# Building an Operator Interface with HP VEE

# Notice

**Warranty Information**

**Restricted Rights Legend**

**Printing History**

Edition 1 - January 1995

# Conventions Used in this Manual

This manual uses the following typographical conventions:

| Example | Represents |
|---|---|
| *HP VEE Reference* | Italicized words are used for book titles and for emphasis. |
| `File` | Computer font represents text you will see on the screen, including menu names, features, buttons, or text you enter. |
| `dir` *filename* | In this context, the word in computer font represents text you type exactly as shown, and the italicized word represents an argument that you must replace with an actual value. |
| `File` $\Longrightarrow$ `Open` | The "$\Longrightarrow$" is used in a shorthand notation to show the location of HP VEE features in the menu. For example, "`File` $\Longrightarrow$ `Open`" means to select the `File` menu and then select `Open`. |
| `Zoom Out \| In 2x \| In 5x` | Choices in computer font, separated with a bar (\|), indicate that you should choose one of the options. |
| $\boxed{\text{Return}}$ | The keycap font graphically represents a key on the keyboard. |
| Press $\boxed{\text{Ctrl}}$+$\boxed{\text{O}}$ | Represents a combination of keys on the keyboard that you should press at the same time. |
| **Dialog Box** | Bold font indicates the first instance of a word defined in the glossary. |

# Contents

**1. Building an Operator Interface with Panels**

What are Panels? . . . . . . . . . . . . . . . . . . . 1-2
Building a Main Panel . . . . . . . . . . . . . . . . . 1-4
  To Add Objects to a Panel . . . . . . . . . . . . . . 1-4
  To Delete Objects from a Panel . . . . . . . . . . . 1-5
Changing the Appearance of a Panel . . . . . . . . . . 1-6
  To Change an Object's Border Appearance . . . . . . 1-7
  To Label Areas . . . . . . . . . . . . . . . . . . . 1-9
  To Align Objects . . . . . . . . . . . . . . . . . . 1-9
  To Change the Grid Size . . . . . . . . . . . . . . 1-10
  To Match Object Sizes . . . . . . . . . . . . . . . 1-11
  To Add a Picture to the Panel . . . . . . . . . . . 1-11

**2. Building UserObject and UserFunction Panels**

What is a UserObject Panel? . . . . . . . . . . . . . 2-3
  To Build a UserObject Panel . . . . . . . . . . . . 2-4
  To Build a Pop-Up Panel . . . . . . . . . . . . . . 2-5
  To Keep a Pop-Up Panel Displayed . . . . . . . . . . 2-6
  To Change the Size of a Pop-Up Panel . . . . . . . . 2-7
  To Change the Location of a Pop-Up Panel . . . . . . 2-7
  To Add a UserObject Panel to the Main Panel . . . . 2-8
  To Add Objects to a Nested UserObject Panel . . . . 2-9
  To Delete Objects from a Nested UserObject Panel . . 2-10
  To Delete a UserObject Panel from the Main Panel . . 2-11
What is a UserFunction Panel? . . . . . . . . . . . . 2-12
  To Build a UserFunction Panel . . . . . . . . . . . 2-12
  To Build a UserFunction Pop-Up Panel . . . . . . . . 2-13
  To Change the Size of a UserFunction Pop-Up Panel . . 2-14
  To Change the Location of a UserFunction Pop-Up Panel 2-14

**3. Using the Keyboard to Navigate a Panel View**
Making Your Panel Keyboard Navigable . . . . . . . . . 3-3
To Give Focus to an Object or Panel . . . . . . . . . . 3-4
To Assign an `OK` Button to the Enter or Esc Key . . . . 3-5
To Assign a Button to a Function Key . . . . . . . . . 3-6
Panel View Navigation Techniques . . . . . . . . . . . . 3-8
Panel View Selection Techniques . . . . . . . . . . . . 3-11

**4. Guidelines for a Good Interface**
Most Important . . . . . . . . . . . . . . . . . . . . . 4-3
Using Colors . . . . . . . . . . . . . . . . . . . . . . 4-4
How to Pick Colors . . . . . . . . . . . . . . . . . 4-6
Using Fonts . . . . . . . . . . . . . . . . . . . . . . 4-8
Layout Tips . . . . . . . . . . . . . . . . . . . . . . 4-11
To Call Attention to an Object . . . . . . . . . . . 4-11

**5. Advanced Topics**
About Built-In Dialog Box Objects . . . . . . . . . . . 5-3
To Output a Value on Timeout . . . . . . . . . . . . 5-6
Building a Custom Dialog Box . . . . . . . . . . . . . 5-8
To Build a Basic Dialog Box . . . . . . . . . . . . 5-8
To Set a Timeout . . . . . . . . . . . . . . . . . . 5-10
To Check for Valid Data . . . . . . . . . . . . . . . 5-13
To Build a Self-Modifying Panel . . . . . . . . . . . 5-14
Building a Status Panel . . . . . . . . . . . . . . . . 5-16
To Build a Status Panel . . . . . . . . . . . . . . . 5-17
To Show a Status Panel . . . . . . . . . . . . . . . 5-17
To Hide a Status Panel . . . . . . . . . . . . . . . 5-18
To Update a Status Panel . . . . . . . . . . . . . . 5-18
A Status Panel Example . . . . . . . . . . . . . . . 5-19
To Animate Pictures . . . . . . . . . . . . . . . . . 5-22
Using Multiple Images . . . . . . . . . . . . . . 5-24

**6. Preparing a Program for Distribution to Others**
Building a Panel to Run on Multiple Platforms . . . . . 6-3
To Use Operating System-Dependent Features . . . . 6-3
General Tips . . . . . . . . . . . . . . . . . . . . . 6-4
To Choose Fonts for Multiple Platforms . . . . . . . 6-5
Font Type . . . . . . . . . . . . . . . . . . . . 6-5
Font Size . . . . . . . . . . . . . . . . . . . . 6-5
Font Style . . . . . . . . . . . . . . . . . . . . 6-6

When Transferring a Program Between UNIX and Windows . . . . . . . . . . . . . . . . . . . 6-6
When Transferring a Program Between Windows Systems 6-7
When Transferring a Program Between UNIX Systems 6-7
To Choose Colors for Multiple Platforms . . . . . . . 6-8
When Transferring a Program to a VGA Windows System . . . . . . . . . . . . . . . . . . . . 6-8
When Transferring a Program Between UNIX Systems 6-8
Building a Panel With Non-English Text . . . . . . . . 6-10
Preventing Program Changes . . . . . . . . . . . . . 6-11
To Prevent Program Changes . . . . . . . . . . . . 6-13
To Secure a Program . . . . . . . . . . . . . . . . 6-14
To Re-Secure a Program . . . . . . . . . . . . . . 6-15
To Secure a UserObject . . . . . . . . . . . . . . 6-15
Creating a Distribution Package . . . . . . . . . . . . 6-17
To Include All Needed Files . . . . . . . . . . . . . 6-17
To Create an Application Icon in Windows . . . . . . 6-18
To Create an Application Icon in VUE . . . . . . . . 6-19

**A. RGB Values for HP VEE Colors**

**Index**

# Figures

1-1. A Detail View . . . . . . . . . . . . . . . . . . . . . . . 1-2
1-2. The Corresponding Panel View . . . . . . . . . . . . . . . 1-3
1-3. Border Styles . . . . . . . . . . . . . . . . . . . . . . . 1-8
1-4. A Border Width Example . . . . . . . . . . . . . . . . . . 1-8
1-5. A `Picture` Object With Different Display Modes . . . . . . 1-13
2-1. Three Ways to Display a UserObject Panel . . . . . . . . . 2-4
3-1. A `Slider` on a Panel . . . . . . . . . . . . . . . . . . 3-4
3-2. A Simple Tabbing Order Example . . . . . . . . . . . . . 3-9
3-3. A Complex Tabbing Order Example . . . . . . . . . . . . 3-9
4-1. A Color Wheel . . . . . . . . . . . . . . . . . . . . . . 4-5
4-2. Displaying Different Color Text . . . . . . . . . . . . . . 4-7
4-3. Font Sizes . . . . . . . . . . . . . . . . . . . . . . . . 4-9
4-4. Serif and Sans Serif Fonts . . . . . . . . . . . . . . . . 4-9
5-1. A Message Box . . . . . . . . . . . . . . . . . . . . . . 5-3
5-2. Message Box Properties . . . . . . . . . . . . . . . . . . 5-5
5-3. Outputting a Default Value on Timeout . . . . . . . . . . 5-7
5-4. A Basic Custom Dialog Box . . . . . . . . . . . . . . . . 5-9
5-5. A Dialog Box With a Timeout . . . . . . . . . . . . . . . 5-11
5-6. The Default Value is Output When the Timeout Period Expires 5-12
5-7. A Dialog Box With Constraint Checking . . . . . . . . . . 5-13
5-8. A Self-Modifying Panel . . . . . . . . . . . . . . . . . . 5-14
5-9. A Status Panel . . . . . . . . . . . . . . . . . . . . . . 5-20
5-10. A Status Panel Example Program . . . . . . . . . . . . . 5-21
5-11. A Simple Animation . . . . . . . . . . . . . . . . . . . . 5-23
5-12. The Ocean: A Simple Animation . . . . . . . . . . . . . 5-24
5-13. A Picture Cache . . . . . . . . . . . . . . . . . . . . . . 5-25
6-1. Specifying System-Specific Threads . . . . . . . . . . . . 6-3

# Tables

5-1. **`Dialog Box`** Object Features . . . . . . . . . . . . . . . . . 5-4
5-2. Differences Between **`showPanel()`** and **`Show Panel On Execute`** . . . . . . . . . . . . . . . . . . . . . . 5-16

1

# Building an Operator Interface with Panels

# Building an Operator Interface with Panels

## What are Panels?

Panel Views (also called panels) are the user interface to your program. You generally create them after you've completed the functional part of your program. The Detail View (your development area) and the Panel View are two views of the same program; just like heads and tails are the two sides of the same coin.

You generally copy only display and data input objects to the panel. Only the objects themselves (and not the lines or pins) are displayed.

A Panel View contains a subset of the **File** menu and no other menus. The panel environment is a safe place from which your user can run the program without concern for modifying the program itself. Figure 1-1 shows a Detail View and Figure 1-2 shows the corresponding Panel View.

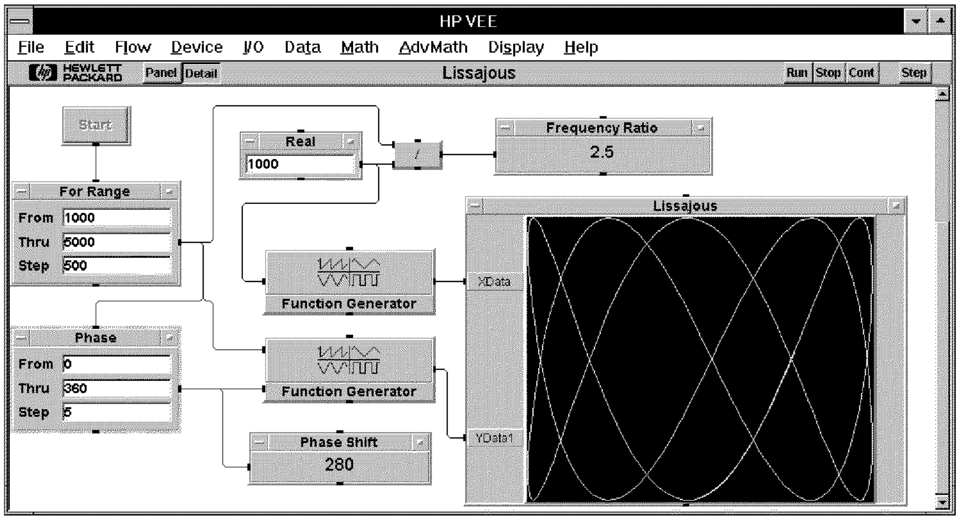


**Figure 1-1. A Detail View**

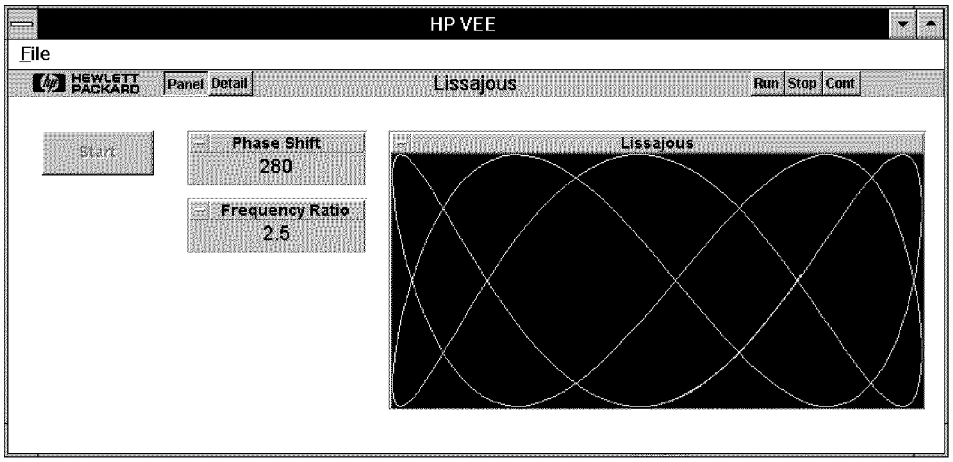


**Figure 1-2. The Corresponding Panel View**

HP VEE has three types of panels; they all are similar in the way you create, modify, and use them:

- Main Panel - The panel associated with the main context. If an object is in the work area and you add it to the panel, it will be in the Main Panel. The Main Panel is the panel that is always visible in the Panel View. Building a Main Panel is described in "Building a Main Panel" in this chapter.
- UserObject Panel - Each UserObject has its own panel. This panel can be used on its own or added to the Main Panel. This panel can be visible all the time or popped up when the UserObject is executing (like a dialog box). Building a UserObject Panel is described in "To Build a UserObject Panel" in Chapter 2.
- UserFunction Panel - Each UserFunction has its own panel (just like a UserObject). This panel pops up when the UserFunction is executing or when specified via a `showPanel()` object. Building a UserFunction Panel is described in "To Build a UserFunction Panel" in Chapter 2.

## Building a Main Panel

The Main Panel is the foundation of your user interface. No matter what type of panel you build, you use the same skills as when building the Main Panel.

## To Add Objects to a Panel

1. Put the object in its open view. Most of the time you want the open view on your panel; once the object is on the panel, you can't minimize or open it.
2. Select the object(s).

   Use either **`Edit`** ⟹ **`Select Objects`** or press (Ctrl) while clicking with the left mouse button on the objects.
3. Select **`Edit`** ⟹ **`Add To Panel`**.

You will now see the Panel View of your program. Switch between the Panel View and the Detail View by pressing the (Panel) and (Detail) buttons on the left side of the tool bar. Any time you want to add other objects to the panel, start from the Detail View and follow the steps above.

**NOTE**

The Panel View is a fixed-size window; there are no scroll bars. Objects selected from the Detail View are put in relatively the same position when added to the panel, but they are moved to fit on the fixed-sized Panel View.

## To Delete Objects from a Panel

1. Make sure you're in the Panel View.
2. Delete the objects.

   Either select **Delete** from the object menu of the object or place the cursor on the object and press Ctrl+D.

   The object remains in the Detail View, but is deleted from the Panel View.

If you delete (**Cut**) an object from the Detail View, it is automatically deleted from the Panel View.

**NOTE**

If you **Cut** and then **Paste** objects on the Detail View, any associated Panel View objects are deleted and you'll have to add them to the panel again. Use **Copy** and **Paste** instead.

**NOTE**

When running a program, you cannot **Size**, **Move**, **Edit Properties**, or **Delete** objects from the panel. You must pause or stop the program first; press Stop once to pause or twice to stop the program.

## Changing the Appearance of a Panel

Generally, an object's appearance on the Panel View is unlinked from its appearance on the Detail View. For example, if you change the color of an object on the Panel View, it does not change the color of the associated object on the Detail View (and vice-versa). However, the data displayed and other execution-related information are linked; if changed in one view, it changes in the other.

**Note: Appearance Independence Exception**

When an object on the Panel View or the Detail View inherits the default properties (you haven't changed colors or fonts from the `default` settings), the properties will change when the system defaults change.

### Properties That Are Not Linked Between Panel and Detail Views

- `Show Title Bar` (if object is shown in the open view)
- `Layout` group properties:
  - ☐ `Horizontal`
  - ☐ `Vertical`
  - ☐ `Rectangular`
  - ☐ `Circular`
  - ☐ `Show Digital Display`
  - ☐ `Show 3-D Border`
  - ☐ `Scaled`
  - ☐ `Show Caption`
  - ☐ `Graph Only`
  - ☐ `Scales`
  - ☐ `Scales & Sliders`
  - ☐ `Traces & Scales`
- `Color` tab: all properties - see the exception above
- `Font` tab: all properties - see the exception above

- `Label Justification` group properties
- `Editing Enabled` property (on `Note Pad` objects)
- `Format` group properties:
  - □ `Radio Buttons | Cyclic Button | List | Drop-Down List | Pop-Up List`
  - □ `Button | Check Box | Vertical Paddle | Horizontal Paddle | Vertical Rocker | Horizontal Rocker | Vertical Slide | Horizontal Slide`
- `Limits` group properties:
  - □ `High Color`
  - □ `Mid Color`
  - □ `Low Color`
- `Sub-Range Configuration` group properties:
  - □ `Colors`
- `Icon` tab properties:
  - □ `Picture` file and display mode
  - □ `Show Title`
- `Appearance` group properties (available on the Panel View only):
  - □ `Size & Position (X, Y, Width, Height)`
  - □ `Border`

All other properties are linked between Panel and Detail Views (a change in one view changes both views).

