ASSIGN Command

ASSIGN

Creates/destroys widgets

NOTE

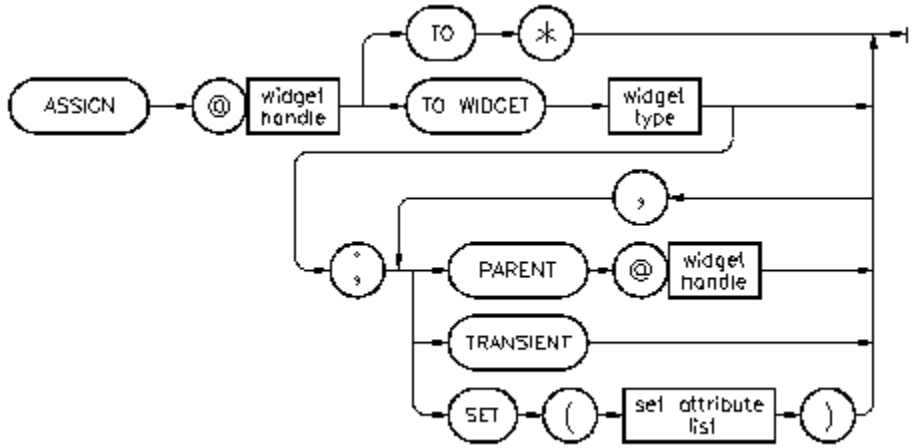
Click the >> bar for additional information on the ASSIGN command.

Legal Usage	Keyboard Executable	Yes
	Programmable	Yes
	In an IF...THEN...	Yes

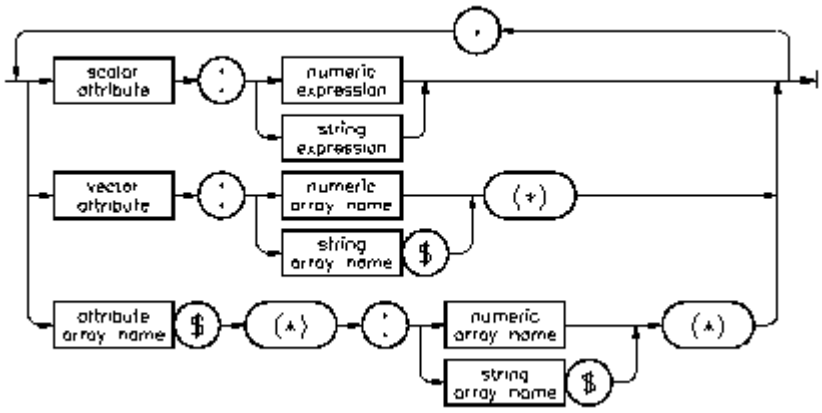
Example Statements

```
ASSIGN @Panel TO WIDGET "PANEL";SET ("X":5,
"Y":5,"WIDTH":500,"HEIGHT":350,"TITLE":
"Engine Monitor")

ASSIGN @Strip TO WIDGET "STRIPCHART";PARENT
@Main1,SET ("X":5,"Y":5,"WIDTH":350,
"HEIGHT":250,"SHOW NUMBERING":0)

ASSIGN @Strip TO *   ! Destroy the @Strip widget
```

Syntax



literal form of set attribute list:



Item	Description	Range
attribute array name	an array of single-valued widget attributes Both the attribute array and either the numeric array or string array must have identical dimensions	depends on widget
numeric array name	array of values to be assigned to this attribute or these attributes	depends on attribute(s)
numeric expression	expression containing the value to be assigned to this attribute	depends on attribute
scalar attribute	a single-valued widget attribute (string expression)	depends on widget
string array name	array of values to be assigned to this attribute or these attributes	depends on attribute(s)
string expression	expression containing the value to be assigned to this attribute	depends on attribute
vector attribute	a multi-valued widget attribute (string expression)	depends on widget

widget handle	name identifying a widget	any valid name
widget type	string expression containing the type of widget to be created	see Description

ASSIGN Command (continued)

Description

The ASSIGN command can be used to:

- Create a new level-0 widget
  - Create a widget as a child of an existing widget
  - Create a transient widget
  - Destroy an existing widget

Within the ASSIGN statement, a "widget handle" (equivalent to an I/O path) is associated with the new widget. The widget handle can be used in subsequent statements, such as STATUS, CONTROL, and ON EVENT, to control the appearance and behavior of the widget.

Also, the widget handle names the widget to be destroyed when ASSIGN @widget handle TO * is used to destroy a widget.

Widget Types

This table shows widget types that can be created with ASSIGN.

Widget	Description
BAR	Displays a numeric value with a rectangular bar
BARS	Displays a bank of numeric values
BITMAP	Displays and creates pixmap files
CASCADE MENU	Menu within PULLDOWN MENU or CASCADE MENU
CLOCK	Graphically displays time
COMBO	Combination of string and list widgets
FILE	Allows selection of disk volume, directory, or file
HPGL VIEW	Displays HPGL graphics files
KEYPAD	Method to enter numbers without using a keyboard
LABEL	Displays text or can be placed as label of another widget
LIMITS	
LIST	

MENU BUTTON	Graphically displays numeric values
MENU SEPARATOR	Scrollable list of textual items
MENU TOGGLE	Appears in PULLDOWN MENU or CASCADE MENU
METER	Line between items in PULLDOWN or CASCADE MENU
NUMBER	Menu button that toggles between states
PANEL	Displays numeric values with a needle
PRINTER	Can enter and format numbers with specified limits
PULLDOWN MENU	Used as a "container" or "parent" widget
PUSHBUTTON	Used to display blocks of test
RADIOBUTTON	Provides a menu of selections
SCROLLBAR	Button that can be placed on the screen or in a PANEL
SEPARATOR	Bank of buttons - only one button at a time can be set
SLIDER	Used to input a relative position
STRING	Line used to visually separate areas of a PANEL
STRIPCHART	Slider pad that moves from minimum to maximum value
SYSTEM	Can enter or display string or array of strings of text
TOGGLEBUTTON	Displays data in a two-dimensional scrolling graph
XYGRAPH	Displays and modifies screens built with Screen Builder
	Used to input or display binary data
	Used to plot data on an X-Y coordinate plane

ASSIGN Command (continued)

PARENT Option

If no parent is specified when creating a new widget, the widget is said to be a "level-0" widget. A level-0 widget is not constrained to be within another widget, and may exist at any place in the HTBasic for Windows output window. The X and Y coordinates of the widget are relative to the upper-left corner of the HTBasic for Windows output window.

Only level-0 widgets may include a title bar, a resize border, and a system menu. The title bar and resize border allow you to change the position and size of the widget.

If a parent is specified, the new widget will be treated as a "child" widget of its parent. If you attempt to move a child widget outside the border of the parent widget, the child will be "clipped" at the parent widget's borders. The child widget's X and Y coordinates are relative to the upper-left corner of the parent widget.

Not all widgets can be parents and not all widgets can be children of parent widgets.

TRANSIENT Option

The TRANSIENT option is used primarily when the resulting widget is to function as a dialog. If you create a widget using the TRANSIENT option, other non-transient widgets cannot be placed on top of the widget.

If the transient widget has a parent, the transient widget is not restricted to lie within the bounds of its parent as are other child widgets. Visually the transient widget appears to be a special type of level-0 widget.

SET Option

All widgets have a variety of attributes that control their appearance and behavior. You can initialize the values of these attributes at the time of creation of the widget by using the SET option.

Attributes are either scalar (may contain a single value) or vector (may be assigned an array of values) and have value of either numeric or string type.

Shorthand: Assigning Attributes

You can use a shorthand method to assign values to several *scalar* attributes without naming them individually on the ASSIGN statement. To do this, you store all the attributes in a string array and all the matching values in another array of the same size.

Then, when you specify both array names in the SET option of the ASSIGN statement, the attribute named in each element of the string array will be assigned the corresponding value in the value array. Elements of the string array that contain nothing, or nothing but blanks, will be ignored. For example:

```
Attribs$(1) = "X"
Attribs$(2) = "Y"
Attribs$(3) = "WIDTH"
Attribs$(4) = "HEIGHT"
Values(1) = 5
Values(2) = 5
Values(3) = 500
Values(4) = 300

ASSIGN @Panel TO WIDGET "PANEL";SET (Attrib$(*):Values(*))
```

HTBasic for Windows

Accessing Help Topics

Help Files

The HTBasic Plus online help system consists of two help (.hlp) files:

- *bplus.hlp* Contains information for HTBasic Plus topics
- *weapons.hlp*: Contains a sample.hlp file you can use as a guideline to build your own .hlp file.

Additional help files for HTBasic are included with HTBasic itself.

Accessing Help Topics

Use the syntax in the following table to access specific help topics. You can type the keyword in all uppercase or all lowercase, but not a mixture of uppercase and lowercase. You can type widget and dialog names in uppercase, lowercase, or a mixture of uppercase and lowercase.

For This Topic:	You Type:	Example
HTBasic for Windows Topic	HELP	HELP
HTBasic for Windows Keyword	HELP keyword [keyword]	HELP IMAGE
HTBasic Plus Keyword	HELP "keyword"	HELP "ASSIGN"
HTBasic Plus Widget	HELP "widget_name"	HELP "BAR WIDGET"
HTBasic Plus Dialog	HELP "dialog_name"	HELP "COMBO DIALOG"
HTBasic Plus Topic	HELP "topic_name"	HELP "Contents"
A Topic in Your Own .hlp File	HELP ",helpfile_name"	HELP",myhelp.hlp"

Accessing Search Entries

The Search entries in the htbasic.hlp file do not include the Search entries in the bplus.hlp file, and vice-versa. Also, when you jump from one help file to the other help file, the Search entries for the first file are still displayed, rather than the Search entries for the file you jumped to.

For example, if you jumped from the htbasic.hlp file to the bplus.hlp file and then accessed the Search entries, the htbasic.hlp file Search entries are displayed, not the Search entries for the bplus.hlp file (and vice-versa). To get around this problem:

- If you are searching main HTBasic for Windows topics (in the htbasic.hlp file) and do not find the Search entry you require, close the htbasic.hlp file and then type *HELP ""* to access the Search entries for HTBasic Plus topics (in the bplus.hlp file).

- If you are searching HTBasic Plus topics (in the bplus.hlp file) and do not find the Search entry you require, close the bplus.hlp file, type *HELP*, and then click Search to access the Search entries for main HTBasic for Windows topics (in htbasic.hlp).

BACKGROUND

Dialogs and Widgets. The value of this attribute specifies the pen to be used for the background area of the widget or dialog. For a discussion of allowed pens, see Settable Pens Table.

ALARM RANGES

This attribute specifies which ranges of the bar, if any, will cause an alarm to be generated when the value of VALUE enters that range.

For example, setting this attribute to "LOW,HIGH" will cause an alarm to be generated any time the VALUE attribute moves out of the MIDDLE range. What actually occurs when an alarm is triggered depends on the value of the ALARM TYPE attribute.

ALARM TYPE

You can set this attribute to one of two values: "BEEP" or "EVENT". If you set it to "BEEP", a beep will be generated any time the VALUE attribute is set to a value which lies in one of the ranges specified in the ALARM RANGES attribute. If you set it to "EVENT", an ALARM event will be generated instead.

BAR WIDTH

This attribute specifies the width of the visible portion of the BAR. BAR WIDTH should be less than or equal to WIDTH. If BAR WIDTH is greater than WIDTH, the widget sets the value of BAR WIDTH to the value of WIDTH. The extra space between BAR WIDTH and WIDTH will be equally allocated as background color on either side of the BAR.

HIGH LIMIT

If this attribute is set, the portion of the BAR above this limit will be painted using the pen color specified by the HIGH PEN attribute. On a monochrome monitor, a line is drawn at the HIGH LIMIT.

To have HIGH LIMIT appear on the BAR, HIGH LIMIT must be less than or equal to MAXIMUM and must be greater than or equal to MINIMUM, and must be greater than or equal to LOW LIMIT.

The value of HIGH LIMIT acts as a trip point between the alarm ranges on the BAR. It is the trip point between the middle range and the high range. The programmer can use HIGH LIMIT, LOW LIMIT, ALARM RANGES, and ALARM TYPE to specify the alarm behavior of the BAR.

HIGH PEN

If the HIGH LIMIT attribute is set, the portion of the bar that exceeds that limit will be painted using this pen color. This attribute has no effect on monochrome monitors. For a discussion of allowed pens, see Settable Pens Table.

LIMITS LINE

LIMITS displays two lines showing the upper and lower limits of the given BAR.

LOGARITHMIC

If the value of this attribute is 1, the BAR operates in a logarithmic mode. If the value is 0, the bar operates in a linear mode.

LOW LIMIT

If this attribute is set, the portion of the BAR below this limit will be painted using the pen color specified by the LOW PEN attribute. On a monochrome monitor, a line is drawn at the LOW LIMIT.

To have LOW LIMIT appear on the BAR, LOW LIMIT must be greater than or equal to MINIMUM and must be less than or equal to MAXIMUM, and must be less than or equal to HIGH LIMIT.

The value of LOW LIMIT acts as a trip point between the alarm ranges on the BAR. It is the trip point between the middle range and the low range. The programmer can use HIGH LIMIT, LOW LIMIT, ALARM RANGES, and ALARM TYPE to specify the alarm behavior of the BAR.

LOW PEN

If the LOW LIMIT attribute is set, the portion of the bar below that limit will be painted using this pen color. This attribute has no effect on monochrome monitors. For a discussion of allowed pens, see Settable Pens Table

MAXIMUM

The value of this attribute specifies the upper bound on the displayed value of the VALUE attribute. That is, when the value of VALUE equals the value of MAXIMUM, the BAR will reach the top or right (depending on the ORIENTATION) of the widget.

MAXIMUM must be greater than MINIMUM. If the ORIGIN attribute is set greater than or equal to MAXIMUM and the VALUE attribute is set greater than or equal to MAXIMUM, no bar is drawn.

MIDDLE PEN

The portion of the bar that lies between the values of the LOW LIMIT and HIGH LIMIT attributes will be painted using this pen color. This attribute has no effect on monochrome monitors. For a discussion of allowed pens, see Settable Pens Table.

MINIMUM

The value of this attribute specifies the lower bound on the displayed value of the VALUE attribute. That is, when the value of VALUE equals the value of MINIMUM, the BAR will reach the bottom or left (depending on the ORIENTATION) of the widget. MINIMUM must be less than MAXIMUM.

ORIENTATION

This attribute determines the direction or layout of the BAR.

ORIENTATION Attribute Values

Allowed Value	Description
"UP", "VERTICAL"	The BAR will be oriented vertically and the VALUE attribute will increase in the up direction
"RIGHT", "HORIZONTAL"	The BAR will be oriented horizontally and the VALUE attribute will increase toward the right

ORIGIN

This attribute specifies the origin of the bar, the point from which the bar is drawn until it reaches the level of VALUE.

For example, a BAR widget with a MINIMUM of 0, MAXIMUM of 100, VALUE of 50, and ORIGIN of 40 will generate a bar height of 10 (VALUE minus ORIGIN, or 50-40 in this case) with the bottom of the bar at 40 and the top at 50. A line is drawn to show the newly established ORIGIN.

ORIGIN PEN

ORIGIN PEN sets the color or the ORIGIN line.

VALUE

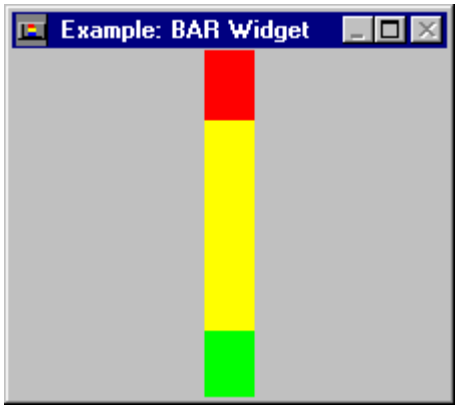
This attribute reflects the current value of the BAR. If the value of VALUE exceeds MAXIMUM or MINIMUM, the value that is displayed will be "clipped" by MAXIMUM or MINIMUM. However, the actual value of VALUE will not be clipped.

BAR Widget

Graphically displays numeric values

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See BAR Widget for a program that creates a BAR widget similar to the one displayed above.

NOTE

See the following program for another example using the BAR widget:

Bar Meter Limits

Attributes

See BAR Widget Attributes for the BAR widget attribute list.

Remarks

The BAR widget graphically displays numeric values using a single bar.

If your computer has a monochrome monitor, the BAR widget will handle the situation automatically. There are no attributes related to color to set for that special case. If you have a monochrome monitor, you will automatically get a frame around the BAR.

A frame consists of an outline around the BAR and lines at the LOW LIMIT and HIGH LIMIT values. The frame pen will be the inverse of the background pen. The background pen and the BAR pen will be the same. For information on creating composite bar displays, see the BARS widget.

The bar size is controlled by writes to the VALUE attribute. The BAR widget does not operate independently. You must write VALUEs to the widget. An event can be generated on any or all of the three ranges: "LOW", "MIDDLE", or "HIGH".

BAR widget events are level-triggered, not edge-triggered. An event is generated any time a VALUE is within the designated ALARM RANGE, not only when the VALUE transitions from one range to the other. You must specifically set the ALARM TYPE to event before an event can be generated. Otherwise, only beeps are generated.

Events

Events for the BAR widget are:

- ALARM
- SYSTEM MENU

ALARM

This event is generated when the VALUE attribute is set to a value within one of the ranges named in the ALARM RANGES attribute and the ALARM TYPE is "EVENT". The only time an ALARM event is generated is when the value of VALUE is set to one, several, or all of the following:

- To or above HIGH LIMIT and the value of ALARM RANGES is "HIGH".
- Between LOW LIMIT and HIGH LIMIT and the value of ALARM RANGES is "MIDDLE".

- To or below LOW LIMIT and the value of ALARM RANGES is "LOW".

Redraw activities, such as changing the ORIENTATION attribute value, will not generate an ALARM event.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

BAR Widget Attributes

BAR Widget Attributes

Attribute	Type	Allowed Values	Default
ALARM RANGES	String	Any comma-separated combination of: "LOW", "MIDDLE", "HIGH", or the null string	The null string
ALARM TYPE	String	"BEEP", "EVENT"	"BEEP"
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
BAR WIDTH	Numeric	Any positive integer	25
BORDER [2]	Numeric	0,1	1
HEIGHT	Numeric	Any integer number (of pixels)	300 pixels
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
HIGH LIMIT	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	MAXREAL
HIGH PEN	Numeri	Any valid BASIC pen number (0-255)	System dependent [1]

	c		
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
LIMITS LINE	Numeric	0,1	0
LOGARITHMIC	Numeric	0, 1	0 (linear)
LOW LIMIT	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	MAXREAL
LOW PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MAXIMUM	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	100
MIDDLE PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
MINIMIZABLE [4]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)

MINIMUM	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	1
MOVABLE [4]	Numeric	0,1	1 (movable, click on title bar and drag)
ORIENTATION	String	"UP", "VERTICAL", "RIGHT", "HORIZONTAL"	"VERTICAL" (max value at top)
ORIGIN	Real	Any	- MAXREAL
ORIGIN PEN	Numeric	Any valid BASIC pen number	Black (0)
RESIZABLE [4]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
STACKING ORDER	Numeric	0 to number of siblings [3]	0
SYSTEM MENU [4]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT	Num	0 to number of items in system menu	0

[4]	eric		
SYSTEM MENU EVENT [4]	Numeric	0 to number of items in system menu -1	0
TITLE [4]	String	Any valid string	BAR
USER DATA	String	Any valid string	None
VALUE	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	1
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	150 pixels
X	Numeric	Any integer, number of pixels from 0,0 [5]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [5]	Autoplacement

[1] Read the CONFIG file or query for the default with a STATUS command

[2] Child widget only

[3] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[4] For level-0 widgets only

[5] Screen or parent work area upper-left corner

ALARM RANGES

This attribute specifies which ranges of the bar, if any, will cause an alarm to be generated when the value of VALUE enters that range. For example, setting this attribute to "LOW,HIGH" will cause an alarm to be generated any time the value of the VALUE attribute strays out of the MIDDLE range. What occurs when an alarm is triggered depends on the value of the ALARM TYPE attribute.

ALARM TYPE

You can set this attribute to one of two values: "BEEP" or "EVENT". If you set it to "BEEP", a beep will be generated any time the VALUE attribute is set to a value which lies in one of the ranges specified in the ALARM RANGES attribute. If you set it to "EVENT", an ALARM event will be generated instead.

AUTOPOSITION

When AUTOPOSITION is set to 1, the BARS widget will automatically adjust the value of the BAR SEPARATION attribute so that the single-bars evenly fill the area designated for the BARS work area.

If the value of this attribute is 1, any time you change the value of either the WIDTH or the BAR WIDTH attribute, the widget will automatically adjust the value of the BAR SEPARATION attribute so that the single-bars evenly fill the area designated for the BARS work area. SCROLLABLE and AUTOPOSITION are mutually exclusive - setting one turns off the other.

AUTOSCALE

If the value is 1, when VALUE exceeds MAXIMUM or is lower than MINIMUM, MAXIMUM and/or MINIMUM will be adjusted on the display in order to keep the value between MINIMUM and MAXIMUM.

BAR BACKGROUND

The value of this attribute specifies the pen color that will be used to paint all of the single-bar backgrounds.

BAR COUNT

The value of BAR COUNT is the number of single-bars in the BARS widget.

BAR LABEL

The string that appears in the label box that is adjacent to each single-bar in the BARS widget. The BAR LABEL string will be centered in the label box.

BAR SEPARATION

This integer value specifies the space in pixels between the single-bars in the BARS widget. When BAR SEPARATION is set to any value, the widget sets its AUTOPOSITION attribute value to 0.

BAR WIDTH

This attribute specifies the width of the visible portion of the single-bars. The BAR WIDTH and BAR SEPARATION attributes work together to set the appearance of the BARS widget work area.

If you set the value of BAR WIDTH and AUTOPOSITION is set to 1, the widget will recalculate and set the value of BAR SEPARATION so that the single-bars evenly fill the area designated for the BARS work area.

CURRENT BAR

The value of this attribute selects the single-bar on which the "Single-Bar" attributes work. Use the value of CURRENT BAR in STATUS and CONTROL commands to get or set individual single-bar attributes.

FIRST BAR

FIRST BAR is operational only when the SCROLLABLE attribute is set. The BARS area is scrolled so that the specified bar is at the left edge for VERTICAL ORIENTATION and the bottom for HORIZONTAL orientation.

FONT

This attribute specifies the font to be used for the labels, limits, and values. For a description of allowed fonts, see Specifying FONT Attribute Values.

HIGH LIMIT

If this attribute is set, the portion of the single-bar above this limit will be painted using the pen color specified by the HIGH PEN attribute. On a monochrome monitor, a line is drawn at the HIGH LIMIT.

To have HIGH LIMIT appear on the bar, HIGH LIMIT must be less than or equal to MAXIMUM and must be greater than or equal to MINIMUM, and must be greater than or equal to LOW LIMIT.

The value of HIGH LIMIT acts as a trip point between the alarm ranges on the single-bar. It is the trip point between the middle range and the high range. The programmer must use HIGH LIMIT, LOW LIMIT, ALARM RANGES, and ALARM TYPE to specify the alarm behavior of the single-bar.

HIGH PEN

If the HIGH LIMIT attribute is set, the portion of the bar that exceeds that limit will be painted using this pen color. This attribute has no effect on monochrome monitors. For a discussion of allowed pens, see Settable Pens Table.

LABEL BACKGROUND

The label background will be painted using this pen color.

LABEL PEN

The label will be painted using this pen color. For a discussion of allowed pens, see Settable Pens Table.

LABEL WIDTH

The value of this attribute specifies the width of the label box near each single-bar. This attribute is significant only when the ORIENTATION attribute has the value "RIGHT" or "HORIZONTAL".

LIMITS BACKGROUND

The limits background will be painted using this pen color.

LIMITS LINE

LIMITS displays two lines showing the upper and lower limits of the given bar.

LIMITS PEN

The limits will be painted using this pen color. For a discussion of allowed pens, see Settable Pens Table.

LIMITS WIDTH

The value of this attribute specifies the width of the limits boxes. This attribute is significant only when the ORIENTATION attribute has the value "UP" or "VERTICAL".

LOGARITHMIC

This attribute sets the scale of the bars to either logarithmic or linear. If the value is 1, all of the bars in the BARS widget operate in logarithmic mode. If the value is 0, all of the bars in the BARS widget operate in linear mode.

LOW LIMIT

If this attribute is set, the portion of the single-bar below this limit will be painted using the pen color specified by the LOW PEN attribute. On a monochrome monitor, a line is drawn at the LOW LIMIT. To have LOW LIMIT appear on the bar, LOW LIMIT must be greater than or equal to MINIMUM and must be less than or equal to MAXIMUM, and must be less than or equal to HIGH LIMIT.

The value of LOW LIMIT acts as a trip point between the alarm ranges on the single-bar. It is the trip point between the middle range and the low range. The programmer must use HIGH LIMIT, LOW LIMIT, ALARM RANGES, and ALARM TYPE to specify the alarm behavior of the single-bar.

LOW PEN

If the LOW LIMIT attribute is set, the portion of the bar below that limit will be painted using this pen color. This attribute has no effect on monochrome monitors. For a discussion of allowed pens, see Settable Pens Table.

MAXIMUM

The value of this attribute specifies the upper bound of the display for the values of either the single-bar VALUE attribute or the multiple-bars VALUES attribute. That is, when any single-bar value equals the value of MAXIMUM, the bar will reach the top or right (depending on the ORIENTATION) of the widget.

If the value is set MAXIMUM or becomes less than MINIMUM and AUTOSCALE, the BARS widget will automatically adjust MAXIMUM and/or MINIMUM to accommodate the new value. MAXIMUM must be greater than MINIMUM. If the ORIGIN attribute is set greater than or equal to MAXIMUM and the VALUE attribute is set greater than or equal to MAXIMUM, no bar is drawn.

The value of MAXIMUM will be displayed with only one digit to the right of the decimal point. For example, if you set the value of MAXIMUM to 123.4567, the display will show 123.5. If you set the value of MAXIMUM to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of MAXIMUM as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of MAXIMUM to the string "0.000237" or "4.167E-4" in the ASSIGN or CONTROL statement.

MIDDLE PEN

The portion of the bar that lies between the values of the LOW LIMIT and HIGH LIMIT attributes will be painted using this pen color. This attribute has no effect on monochrome monitors. For a discussion of allowed pens, see Settable Pens Table.

MINIMUM

The value of this attribute specifies the lower bound of the display for the values of either the single-bar VALUE attribute or the multiple-bars VALUES attribute. That is, when any single-bar value equals the value of MINIMUM, the bar will reach the bottom or left (depending on the ORIENTATION) of the widget.

If the value is set MAXIMUM or becomes less than MINIMUM and AUTOSCALE, the BARS widget will automatically adjust MAXIMUM and/or MINIMUM to accommodate the new value. MINIMUM must be less than MAXIMUM.

The value of MINIMUM will be displayed with only one digit to the right of the decimal point. For example, if you set the value of MINIMUM to 123.4567, the display will show 123.5. If you set the value of MINIMUM to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of MINIMUM as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of MINIMUM to the string "0.000237" or "4.167E-4" in the ASSIGN or CONTROL statement.

ORIENTATION

This attribute determines the direction or layout of the BARS widget.

ORIENTATION Attribute Values

Allowed Value	Description
"UP", "VERTICAL"	The BARS widget will be oriented vertically and the values of VALUES will increase in the up direction
"RIGHT", "HORIZONTAL"	The BARS will be oriented horizontally and the values of VALUES will increase towards the right

ORIGIN

This attribute specifies the origin of the bar, the point from which the bar is drawn until it reaches the level of VALUE. For example, a BAR widget with a MINIMUM of 0, MAXIMUM of 100, VALUE of 50, and ORIGIN of 40 will generate a bar height of 10 (VALUE minus ORIGIN, or 50-40 in this case) with the bottom of the bar at 40 and the top at 50. A line is drawn to show the newly established ORIGIN.

ORIGIN PEN

ORIGIN PEN sets the color or the ORIGIN line.

SCROLL INCREMENT

When SCROLLABLE is set, this attribute determines whether scrolling takes place in increments of PIXELs or entire BARs.

SCROLLABLE

This attribute, when set to 1, enables the BARS area of the widget to scroll. If a combination of BAR COUNT, BAR WIDTH, and BAR SEPARATION is greater than the BARS area, a scrollbar appears. Use the scrollbar to scroll the desired bar into view.

Scrolling occurs in increments of pixels or bars as determined by the SCROLL INCREMENT attribute. Only the bars and their labels scroll - the limits labels remain stationary. SCROLLABLE and AUTOPOSITION are mutually exclusive - setting one turns off the other.

SHOW LABELS

This attribute is used to turn on and off the labels boxes near the single-bars. If its value is 1, all label boxes are visible. If its value is 0, all label boxes are invisible.

SHOW LIMITS

This attribute is used to turn on and off the limits boxes near the single-bars. If its value is 1, all limits boxes are visible. If its value is 0, all limits boxes are invisible.

SHOW VALUES

This attribute is used to turn on and off the values boxes near the single-bars. If its value is 1, all values boxes are visible. If its value is 0, all values boxes are invisible.

VALUE

This attribute reflects the current value of the single-bar. The value of VALUE will be displayed with only one digit to the right of the decimal point. For example, if you set the value of VALUE to 123.4567, the display will show 123.5. If you set the value of VALUE to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of VALUE as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of VALUE to the string "0.000237" or "4.167E-4" in the ASSIGN or CONTROL statement.

VALUE BACKGROUND

The value of this attribute specifies the pen color that will be used to paint all of the value box backgrounds.

VALUE PEN

The value of this attribute specifies the pen color that will be used to paint all of the single-bar values in the value boxes. For a discussion of allowed pens, see Settable Pens Table.

VALUE WIDTH

The value of this attribute specifies the width of the value box adjacent to each single-bar. This attribute is significant only when the ORIENTATION attribute has the value "RIGHT" or "HORIZONTAL". Otherwise, the width of the value box is related to the BAR WIDTH and BAR SEPARATION attributes. That is, VALUE WIDTH is approximately equal to BAR WIDTH plus BAR SEPARATION.

VALUES

This attribute is a numeric or string array of bar values that is used to fill in the single-bar values in the BARS widget. The values of VALUES are updated from bar number 1 through the number of elements in your VALUES array or the number of single-bars, whichever is the smaller number.

The values of VALUES will be displayed with only one digit to the right of the decimal point. For example, if you set a value in the VALUES array to 123.4567, the display will show 123.5. If you set one of the values of VALUES to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the values of VALUES as strings in the string array. For example, if you want to display 0.000237 or 4.167E-4, set the appropriate value in the VALUES array to the string "0.000237" or "4.167E-4" in the ASSIGN or CONTROL statement. For example, if you execute

```
OUTPUT ITEMS$(0) USING "#,ZZZ";5.23456
CONTROL @W;SET ("VALUES":ITEMS$(*))
```

HTBasic for Windows will put "005" in the VALUE attribute for the first bar.

HTBasic for Windows

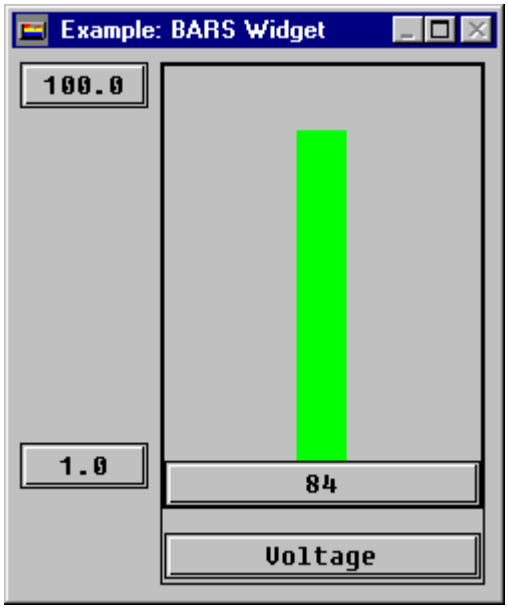
BARS Widget

BARS Widget

Graphically displays a bank of numeric values

Legal Usage	Level-0 Widget:	Yes	
	Parent to:	None	
	Child of:		PANEL

Example Image



Example Program

See BARS Widget for a program that creates a BARS widget with one bar similar to the display shown above. This display is similar to a BAR widget, enhanced with some features of the METER widget such as the limits boxes to the left of the bar and the value and label boxes beneath it.

See the following programs for other examples using the BARS widget:

BARS Test
Engine Monitor - Level-0

Attributes

See BARS Widget Attributes for the BARS widget attribute list.

Remarks

The BARS widget is a variation of the BAR widget, except that the BARS widget can contain several BAR widgets.

If your computer has a monochrome monitor, the BARS widget will handle the situation automatically. There are no attributes related to color to set for that special case. If you have a monochrome monitor, you will automatically get a frame around each single bar.

A frame consists of a border around the single-bar and lines at the LOW LIMIT and HIGH LIMIT values. The frame pen will be the inverse of the background pen. The background pen and the single-bar pen will be the same.

An event can be generated on any or all of the three ranges: "LOW", "MIDDLE", or "HIGH". BARS widget events are level-triggered, not edge-triggered. An event is generated any time a VALUE is within the designated ALARM RANGE, not only when the VALUE transitions from one range to the other. You must specifically set the ALARM TYPE to event before an event can be generated. Otherwise, only beeps are generated.

If you set an event for one of the bars, you set that event for all of the bars in the widget. if an event occurs, you must check all bars to see which bar generated the event. You can set the HIGH, MIDDLE, and LOW limits and the corresponding PEN colors for each individual bar. However, the MAXIMUM and MINIMUM range is the same for all bars.

You cannot individually set the colors for the BACKGROUND and PENS in the LABEL and VALUE boxes for the bars. You cannot turn on LABEL or VALUE boxes for certain bars and not for others.

Events

Events for the BARS widget are:

- ALARM
  - SYSTEM MENU

ALARM

This event is generated when the VALUE attribute is set to a value within one of the ranges named in the ALARM RANGES attribute and the ALARM TYPE is "EVENT".

The only time an ALARM event is generated is when the value of VALUE is set to one, several, or all of the following:

- To or above HIGH LIMIT and the value of ALARM RANGES is "HIGH"
- Between LOW LIMIT and HIGH LIMIT and the value of ALARM RANGES is "MIDDLE"
- To or below LOW LIMIT and the value of ALARM RANGES is "LOW"

Redraw activities, such as changing the ORIENTATION attribute value, will not generate an ALARM event.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

BARS Widget Attributes

BARS Widget Attributes

Attribute	Type	Allowed Values	Default
ALARM RANGES	String	Any combination of: "LOW", "MIDDLE", "HIGH"	Null String
ALARM TYPE	String	"BEEP", "EVENT"	"BEEP"
AUTOPOSITION	Numeric	0,1	1
AUTOSCALE	Numeric	0,1	1
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BAR BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BAR COUNT	Numeric	Any positive integer	1
BAR LABEL	String	Any	Value of CURRENT BAR
BAR SEPARATION	Numeric	Any positive integer	Depends on WIDTH and BAR WIDTH
BAR WIDTH	Numeric	Any positive integer	25

BORDER [3]	Numeric	0,1	1
CURRENT BAR	Numeric	1 to BAR COUNT	1
FIRST BAR	Numeric	1 to 32767	1
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	300 pixels
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
HIGH LIMIT	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	MAXREAL
HIGH PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
LABEL BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
LABEL	N	Any valid BASIC pen number (0-	System

PEN	u m eri c	255)	dependent [2]
LABEL WIDTH	N u m eri c	Any positive integer	Autosizing
LIMITS BACKGROUND	N u m eri c	Any valid BASIC pen number (0-255)	System dependent [2]
LIMITS LINE	N u m eri c	0,1	0
LIMITS PEN	N u m eri c	Any valid BASIC pen number (0-255)	System dependent [2]
LIMITS WIDTH	N u m eri c	Any positive integer	Autosizing
LOGARITHMIC	N u m eri c	0, 1	0 (linear)
LOW LIMIT	N u m eri c	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	MAXREAL
LOW PEN	N u m eri c	Any valid BASIC pen number (0-255)	System dependent [2]
MAXIMIZABLE [5]	N u m eri c	0,1	1 (maximizable, button appears in title bar)
MAXIMUM	N u	LOGARITHMIC is 0: any real number	100

	meric or String	LOGARITHMIC is 1: any positive real number	
MIDDLE_PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MINIMUM	Numeric or String	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	1
MOVABLE [5]	Numeric	0,1	1 (movable, click title bar and drag)
ORIENTATION	String	"UP", "VERTICAL", "RIGHT", "HORIZONTAL"	"VERTICAL" (max value at top)
ORIGIN	Numeric	Any	-MAXREAL
ORIGIN_PEN	Numeric	Any valid BASIC pen number (0-255)	Black (0)
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE	N	0,1	0 (do not

SCREEN	u m eri c		restore the screen)
SCROLL INCREMENT	St rin g	"BAR", "PIXEL"	"BAR"
SCROLLABLE	N u m eri c	0,1	0
SHOW LABELS	N u m eri c	0,1	1
SHOW LIMITS	N u m eri c	0,1	1
SHOW VALUES	N u m eri c	0,1	1
STACKING ORDER	N u m eri c	0 to number of siblings [4]	0
SYSTEM MENU [5]	St rin g or str in g arr ay	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	N u m eri c	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	N u m eri	0 to number of items in system menu -1	0

	c		
TITLE [5]	String	Any valid string	BARS
USER DATA	String	Any valid string	None
VALUE	Numeric or String	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	1
VALUE BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
VALUE PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
VALUE WIDTH	Numeric	Any positive number	Autosizing
VALUES	Numeric array or String array	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	An array of one element which contains 1.0
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1

WIDTH	Numeric	Any integer number (of pixels)	250 pixels
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

AUTO SIZE

The AUTO SIZE attribute, when set to 1 automatically resizes the widget to the size of the bitmap.

BACKGROUND

The BACKGROUND attribute specifies the pen number that determines the color of the frame.

BITMAP FILE

The BITMAP FILE attribute specifies the name of the bitmap file that is to be displayed. BITMAP FILE must be in X Windows Dump File (xwd) or Microsoft ® Windows ® *.BMP format.

BITMAP HEIGHT

Return-Only Attribute. BITMAP HEIGHT returns the value in pixels of the height of the bitmap. BITMAP HEIGHT is 0 if there is no bitmap.

BITMAP WIDTH

Return-Only Attribute. BITMAP WIDTH returns the value in pixels of the width of the bitmap. BITMAP WIDTH is 0 if there is no bitmap.

DUMP AREA

Set-Only Attribute. DUMP AREA can be used only with ASSIGN or CONTROL.
This attribute specifies a file to which a DUMP of the pixmap of an area of a screen is made. Setting this attribute initiates the area definition phase as follows:

- The cursor changes its shape to a cross-hair
- The operator moves the mouse to the upper-left of the area to be dumped
- The operator presses and holds the left mouse button
- The operator drags the cross-hair to the lower-right corner of the area to be dumped, generating a box on the screen that encloses the area to be dumped
- The operator releases the mouse to generate the area dump

DUMP FORMAT

DUMP FORMAT is the format of the BITMAP file created when the DUMP WINDOW and DUMP AREA attributes are invoked. It must be XWD or BMP.

DUMP WINDOW

Set-Only Attribute. DUMP WINDOW specifies a file to which a DUMP of the pixmap of an indicated window is made. Setting this attribute changes the cursor shape to a pointing finger. Click on any visible window to dump that window to the specified BITMAP file.

Clicking on a non-client area (the area of the window with title bar, menu bar, borders, etc.) dumps the entire window. Clicking on a client area (the central area in which information is presented or entered) dumps only that client area.

FONT

The FONT attribute specifies which font is used for the widget LABEL.
For a description of allowed fonts, see Specifying FONT Attribute Values.

FRAME

The FRAME attribute, when set to 1, draws a picture frame around the bitmap.

LABEL

The LABEL attribute specifies the text that will be displayed on the bottom section of the FRAME. When set non-NULL, the bottom section of this frame is enlarged to accommodate the text. If set to NULL, the bottom section is reduced to that of the top section. If the FRAME is set to "0", no label is displayed.

MAX PENS

The MAX PENS attribute specifies the maximum number of pens a BITMAP can change and use. Each system has a finite number of pens: 2, 16, 64, 256, etc. Each BITMAP widget has its own logical color map with each map containing up to the same number of pens as the physical color map.

The BITMAP widget that has the focus will set the physical color map to its color map. All other Windows applications, including the remaining BITMAP widgets, will redefine their pen usage to best use the new physical color map on a closest match basis.

Since it is computationally expensive to remap the color map and pen usage, each BITMAP widget will use a non-overlapping portion of the physical color map. Therefore, remapping will not occur if the sum of all BITMAP widgets' MAX PENS is less than the physical pens available.

If a BITMAP requires 100 different colors (see the "PENS NEEDED" attribute) and MAX PENS is set to "70", the 30 least frequently-used pens of that BITMAP will be mapped into the 70 on a closest-match basis. Setting MAX PENS to "0" makes the BITMAP widget map to the Windows default colors so that remapping never occurs.

MOUSE CLICK

Return-Only Attribute. MOUSE CLICK returns the X and Y coordinates of the last MOUSE CLICK event, relative to the displayed bitmap's upper-left corner.

PEN

The PEN attribute specifies the pen color to used for LABEL.

PENS NEEDED

Return-Only Attribute. PENS NEEDED indicates the number of pens the BITMAP requires.

RETAIN RASTER

RETAIN RASTER, when set to "0", rereads the file when it repaints. When set to "1", RETAIN RASTER keeps a copy of the bitmap and color map in memory, allowing faster repainting.

NOTE

Because your program will use a significantly larger amount of memory when you set RETAIN RASTER to 1, do so only when it is important to redraw the BITMAP widget quickly.

HTBasic for Windows

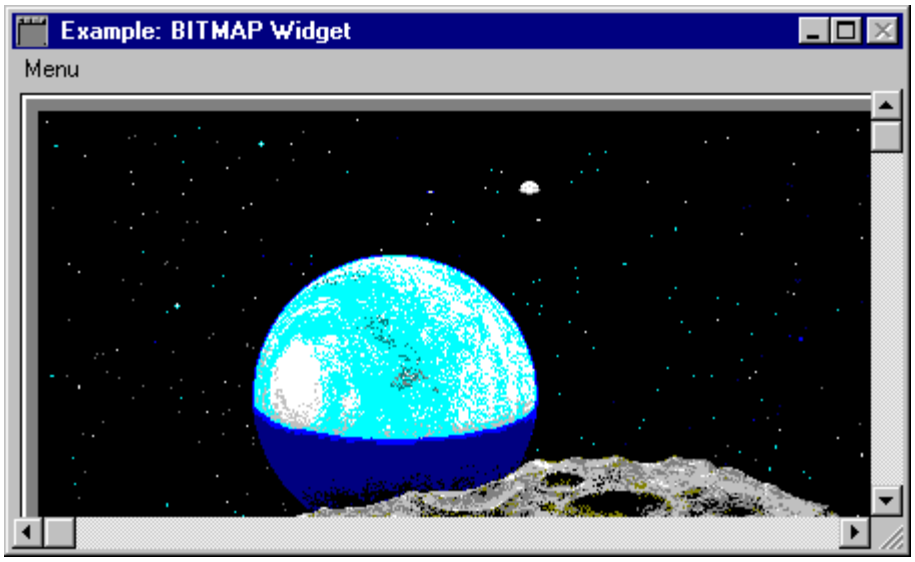
BITMAP Widget

BITMAP Widget

Displays pixmap files in "xwd" or "*.BMP" format

Legal Usage	Level-0 Widget:	Yes	
	Parent to:	None	
	Child of:		PANEL

Example Image



Example Program

See the *BITMAP Widget* program for an example program that allows you to read in and display a bitmap file similar to that shown above, and to store elements back to a file. It also demonstrates the PANEL widget's SCROLLABLE attribute.

The *BITMAP Widget* program creates a PANEL for the BITMAP widget. The SCROLL WIDTH and HEIGHT are set to a small value so that scroll bars will not appear initially. The actual heights are set later to fit the bitmap that has been loaded.

To demonstrate the SCROLLABLE attribute, load a bitmap file and then reduce the size of the panel until the bitmap does not fit in the panel's work area. At that point,

scrollbars on the panel's right side or bottom, or both, will appear to allow you to scroll the bitmap in and out of the work area.

NOTE

See the following program for another example using the BITMAP widget:

Wing Stress/Vibration Analysis

Attributes

See BITMAP Widget Attributes for the BITMAP widget attribute list.

Remarks

The BITMAP widget displays pixmap files in .BMP format. The widget can also create an XWD or BMP file from an entire window in either X11 Window Dump (XWD) or a user-defined area of the screen.

The BITMAP widget detects mouse clicks in the displayed pixmap and generates an event from which you can determine the mouse coordinates at the time of the click. BITMAP can create files from an entire window or from a user-defined area of the screen.

After you create a pixmap file, the file is read into the widget using the BITMAP FILE attribute:

```
ASSIGN @Bitmap TO WIDGET "BITMAP"
CONTROL @Bitmap;SET ("BITMAP FILE":"COVER.BMP")
```

If you set the DUMP WINDOW attribute to a filename (as shown) and then click on the user interface, the user interface will be saved in the file (with a .BMP format).

```
CONTROL @Bitmap;SET ("DUMP FORMAT":"BMP","DUMP WINDOW":   "BITSAVE.BMP")
```

DUMP AREA allows you to store a region of the user interface (as defined by a mouse click-and-drag operation) to a file.

You can set a MOUSE CLICKED event to trap a mouse click on a bitmap, and then use the MOUSE CLICK attribute to return the X,Y coordinates of the pixel clicked on. For an example of this, see the Wing Stress/Vibration Analysis program.

When the BITMAP widget is used to view bitmaps with a large number of colors, color changes may occur in the display. For example, color bitmaps usually have 16 or 256 colors. You can display a color bitmap of 16 colors on a display that supports 256 colors. However, if you try to display a color bitmap of 256 colors on one that only supports 16 colors, some unexpected colorations may occur.

Two bitmap files with 256 colors may not have the same 256 colors. Since the display can be reprogrammed to select the 256 colors from a much larger set of colors, this "color mapping" may change from bitmap file to bitmap file.

For example, suppose you load two bitmap files consecutively that have different color mappings (such as "BMAZTEC" and "BMASTRO"). In this case, the second file loaded will change the display's color map to the colors it requires, regardless of the first file's colors. The colors of the first bitmap may then change to a set of unexpected colors.

Events

Events for the BITMAP widget are:

- MOUSE CLICKED
- SYSTEM MENU

MOUSE CLICKED

The MOUSE CLICKED event is generated when the left-mouse button is released over the bitmap. If there is no bitmap, this event cannot be generated.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

BITMAP Widget Attributes

BITMAP Widget Attributes

Attribute	Type	Allowed Values	Default
AUTO SIZE	Numeric	0,1	1
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BITMAP FILE	String	Existing filename	Null
BITMAP HEIGHT	Numeric	Return only, depends on bitmap size	0
BITMAP WIDTH	Numeric	Return only, depends on bitmap size	0
BORDER [3]	Numeric	0,1	1
DUMP AREA	String	Valid filename	Null
DUMP FORMAT	String	"XWD" or "BMP"	"XWD"
DUMP WINDOW	String	Valid filename	Null
FONT	String	*Font* [1]	System dependent [2]
FRAME	Numeric	0,1	1
HEIGHT	Numeric	Any integer number (of pixels)	300 pixels
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies

LABEL	String	Any string	Same as TITLE attribute
MAX PENS	Numeric	1 to 256	256
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE [5]		0,1	0 (not minimizable, button does not appear in title bar)
MOUSE CLICK	Two-element Integer array	Valid integer	0,0
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PEN	Numeric	0 to 255	System dependent [2]
PENS NEEDED	Numeric	1 to 256	0, determined by pixmap
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
RETAIN RASTER	Numeric	0, 1	0

STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or string array	Any valid string or string array	0
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu-1	0
TITLE [5]	String	Any valid string	BITMAP
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	150 pixels
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

BORDER

Child Widgets and Dialogs Only. When this attribute is set to 1, a border will be drawn around the periphery of the child widget or the dialog.

HTBasic for Windows

Building Your .hlp File

Using a Help-Authoring Tool

You can use any Windows Help-Authoring system to create a Windows .hlp file. However, when building the help files, the context strings you use must match the string you use as the HELP TOPIC topic_id for the widget. After compiling a help file, you should save the compiled program to a new *.hlp file.

Do not save the new program as "htbasic.hlp", "bplus.hlp" or "weapons.hlp", as it will overwrite the original version and you will need to reload HTBasic for Windows.

NOTE

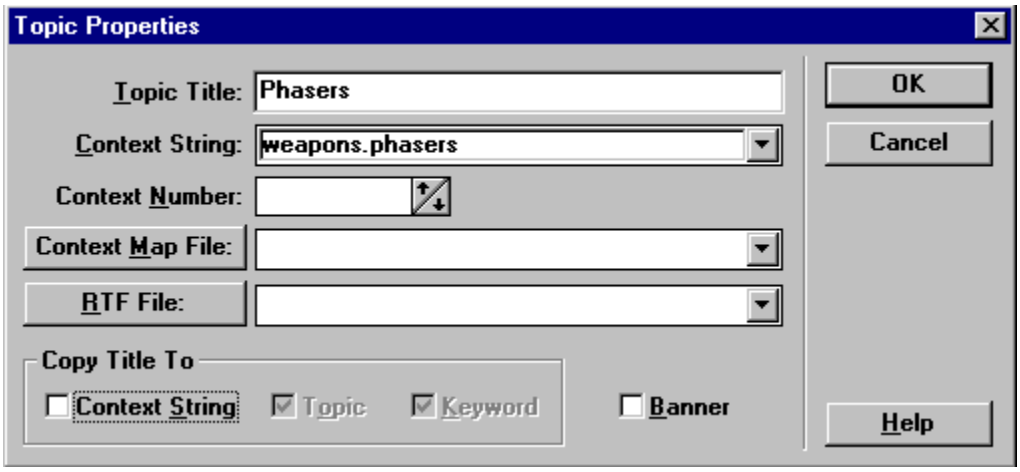
HTBasic for Windows includes a weapons.hlp file and a weapons.rtf file. You can use the weapons.rtf file as desired as a starting point to develop your own .hlp files. To do this, copy the weapons.rtf file into your help-authoring system and develop the topics you need..

Example: Building a .hlp File

For example, suppose your HTBasic for Windows program included a HELP FILE called "weapons.hlp" and a HELP TOPIC of "weapons.phasers" (HELP TOPIC topic_id). The program line may look something like this:

```
440 CONTROL @Main;SET ("HELP FILE":"weapons.hlp","HELP TOPIC": "weapons.phasers")
```

To build the corresponding .hlp file for this program, you would create a topic with context string of weapons.phasers, next create other topics as required, and then compile all topics into a weapons.hlp file. See the following figure for a typical display using the ForeHelp ® Help-Authoring System (other Help-Authoring systems may have a different display).


Topic Properties
Topic Title:
Phasers
Context String:
weapons.phasers
Context Number:
Context Map File:
RTF File:
Copy Title To
Context String
Topic
Keyword
Banner
OK
Cancel
Help

LABEL

The value of this attribute is a string that is used as the header on the CASCADE MENU.

SENSITIVE

This attribute provides you with the capability to keep the LABEL of the CASCADE MENU widget visible, but make it unresponsive to user input. If the value of SENSITIVE is 1, the widget is responsive to user input. If the value is 0, the widget will not respond to user input, and will appear "grayed out" to indicate its unresponsive status.

USER DATA

The contents of this attribute are left to the programmer and are never changed by HTBasic for Windows. It is simply a way to associate program-specific data with a specific widget.

CASCADE MENU Widget

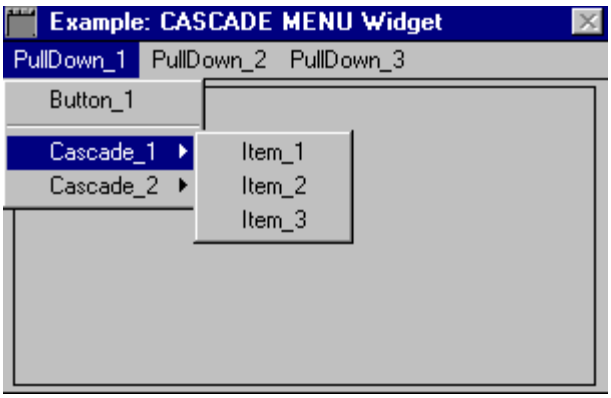
Provides a menu of selections within a PULLDOWN MENU (or other CASCADE MENU) from which the user can choose

---

Legal Usage

Level-0 Widget:	No
Parent to:	MENU BUTTON, MENU TOGGLE, MENU SEPARATOR, CASCADE MENU
Child of:	PULLDOWN MENU, CASCADE MENU

Example Image



Example Program

See the *Menu System* program for an example program that shows how to create a CASCADE MENU widget.

NOTE

See the following programs for other examples using the CASCADE MENU widget:

Environmental Chamber
Function Generator

Attributes

See CASCADE MENU Widget Attributes for CASCADE MENU widget attribute list.

Remarks

The CASCADE MENU widget is used to provide a menu (within the PULLDOWN MENU or other CASCADE MENU) of selections from which the user can choose.

The CASCADE MENU header does not appear in the menu bar, but does appear as one of the selections in a PULLDOWN MENU or other CASCADE MENU. To activate a CASCADE MENU, use one of these two procedures:

- Press the mouse button when the cursor is in the boundary of the PULLDOWN MENU's label area and hold. Then, release the mouse button when the mouse cursor is within the CASCADE MENU borders

- Press the mouse button when the cursor is in the boundary of the PULLDOWN MENU's label area and release. Then, click the mouse button when the mouse cursor is within the CASCADE MENU borders.

Events

None.
.

HTBasic for Windows

CASCADE MENU Widget Attributes

CASCADE MENU Widget Attributes

Attribute	Type	Allowed Values	Default
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
LABEL	String	Any	Null String
SENSITIVE	Numeric	0,1	1
USER_DATA	String	Any valid string	None

ALARM TIME

This attribute sets the time that the alarm will be generated. This time has the format "hh:mm:ss" and uses a 24 hour clock. Setting an alarm time enables the alarm. To disable the alarm, this attribute should be set to the value "0". When the alarm time is reached, this attribute is set to "0" and the alarm is disabled.

FONT

The value is the font that will be used for the text in the CLOCK widget. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact. For a description of allowed fonts, see Specifying FONT Attribute Values.

MILITARY TIME

Sets clock time interval. 0 = standard time (12-hour interval),
1 = military time (24-hour interval).

PEN

The value of this attribute specifies the pen color that will be used to paint the CLOCK. For a discussion of allowed pens, see Settable Pens Table.

SECONDS VISIBLE

This attribute controls the visibility of the seconds time information of the CLOCK.

TIMER DIRECTION

This attribute specifies whether the timer counts up or down.

TIMER REPEAT

This attribute specifies whether the timer repeats after the TIMER VALUE expires.

TIMER STATE

This attribute determines the state of the timer:

- "RUNNING" starts the timer
- "STOPPED" stops the timer
- "RESET" returns it to its original value, either "0" or "TIMER VALUE".

TIMER UPDATE

This attribute sets the update rate of the timer in milliseconds.

TIMER VALUE

This attribute specifies the duration of the timer in milliseconds.

TYPE

This attribute sets the type of clock to be displayed.

- The ANALOG clock has major tick marks at 5 minute intervals and minor tick marks at 1 minute intervals.
- The DIGITAL clock displays a digital clock readout.
- The MIXED clock displays both ANALOG and DIGITAL presentation.
- The TIMER clock displays either a countdown or count up digital timer.

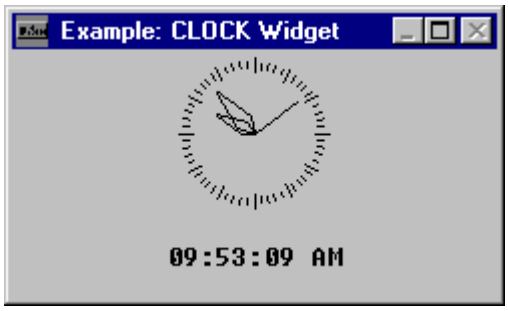
CLOCK Widget

Graphically displays time information in 24-hour clock or count up/ count down timer mode

---

Legal Usage	Level-0 Widget	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See CLOCK Widget for an example program that provides a display similar to that shown above.

NOTE

See the following programs for other examples using the CLOCK widget:

Alarm Clock
*Bomb Squad (*LOAD)*

Attributes

See CLOCK Widget Attributes for the CLOCK widget attribute list.

Remarks

The CLOCK widget graphically displays time information in 24-hour clock or

count up/count down timer mode. CLOCK can generate analog, digital, mixed, or timer displays.

The mode of operation is set by the TYPE attribute, which can be ANALOG, DIGITAL, MIXED (both ANALOG and DIGITAL), or TIMER. You can set an ALARM event on the CLOCK and specify an ALARM TIME (in the form of an "HH:MM:SS" string) for the event to occur.

In TIMER mode, a digital timer is created that counts from a time in milliseconds down to 0, or the reverse. You can:

- Specify the count direction with TIMER DIRECTION
- Specify the initial count with TIMER VALUE
- Start the count by setting TIMER STATE to RUNNING
- Stop the count by setting TIMER STATE to STOPPED
- Clear the timer by setting TIMER STATE to RESET

A TIMER event occurs when the count completes. You can perform TIMER actions cyclically by setting TIMER REPEAT to 1.

Events

Events for the CLOCK widget are:

- ALARM
- SYSTEM MENU
- TIMER

ALARM

This event is generated when the time specified by the ALARM TIME attribute is reached by the CLOCK.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

TIMER

This event is generated each time the TIMER expires (reaches "0" in the countdown mode or TIMER VALUE in the count up mode).

HTBasic for Windows

CLOCK Widget Attributes

CLOCK Widget Attributes

Attribute	Type	Allowed Values	Default
ALARM TIME	String	Valid time format	0
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
FONT	String	*Font* [1]	System dependent [2]
HEIGHT	Numeric	Any integer number (of pixels)	300 pixels
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
MAXIMIZABLE [4]	Numeric	0,1	1 (maximizable, button appears in title bar)
MILITARY TIME	Nu	0,1	0

	meric		
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SECONDS VISIBLE	Boolean	0,1	1
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeri	0 to number of items in system menu	0

	c		
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TIMER DIRECTION	Discrete string	"DOWN","UP"	"DOWN"
TIMER REPEAT	Boolean	0,1	0
TIMER STATE	Discrete string	"RESET", "STOPPED", "RUNNING"	"RESET"
TIMER UPDATE	Numeric	1 to 32767	1000
TIMER VALUE	Long integer	0 to 2147483647	0
TITLE [5]	String	Any valid string	CLOCK
TYPE	Discrete string	"ANALOG", "DIGITAL", "MIXED" ' "TIMER"	"MIXED"
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
VISIBLE	Nu	0,1	1

	meric		
WIDTH	Numeric	Any integer number (of pixels)	150 pixels
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

COLUMNS

This attribute specifies the number of columns (that is, the number of character cells) that will appear in the COMBO widget or dialog. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact.

DROPDOWN BUTTON

This attribute specifies the presence of a button that controls the visibility of the listbox. Without this button, the listbox is always displayed.

EDITABLE

This attribute determines whether the listbox is editable or not.

- If set to "1", you can type into the edit box at the top
- If set to "0", it displays the function selected from the listbox.

FONT

This attribute changes the font in the listbox. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact. For a description of allowed fonts, see Specifying FONT Attribute Values.

ITEM COUNT

Return-Only Attribute. ITEM COUNT returns the number of items in the listbox.

ITEMS

This array is used to populate the listbox with items. Each string in the array becomes a separate item (line) in the listbox portion of the widget or dialog.

LIST BACKGROUND

This attribute determines the color of the list background. For a discussion of allowed pens, see Settable Pens Table.

LIST PEN

This attribute determines the color of the list items. For a discussion of allowed pens, see Settable Pens Table.

PEN

This attribute determines the color of text in the selection field of the widget or dialog. For a discussion of allowed pens, see Settable Pens Table

ROWS

The number of rows specifies the number of character cells that fit within the height of the listbox. ROWS does not affect or include the selection field. It only affects widget or dialog height if DROPDOWN BUTTON is set to "0". See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, and FONT interact.

SCROLLBAR

This attribute specifies whether the listbox has its scrollbar displayed.

SELECTION

This attribute is the index of the element in the ITEMS array that the operator has selected. "-1" indicates no selection. For example, "SELECTION":X directs HTBasic for Windows to return the operator's selection in the variable "X".

Because all arrays are treated as a single-dimension, BASE 0 array, the SELECTION attribute specifies an offset, not necessarily the actual subscript, of the selected element in the ITEMS array.

SENSITIVE

This attribute provides the capability to keep the COMBO widget or dialog visible, but make it unresponsive to user input. If the value of SENSITIVE is 1, the widget or dialog is responsive to user input. If the value is 0, the widget or dialog will not respond to user input, and will appear "grayed out" to indicate its unresponsive status.

SHOW LIST

Set-Only Attribute. SHOW LIST determines whether or not the listbox is shown. Setting SHOW LIST to "1" makes the COMBO widget or dialog behave as if the operator had clicked the DROPDOWN button.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

TEXT

The TEXT attribute sets and returns the string in the selection field.

HTBasic for Windows

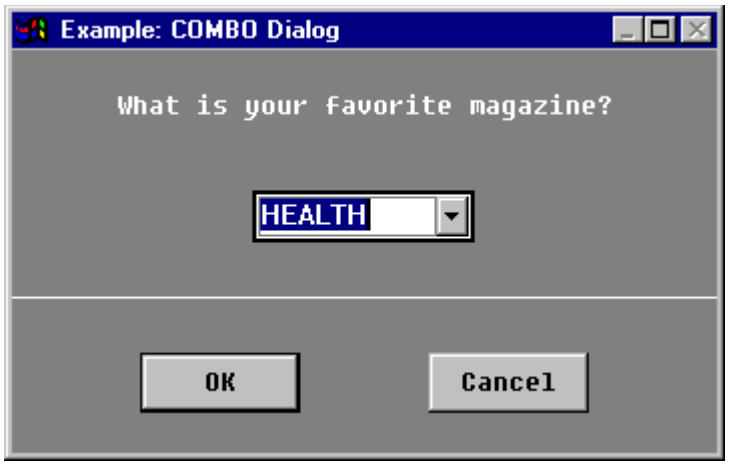
COMBO Dialog

COMBO Dialog

Prompts the user to select from a list of entries or to enter a selection

---

Example Image



Example Program

See COMBO Dialog for an example program that produces a display similar to that shown above.

NOTE

See the following program for another example using the COMBO dialog.

Dialogs Tests

Attributes

See COMBO Dialog Attributes for the COMBO dialog attribute list

Remarks

None.

HTBasic for Windows

COMBO Dialog Attributes

COMBO Dialog Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
COLUMNS	Numeric	0 to 32767 (characters)	10
DEFAULT BUTTON	Numeric	Any valid index into the DIALOG BUTTONS array	0
DIALOG BUTTONS	String array	Any valid string array	A two-element array that contains "OK" and "Cancel"
DROPDOWN BUTTON	Numeric	0,1	1
EDITABLE	Numeric	0,1	1
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
ITEM COUNT	Nu	0 to 32767	0

	meric		
ITEMS	String array	Any	A zero element array
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
LIST BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
LIST PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE	Numeric	0,1	1
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE	Numeric	0,1	1 (resizable, special border appears around the dialog)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)

ROWS	Numeric	0 to 32767 (characters)	5
SCROLLBAR	Numeric	0,1	1
SELECTION	Numeric	Valid index into the ITEMS array, or -1 for no selection. Always 0 based.	-1
SENSITIVE	Numeric	0,1	1 (sensitive)
SHOW LIST	Numeric	0,1	0
TEXT	String	Any	Null string
TITLE	String	Any valid string	COMBO
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Screen or parent work area upper-left corner

COMBO Widget

Creates a scrollable list of items

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image

Example Program

See *COMBO Widget* for an example program that creates a COMBO widget with a display similar to that shown above. In this program, the SELECTION and RETURN events are used to call a handler routine that gets the TEXT returned by the widget, while the handler searches through the ITEMS array attempting to find a match.

NOTE

See the following program for another example using the COMBO widget:

COMBO Test

Attributes

See COMBO Widget Attributes for the COMBO widget attribute list.

Remarks

The COMBO widget is a combination of the STRING widget and the LIST widget that creates a scrollable list of items. The COMBO widget is similar to the LIST widget in that the current selection is displayed in the listbox.

The COMBO widget allows you to present a list of ITEMS to the user, and to return text through the TEXT attribute from that list or a string input by the user. The COMBO widget uses events to detect when it is activated.

When a selection from the listbox is complete, the listbox is hidden, leaving only the selection field and the listbox actuator remaining.

The vertical resize borders act differently for this widget than it does for others. Instead of changing the widget's height, the listbox proportion changes. The same is true for HEIGHT and ROWS attributes.

To employ the COMBO display resizing corners, pull the rubberband box to the size you choose, and release the mouse button. When the listbox appears, it will appear at that size. Only the vertical scrollbar appears in this widget - there is no horizontal scrollbar.

Events

Events for the COMBO widget are:

- KEYSTROKE
- RETURN
- SELECTION
- SYSTEM MENU

KEYSTROKE

Occurs when you type a character in the selection field.

RETURN

Occurs when you press the **Return** key.

SELECTION

Occurs when you select an item from the listbox.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

COMBO Widget Attributes

COMBO Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
COLUMNS	Numeric	0 to 32767 (characters)	10
DROPDOWN BUTTON	Numeric	0,1	1
EDITABLE	Numeric	0,1	1
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Num	Any integer number (of pixels)	Varies

	eric		
ITEM COUNT	Numeric	0 to 32767	0
ITEMS	String array	Any	A zero element array
LIST BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
LIST PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeri	0,1	0 (do not restore the screen)

	c		
ROWS	Numeric	0 to 32767 (characters)	5
SCROLLBAR	Numeric	0,1	1
SELECTION	Numeric	Valid index into the ITEMS array, or -1 for no selection. Always 0 based.	-1
SENSITIVE	Numeric	0,1	1 (sensitive)
SHOW LIST	Numeric	0,1	0
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1

TEXT	String	Any	Null string
TITLE [5]	String	Any valid string	COMBO
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

HTBasic for Windows

CONFIG File Features

With context-sensitive help, when you click on a selected widget with the right mouse button, the help topic is automatically displayed. The help file and help topic for the widget are defined by the HELP FILE and HELP TOPIC attributes. You can use the following CONFIG file features to control the behavior of context-sensitive help. See Using the CONFIG file for details on setting context help for widgets with the CONFIG command.

f1_is_help	Press **F1** key for help access
rht_ms_btn_is_help	Click right mouse button for help access
provide_defaults	The default HELP TOPIC for a widget is its name

HTBasic for Windows

CONFIG File Overview

HTBasic for Windows operation can be customized by changing the parameters in a configuration file called CONFIG. You can change parameters for system MSI, mouse settings, pen colors, and many others.

The CONFIG file is a regular text file that can be modified in the editor of your choice (including BASIC since each line is prefixed with a line number and comment token ("!")). When edited outside of BASIC, the line numbers and comment tokens are optional. The file must be resaved as regular text.

NOTE

Be careful when making changes to the CONFIG file, as invalid values can render HTBasic for Windows almost unusable. Before modifying any values, be sure you understand the effects of your changes.

HTBasic for Windows

CONFIG File Sections

This topic describes CONFIG file sections including:

- System MSI
- System Font
- SLIDER Widget Delays
- Context Help for Widgets
- Screen Builder Widget Order
- Customizing Pens

Click the section name to view each CONFIG file section.

HTBasic for Windows

CONTROL Command

CONTROL

Sets the value(s) of specified attribute(s) for the specified widget

---

NOTE

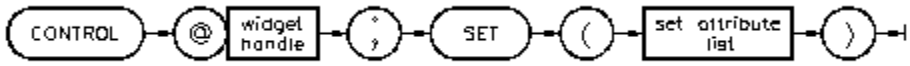
*Click the **>>** bar for additional information on the CONTROL command.*

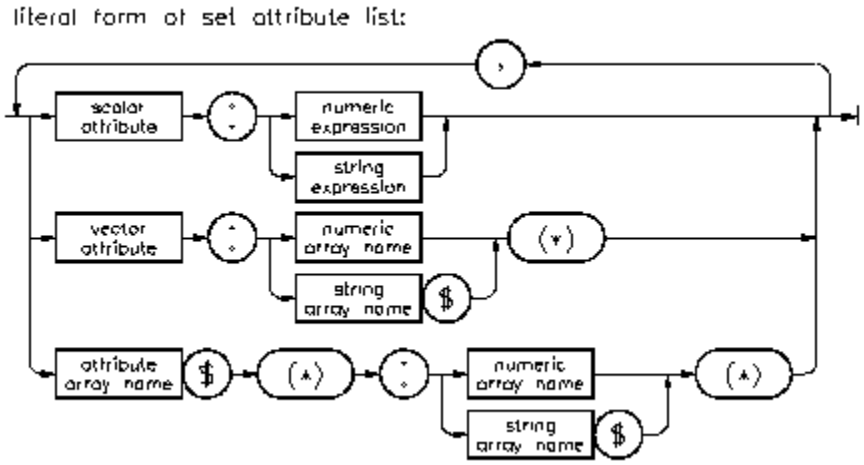
Legal Usage	Keyboard Executable	Yes
	Programmable	Yes
	In an IF...THEN...	Yes

Example Statements

```
CONTROL @Strip2;SET ("CURRENT AXIS":"X",
RANGE":20)
CONTROL @Slider;SET ("VALUE":Setpoint)
```

Syntax





Item	Description	Range
attribute array	an array of single-valued widget attributes	depends on
name	Both the attribute array and either the	widget

	numeric array or string array must have identical dimensions	
numeric array	array of values to be assigned to this	depends on
name	attribute or these attributes	attribute(s)
numeric expression	expression containing the value to be assigned to this attribute	depends on attribute
scalar attribute	a single-valued widget attribute (string expression)	depends on widget
string array	array of values to be assigned to this	depends on
name	attribute or these attributes	attribute(s)
string expression	expression containing the value to be assigned to this attribute	depends on attribute
vector attribute	a multi-valued widget attribute (string expression)	depends on widget
widget handle	name identifying a widget	any valid name
widget type	string expression containing the type of widget to be created	see Description

CONTROL Command (continued)

Description

Each widget has a variety of attributes that control its appearance and behavior. The CONTROL command is used to assign a new value to a widget attribute. The widget must have been created previously using an ASSIGN statement. Attributes are either scalar (may contain a single value) or vector (may be assigned an array of values) and have values of either numeric or string type.

You can use a shorthand method to assign values to several scalar attributes without naming them individually on the ASSIGN statement. to do this, you store all the attributes in a string array and all the matching values in another array of the same size.

Then, when you specify both array names in the SET option of the ASSIGN statement, the attribute named in each element of the string array will be assigned the corresponding value in the value array. Elements of the string array that contain nothing, or nothing but blanks, will be ignored. For example:

```
Attribs$(1) = "X"
Attribs$(2) = "Y"
Attribs$(3) = "WIDTH"
Attribs$(4) = "HEIGHT"
Values(1) = 5
Values(2) = 5>
Values(3) = 500
Values(4) = 300

CONTROL @Panel;SET (Attribs$(*):Values(*))
```

HTBasic for Windows

Changing Widget Attribute Values

Ways to Change Widget Attribute Values

Three ways to change attribute values using the CONTROL command are:

- You can change an attribute value with a single CONTROL statement. For example:

```
CONTROL @Meter;SET("VALUE":22)
```

- You can also change multiple attribute values with multiple CONTROL statements. For example:

```
CONTROL @Bars;SET("BACKGROUND":5)
CONTROL @Bars;SET("SHOW LIMITS":0)
CONTROL @Bars;SET("VALUE":44)
```

- You can also change multiple attribute values with a single CONTROL statement. For example:

```
CONTROL @Bars;SET("BACKGROUND":5,"SHOW LIMITS":0,"VALUE":44)
```

Example: Changing Widget Attributes

For example, for a METER widget you can change the value of the common attribute BACKGROUND using the ASSIGN command as follows, where "2" sets the background color to red:

```
10  ASSIGN @Meter TO WIDGET "METER";SET ("BACKGROUND":2)
20  LOOP
30  END LOOP
40  END
```

Or if you prefer, the BACKGROUND attribute may be SET with the CONTROL statement:

```
10  ASSIGN @Meter TO WIDGET "METER"
20  CONTROL @Meter;SET ("BACKGROUND":2)
30  LOOP
40  END LOOP
50  END
```

Choosing Widget Attribute Order

The order in which attributes are specified following SET in a CONTROL or ASSIGN command is the order in which they are set. Proper ordering of attributes in the statement affects program behavior. Generally, you should SET attributes in a hierarchy from the general to the specific.

For example, in a BARS widget BAR COUNT should be set before CURRENT BAR, as shown in the following line. Otherwise, CURRENT BAR might reference a bar that does not exist, causing an error.

```
CONTROL @Bars;SET ("BAR COUNT":3,"CURRENT BAR":,2)
```

HTBasic for Windows

HTBasic Plus Contents

HTBasic Plus

- Using HTBasic Plus
- HTBasic Plus Programming
- HTBasic Plus Reference

HTBasic for Windows

- HTBasic for Windows

Distributed with release 8.3

HTBasic for Windows

CONFIG File Sections - Context Help for Widgets

Introduction

Context help is information about the widget that is accessed by the user directly from the widget. (see Using Context-Sensitive Help for details) The following table lists the parameters that specify how context help is set and accessed. Help is available for all widgets except the MENU SEPARATOR widget.

Parameter	Setting	Function
provide_defaults	true	Default widget's HELP TOPIC is widget's name
f1_is_help	true	F1 key provides help for widget with focus
rht_ms_btn_is_help	true	Right mouse button provides widget help

HELP TOPIC and HELP FILE Attributes

Each widget has HELP TOPIC and HELP FILE attributes.

- If *provide_defaults* is set to true (the default setting), HELP TOPIC defaults to the widget's name and HELP FILE defaults to the NULL string which indicates to use HTBasic for Windows' help file. Thus, the help provided is how to program with the widget.
- If *provide_defaults* is not set (or set to "false"), there is no help provided. Thus, when the operator selects context Help on that widget, an INFORMATION dialog appears, indicating help is not available for that widget .

Selecting Context Help

Context help is selected in one of the following two ways. You can specify which method(s) are available with *f1_is_help* and *rht_ms_btn_is_help*. If both are set to false, context help will not be provided, regardless of HELP TOPIC and HELP FILE settings.

- Clicking on the widget with the right mouse button
- Focusing on the widget (with the mouse or through keyboard traversal) and pressing

the **F1** function key

HTBasic for Windows

Copying Example Programs

This topic shows how to copy example programs from online Help into the HTBasic for Windows Editor. See Example Programs for a list of the example programs in online Help. There are three main procedures to transfer example programs from online Help into the HTBasic for Windows Editor:

- Using a Generic Printer
- Using the plus examples Directory

Click the item title for information on each method.

NOTE

To add HTBasic Plus extensions to HTBasic for Windows, execute the following statement from the command line or programmatically:

LOAD BIN "BPLUS"

HTBasic for Windows

Creating Pulldown Menus

Overview

This topic gives guidelines to create HTBasic Plus pulldown menus using the Menu Creation widgets.

Menu Creation Widgets

The widgets used to build pulldown menu systems for HTBasic for Windows HTBasic Plus are listed in the following table. Click the widget name for details on each widget.

Widget	Function
PULLDOWN MENU	Creates an "attachment point" for a pulldown menu
CASCADE MENU	Creates an "attachment point" for a cascade menu
MENU BUTTON	Creates an event when click
MENU TOGGLE	Creates an event when clicked and has a VALUE
MENU SEPARATOR	Draws a line inside a menu

Example: Creating a Pulldown Menu System

We will use an example to show how to use Menu Creation widgets and their attributes to create a pulldown menu system. As desired, you can copy and run the Menu System example to use as a reference as you go through this example. Click the **>>** and **<<** bars to view the steps in this example.

SYSTEM MENU is used for this example. SYSTEM MENU is accessed by clicking on the small button on the upper left corner of the window, and is typically used to exit the program. The SYSTEM MENU is an attribute of many widgets, but is not a widget itself. See Creating System Menus for details on SYSTEM MENU.

Creating Pulldown Menus

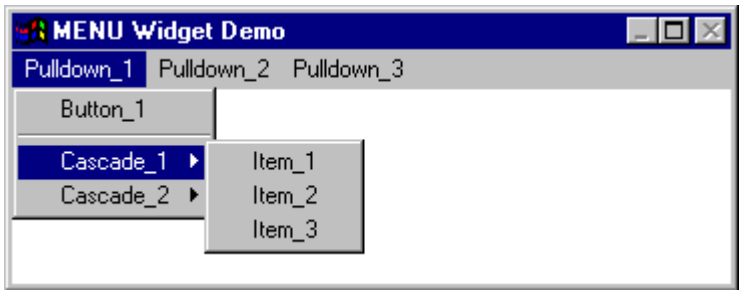
Step 4: Program the Parts (continued)

Create Cascade_1 Menu

This still leaves the Cascade_1 and Cascade_2 menus pending. We will complete Cascade_1 first by assigning its MENU BUTTON entries. This results in the following display.

```
1480    ! Create menu for "Cascade_1" (using @C1 as PARENT).
1490    !
1500    S$="Item_1"
1510    ASSIGN @B131 TO WIDGET "MENU BUTTON";PARENT @C13,SET ("LABEL":S$)
1520    !
1530    S$="Item_2"
1540    ASSIGN @B132 TO WIDGET "MENU BUTTON";PARENT @C13,SET ("LABEL":S$)
1550    !
1560    S$="Item_3"
1570    ASSIGN @B133 TO WIDGET "MENU BUTTON";PARENT @C13,SET ("LABEL":S$)
1580    !
```

Cascade_1 Menu Display



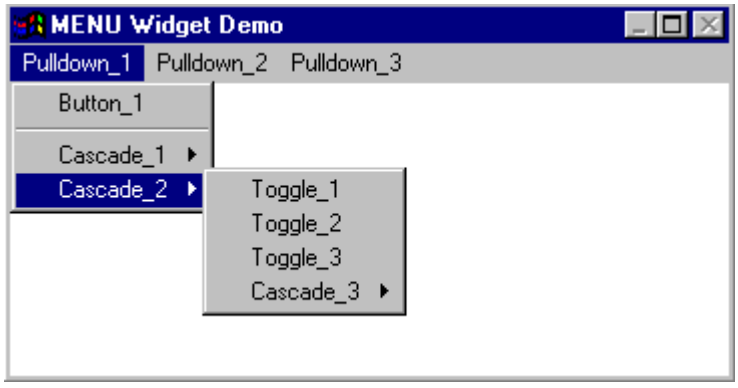
Create Cascade_2 Menu

The Cascade_2 menu is populated using the techniques shown so far, and adding MENU TOGGLE widgets. This results in the following display.

```
1590    ! Create menu for "Cascade_2" (using @C14 as PARENT).
1600    !
1610    S$="Toggle_1"
```

```
1620   ASSIGN @T141 TO WIDGET "MENU TOGGLE";PARENT @C14,SET ("LABEL":S$)
1630   !
1640   S$="Toggle_2"
1650   ASSIGN @T142 TO WIDGET "MENU TOGGLE";PARENT @C14,SET ("LABEL":S$)
1660   !
1670   S$="Toggle_3"
1680  ASSIGN @T143 TO WIDGET "MENU TOGGLE";PARENT @C14,SET ("LABEL":S$)
1690   !
1700   ! Add "Cascade_3" as entry in "Cascade_2".
1710   !
1720   S$="Cascade_3"
1730   ASSIGN @C144 TO WIDGET "CASCADE MENU";PARENT @C14,SET ("LABEL":S$)
1740   !
```

Cascade_2 Menu Display



Creating Pulldown Menus

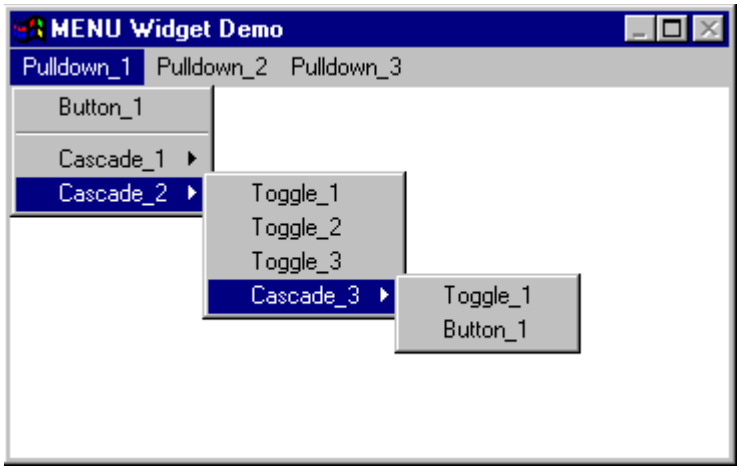
Step 4: Program the Parts (continued)

Create Cascade_3 Menu and Completed Menu

To complete the programming of the pulldown menu system, we populate the menu for Cascade_3 as follows. This results in the following display that shows the completed pulldown menu system for this example.

```
1750    ! Populate menu for "Cascade_3".
1760    !
1770    S$="Toggle_1"
1780    ASSIGN @T1441 TO WIDGET "MENU TOGGLE";PARENT @C144,SET ("LABEL":S$)
1790    !
1800    S$="Button_1"
1810    ASSIGN @B1442 TO WIDGET "MENU BUTTON";PARENT @C144,SET ("LABEL":S$)
1820    !
1830    ! ************************************************
```

Cascade_3 Menu and Completed Pulldown Menu System Display



HTBasic for Windows

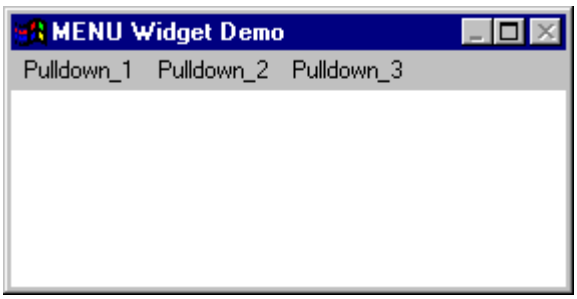
Creating Pulldown Menus

Step 1: Describe System Layout

We will begin the design by outlining the desired menu system and the functions to be performed by each of the menu elements. The menu system resides in a PANEL that contains a PRINTER output widget.

When you select an element of a menu a string indicating the menu selection is dumped to the PRINTER widget. This feedback mechanism allows you to verify if the menus are working as expected. The following figure shows the desired main panel with menus and elements of each pulldown menu.

Main Panel Display



The menu widget PullDown_1 expands as follows:

- Button_1
- Separator
- Cascade_1
  Item_1
  Item_2
  Item_3
- Cascade_2
  Toggle_1
  Toggle_2 Toggle_3
  Cascade_3
  Toggle_1
  Button_1

Cascade_1 and Cascade_2 are "attachment points" for other menus. The elements labeled Toggle_1, Toggle_2, etc. should toggle when selected, which is indicated by the presence or absence of a "checkmark" in front of the button label.

The second top-level menu PullDown_2 expands into:

- Button_1
- Button_2

The third top-level menu PullDown_3 contains:

- Button_1
- Button_2
- Button_3
- Separator
- Quit

The Separator in the first and third pulldown menus performs no other function than visually separating the entries of those menus.

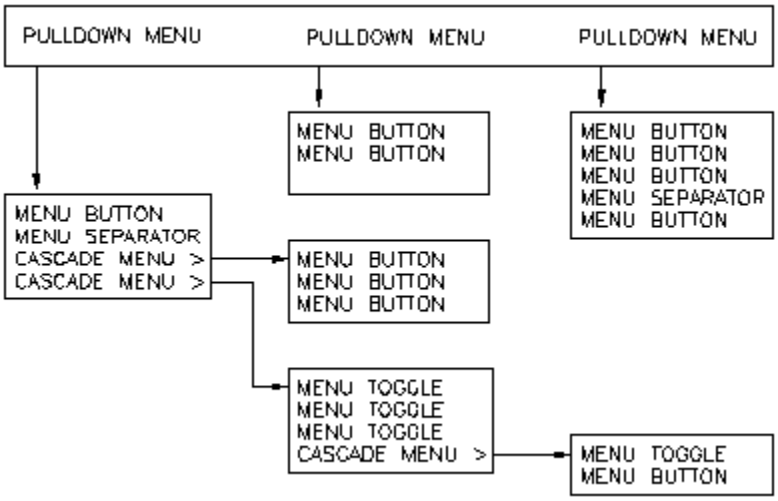
Creating Pulldown Menus

Step 2: Identify the Widgets

The menu system shown in Step 1 (ignoring the SYSTEM MENU) is implemented with the widgets shown in the following figure. To create the menu system, create all these widgets and add the appropriate labels for each one. Each widget has its own name, allowing each to interact individually with the BASIC program that creates it.

These widgets do not need to be created in any particular order. You define parent-child relationships between the widgets, and HTBasic for Windows creates the menu system based on those relationships, as well as the sequence in which you create the widgets.

Widgets to Implement the Example Pulldown Menu System



HTBasic for Windows

Creating Pulldown Menus

Step 3: Define the Hierarchy

For this example pulldown menu system, the widgets have the relationships shown in the following table (where the children of a widget are indented from the parent). This table defines the widgets in the menu system, outlines widget hierarchical relationships, and assigns a label and widget handle (widget name) to each widget.

Widget Relationships

Widget	menu item	label
PANEL		@P
PRINTER		@Prn
PULLDOWN MENU	"PullDown_1"	@Pd1
MENU BUTTON	"Button_1"	@B11
MENU SEPARATOR	----------	@S12
CASCADE MENU	"Cascade_1"	@C13
MENU BUTTON	"Item_1"	@B131
MENU BUTTON	"Item_2"	@B132
MENU BUTTON	"Item_3"	@B133
CASCADE MENU	"Cascade_2"	@C14
MENU TOGGLE	"Toggle_1"	@T141
MENU TOGGLE	"Toggle_2"	@T142
MENU TOGGLE	"Toggle_3"	@T143
MENU TOGGLE	"Cascade_3"	@C144
CASCADE MENU	"Toggle_1"	@T1441
MENU TOGGLE	"Button_1"	@T1442
MENU BUTTON		
PULLDOWN MENU	"PullDown	@Pd2

MENU BUTTON	_2"	@B21
MENU BUTTON	"Button_1"	@B22
MENU BUTTON	"Button_2"	@B23
	"Button_3	
PULLDOWN MENU	"PullDown	@Pd3
MENU BUTTON	_3"	@B31
MENU BUTTON	"Button_1"	@B32
MENU BUTTON	"Button_2"	@B33
MENU	"Button_3"	@S34
SEPARATOR	----------	@Quit
MENU BUTTON	--	
	"QUIT"	

It is the hierarchical relationships of the widgets and the order in which they are created that defines the layout of the menu system. However, note the requirement for ordering widgets to make sure they are in the proper sequence on the display only applies to each level of the menu system.

For example, consider the top-level PULLDOWN MENU widgets. You could create the widgets following the order in the table, and the three would be in the proper order. Or, you could create the three PULLDOWN MENU widgets and then go back and populate the lower-level widgets. The only requirement is that PullDown_1 is created before PullDown_2, and that PullDown_2 is created before PullDown_3.

Creating Pulldown Menus

Step 4: Program the Parts

Program statements follow to create this menu system. The program lines listed refer to the lines in the Menu System example program.

NOTE

You can also build pulldown menus using the Screen Builder Application.

The widgets that are created for this example are (in order):

- Parent PANEL widget
- Topmost PULLDOWN MENU widgets
- PRINTER widget
- PullDown_3 Menu
- PullDown_1 Menu
- Cascade_1 Menu
- Cascade_2 Menu
- Cascade_3 Menu

Create the Parent PANEL Widget

```
420   !
430   ! Note that the main panel is set up so it automatically scales to the
440   ! entire display, except for the DISP, INPUT, and softkeys lines.
450   !
460   DIM S$[80],M$(0:2)[80]
470   INTEGER Cursor,State,N,D(1:4)
480   INTEGER Dw,Dh,X,Y
490   INTEGER Panelwidth, Panelheight, Prx, Pry, Prwidth, Prheight, Iw, Ih
500   !
510   GESCAPE CRT,3;D(*) ! Get display resolution.
520   Dw=D(3)-D(1)+1
530   Dh=D(4)-D(2)+1
540   Panelwidth=Dw*.75 ! Size PANEL.
550   Panelheight=Dh*.75
560   X=(Dw-Panelwidth)/2
570   Y=(Dh-Panelheight)/2
580   !
590   STATUS CRT,10;Cursor ! Turn off cursor, etc.
600   CONTROL CRT,10;0
```

```
610   RUNLIGHT OFF
620   KEY LABELS OFF
630   !
640   ! ************************************************
650   !
660   ! Create the PANEL widget.
670   !
680   CLEAR SCREEN
690   ASSIGN @P TO WIDGET "PANEL";SET ("VISIBLE":0)
700   CONTROL @P;SET ("RESIZABLE":1)
710   CONTROL @P;SET ("X":X,"Y":Y,"WIDTH":Panelwidth,"HEIGHT":Panelheight)
720   CONTROL @P;SET ("TITLE":"MENU Widget Demo")
730   CONTROL @P;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
740   !
```

Creating Pulldown Menus

Step 4: Program the Parts (continued)

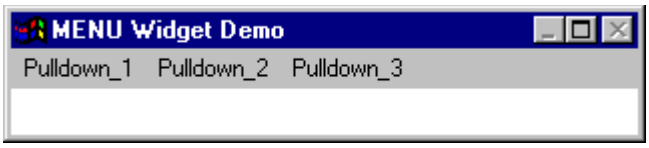
Create Topmost PULLDOWN MENU Widgets

Now that the background is in place, we can start building the menu system itself. First we create the topmost PULLDOWN MENU widgets as follows:

```
750   ! ************************************************
760   !
770   ! Create the PULLDOWN MENU widgets, using the PANEL widget as parent.
780   !
790   S$="PullDown_1"
800   ASSIGN @Pd1 TO WIDGET "PULLDOWN MENU";PARENT @P,SET ("LABEL":S$)
810   !
820   S$="PullDown_2"
830   ASSIGN @Pd2 TO WIDGET "PULLDOWN MENU";PARENT @P,SET ("LABEL":S$)
840   !
850   S$="PullDown_3"
860   ASSIGN @Pd3 TO WIDGET "PULLDOWN MENU";PARENT @P,SET ("LABEL":S$)
870   !
880   ! ************************************************
```

The top-level menu automatically appears in a bar across the PANEL widget (see the following figure). Telling HTBasic for Windows to create a PULLDOWN MENU widget in a PANEL automatically creates a menu bar, and the PULLDOWN MENU entries are loaded into the bar in the order in which they are created.

Top-Level Menu Display



HTBasic for Windows

Creating Pulldown Menus

Step 4: Program the Parts (continued)

Create a PRINTER Widget

Next, we set up the PRINTER widget. We had to wait until the menu bar was created to do this because we could not get a valid INSIDE WIDTH and INSIDE HEIGHT before that point.

```
880     ! ***************************************************
890     !
900     ! Now that you've set up the PULLDOWN MENU, get interior dimensions
910     ! of the PANEL and set up PRINTER widget accordingly.
920     !
930     STATUS @P;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
940     Prx=Iw*.02
950     Pry=Ih*.02
960     Prwidth=Iw*.96
970     Prheight=Ih*.96
980     ASSIGN @Prn TO WIDGET "PRINTER";PARENT @P
990     CONTROL @Prn;SET ("X":Prx,"Y":Pry,"WIDTH":Prwidth,"HEIGHT":Prheight)
1000    !
1010    ! ***************************************************
```

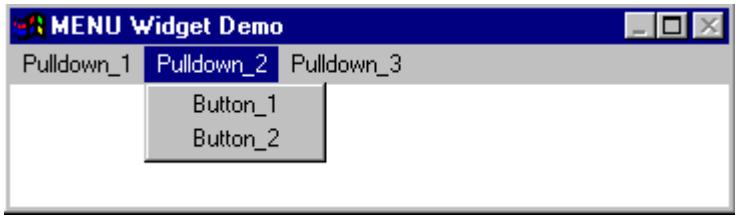
Create Pulldown_2 Menu

Next, we create the pulldown menus themselves. We start with the PullDown_2 menu. The following lines create the following display.

```
1010    ! ***************************************************
1020    !
1030    ! Create menu for "PullDown_2" (using @Pd2 as PARENT).
1040    !
1050    S$="Button_1"
1060    ASSIGN @B21 TO WIDGET "MENU BUTTON";PARENT @Pd2,SET ("LABEL":S$)
1070    !
1080    S$="Button_2"
1090    ASSIGN @B22 TO WIDGET "MENU BUTTON";PARENT @Pd2,SET ("LABEL":S$)
1100    !
```

```
1110    ! ***************************************************
```

Pulldown_2 Menu Display



HTBasic for Windows

Creating Pulldown Menus

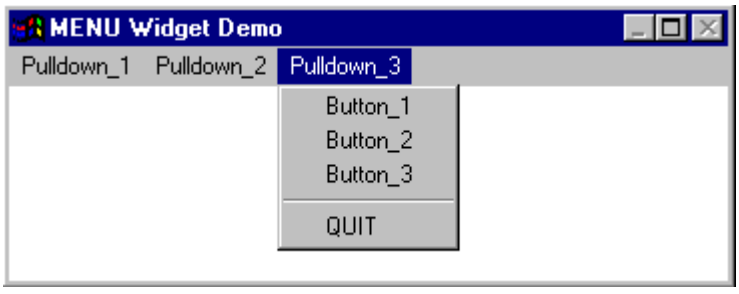
[Step 4: Program the Parts (continued)]

Create PullDown_3 Menu

We then create the PullDown_3 menu, which results in the display shown in the following figure. The "Quit" MENU BUTTON allows you to exit gracefully from this program.

```
1110    ! ******************************************************
1120    !
1130    ! Create menu for "PullDown_3" (using @Pd3 as PARENT).
1140    !
1150    S$="Button_1"
1160    ASSIGN @B31 TO WIDGET "MENU BUTTON";PARENT @Pd3,SET ("LABEL":S$)
1170    !
1180    S$="Button_2"
1190    ASSIGN @B32 TO WIDGET "MENU BUTTON";PARENT @Pd3,SET ("LABEL":S$)
1200    !
1210    S$="Button_3"
1220    ASSIGN @B33 TO WIDGET "MENU BUTTON";PARENT @Pd3,SET ("LABEL":S$)
1230    !
1240    ASSIGN @S34 TO WIDGET "MENU SEPARATOR";PARENT @Pd3
1250    !
1260    S$="Quit"
1270    ASSIGN @Quit TO WIDGET "MENU BUTTON";PARENT @Pd3,SET ("LABEL":S$)
1280    !
1290    ! ******************************************************
```

Pulldown_3 Menu Display



Creating Pulldown Menus

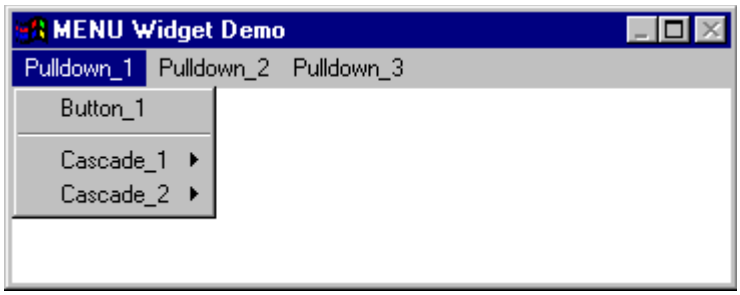
Step 4: Program the Parts (continued)

Create PullDown_1 Menu

The following lines create a MENU SEPARATOR widget and two CASCADE MENU widgets. The MENU SEPARATOR widgets keeps the MENU BUTTON visually distinct from the CASCADE MENU widgets below it. The MENU SEPARATOR is created like all other widgets and draws a line through the pulldown menu. These lines produce the following display.

```
1290    ! ***************************************************
1300    !
1310    ! Create menu for "PullDown_1" (using @Pd1 as PARENT).
1320    !
1330    S$="Button_1"
1340    ASSIGN @B11 TO WIDGET "MENU BUTTON";PARENT @Pd1,SET ("LABEL":S$)
1350    !
1360    ! Add menu separator.
1370    !
1380    ASSIGN @S12 TO WIDGET "MENU SEPARATOR";PARENT @Pd1
1390    !
1400    ! Add CASCADE MENU widgets to "PullDown_1" (using @Pd1 as PARENT).
1410    !
1420    S$="Cascade_1"
1430    ASSIGN @C13 TO WIDGET "CASCADE MENU";PARENT @Pd1,SET ("LABEL":S$)
1440    !
1450    S$="Cascade_2"
1460    ASSIGN @C14 TO WIDGET "CASCADE MENU";PARENT @Pd1,SET ("LABEL":S$)
1470    !
```

PullDown_1 Menu (CASCADE MENUs) Display


MENU Widget Demo
Pulldown_1
Pulldown_2
Pulldown_3
Button_1
Cascade_1
Cascade_2

HTBasic for Windows

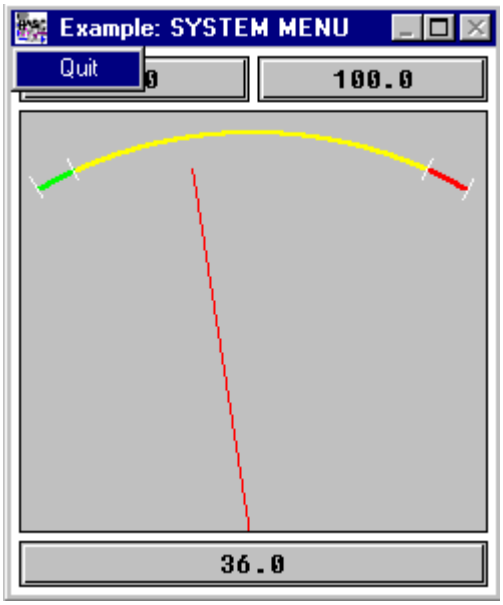
Creating System Menus

Generating a System Menu

The SYSTEM MENU is accessed by clicking on the small button on the upper left corner of the window, and is typically used to exit the program. The SYSTEM MENU is an attribute of many widgets, but is not itself a widget.

Example: Generating a System Menu

For example, the following figure shows a typical SYSTEM MENU for a METER widget. For this example, the SYSTEM MENU consists of the Quit action to allow the operator to exit this widget.
See Menu System (lines 1830 through 2830) for an example of creating a SYSTEM MENU.



HTBasic for Windows

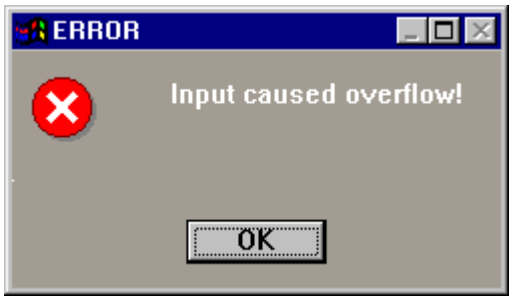
Creating/Modifying Dialogs

Creating Dialogs

To show how to create dialogs, we will use the DIALOG command to create an ERROR dialog. The ERROR dialog displays an error message and halts program execution until the user acknowledges the error. For the default ERROR dialog state, a single "OK" button is displayed.

When the following lines are typed, the display shown appears. This display remains (and program execution is halted) until the user clicks the OK button.

```
1     ! Example: Default ERROR Dialog
2     !
10    DIALOG "ERROR","Input caused overflow!"
20    END
```



For this example, the dialog type is "ERROR" (which must be entered in upper case) and the prompt string is "Input caused overflow!". The SET and RETURN attribute list default values are used.

Since no TIMEOUT timeout value is set, this dialog will remain on the screen until the user selects the OK button or presses the **Spacebar** or the **Return** key. The program then continues.

Modifying Dialogs

As desired, you can use the SET, RETURN, and TIMEOUT attributes to modify the dialog. To continue the ERROR dialog example, we will customize the dialog from the previous display to provide the user with some options.

When the following program is run, the display shown appears. Since TIMEOUT 5 is set, if the user makes no response within 5 seconds after

the dialog appears, the dialog automatically disappears and program execution resumes.


```
1         ! Example: Modifying ERROR Dialog
2         !
10        DIM S1$(0:1)[10]
20        INTEGER Btn
30        S1$(0)="Abort"
40        S1$(1)="Continue"
50        DIALOG "ERROR","Input caused overflow!",Btn;SET("DIALOG BUTTONS":S1$
(*)),TIMEOUT 5
60        IF Btn=-1 THEN                  ! Btn=-1 indicates TIMEOUT
70            DISP "Timeout"
80        ELSE
90            DISP S1$(Btn)
100       END IF
110       END
```

In addition, two new buttons (Abort and Continue) have been added to replace the OK button. The SET statement invokes the DIALOG BUTTONS attribute and string array S1$. S1$ contains the names of buttons to use as well as the button value.

When the dialog pops up, the mouse pointer appears on top of the Abort button (the DEFAULT BUTTON for the ERROR dialog), allowing the user to respond by pressing the **Spacebar** or the **Return** key. You can move the pointer to the Continue button with the Tab key or the mouse and click, at which point the display disappears.

You can determine which button was selected by the user by querying the value of the selected button variable "Btn" (0 for Abort, 1 for Continue, or -1 for TIMEOUT). For dialogs, 0 always refers to the leftmost button,

even if the button labels can be changed.

To find out which button the user pressed, DISP S1$(Btn) queries the selected button variable Btn. This variable contains the index of the button that was selected. For example, if the user clicks on the Abort button, the dialog disappears and the program displays "Abort". If the user clicks on Continue, the program displays "Continue".

You can use the **Tab** key to move from one button to the other and then press the **Spacebar** or the **Return** key to click the button, and you will get an index value back. For two or more buttons, you must use the DEFAULT BUTTON attribute. You can set this attribute to the value of the button index you want as the default.

HTBasic for Windows

Creating/Modifying Widgets

Overview

This topic gives guidelines to create and modify widgets using the ASSIGN command, including:

- Creating Widgets With ASSIGN
- Creating a Default Widget
- Preventing a Widget From Disappearing

NOTE

You can also create widgets and set widget attributes using the Screen Builder Application.

Creating Widgets With ASSIGN

Widgets are created and destroyed with the ASSIGN command. The ASSIGN command operates without suspending program execution. Therefore, widgets can display information on the screen as the program continues to execute. See the ASSIGN command for the syntax diagram and command parameters.

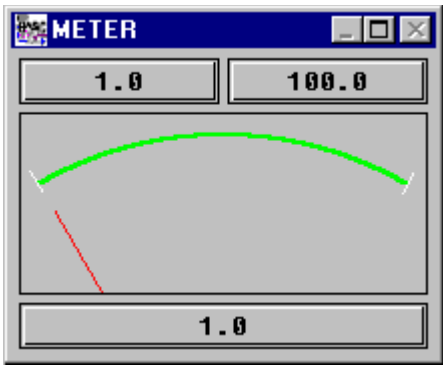
Creating a Default Widget

To create a widget with default attributes, use the ASSIGN command without setting any SET set attribute list parameters. For example:

```
ASSIGN @Meter TO WIDGET "METER"
```

creates a METER widget with default attributes, as shown in the following figure. to display this widget, enter the EDIT mode, type the following lines, and press RUN.

```
10  ASSIGN @Meter TO WIDGET "METER"
20  END
```



Preventing a Widget From Disappearing

When you ran the preceding program, you noted that the METER widget appeared, but then almost immediately disappeared. This is different from the way dialogs react. A dialog will wait indefinitely (unless given a TIMEOUT) for the user to respond.

In contrast, a widget appears and then almost immediately disappears when the program terminates. To keep the widget on the screen to view it, we must prevent the program from ending. For this example, we will insert an empty loop between lines 10 and 20. After a REN command, the program then looks like this:

```
10   ASSIGN @Meter TO WIDGET "METER"
20   LOOP
30   END LOOP
40   END
```

When this program is RUN, the widget will now remain on the screen for viewing. To stop this program from running, remove the focus from the widget and then press the **Stop** or **Pause** key or type *stop* at the command line.

HTBasic for Windows

CONFIG File Sections - Customizing Pens

Introduction

HTBasic for Windows pens that can be customized using the CONFIG file include:

- Settable pens are pens shared with BASIC programs that HTBasic for Windows is allowed to set.
- Logical pens define the area to be colored and the physical pen to use
- Physical pens provide the color for the specified logical pen

Settable Pens

Settable pens are pens shared with BASIC programs that HTBasic for Windows is allowed to set. The default is all pens. You should minimally allow the pens referenced by the Logical Pens Table. Those pens are listed first in each entry of the Settable Pens Table.
.
Widgets that require more colors than those defined in the Logical Pens Table create their own private palette that does not directly conflict with BASIC programs. However, when such a widget receives the input focus, it sets the system palette to match its private palette. All other applications must map into that palette on a closest match basis.

Logical Pens

Logical pens define the area to be colored and which physical pen to use. The actual color of the physical pen is specified in the Physical Pens Table. See Logical Pens Table for a list of logical pen numbers and associated physical pen numbers.

For example, from the Logical Pens Table, Logical Pen number 33 represents the color for Scrollbars. For a 64-color pen system, the associate physical pen number is 11. From the Physical Pens Table, the actual pen 11 color is interactive gray.

*Of the logical pens listed in the Logical Pens table, only those pens with **bold blue** entries are referenced in HTBasic Plus. The other pens remain for compatibility with previous versions of HP BASIC Plus.*

Physical Pens

Physical pens provide the color for the specified logical pen. The Logical Pens Table. specifies which physical pen to use for a specified logical pen. The actual color of the

physical pen is specified in the Physical Pens Table.

The Physical Pens table specifies the color map value that is set for each physical pen. If an entry for a required pen (a pen referenced from a logical pen) is not currently provided in the table (e.g., pens 32-47), insert as required.

The actual display's color map is only altered for the pens that are referenced by a logical pen from the Logical Pens table and are settable as specified above for the current graphics system, either 16-, 64-, or 256-pen.

DEFAULT BUTTON

Dialogs Only. The pushbutton that has the input focus is activated when the operator presses the **Spacebar** or the **Return** key or clicks on the button. If no pushbutton has the input focus when the user presses **Return**, the DEFAULT BUTTON is activated.

This attribute specifies which of the DIALOG BUTTONS will act as the default button in the dialog. That is, which button will be activated when the user presses **Return** while the input focus is on this dialog, but not on another of this dialog's buttons. Set the value of DEFAULT BUTTON to -1 if you do not want any action to occur when the user presses **Return** while no button has the focus.

HTBasic for Windows

DIALOG Command

DIALOG

Generates a dialog of the specified type

NOTE

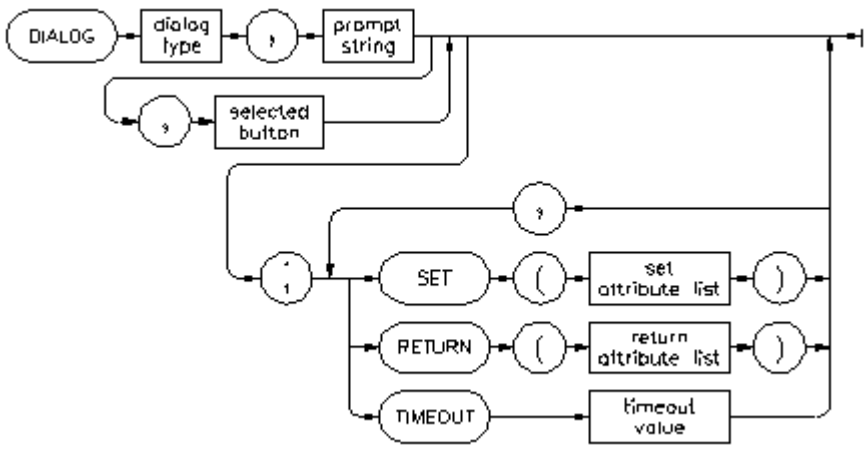
*Click the **>>** bar for additional information on the DIALOG command.*

Legal Usage	Keyboard Executable	Yes
	Programmable	Yes
	In an IF...THEN...	Yes

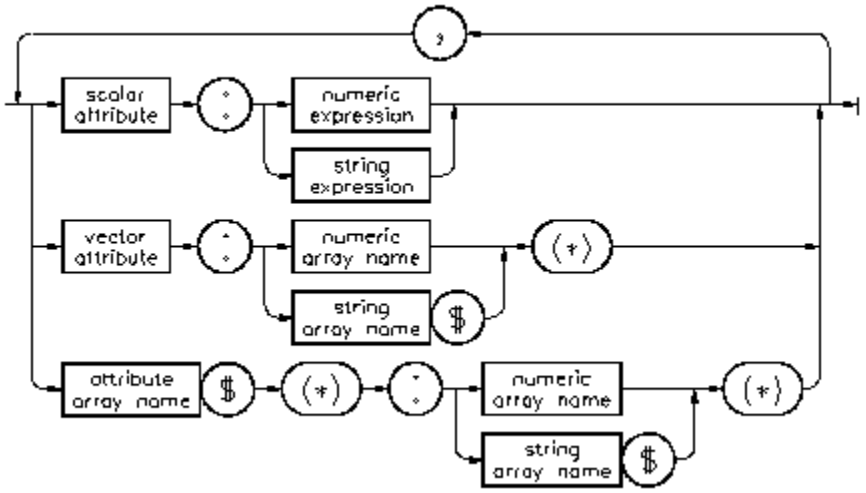
Example Statements

```
DIALOG "WARNING","Reactor Meltdown Imminent",
Btn;SET ("BACKGROUND":2)
DIALOG "STRING","Enter your Operator ID:";
RETURN ("VALUE":Resp$),TIMEOUT 10

DIM Speeds$(0:2)[20]
!
Speeds$(0)="Fast"
Speeds$(1)="Slow"
!
DIALOG "LIST","Pick your speed:",Btn;
SET ("WIDTH":250,"HEIGHT":80, "ITEMS":Speeds$(*)),
RETURN (SELECTION":Resp)
```

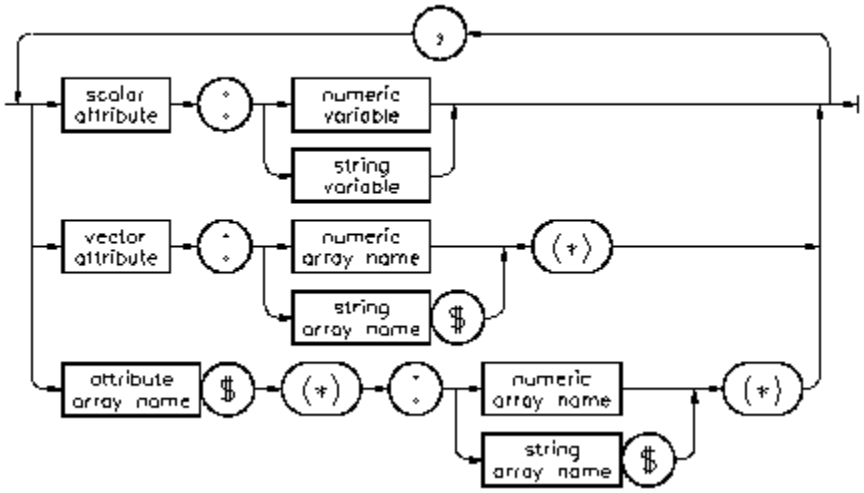
Syntax



literal form of the set attribute list:



literal form of the return attribute list



DIALOG BUTTONS

Dialogs Only. The values in the string array are the button labels that will be displayed on the dialog. The first element of each array is "0", regardless of what you declare in your DIM statement.

For example, if you have DIM Button$ (10:29)[20] and specify a default button of 10, Button$(10) (actually the 11th element of the array) will be the default. To prevent confusion, you should use OPTION BASE 0 for all arrays used with HTBasic for Windows.

DIALOG Command (continued)

Syntax (continued)

Item	Description	Range
attribute array name	an array of single-valued widget attributes. Both the attribute array and either the numeric array or string array must have identical dimensions	depends on widget
dialog type	string expression containing the type of dialog to be created	see Description
numeric array name	array of values to be assigned to this attribute	depends on attribute
numeric expression	expression containing the value to be assigned to the attribute	depends on attribute
prompt string	text to be displayed at the top of the dialog	any valid string
scalar attribute	a single-valued widget attribute (string expression)	depends on widget
selected button	numeric variable to receive index of button pressed by user	any numeric variable valid in this context
string array name	array of values to be assigned to this attribute	depends on attribute
string expression	expression containing the value to be assigned to the attribute	depends on attribute
timeout value	time (in seconds) to wait for user response	numeric expression
vector attribute	a multi-valued widget attribute (string expression)	depends on widget

HTBasic for Windows

DIALOG Command (continued)

Description

The DIALOG statement is a shortcut method for requesting input from the operator. The DIALOG statement functions in a similar fashion to the INPUT and LINPUT statements by collecting operator input without using more complex statements.

Using the DIALOG statement, you can perform the functional equivalent of the following (lengthier) process that would otherwise require other statements.

- You create a PANEL widget that contains a prompt string, a single widget, and some button widgets.
- You interact with the contained widget.
- Then, when you "press" one of the buttons, the system destroys all of the widgets that make up the "dialog PANEL" after passing the selected values from each of the widgets into the variables you have specified.

Types of Dialogs

Dialog	Description
COMBO	Used to enter a string from either an existing list or a string entry field
ERROR	Use to display an error message and halt program execution until the operator acknowledges the error
FILE	Prompts the operator to select a file name from a list of file names
INFORMATION	Used to display a piece of information and halt program execution until the operator acknowledges the information
KEYPAD	Used to enter numbers from a keypad
LIST	Prompts the operator to select an item or set of

NUMBER	items from a list of items
QUESTION	Prompts the operator to enter a number into a panel
STRING	Used to prompt the operator with a question and halt program execution until the operator answers the question
WARNING	Used to prompt the operator for information that the programmer needs to use within the BASIC program
	Used to display a warning message and halt program execution until the operator acknowledges the warning

DIALOG Command (continued)

DIALOG/DEFAULT BUTTONS

Use the DIALOG BUTTONS attribute to create the buttons in the dialog. These buttons appear in a single row at the bottom of the dialog, in the same order (left to right on the screen) in which they appear in the attribute array.

