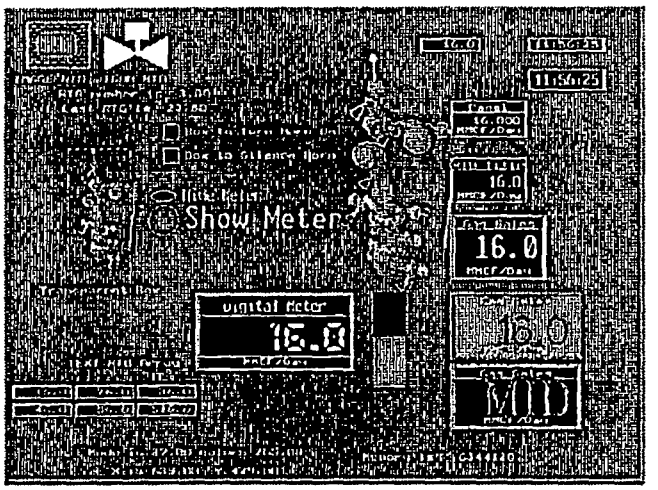
TEST SCADAWARE™

Real Time Graphics (RTG)

Revised: Oct 16, 1995



INTRODUCTION

SCADAWARE provides a procedural language which allows the building of complex Real-Time Graphic (RTG) images. The RTG objects are an extension to the TEST SCADA Protocol (TSP) language and are available in the full (not lite) version of the program. This document provides an overview with examples of how graphic images can be developed.

RTG OBJECTS

An RTG object is an intelligent graphic unit which is animated on the display screen during execution of a TSP procedure. RTG objects can be displayed on a blank graphic screen, or they can be placed on top of an arbitrary image which is prepared with en external program (Paint, 3D-Studio, etc). Each RTG type has a specific purpose and application. The Current RTG Types are:

Text	Plain Text display with optional frame
Panel	Panel meter text display, with Title and Units enclosed
Digital	LED style display for digits only
Box	Colored square or rectangle
Dot	Colored circle or ellipse
Image	Dynamic PCX or GIF file image
Icon	Dynamically colored PCX or GIF image
Vbar	Dynamically colored Vertical bar

Each RTG must be individually defined and configured as part of the TSP procedure which operates each screen. Much of the setup can be automated to reduce programming. However, each individual RTG element must at some point be defined during the screen setup in order for it to appear on-screen.

Defining an RTG requires specification of the following:

1. RTG type: Text, Panel, Dot, etc.
2. Anchor Point on screen (X-Y coordinate)
3. Expression to be evaluated and related to RTG dynamics
4. Colors, sizes, text, and other physical attributes.
5. Mouse Pick action, if required.

These RTG specifications are provided to the program on command lines which all begin with the keyword RTG. At any one point in time, a specific RTG number is begin defined, and all RTG statements refer to the current one until a new one is created. In many cases, the internal RTG numbering is not of concern to the user. There are cases, however, when dynamic reprogramming of certain RTG objects while they are on screen may occur. In these cases, specific RTG numbers will be required so that each individual RTG can be accessed as required during the reprogramming process.

As a simple example, consider the following program fragment:

```
RTG 5, Text, 0.9, 0.1,  8, out, Med
RTG *, value, $$T
```

This first line of this very simple example defines a small text window which we specified as RTG number 5. It's located at 90% of the screen width (left to right), and 10% of the screen height (from the top), so its near the upper left hand corner. The text box will be 8 characters wide in the Medium font (similar to normal VGA).

The second line references the currently defined RTG with the start (*) character. It specifies the value for the text to be the current time by using the special code $T. The extra dollar sign is necessary to prevent the time from being determined at the instant the RTG is defined rather than when it is dynamically updated later on.

This simple example skips the color definitions which will default automatically. When RTG updates take place, the value for this text element will be evaluated (as the current time) and placed in the window on the screen. All the messy details of color selection, font selection, alignment, and previous text overwrite are handled automatically by the RTG. With very little effort, a small clock has been put on the screen which will show the current time as long as the graphic screen is displayed.

Other RTG types will require additional information, such as colors and numeric limits, but all definitions follow a similar pattern. It is also possible to copy and array previously defined RTG's so that similar examples can be duplicated with very little effort. Common definitions may be put into separate procedures and called as subroutines from larger TSP procedures. For example, the clock above could be put into a procedure called graph_clock which is accessed with GOSUB GRAPH_CLOCK in any other TSP procedure. This lets the clock be figured out once, and used many times. A similar approach can be used for other RTG types when only 2 or 3 parameters change with each instance of the object.

CURRENT RTG OBJECT NUMBER

During the definition phase of a graphic procedure, each RTG to appear must be defined. It makes no difference to SCADAWARE in which order the objects are defined. Once displayed, they are updated in the order they were defined. However, the fast update speed makes it the definition order irrelevant.

The current RTG is the RTG number which is currently being specified. The number (from 1 to 128) can be assigned manually on the RTG creation statement, or it can be sequentially defined as each new RTG is specified. The automatic method works best for most applications where the user is not concerned with the specific RTG object number given to any particular screen element. In cases where the number is important, the user can either specify the number on the command line, or he can let it be defined automatically and then stored into a temporary variable for later access.

Various parameters used in RTG statements refer to an RTG number. Whenever an RTG number is required, the user can enter any of the following:

*	The current RTG number
+	The next RTG number in sequence.
-	The previous RTG number.
1..128	A specific RTG number
(expression)	Any valid TSP expression which evaluates to a number

Any of the following fragments are acceptable:

```
RTG 10, Text, ......          ; Define a text RTG as number 10  .
RTG *, value, X+10    ; value for current RTG will be result of X+10
RTG +, VBAR, ...              ; Start new RTG of type VBAR with next number
RTG +, copy, my_meter, 10,20,.. Copy rtg number in variable my_meter
```

You will find that most RTG generation program segments start with a plus (+) reference to move on to the next RTG number. Then, subsequent lines which apply to that same RTG will use the * reference. When this method is used consistently, blocks of code can be easily "cut and pasted" because there are no hard numbers with which to contend.

