left to the right.

SUPPRESSION
Specifies that this field can be suppressed.

name Specifies the name of a field to be suppressed.

Printing of this field is suppressed if this name is identified by a SUPPRESSION command within the form definition.

The same name can be used in one or more fields to suppress these fields as a group.

COLOR
Specifies an OCA or defined color for the text of this field. This subcommand is recognized only by printers that support multiple-color printing. Refer to your printer publication for more information.

colorname

Values for *colorname* are NONE, DEFAULT, BLACK, BLUE, BROWN, GREEN, PINK, RED, TURQ (turquoise), YELLOW, ORANGE, PURPLE, MUSTARD, GRAY, DARKBLUE, DARKGREEN, or DARKTURQ (dark turquoise). The color choices depend on the printer. NONE is the color of the medium. DEFAULT is the printer default color.

Note:

In some printer publications, the color turquoise (TURQ) is called “cyan”, and the color pink (PINK) is called “magenta”.

PPFA supports the following synonyms:

- CYAN for TURQ
- DARKCYAN for DARKTURQ
- DBLUE for DARKBLUE
- DCYAN for DARKTURQ
- DGREEN for DARKGREEN
- DTURQ for DARKTURQ
- MAGENTA for PINK

Color Model

Specifies the color of print for this field supported in MO:DCA for the Red/Green/Blue color model (RGB), the highlight color space, the Cyan/Magenta/Yellow/Black color model (CMYK), and the CIELAB color model.

RGB *rvalue gvalue bvalue*

Three **RGB** integer values are used. The first (rvalue) represents a value for red, the second (gvalue) represents a value for green, and the third (bvalue) represents a value for blue. Each of the three integer values may be specified as a percentage from 0 to 100.

Note: An RGB specification of 0/0/0 is black. An RGB specification of 100/100/100 is white. Any other value is a color somewhere between black and white, depending on the output device.

HIGHLIGHT *hvalue* **COVERAGE** *cvalue* **BLACK** *bvalue*

Indicates the highlight color model. Highlight colors are device dependent, and can be specified for the IBM InfoPrint Hi-Lite Color Printer Model 4005-HCI.

You can use an integer within the range of 0 to 65535 for the *hvalue*.

Note: An *hvalue* of 0 indicates that there is no default value defined; therefore, the default color of the presentation device is used.

COVERAGE indicates the amount of coverage of the highlight color to be used. You can use an integer within the range of 0 to 100 for the *cvalue*. If less than 100 percent is specified, the remaining coverage is achieved with the color of the medium.

Note: Fractional values are ignored. If **COVERAGE** is not specified, a value of 100 is used as a default.

BLACK indicates the percentage of black to be added to the highlight color. You can use an integer within the range of 0 to 100 for the *bvalue*. The amount of black shading applied depends on the **COVERAGE** percentage, which is applied first. If less than 100 percent is specified, the remaining coverage is achieved with black.

Note: If **BLACK** is not specified, a value of 0 is used as a default.

See “Color on the IBM InfoPrint HiLite Color Post Processor” on page 46 for more information.

CMYK *cvalue mvalue yvalue kvalue*

Defines the cyan/magenta/yellow/black color model. *Cvalue* specifies the cyan value. *Mvalue* specifies the magenta value. *Yvalue* specifies the yellow value. *Kvalue* specifies the black value. You can use an integer percentage within the range of 0 to 100 for any of the CMYK values.

CIELAB *Lvalue* **(−)***c1value* **(−)***c2value*
Defines the CIELAB model. Use a range of 0.00 to 100.00 with *Lvalue* to specify the luminance value. Use signed integers from −127 to 127 with *c1value* and *c2value* to specify the chrominance differences.

Lvalue, c1value, c2value must be specified in this order. There are no defaults for the subvalues.

Note: Do not specify both an OCA color with the **COLOR** subparameter and an extended color model on the same **FIELD** or **LAYOUT** command. The output is device dependent and may not be what you expect.

```
FIELD START 1 LENGTH 5
    COLOR BLUE ;
FIELD START 1  LENGTH 1
    RGB 10  75  30 ;
FIELD START 1  LENGTH 2
    cmyk 80 10 10 10 ;
FIELD START 1  LENGTH 2
    CIELAB 80 100 20 ;
FIELD START 1  LENGTH 2
    highlight 5 ;
FIELD START 1  LENGTH 2
    highlight 300 COVERAGE 50 BLACK 30 ;
```

Figure 107. Color Model Usage Using the FIELD Command

BARCODE parameters
Specifies a bar code in a page definition.

The bar code name can be 1-8 characters long. Refer to your printer documentation for additional information about bar code support and the SUPPBLANKS subcommand. Ensure that the bar code fits on the page or you will get errors at print time.

Please read your printer hardware documentation before using bar codes. The documentation will indicate which bar code types, modifiers, modwidth, element heights, and ratio values are valid for the printer.

PPFA does minimal verification of the bar code values. If you use the HEIGHT, MODWIDTH, RATIO and BCCOLOR parameters, ensure that the values you specify are valid for your printer.

For printer optimization, specify **BARCODE** *name options* in the first instance of a specific type of bar code. If this type is used again, position it as usual with **START**, **LENGTH**, and **POSITION**, but specify the barcode information using only **BARCODE** *same-name-as-previously*. The BARCODE subcommand is recognized only by printers that support BCOCA bar code printing; refer to *Advanced Function Presentation: Printer Information* (G544-3290) for more information.

For more information about bar codes, see “Appendix D. More About Bar Code Parameters” on page 349 and refer to *Data Stream and Object Architectures: Bar Code Object Content Architecture Reference*, S544-3766.

TYPE { *n* | *type-name* }

TYPE Specifies the type of bar code symbol to be generated.

Note: If a type indicates "(same as *n*)", you may substitute the number given for the character name.

The following bar code types are supported by PPFA:

type-name
Specifies a specific bar code name to be included in a page definition.

CODE39 (same as 1)
Specifies a bar code type of Code 39 (3-of-9 code), Automatic Identification Manufacturers Uniform Symbol Specification 39.

MSI (same as 2)
Specifies a bar code type of modified Plessey code.

UPCA (same as 3)
Specifies a bar code type of Universal Product Code (United States) and the Canadian Grocery Product Code, Version A

UPCE (same as 5)
Specifies a bar code type of Universal Product Code (United States) and the Canadian Grocery Product Code, Version E

UPC2SUPP (same as 6)
Specifies a bar code type of Universal Product Code (United States) two-digit Supplemental (periodicals).

UPC5SUPP (same as 7)
Specifies a bar code type of Universal Product Code (United States) five-digit Supplemental (paperbacks).

EAN8 (same as 8)
Specifies a bar code type of European Article Numbering 8 (includes Japanese Article Numbering-short).

EAN13 (same as 9)
Specifies a bar code type of European Article Numbering 13 (includes Japanese Article Numbering-standard).

IND2OF5 (same as 10)
Specifies a bar code type of Industrial 2-of-5.

MAT2OF5 (same as 11)
Specifies a bar code type of Matrix 2-of-5.

ITL2OF5 (same as 12)
Specifies a bar code type of Interleaved 2-of-5, Automatic Identification Manufacturers Uniform Symbol Specification-I 2/5.

CDB2OF7 (same as 13)
Specifies a bar code type of Codabar, 2-of-7, Automatic Identification Manufacturers Uniform Symbol Specification-Codabar.

CODE128 (same as 17)
Specifies a bar code type of Code 128, Automatic Identification Manufacturers Uniform Symbol Specification-128.

Note:

There is a subset of CODE128 called EAN128. These EAN128 bar codes can be produced with PPFA by specifying CODE128 for the bar code type in the pagedef and including the ″extra″ parts of the bar code in the data. The UCC-128 bar code format is:

```
startcode FNC1 ai nnnnnnnnnnnnnnnnn m c stopchar
```

The string of n's represents the bar code data. The start code, stop character, and 'c' value are generated by the printer microcode for BCOCA bar codes. The FNC1 is a hexidecimal 8F character. The ″ai″ is an application identifier and needs to be defined for use by each EAN128 application. The ″m″ is a modulo 10 check digit that must be calculated by the application and included in the bar code data.

Not all IBM printers will generate the EAN128 bar codes, thus you may need to verify that the bar code produced in this manner is readable by your bar code scanner.

For more information about the EAN128 bar codes, visit the Uniform Code Council WEB site at http://www.UC-council.org.

EAN2SUP (same as 22)
Specifies a bar code type of European Article Numbering, Two-digit Supplemental.

EAN5SUB (same as 23)
Specifies a bar code type of European Article Numbering, Five-digit Supplemental.

POSTNET (same as 24)
Specifies a bar code type of POSTal Numeric Encoding Technique (United States Postal Service), and defines specific values for the BSD module width, element height, height multiplier, and wide-to-narrow ratio fields.

RM4SCC (same as 26)
Specifies a 4-state customer code defined by the Royal Mail Postal Service of England for bar coding postal code information.

JPOSTAL (same as 27)
A complete Japan Postal Bar Code symbol consists of a set of distinct bars and spaces for each character followed by a modulo 19 checksum character and enclosed by a unique start character, stop character and quiet zones.

APOSTAL (same as 31)
Specifies the barcode type as defined by the Australian Postal Service.

For more information about bar codes, see "Appendix D. More About Bar Code Parameters" on page 349 and refer to *Data Stream and Object Architectures: Bar Code Object Content Architecture Reference*, S544-3766.

MOD Specifies additional processing information about the bar code symbol to be generated (for example, MOD specifies whether a check-digit [9] should be generated for the bar code symbol).

n The meaning of *n* differs between the types. For more information, see Table 18 on page 358.

If **MOD** is not specified, the MOD value defaults as follows, depending on the bar code type specified:

TYPE	MOD	TYPE	MOD
1	1	11	1
2	1	12	1
3	0	13	1
5	0	17	2
6	0	22	0
7	0	23	0
8	0	24	0
9	0	26	0
10	1	27	0
		31	1

HRI Specifies whether the human-readable interpretation (text characters) will be generated and placed above or below the bar code symbol.

ON Specifies that HRI should be generated at the default location for the barcode type.

ABOVE
Specifies that HRI should be placed above the bar code symbol.

BELOW
Specifies that HRI should be placed below the bar code symbol.

OFF Specifies that HRI should not be generated.

Note: If **HRI** is requested, and HRI font isn't, the printer default font is used to render the HRI, instead of the font specified on the FIELD FONT subcommand.

HRIFONT *fontname*
Specifies the font to be used in printing the HRI for the barcode.

SSASTERISK
Specifies whether an asterisk is to be generated as the HRI for bar code start and stop characters.

ON Specifies that start and stop characters should be generated in the HRI.

OFF Specifies that start and stop characters should not be generated in the HRI.

HEIGHT
Specifies the height of bar code element. For UPC and EAN bar codes, the total height includes the bar code and the HRI characters.

9. Check digits are a method of verifying data integrity during the bar code reading process.

If **HEIGHT** is not specified, the printer default height is used.

Note: HEIGHT is ignored by bar code types that explicitly specify the element heights (for example, **POSTNET** or **RM4SCC**).

n Specifies the height of the bar code.

unit Specifies a unit of measurement for the *HEIGHT* parameter. The choices are IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

MODWIDTH

Specifies the width of the smallest defined bar code element, using mils (thousandths of an inch). For bar code types that explicitly specify the module width (for example, **POSTNET** and **RM4SCC**), this field is ignored. The range of values allowed is 1-254. If **MODWIDTH** is not specified, the printer default MODWIDTH is used.

n Specifies the width of each module, using thousandths of an inch (1/1000) as the unit of measurement.

BCCOLOR

Specifies an OCA or defined OCA color to be used in printing the barcode and its HRI.

colorname

Specifies the name of a defined color.

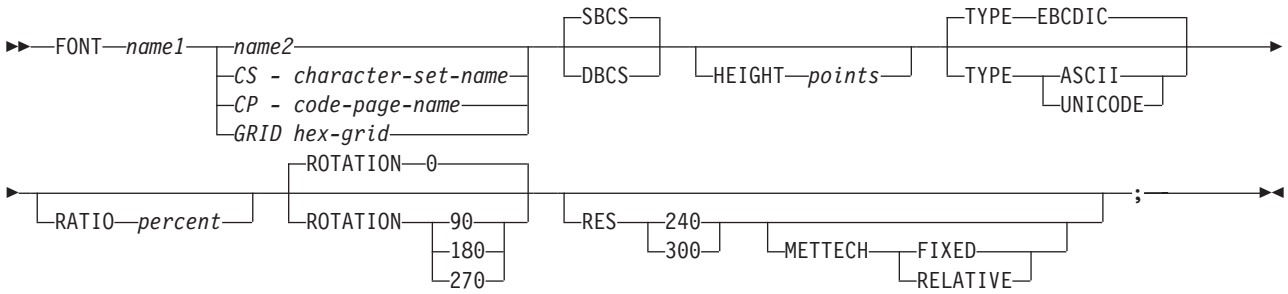
SUPPBLANKS

Suppress the trailing blanks in the data field used to generate the barcode.

RATIO

Specifies the ratio between the width of the wide and the narrow bar code elements. The range of values allowed is 100-500, but you must specify a value appropriate for your printer and bar code type or you will get errors at print time.

FONT Command (Record Format)



The FONT command is used to identify the fonts that are to be specified in the FIELD command. A maximum of 127 font names for each page definition can be identified.

Note: Naming a font with the FONT command does not, by itself, affect your output. You must specify the font in a FIELD or command for the font to become effective. If you do not name a font, the default font will be used.

FONT commands immediately follow the PAGEDEF command. A separate FONT command is required:

For each font used within a page definition
For each rotation of the same font

Note:

FONT *name1* **{** *name2* **| CS** *character-set-name* **CP** *code-page-name* **| GRID** *hex-grid* **}**

Identifies the fonts to be specified in the FIELD command.

name1 Specifies an alphanumeric name of 1 to 16 characters (local name) of the font to be used in this page definition. The name must conform to the token rules and must be unique within this page definition.

name1 is used in the FIELD, or TRCREF commands of a page definition.

name1 is optional if *name2* is specified.

name2 Specifies an alphanumeric name of 1 to 6 characters (user-access name) of the coded font to be used in this page definition. Specify this without the Xn prefix.

character-set-name
Specifies an alphanumeric name of 1 to 6 characters of the character set to be used in this page definition. Specify this without the Cn prefix.

code-page-name
Specifies an alphanumeric name of 1 to 6 characters of the code page to be used in this page definition.

hex-grid
Specifies the 16-character hexadecimal GRID.

Subcommands

SBCS or DBCS
Specifies single-byte or double-byte fonts.

SBCS Specifies that the font is a single-byte character set. This is the default.

DBCS Specifies that the font is a double-byte character set.

HEIGHT
Specifies the height of the outline font.

points Each point is equal to 1/72 of one inch.

TYPE The TYPE subcommand indicates the type of Font being used.

EBCDIC
This parameter is normally used for fonts on OS390–based systems. This is the default.

ASCII This parameter is normally used for fonts on workstation–based systems.

UNICODE
This parameter is used with Unicode type fonts.

RATIO
Specifies the ratio of scaling the width relative to the height in an outline font.

percent Represents the percent of the "normal" width of the character that will be printed. For example, specifying RATIO 50 yields a font with characters half as wide as normal, and specifying RATIO 200 yields a font with characters twice as wide (200% as wide) as normal. If RATIO is specified, you must also specify the HEIGHT.

ROTATION
Specifies the rotation of characters in degrees. The specified value is relative to the inline direction of a layout or field. Valid rotations are 0°, 90°, 180°, or 270°; **0°** is the default.

RESOLUTION
Specifies the resolution and metric technology on a font.

RES or **RESOLUTION**

```
The raster-pattern resolution units in pels per inch
     240            240 pels per inch
     300            300 pels per inch
```

METTECH or **METRICTECHNOLOGY**

```
The metric technology used for this raster font
     FIXED          Fixed-metric technology
     RELATIVE       Relative-metric technology
```

Notes:

1. The resolution and metrictechnology subcommands allow rigorous font specifications for use with font fidelity. See the font fidelity subcommand FONTFID on the FORMDEF command.
2. For a description of metric technologies, refer to:
   - *Intelligent Printer Data Stream Reference*, S544-3417
   - *Font Object Content Architecture Reference*, S544-3285

3. RESOLUTION can be abbreviated as RES; METRICTECHNOLOGY can be abbreviated as METTECH.

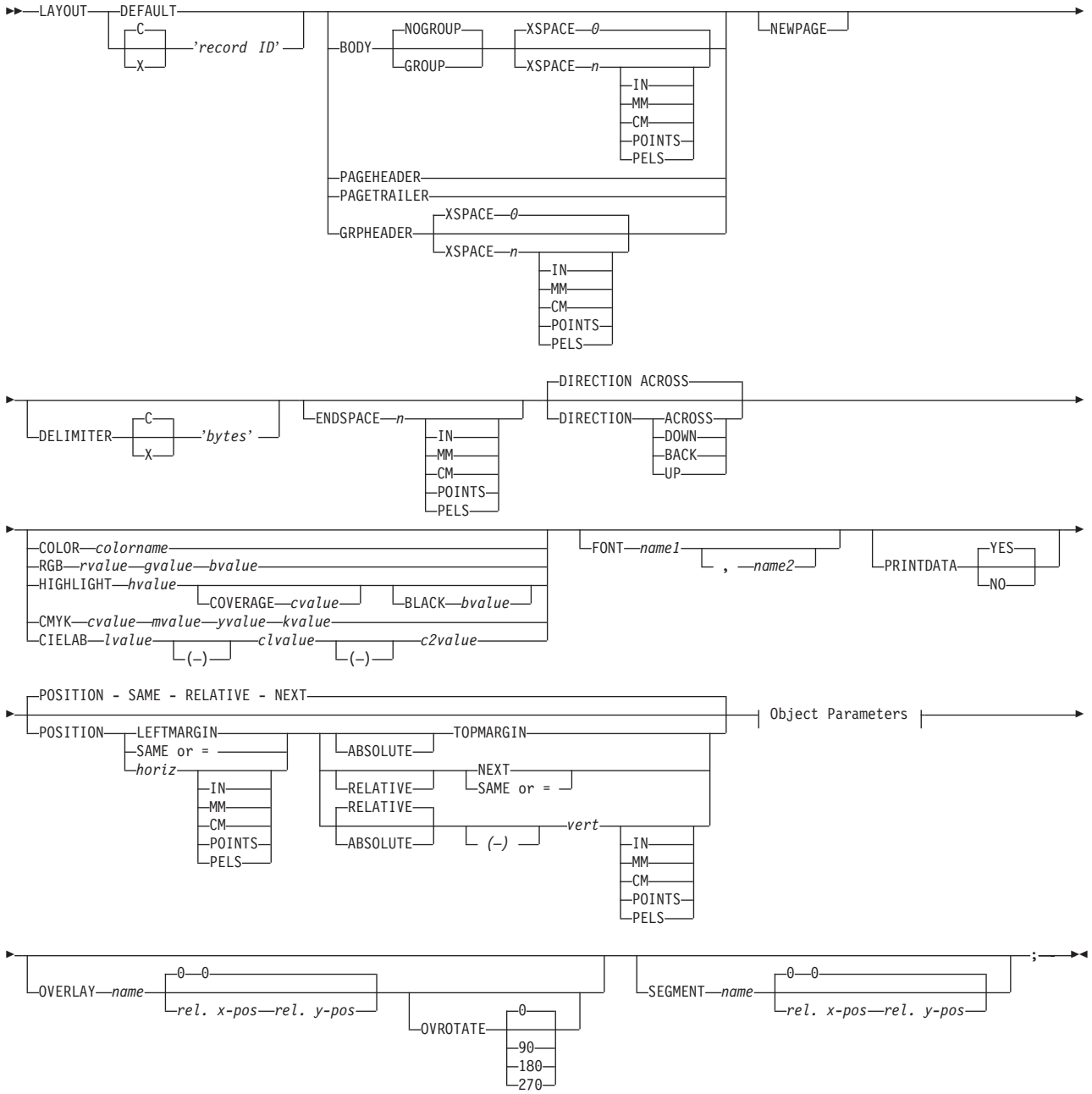
```
FORMDEF  xmp01
  FONTFID  YES ;

  PAGEDEF  xmp01    replace  yes ;
    FONT  xx2  res  240  mettech  fixed ;
    LAYOUT  font  xx2 ;
```

Figure 108. Example of PPFA Support for Font Fidelity

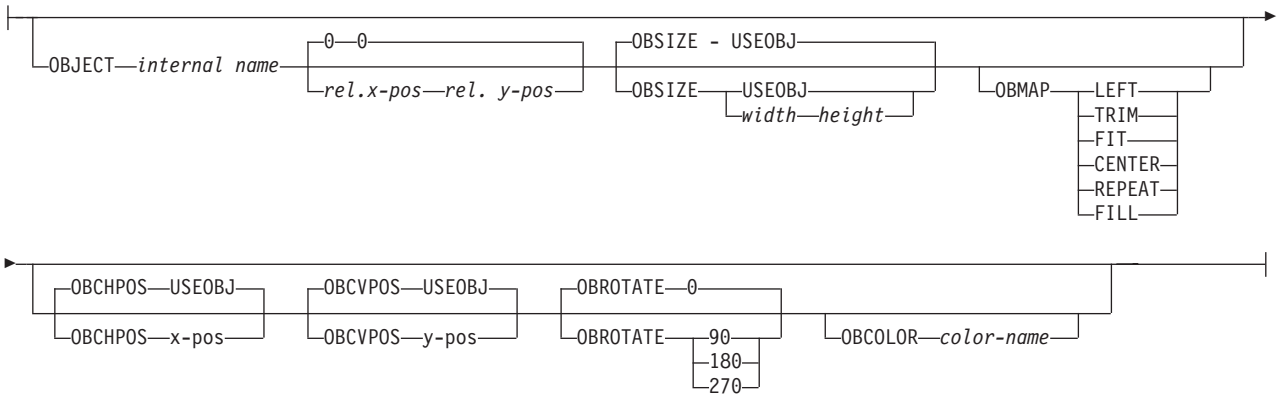
In the example above, the form definition xmp01 specifies font fidelity and the page definition specifies a font that has 240 pels per inch resolution and fixed-metric technology. If a font with exactly those characteristics is not accessible by the printer, an error occurs and processing stops.

LAYOUT Command (Record Format)



LAYOUT (Record Format)

Object Parameters:



The LAYOUT command will be used instead of the PRINTLINE command. User must specify one of these commands and then be consistent throughout the rest of the page definition.

Subcommands

DEFAULT
This keyword is used only when the layout type is either PAGEHEADER or PAGETRAILER, and no name is needed.

'record ID'
The record ID is entered within quotes and must match the record id within the data record exactly byte for byte. PPFA will do no translation on these characters. Hex characters must be entered in uppercase within the quotes. The name will be padded with blanks if the field contains less than 10 characters.

″C″ stands for character input and ″X″ stands for hex bytes. The default is ″C″.

BODY The BODY layout type is used for the majority of data in the user's database, normally printed line by line. This is the default.

GROUP
The GROUP parameter indicates that the existing group header should be saved and used for subsequent pages. If this parameter is not set when processing starts on a Body layout, the active group header record is discarded and not reprinted on subsequent pages.

PAGEHEADER
This layout type specifies a header that is to be printed on each new page. The baseline position of this layout is normally in the top margin, but can be anywhere on a logical page. If RELATIVE is specified, the position is considered to be relative to the page origin. Usually contains customer's name, address, account number, etc. Only one default Page Header layout can be specified in a Pageformat and no input record data can be specified in a default layout.

GRPHEADER
This layout type specifies a header that is to be printed at the beginning of a group of data. If a logical page eject occurs before the group of data ends, the header is printed after the top margin on each new page until the

group ends. The baseline position of this layout can be specified as RELATIVE. It may include column headings.

XSPACE

XSPACE indicates the amount of extra space from the position of the layout to the bottom of the group header area. This allows the user to identify the amount of eXtra space in excess of one text line being used by the header so that the baseline will move down and the following group data will not be placed on top of the header area. This space will not be calculated by PPFA and must be explicitly defined by the user. See example below (shaded space shows group header area):

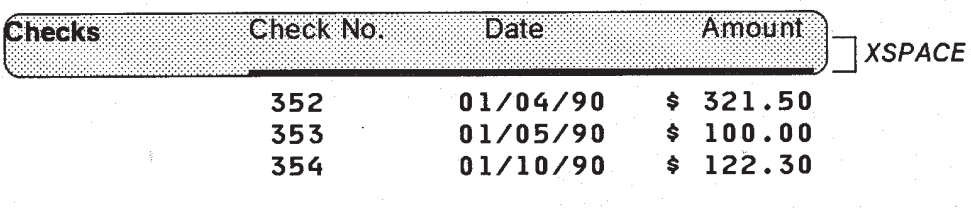


Figure 109. Example Showing the Use of XSPACE.

PAGETRAILER

This layout type specifies a trailer that is to be printed on each new page. The baseline position of this layout is normally in the bottom margin, but can be located anywhere on a logical page and can be specified as RELATIVE. Only one default Page Trailer layout can be specified in a Pageformat and no input record data is processed with a default layout. It may contain the name of the form or a footnote.

NEWPAGE

This parameter indicates that a new page should be started with this layout name. If this is a header or trailer layout, the print position is moved to the start of a new page before this header or trailer becomes the active header or trailer.

DELIMITER

The delimiter is a one or two byte code specified in either character or hex indicates a delimiting character within the customer's database and is used to separate fields. PPFA will do no translation on these characters. Hex characters must be entered in uppercase within the quotation marks.

ENDSPACE

If the remaining body space is less than the value specified, ENDSPACE will cause a logical page eject to be executed. This can be used, for example, on a Group Header layout to ensure that a group header does not print at the end of a page without the first data record of the group. ENDSPACE does not include the space within the bottom margin (specified on the Pagedef or Pageformat command). This indicator is ignored on a Page Header or Page Trailer layout.

DIRECTION

Specifies the print direction of the line relative to the upper-left corner as you view the logical page. Not all printers can print in all print directions. For more information about your printer, refer to your printer documentation.

If DIRECTION is not specified, the direction specified in the PAGEFORMAT command is used. Observe that this direction is additive to the direction specified in the PAGEFORMAT command. See "PAGEFORMAT Command (Record Format)" on page 308.

ACROSS
The layout direction is rotated 0 degrees relative to the direction specified in the PAGEFORMAT (the layouts are oriented in the same direction as the page).

DOWN
The layout direction is rotated 90 degrees relative to the direction specified in the PAGEFORMAT.

BACK The layout direction is rotated 180 degrees relative to the direction specified in the PAGEFORMAT.

UP The layout direction is rotated 270 degrees relative to the direction specified in the PAGEFORMAT.

COLOR
Specifies an OCA or defined color for the text of this field. This subcommand is recognized only by printers that support multiple-color printing. Refer to your printer publication for information about the colors that can printed.

colorname

Values for *colorname* are NONE, DEFAULT, BLACK, BLUE, BROWN, GREEN, PINK, RED, TURQ (turquoise), YELLOW, ORANGE, PURPLE, MUSTARD, GRAY, DARKBLUE, DARKGREEN, or DARKTURQ (dark turquoise). The color choices depend on the printer.

If you do not enter one of these colors, the default color for that printer is used. NONE is the color of the medium. DEFAULT is the printer default color.

Note: In some printer manuals, the color turquoise (TURQ) is called "cyan", and the color pink (PINK) is called "magenta".

PPFA supports the following synonyms:

- CYAN for TURQ
- DARKCYAN for DARKTURQ
- DBLUE for DARKBLUE
- DCYAN for DARKTURQ
- DGREEN for DARKGREEN
- DTURQ for DARKTURQ
- MAGENTA for PINK

FONT Defines the font to be used for the layout.

name1 Specifies the name of a font used to print the data. This font must have been defined in a previous FONT command in this page definition.

If Shift-Out, Shift-In (SOSI) processing is used, *name1* must be the single-byte font.

name2 Specify only when using Shift-Out, Shift-In (SOSI) processing to dynamically switch between a single-byte font and a double-byte font within the layout. *name2* must be the double-byte font.

Notes:

1. If this subcommand is not specified in the print data, print server will use the font indicated. Otherwise, print server will select a default font.
2. When selecting a font in AIX, you should consider that the text will be selected in EBCDIC, not ASCII. Therefore, an EBCDIC font and code page 500 (also called International #5) should be used for *name1*.

PRINTDATA

Specifies whether the line of data associated with the current LAYOUT should be printed. The PRINTDATA subcommand is useful when the data stream is interspersed with lines of comments, blank lines, or lines without data that are not meant to be printed.

YES Specifies the data for the current LAYOUT will be printed. YES is the default.

NO Specifies the data for the current LAYOUT will not be printed.

POSITION

This is for use in positioning Field, Drawgraphic, & Endgraphic text and graphics. If Relative is specified or POSITION is not specified, the baseline of the Position is relative to the previous Layout position.

1. For PageHeader RCD: The baseline position can be anywhere on a logical page, but cannot be specified as Relative.
2. For PageTrailer, GroupHeader and Body RCDs: The baseline position can be anywhere on a logical page and can be specified as Relative.

Specifies the starting position of the layout in the printout.

horizontal position

x-pos Specifies the horizontal offset from the left side of the logical page. The value is a number with up to three decimal places. The valid options for *x-pos* are described in the SETUNITS command for the horizontal value.

MARGIN

Specifies this line starts at the position specified as the horizontal (*x*) value in the previous MARGIN subcommand within this page definition.

SAME Specifies this line starts at the same horizontal offset position as the previous LAYOUT. If applied to the first LAYOUT of a logical page, the horizontal position is 0, which is the default.

= Alternate for SAME.

RELATIVE

Specifies that the following vertical position value is to be processed as a relative value. The LAYOUT is positioned relative to the last LAYOUT placed on the page.

Note: If both TOP and RELATIVE are requested for the *y-pos* value, the RELATIVE request is ignored.

When using RELATIVE positioning, PPFA does not flag off-the-page conditions for the position of a LAYOUT or for any overlays, segments or objects placed relative to that LAYOUT. LAYOUTs that fall outside the bounds of the logical page are flagged by print server at run time.

When specifying RELATIVE, use the minus sign to indicate any negative values for the LAYOUT vertical position; you may use the plus sign to indicate positive values. If no sign is used, a positive value is assumed.

The DIRECTION for a relative LAYOUT must be ACROSS. Fields associated with a relative LAYOUT must have the same DIRECTION as the LAYOUT and must match the pageformat DIRECTION.

If RELATIVE is specified with "SAME" or "=" as the "y" value, the relative value in the LAYOUT will be +0.

Relative positioning is allowed on a LAYOUT command only if the LAYOUT and all its associated FIELD commands are formatted to print in the same direction as the PAGEFORMAT. That is, the DIRECTION parameter in the LAYOUT and any associated FIELD commands must specify (or default to) ACROSS. The DIRECTION in the PAGEFORMAT or PAGEDEF command may be any allowable value: ACROSS, DOWN, BACK, or UP.

vertical position

y-pos Specifies the vertical offset from the top side of the logical page. The value options for *y-pos* are described in the SETUNITS command for the vertical value.

TOP Specifies that the LAYOUT is placed in the position specified as the vertical (*y*) value in the TOPMARGIN subcommand within this page definition.

NEXT Specifies the layout is to be positioned down (on the logical page) one line (as defined in the LINESP subcommand of the last SETUNITS command) from the previous field. The LINESP subcommand of the SETUNITS command establishes the distance from one line to the next.

When NEXT is specified for the first LAYOUT of a logical page, the starting position of the line is one line down from the top of the logical page, as defined by the TOPMARGIN subcommand.

Note: The "down" direction is determined by the direction of the logical page (as specified in the page format), not the LAYOUT direction. NEXT is, therefore, mainly useful in ACROSS LAYOUTs.

Note: For additional details on this area, please refer to the URL:
http://www.ibm.com/printers/R5PSC.NSF/Web/ppfaupdt

SAME Specifies this LAYOUT starts at the same vertical position as the previous LAYOUT.

= Alternate for SAME.

OBJECT *parameters*

Specifies the name of an object that is to be positioned and oriented relative to the location specified in the LAYOUT command in which the OBJECT subcommand was named. The OBJECT, as identified by the internal-name parameter, must have been defined by an OBJECT command. You may place multiple objects on the same LAYOUT command and you may place the same object multiple times. Each placement must have its own set of placement parameters, as follows:

internal-name

Specifies the name of an object that is up to 16 alphanumeric characters in length. The *internal-name* is used to match the LAYOUT OBJECT subcommand to its definition from the OBJECT command. An object must be defined with this internal name by the OBJECT command.

relative-xpos relative-ypos

Specifies the number of units (inches, mm, and so on) that are added to the position of the current LAYOUT to position the top-left corner of the object. The values for the horizontal and vertical positioning are limited by the type of printer used and the L-units specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

Each position specification can be a positive or negative number with up to three decimal places. The units specified can be one of the following: IN, MM, CM, POINTS, or PELS.

OBSIZE

Specifies the size of the object placement area. When no OBSIZE is specified, the default is the size specified in the object. If no size is specified in the object, the size of the page is used. The page width is as specified on the PAGEDEF or PAGEFORMAT commands, or it defaults to 8.3 inches by 10.8 inches.

wd Specifies the width of an object placement area as a number with up to three decimal places. The allowable width may vary with the type of printer used and the L-units specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

hg Specifies the height of the object placement area as a number with up to three decimal places. The allowable height may vary with the type of printer used and the L-units specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

unit Specifies a unit of measurement for the width parameter. The choices are: IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

USEOBJ

Specifies that the size measurements specified in the object are to

be used. If no size is specified in the object, the size of the page is used, which is the length and width as specified on the PAGEDEF or PAGEFORMAT commands, or it defaults to 8.3 inches by 10.8 inches.

OBMAP

Specifies mapping options. The OBMAP parameter defines the mapping of the object to the object placement area. If OBMAP is not coded, the mapping option within the object is used. If the object does not contain a mapping option, then print server sets it to the created default for the container type.

Each object type (OBTYPE on the OBJECT command) dictates the allowable mapping options for that type. When it can, PPFA issues a message when these rules are violated. However, in the case of an object type of page segment (OBTYPE=PSEG), PPFA does not know what types of objects are contained in it; therefore, PPFA cannot enforce the restrictions. See "OBJECT Command (Traditional)" on page 221 for a description of the restrictions.

LEFT Specifies that the object is positioned at the upper, left-hand corner of the object placement area, as defined or defaulted by the *relative-xpos*, *relative-ypos*, OBCHPOS, and OBCVPOS parameters. Any portion of the object that falls outside the object placement area as defined by the OBSIZE parameter is not trimmed and could cause an exception condition by the presentation system.