To specify one of these buttons as the default button, use the DEFAULT BUTTON attribute. Both DIALOG BUTTONS and DEFAULT BUTTON have different default values, depending on the type of dialog created.

selected button Option

If you specify the optional variable for selected button, when the DIALOG statement completes, the variable will contain:

- an index into the DIALOG BUTTONS array that identifies which button the user pushed to terminate the dialog, or
- a -1, indicating a timeout has occurred

The DIALOG BUTTONS array is always treated as OPTION BASE 0, regardless of how it was dimensioned.

SET Option

The SET option is used to specify the initial values for the attributes and to specify the initial values to be displayed by the contained widget (for example, the contents of the STRING widget in the STRING dialog).

RETURN Option

The RETURN option is used to specify the variables that will receive the final values of the dialog attributes just before the dialog is destroyed. These variables are used primarily to communicate the state of the contained widget back to the program when the user terminates the dialog. For example, to find out what the user typed into a STRING dialog, you should RETURN the VALUE attribute to a string variable in your program.

Specifying the same variable for the same attribute in both the SET

and RETURN attribute lists is acceptable, and in fact will be a common practice when the dialog's purpose is to allow the user to modify an existing quantity. For this purpose, you should supply the existing value in the SET attribute list and use the same variable in the RETURN attribute list so that the user's modification will change the program variable.

TIMEOUT Option

If you specify the TIMEOUT option in the DIALOG statement, the program will wait only the specified number of seconds for user input before continuing. If the user does not push a button in the dialog within the allotted time, the DIALOG statement will:

- Copy the current state of the dialog attributes into the variables specified in return attribute list
- Destroy the dialog
- Return a selected button value of -1, if the optional variable that will receive this value has been specified.

HTBasic for Windows

DISABLE EVENT Command

DISABLE EVENT

Prevents HTBasic for Windows from branching upon receipt of a specified event

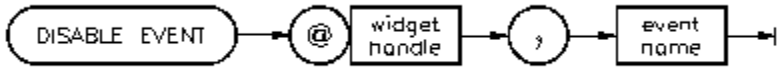
---

Legal Usage	Keyboard Executable	Yes
	Programmable	Yes
	In an IF...THEN...	Yes

Example Statements

```
DISABLE EVENT @Myscrollbar,"CHANGED"
DISABLE EVENT @String1, "KEYSTROKE"
```

Syntax



Item	Description	Range
event name	name of an event this widget can disable (string expression)	depends on the widget
widget handle	name identifying a widget	any valid name

Description

DISABLE EVENT disables HTBasic for Windows from branching upon receipt of a specified event. You can use DISABLE EVENT to temporarily suspend the effects an event would have on your program. For example, you may want the program to accomplish some task without interruption. After that task is accomplished, you can use ENABLE EVENT to reenable the program's sensitivity to that event.

A disabled event can still be logged. Then, when the event is reenabled, the branch will be taken. Only one occurrence of the event will be logged. The events can all be disabled and still accept inputs.

There are three important differences between DISABLE EVENT and OFF EVENT:

- DISABLE EVENT temporarily disables the event, while OFF EVENT permanently deactivates the event

- Only one occurrence of the event will be logged if the event is disabled with DISABLE EVENT. Therefore, the specified branch will be taken once the event is reenabled with ENABLE EVENT.

- The event will NOT be logged and the branch will never be taken if the event is deactivated with OFF EVENT.

In the following example, the first statement enables the event ALARM for the METER widget. Then, the second statement disables the event.

```
ENABLE EVENT @Meter, ALARM .
                .
                .
DISABLE EVENT @Meter, ALARM
```

Destroying Widgets

One method of terminating a widget is to stop the program. However, the usual method to destroy a widget is to use the ASSIGN command, as follows:.

```
ASSIGN @widget handle TO *
```

where widget handle is the name that was given to this widget in the ASSIGN command that created the widget

For example, to destroy the widget @Meter, insert the following line in your program at a point after the widget has performed its function:

```
ASSIGN @Meter to *
```

Widgets are destroyed automatically when the:

- Context in which they are created terminates
- Subprogram in which they are created terminates
- Function in which they are created terminates
- Program in which they are created stops, because the user presses **Shift-Stop** or **Reset**, or through normal program termination

HTBasic for Windows

Dialog Attribute Default Values

Although common attributes have the same type and allowed values for all widgets (or for level-0 widgets or dialogs), they may have *different* default values for different widgets (or for level-0 widgets or dialogs).

For example, the WIDTH attribute for Type (numeric) and Allowed Values (any integer number) are the same for both METER and BARS. However, the METER widget's WIDTH attribute has a default value of 220 pixels, while the BARS widget's WIDTH attribute has a default value of 250 pixels.

HTBasic for Windows

Dialog Attribute Definitions

Dialog *attributes* are qualities or properties of HTBasic for Windows dialogs (and widgets) The function and appearance of a dialog (or widget) is determined by the values of its attributes. There are five types of dialog attributes, as shown in Dialogs Attribute Types. See Dialogs Common Attributes for a list of common attributes for dialogs.

HTBasic for Windows

Dialogs Attributes

This topic provides general information about dialog attributes, including:

- Dialog Attribute Definitions
- Setting/Checking Dialog Attribute Values
- Dialog Attribute Default Values
- Specifying FONT Attribute Values
- Dialog Common Attributes Table

See List of Dialogs for descriptions/listings of dialogs.
See Dialogs Format for the format of dialogs descriptions.

HTBasic for Windows

Dialogs Common Attributes

This topic lists the common attributes for HTBasic Plus dialogs. Click the attribute name for a description of the attribute.

NOTE

Since these common attributes may have different defaults for each dialog, this information may be changed for a specific dialog. See the appropriate dialog description for details.

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
DEFAULT BUTTON	Numeric	Any valid index into the DIALOG BUTTONS array	0
DIALOG BUTTONS	String Array	Any valid string array	Varies
HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
JUSTIFICATION	String	"LEFT","CENTER"	"CENTER"
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE	Numeric	0,1	0 (not minimizable, button

			does not appear in title bar)
MOVABLE	Numeric	0,1	1 (movable, click on title bar and drag)
RESIZABLE	Numeric	0,1	1 (resizable, special border appears around the dialog)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SYSTEM MENU	String or String array	Any string or string array with 1-64 elements	Null
SYSTEM MENU COUNT	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT	Numeric	0 to number of items in system menu - 1	0
TITLE	String	Any valid string	Dialog name
USER DATA	String	Any valid string	Null string
VERSION	String	Any valid string	Current version
WIDTH	Numeric	Any integer number (of pixels)	Varies
X	Numeric	Any integer, number of pixels from 0,0 [2]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [2]	Autoplacement

[1] Read the CONFIG file or query for the default with a STATUS command

[2] Screen or parent work area upper-left corner

HTBasic for Windows

Dialogs Format

Overview

This topic shows the format for HTBasic Plus dialogs descriptions.

- See Dialogs Attributes for information on HTBasic Plus dialogs attributes.
- See List of Dialogs for a listing of HTBasic Plus dialogs.

NOTE

Each dialog description has the general structure shown. However, not all dialogs include every category. See the appropriate dialog description for details on each dialog.

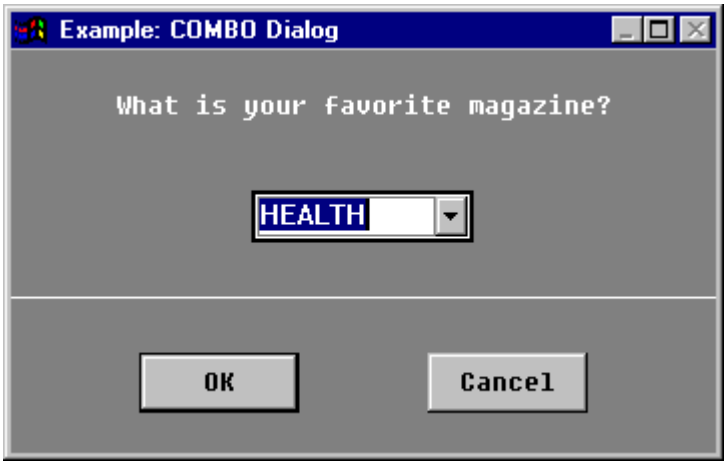
Typical Dialog Format

COMBO Dialog

Prompts the user to select from a list of entries or to enter a selection

---

Example Image



Example Program

This example program produces a COMBO dialog similar to the display shown. above. See the Dialogs program for another example using the COMBO dialog.

```
1       ! Example: COMBO Dialog
2       !
```

```
10        DIM M$(1:7)[20],S$[25],P$[50]
20        INTEGER Btn
30        P$="What is your favorite magazine?"
40        DATA "COSMOPOLITAN","ENQUIRER","DISCOVER","TIME"
50        DATA "HEALTH","SPORTS","NEW YORKER"
60        READ M$(*)
70        !
80        DIALOG "COMBO",P$,Btn;SET ("ITEMS":M$(*)),RETURN ("TEXT":S$)
90        DISP "Magazine selected:";S$
100     END
```

Attributes

See COMBO Dialog Attributes for the COMBO dialog attribute list.

Remarks

None.

HTBasic for Windows

Dialogs/Widgets Attribute Types

Overview

This topic describes the five attribute types for HTBasic Plus dialogs and widgets.

NOTE

OPTION BASE 0 is used for all array indices. Dialogs and widgets always treat the first element in an array as element 0, regardless of how you dimension (DIM) or ALLOCATE.

Dialogs/Widgets Attribute Types

Attribute Type	Description
Complex	Equivalent to HTBasic for Windows COMPLEX data type - has a real and imaginary component
Numeric	Allowed values are INTEGER, REAL, or numeric expression [1]
Numeric Array	Allowed values are INTEGER or REAL arrays
String	Allowed values are string or string expression [2]
String Array	Allowed values are array of string expressions

[1] All branches in the numeric expressions syntax diagram are allowed in CONTROL statements, but only the branch that begins with "numeric variable name" is allowed in STATUS statements.

[2] All branches in the string expressions syntax diagram are allowed in CONTROL statements, but only the branch that begins with "string variable name" is allowed in STATUS statements.

HTBasic for Windows

Dialogs/Widgets Overview

Dialogs Overview

A fundamental entity in HTBasic Plus is the dialog. A dialog acts, from a program's point of view, as an INPUT statement with a TIMEOUT. From the user's point of view, a dialog appears as a popup panel that disappears when the user supplies a response. See List of Dialogs for a description of the HTBasic for Windows dialogs.

Widgets Overview

The other fundamental HTBasic Plus entity is the widget.
Widgets are very flexible items that can be used to build sophisticated user interfaces. For example, you can create "virtual instruments" that include pushbuttons, sliders, and text, meters and graphics displays, and pulldown menus that can be controlled by a mouse.

Widgets can be loosely grouped into five categories, but some widgets may belong to more than one category. See List of Widgets for an alphabetical listing of each widget. See Functional Widget List for a list of widgets by functional category.

Dialogs/Widgets Attributes

Dialogs and widgets have common and specific attribute settings that allow you to customize the dialog or widget. Common attributes includes items such as the position of the object on the screen, the background color of the object, and the size of the object. Most dialogs and widgets also have attributes specific to that object.

For example, the STRIPCHART widget has over thirty specific attributes, including number of channels displayed, color assigned to each channel, and control over the number of points displayed, number of points in the display buffer, and how the display is updated.

HTBasic for Windows

EABLE EVENT Command

ENABLE EVENT

Enables HTBasic for Windows to branch upon receipt of a specified event

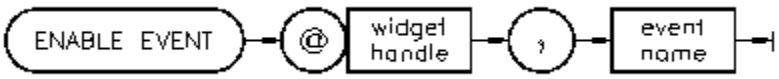
---

Legal Usage	Keyboard Executable	Yes
	Programmable	Yes
	In an IF...THEN...	Yes

Example Statements

```
ENABLE EVENT @Myscrollbar, "CHANGED"
ENABLE EVENT @String1, "KEYSTROKE"
```

Syntax



Item	Description	Range
event name	name of an event this widget can enable (string expression)	depends on the widget
widget handle	name identifying a widget	any valid name

Description

ENABLE EVENT enables HTBasic for Windows to branch upon receipt of a specified event. The branch to be taken and the defined event are specified by the ON EVENT command. The ON EVENT command automatically creates an ENABLE EVENT for the specified widget and event.

For example, the following line enables program branching for the event CHANGED. The actual branch taken is defined by an ON EVENT command.

```
ENABLE EVENT @Myscrollbar, "CHANGED"
```

If an ON EVENT statement has been defined for a widget and an event, when the event occurs an event-initiated branch results. Use DISABLE EVENT to temporarily suspend the effects an event would have on your program.

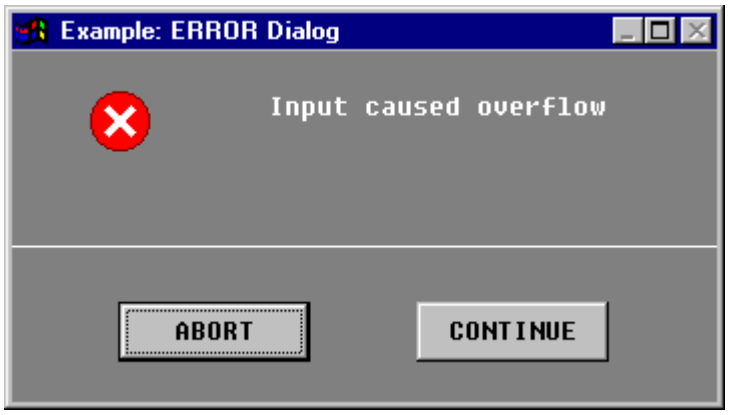
For example, you may want the program to accomplish some task without interruption. After that task is accomplished, you can use ENABLE EVENT to reenable the program's sensitivity to that event.

While the event is disabled, it can still be logged. Then, when it is reenabled, the branch will be taken. Only one occurrence of the event will be logged. You must have at least one currently defined event branch in your program to accept inputs from the mouse or keyboard. The events can all be disabled and still accept inputs.

ERROR Dialog

Displays an error message and suspends program execution
until the operator acknowledges the error

Example Image



Example Program

See ERROR Dialog for an example program that produces a display similiar to that shown above.

NOTE

See the following program for another example using the ERROR dialog.

Dialogs Tests

Attributes

See ERROR Dialog Attributes for the ERROR dialog attribute list.

Remarks

None.

HTBasic for Windows

ERROR Dialog Attributes

ERROR Dialog Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
DEFAULT BUTTON	Numeric	Any valid index into the DIALOG BUTTONS array	0
DIALOG BUTTONS	String array	Any valid string array	A single-element array containing OK.
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE	Numeric	0,1	1
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE	Numeric	0,1	1 (resizable, special border appears

			around the dialog)
RESTORE SCREEN	Numeric	0,1	0 (do not save the screen)
TITLE	String	Any valid string	ERROR
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Screen or parent work area upper-left corner

HTBasic for Windows

Effects on Graphics

Introduction

NOTE

Since HTBasic Plus overlays HTBasic for Windows graphics, we highly recommend running with graphics buffering on ("-graphics buffering on" HTBasic for Windows command line switch).

Without graphics buffering, whenever a widget or dialog is moved, shrunk, or destroyed, all underlying HTBasic for Windows graphics will be erased (unless that widget or dialog has its RESTORE SCREEN attribute set).

COLOR MAP Mode

While HTBasic Plus is loaded, HTBasic for Windows graphics will be placed in the COLOR MAP mode (i.e., PLOTTER IS CRT, "INTERNAL";COLOR MAP) if your computer's display system supports it. This implies that all PLOTTER IS CRT,"INTERNAL" requests will be converted.

After each of the following events, HTBasic Plus will reinitialize a portion of the HTBasic for Windows COLOR MAP to the values specified in your CONFIG file.:

- LOAD BIN "BPLUS"
- MERGE ALPHA WITH GRAPHICS
- PLOTTER IS CRT
- GINIT
- RESET
- SCRATCH A

The portion converted depends on two things:

1. How many colors are available to HTBasic for Windows as specified with the "*-colors*" HTBasic for Windows command line switch.

2. Which HTBasic for Windows pens are referenced in the logical pen portion of your CONFIG file.

Color Mapping in HTBasic for Windows

By default, HTBasic Plus uses 16 colors. If less than 256 colors
are available to HTBasic for Windows, these 16 colors are mapped to pens 0-15 and
pens 0-7 are set the same as the HTBasic for Windows default. If 256 colors are available,
the 16 default HTBasic for Windows colors are mapped to pens 16-31.

If you subsequently execute any SET PEN command, that definition remain in effect until
any of the following events occur:

- MERGE ALPHA WITH GRAPHICS
- PLOTTER IS CRT
- GINIT
- RESET
- SCRATCH A

HTBasic for Windows

HTBasic Plus Error Messages

Nbr	Title	Description
2	Memory overflow	There is not enough free memory for the requested operation. The *-w* command switch may solve the problem.
28	LOG or LGT of non-positive number	The argument to the LOG and LGT functions cannot be negative or zero.
56	File or Path not found	No file or directory exists with this name. You may not have included the proper device or path specifiers. Use CREATE or CREATE DIR if you want to create a new file or directory with this name.
58	Improper file type	The file type is incorrect for the requested operation or an attempt was made to LOAD an old revision PROG file, or a widget create attempt tried to load a non-widget file
59	End of file or buffer	The end-of-file or end-of-buffer was unexpectedly reached during this operation.
170	I/O operation not allowed	An attempt was made to do an illegal operation. Some problems to consider are: - Attempt to use a widget's I/O path in a non-widget I/O command or vice-versa - The device may not support the operation - A primary address is specified and should not be - The operation requires the controller to be or not be the

		active/system controller - USING is not allowed with a LIF ASCII file For more information, check the documentation for the device driver being accessed.
177	Undefined I/O path name	The I/O path name has not been ASSIGNed to a device, file, or buffer.
347	Structures improperly matched	- The attributes array and values array in a SET or RETURN are not the same size - The FOR...NEXT, LOOP...END LOOP, REPEAT...UNTIL, SELECT...END SELECT, or WHILE...END WHILE program structures are nested improperly or there is a missing structured statement.
553	Cannot load object file	The widget could not be loaded. Check the file permissions and the directory where the file should be located. Check in all the possible locations which HTBasic for Windows will try to load the file. Also, the file may be corrupt.
554	Object file not a widget	The file did not have a header which could be recognized as a widget.
557	Undefined widget	The widget specified does not have internal (within HTBasic for Windows) or external (WI prefixed file) code which the binary could locate.
558	Undefined widget attribute	The widget attribute specified in conjunction with SET or RETURN is not valid. See the list of valid attributes for the widget in question.

559	Wrong parameter type for attribute	The parameter being passed to an attribute is of the wrong type. See the list of valid attributes for the widget in question.
560	Menu not allowed in child widget	Menus are not allowed in a child widget. Menus can only be children of a level-0 panel, or of another menu.
561	Widget must have a parent	Widget cannot be created without a parent.
562	Parent widget does not support this type of child	The parent widget does not allow this type of widget to be a child. See if the widget can be used as a level-0 widget or as a child of another widget.
563	SET not allowed for attribute	SET not allowed for attribute.
564	RETURN not allowed for attribute	RETURN not allowed for attribute.
565	VALUE out of range for attribute	Value out of range for attribute. Check the list of possible values for the attribute on the widget.
566	Invalid value for attribute	Invalid value for attribute. The value may be in range, but this particular value is not allowed.
567	Too few elements in array for attribute	Too few elements in array for attribute. Make the array size larger.
569	Invalid font specification	Invalid font specification. See the FONT attribute for the widget being used. Typical font specs look like "10 BY 20, BOLD", etc.
570	Undefined dialog	A DIALOG type must be one of several predefined dialog

	type	box types. Typical types are "INFORMATION", "WARNING", and "STRING".
571	Widget has no events to set	The widget has no events to set. Some widgets have no events associated with them. The widget causing this error is such a widget.
572	Undefined widget event	The event specified is not one of the valid events for that particular widget. See the list of events for the widget in question.
573	Attribute not available to child widget	Attribute not available to child widget. See if the widget can be made a level-0 widget or if the attribute can be deleted.
574	Attribute not available to level-0 widget	Attribute not available to level-0 widget. See if the widget can be made a child of another widget or if the attribute can be deleted.

HTBasic for Windows

Event-Initiated Branching

Introduction

Most widgets have one or more associated *events*. An event results from an action by the user or from an action generated from within the program. You can use the DISABLE EVENT, ENABLE EVENT, OFF EVENT, ON EVENT, and WAIT FOR EVENT commands to generate event-initiated branches for your program. This is called "event-initiated branching".

For example, the SLIDER widget has three defined events: CHANGED, DONE, and SYSTEM MENU:

- The CHANGED event is generated every time the user changes the SLIDER VALUE by clicking on the arrows or in the trough or by clicking and dragging the slider
- The DONE event is generated when the user releases the mouse button
- The SYSTEM MENU event is generated when the user selects items from the SYSTEM MENU.

Event-Initiated Branching Commands

Event-initiated branching is a programming technique that uses defined event actions as interrupts to redirect program flow. The following table summarizes the event-initiated branching commands. Click the command name for a description of the command.

Command	Description
DISABLE EVENT	Disables program branching upon receipt of specified event
ENABLE EVENT	Enables program branching upon receipt of specified event
OFF EVENT	Cancels event branches defined by ON EVENT
ON EVENT	Defines event branch taken after a widget generates that event
WAIT FOR EVENT	Suspends program execution until an event occurs

```
10      ! ************************************************************************
20      !  Example: Alarm Clock
30      !
40      !  This program builds a digital alarm clock. You can set an alarm
50      !  time using a SCROLLBAR. A TOGGLEBUTTON allows you to disable
60      !  or enable alarm operation. A PUSHBUTTON allows you to turn off
70      !  the alarm.
80      !
90      ! ************************************************************************
100     !
110     CLEAR SCREEN
120     OPTION BASE 1
130     !
140     INTEGER Alarmval,Hour,Minute,N,Noisy
150     Noisy=0
160     !
170     DIM A$[8],T$[8]
180     !
190     INTEGER Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
200     !
210     Black=0
220     White=1
230     Red=2
240     Yellow=3
250     Green=4
260     Cyan=5
270     Blue=6
280     Magenta=7
290     !
300     INTEGER A(6),Nlines
310     REAL Dw,Dh,Vh,Pw,Ph,Px,Py
320     !
330     GESCAPE CRT,3;A(*)          ! Get display bounds
340     Dw=A(3)-A(1)+1              ! Compute display width in pixels
350     Dh=A(4)-A(2)+1              ! Compute display height in pixels
360     STATUS CRT,13;Nlines        ! Get number of text lines on display
370     Vh=Dh*(1-6/Nlines)          ! Height above DISP line
380     !
390     Pw=320                      ! PANEL width is half display width
400     Ph=240                      ! Same for PANEL height
410     Px=(Dw-Pw)/2                ! Center panel horizontally
```

```
420   Py=(Vh-Ph)/2                    ! Center panel above DISP line
430   !
440   ! Build the main PANEL
450   !
460   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
470   CONTROL @Main;SET ("X":Px,"Y":Py,"WIDTH":1.2*Pw,"HEIGHT":Ph)
480   CONTROL @Main;SET ("TITLE":" Example: Alarm Clock")
490   CONTROL @Main;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
500   CONTROL @Main;SET ("BACKGROUND":Blue)
510   CONTROL @Main;SET ("SYSTEM MENU":"Quit")
520   ON EVENT @Main,"SYSTEM MENU" GOTO Finis
530   !
540   ! Dimensions and coordinates for child widgets
550   !
560   REAL Ih,Iw                          ! Inside height and width
570   REAL Gw,Gh,Lw,Bw,Lh,Sw,Sh,Bgap      ! Child widget dimensions
580   REAL Y1,Y2,Y3,Y4,X1,X2                    ! Child widget coordinates
590   !
600   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
610   !
620   Gh=Ih*.05                       ! Arbitrary height of vertical gap
630   Gw=Iw*.02                       ! Arbitrary width of horizontal gap
640   Wh=Ih*.3                ! Arbitrary height of LABEL & BUTTON widgets
650   Lw=Iw*.6                ! Arbitrary width of LABEL widgets
660   !
670   X1=Gw                   ! Right side of LABELs & SCROLLBAR
680   X2=X1+Lw+Gw             ! Right side of BUTTONs
690   Bw=Iw-(X2+Gw)           ! BUTTON width
700   Sw=Iw-2*Gw              ! SCROLLBAR width
710   !
720   Y1=Gh                   ! Position of first row of widgets
730   Y2=Y1+Wh+Gh             ! Position of second row of widgets
740   Y3=Y2+Wh                ! Position of bottom of second row of widgets
750   Bgap=Ih-Y3              ! Height of bottom gap on PANEL
760   !
770   ! Create LABEL for time value
780   !
790   ASSIGN @Clock TO WIDGET "LABEL";PARENT @Main
800   CONTROL @Clock;SET ("X":X1,"Y":Y1,"WIDTH":Lw,"HEIGHT":Wh)
810   CONTROL @Clock;SET ("FONT":"18 BY 30,BOLD")
820   CONTROL @Clock;SET ("BACKGROUND":Black,"PEN":Red,"BORDER":0)
830   !
```

```
840   ! Create TOGGLEBUTTON to enable/disable alarm
850   !
860   ASSIGN @Toggle TO WIDGET "TOGGLEBUTTON";PARENT @Main
870   CONTROL @Toggle;SET ("X":X2,"Y":Y1,"WIDTH":Bw,"HEIGHT":Wh)
880   CONTROL @Toggle;SET ("FONT":"18 BY 16,BOLD","LABEL":" ENABLE ALARM")
890   !
900   ! Create LABEL for alarm value
910   !
920   ASSIGN @Alarm TO WIDGET "LABEL";PARENT @Main
930   CONTROL @Alarm;SET ("X":X1,"Y":Y2,"WIDTH":Lw,"HEIGHT":Wh)
940   !
950   ! Create PUSHBUTTON to stop alarm
960   !
970   ASSIGN @Stop TO WIDGET "PUSHBUTTON";PARENT @Main
980   CONTROL @Stop;SET ("X":X2,"Y":Y2,"WIDTH":Bw,"HEIGHT":Wh)
990   CONTROL @Stop;SET ("FONT":"18 BY 16,BOLD","LABEL":" STOP ALARM")
1000  CONTROL @Stop;SET ("LABEL":"STOP ALARM")
1010  !
1020  ! Create SCROLLBAR, then center
1030  !
1040  ASSIGN @Scroll TO WIDGET "SCROLLBAR";PARENT @Main
1050  CONTROL @Scroll;SET ("ORIENTATION":"HORIZONTAL")
1060  STATUS @Scroll;RETURN ("HEIGHT":Sh)
1070  Y4=Y3+(Bgap-Sh)/2
1080  CONTROL @Scroll;SET ("X":X1,"Y":Y4,"WIDTH":Sw)
1090  !
1100  ! Set scale on SCROLLBAR to run through minutes of day.
1110  ! Allow slider to be incremented by hour and minute values.
1120  !
1130  CONTROL @Scroll;SET ("MINIMUM":0,"MAXIMUM":1439)
1140  CONTROL @Scroll;SET ("MINOR INCREMENT":1,"MAJOR INCREMENT":60)
1150  !
1160  ! Set initial time and alarm
1170  !
1180  GOSUB Update
1190  GOSUB Newalarm
1200  !
1210  ! Make application visible
1220  !
1230  CLEAR SCREEN
1240  CONTROL @Main;SET ("VISIBLE":1)
1250  !
```

```
1260   ! Enable widget event
1270   !
1280   ON EVENT @Scroll,"CHANGED" GOSUB Newalarm
1290   ON CYCLE 1 GOSUB Update
1300   !
1310   ! Loop until the user presses a key, then exit
1320   !
1330   LOOP
1340     WAIT FOR EVENT
1350   END LOOP
1360   !
1370   ! ***************** End of Main Program **************************
1380   !
1390   ! This routine updates the clock time. If the alarm is on,
1400   ! (Noisy=1), it will beep.
1410   !
1420   ! Once it updates the time, it checks the time for a match against the
1430   ! alarm. If there is a match, it sets Noisy to cause an alarm, causes
1440   ! a beep, enables an event on the PUSHBUTTON to turn off the alarm,
1450   ! and turns the PUSHBUTTON red.
1460   !
1470 Update:!
1480   IF Noisy=1 THEN BEEP 3000,.15
1490   T$=TIME$(TIMEDATE)
1500   CONTROL @Clock;SET ("VALUE":T$)
1510   !
1520   STATUS @Toggle;RETURN ("VALUE":N)          ! Is alarm enabled?
1530   IF N=1 THEN                                ! Yes --
1540     IF A$=T$ THEN                                  ! Compare time
1550       Noisy=1
1560       BEEP 3000,.15
1570       ON EVENT @Stop,"ACTIVATED" GOSUB Killnoise
1580       CONTROL @Stop;SET ("BACKGROUND":Red)
1590     END IF
1600   END IF
1610   RETURN
1620   !
1630   ! This routine turns off the alarm
1640   !
1650 Killnoise:!
1660   OFF EVENT @Stop,"ACTIVATED"
1670   CONTROL @Stop;SET ("BACKGROUND":Blue)
```

```
1680   Noisy=0
1690   RETURN
1700   !
1710   ! This routine updates the alarm value. The value provided by
1720   ! the SCROLLBAR is converted into a time strng that matches
1730   ! that provided by TIME$(TIMEDATE).
1740   !
1750 Newalarm:!
1760   STATUS @Scroll;RETURN ("VALUE":Alarmval)
1770   !
1780   Hour=Alarmval DIV 60
1790   SELECT Hour
1800   CASE 0
1810     A$="12"
1820   CASE 1 TO 9
1830     A$="0"&VAL$(Hour)
1840   CASE ELSE
1850     A$=VAL$(Hour)
1860   END SELECT
1870   A$=A$&":"
1880   !
1890   Minute=Alarmval MOD 60
1900   SELECT Minute
1910   CASE 0 TO 9
1920     A$=A$&"0"&VAL$(Minute)
1930   CASE ELSE
1940     A$=A$&VAL$(Minute)
1950   END SELECT
1960   A$=A$&":00"
1970   !
1980   CONTROL @Alarm;SET ("VALUE":A$)
1990   RETURN
2000   !
2010 Finis:!
2020   ASSIGN @Main TO *                ! Delete PANEL widget
2030   CLEAR SCREEN
2040   END
```

```
10      ! ********************************************************
20      ! Example: BAR Widget
30      !
40      ! This program creates a BAR widget with three ranges.
50      ! As the bar height progesses from the LOW range through
60      ! the middle range to the HIGH range, the bar color changes
70      ! from green to yellow to red. When the bar is in the LOW
80      ! or the HIGH range, the program beeps throughout the range.
90      !
100     ! ********************************************************
110     !
120     INTEGER N,M
130     ASSIGN @Bar TO WIDGET "BAR"
140     CONTROL @Bar;SET ("TITLE":" Example: BAR Widget")
150     CONTROL @Bar;SET ("X":100,"Y":20,"WIDTH":225,"HEIGHT":200)
160     CONTROL @Bar;SET ("ALARM RANGES":"LOW,HIGH","ALARM TYPE":"BEEP")
170     CONTROL @Bar;SET ("LOW LIMIT":20,"HIGH LIMIT":80)
180     CONTROL @Bar;SET ("SYSTEM MENU":"Quit")
190     ON EVENT @Bar,"SYSTEM MENU" GOTO Finis
200     !
210     FOR M=1 TO 3
220       FOR N=1 TO 100
230         WAIT .05
240         CONTROL @Bar;SET ("VALUE":N)
250       NEXT N
260       WAIT 1
270     NEXT M
280     !
290 Finis:    !
300     ASSIGN @Bar TO *                  ! Delete BAR widget
310     END
```

```
10      ! ************************************************
20      ! Example: BARS Test
30      !
40      ! This program creates a BARS widget with three bars and
50      ! displays the square root (Root) and natural logarithm
60      ! (Ln) of numbers (Number) from 0 to 100.
70      !
80      ! ************************************************
90      !
100    INTEGER M,N
110    REAL Barval(0:2)
120    ASSIGN @Bars TO WIDGET "BARS"
130    CONTROL @Bars;SET ("TITLE":" Example: BARS Test","HEIGHT":300,"WIDTH":300)
140    CONTROL @Bars;SET ("X":50,"Y":25)
150    CONTROL @Bars;SET ("SYSTEM MENU":"Quit")
160    ON EVENT @Bars,"SYSTEM MENU" GOTO Finis
170    !
180    CONTROL @Bars;SET ("BAR COUNT":3)
190    CONTROL @Bars;SET ("CURRENT BAR":1)
200    CONTROL @Bars;SET ("BAR LABEL":"Number")
210    CONTROL @Bars;SET ("CURRENT BAR":2)
220    CONTROL @Bars;SET ("BAR LABEL":"Root")
230    CONTROL @Bars;SET ("CURRENT BAR":3)
240    CONTROL @Bars;SET ("BAR LABEL":"Ln")
250    !
260    FOR M=1 TO 5
270      FOR N=1 TO 100
280        Barval(0)=N
290        Barval(1)=SQR(N)
300        Barval(2)=LOG(N)
310        CONTROL @Bars;SET ("VALUES":Barval(*))
320        WAIT .05
330      NEXT N
340    NEXT M
350 Finis: !
360    ASSIGN @Bars TO *       ! Delete BARS widget
370    END
```

```
10      ! ************************************************
20      ! Example: BARS Widget
30      !
40      ! This program creates a BARS widget with one bar
50      ! and displays voltage values from 1 to 100 volts.
60      !
70      ! ************************************************
80      !
90      INTEGER M,N
100   ASSIGN @Bars TO WIDGET "BARS"
110   CONTROL @Bars;SET ("TITLE":" Example: BARS Widget")
120   CONTROL @Bars;SET ("X":50,"Y":25)
130   CONTROL @Bars;SET ("BAR LABEL":"Voltage")
140   CONTROL @Bars;SET ("SYSTEM MENU":"Quit")
150   ON EVENT @Bars,"SYSTEM MENU" GOTO Finis
160   !
170   FOR M=1 TO 5
180     FOR N=1 TO 100
190       CONTROL @Bars;SET ("VALUE":N)
200       WAIT .05
210     NEXT N
220   NEXT M
230 Finis: !
240   ASSIGN @Bars TO *                ! Delete BARS widget
250   END
```

```
10      ! ********************************************************
20      ! Example: BITMAP Widget
30      !
40      ! This program demonstrates the use of the BITMAP widget. It
50      ! allows you to bring in and display bitmaps, as well as select
60      ! portions of them and save them in a file.
70      !
80      ! It also demonstrates the SCROLLABLE attribute for the PANEL,
90      ! such as the various operations needed to make SCROLL WIDTH
100     ! and SCROLL HEIGHT as correct as possible.
110     !
120     ! ********************************************************
130     !
140     ! Variables used:
150     !
160     !    S$:            General-purpose string
170     !    N:             General-purpose variable
180     !    Btn:           Returns button value from dialogs
190     !    Sc:            Scroll factor
200     !    Bitfile$:              Name of bitmap file to be read
210     !    Dirname$:      Directory name returned from FILE dialog
220     !    B1$(*):                Stores MENU BUTTONS
230     !    A1$,A2$:       Stores dialog attributes
240     !
250     DIM S$[256]
260     INTEGER N,Btn,Sc
270     DIM Bitfile$[100],Dirname$[100],Btns$(1:3)[50],A1$[50],A2$[50]
280     !
290     DATA "BMP File","XWD File","Cancel","DIALOG BUTTONS","SELECTION",20
300     READ Btns$(*),A1$,A2$,Sc
310     !
320     ! Widget dimensions
330     !
340     INTEGER Pw,Ph,Px,Py,Iw,Ih,Ww,Wh,Wx,Wy
350     !
360     ! Variables for display scaling
370     !
380     INTEGER Cursor,D(1:4),Dw,Dh
390     !
400     ! Get display size
410     !
```

```
420   GESCAPE CRT,3;D(*)
430   Dw=D(3)-D(1)
440   Dh=D(4)-D(2)
450   !
460   CLEAR SCREEN
470   !
480   Pw=Dw*.7                        ! PANEL width
490   Ph=Dh*.7                        ! PANEL height
500   Px=(Dw-Pw)/2              ! Center PANEL
510   Py=(Dh-Ph)/2
520   !
530   ! Create PANEL for BITMAP widget. The SCROLL WIDTH and
540   ! HEIGHT are set to a small value so that scroll bars
550   ! will not appear initially. The actual heights are set
560   ! later to fit the bitmap that has been loaded.
570   !
580   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
590   CONTROL @Main;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
600   CONTROL @Main;SET ("TITLE":" Example: BITMAP Widget")
610   CONTROL @Main;SET ("SIZE CONTROL":"SCROLLABLE","MINIMIZABLE":1)
620   CONTROL @Main;SET ("SCROLL WIDTH":1,"SCROLL WIDTH UNITS":Sc)
630   CONTROL @Main;SET ("SCROLL HEIGHT":1,"SCROLL HEIGHT UNITS":Sc)
640   !
650   ! Build menu
660   !
670   ASSIGN @Menu TO WIDGET "PULLDOWN MENU";PARENT @Main
680   CONTROL @Menu;SET ("LABEL":"Menu")
690   ASSIGN @Getfile TO WIDGET "MENU BUTTON";PARENT @Menu
700   CONTROL @Getfile;SET ("LABEL":"Get Bitmap File")
710   ASSIGN @Savefile TO WIDGET "MENU BUTTON";PARENT @Menu
720   CONTROL @Savefile;SET ("LABEL":"Cut Bitmap To File")
730   ASSIGN @Cd TO WIDGET "MENU BUTTON";PARENT @Menu
740   CONTROL @Cd;SET ("LABEL":"Change Directory")
750   ASSIGN @S TO WIDGET "MENU SEPARATOR";PARENT @Menu
760   ASSIGN @Quit TO WIDGET "MENU BUTTON";PARENT @Menu
770   CONTROL @Quit;SET ("LABEL":"Quit")
780   !
790   ! Create and size BITMAP widget. (Setting RETAIN RASTER makes
800   ! the widget redraw quickly when overwritten by a dialog.)
810   !
820   ASSIGN @Bitmap TO WIDGET "BITMAP";PARENT @Main
830   CONTROL @Bitmap;SET ("RETAIN RASTER":1,"AUTO SIZE":1)
```

```
840   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
850   Wx=Iw*.01
860   Wy=Ih*.01
870   Wh=Ih*.98
880   Ww=Iw*.98
890   CONTROL @Bitmap;SET ("X":Wx,"Y":Wy,"WIDTH":Ww,"HEIGHT":Wh)
900   !
910   ! Set events
920   !
930   ON EVENT @Getfile,"ACTIVATED" GOSUB Getbitfile
940   ON EVENT @Savefile,"ACTIVATED" GOSUB Savebits
950   ON EVENT @Cd,"ACTIVATED" GOSUB Chdir
960   ON EVENT @Quit,"ACTIVATED" GOTO Finis
970   !
980   CONTROL @Main;SET ("VISIBLE":1)
990   !
1000  ! Loop and wait for input
1010  !
1020  LOOP
1030    WAIT FOR EVENT
1040  END LOOP
1050  STOP
1060  !
1070  ! ********** End of Main Program **********
1080  !
1090  ! This routine gets a bitmap file and displays it. It gets
1100  ! the bitmap size and sets up PANEL scrolling accordingly.
1110  ! Also, it makes the BITMAP widget invisible so the display
1120  ! will "thrash" as little as possible.
1130  !
1140 Getbitfile: !
1150  S$="Please enter the name of a bitmap (.XWD or .BMP) file:"
1160  DIALOG "FILE",S$,Btn;RETURN ("SELECTION":Bitfile$)
1170  !
1180  IF Btn=0 THEN
1190    CLEAR ERROR
1200    ON ERROR GOSUB Errtrap
1210    CONTROL @Bitmap;SET ("VISIBLE":0,"BITMAP FILE":Bitfile$)
1220    OFF ERROR
1230    IF ERRN<>0 THEN
1240      DIALOG "ERROR","Can't open file / invalid bitmap file."
1250      CONTROL @Bitmap;SET ("BITMAP FILE":"","VISIBLE":1)
```

```
1260     ELSE
1270       CONTROL @Bitmap;SET ("VISIBLE":0)
1280       STATUS @Bitmap;RETURN ("BITMAP HEIGHT":Wh,"BITMAP WIDTH":Ww)
1290       CONTROL @Main;SET ("SCROLL HEIGHT":2+INT(Wh/Sc))
1300       CONTROL @Main;SET ("SCROLL WIDTH":2+INT(Ww/Sc))
1310       CONTROL @Bitmap;SET ("VISIBLE":1)
1320       DIALOG "INFORMATION","File read completed"
1330     END IF
1340   END IF
1350   RETURN
1360   !
1370   ! This routine allows you to cut a section of a bitmap and
1380   ! save it in a file. String variables are used in the FILE
1390   ! dialog to specify the DIALOG BUTTONS attribute and the
1400   ! SELECTION attribute to keep a compact statement length.
1410   !
1420 Savebits: !
1430   S$="Enter file name, then click-and-drag image to save:"
1440   DIALOG "FILE",S$,Btn;SET (A1$:Btns$(*)),RETURN (A2$:Bitfile$)
1450   !
1460   SELECT Btn
1470   CASE 0
1480     S$="BMP"                ! Dump format is MS-Windows .BMP file.
1490   CASE 1
1500     S$="XWD"                ! Dump format is X11 XWD format.
1510   CASE 2
1520     RETURN
1530   END SELECT
1540   !
1550   CLEAR ERROR
1560   ON ERROR GOSUB Errtrap
1570   CONTROL @Bitmap;SET ("DUMP FORMAT":S$,"DUMP AREA":Bitfile$)
1580   OFF ERROR
1590   IF ERRN<>0 THEN
1600     DIALOG "ERROR","Can't open file."
1610   ELSE
1620     DIALOG "INFORMATION","Bitmap saved to "&S$&" file."
1630   END IF
1640   !
1650   RETURN
1660   !
1670   ! This routine changes directories
```

```
1680   !
1690 Chdir: !
1700   S$="Please enter the name of a directory:"
1710   DIALOG "FILE",S$,Btn;RETURN ("DIRECTORY":Dirname$)
1720   !
1730   Err=0
1740   CLEAR ERROR
1750   ON ERROR GOSUB Errtrap
1760   MASS STORAGE IS Dirname$
1770   OFF ERROR
1780   IF ERRN<>0 THEN
1790     DIALOG "ERROR","Can't change directory"
1800   END IF
1810   RETURN
1820   !
1830   ! Dummy routine for trapping errors
1840   !
1850 Errtrap: ERROR RETURN
1860   !
1870 Finis: !
1880   ASSIGN @Main TO *                  ! Delete PANEL widget
1890   CLEAR SCREEN
1900   END
```

```
10    ! ****************************************************************
20    !   Example: Background Bitmap
30    !
40    !   This program demonstrates the BACKGROUND BITMAP attribute
50    !   of the PANEL widget by allowing you to bring in a bitmap
60    !   file and use it to tile the PANEL.
70    !
80    ! ****************************************************************
90    !
100   ! Some variables:
110   !
120   !    S$:            String variable
130   !    M$:            SYSTEM MENU array
140   !    F$:            Bitmap file name
150   !    N:             INTEGER variable
160   !    Btn:           Variable to get button inputs from dialogs
170   !    D(*):          Array to get display dimensions
180   !    Dw,Dh:                 Display dimensions
190   !
200   DIM S$[256],M$(0:1)[32],F$[50]
210   INTEGER N,Btn,D(1:4),Dw,Dh
220   !
230   ! Get display size
240   !
250   GESCAPE CRT,3;D(*)              ! Gets display lines
260   Dw=D(3)-D(1)
270   Dh=(D(4)-D(2))
280   !
290   ! Set up SYSTEM MENU array
300   !
310   M$(0)="Get Bitmap File"
320   M$(1)="Quit"
330   !
340   ! Create the PANEL widget, add a toaster menu
350   !
360   CLEAR SCREEN
370   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
380   CONTROL @Main;SET ("X":50,"Y":20,"WIDTH":.8*Dw,"HEIGHT":.8*Dh)
390   CONTROL @Main;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
400   CONTROL @Main;SET ("TITLE":" Example: Background Bitmap")
410   CONTROL @Main;SET ("SYSTEM MENU":M$(*),"VISIBLE":1)
```

```
420   !
430   ON EVENT @Main,"SYSTEM MENU" GOSUB Handler
440   LOOP
450     WAIT FOR EVENT
460   END LOOP
470   STOP
480   !
490   ! ***************** End of Main Program ***********************
500   !
510 Handler: !
520   STATUS @Main;RETURN ("SYSTEM MENU EVENT":N)
530   SELECT N
540   CASE 0
550     S$="Please enter bitmap file name:"
560     DIALOG "FILE",S$,Btn;RETURN ("SELECTION":F$)
570     IF Btn=0 THEN
580       CLEAR ERROR
590       ON ERROR GOSUB Errtrap
600       CONTROL @Main;SET ("BACKGROUND BITMAP":F$)
610       OFF ERROR
620       IF ERRN<>0 THEN DIALOG "ERROR",ERRM$
630     END IF
640   CASE 1
650     GOTO Finis
660   END SELECT
670   RETURN
680   !
690 Errtrap: ERROR RETURN
700   !
710 Finis: !
720   ASSIGN @Main TO *                ! Delete PANEL Widget
730   END
```

```
10      ! ****************************************************************
20      ! Example: Bar/Meter/Limits
30      !
40      ! This program demonstrates the use of the BAR, METER, and LIMITS
50      ! widgets.
60      !
70      ! The program creates three widgets and sets them up and down through
80      ! their default range of 1 through 100. You can set the LOW LIMIT
90      ! and HIGH LIMIT thresholds on both the widgets using a pair of SLIDER
100     ! widgets. As the program passes through the thresholds, it changes
110     ! the color and string on a LABEL via use of events.
120     !
130     ! The program sets up the widgets, sets up events on the SLIDERS and
140     ! BAR (no event is set up on other widgets, since the method is the
150     ! same), and then goes through a loop from 1 to 100 and then back
160     ! down again, changing the widgets with each count.
170     !
180     ! A routine called "Lobound" handles the SLIDER that sets the lower
190     ! trip limit, a routine called "Hibound" that handles the LIMIT that
200     ! sets the higher trip limit, and a routine called "Trip" that handles
210     ! the BAR events.
220     !
230     ! Some features of this program are:
240     !
250     !   - The program will NOT allow you to set the HIGH LIMIT below the
260     !     LOW LIMIT, or the inverse. Try to scroll into these regions and
270     !     the slidebar will pop back.
280     !
290     !   - Normally a BAR event is "level sensitive": if you set an event
300     !     to happen on the "HIGH" ALARM RANGE, you get an event each and
310     !     every time you write a value to that range. For this program,
320     !     however, an event occurs only on transition between ALARM RANGEs.
330     !
340     ! ****************************************************************
350     !
360     ! Variables used:
370     !
380     !   N:          Loop counter
390     !   Hibound:    Stores METER HIGH LIMIT
400     !   Lobound:    Stores METER LOW LIMIT
410     !   Sts:        Used to RETURN items from widgets
```

```
420   !
430   INTEGER N,Hibound,Lobound,Sts
440   Hibound=100
450   Lobound=1
460   !
470   ! Define colors
480   !
490   INTEGER Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
500   DATA 0,1,2,3,4,5,6,7
510   READ Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
520   !
530   ! Declare variables for display handling
540   !
550   INTEGER D(1:4),Dw,Dh,Cursor
560   INTEGER Pw,Ph,Px,Py,Iw,Ih,Gx,Gy
570   INTEGER Mx,My,Mh,Mw,Brx,Bry,Brh,Brw
580   INTEGER Lmx,Lmy,Lmh,Lmw,Lx,Ly,Lh,Lw
590   INTEGER Lshx,Lshy,Lshw,Lshh,S2x,S2y,S2w,S2h
600   INTEGER Lslx,Lsly,Lslw,Lslh,Slx,Sly,Slw,Slh
610   !
620   ! Get display dimensions
630   !
640   GESCAPE CRT,3;D(*)
650   Dw=D(3)-D(1)
660   Dh=D(4)-D(2)
670   !
680   ! Set up PANEL dimensions, so that PANEL scales to display
690   !
700   Pw=Dw*.8                        ! PANEL width
710   Ph=Dh*.9                        ! PANEL height
720   Px=(Dw-Pw)/2              ! Center PANEL along width
730   Py=(Dh-Ph)/2              ! Center PANEL above DISP line
740   !
750   ! Build main PANEL
760   !
770   CLEAR SCREEN
780   !
790   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
800   CONTROL @Main;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
810   CONTROL @Main;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
820   CONTROL @Main;SET ("TITLE":" Example: Bar Meter Limits")
830   CONTROL @Main;SET ("SYSTEM MENU":"Quit")
```

```
840   !
850   ! Get interior dimensions of PANEL
860   !
870   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
880   !
890   ! Set up widget dimensions
900   !
910   Gx=Iw*.02                          ! Gap between widgets
920   Gy=Gx
930   !
940   Mw=Iw*.45                  ! METER parameters
950   Mh=Ih*.5
960   Mx=Gx
970   My=Gy
980   !
990   Brw=(Iw-(Mw+2*Gx))*.3      ! BAR parameters
1000  Brh=Mh
1010  Brx=Mx+Mw+Gx
1020  Bry=My
1030  !
1040  Lmw=Mw+Gy+Brw              ! LIMITS parameters
1050  Lmh=Ih-(Mh+3*Gy)
1060  Lmx=Gx
1070  Lmy=My+Mh+Gy
1080  !
1090  Lw=Iw-(Mw+Brw+4*Gy)        ! Indicator LABEL parameters
1100  Lh=Lw*.5
1110  Lx=Iw-(Lw+Gx)
1120  Ly=Gy
1130  !
1140  Lshw=(Lw-Gy)/2             ! High limit LABEL
1150  Lshh=Lshw*.4
1160  Lshx=Lx
1170  Lshy=Ly+Lh+Gy
1180  !
1190  Shw=Lshw                   ! High limit SLIDER
1200  Shh=(Ih-(4*Gy+Lh+Lshh))
1210  Shx=Lx
1220  Shy=Lshy+Lshh+Gy
1230  !
1240  Lslw=Lshw                  ! Low limit LABEL
1250  Lslh=Lshh
```

```
1260  Lslx=Lshx+Lshw+Gx
1270  Lsly=Lshy
1280  !
1290  Slw=Shw                                  ! Low limit SLIDER
1300  Slh=Shh
1310  Slx=Lslx
1320  Sly=Shy
1330  !
1340  ! Build rest of PANEL
1350  !
1360  ! Create METER. (There is no need to set ALL these
1370  ! attributes, but this is an example program).
1380  !
1390  ! The three METER ranges are assigned as red, white and blue -
1400  ! an arbitrary choice.
1410  !
1420  ASSIGN @Meter TO WIDGET "METER";PARENT @Main
1430  CONTROL @Meter;SET ("X":Mx,"Y":My,"WIDTH":Mw,"HEIGHT":Mh)
1440  CONTROL @Meter;SET ("HIGH PEN":Red,"MIDDLE PEN":White,"LOW PEN":Blue)
1450  CONTROL @Meter;SET ("VALUE BACKGROUND":Yellow,"VALUE PEN":Black)
1460  CONTROL @Meter;SET ("LIMITS BACKGROUND":Blue,"LIMITS PEN":White)
1470  CONTROL @Meter;SET ("NEEDLE PEN":Blue,"METER BACKGROUND":Green)
1480  CONTROL @Meter;SET ("SWEEP ANGLE":175)
1490  !
1500  ! Create BAR. Basically the same, but not as many options.
1510  !
1520  ASSIGN @Bar TO WIDGET "BAR";PARENT @Main
1530  CONTROL @Bar;SET ("X":Brx,"Y":Bry,"WIDTH":Brw,"HEIGHT":Brh)
1540  CONTROL @Bar;SET ("HIGH PEN":Red,"MIDDLE PEN":White,"LOW PEN":Blue)
1550  !
1560  ! Create LIMITS
1570  !
1580  ASSIGN @Lim TO WIDGET "LIMITS";PARENT @Main
1590  CONTROL @Lim;SET ("X":Lmx,"Y":Lmy,"WIDTH":Lmw,"HEIGHT":Lmh)
1600  CONTROL @Lim;SET ("INSIDE PEN":White,"OUTSIDE PEN":Magenta)
1610  !
1620  ! Create LABEL to indicate METER range
1630  !
1640  ASSIGN @Label TO WIDGET "LABEL";PARENT @Main
1650  CONTROL @Label;SET ("X":Lx,"Y":Ly,"WIDTH":Lw,"HEIGHT":Lh)
1660  CONTROL @Label;SET ("BORDER":0,"BACKGROUND":White,"VALUE":"NORM")
1670  !
```

```
1680   ! Create LABEL for HIGH LIMIT SLIDER
1690   !
1700   ASSIGN @Lsh TO WIDGET "LABEL";PARENT @Main
1710   CONTROL @Lsh;SET ("X":Lshx,"Y":Lshy,"WIDTH":Lshw,"HEIGHT":Lshh)
1720   CONTROL @Lsh;SET ("VALUE":"High")
1730   !
1740   ! Create SLIDER to set HIGH LIMIT
1750   !
1760   ASSIGN @Sh TO WIDGET "SLIDER";PARENT @Main
1770   CONTROL @Sh;SET ("X":Shx,"Y":Shy,"WIDTH":Shw,"HEIGHT":Shh)
1780   CONTROL @Sh;SET ("DIRECT MOVE":1,"AUTO REPEAT":1)
1790   CONTROL @Sh;SET ("VALUE":100)
1800   !
1810   ! Create LABEL for LOW LIMIT SLIDER
1820   !
1830   ASSIGN @Lsl TO WIDGET "LABEL";PARENT @Main
1840   CONTROL @Lsl;SET ("X":Lslx,"Y":Lsly,"WIDTH":Lslw,"HEIGHT":Lslh)
1850   CONTROL @Lsl;SET ("VALUE":"Low")
1860   !
1870   ! Create SLIDER to set LOW LIMIT
1880   !
1890   ASSIGN @Sl TO WIDGET "SLIDER";PARENT @Main
1900   CONTROL @Sl;SET ("X":Slx,"Y":Sly,"WIDTH":Slw,"HEIGHT":Slh)
1910   CONTROL @Sl;SET ("DIRECT MOVE":1,"AUTO REPEAT":1)
1920   !
1930   ! Call bounds handlers to set up initial bounds, then
1940   ! make main PANEL visible. (The bounds handlers will
1950   ! both call the "Trip" routine to initialize the BAR event.)
1960   !
1970   GOSUB Hibound
1980   GOSUB Lobound
1990   CONTROL @Main;SET ("VISIBLE":1)
2000   !
2010   ! Set up the event handlers
2020   !
2030   N=1
2040   ON EVENT @Sl,"DONE" GOSUB Lobound                    ! Event for LOW LIMIT
SLIDER
2050   ON EVENT @Sh,"DONE" GOSUB Hibound                    ! Event for HI LIMIT SLIDER
2060   ON EVENT @Bar,"ALARM" GOSUB Trip                     ! Event for BAR
2070   ON EVENT @Main,"SYSTEM MENU" GOTO Finis              ! Event for QUIT toaster
menu
2080   !
```

```
2090   ! Count from 1 to 100 and back down, and set METER and
2100   ! BAR as this is done.
2110   !
2120   LOOP
2130     FOR N=1 TO 100
2140       CONTROL @Meter;SET ("VALUE":N)
2150       CONTROL @Bar;SET ("VALUE":N)
2160       CONTROL @Lim;SET ("VALUE":N)
2170     NEXT N
2180     FOR N=100 TO 1 STEP -1
2190       CONTROL @Meter;SET ("VALUE":N)
2200       CONTROL @Bar;SET ("VALUE":N)
2210       CONTROL @Lim;SET ("VALUE":N)
2220     NEXT N
2230   END LOOP
2240   !
2250   ! ********************* End of Main Program ************************
2260   !
2270   ! Handler for LOW LIMIT SLIDER:
2280   !
2290 Lobound: !
2300   DISABLE
2310   STATUS @Sl;RETURN ("VALUE":Lobound)              ! Get SLIDER VALUE
2320   IF Lobound>Hibound THEN                          ! If higher than HIGH LIMIT:
2330     Lobound=Hibound                                ! Make sure it is not
2340     CONTROL @Sl;SET ("VALUE":Lobound)
2350   END IF
2360   CONTROL @Meter;SET ("LOW LIMIT":Lobound)         ! Set limit on widgets
2370   CONTROL @Bar;SET ("LOW LIMIT":Lobound)
2380   CONTROL @Lim;SET ("LOW LIMIT":Lobound)
2390   GOSUB Trip                                       ! Adjust BAR event handling
2400   ENABLE
2410   RETURN
2420   !
2430   ! HIGH LIMIT SCROLL BAR event handler
2440   !
2450 Hibound: !
2460   DISABLE
2470   STATUS @Sh;RETURN ("VALUE":Hibound)              ! Get SLIDER VALUE
2480   IF Hibound<Lobound THEN                          ! If HIGH LIMIT < LOW LIMIT:
2490     Hibound=Lobound                                ! Make sure it is not
2500     CONTROL @Sh;SET ("VALUE":Hibound)
```

```
2510   END IF
2520   CONTROL @Meter;SET ("HIGH LIMIT":Hibound) ! Reflect on METER & LIMITS
2530   CONTROL @Bar;SET ("HIGH LIMIT":Hibound)
2540   CONTROL @Lim;SET ("HIGH LIMIT":Hibound)
2550   GOSUB Trip                                          ! Adjust event handling
2560   ENABLE
2570   RETURN
2580   !
2590   ! BAR event handler.
2600   ! You get the current value being sent to the BAR and determine
2610   ! the range it is in. Then, you set the event to occur on the
2620   ! adjoining range(s). This ensures that you only get an event when
2630   ! you transition between ranges.
2640   !
2650   ! As noted, the bounds handlers call Trip to adjust events. Since
2660   ! you are changing the HIGH & LOW LIMITs with the bounds handlers,
2670   ! you want to make sure that the events are rearranged accordingly.
2680   !
2690 Trip: !
2700   DISABLE
2710   SELECT N
2720   CASE <Lobound
2730     CONTROL @Bar;SET ("ALARM RANGES":"MIDDLE","ALARM TYPE":"EVENT")
2740     CONTROL @Label;SET ("BACKGROUND":Blue,"VALUE":"LOW")
2750   CASE <=Hibound
2760     CONTROL @Bar;SET ("ALARM RANGES":"LOW,HIGH","ALARM TYPE":"EVENT")
2770     CONTROL @Label;SET ("BACKGROUND":White,"VALUE":"NORM")
2780   CASE >Hibound
2790     CONTROL @Bar;SET ("ALARM RANGES":"MIDDLE","ALARM TYPE":"EVENT")
2800     CONTROL @Label;SET ("BACKGROUND":Red,"VALUE":"HIGH")
2810   END SELECT
2820   ENABLE
2830   RETURN
2840   !
2850 Finis: !
2860   ASSIGN @Main TO *      ! Delete PANEL widget
2870   END
```

```
10      ! ********************************************************************
20      ! Example: Bomb Squad (*CREATE Version)
30      !
40      ! This program demonstrates the use of the CLOCK widget in TIMER mode
50      ! by playing a game in which the user has to disarm a bomb by cutting
60      ! wires.   There are 10 wires - you must cut the correct four to disarm
70      ! the bomb.   Of the six wires left, four are don't-cares and two wires
80      ! cause the bomb go off immediately.
90      !
100     ! The wires are represented by 10 TOGGLEBUTTONs. This program uses the
110     ! SYSTEM widget to create the TOGGLEBUTTONs. This is convenient since
120     ! otherwise you would have to have some degree of separate code for
130     ! each TOGGLEBUTTON.   The fact that you only get one event for all
140     ! 10 TOGGLEBUTTONs is all right, too, since each time a TOGGLEBUTTON
150     ! event happens, the program scans through all the TOGGLEBUTTONs to
160     ! check their VALUEs.
170     !
180     ! ********************************************************************
190     !
200     RANDOMIZE INT(10^7*FRACT(TIMEDATE))   ! Set random seed
210     !
220     ! Miscellaneous general-purpose variables
230     !
240     INTEGER N,V
250     DIM S$[256],B$(1:1)[64],Eol$[2]
260     Eol$=CHR$(13)
270     !
280     ! Variables for display and widget handling
290     !
300     INTEGER Nlines,D(1:4)                   ! Used for display setup
310     REAL Dw,Dh                              ! Display dimensions
320     REAL Px,Py,Pw,Ph,Iw,Ih                  ! Main PANEL parameters
330     REAL Gap,Bh,Bw,Ch,Cw,Cx,Cy              ! Button/Clock dimensions
340     REAL Prh,Prw,Prx,Pry                    ! Printer widget dimensions
350     !
360     ! Various variables for playing the game
370     !
380     INTEGER Playgame                        ! Indicates game in progress
390     INTEGER Wires(1:10)                     ! Designates wire settings
400     INTEGER Live,Kill,Dontcare,Cut          ! Wire values
410     DATA 1,2,3,4
420     READ Live,Deadly,Dontcare,Cut
```

```
430   INTEGER Livewires,Lethal                    ! Number of live/deadly wires
440   !
450   ! Set up display
460   !
470   CLEAR SCREEN
480   STATUS CRT,13;Nlines                ! Get number of display lines
490   GESCAPE CRT,3;D(*)                  ! Get BASIC display size
500   Dw=D(3)-D(1)                        ! Display width
510   Dh=(D(4)-D(2))*((Nlines-7)/Nlines)          ! Display height above softkeys
520   !
530   ! Set up PANEL coordinates
540   !
550   Pw=Dw*.7
560   Ph=Dh*.9
570   Px=(Dw-Pw)/2
580   Py=(Dh-Ph)/2
590   !
600   ! Create a SYSTEM widget
610   !
620   COM @Sys
630   ASSIGN @Sys TO WIDGET "SYSTEM"
640   !
650   ! Create a PANEL on the SYSTEM widget with SYSTEM MENU attribute
660   !
670   CONTROL @Sys;SET ("*NAME":"Main","*CREATE":"PANEL","VISIBLE":0)
680   CONTROL @Sys;SET ("RESIZABLE":0,"MAXIMIZABLE":0)
690   CONTROL @Sys;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
700   CONTROL @Sys;SET ("TITLE":" Example: Bomb Squad (*CREATE)")
710   CONTROL @Sys;SET ("SYSTEM MENU":"Quit")
720   !
730   ! Get interior dimensions of PANEL
740   !
750   STATUS @Sys;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
760   !
770   ! Set up widget coordinates and dimensions
780   !
790   Gap=Ih*.03                  ! Vertical gap
800   Bh=Ih*.1                    ! Button height
810   Bw=Iw*.2                          ! Button width
820   !
830   Ch=Iw*.20                         ! Clock height
840   Cw=Ch                       ! Clock Width
```

```
850   Cx=Bw+((Iw-Bw)-Cw)/2          ! Clock X location
860   Cy=Gap                        ! Clock Y coordinate
870   !
880   Prh=Ih-(Ch+3*Gap)                   ! Printer height
890   Prw=((Iw-Bw)-2*Gap)           ! Printer width
900   Prx=Bw+Gap                    ! Printer X coordinate
910   Pry=Ch+2*Gap                  ! Printer Y coordinate
920   !
930   ! Set up buttons in PANEL. The SYSTEM widget is used
940   ! to good advantage here, since the same code can
950   ! create all ten TOGGLEBUTTONS.
960   !
970   FOR N=1 TO 10
980     S$=VAL$(N)
990     CONTROL @Sys;SET ("*NAME":"Main/T"&S$,"*CREATE":"TOGGLEBUTTON")
1000    CONTROL @Sys;SET ("X":0,"Y":(N-1)*Bh,"WIDTH":Bw,"HEIGHT":Bh)
1010    CONTROL @Sys;SET ("LABEL":"Wire "&S$)
1020  NEXT N
1030  !
1040  ! Create CLOCK, set up as down-counting TIMER. The TIMER
1050  ! limit does not have to be set - it is 0 by default.
1060  !
1070  CONTROL @Sys;SET ("*NAME":"Main/Clock","*CREATE":"CLOCK")
1080  CONTROL @Sys;SET ("X":Cx,"Y":Cy,"WIDTH":Cw,"HEIGHT":Ch)
1090  CONTROL @Sys;SET ("TYPE":"TIMER","TIMER DIRECTION":"DOWN")
1100  !
1110  ! Create PRINTER
1120  !
1130  CONTROL @Sys;SET ("*NAME":"Main/Printer","*CREATE":"PRINTER")
1140  CONTROL @Sys;SET ("X":Prx,"Y":Pry,"WIDTH":Prw,"HEIGHT":Prh)
1150  !
1160  ! Set up events for SYSTEM MENU, TOGGLEBUTTONS, and CLOCK
1170  !
1180  ON EVENT @Sys,"SYSTEM MENU",15 GOTO Finis
1190  ON EVENT @Sys,"CHANGED" GOSUB Cutwire
1200  ON EVENT @Sys,"TIMER" GOSUB Boomboom
1210  !
1220  ! Turn on panel
1230  !
1240  CONTROL @Sys;SET ("*NAME":"Main","VISIBLE":1)
1250  !
1260  ! Display instructions using a DIALOG
```

```
1270  !
1280  S$="Disarm the bomb before the clock times out"&Eol$
1290  S$=S$&Eol$
1300  S$=S$&"The bomb has ten wires:"&Eol$
1310  S$=S$&Eol$
1320  S$=S$&"   - 4 wires are inert."&Eol$
1330  S$=S$&"   - 4 wires must be cut to disarm the bomb."&Eol$
1340  S$=S$&"   - 2 wires are triggers: cut both, "&Eol$
1350  S$=S$&"         and the bomb goes off."&Eol$
1360  S$=S$&Eol$
1370  S$=S$&"GOOD LUCK!"&Eol$
1380  !
1390  B$(1)="Click Here To Begin Game"
1400  !
1410  DIALOG "INFORMATION",S$;SET ("TITLE":" Bomb Squad
Instructions","JUSTIFICATION":"LEFT","BACKGROUND":9,"PEN":0,"DIALOG BUTTONS":B$(*))
1420  !
1430  ! Main game loop
1440  !
1450  LOOP
1460  !
1470  ! Clear PRINTER widget, set up all the "wires"
1480  !
1490    DISABLE
1500    CONTROL @Sys;SET ("*NAME":"Main/Printer","TEXT":"")
1510    CALL Pr(" Welcome to BOMB SQUAD.")
1520    CALL Pr("")
1530  !
1540    FOR N=1 TO 10
1550      Wires(N)=Dontcare
1560      S$=VAL$(N)
1570      CONTROL @Sys;SET ("*NAME":"Main/T"&S$)
1580      CONTROL @Sys;SET ("SENSITIVE":1,"VALUE":0)
1590    NEXT N
1600  !
1610  ! Set up the deadly wires
1620  !
1630    Lethal=0
1640    REPEAT
1650      N=1+INT(10*RND)
1660      IF Wires(N)=Dontcare THEN
1670        Wires(N)=Deadly
```

```
1680          Lethal=Lethal+1
1690        END IF
1700      UNTIL (Lethal=2)
1710    !
1720    ! Set up the live wires
1730    !
1740      Livewires=0
1750      REPEAT
1760        N=1+INT(10*RND)
1770        IF Wires(N)=Dontcare THEN
1780          Wires(N)=Live
1790          Livewires=Livewires+1
1800        END IF
1810      UNTIL (Livewires=4)
1820    !
1830    ! Set the timer to 30 seconds, and start it running
1840    !
1850      CONTROL @Sys;SET ("*NAME":"Main/Clock")
1860      CONTROL @Sys;SET ("TIMER VALUE":30000,"TIMER STATE":"RUNNING")
1870    !
1880    ! Loop until game over. Note how the "Playgame" variable is
1890    ! set to 1 by EVENT-driven routines to tell the main routine
1900    ! that the game is over and that a new one should be started
1910    ! (by returning to the top of the loop).
1920    !
1930      ENABLE
1940      Playgame=0
1950      REPEAT
1960      UNTIL (Playgame=1)
1970    !
1980    END LOOP
1990    STOP
2000    !
2010    ! This routine checks the status of the "wire" togglebuttons.
2020    ! It relies on the "Wires" array to track the condition
2030    ! of the wire set at any time.
2040    !
2050    ! The following comments may help explain the routine.
2060    !
2070  Cutwire: !
2080    !
2090    ! Check status of all ten wires
```

```
2100  !
2110  FOR N=1 TO 10
2120  !
2130  ! Ignore the wire if it has been cut
2140  !
2150     IF Wires(N)<>Cut THEN
2160  !
2170  ! Otherwise, query the togglebutton value
2180  !
2190        S$=VAL$(N)
2200        CONTROL @Sys;SET ("*NAME":"Main/T"&S$)
2210        STATUS @Sys;RETURN ("VALUE":V)
2220  !
2230  ! Ignore the button if it is not set, otherwise ...
2240  !
2250        IF V=1 THEN
2260  !
2270  ! ... disable the button ...
2280  !
2290           CONTROL @Sys;SET ("SENSITIVE":0)
2300  !
2310  ! ... and take the appropriate measures for the wire value
2320  !
2330           SELECT Wires(N)
2340  !
2350  ! Don't care, just say so
2360  !
2370           CASE Dontcare
2380              CALL Pr("Inert wire.")
2390  !
2400  ! Live wire: decrement the live-wire count - if it reaches 0,
2410  ! you win. Use a DIALOG to indicate the matter and query to see
2420  ! if the user wants to play another game. If it is not 0,
2430  ! announce the wire has been cut and list the number of wires
2440  ! remaining.
2450  !
2460           CASE Live
2470              Livewires=Livewires-1
2480              IF Livewires=0 THEN
2490                 CONTROL @Sys;SET ("*NAME":"Main/Clock")
2500                 CONTROL @Sys;SET ("TIMER STATE":"STOPPED")
2510                 S$="Play another game?"
```

```
2520              DIALOG "QUESTION",S$,Btn;SET ("TITLE":" Bomb Disarmed !!")
2530              SELECT Btn
2540              CASE 0
2550                Playgame=1                  ! Start new game
2560                RETURN
2570              CASE 1
2580                GOTO Finis                  ! Quit program
2590              END SELECT
2600            ELSE
2610              S$=VAL$(Livewires)
2620              CALL Pr("LIVE WIRE -- "&S$&" wires left.")
2630            END IF
2640  !
2650  ! Is deadly -- count down deadly wires, if zero, you are dead
2660  !
2670          CASE Deadly
2680            Lethal=Lethal-1
2690            IF Lethal=0 THEN
2700              GOSUB Boomboom
2710              RETURN
2720            ELSE
2730              CALL Pr("DANGER -- trigger wire, one left!")
2740            END IF
2750          END SELECT
2760  !
2770  ! If you have not either won or been killed, mark this wire as
2780  ! being "cut".
2790  !
2800          Wires(N)=Cut
2810  !
2820        END IF
2830      END IF
2840    NEXT N
2850    RETURN
2860  !
2870  ! This routine tells you that you are dead
2880  !
2890 Boomboom: !
2900    CONTROL @Sys;SET ("*NAME":"Main/Clock","TIMER STATE":"STOPPED")
2910    S$="YOU'RE DEAD!"
2920    S$=S$&Eol$&Eol$
2930    S$=S$&"Play another game?"
```

```
2940   DIALOG "QUESTION",S$,Btn;SET ("TITLE":" Bomb Exploded !!")
2950   IF Btn=0 THEN
2960      Playgame=1
2970      RETURN
2980   ELSE
2990      GOTO Finis
3000   END IF
3010   RETURN
3020   !
3030   ! Go here when done
3040   !
3050 Finis: !
3060   ASSIGN @Sys TO *                    ! Delete SYSTEM widget
3070   END
3080   !
3090   ! *************** End of Main Program ***********************
3100   !
3110   ! Routine to print string in PRINTER widget
3120   !
3130   SUB Pr(S$)
3140      COM @Sys
3150      CONTROL @Sys;SET ("*NAME":"Main/Printer","APPEND TEXT":S$)
3160   SUBEND
```

```
10      ! ************************************************************************
20      ! Example: Bomb Squad (*LOAD Version)
30      !
40      ! This program demonstrates the use of the CLOCK widget in TIMER mode
50      ! by playing a game in which the user has to disarm a bomb by cutting
60      ! wires. There are 10 wires. You must cut the correct four to disarm
70      ! the bomb. Of the six wires left, four are don't-cares and two cause
80      ! the bomb go off immediately.
90      !
100     ! The wires are represented by 10 TOGGLEBUTTONs. This program uses the
110     ! SYSTEM widget to create the TOGGLEBUTTONs. This is convenient, since
120     ! otherwise you would have to have some degree of separate code for
130     ! each TOGGLEBUTTON. The fact that you only get one event for all
140     ! 10 TOGGLEBUTTONs is OK, too, since each time a TOGGLEBUTTON event
150     ! happens the program scans through all the TOGGLEBUTTONs to check
160     ! their VALUEs.
170     !
180     ! ************************************************************************
190     !
200     RANDOMIZE INT(10^7*FRACT(TIMEDATE))   ! Set random seed
210     !
220     ! Miscellaneous general-purpose variables
230     !
240     INTEGER N,V
250     DIM S$[256],B$(1:1)[64],Eol$[2]
260     Eol$=CHR$(13)
270     !
280     ! Variables for playing the game
290     !
300     INTEGER Playgame                    ! Indicates game in progress
310     INTEGER Wires(1:10)                 ! Designates wire settings
320     INTEGER Live,Kill,Dontcare,Cut      ! Wire values
330     DATA 1,2,3,4
340     READ Live,Deadly,Dontcare,Cut
350     INTEGER Livewires,Lethal                    ! Number of live/deadly wires
360     !
370     CLEAR SCREEN
380     !
390     ! Create a SYSTEM widget and load Screen Builder file
400     !
410     COM @Sys
```

```
420   ASSIGN @Sys TO WIDGET "SYSTEM"
430   CONTROL @Sys;SET ("*LOAD":"BSQUAD.SCR")
440   !
450   ! Set up events for SYSTEM MENU, TOGGLEBUTTONS, and CLOCK
460   !
470   ON EVENT @Sys,"SYSTEM MENU",15 GOTO Finis
480   ON EVENT @Sys,"CHANGED" GOSUB Cutwire
490   ON EVENT @Sys,"TIMER" GOSUB Boomboom
500   !
510   ! Turn on panel
520   !
530   CONTROL @Sys;SET ("*NAME":"Main","VISIBLE":1)
540   !
550   ! Display instructions using a DIALOG
560   !
570   S$="Disarm the bomb before the clock times out"&Eol$
580   S$=S$&Eol$
590   S$=S$&"The bomb has ten wires:"&Eol$
600   S$=S$&Eol$
610   S$=S$&"   - 4 are inert."&Eol$
620   S$=S$&"   - 4 must be cut to disarm the bomb."&Eol$
630   S$=S$&"   - 2 are triggers: cut both, "&Eol$
640   S$=S$&"         and the bomb goes off."&Eol$
650   S$=S$&Eol$
660   S$=S$&"GOOD LUCK!"&Eol$
670   !
680   B$(1)="Click Here To Begin Game"
690   !
700   DIALOG "INFORMATION",S$;SET ("TITLE":" Bomb Squad
Instructions","BACKGROUND":9,"PEN":0,"JUSTIFICATION":"LEFT","DIALOG BUTTONS":B$(*))
710   !
720   ! Main game loop.
730   !
740   LOOP
750   !
760   ! Clear PRINTER widget, set up all the "wires".
770   !
780     DISABLE
790     CONTROL @Sys;SET ("*NAME":"Main/Printer","TEXT":"")
800     CALL Pr("Welcome to BOMB SQUAD.")
810     CALL Pr("")
820   !
```

```
830     FOR N=1 TO 10
840       Wires(N)=Dontcare
850       S$=VAL$(N)
860       CONTROL @Sys;SET ("*NAME":"Main/T"&S$)
870       CONTROL @Sys;SET ("SENSITIVE":1,"VALUE":0)
880     NEXT N
890   !
900   ! Set the deadly wires.
910   !
920     Lethal=0
930     REPEAT
940       N=1+INT(10*RND)
950       IF Wires(N)=Dontcare THEN
960         Wires(N)=Deadly
970         Lethal=Lethal+1
980       END IF
990     UNTIL (Lethal=2)
1000  !
1010  ! Set up the live wires.
1020  !
1030    Livewires=0
1040    REPEAT
1050      N=1+INT(10*RND)
1060      IF Wires(N)=Dontcare THEN
1070        Wires(N)=Live
1080        Livewires=Livewires+1
1090      END IF
1100    UNTIL (Livewires=4)
1110  !
1120  ! Set the timer to 30 seconds, start it running.
1130  !
1140    CONTROL @Sys;SET ("*NAME":"Main/Clock")
1150    CONTROL @Sys;SET ("TIMER VALUE":30000,"TIMER STATE":"RUNNING")
1160  !
1170  ! Loop until game over. Note how the "Playgame" variable is
1180  ! set to 1 by EVENT-driven routines to tell the main routine that
1190  ! the game is over and that a new one should be started (by
1200  ! returning to the top of the loop).
1210  !
1220    ENABLE
1230    Playgame=0
1240    REPEAT
```

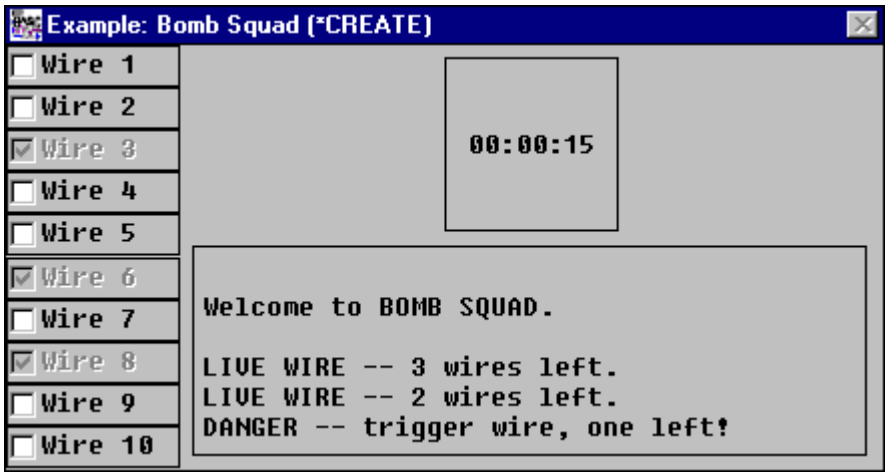
```
1250     UNTIL (Playgame=1)
1260   !
1270   END LOOP
1280   STOP
1290   !
1300   ! This routine checks the status of the "wire" togglebuttons.
1310   ! It relies on the "Wires" array to track the condition of
1320   ! the wire set at any time.
1330   !
1340 Cutwire: !
1350   !
1360   ! Check status of all ten wires.
1370   !
1380   FOR N=1 TO 10
1390   !
1400   ! Ignore the wire if it has been cut.
1410   !
1420     IF Wires(N)<>Cut THEN
1430   !
1440   ! Otherwise, query the togglebutton value.
1450   !
1460       S$=VAL$(N)
1470       CONTROL @Sys;SET ("*NAME":"Main/T"&S$)
1480       STATUS @Sys;RETURN ("VALUE":V)
1490   !
1500   ! Ignore the button if it is not set, otherwise ...
1510   !
1520       IF V=1 THEN
1530   !
1540   ! ... disable the button ...
1550   !
1560         CONTROL @Sys;SET ("SENSITIVE":0)
1570   !
1580   ! ... and take the appropriate measures for the wire value.
1590   !
1600         SELECT Wires(N)
1610   !
1620   ! Don't care, just say so.
1630   !
1640         CASE Dontcare
1650           CALL Pr("Inert wire.")
1660   !
```

```
1670  ! Live wire: decrement the live-wire count -- if it reaches 0,
1680  ! you win. Use a DIALOG to indicate the matter and query to see
1690  ! if the user wants to play another game.    If it is not 0,
1700  ! announce the wire has been cut and list the number of wires
1710  ! remaining.
1720  !
1730          CASE Live
1740            Livewires=Livewires-1
1750            IF Livewires=0 THEN
1760              CONTROL @Sys;SET ("*NAME":"Main/Clock")
1770              CONTROL @Sys;SET ("TIMER STATE":"STOPPED")
1780              S$="Play another game?"
1790              DIALOG "QUESTION",S$,Btn;SET ("TITLE":" Bomb Disarmed !! ")
1800              SELECT Btn
1810              CASE 0
1820                Playgame=1                    ! Start new game.
1830                RETURN
1840              CASE 1
1850                GOTO Finis                    ! End the program.
1860              END SELECT
1870            ELSE
1880              S$=VAL$(Livewires)
1890              CALL Pr("LIVE WIRE -- "&S$&" wires left.")
1900            END IF
1910  !
1920  ! Is deadly -- count down deadly wires, if zero, you are dead.
1930  !
1940          CASE Deadly
1950            Lethal=Lethal-1
1960            IF Lethal=0 THEN
1970              GOSUB Boomboom
1980              RETURN
1990            ELSE
2000              CALL Pr("DANGER -- trigger wire, one left!")
2010            END IF
2020          END SELECT
2030  !
2040  ! If you have not won or been killed, mark this wire as
2050  ! being "cut".
2060  !
2070          Wires(N)=Cut
2080  !
```

```
2090        END IF
2100      END IF
2110    NEXT N
2120    RETURN
2130    !
2140    ! This routine tells you that the bomb exploded and you are dead
2150    !
2160 Boomboom: !
2170    CONTROL @Sys;SET ("*NAME":"Main/Clock","TIMER STATE":"STOPPED")
2180    S$="YOU'RE DEAD!"
2190    S$=S$&Eol$&Eol$
2200    S$=S$&"Play another game?"
2210    DIALOG "QUESTION",S$,Btn;SET ("TITLE":" Bomb Exploded !! ")
2220    IF Btn=0 THEN
2230      Playgame=1
2240      RETURN
2250    ELSE
2260      GOTO Finis
2270    END IF
2280    RETURN
2290    !
2300    ! Go here when done.
2310    !
2320 Finis: !
2330    ASSIGN @Sys TO *        ! Delete SYSTEM widget
2340    END
2350    !
2360    ! ***************** End of Main Program **********************
2370    !
2380    ! Routine to print string in PRINTER widget.
2390    !
2400    SUB Pr(S$)
2410      COM @Sys
2420      CONTROL @Sys;SET ("*NAME":"Main/Printer","APPEND TEXT":S$)
2430    SUBEND
```

Example: Bomb Squad (Using SYSTEM Widget)

This example is a modified version of the Bomb Squad (*LOAD) program, and uses the SYSTEM widget rather than the Screen Builder application to build the widgets for the game. See Bomb Squad (*CREATE) for a program listing. A typical display for the program follows.



```
10       ! ************************************************
20       ! Example: CLOCK Widget
30       !
40       ! This program creates a default CLOCK widget.
50       !
60       ! ************************************************
70       !
80       ASSIGN @Clock TO WIDGET "CLOCK"
90       CONTROL @Clock;SET ("TITLE":" Example: CLOCK Widget")
100     CONTROL @Clock;SET ("HEIGHT":150,"WIDTH":250)
110     CONTROL @Clock;SET ("X":50,"Y":25)
120     CONTROL @Clock;SET ("SYSTEM MENU":"Quit")
130     ON EVENT @Clock,"SYSTEM MENU" GOTO Finis
140     LOOP
150       WAIT FOR EVENT
160     END LOOP
170 Finis: !
180     ASSIGN @Clock TO *              ! Delete CLOCK widget
190     END
```

```
10    ! ********************************************************
20    ! Example: COMBO Dialog
30    !
40    ! This program creates a COMBO dialog. The user can select
50    ! an item from the supplied list (COSMOPOLITAN, etc.) or
60    ! can enter a name from the keyboard. The user selection is
70    ! displayed after a selection is made and Enter is pressed.
80    !
90    ! ********************************************************
100   !
110   DIM M$(1:7)[20],S$[25],P$[50]
120   INTEGER Btn
130   P$="What is your favorite magazine?"
140   DATA "COSMOPOLITAN","ENQUIRER","DISCOVER","TIME"
150   DATA "HEALTH","SPORTS","NEW YORKER"
160   READ M$(*)
170   !
180   DIALOG "COMBO",P$,Btn;SET ("TITLE":" Example: COMBO Dialog","ITEMS":M$
(*)),RETURN ("TEXT":S$)
190   DISP "Magazine selected:";S$
200   END
```

```
10      ! ******************************************************
20      ! Example: COMBO Test
30      !
40      ! This program creates a COMBO widget and allows the
50      ! user to select an animal name from a defined list
60      ! of names or to enter a name from the keyboard. The
70      ! program displays an error message if the name entered
80      ! is not in the defined list.
90      !
100   ! ******************************************************
110   !
120   DIM L$(1:5)[26],S$[50]
130   INTEGER N
140   !
150   DATA " Aardvark"," Sidewinder"," Kiwi"," Pangolin"," Marmoset"
160   READ L$(*)
170   !
180   ASSIGN @Combo TO WIDGET "COMBO";SET ("SYSTEM MENU":"Quit")
190   CONTROL @Combo;SET ("TITLE":" Example: COMBO Test - Select an Animal")
200   CONTROL @Combo;SET ("BACKGROUND":1,"LIST BACKGROUND":1,"ITEMS":L$(*))
210   CONTROL @Combo;SET ("X":50,"Y":25,"WIDTH":325)
220   CONTROL @Combo;SET ("SYSTEM MENU":"Quit")
230   !
240   ON EVENT @Combo,"SELECTION" GOSUB Handler
250   ON EVENT @Combo,"RETURN" GOSUB Handler
260   ON EVENT @Combo,"SYSTEM MENU" GOTO Finis
270   !
280   LOOP
290     WAIT FOR EVENT
300   END LOOP
310   STOP
320     !
330 Handler:     !
340   STATUS @Combo;RETURN ("TEXT":S$)
350   FOR N=1 TO 5
360     IF S$=L$(N) THEN
370       S$="List item #"&VAL$(N)&": "&S$
380       DIALOG "INFORMATION",S$
390       RETURN
400     END IF
410   NEXT N
```

```
420     S$="Animal not in list: "&S$
430     DIALOG "ERROR",S$;SET ("TITLE":" Invalid Selection")
440     RETURN
450         !
460 Finis:      !
470     ASSIGN @Combo TO *              ! Delete COMBO widget
480     END
```

```
10      ! ***********************************************
20      ! Example: COMBO Widget
30      !
40      ! This program creates a COMBO widget and allows
50      ! the user to select an item from a supplied list
60      ! or to enter text via the keyboard.
70      !
80      ! ***********************************************
90      !
100     DIM Items$(0:15)[26],Sel$[1000]
110     INTEGER I,Sel,Fg,Bg,Lb_fg,Lb_bg
120     Items$(0)="   6 BY 12"
130     Items$(1)="   8 BY 16"
140     Items$(2)="   10 BY 20"
150     Items$(3)="   EDITABLE"
160     Items$(4)="   NOT EDITABLE"
170     Items$(5)="   USE DROPDOWN BUTTON"
180     Items$(6)="   NO DROPDOWN BUTTON"
190     Items$(7)="   SHOW LIST"
200     Items$(8)="   HIDE LIST"
210     Items$(9)="   SHOW SCROLLBAR"
220     Items$(10)="   HIDE SCROLLBAR"
230     Items$(11)="   SET COLORS"
240     FOR I=BASE(Items$,1)+12 TO BASE(Items$,1)+SIZE(Items$,1)-1
250       Items$(I)="item "&VAL$(I)
260     NEXT I
270     ASSIGN @Combo TO WIDGET "COMBO";SET ("ITEMS":Items$(*),"TITLE":" Example:
COMBO Widget","COLUMNS":MAXLEN(Items$(0)))
280     CONTROL @Combo;SET ("X":100,"Y":50,"WIDTH":250,"BACKGROUND":1,"LIST
BACKGROUND":1)
290     CONTROL @Combo;SET ("SYSTEM MENU":"Quit")
300     ON EVENT @Combo,"SELECTION" GOSUB Disp_sel
310     ON EVENT @Combo,"KEYSTROKE" GOSUB Disp_keystroke
320     ON EVENT @Combo,"RETURN" GOSUB Disp_return
330     ON EVENT @Combo,"SYSTEM MENU" GOTO Finis
340     LOOP
350       WAIT FOR EVENT
360     END LOOP
370 !
380 Disp_sel: !
390     STATUS @Combo;RETURN ("SELECTION":Sel)
```

```
400     STATUS @Combo;RETURN ("TEXT":Sel$)
410     DISP
420     DISP "selection: ";Sel;"""";Sel$;""""
430     SELECT Sel
440     CASE 0,1,2
450       CONTROL @Combo;SET ("FONT":Sel$,"COLUMNS":MAXLEN(Items$(0)))
460     CASE 3,4
470       CONTROL @Combo;SET ("EDITABLE":Sel=3)
480     CASE 5,6
490       CONTROL @Combo;SET ("DROPDOWN BUTTON":Sel=5)
500     CASE 7,8
510       CONTROL @Combo;SET ("SHOW LIST":Sel=7)
520     CASE 9,10
530       CONTROL @Combo;SET ("SCROLLBAR":Sel=9)
540     CASE 11
550       CONTROL @Combo;SET ("SENSITIVE":0)
560       STATUS @Combo;RETURN ("PEN":Fg,"BACKGROUND":Bg,"LIST PEN":Lb_fg,"LIST
BACKGROUND":Lb_bg)
570       IF FNQuery_colors(Fg,Bg,Lb_fg,Lb_bg) THEN
580         CONTROL @Combo;SET ("PEN":Fg,"BACKGROUND":Bg,"LIST PEN":Lb_fg,"LIST
BACKGROUND":Lb_bg)
590       END IF
600       CONTROL @Combo;SET ("SENSITIVE":1)
610     END SELECT
620     RETURN
630 !
640 Disp_keystroke: !
650     STATUS @Combo;RETURN ("TEXT":Sel$)
660     DISP
670     DISP "keystroke: """;Sel$;""""
680     RETURN
690 !
700 Disp_return: !
710     STATUS @Combo;RETURN ("TEXT":Sel$)
720     DISP
730     DISP "return: """;Sel$;""""
740     RETURN
750 Finis: !
760     ASSIGN @Combo TO *              ! Delete COMBO widget
770     END
780     !
790     DEF FNQuery_colors(INTEGER Fg,Bg,Lb_fg,Lb_bg)
800       INTEGER Num_pens(0:0),Max_pen,Xoffset,Nw_height,Yoffset,Demo_offset
```

```
810       GESCAPE CRT,1;Num_pens(*)
820       Max_pen=Num_pens(0)
830       Xoffset=10+17*CHRX
840       Demo_offset=Xoffset+7*CHRX
850       Yoffset=10
860       !
870       ! Edit colors
880       !
890       ASSIGN @Panel TO WIDGET "PANEL";SET ("TITLE":"Select COMBO
Colors","X":10,"Y":10,"VISIBLE":0),TRANSIENT
900       ASSIGN @Fg_label TO WIDGET "LABEL";SET ("VALUE":"Edit
Pen","JUSTIFICATION":"RIGHT","X":10,"Y":Yoffset,"COLUMNS":15,"BORDER":0),PARENT
@Panel
910       ASSIGN @Fg TO WIDGET "NUMBER";SET
("X":Xoffset,"Y":Yoffset,"COLUMNS":4,"FORMAT":"SHORT
INTEGER","MINIMUM":0,"MAXIMUM":Max_pen,"VALUE":Fg),PARENT @Panel
920       STATUS @Fg;RETURN ("HEIGHT":Nw_height)
930       ASSIGN @Edit_demo TO WIDGET "LABEL";SET ("VALUE":"Edit
Colors","X":Demo_offset,"Y":Yoffset,"COLUMNS":15,"HEIGHT":2*Nw_height,"PEN":Fg,"BACKGR
OUND":Bg),PARENT @Panel
940       Yoffset=Yoffset+Nw_height
950       ASSIGN @Bg_label TO WIDGET "LABEL";SET ("VALUE":"Edit
Background","JUSTIFICATION":"RIGHT","X":10,"Y":Yoffset,"COLUMNS":15,"BORDER":0),PARE
NT @Panel
960       ASSIGN @Bg TO WIDGET "NUMBER";SET
("X":Xoffset,"Y":Yoffset,"COLUMNS":4,"FORMAT":"SHORT
INTEGER","MINIMUM":0,"MAXIMUM":Max_pen,"VALUE":Bg),PARENT @Panel
970       !
980       ! List colors
990       !
1000      Yoffset=Yoffset+Nw_height
1010      ASSIGN @Lb_fg_label TO WIDGET "LABEL";SET ("VALUE":"List
Pen","JUSTIFICATION":"RIGHT","X":10,"Y":Yoffset,"COLUMNS":15,"BORDER":0),PARENT
@Panel
1020      ASSIGN @Lb_fg TO WIDGET "NUMBER";SET
("X":Xoffset,"Y":Yoffset,"COLUMNS":4,"FORMAT":"SHORT
INTEGER","MINIMUM":0,"MAXIMUM":Max_pen,"VALUE":Lb_fg),PARENT @Panel
1030      ASSIGN @List_demo TO WIDGET "LABEL";SET ("VALUE":"List
Colors","X":Demo_offset,"Y":Yoffset,"COLUMNS":15,"HEIGHT":2*Nw_height,"PEN":Fg,"BACKGR
OUND":Lb_bg),PARENT @Panel
1040      Yoffset=Yoffset+Nw_height
1050      ASSIGN @Lb_bg_label TO WIDGET "LABEL";SET ("VALUE":"List
Background","JUSTIFICATION":"RIGHT","X":10,"Y":Yoffset,"COLUMNS":15,"BORDER":0),PARE
NT @Panel
1060      ASSIGN @Lb_bg TO WIDGET "NUMBER";SET
("X":Xoffset,"Y":Yoffset,"COLUMNS":4,"FORMAT":"SHORT
INTEGER","MINIMUM":0,"MAXIMUM":Max_pen,"VALUE":Lb_bg),PARENT @Panel
```

```
1070      Yoffset=Yoffset+Nw_height+10
1080      STATUS @List_demo;RETURN ("WIDTH":W)
1090      STATUS @Panel;RETURN ("WIDTH":Pw,"HEIGHT":Ph,"INSIDE WIDTH":Piw,"INSIDE
HEIGHT":Pih)
1100      Pw=Pw-Piw
1110      Piw=Demo_offset+W+10
1120      Pw=Pw+Piw
1130      ASSIGN @Sep TO WIDGET "SEPARATOR";SET
("X":10,"Y":Yoffset,"WIDTH":Piw-20,"HEIGHT":5),PARENT @Panel
1140      Yoffset=Yoffset+10
1150      ASSIGN @Done TO WIDGET "PUSHBUTTON";SET
("X":10+5*CHRX,"Y":Yoffset,"COLUMNS":10,"LABEL":"DONE"),PARENT @Panel
1160      ASSIGN @Cancel TO WIDGET "PUSHBUTTON";SET
("X":Demo_offset,"Y":Yoffset,"COLUMNS":10,"LABEL":"CANCEL"),PARENT @Panel
1170      STATUS @Cancel;RETURN ("HEIGHT":H)
1180      Ph=Ph-Pih
1190      Pih=Yoffset+H+10
1200      Ph=Ph+Pih
1210      CONTROL @Panel;SET ("WIDTH":Pw,"HEIGHT":Ph,"VISIBLE":1)
1220      !
1230      ON EVENT @Fg,"RETURN",2 GOSUB Set_edit
1240      ON EVENT @Fg,"DONE",2 GOSUB Set_edit
1250      ON EVENT @Bg,"RETURN",2 GOSUB Set_edit
1260      ON EVENT @Bg,"DONE",2 GOSUB Set_edit
1270      ON EVENT @Lb_fg,"RETURN",2 GOSUB Set_list
1280      ON EVENT @Lb_fg,"DONE",2 GOSUB Set_list
1290      ON EVENT @Lb_bg,"DONE",2 GOSUB Set_list
1300      ON EVENT @Done,"ACTIVATED",2 GOTO Done
1310      ON EVENT @Cancel,"ACTIVATED",2 GOTO Cancel
1320      !
1330      LOOP
1340        WAIT FOR EVENT
1350      END LOOP
1360 Set_edit:!
1370      STATUS @Fg;RETURN ("VALUE":Fg)
1380      STATUS @Bg;RETURN ("VALUE":Bg)
1390      CONTROL @Edit_demo;SET ("PEN":Fg,"BACKGROUND":Bg)
1400      RETURN
1410 !
1420 Set_list:!
1430      STATUS @Lb_fg;RETURN ("VALUE":Lb_fg)
1440      STATUS @Lb_bg;RETURN ("VALUE":Lb_bg)
1450      CONTROL @List_demo;SET ("PEN":Lb_fg,"BACKGROUND":Lb_bg)
```

```
1460       RETURN
1470 !
1480 Done: RETURN 1
1490 Cancel: RETURN 0
1500    FNEND
```

```
10      ! ************************************************************
20      ! Example: Calculator
30      !
40      ! This program creates a calculator using PUSHBUTTON widgets.
50      !
60      ! ************************************************************
70      !
80      COM /Stack_com/ INTEGER Max_stack,Stack_ptr,REAL Stack(100)
90      ASSIGN @Calc TO WIDGET "PANEL";SET ("TITLE":" Example:
Calculator","X":20,"Y":20,"WIDTH":300,"HEIGHT":400)
100     STATUS @Calc;RETURN ("INSIDE WIDTH":Iw)
110     CONTROL @Calc;SET ("SYSTEM MENU":"Quit")
120     ON EVENT @Calc,"SYSTEM MENU" GOTO Finis
130     ASSIGN @Display TO WIDGET "LABEL";SET
("X":5,"Y":5,"WIDTH":Iw-10,"HEIGHT":40,"BACKGROUND":0,"PEN":2,"VALUE":0,"JUSTIFICATI
ON":"RIGHT"),PARENT @Calc
140     Xpos=5
150     Ypos=80
160     Item=1
170     RAD
180     ON EVENT @Calc,"RESIZED" GOSUB Resized
190     DIM Val$[64],Tmp$[64]
200     DIM Modes$(1:2)[80]
210     REAL Mem(10)
220     Modes$(1)="DEGREES"
230     Modes$(2)="RADIANS"
240     Val$=""
250     Fkey=0
260     Max_stack=100
270     Stack_ptr=0
280     Base=10
290     Store_mode=0
300     Recall_mode=0
310     !
320     ASSIGN @Button10 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+0*(Iw-10)/5,"Y":Ypos+0*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
330     ASSIGN @Label10 TO WIDGET "LABEL";SET ("X":Xpos+0*(Iw-10)/5,"Y":Ypos+0*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
340     CONTROL @Button10;SET ("BACKGROUND":6,"PEN":1,"LABEL":"7","VISIBLE":1)
350     CONTROL @Label10;SET ("PEN":0,"VALUE":"A")
360     ASSIGN @Button11 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+0*(Iw-10)/5,"Y":Ypos+1*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
```

```
370    ASSIGN @Label11 TO WIDGET "LABEL";SET ("X":Xpos+0*(Iw-10)/5,"Y":Ypos+1*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
380    CONTROL @Button11;SET ("BACKGROUND":6,"PEN":1,"LABEL":"4","VISIBLE":1)
390    CONTROL @Label11;SET ("PEN":0,"VALUE":"D")
400    ASSIGN @Button12 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+0*(Iw-10)/5,"Y":Ypos+2*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
410    ASSIGN @Label12 TO WIDGET "LABEL";SET ("X":Xpos+0*(Iw-10)/5,"Y":Ypos+2*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
420    CONTROL @Button12;SET ("BACKGROUND":6,"PEN":1,"LABEL":"1","VISIBLE":1)
430    CONTROL @Label12;SET ("PEN":0,"VALUE":"","VISIBLE":1)
440    ASSIGN @Button13 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+0*(Iw-10)/5,"Y":Ypos+3*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
450    ASSIGN @Label13 TO WIDGET "LABEL";SET ("X":Xpos+0*(Iw-10)/5,"Y":Ypos+3*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
460    CONTROL @Button13;SET ("BACKGROUND":6,"PEN":1,"LABEL":"0","VISIBLE":1)
470    CONTROL @Label13;SET ("PEN":0,"VALUE":"STO","VISIBLE":1)
480    ASSIGN @Button14 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+0*(Iw-10)/5,"Y":Ypos+4*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
490    ASSIGN @Label14 TO WIDGET "LABEL";SET ("X":Xpos+0*(Iw-10)/5,"Y":Ypos+4*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
500    CONTROL @Button14;SET ("WIDTH":(Iw-20)/2.5-10)
510    CONTROL @Label14;SET ("WIDTH":(Iw-20)/2.5-10)
520    CONTROL @Button14;SET ("BACKGROUND":6,"PEN":1,"LABEL":"ENTER","VISIBLE":1)
530    CONTROL @Label14;SET ("PEN":0,"VALUE":"QUIT","VISIBLE":1)
540    ASSIGN @Button15 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+1*(Iw-10)/5,"Y":Ypos+0*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
550    ASSIGN @Label15 TO WIDGET "LABEL";SET ("X":Xpos+1*(Iw-10)/5,"Y":Ypos+0*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
560    CONTROL @Button15;SET ("BACKGROUND":6,"PEN":1,"LABEL":"8","VISIBLE":1)
570    CONTROL @Label15;SET ("PEN":0,"VALUE":"B")
580    ASSIGN @Button16 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+1*(Iw-10)/5,"Y":Ypos+1*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
590    ASSIGN @Label16 TO WIDGET "LABEL";SET ("X":Xpos+1*(Iw-10)/5,"Y":Ypos+1*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
600    CONTROL @Button16;SET ("BACKGROUND":6,"PEN":1,"LABEL":"5","VISIBLE":1)
610    CONTROL @Label16;SET ("PEN":0,"VALUE":"E")
620    ASSIGN @Button17 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+1*(Iw-10)/5,"Y":Ypos+2*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
630    ASSIGN @Label17 TO WIDGET "LABEL";SET ("X":Xpos+1*(Iw-10)/5,"Y":Ypos+2*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
640    CONTROL @Button17;SET ("BACKGROUND":6,"PEN":1,"LABEL":"2","VISIBLE":1)
```

```
650     CONTROL @Label17;SET ("PEN":0,"VALUE":"e","VISIBLE":1)
660     ASSIGN @Button18 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+1*(Iw-10)/5,"Y":Ypos+3*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
670     ASSIGN @Label18 TO WIDGET "LABEL";SET ("X":Xpos+1*(Iw-10)/5,"Y":Ypos+3*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
680     CONTROL @Button18;SET ("BACKGROUND":6,"PEN":1,"LABEL":".","VISIBLE":1)
690     CONTROL @Label18;SET ("PEN":0,"VALUE":"RCL","VISIBLE":1)
700     ASSIGN @Button20 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+2*(Iw-10)/5,"Y":Ypos+0*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
710     ASSIGN @Label20 TO WIDGET "LABEL";SET ("X":Xpos+2*(Iw-10)/5,"Y":Ypos+0*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
720     CONTROL @Button20;SET ("BACKGROUND":6,"PEN":1,"LABEL":"9","VISIBLE":1)
730     CONTROL @Label20;SET ("PEN":0,"VALUE":"C")
740     ASSIGN @Button21 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+2*(Iw-10)/5,"Y":Ypos+1*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
750     ASSIGN @Label21 TO WIDGET "LABEL";SET ("X":Xpos+2*(Iw-10)/5,"Y":Ypos+1*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
760     CONTROL @Button21;SET ("BACKGROUND":6,"PEN":1,"LABEL":"6","VISIBLE":1)
770     CONTROL @Label21;SET ("PEN":0,"VALUE":"F")
780     ASSIGN @Button22 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+2*(Iw-10)/5,"Y":Ypos+2*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
790     ASSIGN @Label22 TO WIDGET "LABEL";SET ("X":Xpos+2*(Iw-10)/5,"Y":Ypos+2*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
800     CONTROL @Button22;SET ("BACKGROUND":6,"PEN":1,"LABEL":"3","VISIBLE":1)
810     CONTROL @Label22;SET ("PEN":0,"VALUE":"PI","VISIBLE":1)
820     ASSIGN @Button23 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+2*(Iw-10)/5,"Y":Ypos+3*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
830     ASSIGN @Label23 TO WIDGET "LABEL";SET ("X":Xpos+2*(Iw-10)/5,"Y":Ypos+3*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
840     CONTROL @Button23;SET ("BACKGROUND":6,"PEN":1,"LABEL":"CHS","VISIBLE":1)
850     CONTROL @Label23;SET ("PEN":0,"VALUE":"ANGL","VISIBLE":1)
860     ASSIGN @Button24 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+2*(Iw-10)/5,"Y":Ypos+4*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
870     ASSIGN @Label24 TO WIDGET "LABEL";SET ("X":Xpos+2*(Iw-10)/5,"Y":Ypos+4*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
880     CONTROL @Button24;SET ("BACKGROUND":6,"PEN":1,"LABEL":"CLX","VISIBLE":1)
890     CONTROL @Label24;SET ("PEN":0,"VALUE":"","VISIBLE":1)
900     ASSIGN @Button25 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+3*(Iw-10)/5,"Y":Ypos+0*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
```

```
910    ASSIGN @Label25 TO WIDGET "LABEL";SET ("X":Xpos+3*(Iw-10)/5,"Y":Ypos+0*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
920    CONTROL @Button25;SET ("BACKGROUND":6,"PEN":1,"LABEL":"/","VISIBLE":1)
930    CONTROL @Label25;SET ("PEN":0,"VALUE":"HEX","VISIBLE":1)
940    ASSIGN @Button26 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+3*(Iw-10)/5,"Y":Ypos+1*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
950    ASSIGN @Label26 TO WIDGET "LABEL";SET ("X":Xpos+3*(Iw-10)/5,"Y":Ypos+1*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
960    CONTROL @Button26;SET ("BACKGROUND":6,"PEN":1,"LABEL":"*","VISIBLE":1)
970    CONTROL @Label26;SET ("PEN":0,"VALUE":"DEC","VISIBLE":1)
980    ASSIGN @Button27 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+3*(Iw-10)/5,"Y":Ypos+2*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
990    ASSIGN @Label27 TO WIDGET "LABEL";SET ("X":Xpos+3*(Iw-10)/5,"Y":Ypos+2*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
1000   CONTROL @Button27;SET ("BACKGROUND":6,"PEN":1,"LABEL":"-","VISIBLE":1)
1010   CONTROL @Label27;SET ("PEN":0,"VALUE":"OCT","VISIBLE":1)
1020   ASSIGN @Button28 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+3*(Iw-10)/5,"Y":Ypos+3*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
1030   ASSIGN @Label28 TO WIDGET "LABEL";SET ("X":Xpos+3*(Iw-10)/5,"Y":Ypos+3*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
1040   CONTROL @Button28;SET ("BACKGROUND":6,"PEN":1,"LABEL":"+","VISIBLE":1)
1050   CONTROL @Label28;SET ("PEN":0,"VALUE":"BIN","VISIBLE":1)
1060   ASSIGN @Button29 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+3*(Iw-10)/5,"Y":Ypos+4*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
1070   ASSIGN @Label29 TO WIDGET "LABEL";SET ("X":Xpos+3*(Iw-10)/5,"Y":Ypos+4*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
1080   CONTROL @Button29;SET ("BACKGROUND":6,"PEN":1,"LABEL":"f","VISIBLE":1)
1090   CONTROL @Label29;SET ("PEN":0,"VALUE":"","VISIBLE":1)
1100   CONTROL @Button29;SET ("BACKGROUND":3,"PEN":0)
1110   ASSIGN @Button30 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+4*(Iw-10)/5,"Y":Ypos+0*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
1120   ASSIGN @Label30 TO WIDGET "LABEL";SET ("X":Xpos+4*(Iw-10)/5,"Y":Ypos+0*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
1130   CONTROL @Button30;SET ("BACKGROUND":6,"PEN":1,"LABEL":"SIN","VISIBLE":1)
1140   CONTROL @Label30;SET ("PEN":0,"VALUE":"ASN","VISIBLE":1)
1150   ASSIGN @Button31 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+4*(Iw-10)/5,"Y":Ypos+1*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
1160   ASSIGN @Label31 TO WIDGET "LABEL";SET ("X":Xpos+4*(Iw-10)/5,"Y":Ypos+1*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
1170   CONTROL @Button31;SET ("BACKGROUND":6,"PEN":1,"LABEL":"COS","VISIBLE":1)
1180   CONTROL @Label31;SET ("PEN":0,"VALUE":"ACS","VISIBLE":1)
```

```
1190   ASSIGN @Button32 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+4*(Iw-10)/5,"Y":Ypos+2*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
1200   ASSIGN @Label32 TO WIDGET "LABEL";SET ("X":Xpos+4*(Iw-10)/5,"Y":Ypos+2*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
1210   CONTROL @Button32;SET ("BACKGROUND":6,"PEN":1,"LABEL":"TAN","VISIBLE":1)
1220   CONTROL @Label32;SET ("PEN":0,"VALUE":"ATN","VISIBLE":1)
1230   ASSIGN @Button33 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+4*(Iw-10)/5,"Y":Ypos+3*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
1240   ASSIGN @Label33 TO WIDGET "LABEL";SET ("X":Xpos+4*(Iw-10)/5,"Y":Ypos+3*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
1250   CONTROL @Button33;SET ("BACKGROUND":6,"PEN":1,"LABEL":"10^X","VISIBLE":1)
1260   CONTROL @Label33;SET ("PEN":0,"VALUE":"LOG","VISIBLE":1)
1270   ASSIGN @Button34 TO WIDGET "PUSHBUTTON";SET
("X":Xpos+4*(Iw-10)/5,"Y":Ypos+4*(250-20)/4,"WIDTH":(Iw-20)/5-
10,"HEIGHT":30,"VISIBLE":0),PARENT @Calc
1280   ASSIGN @Label34 TO WIDGET "LABEL";SET ("X":Xpos+4*(Iw-10)/5,"Y":Ypos+4*(250-
20)/4-25,"WIDTH":(Iw-20)/5-10,"HEIGHT":25,"BORDER":0,"VISIBLE":0),PARENT @Calc
1290   CONTROL @Button34;SET ("BACKGROUND":6,"PEN":1,"LABEL":"EEX","VISIBLE":1)
1300   CONTROL @Label34;SET ("PEN":0,"VALUE":"LN","VISIBLE":1)
1310   ON ERROR GOTO Process_error
1320   ON EVENT @Button10,"ACTIVATED" GOSUB Button10
1330   CONTROL @Label16;SET ("VISIBLE":1)
1340   ON EVENT @Button11,"ACTIVATED" GOSUB Button11
1350   ON EVENT @Button12,"ACTIVATED" GOSUB Button12
1360   ON EVENT @Button13,"ACTIVATED" GOSUB Button13
1370   ON EVENT @Button14,"ACTIVATED" GOSUB Button14
1380   ON EVENT @Button15,"ACTIVATED" GOSUB Button15
1390   ON EVENT @Button16,"ACTIVATED" GOSUB Button16
1400   ON EVENT @Button17,"ACTIVATED" GOSUB Button17
1410   ON EVENT @Button18,"ACTIVATED" GOSUB Button18
1420   ON EVENT @Button20,"ACTIVATED" GOSUB Button20
1430   ON EVENT @Button21,"ACTIVATED" GOSUB Button21
1440   ON EVENT @Button22,"ACTIVATED" GOSUB Button22
1450   ON EVENT @Button23,"ACTIVATED" GOSUB Button23
1460   ON EVENT @Button24,"ACTIVATED" GOSUB Button24
1470   ON EVENT @Button25,"ACTIVATED" GOSUB Button25
1480   ON EVENT @Button26,"ACTIVATED" GOSUB Button26
1490   ON EVENT @Button27,"ACTIVATED" GOSUB Button27
1500   ON EVENT @Button28,"ACTIVATED" GOSUB Button28
1510   ON EVENT @Button29,"ACTIVATED" GOSUB Button29
1520   ON EVENT @Button30,"ACTIVATED" GOSUB Button30
1530   ON EVENT @Button31,"ACTIVATED" GOSUB Button31
```

```
1540   ON EVENT @Button32,"ACTIVATED" GOSUB Button32
1550   ON EVENT @Button33,"ACTIVATED" GOSUB Button33
1560   ON EVENT @Button34,"ACTIVATED" GOSUB Button34
1570   LOOP
1580     WAIT FOR EVENT
1590   END LOOP
1600 Button10:       !
1610   Number=7
1620   GOSUB Check_storcl
1630   IF Store_mode=0 AND Recall_mode=0 THEN
1640     GOSUB Getval
1650     IF Fkey=0 OR Base<>16 THEN
1660       Modval(Val$,Number)
1670     ELSE
1680       Modval(Val$,0,"A")
1690     END IF
1700     CONTROL @Display;SET ("VALUE":Val$)
1710   ELSE
1720     GOSUB Enable_base
1730   END IF
1740   GOSUB Check_fkey
1750   RETURN
1760 Button11:       !
1770   Number=4
1780   GOSUB Check_storcl
1790   IF Store_mode=0 AND Recall_mode=0 THEN
1800     GOSUB Getval
1810     IF Fkey=0 OR Base<>16 THEN
1820       Modval(Val$,Number)
1830     ELSE
1840       Modval(Val$,0,"D")
1850     END IF
1860     CONTROL @Display;SET ("VALUE":Val$)
1870   ELSE
1880     GOSUB Enable_base
1890   END IF
1900   GOSUB Check_fkey
1910   RETURN
1920 Button12:       !
1930   Number=1
1940   GOSUB Check_storcl
1950   IF Store_mode=0 AND Recall_mode=0 THEN
```

```
1960      GOSUB Getval
1970      Modval(Val$,Number)
1980      CONTROL @Display;SET ("VALUE":Val$)
1990    ELSE
2000      GOSUB Enable_base
2010    END IF
2020    GOSUB Check_fkey
2030    RETURN
2040 Button13:        !
2050    Number=0
2060    GOSUB Check_storcl
2070    IF Store_mode=0 AND Recall_mode=0 THEN
2080      GOSUB Getval
2090      IF Fkey=0 THEN
2100        Modval(Val$,Number)
2110      ELSE
2120        Store_mode=1
2130        GOSUB Enable_all
2140        GOSUB Clear_fn_keys
2150        GOSUB Clear_hex_keys
2160        RETURN
2170      END IF
2180      CONTROL @Display;SET ("VALUE":Val$)
2190    ELSE
2200      GOSUB Enable_base
2210    END IF
2220    GOSUB Check_fkey
2230    RETURN
2240 Button14:        !
2250    IF Fkey=0 THEN
2260      Get_dispval(@Display,Base,Val)
2270      Push_val(Val,@Display)
2280      Restart=1
2290    ELSE
2300      STOP
2310    END IF
2320    GOSUB Check_fkey
2330    RETURN
2340 Button15:        !
2350    Number=8
2360    GOSUB Check_storcl
2370    IF Store_mode=0 AND Recall_mode=0 THEN
```

```
2380     GOSUB Getval
2390     IF Fkey=0 OR Base<>16 THEN
2400       Modval(Val$,Number)
2410     ELSE
2420       Modval(Val$,0,"B")
2430     END IF
2440     CONTROL @Display;SET ("VALUE":Val$)
2450   ELSE
2460     GOSUB Enable_base
2470   END IF
2480   GOSUB Check_fkey
2490   RETURN
2500 Button16:       !
2510   Number=5
2520   GOSUB Check_storcl
2530   IF Store_mode=0 AND Recall_mode=0 THEN
2540     GOSUB Getval
2550     IF Fkey=0 OR (Base<>16 AND Base<>10) THEN
2560       Modval(Val$,Number)
2570     ELSE
2580       IF Base=10 THEN
2590         GOSUB Process_e
2600       ELSE
2610         Modval(Val$,0,"E")
2620       END IF
2630     END IF
2640     CONTROL @Display;SET ("VALUE":Val$)
2650   ELSE
2660     GOSUB Enable_base
2670   END IF
2680   GOSUB Check_fkey
2690   RETURN
2700 Button17:       !
2710   Number=2
2720   GOSUB Check_storcl
2730   IF Store_mode=0 AND Recall_mode=0 THEN
2740     GOSUB Getval
2750     IF Fkey=0 THEN
2760       Modval(Val$,Number)
2770     ELSE
2780       Tentobase(Base,EXP(1),Val$,Error)
2790     END IF
```

```
2800      CONTROL @Display;SET ("VALUE":Val$)
2810    ELSE
2820      GOSUB Enable_base
2830    END IF
2840    GOSUB Check_fkey
2850    RETURN
2860 Button18:        !
2870    IF Fkey=0 THEN
2880      STATUS @Display;RETURN ("VALUE":Val$)
2890      IF (POS(Val$,".")=0) THEN Val$=Val$&"."
2900      CONTROL @Display;SET ("VALUE":Val$)
2910    ELSE
2920      Recall_mode=1
2930      GOSUB Enable_all
2940      GOSUB Clear_fn_keys
2950      GOSUB Clear_hex_keys
2960    END IF
2970    GOSUB Check_fkey
2980    RETURN
2990 Button19:        !
3000 Button20:        !
3010    Number=9
3020    GOSUB Check_storcl
3030    IF Store_mode=0 AND Recall_mode=0 THEN
3040      GOSUB Getval
3050      IF Fkey=0 OR Base<>16 THEN
3060        Modval(Val$,Number)
3070      ELSE
3080        Modval(Val$,0,"C")
3090      END IF
3100      CONTROL @Display;SET ("VALUE":Val$)
3110    ELSE
3120      GOSUB Enable_base
3130    END IF
3140    GOSUB Check_fkey
3150    RETURN
3160 Button21:        !
3170    Number=6
3180    GOSUB Check_storcl
3190    IF Store_mode=0 AND Recall_mode=0 THEN
3200      GOSUB Getval
3210      IF Fkey=0 OR Base<>16 THEN
```

```
3220        Modval(Val$,Number)
3230      ELSE
3240        Modval(Val$,0,"F")
3250      END IF
3260      CONTROL @Display;SET ("VALUE":Val$)
3270    ELSE
3280      GOSUB Enable_base
3290    END IF
3300    GOSUB Check_fkey
3310    RETURN
3320 Button22:        !
3330    Number=3
3340    GOSUB Check_storcl
3350    IF Store_mode=0 AND Recall_mode=0 THEN
3360      GOSUB Getval
3370      IF Fkey=0 THEN
3380        Modval(Val$,Number)
3390      ELSE
3400        Tentobase(Base,PI,Val$,Error)
3410      END IF
3420      CONTROL @Display;SET ("VALUE":Val$)
3430    ELSE
3440      GOSUB Enable_base
3450    END IF
3460    GOSUB Check_fkey
3470    RETURN
3480 Button23:        !
3490    IF Fkey=0 THEN
3500      STATUS @Display;RETURN ("VALUE":Val$)
3510      IF Base=10 THEN
3520        IF Val$[1;1]="-" THEN
3530          Val$=Val$[2]
3540        ELSE
3550          Val$="-"&Val$
3560        END IF
3570      ELSE
3580        Twos_comp(Base,Val$)
3590      END IF
3600      CONTROL @Display;SET ("VALUE":Val$)
3610    ELSE
3620      !* DIALOG "LIST","Trigonometric Mode",Button;SET ("ITEMS":Modes$(*),"SELECTED
ITEM":Item),
```

```
3630     IF Button=0 AND Item<>-1 THEN
3640       IF Item=0 THEN DEG
3650       IF Item=1 THEN RAD
3660     END IF
3670   END IF
3680   GOSUB Check_fkey
3690   RETURN
3700 Button24:         !
3710   CONTROL @Display;SET ("VALUE":0)
3720   GOSUB Check_fkey
3730   RETURN
3740 Button25:         !
3750   IF Fkey=0 THEN
3760     Get_dispval(@Display,Base,Val1)
3770     Val2=FNPop_val(@Display)
3780     Val$=VAL$(Val2/Val1)
3790     Set_dispval(@Display,Base,VAL(Val$))
3800     Push_val(VAL(Val$),@Display)
3810     Restart=1
3820   ELSE
3830     Get_dispval(@Display,Base,Val1)
3840     Base=16
3850     Set_dispval(@Display,Base,Val1)
3860     CONTROL @Label10;SET ("VISIBLE":1)
3870     CONTROL @Label11;SET ("VISIBLE":1)
3880     CONTROL @Label15;SET ("VISIBLE":1)
3890     CONTROL @Label16;SET ("VISIBLE":1)
3900     CONTROL @Label20;SET ("VISIBLE":1)
3910     CONTROL @Label21;SET ("VISIBLE":1)
3920     GOSUB Enable_all
3930   END IF
3940   GOSUB Check_fkey
3950   RETURN
3960 Button26:         !
3970   IF Fkey=0 THEN
3980     Get_dispval(@Display,Base,Val1)
3990     Val2=FNPop_val(@Display)
4000     Val$=VAL$(Val2*Val1)
4010     Set_dispval(@Display,Base,VAL(Val$))
4020     Push_val(VAL(Val$),@Display)
4030     Restart=1
4040   ELSE
```

```
4050      Get_dispval(@Display,Base,Val1)
4060      Base=10
4070      Set_dispval(@Display,Base,Val1)
4080      GOSUB Enable_all
4090      GOSUB Clear_hex_keys
4100      CONTROL @Label16;SET ("VISIBLE":1)
4110    END IF
4120    GOSUB Check_fkey
4130    RETURN
4140 Button27:          !
4150    IF Fkey=0 THEN
4160      Get_dispval(@Display,Base,Val1)
4170      Val2=FNPop_val(@Display)
4180      Val$=VAL$(Val2-Val1)
4190      Set_dispval(@Display,Base,VAL(Val$))
4200      Push_val(VAL(Val$),@Display)
4210      Restart=1
4220    ELSE
4230      Get_dispval(@Display,Base,Val1)
4240      Base=8
4250      Set_dispval(@Display,Base,Val1)
4260      GOSUB Enable_all
4270      GOSUB Clear_nonoct
4280    END IF
4290    GOSUB Check_fkey
4300    RETURN
4310 Button28:          !
4320    IF Fkey=0 THEN
4330      Get_dispval(@Display,Base,Val1)
4340      Val2=FNPop_val(@Display)
4350      Val$=VAL$(Val1+Val2)
4360      Push_val(VAL(Val$),@Display)
4370      Set_dispval(@Display,Base,VAL(Val$))
4380      Restart=1
4390    ELSE
4400      Get_dispval(@Display,Base,Val1)
4410      Base=2
4420      Set_dispval(@Display,Base,Val1)
4430      GOSUB Enable_all
4440      GOSUB Clear_nonbin
4450    END IF
4460    GOSUB Check_fkey
```

```
4470   RETURN
4480 Button29:         !
4490   Fkey=(Fkey+1) MOD 2
4500   IF Fkey=1 THEN
4510     CONTROL @Display;SET ("PEN":3)
4520     GOSUB Enable_pi
4530   ELSE
4540     CONTROL @Display;SET ("PEN":2)
4550     GOSUB Enable_base
4560   END IF
4570   RETURN
4580 Button30:         !
4590   Get_dispval(@Display,Base,Val1)
4600   IF Fkey=0 THEN
4610     Val$=VAL$(SIN(Val1))
4620   ELSE
4630     Val$=VAL$(ASN(Val1))
4640   END IF
4650   Set_dispval(@Display,Base,VAL(Val$))
4660   Push_val(VAL(Val$),@Display)
4670   Restart=1
4680   GOSUB Check_fkey
4690   RETURN
4700 Button31:         !
4710   Get_dispval(@Display,Base,Val1)
4720   IF Fkey=0 THEN
4730     Val$=VAL$(COS(Val1))
4740   ELSE
4750     Val$=VAL$(ACS(Val1))
4760   END IF
4770   Set_dispval(@Display,Base,VAL(Val$))
4780   Push_val(VAL(Val$),@Display)
4790   Restart=1
4800   GOSUB Check_fkey
4810   RETURN
4820 Button32:         !
4830   Get_dispval(@Display,Base,Val1)
4840   IF Fkey=0 THEN
4850     Val$=VAL$(TAN(Val1))
4860   ELSE
4870     Val$=VAL$(ATN(Val1))
4880   END IF
```

```
4890   Set_dispval(@Display,Base,VAL(Val$))
4900   Push_val(VAL(Val$),@Display)
4910   Restart=1
4920   GOSUB Check_fkey
4930   RETURN
4940 Button33:         !
4950   Get_dispval(@Display,Base,Val1)
4960   IF Fkey=0 THEN
4970     Val$=VAL$(10^Val1)
4980   ELSE
4990     Val$=VAL$(LGT(Val1))
5000   END IF
5010   Set_dispval(@Display,Base,VAL(Val$))
5020   Push_val(VAL(Val$),@Display)
5030   Restart=1
5040   GOSUB Check_fkey
5050   RETURN
5060 Button34:         !
5070   Get_dispval(@Display,Base,Val1)
5080   IF Fkey=0 THEN
5090     Val$=VAL$(EXP(Val1))
5100   ELSE
5110     Val$=VAL$(LOG(Val1))
5120   END IF
5130   Set_dispval(@Display,Base,VAL(Val$))
5140   Push_val(VAL(Val$),@Display)
5150   Restart=1
5160   GOSUB Check_fkey
5170   RETURN
5180 Clear_nonbin:   !
5190   CONTROL @Button17;SET ("SENSITIVE":0)
5200   CONTROL @Button22;SET ("SENSITIVE":0)
5210   CONTROL @Button11;SET ("SENSITIVE":0)
5220   CONTROL @Button16;SET ("SENSITIVE":0)
5230   CONTROL @Button21;SET ("SENSITIVE":0)
5240   CONTROL @Button10;SET ("SENSITIVE":0)
5250 Clear_nonoct:   !
5260   CONTROL @Button15;SET ("SENSITIVE":0)
5270   CONTROL @Button20;SET ("SENSITIVE":0)
5280 Clear_hex_keys:   !
5290   CONTROL @Label10;SET ("VISIBLE":0)
5300   CONTROL @Label11;SET ("VISIBLE":0)
```

```
5310   CONTROL @Label15;SET ("VISIBLE":0)
5320   CONTROL @Label16;SET ("VISIBLE":0)
5330   CONTROL @Label20;SET ("VISIBLE":0)
5340   CONTROL @Label21;SET ("VISIBLE":0)
5350   RETURN
5360 Clear_fn_keys:   !
5370   CONTROL @Button29;SET ("SENSITIVE":0)
5380   CONTROL @Button23;SET ("SENSITIVE":0)
5390   CONTROL @Button30;SET ("SENSITIVE":0)
5400   CONTROL @Button32;SET ("SENSITIVE":0)
5410   CONTROL @Button31;SET ("SENSITIVE":0)
5420   CONTROL @Button24;SET ("SENSITIVE":0)
5430   CONTROL @Button34;SET ("SENSITIVE":0)
5440   CONTROL @Button33;SET ("SENSITIVE":0)
5450   CONTROL @Button26;SET ("SENSITIVE":0)
5460   CONTROL @Button14;SET ("SENSITIVE":0)
5470   CONTROL @Button28;SET ("SENSITIVE":0)
5480   CONTROL @Button27;SET ("SENSITIVE":0)
5490   CONTROL @Button18;SET ("SENSITIVE":0)
5500   CONTROL @Button25;SET ("SENSITIVE":0)
5510   RETURN
5520 Enable_all:   !
5530   CONTROL @Button13;SET ("SENSITIVE":1)
5540   CONTROL @Button12;SET ("SENSITIVE":1)
5550   CONTROL @Button17;SET ("SENSITIVE":1)
5560   CONTROL @Button22;SET ("SENSITIVE":1)
5570   CONTROL @Button11;SET ("SENSITIVE":1)
5580   CONTROL @Button16;SET ("SENSITIVE":1)
5590   CONTROL @Button21;SET ("SENSITIVE":1)
5600   CONTROL @Button10;SET ("SENSITIVE":1)
5610   CONTROL @Button15;SET ("SENSITIVE":1)
5620   CONTROL @Button20;SET ("SENSITIVE":1)
5630   CONTROL @Button29;SET ("SENSITIVE":1)
5640   CONTROL @Button23;SET ("SENSITIVE":1)
5650   CONTROL @Button30;SET ("SENSITIVE":1)
5660   CONTROL @Button32;SET ("SENSITIVE":1)
5670   CONTROL @Button31;SET ("SENSITIVE":1)
5680   CONTROL @Button24;SET ("SENSITIVE":1)
5690   CONTROL @Button34;SET ("SENSITIVE":1)
5700   CONTROL @Button33;SET ("SENSITIVE":1)
5710   CONTROL @Button26;SET ("SENSITIVE":1)
5720   CONTROL @Button14;SET ("SENSITIVE":1)
```

```
5730   CONTROL @Button28;SET ("SENSITIVE":1)
5740   CONTROL @Button27;SET ("SENSITIVE":1)
5750   CONTROL @Button18;SET ("SENSITIVE":1)
5760   CONTROL @Button25;SET ("SENSITIVE":1)
5770   RETURN
5780 Enable_e:       !
5790   CONTROL @Label16;SET ("VISIBLE":1)
5800   RETURN
5810 Enable_pi:      !
5820   CONTROL @Button22;SET ("SENSITIVE":1)
5830   CONTROL @Button17;SET ("SENSITIVE":1)
5840   RETURN
5850 Enable_base:    !
5860   Recall_mode=0
5870   Store_mode=0
5880   GOSUB Enable_all
5890   IF Base=2 THEN GOSUB Clear_nonbin
5900   IF Base=8 THEN GOSUB Clear_nonoct
5910   IF Base=10 THEN
5920     GOSUB Clear_hex_keys
5930     GOSUB Enable_e
5940   END IF
5950   RETURN
5960 Check_storcl: !
5970   IF Store_mode THEN
5980     Get_dispval(@Display,Base,Val1)
5990     Mem(Number)=Val1
6000   ELSE
6010     IF Recall_mode THEN
6020       Val1=Mem(Number)
6030       Set_dispval(@Display,Base,Val1)
6040     END IF
6050   END IF
6060   RETURN
6070 Resized: RETURN
6080 Process_e: !
6090   IF (POS(Val$,"E")=0) THEN Val$=Val$&"E"
6100   RETURN
6110 Process_error: !
6120   IF Fkey=1 THEN CONTROL @Display;SET ("PEN":2)
6130   Fkey=0
6140   STATUS @Display;RETURN ("VALUE":Tmp$)
```

```
6150   CONTROL @Display;SET ("VALUE":ERRM$)
6160   WAIT 2
6170   CONTROL @Display;SET ("VALUE":Tmp$)
6180   RETURN
6190 Getval: !
6200   IF Restart THEN
6210     Restart=0
6220     Val$="0"
6230   ELSE
6240     STATUS @Display;RETURN ("VALUE":Val$)
6250   END IF
6260   RETURN
6270 Check_fkey: !
6280   IF Fkey=1 THEN
6290     CONTROL @Display;SET ("PEN":2)
6300     GOSUB Enable_base
6310   END IF
6320   Fkey=0
6330   RETURN
6340 Finis: END
6350   !
6360   SUB Push_val(REAL Val,@Disp)
6370     COM /Stack_com/ INTEGER Max_stack,Stack_ptr,REAL Stack(*)
6380     Stack(Stack_ptr)=Val
6390     IF Stack_ptr<=Max_stack THEN
6400       Stack_ptr=Stack_ptr+1
6410     ELSE
6420       BEEP
6430       STATUS @Disp;RETURN ("VALUE":Tmpval$)
6440       CONTROL @Disp;SET ("VALUE":"ERROR: STACK OVERFLOW!!!")
6450       WAIT 2
6460       CONTROL @Disp;SET ("VALUE":Tmpval$)
6470       SUBEXIT
6480     END IF
6490   SUBEND
6500   DEF FNPop_val(@Disp)
6510     COM /Stack_com/ INTEGER Max_stack,Stack_ptr,REAL Stack(*)
6520     IF Stack_ptr>0 THEN
6530       Stack_ptr=Stack_ptr-1
6540     ELSE
6550       BEEP
6560       STATUS @Disp;RETURN ("VALUE":Tmpval$)
```

```
6570        CONTROL @Disp;SET ("VALUE":"ERROR: STACK UNDERFLOW!!!")
6580        WAIT 2
6590        CONTROL @Disp;SET ("VALUE":Tmpval$)
6600        RETURN 0
6610      END IF
6620      RETURN Stack(Stack_ptr)
6630    FNEND
6640    DEF FNGetval(A$)
6650      IF NUM(A$)>=NUM("0") AND NUM(A$)<=NUM("9") THEN
6660        RETURN VAL(A$)
6670      ELSE
6680        RETURN NUM(A$)-55
6690      END IF
6700    FNEND
6710    DEF FNGetval$(A)
6720      IF A>=10 THEN
6730        RETURN CHR$(A+NUM("A")-10)
6740      ELSE
6750        RETURN VAL$(A)
6760      END IF
6770    FNEND
6780    SUB Basetoten(Base,Value,Str$)
6790      IF Base=10 THEN
6800        Value=VAL(Str$)
6810        SUBEXIT
6820      END IF
6830      Sign=0
6840      IF LEN(Str$)=MAXLEN(Str$) AND FNGetval$(Base-1)=Str$[1;1] THEN
6850        Sign=1
6860        Twos_comp(Base,Str$)
6870      END IF
6880      Value=0.
6890      Max=LEN(Str$)
6900      IF POS(Str$,".")<>0 THEN Max=POS(Str$,".")-1
6910      FOR I=1 TO Max
6920        Value=Base*Value+FNGetval(Str$[I;1])
6930      NEXT I
6940      IF POS(Str$,".")<>0 THEN
6950        J=0
6960        FOR I=Max+2 TO LEN(Str$)
6970          J=J+1
6980          Value=Value+FNGetval(Str$[I;1])*Base^(-J)
```

```
6990        NEXT I
7000      END IF
7010      IF Sign=1 THEN
7020        Value=-Value
7030        Twos_comp(Base,Str$)
7040      END IF
7050    SUBEND
7060    SUB Tentobase(Base,Value,Str$,Error)
7070      Error=0
7080      Str$=""
7090      IF Base=10 THEN
7100        Str$=VAL$(Value)
7110        SUBEXIT
7120      END IF
7130      Sign=0
7140      IF Value<0 THEN
7150        Sign=1
7160        Value=-Value
7170      END IF
7180      Divisor1=Value DIV 1
7190      I=1
7200      WHILE Divisor1<>0
7210        IF I>=MAXLEN(Str$) THEN GOTO Length_err
7220        Str$[I;1]=FNGetval$(Divisor1 MOD Base)
7230        I=I+1
7240        Divisor1=Divisor1 DIV Base
7250      END WHILE
7260      Str$=REV$(Str$)
7270      Divisor2=(Value-Value DIV 1)
7280      I=1
7290      IF Divisor2<>0 THEN
7300        Str2$[I;1]="."
7310        I=I+1
7320      END IF
7330      WHILE Divisor2<>0
7340        Divisor2=Divisor2*Base
7350        Str2$[I;1]=FNGetval$(Divisor2 DIV 1)
7360        Divisor2=Divisor2-Divisor2 DIV 1
7370        I=I+1
7380        IF I>MAXLEN(Str2$) THEN Divisor2=0
7390      END WHILE
7400      IF LEN(Str$)+LEN(Str2$)>MAXLEN(Str$) THEN GOTO Length_err
```

```
7410      Str$=Str$&Str2$
7420      IF Sign=1 THEN
7430        Value=-Value
7440        Twos_comp(Base,Str$)
7450      END IF
7460      IF Value=0 THEN Str$="0"
7470      SUBEXIT
7480 Length_err: !
7490      Error=1
7500      IF Sign=1 THEN Value=-Value
7510    SUBEND
7520    SUB Get_dispval(@Disp,Src_base,Value)
7530      DIM Str$[64]
7540      STATUS @Disp;RETURN ("VALUE":Str$)
7550      IF Dest_base<>10 THEN
7560        Basetoten(Src_base,Value,Str$)
7570      ELSE
7580        Value=VAL(Str$)
7590      END IF
7600    SUBEND
7610    SUB Set_dispval(@Disp,Dest_base,Value)
7620      DIM Value$[64]
7630      IF Dest_base<>10 THEN
7640        Tentobase(Dest_base,Value,Value$,Error)
7650      ELSE
7660        Value$=VAL$(Value)
7670      END IF
7680      IF Error=0 THEN
7690        CONTROL @Disp;SET ("VALUE":Value$)
7700      ELSE
7710        CONTROL @Disp;SET ("VALUE":"ERROR: Value out of range.")
7720        WAIT 3
7730        CONTROL @Disp;SET ("VALUE":0)
7740      END IF
7750    SUBEND
7760    SUB Twos_comp(Base,Str$)
7770      DIM Tmpstr$[64]
7780      Tmpstr$=RPT$(FNGetval$(Base-1),64)
7790      FOR I=1 TO LEN(Str$)
7800        IF Str$[I;1]<>"." THEN
7810          Tmpstr$[MAXLEN(Str$)-LEN(Str$)+I]=FNGetval$(Base-FNGetval(Str$[I;1])-1)
7820        ELSE
```

```
7830          Tmpstr$[MAXLEN(Str$)-LEN(Str$)+I]=Str$[I;1]
7840        END IF
7850      NEXT I
7860      Carry=1
7870      Str$=""
7880      FOR I=MAXLEN(Tmpstr$) TO 1 STEP -1
7890        IF Tmpstr$[I;1]<>"." THEN
7900          Num=FNGetval(Tmpstr$[I;1])
7910          Nextnum=(Num+Carry) MOD Base
7920          Str$=FNGetval$(Nextnum)&Str$
7930          Carry=(Num+Carry) DIV Base
7940          IF Carry=0 THEN GOTO Bail_out
7950        ELSE
7960          Str$="."&Str$
7970        END IF
7980      NEXT I
7990      SUBEXIT
8000 Bail_out: Str$=Tmpstr$[1,I-1]&Str$
8010    SUBEND
8020    SUB Modval(Val$,Value,OPTIONAL Str$)
8030      IF NPAR>2 THEN
8040        IF Val$="0" THEN Val$=""
8050        Val$=Val$&Str$
8060      ELSE
8070        IF Val$="0" THEN Val$=""
8080        Val$=Val$&VAL$(Value)
8090      END IF
8100    SUBEND
```

```
10      ! *******************************************************
20      ! Example: Changing the Font
30      !
40      ! This program shows how to change the font on displayed
50      ! text. The numbers shown in (), such as 6 BY 12, show
60      ! the number of pixels wide X number of pixels high.
70      !
80      ! *******************************************************
90      !
100     DIM Font$[100],Text$[100]
110     !
120     DATA   "6 BY 12",            "Example Text (6 BY 12)"
130     DATA   "6 BY 12,BOLD",     "Example Text (6 BY 12, BOLD)"
140     DATA   "8 BY 16",            "Example Text (8 BY 16)"
150     DATA   "8 BY 16,BOLD",     "Example Text (8 BY 16, BOLD)"
160     DATA   "9 BY 15",            "Example Text (9 BY 15)"
170     DATA   "9 BY 15,BOLD",     "Example Text (9 BY 15, BOLD)"
180     DATA   "10 BY 20",           "Example Text (10 BY 20)"
190     DATA   "10 BY 20,BOLD",    "Example Text (10 BY 20, BOLD)"
200     DATA   "18 BY 30",           "Example Text (18 BY 30)"
210     DATA   "18 BY 30,BOLD",    "Example Text (18 BY 30, BOLD)"
220     DATA   "end","end"
230     !
240     ASSIGN @Disp TO WIDGET "LABEL";SET
("X":50,"Y":25,"WIDTH":500,"HEIGHT":100,"TITLE":" Example: Changing the Font")
250     CONTROL @Disp;SET ("SYSTEM MENU":"Quit")
260     ON EVENT @Disp,"SYSTEM MENU" GOTO Finis
270 Top:   !
280     RESTORE
290     LOOP
300       READ Font$,Text$
310     EXIT IF Font$="end"
320       CONTROL @Disp;SET ("VALUE":Text$,"FONT":Font$)
330       WAIT .75
340     END LOOP
350     GOTO Top
360 Finis: !
370     ASSIGN @Disp TO *                  ! Delete LABEL widget
380     END
```

```
10      ! ********************************************************
20      ! Example: Context-Sensitive Help
30      !
40      ! This program displays three pushbuttons (Fire Phasers,
50      ! Fire Photon Torpedoes, and Stand Down). You can display
60      ! the definitions of these pushbuttons by placing the cursor
70      ! on the desired pushbutton and clicking the RIGHT mouse
80      ! button.
90      !
100     ! ********************************************************
110     !
120     DIM S$[256]                             ! GP string
130     S$="weapons.hlp"                        ! Help file name
140     !
150     ! Create the PANEL widget
160     !
170     ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
180     CONTROL @Main;SET ("X":50,"Y":25,"WIDTH":300,"HEIGHT":175)
190     CONTROL @Main;SET ("TITLE":" Example: Context-Sensitive Help","SYSTEM
MENU":"Quit")
200     CONTROL @Main;SET ("HELP FILE":S$,"HELP TOPIC":"weapons.TOC")
210     !
220     ! Set up buttons
230     !
240     ASSIGN @Phasers TO WIDGET "PUSHBUTTON";PARENT @Main
250     CONTROL @Phasers;SET ("X":45,"Y":15,"WIDTH":200,"HEIGHT":30)
260     CONTROL @Phasers;SET ("LABEL":"Fire Phasers")
270     CONTROL @Phasers;SET ("HELP FILE":S$,"HELP TOPIC":"weapons.phasers")
280     !
290     ASSIGN @Torps TO WIDGET "PUSHBUTTON";PARENT @Main
300     CONTROL @Torps;SET ("X":45,"Y":55,"WIDTH":200,"HEIGHT":30)
310     CONTROL @Torps;SET ("LABEL":"Fire Photon Torpedoes")
320     CONTROL @Torps;SET ("HELP FILE":S$,"HELP TOPIC":"weapons.torp")
330     !
340     ASSIGN @Off TO WIDGET "PUSHBUTTON";PARENT @Main
350     CONTROL @Off;SET ("X":45,"Y":100,"WIDTH":200,"HEIGHT":30)
360     CONTROL @Off;SET ("LABEL":"Stand Down")
370     CONTROL @Off;SET ("HELP FILE":S$,"HELP TOPIC":"weapons.off")
380     !
390     ! Set up events and loop
400     !
```

```
410    ON EVENT @Phasers,"ACTIVATED" GOSUB Phasers
420    ON EVENT @Torps,"ACTIVATED" GOSUB Torps
430    ON EVENT @Off,"ACTIVATED" GOSUB Off
440    ON EVENT @Main,"SYSTEM MENU" GOTO Finis
450    !
460    CONTROL @Main;SET ("VISIBLE":1)
470    !
480    LOOP
490       WAIT FOR EVENT
500    END LOOP
510    STOP
520    !
530    ! ********************* Subroutines Start Here ************************
540    !
550 Phasers: !
560    DIALOG "INFORMATION","Fired phasers!"
570    RETURN
580    !
590 Torps:   !
600    DIALOG "INFORMATION","Fired photon torpedoes!"
610    RETURN
620    !
630 Off:   !
640    DIALOG "INFORMATION","Weapons system standing down."
650    RETURN
660    !
670 Finis:   !
680    ASSIGN @Main TO *                  ! Delete PANEL widget
690    END
```

