Chapter 1 –Getting Started

Introduction

Catalyst Pro is a software interface between you and your building automation or process control system.

With Catalyst Pro you can:

- Monitor and control all aspects of your system in real time using a variety of meters, gauges, switches and graphs.
- Easily change setpoints and schedules.
- Collect and display historical data.
- Collect and manage alarms.
- Backup and restore your systems memory.
- Maintain a system log so you always know who changed what, and when it was changed.

Some of the features include:

Sites:

- An unlimited number of sites can be maintained in the Site Database.
- Sites may be accessed through a modem or directly connected with a cable.
- Sites may be grouped by region, customer, type of job or any other way you choose.
- Connect to multiple sites and several different manufactures equipment simultaneously.

Graphic Displays:

- An unlimited number of displays can be maintained in the Displays Database for each site.
- Displays may be grouped by floor, type or any other way you choose.
- Displays may contain full color pictures or drawings.
- Each display may contain analog and digital meters, knobs, switches, bar charts, etc. for system monitoring or control.
- Click on a switch to control a pump, turn a knob to change a setpoint, etc.
- Click on the display to switch to a different display, view a Schedule or display historical data.
- Data from multiple sites may be placed on a single display.
- View several displays simultaneously.

Schedules:

- An unlimited number of Schedules can be maintained in the Schedule Database.
- Use the built-in schedule for quick setup or customize a schedule to exactly fit your needs.
- Built-in schedule has 4 ON/OFF pairs for Sunday - Saturday, 12 special holiday periods and 8 closed dates.

Trending:

- Automatically collect and maintain historical data on all aspects of your system.
- View historical data with a variety of 2D or 3D line and bar charts.
- Chart appearance is fully customizable.

Alarms:

- All alarms are stored in a central database for quick access.
- Three levels of alarms are supported: Alarms, Warnings and Informational messages.
- Alarms remain active until they are acknowledged at which time you may attach a comment to the alarm (why it happened, what was done about it, etc.)
- Alarms may be sorted by type, date or alphabetically.
- Alarms may be exported to a Microsoft Excel compatible file.

Users:

- An unlimited number of users can be maintained in the User Database.
- Each users access may be customized to enable/disable access to most functions.
- Users can be temporarily suspended.

System Requirements

Minimum System Requirements

- Pentium or compatible system running Windows 95/98/ME/2000/NT/XP
- 32 megabytes of RAM
- Super VGA video card and Super VGA color monitor
- 500 megabyte or larger hard disk
- Mouse or other Windows compatible pointing device

Recommended System:

- Pentium III 500 system running Windows 2000/NT/XP
- 64 megabytes of RAM
- Super VGA video card with at least 800x600 resolution
- 5 gigabyte or larger hard disk
- Mouse or other Windows compatible pointing device

How to use this Manual

Several times throughout this manual and the built-in help files you may see something like the following:

- **File|Setup|Control Points** – Indicates you should first select the "File" menu, then select "Setup" and finally select "Control Points".
- **[Enter], [Esc] or [Space]** – Indicates you should press the key enclosed in the brackets.
- **<Ctrl>-B** – Indicates you should hold down the <Ctrl> key and then press the "B" key.

Several icons are used in this manual to mark certain important or interesting topics:

Indicates an important topic. You should always read and understand these notes.
Indicates a technical note. It is not important for non-technical users to read these.
Indicates a timesaving tip.

Running Catalyst Pro

Select 'Programs' from the 'Start' menu, then select ''Catalyst Pro".

If you would like Catalyst Pro to run every time Windows is loaded, open the 'Startup' folder and create a shortcut to Catalyst Pro. Refer to your Windows documentation for details on how to do this.

Command-Line Parameters

The following command line parameters are supported:

- **/uName** - Automatically log in as user, requires /p also.
- **/pPassword** - Password for auto log-in, requires /u also.
- **/Debug -** Always connect in debug mode (make terminal visible.)

These options should be placed in the "Target" section of the Catalyst Pro shortcut on the desktop.

Example: C:\Catalyst\Catalyst.exe /uJerry /pMyPassword /Debug

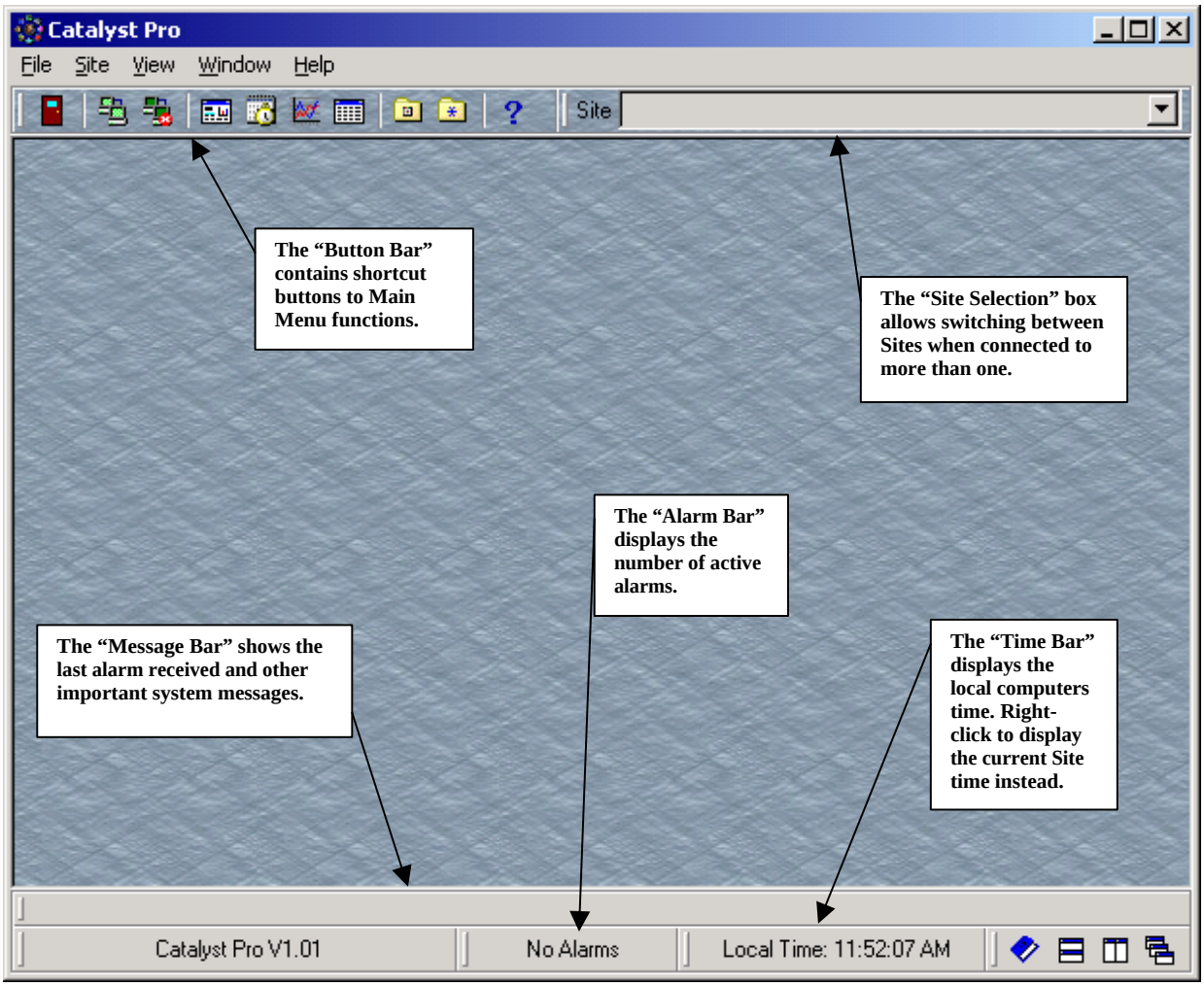
Logging in to Catalyst Pro

At the log in screen (shown below) type your username, press [Tab], then type your password and press [Enter]. The default username is 'Master' and the default password is also 'Master'. You should change this as soon as possible to avoid unauthorized access (See "Users and Security" for more information).

You can have Catalyst Pro automatically log in for you every time it is executed. See "Command Line Parameters" for more information.

Main Menu Screen

Once you have logged in to Catalyst Pro you will see the 'Main Menu' screen (shown below). The main menu is displayed when you are logged in to Catalyst Pro. The area between the Main Menu and the 'Status Bar' at the bottom is where all Graphic Displays, Schedules, history charts, etc. will appear when in use, but for now it is empty.



You can have Catalyst Pro automatically connect with a Site and open a Graphic Display every time it is executed. See "Program Options and Customization'" for more information.

Almost every aspect of Catalyst Pro is customizable. All of the button bars, message bars, main menu selections and even the name of the program itself can be changed or hidden. Right-click on something and chances are it can be customized. See "Program Options and Customization" for more information.

Chapter 2 – Sites

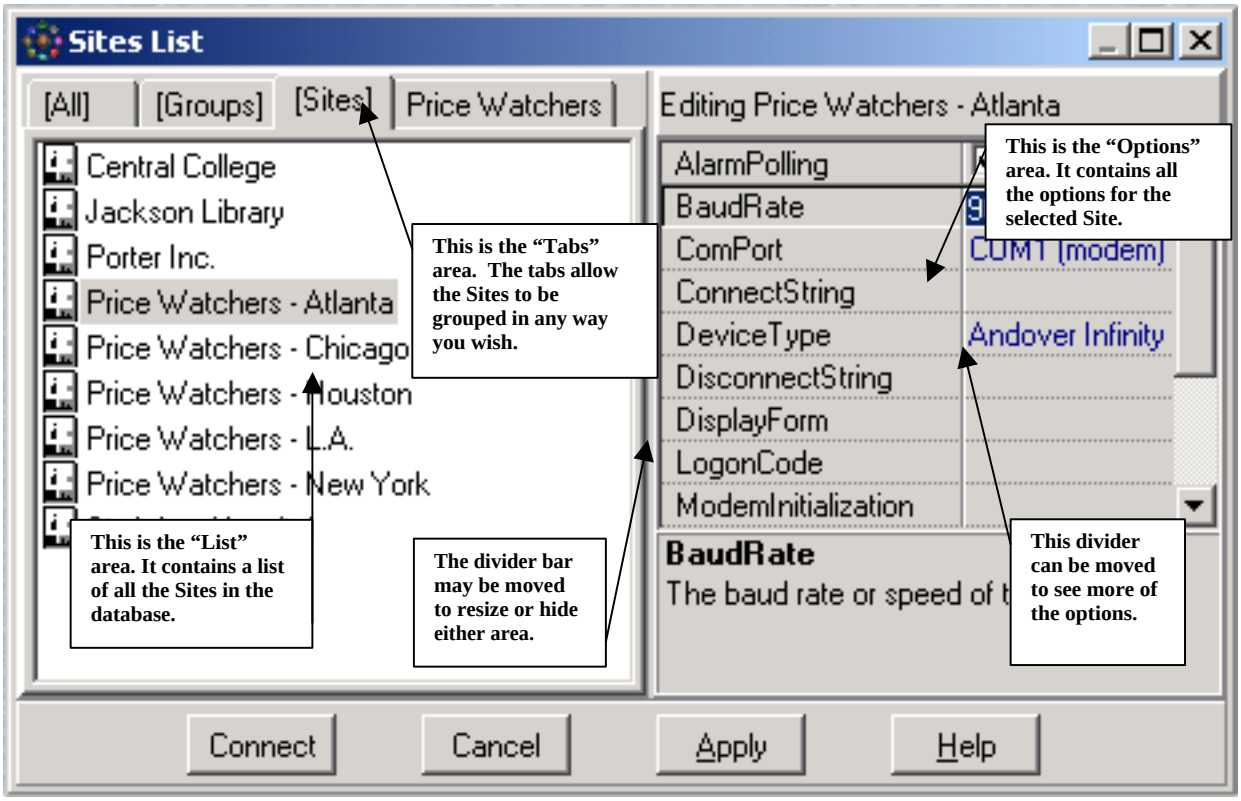
Editing the Site Database

"Site" is the term used to describe a given panel or set of panels at a particular location.

Three Andover Infinity panels networked together at a single location would be considered a single "site". Each Infinity would be individually referred to as a "panel".

A Site can contain an unlimited number of panels and the database will hold a virtually unlimited number of Sites.

✪ The Site Database uses the same type of form that is used throughout Catalyst Pro for lists of Displays, Schedules, Control Points, etc. A working knowledge of how to use these databases is very important. Fortunately, they are also quite easy to understand and use.



The Site Database has three main parts.

- The white area on the left is the list of Sites area.
- The gray area on the right is the options or properties area for the selected Sites.
- The gray area immediately above the list area is the tabs area. The tabs are used to group Sites together in any way you wish.

There is a divider bar between the list area and options area that may be moved to resize both areas. This is useful if you have very long Site descriptions or may be necessary at times to see all of the options. The size and position of all the areas will be remembered and restored each time the form is opened.

Creating a New Site

To add a new site to the database, right-click inside of the list area and select "New Site". You will then be asked to provide a description for the Site. This can be any combination of letters, numbers and underscores that you wish to use. Every Site must have a unique name.

Once the Site has been added to the list, select it by left clicking on it and all of its options will appear in the options area. As you click on each option, a brief description will be displayed in the lower portion of the options area. The exact options for each Site will vary depending on what type of equipment you are connecting to. For the Andover Infinity, you normally only need to provide the **BaudRate**, **ComPort**, **LogonCode** and **Password**. You will also need to supply the **PhoneNumber** if connecting via modem. See the section of this manual specific to your equipment for a detailed explanation of all options.