## To Change an Object's Border Appearance

In the Panel View, you can change the "depth" appearance of an object by specifying a border option.

1. From the object menu, select `Edit Properties`.
2. Select the `Appearance` tab.

3. Choose from the `Border` group: `None`, `Flat`, `Sunken`, or `Raised`. Click `OK`.

Figure 1-3 shows the appearance of the different border styles.

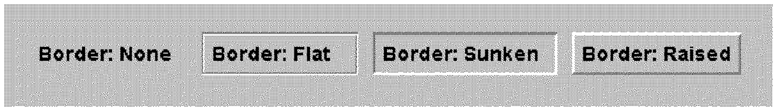


**Figure 1-3. Border Styles**

**Note: When The Border Is Set to** `None`

Note that when the border is set to `None`, you can't use the corner resize shortcut; you must select `Size` from the object menu. Also note that an object's color extends to where the border would be. You can see where the border is when moving or sizing an object.

Regardless of the border setting, the object's size does not change. For example, if you've sized an `OK` object without a border, and then choose to have a border, part of the object text may be obscured by the border. In Figure 1-4, both objects are the same size. The one on the left has `Border` set to `None`; the one on the right has `Border` set to `Flat`.

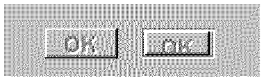


**Figure 1-4. A Border Width Example**

## To Label Areas

To add text to label an area on the panel, use the `Label` object.

1. From the Detail View, select `Display` ⟹ `Label`.
2. From the `Label` object's object menu, select `Edit Properties`.
3. Change the title to the desired label text.
4. Click `OK`.
5. Select the `Label` object and select `Edit` ⟹ `Add To Panel`. Position the label where you want it.

The `Label` object color by default is tied to the default Panel View color (set from `File` ⟹ `Edit Default Preferences`) so that the label background blends invisibly into the Panel View.

You change the label text from either the Detail or Panel View by changing the title text via the Properties dialog box. You can also change the justification of text by specifying `Left`, `Right`, or `Center` justification via the Properties dialog box.

## To Align Objects

When you add an object to the Panel View, its upper-left corner is automatically aligned to a user-definable grid, but the size of the object is not changed. When you move an object on the Panel View, its upper-left corner aligns to the nearest grid detent. To resize the object to the grid, move the cursor to the lower-right corner of the object, get the corner resize cursor, and click.

Follow these steps to align objects to a specific size or location (not on the grid):

1. From the object menu of an object on the Panel View, select `Edit Properties`.
2. Select the `Appearance` tab.

3. Set the values in the `Size & Position` group to change the location or size of the object.
4. Click `OK`.

Repeat these steps until the object is aligned properly.

Another way to align objects is to set the grid size to `1` and then move or size the object as desired.

## To Change the Grid Size

1. Select `File` ⟹ `Edit Properties`.
2. Select the `Panel` tab.
3. Set the `Grid Size`. The smallest grid size is `1`.
4. Click `OK`.

When you change grid size, objects do not automatically align to the grid. When you click on an object, its upper-left corner aligns to the nearest grid detent. When you click on the corner resize area, the object resizes to the nearest grid detent. Any `Move` or `Size` aligns the object to the current grid size.

**NOTE**

Each panel has its own grid. To change the grid size on a UserObject or UserFunction grid, select `Edit Properties` from its object menu and then follow the rest of the steps listed above.

## To Match Object Sizes

You may want to match the size of one object to the size of another object. If the object to match is already sized to the grid, use corner resize until the objects' sizes match.

Follow these steps to match objects' sizes when the size is not a multiple of the grid size:

1. From the object menu of one object, select `Edit Properties`.
2. Select the `Appearance` tab.
3. Under the `Size & Position` group, note the values for `Width` and `Height`.
4. Click `OK`.
5. From the object menu of the other object, select `Edit Properties`.
6. Select the `Appearance` tab.
7. Under the `Size & Position` group, set the values for `Width` and `Height` to the values noted in the previous object.

## To Add a Picture to the Panel

There are two ways to add a graphical image, such as a bitmap, to the Panel View.

**To add an image directly to the background of the Panel View:**

1. Select `File` ⟹ `Edit Properties`.
2. Select the `Panel` tab.
3. Choose a `Background Picture` file.
4. Select a display mode: `Actual`, `Centered`, `Scaled`, or `Tiled`. See Figure 1-5.

5. Click `OK`.

**To place one or more movable and sizable images on the Panel View:**

1. From the Detail View, select `Display` ⟹ `Picture`.
2. Choose a `Picture` file.
3. From the object menu of the `Picture`, select `Edit Properties`.
4. Select a display mode: `Actual`, `Centered`, `Scaled`, or `Tiled`. See Figure 1-5.
5. Click `OK`.
6. Select the `Picture` object.
7. Select `Edit` ⟹ `Add To Panel`.

The currently supported graphics formats are:

Bitmap (*.BMP) - Windows™ and UNIX®
GIF87a (*.GIF) - Windows and UNIX
X11 Bitmap (*.icn or *.xpm) - Windows and UNIX
X11 Window Dump (*.xwd) - UNIX only

**Picture Display Modes**

Figure 1-5 shows the different display modes.

- `Actual` - The picture is displayed in its actual size. The object or panel is resized to the size of the picture.
- `Centered` - The picture is displayed in its actual size. The picture is centered on the object or panel (which remains the same size).
- `Scaled` - The picture scales to the current size of the object or panel. After this choice is selected, the scaling may take a few moments.
- `Tiled` - The picture remains at its actual size, but is repeated horizontally and vertically until it fills the entire object or panel.

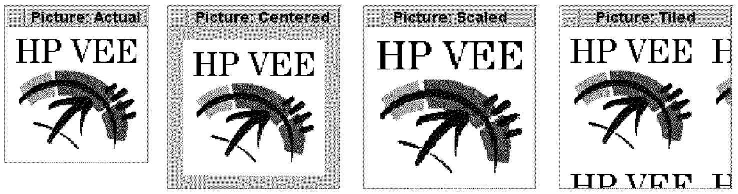


**Figure 1-5. A `Picture` Object With Different Display Modes**

**NOTE**

To increase execution speed, HP VEE caches graphic images. If you've loaded an image into HP VEE and then changed the file that contains your image, you need to select a different image file, and then reselect the original image file. Once you've done that, the new image will be used.

:

# 2

# Building UserObject and UserFunction Panels

# Building UserObject and UserFunction Panels

This chapter describes UserObject and UserFunction Panels and how to build them. It also describes creating, sizing, and positioning Pop-Up Panels.

## What is a UserObject Panel?

A UserObject Panel is very similar to the Main Panel. Each has its own context, so each has its own `Trig Mode` and `Edit` menu, as well as its own Panel View. A UserObject Panel is more flexible than the Main Panel in the ways it can be displayed.

**There are three ways to display UserObject Panels:**

- Independent Panel: A Panel on a UserObject.

  This method is useful when you want to "bundle" your UserObject for someone else to use or to simplify parts of your program for someone else to use without building a Main Panel. This type of panel is not visible from the Main Panel; it is only visible from the Detail View.

- Pop-Up Panel: A Panel on a UserObject with `Show Panel on Execute` specified.

  This method is usually used for dialog boxes that query for information, and then go away. This type of panel pops up on both the Detail View and Panel View at run time.

- A Panel on the Main Panel: A Panel on a UserObject that is then added to the Main Panel.

  This method is used to group functionality. You can also nest panels on other panels.

Figure 2-1 shows the different types of UserObject Panels.

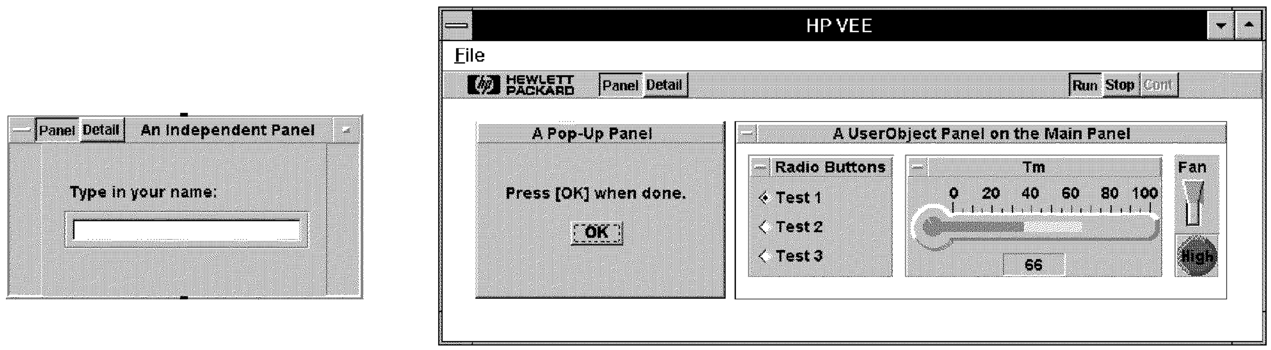


**Figure 2-1. Three Ways to Display a UserObject Panel**

## To Build a UserObject Panel

1. Select the object(s) in the UserObject.

   Use either `Edit` ⟹ `Select Objects` from the UserObject's `Edit` menu, or press (Ctrl) while clicking the left mouse button on the objects you want.

**NOTE**

Objects to be put on the UserObject Panel, must be in the UserObject's Detail View. You cannot select objects from outside the UserObject boundaries or from nested UserObjects.

2. From the UserObject's `Edit` menu, select `Add To Panel`.

Now you've created a UserObject Panel View that is independent of the Main Panel. You switch between the Panel View and the Detail View by pressing the Panel and Detail buttons on the left side of the UserObject's title bar. Any time you want to add other objects to the UserObject panel, start from the Detail View and follow the steps above.

## To Build a Pop-Up Panel

Once you've created a UserObject Panel, follow these steps to make the panel pop up when the UserObject executes:

1. From the UserObject's object menu, select `Edit Properties`.
2. On the `General` tab, on the `Pop-Up Panel` group, select the `Show Panel On Execute` check box.
3. Click OK.

While the program runs and the UserObject executes, the UserObject Panel pops up and is visible on both the Detail View and the Main Panel (if it exists).

**NOTE**

To change the appearance of the Pop-Up Panel, set the `Show Title Bar` and `Show Border` properties in the `Pop-Up Panel` group in the Properties dialog box.

To change the background color of the Panel View, set the `Background` property in the `Panel View` group on the `Colors` tab in the Properties dialog box.

**NOTE**

You generally use a Pop-Up Panel when creating a custom dialog box. For details, refer to "Building a Custom Dialog Box" in Chapter 5.

If you need a simple dialog box, you may not need to create a Pop-Up Panel. Use one of the `Dialog Box` objects. For details, refer to "About Built-In Dialog Box Objects" in Chapter 5.

## To Keep a Pop-Up Panel Displayed

Once you've created a Pop-Up Panel and run it, you may notice that it flashes up and is instantly gone. This action happens because nothing pauses the execution of the UserObject, therefore you can't see or respond to the information on the Pop-Up Panel. Follow these steps to add an **OK** button to maintain execution of the UserObject until you click on the button.

1. Select `Flow` ⟹ `Confirm (OK)`.
2. Place the **OK** object on the Detail View of the UserObject.
3. Select the **OK** object.
4. From the UserObject's `Edit` menu, select `Add To Panel`.

Now when you run the program, the UserObject Panel pops up and stays up until you press the **OK** button.

For details about using [OK] and [Cancel] buttons and creating custom dialog boxes, refer to "Building a Custom Dialog Box" in Chapter 5.

## To Change the Size of a Pop-Up Panel

The size of a Pop-Up Panel is the size of the Panel View of the UserObject. Follow these steps to change the size.

1. Put the UserObject in its Panel View by pressing the (Panel) button on the left side of the UserObject's title bar.
2. Resize the panel to the desired size.

   From the object menu, select `Size`. Or position your cursor at the lower right corner of the object and resize by dragging the corner.

When you resize the UserObject Panel, you're including the size of the title and border.

If the `Show Terminals` property check box is selected, you'll be resizing the entire object including the terminal area. The size of the Pop-Up Panel when it's displayed does not include the terminal area.

## To Change the Location of a Pop-Up Panel

The first time you run your program, the Pop-Up Panel is displayed in the center of the HP VEE work area. You change the location of the Pop-Up Panel by repositioning the panel while the program is running.

1. Press (Run).
2. When the panel pops up, drag it to the desired location.

   Every time the program is run, the panel will pop up to that location. Note that you can position the Pop-Up Panel to any location, the grid does not affect the location.

**NOTE**

The location of the upper-left corner of the entire Pop-Up Panel stays in the same place regardless of the `Pop-Up Panel` properties, or whether `Show Title Bar` and `Show Border` are selected or not. For example, if you have set the location of a Pop-Up Panel and then turn off `Show Title Bar`, the panel will shift so that the upper-left corner of the Pop-Up Panel is now located where the upper-left corner of the Pop-Up Panel's title bar was.

## To Add a UserObject Panel to the Main Panel

Once you've created a UserObject Panel, follow these steps to add the panel to the Main Panel.

1. Put the UserObject in its Panel View.

   Press the (Panel) button on the left side of the title bar.
2. Select the UserObject.
3. Select `Edit` ⟹ `Add To Panel` from the main menu.

Now you'll see the Main Panel with the UserObject Panel displayed on it.

You can nest a UserObject Panel on its "parent" UserObject Panel by following the same steps above, except you must use the `Edit` menu of the parent UserObject.

Once the UserObject Panel has been added to the Main Panel, change its size, location, and other appearance details the same way you change them for objects on the Main Panel as described in "Changing the Appearance of a Panel" in Chapter 1.

**NOTE**

When you nest a panel on another panel (as described above), you're only making a copy of its contents. When you need to add or delete objects from the nested panel, you need to delete the copies and modify the original UserObject Panel and then copy it again. Therefore, make sure that the layout, colors, and fonts of your original UserObject Panel are as you want before adding objects, so that if you need to add or delete objects, you won't need to delete any customization.

## To Add Objects to a Nested UserObject Panel

Once a UserObject Panel has been added to the Main Panel (or to a parent UserObject Panel), it cannot be added to easily. You must add objects to the original UserObject Panel and then add the panel to the Main Panel.

1. From the Main Panel, delete the UserObject Panel.

   From the object menu, select `Delete` or double-click on the object menu button.

2. Return to the Detail View by pressing the (Detail) button on the left side of the tool bar.
3. Put the UserObject in its Detail View by pressing the (Detail) button on the left side of the title bar.
4. Add any desired objects to the Detail View of the UserObject.
5. Select the objects to add to the panel.
6. From the UserObject's `Edit` menu, select `Add To Panel`.

Now that the new UserObject Panel has been created, add this panel to the Main Panel following the steps listed earlier in this chapter in "To Add a UserObject Panel to the Main Panel".

## To Delete Objects from a Nested UserObject Panel

1. From the Main Panel, delete the UserObject Panel.

   From the object menu select `Delete` or double-click on the object menu button.

2. Return to the Detail View by pressing the Detail button on the left side of the tool bar.
3. Put the UserObject in its Panel View by pressing the Panel button on the left side of the title bar.
4. Delete the desired objects by selecting `Delete` from their object menus.

Now add this panel to the Main Panel following the steps listed earlier in this chapter in "To Add a UserObject Panel to the Main Panel".

**NOTE**

If you do not follow the steps above and instead delete an object directly from the Panel View that is nested on the Main Panel, it will be deleted only from that copy. If you make changes to the original UserObject Panel, and copy it again to the Main Panel, the previously deleted object (from the original UserObject Panel) will be there.

## To Delete a UserObject Panel from the Main Panel

1. Go to the Main Panel by pressing the (Panel) button on the left of the tool bar.
2. Delete the UserObject Panel.

   From the object menu select `Delete` or double-click on the object menu button.

The UserObject Panel still exists, but it is no longer added to the Main Panel.

## What is a UserFunction Panel?

A UserFunction Panel is built and runs basically the same way as a UserObject Panel. However a UserFunction Panel is only visible when the UserFunction is executing (a Pop-Up Panel) or if programmatically displayed via **`showPanel()`**. A UserFunction Panel cannot be added to the Main Panel (or parent UserFunction or UserObject Panels).

## To Build a UserFunction Panel

If you have already created a UserObject Panel and then made the UserObject into a UserFunction, the UserFunction Panel automatically exists. If the UserObject Panel was added to the Main Panel, it was automatically deleted because a UserFunction Panel cannot be on the Main Panel.

If you have a UserFunction and want to create a panel for it, follow these steps.

1. Select the object(s) in the UserFunction.

   Use either `Edit` $\Longrightarrow$ `Select Objects` from the UserFunction's `Edit` menu, or press Ctrl while clicking the left mouse button on the objects you want.

**NOTE**

Objects on the UserFunction Panel must be in the UserFunction's Detail View. You cannot select objects from outside the UserFunction boundaries.