HTBasic for Windows

Example: Creating a Widget

Introduction

This topic shows how to create METER and SLIDER widgets using the process outlined in Steps to Create Widgets. When this program is run, you can move the SLIDER button with the mouse and the METER widget will indicate the SLIDER value.

To run this program, copy METER/SLIDER Widgets into the HTBasic for Windows Editor (see Copying Example Programs for procedures). If you want to experiment with changing other attributes, see the METER and SLIDER widget descriptions.

Example Steps

There are three main steps in this example:

- Step 1: Create Widgets
- Step 2: Change Widget Attributes
- Step 3: Query Attributes and Add Event-Initiated Branching

Click the step name or press the **>>** bar for descriptions of these three steps.

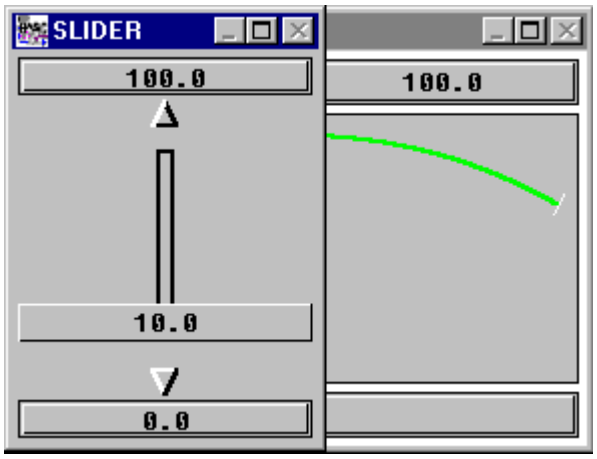
HTBasic for Windows

Example: Creating a Widget - Create METER and SLIDER Widgets

Step 1: Create METER and SLIDER Widgets

The following lines create a SLIDER widget and a METER widget and assign a background color of white to the METER widget. (The SLIDER widget uses default colors.) The LOOP and END LOOP statements are added so that the widgets do not disappear when the program .runs A typical display for this program follows. Since the SLIDER widget was generated last in the program, the SLIDER widget appears on top of the METER widget.

```
10    ASSIGN @Meter TO WIDGET "METER"
20    ASSIGN @Slider TO WIDGET "SLIDER"
30    CONTROL @Meter;SET ("BACKGROUND":1)
40    LOOP
50    END LOOP
60    END
```

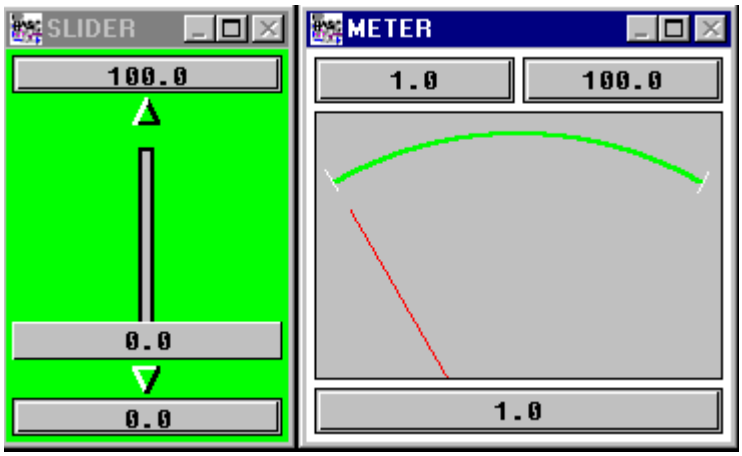


Example: Creating a Widget - Change Widget Attributes

Step 2: Change Widget Attributes

Since SLIDER was the last widget generated, it appears on top of the METER widget. To reposition the SLIDER widget and the METER widget, we will modify the program by adding the positioning attributes "X" and "Y" to each widget so that both widgets are fully visible. In addition, we will change the SLIDER widget background to green.
A typical display for this modified program follows.

```
10    ASSIGN @Meter TO WIDGET "METER"
20    ASSIGN @Slider TO WIDGET "SLIDER"
30    CONTROL @Slider;SET ("X":0,"Y":0,"BACKGROUND":4)
40    CONTROL @Meter;SET ("X":150,"BACKGROUND":1)
50    LOOP
60    END LOOP
70    END
```



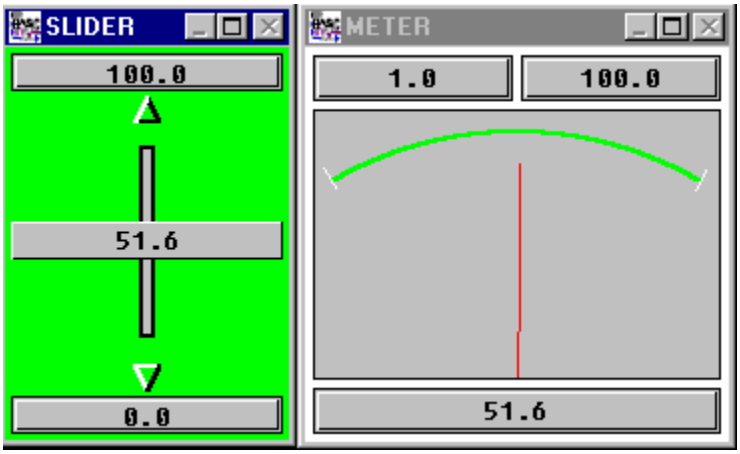
Example: Creating a Widget - Query Attributes and Add Event-Initiated Branching

Step 3: Query Attributes and Add Event-Initiated Branching

Next, we will add ON EVENT and WAIT for EVENT. We will also query the SLIDER widget VALUE attribute value with the STATUS command. A typical display for the modified program follows.

Run this program and select the SLIDER button with the mouse. Move the SLIDER control up and down and notice that the METER will indicate the SLIDER value.

```
10      ASSIGN @Meter TO WIDGET "METER"
20      ASSIGN @Slider TO WIDGET "SLIDER"
30      CONTROL @Slider;SET ("X":0,"Y":0,"BACKGROUND":4)
40      CONTROL @Meter;SET ("X":150,"BACKGROUND":1)
50      ON EVENT @Slider,"CHANGED" GOSUB Event_handler
60      LOOP
70            WAIT FOR EVENT
80      END LOOP
90      !
100 Event_handler: !
110    STATUS @Slider;RETURN ("VALUE":Value)
120    CONTROL @Meter;SET ("VALUE":Value)
130    RETURN
140    END
```



For this program, line 50 calls subroutine Event_handler when the event CHANGED occurs for the SLIDER widget. The CHANGED event is generated every time the user changes the VALUE attribute by clicking on the arrows or in the trough or by clicking

and dragging the slider.

For the Event_handler subroutine, the current VALUE of the SLIDER widget is returned by the STATUS command RETURN option, and is entered into the METER widget VALUE attribute with the CONTROL command SET option. Thus, when you move the SLIDER control up and down, the METER widget will indicate the SLIDER widget value.

