Reference Guide
D301491X412
April 2012

OpenEnterprise Graphics Introduction Reference Guide (V2.83)

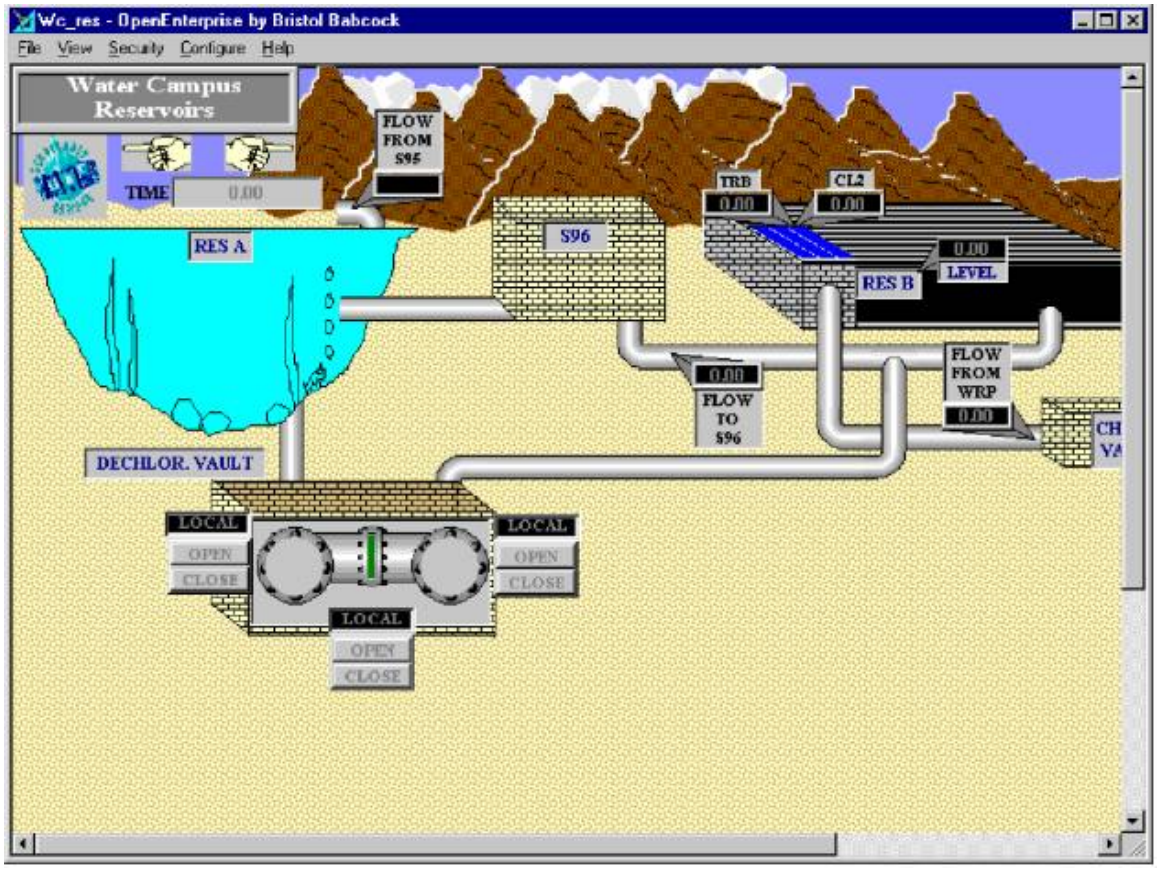
Remote Automation Solutions

Website: www.EmersonProcess.com/Remote

Contents

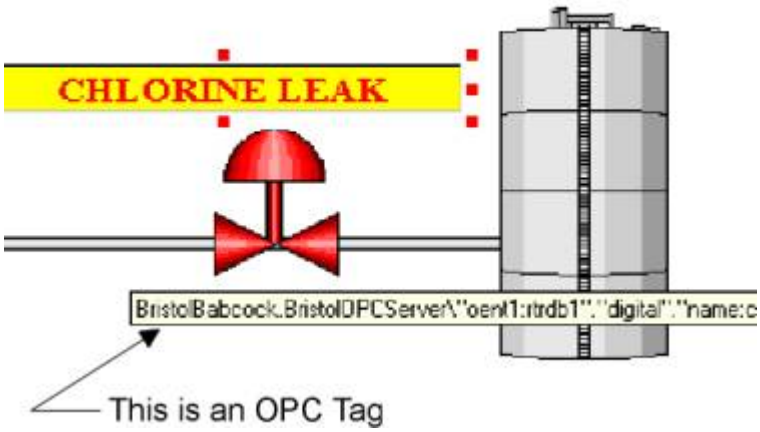
1 Graphics Introduction

The **OpenEnterprise Graphics** software (OEGraphics) is a specially licensed version of the Iconics GraphWorX32 software, which has been specifically modified to support additional functionality. It is used to build and view graphical **displays**, sometimes known as mimic displays, which allow operators to monitor their plant or process. The entire collection of displays used by the operator is referred to as the **human-machine interface (HMI)** or **graphical user interface (GUI)**.



Data is collected from the plant or process via instrumentation devices (e.g. flowmeters, electrical contacts, pressure transmitters) and transmitted to remote process controllers (often known as **RTU**s). Data from the RTUs is then sent to the OpenEnterprise Database, where it can be extracted for use on displays.

Displays consist of a mixture of **static** and **dynamic** objects. These objects are drawn to resemble specific parts of the plant or process such as pumps, valves, compressors, tanks, etc. Static objects do not change, and are not connected to the OpenEnterprise Database. Dynamic objects, however, are linked to 'live' real-time process data in the OpenEnterprise Database through the use of **Object Linking and Embedding (OLE) for Process Control Tags**, or just **OPC Tags**. The dynamic objects can be configured to change color, size, or flash based on how the real-time data changes. For example, an analog value might flash when it has entered an alarm state, or a pump might turn green when it is running, and yellow when it is undergoing maintenance.



Besides allowing an operator to monitor the plant or process, displays can be configured to help the operator control the process by allowing data entry of setpoints, by allowing the operator to start and stop devices depicted on the display such as pumps, motorized valves, etc. In each case, these actions are possible by sending commands from the display down to the OpenEnterprise Database, and then on to the controller network.

Objects which are to be used on multiple displays can be saved as **symbols** in the **symbol library**. They can then be easily imported into any new display which is created. Symbols and displays can also use a technique called **aliasing** which allows symbols and even entire displays to be re-used with *different* entries from the database, depending upon which alias (name) is currently selected.

This manual presents an overview of the display building process, including:

- instructions for starting the Database Explorer and constructing queries of the Database in the Database Object Viewer. (This is necessary in order to include live data on displays via OPC tags.)
- instructions for starting the OpenEnterprise Graphics software, opening a display, changing from ‘configure’ to ‘runtime’ mode, saving the display, etc.
- step-by-step examples for creating various display objects. Various features of OpenEnterprise Graphics are highlighted as part of the examples.
- instructions for working with symbols, setting up aliasing of individual symbols, saving them in the symbol library, and re-using them on other displays.
- suggestions for designing your entire human-machine interface (HMI) system.

IMPORTANT

This Help file is NOT intended to be a comprehensive reference to all features and functions of the OpenEnterprise Graphics package. If you need information on a particular feature not covered here, please refer to the *GraphWorX32 online help*.

1.1 Before You Begin

Before you attempt to build displays, the following must have been done:

You must have installed the OpenEnterprise Graphics package on your workstation. This process is briefly covered in the *OpenEnterprise Installation Guide* (document# D5090-1).

- Your OpenEnterprise Database must have been created and must be running. (Normally, this is done as part of the OpenEnterprise installation process. See the *OpenEnterprise Installation Guide* document# D5090-1.)

- You must know the name of the OpenEnterprise Database (or the IP address *and* name, if it resides on another computer). The default database name is 'rtrdb1' however the name used on your system may be different. Consult your site-specific documentation for details.

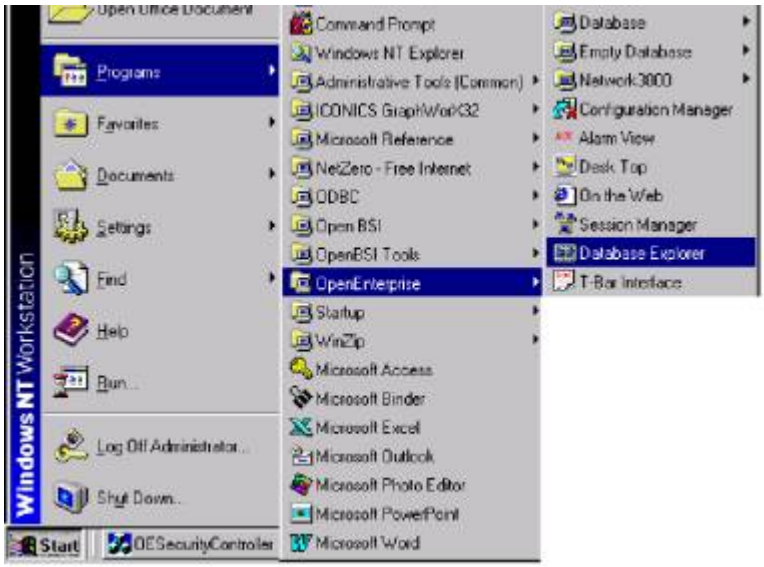
- You must have successfully logged on to your OpenEnterprise Workstation, with sufficient security privileges to access the OpenEnterprise Database. Consult your site-specific documentation for details.

- You must be familiar with how to connect to the database using Database Explorer and how to construct queries of the database using Database Explorer's Database Object Viewer. (This process will be reviewed in this manual; for a full discussion, see the *OpenEnterprise Database Explorer User Guide* document# D5090-2.)

- You must be familiar with the details of the underlying plant or process which will be depicted on the displays, and also must know which tables and attributes of the OpenEnterprise Database contain the data you want to include on the display.

2 Using Database Explorer

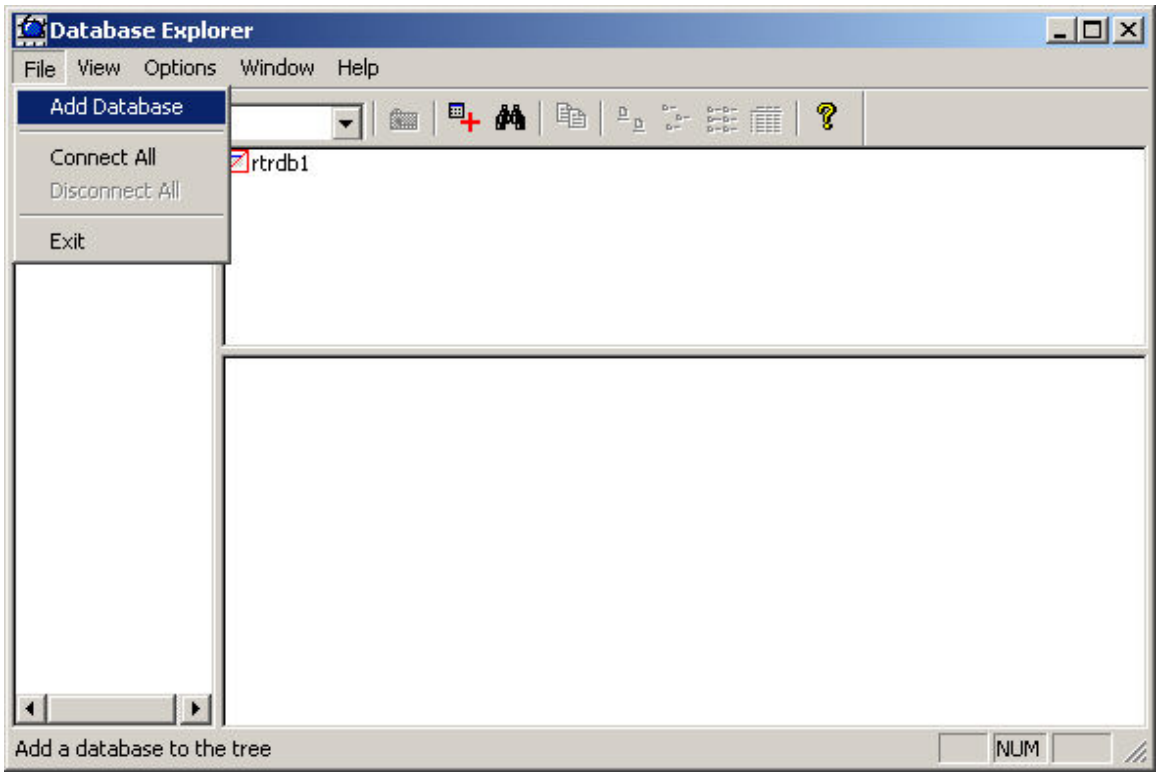
To start Database Explorer, click on **Start→Programs→ OpenEnterprise → Database Explorer**.



If this is the first time you are using the Database Explorer with this particular OpenEnterprise Database, you must explicitly identify for Database Explorer the name and (if located on a computer other than the current one), the location of the Database. A connection must then be requested with the OpenEnterprise Database.

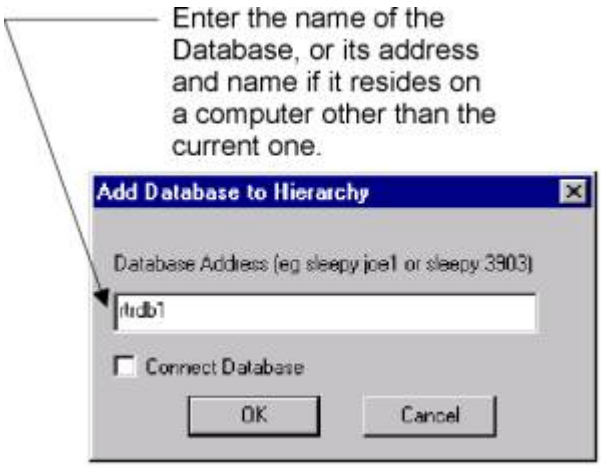
2.1 Adding Your Database

If this is the first time you are using Database Explorer with a particular OpenEnterprise Database, you must add the database to the tree hierarchy of databases accessible to Database Explorer. To do so, click on the 'Add Database' icon, shown above, or click on **File→Add Database**. The Add Database to Hierarchy dialog box will appear.



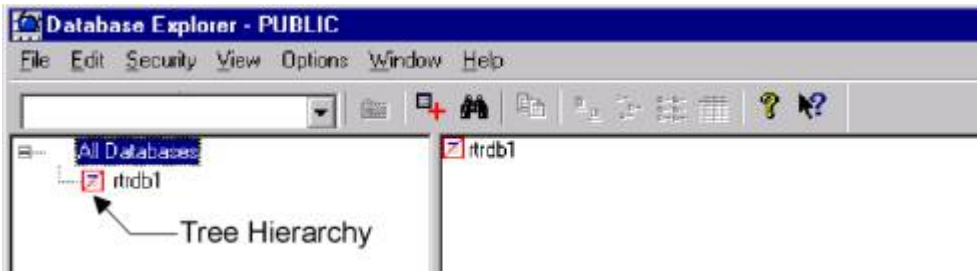
2.2 Using the Add Database to Hierarchy Dialog Box

The Add Database to Hierarchy dialog box allows you to specify, for Database Explorer, the name of your OpenEnterprise Database. In addition, if the OpenEnterprise Database resides on a PC other than the current one, you must also specify its location.



In order to use Database Explorer with your OpenEnterprise Database, it must be made aware of the name of the database, and its location. In the Add Database to Hierarchy dialog box, enter the database name, for example 'rtrdb1', or if the database resides on a different computer than the one you are currently using, the name of the database must be preceded by the IP address of the other computer, for example: '120.0.210.10:mydatabase'. You may also use an alias for the IP address, for example if the text 'MYSERVER' has been defined in your HOSTS file as being equivalent to an IP address of '125.45.2.8', you could enter either 'MYSERVER:rtrdb1' *-or-* you could enter '125.45.2.8:rtrdb1'.

To automatically connect with this database choose the **"Connect Database"** option. Click on **[OK]** to exit the dialog box. The database will be added to the tree hierarchy.



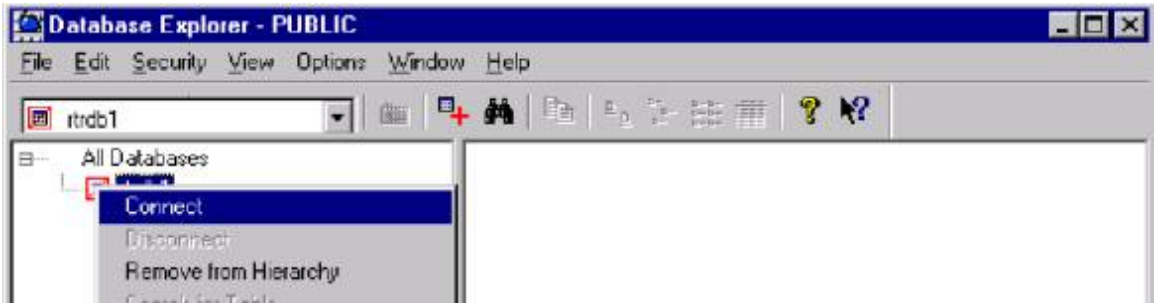
The database icon will be surrounded by a RED box with a diagonal line through it, until an actual connection is established.

2.3 Establishing A Connection With the Database

Once a database has been identified for Database Explorer by adding it to the tree hierarchy, a connection with the Database must be established so that the Database structure can be accessed.

NOTE

Before attempting to establish a connection with a Database, the Database must be running. If it is NOT running, you must explicitly start it. In addition, you must have logged in with security access for this particular Database.



To establish a connection, click on the icon for the database in the tree hierarchy, then press the right mouse button, and choose **"Connect"** from the pop-up menu. (NOTE: This will not be necessary if the **"Connect Database"** option was selected when the database was added.)

When the icon for the database is surrounded by a GREEN box, the connection has been successfully established.