TRIM Specifies position and trim. The object is positioned at the upper, left-hand corner of the object placement area, as defined or defaulted by the *relative-xpos*, *relative-ypos*, OBCHPOS, and OBCVPOS parameters. Any portion of the object that falls outside the object placement area as defined by the OBSIZE parameter is trimmed.

FIT Specifies scale to fit; this is the default value if the OBMAP parameter is not coded. The object is to be scaled to fit within the object placement area, as defined by the OBSIZE parameter. The center of the object is placed in the center of the object placement area and the object is scaled up or down to fit the block. Scaling in the horizontal and vertical directions is symmetrical. The FIT parameter ensures that all of the data in the object is presented in the object placement area at the largest possible size. The object is not trimmed.

CENTER

Specifies that the center of the object be positioned at the center of the object placement area. Any portion of the object that falls outside the object placement area is trimmed.

REPEAT

Specifies that the origin of the data object be positioned with the origin of the object placement area. The object is then replicated in the X and Y directions. If the last replicated data does not fit in the object area, it is trimmed to fit.

FILL Specifies that the center of the data object be positioned coincident with the center of the object placement area. The data object is then scaled, so that it totally fills the object placement area in both the X

and Y directions. This may require that the object be asymmetrically scaled by different scale factors in the X and Y directions.

OBCHPOS

Specifies the horizontal offset of the object contents within the object placement area as a number.

x-pos Specifies a positive or negative number. The valid options for *x-pos* are described in the SETUNITS command for the horizontal value.

USEOBJ

Specifies that the offset value from the object is to be used. If no value is set in the object, the value defaults to 0.

OBCVPOS

Specifies the vertical offset of the object contents within the object placement area, as defined by the OBSIZE parameter. If OBCVPOS is not specified, it defaults to USEOBJ and uses the value set in the object. If no value is set in the object, the value defaults to 0. The OBCHPOS parameter is used only in LEFT and TRIM mapping of the object into the object placement area.

y-pos Specifies a positive or negative number. The valid options for *y-pos* are described in the SETUNITS command for the vertical value.

USEOBJ

Specifies that the offset value from the object is to be used. If no value is set in the object, the value defaults to 0.

OBROTATE { 0 | 90 | 180 | 270 }

Specifies the object rotation with respect to the current LND's coordinate system.

OBCOLOR *color-name*

Specifies the color to be used as the default color or initial color for the object placement area. The OBCOLOR parameter is used only for objects of the PSEG, GOCA, BCOCA, and IOCA type. If the object type is OTHER, this parameter is ignored. Colors specified must be of the standard OCA color space.

color-name

Specifies standard OCA color space color names, which are: NONE, DEFAULT, BLACK, BLUE, BROWN, GREEN, RED, PINK (or MAGENTA), TURQ (or CYAN), YELLOW, DARKBLUE (or DBLUE), ORANGE, PURPLE, MUSTARD, GRAY, DARKGREEN (or DGREEN), and DARKTURQ (DTURQ, or DARKCYAN, or DCYAN).

In the following example, the page definition pd1 has defined an object with an external name of "PSEGxyz", of object type PSEG. The object has an internal name of "xyzintname". The internal name identifies the object for the LAYOUT OBJECT subcommand when the object is placed. Observe that case is not significant on either the internal nor the external names.

```
PAGEDEF pd1 Replace Yes
  COMMENT 'this is my program';
  FONT XF1 ;

  OBJECT xyzIntName
    OBXNAME PSEGxyz
    OBTYPE  PSEG ;

PAGEFORMAT pf1;
  LAYOUT 'abc' POSITION 2 in  1 in;
    OBJECT xyzintname 1.1 in 2.1 in
      OBSIZE 3 in  5 in
      OBMAP FILL
      OBCOLOR BLUE ;
```

Figure 110. Example of PPFA Support for IOB in a PAGEDEF

The LAYOUT in PAGEFORMAT pf1 places the object on the page 1.1 inches to the left and 2.1 inches below the current LAYOUT position. It also maps the object into the object area with the FILL parameter, which centers the object in the object area and totally fills the area, possibly with different scaling factors in the X and Y directions. It will have an area size of 3 by 5 inches, and overrides the default presentation space color to BLUE.

OVERLAY

Specifies the name of an overlay that is to be positioned relative to the location specified in the LAYOUT command in which the OVERLAY subcommand was named. The PAGEFORMAT OVERLAY command may contain the named overlays. The maximum number of overlays specified for a PAGEFORMAT including the LAYOUT OVERLAY subcommand is 254.

Specifies the electronic overlay that is to be used with this subgroup.

name Specifies the user-access name as defined in the OVERLAY command.

Notes:

1. PPFA checks for duplication of local names. If there is a duplication, the page definition is generated, but a warning message is issued.
2. PPFA does not check for duplicate user-access names.

relative_xpos relative_ypos

Specifies the number of units (inches, mm, and so on) that are added to the position of the layout to position the top-left corner of the overlay. The values for horizontal and vertical may be (+) or (-). The maximum value is + or - 32760 L-units. For example:

- OVERLAY NAME1 2 in 1 in
- OVERLAY NAME2 5 mm 1 mm

Note: Any offset coded in the overlay itself is added to this offset.

OVROTATE { *0* **|** *90* **|** *180* **|** *270* **}**

Specifies the rotation of the placed overlay with respect to the x-axis of the page.

See “FORMDEF Command” on page 169 for an OVROTATE example, which is presented in the FORMDEF description.

SEGMENT

Specifies the name of a segment that is to be positioned relative to the location specified in the LAYOUT command in which the SEGMENT subcommand was named. The PAGEFORMAT SEGMENT command may contain the named segments. The maximum number of segments specified for a PAGEFORMAT including the LAYOUT SEGMENT subcommand is 127.

Specifies the page segment that is to be used with this subgroup.

name Specifies the user-access name as defined in the SEGMENT command.

Notes:

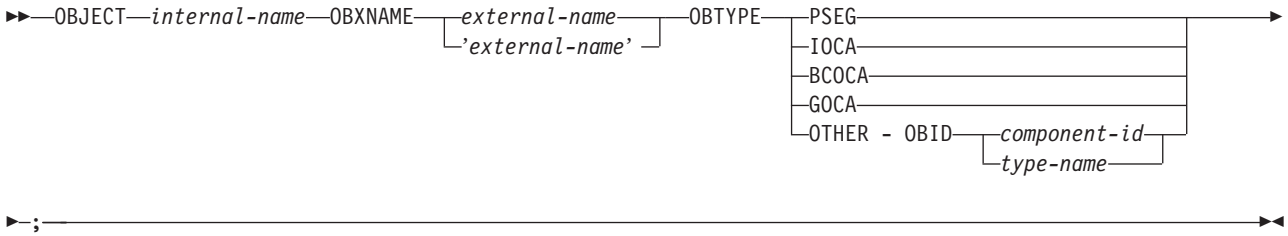
1. PPFA checks for duplication of local names. If there is a duplication, the page definition is generated, but a warning message is issued.
2. PPFA does not check for duplicate user-access names.

relative_xpos relative_ypos

Specifies the number of units (inches, mm, and so on) that are added to the position of the layout to position the top-left corner of the page segment. The values for horizontal and vertical may be (+) or (-). The maximum value is + or - 32760 L-units. For example:

- SEGMENT MYSEG1 2 in 1 in
- SEGMENT MYSEG1 5 mm 1 mm

OBJECT Command (Record Format)



The OBJECT command allows you to define an external object to PPFA. Then you can use the LAYOUT command with the OBJECT subcommand to place the defined object on a page.

You can use one LAYOUT command to place one or many defined objects multiple times with different placement parameters on each placement. On the LAYOUT OBJECT subcommand, enter information about the positioning, rotation, color, object size, and mapping. All positioning is relative to the print line coordinate system. The *internal-name* appears on both the OBJECT command and on the LAYOUT OBJECT subcommand, and is used similar to the way overlays and page segments are defined and placed (or printed).

Notes:

1. The *internal-name* is case insensitive but, other than that, the *internal-name* of the OBJECT command and of the LAYOUT OBJECT subcommand must match exactly.
2. This function requires both print server and printer support. Check your print server and printer documentation.

Subcommands

OBJECT *internal-name*

Identifies the object and also is used to match a LAYOUT OBJECT subcommand. The *internal-name* can be no more than 16 alphanumeric characters.

OBXNAME *external-name*

Specifies the external name of the resource object, which indicates where the actual object is located. For example, in OS/390, the *external-name* is the member name of the actual object in the object library. No prefixes are assumed on the name.

The *external-name* can be no more than 8 alphanumeric characters. If your operating system is AIX, the *external-name* is translated to EBCDIC.

Note: Items within quotation marks are not translated to uppercase or to their EBCDIC code equivalent.

OBTYPE

Used to specify the type of the object. Observe that each of the object types restricts the type of mapping option allowed in the placement of the object (**OBMAP** on the OBJECT subcommand on the LAYOUT command).

PSEG Specifies a page segment object, as described in the *Mixed Object Document Content Architecture (MODCA) Reference Manual*, (SC31-6802). All mapping types (**OBMAP**) are allowed by PPFA; however, print server issues an error if any of the objects contained in the page segment is not compatible with the coded OBMAP parameter.

GOCA

Specifies a graphics object, as described in the *Graphics Object Content Architecture (GOCA) Reference Manual*, (SC31-6804). GOCA allows you to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.

BCOCA

Specifies a bar code object, as described in the *Bar Code Object Content Architecture (BCOCA) Reference Manual*, (S544-3766). BCOCA allows you to specify only the LEFT parameter on the OBMAP subcommand.

IOCA Specifies an image object, as described in the *Image Object Content Architecture (IOCA) Reference Manual*, (SC31-6805). The IOCA object type allows you to specify the TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP command.

OTHER

Specifies other object data. The object data to be included is a paginated presentation object with a format that may or may not be defined by an IBM presentation architecture. When you specify OTHER, you must also specify the OBID parameter. The OTHER object type allows you to specify the TRIM, FIT, LEFT, CENTER, and FILL parameters on the OBMAP subcommand.

OBID

Specifies either a component identifier or a type name from Table 9. The OBID is translated into an Encoded OID and compared to the OID inside the object; they must match.

component-id

Specifies the component identifier.

type-name

Type-name is a name chosen by PPFA as an alternative to coding a component identifier.

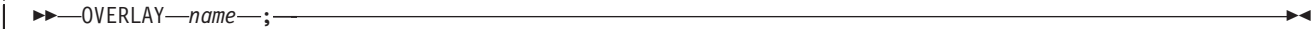
Table 9. Non-OCA Objects supported by IOB.

Type-Name	Component-id	Description of OBID Object Type
EPS	13	Encapsulated PostScript
TIFF	14	Tag Image File Format
WINDIB	17	Device Dependent Bit Map [DIB], Windows Version
OS2DIB	18	Device Dependent Bit Map [DIB], PM Version
PCX	19	Paintbrush Picture File Format
GIF	22	Graphics Interchange Format

Table 9. Non-OCA Objects supported by IOB. (continued)

Type-Name	Component-id	Description of OBID Object Type
JFIF	23	JPEG file Interchange Format
PCLPO	34	PCL Page Object
PDFSPO	25	PDF Single Page Object

OVERLAY Command (Record Format)



This OVERLAY command is used to identify an overlay that will be positioned on a page at some spot other than the position defined within the overlay. This function is similar to the SEGMENT command. A separate OVERLAY command is required for each overlay. A maximum of 254 OVERLAY commands (each of the 254 names must be unique) can be specified for each page format.

The OVERLAY commands are nested within the PAGEFORMAT command.

```
PAGEFORMAT
    [SEGMENT ]
    [ OVERLAY ]
    ...
    [ OVERLAY ]
```

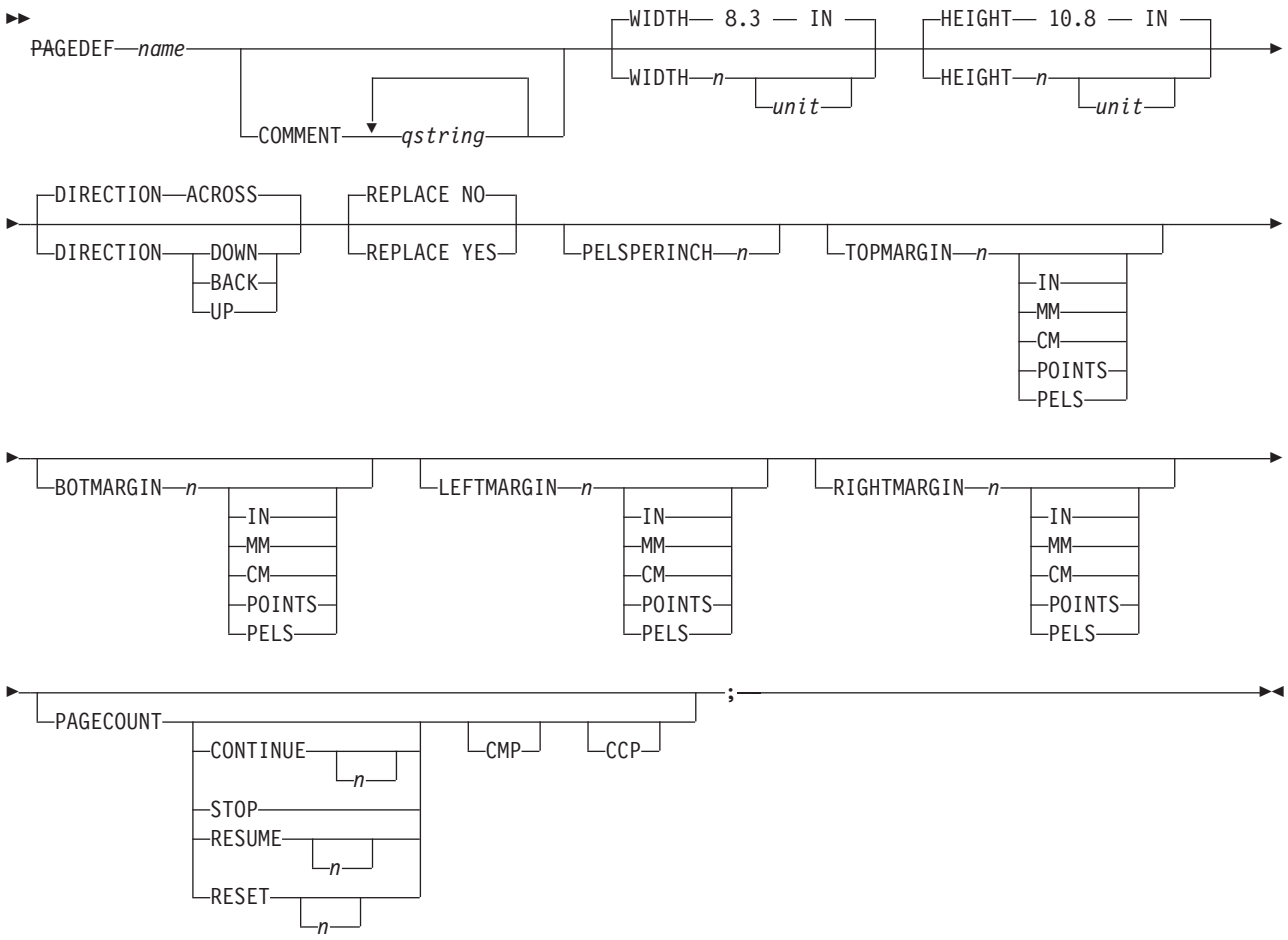
For the overlay to be used, the end-user must embed an Include Page Overlay (IPO) structured field within the line data or unformatted ASCII to be printed. The same name must appear within the structured field as identified by this command, and the page origin must be stated.

OVERLAY *name*;
Identifies the overlay that will be positioned on the page.

name Specifies the user-access name of an overlay to be used with the page definition. The OVERLAY commands can be defined either globally before the first PAGEFORMAT command or nested within the PAGEFORMAT definition.

Note: The prefix 'O1' is not part of the six-character user-access name. The overlay name can be alphanumeric.

PAGEDEF Command (Record Format)



A page definition is a resource used to define how data is to be formatted on a logical page. When generated by PPFA, a page definition is stored as a resource in the page-definition library. This command's subcommands allow you to use with the record format line data.

This command's subcommands must be specified when you define a page definition when using record formatting. All of the PAGEDEF subcommands are optional; and have default values.

PAGEDEF

Identifies the page definition to be used with the print job.

name Defines an alphanumeric name of 1 to 6 characters for the page definition. When page definitions are generated, PPFA assigns the prefix 'P1' to this name as the external resource name.

Subcommands

COMMENT *qstring*

Specifies a user comment. This string comment is placed in the NOP structured field of the page definition.

qstring Specifies a quoted set of strings from 1 to 255 characters in total length.

WIDTH

Defines the width of the logical page.

n A number with up to three decimal places. The width may vary according to the type of printer being used. For more information, refer to your printer documentation. The default is 8.3 IN.

unit Specifies a unit of measurement for the WIDTH subcommand. The choices are IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

HEIGHT

Defines the height of the logical page.

n A number with up to three decimal places. The height may vary according to the type of printer being used. For more information, refer to your printer documentation. The default is 10.8 IN.

unit Specifies a unit of measurement for the HEIGHT subcommand. The choices are IN, MM, CM, POINTS, and PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

DIRECTION

Specifies the print direction of the logical page. Not all printers can print in all print directions. For more information, refer to your printer documentation.

Note: Some printers such as the IBM 3835 Page Printer and the IBM 3900 Advanced Function Printer have a different media origin and require different direction settings than most page printers. For printing in the landscape page presentation when using wide forms, the PRESENT subcommand must be specified on the FORMDEF command to produce readable output. Alternatively, if you have existing page definitions, the UP direction can be used in the page definition without changes to the form definition to produce the same result.

ACROSS
The page is printed with the characters added *left to right* in each line, and the lines added from the top to the bottom.

DOWN
The page is printed with the characters added to the page from *top to bottom,* and the lines added from the right to the left.

BACK The page is printed with the characters added to the page from *right to left,* and the lines added from the bottom to the top.

UP The page is printed with the characters added to the page from *bottom to top,* and the lines added from the left to the right.

DIRECTION effects the meaning of the following new margin parameters:

Notes:

1. If the Direction is Across, the TopMargin refers to the margin in the short end of the physical page where the tops of the characters point toward that same short end.
2. If the Direction is Down, then TopMargin refers to the margin in the long end of the physical page where the tops of the characters point toward that same long end.

REPLACE
Specifies whether this page definition is to replace an existing one with the same resource name in the library.

NO This page definition does not replace one with the same resource name in the library.

If a page definition with the same resource name does not exist in the library, this page definition is stored.

YES If a page definition with the same resource name already exists in the library, this page definition replaces it.

If a page definition with the same resource name does not exist in the library, this page definition is stored.

PELSPERINCH *n*
Specifies the Logical Units in pels per inch for this page definition. Use the PELSPERINCH parameter to tell PPFA the pel resolution of your printer in order to generate more exact object placements.

n Specifies an integer number between 1 and 3,276, which determines the Logical Units in pels per inch.

Note:

If the L-Units are not specified on this page definition, they default to 240 pels per inch.

```
PAGEDEF xmp01 replace yes
   PELSPERINCH 300;
   FONT abc;

 PAGEFORMAT P1
   width  7 in
   height 3 in;
 LAYOUT 'abc';

 PAGEFORMAT P2
   width  7 in
   height 3 in
   PELSPERINCH 1200;
 LAYOUT 'def';
```

Figure 111. PELSPERINCH example

In Figure 98 on page 227, the page definition xmp01 has specified L-Units as 300 pels per inch. Because the PAGEFORMAT P1 does not specify L-Units, it inherits 300 pels per inch. PAGEFORMAT P2 specifies L-Units as 1200 pels per inch.

The width and height in PAGEFORMAT P1 (7 in, 3 in) produces internal and structured field values of 2100 and 900, whereas in PAGEFORMAT P2 the same code produces values of 8400 and 3600, because of the difference in L-Units.

TOPMARGIN
This keyword with parameters specifies the amount of space to be reserved at the top of the page.

The default is 80% of the current line spacing.

BOTMARGIN
This keyword with parameters specifies the amount of space to be reserved at the bottom of the page. Only PageTrailer data can be written into this area. If a graphic has not been ended at the time information is being placed in the bottom margin, the graphic will be ended prior to the bottom margin. The default is 0.

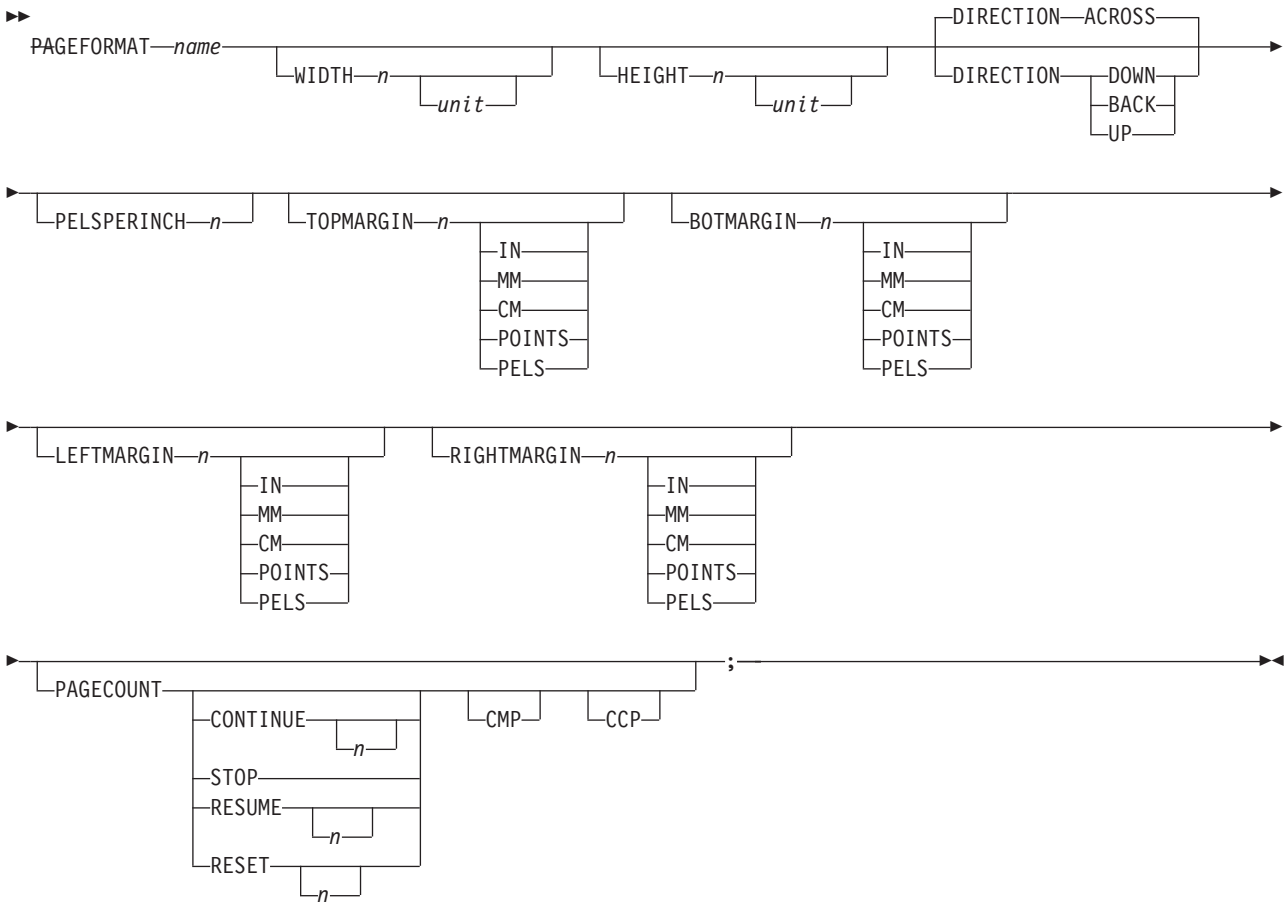
LEFTMARGIN
This keyword with parameters specifies the amount of space to be reserved at the left of the page. This is to be used only in conjunction with the DRAWGRAPHIC commands. Although PPFA will collect the left margin information, it will use the value only within PPFA to define an area. The value itself will not be passed in the datastream. The default is 0.

RIGHTMARGIN
This keyword with parameters specifies the amount of space to be reserved at the right of the page. This is only to be used in conjunction with the DRAWGRAPHIC commands. Although PPFA will collect the right margin information, it will use the value only within PPFA to define an area. The value itself will not be passed in the datastream. The default is 0.

PAGECOUNT
This keyword allows the user to specify how the page counting is to be handled when switching between pageformats.

- CONTINUE - page counting continues from the previous Pageformat - this is the default. The *n* value is only used on the first Pageformat in the job, otherwise it is ignored. If this is the first Pageformat and no *n* value is specified, it defaults to one.
- STOP - page counting stops. Page count will be captured from the previous Pageformat, but will not continue to count.
- RESUME - page counting continues from wherever it was the last time this Pageformat was called. The *n* value sets the value only the first time the Pageformat is invoked.
- RESET - page counting is reset to the value within the *n* value. If no *n* value is entered, then the page numbers are reset to one.
- CMP - Count MO:DCA Pages option. Tells print server to count any imbedded MO:DCA pages in the page count.
- CCP - Count Constant Pages options. Tells print server to count any pages that have no variable data on them.

PAGEFORMAT Command (Record Format)



Page formats are subsets of page definitions. If you want to use more than one set of specifications to format a page within a single print job, you must use more than one page format. To change page formats, use conditional processing or insert an Invoke Data Map structured field in your print file. (Page formats are known to print server as data maps.) If you do not use conditional processing or if you do not insert an Invoke Data Map structured field, print server uses only the first page format in the page definition. Page formats are placed in the page definition in the order in which they are generated.

PAGEFORMAT subcommands have no fixed defaults. The entire PAGEFORMAT command and all of its subcommands can assume defaults. If any PAGEFORMAT subcommand is omitted, its value is selected from the corresponding subcommand in the governing PAGEDEF command.

This command can be omitted for the first page format in a page definition if only one page format is used. If omitted, PPFA assigns a page format name by using the page-definition name, including the 'P1' prefix.

PAGEFORMAT *name*
Specifies an alphanumeric name of 1 to 8 characters. This name must be unique within the page definition.

The following subcommands are used for each page format. They may be issued in the same way as in a page definition. Values specified in the PAGEDEF subcommands are used if any of the following subcommands are not defined within a page format.

Subcommands

WIDTH
Defines the width of the logical page.

n A number with up to three decimal places is used. The width may vary according to the type of printer being used. For more information, refer to your printer documentation.

unit Specifies a unit of measurement for the WIDTH subcommand. The choices are IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

HEIGHT
Defines the height of the logical page.

n A number with up to three decimal places. The height may vary according to the type of printer being used. For more information, refer to your printer documentation.

unit Specifies a unit of measurement for the HEIGHT parameter. The choices are IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command has not been issued.

DIRECTION
Specifies the print direction of the logical page. Not all printers can print in all print directions. For more information, refer to your printer documentation.

Note: Some printers such as the IBM 3835 Page Printer and the IBM 3900 Advanced Function Printer have a different form origin and require different direction settings than most page printers. For printing in the landscape page presentation when using wide forms, the PRESENT subcommand must be specified on the FORMDEF command to produce readable output. Alternatively, if you have existing page definitions, the UP direction can be used in the page definition without changes to the form definition to produce the same result.

ACROSS
The page is printed with the characters added to the page from *left to right*, and the lines added from the top to the bottom.

DOWN
The page is printed with the characters added to the page from *top to bottom*, and the lines added from the right to the left.

BACK The page is printed with the characters added to the page from *right to left*, and the lines added from the bottom to the top.

UP The page is printed with the characters added to the page from *bottom to top*, and the lines added from the left to the right.

DIRECTION effects the meaning of the following new margin parameters.

- If the Direction is Across, then TopMargin refers to the margin in the short end of the physical page where the tops of the characters point toward that same short end.
- If the Direction is Down, then TopMargin refers to the margin in the long end of the physical page where the tops of the characters point toward that same long end.

PELSPERINCH *n*

Specifies the Logical Units in pels per inch for this page format. Use the PELSPERINCH parameter to tell PPFA the pel resolution of your printer in order to generate more exact object placements.

n Specifies an integer number between 1 and 3,276, which determines the Logical Units in pels per inch.

Note: If the L-Units are not specified on the page format, they are inherited from the page definition that contains this page format. See Figure 98 on page 227.

TOPMARGIN

This keyword with parameters specifies the amount of space to be reserved at the top of the page.

The default is 80% of the current line spacing.

BOTMARGIN

This keyword with parameters specifies the amount of space to be reserved at the bottom of the page.

The default is 0.

LEFTMARGIN

This keyword with parameters specifies the amount of space to be reserved at the left of the page. This is only used in conjunction with the DRAWGRAPHIC commands. Although PPFA collects the left margin information, the value is used only within PPFA to define an area. The value itself will not be passed in the datastream.

The default is 0.

RIGHTMARGIN

This keyword with parameters specifies the amount of space to be reserved at the right of the page. This is only to be used in conjunction with the DRAWGRAPHIC commands. Although PPFA collects the right margin information, it will use the value only within PPFA to define an area. This value itself will not be passed in the datastream.

The default is 0.

PAGECOUNT

This keyword allows the user to specify how the page counting is to be handled when switching between page formats.

- CONTINUE - page counting continues from the previous page format - this is the default. The *n* value is only used on the first Pageformat in the job, otherwise it is ignored. If this is the first Pageformat and no *n* value is specified, it defaults to one.

- STOP - page counting stops. Page count will be captured from the previous page format, but will not continue to count.
- RESUME - page counting continues from wherever it was the last time this page format was called. The *n* value sets the value only the first time page format is invoked.
- RESET - page counting is reset to the value within the *n* value. If no *n* value is entered, then the page numbers are reset to one.
- CMP - (Count MO:DCA Pages) option. Tells print server to count any imbedded MO:DCA pages in the page count.
- CCP - (Count Constant Pages) options. Tells print server to count any pages that have no variable data on them.

SEGMENT Command (Record Format)

Use the SEGMENT command only if you want page segments to be loaded to the printer before the page begins printing. If segments are used repeatedly and need to be available in the printer, this eliminates the need to load them each time. However, they do take up raster-pattern storage. If the segments are included on a page but not in the SEGMENT command, they are loaded to the printer as they are used in the print data.

A separate SEGMENT command is required for each page segment with a maximum of 127 SEGMENT commands within a single page format.

```
PAGEFORMAT
 SEGMENT
 ...
 SEGMENT
```

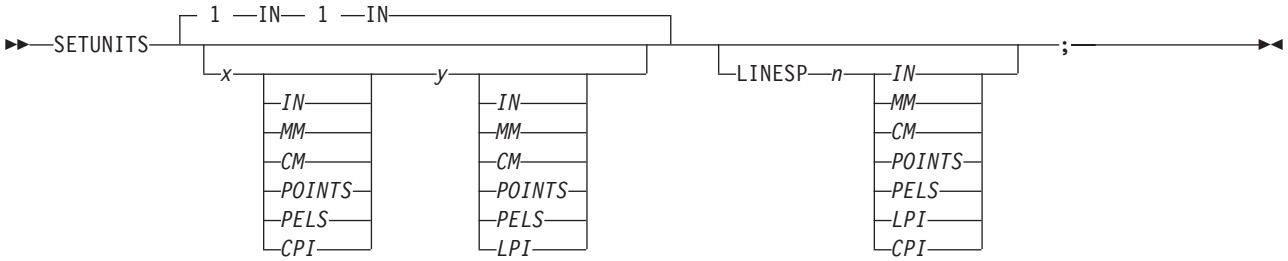
A SEGMENT command is nested within the page format and follows the PAGEFORMAT command.

SEGMENT *name*

Specifies the alphanumeric name of 1 to 6 characters (user-access name) of the page segment. Each name must be unique within a single page format.

Note: The prefix 'S1' is not part of the six-character user-access name.

SETUNITS Command (Record Format)



The SETUNITS command specifies the value and the unit of measurement that will be the defaults for any subsequent measurement parameter in all of the commands and subcommands. These values remain the default values until another SETUNITS command is specified. The SETUNITS command should be specified as the first command in a page definition. If neither this command nor a measurement parameter is specified, the defaults identified within the following description are used.

SETUNITS

Specifies the value and the unit of measurement that will be the defaults for any subsequent measurement parameter in all of the commands and subcommands.

x-pos Specifies the number used for horizontal measurement. A number with up to three decimal places is used. The default is 1. The choices are IN, MM, CM, POINTS, PELS, or LPI. The default is IN.

Note: This value affects subsequent OFFSET subcommands.

y-pos Specifies the number used for vertical measurement. A number with up to three decimal places is used. The default is 1. The choices are IN, MM, CM, POINTS, PELS, or LPI. The default is IN.

Note: This value affects subsequent OFFSET subcommands.

Using CPI and LPI Units of Measurement

The CPI and LPI units of measurement make it possible to write the following command:

```
SETUNITS 10 CPI 6 LPI ;
```

This command sets the units of measurement for horizontal and vertical spacing in terms of characters per inch and lines per inch. You can then use the OFFSET subcommand specifications to increment the spacing one character or one line at a time. The distance specified by *n* characters over and by *n* lines down is defined in the governing SETUNITS command. In this example, there are 10 characters per inch (CPI) and 6 lines per inch (LPI).

Subcommand

LINESP *n unit*

Determines the line density or "leading" of the text. Any unit of measurement can be used. This subcommand value affects the LAYOUT NEXT subcommand.

n The meaning is determined by the type of unit-of-measurement specified in the unit parameter.

LPI The number of lines per inch

All others
The distance between lines

unit Specifies a unit of measurement. The choices are:

IN Inch

LPI Lines-per-inch

MM Millimeter

CM Centimeter

PELS L-units per inch (The number of L-units per inch can be defined by the user or can default to 240 L-units in an inch)

POINTS
Points per inch (72 points in an inch)

Part 4. Appendixes

Appendix A. System Dependencies for PPFA

PPFA is a cross system product that operates on:
- VSE (Virtual Storage Extended)
- OS/390 (Operating System 390)
- VM (Virtual Machine)
- AIX (Advanced Interactive Executive)
- OS/400 (Operating System 400)
- Windows NT (Operating System)

For the level of the operating system on which PPFA can run, refer to the *Licensed Program Specification*.

PPFA creates page definitions and form definitions used for printing by PSF/OS/390, PSF/VM, and OS/400. PPFA creates a data base file member containing AFPDS that can be used to create the OS/400 objects PAGDFN and FORMDF, using the CRTPAGDFN and CRTFORMDFN commands. Page definitions and form definitions created on one system can be used for printing on another system. However, not all versions of print server support all functions provided by PPFA. Use the Programming Guide or User's Guide for your print server system to determine which functions are supported by your system.