The **RemoteName** property should be left blank unless you are connecting to a driver running on a remote computer. If you are running the driver from a remote computer, the **RemoteName** should be the network name of the computer (for example: "R-D_Lab", "MechanicalRoom", "Larrys_Pentium", etc.). If connecting over the Internet, you can supply an I.P. address or a URL (for example: 64.123.55.123, or www.mydriversite.com/InfinityDriver). See “Networking and Remote Access” for more information.

There are several different types of options and the way you change them varies somewhat:

- **Check Boxes** - These are used to select a value of TRUE or FALSE. The AlarmPolling is a good example of this. Check the box if you want this computer to poll the selected Site for alarms once every minute while connected. Un-check the box to disable alarm polling.

- **Drop Down List** - This is used to select one of several preset values. The BaudRate and ComPort options are Drop Down List type options.

- **Edit Boxes** - These are miniature text editors where you can type any text you wish. The Password and LogonCode options are Edit Boxes.

- **Advanced Options** - These are used to set options that require another form or dialog box to be displayed. They have a small "..." button to the right of the options value. Click this button to display the advanced options form.

Once you have set all the required options, click the "Connect" button to connect to the Site.

Deleting Sites

Sites may be deleted from the database by right clicking on the Site in the list area and selecting “Delete”. Because the simple mistake of a user deleting the wrong Site could erase days of work, and potentially years of accumulated data, no files are changed or erased. If you are sure you want to totally erase a Site and all of its data files, use Windows Explorer to delete that Sites folder.

Each Site stores all of its data files in a folder with the same name as the Site description surrounded by square brackets. For example, a Site named “Central College” would store all of its data files in the “C:\Catalyst\Sites\[Central College]” folder. When a Site is deleted from the database, a “~” is added to the folder name. The previous example would then become “C:\Catalyst\Sites\~[Central College]”. The “~” can be removed to restore the Site to the database the next time Catalyst Pro is executed.

Site Groups

Site Groups allow you to connect to multiple sites with a single click of the "Connect" button. Add a new Group the same way you added a new Site. Select the "**SitesInGroup**" option to pick which Sites you want to be in your new group. You may then select the new Group and click the "Connect" button to connect to all sites in the group.

✪ While there is no artificial limit placed on the number of Sites that can be in a Group, any more than 3 to 5 Sites is usually not practical because of the time required to connect to each of them. Also, keep in mind that connecting to 10 sites each with 8,000 Control Points can start to use up a lot of system memory.

Using Tabs to Organize Sites

The tabs may be used to organize your list of Sites by region, state, type of equipment or any other way you wish.

The Tabs area contains 3 default tabs. "All", "Groups" and "Sites".

- The "**All**" tab always contains all of the Sites and Groups in the database. You cannot remove items from the "All" tab without deleting them from the database.
- The "**Groups**" tab always contains all of the Groups in the database. Items may not be removed from the "Groups" tab.
- The "**Sites**" tab always contains all of the Sites in the database. Items may not be removed from the "Sites" tab.

While you cannot remove items from the default tabs, the tabs themselves may be hidden once you have at least one custom tab.

Adding Custom Tabs

To add a new custom tab, right-click in the tab area and select "New Tab" and type in the text you want to appear on the tab.

Once you have a new tab, any number of Sites may be dragged from the "[All]" tab and dropped on the new tab to add it to that list. Multiple Sites may be selected at once by holding down <Shift> or <Ctrl> while clicking on the Sites. <Shift> will select a range of Sites, <Ctrl> allows you to select Sites in any order.

🕒 The name given to a tab may contain an '&' to make the next letter underlined. For example, '&Western US' will appear on the tab as Western US. You can then press 'W' to switch to that tab rather than clicking it.

Connecting to a Site

To connect with a Site, click the left mouse button on the desired Site in the list area and click the 'Connect' button. Catalyst Pro will then attempt to connect with this Site. If all the options for this Site are set correctly you should now be connected. The Site Selection bar at the top of the screen will display the Sites description.

🕒 You can also "double click" the left mouse button on the Sites icon instead of clicking the 'Connect' button.

Connecting to a Group of Sites

To connect with a group of Sites, click the left mouse button on the desired Group in the list area and click the 'Connect' button. Catalyst Pro will then attempt to connect with all Sites in the Group. The Site Selection bar at the top of the screen will display the description of the default Site for this Group. The other Sites may be selected by scrolling through the list of Sites in the Site Selection bar.

You can also "double click" the left mouse button on the Groups icon instead of clicking the 'Connect' button.

Disconnecting from a Site

To disconnect from a Site or group of Sites select "Disconnect" from the "Site" menu or click the "Disconnect" button on the button bar. Catalyst Pro will then log-off and disconnect from all Sites to which it is currently connected.

Offline Editing

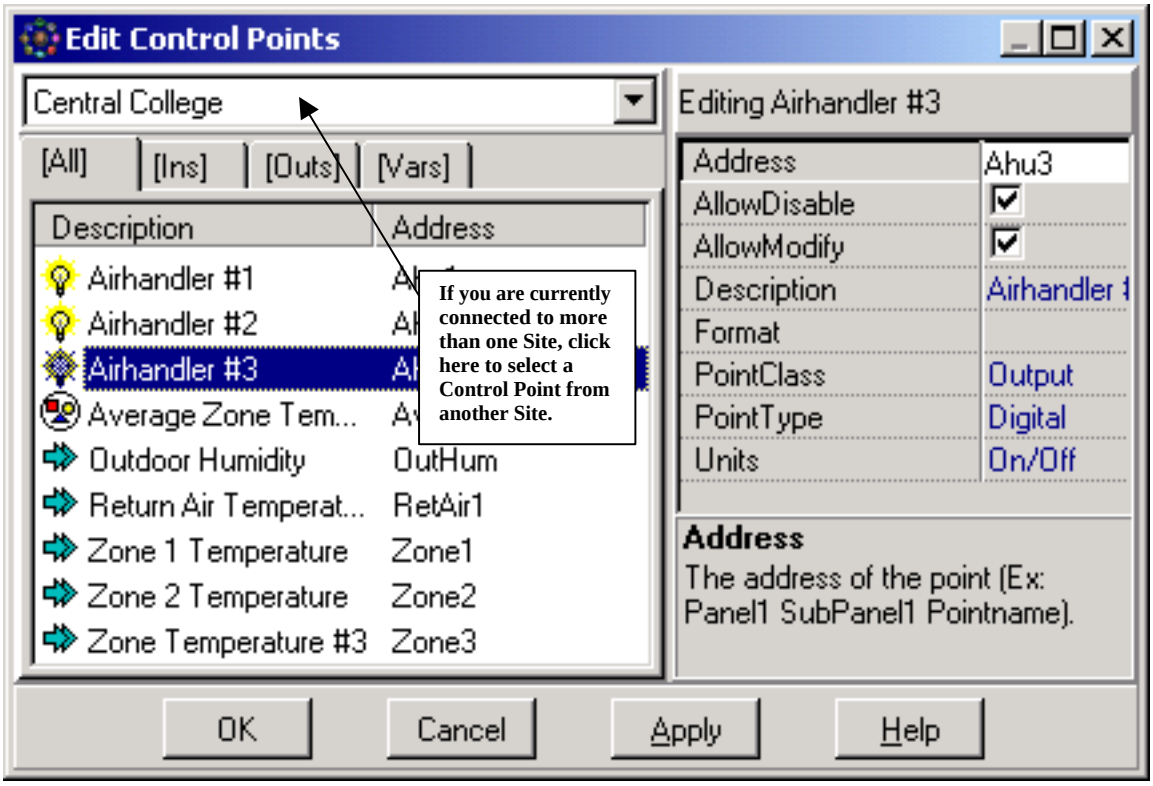
It is possible to connect to a Site in "Offline" mode. This allows you to create Graphic Displays, set up trends and perform most other functions without actually connecting to a Site. Select "Site|More|Offline Edit", highlight a Site and then click the "Connect" button.

Chapter 3 – Control Points

Overview

Control points are descriptions of inputs, outputs and variables stored on panels at a given Site. All Control Points that are to be used in Graphic Displays or historical trending must be entered in to the Control Point database.

(See "Editing the Site Database" for more information on editing database entries).



If you are currently connected to more than one Site, use the drop-down list above the tabs area to choose the Site from which you wish to select or edit a point.

Adding Points to the Database

To add a Control Point to the database, right click in the list area and select "Add Point". A Control Point input form will be displayed that makes it easier to add points to the database. After entering all the options for the point, Catalyst Pro will verify that the point address is valid.

Options for Control Points include:

- **Address** - The address of the point (Infinity Example: Infinity1 EastRoom ZoneTemp). You should always specify the complete path to the point whenever possible.

- **AllowDisable** - Allow the point to be disabled from within this program.

- **AllowModify** - Allow the value of the point to be changed from within this program.

  ✪ Any point that will be used with a switch, knob or other control that can be changed by the user MUST have AllowModify set to TRUE.

- **Description** - A description of the point. The description may be any text you want and it does not have to match the description of the point in the panel. It is used only in this program for the benefit of the users and is not used when communicating with the panels.

- **Format** - The format for displaying the value on Graphic Displays. The format can be a series of "pound" signs including an optional decimal place (####.## to display the value as 1234.56) or it can be customized using the built-in Format Strings. If Format is left blank, the default of ####.## is used.

- **PointClass** - The class of the point. The point Class must be either Input, Output, Variable or Unknown. It is recommended that any point that is not a physical Input or Output be set to Variable. This includes flags, strings and date variables. The point class "Unknown" is only used for points that are automatically imported but for which complete information could not be obtained. You should never purposely set a point class to "Unknown".

- **PointType** - The type of the point. This can be Unknown, Digital, Analog, Tri-State, String, Date, Time or DateTime:

  - **Unknown** - Only used for points that are automatically imported but for which complete information could not be obtained. You should never purposely set a point type to "Unknown".
  - **Digital** - A digital or ON/OFF only point. This would include boolean flags as well as physical outputs or inputs.
  - **Analog** - A linear, non-digital input, output or variable. This includes "numeric" points, temperature inputs, voltage outputs, etc..
  - **Tri-State** - a 3 state output. Possible values are -ON (negative ON), OFF and ON. Numerically they equate to -1, 0 and 1 respectively.
  - **String** - A string of ASCII characters.
  - **Date** - A date format specific to this program. This is not the same as a date variable in the panel. The format is: ((Year-1900)*1,000) + DayOfYear (For example: 01/01/2002 = 102,001). For built-in Date variables, set the point type to "String".
  - **Time** - A time format specific to this program. This is not the same as a time variable in the panel. The format is: (Hour*100)+Minute (For example: 12:30pm = 1,230).
  - **DateTime** - This is a Microsoft Windows compatible DateTime variable. It is included for future versions of the program. It is unlikely that you will need to use this.

- **Units** - The electrical or descriptive units associated with the point. Select the units from the drop-down list or just type the units the way you want them to appear. You can also click the "Edit" button to add your own custom units to the list so you don't have to keep re-typing them.

If any changes are made to the Format or Units, any Graphic Displays that contain the point must be closed and re-opened to view the changes,

Importing Control Points from a Site

Control Points can be automatically imported from most types of panels. This can save hours of time over entering all of the points manually. To import points, right click the list area of the Control Point database and select “Import Points”. See the section of this manual specific to your equipment for further information.

Chapter 4 – Graphic Displays

Overview

Graphic displays are the core feature of Catalyst Pro. Displays allow you to graphically represent your building floor plans, HVAC equipment, lighting layout and all other aspects of your automation or process control system.

Viewing Graphic Displays

To view an existing Graphic Display, select “Displays” from the “View” menu. Highlight any of the Graphic Displays in the database and click the “Select” button.

Depending on what type of equipment you are connected with, it can take between 3-10 seconds for the values to fill in on the display. If after 10 seconds you do not see any values filling in and no meters or gauges have moved, please check your connection to the Site by selecting “Status” from the “Site” menu.

✪ Please keep in mind, especially for RS-232 and modem connections, every Graphic Display that is opened will slow down the updating of all other Graphic Displays. It is recommended that you always close any Graphic Displays that are no longer needed.

Manipulating Graphic Controls

Many of the component on Graphic Displays can be manipulated in some way. Below is a partial list of these controls and how to use them.