IMPORTANT

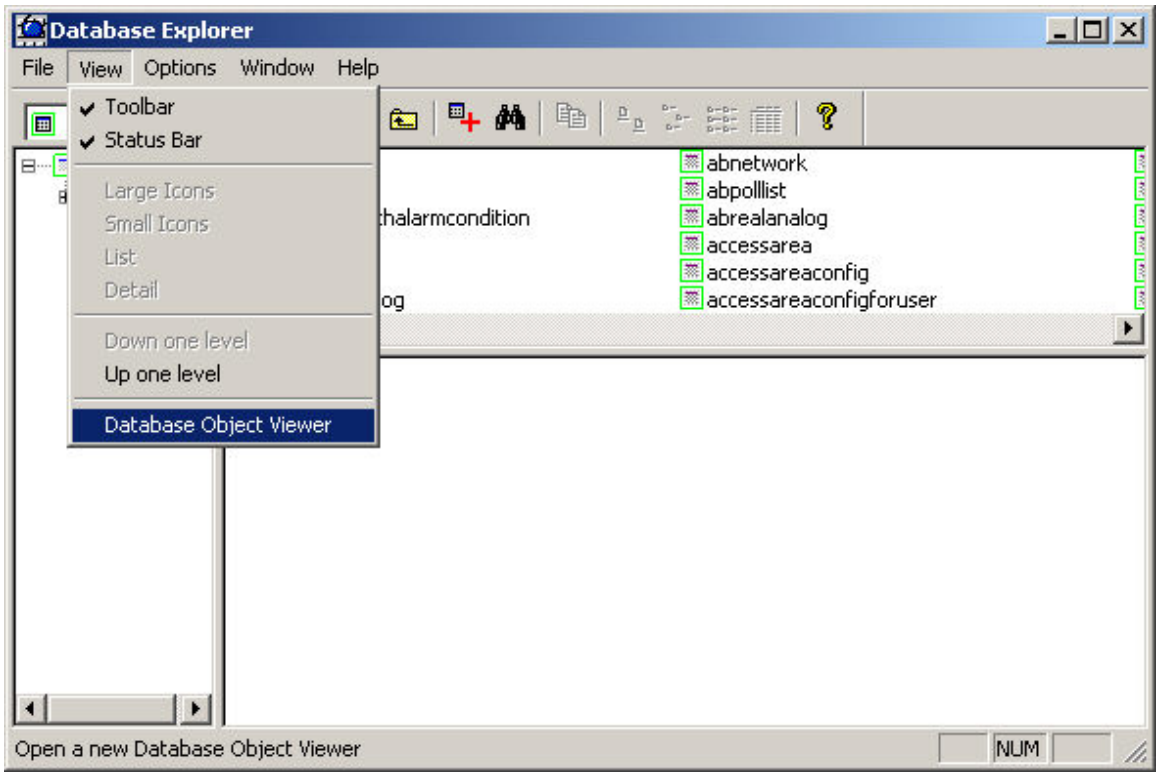
Once a connection has been successfully established, Database Explorer will issue a **query** to the database to obtain details on its internal structure. This structural information is retained if the database is disconnected. Whenever a re-connection is made, the query will be issued again to update the structural detail information.

If, while connected, the database is altered by some external tool (such as the SQL Client) queries to the database may NOT be successful, depending upon how the database was changed. Should this occur, users should disconnect, and then re-connect so a new query can be initiated.

Now that you have successfully started the Database Explorer, and connected to the Database, you can start the Database Object Viewer to query the Database for data that can be used on your displays.

2.4 Starting the Database Object Viewer

Once the Database Explorer has been started, *and* a successful connection has been made to your running database, you can start the Database Object Viewer by clicking on the binocular icon on the toolbar, shown above, or on the **View→ Database Object Viewer** menu.

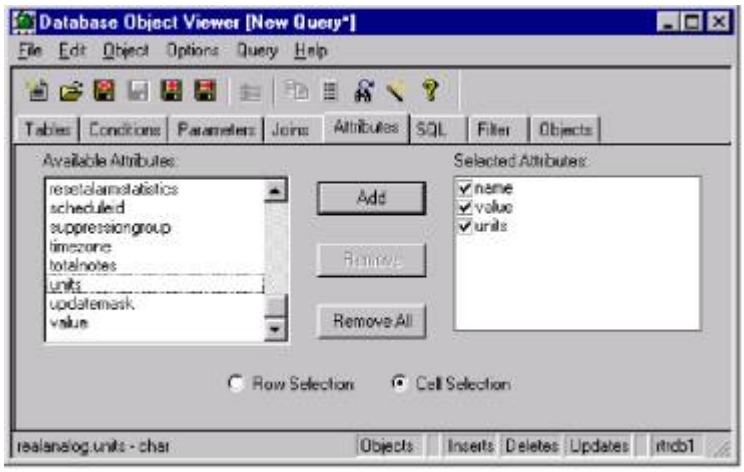


The Database Object Viewer will appear, with the 'Tables' page displayed first. You can now proceed to query the database for data.

2.5 Querying the Database Using Database Object Viewer

2.5.1.1.1.1 Step 1. - Start creating your query by selecting the Table containing the data:

On the 'Tables' page of the Database Object Viewer, there are numerous tables listed. Typically, you will want to select *either* the 'realanalog' table if you are want to display data from ACCOL analog or analog alarm signals, *-or-* the 'digital' table if you want to display data from ACCOL logical or logical alarm signals. When you have selected one of these tables, click on the **[Add]** push button.



IMPORTANT:

DO NOT select more than one table per query, unless the tables you choose are very small, or you are performing a 'Join' operation, otherwise you can potentially exceed the system's available memory, or overload communications by creating an extremely large query.

2.5.1.1.1.2 Step 2. - Choose the Attributes included in your query:

On the 'Attributes' page of the Database Object Viewer, click on the desired attribute in the **"Available Attributes"** list box, then click on the **[Add]** push button, and repeat this process for each additional attribute you want to view.

For most of the examples in this manual, the following attributes are required: 'name', 'value', and 'units'.

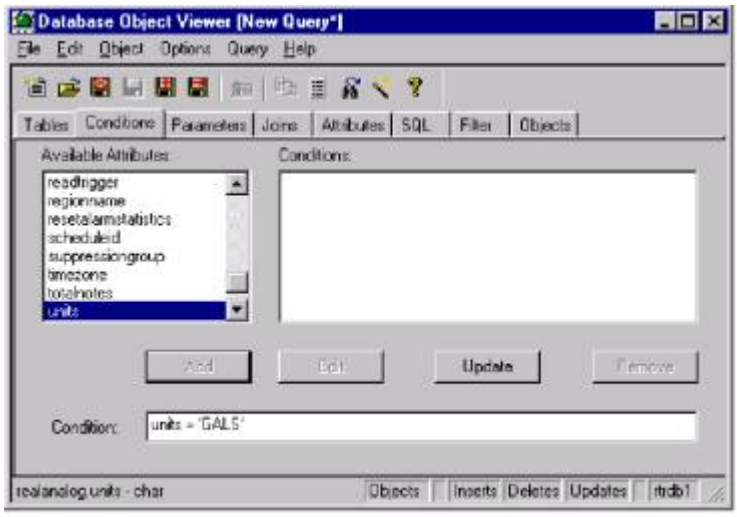
2.5.1.1.1.3 Step 3. - Specify Conditions of the query:

On the 'Conditions' page of the Database Object Viewer, you can specify that you only want table entries where certain attributes meet some specific criteria.

For example, if we only want data from those signals which we know are in units of gallons 'GALS', we can put a condition on our query that units must equal 'GALS'. To do this, click on 'units' in the **"Available Attributes"** list box, then click on the **[Add]** push button.

The word 'units' will appear in the **"Condition"** line, add the following text to the line:

= 'GALS'



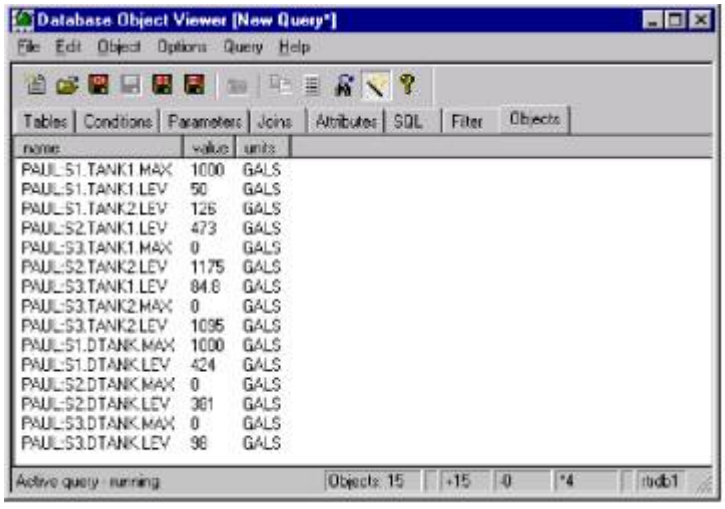
Now, click on the **[Update]** push button, and the condition you made will appear in the **"Conditions"** box above.

The 'Conditions' page is OPTIONAL. If you do not specify any conditions, however, you will receive a warning that no conditions have been specified. This warning can be ignored; it is displayed because if you have a large database with many signals, the response to your query could be very large, thereby placing a burden on communications.

2.5.1.1.1.4 Step 4. - Run the query:

Click on the 'Active Query' icon, and then run the query by clicking on **"Query"** in the menu bar, and **"Apply"** in the pull down menu, or just click on the 'Run Query' icon. The results of the query will appear on the 'Objects' page of the Database Object Viewer.

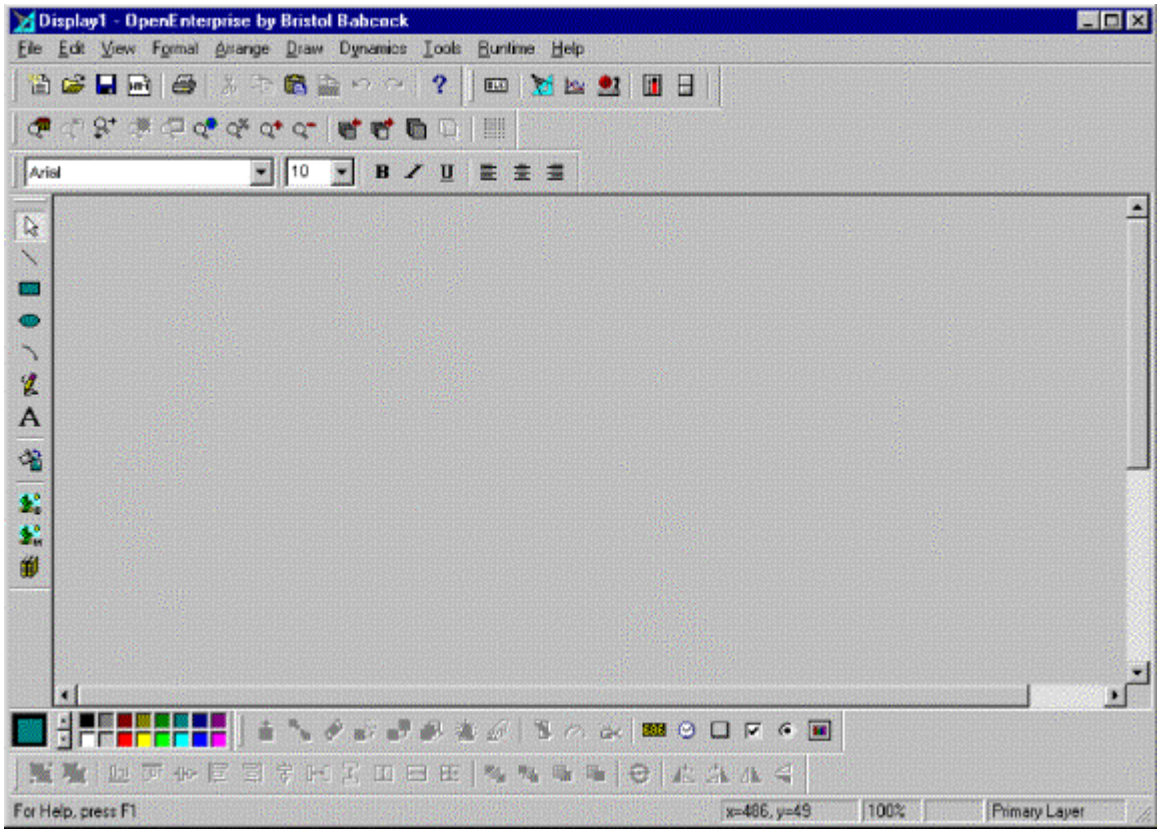
IMPORTANT: Keep the 'Objects' page of the Database Object Viewer (which contains your query results) visible on the screen; you will need it in order to add 'live' data to displays in the OpenEnterprise Graphics package.



3 Starting OpenEnterprise Graphics

By default, the startup sequence for the OpenEnterprise Graphics package is to click on **Start→Programs→OpenEnterprise→Workstation**. This sequence may have been modified, however, when OpenEnterprise Graphics was initially installed.

If you cannot find OpenEnterprise Graphics, look for GraphWorX32, and start it, instead. After the splash screen has appeared, the screen should appear as shown, below, with an all-new empty display open.



4 Opening a Display

Once OE Graphics is running, you can choose to create an all-new display, or you can open an existing display.

4.1 Opening An All-New Display

When you first start OE Graphics, an all-new empty display is opened automatically, and the screen will appear as shown at the top of this page.

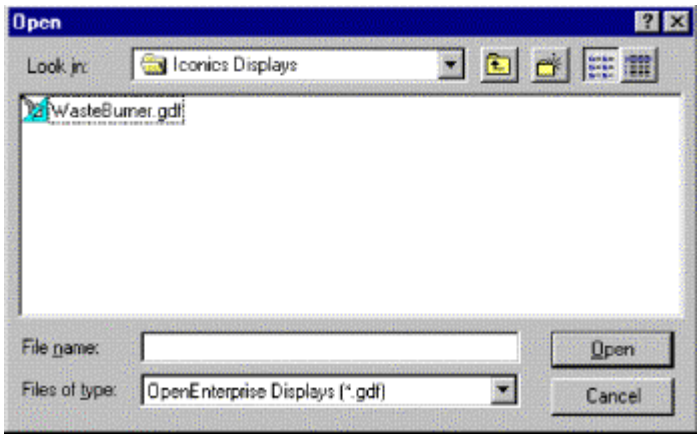
If you already have other displays open, however, and want to create an all-new display, you can do so by clicking on the icon shown at left, or by clicking on **File→New**.

An empty display will now be opened, and you can begin to add graphic elements to it.

4.2 Opening An Existing Display

To open a display which has already been created, click on the Open icon, shown above, or click on **File→Open**.

The standard Windows™ Open dialog box will appear.



Use the controls to locate the directory containing your displays. The recommended directory for storing OpenEnterprise displays is **\OE Store\ Displays** so you might want to look there first.

When you find the display you want to open, double-click on the name of the display file (must end in '.GDF'), *-or-* click once on its name, then click on the **[Open]** button, and the display will appear on the screen.

4.3 Using the Example Displays

If you can't locate any displays, the OpenEnterprise Installation CD comes with a selection of example displays, which you may use as a starting point for creating your own displays.

The example displays are located in the **\Examples\Workstation** sub-directories on your OpenEnterprise Installation CD.

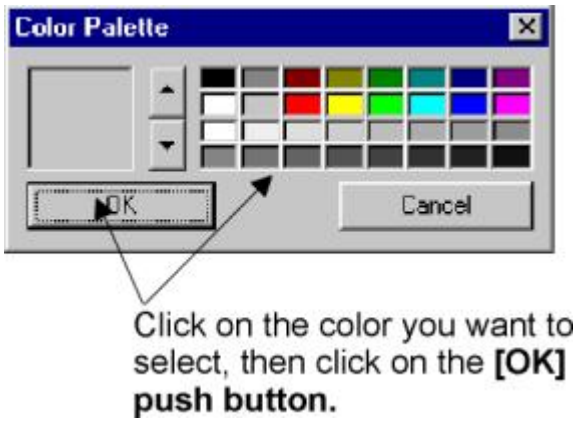
5 A Quick Look At the Drawing Tools

This section will briefly discuss the most commonly used drawing tools. For more detailed information, please see the *GraphWorX32 User Manual.*

5.1 Choosing Colors

One of the first things you will want to do when you open up a display is to set the background color of the display. To do this, click on **Format → Background Color**.

The Color Palette will appear. Click on the color of your choice for the background, then click on the **[OK]** push button.



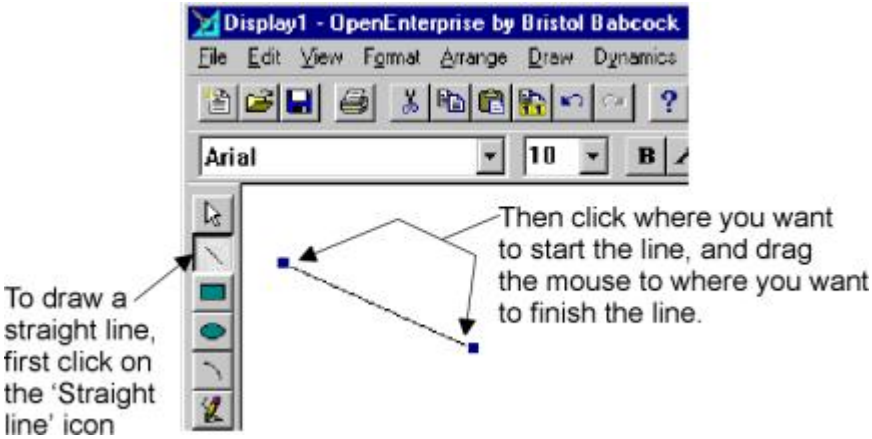
The **"Format"** pull down menu also has selections for setting the line color, line style, display properties, etc. For more details, see the *GraphWorX32 User Guide*.

5.2 Drawing A Straight Line

To draw a straight line, click on the 'Straight Line' icon, then click in the display where you want the line to begin, and drag the mouse until reach the desired endpoint, then release the mouse.

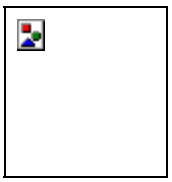
To change the line color, you must use **Format→ Line Color**, and select the color from the color palette.

To move the line after it is finished, select it and drag it to the desired location.

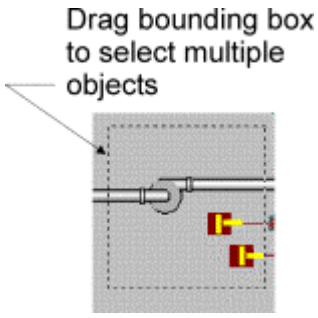


5.3 Selecting Objects on Displays

To select an object on a display just click on it. A group of eight edit handles will appear around the object. (The picture at right shows a pump symbol surrounded by edit handles.) You can click and drag on these handles to re-size the object in various directions. To move the object, just click on the object (not the handles) and hold down the left mouse key while you move the object to the desired new position.



To select more than one object, drag the mouse to form a bounding box around the objects you want to select, then release the mouse button and all the complete objects in the box will be selected.



