Static Text

Static text controls must be of type "Static", and can be optionally linked to STR# or TEXT resources. If not linked to a resource, then the control's title is displayed as its content, along with whatever frame and background you have chosen.

Static Boxes & Lines

Although static text controls are the simplest types of controls, they still support a full range of text styles, colors, indents, frames, etc. One important variation is when the static text control is not given a title. In this case it can be used to display a box (set a non-zero frame size) that encloses other controls. If the box is transparent (does not have a solid body), then ViewIt allows mouse clicks to pass through the content area of the box, meaning that you don't need to worry about whether the controls bounded by the box are before or after it in the control list. The "Box (Transparent)" control in the Import menu is an example of a transparent static box.

Another important variation of untitled static text controls is their use to draw lines. In this case, the content of the control is used as the line, and the frame is zero (for a < 3 pixel-wide line you'll also need to set the indent to 1 since ViewIt does not allow controls to be narrower than 3 pixels). To draw a patterned or colored line, link the control to a pattern resource and/or color its content or body. The "Line w/ Pattern (PAT#)" control uses this trick to draw a gray line.

STR# Strings

Static text controls can be linked to STR# resources to display the string specified by the control's value. A control linked to string list STR# 1001 with Min = Max = Value = 5, for example, would display the fifth string in STR# 1001.

Two Exceptions: (1) If the first 4 characters of the string are "PICT", "ICON", "SICN", "cicn", "PAT ", "PAT#", "CURS", "acur", or "clut", then the resource is displayed instead of the string (see "Icons, Picts, ..." topic). (2) If control's Min = 0, then the string list is used as parameter text (described below).

STR# Parameters

If the control's Min is set to zero, then STR# lists are used as "parameter" strings. In this case BaseCt replaces all "^n" substrings in the control's title with the corresponding strings from the STR# resource. This is similar to, but more powerful than, the Dialog Manager's support for "parameter text".

For example, if the control's title is "^2 loves ^4", then BaseCt creates a copy of this title and replaces "^2" and "^4" with the second and fourth strings from the linked STR# each time it redraws the control. A programmer can thus change the appearance of the item at any time by either changing the strings in the STR# or by calling "SetCTitle" to reset the control title to text that references different strings (such as changing "^2 loves ^4" to "^3 dislikes ^5").

Note that changes made to the STR# list will not be seen until the control is redrawn. If necessary, use DrwCtl to redraw the control. For example,

```
FaceIt(0,GetCtl,0,0,1,3);
FaceIt(0,DrwCtl,ord(cControl),0,0,0);
```

would redraw the third control in the first view of the front window. If using DrwCtl to draw a control, the control should have a solid body so that old text is completely erased before new text is drawn.

TEXT Block

A third variation of static text occurs when linked to a TEXT resource. In this case the current content of the TEXT resource is displayed. If the TEXT resource in memory is changed by the program, then any call that causes the control to be redrawn will update the displayed text.

The following code, for example, will display the text in a 1000 character C-type string in a static text control (view 2, control 5) that is linked to a TEXT resource ("fRec." dropped for simplicity):

```
FaceIt(0,GetCtl,0,0,2,5);
SetHandleSize(cResHdl,1000L);
BlockMove(&myString[0],*cResHdl,1000L);
FaceIt(0,DrwCtl,(long)cControl,2,0,0);
```

where the TEXT resource must not be purgeable, the "2" in DrwCtl redraws only the content area of the static control, and the control must have a solid body.

Options

The following bit values can be added to VarCode to set various options:

32: A static text control normally draws text at the top of the control using the current text justification. If 32 is added to the VarCode, then the first text line is centered vertically in the control.

16384: If a static text control is not linked to a TEXT resource, then it is normally limited to 255 characters (the maximum size of its control title). If 16384 is added to VarCode, then a separate handle is used to store the control's text and the limit of 255 characters does not apply. This handle can be found in cHiData after calling GetCtl, and could be used in code like that shown above for moving text into a TEXT handle (replace cResHdl with cHiData). You can also use data linking to get and set such text.

Data Linking

Static text that is not linked to a TEXT resource has a "value" equal to the control's title, and can be linked to any type of program variable (see "Data Links" in the ViewIt Guide for more info on data linking). This makes it easy, for example, to have static text items in ViewIt windows directly display integer, real, or string program variables.

Static text controls linked to TEXT resources base data linking on the linked TEXT resource in memory. The only advantage to data linking in this case is that it can be used to get ViewIt to do the SetHandleSize, BlockMove, and DrwCtl that would otherwise need to be made to directly update the resource in memory and redraw the control.

Static text that is not linked to a TEXT resource but has 16384 added to VarCode bases data linking on a private text handle in memory, and can be data linked in the same way as controls linked to TEXT resources.