



## Article: Links and Forms using Pdfmarks

### Directory

- [Table of Contents](#)
- [Links](#)
- [Forms](#)
- Doc Info
- Document-Level and Page-Level Actions
- Annotations
- Using Structure in T<sub>E</sub>X
- Acrobat JavaScript
- Appendix: [Acrobat 5.0: What's New](#)

## 1. Hypertext Links

- Introduction

### 1.1. AcroT<sub>E</sub>X

- PostScript Users • T<sub>E</sub>X Users • Configuring the Dvi to PostScript Driver

### 1.2. Some T<sub>E</sub>X Macros for Links

### 1.3. Actions

- Two Methods for Creating Action Links

### 1.4. Link Appearance

- Default Appearance • Invisible Rectangle • Changing the Color of Border • Thicker Borders • Dashed Borders • Highlighting the Button • Coloring the Text

### 1.5. Within Document Jumps

- Jump to a Page Number • Jump to Relative Page • Defining a Named Destination • Jump to a Named Destination

### 1.6. Jumping to a Remote File

- Go to a File • Go to a Page in a File • Go to a Named Destination • Go to an URL

### **1.7. The Launch Action**

- Go to an URL

### **1.8. Controlling the View**

- /Fit • /FitB • /FitH • /FitBH • /FitV • /FitBV • /XYZ
- /FitR

### **1.9. Named Actions**

### **1.10. Multiple Actions**

### **1.11. Linking Icons and Pictures**

- Using Fonts and Rules • Using Tiff images • Using EPS Images

### **1.12. Jumping to Local Files**

### **1.13. Hypertext using Y&Y**

- Jump to a Page • Open a File • Define a Y&Y Mark
- Jump to a Mark • Jump to an URL

## **2. Forms**

- Introduction

### **2.1. Quick Survey of Form Annotations**

### **2.2. The AcroForm Dictionary**

### **2.3. T<sub>E</sub>X Macros for Forms**

## **2.4. Field Properties**

- The Field flag /Ff • The Field Type /FT • Highlighting /H • The Flags Key /F • Border Style /BS

## **2.5. Button Field: Push Buttons**

- Additional Keys: /DA and /MK • Standard Push Button
- A Rotated Push Button • XObjects: Push Button using /AP

## **2.6. Button Field: The Checkbox**

- The /DA and /MK Keys • A Standard Checkbox • Re-defining /Yes • Checkbox with Shadow • A Checkbox with two Appearances • Tic-Tac-Doe

## **2.7. Button Field: Radio Buttons**

- A Standard Radio Button Field • Default Value, Initial Value, Clear to Default • Radio Button Field using Macros
- A Simple Method

## **2.8. Choice Field: The List Box**

- Standard List Box

## **2.9. Choice Field: The Pop-Up Box**

- Standard Pop-Up Box • List Box and /DV

## **2.10. Choice Field: The Combo Box**

- Standard Combo Box

## **2.11. Text Fields**

- Options: /Q and /MaxLen
- Standard Text Field
- Data Sharing/Different Appearances
- Form Field Hierarchies
- A Location Bar
- A Multiline Text Field

## **2.12. Enhancing your Text**

- Changing the Rendering Mode
- Changing the Word and Character Spacing
- Changing the Horizontal Scale
- Line Breaking with a Horizontal Scale Change

## **2.13. Cos Objects**

- Defining Cos Objects
- Predefined Cos Objects
- Indirect Naming within pdfmarks
- Placing Information in Cos Objects
- Creating a Stream: An Example

## **2.14. Defining/Using XObjects**

- Commentary on an Encapsulated Graphic
- Using DVIPSONE
- Using DVIPS
- XObjects Listings

## **3. Acrobat 5.0: What's New**

### **3.1. Bookmarks**

- Bookmarks with Color and Style

### **3.2. Viewer Preferences**

- DisplayDocTitle

**3.3. JavaScript and FDF**

- The /After and /Before Keys
- The /Doc Key

**3.4. Document-Level Actions**

**3.5. Forms**

- Arbitrary Font Definitions



# 1. Forms

## • Introduction

If you thought the article on [Links](#) was complicated, then you haven't seen anything yet! The voluminous topic of [Form Fields](#) is greatly more complex than [Links](#). Let us separate the *creation* of form fields from the *application* of form fields.

**Creation.** Initially, we discuss how to create all the common types of form fields using `pdfmarks`: text fields, buttons, check boxes, radio buttons, pop-up lists, and combo boxes. Part of what goes into the creation of a form field is defining the *appearance* of the field. Certainly, a lot of discussion will go into these mechanical, technical details.

In terms of sources of information, **Portable Document Format Reference Manual, Version 1.2**, [\[pdfs\]](#), is the ultimate reference for forms. The **pdfmark Reference Manual, Technical Note #5150**, [\[pdfm\]](#), contains many examples for creating form fields using `pdfmarks` and was the starting point of my study. Additionally, [Thomas Merz's The Pdfmark Primer](#), freely available over the WEB, is another

reference. The [Primer](#) has examples of inserting [JavaScript](#) using [pdf-marks](#).

**Application.** The value of form fields is in its richness of application which is well beyond the simple capabilities of hypertext links.

Part of my goals of this article is to present application to form fields. These applications can be broken down into four categorizes:

- Using buttons, check boxes, and radio buttons to perform *actions*. This was done extensively in the games [Algeboard](#) and [Giants of Calculus](#).
- Using [JavaScript](#) to process and manipulate form fields. An example of this is the [JavaScript](#) version of [Algeboard/JS](#). (Forms 3.5 is required.)
- Processing form field data using CGI script. A very useful tool for writing CGI script to process [Adobe](#) for data is the [FDF Toolkit](#) by [Adobe](#). An example of this CGI processing is the demo [Quiz Grading Program](#).
- Using a *combination* of the above methods.

This is a rather ambitious undertaking for a target audience so small:

$$\text{AcroTeX\_target\_population} \approx (\text{world\_population}) \times 10^{-6}$$

## 1.1. Quick Survey of Form Annotations

### Button Fields

#### ▲ Push Button

Marital Status?

Married!

#### ▲ Check Box

Marital Status?

Single

Married

#### ▲ Radio Button

### Choice Fields

#### ▲ Pop-up List

#### ▲ Combo Box


#### ▲ List Box

### Text Fields

#### ▲ Text Box

#### ▲ Multiline Text

The difference between the Pop-up Box and the Combo Box is that you can *edit* a Combo Box. (Enter your own major.)

**Note:** Enter your name in the text box above. Your name will be transferred to another portion of the document. **Watch for it!** 

## 1.2. The AcroForm Dictionary

The properties of the form are encoded in the [AcroForm](#) dictionary, which must be referenced in the Catalog dictionary using the key **AcroForm**.

The following listing was taken from [\[pdfm, pp. 42–46\]](#), and modified for use by  $\text{\TeX}$ .

```
%
%   Fields   : The array from where all fields in the form can be found
%   DR       : Default Resources
%   DA       : Default Appearance
%   CO       : Calculation Order
%   NeedAppearances : boolean, set to true
\special{ps: [ /_objdef {afields} /type /array /OBJ pdfmark
              [ /_objdef {aform} /type /dict /OBJ pdfmark]
\special{ps: [ {aform} << /Fields {afields}
              /DR << /Font << >> >>
              /DA (/Helv 12 Tf 0 g)
              /CO {corder}
              /NeedAppearances true >>
/PUT pdfmark}
\special{ps: [/_objdef {corder} /type /array /OBJ pdfmark}
%
% Put the /AcroForm entry in Catalog dictionary
\special{ps: [ {Catalog} << /AcroForm {aform} >> /PUT pdfmark}
```

This finishes the creation of the `/AcroForm` dictionary. I have stripped it down to its absolute minimum.

*AcroForm Notes:* You'll note that within default resources, `/DR`, the font dictionary, `/Font`, has been left `null`. This does not seem to bother [Acrobat](#) since the Base 13 fonts are guaranteed resources—see the table listing of the Base 13 fonts below.

- The value of the key `/Fields` is an array of indirect references to all the form fields in the document. By examining the `pdf` source file, I notice that this array is *empty* initially, and gets 'filled' only when the file is *optimized* with [Exchange](#).

- On my **Win95** machine, the [Acrobat Reader](#) functions perfectly on form fields created by using this `/AcroForm` dictionary. It may be important to note that when a file, such as this one, is *optimized* using [Acrobat Exchange](#), much of the missing information is placed in the `pdf` file by [Exchange](#).

- If you initially set `/NeedAppearances` to `false`, which is the default value, the form fields do not appear. Putting `/NeedAppearances` to `true`, requests the [Reader \(Acrobat/Exchange\)](#) to generate the appearances (the `/AP` key and its value) based on the provided

key-value pairs. Once the document is saved using [Acrobat](#), the generated appearance code is saved as well, and the `/NeedAppearances` key-value pair is removed from the `Acroform` dictionary. ■

The following are the Base 13 fonts plus the `Symbol` font. These are predefined by [Acrobat](#) and guaranteed to exist. When defining form fields, these fonts are referred to by their four character names.

The Base 13 Fonts Plus Symbol			
Times-Roman	TiRo	Times-Bold	TiBo
Times-Italic	TiIt	Times-BoldItalic	TiBI
Helvetica	Helv	Helvetica-Bold	HeBo
Helvetica-Oblique	HeOb	Helvetica-BoldOblique	HeBO
Courier	Cour	Courier-Bold	CoBo
Courier-Oblique	CoOb	Courier-BoldOblique	CoBO
ZapfDingbats	ZaDb	Symbol	Symb

TABLE 1

### 1.3. $\text{T}_{\text{E}}\text{X}$ Macros for Forms

Throughout this article, the macros developed in [Links](#) will be used. So let's begin by inputting these macros:

```
%
```

```
% Input links macro
```

```
\input links_m
```

We also input the `Acroform` dictionary, as discussed in [SECTION 1.2](#).

```
%
```

```
% Input Acroform Dictionary
```

```
\input acroform
```

```
%
```

```
% Input Encapsulated Postscript code for Form XObjects.
```

```
\input xobjects.eps
```

Let's make some general comments about the contents of these three files.

*Links\_m Notes:* This is the file extensively discussed in the article [Links](#). A macro *not* discussed in [Links](#) is the `/Bbox` macro.

- When creating forms from `pdfmarks`, it is necessary to have a convenient way of defining “invisible rectangles.”

```
%
```

```
\def\Bbox[#1,#2]{\vbox to #2{\hbox to #1{\hfill}\vfill}}
```

```
%
```

This macro will be used to define the dimensions of the bounding rectangle of a form annotation. ■

*Acroform Notes:* We begin by inputting the `acroform` dictionary, as discussed in [SECTION 1.2](#).

■ The **Field flags** key, /Ff, is used to specify a variety of properties of a form field. These properties can be combined by the bitwise operation ‘or’ to create a field with several of these attributes. Some of these properties, however, don’t make sense for certain fields (e.g., the ‘sort’ bit is ignored in a push-button annotation).

When using [Acrobat Exchange](#), these are among the many properties to choose from in the process of defining the a form field; we shall presently see how these properties are defined using pdfmarks.

%

**% Values of the Field flag key, /Ff**

**% Usage:’/Ff \FfPushButton \FfNoExport or’**

```

\def\FfReadOnly{1 }           % bit 1: Read only field
\def\FfRequired{1 1 bitshift } % bit 2: Required field (Submit)
\def\FfNoExport{1 2 bitshift } % bit 3: Used with Submit Action
\def\FfMultiLine{1 12 bitshift } % bit 13: For Multiline text field
\def\FfPassword{1 13 bitshift } % bit 14: Password field
\def\FfNotNullState{1 14 bitshift }% bit 15: Used with Radio Buttons
\def\FfRadio{1 15 bitshift }    % bit 16: Radio Button Flag
\def\FfPushButton{1 16 bitshift } % bit 17: Push Button Flag
\def\FfPopUpList{1 17 bitshift } % bit 18: Pop up List Flag
\def\FfEdit{1 18 bitshift }     % bit 19: Edit/NoEdit
\def\FfSort{1 19 bitshift }     % bit 20: Sort List
\def\FfComboBox{\FfPopUpList \FfEdit or } % Combo = Edit PopUpList

```

■ These same macros can be implemented as `PostScript` procedures as well. Simply insert the `PostScript` definitions in the user prologue; for example, we can define

```
\special{!/FfNoExport {1 2 bitshift} def} (1)
```

`DVIPSONE`, or for `DVIPS`,

```
\special{!userdict begin /FfNoExport {1 2 bitshift} def end} ■
```

*XObjects Notes*: Input the `XObjects` used for button faces at the beginning of the document. See the listing in [SECTION 1.14](#). ■

## 1.4. Field Properties

Currently there are three types of form annotations:

- Button Fields: push-buttons, check boxes, and radio-buttons.
- Choice Fields: list boxes, pop-up boxes, and combo boxes.
- Text Fields

Examples of each these can be found in [SECTION 1.1](#) as well as in the numerous examples that follow.