If a display contains many objects close together, and you are unable to select the right one, you can select any object on the display, and repeatedly press the **[Tab]** key. As you keep pressing the **[Tab]** key, a bounding box will move through each of the objects on the display, allowing you to eventually select the desired object.

5.4 Drawing A Box

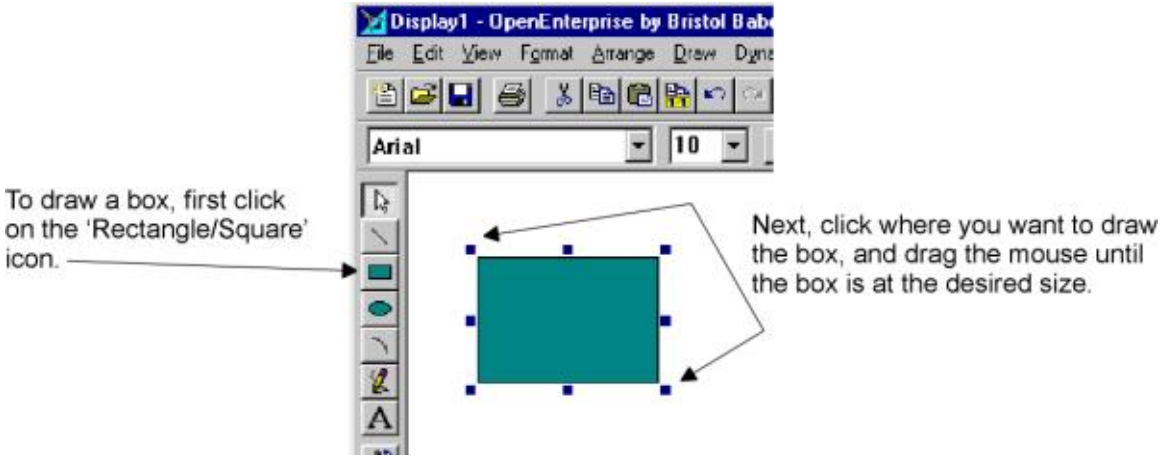
To draw a box, click on the 'Rectangle/Square' icon, then click in the location in the display where you want to draw the box, and drag the mouse until the box is the desired size, then release the mouse.

To draw a square, follow the same procedure, except hold down the **[Shift]** key while you are dragging the mouse.

If you want to re-size the box, you can drag on the handles which appear around its edge when it is selected.

To move the box after it is finished, select it, and drag it to the desired location.

If you want to change the fill color of the box, select the box, then click on the desired color in the color palette. If you want to turn on/off the fill color, you can click on **Format→Toggle Fill** (or just click on the 'Fill/Unfill' icon).



5.5 Drawing An Ellipse or A Circle

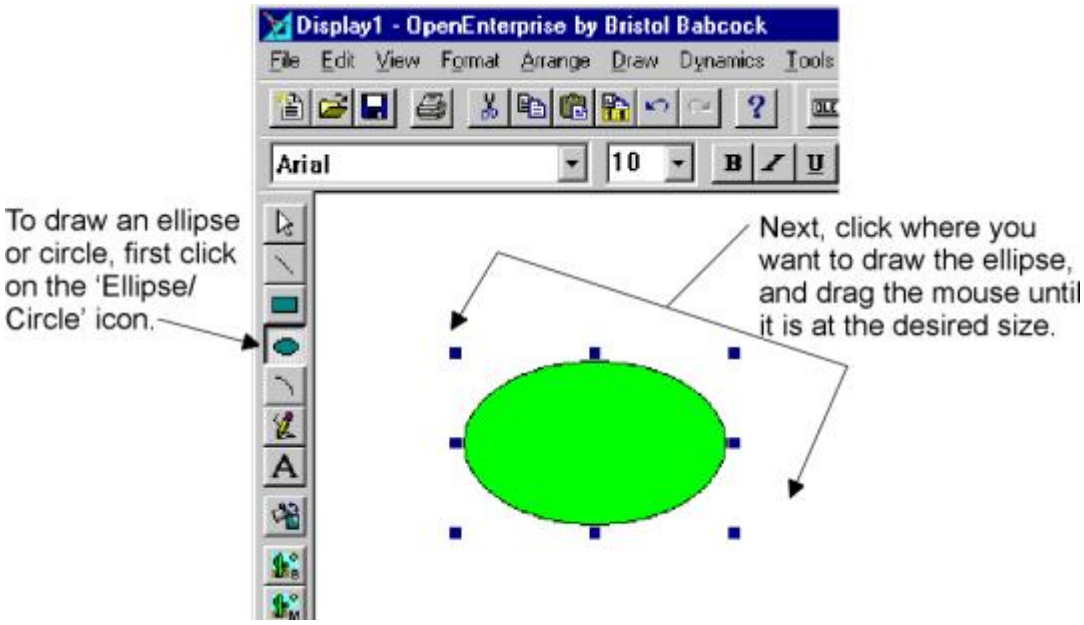
To draw an ellipse or circle, click on the 'Ellipse/Circle' icon, then click in the location in the display where you want to draw the ellipse, and drag the mouse until the box is the desired size, then release the mouse.

To draw a circle, follow the same procedure, except hold down the **[Shift]** key while you are dragging the mouse.

If you want to re-size the ellipse/circle, you can drag on the handles which appear when it is selected.

To move the ellipse/circle after it is finished, select it, and drag it to the desired location.

If you want to change the fill color of the ellipse/circle, select it, then click on the desired color in the color palette. If you want to turn on/off the fill color, you can click on **Format→Toggle Fill** (or just click on the 'Fill/Unfill' icon).



For information on the other drawing tools, see the *GraphWorX32 User Manual*.

5.6 Saving A Display

To save a display, click on the 'Save' icon, shown above, or click on **File→Save**. The Save or Save As dialog box will appear. It is recommended that you save your displays in the directory: **\OE Store\ Displays**

6 Configure Mode and Runtime Mode

Configure Mode is used to build your displays, add dynamic data, etc. Live data is NOT visible in Configure Mode.

NOTE: In order to make changes to your display, or to add 'live' data to your display, the OpenEnterprise Graphics package must be in 'Configure' mode - - if you are in 'Runtime' mode, click on the 'Configure' menu bar item to return to 'Configure'. If there is no 'Configure' item but you see a 'Runtime' item, you are *already* in 'Configure' mode.

7 Examples of Creating Various Display Objects

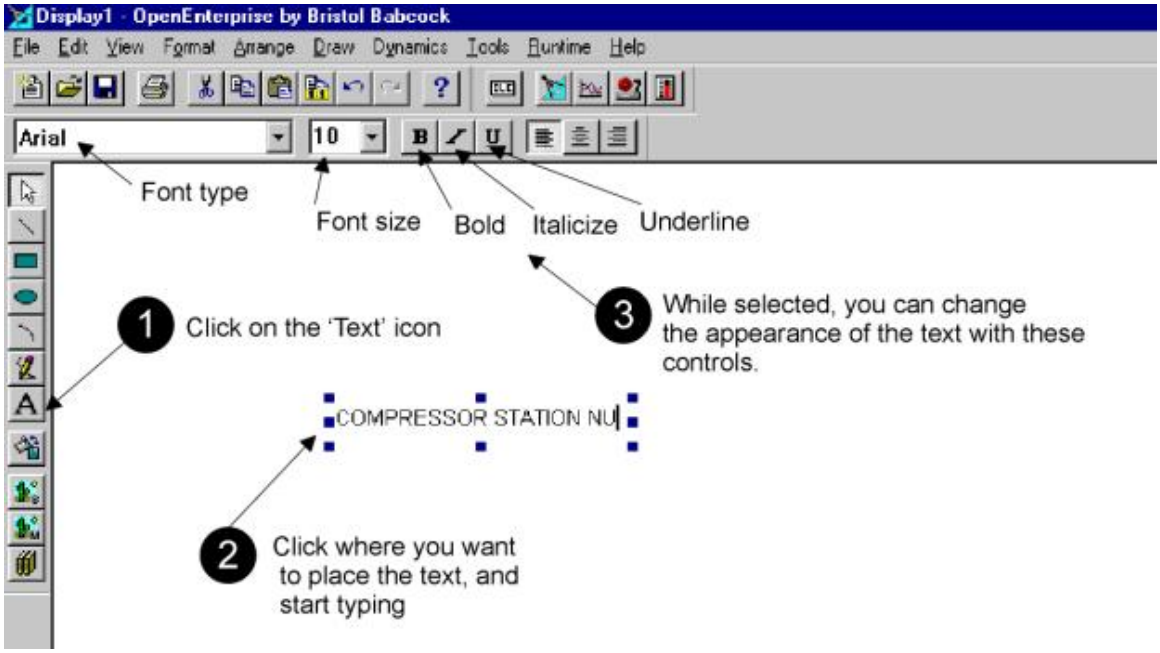
The examples which follow should generally be performed in consecutive order, since most examples build on the information presented in the previous examples.

Each of the examples assume you have successfully connected to your running Database via Database Explorer, and that you understand how to query the database for information using the Database Object Viewer. If you are unsure about these things, see Querying the Database Using Database Object Viewer.

The examples also assume you have already started the OpenEnterprise Graphics package, that you have opened up a display, that you know how to save it, and how to switch back and forth between 'Configure' mode and 'Runtime' mode. These subjects are discussed earlier in this manual.

7.1 Creating A Text Label on A Display

Follow the follow the numbered instructions on the image below.



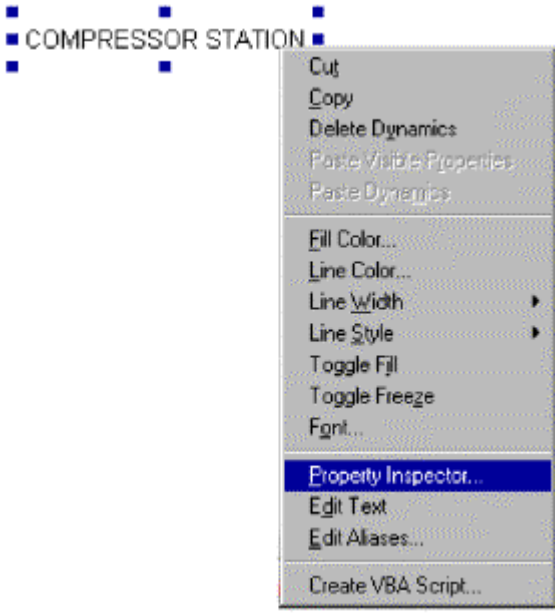
1. *Either* click the 'Text' icon, *-or-* click on **Draw→Text**.

2. Click on the location in the display where you want to add the text label, and start typing.

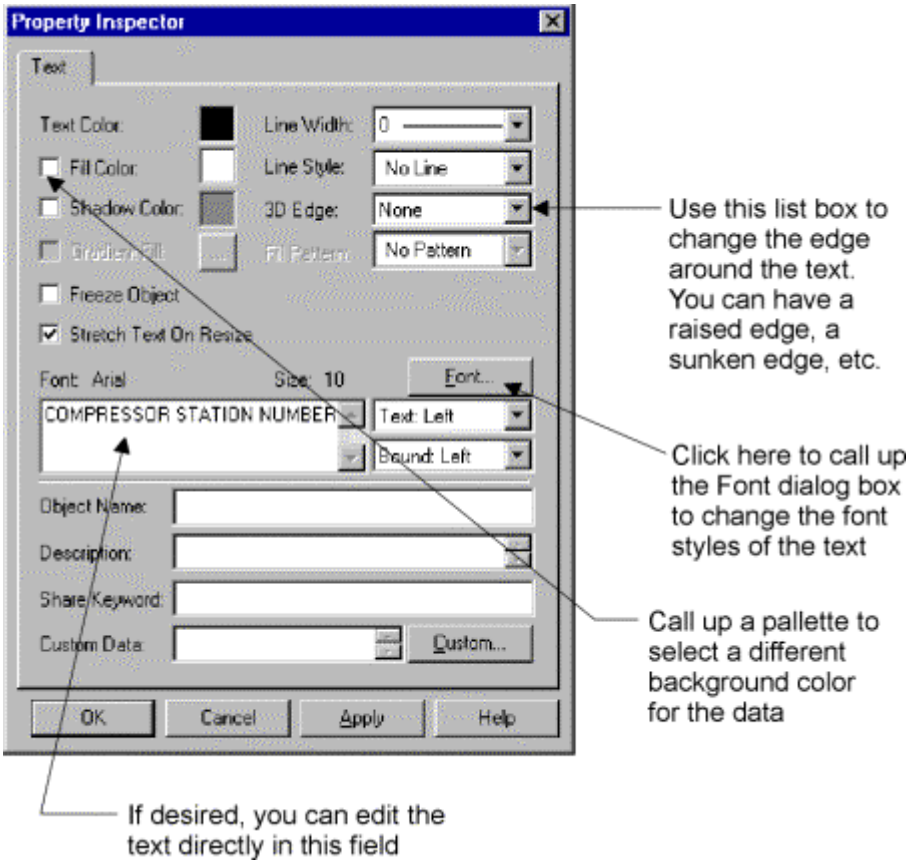
3. To change the font, size, or style of the label, select it, and use the controls at the top of the screen.

7.2 Making Changes To An Existing Text Label

If you want to change the text label you've just created, you must right click on it, and choose "Property Inspector" from the pop-up menu.



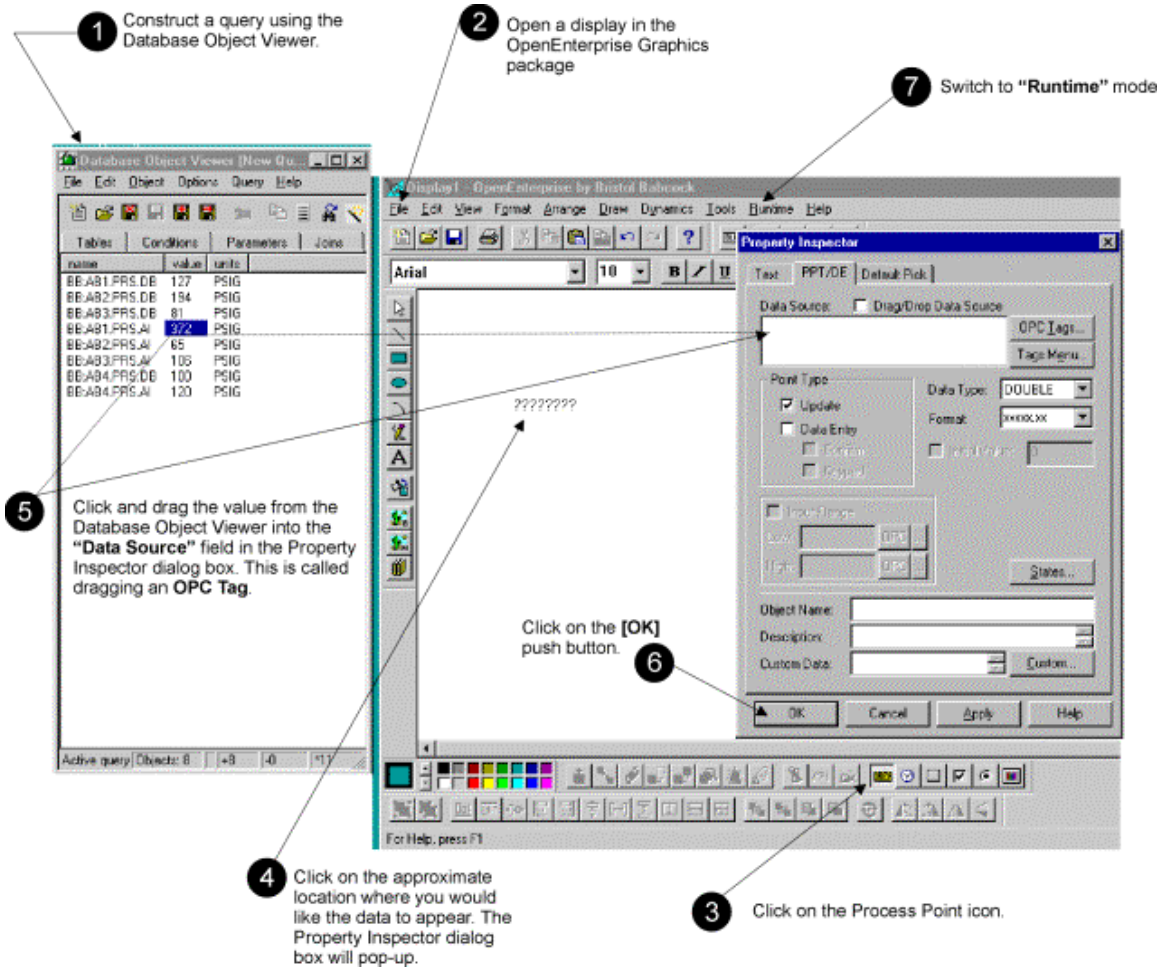
(You could also have chosen **Edit →Text** to edit the text directly on the screen, but the Property Inspector provides more options.)
You should experiment with the Property Inspector dialog box to see the various options for representing text.



7.3 Example #2 - Displaying A Numerical Value From the Database (Process Point)

7.3.1 Displaying A Numerical Value

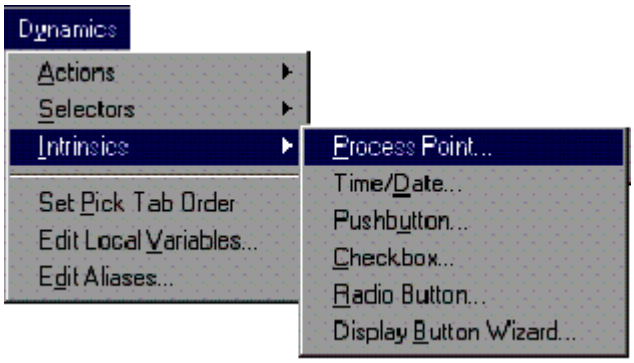
Follow the numbered instructions detailed below.



1. Using the Database Object Viewer, construct an active query of the database which calls up the signal data you are interested in from the 'realanalog' table. Leave the window open. (If you're not sure how to do this, please read pages 8 through 10).
2. Step 2. Open a display in the OpenEnterprise Graphics package, and move it side-by-side with the Database Object Viewer window, so that both windows are in view.