RTG NUMBER RANGES

Most RTG statements can be performed on a range of RTG numbers by specifying a start and end number, or a special code and a count. If the exact number of the starting RTG is specified, then the second number in the pair is the ending RTG number. If the first RTG is specified with a + or * code, then the second number is a count which will be added to the value of + or * to determine the final RTG number.

RTG Ranges are specified like channel ranges, with a colon in between the two numbers.

Consider the following:

RTG 1:5, color, black, white, blue....

This statement executes a color function (described later) on RTG's 1 through 5, inclusive. It has the same effect as entering 5 separate but similar COLOR statements.

Using a RTG number codes done by specifying the count, rather than the ending RTG number. If the current RTG number is 5, then the line

RTG *:3, color, black, white, blue....

Will have effect on RTG numbers 5, 6, and 7. This is because * evaluates to 5, and the 3 indicates a range of 3 values (5, 6, and 7).

NOTE: Current (June 95) versions will not allow a range of 1. Any references to ranges must evaluate to 2 or more RTG's.

X-Y LOCATIONS AND DIMENSIONS

Each RTG must be located on the screen with an X-Y location. The location can be specified in X pixels (left to right) and Y pixels (top to bottom). The range of pixels will depend on the screen resolution in service. Typical VGA is 640x480 (locations 0-639 and 0-479). Using pixel locations allows for precise locations to be used on a fixed size screen.

An alternate method of providing screen coordinates is to use percentages of screen width and height. Any X or Y coordinate provided as a number less than 1 will be considered a percentage rather than an exact pixel location. For example, an X coordinate of 25% on a standard VGA screen will evaluate to X axis pixel position 160 (25% of 640). Using the percentage method is often easier than calculating exact pixels. It also offers the advantage of making the RTG definitions flexible if the screen resolution is changed, to say 800x600.

Some RTG definitions require a size parameter (vertical bars, for example). Here, the user can enter either pixel counts or percentages of screen size. The system will evaluate the entry using the same rules as X-Y coordinates; numbers 0-0.99 are percents, 1 and greater are pixels.

Note that two new system functions are available to return the maximum X and maximum Y pixel available at any time. These functions are @MAXX(0) and MAXY(0). These may be useful for special tricks on screens which must apply to variable resolution screens.

RTG ANCHOR POINTS

Each RTG type has a default "anchor point" which defines a point on the object which corresponds to the X-Y screen location provided during setup. Specifying a particular X-Y location anchors a certain point on the RTG to that screen location. There are nine possible locations on each RTG, defined along the edges and at the center as follows:

Top Left (TL)	Top Center (TC)	Top Right (TR)
Center Left (CL)	Center Center (CC)	Center Right (CR)
Bottom Left (BL)	Bottom Center (BC)	Bottom Right (BR)

Although each type has a default anchor, an override can be given to specify any of the nine possible positions. This is done by appending a slash (/) and the two letter position code to the keyword identifying the RTG type. For example, to position text at the center of the area rather than at the bottom right we would use TEXT/CC when initializing the RTG. If no / is provided, the default is used.

If the / is appended, then the program decodes the two letter code and redefines the anchor point for that particular RTG (and any RTG's later copied or arrayed from that definition).

The default anchor point for each RTG type is as follows:

TYPE	DEFAULT ANCHOR	
Text	Bottom Right	/BR
Panel	Bottom Right	/TL
Digital	Bottom Right	/BR
Box	Top Left	/TL
Dot	Center Center	/CC
Image	Top Left	/TL
Icon	Top Left	/TL
Vbar	Top Left	/TL

RTG COMMAND LINES

In most cases, each RTG command line performs a specific action on the currently defined RTG object. All RTG numbers begin with the TSP keyword RTG, followed by a numeric reference (or a special code as described above). Various options follow the rtg number, and the type and meaning vary depending on the type of RTG. Some of the RTG options apply to all types, while others make no sense when applied to a certain type. The object-oriented nature of the RTG system allows unnecessary statements to occur without harm.

For example, specifying the font type for the digital meter has no effect because the digital meter type has a fixed font type of LED. Similarly, specifying a title for a Vertical Bar makes no sense because VBAR's have no titles. However, using these options will not cause harm because all objects store these attributes even if not used.

The specific RTG type statements which initiate and initialize a new RTG vary the most because each type has different requirements. Other statements, such as color specifications, are more consistent amount the various types because all RTG's treat colors pretty much the same way.

There are several RTG commands which affect the entire RTG system rather than just an individual object. These are used to apply the same action to all active object in a single line. These forms do not have a specific RTG number in their syntax. The special commands are:

RTG CLEAR Remove and destroy all active RTG objects. This is useful for screen chaining.
RTG UPDATE Re-evaluate and update all active and on-screen objects.
RTG AUTO OFF|ON Controls automatic Channel-type RTG option

Note that both Clear and Update are also available for individual channels. They are implemented as global controls to allow simple clearing and update of all current objects without concern of object counts and numbering.

RTG VALUES, CHANNELS, AND LIMITS

Some RTG object have an expression string associated with them to determines the numeric or text value of the object. A simple example would have a single channel, such as A1 or V10, as the RTG value. At each screen update, the system will evaluate the value of each RTG and adjust the display accordingly. Some types, such as text, will simply show the value. Vertical bars will be redrawn to reflect the new 0-100% scale of the value.

Another way to determine the text conent of the RTG is by assigning it to a specific TSP channel tag name. This is similar to the Value described above, except that some features of the RTG are automatically derrived directly from the channel. Any RTG objects which relate directly to a TSP channel should use the Channel option. Other values, such as text expressions, will use some other TSP expression to determine the high, med, or low position of the RTG. Note that some RTG types, such

as the dot and box, do not require any value or channel. This allows them to function as mouse pick points without the overhead of value processing.

Associated with each object are high and low limits. These limits define three states for each RTG - Lo, Medium, and High. Each state has a color assigned to it such that the RTG will be painted in a color which corresponds to current state. Once the limits are defined, the RTG does all the redraw and color selection automatically based on its current value. RTG's defined with a TSP expression value will require their limits to be specified as part of the RTG definition.