Form fields are of `/Subtype /Widget`, just as hypertext links were of `/Subtype /Link`.

The following is a general outline of the structure of a `Widget`:

```
%
```

```

% Widget template
[ /_objdef {mywidget}          % optional indirect reference to widget
  /Subtype /Widget             % Form = Widget
  /Rect [ x11 y11 xur yur ] % We'll type /Rect \Rect
  /T (<title>)                 % all have titles except the radio buttons
  < Key-value pairs that characterize the type of Widget >
  < Key-value pairs that describe the appearance of the Widget >
  < (optional) Actions to be performed >
  < Other optional key-value pairs >
/ANN pdfmark

```

*Template Notes:* <Key-value pairs that characterize the type of Widget>: The type of widget is determined by the /FT key and the /Ff key. /FT /Btn determines a button field, /FT /Ch defines a choice field, and /FT /Tx identifies a text field. Additional keys may be necessary; The /Ff key is used to refine the classification of the widget (e.g., list box versus combo box).

- All form annotations are required to have a title key (all *except* the kids of a radio button field): '/T (<mytitle>)'.

- Beginning with **Forms 3.5**, there is available a USER'S NAME field. To enter the user's name, use the /TU key: '/TU (<user name>)'.

- <Key-value pairs that describe the appearance of the Widget>: Includes such attributes as thickness, color, and style of the boundary; and color, font, and size of the text.

It is possible to define any of the standard, built-in appearances (the ones available through [Exchange](#)), or [Form XObjects](#) can be used to create a customized appearance. In both cases, the `/AP` or the `/MK` key is used.

- `<(optional) Actions to be performed>`: Any action can be defined using the `/A` key. Always use the ‘[Custom Action Method](#)’ as explained in the article on [Links](#).

**Important.** When defining actions associated with a form field, the key `/A` is used instead of `/Action` key that was used for links.

- `<Other optional key-value pairs>`: Additional keys can be used to define such attributes as highlighting of the annotation, alignment of text within the field, etc. ■

- **The Field flag `/Ff`**

The key `/Ff` is very important for setting many attributes of a form field; it is also used to distinguished between different types of fields.

[TABLE 2](#) on page 18 lists the various values of the `/Ff` key, as well as their  $\text{\TeX}$  macro equivalents. (The macros are listed on [page 14](#) and are contained in the input file `acroform.tex`).

The different attributes can be combined through addition or the logical ‘or’. Thus, `/Ff \FfPushButton \NoExport or` is equivalent to typing `/Ff 65540`.

In the [Adobe](#) documentation, the bit number is ‘base 1.’ That is, the least significant bit is termed bit 1. (This is usually referred to as ‘bit 0’ in programming circles.) Thus, for example, bit 19 is computed as  $2^{19-1} = 2^{18} = 262144$ .

**TABLE 2 Key-Value Pairs for Field Flag /Ff**

Key-Value	Description	$\TeX$ Macro Version
<code>/Ff 1</code>	bit 1: Read Only	<code>/Ff \FfReadOnly</code>
<code>/Ff 2</code>	bit 2: Required	<code>/Ff \FfRequired</code>
<code>/Ff 4</code>	bit 3: No Export	<code>/Ff \FfNoExport</code>
<code>/Ff 4096</code>	bit 13: Multi Line Text	<code>/Ff \FfMultiline</code>
<code>/Ff 8192</code>	bit 14: Password Field	<code>/Ff \FfPassword</code>
<code>/Ff 16384</code>	bit 15: No Null State	<code>/Ff \FfNoNullState</code>
<code>/Ff 32768</code>	bit 16: Radio Button	<code>/Ff \FfRadio</code>
<code>/Ff 65536</code>	bit 17: Push Button	<code>/Ff \FfPushButton</code>
<code>/Ff 131072</code>	bit 18: Pop Up List	<code>/Ff \FfPopUpList</code>
<code>/Ff 262144</code>	bit 19: Edit	<code>/Ff \FfEdit</code>
<code>/Ff 524288</code>	bit 20: Sort List	<code>/Ff \FfSort</code>

- **The Field Type /FT**

The Field Type key, /FT, is used to signal the type of field the annotation represents: Button Field, Choice Field, or Text Field. The field type is further refined by the /Ff, **Flag field**.

TABLE 3		Types of Fields	
<b>Button Field</b>			
Push-button	/FT /Btn	/FT /Btn	
	/Ff \FfPushButton	/Ff 65536	
Check-Box	/FT /Btn		
Radio Button	/FT /Btn	/FT /Btn	
	/Ff \FfRadio	/Ff 32768	
<b>Choice Fields</b>			
List Box	/FT /Ch		
Pop-up Box	/FT /Ch	/FT /Ch	
	/Ff \FfPopUpList	/Ff 131072	
Combo Box	/FT /Ch	/FT /Ch	
	/Ff \FfComboBox	/Ff 393216	
<b>Text Field</b>			
Text	/FT Tx		

*Table Notes:* In [TABLE 3](#), the values of the `/Ff` key are conveniently expressed using the  $\TeX$  macros of [TABLE 2](#); the numerical values of the `/Ff` key are given for PostScript users as well.

- It may be desirable to combine several `/Ff` flags. For example, you might want a *sorted* Pop-up list; in this case, specify `'/Ff \Ff-PopUpList \FfSort or'`.

- For Radio buttons, you might not want an 'off state'. (The radio button cannot be cleared by a `Reset` command.) You would specify, `'/Ff \FfRadio \FfNoNullState or'`. ■

## ● **Highlighting /H**

Highlighting is controlled by the value of the `/H` key. The values of the `/H` key can be found in [TABLE 4](#) below along with an example of each type of highlighting.

According to [[pdfm, p. 90](#)], an annotation is highlighted when

1. the user clicks the mouse inside the active area;
2. when the mouse button is down and the user moves the mouse out of the active area (highlighting is removed);
3. with the mouse button down, the user moves the mouse into an active area (highlighting is shown); and

4. when the mouse button is released when inside an active area (the highlighting is removed).

/H /I	Invert Highlighting
/H /N	No highlighting
/H /O	Outline Highlighting
/H /P	Push highlighting. If no down appearance is specified, the annotation is drawn inset into the page; otherwise the down appearance is shown

- **The Flags Key /F**

The value of the flags key, /F, is an integer whose value is interpreted as a bit field. The flags field provides a couple of services that are quite useful. (The two most important are the hidden field and the print field.) The flags key values can be combined by addition; thus, from TABLE 5, /F 6 is a *hidden* annotation that may be *printed* (provided the field is later reset to un-hidden).

TABLE 5 Key-Value Pairs for Flags /F	
/F 1	Invisible Flag: This flag directs the <a href="#">Reader</a> how to display an annotation whose handler is not available. (See <a href="#">[pdfs, p. 88]</a> .)
/F 2	Hidden Flag: When bit 2 is set (/F 2) the annotation is hidden (not shown) with no user interaction.
/F 4	Print Flag: When bit 3 is set (/F 4) the annotation is printed; otherwise it is not.

[Hidden Fields.](#) Hidden fields have a variety of applications: special effects, help menus, hidden data, etc. They can optionally be shown to the user by clearing bit 2 of the /F key.

[Printable Fields.](#) An electronic document may need buttons so the user can navigate within the document or submit form data the user has filled in. When and if the document is to be printed, it may not be desirable for the buttons to be printed; accordingly, the document author can create a button with the print flag cleared (which is the default). Or, perhaps, the form annotations need to be printed; in this case, the print field flag is set.

- **Border Style /BS**

The border style is controlled by the `/BS` key. The value of `/BS` is a dictionary of key-value pairs that *may* specify the width and style of the border. Below is a list of permissible entries in the `/BS` dictionary.

TABLE 6 Key-Value Pairs for the BS dictionary	
<b>W (width)</b>	
<code>/W &lt;n&gt;</code>	Border <n> pts in width (default: 1 pt)
<b>S (subtype)</b>	
<code>/S /S</code>	Solid border
<code>/S /B</code>	Beveled border
<code>/S /I</code>	Inset border (opposite of beveled)
<code>/S /U</code>	Underlined border
<code>/S /D</code>	Dashed border (use <code>/D</code> key for dash pattern)
<b>D (dash array)</b>	
<code>/D [&lt;n&gt;]</code>	dash pattern (default: [3])

For example, the code `/BS << /W 2 /S /B >>` gives an annotation with a beveled border (`/S /B`) that is 2 pts wide (`/W 2`).

## 1.5. Button Field: Push Buttons

Before getting into the actual construction of push buttons, it is necessary to introduce the keys `/MK` and `/DA`.

- **Additional Keys: `/DA` and `/MK`**

`/DA`. This key determines the default appearance. Through it you can specify the *font style*, the *font size*, and *font color*. Let's illustrate through a couple of examples.

The code

```
▶ /DA (/Helv 12 Tf 0 g)
```

specifies that the text appearing on the button should be **Helvetica**, 12 pts in size, (`Tf` is the set font and size PDF text state operator) and the color of the text is black (`0 g = 0 setgray`). Notice that we use the for character font names defined by [Acrobat](#), see [TABLE 1](#).

The code

```
▶ /DA (/HeB0 0 Tf 0 0 1 rg)
```

sets the text as **Helvetica-BoldOblique**, the `0 Tf` sets the font size on `AUTO`, and the color is **blue**: `0 0 1 rg = 0 0 1 setrgbcolor`.



[/MK](#). The key `/MK` is a dictionary whose entries describe the appearance of the button. I'll illustrate the entries by example.

▶ Dictionary entries of the `/MK` key:

```
/MK << /BC [ 0 0 0 ]           % border color in rgb
      /BG [.7529 .7529 .7529 ] % background color in rgb
      /CA (Button)             % text of normal appearance of button
      /RC (Go To)              % text of rollover appearance
      /AC (Page 3)             % text of pressed appearance
>>
```

These are the `/MK`-specifications for [EXAMPLE 1.1](#). ■

Thus, the two keys `/DA` and `/MK` together define the appearance of the button. These keys are used to help produced the 'stock' buttons that are normally available through [Exchange](#). You can also create your own button faces using [Form XObjects](#), see [EXAMPLE 1.3](#)

**Example 1.1. Standard Push Button.** Let's create a beveled button designed to jump to page three of the current document.

```
\htxt{\Bbox[72pt,20pt]}\special{ps: %
[ /Subtype /Widget                % Widget = form annotation
  /Rect \Rect                      % Bounding Box
  /T (pb1)                         % Title key
  /FT /Btn                         % Button Field
  /Ff \FfPushButton                % /Ff 65536 = push button
  /H /P /F 4                       % Push Highlight and Printable
  /BS << /W 2 /S /B >>             % BS dictionary
  /MK << /BC [ 0 0 0 ]             % Boundary color
    /BG [.7529 .7529 .7529]        % Background Color (gray)
    /CA (Button)                   % Normal text on button
    /RC (Go To)                    % Rollover text on button
    /AC (Page 3) >>               % Pushed text on button
  /DA (/Helv 10 Tf 0 g)            % Black Helvecia font
  /A << /S /GoTo                   % Action: GoTo Page 3
    /D [ {Page3} /XYZ null null null ] >>
/ANN pdfmark}\unhbox\bbox
```

*Example Notes:* The same button can be created using [Exchange](#).

- The use of /RC and /AC are optional; you may not want a rollover or push appearance.

- The type of button can be changed from beveled /S/B within the boundary style dictionary, /BS, to one of the alternates given in **TABLE 6**.

- Generally, you can create any number of variations on the same button by changing the values of the different keys.