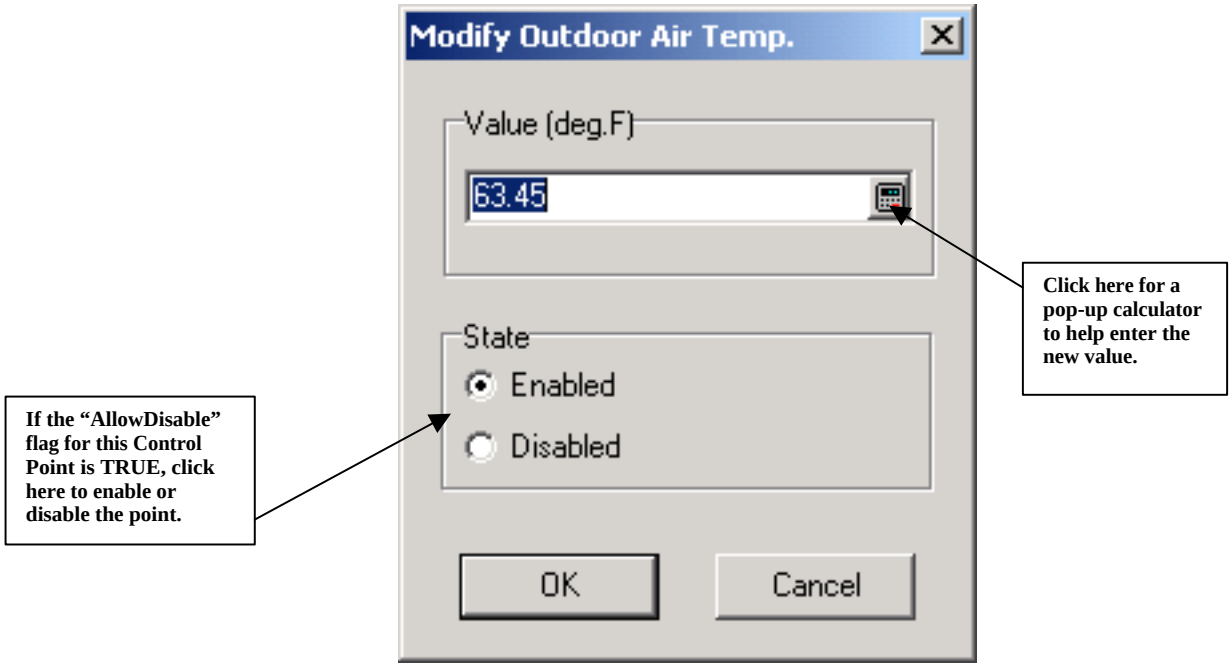
- **Button** – Click the button once to perform whatever action has been assigned to them.
- **Lever, Rocker or Toggle Switch** – Click the switch once to toggle it from its current position. Click a second time to return it to its original position.
- **Knob** – Left Click on the colored indicator and drag the mouse in a circular motion the way you want the knob to turn. It is sometimes possible to move the mouse farther away from the knob while dragging it to get a higher precision.
- **Rotary and Slide Switches** – For multi-position switches, either click on the position you want the switch to be in, or left click the handle of the switch and drag it in to the proper position.
- **Analog Sliders** – Click the colored indicator on the slider and drag it to the desired position. You can also click above and below the indicator (or left and right for horizontal sliders) to move it in small increments towards the value.

Manually Modifying Point Values

Any component on a Graphic Display that has a Control Point assigned to it can be manually modified (assuming the AllowModify flag for that Control Point is set to TRUE). To modify the point value, right click the mouse on the component and select "Modify Point".

- If the "Modify Point" option does not appear, there is either no Control Point assigned to the component or the Control Point has not been set to allow modification (see "Adding Points to the Database" for more information.)

The exact dialog box that appears is dependant on the type of Control Point that is assigned to the component. Below is an example for an analog input such as an outdoor air temperature sensor.



Type the new value for the Control Point and click the OK button. The Control Point will then be modified on the panel.

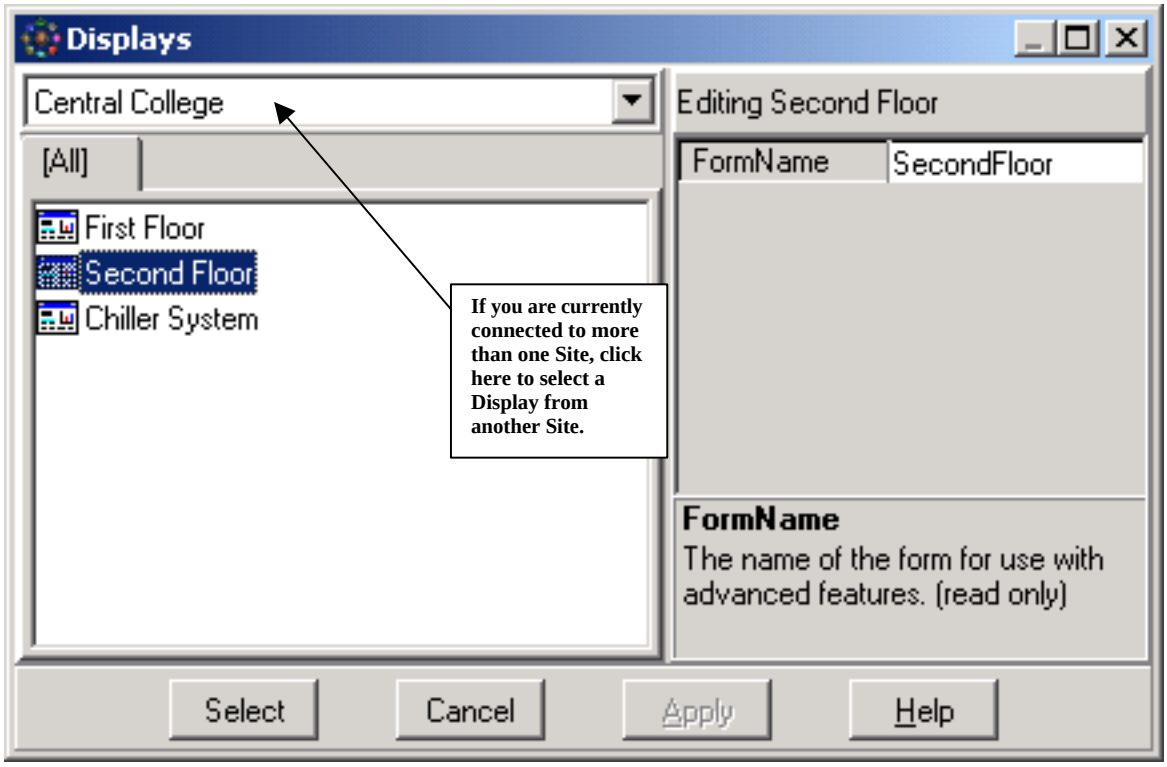
- On most types of panels, simply changing the value will not work unless the point is also disabled. Disabling a point will effectively freeze the value and will prevent the panel from seeing its true value. This is usually only done for testing purposes or as a temporary solution to a bad sensor. It is not recommended that you disable points unless you are sure it is necessary.

Editing the Displays Database

The Displays Database is where the list of Displays for each Site is stored. It is very similar to the other database screens used in this program. See “Editing the Site Database” for more information on editing the entries.

There is one important difference with the Displays Database compared to the Sites Database. There is a drop-down list above the tabs area that contains the list of all Sites you are currently connected to (In this example, "Central College" is the selected Site).

Each Site maintains it's own list of Displays so if you are connected to several Sites, use the drop-down list to view the Displays for each of them.



Creating a New Display

To create a new Display, right-click in the list area and select "Add Display".

- There is no real difference between Graphic Displays and Custom Forms. They are stored in different locations on your computer and are maintained in separate databases, but are otherwise virtually identical. A distinction is made only for the benefit of the more casual users who want to design Displays but do not wish to use to use the Windows components or write script code. See “Custom Forms” for the more advanced features of Displays and Custom Form.

The first thing you will need to do is select a form "Template". A form template is a special file that tells the program what the Display should look like and how it should behave. When the program is first installed, the only template available is "Blank Display". This is the basic blank Display form with the scripting language set to Visual Basic Scripting.

✪ Please note that you do not have to use or write any script code in order to make your own Displays. The scripting language is only needed for advanced features but it still must be specified when you create a new Display.

🕒 Once you have designed a Display, for instance, a fan control Display, you can save the Display as a form template. The next time you need a fan control Display you can select the template you saved earlier, pick the new Control Points and your work is done.

After selecting the template, type in a description for the Display. This is the description that will appear in the Displays database, and also on the caption bar of the Display itself.

Next you will be asked for a unique name for the Display. This cannot be the same name as any other Display, Form, Schedule, etc. that is used for this Site (the program will let you know if the name is already in use). It is only used for script programming but must be supplied even if you are not going to be doing any scripting. You can usually just accept the default name it suggests.

Your Display should now be created and added to the database. Highlight your new Display and click the "Select" button to edit or view it.

Options

The only option available in the Options area is the FormName, which cannot be changed once the Display has been created. All other options for the Display are changed in the Form Editor.

Editing Graphic Displays

Before editing your first Graphic Display, let's define a few terms:

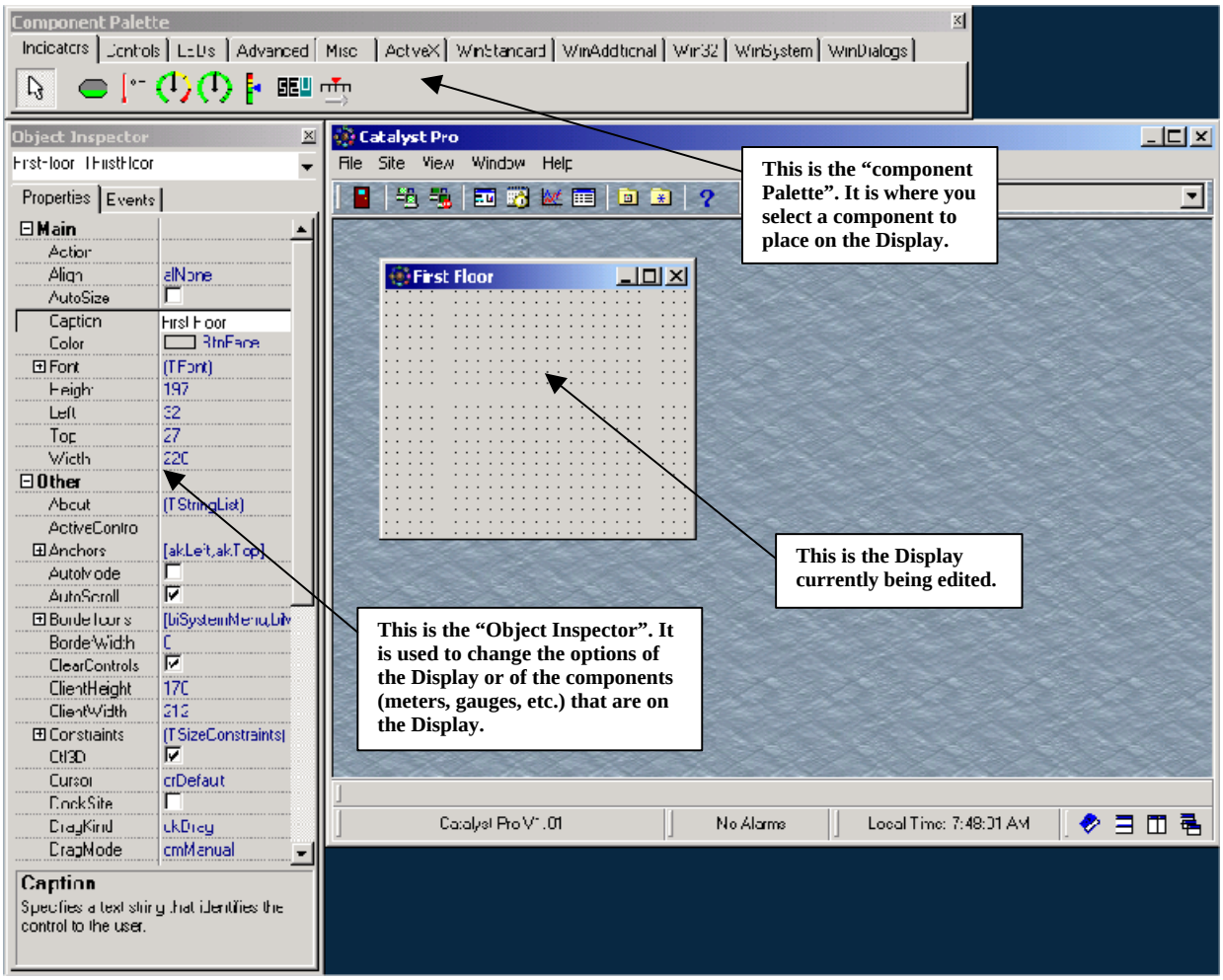
- **Display** (also called a "Form") – This is the form or window that you are editing and can contain meters, gauges, text, pictures, etc.

- **Component** – This is any meter, switch, picture, text string or button that is placed on the form. Basically, every object on the form is generically referred to as a "component". Each Display can contain a virtually unlimited number of components.

- **Property** (or "Option") – These are variables specific to each component that control the size, color, look and behavior of the component. For Example, the "Caption" property of a button is used to specify the text that will appear on the button. All components have a pre-set number of properties. Many of the properties are the same from one component to the next, but some components also have unique properties.

To edit a Graphic Display, select the desired Display from the Displays database and click "Select".

You should now have a small blank display visible on the screen. To edit the Display, right click the Display and select "Edit Mode". Your screen should then look something like this:



This is the Display editor. It consists of the Display you are editing, a "Component Palette" and an "Object Inspector".

The **Object Inspector** is the list of properties or options for the selected component (the color, size, position, etc.). The Object Inspector works exactly like the options section of the Site database form. It has two categories of properties; "Main" and "Other". You will usually only use the "Main" properties. The "Other" properties are used for advanced script programming, but feel free to experiment with them.

The **Component Palette** is the list of available components that can be placed on the Display.

✪ The first time you use the Display editor, the Object Inspector and the Component Palette will be placed in fairly random positions. Move them wherever you would like them to be and they will stay there every time you edit a form.

You can right click on the Component Palette to import ActiveX controls, customize the tabs on the palette or rearrange the components on the tabs.

✪ **IMPORTANT**: Not all ActiveX controls are designed to be used in this manner. Some even have code in them to stop them from being used like this. It is possible that some ActiveX controls will generate error messages when placed on a form. It is even possible that some will generate so many errors that it will not be possible to remove them from the form, thus making the form unusable. Make sure to test any ActiveX controls you want to use on a blank form before you place them on a good form.

Adding Components to a Graphic Display

To place a component on a Display:

1. Click one of the icons on the component palette and then click anywhere on the Display. For this example, select "Meter" from the "Indicators" tab. (Keep the mouse over an icon for a few seconds to see the name of a component.)