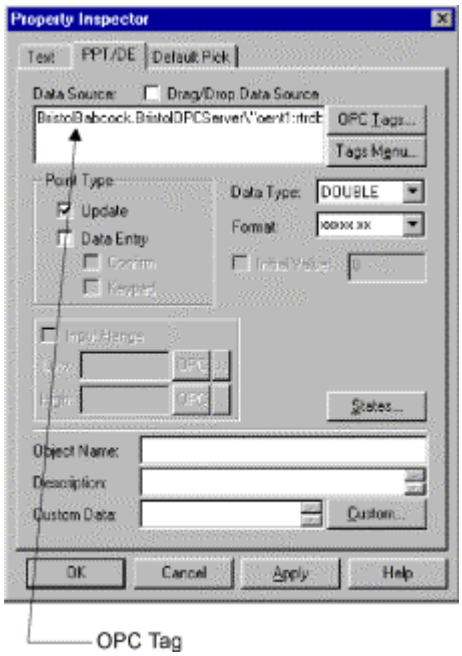
3. *Either* click on the process point icon, 

-or- click on **Dynamics → Intrinsics → Process Point**.

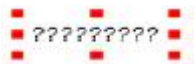


4. Click on the approximate location in the display where you want the data to appear (you can adjust the location later, if necessary.) The Property Inspector dialog box will pop-up.

5. Drag whichever value you want to display from the Database Object Viewer window into the **"Data Source"** field in the Property Inspector dialog box. This is called dragging an **OPC Tag**.



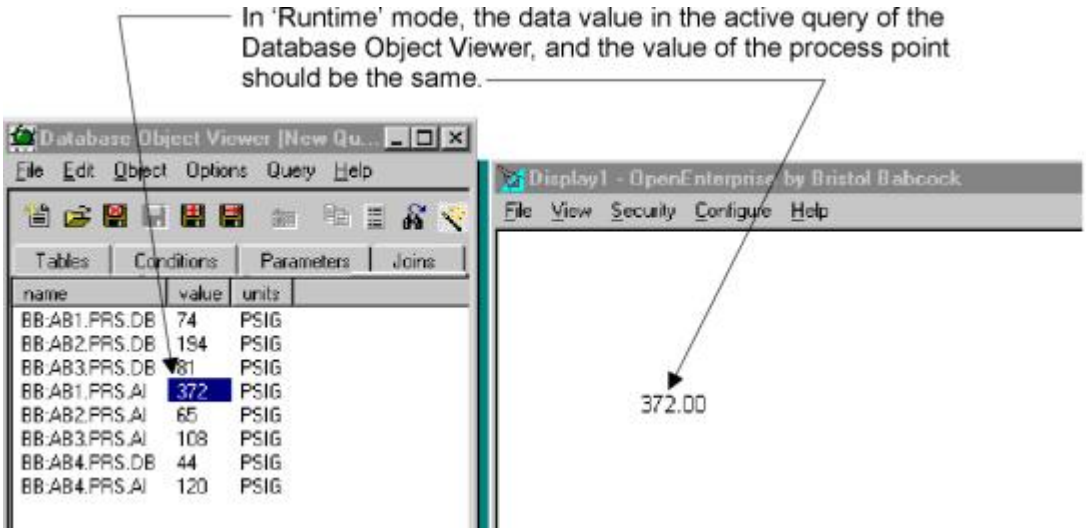
6. Click on the **[OK]** push button to save the process point. The process point will appear as a series of question marks '?????' surrounded by drag handles.



The drag handles can be used to manually re-size the text. The color red indicates that this object in the display contains 'live' data.

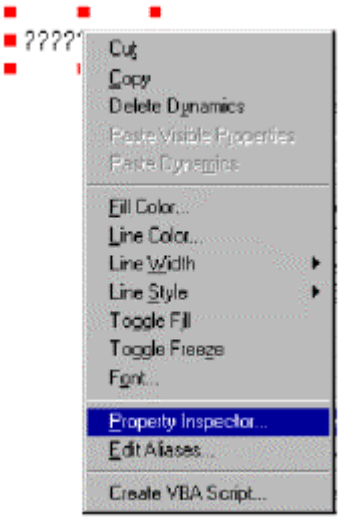
NOTE: The OPC tag is now visible within the display if you move the cursor over the process point.

7. Switch to 'Runtime' mode to view the active data on the display, by clicking on **"Runtime"** in the menu bar. The live data value should appear on the display, and should match that in the active query of the Database Object Viewer.



7.3.2 Variations on Example #2

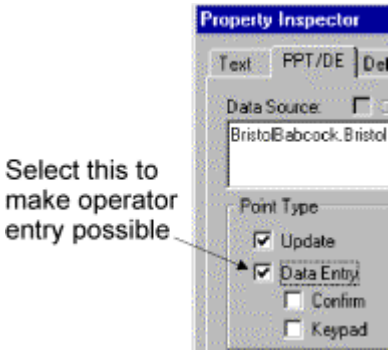
- **Making changes to an existing Process Point:** If you want to change the process point, you just created, you can recall the Property Inspector dialog box to the screen and make changes. To do this, right click on the process point in 'Configure' mode, and select **"Property Inspector"** from the pop-up menu.



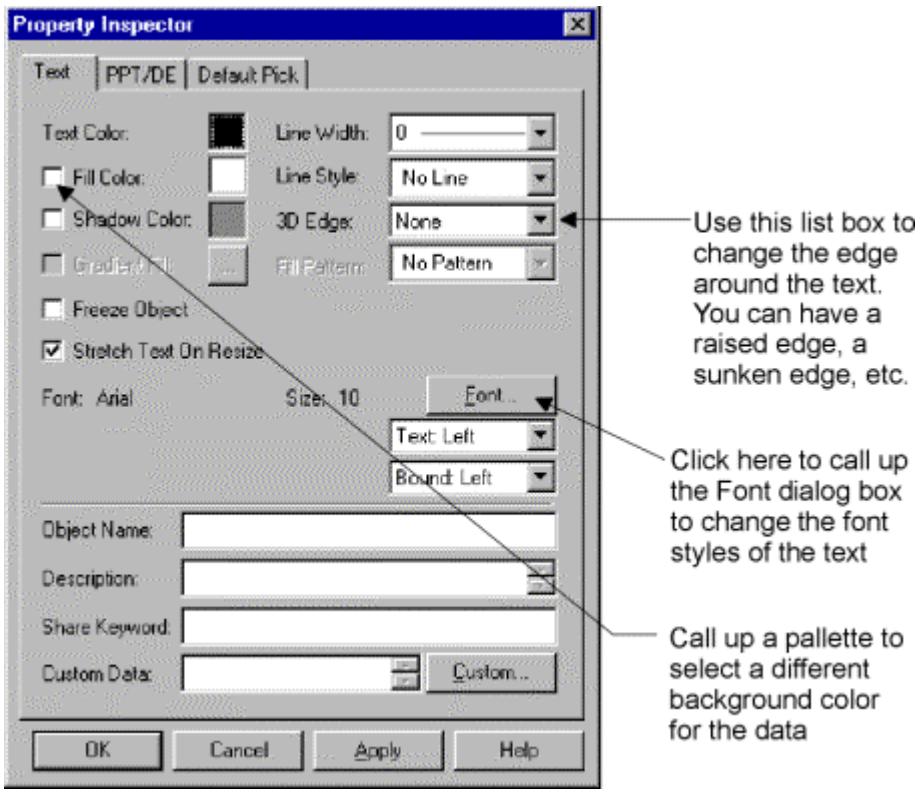
- **Displaying non-numerical data in the Process Point:**

  There is no restriction on the type of attribute you drag into the Property Inspector dialog box. While the example described using a numerical value using the 'Value' attribute, you could have dragged the 'Units' or 'Name' attribute into the **"Data Source"** field.

- **Making the Process Point changeable by the Operator:** If you want the process point to be changeable by the operator, for example, to enter setpoint levels, simply select **"Data Entry"** as part of the **"Point Type"** in the PPT/DE page of the Property Inspector dialog box. In 'Runtime' mode, the operator can enter new values into this process point which are copied to the database, and down into the RTU.



- **Changing Background & Foreground Colors of the Process Point, Changing the Font:**



The 'Text' page of the Property Inspector dialog box includes various fields for changing the style of how the text appears. You should experiment with this to see a style you prefer.

7.3.3 Common Mistakes Which Occur When Creating Process Points

- **Dragging data without *first* clicking the 'Process Point' icon, and clicking in the display**

  If you do this, you will be dragging just the text for the OPC tag onto your display, which is useless by itself. Click on the tag in the display, and press the **[Delete]** key, then start again.

- **Dragging the wrong OPC tag**

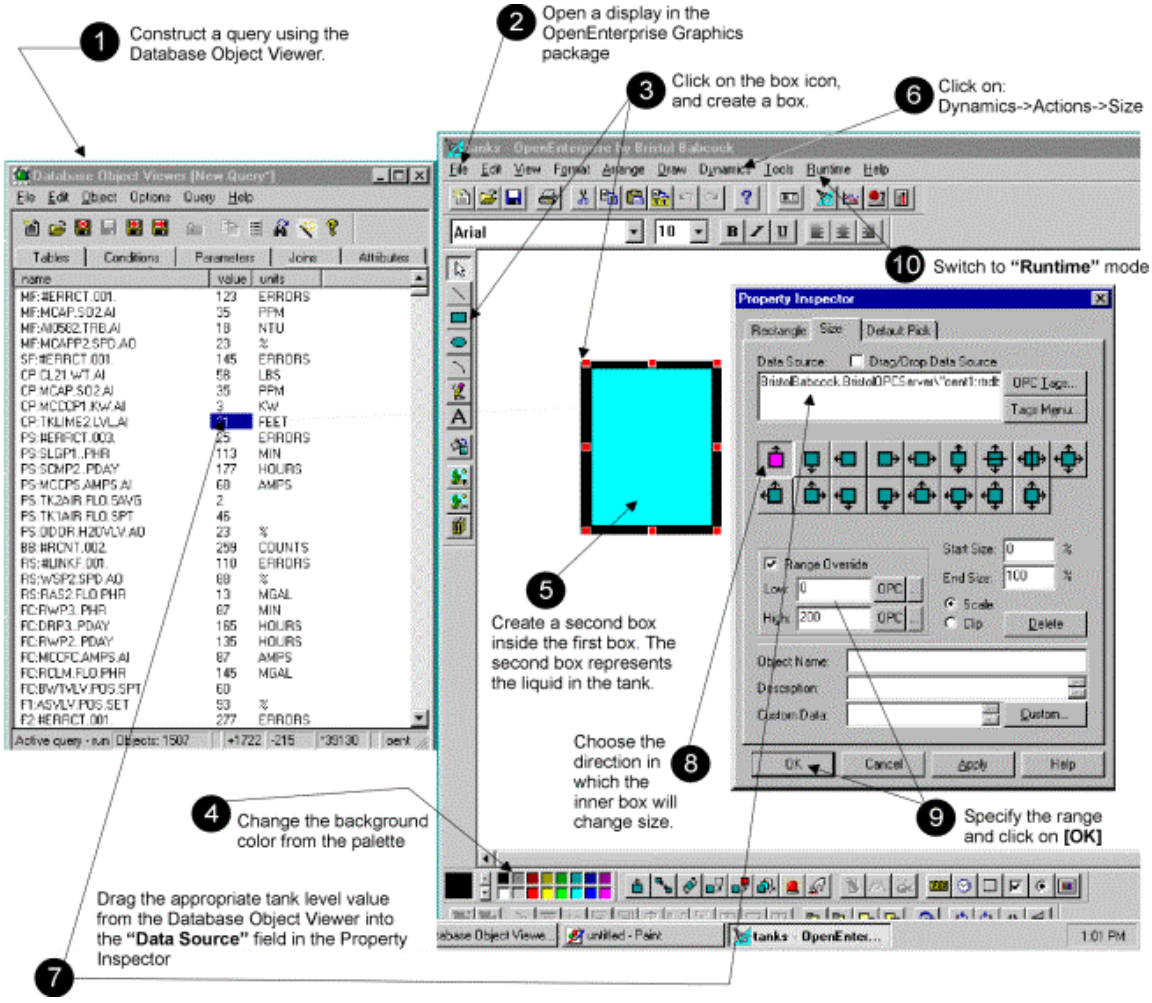
This is the most common error when creating a process point. Depending upon which attribute (column) you drag from, you might see data in the wrong format (e.g. text instead of numerical data) or you might be dragging from a row above or below the one you want. If the displayed value does NOT match the value displayed in the active query, you can just delete the current tag in the **"Data Source"** field, and drag in the correct one. Alternatively, just click on the value in the display, press the **[Delete]** key, and start again.

- **Forgetting to use an active query**

If you forgot to click on the 'Active Query' icon (shown above) before running your query, the query will only be updated if you manually run the query again, and consequently, the data on your display WILL NOT change when you go into 'Runtime' mode. Click on the 'Active Query' icon and re-run the query. If this does NOT fix the problem, verify that your connection to the database is functioning.

7.4 Example #3 - Displaying the changing level of liquid in a tank

Follow the numbered instructions below.

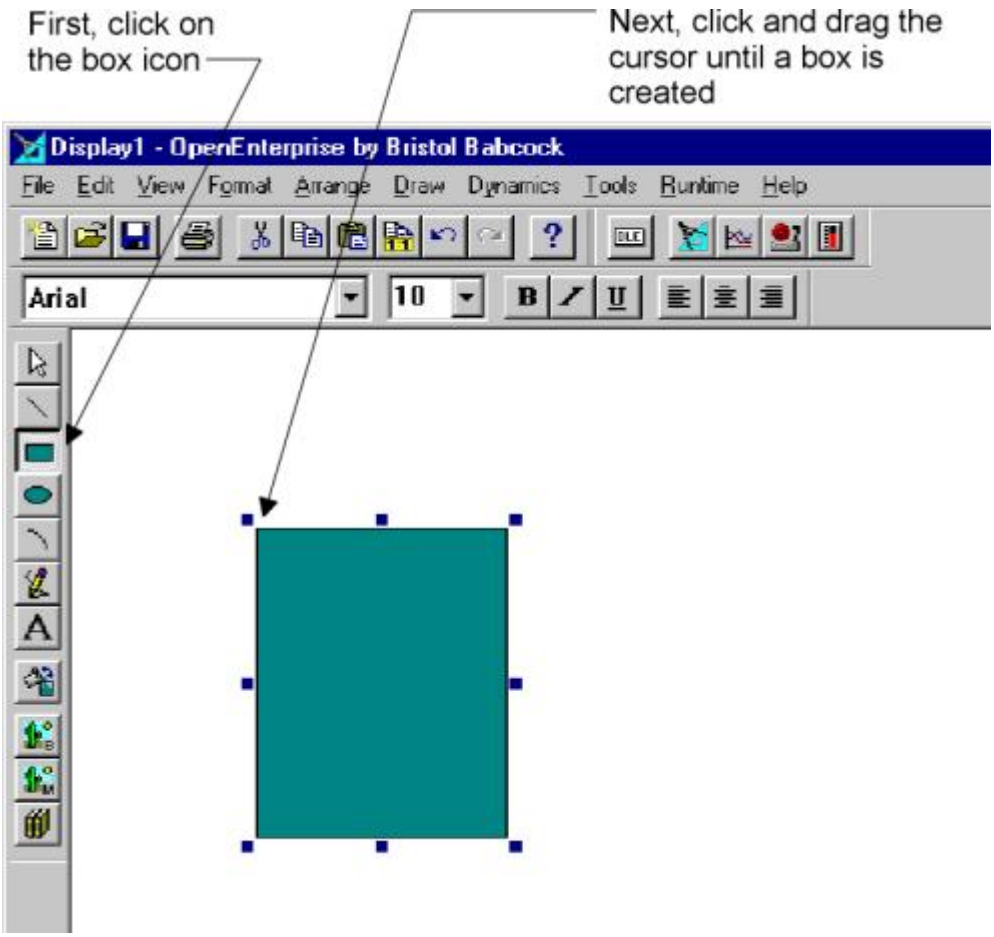


NOTE: The tank we are drawing in this example is a simple box; you can be much more elaborate in your tank design if you desire. In addition, you may wish to check the symbol library to see if there is a tank symbol already made that you want to use.

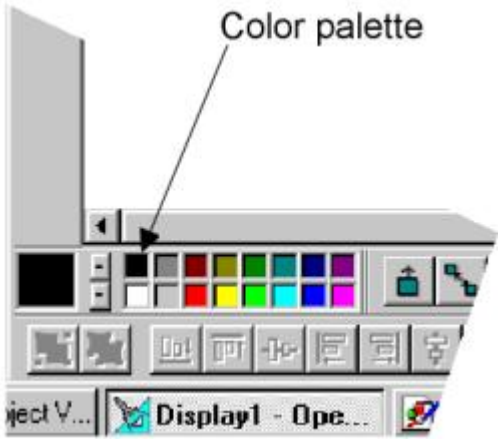
1. Using the Database Object Viewer, construct an active query of the database which calls up the tank level signal data you are interested in from the 'realanalog' table. Leave the window open.

2. Open a display in the OpenEnterprise Graphics package, and move it side-by-side with the Database Object Viewer window, so that both windows are in view.

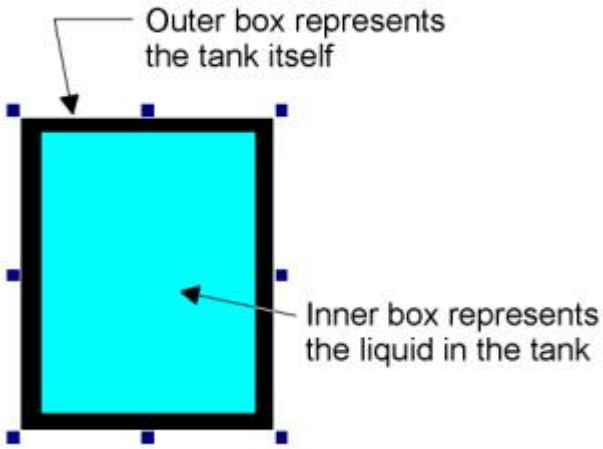
3. Click on the box icon, then move the cursor to the location where you want to add the tank. Click and drag until you have created a box which represents the tank.



4. Change the background color of the tank to black by clicking on black in the color palette while the tank is selected. Alternatively, you can right click on the tank and change both the **"Fill Color"** and the **"Line Color"** to black, separately.