Lines 60 through 80 continuously loop the program until the event CHANGED occurs for the SLIDER widget. When WAIT FOR EVENT is included, it allows the computer to do other things while waiting for the event to happen. This is important so as not to waste CPU resources that could be used for other tasks.

```
10      ! ************************************************************
20      ! Example: Cyclical PUSHBUTTON
30      !
40      ! This example implements an object which cycles through its values
50      ! each time it is pushed. It uses a PUSHBUTTON widget. When the button
60      ! is pressed, the "LABEL" and "STATE" change. Any number of choices
70      ! could be used. This type of control, however, works best if the
80      ! number of choices is kept small.
90      !
100     ! ************************************************************
110     !
120     INTEGER Tab_1,Tab_1_cols,Tab_2,Tab_2_cols,Width
130     INTEGER States,State,I
140     DIM Labels$(0:10)[10]
150     !
160     Tab_1=10
170     Tab_1_cols=25
180     Tab_2=Tab_1+(Tab_1_cols+1)*CHRX
190     Tab_2_cols=10
200     Width=Tab_2+(Tab_2_cols+3)*CHRX+Tab_1
210     States=5
220     !
230     ! Create labels for each state
240     !
250     FOR I=BASE(Labels$,1) TO BASE(Labels$,1)+SIZE(Labels$,1)-1
260       Labels$(I)="State "&VAL$(I)
270     NEXT I
280     !
290     ! Create widgets
300     !
310     ASSIGN @Panel TO WIDGET "PANEL";SET
("X":100,"Y":50,"WIDTH":Width,"HEIGHT":10*CHRY,"TITLE":" Example: Cyclical
PUSHBUTTON")
320     CONTROL @Panel;SET ("SYSTEM MENU":"Quit")
330     ON EVENT @Panel,"SYSTEM MENU" GOTO Finis
340     !
350     ASSIGN @Cyclic_label TO WIDGET "LABEL";PARENT @Panel,SET
("X":10,"Y":Tab_1,"COLUMNS":Tab_1_cols,"BORDER":0)
360     CONTROL @Cyclic_label;SET ("VALUE":"Multi-state PUSHBUTTON:
","JUSTIFICATION":"RIGHT")
370     ASSIGN @Cyclic TO WIDGET "PUSHBUTTON";PARENT @Panel,SET
("X":Tab_2,"Y":10,"COLUMNS":Tab_2_cols)
```

```
380     CONTROL @Cyclic;SET ("STATES":States,"LABELS":Labels$(*),"PANEL DEFAULT":1)
390     !
400     ASSIGN @States_label TO WIDGET "LABEL";PARENT @Panel,SET
("X":Tab_1,"Y":3*CHRY,"COLUMNS":Tab_1_cols,"BORDER":0)
410     CONTROL @States_label;SET ("JUSTIFICATION":"RIGHT","VALUE":"Number of States
(1-"&VAL$(SIZE(Labels$,1))&"): ")
420     ASSIGN @States TO WIDGET "NUMBER";PARENT @Panel,SET
("X":Tab_2,"Y":3*CHRY,"COLUMNS":Tab_2_cols)
430     CONTROL @States;SET ("FORMAT":"SHORT
INTEGER","MINIMUM":1,"MAXIMUM":SIZE(Labels$,1),"VALUE":States)
440     !
450     ASSIGN @State_label TO WIDGET "LABEL";PARENT @Panel,SET
("X":Tab_1,"Y":5*CHRY,"COLUMNS":Tab_1_cols,"BORDER":0)
460     CONTROL @State_label;SET ("JUSTIFICATION":"RIGHT","VALUE":"Current State: ")
470     ASSIGN @State TO WIDGET "NUMBER";PARENT @Panel,SET
("X":Tab_2,"Y":5*CHRY,"COLUMNS":Tab_2_cols)
480     CONTROL @State;SET ("FORMAT":"SHORT
INTEGER","MINIMUM":0,"MAXIMUM":States-1,"VALUE":0)
490     !
500     ! Define events
510     !
520     ON EVENT @Cyclic,"ACTIVATED",1 GOSUB Disp_state
530     ON EVENT @States,"RETURN" GOSUB Set_states
540     ON EVENT @States,"DONE" GOSUB Reset_states
550     ON EVENT @State,"RETURN" GOSUB Set_state
560     ON EVENT @State,"DONE" GOSUB Reset_state
570     !
580     LOOP
590       WAIT FOR EVENT
600     END LOOP
610     !
620     ! Event branches
630     !
640 Set_states: !
650     STATUS @States;RETURN ("VALUE":States)
660     CONTROL @State;SET ("MAXIMUM":States-1)
670     CONTROL @Cyclic;SET ("STATES":States)
680     RETURN
690     !
700 Reset_states: !
710     STATUS @Cyclic;RETURN ("STATES":States)
720     CONTROL @States;SET ("VALUE":States)
730     RETURN
740     !
```

```
750 Set_state:    !
760    STATUS @State;RETURN ("VALUE":State)
770    CONTROL @Cyclic;SET ("STATE":State)
780    RETURN
790    !
800 Reset_state: !
810    STATUS @Cyclic;RETURN ("STATE":State)
820    CONTROL @State;SET ("VALUE":State)
830    RETURN
840    !
850 Disp_state: !
860    STATUS @Cyclic;RETURN ("STATE":State)
870    CONTROL @State;SET ("VALUE":State)
880    RETURN
890    !
900 Finis: END
```

```
10    ! ********************************************************************
20    ! Example: Dialogs Tests
30    !
40    ! This program tests HTBasic for Windows dialogs.
50    !
60    ! ********************************************************************
70    !
80    ! Seed random-number generator with current time (manipulating to
90    ! produce a seed that varies rapidly over a wide range, ensuring
100   ! diversity in random-number sequences):
110   !
120   CLEAR SCREEN
130   RANDOMIZE INT(FRACT(TIMEDATE)*10^7)
140   OPTION BASE 0
150   !
160   ! Variables to store PEN colors:
170   !
180   INTEGER Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
190   DATA 0,1,2,3,4,5,6,7
200   READ Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
210   !
220   ! Some other variables:
230   !
240   !    S$:   String variable
250   !    P$:   Stores a prompt
260   !    D$:   Returns directory from FILE dialog
270   !    F$:   Returns file from FILE dialog
280   !    N:    INTEGER variable
290   !    X:    REAL vaiable
300   !    Btn:  Variable to get button inputs from dialogs
310   !    D(*): Array to get display dimensions for BASIC
320   !
330   DIM S$[256],P$[100],M$(0:6)[50]
340   INTEGER N,Btn,D(1:4)
350   REAL X
360   !
370   DATA "COSMOPOLITAN","ENQUIRER","DISCOVER","TIME","HEALTH"
380   DATA "SPORTS ILLUSTRATED","NEW YORKER"
390   READ M$(*)
400   !
410   ! Variables to store widget & display coordinates and dimensions:
```

```
420   !
430   INTEGER Px,Py,Pw,Ph           ! Main PANEL
440   INTEGER Iw,Ih                 ! Inside dimensions of main PANEL
450   INTEGER Dw,Dh                 ! Display dimensions
460   !
470   ! Get display size
480   !
490   GESCAPE CRT,3;D(*)
500   Dw=D(3)-D(1)
510   Dh=D(4)-D(2)
520   !
530   CLEAR SCREEN
540   !
550   ! Set default coordinates for main PANEL
560   !
570   Pw=Dw*.75
580   Ph=Dh*.75
590   Px=(Dw-Pw)/2
600   Py=(Dh-Ph)/2
610   !
620   ! Create the PANEL widget
630   !
640   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
650   CONTROL @Main;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
660   CONTROL @Main;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
670   CONTROL @Main;SET ("TITLE":" Example: Dialogs")
680   !
690   ! Set up menu
700   !
710   S$="Dialog Tests"
720   ASSIGN @Menu TO WIDGET "PULLDOWN MENU";PARENT @Main,SET ("LABEL":S$)
730   !
740   S$="ERROR"
750   ASSIGN @Errtest TO WIDGET "MENU BUTTON";PARENT @Menu,SET ("LABEL":S$)
760   !
770   S$="INFORMATION"
780   ASSIGN @Infotest TO WIDGET "MENU BUTTON";PARENT @Menu,SET ("LABEL":S$)
790   !
800   S$="QUESTION"
810   ASSIGN @Qtest TO WIDGET "MENU BUTTON";PARENT @Menu,SET ("LABEL":S$)
820   !
830   S$="WARNING"
```

```
840   ASSIGN @Warntest TO WIDGET "MENU BUTTON";PARENT @Menu,SET ("LABEL":S$)
850   !
860   ASSIGN @S1 TO WIDGET "MENU SEPARATOR";PARENT @Menu
870   !
880   S$="STRING"
890   ASSIGN @Stringtest TO WIDGET "MENU BUTTON";PARENT @Menu,SET ("LABEL":S$)
900   !
910   S$="NUMBER"
920   ASSIGN @Numtest TO WIDGET "MENU BUTTON";PARENT @Menu,SET ("LABEL":S$)
930   !
940   S$="KEYPAD"
950   ASSIGN @Keytest TO WIDGET "MENU BUTTON";PARENT @Menu,SET ("LABEL":S$)
960   !
970   ASSIGN @S2 TO WIDGET "MENU SEPARATOR";PARENT @Menu
980   !
990   S$="LIST"
1000  ASSIGN @Listtest TO WIDGET "MENU BUTTON";PARENT @Menu,SET ("LABEL":S$)
1010  !
1020  S$="COMBO"
1030  ASSIGN @Combotest TO WIDGET "MENU BUTTON";PARENT @Menu,SET
("LABEL":S$)
1040  !
1050  ASSIGN @S3 TO WIDGET "MENU SEPARATOR";PARENT @Menu
1060  !
1070  S$="FILE"
1080  ASSIGN @Filetest TO WIDGET "MENU BUTTON";PARENT @Menu,SET ("LABEL":S$)
1090  !
1100  ASSIGN @S4 TO WIDGET "MENU SEPARATOR";PARENT @Menu
1110  !
1120  S$="Quit"
1130  ASSIGN @Quit TO WIDGET "MENU BUTTON";PARENT @Menu,SET ("LABEL":S$)
1140  !
1150  ! Build PRINTER widget
1160  !
1170  COM @Prn
1180  ASSIGN @Prn TO WIDGET "PRINTER";PARENT @Main
1190  STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
1200  CONTROL @Prn;SET ("X":0,"Y":0,"WIDTH":Iw,"HEIGHT":Ih)
1210  CONTROL @Prn;SET ("BACKGROUND":Blue,"PEN":White)
1220  !
1230  ! Set up events for menu entries and PANEL resize, then loop.
1240  !
```

```
1250   ON EVENT @Errtest,"ACTIVATED" GOSUB Errtest
1260   ON EVENT @Infotest,"ACTIVATED" GOSUB Infotest
1270   ON EVENT @Qtest,"ACTIVATED" GOSUB Qtest
1280   ON EVENT @Warntest,"ACTIVATED" GOSUB Warntest
1290   ON EVENT @Stringtest,"ACTIVATED" GOSUB Stringtest
1300   ON EVENT @Numtest,"ACTIVATED" GOSUB Numtest
1310   ON EVENT @Keytest,"ACTIVATED" GOSUB Keytest
1320   ON EVENT @Listtest,"ACTIVATED" GOSUB Listtest
1330   ON EVENT @Combotest,"ACTIVATED" GOSUB Combotest
1340   ON EVENT @Filetest,"ACTIVATED" GOSUB Filetest
1350   !
1360   ON EVENT @Quit,"ACTIVATED" GOTO Finis
1370   !
1380   CONTROL @Main;SET ("VISIBLE":1)
1390   CALL Printit("Click Dialog Tests to select a test from the menu.")
1400   !
1410   LOOP
1420     WAIT FOR EVENT
1430   END LOOP
1440   !
1450   ! ******************* End of Main Program **************************
1460   !
1470 Errtest: !
1480   P$="Input caused overflow"
1490   DIALOG "ERROR",P$;SET ("TITLE":" Example: ERROR Dialog")
1500   RETURN
1510   !
1520 Infotest: !
1530   P$="Here is the information required"
1540   DIALOG "INFORMATION",P$;SET ("TITLE":" Example: INFORMATION Dialog")
1550   RETURN
1560   !
1570 Qtest: !
1580   P$="Do you want to exit?"
1590   DIALOG "QUESTION",P$,Btn;SET ("TITLE":" Example: QUESTION Dialog")
1600   SELECT Btn
1610   CASE 0
1620     CALL Printit("QUESTION Dialog: Btn = 0:   YES")
1630   CASE 1
1640     CALL Printit("QUESTION Dialog: Btn = 1:   NO")
1650   END SELECT
1660   RETURN
```

```
1670   !
1680 Warntest: !
1690   P$="Core meltdown in one minute!!"
1700   DIALOG "WARNING",P$;SET ("TITLE":" Example: WARNING Dialog")
1710   RETURN
1720   !
1730 Stringtest: !
1740   P$=" Please enter your name:"
1750   DIALOG "STRING",P$,Btn;SET ("TITLE":" Example: STRING Dialog"),RETURN
("VALUE":S$)
1760   SELECT Btn
1770   CASE 0
1780     CALL Printit("STRING Dialog: "&S$)
1790   CASE 1
1800     CALL Printit("STRING Dialog: No string input")
1810   END SELECT
1820   RETURN
1830   !
1840 Numtest: !
1850   P$="Please input a number:"
1860   DIALOG "NUMBER",P$,Btn;SET ("TITLE":" Example: NUMBER Dialog"),RETURN
("VALUE":X)
1870   SELECT Btn
1880   CASE 0
1890     CALL Printit("NUMBER Dialog: "&VAL$(X))
1900   CASE 1
1910     CALL Printit("NUMBER Dialog: No number input")
1920   END SELECT
1930   RETURN
1940   !
1950 Keytest: !
1960   P$="Please input a number:"
1970   DIALOG "KEYPAD",P$,Btn;SET ("TITLE":" Example: KEYPAD Dialog"),RETURN
("VALUE":X)
1980   SELECT Btn
1990   CASE 0
2000     CALL Printit("KEYPAD Dialog: "&VAL$(X))
2010   CASE 1
2020     CALL Printit("KEYPAD Dialog: No number input")
2030   END SELECT
2040   RETURN
2050   !
2060 Listtest: !
```

```
2070   P$="What is your favorite magazine?"
2080   DIALOG "LIST",P$,Btn;SET ("TITLE":" Example: LIST Dialog","ITEMS":M$(*)),RETURN
("SELECTION":N)
2090   IF Btn=0 AND N=-1 THEN CALL Printit("LIST Dialog: No selection from list")
2100   IF Btn=0 AND N<>-1 THEN CALL Printit("LIST Dialog: "&M$(N))
2110   RETURN
2120   !
2130 Combotest: !
2140   P$="What is your favorite magazine?"
2150   DIALOG "COMBO",P$,Btn;SET ("TITLE":" Example: COMBO Dialog","ITEMS":M$
(*)),RETURN ("TEXT":S$)
2160   IF Btn=0 AND S$="" THEN CALL Printit("COMBO Dialog: No selection from list")
2170   IF Btn=0 AND S$<>"" THEN CALL Printit("COMBO Dialog: "&S$)
2180   RETURN
2190   !
2200 Filetest: !
2210   P$="Please select a file:"
2220   DIALOG "FILE",P$,Btn;SET ("TITLE":" Example: FILE Dialog"),RETURN
("SELECTION":S$)
2230   SELECT Btn
2240   CASE 0
2250     CALL Printit("FILE Dialog: "&S$)
2260   CASE 1
2270     CALL Printit("FILE Dialog: No file input")
2280   END SELECT
2290   RETURN
2300   !
2310 Finis: !
2320   ASSIGN @Main TO *                ! Delete main panel
2330   END
2340   !
2350   ! Subprogram to print text to PRINTER widget
2360   !
2370   SUB Printit(S$)
2380     COM @Prn
2390     CONTROL @Prn;SET ("APPEND TEXT":S$)
2400   SUBEND
```

```
10      ! ***************************************************
20      ! Example: ERROR Dialog
30      !
40      ! This program creates an ERROR dialog. If the user selects
50      ! a button (ABORT or CONTINUE) within 10 seconds, the program
60      ! displays the choice. If the user does not make a choice
70      ! within 10 seconds, "Timeout occurred - no entry" is displayed.
80      !
90      ! ***************************************************
100    !
110    DIM T$[16],P$[40],A1$[20],S1$(1:2)[10],A2$[20]
120    INTEGER Btn
130    DATA "ERROR","Input caused overflow    "
140    READ T$,P$
150    DATA "DIALOG BUTTONS","ABORT","CONTINUE","DEFAULT BUTTON"
160    READ A1$,S1$(*),A2$
170     !
180    DIALOG T$,P$,Btn;SET ("TITLE":" Example: ERROR Dialog",A1$:S1$
(*),A2$:0),TIMEOUT 10
190    IF Btn=-1 THEN
200       DISP "Timeout occurred - no entry"
210    ELSE
220       DISP S1$(Btn+1)
230    END IF
240    END
```

```
10      ! ********************************************************
20      ! Example: Engine Monitor - Child PANEL
30      !
40      ! This program creates a simulated engine monitor display
50      ! that shows Torque, Pressure, Dwell, Timing, and RPM for
60      ! a theoretical engine. A status log for updated values is
70      ! also included.
80      !
90      ! ********************************************************
100     !
110     DIM Buf$[200]
120     !
130     ! Create the main panel widget
140     !
150     ASSIGN @Panel TO WIDGET "PANEL";SET
("X":100,"Y":30,"WIDTH":400,"HEIGHT":320,"TITLE":" Example: Engine Monitor - Child
PANEL","MAXIMIZABLE":1)
160     CONTROL @Panel;SET ("SYSTEM MENU":"Quit")
170     ON EVENT @Panel,"SYSTEM MENU" GOTO Finis
180     !
190     ASSIGN @M1lbl TO WIDGET "LABEL";SET
("X":5,"Y":5,"WIDTH":90,"HEIGHT":20,"VALUE":"Torque","BORDER":0),PARENT @Panel
200     ASSIGN @Meter1 TO WIDGET "METER";SET
("X":5,"Y":25,"WIDTH":90,"HEIGHT":160,"ARC WIDTH":3,"ORIENTATION":"RIGHT","SHOW
LIMITS":0),PARENT @Panel
210     !
220     ASSIGN @B2lbl TO WIDGET "LABEL";SET
("X":125,"Y":5,"WIDTH":80,"HEIGHT":20,"VALUE":"Pressure","BORDER":0),PARENT @Panel
230     ASSIGN @Bar2 TO WIDGET "BAR";SET
("X":130,"Y":25,"WIDTH":60,"HEIGHT":100),PARENT @Panel
240     ASSIGN @B2dsp TO WIDGET "LABEL";SET
("X":130,"Y":125,"WIDTH":60,"HEIGHT":20,"BORDER":0),PARENT @Panel
250     !
260     ! Create a child PANEL that contains a bank of engine measurements
270     !
280     ASSIGN @Subpanel TO WIDGET "PANEL";PARENT @Panel,SET
("X":260,"Y":5,"WIDTH":95,"HEIGHT":145)
290     !
300     ASSIGN @Lbl1 TO WIDGET "LABEL";PARENT @Subpanel,SET
("X":10,"Y":0,"WIDTH":75,"HEIGHT":20,"BORDER":0,"VALUE":"Dwell")
310     ASSIGN @Disp1 TO WIDGET "LABEL";PARENT @Subpanel,SET
("X":10,"Y":20,"WIDTH":75,"HEIGHT":20)
320     !
```

```
330     ASSIGN @Label2 TO WIDGET "LABEL";PARENT @Subpanel,SET
("X":10,"Y":50,"WIDTH":75,"HEIGHT":20,"BORDER":0,"VALUE":"Timing")
340     ASSIGN @Disp2 TO WIDGET "LABEL";PARENT @Subpanel,SET
("X":10,"Y":70,"WIDTH":75,"HEIGHT":20)
350     !
360     ASSIGN @Label3 TO WIDGET "LABEL";PARENT @Subpanel,SET
("X":10,"Y":100,"WIDTH":75,"HEIGHT":20,"BORDER":0,"VALUE":"RPM")
370     ASSIGN @Disp3 TO WIDGET "LABEL";PARENT @Subpanel,SET
("X":10,"Y":120,"WIDTH":75,"HEIGHT":20)
380     !
390     ASSIGN @Lbl TO WIDGET "LABEL";SET
("X":130,"Y":150,"WIDTH":240,"HEIGHT":20,"VALUE":"Status Log","BORDER":0),PARENT
@Panel
400     ASSIGN @Text TO WIDGET "PRINTER";SET
("X":130,"Y":170,"WIDTH":240,"HEIGHT":100),PARENT @Panel
410     !
420     Value=50
430     Siz=15
440 Loop_val: !
450     FOR I=1 TO 10000
460       IF Value<25 OR Value>=85 THEN Value=50
470       Torq=INT(Value+Siz*RND)
480       Pres=INT(Value+Siz*RND)
490       Dwell=INT(Value+Siz*RND)
500       Timing=INT(Value+Siz*RND)
510       Rpm=INT(Value+Siz*RND)
520       Value=INT(Value+Siz*(RND-.5))
530       !
540       CONTROL @Meter1;SET ("VALUE":Torq)
550       CONTROL @Bar2;SET ("VALUE":Pres)
560       CONTROL @B2dsp;SET ("VALUE":Pres)
570       CONTROL @Disp1;SET ("VALUE":Dwell)
580       CONTROL @Disp2;SET ("VALUE":Timing)
590       CONTROL @Disp3;SET ("VALUE":Rpm)
600       !
610       OUTPUT Buf$ USING "#,K,DDDD,3X,DDD,DDD,DDD";"Update #";I,Torq,Pres,Dwell
620       CONTROL @Text;SET ("APPEND TEXT":Buf$)
630     NEXT I
640     GOTO Loop_val
650     STOP
660 Finis: END
```

```
10      ! ******************************************************
20      ! Example: Engine Monitor - Level-0 Widgets
30      !
40      ! This program displays a simulated engine monitor for
50      ! a theoretical engine. Torque, pressure, and dwell are
60      ! displayed, as well as a Status Log.
70      !
80      ! ******************************************************
90      !
100     DIM Buf$[200]
110     !
120     ! Create the widgets
130     !
140     ASSIGN @Meter1 TO WIDGET "METER";SET
("X":200,"Y":10,"WIDTH":150,"HEIGHT":175,"TITLE":"Torque","ARC WIDTH":3)
150     ASSIGN @Bars1 TO WIDGET "BARS";SET
("X":375,"Y":10,"WIDTH":150,"HEIGHT":250,"TITLE":"Pressure","SHOW LIMITS":0)
160     ASSIGN @Dispdwell TO WIDGET "LABEL";SET
("X":200,"Y":200,"WIDTH":150,"HEIGHT":60,"TITLE":"Dwell")
170     ASSIGN @Logtext TO WIDGET "PRINTER";SET
("X":200,"Y":275,"WIDTH":325,"HEIGHT":100,"TITLE":"Status Log")
180     ASSIGN @Label TO WIDGET "LABEL"
190     CONTROL @Label;SET ("X":10,"Y":10,"WIDTH":150,"HEIGHT":25,"TITLE":" Quit
","SYSTEM MENU":"Quit")
200     ON EVENT @Label,"SYSTEM MENU" GOTO Finis
210     !
220     !   Run the widgets for the engine monitor
230     !
240     Value=50
250     Siz=15
260 Loop_val: !
270     FOR I=1 TO 10000
280       !
290       IF Value<25 OR Value>=85 THEN Value=50
300       Torq=INT(Value+Siz*RND)
310       Pres=INT(Value+Siz*RND)
320       Dwell=INT(Value+Siz*RND)
330       Value=INT(Value+Siz*(RND-.5))
340       !
350       CONTROL @Meter1;SET ("VALUE":Torq)
360       CONTROL @Bars1;SET ("VALUE":Pres)
370       CONTROL @Dispdwell;SET ("VALUE":Dwell)
```

```
380       OUTPUT Buf$ USING "#,K,DDDD,X,DDD,DDD,DDD";"Update #";I,Torq,Pres,Dwell
390       CONTROL @Logtext;SET ("APPEND TEXT":Buf$)
400     NEXT I
410     GOTO Loop_val
420     !
430     STOP
440 Finis: END
```

```
10      ! ********************************************************
20      ! Example: Engine Monitor - Panel of Widgets
30      !
40      ! This program displays a simulated engine monitor for
50      ! a theoretical engine. Torque, pressure, and dwell are
60      ! displayed, as well as a Status Log. V
70      !
80      ! ********************************************************
90      !
100     DIM Buf$[200]
110     !
120     ASSIGN @Panel TO WIDGET "PANEL";SET
("X":5,"Y":5,"WIDTH":400,"HEIGHT":320,"TITLE":" Example: Engine Monitor - Panel of
Widgets","MAXIMIZABLE":1)
130     CONTROL @Panel;SET ("SYSTEM MENU":"Quit")
140     ON EVENT @Panel,"SYSTEM MENU" GOTO Finis
150     !
160     ASSIGN @M1lbl TO WIDGET "LABEL";SET
("X":5,"Y":5,"WIDTH":90,"HEIGHT":20,"VALUE":"Torque","BORDER":0),PARENT @Panel
170     ASSIGN @Meter1 TO WIDGET "METER";SET
("X":5,"Y":30,"WIDTH":90,"HEIGHT":160,"ARC WIDTH":3,"ORIENTATION":"RIGHT","SHOW
LIMITS":0),PARENT @Panel
180     !
190     ASSIGN @B2lbl TO WIDGET "LABEL";SET
("X":150,"Y":5,"WIDTH":80,"HEIGHT":20,"VALUE":"Pressure","BORDER":0),PARENT @Panel
200     ASSIGN @Bar2 TO WIDGET "BAR";SET
("X":160,"Y":30,"WIDTH":60,"HEIGHT":90),PARENT @Panel
210     ASSIGN @B2dsp TO WIDGET "LABEL";SET
("X":160,"Y":125,"WIDTH":60,"HEIGHT":20,"BORDER":0),PARENT @Panel
220     !
230     ASSIGN @Lbl1 TO WIDGET "LABEL";SET
("X":270,"Y":5,"WIDTH":75,"HEIGHT":20,"BORDER":0,"VALUE":"Dwell"),PARENT @Panel
240     ASSIGN @Disp1 TO WIDGET "LABEL";SET
("X":270,"Y":30,"WIDTH":75,"HEIGHT":20),PARENT @Panel
250     ASSIGN @Lbl TO WIDGET "LABEL";SET
("X":130,"Y":155,"WIDTH":240,"HEIGHT":20,"VALUE":"Status Log","BORDER":0),PARENT
@Panel
260     ASSIGN @Text TO WIDGET "PRINTER";SET
("X":130,"Y":185,"WIDTH":240,"HEIGHT":100),PARENT @Panel
270     !
280     Value=50
290     Siz=15
300 Loop_val:    !
310     FOR I=1 TO 10000
```

```
320       IF Value<25 OR Value>=85 THEN Value=50
330       Torq=INT(Value+Siz*RND)
340       Pres=INT(Value+Siz*RND)
350       Dwell=INT(Value+Siz*RND)
360       Value=INT(Value+Siz*(RND-.5))
370       !
380       CONTROL @Meter1;SET ("VALUE":Torq)
390       CONTROL @Bar2;SET ("VALUE":Pres)
400       CONTROL @B2dsp;SET ("VALUE":Pres)
410       CONTROL @Disp1;SET ("VALUE":Dwell)
420       OUTPUT Buf$ USING "#,K,DDDD,3X,DDD,DDD,DDD";"Update #";I,Torq,Pres,Dwell
430       CONTROL @Text;SET ("APPEND TEXT":Buf$)
440     NEXT I
450     GOTO Loop_val
460     !
470     STOP
480 Finis: END
```

```
10      !  ********************************************************
20      !  Example: Environmental Chamber
30      !
40      !  This program shows how a user can build a custom display
50      !  panel. This program uses a strip chart, labels, menus,
60      !  dialog box, buttons, string input, and meters to simulate
70      !  a temperature/humidity chamber that can be programmed for
80      !  different temperature/humidity profiles.
90      !
100     !  ********************************************************
110     !
120     REAL Points(0:1)
130     DIM Buf$[100]
140     DIM Btn$(1:1,1:3)[80]
150     DIM Time(50),Temp(50),Humid(50),Ramp(50)
160     Btn$(1,1)="Options, Temperature"
170     Btn$(1,2)="Options, Humidity"
180     Btn$(1,3)="Options, Profile Manager"
190 Logimage: IMAGE #,K,K,K,K,K,DDDD.D,K,DDD.D
200 Logimage2: IMAGE #,K,DDDD,K,DDDD.D,K,DDD.D,K,DDD
210     !
220     ASSIGN @Main TO WIDGET "PANEL";SET
("X":0,"Y":0,"WIDTH":600,"HEIGHT":425,"TITLE":" Example: Environmental Chamber","SIZE
CONTROL":"RESIZE CHILDREN")
230     CONTROL @Main;SET ("SYSTEM MENU":"Quit")
240     ON EVENT @Main,"SYSTEM MENU" GOTO Finis
250     !
260     ASSIGN @Pulldown1 TO WIDGET "PULLDOWN MENU";PARENT @Main,SET
("LABEL":"Chart","SENSITIVE":0)
270     ASSIGN @Pd_btn1 TO WIDGET "MENU BUTTON";PARENT @Pulldown1,SET
("LABEL":"Temperature")
280     ASSIGN @Pd_btn2 TO WIDGET "MENU BUTTON";PARENT @Pulldown1,SET
("LABEL":"Humidity")
290     ASSIGN @Pd_btn3 TO WIDGET "MENU BUTTON";PARENT @Pulldown1,SET
("LABEL":"Profile Manager")
300     ASSIGN @Pulldown2 TO WIDGET "PULLDOWN MENU";PARENT @Main,SET
("LABEL":"Operation","SENSITIVE":0)
310     ASSIGN @Pd_btn4 TO WIDGET "MENU BUTTON";PARENT @Pulldown2,SET
("LABEL":"Manual")
320     ASSIGN @Pd_cscd TO WIDGET "CASCADE MENU";PARENT @Pulldown2,SET
("LABEL":"Automatic")
330     ASSIGN @Pd_btn5 TO WIDGET "MENU BUTTON";PARENT @Pd_cscd,SET
("LABEL":"(Re)start")
```

```
340     ASSIGN @Pd_btn6 TO WIDGET "MENU BUTTON";PARENT @Pd_cscd,SET
("LABEL":"Continue")
350     !
360     ON EVENT @Pd_btn1,"ACTIVATED" GOSUB Btn1
370     ON EVENT @Pd_btn2,"ACTIVATED" GOSUB Btn2
380     ON EVENT @Pd_btn3,"ACTIVATED" GOSUB Btn3
390     ON EVENT @Pd_btn4,"ACTIVATED" GOSUB Btn4
400     ON EVENT @Pd_btn5,"ACTIVATED" GOSUB Btn5
410     ON EVENT @Pd_btn6,"ACTIVATED" GOSUB Btn6
420     !
430     !   Create the widgets for the temperature profile.
440     !
450     ASSIGN @Main1 TO WIDGET "PANEL";PARENT @Main,SET
("X":0,"Y":0,"WIDTH":600,"HEIGHT":279,"BORDER":0,"VISIBLE":0,"SIZE CONTROL":"RESIZE
CHILDREN")
460     ASSIGN @Strip TO WIDGET "STRIPCHART";PARENT @Main1,SET
("X":5,"Y":5,"WIDTH":350,"HEIGHT":250,"SHOW NUMBERING":0)
470     ASSIGN @Lbl0 TO WIDGET "LABEL";PARENT @Main1,SET
("X":5,"Y":260,"WIDTH":350,"HEIGHT":20,"BORDER":0,"VALUE":"Event History -- Temperature")
480     ASSIGN @Lbl1 TO WIDGET "LABEL";PARENT @Main1,SET
("X":365,"Y":15,"WIDTH":120,"HEIGHT":20,"BORDER":0,"VALUE":"Curr Temp")
490     ASSIGN @Disp1 TO WIDGET "METER";PARENT @Main1,SET
("X":365,"Y":35,"WIDTH":120,"HEIGHT":150,"MINIMUM":-75,"MAXIMUM":+75)
500     ASSIGN @Lbl2 TO WIDGET "LABEL";PARENT @Main1,SET
("X":365,"Y":200,"WIDTH":120,"HEIGHT":20,"BORDER":0,"VALUE":"Set Point")
510     ASSIGN @Disp2 TO WIDGET "LABEL";PARENT @Main1,SET
("X":365,"Y":220,"WIDTH":120,"HEIGHT":25)
520     ASSIGN @Slider TO WIDGET "SLIDER";PARENT @Main1,SET
("X":505,"Y":5,"HEIGHT":250,"WIDTH":60,"MINIMUM":-50,"MAXIMUM":50)
530     !
540     ! Set up the strip chart
550     !
560     CONTROL @Strip;SET ("CURRENT AXIS":"Y","TICK
SPACING":10,"ORIGIN":-50,"RANGE":100)
570     CONTROL @Strip;SET ("CURRENT AXIS":"X","RANGE":20)
580     CONTROL @Disp1;SET ("LOW LIMIT":-45,"HIGH LIMIT":+45,"ALARM
RANGES":"LOW,HIGH")
590     STATUS @Disp1;RETURN ("LOW PEN":Low_pen,"HIGH PEN":High_pen)
600     CONTROL @Disp1;SET ("LOW PEN":High_pen,"MIDDLE PEN":Low_pen)
610     !
620     ! Create the widgets for the humidity profile
630     !
640     ASSIGN @Main2 TO WIDGET "PANEL";PARENT @Main,SET
("X":0,"Y":0,"WIDTH":600,"HEIGHT":279,"VISIBLE":0,"SIZE CONTROL":"RESIZE CHILDREN")
650     ASSIGN @Strip2 TO WIDGET "STRIPCHART";PARENT @Main2,SET
("X":5,"Y":5,"WIDTH":350,"HEIGHT":250,"SHOW NUMBERING":0)
```

```
660     ASSIGN @Lbl02 TO WIDGET "LABEL";PARENT @Main2,SET
("X":5,"Y":260,"WIDTH":350,"HEIGHT":20,"BORDER":0,"VALUE":"Event History -- Humidity")
670     ASSIGN @Lbl12 TO WIDGET "LABEL";PARENT @Main2,SET
("X":365,"Y":15,"WIDTH":120,"HEIGHT":20,"BORDER":0,"VALUE":"Curr Humidity")
680     ASSIGN @Disp12 TO WIDGET "METER";PARENT @Main2,SET
("X":365,"Y":35,"WIDTH":120,"HEIGHT":150,"MINIMUM":0)
690     ASSIGN @Lbl22 TO WIDGET "LABEL";PARENT @Main2,SET
("X":365,"Y":200,"WIDTH":120,"HEIGHT":20,"BORDER":0,"VALUE":"Set Point")
700     ASSIGN @Disp22 TO WIDGET "LABEL";PARENT @Main2,SET
("X":365,"Y":220,"WIDTH":120,"HEIGHT":25)
710     ASSIGN @Slider2 TO WIDGET "SLIDER";PARENT @Main2,SET
("X":505,"Y":5,"HEIGHT":250,"WIDTH":60)
720     !
730     ! Set up the strip chart
740     !
750     CONTROL @Strip2;SET ("CURRENT AXIS":"Y","TICK
SPACING":10,"ORIGIN":0,"RANGE":100)
760     CONTROL @Strip2;SET ("CURRENT AXIS":"X","RANGE":20)
770     CONTROL @Disp12;SET ("LOW LIMIT":5,"HIGH LIMIT":95,"ALARM
RANGES":"LOW,HIGH")
780     STATUS @Disp12;RETURN ("LOW PEN":Low_pen,"HIGH PEN":High_pen)
790     CONTROL @Disp12;SET ("LOW PEN":High_pen,"MIDDLE PEN":Low_pen)
800     !
810     ! Create widgets for profile manager
820     !
830     ASSIGN @Main3 TO WIDGET "PANEL";PARENT @Main,SET
("X":0,"Y":0,"WIDTH":600,"HEIGHT":279,"BORDER":0,"VISIBLE":0,"SIZE CONTROL":"RESIZE
CHILDREN")
840     ASSIGN @Lbl3 TO WIDGET "LABEL";PARENT @Main3,SET
("X":255,"Y":200,"WIDTH":90,"HEIGHT":20,"BORDER":0,"VALUE":"Low Limit")
850     ASSIGN @Lbl4 TO WIDGET "LABEL";PARENT @Main3,SET
("X":255,"Y":230,"WIDTH":90,"HEIGHT":20,"BORDER":0,"VALUE":"High Limit")
860     ASSIGN @Meter1 TO WIDGET "METER";PARENT @Main3,SET
("X":345,"Y":35,"WIDTH":120,"HEIGHT":150,"MINIMUM":-75,"MAXIMUM":+75)
870     ASSIGN @Meter2 TO WIDGET "METER";PARENT @Main3,SET
("X":465,"Y":35,"WIDTH":120,"HEIGHT":150,"MINIMUM":0)
880     ASSIGN @Lmeter1 TO WIDGET "LABEL";PARENT @Main3,SET
("X":345,"Y":15,"WIDTH":120,"HEIGHT":20,"BORDER":0,"VALUE":"Curr Temp")
890     ASSIGN @Lmeter2 TO WIDGET "LABEL";PARENT @Main3,SET
("X":465,"Y":15,"WIDTH":120,"HEIGHT":20,"BORDER":0,"VALUE":"Curr Humidity")
900     ASSIGN @Temp_low TO WIDGET "NUMBER";PARENT @Main3,SET
("X":345,"Y":200,"WIDTH":120,"HEIGHT":20,"REAL NOTATION":"FIXED","REAL
RESOLUTION":1)
910     ASSIGN @Temp_high TO WIDGET "NUMBER";PARENT @Main3,SET
("X":345,"Y":230,"WIDTH":120,"HEIGHT":20,"REAL NOTATION":"FIXED","REAL
RESOLUTION":1)
```

```
920    ASSIGN @Humid_low TO WIDGET "NUMBER";PARENT @Main3,SET
("X":465,"Y":200,"WIDTH":120,"HEIGHT":20,"REAL NOTATION":"FIXED","REAL
RESOLUTION":1)
930    ASSIGN @Humid_high TO WIDGET "NUMBER";PARENT @Main3,SET
("X":465,"Y":230,"WIDTH":120,"HEIGHT":20,"REAL NOTATION":"FIXED","REAL
RESOLUTION":1)
940    ASSIGN @Time TO WIDGET "NUMBER";PARENT @Main3,SET
("X":10,"Y":50,"WIDTH":75,"HEIGHT":20,"REAL NOTATION":"FIXED","REAL RESOLUTION":1)
950    ASSIGN @Temp TO WIDGET "NUMBER";PARENT @Main3,SET
("X":95,"Y":50,"WIDTH":50,"HEIGHT":20,"REAL NOTATION":"FIXED","REAL RESOLUTION":1)
960    ASSIGN @Humid TO WIDGET "NUMBER";PARENT @Main3,SET
("X":155,"Y":50,"WIDTH":50,"HEIGHT":20,"REAL NOTATION":"FIXED","REAL RESOLUTION":1)
970    ASSIGN @Slope TO WIDGET "NUMBER";PARENT @Main3,SET
("X":215,"Y":50,"WIDTH":50,"HEIGHT":20,"REAL NOTATION":"FIXED","REAL RESOLUTION":1)
980    ASSIGN @Save_me TO WIDGET "PUSHBUTTON";PARENT @Main3,SET
("X":275,"Y":50,"WIDTH":60,"HEIGHT":24,"LABEL":"Set","BORDER":0)
990    ASSIGN @Kill_me TO WIDGET "PUSHBUTTON";PARENT @Main3,SET
("X":275,"Y":80,"WIDTH":60,"HEIGHT":24,"LABEL":"Clear","BORDER":0)
1000   ASSIGN @Time_disp TO WIDGET "LABEL";PARENT @Main3,SET
("X":10,"Y":30,"WIDTH":75,"HEIGHT":20,"VALUE":"Time")
1010   ASSIGN @Temp_disp TO WIDGET "LABEL";PARENT @Main3,SET
("X":95,"Y":30,"WIDTH":50,"HEIGHT":20,"VALUE":"Temp")
1020   ASSIGN @Humid_disp TO WIDGET "LABEL";PARENT @Main3,SET
("X":155,"Y":30,"WIDTH":50,"HEIGHT":20,"VALUE":"Humid")
1030   ASSIGN @Ramp_disp TO WIDGET "LABEL";PARENT @Main3,SET
("X":215,"Y":30,"WIDTH":50,"HEIGHT":20,"VALUE":"Ramp")
1040   ASSIGN @Save_disp TO WIDGET "LABEL";PARENT @Main3,SET
("X":275,"Y":30,"WIDTH":60,"HEIGHT":20,"VALUE":"Config")
1050   ASSIGN @Text2 TO WIDGET "PRINTER";PARENT @Main3,SET
("X":5,"Y":80,"WIDTH":250,"HEIGHT":200)
1060   CONTROL @Meter1;SET ("LOW LIMIT":-45,"HIGH LIMIT":+45,"ALARM
RANGES":"LOW,HIGH")
1070   STATUS @Meter1;RETURN ("LOW PEN":Low_pen,"HIGH PEN":High_pen)
1080   CONTROL @Meter1;SET ("LOW PEN":High_pen,"MIDDLE PEN":Low_pen)
1090   CONTROL @Meter2;SET ("LOW LIMIT":5,"HIGH LIMIT":95,"ALARM
RANGES":"LOW,HIGH")
1100   STATUS @Meter2;RETURN ("LOW PEN":Low_pen,"HIGH PEN":High_pen)
1110   CONTROL @Meter2;SET ("LOW PEN":High_pen,"MIDDLE PEN":Low_pen)
1120   GOSUB Setup_profile
1130   !
1140   ! Printer Widget appears in all subpanels.
1150   !
1160   ASSIGN @Text TO WIDGET "PRINTER";PARENT @Main,SET
("X":5,"Y":280,"WIDTH":580,"HEIGHT":80)
1170   !
1180   ! From here to "Afteron" represents:   ON EVENT @Btn1 GOSUB Servicebtn
1190   !
```

```
1200   ON EVENT @Slider,"CHANGED" GOSUB Changesetpt
1210   ON EVENT @Slider,"DONE" GOSUB Changesetpt
1220   ON EVENT @Slider2,"CHANGED" GOSUB Changesetpt
1230   ON EVENT @Slider2,"DONE" GOSUB Changesetpt
1240   ON EVENT @Temp_low,"DONE" GOSUB Change_tlow
1250   ON EVENT @Temp_high,"DONE" GOSUB Change_thigh
1260   ON EVENT @Humid_low,"DONE" GOSUB Change_hlow
1270   ON EVENT @Humid_high,"DONE" GOSUB Change_hhigh
1280   ON EVENT @Save_me,"ACTIVATED" GOSUB Modify_profile
1290   ON EVENT @Kill_me,"ACTIVATED" GOSUB Clear_profile
1300 Afteron:!
1310   !
1320   !   Get down to work
1330   !
1340   Setpt=23
1350   Setpt2=30
1360   Curtemp=0
1370   Curhumid=10
1380   Siz=2
1390   Index=1
1400   CONTROL @Slider;SET ("VALUE":Setpt)
1410   CONTROL @Slider2;SET ("VALUE":Setpt2)
1420   CONTROL @Time;SET ("VALUE":0)
1430   CONTROL @Temp;SET ("VALUE":Setpt)
1440   CONTROL @Humid;SET ("VALUE":Setpt2)
1450   CONTROL @Slope;SET ("VALUE":0)
1460   CONTROL @Main1;SET ("VISIBLE":1)
1470   CONTROL @Pulldown1;SET ("SENSITIVE":1)
1480   CONTROL @Pulldown2;SET ("SENSITIVE":1)
1490   !
1500 Gotnewsetpt:!
1510   Reloop=0
1520   CONTROL @Disp2;SET ("VALUE":Setpt)
1530   CONTROL @Disp22;SET ("VALUE":Setpt2)
1540   Evt=Evt+1
1550   OUTPUT Buf$ USING Logimage;"#";Evt;": Time = ";TIME$(TIMEDATE);"     Setpoint
changed to ";Setpt;" Deg. C,     ";Setpt2;" % Hum."
1560   CONTROL @Text;SET ("APPEND TEXT":Buf$)
1570   LOOP
1580     !
1590     IF Auto=1 THEN GOSUB Get_next_setpt
1600     Diff=Setpt-Curtemp
```

```
1610      Diff2=Setpt2-Curhumid
1620      Noise=Siz*(RND-.5)*2
1630      Delta=Diff/20+Noise
1640      Delta2=Diff2/20+Noise
1650      Curtemp=Curtemp+Delta
1660      Curhumid=Curhumid+Delta2
1670      !
1680      CONTROL @Strip;SET ("POINT LOCATION":I)
1690      CONTROL @Strip2;SET ("POINT LOCATION":I)
1700      Points(0)=Curtemp
1710      Points(1)=Setpt
1720      I=I+1
1730      CONTROL @Strip;SET ("VALUES":Points(*))
1740      CONTROL @Disp1;SET ("VALUE":Curtemp)
1750      CONTROL @Meter1;SET ("VALUE":Curtemp)
1760      Points(0)=Curhumid
1770      Points(1)=Setpt2
1780      CONTROL @Strip2;SET ("VALUES":Points(*))
1790      CONTROL @Disp12;SET ("VALUE":Curhumid)
1800      CONTROL @Meter2;SET ("VALUE":Curhumid)
1810      !
1820    EXIT IF Reloop
1830    END LOOP
1840    GOTO Gotnewsetpt
1850    !
1860 Servicebtn: !
1870    STATUS @Str1;RETURN ("VALUE":Buf$)
1880    Setpt=VAL(Buf$)
1890    Reloop=1
1900    RETURN
1910    STOP
1920 Btn1:! Temperature
1930    GOSUB Turn_off_setup
1940    GOSUB Turn_off_humid
1950    GOSUB Turn_on_temp
1960    RETURN
1970 Btn2:! Humidity
1980    GOSUB Turn_off_setup
1990    GOSUB Turn_off_temp
2000    GOSUB Turn_on_humid
2010    RETURN
2020 Btn3:!
```

```
2030   GOSUB Turn_off_humid
2040   GOSUB Turn_off_temp
2050   GOSUB Turn_on_setup
2060   RETURN
2070 Btn4:! Manual
2080   Auto=0
2090   STATUS @Strip;RETURN ("VISIBLE":Temp_vis)
2100   STATUS @Strip2;RETURN ("VISIBLE":Humid_vis)
2110   CONTROL @Slider2;SET ("VALUE":Setpt2)
2120   CONTROL @Slider;SET ("VALUE":Setpt)
2130   IF Temp_vis THEN CONTROL @Slider;SET ("VISIBLE":1)
2140   IF Humid_vis THEN CONTROL @Slider2;SET ("VISIBLE":1)
2150   RETURN
2160 Btn5:!
2170   Auto=1
2180   CONTROL @Slider;SET ("VISIBLE":0)
2190   CONTROL @Slider2;SET ("VISIBLE":0)
2200   Index=1
2210   Time_start=TIMEDATE
2220   RETURN
2230 Btn6:!
2240   Auto=1
2250   CONTROL @Slider;SET ("VISIBLE":0)
2260   CONTROL @Slider2;SET ("VISIBLE":0)
2270   RETURN
2280   !
2290 Turn_off_temp:!
2300   CONTROL @Main1;SET ("VISIBLE":0)
2310   RETURN
2320 Turn_off_humid:!
2330   CONTROL @Main2;SET ("VISIBLE":0)
2340   RETURN
2350 Turn_on_temp:!
2360   CONTROL @Main1;SET ("VISIBLE":1)
2370   IF (Auto=0) THEN CONTROL @Slider;SET ("VISIBLE":1)
2380   RETURN
2390 Turn_on_humid:!
2400   CONTROL @Main2;SET ("VISIBLE":1)
2410   IF (Auto=0) THEN CONTROL @Slider2;SET ("VISIBLE":1)
2420   RETURN
2430 Turn_off_setup:!
2440   CONTROL @Main3;SET ("VISIBLE":0)
```

```
2450   RETURN
2460 Turn_on_setup:!
2470   CONTROL @Main3;SET ("VISIBLE":1)
2480   STATUS @Meter1;RETURN ("LOW LIMIT":Low_lim,"HIGH LIMIT":High_lim)
2490   CONTROL @Temp_low;SET ("VALUE":Low_lim)
2500   CONTROL @Temp_high;SET ("VALUE":High_lim)
2510   STATUS @Meter2;RETURN ("LOW LIMIT":Low_lim,"HIGH LIMIT":High_lim)
2520   CONTROL @Humid_high;SET ("VALUE":High_lim)
2530   CONTROL @Humid_low;SET ("VALUE":Low_lim)
2540   RETURN
2550 Changesetpt:!
2560   IF Auto=0 THEN
2570     STATUS @Slider;RETURN ("VALUE":Setpt)
2580     STATUS @Slider2;RETURN ("VALUE":Setpt2)
2590     CONTROL @Disp2;SET ("VALUE":Setpt)
2600     CONTROL @Disp22;SET ("VALUE":Setpt2)
2610     Evt=Evt+1
2620     OUTPUT Buf$ USING Logimage;"#";Evt;": Time = ";TIME$(TIMEDATE);"     Setpoint
changed to ";Setpt;" Deg. C, ";Setpt2;" % Hum."
2630     CONTROL @Text;SET ("APPEND TEXT":Buf$)
2640   END IF
2650   RETURN
2660 Change_tlow:!
2670   STATUS @Temp_low;RETURN ("VALUE":Low_tlim)
2680   CONTROL @Disp1;SET ("LOW LIMIT":Low_tlim)
2690   CONTROL @Meter1;SET ("LOW LIMIT":Low_tlim)
2700   RETURN
2710 Change_thigh:!
2720   STATUS @Temp_high;RETURN ("VALUE":High_tlim)
2730   CONTROL @Disp1;SET ("HIGH LIMIT":High_tlim)
2740   CONTROL @Meter1;SET ("HIGH LIMIT":High_tlim)
2750   RETURN
2760 Change_hlow:!
2770   STATUS @Humid_low;RETURN ("VALUE":Low_hlim)
2780   CONTROL @Disp12;SET ("LOW LIMIT":Low_hlim)
2790   CONTROL @Meter2;SET ("LOW LIMIT":Low_hlim)
2800   RETURN
2810 Change_hhigh:!
2820   STATUS @Humid_high;RETURN ("VALUE":High_hlim)
2830   CONTROL @Disp12;SET ("HIGH LIMIT":High_hlim)
2840   CONTROL @Meter2;SET ("HIGH LIMIT":High_hlim)
2850   RETURN
```

```
2860 Modify_profile:!
2870   STATUS @Time;RETURN ("VALUE":Time(Next))
2880   STATUS @Temp;RETURN ("VALUE":Temp(Next))
2890   STATUS @Humid;RETURN ("VALUE":Humid(Next))
2900   STATUS @Slope;RETURN ("VALUE":Ramp(Next))
2910   OUTPUT Buf$ USING Logimage2;"Time = ";Time(Next);"     Temp=";Temp(Next);" Deg. C,
Humid=";Humid(Next);" % Ramp=";Ramp(Next)
2920   CONTROL @Text2;SET ("APPEND TEXT":Buf$)
2930   Next=Next+1
2940   RETURN
2950 Clear_profile:!
2960   Next=1
2970   CONTROL @Text2;SET ("TEXT":"")
2980   RETURN
2990 Setup_profile: !
3000   DATA 0,15,35,55,75
3010   DATA 10,50,0,-30,23
3020   DATA 10,90,10,15,20
3030   DATA 2, 5, 7, 5, 10
3040   READ Time(1),Time(2),Time(3),Time(4),Time(5)
3050   READ Temp(1),Temp(2),Temp(3),Temp(4),Temp(5)
3060   READ Humid(1),Humid(2),Humid(3),Humid(4),Humid(5)
3070   READ Ramp(1),Ramp(2),Ramp(3),Ramp(4),Ramp(5)
3080   FOR I=1 TO 5
3090     OUTPUT Buf$ USING Logimage2;"Time = ";Time(I);"     Temp=";Temp(I);" Deg. C,
Humid=";Humid(I);" % Ramp=";Ramp(I)
3100     CONTROL @Text2;SET ("APPEND TEXT":Buf$)
3110   NEXT I
3120   Next=6
3130   RETURN
3140 Get_next_setpt: !
3150   Delta_time=TIMEDATE-Time_start
3160   Time(Next)=Time(Next-1)+Ramp(Index)+10
3170   WHILE (Delta_time>Time(Index+1) AND Index<>Next)
3180     Index=Index+1
3190   END WHILE
3200   IF Index=Next THEN
3210     Auto=0
3220     DIALOG "INFORMATION","The Temperature/Humidity profile has completed!";SET
("X":200,"Y":300)
3230     GOTO Btn4
3240   END IF
3250   IF (Delta_time<Time(Index)+Ramp(Index) AND Delta_time>Time(Index)) THEN
```

```
3260      IF (Index=1) THEN STATUS @Disp2;RETURN ("VALUE":Foo)
3270      IF (Index=1) THEN Temp(0)=Foo     ! Cannot put Temp(0) in place of Foo
3280      IF (Index=1) THEN STATUS @Disp22;RETURN ("VALUE":Foo)
3290      IF (Index=1) THEN Humid(0)=Foo
3300      Mult=(Delta_time-Time(Index))/Ramp(Index)
3310      Setpt=Temp(Index-1)+(Temp(Index)-Temp(Index-1))*Mult
3320      Setpt2=Humid(Index-1)+(Humid(Index)-Humid(Index-1))*Mult
3330    ELSE
3340      Setpt=Temp(Index)
3350      Setpt2=Humid(Index)
3360    END IF
3370    CONTROL @Disp22;SET ("VALUE":Setpt2)
3380    CONTROL @Disp2;SET ("VALUE":Setpt)
3390    IF (Setpt<>Prevst) OR (Setpt2<>Prevst2) THEN
3400      Evt=Evt+1
3410      OUTPUT Buf$ USING Logimage;"#";Evt;": Time = ";TIME$(TIMEDATE);"     Setpoint
changed to ";Setpt;" Deg. C, ";Setpt2;" % Hum."
3420      CONTROL @Text;SET ("APPEND TEXT":Buf$)
3430    END IF
3440    Prevst=Setpt
3450    Prevst2=Setpt2
3460    RETURN
3470 Finis: END
```

```
10      ! ****************************************
20      ! Example: FILE Dialog
30      !
40      ! This program produces a FILE Dialog. A file name,
50      ! which the   user selects from the list or types in,
60      ! is returned through the S$ string variable. The
70      ! button selected (OK or Cancel) is also displayed.
80      !
90      ! ****************************************
100   !
110   CLEAR SCREEN
120   INTEGER Btn
130   DIM S$[64]
140   DIALOG "FILE","Please select a file:",Btn;SET ("TITLE":" Example: FILE Dialog"),RETURN
("SELECTION":S$)
150   PRINT "Button                File"
160   PRINT
170   IF Btn=0 THEN
180     PRINT "   OK",S$
190   ELSE
200     PRINT "Cancel",S$
210   END IF
220   END
```

```
10       ! ************************************************
20       ! Example: FILE Widget
30       !
40       ! This program generates a FILE widget.
50       !
60       ! ************************************************
70       !
80       ASSIGN @File TO WIDGET "FILE";SET ("VISIBLE":0)
90       CONTROL @File;SET ("TITLE":" Example: FILE Widget")
100     CONTROL @File;SET ("DIRECTORY":"C:/DOC","PATTERN":"*.TXT")
110     CONTROL @File;SET ("SYSTEM MENU":"Quit")
120     CONTROL @File;SET ("X":100,"Y":50,"VISIBLE":1)
130     ON EVENT @File,"SELECTION" GOSUB Get_selection
140     ON EVENT @File,"SYSTEM MENU" GOTO Finis
150     LOOP
160       WAIT FOR EVENT
170     END LOOP
180 Get_selection: DIM File_name$[256]
190     STATUS @File;RETURN ("SELECTION":File_name$)
200     DISP File_name$
210     RETURN
220 Finis: END
```

```
10      ! ************************************************************
20      ! Example: Frequency Response
30      !
40      ! This program generates a theoretical frequency response
50      ! display (response in dB vs. frequency in Hz) for two channels.
60      !
70      ! ************************************************************
80      !
90      REAL Freq(1:31),Channel_1(1:31),Channel_2(1:31)
100   !
110   ! Define test frequencies.
120   !
130   DATA 20,25,32,40,50,63,80,100,125,160
140   DATA 200,250,320,400,500,630,800,1000,1250,1600
150   DATA 2000,2500,3200,4000,5000,6300,8000,10000,12500,16000,20000
160   READ Freq(*)
170   !
180   ! Take voltage measurements for both channels, convert to dB.
190   !
200   FOR I=1 TO 31
210     Channel_1(I)=20*LGT(FNMeasure(Freq(I),1))
220     Channel_2(I)=20*LGT(FNMeasure(Freq(I),2))
230   NEXT I
240   !
250   ! Create and set up the graph
260   !
270   ASSIGN @Graph TO WIDGET "XY GRAPH";SET ("SHARED X":1,"VISIBLE":0,"TITLE":"
Example: Frequency Response","WIDTH":400,"HEIGHT":300)
280   CONTROL @Graph;SET ("SYSTEM MENU":"Quit")
290   ON EVENT @Graph,"SYSTEM MENU" GOTO Finis
300   !
310   ! Set X axis attributes
320   !
330   CONTROL @Graph;SET ("CURRENT
AXIS":"X","AUTOSCALE":1,"LOGARITHMIC":1,"AXIS LABEL":"Frequency (Hz)")
340   !
350   ! Set Y axis attributes
360   !
370   CONTROL @Graph;SET ("CURRENT AXIS":"Y","AUTOSCALE":1,"AXIS
LABEL":"Response (dB)")
```

```
380     !
390     ! Enter the data into the graph
400     !
410     CONTROL @Graph;SET ("X DATA":Freq(*))
420     CONTROL @Graph;SET ("CURRENT TRACE":1,"Y DATA":Channel_1(*),"TRACE
LABEL":"Channel 1")
430     CONTROL @Graph;SET ("CURRENT TRACE":2,"Y DATA":Channel_2(*),"TRACE
LABEL":"Channel 2")
440     !
450     ! Display the graph
460     !
470     CONTROL @Graph;SET ("VISIBLE":1)
480     LOOP
490       WAIT FOR EVENT
500     END LOOP
510 Finis: END
520     !
530     ! This function simulates some data for the use of this program
540     !
550     DEF FNMeasure(Freq,Chan)
560     !
570     ! Simulate measured data for second order lowpass filters
580     !
590       F=Freq*(.8+Chan/5)
600       RETURN 1/(1-F/(3000-Chan*400)+(F/3000)^2)
610     FNEND
```

```
10      ! ************************************************************************
20      ! Example: Function Generator
30      !
40      ! This program shows how a panel, menu, and pushbuttons can be
50      ! used to construct a front panel for a function generator.
60      !
70      ! ************************************************************************
80      !
90      ! This part of the program creates a panel with one menu selection.
100     ! Four menu buttons are put into the menu. When any of these menu
110     ! items is selected, an appropriate subroutine is called.
120     !
130     DIM Attr$(3)[15],Attr(3)
140     COM
/Fgen/Freq,Ampl,Ampl_unit,Func,Offset,@Freq_disp,@Ampl_disp,@Offs_disp,@Func_disp,@A
mplitude,@Am,@Pm
150     ASSIGN @Fgen_panel TO WIDGET "PANEL";SET ("TITLE":" Example: Function
Generator","X":150,"Y":80,"WIDTH":240,"HEIGHT":200,"MAXIMIZABLE":0,"RESIZABLE":0)
160     CONTROL @Fgen_panel;SET ("SYSTEM MENU":"Quit")
170     ON EVENT @Fgen_panel,"SYSTEM MENU" GOTO Finis
180     !
190     ASSIGN @Fgen_main TO WIDGET "PULLDOWN MENU";SET ("LABEL":"Control
Menu"),PARENT @Fgen_panel
200     ASSIGN @Frequency TO WIDGET "MENU BUTTON";SET
("LABEL":"Frequency"),PARENT @Fgen_main
210     ASSIGN @Amplitude TO WIDGET "MENU BUTTON";SET
("LABEL":"Amplitude"),PARENT @Fgen_main
220     ASSIGN @Dc_offset TO WIDGET "MENU BUTTON";SET ("LABEL":"DC Offset"),PARENT
@Fgen_main
230     ASSIGN @Function TO WIDGET "MENU BUTTON";SET ("LABEL":"Function"),PARENT
@Fgen_main
240     ASSIGN @Modulation TO WIDGET "CASCADE MENU";SET
("LABEL":"Modulation"),PARENT @Fgen_main
250     ASSIGN @Am TO WIDGET "MENU TOGGLE";SET ("LABEL":"AM"),PARENT
@Modulation
260     ASSIGN @Pm TO WIDGET "MENU TOGGLE";SET ("LABEL":"PM"),PARENT
@Modulation
270     !
280     !   Initial values for the controls
290     !
300     Freq=1000
310     Ampl=.001
```

```
320     Ampl_unit=0
330     Offset=0
340     Func=0
350     !
360     ! Displays for the values of the controls. The displays are actually
370     ! pushbuttons and the label for each button is used to display its
380     ! current value.
390     !
400     Attr$(0)="X"
410     Attr$(1)="Y"
420     Attr$(2)="WIDTH"
430     Attr$(3)="HEIGHT"
440     Attr(0)=30                      ! X position
450     Attr(1)=8                       ! Y position
460     Attr(2)=178                     ! width
470     Attr(3)=33                      ! height
480     !
490     ASSIGN @Freq_disp TO WIDGET "PUSHBUTTON";SET (Attr$(*):Attr(*),"LABEL":VAL$
(Freq)&" Hz"),PARENT @Fgen_panel
500     Attr(1)=Attr(1)+Attr(3)
510     ASSIGN @Ampl_disp TO WIDGET "PUSHBUTTON";SET (Attr$(*):Attr(*),"LABEL":VAL$
(Ampl)&" V p-p"),PARENT @Fgen_panel
520     Attr(1)=Attr(1)+Attr(3)
530     ASSIGN @Offs_disp TO WIDGET "PUSHBUTTON";SET (Attr$(*):Attr(*),"LABEL":VAL$
(Offset)&" V offset"),PARENT @Fgen_panel
540     Attr(1)=Attr(1)+Attr(3)
550     ASSIGN @Func_disp TO WIDGET "PUSHBUTTON";SET (Attr$
(*):Attr(*),"LABEL":"SINE"),PARENT @Fgen_panel
560     !
570     ! When either the menu is pulled down, or the display is clicked, call
580     ! a routine to change that control.
590     !
600     ON EVENT @Frequency,"ACTIVATED",1 CALL Set_frequency
610     ON EVENT @Function,"ACTIVATED",1 CALL Set_function
620     ON EVENT @Amplitude,"ACTIVATED",1 CALL Set_amplitude
630     ON EVENT @Dc_offset,"ACTIVATED",1 CALL Set_offset
640     ON EVENT @Freq_disp,"ACTIVATED",1 CALL Set_frequency
650     ON EVENT @Func_disp,"ACTIVATED",1 CALL Set_function
660     ON EVENT @Ampl_disp,"ACTIVATED",1 CALL Set_amplitude
670     ON EVENT @Offs_disp,"ACTIVATED",1 CALL Set_offset
680     !
690     ! Event handling for modulation toggles in the menu
700     !
```

```
710     ON EVENT @Am,"CHANGED",1 CALL Toggle_am
720     ON EVENT @Pm,"CHANGED",1 CALL Toggle_pm
730     !
740     LOOP
750       WAIT FOR EVENT
760     END LOOP
770 Finis: END
780     SUB Set_frequency
790     !
800     ! This routine sets the value of frequency.
810     !
820       COM
/Fgen/Freq,Ampl,Ampl_unit,Func,Offset,@Freq_disp,@Ampl_disp,@Offs_disp,@Func_disp,@A
mplitude,@Am,@Pm
830       !
840       ! The user is allowed to enter the new frequency in any of three units.
850       ! A cancel button is included in case the user makes a mistake.
860       !
870       DIM Btn$(3)[6],Freq$[20]
880       Btn$(0)="Hz"
890       Btn$(1)="kHz"
900       Btn$(2)="MHz"
910       Btn$(3)="Cancel"
920       !
930       ! Use a STRING dialog to get the new frequency. This dialog returns
940       ! a string so a conversion must be done.
950       !
960       DIALOG "STRING","",Btn;SET ("VALUE":VAL$(Freq),"TITLE":"Enter
Frequency","DIALOG BUTTONS":Btn$(*),"DEFAULT BUTTON":0),RETURN ("VALUE":Freq$)
970       !
980       IF Btn=3 THEN SUBEXIT                    ! Cancel was clicked
990       ON ERROR GOTO Cant_convert
1000      Freq=VAL(Freq$)                          ! convert to a number
1010      OFF ERROR
1020      !
1030      ! The value is checked for validity. The valid range might depend on
1040      ! function, but that case is not handled.
1050      !
1060      Freq=DROUND(Freq*10^(3*Btn),11)      ! apply the suffix and round
1070      IF Freq>2.0E+7 THEN Frequ=2.0E+7
1080      CONTROL @Freq_disp;SET ("LABEL":VAL$(Freq)&" Hz")
1090      !
1100      ! If actual instruments were being controlled, the OUTPUT statement
```

```
1110      ! would be here.
1120      !
1130      SUBEXIT
1140 Cant_convert:     !
1150      !
1160      ! The string could not be converted to a number. Display an error
1170      ! message and ask again.
1180      !
1190      DIALOG "ERROR",Freq$&CHR$(10)&"is not recognizable as a number.";SET
("TITLE":"Can't convert to a number")
1200      OFF ERROR
1210      GOTO 960
1220    SUBEND
1230    SUB Set_function
1240      COM
/Fgen/Freq,Ampl,Ampl_unit,Func,Offset,@Freq_disp,@Ampl_disp,@Offs_disp,@Func_disp,@A
mplitude,@Am,@Pm
1250      DIM Func$(5)[10]
1260      Func$(0)="SINE"
1270      Func$(1)="SQUARE"
1280      Func$(2)="TRIANGLE"
1290      Func$(3)="POS RAMP"
1300      Func$(4)="NEG RAMP"
1310      Func$(5)="DC only"
1320      !
1330      ! The button value is placed in a temporary variable in case Cancel
1340      ! is clicked. Only if OK is clicked is the value transferred to
1350      ! the actual variable which contains the function.
1360      !
1370      DIALOG "LIST","",Btn;SET ("TITLE":"Select Function","ITEMS":Func$
(*),"SELECTION":Func,"DEFAULT BUTTON":0),RETURN ("SELECTION":A)
1380      IF Btn=1 THEN SUBEXIT
1390      Func=A
1400      CONTROL @Func_disp;SET ("LABEL":Func$(Func))
1410      !
1420      ! If DC only function is selected then the amplitude control has
1430      ! no effect.    The button and menu are deactivated.
1440      !
1450      IF Func=5 THEN
1460        CONTROL @Ampl_disp;SET ("SENSITIVE":0)
1470        CONTROL @Amplitude;SET ("SENSITIVE":0)
1480      ELSE
1490        CONTROL @Ampl_disp;SET ("SENSITIVE":1)
```

```
1500        CONTROL @Amplitude;SET ("SENSITIVE":1)
1510      END IF
1520    SUBEND
1530    SUB Set_amplitude
1540      COM
/Fgen/Freq,Ampl,Ampl_unit,Func,Offset,@Freq_disp,@Ampl_disp,@Offs_disp,@Func_disp,@A
mplitude,@Am,@Pm
1550      DIM Btn$(5)[7],Ampl$[20]
1560      Btn$(0)=" V p-p"
1570      Btn$(1)=" mV p-p"
1580      Btn$(2)=" V RMS"
1590      Btn$(3)=" mV RMS"
1600      Btn$(4)=" dBm"
1610      Btn$(5)="Cancel"
1620      DIALOG "STRING","",Btn;SET ("VALUE":VAL$(Ampl),"TITLE":"Enter
Amplitude","DIALOG BUTTONS":Btn$(*),"DEFAULT BUTTON":0),RETURN ("VALUE":Ampl$)
1630      IF Btn=5 THEN SUBEXIT                  ! Cancel was clicked
1640      ON ERROR GOTO Cant_convert
1650      Ampl=VAL(Ampl$)
1660      OFF ERROR
1670      Ampl_unit=Btn
1680      IF Btn=1 OR Btn=3 THEN
1690        Ampl=Ampl*.001                       ! A mV button was selected
1700        Ampl_unit=Ampl_unit-1                ! Change to V unit
1710      END IF
1720      IF Btn<4 THEN
1730        !
1740        ! A volt button was used. Round and check for limits. The limit
1750        ! might also depend on the dc offset.
1760        !
1770        Ampl=ABS(PROUND(Ampl,-3))
1780        IF Ampl>5 THEN Ampl=5
1790      ELSE
1800        !
1810        ! dBm was selected. Round and check for limits.
1820        !
1830        Ampl=PROUND(Ampl,-2)
1840        IF Ampl>13 THEN Ampl=13
1850        IF Ampl<-60 THEN Ampl=-60
1860      END IF
1870      CONTROL @Ampl_disp;SET ("LABEL":VAL$(Ampl)&Btn$(Ampl_unit))
1880      SUBEXIT
1890 Cant_convert:    !
```

```
1900      DIALOG "ERROR",Ampl$&CHR$(10)&"is not recognizable as a number.";SET
("TITLE":"Can't convert to a number")
1910      GOTO 1620
1920    SUBEND
1930    SUB Set_offset
1940      COM
/Fgen/Freq,Ampl,Ampl_unit,Func,Offset,@Freq_disp,@Ampl_disp,@Offs_disp,@Func_disp,@A
mplitude,@Am,@Pm
1950      DIM Btn$(2)[6],Offs$[20]
1960      Btn$(0)=" V"
1970      Btn$(1)=" mV"
1980      Btn$(2)="Cancel"
1990      DIALOG "STRING","",Btn;SET ("VALUE":VAL$(Offset),"TITLE":"Enter DC
Offset","DIALOG BUTTONS":Btn$(*),"DEFAULT BUTTON":0),RETURN ("VALUE":Offs$)
2000      IF Btn=2 THEN SUBEXIT   ! Cancel was selected
2010      ON ERROR GOTO Cant_convert
2020      Offset=VAL(Offs$)
2030      OFF ERROR
2040      IF Btn=1 THEN Offset=Offset*.001                ! mV button
2050      Offset=PROUND(Offset,-3)                         ! Round to 3 digits
2060      !
2070      IF ABS(Offset)>5 THEN Offset=SGN(Offset)*5   ! no more than 5 V offset
2080      CONTROL @Offs_disp;SET ("LABEL":VAL$(Offset)&" V offset")
2090      SUBEXIT
2100 Cant_convert:    !
2110      DIALOG "ERROR",Offs$&CHR$(10)&"is not recognizable as a number.";SET
("TITLE":"Can't convert to a number")
2120      GOTO 1990
2130    SUBEND
2140    SUB Toggle_am
2150      COM
/Fgen/Freq,Ampl,Ampl_unit,Func,Offset,@Freq_disp,@Ampl_disp,@Offs_disp,@Func_disp,@A
mplitude,@Am,@Pm
2160    !
2170    ! The menu widgets handle displaying whether modulation is on or
2180    ! off automatically. This routine determines what the state is and
2190    ! does the appropriate I/O.
2200    !
2210      STATUS @Am;RETURN ("VALUE":State)
2220      IF State THEN
2230    !
2240    ! Send the command to turn AM on
2250    !
2260      ELSE
```

```
2270   !
2280   ! Send the command to turn AM off
2290   !
2300      END IF
2310   SUBEND
2320   SUB Toggle_pm
2330      COM
/Fgen/Freq,Ampl,Ampl_unit,Func,Offset,@Freq_disp,@Ampl_disp,@Offs_disp,@Func_disp,@A
mplitude,@Am,@Pm
2340   !
2350   ! The menu widgets handle displaying whether modulation is on or
2360   ! off automatically. This routine determines what the state is and
2370   ! then does the appropriate I/O.
2380   !
2390      STATUS @Pm;RETURN ("VALUE":State)
2400      IF State THEN
2410   !
2420   ! Send the command to turn PM on
2430   !
2440      ELSE
2450   !
2460   ! Send the command to turn PM off
2470   !
2480      END IF
2490   SUBEND
```

```
10      ! ************************************************
20      ! Example: HPGL VIEW Viewer
30      !
40      ! This program demonstrates the use of the HPGL VIEW widget.
50      ! You can use the Menu to import any HPGL file (files with
60      ! a .gl extension).
70      !
80      ! ************************************************
90      !
100   ! Variables Used:
110   !
120   !    S$:            General-purpose string
130   !    N:             General-purpose variable
140   !    Btn:           Returns button value from dialogs
150   !    Glfile$:               Name of HPGL file to be read
160   !    Dirname$:      Directory name returned from FILE dialog
170   !
180   DIM S$[256]
190   INTEGER N,Btn,Err
200   DIM Glfile$[100],Dirname$[100]
210   !
220   ! Widget dimensions
230   !
240   INTEGER Pw,Ph,Px,Py,Iw,Ih,Hw,Hh,Hx,Hy
250   !
260   ! Variables for display scaling
270   !
280   INTEGER Dw,Dh,D(1:4)
290   !
300   ! Get display size
310   !
320   GESCAPE CRT,3;D(*)
330   Dw=D(3)-D(1)
340   Dh=D(4)-D(2)
350   !
360   Pw=Dw*.7                        ! PANEL width
370   Ph=Dh*.7                        ! PANEL height
380   Px=(Dw-Pw)/2              ! Center PANEL
390   Py=(Dh-Ph)/2
```

```
400   !
410   ! Create PANEL for HPGL widget
420   !
430   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
440   CONTROL @Main;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
450   CONTROL @Main;SET ("TITLE":" Example: HPGL Viewer")
460   CONTROL @Main;SET ("SIZE CONTROL":"RESIZE CHILDREN")
470   CONTROL @Main;SET ("MINIMIZABLE":1)
480   !
490   ! Build menu
500   !
510   ASSIGN @Menu TO WIDGET "PULLDOWN MENU";PARENT @Main
520   CONTROL @Menu;SET ("LABEL":"Menu")
530   ASSIGN @Getfile TO WIDGET "MENU BUTTON";PARENT @Menu
540   CONTROL @Getfile;SET ("LABEL":"Get HPGL File")
550   ASSIGN @Cd TO WIDGET "MENU BUTTON";PARENT @Menu
560   CONTROL @Cd;SET ("LABEL":"Change Directory")
570   ASSIGN @S TO WIDGET "MENU SEPARATOR";PARENT @Menu
580   ASSIGN @Quit TO WIDGET "MENU BUTTON";PARENT @Menu
590   CONTROL @Quit;SET ("LABEL":"Quit")
600   !
610   ! Create and size HPGL VIEW widget. (Setting RETAIN RASTER
620   ! redraws the widget quickly when overwritten by a dialog.)
630   !
640   ASSIGN @Hpgl TO WIDGET "HPGL VIEW";PARENT @Main
650   CONTROL @Hpgl;SET ("BACKGROUND":0)
660   CONTROL @Hpgl;SET ("RETAIN RASTER":0)
670   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
680   Hx=Iw*.01
690   Hy=Ih*.01
700   Hh=Ih*.98
710   Hw=Iw*.98
720   CONTROL @Hpgl;SET ("X":Hx,"Y":Hy,"WIDTH":Hw,"HEIGHT":Hh)
730   !
740   ! Set events
750   !
760   ON EVENT @Getfile,"ACTIVATED" GOSUB Gethpgl
770   ON EVENT @Cd,"ACTIVATED" GOSUB Chdir
780   ON EVENT @Quit,"ACTIVATED" GOTO Finis
790   !
800   CONTROL @Main;SET ("VISIBLE":1)
810   !
```

```
820     ! Loop and wait for input
830     !
840     LOOP
850       WAIT FOR EVENT
860     END LOOP
870     STOP
880     !
890     ! This routine gets an HPGL file and displays it
900     !
910 Gethpgl:        !
920     S$="Please enter the name of an HPGL file:"
930     DIALOG "FILE",S$,Btn;RETURN ("SELECTION":Glfile$)
940     !
950     IF Btn=0 THEN
960       CLEAR ERROR
970       ON ERROR GOSUB Seterr
980       CONTROL @Hpgl;SET ("HPGL FILE":Glfile$)
990       OFF ERROR
1000      IF ERRN<>0 THEN
1010        DIALOG "ERROR","Cannot open file/invalid HPGL file"
1020      ELSE
1030        DIALOG "INFORMATION","File read completed"
1040      END IF
1050    END IF
1060    RETURN
1070    !
1080    ! This routine changes directories
1090    !
1100 Chdir:       !
1110    S$="Please enter the name of a directory:"
1120    DIALOG "FILE",S$,Btn;RETURN ("DIRECTORY":Dirname$)
1130    CLEAR ERROR
1140    ON ERROR GOSUB Seterr
1150    MASS STORAGE IS Dirname$
1160    OFF ERROR
1170    IF ERRN<>0 THEN
1180      DIALOG "ERROR","Cannot change directory"
1190    END IF
1200    RETURN
1210    !
1220    ! Dummy routine for error traps
1230    !
```

```
1240 Seterr: ERROR RETURN
1250   !
1260 Finis:      !
1270   ASSIGN @Main TO *                  ! Deletes PANEL widget
1280   END
```

```
10       ! ******************************************************
20       ! Example: HPGL VIEW Widget
30       !
40       ! This program displays five example HPGL File Drawings.
50       ! You can bring any of the drawings to the front by clicking
60       ! the drawing.
70       !
80       ! ******************************************************
90       !
100    ASSIGN @Hpgl1 TO WIDGET "HPGL VIEW";SET
("X":0,"Y":0,"BACKGROUND":0,"TITLE":" Mechanical Drawing","HPGL
FILE":"MECH.GL","RETAIN RASTER":1)
110    CONTROL @Hpgl1;SET ("SYSTEM MENU":"Quit")
120    ON EVENT @Hpgl1,"SYSTEM MENU" GOTO Finis
130    !
140    ASSIGN @Hpgl2 TO WIDGET "HPGL VIEW";SET ("X":20,"Y":20,"TITLE":" Heat
Radiation","HPGL FILE":"GSTORE.GL","BACKGROUND":0,"RETAIN RASTER":1)
150    CONTROL @Hpgl2;SET ("SYSTEM MENU":"Quit")
160    ON EVENT @Hpgl2,"SYSTEM MENU" GOTO Finis
170    !
180    ASSIGN @Hpgl3 TO WIDGET "HPGL VIEW";SET ("X":40,"Y":40,"TITLE":" Ice Berg
Flows","HPGL FILE":"ICE.GL","BACKGROUND":6,"RETAIN RASTER":1)
190    CONTROL @Hpgl3;SET ("SYSTEM MENU":"Quit")
200    ON EVENT @Hpgl3,"SYSTEM MENU" GOTO Finis
210    !
220    ASSIGN @Hpgl4 TO WIDGET "HPGL VIEW";SET ("X":60,"Y":60,"TITLE":" Topographical
Map","HPGL FILE":"CONTOR.GL","BACKGROUND":0,"RETAIN RASTER":1)
230    CONTROL @Hpgl4;SET ("SYSTEM MENU":"Quit")
240    ON EVENT @Hpgl4,"SYSTEM MENU" GOTO Finis
250    !
260    ASSIGN @Hpgl5 TO WIDGET "HPGL VIEW";SET ("X":80,"Y":80,"TITLE":" Space
Shuttle","HPGL FILE":"SHUTTLE.GL","BACKGROUND":0,"RETAIN RASTER":1)
270    CONTROL @Hpgl5;SET ("SYSTEM MENU":"Quit")
280    ON EVENT @Hpgl5,"SYSTEM MENU" GOTO Finis
290    !
300    LOOP
310      WAIT FOR EVENT
320    END LOOP
330 Finis:    !
340    ASSIGN @Hpgl1 TO *              ! Delete HPGL VIEW widget #1
350    ASSIGN @Hpgl2 TO *              ! Delete HPGL VIEW widget #2
```

```
360     ASSIGN @Hpgl3 TO *              ! Delete HPGL VIEW widget #3
370     ASSIGN @Hpgl4 TO *              ! Delete HPGL VIEW widget #4
380     ASSIGN @Hpgl5 TO *              ! Delete HPGL VIEW widget #5
390     END
```

```
10       ! ********************************************************
20       ! Example: HPGL VIEW Widget in PANEL
30       !
40       ! This program first displays a mechanical drawing and
50       ! then displays a "Parts Dimension" display of the drawing.
60       ! You can use the scrollbars to look at any part of the
70       ! drawing.
80       !
90       ! ********************************************************
100    !
110    INTEGER Screen(1:4),S_width,S_height,Width,Height
120    GESCAPE CRT,3;Screen(*)
130    ASSIGN @Panel TO WIDGET "PANEL";SET
("X":0,"Y":0,"WIDTH":Screen(3)+1,"HEIGHT":Screen(4)+1,"TITLE":" Part Dimensions")
140    CONTROL @Panel;SET ("SYSTEM MENU":"Quit")
150    ON EVENT @Panel,"SYSTEM MENU" GOTO Finis
160    STATUS @Panel;RETURN ("INSIDE WIDTH":S_width,"INSIDE HEIGHT":S_height)
170    ASSIGN @Hpgl1 TO WIDGET "HPGL VIEW";SET
("BACKGROUND":0,"X":0,"Y":0,"WIDTH":S_width,"HEIGHT":S_height,"BORDER":0,"HPGL
FILE":"MECH.GL","RETAIN RASTER":1),PARENT @Panel
180    Width=(Screen(3)+1)/3
190    Height=(Screen(4)+1)/3
200    CONTROL @Panel;SET
("X":(Screen(3)-Width)/2,"Y":(Screen(4)-Height)/2,"WIDTH":Width,"HEIGHT":Height)
210    CONTROL @Panel;SET ("BACKGROUND":0,"SIZE
CONTROL":"SCROLLABLE","SCROLL WIDTH":S_width,"SCROLL HEIGHT":S_height)
220    ON EVENT @Panel,"RESIZED" GOTO Looping
230 Looping: LOOP
240      WAIT FOR EVENT
250    END LOOP
260 Finis: END
```

```
10    ! ************************************************************************
20    ! Example: Hammer Game
30    !
40    ! The object of this game is to click the pushbutton before it moves.
50    ! How often the button moves and how large it is are adjustable with
60    ! sliders. Every time the button is successfully clicked, the score
70    ! increases by one point.
80    !
90    ! ************************************************************************
100   !
110   ! Move the button every two seconds
120   !
130   Speed=2
140   ASSIGN @Speed TO WIDGET "SLIDER";SET
("ORIENTATION":"HORIZONTAL","X":415,"Y":10,"TITLE":" Speed
","LOGARITHMIC":1,"VALUE":Speed,"WIDTH":250,"HEIGHT":60)
150   CONTROL @Speed;SET ("MAXIMUM":2,"MINIMUM":.2)
160   ON EVENT @Speed,"DONE",3 GOSUB Speed_change
170   !
180   ! Set button size to 25 pixels on a side
190   !
200   Size=25
210   ASSIGN @Size TO WIDGET "SLIDER";SET
("ORIENTATION":"HORIZONTAL","X":150,"Y":10,"TITLE":" Size
","LOGARITHMIC":1,"VALUE":Size,"WIDTH":250,"HEIGHT":60)
220   CONTROL @Size;SET ("MAXIMUM":100,"MINIMUM":10)
230   ON EVENT @Size,"DONE",3 GOSUB Size_change
240   !
250   ! Set up the score
260   !
270   Score=0
280   ASSIGN @Score TO WIDGET "LABEL";SET ("VALUE":VAL$(Score),"TITLE":"
Hits","X":10,"Y":10,"WIDTH":125,"HEIGHT":60)
290   !
300   ! Draw the moving widget
310   !
320   ASSIGN @Button TO WIDGET "PUSHBUTTON";SET
("WIDTH":50,"HEIGHT":30,"LABEL":"","TITLE":"","X":300)
330   !
340   ! When the button is pushed, score a hit
350   !
```

```
360    ON EVENT @Button,"ACTIVATED",3 GOSUB Hit
370    !
380    ! Provide a button to stop the game
390    !
400    ASSIGN @Stop TO WIDGET "PUSHBUTTON";SET
("RESIZABLE":0,"WIDTH":125,"HEIGHT":30,"LABEL":"Quit","TITLE":"","X":10,"Y":80)
410    ON EVENT @Stop,"ACTIVATED",3 GOTO Stop_game
420    !
430    ! Move the button on a regular basis
440    !
450    ON CYCLE Speed GOSUB Move_button
460    LOOP
470       WAIT FOR EVENT
480    END LOOP
490 Hit:                                    ! Button was hit
500    Score=Score+1
510    CONTROL @Score;SET ("VALUE":VAL$(Score))
520    GOSUB Move_button                    ! Too easy to hit again
530    RETURN
540 Move_button:                            ! Move the button to new location
550    !
560    ! The scaling should depend on the size of the display
570    !
580    CONTROL @Button;SET ("X":RND*500+60,"Y":RND*400)
590    RETURN
600 Speed_change:                           ! Speed slider was moved
610    !
620    STATUS @Speed;RETURN ("VALUE":Speed)
630    ON CYCLE Speed GOSUB Move_button
640    RETURN
650 Size_change:                            ! Size slider was moved
660    STATUS @Size;RETURN ("VALUE":Size)
670    CONTROL @Button;SET ("WIDTH":Size,"HEIGHT":Size)
680    RETURN
690 Stop_game:   !
700    OFF CYCLE
710    ASSIGN @Button TO *
720    ASSIGN @Size TO *
730    ASSIGN @Speed TO *
740    ASSIGN @Score TO *
750    ASSIGN @Stop TO *
760    STOP
```

```
770     END
```

```
10      ! ************************************************
20      ! Example: INFORMATION Dialog
30      !
40      ! This program provides an INFORMATION dialog.
50      !
60      ! ************************************************
70      !
80      DIM P$[26]
90      P$="Here is the information"
100   DIALOG "INFORMATION",P$;SET ("TITLE":" Example: INFORMATION Dialog")
110   END
```

```
10      ! ************************************************************************
20      ! Example: Ice Cream Sundae
30      !
40      ! This program is an example of the use of the LIST widget.
50      ! It displays a panel that contains two LIST widgets -- one in which
60      ! the MULTISELECT attribute is not set, and another in which the
70      ! MULTISELECT attribute is set.
80      !
90      ! If MULTISELECT is not set (0), only one element can be selected
100   ! from the widget. If you click on a LIST widget in that mode and then
110   ! read it, you will get the number of the entry into the LIST.
120   !
130   ! If MULTISELECT is set (1), several elements can be selected from
140   ! the widget. You give the widget an array of a size that matches the
150   ! number of elements in the LIST. When you click on the LIST entries,
160   ! as they are selected their corresponding array entries are set to 1.
170   ! The unselected array entries are set to 0.   (If you click again on a
180   ! selected entry, its array value is cleared back to 0.)
190   !
200   ! The MULTISELECT:0 LIST allows you to select a flavor of ice cream
210   ! while the MULTISELECT:1 LIST allows you to select all the toppings
220   ! you like.   When you are done, you press the "GIMME!" button to get
230   ! your selection. (Actually, all you get when you press the button is
240   ! an INFORMATION DIALOG that tells you you are out of luck.)
250   !
260   ! ************************************************************************
270   !
280   CLEAR SCREEN
290   OPTION BASE 1
300   !
310   ! Set color values:
320   !
330   INTEGER Black,White,Red,Yellow,Green,Blue,Magenta
340   Black=0
350   White=1
360   Red=2
370   Yellow=3
380   Green=4
390   Blue=6
400   Magenta=7
410   !
```

```
420   ! Some variables:
430   !
440   !    Buffer$:          Used to display values from MULTISELECT:1 list
450   !    Select(*):    Gets status of selections from MULTISELECT list
460   !    A(*):         Gets values from GESCAPE statement to find display size.
470   !    Nlines:           Gets number of lines of text on display
480   !    N:            General-purpose variable
490   !
500   DIM Buffer$[32]
510   INTEGER Select(9),A(6),Nlines,N
520   !
530   ! Get display resolution
540   !
550   REAL Dw,Dh,Vh
560   GESCAPE CRT,3;A(*)
570   Dw=A(3)-A(1)+1
580   Dh=A(4)-A(2)+1
590   STATUS CRT,13;Nlines
600   Vh=Dh*(1-6/Nlines)
610   !
620   ! Set up dimensions and location for main panel
630   !
640   REAL Pw,Ph,Px,Py,Iw,Ih
650   Pw=340
660   Ph=310
670   Px=(Dw-Pw)/2
680   Py=(Vh-Ph)/2
690   !
700   ! Set up the main panel
710   !
720   ASSIGN @P TO WIDGET "PANEL";SET ("VISIBLE":0)
730   CONTROL @P;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
740   CONTROL @P;SET ("TITLE":" Coyote's Ice Cream Emporium")
750   CONTROL @P;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
760   STATUS @P;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
770   !
780   ! The panel contains the two list widgets, with a title
790   ! LABEL and a value LABEL for both. There are also two
800   ! PUSHBUTTONs, one to "GIMME" your ice cream, and another
810   ! to exit the program. The following variables assign
820   ! assign sizes to these widgets and their locations in
830   ! the panel.
```

```
840   !
850   ! These assignments are arranged interdependently, so if
860   ! you change one the others are adjusted automatically.
870   !
880   REAL Gaph,Btnw,Lblh,Listw,Listh,C1,C2,R1,R2,R3,R4,R5
890   !
900   Gaph=Ih*.02                    ! Vertical gap
910   Listw=Iw*.44                   ! Width of LISTS (and corresponding LABELs)
920   Listh=Ih*.6                          ! Height of LIST widgets
930   Btnw=Iw*.2                     ! Width of the two buttons
940   Lblh=Ih*.1                           ! Height of the four labels
950   !
960   C1=(Iw/2-Listw)/2                    ! Column 1 is the location for the
MULTISELECT:0 LIST
970   C2=(Iw/4)-Btnw/2               ! Column 2 is the location for the GIMME button
980   C3=Iw/2+C1                     ! Column 3 is the location for the MULTISELECT:1
widget
990   C4=Iw/2+C2                     ! Column 4 is the location of the EXIT button
1000  !
1010  R1=Gaph                              ! Row 1 is the location for the title LABELs
1020  R2=R1+Lblh+Gaph                ! Row 2 is the location for the LIST widgets
1030  R3=R2+Listh+Gaph               ! Row 3 is the location of the value LABELs
1040  R4=R3+Lblh                     ! Row 4 is the bottom of the value LABELs
1050  R5=R4+((Ih-R4)-Lblh)/2         ! Row 5 is location of the buttons
1060  !
1070  DIM Menu$(9)[20],Topping$(9)[20]     ! Entry arrays for LIST widgets
1080  !
1090  ! Entry array for MULTISELECT:0 widget
1100  !
1110  Menu$(1)="   VANILLA"
1120  Menu$(2)="   CHOCOLATE"
1130  Menu$(3)="   STRAWBERRY"
1140  Menu$(4)="   NEAPOLITAN"
1150  Menu$(5)="   FUDGE RIPPLE"
1160  Menu$(6)="   HEATH BAR"
1170  Menu$(7)="   BUTTERFINGER"
1180  Menu$(8)="   MOCHA"
1190  Menu$(9)="   PEPPERMINT"
1200  !
1210  ! Entry array for MULTISELECT:1 widget
1220  !
1230  Topping$(1)="   NUTS"
1240  Topping$(2)="   BANANAS"
```

```
1250  Topping$(3)="   CHOCOLATE"
1260  Topping$(4)="   HOT FUDGE"
1270  Topping$(5)="   STRAWBERRIES"
1280  Topping$(6)="   WHIPPED CREAM"
1290  Topping$(7)="   SPRINKLES"
1300  Topping$(8)="   CHERRY"
1310  Topping$(9)="   BUTTERSCOTCH"
1320  !
1330  ! The following variables provide attributes for
1340  ! the INFORMATION dialog. Since you must declare
1350  ! everything for a dialog in the statement that
1360  ! creates it, it is convenient to define attributes
1370  ! by putting the attribute names in a string array
1380  ! and the values in a matching array. You can then
1390  ! use the arrays to set up the values in a dialog
1400  ! declaration without having to put everything on
1410  ! one line.
1420  !
1430  ! One thing to remember is that you can match your
1440  ! attribute array to a numeric or string value array -
1450  ! but you CANNOT mix the types in the value array.
1460  ! So, if you want to set attributes with numeric or
1470  ! string values, you must segregate the attributes
1480  ! into separate attribute arrays and then use them
1490  ! attribute arrays and then use them with the
1500  ! appropriate value arrays.
1510  !
1520  ! Since all the attributes but one invoked with the
1530  ! DIALOG command have numeric values, only one array
1540  ! is set up. (A font is set as well, but that is
1550  ! done in the DIALOG invocation itself.)
1560  !
1570  ! The X and Y origin of the DIALOG is relative to the
1580  ! 0,0 coordinate of the display, NOT to the parent
1590  ! widget. Declaring a dialog box with a parent has
1600  ! the effect that a user will not be able to click
1610  ! the parent widget back over the top of the DIALOG.
1620  !
1630  DIM Ds$(6)[12],Pr$[32],F$[16]
1640  DIM Dv(6)
1650  !
1660  Ds$(1)="WIDTH"
```

```
1670  Dv(1)=Pw*.75
1680  Ds$(2)="HEIGHT"
1690  Dv(2)=Ph*.65
1700  Ds$(3)="X"
1710  Dv(3)=Px+(Pw-Dv(1))/2
1720  Ds$(4)="Y"
1730  Dv(4)=Py+(Ph-Dv(2))/2
1740  Ds$(5)="BACKGROUND"
1750  Dv(5)=Blue
1760  Ds$(6)="PEN"
1770  Dv(6)=White
1780  Pr$="Sorry, all out!"
1790  F$="12 BY 14,BOLD"
1800  !
1810  ! Set up the label for the MULTISELECT:0 LIST. The border is
1820  ! turned off on the LABELs, so they will not look like buttons.
1830  !
1840  ASSIGN @Lbl1 TO WIDGET "LABEL";PARENT @P
1850  CONTROL @Lbl1;SET ("X":C1,"Y":R1,"WIDTH":Listw,"HEIGHT":Lblh)
1860  CONTROL @Lbl1;SET ("BACKGROUND":White,"PEN":Black)
1870  CONTROL @Lbl1;SET ("BORDER":0)
1880  CONTROL @Lbl1;SET ("FONT":"14 BY 14,BOLD")
1890  CONTROL @Lbl1;SET ("VALUE":"PICK YOUR FLAVOR:")
1900  !
1910  ! Set up the MULTISELECT:0 LIST
1920  !
1930  ASSIGN @L1 TO WIDGET "LIST";PARENT @P
1940  CONTROL @L1;SET ("FONT":"10 BY 16,BOLD")
1950  CONTROL @L1;SET ("X":C1,"Y":R2,"WIDTH":Listw,"HEIGHT":Listh)
1960  CONTROL @L1;SET ("BACKGROUND":White,"PEN":Blue)
1970  CONTROL @L1;SET ("ITEMS":Menu$(*))
1980  !
1990  ! Set up the MULTISELECT:0 value LABEL
2000  !
2010  ASSIGN @Lbl2 TO WIDGET "LABEL";PARENT @P
2020  CONTROL @Lbl2;SET ("X":C1,"Y":R3,"WIDTH":Listw,"HEIGHT":Lblh)
2030  CONTROL @Lbl2;SET ("BACKGROUND":White,"PEN":Blue)
2040  CONTROL @Lbl2;SET ("BORDER":0)
2050  CONTROL @Lbl2;SET ("FONT":"10 BY 16,BOLD")
2060  CONTROL @Lbl2;SET ("VALUE":"")
2070  !
2080  ! Set up the MULTISELECT:1 title LABEL
```

```
2090  !
2100  ASSIGN @Lbl3 TO WIDGET "LABEL";PARENT @P
2110  CONTROL @Lbl3;SET ("X":C3,"Y":R1,"WIDTH":Listw,"HEIGHT":Lblh)
2120  CONTROL @Lbl3;SET ("BACKGROUND":White,"PEN":Black)
2130  CONTROL @Lbl3;SET ("BORDER":0)
2140  CONTROL @Lbl3;SET ("FONT":"14 BY 14,BOLD")
2150  CONTROL @Lbl3;SET ("VALUE":"SELECT YOUR TOPPINGS:")
2160  !
2170  ! Set up the MULTISELECT:1 LIST
2180  !
2190  ASSIGN @L2 TO WIDGET "LIST";PARENT @P
2200  CONTROL @L2;SET ("FONT":"10 BY 16,BOLD")
2210  CONTROL @L2;SET ("X":C3,"Y":R2,"WIDTH":Listw,"HEIGHT":Listh)
2220  CONTROL @L2;SET ("BACKGROUND":White,"PEN":Blue)
2230  CONTROL @L2;SET ("ITEMS":Topping$(*))
2240  CONTROL @L2;SET ("MULTISELECT":1)
2250  !
2260  ! Set up the MULTISELECT:1 value LABEL
2270  !
2280  ASSIGN @Lbl4 TO WIDGET "LABEL";PARENT @P
2290  CONTROL @Lbl4;SET ("X":C3,"Y":R3,"WIDTH":Listw,"HEIGHT":Lblh)
2300  CONTROL @Lbl4;SET ("BACKGROUND":White,"PEN":Blue)
2310  CONTROL @Lbl4;SET ("BORDER":0)
2320  CONTROL @Lbl4;SET ("FONT":"10 BY 16, BOLD")
2330  CONTROL @Lbl4;SET ("VALUE":"0 0 0 0 0 0 0 0 0")
2340  !
2350  ! Set up the GIMME button
2360  !
2370  ASSIGN @B1 TO WIDGET "PUSHBUTTON";PARENT @P
2380  CONTROL @B1;SET ("X":C2,"Y":R5,"WIDTH":Btnw,"HEIGHT":Lblh)
2390  CONTROL @B1;SET ("BACKGROUND":Red,"PEN":Black)
2400  CONTROL @B1;SET ("FONT":"10 BY 12")
2410  CONTROL @B1;SET ("LABEL":"GIMME!")
2420  !
2430  ! Set up the EXIT button
2440  !
2450  ASSIGN @B2 TO WIDGET "PUSHBUTTON";PARENT @P
2460  CONTROL @B2;SET ("X":C4,"Y":R5,"WIDTH":Btnw,"HEIGHT":Lblh)
2470  CONTROL @B2;SET ("BACKGROUND":Green,"PEN":Black)
2480  CONTROL @B2;SET ("FONT":"10 BY 12")
2490  CONTROL @B2;SET ("LABEL":"EXIT")
2500  !
```

```
2510   ! Turn on the panel and show the widgets
2520   !
2530   CLEAR SCREEN
2540   CONTROL @P;SET ("VISIBLE":1)
2550   !
2560   ! Set events and wait for an event to happen
2570   !
2580   ON EVENT @L1,"SELECTION" GOSUB Onesel          ! MULTISELECT:0 LIST select
2590   ON EVENT @L2,"SELECTION" GOSUB Multisel        ! MULTISELECT:1 LIST select
2600   ON EVENT @B1,"ACTIVATED" GOSUB Icecream              ! Click on GIMME
button
2610   ON EVENT @B2,"ACTIVATED" GOTO Finis            ! Click on EXIT button
2620   !
2630   LOOP
2640     WAIT FOR EVENT
2650   END LOOP
2660   !
2670   ! ********** End of Main Program **************************
2680   !
2690   ! This routine handles a mouse click on the MULTISELECT:0 LIST.
2700   ! list. This routine gets the SELECTION value from the LIST
2710   ! widget and then puts it into the corresponding value LABEL.
2720   !
2730   ! The SELECTION value is the index into the entry array. The
2740   ! index assumes a base array index of 0 even if OPTION BASE 1
2750   ! is set (as it is in this program). This means that if you
2760   ! selected the entry corresponding to array element 1, you
2770   ! would get a value back of 0.
2780   !
2790 Onesel:!
2800   STATUS @L1;RETURN ("SELECTION":N)
2810   CONTROL @Lbl2;SET ("VALUE":N)
2820   RETURN
2830 !
2840 ! This is the handler routine for the MULTISELECT:1 LIST.
2850 ! This routine gets the SELECTION array and then lists ALL
2860 ! the values in the array in the corresponding LABEL.
2870 ! The array entry will be 1 for a selected entry, and 0
2880 ! for an unselected entry.
2890 !
2900 Multisel:!
2910   STATUS @L2;RETURN ("SELECTION":Select(*))
```

```
2920   Buffer$=""
2930   FOR N=1 TO 9
2940      Buffer$=Buffer$&VAL$(Select(N))&" "
2950   NEXT N
2960   CONTROL @Lbl4;SET ("VALUE":Buffer$)
2970   RETURN
2980 !
2990 ! This is the handler for the GIMME button. It displays a
3000 ! a DIALOG and tells the user that he or she is out of luck.
3010 ! (It is for display only and for an example of using a
3020 ! DIALOG box.) A timeout is set so that the DIALOG
3030 ! disappears after 5 seconds if the user does nothing.
3040 !
3050 Icecream: !
3060   BEEP 10000,.01
3070   DIALOG "INFORMATION",Pr$;SET (Ds$(*):Dv(*),"FONT":F$),TIMEOUT 5
3080   RETURN
3090 !
3100 ! This code closes the main panel, clears the screen, and displays "DONE"
3110 !
3120 Finis:!
3130   ASSIGN @P TO *                          ! Closes widget
3140   CLEAR SCREEN
3150   DISP "DONE"
3160   END
```

```
10    ! ************************************
20    ! Example: KEYPAD Dialog
30    !
40    ! This program produces a KEYPAD dialog and
50    ! displays the number entered by the user.
60    !
70    ! ************************************
80    !
90     DIM P$[25]
100    INTEGER Btn
110    P$="Please enter a number:"
120    DIALOG "KEYPAD",P$,Btn;SET ("TITLE":" Example: KEYPAD Dialog"),RETURN
("VALUE":X)
130    SELECT Btn
140    CASE 0
150       DISP " Number entered:",VAL$(X)
160    CASE 1
170       DISP " Numeric input canceled."
180    END SELECT
190    END
```

```
10      ! ********************************************************
20      ! Example: KEYPAD Widget
30      !
40      ! This program uses the KEYPAD widgt to generate
50      ! a keypad you can use to enter numbers.
60      !
70      ! ********************************************************
80      !
90      ASSIGN @Keypad TO WIDGET "KEYPAD";SET ("REAL NOTATION":"FIXED")
100   CONTROL @Keypad;SET ("TITLE":" Example: KEYPAD Widget")
110   CONTROL @Keypad;SET ("MINIMUM":0,"MAXIMUM":1000)
120   CONTROL @Keypad;SET ("X":100,"Y":50,"WIDTH":250,"HEIGHT":300)
130   CONTROL @Keypad;SET ("CHECK FOR DONE":1)
140   CONTROL @Keypad;SET ("SYSTEM MENU":"Quit")
150   ON EVENT @Keypad,"RETURN" GOSUB Get_number
160   ON EVENT @Keypad,"DONE" GOSUB Get_number
170   ON EVENT @Keypad,"SYSTEM MENU" GOTO Finis
180   LOOP
190     WAIT FOR EVENT
200   END LOOP
210 Get_number: STATUS @Keypad;RETURN ("MODIFIED":New_number,"VALUE":Value)
220   IF New_number THEN
230     DISP "New number: ";Value
240     CONTROL @Keypad;SET ("MODIFIED":0)
250   END IF
260   RETURN
270 Finis:    !
280   ASSIGN @Keypad TO *             ! Delete KEYPAD widget
290   END
```

```
10      ! ******************************************************
20      ! Example: LABEL Widget
30      !
40      ! This program generates a LABEL widget.
50      !
60      ! ******************************************************
70      !
80      INTEGER N
90      DIM S$[256]
100   ASSIGN @L TO WIDGET "LABEL";SET ("VISIBLE":0)
110   CONTROL @L;SET ("COLUMNS":28,"ROWS":8,"TITLE":" Example: LABEL Widget")
120   CONTROL @L;SET ("X":100,"Y":50,"JUSTIFICATION":"TOP,LEFT","WORD WRAP":1)
130   CONTROL @L;SET ("SYSTEM MENU":"Quit")
140    !
150   FOR N=1 TO 8
160     S$=S$&" ITEM "&VAL$(N)&": VALUE "&VAL$(N)&" "
170     IF N<8 THEN S$=S$&CHR$(10)
180   NEXT N
190   CONTROL @L;SET ("VALUE":S$,"VISIBLE":1)
200   ON EVENT @L,"SYSTEM MENU" GOTO Finis
210   LOOP
220     WAIT FOR EVENT
230   END LOOP
240   !
250 Finis: !
260   ASSIGN @L TO *      ! Deletes LABEL widget
270   END
```

```
10      ! ********************************************************
20      ! Example: LIMITS Widget
30      !
40      ! This program generates a LIMITS widget. In the program,
50      ! the value is increased from 1 to 99. The LOW limit is
60      ! set to 20 and the HIGH limit is set to 80. Since the
70      ! ALARM RANGES is set to "OUTSIDE" ("O"), and ALARM TYPE
80      ! is "BEEP", the alarm sounds when the pointer is in the
90      ! red ranges.
100     !
110     ! ********************************************************
120     !
130     DIM Value(1:100)
140     ASSIGN @Lim TO WIDGET "LIMITS"
150     CONTROL @Lim;SET ("TITLE":" Example: LIMITS Widget")
160     CONTROL @Lim;SET ("LOW LIMIT":20,"HIGH LIMIT":80)
170     CONTROL @Lim;SET ("ALARM TYPE":"BEEP","ALARM RANGES":"O")
180     CONTROL @Lim;SET ("MINIMUM":0,"MAXIMUM":100)
190     CONTROL @Lim;SET ("SYSTEM MENU":"Quit")
200     ON EVENT @Lim,"SYSTEM MENU" GOTO Finis
210     LOOP
220       FOR I=1 TO 99
230         Value(I)=I
240         CONTROL @Lim;SET ("VALUE":Value(I))
250         WAIT .1
260       NEXT I
270     END LOOP
280 Finis:    !
290     ASSIGN @Lim TO *        ! Delete LIMITS widget
300     END
```

```
10      ! ******************************************************
20      ! Example: LIST Dialog
30      !
40      ! This program shows a way to use the MULTISELECT attribute.
50      ! For example, if you click on COSMOPOLITAN, TIME, and SPORTS
60      ! ILLUSTRATED, then click on OK the program displays
70      ! OK 1 0 0 1 0 1 0. The '1's correspond to the selected items.
80      !
90      ! ******************************************************
100   !
110   CLEAR SCREEN
120   INTEGER Btn,Sel(1:7),V(1:2)
130   DIM L$(1:7)[20],P$[40],A$(1:2)[16],S$[16]
140   DATA "COSMOPOLITAN","ENQUIRER","DISCOVER","TIME"
150   DATA "HEALTH","SPORTS ILLUSTRATED","NEW YORKER"
160   READ L$(*)
170      !
180   DATA "MULTISELECT","COLUMNS",1,30, "SELECTION"
190   READ A$(*),V(*),S$
200   P$="Which magazines do you read?"
210   DIALOG "LIST",P$,Btn;SET ("TITLE":" Example: LIST Dialog","ITEMS":L$(*),A$
(*):V(*)),RETURN (S$:Sel(*))
220   PRINT " Button        Selection"
230   PRINT
240   IF Btn=0 THEN
250     PRINT USING "8A,3X,7(D,X)";" OK",Sel(*)
260   ELSE
270     PRINT USING "8A,3X,7(D,X)";"Cancel",Sel(*)
280   END IF
290   END
```

```
10     ! *******************************************************
20     ! Example: LIST Widget
30     !
40     ! This program creates a LIST widget with a defined
50     ! list of animals. When the user selects an animal's
60     ! name, an INFORMATION dialog appears that displays
70     ! the name of the animal selected.
80     !
90     ! *******************************************************
100    !
110    DIM L$(1:5)[26]
120    INTEGER N
130    !
140    DATA "   Aardvark","   Sidewinder","   Kiwi","   Pangolin","   Marmoset"
150    READ L$(*)
160    !
170    ASSIGN @List TO WIDGET "LIST";SET ("SYSTEM MENU":"Quit")
180    CONTROL @List;SET ("X":100,"Y":50,"WIDTH":400,"BACKGROUND":1)
190    CONTROL @List;SET ("TITLE":"Example: LIST Widget - Select Your Favorite Animal")
200    CONTROL @List;SET ("ITEMS":L$(*))
210    !
220    ON EVENT @List,"SELECTION" GOSUB Handler
230    ON EVENT @List,"SYSTEM MENU" GOTO Finis
240    !
250    LOOP
260      WAIT FOR EVENT
270    END LOOP
280    STOP
290    !
300 Handler: !
310    STATUS @List;RETURN ("SELECTION":Sel)
320    DIALOG "INFORMATION",L$(Sel+1);SET ("TITLE":" Animal Selected")
330    RETURN
340    !
350 Finis: !
360    ASSIGN @List TO *        ! Delete LIST widget
370    END
```

```
10      ! ************************************************************
20      ! Example: Lissajous Patterns
30      !
40      ! This example plots one sine wave vs. another. The phase
50      ! and ratio of frequencies are varied. The resulting curves
60      ! are Lissajous curves.
70      !
80      ! ************************************************************
90      !
100     ! Set up an XY display for the curves
110     ! Most of the attributes are programmed outside the loop
120     !
130     ASSIGN @Lissajous TO WIDGET "XY GRAPH";SET ("TITLE":" Example: Lissajous
Patterns","TRACE COUNT":1,"POINT
CAPACITY":512,"HEIGHT":250,"WIDTH":250,"X":100,"Y":100)
140     CONTROL @Lissajous;SET ("TRACE BACKGROUND":9,"TRACE PEN":0)
150     CONTROL @Lissajous;SET ("SYSTEM MENU":"Quit")
160     ON EVENT @Lissajous,"SYSTEM MENU" GOTO Finis
170     !
180     ! Set the limits in the graph to -1 to +1 for both axes
190     !
200     CONTROL @Lissajous;SET ("CURRENT AXIS":"X","ORIGIN":-1,"RANGE":2,"SHOW
NUMBERING":0,"SHOW TICKS":0)
210     CONTROL @Lissajous;SET ("CURRENT AXIS":"Y","ORIGIN":-1,"RANGE":2,"SHOW
NUMBERING":0,"SHOW TICKS":0)
220     !
230     ! Two string displays which show the current values of the loop veriables
240     !
250     ASSIGN @Show_phase TO WIDGET "STRING";SET ("TITLE":"    Phase","X":50,"Y":10)
260     ASSIGN @Show_ratio TO WIDGET "STRING";SET ("TITLE":"    Ratio","X":250,"Y":10)
270     !
280     ! Calculate the reference sine wave and set the X data to it
290     !
300     INTEGER I,N
310     N=50
320     ALLOCATE F1(N),F2(N)
330     ALLOCATE Fbig(0:N*2)
340     FOR I=0 TO N
350       F1(I)=SIN(I*2*PI/N)
360     NEXT I
370     CONTROL @Lissajous;SET ("X DATA":F1(*))
```

```
380   !
390   ! The Ratio variable is used the vary the ratio of the frequencies
400   ! of the two sine waves
410   !
420   LOOP
430     FOR Ratio=1 TO 5
440       Tmp1=(2*PI/N)*Ratio
450       FOR I=0 TO N*2
460         Fbig(I)=SIN(I*Tmp1)
470       NEXT I
480       CONTROL @Show_ratio;SET ("VALUE":VAL$(Ratio))
490   !
500   ! This loop offsets the phase of the second sine wave
510   !
520       FOR Phase=0 TO N-1
530         CONTROL @Show_phase;SET ("VALUE":VAL$(Phase))
540         Start=Phase/360*N
550         Start=Phase
560         MAT F2=Fbig(Start:Start+N)
570         CONTROL @Lissajous;SET ("Y DATA":F2(*))
580         WAIT .1
590       NEXT Phase
600     NEXT Ratio
610   END LOOP
620 Finis:    !
630   ASSIGN @Lissajous TO *              ! Delete XY GRAPH widget
640   END
```

```
10      ! ********************************************************
20      ! Example: METER Widget
30      !
40      ! This program generates a METER widget with a 360 degree
50      ! scale. When the pointer is within the HIGH and LOW
60      ! LIMITS areas, beeps are generated.
70      !
80      ! ********************************************************
90      !
100     INTEGER N,M
110     ASSIGN @Meter TO WIDGET "METER"
120     CONTROL @Meter;SET ("TITLE":" Example: METER Widget")
130     CONTROL @Meter;SET ("X":50,"Y":25,"WIDTH":250,"HEIGHT":250)
140     CONTROL @Meter;SET ("ALARM RANGES":"LOW,HIGH","ALARM TYPE":"BEEP")
150     CONTROL @Meter;SET ("LOW LIMIT":10,"HIGH LIMIT":90,"SWEEP ANGLE":360)
160     CONTROL @Meter;SET ("MIDDLE PEN":0,"NEEDLE PEN":6,"NEEDLE WIDTH":2)
170     CONTROL @Meter;SET ("SYSTEM MENU":"Quit")
180     ON EVENT @Meter,"SYSTEM MENU" GOTO Finis
190     !
200     FOR M=1 TO 3
210       FOR N=1 TO 100
220         WAIT .1
230         CONTROL @Meter;SET ("VALUE":N)
240       NEXT N
250     NEXT M
260     !
270 Finis:    !
280     ASSIGN @Meter TO *                ! Delete METER widget
290     END
```

```
10      ! ******************************************************
20      ! Example: METER/SLIDER Widgets
30      !
40      ! This program generates a METER and a SLIDER widget.
50      ! As you change the value of the SLIDER widget by moving
60      ! the slider bar, the changed value is displayed on the
70      ! METER widget.
80      !
90      ! ******************************************************
100   !
110   ASSIGN @Meter TO WIDGET "METER"
120   CONTROL @Meter;SET ("X":10,"Y":50,"TITLE":" METER","BACKGROUND":1)
130   CONTROL @Meter;SET ("HEIGHT":225)
140   CONTROL @Meter;SET ("SYSTEM MENU":"Quit")
150   !
160   ASSIGN @Slider TO WIDGET "SLIDER"
170   CONTROL @Slider;SET ("X":250,"Y":50,"WIDTH":150,"TITLE":" SLIDER")
180   CONTROL @Slider;SET ("BACKGROUND":9)
190   CONTROL @Slider;SET ("SYSTEM MENU":"Quit")
200   !
210   ON EVENT @Meter,"SYSTEM MENU" GOTO Finis
220   ON EVENT @Slider,"SYSTEM MENU" GOTO Finis
230   ON EVENT @Slider,"CHANGED" GOSUB Event_handler
240   !
250   LOOP
260     WAIT FOR EVENT
270   END LOOP
280   !
290 Event_handler:   !
300   STATUS @Slider;RETURN ("VALUE":Value)
310   CONTROL @Meter;SET ("VALUE":Value)
320   RETURN
330   !
340 Finis:    !
350   ASSIGN @Meter TO *              ! Delete METER widget
360   ASSIGN @Slider TO *             ! Delete SLIDER widget
370   END
```

```
10    ! ********************************************************************
20    ! Example: Menu System
30    !
40    ! This program demonstrates how to build a pulldown menu system.
50    ! It builds a panel with a menu and a printer widget to tell you
60    ! the menu entry you clicked.
70    !
80    ! ********************************************************************
90    !
100   INTEGER Black,White,Red,Yellow,Green,Blue,Cyan,Magenta
110   DATA 0,1,2,3,4,5,6,7
120   READ Black,White,Red,Yellow,Green,Blue,Cyan,Magenta
130   !
140   ! Define some variables:
150   !
160   !     S$:                         General-purpose string variable
170   !     M$(*):                            System menu elements
180   !     State:                            Gets state of MENU TOGGLE widgets
190   !     N:                          General-purpose INTEGER variable
200   !     D(*):                       Array to get hard-clip values
210   !     Dw,Dh:                            Display dimensions
220   !     X,Y:                        Location of main PANEL
230   !     Panelwidth, Panelheight:    Dimensions for main PANEL
240   !     Prx,Pry,Prwidth,Prheight:   Location & dimensions of main PANEL
250   !     Ih,Iw:                            Interior dimensions of PANEL
260   !
270   ! The main panel is set up so it automatically scales to the
280   ! entire display, except for the DISP, INPUT, and softkeys lines.
290   !
300   DIM S$[80],M$(0:2)[80]
310   INTEGER State,N,D(1:4)
320   INTEGER Dw,Dh,X,Y
330   INTEGER Panelwidth,Panelheight,Prx,Pry,Prwidth,Prheight,Iw,Ih
340   !
350   GESCAPE CRT,3;D(*)
360   Dw=D(3)-D(1)+1
370   Dh=D(4)-D(2)+1
380   Panelwidth=Dw*.75
390   Panelheight=Dh*.75
400   X=(Dw-Panelwidth)/2
410   Y=(Dh-Panelheight)/2
```

```
420   !
430   ! Create the PANEL widget
440   !
450   CLEAR SCREEN
460   ASSIGN @P TO WIDGET "PANEL";SET ("VISIBLE":0)
470   CONTROL @P;SET ("RESIZABLE":1)
480   CONTROL @P;SET ("X":X,"Y":Y,"WIDTH":Panelwidth,"HEIGHT":Panelheight)
490   CONTROL @P;SET ("TITLE":" Example: Menu System")
500   CONTROL @P;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
510   !
520   ! Create the PULLDOWN MENU widgets, using the PANEL widget as parent
530   !
540   S$="PullDown_1"
550   ASSIGN @Pd1 TO WIDGET "PULLDOWN MENU";PARENT @P,SET ("LABEL":S$)
560   !
570   S$="PullDown_2"
580   ASSIGN @Pd2 TO WIDGET "PULLDOWN MENU";PARENT @P,SET ("LABEL":S$)
590   !
600   S$="PullDown_3"
610   ASSIGN @Pd3 TO WIDGET "PULLDOWN MENU";PARENT @P,SET ("LABEL":S$)
620   !
630   ! Now that the PULLDOWN MENU has been set up, get interior dimensions
640   ! of PANEL and set up PRINTER widget accordingly.
650   !
660   STATUS @P;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
670   Prx=Iw*.02
680   Pry=Ih*.02
690   Prwidth=Iw*.96
700   Prheight=Ih*.96
710   ASSIGN @Prn TO WIDGET "PRINTER";PARENT @P
720   CONTROL @Prn;SET ("X":Prx,"Y":Pry,"WIDTH":Prwidth,"HEIGHT":Prheight)
730   !
740   ! Create menu for "PullDown_2" (using @Pd2 as PARENT)
750   !
760   S$="Button_1"
770   ASSIGN @B21 TO WIDGET "MENU BUTTON";PARENT @Pd2,SET ("LABEL":S$)
780   !
790   S$="Button_2"
800   ASSIGN @B22 TO WIDGET "MENU BUTTON";PARENT @Pd2,SET ("LABEL":S$)
810   !
820   ! Create menu for "PullDown_3" (using @Pd3 as PARENT)
830   !
```

```
840   S$="Button_1"
850   ASSIGN @B31 TO WIDGET "MENU BUTTON";PARENT @Pd3,SET ("LABEL":S$)
860   !
870   S$="Button_2"
880   ASSIGN @B32 TO WIDGET "MENU BUTTON";PARENT @Pd3,SET ("LABEL":S$)
890   !
900   S$="Button_3"
910   ASSIGN @B33 TO WIDGET "MENU BUTTON";PARENT @Pd3,SET ("LABEL":S$)
920   !
930   ASSIGN @S34 TO WIDGET "MENU SEPARATOR";PARENT @Pd3
940   !
950   S$="Quit"
960   ASSIGN @Quit TO WIDGET "MENU BUTTON";PARENT @Pd3,SET ("LABEL":S$)
970   !
980   ! Create menu for "PullDown_1" (using @Pd1 as PARENT)
990   !
1000  S$="Button_1"
1010  ASSIGN @B11 TO WIDGET "MENU BUTTON";PARENT @Pd1,SET ("LABEL":S$)
1020  !
1030  ! Add menu separator
1040  !
1050  ASSIGN @S12 TO WIDGET "MENU SEPARATOR";PARENT @Pd1
1060  !
1070  ! Add CASCADE MENU widgets to "PullDown_1" (using @Pd1 as PARENT)
1080  !
1090  S$="Cascade_1"
1100  ASSIGN @C13 TO WIDGET "CASCADE MENU";PARENT @Pd1,SET ("LABEL":S$)
1110  !
1120  S$="Cascade_2"
1130  ASSIGN @C14 TO WIDGET "CASCADE MENU";PARENT @Pd1,SET ("LABEL":S$)
1140  !
1150  ! Create menu for "Cascade_1" (using @C1 as PARENT)
1160  !
1170  S$="Item_1"
1180  ASSIGN @B131 TO WIDGET "MENU BUTTON";PARENT @C13,SET ("LABEL":S$)
1190  !
1200  S$="Item_2"
1210  ASSIGN @B132 TO WIDGET "MENU BUTTON";PARENT @C13,SET ("LABEL":S$)
1220  !
1230  S$="Item_3"
1240  ASSIGN @B133 TO WIDGET "MENU BUTTON";PARENT @C13,SET ("LABEL":S$)
1250  !
```

```
1260  ! Create menu for "Cascade_2" (using @C14 as PARENT)
1270  !
1280  S$="Toggle_1"
1290  ASSIGN @T141 TO WIDGET "MENU TOGGLE";PARENT @C14,SET ("LABEL":S$)
1300  !
1310  S$="Toggle_2"
1320  ASSIGN @T142 TO WIDGET "MENU TOGGLE";PARENT @C14,SET ("LABEL":S$)
1330  !
1340  S$="Toggle_3"
1350  ASSIGN @T143 TO WIDGET "MENU TOGGLE";PARENT @C14,SET ("LABEL":S$)
1360  !
1370  ! Add "Cascade_3" as entry in "Cascade_2"
1380  !
1390  S$="Cascade_3"
1400  ASSIGN @C144 TO WIDGET "CASCADE MENU";PARENT @C14,SET ("LABEL":S$)
1410  !
1420  ! Populate menu for "Cascade_3"
1430  !
1440  S$="Toggle_1"
1450  ASSIGN @T1441 TO WIDGET "MENU TOGGLE";PARENT @C144,SET ("LABEL":S$)
1460  !
1470  S$="Button_1"
1480  ASSIGN @B1442 TO WIDGET "MENU BUTTON";PARENT @C144,SET ("LABEL":S$)
1490  !
1500  ! Create SYSTEM MENU
1510  !
1520  M$(0)="SysMenu1"
1530  M$(1)="SysMenu2"
1540  M$(2)="SysMenu3"
1550  CONTROL @P;SET ("SYSTEM MENU":M$(*))
1560  !
1570  CONTROL @P;SET ("VISIBLE":1)
1580  !
1590  ! Set up menu events
1600  !
1610  ON EVENT @B11,"ACTIVATED" GOSUB Button11
1620  ON EVENT @B21,"ACTIVATED" GOSUB Button21
1630  ON EVENT @B22,"ACTIVATED" GOSUB Button22
1640  ON EVENT @B31,"ACTIVATED" GOSUB Button31
1650  ON EVENT @B32,"ACTIVATED" GOSUB Button32
1660  ON EVENT @B33,"ACTIVATED" GOSUB Button33
1670  ON EVENT @B131,"ACTIVATED" GOSUB Button131
```

```
1680   ON EVENT @B132,"ACTIVATED" GOSUB Button132
1690   ON EVENT @B133,"ACTIVATED" GOSUB Button133
1700   ON EVENT @T141,"CHANGED" GOSUB Toggle141
1710   ON EVENT @T142,"CHANGED" GOSUB Toggle142
1720   ON EVENT @T143,"CHANGED" GOSUB Toggle143
1730   ON EVENT @T1441,"CHANGED" GOSUB Toggle1441
1740   ON EVENT @B1442,"ACTIVATED" GOSUB Button1442
1750   !
1760   ON EVENT @P,"SYSTEM MENU" GOSUB Sysmenu
1770   ON EVENT @Quit,"ACTIVATED" GOTO Finis
1780   !
1790   LOOP
1800     WAIT FOR EVENT
1810   END LOOP
1820   STOP
1830   !
1840   ! *************** End of Main Program **************************
1850   !
1860   ! The following routines are handlers for the various menu buttons.
1870   ! They print the menu item description to the PRINTER widget.
1880   !
1890 Button11: !
1900   S$="PullDown_1 / Button_1"
1910   CONTROL @Prn;SET ("APPEND TEXT":S$)
1920   RETURN
1930   !
1940 Button21: !
1950   S$="PullDown_2 / Button_1"
1960   CONTROL @Prn;SET ("APPEND TEXT":S$)
1970   RETURN
1980   !
1990 Button22: !
2000   S$="PullDown_2 / Button_2"
2010   CONTROL @Prn;SET ("APPEND TEXT":S$)
2020   RETURN
2030   !
2040 Button31: !
2050   S$="PullDown_3 / Button_1"
2060   CONTROL @Prn;SET ("APPEND TEXT":S$)
2070   RETURN
2080   !
2090 Button32: !
```

```
2100   S$="PullDown_3 / Button_2"
2110   CONTROL @Prn;SET ("APPEND TEXT":S$)
2120   RETURN
2130   !
2140 Button33: !
2150   S$="PullDown_3 / Button_3"
2160   CONTROL @Prn;SET ("APPEND TEXT":S$)
2170   RETURN
2180   !
2190 Button131: !
2200   S$="PullDown_1 / Cascade_1 / Item_1"
2210   CONTROL @Prn;SET ("APPEND TEXT":S$)
2220   RETURN
2230   !
2240 Button132: !
2250   S$="PullDown_1 / Cascade_1 / Item_2"
2260   CONTROL @Prn;SET ("APPEND TEXT":S$)
2270   RETURN
2280   !
2290 Button133: !
2300   S$="PullDown_1 / Cascade_1 / Item_3"
2310   CONTROL @Prn;SET ("APPEND TEXT":S$)
2320   RETURN
2330   !
2340 Toggle141: !
2350   S$="PullDown_1 / Cascade_2 / Toggle_1: "
2360   STATUS @T141;RETURN ("VALUE":State)
2370   CONTROL @Prn;SET ("APPEND TEXT":S$&VAL$(State))
2380   RETURN
2390   !
2400 Toggle142: !
2410   S$="PullDown_1 / Cascade_2 / Toggle_2: "
2420   STATUS @T142;RETURN ("VALUE":State)
2430   CONTROL @Prn;SET ("APPEND TEXT":S$&VAL$(State))
2440   RETURN
2450   !
2460 Toggle143: !
2470   S$="PullDown_1 / Cascade_2 / Toggle_3: "
2480   STATUS @T143;RETURN ("VALUE":State)
2490   CONTROL @Prn;SET ("APPEND TEXT":S$&VAL$(State))
2500   RETURN
2510   !
```

```
2520 Toggle1441: !
2530   S$="PullDown_1 / Cascade_2 / Cascade_3 / Toggle_1: "
2540   STATUS @T1441;RETURN ("VALUE":State)
2550   CONTROL @Prn;SET ("APPEND TEXT":S$&VAL$(State))
2560   RETURN
2570   !
2580 Button1442: !
2590   S$="PullDown_1 / Cascade_2 / Cascade_3 / Button_1"
2600   CONTROL @Prn;SET ("APPEND TEXT":S$)
2610   RETURN
2620   !
2630 Sysmenu: !
2640   STATUS @P;RETURN ("SYSTEM MENU EVENT":N)
2650   S$="SYSTEM MENU / SysMenu"&VAL$(N+1)
2660   CONTROL @Prn;SET ("APPEND TEXT":S$)
2670   RETURN
2680   !
2690 Finis: !
2700   ASSIGN @P TO *                    ! Delete PANEL widget
2710   END
```

```
10      ! ************************************
20      ! Example: NUMBER Dialog
30      !
40      ! This program produces a NUMBER dialog and
50      ! prompts the user for a number. The number
60      ! entered is displayed.
70      !
80      ! ****************************************
90      !
100     INTEGER Btn
110     DIM P$[25]
120     P$="Please input a number:"
130     DIALOG "NUMBER",P$,Btn;SET ("TITLE":" Example: NUMBER Dialog"),RETURN
("VALUE":X)
140     SELECT Btn
150     CASE 0
160       DISP "Number entered:",X
170     CASE 1
180       DISP "Numeric input canceled"
190     END SELECT
200     END
```

```
10      ! ********************************************************
20      ! Example: NUMBER Widget
30      !
40      ! This program creates a NUMBER widget. You can enter
50      ! a number from 0 through 1000 from the keyboard. When you
60      ! you press Enter, the number entered is displayed on the
70      ! screen. If you attempt to enter a number greater than
80      ! 1000, an error is generated and the previously-entered
90      ! number is displayed.
100     !
110     ! ********************************************************
120     !
130     ASSIGN @Number TO WIDGET "NUMBER";SET ("REAL NOTATION":"FIXED")
140     CONTROL @Number;SET ("TITLE":" Example: NUMBER Widget")
150     CONTROL @Number;SET ("X":50,"Y":25,"WIDTH":250)
160     CONTROL @Number;SET ("MINIMUM":0,"MAXIMUM":1000)
170     CONTROL @Number;SET ("CHECK FOR DONE":1)
180     CONTROL @Number;SET ("SYSTEM MENU":"Quit")
190     !
200     ON EVENT @Number,"SYSTEM MENU" GOTO Finis
210     ON EVENT @Number,"RETURN" GOSUB Get_number
220     ON EVENT @Number,"DONE" GOSUB Get_number
230     LOOP
240       WAIT FOR EVENT
250     END LOOP
260 Get_number: !
270     STATUS @Number;RETURN ("MODIFIED":New_number,"VALUE":Value)
280     IF New_number THEN
290       DISP "New number: ";Value
300       CONTROL @Number;SET ("MODIFIED":0)
310     END IF
320     RETURN
330     !
340 Finis:    !
350     ASSIGN @Number TO *             ! Delete NUMBER widget
360     END
```

```
10      ! ************************************************************
20      ! Example: Oven Control
30      !
40      ! This program creates a METER widget and a SCROLLBAR widget.
50      ! As you set the SCROLLBAR bar, the associated value is
60      ! displayed on the METER widget. Note that the SCROLLBAR
70      ! value increases as the bar is moved from top to bottom.
80      !
90      ! ************************************************************
100   !
110   COM INTEGER Count,REAL Setpt
120   !
130   ! Create the widgets
140   !
150   ASSIGN @Main TO WIDGET "PANEL"
160   CONTROL @Main;SET ("X":100,"Y":50,"WIDTH":250,"HEIGHT":200)
170   CONTROL @Main;SET ("TITLE":" Example: Oven Control")
180   CONTROL @Main;SET ("SYSTEM MENU":"Quit")
190   !
200   ASSIGN @Disp1 TO WIDGET "METER";PARENT @Main
210   CONTROL @Disp1;SET ("X":5,"Y":5,"WIDTH":150,"HEIGHT":150)
220   CONTROL @Disp1;SET ("BACKGROUND":1)
230   !
240   ASSIGN @Inp TO WIDGET "SCROLLBAR";PARENT @Main
250   CONTROL @Inp;SET ("X":190,"Y":5,"WIDTH":20,"HEIGHT":150)
260   !
270   ! Set up the EVENT handlers
280   !
290   ON EVENT @Main,"SYSTEM MENU" GOTO Finis
300   ON EVENT @Inp,"DONE" GOSUB Serviceinp
310   ON EVENT @Inp,"CHANGED" GOSUB Serviceinp
320   !
330   ! Generate values
340   !
350   Setpt=50
360   Curtemp=0
370   Siz=2
380   !
390   CONTROL @Inp;SET ("VALUE":Setpt)
400   GOSUB Serviceinp
410   LOOP
```

```
420       Diff=Setpt-Curtemp
430       Noise=Siz*(RND-.5)*2
440       Delta=Diff/10+Noise
450       Curtemp=Curtemp+Delta
460       CONTROL @Disp1;SET ("VALUE":Curtemp)
470     END LOOP
480     !
490 Serviceinp:     !
500     STATUS @Inp;RETURN ("VALUE":Setpt)
510     Setpt=INT(Setpt)
520     RETURN
530     !
540 Finis:     !
550     ASSIGN @Main TO *                  ! Delete PANEL widget
560     END
```

```
10      ! ************************************************************
20      ! Example: PID Controller
30      !
40      ! This example program simulates a PID Controller.
50      !
60      ! ************************************************************
70      !
80      DIM Points(3),Errors(64),Enum$(2)[10]
90      !
100     ! Construct the panel that controls the PID itself.
110     ! The panel indicates the current error and whether the loop is locked.
120     !
130     ASSIGN @Pid_panel TO WIDGET "PANEL";SET
("X":0,"Y":0,"WIDTH":400,"HEIGHT":245,"TITLE":"PID
Controller","RESIZABLE":0,"MAXIMIZABLE":0)
140     CONTROL @Pid_panel;SET ("TITLE":" Example: PID Controller")
150     CONTROL @Pid_panel;SET ("SYSTEM MENU":"Quit")
160     ON EVENT @Pid_panel,"SYSTEM MENU" GOTO Finis
170     !
180     ! Slider to control Kp
190     !
200     ASSIGN @Kp_slider TO WIDGET "SLIDER";PARENT @Pid_panel,SET
("X":5,"Y":30,"WIDTH":70,"HEIGHT":180,"DECIMAL DIGITS":2)
210     CONTROL @Kp_slider;SET ("MINIMUM":0,"MAXIMUM":1,"MINOR INCREMENT":.01)
220     ASSIGN @Kp_note TO WIDGET "STRING";PARENT @Pid_panel,SET
("X":5,"Y":0,"WIDTH":70,"HEIGHT":20,"VALUE":"     Kp")
230     !
240     ! Slider to control Ki
250     !
260     ASSIGN @Ki_slider TO WIDGET "SLIDER";PARENT @Pid_panel,SET
("X":80,"Y":30,"WIDTH":70,"HEIGHT":180,"DECIMAL DIGITS":2)
270     CONTROL @Ki_slider;SET ("MINIMUM":0,"MAXIMUM":1,"MINOR INCREMENT":.01)
280     ASSIGN @Ki_note TO WIDGET "STRING";PARENT @Pid_panel,SET
("X":80,"Y":0,"WIDTH":70,"HEIGHT":20,"VALUE":"     Ki")
290     !
300     ! Slider to control Kd
310     !
320     ASSIGN @Kd_slider TO WIDGET "SLIDER";PARENT @Pid_panel,SET
("X":155,"Y":30,"WIDTH":70,"HEIGHT":180,"DECIMAL DIGITS":2)
330     CONTROL @Kd_slider;SET ("MINIMUM":0,"MAXIMUM":1,"MINOR INCREMENT":.01)
340     ASSIGN @Kd_note TO WIDGET "STRING";PARENT @Pid_panel,SET
("X":155,"Y":0,"WIDTH":70,"HEIGHT":20,"VALUE":"     Kd")
```

```
350    !
360    ! Meter that displays the error
370    !
380    ASSIGN @Error_meter TO WIDGET "METER";PARENT @Pid_panel,SET
("X":235,"Y":30,"WIDTH":150,"HEIGHT":140,"MINIMUM":-1,"MAXIMUM":1,"AUTOSCALE":0)
390    ASSIGN @Meter_note TO WIDGET "STRING";PARENT @Pid_panel,SET
("X":235,"Y":0,"WIDTH":150,"HEIGHT":20,"VALUE":"         Error")
400    !
410    ! String that displays whether the loop is locked or unlocked
420    !
430    ASSIGN @Lock_indicate TO WIDGET "STRING";PARENT @Pid_panel,SET
("X":270,"Y":180,"VALUE":"UNLOCKED","WIDTH":80,"HEIGHT":30,"BACKGROUND":2)
440    !
450    ! Slider that sets the set point for the loop
460    !
470    ASSIGN @Setpt_slider TO WIDGET "SLIDER";SET
("X":410,"Y":0,"WIDTH":150,"TITLE":"Set Point","DECIMAL DIGITS":2)
480    CONTROL @Setpt_slider;SET ("MINIMUM":0,"MAXIMUM":1,"MAJOR
INCREMENT":.01,"MINOR INCREMENT":.1)
490    !
500    ! Strip chart to plot the set point and actual value
510    !
520    ASSIGN @Graph TO WIDGET "STRIPCHART";SET ("X":0,"Y":245,"TRACE
COUNT":2,"SHARED X":1,"HEIGHT":235,"RESIZABLE":0,"MAXIMIZABLE":0)
530    CONTROL @Graph;SET ("CURRENT AXIS":"Y","ORIGIN":-.25,"RANGE":1.5)
540    CONTROL @Graph;SET ("CURRENT AXIS":"X","ORIGIN":0,"RANGE":100)
550    CONTROL @Graph;SET ("CURRENT TRACE":1,"TRACE LABEL":"Set point")
560    CONTROL @Graph;SET ("CURRENT TRACE":2,"TRACE LABEL":"Actual")
570    CONTROL @Graph;SET ("TITLE":" Set Point Value")
580    !
590    ! Cyclic control for the order of the load
600    !
610    Enum$(0)="1st"
620    Enum$(1)="2nd"
630    Load_order=1
640    ASSIGN @Load TO WIDGET "PUSHBUTTON";SET
("X":410,"Y":250,"WIDTH":150,"HEIGHT":50)
650    CONTROL @Load;SET ("TITLE":"Load
Order","RESIZABLE":0,"MAXIMIZABLE":0,"LABEL":Enum$(Load_order))
660    ON EVENT @Load,"ACTIVATED",1 GOSUB Load_change
670    !
680    ! Initialize the sliders that control the PID
690    !
700    Ki=.17
```

```
710   CONTROL @Ki_slider;SET ("VALUE":Ki)
720   Kp=.4
730   CONTROL @Kp_slider;SET ("VALUE":Kp)
740   Kd=.37
750   CONTROL @Kd_slider;SET ("VALUE":Kd)
760   !
770   ! Initialize the setpoint slider
780   !
790   Setpt=.9
800   CONTROL @Setpt_slider;SET ("VALUE":Setpt)
810   !
820   ! Enable the event handling for the sliders
830   !
840   ON EVENT @Ki_slider,"DONE",2 GOSUB Ki_input
850   ON EVENT @Kp_slider,"DONE",2 GOSUB Kp_input
860   ON EVENT @Kd_slider,"DONE",2 GOSUB Kd_input
870   ON EVENT @Setpt_slider,"DONE",2 GOSUB Setpt_input
880   !
890   ! Initialize the actual value of the output and the array
900   ! that contains previous error values
910   !
920   Actual=0
930   FOR I=0 TO 62
940     Errors(I)=0
950   NEXT I
960   Sample=0
970   LOOP
980     Avg=0
990     FOR I=62 TO 0 STEP -1
1000      Avg=Avg+Errors(I)
1010      Errors(I+1)=Errors(I)
1020    NEXT I
1030    Errors(0)=Setpt-Actual
1040    Avg=(Avg+Errors(0))/64
1050    To_load=(Kp*Errors(0)+Ki*Avg+Kd*(Errors(0)-Errors(1)))
1060    SELECT Load_order
1070    CASE 0
1080    !
1090    ! First order load
1100    !
1110      Actual=(To_load*1.7)+Actual
1120    CASE 1
```

```
1130      !
1140      ! Second order load
1150      !
1160         Temp=To_load*5+Temp
1170         Actual=(Temp*.52+Actual)/2
1180      END SELECT
1190      Points(0)=Setpt
1200      Points(1)=Actual
1210      CONTROL @Graph;SET ("POINT LOCATION":Sample,"VALUES":Points(*))
1220      Sample=Sample+1
1230      CONTROL @Error_meter;SET ("VALUE":Errors(0))
1240      IF ABS(Errors(0))+ABS(Errors(1))<.02 THEN
1250         CONTROL @Lock_indicate;SET ("VALUE":"LOCKED","BACKGROUND":4)
1260      ELSE
1270         CONTROL @Lock_indicate;SET ("VALUE":"UNLOCKED","BACKGROUND":2)
1280      END IF
1290    END LOOP
1300 Ki_input: STATUS @Ki_slider;RETURN ("VALUE":Ki)
1310    Ki=PROUND(Ki,-2)
1320    RETURN
1330 Kp_input: STATUS @Kp_slider;RETURN ("VALUE":Kp)
1340    Kp=PROUND(Kp,-2)
1350    RETURN
1360 Kd_input: STATUS @Kd_slider;RETURN ("VALUE":Kd)
1370    Kd=PROUND(Kd,-2)
1380    RETURN
1390 Setpt_input: STATUS @Setpt_slider;RETURN ("VALUE":Setpt)
1400    Setpt=PROUND(Setpt,-2)
1410    RETURN
1420 Load_change: Load_order=Load_order+1
1430    IF Load_order>1 THEN Load_order=0
1440    CONTROL @Load;SET ("LABEL":Enum$(Load_order))
1450    RETURN
1460 Finis:   !
1470    ASSIGN @Pid_panel TO *                    ! Delete PANEL widget
1480    END
```

```
10        ! ************************************************************
20        ! Example: PRINTER Widget
30        !
40        ! This program generates a PRINTER widget.
50        !
60        ! ************************************************************
70        !
80       DIM S$[50],P$[50],T$[50]
90       INTEGER Lines
100    !
110    ASSIGN @Prn TO WIDGET "PRINTER"
120    CONTROL @Prn;SET ("TITLE":" Example: PRINTER Widget")
130    CONTROL @Prn;SET ("X":50,"Y":25,"WIDTH":250,"HEIGHT":125)
140    CONTROL @Prn;SET ("SYSTEM MENU":"Quit")
150    !
160    ON EVENT @Prn,"SYSTEM MENU" GOTO Finis
170    !
180    S$=TIME$(TIMEDATE)&" PRINT TEST LINE 1"
190    CONTROL @Prn;SET ("APPEND TEXT":S$)
200    Lines=1
210    LOOP
220      REPEAT
230        P$=S$
240        S$=TIME$(TIMEDATE)
250      UNTIL S$<>P$
260      IF Lines<5 THEN
270        Lines=Lines+1
280        T$=S$&" PRINT TEST LINE "&VAL$(Lines)
290        CONTROL @Prn;SET ("CURRENT LINE":1,"INSERT TEXT":T$)
300      ELSE
310        CONTROL @Prn;SET ("CURRENT LINE":5)
320        STATUS @Prn;RETURN ("CURRENT TEXT":T$)
330        CONTROL @Prn;SET ("CURRENT LINE":5,"DELETE LINES":1)
340        CONTROL @Prn;SET ("CURRENT LINE":1,"INSERT TEXT":T$)
350      END IF
360    END LOOP
370    !
380 Finis: !
390    ASSIGN @Prn TO *                  ! Delete PRINTER widget
400    END
```

```
10      ! ***************************************************
20      ! Example: PUSHBUTTON Events
30      !
40      ! This program generates a bank of four PUSHBUTTON
50      ! widgets. When you click any pushbutton, the entire
60      ! bank moves in the direction specified on the pushbutton.
70      !
80      ! ***************************************************
90      !
100     ASSIGN @Main TO WIDGET "PANEL"
110     CONTROL @Main;SET ("TITLE":" Example: PUSHBUTTON Events")
120     CONTROL @Main;SET ("HEIGHT":130,"WIDTH":275,"X":120,"Y":60)
130     CONTROL @Main;SET ("SYSTEM MENU":"Quit")
140     !
150     ! Create a bank of PUSHBUTTON widgets
160     !
170     ASSIGN @Pb1 TO WIDGET "PUSHBUTTON";PARENT @Main,SET ("X":40,"Y":10,"TAB
STOP":1)
180     ASSIGN @Pb2 TO WIDGET "PUSHBUTTON";PARENT @Main,SET ("X":140,"Y":10,"TAB
STOP":1)
190     ASSIGN @Pb3 TO WIDGET "PUSHBUTTON";PARENT @Main,SET ("X":40,"Y":60,"TAB
STOP":1)
200     ASSIGN @Pb4 TO WIDGET "PUSHBUTTON";PARENT @Main,SET ("X":140,"Y":60,"TAB
STOP":1)
210     !
220     ! Label the buttons
230     !
240     CONTROL @Pb1;SET ("LABEL":"MOVE LEFT")
250     CONTROL @Pb2;SET ("LABEL":"MOVE RIGHT")
260     CONTROL @Pb3;SET ("LABEL":"MOVE UP")
270     CONTROL @Pb4;SET ("LABEL":"MOVE DOWN")
280     !
290     ! Set up the button event handlers
300     !
310     ON EVENT @Main,"SYSTEM MENU" GOTO Finis
320     ON EVENT @Pb1,"ACTIVATED" GOSUB Moveleft
330     ON EVENT @Pb2,"ACTIVATED" GOSUB Moveright
340     ON EVENT @Pb3,"ACTIVATED" GOSUB Moveup
350     ON EVENT @Pb4,"ACTIVATED" GOSUB Movedown
360     !
370     LOOP
```

```
380       WAIT FOR EVENT
390     END LOOP
400     !
410     ! Service the button events
420     !
430 Moveleft:     !
440     CONTROL @Main;SET ("X":50)
450     BEEP
460     RETURN
470     !
480 Moveright:   !
490     CONTROL @Main;SET ("X":150)
500     BEEP
510     RETURN
520     !
530 Moveup:   !
540     CONTROL @Main;SET ("Y":30)
550     BEEP
560     RETURN
570     !
580 Movedown:   !
590     CONTROL @Main;SET ("Y":100)
600     BEEP
610     RETURN
620 Finis:     !
630     ASSIGN @Main TO *                  ! Delete PANEL widget
640     END
```

```
10      ! ************************************************
20      ! Example: PUSHBUTTON Widget
30      !
40      ! This program generates a PUSHBUTTON widget with
50      ! three states. Each time the button is pressed, the
60      ! display cycles through Label ONE, Label TWO, and
70      ! Label THREE, and the associated state (0, 1, or 2).
80      ! is displayed.
90      !
100    ! ************************************************
110    !
120    DIM L$(1:3)[50]
130    INTEGER D(1:4),Dw,Dh,Bh,Bw,Bx,By,N
140    DATA "Label ONE","Label TWO","Label THREE"
150    READ L$(*)
160    !
170    GESCAPE CRT,3;D(*)
180    Dw=D(3)-D(1)
190    Dh=(D(4)-D(2))
200    Bw=128
210    Bh=Bw/2
220    Bx=(Dw-Bw)/2
230    By=(Dh-Bh)/2
240    !
250    ASSIGN @Btn TO WIDGET "PUSHBUTTON"
260    CONTROL @Btn;SET ("TITLE":" Example: PUSHBUTTON Widget")
270    CONTROL @Btn;SET ("SYSTEM MENU":"Quit")
280    CONTROL @Btn;SET ("X":Bx/2,"Y":By/2,"WIDTH":2.25*Bw,"HEIGHT":Bh)
290    CONTROL @Btn;SET ("LABELS":L$(*),"STATES":3)
300    !
310    ON EVENT @Btn,"ACTIVATED" GOSUB Handler
320    ON EVENT @Btn,"SYSTEM MENU" GOTO Finis
330    !
340    LOOP
350      WAIT FOR EVENT
360    END LOOP
370    STOP
380    !
390 Handler:   !
400    STATUS @Btn;RETURN ("STATE":N)
410    DISP "State = "&VAL$(N)
```

```
420     RETURN
430     !
440 Finis:   !
450     ASSIGN @Btn TO *                ! Delete PUSHBUTTON widget
460     END
```

```
10      ! ************************************************************
20      ! Example: Passwords
30      !
40      ! This program demonstrates the operation of the STRING input widget
50      ! by creating a password panel that prompts the user for a name and
60      ! password. The program asks you for a name and password before it
70      ! brings up the panel.
80      !
90      ! ************************************************************
100   !
110   CLEAR SCREEN
120   OPTION BASE 1
130   !
140   ! Goodname is a flag that says a valid name has been obtained and
150   ! a password should be input. Done indicates a valid name and
160   ! password have been input.
170   !
180   INTEGER Goodname,Done
190   Goodname=0
200   Done=0
210   !
220   ! Strings Used:
230   !
240   ! Name$:          Stores user name for matching later
250   ! Pass$:          Stores user password for matching later
260   ! Inval$:         Input value from STRING widgets
270   ! B$:             General-purpose buffer
280   !
290   DIM Name$[80],Pass$[80],Inval$[80],B$[80]
300   !
310   ! Get name and password
320   !
330   INPUT "Please Input Your Name:",Name$
340   INPUT "Please Input a Password:",Pass$
350   !
360   DISP "Building Panel, Please Wait"
370   !
380   ! Define colors
390   !
400   INTEGER Black,White,Red,Blue
410   Black=0
```

```
420   White=1
430   Red=2
440   Blue=6
450   !
460   ! Get display dimensions
470   !
480   INTEGER A(6),Nlines
490   REAL Dw,Dh,Vh
500   GESCAPE CRT,3;A(*)            ! Get screen size
510   Dw=A(3)-A(1)+1
520   Dh=A(4)-A(2)+1
530   STATUS CRT,13;Nlines          ! Get number of lines of text on display
540   Vh=Dh*(1-6/Nlines)            ! Height of display above DISP line
550 !
560   REAL Px,Py,Pw,Ph,Ih,Iw
570   Pw=330                        ! Main PANEL width
580   Ph=260                        ! Main PANEL height
590   Px=(Dw-Pw)/2                  ! Center PANEL along width
600   Py=(Vh-Ph)/2                  ! Place above DISP line
610   !
620   ! Build main PANEL
630   !
640   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)          ! Make invisible
650   CONTROL @Main;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
660   CONTROL @Main;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
670   CONTROL @Main;SET ("TITLE":" Example: Passwords")
680   CONTROL @Main;SET ("SYSTEM MENU":"Quit")
690   ON EVENT @Main,"SYSTEM MENU" GOTO Finis
700   !
710   ! Get inside dimensions of PANEL
720   !
730   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
740   !
750   ! Set up dimensions for widgets
760   !
770   REAL Gapw,Gaph,Wh,Ww1,Ww2
780   REAL R1,R2,R3,R4,R5,C1,C2
790   !
800   ! Set dimensions for interior widgets
810   !
820   Gapw=Iw*.02             ! Horizontal gap
830   Gaph=Ih*.02             ! Vertical gap
```

```
840   !
850   Wh=Ih*.16                       ! Height of all child widgets
860   Ww1=Iw*.70              ! Width of STRING widgets and their LABELs
870   Ww2=Iw-Gapw*2           ! Width of prompt LABEL
880   !
890   R1=Gaph                         ! Row position of LABEL for name-input STRING
900   R2=R1+Wh                ! Row position of name-input STRING
910   R3=R2+Wh+Gaph                   ! Row position of LABEL for password-input STRING
920   R4=R3+Wh                ! Row position of password-input STRING
930   R5=R4+Wh                ! Bottom of password-input STRING
940   R6=((Ih-R5)-Wh)/2+R5    ! Row of prompt LABEL (centered at bottom)
950   !
960   C1=(Iw-Ww2)/2           ! Column position of prompt LABEL
970   C2=(Iw-Ww1)/2           ! Column position of STRINGs and their LABELs
980   !
990   ! Create LABEL for name-input STRING (no border, blends with PANEL)
1000  !
1010  ASSIGN @Lbl1 TO WIDGET "LABEL";PARENT @Main
1020  CONTROL @Lbl1;SET ("X":C2,"Y":R1,"WIDTH":Ww1,"HEIGHT":Wh)
1030  CONTROL @Lbl1;SET ("BORDER":0)
1040  CONTROL @Lbl1;SET ("FONT":"8 BY 16,BOLD")
1050  CONTROL @Lbl1;SET ("VALUE":" Enter your name:")
1060  !
1070  ! Create name-input STRING
1080  !
1090  ASSIGN @Str1 TO WIDGET "STRING";PARENT @Main
1100  CONTROL @Str1;SET ("X":C2,"Y":R2,"WIDTH":Ww1,"HEIGHT":Wh)
1110  CONTROL @Str1;SET ("BACKGROUND":Black,"PEN":Red)
1120  CONTROL @Str1;SET ("FONT":"8 BY 16")
1130  CONTROL @Str1;SET ("VALUE":"")
1140  !
1150  ! Create LABEL for password-input STRING
1160  !
1170  ASSIGN @Lbl2 TO WIDGET "LABEL";PARENT @Main
1180  CONTROL @Lbl2;SET ("X":C2,"Y":R3,"WIDTH":Ww1,"HEIGHT":Wh)
1190  CONTROL @Lbl2;SET ("BORDER":0)
1200  CONTROL @Lbl2;SET ("FONT":"8 BY 16,BOLD")
1210  CONTROL @Lbl2;SET ("VALUE":" Enter your password:")
1220  !
1230  ! Create STRING for password-input STRING. Both the background
1240  ! and font are black so the user cannot see what is typed in.
1250  ! A font type is not set, since it cannot be seen.
```

```
1260   !
1270   ASSIGN @Str2 TO WIDGET "STRING";PARENT @Main
1280   CONTROL @Str2;SET ("X":C2,"Y":R4,"WIDTH":Ww1,"HEIGHT":Wh)
1290   CONTROL @Str2;SET ("BACKGROUND":Black,"PEN":Red,"PASSWORD
CHARACTER":"*")
1300   CONTROL @Str2;SET ("VALUE":"")
1310   !
1320   ! Create prompt LABEL
1330   !
1340   ASSIGN @Lbl3 TO WIDGET "LABEL";PARENT @Main
1350   CONTROL @Lbl3;SET ("X":C1,"Y":R6,"WIDTH":Ww2,"HEIGHT":Wh)
1360   CONTROL @Lbl3;SET ("FONT":"7 BY 14")
1370   CONTROL @Lbl3;SET ("BACKGROUND":Blue,"PEN":White)
1380   GOSUB Setprompt
1390   !
1400   ! Set up events. EVENTS are set up on the password STRING for both
1410   ! each KEYSTROKE and when Return is pressed, allowing a handler
1420   ! routine to generate a BEEP every time a key is pressed (for
1430   ! feedback, since you cannot see what you are typing) and allowing
1440   ! another handler routine to take the entire password when Return
1450   ! is pressed.
1460   !
1470   ON EVENT @Str1,"RETURN" GOSUB Getname
1480   ON EVENT @Str2,"RETURN" GOSUB Getpass
1490   !
1500   CLEAR SCREEN
1510   CONTROL @Main;SET ("VISIBLE":1)
1520   !
1530   ! Main loop, wait for event
1540   !
1550   LOOP
1560      IF Done=1 THEN Finis
1570   END LOOP
1580 !
1590 ! ********** End of Main Program ******************
1600 !
1610 ! This routine puts the standard prompt
1620 ! in the prompt LABEL
1630 !
1640 Setprompt: !
1650   B$="Please enter name, then password."
1660   GOSUB Setlabel
```

```
1670   RETURN
1680    !
1690    ! This routine puts a string passed to it through B$ in the
1700    ! prompt LABEL
1710    !
1720 Setlabel: !
1730   CONTROL @Lbl3;SET ("VALUE":B$)
1740   RETURN
1750    !
1760    ! This routine is invoked when Enter is pressed after inputting a
1770    ! name in the name-input STRING.    First, it tests to see if a name
1780    ! has already been input. If so, it outputs a message and exits.
1790    !
1800    ! Otherwise, it checks the input value for a match with the password.
1810    ! If it does not match, it gives a warning message and sets Goodname
1820    ! to 0.
1830    !
1840    ! If it does match, it changes the prompt to ask for a password only.
1850    ! In either case, it clears the STRING.
1860    !
1870 Getname: !
1880   DISABLE
1890   STATUS @Str1;RETURN ("VALUE":Inval$)
1900   IF Goodname=1 THEN
1910     BEEP 2000,.01
1920     B$="Please enter password"
1930     GOSUB Setlabel
1940   ELSE
1950     IF Inval$<>Name$ THEN
1960       BEEP 2000,.1
1970       B$="Invalid name"
1980       GOSUB Setlabel
1990       GOSUB Setprompt
2000       Goodname=0
2010     ELSE
2020       B$="Name accepted, please enter password."
2030       GOSUB Setlabel
2040       Goodname=1
2050     END IF
2060   END IF
2070   CONTROL @Str1;SET ("VALUE":"")
2080   ENABLE
```

```
2090   RETURN
2100    !
2110    ! This routine checks for a valid password. First, it checks
2120    ! to see if the user has provided a valid name. If not, it beeps
2130    ! and tells the user to input a valid name.
2140    !
2150    ! Otherwise, it tries to match the input string against the password.
2160    ! If no match, it tells the user and returns. If it matches, it also
2170    ! tells the user and sets a flag to tell the main routine it is done.
2180    !
2190 Getpass: !
2200   DISABLE
2210   IF Goodname=0 THEN
2220     BEEP 2000,.01
2230     B$="Please input your name first."
2240     GOSUB Setlabel
2250     WAIT 2
2260     GOSUB Setprompt
2270   ELSE
2280     STATUS @Str2;RETURN ("VALUE":Inval$)
2290     IF Inval$<>Pass$ THEN
2300       BEEP 2000,.01
2310       B$="Invalid password, try again."
2320       GOSUB Setlabel
2330       CONTROL @Str2;SET ("VALUE":"")
2340     ELSE
2350       B$="Welcome to the club!"
2360       GOSUB Setlabel
2370       WAIT 10
2380       Done=1
2390     END IF
2400   END IF
2410   ENABLE
2420   RETURN
2430    !
2440 Finis: !
2450   ASSIGN @Panel TO *                ! Delete PANEL widget
2460   END
```

```
10        ! ************************************************
20        ! Example: QUESTION Dialog
30        !
40        ! This program provides a QUESTION dialog and displays
50        ! the user response (Yes or No) to the question.
60        !
70        ! ************************************************
80        !
90         DIM P$[20]
100       INTEGER Btn
110       P$="Do you want to exit?"
120       DIALOG "QUESTION",P$,Btn;SET ("TITLE":" Example: QUESTION Dialog")
130       IF Btn=0 THEN DISP "Yes"
140       IF Btn=1 THEN DISP "No"
150       END
```

```
10      ! ****************************************************************
20      ! Example: RADIOBUTTON Widget
30      !
40      ! This program generates a panel with two tab groups of RADIOBUTTON
50      ! widgets. You can set buttons and observe their behavior. A
60      ! PRINTER widget is used to display the state of the buttons used.
70      !
80      ! ****************************************************************
90      !
100   ! Define colors
110   !
120   INTEGER Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
130   DATA 0,1,2,3,4,5,6,7
140   READ Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
150   !
160   ! Set the PRINTER widget handle COMmon
170   !
180   COM @Printer
190   !
200   ! Variables for display and widget handling
210   !
220   INTEGER Nlines,D(1:4)                    ! Used for display setup
230   REAL Dw,Dh                               ! Display dimensions
240   REAL Px,Py,Pw,Ph,Iw,Ih                   ! Main PANEL parameters
250   REAL Gaph,Gapw,Bh,Bw,Sh,Sw,Prh,Prw       ! Widget dimensions
260   REAL R1,R2,R3,R4,R5,C1,C2,C3,C4,C5       ! Widget coordinates
270   !
280   ! Set up display
290   !
300   STATUS CRT,13;Nlines
310   GESCAPE CRT,3;D(*)
320   Dw=D(3)-D(1)
330   Dh=(D(4)-D(2))*((Nlines-7)/Nlines)
340   !
350   ! Set up PANEL coordinates
360   !
370   Pw=Dw*.5
380   Ph=Dh*.65
390   Px=(Dw-Pw)
400   Py=(Dh-Ph)
410   !
```

```
420   ! Build PANEL widget and add SYSTEM MENU to PANEL
430   !
440   CLEAR SCREEN
450   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
460   CONTROL @Main;SET ("RESIZABLE":0,"MAXIMIZABLE":0)
470   CONTROL @Main;SET ("X":Px/4,"Y":Py/2,"WIDTH":1.3*Pw,"HEIGHT":1.5*Ph)
480   CONTROL @Main;SET ("TITLE":" Example: RADIOBUTTON Widget")
490   CONTROL @Main;SET ("SYSTEM MENU":"Quit")
500   !
510   ! Get interior dimensions
520   !
530   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
540   !
550   ! Set up widget coordinates and dimensions
560   !
570   Gaph=Ih*.03                   ! Vertical gap
580   Gapw=Iw*.03                   ! Horizontal gap
590   Bh=Ih*.15                           ! Button height
600   Bw=(Iw-Gapw*6)/5                    ! Button width
610   Sh=Gaph                             ! Separator height
620   Prh=Ih-((Bh+Sh)*2+Gaph*6)     ! Printer height
630   Prw=(Iw-2*Gapw)                     ! Printer width
640   Sw=Prw                        ! Separator width
650   !
660   R1=Gaph                             ! Top row
670   R2=R1+Bh+Gaph                       ! Row 2 = row 1 + button height + gap
680   R3=R2+Sh+Gaph                       ! Row 3 = row 2 + button height + gap
690   R4=R3+Bh+Gaph                       ! Row 4 = row 3 + button height + gap
700   R5=R4+Sh+Gaph                       ! Row 5 = row 4 + button height + gap
710   !
720   C1=Gapw                             ! Left column
730   C2=C1+Bw+Gapw                       ! Col 2 = col 1 + button width + gap
740   C3=C2+Bw+Gapw                       ! Col 3 = col 2 + button width + gap
750   C4=C3+Bw+Gapw                       ! Col 4 = col 3 + button width + gap
760   C5=C4+Bw+Gapw                       ! Col 5 = col 4 + button width + gap
770   Pw=C5+Bw+Gapw                       ! Panel width = col 5 + button width + gap
780   !
790   ! Set up first tab group of RADIOBUTTONs. Set tab stop on A1.
800   !
810   ASSIGN @A1 TO WIDGET "RADIOBUTTON";PARENT @Main
820   CONTROL @A1;SET ("LABEL":"A1","TAB STOP":1)
830   CONTROL @A1;SET ("X":C1,"Y":R1,"WIDTH":Bw,"HEIGHT":Bh)
```

```
840   !
850   ! No tab stop on A2
860   !
870   ASSIGN @A2 TO WIDGET "RADIOBUTTON";PARENT @Main
880   CONTROL @A2;SET ("LABEL":"A2","TAB STOP":0)
890   CONTROL @A2;SET ("X":C2,"Y":R1,"WIDTH":Bw,"HEIGHT":Bh)
900   !
910   ! No tab stop on A3
920   !
930   ASSIGN @A3 TO WIDGET "RADIOBUTTON";PARENT @Main
940   CONTROL @A3;SET ("LABEL":"A3","TAB STOP":0)
950   CONTROL @A3;SET ("X":C3,"Y":R1,"WIDTH":Bw,"HEIGHT":Bh)
960   !
970   ! No tab stop on A4
980   !
990   ASSIGN @A4 TO WIDGET "RADIOBUTTON";PARENT @Main
1000  CONTROL @A4;SET ("LABEL":"A4","TAB STOP":0)
1010  CONTROL @A4;SET ("X":C4,"Y":R1,"WIDTH":Bw,"HEIGHT":Bh)
1020  !
1030  ! No tab stop on A5
1040  !
1050  ASSIGN @A5 TO WIDGET "RADIOBUTTON";PARENT @Main
1060  CONTROL @A5;SET ("LABEL":"A5","TAB STOP":0)
1070  CONTROL @A5;SET ("X":C5,"Y":R1,"WIDTH":Bw,"HEIGHT":Bh)
1080  !
1090  ! Separator to mark off first tab group
1100  !
1110  ASSIGN @Sep1 TO WIDGET "SEPARATOR";PARENT @Main
1120  CONTROL @Sep1;SET ("X":C1,"Y":R2,"WIDTH":Sw,"HEIGHT":Sh)
1130  !
1140  ! Set up second tab group, set TAB STOP on B1
1150  !
1160  ASSIGN @B1 TO WIDGET "RADIOBUTTON";PARENT @Main
1170  CONTROL @B1;SET ("LABEL":"B1","TAB STOP":1)
1180  CONTROL @B1;SET ("X":C1,"Y":R3,"WIDTH":Bw,"HEIGHT":Bh)
1190  !
1200  ! B2, no tab stop
1210  !
1220  ASSIGN @B2 TO WIDGET "RADIOBUTTON";PARENT @Main
1230  CONTROL @B2;SET ("LABEL":"B2","TAB STOP":0)
1240  CONTROL @B2;SET ("X":C2,"Y":R3,"WIDTH":Bw,"HEIGHT":Bh)
1250  !
```

```
1260  ! B3, no tab stop
1270  !
1280  ASSIGN @B3 TO WIDGET "RADIOBUTTON";PARENT @Main
1290  CONTROL @B3;SET ("LABEL":"B3","TAB STOP":0)
1300  CONTROL @B3;SET ("X":C3,"Y":R3,"WIDTH":Bw,"HEIGHT":Bh)
1310  !
1320  ! B4, no tab stop
1330  !
1340  ASSIGN @B4 TO WIDGET "RADIOBUTTON";PARENT @Main
1350  CONTROL @B4;SET ("LABEL":"B4","TAB STOP":0)
1360  CONTROL @B4;SET ("X":C4,"Y":R3,"WIDTH":Bw,"HEIGHT":Bh)
1370  !
1380  ! B5, no tab stop
1390  !
1400  ASSIGN @B5 TO WIDGET "RADIOBUTTON";PARENT @Main
1410  CONTROL @B5;SET ("LABEL":"B5","TAB STOP":0)
1420  CONTROL @B5;SET ("X":C5,"Y":R3,"WIDTH":Bw,"HEIGHT":Bh)
1430  !
1440  ! Separator for second tab group
1450  !
1460  ASSIGN @Sep2 TO WIDGET "SEPARATOR";PARENT @Main
1470  CONTROL @Sep2;SET ("X":C1,"Y":R4,"WIDTH":Sw,"HEIGHT":Sh)
1480  !
1490  ! PRINTER widget
1500  !
1510  ASSIGN @Printer TO WIDGET "PRINTER";PARENT @Main,SET ("TAB STOP":1)
1520  CONTROL @Printer;SET ("TAB STOP":1)
1530  CONTROL @Printer;SET ("BACKGROUND":White)
1540  CONTROL @Printer;SET ("PEN":Black)
1550  CONTROL @Printer;SET ("X":C1,"Y":R5,"WIDTH":Prw,"HEIGHT":Prh)
1560  !
1570  ! Set up events
1580  !
1590  ON EVENT @A1,"CHANGED" GOSUB A1
1600  ON EVENT @A2,"CHANGED" GOSUB A2
1610  ON EVENT @A3,"CHANGED" GOSUB A3
1620  ON EVENT @A4,"CHANGED" GOSUB A4
1630  ON EVENT @A5,"CHANGED" GOSUB A5
1640  !
1650  ON EVENT @B1,"CHANGED" GOSUB B1
1660  ON EVENT @B2,"CHANGED" GOSUB B2
1670  ON EVENT @B3,"CHANGED" GOSUB B3
```

```
1680   ON EVENT @B4,"CHANGED" GOSUB B4
1690   ON EVENT @B5,"CHANGED" GOSUB B5
1700   !
1710   ON EVENT @Main,"SYSTEM MENU" GOTO Finis
1720   !
1730   ! Turn on panel
1740   !
1750   CONTROL @Main;SET ("VISIBLE":1)
1760   !
1770   ! Main loop, wait for event
1780   !
1790   LOOP
1800      WAIT FOR EVENT
1810   END LOOP
1820   STOP
1830   !
1840   ! Handlers for buttons
1850   !
1860 A1: !
1870   Show_value(@A1,"Button A1, VALUE:")
1880   RETURN
1890   !
1900 A2: !
1910   Show_value(@A2,"Button A2, VALUE:")
1920   RETURN
1930   !
1940 A3: !
1950   Show_value(@A3,"Button A3, VALUE:")
1960   RETURN
1970   !
1980 A4: !
1990   Show_value(@A4,"Button A4, VALUE:")
2000   RETURN
2010   !
2020 A5: !
2030   Show_value(@A5,"Button A5, VALUE:")
2040   RETURN
2050   !
2060 B1: !
2070   Show_value(@B1,"Button B1, VALUE:")
2080   RETURN
2090   !
```

```
2100 B2: !
2110   Show_value(@B2,"Button B2, VALUE:")
2120   RETURN
2130   !
2140 B3: !
2150   Show_value(@B3,"Button B3, VALUE:")
2160   RETURN
2170   !
2180 B4: !
2190   Show_value(@B4,"Button B4, VALUE:")
2200   RETURN
2210   !
2220 B5: !
2230   Show_value(@B5,"Button B5, VALUE:")
2240   RETURN
2250   !
2260 Finis: !
2270   ASSIGN @Main TO *                  ! Delete PANEL widget
2280   END
2290   !
2300   ! ********** End of Main Program ****************
2310   !
2320   ! SUB to handle display of button inputs
2330   ! on PRINTER widget
2340   !
2350   SUB Show_value(@B,Text$)
2360     COM @Printer
2370     INTEGER Checked
2380     STATUS @B;RETURN ("VALUE":Checked)
2390     CONTROL @Printer;SET ("APPEND TEXT":Text$&VAL$(Checked))
2400   SUBEND
```

```
10      ! ****************************************************************
20      ! Example: Rock/Paper/Scissors Game
30      !
40      ! This program is designed to show the operation of LABEL and
50      ! PUSHBUTTON widgets. It plays the game of "rock/paper/scissors"
60      ! in which each of two players guesses one of three items -
70      ! rock, paper, or scissors. The guesses are then compared
80      ! for the following outcomes:
90      !
100     !   - An identical guess is a tie
110     !   - Paper covers rock, paper wins
120     !   - Scissors cuts paper, scissors wins
130     !   - Rock breaks scissors, rock wins
140     !
150     ! The program sets up a panel with a large LABEL to prompt the user
160     ! for his or her guess, sets up a PUSHBUTTON for each guess, then
170     ! goes into a loop where it waits for PUSHBUTTON input.
180     !
190     ! Each one of the three PUSHBUTTON widgets causes an event that invokes
200     ! the appropriate handler routine. These three routines each contain
210     ! a SELECT statement that matches the PUSHBUTTON against a guess made
220     ! by the program -- rock, scissors, or paper.    On the basis of the
230     ! match, each routine then provides a dialog indicating the match and
240     ! calls the appropriate Win, Lose, or Tie routine.
250     !
260     ! ****************************************************************
270     !
280     RANDOMIZE INT(FRACT(TIMEDATE)*10^7)
290     !
300     ! Variables Used:
310     !
320     ! Me,You,Tie:                   Store game outcomes
330     ! Rock,Paper,Scissors:          Guess values
340     ! B$,S$:                        General purpose strings
350     !
360     INTEGER Me,You,Tie,Rock,Paper,Scissors
370     DATA 0,0,0,0,1,2
380     READ Me,You,Tie,Rock,Paper,Scissors
390     !
400     INTEGER Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
410     DATA 0,1,2,3,4,5,6,7
```

```
420   READ Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
430   !
440   DIM B$[50],S$[100]
450   INTEGER Nlines,D(1:4),Dw,Dh                  ! Variables for display handling
460   INTEGER Pw,Ph,Px,Py,Iw,Ih                    ! PANEL dimensions & coordinates
470   INTEGER Gapw,Gaph,W1,H1,W2,H2                ! Widget scaling factors
480   INTEGER R1,R2,R3,R4,R5,R6,C1,C2,C3
490   !
500   ! Get display dimensions
510   !
520   STATUS CRT,13;Nlines
530   GESCAPE CRT,3;D(*)
540   Dw=D(3)-D(1)
550   Dh=(D(4)-D(2))*((Nlines-7)/Nlines)
560   !
570   Pw=Dw*.5
580   Ph=Dh*.8
590   Px=(Dw-Pw)/2
600   Py=(Dh-Ph)/2
610   !
620   CLEAR SCREEN
630   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
640   CONTROL @Main;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
650   CONTROL @Main;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
660   CONTROL @Main;SET ("TITLE":" Example: Rock/Paper/Scissors Game")
670   CONTROL @Main;SET ("SYSTEM MENU":"Quit")
680   !
690   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
700   !
710   Gapw=Iw*.03                  ! Width gap
720   Gaph=Ih*.03                  ! Height gap
730   W1=(Iw-Gapw*4)/3                  ! Width of PUSHBUTTONs and most LABELS
740   H1=(Ih-Gaph*8)/4                  ! Height of PUSHBUTTONs and all LABELS
750   W2=Iw-Gapw*2                 ! Width of main LABEL and all SEPARATORs
760   H2=Gaph                           ! Height of SEPARATORs
770   !
780   C1=Gapw                           ! Column coordinates, based on PUSHBUTTON
width
790   C2=C1+W1+Gapw
800   C3=C2+W1+Gapw
810   !
820   R1=Gaph                           ! Row coordinates:   row for main LABEL
```

```
830   R2=R1+H1+Gaph                          ! Row for first SEPARATOR
840   R3=R2+H2+Gaph                          ! Row for input PUSHBUTTONs
850   R4=R3+H1+Gaph                          ! Row for second SEPARATOR
860   R5=R4+H2                       ! Row for result-title LABELs
870   R6=R5+H1                       ! Row for result-value LABELs
880   !
890   ! Now add child widgets: main LABEL
900   !
910   ASSIGN @Label TO WIDGET "LABEL";PARENT @Main
920   CONTROL @Label;SET ("X":C1,"Y":R1,"WIDTH":W2,"HEIGHT":H1)
930   CONTROL @Label;SET ("BORDER":0,"BACKGROUND":Black,"PEN":Red)
940   CONTROL @Label;SET ("PEN":White,"VALUE":"Make a guess!")
950   !
960   ! Create SEPARATOR widget
970   !
980   ASSIGN @S1 TO WIDGET "SEPARATOR";PARENT @Main
990   CONTROL @S1;SET ("X":C1,"Y":R2,"WIDTH":W2,"HEIGHT":H2)
1000  !
1010  ! User input keys
1020  !
1030  ASSIGN @Rock TO WIDGET "PUSHBUTTON";PARENT @Main
1040  CONTROL @Rock;SET ("X":C1,"Y":R3,"WIDTH":W1,"HEIGHT":H1)
1050  CONTROL @Rock;SET ("BACKGROUND":Blue,"PEN":White)
1060  CONTROL @Rock;SET ("LABEL":"ROCK")
1070  !
1080  ASSIGN @Scissors TO WIDGET "PUSHBUTTON";PARENT @Main
1090  CONTROL @Scissors;SET ("X":C2,"Y":R3,"WIDTH":W1,"HEIGHT":H1)
1100  CONTROL @Scissors;SET ("BACKGROUND":Blue,"PEN":White)
1110  CONTROL @Scissors;SET ("LABEL":"SCISSORS")
1120  !
1130  ASSIGN @Paper TO WIDGET "PUSHBUTTON";PARENT @Main
1140  CONTROL @Paper;SET ("X":C3,"Y":R3,"WIDTH":W1,"HEIGHT":H1)
1150  CONTROL @Paper;SET ("BACKGROUND":Blue,"PEN":White)
1160  CONTROL @Paper;SET ("LABEL":"PAPER")
1170  !
1180  ! Next SEPARATOR widget
1190  !
1200  ASSIGN @S2 TO WIDGET "SEPARATOR";PARENT @Main
1210  CONTROL @S2;SET ("X":C1,"Y":R4,"WIDTH":W2,"HEIGHT":H2)
1220  !
1230  ! Results titles and values. Note that none of these LABELS have
1240  ! borders; the titles also have backgrounds that have the same color
```

```
1250  ! as the PANEL background, so they appear to be written directly on
1260  ! the PANEL.
1270  !
1280  ASSIGN @Me_title TO WIDGET "LABEL";PARENT @Main
1290  CONTROL @Me_title;SET ("X":C1,"Y":R5,"WIDTH":W1,"HEIGHT":H1)
1300  CONTROL @Me_title;SET ("BORDER":0,"VALUE":"ME")
1310  !
1320  ASSIGN @Me_value TO WIDGET "LABEL";PARENT @Main
1330  CONTROL @Me_value;SET ("X":C1,"Y":R6,"WIDTH":W1,"HEIGHT":H1)
1340  CONTROL @Me_value;SET ("BORDER":0,"BACKGROUND":Black,"PEN":Red)
1350  CONTROL @Me_value;SET ("VALUE":Me)
1360  !
1370  ASSIGN @You_title TO WIDGET "LABEL";PARENT @Main
1380  CONTROL @You_title;SET ("X":C2,"Y":R5,"WIDTH":W1,"HEIGHT":H1)
1390  CONTROL @You_title;SET ("BORDER":0,"VALUE":"YOU")
1400  !
1410  ASSIGN @You_value TO WIDGET "LABEL";PARENT @Main
1420  CONTROL @You_value;SET ("X":C2,"Y":R6,"WIDTH":W1,"HEIGHT":H1)
1430  CONTROL @You_value;SET ("BORDER":0,"BACKGROUND":Black,"PEN":Green)
1440  CONTROL @You_value;SET ("VALUE":You)
1450  !
1460  ASSIGN @Tie_title TO WIDGET "LABEL";PARENT @Main
1470  CONTROL @Tie_title;SET ("X":C3,"Y":R5,"WIDTH":W1,"HEIGHT":H1)
1480  CONTROL @Tie_title;SET ("BORDER":0,"VALUE":"TIE")
1490  !
1500  ASSIGN @Tie_value TO WIDGET "LABEL";PARENT @Main
1510  CONTROL @Tie_value;SET ("X":C3,"Y":R6,"WIDTH":W1,"HEIGHT":H1)
1520  CONTROL @Tie_value;SET ("BORDER":0,"BACKGROUND":Black,"PEN":Yellow)
1530  CONTROL @Tie_value;SET ("VALUE":Tie)
1540  !
1550  ! PANEL completed, set up EVENTS
1560  !
1570  ON EVENT @Rock,"ACTIVATED" GOSUB Gotrock
1580  ON EVENT @Scissors,"ACTIVATED" GOSUB Gotscissors
1590  ON EVENT @Paper,"ACTIVATED" GOSUB Gotpaper
1600  ON EVENT @Main,"SYSTEM MENU" GOTO Finis
1610  !
1620  ! Turn on the PANEL, make everything visible, then loop
1630  !
1640  CONTROL @Main;SET ("VISIBLE":1)
1650  !
1660  ! Main loop, set random-number seed, then loop
```

```
1670   !
1680   LOOP
1690     WAIT FOR EVENT
1700   END LOOP
1710   STOP
1720   !
1730   ! ****************   End of Main Program **********************
1740   !
1750   ! The three following handler routines are all basically similar:
1760   ! each is invoked by one of the three PUSHBUTTONs and uses a SELECT
1770   ! statement to figure the appropriate outcome. For each outcome, it
1780   ! loads a string with the appropriate result and calls the appropriate
1790   ! Win, Lose, or Tie routine.
1800   !
1810 Gotrock: !
1820   SELECT INT(3*RND)
1830   CASE Rock
1840     B$="ROCK ON ROCK"
1850     GOSUB Tie
1860   CASE Scissors
1870     B$="ROCK BREAKS SCISSORS"
1880     GOSUB Win
1890   CASE Paper
1900     B$="PAPER COVERS ROCK"
1910     GOSUB Lose
1920   END SELECT
1930   RETURN
1940   !
1950 Gotscissors: !
1960   SELECT INT(3*RND)
1970   CASE Rock
1980     B$="ROCK BREAKS SCISSORS"
1990     GOSUB Lose
2000   CASE Scissors
2010     B$="SCISSORS ON SCISSORS"
2020     GOSUB Tie
2030   CASE Paper
2040     B$="SCISSORS CUTS PAPER"
2050     GOSUB Win
2060   END SELECT
2070   RETURN
2080   !
```

```
2090 Gotpaper: !
2100   SELECT INT(3*RND)
2110   CASE Rock
2120     B$="PAPER COVERS ROCK"
2130     GOSUB Win
2140   CASE Scissors
2150     B$="SCISSORS CUTS PAPER"
2160     GOSUB Lose
2170   CASE Paper
2180     B$="PAPER ON PAPER"
2190     GOSUB Tie
2200   END SELECT
2210   RETURN
2220   !
2230   ! The three results routines -- Win, Tie, Lose -- are also similar to
2240   ! each other. Each takes the result of the match from the calling
2250   ! routine, concatenates it with the appropriate conclusion, displays
2260   ! it with a dialog, then increments the appropriate tally and returns.
2270   !
2280 Win: !
2290   S$=B$&": You WIN!"
2300   DIALOG "INFORMATION",S$;SET ("TITLE":" Results","PEN":Green)
2310   You=You+1
2320   CONTROL @You_value;SET ("VALUE":You)
2330   RETURN
2340   !
2350 Tie: !
2360   S$=B$&": Tie!"
2370   DIALOG "INFORMATION",S$;SET ("TITLE":" Results","PEN":Yellow)
2380   Tie=Tie+1
2390   CONTROL @Tie_value;SET ("VALUE":Tie)
2400   RETURN
2410   !
2420 Lose: !
2430   S$=B$&": You LOSE!"
2440   DIALOG "INFORMATION",S$;SET ("TITLE":" Results","PEN":Red)
2450   Me=Me+1
2460   CONTROL @Me_value;SET ("VALUE":Me)
2470   RETURN
2480   !
2490 Finis: !
2500   ASSIGN @Main TO *                ! Delete the PANEL widget
```

```
2510   END
```

```
10      ! ************************************************
20      ! Example: SCROLLBAR Widget
30      !
40      ! This program generates a horizontal SCROLLBAR widget
50      ! and displays the value set with the slider.
60      !
70      ! ************************************************
80      !
90      ASSIGN @Scroll TO WIDGET "SCROLLBAR"
100   CONTROL @Scroll;SET ("TITLE":" Example: SCROLLBAR Widget")
110   CONTROL @Scroll;SET ("ORIENTATION":"HORIZONTAL")
120   CONTROL @Scroll;SET ("X":50,"Y":25,"HEIGHT":50,"WIDTH":275)
130   CONTROL @Scroll;SET ("SYSTEM MENU":"Quit")
140   !
150   ON EVENT @Scroll,"SYSTEM MENU" GOTO Finis
160   ON EVENT @Scroll,"DONE" GOSUB Dispval
170   LOOP
180     WAIT FOR EVENT
190   END LOOP
200   !
210 Dispval: !
220   STATUS @Scroll;RETURN ("VALUE":Scrollval)
230   DISP Scrollval
240   RETURN
250   !
260 Finis: !
270   CLEAR SCREEN
280   ASSIGN @Scroll TO *              ! Delete SCROLLBAR widget
290   END
```

```
10      ! ********************************************************************
20      ! Example: SLIDER Test
30      !
40      ! This program consists of a SLIDER and a LABEL in a PANEL with
50      ! a pulldown menu system. You set the SLIDER to a value, and the
60      ! LABEL shows the value.
70      !
80      ! ********************************************************************
90      !
100   ! Define colors
110   !
120   INTEGER Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
130   DATA 0,1,2,3,4,5,6,7
140   READ Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
150   !
160   ! Log and Direct are flags for LOGARITHMIC and DIRECT MOVE mode
170   ! settings. V is the value returned from the SLIDER.
180   !
190   INTEGER Log,Direct
200   REAL V
210   !
220   ! Variables for display handling
230   !
240   INTEGER Nlines,D(1:4),Dw,Dh,Iw,Ih,Gx,Gy              ! Display size parameters
250   INTEGER Pw,Ph,Px,Py                          ! PANEL parameters
260   INTEGER Lx,Ly,Lw,Lh,Sx,Sy,Sw,Sh              ! LABEL & SLIDER parameters
270   !
280   ! Get display size
290   !
300   STATUS CRT,13;Nlines
310   GESCAPE CRT,3;D(*)
320   Dw=D(3)-D(1)
330   Dh=(D(4)-D(2))*((Nlines-7)/Nlines)
340   !
350   ! Set main PANEL dimensions and origin
360   !
370   Pw=Dw*.35
380   Ph=Dh*.8
390   Px=(Dw-Pw)/2
400   Py=(Dh-Ph)/2
410   !
```

```
420   ! Build main PANEL
430   !
440   CLEAR SCREEN
450   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
460   CONTROL @Main;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
470   CONTROL @Main;SET ("RESIZABLE":0,"MAXIMIZABLE":0)
480   CONTROL @Main;SET ("BACKGROUND":9,"TITLE":" Example: SLIDER Test")
490   !
500   ! Build menu
510   !
520   ASSIGN @Menu TO WIDGET "PULLDOWN MENU";PARENT @Main
530   CONTROL @Menu;SET ("LABEL":" Menu")
540   ASSIGN @Log TO WIDGET "MENU BUTTON";PARENT @Menu
550   CONTROL @Log;SET ("LABEL":" Set LOG mode")
560   Log=0
570   ASSIGN @Direct TO WIDGET "MENU BUTTON";PARENT @Menu
580   CONTROL @Direct;SET ("LABEL":" Set DIRECT MOVE")
590   Direct=0
600   ASSIGN @S1 TO WIDGET "MENU SEPARATOR";PARENT @Menu
610   ASSIGN @Quit TO WIDGET "MENU BUTTON";PARENT @Menu
620   CONTROL @Quit;SET ("LABEL":" Quit")
630   !
640   ! PANEL done, get inside dimensions
650   !
660   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
670   !
680   ! Set widget parameters
690   !
700   Gx=Iw*.1                        ! Set gaps
710   Gy=Ih*.05
720   Lx=Gx                           ! Position & size LABEL
730   Ly=Gy
740   Lw=Iw*.4
750   Lh=Iw*.2
760   Sx=Lx+Lw+Gx                     ! Position & size SLIDER
770   Sy=Gy
780   Sw=Iw-(Sx+Gx)
790   Sh=Ih-(2*Gy)
800   !
810   ! Create LABEL widget
820   !
830   ASSIGN @Label TO WIDGET "LABEL";PARENT @Main
```

```
840    CONTROL @Label;SET ("X":Lx,"Y":Ly,"WIDTH":Lw,"HEIGHT":Lh)
850    CONTROL @Label;SET ("BACKGROUND":1,"PEN":Black)
860    !
870    ! Create SLIDER widget
880    !
890    ASSIGN @Slider TO WIDGET "SLIDER";PARENT @Main
900    CONTROL @Slider;SET ("X":Sx,"Y":Sy,"WIDTH":Sw,"HEIGHT":Sh)
910    CONTROL @Slider;SET ("AUTO REPEAT":1,"LOGARITHMIC":0)
920    CONTROL @Slider;SET ("MAXIMUM":100,"MINIMUM":1)
930    CONTROL @Slider;SET ("MAJOR INCREMENT":10,"MINOR INCREMENT":1)
940    !
950    ! Enable events
960    !
970    ON EVENT @Log,"ACTIVATED" GOSUB Set_log
980    ON EVENT @Direct,"ACTIVATED" GOSUB Set_direct
990    ON EVENT @Slider,"CHANGED" GOSUB Set_label
1000   ON EVENT @Quit,"ACTIVATED" GOTO Finis
1010   !
1020   ! Initialize menu labels, make all widgets visible
1030   !
1040   GOSUB Set_label
1050   CONTROL @Main;SET ("VISIBLE":1)
1060   !
1070   ! Wait for event to happen
1080   !
1090   LOOP
1100     WAIT FOR EVENT
1110   END LOOP
1120   STOP
1130   !
1140   ! ********* Subroutines *****************
1150   !
1160   ! Toggle between log and linear mode
1170   !
1180 Set_log: !
1190   SELECT Log
1200   CASE 0
1210     CONTROL @Slider;SET ("LOGARITHMIC":1)
1220     CONTROL @Log;SET ("LABEL":"Set LINEAR Mode")
1230     Log=1
1240   CASE 1
1250     CONTROL @Slider;SET ("LOGARITHMIC":0)
```

```
1260      CONTROL @Log;SET ("LABEL":"Set LOG Mode")
1270      Log=0
1280    END SELECT
1290    RETURN
1300    !
1310    ! Toggle DIRECT MOVE on or off
1320    !
1330 Set_direct: !
1340    SELECT Direct
1350    CASE 0
1360      CONTROL @Slider;SET ("DIRECT MOVE":1)
1370      CONTROL @Direct;SET ("LABEL":"Clear DIRECT MOVE")
1380      Direct=1
1390    CASE 1
1400      CONTROL @Slider;SET ("DIRECT MOVE":0)
1410      CONTROL @Direct;SET ("LABEL":"Set DIRECT MOVE")
1420      Direct=0
1430    END SELECT
1440    RETURN
1450    !
1460    ! Set label value from SLIDER widget
1470    !
1480 Set_label: !
1490    STATUS @Slider;RETURN ("VALUE":V)
1500    CONTROL @Label;SET ("VALUE":V)
1510    RETURN
1520    !
1530 Finis: !
1540    ASSIGN @Main TO *                    ! Delete PANEL widget
1550    END
```

```
10      ! ********************************************************
20      ! Example: SLIDER Widget
30      !
40      ! This program creates a SLIDER widget and displays
50      ! the value set by the slider.
60      !
70      ! ********************************************************
80      !
90      CLEAR SCREEN
100   ASSIGN @Slider TO WIDGET "SLIDER"
110   CONTROL @Slider;SET ("TITLE":" Example: SLIDER Widget")
120   CONTROL @Slider;SET ("X":150,"Y":25,"WIDTH":250,"HEIGHT":200)
130   CONTROL @Slider;SET ("SYSTEM MENU":"Quit")
140   !
150   ON EVENT @Slider,"SYSTEM MENU" GOTO Finis
160   ON EVENT @Slider,"DONE" GOSUB Dispval
170   LOOP
180     WAIT FOR EVENT
190   END LOOP
200   !
210 Dispval: !
220   STATUS @Slider;RETURN ("VALUE":Sliderval)
230   DISP Sliderval
240   RETURN
250 Finis: !
260   ASSIGN @Slider TO *              ! Delete SLIDER widget
270   END
```

```
10      ! ***********************************************
20      ! Example: STRING Dialog
30      !
40      ! This program creates a STRING dialog. If you move
50      ! the mouse cursor or tab into the text field below
60      ! the prompt, you can input text into the dialog that
70      ! is returned through the S$ string.
80      !
90      ! ***********************************************
100   !
110   DIM P$[25],S$[50]
120   INTEGER Btn
130   P$="Please enter your name"
140   DIALOG "STRING",P$,Btn;SET ("TITLE":" Example: STRING Dialog"),RETURN
("VALUE":S$)
150   SELECT Btn
160   CASE 0
170     DISP "Name entered:",S$
180   CASE 1
190     DISP "Input canceled"
200   END SELECT
210   END
```

```
10      ! ***************************************************************
20      ! Example: STRING Editor
30      !
40      ! This program demonstrates the use of the STRING widget as a text
50      ! editor when in MULTILINE mode.
60      !
70      ! This program does not do much error-checking. A practical program
80      ! should include more error-checking.
90      !
100     ! ***************************************************************
110     !
120     ! Variables Definitions:
130     !
140     !    S$:                General-purpose string
150     !    Search:                    Text search string
160     !    F$:                File string
170     !    N:                 General-purpose variable
180     !    Btn:               Returns button value from dialogs
190     !
200     DIM S$[256],Search$[256],F$[128]
210     INTEGER N,Btn
220     INTEGER Cursor,D(1:4)
230     !
240     ! Widget dimensions
250     !
260     INTEGER Pw,Ph,Px,Py,Iw,Ih,Sw,Sh,Sx,Sy
270     !
280     ! Variables for display scaling
290     !
300     INTEGER Dw,Dh
310     !
320     ! Get display size
330     !
340     GESCAPE CRT,3;D(*)
350     Dw=D(3)-D(1)
360     Dh=D(4)-D(2)
370     CLEAR SCREEN
380     !
390     Pw=Dw*.7                                ! PANEL width
400     Ph=Dh*.7                                ! PANEL height
410     Px=(Dw-Pw)/2                    ! Center PANEL
```

```
420   Py=(Dh-Ph)/2
430   !
440   ! Create PANEL for STRING widget
450   !
460   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
470   CONTROL @Main;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
480   CONTROL @Main;SET ("TITLE":" Example: String Editor")
490   CONTROL @Main;SET ("RESIZABLE":0,"SIZE CONTROL":"RESIZE CHILDREN")
500   CONTROL @Main;SET ("SYSTEM MENU":"Quit")
510   !
520   ! Build "File" menu
530   !
540   ASSIGN @File TO WIDGET "PULLDOWN MENU";PARENT @Main
550   CONTROL @File;SET ("LABEL":"File ")
560   !
570   ASSIGN @New TO WIDGET "MENU BUTTON";PARENT @File
580   CONTROL @New;SET ("LABEL":"New")
590   ASSIGN @Openfile TO WIDGET "MENU BUTTON";PARENT @File
600   CONTROL @Openfile;SET ("LABEL":"Open...")
610   ASSIGN @Savefile TO WIDGET "MENU BUTTON";PARENT @File
620   CONTROL @Savefile;SET ("LABEL":"Save")
630   ASSIGN @Merge TO WIDGET "MENU BUTTON";PARENT @File
640   CONTROL @Merge;SET ("LABEL":"Merge...")
650   !
660   ASSIGN @S1 TO WIDGET "MENU SEPARATOR";PARENT @File
670   !
680   ASSIGN @Quit TO WIDGET "MENU BUTTON";PARENT @File
690   CONTROL @Quit;SET ("LABEL":"Quit")
700   !
710   ! Build "Edit" menu
720   !
730   ASSIGN @Edit TO WIDGET "PULLDOWN MENU";PARENT @Main
740   CONTROL @Edit;SET ("LABEL":"Edit")
750   !
760   ASSIGN @Cut TO WIDGET "MENU BUTTON";PARENT @Edit
770   CONTROL @Cut;SET ("LABEL":"Cut")
780   ASSIGN @Copytext TO WIDGET "MENU BUTTON";PARENT @Edit
790   CONTROL @Copytext;SET ("LABEL":"Copy")
800   ASSIGN @Paste TO WIDGET "MENU BUTTON";PARENT @Edit
810   CONTROL @Paste;SET ("LABEL":"Paste")
820   ASSIGN @Replace TO WIDGET "MENU BUTTON";PARENT @Edit
830   CONTROL @Replace;SET ("LABEL":"Replace")
```

```
840   !
850   ASSIGN @S2 TO WIDGET "MENU SEPARATOR";PARENT @Edit
860   !
870   ASSIGN @Findline TO WIDGET "MENU BUTTON";PARENT @Edit
880   CONTROL @Findline;SET ("LABEL":"Line Number...")
890   ASSIGN @Findstr TO WIDGET "MENU BUTTON";PARENT @Edit
900   CONTROL @Findstr;SET ("LABEL":"String...")
910   !
920   ! Create and size STRING widget, set MULTILINE operation
930   !
940   ASSIGN @S TO WIDGET "STRING";PARENT @Main
950   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
960   CONTROL @S;SET ("X":0,"Y":0,"WIDTH":Iw,"HEIGHT":Ih)
970   CONTROL @S;SET ("MULTILINE":1,"SCROLLBARS":1)
980   !
990   ! Set events
1000  !
1010  ON ERROR GOSUB Errtrap
1020  !
1030  ON EVENT @New,"ACTIVATED" GOSUB Newtext
1040  ON EVENT @Openfile,"ACTIVATED" GOSUB Getfile
1050  ON EVENT @Savefile,"ACTIVATED" GOSUB Savefile
1060  ON EVENT @Merge,"ACTIVATED" GOSUB Mergetext
1070  ON EVENT @Cut,"ACTIVATED" GOSUB Cut
1080  ON EVENT @Copytext,"ACTIVATED" GOSUB Copytext
1090  ON EVENT @Paste,"ACTIVATED" GOSUB Paste
1100  ON EVENT @Replace,"ACTIVATED" GOSUB Replace
1110  ON EVENT @Findline,"ACTIVATED" GOSUB Findline
1120  ON EVENT @Findstr,"ACTIVATED" GOSUB Findstr
1130  !
1140  ON EVENT @Quit,"ACTIVATED" GOTO Finis
1150  ON EVENT @Main,"SYSTEM MENU" GOTO Finis
1160  !
1170  CONTROL @Main;SET ("VISIBLE":1)
1180  !
1190  ! Loop and wait for input
1200  !
1210  LOOP
1220     WAIT FOR EVENT
1230  END LOOP
1240  STOP
1250  !
```

```
1260   ! ******** End of Main Program **********************
1270   !
1280   ! Error handler -- it flag errors and then bypasses the
1290   ! incorrect statement.
1300   !
1310 Errtrap: !
1320   DIALOG "ERROR",ERRM$
1330   ERROR RETURN
1340   !
1350   ! ******** Edit Routines ***************************
1360   !
1370 Newtext: !
1380   STATUS @S;RETURN ("TEXT LENGTH":N)
1390   IF N>0 THEN
1400     DIALOG "QUESTION","Do you want to clear the text?",Btn
1410     IF Btn=0 THEN CONTROL @S;SET ("VALUE":"")
1420   END IF
1430   RETURN
1440   !
1450 Getfile: !
1460   DIALOG "FILE","File to edit?",Btn;RETURN ("SELECTION":F$)
1470   IF Btn=0 THEN CONTROL @S;SET ("VALUE":"","READ FILE":FNValid_name$(F$))
1480   RETURN
1490   !
1500 Savefile: !
1510   DIALOG "FILE","Write to file?",Btn;RETURN ("SELECTION":F$)
1520   IF Btn=0 THEN CONTROL @S;SET ("WRITE FILE":FNValid_name$(F$))
1530   RETURN
1540   !
1550 Mergetext: !
1560   DIALOG "FILE","File to merge?",Btn;RETURN ("SELECTION":F$)
1570   IF Btn=0 THEN CONTROL @S;SET ("READ FILE":FNValid_name$(F$))
1580   RETURN
1590   !
1600 Cut: !
1610   CONTROL @S;SET ("CUT SELECTION":1)
1620   RETURN
1630   !
1640 Copytext: !
1650   CONTROL @S;SET ("COPY SELECTION":1)
1660   RETURN
1670   !
```

```
1680 Paste: !
1690   CONTROL @S;SET ("PASTE":1)
1700   RETURN
1710   !
1720 Replace: !
1730   CONTROL @S;SET ("DELETE SELECTION":1,"PASTE":1)
1740   RETURN
1750   !
1760 Findline: !
1770   S$="Go to what line?"
1780   T$="SHORT INTEGER"
1790   DIALOG "NUMBER",S$,Btn;SET ("FORMAT":T$),RETURN ("VALUE":N)
1800   IF Btn=0 THEN CONTROL @S;SET ("LINE NUMBER":N)
1810   RETURN
1820   !
1830 Findstr: !
1840   S$="Search for what text?"
1850   DIALOG "STRING",S$,Btn;RETURN ("VALUE":Search$)
1860   IF Btn=0 AND Search$<>"" THEN CONTROL @S;SET ("SEARCH":Search$)
1870   RETURN
1880   !
1890 Finis: !
1900   ASSIGN @Main TO *                    ! Deletes PANEL widget
1910   CLEAR SCREEN
1920   END
1930   !
1940   DEF FNValid_name$(F$)
1950     INTEGER Colon_pos
1960     IF POS(SYSTEM$("VERSION:OS"),"HP-UX") THEN Colon_pos=POS(F$,":")
1970     IF NOT Colon_pos THEN Colon_pos=LEN(F$)+1
1980     RETURN F$[1,Colon_pos-1]
1990   FNEND
```

```
10      ! ******************************************
20      ! Example: STRING Widget
30      !
40      ! This program generates a STRING widget and
50      ! allows the user to enter text from the keyboard.
60      ! The text entered by the user is displayed after
70      ! the Enter key is pressed. To exit the program,
80      ! type QUIT and press Enter OR click the PANEL icon
90      ! and then click Quit.
100     !
110     ! ******************************************
120     !
130     CLEAR SCREEN
140     DIM S$[255]
150     ASSIGN @String TO WIDGET "STRING"
160     CONTROL @String;SET ("TITLE":" Example: STRING Widget")
170     CONTROL @String;SET ("X":100,"Y":100,"COLUMNS":30)
180     CONTROL @String;SET ("VALUE":"Enter text here")
190     CONTROL @String;SET ("SYSTEM MENU":"Quit")
200     !
210     ON EVENT @String,"RETURN" GOSUB Handler
220     ON EVENT @String,"SYSTEM MENU" GOTO Finis
230     !
240     LOOP
250       WAIT FOR EVENT
260     END LOOP
270     !
280 Handler: !
290     STATUS @String;RETURN ("VALUE":S$)
300     CONTROL @String;SET ("VALUE":"")
310     IF S$="QUIT" THEN Finis
320     DISP "Text entered:",S$
330     RETURN
340 Finis: !
350     ASSIGN @String TO *             ! Delete STRING widget
360     END
```

```
10      ! ****************************************************************
20      ! Example: STRIPCHART (Counter States)
30      !
40      ! This program demonstrates stripchart operation. It generates
50      ! a PANEL and places a STRIPCHART in it. The STRIPCHART widget
60      ! then displays the timing states of a 7-bit digital counter.
70      !
80      ! ****************************************************************
90      !
100   ! Define colors and array of colors
110   !
120   INTEGER Black,White,Red,Yellow,Green,Cyan,Blue,Magenta,Colors(1:7)
130   DATA 0,1,2,3,4,5,6,7
140   READ Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
150   Colors(1)=White
160   Colors(2)=Red
170   Colors(3)=Yellow
180   Colors(4)=Blue
190   Colors(5)=Cyan
200   Colors(6)=Green
210   Colors(7)=Magenta
220   !
230   ! Other variables:
240   !
250   !   Display(*):                 Used to get display dimensions
260   !   B(*),C(*),D(*),Ctr,Bitstr$:         Used to implement 7-bit counter
270   !   Total:                      Cumulative count of 7-bit counter
280   !   N:                          Useful counter
290   !   Bitstr$:                          Gets binary digits
300   !
310   INTEGER Display(1:4),B(1:7),C(1:7),D(1:7),Ctr,N,Dw,Dh,W,H,X,Y
320   DIM Bitstr$[32]
330   REAL Total
340   !
350   GESCAPE CRT,3;Display(*)
360   Dw=Display(3)-Display(1)
370   Dh=Display(4)-Display(2)
380   CLEAR SCREEN
390   !
400   W=Dw
410   H=Dh
```

```
420    X=0
430    Y=0
440    GOSUB Buildstrip
450    !
460    ! Get ready for main loop. The B(*) and D(*) arrays are used
470    ! to translate the counter count to trace Y coordinates. The Ctr
480    ! variable handles the count and the Total keeps a running tally.
490    !
500    DATA 1,3,5,7,9,11,13,0,0
510    READ B(*),Ctr,Total
520    MAT D=B
530    !
540    ! Set up Quit event, turn on PANEL
550    !
560    ON EVENT @Strip,"SYSTEM MENU" GOTO Finis
570    CONTROL @Strip;SET ("VISIBLE":1)
580    !
590    ! Loop and update traces
600    !
610    LOOP
620       GOSUB Gettrace
630    END LOOP
640    STOP
650    !
660    ! ********** End of Main Program ******************
670    !
680 Gettrace: !
690    !
700    ! The Gettrace routine handles the update of the traces on the display.
710    ! To begin, take the current counter value and use it to update the trace.
720    ! The traces are drawn from one Y value to another:
730    !
740    !      Bit 0:       0 is Y=1       1 is Y=2
750    !      Bit 1:       0 is Y=3       1 is Y=4
760    !      Bit 2:       0 is Y=5       1 is Y=6
770    !      Bit 3:       0 is Y=7       1 is Y=8
780    !      Bit 4:       0 is Y=9       1 is Y=10
790    !      Bit 5:       0 is Y=11      1 is Y=12
800    !      Bit 6:       0 is Y=13      1 is Y=14
810    !
820    ! The B(*) array is effectively simply a set of constants that contains
830    ! the Y-values that correspond to a 0 value for each trace. The C(*)
```

```
840   ! array is generated using the B(*) array and the counter value to give
850   ! the Y-values for the actual current count.
860   !
870   ! To create the C(*) array from the count, the count is converted
880   ! into a string using IVAL that gives the counter value in binary
890   ! (that is, a counter value of 13 gives a string "0000000000001101").
900   !
910   ! Then, the string is scanned from right to left (which scans through
920   ! the count from Bit 0 to Bit 6), and the character "0" or "1" is converted
930   ! to its numeric equivalent. The result is added to the value in B(*)
940   ! for the corresponding bit to load into C(*) for the corresponding bit.
950   !
960   ! The D(*) array keeps the value from the previous ON CYCLE event.
970   ! This is done because when you make a transition from a 0 to a 1
980   ! on a trace you have to plot the old Y value and new Y value at
990   ! the same X value. If you had instead just plotted each new value
1000  ! with each ON CYCLE event, the trace transitions would be plotted as
1010  ! angles with the traces rising and falling between subsequent X values.
1020  !
1030  Bitstr$=IVAL$(Ctr,2)
1040  L=LEN(Bitstr$)
1050  FOR N=1 TO 7
1060     C(N)=B(N)+VAL(Bitstr$[L-(N-1);1])
1070  NEXT N
1080  !
1090  ! Draw the traces -- set X location and then plot the previous trace
1100  ! values followed by the current trace values.
1110  !
1120  CONTROL @Strip;SET ("POINT LOCATION":Total)
1130  CONTROL @Strip;SET ("VALUES":D(*),"VALUES":C(*))
1140  !
1150  ! Save the current trace values, and modulo update the 7-bit counter.
1160  !
1170  MAT D=C
1180  Ctr=(Ctr+1) MOD 128
1190  !
1200  ! Increment the running total. If numeric overflow, reset STRIPCHART
1210  ! and start over again. To do this, you black out all the traces,
1220  ! change the X origin back to 0, draw the traces back to the origin
1230  ! while they are black (so they will be invisible against the black
1240  ! background), and then set the colors again.
1250  !
```

```
1260   Total=Total+1
1270   IF Total=32767 THEN
1280     Total=0
1290     ASSIGN @Strip TO *
1300     GOSUB Buildstrip
1310   END IF
1320   RETURN
1330   !
1340   ! ********** Build Stripchart ************************
1350   !
1360 Buildstrip: !
1370   !
1380   ! Set up main STRIPCHART parameters -- 7 traces, and scroll 1/40th at
1390   ! a time since the traces are 40 units long.
1400   !
1410   ASSIGN @Strip TO WIDGET "STRIPCHART";SET ("VISIBLE":0)
1420   CONTROL @Strip;SET ("TITLE":"Example: STRIPCHART (Counter States)")
1430   CONTROL @Strip;SET ("X":50,"Y":25,"WIDTH":.75*W,"HEIGHT":.75*H)
1440   CONTROL @Strip;SET ("TRACE BACKGROUND":Black)
1450   CONTROL @Strip;SET ("TRACE COUNT":7,"MINIMUM SCROLL":1/40)
1460   !
1470   ! Set up X-axis parameters -- initial origin (before scrolling) of 1,
1480   ! trace display width of 40, keep numbering on and set it to 5 digits.
1490   !
1500   CONTROL @Strip;SET ("CURRENT AXIS":"X","ORIGIN":0,"RANGE":40)
1510   CONTROL @Strip;SET ("NUMBER FORMAT":"FIXED","DIGITS":5)
1520   !
1530   ! Set up Y-axis parameters -- enough range for 7 traces, turn off
1540   ! numbering.
1550   !
1560   CONTROL @Strip;SET ("CURRENT AXIS":"Y")
1570   CONTROL @Strip;SET ("ORIGIN":0,"RANGE":15,"SHOW NUMBERING":0)
1580   !
1590   ! Set up seven traces -- different color and label for each.   (The
1600   ! first 8 traces actually have distinctive colors, but they are set
1610   ! to show how to do it.)
1620   !
1630   FOR N=1 TO 7
1640     CONTROL @Strip;SET ("CURRENT TRACE":N,"TRACE PEN":Colors(N))
1650     CONTROL @Strip;SET ("TRACE LABEL":"BIT "&VAL$(N-1))
1660   NEXT N
1670   !
```

```
1680   CONTROL @Strip;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
1690   CONTROL @Strip;SET ("SYSTEM MENU":"Quit","VISIBLE":1)
1700   RETURN
1710   !
1720   ! ********************   Go Here on Exit ************************
1730   !
1740   ! The ON CYCLE is disabled before destroying the STRIPCHART.
1750   ! Otherwise, it might call the handler and cause an error, since the
1760   ! handler does not have any widgets remaining.
1770   !
1780 Finis: !
1790   OFF CYCLE                       ! Stop ON CYCLE *FIRST!*
1800   ASSIGN @Strip TO *              ! Delete STRIPCHART widget
1810   END
```

```
10      ! ********************************************************
20      ! Example: STRIPCHART (Rescale)
30      !
40      ! This program builds a STRIPCHART widget. As the chart
50      ! scrolls, the vertical scale factor automatically changes
60      ! to accomodate the display values.
70      !
80      ! ********************************************************
90      !
100     INTEGER I
110     REAL D(1:4)
120     ASSIGN @Graph TO WIDGET "STRIPCHART"
130     CONTROL @Graph;SET ("TITLE":"Example: STRIPCHART (Rescale)")
140     CONTROL @Graph;SET ("X":50,"Y":25,"VISIBLE":0)
150     CONTROL @Graph;SET ("CURRENT AXIS":"X","ORIGIN":0,"RANGE":4)
160     CONTROL @Graph;SET ("NUMBER FORMAT":"MINUTES","DIGITS":9)
170     CONTROL @Graph;SET ("CURRENT AXIS":"Y","LOGARITHMIC":1,"ORIGIN":.01)
180     CONTROL @Graph;SET ("RANGE":4,"AUTOSCALE":1,"VISIBLE":1)
190     CONTROL @Graph;SET ("SYSTEM MENU":"Quit")
200     ON EVENT @Graph,"SYSTEM MENU" GOTO Finis
210     T=0
220     WHILE 1
230       T=T+.1
240       FOR I=1 TO 4
250         D(I)=EXP((-62+25*I+10*SIN(T*I*3)+T*(I-2.5))/12)
260       NEXT I
270       WAIT .05
280       CONTROL @Graph;SET ("POINT LOCATION":T,"VALUES":D(*))
290     END WHILE
300 Finis:    !
310     ASSIGN @Graph TO *              ! Delete STRIPCHART widget
320     END
```

```
10      ! ************************************************
20      ! Example: STRIPCHART (Scrolling)
30      !
40      ! This program builds a scrolling STRIPCHART widget
50      ! that displays the values of ten digits.
60      !
70      ! ************************************************
80      !
90      INTEGER I
100   DIM Vals(1:16)
110   ASSIGN @Strip TO WIDGET "STRIPCHART"
120   CONTROL @Strip;SET ("TITLE":" Example: STRIPCHART (Scrolling)")
130   CONTROL @Strip;SET ("SYSTEM MENU":"Quit")
140   CONTROL @Strip;SET ("VISIBLE":0,"SHARED X":1)
150   CONTROL @Strip;SET ("X":50,"Y":25,"WIDTH":400,"HEIGHT":325)
160   CONTROL @Strip;SET ("TRACE COUNT":10,"CURRENT TRACE":0)
170   CONTROL @Strip;SET ("POINT CAPACITY":10000)
180   CONTROL @Strip;SET ("CURRENT AXIS":"X","ORIGIN":0,"RANGE":50)
190   CONTROL @Strip;SET ("DIGITS":11,"NUMBER FORMAT":"FIXED","USER SCROLL":1)
200   CONTROL @Strip;SET ("CURRENT AXIS":"Y","ORIGIN":-.5,"RANGE":10)
210   ON EVENT @Strip,"SYSTEM MENU" GOTO Finis
220   FOR I=1 TO 10
230     Vals(I)=(I-5)*.05
240     CONTROL @Strip;SET ("CURRENT TRACE":I,"TRACE LABEL":"DIGIT "&VAL$(I-1))
250   NEXT I
260   CONTROL @Strip;SET ("VISIBLE":1)
270   ON EVENT @Strip,"SCROLLED" GOSUB Evnt
280   FOR A=0 TO 1000000
290     CONTROL @Strip;SET ("POINT LOCATION":A,"VALUES":Vals(*))
300     Vals(1)=Vals(1)+1
310     I=1
320     WHILE Vals(I)>9.5
330       Vals(I)=Vals(I)-10
340       I=I+1
350       Vals(I)=Vals(I)+1
360     END WHILE
370   NEXT A
380 Evnt: RETURN
390   !
400 Finis:    !
410   ASSIGN @Strip TO *                ! Delete STRIPCHART widget
```

```
420  END
```

```
10      ! ************************************************
20      ! Example: STRIPCHART (Sine Waves)
30      !
40      ! This program builds a scrolling STRIPCHART widget
50      ! that displays four sine waves.
60      !
70      ! ************************************************
80      !
90      INTEGER I
100     REAL D(1:4)
110     ASSIGN @Graph TO WIDGET "STRIPCHART"
120     CONTROL @Graph;SET ("TITLE":" Example: STRIPCHART (Sine Waves)")
130     CONTROL @Graph;SET ("X":50,"Y":25,"VISIBLE":0)
140     CONTROL @Graph;SET ("CURRENT AXIS":"X","ORIGIN":0,"RANGE":4)
150     CONTROL @Graph;SET ("NUMBER FORMAT":"MINUTES","DIGITS":9)
160     CONTROL @Graph;SET ("CURRENT AXIS":"Y","ORIGIN":-50)
170     CONTROL @Graph;SET ("RANGE":100,"VISIBLE":1)
180     CONTROL @Graph;SET ("SYSTEM MENU":"Quit")
190     ON EVENT @Graph,"SYSTEM MENU" GOTO Finis
200     Start=TIMEDATE
210     WHILE 1
220       T=TIMEDATE-Start
230       FOR I=1 TO 4
240         D(I)=-62+25*I+10*SIN(T*I*3)
250       NEXT I
260       CONTROL @Graph;SET ("POINT LOCATION":T,"VALUES":D(*))
270     END WHILE
280 Finis:    !
290     ASSIGN @Graph TO *              ! Delete STRIPCHART widget
300     END
```

```
10      ! ************************************************************
20      ! Example: STRIPCHART Widget
30      !
40      ! This program builds a scrolling STRIPCHART widget
50      ! that displays three triangular waveforms.
60      !
70      ! ************************************************************
80      !
90      DIM Nval(1:3)
100     ASSIGN @Stp TO WIDGET "STRIPCHART"
110     CONTROL @Stp;SET ("X":50,"Y":25,"WIDTH":450,"HEIGHT":300)
120     CONTROL @Stp;SET ("TITLE":" Example: STRIPCHART Widget")
130     CONTROL @Stp;SET ("SYSTEM MENU":"Quit")
140     ON EVENT @Stp,"SYSTEM MENU" GOTO Finis
150     !
160     ! Set three traces, scroll 1/10th of display, update all traces in parallel
170     !
180     CONTROL @Stp;SET ("TRACE COUNT":3,"MINIMUM SCROLL":10,"SHARED X":1)
190     !
200     ! Set X-axis and Y-axis origin, range, and labeling
210     !
220     CONTROL @Stp;SET ("CURRENT AXIS":"Y")
230     CONTROL @Stp;SET ("ORIGIN":-4,"RANGE":8,"AXIS LABEL":"      Y-Axis Information")
240     CONTROL @Stp;SET ("CURRENT AXIS":"X")
250     CONTROL @Stp;SET ("ORIGIN":0,"RANGE":10,"AXIS LABEL":"X-Axis Information")
260     !
270     ! Set three traces and set POINT CAPACITY = 0 (do not save data)
280     !
290     CONTROL @Stp;SET ("CURRENT TRACE":1)
300     CONTROL @Stp;SET ("TRACE PEN":1,"TRACE LABEL":"Trace 1")
310     CONTROL @Stp;SET ("CURRENT TRACE":2)
320     CONTROL @Stp;SET ("TRACE PEN":2,"TRACE LABEL":"Trace 2")
330     CONTROL @Stp;SET ("CURRENT TRACE":3)
340     CONTROL @Stp;SET ("TRACE PEN":3,"TRACE LABEL":"Trace 3")
350     CONTROL @Stp;SET ("CURRENT TRACE":0,"POINT CAPACITY":0)
360     !
370     ! Display a pattern of triangle waves
380     !
390     DATA 1,-1,3
400     READ Nval(*)
410     FOR N=0 TO 1000
```

```
420       WAIT 1
430       CONTROL @Stp;SET ("POINT LOCATION":N,"VALUES":Nval(*))
440       Nval(1)=-Nval(1)
450       Nval(2)=-Nval(2)
460       Nval(3)=-Nval(3)
470     NEXT N
480     !
490 Finis:    !
500     ASSIGN @Stp TO *                  ! Delete STRIPCHART widget
510     END
```

```
10      ! ***************************************************
20      ! Example: SYSTEM Widget Event Disabling
30      !
40      ! This program builds a set of widgets using the
50      ! SYSTEM widget. You can enable/disable ten events.
60      ! In addition, a status log lists all events that
70      ! occur, such as moving the SLIDER bar, etc.
80      !
90      !                       NOTE
100     !
110     ! You may need to maximize the display to see all
120     ! of the widgets.
130     !
140     ! ***************************************************
150     !
160     DIM Sys_menu$(0:13)[80],Event$(1:2)[80],Panel$[80]
170     DIM Printer$[80],Slider$[80],Meter$[80]
180     INTEGER I,Space,Sys_event,Cur_event,Events
190     DATA ACTIVATED, ALARM, CHANGED, DONE, INVALID NUMBER
200     DATA KEYSTROKE, REPAINT, RESIZED, RETURN, SELECTION, *
210     Events=0
220     LOOP
230       Sys_menu$(Events)="DISABLE "
240       READ Sys_menu$(Events)[9]
250     EXIT IF Sys_menu$(Events)[9]="*"
260       Events=Events+1
270     END LOOP
280     Sys_menu$(Events)="Quit "
290     REDIM Sys_menu$(0:Events)
300     Panel$="Panel"
310     Printer$=Panel$&"/Printer"
320     Slider$=Panel$&"/Slider"
330     Meter$=Panel$&"/Meter"
340     !
350     ASSIGN @Syst TO WIDGET "SYSTEM"
360     CONTROL @Syst;SET ("*LOAD":"SBEVNTSA.SCR","*QUEUE EVENTS":1)
370     CONTROL @Syst;SET ("*NAME":"Panel","SYSTEM MENU":Sys_menu$(*))
380     CONTROL @Syst;SET ("SIZE CONTROL":"RESIZE CHILDREN")
390     ON EVENT @Syst,"SYSTEM MENU" GOSUB Sys_menu
400     ON Events GOTO E0,E1,E2,E3,E4,E5,E6,E7,E8,E9,E10,E11,E12
410     !
```

```
420 E12: ON EVENT @Syst,Sys_menu$(12)[POS(Sys_menu$(12)," ")+1] GOSUB Event_12
430 E11: ON EVENT @Syst,Sys_menu$(11)[POS(Sys_menu$(11)," ")+1] GOSUB Event_11
440 E10: ON EVENT @Syst,Sys_menu$(10)[POS(Sys_menu$(10)," ")+1] GOSUB Event_10
450 E9: ON EVENT @Syst,Sys_menu$(9)[POS(Sys_menu$(9)," ")+1] GOSUB Event_9
460 E8: ON EVENT @Syst,Sys_menu$(8)[POS(Sys_menu$(8)," ")+1] GOSUB Event_8
470 E7: ON EVENT @Syst,Sys_menu$(7)[POS(Sys_menu$(7)," ")+1] GOSUB Event_7
480 E6: ON EVENT @Syst,Sys_menu$(6)[POS(Sys_menu$(6)," ")+1] GOSUB Event_6
490 E5: ON EVENT @Syst,Sys_menu$(5)[POS(Sys_menu$(5)," ")+1] GOSUB Event_5
500 E4: ON EVENT @Syst,Sys_menu$(4)[POS(Sys_menu$(4)," ")+1] GOSUB Event_4
510 E3: ON EVENT @Syst,Sys_menu$(3)[POS(Sys_menu$(3)," ")+1] GOSUB Event_3
520 E2: ON EVENT @Syst,Sys_menu$(2)[POS(Sys_menu$(2)," ")+1] GOSUB Event_2
530 E1: ON EVENT @Syst,Sys_menu$(1)[POS(Sys_menu$(1)," ")+1] GOSUB Event_1
540 E0: ON EVENT @Syst,Sys_menu$(0)[POS(Sys_menu$(0)," ")+1] GOSUB Event_0
550    !
560    LOOP
570       WAIT FOR EVENT
580    END LOOP
590    !
600 Event_0: Cur_event=0
610    GOSUB Event_handler
620    RETURN
630    !
640 Event_1: Cur_event=1
650    GOSUB Event_handler
660    RETURN
670    !
680 Event_2: Cur_event=2
690    GOSUB Event_handler
700    RETURN
710    !
720 Event_3: Cur_event=3
730    GOSUB Event_handler
740    RETURN
750    !
760 Event_4: Cur_event=4
770    GOSUB Event_handler
780    RETURN
790    !
800 Event_5: Cur_event=5
810    GOSUB Event_handler
820    RETURN
830    !
```

```
840 Event_6: Cur_event=6
850     GOSUB Event_handler
860     RETURN
870     !
880 Event_7: Cur_event=7
890     GOSUB Event_handler
900     RETURN
910     !
920 Event_8: Cur_event=8
930     GOSUB Event_handler
940     RETURN
950     !
960 Event_9: Cur_event=9
970     GOSUB Event_handler
980     RETURN
990     !
1000 Event_10: Cur_event=10
1010    GOSUB Event_handler
1020    RETURN
1030    !
1040 Event_11: Cur_event=11
1050    GOSUB Event_handler
1060    RETURN
1070    !
1080 Event_12: Cur_event=12
1090    GOSUB Event_handler
1100    RETURN
1110    !
1120 Event_handler:!
1130    CONTROL @Syst;SET ("*EVENT NAME FILTER":Sys_menu$(Cur_event)
[POS(Sys_menu$(Cur_event)," ")+1])
1140    LOOP
1150      STATUS @Syst;RETURN ("*QUEUED EVENT":Event$(*))
1160    EXIT IF NOT LEN(Event$(2))
1170      IF Event$(1)=Slider$ THEN
1180        STATUS @Syst;RETURN ("*NAME":Slider$,"VALUE":Value)
1190        CONTROL @Syst;SET ("*NAME":Meter$,"VALUE":Value)
1200      END IF
1210      CONTROL @Syst;SET ("*NAME":Printer$,"APPEND TEXT":Event$(1)&":"&Event$(2))
1220    END LOOP
1230    CONTROL @Syst;SET ("*EVENT NAME FILTER":"")
1240    RETURN
```

```
1250    !
1260 Sys_menu:!
1270   STATUS @Syst;RETURN ("*NAME":Panel$,"SYSTEM MENU EVENT":Sys_event)
1280   Space=POS(Sys_menu$(Sys_event)," ")
1290   SELECT Sys_menu$(Sys_event)[1,Space-1]
1300   CASE "ENABLE"
1310     ENABLE EVENT @Syst,Sys_menu$(Sys_event)[Space+1]
1320     Action$="DISABLE"
1330   CASE "DISABLE"
1340     DISABLE EVENT @Syst,Sys_menu$(Sys_event)[Space+1]
1350     Action$="ENABLE"
1360   CASE "Quit"
1370     STOP
1380   END SELECT
1390   CONTROL @Syst;SET ("*NAME":Printer$,"APPEND TEXT":Sys_menu$(Sys_event))
1400   Sys_menu$(Sys_event)=Action$&Sys_menu$(Sys_event)[Space]
1410   CONTROL @Syst;SET ("*NAME":"Panel","SYSTEM MENU":Sys_menu$(*))
1420   CONTROL @Syst;SET ("*EVENT NAME FILTER":"SYSTEM MENU")
1430   STATUS @Syst;RETURN ("*FLUSH QUEUED EVENTS":Events)
1440   CONTROL @Syst;SET ("*EVENT NAME FILTER":"")
1450   RETURN
1460   END
```

```
10      ! ************************************************************
20      ! Example: SYSTEM Widget Event Handler
30      !
40      ! This program shows how to handle queued events from a SYSTEM
50      ! widget.   A SYSTEM widget has an attribute named *QUEUE EVENTS,
60      ! which is normally turned off. In that state, the SYSTEM widget
70      ! will not queue up events -- you get one and all later ones are lost.
80      !
90      ! If you set *QUEUE EVENTS to 1, however, events are queued --
100     ! which leads to the problem of sorting out events from each other.
110     !
120     ! This program shows how this is done by queueing up events from
130     ! three SLIDER widgets contained in a SYSTEM widget. When you
140     ! tell the program to flush the queue, it lists the queued events
150     ! by source SLIDER and event type and gives the number of
160     ! events in each category.
170     !
180     ! This is a somewhat contrived program in that the level of
190     ! event handling demonstrated here is overkill.   Under normal
200     ! circumstances, event queueing should not be a concern. In
210     ! fact, this program has to go to considerable lengths to force
220     ! the events to be queued -- by raising the SYSTEM PRIORITY
230     ! to mask out the SLIDER events, and only lowering it when
240     ! it polls a TOGGLEBUTTON and gets back a 1.
250     !
260     ! The SYSTEM MENU event that allows you to exit the program
270     ! is set to a high priority level, so you can exit the program at
280     ! any time.
290     !
300     ! ************************************************************
310     !
320     CLEAR SCREEN
330     INTEGER N                       ! General-purpose variable
340     INTEGER D(1:4),Dw,Dh,Iw,Ih      ! Variables for display handling
350     INTEGER Px,Py,Pw,Ph             ! PANEL parameters
360     INTEGER Gx,Gy                   ! Internal layout gaps
370     INTEGER Nc,R,Sx,Sy,Sw,Sh        ! Parameters for SYSTEM widget SLIDERS
380     INTEGER Prx,Pry,Prw,Prh                 ! Parameters for PRINTER widget
390     INTEGER Tx,Ty,Th,Tw             ! Parameters for TOGGLE BUTTON
400     DIM Ev$(1:2)[50],S$[50]         ! Return event source - GP string
410     !
```

```
420   ! Set up a PANEL in the center of the printing display and put
430   ! a SYSTEM MENU on it so you can quit the program.
440   !
450   STATUS CRT,13;N
460   GESCAPE CRT,3;D(*)
470   Dw=D(3)-D(1)
480   Dh=(D(4)-D(2))*((N-7)/N)
490   !
500   Pw=Dw*.99
510   Ph=Dh*.99
520   Px=(Dw-Pw)/2
530   Py=(Dh-Ph)/2
540   !
550   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
560   CONTROL @Main;SET ("X":Px+100,"Y":Py+50,"WIDTH":.75*Pw,"HEIGHT":.75*Ph)
570   CONTROL @Main;SET ("SYSTEM MENU":"Quit")
580   CONTROL @Main;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
590   CONTROL @Main;SET ("TITLE":" Example: SYSTEM Widget Event Handling")
600   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
610   !
620   ! Create a SYSTEM widget as a child of the main PANEL, populate
630   ! with SLIDER widgets.
640   !
650   ASSIGN @Sys TO WIDGET "SYSTEM";PARENT @Main
660   !
670   Gx=Iw*.01
680   Gy=Gx
690   !
700   Nc=3
710   Sh=Ih*.85
720   Sw=Iw*.11
730   Sy=Gy
740   !
750   FOR C=1 TO Nc
760     Sx=(C-1)*Sw+Gy
770     S$="S"&VAL$(C)
780     CONTROL @Sys;SET ("*NAME":S$,"*CREATE":"SLIDER")
790     CONTROL @Sys;SET ("X":Sx,"Y":Sy,"WIDTH":Sw,"HEIGHT":Sh)
800   NEXT C
810   !
820   ! Set up a TOGGLEBUTTON to indicate when to trap events
830   !
```

```
840   Th=Ih-(Sh+3*Gy)
850   Tw=Iw/2
860   Tx=Gx+(3*Sw-Tw)/2
870   Ty=Ih-(Th+Gy)
880   ASSIGN @Toggle TO WIDGET "TOGGLEBUTTON";PARENT @Main
890   CONTROL @Toggle;SET ("X":Tx+50,"Y":Ty,"WIDTH":Tw,"HEIGHT":Th)
900   CONTROL @Toggle;SET ("LABEL":"Flush Queue")
910   !
920   ! Set up a PRINTER widget on the main PANEL (not as part of
930   ! the SYSTEM widget) to log event handling
940   !
950   Prw=Iw-(3*Sw+3*Gy)
960   Prh=Ih-2*Gy
970   Prx=Iw-(Prw+Gy)
980   Pry=Gy
990   !
1000  ASSIGN @Prn TO WIDGET "PRINTER";PARENT @Main
1010  CONTROL @Prn;SET ("X":Prx,"Y":Pry,"WIDTH":Prw,"HEIGHT":Prh)
1020  !
1030  ! Set up events to trap the SLIDERs in the SYSTEM widget, and
1040  ! set up a high priority trap on the "Quit" button in the SYSTEM
1050  ! MENU
1060  !
1070  CONTROL @Sys;SET ("*QUEUE EVENTS":1)
1080  ON EVENT @Sys,"CHANGED" GOSUB Handler
1090  ON EVENT @Sys,"DONE" GOSUB Handler
1100  ON EVENT @Main,"SYSTEM MENU",15 GOTO Finis
1110  !
1120  ! Set event to quit on SYSTEM MENU, make whole thing visible, loop
1130  ! -- mask out normal events until user toggles the TOGGLEBUTTON
1140  !
1150  CONTROL @Main;SET ("VISIBLE":1)
1160  LOOP
1170    SYSTEM PRIORITY 1
1180    REPEAT
1190      STATUS @Toggle;RETURN ("VALUE":N)
1200    UNTIL (N<>0)
1210    CONTROL @Toggle;SET ("VALUE":0)
1220    CONTROL @Prn;SET ("TEXT":"")
1230    SYSTEM PRIORITY 0
1240  END LOOP
1250  !
```

```
1260   ! This event handler confirms SLIDER events by printing out the
1270   ! event and its source. It performs a loop, checking to see
1280   ! if there are events in the queue. If there are, it reads the
1290   ! first event in the queue and flushes all the same events.
1300   ! (You could also read them out if you like.) It continues
1310   ! doing this until it cleans out the queue.
1320   !
1330 Handler: !
1340   LOOP
1350   !
1360   ! Determine total number of events in queue
1370   !
1380     CONTROL @Sys;SET ("*EVENT WIDGET FILTER":"")
1390     CONTROL @Sys;SET ("*EVENT NAME FILTER":"")
1400     STATUS @Sys;RETURN ("*QUEUED EVENTS":N)
1410   !
1420   EXIT IF N=0
1430   !
1440     S$="> Total events in queue:   "&VAL$(N)
1450     CONTROL @Prn;SET ("APPEND TEXT":S$)
1460     !
1470     ! Get ID of first source in queue.
1480     !
1490     STATUS @Sys;RETURN ("*QUEUED EVENT":Ev$(*))
1500     !
1510     ! Determine how many of the same events are in the
1520     ! queue and then flush them
1530     !
1540     CONTROL @Sys;SET ("*EVENT WIDGET FILTER":Ev$(1))
1550     CONTROL @Sys;SET ("*EVENT NAME FILTER":Ev$(2))
1560     STATUS @Sys;RETURN ("*FLUSH QUEUED EVENTS":N)
1570     !
1580     S$="   Widget / Event / Total: "
1590     S$=S$&Ev$(1)&" / "&Ev$(2)&" / "&VAL$(N+1)
1600     CONTROL @Prn;SET ("APPEND TEXT":S$)
1610     !
1620     CONTROL @Prn;SET ("APPEND TEXT":"")
1630     !
1640   END LOOP
1650   RETURN
1660   !
1670 Finis: !
```

```
1680   ASSIGN @Main TO *                ! Delete PANEL widget
1690   CLEAR SCREEN
1700   END
```

```
10      ! ******************************************************
20      ! Example: SYSTEM Widget Event Queuing
30      !
40      ! This program shows one way to display queued
50      ! events. You can set the high limit and the low
60      ! limit for the METER widget. If you set QueueEvent,
70      ! an event description is displayed each time the
80      ! value crosses the high or low limit value.
90      !
100     ! ******************************************************
110     !
120     COM @System
130     INTEGER Panel,Slider,Meter,Bar,Toggle,Printer
140     INTEGER Keep_looping,Queuing,Queued,Flushed,System_event,Btn
150     REAL Min,Max,Low_lim,High_lim
160     DIM Event$(0:1)[80]
170     !
180     Panel=1
190     Slider=2
200     Meter=3
210     Bar=4
220     Toggle=8
230     Printer=9
240     Keep_looping=1
250     !
260     ASSIGN @System TO WIDGET "SYSTEM";SET ("*LOAD":"SBEVNTS.SCR")
270     STATUS @System;RETURN ("*WIDGETS":Num_widgets)
280     ALLOCATE Names$(1:Num_widgets)[80]
290     STATUS @System;RETURN ("*WIDGET NAMES":Names$(*))
300     GOSUB Init
310     !
320     ! Set up events
330     !
340     CONTROL @System;SET ("*NAME":Names$(Meter),"ALARM TYPE":"EVENT","ALARM
RANGES":"HIGH,MIDDLE,LOW")
350     ON EVENT @System,"CHANGED" GOSUB Event_handler
360     ON EVENT @System,"ALARM",2 GOSUB Event_handler
370     ON EVENT @System,"ACTIVATED" GOSUB Event_handler
380     ON EVENT @System,"SYSTEM MENU" GOSUB Event_handler
390     !
400     REPEAT
```

```
410       WAIT FOR EVENT
420     UNTIL NOT Keep_looping
430     !
440     STOP
450     !
460 Init:!
470     Queuing=0
480     CONTROL @System;SET ("*NAME":Names$(Toggle),"VALUE":Queuing,"*QUEUE
EVENTS":Queuing)
490     Min=0.
500     Max=100.0
510     Low_lim=25.0
520     High_lim=75.0
530     GOSUB Set_limits
540     CONTROL @System;SET ("*NAME":Names$(Slider),"VALUE":Min)
550     CONTROL @System;SET ("*NAME":Names$(Meter),"VALUE":Min)
560     CONTROL @System;SET ("*NAME":Names$(Bar),"VALUE":Min)
570     !
580     ! Send names to PRINTER widget
590     !
600     CONTROL @System;SET ("*NAME":Names$(Printer),"HIDDEN LINES":1000)
610     CONTROL @System;SET ("TEXT":"This example contains "&VAL$(Num_widgets)&"
widgets:")
620     CONTROL @System;SET ("APPEND TEXT":"")
630     CONTROL @System;SET ("APPEND TEXT":Names$(*))
640     CONTROL @System;SET ("APPEND TEXT":"")
650     RETURN
660     !
670 Set_limits:!
680     CONTROL @System;SET ("*NAME":Names$(Meter),"LOW LIMIT":Low_lim,"HIGH
LIMIT":High_lim)
690     CONTROL @System;SET ("*NAME":Names$(Bar),"LOW LIMIT":Low_lim,"HIGH
LIMIT":High_lim)
700     RETURN
710     !
720 Event_handler:!
730     REPEAT
740       STATUS @System;RETURN ("*QUEUED EVENT":Event$(*),"*QUEUED
EVENTS":Queued)
750       SELECT Event$(1)
760       CASE "ACTIVATED"
770         SELECT VAL(Event$(0)[LEN(Event$(0))])
780         CASE 1
```

```
790           DIALOG "NUMBER","Enter new LOW LIMIT",Btn;SET ("REAL
NOTATION":"FIXED","REAL
RESOLUTION":3,"MINIMUM":Min,"MAXIMUM":Max,"VALUE":Low_lim),RETURN
("VALUE":Value)
800           IF Btn=0 THEN
810             Low_lim=Value
820             GOSUB Set_limits
830           END IF
840         CASE 2
850           DIALOG "NUMBER","Enter new HIGH LIMIT",Btn;SET ("REAL
NOTATION":"FIXED","REAL
RESOLUTION":3,"MINIMUM":Min,"MAXIMUM":Max,"VALUE":High_lim),RETURN
("VALUE":Value)
860           IF Btn=0 THEN
870             High_lim=Value
880             GOSUB Set_limits
890           END IF
900         END SELECT
910       CASE "CHANGED"
920         IF Queued THEN GOSUB Flush_queue
930         SELECT Event$(0)
940         CASE Names$(Slider)
950           STATUS @System;RETURN ("*NAME":Names$(Slider),"VALUE":Value)
960           CONTROL @System;SET ("*NAME":Names$(Meter),"VALUE":Value)
970         CASE Names$(Toggle)
980           STATUS @System;RETURN ("*NAME":Names$(Toggle),"VALUE":Queuing)
990           CONTROL @System;SET ("*QUEUE EVENTS":Queuing)
1000        END SELECT
1010      CASE "ALARM"
1020        IF Queued THEN GOSUB Flush_queue
1030        STATUS @System;RETURN ("*NAME":Names$(Meter),"VALUE":Value)
1040        CONTROL @System;SET ("*NAME":Names$(Bar),"VALUE":Value)
1050      CASE ""                                   ! No events queued
1060      CASE ELSE                         ! SYSTEM MENU
1070        IF Queuing THEN
1080          System_event=VAL(Event$(1)[13])
1090        ELSE
1100          STATUS @System;RETURN ("*NAME":Names$(Panel),"SYSTEM MENU
EVENT":System_event)
1110        END IF
1120        SELECT System_event
1130        CASE 0
1140          GOSUB Init
1150        CASE 1
```

```
1160            DIALOG "QUESTION","Do you want to quit?",Keep_looping
1170          END SELECT
1180        END SELECT
1190     UNTIL NOT Queued
1200     RETURN
1210     !
1220 Flush_queue:   !
1230     CONTROL @System;SET ("*EVENT WIDGET FILTER":Event$(0),"*EVENT NAME
FILTER":Event$(1))
1240     STATUS @System;RETURN ("*FLUSH QUEUED EVENTS":Flushed,"*QUEUED
EVENTS":Queued)
1250     CONTROL @System;SET ("*EVENT WIDGET FILTER":"","*EVENT NAME FILTER":"")
1260     IF Flushed THEN
1270        CONTROL @System;SET ("*NAME":Names$(Printer),"APPEND TEXT":"Flushed
"&VAL$(Flushed)&" "&Event$(0)&":"&Event$(1)&" events.")
1280        CONTROL @System;SET ("APPEND TEXT":"    "&VAL$(Queued)&" events remaining")
1290     END IF
1300     RETURN
1310     END
```

```
10      ! ************************************************************
20      ! Example: SYSTEM Widget As a Child
30      !
40      ! This program shows the user of the SYSTEM widget as a child
50      ! widget. The program displays a PANEL widget, puts a PRINTER
60      ! widget on it, and then creates a SYSTEM widget containing an
70      ! array of buttons. When you click a button, the event is listed
80      ! in the PRINTER widget.
90      !
100   ! ************************************************************
110   !
120   CLEAR SCREEN
130   INTEGER D(1:4),Dw,Dh,Nlines             ! Display handling variables
140   INTEGER Px,Py,Pw,Ph                     ! Main PANEL parameters
150   INTEGER Gx,Gy                           ! Internal spacing
160   INTEGER Nr,Nc,R,C,Bx,By,Bw,Bh                   ! PUSHBUTTON variables
170   INTEGER Prx,Pry,Prw,Prh                         ! PRINTER parameters
180   DIM Ev$(1:2)[50],S$[50]                 ! Returns event source - GP string
190   !
200   ! Set up a PANEL in the center of the printing display (above the
210   ! display (above the softkeys, etc.) and put a SYSTEM MENU on it
220   ! so you can quit the program.
230   !
240   STATUS CRT,13;Nlines
250   GESCAPE CRT,3;D(*)
260   Dw=D(3)-D(1)
270   Dh=(D(4)-D(2))*((Nlines-7)/Nlines)
280   !
290   Pw=Dw*.9
300   Ph=Dh*.9
310   Px=(Dw-Pw)/2
320   Py=(Dh-Ph)/2
330   !
340   ASSIGN @Main TO WIDGET "PANEL";SET ("VISIBLE":0)
350   CONTROL @Main;SET ("X":Px,"Y":Py,"WIDTH":Pw,"HEIGHT":Ph)
360   CONTROL @Main;SET ("SYSTEM MENU":"Quit")
370   CONTROL @Main;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
380   CONTROL @Main;SET ("TITLE":" Example: SYSTEM Widget As a Child")
390   STATUS @Main;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
400   !
410   ! Create a SYSTEM widget as a child of the main PANEL and
```

```
420   ! populate with PUSHBUTTONS.
430   !
440   ASSIGN @Sys TO WIDGET "SYSTEM";PARENT @Main
450   !
460   Gx=Iw*.02
470   Gy=Gx
480   !
490   Nr=9
500   Nc=6
510   Bh=(Ih-2*Gy)/Nr
520   Bw=Iw*.08
530   !
540   FOR R=1 TO Nr
550     By=Gy+(R-1)*Bh
560     FOR C=1 TO Nc
570       Bx=Gx+(C-1)*Bw
580       S$="B"&VAL$(R)&VAL$(C)
590       CONTROL @Sys;SET ("*NAME":S$,"*CREATE":"PUSHBUTTON")
600       CONTROL @Sys;SET ("X":Bx,"Y":By,"WIDTH":Bw,"HEIGHT":Bh)
610       CONTROL @Sys;SET ("LABEL":S$)
620     NEXT C
630   NEXT R
640   !
650   ! Set up a PRINTER widget on the main PANEL (not as part of
660   ! the SYSTEM widget) to log events.
670   !
680   Prw=Iw-(Nc*Bw+3*Gx)
690   Prh=Ih-2*Gy
700   Prx=Iw-(Prw+Gy)
710   Pry=Gy
720   ASSIGN @Prn TO WIDGET "PRINTER";PARENT @Main
730   CONTROL @Prn;SET ("X":Prx,"Y":Pry,"WIDTH":Prw,"HEIGHT":Prh)
740   !
750   ! Set up an event to trap the PUSHBUTTONs in the SYSTEM widget.
760   !
770   ON EVENT @Sys,"ACTIVATED" GOSUB Handler
780   !
790   ! Set event to quit on SYSTEM MENU, make whole thing visible,
800   ! loop and wait for an event.
810   !
820   ON EVENT @Main,"SYSTEM MENU" GOTO Finis
830   CONTROL @Main;SET ("VISIBLE":1)
```

```
840    !
850    LOOP
860       WAIT FOR EVENT
870    END LOOP
880    STOP
890    !
900    ! This event handler confirms PUSHBUTTON events by printing out
910    ! the event and its source.
920    !
930 Handler:   !
940    STATUS @Sys;RETURN ("*QUEUED EVENT":Ev$(*))
950    S$="Name/Event: "&Ev$(1)&"/"&Ev$(2)
960    CONTROL @Prn;SET ("APPEND TEXT":S$)
970    RETURN
980    !
990 Finis:   !
1000   ASSIGN @Main TO *                  ! Delete PANEL widget
1010   CLEAR SCREEN
1020   END
```

```
10      ! ************************************************************************
20      ! Example: Slot Machine
30      !
40      ! This example program simulates a slot machine.
50      ! It illustrates the use of STRING widgets with different fonts, and
60      ! shows how attribute arrays can be used. The PUSHBUTTON object is used
70      ! to initiate an action. The PRINTER widget is used to display information
80      ! to the user.
90      !
100   ! The arrays Attr_n$ and Attr are used to program six numeric attributes
110   ! at once.   The arrays Attr_s$ and Attr_sv$ are used to program two string
120   ! attributes.
130   !
140   ! ************************************************************************
150   !
160   DIM Attr_n$(5)[11],Attr_s$(1)[10],Attr(5),Attr_sv$(1)[10],Poss$(6)[6]
170   Attr_n$(0)="X"
180   Attr_n$(1)="Y"
190   Attr_n$(2)="WIDTH"
200   Attr_n$(3)="HEIGHT"
210   Attr_n$(4)="RESIZABLE"
220   Attr_n$(5)="MAXIMIZABLE"
230   Attr(0)=100                 ! x position
240   Attr(1)=20                          ! y position
250   Attr(2)=120                 ! width
260   Attr(3)=50                          ! height
270   Attr(4)=0                   ! resizable
280   Attr(5)=0                   ! maximizable
290   Attr_s$(0)="FONT"
300   Attr_s$(1)="TITLE"
310   Attr_sv$(0)="20 BY 30"
320   Attr_sv$(1)=""              ! Makes the title area disappear
330   !
340   ASSIGN @Slot1 TO WIDGET "STRING"
350   CONTROL @Slot1;SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$(*))
360   CONTROL @Slot1;SET ("BACKGROUND":1,"PEN":2)
370   !
380   ! Position the second STRING widget to the right of the first
390   !
400   Attr(0)=Attr(0)+Attr(2)
410   ASSIGN @Slot2 TO WIDGET "STRING"
```

```
420   CONTROL @Slot2;SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$(*))
430   CONTROL @Slot2;SET ("BACKGROUND":1,"PEN":2)
440   !
450   ! Position the third STRING widget to the right of the second
460   !
470   Attr(0)=Attr(0)+Attr(2)
480   ASSIGN @Slot3 TO WIDGET "STRING"
490   CONTROL @Slot3;SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$(*))
500   CONTROL @Slot3;SET ("BACKGROUND":1,"PEN":2)
510   !
520   ! Create a button to quit the game
530   !
540   ASSIGN @Quit TO WIDGET "PUSHBUTTON"
550   CONTROL @Quit;SET ("X":150,"Y":100,"RESIZABLE":0,"MAXIMIZABLE":0)
560   CONTROL @Quit;SET ("TITLE":"","FONT":"20 BY 30","LABEL":" QUIT ")
570   ON EVENT @Quit,"ACTIVATED" GOTO Finis
580   !
590   ! Create a button to simulate pulling the lever
600   !
610   ASSIGN @Pull TO WIDGET "PUSHBUTTON"
620   CONTROL @Pull;SET ("X":300,"Y":100,"RESIZABLE":0,"MAXIMIZABLE":0)
630   CONTROL @Pull;SET ("TITLE":"","FONT":"20 BY 30","LABEL":" PULL ")
640   !
650   ! The value remaining in the bankroll is displayed in another string
660   ! widget which retains its title area
670   !
680   Bankroll=100
690   ASSIGN @Bank TO WIDGET "STRING"
700   CONTROL @Bank;SET ("X":300,"Y":175,"MAXIMIZABLE":0)
710   CONTROL @Bank;SET ("TITLE":" BANKROLL","WIDTH":150)
720   CONTROL @Bank;SET ("VALUE":"   $"&VAL$(Bankroll))
730   !
740   ! Explain the costs and payoffs in a PRINTER widget
750   !
760   ASSIGN @Info TO WIDGET "PRINTER"
770   CONTROL @Info;SET ("X":100,"Y":175,"ROWS":5,"COLUMNS":21)
780   CONTROL @Info;SET ("RESIZABLE":0,"MAXIMIZABLE":0,"TITLE":" ODDS")
790   CONTROL @Info;SET ("TEXT":"Each pull costs $5")
800   CONTROL @Info;SET ("APPEND TEXT":"3 bars        pays $100")
810   CONTROL @Info;SET ("APPEND TEXT":"Any 2 bars pays $20")
820   CONTROL @Info;SET ("APPEND TEXT":"Any 2 7s     pays $10")
830   CONTROL @Info;SET ("APPEND TEXT":"Any pair     pays $5")
```

```
840   !
850   ! This array contains the possible values on the wheels
860   !
870   Poss$(0)=" BAR"
880   Poss$(1)=" BELL"
890   Poss$(2)="APPLE"
900   Poss$(3)="CHERRY"
910   Poss$(4)=" PLUM"
920   Poss$(5)="   7"
930   !
940   ! When the pull button is clicked, get some new values
950   !
960   ON EVENT @Pull,"ACTIVATED",1 GOSUB Pulled
970   !
980   LOOP
990     WAIT FOR EVENT
1000  END LOOP
1010 Pulled:!
1020  Bankroll=Bankroll-5
1030  CONTROL @Bank;SET ("VALUE":"   $"&VAL$(Bankroll))
1040  FOR I=1 TO 8+RND*8
1050    Val1=INT(RND*6)
1060    CONTROL @Slot1;SET ("VALUE":Poss$(Val1))
1070    WAIT .05
1080    Val2=INT(RND*6)
1090    CONTROL @Slot2;SET ("VALUE":Poss$(Val2))
1100    WAIT .05
1110    Val3=INT(RND*6)
1120    CONTROL @Slot3;SET ("VALUE":Poss$(Val3))
1130    WAIT .05
1140  NEXT I
1150  !
1160  ! Assume no payoff
1170  !
1180  Win=0
1190  !
1200  ! Check for a pair
1210  !
1220  IF Val1=Val2 OR Val1=Val3 OR Val2=Val3 THEN Win=5
1230  !
1240  ! Check for a pair of 7s
1250  !
```

```
1260   IF (Val1=5 AND (Val1=Val2 OR Val1=Val3)) OR (Val2=5 AND (Val2=Val3)) THEN Win=10
1270   !
1280   ! Check for a pair of bars
1290   !
1300   IF (Val1=0 AND (Val1=Val2 OR Val1=Val3)) OR (Val2=0 AND (Val2=Val3)) THEN Win=20
1310   !
1320   ! Check for three bars
1330   !
1340   IF Val1=0 AND Val2=0 AND Val3=0 THEN Win=100
1350   !
1360   ! Declare any winnings
1370   !
1380   IF Win>0 THEN DIALOG "INFORMATION","You WON $"&VAL$(Win);TIMEOUT 10
1390   Bankroll=Bankroll+Win
1400   CONTROL @Bank;SET ("VALUE":"    $"&VAL$(Bankroll))
1410   IF Bankroll<=0 THEN STOP
1420   RETURN
1430   !
1440 Finis:   !
1450   ASSIGN @Slot1 TO *              ! Delete STRING widget
1460   ASSIGN @Slot2 TO *              ! Delete STRING widget
1470   ASSIGN @Slot3 TO *              ! Delete STRING widget
1480   ASSIGN @Quit TO *               ! Delete PUSHBUTTON widget
1490   ASSIGN @Pull TO *               ! Delete PUSHBUTTON widget
1500   ASSIGN @Bank TO *               ! Delete STRING widget
1510   ASSIGN @Info TO *               ! Delete PRINTER widget
1520   END
```

```
10      ! ********************************************************
20      ! Example: TOGGLEBUTTON Widget
30      !
40      ! This program uses a TOGGLEBUTTON widget. It provides a
50      ! row of TOGGLEBUTTONS and lists the button and its state
60      ! as the button is pressed.
70      !
80      ! ********************************************************
90      !
100   ! Define colors:
110   !
120   INTEGER Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
130   DATA 0,1,2,3,4,5,6,7
140   READ Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
150   !
160   INTEGER Sts                   ! Gets TOGGLEBUTTON value
170   DIM V1$[1],V2$[1],V3$[1]      ! Converts value to string
180   !
190   INTEGER Nlines,D(1:4)         ! Used for display handling
200   INTEGER Dw,Dh                 ! Gives display size
210   INTEGER X,Y,Pw,Ph,Iw,Ih               ! PANEL parameters
220   INTEGER R1,R2,C1,C2,C3                ! Coordinates for widgets
230   INTEGER Bh,Bw,Lw              ! Widget sizes
240   !
250   STATUS CRT,13;Nlines
260   GESCAPE CRT,3;D(*)
270   Dw=D(3)-D(1)
280   Dh=(D(4)-D(2))*((Nlines-7)/Nlines)
290   !
300   Ph=170                        ! Interior height
310   Pw=280                        ! Interior width
320   X=(Dw-Pw)/2                   ! Center panel on screen
330   Y=(Dh-Ph)/2
340   !
350   CLEAR SCREEN
360   ASSIGN @Panel TO WIDGET "PANEL";SET ("VISIBLE":0)
370   CONTROL @Panel;SET ("MAXIMIZABLE":0,"RESIZABLE":0)
380   CONTROL @Panel;SET ("X":X,"Y":Y,"WIDTH":Pw,"HEIGHT":1.2*Ph)
390   CONTROL @Panel;SET ("TITLE":" Example: TOGGLEBUTTON Widget")
400   CONTROL @Panel;SET ("SYSTEM MENU":"Quit")
410   !
```

```
420   ! Determine inside dimensions of PANEL
430   !
440   STATUS @Panel;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
450   !
460   Gaph=Ih*.04
470   Gapw=Iw*.03
480   Bh=(Ih-3*Gaph)/2                    ! Button height
490   Bw=(Iw-4*Gapw)/3                    ! Button width
500   R1=Gaph                             ! Top row
510   R2=R1+Bh+Gaph                       ! Row 2 = row 1 + button height + gap
520   C1=Gapw                             ! Left column
530   C2=C1+Bw+Gapw                       ! Col 2 = col 1 + button width + gap
540   C3=C2+Bw+Gapw                       ! Col 3 = col 2 + button width + gap
550   Lw=Bw*3+Gapw*2                      ! Label width = 3 buttons plus gaps
560   !
570   ! Set up TOGGLEBUTTONS
580   !
590   ASSIGN @T1 TO WIDGET "TOGGLEBUTTON";PARENT @Panel
600   CONTROL @T1;SET ("X":C1,"Y":R1,"WIDTH":Bw,"HEIGHT":Bh)
610   CONTROL @T1;SET ("PEN":Blue)
620   CONTROL @T1;SET ("LABEL":"T1")
630   !
640   ASSIGN @T2 TO WIDGET "TOGGLEBUTTON";PARENT @Panel
650   CONTROL @T2;SET ("X":C2,"Y":R1,"WIDTH":Bw,"HEIGHT":Bh)
660   CONTROL @T2;SET ("PEN":Blue)
670   CONTROL @T2;SET ("LABEL":"T2")
680   !
690   ASSIGN @T3 TO WIDGET "TOGGLEBUTTON";PARENT @Panel
700   CONTROL @T3;SET ("X":C3,"Y":R1,"WIDTH":Bw,"HEIGHT":Bh)
710   CONTROL @T3;SET ("PEN":Blue)
720   CONTROL @T3;SET ("LABEL":"T3")
730   !
740   ! Create LABEL widget
750   !
760   ASSIGN @Label TO WIDGET "LABEL";PARENT @Panel
770   CONTROL @Label;SET ("X":C1,"Y":R2,"WIDTH":Lw,"HEIGHT":Bh)
780   CONTROL @Label;SET ("PEN":Red)
790   !
800   ! Set up events
810   !
820   ON EVENT @T1,"CHANGED" GOSUB Handler
830   ON EVENT @T2,"CHANGED" GOSUB Handler
```

```
840    ON EVENT @T3,"CHANGED" GOSUB Handler
850    ON EVENT @Panel,"SYSTEM MENU" GOTO Finis
860    !
870    ! Turn on PANEL
880    !
890    GOSUB Handler
900    CONTROL @Panel;SET ("VISIBLE":1)
910    !
920    ! Main loop, wait for event
930    !
940    LOOP
950       WAIT FOR EVENT
960    END LOOP
970    STOP
980    !
990    ! Handlers for buttons
1000   !
1010 Handler: !
1020   STATUS @T1;RETURN ("VALUE":Sts)
1030   V1$=VAL$(Sts)
1040   STATUS @T2;RETURN ("VALUE":Sts)
1050   V2$=VAL$(Sts)
1060   STATUS @T3;RETURN ("VALUE":Sts)
1070   V3$=VAL$(Sts)
1080   CONTROL @Label;SET ("VALUE":"T1:"&V1$&" / T2:"&V2$&" / T3:"&V3$)
1090   RETURN
1100   !
1110 Finis: !
1120   ASSIGN @Panel TO *                ! Delete PANEL widget
1130   END
```

```
10      ! ***************************************************
20      ! Example: Tab Groups
30      !
40      ! This program creates four tab groups of widgets:
50      !
60      !     Group 1: R1_C1, R1_C2, R1_C3
70      !     Group 2: R2_C1, R2_C2, R2_C3
80      !     Group 3: R3_C1, R3_C2, R3_C3
90      !     Group 4: R4_C1, R4_C2, R4_C3
100     !
110     ! You can use the Tab key to traverse the four
120     ! tab groups (from R1_C1 to R2_C1 to R3_C1 to R4_C1).
130     ! You must use the Arrow keys to traverse within
140     ! a tab group (e.g., from R1_C1 to R1_C2 to R1_C3, etc.)
150     !
160     ! ***************************************************
170     !
180     DIM L$[5]
190     INTEGER D(1:4)
200     INTEGER Ih,Iw,Pw,Ph,X,Y,Xn,Yn,W,H,Butw,Buth,R,C1,C2,C3
210     !
220     GESCAPE CRT,3;D(*)
230     Pw=(D(3)-D(1))/2
240     Ph=(D(4)-D(2))/2
250     X=Pw/2
260     Y=Ph/2
270     !
280     CLEAR SCREEN
290     !
300     ASSIGN @Tabgp TO WIDGET "PANEL";SET ("VISIBLE":0)
310     CONTROL @Tabgp;SET ("TITLE":" Example: Tab Groups")
320     CONTROL @Tabgp;SET ("X":X,"Y":Y,"WIDTH":Pw,"HEIGHT":Ph)
330     CONTROL @Tabgp;SET ("MAXIMIZABLE":0,"RESIZABLE":0,"MOVABLE":0)
340     CONTROL @Tabgp;SET ("SYSTEM MENU":"Quit")
350     CONTROL @Tabgp;SET ("VISIBLE":1)
360     !
370     STATUS @Tabgp;RETURN ("INSIDE WIDTH":Iw,"INSIDE HEIGHT":Ih)
380     !
390     Gapw=8                  ! column gap size
400     Gaph=8                  ! row gap size
410     Butw=(Iw-4*Gapw)/3      ! button width
420     Buth=(Ih-5*Gaph)/4      ! button height
```

```
430   C1=Gaph
440   C2=C1+Butw+Gapw
450   C3=C2+Butw+Gapw
460   !
470   R=Gaph
480   L$="R1_C1"
490   CALL Makebutton(@Tabgp,@B1,L$,C1,R,Butw,Buth,1)
500   !
510   L$="R1_C2"
520   CALL Makebutton(@Tabgp,@B2,L$,C2,R,Butw,Buth,0)
530   !
540   L$="R1_C3"
550   CALL Makebutton(@Tabgp,@B3,L$,C3,R,Butw,Buth,0)
560   !
570   R=R+Buth+Gaph
580   L$="R2_C1"
590   CALL Makebutton(@Tabgp,@B4,L$,C1,R,Butw,Buth,1)
600   !
610   L$="R2_C2"
620   CALL Makebutton(@Tabgp,@B5,L$,C2,R,Butw,Buth,0)
630   !
640   L$="R2_C3"
650   CALL Makebutton(@Tabgp,@B6,L$,C3,R,Butw,Buth,0)
660   !
670   R=R+Buth+Gaph
680   L$="R3_C1"
690   CALL Makebutton(@Tabgp,@B7,L$,C1,R,Butw,Buth,1)
700   !
710   L$="R3_C2"
720   CALL Makebutton(@Tabgp,@B8,L$,C2,R,Butw,Buth,0)
730   !
740   L$="R3_C3"
750   CALL Makebutton(@Tabgp,@B9,L$,C3,R,Butw,Buth,0)
760   !
770   R=R+Buth+Gaph
780   L$="R4_C1"
790   CALL Makebutton(@Tabgp,@B10,L$,C1,R,Butw,Buth,1)
800   !
810   L$="R4_C2"
820   CALL Makebutton(@Tabgp,@B11,L$,C2,R,Butw,Buth,0)
830   !
840   L$="R4_C3"
```

```
850    CALL Makebutton(@Tabgp,@B12,L$,C3,R,Butw,Buth,0)
860    !
870    ON EVENT @Tabgp,"SYSTEM MENU" GOTO Finis
880    !
890    LOOP
900      WAIT FOR EVENT
910    END LOOP
920    !
930 Finis:    !
940    ASSIGN @Tabgp TO *              ! Delete PANEL widget
950    END
960    !
970    SUB Makebutton(@Tabgp,@B,L$,INTEGER X,Y,W,H,Ts)
980      ASSIGN @B TO WIDGET "PUSHBUTTON";PARENT @Tabgp,SET ("VISIBLE":0)
990      CONTROL @B;SET ("X":X,"Y":Y,"WIDTH":W,"HEIGHT":H)
1000     CONTROL @B;SET ("LABEL":L$)
1010     CONTROL @B;SET ("TAB STOP":Ts,"VISIBLE":1)
1020   SUBEND
```

```
10      ! ************************************************************
20      ! Example: Tic-Tac-Toe
30      !
40      ! This example program plays tic-tac-toe.
50      ! It illustrates the use of STRING widgets with different fonts, and
60      ! shows how attribute arrays can be used. The PUSHBUTTON object
70      ! is used to initiate an action. Each of the squares is a pushbutton.
80      ! When a button is clicked it is given to a player and deactivated.
90      !
100   ! The arrays Attr_n$ and Attr are used to program six numeric attributes
110   ! at once. The arrays Attr_s$ and Attr_sv$ are used to program three
120   ! string attributes.
130   !
140   ! ************************************************************
150   !
160   DIM Attr_n$(5)[11],Attr_s$(2)[10],Attr(5),Attr_sv$(2)[10],Xo$(1)[1]
170   DIM Sel$(4)[1],Grid$(1)[10],Buttons$(2)[5]
180   DIM Xo(3,3)
190   Dims=4
200   Attr_n$(0)="X"
210   Attr_n$(1)="Y"
220   Attr_n$(2)="WIDTH"
230   Attr_n$(3)="HEIGHT"
240   Attr_n$(4)="RESIZABLE"
250   Attr_n$(5)="MAXIMIZABLE"
260   Attr(0)=100                   ! x position
270   Attr(1)=20                          ! y position
280   Attr(2)=50                          ! width
290   Attr(3)=50                          ! height
300   Attr(4)=0                     ! not resizable
310   Attr(5)=0                     ! not maximizable
320   Attr_s$(0)="FONT"
330   Attr_s$(1)="TITLE"
340   Attr_s$(2)="LABEL"
350   Attr_sv$(0)="20 BY 30"        ! A large font
360   Attr_sv$(1)=""                ! Makes the title area disappear
370   Attr_sv$(2)=""                ! Use a blank label to start
380   Sel$(0)="0"
390   Sel$(1)="1"
400   Sel$(2)="2"
410   Sel$(3)="3"
```

```
420     Sel$(4)="4"
430     Grid$(0)="3 BY 3"
440     Grid$(1)="4 BY 4"
450     Buttons$(0)="1"
460     Buttons$(1)="2"
470     Buttons$(2)="Quit"
480     DIALOG "QUESTION","How many players?",Button;SET ("DIALOG BUTTONS":Buttons$
(*))
490     IF Button=2 THEN STOP
500     Players=Button+1
510     DIALOG "LIST","Select Playing Grid",Button;SET ("ITEMS":Grid$(*),"SELECTED
ITEM":0),RETURN ("SELECTED ITEM":Grid)
520     Dims=3
530     IF Grid=1 THEN Dims=4
540     IF Button=1 THEN STOP
550     DIALOG "LIST","Select Level of Difficulty",Button;SET ("ITEMS":Sel$(*),"SELECTED
ITEM":1),RETURN ("SELECTED ITEM":Level)
560     IF Button=1 THEN STOP
570     !
580     ! Nine pushbutton widgets are created in three rows and three columns.
590     ! The use of the attribute arrays allows all the widgets to be changed
600     ! easily and yet have them still be aligned.
610     !
620     ASSIGN @Box11 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*))
630     Attr(0)=Attr(0)+Attr(2)                 ! x position moved one button width right
640     !
650     ASSIGN @Box12 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*))
660     Attr(0)=Attr(0)+Attr(2)                 ! x position moved one button width right
670     !
680     ASSIGN @Box13 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*))
690     Attr(0)=Attr(0)+Attr(2)
700     !
710     ASSIGN @Box14 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*),"VISIBLE":0)
720     Attr(0)=Attr(0)-3*Attr(2)               ! x position moved two button widths left
730     Attr(1)=Attr(1)+Attr(3)                 ! y position moved one button height down
740     !
750     ASSIGN @Box21 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*))
760     Attr(0)=Attr(0)+Attr(2)
770     !
```

```
780     ASSIGN @Box22 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*))
790     Attr(0)=Attr(0)+Attr(2)
800     !
810     ASSIGN @Box23 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*))
820     Attr(0)=Attr(0)+Attr(2)
830     !
840     ASSIGN @Box24 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*),"VISIBLE":0)
850     Attr(0)=Attr(0)-3*Attr(2)
860     Attr(1)=Attr(1)+Attr(3)
870     !
880     ASSIGN @Box31 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*))
890     Attr(0)=Attr(0)+Attr(2)
900     !
910     ASSIGN @Box32 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*))
920     Attr(0)=Attr(0)+Attr(2)
930     !
940     ASSIGN @Box33 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*))
950     Attr(0)=Attr(0)+Attr(2)
960     !
970     ASSIGN @Box34 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*),"VISIBLE":0)
980     Attr(0)=Attr(0)-3*Attr(2)
990     Attr(1)=Attr(1)+Attr(3)
1000    !
1010    ASSIGN @Box41 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*),"VISIBLE":0)
1020    Attr(0)=Attr(0)+Attr(2)
1030    !
1040    ASSIGN @Box42 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*),"VISIBLE":0)
1050    Attr(0)=Attr(0)+Attr(2)
1060    !
1070    ASSIGN @Box43 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*),"VISIBLE":0)
1080    Attr(0)=Attr(0)+Attr(2)
1090    !
1100    ASSIGN @Box44 TO WIDGET "PUSHBUTTON";SET (Attr_n$(*):Attr(*),Attr_s$(*):Attr_sv$
(*),"VISIBLE":0)
1110    IF Dims=4 THEN
```

```
1120     CONTROL @Box44;SET ("VISIBLE":1)
1130     CONTROL @Box43;SET ("VISIBLE":1)
1140     CONTROL @Box42;SET ("VISIBLE":1)
1150     CONTROL @Box41;SET ("VISIBLE":1)
1160     CONTROL @Box34;SET ("VISIBLE":1)
1170     CONTROL @Box24;SET ("VISIBLE":1)
1180     CONTROL @Box14;SET ("VISIBLE":1)
1190   END IF
1200   ASSIGN @Quit TO WIDGET "PUSHBUTTON"
1210   CONTROL @Quit;SET ("TITLE":"","LABEL":"Quit","X":10,"Y":22)
1220   CONTROL @Quit;SET ("HEIGHT":40,"WIDTH":60)
1230   ON EVENT @Quit,"ACTIVATED" GOTO Finis
1240   !
1250   ! The Turn variable keeps track of whose
1260   ! turn it is. Zero means X, one means O.
1270   !
1280   Turn=0
1290   Xo$(0)="X"
1300   Xo$(1)="O"
1310   !
1320   ! A STRING widget to remind the players whose turn it is
1330   !
1340   Attr(0)=Attr(0)-3*Attr(2)                    ! Move x position back to left edge
1350   IF Dims=4 THEN Attr(1)=Attr(1)+Attr(3)       ! Move y position to bottom
1360   Attr(2)=Dims*Attr(2)                         ! Set width to three buttons wide
1370   !
1380   ASSIGN @Who TO WIDGET "STRING";SET (Attr_n$(*):Attr(*),"TITLE":"","VALUE":"It is
"&Xo$(Turn)&"'s turn")
1390   !
1400   ! This array records who has which square. It is initialized to values
1410   ! which are unequal and do not correspond to either an X or an O.
1420   !
1430 Start: !
1440   FOR I=0 TO Dims-1
1450     FOR J=0 TO Dims-1
1460       Xo(I,J)=-1-I-2*J
1470     NEXT J
1480   NEXT I
1490   !
1500   ! Each pushbotton calls its own subroutine.
1510   !
1520   ON EVENT @Box11,"ACTIVATED" GOSUB Hit_box11
```

```
1530   ON EVENT @Box12,"ACTIVATED" GOSUB Hit_box12
1540   ON EVENT @Box13,"ACTIVATED" GOSUB Hit_box13
1550   ON EVENT @Box14,"ACTIVATED" GOSUB Hit_box14
1560   ON EVENT @Box21,"ACTIVATED" GOSUB Hit_box21
1570   ON EVENT @Box22,"ACTIVATED" GOSUB Hit_box22
1580   ON EVENT @Box23,"ACTIVATED" GOSUB Hit_box23
1590   ON EVENT @Box24,"ACTIVATED" GOSUB Hit_box24
1600   ON EVENT @Box31,"ACTIVATED" GOSUB Hit_box31
1610   ON EVENT @Box32,"ACTIVATED" GOSUB Hit_box32
1620   ON EVENT @Box33,"ACTIVATED" GOSUB Hit_box33
1630   ON EVENT @Box34,"ACTIVATED" GOSUB Hit_box34
1640   ON EVENT @Box41,"ACTIVATED" GOSUB Hit_box41
1650   ON EVENT @Box42,"ACTIVATED" GOSUB Hit_box42
1660   ON EVENT @Box43,"ACTIVATED" GOSUB Hit_box43
1670   ON EVENT @Box44,"ACTIVATED" GOSUB Hit_box44
1680   LOOP
1690     CONTROL @Who;SET ("VALUE":"It is "&Xo$(Turn)&"'s turn")
1700     IF Turn=0 OR Players=2 THEN
1710       WAIT FOR EVENT
1720     ELSE
1730       Computer_play(Xo(*),Turn,Level,Row,Col,Dims)
1740       Set_box(Xo$
(Turn),Row,Col,@Box11,@Box12,@Box13,@Box14,@Box21,@Box22,@Box23,@Box24,@Box3
1,@Box32,@Box33,@Box34,@Box41,@Box42,@Box43,@Box44)
1750       Xo(Row,Col)=Turn
1760       Turn=(Turn=0)
1770     END IF
1780   !
1790   ! These IF statements check if anyone has three in a row
1800   !
1810     Winner=-1
1820     Winner=FNCheck_winner(Xo(*),Dims)
1830     IF Winner>=0 THEN GOTO Win
1840   !
1850   ! Check to see if all the squares have been taken
1860   !
1870     Tie=FNCheck_tie(Xo(*),Dims)
1880     IF Tie THEN GOTO Tie
1890   END LOOP
1900   !
1910   ! Each square has its own subroutine. They are nearly identical.
1920   ! The appropriate square is given to the player whose turn it is.
1930   ! An X or O is in placed in the square. The pushbutton for that
```

```
1940   ! square is deactivated. The turn indicator is flipped.
1950   !
1960 Hit_box11: !
1970   Xo(0,0)=Turn
1980   CONTROL @Box11;SET ("LABEL":Xo$(Turn))
1990   Turn=(Turn=0)
2000   OFF EVENT @Box11,"ACTIVATED"
2010   RETURN
2020   !
2030 Hit_box12: !
2040   Xo(0,1)=Turn
2050   CONTROL @Box12;SET ("LABEL":Xo$(Turn))
2060   Turn=(Turn=0)
2070   OFF EVENT @Box12,"ACTIVATED"
2080   RETURN
2090   !
2100 Hit_box13: !
2110   Xo(0,2)=Turn
2120   CONTROL @Box13;SET ("LABEL":Xo$(Turn))
2130   Turn=(Turn=0)
2140   OFF EVENT @Box13,"ACTIVATED"
2150   RETURN
2160   !
2170 Hit_box14: !
2180   Xo(0,3)=Turn
2190   CONTROL @Box14;SET ("LABEL":Xo$(Turn))
2200   Turn=(Turn=0)
2210   OFF EVENT @Box14,"ACTIVATED"
2220   RETURN
2230   !
2240 Hit_box21: !
2250   CONTROL @Box21;SET ("LABEL":Xo$(Turn))
2260   Xo(1,0)=Turn
2270   Turn=(Turn=0)
2280   OFF EVENT @Box21,"ACTIVATED"
2290   RETURN
2300   !
2310 Hit_box22: !
2320   Xo(1,1)=Turn
2330   CONTROL @Box22;SET ("LABEL":Xo$(Turn))
2340   Turn=(Turn=0)
2350   OFF EVENT @Box22,"ACTIVATED"
```

```
2360   RETURN
2370   !
2380 Hit_box23: !
2390   Xo(1,2)=Turn
2400   CONTROL @Box23;SET ("LABEL":Xo$(Turn))
2410   Turn=(Turn=0)
2420   OFF EVENT @Box23,"ACTIVATED"
2430   RETURN
2440   !
2450 Hit_box24: !
2460   Xo(1,3)=Turn
2470   CONTROL @Box24;SET ("LABEL":Xo$(Turn))
2480   Turn=(Turn=0)
2490   OFF EVENT @Box24,"ACTIVATED"
2500   RETURN
2510   !
2520 Hit_box31: !
2530   Xo(2,0)=Turn
2540   CONTROL @Box31;SET ("LABEL":Xo$(Turn))
2550   Turn=(Turn=0)
2560   OFF EVENT @Box31,"ACTIVATED"
2570   RETURN
2580   !
2590 Hit_box32: !
2600   Xo(2,1)=Turn
2610   CONTROL @Box32;SET ("LABEL":Xo$(Turn))
2620   Turn=(Turn=0)
2630   OFF EVENT @Box32,"ACTIVATED"
2640   RETURN
2650   !
2660 Hit_box33: !
2670   Xo(2,2)=Turn
2680   CONTROL @Box33;SET ("LABEL":Xo$(Turn))
2690   Turn=(Turn=0)
2700   OFF EVENT @Box33,"ACTIVATED"
2710   RETURN
2720   !
2730 Hit_box34: !
2740   Xo(2,3)=Turn
2750   CONTROL @Box34;SET ("LABEL":Xo$(Turn))
2760   Turn=(Turn=0)
2770   OFF EVENT @Box34,"ACTIVATED"
```

```
2780   RETURN
2790   !
2800 Hit_box41: !
2810   Xo(3,0)=Turn
2820   CONTROL @Box41;SET ("LABEL":Xo$(Turn))
2830   Turn=(Turn=0)
2840   OFF EVENT @Box41,"ACTIVATED"
2850   RETURN
2860   !
2870 Hit_box42: !
2880   Xo(3,1)=Turn
2890   CONTROL @Box42;SET ("LABEL":Xo$(Turn))
2900   Turn=(Turn=0)
2910   OFF EVENT @Box42,"ACTIVATED"
2920   RETURN
2930   !
2940 Hit_box43: !
2950   Xo(3,2)=Turn
2960   CONTROL @Box43;SET ("LABEL":Xo$(Turn))
2970   Turn=(Turn=0)
2980   OFF EVENT @Box43,"ACTIVATED"
2990   RETURN
3000   !
3010 Hit_box44: !
3020   Xo(3,3)=Turn
3030   CONTROL @Box44;SET ("LABEL":Xo$(Turn))
3040   Turn=(Turn=0)
3050   OFF EVENT @Box44,"ACTIVATED"
3060   RETURN
3070 Win: !
3080   !
3090   ! Announce the winner
3100   !
3110   DIALOG "INFORMATION",Xo$(Winner)&" wins";SET
("X":50,"Y":0,"WIDTH":300,"HEIGHT":300,"FONT":"20 BY 30")
3120   GOTO Cleanup
3130 Tie:!
3140   !
3150   ! Declare a tie
3160   !
3170   DIALOG "INFORMATION","Tie";SET
("X":50,"Y":0,"WIDTH":300,"HEIGHT":300,"FONT":"20 BY 30")
3180   !
```

```
3190   ! Make the labels in all the squares blank
3200   !
3210 Cleanup: !
3220   CONTROL @Box11;SET ("LABEL":"")
3230   CONTROL @Box12;SET ("LABEL":"")
3240   CONTROL @Box13;SET ("LABEL":"")
3250   CONTROL @Box14;SET ("LABEL":"")
3260   CONTROL @Box21;SET ("LABEL":"")
3270   CONTROL @Box22;SET ("LABEL":"")
3280   CONTROL @Box23;SET ("LABEL":"")
3290   CONTROL @Box24;SET ("LABEL":"")
3300   CONTROL @Box31;SET ("LABEL":"")
3310   CONTROL @Box32;SET ("LABEL":"")
3320   CONTROL @Box33;SET ("LABEL":"")
3330   CONTROL @Box34;SET ("LABEL":"")
3340   CONTROL @Box41;SET ("LABEL":"")
3350   CONTROL @Box42;SET ("LABEL":"")
3360   CONTROL @Box43;SET ("LABEL":"")
3370   CONTROL @Box44;SET ("LABEL":"")
3380   !
3390   ! Give the turn indicator to X and start a new game
3400   !
3410   Turn=0
3420   GOTO Start
3430 Finis: END
3440   DEF FNCheck_winner(Xo(*),Dims)
3450     Winner=-1
3460     Fail=0
3470   !
3480   ! Check Rows
3490   !
3500     FOR I=0 TO Dims-1
3510       FOR J=0 TO Dims-2
3520         IF Xo(I,J+1)<>Xo(I,J) THEN
3530           Fail=1
3540           GOTO 3590
3550         END IF
3560         Tmp=Xo(I,J)
3570       NEXT J
3580       IF Fail=0 THEN RETURN Tmp
3590       Fail=0
3600     NEXT I
```

```
3610  !
3620  ! Check Columns
3630  !
3640    Fail=0
3650    FOR J=0 TO Dims-1
3660      FOR I=0 TO Dims-2
3670        IF Xo(I+1,J)<>Xo(I,J) THEN
3680          Fail=1
3690          GOTO 3740
3700        END IF
3710        Tmp=Xo(I,J)
3720      NEXT I
3730      IF Fail=0 THEN RETURN Tmp
3740      Fail=0
3750    NEXT J
3760  !
3770  ! Check Diagonals
3780  !
3790    Tmp=Xo(0,0)
3800    Fail=0
3810    FOR J=0 TO Dims-2
3820      IF Xo(J,J)<>Xo(J+1,J+1) THEN
3830        Fail=1
3840        GOTO 3880
3850      END IF
3860    NEXT J
3870    IF Fail=0 THEN RETURN Tmp
3880    Tmp=Xo(Dims-1,0)
3890    Fail=0
3900    FOR J=0 TO Dims-2
3910      IF Xo(Dims-1-J,J)<>Xo(Dims-2-J,J+1) THEN
3920        Fail=1
3930        RETURN -1
3940      END IF
3950    NEXT J
3960    IF Fail=0 THEN RETURN Tmp
3970    RETURN -1
3980  FNEND
3990  DEF FNCheck_tie(Xo(*),Dims)
4000    Tie=1
4010    FOR I=0 TO Dims-1
4020      FOR J=0 TO Dims-1
```

```
4030          IF Xo(I,J)<0 THEN Tie=0
4040        NEXT J
4050      NEXT I
4060      RETURN Tie
4070    FNEND
4080    SUB Computer_play(Xo(*),Turn,Level,Row,Col,Dims)
4090 Computer_play: !
4100      DIM Weights(3,3)
4110      REDIM Weights(Dims-1,Dims-1)
4120      Opponent=(Turn+1) MOD 2
4130      MAT Weights=(0)
4140      IF Level>1 THEN ASSIGN @W TO WIDGET "LABEL";SET ("VALUE":"I'm calculating.
Please Wait...","COLUMNS":35,"ROWS":2)
4150      Find_weights(Xo(*),Weights(*),Turn,Turn,Level,Dims)
4160      Eval_weights(Xo(*),Weights(*),Row,Col,Dims)
4170      IF Level>1 THEN ASSIGN @W TO *
4180    SUBEND
4190    SUB Find_weights(X(*),Weights(*),Mark,Turn,Level,Dims)
4200 Find_weights: !
4210      DIM X1(3,3)
4220      REDIM X1(Dims-1,Dims-1)
4230      MAT X1=X
4240      REAL Temp
4250      Opponent=(Turn+1) MOD 2
4260      Opp_mark=(Mark+1) MOD 2
4270      FOR I=0 TO Dims-1
4280        FOR J=0 TO Dims-1
4290          IF X1(I,J)<0 THEN
4300            Temp=X1(I,J)
4310            X1(I,J)=Turn
4320            M=FNCheck_winner(X1(*),Dims)
4330            IF M=Mark THEN
4340              Weights(I,J)=Weights(I,J)+1
4350            ELSE
4360              IF M=Opp_mark THEN
4370                Weights(I,J)=Weights(I,J)+1
4380              ELSE
4390                IF Level>0 THEN CALL
Find_weights(X1(*),Weights(*),Mark,Opponent,Level-1,Dims)
4400              END IF
4410            END IF
4420            X1(I,J)=Temp
4430          END IF
```

```
4440        NEXT J
4450      NEXT I
4460    SUBEND
4470    SUB Eval_weights(X(*),W(*),R,C,Dims)
4480      DIM Opt_rows(16),Opt_cols(16)
4490      Best=MAX(W(*))
4500      Count=0
4510      FOR I=0 TO Dims-1
4520        FOR J=0 TO Dims-1
4530          IF W(I,J)=Best AND X(I,J)<0 THEN
4540            Opt_rows(Count)=I
4550            Opt_cols(Count)=J
4560            Count=Count+1
4570          END IF
4580        NEXT J
4590      NEXT I
4600      Index=MAX(0,PROUND(Count*RND-.5,0))
4610      R=Opt_rows(Index)
4620      C=Opt_cols(Index)
4630    SUBEND
4640    SUB
Set_box(X$,Row,Col,@B11,@B12,@B13,@B14,@B21,@B22,@B23,@B24,@B31,@B32,@B33
,@B34,@B41,@B42,@B43,@B44)
4650      SELECT 10*Row+Col
4660      CASE 0
4670        CONTROL @B11;SET ("LABEL":X$)
4680      CASE 1
4690        CONTROL @B12;SET ("LABEL":X$)
4700      CASE 2
4710        CONTROL @B13;SET ("LABEL":X$)
4720      CASE 3
4730        CONTROL @B14;SET ("LABEL":X$)
4740      CASE 10
4750        CONTROL @B21;SET ("LABEL":X$)
4760      CASE 11
4770        CONTROL @B22;SET ("LABEL":X$)
4780      CASE 12
4790        CONTROL @B23;SET ("LABEL":X$)
4800      CASE 13
4810        CONTROL @B24;SET ("LABEL":X$)
4820      CASE 20
4830        CONTROL @B31;SET ("LABEL":X$)
4840      CASE 21
```

```
4850        CONTROL @B32;SET ("LABEL":X$)
4860      CASE 22
4870        CONTROL @B33;SET ("LABEL":X$)
4880      CASE 23
4890        CONTROL @B34;SET ("LABEL":X$)
4900      CASE 30
4910        CONTROL @B41;SET ("LABEL":X$)
4920      CASE 31
4930        CONTROL @B42;SET ("LABEL":X$)
4940      CASE 32
4950        CONTROL @B43;SET ("LABEL":X$)
4960      CASE 33
4970        CONTROL @B44;SET ("LABEL":X$)
4980      END SELECT
4990    SUBEND
```

```
10      ! ***********************************************
20      ! Example: Using ON EVENT
30      !
40      ! This program shows one way ON EVENT can be
50      ! used for error handling.
60      !
70      ! ***********************************************
80      !
90       ON SIGNAL 15 CALL Sig15_handler
100    ON ERROR CALL Error_handler
110    A=0/0
120    DIALOG "INFORMATION","After divide by zero error"
130    END
140    !
150    SUB Sig15_handler
160       ASSIGN @Printer TO WIDGET "PRINTER"
170       CONTROL @Printer;SET ("X":0,"Y":0,"TITLE":"Event Record")
180       CONTROL @Printer;SET ("TEXT":"ERROR: "&ERRM$)
190       CONTROL @Printer;SET ("APPEND TEXT":"ON SIGNAL 15 System Priority:
"&SYSTEM$("SYSTEM PRIORITY"))
200       STATUS @Printer;RETURN ("WIDTH":X,"HEIGHT":Y)
210       !
220       ASSIGN @Button TO WIDGET "PUSHBUTTON";SET
("VISIBLE":0,"RESIZABLE":0,"TITLE":"","LABEL":"PUSH ME")
230       STATUS @Button;RETURN ("WIDTH":W,"HEIGHT":H)
240       CONTROL @Button;SET ("X":(X-W)/2,"Y":(Y-H)/2,"VISIBLE":1)
250       ON EVENT @Button,"ACTIVATED",15 GOTO Okd
260       LOOP
270          WAIT FOR EVENT
280          CONTROL @Printer;SET ("APPEND TEXT":"WAIT FOR EVENT terminated and
branch taken (this statement never executed).")
290       END LOOP
300 Okd: CONTROL @Printer;SET ("APPEND TEXT":"Branch taken.")
310       WAIT 1
320    SUBEND
330    SUB Error_handler
340       DIALOG "INFORMATION","Now in error handler. System Priority: "&SYSTEM$
("SYSTEM PRIORITY")&CHR$(10)&"asserting signal 15"
350       SIGNAL 15
360       ERROR SUBEXIT
370    SUBEND
```

```
10      ! ****************************************
20      ! Example: WARNING Dialog
30      !
40      ! This program generates a WARNING dialog.
50      !
60      ! ****************************************
70      !
80      DIM P$[30]
90      P$="CORE MELTDOWN IN ONE MINUTE!!"
100   DIALOG "WARNING",P$;SET ("TITLE":" Example: WARNING Dialog")
110   END
```

```
10      ! ***********************************************
20      ! Example: Wing Stress/Vibration Analysis
30      !
40      ! This program displays a simulated aircraft
50      ! fuselage. When the user clicks one of the
60      ! four arrows displayed, the appropriate wing
70      ! stress/vibration analysis graph is displayed.
80      !
90      !                    NOTE
100     !
110     ! You may need to maximize the screen to see
120     ! the full picture.
130     !
140     ! ***********************************************
150     !
160     INTEGER Num_points
170     ASSIGN @Panel TO WIDGET "PANEL"
180     CONTROL @Panel;SET ("WIDTH":650,"HEIGHT":600)
190     CONTROL @Panel;SET ("TITLE":" Example: Wing Stress/Vibration Analysis","SIZE
CONTROL":"RESIZE CHILDREN")
200     CONTROL @Panel;SET ("SIZE CONTROL":"RESIZE CHILDREN")
210     CONTROL @Panel;SET ("SYSTEM MENU":"Quit")
220     ON EVENT @Panel,"SYSTEM MENU" GOTO Finis
230     !
240     ASSIGN @Label TO WIDGET "LABEL";SET ("X":100,"Y":5,"WIDTH":400),PARENT
@Panel
250     CONTROL @Label;SET ("VALUE":"Click arrows to display results")
260     !
270     ASSIGN @Bitmap TO WIDGET "BITMAP";SET ("X":15,"Y":40,"BORDER":0,"RETAIN
RASTER":1,"BITMAP FILE":"WINGS.BMP","LABEL":"XRJ-711 Wind Tunnel Test"),PARENT
@Panel
280     !
290     ON EVENT @Bitmap,"MOUSE CLICKED" GOSUB Mouse_click
300 Looping: GOTO Looping
310     !
320 Mouse_click:                        !
330     INTEGER Mouse_pos(1:2),X,Y
340     STATUS @Bitmap;RETURN ("MOUSE CLICK":Mouse_pos(*))
350     X=Mouse_pos(1)
360     Y=Mouse_pos(2)
370     IF X>=194 AND X<=213 AND Y>=230 AND Y<=260 THEN
```

```
380       GOSUB Sbup_disp
390     ELSE
400       IF X>=194 AND X<=213 AND Y>=292 AND Y<=322 THEN
410         GOSUB Sbdn_disp
420       ELSE
430         IF X>=450 AND X<=470 AND Y>=230 AND Y<=260 THEN
440           GOSUB Prup_disp
450         ELSE
460           IF X>=450 AND X<=470 AND Y>=292 AND Y<=322 THEN
470             GOSUB Prdn_disp
480           ELSE
490             DISP
500           END IF
510         END IF
520       END IF
530     END IF
540     RETURN
550     !
560     ! Data Section
570     !
580 Sbup_num_points: DATA 9
590 Sbup_x_data: DATA 0, 50, 100, 150, 200, 250, 300, 350, 400
600 Sbup_y_data: DATA 0, 30, 400,   20, 300,     5,     7,     4,     1
610     !
620 Sbdn_num_points: DATA 9
630 Sbdn_x_data: DATA 0, 50, 100, 150, 200, 250, 300, 350, 400
640 Sbdn_y_data: DATA 0, 60, 300,   40, 250,     5,     7,     4,     1
650     !
660 Prup_num_points: DATA 9
670 Prup_x_data: DATA 0, 50, 100, 150, 200, 250, 300, 350, 400
680 Prup_y_data: DATA 0, 35, 450,   30, 200,     5,     7,     4,     1
690     !
700 Prdn_num_points: DATA 9
710 Prdn_x_data: DATA 0, 50, 100, 150, 200, 250, 300, 350, 400
720 Prdn_y_data: DATA 0, 40, 350,   35, 230,     5,     7,     4,     1
730     !
740     ! Display Section
750     !
760 Sbup_disp:!
770     ASSIGN @Sbup_panel TO WIDGET "PANEL";SET
("X":4,"Y":4,"WIDTH":485,"HEIGHT":345,"SIZE CONTROL":"RESIZE CHILDREN")
780     CONTROL @Sbup_panel;SET ("TITLE":"Starboard Wing: Upper Skin Strain")
```

```
790     ASSIGN @Sbup_xy TO WIDGET "XY GRAPH";SET
("X":10,"Y":10,"WIDTH":454,"HEIGHT":265,"VISIBLE":0),PARENT @Sbup_panel
800     CONTROL @Sbup_xy;SET ("CURRENT AXIS":"X","AUTOSCALE":1,"AXIS
LABEL":"Vibration Frequency (Hz)")
810     CONTROL @Sbup_xy;SET ("CURRENT AXIS":"Y","AUTOSCALE":1,"AXIS LABEL":"Micro
Strain")
820     RESTORE Sbup_num_points
830     READ Num_points
840     ON ERROR GOTO Sbup_alloc
850     DEALLOCATE Sbup_x(*),Sbup_y(*)
860     !
870 Sbup_alloc: OFF ERROR
880     ALLOCATE INTEGER Sbup_x(1:Num_points),Sbup_y(1:Num_points)
890     RESTORE Sbup_x_data
900     READ Sbup_x(*)
910     RESTORE Sbup_y_data
920     READ Sbup_y(*)
930     CONTROL @Sbup_xy;SET ("X DATA":Sbup_x(*),"Y DATA":Sbup_y(*),"VISIBLE":1)
940     ASSIGN @Sbup_button TO WIDGET "PUSHBUTTON";SET
("X":200,"Y":278,"HEIGHT":30,"LABEL":"DONE"),PARENT @Sbup_panel
950     ON EVENT @Sbup_button,"ACTIVATED" GOSUB Sbup_done
960     RETURN !sbup_disp
970     !
980 Sbup_done: ASSIGN @Sbup_panel TO *
990     DEALLOCATE Sbup_x(*),Sbup_y(*)
1000    RETURN
1010    !
1020 Sbdn_disp:!
1030    ASSIGN @Sbdn_panel TO WIDGET "PANEL";SET
("X":4,"Y":420,"WIDTH":485,"HEIGHT":345,"SIZE CONTROL":"RESIZE CHILDREN")
1040    CONTROL @Sbdn_panel;SET ("TITLE":"Starboard Wing: Lower Skin Strain")
1050    ASSIGN @Sbdn_xy TO WIDGET "XY GRAPH";SET
("X":10,"Y":10,"WIDTH":450,"HEIGHT":265,"VISIBLE":0),PARENT @Sbdn_panel
1060    CONTROL @Sbdn_xy;SET ("CURRENT AXIS":"X","AUTOSCALE":1,"AXIS
LABEL":"Vibration Frequency (Hz)")
1070    CONTROL @Sbdn_xy;SET ("CURRENT AXIS":"Y","AUTOSCALE":1,"AXIS
LABEL":"Micro Strain")
1080    RESTORE Sbdn_num_points
1090    READ Num_points
1100    ON ERROR GOTO Sbdn_alloc
1110    DEALLOCATE Sbdn_x(*),Sbdn_y(*)
1120    !
1130 Sbdn_alloc: OFF ERROR
1140    ALLOCATE INTEGER Sbdn_x(1:Num_points),Sbdn_y(1:Num_points)
```

```
1150   RESTORE Sbdn_x_data
1160   READ Sbdn_x(*)
1170   RESTORE Sbdn_y_data
1180   READ Sbdn_y(*)
1190   CONTROL @Sbdn_xy;SET ("X DATA":Sbdn_x(*),"Y DATA":Sbdn_y(*),"VISIBLE":1)
1200   ASSIGN @Sbdn_button TO WIDGET "PUSHBUTTON";SET
("X":200,"Y":278,"HEIGHT":30,"LABEL":"DONE"),PARENT @Sbdn_panel
1210   ON EVENT @Sbdn_button,"ACTIVATED" GOSUB Sbdn_done
1220   RETURN
1230   !
1240 Sbdn_done: ASSIGN @Sbdn_panel TO *
1250   DEALLOCATE Sbdn_x(*),Sbdn_y(*)
1260   RETURN
1270   !
1280 Prup_disp:!
1290   ASSIGN @Prup_panel TO WIDGET "PANEL";SET
("X":535,"Y":4,"WIDTH":485,"HEIGHT":345,"SIZE CONTROL":"RESIZE CHILDREN")
1300   CONTROL @Prup_panel;SET ("TITLE":"Port Wing: Upper Skin Strain")
1310   ASSIGN @Prup_xy TO WIDGET "XY GRAPH";SET
("X":10,"Y":10,"WIDTH":454,"HEIGHT":265,"VISIBLE":0),PARENT @Prup_panel
1320   CONTROL @Prup_xy;SET ("CURRENT AXIS":"X","AUTOSCALE":1,"AXIS
LABEL":"Vibration Frequency (Hz)")
1330   CONTROL @Prup_xy;SET ("CURRENT AXIS":"Y","AUTOSCALE":1,"AXIS LABEL":"Micro
Strain")
1340   RESTORE Prup_num_points
1350   READ Num_points
1360   ON ERROR GOTO Prup_alloc
1370   DEALLOCATE Prup_x(*),Prup_y(*)
1380   !
1390 Prup_alloc: OFF ERROR
1400   ALLOCATE INTEGER Prup_x(1:Num_points),Prup_y(1:Num_points)
1410   RESTORE Prup_x_data
1420   READ Prup_x(*)
1430   RESTORE Prup_y_data
1440   READ Prup_y(*)
1450   CONTROL @Prup_xy;SET ("X DATA":Prup_x(*),"Y DATA":Prup_y(*),"VISIBLE":1)
1460   ASSIGN @Prup_button TO WIDGET "PUSHBUTTON";SET
("X":200,"Y":278,"HEIGHT":30,"LABEL":"DONE"),PARENT @Prup_panel
1470   ON EVENT @Prup_button,"ACTIVATED" GOSUB Prup_done
1480   RETURN
1490   !
1500 Prup_done: ASSIGN @Prup_panel TO *
1510   DEALLOCATE Prup_x(*),Prup_y(*)
1520   RETURN
```

```
1530   !
1540 Prdn_disp:!
1550   ASSIGN @Prdn_panel TO WIDGET "PANEL";SET
("X":535,"Y":420,"WIDTH":485,"HEIGHT":345,"SIZE CONTROL":"RESIZE CHILDREN")
1560   CONTROL @Prdn_panel;SET ("TITLE":"Port Wing: Lower Skin Strain")
1570   ASSIGN @Prdn_xy TO WIDGET "XY GRAPH";SET
("X":10,"Y":10,"WIDTH":450,"HEIGHT":265,"VISIBLE":0),PARENT @Prdn_panel
1580   CONTROL @Prdn_xy;SET ("CURRENT AXIS":"X","AUTOSCALE":1,"AXIS
LABEL":"Vibration Frequency (Hz)")
1590   CONTROL @Prdn_xy;SET ("CURRENT AXIS":"Y","AUTOSCALE":1,"AXIS LABEL":"Micro
Strain")
1600   RESTORE Prdn_num_points
1610   READ Num_points
1620   ON ERROR GOTO Prdn_alloc
1630   DEALLOCATE Prdn_x(*),Prdn_y(*)
1640   !
1650 Prdn_alloc: OFF ERROR
1660   ALLOCATE INTEGER Prdn_x(1:Num_points),Prdn_y(1:Num_points)
1670   RESTORE Prdn_x_data
1680   READ Prdn_x(*)
1690   RESTORE Prdn_y_data
1700   READ Prdn_y(*)
1710   CONTROL @Prdn_xy;SET ("X DATA":Prdn_x(*),"Y DATA":Prdn_y(*),"VISIBLE":1)
1720   ASSIGN @Prdn_button TO WIDGET "PUSHBUTTON";SET
("X":200,"Y":278,"HEIGHT":30,"LABEL":"DONE"),PARENT @Prdn_panel
1730   ON EVENT @Prdn_button,"ACTIVATED" GOSUB Prdn_done
1740   RETURN
1750   !
1760 Prdn_done: ASSIGN @Prdn_panel TO *
1770   DEALLOCATE Prdn_x(*),Prdn_y(*)
1780   RETURN
1790   !
1800 Finis:   !
1810   ASSIGN @Panel TO *                ! Delete PANEL widget
1820   END
```

```
10      ! *******************************************************
20      ! Example: XY GRAPH Shared Traces
30      !
40      ! This program shows one way to generate several
50      ! traces using a single set of data.
60      !
70      !                          NOTE
80      !
90      ! To exit this program, type stop or press CONTINUE.
100     !
110     ! *******************************************************
120     !
130     ASSIGN @Graph TO WIDGET "XY GRAPH"
140     CONTROL @Graph;SET ("TITLE":" Example: XY GRAPH Shared Traces")
150     CONTROL @Graph;SET ("SHARED X":1,"TRACE COUNT":20)
160     CONTROL @Graph;SET ("CURRENT TRACE":0,"POINT CAPACITY":101,"TRACE
VISIBLE":0)
170     CONTROL @Graph;SET ("CURRENT AXIS":"X","AUTOSCALE":1)
180     CONTROL @Graph;SET ("CURRENT AXIS":"Y","AUTOSCALE":1)
190     !
200     INTEGER I,J,X(0:100)
210     FOR I=0 TO 100
220       X(I)=I*2
230     NEXT I
240     CONTROL @Graph;SET ("CURRENT TRACE":1,"X DATA":X(*))
250     DIM Y(0:100)
260     FOR I=1 TO 20
270       FOR J=0 TO 100
280         Y(J)=I+SIN((I+I/4)*J*PI/50)
290       NEXT J
300       CONTROL @Graph;SET ("CURRENT TRACE":I,"Y DATA":Y(*))
310     NEXT I
320     CONTROL @Graph;SET ("CURRENT TRACE":0,"TRACE VISIBLE":1)
330     PAUSE
340     !
350 Finis: !
360     ASSIGN @Graph TO *              ! Delete XY GRAPH widget
370     END
```

```
10      ! ****************************************************************
20      ! Example: XY GRAPH Widget
30      !
40      ! This program displays two random traces and allows you
50      ! to set various trace marker modes on the two traces.
60      !
70      ! ****************************************************************
80      !
90      RANDOMIZE INT(10^7*FRACT(TIMEDATE))
100   !
110   ! Define colors
120   !
130   INTEGER Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
140   DATA 0,1,2,3,4,5,6,7
150   READ Black,White,Red,Yellow,Green,Cyan,Blue,Magenta
160   !
170   ! Variables Definitions:
180   !
190   INTEGER N
200   REAL R
210   !
220   ! Trace data buffers
230   !
240   REAL X(0:20),Y1(0:20),Y2(0:20)
250   !
260   ! SYSTEM MENU entries
270   !
280   DIM M$(0:5)[16]
290   DATA "No Markers","One Marker","Two Markers"
300   DATA "Delta Markers","Ratio Markers","Quit"
310   READ M$(*)
320   !
330   ! Variables to hold display coordinates
340   !
350   INTEGER D(1:4),Cursor,Dw,Dh,Gx,Gy,Gw,Gh
360   !
370   ! Get display size
380   !
390   GESCAPE CRT,3;D(*)
400   Dw=D(3)-D(1)
410   Dh=D(4)-D(2)
```

```
420   !
430   CLEAR SCREEN
440   !
450   ! Create GRAPH dimensions
460   !
470   Gw=Dw*.7
480   Gh=Dh
490   Gx=(Dw-Gw)/2
500   Gy=(Dh-Gh)/2
510   !
520   ! Build XY GRAPH.
530   !
540   ASSIGN @Graph TO WIDGET "XY GRAPH";SET ("VISIBLE":0)
550   CONTROL @Graph;SET ("SYSTEM MENU":M$(*),"MINIMIZABLE":1)
560   CONTROL @Graph;SET ("TITLE":" Example: XY GRAPH Widget")
570   CONTROL @Graph;SET ("SHARED X":1,"TRACE COUNT":2)
580   CONTROL @Graph;SET ("SHOW GRID":1,"TRACE BACKGROUND":White)
590   CONTROL @Graph;SET ("X":Gx,"Y":Gy+50,"WIDTH":Gw,"HEIGHT":.65*Gh)
600   !
610   ! Set X axis attributes
620   !
630   CONTROL @Graph;SET ("CURRENT AXIS":"X","ORIGIN":0,"RANGE":1)
640   !
650   ! Set Y axis attributes
660   !
670   CONTROL @Graph;SET ("CURRENT AXIS":"Y","ORIGIN":0,"RANGE":100)
680   !
690   ! Set up X data
700   !
710   FOR N=0 TO 20
720     X(N)=N/20
730   NEXT N
740   CONTROL @Graph;SET ("CURRENT TRACE":1,"X DATA":X(*))
750   !
760   ! Draw the curves
770   !
780   R=0
790   FOR N=0 TO 20
800     Y1(N)=R
810     R=R+6*RND
820   NEXT N
830   CONTROL @Graph;SET ("TRACE PEN":Red,"Y DATA":Y1(*))
```

```
840   !
850   R=0
860   FOR N=0 TO 20
870     Y2(N)=R
880     R=R+9*RND
890   NEXT N
900   CONTROL @Graph;SET ("CURRENT TRACE":2,"Y DATA":Y2(*))
910   CONTROL @Graph;SET ("TRACE PEN":Blue)
920   !
930   ! Loop and wait to exit
940   !
950   CONTROL @Graph;SET ("VISIBLE":1)
960   ON EVENT @Graph,"SYSTEM MENU" GOSUB Handler
970   !
980   LOOP
990     WAIT FOR EVENT
1000  END LOOP
1010  STOP
1020  !
1030  ! ********** End of Main Program **********
1040  !
1050  ! This handler traps the SYSTEM MENU event and determines which
1060  ! entry caused the trap. It then performs the appropriate actions.
1070  !
1080 Handler: !
1090  STATUS @Graph;RETURN ("SYSTEM MENU EVENT":N)
1100  SELECT N
1110  !
1120  ! No markers
1130  !
1140  CASE 0
1150    CONTROL @Graph;SET ("MARKER":"NONE")
1160  !
1170  ! One marker
1180  !
1190  CASE 1
1200    CONTROL @Graph;SET ("MARKER":"ONE","MARKER1 TRACE":1)
1210    CONTROL @Graph;SET ("MARKER1 X":X(10),"MARKER1 Y":Y1(10))
1220  !
1230  ! Two markers
1240  !
1250  CASE 2
```

```
1260     CONTROL @Graph;SET ("MARKER":"TWO")
1270     CONTROL @Graph;SET ("MARKER1 TRACE":1,"MARKER2 TRACE":2)
1280     CONTROL @Graph;SET ("MARKER1 X":X(10),"MARKER2 X":X(10))
1290     CONTROL @Graph;SET ("MARKER1 Y":Y1(10),"MARKER2 Y":Y2(10))
1300   !
1310   ! Delta markers
1320   !
1330   CASE 3
1340     CONTROL @Graph;SET ("MARKER":"DELTA")
1350     CONTROL @Graph;SET ("MARKER1 TRACE":1,"MARKER2 TRACE":2)
1360     CONTROL @Graph;SET ("MARKER1 X":X(10),"MARKER2 X":X(10))
1370     CONTROL @Graph;SET ("MARKER1 Y":Y1(10),"MARKER2 Y":Y2(10))
1380   !
1390   ! Ratio markers
1400   !
1410   CASE 4
1420     CONTROL @Graph;SET ("MARKER":"RATIO")
1430     CONTROL @Graph;SET ("MARKER1 TRACE":1,"MARKER2 TRACE":2)
1440     CONTROL @Graph;SET ("MARKER1 X":X(10),"MARKER2 X":X(10))
1450     CONTROL @Graph;SET ("MARKER1 Y":Y1(10),"MARKER2 Y":Y2(10))
1460   !
1470   ! Exit program
1480   !
1490   CASE 5
1500     ASSIGN @Graph TO *              ! Delete XY GRAPH widget
1510     STOP
1520   END SELECT
1530   RETURN
1540   !
1550   END
```

