Each annotation is created with 10 text info variables, numbered 0 through 9. Each text info variable is initialized to the default font, font size and style for the graph or page layout hosting the annotation. The X and Y positions are initially undefined.

For background information and an example, see **Text Info Variables** on page III-51.

Here are the supported text info variable escape codes. <digit> means one of 0, 1, 2, ... 9:

| | |
|---|---|
| \[<digit> | Saves the font, size, style and current X and Y positions in the text info variable. |
| \]<digit> | Restores all but the X and Y positions from the text info variable. |
| \X<digit> | Restores the X position from the text info variable. |
| \Y<digit> | Restores the Y position from the text info variable. |
| \F]<digit> | Restores the font from the text info variable. |
| \Z]<digit> | Restores the font size from the text info variable. |
| \f]<digit> | Restores the style from the text info variable. |

Text info variable 0 has a special property. It defines the "main" font size which is restored using the \M escape sequence. \M also sets the Y offset for the baseline to zero. No other text info variable 0 settings (font, style, or offsets) are set by \M.

## Dynamic Text Escape Codes

You can enter the dynamic text escape sequence which inserts dynamically evaluated text into any kind of annotation using the escape code sequence:

Dynamic text allows you to create automatically updated annotations. The syntax for inserting dynamic text is:

```
\{dynamicText}
```

*dynamicText* may contain numeric and string expressions which reference global variables and waves and which invoke user-defined functions. Igor automatically reevaluates *dynamicText* when a global variable or a wave referenced directly in *dynamicText* or indirectly through a function call changes. It also reevaluates *dynamicText* whenever the annotation is updated.

**Note**: The use of dynamic text creates dependencies that can be fragile and confusing. In most cases, it is better to generate static text programmatically, as described in **Generating Text Programmatically** on page III-53.

**Note**: You can not reference local variables in *dynamicText* because local variables exist only while a procedure is executing and Igor must reevaluate it at other times.

The numeric and string expressions are evaluated in the context of the root data folder. Use the full data folder path of any non-root objects in the expressions.

*dynamicText* can take two forms, an easy form for a single numeric expression and a more complex form that provides precise control over the formatting of the result.

The easy form is:

```
\{ numeric-expression }
```

This evaluates the expression and prints with generic ("%g") formatting. For example:

```
TextBox "Two times PI is \\{2*PI}"
```

creates a textbox with this text:

```
Two times PI is 6.28319
```