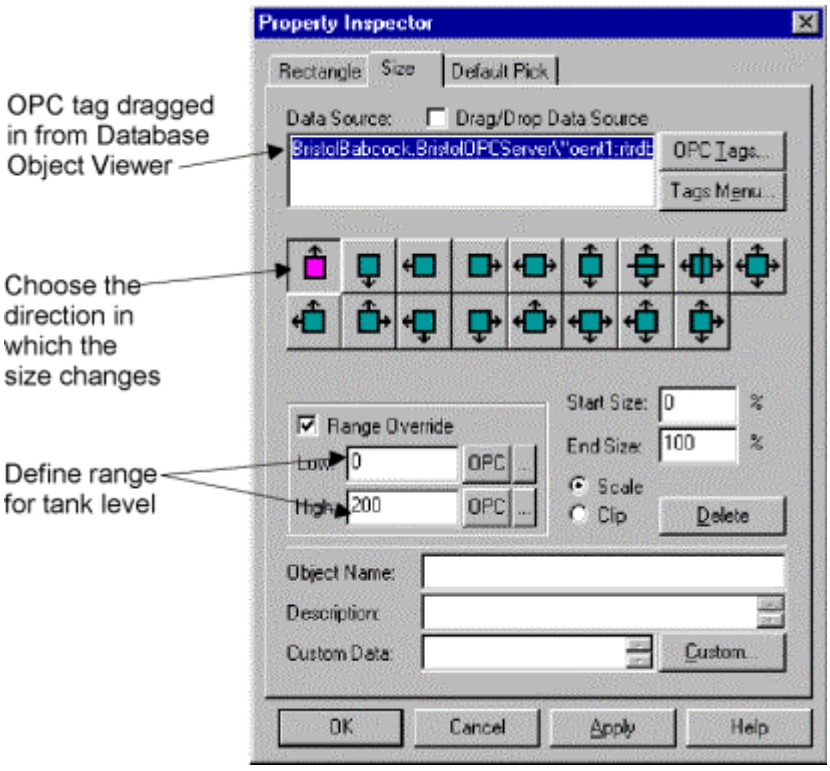


5. Using the same method described in steps 3 and 4, create another slightly smaller box with a blue color which will represent the liquid in the tank, and drag it just inside the first box, so all of its edges appear inside the first box.



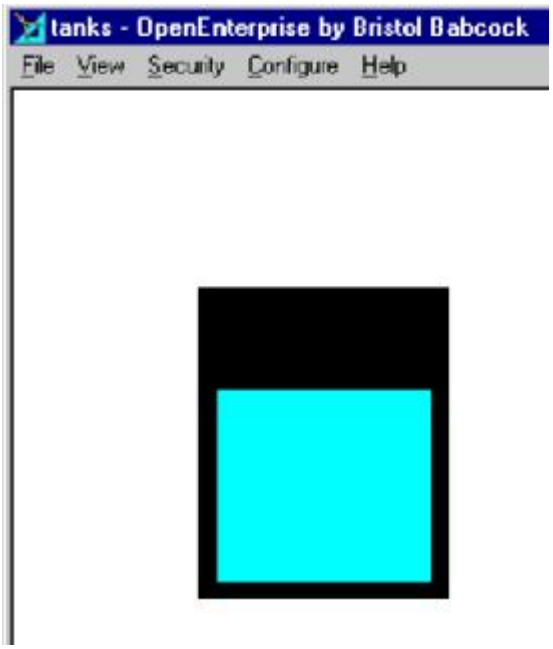
6. With the smaller blue box selected, click on **Dynamics → Actions → Size** and the Property Inspector dialog box will appear.

7. Drag the appropriate tank level value from the Database Object Viewer into the **"Data Source"** field of the Property Inspector dialog box.



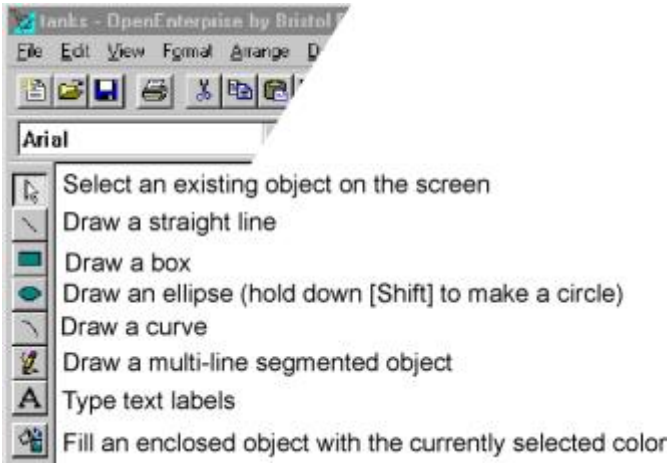
8. Choose the direction in which the inner box changes size to depict the liquid level.
9. Click on the **"Range Override"** check box, and enter values for **"Low"** and **"High"** which correspond to the minimum and maximum tank levels to be displayed. Then click on the **[OK]** push button.

10. Switch to 'Runtime' mode to view the active data on the display, by clicking on **"Runtime"** in the menu bar. The inner box representing the liquid should change size as the value changes in the active query of the Database Object Viewer.



7.4.1 Variations on Example #3

Making more realistic looking objects using the drawing tools:



The tank we just drew consisted of two boxes - - one for the tank, and one for the liquid in the tank If you want to make a more realistic looking tank for this example, you can use the drawing tools provided in the toolbar. In fact, it is a good idea to experiment with the drawing tools to see what sorts of objects you can create.

The straight line and curve tools operate similarly to the box tool; just click on their icons in the toolbar, then click within the display and drag the cursor until the tool draws the line or curve as you want it. The ellipse tool also operates similarly, however, if you want to create a circle, hold down the **[Shift]** key while using the ellipse tool.

One thing to remember when using the multi-line segmented tool is that to exit that tool, you must double-click the mouse on the endpoint where you want to stop drawing lines; otherwise it will continue to try to draw more line segments.

For more information on using these tools, see the *GraphWorX32 User Guide*.

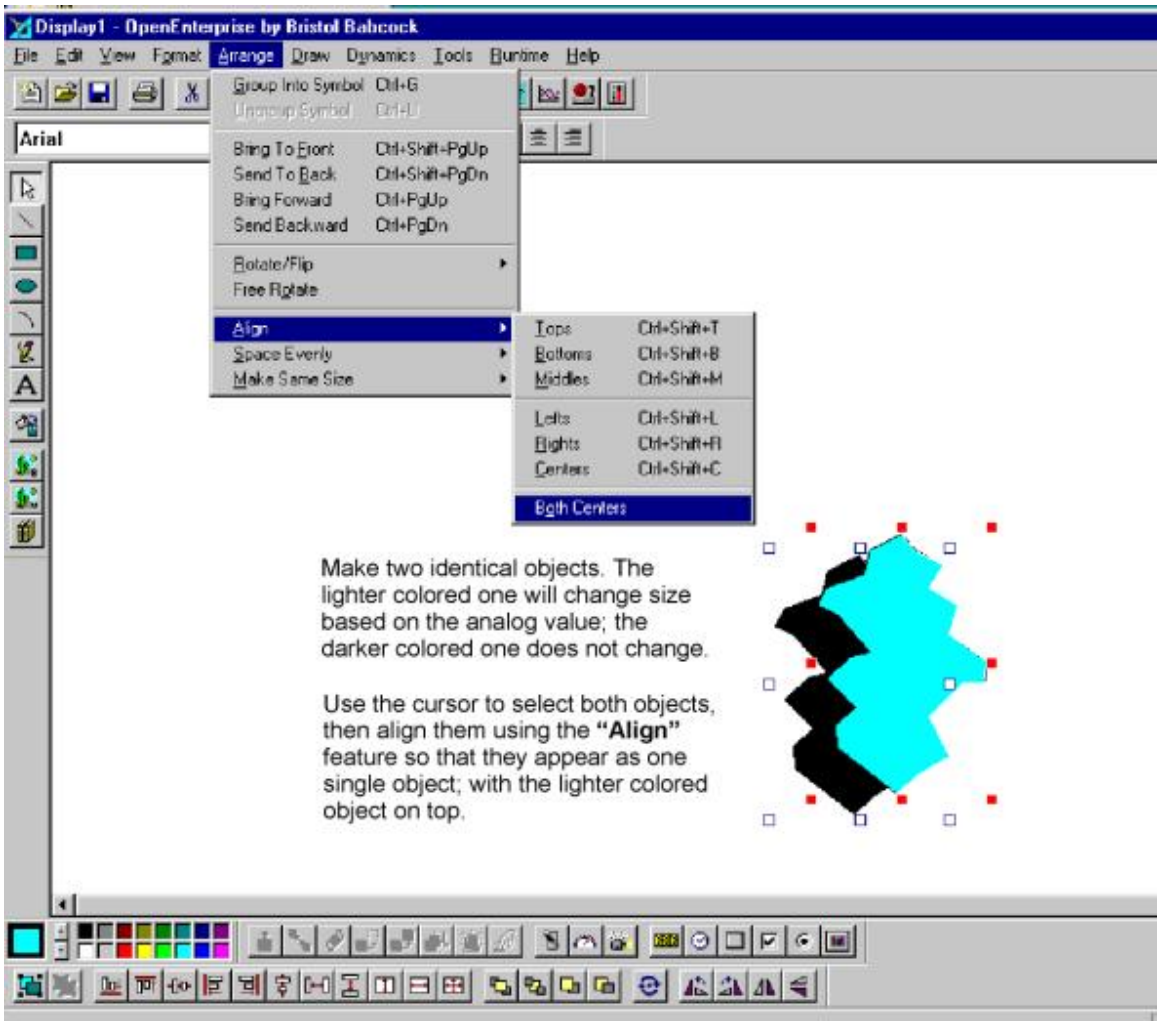
Making changes and corrections to objects:

To make changes to the non-dynamic parts of the object, just click on them in 'Configure' mode, and make changes using the drawing tools, etc. To make changes in the dynamic properties (i.e. sizing based on an analog value) right-click on the object in 'Configure' mode, and make any necessary changes in the Property Inspector. If the dynamic properties are too difficult to change in this way, you can select the object by clicking on it until you see red handles, then, from the menu bar, click on **Edit→Delete Selected Dynamics**. You can also do this by right-clicking on the object, and choosing **"Delete Dynamics"** from the pop-up menu. Now re-create the dynamics as you want them.

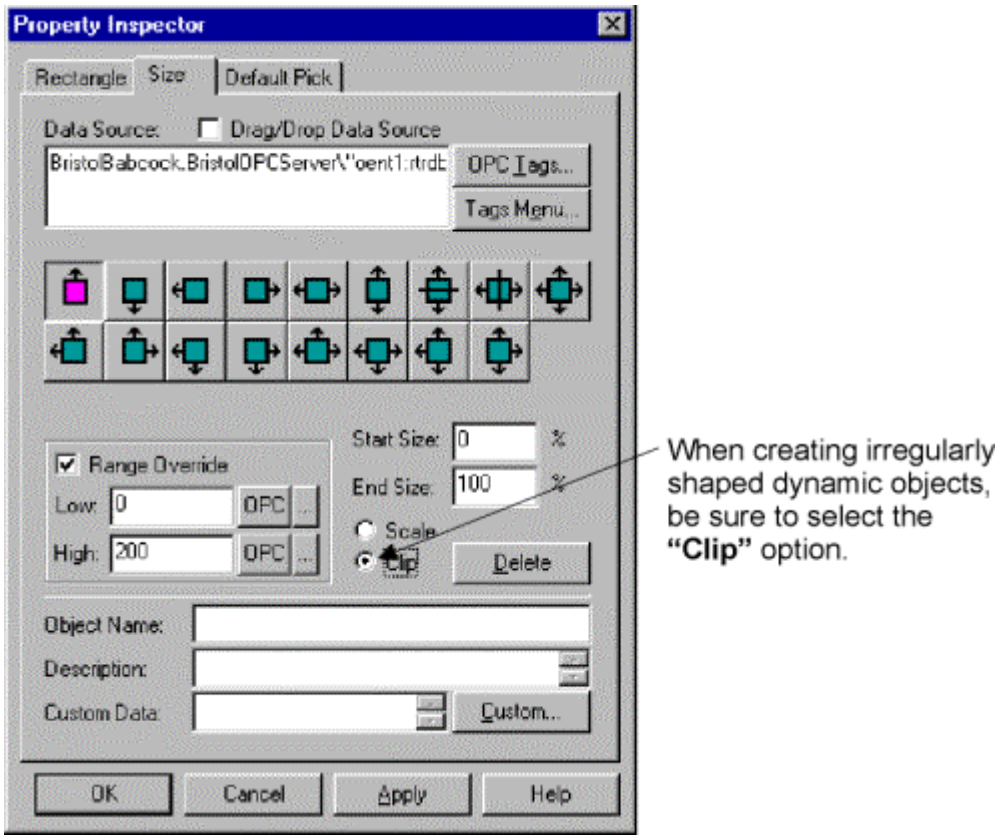
Using Irregularly Shaped Borders:

You can use the drawing tools to create an irregularly shaped object for your tank (for example, a cut-away view) which will be re-sized based on a real-time value in the database.

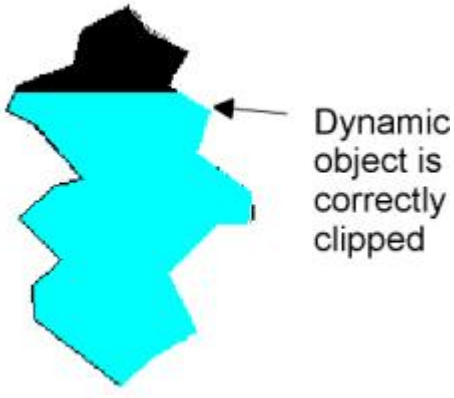
To do this, create the irregular (but enclosed) object with the segmented line tool, then duplicate it by depressing the **[Ctrl]+[D]** keys. Now you have two objects; change one object's fill color to be lightly colored (to show the liquid level) and the other object's fill color to be darker colored (to represent the background). Align the objects by dragging the cursor around them both to select them, then click on **Arrange→Align→Both Centers**. If necessary, use the **"Bring to Front"** and **"Send to Back"** options to make sure the lighter colored object is on top.

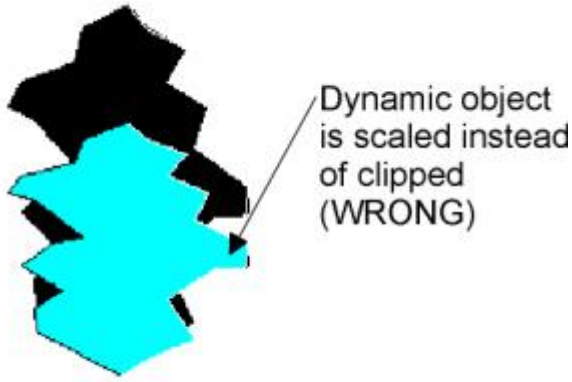


Now, click on **Dynamics→Actions→Size** and the Property Inspector dialog box will appear. Use the exact same methods you used before when you were making the tank using boxes. The only difference is, this time, make sure you select the **"Clip"** option as shown in the figure, below:



If you forget to use clipping, the entire dynamic object will be scaled, and the result will look wrong for this type of object. The figure on the left, below, shows the results of scaling an irregular dynamic object (wrong method), and the figure on the right shows the results of clipping the irregular dynamic object (correct method).



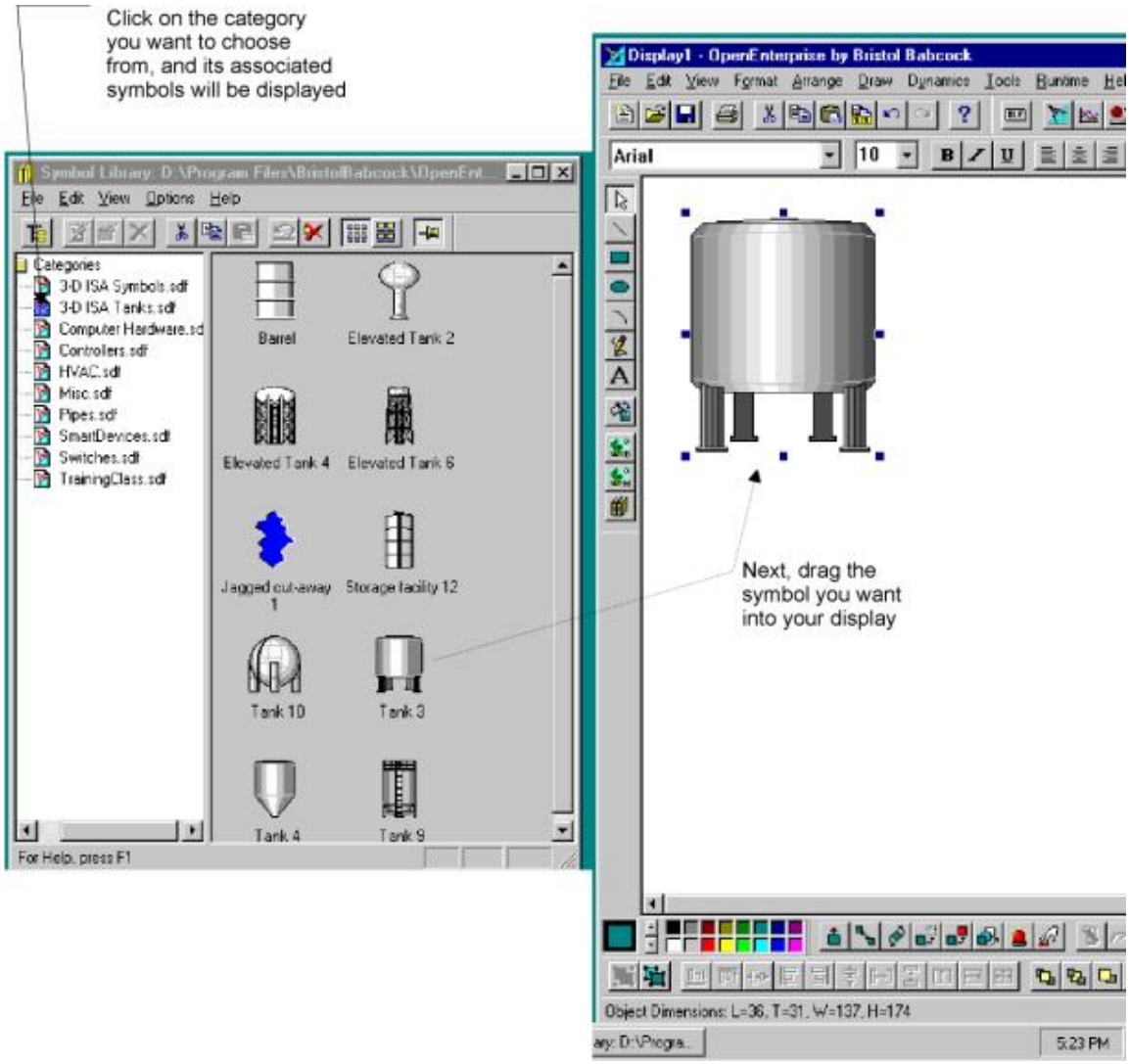


Using A Pre-Drawn Object From the Symbol Library

If you are having trouble drawing a satisfactory tank, you can always look in the symbol library to see if there is one you would prefer to use.