- The action key, /A uses the ‘custom action method’ to describe the action. See **EXAMPLE 1.8** for information on this action. Forms use the /A key, while a link uses the /Action key. Example 1.1. ■

**Example 1.2. A Rotated Push Button.** We can modify the last button by rotating it 90° using the /R key within the /MK dictionary. To begin with, you’ll notice the dimensions of the bounding box /Bbox have been interchanged from [72pt,20pt] to [20pt,72pt]; also, the \hbox has been lowered 60pt (72 pt – \baselineskip).

```
\txt{\lower60pt\Bbox[20pt,72pt]}\special{ps: %
[ /Subtype /Widget                               % Widget = form annotation
  /Rect \Rect                                     % Bounding Box
  /T (pb2)                                        % Title key
  /FT /Btn                                       % Button Field
  /Ff \FfPushButton                             % /Ff 65536 = push button
  /H /P /F 4                                    % Push Highlight and Printable
  /BS << /W 2 /S /B >>                         % BS dictionary
```

```

/MK << /BC [ 0 0 0 ]           % Boundary color
      /BG [.7529 .7529 .7529]   % Background Color (gray)
      /CA (Button)             % Normal text on button
      /RC (Go To)             % Rollover text on button
      /AC (Page 3)            % Pushed text on button
      /R 90                    % rotate 90 degrees
/DA (/Helv 10 Tf 0 g)         % Black Helvecia font
/A << /S /GoTo                % Action: GoTo Page 3
      /D [ {Page3} /XYZ null null null ] >>
/ANN pdfmark}\unhbox\bbox

```

**Example 1.3.** [XObjects: Push Button using /AP](#). If you want to create your own special button—one that cannot be constructed from the ‘in stock parts,’ as illustrated in [EXAMPLE 1.1](#)—you need to use [Form XObjects](#).

**The /AP key.** The *Appearance Key* is used to describe (customized) face appearances. The key /AP is a dictionary that contains a key for a *normal appearance* /N, and optionally, a key for a *rollover appearance*, /R, and *pushed appearance* /D. The values of these keys are either an indirect reference to an [XObject](#) or a dictionary.

The code of the button below uses the appearance key /AP; here's a closer look at /AP.

```
/AP << /N {eCalc}           % ● Normal appearance
      /R {reCalc}           % ● Rollover appearance
      /D {peCalc}           % ● Down (pushed) appearance
>>
```

The value of each of the keys /N, /R and /D is an indirect reference to a [Form XObject](#). Click on the green bullets to see the code which in the [SECTION 1.14](#) on page 84. ■

▶

```
\leavevmode\htxt{\Bbox[80pt,20pt]}
\special{ps: %
[ /Rect \Rect
  /Subtype /Widget /T (GoToeCalc)
  /F 4 /FT /Btn /Ff \FfPushButton \FfNoExport or
  /H /P
  /AP << /N {eCalc} /R {reCalc} /D {peCalc} >>
  /A << /S /GoTo /D [ {Page1} /Fit ] >>
  /ANN pdfmark}\unhbox\bbox
```

Rotating the faces requires an entirely different set of eps graphics.

```

▶ \htxt{\lower70pt\Bbox[20pt,80pt]}%
\special{ps: %
[ /Rect \Rect
 /Subtype /Widget /T (rotGoToeCalc)
 /F 4 /FT
 /Btn /Ff \FfPushButton \FfNoExport or
 /H /P
 /AP << /N {roteCalc} /R {rotreCalc} /D {rotpeCalc} >>
 /A << /S /GoTo /D [ {Page1} /Fit ] >>
 /ANN pdfmark
}\unhbox\bbox

```

Example 1.3. ■

## 1.6. Button Field: The Checkbox

The forms plug-in of [Exchange](#) comes with some “off-the-shelf” checkboxes; checkboxes that use the symbols: check, circle, cross, diamond, square and star—all from the [ZapfDingbats](#) font set.

- **The /DA and /MK Keys**

**/DA.** This key determines the default appearance. Through it you can specify the *font style*, the *font size*, and *font color*. For checkboxes, we want to use the [ZapfDingbats](#) font, so by [TABLE 1](#), we specify the font as `/ZaDb`.

The code

```
▶ /DA (/ZaDb 12 Tf 0 g)
```

specifies that the text appearing on the button should be **Helvetica**, 12 pts in size, (**Tf** is the set font and size PDF text state operator) and the color of the text is black (**0 g = 0 setgray**). Notice that we use the for character font names defined by [Acrobat](#), see [TABLE 1](#).

In the code

```
▶ /DA (/ZaDb 0 Tf 0 0 1 rg)
```

the **0 Tf** sets the font size to **AUTO**—automatically adjusting the font to fit into the box—and the color is **blue: 0 0 1 rg = 0 0 1 setrgbcolor**. ■

**/MK**. The key **/MK** is a dictionary whose entries describe the appearance of the button. For checkboxes, the **/CA** key appears in the **/MK** dictionary and is used to define the check symbol to be used in the appearance.

## ▶ Dictionary entries of the /MK key:

```

/MK << /BC [ 0 0 0 ]           % black border in rgb           (2)
      /CA (4)                   % /CA (4) is check, see TABLE 7
>>

```

These are the /MK-specifications for [EXAMPLE 1.4](#). ■

Thus, the two keys /DA and /MK together define the appearance of the button. These keys are used to help produced the ‘stock’ buttons that are normally available through [Exchange](#). You can also create your own button faces using [Form XObjects](#), see [EXAMPLE 1.3](#).

[TABLE 7](#) lists the values of the key-values used to designate the type of ‘check’ to appear in the checkbox. These are entries in the /MK dictionary. See [\(2\)](#) for an example.

TABLE 7	Check Box Symbols
/CA (4)	Check
/CA (1)	Circle
/CA (8)	Cross
/CA (u)	Diamond
/CA (n)	Square
/CA (H)	Star

The checkboxes in TABLE 7 were defined using the default appearance: /DA (/ZaDb 0 Tf 0 g). The symbols, therefore, are set on ‘AUTO’ sizing; this means they are enlarged as much as possible to fit inside the bounding box.

**Example 1.4. A Standard Checkbox.** Here is a standard checkbox that is certainly easy to construct using [Exchange](#). The use of `pdfmarks` is important for documents that are constantly undergoing revisions. The author cannot afford the time to place form elements in by hand after every revision.

## ▶ Check here!

```

\htxt{\lower1pt\Bbox[10pt,10pt]}\special{ps: %
[ /Rect \Rect
/T (CB4)
/FT /Btn           % Check box: Table 3
/F 4               % Printable: See Table 5
/H /N              % highlight = none
/BS << /W 1 /S /S >> % Border Style: See Table 6
/MK << /BC [ 0 0 0 ] % See line (2)
      /CA (4) >>   % Standard Check: See Table 7
/DA (/ZaDb 10 Tf 0 g) % Use black 10pt ZapfDingbats font
/Subtype /Widget /ANN pdfmark}\unhbox\bbox{ } Check here!

```

The default names of the values of a checkbox are ‘Yes’ and ‘Off’. If we want the checkbox to have an initial value of ‘Yes’, we use the *value key* to specify ‘/V /Yes’:

## ▶ Do you want to buy?

```

\htxt{\lower1pt\Bbox[10pt,10pt]}\special{ps: %
[ /Rect \Rect
/T (CB4Yes)
/FT /Btn           % check box: Table 3
/F 4 /H /P         % printable, push highlighting
/V /Yes            % Set Checkbox to ‘Yes’
/BS << /W 1 /S /S >>
/MK << /BC [ 0 0 0 ] /CA (4) >>

```

```

/DA (/ZaDb 10 Tf 0 g)
/Subtype /Widget
/ANN pdfmark}\unhbox\bbox{} Do you want to but?

```

Example 1.4. ■

**Example 1.5. Redefining /Yes.** The default names, or *values*, of the current state of a checkbox are ‘Yes’ and ‘Off’. Adobe requires the unchecked box to have a value of ‘Off’, but when the box is checked, the form field can take on whatever value desired by the document author. This may be useful if the form data is to be stored in a database in which the fields already pre-defined names.

The value of ‘Yes’ can be changed by modifying the appearance state key, /AP. The following example illustrates the technique.

▶ Check here!

```

\txxt{\loweript\Bbox[10pt,10pt]}\special{ps: %
[ /Rect \Rect
/T (CB5)
/FT /Btn % Check box: Table 3
/F 4 /H /O % Outline Highlighting
/AP << /N << /Checked /null >> >> %<-- define ‘Yes’ to be ‘Checked’
/BS << /W 1 /S /S >> % Solid border, 1pt in width
/MK << /BC [ 1 0 0 ] /CA (H) >> % Red border; use star!
/DA (/ZaDb 10 Tf 1 0 0 rg) % Red star!
/Subtype /Widget /ANN pdfmark}\unhbox\bbox{} Check here!

```


*Example Notes:* Having changed of value of ‘Yes’, should now you want the checkbox to be initially on—and the user must go to the trouble of turning it off explicitly—just include the key-value pair:

```
/V /Checked
```

Example 1.5. ■

**Example 1.6.** [Checkbox with Shadow](#). We can go slightly beyond the packaged checkboxes that [Acrobat](#) provides. In the example that follows, I’ve used a shadowed box, taken from the [ZapfDingbats](#) font set. At the beginning of this file, I have the  $\TeX$  code

```
\font\zd=zd at 14pt
```

The name of the [ZapfDingbats](#) font set is `zd`; actually, this is a command to  $\TeX$ , and `zd` refers to the file `zd.tfm`. Within this font set, I use `{\zd\char"72}` which gets typeset as , a nice shadowed box.

▶  Click Me!

```
\htxt{\lower1pt\hbox{\zd\char"72}}\special{ps: %
[ /Rect \Rect
/T (CB6)
/FT /Btn
/F 4 /H /N % printable; no highlighting
/MK << /CA (8) >> % cross
/DA (/ZaDb 10 Tf 0 0 1 rg) % blue
/Subtype /Widget
```

```
/ANN pdfmark}\unhbox\bbox{} Click Me!
```

Example 1.6. ■

**Example 1.7.** A [Checkbox with two Appearances](#). Here is an example of a checkbox: The ‘Off’ appearance is a check mark and the ‘Yes’ appearance is a cross.

▶  Make your choice: Check or Cross!

```
\htxt{\lower1pt\hbox{\zd\char"72}}\special{ps:%
[ /Rect \Rect          % Click on green to see XObject code
  /T (CB7)             % • /Crossed {xCross}
  /FT /Btn /F 4 /H /N   % • /Off {xCheck}
  /AP << /N << /Crossed {xCross} /Off {xCheck} >> >>
  /Subtype /Widget
/ANN pdfmark}\unhbox\bbox{} Make your choice: Check or Cross!
```

*Example Notes:* This annotation uses the *appearance key* /AP. The only entry in the dictionary /AP is the key-value pair

▶ /N << /Crossed {xCross} /Off {xCheck} >>

This is the *normal appearance* key and its value, a dictionary.

- The *normal appearance* key /N has two entries in its dictionary: /Off {xCheck} defines the ‘Off’ appearance, its value is an indirect reference to an [XObject](#), {xCheck}; /Crossed {xCross} defines the ‘On’ or ‘Yes’ appearance of the checkbox, its value is an indirect reference to the [XObject](#), {xCross}.

■ The example also demonstrates another way of redefining of value of the ‘Yes’ appearance. If you save the form data to a local file using [Exchange](#) you will see the name of the ‘Yes’ value is indeed ‘Crossed’.

Example 1.7. ■

**Example 1.8.** [Tic-Tac-Doe](#). As an application of the techniques of [EXAMPLE 1.7](#), let’s construct a simple [Tic-Tac-Toe Game](#).

The [Tic-Tac-Toe](#) Board has overlapping link and checkbox regions. Initially, the board is empty, the links are active and the checkboxes are *hidden*. Click on one of the nine regions to prompt a link annotation to perform the action of showing the hidden checkbox.

The logo for Tic-Tac-Toe is displayed on a yellow rectangular background. The word "Tic" is in red, "Tac" is in green, and "Toe" is in blue, with hyphens between the words.

Below is a slightly modified version of the game; I’ve removed the colored background and the [Tic-Tac-Toe](#) logo. “Hey, it rhymes!” (To quote my son, Alexander, of 4.5 years.)