2. Select a Control Point. Select a Control Point from the database that is appropriate for the selected component. For a meter component, a temperature, humidity or fan speed input would be a good choice. (Some components do not require a Control Point. A component such as a button does not display or directly control a point's value so a Control Point would not have to be selected.)

3. Select a Component Template. Component templates are a way to quickly customize a component without having to constantly change dozens of properties every time you want a particular style of meter. Choose one of the templates provided and click "Select". (Some components do not come with pre-designed templates but you can always create your own. See "Component Templates" for more information.)

Once a component has been placed on a form, it can be moved by left clicking on it and dragging it to the desired location. The component can be resized by left clicking and dragging any of the dark colored "handles" on the corners and edges of the component.

Leave edit mode by pressing F9, or by right clicking on the Display and selecting "View Mode". The Display should now update within a few seconds with the point values from the panel.

✪ There is an "Undo" available in the right click menu of edit mode, but until you become more familiar with editing forms, save your work often and make backup copies at regular intervals (using the "Make Backup Copy of Form" option in the right click menu of edit mode).

🕒 Once you have completed designing a Display, you can select "Quick Load" from the right click menu in edit mode. This will load a small program in to the panel that will greatly speed up access to the point data for most types of panels.

Changing Component Properties

With a Graphic Display opened and in edit mode, left click on the component you wish to change. Multiple components may be selected by holding down the <Shift> key as you click on the components. You can also hold the left mouse button down and draw a box around multiple components. This will select all components within the box. A third method to select components is the "component list" at the top of the object inspector. It is a drop down list of all components on the form. It may be necessary to use this if, for instance, a component is behind another component.

After selecting one or more components, find the property you wish to change in the object inspector and click on it. The lower portion of the object inspector will display a short description of what the property is for.

There are several different types of properties and the way you change them varies somewhat:

- **Check Boxes** - These are used to select a value of TRUE or FALSE. The "Visible" property of most components is a good example of this. Check the box if you want this component to be visible. Un-check the box to make the component invisible when in view mode.

- **Drop Down List** - This is used to select one of several preset values. Click the downward pointing arrow to select a value. The "PointerColor" of the Meter component is a Drop Down List type property. You can sometimes also double click these types of properties to display an advanced dialog box (for example double click any "Color" property for a custom color creation dialog box.)

- **Edit Boxes** - These are miniature text editors where you can type any text you wish. The "Label1Text" property of the Meter component is an Edit Box.

- **Advanced Properties** - These are used to set options that require another form or dialog box to be displayed. They have a small "..." button to the right of the options value. Click this button to display the advanced properties form. The "Action" property of the "ActionButton" component is an example of an advanced property.

Feel free to experiment with all of the properties. You do not have to worry about corrupting or destroying the Graphic Display by simply changing property values. Some property changes may not have the effect you wanted, but you can always change the property back to its original value or delete the component and start over.

✪ There are dozens of components and each of them can have dozens of options. It is easy to become overwhelmed if you try to understand them all right now. You will usually not have to change many of the options (if any at all). The large number of options is what gives Catalyst Pro its tremendous flexibility but you can ignore most of them for now.

✪ The object inspector includes a tab marked "Events". The "Events" are for advanced script programming and can be ignored by most users. See "Custom Forms" for more information on Events.

Edit Mode Menu

Below is a description of the menu selections in edit mode. Access this menu by right clicking on the Graphic Display while in edit mode. Note that sometimes a menu selection is disabled or not visible at all. This occurs when that selection is not currently applicable. For example, "Select Control Point(s)" is only visible when you right click a component on the Graphic Display that can accept a Control Point, such as a meter or switch.

- **Select Point(s)** – Allows you to change the Control Point that is assigned to the component you clicked on.

- **Component** - Contains the following sub-menu selections:

    - **Add Point Link** – Allows you to assign a Control Point to an ActiveX control. See "Using ActiveX Controls" for more information.

    - **Edit Point Links** – Displays a list of all components on the Graphic Display that have Control Points assigned to them and allows you to change the points assigned to them. This makes it easier to create multiple Graphic Displays that are almost identical except for the Control Points such as VAV or Fan systems.

    - **Save as Template…** - Saves the component as a "template" so you can easily create similar components in the future. See "Component Templates" for more information.

    - **Load From Template…** - Load the components properties from a "template". This allows you to quickly change the components appearance.

- **Form** - Contains the following sub-menu selections:

    - **Save Form** – Saves the Graphic Display to disk. Use this often while designing new Graphic Displays.

    - **Make Backup Copy of Form** – Makes a backup copy of the Graphic Display with the same name as the original but with a ".Bak" file extension. You should always make backup copies of Graphic Displays periodically while you are designing them.

    - **Save Form as Template…** - Saves the entire form as a form template. This is useful when you need to design several Graphic Displays that are identical except for the Control Points that are used. See "Form Templates" for more information.

    - **Enable QuickLoad** – Used to speed up access to the Control Point data when viewing the Graphic Display. The way this works varies from one type of panel to another. See the section of this manual specific to your hardware for more information.

- **Edit** - Contains the following sub-menu selections:
    - **Delete** – Delete the currently selected components.
    - **Cut** – Delete the currently selected components and place them in the Windows Clipboard so they may be pasted on to another Graphic Display.
    - **Copy** – Place the currently selected components in the Windows Clipboard so they may be pasted on to another Graphic Display. This function does not delete the components from the current Graphic Display.
    - **Paste** – Paste the contents of the Windows Clipboard to the current Graphic Display. Note that this only works with components that have been cut, or copied from a Graphic Display, not from other Windows programs.

 It is possible to cut or copy a component (or several components at once) from a Graphic Display and then paste them in to a text editor. You can then edit the components in the text editor and paste them back on to the Graphic Display. This is recommended only for advanced users as invalid property values can cause very unpredictable results.

- **Grid** - Contains the following sub-menu selections:
    - **Show Grid** – Hides or displays the grid of dots on the Graphic Display in edit mode. The grid can be helpful as an aid in placing and lining up components.
    - **Snap to Grid** – Forces components to only be placed on grid marks. This can be helpful to keep components lined up with each other. Uncheck this option to place a component anywhere on the Graphic Display.
    - **Grid Spacing** – Allows you to change the spacing between the grid dots.
- **Adjust** - Contains the following sub-menu selections:
    - **Align to Grid** – Aligns the selected components to the nearest grid dots. This is used to align certain components to the grid when “Snap to Grid” is off.
    - **Bring to Front** –Will force the selected components to come to the foreground if there are other components overlapping them. Some components cannot be placed in front of other components.
    - **Send to Back** - Will force the selected components to go to the background (behind other components.) Some components cannot be placed behind other components.
    - **Align…** - Allows you to align several components to each other. For example, force three meters to move so that the left edges are aligned with each other, or to center a component in the Graphic Display.
    - **Size…** - Allows you to resize several components at once. For example, make three buttons all the same width, or height.
    - **Scale…** - Allows you to resize components based on a percentage. For example, make a switch 150% bigger.

  - **Tab Order…** - Allows you to change the tab order of the components. The tab order determines what order the components are highlighted when you press [Tab] in view mode. This is not normally necessary for Graphic Displays.

- **Show** - Contains the following sub-menu selections:

  - **Script Editor** – Displays the script programming editor or brings it to the foreground if it is already open. This is not normally necessary for Graphic Displays. See “Custom Forms” for more information.

  - **Component Palette** – Displays the Component Palette if it is not currently visible.

  - **Object Inspector** - Displays the Object Inspector if it is not currently visible.

  - **Alignment Palette** – Displays a floating dialog box that allows you to quickly align components on the Graphic Display.

- **Undo** – Undo a change to a component. This is helpful if you accidentally delete or move a component. It is not possible to undo changes to components properties. You can undo a virtually unlimited number of times.

- **Redo** – Undo an undo. For example, if you delete a component, undo the delete and then select redo, the component will be deleted again. It is easier to use than it is to explain. Experiment with this feature to get a better understanding of how it works.

- **View Mode** – Close edit mode and view the Graphic Display normally.

Component Templates

Component templates are used to quickly set all of the properties of a component. Any visual component used in the form editor can use templates.

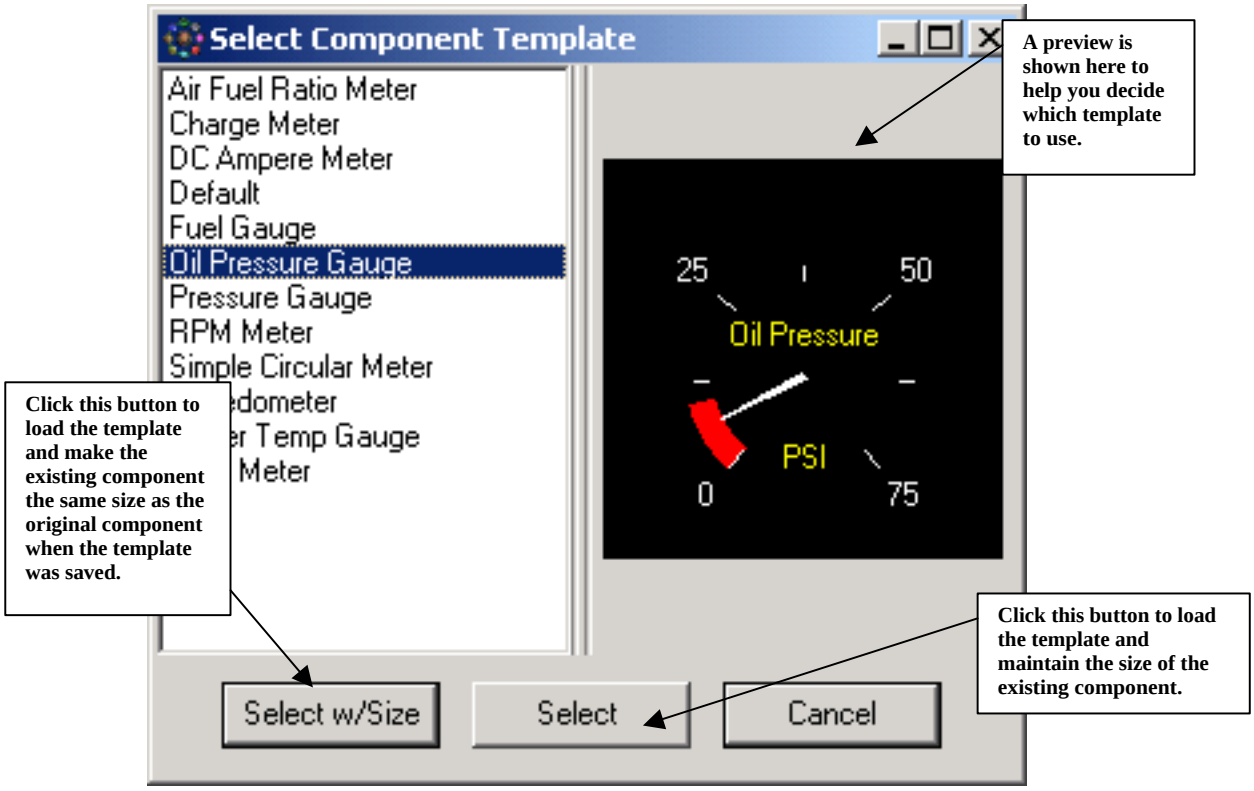
Saving Component Templates

- Set all of the components properties and test the component to be sure it works as desired.
- In edit mode, right click on the component and select “Save as Template…” from the “Component” menu.
- Type a filename for the template. The filename should describe the component, for example, “Air Pressure – Range 150 to 375”. The template will be saved on disk and can be applied to any new identical type component you place on the Graphic Display.

Loading Component Templates

When placing a new component on a Graphic Display you will automatically be asked to select a template if any templates for that type of component exist.

If you wish to apply a template to a component that is already on the Graphic Display, right click on the component and select “Load From Template…” from the “Component” menu. In either case, if any templates for that type of component exist you will see a form similar to the following:



To apply a template to the component, highlight the desired template name in the list to the left and click the “Select” or “Select w/Size” button.

Form Templates

Form templates are used to quickly create Graphic Displays that are identical or very similar to Graphic Displays that have already been designed. For example, in a building with eight floors with nearly identical floor plans, create a Graphic Display for the first floor, save it as a form template, and then use that template when creating the Graphic Displays for the other seven floors.

Saving Form Templates

- Design and test the Graphic Display to be sure it works as desired.
- In edit mode, right click on the form and select “Save Form as Template…” from the “Form” menu.
- Type a filename for the template. The filename should describe the form, for example, “4 Stage Rooftop unit”.

The template will be saved on disk and can be applied to any new Graphic Display you create.