While page definitions and form definitions created on one system can be used on any of the systems, the method of creating these resources is different.

Each system is presented to show how PPFA creates page definitions and form definitions. In the examples, the prefixes F1 and P1 are automatically added by PPFA to the user name designated for form definitions and page definitions.

VSE Environment

PPFA can operate in any partition of VSE. It operates in batch mode but is able to operate in a partition occupied by an interactive processor.

Storing PPFA Resources

Form definitions and page definitions are stored by name in a library. In VSE, sub-libraries are created for form-definition and page-definition storage within the system library.

The following job control statements (JCS) give an example of a PPFA execution under VSE. The 'C' in Column 72 indicates a continuation.

```
                                                                 72 Column
* $$ JOB                                                           |
// CLASS=0                                                         |
// JOB    PPFAEXEC                                                 ↓
// ASSGN  SYSLST,00E                                               C
// OPTION DUMP
// LIBDEF PHASE,SEARCH=(ppfa.program),TEMP
// EXEC   PGM=AKQPPFA,SIZE=AUTO,
          PARM='FORMLIB=ppfa.formdef,PAGELIB=ppfa.pagedef,C
               size=128K'

   PPFA control statements )
                               )
                               >    SYSIPT file
                               )
/*
/&
* $$ EOJ
```

Rules for VSE

The rules for VSE commands in a PPFA execution follow:

- All characters in the EXEC statement parameters must be uppercase. Each keyword in a parameter must be unique; PPFA issues an error message if any keywords are duplicated.
- AKQPPFA is the program name.
- SIZE= is the maximum available storage in the program. The SIZE parameter is not used to specify a PPFA work area size.
- PARM= is used to input PPFA parameters.
  - FORMLIB= (or PAGELIB=) libraryname.sublibraryname
    - All library names are alphanumeric (1 to 7 characters); the first character must be alphabetic.
    - All sublibrary names are alphanumeric (1 to 8 characters) including the first character.
  - size=*nn*K or *nnn*M
    - Defines the work area in which PPFA compiles the page definitions and form definitions. The default is 128k and the minimum 4K.
- The format for the FORMLIB or PAGELIB parameters is:
  - FORMLIB= (or PAGELIB=) libraryname.sublibraryname, where library names are 1 to 7 characters long and sublibrary names are 1 to 8 characters long.
  - All characters (library and sublibrary names) are alphanumeric, except that the first character must be alphabetic.
- Libraries must be defined prior to PPFA execution; Otherwise, an ABEND occurs. PPFA can perform a syntax check without libraries being defined, but it cannot define its own libraries;
- The SYSIPT file drives PPFA. It contains the commands used to build form definitions and page definitions. The records are fixed-length records of either 80 or 81 bytes, which can be blocked. The last 8 bytes of the records are treated as comments.

OS/390 Environment

The following example shows you how to create page definitions and form definitions in the OS/390 environment.

Form definitions and page definitions are stored by name in a library.

The following job control language (JCL) statements are an example of PPFA execution under OS/390:

```
//JOBPPFA     JOB TOKYO
//STEP        EXEC PGM=AKQPPFA
//STEPLIB     DD DSN=ppfa.program,DISP=SHR
//SYSPRINT    DD SYSOUT=A
//FORMLIB     DD DSN=ppfa.formlib,DISP=SHR
//PAGELIB     DD DSN=ppfa.pagelib,DISP=SHR
//SYSIN       DD *

   PPFA control statements
   .
   .
   .
/*
```

The SYSIN file contains the commands used to build form definitions and page definitions. The records can be fixed length or variable length, and they can be blocked. The maximum length for fixed-length records is 100 bytes; the maximum length for variable-length records is 104 bytes. In the case of fixed 80-byte records, the last 8 bytes are treated as comments.

The record format for the page-definition and form-definition data sets must be variable blocked (VBM). The block size and record length must be 8209 and 8205. PPFA uses all of the available storage in the program.

Note: When concatenating multiple data sets in the SYSIN data definition, you must ensure that the data set with the largest block size is first in the concatenation order. Otherwise, the output may not be what you expect.

VM Environment

To create a page definition and form definition running PPFA under VM, use the following command syntax:

Note: The defaults require only filename (fn) and filetype (ft) for your PPFA source file.

```
PPFA fn ft [ fm ] [ ( [PAGEDEF ( ft [ fm | A1 ] )
                      [FORMDEF ( ft [ fm | A1 ] )
                      [LISTING ( ft [ fm | A1 ] )
                      [SIZE   nnnn{K|M} ] ) ]
```

PPFA is the command to run PPFA on VM. The filename (fn) is the name of your file that contains the PPFA control statements. The filename (fn) and filetype (ft) are required parameters. When you specify only the fn and ft, the filemode goes to your default disk.

The record format of the PPFA input source file is either V or F. The variable record length is a maximum of 100 bytes. In the case of a fixed 80-byte record, the last 8 bytes are treated as comments.

The PPFA command may include any of four optional parameters: PAGEDEF, FORMDEF, LISTING, and SIZE.

- Each keyword parameter can be abbreviated as two letters.
- All parameters in the command can be omitted. However, any optional parameter following an open parenthesis must be specified.
- Operands must be enclosed in parentheses when more than one operand is specified for one parameter. Parentheses can be omitted when only one operand is specified for one parameter. Also, the final closing parenthesis can be omitted.
- Any operand string longer than eight characters is truncated to the first eight characters.
- Any parameter or operand can be separated from others by parentheses or blanks. The only exceptions are the K and the M operands of a size parameter. For example, in size 256K you cannot separate the 256 from the K.
- The same parameter must not be specified more than once in a command. If duplicate parameters or operands appear, PPFA issues an error message and terminates the program.
- For errors associated with a VM execution command, PPFA issues an error message with a return code 20, and does not generate any files (object or listing).
- No optional parameters can follow the open parenthesis occurring after the input source file ID.
- The size parameter varies according to the size of the command stream. Most command streams do not need a size value because the default specifies enough space for processing.

PAGEDEF Parameter

PAGEDEF (which can be abbreviated as PA) is the keyword used to specify the name of a page-definition resource. (The filetype is required; the filemode is optional. If you do not specify a filemode, A1 is assumed.) The page-definition filename is obtained from your input file, and P1 is prefixed to that name.

As an example, for the command

```
PPFA PCOM DATA A1 ( PAGEDEF ( PAGEOBJ B1 ) )
```

the input file, PCOM DATA A1, contains the following control statements:

```
PAGEDEF PAGE1;
  PRINTLINE;
FORMDEF FORM1;
```

The result is a page-definition resource file with the filename P1PAGE1, the filetype PAGEOBJ, and the filemode B1.

If the page definition parameter is not used, a page-definition resource with the default name P1 (the page definition name from input file) PDEF38PP A1 is created.

The record format of the object file is VM and VA (5A records). 5A records contain the character X'5A' in the first byte of each record. The record size is up to 8205 bytes.

FORMDEF Parameter

FORMDEF (which can be abbreviated as FO) is the keyword used to specify the name of a form-definition resource. (The filetype is required; the filemode is optional.) The filename is obtained from your input file, and F1 is prefixed to that name. As an example, for the command

```
PPFA PCOM DATA A1 ( FORMDEF ( FORMOBJ B1 ) )
```

the input file, PCOM DATA A1, contains the following control statements:

```
PAGEDEF PAGE1;
  PRINTLINE;
FORMDEF FORM1;
```

The result is a form-definition resource file with the filename F1FORM1, the filetype FORMOBJ, and the filemode B1.

If the form-definition parameter is not used, a form-definition resource with the default name F1 (form-definition name from input file) FDEF38PP A1 is created.

The record format of the object file is VM and VA (5A records). The record size is up to 8205 bytes.

LISTING Parameter

LISTING (which can be abbreviated as LI) is the keyword used to specify the name of an output listing file. You can specify the filetype and filemode of the resource; the filetype is required. If you do not specify a filemode, A1 is assumed. The filename is the same as the PPFA input filename.

As an example, for the command

```
PPFA PCOM DATA A1 ( LISTING ( LISTOUT B1 )
```

the result is an output listing file with the name PCOM LISTOUT B1.

If the LISTING parameter is not used, an output listing file with the default name (PPFA input filename) LISTING A1 is created.

The record format of an output listing file is VA. The record length is 121 bytes (120 bytes + 1 byte (channel control number)). CC numbers are 0 to 12 in the first column of the line data file.

RUN and OPTIONS file

This is an example of the VM files that print your data file with the form definition and page definition that you specify.

VM EXEC Example

```
*************************************************************
/*THE ENVIRONMENT IS NOW SET UP TO PRINT */
'CP SP PRT TO NET NOHOLD CLASS A FORM PRT035 COPY 1';
'CP TAG DEV PRT WASVM SYSTEM';
'PSF EXAMP1 PRTDATA A1 ( OPTIONS (EXAMP1) )';
/*RESTORE THE ENVIRONMENT TO PRINT SOMETHING OTHER THAN THIS EXAMPLE*/
*************************************************************
```

VM OPTIONS Example

```
*************************************************************
FORMDEF ( F1EXAMP1   FDEF38PP    ) SEND
*************************************************************
PAGEDEF ( P1EXAMP1    PDEF38PP    ) SYSDISK
*************************************************************
OVERLAY ( *          OVLY38PP    ) SYSDISK
*************************************************************
* COMMON OPTIONS
*************************************************************
CC
NOTRC
BIN 1
CKPTPAGE 0
DATACK UNBLOCK
NODUMP
FILE SEND
FONT     ( * FONT3820   ) SYSDISK
MESSAGES NO
NOOPT
PAGESEG ( * PSEG38PP   ) SYSDISK
TRACE OFF
*************************************************************
```

AIX Environment

The **ppfa** command creates form definitions and page definitions on the AIX operating system. After they are created, you can transfer the form definitions and page definitions to other operating systems (such as OS/390, VM, or VSE) to use as AFP resources.

Syntax

ppfa [**-f***path.ext*] [**-p***path.ext*] [**-s***path.ext*] [**-x**] *inputfile*

Flags and Values

You can specify these flags and values with the **ppfa** command.

inputfile
: The file containing the PPFA source statements to be “processed”.

-f*path.ext*

Add path and extension information to the names of form definitions generated by PPFA. (The *name* itself will come from the FORMDEF command.)

-p*path.ext*

Add path and extension information to the names of page definitions generated by PPFA. (The *name* itself will come from the PAGEDEF command.)

-s*path.ext*

Add path and extension information to the listing file. The *name* of the listing file will be the same as the *name* of the inputfile.

Thus, for "FORMDEF name" when PPFA was invoked with:
ppfa -fpath.ext infile

it generates form definition:
/path/name.ext

Also, for "PAGEDEF name" when PPFA was invoked with:
ppfa -p/root/abc/def.xyz.nnn infile

it generates page definition:
/root/abc/def.xyz/name.nnn

In another example, if you enter:
`ppfa -pabc/def.xyz` *input.file*

and it has a PAGEDEF statement in the source, then the page definition created will be either:
abc/def/P1NAME.xyz or
./abc/def/P1NAME.xyz

However, if you enter:
`ppfa -p/abc/def.xyz` *input.file*

PPFA generates the file:
/abc/def/P1name.xyz not
.//abc/def/P1name.xyz

-x Causes **ppfa** to interpret information found in columns 1-72 of the *inputfile*. The information in the rest of the columns will be ignored. This is useful if you are downloading a Fixed-80 file from the host.

Examples

1. To create a form definition from an input file called **johnb** in the current library containing the PPFA source statements, enter:
   ```
   ppfa johnb
   ```
   The generated form definition is stored in the current library.
2. To create a form definition from an input file called **maryc** containing the PPFA source statements, and then storing the generated form definition in the **/usr/lpp/resources** library, enter:
   ```
   ppfa -f/usr/lpp/resources maryc
   ```

Files

/usr/lpp/ppfa/bin/ppfa
PPFA program

/usr/lpp/psf/ppfa
Source code for the form definitions and page definitions supplied with InfoPrint Manager for AIX

OS/400 Environment

General Information

Support has been added to give the OS/400 users full AFP function.

Note: In this document, the following terms are used interchangeably:
- ***LINE** and line data
- ***AFPDSLINE** and mixed data
- **PAGDFN** and page definition
- **FORMDF** and form definition

This section describes the following printer file attributes:
- **DEVTYPE**
- **CTLCHAR**
- **TBLREFCHR**
- **AFPCHARS**
- **PAGDFN**
- **FORMDF**

This section also provides information about line data application considerations, device type considerations, OS/400 printer file parameters, carriage control characters (ANSI and Machine), Table Reference Characters, IGC parameters, Medium-Map-Name (**INVMMAP**) DDS keyword, restrictions when using **PAGDFN** and **FORMDF**, as well as information about the **CVTPPFASRC** command.

DEVTYPE Values

To place line data or mixed data onto the printer spool, specify either the ***LINE** or ***AFPDSLINE** values with the **DEVTYPE** parameter for the **CRTPRTF, CHGPRTF,** and **OVRPRTF CL** commands.

***LINE** Line data is placed onto the spool. For ***LINE**, specify any of the following:
- **CTLCHAR(*FCFC)**
- **CTLCHAR(*MACHINE)**
- **CTLCHAR(*NONE)**

To place line data and skipping or spacing controls directly onto the printer spool without converting it to another data stream, specify ***LINE**. The line data is not in a printer-ready format and, like AFPDS, will be converted to the appropriate printer format at print time.

***AFPDSLINE**
Line data and AFPDS (mixed data) is placed onto the printer spool. For ***AFPDSLINE**, specify any of the following:
- **CTLCHAR(*FCFC)**
- **CTLCHAR(*MACHINE)**

You can specify page definitions to format traditional application line data without making any application programming changes. If, however, you want to use any one of the following AFPDS structured fields (which can be intermixed with line data), you must specify, in hex, a X'5A' record in the output buffer. Refer to *Advanced Function Printing: Programming Guide and Line Data Reference* , (S544-3884), for more information about mixed documents and MO:DCA.

- Invoke Data Map
- Invoke Medium Map
- Invoke Page Segment
- Include Page Overlay
- Presentation Text

Refer to the *Advanced Function Presentation Programming Guide and Line Data Reference* (S544-3884) for more information about mixed data streams.

CTLCHAR Values

When machine code control characters exist in the data (rather than ANSI control characters), specify the **CTLCHAR** parameter with a value of ***MACHINE** on the **CRTPRTF, CHGPRTF**, or **OVRPRTF CL** commands.

See Table 12 on page 334 for information about machine code control characters.

TBLREFCHR Parameter

To indicate whether a table reference character (TRC) exists in the data, specify the **TBLREFCHR** parameter with the **CRTPRTF, CHGPRTF**, or **OVRPRTF CL** commands.

If forms control characters are used with the data, the TRC follows the forms control character, but precedes the data bytes. If forms control characters are not used, the TRC is the first byte of the data record. As with forms control characters, if table reference characters are used, every data record must contain a TRC byte.

Note: The **TBLREFCHR** parameter is ignored if specified for ***USERASCII**, ***SCS**, ***IPDS**, and ***AFPDS** device types.

See “Table Reference Characters (TRC)” on page 335 for more information.

AFPCHARS Parameter

To identify up to four 4-byte names of coded fonts, specify the **AFPCHARS** parameter with the **CRTPRTF, CHGPRTF**, or **OVRPRTF CL** commands. The 4-byte names are concatenated to ‘X0’ to identify up to four coded fonts that are needed when **TBLREFCHR** is used within the data.

PAGDFN Parameter

To use or identify a fully-qualified page definition, specify the **PAGDFN** parameter with the **CRTPRTF, CHGPRTF**, or **OVRPRTF CL** commands.

A page definition is an AFP resource object that allows line data and mixed data to be formatted, independent of the application. You can specify page definitions with ***LINE** or ***AFPDSLINE** data. After PSF/400 completes formatting, it converts the line data and page definition to IPDS.

Whenever you specify line data or mixed data through NJE or PrintManager/400, with output directed to an AFP printer but do not specify a page definition, an inline page definition is built from the print parameters and passed to PSF/400.

Depending upon the specifications given, some printer file parameters may be ignored when the spooled file is printed through PSF/400. For example, if you specify a page definition on the **CRTPRTF, CHGPRTF**, or **OVRPRTF CL** command, and also specify line data or mixed data, an inline page definition will not be built from the printer file parameters. In this case, if you send the data to an AFP printer, PSF/400 ignores the following print parameters:

- **CDEFNT**
- **CHRID**
- **CPI**
- **FNTCHRSET**
- **FOLD**
- **FONT**
- **LPI**
- **MULTIUP**
- **PAGESIZE**
- **PAGRTT**
- **REDUCE**

However, in this example, if you send the data to a non-AFP printer with a devtype of ***LINE**, the page definition parameter is ignored and the print parameters are used. The line data is converted to SCS or IPDS.

If you specify a page definition, but do not specify a form definition, an inline form definition is built from the appropriate printer file parameters, including:

- **DRAWER**
- **DUPLEX**
- **FORMFEED**
- **PAGRTT**
- **PRTQLTY**

FORMDF Parameter

To use or identify a fully-qualified form definition, specify the **FORMDF** parameter with the **CRTPRTF, CHGPRTF** or **OVRPRTF CL** commands.

A form definition is an AFP resource object that defines the characteristics of the form, including:

- overlays
- position of page data on the form
- rotation
- modification to pages

You can specify a form definition with the following data:

- ***AFPDS**
- ***AFPDSLINE**
- ***LINE**
- ***IPDS**
- ***SCS**

PSF/400 accepts ***AFPDS**, ***AFPDSLINE**, and ***LINE** data, and requires a form definition to print. (When you specify an AFP printer, the ***IPDS** and ***SCS** data streams are converted to AFPDS.)

Whenever you specify any output directed to an AFP printer but do not specify a form definition, an inline form definition is built from the print parameters and passed to PSF/400.

Depending upon the specifications given, some printer file parameters may be ignored when output is printed through PSF/400. For example, if you specify a form definition on the **CRTPRTF**, **CHGPRTF** or **OVRPRTF CL** command and also specify line data or mixed data, an inline form definition will not be built from the printer file parameters. In this example, if you send the data to an AFP printer PSF/400 ignores the following printer file parameters:

- **BACKMGN**
- **CORNER STPL**
- **DRAWER**
- **DUPLEX**
- **EDGESTITCH**
- **FORMFEED**
- **FRONTMGN**
- **MULTIUP (N_UP)**
- **PAGRTT**
- **PRTQLTY**
- **REDUCE (N_UP)**
- **SADLSTITCH**

However, if you send the data to a non-AFP printer with a devtype of ***IPDS** or ***SCS** in this example, the form definition parameter is ignored and the printer file parameters are used.

When specifying a form definition on the printer file, the values you specify on the **DRAWER** and **DUPLEX** parameters will override the drawer and duplex values specified in the form definition. If you want to use the drawer and duplex values specified in the form definition, you must specify **DRAWER(*FORMDF)** and **DUPLEX(*FORMDF)** on the printer file.

To indicate an output bin for a form definition, specify the **OUTBIN** parameter on the print file.

Application Considerations for Line Data

Line data and mixed data on OS/400 are used by OS/390 AFP users who are migrating data to OS/400 and OS/400 users who are generating AFPDS, IPDS, or SCS data streams.

If you are a OS/390 user, you should be familiar with the concepts of AFP and page definitions. When using line data, page definitions, and form definitions, your applications are generating line data with either an ANSI or machine code control character in column 1 of the spooled output. To migrate data to OS/400 prior to OS/400 Version 3, Release 2, and Version 3, Release 7, OS/390 users had to use PrintManager/400 API calls to rewrite their applications. With line data support, applications do not need to be rewritten. To migrate data to OS/400, specify **CTLCHAR(*FCFC)** for ANSI code control, or **CTLCHAR(*MACHINE)** for machine code control. You should also specify **DEVTYPE(*LINE)** or **DEVTYPE(*AFPDSLINE)**.

If you are an OS/400 user, and want to use line data, page definitions, and form definitions, you must determine whether your application generates an ANSI control character in column 1 of your spooled output.

If your application does generate an ANSI control character in column 1 of your spooled output to control skipping and spacing, specify **CTLCHAR(*FCFC)** on the printer file. Also, to convert to line data, specify the following in the OS/400 printer file:

- **DEVTYPE(*LINE)**
- **PAGDFN** (a page definition)
- **FORMDF** (a form definition) optional format

Note: Specifying a form definition in this format is optional; you could specify that an inline form definition be built from the print file parameters by indicating **FORMDF(*NONE)** on the OS/400 printer file.

You can change your application to place an ANSI control character in column 1 of your spooled output to control skipping and spacing, by using a language or application construct (such as a **SKIP** or **SPACE** option on a COBOL **WRITE** statement), or by making an RPG output specification.

When you specify a device type of ***AFPDS**, ***IPDS**, or ***SCS**, control information is used to generate the appropriate skipping or spacing commands in the specified data stream. The control information for ***AFPDS** and ***LINE** that is passed by the compilers and application is converted to a machine code control character. Thus, applications that do not use ANSI control characters can generate line data with control characters onto the spool and use a page definition for post spool formatting, if you specify **CTLCHAR(*NONE)** and **DEVTYPE(*LINE)**.

The system will insert ANSI control characters into column 1 and change the spooled file attributes to CTLCHAN (*FCFC).

Device Type Considerations

When using line data, you can specify various combinations of **DEVTYPE(*LINE)**, **PAGDFN** and **FORMDF** parameter support on the print file. For example:

- Specify DEVTYPE(*LINE), PAGDFN and FORMDF
  - When you print to an AFP printer, PSF/400 uses the **PAGDFN** and **FORMDF** parameters to transform the data to IPDS.
  - When you print to a non-AFP printer, the **PAGDFN** and **FORMDF** parameters are ignored. The parameters on the print file are used, and the line data is transformed to IPDS or SCS.
- Specify DEVTYPE(*LINE), no PAGDFN, with FORMDF
  - When you print to an AFP printer, an inline page definition is built from the print file parameters. PSF/400 uses the inline page definition and user-specified **FORMDF** parameter to transform the data to IPDS.
  - When you print to a non-AFP printer, the **FORMDF** parameter is ignored. The print file parameters are used, and the line data is transformed to IPDS or SCS.
- Specify DEVTYPE(*LINE), PAGDFN, no FORMDF
  - When you print to an AFP printer, an inline form definition is built from the print file parameters. PSF/400 uses the user-specified **PAGDFN** parameter and the inline form definition to tranform the data to IPDS.
  - When you print to a non-AFP printer, the **PAGDFN** parameter is ignored. The print file parameters are used, and the line data is transformed to IPDS or SCS.
- Specify DEVTYPE(*LINE), no PAGDFN, no FORMDF

- When you print to an AFP printer, an inline **PAGDFN** and **FORMDF** is built from the print file parameters. PSF/400 uses the inline page definition and inline form definition to transform the data to IPDS.
- When you print to a non-AFP printer, the print file parameters are used, and the line data is transformed to IPDS or SCS.

The support for combinations of **PAGDFN** and **FORMDF** for **DEVTYPE(*AFPDSLINE)** are similar to **DEVTYPE(*LINE)** line data. However, you should be aware of the following exceptions:

- When you send data to a non-AFP printer, the data can not be transformed to IPDS or to SCS. The spooled file must be printed on an AFP printer.
- Although you are not required to specify **PAGDFN** or **FORMDF** with ***AFPDSLINE** data [10], certain **AFPDS** commands in the data stream (for example, Invoke Medium Map) may reference named structured fields in the **PAGDFN** or **FORMDF** that may not match those in the inline **PAGDFN** and **FORMDF**.

The following parameters for line data can be changed with the **CHGSPLFA** command, after the data is spooled and before the spool file is printed:

- **AFPCHARS**
- **FORMDF**
- **PAGDFN**

OS/400 Printer File Parameters

The following table summarizes the print file keyword support provided when line data or mixed data is specified.

Support of OS/400 printer file parameters

When line data is redirected to a non-AFP printer, the attribute of the print file is used to print the file.

Note: Line data (***LINE**) specified with a page definition or mixed data can not be redirected.

The following table summarizes the support for print file keywords when Line data or Mixed data is specified.

Table 10. OS/400 Printer File Parameter Table

Print keyword	Line data or Mixed data specified
AFPCHARS	Is supported when printing to an AFP printer. Ignored when line data is redirected to a non-AFP printer.
ALIGN	Is supported
CTLCHAR	Is supported
AUT	Is supported

10. You are not required to specify **PAGDFN** or **FORMDF** with ***AFPDSLINE** data because it can be built inline from a print file.

Table 10. OS/400 Printer File Parameter Table (continued)

Print keyword	Line data or Mixed data specified
BACKMGN	Ignored when a FORMDF is specified and printing to an AFP printer. Margin offset information is specified in the FORMDF. Used when printing to an AFP printer and no FORMDF is specified. Inline FORMDF is built from print parameters. Valid only with ***AFPDS** data stream.
BACKOVL	Is supported. Ignored when print file (IPDS) is redirected to a non-AFP printer.
CDEFNT	Ignored when a PAGDFN is specified and printing to an AFP printer. Font is gotten from AFPCHARS parameter or is specified in PAGDFN. Used when printing to an AFP printer and no PAGDFN is specified. Inline PAGDFN is built from print parameters.
CHLVAL	Is ignored. The PAGDFN contains information for mapping channel numbers to line numbers.
CHRID	Ignored when a PAGDFN is specified and printing to an AFP printer. Code page is gotten from AFPCHARS parameter or is specified in PAGDFN. Used when printing to an AFP printer and no PAGDFN is specified. Inline PAGDFN is built from print parameters.
COPIES	Is supported
CORNERSTPL	Ignored when a FORMDEF is specified and pointing to an AFP printer. Used when printing to an AFP printer and no FORMDEF is specified.
CPI	Ignored when a PAGDFN is specified and printing to an AFP printer. Font is gotten from AFPCHARS parameter or is specified in PAGDFN Used when printing to an AFP printer and no PAGDFN is specified. Inline PAGDFN is built from print parameters.
CTLCHAR	Is supported
CVTCINDTA	Is supported. Line data can be converted to AFPDS before it is placed onto spool. A page definition must be specified. The DEVTYPE is changed to *AFPDS.
DEV	Is supported
DEVTYPE	*LINE and *AFPDSLINE only
DFRWRT	Is supported
DRAWER	Ignored when a FORMDF is specified and printing to an AFP printer. Drawer is gotten from FORMDF. Used when printing to an AFP printer and no FORMDF is specified. Inline FORMDF is built from print parameters (for SCS or IPDS data streams).

Table 10. OS/400 Printer File Parameter Table (continued)

Print keyword	Line data or Mixed data specified
DUPLEX	Ignored when a FORMDF is specified and printing to an AFP printer. Duplex is gotten from FORMDF. Used when printing to an AFP printer and no FORMDF is specified. Inline FORMDF is built from print parameters (for SCS or IPDS data streams).
EDGESTITCH	Ignored when a FORMDEF is specified and pointing to an AFP printer. Used when printing to an AFP printer and no FORMDEF is specified.
FIDELITY	Is supported when printing to an AFP printer. Ignored when line data is redirected to a non-AFP printer. Default is content fidelity.
FILE	Is supported
FILESEP	Is supported
FNTCHRSET	Ignored when a PAGDFN is specified and printing to an AFP printer. Font is gotten from AFPCHARS parameter or is specified in PAGDFN. Used when printing to an AFP printer and no PAGDFN is specified. Inline PAGDFN is built from print parameters.
FOLD	Is ignored if a PAGDFN or FORMDF is specified and printing to an AFP printer because when PAGDFN or FORMDF is specified, the page size and where data is positioned on a page is unknown.
FONT	Ignored when a PAGDFN is specified and printing to an AFP printer. Font is gotten from AFPCHARS parameter or is specified in PAGDFN. Used when printing to an AFP printer and no PAGDFN is specified. Inline PAGDFN is built from print parameters.
FORMDF	Is supported when printing to an AFP printer. Ignored when line data is redirected to a non-AFP printer.
FORMFEED	Ignored when a FORMDF is specified and printing to an AFP printer. FORMFEED is gotten from FORMDF. Used when printing to an AFP printer and no FORMDF is specified. Inline FORMDF is built from print parameters
FORMTYPE	Is supported
FRONTMGN	Ignored when a FORMDF is specified and printing to an AFP printer. Margin offset information is specified in the FORMDF. Used when printing to an AFP printer and no FORMDF is specified. Inline FORMDF is built from print parameters. Valid only with ***AFPDS** data stream.
FRONTOVL	Is supported. Ignored when print file (IPDS) is redirected to a non-AFP printer.
HOLD	Is supported

Table 10. OS/400 Printer File Parameter Table (continued)

Print keyword	Line data or Mixed data specified
LPI	Ignored when a PAGDFN is specified and printing to an AFP printer. Lines Per Inch is gotten from PAGDFN. Used when printing to an AFP printer and no PAGDFN is specified. Inline PAGDFN is built from print parameters.
LVLCHK	Ignored. Line data and Mixed data are not valid with a DDS print file
MAXRCDS	Is supported
MULTIUP	Ignored when a PAGDFN or FORMDF is specified and printing to an AFP printer. Multiup is specified in the FORMDF(N-UP) or can be achieved in the PAGDFN. Used when printing to an AFP printer and no FORMDF or PAGDFN is specified. Inline FORMDF and PAGDFN is built from print parameters.
OPTION	Is supported
OUTBIN	Is supported when printing to an AFP printer. Ignored when line data is redirected to a non-AFP printer
OUTPTY	Is supported
OUTQ	Is supported
OVRFLW	The overflow message will only be issued for Line data. It will not be issued for Mixed data.
PAGDFN	Is supported when printing to an AFP printer. Ignored when line data is redirected to a non-AFP printer.
PAGERANGE	Is supported
PAGESIZE	Ignored when a PAGDFN is specified and printing to an AFP printer. Page size is gotten from PAGDFN. Used when printing to an AFP printer and no PAGDFN is specified. Inline PAGDFN is built from print parameters.
PAGRTT	Ignored when a FORMDF is specified and printing to an AFP printer. Rotation is specified in FORMDF. Used when printing to an AFP printer and no FORMDF is specified. Inline FORMDF is built from print parameters (for SCS or IPDS data streams). IF PAGRTT(*COR) is specified, then this parameter is ignored unless both PAGDFN and FORMDF are not specified.
PRTQLTY	Ignored when a FORMDF is specified and printing to an AFP printer. Print quality is gotten from FORMDF. Used when printing to an AFP printer and no FORMDF is specified. Inline FORMDF is built from print parameters.
PRTTXT	If the file is opened *LINE and no PAGDFN is specified, print text will be supported. If the file is opened as *AFPDSLINE, print text will not be supported.

Table 10. OS/400 Printer File Parameter Table (continued)

Print keyword	Line data or Mixed data specified
REDUCE	Ignored when a PAGDFN or FORMDF is specified and printing to an AFP printer. Multiup is specified in the FORMDF(N-UP) or can be achieved in the PAGDFN. Used when printing to an AFP printer and no FORMDF or PAGDFN is specified. Inline FORMDF and PAGDFN is built from print parameters. REDUCE(*TEXT) is supported when printing to a non-AFP printer.
REPLACE	Is supported
RPLUNPRT	Is supported
SADLSTITCH	Ignored when a FORMDEF is specified and pointing to an AFP printer. Used when printing to an AFP printer and no FORMDEF is specified.
SAVE	Is supported
SCHEDULE	Is supported
SCRFILE	Ignored. Line data and mixed data are not valid with a DDS print file.
SCRMBR	Ignored. Line data and mixed data are not valid with a DDS print file.
SHARE	Is supported
SPOOL	*YES only. Direct IO (SPOOL(*NO)) is not supported.
TBLREFCHR	Is supported when printing to an AFP printer. Ignored when line data is redirected to a non-AFP printer (IPDS).
TEXT	Is supported
UOM	Is supported
USRDFNDTA	Is supported when printing to an AFP printer.
USRDFNOBJ	Is supported when printing to an AFP printer.
USRDFNOPT	Is supported when printing to an AFP printer.
USRDTA	Is supported
WAITFILE	Is supported

Carriage Control (CC) Characters

The carriage control character can be represented as either ANSI or machine code.

ANSI carriage control is a standard representation that is used with printers from many different manufacturers. Table 11 on page 334 lists the ANSI codes and their functions. Machine code control characters were defined by IBM; they correspond to channel command words issued by the operating system. Table 12 on page 334 lists the IBM machine code values and functions.

Note: You may not use both ANSI and machine codes within a single data set.

There are differences in the conventions used by OS/400 for ANSI and machine code line spacing. The OS/400 convention for ANSI handles line spacing and then

causes the line to be printed. The OS/400 convention for machine codes causes the line to be printed and then the spacing action is performed.

ANSI Carriage Control Characters

Table 11. ANSI Carriage Control Characters

Hexadecimal Value	Character	Function
X'40'	(space)	Space 1 line, then print (single spacing)
X'F0'	0	Space 2 lines, then print (double spacing)
X'60'	–	Space 3 lines, then print (triple spacing)
X'4E'	+	Suppress spacing, then print (overstrike previous line)
X'F1'	1	Print the data at line position defined as Channel 1 (by convention, the first line on a new page)
X'F2'	2	Print the data at the line position defined as Channel 2
X'F3'	3	Print the data at the line position defined as Channel 3
X'F4'	4	Print the data at the line position defined as Channel 4
X'F5'	5	Print the data at the line position defined as Channel 5
X'F6'	6	Print the data at the line position defined as Channel 6
X'F7'	7	Print the data at the line position defined as Channel 7
X'F8'	8	Print the data at the line position defined as Channel 8
X'F9'	9	Print the data at the line position defined as Channel 9
X'C1'	A	Print the data at the line position defined as Channel 10
X'C2'	B	Print the data at the line position defined as Channel 11
X'C3'	C	Print the data at the line position defined as Channel 12
Note: When ANSI carriage controls are used, only the values that appear in this table are considered valid by PSF/400. PSF/400 treats any other ANSI carriage control value as invalid and prints any data on the line using single spacing.		