If RTG AUTO is on (default is off), then an RTU defined with the Channel option will pick up the hi-low limits directly from the channel description in the TSP database.

RTG DEFINITION STATEMENTS

The following section describes the different RTG types with examples of their definition statements. A later section will cover various options which affect all the types in similar ways. The format used here will be to show a prototype line with abbreviations used for each parameter to show the order in which they occur on the line. Parameters enclosed in [brackets] are optional and will take on a default as described for each type.

As in all TSP statements, the spacing of parameters on the line do not matter. The order, however, is very important. Parameters must be separated by either spaces or commas (but not both). If parameters are to be intentionally skipped, commas must be used to provide a blank parameter. Subsequent parameters will be accessed in proper order only if a blank space is used for missing values.

TEXT RTG

Real Time Graphic Text is used to display text of any type, including numbers, in a specific rectangular area of the screen. Simple text which is printed only once during a display can be more easily done with the normal TSP Cursor command. But text which represents a changing value, such as an analog channel, can be automatically displayed and updated with a Text RTG. The contents of the display area can be any valid TSP expression. A text RTG can show channel values, TSP numeric expressions, and plain text data such as time and date, or even words.

TEXT RTG WITH FRAME

TEXT RTG WITH LABELS ALL AROUND

The size of the box will be automatically determined based on the number of characters, and the size of the specified font. If no font and size is specified, the last system font and size determined by the TSP FONT command will be used. However, the user will normally enter the desired font to avoid surprises if the system font is changed at a later date. The same is true for the Frame which will default to that specified in the last SET FRAME statement executed by the system.

```
          3     4       5          6              7          8      9
RTG, #, TEXT, X_POS, Y_POS, [CHAR_COUNT], [FRAME], [FONT], [SIZE]
```

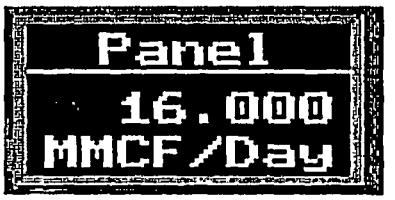
X_Pos	Anchor X pixel or % width value (number or expression)
Y_Pos	Anchor Y pixel or % width value (number or expression)
Char_count	Number of characters, which determines width of display
Frame	None, In, Out, Double - Frame type
Font	Any valid system font name (see Font command)
Size	Font size parameter (1-10) for stroked fonts.

```
RTG 5, text/CC, 0.5, 0.2, 10, Double, med
```

This example locates text at 50% across, 10% down, with the anchor point at the center of the text area. There will be space for 10 characters in the medium font. A double edge frame will be drawn.

PANEL RTG

Panel displays are derived from the simple Text RTG object described above. The difference is that a Panel will be a larger area with additional space for a Title (toptext) text above, and units (bottom text) text below. (The title and units are specified later with an option line).



PANEL RTG DISPLAY

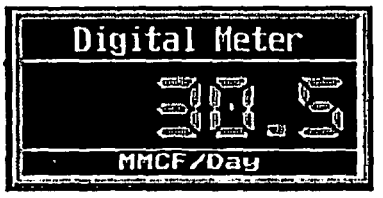
The initial setup for a Panel RTG is exactly like a text RTG, and the details will not be repeated here.

```
              3     4       5          6              7          8      9
RTG, #, Panel  X_POS, Y_POS, [CHAR_COUNT], [FRAME], [FONT], [SIZE]
```

DIGITAL RTG

The Digital RTG is a derivative of the Panel type which shows text along with a Title and Units. The difference (at the present time) is that the font is limited to the 30 point LED font, and the box dimensions are fixed at 160 pixels wide by 80 pixels high. This form factor provides a display of 4 across and 6 down on a standard VGA screen. The Digital type also has horizontal lines drawn to separate the text area from the display area.

Future additions to the Digital type will provide features such as up/down controls, analog bar display, etc.



DIGITAL METER RTG DISPLAY

All aspects of the Text and Panel definition line are allowed, although all after the Y_pos are

ignored. Refer to the Text RTG definition for details of the line.

```
                3      4      5      6              7       8      9
Rtg  #, Panel, X_Pos, y_pos, Char_count, Frame, Font, Size
```

DOT RTG

The RTG DOT is used to display dynamically colored circles and ellipses with optional associated text. This is useful for menu hit displays, and for assigning mouse pick actions to portions of the screen. The DOT requires at least an X axis dimension, in which case the program will determine the Y axis pixel size required to form a circle. If a Y axis size is also specified, then the system will produce an ellipse with those dimensions. The default anchor is the center of the dot unless overridden with a DOT/xx type entry.

CIRCLE AND ELLIPSE DOT RTG WITH TITLE TEXT

Title Text associated with the dot will be sized according to the Y dimension. Text is placed to the right of the DOT and is drawn in the "fill" color assigned to the dot (colors are specified later).

```
          3    4    5          6          7
RTG #. Dot. x_pos. y_pos. X_diameter. [Y_diameter]
```

X_Pos Anchor X pixel or % width value (number or expression)
Y_Pos Anchor Y pixel or % width value (number or expression)
X_diameter X pixel or %width for X axis (across) the dot.
Y_diameter Optional Y pixel or %width for Y axis (down) the dot.

```
RTG. 4. dot. 0.15. 0.25. 10 ; circular dot at 15% X. 25% Y. 10 pixels wide
RTG. 7. dot. 50 100. 30,10 ; eliptical dot at 50 px X. 100 px Y. 30X,10Y
```

BOX RTG

RTG Box's are very similar to dots, except that squares and rectangles are drawn instead of circles and ellipses. The X and Y dimensions form the sides of the box. If not Y size is provided, the system will determine the pixel count required to form a square. If a Y is given, the a rectangle of the specified dimensions will be formed. Title Text is drawn to the right similar to the method used for Dots.

