BPLUS (5): Quick Tutor -- Widget & Attribute Summary

v2.4 / 5 of 18 / 01 feb 99 / greg goebel / public domain / bp05_qt4

* Now that we know what widgets are all about, it's time for a tour of the entire set. This section provides short summaries of later chapters; if you want to learn more about a widget, please consult the appropriate chapter. (Some of these widgets were adequately introduced earlier, but for thoroughness I will review the complete set.)

The first section of this chapter reviews the widgets; the second section outlines attributes which are commonly used with all or nearly all dialogs and widgets. Attributes specific to the individual kinds of widgets are discussed in the chapters that describe the widgets.

* Contents:

[1] A WIDGET SUMMARY

* The widgets supported by BPlus are listed below.

* PANEL & PANEL SEPARATOR:

The PANEL is just a simple rectangle that acts as a template to hang other widgets on. ASSIGNing a PANEL SEPARATOR to a PANEL allows you to draw a line across a PANEL, and you can set the ORIENTATION of this line to HORIZONTAL or VERTICAL.

Advanced features added in BPlus 2.0 allow the PANEL to automatically resize its child widgets when the PANEL is modified; to allow scrolling of a user interface that is too big to fit into the PANEL; and to allow "tiling" (like MS-Windows) of a background bitmap pattern.

* LABEL:

The LABEL allows you to display a VALUE of some sort -- either a number, a complex number, or a string, it's not picky about its inputs:

```
ASSIGN @Lbl TO WIDGET "LABEL";PARENT @Main
CONTROL @Lbl;SET ("VALUE":someval)
```

You can also set it to display a given number of ROWs and COLUMNs of text:

```
CONTROL @Lbl;SET ("ROWS":3,"COLUMNS":15)
```

* PUSHBUTTON, TOGGLEBUTTON, RADIOBUTTON:

The PUSHBUTTON is a simple input device, which allows a mouse click to cause an "ACTIVATED" event to control program flow.

```
ASSIGN @Btn TO WIDGET "PUSHBUTTON";PARENT @Main,SET ("LABEL":"Push Me")
ON EVENT @Btn,"ACTIVATED" GOSUB Handler
```

BPlus 2.0 added a useful feature to the PUSHBUTTON: you can give it a string array as a LABEL value, and then the user will cycle through the LABELs as it is clicked with the mouse:

```
DIM L$(0:2)[16]
DATA "Code Red","Code Yellow","Code Green"
READ L$(*)
CONTROL @Btn;SET ("LABELS":L$(*),"STATES":2)
```

The TOGGLEBUTTON is similar to the PUSHBUTTON, except that it has a VALUE attribute associated with it; each time you click on the TOGGLEBUTTON, the VALUE toggles between 0 and 1 (0 is the default). The TOGGLEBUTTON also has an indicator on it, a little box, that switches from clear when the VALUE is 0 to black when the VALUE is 1. Like the PUSHBUTTON widget, the TOGGLEBUTTON has a single event, but it is named CHANGED, not ACTIVATED:

```
  ASSIGN @Toggle TO WIDGET "TOGGLEBUTTON";PARENT @Main
  CONTROL @Toggle;SET ("LABEL":"Click Me")
  ON EVENT @Toggle,"CHANGED" GOSUB Handler
  ...
Handler: STATUS @Toggle;RETURN ("VALUE":Tval)
  DISP Tval
  RETURN
```

The TOGGLEBUTTON is useful as an on-off switch.

The RADIOBUTTON functions identically to the TOGGLEBUTTON, with two exceptions: the indicator is a diamond, not a little box; and if you set up a tab group of such widgets:

```
ASSIGN @Radio1 TO WIDGET "RADIOBUTTON";PARENT @Main
ASSIGN @Radio2 TO WIDGET "RADIOBUTTON";PARENT @Main,SET ("TAB STOP":0)
ASSIGN @Radio3 TO WIDGET "RADIOBUTTON";PARENT @Main,SET ("TAB STOP":0)
ASSIGN @Radio4 TO WIDGET "RADIOBUTTON";PARENT @Main,SET ("TAB STOP":0)
```

-- then you will only be able to set the value of *one* of these RADIOBUTTONs in this tab group to 1 at a time; setting one clears any other that is set. (You will get a CHANGED event for both the RADIOBUTTON that is set and the RADIOBUTTON that is cleared.) The RADIOBUTTON is useful for selecting from a list of mutually-exclusive choices.