Machine Carriage Control Characters

Table 12. Machine Code Control Characters

Control Character Value (in hexadecimal)	Function
X'03'	No operation
X'09'	Print and space 1 line (single spacing)
X'11'	Print and space 2 lines (double spacing)

Table 12. Machine Code Control Characters (continued)

Control Character Value (in hexadecimal)	Function
X'19'	Print and space 3 lines (triple spacing)
X'01'	Print without spacing (overstrike next line)
X'89'	Print the data, then skip to the line position defined as Channel 1 (by convention, the first line on a new page)
X'91'	Print the data, then skip to the line position defined as Channel 2
X'99'	Print the data, then skip to the line position defined as Channel 3
X'A1'	Print the data, then skip to the line position defined as Channel 4
X'A9'	Print the data, then skip to the line position defined as Channel 5
X'B1'	Print the data, then skip to the line position defined as Channel 6
X'B9'	Print the data, then skip to the line position defined as Channel 7
X'C1'	Print the data, then skip to the line position defined as Channel 8
X'C9'	Print the data, then skip to the line position defined as Channel 9
X'D1'	Print the data, then skip to the line position defined as Channel 10
X'D9'	Print the data, then skip to the line position defined as Channel 11
X'E1'	Print the data, then skip to the line position defined as Channel 12
X'0B'	Space 1 line without printing
X'13'	Space 2 lines without printing
X'1B'	Space 3 lines without printing
X'8B'	Skip to Channel 1 immediate (by convention, the first line on a new page)
X'93'	Skip to the Channel 2 position immediate
X'9B'	Skip to the Channel 3 position immediate
X'A3'	Skip to the Channel 4 position immediate
X'AB'	Skip to the Channel 5 position immediate
X'B3'	Skip to the Channel 6 position immediate
X'BB'	Skip to the Channel 7 position immediate
X'C3'	Skip to the Channel 8 position immediate
X'CB'	Skip to the Channel 9 position immediate
X'D3'	Skip to the Channel 10 position immediate
X'DB'	Skip to the Channel 11 position immediate
X'E3'	Skip to the Channel 12 position immediate
Note: PSF/400 ignores the following hexadecimal machine-code carriage control characters and does not print lines containing them: X'02' through X'07', X'0A', X'12', X'23', X'43', X'63', X'6B', X'73', X'7B', X'EB', X'F3', and X'FB'. PSF/400 treats any other carriage control value as invalid and prints any data on the line using single spacing.	

Table Reference Characters (TRC)

Table Reference Characters (TRCs) allow an additional byte to appear at the beginning of a line to indicate which one of up to four different character arrangement tables (coded fonts specified by AFPCHARS parameter) will be used to print the line. This byte, the *table reference character* contains a value of X'F0',

X'F1', X'F2', or X'F3', corresponding to the relative position of the desired coded font in the list of coded fonts specified by the AFPCHARS parameter. If carriage control bytes are used with the data, the table reference character follows the carriage control byte but precedes the data bytes. If a carriage control bytes are not used, the table reference character is the first byte of the data record. As with carriage control, if table reference characters are used, every data record must contain a TRC byte.

Figure 112 summarizes the valid forms of line data.

Note: The TRC is used for traditional (not record formatting) processing only.

D A T A

A. Simple data line

CC	D A T A

B. Data line with carriage control byte

TRC	D A T A

C. Data line with table reference character

CC	TRC	D A T A

D. Data line with carriage control byte and table reference character
Figure 112. Valid Line Data Records

IGC Parameters

The IGC (Ideo Graphic Characters) parameters of an OS/400 printer file are described here.

IGCDTA

Indicates double byte character set (DBCS) data may be used in the file. The user for a line or mixed data file will need to indicate that there is SO/SI present in the data by setting IGCDTA to *YES.

IGCCPI

For AFP printers, this parameter is ignored, as the pitch of the DBCS data is determined by the selected font.

For non-AFP printers, when line data is transformed to SCS, this parameter is used to specify the pitch of the DBCS data. DBCS SO/SI can not be transformed when going to an IPDS printer.

Mixed data can not be transformed when going to a SCS or IPDS printer.

IGCSOSI

This keyword indicates what action should be taken when SO/SI are found in the data. If the data is mixed, the SO/SI should be taken out and appropriate spaces inserted based on the value of this keyword.

*YES

The SO/SI characters will be printed as blanks.

*NO

The system does not print the shift control characters. These characters do not occupy a position on the printed output.

*RIGHT

The system prints two blanks when printing the shift-in characters but does not print shift-out characters.

IGCEXNCHR
Ignored, as extension character processing only applies to SCS DBCS printer, not AFP attached printers.

IGCCHRRTT
For AFP printers, this parameter is ignored. Character rotation can be specified in the PAGDFN.

For non-AFP printers, when line data is transformed to SCS, this parameter is used to rotate the DBCS data.

INVMMAP (Medium-Map-Name) DDS Keyword

INVMMAP is a record level keyword in DDS used to invoke a medium map. Invoke Medium Map (IMM) specifies the name of a medium map in a form definition. Use the **IMM** in the form definition to select or change print parameters such as input drawer, page rotation, overlays.

The medium map name is limited to 8 characters. You can specify the medium map name as a constant or a program-to-system field.
- medium-map-name
- field1

The **INVMMAP** keyword is valid only with **DEVTYPE(*AFPDS)**. Also, a form definition must be specified on the print file. If **DEVTYPE** is changed to anything other than ***AFPDS**, the **INVMMAP** keyword is ignored and a warning message will be issued at print time.

PSF/400 ends printing on the current sheet when a invoke medium map is encountered.

INVMMAP, **SKIP**, and **SPACE** keywordsare processed in the following order:
- **SKIPB**
- **SPACEB**
- **INVMMAP**
- **SPACEA**
- **SKIPA**

The medium map specified remains in effect for the rest of the file unless changed by another **INVMMAP** keyword.

The invoke medium map keyword is validated at print time. An error message will be issued if it is not valid.

Option indicators are valid for the **INVMMAP** keyword.

Figure 113 on page 338 shows how to specify the **INVMMAP** keyword.

```
0        1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
      *
                R RECORD1
           02                                 INVMMAP(MAP1)
                R RECORD2                     INVMMAP(&MAP)
                  MAP            8A  P

      *
```

Figure 113. Specifying the INVMMAP keyword

If indicator 02 is on, RECORD1 uses a new medium map (MAP1).

RECORD2 allows the application program to specify the name of medium map by setting program variable MAP.

INVDTAMAP (invoke data map) keyword

INVDTAMAP is a record-level keyword used to invoke a new data map. It specifies the name of the data map in a page definition. The page definition is then used to map the line data. Data maps in page definitions can perform functions such as including multiple-up or rotated printing, changing fonts, and lines per inch. You must have PSF/400 installed in order to use this keyword.

The format of the keyword is:

```
INVDTAMAP{data-map-name | &data-map-name-field}
```

The data-map-name parameter is required and defines a data map in the page definition. This parameter is 8 characters in length. The data map name can either be specified as a constant or a program-to-system field.

This keyword is valid with DEVTYPE(*LINE) or DEVTYPE(*AFPDSLINE). A page definition must be specified on the print file. If DEVTYPE is changed to anything other than *LINE or *AFPDSLINE, the keyword will be ignored and a warning message will be issued at print time.

The INVDTAMAP, SKIP, and SPACE keywords are processed in the following order:

- SKIPB
- SPACEB
- INVDTAMAP
- SPACEA
- SKIPA

The following example shows how to specify the INVDTAMAP keyword:

```
0        1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
*
                R RECORD1
           02                                 INDTAMAP(MAP1)
                R RECORD2                     INDTAMAP(&MAP);
                  MAP            8A  P

*
```

Restrictions When Using PAGDFN and FORMDF

Device type ***LINE** or ***AFPDSLINE** and **SPOOL(*NO)** will not be allowed on the **CRTPRTF** or **CHGPRTF** commands. If this condition exists, Message **CPD7341**, indicating the value **SPOOL(*NO)** is not valid with device type, will be issued.

Today when native application output is printed, an inline **FORMDF** is generated by the OS/400 which utilizes the printer file parameters for drawer, duplex, outbin, etc. When users specify a form definition on their printer file, this will no longer be done. The form definition will be used for all media handling and the printer file parameters for this information will be ignored. The first media map in the **FORMDF** will be used unless the data itself calls out a different media map (such as with the new DDS keyword).

Native applications which write normal control list-type data (such as RPG and COBOL applications) without utilizing the CTLCHAR parameter of the printer file, will generate *MACHINE control character line data. There will be no support for generating ANSI control character line data except by the application actually writing that type of data using CTLCHAR(*FCFC).

CVTPPFASRC Command

The **CVTPPFASRC** command creates a data base file member from the information contained in the library and file containing the PPFA source. The PPFA source file determines the name of the output data base file members that is used as input to the CRTFORMDF or CRTPAGDFN command.

If you want to create an object of type *FORMDF, use the CRTFORMDF command.

If you want to create an object of type *PAGDFN, use the CRTPAGDFN command.

Notes:

1. Form definitions and page definitions are limited to AFP printing with PSF/400.
2. Those interested in generating page and form definitions for use with the CRTAFPDTA command (this command is part of InfoPrint Server/400) should refer to the *InfoPrint Server for OS/400: User's Guide*.

Syntax

OS/400 is path-dependent; therefore, sequence and completeness of syntax is very important. The CVTPPFASRC command has two required parameters (FILE and MBR) and four subcommands (FORMDFFILE, PAGDFNFILE, OUTPUT, and OPTION).

```
CVTPPFASRC
  FILE (*LIBL / input-PPFA-source-file-name
    | *CURLIB / input-PPFA-source-file-name
    | library-name  / input-PPFA-source-file-name)

  MBR ( input-member -name )

  [FORMDFFILE (*NONE
    | *LIBL / form-definition-file
    | *CURLIB / form-definition-file
    | library-name / form-definition-file ) ]

  [PAGEDFFILE (*NONE
    | *LIBL / page-definition-file
    | *CURLIB / page-definition-file
    | library-name / page-definition-file ) ]

  [OUTPUT (*PRINT
    | *NONE ) ]

  [OPTION ( { *SRC | *NOSRC }
    { *SECLVL | *NOSECLVL } ) ]
```

Figure 114. Specifying the CVTPPFASRC command

Note: ***NONE** may be specified for either **FORMDFFILE** or **PAGDFNFILE**, but not for both.

Subcommands and Parameters

The Convert PPFA Source (CVTPPFASRC) command creates a data base file member from the information contained in the PPFA source file. The data base file member can then be used either as input to the CRTFORMDF command to create an object of type *FORMDF or as input to the CRTPAGDFN command to create an object of type *PAGDFN.

The following parameters are required:

FILE Specifies the location of the PPFA source file and library. The possible library values are:

***LIBL** Specifies the PPFA source file is in the library list.

***CURLIB**
Specifies the current library for the job is searched. If no library is specified as the current library for the job, the QGPL library is used for PPFA source.

library name
Specifies the PPFA source file is in a specific library

File Name
File name of the PPFA source file

MBR Specifies the PPFA source file member.

Member Name
Specifies the PPFA source member.

The following parameters are optional:

FORMDEFILE

Specifies the output target file and library to which the compiled form definition member is placed. This is a flat file, and not yet an OS/400 object.

The member name will be the characters ″F1″ followed by the FORMDEF name as coded on the FORMDEF command in the PPFA source member. This output member must then be run as input to the CRTFORMDF command in order to create the form definition object. This will then be the OS/400 object.

***LIBL** The PPFA form definition output file is in the library list.

***CURLIB**

The current library for the job is searched. If no library is specified as the current library for the job, the QGPL library is used for PPFA form definition output.

***FILE** The PPFA form definition output is the same as the source file.

library name

The PPFA form definition output is to go to a specific library.

File Name

The file name in which to place the form definition output of the PPFA.

PAGDFNFILE

Specifies the output target file and library to which the compiled page definition member is placed. This is a flat file, and not yet an OS/400 object.

The member name will be the characters ″P1″ followed by the PAGEDEF name as coded on the PAGEDEF command in the PPFA source member. This output member must then be run as input to the CRTPAGDEN command in order to create a page definition object of type *PAGDFN. This will then be the OS/400 object.

***LIBL** The PPFA page definition output file is in the library list.

***CURLIB**

The current library for the job is searched. If no library is specified as the current library for the job, the QGPL library is used for PPFA page definition output.

***FILE** The PPFA page definition output is the same as the source file.

library name

The PPFA page definition output is to go to a specific library.

File Name

The file name in which to place the page definition output of the PPFA.

OUTPUT

Specifies the destination of the listing file and error messages that PPFA produces during compilation.

***PRINT**

The listing file is sent to the system queue for printing.

***NONE**
The listing file is not generated.

OPTION
Specifies the type of output produced when the file is created.

***SRC** A printout of the source statements, including a list of errors is created.

***NOSRC**
No printout of the source statements is created unless errors are detected. If errors are detected, they are listed along with the sequence number that caused the error.

***SECLVL**
The online help information appears in the PPFA printout.

***NOSECLVL**
The messages section of the PPFA printout does not contain the online help information for messages issued during PPFA processing.

Windows NT and Windows 2000 Environment

The **ppfa** command creates form definitions and page definitions on the NT operating system. After they are created, you can transfer the form definitions and page definitions to other operating systems (such as OS/390, VM, or VSE) to use as AFP resources.

Syntax

ppfa [**-f***path.ext*] [**-p***path.ext*] [**-s***path.ext*] [**-x**] *inputfile*

Flags and Values

You can specify these flags and values with the **ppfa** command.

inputfile
The file containing the PPFA source statements to be "processed".

-f*path.ext*
Add path and extension information to the names of form definitions generated by PPFA. (The *name* itself will come from the FORMDEF command.)

-p*path.ext*
Add path and extension information to the names of page definitions generated by PPFA. (The *name* itself will come from the PAGEDEF command.)

-s*path.ext*
Add path and extension information to the listing file. The *name* of the listing file will be the same as the *name* of the inputfile.

Thus, for "FORMDEF name" when PPFA was invoked with:
ppfa -fpath.ext infile

it generates form definition:
\path\F1NAME.ext

Also, for "PAGEDEF name" when PPFA was invoked with:
ppfa -p\root\abc\def.xyz.nnn infile

it generates page definition:
\root\abc\def.xyz\P1NAME.nnn

In another example, if you enter:
`ppfa -pabc\def.xyz` *input.file*

and it has a PAGEDEF statement in the source, then the page definition created will be either:
abc\def\P1NAME.xyz or
.\abc\def\P1NAME.xyz

However, if you enter:
`ppfa -p\abc\def.xyz` *input.file*

PPFA generates the file:
\abc\def\P1NAME.xyz, not
.\abc\def\P1NAME.xyz

-x Causes **ppfa** to interpret information found in columns 1-72 of the *inputfile*. The information in the rest of the columns will be ignored. This is useful if you are downloading a Fixed-80 file from the host.

Examples

1. To create a form definition from an input file called **johnb** in the current library containing the PPFA source statements, enter:
   ```
   ppfa johnb
   ```
   The generated form definition is stored in the current library.
2. To create a form definition from an input file called **maryc** containing the PPFA source statements, and then storing the generated form definition in the **/usr/lpp/resources** library, enter:
   ```
   ppfa -f\usr\lpp\resources maryc
   ```

Appendix B. More about Direction

In PPFA, directions specified with the PRINTLINE and TRCREF commands are relative to the direction specified in the PAGEFORMAT command. If no PAGEFORMAT command has been specified, the direction specified in the PAGEDEF command is used. If no direction has been specified in either of these commands, the default direction for the page format is ACROSS.

The PRINTLINE and TRCREF commands *add* their DIRECTION values to the DIRECTION value specified with the PAGEFORMAT command. Thus, you may select a PAGEFORMAT direction and code PRINTLINEs and TRCREFs relative to the PAGEFORMAT direction. For more information about the PRINTLINE and TRCREF commands, see “Chapter 3. Using Page Definition Commands for Traditional Line Data” on page 33.

For instance, if a page is to be printed in the landscape page presentation on a printer that requires the DOWN or UP print direction to generate landscape output, the PAGEFORMAT command can specify DOWN as its DIRECTION. Once this direction is established, you can view the page as a landscape page and specify the PRINTLINE and the TRCREF commands with the ACROSS direction. Output specified in this way prints ACROSS relative to the landscape page, as shown in Figure 115.

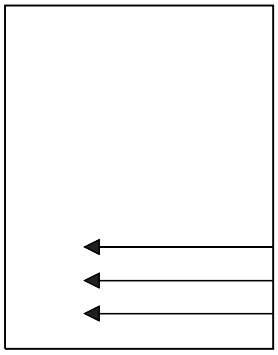


Figure 115. Printing Across a Landscape Page

Note that if you specify the DOWN direction for the PRINTLINE or the TRCREF command in this case, the output looks like Figure 116 on page 346 because the direction of the page format is also DOWN.

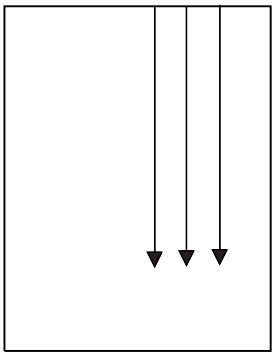


Figure 116. Printing Down a Portrait Page

Table 13 shows the final result when all of the possible combinations of DIRECTION are specified. The final direction that PPFA computes from the PAGEFORMAT, PRINTLINE, and TRCREF commands determines the prefix assigned to the font names specified in the page definition. The final direction is particularly important when printing on the 3800 printer because its unbounded-box font architecture requires a separate font for each combination of print direction and character rotation. This information is encoded in the prefix of the font name (X1, X3, XA, and XF, for example).

Table 13. The Effect of Additive DIRECTIONs on Formatting and Font Prefixes

Page Format	**PRINTLINE or TRCREF**	**Final Result**	**3800 Font Prefix**			
			0°	90°	180°	270°
Across	Across	Across	X1	X5	X9	XD
Across	Down	Down	X2	X6	XA	XE
Across	Back	Back	X3	X7	XB	XF
Across	Up	Up	X4	X8	XC	XG
Down	Across	Down	X2	X6	XA	XE
Down	Down	Back	X3	X7	XB	XF
Down	Back	Up	X4	X8	XC	XG
Down	Up	Across	X1	X5	X9	XD
Back	Across	Back	X3	X7	XB	XF
Back	Down	Up	X4	X8	XC	XG
Back	Back	Across	X1	X5	X9	XD
Back	Up	Down	X2	X6	XA	XE
Up	Across	Up	X4	X8	XC	XG
Up	Down	Across	X1	X5	X9	XD
Up	Back	Down	X2	X6	XA	XE
Up	Up	Back	X3	X7	XB	XF

The entries in the **Final Result** column can be computed using a simple algorithm. If you assume that ACROSS is 0, DOWN is 1, BACK is 2, and UP is 3, you can add the direction specifications in the two commands, subtracting 4 when the result is 4 or greater, to compute the final direction.

Appendix C. Differences in Measurements and REPEATs with AFP Utilities

When repeating a DRAWRULE (OGL), PRINTLINE (PPFA), DRAWGRAPHIC (PPFA), or "Line" (PMF), there are differences in the measurements of the repeated lines. For OGL, REPEAT indicates the number of repetitions *in addition to* the first. For DRAWGRAPHIC (PPFA), REPEAT is the same as OGL. Therefore, REPEAT 1 yields 2 DRAWRULEs. For PPFA, REPEAT indicates the total number of PRINTLINEs. Therefore, REPEAT 2 yields 2 PRINTLINEs.

Another difference occurs when the linespacing (set by SETUNITS in OGL and PPFA, and by a screen item in PMF) results in the distance from one line to the next not being a whole number of pels. Each product handles the fractional pel differently. Because the printer cannot print parts of a pel, fractional pels cannot be represented at the printer. When linespacing calculations result in a fractional pel per linespace, the following occurs:

OGL Carries the fractions until they add up to a whole pel, then adds it in. This results in the final spot of a repeat being within a pel of where it is expected. Therefore, not all of the spaces between rules are even; they can vary by one pel.

PPFA Truncates the fractional pel prior to the repeat. Therefore, the spaces between the lines are even, but the total might be shorter than expected.

PMF Rounds the fractional pel prior to the repeat. Therefore, the spaces between the lines are even, but the total might be shorter or longer than expected. If the fractional pel is less than 0.5, it is handled the same as PPFA and the linespace will be shorter. If the fractional pel is greater than or equal to 0.5, the linespace will be longer.

Use linespacing in all products that result in a whole number of pels. To resolve existing problems, select the resource that you don't want to change, and code the remaining resource without using REPEAT because of the way the other products handle the fractional pels.

For example, if you want to print at 9 lines per inch, and repeat this for 20 lines, the following will occur. Starting at zero, and adding 9 lines per inch (converted to pels this is 240/9 = 26.6670), you will see the results illustrated in Table 14.

Table 14. Differences in Measurements and REPEATs with AFP Utilities

	Mathematics		OGL		PPFA		PMF	
Repetition	**Position**	**FromLast**	**Position**	**FromLast**	**Position**	**FromLast**	**Position**	**FromLast**
	0.000	-.---	0	---	0	---	0	---
1	26.667	26.667	26	26	26	26	27	27
2	53.333	26.667	53	27	52	26	54	27
3	80.000	26.667	80	27	78	26	81	27
4	106.667	26.667	106	26	104	26	108	27
5	133.333	26.667	133	27	130	26	135	27
6	160.000	26.667	160	27	156	26	162	27
7	186.667	26.667	186	26	182	26	189	27

Table 14. Differences in Measurements and REPEATs with AFP Utilities (continued)

	Mathematics		OGL		PPFA		PMF	
Repetition	**Position**	**FromLast**	**Position**	**FromLast**	**Position**	**FromLast**	**Position**	**FromLast**
8	213.333	26.667	213	27	208	26	216	27
9	240.000	26.667	240	27	234	26	243	27
10	266.667	26.667	266	26	260	26	270	27
11	293.333	26.667	293	27	286	26	297	27
12	320.000	26.667	320	27	312	26	324	27
13	346.667	26.667	346	26	338	26	351	27
14	373.333	26.667	373	27	364	26	378	27
15	400.000	26.667	400	27	390	26	405	27
16	426.667	26.667	426	26	416	26	432	27
17	453.333	26.667	453	27	442	26	459	27
18	480.000	26.667	480	27	468	26	486	27
19	506.667	26.667	506	26	494	26	513	27
20	533.333	26.667	533	27	520	26	540	27

To resolve differences in how OGL, PPFA, and PMF handle repeated values, one of the following approaches may be taken:

- Don't use REPEAT

or

- Code units as PEL(s)

  Note that in all of these products (except PPFA), a PEL is 1/240 of an inch. For PPFA, the PEL size can be set by the user, but defaults to 1/240 of an inch.

Appendix D. More About Bar Code Parameters

This section contains supplemental information about Bar Code Object Content Architecture (BCOCA) specified by the **BARCODE** subcommand of the **FIELD** command, and includes the following topics:

- Bar code data
- MOD parameter

For more complete information, refer to *Data Stream and Object Architectures: Bar Code Object Content Architecture Reference* (S544-3766).

Bar Code Data

The data is specified as a series of single-byte code points from a specific code page. Some symbologies limit the valid code points to just the ten numerals (0 through 9), other symbologies allow a richer set of code points. The bar code symbol is produced from these code points; the code points are also used, along with a particular type style, when producing the HRI.

Table 15 on page 350 lists, for each symbology, the valid code page from which characters are chosen and the type style used when printing HRI in terms of an IBM registered CPGID and FGID. More information about these values can be found in *IBM Advanced Function Presentation Fonts: Font Summary* and in *IBM Advanced Function Presentation: Technical Reference for Code Pages*.

Table 15. Valid Code Pages and Type Styles

Type	Bar Code Symbology	EBCDIC-Based CPGID	FGID
1	Code 39 (3-of-9 Code), AIM USS-39	500	Device specific
2	MSI (modified Plessey code)	500	Device specific
3	UPC/CGPC — Version A	893	3 (OCR-B)
5	UPC/CGPC — Version E	893	3 (OCR-B)
6	UPC — Two-digit Supplemental (Periodicals)	893	3 (OCR-B)
7	UPC — Five-digit Supplemental (Paperbacks)	893	3 (OCR-B)
8	EAN-8 (includes JAN-short)	893	3 (OCR-B)
9	EAN-13 (includes JAN-standard)	893	3 (OCR-B)
10	Industrial 2-of-5	500	Device specific
11	Matrix 2-of-5	500	Device specific
12	Interleaved 2-of-5, AIM USS-I 2/5	500	Device specific
13	Codabar, 2-of-7, AIM USS-Codabar	500	Device specific
17	Code 128, AIM USS-128	1303	Device specific
22	EAN Two-digit Supplemental	893	3 (OCR-B)
23	EAN Five-digit Supplemental	893	3 (OCR-B)
24	POSTNET	500	None
26	RM4SCC	500	None
27	JPOSTAL	500	None
31	APOSTAL	500	Device Specific

As shown in Table 15, the font used to print HRI depends on the symbology. Some symbologies use OCR-B; others use a device-specific font (usually OCR-A).

Table 16 lists the valid characters for each symbology and specifies how many characters are allowed for a bar code symbol.

Table 16. Valid Characters and Data Lengths

Code	Bar Code Type	Valid Characters	Valid Data Length
X'01'	Code 39 (3-of-9 Code), AIM USS-39	0123456789 ABCDEFGHIJKLM NOPQRSTUVWXYZ -.$/+% and the space character A total of 43 valid characters.	Symbology: unlimited BCOCA range: 0 to 50 characters (see note 1 on page 354)

Table 16. Valid Characters and Data Lengths (continued)

Code	Bar Code Type	Valid Characters	Valid Data Length
X'02'	MSI (modified Plessey code)	0123456789	3 to 15 characters for Modifier X'X'01'' 2 to 14 characters for Modifier X'X'02'' 1 to 13 characters for all other modifiers
X'03'	UPC/CGPC - Version A	0123456789	11 characters
X'05'	UPC/CGPC - Version E	0123456789	10 characters
X'06'	UPC - Two-digit Supplemental (Periodicals)	0123456789	2 characters for Modifier X'00' 13 characters for Modifier X'01' 12 characters for Modifier X'02'
X'07'	UPC - Five-digit Supplemental (Paperbacks)	0123456789	5 characters for Modifier X'00' 16 characters for Modifier X'01' 15 characters for Modifier X'02'
X'08'	EAN-8 (includes JAN-short)	0123456789	7 characters
X'09'	EAN-13 (includes JAN-standard)	0123456789	12 characters
X'0A'	Industrial 2-of-5	0123456789	Symbology: unlimited BCOCA range: 0 to 50 characters (see note 1 on page 354)
X'0B'	Matrix 2-of-5	0123456789	Symbology: unlimited BCOCA range: 0 to 50 characters (see note 1 on page 354)
X'0C'	Interleaved 2-of-5, AIM USS-I 2/5	0123456789	Symbology: unlimited BCOCA range: 0 to 50 characters (see note 1 on page 354)
X'0D'	Codabar, 2-of-7, AIM USS-Codabar	0123456789 -$:/.+ABCD 16 characters plus 4 start/stop characters (ABCD) (see note 2 on page 354	Symbology: unlimited BCOCA range: 0 to 50 characters (see note 1 on page 354)

Table 16. Valid Characters and Data Lengths (continued)

Code	Bar Code Type	Valid Characters	Valid Data Length
X'11'	Code 128, AIM USS-128	All characters defined in the Code 128 code page	Symbology: unlimited BCOCA range: 0 to 50 characters (see note 1 on page 354)
X'16'	EAN Two-digit Supplemental	0123456789	2 characters for Modifier X'00' 14 characters for Modifier X'01'
X'17'	EAN Five-digit Supplemental	0123456789	5 characters for Modifier X'00' 17 characters for Modifier X'01'
X'18'	POSTNET	0123456789	5 characters for Modifier X'00' 9 characters for Modifier X'01' 11 characters for Modifier X'02' BCOCA range for Modifier X'03': 0 to 50 characters (see note 1 on page 354)
X'1A'	Royal Mail (RM4SCC modifier X'00') See note	0123456789 ABCDEFGHIJKLM NOPQRSTUVWXYZ	Symbology: unlimited BCOCA range: 0 to 50 characters (see note 1 on page 354)
	Royal Mail (Dutch KIX variation, modifier X'01')	0123456789 ABCDEFGHIJKLM NOPQRSTUVWXYZ abcdefghijklmnopq rstuvwxyz	Symbology: unlimited BCOCA range: 0 to 50 characters (see note 1 on page 354)
X'1B'	Japan Postal Bar Code (modifier X'00')	0123456789 ABCDEFGHIJKLM NOPQRSTUVWXYZ - (hyphen)	Symbology: 7 or more modifier X'00': BCOCA range: 7 to 50 characters
	Japan Postal Bar Code (modifier X'01')	0123456789 CC1,CC2,CC3,CC4, CC5,CC6,CC7,CC8 - (hyphen),start,stop	BCOCA range: 23 characters; refer to the modifier X'01' description

Table 16. Valid Characters and Data Lengths (continued)

Code	Bar Code Type	Valid Characters	Valid Data Length
X'1F'	Australia Post Bar Code - refer to the modifier description to see which characters are valid in specific parts of the symbol.		
	Modifier X'01' - Standard Customer Barcode	0123456789	Symbology: 8 digits BCOCA range: 8 digits
	Modifier X'02' - Customer Barcode 2 using Table N	0123456789	Symbology: 8 - 16 digits BCOCA range: 8 - 16 digits
	Modifier X'03' - Customer Barcode 2 using Table C	0123456789 ABCDEFGHIJKLM NOPQRSTUVWXYZ abcdefghijklmnopq rstuvwxyz (space) # (number sign)	Symbology: 8 - 13 characters BCOCA range: 8 - 13 characters
	Modifier X'04' - Customer Barcode 2 using proprietary encoding	0123456789 for sorting code 0-3 for customer information	Symbology: 8 - 24 digits BCOCA range: 8 - 24 digits
	Modifier X'05' - Customer Barcode 3 using Table N	0123456789	Symbology: 8 - 23 digits BCOCA range: 8 - 23 digits
	Modifier X'06' - Customer Barcode 3 using Table C	0123456789 ABCDEFGHIJKLM NOPQRSTUVWXYZ abcdefghijklmnopq rstuvwxyz (space) # (number sign)	Symbology: 8 - 18 characters BCOCA range: 8 - 18 characters
	Modifier X'07' - Customer Barcode 3 using proprietary encoding	0123456789 for sorting code 0-3 for customer information	Symbology: 8 - 39 digits BCOCA range: 8 - 39 digits
	Modifier X'08' - Reply Paid Barcode	0123456789	Symbology: 8 digits BCOCA range: 8 digits

Table 16. Valid Characters and Data Lengths (continued)

Code	Bar Code Type	Valid Characters	Valid Data Length

Notes:

1. All BCOCA receivers must support at least the BCOCA range. Some receivers support a larger data length.
2. Some descriptions of Codaba show the characters ″T,N,*,E″ as stop characters (representing the stop characters ″A,B,C,D″), but the Codabar symbology actually only allows ″A,B,C,D″ as start and stop characters. This alternate representation (″T,N,*,E″) is used only to distinguish between the start and stop characters when describing a Codabar symbol; when coding a BCOCA Codabar symbol, start and stop characters must be represented using A, B, C, or D.
3. The data for the UPC and EAN symbologies is numeric and of a fixed length, but not all numbers of the appropriate length are valid. This is because the coding scheme is designed to uniquely identify both a product and its manufacturer. The first part of the symbol represents the manufacturer and is defined in the symbology specification (not all numbers are valid in this part of the symbol). The second part of the symbol represents a unique product identifier code assigned by the manufacturer. Refer to the appropriate symbology specification for more details.
4. See RM4SCC on page 363 for additional information.

Table 17. Characters and Code Points used in the BCOCA Symbologies; Excluding Code 128

Character	EBCDIC Code Point
0	X'F0'
1	X'F1'
2	X'F2'
3	X'F3'
4	X'F4'
5	X'F5'
6	X'F6'
7	X'F7'
8	X'F8'
9	X'F9'
A	X'C1'
B	X'C2'
C	X'C3'
D	X'C4'
E	X'C5'
F	X'C6'
G	X'C7'
H	X'C8'
I	X'C9'
J	X'D1'
K	X'D2'
L	X'D3'
M	X'D4'

Table 17. Characters and Code Points used in the BCOCA Symbologies; Excluding Code 128 (continued)

Character	EBCDIC Code Point
N	X'D5'
O	X'D6'
P	X'D7'
Q	X'D8'
R	X'D9'
S	X'E2'
T	X'E3'
U	X'E4'
V	X'E5'
W	X'E6'
X	X'E7'
Y	X'E8'
Z	X'E9'
a	X'81'
b	X'82'
c	X'83'
d	X'84'
e	X'85'
f	X'86'
g	X'87'
h	X'88'
i	X'89'
j	X'91'
k	X'92'
l	X'93'
m	X'94'
n	X'95'
o	X'96'
p	X'97'
q	X'98'
r	X'99'
s	X'A2'
t	X'A3'
u	X'A4'
v	X'A5'
w	X'A6'
x	X'A7'
y	X'A8'
z	X'A9'

Table 17. Characters and Code Points used in the BCOCA Symbologies; Excluding Code 128 (continued)

Character	EBCDIC Code Point
- (hyphen)	X'60'
# (number sign)	X'7B'
.	X'4B'
$	X'5B'
/	X'61'
+	X'4E'
%	X'6C'
:	X'7A'
Space	X'40'

The Code 128 code page (CPGID = 1303) is defined as shown in Figure 117 on page 357.

Hex DIGITS 1st → 2nd ↓	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	NUL SE010000	DLE SE170000			(SP) SP010000	& SM030000	– SP100000					^ SD150000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	SOH SE020000	DC1 SE180000					/ SP120000		a LA010000	j LJ010000	~ SD190000		A LA020000	J LJ020000		1 ND010000
-2	STX SE030000	DC2 SE190000	FS SE350000	SYN SE230000					b LB010000	k LK010000	s LS010000		B LB020000	K LK020000	S LS020000	2 ND020000
-3	ETX SE040000	DC3 SE200000							c LC010000	l LL010000	t LT010000		C LC020000	L LL020000	T LT020000	3 ND030000
-4									d LD010000	m LM010000	u LU010000		D LD020000	M LM020000	U LU020000	4 ND040000
-5	HT SE100000		LF SE110000						e LE010000	n LN010000	v LV010000		E LE020000	N LN020000	V LV020000	5 ND050000
-6		BS SE090000	ETB SE240000						f LF010000	o LO010000	w LW010000		F LF020000	O LO020000	W LW020000	6 ND060000
-7			ESC SE280000	EOT SE050000					g LG010000	p LP010000	x LX010000		G LG020000	P LP020000	X LX020000	7 ND070000
-8		CAN SE250000							h LH010000	q LQ010000	y LY010000		H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9		EM SE260000						` SD130000	i LI010000	r LR010000	z LZ010000		I LI020000	R LR020000	Z LZ020000	9 ND090000
-A						! SP020000		: SP130000				[SM060000			FN2 SE400000	FN3 SE410000
-B	VT SE120000				. SP110000	$ SC030000	, SP080000	# SM010000				] SM080000				
-C	FF SE130000			DC4 SE210000	< SA030000	* SM040000	% SM020000	@ SM050000								
-D	CR SE140000	GS SE360000	ENQ SE060000	NAK SE220000	(SP060000	) SP070000	_ SP090000	' SP050000								
-E	SO SE150000	RS SE370000	ACK SE070000		+ SA010000	; SP140000	> SA050000	= SA040000				FN4 SE420000				
-F	SI SE160000	US SE380000	BEL SE080000	SUB SE270000	| SO130000		? SP150000	" SP040000	FN1 SE390000							DEL SE330000

Figure 117. Code 128 Code Page (CPGID = 1303)

Note: All START, STOP, SHIFT, and CODE characters are generated by the printer to produce the shortest bar code possible from the given data; these characters are not specified in the Bar Code Symbol Data. All code points not listed in the table are undefined. The code points that do not have graphic characters shapes, such as X'00' (NUL) and X'8F' (FN1), are control codes defined within the Code 128 symbology; in the HRI, control codes print in a device-dependent manner.