BOX RTG'S WITH TITLE TEXT

```
          3    4      5      6      7
RTG #, Box x_pos, y_pos, X_size [Y_size]
```

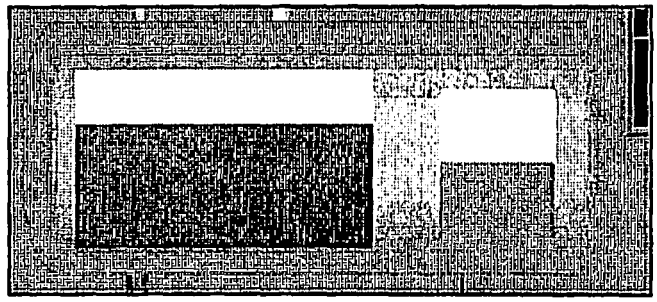
X_Pos Anchor X pixel or % width value (number or expression)
Y_Pos Anchor Y pixel or % width value (number or expression)
X_size X pixel or %width for X axis (across) the box

Y_size Optional Y pixel or %width for Y axis (down) the box.

```
RTG, 3, box, 0.15, 0.25, 10 : Square box at 15% X, 25% Y, 10 pixels wide
RTG, 1, box, 50 100, 30,10 : Rectangle at 50 px X, 100 px Y, 30X,10Y
```

VERTICAL BAR RTG

Vertical Bars (VBAR) are used to provided animated simulations of a varying level, usually overlaid on top another image representing a physical object such as a liquid tank. The vertical bar is positioned and sized using the normal RTG methods of either pixel count or screen percent. The full range of the bar must be specified so that the program knows what engineering value is the bar's bottom, and which is the top. Once the scale is provided, the program can evaluate the RTG value and determine which color must be used to indicate the level.



TWO VERTICAL BARS OVER TANK

The "fill" color forms the background of the vertical bar for all displays. The colored portion is drawn from bottom to top depending on the percent of full scale. The color used will depend on the current value in relation to the RTG's low, med, or high setting. One color is used to paint the entire colored portion. Stacked colors (indicating the various limit settings) are not allowed at this time.

```
          3     4      5       6       7      8        9
RTG, #, VBAR, X_POS, Y_POS, X_size, Y_size, lo_val, high_val
```

X_Pos	Anchor X pixel or % width value (number or expression)
Y_Pos	Anchor Y pixel or % width value (number or expression)
X_size	X pixel or %width for X axis (across) the bar
Y_size	Optional Y pixel or %width for Y axis (down) the bar
Lo_Val	Engineering units value at bottom of bar
High_val	Engineering units value at top of bar

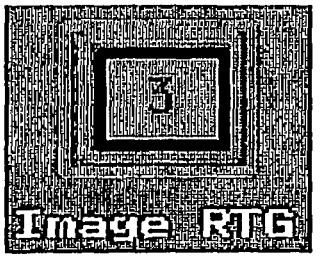
```
Rtg, 4, vbar, 0.10, 0.20, 30, 0.5, 0,1000
```

This example creates a vertical bar located 10% screen with across, and 20% down. The bar is 30 pixels wide, and 50% of the screen size in height. The bar is empty for result values of 0 or less, and full at result values of 1000 or more.

As with all RTG objects, a RTG COLOR statement will follow the definition line to specify the colors are used to dynamically paint the image on the screen.

IMAGE RTG

An RTG Image is a dynamic display using 1 of three graphic files (PCX or GIF). The file selected (in real time) will depend on the result value's position in the low, med, and high limit range. The three images have nothing in common other than the anchor point, and optional transparent color. Each image is individually painted on the screen to suit the current value of the RTG. When the value changes ranges, the old image is removed and the proper one is repainted.



Note that the images can be similar, or completely different. One application may be to show one device in three different positions, such as a lever or control switch. Another application may be to show three different "signs," such as CLEAR, CAUTION, and STOP! The images do not have to be the same size, and their contents are irrelevant to the RTG operation.

```
          3       4       5       6
RTG #  image, X_pos, y_pos, [scale]
```

X_POS Anchor X position, pixels or percent screen width
Y_POS Anchor Y position, pixels or percent screen height
Scale Optional scale factor, 0.1 to 10, implemented June 95

Note that the names of the 3 image files are not specified at creation time. The file names are provided late in place of the usual color names in the RTG COLOR statement which follows the definition line. The COLOR line used for images is different from all other COLOR lines which all follow a standard format. The Color Line which applies to Images specifies only the three file names, and an optional transparent paint color. The file names can include an extension of PCX or GIF, or the system default (set with SET GFILE) will be assumed.

```
          3      4       5       6          7
RTG #  COLOR, file_l, file_m, file_h, [transp_color]
```

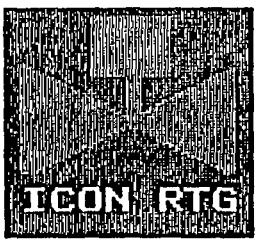
File_l Name of graphic file for Lo limit display
File_m Name of graphic file for Medium limit display
File_h Name of graphic file for High limit display
transp_color Optional color used for transparent paints.

```
RTG 3, color, lo.pcx, med.pcx, high.gif
```

This example specifies 3 file names, LO.PCX for the low limit, MED.PCX for the medium range, and HIGH.GIF for the high limit. Note that the filet types do not have to be the same. Because no transparent color was provided, the images will be pasted on top of the current display.

ICON RTG

An ICON RTG is similar to an image in that a graphic file is painted on the screen. However, the ICON uses a single file instead of three separate files. The differentiation among the three limit ranges is done by dynamically re-coloring the single image. Note the difference: Images use three separate files, while Icons use one file which is colored 3 ways.



Typical ICONs would be valve or pump symbols which are colored red or green to show a status. Rather than have 2 separate image files (one red, the other green), the ICON RTG lets you generate one file which is colored depending on the needs of the RTG at the moment.

Operation of an ICON requires that a single color be selected and specified as the "Draw" color. This color is the one that will be replaced at display time with the proper color. If black is used as the key, then the program will replace all black pixels in the image file with the proper color. This allows dynamic coloring, but prohibits the use of black in the image for purposes other than defining the dynamic color pixels.

Any color can be used as the draw color, so it is more likely that a seldom used color (like brown) will be used to indicate which pixels need coloring. The image must be developed with the standard palette used in the rest of the graphic images.