* NUMBER, KEYPAD:

The NUMBER widget is very similar to a STRING widget, except that it only allows the user to input a numeric value. You can specify:

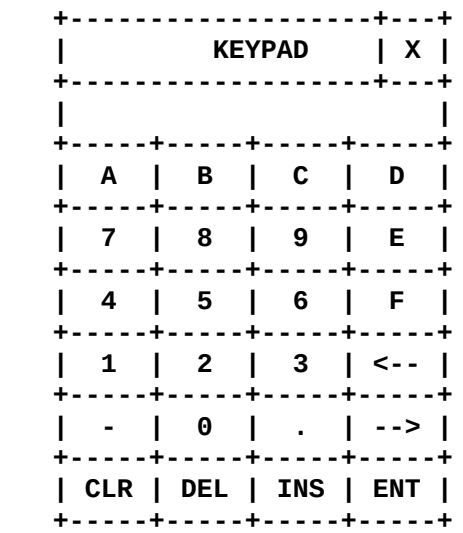
- The numeric FORMAT that it will accept (REAL, SHORT INTEGER, LONG INTEGER, BINARY, OCTAL, or HEX).
- The REAL NOTATION for REAL inputs (FIXED, SCIENTIFIC, or ENGINEERING).
- The REAL RESOLUTION of floating-point inputs.
- The MINIMUM or MAXIMUM allowed inputs.
- The round-off value of the input (INCREMENT).

Like the STRING widget, you can trap DONE, KEYSTROKE, and RETURN events; you can also trap an INVALID NUMBER event for a bogus user input.

The KEYPAD widget has the same functionality as the NUMBER widget, but displays a numeric keypad. For example, executing:

```
ASSIGN @Key TO WIDGET "KEYPAD"
```

-- yields the widget:



* STRING:

The STRING input widget puts up a field into which the user can type a text string; you can set an event on each key (KEYSTROKE), when the user presses the Enter key (RETURN), or when the user moves focus away from the STRING widget (DONE).

The STRING widget was greatly enhanced in the second version of BPlus to provide advanced attributes that allow the STRING widget to be used to build simple text-editing tools, with features such as cutting and pasting, text searching, and file-I/O. These features are too complicated to discuss in this short space, however, and are detailed in a later chapter.

* COMBO:

The COMBO widget is similar to the COMBO dialog; it combines the features of the STRING widget with the ability to enter from a predefined list:

```
   ASSIGN @Combo TO WIDGET "COMBO":SET ("ITEMS":L$(*))
```

You can use a SELECTION event to trap a selection from the list and use the SELECTION attribute to get the list index. You can also use a RETURN event to trap an entry in the string-input part of the widget, and then use the TEXT attribute to read the data.

```
   ON EVENT @Combo,"SELECTION" GOSUB Handler
   ON EVENT @Combo,"RETURN" GOSUB Handler
   ...
 Handler: STATUS @Combo;RETURN ("TEXT":T$)
   DISP T$
   RETURN
```

* LIST:

The LIST widget acts similarly to the LIST dialog: it allows you to display a list of ITEMS (derived from a string array) and then click on an entry in the list to get back the list item. As with the LIST dialog, you can also set a MULTISELECT attribute, in which case the LIST returns values of 0 or 1 to a numeric array that matches the string array, with the 1s matching the items that are set.

For example, if you execute:

```
   DIM L$(0:4)[32]
   ASSIGN @List TO WIDGET "LIST";SET ("COLUMNS":8,"ROWS":3)
   DATA "ITEM 1","ITEM 2","ITEM 3","ITEM 4","ITEM 5"
   READ L$(*)
   CONTROL @List;SET ("ITEMS":L$(*))
   ON EVENT @List,"SELECTION" GOSUB Handler
```

-- you get the widget:

```
+----+---+
|  LI| X |
+----+---+
| ITEM 1 |
| ITEM 2 |
| ITEM 3 |
| ITEM 4 |
| ITEM 5 |
+--------+
```

If your "Handler" routine has the statements:

```
STATUS @List;RETURN ("SELECTION":Nval)
DISP Nval
```

-- then if you click on "ITEM 2", you would get a "1". (Clicking on "ITEM 1" gives a "0".) If you had set:

```
CONTROL @List;SET ("MULTISELECT":1)
```

-- then you would change these "Handler" statements to:

```
STATUS @List;RETURN ("SELECTION":A(*))
DISP A(*)
```