Using Form Templates

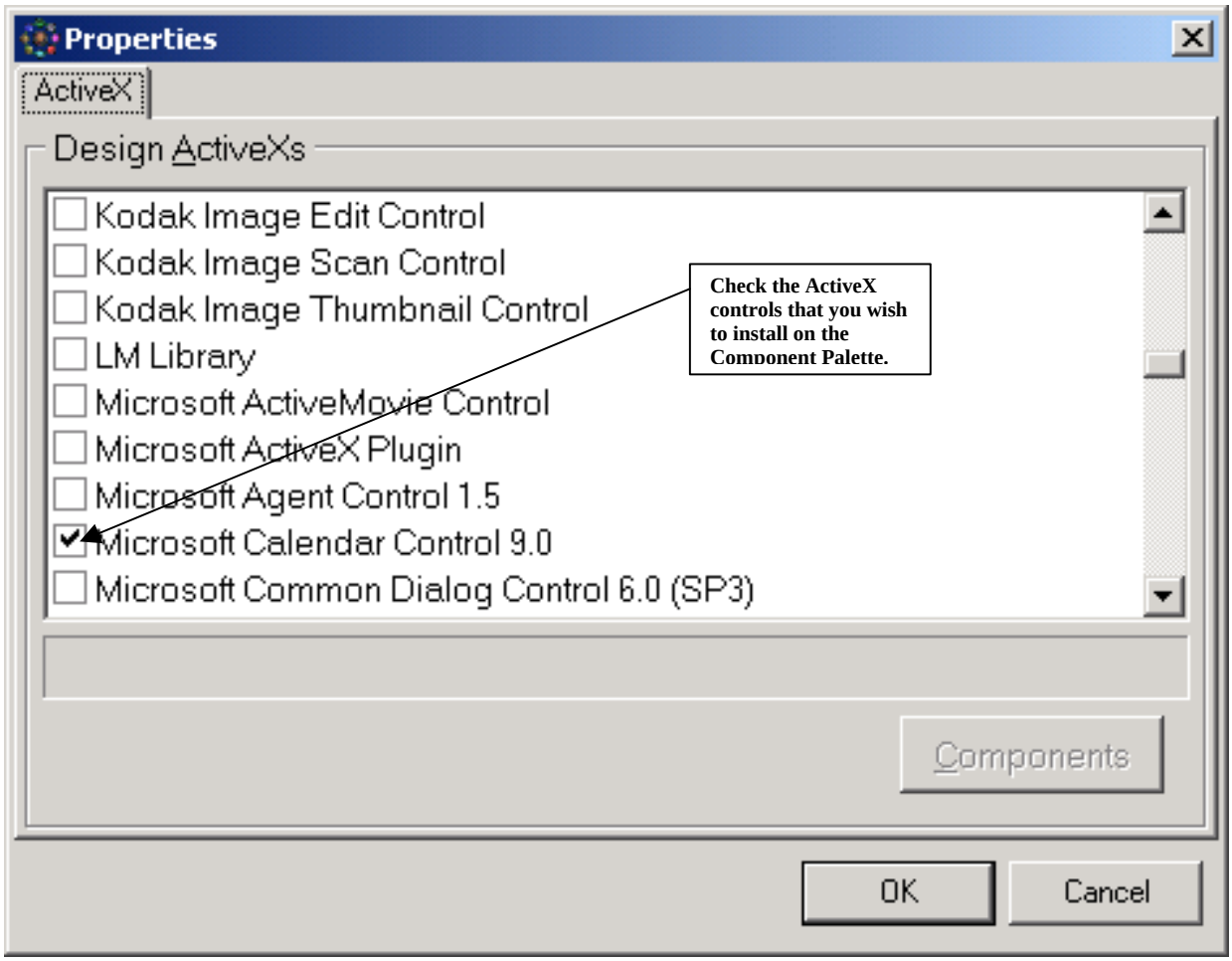
Form templates can only be applied to a Graphic Display when it is first created. See “Creating a New Display” for more information.

Once a new Graphic Display has been created using a form template, you may need to change some or all of the Control Points that are used on it. The quickest way to change multiple Control Points is to select “Edit Point Links” from the “Component” menu.

Using ActiveX Controls

✪ **IMPORTANT**: Not all ActiveX controls are designed to be used in this manner. Some even have code in them to stop them from being used like this. It is possible that some ActiveX controls will generate error messages when placed on a form. It is even possible that some will generate so many errors that it will not be possible to remove them from the form, thus making the form unusable. Make sure to test any ActiveX controls you want to use on a blank form before you place them on a good form.

To import ActiveX controls for use on Graphic Displays, right click on the Component Palette and select "ActiveX…" . and you should see a form similar to the following:



Check all of the ActiveX controls you want to use on Graphic Displays and click the "OK" button. The ActiveX controls should then appear on the "ActiveX" tab of the Component Palette.

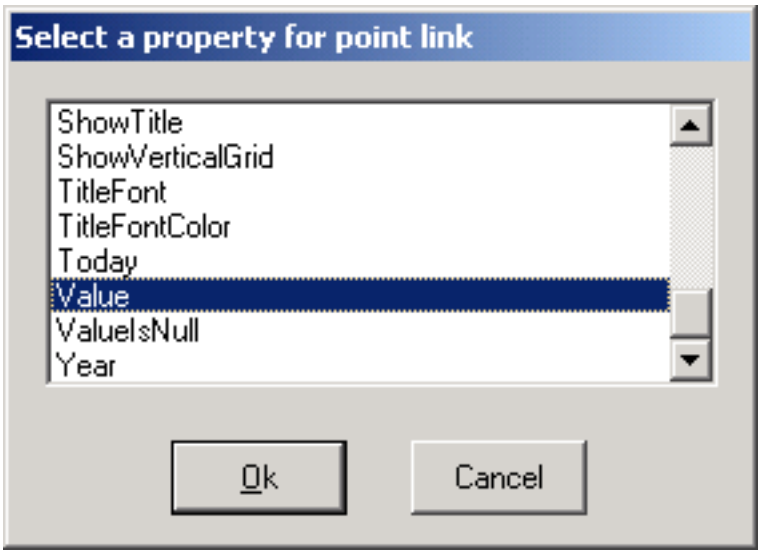
Linking Control Points to ActiveX Properties

Linking Control Points to an ActiveX control allows the ActiveX control to change its look or behavior based on the value of a Control Point in a panel.

Since there are literally thousands of ActiveX controls available it is impossible to automatically know what property of an ActiveX control needs to be linked to a Control Point. You must specify the property manually.

Once you have placed an ActiveX control on the Graphic Display, right click on it and select “Add Point Link…” from the “Component” menu.

You should see a form similar to the following:



This is a list of all the compatible properties of the ActiveX control. Select a property from the list and click the “OK” button.

You will then need to select a Control Point to assign to this property. Once you return to view mode, the value of the assigned Control Point will be read from the panel and the program will attempt to change the value of the property to match the value read from the panel.

While it is very simple to import ActiveX controls and add point links to their properties, it is not always obvious which property needs the point link for the control to operate properly. Feel free to experiment with different properties but until you have determined which property to use, it is best to do your testing on a blank form.

How do I …

The following are answers to common questions from our past products as well as some general tips on creating Graphic Displays.

How do I show a picture in the background of a Graphic Display?

- Select the “Image” component from the “Misc” tab on the Component Palette.
- Click anywhere on the Graphic Display to place the Image component.
- Click the “…” button of the “Picture” property in the object inspector.
- Click the “Open” button of the form that will appear (the button with the opened folder on it.)
- Browse to any folder on your computer, select a compatible picture file and click the “Open” button.
- Click the “OK” button.
- If you want the picture to stretch to fill the entire Graphic Display, check the “Stretch” property and set the “Align” property to “alClient”.
- If you would rather have the picture maintain its actual size, check the “AutoSize” property instead.

Once you have loaded a picture in to an Image component, it becomes part of the Graphic Display and is saved along with it. You do not need to keep a copy of the picture file.

How do I show a different Graphic Display when the user clicks on my Graphic Display?

The easiest way is to:

- Place an “ActionButton” from the “Misc” tab of the Component Palette on your Graphic Display.
- Click the “…” button of the “Action” property in the object inspector.
- Select “View a Graphic Display” from the form shown and click the “OK” button.
- Click the “…” button of the “ActionParameter” property in the object inspector.
- Select the display you want the button to open and click the “Open” button.

If you want to show a Graphic Display when the user clicks on an image, meter, or anything other than an ActionButton:

- Select the component that you want the user to click to show the Graphic Dislpay.
- On the object inspector, click the “Events” tab.
- Double click the “OnClick” event (Not all components have an “OnClick” event. The “OnMouseUp” event will also work.)
- The script editor should now be visible with the cursor flashing between the “Sub xxx” and “End Sub”
- Type the following:

  FormViewByPath "C:\Catalyst\Sites\[Your Site Name]\Displays\First Floor.Dis", ”", ""

  (Change the path, Site name and filename as needed. Notice the two pairs of double quotes at the end separated by commas. See Script Commands for more information.)
- Go to view mode and click on the component. If it doesn’t work or you get an error message, repeat the above steps and check your spelling.

How do I show the value of a Control Point as text?

- Place a "PointLabel" component from the "Misc" tab of the component palette on the Graphic Display.
- Select the Control Point and component template desired.
- Edit the "CaptionMask" property in the object inspector.
- Type any text you wish and use these codes to insert values in to the text:
  - @S for the site name
  - @P for the point description
  - @A for the point address
  - @T for the type
  - @C for the class
  - @U for the units
  - @V for the value
  - @R for carriage return

  For example:
  - "@P = @V @U" will display "Zone1 = 72.34 Deg. F". (This is the default.)
  - "Average Zone Temp is @V @U" will display "Average Zone Temp is 73.4 Deg. F"
  - "@S Economizer mode is @V" will display "Central College Economizer mode is ON".

How do I make text change color or flash if the Control Point value is out of range?

- Follow the above steps to place a "PointLabel" on the display and select "Blink if Out of Range" as the component template.
- Change the "AlarmLowLimit" and the "AlarmHighLimit" to define the allowable range.
- Change the "AlarmHighColor" and "AlarmLowColor" to change the color of the blinking text.

How do I put a link to a web site on a Graphic Display?

- Place a "URLLabel" component from the "Misc" tab of the component palette on the Graphic Display.
- Set the "Caption" property of the URLLabel component to any text you wish.
- Set the "URL" property of URLLabel component to the web site address (for example: www.MyWebSite.Com".)

✪ You can execute any program, play audio or view movies with this exact same procedure by typing in the full path and filename of any file type known by Windows instead of a web site address.

How do I add my own custom help files to a Graphic Display?

If you just wish to add explanatory text to a Graphic Display:

- Place a "Memo" component from the "WinStandard" tab of the component palette on the Graphic Display. This will also work with the "RichEdit" component on the "Win32" tab.
- Click the "…" button of the "Lines" property in the object inspector.
- Type or paste in the text or use the "Open" button to import text in to the Memo component.
- If the text is not all visible in the Memo component, change the ScrollBars property so the user can scroll through the text.
- Check the "ReadOnly" property in the object inspector.

If you want to add actual Windows Help files:

✪ You can execute any program, play audio or view movies with this exact same procedure by selecting a file of any type known by Windows instead of a Help file.

- Place an “ActionButton” from the “Misc” tab of the Component Palette on your Graphic Display.
- Click the “…” button of the “Action” property in the object inspector.
- Select “Custom Tool” from the form shown and click the “OK” button.
- Click the “…” button of the “ActionParameter” property in the object inspector.
- Select “All Files (*.*)” in the “Files of Type” box.
- Browse to any folder on your computer, select a Windows Help file and click the “Open” button.

Chapter 5 - Trends

Trends allow you to graphically view the historical values of inputs, outputs and variables. Trends can be configured to read historical data every time they are viewed or data can be collected automatically by using Scheduled Events. See “Event Scheduler” for more information on automatic data collection.

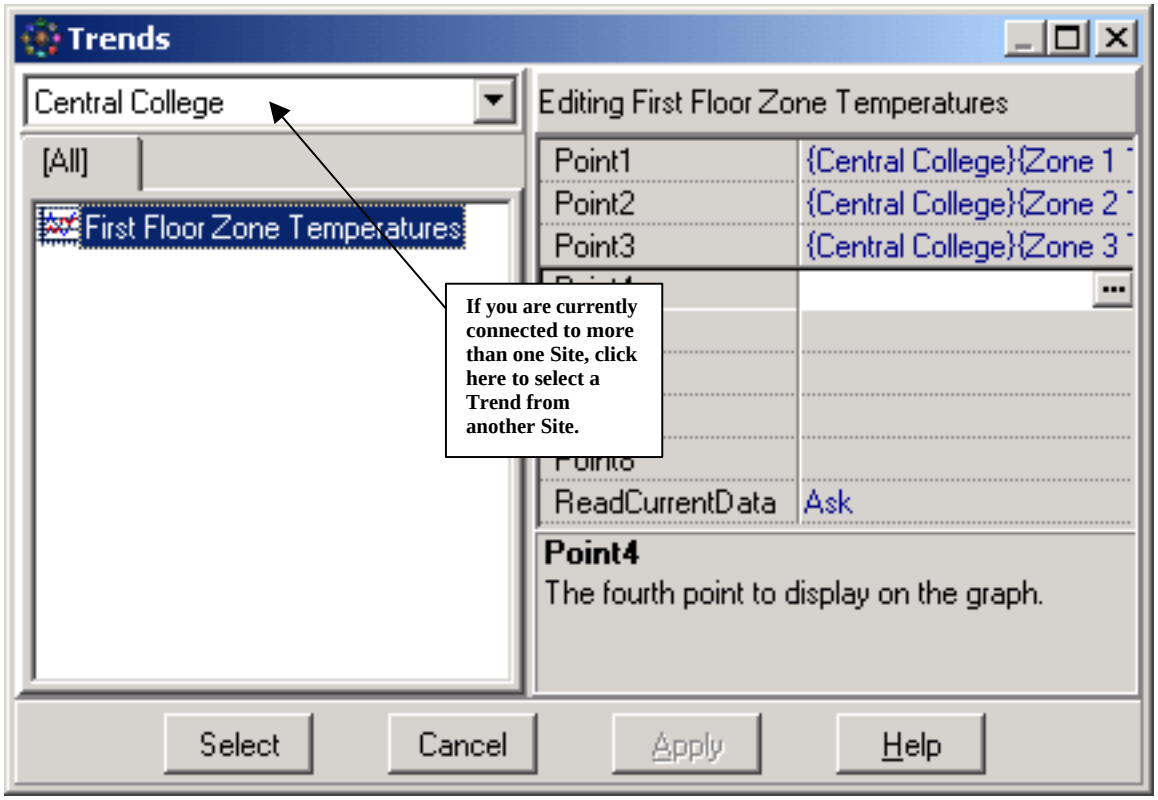
Each Trend will save a virtually unlimited number of historical values. Once the Trend is configured, every historical value ever read will be stored in the local Trend file. This makes it possible to view historical values months or even years after it has been collected. The actual number of values stored is limited only by available space on the local hard disk.