FONT

Specifies the font to be used for any text in the FILE widget. For a description of allowed fonts, see Specifying FONT Attribute Values. This attribute is accepted for compatibility with previous versions of HTBasic for Windows, but will not change the appearance of the widget.

PEN

Specifies the color to be used for the TEXT in the widget. For a discussion of allowed pens, see Settable Pens Table.

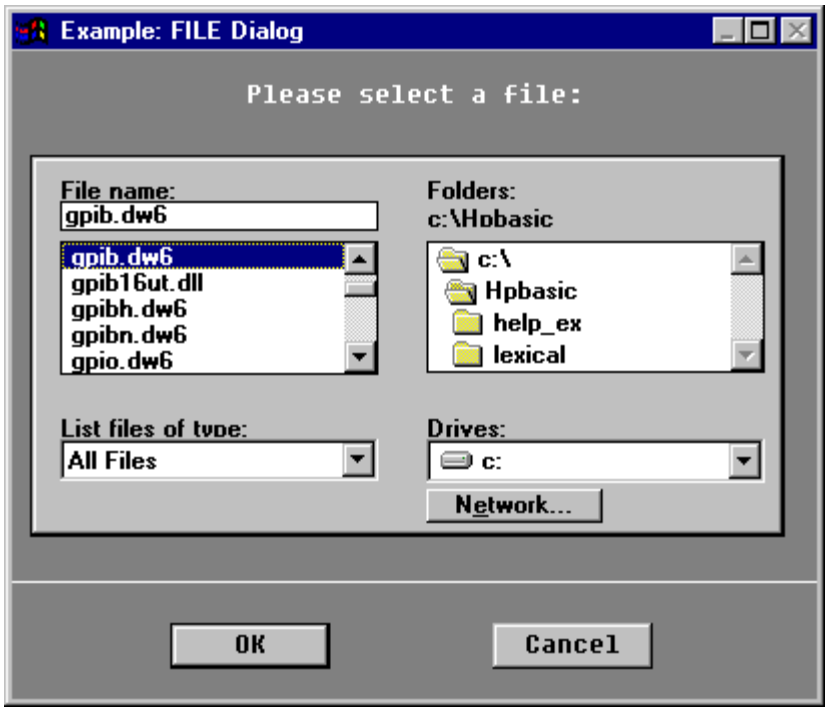
HTBasic for Windows

FILE Dialog

FILE Dialog

Prompts the operator to enter a filename

Example Image



Example Program

See FILE Dialog for an example program that produces a display similiar to that shown above.