Then clicking on "ITEM 2" would give:

```
0       1       0      0      0
```

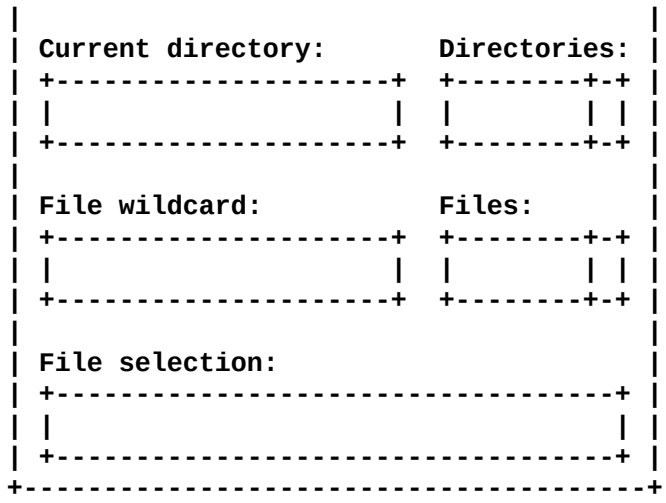
* FILE:

The FILE widget is very similar to the FILE dialog; if you execute:

```
ASSIGN @F TO WIDGET "FILE"
```

-- you get the widget:

```
+---+------------------------------+---+
| = |       File Widget Demo       | X |
+---+------------------------------+---+
```



As with the FILE dialog, there are SELECTION, PATTERN, and DIRECTORY attributes to allow you to set the file, wildcard pattern, and directory. There is also a SELECTION event that occurs when a file has been selected.

* PRINTER:

The PRINTER widget simply provides a printing output device; you can list a line of text on the PRINTER widget with the APPEND TEXT attribute:

```
CONTROL @Prn;SET ("APPEND TEXT":Str$)
```

You can also dump an entire array of strings at one time using APPEND TEXT.

You can print a blank line by using APPEND TEXT with a null string. You can clear the PRINTER widget and start displaying new text by using the TEXT attribute instead of the APPEND TEXT attribute.

Other attributes were added in BPlus 2.0 to allow allow selection of any line in the PRINTER display (CURRENT LINE); to highlight the current line and allow it to be set with a mouse (HIGHLIGHT CURRENT); to read the current line and lines following into a string or string array (CURRENT TEXT); and to INSERT LINES or DELETE LINES relative to the current line.

As noted earlier, you cannot perform a PRINTER IS to a PRINTER widget.

* HPGL VIEW, BITMAP:

The HPGL VIEW widget accepts a text file full of HPGL plotter commands and displays the plot. You give it the file name using the HPGL FILE attribute:

```
CONTROL @Graf;SET ("HPGL FILE":"Picture")
```

You can set a RETAIN RASTER attribute to keep the HPGL VIEW widget from completely redrawing its graphics every time you disturb it; when RETAIN RASTER is set, BPlus takes a "snapshot" of the graphics and retains it for repainting later.

Note that only the most fundamental HPGL commands are implemented -- for example, area fills don't work. Also note that you cannot use an HPGL VIEW widget as the PLOTTER IS device.

The BITMAP widget similarly allows you to read and display either an X11 .XWD or MS-Windows .BMP bitmapped-graphics file:

```
ASSIGN @Bitz TO WIDGET "BITMAP";SET ("AUTO SIZE":1)
CONTROL @Bitz;SET ("BITMAP FILE":"CANDY.BMP")
```

The AUTO SIZE attribute specifies that the widget resize itself to the bitmap. Note that you can trap a MOUSE CLICKED event from the BITMAP widget; this event is caused by clicking on a pixel in the bitmap. You can then determine the coordinates of the pixel with the MOUSE CLICK attribute.

The BITMAP widget can also be used to grab a BPlus user interface image, or a portion of one, and store it in a bitmap graphics file.

* SLIDER, SCROLLBAR:

The SLIDER is a numeric-input widget; create a SLIDER:

```
  ASSIGN @Slider TO WIDGET "SLIDER";PARENT @Main
  ON EVENT @Slider,"DONE" GOSUB Handler
  ...
Handler: STATUS @Slider;RETURN ("VALUE":Slideval)
  DISP Slideval
  RETURN
```

-- and you get a device that looks like the slider on a piece of audio equipment:

```
+-----+---+
|  SLI| X |
+-----+---+
| [ 100 ] |
|    ^    |
|    |    |
|    |    |
| [ 35  ] |
|    |    |
|    v    |
| [  1  ] |
+---------+
```

If you click on the arrows at the ends, the slidebar moves in that direction by an increment of 1; if you click on the trough between the slidebar and an arrow, the slidebar will be moved in that direction by an increment of 10. These define the MINOR INCREMENT and MAJOR INCREMENT attributes, and you can change them if you like; you can also set an attribute called DIRECT MOVE that disables MAJOR INCREMENT, and has the effect that if you click on the trough, the slidebar will move immediately to that point.