Editing the Trends Database

The Trends Database is where the list of Trends for each Site is stored. It is very similar to the other database screens used in this program. See “Editing the Site Database” for more information on editing the entries.

There is one important difference with the Trends Database compared to the Sites Database. There is a drop-down list above the tabs area that contains the list of all Sites you are currently connected to (In this example, "Central College" is the selected Site).

Each Site maintains it's own list of Trends so if you are connected to several Sites, use the drop-down list to view the Trends for each of them.



Creating a New Trend

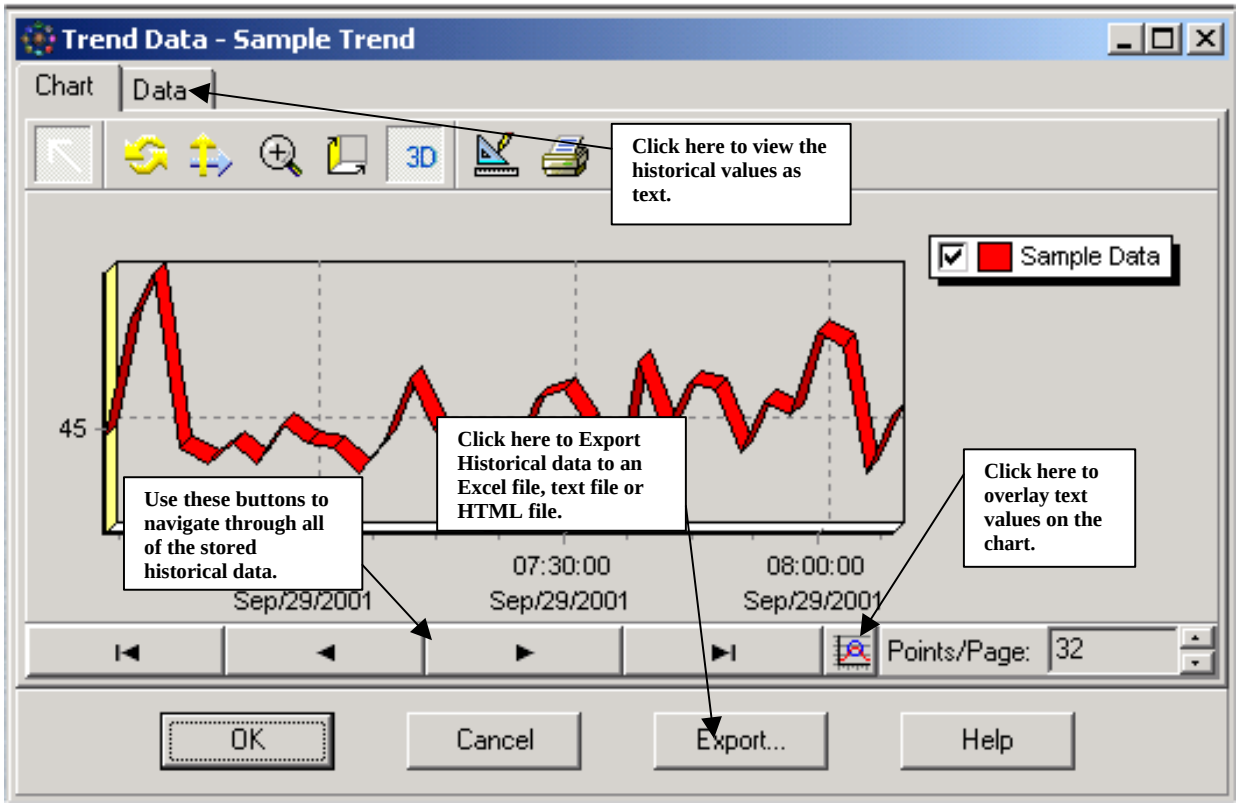
To create a new Trend:

- Right-click in the list area and select "Add Trend".
- Type a descriptive name for this Trend.
- Select up to eight Control Points to add to this trend by clicking the "…" button of the "Point1" - "Point8" options.
- Select the "ReadCurrentData" option and choose one of the following:
    - **Always** – Always read the most recent historical values from the Site when the Trend is viewed.
    - **Never** – Do not read the most recent historical values from the Site when the Trend is viewed.
    - **Ask** – Ask the user if they wish to read the most recent historical values from the Site when the Trend is viewed.

✪ Note that the eight Control Points do not have to be from the same Site. You do not need to be connected to the other Sites to view the historical data from them that is already saved on disk; however, you would need to be connected to a given Site to read the most recent historical values for those Control Points. See "Control Points" for more information on selecting Control Points from multiple Sites.

Viewing Trends

Highlight a Trend in the trends database and click the “Select” button. You should then see a form similar to the following:



The Trend data files will store a virtually unlimited amount of historical data. Because the amount of data can become quite large, the Trend form "pages" through the data, showing it in smaller chunks. The number of values displayed on the graph at one time is adjustable by changing the "Points/Page" value.

When navigating through the data using the four buttons below the graph, the "left" button will page backward in time by half the "Points/Page" value, and the "right" button will page forward in time by half of the "Points/Page" value. For example, if you are viewing the most recent 32 readings and "Points/Page" is set to 32, pressing "Left" will go back in time by 16 readings. The result being that the most recent 16 readings have scrolled off the chart to the right and you are now seeing the 32 readings just prior to that.

To see the numeric values of the data superimposed on the chart, press the "Values" button (small button to the left of "Points/Page".) Pressing it multiple times cycles through all visible points.

The chart itself has many options to change its look and operation. There are far too many to try to list them all here. Click the Edit button on the charts button bar to view the Edit Chart Form. Most of the options there have built-in help. Click the "?" icon in the upper-right of the form and then click on the item you want more information about. It should not be necessary to ever change the options unless you want to customize the chart. The default look and behavior should suit most needs.

Click the Data tab at the top of the form to see the current page of data in numerical format.

Click the Export button if you wish to save a screen shot of the chart, export the raw data values to a text file, Excel spreadsheet, or an HTML file or even email the data to someone.

Chapter 6 - Alarms

Overview

The alarm database holds all alarms that have been received by the program. The messages are stored in a central database for easy access.

There are three levels of messages:

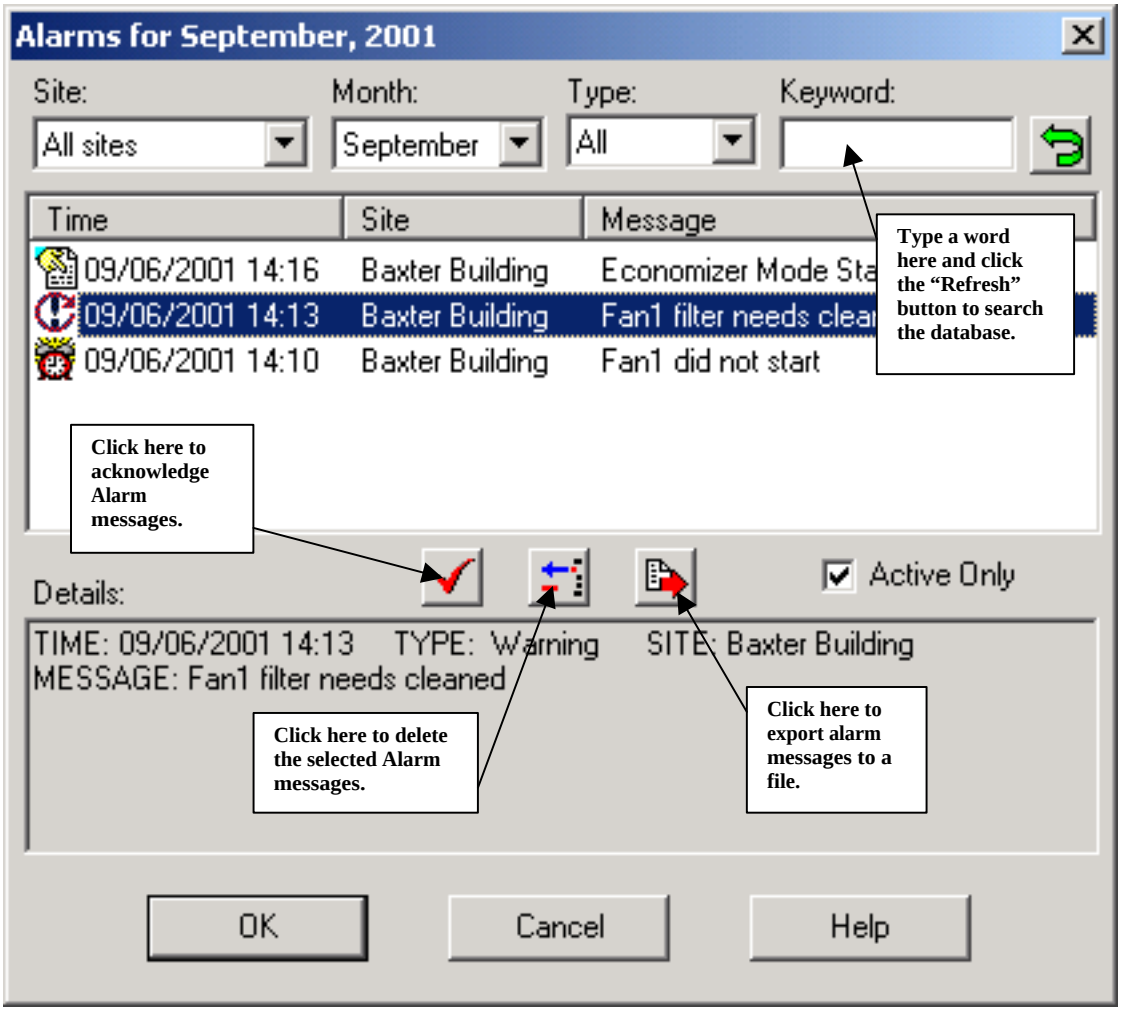
- **Alarm Messages** - Used for urgent messages such as equipment failures.
- **Warning Messages** - Used for important, but not urgent messages such as dirty filters.
- **Information Messages** - These generally don't require action by anyone, but are used to inform you about changes taking place such as seasonal changeovers.

All three types of messages are generically referred to as "Alarms".

✪ A panel at a given Site must generate Alarm messages. These messages are then intercepted and stored in the Alarm database. See the section of this manual specific to your type of equipment for more information on generating Alarm messages.

The Alarm Database

The alarm database holds all alarm messages received by the program. Alarms in the database can be sorted in several ways; by message type, time, Site or alphabetically. They can be filtered to show only active (un-acknowledged) messages, or only messages that contain a certain keyword or string of characters.



Acknowledging Alarms

To acknowledge alarms, select one or more and press the Acknowledge button. Hold down <Ctrl> to select multiple alarms or hold down <Shift> to select a range of alarms to acknowledge. The alarm is then permanently stamped with the time, date, username of the person that acknowledged it, and an optional comment entered by the user.

Deleting Alarms

Alarms that have been resolved may optionally be deleted. Since you can filter out non-active alarms, it is not necessary to delete alarms unless you just wish to remove the clutter or you are receiving an unusually high number of alarms (thousands per month). There is no hard-set limit on the number of alarms that can be stored or viewed but thousands of monthly alarms can make the alarm list scroll rather slowly on slower computers.

To delete alarms, select one or more and press the Delete button. Hold down <Ctrl> to select multiple alarms or hold down <Shift> to select a range of alarms to delete.

✪ Deleting alarms permanently deletes them from the database. There is no way to retrieve them once they have been deleted.

Exporting Alarms

All alarms in the current list may be exported to a Microsoft Excel Compatible file by pressing the Export button.

Chapter 7 – Event Scheduler

Overview

Scheduled Events provide a way to backup a Sites memory to disk, collect historical trend data or execute a Custom Form at a specified date and time without user interaction.

For a Scheduled Event to properly execute, a few requirements must be met:

- This software must be running on a computer that has access to the specified Site.
- While events can be executed for a given Site when connected to a different Site, if both Sites use the same COM port or modem, the event will fail. The software will not automatically disconnect from any Site to execute an event for another Site.
- If the Site is accessed via COM port, the port must not be in use by any other program such as an Internet connection or terminal program.