2. From the UserFunction's `Edit` menu, select `Add To Panel`.

   Now you've created a UserFunction Panel View. Switch between the Panel View and the Detail View by selecting `To Panel` or `To Detail` from the UserFunction's object menu.

3. Select `Close` when you are finished editing.

Any time you want to add other objects to the panel, start from the UserFunction's Detail View and follow the previous steps.

## To Build a UserFunction Pop-Up Panel

1. Select `Edit` ⟹ `Edit UserFunction`.
2. Select the UserFunction name.
3. From the UserFunction's object menu, select `Edit Properties`.
4. On the `General` tab, on the `Pop-Up Panel` group, select the `Show Panel On Execute` check box.
5. Click `OK`.

The UserFunction Pop-Up Panel operates the same way as a UserObject Pop-Up Panel as explained in "To Build a Pop-Up Panel" earlier in this chapter.

**To Build a Status Panel**

A status panel is programmatically "popped up" with `showPanel()`. For details refer to "Building a Status Panel" in Chapter 5.

## To Change the Size of a UserFunction Pop-Up Panel

1. Select `Edit` ⟹ `Edit UserFunction`.
2. Select the UserFunction name.
3. From the UserFunction object menu, select `To Panel` to see the Panel View.
4. Resize the panel to the desired size.

   From the object menu, select `Size`. Or position your cursor at the lower right corner of the object and resize by dragging the corner.

When you resize the UserFunction Panel, you're resizing the entire UserFunction area including the area with the `Close` button, the title, and the border.

If the `Show Terminals` property check box is selected, you'll be resizing the entire object including the terminal area. The Pop-Up Panel size when it pops up does not include the terminal area or the `Close` button area.

## To Change the Location of a UserFunction Pop-Up Panel

Follow the same steps as with a UserObject Panel shown in "To Change the Location of a Pop-Up Panel" earlier in this chapter.

3

# Using the Keyboard to Navigate a Panel View

## Using the Keyboard to Navigate a Panel View

Perhaps your users prefer to navigate around the Panel View using only the keyboard or perhaps they do not have access to a mouse in their environment. For these users, build a panel that is keyboard “navigable”.

# Making Your Panel Keyboard Navigable

This chapter describes making a keyboard-navigable Panel View.

In general, when using the keyboard to run a program, first an object must get **focus**, then the object's data value or state can be changed.

**Definition**

*Focus* is a term indicating a location where you can perform an action. In HP VEE, the object or field that has focus is either surrounded by a marquee (a dotted rectangle) or, for a type-in field, a vertical "caret" is shown and existing text is shown in inverted colors if the object was tabbed to; if a object was clicked-on, it gets focus but the appearance of the existing text is not changed.

There are several things to remember about keyboard navigation:

- It's hard for the user to tell when the object on the Panel View has operated (except OK buttons which are grayed out unless they can be selected). Inform the user that an object has operated or have the user indicate when they have finished making changes (for example, by pressing an OK button).
- Without a mouse, the user can't access an object's object menu (this access is useful with `Display` object menu choices such as `Zoom` and `Auto Scale`). You can add control input terminals for some of these parameters, if you need to let your user use them.
- There is an HP VEE-specified pattern of tabbing through a Panel View. The pattern is discussed in "Panel View Navigation Techniques" later in this chapter.
- Different objects have different rules for keyboard interaction. You must tell the user what they are; see "Panel View Selection Techniques" later in this chapter for details.

## To Give Focus to an Object or Panel

**Some information about focus:**

- You can modify the data on an object only when it has focus.
- When a panel pops up, only objects on the Pop-Up Panel get focus until the panel is closed.
- Only objects with data fields, buttons, or a selection area can get focus. Note that *every* data field, button, or selection area can be tabbed into.

  For example, if you have a `Slider` on the panel (as shown in Figure 3-1); the first (Tab) gives focus to the `Min Value` field.

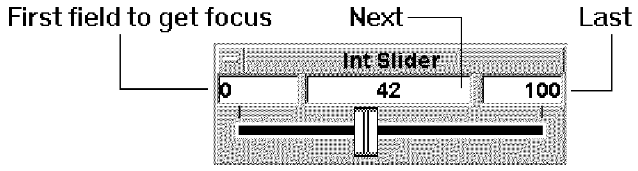


**Figure 3-1. A `Slider` on a Panel**

  To prevent users from tabbing through these fields and changing them, add control input terminals for fields such as `Min Value` and `Max Value`.

- `Selection Control`, `Toggle Control`, and `Slider` objects execute as soon as the program starts to run unless the `Wait for Input` property has been set. This property allows these objects to execute only after they get focus and have values changed or selected.
- If (Esc) or (Enter) is assigned to an `OK` button and then pressed, the `OK` button gets and keeps the focus.
- If a function key is assigned to an `OK` button and then pressed, the focus stays on the object that had focus before the function key was pressed.

**NOTE**

Do not use HP VEE's built-in `Dialog Box` objects on a keyboard-driven panel that contains other objects. After the `Dialog Box` object finish executing, no other object gets focus, so you can't navigate to other objects via the keyboard. For information about `Dialog Box` objects, refer to "About Built-In Dialog Box Objects" in Chapter 5.

## To Assign an OK Button to the Enter or Esc Key

When creating a panel, you can let your user press the Enter key (Return on UNIX) or Esc key to "press" a **`Confirm (OK)`** button. Otherwise, the user has to Tab to the button and press Space. This feature is especially useful on custom dialog boxes made from Pop-Up Panels. For more information about building custom dialog boxes refer to "Building a Custom Dialog Box" in Chapter 5.

To assign these keyboard keys to **`Confirm (OK)`**, follow these steps:

1. Select **`Flow`** $\Longrightarrow$ **`Confirm (OK)`**.
2. Add the **`Confirm (OK)`** button to the panel.
3. From the **`OK`** button's object menu, select **`Edit Properties`**.
4. From the **`General`** tab, the **`Panel View Operations`** group, select **`Assign to [Enter] key`** and/or **`Assign to [Esc] key`** check box(es).

**NOTE**

You can assign the (Enter) key to only one OK button per context and you can assign the (Esc) key to only one OK button per context. If you assign either of these keys to more then one OK button per context, the previous assignment is automatically turned off.

5. Click OK.

When the program runs, any OK button assigned to (Enter) has its label appear in a bold font to indicate that it is the default button.

**NOTE**

You can only use (Enter) and (Esc) to "press" an OK button from the Panel View of your program (or UserObject/UserFunction).

## To Assign a Button to a Function Key

When creating a panel, you can let your user press a function key (F1, F2, F3, etc. at the top of the keyboard) to "press" a `Confirm (OK)` button. Otherwise, the user would have to (Tab) to the button and press (Space). To assign keyboard function keys to `Confirm (OK)`, follow these steps:

1. Select `Flow` $\Longrightarrow$ `Confirm (OK)`.
2. Add the `Confirm (OK)` button to the panel.
3. From the OK button's object menu, select `Edit Properties`.

4. From the `General` tab, the `Function Keys` group, select the `Assign to Function Key` check box.
5. Select the function key to assign to this `OK` button (`F1` through `F12`). Some keyboards have only eight function keys, but HP VEE allows you to select any of the choices to allow cross-platform development; see "General Tips" in Chapter 6 for more information. On Windows, the `F10` key is a reserved key that functions the same as Alt.

**NOTE**

You can assign a specific function key to multiple **OK** buttons. When the function key is pressed, all of the associated buttons that are currently active will operate.

6. Click `OK`.

The title of the `OK` button is automatically changed to reflect the function key assignment. You can change the title to exclude the function key information.

**NOTE**

You can use function keys to "press" an **OK** button from either the Panel View or the Detail View.

## Panel View Navigation Techniques

To navigate around a Panel View, use the Tab key to go forward and Shift+Tab to go back. The order in which the objects and panels are traversed is as follows:

- After you press Run, HP VEE divides the panel into groups of objects. The first group is determined by the topmost (lowest y-coordinate value) object and contains all objects that have a y coordinate close to that of the topmost object. The first object to get focus is the leftmost object in the group. Tab moves the focus from left to right.

**NOTE**

If the leftmost object has `Auto Execute` set, it does not get the focus first, but you can Tab to the object.

The y-position of an object or panel is determined by the location of its upper-left corner.

- Once the focus has gone as far right as it can in the first group, it continues with the leftmost object in the next group.

  The next group is determined by the topmost object that was not in the previous group. Tab starts at the leftmost object in this group and continues to the right. This ordering repeats throughout the panel.

Figure 3-2 shows a typical grouping of objects.

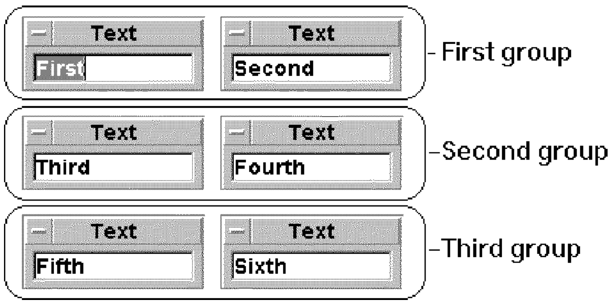


**Figure 3-2. A Simple Tabbing Order Example**

Figure 3-3 shows a more complex grouping of objects. At first it appears that the `Third Text` object should be included in the first group. But the first group is determined by the `Second Text` object, therefore the `Third Text` object is too far down to fit into the first group.

The third group is determined by the `Fifth Text` object. The `Fourth` and `Sixth` objects are close enough to be in that group.

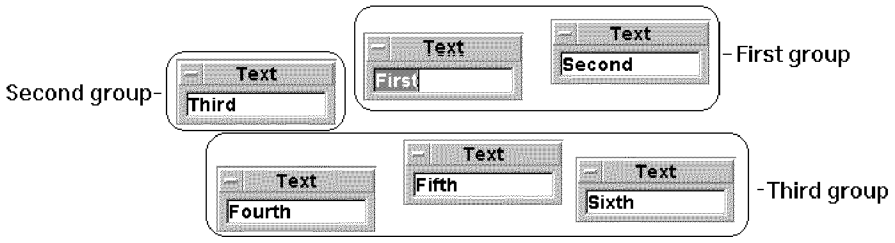


**Figure 3-3. A Complex Tabbing Order Example**

**NOTE**

An OK button can only be tabbed to when it is activated (when the text is not grayed out).

- If an object has multiple fields, (Tab) takes you through all the fields before going to the next object.
- If you are in a UserObject Panel nested on another panel, (Tab) takes you through all the objects in the UserObject before continuing to the next object out of the UserObject Panel.
- If you are in a Pop-Up Panel (UserObject or UserFunction), you can (Tab) only within the Pop-Up Panel until it finishes executing.

Remember that you can (Tab) to objects whether or not they've executed (except **OK** objects).

## Panel View Selection Techniques

When using a mouse, you click to select a choice or change a value. When using a keyboard, you tab to the object to give it focus and then you select a choice or change a value by using the following techniques:

- `Radio Buttons` and `List`:

  ▲ and ▼ to select a choice.
  (Space) to select the current choice.

- `Cyclic Button`:

  (Space) to change the choice.

- `Drop-Down List`:

  ▼ to get the list.
  ▲ and ▼ to select a choice.
  (Space) or (Return) to put the list away.
  (Esc) to cancel the choice.

- `Pop-Up List`:

  (Space) to get the list.
  ▲ and ▼ to select a choice.
  (Space) or (Return) to put the list away.
  (Esc) to cancel the choice.

- All `Toggle Control` objects:

  (Space) to change the value.

- All Edit Fields (such as `Constant` objects):

  Type in a new value.
  (Tab) or (Enter) to continue to the next field.
  (Esc) cancel the changes.

- `Start` and `OK` Buttons:

  (Space) to select them.

- Objects with multiple fields (such as `Function Generator`):

  Select a choice or change a value using one of the techniques listed above depending on the type of field. And then (Tab) to the next field.

**NOTE**

If you have (Esc) or (Enter) assigned to an (OK) object and you use (Esc) or (Enter) in an Edit field (such as in `Constant` and `Slider` objects), the (OK) button will be pressed after the Edit field data is entered.

When a `Data` ⟹ object has `Wait For Input` set, it executes when you select a choice or change a value as described above. When a `Data` ⟹ object has `Auto Execute` set, it starts the program running when you select a choice or change a value as describe above. To see an example that demonstrates `Wait for Input`, refer to "To Build a Self-Modifying Panel" in Chapter 5.

**NOTE**

Data in `Data` ⟹ `Constant` and `Data` ⟹ `Slider` objects must be changed for `Wait For Input` or `Auto Execute` to work. Replacing a character with the same character is interpreted as a change.

**NOTE**

When (Ctrl)+(C) is pressed to pause the program, it cancels any changes to the edit field that has focus.

4

# Guidelines for a Good Interface

# Guidelines for a Good Interface

This chapter contains guidelines and general interface design information. This information is not a substitute for Human Factors help or user input.

## Most Important

- Get advice from a professional user-interface designer. The information in this chapter contains only rough guidelines. If you have access to human factors engineers or graphical designers, ask them to look at your Panel Views and give you input.
- Get user feedback. Let your users use the panel. Ask them if they can find the information needed. Make sure they can identify what they should do. Verify that the colors and fonts are useful and that they allow comfortable long-term use.

**General User Interface Design Guidelines**

- Know your users and their environment.
- Empower users. Let them control the basic characteristics of the program. Minimize the number of rules users must learn to use the system.
- Be consistent in user interactions, data presentation, and layout. Group and order the information displayed.
- Accommodate different levels of users via defaults, help, or templates for novice or intermittent users, and accelerators or shortcuts for advanced or frequent users.
- Build users' confidence. Help them accomplish their tasks quickly and without frustration. Do not use accusative wording like "user error".
- Provide feedback to users. Let them confirm data input or directly see the result of their actions.
- Orient users. Provide them with enough information to understand the current state of the program and how to respond to this state.
- Design for error recovery. When users make mistakes, they need a chance to recover from them.
- Verify your design with the real users of the program.

## Using Colors

When using colors in your panels, consider the limitations of colors for your audience and the specific uses for color.

**Facts about Colors**

- Older people are often less sensitive to color and may need brighter color levels. The ability to discern shades of blue may be diminished. However, younger users may find brighter colors more fatiguing.
- Eight percent of men and 0.5 percent of women have some form of color deficiency. The most common deficiency is an inability to distinguish between red and green.
- There are individual variations in color perception. Different monitors, different environmental lighting, cultural and occupational connotations, and personal preferences for colors can all affect the way people see, distinguish, and perceive color.
- Be aware that colors have different meanings in different cultures; it's best to use neutral colors for large areas.

**Redundant Cues**

Because of the reasons listed above, use color only as an additional form of communication. For example, to distinguish a `Stop` from a `Go` button, don't just change the color from Red to Green; also make sure there is explanatory text that changes. One way to make sure your color is used as a redundant cue is to first create the panel without specifying colors, and then add them as needed.

- Use as few colors as possible. More than four or five different colors on a single panel or seven different colors in your entire user interface, may be distracting and confusing.
- Colors project an image about your program. Bright colors may project a toy-like quality. Muted or subtle colors may create an impression of sophistication or professionalism.

- For background and text (foreground) combinations, use colors that have high contrast; but avoid putting colors from opposite sides of the color wheel together (like yellow and blue). The contrast is good, but boundaries may "bleed" into one another. Figure 4-1 shows the colors on a color wheel.

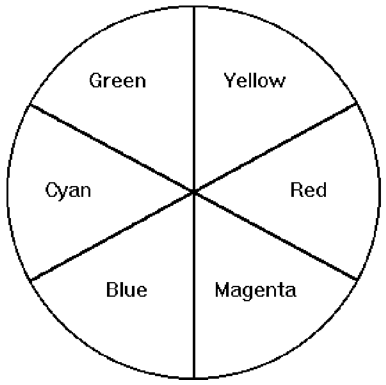


**Figure 4-1. A Color Wheel**

- Use generally-accepted color encodings, such as Green for "Go" or "Normal Status" and Red for "Stop" or "Emergency". This color coding may be dependent on cultural or occupational standards.
- Consider the printing needs of your audience. If the user prints a color screen to a color printer, the colors on the printer won't necessarily match those on the screen due to printer limitations. If the screen is printed to a non-color printer, make sure the colors translate well to black and white. Note that on the `Default Preferences` Properties dialog box, the `Printing` tab `Screen Element` settings do not apply to color changes from the default colors. When you've changed colors from the defaults, HP VEE automatically gray-scales the colors on the printout.

  **So if it's so troublesome, why use color?**
- To enhance the aesthetics. An interface with pleasant-looking colors is easier to operate and view for long periods of time.
- To group objects together visually.
- To aid navigation and recognition.

- To enhance readability.
- To show what is similar and what is different. Different colors distinguish different groups of information from each other.
- To call attention to an object or event. It's easy to find information if it is colored differently.
- To accentuate information encoding. Color information can aid understanding, especially when using common encodings like Yellow for caution, Red for danger or failed, Blue for water or cooling.

Use a basic overall color scheme, keeping similar functions the same color and only use contrasting colors when needed.

## How to Pick Colors

Here are some guidelines to help you pick colors that enhance the usability and readability of your panel.