You can also change the MINIMUM and MAXIMUM limits, set the SLIDER to a LOGARITHMIC-scaled operation, and change the ORIENTATION from the default VERTICAL to HORIZONTAL -- as well as make many other cosmetic changes.

The SCROLLBAR is a stripped-down version of the SLIDER; if you execute:

```
ASSIGN @Scroll TO WIDGET "SCROLLBAR";PARENT @Main
CONTROL @Scroll;SET ("ORIENTATION":"HORIZONTAL")
```

-- gives a device of the form:

```
+----------------------+
|<   [ ]              >|
+----------------------+
```

This works exactly the same as the SLIDER, except it lacks limits and VALUE boxes, and cannot be set to LOGARITHMIC mode. (When vertical, the MAXIMUM limit is at the bottom, not the top.)

* CLOCK:

The CLOCK widget displays a time-of-day clock whose TYPE can be set to ANALOG or DIGITAL or MIXED (both). You can set an ALARM TIME and then trap an ALARM event when the CLOCK reaches that time.

You can also set the CLOCK TYPE to a TIMER mode, and operate it in an UP or DOWN counting mode (as set by the TIMER DIRECTION attribute), and in a cyclical mode (as set by the TIMER REPEAT); you can also trap a TIMER event in this mode. (Note that in TIMER mode you only get a digital display.)

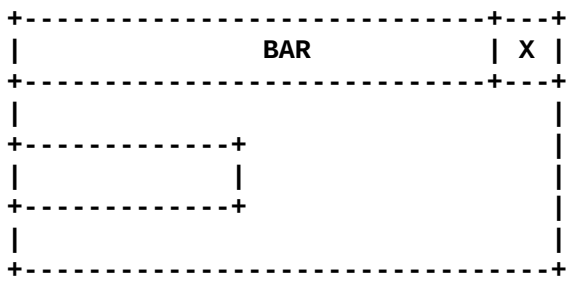
* BAR:

The BAR widget defines a bar indicator. You can set the MAXIMUM and MININUM of the bar, and set upper and lower thresholds; as you write values to the BAR, it will move up and down. These statements construct a BAR widget that travels from 1 to 100:

```
ASSIGN @Bar TO WIDGET "BAR";SET ("ORIENTATION":"HORIZONTAL")
FOR N=1 TO 100
  CONTROL @Bar;SET ("VALUE":N)
```

```
NEXT N
```

This gives:



Note that the BAR only moves when you SET a VALUE to it; it will *not* operate on its own.

The BAR can be divided into three ranges -- LOW, MIDDLE, and HIGH -- by specifying LOW and HIGH LIMITs:

```
CONTROL @Bar;SET ("LOW LIMIT":10,"HIGH LIMIT":90)
```

Then you can set an alarm on one or more of these ALARM RANGES:

```
CONTROL @Bar;SET ("ALARM RANGES":"LOW,HIGH")
```

By default, you'll get a BEEP every time the VALUE falls below the LOW LIMIT or above the HIGH LIMIT. You can change the BEEP to an event trap with:

```
CONTROL @Bar;SET ("ALARM TYPE":"EVENT")
ON EVENT @Bar,"ALARM" GOSUB Handler
```

Other BAR attributes allow you to change the MINIMUM and MAXIMUM limits, and set BAR background color, and the color of the LOW, MIDDLE, and HIGH ranges. You can also set the ORIENTATION to VERTICAL or HORIZONTAL, as shown above.

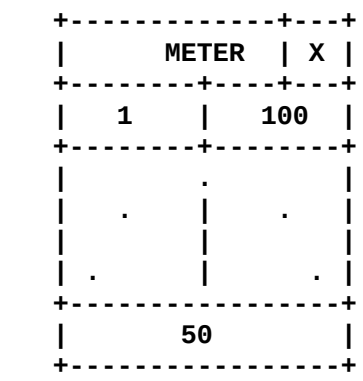
* METER:

The METER widget defines a moving-needle indicator; it it is programmed identically to the BAR widget, though it has a very different appearance and several additional features. If you create a METER widget:

```
ASSIGN @M TO WIDGET "METER"
FOR N=1 TO 100
```

```
  CONTROL @M;SET ("VALUE":N)
NEXT N
```

-- you get the widget:



Note that the METER widget, unlike the BAR widget, displays its MINIMUM and MAXIMUM limits, and well as its current VALUE.