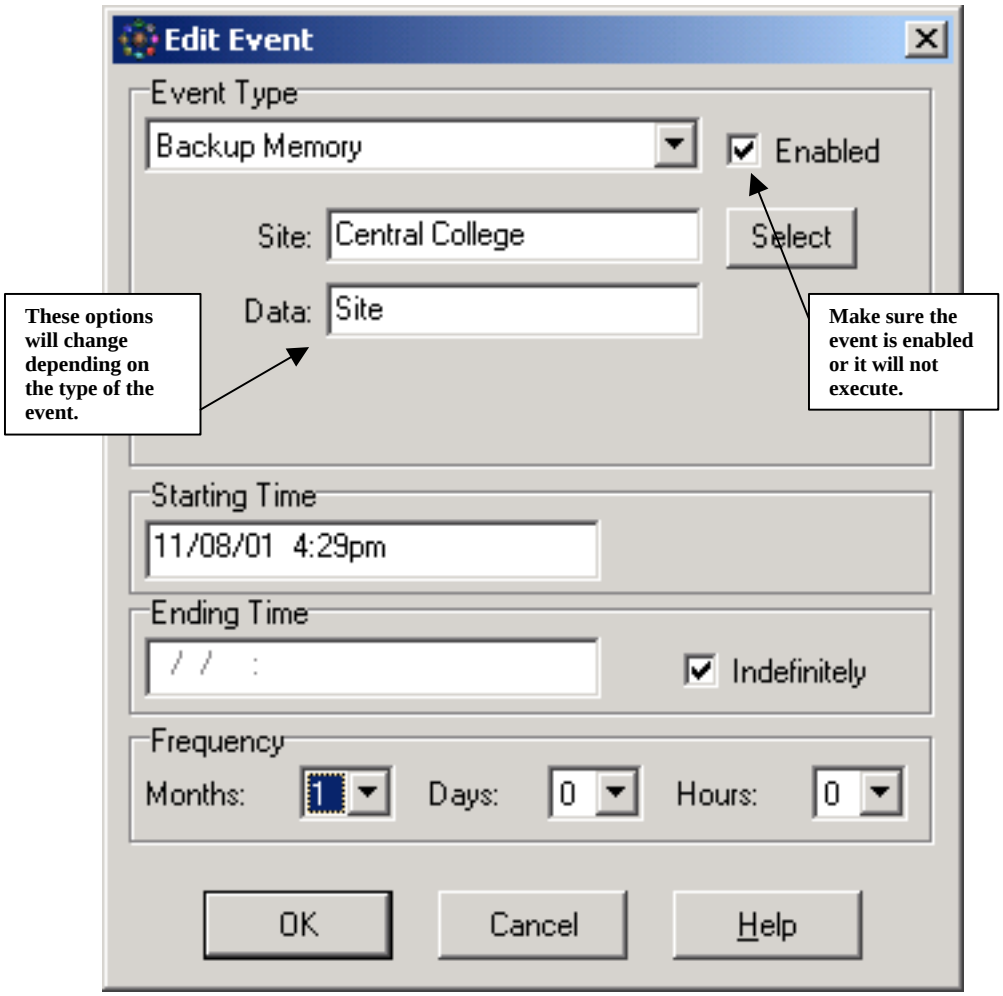
When it is time for an event to execute, if the software is already connected to the proper Site, the event will just execute and nothing else will be done. Otherwise, the software will connect to the proper Site, the event will be executed and then the software will disconnect from the Site. In either case, any errors that occur during the process will be stored in the Alarm database as Warning messages and the Alarm panel on the status bar will flash to alert you to the problem.

✪ While every effort has been made to make the events execute as smoothly as possible when already connected to other Sites and/or when Graphic Displays are being viewed, there can be rare circumstances or certain forms that may cause the event to fail or otherwise interfere with the process. In 95% of the cases the events will work fine but to ensure maximum reliability it is best to not leave unneeded Graphic Displays or forms open if you know an event is pending that day.

Editing a Scheduled Event

To access the Scheduled Events database, select “Program Options” from the “File|Setup” menu. Click the “…” button of the Scheduled Events options.

See “Editing the Site Database” for more information about editing database entries.



There are three types of Scheduled Events: Backup Memory, Collect Trend Data and View Form.

Backup Memory

This event will automatically backup a Sites memory to disk. Two parameters are required for this type of event:

- **Site** - This is the Site you wish to backup.
- **Data** - This is the data from the Site you wish to backup and will vary based on the type of hardware the Site contains. See “Backup Memory” in the section of this manual specific to your hardware for more information.

The backup files are automatically named with the format MMMDD-YYYY.Dmp (for example Oct13-2001.Dmp). If a file with that name already exists, a "(2)" will be appended to the filename. Up to "(100)" files may be saved for each Site every day. The files are stored in the \Sites\[SiteName]\Backups folder.

Collect Trend Data

This event will automatically collect historical trend data from a single Site. Two parameters are required for this type of event:

- **Site** - This is the Site from which you wish to collect the data.
- **Trends** - One or more trend entries from the Trend Database that you wish to collect. Please note that some trends may have points from multiple Sites. Since each event can only connect to one Site at a time, multiple events will be needed to collect all the data for those trends. If a trend only contains points from a single Site, only one event is needed.

Trend data collected by events is stored in the existing trend point data files and can be viewed at any time by selecting the trend from the Trend Database.

View Form

This event will automatically display a Custom Form. The Custom Forms can be designed to do virtually anything that you may need to do but may require some script programming. Three parameters are available for this type of event:

- **Site** - This is the Site you wish to connect to before viewing the form (optional).
- **Form** - The form you wish to be displayed. This can be any type of Custom Form (required).
- **Parameters** - This is a list of optional parameters to pass to the form. It can be a string of characters such as CHILLER REPORT or a series of parameters enclosed in brackets such as [ZONETEMP][PANEL1][14.25][TRUE]. Multiple parameters are passed to the form as a variant array. See the “FormParametersGet” script command in the manual for information on how to access these parameters in a Custom Form.

✪ Any form displayed by a Scheduled Event will be displayed as a Windows "modal" form. This means that the form will take the focus away from any other forms that may be open and the software will do nothing else until the form is closed, including disconnecting from the specified Site. If the script code within the Custom Form does not automatically close the form, the form will stay open and the software will remain connected to the Site until someone manually closes the form.

The remaining options specify when the event should execute:

- **Enabled** - This box must be checked to enable the event to execute. Uncheck this box to temporarily disable the event from executing.
- **Starting Time** - The date and time for the event to execute for the first time.
- **Ending Time** - The date and time for the event to stop executing.
- **Indefinitely** - Check this box to have the event execute continually until it is deleted or manually disabled.

- **Months** - The number of months to wait between executions of the event. Note that this is "months" and not thirty day periods. If the starting time is set to January 2nd and months is set to "1", the event will execute on the 2nd of each month.

  If the day of month of the starting time is greater than the number of days in the current month, the event will execute on the first day of the following month. For example, if the starting time is set to January 30th and months is set to "1", the event will execute on January 30th, then again on March 1st because there is no February 30th. The event would then continue to execute on March 30th, April 30th, etc.

- **Days** - The number of days to wait between executions of the event.

- **Hours** - The number of hours to wait between executions of the event.

It is not possible to execute a single event more often than once an hour. If you need timing of less than one hour, use multiple events.

Chapter 8 – Schedules

Viewing Schedules

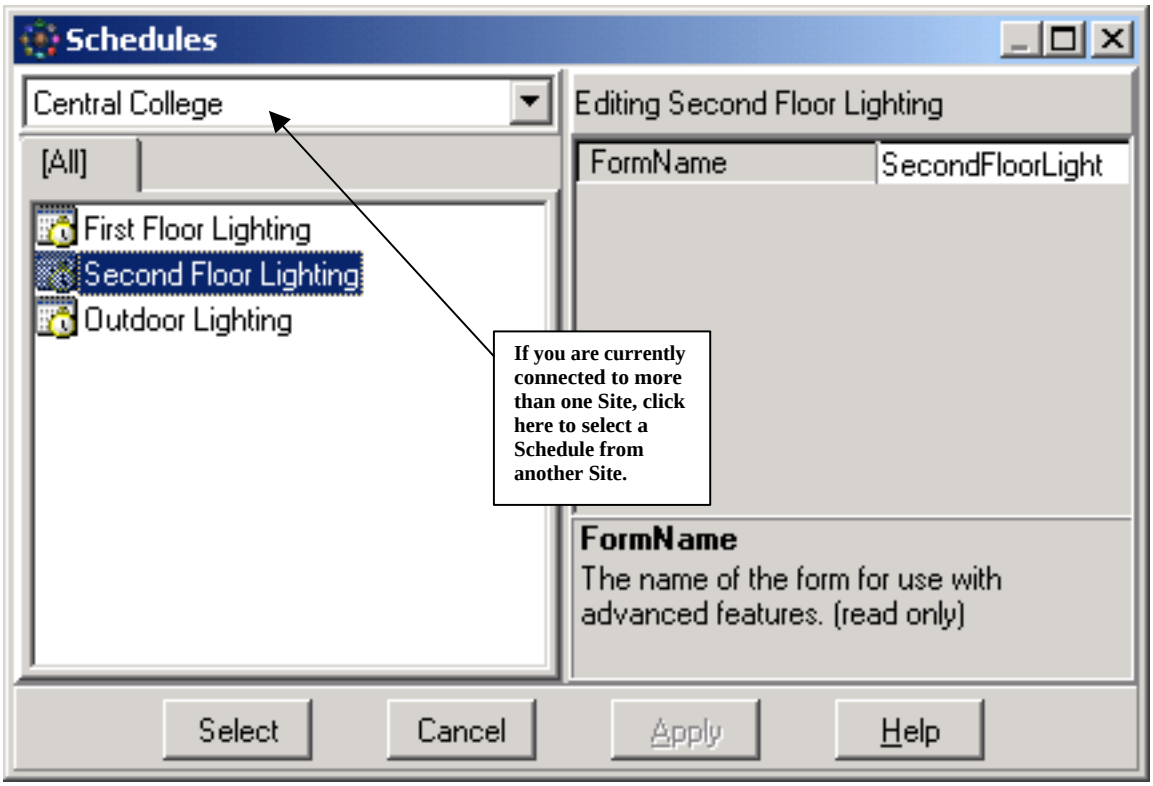
To view an existing Schedule, select “Schedules” from the “View” menu. Highlight any of the Schedules in the database and click the “Select” button. Because scheduling varies widely between different types of hardware, the built-in Schedules are customized to each of them. See “Schedules” in the section of this manual specific to your hardware for more information.

Editing the Schedule Database

The Schedule Database is where the list of Schedules for each Site is stored. It is very similar to the other database screens used in this program. See “Editing the Site Database” for more information on editing the entries.

There is one important difference with the Schedules Database compared to the Sites Database. There is a drop-down list above the tabs area that contains the list of all Sites you are currently connected to (In this example, "Central College" is the selected Site).

Each Site maintains it's own list of Schedules so if you are connected to several Sites, use the drop-down list to view the Schedules for each of them.



Creating a New Schedule

To create a new Schedule, right-click in the list area and select "Add Schedule”.

The first thing you will need to do is select the type of Schedule you want. Some type of hardware will have many schedule types; others may only have one or two. See “Schedules” in the section of this manual specific to your hardware for more information.

After selecting the Schedule type, enter a description for the Schedule. This is the description that will appear in the Schedule database, and also on the caption bar of the Schedule itself.

Next you will be asked for a unique name for the Schedule. This cannot be the same name as any other Display, Form, Schedule, etc. that is used for this Site (the program will let you know if the name is already in use). You can usually just accept the default name it suggests.

Your Schedule should now be created and added to the database. Highlight your new Schedule and click the "Select" button to view it.

Chapter 9 – Backup and Restore Memory

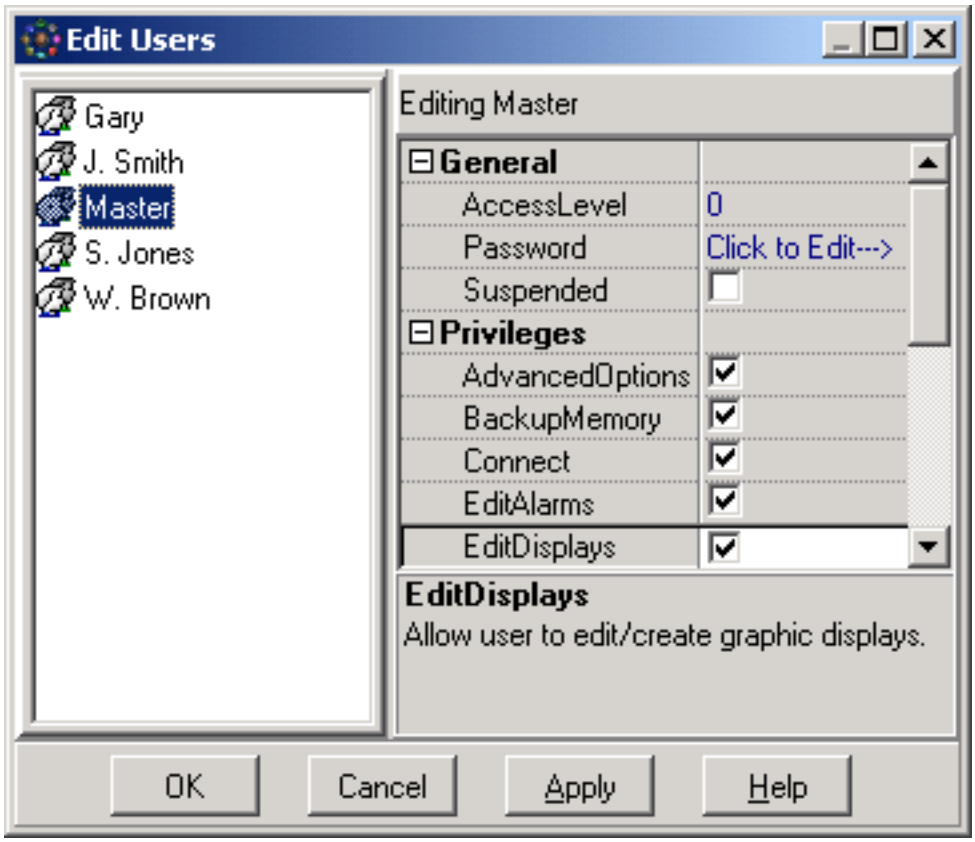
Because the format for backup and restore of a Sites memory varies widely between different types of hardware, these functions are customized to each of them. See “Backup Memory” and “Restore Memory” in the section of this manual specific to your hardware for more information.

Chapter 10 – Users and Security

The User Database is where the system administrator can create, delete and customize the access level of all users.

✪ At least one user must always have the "Edit Users" option enabled. If no users are allowed to edit the User Database you will never be able to make any changes to the database.