Generally you want to use a neutral-colored background, White or Gray is best. However, lighter colored backgrounds increase the perception of flicker (the amount depends on the system monitor). Shades of Blue are also very good for backgrounds (for both objects and panels).

If your background color is too light, the border highlight of a "3-D" object may not stand out. If the background color is too dark, the object's border shadow may not stand out.

When displaying text, pick colors that have good contrast with the background. For example, if the background color is White, the following colors are the best and worst for text:

- Black - Best
- Blue - Best
- Red - Best
- Yellow - Worst (low contrast)
- Cyan - Worst (low contrast)

If the background color is Black, the following colors are the best and worst for text:

- Yellow - Best
- White - Best
- Green - Best
- Blue - Worst
- Red - Worst (may "bleed" into the background)
- Magenta - Worst (may "bleed" into the background)

Note that on HP VEE display objects (such as `X vs Y`), you'll have a black background for the graph area and you'll want to pick good colors for the graph lines.

Figure 4-2 shows how these colors can be used:

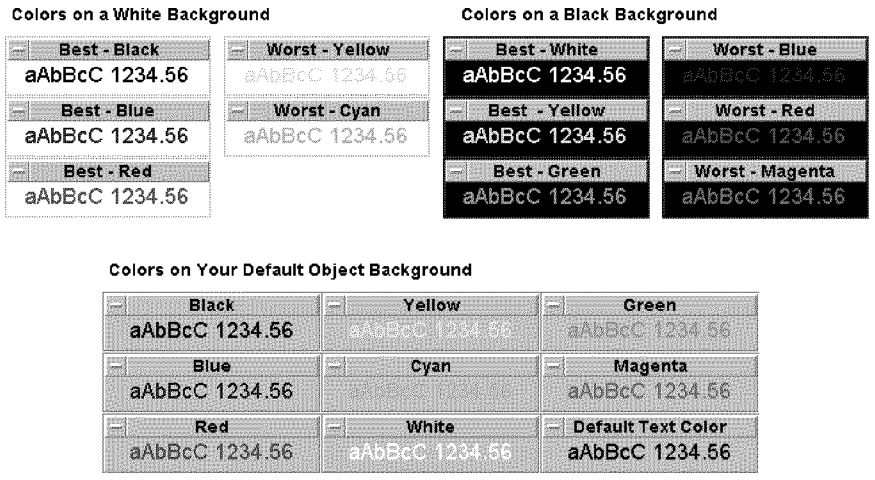


**Figure 4-2. Displaying Different Color Text**

This program is saved as `colors.vee` in your `manual` examples directory. Open this example to see what these colors look like on your screen.

Regardless of the background color, make sure that the color of the text displayed is about the same level of brightness and has good contrast. These guidelines for object background and text also apply to panel backgrounds and objects placed on them.

Generally shades of Blue work best as a background. To use Blue as a text color, use a greenish blue (like Teal).

## Using Fonts

**Facts about Fonts**

- It takes users about 25% more time to read text on a computer screen than on paper. Comprehension is about the same. Therefore, putting large amounts of text on your panel will slow down your user.
- There are proportional and non-proportional fonts. A proportional font uses different widths for different letters; for example, the space used to print the letter "w" is wider than the space used to print the letter "i". Proportional fonts are generally easier to read and look better. Some common proportional fonts are: **Lucida**®, **Arial**®, and **Helvetica**™.

  A non-proportional font uses the same width for all letters and is used when you need to vertically align data in a table. Some common non-proportional fonts (also called mono-spaced fonts) are **Courier New**, **Courier**, and **MS LineDraw**.

  The fonts available in HP VEE depend on the fonts installed on your system.

  **AaBbYyZz - A proportional font**

  `AaBbYyZz - A non-proportional font`

- If your program is run on a different system than the development system, some fonts may not be available. Refer to "To Choose Fonts for Multiple Platforms" in Chapter 6 for information on using fonts on multiple systems.
- Font size indicates the height of characters, not the width. So if you change the font type, but keep the font size the same, you may have to resize the object to accommodate the text width.

- Generally, use a 12-point font size for text meant for general reading. You may want to use a larger font or a bold font for emphasis. Although font size on paper is an absolute size (like inches or centimeters), fonts on a computer screen are based on "logical" font size, which varies according to your monitor resolution. So no matter what size font you use, you'll need to make sure it is readable on the screen. Figure 4-3 shows different font sizes.

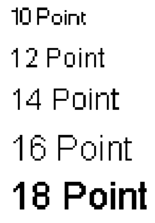

10 Point
12 Point
14 Point
16 Point
**18 Point**

**Figure 4-3. Font Sizes**

**NOTE**

Some windowing systems may not scale large-sized fonts well. SunOS Open Windows version 3.0 does not scale some characters properly above 96 point.

- Some studies show that serif fonts (those with thin lines at the top and/or bottom of the main strokes of the letter) allow the user to read faster than sans serif fonts. You'll need to see if this makes a difference for your users. Figure 4-4 shows a serif and a sans serif font.

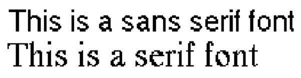

This is a sans serif font
This is a serif font

**Figure 4-4. Serif and Sans Serif Fonts**

- Text is faster to read if it is left-justified with a ragged right margin.

- Use no more than three different font types, and four font colors per program. Use fonts consistently to help the user understand which text is more important. Generally, larger text is the most important.

  Use font types and sizes consistently to make it easier for you to align and balance the text across the screen.
- Blue is generally a bad color for text - the eye is least sensitive to this color.
- Verify your font choices with the users of your program to ensure legibility.

## Layout Tips

- Maintain a consistent layout even as the panel's content changes. For example, if you have a message area at the bottom of the screen, a display area on the side, and a status area at the top - stay with this layout; don't move elements around as the program runs.
- Group related information.
- Balance information through the available screen area.
- Remember to leave "white" space (blank areas without objects) to make your panel easier to use. If the screen is full, with objects positioned against other objects with no space between them, it is harder for your user to find needed information.
- Always place navigation controls in the same place on every panel.
- Align and size objects in related groups so they are the same size. Align them horizontally or vertically so that there is not a ragged edge to disrupt identifying and scanning the objects. Make the spacing between objects a constant distance.
- Use dialog boxes to present infrequent or low-use features. This lets the user respond and be done with this information, and reduces clutter on the Main Panel.

## To Call Attention to an Object

Use visual attributes to call attention to areas on the screen:

- Use object size. The eye is naturally drawn to larger objects. There also is a connotation of importance. If you are differentiating information by size, use a maximum of five different sizes.
- Use border characteristics to distinguish objects. Use lighter colors on the raised surfaces and darker colors on the recessed surfaces. Refer to Figure 1-3 to see the different border styles.

- Use different colors to indicate different levels of importance. You should use a maximum of four or five different colors on a panel. And a maximum of seven different colors in your entire user interface. If you decide to use more, supply a legend to explain each color's purpose or meaning.
- For fine details or small areas, black, white, and gray give the best readability.
- Brighter colors attract more attention than dimmer colors (for example, using Blue, Yellow, or Red, instead of Blue Gray, Gold, or Light Red). Users can differentiate between a maximum of five different brightnesses.
- Lighter colors and warmer colors (such as Red) appear to "advance" to the user. Darker colors and cooler colors (such as Blue) appear to "recede".
- Use font size. Use at least four points of size difference to denote relative importance.
- Use font styles such as **bold** and *italics*. Italics makes text slower to read, so don't use italics for text requiring quick reading.
- Blinking/Flashing is an excellent way to get attention. With HP VEE, some ways to direct attention are to pop-up a panel, change bitmaps in a `Picture` object, change colors and text in a `Color Alarm` object, or use a `Dialog Box` object. No more that four items should be blinking or flashing at a time.

Another way to draw attention is to use sound. Use the `Beep` object (if the hardware supports it).

Each of these attributes should be used in a consistent way throughout your program.

5

# Advanced Topics

## Advanced Topics

This chapter presents some advanced topics and tasks you may use when creating your operator interface;

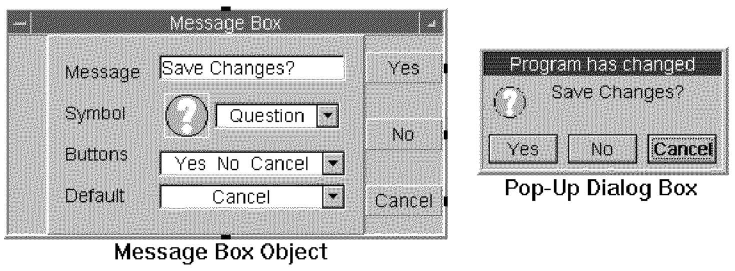
- Information About Built-In `Dialog Box` Objects
- Building a Custom Dialog Box
- Building a Status Panel
- Using Animation

## About Built-In Dialog Box Objects

Before creating a custom dialog box, consider if the built-in `Dialog Box` objects (located under the `Data` menu) will meet your needs. They are easy to use and have many useful features.

The differences between `Dialog Box` objects and Pop-Up Panels (used to build custom dialog boxes) are:

- `Dialog Box` objects do not need to be added to a panel, they pop up on both the Panel View and the Detail View automatically. Figure 5-1 shows a `Message Box` object and the panel it generates.



Message: Save Changes?
Symbol: Question
Buttons: Yes No Cancel
Default: Cancel

**Figure 5-1. A Message Box**

- `Dialog Box` objects perform only one operation per object - they ask for specific user input. If you need to ask the user several questions at once, or to display data, then create a custom dialog box.
- `Dialog Box` objects are modal. In other words, while they are popped up, the rest of the program is stopped.

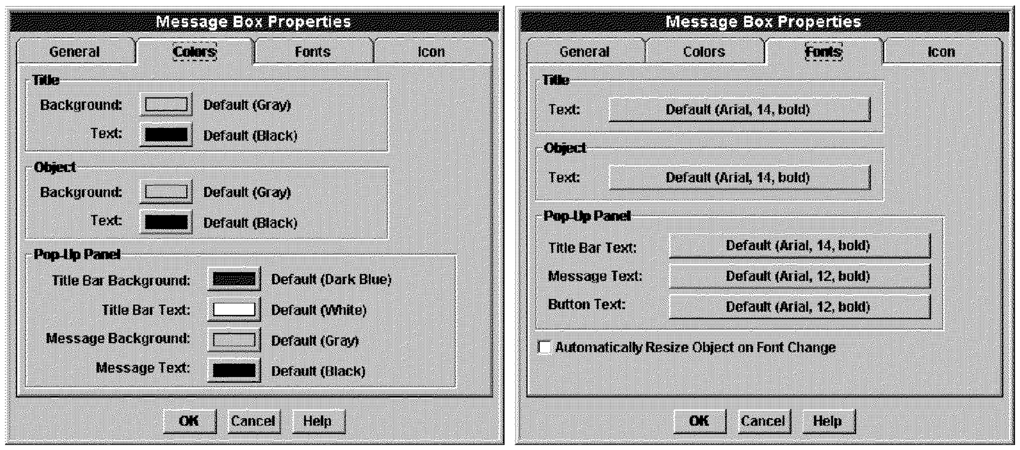
  This is different than Pop-Up Panels; while Pop-Up Panels are popped up, the rest of the program continues to execute.
- `Dialog Box` objects should not be used on a keyboard-driven panel. After they finish executing, no other object gets focus, so you can't navigate to other objects via the keyboard.
- `Dialog Box` objects already have many built-in features. You don't have to write a program to get features such as constraint checking or timeouts. The built-in features of the `Dialog Box` objects are listed in Table 5-1.

**Table 5-1.** `Dialog Box` **Object Features**

| Dialog Box Object | Constraint Checking | Password Masking | Timeout | Custom Button Text | Formatting Options |
|---|---|---|---|---|---|
| `Text Input`<br>Gets text input | √ | √ | √ | √ | √ Field Width |
| `Integer Input`<br>Gets integer input | √ | √ | √ | √ | √ Field Width |
| `Real Input`<br>Gets real input | √ | √ | √ | √ | √ Field Width |
| `Message Box`<br>Displays information | n.a. | n.a. | √ | √ | √ Multiple Line Messages |
| `List Box`<br>Allows single or multiple selection from a list | n.a. | n.a. | √ | √ | √ List Height |
| `File Name Selection`<br>Allows file name selection | √ [1] | n.a. | | | |

[1] Gives a warning if overwriting an existing file. Doesn't allow opening a non-existent file.

`Dialog Box` objects have additional color and font properties so you can easily control the appearance of the pop-up panel. Figure 5-2 shows these properties.



Title
Background: Default (Gray)
Text: Default (Black)
Object
Background: Default (Gray)
Text: Default (Black)
Pop-Up Panel
Title Bar Background: Default (Dark Blue)
Title Bar Text: Default (White)
Message Background: Default (Gray)
Message Text: Default (Black)

Title
Text: Default (Arial, 14, bold)
Object
Text: Default (Arial, 14, bold)
Pop-Up Panel
Title Bar Text: Default (Arial, 14, bold)
Message Text: Default (Arial, 12, bold)
Button Text: Default (Arial, 12, bold)
☐ Automatically Resize Object on Font Change

**Figure 5-2. Message Box Properties**

`Dialog Box` objects are easy to use without a mouse. You press the Enter key (Return on UNIX) to "press" the default button; or press the Esc key to "press" the rightmost button (usually `Cancel`). Use Tab and Shift+Tab to move the focus to input fields or buttons. Press Space to "press" a button that has focus. Refer to "Making Your Panel Keyboard Navigable" in Chapter 3 for information about focus.

For details on the `Dialog Box` objects, refer to *HP VEE Reference*.

## To Output a Value on Timeout

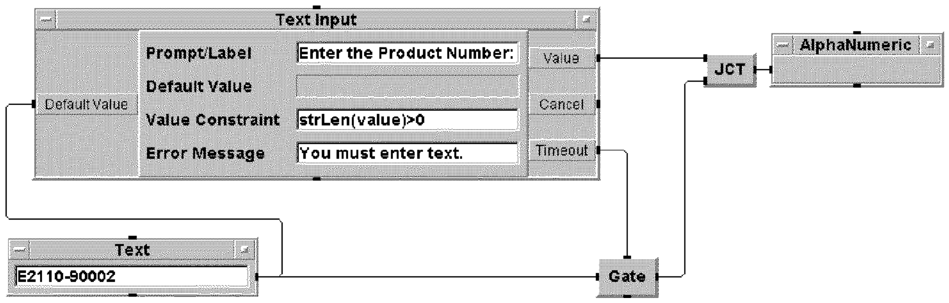
When you set a `Timeout` on a `Dialog Box` object (via the Properties dialog box), and the timeout period expires, the value `1` is output on the `Timeout` data output pin. Often you want a value (such as the `Default` value) output when the timeout period expires; to do this, follow these steps:

1. Set the `Timeout`.

   Select `Edit Properties` from the object menu. Select the `Timeout Enabled` check box. Set the `Pop-Up Duration`. Click `OK`.

2. Add the `Default Value` data input pin.
3. Select `Data` ⟹ `Constant` ⟹ `Text` to input the default value. (Select a `Data` ⟹ `Constant` object appropriate to your needs to input the default value.)
4. Input the default value into the `Text` object.
5. Connect the `Text` data output pin to the `Default Value` data input pin.
6. Select `Flow` ⟹ `Gate`.
7. Connect the data output pin of the `Text` to the data input pin of the `Gate`.
8. Connect the `Timeout` data output pin to the sequence input pin of the `Gate`. When the timeout period expires, the default value is output through the `Gate`.

Figure 5-3 shows a program that uses a `JCT` object to output either the entered value or the default value if the timeout occurred. This program is saved as **`timeout.vee`** in your **`manual`** examples directory.



Prompt/Label: Enter the Product Number:
Default Value:
Value Constraint: strLen(value)>0
Error Message: You must enter text.
E2110-90002

**Figure 5-3. Outputting a Default Value on Timeout**

## Building a Custom Dialog Box

When a built-in **Dialog Box** object doesn't fit your needs, create a custom dialog box. This section explains how to create custom dialog boxes from Pop-Up Panels. These steps show you how to create basic dialog boxes. Once they are created, you add the controls, displays or other objects as needed. If you prefer not to build programs by following the steps below, open the files (via **Help** ⟹ **Open Example**) that contain the dialog box programs and modify them to suit your needs.

To create a Pop-Up Panel, follow the steps in "To Build a UserObject Panel" in Chapter 2. You can make the Pop-Up Panel from either a UserObject or a UserFunction. Unless otherwise specified, the instructions below assume the use of a UserObject.

## To Build a Basic Dialog Box

1. Once you've built a Pop-Up Panel, follow the steps in "To Keep a Pop-Up Panel Displayed" in Chapter 2 twice. The first time to add an **OK** object so that users can confirm their choices, and the second time to add an **OK** object to be the **Cancel** button so that users can cancel their choices.
2. Assign the (Esc) key to the **Cancel** button.

   Select **Edit Properties** from the second **OK** object's object menu. Change the **Title** to **Cancel** and select the **Assign to [Esc] key** check box. Click **OK**.

   Now the **OK** button is assigned to the (Enter) key (by default) and the **Cancel** button is assigned to the (Esc) key.
3. Select **Flow** ⟹ **Exit UserObject** and place it in the Detail View of the UserObject.
4. Connect the data output pin of the **Cancel** button to the sequence input pin of **Exit UserObject**.