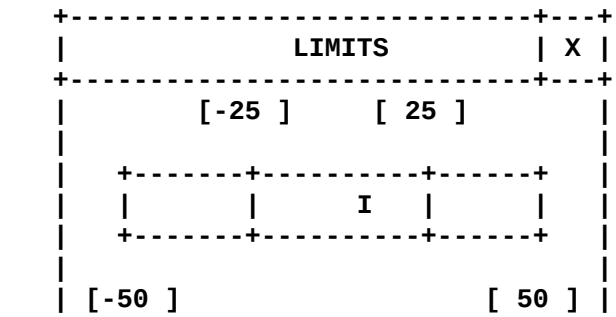
All attributes relevant to the BAR widget operate on the METER widget, and event handling is the same. However, the METER widget has other attributes to control its appearance, such as the SWEEP ANGLE (you can set it up to 360 degrees), the thickness of the needle and the arc, the background colors and font colors of the limits and VALUE boxes, and so on.

* LIMITS:

The LIMITS widget is similar to the BAR and METER widgets; it's something like a METER that's been bent flat. While a BAR widget indicates a changing value by changing the length of a bar, a LIMITS widget indicates a changing value by moving a "marker" through a fixed range. For example:

```
ASSIGN @Lim TO WIDGET "LIMITS"
CONTROL @Lim;SET ("MINIMUM":-50,"MAXIMUM":50)
CONTROL @Lim;SET ("LOW LIMIT":25,"HIGH LIMIT":25)
```

-- gives a LIMITS widget of the form:



```
+-------------------------------+
```

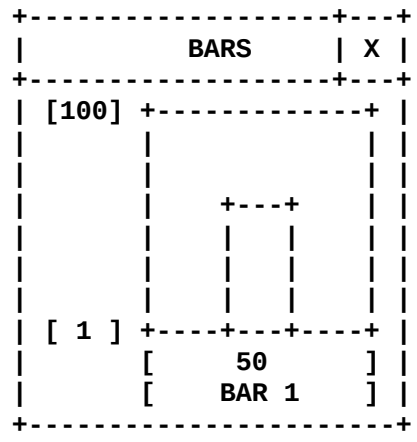
The LIMITS widget uses similar attributes to the BAR and METER widgets.

* BARS:

The BARS widget is an extension of the BAR widget; if you simply create a BARS widget:

```
ASSIGN @Bars TO WIDGET "BARS";SET ("VALUE":50)
```

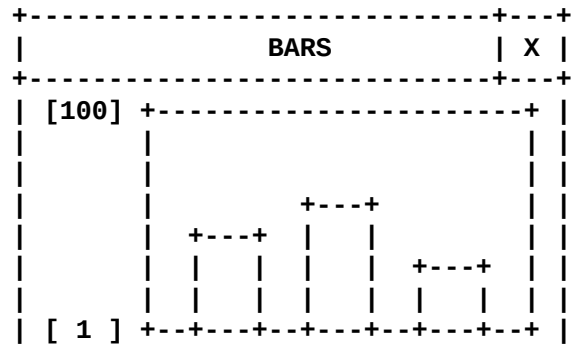
-- you get a single bar indicator, but with limits, value, and label boxes attached, similar to the METER widget:



However, you can add more bars to the widget simply by setting the BAR COUNT attribute to the desired number of bars:

```
ASSIGN @Bars TO WIDGET "BARS";SET ("BAR COUNT":3,"VALUES":Barval(*))
```

-- which gives:



```
|         [ 50  ][ 70  ][ 35  ]   |
|         [BAR 1][BAR 2][BAR 3]   |
+---------------------------------+
```

Now all 3 bars can be updated at once using the VALUES (not VALUE!) attribute with an array.

Some attributes of the BARS widget affect the entire widget, such as MAXIMUM and MINIMUM; a few can be set for an individual bar. You select an individual bar by using the CURRENT BAR attribute; you can then set parameters like the HIGH and LOW alarm ranges and the colors of the bar in the specified ranges.

You can set an ALARM event for the entire widget; then any bar value that falls into the specified range will trip an event.

* STRIPCHART, XYGRAPH:

The STRIPCHART widget provides a scrolling graphics display that can display up to 100 separate traces. This is one of the most complicated BPlus widgets, and summarizing its operation is difficult.

The STRIPCHART widget resembles a plotter display; it has a area in which to draw traces, plus numbering and tick-mark boxes along the X and Y axes. The STRIPCHART has three levels of attributes: those that let you to configure the overall operation of the widget; those that let you to configure the appearance of the X and Y axes boxes of the STRIPCHART, and those that let you to change colors (and other features) of each trace of the STRIPCHART.