MOD Parameter

The modifier field gives additional processing information about the bar code symbol to be generated. For example, it indicates whether a check-digit is to be generated for the bar code symbol.

Table 18 shows the modifier values for each bar code type.

Table 18. Modifier Values by Bar Code Type

Bar Code Type	MOD Value
1 – Code 39 (3-of-9 Code), AIM USS-39	X'01' and X'02'
2 – MSI (modified Plessey code)	X'01' through X'09'
3 – UPC/CGPC Version A	X'00'
5 – UPC/CGPC Version E	X'00'
6 – UPC - Two-digit Supplemental	X'00' - X'02'
7 – UPC - Five-digit Supplemental	X'00' - X'02'
8 – EAN 8 (includes JAN-short)	X'00'
9 – EAN 13 (includes JAN-standard)	X'00'
10 – Industrial 2-of-5	X'01' and X'02'
11 – Matrix 2-of-5	X'01' and X'02'
12 – Interleaved 2-of-5, AIM USS-I 2/5	X'01' and X'02'
13 – Codabar, 2-of-7, AIM USS-Codabar	X'01' and X'02'
17 – Code 128, AIM USS-128	X'02'
22 – EAN Two-digit Supplemental	X'00' and X'01'
23 – EAN Five-digit Supplemental	X'00' and X'01'
24 – POSTNET	X'00' through X'03'
26 – RM4SCC	X'00' and X'01'
27 – JPOSTAL	X'00' and X'01'
31 – APOSTAL	X'01' - X'08'

The modifier values, by bar code type, are as follows:

Code 39 (3-of-9 Code), AIM USS-39

X'01' Present the bar code without a generated check digit.

X'02' Generate a check digit and present it with the bar code.

Note: The Code 39 character set contains 43 characters including numbers, upper-case alphabetics, and some special characters. The Code 39 Specification also provides a method of encoding all 128 ASCII characters by using 2 bar code characters for those ASCII characters that are not in the standard Code 39 character set. This is sometimes referred to as ″Extended Code 39″ and is supported by all BCOCA receivers. In this case, the 2 bar code characters used to specify the ″extended character″ will be shown in the Human-Readable Interpretation and the bar code scanner will interpret the 2-character combination bar/space pattern appropriately.

- **MSI (modified Plessey code)**

  X'01' Present the bar code without check digits generated by the printer.

  X'02' Present the bar code with a generated IBM modulo-10 check digit. This check digit will be the second check digit; the first check digit is the last character of the data as defined in the associated FIELD START and LENGTH subcommands.

X'03' Present the bar code with two check digits. Both check digits are generated using the IBM modulo-10 algorithm.

X'04' Present the bar code with two check digits. The first check digit is generated using the NCR modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals the remainder; exception condition EC-0E00 exists if the first check-digit calculation results in a value of 10.

X'05' Present the bar code with two check digits. The first check digit is generated using the IBM modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals the remainder; exception condition EC-0E00 exists if the first check-digit calculation results in a value of 10.

X'06' Present the bar code with two check digits. The first check digit is generated using the NCR modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals 11 minus the remainder; a first check digit value of 10 is assigned the value zero.

X'07' Present the bar code with two check digits. The first check digit is generated using the IBM modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals 11 minus the remainder; a first check digit value of 10 is assigned the value zero.

X'08' Present the bar code with two check digits. The first check digit is generated using the NCR modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals 11 minus the remainder; exception condition EC-0E00 exists if the first check-digit calculation results in a value of 10.

X'09' Present the bar code with two check digits. The first check digit is generated using the IBM modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals 11 minus the remainder; exception condition EC-0E00 exists if the first check-digit calculation results in a value of 10.

- **UPC/CGPC—Version A**

  X'00' Present the standard UPC-A bar code with a generated check digit. The data to be encoded consists of eleven digits. The first digit is the number-system digit; the next ten digits are the article number.

- **UPC/CGPC—Version E**

  X'00' Present a UPC-E bar code symbol. Of the 10 input digits, six digits are encoded. The check digit is generated using all 10 input data digits. The check digit is not encoded; it is only used to assign odd or even parity to the six encoded digits.

- **UPC—Two-Digit Supplemental**

  X'00' Present a UPC two-digit supplemental bar code symbol. This option assumes that the base UPC Version A or E symbol is presented as a separate bar code object. The bar and space patterns used for the two supplemental digits are left-odd or left-even parity, with the parity determined by the digit combination.

  X'01' The two-digit UPC supplemental bar code symbol is preceded by a UPC

Version A, Number System 0, bar code symbol. The bar code object contains both the UPC Version A symbol and the two-digit supplemental symbol. The input data consists of the number system digit, the ten-digit article number, and the two supplement digits, in that order. A check digit is generated for the UPC Version A symbol. The two-digit supplemental bar code is presented after the UPC Version A symbol using left-hand odd and even parity as determined by the two supplemental digits.

X'02' The two-digit UPC supplemental bar code symbol is preceded by a UPC Version E symbol. The bar code object contains both the UPC Version E symbol and the two-digit supplemental symbol. The input data consists of the ten-digit article number and the two supplemental digits. The bar code object processor generates the six-digit UPC Version E symbol and a check digit. The check digit is used to determine the parity pattern of the six-digit Version E symbol. The two-digit supplemental bar code symbol is presented after the Version E symbol using left-hand odd and even parity as determined by the two digits.

- **UPC—Five-Digit Supplemental**

  X'00' Present the UPC five-digit supplemental bar code symbol. This option assumes that the base UPC Version A or E symbol is presented as a separate bar code object. A check digit is generated from the five supplemental digits and is used to assign the left-odd and left-even parity of the five-digit supplemental bar code. The supplemental check digit is not encoded or interpreted.

  X'01' The five-digit UPC supplemental bar code symbol is preceded by a UPC Version A, Number System 0, bar code symbol. The bar code object contains both the UPC Version A symbol and the five-digit supplemental symbol. The input data consists of the number system digit, the ten-digit article number, and the five supplement digits, in that order. A check digit is generated for the UPC Version A symbol. A second check digit is generated from the five supplement digits. It is used to assign the left-hand odd and even parity of the five-digit supplemental bar code symbol. The supplement check digit is not encoded or interpreted.

  X'02' The five-digit UPC supplemental bar code symbol is preceded by a UPC Version E symbol. The bar code object contains both the UPC Version E symbol and the five-digit supplemental symbol. The input data consists of the ten-digit article number and the five-digit supplemental data. The bar code object processor generates the six-digit UPC Version E symbol and check digit. The check digit is used to determine the parity pattern of the Version E symbol. The five-digit supplemental bar code symbol is presented after the Version E symbol. A second check digit is calculated for the five-digit supplemental data and is used to assign the left-hand odd and even parity. The supplement check digit is not encoded or interpreted.

- **EAN-8 (includes JAN-short)**

  X'00' Present an EAN-8 bar code symbol. The input data consists of seven digits: two flag digits and five article number digits. All seven digits are encoded along with a generated check digit.

- **EAN-13 (includes JAN-standard)**

X'00' Present an EAN-13 bar code symbol. The input data consists of twelve digits: two flag digits and ten article number digits, in that order. The first flag digit is not encoded. The second flag digit, the article number digits, and generated check digit are encoded. The first flag digit is presented in HRI form at the bottom of the left *quiet zone*. The first flag digit governs the A and B number-set pattern of the bar and space coding of the six digits to the left of the symbol center pattern.

- **Industrial 2-of-5**

  X'01' Present the bar code without a generated check digit.

  X'02' Generate a check digit and present it with the bar code.

- **Matrix 2-of-5**

  X'01' Present the bar code symbol without a generated check digit.

  X'02' Generate a check digit and present it with the bar code.

- **Interleaved 2-of-5, AIM USS-I 2/5**

  The Interleaved 2-of-5 symbology requires an even number of digits. The printer will add a leading zero if necessary in order to meet this requirement.

  X'01' Present the bar code symbol without a check digit.

  X'02' Generate a check digit and present it with the bar code.

- **Codabar, 2-of-7, AIM USS-Codabar**

  X'01' Present the bar code without a generated check digit. The input data consists of a start character, digits to be encoded, and a stop character, in that order.

  X'02' Generate a check digit and present it with the bar code. The input data consists of a start character, digits to be encoded, and a stop character, in that order.

- **Code 128, AIM USS-128**

  X'02' Generate a check digit and present it with the bar code.

Note: UCC/EAN 128 is a subset of Code 128 that begins with an FN1 character, followed by an Application Identifier and the data to be bar encoded. All of these characters (including the FN1 character) must be supplied within the Bar Code Symbol Data (BSA).

UCC/EAN 128 also requires that the symbol begins in subset C (that is start with a ″start (Code C)″ character), but the original Code 128 symbology specification required the symbol to begin in subset B (this is described in Appendix G of the symbology specification). Some bar code scanners can handle either start character for an UCC/EAN 128 symbol, but others require the ″start (Code C)″ character. Some older IPDS printers follow the original Code 128 specification and therefore will start all UCC/EAN 128 symbols with a ″start (Code B)″ character.

- **EAN Two-Digit Supplemental**

X'00' Present the EAN two-digit supplemental bar code symbol. This option assumes that the base EAN-13 symbol is presented as a separate bar code object. The value of the two digit supplemental data determines their bar and space patterns chosen from number sets A and B.

X'01' The two-digit supplemental bar code symbol is preceded by a normal EAN-13 bar code symbol. The bar code object contains both the EAN-13 symbol and the two-digit supplemental symbol. The two-digit supplemental bar code is presented after the EAN-13 symbol using left hand odd and even parity as determined by the two supplemental digits chosen from number sets A and B.

Note: Used for both books and paperbacks.

- **EAN Five-Digit Supplemental**

  X'00' Present the EAN five-digit supplemental bar code. This option assumes that the base EAN-13 symbol is presented as a separate bar code object. A check digit is calculated from the five supplemental digits. The check digit is also used to assign the bar and space patterns from number sets A and B for the five supplemental digits. The check digit is not encoded or interpreted.

  X'01' The five-digit supplemental bar code symbol is preceded by a normal EAN-13 bar code symbol. The bar code object contains both the EAN-13 symbol and the five-digit supplemental symbol. A check digit is generated from the five-digit supplemental data. The check digit is used to assign the bar and space patterns from number sets A and B. The check digit is not encoded or interpreted.

  Note: Used for books and paperbacks.

- **POSTNET**

  For all POSTNET modifiers that follow, the BSA HRI flag field and the BSD module width, element height, height multiplier, and wide-to-narrow ratio fields are not applicable to the POSTNET bar code symbology. These fields are ignored because the POSTNET symbology defines specific values for these parameters.

  X'00' Present a POSTNET ZIP Code bar code symbol. The ZIP Code to be encoded is defined as a five-digit, numeric (0–9), data variable to the BSA data structure. The POSTNET ZIP Code bar code consists of a leading frame bar, the encoded ZIP Code data, a correction digit, and a trailing frame bar.

  X'01' Present a POSTNET ZIP+4 bar code symbol. The ZIP+4 code to be encoded is defined as a nine-digit, numeric (0–9), data variable to the BSA data structure. The POSTNET ZIP+4 bar code consists of a leading frame bar, the encoded ZIP+4 data, a correction digit, and a trailing frame bar.

  X'02' Present a POSTNET Advanced Bar Code (ABC) bar code symbol. The ABC code to be encoded is defined as an eleven-digit, numeric (0–9), data variable to the BSA data structure. The POSTNET ABC bar code consists of a leading frame bar, the encoded ABC data, a correction digit, and a trailing frame bar.

  Note: An 11-digit POSTNET bar code is called a *Delivery Point bar code*.

X'03' Present a POSTNET variable-length bar code symbol. The data to be encoded is defined as an n-digit, numeric (0–9), data variable to the BSA data structure. The bar code symbol is generated without length checking; the symbol is not guaranteed to be scannable or interpretable. The POSTNET variable-length bar code consists of a leading frame bar, the encoded data, a correction digit, and a trailing frame bar.

- **RM4SCC**

  A 4 state customer code defined by the Royal Mail Postal service of England for use in bar coding postal code information.

  X'00' Present a RM4SCC bar code symbol with a generated start bit, checksum character, and stop bit. The start and stop bits identify the beginning and end of the bar code symbol and also the orientation of the symbol.

  X'01' Present a RM4SCC bar code symbol with no start bar, no checksum character, and no stop bar.

  Note: Modifier X'01' is also known as ″Dutch Kix Postal Bar Code″. In addition to the characters allowed in Modifier X'00', it allows lowercase alphabetical characters which will be folded to uppercase by the printer.

- **JPOSTAL**

  A complete Japan Postal Bar Code symbol consisting of a set of distinct bars and spaces for each character, followed by a modulo 19 checksum character and enclosed by a unique start character, stop character, and quiet zones.

  X'00' Present a Japan Postal Bar Code symbol with a generated start character, checksum character, and stop character.

  The generated bar code symbol consists of a start code, a 7-digit new postal code, a 13-digit address indication number, a check digit, and a stop code. The variable data to be encoded (BSA bytes 5-n) is used as follows:

  1. The first few digits represent the new postal code in either the form nnn-nnnn or the form nnnnnnn; the hyphen, if present, is ignored and the other 7 digits must be numeric. The 7 digits are placed in the new postal code field of the bar code symbol.
  2. If the next digit is a hyphen, it is ignored and is not used in generating the bar code symbol.
  3. The remainder of the BSA data is the address indication number, which can contain numbers, hyphens, and alphabetic characters (A-Z). Each number and each hyphen represents one digit in the bar code symbol; each alphabetic character is represented by a combination of a control code (CC1, CC2, or CC3) and a numerical code, and handled as two digits in the bar code symbol. Thirteen digits of this address indication number data are placed in the address indication number field of the bar code symbol.
     - If less than 13 additional digits are present, the shortage is filled in with the bar code corresponding to control code CC4 up to the thirteenth digit.
     - If more than 13 additional digits are present, the first 13 are used and the remainder ignored, with no exception condition reported.

However, if the thirteenth digit is the control code for an alphabetic (A-Z) character, only the control code is included and the numeric part is omitted.

X'01' Present a Japan Postal Bar Code symbol directly from the bar code data. Each valid character in the BSA data field is converted into a bar/space pattern, with no validity or length checking. The printer does not generate start, stop, or check digits.

To produce a valid bar code symbol, the bar code data must contain a start code, a 7-digit new postal code, a 13-digit address indication number, a valid check digit, and a stop code. The new postal code must consist of 7 numeric digits. The address indication number must consist of 13 characters, which can be numeric, hyphen, or control characters (CC1 through CC8). The following table lists the valid code points for modifier X'01':

Table 19. Valid EBCDIC-based Code Points for Japan Postal Bar Code

Bar Code Character	Code Point	Numerical Checking Value	Bar Code Character	Code Point	Numerical Checking Value
start	X'4C'		0	X'F0'	0
stop	X'6E'		1	X'F1'	1
hyphen	X'60'	10	2	X'F2'	2
CC1	X'5A'	11	3	X'F3'	3
CC2	X'7F'	12	4	X'F4'	4
CC3	X'7B'	13	5	X'F5'	5
CC4	X'E0'	14	6	X'F6'	6
CC5	X'6C'	15	7	X'F7'	7
CC6	X'50'	16	8	X'F8'	8
CC7	X'7D'	17	9	X'F9'	9
CC8	X'4D'	18			

Note: Do not attempt to use the Start and Stop characters in calculating the check digit. You can use the remaining characters to generate check digits; they are the only characters that are valid for check digits. Use the Numeric Checking Values to calculate the check digits.

Note:

You supply data generation for mod 1. The check digit is the sum of the digits modulo 19, which is a remainder of X. The check digit is 19 minus X, converted to hex. If this is done incorrectly, print server displays message ‘APS830I’.

The hyphen has a hex value of X'60' and a checking digit numerical of 10.

The following example is a generation of the customer bar code:

```
address
   154
   3-16-4, Wakabayshi, Setagaya-ku
```

```
New postal code + address indication number:

   154-0023-3-16-4
```

where, at this point, 154-0023 is the new postal code and 3 - 1 6 - 4 is the address indication number.

Delete hyphens between the third and fourth digits of the new postal code and between the new postal code and address indication number, as follows:

```
15400233-16-4
```

If the address indication number is shorter than 13 digits, use CC4s to fill the remaining spaces, as in the following example.

```
15400233-16-4 CC4 CC4 CC4 CC4 CC4 CC4 CC4
```

The first 7 digits are ignored as the postal code and the remaining digits are the address indication number. Remember to count hyphens as digits. In the previous example, the postal code is 1540023 and the address indication number is 3 - 1 6 - 4 plus seven CC4 characters.

Calculate the check digit (CD), based on the table of correspondence between characters for bar code and checking numerals. See Table 19 on page 364 for more information about check digits.

1+5+4+0+0+2+3+3+10+1+6+10+4+14+14+14+14+14+14+CD = 147 + CD = integral multiple of 19. Using the integral multiple of 19, 152 − 147 = 5 for the check digit, based on the table of correspondence between characters for bar code and checking numerals. Five corresponds to checking numerical five.

For the previous postal code and address indication number, calculate the hex value of the check digit. The following table shows how to convert the data to hex values. Add the check digit (CD), start code (STC), and stop code (SPC), as follows:

Table 20. Table Shows How to Convert Data to Hex Values.

Start Code (STC)	**HEX**
1	F1
5	F5
4	F4
0	F0
0	F0
2	F2
3	F3
3	F3
-	60
1	F1

Table 20. Table Shows How to Convert Data to Hex Values. (continued)

Start Code (STC)	HEX
6	F6
-	60
4	F4
CC4	E0
CC4	E0
CC4	E0
CC4	E0
CC4	E0
CC4	E0
CC4	E0
CD(5)	F5
SPC	6E

Notice that the check digit (CD) equals 5 and is converted to the hex value of F5.

The following are examples of various Japanese postal barcodes.

```
PAGEDEF SLSRPT;
   PRINTLINE POSITION 2 IN 2 IN;
   FIELD START 1 LENGTH 23
   POSITION CURRENT NEXT
   DIRECTION ACROSS
   BARCODE JAPAN  TYPE JPOSTAL MOD 1;
```

This barcode used numeric postal codes only. The 7-digit field contains the start, stop, and checksum characters. The printer does not generate start, stop, or checksum characters.

```
PAGEDEF SLSRPT;
   PRINTLINE POSITION 2 IN 2 IN;
   FIELD START 1 LENGTH 23
   POSITION CURRENT NEXT
   DIRECTION ACROSS
   BARCODE JAPAN  TYPE JPOSTAL MOD 1;
```

This barcode used alphanumeric postal codes only. The 13-digit field contains start, stop, checksum, and command codes. The printer does not generate start, stop, or checksum characters.

```
PAGEDEF SLSRPT;
   PRINTLINE POSITION 2 IN 2 IN;
   FIELD START 1 LENGTH 7
   POSITION CURRENT NEXT
   DIRECTION ACROSS
   BARCODE JAPAN  TYPE JPOSTAL MOD 0;
```

This barcode used numeric postal codes only. This is a 7-digit character field.

```
PAGEDEF SLSRPT;
   PRINTLINE POSITION 2 IN 2 IN;
   FIELD START 1 LENGTH 13
   POSITION CURRENT NEXT
   DIRECTION ACROSS
   BARCODE JAPAN  TYPE JPOSTAL MOD 0;
```

This barcode used alphanumeric postal codes only. This is a 13-digit character field.

- **APOSTAL**

  A complete Australian Postal Bar Code symbol consisting of a set of distinct bars and spaces for each character. The bar code is set to 8 numeric digits representing the Sorting Code, and can be followed by from 0 to 31 numeric digits of Customer Information.

Australia Post Bar Code

A bar code symbology defined by Australia Post for use in Australian postal systems. There are several formats of this bar code which are identified by the modifier byte as follows:

Modifier	Type of bar code	Valid bar code data
X'01'	Standard Customer Barcode (format code = 11)	An 8 digit number representing the Sorting Code
X'02'	Customer Barcode 2 using Table N (format code = 59)	An 8 digit number representing the Sorting Code followed by up to 8 numeric digits representing the Customer Information
X'03'	Customer Barcode 2 using Table C (format code = 59)	An 8 digit number representing the Sorting Code followed by up to 5 characters (A-Z, a-z, 0-9, space, #) representing the Customer Information
X'04'	Customer Barcode 2 using proprietary encoding (format code = 59)	An 8 digit number representing the Sorting Code followed by up to 16 numeric digits (0-3) representing the Customer Information. Each of the 16 digits specify one of the 4 types of bar.
X'05'	Customer Barcode 3 using Table N (format code = 62)	An 8 digit number representing the Sorting Code followed by up to 15 numeric digits representing the Customer Information
X'06'	Customer Barcode 3 using Table C (format code = 62)	An 8 digit number representing the Sorting Code followed by up to 10 characters (A-Z, a-z, 0-9, space, #) representing the Customer Information
X'07'	Customer Barcode 3 using proprietary encoding (format code = 62)	An 8 digit number representing the Sorting Code followed by up to 31 numeric digits (0-3) representing the Customer Information. Each of the 31 digits specify one of the 4 types of bar.
X'08'	Reply Paid Barcode (format code = 45)	An 8 digit number representing the Sorting Code

Check Digit Calculation Method

The proprietary encoding allows the customer to specify the types of bars to be printed directly by using 0 for a full bar, 1 for an ascending bar, 2 for a descending bar, and 3 for a timing bar. If the customer does not specify enough Customer Information to fill the field, the printer uses a filler bar to extend pad the field out to the correct number of bars.

The printer will encode the data using the proper tables, generate the start and stop bars, generate any needed filler bars, and generate the Reed Solomon ECC bars.

Human readable interpretation (HRI) can be selected with this bar code type. The format control code, Delivery Point Identifier, and customer information field (if any) appears in the HRI, but the ECC does not.

Some bar code types and modifiers call for the calculation and presentation of check digits. Check digits are a method of verifying data integrity during the bar coding reading process. Except for UPC Version E, the check digit is always presented in the bar code bar and space patterns, but is not always presented in the HRI. The following table shows the check digit calculation methods for each bar code type and the presence or absence of the check digit in the HRI.

Table 21. Check Digit Calculation Methods For Each Bar Code

Bar Code Type	Modifier	In HRI?	Check Digit Calculation
1 – Code 39 (3-of-9 Code), AIM USS-39	X'02'	Yes	Modulo 43 of the sum of the data characters' numerical values as described in a Code 39 specification. The start and stop codes are not included in the calculation.
2 – MSI (modified Plessey code)	X'02' – X'09'	No	IBM Modulus 10 check digit: 1. Multiply each digit of the original number by a weighting factor of 1 or 2 as follows: multiply the units digit by 2, the tens digit by 1, the hundreds digit by 2, the thousands digit by 1, and so forth. 2. Sum the digits of the products from step 1. This is not the same as summing the values of the products. 3. The check digit is described by the following equation where "sum" is the resulting value of step 2: (10 - (sum modulo 10)) modulo 10

Table 21. Check Digit Calculation Methods For Each Bar Code (continued)

Bar Code Type	Modifier	In HRI?	Check Digit Calculation
			IBM Modulus 11 check digit: 1. Multiply each digit of the original number by a repeating weighting factor pattern of 2, 3, 4, 5, 6, 7 as follows: multiply the units digit by 2, the tens digit by 3, the hundreds digit by 4, the thousands digit by 5, and so forth. 2. Sum the products from step 1. 3. The check digit depends on the bar code modifier. The check digit as the remainder is described by the following equation where “sum” is the resulting value of step 2: (sum modulo 11) The check digit as 11 minus the remainder is described by the following equation: (11 - (sum modulo 11)) modulo 11
			NCR Modulus 11 check digit: 1. Multiply each digit of the original number by a repeating weighting factor pattern of 2, 3, 4, 5, 6, 7, 8, 9 as follows: multiply the units digit by 2, the tens digit by 3, the hundreds digit by 4, the thousands digit by 5, and so forth. 2. Sum the products from step 1. 3. The check digit depends on the bar code modifier. The check digit as the remainder is described by the following equation where “sum” is the resulting value of step 2: (sum modulo 11) The check digit as 11 minus the remainder is described by the following equation: (11 - (sum modulo 11)) modulo 11

Table 21. Check Digit Calculation Methods For Each Bar Code (continued)

Bar Code Type	Modifier	In HRI?	Check Digit Calculation
3 – UPC/CGPC Version A	X'00'	Yes	UPC/EAN check digit calculation: 1. Multiply each digit of the original number by a weighting factor of 1 or 3 as follows: multiply the units digit by 3, the tens digit by 1, the hundreds digit by 3, the thousands digit by 1, and so forth. 2. Sum the products from step 1. 3. The check digit is described by the following equation where "sum" is the resulting value of step 2: (10 - (sum modulo 10)) modulo 10
5 – UPC/CGPC Version E	X'00'	Yes	See UPC/CGPC Version A
8 – EAN 8 (includes JAN-short)	X'00'	Yes	See UPC/CGPC Version A
9 – EAN 13 (includes JAN-standard)	X'00'	Yes	See UPC/CGPC Version A
10 – Industrial 2-of-5	X'02'	Yes	See UPC/CGPC Version A
11 – Matrix 2-of-5	X'02'	Yes	See UPC/CGPC Version A
12 – Interleaved 2-of-5	X'02'	Yes	See UPC/CGPC Version A
13 – Codabar, 2-of-7, AIM USS-Codabar	X'02'	No	Codabar check digit calculation: 1. Sum of the data characters' numerical values as described in a Codabar specification. All data characters are used, including the start and stop characters. 2. The check digit is described by the following equation where "sum" is the resulting value of step 1: (16 - (sum modulo 16)) modulo 16
17 – Code 128, AIM USS-128	X'02'	No	Code 128 check digit calculation: 1. Going left to right starting at the start character, sum the value of the start character and the weighted values of data and special characters. The weights are 1 for the first data or special character, 2 for the second, 3 for the third, and so forth. The stop character is not included in the calculation. 2. The check digit is modulo 103 of the resulting value of step 1.
24 – POSTNET	X'00' – X'03'	NA	The POSTNET check digit is (10 - (sum modulo 10)) modulo 10, where sum is the sum of the ZIP code data.

Table 21. Check Digit Calculation Methods For Each Bar Code (continued)

Bar Code Type	Modifier	In HRI?	Check Digit Calculation
26 – RM4SCC	X'00'	NA	The RM4SCC checksum digit is calculated using an algorithm that weights each of the 4 bars within a character in relation to its position within the character.
	X'01'	NA	None.
27 – JPOSTAL 27	X'00'	N/A	The Japan Postal Bar Code check digit calculation: Convert each character in the bar code data into decimal numbers. Numeric characters are converted to decimal; each hyphen character is converted to the number 10, each alphabetic character is converted to two numbers according to the symbology definition. For example, A becomes “11 and 0”, B becomes “11 and 1”,..., J becomes “11 and 9”, K becomes “12 and 0”, L becomes “12 and 1”, ..., T becomes “12 and 9”, U becomes “13 and 0”, V becomes “13 and 1”, ..., and Z becomes “13 and 5”. Sum the resulting decimal numbers and calculate the remainder modulo 19. The check digit is 19 minus the remainder.
	X'01'	N/A	None
31 – APOSTAL	X'01' – X'08'	No	The Australian Post Bar Code uses a Reed Solomon error correction code based on Galois Field 64.

Appendix E. PPFA Keywords

A keyword is a word in PPFA that must be entered exactly as shown. Keywords cannot be used as names for a font, segment, definition, overlay, defined color, or objects.

The following is a list of PPFA reserved keywords:

ABSOLUTE	DRAWGRAPHIC	NOGROUP	RADIUS
ADJUST	DUPLEX	NOPRELOAD	RASTER
ALIGN	ELLIPSE	NORASTER	RATIO
AXIS1	ENDGRAPHIC	OBID	RECID
AXIS2	ENDSPACE	OBJECT	REFERENCE
BACK	ENDSUBPAGE	OBKEEP	REPEAT
BARCODE	FIELD	OBNOKEEP	REPLACE
BCCOLOR	FINISH	OBTYPE	RES
BIN	FLASH	OBXNAME	RESOLUTION
BINERROR	FLDNUM	OB2ID	ROTATION
BODY	FONT	OB2RESOURCE	ROUNDED
BOTH	FONTFID	OB2TYPE	SBCS
BOX	FORMDEF	OB2XNAME	SCOPE
BOXSIZE	FRONT	OFFSET	SEGMENT
CHANNEL	GRAPHID	OPCOUNT	SETUNITS
CIRCLE	GROUP	OPERATION	SPACE_THEN_PRINT
CLRTRAP	GRPHEADER	OPOFFSET	SSASTERISK
CMYKEURO	HEIGHT	OPPOS	START
CMYKSWOP	HRI	OTHERWISE	SUBGROUP
COLOR	HRIFONT	OUTBIN	SUPPBLANKS
COLORVALUERR	INVOKE	OVERLAY	SUPPRESSION
COMMENT	JOG	OVROTATE	TEXT
CONDITION	LAYOUT	PAGECOUNT	TO
CONSTANT	LENGTH	PAGEDEF	TONERSAVER
COPIES	LINEONE	PAGEFORMAT	TRCREF
COPY	LINESP	PAGEHEADER	TYPE
COPYGROUP	LINETYPE	PAGENUM	WHEN
CORNERLENGTH	LINEWT	PAGETRAILER	WIDTH
CUTSHEET	MEDIUM	PLACE	XSPACE
CVERROR	METRICTECHNOLOGY	POSITION	ZFOLD
DBCS	METTECH	PRELOAD	
DEFINE	MOD	PRESENT	
DELIMITER	N	PRINTDATA	
DIRECTION	N_UP	PRINTLINE	
	NEWPAGE	PROCESSING	
		QUALITY	

Appendix F. PPFA Media Names

Figure 118 on page 377 lists the PPFA media names, media types, and component identifiers.

Note: The range of component ids from 12,288 to 268,435,455 is reserved for user defined media types.

Media Name	Media Type	Component ID
BSNS ENV	North American business envelope	143
COM 10 ENV	Com10 envelope (9.5 x 4.125 in)	75
C5 ENV	C5 envelope (229 x 110 mm)	79
DL ENV	DL envelope (220 x 110 mm)	77
EXEC	North American executive (7.25 x 10.5 in)	65
INDEX CD	Index Card	150
ISO A3	ISO A3 white (297 x 420 mm)	10
ISO A3 CO	ISO A3 colored	11
ISO A4	ISO A4 white (210 x 297 mm)	0
ISO A4 CO	ISO A4 colored	1
ISO A4 TAB	ISO A4 tab (225 x 297 mm)	7
ISO A4 THD	ISO 1/3 A4	5
ISO A4 TR	ISO A4 transparent	2
ISO A5	ISO A5 white (148.5 x 210 mm)	20
ISO A5 CO	ISO A5 colored	21
ISO A6 PC	ISO A6 Postcard	152
ISO B4	ISO B4 white (257 x 364 mm)	30
ISO B4 CO	ISO B4 colored	31
ISO B4 ENV	ISO B4 envelope	83
ISO B5	ISO B5 white (176 x 250 mm)	40
ISO B5 CO	ISO B5 colored	41
ISO B5 ENV	ISO B5 envelope	73
ISO C4 ENV	ISO C4 envelope	93
ISO C5 ENV	ISO C5 envelope	103
ISO LNG ENV	ISO long envelope	113
JIS B4	JIS B4 (257 x 364 mm)	42
JIS B5	JIS B5 (182 x 257 mm)	43
JP PC	Japan postcard (Hagaki) (100 x 148 mm)	81
JP PC ENV	Japan postcard envelope (200 x 150 mm)	80
LEDGER	North American ledger (11 x 17 in)	67
LEGAL	North American legal white (8.5 x 14 in)	60
LEGAL CO	North American legal colored	61
LEGAL TAB	Legal tab (9 x 14 in)	146
LEGAL 13	North American legal 13 (Folio) (8.5 x 14 in)	63
LETTER	North American letter white (8.5 x 11 in)	50
LETTER CO	North American letter colored	51
LETTER TAB	Letter tab (9 x 11 in)	145
LETTER TR	North American letter transparent	52
MON ENV	Monarch envelope (7.5 x 3.875 in)	76
STATEMNT	North American statement (5.5 x 8.5 in)	69
US PC	US Postcard	151
9x12 ENV	North American 9 x 12 envelope	133
10x13 ENV	North American 10 x 13 envelope	123
9x12 MAN	Manual (9 x 12 in)	147
8x10.5 MED	Media (8 x 10.5 in)	148

Figure 118. Registered Media Types Sorted By Media Name

Appendix G. Fill Patterns for Drawgraphic Commands

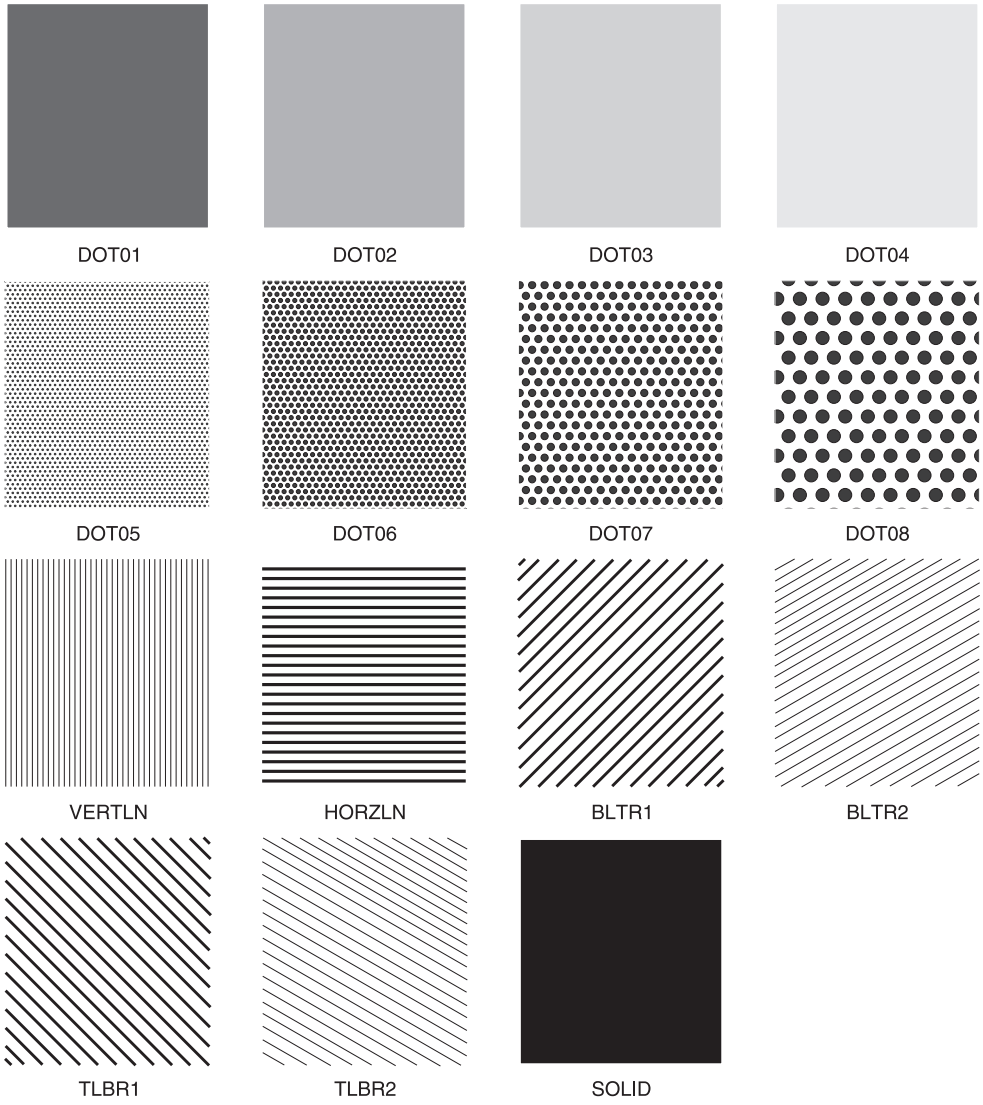


Figure 119. Fill Patterns for Drawgraphic Commands

Appendix H. PPFA Messages and Codes

At the end of processing for each command, the maximum error level encountered during processing is printed on the system printer, providing the error was not caused by the system printer itself. The meaning of the return codes is shown in Table 22.