%

% TeX macro to create two overlaid regions: a link and a checkbox

## Section 1: Forms

```
\def\TTT#1#2{\htxt{\Bbox[20pt,20pt]}\special{ps: %  
[ /Rect \Rect /H /O  
  /Action << /S /Hide /T (TTT#1#2) /H false >>  
  /Subtype /Link  
/ANN pdfmark  
[ /Rect \Rect /T (TTT#1#2)  
  /FT /Btn /F 6 /H /N  
  /AP << /N << /On {x0} /Off {xX} >> >>  
  /Subtype /Widget  
/ANN pdfmark}\unhbox\bbox}%
```

Now let's store the titles of these nine checkboxes for later use.

```
%  
% This uses /PUTINTERVAL, see \[pdfm, p. 23\]  
\special{ps:  
[ /_objdef {TTTFields} /type /array /OBJ pdfmark  
[ {TTTFields} 0 [ (TTT11) (TTT12) (TTT13)  
                  (TTT21) (TTT22) (TTT23)  
                  (TTT31) (TTT32) (TTT33) ]  
/PUTINTERVAL pdfmark}
```

Now let's construct the reset button. This button uses multiple actions: first, we hide all of the checkbox fields; secondly, we reset them to their original “/Off” setting.

```
\def\TTTResetButton{\htxt{\Bbox[40pt,20pt]}\special{ps: %
```

## Section 1: Forms

```
[ /Rect \Rect
  /FT /Btn
  /T (ResetTTT)
  /Ff \FfNoExport \FfPushButton or
  /H /P /F 4
  /BS << /W 1 /S /B >>
  /DA (/Helv 10 Tf 0 g)
  /MK << /BC [ 0 0 0 ] /BG [.7529 .7529 .7529 ]
      /CA (Restart) /RC (Game) /AC (Begin!) >>
  /A << /S /Hide /H true /T {TTTFields} /Next
      << /S /ResetForm /Fields {TTTFields} >> >>
/Subtype /Widget /ANN pdfmark}\unhbox\bbox}
```

%

% Now for the actual typesetting of Tic-Tak-Toe

%

```
\centerline{\vtop{\offinterlineskip
\hbox{\TTT11\TTT12\TTT13}
\hbox{\TTT21\TTT22\TTT23\rlap{\quad\TTTResetButton}}
\hbox{\TTT31\TTT32\TTT33}}}
```

*Example Notes:* There are a number of improvements needed to this game: Remove the borders of the boxes, spread the buttons out and insert the horizontal and vertical rulings of a standard Tic-Tac-Toe game.

- Sometimes, when we make a choice, we want to *take it back*. Technically this is illegal, but you can include a reset button for *each* of the nine positions.

- There is another way of constructing this game without the need for a reset button for each position. Create a linked list of buttons faces. The action of each button is to hide itself and show the next button. In this way, you can cycle through the different faces. Why don't you give it a try!

Example 1.8. ■

## 1.7. Button Field: Radio Buttons

In this section we demonstrate how to construct a radio button field. The keys and techniques used are quite similar to those enumerated in [SECTION 1.6](#) on checkboxes.

A radio button field is two or more checkboxes; one of the boxes is the **Parent** and the others are the **Kids**.

**Important.** Only the **Parent** checkbox has a title key **T**. The **Parent** and its **Kids** all have indirect object references. The values of the checkboxes in the radio button field all must have *different values*. This is accomplished by using the technique of [EXAMPLE 1.5](#).

**Example 1.9.** A Standard Radio Button Field. Below is a radio button field of a mathematical nature.

**Problem.** Let  $f(x) = \sin(x^2)$ , which of the following is  $f'(x)$ ?

$$f'(x) = \cos(x^2)$$

$$f'(x) = 2x \cos(x^2)$$

$$f'(x) = \sin(2x)$$

$$f'(x) = \cos(2x)$$

The code for this radio button field follows.

```
{\bf Problem.} Let  $f(x)=\sin(x^2)$ , which of the following
is  $f'(x)$ ?
\begingroup\parindent30truept\parskip0pt
%
% Parent: Radio button 1
\item{\htxt{\Bbox[10pt,10pt]}\special{ps:
[/_objdef {RBF91} % Indirect reference
/Rect \Rect /T (RBF9) % Only Parent gets a title /T
/FT /Btn % Checkbox type
/Ff \FfRadio \FfNoNullState or % Radio button
/H /P /F 4
/AP << /N << /C1 /null >> >> % Define value to be C1
/DA (/ZaDb 0 Tf 0 g ) % Auto sizing of font
/Kids [{RBF92}{RBF93}{RBF94}] % /Kids array of 'kids'
```

## Section 1: Forms

```
/BS << /W 1 /S /S >> /MK << /BC [ 0 0 0 ] /CA (8) >>
/Subtype /Widget /ANN pdfmark}\unhbox\bbox} $f'(x)=\cos(x^2)$
%
% Kid: Radio button 2
\item{\htxt{\Bbox[10pt,10pt]}\special{ps:
[ /_objdef {RBF92} % Indirect reference
/Rect \Rect
/H /P
/AP << /N << /C2 /null >> >> % Define value to be C2
/Parent {RBF91} % Whose the parent?
/BS << /W 1 /S /S >> /MK << /BC [ 0 0 0 ] /CA (8) >>
/Subtype /Widget /ANN pdfmark}\unhbox\bbox} $f'(x)=2x\cos(x^2)$
%
% Kid: Radio button 3
\item{\htxt{\Bbox[10pt,10pt]}\special{ps:
[ /_objdef {RBF93} % Indirect reference
/Rect \Rect
/H /P
/AP << /N << /C3 /null >> >> % Define value to be C3
/Parent {RBF91} % Whose the parent?
/BS << /W 1 /S /S >> /MK << /BC [ 0 0 0 ] /CA (8) >>
/Subtype /Widget /ANN pdfmark}\unhbox\bbox} $f'(x)=\sin(2x)$
%
% Kid: Radio button 4
\item{\htxt{\Bbox[10pt,10pt]}\special{ps:
[ /_objdef {RBF94} % Indirect reference
```

```

/Rect \Rect
/H /P
/AP << /N << /C4 /null >> >> % Define value to be C3
/Parent {RBF91} % Whose the parent?
/BS << /W 1 /S /S >> /MK << /BC [ 0 0 0 ] /CA (8) >>
/Subtype /Widget /ANN pdfmark}\unhbox\bbox} $f'(x)=\cos(2x)$
\endgroup

```

*Example Notes:* Notice that *only* the parent has the title key, /T; also, *only* the parent has the /FT and /Ff flags.

- Within the parent, the /Kids key is used to list out the indirect addresses, in array form, of the kids. Each of the kids has a /Parent key the value of which is the indirect reference to the parent.

- The appearances of the radio buttons do not have to be identical: they can differ in check mark symbol, or color, or boundary thickness, etc. See the next example. Example 1.9. ■

**Example 1.10.** [Default Value, Initial Value, Clear to Default.](#) In this example, we demonstrate how to give the radio button field an initial value. Here, the /V key is used.

A /ResetForm is also provided; when a form field is reset, it is reset to its *default value*, the value defined by the /DV key.

- ▶ Filing a tax return?
  - Not Filing
  - Filing

*Example Notes:* Click on ‘Not Filing’ first, then push the ‘Clear to Default’ button. The field will return to its original state.

- In the listings below, the value of the first button is ‘C1’ and that of the second is ‘C2’.

- Without the /V key, the field would be in the ‘off’ state initially. Clicking on the ‘Clear to Default’ button would cause ‘C2’ to be the current value.

- Without the /DV key (and with the /V key) the field would open up with ‘C2’ as the value of the field, but now the ‘Clear to Default’ button turns off *all the buttons*.

- Note that the appearance of the second radio button is different from the first. Radio buttons of the same field share certain common properties, but they do not have to share the same appearance. ■

Here is the code for the above button field, including all the T<sub>E</sub>X structure.

```
\line{% Begin a hbox of width \hsize
%
```

## Section 1: Forms

```
% Make a gray background using a \vrule
\pushcolor{.7529 .7529 .7529 }\smash{\rlap
{\vrule height5pt depth34pt width\hsize}}\popcolor %
%
% Construct Radio Button in \vtop, we'll have two columns
\vtop{\hsize=.49\hsize\parskip0pt
\rtr Filing a tax return?\par      % Note: \rtr makes a red triangle
\hglue3pt\htxt{\lower1pt\Bbox[10pt,10pt]}\special{ps: %
[ /_objdef {RBF101}
/Rect \Rect /T {RBF101}           % This is the Parent
/FT /Btn /Ff \FfRadio             % Radio Field
/F 4 /H /P                         % Push highlighting
/BS << /S /S >>                   % Solid Border (/W 1 default)
/DA (/ZaDb 10 Tf 0 g)             % Default Resources
/MK << /BC [ 0 0 0 ] /CA (1) >> % /CA (1) = bullet
/AP << /N << /C1 /null >> >>     % Give the value a name: C1
/Kids [{RBF102}]                  % List the kids
/Subtype /Widget /ANN pdfmark}\unhbox\bbox{} Not Filing

\hglue3pt\htxt{\lower1pt\Bbox[10pt,10pt]}\special{ps: %
[ /_objdef {RBF102}
/Rect \Rect /Parent {RBF101}     % List Parent
/H /P /F 4
/BS << /S /I >>                   % Inset Border!
/MK << /BC [ 0 0 0 ] /CA (1) >> % /CA (1) = bullet
/AP << /N << /C2 /null >> >>     % Give the value a name: C2
```

## Section 1: Forms

```
/DV /C2 % Make C2 default value (on reset)
/V /C2 % Make C2 initial value of field
/Subtype /Widget /ANN pdfmark}\unhbox\bbox{} Filing}
\hfil % Fill gap between boxes
%
% Construct Inset Pushbutton to ResetForm
\top{\hsize=.49\hsize
\leavevmode\htxt{\lower20pt\Bbox[90pt,20pt]}\special{ps:
[ /Rect \Rect
/T (RtrnDeflts)
/FT /Btn /Ff \FfPushButton
/H /P /F 4 /BS << /S /I >>
/MK << /BC [ 0 0 0 ] % No /BG specified; therefore, invisible! (3)
/CA (Clear to Defaults) /AC (Boo!) >>
/DA (/HeB0 10 Tf 0 0 1 rg)
/A << /S /ResetForm /Fields [(RBF10)] >>
/Subtype /Widget /ANN pdfmark}\unhbox\bbox{}
} % end \line
```

*Code Notes:* In line (3), the `/BG` key was not specified. This implies the background color is the same as the underlying page. In this case, I've created a gray background and so the button appears the same gray color too.

Example 1.10. ■

**Example 1.11.** [Radio Button Field using Macros](#). Having seen [EXAMPLE 1.10](#), we are ready to automate the process of constructing a radio button field using  $\text{\TeX}$  macros.

Here is one set of macros that will produce the desired effect.

```
%
% Introduce some useful counters.
\newtoks\Kids \newcount\altno
%
% Macro to create a radio ‘environment’.
\def\BeginRadio#1{\gdef\RadioTitle{#1}\global\Kids={}\global\altno=0
  \begingroup\parindent30truept\parskip0pt
  \def\Item{\global\advance\altno1      % \altno = radio button num
    %
    % \Kids keeps track of the kids. Here we expand \Kids and
    % include the new kid on the block using \xdef\temp. Then
    % we redefine \Kids with the new entry included.
    %
    \xdef\temp{\the\Kids{\RadioTitle\the\altno}}
    \global\Kids=\expandafter{\temp}
    %
    % I’m using \item{} but this part can be changed to anything
    %
    \item{\htxt{\Bbox[10pt,10pt]}\special{ps: %
      [ /_objdef {#1\the\altno}          % object reference
```

```

/Rect \Rect                                % bounding rectangle
\ifnum\altno=1                              % Insert Parent stuff
/T (#1) /FT /Btn /Ff \FfRadio \FfNoNullState or
/Kids {#1Kids} /DA (/ZaDb 0 Tf 0 g)
\else /Parent {#11} \fi                    % Insert Kids stuff
% Define value: A1, A2, etc.
/AP << /N << /A\the\altno\space/null >> >>
/H /P /F 4                                  % Push and Print
/BS << /W 1 /S /S >>                       % Solid borders, width 1pt
/MK << /BC [ 0 0 0 ] /CA (8) >> % Black border with cross
/Subtype /Widget /ANN pdfmark}\unhbox\bbox}%
} % end \Item
} % end \BeginRadio
%
% Now terminate the radio ‘environment’.

```

```

\def\EndRadio{\endgroup\special{ps:
[ /_objdef {\RadioTitle Kids} /type /array /OBJ pdfmark
[ {\RadioTitle Kids} 0 [\the\Kids] /PUTINTERVAL pdfmark}}

```

The `\EndRadio` uses `/_objdef ... /type /array /OBJ pdfmark` to create an array, then inserts into this newly created array the listing of the `\Kids` token register using `/PUTINTERVAL`. See [SECTION 1.13](#) for additional details.