Click on the 'Symbol Library' icon, shown below, *-or-* click on **Draw→ Import → Symbol.**

The Symbol Library will be opened.



Click on the category of symbols you want to look at, and the available symbols in that category will appear. When you see a symbol you want to use, *drag* it from the symbol library into your display.

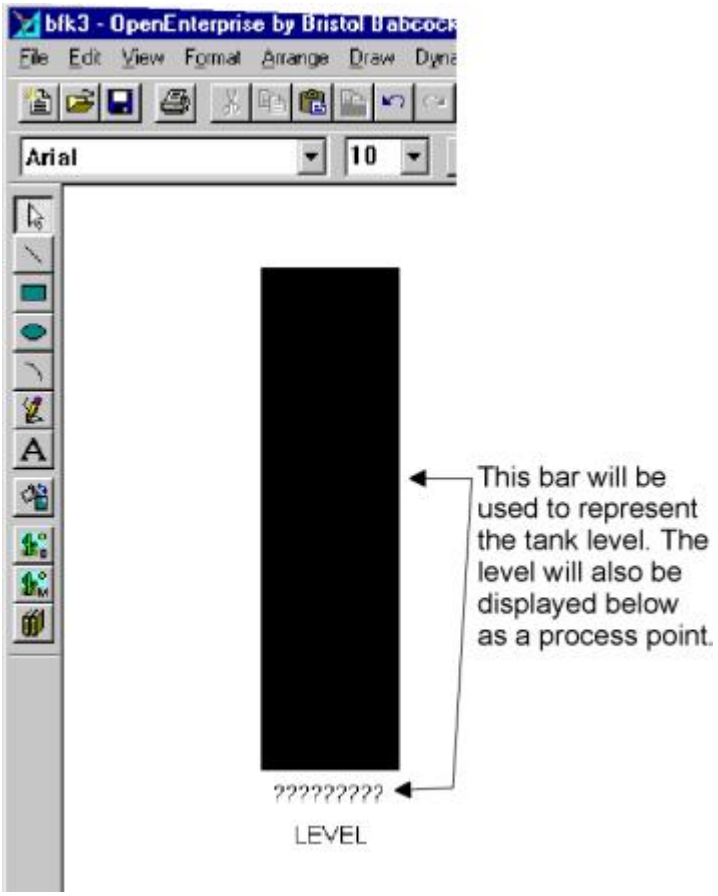
7.5 Example #4 - Creating A Sliding Setpoint Control With Which the Operator Can Update a Value in the Database & RTU

NOTE: Beginning with this example, we assume that you are familiar enough with the general display building process to know that you must run a query with the Database Object Viewer, leave it active while you call up your display, drag it into the data source field of the Property Inspection, etc. These items will no longer be listed as steps in the example, because their inclusion is implied. In addition, by this example, we assume that you are familiar with use of the drawing tools, and how to make use of the topics in the previous examples such as creating a process point, making an object change size based on an analog value, etc.

Suppose your controller includes ACCOL logic which allows an operator to adjust a setpoint for a tank level; the operator alters the setpoint causing other logic controlling input and output valves to the tank to open or close in order for the tank level to reach the setpoint. There are many ways to incorporate this operator setpoint within a display.

You could, for example, create a data entry process point in which the operator manually enters a new setpoint.

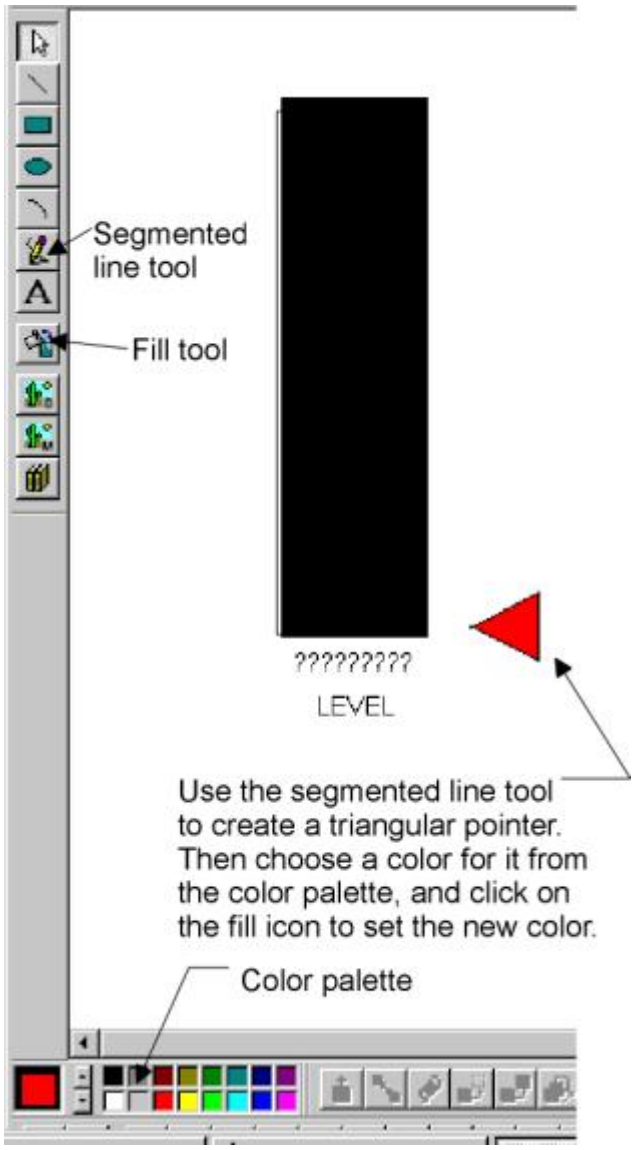
Another approach, which is the subject of this example, is to create a sliding setpoint control, which the operator drags on the screen to dynamically change the setpoint; as the value changes, it is updated in the database, and copied down to the RTU.



1. Start out by creating a tank liquid level graphic which re-sizes based on the tank level value. In this case, just show the level as a dark colored bar, on top of an identically sized lighter colored object. (The light colored bar underneath does not change - - only make the dark colored object dynamic.) It would also be a good idea to display the tank level as a process point just below the graphical representation on the display.

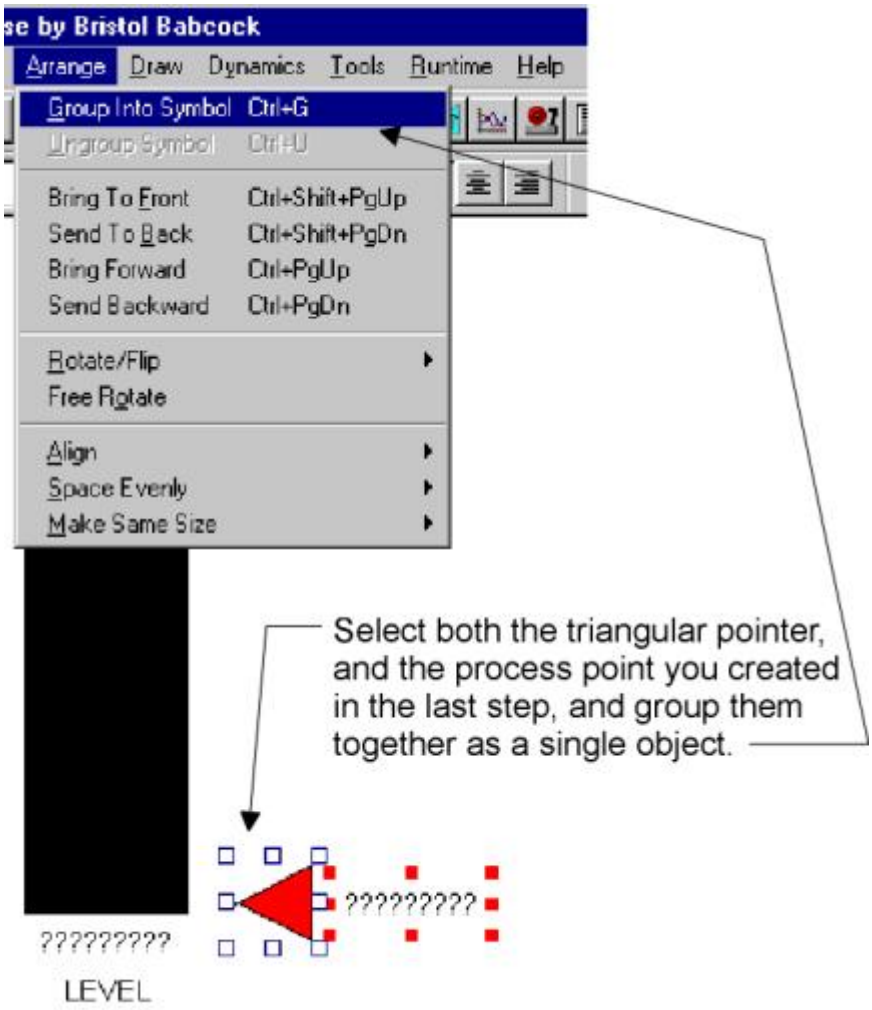
   If you're unsure how to create this graphic, please review Example #3 beginning on page 26. If you're unsure how to create a process point, please review Example #2 beginning on page 21.

2. Using the segmented line tool, draw a triangular pointer, which will be used as the setpoint slider. (You can draw the pointer on another part of the display, and drag it over later; it doesn't have to be right next to the bar representing the tank level yet; eventually however, the tip of the pointer should be next to the bar representing the tank level.) Double-click when you have finished enclosing the pointer (to exit the segmented line tool) then choose a color for the pointer from the color palette, and click on the fill tool to set the new fill color.

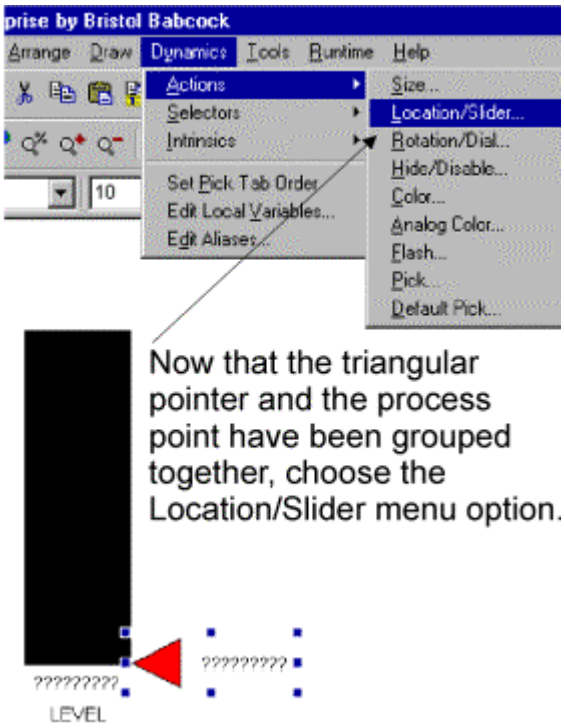


3. Create a process point next to the triangular pointer (created in Step 2). This process point should use the value of whichever signal is your setpoint signal. Do NOT make it a data entry point; just make it a normal process point which displays the value of the setpoint signal.

4. Select *both* the triangular pointer *and* the process point next to it, then group them together as a single object by clicking on **Arrange→Group into Symbol**.

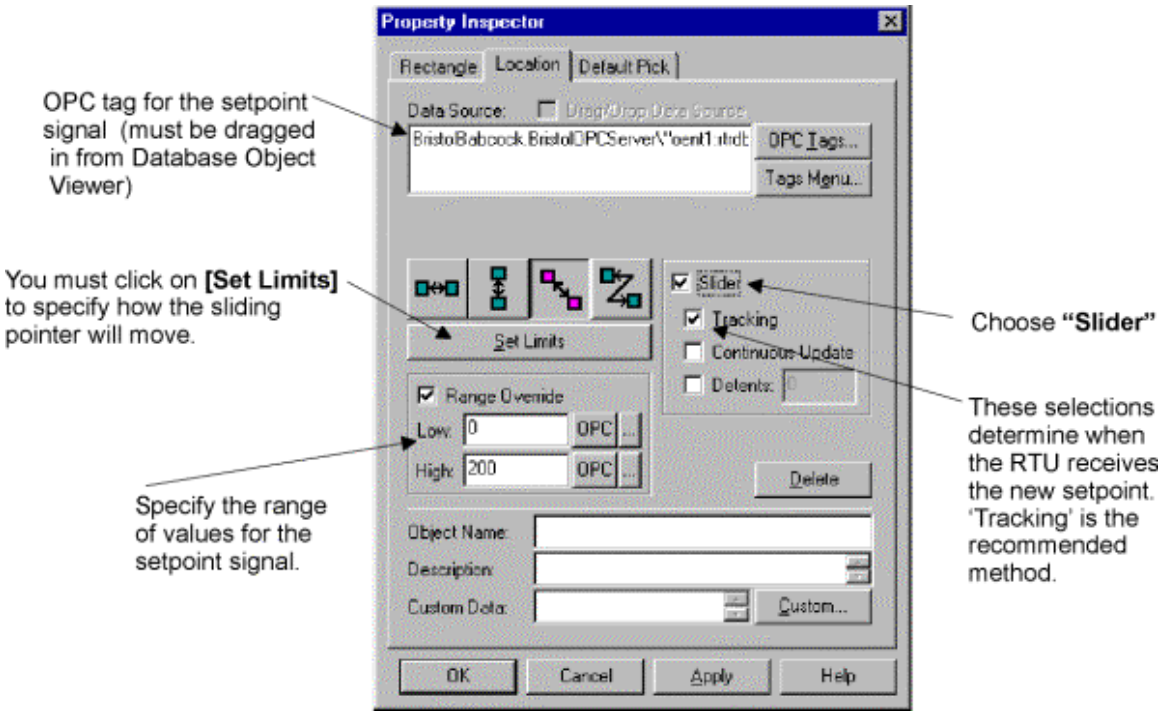


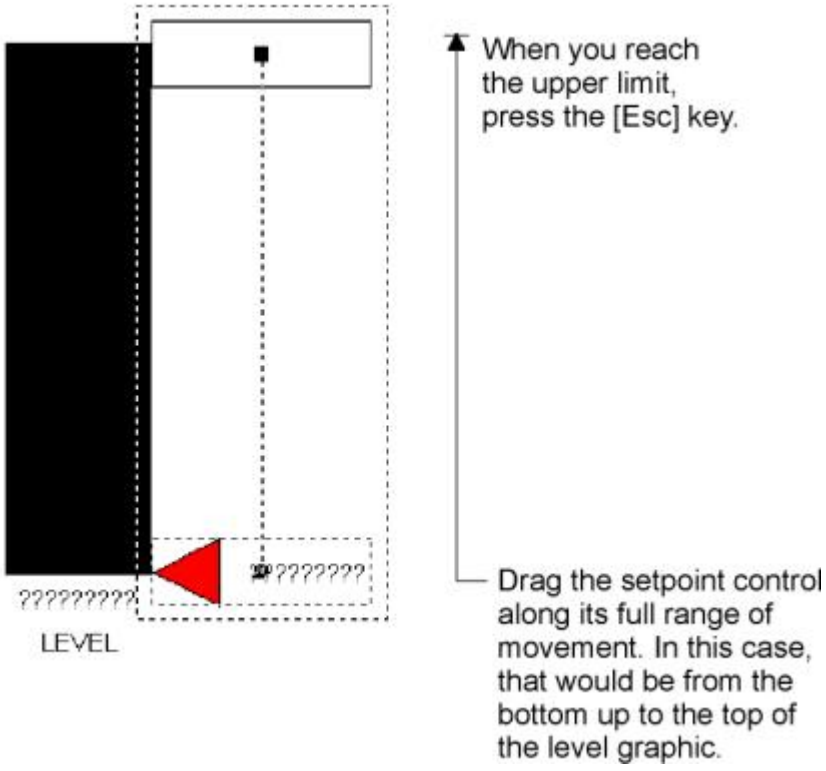
5. Now that the process point and triangular pointer are grouped together as a single object, you should move them next to the bottom edge of the tank level graphic. Now we can add dynamic capability to them. To do this, click on **Dynamics →Actions→ Location / Slider**. The Property Inspector dialog box will appear.



6. Complete entries in the Property Inspector dialog box as follows:

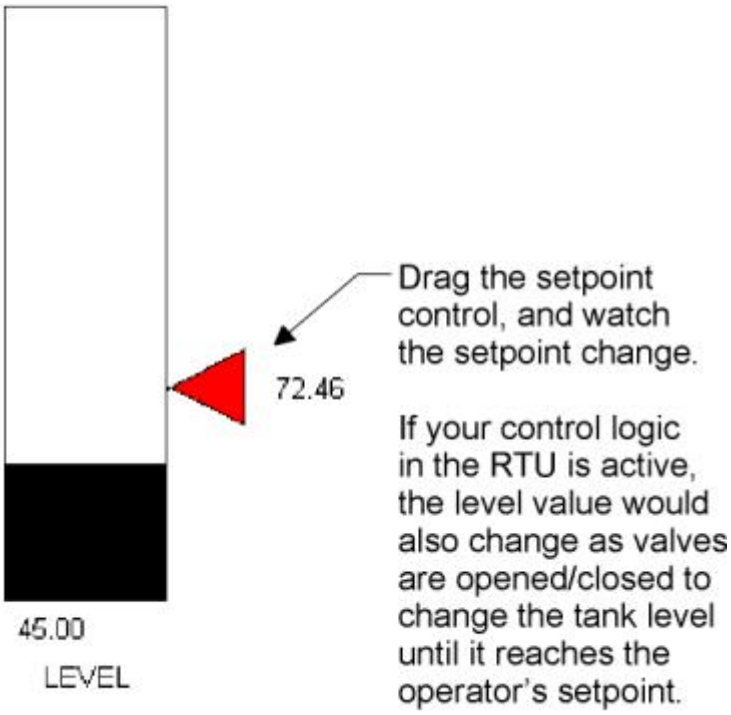
   - Drag the value representing the setpoint signal into the **"Data Source"** field. (This is the same signal value you used for the process point in step 3.)
   - Choose the **"Slider"** option.
   - Determine how you want data to be updated at the RTU by the setpoint control. We recommend you choose **"Tracking"** because this will cause the change to be written only when the operator releases the mouse at the desired setpoint. If you choose **"Continuous"** the data will continually be written to the RTU as you move the mouse; this can impose a burden on your system communications if you are using radios, for example, because OpenEnterprise will initiate a new communication transaction for each incremental move of the mouse. If you choose **"Detent"** you can specify an incremental value after which data will be written to the RTU, mouse changes which are less than that increment will not be sent to the RTU.
   - Enter the lowest and highest possible values for the setpoint signal.
   - Click on the **[Set Limits]** push button, and define the limits for the sliding setpoint control (See step 7).