Table 22. Return Codes

Return Code	Severity	Description
Return Code 0	I = Information; the command is processed.	PPFA did not encounter any problems. No warning, error, severe-error, or termination-error message was issued.

Table 22. Return Codes (continued)

Return Code	Severity	Description
Return Code 4	W = Warning; the command is processed.	PPFA encountered at least one non-terminating error, solved by an assumption. At least one warning message was issued. No error, severe-error, or terminating-error message was issued. The requested function was probably correctly performed. The program executed to completion.
Return Code 8	E = Error; the command is partially processed.	PPFA encountered at least one error, but no severe or terminating error. A requested function may be partially incomplete.
Return Code 12	S = Severe error; the command is not processed.	PPFA encountered a severe error. The program executed to completion, but some of the functions requested were not performed.
Return Code 16	T = Termination error; the job is terminated.	PPFA encountered a terminating error. The program terminated prematurely.

PPFA Messages and Their Meanings

The general format of the error message is as follows:

All messages consist of a standard seven-character prefix, followed by the message text:

AKQnnnS THIS IS THE MESSAGE TEXT . . .
AKQ is the three-character identifier of Page Printer Formatting Aid for AIX (PPFA).
nnn is the message number.
S is the message-severity indicator. The indicators are defined in Table 22 on page 379.

Note: You cannot use the **psfmsg** command to view PPFA messages.

In addition, PPFA errors are written to a listing file. AIX messages are written to standard error. Sometimes, AIX-specific errors mean that PPFA errors are not written to a listing file.

Note: PPFA issues a maximum of 269 user errors generated within a source file, and one additional message is used for the message queue to indicate an out-of-storage condition.

AKQ001E END OF COMMENT (*/) IS NOT SPECIFIED.

Explanation: The end mark of a comment (*/) is not specified.

System Action: The page definition or form definition is not generated. The syntax check may be ended.

Operator Response: Specify the end mark of a comment.

AKQ002E DBCS STRING DOES NOT END WITH SHIFT-IN.

Explanation: DBCS strings in comments must terminate with shift-in.

System Action: The form definition or page definition is not generated. The syntax check continues, assuming shift-in.

Operator Response: Specify a valid DBCS string enclosed by SO and SI.

AKQ003E LITERAL DOES NOT END WITH APOSTROPHE.

Explanation: A literal must end with an apostrophe.

System Action: The page definition is not generated. The syntax check continues, assuming an apostrophe.

Operator Response: Specify a valid literal enclosed by apostrophes. Note that an apostrophe in a literal is specified by consecutive double apostrophes.

AKQ004E DBCS LITERAL DOES NOT END WITH SHIFT-IN AND APOSTROPHE.

Explanation: A DBCS literal must end with shift-in and apostrophe.

System Action: The page definition is not generated. The syntax check continues, assuming the end of the DBCS literal at the end of a record.

Operator Response: Specify a valid literal ended by shift-in and apostrophe.

AKQ101E COMMAND SEQUENCE IS INVALID.

Explanation: The command sequence is invalid.

System Action: A page definition or form definition is not generated. The syntax check continues from a valid command.

Operator Response: Specify commands in a valid sequence.

AKQ102E INVALID COMMAND (erroneous entry) IS SPECIFIED.

Explanation: An invalid command is specified in the input data.

System Action: A page definition or form definition is not generated. The syntax check continues from a valid command.

Operator Response: Specify a valid command.

AKQ103E INVALID SUBCOMMAND (value) IS SPECIFIED.

Explanation: An invalid subcommand was specified in the input data. This message is often issued when a semicolon (;) is missing

System Action: A page definition or form definition is not generated. The syntax check continues from the next keyword.

Operator Response: Specify a valid subcommand.

AKQ104E (command or parameter name) NAME IS NOT SPECIFIED.

Explanation: The required name is not specified.

System Action: A page definition or form definition is not generated. The syntax check continues, assuming blanks or default as the name.

Operator Response: Specify the required name.

AKQ105E REQUIRED PARAMETER IN (subcommand name) IS NOT SPECIFIED.

Explanation: The subcommand indicated in the message requires a correct PPFA format.

System Action: A page definition or form definition is not generated. The syntax check continues, assuming the default values.

Operator Response: Refer to the command reference section of this publication for help in specifying a valid subcommand parameter.

AKQ106E (command or parameter name) NAME IS SPECIFIED WITH INVALID SYNTAX.

Explanation: The required name is specified with invalid syntax. See Table 7 on page 151 for the correct length of names.

System Action: A page definition or form definition is not generated. The syntax check continues.

Operator Response: Specify a valid name.

AKQ107E PARAMETER IN (subcommand name) IS INVALID.

Explanation: The parameter in the subcommand is invalid (invalid format or out of range).

System Action: A page definition or form definition is not generated. The syntax check continues, assuming the default values as the parameter.

Operator Response: Specify a valid parameter value.

AKQ108E (subcommand name) SUBCOMMAND IS DUPLICATED IN ONE COMMAND.

Explanation: The subcommand indicated in the message was specified more than once in the same command. Only one such subcommand is permitted within this command.

System Action: A page definition or form definition is not generated. The syntax check continues, ignoring the duplicate subcommand.

Operator Response: Delete one subcommand.

AKQ109E (subcommand name) SUBCOMMAND CONFLICTS WITH (subcommand name) SUBCOMMAND.

Explanation: One subcommand conflicts with another (FONT, PRINTLINE, FIELD, OPCOUNT, OPPOS

System Action: A page definition or form definition is not generated. The syntax check continues, ignoring the latter subcommand.

User Response: Delete one of the subcommands.

AKQ110E THE VALUE OF THE (command name) SUBCOMMAND IS TOO LARGE OR TOO SMALL.

Explanation: The parameter in the subcommand is out of range.

IN 136.5
MM 3467.1
CM 346.7
POINTS
9828.0
PELS (L-units)
32760

These values are specified in:
FORMDEF N_UP OVERLAY relative_xpos relative_ypos
PAGEDEF PRINTLINE OVERLAY / SEGMENT relative_xpos relative_ypos

Note: The values specified for the CPI and LPI are set in the SETUNITS subcommand.

System Action: No form definition or page definition is generated. PPFA continues syntax checking.

Operator Response: Specify a valid parameter value.

AKQ111E SUBCOMMAND SEQUENCE IS INVALID: (subcommand name) OCCURS AFTER (subcommand name)

Explanation: A WHEN subcommand occurs after an OTHERWISE subcommand in a CONDITION command.

System Action: A page definition or form definition is not generated. The syntax check continues, ignoring the subcommand.

Operator Response: Reorder or rewrite the conditions.

AKQ112E CONDITION COMMAND DOES NOT ALLOW '*' IN ITS START SUBCOMMAND.

Explanation: A relative position ('*', '* + n', or '* - n') was specified in a START subcommand of a CONDITION command.

System Action: A page definition or form definition is not generated. The syntax check continues from the valid subcommand.

Operator Response: Specify an absolute starting position.

AKQ113E MORE THAN ONE 'WHEN' SUBCOMMAND SPECIFIED THE CHANGE PARAMETER.

Explanation: More than one WHEN subcommand specified CHANGE for its field comparison.

System Action: A page definition or form definition is not generated. The syntax check continues from the valid subcommand.

Operator Response: Remove the extra subcommands specifying the CHANGE parameter.

AKQ114E NUMBER OF PARAMETERS EXCEED LIMIT FOR (subcommand name) SUBCOMMAND OR KEYWORD.

Explanation: The named subcommand/keyword in the messages limits the number of parameters that may be coded with a single subcommand or keyword. The number of parameters that can be coded with the named subcommand or keyword is defined in the command reference sections of this publication; see Chapter 9. Form Definition Command Reference and Chapter 10. Page Definition Command Reference (Traditional).

System Action: The form definition is not generated. The syntax check continues from the valid subcommand.

Operator Response: Remove the extra parameters.

AKQ115E REQUIRED PARAMETER(S) (PARM1, PARM2, . . .) IN (COMMAND OR SUBCOMMAND) IS (ARE) NOT SPECIFIED.

Explanation: This is a generic message which indicates one or more missing parameters on a subcommand or command. For example a DRAWGRAPHIC BOX must have a BOXSIZE subcommand coded.

System Action: A page or form definition is not generated.

Operator Response: Provide the correct parameter(s) on the command or subcommand.

AKQ116E PARAMETER (PARM1) IN (COMMAND OR SUBCOMMAND) IS INVALID.

Explanation: This is a generic message which indicates that a parameter in a subcommand or command is invalid.

System Action: A page or form definition is not generated.

Operator Response: Provide the correct parameter on the command or subcommand.

AKQ117E PARAMETER (PARM1) IN (COMMAND OR SUBCOMMAND) IS DUPLICATED.

Explanation: This is a generic message which indicates that a parameter in a subcommand or command is coded more than once. For example, . . . LINEWT LIGHT BOLD . . . shows two different line weights in the same subcommand.

System Action: A page or form definition is not generated.

Operator Response: Remove one of the parameters.

AKQ118E MUTUALLY EXCLUSIVE PARAMETERS ON THE (INSERT1) COMMAND OR SUBCOMMAND ARE DUPLICATED.

Explanation: A command or subcommand contains more than one mutually exclusive parameter. For example, the PAGECOUNT subcommand on the PAGEDEF command cannot have both STOP and CONTINUE coded.

System Action: A page or form definition is not generated.

Operator Response: Remove one of the parameters.

AKQ119E GRAPHICS-TYPE (BOX, LINE, CIRCLE, ELLIPSE) MUST IMMEDIATELY FOLLOW DRAWGRAPHIC.

Explanation: The DRAWGRAPHIC command must have the graphics type (BOX, LINE, CIRCLE, ELLIPSE) immediately following the command.

System Action: A page or form definition is not generated.

Operator Response: Code one of the graphics types.

AKQ201E (subcommand name) SUBCOMMAND IS NOT SPECIFIED.

Explanation: The required subcommand is not specified.

System Action: A page definition or form definition is not generated. The syntax check continues, assuming the default.

Operator Response: Specify the required subcommand.

AKQ202E SPECIFIED (command name) NAME IS NOT DEFINED.

Explanation: A resource name (OVERLAY, SUPPRESSION, FONT, or OBJECT) is not defined.

System Action: A page definition or form definition is not generated. The syntax check continues.

Operator Response: Correct the name.

AKQ203W (command name) NAME IS DUPLICATED.

Explanation: The required name must be unique for OVERLAY, COPYGROUP, FONT, PAGEFORMAT, OBJECT, or SUPPRESSION.

System Action: A page definition or form definition is generated.

Operator Response: Specify a unique name.

AKQ204E (object) NAME IS DUPLICATED.

Explanation: The name must be unique (OVERLAY, COPYGROUP, FONT, PAGEFORMAT, SEGMENT).

System Action: A page definition or form definition is not generated. The syntax check continues.

Operator Response: Specify a unique name.

AKQ205E PAGEFORMAT (pageformat name) WAS NOT FOUND IN THIS PAGE DEFINITION.

Explanation: A WHEN or OTHERWISE subcommand of CONDITION specifies a PAGEFORMAT name not found in the page definition being processed.

System Action: A page definition or form definition is not generated. The syntax check continues.

Operator Response: Specify a *pageformat name* that is in the page definition.

AKQ206E CONDITION (condition name) HAS ALREADY BEEN DEFINED.

Explanation: A CONDITION command specifies LENGTH, WHEN, or OTHERWISE, and the condition with this condition name has already been defined by an earlier CONDITION command.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Define the condition only the first time it occurs.

AKQ210E THE RELATIVE POSITION VALUE EXCEEDS THE ALLOWED RANGE

System Action: The value specified for the relative x position or relative y position on the N_UP subcommand (for an OVERLAY) or PRINTLINE command (for an OVERLAY or SEGMENT) exceeds the range of +32760 to -32760 L-units. For example, assuming the default of 240 pels per inch is being used, the values must be equal to, or less than the following:

IN 136.5
MM 3467.1
CM 346.7
POINTS
9828.0
PELS (L-units)
32760 (+ or -)
CPI *
LPI *

The value specified for CPI or LPI in the SETUNITS command will determine whether the value will exceed 32760 L-units.

System Action: The page definition or form definition is not generated. The syntax check continues.

Operator Response: Correct the relative x and y position values within the allowed range.

AKQ211E FRONT/BACK SIDE IS NOT SPECIFIED FOR DUPLEX.

Explanation: The SUBGROUP specified with BACK does not exist after the SUBGROUP specified with FRONT, or the SUBGROUP specified with FRONT does not exist before the SUBGROUP specified with BACK.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Specify subgroups for both sides.

AKQ212W PAPER SIDE IS SPECIFIED FOR SIMPLEX.

Explanation: A subgroup specified with BOTH, FRONT, or BACK is invalid with single-sided printing.

System Action: A form definition is generated, ignoring the subcommand specifying the paper side.

User Response: Either delete the subcommand that specified the paper side or specify DUPLEX.

AKQ213E LOGICAL PAGE POSITION EXCEEDS THE LIMIT.

Explanation: The logical page position specified by the OFFSET subcommand in the FORMDEF or COPYGROUP command exceeds the limits.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ214E MORE THAN 127 SUPPRESSIONS ARE SPECIFIED IN ONE FORMDEF.

Explanation: More than 127 suppressions are specified in one FORMDEF.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ215E MORE THAN 127 OVERLAYS ARE SPECIFIED IN ONE COPYGROUP.

Explanation: More than 127 OVERLAYs are specified in one copy group. PPFA can issue this message for an N_UP subcommand that specifies more than 127 overlays.

System Action: No form definition is generated. The syntax check continues.

User Response: Correct the error.

AKQ216E MORE THAN ONE RASTER OVERLAY IS SPECIFIED IN ONE COPYGROUP.

Explanation: More than one raster OVERLAY is specified in one copy group.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ217W LOGICAL PAGE POSITION FOR BACK SIDE OF PAGE SPECIFIED IN SIMPLEX PROCESSING

Explanation: The logical-page position specified by the OFFSET subcommand in a FORMDEF or COPYGROUP command for the back side of a page was specified, but simplex was specified in a COPYGROUP command.

System Action: A form definition is generated, with the back side logical page position included, as if duplex had been specified. The syntax check continues.

Operator Response: Correct the error by specifying duplex in the COPYGROUP command or remove the second set of coordinates in the OFFSET subcommand.

AKQ218E MORE THAN 255 COPIES ARE SPECIFIED IN ONE COPYGROUP.

Explanation: More than 255 copies are specified in a COPYGROUP.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ219E MORE THAN 127 SUBGROUPS ARE SPECIFIED IN ONE COPYGROUP.

Explanation: More than 127 subgroups are specified in a COPYGROUP.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ220E MORE THAN 8 OVERLAYS ARE SPECIFIED IN ONE SUBGROUP.

Explanation: More than eight overlays are specified in one SUBGROUP.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ221E MORE THAN 8 SUPPRESSIONS ARE SPECIFIED IN ONE SUBGROUP.

Explanation: More than eight suppressions are specified in one SUBGROUP.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ222W DIFFERENT NUMBERS OF COPIES ARE SPECIFIED FOR EACH SIDE OF DUPLEX.

Explanation: The number of copies for BACK side is not equal to those for FRONT side.

System Action: A form definition is generated assuming the number of copies specified for front side.

Operator Response: Check the number of copies.

AKQ223E LOGICAL PAGE POSITION FOR (page side) SIDE OF PAGE EXCEEDS THE LIMIT.

Explanation: The logical-page position specified by the OFFSET subcommand in a FORMDEF or COPYGROUP command exceeds the limit for the current side of the page.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Correct the positioning OFFSET parameter.

AKQ224E MORE THAN 127 OVERLAYS ARE SPECIFIED IN A PAGEFORMAT.

Explanation: The maximum number of OVERLAY commands is 127. PPFA can issue this message for the OVERLAY subcommand of the PRINTLINE command.

System Action: A page definition is not generated. The syntax check continues.

User Response: Specify a valid number of OVERLAY commands.

AKQ225E CONSTANT SUBCOMMAND PARAMETER (parameter) SPECIFIED IN SIMPLEX PROCESSING

Explanation: The BACK or BOTH parameter has been specified for the CONSTANT subcommand within simplex processing.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Correct this CONSTANT subcommand or indicate DUPLEX.

AKQ226E DIRECTION SUBCOMMAND ONLY ALLOWED WITH PRESENT SUBCOMMAND.

Explanation: The DIRECTION subcommand has been specified, but the PRESENT subcommand has not.

System Action: A form definition is not generated. The syntax check continues.

Operator Response: Either add the PRESENT subcommand or remove the DIRECTION subcommand.

AKQ227E THE ORIGIN OF THE RESOURCE (name) NAMED IN THE PRINTLINE COMMAND IS OFF THE LOGICAL PAGE.

Explanation: The relative position of the PRINTLINE overlay or segment named is off the logical page. The origin of the overlay or segment specified for the resource named in the N_UP subcommand is off the medium.

System Action: The page definition that has the overlay or segment in question is not generated. PPFA continues the syntax check, ignoring the problem.

Operator Response: Correct the x-position and y-position for the OVERLAY or SEGMENT subcommand.

AKQ228E THE ORIGIN OF THE OVERLAY (*overlay name*) NAMED IN THE (*command*) COMMAND IS OFF THE MEDIUM

Explanation: The resource position values will position the resource such that at least part of the resource will be off the medium (physical page).

System Action: The form definition that has the overlay in question is not generated. PPFA continues the syntax check, ignoring the problem.

User Response: Correct the relative x-position and relative y-position values for the OVERLAY named in the N_UP subcommand.

AKQ229W SUBGROUPS FOR FRONT AND BACK OF SAME SHEET USED DIFFERENT BINS.

Explanation: In your subgroup command you specified FRONT and BACK parameters. However, your COPYGROUP has different bins specified.

System Action: A form definition is generated that specifies the bin used for the front side.

Operator Response: Check the number of copies and correct the bin setting.

AKQ231E PRINTLINE OR LAYOUT IS NOT SPECIFIED.

Explanation: There is no PRINTLINE or LAYOUT command in the page format.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Specify either a PRINTLINE or LAYOUT command.

AKQ232E REQUIRED SUBCOMMAND TEXT OR LENGTH IS NOT SPECIFIED.

Explanation: A FIELD subcommand must have a TEXT or LENGTH subcommand.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Specify either a TEXT subcommand or a LENGTH subcommand.

AKQ233E THE LOGICAL PAGE SIZE IS TOO LARGE OR TOO SMALL.

Explanation: The specified page size is too large or too small. The page size must be from 1 to 32767 pels. The HEIGHT and WIDTH subcommands must have values between 1 and 32767 PELS, inclusive, or the same measurements expressed in other units.

System Action: A page definition is not generated. The syntax check continues, assuming the defaults.

Operator Response: Correct the error.

AKQ234E POSITION OF LINEONE EXCEEDS THE LOGICAL PAGE BOUNDARY.

Explanation: The TOP or MARGIN position specified by the LINEONE subcommand exceeds the logical page boundary. This error message is issued only if TOP or MARGIN is specified.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Specify a valid position value.

AKQ235E MORE THAN 127 SEGMENTS ARE SPECIFIED IN ONE PAGEFORMAT.

Explanation: More than 127 segments are specified in a single PAGEFORMAT command. PPFA can issue this message for the SEGMENT subcommand of the PRINTLINE command.

System Action: No page definition is generated. The syntax check continues.

User Response: Correct the error.

AKQ238E MORE THAN 127 FONTS ARE SPECIFIED IN ONE PAGEFORMAT.

Explanation: More than 127 fonts are specified in one PAGEFORMAT or the specified TRC number exceeds 126. PPFA counts each use of a font in more than one direction or rotation as a separate font.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ239E PRINT POSITION EXCEEDS THE LOGICAL PAGE BOUNDARY.

Explanation: The print position specified by POSITION subcommand exceeds the logical page boundary.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ240E NUMBER OF PRINTLINES, FIELDS, AND CONDITIONS EXCEEDS 65,535 IN ONE PAGEFORMAT.

Explanation: The total number of PRINTLINEs, FIELDs, and CONDITIONs exceeds 65,535 in one page format.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Reduce the number of PRINTLINEs, FIELDs, or CONDITIONs in the page format.

AKQ241E TOTAL LENGTH OF TEXT DATA EXCEEDS 65,534 BYTES.

Explanation: The total length of text may be up to 65,534 bytes.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ242E THE VALUE OF THE STARTING POSITION OF A RECORD IS TOO LARGE OR TOO SMALL.

Explanation: The START position of a record exceeds the maximum (65,535) or minimum (1) value.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ243E DBCS LENGTH IS NOT A MULTIPLE OF 2.

Explanation: The number of bytes of DBCS must be a multiple of two. This means that the value of the LENGTH parameter must be a multiple of two.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Specify a valid length or a valid DBCS.

AKQ244E INVALID CODE IS SPECIFIED IN THE TEXT.

Explanation: SBCS text must be within code range X'00' to X'FE'.

Valid double-byte character set (DBCS) codes are between X'41' and X'FE' for each byte. PPFA checks this range. Code X'4040' (blank) is the only exception. For example, the following are valid DBCS codes: X'4040', X'4141', X'41FE', X'FE41', X'FEFE'.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Specify a valid code.

AKQ245E HEXADECIMAL TEXT IS INVALID.

Explanation: Hexadecimal text is specified in an invalid format. Hexadecimal text must have an even length parameter and be in hexadecimal notation ('0' to 'F').

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Specify valid hexadecimal text.

AKQ246E NULL LITERAL IS SPECIFIED.

Explanation: The literal has no string.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Specify a valid literal.

AKQ247E KANJI NUMBER TEXT IS INVALID.

Explanation: A kanji number is specified in invalid format. Kanji number text must be a string of kanji numbers delimited by commas. Each kanji number must be a decimal number equal to a valid DBCS code, minus X'4000'.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Specify valid kanji number(s) in a valid format.

AKQ248E TEXT ATTRIBUTE CONFLICTS WITH FONT.

Explanation: SBCS font is specified for DBCS text (type G, K), or DBCS font is specified for SBCS text (type C).

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ249E TEXT ATTRIBUTE CONFLICTS WITH TEXT TYPE.

Explanation: The literal type conflicts with text type. SBCS literal is specified as type G or X, and DBCS literal is specified as type C, X, or K.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ250E TRC NUMBER IS DUPLICATED.

Explanation: The specified TRC number is duplicated in one page format.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Correct the error.

AKQ251W SPECIFIED LENGTH IS SHORTER THAN THE TEXT AND WAS TRUNCATED.

Explanation: The LENGTH parameter of the TEXT subcommand is shorter than the length of the specified literal, which is truncated to a specified length.

System Action: The operation continues, truncating the literal.

Operator Response: Check the truncation.

AKQ252E TEXT IS NOT THE LENGTH SPECIFIED BY THE LENGTH SUBCOMMAND.

Explanation: The length of the comparison text in a WHEN or OTHERWISE subcommand of a CONDITION command is not equal to the length specified by the LENGTH subcommand of that CONDITION command.

System Action: A page definition is not generated. The syntax check continues.

Operator Response: Change the comparison text or the LENGTH parameter so that they match.

AKQ253E TEXT IN THE 'WHEN' SUBCOMMAND IS TOO LONG.

Explanation: Constant text in a WHEN subcommand of a CONDITION command is too long to fit into an 8150-byte CCP structured field.

System Action: A page definition or form definition is not generated. The syntax check continues.

Operator Response: Shorten the field to 8000 bytes or fewer, and shorten the comparison text accordingly.

AKQ254E (text type) LITERAL WAS EXPECTED BUT (text type) WAS FOUND.

Explanation: An SBCS literal occurs where a DBCS one was expected, or vice versa.

System Action: A page definition or form definition is not generated. The syntax check continues.

Operator Response: In a FIELD command, do not use a DBCS literal without specifying a DBCS font. In a CONDITION command, do not mix SBCS and DBCS literals in the comparison text of a single WHEN subcommand.

AKQ255E INVOKE SPECIFIES A SIDE FOR WHICH NO PLACE SUBCOMMANDS PUT DATA.

Explanation: The N_UP PLACE subcommand contains an error that makes it incompatible with the value specified in the INVOKE subcommand. Either INVOKE BACK was specified, but PLACE *n* BACK was not specified, or INVOKE FRONT was specified, but PLACE *n* FRONT was not specified.

System Action: No form definition is generated. Processing continues.

User Response: Specify the same value (FRONT or BACK) for both the INVOKE and PLACE subcommands.

AKQ256E INCORRECT NUMBER OF PLACE SUBCOMMANDS.

Explanation: The required number of PLACE subcommands must be specified.

System Action: No form definition is generated. Processing continues.

User Response: When using N_UP PLACE subcommands with single-sided printing, the number of PLACE subcommands must equal the value specified on N_UP. When using duplex printing, the number of PLACE subcommands must equal two times the value specified on N_UP.

AKQ257W CONSTANT (*parameter*) FOUND WITH PLACE SUBCOMMAND.

Explanation: The CONSTANT (*parameter*) subcommand can not be specified when N_UP PLACE subcommands are specified.

System Action: A form definition is generated without constant forms control. The syntax check continues.

User Response: Delete the CONSTANT (*parameter*) from the FORMDEF or COPYGROUP command.

AKQ258W MORE THAN 122 OPERATION POSITIONS SPECIFIED FOR A FINISH OPERATION.

Explanation: More than 122 operation finishing positions are specified.

System Action: A form definition will be generated with 122 finishing positions. All others will be ignored.

User Response: Move extraneous operator position values.

AKQ259W OPCOUNT AND OPPOS VALUES SPECIFIED. OPCOUNT IGNORED.

Explanation: Both OPCOUNT and OPPOS are specified.

System Action: A form definition is not generated.

Operator Response: If OPCOUNT is specified, OPPOS is ignored. When using OPPOS for controlling the position of each operation on the operation axis, OPCOUNT is ignored.

AKQ260E (*insert-1*) not allowed with/on a (*insert-2*).

Explanation: This is a generic message which indicates a contextually incorrect combination of PPFA commands or subcommands.

System Action: A page definition is not generated.

Operator Response: Correct the incorrect parameter and rerun the job.

AKQ261E (*insert-1*) requires (*insert-2*).

Explanation: This is a generic message which indicates a missing PPFA command or subcommand.

System Action: A page definition is not generated.

Operator Response: Add the required parameter and rerun the job.

AKQ262E (*insert-1*) specifies a (*insert-2*) which is not a (*insert-3*).

Explanation: This is a generic message which indicates a contextually incorrect combination of PPFA commands or subcommands. For example that an ENDGRAPHIC command has specified or defaulted to a GRAPHID that does not match a floating DRAWGRAPHIC BOX or DRAWGRAPHIC LINE.

System Action: A page definition is not generated.

Operator Response: Correct the indicated problem.

AKQ263E (*insert-1*) exceeds (*insert-2*).

Explanation: This is a generic message which indicates an out of bound condition for some parameters. For example that a DRAWGRAPHIC CIRCLE is positioned off the logical page.

System Action: A page definition is not generated.

Operator Response: Correct the indicated problem.

AKQ264W (*insert-1*) is ignored (*insert-2*).

Explanation: This is a generic message which indicates that a contextually incorrect combination of PPFA commands or subcommands is clearly incorrect and is just ignored. For example, if a LINEONE subcommand was coded on a Record Format PAGEDEF (e.g. one using LAYOUT), the LINEONE subcommand would just be ignored.

System Action: A page definition is generated.

Operator Response: No action necessary unless the result is not what you wanted.

AKQ265W (*insert-1*) exceeds (*insert-2*).

Explanation: This is a generic message which indicates an out of bound condition for some parameters which is not necessarily critical. For example, when a DRAWGRAPHIC CIRCLE is positioned outside the margin boundary but still on the logical page.

System Action: A page definition is generated.

Operator Response: No action necessary unless the result is not what you wanted.

AKQ266E PAGEDEF CONTAINS BOTH LAYOUT AND PRINTLINE COMMANDS.

Explanation: Lines are placed in a record format page definition using LAYOUT commands, otherwise lines are placed with PRINTLINE commands. They cannot be mixed in the same page definition.

System Action: A page definition is not generated.

Operator Response: Remove either the LAYOUT commands or the PRINTLINE commands.

AKQ267E MORE THAN ONE DEFAULT PAGEHEADER OR PAGETRAILER IN A PAGEFORMAT.

Explanation: Only one LAYOUT DEFAULT PAGEHEADER or PAGETRAILER can be coded in a page definition.

System Action: A page definition is not generated.

Operator Response: Remove one of the duplicates.

AKQ268E SPECIFIED MARGINS FOR THIS PAGEFORMAT OVERLAP.

Explanation: Either the left margin is defined on or right of the right margin or the top margin is defined on or below the bottom margin.

System Action: A page definition is not generated.

Operator Response: Redefine the margins so that they do not overlap.

AKQ269E A RECORD FORMAT PAGEDEF REQUIRES AT LEAST ONE FONT DEFINITION.

Explanation: At least one font must be defined whether or not one is referenced.

System Action: A page definition is not generated.

Operator Response: Define a font.

AKQ2MMS NUMBER OF MESSAGES EXCEEDS THE 270 ALLOWED LIMIT. PROGRAM TERMINATES.

Explanation: PPFA allows only 269 messages, plus this one. When this limit is reached, the messages are printed and the program terminates.

System Action: The program terminates.

Operator Response: Correct the PPFA code for the messages issued and redo.

AKQ301I PAGE PRINTER FORMATTING AID ENDED, MAX RETURN CODE = (max return code).

Explanation: This message accompanies the output listings of all form definitions and page definitions with the maximum return code for that particular object. Only when the return code is less than 8 is the object generated.

System Action: None.

Operator Response: None.

AKQ302I NO ERRORS FOUND IN (resource name) DEFINITION.

Explanation: One definition is processed. No statements were flagged in this definition.

System Action: This definition is generated, and stored or replaced.

Operator Response: None.

AKQ303S NO CONTROL STATEMENT(S) ARE SPECIFIED IN INPUT DATA.

Explanation: There are no control statements in the input data.

System Action: The operation terminates.

Operator Response: Specify a valid PPFA command.

AKQ304S DEFINITION STATEMENT IS NOT SPECIFIED.

Explanation: There is no FORMDEF or PAGEDEF command in the system input command stream.

System Action: The operation terminates.

Operator Response: Specify valid definition commands.

AKQ305S THIS DEFINITION IS NOT STORED BECAUSE MEMBER ALREADY EXISTS.

Explanation: This form definition or page definition is not saved because a file with the same name already exists in the directory (REPLACE option is NO).

System Action: A page definition or form definition is not generated. The syntax check continues to next definition.

Operator Response: Check the specified form definition or page definition name, and specify REPLACE subcommand YES. Specify another form definition or page definition name.

AKQ311I FORMDEF (form definition name) IS GENERATED AND STORED. MAX RETURN CODE = (max return code).

Explanation: The form definition is generated and stored.

System Action: A form definition is generated.

Operator Response: None.

AKQ312I FORMDEF (command name) IS GENERATED AND REPLACED. MAX RETURN CODE = (max return code).

Explanation: The form definition is generated and is replaced. The maximum return code is listed.

System Action: A form definition is generated.

Operator Response: None.

AKQ313E FORMDEF (form definition name) IS NOT GENERATED. MAX RETURN CODE = (max return code).

Explanation: The form definition is not generated because of an error. The error is indicated by another message.

System Action: A form definition is not generated.

Operator Response: Correct the error.

AKQ321I PAGEDEF (page definition name) IS GENERATED AND FILED. MAX RETURN CODE = (max return code).

Explanation: The page definition is generated and stored.

System Action: A page definition is generated.

Operator Response: None.

AKQ322I PAGEDEF (page definition name) IS GENERATED AND REPLACED. MAX RETURN CODE = (max return code).

Explanation: The page definition is generated and is replaced.

System Action: A page definition is generated.

Operator Response: None.

AKQ323E PAGEDEF (page-definition name) IS NOT GENERATED. MAX RETURN CODE = (max return code).

Explanation: The page definition is not generated because of an error. The error is indicated by another message.

System Action: A page definition is not generated.

Operator Response: Correct the error.

AKQ350T AN UNRECOVERABLE PROGRAM ERROR OCCURRED.