NOTE

See the following program for another example using the FILE dialog.

Dialogs Tests

Attributes

See FILE Dialog Attributes for the FILE dialog attribute list.

Remarks

The FILE dialog provides interactive traversal of existing directories and files.

HTBasic for Windows

FILE Dialog Attributes

FILE Dialog Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
DEFAULT BUTTON	Numeric	Any valid index into the DIALOG BUTTONS array	0
DIALOG BUTTONS	String array	Any valid string array	A two-element array that contains OK.and Cancel
DIRECTORY	String	An existing, readable directory	current MSI directory
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE	Numeric	0,1	1
PATTERN	String	Any legal UX wildcard pattern	* (display all files in

			current directory)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE	Numeric	0,1	1 (resizable, special border appears around the dialog)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SELECTION	String	Any	Null string
TITLE	String	Any valid string	FILE
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Screen or parent work area upper-left corner

DIRECTORY

This attribute is a valid directory and MSVS name. DIRECTORY specifies the directory to be CATalogued in the "Directories" and "Files" COMBO boxes. Users can also change this directory interactively by entering a new directory in the "Current Directory" by typing it in or selecting a directory

FONT

Specifies the font to be used for any text in the FILE dialog. For a description of allowed fonts, see Specifying FONT Attribute Values. This attribute is accepted for compatibility with previous versions of HTBasic for Windows, but will not change the appearance of the dialog except for the prompt and buttons.

PATTERN

Specifies the UX wildcard used to filter the display of files.

PEN

Specifies the color to be used for the TEXT in the dialog. For a discussion of allowed pens, see Settable Pens Table.

SELECTION

Specifies the current file selection. It includes the directory, filename, and MSVS Users can also change the file interactively in the "File selection:" panel by typing it in or selecting a file name from the "Files" listbox.

HTBasic for Windows

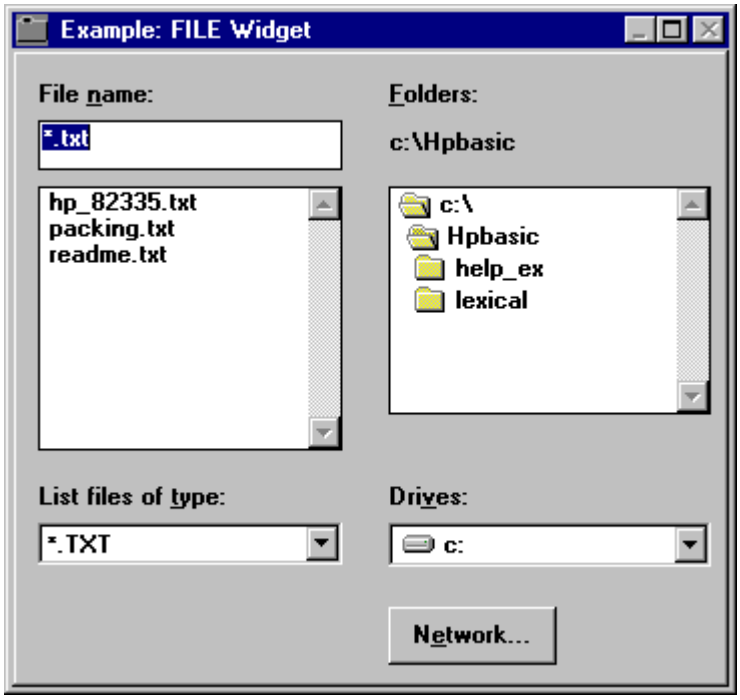
FILE Widget

FILE Widget

Provides interactive traversal of existing directories and files.
(Widget is not valid for Windows 3.1)

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See *FILE Widget* for a program creates a FILE widget and provides a display similar to that shown above.

Attributes

See FILE Widget Attributes for the FILE widget attribute list.

Remarks

The FILE widget is the input device for entering filenames. This widget provides interactive traversal of existing directories and files. FILE is the input device for filename.

NOTE

The FILE widget is not valid for Windows 3.1.

The DIRECTORY attribute is a valid directory and MSVS name. DIRECTORY specifies the directory to be CATalogued in the "Directories" and "Files" LIST boxes. You can also change this directory interactively by entering a new directory by typing it in the File Name field or selecting a directory from any of the following: directories list box, drives, COMBO box, or Network dialog.

Events

Events for the FILE widget are:

- SELECTION
- SYSTEM MENU

SELECTION

This event is generated any time the user makes a selection from the Files:COMBO box or presses **Return** after entering a filename in the edit field.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

FILE Widget Attributes

FILE Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
DIRECTORY	String	An existing, readable directory	current MSI directory (".")
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Varies
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE [5]	Nu	0,1	0 (not minimizable,

	meric		button does not appear in title bar)
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PATTERN	String	Any legal UX wildcard pattern	* (display all files in current directory)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SELECTION	String	Any	Null string
STACKING ORDER	Numeric	0 to the number of siblings [4]	0
SYSTEM MENU [5]	String or string array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0

SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu - 1	0
TITLE [5]	String	Any valid string	FILE
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Varies
X	Numeric	Any integer, number of pixels from 0,0 [6]	0
Y	Numeric	Any integer, number of pixels from 0,0 [6]	0

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

HTBasic for Windows

File Installation Locations

HTBasic for Windows installs its files, by default, in C:/PROGRAM FILES/HTBASIC. BPLUS.DW6 and all other WI*.DLL files must remain in this directory. The remaining ancillary files can be located in a number of places, as described in Search Sequence for Ancillary Files.

HTBasic for Windows

File System Issues

This topic describes various aspects of HTBasic for Windows' use of the file system, such as naming conventions, installation and search directories, including:

- File Types
- File Installation Locations
- Search Sequence for Ancillary Files

Click the item name for a description of the item.

HTBasic for Windows

File Types

The following table shows HTBasic Plus file types.

The HTBasic for Windows binary does not include any of the ancillary files. These files must be in a directory that HTBasic for Windows searches.

File Type	Description
BPLUS.DW6	HTBasic Plus binary
CONFIG	Configuration file
BMxxxx or .BMP	Bitmap files
WIxxxx.DLL	Widget files
HLxxxx or .HLP	Online Help files
GLxxxx or .HPG	HPGL files
.SCR	Screen Builder files

HTBasic for Windows

Getting Started With Dialogs

Introduction

A dialog is an HTBasic Plus entity that is created on the screen with the DIALOG statement from an executing BASIC program or from the command line. From a programming point of view, the DIALOG statement is similar to the INPUT statement of BASIC. From the user's point of view, a dialog is a popup that disappears after the user supplies a response.

Using the DIALOG Command

Dialogs are created and modified with the DIALOG command. The DIALOG command consists of a dialog type (the name of the dialog), a prompt string, a selected button option, optional SET and RETURN attributes, and a TIMEOUT option. See the DIALOG command description for details on the DIALOG command.

All DIALOG command statements MUST fit on one line, since the program will wait until the DIALOG statement is executed before proceeding.

For an overview of using dialogs, copy the example help program Dialogs into your HTBasic for Windows Editor and run the program. Click Dialog Tests to select a dialog. See Copying Example Programs for procedures to copy example programs.

HTBasic for Windows

Glossary

HTBasic for Windows

HTBasic Plus Example Programs

Overview

This topic lists the program names for the example programs in online Help. See Copying Example Programs for procedures to copy example programs into the HTBasic for Windows Editor.

NOTE

These example programs are also stored in the plus examples directory in your install directory (default is c:/program files/htbasic/plus examples).

Example Programs Listing

Click the program name of the example for a listing of the program code. The filename listed in the table is the name of the file that is stored in the *plus examples* directory.

Program Name	filename	Program Name	filename
Alarm Clock	alarmclk	LIST Widget	listwdgt
Background Bitmap	bkgndbmp	Menu System	menusyst
Bar Meter Limits	barmtrlm	METER Widget	metrwdgt
BAR Widget	barwdgt	METER/SLIDER Widgets	metrsldr
BARS Test	barstest	NUMBER Dialog	nbrdial
BARS Widget	barswdgt	NUMBER Widget	nbrw
BITMAP Widget	bmapwdgt	Oven Control	oven
Bomb Squad (*CREATE)	bsquadcr	Passwords	passwd
Bomb Squad (*LOAD)	bsquadld	PID Controller	pid
Calculator	calculat	PRINTER Widget	prtrwdgt
Changing the Font	chgfont	PUSHBUTTON Events	pbevents
CLOCK Widget	clkwdgt	PUSHBUTTON Widget	pbwdgt
COMBO Dialog	combodlg	QUESTION Dialog	qdlg
COMBO Test	combotst	RADIOBUTTON Widget	rbwdgt
COMBO Widget	combowdgt	Rock-Paper-Scissors Game	rps
		SCROLLBAR Widget	sbwdgt

Context-Sensitive Help	combdial
Cyclical PUSHBUTTON	combtest
Dialogs Tests	combwdgt
Engine Monitor - Child	ctxthelp
Engine Monitor - Level-0	cyclpush
Engine Monitor - Panel	dialogs
Environmental Chamber	emchild
ERROR Dialog	emlvl0
FILE Dialog	empanel
FILE Widget	chamber
Frequency Response	errdial
Function Generator	filedial
Hammer Game	filewdgt
HPGL VIEW in PANEL	freqresp
HPGL VIEW Viewer	funcgen
HPGL VIEW Widget	hammer
Ice Cream Sundae	hpglpanl
INFORMATION Dialog	hpglview
KEYPAD Dialog	hpglwdgt
KEYPAD Widget	icecream
LABEL Widget	infodial
LIMITS Widget	keydial
Lissajous Patterns	keywdgt
LIST Dialog	lablwdgt
SLIDER Test	dgt
SLIDER Widget	
Slot Machine	oven cont
STRING Dialog	
STRING Editor	
STRING Widget	password
STRIPCHART (counter)	
STRIPCHART(rescale)	
STRIPCHART(scrolling)	pidcntlr
STRIPCHART (sine waves)	
STRIPCHART Widget	
SYSTEM Widget as a Child	ptrwdgt
SYSTEM Widget Event Disabling	pushevnt
SYSTEM Widget Event Handler	
SYSTEM Widget Event Queuing	pushbtn
Tab Groups	
Tic-Tac-Toe	
TOGGLEBUTTON Widget	quesdial
Using ON EVENT	
WARNING Dialog	
Wing Stress/Vibration Analysis	radiobtn
XY GRAPH Shared Traces	rcksci
XY GRAPH Widget	prscrlwdgt
	slidrtst
	slidrwgt
	slot

s

str
gd
ial

st
ng
ed
it

str
g
w
dg
t

str
ip
ctr

str
ipr
es

str
ip
sc
r

str
ip
si
n

str
p
w
dg
t

sy
st
ch
d

sy
st
di
s

sy
st
ha
nd

sy
st
qu
e

ta
bg
ro
up

tic
ta

ct
o

to
gg
lbt
n

on
ev
en
t

w
ar
nd
ial

str
es
vi
b

xy
tra
ce

xy
w
dg
t

HTBasic Plus Programming Topics

HTBasic Plus Examples

- HTBasic Plus Example Programs

Widgets Overview

- Widget Displays
- Widget Attributes
- Widget Hierarchy
- Widget Format
- List of Widgets

Programming Widgets

- Steps to Create Widgets
- Creating/Modifying Widgets
- Destroying Widgets
- Setting/Changing Attributes
- Querying Widget Attributes
- Event-Initiated Branching

Using Special Widgets

- PANEL Widget
- SEPARATOR Widget
- SYSTEM Widget

Other Widget Programming Topics

- Using Tab Groups
- Creating Pulldown Menus
- Creating System Menus

Dialogs Description

- Dialogs Attributes
- Dialogs Format
- List of Dialogs

Programming Dialogs

- Getting Started With Dialogs
- Creating/Modifying Dialogs
- Using Array Variables

HTBasic Plus Reference Topics

HTBasic for Windows Interactions

- Using Registers
- Sharing Input Devices/Display
- File System Issues
- Custom Configurations

Screen Builder Application

- Screen Builder Overview
- Screen Builder Initial Condition
- Screen Builder Interactive Features
- Screen Builder Menu
- Screen Building Process
- Screen Builder Example

Keywords Reference

- List of Keywords
- Keyword Format
- Keyword Conventions

General Reference

- Example Programs
- Error Messages
- Glossary

HEIGHT

Dialogs and Widgets. The value of this attribute specifies the height of the dialog or widget, in units of pixels. The value of HEIGHT is set automatically when you set the ROWS attribute on some widgets. See INTERDEPENDENT ATTRIBUTES for information on how HEIGHT interacts with the ROWS,COLUMNS, WIDTH, and FONT attributes for some widgets.

HELP FILE

Widgets Only. The HELP FILE attribute provides access to user-developed online help files or to HTBasic for Windows-provided help files. If both HELP FILE and HELP TOPIC are turned off in the CONFIG file, access to all help is turned off. If HELP FILE or HELP TOPIC is turned off locally, a screen indicating that no help is available will be displayed when the right mouse button or the **F1** key is pressed.

HELP TOPIC

Widgets Only. The HELP TOPIC attribute provides access to an HTBasic for Windows *index* or *topic_id* from a file. If both HELP FILE and HELP TOPIC are turned off in the CONFIG file, access to all help is turned off. If HELP FILE or HELP TOPIC is turned off locally, a screen indicating no help is available will be displayed when the right mouse button or the **F1** key is pressed.

HPGL

Hewlett Packard Graphics Language. Used to communicate between HTBasic and plotters.

HPGL FILE

The contents of the named file will be interpreted as HPGL commands with which to paint the contents of the HPGL VIEW widget. The BASIC file type must be regular DOS (not BDAT or ASCII).

RETAIN RASTER

When RETAIN RASTER is set (value of 1), the HTBasic for Windows widget management software will pop the widget to the "top" on the screen, draw it in its entirety, take a "snapshot" of it, and retain the pixel map. Hence, the HPGL VIEW widget will be redrawn quickly each time it is moved or repaired.

Thereafter, if you change the widget's HEIGHT, WIDTH, BACKGROUND, or HPGL FILE attribute values, the widget management software will pop the widget to the "top" on the screen, draw it in its entirety, take a "snapshot" of it, and retain the pixel map.

NOTE

Because your program will use a significantly larger amount of memory when you set RETAIN RASTER to 1, do so only when it is important to redraw the HPGL widget quickly.

HTBasic for Windows

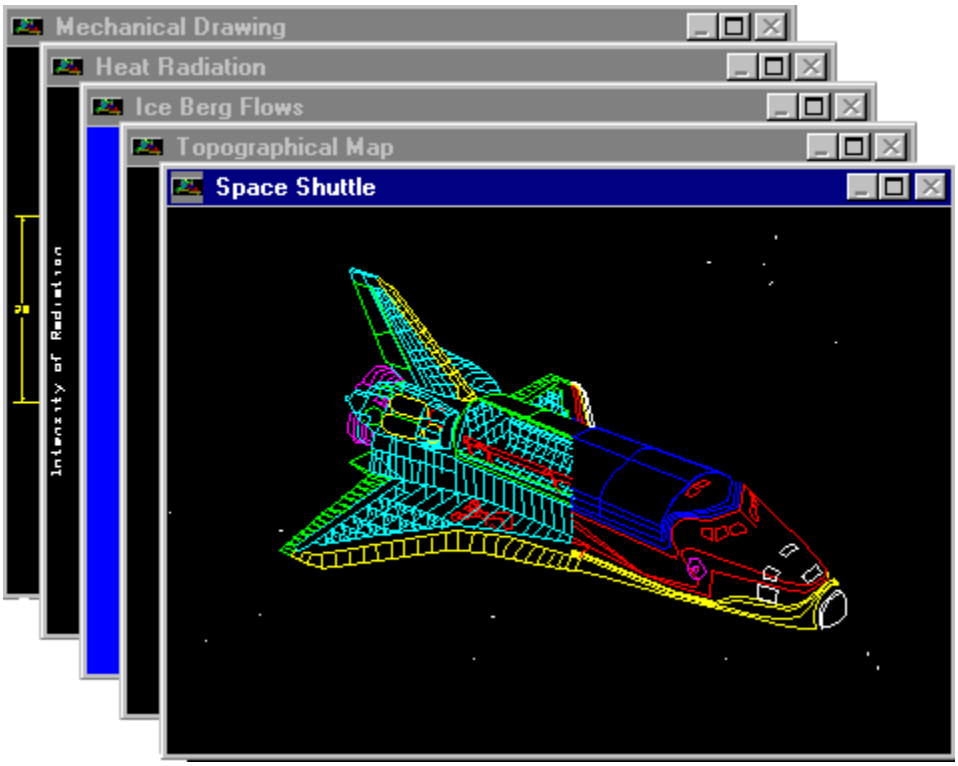
HPGL VIEW Widget

HPGL VIEW Widget

Displays HPGL graphics files

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See *HPGL VIEW Widget* for an example program that provides a display similar to that shown above.

NOTE

See the following programs for other examples using the HPGL VIEW widget:

HPGL VIEW Viewer

HPGL VIEW in PANEL

Attributes

See HPGL VIEW Widget Attributes for the HPGL VIEW widget attribute list.

Remarks

The HPGL VIEW widget displays HPGL graphics files. HPGL VIEW supports the following (most frequently used) HPGL commands. HPGL commands that are not implemented are ignored by the HPGL VIEW widget.

- IP
- IW
- PA
- PD
- PR
- PU
- SP

The HPGL VIEW widget provides functionality similar to the PRINTER widget, but allows you to display plotter graphics commands instead of text. For example, execute the following code to create an HPGL VIEW widget and set it to display the file "Graffile" (an arbitrary HPGL graphics file). If the file is not found or it is not an HPGL file, an error message is generated.

```
ASSIGN @Hpgl TO WIDGET "HPGL VIEW" CONTROL @Hpgl;SET ("HPGL FILE":"Graffile")
```

For an HPGL VIEW widget to display graphics generated by an HTBasic for Windows program, you first redirect the output to a file with PLOTTER IS statement, and then set the widget to that file.

The HPGL VIEW widget supports only a subset of all HPGL commands. For example, area fills are not supported and files that include them will not display correctly.Since the RETAIN RASTER attribute default setting is 0, any repainting will require rereading the file. However, if you set RETAIN RASTER to 1, HTBasic for Windows will take a snapshot of the pixels and retain them. Thus, repaints will be drawn from memory instead of from the file.

Events

The event for the HPGL VIEW widget is:

- SYSTEM MENU

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

HPGL VIEW Widget Attributes

HPGL VIEW Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
BORDER [2]	Numeric	0,1	1
HEIGHT	Numeric	Any integer number (of pixels)	300 pixels
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
HPGL_FILE	String	The string value must point to an existing, readable file	None
INSIDE_HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE_WIDTH	Numeric	Any integer number (of pixels)	Varies
MAXIMIZABLE [4]	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE [4]	Numeric	0,1	0 (not minimizable, button does not appear in title

			bar)
MOVABLE [4]	Numeric	0,1	1 (movable, click on title bar and drag)
RESIZABLE [4]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
RETAIN RASTER	Numeric	0,1	0
STACKING ORDER	Numeric	0 to number of siblings [3]	0
SYSTEM MENU [4]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [4]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [4]	Numeric	0 to number of items in system menu -1	0
TITLE [4]	String	Any valid string	HPGL VIEW
USER DATA	String	Any valid string	None

VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	400 pixels
X	Numeric	Any integer, number of pixels from 0,0 [5]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [5]	Autoplacement

[1] Read the CONFIG file or query for the default with a STATUS command

[2] Child widget only

[3] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[4] For level-0 widgets only

[5] Screen or parent work area upper-left corner

HTBasic for Windows

HTBasic for Windows Interactions

This topic describes interactions between HTBasic for Windows HTBasic Plus and HTBasic for Windows. It presents suggestions to optimize the operation of these two systems, including:

- Using Registers
- Sharing Input Devices/Display
- File System Issues
- Using the CONFIG File

Click the item title to view each item in this topic.

Help

The online help system that includes information on all HTBasic for Windows keywords, programming examples, and other online information. Online help topics are linked to one another through hyperlinks.

HTBasic for Windows

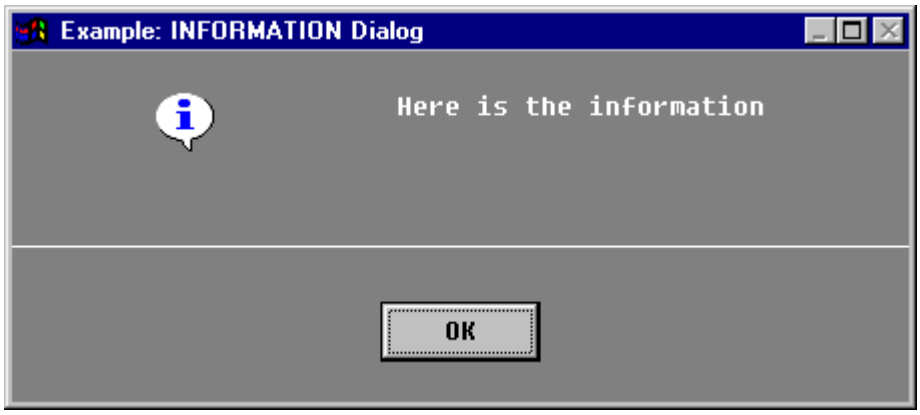
INFORMATION Dialog

INFORMATION Dialog

Displays information and suspends program execution until the operator acknowledges the information

---

Example Image



Example Program

See INFORMATION Dialog for an example program that produces a display similiar to that shown above.

NOTE

See the following program for another example using the INFORMATION dialog.

Dialogs Tests

Attributes

See INFORMATION Dialog Attributes for the INFORMATION dialog attribute list.

Remarks

None.

HTBasic for Windows

INFORMATION Dialog Attributes

INFORMATION Dialog Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
DEFAULT BUTTON	Numeric	Any valid index into the DIALOG BUTTONS array	0
DIALOG BUTTONS	String array	Any valid string array	A single element array containing "OK"
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE	Numeric	0,1	1
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABL	Nu	0,1	1 (resizable,

E	meri c		special border appears around the dialog)
RESTORE SCREEN	Nu meri c	0,1	0 (do not restore the screen)
TITLE	Strin g	Any valid string	INFORMATIO N
USER DATA	Strin g	Any valid string	None
VERSION	Strin g	Any valid string	Current version
WIDTH	Nu meri c	Any integer number (of pixels)	Autosizing
X	Nu meri c	Any integer, number of pixels from 0,0 [3]	Autoplacement
Y	Nu meri c	Any integer, number of pixels from 0,0 [3]	Autoplacement

[1] For an explanation of Font and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Screen or parent work area upper-left corner

FONT

Specifies the font to be used for any text in the INFORMATION dialog. For a description of allowed fonts, see Specifying FONT Attribute Values.

PEN

Specifies the color to be used for the foreground area of the dialog.
For a discussion of allowed pens, see Settable Pens Table.

INSIDE HEIGHT

Widgets. *Return-only Attribute*. The value of INSIDE HEIGHT is the height of the work area of the widget. The work area is the area of the widget excluding the title bar and resize border. Get the value of this attribute with a STATUS command and use the value for positioning child widgets

INSIDE WIDTH

Widgets. *Return-Only Attribute*. The value of INSIDE WIDTH is the width of the work area of the widget. The work area is the area of the widget excluding the title bar and resize border. Get the value of this attribute with a STATUS command and use the value for positioning child widgets.

INTERDEPENDENT ATTRIBUTES

The COLUMNS attribute, FONT attribute, and WIDTH common attribute are interdependent. Similarly, the ROWS attribute, FONT attribute, and HEIGHT common attribute are interdependent.

If You Set:	Then:
ROWS	The value of HEIGHT will be automatically adjusted to provide the number of specified ROWS based on the value of the FONT attribute. Applies only to the listbox portion. Selection field is not included in the ROWS attribute's argument.
COLUMNS	The value of WIDTH will be automatically adjusted to provide the number of specified COLUMNS based on the value of the FONT attribute.
WIDTH	The value of COLUMNS will be automatically adjusted to provide the specified WIDTH based on the value of the FONT attribute. Specifying a -1 yields a default value.
HEIGHT	The value of ROWS will be automatically adjusted to provide the specified HEIGHT based on the value of the FONT attribute. Applies only to the listbox portion. Display and edit are not included in the ROWS attribute's argument. Specifying a -1 yields a default value.
FONT	The values of ROWS and COLUMNS will be automatically adjusted to fit into the size of the widget (that is, WIDTH by HEIGHT does

not change).

For Constant Widget Size:

If you want to place a widget in a PANEL or otherwise so that it will not grow to encroach on its neighbor widgets, set the values of WIDTH and HEIGHT to the desired numbers of pixels. ROWS and COLUMNS will adjust based on the value of the FONT attribute.

Setting the value of FONT will not change the size of a widget. WIDTH and HEIGHT set the exterior dimensions of the widget, whereas ROWS and COLUMNS affect only the work area dimensions.

Incompatible Attribute Values:

You do not have to worry about setting these values incompatibly, because the attribute you set last in your program will take precedence, as shown in the table above.

For example, if you set ROWS, COLUMNS, FONT, WIDTH, and HEIGHT in that order in your program, the widget will use your FONT, WIDTH, and HEIGHT values and will ignore your ROWS and COLUMNS values. Instead of using your values, the widget will automatically set the ROWS and COLUMNS attribute values.

INTERDEPENDENT ATTRIBUTES

The COLUMNS attribute, FONT attribute, and WIDTH common attribute are interdependent.

If You Set:	Then:
COLUMNS	The value of WIDTH will be automatically adjusted to provide the number of specified COLUMNS based on the value of the FONT attribute.
WIDTH	The value of COLUMNS will be automatically adjusted to provide the specified WIDTH based on the value of the FONT attribute.
FONT	The value of COLUMNS will be automatically adjusted to fit into the size of the widget (WIDTH does not change)

For Constant Widget Size:

If you want to place a widget in a PANEL or otherwise so that it will not grow to encroach on its neighbor widgets, set the value of WIDTH to the desired numbers of pixels. COLUMNS will adjust based on the value of the FONT attribute.

Setting the value of FONT will not change the size of a widget. WIDTH set the exterior dimensions of the widget, whereas COLUMNS affect only the work area dimensions.

Incompatible Attribute Values:

You do not have to worry about setting these values incompatibly, because the attribute you set last in your program will take precedence, as discussed in the table above.

For example, if you set COLUMNS, FONT, and WIDTH in that order in your

program, the widget will use your FONT and WIDTH values and will ignore your COLUMNS value. Instead of using your value, the widget will automatically set the COLUMNS attribute value.

HTBasic for Windows

Improving Display Appearance

The following table shows some statements you can use to control the appearance of the display. You can execute these statements from the command line while you interact with HTBasic for Windows. Or, you can make them part of your application.

To do This:	Use This Command:
Turn text cursor off	CONTROL CRT,10;0
Turn run light off	RUNLIGHT OFF
Turn softkey labels off	KEY LABELS OFF
Clear text from the display	CLEAR SCREEN
Clear graphics from the display	GCLEAR

HTBasic for Windows

Interacting With Widgets

The following table shows how to use the mouse and keyboard functions to interact with some common HTBasic for Windows widgets.

Using Mouse/Keyboard Functions

Function	Using Mouse	Using Keyboard
Selecting a Menu Button	1. Move pointer over menu's name and click mouse button 2a. Move pointer over menu button and click mouse button **or** 2b. Click mouse button on menu's name and hold 3. Move pointer to menu button and release	1. Press **Extend char** or **Alt** and hold 2. Press **Left Arrow** or **Right Arrow** to move to next or previous menu 3. Press **Up Arrow** or **Down Arrow** to move to button in menu 4. Press **Return** when button has the focus
Selecting a TOGGLEBUTTON or RADIOBUTTON	Move pointer over the button and click mouse button	1. Press **Tab** or **Shift-Tab** to reach the button 2. Press **Space** or **Return**

Selecting a LIST Item	1. Use scrollbar to view all items 2. Click mouse button on the item	1. Use arrow keys to highlight the item 2. Press **Return**
Scrolling with a SLIDER Widget	Drag the slider by moving the mouse while holding down the mouse button	1. Press arrow keys until pointer moves to the area you want to scroll 2. Use the **Up Arrow**, **Down Arrow**, **Prev** keys as required
Entering Text Into a STRING Widget	1. Move pointer to the text field to get text cursor 2. Type in text desired	1. Press **Tab** or **Shift-Tab** to reach field 2. Type in text desired

JUSTIFICATION

Dialogs and LABEL Widget Only. The value of this attribute specifies whether to left justify or center justify the "prompt string" text in the dialog or LABEL widget text area.

CHECK FOR DONE

This attribute, when non-zero, enables loss of focus formatting and the "DONE" event.

COLUMNS

The COLUMNS attribute specifies the width of the widget or dialog display field in units of character cells. See INTERDEPENDENT ATTRIBUTES for the interactions among the COLUMNS, WIDTH, and FONT attributes.

FONT

This attribute determines which font will be used to display text on the KEYPAD widget. See INTERDEPENDENT ATTRIBUTES for the interactions among the COLUMNS, WIDTH, and FONT attributes.
For a description of allowed fonts, see Specifying FONT Attribute Values.

FORMAT

The FORMAT attribute specifies which numeric formatting type is used for the KEYPAD widget. Allowable types are:

- REAL: the default (-1.79769313486 E +308 to +1.79769313486 E +308)
- SHORT INTEGER: two-byte integer (-32768 to 32767 range)
- LONG INTEGER: four-byte integer (-2147483648 to 2147483647 range)
- BINARY: four-byte integer; two's complement (no "-" allowed)
- OCTAL: four-byte integer; two's complement (no "-" allowed)
- HEX: four-byte integer; two's complement (no "-" allowed)

FORMAT LENGTH

The FORMAT LENGTH determines the minimum number of characters used to display formatted numbers (padded with 0s on the left for integer and real formats).

The output is never truncated and the actual length may be greater. (See the TEXT LENGTH attribute.) FORMAT LENGTH is primarily intended for BINARY, OCTAL, and HEX formats to force formatting to a greater number of bytes.

INCREMENT

The INCREMENT attribute rounds the VALUE attribute to the nearest specified INCREMENT when programmatically setting VALUE, pressing **Return**, or loss of focus when CHECK FOR DONE is set. Declaring an INCREMENT of "0" directs HTBasic for Windows to not alter the VALUE. "0" is the default.

KEY BACKGROUND

This attribute determines the pen color used for the background of the buttons on the keypad. For a discussion of allowed pens, see Settable Pens Table. This attribute is accepted for compatibility with previous versions, but will not change the appearance of the widget.

KEY PEN

This attribute determines the pen color used for the characters on the buttons. For a discussion of allowed pens, see Settable Pens Table This attribute is accepted for compatibility with previous versions, but will not change the appearance of the widget.

MAXIMUM

The MAXIMUM attribute determines the maximum value the KEYPAD widget will accept. When changing from one FORMAT to another, KEYPAD will use the lesser of the current maximum value and FORMAT's intrinsic maximum (for positive values).

For example, switching formats from LONG INTEGER to SHORT INTEGER will reduce the maximum KEYPAD value from a four-byte integer to a two-byte integer (see the FORMAT attribute for ranges).

MAXIMUM LENGTH

The MAXIMUM LENGTH attribute specifies the maximum number of characters you can enter into the widget: "0" means no limit. (Reformatting may generate a longer display than the MAXIMUM LENGTH, however.)

MINIMUM

The MINIMUM attribute determines the minimum value the KEYPAD widget will accept. When changing from one FORMAT to another, KEYPAD will use the greater of the current minimum value and FORMAT's intrinsic MINIMUM (for negative values).

For example, switching formats from LONG INTEGER to SHORT INTEGER will reduce the minimum KEYPAD value from a four-byte integer to a two-byte integer (see the FORMAT attribute for ranges).

MODIFIED

The MODIFIED attribute indicates the operator has changed the contents of KEYPAD. When responding to the DONE/RETURN event, check the MODIFIED attribute to see if it has been modified, then set MODIFIED=0 for the next cycle.

PEN

This attribute determines the pen color used for the characters in the display field. For a discussion of allowed pens, see Settable Pens Table.

REAL NOTATION

The REAL NOTATION attribute can be used in conjunction with the "FORMAT":"REAL" attribute only. REAL NOTATION allows you to select the "FORMAT:REAL" presentation in one of these notations:

- FIXED (Example: 12345.6789)
- SCIENTIFIC (Example: 1.233456789E+04)
- ENGINEERING (Example: 12.3456789E+03)

REAL RESOLUTION

The REAL RESOLUTION attribute can be used in conjunction with the "FORMAT":"REAL" attribute only. REAL RESOLUTION allows you to select the number of digits to be displayed to the right of the decimal point.

SHOW EDIT

This attribute determines whether or not the display is shown.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

TEXT LENGTH

Return-Only Attribute. TEXT LENGTH returns the length in characters of the current formatted number.

VALUE

This attribute reflects the current value of the KEYPAD. It is always greater than or equal to MINIMUM and less than or equal to MAXIMUM and rounded to the nearest INCREMENT. When using string to set VALUE, the string is verified to be a valid number within range and increment. If the value of VALUE exceeds MAXIMUM or MINIMUM, Error 565 is generated.

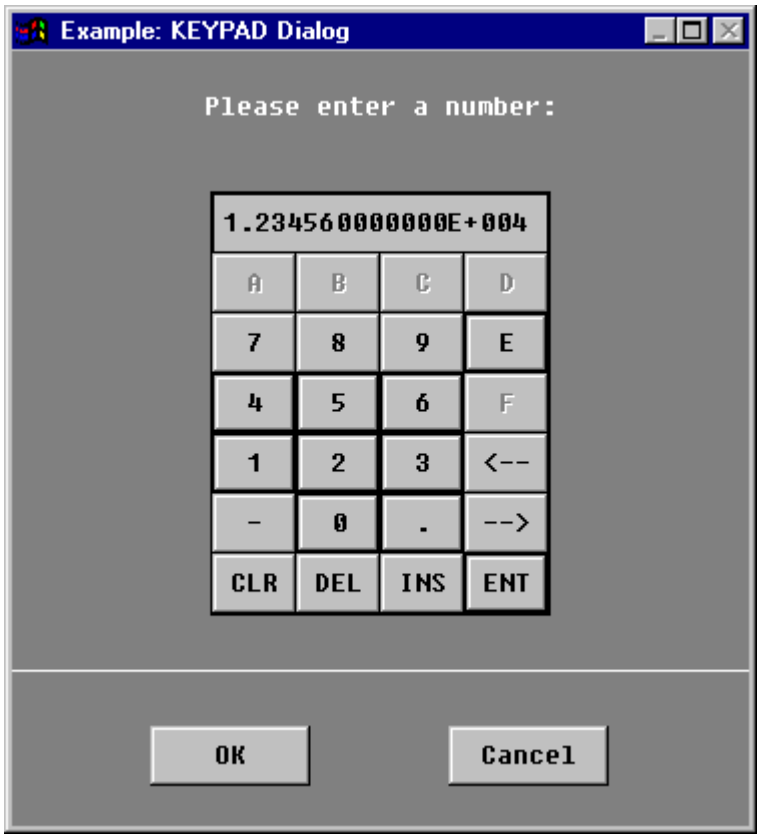
HTBasic for Windows

KEYPAD Dialog

KEYPAD Dialog

Prompts the user to make numeric entries using a mouse,
a touchscreen, or a keyboard

Example Image



See KEYPAD Dialog for an example program that produces a display similiar to that shown above.

NOTE

See the following program for another example using the KEYPAD dialog.

Dialogs Tests

Attributes

See KEYPAD Dialog Attributes for the KEYPAD dialog attribute list.

Remarks

None.

HTBasic for Windows

KEYPAD Dialog Attributes

KEYPAD Dialog Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [2]	Numeric	0,1	1
CHECK FOR DONE	Numeric	0,1	1
COLUMNS	Numeric	Any non-negative number	20
DEFAULT BUTTON	Numeric	Any valid index into the DIALOG BUTTONS array	0
DIALOG BUTTONS	String array	Any valid string array	A two-element array that contains "OK" and "Cancel"
EDITABLE	Numeric	0,1	1
FONT	String	*Font* [1]	System dependent [2]
FORMAT	String	"REAL", "SHORT INTEGER", "LONG INTEGER", "INTEGER", "BINARY", "OCTAL", "HEX"	"REAL"
FORMAT LENGTH	Numeric	0 to 328	0
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing

INCREMENT	Numeric	Any non-negative number	0
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
KEY BACKGROUND	Numeric	Any valid BASIC pen number	System dependent [2]
KEY PEN	Numeric	Any valid pen number	System dependent [2]
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MAXIMUM	Numeric	Between or equal to FORMAT's MINIMUM and MAXIMUM	MAXREAL
MAXIMUM LENGTH	Numeric	0 to 328	328
MINIMIZABLE	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MINIMUM	Numeric	Between or equal to FORMAT's MINIMUM and MAXIMUM	MINREAL
MODIFIED	Numeric	0,1	0
MOVABLE	Numeric	0,1	1
PEN	Numeric	0 to 255	System dependent [2]
REAL NOTATION	String	"FIXED", "SCIENTIFIC", "ENGINEERING"	"SCIENTIFIC"
REAL RESOLUTION	Numeric	0 to 16	12

RESIZABLE	Numeric	0,1	1 (resizable, special border appears around the dialog)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SHOW EDIT	Numeric	0,1	1
TEXT LENGTH	Numeric	Any valid integer	Return only
TITLE	String	Any valid string	KEYPAD
USER DATA	String	Any valid string	None
VALUE	Numeric or string	Values allowed by "FORMAT", "MINIMUM",and "MAXIMUM"	0
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Screen or parent work area upper-left corner

EDITABLE

This attribute determines whether the listbox is editable or not.

- If set to "1", you can type into the edit box at the top
- If set to "0", it displays the function selected from the listbox.

FONT

This attribute determines which font will be used to display text on the KEYPAD dialog. See INTERDEPENDENT ATTRIBUTES for the interactions among the COLUMNS, WIDTH, and FONT attributes. For a description of allowed fonts, see Specifying FONT Attribute Values.

FORMAT

The FORMAT attribute specifies which numeric formatting type is used for the KEYPAD dialog. Allowable types are:

- REAL: the default (-1.79769313486 E +308 to +1.79769313486 E +308)
- SHORT INTEGER: two-byte integer (-32768 to 32767 range)
- LONG INTEGER: four-byte integer (-2147483648 to 2147483647 range)
- BINARY: four-byte integer; two's complement (no "-" allowed)
- OCTAL: four-byte integer; two's complement (no "-" allowed)
- HEX: four-byte integer; two's complement (no "-" allowed)

KEY BACKGROUND

This attribute determines the pen color used for the background of the buttons on the keypad. For a discussion of allowed pens, see Settable Pens Table. This attribute is accepted for compatibility with previous versions, but will not change the appearance of the dialog.

KEY PEN

This attribute determines the pen color used for the characters on the buttons. For a discussion of allowed pens, see Settable Pens Table. This attribute is accepted for compatibility with previous versions, but will not change the appearance of the dialog.

MAXIMUM

The MAXIMUM attribute determines the maximum value the KEYPAD dialog will accept. When changing from one FORMAT to another, KEYPAD will use the lesser of the current maximum value and FORMAT's intrinsic maximum (for positive values).

For example, switching formats from LONG INTEGER to SHORT INTEGER will reduce the maximum KEYPAD value from a four-byte integer to a two-byte integer (see the FORMAT attribute for ranges).

MAXIMUM LENGTH

The MAXIMUM LENGTH attribute specifies the maximum number of characters you can enter into the dialog: "0" means no limit. (Reformatting may generate a longer display than the MAXIMUM LENGTH, however.)

MINIMUM

The MINIMUM attribute determines the minimum value the KEYPAD dialog will accept. When changing from one FORMAT to another, KEYPAD will use the greater of the current minimum value and FORMAT's intrinsic MINIMUM (for negative values).

For example, switching formats from LONG INTEGER to SHORT INTEGER will reduce the minimum KEYPAD value from a four-byte integer to a two-byte integer (see the FORMAT attribute for ranges).

MODIFIED

The MODIFIED attribute indicates the operator has changed the contents of the KEYPAD dialog. When responding to the DONE/RETURN event, check the MODIFIED attribute to see if it has been modified, then set MODIFIED=0 for the next cycle.

VALUE

This attribute reflects the current value of the KEYPAD dialog. It is always greater than or equal to MINIMUM and less than or equal to MAXIMUM and rounded to the nearest INCREMENT.

When using string to set VALUE, the string is verified to be a valid number within range and increment. If the value of VALUE exceeds MAXIMUM or MINIMUM, Error 565 is generated.

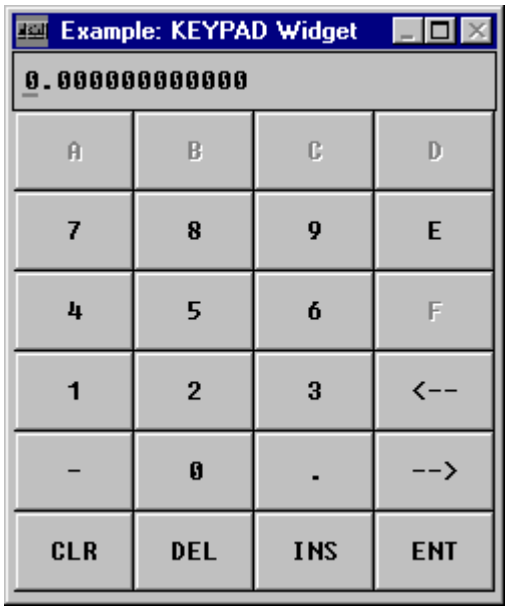
KEYPAD Widget

Provides a graphical keypad presentation from which you can make entries with a mouse, a touchscreen, or a keyboard

---

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See *KEYPAD Widget for a* program that generates a KEYPAD widget and provides a display similar to that shown above.

Attributes

See KEYPAD Widget Attributes for the KEYPAD widget attribute list.

Remarks

The KEYPAD widget is an enhanced version of the NUMBER widget. KEYPAD provides a graphical keypad presentation from which you can make entries with a mouse, touchscreen, or keyboard. The keys on the keypad provide these functions:

- Keys **A** through **F** are used when formatting "HEX"
- Key **E** is used when formatting "REAL"
- Keys that do not apply to the current format are grayed and made non-sensitive. For example, if you choose the "OCTAL" format, the only active numbered keys available will be 0 through 7.
- ENT provides a return key
- \- provides negative numbers or exponents
- CLR clears digits from the text cursor to the end
- DEL deletes the current selection, either all highlighted characters or the character immediately under the cursor
- INS toggles to the insert mode.

Additional attributes for the KEYPAD widget are KEY BACKGROUND and KEY PEN to set key colors, and SHOW EDIT which allows you to turn off the edit box on the KEYPAD. The KEYPAD widget requires that a number be entered in an acceptable format, so you cannot enter an incorrect character.

The KEYPAD widget also provides the ability to translate numbers entered in binary, octal, or hex directly into BASIC numeric variables using the FORMAT attribute. FORMAT can have values of REAL, SHORT INTEGER, LONG INTEGER, BINARY, OCTAL, and HEX.

The following FORMAT LENGTH attribute gives a default field of 4 hexadecimal digits for the user to enter:

```
CONTROL @Num;SET ("FORMAT":"HEX","FORMAT LENGTH":4)
```

You can specify a round-off INCREMENT and (for REAL numbers) the number of fractional digits displayed (using REAL RESOLUTION) and the format for REAL numbers (using REAL NOTATION which can be set to FIXED, SCIENTIFIC, or ENGINEERING).

The KEYPAD widget supports a RETURN event, which occurs when the user presses the Enter key to enter a number and a DONE event, which occurs when the user changes input focus to another window.

Events

Events for the KEYPAD widget are:

- DONE
- INVALID NUMBER
- KEYSTROKE
- RETURN
- SYSTEM MENU

DONE

This event is generated when the widget loses focus and CHECK FOR DONE is set.

INVALID NUMBER

This event is generated when an entry causes the current display to be unable to build a number in the current FORMAT. This event also occurs when DONE or RETURN generates a number outside MINIMUM or MAXIMUM.

KEYSTROKE

Occurs when you enter a character that could change the display.

RETURN

Occurs when you press the **Return** key or the **Enter** key.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

KEYPAD Widget Attributes

KEYPAD Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
CHECK FOR DONE	Numeric	0,1	1
COLUMNS	Numeric	Any non-negative number	20
FONT	String	*Font* [1]	System dependent [2]
FORMAT	String	"REAL", "SHORT INTEGER", "LONG INTEGER", "INTEGER", "BINARY", "OCTAL", "HEX"	"REAL"
FORMAT LENGTH	Numeric	0 to 328	0
HEIGHT	Numeric	Any integer number (of pixels)	Varies
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null

INCREMENT	Numeric	Any non-negative number	0
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
KEY BACKGROUND	Numeric	Any valid BASIC pen number	System dependent [2]
KEY PEN	Numeric	Any valid pen number	System dependent [2]
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MAXIMUM	Numeric	Between or equal to FORMAT's MINIMUM and MAXIMUM	MAXREAL
MAXIMUM LENGTH	Numeric	0 to 328	328
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MINIMUM	Numeric	Between or equal to FORMAT's MINIMUM and MAXIMUM	MINREAL
MODIFIED	Num	0,1	0

	eric		
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PEN	Numeric	0 to 255	System dependent [1]
REAL NOTATION	String	"FIXED", "SCIENTIFIC", "ENGINEERING"	"SCIENTIFIC"
REAL RESOLUTION	Numeric	0 to 16	12
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SHOW EDIT	Numeric	0,1	1
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT	Num	0 to number of items in system menu	0

[5]	eric		
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TEXT LENGTH	Numeric	Any valid integer	Return only
TITLE [5]	String	Any valid string	KEYPAD
USER DATA	String	Any valid string	None
VALUE	Numeric or string	Values allowed by "FORMAT", "MINIMUM", and "MAXIMUM"	0
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Varies
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

HTBasic for Windows

Keyword Conventions

Overview

This topic summarizes conventions for HTBasic Plus keywords. See Keyword Format for the format of a typical HTBasic Plus keyword.

HTBasic Plus Keyword Conventions

Part	Function	Description
Heading		Each keyword heading consists of the keyword name (such as **DISABLE EVENT**) and a summary description of the keyword function.
Legal Usage	Keyboard Executable	"Yes" means that a properly constructed statement containing the keyword can be typed into the keyboard input line and executed by pressing **Enter** or **Return**.
	Programmable	"Yes" means that a properly constructed statement containing the keyword can be placed after a line number (or line label) and stored in a program
	In an IF...THEN...	"Yes" means that a properly constructed statement containing the keyword can be placed after "THEN" in a single-line IF...THEN statement. Keywords that are prohibited in a single-line IF...THEN statement are not necessarily prohibited in a multiple-line IF...THEN structure.
Exam		Shows typical use of the keyword in

ple		statement(s)
State ment s		
Synta x	Syntax Diagram Charact ers	All characters enclosed by a round box (such as *event name*) and single characters shown in circular envelopes (such as ,) must be entered as shown.
	Syntax Diagram Paramet ers	You must enter appropriate values for the named parameters (such as event name) enclosed by rectangular envelopes. See the table following the syntax diagram or the Glossary for a description of the parameters and ranges of allowed values.
	Syntax Diagram Stateme nt Lines	Statement elements are connected by lines. Each line can be followed in only one direction, as indicated by the arrow at the end of the line. Any combination of statement elements that can be generated by following the lines in the proper direction is syntactically correct.
	Syntax Diagram Optional Element s	An element is optional if there is a path around it. Optional elements usually have default values. See the table following the diagram for default values to be used when an optional item is not included in a statement.
	Adding Comme nts	Comments may be added to any valid line. A comment is created by placing an exclamation point after a statement or after a line number or line label. The text

following the exclamation point may contain any
characters in any order.

```
100  PRINT "Hello"      ! This is a comment.
110  ! This is also a comment.
```

Keywords and Spaces

Syntax diagrams do not necessarily deal with the proper use of spaces (ASCII blanks). In general, whenever you are traversing a line any number of spaces may be entered. If two envelopes are touching, it indicates no spaces are allowed between the two items. However, this convention may not be possible in drawings with optional paths.

The computer uses spaces as well as required punctuation to distinguish the boundaries between various keywords, names, and other items. In general, at least one space is required between a keyword and a name if they are not separated by other punctuation.

Spaces cannot be placed in the middle of keywords or other reserved groupings of symbols. Also, keywords are recognized whether they are typed in uppercase or lowercase. To use the letters of a keyword as a name, the name must contain some mixture of uppercase and lowercase letters.

Keyboard

In HTBasic for Windows documentation, specific keys

	Conventions	are mentioned. Because many key labels are different on each keyboard, your exact key labels may not be used in an example. For example, Enter and Return normally have the same meaning, but only one of them appears on a keyboard.
Description		Provides a description of the keyword functions.

HTBasic for Windows

HTBasic Plus Keywords Format

Overview

This topic shows the format used for HTBasic Plus keywords. See Keywords Conventions for descriptions of each part of this format.

HTBasic Plus Keywords Format

DISABLE EVENT (Heading)

Prevents HTBasic for Windows from branching upon receipt of a specified event

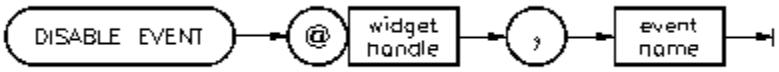
---

Legal Usage	Keyboard Executable	Yes
	Programmable	Yes
	In an IF...THEN...	Yes

Example Statements

```
DISABLE EVENT @Myscrollbar, "Changed"
DISABLE EVENT @String1, "KEYSTROKE"
```

Syntax



Item	Description	Range
event name	name of an event this widget can disable	depends on the widget
widget handle	name identifying a widget	any valid name

Description

Use the DISABLE EVENT keyword to temporarily suspend the effects an event would have on your program...

COLUMNS

This attribute specifies the number of columns (number of character cells) that will appear in the LABEL widget. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact.

FONT

The value is the font that will be used for the text in the LABEL widget. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact. For a description of allowed fonts, see Specifying FONT Attribute Values.

JUSTIFICATION

The value of VALUE (the LABEL string) will be positioned in the widget according to the value you enter here.

PEN

The value of the VALUE attribute will be painted using this pen color. For a discussion of allowed pens, see Settable Pens Table.

ROWS

The number of rows specifies the number of character cells that fit within the LABEL HEIGHT. You specify the number of rows that you want to appear in the LABEL widget. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact.

VALUE

The string numeric or complex value you enter for this attribute will be displayed in the LABEL widget.

WORD WRAP

Use this attribute to specify the end-of-line behavior in the LABEL widget. If the value is 1, the lines will wrap when the line length exceeds the window size. Lines will wrap on a white space character only (space, tab, return, line feed). If the value is 0 (the default), the lines will extend beyond the window size.

HTBasic for Windows

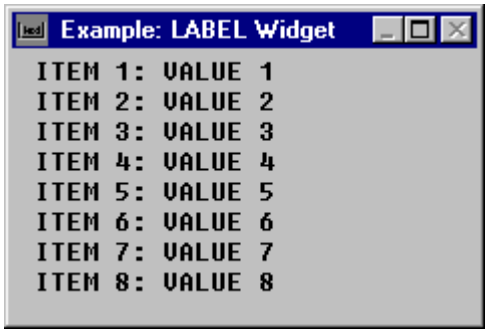
LABEL Widget

LABEL Widget

Displays a string of text

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See *LABEL Widget* for a program shows how to build a table of text items using the LABEL widget. In this program, you create a multi-line LABEL widget and then set its VALUE attribute to a string with embedded line feed characters (using CHR$(10)) to separate it into multiple lines.

NOTE

See the following programs for other examples using the LABEL widget:

Calculator
Changing the Font
Engine Monitor - Child
Engine Monitor - Level-0
Engine Monitor - Panel
Environmental Chamber
Hammer Game

Ice Cream Sundae
Passwords
Rock-Paper-Scissors Game
SLIDER Test
Tic-Tac-Toe
TOGGLEBUTTON Widget
Wing Stress/Vibration Analysis

Attributes

See LABEL Widget Attributes for the LABEL widget attribute list.

Remarks

The LABEL widget allows you to specify the justification (LEFT, RIGHT, etc.) of the information in the LABEL. You can also give it a numeric variable or value, instead of a string. For example:

```
ASSIGN @Lbl TO WIDGET "LABEL"; SET ("VALUE":"Hi There")
```

Many widgets have a LABEL attribute that allows you to specify some describing text, but the LABEL widget itself uses a VALUE attribute to display its information. The LABEL widget has no LABEL attribute. The LABEL widget can be placed as the label of another widget.

Events

The event for the LABEL widget is:

- SYSTEM MENU

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

LABEL Widget Attributes

LABEL Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
COLUMNS	Numeric	Any positive integer	20
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	One row
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title

			bar)
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
ROWS	Numeric	Any positive integer	1
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeri	0 to number of items in system menu	0

	c		
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TITLE [5]	String	Any valid string	LABEL
USER DATA	String	Any valid string	None
VALUE	String, Numeric, Complex	Any	The null string
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	20 columns
WORD WRAP	Numeric	0,1	0
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of Font and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

ALARM RANGES

This attribute specifies which ranges of the LIMITS bar, if any, will cause an alarm to be generated when VALUE enters that range. For example, setting this attribute to "OUTSIDE,INSIDE" (or "INSIDE,OUTSIDE") will cause an alarm to be generated any time the VALUE attribute is set. What actually occurs when an alarm is triggered depends on the value of the ALARM TYPE attribute.

VALUE is INSIDE if it is greater than or equal to, low, and less than or equal to, high. VALUE is OUTSIDE if it is less than low, and greater than high. "I" and "O" can be substituted for "INSIDE" and "OUTSIDE". Setting the null string turns off the ALARM RANGES attribute.

ALARM TYPE

You can set this attribute to one of two values: "BEEP" or "EVENT". If you set it to "BEEP", a beep will be generated any time the VALUE attribute is set to a value which lies in one of the ranges specified in the ALARM RANGES attribute. If you set it to "EVENT", an ALARM event will be generated instead.

AUTOSCALE

If the value is 1, when VALUE exceeds MAXIMUM or is lower than MINIMUM, MAXIMUM and/or MINIMUM will be adjusted on the display to keep the value between MINIMUM and MAXIMUM.

BAR WIDTH

This attribute specifies the width of the limits bar. The extra space between BAR WIDTH and HEIGHT for ORIENTATION: HORIZONTAL, WIDTH for ORIENTATION:RIGHT will be equally allocated as background color on either side of the limits bar.

FONT

This attribute specifies the font to be used for the labels, limits, and values.
For a description of allowed fonts, see Specifying FONT Attribute Values.

HIGH LIMIT

If this attribute is set, the portion of the LIMITS bar above this limit will be painted using the pen color specified by the OUTSIDE PEN attribute. On a monochrome monitor, a line is drawn at the HIGH LIMIT.

To have HIGH LIMIT appear on the LIMITS bar, HIGH LIMIT must be less than or equal to MAXIMUM, greater than or equal to MINIMUM, and greater than or equal to LOW LIMIT. A VALUE equal to HIGH LIMIT is inside the limits boundary.

A VALUE greater than HIGH LIMIT is outside the limits boundary. HIGH LIMIT, LOW LIMIT, ALARM RANGES, and ALARM TYPE can be used to specify the alarm behavior of the LIMITS bar.

The value of HIGH LIMIT will be displayed with only one digit to the right of the decimal point. For example, if you set the value of HIGH LIMIT to 123.4567, the display will show 123.5. If you set the value of HIGH LIMIT to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of HIGH LIMIT as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of HIGH LIMIT to the string "0.000237" or "4.167E-4" in the ASSIGN or CONTROL statement.

INSIDE PEN

This attribute sets the pen color of the inside portion of the bar (the portion between the limits). For a discussion of allowed pens, see Settable Pens Table.

LIMITS BACKGROUND

This attribute sets the background color of the labels displaying the limits. For a discussion of allowed pens, see Settable Pens Table.

LIMITS PEN

This attribute sets the color of the text of the limits labels.

LIMITS WIDTH

This attribute sets the width of the limits labels. For a discussion of allowed pens, see Settable Pens Table.

LOGARITHMIC

If the value of this attribute is 1, the LIMITS bar operates in a logarithmic mode. If the value is 0, the LIMITS bar operates in a linear mode.

LOW LIMIT

This attribute sets the lower limit of the LIMITS bar. Below this limit, the LIMITS bar will be painted using the pen color specified by the OUTSIDE PEN attribute. On a monochrome monitor, a line is drawn at the LOW LIMIT.

To have LOW LIMIT appear on the bar, LOW LIMIT must be greater than or equal to MINIMUM, less than or equal to MAXIMUM, and less than or equal to HIGH LIMIT. A VALUE equal to LOW LIMIT is inside the range. A value less than LOW LIMIT is outside the range.

HIGH LIMIT, LOW LIMIT, ALARM RANGES, and ALARM TYPE can be used to specify the alarm behavior of the LIMITS. The value of LOW LIMIT will be displayed with only one digit to the right of the decimal point. For example, if you set the value of LOW LIMIT to 123.4567, the display will show 123.5. If you set the value of LOW LIMIT to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of LOW LIMIT as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of LOW LIMIT to the string "0.000237" or "4.167E-4" in the ASSIGN or CONTROL statement.

MARKER PEN

This attribute sets the color of the VALUE marker on the LIMITS bar. For a discussion of allowed pens, see Settable Pens Table.

MARKER WIDTH

The marker displays the relative position of VALUE on the LIMITS bar. This attribute sets the thickness of that marker. You can remove the MARKER bar by setting a MARKER WIDTH of "0".

MAXIMUM

The value of this attribute specifies the upper bound on the displayed value of the VALUE attribute. When the value of VALUE equals the value of MAXIMUM, the BAR will reach the top or right (depending on the ORIENTATION) of the widget. MAXIMUM must be greater than MINIMUM.

The value of MAXIMUM will be displayed with only one digit to the right of the decimal point. For example, if you set the value of MAXIMUM to 123.4567, the display will show 123.5. If you set the value of MAXIMUM to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of MAXIMUM as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of MAXIMUM to the string "0.000237" or "4.167E-4" in the ASSIGN or CONTROL statement.

MINIMUM

The value of this attribute specifies the lower bound on the displayed value of the VALUE attribute. When the value of VALUE equals the value of MINIMUM, the BAR will reach the bottom or left (depending on the ORIENTATION) of the widget. MINIMUM must be less than MAXIMUM.

The value of MINIMUM will be displayed with only one digit to the right of the decimal point. For example, if you set the value of MINIMUM to 123.4567, the display will show 123.5. If you set the value of MINIMUM to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of MINIMUM as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of MINIMUM to the string "0.000237" or "4.167E-4" in the ASSIGN or CONTROL statement.

ORIENTATION

This attribute determines the direction or layout of the BAR.

ORIENTATION Attribute Values

Allowed Value	Description
"VERTICAL", "UP"	The bar will be oriented vertically and the VALUE attribute will increase in the up direction
"HORIZONTAL" "RIGHT"	The bar will be oriented horizontally and the VALUE attribute will increase toward the right

OUTSIDE PEN

This attribute sets the pen color of the areas outside the LOW and HIGH LIMITs on the LIMITS bar. For a discussion of allowed pens, see Settable Pens Table.

VALUE

This attribute specifies the current value of the bar. If VALUE exceeds MAXIMUM or MINIMUM, the value that is displayed will be "clipped" by MAXIMUM or MINIMUM. However, the actual value of VALUE will not be clipped.

The value of VALUE will be displayed with only one digit to the right of the decimal point. For example, if you set the value of VALUE to 123.4567, the display will show 123.5. If you set the value of VALUE to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of VALUE as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of VALUE to the string "0.000237" or "4.167E-4" in the ASSIGN or CONTROL statement.

VALUE BACKGROUND

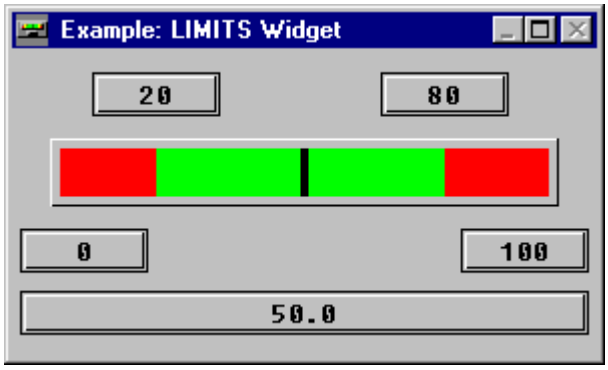
This attribute sets the background color of the labels displaying the value. For a discussion of allowed pens, see Settable Pens Table.

LIMITS Widget

Graphically displays numeric values

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See *LIMITS Widget* for a program that creates a LIMITS widget with a display similar to that shown above.

NOTE

See the following program for another example using the LIMITS widget.

Bar Meter Limits

Attributes

See LIMITS Widget Attributes for the LIMITS widget attribute list.

Remarks

The LIMITS widget displays a range of values as a marker moves through the range. The LIMITS widget is another form of the BAR or METER widget. However, the LIMITS

widget differs from the BAR and METER widgets in that it does not support LOW PEN/ MIDDLE PEN/HIGH PEN. Instead, it supports only OUTSIDE PEN (the color outside the limits)and INSIDE PEN (the color inside the limits). You cannot set the upper and lower color bounds to different colors.

If your computer has a monochrome monitor, the LIMITS widget will handle the situation automatically. There are no attributes related to color to set for this special case. If you have a monochrome monitor, the LIMITS background is shaded light and the value marker and limit lines are black.

Events

Events for the LIMITS widget are:

- ALARM
- SYSTEM MENU

ALARM

This event is generated when the VALUE attribute is set to a value within one of the ranges named in the ALARM RANGES attribute and ALARM TYPE is "EVENT". Redraw activities, such as changing the ORIENTATION attribute value, will not generate an ALARM event.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

LIMITS Widget Attributes

LIMITS Widget Attributes

Attribute	Type	Allowed Values	Default
ALARM RANGES	String	"OUTSIDE" ("O"), "INSIDE" ("I"), "OUTSIDE,INSIDE" ("I, O")	The null string
ALARM TYPE	String	"BEEP", "EVENT"	"BEEP"
AUTOSCALE	Numeric	0,1	1
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BAR WIDTH	Numeric	Any positive integer	30
BORDER [3]	Numeric	0,1	1
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	180 pixels
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
HIGH LIMIT	Numeric or	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	70

	String		
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
LIMITS BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
LIMITS PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
LIMITS WIDTH	Numeric	Any integer number (of pixels)	300
LOGARITHMIC	Numeric	0, 1	0 (linear)
LOW LIMIT	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	30
MARKER PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
MARKER WIDTH	Numeric	Any integer number (of pixels)	5
MAXIMIZABLE [5]	Num	0,1	1 (maximizable, button

	eric		appears in title bar)
MAXIMUM	Numeric or String	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	100
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MINIMUM	Numeric or String	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	0
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
ORIENTATION	String	"UP", "VERTICAL", "RIGHT", "HORIZONTAL"	"HORIZONTAL" (max value at right)
OUTSIDE PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)

STACKING ORDER	Numeric	0 to the number of siblings [4]	0
SYSTEM MENU [5]	String or string array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TITLE [5]	String	Any valid string	LIMITS
USER DATA	String	Any valid string	None
VALUE	Numeric or String	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	50
VALUE BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
VALUE PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1

WIDTH	Numeric	Any integer number (of pixels)	300 pixels
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

VALUE PEN

This attribute sets the color of the text of the value labels.

COLUMNS

This attribute specifies the number of columns (number of character cells) that will appear in the LIST widget. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact.

FONT

This attribute specifies the font to be used for the text in the LIST widget. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact. For a description of allowed fonts, see Specifying FONT Attribute Values.

ITEM COUNT

Return-Only Attribute. ITEM COUNT returns the number of items in the listbox.

ITEMS

This array is used to populate the LIST with items. Each string in the array becomes a separate item (line) in the LIST widget.

MULTISELECT

Use this attribute to enable the selection of either one item or multiple items from a list.

- If you set MULTISELECT to 0, the user can select only one item from the list (either with the mouse or with keyboard selection).
- If you set MULTISELECT to 1, the user can select multiple items, contiguous or not, from the list. You can select or de-select items from the list.

PEN

The list items will be painted using this pen color. For a discussion of allowed pens, see Settable Pens Table.

ROWS

The number of rows specifies the number of character cells that fit within the LIST HEIGHT. You specify the number of rows that you want to appear in the LIST widget. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact.

SELECTION

If the value of the MULTISELECT attribute is 0, the variable associated with SELECTION must be a single numeric variable. That variable specifies the index of the element in the ITEMS array that is selected. A value of "-1" indicates no selection.

If the value of the MULTISELECT attribute is "1", the variable associated with SELECTION must be a numeric array. That array must be at least as large as the ITEMS array. Each element of the SELECTION array will return either a "1" (indicating that the corresponding items element is selected) or a "0" (indicating that the corresponding items element is not selected).

Because all arrays are treated as a single-dimension, BASE 0 array, the SELECTION attribute used in LIST returns an integer that specifies an offset, not necessarily the actual subscript, of the selected element in the ITEMS array.

SENSITIVE

This attribute provides you with the capability to keep the LIST widget visible, but make it unresponsive to user input.

- If the value of SENSITIVE is 1, the widget is responsive to user input.
- If the value of SENSITIVE is 0, the widget will not respond to user input, and will appear "grayed out" to indicate its unresponsive status.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

Arrow-Key Behavior When The LIST Widget Has The Input Focus

Vertical Arrow-Key Behavior	Horizontal Arrow-Key Behavior
scroll	do nothing

TOP ITEM

You can use TOP ITEM to set or get the item that appears as the top item in the listbox. That is, you can use TOP ITEM to "place" a window (that is, the listbox) over a long list of items. You set the value of TOP ITEM to the index in the ITEMS array of the item that you want on the top of the displayed list.

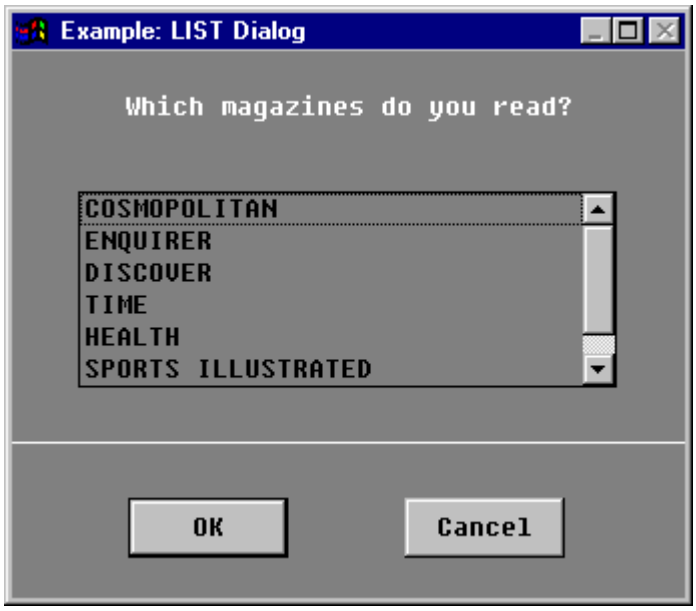
HTBasic for Windows

LIST Dialog

LIST Dialog

Prompts the operator to select a specific item or set of items
from a list of items

Example Image



Example Program

See LIST Dialog for an example program that produces a display similiar to that shown above.

NOTE

See the following program for another example using the LIST dialog:

Dialogs Tests

Attributes

See LIST Dialog Attributes for the LIST dialog attribute list.

Remarks

None.

HTBasic for Windows

LIST Dialog Attributes

LIST Dialog Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
COLUMNS	Numeric	Any positive integer	20
DEFAULT BUTTON	Numeric	Any valid index into the DIALOG BUTTONS array	0
DIALOG BUTTONS	String array	Any valid string array	A two-element array that contains OK and Cancel
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
ITEM COUNT	Numeric	0 to 32767	0
ITEMS	String array	Any	A zero-element array
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE	Numeric	0,1	0 (not minimizable, button does not appear in title bar)

MOVABLE	Numeric	0,1	1 (movable, click on title bar and drag)
MULTISELECT	Numeric	0,1	0 (one ITEM only)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE	Numeric	0,1	1 (resizable, special border appears around the dialog)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
ROWS	Numeric	Any positive integer	3
SELECTION	Numeric or Numeric array	Single Numeric: value is any valid index into the ITEMS array, or -1 Numeric array: the array contains a 1 in each element that represents a user-selected ITEM, else 0	-1
SENSITIVE	Numeric	0,1	1 (sensitive)
TITLE	String	Any valid string	LIST
TOP ITEM	Numeric	Any valid index into ITEMS array	0 or 1
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version

WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Screen or parent work area upper-left corner

COLUMNS

This attribute specifies the number of columns (number of character cells) that will appear in the LIST dialog. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact.

FONT

This attribute specifies the font to be used for the text in the LIST dialog. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact.For a description of allowed fonts, see Specifying FONT Attribute Values.

ITEMS

This array is used to populate the LIST dialog with items. Each string in the array becomes a separate item (line) in the LIST dialog.

ROWS

The number of rows specifies the number of character cells that fit within the LIST dialog's HEIGHT. You specify the number of rows that you want to appear in the LIST dialog. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact.

SENSITIVE

This attribute provides you with the capability to keep the LIST dialog visible, but make it unresponsive to user input.

- If the value of SENSITIVE is 1, the dialog is responsive to user input.
- If the value of SENSITIVE is 0, the dialog will not respond to user input, and will appear "grayed out" to indicate its unresponsive status.

HTBasic for Windows

LIST Widget

LIST Widget

Provides a scrollable list of textual items

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image

Example Program

See *LIST Widget* for a program that provides a LIST widget similar to the display shown above.

NOTE

See the following program for another example using the LIST widget.

Ice Cream Sundae

Attributes

See LIST Widget Attributes for the LIST widget attribute list.

Remarks

The LIST widget is similar to a COMBO widget except the LIST widget does not allow you to enter a string, but only to select from a list of items. A vertical scrollbar always appears in this widget. There is no horizontal scrollbar.

Events

Events for the LIST widget are:

- SELECTION
- SYSTEM

SELECTION

This event will be generated any time the user changes selections in the list. If the user clicks the mouse anywhere that is not a valid list item entry, a SELECTION event will be generated and -1 is returned for the SELECTION attribute.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

LIST Widget Attributes

LIST Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
COLUMNS	Numeric	Any positive integer	10
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
ITEM COUNT	Numeric	0 to 32767	0
ITEMS	String	Any	A zero-element array

	array		
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
MULTISELECT	Numeric	0,1	0 (one ITEM only)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
ROWS	Numeric	Any positive integer	5
SELECTION	Numeric or N	Single Numeric: value is any valid index into the ITEMS array, or -1 Numeric array: the array contains a 1 in	-1

	numeric array	each element that represents a user-selected ITEM, else 0	
SENSITIVE	Numeric	0,1	1 (sensitive)
STACKING ORDER	Numeric	0 to the number of siblings [4]	0
SYSTEM MENU [5]	String or string array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TITLE [5]	String	Any valid string	LIST
TOP ITEM	Numeric	Any valid index into ITEMS array	0
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
VISIBLE	N	0,1	1

	u m eri c		
WIDTH	N u m eri c	Any integer number (of pixels)	Autosizing
X	N u m eri c	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	N u m eri c	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

HTBasic for Windows

List of Dialogs

Overview

This topic summarizes HTBasic Plus dialogs. Click the dialog name for a description of the dialog.

- See Dialogs Attributes for information on HTBasic Plus dialogs attributes.
- See Dialogs Format for a HTBasic Plus dialogs format and structure.

List of Dialogs

Dialog	Function
COMBO	Allows entry by text input or from list
ERROR	Displays error message, user responds with mouse button
FILE	Allows user to select a file from a directory
INFORMATION	Displays information, user responds with mouse button
KEYPAD	Similar to NUMBER, but uses a calculator keypad for entry
LIST	Allows user to select from a list of items
NUMBER	Allows input of numeric data
QUESTION	Displays question, user responds with mouse button
STRING	Prompts user for text string
WARNING	Displays warning, user responds with mouse button

HTBasic for Windows

List of Keywords

Overview

This topic lists HTBasic Plus keywords.

- See Keyword Conventions for HTBasic Plus programming conventions
- See Keyword Format for descriptions of HTBasic Plus keywords

Click the keyword name for a description of the keyword.

List of Keywords

Keyword	Description
ASSIGN	Creates/destroys widgets
CONTROL	Sets value(s) of specified attribute(s) for widget
DIALOG	Generates a dialog of the specified type
DISABLE EVENT	Prevents branching upon receipt of specified event
ENABLE EVENT	Enables branching upon receipt of specified event
OFF EVENT	
ON EVENT	Cancels event branches defined by ON EVENT
STATUS	Defines event branch taken after event generated
WAIT FOR EVENT	Returns value of each listed attribute for widget
	Suspends program execution until an event occurs

HTBasic for Windows

List of Widgets

Overview

This topic lists HTBasic Plus widgets alphabetically and by functional group.

- Click the widget name for a description of the widget.
- See List of Widgets - by Group for a list of widgets by functional group.

List of Widgets - Alphabetical

Widget	Description
BAR	Graphically displays a numeric value with a rectangular bar
BARS	Graphically displays a bank of numeric values
BITMAP	Displays and creates pixmap files
CASCADE MENU	Menu of selections within PULLDOWN MENU or other CASCADE MENU
CLOCK	Graphically displays time
COMBO	Combination of string and list widgets that is used to enter a string
FILE	Allows selection of disk volume, directory, or file
HPGL VIEW	Displays HPGL graphics files
KEYPAD	Provides a method for entering numbers without using a keyboard
LABEL	Displays a string of text or can be placed as the label of another widget
LIMITS	
LIST	Graphically displays numeric values
MENU BUTTON	Scrollable list of textual items
MENU SEPARATOR	Appears in PULLDOWN MENU or CASCADE MENU after menu is activated
MENU TOGGLE	Horizontal line between items in PULLDOWN MENU or CASCADE MENU
METER	Menu button that toggles between states
NUMBER	Graphically displays numeric values with a needle
PANEL	Used to enter and format numbers with specified limits and increments
PRINTER	Used as a "container" or "parent" widget to help group other widgets
PULLDOWN MENU	Used to display blocks of test (with the option to scroll the text)
PUSHBUTTON	Provides a menu of selections from which the user can choose
RADIOBUTTON	Standalone button that can be placed on the screen or in a PANEL
	Bank of buttons in which only one button at a time can be depressed
	Used to input a relative position

SCROLLBAR	Line used to visually separate areas of a PANEL
SEPARATOR	Slider pad that moves in a trough from minimum value to maximum value
SLIDER	
STRING	Used to enter or display an arbitrary string or array of strings of text
STRIPCHART	Displays data in a two-dimensional scrolling graph
SYSTEM	Retrieves and modifies screens built with Screen Builder application
TOGGLEBUTTON	Used to input or display binary data
XYGRAPH	Used to plot data on an X-Y coordinate plane

HTBasic for Windows

List of Widgets - by Group

Group	Widget	Description
Text Display	LABEL	Allows labels to be placed on user interface
	LIST	Allows a selection from a list of items
	PRINTER	Allows output of multiple lines of text
Data Output	BAR	Single-bar bargraph
	BARS	Multiple-bar bargraph
	BITMAP	Allows display of XWD or BMP bitmap files
	CLOCK	Displays time information
	HPGL VIEW	Allows viewing of HPGL files
	LIMITS	Similar to METER widget
	METER	An arc-shaped scale with a sweeping needle
	STRIPCHART	Allows drawing of stripcharts
	XY GRAPH	Allows drawing of XY graphs
Data Input	COMBO	Allows entry or selection from a list of items
	FILE	Allows selection of disk volume, directory, or file
	KEYPAD	Similar to NUMBER widget
	NUMBER	Allows input of numeric data
	PUSHBUTTON	Provides a "pushbutton"
	RADIOBUTTON	A button that acts like a car radio button
	SCROLLBAR	Simplified version of SLIDER
	SLIDER	Allows numeric input with mouse
	STRING	Allows a string to be entered from user interface
	TOGGLEBUTTON	A button that can be toggled between two states
Menu Creation	CASCADE MENU	Used for cascading multiple pulldown menus
	MENU BUTTON	Menu selection in pulldown or cascade menus
	MENU SEPARATOR	Draws line in menus as item separator
	MENU TOGGLE	Toggles between two states
	PULLDOWN	Used for building pulldown menu systems

Special	MENU	
	PANEL	Provides a "panel" in which to place other widgets
	SEPARATOR	Draws a line to separate regions on PANEL
	SYSTEM	Used to build user interfaces and widget array

HTBasic for Windows

Logical Pens Table

This table shows the logical pen numbers and associated physical pen numbers for HTBasic for Windows. See Physical Pens Table for the actual colors of the physical pen numbers listed. See Settable Pens Table for settable pens for HTBasic for Windows.
*Of the logical pens listed in the Logical Pens table, only those pens with **bold blue** entries are referenced in HTBasic Plus. The other pens remain for compatibility with previous versions of HP BASIC Plus.*

Pen	Pen Name	Physical Pen for System Containing ...			
		2 colors	**3-63 colors**	**64-255 colors**	**256 colors**
0	ActiveBorder	0	14	14	30
1	ActiveTitleBar	1	14	14	30
2	ActiveTBarForeground	0	1	1	17
3	**AppBackground**	**0**	**9**	**9**	**25**
4	AppBorder	0	14	14	30
5	AppTitleBar	0	14	14	30
6	AppTBarForeground	1	1	1	17
7	ButtonBackground	0	11	11	27
8	ButtonForeground	1	0	0	16
9	**DialogBoxBackground**	**0**	**8**	**8**	**24**
10	DialogBoxBorder	1	14	14	30
11	**DialogBoxForeground**	**1**	**1**	**1**	**17**
12	DialogBoxTitleBar	0	13	13	29
1	DialogBoxTBarForegrou	1	1	1	17

3	nd				
14	ErrorTitleBar	0	2	2	18
15	**GrayForeground**	**0**	**15**	**15**	**31**
16	Highlight	1	13	13	29
17	HighlightForeground	0	3	3	19
18	HyperText	1	6	6	22
19	InactiveBorder	0	11	11	27
20	InactiveTitleBar	0	11	11	27
21	Inactive TBarForeground	1	0	0	16
22	ITGDisplayTraceBlue	1	5	5	21
23	ITGDisplayTraceGreen	1	4	4	20
24	ITGPushButton	0	12	12	28
25	ITGResetButton	0	10	10	26
26	**ListBoxForeground**	**1**	**0**	**0**	**16**
27	**ListBoxBackground**	**0**	**15**	**15**	**31**
28	MenuBar	0	11	11	27
29	MenuForeground	1	0	0	16
30	PopupWindowBackground	0	8	8	24
31	PopupWindowFrame	1	0	0	16
32	PopupWindowForeground	1	1	1	17
33	**Scrollbars**	**0**	**11**	**11**	**27**
34	**WindowBackground**	**0**	**1**	**1**	**17**
35	WindowFrame	1	0	0	16
36	**WindowForeground**	**1**	**0**	**0**	**16**
37	Workspace	0	15	15	31

38	WorkspaceBorder	0	14	14	30
39	WorkspaceTitleBar	0	14	14	30
40	WorkspaceTBarForeground	1	0	0	16
41	**WidgetGreen**	**1**	**4**	**4**	**20**
42	**WidgetYellow**	**1**	**3**	**3**	**19**
43	**WidgetRed**	**1**	**2**	**2**	**18**
44	**WidgetWhite**	**1**	**1**	**1**	**17**
45	**GraphBackground**	**0**	**13**	**13**	**29**
46	**GraphTrace1**	**1**	**3**	**3**	**19**
47	**GraphTrace2**	**1**	**5**	**5**	**21**
48	**GraphTrace3**	**1**	**7**	**7**	**23**
49	**GraphTrace4**	**1**	**4**	**4**	**20**
50	**GraphTrace5**	**1**	**2**	**2**	**18**
51	**GraphTrace6**	**1**	**14**	**14**	**30**
52	**GraphTrace7**	**1**	**11**	**11**	**27**
53	**GraphTrace8**	**1**	**1**	**1**	**17**
54	RMBBackground	0	0	0	0
55	AppBackgroundCmp	1	6	54	230
56	WindowBackgroundCmp	1	14	62	238
57	PopupBackgroundCmp	1	7	55	231
58	DialogBoxBackgroundCmp	1	7	55	231

MAXIMIZABLE

Dialogs and Level-0 Widgets Only. If the value of this attribute is set to 1, a button will be drawn at the far-right in the title bar of the widget When the user clicks on that button, the dialog or widget will expand to the maximum size of the screen. The dialog or widget will return to its original size if the user clicks on that button again.

If the value of this attribute is set to 0, no button appears in the title bar and the dialog or widget is not maximizable. You cannot maximize the dialog or widget from the keyboard (without a mouse interface).

FOCUS

Widgets Only. A value of 0 gives focus to the specified widget. A value of 1 moves focus from the Widget to the Basic window. Using the FOCUS attribute it is possible to change focus to a particular widget or to the Basic window.

LABEL

The string you enter for this attribute will be entered on the MENU BUTTON.

SENSITIVE

This attribute provides you with the capability to keep the MENU BUTTON widget visible, but make it unresponsive to user input.

- If the value of SENSITIVE is 1, the widget is responsive to user input.
- If the value of SENSITIVE is 0, the widget will not respond to user input, and will appear "grayed out" to indicate its unresponsive status.

USER DATA

The contents of this attribute are left to the programmer and are never changed by HTBasic for Windows. It is simply a way to associate program-specific data with a specific widget or dialog.

HTBasic for Windows

MENU BUTTON Widget

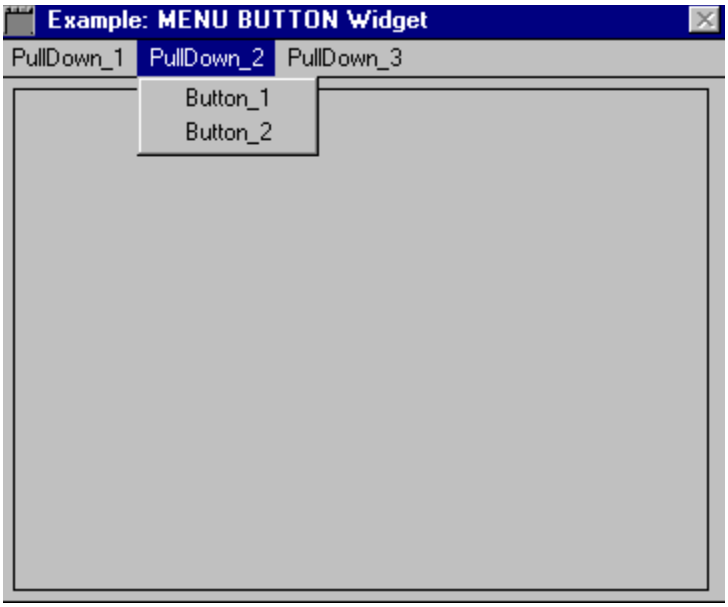
MENU BUTTON Widget

Appears in a PULLDOWN MENU or a CASCADE MENU after the operator activates the menu

---

Legal Usage	Level-0 Widget:	No
	Parent to:	None
	Child of:	PULLDOWN MENU, CASCADE MENU

Example Image



Example Program

See the *Menu System* program for an example program that shows how to create a MENU BUTTON widget.

NOTE

See the following programs for other examples using the MENU BUTTON widget:

Environmental Chamber
Function Generator

HPGL VIEW Viewer
Passwords
SLIDER Test
STRING Editor

Attributes

See MENU BUTTON Widget Attributes for for the MENU BUTTON widget attribute list.

Remarks

The MENU BUTTON widget is a widget that appears in a PULLDOWN MENU or a CASCADE MENU after the user activates the menu.

To activate a MENU BUTTON, follow one of these procedures:

- Press the mouse button when the cursor is in the boundary of the PULLDOWN MENU label area and hold. Then release the mouse button when the mouse cursor is within the borders of the MENU BUTTON.

- Press the mouse button when the cursor is in the boundary of the PULLDOWN MENU label area and release. Then click the mouse button when the mouse cursor is within the borders of the MENU BUTTON.

Events

The event for MENU BUTTON is:

/0
ACTIVATED

This event occurs only when the mouse button is pushed and released while the cursor is within the border of the MENU BUTTON.

HTBasic for Windows

MENU BUTTON Widget Attributes

MENU BUTTON Widget Attributes

Attribute	Type	Allowed Values	Default
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
LABEL	String	Any	The null string
SENSITIVE	Numeric	0,1	1
USER_DATA	String	Any	The null string

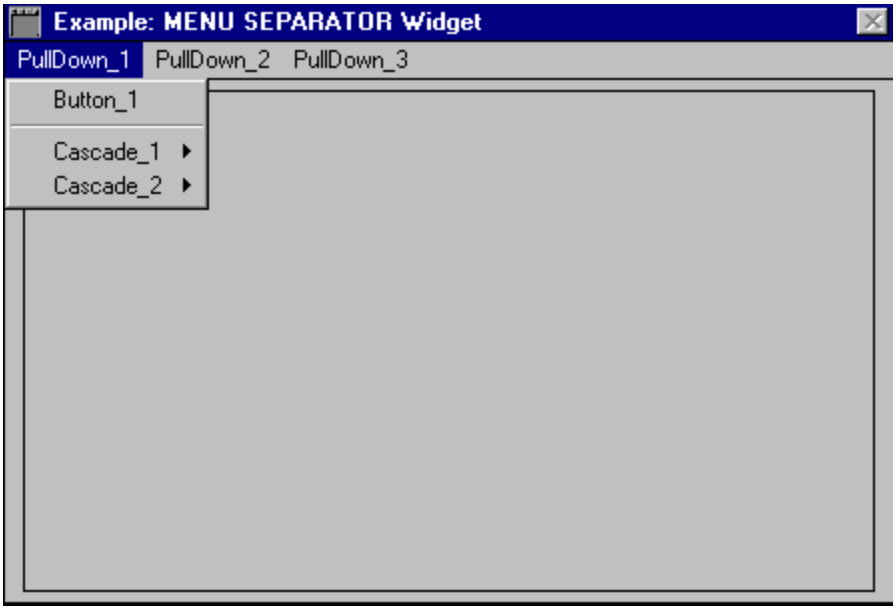
MENU SEPARATOR Widget

MENU SEPARATOR Widget

Provides a horizontal line between the widgets in a PULLDOWN MENU or CASCADE MENU

Legal Usage	Level-0 Widget:	No
	Parent to:	None
	Child of:	PULLDOWN MENU, CASCADE MENU

Example Image



Example Program

See the Menu System program for an example program that shows how to create a MENU SEPARATOR widget.

See the following programs for other examples using the MENU SEPARATOR widget:

HPGL VIEW Viewer
SLIDER Test
STRING Editor

Attributes

See MENU SEPARATOR Widget Attributes for the MENU SEPARATOR widget attribute list.

Remarks

The MENU SEPARATOR widget is a horizontal line between the widgets in a PULLDOWN MENU or a CASCADE MENU.

The only attribute for MENU SEPARATOR is USER DATA. The contents of this attribute are left to the user, and are never changed by HTBasic for Windows.

Events

None.

HTBasic for Windows

MENU SEPARATOR Widget Attributes

MENU SEPARATOR Widget Attributes

Attribute	Type	Allowed Values	Default
USER DATA	String	Any	The null string

LABEL

The value of this attribute is the text that appears to the right of the MENU TOGGLE.

SENSITIVE

This attribute provides you with the capability to keep the LABEL text visible, but make the MENU TOGGLE unresponsive to user input.

- If the value of SENSITIVE is 1, the widget is responsive to user input.
- If the value of SENSITIVE is 0, the widget will not respond to user input, and will appear "grayed out" to indicate its unresponsive status.

VALUE

The value of this attribute specifies the state of the MENU TOGGLE. If the value is 1, the MENU TOGGLE is depressed. If the value is 0, the MENU TOGGLE is released.

HTBasic for Windows

MENU TOGGLE Widget

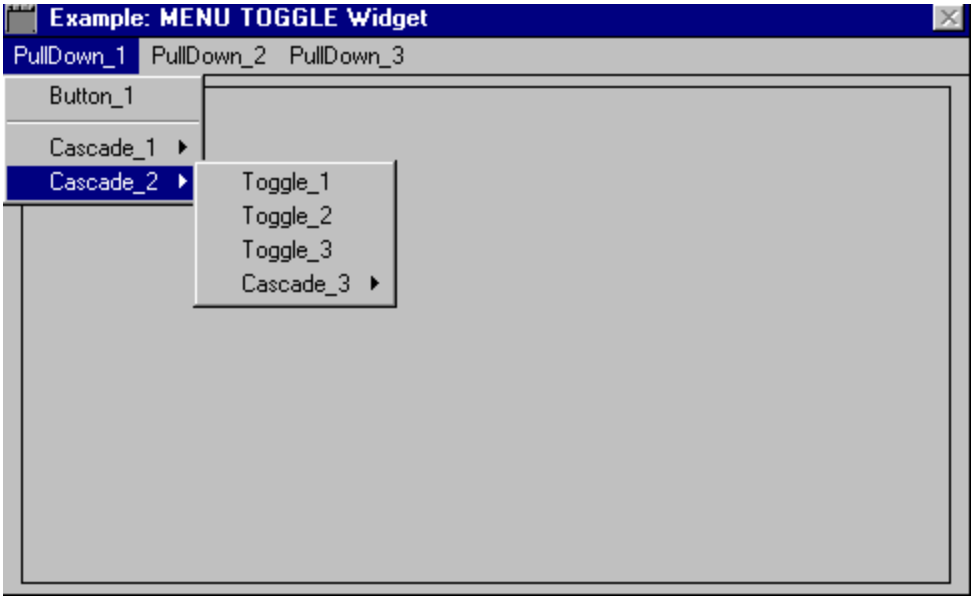
MENU TOGGLE Widget

Appears in a PULLDOWN MENU or a CASCADE MENU after the operator activates the menu

---

Legal Usage	Level-0 Widget:	No
	Parent to:	None
	Child of:	PULLDOWN MENU, CASCADE MENU

Example Image



Example Program

See the *Menu System* program for an example program that shows how to create a MENU TOGGLE widget.

NOTE

See the following program for another example using the MENU TOGGLE widget:

Function Generator

Attributes

See MENU TOGGLE Widget Attributes for the MENU TOGGLE widget attribute list.

Remarks

The MENU TOGGLE widget is a widget that appears in a PULLDOWN MENU or a CASCADE MENU after the user activates the menu. To activate a MENU TOGGLE, follow one of these procedures:

- Press the mouse button when the cursor is in the boundary of the PULLDOWN MENU label area and hold. Then release the mouse button when the mouse cursor is within the borders of the MENU TOGGLE.

- Press the mouse button when the cursor is in the boundary of the PULLDOWN MENU label area and release. Then click the mouse button when the mouse cursor is within the borders of the MENU TOGGLE.

Events

The event for MENU TOGGLE widget is:

- CHANGED

CHANGED

This event is generated when the user clicks on the MENU TOGGLE widget.

HTBasic for Windows

MENU TOGGLE Widget Attributes

MENU TOGGLE Widget Attributes

Attribute	Type	Allowed Values	Default
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
LABEL	String	Any	The null string
SENSITIVE	Numeric	0,1	1(is sensitive)
USER_DATA	String	Any	The null string
VALUE	Numeric	0,1	0

ALARM RANGES

This attribute specifies which ranges of the METER, if any, will cause an alarm to be generated when the value of the VALUE attribute enters those ranges.

For example, setting this attribute to "LOW,HIGH" will cause an alarm to be generated any time the VALUE attribute moves out of the MIDDLE range. What occurs when an alarm is triggered depends on the value of the ALARM TYPE attribute.

ALARM TYPE

You can set this attribute to one of two values: "BEEP" or "EVENT". If you set it to "BEEP", a beep will be generated any time the VALUE attribute is set to a value which lies in one of the ranges specified in the ALARM RANGES attribute. If you set it to "EVENT", an ALARM event will be generated instead.

ARC WIDTH

The value of this attribute sets the width in pixels of the line that draws the arc.

AUTOSCALE

This attribute enables the METER's autoscaling. If the value is 1, when the value of VALUE exceeds MAXIMUM or is lower than MINIMUM, MAXIMUM and/or MINIMUM will be automatically adjusted on the display to keep the value of VALUE between MINIMUM and MAXIMUM.

CENTER

Centers the meter in the middle of the panel. This is especially for sweep angles greater than 180 degrees providing additional flexibility in the meter's appearance.

FONT

This attribute specifies the font to be used for the limits and value. For a description of allowed fonts, see Specifying FONT Attribute Values.

HIGH LIMIT

An indicator appears along the METER's arc at the point specified by this value. For this indicator to appear, HIGH LIMIT must be greater than or equal to MINIMUM, less than or equal to MAXIMUM, and greater than or equal to LOW LIMIT.

HIGH PEN

The value of this attribute specifies the pen color that will be used to draw the arc line between the HIGH LIMIT tick mark and the MAXIMUM value. For a discussion of allowed pens, see Settable Pens Table.

LIMITS BACKGROUND

The value of this attribute specifies the pen color used to paint the background of the limits boxes of the METER. For a discussion of allowed pens, see Settable Pens Table.

LIMITS PEN

This value of this attribute specifies the pen color used to paint the limits characters. For a discussion of allowed pens, see Settable Pens Table.

LOGARITHMIC

If the value of this attribute is 1, the METER operates in a logarithmic mode. If the value is 0, the METER operates in a linear mode.

LOW LIMIT

An indicator appears along the METER's arc at the point specified by this value. To appear, LOW LIMIT must be greater than or equal to MINIMUM, less than or equal to MAXIMUM, and less than or equal to HIGH LIMIT.

LOW PEN

The value of this attribute specifies the pen color that will be used to draw the arc line between the MINIMUM value and the LOW LIMIT tick mark. For a discussion of allowed pens, see Settable Pens Table.

MAXIMUM

This attribute specifies the value that VALUE will have when the needle is at the maximum end of the METER. MAXIMUM must be greater than MINIMUM. The value of MAXIMUM will be displayed with only one digit to the right of the decimal point.

For example, if you set the value of MAXIMUM to 123.4567, the display will show 123.5. If you set the value of MAXIMUM to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of MAXIMUM as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of MAXIMUM to the string 0.000237 or 4.167E-4 in the ASSIGN or CONTROL statement.

METER BACKGROUND

The value of this attribute specifies the pen color used to paint the background of the METER's work area. For a discussion of allowed pens, see Settable Pens Table.

MIDDLE PEN

The value of this attribute specifies the pen color that will be used
to draw the arc line between the LOW LIMIT and HIGH LIMIT tick marks.
For a discussion of allowed pens, see Settable Pens Table.

MINIMUM

This attribute specifies the value that VALUE will have when the needle is at the extreme left or bottom end of the METER. MINIMUM must be less than MAXIMUM. The value of MINIMUM will be displayed with only one digit to the right of the decimal point.

For example, if you set the value of MINIMUM to 123.4567, the display will show 123.5. If you set the value of MINIMUM to 0.0001375, the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of MINIMUM as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of MINIMUM to the string "0.000237" or "4.167E-4" in the ASSIGN or CONTROL statement.

NEEDLE PEN

The value of this attribute specifies the pen color that will be used to paint the needle. For a discussion of allowed pens, see Settable Pens.

NEEDLE WIDTH

The value of this attribute specifies the width in pixels of the needle.

ORIENTATION

This attribute determines the direction or layout of the METER.

ORIENTATION Attribute Values

Allowed Value	Description
"UP", "VERTICAL"	The METER will be oriented vertically, MINIMUM at left, MAXIMUM at right, and the value of VALUE will increase in the clockwise direction.
"DOWN"	The METER will be oriented vertically, MINIMUM at left, MAXIMUM at right, and the value of VALUE will increase in the counterclockwise direction.
"RIGHT", "HORIZONTAL"	The METER will be oriented horizontally, MINIMUM at bottom, MAXIMUM at top, and the value of VALUE will increase in the counterclockwise direction.
"LEFT"	The METER will be oriented horizontally, MINIMUM at bottom, MAXIMUM at top, and the value of VALUE will increase in the clockwise direction.

SHOW LIMITS

This attribute is used to turn on and off the limits boxes on the meter. If its value is 1, all limits boxes are visible. If the value is 0, all limits boxes are invisible.

SHOW VALUE

This attribute is used to turn on and off the value box that appears under (assuming ORIENTATION is set to "UP" or "VERTICAL") the METER needle. If the value is 1, the value box is visible. If the value is 0, the value box is invisible.

SWEEP ANGLE

This attribute specifies the angle subtended by the sweep of the needle from MINIMUM to MAXIMUM along the arc. The value of SWEEP ANGLE can vary from 1 to 359 degrees.

Use this attribute to position the METER within its window. The METER widget uses an autoposition algorithm that positions the "meter movement" based on the specified WIDTH, HEIGHT, and SWEEP ANGLE attribute values. The needle pivot point, needle length, and centering of the arc within the window are positioned automatically.

TICK LENGTH

The value of this attribute specifies the length in pixels of the tick marks that appear along the METER's arc between the LOW LIMIT, MIDDLE, and HIGH LIMIT bands. Set the value of TICK LENGTH to 0 to turn the tick marks completely off (make them invisible).

TICK LOCATION

The value of this attribute determines whether the meter's tick marks will lie INSIDE, OUTSIDE, or CENTERED upon the meter's arc.

TICK PEN

This value of this attribute specifies the pen color used to paint the lines of the tick marks. For a discussion of allowed pens, see Settable Pens Table.

VALUE

This attribute reflects the current reading of the METER. The value of VALUE will be displayed with only one digit to the right of the decimal point.

For example, if you set the value of VALUE to 123.4567, the display will show 123.5. If you set the value of VALUE to 0.0001375,the display will show 1.4E-4.

If you need to display numbers in a different way for your application, you can set the value of VALUE as a string. For example, if you want to display 0.000237 or 4.167E-4, set the value of VALUE to the string 0.000237 or 4.167E-4 in the ASSIGN or CONTROL statement.

VALUE BACKGROUND

This value of this attribute specifies the pen color used to paint the background of the value box. For a discussion of allowed pens, see Settable Pens Table.

VALUE PEN

This value of this attribute specifies the pen color used to paint the value characters. For a discussion of allowed pens, see Settable Pens Table.

HTBasic for Windows

METER Widget

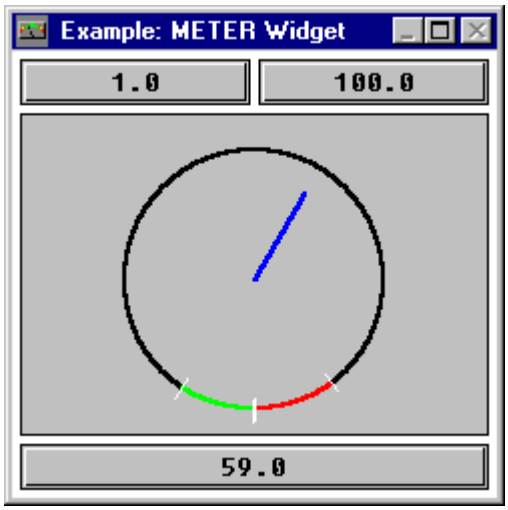
METER Widget

Graphically displays a numeric value with a needle that sweeps along an arc

---

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See METER Widget for a program that creates a METER widget with a display similar to that shown above, except that the needle sweep is 360 degrees.

NOTE

See the following programs for other examples using the METER widget:

Bar Meter Limits
Engine Monitor - Level-0
Engine Monitor - Panel
Environmental Chamber
METER/SLIDER Widgets

Oven Control
PID Controller

Attributes

See METER Widget Attributes for the METER widget attribute list.

Remarks

The METER widget graphically displays a numeric value with a needle that sweeps along an arc.The METER widget programming operation is virtually identical to BAR widget programming. The METER widget displays its MINIMUM limit, MAXIMUM limit, and CURRENT VALUE in boxes. You can set the SWEEP ANGLE to 360 degrees to give this widget a tachometer-like appearance.

An event can be generated on any or all of the three ranges: "LOW", "MIDDLE", or "HIGH". METER widget events are level-triggered, not edge-triggered. An event is generated any time a VALUE is within the designated ALARM RANGE, not only when the VALUE transitions from one range to the other. You must specifically set the ALARM TYPE to event before an event can be generated. Otherwise, only beeps are generated.

Events

Events for the METER widget are:

- ALARM
- SYSTEM MENU

ALARM

This event is generated when the VALUE attribute is set to a value within one of the ranges named in the ALARM RANGES attribute and ALARM TYPE is EVENT. The only time an ALARM event is generated is when the value of VALUE is set to one of the following:

- To or above HIGH LIMIT and the value of ALARM RANGES is "HIGH"
- Between and including LOW LIMIT and HIGH LIMIT and the value of ALARM RANGES is "MIDDLE"
- To or below LOW LIMIT and the value of ALARM RANGES is "LOW"

Redraw activities, such as changing the ORIENTATION attribute value,
will not generate an ALARM event.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

METER Widget Attributes

METER Widget Attributes

Attribute	Type	Allowed Values	Default
ALARM RANGES	String	Any combination of: "LOW", "MIDDLE", "HIGH"	The null string
ALARM TYPE	String	"BEEP", "EVENT"	"BEEP"
ARC WIDTH	Numeric	1 to 10	2 (pixels)
AUTOSCALE	Numeric	0,1	0
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
CENTER	Numeric	0,1	1
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	180 pixels
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null

HIGH LIMIT	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	MAXREAL
HIGH PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
LIMITS BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
LIMITS PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
LOGARITHMIC	Numeric	0, 1	0 (linear)
LOW LIMIT	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	MAXREAL
LOW PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MAXIMUM	Nu	LOGARITHMIC is 0: any real number	100

	meric or string	LOGARITHMIC is 1: any positive real number	
METER BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
MIDDLE PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MINIMUM	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	1
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
NEEDLE PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
NEEDLE WIDTH	Numeric	Non-negative integer	1 (pixel)
ORIENTATION	String	"UP", "VERTICAL", "DOWN", "LEFT", "HORIZONTAL", "RIGHT"	"UP" (max value at top)
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)

RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SHOW LIMITS	Numeric	0,1	1 (show limits in boxes)
SHOW VALUE	Numeric	0,1	1 (show value in a box)
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SWEEP ANGLE	Numeric	Any integer from 1 to 359 (degrees)	60 (degrees)
SYSTEM MENU [5]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TICK LENGTH	Numeric	0 to 100	12 (pixels)
TICK LOCATION	String	"CENTER", "INSIDE", "OUTSIDE"	"CENTER"
TICK PEN	Nu	Any valid BASIC pen number (0-255)	System dependent [2]

	meric		
TITLE [5]	String	Any valid string	METER
USER DATA	String	Any valid string	None
VALUE	Numeric	LOGARITHMIC is 0: any real number LOGARITHMIC is 1: any positive real number	1
VALUE BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
VALUE PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	220 pixels
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

/-b

Dialogs and Level-0 Widgets Only. If the value of this attribute is set to 1, a button will be drawn at the far-right in the title bar of the widget or dialog (left of the maximizable button). When the user clicks on that button, the widget or dialog will be changed to an icon.

The widget or dialog will return to its original size if the user double-clicks on its icon. If the value of this attribute is set to 0, no button appears in the title bar and the widget or dialog is not minimizable. You cannot minimize the widget or dialog from the keyboard (without a mouse interface).

MOVEABLE

D*ialogs and Level-0 Widgets Only*. If the value of this attribute is set to 1, the operator can press-and-hold the left mouse button when the cursor is in the title bar and can drag the dialog or widget to the new position. When the mouse button is released, the dialog or widget is placed at the new position. You cannot move the dialog or widget from the keyboard (without a mouse interface).

HTBasic for Windows

Modifying/Saving CONFIG

You can edit the HTBasic for Windows CONFIG file with the HTBasic editor (use GET <*file-pathname*>), the Notepad application,or with a text editor of your choice as long as it is able to save the file in raw ASCII format. Be careful to preserve the existing format conventions.

The customized CONFIG file must be RE-SAVED (or SAVED, if created from scratch) as a DOS-type file. If you do not want to overwrite the original CONFIG file, you should use the following procedure, which sets aside the original CONFIG as "CONFIG.ORG" and creates a new file where you can RE-SAVE the customized version:

```
RENAME "CONFIG" to "CONFIG.ORG"
CREATE "CONFIG",17000
RE-SAVE "CONFIG"
```

CHECK FOR DONE

This attribute, when non-zero, enables loss of focus formatting and the "DONE" event.

COLUMNS

The COLUMNS attribute specifies the width of the widget display field in units of character cells.

FONT

This attribute determines which font will be used to display text on the NUMBER widget.For a description of allowed fonts, see Specifying FONT Attribute Values.

FORMAT

The FORMAT attribute specifies which numeric formatting type is used for the NUMBER widget. Allowable types are:

- REAL: the default (-1.79769313486 E +308 to +1.79769313486 E +308)
- SHORT INTEGER: two-byte integer (-32768 to 32767 range)
- LONG INTEGER: four-byte integer (-2147483648 to 2147483647 range)

- BINARY: four-byte integer; two's complement (no "-" allowed)
- OCTAL: four-byte integer; two's complement (no "-" allowed)
- HEX: four-byte integer; two's complement (no "-" allowed)

FORMAT LENGTH

FORMAT LENGTH determines the minimum number of characters used to display formatted numbers (padded with 0s on the left for integer and real formats). The output is never truncated and the actual length may be greater. (See the "TEXT LENGTH" attribute.) FORMAT LENGTH is primarily intended for BINARY, OCTAL, and HEX formats to force formatting to a greater number of bytes.

INCREMENT

The INCREMENT attribute rounds the VALUE to the nearest specified INCREMENT when programmatically setting VALUE, pressing the **Return** key, or loss of focus when CHECK FOR DONE is set. Declaring an INCREMENT of "0" directs HTBasic for Windows not to alter the VALUE. "0" is the default.

MAXIMUM

The MAXIMUM attribute determines the maximum value the NUMBER widget will accept. When changing from one FORMAT to another, NUMBER will use the lesser of the current maximum value and FORMAT's intrinsic maximum (for positive values).

For example, switching formats from LONG INTEGER to SHORT INTEGER will reduce the maximum NUMBER value from a four-byte integer to a two-byte integer (see "FORMAT" attribute for ranges).

MAXIMUM LENGTH

The MAXIMUM LENGTH attribute specifies the maximum number of characters you can enter into the widget - "0" means no limit. (Reformatting may generate a longer display than the MAXIMUM LENGTH, however.)

MINIMUM

The MINIMUM attribute determines the minimum value the NUMBER widget will accept. When changing from one FORMAT to another, NUMBER will use the greater of the current minimum value and FORMAT's intrinsic MINIMUM (for negative values).

For example, switching formats from LONG INTEGER to SHORT INTEGER will reduce the minimum NUMBER value from a four-byte integer to a two-byte integer (see "FORMAT" attribute for ranges).

MODIFIED

The MODIFIED attribute indicates the operator has changed the contents of NUMBER. When responding to the DONE/RETURN event, check the MODIFIED attribute to see if it has been modified. Then, set MODIFIED=0 for the next cycle.

PEN

This attribute determines the pen color used for the characters in the edit field. For a discussion of allowed pens, see Settable Pens Table.

REAL NOTATION

The REAL NOTATION attribute can be used in conjunction with the "FORMAT":"REAL" attribute only. REAL NOTATION allows you to select your "FORMAT:REAL" presentation in one of the following notations:

- FIXED (Example: 12345.6789)
- SCIENTIFIC (Example: 1.233456789E+04)
- ENGINEERING (Example: 12.3456789E+03)

REAL RESOLUTION

The REAL RESOLUTION attribute can be used in conjunction with the "FORMAT":"REAL" attribute only. REAL RESOLUTION allows you to select the number of digits to be displayed to the right of the decimal point.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

TEXT LENGTH

Return-Only Attribute. TEXT LENGTH returns the length in characters of the current formatted number.

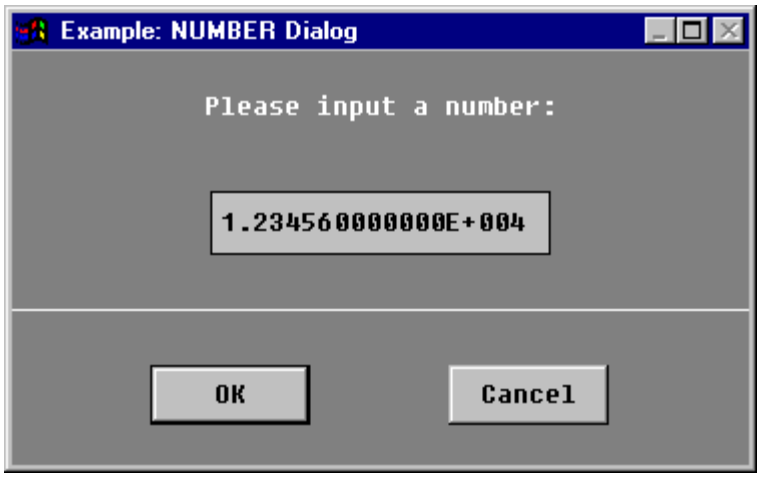
VALUE

This attribute reflects the current value of the NUMBER widget. It is always greater than or equal to MINIMUM, less than or equal to MAXIMUM, and rounded to the nearest INCREMENT. When using string to set VALUE, the string is verified to be a valid number within range and increment. If the value of VALUE exceeds MAXIMUM or MINIMUM, Error 565 is generated.

NUMBER Dialog

Prompts the user to make numeric entries with the keyboard

Example Image



Example Program

See NUMBER Dialog for an example program that produces a display similar to that shown above.

NOTE

See the following program for another example using the NUMBER dialog:

Dialogs Tests

Attributes

See NUMBER Dialog Attributes for the NUMBER dialog attribute list.

Remarks

None.

HTBasic for Windows

NUMBER Dialog Attributes

NUMBER Dialog Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [2]	Numeric	0,1	1
CHECK FOR DONE	Numeric	0,1	1
COLUMNS	Numeric	Any non-negative number	20
DEFAULT BUTTON	Numeric	Any valid index into the DIALOG BUTTONS array	0
DIALOG BUTTONS	String array	Any valid string array	A two-element array that contains "OK" and "Cancel"
EDITABLE	Numeric	0,1	1
FONT	String	*Font* [1]	System dependent [2]
FORMAT	String	"REAL", "SHORT INTEGER", "LONG INTEGER", "INTEGER", "BINARY","OCTAL", "HEX"	"REAL"
FORMAT LENGTH	Numeric	0 to 328	0
HEIGHT	Nu	Any integer number (of	Autosizing

	meric	pixels)	
INCREMENT	Numeric	Any non-negative number	0
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MAXIMUM	Numeric	Between or equal to FORMAT's MINIMUM and MAXIMUM	MAXREAL
MAXIMUM LENGTH	Numeric	0 to 328	328
MINIMIZABLE	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MINIMUM	Numeric	Between or equal to FORMAT's MINIMUM and MAXIMUM	MINREAL
MODIFIED	Numeric	0,1	0
MOVABLE	Numeric	0,1	1
PEN	Numeric	0 to 255	System dependent [2]
REAL NOTATION	String	"FIXED", "SCIENTIFIC", "ENGINEERING"	"SCIENTIFIC"
REAL RESOLUTION	Numeric	0 to 16	12
RESIZABLE	Numeri	0,1	1 (resizable, special

	c		border appears around the dialog)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
TEXT LENGTH	Numeric	Any valid integer	Return only
TITLE	String	Any valid string	NUMBER
USER DATA	String	Any valid string	None
VALUE	Numeric or string	Values allowed by "FORMAT", "MINIMUM", and "MAXIMUM"	0
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Screen or parent work area upper-left corner

CHECK FOR DONE

This attribute, when non-zero, enables loss of focus formatting and the "DONE" event.

COLUMNS

The COLUMNS attribute specifies the width of the dialog display field in units of character cells.

FONT

The FONT attribute determines which font will be used to display text on the dialog.

FORMAT

The FORMAT attribute specifies which numeric formatting type is used for the NUMBER dialog. Allowable types are:

- REAL: the default (-1.79769313486 E +308 to +1.79769313486 E +308)
- SHORT INTEGER: two-byte integer (-32768 to 32767 range)
- LONG INTEGER: four-byte integer (-2147483648 to 2147483647 range)

- BINARY: four-byte integer; two's complement (no "-" allowed)
- OCTAL: four-byte integer; two's complement (no "-" allowed)
- HEX: four-byte integer; two's complement (no "-" allowed)

FORMAT LENGTH

The FORMAT LENGTH determines the minimum number of characters used to display formatted numbers (padded with 0s on the left for integer and real formats). The output is never truncated and the actual length may be greater. (See the "TEXT LENGTH" attribute.) FORMAT LENGTH is primarily intended for BINARY, OCTAL, and HEX formats to force formatting to a greater number of bytes.

INCREMENT

The INCREMENT attribute rounds the VALUE to the nearest specified INCREMENT when programmatically setting VALUE, pressing the **Return** key, or loss of focus when CHECK FOR DONE is set. Declaring an INCREMENT of "0" directs HTBasic for Windows not to alter the VALUE. "0" is the default.

MAXIMUM

The MAXIMUM attribute determines the maximum value the NUMBER dialog will accept. When changing from one FORMAT to another, NUMBER will use the lesser of the current maximum value and FORMATs intrinsic maximum (for positive values).

For example, switching formats from LONG INTEGER to SHORT INTEGER will reduce the maximum NUMBER value from a four-byte integer to a two-byte integer (see "FORMAT" attribute for ranges).

MAXIMUM LENGTH

The MAXIMUM LENGTH attribute specifies the maximum number of characters you can enter into the dialog - "0" means no limit. (Reformatting may generate a longer display than the MAXIMUM LENGTH, however.)

MINIMUM

The MINIMUM attribute determines the minimum value the NUMBER dialog will accept. When changing from one FORMAT to another, NUMBER will use the greater of the current minimum value and FORMATs intrinsic MINIMUM (for negative values).

For example, switching formats from LONG INTEGER to SHORT INTEGER will reduce the minimum NUMBER value from a four-byte integer to a two-byte integer (see "FORMAT" attribute for ranges).

MODIFIED

The MODIFIED attribute indicates the operator has changed the contents of NUMBER. When responding to the DONE/RETURN event, check the MODIFIED attribute to see if it has been modified, then set MODIFIED=0 for the next cycle.