5. Select `Data` ⟹ `Constant` ⟹ `Text`. (This could be any object to get user input.)
6. Connect the data output pin of the `OK` button to the sequence input pin of the `Text` object.
7. Create a data output terminal on the UserObject.
8. Connect the data output pin of the `Text` object to the data output pin of the UserObject.
9. Select `Flow` ⟹ `Exit UserObject`.
10. Connect the sequence output pin of the `Text` object to the sequence input pin of the `Exit UserObject`.

Now the dialog box outputs the value if `OK` (or Enter) is pressed. No value is output if `Cancel` (or Esc) is pressed.

Figure 5-4 shows the program created using these steps.

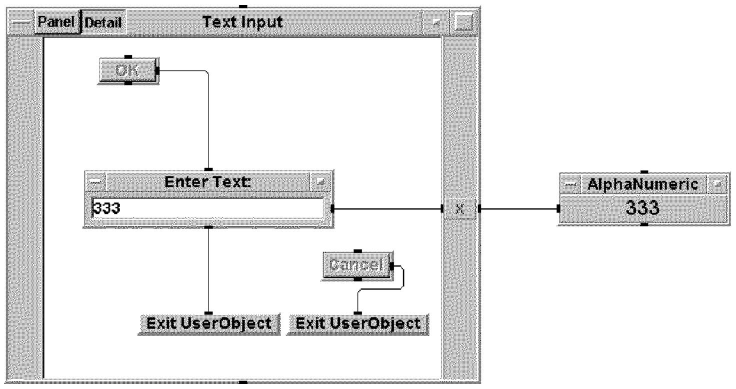


**Figure 5-4. A Basic Custom Dialog Box**

This program is saved as `dialog1.vee` in your `manual` examples directory.

**Details about this example:**

The `OK` button must be connected to the `Text` object; if it was not, the `Text` object would operate as soon as the UserObject began to run. Therefore, the default data would get propagated to the UserObject output data pin. The `OK` button prevents the `Text` object from operating until the user, after typing in the `Text` object, presses `OK`.

The threads with `OK` and `Cancel` must be connected to `Exit UserObject` objects. If they were not, the user would have to press both `OK` and `Cancel` for data to be output from the UserObject, because a UserObject does not propagate its output data until all objects inside it have finished operating.

To see these execution principles demonstrated, follow these steps:

1. Select `File` ⟹ `Edit Default Preferences`.
2. In the `Debug Animation` group, make sure that `Execution Flow` is selected.
3. Click `OK`.
4. Select `Edit` ⟹ `Animate`.
5. Show the Detail View of the UserObject.
6. Run the program.

## To Set a Timeout

If you want to have the dialog box go away after a period of time, set a timeout. Follow these steps to add a timeout to the custom dialog box created in "To Build a Basic Dialog Box" earlier in this chapter.

1. Select `Flow` ⟹ `Delay` and place it in the Detail View of the UserObject.
2. Select `Flow` ⟹ `Exit UserObject`.
3. Connect the data output pin of the `Delay` object to the sequence input pin of the `Exit UserObject`.
4. Specify the pop-up duration (in seconds) in the `Delay` edit field. When the time is up, the Pop-Up Panel is put away.
5. Add a second data output terminal to the UserObject.
6. Connect the `Delay` object's data output pin to the second data output pin.

When the timeout period expires, an empty container is output on this terminal.

Figure 5-5 shows the program created using these steps.

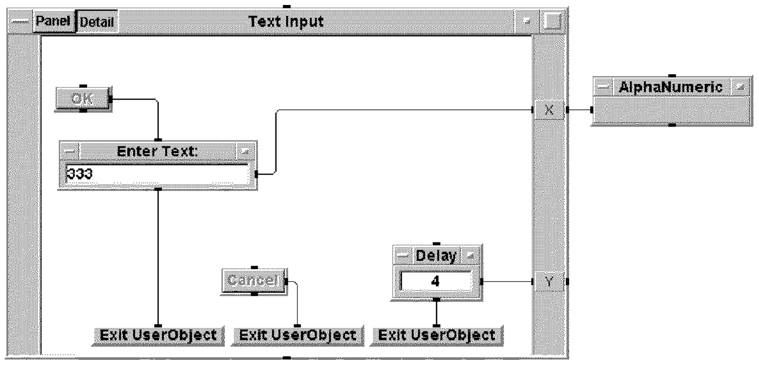


**Figure 5-5. A Dialog Box With a Timeout**

This program is saved as `dialog2.vee` in your `manual` examples directory.

When the timeout expires in `dialog2.vee`, no value is output from the UserObject. Often you want the UserObject to output the default value when it times out as shown in Figure 5-6.

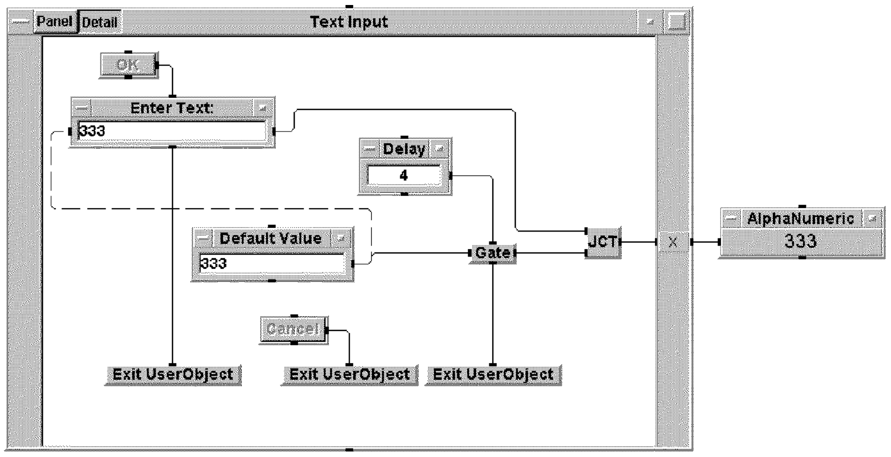


**Figure 5-6. The Default Value is Output When the Timeout Period Expires**

This program is saved as `dialog3.vee` in your `manual` examples directory.

## To Check for Valid Data

Figure 5-7 shows a program that checks the input value against some constraint, before outputting the value. If the value does not meet the constraint, another dialog box pops up to inform the user.

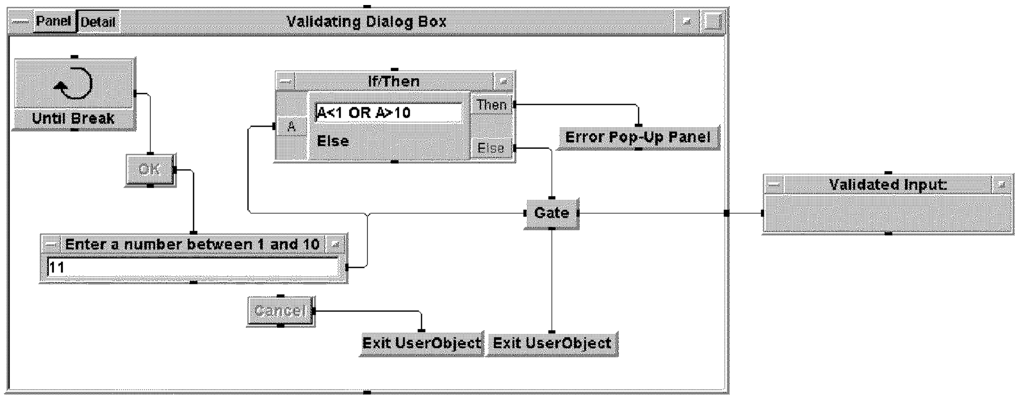


**Figure 5-7. A Dialog Box With Constraint Checking**

This program is saved as `dialog4.vee` in your `manual` examples directory.

The important aspects of this example are:

- The `Until Break` object allows you to retype a number after the `Error Pop-Up Panel` executes, or to retype a number after the `Enter a number` . . . object has already executed once.
- The `Error Pop-Up Panel` is a UserObject Pop-Up Panel that displays a message, and responds to (Enter) or (Esc). It times out after five seconds.

## To Build a Self-Modifying Panel

To have a panel present choices to the user based on previous user selections, create a "self-modifying" panel.

Figure 5-8 shows a program that changes the choices in a `Drop-Down List`, based on the selection made in a `Radio Buttons` object. The Pop-Up Panel is shown in the upper right of the figure.

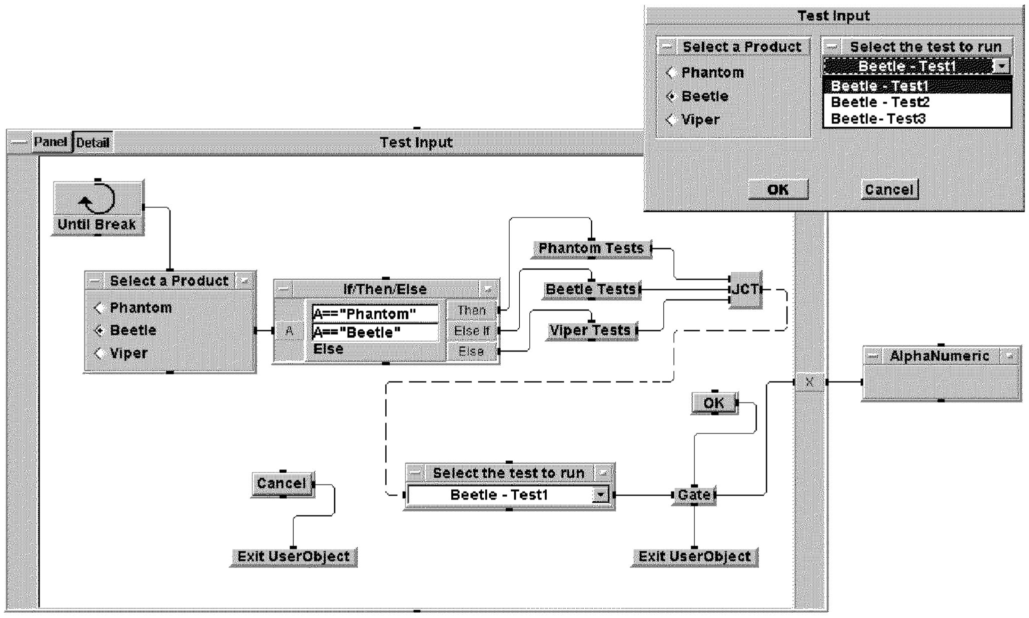


**Figure 5-8. A Self-Modifying Panel**

This program is saved as `dialog5.vee` in your `manual` examples directory.

The important aspects of this example are:

- The `Select a Product` (`Radio Buttons`) object has `Wait for Input` set on its Properties dialog box.
- `Until Break` allows the user to change choices. If `Until Break` was not there, the user could select only one product name and the thread would

execute only once. The `Until Break` loop allows the user to try various selections before finalizing the choice by pressing `OK`.

- `Phantom Tests`, `Beetle Tests`, and `Viper Tests` are `Text` arrays that contain the list of tests to input to the `Select the test to run`, `Drop-Down List` object via an `Enum Values` control input pin.

## Building a Status Panel

To display a status panel while your tests run and to update it with current test data, use a UserFunction Panel. Show it with the **`showPanel()`** object and run it with the **`Call Function`** object.

When you use **`showPanel()`**, you cannot use **`Show Panel On Execute`** (on the UserFunction Properties dialog box). Table 5-2 summarizes the differences between the way **`showPanel()`** and **`Show Panel On Execute`** execute.

**Table 5-2.**
**Differences Between `showPanel()` and `Show Panel On Execute`**

| `showPanel()` | `Show Panel On Execute` |
|---|---|
| The panel stays up until a **`hidePanel()`** object is executed or the program stops. | The panel stays up until the UserFunction is finished executing or the program stops. |
| The UserFunction is executed with **`Call Function`**, and the panel is displayed with **`showPanel()`**. Therefore, the UserFunction can be executed multiple times while the panel remains visible. | The UserFunction is executed and the panel is displayed with **`Call Function`**. Therefore, each time the UserFunction is called, the panel is automatically displayed; after the UserFunction has finished executing, the panel is automatically hidden. |
| The displayed position can be set programmatically at run time. | The displayed position is set manually at development time. |
| The displayed panel size can be different than the size of the Panel View. | The displayed panel size is the same size as the Panel View. |
| The user cannot move the panel when it's visible. | The user can move the panel when it's visible. |
| The panel can be located outside of the HP VEE window. It doesn't need to fit within the HP VEE window, and the parts outside the HP VEE window are not visible. | The panel must be located within the HP VEE window. |

## To Build a Status Panel

Build a status panel by building a UserFunction Panel as described in "To Build a UserFunction Panel" in Chapter 2. Make sure the `Show Panel On Execute` check box is not set in the UserFunction's Properties dialog box.

## To Show a Status Panel

Use the `showPanel()` object to show a status panel.

1. Select `Device` ⟹ `Panel` ⟹ `showPanel()`.

   You can select any of the of the `showPanel()` objects; the different objects show examples of the different parameters you can use.

2. Type the UserFunction name between the quotes, replacing `UFname`.

3. To specify a location, type the `x` and `y` coordinates to specify the position in the HP VEE window of the UserFunction Panel's upper-left corner. The coordinates for the upper-left corner of the HP VEE window is `0, 0`. The edit field on the `showPanel` object looks like: `showPanel("MyFunc",300,300)`.

   If you do not specify a location, the panel is displayed in the center of the screen.

4. To specify the size of the panel displayed, you must have specified the location. Specify the `width` and `height` of the panel in pixels.

   If this size is smaller than the existing UserFunction Panel View, the upper-left part of the panel is displayed. If the the size specified is bigger than the existing Panel View, the panel displayed has additional area added on its bottom right.

   If there is a background picture, it is displayed in the size specified, and is shown according to its display mode. For example, if the display mode is `Scaled`, the background image is scaled to the size specified in the `showPanel()` object.

When you specify the location and size, the edit field on the `showPanel` object looks like: `showPanel("MyFunc",300,300,100,100)`.

You can display only one panel for each UserFunction. If you execute a second `showPanel()`, the first panel is "hidden" and the panel is shown as specified in the second `showPanel()` call.

## To Hide a Status Panel

When the program finishes executing, all panels shown with `showPanel()` are no longer displayed. To hide a panel while the program is running, follow these steps:

1. Select `Device` ⟹ `Panel` ⟹ `hidePanel()`.
2. Type the UserFunction name between the quotes, replacing `UFname`.

   When this object operates, the named panel is hidden.

## To Update a Status Panel

The `showPanel()` object only shows the panel; it does not execute the UserFunction. Use the `Call Function` object (located in `Device` ⟹ `Function` ⟹ `Call`) to send new data to the status panel and update the display. You can also call the UserFunction from any expression such as in a `Formula` or a `Sequencer` object.

You can also use `showPanel()` in any expression such as in a `Formula` or a `Sequencer` object. For example, use `showPanel()` as an expression in the first transaction of a `Sequencer`. As each test runs, the data on the status panel is updated. Use `hidePanel()` in the last transaction of the `Sequencer`.

After you've sent new data via the `Call Function` object, you do not need to call `showPanel()` again; the contents of the displayed panel are updated automatically.

## A Status Panel Example

Figure 5-9 shows a simple status panel that is updated with information. After you make a choice in the `List Box` object, the data associated with that test is displayed on the panel.

Figure 5-10 shows the program that displays the status panel. This program is saved as `status.vee` in your `manual` examples directory.