Explanation: There was an error in PPFA logic.

System Action: The operation terminates.

Operator Response: Use local problem-reporting procedures to report this message.

AKQ360E FONT COMMAND DOES NOT CONTAIN SUFFICIENT INFORMATION.

Explanation: The FONT command referred to does not contain enough information to generate a valid MCF. This is caused by having a CS parameter without a CP parameter, or vice versa.

System Action: A page definition is not generated.

Operator Response: Correct the referenced FONT command.

AKQ361E FONT COMMAND SPECIFIES CONFLICTING PARAMETERS.

Explanation: A FONT is specified in more than one way, only one of the following is allowed:
Coded Font
Character Set, Code Page pair (CS and CP parameters)
GRID

System Action: A page definition is not generated.

Operator Response: Correct the referenced FONT command.

AKQ362E FONT RATIO SPECIFIED WITHOUT FONT HEIGHT.

Explanation: To scale a font, both the HEIGHT and RATIO **must** be specified. If a RATIO subcommand is found without a HEIGHT subcommand, the scaling information can not be calculated by PPFA.

System Action: A page definition is not generated.

Operator Response: Correct the referenced FONT command.

AKQ363W HEIGHT SPECIFIED, WIDTH IN GRID IGNORED.

Explanation: You have specified both a HEIGHT and GRID in the FONT command.

System Action: A page definition is not generated.

Operator Response: Correct the referenced FONT command.

AKQ364E INVALID DIRECTION WITH RELATIVE PRINTLINE

Explanation: You specified an incorrect direction with the relative printline in your page definition source. The field direction must match the direction of the printline. The printline direction must be ACROSS.

System Action: A page definition is not generated.

Operator Response: Correct the referenced DIRECTION subcommand.

AKQ365W COLOR AND EXTENDED COLOR SPECIFIED

Explanation: Both COLOR and one of the extended color keywords (RGB, CMYK, HIGHLIGHT, CIELAB) was specified.

System Action: Both requests are placed into the output resource. Output depends on printer function.

Operator Response: If output does not print as expected, remove one of the specifications.

AKQ370E BARCODE NAME WAS NOT PREVIOUSLY DEFINED.

Explanation: You attempted to reference a barcode name that had not been previously defined.

System Action: A page definition is not generated.

Operator Response: Correct the referenced BARCODE subcommand of the FIELD command.

AKQ371E BARCODE NAME WAS PREVIOUSLY DEFINED.

Explanation: You attempted to define a barcode name that had been previously defined.

System Action: A page definition is not generated.

Operator Response: Correct the referenced BARCODE subcommand of the FIELD command.

AKQ372W BARCODE MODIFICATION UNDEFINED FOR TYPE GIVEN.

Explanation: You specified a modification for a bar code that is not defined for the type specified.

See "Appendix D. More About Bar Code Parameters" on page 349

on page 349 for more information.

System Action: A page definition is generated as specified. This is done so that, as new bar code types and modifications are introduced, you can create page definitions for them. However, you will receive this warning, because the specification could also be an error.

Operator Response: Correct the referenced BARCODE subcommand of the FIELD command, if appropriate.

AKQ373W BARCODE TYPE IS UNDEFINED.

Explanation: You specified a bar code type that is not defined.

System Action: A page definition is generated as specified. This is done so that, as new bar code types and modifications are introduced, you can create page definitions for them. However, you will receive this warning, because this specification could also be an error.

Operator Response: Correct the referenced BARCODE subcommand of the FIELD command, if appropriate.

AKQ374W INVALID DATA LENGTH FOR SELECTED BARCODE TYPE AND MODIFICATION.

Explanation: You specified a data length for a defined barcode type and modification that is invalid for that combination of type and modification.

See "Appendix D. More About Bar Code Parameters" on page 349 for more information.

System Action: A page definition is generated as specified. This is done so that, as new bar code types and modifications are introduced, you can create page definitions for them. However, you will receive this warning, because this specification could also be an error.

Operator Response: Correct the referenced BARCODE subcommand of the FIELD command, if appropriate.

AKQ401E EXEC PARAMETER IS INVALID.

Explanation: The program parameter specification is invalid.

System Action: A page definition or form definition is not generated. The syntax check continues.

Operator Response: Specify a valid program parameter.

AKQ402T ERROR OCCURRED DURING ATTEMPT TO OBTAIN STORAGE

Explanation: conditions generate this message:

1. Exceeds the available size to hold the compiled data for the page definition and form definition.
2. Insufficient available disk space on the file system to write the output of the compiler.
3. Exceeds the limit of 269 user errors generated within a PPFA source file.

System Action: The operation terminated.

Operator Response:

1. Increase the region or VM program size.
2. Increase the size of the file system or specify a directory on another file system that has more disk space.
3. Fix the errors reported to this point and re-run PPFA.

AKQ403T ERROR OCCURRED DURING ATTEMPT TO FREE STORAGE.

Explanation: A system error occurred while PPFA attempted to free disk space at the end of an execution.

System Action: The operation terminates.

Operator Response: Use local problem-reporting procedures to report this message.

AKQ404T SYSIPT OPEN FAILURE.

Explanation: SYSIPT cannot be opened.

System Action: The operation terminates.

Operator Response: Assign a valid input data file.

AKQ405T INSUFFICIENT STORAGE TO EXECUTE PPFA.

Explanation: The region size is too small to execute PPFA.

System Action: The operation terminates.

Operator Response: Increase the region size available to the job.

AKQ410T (Librarian error message).

Explanation: The message describes a librarian error.

System Action: The operation terminates.

Operator Response: Contact a system programmer.

AKQ411T FORMDEF LIBRARY OPEN FAILURE.

Explanation: The FORMDEF library cannot be opened.

System Action: The operation terminates.

Operator Response: Assign a valid FORMDEF library.

AKQ412T FORMDEF LIBRARY I/O ERROR.

Explanation: An I/O error occurred during an attempted access of a form definition directory.

System Action: The operation terminates.

Operator Response: Check the permissions of the directory. If you do not have access, contact the owner of the directory. If this does not resolve the problem, contact a system programmer.

AKQ413T FORMDEF DIRECTORY CANNOT BE UPDATED.

Explanation: The FORMDEF member cannot be registered on the directory.

System Action: The operation terminates.

Operator Response: Contact a system programmer.

AKQ414T FORMDEF LIBRARY CLOSE FAILURE.

Explanation: A form definition directory cannot be closed.

System Action: The operation terminates.

Operator Response: Use local problem-reporting procedures to report this message.

AKQ415T PAGEDEF LIBRARY OPEN FAILURE.

Explanation: The PAGEDEF library cannot be opened.

System Action: The operation terminates.

Operator Response: Assign a valid PAGEDEF library.

AKQ416T PAGEDEF LIBRARY I/O ERROR.

Explanation: I/O error occurs during an attempted access of a page definition directory.

System Action: The operation terminates.

Operator Response: Check the permissions of the directory. If you do not have access, contact the owner of the directory. If this does not resolve the problem, contact a system programmer.

AKQ417T PAGEDEF DIRECTORY CANNOT BE UPDATED.

Explanation: A page definition file cannot be registered on the directory.

System Action: The operation terminates.

Operator Response: Contact a system programmer.

AKQ418T PAGEDEF LIBRARY CLOSE FAILURE.

Explanation: A page definition directory cannot be closed.

System Action: The operation terminates.

Operator Response: Use local problem-reporting procedures to report this message.

AKQ420T SYSTEM ERROR. ABEND CODE = (ABEND code).

Explanation: System forces PPFA to terminate abnormally.

System Action: The operation terminates.

Operator Response: Contact a system programmer. Refer to the documentation for your operating system.

AKQ421T FORMDEF LIBRARY IS FULL.

Explanation: The file system into which PPFA attempted to save the form definition is full.

System Action: The operation terminates.

Operator Response: Increase the size of the file system or specify a directory on a file system that has more disk space.

AKQ422T PAGEDEF LIBRARY IS FULL.

Explanation: The file system into which PPFA attempted to save the page definition is full.

System Action: The operation terminates.

Operator Response: Increase the size of the file system or specify a directory on a file system that has more disk space.

AKQ501T SYSIN OPEN FAILURE.

Explanation: The PPFA input source file cannot be opened.

System Action: The operation terminates.

Operator Response: Specify a valid input source file.

AKQ502T SPANNED RECORD OF SYSIN IS NOT SUPPORTED.

Explanation: The spanned record of the PPFA input source file is not supported.

System Action: The operation terminates.

Operator Response: Specify a valid input record format.

AKQ503T UNDEFINED LENGTH RECORD OF SYSIN IS NOT SUPPORTED.

Explanation: An undefined length record of PPFA input source file is not supported.

System Action: The operation terminates.

Operator Response: Specify a valid input record format.

AKQ504T LOGICAL RECORD LENGTH OF SYSIN EXCEEDS LIMIT.

Explanation: The logical record length of the PPFA input source file exceeds limit which is 100 bytes except for the OS/390 variable length which is 104 and AIX which is 254.

System Action: The operation terminates.

Operator Response: Correct the logical record length of the file.

AKQ510T FORMDEF/PAGEDEF LIBRARY OPEN FAILURE.

Explanation: The FORMDEF or PAGEDEF directory cannot be opened.

System Action: The operation terminates.

Operator Response: Specify a valid FORMDEF or PAGEDEF or check to make sure that the directory is correct.

AKQ511T I/O ERROR OCCURRED DURING (FORMDEF/PAGEDEF) DIRECTORY SEARCH. RETURN CODE = (return code) REASON CODE = (reason code)

Explanation: I/O error occurred while performing FIND function.

System Action: The operation terminates.

Operator Response: Contact a system programmer.

AKQ512T LOGICAL RECORD LENGTH OF FORMDEF/PAGEDEF EXCEEDS LIMIT.

Explanation: The logical record length exceeds maximum or minimum value.

System Action: The operation terminates.

Operator Response: Specify a filename that has a valid record length.

AKQ513T BLOCK SIZE OF FORMDEF/PAGEDEF EXCEEDS LIMIT.

Explanation: The block size exceeds maximum or minimum value.

System Action: The operation terminates.

Operator Response: Assign a filename that has a valid block size.

AKQ514T UNDEFINED LENGTH RECORD IS NOT SUPPORTED IN FORMDEF/PAGEDEF LIBRARY.

Explanation: An undefined length record is not supported in FORMDEF/PAGEDEF directory.

System Action: The operation terminates.

Operator Response: Assign a valid record format.

AKQ515T FIXED LENGTH RECORD IS NOT SUPPORTED IN FORMDEF/PAGEDEF LIBRARY.

Explanation: The fixed length record is not supported in the FORMDEF or PAGEDEF library.

System Action: The operation terminates.

Operator Response: Assign a valid record format.

AKQ516T NO CONTROL CHARACTER RECORD IS SUPPORTED IN FORMDEF/PAGEDEF LIBRARY.

Explanation: No control character record is supported in FORMDEF/PAGEDEF directory.

System Action: The operation terminates.

Operator Response: Assign a valid record format.

AKQ517T NO SPACE IN FORMDEF/PAGEDEF DIRECTORY.

Explanation: No space was available in the FORMDEF directory or the PAGEDEF directory to add or replace the resource.

System Action: The operation terminates.

Operator Response: Increase the directory space or specify a directory on another file system that has more disk space.

AKQ518T I/O ERROR OCCURRED WHILE UPDATING FORMDEF/PAGEDEF DIRECTORY. RETURN CODE=(return code). REASON CODE = (reason code).

Explanation: A permanent I/O error was detected, or the specified data control block is not opened, or insufficient disk space exists to perform the write function.

System Action: The operation terminates.

Operator Response: Contact a system programmer.

AKQ519T I/O ERROR OCCURRED DURING WRITE.

Explanation: The error message is displayed.

System Action: The operation terminates.

Operator Response: Contact a system programmer.

AKQ520T SPANNED RECORD IS NOT SUPPORTED IN FORMDEF/PAGEDEF LIBRARY.

Explanation: The spanned record is not supported in the FORMDEF or PAGEDEF library.

System Action: The operation terminates.

Operator Response: Remove the SPAN attribute and assign a valid dataset.

AKQ522T BLOCK SIZE IS NOT SPECIFIED FOR FORMDEF/PAGEDEF DATA SET.

Explanation: A block size is not specified for FORMDEF/PAGEDEF data set.

System Action: The operation terminates.

Operator Response: Specify a BLKSIZE in the DD statement.

AKQ540T SYSTEM ABEND (code) OCCURRED IN PPFA PROCESS.

Explanation: A system ABEND (code) occurred in PPFA/OS/390 process. Termination processing was performed by the ESTAE macro instruction.

System Action: The operation terminates.

Operator Response: Contact a system programmer. Refer to System Messages for your operating system.

AKQ541T USER ABEND (code) OCCURRED IN PPFA/OS/390 PROCESS.

Explanation: A user ABEND (code) occurred in PPFA/OS/390 process. Termination processing was performed by the ESTAE macro instruction.

System Action: The operation terminates.

Operator Response: Use local problem-reporting procedures to report this message.

AKQ600T INPUT FILENAME NOT SPECIFIED.

Explanation: You did not specify an input filename.

System Action: The operation terminates.

Operator Response: Enter the input filename.

AKQ601T INPUT FILETYPE NOT SPECIFIED.

Explanation: You did not specify an input filetype.

System Action: The operation terminates.

Operator Response: Enter the input filetype.

AKQ602T COMMAND SYNTAX IS NOT VALID.

Explanation: The command syntax you entered was not accepted.

System Action: The operation terminates.

Operator Response: Enter a valid command.

AKQ603T FILEMODE FOR (FORMDEF/PAGEDEF/LISTING) IS INVALID.

Explanation: You entered an invalid filemode for FORMDEF, PAGEDEF, or LISTING.

System Action: The operation terminates.

Operator Response: Enter a valid file extension.

AKQ604T INVALID PARAMETER IS SPECIFIED IN (FORMDEF/PAGEDEF/LISTING/SIZE) OPTION.

Explanation: You entered an invalid parameter for FORMDEF, PAGEDEF, LISTING, or SIZE.

System Action: The operation terminates.

Operator Response: Enter a valid option parameter.

AKQ605T (FORMDEF/PAGEDEF/LISTING/SIZE) KEYWORD IS DUPLICATED.

Explanation: You entered a duplicate keyword for FORMDEF, PAGEDEF, LISTING, or SIZE.

System Action: The operation terminates.

Operator Response: Enter a unique keyword.

AKQ606T FILETYPE FOR (FORMDEF/PAGEDEF/LISTING) NOT SPECIFIED.

Explanation: The filetype for FORMDEF, PAGEDEF, or LISTING was not entered.

System Action: The operation terminates.

Operator Response: Enter an appropriate filetype.

AKQ607T INVALID KEYWORD SPECIFIED.

Explanation: The keyword you entered was not accepted.

System Action: The operation terminates.

Operator Response: Enter a valid keyword.

AKQ608T INVALID SIZE PARAMETER SPECIFIED.

Explanation: The size parameter specified is not valid.

System Action: The operation terminates.

Operator Response: Enter a valid size parameter.

AKQ610T SIZE PARAMETER VALUE EXCEEDS THE ALLOWABLE MAXIMUM.

Explanation: The size entered exceeds the maximum allowable.

System Action: The operation terminates.

Operator Response: Enter a valid size value.

AKQ611T SIZE PARAMETER VALUE IS TOO SMALL.

Explanation: The size entered is too small for executing in PPFA/VM.

System Action: The operation terminates.

Operator Response: Enter a valid size value.

AKQ612T INVALID FILE IDENTIFIER '*' SPECIFIED FOR INPUT FILE.

Explanation: '*' is specified for input filename or filetype.

System Action: The operation terminates.

Operator Response: Enter a valid filename or filetype.

AKQ613T SIZE PARAMETER VALUE IS MISSING.

Explanation: You did not specify a size parameter

System Action: The operation terminates.

Operator Response: Specify a valid size parameter.

AKQ620T INPUT FILE WAS NOT FOUND.

Explanation: The input filename entered was not found.

System Action: The operation terminates.

Operator Response: Correct the input filename.

AKQ621T NO READ/WRITE (file mode) DISK ACCESSED FOR (INPUT/LISTING/FORMDEF/PAGEDEF /OUTPUT).

Explanation: The disk on which the file is saved cannot be read from or written to because it either was not accessed or was accessed using an invalid access mode.

System Action: The operation terminates.

Operator Response: Access the file system using a valid access mode.

AKQ622T INPUT FILE EXCEEDS THE ALLOWABLE LOGICAL RECORD LENGTH MAXIMUM.

Explanation: The logical record length of the input file exceeds the limit which is 100 bytes except the OS/390 variable record length is 104 and AIX is 254.

System Action: The operation terminates.

Operator Response: Correct the logical record length of the file.

AKQ624T I/O ERROR OCCURRED IN (AKQINIO/AKQLBIO/AKQPRIO) MODULE. RC = (return code from FWRITE/FGETS macro instruction).

Explanation: An I/O error occurred during either FGETS or FWRITE processing of module AKQINIO, AKQLBIO, or AKQPRIO.

System Action: The operation terminates.

Operator Response: Contact your system programmer. Refer to the return code in *AIX Operating System Messages*

AKQ625T DISK (file mode) IS FULL.

Explanation: Not enough space is available on the specified file system to write the file.

System Action: The operation terminates.

Operator Response: Erase some files from the specified file disk and re-execute.

AKQ639T ABEND EXIT ROUTINE FAILED TO EXECUTE. RC = (return code from ABNEXIT macro)

Explanation: ABEND exit routine cannot be established.

System Action: The operation terminates.

Operator Response: Contact your system programmer. Refer to the return code in *AIX Operating System Messages*

AKQ640T SYSTEM ABEND (code) OCCURRED IN PPFA/VM PROCESS.

Explanation: A system ABEND occurred during processing. The ABEND exit routine ended processing.

System Action: The operation terminates.

Operator Response: Use local problem-reporting procedures to report this message.

AKQ641T USER ABEND (code) OCCURRED IN PPFA/VM PROCESS.

Explanation: A user-initiated ABEND occurred during processing. The ABEND exit routine ended the processing.

System Action: The operation terminates.

Operator Response: Use local problem-reporting procedures to report this message.

AKQ700I SIZE PARAMETER IS NO LONGER NECESSARY IN PPFA/370.

Explanation: The storage required to contain the messages and control blocks is not automatically set at 32K and 128K respectively. If the control block storage is used up, an additional 128K will be gotten and chained to the previous. All storage necessary to perform the compile will be obtained during processing.

System Action: The compile process continues.

Operator Response: None.

Notices

Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services or features discussed in this document in your country. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program or service is not intended to state or imply that only that IBM product, program or service may be used. Any functionally equivalent product, program or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10594-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer or express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the material for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one), and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Drop 001W
Boulder, CO 80301
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Programming Interfaces

This publication includes documentation of intended Programming Interfaces that let the customer write programs to obtain the services of the Page Printer Formatting Aid (PPFA).

Trademarks

The following terms appear in this publication and are either trademarks or registered trademarks of the IBM Corporation:

- Advanced Function Common Control Unit
- Advanced Function Presentation
- Advanced Function Printing
- AFCCU
- AFP
- AIX
- AIX/6000
- OS/400
- Bar Code Object Content Architecture
- BCOCA
- IBM
- Infoprint®

Infoprint Server for OS/390®
Intelligent Printer Data Stream
IPDS
Mixed Object Document Content Architecture
MO:DCA
PrintManager
Print Services Facility
PSF
OS/400
RISC System/6000
S/370

The following terms appear in this publication and are trademarks of other companies:

- Windows and Windows NT are registered tradmarks of the Microsoft Corporation.
- UNIX is a registered trademark of The Open Group in the United States and other countries.

EuroReady

The Page Printer Formatting Aid (PPFA) is capable of processing data containing the euro sign. Font character sets and code pages that contain and map the euro sign consistently with the application must be present either in a host library or in the printer. AFP fonts that support the euro sign are included in the AFP Font Collection.

Year 2000 Ready

PPFA does not have date dependencies and is therefore Year 2000 ready. When used in accordance with its associated documentation, PPFA is capable of correctly processing, providing, and receiving date data within and between the twentieth and twenty-first centuries, provided all other products used with PPFA (including software, hardware and firmware) properly exchange accurate date data with it.

Glossary

Source Identifiers

This publication includes terms and definitions from the *IBM Dictionary of Computing*, ZC20-1699.

Definitions reprinted from the *American National Dictionary for Information Processing Systems* are identified by the symbol (A) following the definition.

Definitions reprinted from a published section of the International Organization for Standardization's *Vocabulary—Information Processing* or from a published section of the ISO *Vocabulary—Office Machines* are identified by the symbol (1) following the definition. Because many ISO definitions are also reproduced in the *American National Dictionary for Information Processing Systems* , ISO definitions may also be identified by the symbol (A).

Definitions reprinted from working documents, draft proposals, or draft international standards of ISO Technical Committee 97, Subcommittee 1 (Vocabulary) are identified by the symbol (T) following the definition, indicating that final agreement has not yet been reached among its participating members.

Definitions that are specific to IBM products are so labeled, for example, "In SNA," or "In the 3820."

The following definitions are provided as supporting information only, and are not intended to be used as a substitute for the semantics described in the body of this document.

References

The following cross-references are used in this glossary:

Contrast with. This refers to a term that has an opposed or substantively different meaning.

See. This refers the reader to multiple-word terms that have the same last word.

See also. This refers the reader to related terms that have a related, but not synonymous, meaning.

Synonym for. This indicates that the term has the same meaning as a preferred term, which is defined in its proper place in the glossary.

Synonymous with. This is a backward reference from a defined term to all other terms that have the same meaning.

Terms

A

ACIF. (1) AFP conversion and indexing facility. (2) A print server utility program that converts a print file into AFP, MO:DCA-P, creates an index file for input data, and collects resources used by an AFP document into a separate file.

advanced function printing (AFP). The ability of program products to place text and image data at any addressable point on the page.

AFP. Advanced function printing.

AIX operating system. IBM's implementation of the UNIX operating system. The RS/6000© system, amoung others, runs the AIX operating system.

all-points addressability. The capability to address, reference, and position data elements at any addressable position in a presentation space or on a physical medium. An example of all points addressability is the positioning of text, graphics, and images at any addressable point on the physical medium. See also *picture element*.

all-points-addressable mode. Synonym for *page mode*.

alphanumeric string. A sequence of characters consisting solely of the letters a through z and the numerals 0 through 9.

American National Standards Institute (ANSI). An organization consisting of producers, consumers, and general interest groups. ANSI establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States. It is the United States constituent body of the International Organization for Standardization (ISO).

ANSI. See *American National Standards Institute*.

APA. All points addressable.

application. (1) The use to which an information system is put. (2) A collection of software components used to perform specific types of work on a computer.

application program. A program written for or by a user that applies to the user's work.

ascender. The parts of certain lowercase letters, such as b, d, or f, which at zero-degree character rotation rise above the top edge of other lowercase letters such as a, c, and e. Contrast with *descender*.

attribute. A property or characteristic of one or more constructs. For example, *character attribute*, *color attribute*, *current drawing attributes*, *default drawing attributes*, *line attributes*, *marker attributes*, and *pattern attributes*.

B

bar. In bar codes, the darker element of a printed bar code symbol.

bar code. An array of parallel rectangular bars and spaces that together represent data elements or characters of a particular type. The bars and spaces are arranged in a predetermined pattern following unambiguous rules defined by the symbology.

bar code command set. In the IPDS architecture, a collection of commands used to present bar code symbols in a page, page segment, or overlay.

bar code density. The number of characters per inch (cpi) in a bar code symbology. In most cases, the range is three to ten cpi.

bar code object area. The rectangular area on a logical page into which a bar code presentation space is mapped.

Bar Code Object Content Architecture (BCOCA). An architected collection of constructs used to interchange and present bar code data.

bar code symbol. A combination of characters including start and stop characters, quiet zones, data characters, and check characters required by a particular bar code type, that form a complete, scannable entity.

bar code symbology. A bar code language. Bar code symbologies are defined and controlled by various industry groups and standards organizations. Bar code symbologies are described in public domain bar code specification documents. Synonymous with *symbology*. Examples of bar code symbology include: *Canadian Grocery Product Code (CGPC)*, *European Article Numbering (EAN)*, *Japanese Article Numbering (JAN)*, and *Universal Product Code (UPC)*.

bar height. In bar codes, the bar dimension perpendicular to the bar width. Synonymous with *bar length* and *height*.

bar length. In bar codes, the bar dimension perpendicular to the bar width. Synonymous with *bar length* and *height*.

bar width. In bar codes, the thickness of a bar measured from the edge closest to the symbol start character to the trailing edge of the same bar.

baseline. A conceptual line with respect to which successive characters are aligned.

baseline direction. The direction in which successive lines of text appear on a logical page.

BCOCA. See *Bar Code Object Content Architecture*.

bin. The standard-size paper source on cut-sheet page printers that have more than one paper source. Each printer is set up with either A4 or letter-size paper as the standard size. Contrast with *cassette*.

BITS. A data type for architecture syntax, indicating one or more bytes to be interpreted as bit string information.

body. (1) On a printed page, the area between the top and bottom margins that can contain data. (2) In a book, the portion between the front matter and the back matter.

boldface. (1) A heavy-faced type. (2) Printing in heavy-faced type.

C

carriage control character. If present, the first character of an output record (line) that is to be printed or spaced; it determines how many lines should be skipped before the line.

cassette. A removable storage device that is the source for alternate sizes of paper on page printers that have more than one paper source. Contrast with *bin*.

CDB2OF7. A parameter that specifies a bar code type of Codabar, 2-of-7, Automatic Identification Manufacturers Uniform Symbol Specification-Codabar.

CGPC. See *Canadian Grocery Product Code*.

CHAR. A data type for architecture syntax, indicating one or more bytes to be interpreted as character information.

character. (1) A member of a set of elements used for the organization, control, or representation of data. A character can be either a graphic character or a control character. (2) In bar codes, a single group of bars and

spaces that represent an individual number, letter, punctuation mark, or other symbol.

character ascender. See *ascender*.

character attribute. A characteristic that controls the appearance of a character or character string.

character baseline. A conceptual reference line that is coincident with the X axis of the character coordinate system.

character code. An element of a code page or a cell in a code table to which a character can be assigned. The element is associated with a binary value. The assignment of a character to an element of a code page determines the binary value that will be used to represent each occurrence of the character in a character string.

character descender. See *descender*.

character identifier. The unique name for a graphic character.

character rotation. The alignment of a character with respect to its character baseline, measured in degrees in a clockwise direction. Examples are 0º, 90º, 180º, and 270º. Zero-degree character rotation exists when a character is in its customary alignment with the baseline. Character rotation and font inline sequence are related in that character rotation is a clockwise rotation; font inline sequence is a counter-clockwise rotatione.

character set. A finite set of different graphic or control characters that is complete for a given purpose. For example, the character set in ISO Standard 646, *7-bit Coded Character Set for Information Processing Interchange*

character set attribute. An attribute used to specify a coded font.

code page. (1) A resource object containing descriptive information, graphic character identifiers, and code points corresponding to a coded graphic character set. Graphic characters can be added over time; therefore, to specifically identify a code page, both a GCSGID and a CPGID should be used. See also *coded graphic character set*. (2) A set of assignments, each of which assigns a code point to a character. Each code page has a unique name or identifier. Within a given code page, a code point is assigned to one character. More than one character set can be assigned code points from the same code page.

Code Page Global Identifier (CPGID). A unique code page identifier that can be expressed as either a two-byte binary or a five-digit decimal value.

code point. A unique bit pattern that can serve as an element of a code page or a site in a code table, to which a character can be assigned. The element is associated with a binary value. The assignment of a character to an element of a code page determines the binary value that will be used to represent each occurrence of the character in a character string. Code points are one or more bytes long.

Code39. A bar code symbology characterized by a variable-length, bidirectional, discrete, self-checking, alphanumeric code. Three of the nine elements are wide and six are narrow. It is the standard for LOGMARS (the Department of Defense) and the AIAG.

Code128. A bar code symbology characterized by a variable-length, alphanumeric code with 128 characters.

Codabar. A bar code symbology characterized by a discrete, self-checking, numeric code with each character represented by a stand-alone group or four bars and three spaces between them.

coded font. (1) A resource containing elements of a code page and a font character set, used for presenting text, graphics character strings, and bar code HRI. See also *code page* and *font character set*. (2) In FOCA, a resource containing the resource names of a valid pair of font character set and code page resources. The graphic character set of the font character set must match the graphic character set of the code page for the coded font resource pair to be valid. (3) In the IPDS architecture, a raster font resource containing code points that are directly paired to font metrics and the raster representation of character shapes, for a specific graphic character set. (4) In the IPDS architecture, a font resource containing descriptive information, a code page, font metrics, and a digital-technology representation of character shapes for a specific graphic character set.

Coded Graphic Character Set Global Identifier (CGCSGID). A four-byte binary or a ten-digit decimal identifier consisting of the concatenation of a GCSGID and a CPGID. The CGCSGID identifies the code point assignments in the code page for a specific graphic character set, from among all the graphic characters that are assigned in the code page.

color attribute. An attribute that affects the color values provided in a graphics primitive, a text control sequence, or an IPDS command. Examples of color attributes are foreground color and background color.

color model. The method by which a color is specified. For example, the RGB color space specifies color in terms of three intensities for red (R), green (G), and blue (B).

command. A request for performance of an operation or execution of a program. In Page Printer Formatting Aid, commands are control statements for major formatting functions. For example, FORMDEF and COPYGROUP are commands. Commands are further specified by subcommands and parameters.

command stream. The sequence of Page Printer Formatting Aid commands that is submitted with the job control statements in a Page Printer Formatting Aid execution. The commands and subcommands are the control statements that define the object or objects to be generated.

compatibility mode. Use of Table Reference Characters (TRCs) that are acceptable to line printers and page printers and that access page definitions with little or no change to the user's data or to the job command stream. Contrast with *page mode*.

composed-text data file. A file containing text data and text control information that dictates the format, placement, and appearance of the data to be printed.

conditional processing. A page definition function that allows input data records to partially control their own formatting.

construct. An architected set of data such as a structured field or a triplet.

control character. (1) A character that denotes the start, modification, or end of a control function. A control character can be recorded for use in a subsequent action, and it can have a graphic representation. See also *character*. (2) A control function the coded representation of which consists of a single code point.

copy group. A subset of a form definition containing a set of controls for the physical pages of a printout. Such functions as the selection of either of two paper sources on the page printer, the use of duplex printing, or the positioning of the reference point for all printing on the sheet are available in the copy group.

cm. Centimeters.

CMS. Conversational Monitor System.

cpi. Characters per inch.

cut-sheet media. Unconnected sheets. Contrast with *continuous-form media*.

D

data map. An internal object whose structured fields control the formatting of data on a logical page of a printout. Created by a PAGEDEF command or a PAGEFORMAT command.

data stream. A continuous stream of data that has a defined format. An example of a defined format is a structured field.

DBCS. Double-byte character set.

default. Pertaining to an attribute, value, or option that is assumed when none is explicitly specified and one is needed to continue processing.

density. The number of characters per inch (cpi) in a bar code symbology. In most cases, the range is three to ten cpi.

descender. In a font, the distance from the baseline to the bottom of the character box. This value may differ for different characters in a given font. Contrast with *ascender*.

direction. The print position of data in a logical page, line, or field. In Page Printer Formatting Aid, the ultimate reference point for all direction controls on a page is the hardware origin. Secondary and tertiary reference points are possible as well, allowing more than one print direction on a page.

document. (1) A machine-readable collection of one or more objects that represents a composition, a work, or a collection of data. (2) A publication or other written material.

double-byte character set (DBCS). A character set, such as a set of Japanese ideographs, requiring two bytes to identify each character.

duplex printing. Printing on both sides of a sheet.

E

EAN. See *European Article Numbering*.

EAN2SUP. A parameter that specifies a bar code type of European Article Numbering, Two-digit Supplemental.

EAN5SUB. A parameter that specifies a bar code type of European Article Numbering, Five-digit Supplemental.

EAN8. A parameter that specifies a bar code type of European Article Numbering 8 (includes Japanese Article Numbering-short).

EAN13. A parameter that specifies a bar code type of European Article Numbering 13 (includes Japanese Article Numbering-standard).

EBCDIC. See *Extended Binary-Coded Decimal Interchange Code*.

electronic overlay. In IBM Print Server Facility, a collection of constant data that are electronically composed in the host processor and can be merged with variable data on a sheet during printing. Contrast with *page segment*. See also *overlay*, *preprinted form*.

European Article Numbering (EAN). The bar code symbology used to code grocery items in Europe.

Extended Binary-Coded Decimal Interchange Code (EBCDIC). A coded character set that consists of eight-bit coded characters.

external library resource (member). Objects that can be used by other program products while running print jobs; for example, coded fonts, code pages, font character sets, form definitions, page definitions, and page segments. Synonym for *resource object*.

external object. Synonym for *resource object*.

F

FCB. Forms control buffer.

field. (1) In a record, a specified area used for a particular class of data; for example, a group of character positions used to enter or display wage rates on a screen. (2) In Page Printer Formatting Aid, any area of a record singled out for particular formatting treatment.

field processing. Mapping individual fields to a page of output with special formatting controls.

file. A named set of records stored or processed as a unit. (T)

fixed medium information. Information that can be applied to a sheet by a printer or printer-attached device that is independent of data provided through the data stream. Fixed medium information does not mix with the data provided by the data stream and is presented on a sheet either before or after the text, image, graphics, or bar code data provided within the data stream. Fixed medium information can be used to create “pre-printed forms”, or other types of printing, such as colored logos or letterheads, that cannot be created conveniently within the data stream.

FOCA. See *Font Object Content Architecture*.

font. A family or assortment of characters of a given size and style; for example, 9-point Bodoni Modern. (A)

font character set. A FOCA resource containing descriptive information, font metrics, and the digital representation of character shapes for a specified graphic character set.

Font Object Content Architecture (FOCA). An architected collection of constructs used to describe fonts and to interchange those font descriptions.

Font Typeface Global Identifier (FGID). See *global resource identifier (GRID)*.

form. A physical piece of paper or other medium on which output data is printed. For cut-sheet printers, a form is one sheet of paper or other medium. For continuous-forms printers, the form is the area of paper (or other medium) defined to the printer as a single physical page, which for fan-fold paper is normally the area between perforations. See also *medium*, *sheet*, and *page*.

format. The arrangement or layout of data on a physical medium or in a presentation space.

formatted data. In FD:OCA, data whose implied syntax and semantics are represented by architected controls that accompany the data.

formatted data object (FDO). An object that contains formatted data. See also *object*.

Formatted Data Object Content Architecture (FD:OCA). An architected collection of constructs used to interchange formatted data.

formatter. A process used to prepare a document for presentation.