7. Drag the setpoint control from the bottom of the tank level graphic to the top (this represents the full range of movement for the control). Then press the **[Esc]** key and you will return to the Property Inspector dialog box. Click on **[OK]**.

8. Go to 'Runtime' mode, and test the setpoint control. Drag it with the mouse and watch the value of the setpoint process point change. If your RTU logic was active, and controlling a process, this would cause valves to open/close etc. in order to raise or lower the liquid level in the tank (as displayed by the tank level graphic) in order to meet the new operator setpoint.

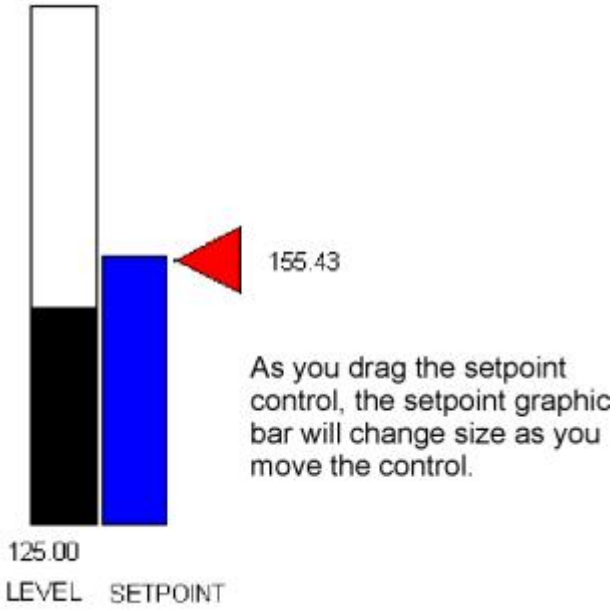


Variations on Example #4

7.5.1 Variations on Example #4

Making the setpoint visible as a second graphic next to the tank level graphic:

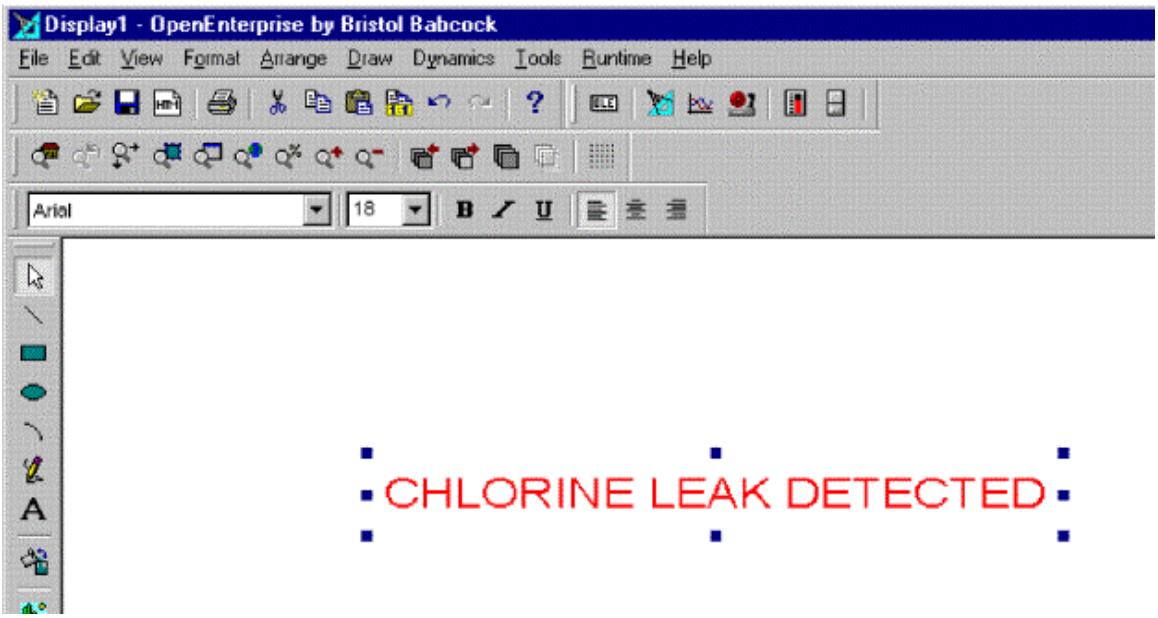
In the example we just completed, the setpoint is manipulated by moving the setpoint control next to the level, but the change in the level is not immediately reflected, since it depends on changes in the RTU and the process. You could create a second bar graphic representing the setpoint value next to the level graphic. When you move the setpoint control in runtime mode, the setpoint bar should move along with it.



7.6 Example #5 – Displaying A Message Based on A Logical Value in the Database

1. On a display, use the text tool to enter the text label you want to present. For example, let's say that in the event of a chlorine leak detection, you want to display in large red letters "CHLORINE LEAK DETECTED". (For help on using the text tool, see *Example #1 - Creating A Text Label on A Display)*

2. Select the text by clicking on it; it will be surrounded by edit handles.



3. Now that the text is selected click as follows: **Dynamics→Actions→Hide/Disable.** The Property Inspector dialog box will appear.

4. On the 'Hide' page of the Property Inspector dialog box, drag in the OPC tag from the Database Object Viewer, and place it in the **"Data Source"** field. Choose **"Hide/Disable when False"** because when the Chlorine Leak signal is OFF, we don't want to display the 'CHLORINE LEAK DETECTED' message; we only want to display it when the Chlorine Leak signal is ON (True). Click on **[OK]**.

5. To test that the label is displayed correctly, simulate a chlorine leak by turning the associated chlorine leak signal ON via Open BSI DataView, then verify that the chlorine leak label is displayed in Runtime Mode. (Be sure to turn the signal OFF when you're finished testing.)

7.6.1 Variation on Example #5

Making the 'CHLORINE LEAK DETECTED' label flash on and off when it is ON.

In the example we just completed, we displayed the 'CHLORINE LEAK DETECTED' label when the associated signal was ON. Suppose, to grab the attention of the operator, we also want the label to FLASH on and off.

To do this click as follows:

Dynamics→Actions→Flash

Go to the 'Flash' page of the dialog box. Specify the state when flashing is to appear (in this case **"Flash When True"**) and if desired, adjust the **"Flash Rate"**.

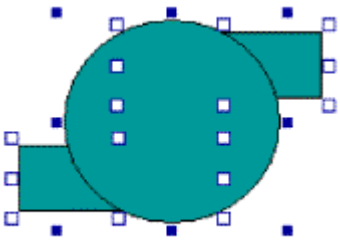
Click on **[OK]** and go into Runtime Mode to verify that the text flashes as expected.

Example #6 – Changing the Color of A Pump Based on a Logical Value

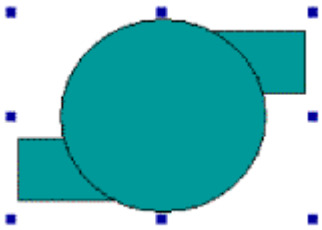
7.7 Example #6 – Changing the Color of A Pump Based on a Logical Value

1. On a display, draw a simple pump using a circle and two rectangles. (See figure, below left). Use the **Arrange→Send to Back** and **Arrange→Bring to Front** commands as necessary to put the rectangles behind the circle. (See figure, below right). It's true, it doesn't look much like a pump, but it's fine for the purposes of this example.

2. Select the pump by dragging a bounding box around it. When you release the mouse, edit handles will appear around all of the pieces.



3. Group the pump into a single symbol by using **Arrange→Group into Symbol**



4. Click on Dynamics→Actions→Color

5. Let's say that when the pump is running, you want to show it in RED. Drag the OPC tag for the pump status signal value into the upper part of the **"Data Source"** field, then click on the **[Add]** button. This will cause the tag to be copied to the lower part of the field. Once here, its value can be used to change the color of the object. Now choose 'RED' for the **"Fill Color"** in the **"Apply Change to"** part of the dialog box. Then choose **"Change Color on True"** because we want the pump to appear in RED when it was ON. (If we had wanted it to appear in RED when the associated signal was OFF, we would have chose **"Change Color on False"**). Finally, click on **[OK]**.

6. Switch to Runtime Mode and verify that the pump changes color based on its status.

7.7.1 Variations on Example #6

Showing 'RUNNING' or 'STOPPED' next to the pump, based on its status

In the example we just completed, all we do is show that the pump is running by changing the pump's color. What if your operator is color blind?

In a real world example, you probably would also want to show a label 'RUNNING' next to it, and a 'STOPPED' label next to it when it was stopped. This ensures there is no confusion about what a particular color indicates. (You could do that using the techniques described in *Example #5 – Displaying A Message Based on A Logical Value in the Database*)

Allowing the Operator to Call Up Another Display by clicking on the pump symbol

IMPORTANT: Be sure you save your current display *before* you try this.

Select the pump we have already configured to change color.

Choose **Dyamics → Actions → Pick** to call up the Property Inspector dialog box. From the 'Pick' page of the dialog box, choose **"Load Display"** from the **Actions"** list box, then use the **[Browse]** key to locate/specify the display you want to have loaded when the operator clicks on the pump. Click on **[OK]** when finished, then test it in Runtime Mode.

Allowing the Operator to START/STOP the pump by clicking on the pump symbol

If you want the operator to be able to START/STOP the pump from your display, you could create dedicated START and STOP labels. If your system uses the *same* signal for both starting and stopping a pump (like an ON/OFF switch), you could configure the pump symbol to be an ON/OFF switch.

NOTE: In this case, the pump's color is still tied to the pump status signal, but you click on the pump to START/STOP it. (The START/STOP signal is *different* from the status signal.)

To do this, select the pump we have already configured to change color, and click on **Dynamics → Actions → Pick**.

Click on the 'Pick' page of the dialog box, and choose 'Toggle Value' for the **"Action"** and drag in the start/stop value OPC tag from the Database Object Viewer. Specify 0 as the **"Toggle Value1"** state (STOP pump) and 1 as the **"Toggle Value2"** state.

In Runtime Mode, the operator can now start/stop the pump by clicking on the pump symbol.

There are variations to this example, as well. For example, instead of clicking on the pump symbol, we could have created **[Start]** and **[Stop]** push buttons for the operator to click on using **Dynamics→Instrinsics→Pushbutton**. Try that as an exercise.

8 What Now?

Well, we've really only scratched the surface. There are numerous other dynamics / actions you can configure. The best way to learn about them is to experiment. Try things out, play with the example displays, and when necessary, review the on-line help and the *GraphWorX32* manual. There are also some additional examples discussed in the *'OE Graphics'* section of the *OpenEnterprise Reference Guide* (document# D5092).

Before you do this though, there are three more important topics that need to be covered. They are important because they can save you hours of time and effort if you understand them, and use them when building the actual displays for your system. These topics are covered in the following sections:

- *Saving Your Own Symbols in the Symbol Library, and Updating Objects through Share Keywords.*
- *Using Aliasing to allow you to re-use the Same Objects With Different Data From the Database*
- *Tips For Planning Your Human-Machine Interface (HMI) System*

8.1 Saving Your Own Symbols in the Symbol Library, and Updating Objects through Share Keywords

Earlier in this manual, we described how to retrieve pre-defined symbols from the symbol library. What if you want to create your own symbols? Let's say you have defined a symbol that is good enough that you want to re-use it. For example, suppose you have created a symbol for a water tank, and you want to use it throughout your displays. You can save it in the symbol library.

Saving objects as symbols in the symbol library

Updating Shared Objects

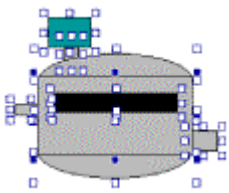
Using Aliasing to allow you to re-use the Same Objects With Different Data From the Database

Other Uses of Aliasing

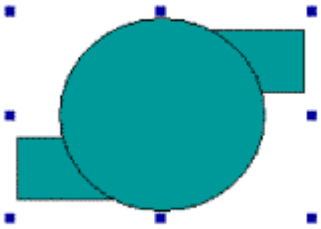
8.1.1 Saving objects as symbols in the symbol library

Let's say your display uses some simple shapes to represent a meter.

1. Select the entire object.



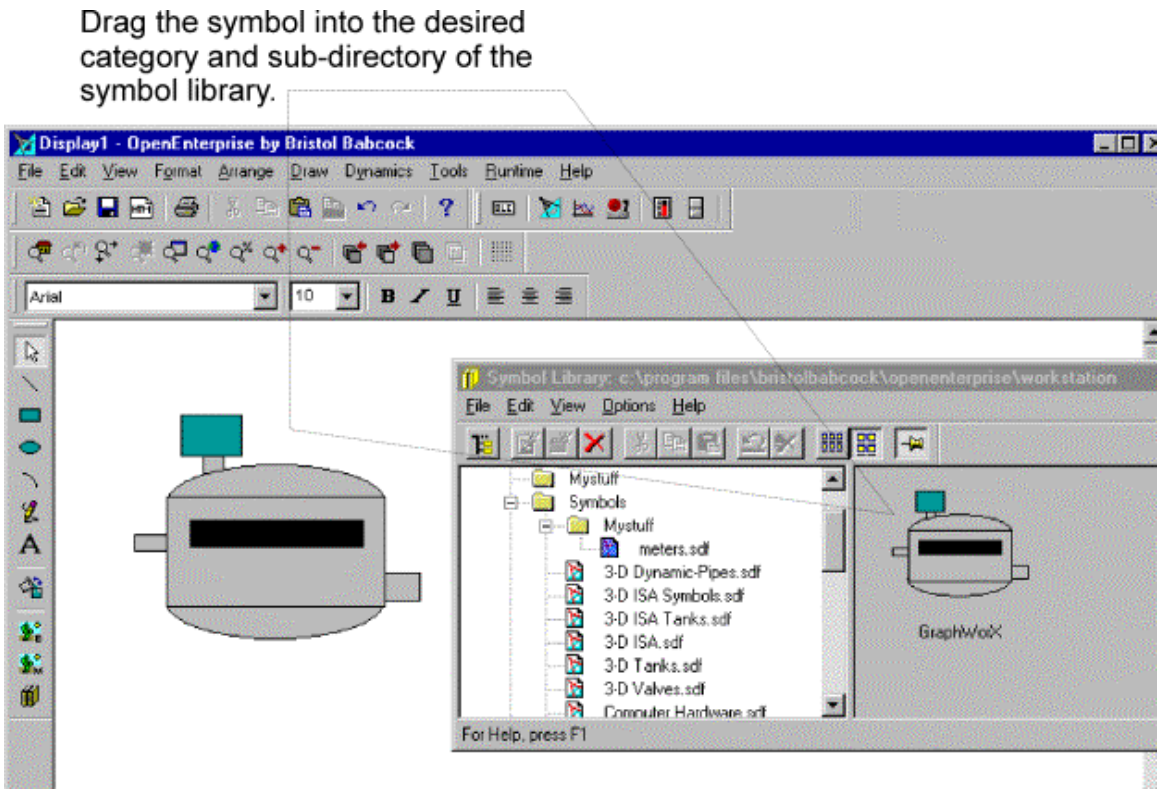
2. Group the object into a symbol by clicking on: **Arrange→Group Into Symbol**



3. *Right*-click on the object and choose **"Property Inspector"** from the pop-up menu. Assign a "**Share Keyword"** in the Property Inspector. A share keyword is just a name, but it has an important function if you make subsequent changes to the symbol.

4. The Symbol Library is divided up into a tree of folders. Each folder has one or more category pages; the category pages contain individual symbols. If desired, create a folder and category page of your own by clicking on the **"Categories"** folder, and clicking on **File→Add→Subdirectory** and then **File→Add→Category**. Alternatively, you can use existing folders and category pages.

5. Drag the symbol into the desired category page and symbol sub-directory.



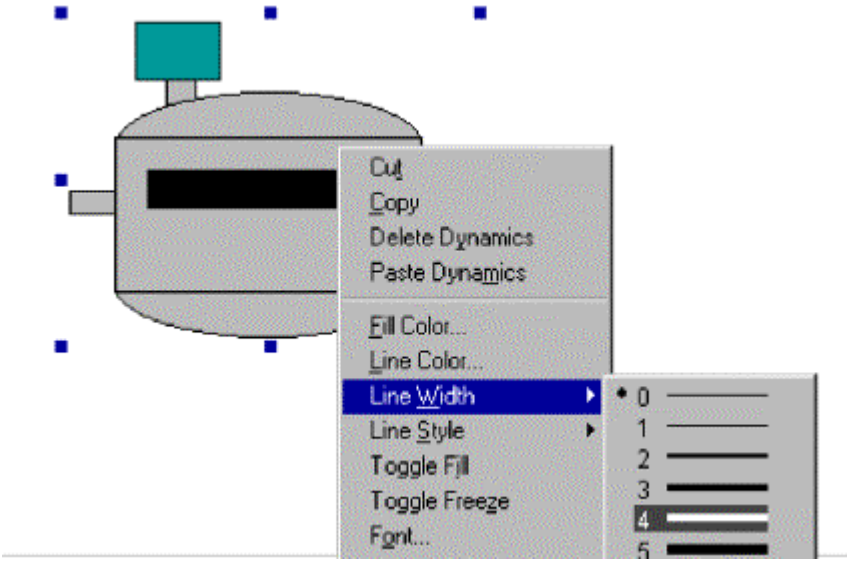
Now you can proceed to use the symbol in various displays, by dragging it out of the symbol library, when needed. (See '

Using A Pre-Drawn Object From the Symbol Library*'* which was part of Example 4.*)*

8.1.2 Updating Shared Objects

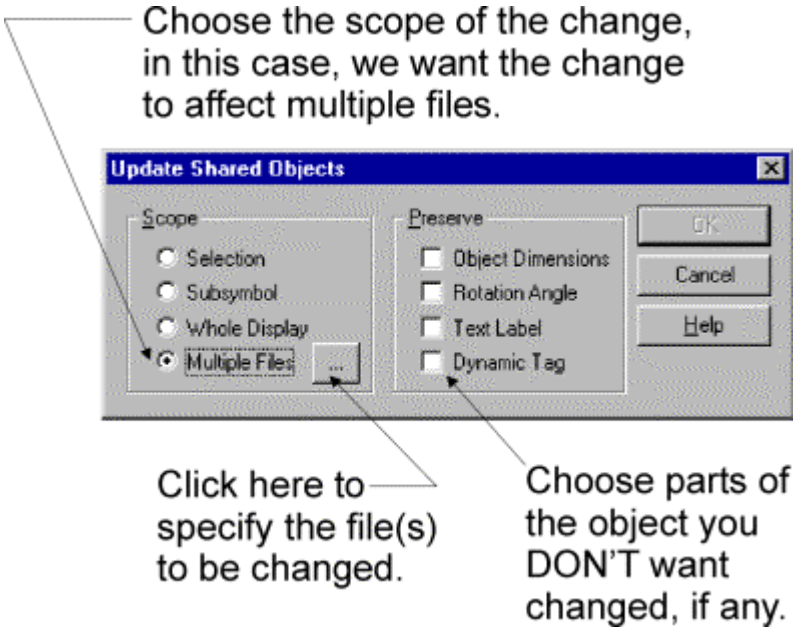
Suppose you have now used the symbol multiple times on multiple displays, but now, you decide, for whatever reason, that you want to change the symbol's line style to use thicker lines (as shown at right). If you had to manually change each individual symbol, it could result in hours of work. Because, you assigned a **share keyword**, however, you only need to make the change *once*.