**Example 1.12.** [A Simple Method](#). In the previous examples, a fairly complicated method of constructing radio button fields was demonstrated. Recently, Sebastian Rahtz has begun working on an extension of his `hyperref` package to include a forms capability. His method is simpler and very straight forward.

Basically, his method is to construct button fields with a `radio` attribute. The buttons all have the same title, but each has a different value.

► What level of expertise do you have in the world of T<sub>E</sub>X?

T<sub>E</sub>X Novice

T<sub>E</sub>X Master

T<sub>E</sub>X Grandmaster

```
\txt{\lower1pt\Bbox[10pt,10pt]}\special{ps: %
[ /Subtype /Widget
  /F 4 /T (TeX)
  /FT /Btn /Ff \FfRadio \FfNoNullState or      % 49152
  /H /P /BS << /W 1 /S /S >>
  /MK << /BC [ 1 0 0 ] /CA (u) >>
  /DA (/ZaDb 10 Tf 0 0 0 rg)
  /V /Yes
  /AP << /N << /level1 /null >> >>
/Rect \Rect /ANN pdfmark}
\unhbox\bbox \TeX{ } Novice\quad
\txt{\lower1pt\Bbox[10pt,10pt]}\special{ps: %
```

```
[ /Subtype /Widget
  /F 4 /T (TeX)
  /FT /Btn /Ff \FfRadio \FfNoNullState or
  /H /P /BS << /W 1 /S /S >>
  /MK << /BC [ 1 0 0 ] /CA (u) >>
  /DA (/ZaDb 10 Tf 0 0 0 rg) /V /Off
  /AP << /N << /level2 /null >> >>
/Rect \Rect /ANN pdfmark}
\unhbox\bbox \TeX{} Master\quad
\htxt{\lower1pt\Bbox[10pt,10pt]}\special{ps: %
[ /Subtype /Widget
  /F 4 /T (TeX)
  /FT /Btn /Ff \FfRadio \FfNoNullState or
  /H /P /BS << /W 1 /S /S >>
  /MK << /BC [ 1 0 0 ] /CA (u) >>
  /DA (/ZaDb 10 Tf 0 0 0 rg)
  /V /Off
  /AP << /N << /level3 /null >> >>
/Rect \Rect /ANN pdfmark}
\unhbox\bbox \TeX{} Grandmaster
```

*Example Notes:* When the button field is optimized using [Exchange](#), the field is rewritten to look like my previous examples. In this example, we let [Exchange](#) do all the work for us. Example 1.12. ■

## 1.8. Choice Field: The List Box

A list box is a list of items displayed within a rectangle. The user can choose at most one of the items from the list.

To define a list box, by use `/FT /Ch` with no qualifying field flags, `/Ff`. There is an `/Opt` key that is used to define the objects the user is to choose from. Read the comments at the end of the examples for more details on the `/Opt` key.

### Example 1.13. [Standard List Box.](#)

► To the right are listed some of our fine garments. We are sure you will agree with us that the quality of these textile products cannot be matched by any of our competitors. We are so confident that you'll make a purchase, we are willing to give a hefty discount. All items are now available for a flat fee of **\$19.95**. That's **\$19.95!** Buy now, *replace later!* ■

Here is the code, with the special paragraph effects removed.

```
\htxt{\Bbox[42pt,50pt]}\special{ps: %
[ /Rect \Rect
  /T (LB13)
  /FT /Ch /Ff \FfSort      % /FT /Ch = List Box; and sort list
  /F 4                    % Printable
```

```

/BS << /S /I >>           % Inset border 1pt wide (/W 1 is default)
/MK << /BC [ 0 0 0 ] >>   % Black borders, transparent background
/DA (/Helv 10 Tf 1 0 0 rg ) % 12 pt, red Helv type
/Opt [(Socks)(Pants)(Shirts)(Coats)(Ties)(Belts)(Shoes)] % The List
/DV (Pants)                % Default value: Pants
/Subtype /Widget /ANN pdfmark}\unhbox\bbox

```

*Example Notes:* The `/Opt` key is used to define the list. The value of the `/Opt` key is an *array of strings* or an *array of arrays*. The latter case of an array of arrays is taken up in the next example.

- In our case, the entries in the `/Opt` array serve a dual purpose: They are the text that appears in the list box; and (2) they are the potential *values* of the list box annotation.

- I've used `/Ff \FfSort`. **This does nothing** other than to put a check on the 'Sort Item' box in the [Forms fill-in Menu](#) of [Exchange](#). The list is sorted by the [Forms Plug-in: Bring the forms fill-in menu up for the list and click on the 'O.k' button](#); the list is now sorted.

- The key `/DV` determines the item that is highlighted when the list is initially viewed. This is also the item that the list box will reset to under the command action `/ResetForm`.

- A List Box is scrollable. If there are more items listed than the size of the bounding rectangle allows, when the rectangle becomes active, scroll bars appear in the right boundary.

- Enough space should be left to the right of the text to allow the scroll bar to appear without covering up the text. I added an additional 6pt onto the width of the box for this reason.
- The highlighting key, /H, seems to have no effect on a choice field annotation. Example 1.13. ■

### 1.9. Choice Field: The Pop-Up Box

A pop-up box is a list box that “pops-up”! When there is limited real estate available on the page, the Pop-up Box may be useful. The Pop-up Box and the **Combo Box** are quite similar; there is no editing of the alternatives in a Pop-up Box, there is in a Combo Box.

To define a Pop-up Box, use the keys

```
/FT /Ch /Ff 131072,
```

or you can use the macro approach,

```
/FT /Ch /Ff \FfPopUpList.
```

(See **TABLE 3**)

The list of alternatives is defined using the /Opt key. For a discussion of the /Opt key, see **EXAMPLE 1.13** and the comments following the next example.

**Example 1.14.** [Standard Pop-Up Box](#). In this example, we present a standard construction of a Pop-Up Box. Note the value of the value of the `/Opt` key. Consider the following (I hope) working example. Before trying out this example, read the *Script Notes* first.

► Where do you want to go? Make a choice:

```
%
% First define a convenient URL macro
\def\URL{http://www.math.uakron.edu/\noexpand~dpstory}
%
% Now define the pop-up box
\htxt{\lower4pt\Bbox[144pt,16pt]}\special{ps: %
[ /Subtype /Widget
  /Rect \Rect
  /T (dest)
  /FT /Ch /Ff \FfPopUpList /F 4 % Print the Pop-Up Box
  /BS << /S /I >> % Inset the border
  /MK << /BC [ 0 0 0 ] /BG [0.98 0.92 0.73] >> % Boundary colors
  /Opt [ % Define Url's and titles
    [ (\URL)( Homepage of D. P. Story)]
    [ (\URL/e-calculus.html)( e-Calculus Homepage)]
    [ (\URL/mpt_home.html)( Algebra Review Homepage)]
    [ (\URL/tutorial/mainmenu.pdf)( Main Menu (PDF))]
```

## Section 1: Forms

```
[ (\URL/tutorial/maintut.pdf)( e-Calculus Main Page (PDF))]
[ (\URL/tutorial/mainmpt.pdf)( Algebra Review Page (PDF))]
] % End /Opt
/DV (\URL) % Default value
/DA (/TiBI 10 Tf 0 0 1 rg) % Blue Bold Times Italics 12 pt
/ANN pdfmark}\unhbox\bbox% % Run the go button in now
%
% Define a button with /SubmitForm as action
\htxt{\lower4pt\Bbox[22pt,16pt]}\special{ps:
[ /Rect \Rect
/T (G014)
/FT /Btn /Ff \FfPushButton \FfNoExport or
/H /P /F 4 /BS << /S /B >>
/MK << /BC [ 0 0 0 ] /BG [.7529 .7529 .7529 ] /CA (Go!) >>
/DA (/TiB0 10 Tf 0 0 1 rg)
/A << /S /SubmitForm
/F (http://www.math.uakron.edu/cgi-bin/nph-cgiwrap/\
dpstory/scripts/nph-redir.cgi)
/Fields [(dest)] /Flags 4 >>
/Subtype /Widget /ANN pdfmark}\unhbox\bbox
```

*Pop-Up Notes:* The /Opt key requires some comment. Here, /Opt is an *array of arrays*. Each member if the array consists of two entries, both strings. The *first* string is the *value* string, the *second* string is the *appearance* string.

- Though not included explicitly, the value key `/V` is one used to store the current value of the field. ■

*SubmitForm Notes:* The action key `/A` defines the `/SubmitForm` action. The value of the `/Fields` key is an array listing the fields whose values are to be submitted to the script on the server defined by the `/F` key. See [pdfs,p. 131-132] for more details on the `/SubmitForm` action and its various keys. ■

*Script Notes:* I used Redirection Script, `nph-redir.cgi`, Version 2.3, by Jeff Carnahan ([jeffc@terminalp.com](mailto:jeffc@terminalp.com)). This script can be found at <http://www.terminalp.com/scripts/>.

- The script works perfectly with *within* an `html` documents—I've installed some pop-up menus on my own `html` pages—suggesting the script is properly installed. The above example though has its problems. On my **Win95** system, [Netscape Communicator 4.01](#) crashes near the end of loading file. Using [Netscape Navigator Gold 3.01](#), the above example works flawlessly. [Microsoft Internet Explorer 3.0](#) does not work properly either.

- The script works properly using [Netscape Communicator 4.05](#).

Example 1.14. ■

**Example 1.15.** [List Box and /DV](#). For the List Box and the [Combo Box](#), the value of the *default value key*, /DV, need not be a value of the field. As a result, you can use /DV as a labeling field.

This particular is a copy and paste example from [SECTION 1.1](#).

In [Acrobat](#), form fields of the same *type* that have the same title /T, share the same *value* (the current value of the /V key). The example below is a copy and paste job from [SECTION 1.1](#). If you entered your own particular information on that page, your response will be shown below.

▶

```

\rtr\htxt{\lower4pt\Bbox[80pt,16pt]}\special{ps: %
[ /Rect \Rect /T (testPU)
 /FT /Ch /Ff \FfPopUpList
 /F 4 /BS << /S /I >>
 /MK << /BC [ 0 0 0 ] /BG [0.98 0.92 0.73] >>
 /Opt [(Mathematics)(Statistics)(Computer Sci.)(None of these)]
 /DA (/TiRo 10 Tf 0 g) % Black 10 pt Times Roman
 /DV (Your Major?) % Initial value
/Subtype /Widget /ANN pdfmark}\unhbox\bbox% Now follow up with
\htxt{\lower4pt\Bbox[30pt,16pt]}\special{ps: % a reset button
[ /Rect \Rect /T (Reset15)

```

```

/FT /Btn /Ff \FfPushButton
/F 4 /H /O /BS << /S /I >>
/MK << /CA (Reset) /BC [ 0 0 0 ] /BG [0.98 0.92 0.73] >>
/DA (/TiRo 10 Tf 0 g) % Black 10 pt Times Roman
/A << /S /ResetForm /Fields [(testPU)] >> % reset form 'testPU'
/Subtype /Widget /ANN pdfmark}\unhbox\bbox

```

*Example Notes:* The value of /DV becomes inaccessible once you open the Pop-up list. If you don't open the list and save the data using [Exchange](#) you'll see that 'V (Your Major?)' is saved as the field value. I would assume, this default value would be the value exported by a submit button as well. Example 1.15. ■

## 1.10. Choice Field: The Combo Box

A Combo Box is a Pop-up Box in which editing by the user is permitted. To obtain a Combo Box, use

```
/FT /Ch /Ff 393216.
```

This number can be obtained by adding the flag value for a Pop-up List, 131072, and the flag value for editing, 262144. Using the T<sub>E</sub>X macro definitions on [page 14](#) we see that a Combo Box is defined by

```
/FT /Ch /Ff \FfComboBox.
```

**Example 1.16. Standard Combo Box.** Consider the following question.

► In which century was Sir Issac Newton born?

*Note:* If none of the alternatives appeals to you, you may enter your own response.

```
\txt{\lower4pt\Bbox[72pt,16pt]\special{ps: %
[ /Rect \Rect
/T (ComB16)
/FT /Ch /Ff \FfComboBox          % = /Ff 393216
/F 4 /BS << /S /I >>             % inset border, 1pt wide (default)
/MK << /BC [ 0 0 0 ] /BG [0.98 0.92 0.73] >>
/Opt [( 20th Century)( 19th Century)( 18th Century)]
/DA (/TiRo 10 Tf 0 g)
/V ( 20th Century)
/AA << /K << /S /JavaScript
      /JS (this.getField("ComB16R").value=event.value+"."); >>
>>
/Subtype /Widget /ANN pdfmark}\unhbox\bbox
```

*Example Notes:* In the case of the Combo Box, it doesn't make sense for the value of the /Opt key to be an *array of arrays*.