Icons can be transparent if a transparent color is specified on the COLOR line. This allows them to be painted over background images in a non-destructive manner, and removed if necessary without disturbing the underlying image.

```
              3      4      5         6          7
RTG  #, ICON, X_Pos, y_pos, Filename, [scale]
```

X_POS Anchor X position, pixels or percent screen width
Y_POS Anchor Y position, pixels or percent screen height
Filename ICON file, with or without file type.
Scale Optional scale factor, 0.1 to 10, implemented June 95

```
RTG, 11, Icon, 0.1, 0.2, h_valve
RTG *, color, black, red, green, blue
```

This example specifies an icon contained in a file called h_valve (with a type of PCX or GIF as determined by system defaults). It will be anchored at 10% across, and 20% down. The icon's draw color is black, and the three level colors are red, green, and blue.

RTG COLORS

The colors used to display RTG objects are specified in similar ways for all types except the IMAGE as described above. Most types support 5 color specifications, which appear in this order in the RTG COLOR statement:

Draw of Fill	Used for background, Title, Units, and Edges.
Low	Used for primary color when range is low.
Medium	Used for primary color when range is medium.
High	Used for primary color when range is high.
Transparent	Used as key color when painting images.

As in any SCADAWARE application, colors can be specified with text names (like RED, Blue, etc) or with numbers. Names can have a plus (+) appended which indicates "Bright". For example, BLUE means normal blue, while BLUE+ means bright blue. The default colors, as defined by all IBM compatible displays, are:

0	Black	8	Dark Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Brown	14	Light Yellow (Brown)
7	Gray	15	White

The RTG system also allows the use of the word 'NONE,' or 'TRAN' (transparent) to indicate

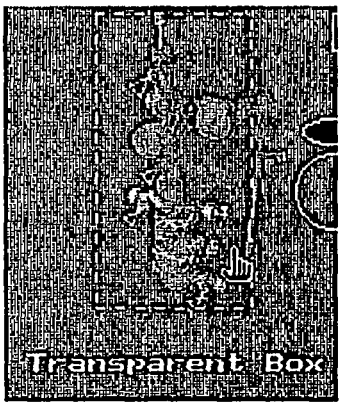
that no colors should be used. This seemingly useless feature is actually very handy when defining boxed areas which do not appear on screen, yet allow mouse picks to occur.

TRANSPARENT COLORS

Placing images on an existing screen can be done in an opaque fashion (the default method), or it can be done "transparently." In opaque mode, the image is placed on the screen complete with its own background. The existing screen background is completely covered with the new opaque image.

A transparent image is one in which a specific key color is identified as the image's "background." When doing a transparent paint, any pixels in the image with the specified key color are not transferred to the screen display. This allows for an "overpaint" effect, where the existing screen image forms the new background for the image being painted.

The transparent color is applicable to IMAGES and ICONS only. The key color is specified as the last item in an RTG COLOR statement, immediately following the high range filename (images) or high range color (icons).



This transparent box is indicated by the dashed lines around the wellhead image. The image was put on the screen with a PAINT statement, and then enclosed with the hidden box using RTG statements. The positioning of the box is not automatically tied to the image on the screen because the image was part of an overall background which was contained in an overlay file. The RTG box was generated like any other BOX except that its color was specified as NONE. The mouse actions available with this transparent box is the same as a normal box or any other RTG object.

Rather than use PAINT and an associated transparent box, It may be easier to use the Image RTG to provide a mouse selection. Like any other RTG, an image can be assigned a Pick action, thereby letting the user select the image with an automatic pick box.

RTG LABEL TEXT

Each RTG object can have text labels placed at various points on the screen which have a predefined location relative to the object itself. This text is normally used to label the RTG for identification purposes, or to describe mouse hit actions (for buttons). The text is handled slightly differently for the various types. Assignment of the text is done in a consistant manner so that each type if very similar to the others.

Each RTG can have 4 basic message strings attached to it. These are positioned on the Top, Bottom, Left, and Right of the rectangle covered by the RTG. When text is specified for these locations, the system automatically sizes and centers the text message such that it will appear in a consistant manner for all RTG types.

Most RTG types have the label text outside the RTG limits. The text messages for Panel and Digital Meter displays is done slightly differently in that the Top (Title) and Bottom (units) text is placed within the boundary of the RTG object. The keyword TITLE can be used in place of TOP, and UNITS can be used in place of BOTTOM.

Text labels are assigned with the keywords TOp, BOttom, LEft, and RIght. More than one keyword can appear on the same line. The following are examples:

```
RTG 5, Top, TOP MESSAGE, bot, BOTTOM MESSAGE
RTG 5:8, Bottom, MMCF/Day  ; set same units for RTG 5 thru 8
```

RTG VALUE

Each RTG has a value expression associated with it which is evaluated each time the RTG object is updated. This value can be text or numeric.

Example of valid TSP numeric expressions are:

```
A1              : channel name
BATT_VLT        ; Tag name
M1+M2           ; Expression, as in sum of 2 meter channels
2*(x+4) /Y      ; Complex calculation
```

Numeric values are compared to the Low and High limit setting to determine the display color. Results lower or equal to the low range will appear in the low color. Results between low and high limit show up in the Medium color. Results over the high limit are shown in the high color.

Text values are preceded with a special character which positions the text string:

```
|ANY TEXT       ; Text to be centered in field
<ANY TEXT       ; Left Justified Text in field
>ANY TEXT       ; Right justified text in field
```

Text expression examples are identified with a leading dollar ($) symbol to indicate a replacement variable ($T, $D, $@, etc), or a string variable ($%1, $%2, etc). The content of the string expression is determined on each update and the resulting text is put into the text area. In the case of string text displays, no evaluation of color range (lo, med, high) can be made. Text only displays are always shown in the low color.

Limits are set with an RTG LIMIT statement with the following form:

```
                3         4          5
RTG, #, Limit, low_limit, hi_limit
```

The low limit and high limit values can be provided as any valid numeric value, including constants or expressions. However, the expression is evaluated at the time the object is created. The resulting constant is used as the limit value. Therefore, referencing a channel or variable name is acceptable, but the value of the limit will be set to the resulting value at the time the LIMIT line is executed.