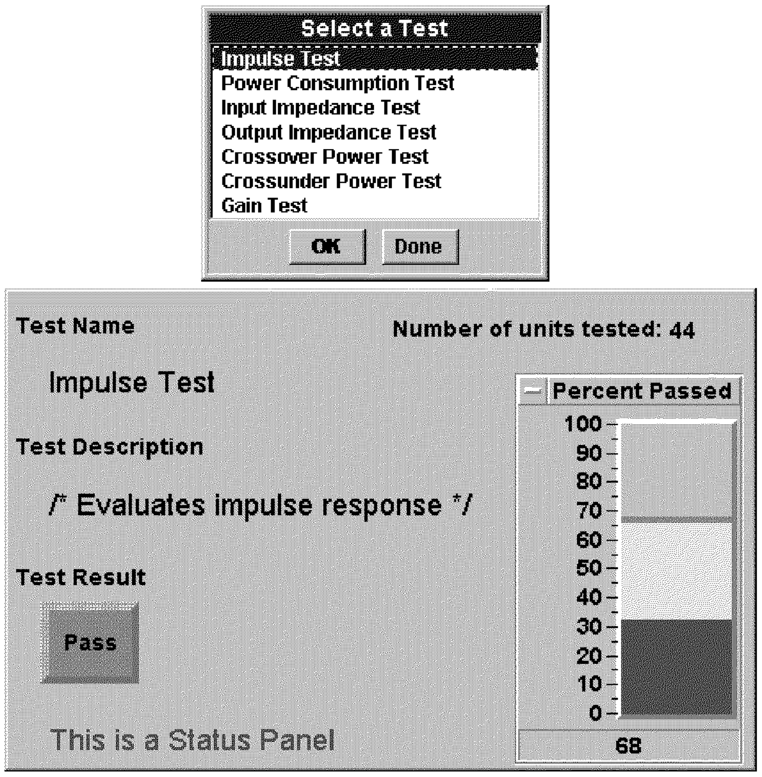


**Figure 5-9. A Status Panel**

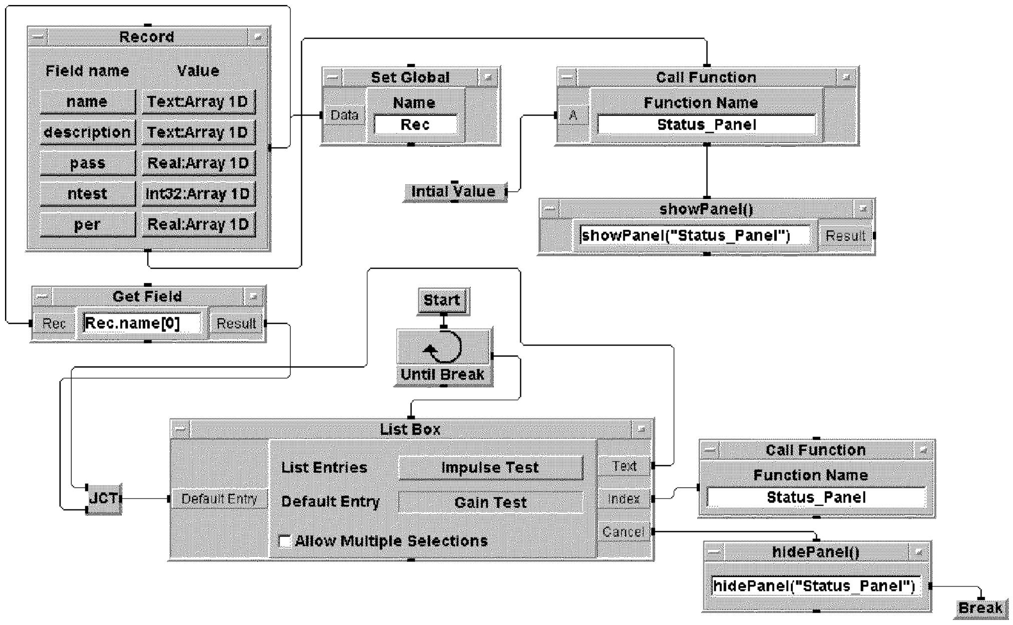


**Figure 5-10. A Status Panel Example Program**

The important aspects of this example are:

- The `Record` constant sets the record of test results. This could be the log output from a `Sequencer`.
- The `Until Break` object lets you choose the test information multiple times.
- The `Get Field` object resets the `List Box` to the value selected in the last iteration.
- When the `Done` button is pressed on the `List Box`, the status panel is hidden and the `Until Break` loop is completed.

The UserFunction contains a Panel View that is updated each time the UserFunction is called.

The `mfgtest.vee` program in the `new` examples directory demonstrates a more complete program that uses status panels in conjunction with the `Sequencer`.

## To Animate Pictures

You can use **showPanel()** and **hidePanel()** to perform animation. Follow these steps to create a simple animation program:

1. Create a Panel View by adding an object to the Panel and then deleting the object.
2. Select a background color for the Main Panel.

   This color is important, because you'll use that same color for the background of the animation file(s). The RGB values for the HP VEE colors are listed in Appendix A to help you match colors.
3. Create a graphics file of the image you want to animate.
4. Select **Display** ⟹ **Picture**. Specify your graphics file via the Properties dialog box.
5. Place the **Picture** object into a UserObject and add it to the UserObject Panel.
6. From the **Picture** object menu, select **Edit Properties**.
7. Turn off the **Picture** border and title bar, and place the **Picture** in the upper-left corner of the UserObject Panel.
8. From the UserObject object menu, select **Edit Properties**.
9. Turn off the UserObject Panel's title bar and border from the **Pop-Up Panel** group.
10. Change the Panel View background color to the Main Panel's background color.
11. Click **OK**.
12. Make the UserObject into a UserFunction.
13. Delete the **Call Function** object.
14. Select **Device** ⟹ **Panel** ⟹ **showPanel()**.
15. Type the UserFunction name between the quotes, replacing **UFname**.
16. Add two data input terminals.

17. Use the terminal names in the edit field so it looks like `showPanel("MyFunc",a,b)`.
18. Select `Flow` ⟹ `Repeat` ⟹ `For Range`. Do this twice.
19. For each `For Range` object, fill in the range of `x` or `y` coordinates.
20. Connect each `For Range` object to one of the data input pins on the `showPanel()`.
21. Press the `Panel` button to see the Panel View.
22. Run the program and see how the picture moves across your Panel View.

Figure 5-11 shows the program created using these steps.

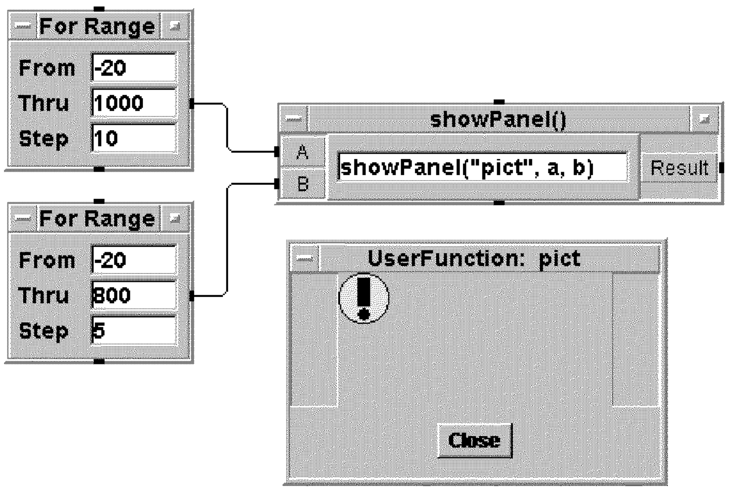


**Figure 5-11. A Simple Animation**

This program is saved as `animate.vee` in your `manual` examples directory.

Figure 5-12 shows the Panel View of a program that uses simple animation in HP VEE.

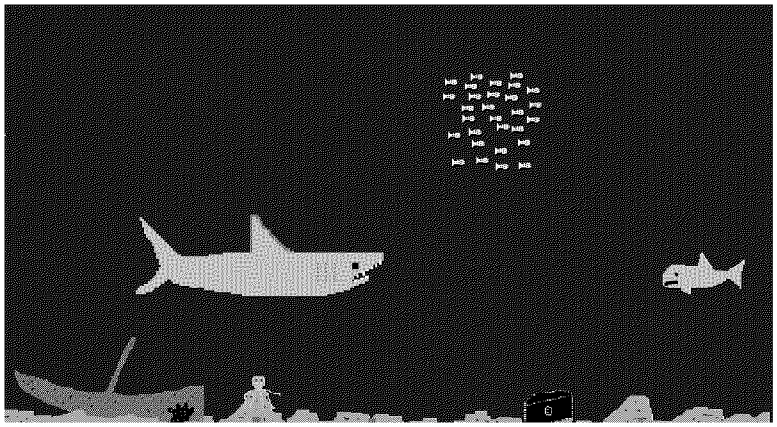


**Figure 5-12. The Ocean: A Simple Animation**

This program is saved as `ocean.vee` in the `new` examples directory. The `wheel.vee` and `slots.vee` programs in the `new` directory demonstrate more complex animation.

Using Multiple Images

You can create animations with multiple images by adding a data input terminal to the `Picture` object, and programmatically inputting the file name to the `Picture` at run time. When the `Picture` display mode is `Scaled`, images take longer to display than if `Centered` or `Actual` are selected. If you use multiple images for animation, you can decrease execution time by loading the images into HP VEE before running your program and having HP VEE cache the images into memory.

To cache the images before running, follow these steps:

1. Select `Device` $\Longrightarrow$ `UserObject`.
2. Select `Display` $\Longrightarrow$ `Picture`. Put as many `Picture` objects inside the UserObject as you have images.
3. Load each of the image files into its own `Picture` object via the `Picture` object's Properties dialog box.
4. Select `Flow` $\Longrightarrow$ `If/Then/Else` and place it outside the UserObject.

5. In the `If/Then/Else` object, type `0`.
6. From the `If/Then/Else`, delete the data input pin.
7. Connect the `Then` data output pin of the `If/Then/Else` to the sequence input pin of the `UserObject`. Now the UserObject will never execute, but when the program is loaded, all the picture images are loaded into a cache.

Figure 5-13 shows the program created using these steps.

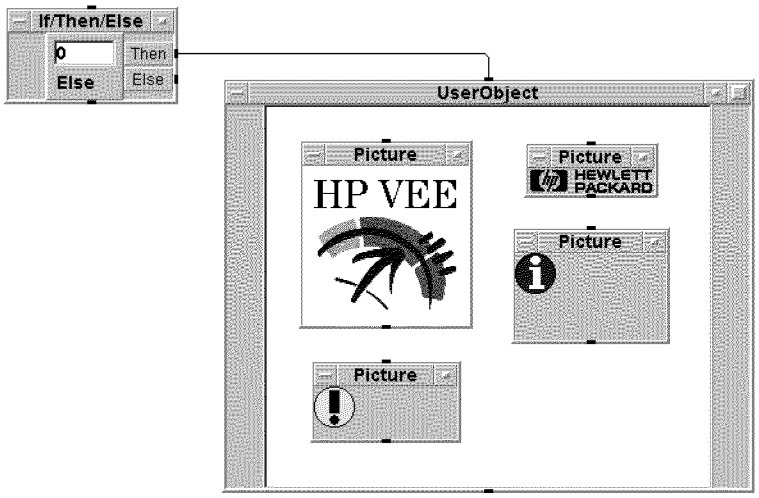


**Figure 5-13. A Picture Cache**

# 6

# Preparing a Program for Distribution to Others

# Preparing a Program for Distribution to Others

There are many considerations when delivering a program for others to use. This chapter discusses the following topics:

- Building a Panel to Run on Multiple Platforms
- Building a Panel with Non-English Text
- Making a Program Secure from User Changes
- Creating a Distribution Package

## Building a Panel to Run on Multiple Platforms

One of the features of HP VEE is the ability to create a program on one platform and deliver it for use on another platform. It's common to deliver a program to a destination system that has different hardware, operating system, or screen resolution.

## To Use Operating System-Dependent Features

The two objects `whichOS()` and `whichPlatform()` (located under `Data` ⟹ `System Info` ⟹), allow you to use operating system-dependent features such as calls to the operating system with `Execute Program`, or use DDE when running on Windows and use named pipes when running on UNIX.

Figure 6-1 shows how the `whichOS()` and `whichPlatform()` objects can be used. In this program, a different subthread is executed depending on the system (operating system or hardware platform). This program is saved as `whichos.vee` in your `concepts` examples directory.

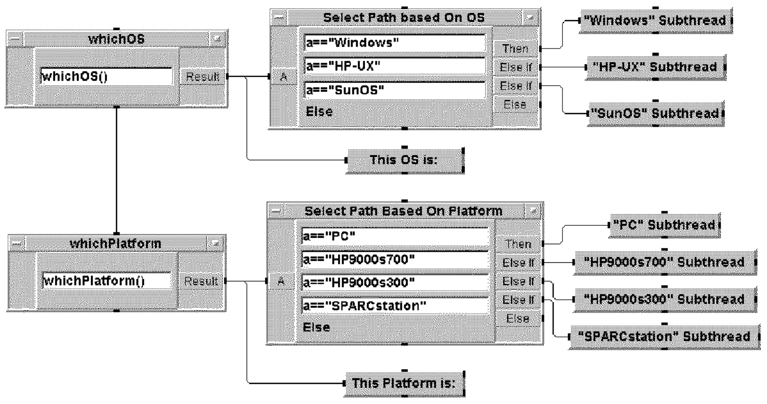


**Figure 6-1. Specifying System-Specific Threads**

You can also use `whichOS()` and `whichPlatform()` in any expression such as in a `Formula` or a `Sequencer` object.

## General Tips

- When developing your program, use a screen resolution the same as (or similar to) the destination system. The Panel View is a specific-sized window (by the number of pixels) that does not scroll. You need to make sure that the Panel View will fit on the destination system's screen.

  On UNIX, you can specify the size of the Panel View before you create it. When you start HP VEE, use the command line line option `-geometry`, to specify the size of the Detail View, and therefore specifying the size of the Panel View when it is created.
- If there are multiple destination system types, when developing your program use a system that is the "lowest common denominator" so you can get a realistic feel for the worst case for resolution (text readability), execution speed, and color use.
- Test your program on the destination system before finalizing the program.
- If developing on UNIX and delivering on Windows, remember not to assign the `F10` function key to a `Confirm (OK)` object. On Windows, `F10` is a reserved key that functions the same as (Alt).
- If transferring your program between UNIX and Windows, consider the following differences:
  - □ There are eight function keys on some keyboards and 12 on others.
  - □ The (Enter) key on Windows is the same as the (Return) on UNIX. This is important when giving instructions to your user.
  - □ There may be other keyboard differences that are important when giving instructions to your user. For example, (Esc) may be in a different location on the keyboard.
  - □ Don't use binary data files. Save data files as ASCII.

## To Choose Fonts for Multiple Platforms

Different computers have different available fonts. When creating your program, make sure to use the fonts that are on the destination platform or will translate comparably.

HP VEE uses "logical font sizes" to ensure that text will fit on an object regardless of the screen resolution. On paper, a 12-point font size is always the same height, but on a computer screen a "logical font size" (which is approximately 30-40% bigger than the physical font size) is scaled based on the screen resolution. Because HP VEE object appearance is also based on the screen resolution, the fonts fit on the objects, no matter what the screen resolution.

**NOTE**

HP VEE object size is specified by the number of pixels, so if an object is shown on a high-resolution monitor, the object appears smaller. If the same object is shown on a low-resolution monitor, it will appear bigger.

Font Type

When you load an HP VEE program onto a different system, HP VEE looks for the fonts to use by the specific font names. If it does not find a font with exactly the same name, it asks for a different font as discussed in the following sections.

Font Size

HP VEE uses the same font size as specified in the program. If the size is not available for the font type it uses the closest size, with a preference to the smaller size so that text is not clipped.

Font Style

If the font style (such as bold or italics) for the specified font type and font size is not available on the destination system, it is not used.

When Transferring a Program Between UNIX and Windows

When your program is delivered on a different operating system, fonts types get translated as shown in the following table.

| MS Windows Font | UNIX Font |
|---|---|
| *specific font name* → | ←*specific font name* |
| Arial → | ← lucida |
| Courier New → | ← courier |
| Times New Roman® → | ← times™ |
| MS Mincho (Kanji) → | ← Gothic (Kanji) |
| *all other fonts* → | lucida |
| Arial | ← *all other fonts* |

As you can see from the table above, it's best to use a font type that is common to both systems, or else use Arial, Courier New, and Times New Roman on Windows and lucida, courier, and times on UNIX. These fonts are the most common and they translate well. Note that Courier New and courier are the only non-proportional fonts in the list above.

**NOTE**

Arial and Courier New are narrower fonts than lucida and courier. So if you create your program on Windows and use Arial and Courier New, you must "over size" your objects so that when your program is run on UNIX, the text fits on the object.

Conversely, if you create your program on UNIX and use lucida and courier, when your program is loaded on Windows there will be extra space around the text.

**When Transferring a Program Between Windows Systems**

It's best to use fonts you know will be on the destination machine when creating the Panel View.

To find the fonts that are available from Windows (on the destination system), follow these steps:

1. From the **`Main`** group select **`Control Panel`**.
2. Select **`Fonts`**.
3. You'll see a list of all the fonts available on the system.
4. HP VEE uses only TrueType$^{TM}$ fonts in regular, bold, italic, and bold italic styles (styles such as half bold are not supported in HP VEE).

HP VEE supports ASCII, Symbol, and Kanji character sets.

**When Transferring a Program Between UNIX Systems**

It's best to use fonts you know will be on the destination machine when creating the Panel View.

To find the fonts that are available from UNIX (on the destination system), follow these steps:

1. Run **`xlsfonts | more`** from the command line.
2. You'll see a list with each font containing many specifications on the same line. Those with the last field of **`iso8859-`***x*, **`symbol`**, or **`japanese-15`** are fonts that HP VEE uses. Unless you're on an older version of X11 (R4 or older), all these fonts are scalable (i.e., all font sizes are available).

## To Choose Colors for Multiple Platforms

**Generally, Colors Are the Same**

Generally, there is no problem with color when transferring your program between UNIX systems or Window systems with S-VGA (800-by-600) or higher resolution. The colors on the destination system are the same as on the development system. The only differences are the monitor resolution and other monitor characteristics (such as brightness).

### When Transferring a Program to a VGA Windows System

When your development system has 256 or more colors available and then you take your program to a system with a VGA (16-color) system, all the colors will exist on the VGA system, but some will be dithered. Avoid using the colors that dither the most; they are:

- Blue Gray
- Light Blue Gray
- Medium Light Gray
- Beige
- Light Beige
- Lightest Gray
- Dark Gold

### When Transferring a Program Between UNIX Systems

There is a finite number of colors that can be displayed on a UNIX system. For example, a system could display 256 different colors at one time - the colors themselves vary as different applications request specific colors when starting, and release them when done.

When applications ask for the colors they need, there are usually enough colors to give them. But sometimes all the colors have been allocated, so the next application that asks for colors may not get the colors it wants.

A bitmap with many colors may use up large amounts of the available colors in the system.