- Notice that I have inserted an extra space—( 20th Century) instead of (20th Century)—to create a more pleasing visual display of the choices.

- Such a form annotation can be processed by using either CGI script or JavaScript, Forms 3.5 or above required.

- The correct answer to the question is the ‘17th Century’. Your answer was the

- This particular annotation is processed by JavaScript. As was noted in EXAMPLE 1.15, annotations having *same type* and the *same title* share data. In this example, I wanted to use the data from a *Combo Box* in the *Text Field* of the previous comment. Prior to Forms 3.5, this kind of data transfer could only be done with CGI script; now, it’s simple enough to move the data with a single line of JavaScript.

Example 1.16. ■

### 1.11. Text Fields

To construct a Text Field, use /FT/Tx. This would define a *single line* text field. For Multiline Text fields, set the *field flag*: /Ff 4096 or /Ff \FfMultine.

In addition to setting up the basic field, you can specify the *quadding*: text left-justified, text centered, or text right-justified. It is also possible to limit the number of characters entered into a specific field.

- **Options: /Q and /MaxLen**

Quadding is specified as the value of the /Q key.

/Q 0	left-justified (the default)
/Q 1	centered text
/Q 2	right-justified

Quadding is active for all fields having variable text: Choice Fields and Text Fields. (See [EXAMPLE 1.19](#) below)

When inputting data into a Text Field, it is sometimes necessary to limit the number of characters. (Think of a request for the state of residence of a user. Each of the fifty states in the United States has a two character abbreviation. In this case, it is useful to limit input to two characters.) Use the /MaxLen key:

/MaxLen <n>	where <n> is the maximum number of characters accepted
-------------	---

See [EXAMPLE 1.19](#) below for an example of use.

**Example 1.17. Standard Text Field.** Here is a beginner's level plain vanilla Text field. If you had entered your name in the Text Box way back on [page 9](#), your name should appear below.

The Text Box on page 9 and the one below have the same title, /T (YourName); therefore, they will share data. You'll note that even though they share data, they don't necessarily share the same appearance!



Simple Text Box:

```
\txt{\lower4pt\Bbox[180pt,16pt]}\special{ps: %
[ /Rect \Rect
  /T (YourName)
  /FT /Tx
  /F 4 /BS << /S /I >>
  /MK << /BC [ 0 0 0 ] >>
  /DA (/TiB0 10 Tf 0 0 1 rg)
  /DV (Enter your name)
  /V (Enter your name)
/Subtype /Widget /ANN pdfmark}\unhbox\bbox
```

Example 1.17. ■

**Example 1.18. Data Sharing/Different Appearances.** Here is an example to illustrate data sharing between text fields having a totally different appearance. Notice the two fields have different fonts, different sizes for the bounding boxes, different border styles, different background and border colors. (Oops! I forgot to change the font size.)

▶

```
\rtr\htxt{\lower4pt
\Bbox[100pt,16pt]}%
\special{ps: [ /Rect \Rect
/T (Tx18)
/FT /Tx /F 4 /BS << /S /I >>
/MK << /BC [ 0 0 0 ] >>
/DA (/TiRo 10 Tf 0 0 1 rg)
/DV (Default Inserted)
/V (Enter your name)
/Subtype /Widget /ANN pdfmark}%
\unhbox\bbox
```

▶

```
\rtr\htxt{\lower4pt
\Bbox[80pt,16pt]}%
\special{ps: [ /Rect \Rect
/T (Tx18)
/FT /Tx /F 4 /BS << /S /S >>
/MK << /BC [ 1 0 0 ]
/BG [.7529 .7529 .7529]>>
/DA (/HeOb 10 Tf 1 0 0 rg)
/DV (Default Inserted)
/V (Enter your name)
/Subtype /Widget /ANN pdfmark}%
\unhbox\bbox
```

*Example Notes:* The initial value you see in these two form fields is the value of the /V key. On reset, the value of the /DV will be shown. Here is a poor man's reset button: **Reset the Field!**

■ A “Poor Man’s Button” is one for which colored rules and a hypertext link are used to construct the button instead of a button created by a form annotation. The code for the “Poor Man’s Reset Button”:

```
\txt{\bf\hb{\,\\strut Reset the Field!\,}}\special{ps:[/Rect \Rect
/Color [0 0 1] /Border [0 0 2 PDFtoTeX] /Action << /S /ResetForm
/Fields [(Tx18)] >> /Subtype /Link /ANN pdfmark}\smash
{\pushcolor{.7529 .7529 .7529}\rlap{\vrule height\ht\bbox
depth\dp\bbox width\wd\bbox}\popcolor\unhbox\bbox} % end smash
```

■ See [Links](#) for a discussion of the meaning of these various symbols. Example 1.18. ■

**Example 1.19.** [Form Field Hierarchies](#). To quote [Carl Orthlieb](#) in his article [\[orth\]](#) “Acrobat Forms Field Naming Hints,”

Note that this use of hierarchy can be useful in a number of ways including speeding up authoring and easier manipulation of groups of fields via JavaScript. In addition, a form field hierarchy can improve performance of the forms engine if there are many fields in the form.

To create hierarchy of form fields in [Exchange](#), we create the fields and name them using a dot (or period). Thus, if we wanted to create a series of text fields organized by ‘Name’, we might name our fields:

‘Name.First’, ‘Name.Last’, ‘Name.Mi’. Using the pdfmark operator, we use this naming scheme for the value of the /T key.

Last Name:

First Name:

Middle Initial:

▶ Let’s test out the manipulation of groups by ripping off a little [JavaScript](#) from [Carl Orthlieb’s](#) article in [\[orth\]](#).

The details follow:

```
%  
% Now create the ‘Name’ text fields  
Last Name: \htxt{\lower4pt\Bbox[180pt,16pt]}\special{ps: %  
[ /Rect \Rect  
  /FT /Tx /T (Name.Last) % Fully qualified title: ‘Name.Last’  
  /F 4 /BS << /S /I >>  
  /MK << /BC [ 0 0 0 ] >>  
  /DA (/Helv 10 Tf 0 0 1 rg)  
/Subtype /Widget /ANN pdfmark}\unhbox\bbox  
  
First Name: \htxt{\lower4pt\Bbox[180pt,16pt]}\special{ps: %
```

## Section 1: Forms

```
[ /Rect \Rect
  /FT /Tx /T (Name.First) % Fully qualified title: 'Name.First'
  /F 4 /BS << /S /I>>
  /MK << /BC [ 0 0 0 ] >>
  /DA (/Helv 10 Tf 0 0 1 rg)
/Subtype /Widget /ANN pdfmark}\unhbox\bbox
```

Middle Initial: \htxt{\lower4pt\Bbox[16pt,16pt]}\special{ps: %

```
[ /Rect \Rect
  /FT /Tx /T (Name.Mi)
  /F 4 /MaxLen 1 /Q 1      % Note: One one char permitted and ...
  /BS << /S /I>>          % ...the one char is centered.
  /MK << /BC [ 0 0 0 ] >>
  /DA (/Helv 10 Tf 0 0 1 rg)
/Subtype /Widget /ANN pdfmark}\unhbox\bbox
```

%  
% Create a button to change background color

\htxt{\lower4pt\Bbox[60pt,16pt]}\special{ps:

```
[ /Rect \Rect\space/T (ChgBGJS)
  /FT /Btn /Ff \FfPushButton \FfNoExport or
  /MK << /BC [ 0 0 0 ] /BG [.7529 .7529 .7529 ] /CA (Change BG!) >>
  /BS << /S /B >> /DA (/TiB0 10 Tf 0 0 1 rg)
  /A << /S /JavaScript /JS
    (var name = this.getField("Name");\string\r % \r = cr
     color.gold = new Array("CMYK", 0, 0.10, 0.84, 0);\string\r
     name.bgColor = color.gold;) >>
```

```
/Subtype /Widget /ANN pdfmark}\unhbox\bbox
```

Example 1.19. ■

**Example 1.20.** [A Location Bar.](#)

The Pdf Primer, written by [Thomas Merz](#), is a fine survey of pdfmarks; additional comments on pdfmarks, with a T<sub>E</sub>X twist, can be found in Pdfmarks: Links and Forms by D. P. Story.

Location:

Here is the code

```
%
% Use JavaScript to transfer the location
\leavevmode\htxt{\strut\hbr{The Pdf Primer}}\special{ps:
[ /Rect \Rect /FT /Btn /Ff \FfPushButton \FfNoExport or /T (Merz)
/A << /S /URI /URI (http://www.ifconnection.de/\noexpand~tm/) >>
/AA << /E << /S /JavaScript /JS (this.getField("Locator").value =
      "http://www.ifconnection.de/\noexpand~tm/";) >>
      /X << /S /JavaScript /JS (this.getField("Locator").value="");>>
  >>
/Subtype /Widget /ANN pdfmark}\unhbox\bbox, written by ...
%
% Now construct a location bar
Location: \htxt{\lower4pt\Bbox[180pt,16pt]}\special{ps:
[ /Rect \Rect /FT /Tx /T (Locator) /MK << /BC [ 0 0 0 ] >>
  /BS << /S /S >> /DA (/Cour 8 Tf 0 g)
  /Subtype /Widget /ANN pdfmark}\unhbox\bbox
```

*Example Notes:* The /E is the “mouse enter” event key and /X is the “mouse exit” event key.

- /AA is the *additional actions* key, see [pdfs, p. 101].
- This is an interesting idea, but would use a lot of resources if you were to do it on each page. Example 1.20. ■

**Example 1.21. A Multiline Text Field.** A multiline text field is set up when you specify

```
/FT /Tx /Ff 4096
```

(see TABLE 2), or, using the T<sub>E</sub>X macros defined on page 14, you can type

```
/FT /Tx /Ff \FfMultiLine
```

► Multiline Text:

```
\htxt{\lower85pt\Bbox[180pt,96pt]}
\special{ps: [ /Rect \Rect /T (SelfDescrip)
  /FT /Tx /Ff \FfMultiLine      % /Ff 4096
  /F 4 /BS << /S /I >>         % /W 1 is the default
  /MK << /BC [ 0 0 0 ] >>      % black border, transparent background
  /DA (/TiRo 10 Tf 0 0 1 rg)
  /V (Briefly describe your qualifications.)
  /Subtype /Widget /ANN pdfmark}\unhbox\bbox
```

Example 1.21. ■

## 1.12. Enhancing your Text

According to [pdfs, p. 125], it is possible to include any valid *text-state operator* in the default appearance string, **DA**, of fields comprising of variable text. (Variable text fields include the **Text Field**, the **List Field**, the **Pop-up List** and the **Combo Box**.)

A list of valid text-state operators is on page 212 of [pdfs]. Of particular interest are the operators **Tr** (rendering mode), **Tc** (character spacing), **Tw** (word spacing), and **Tz** (horizontal spacing). The color **RG** is also useful in conjunction with **Tr**.

**Example 1.22. Changing the Rendering Mode.** The **Tr** operator is used to set the *text rendering mode*. Only three of the eight modes seem useful at this time: ‘0 Tr’ (fill text, the default); ‘1 Tr’ (stroke text); ‘2 Tr’ (stroke and fill text).

### ► Rendering Mode 1, Stroke Text.

```
[ /Rect \Rect
  /T (D) /FT /Tx /MaxLen 1
  /F 4 /BS << /S /I >> /Q 1
  /MK << /BC [ 0 .6 0 ] >>
  /DA (1 Tr /HeBo 48 Tf 1 0 0 rg) % rendering mode 1
  /DV (D)
  /Subtype /Widget /ANN pdfmark
```

**Rendering**

**Mode 1**

*Example Notes:* This example calls for *rendering mode 1* (stroke), Helvetica-Bold at 48 pt (40 pt at 1200 magnification), and a red text, ‘1 0 0 rg’. The red does not appear because we are not filling the text. Without the ‘1 Tr’, rendering mode 0 would be in effect (fill), the letter would be painted red. ■

Now let’s fill the text with rendering mode 2.