RTG CHANNEL

Another way to specify the value for an RTG is to assign it directly to a TSP channel. The CHANNEL (or CHAN) option is used instead of the VAL option described above. When the value is specified with CHAN, and RTG AUTO is ON, then the RTG can use the current settings of the specified TSP channel to build itself. This relieves the programmer from specifying several items:

1. Hi and Low limits for any channel type
2. Test description and Units (if a value type channel)

This similifies program maintenance because changes made to a channel configuration will automtically affect an RTG without the need to edit the procedure which uses the RTG. For example, if an RTG is attached to a battery voltage channel, the high and low setpoints for the RTG will be determined at runtime, based on the current hi-lo for that specific channel.

RTG AUTO: The automatic text assignment requires that RTG AUTO be ON, which is NOT the default. If AUTO is off, then no automatic text is inserted into the RTG object. The RTG AUTO OFF|ON command is global and remains in effect until changed by another RTG AUTO line.

Another feature of RTG channels is that the display colors will follow the alarm state of the channel rather than simply its value. When a channel is normal, the normal (mid) color is used for the display. If it goes high or low, the appropriate color is flashed with the mid color until the channel is acnkowledged. This allows graphic displays to show current alarm status as well as the channel value.

A channel RTG in alarm can be ack'd by selecting the RTG with the right mouse button. This will present a confirm box which requies the operator to specificaly ack that particular alarm. This can be used instead an overall ACK which clears alarms for the entire RTU.

The F2 ACK button press has been enhanced (June 95) to allow ACK of any channels present in an RTG graphic display. Only RTG objects defined with the CHAN option are affected. Any channel references in a VALue type object are not affected by an F2 ACK.

Objects assigned to Channels will be "caged" with a dashed box when the mouse is within range to ACk the RTG channel. The mouse cursor will not change to the Uphand unless the RTG also has a pick action assigned to it. Therefore, caged RTG's with an arrow can only be ACK'd. Caged RTG's with an UpHand can be ACk'd or Picked.

ANNUNCIATOR TYPE DISPLAY

The text presented in an RTG is normally derrived from the expression (or channel) value assigned to that RTG. A June 95 option provides for alternating text displays which can be assigned 2 (status channels) or 3 (value channels) text messages. The specified messages are put in the RTG display rather than the text resulting from the RTG's value expression. In other words, the expression is used only to determine the current range (off/on, or lo-med-hi). The text put in the RTG will be one of the Annunciator messges provided when the RTG was defined.

This feature is useful for annunciator type windows (or menu hits) which should show different text messages depending on the value of some TSP expression or channel. An RTG can be assigned to a status channel, with the resulting text selected from 1 of 2 messages, depending on the current value of the RTG.

Annunciator text is specified after an RTG object is defined, and should normally follow the CHAN or VAL line which specifes the expression associated with the RTG's value. When the RTG is updated, the expression wil be evaluated, and the appropriate annuniator text will be presented on the screen.

Text associated with annunciator displays is automatically centered unless one of the format codes <or > is put in the first position of the text.

Example:

```
RTG,+, Text, 0.5, 0.5, 18    ; text window 10 spaces wide.
RTG *, Chan, RTUBATT   ; associate with battery voltage
rtg *, annun, Low Voltage, Normal Voltage, HIGH VOLTAGE
```

In the above example, the RTG would have shown the numeric value of the voltage if the

ANNUN line hadn't been provided. The way the example is written, the text display will contain one of the three messges, depending on the current value of the channel RTUBATT.

The Annunciator keyword can be abbrevited down to "AN" if desired, althouth"ANNUN" is recommended. At least one mesasge must follow the keyword. If only one is provided, it is used for all three positions (lo,med, and hi). If two are provided, then the High is the same as the Mid. This allows status type channels to be defined with only two messages instead of three.

RTG CLEAR, HIDE, AND SHOW

RTG's are defined when a RTG TYPE keyword (Dot, Box, etc) is found in position 3 of the RTG line. Each time a new RTG is created, it is assigned a number as discussed earlier. The RTG remains in service until it is cleared. Clearing can be done for individual objects with the CLEAR keyword:

CLear — Remove and destroy the specified RTG blocks

```
RTG 5:7 CLEAR  ; clear rtgs 5 through 7
```

Clearing an RTG removes if from the screen as well as from the active RTG list. Once cleared, an RTG cannot be referenced again without a complete reinitilization. Most often, RTG's will be defined at the start of a procedure and left in effect until the screen is removed. However, this option is provided to allow individual removal and possible reprogramming of individual RTG's while the screen is still active.

It is also possible to hide an active RTG so that it is temporarily removed from service. The definition remains in place while the RTG is hidden, but no update or mouse pick action is performed. The RTG is available for activation with a SHOW command, which is the opposite of the HIDE command.

HIde — Temporarily remove an RTG from the screen

```
RTG 5 Hide
```

SHow — Restore a Hidden RTG to the screen.

```
RTG 1:4 Show
```

MISC RTG OPTIONS

Several other keywords are available to override settings provided in the RTG initialization line, and to provide other information used by various RTG types. These options are entered on subsequent lines following the RTG statement which created an RTG. Normally, all lines associated with a particular RTG are processed one after another as a group. However, because the specific RTG number can be entered on each RTG line, it is possible to group statements to suit the programmer.

The RTG options are entered with a keyword followed by a value. Multiple option lines can be used on a single RTG, and these lines can usually be processed at any time, including after the RTG is on-screen. This allows dynamic reprogramming of RTGs in certain cases where the new option affects the update procedure, and not the initial display procedure.

Note that some of these options need never be used because equivalent functions are provided by other RTG statement types. They are mostly used to override settings in specific RTG's which were duplicated from a previously defined RTG (see COPY and ARRAY Below).