PEN

This attribute determines the pen color used for the characters in the display field. For a discussion of allowed pens, see Settable Pens Table.

REAL NOTATION

The REAL NOTATION attribute can be used in conjunction with the "FORMAT":"REAL" attribute only. REAL NOTATION allows you to select your "FORMAT:REAL" presentation in one of the following notations:

- FIXED (Example: 12345.6789)
- SCIENTIFIC (Example: 1.233456789E+04)
- ENGINEERING (Example: 12.3456789E+03)

REAL RESOLUTION

The REAL RESOLUTION attribute can be used in conjunction with the "FORMAT":"REAL" attribute only. REAL RESOLUTION allows you to select the number of digits to be displayed to the right of the decimal point.

TEXT LENGTH

Return-Only Attribute. TEXT LENGTH returns the length in characters of the current formatted number.

VALUE

This attribute reflects the current value of the NUMBER. It is always
greater than or equal to MINIMUM and less than or equal to MAXIMUM
and rounded to the nearest INCREMENT. When using string to set VALUE,
the string is verified to be a valid number within range and increment.
If the value of VALUE exceeds MAXIMUM or MINIMUM, Error 565 is generated.

NUMBER Widget

Provides an input/output environment for building and formatting numbers

---

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image

Example Program

See *NUMBER Widget* for a program that provides a display similar to that shown above.

NOTE

See Environmental Chamber for another example using the NUMBER widget.

Attributes

See NUMBER Widget Attributes for the NUMBER widget attribute list.

Remarks

The NUMBER widget provides an input/output environment for building and formatting numbers.

The NUMBER widget enforces that a number be entered in an acceptable format, so you cannot enter an incorrect character. The NUMBER widget also provides the ability to translate numbers entered in binary, octal, or hex directly into BASIC numeric variables using the FORMAT attribute.

FORMAT can have values of REAL, SHORT INTEGER, BINARY, OCTAL, and HEX.

In the following example, the FORMAT LENGTH attribute gives a default field of 4 hex digits for the user to enter.

```
CONTROL @Num;SET ("FORMAT":"HEX","FORMAT LENGTH":4)
```

You can specify a round-off INCREMENT and, for REAL numbers, the number of fractional digits displayed (using the REAL RESOLUTION attribute) and the format for REAL numbers (using the REAL NOTATION attribute which can be set to FIXED, SCIENTIFIC, or ENGINEERING).

Input is restricted to valid numbers within the limits, formats, and increments that have been specified. Once a number is entered or the view panel loses its focus, this display is reformatted to the current format increment. MINIMUM, MAXIMUM, REAL NOTATION, and REAL RESOLUTION are verified. If you exceed MINIMUM or MAXIMUM, the previous value is restored.

Events

Events for the NUMBER widget are:

- DONE
- INVALID NUMBER
- KEYSTROKE
- RETURN
- SYSTEM MENU

DONE

This event is generated when the widget loses focus and CHECK FOR DONE is set.

INVALID NUMBER

This event is generated when an entry causes the current display to be unable to build a number in the current "FORMAT". This event also occurs when "DONE" or "RETURN" generates a number outside "MINIMUM" or "MAXIMUM".

KEYSTROKE

Occurs when you enter a character that could change the display.

RETURN

Occurs when the **Return** key or **Enter** key is pressed.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

NUMBER Widget Attributes

NUMBER Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
CHECK FOR DONE	Numeric	0,1	1
COLUMNS	Numeric	Any non-negative number	20
FONT	String	*Font* [1]	System dependent [2]
FORMAT	String	"REAL", "SHORT INTEGER", "LONG INTEGER", "INTEGER", "BINARY", "OCTAL", "HEX"	"REAL"
FORMAT LENGTH	Numeric	0 to 328	0
HEIGHT	Numeric	Any integer number (of pixels)	Varies
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null

INCREMENT	Numeric	Any non-negative number	0
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MAXIMUM	Numeric	Between or equal to FORMAT's MINIMUM and MAXIMUM	MAXREAL
MAXIMUM LENGTH	Numeric	0 to 328	328
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MINIMUM	Numeric	Between or equal to FORMAT's MINIMUM and MAXIMUM	MINREAL
MODIFIED	Numeric	0,1	0
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PEN	Num	0 to 255	System dependent [1]

	eric		
REAL NOTATION	String	"FIXED", "SCIENTIFIC", "ENGINEERING"	"SCIENTIFIC"
REAL RESOLUTION	Numeric	0 to 16	12
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TEXT LENGTH	Nu	Any valid integer	Return only

	meric		
TITLE [5]	String	Any valid string	NUMBER
USER DATA	String	Any valid string	None
VALUE	Numeric or string	Values allowed by "FORMAT", "MINIMUM",and "MAXIMUM"	0
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Varies
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

Notepad

A text editing application, one of the Windows accessories.

HTBasic for Windows

OFF EVENT Command

OFF EVENT

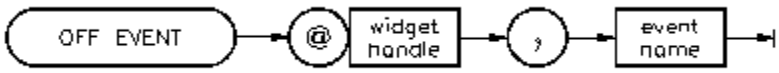
Cancels event branches defined by ON EVENT

Legal Usage	Keyboard Executable	Yes
	Programmable	Yes
	In an IF...THEN...	Yes

Example Statements

```
OFF EVENT @Pushbutton_3,"ACTIVATED"
OFF EVENT @Slider, "DONE"
```

Syntax



Item	Description	Range
event name	name of an event this widget can generate	depends on the widget
widget handle	name identifying a widget	any valid name

Description

The OFF EVENT statement undefines and disables a widget event that was defined and enabled earlier by an ON EVENT statement.

In the following example, the first line causes the program to go to the subroutine Event_handler when the SLIDER widget event CHANGED occurs. Then, the next line cancels the event branch. Thus, after the second line, the occurrence of the event CHANGED will not cause a branch to Event_handler.

```
ON EVENT @Slider;"CHANGED" GOSUB Event_handler
                .
                .
OFF EVENT @Slider, "CHANGED"
```

There are three important differences between the OFF EVENT and DISABLE EVENT statements:

- DISABLE EVENT temporarily disables the event, whereas OFF EVENT permanently deactivates the event.
- Only one occurrence of the event will be logged if the event is disabled with a DISABLE EVENT statement. Therefore, the branch will be taken once the event is reenabled with an ENABLE EVENT statement.
- The event will NOT be logged and the branch will never be taken if the event is deactivated with an OFF EVENT statement.

HTBasic for Windows

ON EVENT Command

ON EVENT

Defines an event branch that is taken after a widget generates that event

---

NOTE

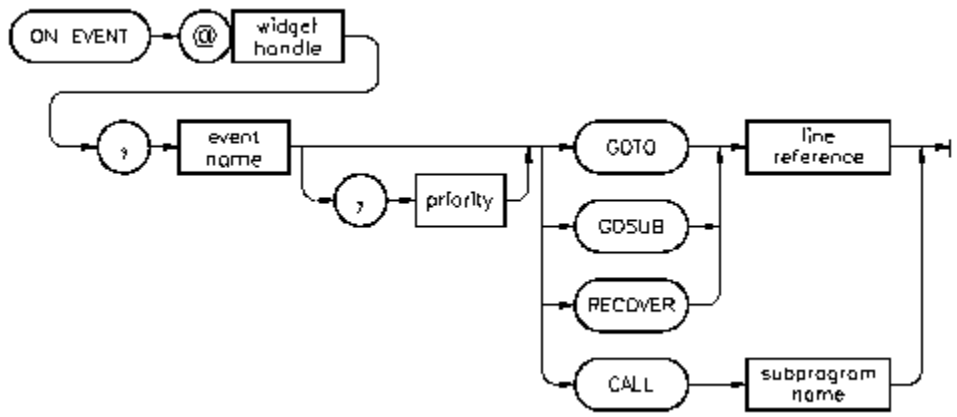
Click the >> bar for additional information on the ON EVENT command.

Legal Usage	Keyboard Executable	Yes
	Programmable	Yes
	In an IF...THEN...	Yes

Example Statements

```
ON EVENT @Pushbutton_3,"ACTIVATED" GOSUB
         Clear_profile
ON EVENT @Slider, "DONE" GOSUB
         Change_temphigh
ON EVENT @Sldr,"CHANGED", VAL(SYSTEM$
         ("SYSTEM PRIORITY"))+1
         GOSUB Chg_stpnt
```

Syntax



Item	Description	Range
event name	name of an event this widget can generate	depends on the widget

line reference	INTEGER constant identifying a program line or line label	1 to 32767 or line label
priority	numeric expression,, rounded to an INTEGER. Default = 1.	1 to 15 (highest)
subprogram name	name of a SUB or CSUB subprogram	any valid name
widget handle	name identifying a widget	any valid name

Description

See Description for the ON EVENT description.

ON EVENT Command (continued)

Description

The ON EVENT statement not only sets up the ON EVENT branch, but also enables the event. For an event to cause branching, the event must be defined AND an event branch must be specified with the ON EVENT command. ON EVENT defines the event branch that is taken after the specified widget generates the specified event.

For example, the following line causes the program to go to the subroutine Event_handler when the SLIDER widget event CHANGED occurs (assuming @Slider is the SLIDER widget handle).

```
ON EVENT @Slider;"CHANGED" GOSUB Event_handler
```

The most recent ON EVENT (or OFF EVENT) statement for a given widget and event combination overrides any previous ON EVENT definition for that combination. If the overriding ON EVENT definition occurs in a context different from the one in which the overridden ON EVENT occurs, the overridden ON EVENT is restored when the calling context is restored.

Any specified *line reference* for GOTO or GOSUB must be in the same context as the ON EVENT statement. CALL and GOSUB will return to the next line that would have been executed if the ON EVENT widget event had not been serviced. The system priority is restored to that which existed before the ON EVENT branch was taken.

RECOVER forces the program to go directly to the specified line in the context containing that ON EVENT statement. When RECOVER forces a change of context, the system priority is restored to that which existed in the original (defining) context at the time that context was exited.

NOTE

The priority specified in the ON EVENT statement (as in all ON-event statements) must be higher than the current system priority in order for the event to be recognized.

When you nest ON EVENT statements, be aware that the system priority is raised to the one you specified in the ON EVENT statement, when that event is serviced for CALL and GOSUB.

To ensure that the events are recognized for all of your ON EVENT statements, specify a higher priority each time you go deeper into the nesting. To do this, query for the current system priority and then increase it by one, instead of specifying the priority as a number between the event name and GOTO, GOSUB, RECOVER, or CALL.

Use the following command sequence within the ON EVENT statement to do this. This technique will cause an error if the current system priority is 15.)

```
VAL(SYSTEM$("SYSTEM PRIORITY"))+1
```

CALL and RECOVER remain active when the context changes to a subprogram or function, unless the change in context is caused by a keyboard-originated call. GOSUB and GOTO remain active when the context changes to a subprogram, but the branch cannot be taken until the calling context is restored.

ON EVENT is disabled by DISABLE EVENT or DISABLE, is reenabled by ENABLE EVENT or ENABLE, and is deactivated by OFF EVENT.

BACKGROUND BITMAP

This attribute specifies the name of a bitmap file for tiling the PANELs background instead of a solid color. BITMAP FILE must be in X Windows Dump File (xwd) or Microsoft ® Windows ® *.BMP format. This is the same as using the BITMAP widget with "FRAME":0, "BORDER":0, and "MAX PENS":0.

MINIMUM X ORIGIN

Read-Only Attribute. Maximum offset, based on the current panel size, of SCROLL X ORIGIN.

MINIMUM Y ORIGIN

Read-Only Attribute. Maximum offset, based on the current panel size, of SCROLL Y ORIGIN.

SCROLL HEIGHT

This attribute specifies the virtual client area height in increments of "SCROLL HEIGHT UNITS."

SCROLL HEIGHT * SCROLL HEIGHT UNITS

must be <= 32767.

SCROLL HEIGHT UNITS

This attribute specifies the number of pixels per unit of scroll height. That is, each increment of the vertical scroll bar will scroll the client area by this many pixels.

SCROLL JUSTIFICATION

SCROLL JUSTIFICATION is used with "SIZE CONTROL":
"SCROLLABLE" when current client area height is not an
integral of "SCROLL HEIGHT UNITS". This attribute specifies
with which edge of the work area to align "SCROLL HEIGHT UNITS".

SCROLL WIDTH

This attribute specifies the virtual client area width in increments of SCROLL WIDTH UNITS.

SCROLL WIDTH * SCROLL WIDTH UNITS

must be <= 32767.

SCROLL WIDTH UNITS

This attribute specifies the number of pixels per unit of scroll width. That is, each increment of the horizontal scroll bar will scroll the client area by this many pixels.

SCROLL X ORIGIN

This attribute specifies the virtual client area origin relative to the upper left of the displayed area in increments of "SCROLL WIDTH UNITS." The maximum offset equals

(actual client width - virtual client width) -OR- 0

whichever is less ("actual client width" is determinable with the INSIDE WIDTH attribute). If the panel is resized, either programmatically or by the operator, so that SCROLL X ORIGIN exceeds the maximum offset, SCROLL X ORIGIN will be changed to maximum offset.

SCROLL Y ORIGIN

This attribute specifies the virtual client area origin relative to the upper left of the displayed area in increments of "SCROLL HEIGHT UNITS." The maximum offset equals

(actual client height-virtual client height)-OR- 0

whichever is less ("actual client height" is determinable with the INSIDE HEIGHT attribute). If the panel is resized, either programmatically or by the operator, so that SCROLL Y ORIGIN exceeds the maximum offset, SCROLL Y ORIGIN will be changed to maximum offset.

SIZE CONTROL

This attribute specifies what the PANEL does to the client area or child widgets when the panel is resized.

- If "NONE", no action occurs
- If "SCROLLABLE", scroll bars will appear IF the virtual client area is greater than the displayed size
- If "RESIZE CHILDREN", child widgets will automatically be resized proportionally.

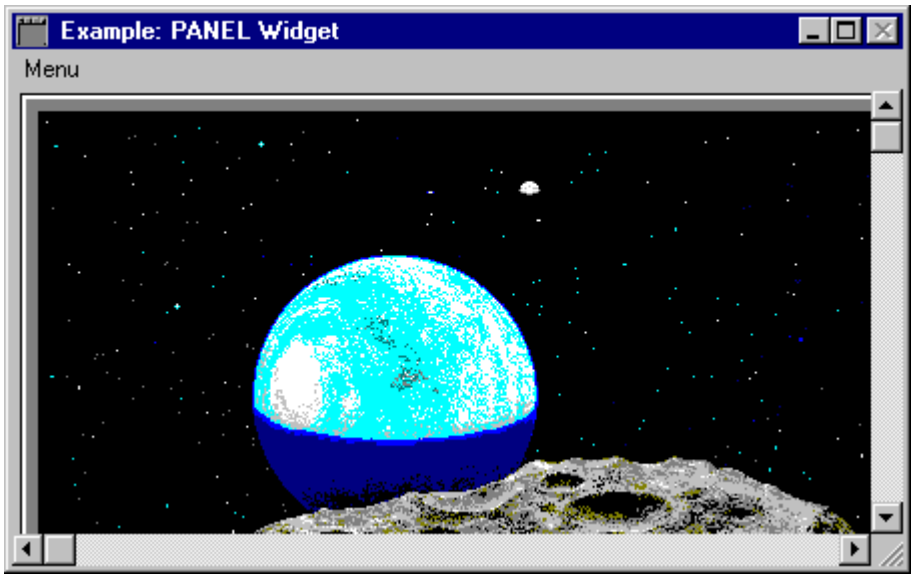
PANEL Widget

Used as a "container" or "parent" widget to most other widgets

---

Legal Usage	Level-0 Widget:	Yes
	Parent to:	All widgets except: MENU BUTTON, MENU TOGGLE, MENU SEPARATOR, and CASCADE MENU
	Child of:	PANEL

Example Image



Example Program

See the *BITMAP Widget* program for an example of the PANEL features. This program provides a display similar to that shown above when the appropriate bitmap file is included in the PANEL widget.

NOTE

See the following programs for other examples using the PANEL widget:

Background Bitmap
Calculator
Context-Sensitive Help
Engine Monitor - Child
Engine Monitor - Panel
Environmental Chamber
Function Generator
HPGL VIEW Viewer
HPGL VIEW in PANEL
Ice Cream Sundae
Menu System
Oven Control
Passwords
PID Controller
PUSHBUTTON Events
RADIOBUTTON Widget
Rock-Paper-Scissors Game
SLIDER Test
STRING Editor
SYSTEM Widget as a Child
Tab Groups
TOGGLEBUTTON Widget
Wing Stress/Vibration Analysis

Attributes

See PANEL Widget Attributes for the PANEL widget attribute list.

Remarks

The PANEL widget is used as a "container" or "parent" widget in which you can place other widgets. A PANEL widget can be both a parent and child widget at the same time. The PANELs "client area" is the central area of the panel window into which child widgets are placed.

PANEL introduces the concept of a "virtual client area". This allows you to create a scrollable client area that is larger than the actual client area (the area currently displayed). With a virtual client area, you can easily manage big child widgets or a number of child widgets that would not normally fit on the screen.

A PANEL widget provides a convenient way to manage a group of component

(children) widgets that logically belong together as part of some user interface. The PANEL widget can be defined as a parent widget, with each of the widgets placed on the PANEL defined as a child widget. When the PANEL is hidden, the child widgets are also hidden also. When the PANEL is moved, the child widgets are moved together, as a group. Child widgets cannot extend beyond the border of their PANEL parent.

When a PANEL widget is a level-0 widget (it can be a child of another PANEL), it can be MOVABLE, MAXIMIZABLE, MINIMIZABLE, RESIZABLE, and it can be given a TITLE. You can set its origin and dimensions and make it visible or invisible. If the PANEL widget disappears, all of its child widgets also disappear. You can make the PANEL invisible when you create it, populate it with child widgets, and then make the whole assemblage visible at one time.

The PANEL has a SIZE CONTROL attribute that allows it to automatically resize its child widgets or to provide a scrollable client area. The default setting is NONE. However, if you set it to RESIZE CHILDREN, all the child widgets are automatically rescaled when you resize the PANEL.

If you set SIZE CONTROL to SCROLLABLE and then resize the PANEL so that it is smaller than a defined "scroll area", scroll bars appear to allow you to scroll the user interface in the PANEL. A set of related attributes allow you to specify the:

- Scroll increments in pixels (SCROLL WIDTH UNITS and SCROLL HEIGHT UNITS)
- Corner of the PANEL to which the scrolling is relative (SCROLL JUSTIFICATION)
- Current scroll origin (SCROLL X ORIGIN and SCROLL Y ORIGIN)
- Size of the scroll area (SCROLL WIDTH and SCROLL HEIGHT).

SCROLL WIDTH UNITS and SCROLL HEIGHT UNITS specify the number of pixels to scroll the PANEL work area. However, the SCROLL WIDTH and the SCROLL HEIGHT are not in pixels, but in multiples of SCROLL WIDTH UNITS and SCROLL HEIGHT UNITS.

For example, if you set SCROLL WIDTH UNITS to 20 (pixels) and then set SCROLL WIDTH to 100, the width over which the PANEL interior scrolls is not 100 pixels, but is 2000 pixels (or 20 * 100). If SCROLL WIDTH or SCROLL HEIGHT evaluates to over 32,767 pixels, an error message is generated.

When placing child widgets into a scrollable PANEL, the origin of the child widgets is relative to the virtual client area, not the actual client area. The SIZE CONTROL attribute allows you to create a scrollable virtual client area. Use SCROLL WIDTH and SCROLL HEIGHT to define the size of the virtual client area.

Events

Events for the PANEL widget are:

- REPAINT
- RESIZED
- SYSTEM MENU

REPAINT

When anything happens on the screen that causes the PANEL to be repainted or redrawn, a REPAINT event is generated. Two examples of events that cause a REPAINT event to be generated are:

- When a dialog comes up over the PANEL, is answered, and then goes away
- When the operator moves or resizes a widget on top of the PANEL

Use the REPAINT event to notify other parts of the program that HTBasic for Windows has changed the screen. The REPAINT event lets other HTBasic graphics know that they need to repaint.

RESIZED

Since a PANEL widget may contain child widgets, resizing it will affect the positioning of the child widgets. Therefore, the PANEL widget will generate a RESIZED event whenever the user resizes it.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

PANEL Widget Attributes

PANEL Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND [2]	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
BACKGROUND BITMAP	String	Existing filename	Null
BORDER [3]	Numeric	0,1	1
HEIGHT	Numeric	Any integer number (of pixels)	250 pixels
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE [5]	Numeri	0,1	0 (not minimizable, button does not

	c		appear in title bar)
MINIMUM X ORIGIN	Numeric	0 to -32768 pixels	-3624 pixels
MINIMUM Y ORIGIN	Numeric	0 to -32768 pixels	-2293 pixels
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SCROLL HEIGHT	Numeric	In SCROLL HEIGHT UNITS such that 1 <= (SCROLL HEIGHT*SCROLL HEIGHT UNITS) <= 32767	10*initial height
SCROLL HEIGHT UNITS	Numeric	1 to 32767	1
SCROLL JUSTIFICATION	String	"BOTTOM", "TOP"	"TOP"
SCROLL WIDTH	Numeric	In SCROLL WIDTH UNITS such that 1 <= (SCROLL WIDTH*SCROLL WIDTH UNITS) <= 32767	10*initial width
SCROLL WIDTH UNITS	Numeri	1 to 32767	1

	c		
SCROLL X ORIGIN	Numeric	-32768 to 0	0
SCROLL Y ORIGIN	Numeric	-32768 to 0	0
SIZE CONTROL	String	"NONE", "SCROLLABLE", "RESIZE CHILDREN"	"NONE"
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TITLE [5]	String	Any valid string	PANEL
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Nu	Any integer number (of pixels)	400 pixels

		meric		
X		Num eric	Any integer, number of pixels from 0,0 [5]	Autoplacement
Y		Num eric	Any integer, number of pixels from 0,0 [5]	Autoplacement

[1] Read the CONFIG file or query for the default with a STATUS command

[2] Only used when BACKGROUND BITMAP is NULL or specified file is not found

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

APPEND TEXT

The value of the APPEND TEXT attribute is the text that is appended to the text provided by the TEXT attribute. That is, APPEND TEXT is used to update the PRINTER widget with new lines of text.

If you execute a STATUS command using a string variable, the last line of the APPEND TEXT will be returned to that string variable.
If you execute a STATUS command using a string array (with n elements) that has fewer elements than VALID LINES, the last n lines will be returned to that string array.

COLUMNS

This attribute specifies the number of columns (number of character cells) that will fit across PRINTER widget WIDTH. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact.

If a text line in PRINTER is longer than is specified by this attribute and the value of LINE WRAP is 0, a horizontal scrollbar appears. If the value of LINE WRAP is 1 and there are more characters than specified, the lines wrap around on character cell-boundaries. If a vertical scrollbar is displayed, it consumes some of the COLUMNS space.

CURRENT LINE

The CURRENT LINE attribute is used as a point of reference for the actions of the CURRENT TEXT, DELETE LINES, HIGHLIGHT CURRENT, and INSERT LINE attributes.

CURRENT TEXT

Return-Only Attribute. CURRENT TEXT reads text beginning with CURRENT LINE.

DELETE LINES

Set-Only Attribute. DELETE LINES deletes a specified number of lines, starting with the CURRENT LINE.

DISPLAY TEXT

DISPLAY TEXT specifies whether or not text sent to the PRINTER is displayed. This allows you to perform many text operations (APPEND, INSERT, DELETE) without updating the display and then do one paint, thus saving considerable overhead.

ENABLE CLEAR DISP

This attribute specifies whether or not the **Clear Display** key will clear the contents of PRINTER.

FONT

The value of this attribute specifies the font used for the text. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact. For a description of allowed fonts, see Specifying FONT Attribute Values.

HIDDEN LINES

The PRINTER widget maintains a total number of lines of text at any one time. The total number is equal to the number of ROWS plus the number of HIDDEN LINES. Since the PRINTER-widget size is adjustable both programmatically and by the operator, the number of visible ROWS can change. If you adjust the size of the PRINTER widget, there are three cases of interest :

- If you have a widget with 3 visible ROWS and 3 HIDDEN LINES (for a total of 6 lines) and you resize it such that only 2 ROWS are visible, HIDDEN LINES is set to 4.

- If you have a widget with 3 visible ROWS and 3 HIDDEN LINES (for a total of 6 lines) and you resize it such that 5 ROWS are visible, HIDDEN LINES is set to 1.

- If you have a widget with 3 visible ROWS and 3 HIDDEN LINES (for a total of 6 lines) and you resize it such that 8 ROWS are visible, HIDDEN LINES is set to 0 and the total number of lines is increased to 8. Now if you resize the widget such that 3 ROWS are visible, HIDDEN LINES is set to 5.

If HIDDEN LINES is set to a value that is less than the value of VALID LINES - ROWS, the oldest lines are lost. Whenever there is off-screen text in the PRINTER widget, a vertical scrollbar appears so that the operator can scroll the window to read the text.

HIGHLIGHT CURRENT

This attribute specifies whether or not the CURRENT LINE is displayed in inverse video. When set to "1", the operator can change the CURRENT LINE by pointing and clicking the mouse.

INSERT TEXT

Set-Only Attribute. INSERT TEXT inserts text above the CURRENT LINE.

LINE WRAP

If the value of this attribute is 1, the text in the PRINTER widget will wrap around at the border of the widget and no horizontal scrollbar will appear. If the value of this attribute is 0, a horizontal scrollbar will appear and you can read the text that exceeds the window-WIDTH by scrolling.

PEN

The value of this attribute specifies the pen color in which the text will be painted. For a discussion of allowed pens, see Settable Pens Table.

ROWS

The number of rows specifies the number of character cells that fit vertically in the PRINTER HEIGHT. You specify the number of rows that you want to appear in the PRINTER widget. If a horizontal scrollbar appears, it consumes some of the ROWS space. See INTERDEPENDENT ATTRIBUTES for details on how ROWS, HEIGHT, COLUMNS, WIDTH, and FONT interact.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

Arrow-Key Behavior When The PRINTER Widget Has The Input Focus

Vertical Arrow-Key Behavior	Horizontal Arrow-Key Behavior
scroll	scroll, if enabled

In addition to the arrow keys, the following keys also work:

- **Clear display**
  - **Prev**
  - **Next**
  - **Home**

All of the above keys and the arrow keys can be preceded by a **Shift** key

TEXT

The value of the TEXT attribute is the text that is to be initially displayed in the PRINTER widget. When you set this attribute, all existing lines in the PRINTER are cleared. If you execute a STATUS command using a scalar, the first line of the TEXT will be returned to that scalar.

If you execute a STATUS command using an array (with n elements) that has fewer elements than VALID LINES, the first n lines will be returned to that string array.

TOP LINE

This attribute sets which line will be the first line displayed in the viewing work area of the PRINTER. For example, if you set the value of TOP LINE to 5, the line that is displayed at the top of the work area is line 5. Line 1 is the first line that is available to be displayed.

The PRINTER widget may adjust your request for displaying the top line to prevent displaying blank lines. You may not set the value of TOP LINE to 0. If you execute a STATUS command to get the value of TOP LINE, and it returns 0, that means that there are no lines in the PRINTER widget.

VALID LINES

This attribute is primarily intended to be used as a RETURN-only attribute. It returns the number of lines present in the PRINTER widget. In SET mode, the VALID LINES attribute may be used to eliminate unwanted lines.

For example, if you set the value of VALID LINES to 10, the "oldest" 10 lines will remain and the most recently appended lines will be lost.

To clear the PRINTER widget, set the value of VALID LINES to 0. If you attempt to set VALID LINES to a value greater than the total number of lines, a "Value out of range" error will result.

HTBasic for Windows

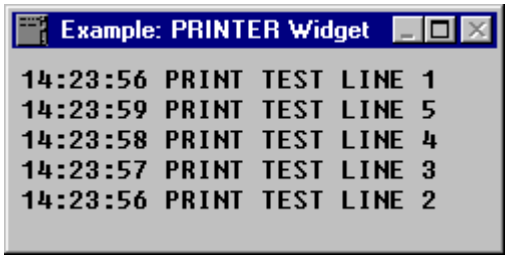
PRINTER Widget

PRINTER Widget

Displays lines of text

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See *PRINTER Widget* for a program that produces a display similar to that shown above.

NOTE

See the following programs for other examples using the PRINTER widget.

Engine Monitor - Level-0
Environmental Chamber
Menu System
RADIOBUTTON Widget
Slot Machine
SYSTEM Widget as a Child
SYSTEM Widget Event Handler
Using ON EVENT

Attributes

See PRINTER Widget Attributes for the PRINTER widget attribute list.

Remarks

The PRINTER widget emulates the behavior of a printer by dumping lines of text to a display and scrolling the lines. The PRINTER widget normally has a text buffer of 100 lines, as given by the sum of the ROWS attribute (which gives the space for lines to be displayed) and the HIDDEN LINES attribute (which gives the space for the remainder that cannot be seen).

The VALID LINES attribute gives the number of lines that are actually used, and you can reposition the text in the display to any line in the buffer with the TOP LINE attribute. The APPEND TEXT attribute prints a new line of text in the widget every time it is used. To skip a line, use APPEND TEXT with a null string. You can also invoke it with a string array as a parameter.

Events

The event for the PRINTER widget is:

- SYSTEM MENU

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

PRINTER Widget Attributes

PRINTER Widget Attributes

Attribute	Type	Allowed Values	Default
APPEND TEXT	String or string array	Any	The null string
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
COLUMNS	Numeric	Any positive integer	50
CURRENT LINE	Numeric	1 to VALID LINES	0
CURRENT TEXT	String or string array	Valid string or string array	Null
DELETE LINES	Numeric	0 to VALID LINES - CURRENT LINE +1	
DISPLAY	N	0,1	1

TEXT	u m eri c		
ENABLE CLEAR DISP	N u m eri c	0,1	1
FONT	St rin g	*Font* [1]	
HEIGHT	N u m eri c	Any integer number (of pixels)	Autosizing based on (16*font height)
HELP FILE	St rin g	Any valid file name	Null
HELP TOPIC	St rin g	Any valid index or topic_id	Null
HIDDEN LINES	N u m eri c	0 to 32767-ROWS	84 (total is 100)
HIGHLIGH T CURREN T	N u m eri c	0,1	0
INSERT TEXT	St rin g or str in g arr ay	Any valid string	Null
INSIDE HEIGHT	N u m eri c	Any integer number (of pixels)	Varies
INSIDE WIDTH	N u m eri c	Any integer number (of pixels)	Varies
LINE WRAP	N u m	0,1	0

	eric		
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
ROWS	Numeric	Any positive integer	16
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or String arr	Any valid string or string array	Null

	ay		
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TEXT	String or string array	Any	Null string
TITLE [5]	String	Any valid string	PRINTER
TOP LINE	Numeric	1 through VALID LINES	0 (no lines)
USER DATA	String	Any valid string	None
VALID LINES	Numeric	Any non-negative number	0
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing based on (50*font width)

X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

LABEL

The value of this attribute is a string that is used as the header on the PULLDOWN MENU. The string is placed in the PANELs menu bar.

SENSITIVE

This attribute provides you with the capability to keep the LABEL of the PULLDOWN MENU widget visible, but make it unresponsive to user input.

- If the value of SENSITIVE is 1, the widget is responsive to user input.
- If the value of SENSITIVE is 0, the widget will not respond to user input and will appear "grayed out" to indicate its unresponsive status.

USER DATA

The contents of this attribute are left to the programmer and are never changed by HTBasic for Windows. It is simply a way to associate program-specific data with a specific widget.

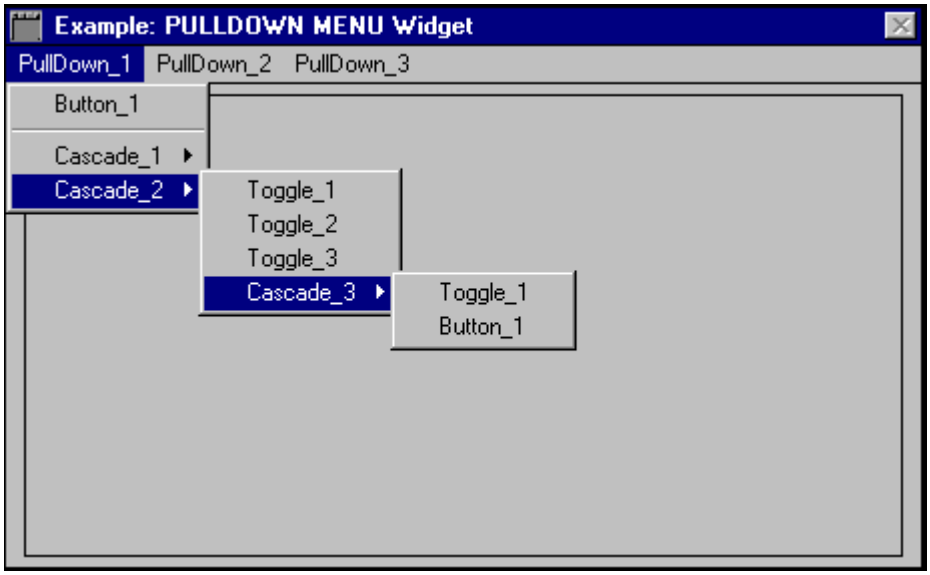
PULLDOWN MENU Widget

Provides a menu of selections from which the operator can choose

---

Legal Usage	Level-0 Widget:	Yes
	Parent to:	MENU BUTTON, MENU TOGGLE, MENU SEPARATOR, CASCADE MENU
	Child of:	PANEL

Example Image



Example Program

See the *Menu System* program for an example program that shows how to create a PULLDOWN MENU widget.

NOTE

See the following programs for other examples using the PULLDOWN MENU widget:

Environmental Chamber

Function Generator
HPGL VIEW Viewer
Passwords
SLIDER Test
STRING Editor

Attributes

See PULLDOWN MENU Widget Attributes for the PULLDOWN MENU widget attribute list.

Remarks

The PULLDOWN MENU widget is used to provide a menu of selections from which the user can choose. To activate a PULLDOWN MENU with a mouse, perform one of these procedures:

- Depress the left mouse button when the cursor is in the boundary of the PULLDOWN MENUs label area and hold, dragging down to the topic you want, then releasing to access that topic.
- Depress the left mouse button when the cursor is in the boundary of the PULLDOWN MENUs label area and release, showing the available topics. Then move the cursor to the desired available topic, depress the mouse button and release to access that topic.

To activate a PULLDOWN MENU using just the keyboard:

- Capture and hold the focus by depressing the **Alt** or **Extend char** key. Keep holding the key down.
- Press and release the **Tab** key. This focuses the cursor onto the menu bar.
- Move through the panel selections with the arrow keys.
- When you have reached the item you want, select it by pressing the **Return** key
- Release the **Alt** or **Extend char** key

Events

None.

HTBasic for Windows

PULLDOWN MENU Widget Attributes

PULLDOWN MENU Widget Attributes

Attribute	Type	Allowed Values	Default
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
LABEL	String	Any	Null String
SENSITIVE	Numeric	0,1	1
USER_DATA	String	Any valid string	None

COLUMNS

This attribute specifies the number of columns (number of character cells) that will fit across the PUSHBUTTON width. See INTERDEPENDENT ATTRIBUTES for details on how COLUMNS, WIDTH, and FONT interact.

FONT

Specifies the font to be used for the text in the LABEL attribute. For a description of allowed fonts, see Specifying FONT Attribute Values.

LABEL

The string you enter for this attribute will be displayed on the PUSHBUTTON. That is, it sets the label for the current state (see the STATE attribute).

LABELS

The LABELS attribute sets labels for multiple button states. For example, if three labels are sent to the PUSHBUTTON, it will have three potential states (the actual number is set by the STATES attribute).

When you click on the PUSHBUTTON, the current label reflects the current state, and the PUSHBUTTON advances to subsequent states with more clicks. A LABELS string array can have less or more labels than there are states, although usually the number is the same.

PANEL DEFAULT

This attribute takes effect for non-level-0 PUSHBUTTON widgets only.
If the value is 1, this PUSHBUTTON is the default for this PANEL.
Pressing **Return** when the focus is anywhere in the PANEL will activate this PUSHBUTTON. Only one PUSHBUTTON can be the default button in a level-0 PANEL. If you set one, all others are reset.

PEN

The value of this attribute specifies the pen color that will be used to paint the LABEL text on the PUSHBUTTON. For a discussion of allowed pens, see Settable Pens Table.

SENSITIVE

This attribute provides you with the capability to keep the PUSHBUTTON widget visible, but make it unresponsive to user input.

- If the value of SENSITIVE is 1, the widget is responsive to user input.
- If the value is 0, the widget will not respond to user input, and will appear "grayed out" to indicate its unresponsive status.

STATE

STATE represents the current state of the button. When set, STATE will change the label to the string represented by an index into the LABELS attribute. The index is assumed to be OPTION BASE 0.

STATES

STATES sets the number of states, from 1 to 256.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

Arrow-Key Behavior When PUSHBUTTON Widget Has Input Focus

Vertical Arrow-Key Behavior	Horizontal Arrow-Key Behavior
move to next widget	move to next widget

HTBasic for Windows

PUSHBUTTON Widget

PUSHBUTTON Widget

A stand-alone button that can be placed anywhere on the screen or in a PANEL

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image

Example Program

See *PUSHBUTTON Widget* for a program that creates a PUSHBUTTON widget and provides a display similar to that shown above.

See the following programs for other examples using the PUSHBUTTON widget:

- *Alarm Clock*
- *Calculator*
- *Context-Sensitive Help*
- *Cyclical PUSHBUTTON*
- *Environmental Chamber*
- *Function Generator*
- *Hammer Game*
- *Ice Cream Sundae*
- *PID Controller*
- *PUSHBUTTON Events*
- *Rock-Paper-Scissors Game*
- *Slot Machine*
- *Tab Groups*
- *Tic-Tac-Toe*
- *Using ON EVENT*
- *Wing Stress/Vibration Analysis*

Attributes

See PUSHBUTTON Widget Attributes for the PUSHBUTTON widget attribute list.

Remarks

The PUSHBUTTON widget is a stand-alone button that can be placed anywhere on the screen or in a PANEL.

Clicking the mouse button (or pressing the **Spacebar** or the **Return** key) while the focus in on the PUSHBUTTON causes an ACTIVATED event to be generated. For example, if you execute the following statements, when you click on the "Push Me" button the "Handler" routine will be executed.

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

If you specify a string array as a parameter to LABELS, and specify the number of elements in the array with the STATES attribute, PUSHBUTTON will cycle through the different labels. You can query the current PUSHBUTTON label setting with the STATE attribute.

Clicking the mouse button (or pressing the **Spacebar** or the **Return** key) while the focus is on the PUSHBUTTON causes an ACTIVATED event to be generated. PUSHBUTTON can be a multi-state device (see LABEL and STATES attributes) where each push of the button cycles to the next state.

Events

Events for the PUSHBUTTON widget are:

- ACTIVATED
- SYSTEM MENU

ACTIVATED

This event occurs only when the mouse button is pushed and released while the the cursor is within the border of the PUSHBUTTON.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

PUSHBUTTON Widget Attributes

PUSHBUTTON Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
COLUMNS	Numeric	Any positive integer	10
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
INSIDE_HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE_WIDTH	Numeric	Any integer number (of pixels)	Varies
LABEL	String	Any valid string	Initial title
LABELS	String array	Any valid string array	One-element array containing initial title

MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PANEL DEFAULT	Numeric	0,1	0
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SENSITIVE	Numeric	0,1	1
STACKING ORDER	Numeric	0 to number of siblings [4]	0
STATE	Numeri	0 to 255	0

	c		
STATES	Numeric	1 to 256	1
SYSTEM MENU [5]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TITLE [5]	String	Any valid string	PUSHBUTTON
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

Y		Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

HTBasic for Windows

Physical Pens Table

This table shows pen numbers, colors, and red/green/blue values for HTBasic for Windows physical pens. See Logical Pens Table for associated logical pen numbers. See Settable Pens Table for settable pens for HTBasic for Windows.

Pen	Color	Red	Green	Blue
0	black	0	0	0
1	white	255	255	255
2	red	255	0	0
3	yellow	255	255	0
4	green	0	255	0
5	cyan	0	255	255
6	blue	0	0	255
7	magenta	255	0	255
8	dialog gray	162	156	146
9	parchment white	220	211	184
10	forest green	70	180	70
11	inactive gray	176	174	169
12	evening gold	150	110	75

13	french gray	105	95	80
14	lavender	119	123	200
15	beige gray	148	139	123
16	black	0	0	0
17	white	255	255	255
18	red	255	0	0
19	yellow	255	255	0
20	green	0	255	0
21	cyan	0	255	255
22	blue	0	0	255
23	magenta	255	0	255
24	dialog gray	162	156	146
25	parchment white	220	211	184
26	forest green	70	180	70
27	inactive gray	176	174	169
28	evening gold	150	110	75
29	french gray	105	95	80
30	lavender	11	12	20

		9	3	0
31	beige gray	148	139	123
48	black	0	0	0
49	white	255	255	255
50	red	255	0	0
51	yellow	255	255	0
52	green	0	255	0
53	cyan	0	255	255
54	blue	0	0	255
55	magenta	255	0	255
56	dialog gray	162	156	146
57	parchment white	220	211	184
58	forest green	70	180	70
59	inactive gray	176	174	169
60	evening gold	150	110	75
61	french gray	105	95	80
62	lavender	119	123	200
63	beige gray	148	139	123

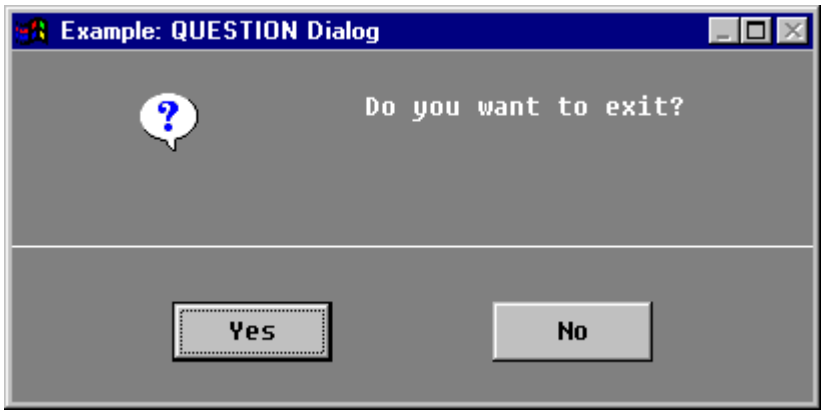
24 0	black	0	0	0
24 1	white	25 5	25 5	25 5
24 2	red	25 5	0	0
24 3	yellow	25 5	25 5	0
24 4	green	0	25 5	0
24 5	cyan	0	25 5	25 5
24 6	blue	0	0	25 5
24 7	magenta	25 5	0	25 5
24 8	dialog gray	16 2	15 6	14 6
24 9	parchment white	22 0	21 1	18 4
25 0	forest green	70 0	18 0	70 0
25 1	inactive gray	17 6	17 4	16 9
25 2	evening gold	15 0	11 0	75
25 3	french gray	10 5	95 0	80
25 4	lavender	11 9	12 3	20 0
25 5	beige gray	14 8	13 9	12 3

QUESTION Dialog

Prompts the operator with a question and suspends program execution until the operator answers the question

---

Example Image



Example Program

See QUESTION Dialog for an example program that produces a display similiar to that shown above.

NOTE

See the following program for another example using the QUESTION dialog:

Dialogs Tests

Attributes

See QUESTION Dialog Attributes for the QUESTION dialog attribute list.

Remarks

None.

HTBasic for Windows

QUESTION Dialog Attributes

QUESTION Dialog Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
DEFAULT BUTTON	Numeric	Any valid index into the DIALOG BUTTONS array	0
DIALOG BUTTONS	String array	Any valid string array	A two-element array containing Yes and No
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE	Numeric	0,1	0 (minimizable, button not present)
MOVABLE	Numeric	0,1	1
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE	Numeric	0,1	1 (resizable, special border appears

			around the dialog)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
TITLE	String	Any valid string	QUESTION
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Screen or parent work area upper-left corner

FONT

Specifies the font to be used for any text in the QUESTION dialog.For a description of allowed fonts, see Specifying FONT Attribute Values.

PEN

Specifies the color to be used for the foreground area of the dialog.
For a discussion of allowed pens, see Settable Pens Table.

HTBasic for Windows

Querying Widget Attributes

Querying Widget Attributes

After you have created a widget with an ASSIGN command, you can use the STATUS command to query the values of the widget attributes. The STATUS command syntax is:

STATUS @*widget handle*;RETURN (*return attribute list*)

Since attribute values may be changed by the user or by some sequence of events, to ensure correct operation of your program, you may want to query the widget for the current value of an attribute before using it in your program.

Returning Attribute Values

There are three ways to query and return attribute values using the STATUS command:

- You can return one attribute value with a single STATUS statement. For example:

```
STATUS @Meter;RETURN ("VALUE":Meterval)
```

- You can also return multiple attribute values with multiple STATUS statements. For example:

```
STATUS @Bars;RETURN ("BACKGROUND":Barsbackgnd)
STATUS @Bars;RETURN ("SHOW LIMITS":Status)
STATUS @Bars;RETURN ("VALUE":Bar1val)
```

- You can also return multiple attribute values with a single STATUS statement. For example:

```
STATUS @Bars;RETURN ("BACKGROUND":Barsbackgnd,"SHOW LIMITS":Status,
"VALUE":Bar1val)
```

COLUMNS

This attribute specifies the number of columns (number of character cells) that will fit across the LABEL attribute in the RADIOBUTTON WIDTH. See INTERDEPENDENT ATTRIBUTES for details on how COLUMNS, WIDTH, and FONT interact.

FONT

Specifies the font to be used for the text in the LABEL attribute. For a description of allowed fonts, see Specifying FONT Attribute Values.

LABEL

The string you enter for this attribute will be displayed in the RADIOBUTTON.

PEN

The value of this attribute specifies the pen color that will be used to paint the LABEL text in the RADIOBUTTON. For a discussion of allowed pens, see Settable Pens Table.

SENSITIVE

This attribute provides you with the capability to keep the RADIOBUTTON widget visible, but make it unresponsive to user input.

- If the value of SENSITIVE is 1, the widget is responsive to user input.
- If the value of SENSITIVE is 0, the widget will not respond to user input, and will appear "grayed out" to indicate its unresponsive status.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets. To use the RADIOBUTTON widget in a bank of buttons in which only one button at a time can be depressed, you must establish a tab group of RADIOBUTTON widgets.

Arrow-Key Behavior When RADIOBUTTON Widget Has Input Focus

Vertical Arrow-Key Behavior	Horizontal Arrow-Key Behavior
move to next widget	move to next widget

VALUE

The value of this attribute specifies the state of the RADIOBUTTON.

- If the value is 1, this RADIOBUTTON is the one in the bank of RADIOBUTTONs that has been "pushed" or selected by the operator.
- If the value is 0, this RADIOBUTTON is not the one in the bank of RADIOBUTTONs that has been "pushed" or selected by the operator.

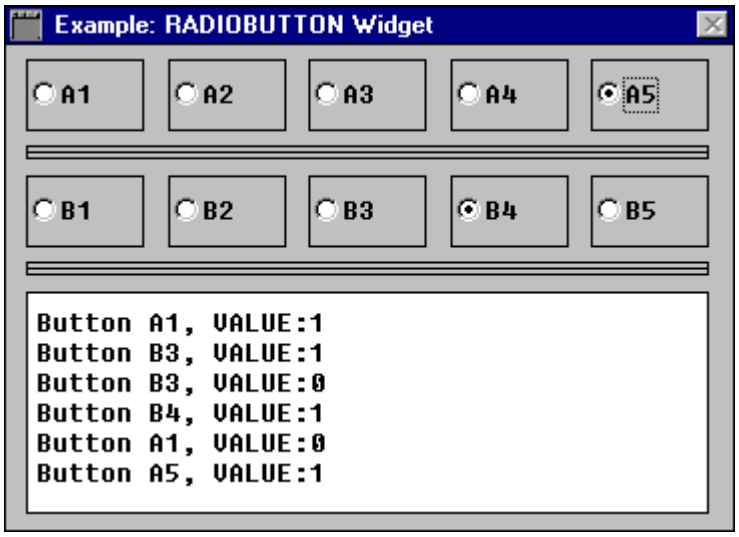
RADIOBUTTON Widget

Provides a bank of buttons in which only one button at a time can be depressed

---

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See *RADIOBUTTON Widget* for an example program that produces a display similar to that shown above.

Attributes

See RADIOBUTTON Widget Attributes for the RADIOBUTTON widget attribute list.

Remarks

The RADIOBUTTON widget displays a bank of buttons, only one of which can be depressed at a time. The RADIOBUTTON widget exhibits the properties of an old automobile radio's buttons. That is, when you press one button, the others

pop out or are released. The RADIOBUTTON widget works correctly only when it is a member of a "tab group".

The RADIOBUTTON widget is an extension of the TOGGLEBUTTON concept. Like a TOGGLEBUTTON, a RADIOBUTTON has a VALUE attribute. Click on the button once, and VALUE is set to 1. Click on it again, and VALUE is cleared back to 0.

A CHANGED event can be enabled to occur whenever you click on it. Each RADIOBUTTON has a circle in it. Set the RADIOBUTTON and the circle appears filled. Clear the RADIOBUTTON and the circle is cleared.

If you set or clear a TOGGLEBUTTON it has no direct effect on anything else in the user interface. However, if you create a tab group of RADIOBUTTONs, only one RADIOBUTTON in the group can be set at a time. If you set a different RADIOBUTTON in the tab group, any other RADIOBUTTON already set will be cleared.

The RADIOBUTTON widget operation is as described here only when it is a member of a "tab group".If one RADIOBUTTON in a tab group has been set to 1 and you set another RADIOBUTTON to 1, TWO CHANGED events occur - one for the original button as it is cleared, and another for the new button as it is set.

When you click on a RADIOBUTTON, an event is generated for the button and the program lists the state (set or 1, clear or 0) of the button in the PRINTER widget. If you set a button, and there is already a button set in the same tab group, the "old" button will be cleared, generating an event to have its cleared state listed in the PRINTER widget. Also, the "new" button will be set, generating a second event to have its set state listed in the PRINTER widget.

Clicking on a button in a tab group causes that button's circle to be set, with no other effects in any other group. However, if you then click on another button in a tab group, the first button's circle in that group will be cleared and the new button's circle will be set. RADIOBUTTON behavior is convenient for selecting a value from a set of members that are mutually exclusive. That is, a set where you can choose one of several, but never more than one at a time.

Events

Events for the RADIOBUTTON widget are:

- CHANGED
- SYSTEM MENU
- CHANGED

An event is generated when the user "presses" the RADIOBUTTON by clicking on it with the left mouse button or by pressing the **Spacebar** while the focus is on the RADIOBUTTON. Within a tab group of RADIOBUTTON widgets, two CHANGED events will be generated when you click on one of the RADIOBUTTON widgets:

- First, the previously selected button generates a CHANGED event as it is deselected.
- Second, the button you select now generates a CHANGED event

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

RADIOBUTTON Widget Attributes

RADIOBUTTON Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
COLUMNS	Numeric	Any positive integer	10
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
INSIDE_HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE_WIDTH	Numeric	Any integer number (of pixels)	Varies
LABEL	String	Any valid string	Null string
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title

			bar)
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SENSITIVE	Numeric	0,1	1
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeri	0 to number of items in system menu	0

	c		
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TITLE [5]	String	Any valid string	RADIOBUTTON
USER DATA	String	Any valid string	None
VALUE	Numeric	0,1	0
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

RESIZABLE

Dialogs and Level-0 Widgets Only. If the value of this attribute is set to 1, a special "resize border" will be drawn around the widget or dialog. If the value of this attribute is set to 0, no border appears around the widget or dialog and it is not resizable. You cannot resize the widget or dialog from the keyboard (without a mouse interface). To use the resize border:

1. Move the mouse until the pointer-cursor is on the border (the pointer-cursor changes into the resize-cursor)
2. Press and hold the left mouse button
3. Move the mouse to resize the widget
4. Release the mouse button. The widget will redraw to the new size.

RESTORE SCREEN

This attribute provides the capability to save the HTBasic screen under the dialog or widget. If RESTORE SCREEN is set to 1, the widget management software will save a "snapshot" of the BASIC screen area where the dialog or widget will appear, with whatever graphics or text the screen area contains.

When the widget is destroyed or the dialog terminates, the screen area under the widget or dialog will be restored to its original appearance. If the value is set to 0, nothing is saved and the area under the dialog or widget will be black when the widget is destroyed or the dialog terminates.

When you set RESTORE SCREEN to 1, the MOVABLE attribute is automatically set to 0. That way, the operator will be unable to move the widget and cause the screen to be restored improperly. It is highly recommended that the RESIZABLE attribute also be set to 0.

NOTE

Because your program will use a significantly larger amount of memory when you set RESTORE SCREEN to 1, do so only when it is important to save the text or graphics that are already displayed on the screen.

DIRECT MOVE

If the value is 1, you can click at any location in the SCROLLBAR trough and the indicator jumps to that location. Setting DIRECT MOVE to 1 overrides the MAJOR INCREMENT attribute.

MAJOR INCREMENT

The value of this attribute specifies the increment by which the VALUE of the SCROLLBAR jumps each time the user clicks in the SCROLLBAR trough. This value should be less than MAXIMUM - MINIMUM.

MAXIMUM

This attribute specifies the value that VALUE will have when the slider is at the extreme right or bottom of the SCROLLBAR. MAXIMUM may be less than MINIMUM, in which case the sense of the SCROLLBAR is reversed. That is, for a vertical SCROLLBAR, values are larger for positions nearer the bottom, and for a horizontal SCROLLBAR, values are larger for positions nearer the right.

MINIMUM

This attribute specifies the value that VALUE will have when the slider is at the extreme left or top of the SCROLLBAR. MINIMUM may be greater than MAXIMUM, in which case the sense of the SCROLLBAR is reversed. That is, for a vertical SCROLLBAR, values are larger for positionsnear the top, for a horizontal SCROLLBAR, values are larger for positions nearer the left.

MINOR INCREMENT

The value of this attribute specifies the increment by which the value of the VALUE attribute of the SCROLLBAR jumps each time the user clicks on the SCROLLBAR arrows. This value should be less than MAJOR INCREMENT.

ORIENTATION

This attribute determines the direction or layout of the SCROLLBAR.

ORIENTATION Attribute Values

Allowed Value	Description
"UP", "VERTICAL"	The SCROLLBAR will be oriented vertically and the VALUE attribute will decrease in the up direction
"RIGHT", "HORIZONTAL"	The SCROLLBAR will be oriented horizontally and the VALUE attribute will increase towards the right

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

Arrow-Key Behavior When SCROLLBAR Widget Has Input Focus

Vertical Arrow-Key Behavior	Horizontal Arrow-Key Behavior
scroll	scroll

VALUE

This attribute reflects the current setting of the SCROLLBAR slider.
It can be outside the range MINIMUM through MAXIMUM.

SCROLLBAR Widget

Inputs a relative position between a minimum and maximum value

---

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image

Example Program

See *SCROLLBAR Widget* for a program that provides a display.similar to that shown above.

NOTE

See the following programs for other examples using the SCROLLBAR widget:

Alarm Clock
Oven Control

Attributes

See SCROLLBAR Widget Attributes for the SCROLLBAR widget attribute list.

Remarks

The SCROLLBAR widget is used to input a relative position between a minimum and a maximum value. For a vertical scrollbar, the minimum value is at the top. For a horizontal scrollbar, the minimum value is at the left. Clicking and holding on arrows will autorepeat.

The SCROLLBAR widget is a modified version of the SLIDER widget. The purpose of the SCROLLBAR widget is to allow you to "scroll" a PANEL that is too large for the

display.

The SCROLLBAR widget has no LOGARITHMIC mode. It can only be operated in a linear fashion. It also counts in the reverse direction from a SLIDER. If you move the slide up in a SLIDER, the VALUE increments, but in a SCROLLBAR, the VALUE decrements.

The SCROLLBAR widget consists of a trough with a slide, with arrows at each end. The SCROLLBAR WIDTH cannot be set, as it has a fixed WIDTH on a given display. To position the SCROLLBAR properly you must read its WIDTH and then move the widget accordingly.

Events

Events for the SCROLLBAR widget are:

- CHANGED
- DONE
- SYSTEM MENU

CHANGED

This event is generated every time the user changes the VALUE by clicking on arrows or in trough or by clicking and dragging the slider.

DONE

This event is generated only when the user releases the mouse button.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

SCROLLBAR Widget Attributes

SCROLLBAR Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
BORDER [2]	Numeric	0,1	1
DIRECT_MOVE	Numeric	0,1	0 (use major and minor increments)
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
INSIDE_HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE_WIDTH	Numeric	Any integer number (of pixels)	Varies
MAJOR_INCREMENT	Numeric	Any non-negative real number	10
MAXIMIZABLE [4]	Numeric	0,1	1 (maximizable, button appears in title bar)

MAXIMUM	Numeric	Any real number	100
MINIMIZABLE [4]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MINIMUM	Numeric	Any real number	0
MINOR INCREMENT	Numeric	Any non-negative real number	1
MOVABLE [4]	Numeric	0,1	1 (movable, click on title bar and drag)
ORIENTATION	String	"UP", "VERTICAL", "RIGHT", "HORIZONTAL"	"VERTICAL" (minimum value at top)
RESIZABLE [4]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
STACKING ORDER	Numeric	0 to number of siblings [3]	0
SYSTEM MENU [4]	String or String	Any valid string or string array	Null

	array		
SYSTEM MENU COUNT [4]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [4]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TITLE [4]	String	Any valid string	SCROLLBAR
USER DATA	String	Any valid string	None
VALUE	Numeric	Any real number	0
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [5]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [5]	Autoplacement

[1] Read the CONFIG file or query for the default with a STATUS command

[2] Child widget only

[3] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[4] For level-0 widgets only

[5] Screen or parent work area upper-left corner

ORIENTATION

This attribute determines the direction or layout of the SEPARATOR.

NOTE

There is no difference between "UP", "DOWN", and "VERTICAL" orientation for the SEPARATOR. Similarly, there is no difference between "LEFT", "RIGHT", and "HORIZONTAL" orientation for the SEPARATOR.

ORIENTATION Attribute Values

Allowed Value	Description
"LEFT", "RIGHT", "HORIZONTAL"	The SEPARATOR will be oriented horizontally.
"UP", "DOWN", "VERTICAL"	The SEPARATOR will be oriented vertically.

PEN

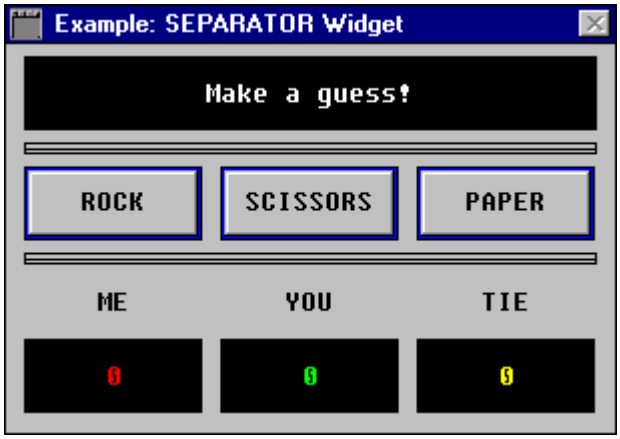
The value of this attribute specifies the pen color that will be used to paint the SEPARATOR. For a discussion of allowed pens, see Settable Pens Table.

SEPARATOR Widget

Provides a line that is used to visually separate areas of a PANEL

Legal Usage	Level-0 Widget:	No
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See Rock-Paper-Scissors Game for an example program using the SEPARATOR widget. The program provides a display similar to that shown above.

NOTE

See the following program for another example using the SEPARATOR widget:

RADIOBUTTON Widget

Attributes

See SEPARATOR Widget Attributes for the SEPARATOR widget attribute list.

Remarks

The SEPARATOR widget is a line that is used to visually separate areas of a PANEL. The SEPARATOR widget provides a useful accessory to a PANEL. It can only be a child of a PANEL. The SEPARATOR widget draws a line across portions of the PANEL to allow separation of logical areas. You can set a HEIGHT on the SEPARATOR to give it a more distinctive appearance.

Events

None.

HTBasic for Windows

SEPARATOR Widget Attributes

SEPARATOR Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
BORDER [2]	Numeric	0,1	1
HEIGHT	Numeric	Any integer number (of pixels)	15 pixels
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
INSIDE_HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE_WIDTH	Numeric	Any integer number (of pixels)	Varies
ORIENTATION	String	"UP", "DOWN", "VERTICAL", "LEFT", "RIGHT", "HORIZONTAL"	"HORIZONTAL"
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
STACKING_ORDER	Numeric	0 to number of siblings [3]	0
USER	St	Any valid string	None

DATA	rin g		
VERSION	St rin g	Any valid string	Current version
VISIBLE	N u m eri c	0,1	1
WIDTH	N u m eri c	Any integer number (of pixels)	Width of the parent widget
X	N u m eri c	Any integer, number of pixels from 0,0 [4]	Autoplacement
Y	N u m eri c	Any integer, number of pixels from 0,0 [4]	Autoplacement

[1] Read the CONFIG file or query for the default with a STATUS command

[2] Child widget only

[3] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[4] Screen or parent work area upper-left corner

AUTO REPEAT

The AUTO REPEAT attribute moves the SLIDER continuously when the user presses the arrows or trough on the SLIDER display. The speed of the SLIDERs movement is controlled by its setting in the CONFIG file. The speed can be changed using the *initial_delay* and *repeat_delay* variables in the "@scrollbar" section.

COLUMNS

This attribute specifies the number of columns (number of character cells) that will fit across the SLIDER widget's VALUE box (the SLIDER pad).

DECIMAL DIGITS

The value of this attribute sets the number of digits after the decimal point that you want to display in the slider bar. The physical width of the SLIDER widget (controlled by the WIDTH attribute) limits the number of digits that can actually be displayed.

DIRECT MOVE

If the value is 1, you can click at any location in the SLIDER trough and the indicator jumps to that location. Setting DIRECT MOVE to 1 overrides the MAJOR INCREMENT attribute.

DISPLAY BACKGROUND

This attribute changes the color of the minimum, maximum, and value backgrounds. For a discussion of allowed pens, see Settable Pens Table.

EXPONENTIAL FORMAT

If you set this attribute to 1, the value of VALUE will be displayed with an exponent in the slider bar. If the value is 0, the value of VALUE will be displayed as a fixed REAL number.

FONT

Specifies the font to be used for the text in the VALUE and LIMITS attributes.
For a description of allowed fonts, see Specifying FONT Attribute Values.

LOGARITHMIC

If the value is 1, the SLIDER operates in a logarithmic mode.
If the value is 0, the SLIDER operates in a linear mode.

MAJOR INCREMENT

The value of this attribute specifies the increment by which the VALUE of the SLIDER jumps each time the user clicks in the SLIDER trough.

- If LOGARITHMIC is 0, MAJOR INCREMENT is in the same units as VALUE.
- If LOGARITHMIC is 1, MAJOR INCREMENT is in units of one one-hundredth of a decade.

MAXIMUM

This attribute specifies the value that VALUE will have when the slider is at the extreme top or right of the SLIDER. MAXIMUM may be less than MINIMUM, in which case the sense of the SLIDER is reversed. That is, for a vertical SLIDER, values are larger for positions nearer the bottom, and for a horizontal SLIDER, values are larger for positions nearer the right.

MINIMUM

This attribute specifies the value that VALUE will have when the slider is at the extreme bottom or left of the SLIDER. MINIMUM may be greater than MAXIMUM, in which case the sense of the SLIDER is reversed. That is, for a vertical SLIDER, values are larger for positions near the top, and for a horizontal SLIDER, values are larger for positions nearer the left.

MINOR INCREMENT

The value of this attribute specifies the increment by which the VALUE of the SLIDER jumps each time the user clicks on the slider arrows. If LOGARITHMIC is 0, MINOR INCREMENT is in the same units as VALUE and the SLIDER arrows are turned off. If LOGARITHMIC is 1, MINOR INCREMENT is in units of one one-hundredth of a decade.

ORIENTATION

This attribute determines the direction or layout of the SLIDER.

ORIENTATION Attribute Values

Allowed Value	Description
"UP", "VERTICAL"	The SLIDER will be oriented vertically and the VALUE attribute will increase in the up direction
"RIGHT", "HORIZONTAL"	The SLIDER will be oriented horizontally and the VALUE attribute will increase toward the right

PEN

PEN controls the color of the text and slot border. For a discussion of allowed pens, see Settable Pens Table.

SHOW LIMITS

This attribute is used to turn on and off the limits boxes on the SLIDER. If its value is 1, the limits boxes are visible. If its value is 0, the limits boxes are invisible.

SHOW VALUE

This attribute is used to turn on and off the value box in the SLIDER bar. If its value is 1, the value box is visible. If its value is 0, the value box is invisible.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

Arrow-Key Behavior When The SLIDER Widget Has The Input Focus

Vertical Arrow-Key Behavior	Horizontal Arrow-Key Behavior
slide	slide

VALUE

This attribute reflects the current setting of the SLIDER. It is always displayed and is always greater than or equal to MINIMUM and less than or equal to MAXIMUM.

HTBasic for Windows

SLIDER Widget

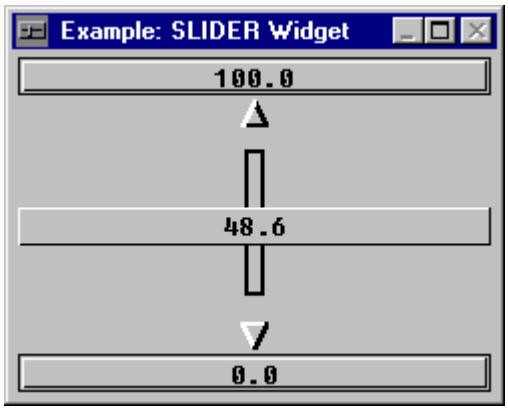
SLIDER Widget

Provides a slider pad that moves in a trough from a minimum value to a maximum value

---

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See *SLIDER Widget* for a program that provides a display similar to that shown above.

NOTE

See the following programs for other examples using the SLIDER widget:

Bar Meter Limits
Environmental Chamber
Hammer Game
METER/SLIDER Widgets
PID Controller
SLIDER Test

Attributes

See SLIDER Widget Attributes for the SLIDER widget attribute list.

Remarks

The SLIDER widget is used to display and set a numeric value with a slider pad that moves in a trough from a minimum value to a maximum value.The SLIDER widget is used to display and set a numeric value in the style of an audio mixing-panel slider. For a vertical slider, the maximum value is at the top. For a horizontal slider, the maximum value is at the right.

You can use the mouse to move the slidebar in the center of the SLIDER widget. The value on the slidebar will change to reflect its position along the scale of minimum to maximum, and the event generated in this program will DISP that value.

You can change maximum range as required, using the MINIMUM and MAXIMUM attributes. You can click the mouse on the arrows at the ends of the scale and the slidebar will move in the appropriate direction by the MINOR INCREMENT setting. Click on the scale between an arrow and the slidebar, and the slidebar will move in that direction by the MAJOR INCREMENT setting.

You can change the MINOR INCREMENT and MAJOR INCREMENT setting from their defaults of 1 and 10 as required. If you click at the ends of the scale, the SLIDER only increments once. You can use AUTO REPEAT to allows the SLIDER to increment as long as the mouse button is depressed (or the slider runs into the end). You can also move the slidebar using these keys:

Key(s)	Action
Up Arrow/Right Arrow	Increment by MINOR INCREMENT
Down Arrow/Left Arrow	Decrement by MINOR INCREMENT
Shift-Up Arrow/Shift-Right Arrow	Move to MAXIMUM
Shift-Down Arrow/Shift-Left Arrow	Move to MINIMUM
Prev	Decrement by MAJOR INCREMENT

Increment by MAJOR INCREMENT

If DIRECT MOVE is set to 1, when you click on the space between an arrow and the slidebar the slidebar will move directly to that position. DIRECT MOVE disables MAJOR INCREMENT.

You can set LOGARITHMIC to 1 and the slidebar will increment in a logarithmic fashion. You can set ORIENTATION to HORIZONTAL as an alternative to the default orientation of VERTICAL, and the SLIDEBAR will have a horizontal format. You can turn the limits boxes on and off and specify (to an extent) numeric formats.

Events

Events for the SLIDER widget are:

- CHANGED
- DONE
- SYSTEM MENU

CHANGED

This event is generated every time the user changes the VALUE by clicking on arrows or in trough or by clicking and dragging the slider.

DONE

This event is generated only when the user releases the mouse button.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

SLIDER Widget Attributes

SLIDER Widget Attributes

Attribute	Type	Allowed Values	Default
AUTO REPEAT	Numeric	0,1	0
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
COLUMNS	Numeric	Any positive integer	7
DECIMAL DIGITS	Numeric	0 through 15	1
DIRECT MOVE	Numeric	0,1	0 (use major and minor increments)
DISPLAY BACKGROUND	Numeric	Any valid BASIC pen value (0-255)	System dependent [2]
EXPONENTIAL FORMAT	Numeric	0,1	0 (fixed format)
FONT	String	*Font* [1]	
HEIGHT	Numeri	Any integer number (of pixels)	Autosizing

	c		
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
INSIDE_HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE_WIDTH	Numeric	Any integer number (of pixels)	Varies
LOGARITHMIC	Numeric	0, 1	0 (linear)
MAJOR INCREMENT	Numeric	Any real number	10
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MAXIMUM	Numeric	Any real number	100
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MINIMUM	Numeric	Any real number	0
MINOR INCREMENT	Numeric	Any real number	1

MOVABLE [5]	Numeric	0,1	1 (movable, click title bar and drag)
ORIENTATION	String	"UP", "VERTICAL", "RIGHT", "HORIZONTAL"	"VERTICAL" (max value at top)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SHOW LIMITS	Numeric	0,1	1
SHOW VALUE	Numeric	0,1	1
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or string array	Any valid string or string array	Null
SYSTEM MENU	Nu	0 to number of items in system menu	0

COUNT [5]	meric		
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TITLE [5]	String	Any valid string	SLIDER
USER DATA	String	Any valid string	None
VALUE	Numeric	Any real number	1
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

HTBasic for Windows

CONFIG File Sections - SLIDER Widget Delays

When clicking and holding on an arrow of a SLIDER widget, you can specify when repeat starts and the interval between repeats. SLIDER parameters that can be customized are listed in the following table.

Parameter	Default (msec)	Function
initial_delay	500	Delay before auto-repeat starts
repeat_delay	100	Delay between auto-repeat scrolls

STACKING ORDER

Widgets Only. The value of STACKING ORDER represents the depth of a particular widget in the widget hierarchy on the screen. The STACKING ORDER of the widget that is popped to the top on the screen is 0, the widget just below the top widget has a STACKING ORDER of 1, below that 2, and so on.

STACKING ORDER applies to the hierarchy inside the child stack as well as the hierarchy level-0, parent, and child stacks. To bring a level-0 widget to the top on the screen, do one of the following:

- Set the value of its STACKING ORDER attribute to 0 with a CONTROL command

or

- Click on the title bar of the widget

After you do this, the other level-0 widgets on the screen will have their STACKING ORDER attributes set to a new value by the widget management software - a value that represents their new order in the hierarchy.

The STACKING ORDER attribute value is dynamic, based on what is happening with other widgets on the screen. Therefore, you should not assume each widget's STACKING ORDER value. Always get the value of STACKING ORDER just before you need to use it in your program. New (just created) widgets have a STACKING ORDER of 0 and therefore will appear on top of the previously created widgets.

Transient widgets have special implications for the STACKING ORDER attribute. A transient widget cannot be "stacked" under its parent widget. If you attempt to set the STACKING ORDER of a parent of a transient widget to 0, the transient widget's STACKING ORDER will be set to 0 and the parent's STACKING ORDER will be set to 1.

However, it is possible for a transient widget's STACKING ORDER value to be more than one number less than its parent's: the level-0 transient widget may be "on top" of the screen (STACKING ORDER = 0), while its parent is covered completely with other level-0 widgets and has a STACKING ORDER of, say, 9.

The HTBasic for Windows widget management software is in control of each

widget's STACKING ORDER at all times, except when you set the STACKING ORDER from your program. The widget management software maintains a hierarchy of widgets. The nature of that hierarchy is:

- There is one and only one widget "on top" on the screen at any one time. That is, with a STACKING ORDER of 0. There are level-0, parent widgets each of which can contain any number of child widgets

- The widget management software maintains a STACKING ORDER hierarchy for the level-0 windows and a separate STACKING ORDER hierarchy for the child widgets within each level-0 widget.

- If you set the STACKING ORDER of a child widget to zero, but it is visually occluded by another level-0 widget, the child widget will not pop to the top position on the screen. It will pop to the top of its parent widget's hierarchy, but will remain occluded by the level-0 widget.

- The visual appearance of the screen may or may not accurately represent the state of the widget management software's hierarchy.

For example, if two level-0 widgets are on the screen and they do not visually overlap, it will appear as if they both have a STACKING ORDER of 0. However, only one of the widgets on the screen can have a STACKING ORDER of 0. If you move the STACKING ORDER-of-1 widget toward the STACKING ORDER-of-0 widget, it will slide "under" the true "on top" widget.

HTBasic for Windows

STATUS Command

STATUS

Returns the value of each listed attribute for the specified widget.

NOTE

*Click the **>>** bar for additional information on the STATUS command.*

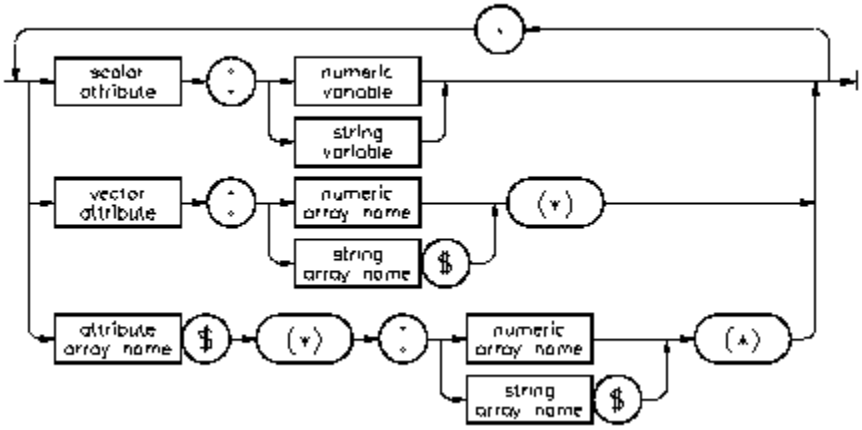
Legal Usage	Keyboard Executable	Yes
	Programmable	Yes
	In an IF...THEN...	Yes

Example Statements

```
STATUS @Field1;RETURN ("VALUE":Limit$)
STATUS @Input;RETURN ("VALUE":Setpoint)
STATUS @Meter1;RETURN ("LOW LIMIT":Low_lim,
       "HIGH LIMIT":High_lim)
```

Syntax

literal form of the return attribute list:



Item	Description	Range
attribute array name	An array of single-valued widget attributes. Both the attribute array and either the	depends on widget

	numeric array or string array must have identical dimensions.	
numeric array name	array of values to receive the current value(s) of the specified attribute(s)	depends on attribute
numeric variable	variable to receive the current value of the specified attribute	depends on attribute
scalar attribute	a single-valued widget attribute (string expression)	depends on widget
string array name	array of values to receive the current value(s) of the specified attribute(s)	depends on attribute
string variable	variable to receive the current value of the specified attribute	depends on attribute
vector attribute	a multi-valued widget attribute (string expression)	depends on widget
widget handle	name identifying an existing widget	any valid name

STATUS Command (continued)

Description

Each widget has a variety of attributes that control its appearance and behavior. The STATUS command is used to query the value of a widget attribute. The widget must have been created previously using an ASSIGN command. Attributes are either scalar (may contain a single value) or vector (may be assigned an array of values) and have values of either numeric or string type.

A shorthand method is available that permits you to query values of several attributes without naming them individually on the STATUS statement. (Only scalar attributes may be queried with this shorthand method.)

- You store all the attributes in a string array and supply an array to receive attribute values.
- Then, when you supply the array names to the STATUS statement, the value of each attribute named in each element of the attribute array will be returned in the corresponding element of the value array
- Elements of the attribute array that contain nothing, or nothing but blanks, will be ignored and the corresponding element of the value array will remain unchanged.

Since widget handles are equivalent to I/O path names, you may use the STATUS statement to query the value of registers, which provide information about the widget. For widgets, Status Register 0 and Status Register 1 are defined.

Status Register 0 is defined for all I/O paths. For example:

```
STATUS @Io_path,0;Numeric_var
```

For widgets, this returns a 5 to numeric_var (5 means @Io_path is a widget).

Status Register 1 is defined for all I/O paths assigned to a device. For example:

STATUS @Pb_12,1;Numeric_var

For widgets, this will return a 6 to numeric_var (6 means @Pb_12 is a device associated with the internal graphics CRT).

Any status register greater than 1 will cause *Error 155 - Bad interface register* number. Using ENTER, OUTPUT, TRANSFER, etc., (all other commands associated with I/O paths assigned to devices) generates *Error 170 - I/O operation not allowed*

AUTOSCROLL

When the AUTOSCROLL attribute is set to "1," the operator can enter characters in the STRING (which are entered into the VALUE attribute) up to the value of MAXIMUM LENGTH. As soon as the operator enters more characters than the STRING widget will hold, the text scrolls to the left.

COLUMNS

This attribute specifies the number of columns (number of character cells) that will fit across the STRING widget's WIDTH. See INTERDEPENDENT ATTRIBUTES for details on how COLUMNS interacts with FONT and WIDTH.

CONVERT CASE

This attribute provides for case conversion of all letters. "NONE" provides for no case conversion. "UPSHIFT" converts all letters to capital letters. "DOWNSHIFT" converts all capital letters to lower-case letters. Pre-existing text will not be changed.

COPY SELECTION

Set-Only Attribute. COPY SELECTION copies the current selection to the clipboard. The current selection is the text portion of the STRING that has been highlighted (selected) by pressing, dragging, and releasing the left mouse button.

CURSOR POSITION

The CURSOR POSITION attribute specifies the position of the cursor in the current line. The first character's position is 0.

CUT SELECTION

Set-Only Attribute. CUT SELECTION copies the current selection into the clipboard and deletes it from the STRING widget. The current selection is the text portion of the STRING that has been highlighted (selected) by pressing, dragging, and releasing the left mouse button.

DELETE SELECTION

Set-Only Attribute. DELETE SELECTION deletes the current selection and does not copy the current selection into the clipboard. The current selection is the text portion of the STRING that has been highlighted (selected) by pressing, dragging, and releasing the left mouse button.

FONT

Specifies the font to be used for the text (the string value of the VALUE attribute) in the STRING widget. See INTERDEPENDENT ATTRIBUTES for details on how FONT interacts with COLUMNS, ROWS, HEIGHT, and WIDTH. For a description of allowed fonts, see Specifying FONT Attribute Values.

HORIZONTAL SCROLL

Set-Only Attribute. HORIZONTAL SCROLL specifies how many horizontal characters to scroll the contents of the widget display. Negative numbers scroll left, while positive numbers scroll right.

LINE

Return-Only Attribute. LINE returns the text of the current line, including any <CR> <LF>.

LINE LENGTH

Return-Only Attribute. LINE LENGTH returns the length of the current line, including any <CR> <LF>.

LINE NUMBER

This attribute specifies which line the cursor is on, scrolling the display (if necessary) to bring that line into view. The first line number is 0.

LINES

Return-Only Attribute. LINES returns the number of lines in the widget.

MAXIMUM LENGTH

The value of this attribute specifies the number of characters that are allowed in the STRING. This includes the <CR> <LF> at the end of each line in a multiline STRING. A value of 0 means no limit.

MODIFIED

The MODIFIED attribute indicates the operator has changed the contents of STRING. When responding to the DONE/RETURN event, check the MODIFIED attribute to see if it has been modified, then set MODIFIED=0 for the next cycle.

MULTILINE

This attribute specifies whether or not STRING will accept more than one line of text. When set to "1", pressing **Return** advances one line in overwrite mode or insert one line in insert mode.

PASSWORD CHARACTER

The PASSWORD CHARACTER attribute allows you to select the character to be displayed on the view panel when you enter text. For example, if you select x as your password character, when you type a six-character string, the view panel will display "xxxxxx". This has no effect on the VALUE attribute.

PASTE

Set-Only Attribute. PASTE inserts the contents of the clipboard before the cursor position.

PEN

The text will be painted using this pen color. For a discussion of allowed pens, see Settable Pens Table.

READ FILE

Set-Only Attribute. READ FILE specifies the name of a file to read. The read file is inserted before the current cursor position. Set MULTILINE to "1" when using READ FILE if the file being read has more than one line.

ROWS

The number of rows specifies the number of character cells that make the STRING height. You specify the number of rows that you want to appear in the STRING widget. See INTERDEPENDENT ATTRIBUTES for details on how ROWS interacts with FONT and HEIGHT.

SCROLLBARS

This attribute specifies whether or not to create horizontal and vertical scrollbars automatically when the widget contains more text than can be displayed.

SEARCH

Set-Only Attribute. SEARCH searches from the current cursor position toward the end of the text for the specified string. The search is not case sensitive. That is, upper-case and lower-case characters are not viewed differently. For example, both "Now" and "now" would be found.

When a match is found, the view panel is positioned to the match. The match is highlighted and it becomes the current selection. If no match is found, *Error 59: End of file or buffer found* is generated.

SELECTION

The SELECTION attribute is the text of the current selection including any <CR><LF>s. The current selection is the contents of the view panel that have been highlighted (selected) by pressing, dragging, and releasing the left mouse button on the view panel's contents, set programmatically by CURSOR POSITION and SELECTION LENGTH, or a previous set of SELECTION.

If there is no current selection (SELECTION LENGTH is 0), setting this attribute inserts the text at the current CURSOR POSITION.

SELECTION LENGTH

SELECTION LENGTH specifies the length (in characters) of the current selection, including any <CR><LF>s. The current selection is the contents of the view panel that have been highlighted (selected) by pressing, dragging, and releasing the left mouse button on the view panel's contents.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

Arrow-Key Behavior When The STRING Widget Has The Input Focus

Vertical Arrow-Key Behavior	Horizontal Arrow-Key Behavior
move cursor and scroll, if necessary	move cursor and scroll, if necessary

TEXT LENGTH

Return-Only Attribute. TEXT LENGTH returns the number of characters in the widget. It includes <CR><LF> for every line except the last one.

VALUE

The value of this attribute is the text that will be displayed in the STRING widget. When the MULTILINE attribute is set to "1", the embedded <CR><LF> in scalar string indicates the end of the line - subsequent text starts a new line.

Each element of a string array is displayed as a separate line. When the MULTILINE attribute is set to "0", only the last line of a string array or scalar string containing <CR><LF> is displayed.

VERTICAL SCROLL

Set-Only Attribute. VERTICAL SCROLL specifies how many vertical lines to scroll the contents of the widget. Negative numbers scroll up and positive numbers scroll down.

WRITE FILE

Set-Only Attribute. WRITE FILE writes the widget's contents to the specified file.

STRING Dialog

Prompts the operator for textual input

Example Image

Example Program

See STRING Dialog for an example program that produces a display similiar to that shown above.

NOTE

See the following program for another example using the STRING dialog:

Dialogs Tests

Attributes

See STRING Dialog Attributes for the STRING dialog attribute list.

Remarks

The STRING dialog responds to the special keyboard keys as follows:

Key	Action

End	Clears from cursor to end of line
End	Clears entire line
Shift-Insert	Inserts a line, if MULTILINE:1
Shift-Delete	Deletes entire line
Insert	Toggles between insert (bar cursor) and overwrite (underline cursor)
Delete	Deletes current selection if highlighted. Otherwise, deletes character at current cursor position.
Home	Deletes from current character to last character in last line
Left Arrow, **Right Arrow,** **Up Arrow,** **Down Arrow**	Moves cursor and scrolls if necessary
Shift-Left	Moves and scrolls (if necessary) to the first character in the line
Shift-Right	Moves and scrolls (if necessary) to the last character in the line
Shift-Up	Moves and scrolls (if necessary) to the first character in the first line
Shift-Down	Moves and scrolls (if necessary) to the last character in the last line
Page Up/ **Page**	Scrolls one screen, if MULTILINE:1

Down

HTBasic for Windows

STRING Dialog Attributes

STRING Dialog Attributes

Attribute	Type	Allowed Values	Default
AUTOSCROLL	Numeric	0,1	1
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
COLUMNS	Numeric	Any positive integer	20
CONVERTCASE	String	"NONE", "UPSHIFT" , "DOWNSHIFT"	"NONE"
COPYSELECTION	Numeric	0,1	
CURSORPOSITION	Numeric	0 to "LINE LENGTH"-1	0
CUTSELECTION	Numeric	0,1	
DEFAULTBUTTON	Numeric	Any valid index into the DIALOG BUTTONS array	0
DELETESELECTION	Numeric	0,1	
DIALOGBUTTONS	String array	Any valid string array	A two-element array that contains OK and Cancel
FONT	String	*Font* [1]	

HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
HORIZONTAL SCROLL	Numeric	-32767 to 32767	
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
LINE	String	Any	Null
LINE LENGTH	Numeric	0 to 32767	0
LINE NUMBER	Numeric	0 to "LINES"-1	0
LINES	Numeric	0 to 32767	0
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MAXIMUM LENGTH	Numeric	0 to 65535	0
MINIMIZABLE	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MODIFIED	Numeric	0,1	0
MOVABLE	Numeric	0,1	1 (movable, click title bar and drag)
MULTILINE	Numeric	0,1	0
PASSWORD CHARACTER	String (a single	Any	Null

	character)		
PASTE	Numeric	0,1	
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
READ FILE	String	Existing filename	
RESIZABLE	Numeric	0,1	1 (resizable, special border appears around the dialog)
RESTORE SCREEN	Numeric	0,1	0 (do not save the screen)
ROWS	Numeric	Any positive integer	1
SCROLLBARS	Numeric	0,1	0
SEARCH	String	Any	
SELECTION	String	Any	Null
SELECTION LENGTH	Numeric	0 to 32767	0
TEXT LENGTH	Numeric	Any	0
TITLE	String	Any valid string	STRING
USER DATA	String	Any valid string	None
VALUE	String or string arra	Any	Null

	y		
VERSION	String	Any valid string	Current version
VERTICAL SCROLL	Numeric	Any	-
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
WRITE FILE	String	Valid filename	-
X	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [3]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Screen or parent work area upper-left corner

AUTOSCROLL

When the AUTOSCROLL attribute is set to "1," the operator can enter characters in the STRING (which are entered into the VALUE attribute) up to the value of MAXIMUM LENGTH. As soon as the operator enters more characters than the STRING dialog will hold, the text scrolls to the left.

COLUMNS

This attribute specifies the number of columns (number of character cells) that will fit across the STRING dialog's WIDTH. See INTERDEPENDENT ATTRIBUTES for details on how COLUMNS interacts with FONT and WIDTH.

CUT SELECTION

Set-Only Attribute. CUT SELECTION copies the current selection into the clipboard and deletes it from the STRING dialog. The current selection is the text portion of the STRING that has been highlighted (selected) by pressing, dragging, and releasing the left mouse button.

FONT

Specifies the font to be used for the text (the string value of the VALUE attribute) in the STRING dialog. See INTERDEPENDENT ATTRIBUTES for details on how FONT interacts with COLUMNS, ROWS, HEIGHT, and WIDTH. For a description of allowed fonts, see Specifying FONT Attribute Values.

HORIZONTAL SCROLL

Set-Only Attribute. HORIZONTAL SCROLL specifies how many horizontal characters to scroll the contents of the dialog display. Negative numbers scroll left, and positive numbers scroll right.

LINES

Return-Only Attribute. LINES returns the number of lines in the dialog.

ROWS

The number of rows specifies the number of character cells that make the STRING height. You specify the number of rows that you want to appear in the STRING dialog. See INTERDEPENDENT ATTRIBUTES for details on how ROWS interacts with FONT and HEIGHT.

SCROLLBARS

This attribute specifies whether or not to create horizontal and vertical scrollbars automatically when the dialog contains more text than can be displayed.

TEXT LENGTH

Return-Only Attribute. TEXT LENGTH returns the number of characters in the dialog. It includes <CR><LF> for every line except the last one.

VALUE

The value of this attribute is the text that will be displayed in the STRING dialog. When the MULTILINE attribute is set to "1", the embedded <CR><LF> in scalar string indicates the end of the line - subsequent text starts a new line.

Each element of a string array is displayed as a separate line. When the MULTILINE attribute is set to "0", only the last line of a string array or scalar string containing <CR><LF> is displayed.

VERTICAL SCROLL

Set-Only Attribute. VERTICAL SCROLL specifies how many vertical lines to scroll the contents of the dialog. Negative numbers scroll up and positive numbers scroll down.

WRITE FILE

Set-Only Attribute. WRITE FILE writes the dialog's contents to the specified file.

STRING Widget

Allows the user to enter a string of text

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image

Example Program

See *STRING Widget* for a program that creates a STRING widget with a display similar to that shown above.
See the following programs for other examples using the STRING widget.

- *Lissajous Patterns*
- *Passwords*
- *PID Controller*
- *Slot Machine*
- *STRING Editor*
- *Tic-Tac-Toe*

Attributes

See STRING Widget Attributes for the STRING widget attribute list.

Remarks

The STRING widget allows the user to enter a string of text. The STRING widget can be used as a rudimentary editor, providing file read/write functions in addition to cut, copy, delete and paste.

The CUT SELECTION, COPY SELECTION, DELETE SELECTION, and PASTE attributes are not meant to be interactive from the mouse. You must add a button or other interactive widget to create a fully interactive editor.

The STRING widget allows the user interface to accept strings of text from the user. The MULTILINE attribute allows the STRING widget to accept multiline inputs.

Cut, copy, and delete work on the current selection:

- To make a selection, click and drag the mouse on the desired area to highlight it.
- To highlight a selection that takes more than one screen To display, click on an area and drag the mouse offscreen, either above or below the window, and the highlighted area will scroll until you release the mouse button or move the cursor back on the screen.

Cut, Copy, and Paste work with a global "clipboard" buffer area shared by all widgets and applications:

- Text cut or copied from a STRING replaces the buffer contents.
- Paste copies the buffer contents to the widget or application at the current cursor position. In this way, text can be cut or copied from one STRING to another.
- Clicking the mouse over text positions the cursor at the text.

STRING responds to the special keyboard keys as follows:

Key	Action
End	Clears from cursor to end of line
Shift-End	Clears entire line
Shift-Insert	Inserts a line, if MULTILINE:1
Shift-Delete	Deletes entire line
Insert	Toggles between insert (bar cursor) and overwrite (underline cursor)
Delete	Deletes current selection if highlighted.

	Otherwise, deletes character at current cursor position.
Home	Deletes from current character to last character in last line
Left Arrow, **Right Arrow,** **Up Arrow,** **Down Arrow**	Moves cursor and scrolls if necessary
Shift-Left	Moves and scrolls (if necessary) to the first character in the line
Shift-Right	Moves and scrolls (if necessary) to the last character in the line
Shift-Up	Moves and scrolls (if necessary) to the first character in the first line
Shift-Down	Moves and scrolls (if necessary) to the last character in the last line
Page Up/ **Page Down**	Scrolls one screen, if MULTILINE:1

Events

Events for the STRING widget are:

- DONE
- KEYSTROKE
- RETURN
- SYSTEM MENU

DONE

This event is generated when the widget loses the focus.

KEYSTROKE

When the focus is in the STRING widget, this event is generated as you depress a key that changes the contents of STRING.

RETURN

This event is generated when the operator presses the **Return** key.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

STRING Widget Attributes

STRING Widget Attributes

Attribute	Type	Allowed Values	Default
AUTOSCROLL	Numeric	0,1	1
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
COLUMNS	Numeric	Any positive integer	20
CONVERTCASE	String	"NONE", "UPSHIFT" , "DOWNSHIFT"	"NONE"
COPYSELECTION	Numeric	0,1	
CURSORPOSITION	Numeric	0 to "LINE LENGTH"-1	0
CUTSELECTION	Numeric	0,1	
DELETESELECTION	Numeric	0,1	
FONT	String	*Font* [1]	

HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
HORIZONTAL SCROLL	Numeric	-32767 to 32767	
INSIDE_HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE_WIDTH	Numeric	Any integer number (of pixels)	Varies
LINE	String	Any	Null
LINE_LENGTH	Numeric	0 to 32767	0
LINE_NUMBER	Numeric	0 to "LINES"-1	0
LINES	Numeric	0 to 32767	0
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title bar)
MAXIMUM LENGTH	Numeric	0 to 65535	0
MINIMIZA	N	0,1	0 (not

BLE [5]	u m eri c		minimizable, button does not appear in title bar)
MODIFIED	N u m eri c	0,1	0
MOVABLE [5]	N u m eri c	0,1	1 (movable, click title bar and drag)
MULTILINE	N u m eri c	0,1	0
PASSWORD CHARACTER	St rin g (a si ng le ch ar ac ter)	Any	Null
PASTE	N u m eri c	0,1	
PEN	N u m eri c	Any valid BASIC pen number (0-255)	System dependent [2]
READ FILE	St rin g	Existing filename	-
RESIZABLE [5]	N u m eri c	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	N u	0,1	0 (do not restore the

	meric		screen)
ROWS	Numeric	Any positive integer	1
SCROLLBARS	Numeric	0,1	0
SEARCH	String	Any	-
SELECTION	String	Any	Null
SELECTION LENGTH	Numeric	0 to 32767	0
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or string array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TEXT LENGTH	Nu	Any	0

	meric		
TITLE [5]	String	Any valid string	STRING
USER DATA	String	Any valid string	None
VALUE	String or string array	Any	Null
VERSION	String	Any valid string	Current version
VERTICAL SCROLL	Numeric	Any	-
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
WRITE FILE	String	Valid filename	-
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the

widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

AUTOSCALE

The AUTOSCALE function is limited to the Y axis on the STRIPCHART. If a data point is added whose Y value is beyond the edge of the chart and the AUTOSCALE mode for the Y axis is on (value of 1), the window is rescaled to make room for that data point.

This mode should not be used in time-critical STRIPCHART applications because it can cause a delay at an unpredictable time while the entire chart is redrawn. The use of this limited AUTOSCALE feature on a STRIPCHART does not eliminate the need to set the ORIGIN and RANGE attributes to reasonable initial values for both X and Y axes.

AUTOTICK

This attribute automatically enables the setting of tick attributes to reasonable values. If you set TICK SPACING, AUTOTICK is turned off (set to 0).

AXIS FONT

You can specify the font to be used for the axis label with this attribute. The default value of the AXIS FONT attribute is not the system font but " " or null, which implicitly specifies an small English font.

AXIS LABEL

The string value of this attribute appears adjacent to the axis.

BIG NUMBERING

This attributes a bigger font for the axis labels when set to 1. If set to 0, the font will be changed to the default, smaller size.

COMPRESS X, COMPRESS Y

The COMPRESS X/COMPRESS Y attributes store data with or without compression. In the default (1) mode, data is stored with compression, and values read by the VALUES or MARKERS attributes may be different than the original values set. In the uncompressed (0) mode, data is stored without compression and values read are the same as values set.

In the "SHARED X" mode, X data is interdependent with other traces. Therefore, "COMPRESS X" must be set after the "SHARED X" attribute is set if you want to store the original X data in "SHARED X" mode.
The trace is erased when either "COMPRESS X" or "COMPRESS Y" is changed.

CURRENT AXIS

The string value of this attribute specifies the axis to which all of the "Per Axis" attributes apply. Therefore, to set the attributes of an individual axis, you must first set the value of CURRENT AXIS to that individual axis.

The allowed values for CURRENT AXIS are "X" and "Y". The X axis is always the one that scrolls. If the ORIENTATION attribute is set to "LEFT" or "RIGHT", the X axis will be the vertical one.

CURRENT TRACE

The integer value of this attribute specifies the trace to which all of the "Per Trace" attributes apply. Therefore, to set the attributes of an individual trace, you must first set the value of CURRENT TRACE to the number of that individual trace.

The value of CURRENT TRACE sets which trace is read or written by STATUS or CONTROL calls. If the value of CURRENT TRACE is 0 and then you set a "Per Trace" attribute with a CONTROL statement, the widget will set that attribute in all traces. If you try to get the value of any "Per Trace" attribute with a STATUS statement while CURRENT TRACE is 0, an error will be generated.

DIGITS

The DIGITS attribute tells the chart how many character positions to make room for in the axis numbering windows. For example, if you want a time format like "12:40:51", set the value of DIGITS to 8. This attribute is needed only for the X axis. The number of character positions for the Y axis is set automatically, because the Y axis does not scroll.

GRID PEN

Specifies the color of a grid to make it visible. Its default value is 0 (black), which is not visible if TRACE BACKGROUND is set to 0.

LOG TICKS

This attribute sets the style of ticks to be used for log-scaled axes. There are five styles available, ranging from style 1, the most sparse to style 5, the most dense. In this table, major ticks are shown in larger type and minor ticks are shown as smaller subscripts. The minor ticks are displayed only if the MINOR TICKS attribute value is non-zero.

Style Number	Log Ticks at:
style 1	1_{3} 10
style 2	1_{2} 5 10
style 3	$1_{2}3_{4}5_{6}7_{8}9$ 10
style 4	$1_{1.5}2_{2.5}$ $3_{4}5_{6}7_{8}9$ 10
style 5	$1_{1.5}$ $2_{3}4$ $5_{6}7_{8}9$ 10

LOGARITHMIC

If the value is 1, the axis is set to logarithmic scale. If the value is 0, the axis is set to linear scale.

MARKER

Determines the type and number of trace markers.

- "NONE" displays no markers on the trace.
- "ONE" displays one marker on the trace.
- "TWO" displays two markers on the trace.
- "DELTA" provides two markers - the difference between them is displayed below the trace.
- "RATIO" provides two markers - the ratio between them is displayed below the trace.

NOTE

For mouse operation, the left button grabs and moves the closer marker to the cursor. The right button grabs and moves the further marker to the cursor when these are two markers on the trace.

MARKER1 TRACE, MARKER2 TRACE

This attribute specifies the trace on which the markers are displayed. Trace selection can be changed by clicking on the graph's MARKER button.

MARKER1 X, MARKER1 Y, MARKER2 X, MARKER2 Y

This attribute specifies the X and Y position of the 2 markers.

Marker1/Marker2

The **Spacebar** or **Return** key changes the trace to which the marker tracks.

Delta/Ratio

The **Space** or **Return** key toggles between the Delta display and Delta ratio displays.

Marker1/Marker2 Value Fields

The cursor keys move the markers. When the marker is tracking to a trace, the **Up** or **Right** keys move the marker toward the end of the data, and the **Down** or **Left** keys move the marker toward the top of the data. The **Home** key moves the marker to the data point that is nearest to the upper left corner on the trace area.

When there are no traces for the marker to move toward, the cursor keys move the marker in each direction and the **Home** key (or your keyboard equivalent) moves the marker to the upper-left corner on the trace area.

MINIMUM SCROLL

The value of MINIMUM SCROLL sets the minimum amount that the plot surface is scrolled, in units of percent of plot area width. MINIMUM SCROLL provides flexible and efficient scrolling capabilities to the STRIPCHART. Since scrolling consumes computer time, you may want to have the window scroll a greater distance than the distance it takes to fit a new data point on the chart.

MINOR TICKS

The value of this attribute specifies the number of minor ticks per major tick interval. A value of 1 puts one minor tick halfway between major ticks. A value of 2 puts minor ticks at 1/3 and 2/3 of the way between major ticks, etc. With a value of 0, there are no minor ticks. If LOGARITHMIC mode is on (value of 1), the value of MINOR TICKS must be non-zero for minor ticks to be displayed.

NUMBER FORMAT

This attribute offers a choice between different ways to format the numbers that are drawn adjacent to major tick marks. The time formats allow almost any representation of a time value you might want. In all cases, the numbers are drawn using only as much precision as is needed to distinctly identify the value of each tick mark.

For example, if the "HOURS" format is chosen and the TICK SPACING attribute is set to 60 (seconds), the numbers will be drawn as HH:MM. The seconds field is dropped because it is not needed in order to accurately represent the tick values.

Allowed Values For NUMBER FORMAT

Allowed Value	Description
AUTO	Selects between FIXED, ENGINEERING, and SCIENTIFIC to represent the number with the smallest number of characters
FIXED	Displays fixed-point number format
SCIENTIFIC	Displays standard scientific notation, for example: 3.38E-12
ENGINEERING	Uses engineering format with the exponent represented by the following single letter designators (listed with their equivalent scientific notation exponents):

Letter Designator	Equivalent Scientific Notation
m	E-03
u	E-06
n	E-09
p	E-12
f	E-15
a	E-18
k	E+03

M	E+06
G	E+09
T	E+12
P	E+15
E	E+18

The following time formats all assume that the POINT LOCATION values are in seconds, as given by the HTBasic TIMEDATE function. The STRIPCHART does the conversion from seconds to the formats in the following table.

NOTE

The allowed values for the "MINUTES", "HOURS", and "DAYS" formats do not wrap around as the values for "CLOCK12" and "CLOCK24" do. Instead, the most significant fields accumulate indefinitely. For example, if the value of NUMBER FORMAT is "DAYS", the seconds, minutes, and hours fields will have maximum values of 59, 59, and 23, respectively, but the days field can accumulate indefinitely.

Time-related Allowed Values for NUMBER FORMAT

Allowed Value	Format Displayed in Tick Labels
MINUTES	minutes:seconds
HOURS	hours:minutes:seconds
DAYS	days:hours:minutes:seconds
CLOCK12	hours:minutes:secondsA (AM) OR hours:minutes:secondsP (PM) - The display wraps around to 1:00 after 12:59. - Any tick labels representing negative time values will display the time as if the clock were turned back from midnight. For example, a tick at -20 seconds would be labeled as 11:59:40P.
CLOCK24	hours:minutes:seconds

- The display wraps around to 00:00 after 23:59.

- Any tick labels representing negative time values will display the time as if the clock were turned back from midnight. For example, a tick at -20 seconds would be labeled as 23:59:40.

ORIENTATION

The value of this attribute sets the orientation of the chart in its window. This transformation is the LAST to be applied, so it affects all of the other directional attributes equally. "UP" is the standard orientation in which:

- The data origin is in the lower-left corner of the display window
- The STRIPCHART data-surface scrolls only to the left
- The X axis is horizontal
- The Y axis is vertical

If the ORIENTATION is changed, the normal top of the chart now points in the new direction. For example, an orientation of "LEFT" causes:

- The data origin to be in the lower right corner of the display window
- The STRIPCHART surface to scroll only down
- The X axis to be vertical
- The Y axis to be horizontal

If you want to flip one of the axes over, use a negative value for the RANGE attribute and choose an appropriate value for the ORIGIN attribute.

ORIGIN

The application of the value of this attribute depends on the value of the CURRENT AXIS attribute. If CURRENT AXIS is "X", the value of ORIGIN specifies the location in the coordinate plane of the left edge of the viewing window (assuming that ORIENTATION is "UP" or "VERTICAL").

If CURRENT AXIS is "Y", the value of ORIGIN specifies the location in the coordinate plane of the bottom edge of the viewing window (assuming that ORIENTATION is "UP" or "VERTICAL").

If the LOGARITHMIC attribute is turned on (value is 1), the value of ORIGIN is required to be non-zero and positive. When you set the value of LOGARITHMIC to 1, if the current value of ORIGIN is zero or negative, the value of ORIGIN will be set to 1.

The value of ORIGIN can be changed by the operator if the USER SCROLL attribute value is set to 1. In this case, the new value can be read back from ORIGIN by the HTBasic program.

PEN

The PEN attribute specifies the pen color to used for all labeling and axis numbering.

POINT CAPACITY

The POINT CAPACITY value specifies the total number of points that will be stored for each trace in the STRIPCHART. The value of POINT CAPACITY is also the size of the data "container" or "internal buffer" in memory. Memory consumption of STRIPCHART varies according to the setting of "COMPRESS X" and "COMPRESS Y". This table shows the levels of memory consumption according to different settings.

COMPRESS X/Y Memory Consumption

COMPRESS X	COMPRESS Y	bytes/ point
1	1	4 (default)
1	0	10
0	1	10
0	0	16

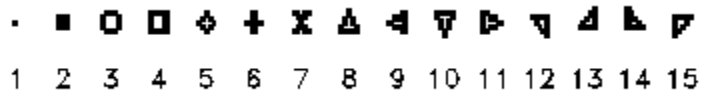
For example, if you set the value of POINT CAPACITY to 1000, you have reserved 4000 bytes of memory as an internal buffer. Setting the SHARED X attribute value to 1 reduces the amount of memory used by the STRIPCHART.

POINT LOCATION

The value of POINT LOCATION is used as the X coordinate for a point (a point is an X-Y pair). Whenever you set the values of the VALUES attribute (for multiple traces) or the value of the VALUE attribute (the Per Trace attribute), you provide the Y coordinate for the point.

POINT SYMBOL

This attribute selects the symbol to be used to display data points. Normally on a color display, color is used to distinguish traces. However, on a monochrome display, something else is needed. The point symbols are drawn using a 6x6 grid of pixels, with the following available shapes:



If you set the value of POINT SYMBOL to be the negative of the desired symbol value, (for example, to -5 for diamond symbols), no line segments will be drawn between the point symbols. On a color display, POINT SYMBOL defaults to off (value of 0). On a monochrome display, the default is the value of CURRENT TRACE plus one.

RANGE

The value of RANGE specifies the range of data values to be shown, starting from the ORIGIN value. The value of RANGE specifies the relative location in the coordinate plane of the right (if CURRENT AXIS is "X") or top (if CURRENT AXIS is "Y") edge of the viewing window (assuming that the value of the ORIENTATION attribute is "UP" or "VERTICAL").

The value of RANGE can be either positive or negative and the chart will be drawn appropriately. As such, this attribute can be used to effectively flip the chart over. If the LOGARITHMIC attribute is turned on (value is 1), the value of RANGE specifies the number of decades that the window will display. Do not set the value of RANGE to 0. Doing this will cause an error.

SHARED X

This mode conserves memory in the case where all STRIPCHART traces are updated in lock step and the X coordinates of each new set of points are identical. Only one X coordinate is maintained for each set of Y coordinates. When in this mode, the POINT LOCATION attribute is used to set the X coordinate and the VALUES attribute is used to input an array of data values, one for each trace.

SHOW GRID

This attribute displays solid and dotted grid lines. If SHOW GRID is set to "NONE", no grid is displayed. If SHOW GRID is set to "MAJOR", a solid grid is displayed.

SHOW NUMBERING

This attribute turns on or off the numbering window for the current axis.

SHOW TICKS

This attribute turns on or off the edge scale windows for the current axis.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

Arrow-Key Behavior When STRIPCHART Widget Has Input Focus

Vertical Arrow-Key Behavior	Horizontal Arrow-Key Behavior
scroll	scroll

TICK ORIGIN

The value of this attribute sets the starting point for major ticks. If ticks are to be placed at values of 5, 15, 25, etc., TICK ORIGIN is set to 5 and TICK SPACING is set to 10. If the value of LOGARITHMIC is 1, the value of TICK ORIGIN is forced to 0 and cannot be changed.

TICK SPACING

The value of TICK SPACING specifies the spacing between major ticks. Do not set the value of TICK SPACING to 0. Doing this will cause an error. If the value of LOGARITHMIC is 1, the value of TICK SPACING is forced to 1 and cannot be changed. If you set TICK SPACING, AUTOTICK is turned off (set to 0).

TRACE BACKGROUND

The value of this attribute is the pen used for the background of the trace area and the trace label area. For a discussion of allowed pens, see Settable Pens Table.

TRACE COUNT

The value of this attribute is the number of traces in the graph.
This attribute significantly affects the computer's memory usage.

TRACE LABEL

The value of this attribute specifies the name of the trace to be used in the key drawn along the bottom of the graph.

TRACE PEN

The value of this attribute specifies the pen color that will be used for the current trace. For a discussion of allowed pens, see Settable Pens Table.

TRACE VISIBLE

This attribute controls the visibility of the trace and its label. You can still add data to an invisible trace. If you set the value of TRACE VISIBLE to 1, the trace and its label will be visible. If you set the value of TRACE VISIBLE to 0, the trace and its label will disappear.

USER SCROLL

This attribute functions only on the X axis. If the value is 1, a scrollbar appears that scrolls the X axis. If the value is 0, no scrollbar appears.
If the operator scrolls the window away from the leading edge (the point where new data is being added to the chart), the data in the window stops scrolling and the window stays in the same location over the data. Later, if the operator scrolls back to the leading edge of the data, the data again starts scrolling in the window.

VALID POINTS

The value of this attribute represents the number of points that have been added to the trace. The widget will automatically increment the value of VALID POINTS when points are added to the trace. To clear the trace, set the value of VALID POINTS to 0. The value of VALID POINTS will always be less than or equal to the value of POINT CAPACITY.

VALUE

The value of VALUE is a single Y axis value to be paired with the most recently set X axis value of the POINT LOCATION attribute to form a data point on the trace that is specified by the CURRENT TRACE attribute.

VALUES

The STRIPCHART uses the data in this array for Y values to add points to the traces, using the most recent X value. The X value is set using the "POINT LOCATION" attribute. The first value in the VALUES array is used for trace 1, the second for trace 2, etc. The process stops when it runs out of traces or array values.

HTBasic for Windows

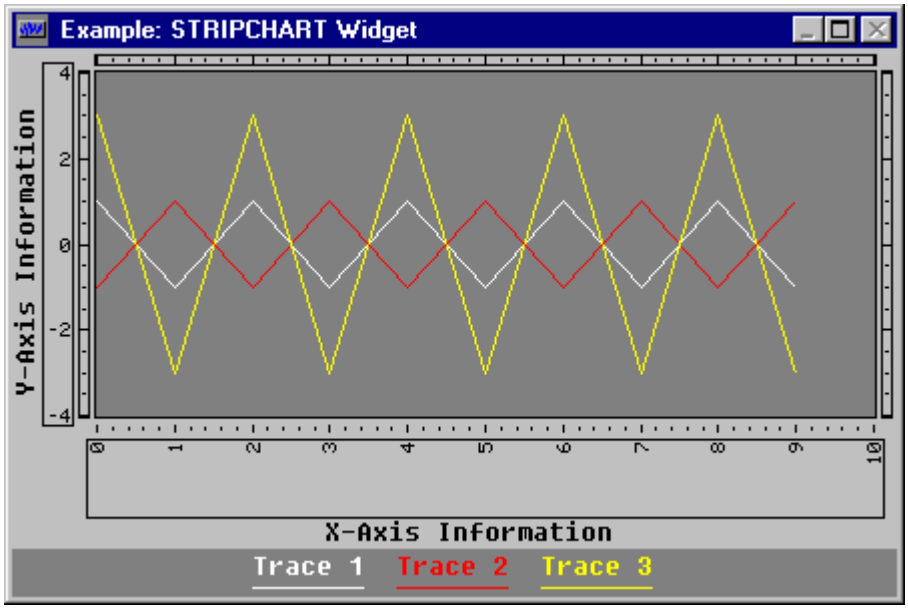
SRIPCHART Widget

STRIPCHART Widget

Displays data in a two-dimensional scrolling graph

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See *STRIPCHART Widget* for a program that creates a STRIPCHART widget with three traces, and traces a pattern of triangle waves similar to the display shown above.

See the following programs for other examples using the STRIPCHART widget:

Environmental Chamber
PID Controller
STRIPCHART (counter)
STRIPCHART (rescale)

STRIPCHART (scrolling)
STRIPCHART (sine waves)

Attributes

See STRIPCHART Widget Attributes for the STRIPCHART widget attribute list.

Remarks

The STRIPCHART widget displays data in a two-dimensional scrolling graph. It models a physical stripchart.

The STRIPCHART widget allows you to trace up to 100 different streams of values on a scrolling graphics display. You can set ranges and tick styles, labels and numeric formats along both the X and Y axes, and different colors for the traces. The STRIPCHART widget has three classes of attributes:

- Those that apply to the entire widget ("global" attributes)
- Those that apply to either the X or Y axis
- Those that apply to an individual trace

Attributes that apply to the entire widget are ORIENTATION (VERTICAL or HORIZONTAL), TRACE BACKGROUND (sets the PEN color of the trace display area) and TRACE COUNT which gives the number of traces that you want to display.

MINIMUM SCROLL tells how much the traces scrolling across the STRIPCHART display will scroll at one time, in terms of a percent of the X-axis. A small value (for example, one that defines one pixel of the width) will give a very smooth scroll, but usually is not useful. For example, if your X-axis range is 10, and you perform updates in increments of 1, you should set MINIMUM SCROLL to 10 (percent).

You can write each trace independently, but it is usually more practical to set up the STRIPCHART so that you can write a single X value (using the POINT LOCATION attribute) and then update all the traces at one time using an array (using the VALUES attribute). The SHARED X attribute is set to 1 to allow updating all the traces in parallel.

Use the CURRENT AXIS attribute to select one of the two STRIPCHART display axes, X or Y. When an axis is selected, you can define its ORIGIN and RANGE (if the axis is LOGARITHMIC-scaled), an AXIS LABEL, and various formats for TICK SPACING and appearance, various NUMBER FORMATs (AUTOmatic, FIXED, ENGINEERING, and SCIENTIFIC, and time formats such as CLOCK12).

If you do not want to scale the axes, set the axes to AUTOSCALE and STRIPCHART will adjust the axes to the data displayed. If you do not want to set numbering and tick marks, set SHOW NUMBERING and SHOW TICKS to 0

The SHOW GRID attribute allows a grid to be displayed on the graphic surface. The MARKER attribute allows "markers" to be displayed on one or two traces. The markers can be moved with the mouse and interrogated for the trace values.

MARKER can be set to NONE (no markers), ONE marker, TWO markers, DELTA (difference between X and Y marker values), and RATIO (ratio of marker values). When markers are turned on, the appropriate marker values are displayed at the bottom of the STRIPCHART.

Traces to be marked are designated with the attributes MARKER1 TRACE and MARKER2 TRACE. The positions of the markers on the trace can be set or queried using the attributes MARKER1 X / MARKER1 Y, and MARKER2 X / MARKER2 Y. You can trap marker movements with the MARKER1 MOVED and MARKER2 MOVED events.