### ► Rendering Mode 2, Stroke and Fill Text.

```
[ /Rect \Rect
  /T (P) /FT /Tx /MaxLen 1
  /F 4 /BS << /S /I >> /Q 1
  /MK << /BC [ 0 .6 0 ] >>
  /DA (2 Tr /HeBo 48 Tf 1 0 0 rg) % rendering mode 2
  /DV (P)
  /Subtype /Widget /ANN pdfmark
```

## Rendering

### Mode 2

*Example Notes:* In *rendering mode 2*, ‘2 Tr’, the text is stroked (the black outline of the letter) and filled. The fill color is determined by the **rg** operator; Here, since the text is filled, the **rg** operator paints the interior red, ‘1 0 0 rg’. ■

As a final variation, let’s paint the stroked letter. To do that, the **RG** (set stroke color) operator is used. The use of **RG** is quite similar to the more familiar **rg** (set fill color) operator.

## ► Rendering Mode 2, Stroke and Fill Text with Color.

```
[ /Rect \Rect
  /T (S) /FT /Tx /MaxLen 1
  /F 4 /BS << /S /I >> /Q 1
  /MK << /BC [ 0 .6 0 ] >>
  /DA (2 Tr /HeBo 48 Tf 0 0 1 RG 1 0 0 rg)
  /DV (S)
Mode 2 /Subtype /Widget /ANN pdfmark
```

**Rendering**

The ‘2 Tr’ in the default appearance string defines *rendering mode 2*, stroking and filling the text; the insertion of ‘0 0 1 RG’, this calls for the stroked text to be colored blue. Example 1.22. ■

**Example 1.23. Changing the Word and Character Spacing.** The **Tc** and **Tw** can be used to obtain an interesting spread out look.

```
\htxt{\Bbox[296pt,20pt]}\special{ps:
[ /Rect \Rect /T (YourName)
 /FT /Tx /F 4 /BS << /S /I >>
 /MK << /BC [ 0 .6 0 ] >>
 /DA (6 Tw 6 Tc /HeBo 15 Tf 1 0 0 rg)
 /DV (Enter your name here)
/Subtype /Widget /ANN pdfmark}\unhbox\bbox
```

*Example Notes:*The **Tc** operator takes a single argument, the argument is the amount of space after a character, while **Tw** sets the word spacing with its argument. Here the word spacing, ‘6 Tw’, is set to 6 text-space units, as is the character spacing, ‘6 Tc’. Example 1.23. ■

**Example 1.24. Changing the Horizontal Scale.** The operator **Tz** sets the horizontal spacing, its argument is a number expressed as a percent of the normal scaling. This operator can be used to “compress” the text.

```
\htxt{\Bbox[150pt,20pt]}\special{ps:
[ /Rect \Rect
/T (YourName) /FT /Tx
/F 4 /BS << /S /I >>
/MK << /BC [ 0 .6 0 ] >>
/DA (50 Tz /HeBo 15 Tf 1 0 0 rg)
/DV (Enter your name here)
/Subtype /Widget /ANN pdfmark}\unhbox\bbox
```

The horizontal scaling parameter is set to 50, ‘50 Tz’; thus, the scaling is 50% of normal. Example 1.24. ■

**Example 1.25.** [Line Breaking with a Horizontal Scale Change.](#)

There is a slight problem with changing the horizontal scaling using the **Tz** operator within variable text fields.

**Width 150 pt**

**Width 300 pt**

When you create a text field 150 points wide, for example, and set horizontal scaling at 50%, the line breaking algorithm of the Acrobat Reader/Exchange breaks the text as if *normal scaling were in effect*. The result is the **Width 150 pt** text box seen above. Note that the

text extends across only half the width of the box; line breaks are based on normal scaling, but the text is half the normal width.

This problem can be circumvented by creating a text box 300 pt wide. The Reader/Exchange will break lines for that width, and the text will be only 150 pt wide. See the text field labeled **Width 300 pt**.

So how come the two boxes appear to be the same width? Read the code below.

```
%
% Define \InputMessage macro for convenienc. Use JavaScript to
% transfer the data to the other two text fields (both have
% the same /Title, so text gets transferred to both). If you
% own text in the box, you must ‘‘commit’’ your choice by clicking
% the mouse somewhere outside the box.
\def\InputMessage{\htxt{\Bbox[300pt,70pt]}\special{ps: %
[ /Rect \Rect /T (InputMessage)
 /FT /Tx /Ff \FfMultiLine
 /F 4 /BS << /S /I >>
 /MK << /BC [ 0 .6 0 ] >>
 /DA (/HeBo 14 Tf 1 0 0 rg)
 /DV {datastring} % See Creating a Stream: An Example
 /AA << /V << /S /JavaScript
 /JS (this.getField("OutScaleMess").value=event.value;) >> >>
 /Subtype /Widget /ANN pdfmark}\unhbox\bbox}
```

## Section 1: Forms

```
% \OutMessi macro, the one labeled Width 150 pt
\def\OutMessi{\htxt{\Bbox[150pt,70pt]}\special{ps: % 150 points wide
[ /Rect \Rect /T (OutScaleMess)
 /FT /Tx /Ff \FfMultiLine\space\FfReadOnly\space or
 /F 4 /BS << /S /I >> /MK << /BC [ 0 .6 0 ] >>
 /DA (50 Tz /HeBo 14 Tf 0 0 1 rg) % 50% Horizontal scaling
 /DV {datastring} % See Creating a Stream: An Example
/Subtype /Widget /ANN pdfmark}\unhbox\bbox}
```

```
% \OutMessii macro, the one labeled Width 300 pt
% Two super-imposed text fields.
\def\OutMessii{%
%
% The bottom one is 300 pt wide; this text field has an invisible
% border.
\htxt{\Bbox[300pt,70pt]}\special{ps: % 300 points wide
[ /Rect \Rect /T (OutScaleMess)
 /FT /Tx /Ff \FfMultiLine\space\FfReadOnly\space or
 /F 4 /BS << /S /I >> % /MK << /BC [ 0 .6 0 ] >> % commented out
 /DA (50 Tz /HeBo 14 Tf 0 0 1 rg) % 50% Horizontal scaling
 /DV {datastring} % See Creating a Stream: An Example
/Subtype /Widget /ANN pdfmark}% % don't \unhbox\bbox here!
%
% The top text field is only 150 pt wide. It's only role is to draw a
% border around itself.
\htxt{\Bbox[150pt,70pt]}\special{ps: % 150 points wide
```

```
[ /Rect \Rect /T (Dummy)
  /FT /Tx /Ff \FfReadOnly
  /F 4 /BS << /S /I >> /MK << /BC [ 0 .6 0 ] >>
/Subtype /Widget /ANN pdfmark}\unhbox\bbox} % now \unhbox\bbox
```

Notice that we didn't “\unhbox\bbox” the bottom field. To do so, would take up 300 points of horizontal space—this we don't want to do. This field must not take up any space.

```
%
% Now, for people who use TEX, we lay it all out in a nice format.
{\leavevmode\parskip0pt\offinterlineskip
\InputMessage\par
\line{\OutMessi\OutMessii}\par}
```

**Concluding Remarks.** Additional details can be found in the article “[Enhancing Text in Text Fields using Pdfmarks](#)”, written for the [PDFZone.COM](#) (<http://www.pdfzone.com/>). See also the crude [Valentine's Day Card](#) that demonstrates the techniques introduced in this section, [Section 1.12](#). ■

### 1.13. Cos Objects

Cos Objects are fundamental building blocks of PDF files. A PostScript language program, and T<sub>E</sub>X through its PostScript verbatim `\specials`, can create composite Cos objects (in the form of arrays,

dictionaries, and streams), name them, and create indirect references to them. The `/OBJ pdfmark` key-value pair is used to create a Cos object.

- **Defining Cos Objects**

The syntax for specifying a Cos object is

```
[ /_objdef {OBJNAME} /type <name> /OBJ pdfmark
```

The possible choices for `<name>` are `/array`, `/dict`, and `/stream`. Of course, the T<sub>E</sub>X version of this comes in a ‘special’ wrapper

```
\special{ps: ... ..}.
```

Throughout this article, we have created a number of Cos objects:

1. **Composite Array Cos Objects:** See [SECTION 1.2](#) on the [AcroForm Dictionary](#); [EXAMPLE 1.8](#), the [Tic-Tac-Toe game](#); and in [EXAMPLE 1.11](#) to construct [Radio Buttons](#).
2. **Composite Dictionary Cos Objects:** See [SECTION 1.2](#) once again.
3. **Composite Stream Cos Objects:** We have not seen this type yet; however, see [Creating a Stream: An Example](#) below.

## • Predefined Cos Objects

[Acrobat](#) predefines certain Cos objects that the author of a PDF document can reference and PUT information into. The list, reference [\[pdfm, p. 22\]](#), follows:

- `{Catalog}`—the Catalog dictionary;
- `{DocInfo}`—the Info dictionary;
- `{PageN}`—the dictionary for page N;
- `{ThisPage}`—the dictionary for the *current page*;
- `{PrevPage}`—the dictionary for the page *before* the current one;
- `{NextPage}`—the dictionary for the page *after* the current one.

There are many uses of these predefined dictionaries; see [\[primer\]](#), [The Pdf Primer](#) by [Thomas Merz](#), for a complete discussion with examples.

The page dictionaries were used in many examples in the article on [Links](#) and were used to specify the target page of a hypertext jump. (These are not the only uses of the page dictionaries, by the way.)

Information, in the form of key-value pairs, may be inserted into these dictionaries using the `/PUT` key. Here is an example:

```
\special{ps:  
[ {Catalog} <<  
    /ViewerPreferences <<
```

```
    /CenterWindow true
    /HideWindowUI true
    /HideToolbar true
    /HideMenubar true
    /FitWindow true
>> >> /PUT pdfmark}
```

You'll have to try this one out to see what it does.

- **Indirect Naming within pdfmarks**

In addition to the `/_objdef ... /OBJ pdfmark` syntax for creating a Cos object, the `/_objdef {OBJNAME}` key-value pair can be included in the following pdfmark commands: the annotation, `/ANN`; encapsulated graphics, `/BP`; destination, `/DEST`; the (obsolete) link, `/LNK`; and embedded postscript, `/PS`. Once an `/_objdef {OBJNAME}` has been included in an annotation, say, that annotation can be referred to by the name, `{OBJNAME}`, in other PDF objects.

This referencing mechanism manifested itself in this article in [Radio Button Fields](#). The parent and kids of a [Radio Field](#) are named using the syntax `/_objdef {OBJNAME}` and later references made within the annotations themselves, see [EXAMPLE 1.9](#), [EXAMPLE 1.10](#), or [EXAMPLE 1.11](#).

- **Placing Information in Cos Objects**

To place information in composite objects created by OBJ, use PUT or PUTINTERVAL. Additionally, CLOSE is a stream that has been opened with PUT.

We have seen instances of PUT and PUTINTERVAL already:

1. PUT: In SECTION 1.2, we PUT information into a dictionary. In an example above, we PUT information into the {Catalog} dictionary. **Creating a Stream: An Example**, listed below, illustrates how to insert a string into a stream.
2. PUTINTERVAL: This is used to put information into an array. See EXAMPLE 1.8 on Tic-Tac-Toe. In that example, we places the titles of each of the nine squares of the board game into an array structure for convenient reference by some of the annotations.
3. CLOSE: This is used to close off a steam. See **Creating a Stream: An Example** below.

- **Creating a Stream: An Example**

Text strings can be inserted into a PDF file in the form of ‘streams’. The source of the stream data may either be a *string* or a *file*. Stream data is always compressed using either LZW or ZIP, depending on the compatibility level set for the Distiller program.

Let's create a stream of data:

```
\special{ps:  
[ /_objdef {datastring} /type /stream /OBJ pdfmark  
[ {datastring} (Enter your autobiography in this space. Don't worry  
about privacy issues, the data are not saved.) /PUT pdfmark}
```

We can still add to the data to {datastring} since we have not closed the stream yet.

```
\special{ps:  
[ {datastring} (\string\r\string\r Or is it?) /PUT pdfmark  
[ {datastring} /CLOSE pdfmark}
```

Now we have closed off the stream. We can't write to this steam again.

Now, let's move on to the problem of using this stream data. We'll illustrate with a multiline text field.

```

\hxtt{\Bbox[144pt,74pt]}\special{ps:
[ /Rect \Rect /FT /Tx /Ff \FfMultiLine /T (mytext)
  /BS << /S /I >> /DA (/Helv 10 Tf 1 0 0 rgb)
  /MK << /BC [ 0 0 0 ] >>
  /V {datastring}
/Subtype /Widget /ANN pdfmark}\unhbox\bbox


```

## 1.14. Defining/Using XObjects

Beginning with [Acrobat Distiller 3.0](#), a PostScript language program could specify a set of graphical operations to be encapsulated and treated as a single object.