The overall ("global") attributes allow you to set the ORIENTATION (VERTICAL or HORIZONTAL) of the STRIPCHART, the background color of the trace display, the number of traces (TRACE COUNT), and some other features.

You can set the global attribute CURRENT AXIS to select the X or Y axis and then use other attributes to set up an ORIGIN, RANGE, AXIS LABEL, and the numeric formats for that axis. If you don't want to mess with this stuff, you can set the axis to AUTOSCALE and not worry about the scaling, or use other attributes to turn off the numbering and tick mark boxes entirely.

You can then set the global attribute CURRENT TRACE to pick one of your traces and use other attributes to assign that trace a TRACE PEN color and a TRACE LABEL (the TRACE LABELs then appear at the bottom of the widget in their appropriate TRACE PEN color).

Once you have configured the appearance of the widget, you can (assuming that you have set the SHARED X attribute to 1) then update all the traces in parallel by writing an X value (using the global POINT LOCATION attribute) and then an array of Y values (using the global VALUES attribute); as you increment the POINT LOCATION, the STRIPCHART trace display scrolls left.

The XYGRAPH widget allows you to display the graphs of up to 100 different values simultaneously. Configuring the XYGRAPH is almost identical to configuring the STRIPCHART and the two have an almost identical appearance; however, loading data into the XYGRAPH is entirely different from loading it into the STRIPCHART.

In the XYGRAPH, you display a data set by setting a CURRENT TRACE, set the X DATA attribute

with an array that contains the x data for the trace, and then set the Y DATA attribute with an array that contains the matching y data for the trace:

```
CONTROL @Graf;SET ("CURRENT TRACE":3,"X DATA":X3(*),"Y DATA":Y3(*))
```

The trace is then drawn. You can also set a SHARED X attribute to allow a single x-data array to be used for all y-data arrays.

BPlus version 2 added two features to the STRIPCHART and XYGRAPH: it allows you to set a background graticule for the plotting area (a minor feature but one whose lack led to complaints in the first version); and the ability to place "markers" on traces that the user could then move with a mouse to determine trace values or differences between trace values.

* PULLDOWN MENU:

There are five widgets in BPlus used to create pulldown menu systems: PULLDOWN MENU, CASCADE MENU, MENU BUTTON, MENU TOGGLE, and MENU SEPARATOR.

When you create a PULLDOWN MENU widget, using a level-0 PANEL as a parent, you get a pulldown menu bar across the PANEL below the PANEL's title bar; the LABEL for the PULLDOWN MENU widget will appear in the bar.

The PULLDOWN MENU provides an "attachment point" for child widgets of the other MENU widget types; as you create instances of the other MENU widget types with a specific PULLDOWN MENU as a parent, the other MENU widget types build up a pulldown menu panel.

As you create other PULLDOWN MENU widgets, using the PANEL as a parent, their labels appear across the menu bar; they provide "attachment points" to other pulldown menus.

* CASCADE MENU:

This widget is very similar to the PULLDOWN MENU widget, except that it is created as a child of a PULLDOWN MENU widget to provide a second-level menu. Like the PULLDOWN MENU widget, it provides an "attachment point" for other menu widgets.

You can also create CASCADE MENU widgets as children of other CASCADE MENU widgets, allow you to build a hierarchical tree of menus.

* MENU BUTTON:

This widget is created as the child of PULLDOWN or CASCADE MENU widgets, and creates an entry in a pulldown menu. It acts like a PUSHBUTTON in that you can set an ACTIVATED event on it to execute the desired menu action.

* MENU TOGGLE:

This widget is identical to the MENU BUTTON widget, except that it has a VALUE attribute that

toggles between 0 and 1 every time it is selected. (A "*" also marks the menu entry when the MENU TOGGLE is set to 1.) The event in this case is named CHANGED.

* MENU SEPARATOR:

All this widget does is put a horizontal line in a pulldown menu to separate the different menu entries.

For an example of building a pulldown menu system, the following statements;

```
ASSIGN @Pm TO WIDGET "PULLDOWN MENU";PARENT @Main,SET ("LABEL":"Menu")
ASSIGN @Mb1 TO WIDGET "MENU BUTTON";PARENT @Pm,SET ("LABEL":"Button 1")
ASSIGN @Tb1 TO WIDGET "MENU TOGGLE";PARENT @Pm,SET ("LABEL":"Toggle 1")
ASSIGN @Cm TO WIDGET "CASCADE MENU";PARENT @Pm,SET ("LABEL":"Cascade")
ASSIGN @Mb2 TO WIDGET "MENU BUTTON";PARENT @Cm,SET ("LABEL":"Button 2")
ASSIGN @Tb2 TO WIDGET "MENU TOGGLE";PARENT @Cm,SET ("LABEL":"Toggle 2")
!
ON EVENT @Mb1,"ACTIVATED" GOSUB Handle_btn_1
ON EVENT @Tb1,"CHANGED" GOSUB Handle_togl_1
ON EVENT @Mb2,"ACTIVATED" GOSUB Handle_btn_2
ON EVENT @Tb2,"CHANGED" GOSUB Handle_togl_2
```