Events

Events for the STRIPCHART widget are:

- MARKER1 MOVED, MARKER2 MOVED
- SCROLLED
- SYSTEM MENU

MARKER1 MOVED, MARKER2 MOVED

This event is generated when a marker is moved.

SCROLLED

This event is generated when the operator scrolls the STRIPCHART with the internal scrollbar. That is, when the mouse button is released after moving the scrollbar slider or after moving the scrollbar slider with the arrow keys from the keyboard.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

STRIPCHART Widget Attributes

STRIPCHART Widget Attributes

Attribute	Type	Allowed Values	Default
AUTOSCALE [3]	Numeric	0,1	0
AUTOTICK [3]	Numeric	0,1	1
AXIS LABEL [3]	String	Any string	Null string
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
BIG NUMBERING	Numeric	0,1	0
BORDER [2]	Numeric	0,1	1
COMPRESS X, COMPRESS Y [4]	Numeric	0,1	1
CURRENT AXIS	String	"X", "Y"	"X"
CURRENT TRACE	Numeric	0 to value of TRACE COUNT	1
DIGITS [3]	Numeric	1 to 18	6

GRID PEN	Numeric	0 to 255	0
HEIGHT	Numeric	Any integer number (of pixels)	300 pixels
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
LOG TICKS [3]	Numeric	1 to 5	5
LOGARITHMIC [3]	Numeric	0,1	0
MARKER	String	"NONE", "ONE", "TWO", "DELTA", "RATIO"	"NONE"
MARKER1 TRACE, MARKER2 TRACE	Numeric	1 to value of TRACE COUNT	1
MARKER1 X, MARKER1 Y, MARKER2 X, MARKER2 Y	Numeric	Any real or integer number	0
MAXIMIZA	N	0,1	1

BLE [6]	u m eri c		(maximizable, button appears in title bar)
MINIMIZABLE [6]	N u m eri c	0,1	0 (not minimizable, button does not appear in title bar)
MINIMUM SCROLL	N u m eri c	0 to 100	5
MINOR TICKS [4]	N u m eri c	0 to 100	Auto adjusted
MOVABLE [6]	N u m eri c	0,1	1 (movable, click on title bar and drag)
NUMBER FORMAT [3]	St rin g	"AUTO", "FIXED", "ENGINEERING", "SCIENTIFIC", "MINUTES", "HOURS", "DAYS", "CLOCK12", "CLOCK24"	"AUTO"
ORIENTATION	St rin g	"UP", "DOWN", "LEFT", "RIGHT"	"UP"
ORIGIN [3]	N u m eri c	Any real number	0.0
PEN	N u m eri c	0 to 255	System Dependent [1]
POINT CAPACITY [4]	N u m eri c	0 to the system's memory capacity	100
POINT LOCATIO	N u	Any real or integer number	1

N	meric		
POINT SYMBOL [4]	Numeric	0 to 15	- Color monitor: 0 (off) - Monochrome monitor: (value of CURRENT TRACE + 1)
RANGE [3]	Numeric	Any non-zero real number	1
RESIZABLE [6]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SHARED X	Numeric	0,1	0
SHOW GRID	String	"NONE", "MAJOR", "MINOR"	"NONE"
SHOW NUMBERING [3]	Numeric	0,1	1
SHOW TICKS [3]	Numeric	0,1	1
STACKING ORDER	Numeric	0 to number of siblings [5]	0
SYSTEM MENU [6]	String or St	Any valid string or string array	Null

	string array		
SYSTEM MENU COUNT [6]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [6]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TICK ORIGIN [3]	Numeric	Any real number	0
TICK SPACING [3]	Numeric	Any non-zero real number	1
TITLE [6]	String	Any valid string	STRIPCHART
TRACE BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
TRACE COUNT	Numeric	1 to 100	4
TRACE LABEL [4]	String	Any string	Null string
TRACE PEN [4]	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
TRACE VISIBLE [4]	Numeri	0,1	1

	c		
USER DATA	String	Any valid string	None
USER SCROLL [3]	Numeric	0,1	0
VALID POINTS [4]	Numeric	0 to the value of POINT CAPACITY	0
VALUE [4]	Numeric	Any real or integer number	Empty
VALUES	Numeric array	Any real or integer numbers	Empty array
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	400 pixels
X	Numeric	Any integer, number of pixels from 0,0 [7]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [7]	Autoplacement

[1] Read the CONFIG file or query for the default with a STATUS command

[2] Child widget only

[3] *This is a STRIPCHART Per Axis attribute.* To set or get this Per Axis attribute, you must first set the value of CURRENT AXIS to the axis for which you want to set or get this attribute.

[4] *This is a STRIPCHART Per Trace attribute*. To set or get this Per Trace attribute, you must first set the value of CURRENT TRACE to the trace for which you want to set or get this attribute. If you set CURRENT TRACE to 0, the specific Per Trace attribute will be set for all traces.

[5] A sibling is another child widget with the same parent, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[6] For level-0 widgets only

[7] Screen or parent work area upper-left corner

***CREATE**

Set-Only Attribute. *CREATE is used to create the specified widget and associate it with the previously specified "*NAME". If the previously specified "*NAME" is already associated with a widget, the create fails. If the SYSTEM widget is a child widget, all root widgets through the SYSTEM widget will appear as children of the SYSTEM widget's parent. Each new widget in a SYSTEM widget must have a unique name. For example:

```
 ASSIGN @A TO WIDGET "SYSTEM"
CONTROL @A;SET("*NAME":"Panel1","*CREATE":"PANEL")
CONTROL @A;SET("*NAME":"Panel1/Meter1","*CREATE":"METER")
                    .
                    .
                    .
CONTROL @A;SET("*NAME":"Panel1","VISIBLE"1)
CONTROL @A;SET("*NAME":"Panel1/Meter1","VALUE":50.1,
                                        "ARC WIDTH":10)
```

*EVENT NAME FILTER

This attribute specifies the name of the event to retrieve from the "*QUEUED EVENTS" or "*FLUSH QUEUED EVENTS".

*EVENT WIDGET FILTER

This attribute specifies the "*NAME" of the widget for which to retrieve "*QUEUED EVENTS" or "*FLUSH QUEUED EVENTS".

*FLUSH QUEUED EVENTS

Read-Only Attribute. *FLUSH QUEUED EVENTS returns "0", if queuing is off. When queuing is on, it deletes all queued events specified by the "*EVENT WIDGET FILTER" and "*EVENT NAME FILTER" attributes from the queue and returns the number deleted.

*LOAD

Set-Only Attribute. *LOAD loads, creates, and initializes the widgets named in the specified screen file (created by the SCREEN BUILDER). The *LOAD attribute can only be used with level-0 SYSTEM widgets.

*NAME

This attribute is used to specify the name of the widget you want to access inside a SYSTEM widget. All subsequent attributes in any STATUS or CONTROL statements will be applied to the named widget until an attribute that begins with "*" is specified, at which point "*NAME" must be respecified. The SYSTEM widget will accept names that do not exist to allow creation of new widgets using the "*CREATE" attribute.

When "*NAME" is used in a STATUS statement and the associated string variable is not NULL, the contents of that string variable will actually be used to SET the current widget name. This allows you to use a single statement to both specify the widget to query and the attribute to retrieve. However, if the string variable associated with "*NAME" in a STATUS statement is NULL, the name of the current widget is returned. For example:

```
ASSIGN @A TO WIDGET "SYSTEM":SET("*LOAD":"screen")
CONTROL @A;SET("*NAME":"Panel1","VISIBLE":1)
CONTROL @A;SET("*NAME":"Panel1/Meter1","VALUE":50.1,
                "ARC WIDTH":10)
          .
          .
          .
P$="Panel1/Meter1"
STATUS @A;RETURN("*NAME":P$,"NEEDLE WIDTH":I)
```

*QUEUE EVENTS

This attribute specifies whether or not to maintain a queue of events generated by the managed widgets. An event queue is required when either multiple events managed by the SYSTEM widget have the same event handler or the SYSTEM widget manages a number of widgets that have the same event name.

The event queue prevents the loss of events generated while servicing others. The only limit on queue size is the available memory. When event queuing is enabled, you must empty the queue with some combination of the "*QUEUED EVENT" and "*FLUSH QUEUED EVENTS" attributes. Setting "*QUEUE EVENTS" to "0" destroys an existing event queue.

*QUEUED EVENT

Read-Only Attribute. *QUEUED EVENT returns a two-element string array in which the first element is the widget name (*NAME), and the second element is the event name from the triggered ON EVENT statement.

When queuing is on, it returns the oldest event specified by the "*EVENT WIDGET FILTER" and the "*EVENT WIDGET NAME" attributes, and removes that event from the queue. When queuing is on and none of the specified events are in the queue, NULL strings are returned.

When queuing is on and the queued event is "SYSTEM MENU", the string returned is actually "SYSTEM MENU:n", where n is the offset of the system menu item that generated the event.

*QUEUED EVENTS

Read-Only Attribute. *QUEUED EVENTS returns the number of events currently in the event queue.

*WIDGET NAMES

Read-Only Attribute. *WIDGET NAMES returns the names of all widgets contained in the SYSTEM widget.

*WIDGETS

Read-Only Attribute. *WIDGETS returns the number of widgets contained in the SYSTEM widget.

SYSTEM MENU

Level-0 Widgets Only. The SYSTEM MENU attribute creates a system menu box in the upper-left corner of the title bar. The SYSTEM MENU will contain the equivalent of a MENU BUTTON for each non-null element in the string array, and all elements after the first null string are ignored. Only one MENU BUTTON is created if the type is a scalar string. The label(s) for the MENU BUTTON(s) are the contents of the string or string array.

SYSTEM MENU COUNT

Level-0 Widgets Only. Return-Only Attribute. The SYSTEM MENU COUNT attribute returns the number of items currently in the system menu.

SYSTEM MENU EVENT

Level-0 Widgets Only. Return-Only Attribute. The SYSTEM MENU EVENT attribute returns the offset of the system menu item that generated the last SYSTEM MENU event (the first item is always offset 0). If there is only one item in the SYSTEM MENU, there is no need to query this attribute.

HTBasic for Windows

SSTEM Widget

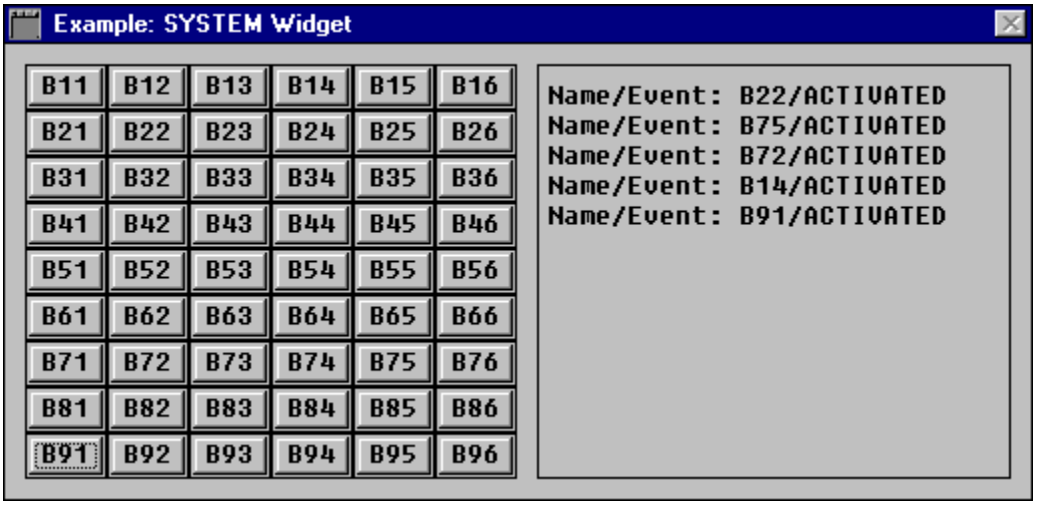
SYSTEM Widget

Acts as a container and manager for the widgets specified in files generated by the Screen Builder application

---

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See SYSTEM Widget as a Child for an example program that generates a display similar to that shown above.

NOTE

See the following programs for other examples using the SYSTEM widget:

Bomb Squad (Using Screen Builder)
Bomb Squad (Using SYSTEM Widget)
SYSTEM Widget as a Child
SYSTEM Widget Event Disabling
SYSTEM Widget Event Handler

SYSTEM Widget Event Queuing

Attributes

See SYSTEM Widget Attributes for the SYSTEM widget attribute list.

Remarks

The SYSTEM widget acts as a container and manager for the widgets specified in files generated by the Screen Builder application. The SYSTEM widget is a unique widget in that it does not create a window. Rather, it manages other widgets. The SYSTEM widget can also manage a queue of events generated by the widgets it contains.

The SYSTEM widget provides attributes to select and modify the attributes of the widgets it manages or to manage their events. It also provides the ability to programmatically create new widgets.

The convention used to name the widgets of a SYSTEM widget is similar to that used for hierarchical file systems. The top-level widgets have simple names while their child widgets always have the names of their parents affixed to the beginning of their simple names. For example:

"Meter1"
"Panel0/Panel2/Meter0"

The SYSTEM widget does not have any of the generic attributes usually associated with other widgets. The SYSTEM widget has only the following SYSTEM-specific attributes which have the "*" prefix. Any other attribute will be forwarded to one of the managed widgets as specified by the most recent *NAME attribute.

The Screen Builder is used to generate a file that describes a user interface, and the SYSTEM widget is used to integrate this file into a program. The SYSTEM widget's *only* function is to help control a system of widgets.

The SYSTEM widget has no attributes that control its appearance. However, it has a set of specific attributes that are used to change the widgets it controls. Because the special nature of these attributes, their names are prefixed with an asterisk. For example, "*NAME" is the SYSTEM widget attribute used to select one of its widgets. You can use the SYSTEM widget in one of two ways:

- Using Screen Builder (*LOAD)
- Using SYSTEM Widget Alone (*CREATE)

Using Screen Builder (*LOAD)

The Screen Builder generates a file containing the names and specified attributes of the widgets defined with that application. This file is read by the SYSTEM widget to connect it to a program, using the "*LOAD" attribute:

```
ASSIGN @Sys TO WIDGET "SYSTEM";SET("*LOAD":<filename>)
```

This must be done every time you run the program. The file must be present, or the program will stop with an error. Once the Screen Builder file has been loaded, any of the widgets 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)
```

Once a widget has been selected with *NAME, all subsequent widget CONTROL and STATUS statements are passed through the SYSTEM widget and affect that widget and that widget only until a different *NAME is set. Similarly, if, for example, "Main" has a child widget named "Meter2", that child can be accessed with:

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

This scheme implies that widgets - at least those defined by the Screen Builder application - can now be defined in string (or string array) variables:

```
W$="Main/Bar1" STATUS @Sys;RETURN("*NAME":W$, "X":I, "Y":J)
```

Events can be trapped through the SYSTEM widget as well, although all you can specify is the event type to be trapped, not the particular event source:

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

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)
```

By default, the SYSTEM widget only keeps track of the latest event. That is, if you do not read the events as they happen, you will lose all but the last event.

In some cases this is not a problem. The user clicks on a button, the program traps the event and does something, and then the user does something else. However, in user interfaces that contain multiple widgets that have the same event name, you must arrange for the SYSTEM widget to queue up events.

You can set up event queuing by setting the *QUEUE EVENTS attribute to 1. The SYSTEM widget will then store all events that happen within it in the order they occur. You can determine the number of events in the queue with the *QUEUED EVENTS attribute:

```
STATUS @Sys;RETURN ("*QUEUED EVENTS":Numevents)
```

and then read each event in sequence using the *QUEUED EVENT attribute. The SYSTEM widget has three attributes that have similar names:

- QUEUE EVENTS -- to set event handling
- QUEUED EVENT -- to get the last event
- QUEUED EVENTS -- to get the number of events in the queue

You may want to get all events of a certain type at the same time, however, to allow you to sort them out. The SYSTEM widget has two attributes to allow you to specify which widget and event type will be read from the queue (leaving all the others remaining):

- EVENT NAME FILTER
- EVENT WIDGET FILTER

Using SYSTEM Widget Alone (*CREATE)

The SYSTEM widget can also be used without a Screen Builder file by using the *CREATE attribute. For example, to create a PANEL named "Main", execute the following (*NAME must be specified before *CREATE can create the widget):

```
CONTROL @Sys; SET("*NAME":"Main","*CREATE":"PANEL")
```

You can specify widget pathnames to define a parent-child relationship. For example, to create a child CLOCK widget named "Clock", execute:

```
CONTROL @Sys; SET("*NAME":"Main/Clock","*CREATE":"CLOCK")
```

Creating an entire user interface with the SYSTEM widget using *CREATE can have some disadvantages. For example, if all widgets are different, it takes more work than building the user interface in the normal fashion. The advantage of doing this, however, is when you have a set of widgets that are exactly the same - say, a grid of PUSHBUTTON, STRING, or LABEL widgets. Since you define the widgets using names given as strings in a string variable or string array you avoid the normal HTBasic for Windows restriction of having a widget handle for each widget and a separate ON EVENT handler for each event.

Events

The SYSTEM widget has no events of its own, but inherits the events defined by its widgets. ON EVENT branches for each of the contained widgets must be set up using the SYSTEM widget's I/O path.

For widgets that share the same event name (e.g., "CHANGED"), only one ON EVENT branch can be established for all of them. Once the branch has been taken, use the "*QUEUED EVENT" attribute to determine which widget and/or which event caused the branch.

HTBasic for Windows

SYSTEM Widget Attributes

SYSTEM Widget Attributes

Attribute	Type	Allowed Values	Default
*CREATE	String	Valid widget name	Null
*EVENT NAME FILTER	String	Existing event name	Null
*EVENT WIDGET FILTER	String	Existing widget *NAME	Null
*FLUSH QUEUED EVENTS	Long integer	Number of managed widgets	Null
*LOAD	String	Any Screen Builder file name	Null
*NAME	String	User-defined name	Null
*QUEUE EVENTS	Numeric	0 or 1	0
*QUEUED EVENT	Two-element string array	Element 1: widget name Element 2: widget event	0
*QUEUED EVENTS	Long integer	Number of events in queue	0
*WIDGET NAMES	String array	Names of managed widgets	Null
*WIDGETS	Numeric	Number of managed widgets	Null

Screen Builder

An application that allows the user to build a user interface by selecting from a menu of graphic objects (dialogs and widgets) and editing on the screen.

HTBasic for Windows

Screen Builder Application

This topic describes the Screen Builder Application, including:

- Screen Builder Overview
- Screen Builder Initial Condition
- Screen Builder Interactive Features
- Screen Builder Menu
- Screen Building Process
- Screen Builder Example

Click the item name to view each of these topic items.

HTBasic for Windows

Screen Builder Example

Introduction

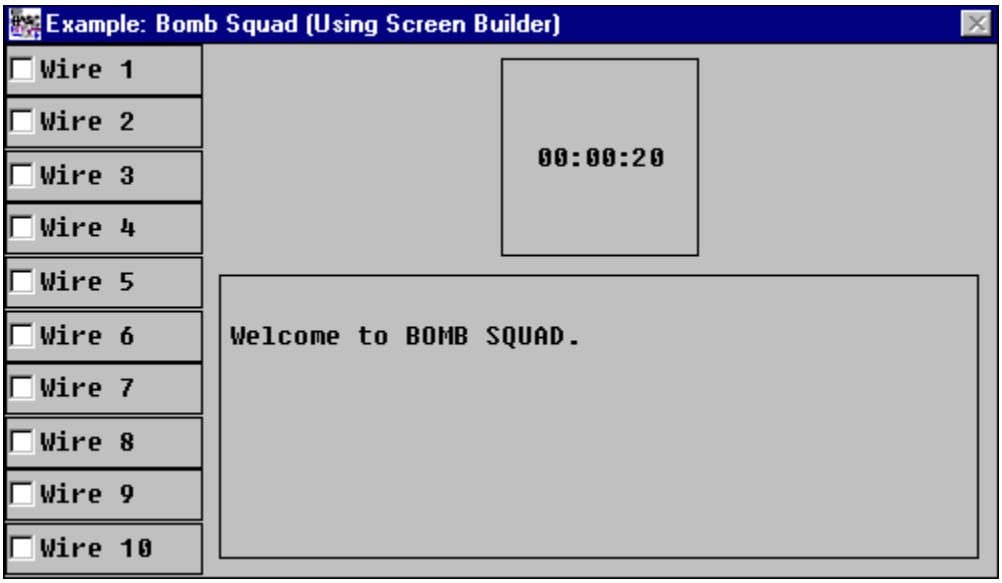
This topic shows an example of building a user interface using the Screen Builder Application and the SYSTEM widget. The program builds a game called "Bomb Squad" in which you pretend you have a bomb wired with ten wires.

- If you disconnect the correct four wires, the bomb is disarmed
- If you disconnect the wrong two wires, the bomb explodes
- The other four wires are inactive and disconnecting causes no action

In addition, the bomb is ticking as you try to disarm it. If you do not disarm it in time, the bomb explodes. A typical display to begin the game follows. Click the **>>** bar to view the description and steps of this example.

NOTE

See Bomb Squad (Using SYSTEM Widget) for a similar example using the SYSTEM widget to build the interface.



HTBasic for Windows

Screen Builder Example

Steps to Build the Interface

We will build the interface for this game with ten TOGGLEBUTTONs (the wires), a CLOCK widget set to TIMER mode (the bomb timer), and a PRINTER widget (for user feedback). We will use the Screen Builder application to design the user interface. The steps to build the interface follow. Click the step title or the **>>** bar to view the description and steps of this example.

HTBasic for Windows

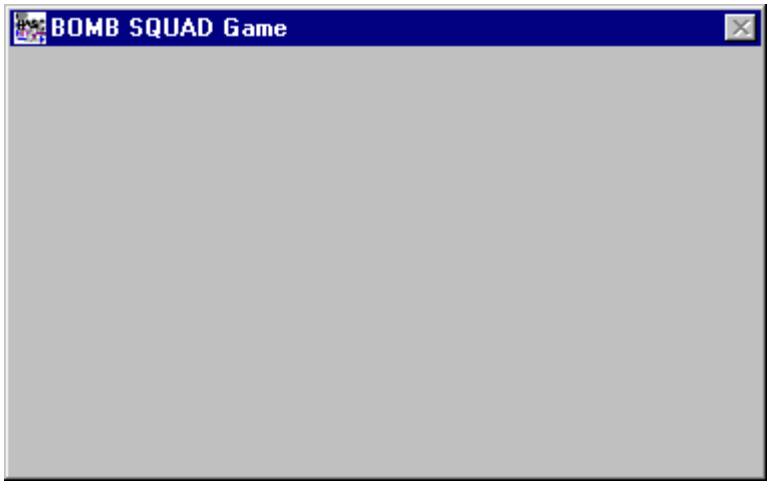
Screen Builder Example - Start the Screen Builder Application

Step 1. Start the Screen Builder Application

- Configure the parent PANEL widget from the one provided by the Screen Builder
- Select "Re-name" in the Widget menu and enter "Main" as the new name
- Select "Edit" in the Widget menu and set the following attributes. Click the **Close** button after setting each attribute.

  - TITLE to "BOMB SQUAD Game"
  - MAXIMIZABLE and RESIZABLE to FALSE
  - SYSTEM MENU to the string "Quit"

A typical display of the Screen Builder output (the modified PANEL widget) follows.
Click the **>>** bar to go to Step 2.

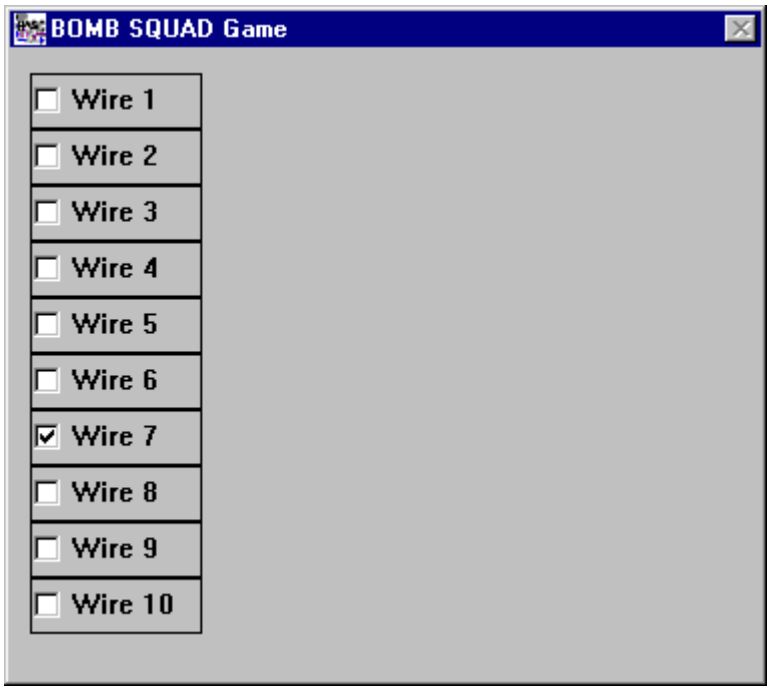


HTBasic for Windows

Screen Builder Example - Create Ten TOGGLEBUTTONs

Step 2. Create Ten TOGGLEBUTTONs

- Create the first button by clicking on the TOGGLEBUTTON icon, selecting "Main" as its parent and "T1" as its name.
- Create the other nine TOGGLEBUTTONs by choosing "Copy" in the Widget menu and selecting "Main/T1" as the widget to copy. When the Copy dialog appears, request nine copies in the Y-direction with zero spacing, beginning with the name "T2".
- Use the "Edit" menu entry to set the LABEL attribute of the ten TOGGLEBUTTONs with pathnames "Main/T1" through "Main/T10" to "Wire 1" through "Wire 10".A typical display.follows. Click the **>>** bar to go to Step 3.

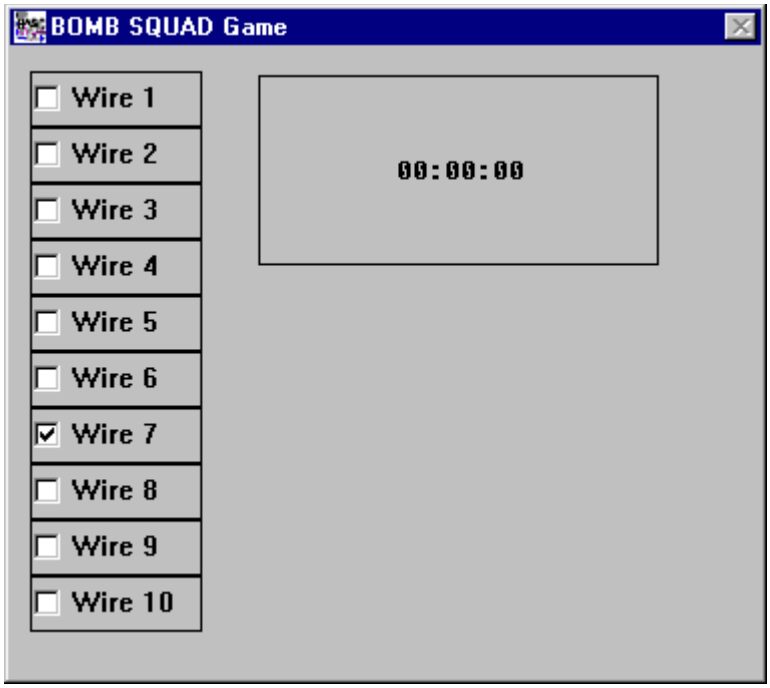


HTBasic for Windows

Screen Builder Example - Create a CLOCK Widget

Step 3: Create a CLOCK Widget

Create a CLOCK widget by clicking on the CLOCK icon and setting its name to "Clock" (pathname "Main/Clock"). Use the mouse to relocate it and make it "flatter". Set its TYPE attribute to TIMER. A typical display follows. Click the **>>** bar to go to Step 4.

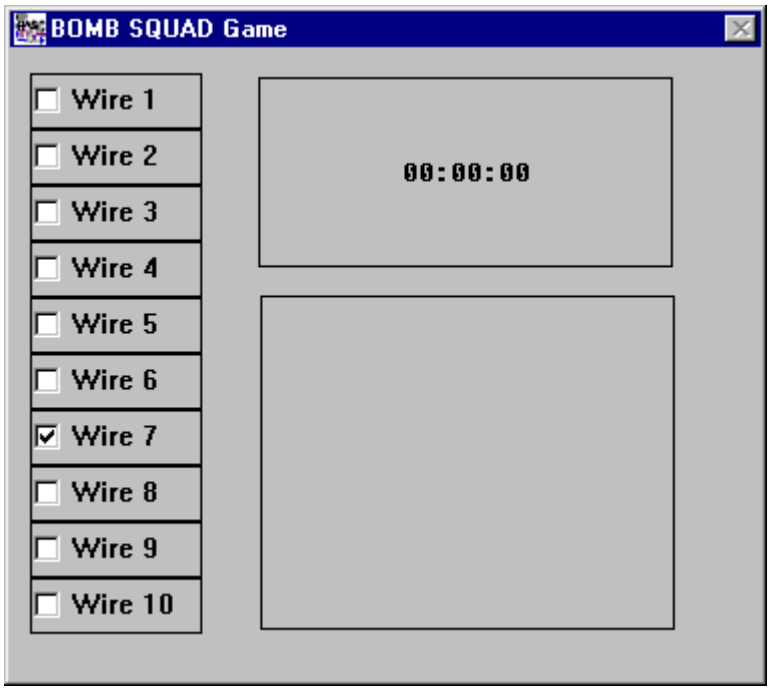


HTBasic for Windows

Screen Builder Example - Create the Final Interface

Step 4. Create the Final Interface

Create a PRINTER widget by clicking on the PRINTER icon and setting its name to "Printer" (pathname "Main/Printer"). Resize it and relocate it. A typical display of the game's final user interface.follows. Click the **>>** bar to go to Step 5.



HTBasic for Windows

Screen Builder Example - Save the Interface in a File

Step 5. Save the Interface in a File

Select the "Save As:" entry of the Screen Builder's File menu, and store this interface in the file "EX_BSQUAD.SCR". This completes the Screen Builder Application Example. See Screen Builder Application to return to the Screen Builder topics.

HTBasic for Windows

Screen Builder Initial Condition

Screen Builder Windows

When the Screen Builder is started, two windows are created and brought to the foreground. The window to the left, named "HTBasic Screen Builder", called the "main window", gets the focus and contains an array of icons representing all widgets available for screen building.

The window to the right, named HTBasic Screen Builder Output, contains "Panel0". This is a PANEL widget generated by the application as a starting point for designing a screen. Most screens consist of a panel populated with widgets. However, you can modify, or even delete, "Panel0" if you choose.

If you attempt to close or exit the Screen Builder and your current screen has not yet been saved, you will be asked to confirm your Close selection.

NOTE

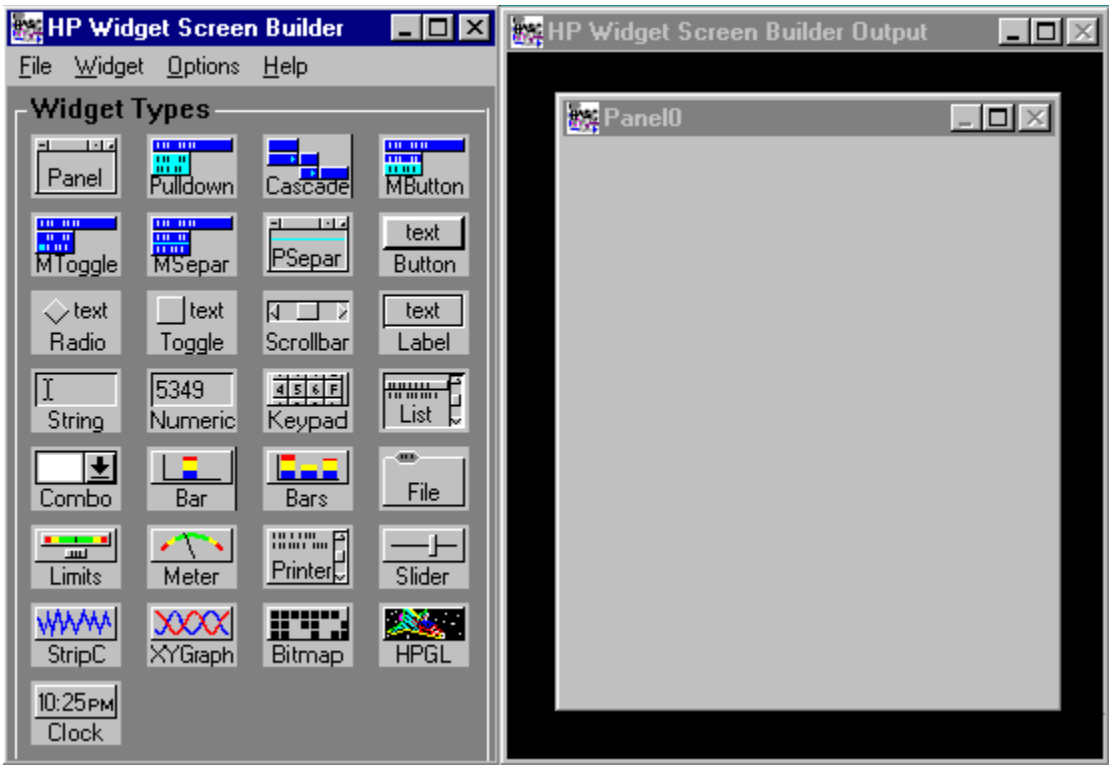
The Screen Builder only has knowledge of widgets specified in the "screen builder widget order" section of the CONFIG file. If you have modified the CONFIG file to not order some widgets, the icons for those widgets will not be shown.

NOTE

You can click the right mouse button on a widget icon to display the help file topic for the widget.For example, clicking the right mouse button on the Bar icon displays the help file description for the BAR widget.

Screen Builder Initial Display

The following figure shows the Screen Builder (default) initial display.


HP Widget Screen Builder
File
Widget
Options
Help
Widget Types
Panel
Pulldown
Cascade
MButton
MToggle
MSepar
PSepar
text
Button
text
Radio
text
Toggle
Scrollbar
text
Label
String
5349
Numeric
Keypad
List
Combo
Bar
Bars
File
Limits
Meter
Printer
Slider
StripC
XYGraph
Bitmap
HPGL
10:25PM
Clock
HP Widget Screen Builder Output
Panel0

HTBasic for Windows

Screen Builder Interactive Features

Introduction

Most of the Screen Builder's functions are accessed using the pulldown menus on the main window. However, some are also accessible by pointing and clicking with the mouse buttons, thus allowing you to work faster.

Moving/Resizing Widgets

For instance, you may begin to add a METER widget to the "working" panel by selecting the "Create..." button on the "Widget" menu, or by clicking once on the METER icon found on the main window. Once a widget is created, a process which involves assigning a parent widget (or possibly none) and a name to it, the widget's position and size can be interactively modified as follows:

- To move a widget, place the cursor in the widget's center (generating a cross-arrow shape), hold down the left mouse button, and drag the widget to the panel or screen location of choice.
- To resize a widget, move the cursor to the widget's edge until the cursor shape changes to a resize cursor, then click and drag the widget's edge. Release when it is the size you want.

It is possible to move or resize a widget outside the boundaries of its parent rendering it invisible. The only way to recover that widget is to change its X, Y, WIDTH, or HEIGHT attribute through the "Edit..." selection of the Screen Builder's "Widget" menu.

Modifying Widget Attributes

To modify the attributes of a widget, move the cursor to the widget's center and double-click the left mouse button. This will bring up a window with the name of the widget, a list of attributes (once you click on the adjacent button), and a panel of options.

The previous technique cannot be used to set the attributes of certain widgets, such as PULLDOWN and CASCADE MENUs. Instead, use the "Edit..." selection of Screen Builder's "Widget" menu.

At this point, you may close that window without changing anything by clicking on the "Close" button, or you may click on the "Attributes" button to bring a list of available attributes. Clicking on any of list items will bring one or more additional dialogs,

depending on what attribute is being set.

Monitoring the Widget Editing Process

It is possible to monitor the widget editing process by bringing up the "Status" window accessible through the "Options" menu. This window, which consists of two panels, reports current information about the screen being built, and about the widget currently under the cursor. (see Screen Builder Menu).

The final state of all changes made interactively will be saved in the screen file selected through the "File" menu.

HTBasic for Windows

Screen Builder Menu

The Screen Builder Menu provides quick access to File, Widget, and Options elements. See the following table for Screen Builder Menu elements.

Since the main window is a relatively narrow window, you may need to maximize it to see these choices.

Element	Action	Description
File	**Open**	Brings up a file dialog to enter the name of an existing screen file. Current widgets managed by the Screen Builder are deleted and those specified in the screenfile are created.
	New	Delete all widgets currently managed by the Screen Builder.
	Save	Save the current state of widgets managed by the Screen Builder in the current screen file (set by previous Open, Save, or Save as).
	Save As	Same as "Save", except it always prompts for a file name.
	Exit	Leave the Screen Builder application.

Widget	**Create**	Brings up a listbox with the names of available widgets. To create a widget: 1. Select the widget type, then its parent. If the new widget is a level-0 parent, select the No Parent entry. 2. For either selection, double-click the listbox entry or click the OK button after selecting the desired listbox entry. 3. When Screen Builder requests a name,select the default version provided or overwrite the default with a name of your choosing.
	Edit	Allows the user to modify the attributes of a widget. To edit a widget: 1. Select the widget by double-clicking on the listbox entry or by clicking once to highlight its name, then clicking the OK button. 2. A window is now displayed that will allow you to modify its attributes. To access these attributes, click the "drop-down" button across from the word "Attribute".

Two editing options are also available:

Widget enabled: Allows you to interact with the widget via the mouse and keyboard traversal while in the Screen Builder application. Changes made to a widget (for example, entries into a STRING widget) will not be saved by the Screen Builder.

Attributes auto-updated: Enables the real-time tracking of the value of an attribute. The Screen Builder will periodically update the value of the attributes currently being selected. This option is designed to operate with the "Widget enabled" option.

For example, to set the value of a SLIDER widget to "50":

1 - Double-click on the SLIDER and select the "VALUE" attribute in the attribute listbox.

2 - Select the "Widget enabled" option, allowing you to interact with the SLIDER, and the "Attributes auto-updated" option, which lets you know when VALUE reaches "50".

3 - Move the SLIDER to "50" and select the "SET" button displayed on the VALUE attribute window to record the change to the SLIDER

Close Closes the window and all attribute windows currently displayed

Delete This selection displays a listbox with the names of configured widgets. Select one or more widgets to remove them from the working area.

Re-na Rename a widget.

	me	
	Copy	Duplicate a widget and all its attribute settings. In one operation, you can make multiple copies in the X and/or Y direction at a specified spacing.

Options	**Grid**	Sets an invisible grid of the specified horizontal and vertical spacing. When a widget managed by the Screen Builder is "dragged-and-dropped" or resized, it will gravitate to the nearest grid point.
	Status	Shows a "Screen" status panel and a "Current Widget" status panel. The "Screen" panel includes information about

the

cursor position relative to the top-left corner of the screen, the

current grid setting, and the number of widgets populating the

screen.

The cursor position is updated only when the Screen Builder's

main window has the focus. The "Current Widget" panel provides

information about the name of the widget, its parent, its widget

type, its size, and its origin relative to its parent.

HTBasic for Windows

Screen Builder Overview

The Screen Builder application allows you to interactively generate a screen, modify its appearance, and save it for later inclusion in a BASIC program where the associated functionality can be implemented.

One of the most valuable functions of the Screen Builder is the ability it provides the user to interactively set (and save) the attributes of the widgets used to build the screen. This significantly reduces the effort of setting the initial attributes of widgets with CONTROL statements in an HTBasic for Windows program

The output of the Screen Builder is a text file describing a complete screen hierarchy, including the (hierarchical) names of widgets and their attributes, which can be interactively set.

The text file generated by the Screen Builder is designed to be loaded by the SYSTEM widget which provides the ability of programming events for those widgets.

To start the Screen Builder application, select the Screen Builder icon in the HTBasic for Windows program group.

HTBasic for Windows

CONFIG File Sections - Screen Builder Widget Order

Widgets Must be in CONFIG File

The Screen Builder determines which widgets are available and in what order to present them by the list of widgets in this section of the CONFIG file. Any widget not listed in this section (with the exception of the menu widgets PULLDOWN MENU, CASCADE MENU, MENU BUTTON, MENU TOGGLE, and MENU SEPARATOR which are permanently loaded) will not be available to the Screen Builder.

Reordering Widget Presentation Listing

If you prefer a different presentation order in the Screen Builder (such as alphabetical) other than the default functional grouping, you can reorder the list of widgets to your preference. See Screen Builder Initial Display for the default display order of the widgets that can be reordered.

HTBasic for Windows

Screen Building Process

The Screen Building Process

The process of building a screen (for later use in an HTBasic for Windows program) is:

1. Pick instances of widgets (graphics elements) from a menu and place them on a panel or a working area
2. Move, delete, or modify those widgets to achieve the desired screen layout
3. Save the results for final use or for later retrieval and further editing

NOTE

You can click the right mouse button on a widget icon to display the help file topic for the widget.For example, clicking the right mouse button on the Bar icon displays the help file description for the BAR widget.

Screen Builder Capabilities

To provide this functionality, Screen Builder includes these capabilities:

Capability	Description
File I/O	Allows the user to write screen configurations to a file, and to read them back in.
Editing Functions	Add, delete, and copy widgets to a screen layout to change their appearance (name, size, position, etc.), and to set other widget attributes.
Status Reporting	Get instantaneous information on widgets and other editing parameters by tracking the mouse position.

HTBasic for Windows

Search Sequence for Ancillary Files

Introduction

HTBasic Plus searches for its ancillary files in a number of locations. A search will terminate as soon as a file is found, even if the file is the wrong type if it happens to have the same name as the desired file. The search sequence is the same for all ancillary files except CONFIG.

The file specifier of an ancillary file can include three portions: <*drive* >, <*directory* >, and <*filename*>. During the search process, HTBasic for Windows will break up the file specifier and use combinations of its components: <*directory*><*filename*> (drive stripped) or just the <*filename*> (directory and drive stripped).

Additionally, HTBasic for Windows allows you to customize the search sequence by specifying a system MSI in the CONFIG file. This customization allows you to place all HTBasic Plus files (except CONFIG, BPLUS.DW6, and all WI*.DLL) in a non-standard location.

Search Sequence for CONFIG File

1. CONFIG in current MSI
2. $HOME/CONFIG ($HOME is a Windows environment variable)
3. CONFIG in install location
4. /PLUS/CONFIG in current drive
5. If current drive is not the same as the installation drive, /PLUS/CONFIG in install drive.

Search Sequence for Ancillary Files That Include Directory or Drive

1. If the system MSI is specified in CONFIG, relative to that MSI (original drive stripped)
2. In the specified directory and/or drive
3. If the system MSI is specified in CONFIG, the filename in that MSI
   (original directory and drive stripped)

4. If the directory is specified, relative to the installation location
5. If the directory is specified, relative to "/PLUS" on the current drive (original drive stripped)
6. Filename in the current MSI (original directory and drive stripped)

7. Filename in the installation location for the current drive (original directory
   and drive stripped)

8. Filename in "/PLUS" on current drive (original directory and drive stripped)
9. If the current drive is not the same as the install drive, repeat Steps 2, 5, and 8 using the install drive

Search Sequence for Ancillary Files That Include Only a Filename

1. If the system MSI is specified in CONFIG, the filename in that MSI
2. Filename in current MSI
3. Filename in installation location
4. Filename in "/PLUS" on current drive
5. If the current drive is not the same as the boot drive, filename in "/PLUS" on install drive

HTBasic for Windows

Searching for CONFIG File

CONFIG File Setup

During the installation of HTBasic for Windows, SETUP installs a default CONFIG file to (typically) C:/HTBASIC/CONFIG.
SETUP makes no attempt to preserve any modifications to any existing CONFIG file(s). If SETUP finds an existing CONFIG in the installation directory, it will rename it to CONFIG.001, CONFIG.002, CONFIG.003, etc.

Searching for CONFIG

When either of the following statements is executed:

- LOAD BIN "BPLUS"
- SCRATCH A (with the BPLUS binary loaded)

HTBasic for Windows will search for the CONFIG file and read its contents to begin operation with the specified parameters. When searching for the CONFIG file, HTBasic for Windows gets the first configuration file from one of the following locations:

- CONFIG in current MSI
- If $HOME environment CONFIG variable is set, in that directory
- In the installation directory (typically C:/HTBASIC/CONFIG)

HTBasic for Windows

Settable Pens Table

This table shows the settable pens for HTBasic for Windows. The **Settable Pens** column refers to the physical pen numbers shown in the Physical Pens Table. See Logical Pens Table for logical pen numbers associated with the physical pen numbers.

System	Settable Pens
2 - 16 pen system	0-15
17 - 64 pen system	0-15, 54, 55, 62, 16-63
65 - 256 pen system	16-31, 230, 231, 238, 0-15, 32-255

HTBasic for Windows

Setting Input Focus

When you click on the mouse within the border of a dialog or widget, that dialog or widget gains the input focus. That is, any mouse clicks or keystrokes that you make will be input to the dialog or widget that has the focus. To return the focus to HTBasic for Windows, click on the HTBasic for Windows background, title bar, or frame.

When a widget receives the input focus it changes the color or intensity of its title bar and border. Similarly, when the input focus is removed from a widget, its title bar and border revert to the previous setting.

Some widgets, such as STRING and TOGGLEBUTTON, also have other ways to indicate when they have the focus. For example, a frame is drawn around a button or a cursor appears in a text field when the widget has the focus.

Programmatically Setting Focus. Using the FOCUS attribute it is possible to change focus to a particular widget or to the Basic window. For Widgets Only setting the FOCUS attribute to a value of 0 gives focus to the specified widget. A value of 1 moves focus from the Widget to the Basic window.

HTBasic for Windows

Setting Widget Attributes

Introduction

Widget attributes are set with ASSIGN or CONTROL statements. You can set the values for each attribute through a combination of keyword syntax use and attribute value specification. Attribute values are returned with the STATUS statement. You can check the value of any attribute at any time by using STATUS to return the attribute value to a variable or variable array.

Setting Widget Attributes

After you have created a widget with an ASSIGN command, as desired you can set an attribute value or change the value of any of its previously-set attributes with a CONTROL command by using the CONTROL command SET syntax. CONTROL designates the widget to be changed, while SET designates the attribute and value setting for that attribute. The CONTROL command syntax is:

CONTROL @*widget handle*;SET (*set attribute list*)

Most widget attributes allow you to SET values in the widget and to RETURN values from the widget. However, some widgets allow only setting values with SET, while others allow only RETURNing values of present settings. See the appropriate widget description for SET and RETURN capabilities of each attribute in the widget.

HTBasic for Windows

Setting/Changing Widget Attributes

This topic gives guidelines to set and change widget attributes using the CONTROL command, including:

- Setting Widget Attributes
- Choosing Widget Attribute Order
- Changing Widget Attribute Values

HTBasic for Windows

Setting/Checking Dialog Attribute Values

Attribute values are set with ASSIGN or CONTROL statements. You can set the values for each attribute through a combination of keyword syntax use and attribute value specification. Attribute values are returned with the STATUS command. You can check the value of any attribute at any time by using the STATUS keyword to return the attribute value to a variable or variable array.

HTBasic for Windows

Sharing Input Devices/Display

Since HTBasic for Windows may have to share the use of the display with other applications when running in a windows environment, it is important to know how to control those interactions. This topic gives guidelines to use input devices and display, including:

- Improving Display Appearance
- Effects on HTBasic Graphics
- Sharing Keyboard or Mouse Input

Click the item name to view each item in this topic.

HTBasic for Windows

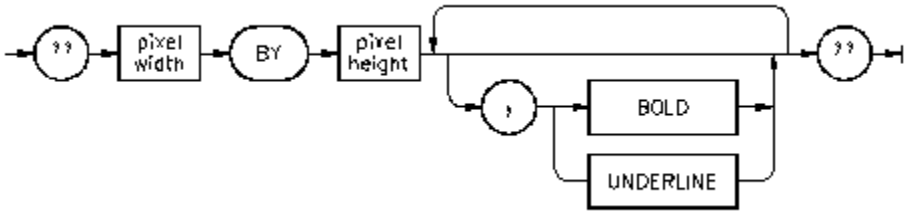
Sharing Keyboard or Mouse Input

If your program does not have any widgets displayed, the keyboard and mouse behave just as they do when HTBasic Plus is not loaded.
If you do have widgets displayed with current input focus, any inputs that are generated from the mouse, keyboard, or touchscreen are first received by HTBasic Plus.

The inputs are examined to see if they can be used for some purpose, such as string input or keyboard operation of a mouse. If the input cannot be used, it will pass the input to HTBasic to use as it deems necessary. To force the input to HTBasic for Windows, click the HTBasic for Windows title bar.

Specifying FONT Attribute Values

When the term *Font* appears in the **Allowed Values** columns, the font string you specify is actually a "suggestion" to HTBasic for Windows. HTBasic for Windows will choose the closest font to what you specify from those that are currently loaded. The FONT string syntax can include both BOLD and UNDERLINE entries, as shown. The default FONT is determined by HTBasic for Windows font as set by its *-fn* command line switch.



Steps to Create Widgets

NOTE

For an example program that creates a METER widget and a SLIDER widget, see Example: Creating a Widget..

Suggested steps to create and modify a widget follow. You can use any or all of the steps, as required for your application.

- Create/destroy the widget with ASSIGN
- Set widget attributes (as desired) with ASSIGN or CONTROL
- Query widget attributes (as desired) with STATUS
- Create event-initiated branching paths (as desired) with DISABLE EVENT, ENABLE EVENT, OFF EVENT, ON EVENT, or WAIT FOR EVENT

HTBasic for Windows

Supported Displays

HTBasic Plus works well with grayscale monochrome displays. However, items that have distinctive colors on a color display may, in some cases, be mapped to indistinguishable shades on a grayscale display. HTBasic Plus will also work on single-plane monochrome displays, but the appearance is not particularly good and this configuration is not recommended.

NOTE

We highly recommend running with graphics buffering ON (the default mode). Otherwise, HTBasic for Windows graphics will be erased whenever a widget is moved, unless the RESTORE SCREEN attribute is set for that widget.

CONFIG File Sections - System Font

The system font default depends upon CRT resolution. If the default font does not meet your needs, specify a different font in the (width)X(height) format (e.g., 8X16). The actual font used will be the nearest from those currently loaded in Windows.

HTBasic for Windows

CONFIG File Sections - System MSI (Mass Storage Is)

System MSI specifies an alternate location for HTBasic for Windows to search for ancillary files. System MSI can include a directory and/or drive specifier (e.g., "E:\SHARED"). System MSI should be set for the following reasons:

Common Files

You have common files, such as Screen Builder files, bitmaps, or HPGL files that you want to share or network.

Not in Default Location

If you move your HTBasic Plus files somewhere non-standard, you must put your CONFIG file in a standard location and in your system MSI section specify where HTBasic for Windows resides.

TITLE

Dialogs and Level-0 Widgets Only. The value of this attribute is the string that will appear in the title bar of the dialog or level-0 widget. If you set this attribute to the null string, the title bar will disappear, and the MAXIMIZABLE, MOVABLE, and SYSTEM MENU attributes are disabled.

COLUMNS

This attribute specifies the number of columns (number of character cells) that will fit across the LABEL attribute in the TOGGLEBUTTON WIDTH. See INTERDEPENDENT ATTRIBUTES for details on how COLUMNS, WIDTH, and FONT interact.

FONT

Specifies the font to be used for the text in the LABEL attribute. For a description of allowed fonts, see Specifying FONT Attribute Values.

LABEL

The value of this attribute is the text that appears to the right of the TOGGLEBUTTON.

PEN

The value of this attribute specifies the pen color used to paint the text in the LABEL attribute. For a discussion of allowed pens, see Settable Pens Table.

SENSITIVE

This attribute provides you with the capability to keep the LABEL text visible, but make the TOGGLEBUTTON unresponsive to operator input.

- If the value of SENSITIVE is 1, the widget is responsive to operator input.
- If the value of SENSITIVE is 0, the widget will not respond to operator input, and will appear "grayed out" to indicate its unresponsive status.

TAB STOP

This attribute allows you to establish "tab groups" of widgets when you fill a parent PANEL widget with child widgets.

Arrow-Key Behavior When TOGGLEBUTTON Widget Has Input Focus

Vertical Arrow-Key Behavior	Horizontal Arrow-Key Behavior
move to next widget	move to next widget

VALUE

The value of this attribute specifies the state of the TOGGLEBUTTON. If the value is 1, the TOGGLEBUTTON is depressed. If the value is 0, the TOGGLEBUTTON is released.

HTBasic for Windows

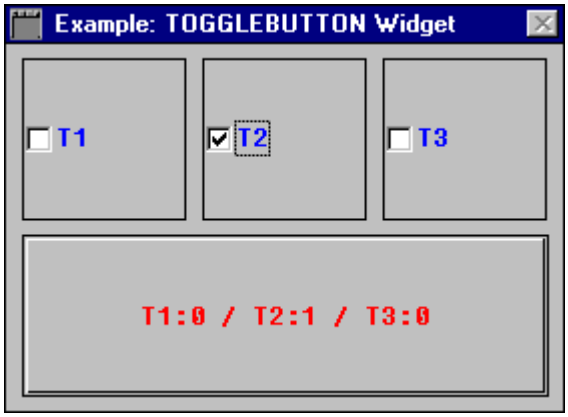
TGGLEBUTTON Widget

TOGGLEBUTTON Widget

Inputs or displays binary data

Legal Usage	Level-0 Widget:	Yes	
	Parent to:	None	
	Child of:		PANEL

Example Image



Example Program

See *TOGGLEBUTTON Widget* for an example program that generates a TOGGLEBUTTON widget with a display similar to that shown above. For this program, when you click on the TOGGLEBUTTONs, the LABEL section reflects the VALUE.

NOTE

See the following programs for other examples using the TOGGLEBUTTON widget:

Alarm Clock
*Bomb Squad (*LOAD)*
SYSTEM Widget Event Handler

Attributes

See TOGGLEBUTTON Widget Attributes for the TOGGLEBUTTON widget attribute list.

Remarks

The TOGGLEBUTTON widget is used to input or display binary data. Clicking the mouse button (or pressing the **Spacebar** or the **Return** key) while the focus is on the TOGGLEBUTTON causes a CHANGED event to be generated.

The TOGGLEBUTTON widget is similar to the PUSHBUTTON widget in that when you click on the widget, an event is generated. The TOGGLEBUTTON is useful for emulating a two-position switch, or for setting a mode of operation.

For the TOGGLEBUTTON widget, the event is called CHANGED and the TOGGLEBUTTON VALUE attribute is toggled between 0 and 1 every time you click on the widget. A small box changes highlighting to reflect the value. The default is 0.

Events

Events for the TOGGLEBUTTON widget are:

- CHANGED
- SYSTEM MENU

CHANGED

This event is generated when the user "presses" the TOGGLEBUTTON either by clicking on it with the left mouse button or by pressing the **Spacebar** or the **Return** key.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

TOGGLEBUTTON Widget Attributes

TOGGLEBUTTON Widget Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
BORDER [3]	Numeric	0,1	1
COLUMNS	Numeric	Any positive integer	10
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
INSIDE_HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE_WIDTH	Numeric	Any integer number (of pixels)	Varies
LABEL	String	Any valid string	Null string
MAXIMIZABLE [5]	Numeric	0,1	1 (maximizable, button appears in title

			bar)
MINIMIZABLE [5]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE [5]	Numeric	0,1	1 (movable, click on title bar and drag)
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE [5]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SENSITIVE	Numeric	0,1	1 (sensitive)
STACKING ORDER	Numeric	0 to number of siblings [4]	0
SYSTEM MENU [5]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [5]	Numeri	0 to number of items in system menu	0

	c		
SYSTEM MENU EVENT [5]	Numeric	0 to number of items in system menu -1	0
TAB STOP	Numeric	0,1	1
TITLE [5]	String	Any valid string	TOGGLEBUTTON
USER DATA	String	Any valid string	None
VALUE	Numeric	0,1	0
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [6]	Autoplacement

[1] For an explanation of Font and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] Child widget only

[4] A sibling is another child widget with the same parent widget, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[5] For level-0 widgets only

[6] Screen or parent work area upper-left corner

USER DATA

The contents of this attribute are left to the programmer and are never changed by HTBasic for Windows. It is a way to associate program-specific data with a specific widget or dialog.

HTBasic for Windows

Using Array Variables

Introduction

As you add attributes to a DIALOG statement, the line may get too long and may become difficult to read. Since a DIALOG statement must fit in one line, it is good practice to use array variables to set dialog attributes.

Placing Attribute Values in String Arrays

In general, you can put all attributes that require a numeric value into one string array, and then invoke them along with a matching numeric array. For example:

```
DATA "ATTRIBUTE 1","ATTRIBUTE 2","ATTRIBUTE 3","ATTRIBUTE 4","ATTRIBUTE 5"
READ An$(*)
DATA 12,32,100,6,19
READ Vn(*)
...
DIALOG Type$,Prompt$,Btn;SET (An$(*):Vn(*))
```

You could also make another set of arrays for the attributes that required string values, say As$(*) and Vs$(*), but mixing the two types is not allowed. For example:

```
DIALOG Type$,Prompt$,Btn;SET (An$(*):Vn(*),As$(*):Vs$(*))
```

Example: Using Array Variables

For example, the ERROR dialog program can also be written as follows.

```
1       ! Example: Using Array Variables
2       !
10      DIM T$[16],P$[40],A1$[20],S1$(1:2)[10],A2$[20]
20      INTEGER Btn
30      DATA "ERROR","Input caused overflow!"
40      READ T$,P$
50      DATA "DIALOG BUTTONS","ABORT","CONTINUE","DEFAULT BUTTON"
60      READ A1$,S1$(*),A2$
70      !
80      DIALOG T$,P$,Btn;SET (A1$:S1$(*),A2$:0),TIMEOUT 5
90      IF Btn=-1 THEN
```

```
100         DISP "NO ENTRY"
110     ELSE
120         DISP S1$(Btn + 1)
130     END IF
140     END
```

With these changes, room is provided to include a TIMEOUT. Adding TIMEOUT also required a change in the handling of the value of "Btn", since Btn returns a "-1" value if the dialog times out. String variables A1$ and A2$ are the names of two dialog attributes, and a string array S1$() is the name for the attribute values.

HTBasic for Windows

Using Context-Sensitive Help

This topic gives guidelines to use context-sensitive help for HTBasic for Windows, including:

- What is Context-Sensitive Help?
- CONFIG File Features
- Building Your .hlp File
- Context-Sensitive Help Example

Click the item name to view each item in this topic.

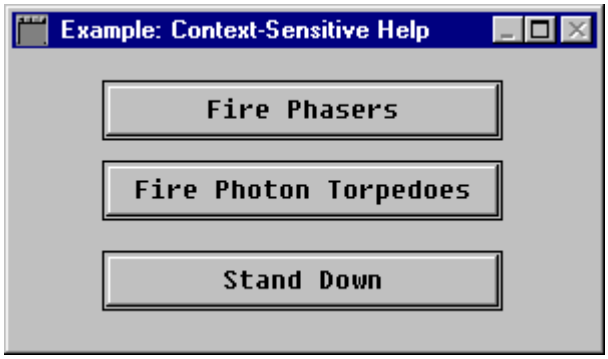
Using HTBasic Plus Topics

HTBasic for Windows

Using HELP FILE/HELP TOPIC Attributes

Introduction

To show how to use the HELP FILE and HELP TOPIC attributes, we will build a user interface using a provided Windows help file called "weapons.hlp". For this example, the user interface consists of a PANEL widget with three PUSHBUTTON widgets, as shown in the following figure. The "weapons.hlp" file is provided in HTBasic for Windows that describes the functions of the PANEL and the three PUSHBUTTONs.



Using an Example Program

The example program Context-Sensitive Help contains the code for this example. To show how this program uses context-sensitive Help, copy this program into the HTBasic for Windows Editor (see Copying Example Programs for the procedure). Then, run the program and click the right mouse button on Weapons Control, Fire Phasers, Fire Photon Torpedoes, or Stand Down. When you click on one of these displays, the appropriate Help topic appears.

Line 210 of the program sets the PANEL HELP TOPIC as "weapons.TOC", so when you click on Weapons Control, the HELP TOPIC associated with "weapons.TOC" appears. Similarly, when you click on Fire Phasers, line 280 causes the HELP TOPIC associated with "weapons.phasers" to appear, etc.

HTBasic for Windows

Using Mouse and Keyboard

Introduction

Applications developed using HTBasic Plus provide a graphical user interface that requires the interaction of the mouse and the keyboard with one or more widgets or dialogs on the screen.

Mouse/Keyboard Levels

Before discussing operations with the mouse, we need to look at how HTBasic for Windows HTBasic Plus shares the mouse (and other resources) with HTBasic for Windows. Depending on the HTBasic Plus features being used, there are three levels of mouse and keyboard behaviors:

Level	Operation	Comments
Level 1	No Dialogs or Widgets - HTBasic for Windows processes keyboard input	All mouse movement, button presses, and keyboard input is processed by BASIC.
Level 2	Dialog displayed (with or without widgets) - HTBasic for Windows processes keyboard input	The next level of resource usage is represented by the execution of a DIALOG statement in an HTBasic for Windows program. When a DIALOG statement is executed, all keyboard input is either redirected to the dialog, as in the case of a "string dialog", or ignored - except for some traversal and selection keys. In this case, the keyboard operates as with an INPUT statement, but the BASIC interpreter does not process the input.
Le	Widgets Displayed -	This level of resource sharing occurs

ve l 3	HTBasic for Windows processes keyboard input	when a widget is displayed. Keyboard input is directed to the window with the focus. If that window cannot use it, it is passed on to BASIC.

HTBasic for Windows

Using Registers

Registers Defined

Since widget handles are equivalent to I/O path names, you may use the STATUS statement to query the value of registers that provide information about the widget. Two registers are defined for widgets: Status Register 0 and Status Register 1.

Status Register 0

Status Register 0 is defined for all I/O paths. For example, for widgets, this statement will return a 5 to numeric_var (5 means @io_path is a widget).

```
STATUS @io_path, 0; numeric_var
```

Status Register 1

Status Register 1 is defined for all I/O paths assigned to a device. For example, for widgets, this statement will return a 6 to numeric_var (6 means @pb_12 is a device associated with the internal graphics CRT).

```
STATUS @pb_12, 1; numeric_var
```

Any status register greater than 1 will cause *Error 155 - Bad interface register number.* Except for ABORTIO, using ENTER, OUTPUT, TRANSFER, etc. (i.e., all other commands that are associated with I/O paths assigned to devices) generates *Error 170 - I/O operation not allowed*.

HTBasic for Windows

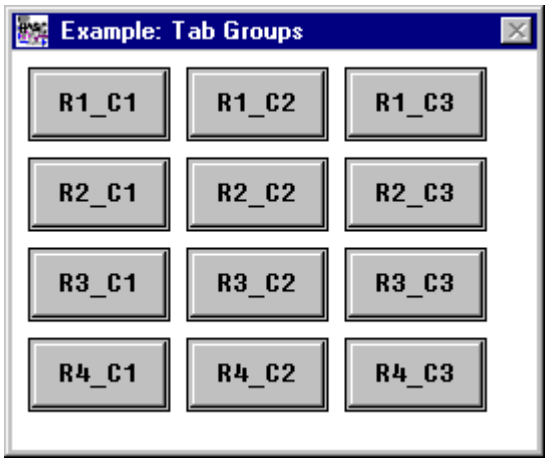
Using Tab Groups

Using Tab Groups

The **Tab** keys provide a method to cycle through the input widgets of a user interface. Most input widgets have an attribute called TAB STOP that is set to 1 by default. When you cycle through a user interface with **Tab** or **Shift-Tab**, the pointer (which may change shape based on the widget that it jumps to) will stop at an input widget if its TAB STOP attribute is 1. If a widget's TAB STOP is 0, the pointer will not stop at the widget when you press the **Tab** keys.

Example: Using Tab Groups

For example, when you copy and run the Tab Groups program (see Copying Example Programs for procedures) the following display appears For this display, the TAB STOP attribute is set to 1 for PUSHBUTTONs on the left column and is set to 0 for PUSHBUTTONs in the middle and right columns.



When you use the keyboard to cycle through this user interface, as you press **Tab** or **Shift-Tab** you can only cycle through the PUSHBUTTONs on the left (R1_C1 through R4_C1). To get to a particular PUSHBUTTON in the middle or right columns, you must tab to the appropriate PUSHBUTTON in the left column in the same row and then use the **Arrow** keys to cycle through the row.

Each row in this display constitutes a tab group. Physical layout does not define the order of the tab group. Tab group order is defined by the sequence in which the input widgets are created. If you ASSIGN one widget with a TAB STOP of 1, any following widgets of the same type with a TAB STOP of 0 are part of that tab group, until you ASSIGN another widget with a

TAB STOP of 1.

HTBasic for Windows

Copying Example Programs Using Generic Printer

Introduction

If you have installed a generic printer driver with a file as the printer destination, you can use the following procedure to copy an example Help program to the HTBasic for Windows Editor.

HTBasic for Windows

Using the CONFIG File

This topic gives guidelines to use the CONFIG file for custom configuration of HTBasic for Windows, including:

- CONFIG File Overview
- CONFIG File Sections
- Modifying/Saving CONFIG File
- Searching for CONFIG File

Click the item name to view each topic item.

HTBasic for Windows

Using Example Programs Using the plus examples Directory

NOTE

This procedure applies ONLY if you installed the example programs by selecting HTBasic Plus Examples during setup.

1. Type load bin "bplus" and press **Enter**.
2. Type HELP "HTBasic Plus Example Programs" and press **Enter**.
3. Note the *filename* listed beside the Program Name of the program you want.
   For example, the filename for the Alarm Clock program is *alarmclk*.

4. Exit the HTBasic for Windows HELP mode
5. Go to the directory where the example programs are stored. For example, if you installed the programs in the default directory, execute:

   msi "c:/program files/htbasic/plus examples"

6. Enter the example code into the editor with GET "*filename*". (*filename* from Step 3)
7. Edit and/or RUN the example program.
8. Re-save the edited program in a new *filename.*

VERSION

Return-Only Attribute. VERSION can only be used in a STATUS statement. VERSION returns the current software revision of the individual widget or dialog as a string consisting of seven characters in the format:: A.XX.XX

where: "A" is a letter
"." is always a decimal point (period)
"X" is a digit

VISIBLE

This attribute provides a flag that determines whether or not the widget or dialog is visible to the user. For a widget or dialog to be visible, its VISIBLE attribute value must be 1 and, if it is a child widget, its parent widget must also be visible.

HTBasic for Windows

WAIT FOR EVENT Command

WAIT FOR EVENT

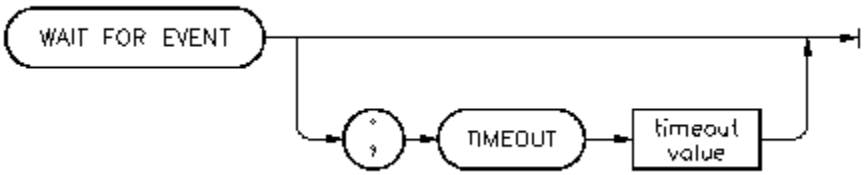
Suspends program execution until an event occurs.

NOTE

*Click the **>>** bar for additional information on the WAIT FOR EVENT command.*

Legal Usage	Keyboard Executable	Yes
	Programmable	Yes
	In an IF...THEN...	Yes
Example Statements	`WAIT FOR EVENT`	
	`WAIT FOR EVENT; TIMEOUT 30`	

Syntax



Item	Description	Range
timeout value	wait up to this many seconds for user input, then resume program execution	0 to any non-negative number

WAIT FOR EVENT Command (continued)

Description

WAIT FOR EVENT suspends program execution until a specified event occurs. When an enabled event occurs, WAIT FOR EVENT terminates and the event triggers the appropriate pending ON EVENT statement. If no events are currently defined, WAIT FOR EVENT returns immediately.

At the WAIT FOR EVENT statement, program execution is suspended until an event occurs. When an enabled event occurs, the WAIT FOR EVENT statement terminates and the event triggers the appropriate pending ON EVENT statement. If no events are currently defined, WAIT FOR EVENT returns immediately.

WAIT FOR EVENT will wait indefinitely for an event unless a TIMEOUT value is specified. The TIMEOUT option specifies the number of seconds after which program execution resumes if no enabled events occur. The corresponding branch may or may not be taken, depending on which context has the highest priority - the current context priority or the defining context priority.

*_WAIT FOR EVENT will wait if any events are defined, even if any or all events are disabled or are associated with widgets that are not visible. If the widgets are not visible, WAIT FOR EVENT will terminate only if the timeout period is reached, or if you press the **Stop** or **Reset** key.*

For example, the following lines allow the program to loop continuously, waiting for the specified event to occur, which allows the computer to do other things while waiting for the event to occur.

```
LOOP
    WAIT FOR EVENT
END LOOP
```

Since the WAIT FOR EVENT statement suspends program execution, the computer is free to service other processes. In the following construct, the computer is "busy waiting" (that is, the CPU stays busy doing nothing):

```
10   GOTO 10
```

If keeping the CPU free to run other processes is important in your program or

computer environment, we recommend using either of the following two constructs:

```
10  LOOP
20      WAIT FOR EVENT
30  END LOOP
```

or:

```
10  WAIT FOR EVENT
20  GOTO 10
```

HTBasic for Windows

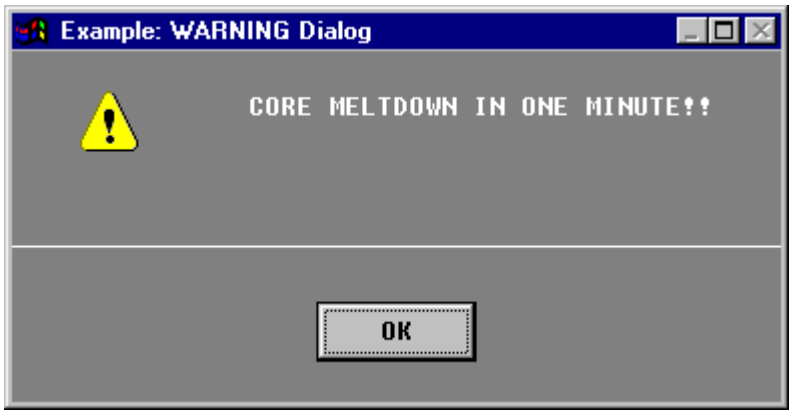
WARNING Dialog

WARNING Dialog

Displays a warning message and suspends program execution until the operator acknowledges the information

---

Example Image



Example Program

See WARNING Dialog for an example program that produces a display similiar to that shown above.

NOTE

See the following program for another example using the WARNING dialog.

Dialogs Tests

Attributes

See WARNING Dialog Attributes for the WARNING dialog attribute list.

Remarks

None.

HTBasic for Windows

WARNING Dialog Attributes

WARNING Dialog Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
DEFAULT BUTTON [3]	Numeric	Any valid index into the DIALOG BUTTONS array	0
DIALOG BUTTONS	String array	Any valid string array	A single-element array containing OK
FONT	String	*Font* [1]	
HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
JUSTIFICATION	String	"LEFT", "CENTER"	"CENTER"
MAXIMIZABLE	Numeric	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE	Numeric	0,1	1
PEN	Numeric	Any valid BASIC pen number (0-255)	System dependent [2]
RESIZABLE	Numeri	0,1	1 (resizable, special

	c		border appears around the dialog)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
TITLE	String	Any valid string	WARNING
USER DATA	String	Any valid string	None
VERSION	String	Any valid string	Current version
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [4]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [4]	Autoplacement

[1] For an explanation of *Font* and the default value, see Specifying FONT Attribute Values

[2] Read the CONFIG file or query for the default with a STATUS command

[3] For "MULTILINE:1", this attribute must be set to "-1" so that **Return** does not terminate the dialog

[4] Screen or parent work area upper-left corner

FONT

Specifies the font to be used for the text (the string value of the VALUE attribute) in the WARNING dialog. For a description of allowed fonts, see Specifying FONT Attribute Values.

PEN

The string value of the VALUE attribute will be painted using this pen color. For a discussion of allowed pens, see Settable Pens Table.

WIDTH

The value of this attribute specifies the width of the dialog or widget, in units of pixels. The value of WIDTH is set automatically when you set the COLUMNS attribute on some widgets. See INTERDEPENDENT ATTRIBUTES for information on how WIDTH interacts with the FONT and COLUMNS attributes.

HTBasic for Windows

What is Context-Sensitive Help?

HTBasic for Windows provides "context-sensitive help" that allows you to link HELP FILE and HELP TOPIC attributes for a specific widget in a user interface. Using context-sensitive help, you can build your own set of help files independently of the help files supplied with HTBasic for Windows.

The HTBasic for Windows Help system allows all widgets (except those such as SEPARATOR) to be connected to online Help through the HELP FILE and HELP TOPIC attributes.

If you press the **F1** softkey when a widget has the input focus **or** click the right mouse button when the pointer is over the widget, the Help system pops up using the designated HELP FILE and HELP TOPIC. (By default, all you get is a Help topic on the widget itself.)

HTBasic for Windows

What is HTBasic Plus?

Introduction

HTBasic Plus is a system of commands, utilities, and applications designed to enhance HTBasic programs. HTBasic for Windows HTBasic Plus provides a set of commands to create dialogs and widgets for effective graphical user interfaces.

In addition, you can create widgets with the Screen Builder Application, and can customize HTBasic Plus using the CONFIG command. See the Glossary for definitions of HTBasic Plus terms.

To add HTBasic Plus capability to HTBasic for Windows, execute the following statement from the command line or programmatically:

```
LOAD BIN "BPLUS"
```

Screen Builder/Custom Configurations

A Screen Builder application is included to help you design user interfaces. You can also perform custom configurations on some HTBasic Plus features. See Screen Builder for information on the Screen Builder application. See Custom Configurations for information on customizing HTBasic for Windows.

Widgets Attribute Default Values

Although common attributes have the same type and allowed values for all widgets (or level-0 widgets or dialogs), they may have *different* default values for different widgets (or level-0 widgets or dialogs).

For example, the WIDTH attribute for Type (numeric) and Allowed Values (any integer number) are the same for both the METER and the BARS widgets. However, the METER widget's WIDTH attribute has a default value of 220 pixels, while the BARS widget's WIDTH attribute has a default value of 250 pixels.

HTBasic for Windows

Widgets Attribute Definitions

Widget Attributes

Attributes are qualities or properties of HTBasic for Windows widgets (and dialogs) The function and appearance of a widget (or dialog) is determined by the values of its attributes. There are five types of widget attributes, as shown in Widgets Attribute Types. See Widgets Common Attributes for a list of common attributes for dialogs.

Each widget has a set of specific attributes that are specific to the widget and a set of common attributes that are common to all widgets. In addition, some widgets have a set of attributes that are applicable only when the widget is used as a level-0 widget.

NOTE

OPTION BASE 0 is used for all array indices. Dialogs and widgets always treat the first element in an array as element 0, regardless of how you dimension (DIM) or ALLOCATE.

Widget Attributes Characteristics

All widget attributes have a:

- Data type, such as string or numeric
- Range of allowed values, such as ["ENGINEERING", "SCIENTIFIC", "FIXED"] or [0..100]
- Default value

Appearance related widget attributes control things such as:

- The colors of the various parts of the widgets
- Whether the widget is visible or has a border
- Whether the widget is movable, resizable, or scrollable
- The width, height, and position of the widget on the screen

Function related widget attributes control things like:

- The widget's title
- Whether the text is left or center justified
- Whether the widget's scale is logarithmic or linear
- The value to be displayed

Widget Common Attributes

See the Widgets Common Attributes table for widgets common attributes Note that some common attributes are defined only for level-0 widgets.

NOTE

Although widgets may have the same common attributes, the default values may differ among widgets. See the appropriate widget for specifics on each attribute.

HTBasic for Windows

Widgets Attributes

This topic provides general information about widget attributes, including:

- Widget Attribute Definitions
- Setting/Changing Widget Attributes
- Querying Widget Attributes
- Widget Attribute Default Values
- Specifying FONT Attribute Values

See List of Widgets for descriptions/listings of widgets.
See Widgets Format for the format of widgets descriptions.
Click the item name to view each topic item.

HTBasic for Windows

Widgets Common Attributes

Overview

This topic lists the common attributes for HTBasic Plus widgets. Click the attribute name for a description of the attribute.

NOTE

Since these common attributes may have different defaults for each widget, this information may be changed for a specific widget. See the appropriate widget description for details.

Widgets Common Attributes

Attribute	Type	Allowed Values	Default
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
BORDER [2]	Numeric	0,1	1
FOCUS	Numeric	0,1	Null
HEIGHT	Numeric	Any integer number (of pixels)	Varies
HELP_FILE	String	Any valid file name	Null
HELP_TOPIC	String	Any valid index or topic_id	Null
INSIDE_HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE_	N	Any integer number (of	Varies

WIDTH	u m eri c	pixels)	
MAXIMIZABLE [4]	N u m eri c	0,1	1 (maximizable, button appears in title bar)
MINIMIZABLE [4]	N u m eri c	0,1	0 (not minimizable, button does not appear in title bar)
MOVABLE [4]	N u m eri c	0,1	1 (movable, click on title bar and drag)
RESIZABLE [4]	N u m eri c	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	N u m eri c	0,1	0 (do not restore the screen)
STACKING ORDER	N u m eri c	0 to number of siblings [3]	0
SYSTEM MENU [4]	St rin g or St rin g arr ay	Any string or string array with 1-64 elements	Null
SYSTEM MENU COUNT [4]	N u m eri c	0 to number of items in system menu	0

SYSTEM MENU	Numeric	0 to number of items in system	0
EVENT [4]	Numeric	menu -1	
TITLE [4]	String	Any valid string	Widget name
USER DATA	String	Any valid string	Null string
VERSION	String	Any valid string	Current version
VISIBLE	Numeric	0,1	1
WIDTH	Numeric	Any integer number (of pixels)	Varies
X	Numeric	Any integer, number of pixels from 0,0 [5]	Autoplacement
Y	Numeric	Any integer, number of pixels from 0,0 [5]	Autoplacement

[1] Read the CONFIG file or query for the default with a STATUS command

[2] Child widget only

[3] A sibling is another child widget with the same parent, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[4] For level-0 widgets only

[5] Screen or parent work area upper-left corner

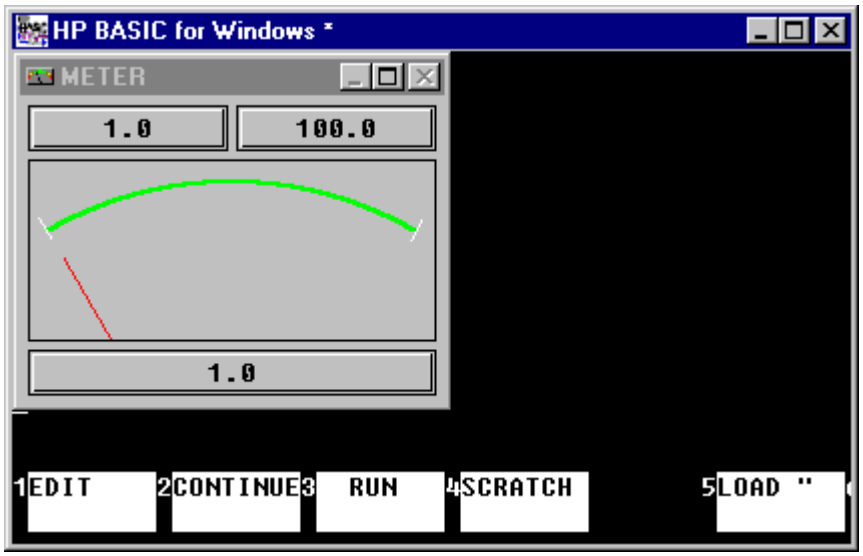
HTBasic for Windows

Widgets Displays

A *widget* is an HTBasic Plus object that is created (and destroyed) using the ASSIGN statement.

Typical Widget Display

The widget is displayed on the screen from an executing HTBasic for Windows program. A typical widget display follows. Widgets are created and destroyed (closed) in the same way as for an I/O path. However, ENTERs and OUTPUTs to widgets are not performed in the same way as for I/O. Also, RESET cannot be used with a widget.



Widget Focus

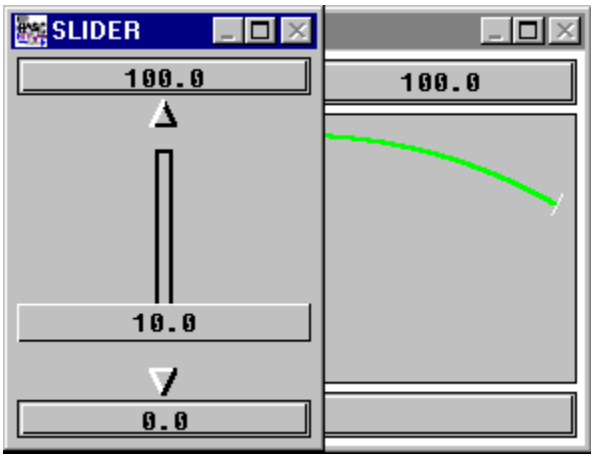
When one or more widgets are displayed on the screen, you can interact only with the widget that has the focus. When you click the mouse within the border of a dialog or widget, that dialog or widget "gains the focus". Any mouse clicks or keystrokes then made are input to the widget that has the focus.

"On Top" Widget

If you create two or more widgets that occupy the same space on the screen, the widget created last in the program will be "on top" on the screen. The widget(s) created previously will be partially or totally covered by the widget created last if they occupy part or all of the same space on the screen. To programmatically change

focus use STACKING ORDER.

For example, as shown in the following figure, the SLIDER widget was created after the METER widget, so the SLIDER widget appears on top in the display. It is perfectly acceptable to create widgets on top of other widgets. The widget management software automatically draws overlapped widgets and repairs covered widgets if you place other widgets on top of existing widgets.



HTBasic for Windows

Widgets Format

This topic shows the format for HTBasic Plus widgets descriptions.

- See Widgets Attributes for information on widgets attributes
- See List of Widgets for descriptions/listings of dialogs/widgets

NOTE

Each widget description has the general structure shown. However, see the appropriate widget description for details on each widget.

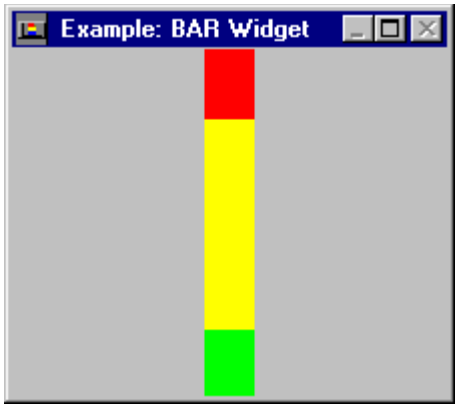
HTBasic Plus Widgets Format

BAR Widget

Graphically displays numeric values

Legal Usage	Level-0 Widget:	Yes	
	Parent to:	None	
	Child of:		PANEL

Example Image



Example Program

See BAR Widget for a program that creates a BAR widget similar to the one displayed above.

NOTE

See the following program for another example using the BAR widget:

Bar Meter Limits

Attributes

See BAR Widget Attributes for the BAR widget attribute list.

Remarks

The BAR widget graphically displays numeric values using a single bar.
.....

Events

Events for the BAR widget are:

- ALARM
  - SYSTEM MENU

ALARM

This event is generated when the VALUE attribute is set to a value
.....

HTBasic for Windows

Widgets Hierarchy

The HTBasic for Windows widget management software controls all widget levels and positions on the screen, and maintains a hierarchy of widget (and dialogs) levels. There are six categories of widgets:

- level-0 widget
- parent widget
- child widget
- sibling widget
- container widget
- transient widget

Click the widget name for a description of the widget categories.

X

The value of X specifies the horizontal location of the left side of the dialog or

widget. The units of X are pixels. For level-0 widgets and for dialogs, the value of X is relative to the left side of the HTBasic for Windows outside window client area. If the widget is a child widget, the value of X is relative to the left side of the parent widget's client area.

AUTOSCALE

With the AUTOSCALE mode on (value of 1), the graph will be automatically sized to encompass all of the data points currently in the graph.

MARKER1 X, MARKER1 Y, MARKER2 X, MARKER2 Y

This attribute specifies the X and Y position of the 2 markers.

NUMBER FORMAT

This attribute offers a choice between different ways to format the numbers that are drawn adjacent to major tick marks. The time formats allow almost any representation of a time value you might want. In all cases, the numbers are drawn using only as much precision as is needed to distinctly identify the value of each tick mark.

For example, if the "HOURS" format is chosen and the TICK SPACING attribute is set to 60 (seconds), the numbers will be drawn as HH:MM. The seconds field is dropped because it is not needed in order to accurately represent the tick values.

Allowed Values For NUMBER FORMAT

Allowed Value	Description
AUTO	Selects between FIXED, ENGINEERING, and SCIENTIFIC to represent the number with the smallest number of characters
FIXED	Displays fixed-point number format
SCIENTIFIC	Displays standard scientific notation, for example: 3.38E-12
ENGINEERING	Uses engineering format with the exponent represented by the following single letter designators (listed with their equivalent scientific notation exponents):

Letter Designator Notation	Equivalent Scientific
m	E-03
u	E-06
n	E-09
p	E-12
f	E-15
a	E-18
k	E+03

M	E+06
G	E+09
T	E+12
P	E+15
E	E+18

The following time formats all assume that the POINT LOCATION values are in seconds, as given by the HTBasic TIMEDATE function. The XY GRAPH does the conversion from seconds to the formats in the following table.

NOTE

The allowed values for the "MINUTES", "HOURS", and "DAYS" formats do not wrap around as the values for "CLOCK12" and "CLOCK24" do. Instead, the most significant fields accumulate indefinitely. For example, if the value of NUMBER FORMAT is "DAYS", the seconds, minutes, and hours fields will have maximum values of 59, 59, and 23, respectively, but the days field can accumulate indefinitely.

Time-related Allowed Values for NUMBER FORMAT

Allowed Value	Format Displayed in Tick Labels
MINUTES	minutes:seconds
HOURS	hours:minutes:seconds
DAYS	days:hours:minutes:seconds
CLOCK12	hours:minutes:secondsA (AM) OR hours:minutes:secondsP (PM) - The display wraps around to 1:00 after 12:59. - Any tick labels representing negative time values will display the time as if the clock were turned back from midnight. For example, a tick at -20 seconds would be labeled as 11:59:40P.
CLOCK24	hours:minutes:seconds

- The display wraps around to 00:00 after 23:59.

- Any tick labels representing negative time values will display the time as if the clock were turned back from midnight. For example, a tick at -20 seconds would be labeled as 23:59:40.

ORIENTATION

The value of this attribute sets the orientation of the XY GRAPH in its window. This transformation is the LAST to be applied, so it affects all of the other directional attributes equally. "UP" is the standard orientation in which:

- The data origin is in the lower-left corner of the display window
- The XY GRAPH data-surface scrolls only to the left
- The X axis is horizontal
- The Y axis is vertical

If the ORIENTATION is changed, the normal top of the graph now points in the new direction. For example, an orientation of "LEFT" causes:

- The data origin to be in the lower right corner of the display window
- The X axis to be vertical
- The Y axis to be horizontal

If you want to flip one of the axes over, use a negative value for the RANGE attribute and choose an appropriate value for the ORIGIN attribute.

ORIGIN

The application of the value of this attribute depends on the value of the CURRENT AXIS attribute. If CURRENT AXIS is "X", the value of ORIGIN specifies the location in the coordinate plane of the left edge of the viewing window (assuming that ORIENTATION is "UP" or "VERTICAL").

If CURRENT AXIS is "Y", the value of ORIGIN specifies the location in the coordinate plane of the bottom edge of the viewing window (assuming that ORIENTATION is "UP" or "VERTICAL").

If the LOGARITHMIC attribute is turned on (value is 1), the value of ORIGIN is required to be non-zero and positive. When you set the value of LOGARITHMIC to 1, if the current value of ORIGIN is zero or negative, the value of ORIGIN will be set to 1.

PEN

The PEN attribute specifies the pen color to used for all labeling and axis numbering.

POINT CAPACITY

The POINT CAPACITY value specifies the total number of points on the XY GRAPH. The value of POINT CAPACITY is also the size of the data "container" in memory. This buffer is where the X DATA and Y DATA attributes store their values.

That is, if you set the X DATA attribute array with values from an array in your program, those values will be stored in the "container" or internal buffer that was created according to the POINT CAPACITY value. Memory consumption of XY GRAPH varies according to the setting of "COMPRESS X" and "COMPRESS Y". The following table shows the levels of memory consumption according to different settings:

COMPRESS X/Y Memory Consumption

COMPRESS X	COMPRESS Y	bytes/ point
1	1	4 (default)
1	0	10
0	1	10
0	0	16

For example, if you set the value of POINT CAPACITY to 1000 and COMPRESS X:1 and COMPRESS Y:1, you have reserved 4000 bytes of memory as an internal buffer. Setting the SHARED X attribute value to 1 reduces the amount of memory used by the XY GRAPH.

SHARED X

This mode conserves memory in the case where all traces share the same set of X axis values. When it is turned on, all traces will use trace 1's X DATA value. Any existing X DATA values in traces 2 and up will be erased.

TAB STOP

The TAB STOP attribute allows you to control some XY GRAPH functions from the keyboard. Markers can be used without a mouse. When the widget is active, the **Tab** key moves the cursor around the marker buttons and the marker value fields. On each field, the marker is controlled as follows:

Arrow-Key Behavior When XY GRAPH Widget Has Input Focus

Vertical Arrow-Key Behavior	Horizontal Arrow-Key Behavior
scroll	scroll

X DATA

This attribute is an array of X data values to be entered into the trace designated by CURRENT TRACE. These X values will be matched up with Y values supplied by the Y DATA attribute. Nothing will happen to the value of X DATA if, when you try to set it, SHARED X mode is on and CURRENT TRACE is not equal to 1.

If you try to set the X DATA array with an array smaller than the value of POINT CAPACITY for the trace, X DATA will accept all of the values from the smaller array. If you try to set the X DATA array with an array larger than the value of POINT CAPACITY for the trace, X DATA will accept the low-index values from the larger array and the highest-index values will be lost.

Y DATA

This attribute is an array of Y data values to be entered into the trace designated by CURRENT TRACE. These Y values will be matched up with X values supplied by the X DATA attribute. If SHARED X mode is on, trace 1's X values will be used.

HTBasic for Windows

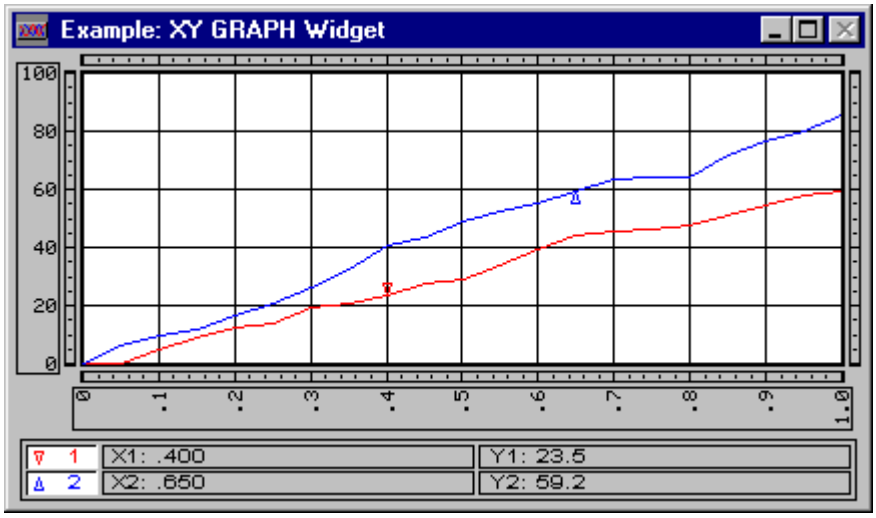
XY GRAPH Widget

XY GRAPH Widget

Used to plot data on X-Y coordinate plane

Legal Usage	Level-0 Widget:	Yes
	Parent to:	None
	Child of:	PANEL

Example Image



Example Program

See *XY GRAPH Widget* for an example program that generates a display similar to that shown above.
See the following programs for other examples using the XY GRAPH widget:

Frequency Response
Lissajous Patterns
Wing Stress/Vibration Analysis
XY GRAPH Shared Traces

Attributes

See XY GRAPH Widget Attributes for the XY GRAPH widget attribute list.

Remarks

The XY GRAPH widget plots data on an X-Y coordinate plane. The XY GRAPH widget has most of the same attributes as the STRIPCHART widget. However, the XY GRAPH is intended to display sets of X,Y data points.

For the STRIPCHART widget, you assign an X-coordinate POINT LOCATION along the axis and then load an array of Y-coordinate VALUES for that X-coordinate. As you keep incrementing the POINT LOCATION and writing an array with new Y-coordinate VALUES, the STRIPCHART display scrolls and all the traces are updated in parallel.

For the XY GRAPH widget, the display appears the same and is set up the same, but it does not scroll (the X axis is fixed). Also, instead of setting an X value and then updating the Y values of all traces, you simultaneously provide ALL the X and Y coordinates for a trace to plot that trace along its entire path. For example:

```
CONTROL @Xy; SET ("CURRENT TRACE":3,"X DATA":X3(*),"Y DATA":Y3(*))
CONTROL @Xy; SET ("CURRENT TRACE":4,"X DATA":X4(*),"Y DATA":Y4(*))
CONTROL @Xy; SET ("CURRENT TRACE":5,"X DATA":X5(*),"Y DATA":Y5(*))
```

If all your data sets have the same X coordinates, you can set SHARED X and only load the X data set once. For example:

```
CONTROL @Xy; SET ("SHARED X":1,"CURRENT TRACE":1,"X DATA":X(*))
CONTROL @Xy; SET ("CURRENT TRACE":3,"Y DATA":Y3(*))
CONTROL @Xy; SET ("CURRENT TRACE":4,"Y DATA":Y4(*))
CONTROL @Xy; SET ("CURRENT TRACE":5,"Y DATA":Y5(*))
```

For SHARED X mode on the XY GRAPH, the X DATA array is associated with trace 1. Everything else - format options, CURRENT AXIS, CURRENT TRACE, etc. - is the same as for the STRIPCHART widget

The MARKER attribute allows you to set the operation of one or two trace markers that the user can move along the specified trace. MARKER can be set to several modes: NONE (no markers), ONE marker, TWO markers, DELTA (difference between X & Y marker values), and RATIO (ratio of marker values). When markers are turned on, the appropriate marker values are displayed on the bottom of the XY GRAPH widget.

Traces to be marked are designated with the attributes MARKER1 TRACE and MARKER2 TRACE. The positions of the markers on the trace can be set or queried using the attributes MARKER1 X / MARKER1 Y, and MARKER2 X / MARKER2 Y. You can trap marker movements with the MARKER1 MOVED and MARKER2 MOVED

events.

Events

Events for the XY GRAPH widget are:

- MARKER1 MOVED, MARKER2 MOVED
- SYSTEM MENU

MARKER1 MOVED, MARKER2 MOVED

This event is generated when a marker is moved.

SYSTEM MENU

This event is generated when the operator selects items from the SYSTEM MENU.

HTBasic for Windows

XY GRAPH Widget Attributes

XY GRAPH Widget Attributes

Attribute	Type	Allowed Values	Default
AUTOSCALE [3]	Numeric	0,1	0
AUTOTICK [3]	Numeric	0,1	1
AXIS LABEL [3]	String	Any string	Null string
BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
BIG NUMBERING	Numeric	0,1	0
BORDER [2]	Numeric	0,1	1
COMPRESS X, COMPRESS Y [4]	Numeric	0,1	1
CURRENT AXIS	String	"X", "Y"	"X"
CURRENT TRACE	Numeric	0 to value of TRACE COUNT	1
GRID PEN	Numeric	0 to 255	0

HEIGHT	Numeric	Any integer number (of pixels)	Autosizing
HELP FILE	String	Any valid file name	Null
HELP TOPIC	String	Any valid index or topic_id	Null
INSIDE HEIGHT	Numeric	Any integer number (of pixels)	Varies
INSIDE WIDTH	Numeric	Any integer number (of pixels)	Varies
LOG TICKS [3]	Numeric	1 to 5	4
LOGARITHMIC [3]	Numeric	0,1	0
MARKER	String	"NONE", "ONE", "TWO", "DELTA", "RATIO"	"NONE"
MARKER1 TRACE, MARKER2 TRACE	Numeric	1 to value of TRACE COUNT	1
MARKER1 X, MARKER1 Y, MARKER2 X, MARKER2 Y	Numeric	Any real or integer number	None
MAXIMIZABLE [6]	Numeric	0,1	1 (maximizable, button appears in title bar)

MINIMIZABLE [6]	Numeric	0,1	0 (not minimizable, button does not appear in title bar)
MINOR TICKS [3]	Numeric	0 to 100	1
MOVABLE [6]	Numeric	0,1	1 (movable, click on title bar and drag)
NUMBER FORMAT [3]	String	"AUTO", "FIXED", "ENGINEERING", "SCIENTIFIC", "MINUTES", "HOURS", "DAYS", "CLOCK12", "CLOCK24"	"AUTO"
ORIENTATION	String	"UP", "DOWN", "LEFT", "RIGHT"	"UP"
ORIGIN [3]	Numeric	Any real number	0.0
PEN	Numeric	0 to 255	System Dependent [1]
POINT CAPACITY [4]	Numeric	0 to the system's memory capacity	100
POINT SYMBOL [4]	Numeric	0 to 15	- Color monitor: 0 (off) - Monochrome monitor: (value of CURRENT TRACE + 1)
RANGE [3]	Nu	Any non-zero real number	1

	meric		
RESIZABLE [6]	Numeric	0,1	1 (resizable, special border appears around the widget)
RESTORE SCREEN	Numeric	0,1	0 (do not restore the screen)
SHARED X	Numeric	0,1	0
SHOW GRID	String	"NONE", "MAJOR", "MINOR"	"NONE"
SHOW NUMBERING [3]	Numeric	0,1	1
SHOW TICKS [3]	Numeric	0,1	1
STACKING ORDER	Numeric	0 to number of siblings [5]	0
SYSTEM MENU [6]	String or String array	Any valid string or string array	Null
SYSTEM MENU COUNT [6]	Numeric	0 to number of items in system menu	0
SYSTEM MENU EVENT [6]	Num	0 to number of items in system menu -1	0

	eric		
TAB STOP	Numeric	0,1	1
TICK ORIGIN [3]	Numeric	Any real number	0
TICK SPACING [3]	Numeric	Any non-zero real number	1
TITLE [6]	String	Any valid string	XY GRAPH
TRACE BACKGROUND	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
TRACE COUNT	Numeric	1 to 100	4
TRACE LABEL [4]	String	Any string	Null string
TRACE PEN [4]	Numeric	Any valid BASIC pen number (0-255)	System dependent [1]
TRACE VISIBLE [4]	Numeric	0,1	1
USER DATA	String	Any valid string	None
VALID POINTS [4]	Numeric	0 to the value of POINT CAPACITY	0
VERSION	String	Any valid string	Current version
VISIBLE	Nu	0,1	1

	meric		
WIDTH	Numeric	Any integer number (of pixels)	Autosizing
X	Numeric	Any integer, number of pixels from 0,0 [7]	Autoplacement
X DATA [4]	Numeric array	An array of real or integer numbers	None
Y	Numeric	Any integer, number of pixels from 0,0 [7]	Autoplacement
Y DATA [4]	Numeric array	An array of real or integer numbers	None

[1] Read the CONFIG file or query for the default with a STATUS command

[2] Child widget only

[3] *This is an XY GRAPH Per Axis attribute.* To set or get this Per Axis attribute, you must first set the value of CURRENT AXIS to the axis for which you want to set or get this attribute.

[4] *This is an XY GRAPH Per Trace attribute*. To set or get this Per Trace attribute, you must first set the value of CURRENT TRACE to the trace for which you want to set or get this attribute. If you set CURRENT TRACE to 0, the specific Per Trace attribute will be set for all traces.

[5] A sibling is another child widget with the same parent, or at the same level in the widget hierarchy, as the widget for which you are setting the value of the STACKING ORDER

[6] For level-0 widgets only

[7] Screen or parent work area upper-left corner

Y

The value of Y specifies the vertical location of the top of the dialog or widget.

The units of Y are pixels. For level-0 widgets and for dialogs, the value of Y is relative to the top of the HTBasic for Windows outside window client area. If the widget is a child widget, the value of Y is relative to the top of the parent widget's client area.

ancillary files

The set of files needed for proper operation of HTBasic for Windows HTBasic Plus. Depending on the computer system, these files reside in different directories.

attributes

Qualities of a dialog or a widget. Each attribute has a value or set of values that the programmer sets or gets with CONTROL or STATUS, respectively. The values of the attributes determine the function and appearance of the dialogs and/or widgets. Most attributes have default values.

child widget

A child widget is a widget that is one level below its parent in the widget hierarchy. A widget becomes a child widget when you assign it to a parent widget with the PARENT option to the ASSIGN statement. A widget can be both a parent widget and a child widget at the same time.

If a parent widget is specified, the new widget will be treated as a "child" widget of its parent. Child widgets are constrained to be within the parent widget boundaries. If you attempt to move a child widget outside the border of the parent widget, the child widget will be "clipped" at the parent widget's borders.

The child widget's X and Y coordinates are relative to the upper-left corner of the parent widget. Since child widgets are contained by their parent widget, child widgets cannot be level-0 widgets.

click

To press and immediately release the mouse button.

common dialog attribute

An attribute of all of the dialogs.

common widget attribute

An attribute of all of the widgets.

container widget

It is always a parent widget. Most commonly it is a PANEL widget in which you place a set of child widgets. That is, the child widgets are contained in the PANEL widget. A container widget can be, but does not have to be, a level-0 widget.

context

An instance of an environment consisting of a specific instance of all data types and system parameters that may be accessed by a program at a specific point in the program's execution. Context changes occur when subprograms or functions are invoked or exited.

context-sensitive help

Online information that is relevant to what the user is doing with a widget or a dialog. When enabled via the attributes HELP FILE and HELP TOPIC, context-sensitive help is activated by clicking the right mouse button.

dialog

One of the fundamental HTBasic for Windows entities that is created on the computer screen with the HTBasic for Windows DIALOG command from an executing BASIC program, or from the command line.

A dialog acts, from a program's point of view, as an INPUT statement with a TIMEOUT. From the user's point of view, a dialog appears as a popup panel that disappears when the user supplies a response. See List of Dialogs for an overview description of each dialog.

event-initiated branching

A programming technique that uses interrupts to redirect program flow.

focus

When you click the pointer on the screen (with the mouse or with the keyboard) within the border of a dialog or widget, the dialog or widget "gains the focus". That is, any mouse clicks or keystrokes that you make will be input to the widget that has the focus.

Common Widget Attribute allowing for programatic change of focus to a particular widget or to the Basic window.

See also STACKING ORDER, FOCUS

level-0

The first level of the widget management software's hierarchy of widgets.
The first level of the hierarchy is the one just below the screen level.

level-0 widget

The first level of the widget hierarchy is the one just below the screen level, and is called level-0. You create a level-0 widget by *not* using the PARENT option to the ASSIGN command when you create the widget.

A level-0 widget is not constrained to be within another widget and may exist at any place within the HTBasic for Windows output window. The X and Y coordinates of the level-0 widget are relative to the upper-left corner of the HTBasic for Windows output window.

A level-0 widget has a title bar at the top and a resize border around it. You can assign most of the widgets to be level-0 widgets. (All dialogs are level-0.) Only level-0 widgets can have a title bar on top of the widget, and a resize border around the widget. The title bar and resize border allow you to change the position and size of the widget.

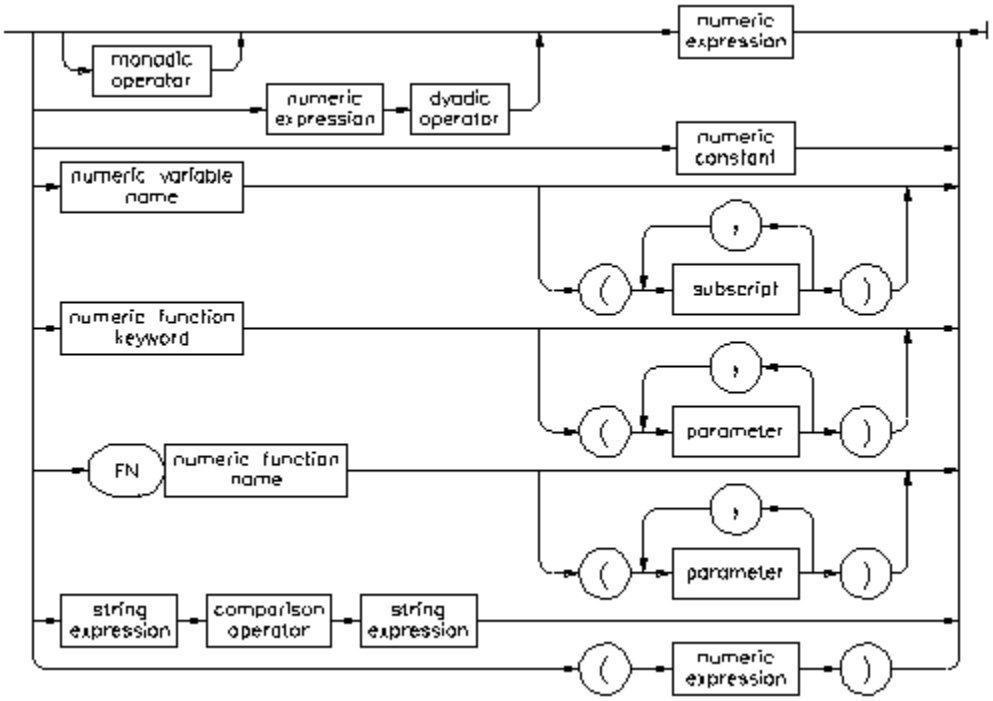
All widgets, with the exception of CASCADE MENU, MENU BUTTON, MENU SEPARATOR, MENU TOGGLE, PULLDOWN MENU, and SEPARATOR, can be level-0 widgets.

menu bar

The area just below the title bar in a level-0 widget. The menu bar contains menu widgets.

numeric expression

An expression that evaluates to a number. The syntax is:



Item	Description
monadic operator	An operator that performs its operations on the expression immediately to the right: + - NOT
dyadic operator	An operator that performs its operation on the two expressions it is between:^ * / MOD DIV + - = <<>> << >> <<= >>= AND OR EXOR MODULO
numeric constant	A numeric quantity whose value is expressed using numerals, decimal point, and optional exponent notation
numeric variable name	The name of a numeric variable or the name of a numeric array from which an element is extracted using subscripts

subscript	A numeric expression used to select an element of an array
numeric function keyword	A keyword that invokes a machine-resident function which returns a numeric value
numeric function name	The name of a user-defined function that returns a numeric value
parameter	A numeric expression, string expression, or I/O path name that is passed to a function
comparison operator	An operator that returns a 1 (true) or a 0 (false) based on the results of a relational test of the operands it separates: >> << <<= >>= = <<>>

operator

The person who interacts (using the mouse or keyboard) with the widgets and dialogs on the screen as the program is running.

parent widget

A parent widget is one level above all of its children in the widget hierarchy and contains its child widgets. A parent widget can be a level-0 widget, but is not necessarily one. When a parent widget is a level-0 widget, it has the same properties as a level-0 widget.

A widget becomes a parent when you assign child widgets to it using the PARENT option in the child widgets' ASSIGN statements. A widget can be both a parent and child widget at the same time.

pen

Is used to specify the colors in widgets, and may be set to any legal HTBasic for Windows pen number.

pixel

The image on a computer monitor comprises an array of dots that vary in color and intensity. A single dot is called a pixel.

pointer

The arrow-shaped cursor that appears within widgets and dialogs. You move the focus from one widget to the next by moving the pointer from one widget to the next.

resize border

The border that appears around level-0 widgets and dialogs and is used to resize them. You cannot use the resize border from the keyboard. To use the resize border:

- Use the mouse to move the pointer to the resize border
- When the pointer changes shape, press-and-hold the left mouse button and drag the border to the new position
- Release the left mouse button and the widget will redraw at the new size

screen origin

The upper-left corner of the screen, that has the coordinates (0,0).

NOTE

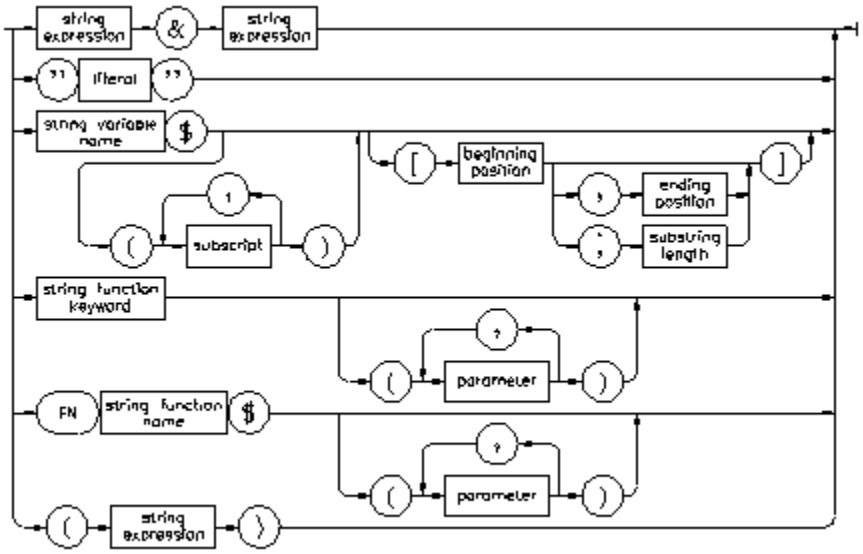
The screen origin in HTBasic Plus is different than for HP BASIC/9000. In HTBasic for Windows, the screen origin is in the lower-left corner of the screen.

sibling widgets

Sibling widgets are child widgets with the same parent, or at the same level in the widget hierarchy, as the widget for which you are setting the STACKING ORDER attribute value.

string expression

An expression that evaluates to a string. The syntax is:

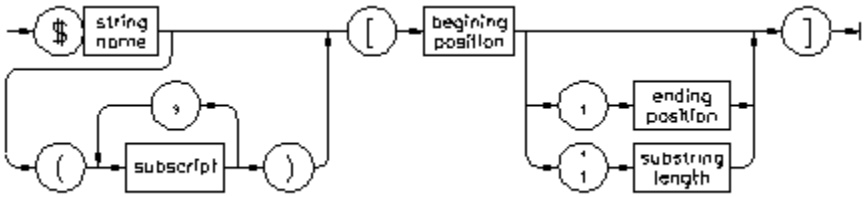


Item	Description
literal	A string constant composed of any characters available on the keyboard, including those generated with the **ANY CHAR** key
string variable name	The name of a string variable or the name of a string array from which a string is extracted using subscripts
subscript	A numeric expression used to select an element of an array
beginning position	A numeric expression specifying the position of the first character in a substring (see substring)
ending position	A numeric expression specifying the position of the last character in a substring (see substring)

substring length	A numeric expression specifying the maximum number of characters to be included in a substring (see substring)
string function keyword	A keyword that invokes a machine-resident function which returns a string value. String function keywords always end with a dollar sign ($).
string function name	The name of a user-defined function that returns a string value
parameter	A numeric expression, string expression, or I/O path that is passed to a function

substring

A substring is a contiguous series of characters that comprises all or part of a string. Substrings may be accessed by specifying a beginning position, or a beginning position and an ending position, or a beginning position and maximum substring length. The syntax is:



The beginning position must be at least one and no greater than the current length plus one. When only the beginning position is specified, the substring includes all characters from that position to the current end of the string.

The ending position must be no less than the beginning position minus one and no greater than the dimensioned length of the string. When both beginning and ending positions are specified, the substring includes all characters from the beginning position to the ending position or current end of the string, whichever is less.

The maximum substring length must be at least zero and no greater than one plus the dimensioned length of the string minus the beginning position. When a beginning position and substring length are specified, the substring starts at the beginning position and includes the number of characters specified by the substring length. If there are not enough characters available, the substring includes only the characters from the beginning position to the current end of the string.

system font

The default font used in dialogs and widgets. It can be changed in the configuration file. The default depends on the CRT resolution.

tab group

A group of *like, child widgets* that are created within ASSIGN statements, one after another in the program. You create a tab group so the operator can move the focus from one widget in the group to the next (and wrap around again), without having to traverse every widget in the PANEL.

Only the widgets that have a TAB STOP attribute can be members of a tab group and can accept the system focus. Level-0 widgets cannot be members of a tab group.

A common example of a tab group is a row of buttons lined up across the bottom of a PANEL. The buttons present the operator with a group of related choices, such as YES, NO and CANCEL.

title bar

The area at the top of a level-0 widget that contains the title text for the widget, as well as the window menu button, minimize button, and maximize button.

transient widget

A transient widget is created using the TRANSIENT option to the ASSIGN command. A transient widget is created from a parent widget as a result of some program or user action, and is used to create a custom dialog.

The TRANSIENT option is primarily used when the resulting widget is to function as a dialog. If you create a transient widget, other non-transient widgets cannot be placed on top of the transient widget.

If the transient widget has a parent widget, the transient widget is not constrained to lie within the bounds of its parent as are other child widgets. Visually, the transient widget appears to be a special type of level-0 widget.

widget

One of the fundamental HTBasic for Windows entities. A widget is created on the computer screen with the ASSIGN statement from an executing BASIC program.

Widgets are very flexible items that can be used to build sophisticated user interfaces. For example, you can create "virtual instruments" that include pushbuttons, sliders, and text, meters and graphics displays, and pulldown menus that can be controlled by a mouse.

Widgets can be loosely grouped into five categories, but some widgets may belong to more than one category. See List of Widgets for an overview description of each widget.

widget management software

The software that controls all widget levels and positions on the computer screen. It maintains a hierarchy for widgets and dialogs and keeps track of all level-0/parent/child widget relationships.

work area

The area in which the widget performs its essential function. For example, in the PRINTER widget, the area that contains the text is the work area. In the METER widget, the work area is the area that contains the meter arc, needle, tick marks, limits boxes, and value box. A level-0 widget surrounds the work area with a title bar, resize border, and (sometimes) a menu bar.