The option keywords follow, with significant characters in upper case:

UPdate — Update the specified RTG.

```
RTG 5 Update     ; update a single RTG
```

FILL — Specify a new fill color.

```
RTG 17 Fill Black  ; change fill color to black
```

DRaw	Specify a new draw color. RTG 3 Draw White
TExt	Specify color for Text associated with an object RTG 11 Text Black ; show text in black
ON	Specify a new text background color (normally used with draw); RTG 6 Draw White On Blue
MAsk	Specify thee text format 'mask' used to align characters. RTG 5 Mask ####.## ; show 4 digits and 2 decimals
TOp (or Title)	Specify top Label for an RTG (Title area in Panels) RTG 5, Title, Gas Sales Rate ; note use of commas
BOttom (or Units)	Specify Label Text for bottom (Units area in panels) RTG, 6, Units, Volts RTG 7, Top Battery Voltage, Bottom, Volts
LEft	Specity label text to appear to left of an RTG. RTG 9, Left, Start Pump
RIght	Specify label text to appear to right of an RTG RTG 11, Left, SALES RATE, right, MMCF/Day
PRompt	Provide prompt to display when confirming mouse pick. This forces a prompt to appear regardless of MOUSE CONFIRM OFF/ON setting. RTG 3, prompt, Open Valve?
ALign	Set text alignment within specified area to left, right, or center. RTG *, Align, Left
FRame	Specify Frame type as none, single, in, out, or double. RTG 5, frame, out RTG 7, frame, double

RTG COPY

Often it is required to put up numerous RTG object which are very similar and differ in only one or two respects, such as the calculated value. The RTG COPY function allows a single RTG to be defined in detail only once, and then copied and modified to suit other applications on the same screen. For example, a particular size and color set can be determined for a Panel display and specified in one set of RTG statements. Other, similar RTG's can be easily defined by copying the prototype and changing only the components of the RTG which differ.

Copying an RTG requires only the existing source RTG number, and the screen location for the new copy. All aspects of the original are copied except the X-Y location. It is possible to override the Anchor position in the COPY line such that the new one will be anchored at a different point than the original. Syntax for the RTG copy is as follows:

```
        2          3      4          5          6
RTG new_number, COPY, source_ number, x_position, y_position
```

New_number	Number of RTG to create (can be + sign)
Source_number	Number of RTG to be copied (can be - sign)
X_position	X coordinate for new RTG
Y_position	Y coordinate for new RTG

Consider this small program segment:

```
RTG +, Panel, 100,20, 8,. double, Med  ; build a text box
RTG *, value, A7       ; show value of channel A7
RTG *, mask ###.##
rtg *, title, Battery No. 1, units, Volts
RTG *, color, white, red, green, black
; end of definition

RTG +, copy, *, 100, 50
rtg *, value, A8       ; only difference is location and channel name
rtg *, title, Battery No. 2
```

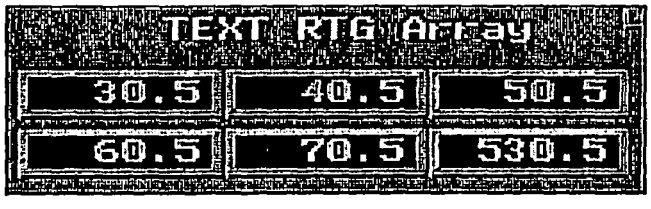
This example uses 5 lines to generate a new Panel RTG. A second similar RTG is copied to a different location, with a change necessary to the Title and Value only. all other aspects of the original RTG are maintained.

Note the use of + and * to reference the RTG numbers without identifying their exact value. The system tracks the current and previous numbers so that "generic" code can be written and duplicated without error.

RTG ARRAY

RTG ARRAY allows for fast, accurate placement of a rectangular array of similar RTG's. ARRAY is similar to COPY, except that ARRAY generates multiple RTG's in a rigid layout instead of allowing free placement of a single copy.

Generating an ARRAY is done by specifying a range of RTG's to be generated, followed by the source RTG, similar to COPY. Also necessary is the column count of the desired array. The system will determine the required number of rows depending on the number of RTG's being generated. RTG numbers are assigned in sequence, left to right, top to bottom. The source RTG will be in the upper left corner, with the first new one in the next position to the left (or just below the source if a 1 column array).



When the ARRAY has been generated, the RTG number is set to the first new one which was created as a result of the array. This lets the user make the necessary changes for the first new one, then use + in subsequent lines to move on to the other newly created array members.

The syntax of the ARRAY statement is:

```
                  2       3       4          5        6        7
RTG start_rtg:end_rtg, ARRAY, source_rtg, num_cols, [x_space], [y_space]
```

Start_rtg	First new RTG to be created
end_rtg	Last new RTG to be created
Source_rtg	Number of RTG to be copied into the array
Num_cols	Number of columns in the array
X_space	Optional X space (pixels or percent) between array columns
Y_space	Optional Y space (pixels or percent) between array rows

Consider this fragment:

```
RTG +, Panel, 100,20, 8, double, Med  ; build a text box
RTG *, value, A7      ; show value of channel A7
RTG *, mask ###.##
rtg *, title, Battery No. 1, units, Volts
RTG *, color, white, red, green, black
: end of definition

RTG +:3, array,*,  2, 10, 5
rtg *, value, A8      ; First new member reads channel 8
rtg *, title, Battery No. 2
rtg +, value, A9      ; second new member reads A9
rtg *, title, Battery No. 3
rtg +, value, A10     ; third new member reads A9
rtg *, title, Battery No. 4
```

This example is similar to the copy above in that a single Panel is created and configured as desired. The ARRAY statement creates 3 more similar objects with the previously created one in the top left corner of a 2 x 2 matrix. Note that the array adds 3 to the one existing RTG, for a total of 4. The lines following the RTG serve to modify the newly created ones, with a + sign used to increment the RTG number counter down the list.

Note: An Array of 1 is not allowed (as of June 95). The resulting RTG count must be 3 or more (1 plus the original RTG).

RTG MOUSE PICKS

Each RTG object can have a mouse hit action associated with it. RTG objects which are assigned mouse pick commands will become active whenever the mouse is within the rectangular boundary of the RTG object. This is noted by a change in the mouse symbol from the normal arrow (or other system default) to an UPHAND symbol. Whenever the UPHAND is visible, the mouse action can be selected with the left mouse button.