-- give the menu system:

```
+------------------------------
| Menu
+------------+-----------------
| Button 1   |
| Toggle 1   |
| ---------- +----------+
| Cascade  > | Button 2 |
+------------+ Toggle 2 |
|            +----------+
```

Note that the organization of the pulldown menu system arises from the sequence in which the widgets are created in the program.

* SYSTEM WIDGET:

One of the interesting additions BPlus in its second edition was the so-called SYSTEM widget. This was developed for use with the Screen Builder application. The Screen Builder doesn't generate code; it instead generates a file that describes the panel you have generated with it. The SYSTEM widget allows you to connect this file to a program to access its widgets:

```
ASSIGN @Sys TO WIDGET "SYSTEM";SET ("*LOAD":Filename$)
```

Once the Screen Builder file has been loaded, then any widget defined in the file can be controlled via the SYSTEM widget by selecting it with the *NAME attribute. For example, if a main PANEL is

defined in the file with the name "Main", it can be accessed with:

```
CONTROL @Sys;SET ("*NAME":"Main","WIDTH":10,"HEIGHT":20)
```

Similarly, if "Main" has a child widget named "Meter2", that child can be accessed by using a "pathname" that specifies both the parent and child:

```
CONTROL @Sys;SET ("*NAME":"Main/Meter2","X":10,"Y":20)
```

Events can be trapped through the SYSTEM widget as well, though all you can do is specify the event type to be trapped -- not the specific event source:

```
ON EVENT @Sys,"CLICKED" GOTO Handler
ON EVENT @Sys,"DONE" GOTO Handler
```

You can set events on a SYSTEM menu for the widgets it contains, though you can't specify which widget the event belongs to. When the event trap occurs, you can determine the source by using the *QUEUED EVENT attribute:

```
Ev$(1:2)[50]
STATUS @Sys;RETURN ("*QUEUED EVENT":Ev$(*)
PRINT "Widget name: ";Ev$(1);"  Widget event: ";Ev$(2)
```

SYSTEM widget event handling, however, is a complicated subject and will be discussed in more detail in a later chapter.

The SYSTEM widget also has an attribute named *CREATE that allows you to create widgets without using the Screen Builder. For example:

```
FOR N=1 TO 10  ! Create ten PUSHBUTTONs.
  CONTROL @Sys;SET ("*NAME":"B"&VAL$(N),"*CREATE":"PUSHBUTTON")
NEXT N
```

You could then use *NAME to access any of the ten PUSHBUTTONs in the SYSTEM widget. You can use a SYSTEM widget as a child of a PANEL defined in the coventional fashion to set up a button or label array for the PANEL.

[2] GENERIC WIDGET & DIALOG ATTRIBUTE REFERENCE

* There is a set of common attributes, as described by the following list. All dimensions and locations

are described in pixels:

BACKGROUND **Type: numeric**
Values: legal PEN values (0-255)
WIDGETS & DIALOGS **Default: depends on system defaults**

Specifies PEN color for widget or dialog background.

BORDER **Type: numeric**
Values: 0 or 1
WIDGETS **Default: 1**

If 1, a border is drawn around the widget.

DEFAULT BUTTON **Type: numeric**
Values: valid index into DIALOG BUTTONS
DIALOGS **Default: index to "OK" button**

This is the button that is activated when you press Enter.

DIALOG BUTTONS **Type: string array**
Values: valid strings
DIALOGS **Default: varies**

Allows you to read or change dialog buttons.

FONT **Type: string**
Values: font names
most WIDGETS, DIALOGS **Default: depends on system defaults**

Specifies text style.

HEIGHT **Type: numeric**
Values: widget or dialog height in pixels
WIDGETS & DIALOGS **Default: varies**

Specifies height in pixels of widget or dialog. May depend on ROWS attribute on some widgets or dialogs.

INSIDE WIDTH **Type: numeric**
Values: widget interior width in pixels
WIDGETS **Default: varies**

Read-only attribute, gives interior width of widget.

INSIDE HEIGHT **Type: numeric**
Values: widget interior height in pixels

WIDGETS Default: varies

Read-only attribute, gives interior height of widget.