To do this, *right*-click on the symbol in one of the displays, and choose the line width option from the pop-up menu. (We're choosing line width for this example, but we could have chosen other items from the menu.)



NOTE: You should never *ungroup* a symbol, or the shared object name will be lost. To edit parts of the symbol choose **"Edit Symbol"** - - in this case those parts should have had shared keywords defined.

When the change has been completed, choose **Edit→Update Shared Objects** and you will be prompted to identify the scope of the change (current display, multiple files), and if multiple files, select which files are to be updated. Also, specify if there are parts of the object to be preserved.

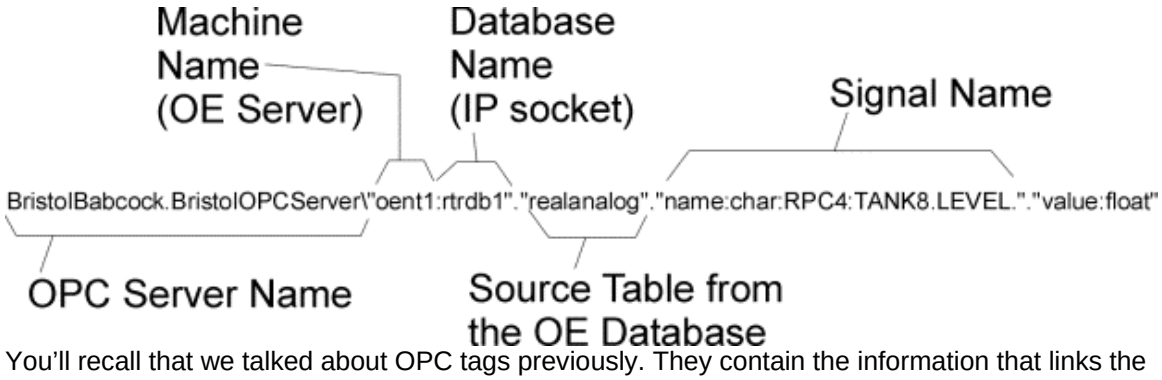


NOTE:

Once you have completed updating the shared objects in your display(s), if you can save the new version of the symbol in the symbol library by saving it as if it was a new symbol. (The old version will still remain in the symbol library, unless you delete it.)

Using Aliasing for object re-use

8.1.3 Using Aliasing to allow you to re-use the Same Objects With Different Data From the Database



You'll recall that we talked about OPC tags previously. They contain the information that links the dynamic objects in your displays with the actual data values driving them in the OpenEnterprise Database.

The OPC tags are stored in your display, to allow it to reference the correct database attributes when required.

Suppose, however, you have a system with 8 water tanks, which are identical in every way, *except* for the signal names in their OPC tags. You *could* create 8 separate displays (1 for each water tank). That would be difficult to support, however, because if you decided later that you wanted to change the layout of the displays, add items, etc., you would have to do that on 8 different displays.

An easier way to do that, is to make use of a technique called **aliasing**. Aliasing is a powerful technique for substituting, at runtime, certain parts of the OPC tag. This allows objects or displays to be re-used dynamically for *different* OPC tags.

IMPORTANT:

In order for **aliasing** to be used, you must adopt a consistent naming scheme for similar objects in your database. For example, if you have 24 separate, identical pumps in your database, the signals should be consistent from pump to pump, i.e. PUMP1.RUN, PUMP2.RUN, PUMP3.RUN, etc. for all 24 pumps. If instead, you are inconsistent in naming signals i.e. PUMP.RUN.1, and PMPNUM3.RUN, etc., then aliasing CANNOT BE USED effectively.

The first step in using aliasing is to modify the OPC tags stored in your display files. Any part of a signal name which varies can be replaced with an **alias**.

For example, if your system has 8 identical water tanks, rather than creating 15 separate displays, one for each tank, you could create one display to handle all of them. That single display would use aliases to dynamically call in the correct signal data from the database for whichever tank the operator wanted to see.

To do this, your database must use a consistent naming convention, e.g.

TANK1.LEVEL.

TANK2.LEVEL.

TANK3.LEVEL.

:

:

TANK15.LEVEL

Now create a simply display for tank 1, using the TANK1.LEVEL signal. All text displayed should be taken from attributes in the database (name, value and units attributes in the Realanalog table.)

Verify that it works correctly in Runtime Mode.

Now, go to Configure Mode, and use the **Edit→Replace** command to change the OPC tags in the display to include an alias. Instead of TANK1, we want to use TANK<<tanknum>> where tanknum is an alias that gets replaced with a value of 1 to 8, depending upon which tank we want to look at. (IMPORTANT: Aliases are case-sensitive, i.e. Tanknum, TANKNUM, and tanknum would all be considered *different* aliases.) Click on **[OK]**, and execute the changes for all objects in the display.

Next, add a button to the display which will allow the operator to select which tank should be displayed. To do this, choose **Dynamics→Instrinsics→Pushbutton**.

On the 'Button' page of the Property Inspector, enter the label 'Choose Tank Number'.

On the 'Pick' page of the Property Inspector, choose **'Alias Dialog'** for the **"Action"**, then click on the **[Set Aliases]** push button.

In the Set Aliases dialog box, use the **[Aliases]** button to select the alias 'tanknum' (which should be the only one currently defined. 'tanknum' will be copied into the **"Alias Name"** field. Then enter the number '1' in the **"Alias Definition"** field, and click on the **[Add]** button. The pair 'tanknum' and '1' will be copied into the center box of the dialog box. Now enter '2' in the **"Alias Definition"** field, and click on **[Add]**. Repeat this process until all 8 tanks have alias definitions which go with 'tanknum'.

Click on **[OK]** and exit the Property Inspector as usual. Go into Runtime Mode. In Runtime Mode, a pushbutton now appears called 'Choose Tank Number'.

When the operator clicks on the push button, the Set New Alias Values dialog box appears. The operator chooses the desired tank number from the list box (or just enters the number), and it will be substituted in place of 'tanknum' in the OPC tags for this display. As a result, any of the 8 different tanks can be selected and displayed from this one display.

8.1.4 Other Uses of Aliasing:

- **Saving aliases as part of symbols**

  When we created symbols before, we didn't include dynamic information with them, because we were using different signals with the symbol each time we re-used it. Aliasing allows us to store dynamic OPC tag information when creating symbols for the symbol library to minimize the amount of typing when entering dynamic information. With this technique, you don't have to keep dragging tags from the Database Object Viewer for each use of the symbol because you do it once, and then use an alias in the tag which allows you to select it when re-using the symbol. For example, once we defined aliases for it, we could have grouped our entire tank and its associated dynamics together as a symbol with a shared object name, and saved it in the symbol library. Then we could have re-used it on other displays, and simply inserted the desired alias information when placing the symbol by using **Dyamics→Edit Aliases**. (NOTE: **Dynamics→Edit Aliases** makes the choice of aliases ahead-of-time, and permanently sets them for a particular display. In this case, no pushbutton for setting aliases is needed by the operator, because the operator does not make the choice, the choice was made when the symbol was placed within the display.)

- **Using aliases to pass data between OpenEnterprise Components**

Another powerful use of aliases is to allow data to be passed between OpenEnterprise components. Users can create customized trends, displays, etc. that can be activated through one OE component, and then data can be passed in using aliases, and the aliases dynamically filled in from columns of a Database Object Viewer query. See the *'OE Menus'* section of the *OE Reference Guide* (document# D5092) for more information.

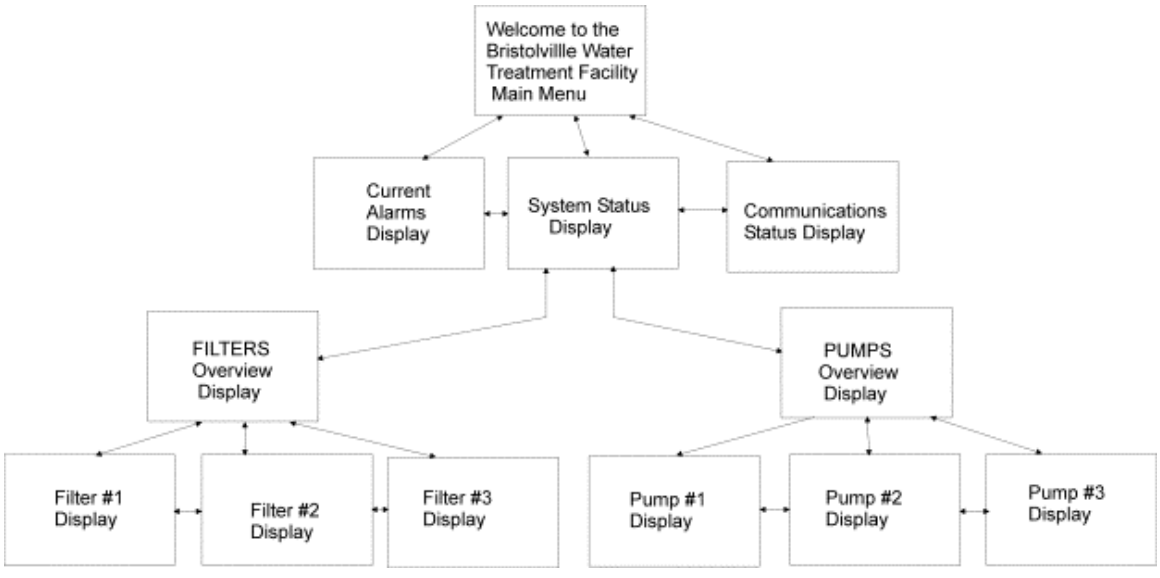
Tips For Planning Your Human-Machine Interface (HMI) System

9 Tips For Planning Your Human-Machine Interface (HMI) System

So far, we have discussed the 'mechanics' of how to create a particular display. The examples presented some basic techniques, and if you needed to do something not described in the examples, you referred to the *GraphWorX32 User Guide* to find out how to do it. Once you are familiar with how to create a display, you may be tempted to rush forward and start generating all of the displays for your particular system - - before you do this, though, *we strongly recommend you spend some time planning your* ***human-machine interface (HMI)*** *system.*

The goal of a human-machine interface is to provide your operators with the information they need in a clear, understandable format. Operators should not have to hunt for information during a critical situation; displays should be organized such that an operator can access the information he or she needs with a minimal number of mouse clicks.

Displays should be designed to be clear, and easy to use. Displays should not be overloaded with information; it is better to create a larger number of logically thought-out displays than to try to cram lots of information on a few displays, and then force the operator to sort out what is important.



With these thoughts in mind, the following steps outline a recommended approach for creating your HMI system:

Step 1 Plan out your display hierarchy

Step 2 Adopt a consistent method for navigation throughout the display hierarchy

Step 3 Determine what objects you will need on your displays

Step 4 Establish a signal naming, and an aliasing convention

Step 5 Build a few of your more complex objects for testing purposes

Step 6. Build your display hierarchy using a framework of empty displays

Step 7 Review your design with the users of the system, and revise, as necessary

Step 8 Build all of the objects for your system

Step 9 Build the Displays

Step 10 Test Your HMI System and Revise as Necessary#_ftn1

9.1 Step 1 Plan out your display hierarchy

Generally, at the top of the display hierarchy is some sort of startup or welcome screen. It might include a map of your system, or a menu screen.

A series of overview displays generally make up the next level of your hierarchy. You might want to have one or more overviews of your process, an alarm overview display, a communications status display, etc.

The next level of displays typically correspond to functional or geographical sections of your process. For example, if your system has 34 compressor stations, you might need to have a detailed display for each compressor station. If the compressor stations are all identical, you may be able to use a single display with aliases, which allow the operator to select the desired station, thereby reducing the amount of disk space used by the displays.

Your system might include additional displays for more detailed information for each station such as alarms, trends, communications, etc.

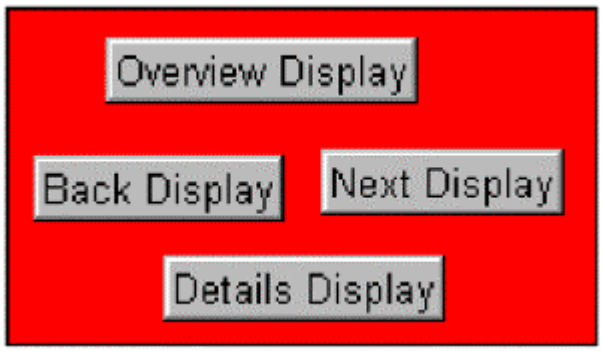
You might want to embed certain displays within other displays, for example, a small display for starting or stopping a pump, or a ‘faceplate’ display for entering setpoints.

Whichever way you choose to setup your hierarchy, it should follow a logical, consistent format. Operators should easily be able to navigate from one display to another, and to go up and down through the different levels of the hierarchy. ***If it takes more than 2 or 3 mouse clicks to reach the information you need, you should re-think your hierarchy.***

You should also consider what sort of display properties you want to set. For example, what size will your displays be? Will they appear in multiple windows, side-by-side, or will each new display called up fill the entire window? Will the displays be scaleable? All of these, and other properties of the displays, should be determined and set as preferences in the Display Properties dialog box *before* the displays are built, otherwise they will have to be set individually for each display.

9.2 Step 2 Adopt a consistent method for navigation throughout the display hierarchy

Each and every display should include a toolbar or menu object with links to other displays in the system. Although the display links in it will vary from display to display, you should include this object in the same place on every display, so operators know how to get to it quickly.



9.3 Step 3 Determine what objects you will need on your displays

The display building process will be much easier if you use a consistent library of objects for all your displays. You might, for example, determine that you will need three different pump symbols, two types of valve symbols, and one type of liquid tank, and that you want all of your process points to follow a particular format. These symbols, when you have built them (as part of a later step in the process) can be saved in a library, and re-used on all of your displays. ***Be sure to check and see what symbols are already available in your symbol library - - it may include most of the objects you need.***

9.4 Step 4 Establish a common signal naming convention for your database, and an aliasing convention for your displays

As we discussed earlier in this manual, aliasing is a powerful method which allows you to re-use objects, symbols, and displays, multiple times. In order for it to be used properly, however, your database must use a signal naming convention which includes common portions which can be used for aliasing. You can then define a strict aliasing convention which must be followed when building your displays.

In your database, therefore, if you have 50 identical compressor stations, each with its own set of signals from 50 different RTUs, make sure the signals names follow the same format in each RTU. For example, if you use signals like COMPRS8.POWER.STRT in one RTU, do NOT use COMPRESS.START.PWR5 in another. Use the same signal name format in each RTU.

Similarly, if you intend to define aliases for the station number, pump number, and tank number, make sure they are used consistently from display to display:

i.e. if you use:

station<<station-number>>

on one display, don't use:

station<<stationnum>>

on another display. Use aliases consistently from display to display.

Tips For Planning Your Human-Machine Interface (HMI) System

9.5 Step 5 Build a few of your more complex objects for testing purposes

Create some of your more complex objects, to verify that they work as expected. Setup aliases and verify that they work correctly. Be sure when saving symbols to use the shared object name, as discussed earlier in this manual.

Tips For Planning Your Human-Machine Interface (HMI) System

9.6 Step 6. Build your display hierarchy using a framework of empty displays

Build empty displays (without objects or data) for your entire display hierarchy. Make sure each display includes the toolbar or menu object used for navigation purposes. Test to make sure that the navigation functions work correctly throughout the display hierarchy. Put a few objects on the displays for demonstration purposes.

Tips For Planning Your Human-Machine Interface (HMI) System

9.7 Step 7 Review your design with the users of the system, and revise, as necessary

Now that you have a basic framework of displays, together with some demo objects, you should show the system to the people who will use it. *This is a very important step.* You may find, for example, that the operators need additional information on certain displays, or that they need to have multiple displays open to perform certain tasks. Perhaps your hierarchy is too cumbersome, or requires too many mouse clicks to locate certain displays. Revise the system, as necessary, to accommodate your user's concerns.

Tips For Planning Your Human-Machine Interface (HMI) System

9.8 Step 8 Build all of the objects for your system

Create all of the objects needed for your system (pumps, valves, tanks, compressors, analog setpoints, faceplates, etc.) and set up aliases for them according to the previously established convention. Save the objects in the symbol library with shared object names.

Tips For Planning Your Human-Machine Interface (HMI) System

9.9 Step 9 Build the Displays

Finally, you should build the displays using the objects from the symbol library. Each display should be included in the proper place of the previously defined hierarchy.

Tips For Planning Your Human-Machine Interface (HMI) System

9.10 Step 10 Test Your HMI System and Revise as Necessary

Test each of your displays, and verify that they function as expected. Make any necessary corrections.

Tips For Planning Your Human-Machine Interface (HMI) System

10 Index

Reference Guide
D301491X412
April 2012

DISCLAIMER

Engineered and supported by:

Remote Automation Solutions,

Blackpole Road, Worcester, WR3 8YB, UK

Registered office: Meridian East, Leicester, LE19 1UX

Registered in England and Wales, Registration No. 00671801

VAT Reg No. GB 705 353 652

Emerson Process Management
Remote Automation Solutions
1100 Buckingham St
Watertown, CT 06795
T 1 (860) 945 2200
F 1 (860) 945 2278
www.EmersonProcess.com/Remote
binfo@EmersonProcess.com

Emerson Process Management
Remote Automation Solutions
Blackpole Road
Worcester, WR3 8YB
T 44 (0) 1905 856848
F 44 (0) 1905 856930
www.EmersonProcess.com/Remote
oedsupport@EmersonProcess.com