HP VEE asks for the colors it needs. If there are not enough colors available, HP VEE checks the other applications on the system and displays an existing color that is close to the one needed.

So if there are many colorful bitmaps and color-intensive applications running on the destination system, the colors in your program may not look exactly as you specified. If you have bitmaps in your program, they may not display using all the correct colors. If this is a problem for you, you may want to shut down these applications or not display these bitmaps before running HP VEE.

## Building a Panel With Non-English Text

To build a panel with non-English text you need to have your environment set up to display non-English characters. For information on how to do this, refer to *HP VEE Advanced Programming Techniques*.

There are a few things in HP VEE that are in English and cannot change on your panel:

- Some objects have English text. For example, **`Data`** ⟹ **`Selection Control`** ⟹ **`Pop-Up List`**. When the list is popped up, the title of the pop up is **`Make Selection`**. The buttons are **`OK`** and **`Cancel`**.
- The menu choices in the Panel View are in English, as are the associated dialog boxes, such as **`Open`**, **`Save`**, and **`Save As`**.
- The object menu choices are in English.

After setting the environment to display non-English characters, you can type non-English text in HP VEE in any place you can type English text. For example, **`Labels`**, **`Constant`** objects, titles on objects, **`Confirm (OK)`** buttons, and **`Dialog Box`** objects (and the buttons on them).

## Preventing Program Changes

When delivering a program for others to run, you generally don't want to let them modify it. There are two basic ways to prevent accidental modification:

- **File** $\Longrightarrow$ **Secure** - When **Secure** is selected, the program's Detail View is deleted. The user can **Open** and run the secured version the same way as the original program, but there is no Detail View. Because there is no "Unsecure", you must save the unsecured version before securing it.
- `-r` *file name* - This command line option is used when running an HP VEE program. When HP VEE starts, it loads the program in *file name*, and runs the program. As soon as the program is done, HP VEE exits. If you use this option, you need to save your program from the Panel View so that your users see the panel as the program executes.

The table below compares the differences between using **Secure** and the **-r** command line option.

**Comparing `Secure` with `-r`**

| `Secure` | `-r` *filename* |
|---|---|
| Highest level of security. | Lower level of security. The appearance of the Detail View can be modified if the **Panel/Detail** buttons are available (see the note below). |
| The **Secured** file is a separate file from your program file, so you have two files to maintain. | The program file is run, so you have only one file to maintain. |
| The **Secured** file loads faster than the program file because it is a smaller file. | The program file loads slower than the **Secured** file because it is a larger file. |
| The user can pause, stop, and run the program. | The user does not have control over the program execution. When the program stops running, the window goes away. The user has no control over this. |
| There is no **Panel/Detail** button (the user can't access the Detail View). | The **Panel/Detail** button is available (unless **-notoolbar** is also specified - see the note on the next page). |

| | |
|---|---|
| Objects can't be moved, resized, or deleted. | Objects can't be moved, resized, or deleted on the Panel View, but from the Detail View, the following object menu choices are available: `Move`, `Size`, `Minimize`, `Help`, `Edit Properties`, and `Edit Description`. |
| There is no access to object menu choices. | There is access to object menu choices such as `Zoom`, `Auto Scale`, and `Clear Control`. |
| The `File` menu is the only menu available. | No Panel View menus are available. |
| The `File` menu choices available are: `New`, `Open`, `Save`, `Save As`, `Show Description`, `Print Screen`, and `Exit`. | No Panel View menu choices are available. |

**NOTE**

When using the `-r` command line option, you may also want to use the `-notoolbar` command line option to prevent the user from having access to the `Panel` and `Detail` buttons. This prevents the user from accessing the Detail View and making any changes.

**NOTE**

When using the `-r` command line option, if an error occurs, the user will see the program error message boxes. When the user presses OK, the error message box closes, the program stops, and the window closes.

To prevent the user from seeing error messages, use the command line option `-noerrdisp`. If the program does not run due to an error, HP VEE exits immediately without displaying an error message.

## To Prevent Program Changes

The following list contains some tips to use when getting ready to deliver your program:

- To prevent users from editing text in `Note Pad` objects, turn off editing for the `Note Pad`. From the Panel View, select the object's Properties dialog box; in the `Editing` group, unselect the `Enabled` check box.
- To prevent users from changing maximum and minimum values on `Slider`, `Meter`, `Thermometer`, `Fill Bar`, and `Tank` objects, add control pins for `Max Value` and `Min Value`.
- To prevent users from moving UserObject or UserFunction Pop-Up Panels, use a UserFunction Panel instead and display it programmatically with the `showPanel()` object. Refer to "Building a Status Panel" in Chapter 5 for details about `showPanel()`.
- To prevent users from changing the scale on HP VEE graphical `Display` objects, change the `Layout`. From the Panel View, select the object's Properties dialog box; in the `Layout` group, select `Graph Only` or `Scales`.
- To prevent any of the HP VEE default colors and fonts on the destination system from being used, save the program with all your default colors and fonts.

  1. Select `File` ⟹ `Edit Default Preferences`.
  2. Select the `Save Default Colors/Fonts with Program` check box.
  3. Click `OK`.
  4. Select `File` ⟹ `Save`.

     All the colors and fonts used in your program are now explicitly saved in your program.

## To Secure a Program

If you decide to `Secure` your program before distributing it, follow these steps:

1. Select `File` ⟹ `Save` to save any changes to the unsecured program file.

**Caution**

If you do not save these changes, the secured file will be out of synchronization with the unsecured version. You will have program changes in the secured version that are not in the unsecured version.

2. Select `File` ⟹ `Secure`.

   Now you see just the secured Panel View.

3. Select `File` ⟹ `Save` to save the secured program file.

**Caution**

There is no way to unsecure a file; you must save the unsecured version separately.

Make sure you save the secured version to a different file name than the unsecured version. For example, save the unsecured program as `prog.vee` and save the secured program as `prog.sec`.

**NOTE**

Only programs with Panel Views can be secured. If you do not have a Panel View, but have Pop-Up Panels, or you want your program to run without user intervention, you must first create a Panel View by adding an object to the panel as described in "To Add Objects to a Panel" in Chapter 1. Then delete the object from the Panel View as described in "To Delete Objects from a Panel" in Chapter 1. The Panel View will remain. Follow the steps listed above to secure the program.

## To Re-Secure a Program

When distributing a `Secured` program to your users, each time you change the original program, you must save it secured again. and redistribute it. Refer to "To Secure a Program" earlier in this chapter.

## To Secure a UserObject

When you're distributing a UserObject for others to use, you may want to `Secure` the UserObject so that only the icon view or Panel View is available to them.

1. From the UserObject's object menu, select `Secure`.
2. You'll see a dialog box prompting you to save the unsecured version of the UserObject.

   To save an unsecured version of the UserObject, type the name of the file and click on `Save, then secure`. Remember, there is no "unsecure" so you must save the unsecured version of a UserObject so you can edit it in the future.

If the program containing the UserObject has already been saved, or you've saved the UserObject via `File` ⟹ `Save Objects`, you may not want to save the UserObject again. In this case, select `Secure without saving`.

If the UserObject had a Panel View, now only the Panel View exists. If the UserObject did not have a Panel View, only the icon view exists.

3. Now save the secured UserObject.

   Select the UserObject by clicking on it (so that it has a shadow). Select `File` ⟹ `Save Objects`. Make sure you save it to a different file name than the unsecured UserObject.

4. When others want to use the secured UserObject, they select `File` ⟹ `Merge`, and then select the file name of the secured UserObject. They use the secured UserObject in the program just like any other object.

## Creating a Distribution Package

Now your program is complete. You know the colors and fonts used will work on the destination system. You've decided how to prevent users from accidentally modifying your program (`Secure` vs. `-r`).

This section explains the files you need and some methods that allow users to run your application easily.

### To Include All Needed Files

When distributing your program, you need to include your instrument configuration file. It is located in `C:\VEE\VEE.IO` on Windows and *$HOME*`/.veeio` on UNIX. Other files you must include (if you use them) are:

- Data files.
- `*.cid` files for instrument drivers you've written.
- `*.cid` files for instrument drivers not installed on the destination system.
- Graphics files you've created.
- DLLs or compiled functions that you call.

## To Create an Application Icon in Windows

It is convenient for your users to double-click on a program icon to start up HP VEE and your program. To set up the icon, follow these steps on the destination system.

1. Create a group for your application.

   From the Windows Program Manager, select `File` $\Longrightarrow$ `New`. Select `Program Group` and click `OK`.

2. Type the title of your group in the `Description` field and click `OK`. The group is now created.
3. Make sure your group is active (with the highlighted title bar).
4. From the Windows Program Manager, select `File` $\Longrightarrow$ `New`. Select `Program Item` and click `OK`.
5. Type the name of the icon into the `Description` field.
6. Type the HP VEE command line into the `Command Line` field. It could look something like `C:\VEE\VEE.EXE -r -notoolbar MY_APP.VEE`. The entire list of HP VEE command line options are listed in *How To Use HP VEE*.
7. Specify a `Working Directory`, if needed.
8. Click on `Change Icon` to select an icon. If you have a `*.ICO` file, you can select it by using `Browse`. Or you can use the HP VEE icon which is located in `C:\VEE\VEEY3.DLL`. You can also pick an icon from the set that is included by Microsoft. Click `OK`.
9. Click `OK`. The icon is displayed in the group. When a user double-clicks on it, your application will be launched.

For details about this procedure, refer to the MS Windows documentation and online help.

## To Create an Application Icon in VUE

It is convenient for your users to double-click on a program icon to start up HP VEE and your program. To set up the icon, follow these steps on the destination system:

1. Select the `Personal Toolbox` by double-clicking on the `Toolbox` on the `Front Panel`.
2. Select the `CreateAction` icon.
3. Type the name of the icon in the `Name:` field.
4. Type the HP VEE command line into the `Command Line` field. It could look something like `veetest -r -notoolbar ~/apps/myprog.vee`. The entire list of HP VEE command line options are listed in *How To Use HP VEE*.
5. Select a `Window Type` of `X Windows`. HP VEE creates its own window.
6. If you've created an icon to run your application, type the name in the `Large Icon` and/or the `Small Icon` fields. Or you can use the HP VEE icon which is located in `/usr/lib/veetest/config/vee.xpm` (or `/usr/lib/veerun/config/vee.xpm` if the destination system is using HP VEE RunOnly).
7. Click `Apply`.
8. Verify the icon that was created does what you want it to do.
9. Once the icon is verified, click `Close` in the `CreateAction` dialog box.

For details about this procedure, refer to the HP VUE documentation and online help.

A

# RGB Values for HP VEE Colors

# RGB Values for HP VEE Colors

Use these RGB values when you're making a custom bitmap and need to match an HP VEE color, for example, to match a panel background color.

These RGB values may change in future releases of HP VEE

| | | | |
|---|---|---|---|
| Warning Red | r:230 | g:30 | b:30 |
| Dark Red | r:128 | g:0 | b:0 |
| Brown | r:150 | g:110 | b:75 |
| Dark Brown | r:96 | g:64 | b:64 |
| Dark Beige | r:105 | g:95 | b:80 |
| Black | r:0 | g:0 | b:0 |
| Darkest Gray | r:96 | g:96 | b:96 |
| | | | |
| Red | r:255 | g:0 | b:0 |
| Hot Pink | r:255 | g:0 | b:128 |
| Dark Magenta | r:128 | g:0 | b:128 |
| Dark Purple | r:90 | g:0 | b:170 |
| Dark Blue | r:0 | g:0 | b:128 |
| Beige | r:148 | g:139 | b:123 |
| Dark Gray | r:128 | g:128 | b:128 |
| | | | |
| Light Red | r:255 | g:128 | b:128 |
| Pink | r:255 | g:128 | b:192 |
| Magenta | r:255 | g:0 | b:255 |
| Purple | r:128 | g:0 | b:255 |
| Med Dark Blue | r:70 | g:0 | b:240 |
| Blue | r:0 | g:0 | b:255 |
| Med Dark Gray | r:160 | g:160 | b:164 |

| | | | |
|---|---|---|---|
| `Gold` | r:176 | g:145 | b:91 |
| `Orange` | r:255 | g:128 | b:0 |
| `Light Magenta` | r:255 | g:128 | b:255 |
| `Lavender` | r:128 | g:128 | b:255 |
| `Dark Green` | r:0 | g:96 | b:0 |
| `Med Light Blue` | r:0 | g:128 | b:255 |
| `Gray` | r:192 | g:192 | b:192 |
| `Light Gold` | r:203 | g:182 | b:128 |
| `Dark Gold` | r:149 | g:108 | b:54 |
| `Green` | r:0 | g:193 | b:19 |
| `Med Dark Green` | r:0 | g:130 | b:70 |
| `Dark Teal` | r:0 | g:128 | b:128 |
| `Light Blue` | r:96 | g:192 | b:255 |
| `Med Light Gray` | r:208 | g:208 | b:208 |
| `Yellow` | r:255 | g:255 | b:0 |
| `Dark Yellow` | r:192 | g:192 | b:0 |
| `Light Teal` | r:0 | g:192 | b:192 |
| `Teal` | r:0 | g:160 | b:160 |
| `Blue Gray` | r:120 | g:184 | b:210 |
| `Light Blue Gray` | r:195 | g:211 | b:219 |
| `Light Gray` | r:224 | g:224 | b:224 |
| `Light Yellow` | r:255 | g:255 | b:128 |
| `Yellow Green` | r:128 | g:255 | b:0 |
| `Light Green` | r:128 | g:255 | b:128 |
| `Cyan` | r:0 | g:255 | b:255 |
| `Light Beige` | r:220 | g:211 | b:184 |
| `White` | r:255 | g:255 | b:255 |
| `Lightest Gray` | r:240 | g:240 | b:240 |

# Index

# Index

A
accessing
  object menus, 1-5, 3-3
**Actual**
  display mode, 1-11, 1-12
adding
  control pins to prevent changes, 6-13
  labels to panels, 1-9
  objects to nested UserObject Panels, 2-9
  objects to panels, 1-4
  pictures to panels, 1-11
  UserObject Panels to the Main Panel, 2-8
aligning objects, 1-9
Alt, 3-6, 6-4
**animate.vee** example program, 5-23
animating
  pictures, 5-22
  using multiple images, 5-24
appearance
  of object borders, 1-7
**Appearance** group properties, 1-7
ASCII data files, 6-4
assigning
  Enter to **OK**, 3-5
  Esc to **OK**, 3-5
  **F10** to **Confirm (OK)**, 6-4
  function keys to **OK**, 3-6
attention
  calling to objects, 4-11
**Auto Execute**, 3-8, 3-12

B
background
  adding pictures to panels, 1-11
  changing panel color, 2-5, 5-22
  color choices, 4-4, 4-6
**Beep**, 4-12
binary data files, 6-4
bitmap, 1-12
**Border**, 1-7
borders
  when sizing objects, 1-8
building
  custom dialog boxes, 5-8

Main Panel, 1-4
panels, 1-4
panels with non-English text, 6-10
Pop-Up Panels, 2-5
programs to run on multiple systems, 6-3
self-modifying panels, 5-14
status panels, 2-13, 5-16
UserFunction Panels, 2-12–14
UserFunction Pop-Up Panels, 2-13
UserObject Panels, 2-3–11
VUE application icon, 6-19
Windows application icon, 6-18
**Button**, 1-7

C
caching
pictures, 1-13, 5-24
**Call Function**, 5-18
calling
attention to objects, 4-11
**Centered**
display mode, 1-11, 1-12
changing
grid size, 1-10
image files, 1-13
**Label** text, 1-9
**Label** text justification, 1-9
location of Pop-Up Panels, 2-7
location of UserFunction Pop-Up Panels, 2-14
object border, 1-7
object colors on a panel, 1-6
object fonts on a panel, 1-6
panel background color, 2-5
Pop-Up Panels size, 2-7
the appearance of a panel, 1-6
UserFunction Pop-Up Panels size, 2-14
values via the keyboard , 3-11
**Check Box**, 1-7
***.cid**, 6-17
**Circular**, 1-6
**Color Alarm**, 4-12
properties, 1-6
colors
as a redundant cue, 4-4
changing on a panel, 1-6
choosing for cross-system delivery, 6-8
dithered, 6-8
keeping defaults with programs, 6-13
label, 1-9

printing, 4-5
RGB values for, A-2
selecting, 4-6
setting for **Dialog Box** object panels, 5-4
UNIX system, 6-8
used for backgrounds, 4-4, 4-6
used for text, 4-6
uses for, 4-5
using, 4-4
using default, 1-6
VGA systems, 6-8
**Colors**, 1-7
**colors.vee** example program, 4-7
command line options
**-geometry**, 6-4
**-noerrdisp**, 6-12
**-notoolbar**, 6-12
**-r**, 6-11, 6-12, 6-18, 6-19
**-toolbar**, 6-18, 6-19
**Confirm (OK)**, 6-10
assigning Enter to, 3-5
assigning Esc to, 3-5
assigning **F10** to, 6-4
assigning function keys to, 3-6
displaying Pop-Up Panels, 2-6
selecting, 3-11
tabbing to, 3-9
**Constant** objects, 6-10
changing, 3-11
constraint checking
in **Dialog Box** objects, 5-4
with custom dialog boxes, 5-13
**CreateAction**, 6-19
creating
panels, 1-4
VUE application icon, 6-19
Windows application icon, 6-18
cross-platform development, 6-3
choosing colors for, 6-8
choosing fonts for, 6-5
Ctrl+C , 3-12
custom button text
in **Dialog Box** objects, 5-4
custom dialog boxes, 2-5
building, 5-8
constraint checking with, 5-13
differences from **Dialog Box** objects, 5-3
outputting a value on timeout, 5-12
self-modifying, 5-14

  setting a timeout on, 5-10
cutting objects from a panel, 1-5
**Cyclic Button**, 1-7
  selecting, 3-11