JUSTIFICATION Type: string
Values: "LEFT" or "CENTER"
some WIDGETS, DIALOGS Default: "CENTER"

Specifies justification of text in dialog. (Also used in some widgets along with a "RIGHT" justification.)

MAXIMIZABLE Type: numeric
Values: 0 or 1
WIDGETS (0) & DIALOGS Default: 1 (YES)

If 1 and mouse events are enabled, a MAXIMIZABLE box will appear in the title bar, allowing user to pop panel to a full screen display (and back).

MINIMIZABLE Type: numeric
Values: 0 or 1
WIDGETS (0) & DIALOGS Default: 0 (NO)

If 1 and mouse events are enabled, a MINIMIZABLE button will appear in the title bar, allowing user to pop panel to hide the widget in the Minimized Windows application. (BPlus 2.0 & later only.)

MOVABLE Type: numeric
Values: 0 or 1
WIDGETS (0) & DIALOGS Default: 1 (YES)

If 1, the widget (level-0 only) or dialog can be moved around on the display by the user with a mouse. If 0, the object is "locked down".

PEN Type: numeric
Values: PEN numbers (0-255)
most WIDGETS, DIALOGS Default: depends on system defaults

Specifies text color.

RESIZABLE Type: numeric
Values: 0 or 1
WIDGETS (0) & DIALOGS Default: 1 (YES)

If 1, a RESIZE border is drawn around the widget (level-0 only) or dialog, which a user can "click and drag" to change the size of the object.

RESTORE SCREEN Type: numeric
Values: 0 or 1
WIDGETS & DIALOGS Default: 0 (don't restore screen)

Specifies whether the display underneath the widget or dialog will be preserved when the object is removed. If RESTORE SCREEN is set (1), a "snapshot" of the display underneath is saved when the object is created, which means that the object should not be moved or resized (since that original "snapshot" would then no longer be valid).

STACKING ORDER	Type:	numeric
	Values:	0 to the number of siblings
WIDGETS (0)	Default:	0

Given a set of level-0 widgets that overlay each other, defines which one (STACKING ORDER = 0) will be on top.

SYSTEM MENU	Type:	string, string array (level-0 only)
WIDGETS (0)	Values:	legal string values

Allows you to define a "toaster menu" in a level-0 widget by giving it a menu entry as a string or a set of menu entries as a string array. (BPlus 2.0 & later only.)

SYSTEM MENU COUNT	Type:	numeric (level-0 only, read only)
WIDGETS (0)	Values:	positive integer

Returns the number of items in the SYSTEM MENU. (BPlus 2.0 & later only.)

SYSTEM MENU EVENT	Type:	numeric (level-0 only, read only)
	Values:	0 to items -1
WIDGETS (0)	Default:	0

Returns the string array index of the SYSTEM MENU item that generated the last SYSTEM MENU event; the array is always considered to start with an index of 0. (BPlus 2.0 & later only.)

TITLE	Type:	string
	Values:	any string
WIDGETS (0) & DIALOGS	Default:	widget name

String that will appear in the widget (level-0 only) or dialog header.

USER DATA	Type:	string
	Values:	any string
WIDGETS & DIALOGS	Default:	"" (null string)

A user-defined string that can be associated with a widget or dialog.

VERSION	Type:	string
WIDGETS & DIALOGS	Values:	any string

Read-only; returns the version level of BPlus (2.0 & later only).

VISIBLE **Type: numeric**
Values: 0 or 1
WIDGETS **Default: 1 (visible)**

If 1, widget is visible; if 0, widget is hidden.

WIDTH **Type: numeric**
Values: object width in pixels
WIDGETS & DIALOGS **Default: depends on system defaults**

Specifies width in pixels of widget or dialog.

X **Type: numeric**
Values: object x-location in pixels
WIDGETS & DIALOGS **Default: autoplacement**

Specifies the location along the horizontal axis of the upper left corner of the object, with respect to the upper-left-corner of the display (for a standalone widget or all dialogs) or the upper-left-corner of the parent panel (for a widget that is part of a PANEL).

Y **Type: numeric**
Values: object y-location in pixels
WIDGETS & DIALOGS **Default: autoplacement**

Specifies the location along the vertical axis of the upper left corner of the object, with respect to the upper-left-corner of the display (for a standalone widget or all dialogs) or the upper-left-corner of the parent panel (for a widget that is part of a panel).

All level-0 widgets (as of Bplus 2.0) can generate the following events:

CLOSE **Occurs when the widget is destroyed.**

SYSTEM MENU Occurs when the SYSTEM MENU is activated.

[<>]