Formdef. See *Form Definition*.

form definition. In IBM Print Server Facility, a resource object that defines the characteristics of the form, which include: overlays to be used, text suppression, position of page data on the form, and modifications and number of copies of a page.

forms control buffer (FCB). A line printer control. In the 3800 Printing Subsystem, a buffer for controlling the vertical format of printed output.

forms flash. (1) In the 3800 Printing Subsystem, the function of the printer that allows user-prepared images to be printed with variable page data. An operator must insert the desired image holder when forms overlay printing is desired. (2) The photographic negative of a predefined design to be exposed to the photoconductor by a flash of light. The forms overlay can be merged with variable data during printing. See also *electronic overlay*.

G

GCGID. See *Graphic Character Global Identifier*.

GCSGID. See *Graphic Character Set Global Identifier*.

GID. See *global identifier*.

Global Identifier (GID). Any of the following:

- Code Page Global ID (CPGID)
- Graphic Character Global Identifier (GCGID)
- Font Typeface Global Identifier (FGID)
- Graphic Character Set Global Identifier (GCSGID)
- Coded Graphic Character Set Global Identifier (CGCSGID)
- In MO:DCA, an encoded graphic character string that provides a reference name for a document element.

- Global Resource Identifier (GRID)
- Object Identifier (OID)
- Coded Character Set Identifier (CCSID).

global resource identifier (GRID). An eight-byte identifier that identifies a coded font resource. A GRID contains the following fields in the order shown:

1. GCSGID of a minimum set of graphic characters required for presentation. It can be a character set that is associated with the code page, or with the font character set, or with both.
2. CPGID of the associated code page
3. FGID of the associated font character set
4. Font width in 1440ths of an inch.

GOCA. See *Graphics Object Content Architecture*.

graphic character. A member of a set of symbols that represent data. Graphic characters can be letters, digits, punctuation marks, or other symbols. Synonymous with *glyph*. See also *character*.

Graphic Character Global Identifier (GCGID). An alphanumeric character string used to identify a specific graphic character. A GCGID can be from four-bytes to eight-bytes long.

graphic character identifier. The unique name for a graphic character in a font or in a graphic character set. See also *character identifier*.

Graphic Character Set Global Identifier (GCSGID). A unique graphic character set identifier that can be expressed as either a two-byte binary or a five-digit decimal value.

graphics command set. In the IPDS architecture, a collection of commands used to present GOCA data in a page, page segment, or overlay.

graphics object. An object that contains graphics data. See also *object*.

graphics object area. A rectangular area on a logical page into which a graphics presentation space window is mapped.

Graphics Object Content Architecture (GOCA). An architected collection of constructs used to interchange and present graphics data.

GRID. See *global resource identifier*.

guard bars. The bars at both ends and the center of an EAN, JAN, or UPC symbol, that provide reference points for scanning.

H

height. (1) In Page Printer Formatting Aid, refers to the vertical dimension of a logical page and is controlled by the HEIGHT subcommand. (2) In bar codes, the bar dimension perpendicular to the bar width. Synonymous with *bar height* and *bar length*.

hexadecimal. A number system with a base of sixteen. The decimal digits 0 through 9 and characters A through F are used to represent hexadecimal digits. The hexadecimal digits A through F correspond to the decimal numbers 10 through 15, respectively. An example of a hexadecimal number is X'1B', which is equal to the decimal number 27.

highlighting. The emphasis of displayed or printed information. Examples are increased intensity of selected characters on a display screen and exception highlighting on an IPDS printer.

host. (1) In the IPDS architecture, a computer that drives a printer. (2) In IOCA, the host is the controlling environment.

HRI. See *human-readable interpretation*.

human-readable interpretation (HRI). The printed translation of bar code characters into equivalent Latin alphabetic characters, Arabic numeral decimal digits, and common special characters normally used for printed human communication.

I

image. An electronic representation of a picture produced by means of sensing light, sound, electron radiation, or other emanations coming from the picture or reflected by the picture. An image can also be generated directly by software without reference to an existing picture.

image content. Image data and its associated image data parameters.

Image Object Content Architecture (IOCA). An architected collection of constructs used to interchange and present images.

in. Inches.

IND2OF5. A parameter that specifies a bar code type of Industrial 2-of-5.

Infoprint. A solution of software and hardware products that can supplement or replace the offset presses and copiers in print shops with high-quality, non-impact, black and white or process color printers. Infoprint takes documents from creation to the final product.

Infoprint Manager for AIX or Windows NT/2000. A software component of IBM Infoprint. IBM Infoprint Manager for AIX or Windows NT/2000 handles the scheduling, archiving, retrieving, and assembly of a print job and its related resource files. It also tracks the finishing and packaging of the printed product.

inline. In printing, the direction of successive characters in a line of text. Synonymous with *inline direction*.

inline direction. Synonym for *inline*.

Intelligent Printer Data Stream (IPDS). An architected host-to-printer data stream that contains both data and controls defining how the data is to be presented.

International Organization for Standardization (ISO). An organization of national standards bodies from various countries established to promote development of standards to facilitate international exchange of goods and services, and develop cooperation in intellectual, scientific, technological, and economic activity.

Invoke Data Map. A control record placed in the user's data to begin a new page format.

Invoke Medium Map. A control record placed in the user's data to begin a new copy group.

IOCA. See *Image Object Content Architecture*.

IPDS. See *Intelligent Printer Data Stream*.

ISO. See *International Organization for Standardization*.

italics. A typeface with characters that slant upward to the right. In FOCA, italics is the common name for the defined inclined typeface posture attribute or parameter.

ITL2OF5. A parameter that specifies a bar code type of Interleaved 2-of-5, Automatic Identification Manufacturers Uniform Symbol Specification-I 2/5.

J

JAN. See *Japanese Article Numbering*.

Japanese Article Numbering (JAN). The bar code symbology used to code grocery items in Japan.

jog. Offset stacking of individual sheets or sets of sheets in the output hopper of a page printer or copy mark in a continuous forms printer.

K

kanji. A graphic character set consisting of symbols used in Japanese ideographic alphabets. Each character is represented by 2 bytes.

keyword. A two-part self-defining parameter consisting of a one-byte identifier and a one-byte value.

L

landscape presentation. The position of a printed sheet that has its long edges at the top and bottom and its short edges at the sides. Contrast with *portrait presentation*.

language. A set of symbols, conventions, and rules that is used for conveying information.

leading. A printer's term for the distance between lines of type measured in points. It refers to the lead slug placed between lines of type in traditional typesetting.

library. System storage for generated form definitions and page definitions.

library resource (member). A named collection of records or statements in a library.

library resource name. A name by which an object may be called from a library by IBM Print Server Facility as part of a print job. Includes the two-character prefix for the type of object, such as P1 for page definitions, F1 for form definitions, or O1 for overlays (also known as *resource name*).

line attributes. Those attributes that pertain to straight and curved lines. Examples of line attributes are line type and line width.

line data files. Files formatted for printing on line printers.

line printer. A device that prints a line of characters as a unit. (I) (A) Synonymous with *line-at-a-time printer*. Contrast with *page printer*.

line type. A line attribute that controls the appearance of a line. Examples of line types are dashed, dotted, and solid. Contrast with *line width*.

line width. A line attribute that controls the appearance of a line. Examples of line width are light, medium, and bold. Contract with *line type*.

lines per inch (lpi). (1) On a printer, a measurement of the number of lines per vertical inch of paper. (2) A unit of measure for specifying the baseline increment.

local name. A name for a suppression, an overlay, or a font that is used only within the Page Printer Formatting Aid command stream. Contrast with *user-access name*.

location. A site within a data stream. A location is specified in terms of an offset in the number of structured fields from the beginning of a data stream, or in the number of bytes from another location within the data stream.

logical page. (1) The area on a surface of a form that is formatted for printing. (2) A collection of data that can be printed on one side of a sheet of paper. See also *form* and *page*.

logical page origin. (1) The user-defined point that acts as a reference for all positioning of printed material on the page. (2) The point nearest the hardware origin where printing can occur.

Logical unit (L-unit). A unit of linear measurement expressed with a unit base and units per unit-base value. For example , in Page Printer Formatting Aid, 1 logical unit = 1/240 inch (unit base = 10 inches, units per unit base = 2400).

lpi. Lines per inch.

lowercase. Pertaining to small letters as distinguished from capital letters. Examples of small letters are *a*, *b*, and *g*. Contrast with *uppercase*.

L-unit. A unit of linear measurement expressed with a unit base and units per unit-base value. In other words, the number of units in a linear inch. Synonymous with *logical unit*.

M

MAT2OF5. A parameter that specifies a bar code type of Matrix 2-of-5.

media origin. The first hardware addressable point on the physical page. The point from which the logical page origin is positioned by the medium map.

medium. The physical material (for example, paper) on which data is printed. See also *form*.

medium map. An internal object whose structured fields control the physical sheets of a printout, including the choice of duplex printing, the beginning print position, and the paper source to use. Controlled by a COPYGROUP command in a Page Printer Formatting Aid command stream.

medium overlay. Synonym for *overlay*.

mixed data files. Files consisting of composed and uncomposed portions.

mm. Millimeters.

MOD. A parameter that specifies additional processing information about the bar code symbol to be generated. Refer to *Data Stream and Object Architecture: Bar Code Object Content Architecture Reference* (S544-3766) for more information.

Mixed Object Document Content Architecture (MO:DCA). (1) An architected, device-independent data stream for interchanging documents. (2) Print data that has been composed into pages. Text formatting programs can produce composed text data consisting entirely of structured fields.

MO:DCA. See *Mixed Object Document Content Architecture*.

MO:DCA-P. Mixed Object Document Content Architecture for Presentation.

module. In a bar code symbology, the nominal width of the smallest element of a bar or space. Actual bar code symbology bars and spaces can be a single module wide or some multiple of the module width. The multiple need not be an integer.

MODWIDTH. A parameter that specifies the width of the smallest defined bar code element, using mils (thousandths of an inch).

MSI. A parameter that specifies a bar code type of modified Plessey code.

multiple up. The printing of more than one page on a single side of a sheet of paper.

MVS or OS/390. Multiple Virtual Storage. (Changed to OS/390).

N

name. A table heading for architecture syntax. The entries under this heading are short names that give a general indication of the contents of the construct.

noncompatibility mode. The use of table reference character (TRC) numbers not compatible with a line printer.

normal duplex printing. Duplex printing for sheets that are to be bound on the long edge of the paper, regardless of whether the printing is portrait or landscape. Contrast with *tumble duplex printing*.

N_UP. The printing of more than one logical page on a single side of a medium.

O

object. A collection of data referred to by a single name. Form definitions and page definitions stored in a library are resources.

offset. A table heading for architecture syntax. The entries under this heading indicate the numeric displacement into a construct. The offset is measured in bytes and starts with byte zero. Individual bits can be expressed as displacements within bytes.

order. In GOCA, a graphics construct that the controlling environment builds to instruct a drawing processor about what to draw and how to draw it.

orientation. The angular distance a presentation space or object area is rotated in a specified coordinate system, expressed in degrees and minutes. For example, the orientation of printing on a physical medium, relative to the X_m axis of the X_m,Y_m coordinate system.

origin. A picture element (pel)

outline font. A shape technology in which the graphic character shapes are represented in digital form by a series of mathematical expressions that define the outer edges of the strokes. The resultant graphic character shapes can be either solid or hollow.

overlay. A collection of predefined data such as lines, shading, text, boxes, bar codes, or logos, that can be merged with variable data on a page during printing. See *electronic overlay*.

Overlay Generation Language (OGL). A programming language used to produce electronic overlays.

P

page. (1) A collection of data that can be printed on one side of a sheet of paper or a form. (2) The boundary for determining the limits of printing. See also *logical page* and *physical page*.

page definition. A resource containing a set of Page Printer Formatting Aid formatting controls for printing pages of data. Includes controls for number of lines per printed sheet, font selection, print direction, and mapping of individual fields in the data to positions on the printed sheets.

page ejection. The point at which the printer finishes printing on one sheet and moves to the beginning of the next sheet.

page format. A subset of a page definition, containing all the same controls for formatting printed output as a page definition. Includes controls for number of lines per printed sheet, font selection, print direction, and mapping of individual fields in the data to positions on the printed sheets.

page mode. The mode of operation in which an AFP printer can accept a page of data from a host processor to be printed on an all-points-addressable output medium. Printed data can consist of pages composed of text, images, overlays, and page segments. Contrast with *compatibility mode*.

page printer. A device that prints a page at a time. Contrast with *line printer*.

Page Printer Formatting Aid for AIX (Page Printer Formatting Aid). An IBM licensed program that allows you to create and store form definitions and page definitions, which are resource objects for print-job management. By writing a command stream specifying form definitions, page definitions, or both, for executing Page Printer Formatting Aid, you can store the objects specified in the library. These objects can then be used to format printed output.

page segment. (1) An object that can contain text and images and be included at any addressable point on a page or electronic overlay. It assumes the environment of an object it is included in. (2) A library resource that contains the definition of a page segment. Contrast with *electronic overlay*.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (I) (A) (2) In Page Printer Formatting Aid, the values specified for a subcommand.

partition. (1) Dividing the medium presentation space into a specified number of equal-sized areas in a manner determined by the current physical media. (2) In FD:OCA, a conceptual subdivision of a string of data fields. A partition can be further divided into subpartitions.

pel. Picture element. The smallest printable or displayable unit on a physical medium. In computer graphics, the smallest element of a physical medium that can be independently assigned color and intensity. Synonymous with *picture element* and *pixel*.

PELS. In Page Printer Formatting Aid, a unit of measure under the SETUNITS command. See also *logical unit*.

physical page. A single surface (front or back) of a sheet. See also *form* and *page*.

picture element. (1) In computer graphics, the smallest element of a display space that can be independently assigned color and intensity. (T) (2) The smallest area that can be individually toned by the printer.

pixel. The smallest printable or displayable unit on a physical medium. Synonymous with *pel* and *picture element*.

PMF. Print Management Facility

point. In printing, a unit of about 1/72 of an inch used in measuring typographical material, for example: 10-point Helvetica. There are 12 points to a pica.

portrait presentation. The position of a printed sheet that has its short edges at the top and bottom and its long edges at the sides. Contrast with *landscape presentation*.

position. The location specified for a line or field on the output page.

POSTNET. A parameter that specifies a bar code type of POSTal Numberic Encoding Technique (United States Postal Service), and defines specific values for the BSD module width, element height, height multiplier, and wide-to-narrow ratio fields.

PPFA. Page Printer Formatting Aid.

preprinted form. A sheet of paper containing a preprinted design of constant data. Variable data can be merged with the constant data on such a form. See also *electronic overlay*, *forms flash*.

print line. A single line of text. In the formatting of line data, it refers to the output generated by one data record. Governed by the PRINTLINE command.

Print Management Facility (PMF). A program that can create fonts, segments, page definitions, and form definitions.

Print Server Facility (PSF). A program that produces printer commands from the data sent to it.

printer–attached device. Either a preprocessor or postprocessor attached to the printer.

PSF. Print Server Facility.

R

range. A table heading for architecture syntax. The entries under this heading give numeric ranges applicable to a construct. The ranges can be expressed in binary, decimal, or hexadecimal. The range can consist of a single value.

raster. (1) In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space. (T) (2) In AFP printers, an on-or-off pattern of electrostatic images produced by the laser print head.

RASTER / NORASTER subcommand. A subcommand that specifies whether an overlay is to be kept in the printer (3800 only) as raster data.

RATIO. A parameter that specifies the ratio of the wide-element dimension to the narrow-element dimension whenever two different size elements exist.

ratio. The relationship in quantity, amount, or size between two or more things.

record. (1) In programming languages, an aggregate that consists of data objects, possibly with different attributes, that usually have identifiers attached to them. In some programming languages, records are called structures. (I) (2) A set of data treated as a unit. (T) (3) A set of one or more related data items grouped for processing.

RM4SCC. A parameter that specifies a 4-state customer code defined by the Royal Mail Postal Service of England for bar coding postal code information. See *Royal Mail 4 State Customer Code*.

resource. A collection of printing instructions, and sometimes data to be printed, that consists entirely of structured fields. A resource object is stored as a member of a library and can be called for by IBM Print Server Facility when needed. The different resource objects are: page segments, overlays, form definitions, and page definitions.

RNORMAL. Rotated normal. A Page Printer Formatting Aid parameter that specifies the type of duplex printing. It means the tops of both sides of a duplex-printed sheet are toward the same physical edge of the sheet, for side binding of the document. Used with landscape-presentation pages.

rotation. The orientation of the characters of a font with respect to the baseline.

Royal Mail 4 State Customer Code (RM4SCC). A two-dimensional bar code symbology developed by the United Kingdom's Royal Mail postal service for use in automated mail-sorting processes.

RTUMBLE. Rotated tumble. A Page Printer Formatting Aid parameter that specifies a type of duplex printing. It means the top of one side of a duplex-printed sheet and the bottom of the other are toward one physical edge of the sheet, for top binding of the document. Used with landscape-presentation pages.

rule. A solid line of any line width.

S

SBCS. Single-byte character set.

scanner. In bar codes, an electronic device that converts optical information into electrical signals. Sometimes called a *reader* or *decoder*.

segment. (1) A collection of composed text and images, prepared before formatting and included in a document when it is printed. See *page segment*. (2) The resource that contains the structured-field definition of a page segment.

sheet. A single piece of paper. For cut-sheet printers, a synonym for *form*.

shift-in and shift-out characters (SOSI). Characters used to delimit literals in Page Printer Formatting Aid command streams: X'0E' and X'0F'.

simplex printing. A method used to print data on one side of a sheet; the other side is left blank. Contrast with *duplex printing*.

single-byte character set. A character set whose codes require a single byte of data. The character set used for English is an example.

skip-to-channel control. A line printer control appearing in line data. Allows space to be left between print lines. Compatible with page printers when the data is formatted by page definitions.

space. In bar codes, the lighter element of a printed bar code symbol, usually formed by the background between bars.

space width. In bar codes, the thickness of a bar code symbol space measured from the edge closest to the symbol start character to the trailing edge of the same space.

SSASTERISK. A parameter that specifies whether an asterisk is to be generated as the HRI for **CODE39** bar code start and stop characters.

start-stop character or pattern. In bar codes, a special bar code character that provides the scanner with start and stop reading instructions as well as a scanning direction indicator. The start character is normally at the left end and the stop character at the right end of a horizontally-oriented bar code symbol.

structured field. A self-identifying string of bytes and its data or parameters.

subcommand. (1) In Page Printer Formatting Aid, the next level of control below commands. (2) A request for an operation that is within the scope of work requested by a previously issued command.

subgroup. A subset of a form definition that is used to reprint the same page of data more than once. Subgroups provide for variations in the same page of data within one print job. Modifications that distinguish one subgroup from another are number of copies, type of duplex printing, inclusion of overlays, inclusion of suppressions, and (only for the 3800 printer) forms flash. A set of modifications within a copy group that applies to a certain number of copies of a form. A copy group can contain more than one subgroup.

subpage. A part of a logical page on which line data may be placed. In the page definition, multiple subpages can be placed on a physical page based on changes in the print data.

suppression. The electronic equivalent of a spot carbon, preventing selected data from being printed on certain copies.

symbology. A bar code language. Bar code symbologies are defined and controlled by various industry groups and standards organizations. Bar code symbologies are described in public domain bar code specification documents. Synonymous with *bar code symbology*. See also *Canadian Grocery Product Code (CGPC)*, *European Article Numbering (EAN)*, *Japanese Article Numbering (JAN)*, and *Universal Product Code (UPC)*.

syntax. The rules governing the structure of a construct.

T

table reference character (TRC). Usually, the second byte on a line in the user's data. This byte contains a value (0 - 126) that is used to select a font to be used to print that line.

tate. The Japanese word for top-to-bottom, as applied to the formatting of writing and printing. The traditional arrangement of Japanese kanji characters on the page. Pronounced ***ta***-tay.

text. A graphic representation of information on an output medium. Text can consist of alphanumeric characters and symbols arranged in paragraphs, tables, columns, and other shapes.

TRC. Table reference character.

truncation. Planned or unplanned end of a presentation space or data presentation.

tumble duplex printing. Duplex printing for sheets that are to be bound on the top, as is often done for legal documents. The top of one side of each sheet is at the same edge as the bottom of the other side. Contrast with *normal duplex printing*.

triplet. A three-part self-defining variable-length parameter consisting of a length byte, an identifier byte, and one or more parameter-value bytes.

type. A table heading for architecture syntax. The entries under this heading indicate the types of data present in a construct. Examples include: BITS, CHARCODE, SBIN, UBIN, UNDF.

TYPE. A parameter that specifies the kind of bar code symbol to be generated. For example, CODE39, MSI, UPCA, UPCE, and so on.

type font. See *font*.

type weight. A parameter indicating the degree of boldness of a typeface. A character's stroke thickness determines its type weight. Examples are light, medium and bold.

type width. A parameter indicating a relative change from the font's normal width-to-height ratio. Examples are normal, condensed and expanded.

U

unformatted print data. Data that is not formatted for printing. A page definition can contain controls that map unformatted print data to its output format.

Uniform Symbol Specification (USS). A series of bar code symbology specifications published by AIM; currently included are USS-Interleaved 2 of 5, USS-39, USS-93, USS-Codabar, and USS-128.

Universal Character Set (USC). A printer feature that permits the use of a variety of character arrays. Synonymous with *font*.

Universal Product Code (UPC). A standard bar code symbology, commonly used to mark the price of items in stores, that can be read and interpreted by a computer.

unprintable area. The area of a sheet of paper on which no printing can be done because of printer and hardware limitations.

UPC. See *Universal Product Code*.

UPCA. A parameter that specifies a bar code type of Universal Product Code (United States) and the Canadian Grocery Product Code, Version A.

UPCE. A parameter that specifies a bar code type of Universal Product Code (United States) and the Canadian Grocery Product Code, Version E.

UPC2SUPP. A parameter that specifies a bar code type of Universal Product Code (United States) two-digit Supplemental (periodicals).

UPC5SUPP. A parameter that specifies a bar code type of Universal Product Code (United States) five-digit Supplemental (paperbacks).

uppercase. Pertaining to capital letters. Examples of capital letters are *A*, *B*, and *C*. Contrast with *lowercase*.

user-access name. The library resource name of a font or an overlay, less its two-character prefix. Contrast with *local name*.

USS. See *Uniform Symbol Specification*.

W

width. In Page Printer Formatting Aid, refers to the horizontal dimension of a logical page, is specified in the page definition, and is controlled by the WIDTH subcommand.

X

x-coordinate. The horizontal or inline position that defines a page origin or the starting point of a line or field.

Y

y-coordinate. The vertical or baseline position that defines a page origin or the starting point of a line or field.

Bibliography

Bibliography This bibliography lists the titles of publications containing additional information about Printer Services Facility (PSF), Advanced Function Presentation (FOP), the MVS and other operating systems and related products.

The titles and order numbers may change from time to time. To verify the current title or order number, consult your IBM marketing representative.

You can obtain many of the publications listed in this bibliography from the Printing Systems Digital Library:

`http://www.ibm.com/printers/r5psc.nsf/web/manuals`

Advanced Function Presentation (AFP)

The following publications contain information about IBM's Advanced Function Presentation (AFP) concepts and procedures, and the AFP architecture:

Publication	Order Number
Advanced Function Presentation: Printer Information	G544-3290
Advanced Function Presentation: Printer Summary	G544–3135
Advanced Function Presentation: Programming Guide and Line Data Reference	S544–3884
Advanced Function Printing: Host Font Data Stream Reference	S544–3289
AFP Toolbax for Multiple Operating Systems User's Guide	G544–5292
AFP Workbench for Windows: Using the Viewer Application	G544–3813
Guide to Advanced Function Presentation	G544–3876
IBM Page Printer Formatting Aid: User's Guide	S544–5284
IBM Overlay Generation Language/370: User's Guide	S544–3702
Mixed Object Document Content Architecture Reference	SC31-6802
Printing and Publishing Cluster Collection CD-ROM	SK2T-2921
Font Object Content Architecture Reference	S544–3285
Graphic Object Content Architecture Reference	SC31–6804
Image Object Content Architecture Reference	SC31–6805
Intelligent Printer Data Stream Reference	S544–3417
Presentation Text Object Content Architecture Reference	SC31–6803

Print Service Facility (PSF) for AIX

The following publications contain information about AIX general concepts and procedures, and about IBM PSF for AIX that uses the form definitions and page definitions created with PPFA:

Publication	Order Number
AIX and Related Products Documentation	SC23–2456
Facts About PSF for AIX	G544–5305
IBM Page Printer Formatting aid/6000: User's Guide	S544–3918
IBM Print Services Facility for AIX: AIX for Print Services Facility Users	G544–3766
IBM Print Services Facility for AIX: AIX for Users of Print Services Facility	G544–3877
IBM Print Services Facility for AIX: Advanced Function Presentation Conversion and Indexing Facility	G544–3930
IBM Print Services Facility for AIX: Guide for Printer and COM Operators	S544–5286
IBM Print Services Facility for AIX: Licensed Program specifications	G544–3815
IBM Print Services Facility for AIX: Print Administration	S544–3817
IBM Print Services Facility for AIX: Print Services Facility for AIX Users	G544–3814
IBM Print Services Facility for AIX: Print Submission	S544–3878
Using IBM Infoprint for Production Printing	S544–5473

BCOCA

The following publication contains information about bar code concepts related to the PPFA BARCODE subcommand:

Publication	Order Number
Data Stream and Object Architectures: Bar Code Object Content Architecture Reference	S544–3766

OS/400

The following publications contain information about OS/400 that uses the form definition and page definitions created by PPFA, and InfoPrint Server/400 information:

Publication	Order Number
OS/400 Guide to AFP and PSF	S544–5319
OS/400 Printer Device Programming	SC41–3713
OS/400 Data Description Specifications	SC41–962
OS/400 Command Language Reference	SC41–3722
IBM OS/400 printing IV	GG24–4389
OS/400 Advanced Utility User's Guide	S544–5351
IBM Advanced Function Presentation: Toolbox for OS/400 User's Guide	S544–5368
InfoPrint Server for OS/400: User's Guide	

VSE, MVS and VM

The following publications contain information about the general concepts and procedures for VSE, MVS and VM environments:

Publication	Order Number
Print Services Facility/VSE: Application Programming Guide	S544–3666
Print Services Facility/VSE: System Programming Guide	S544–3665
Print Services Facility/MVS: Application Programming Guide	S544–3673
Print Services Facility/MVS: System Programming Guide	S544–3672
Print Services Facility/VM: Application Programming Guide	S544–3677
Print Services Facility/VM: System Programming Guide	S544–3680

Infoprint Server for OS/390

The following publications contain information about the infoprint server for OS/390:

Publication	Order Number
OS/390 Infoprint Server Customization	G544-5694
OS/390 Infoprint Server Introduction	G544–5696
OS/390 Infoprint Server Messages and Diagnosis	G544–5690
OS/390 Infoprint Server Migration	G544–5697
OS/390 Infoprint Server Operation and Administration	S544–5693
OS/390 Infoprint Server User's Guide	S544–5692

Print Services Facility (PSF) for OS/390

The following publications contain information about the print services facility (PSF) for OS/390:

Publication	Order Number
IBM IP Printway Guide	S544–5379
IBM NetSpool Guide	G544–5301
PSF for AIX: Upload Configuration Guide for SNA	S544–5422
PSF for AIX: Upload Configuration Guide for TCP/IP	S544–5423
PSF for OS/390 Collection Kit CD-ROM	SK2T-9267
PSF for OS/390: Customization	S544–5622
PSF for OS/390: Diagnosis	G544–5623
PSF for OS/390: Download for OS/390	S544–5624
PSF for OS/390: Introduction	S544–5625
PSF for OS/390: Licensed Program Specifications	G544–5626
PSF for OS/390: Messages and Codes	G544–5627
PSF for OS/390: User's Guide	S544–5630
AFP Conversion and Indexing Facility: User's Guide	S544–5285
PSF: Security Guide	S544–3291
Program Directory for Print Services Facility for OS/390	None
Program Directory for Download for OS/390	None
Program Directory for IP PrintWay Feature of PSF for OS/390	None

Program Directory for IP NetSpool Feature of PSF for OS/390	None

Fonts

The following publications contain information about the fonts used by IBM:

Publication	Order Number
ABOUT TYPE: IBM's Technical Reference for 240-pel Digitized Type	S544–3516
IBM AFP Fonts: Font Samples	S544–3792
IBM AFP Fonts: Font Summary	G544–3810
IBM AFP Fonts: Font Summary for AFP Font Collection	S544–5633
IBM AFP Fonts: IBM's Typographic Primer for Digitized Type	G544–3183
IBM AFP Fonts: Introduction to Typography	G544–3122
IBM AFP Fonts: Technical Reference for Code Pages	S544–3802
IBM AFP Fonts: Technical Reference for IBM Chinese, Japanese, and Korean Fonts	S544–5330
IBM AFP Fonts: Technical Reference for IBM Expanded Core Fonts	S544–5625
IBM AFP Fonts: Type Transformer User's Guide	G544–3796

Text Processing

The following publications contain information about text processing:

Publication	Order Number
DCF/DLF General Information	GH20–9158
Document Composition Facility: Bar Code User's Guide	S544–3115
Document Composition Facility: SCRIPT/VS Text Programmer's Guide	SH35–0069
Publishing Systems BookMaster General Information	GC34–5006
Publishing Systems BookMaster User's Guide	SC34–5009
Using DisplayWrite/370	SH12–5172
IBM AFP Fonts: Technical Reference for Code Pages	S544–3802
IBM AFP Fonts: Technical Reference for IBM Chinese, Japanese, and Korean Fonts	S544–5330
IBM AFP Fonts: Technical Reference for IBM Expanded Core Fonts	S544–5625
IBM AFP Fonts: Type Transformer User's Guide	G544–3796

Infoprint Manager

The following publications contain information about IBM's Infoprint Manager:

Publication	Order Number
IBM Infoprint Manager: Reference	S544–5475
IBM Infoprint Manager for AIX: Administrator's Guide	S544–5595
IBM Infoprint Manager for AIX: User's and Operator's Guide	S544–5596
IBM Infoprint Manager for Windows NT and Windows 2000: Planning Guide	G544–5716
IBM Infoprint Manager for Windows NT and Windows 2000: Getting Started	G544–5717
IBM Infoprint Manager for Windows NT and Windows 2000: Configuration Guide	Web-Based
IBM Infoprint Manager for Windows NT and Windows 2000: PSF Direct Network Configuration Guide	Web-Based

Printers

The following publications contain information about IBM's printers:

Publication	Order Number
Reference Manual for the IBM 3800 Printing Subsystem Models 3 and 6	GA32–0050
IBM PagePrinter 3812 Introduction and Planning Guide	G544–3265
IBM 3816 Page Printer Operating Instructions	GA34–2075
IBM 3825 Page Printer Product Description	G544–3482
IBM 3827 Page Printer Product Description	G544–3194
IBM 3828 Advanced Function MICR Printer Product Description	G544–3361
IBM 3900 Advanced Function Printer Product Description	GA32–0135
IBM 3912 and 3916 Page Printer Getting Started	S544–3898
IBM LaserPrinter 4028 Introduction and Planning Guide	S544–4258

User's Guide for the IBM LaserPrinter 4029 Series	ZA40–0542
IBM 4224 Printer Models 1xx and 2xx Product and Programming Description Manual	GC31–2551
IBM 4230 Printer Product and Programming Description Models 102 and 202	GC40–1701
IBM 4234 Printer Models 007, 008, 011 and 012 Product and Programming Description	GC31–3879
InfoPrint 60, 3130, 3160, and 3935 Advanced Function Printer Attachment Configuration Handbook	S544–3977
InfoPrint 60 Finisher Application Design Guide	S544–5643
InfoPrint 62 Introduction and Planning Guide	G544–5384
InfoPrint 3000 Introduction and Planning Guide	G544–5563
InfoPrint 4000 and 3900 Advanced Function Printers Introduction and Planning Guide	G544–5427
InfoPrint Color 100 Introduction and Planning Guide	G544–5612
InfoPrint Hi-Lite Color Introduction and Planning Guide	G544–5420
IOCP and ESCON Reference	GC38–0401

TCP/IP

The following publications contain information about TCP/IP:

Publication	Order Number
Internetworking with TCP/IP, Principles, Protocols and Architecture	SC31–6144
TCP/IP Tutorial and Technical Reference	GG24–3376
InfoPrint Hi-Lite Color Introduction and Planning Guide	G544–5420

TCP/IP for MVS

The following publications contain information about TCP/IP for MVS:

Publication	Order Number
TCP/IP for MVS: Customization and Administration Guide	SC31–7134
TCP/IP for MVS: Application Programming Interface Reference	SC31–7187
TCP/IP for MVS: Programmer's Reference	SC31–7135

VTAM and NCP

The following publications contain information about VTAM and NCP:

Publication	Order Number
ACF/INCP/SSP Version 3 Resource Reference	SC30–3254
Advanced Communications Function for VTAM, Version 3, Programming	SC23–0115
Advanced Communications Function for VTAM, Version 2, Programming	SC27–0611
Advanced Communications Function for VTAM, Version 3, Customization	SC23–0112
NCP Resource Definition Guide	SC30–3449
NCP: Resource Definition Reference	SC30–3254
NCP: System Support	SC30–3447
Network Program Products: Bibliography and Master Index	GC30–3353
Network Program Products: General Information	GC23–0108
VTAM Resource Definition Reference	SC31–6552
VTAM Version 3 Diagnosis Reference	LY30–5582
VTAM Version 3 Messages and Codes	SC31–6433
VTAM Version 3 Operation	SC23–0113
VTAM Version 4 Diagnosis Guide	LX75–0204
VTAM Version 4 Diagnosis Quick Reference	LX75–0203
VTAM Version 4 Messages and Codes	SC31–6418

System Network Architecture (SNA)

The following publications contain information about System Network Architecture:

Publication	Order Number
SNA Customization	LY43–0110
SNA Formats (SNA Reference Summary)	GA27–3136
SNA Resource Definition Reference	SC31–8565
SNA Technical Overview	GC30–3073

SNA transaction Programmer's Reference Manual for LU Type 6.2	GC30–3084
Systems Network Architecture Formats	GA27–3136

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Readers' Comments — We'd Like to Hear from You

IBM Page Printer Formatting Aid: User's Guide

Publication No. S544-5284-05

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
S544-5284-05

IBM

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department H7FE, Building 003G
Boulder, CO 80301-9817

Fold and Tape **Please do not staple** Fold and Tape

S544-5284-05

Cut or Fold
Along Line

Program Number: 5688-190
5765-E42
5798-AF2
5639-I27

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

S544-5284-05