The `pdfmark` operators using the `BP` (Begin Picture) and `EP` (End Picture) enclose the graphic operations.

- **Commentary on an Encapsulated Graphic**

Study the sample listings at the end of this file. In particular, let's look at the e- button face, `{eCalc}`, reproduced, in part below.

```

[/BBox [0 0 400 100] /_objdef {eCalc} /BP pdfmark
0 0 .5 setrgbcolor 0 0 400 100 rectfill 1 setgray 2 2 moveto
:      :      :      :      :      :      :
1 0 0 setrgbcolor 15 22.5 moveto (e-Calculus) show

```

```
[/EP pdfmark
```

You will notice that the graphical operations take place within a bounding box, `/BBox [ 0 0 400 100 ]`, in this case.

After the bounding box comes an object definition followed by the `/BP` operator: `/_objdef {eCalc} /BP pdfmark`. This not only names the object, but marks the beginning of the picture.

After the actual PostScript code defining the graphic element comes the `[ /EP pdfmark` (End Picture).

Now for driver dependent comments: [DVIPSONE](#) versus [DVIPS](#).

For [DVIPSONE](#), the `postscript` special is used. This special does an automatic `gsave/grestore`. To illustrate, define `{eCalc}` as follows:

```
\special{postscript % Normal Appearance
[/BBox [0 0 400 100] /_objdef {eCalc} /BP pdfmark
0 0 .5 setrgbcolor 0 0 400 100 rectfill 1 setgray 2 2 moveto
:      :      :      :      :      :      :
1 0 0 setrgbcolor 15 22.5 moveto (e-Calculus) show
[/EP pdfmark}%
```

For [DVIPS](#), we perform the above actions manually as follows:

```

\special{ps: gsave % Normal Appearance
[/BBox [0 0 400 100] /_objdef {eCalc} /BP pdfmark
0 0 .5 setrgbcolor 0 0 400 100 rectfill 1 setgray 2 2 moveto
:      :      :      :      :      :      :
1 0 0 setrgbcolor 15 22.5 moveto (e-Calculus) show
[/EP pdfmark grestore}%

```

One application of encapsulated graphics is that they can be used as appearance faces of form annotations. (Recall, [EXAMPLE 1.3.](#)) This is a major application.

In the next two subsections, examples specialized to [DVIPSONE](#) and [DVIPS](#). The techniques for the two drivers are slightly different.

- **Using DVIPSONE**

Aside from this, the graphics can simply be shown. The `pdfmark` operator `SP` (Show Picture) inserts the picture into the document. For example,



is produced by the code

```

\special{postscript .2 .2 scale [ {eCalc} /SP pdfmark}

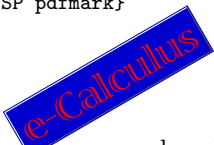
```

The bounding box specifies a 400pt by 100pt rectangle. Cutting it down by 20% gives 80pt by 20pt button face.

The graphic can be manipulated in various ways.

```
\special{postscript 80 0 rmoveto currentpoint translate
      -.2 .2 scale [ {eCalc} /SP pdfmark}
```

produces



Now try a 30° rotation. The graphic ... was produced by the code

```
\special{postscript 20 60 cos mul 0 rmoveto currentpoint translate
      .2 .2 scale 30 rotate [ {eCalc} /SP pdfmark}
```

Let's rescale the button face to 15%, rotate it 90° clockwise, and attach an hypertext link. The code is as follows:

```
\special{ps: gsave currentpoint translate -90 rotate}%
\htxt{\Bbox[60pt,15pt]}\special{ps: [ /Rect \Rect /Action /Page 1
/Subtype /Link /ANN pdfmark]\rlap{\unhbox\bbox}\special{postscript
0 15 rmoveto currentpoint translate .15 .15 scale [ {eCalc}
/SP pdfmark]\special{ps: grestore}}
```

The dimensions of the bounding box are 60pt by 15pt, this was obtained by taking 15% of the original dimensions of the bounding box of `{eCalc}`, see [below](#), which was 400pt by 100pt.

It should be mentioned that the reason `\special{postscript ...}` is used in the above `/SP` examples is because this particular special, in addition to enclosing everything in `gsave/grestore` also moves the origin to the `currentpoint`, i.e., a `currentpoint translate` is performed.

- **Using DVIPS**

The effects of the previous subsection can be duplicated using [DVIPS](#), but with quite a bit more work, naturally.

To begin with, it is useful to define a PostScript procedure:

```
%  
% Change scale to dvips's dots, translate origin to current point.  
\special{!userdict begin /cstr {currentpoint translate  
  1 PDFtoDvips DVImag mul -1 PDFtoDvips DVImag mul scale}def end}
```

In the examples that follow, [DVIPSONE](#) was used to generate the PostScript code for the pictures. The accompanying listed code is taken

from a file compiled using [DVIPS](#). I have verified on my [MikTeX System](#) that these examples behave as advertised and are identical to their [DVIPSONE](#) counterparts.

We now continue to paraphrase the dialog and examples of the previous section.

Aside from this, the graphics can simply be shown. The `pdfmark` operator `SP` (Show Picture) inserts the picture into the document. For example,



is produced by the code

```
\special{ps:gsave cstr .2 .2 scale [{eCalc} /SP pdfmark grestore}
```

The bounding box specifies a 400pt by 100pt rectangle. Cutting it down by 20% gives 80pt by 20pt button face.

The graphic can be manipulated in various ways.

```
\special{ps:gsave cstr 80 0 rmoveto currentpoint translate -.2  
      .2 scale [ {eCalc} /SP pdfmark grestore}
```

produces

Now try a 30° rotation. The graphic ... was produced by the code

```
\special{ps: gsave cstr 20 60 cos mul 0 rmoveto currentpoint
  translate .2 .2 scale 30 rotate
  [ {eCalc} /SP pdfmark grestore}
```


Let's rescale the button face to 15%, rotate it 90° clockwise, and attach an hypertext link. The code is as follows:

```
\special{ps:gsave currentpoint currentpoint translate 90 rotate
neg exch neg exch translate}\htxt{\Bbox[60pt,15pt]}\special{ps:
[ /Rect \Rect /Page 1 /Subtype /Link /ANN pdfmark]%
\rlap{\unhbox\bbox}\special{ps:grestore}\special{ps:gsave cstr
.15 .15 scale -90 rotate [ {eCalc} /SP pdfmark grestore}
```

The dimensions of the bounding box are 60pt by 15pt, this was obtained by taking 15% of the original dimensions of the bounding box of {eCalc}, see [below](#), which was 400pt by 100pt.

---

This pretty much finishes the current discussion of links and forms. In the process of writing these articles, I certainly learned quite a lot, and my knowledge of these topics is much better organized. I had originally planned to do more with [CGI](#) and [JavaScript](#), but I think, this is enough for now.

Contact me through e-mail, [dpstory@uakron.edu](mailto:dpstory@uakron.edu), if you have any comments or have found errors/typos, or have any suggestions as to future topics. I am particularly interested in interesting examples and techniques that I have not covered or thought of. 



xobject  
listings



biblio



first page



links



homepage

- **XObjects Listings**

The code for these XObjects was input at the beginning of the file, right after the [AcroForm Dictionary](#).

In the listings below, users of [DVIPS](#) should use

```
\special{ps:gsave ... .. grestore}
```

when creating these encapsulated graphics, instead of

```
\special{postscript ... .. },
```

Begin Listing:

```
%  
% Cross and Check  
%
```

## Section 1: Forms

```
\special{postscript % Cross char from ZapfDingbats  
[/BBox [0 0 100 100] /_objdef {xCross} /BP pdfmark  
0 0 1 setrgbcolor /ZapfDingbats 100 selectfont 9.7 7.3 moveto (8) show  
[/EP pdfmark}%
```

```
\special{postscript % Check char from ZapfDingbats  
[/BBox [0 0 100 100] /_objdef {xCheck} /BP pdfmark  
0 0 1 setrgbcolor /ZapfDingbats 100 selectfont 9.7 7.3 moveto (4) show  
[/EP pdfmark}%
```

## Section 1: Forms

```
%  
% e-Calculus Button /N, /R, and /D respectively  
%  
\special{postscript % Normal Appearance  
[/BBox [0 0 400 100] /_objdef {eCalc} /BP pdfmark  
0 0 .5 setrgbcolor 0 0 400 100 rectfill 1 setgray 2 2 moveto  
2 98 lineto 398 98 lineto 396 96 lineto 4 96 lineto 4 4 lineto fill  
0.34 setgray 398 98 moveto 398 2 lineto 2 2 lineto 4 4 lineto  
396 4 lineto 396 96 lineto fill /Viva-Regular 72 selectfont  
1 0 0 setrgbcolor 15 22.5 moveto (e-Calculus) show  
[/EP pdfmark}]%
```

```
\special{postscript % Rollover Appearance  
[/BBox [0 0 400 100] /_objdef {reCalc} /BP pdfmark  
0 0 1 setrgbcolor 0 0 400 100 rectfill 1 setgray 2 2 moveto  
2 98 lineto 398 98 lineto 396 96 lineto 4 96 lineto 4 4 lineto fill  
0.34 setgray 398 98 moveto 398 2 lineto 2 2 lineto 4 4 lineto  
396 4 lineto 396 96 lineto fill /Viva-Regular 72 selectfont  
1 0 0 setrgbcolor 15 22.5 moveto (e-Calculus) show  
[/EP pdfmark}]%
```

## Section 1: Forms

```
\special{postscript % Pushed Appearance
[/BBox [0 0 400 100] /_objdef {peCalc} /BP pdfmark
0 0 1 setrgbcolor 0 0 400 100 rectfill 0.34 setgray 2 2 moveto
2 98 lineto 398 98 lineto 396 96 lineto 4 96 lineto 4 4 lineto fill
1 setgray 398 98 moveto 398 2 lineto 2 2 lineto 4 4 lineto
396 4 lineto 396 96 lineto fill /Viva-Regular 72 selectfont
1 0 0 setrgbcolor 15 22.5 moveto (e-Calculus) show
[/EP pdfmark}%
```

%

% The X and O for Tic-Tac-Toe

%

```
\special{postscript % The Tic-Tac-Toe "0"
[/BBox [0 0 100 100] /_objdef {x0} /BP pdfmark
.7529 setgray 0 0 100 100 rectfill 1 setgray
2 2 moveto 2 98 lineto 98 98 lineto 96 96 lineto 4 96 lineto
4 4 lineto fill 0.34 setgray 98 98 moveto 98 2 lineto 2 2 lineto
4 4 lineto 96 4 lineto 96 96 lineto fill 0 setgray 22.5 22.5 moveto
1 0 0 setrgbcolor /Helvetica 72 selectfont (0) show
[/EP pdfmark}%
```

```
\special{postscript % The Tic-Tac-Toe "X"  
[/BBox [0 0 100 100] /_objdef {xX} /BP pdfmark  
.7529 setgray 0 0 100 100 rectfill 1 setgray  
2 2 moveto 2 98 lineto 98 98 lineto 96 96 lineto 4 96 lineto  
4 4 lineto fill 0.34 setgray 98 98 moveto 98 2 lineto 2 2 lineto  
4 4 lineto 96 4 lineto 96 96 lineto fill 0 setgray 27 22.5 moveto  
0 0 1 setrgbcolor /Helvetica 72 selectfont (X) show  
[/EP pdfmark}%
```

## Bibliography

- [pdfm] pdfmark Reference Manual, Technical Note #5150, by Tim Bienz and Gary Staas, Adobe Systems Incorporated, October 24, 1996.
- [pdfs] Portable Document Format Reference Manual, Version 1.2, by Tim Bienz, Richard Cohn, and James R. Meehan, Adobe Systems Incorporated, November 12, 1996.
- [primer] The Pdfmark Primer, Thomas Merz, freely available over the Web, <http://www.ifconnection.de/~tm/>, 1998
- [orth] Acrobat Forms Field Naming Hints, Carl Orthlieb, PurePDF, <http://www.purepdf.com/pp2-23/>, 1998.
- [tex] *The T<sub>E</sub>Xbook*, Donald Knuth, Addison-Wesley Publishing Co., June 1992.
- [webpub] *Web Publishing with Acrobat/PDF*, Thomas Merz, Springer-Verlag, 1998