If the system setting MOUSE CAGE ON is in effect, a dashed box is displayed around the rectangular area of an active RTG whenever the mouse is within range. This cage effect allows transparent pick areas to be defined at any position of the screen, perhaps overlaying a graphic image which denotes various choices.

The syntax for the pick option is as follows:

```
1   2  3      ------ 4 -------
RTG, #, Pick, Any Valid TSP Command or special Pick option
```

Note that everything on the command line following the PICK keyword is stored as the pick command. The first character of the command can be an optional special code which indicates either an internal RTG function, or a prompt dialog override. These special codes are:

!	Force a confirm dialog, regardless of system setting
*	Never present confirm dialog, regardless of system setting
?	Special RTG internal function (only 1 at present)
	?Exit Terminate RTG display

The system setting for MOUSE CONFIRM ON will determine if every pick action will be prompted with the confirmation dialog. If on, then the user is required to acknowledge every pick action prior to execution. The ! override can be provided in the PICK command string to either force a confirmation, or * can be used to prevent one, regardless of the default system setting.

The ?EXIT is a special RTG command which effectively terminates the current RTG screen. This is handy for menu hits to exit the screen and return to the normal SCADAWARE prompt or menu system.

RTG MOUSE PROMPT

Mouse picks which require confirmation will use a default prompt dialog box which says "CONFIRM PICK ACTION?" This default prompt can be changed with a RTG PROMPT line such that a custom message can be presented during the confirmation dialog. Providing a custom prompt has the side effect of forcing the prompt to appear whenever the mouse is hit for this RTG. Providing a prompt causes the same effect as using the bang ! symbol at the start of the pick command.

Consider this fragment:

```
RTG *, pick, Poll RTU Now
RTG *, prompt, Want to poll the RTU?
```

When this sequence is used on an RTG, a mouse click will cause a dialog box to appear near the corner of the RTG closest to the center of the screen. The dialog box will show the text string "Want to poll the RTU?" and wait for a Yes or No key-hit or mouse click. The dialog will remain on the screen for the number of seconds specified in the last MOUSE CONFIRM OFF/ON SECS command.

GRAPHIC SCREEN PRINTOUTS

Graphic screens are printed with the GPRINT command. This function takes the VGA color screen image and converts it to a black and white image suitable for output to a laser or dot-matrix printer. Output quality will depend on the complexity of the image, the colors, and the printer capability. Generally speaking, larger images appear better on the printed page. However, the GPRINT command allows scaling of screen images to any size, so some experimentation may be required to find the best combination of quality and size.

The GPRINT command can only be used with task 0 is in graphics mode. It is typically placed in the PICK statement of an RTG menu (box) element, although any valid command method can be used. GPRINT occupies all of the task's processor time while the graphic image is being generated and printed. A "percent complete" display is provided to show progress during the printing process.

The syntax of the GPRINT command is as follows:

GPRINT [keyword with option(s)] ... [keyword with option(s)]

GPRINT with no parameters will print the complete CRT display directly to the printer. Options are available to set the screen area to print, the output size on the paper, margin offsets, and printer driver. Options also exist to allow printing to a disk file rather than the physical printer.

The GPRINT keywords (2 significant letters) and their associated options are:

ARea x1 y1 x2 y2	Designate screen area to print, in percent or pixels
TO filename	Print to specified file, erasing any existing similar file.
APpend filename	Add printout to existing file.
DRiver drivername	Overrides the default graphics printer set with SET GPRINT
Eject	Eject the printed page when output is complete.
SIze x_dim y_dim	Specifies the X and Y dimension of the printout, in inches.
DOwn x_dim	Specifies page offset from top, in inches.
LEft y_dim	Specifies page offset from left, in inches.
STOP	Halt printing of the current screen.

Examples:

```
GPRINT Area, 0.1, 0.1, 0.9, 0.9, Down, 1.5, left, 1.0, size, 4.5, 3.0
```

The example prints a screen area 10% from the top left screen corner, to 90% of the maximum screen dimension. The output is offset on the printed page 1.5 inches down, 1 inch from the left. The output image will be scaled to fit a rectangle 4.5 inches across, 3 inches high.

The printer driver used by GPRINT defaults HPLSRH, to a high resolution (150 x 150) HP LaserJet. Many other drivers are available in the printer library, called RTUPRINT.LIB, which must be located in the same directory as the graphic screens. This directory defaults to ..\SCREENS\. This directory also contains the font file, RTUFONT.LIB.

A list of some common drivers are provided below. Note that many printer models emulate a popular printer such as a Hewlett-Packard or EPSON.

PRINTER	RESOLUTION	MODE	DRIVER NAME
LaserJet / DeskJet	75 x 75	B&W	HPLSRL
LaserJet / DeskJet	100 x100	B&W	HPLSRM
LaserJet / DeskJet	150 x 150	B&W	HPLSRH
LaserJet / DeskJet	300 x 300	B&W	HPLSRVH
LaserJet 4	600 x 600	B&W	HPLSRVVH
LaserJet 4 Postscrip	300 x 300	B&W	PS
DeskJet 500C/550C	75 x 75	Color	HPDSKCL
DeskJet 500C/550C	100 x 100	Color	HPDSKCM
DeskJet 500C/550C	150 x 150	Color	HPDSKCH
DeskJet 500C/550C	300 x 300	Color	HPDSKCVH
EPSON LQ, SQ, Action	120 x 120	B&W	EPSON2M
EPSON LQ, SQ, Action	180 x 180	B&W	EPSON2H
EPSON LQ, SQ, Action	360 x 360	B&W	EPSON2VH
EPSON FX, MX	120 x 120	B&W	EPSON9M
EPSON FX, MX	240 x 216	B&W	EPSON9VH

Contact TEST for information on the many other printer drivers included in RTUPRINT.LIB.

Note that the SET GPRINT driver command can be used to change the default printer driver. Driver changes made with the SET option are permanent. Changes made on an individual GPRINT command line affect only that printout.

---------- END ---------