D
data files
  binary vs. ASCII, 6-4
**Debug Animation**, 5-10
default
  colors, 1-6
  colors and fonts, 1-6
  colors and fonts saved in program, 6-13
  fonts, 1-6
  label color, 1-9
**Delete**
  accessing while program is running, 1-5
deleting
  objects from a panel, 1-5
  objects from nested UserObject Panels, 2-10
  UserObject Panels from Main Panel, 2-11
delivering a program on different systems, 6-3
Detail View
  appearance links with Panel View objects, 1-6
  contrasting with Panel View, 1-2
  switching to Panel View, 1-4
  switching to UserFunction Panel View, 2-13
  switching to UserObject Panel View, 2-4
**dialog1.vee** example program, 5-9
**dialog2.vee** example program, 5-11
**dialog3.vee** example program, 5-12
**dialog4.vee** example programs, 5-13
**dialog5.vee** example program, 5-14
**Dialog Box** objects, 4-12, 5-3, 6-10
  built-in features, 5-4
  differences from custom dialog boxes, 5-3
  focus on, 3-4
  panel properties, 5-4
  using the keyboard with, 5-5
differences between **Dialog Box** objects and custom dialog boxes, 5-3
differences between **showPanel()** and **Show Panel On Execute**, 5-16
displaying
  Pop-Up Panels, 2-6
  status panels, 5-17
  UserFunction Panels, 2-12
  UserObject Panels, 2-3
display modes, 1-11, 1-12
distribution files, 6-17
dithered colors, 6-8

**Drop-Down List**, 1-7
  selecting, 3-11

E

edit fields, 3-11, 3-12
  tabbing through, 3-4
**Editing Enabled** property, 1-7
**Edit Properties**
  accessing while program is running, 1-5
Enter, 3-11, 6-4
  assigning to **OK**, 3-5
  getting focus, 3-4
Esc, 3-11
  assigning to **OK**, 3-5
  getting focus, 3-4
example programs
  **animate.vee**, 5-23
  **colors.vee**, 4-7
  **dialog1.vee**, 5-9
  **dialog2.vee**, 5-11
  **dialog3.vee**, 5-12
  **dialog4.vee**, 5-13
  **dialog5.vee**, 5-14
  **mfgtest.vee**, 5-21
  **ocean.vee**, 5-24
  **slots.vee**, 5-24
  **status.vee**, 5-19
  **timeout.vee**, 5-6
  **wheel.vee**, 5-24
  **whichos.vee**, 6-3
**Execution Flow**, 5-10
**Exit UserObject**, 5-8

F

**F10**, 3-6, 6-4
fields
  tabbing through, 3-10
files needed, 6-17
**Fill Bar**, 6-13
  properties, 1-6
**Flat**
  border, 1-7
focus
  definition of, 3-3
  on objects, 3-4
  on **OK**, 3-4
  on Pop-Up Panels, 3-4
  tabbing to, 3-4
  when selecting, 3-11

with **`Dialog Box`** objects, 3-4
fonts
available on UNIX, 6-7
available on Windows, 6-7
changing on a panel, 1-6
choosing for cross-system delivery, 6-5
keeping defaults with programs, 6-13
logical font size, 4-8, 6-5
non-proportional, 4-8, 6-6
proportional, 4-8, 6-6
setting for **`Dialog Box`** object panels, 5-4
size, 4-8, 6-5
style, 6-6
type, 6-5
UNIX and Windows, 6-6
using, 4-8
using default, 1-6
when changing on a panel, 1-6
**`Format`** group properties, 1-7
formatting
in **`Dialog Box`** objects, 5-4
**`Formula`**, 5-18
**`Front Panel`**, 6-19
**`Function Key`** group, 3-6
function keys
assigning to **`OK`**, 3-6
getting focus, 3-4

G
**`-geometry`** command line option, 6-4
GIF, 1-12
giving
focus to objects, 3-4
graphic images
adding to a panel, 1-11
animating, 5-22
caching, 1-13, 5-24
formats supported, 1-12
graphics images
including files for distribution, 6-17
**`Graph Only`**, 1-6, 6-13
grid
setting detent on, 1-10
using to align objects, 1-9
guidelines
for user interfaces, 4-2–12

**H**

height
  matching object's, 1-11
**Height**, 1-7
**hidePanel()**, 5-18
  when animating, 5-22
hiding
  status panels, 5-18
**High Color**, 1-7
**Horizontal**, 1-6
**Horizontal Paddle**, 1-7
**Horizontal Rocker**, 1-7
**Horizontal Slide**, 1-7

**I**

*.icn, 1-12
***.ICO**, 6-18
icons
  **vee.xpm**, 6-19
  **VEEY3.DLL**, 6-18
**Icon** tab properties, 1-7
images
  adding to a panel, 1-11
  animating, 5-22
  caching, 1-13, 5-24
  formats supported, 1-12
including files for distribution, 6-17
Independent Panel
  definition of, 2-3
**Indicator** object
  properties, 1-6
**Indicator** objects, 6-13
inherating default colors and fonts, 1-6
instrument configuration file, 6-17
**iso8859-**$x$, 6-7

**J**

**japanese-15**, 6-7

**K**

keeping
  Pop-Up Panels displayed, 2-6
keyboard
  navigation, 3-8
  operations, 3-3
  selection techniques, 3-11
  using with **Dialog Box** objects, 5-5

L

**Label**, 6-10
  changing justification, 1-9
  changing text, 1-9
labeling areas, 1-9
**Label Justification** group properties, 1-6
laying out
  panels, 4-11
**Layout**
  group properties, 1-6, 6-13
**Limits** group properties, 1-7
lining up objects, 1-9
links between Panel View and Detail View objects, 1-6
**List**, 1-7
  selecting, 3-11
**List Box**, 5-19
location
  of Pop-Up Panels, 2-7
  of status panels, 5-17
  of UserFunction Pop-Up Panels, 2-14
logical font size, 4-8, 6-5
**Low Color**, 1-7

M

Main Panel
  adding objects to, 1-4
  adding UserObject Panels to, 2-8
  building, 1-4
  definition of, 1-3
  deleting UserObject Panels from, 2-11
  switching between Datail and Panel Views, 1-4
marquee
  definition of, 3-3
matching object sizes, 1-11
merging
  secured UserObjects, 6-16
**Meter**, 6-13
  properties, 1-6
**mfgtest.vee** example program, 5-21
**Mid Color**, 1-7
mono-spaced fonts, 4-8
mouseless operations, 3-3
**Move**
  accessing while program is running, 1-5
multiple fields
  selecting, 3-11
  tabbing through, 3-10

N
navigating
  via keyboard, 3-8
needed files, 6-17
nested UserObject Panels
  adding objects to, 2-9
  creating, 2-8
  definition of, 2-3
  deleting objects from, 2-10
**-noerrdisp**, 6-12
**None**
  border, 1-7
non-English text, 6-10
non-proportional fonts, 4-8, 6-6
**Note Pad**, 1-7
  disabling editing on, 6-13
**-notoolbar**, 6-12, 6-18, 6-19

O
object menu
  accessing, 3-3
  accessing while program is running, 1-5
objects
  adding to nested UserObject Panels, 2-9
  adding to panels, 1-4
  aligning, 1-9
  calling attention to, 4-11
  changing border of, 1-7
  deleting from a panel, 1-5
  deleting from nested UserObject Panels, 2-10
  giving focus to, 3-4
  matching sizes of, 1-11
  menu access, 3-3
  menu access while program is running, 1-5
  resizing, 1-9
  selecting, 1-4
  selecting with multiple fields, 3-11
  sizing with borders, 1-8
  tabbing through multiple fields, 3-10
**ocean.vee** example program, 5-24
**OK**, 6-10
  assigning Enter to, 3-5
  assigning Esc to, 3-5
  assigning function keys to, 3-6
  displaying Pop-Up Panels, 2-6
  getting focus, 3-4
  selecting, 3-11
  tabbing to, 3-9
operating system-dependent features
  using , 6-3

order of tabbing, 3-8
outputting
  a value on timeout, 5-6, 5-12

**P**

panels
  adding a label to, 1-9
  adding objects to, 1-4
  adding pictures to, 1-11
  aligning objects on, 1-9
  building UserFunction, 2-12–14
  building UserObject, 2-3–11
  changing grid size on, 1-10
  changing the appearance of, 1-6
  deleting objects from, 1-5
  layout, 4-11
  self-modifying, 5-14
  tabbing through, 3-10
  user interface guidelines, 4-2–12
Panel View
  appearance links with Detail View objects, 1-6
  available after securing, 6-14
  building, 1-4
  changing the appearance of, 1-6
  contrasting with Detail View, 1-2
  definition of, 1-2
  deleting objects from, 1-5
  switching to Detail View, 1-4
  switching to UserFunction Detail View, 2-13
  switching to UserObject Detail View, 2-4
  window size, 1-4
**Panel View Operations** group, 3-5
password masking
  in **Dialog Box** objects, 5-4
pausing a program, 1-5
**Personal Toolbox**, 6-19
picking
  colors, 4-6
**Picture**, 1-7, 4-12
pictures
  adding to panels, 1-11
  animating, 5-22
  caching, 1-13, 5-24
  formats supported, 1-12
**Pop-Up List**, 1-7
  selecting , 3-11
Pop-Up Panels
  building, 2-5, 2-13
  building a custom dialog box with, 5-8

changing location of, 2-7, 2-14
definition of, 2-3
differences from **Dialog Box** objects, 5-3
focus on, 3-4
keeping displayed, 2-6
resizing, 2-7, 2-14
tabbing through, 3-10
**Position**, 1-7
positioning
objects, 1-9
Pop-Up Panels, 2-7
status panels, 5-17
text on labels, 1-9
UserFunction Pop-Up Panels, 2-14
preventing
**Note Pad** editing, 6-13
panel moving, 6-13
program changing, 6-11, 6-13
scale changing, 6-13
printing
colors screens, 4-5
program
pausing, 1-5
stopping, 1-5
**Program Group**, 6-18
**Program Icon**, 6-18
properties
linked between Panel View and Detail View, 1-6
not linked between Panel View and Detail View, 1-6
proportional fonts, 4-8, 6-6

R
**-r**, 6-12, 6-18, 6-19
definition of, 6-11
differences from **Secure**, 6-11
**Radio Buttons**, 1-7
selecting, 3-11
**Raised**
border, 1-7
**Rectangular**, 1-6
resecuring programs, 6-15
resizing
objects, 1-9
Pop-Up Panels, 2-7
status panels , 5-17
UserFunction Pop-Up Panels, 2-14
Return, 6-4
assigning to **OK**, 3-5
same as Enter on Windows, 3-5

RGB color values, A-2

S
sans serif fonts, 4-9
**Save Default Colors/Fonts with Program**, 6-13
**Scaled**, 1-6
  display mode, 1-11, 1-12
**Scales**, 1-6, 6-13
**Scales & Sliders**, 1-6
screen resolution, 6-4
**Secure**
  definition of, 6-11
  differences from **-r**, 6-11
securing
  keep separate files, 6-14
  programs, 6-14
  UserObject Panels, 6-15
selecting
  background color, 4-4, 4-6
  background panel color, 5-22
  colors, 4-6
  font size, 4-8
  multiple field objects , 3-11
  objects, 1-4
  text color, 4-6
  via the keyboard, 3-11
**Selection Control** objects, 1-7, 3-4
  changing values on, 3-11
**Sequencer**, 5-18
serif fonts, 4-9
setting
  a picture file, 1-11
  colors and fonts for **Dialog Box** object panels, 5-4
  environment for non-English text, 6-10
  objects to a specific size, 1-11
  picture diaplay modes, 1-11, 1-12
  status panel location, 5-17
  status panel size, 5-17
  timouts, 5-10
**Show 3-D Border**, 1-6
**Show Border**, 2-5, 2-7
**Show Caption**, 1-6
**Show Digital Display**, 1-6
showing
  status panels, 5-17
**showPanel()**, 2-13, 6-13
  differences from **Show Panel On Execute**, 5-16
  when animating, 5-22
**Show Panel on Execute**, 2-5, 2-13

**Show Terminals**, 2-7, 2-14
**Show Title**, 1-7
**Show Title Bar**, 1-6, 2-5, 2-7
**Size**, 1-7
  accessing while program is running, 1-5
sizes
  of fonts, 6-5
  of objects, 6-5
  of panels , 6-4
  of Panel View, 1-4
sizing
  objects, 1-9
  objects to same size, 1-11
  Pop-Up Panels, 2-7
  UserFunction Pop-Up Panels, 2-14
**Slider**, 3-4, 6-13
  properties, 1-6
**slots.vee** example program, 5-24
Space, 3-11
**Start**
  selecting, 3-11
status panels, 2-13
  building, 5-16
  displaying, 5-17
  example of, 5-19
  hiding, 5-18
  positioning, 5-17
  sizing, 5-17
  updating, 5-18
**status.vee** example program, 5-19
stopping a program, 1-5
**Sub-Range Configuration** group properties, 1-7
**Sunken**
  border, 1-7
switching
  between Detail and Panel Views on Main Panel, 1-4
  between Detail and Panel Views on UserFunction Panel, 2-13
  between Detail and Panel Views on UserObject Panel, 2-4
**symbol**, 6-7

T tabbing
  examples of, 3-8
  **OK**, 3-9
  order, 3-8
  through edit fields, 3-4
  through multiple fields, 3-10
  through Pop-Up Panels, 3-10
  through UserObject Panels, 3-10

  to give focus, 3-4
**Tank**, 6-13
  properties, 1-6
text
  color choices, 4-6
text justification
  on labels, 1-9
**Thermometer**, 6-13
  properties, 1-6
**Tiled**
  display mode, 1-11, 1-12
timeout
  in **Dialog Box** objects, 5-4
  outputting a value, 5-6, 5-12
  setting on a custom dialog box, 5-10
**timeout.vee** example program, 5-6
**Toggle Control** objects, 1-6, 1-7, 3-4
  changing state of, 3-11
**Toolbox**, 6-19
**Traces & Scales**, 1-6
transferring programs
  between UNIX and Windows, 6-6
  between UNIX systems, 6-7, 6-8
  between Windows systems, 6-7
  to VGA systems, 6-8
TrueType fonts, 6-7

U UNIX
  available fonts, 6-7
  choosing colors for , 6-8
  choosing fonts for, 6-6
  command line, 6-19
  **-geometry**, 6-4
  graphic formats supported on, 1-12
  instrument configuration file, 6-17
  keyboard differences from Windows, 6-4
  Return, 6-4
  VUE application icon, 6-19
updating
  status panels, 5-18
usability guidelines, 4-2–12
UserFunction Panels
  building, 2-12–14
  building Pop-Up, 2-13
  building status panels with, 5-16
  definition of, 1-3, 2-12
  displaying, 2-12
  switching between Detail and Panel Views, 2-13

UserFunction Pop-Up Panels
  building custom dialog boxes with , 5-8
  changing location of, 2-14
  focus on, 3-4
  preventing moving, 6-13
  resizing, 2-14
  tabbing through, 3-10
user interface guidelines, 4-2–12
UserObject Panels
  adding objects to nested, 2-9
  adding to the Main Panel, 2-8
  building, 2-3–11
  building Pop-Up, 2-5
  definition of, 1-3, 2-3
  deleting from Main Panel, 2-11
  deleting objects from nested, 2-10
  displaying, 2-3
  merging, 6-16
  nesting, 2-8
  securing, 6-15
  switching between Detail and Panel Views, 2-4
  tabbing through, 3-10
UserObject Pop-Up Panels
  building custom dialog boxes with, 5-8
  focus on, 3-4
  preventing moving, 6-13
  tabbing through, 3-10
using
  colors, 4-4
  **Dialog Box** objects, 5-3
  fonts, 4-8
  operation system-dependent features, 6-3

V
**.veeio**, 6-17
**VEE.IO**, 6-17
**vee.xpm**, 6-19
**VEEY3.DLL**, 6-18
**Vertical**, 1-6
**Vertical Paddle**, 1-7
**Vertical Rocker**, 1-7
**Vertical Slide**, 1-7

W **Wait for Input**, 3-4, 3-12, 5-14
**wheel.vee** example program, 5-24
**whichOS()**, 6-3
**whichos.vee** example program, 6-3
**whichPlatform()**, 6-3
width
  matching object's, 1-11
**Width**, 1-7
Windows
  application icon, 6-18
  available fonts, 6-7
  choosing fonts for, 6-6
  command line, 6-18
  Enter, 6-4
  graphic formats supported on, 1-12
  instrument configuration file, 6-17
  keyboard differences from UNIX, 6-4
  VGA systems, 6-8
window size
  of Panel View, 1-4

X **xlsfonts**, 6-7
*.xpm, 1-12
*.xwd, 1-12