See “Editing the Site Database” for more information about editing database entries.



To create a new User, right click in the list area and select "Add User”.

Each User has the following access options:

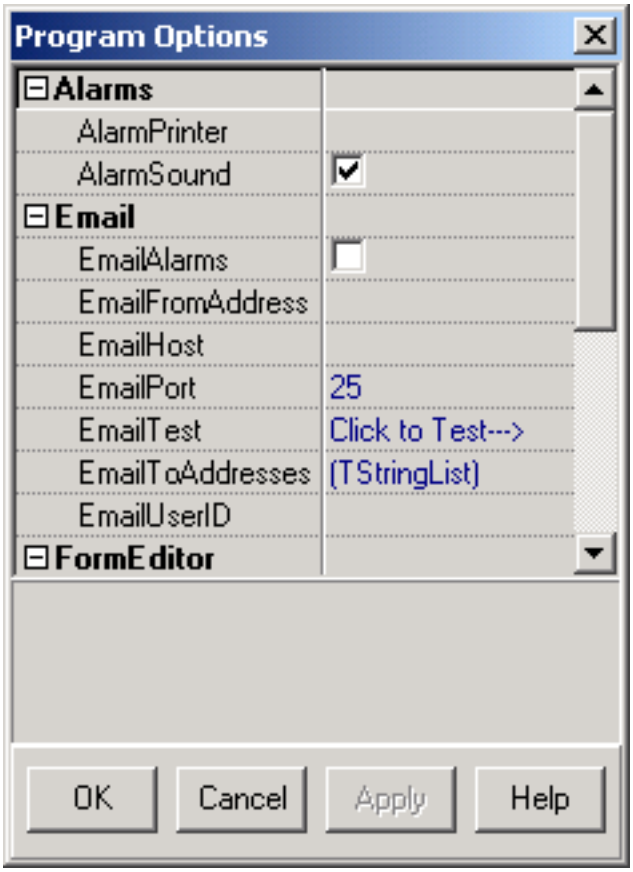
- **AccessLevel** - Optional access level (0-255) used by some custom functions. This value can be checked by Custom Forms to allow/disallow some users from using the form. This can normally be left at zero unless you have a reason to change it.
- **AdvancedOptions** - Allow user to edit advanced options such as Main Form Properties.
- **BackupMemory** - Allow user to backup a sites memory to disk.

- **Connect** - Allow user to connect/disconnect from sites.
- **EditAlarms -** Allow user to edit the alarm database. This option will allow the user to delete and acknowledge alarms.
- **EditDisplays** - Allow user to edit/create Graphic Displays. Keep in mind that it is possible to work around some of the security options and password level limitations using script code in Graphic Displays.
- **EditForms** - Allow user to edit/create Custom Forms. Keep in mind that it is possible to work around some of the security options and password level limitations using script code in Custom Forms.
- **EditPoints** - Allow user to edit the Control Point database.
- **EditSchedules** - Allow user to edit/create Schedule forms. Note that this is not the same as changing the times and dates on a Schedule form (see below).
- **EditSites** - Allow user to edit the Site database.
- **EditTrends** - Allow user to edit/create historical trend forms.
- **EditUsers** - Allow user to edit the user database. **At least one user must always have this privilege.**
- **ExitProgram** - Allow user to exit and close the program. If disabled, the user will only be able to log out of the program and return to the password screen.
- **ModifySchedules** - Allow user to modify schedule times and dates.
- **ModifySetpoints** - Allow user to modify/enable/disable setpoints.
- **Password** - The password for this user to log in to this program.
- **RestoreMemory** - Allow user to restore a sites memory from disk.
- **Suspended** - Do not allow this user access to the program. This is useful to temporarily keep a user from accessing the program without having to delete them from the database.

Chapter 11 – Program Options and Customization

Program Options

These options affect the overall operation of the software. Access these options from the File|Setup menu. To access these options, select “Program Options” from the “File|Setup” menu.



- **ActivityLogging** - Record important user activity in log files.
- **AlarmPrinter** - The name of the printer to use for printing incoming alarms. Please note that if a laser or inkjet printer is used, the printer itself will automatically eject the page within a few seconds of the alarm printing. This results in the alarms printing one per page. For this reason, a dot-matrix printer is recommended for alarm printing.
- **AlarmSound** - Play a repeating sound when there are active alarms. The alarm sound stops when the alarm database is viewed or you can right-click on the Alarm Panel on the status bar and select "Silence Alarm Sound". Turn this option off to disable the alarm sound.
- **AskForPoints** - Automatically ask for Control Points when a component is placed on a form while designing Graphic Displays.

- **AskForTemplate** - Automatically ask for a template when a component is placed on a form while designing Graphic Displays.

- **ConnectToSite** -Specifies a Site to connect to at program startup.

- **DisplayForm -** Specifies a form to display at program startup.

- **EmailAlarms** - Send an Email for every alarm received. An Internet connection must be maintained or be available at all times to use this option.

- **EmailFromAddress** - The "Reply To" address for sending Email messages. Your email provider requires this before it will allow email to be sent. It will usually be your email address.

- **EmailHost** - The name or IP address of the outgoing mail server, usually in the "smtp.myserver.com" format (required).

- **EmailPort** - The port number to use to send Email, usually 25 but could possibly be different. Contact you service provider if you are unsure.

- **EmailTest** - Click to test Email connection. Any changes that you have made to the other email options must be applied before testing (click the "Apply" button). A sample email message will be sent to the addresses provide in the EmailToAddress list.

- **EmailToAddresses** - The list of email addresses to which alarm messages will be sent (required).

  The format is a simple list of email addresses:

  joe@aol.com
  manager@mycompany.net
  pager@pagenet.com

- **EmailUserID** - The User ID your email host may require before sending email (not always needed). This would normally be the user name on the email account.

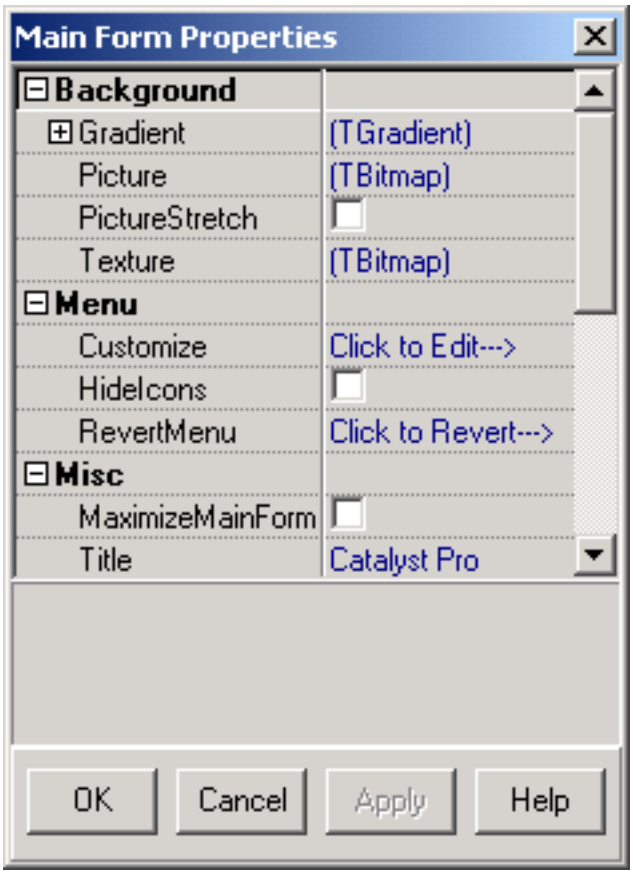
- **MainPrinter** - The name of the printer to use for printing forms and data.

- **Scheduled Events** - Scheduled Events provide a way to backup a Sites memory to disk, collect historical trend data or execute a Custom Form at a specified date and time without user interaction. Click the "..." to create or edit Scheduled Events.

- **ShowCategories** - Organize component properties by category in the form designer. If selected, the most common properties will be displayed in a "Main" group and the more advanced properties in the "Other" group. If not selected, all properties will be listed alphabetically.

Main Form Properties

The following options affect the overall look and feel of the software. These options are accessed by right clicking on the background of the main form or selecting “Main Form Properties” from the “File|Setup” menu.



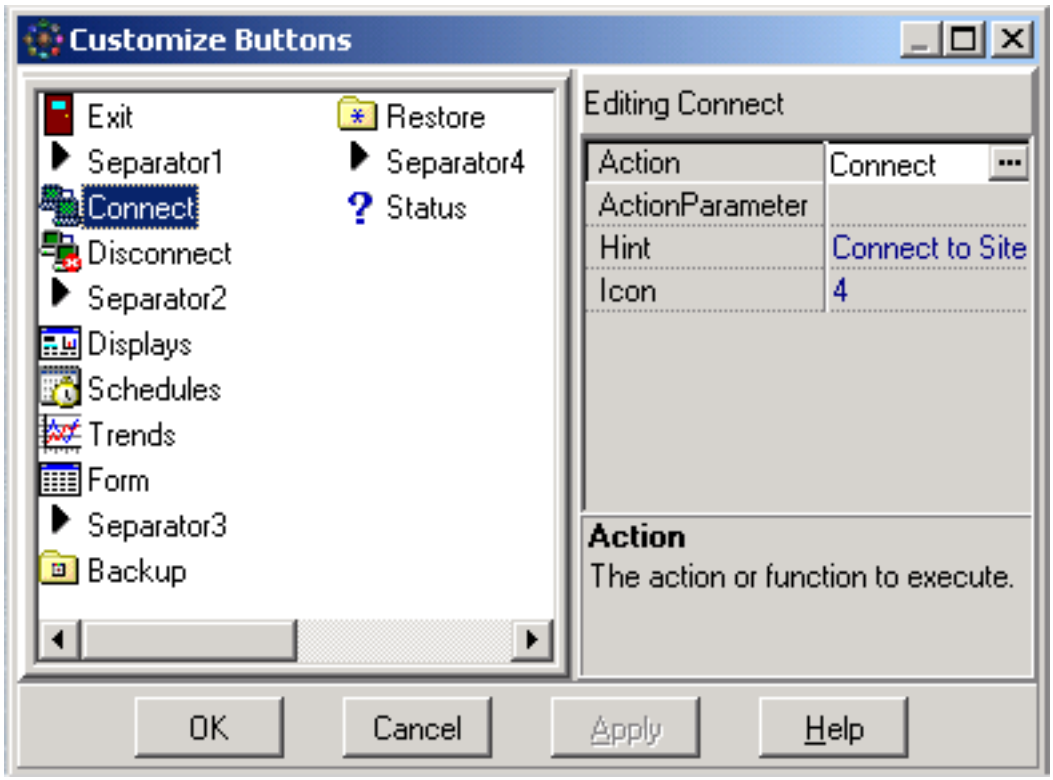
- **Gradient** - Gradient pattern for background. On slower computers or computers with older video cards, this could slow down the scrolling and movement of windows displayed within the software.
- **HideAlarms** - Do not display the "Alarms" panel.
- **Hide Icons** - Hide the icons in the main menu.
- **HideMessage** - Do not display the "Message" panel. The message panel displays the last alarm received and other important information.
- **HideSite** - Do not display the "Site" selection box. The Site selection box is necessary to switch between Sites when connected to more than one at the same time.
- **HideStatusBar** - Do not display the status bar. The status bar contains important information and you will normally want it to be visible.

- **HideTime** - Do not display the "Time" panel.
- **HideTitle** - Do not display the "Title" panel. The title panel shows the name of the program and the current version.
- **HideToolbar** - Do not display the toolbar on the main form.
- **MaximizeMainForm** - Always maximize the main form when the program is executed.
- **Menu/Customize** - Customize the Main Menu. This allows you to rearrange, add items or delete items from the main menu. This is an advanced feature for advanced users.
- **Menu/Revert** - This is used to reload the main menu with the default settings. Use this if you decide you do not like the changes you made to the main menu.
- **Picture** - .BMP file to display on the background of the main form. On slower computers or computers with older video cards, this could slow down the scrolling and movement of windows displayed within the software.
- **PictureStretch** - Stretch background picture to fill the entire main form, rather than center it.
- **Texture** - .BMP file to use as background texture in main form (overrides gradient). The .Bmp file will be "tiled" like the background texture of the Windows Desktop.
- **Title** - The name of the program.
- **ToolbarBitmap** - A .BMP file to display in the background of the toolbar.
- **ToolbarColor** - Color of toolbar background (double-click for custom color). Use this to specify a solid color for the toolbar.

Customizing Buttons

This feature allows you to customize the buttons on each button bar. Access this feature by right clicking on the button bar you wish to change.

New button bars may be added to the status bar at the bottom of the main form as well as to the toolbar at the top of the main form.



Right click on a button in the list to rename or delete it. To create a new button, right click in the white area of the list and select "Add Button". The order of the buttons may be changed by dragging the buttons with the mouse and dropping them in the desired location.

The options on the right side are as follows:

- **Action** - Select a pre-defined action that is built-in to the software. Custom functions may be added by selecting "View Form" as the action. A form must then be created to actually perform the function.

  An external file may be executed by selecting "Custom Tool" for the action. The ActionParameter can be set to any file type Windows understands. For instance, selecting an .Exe file will run the program, selecting a .Txt file will load the file in to your default text editor and a web site address will automatically run your web browser.

- **ActionParameter** - The meaning of the ActionParameter changes depending on the type of action. For many actions, no parameter is required. For others, like "View Form", the parameter should be set to the file path and name of the form to view. If no parameter is supplied, the Form Database will be displayed so the user may pick a form to view. Similarly, for "Connect", you can set the ActionParameter to the Site you wish to connect with. If the ActionParameter is left blank, the Site Database will be displayed.

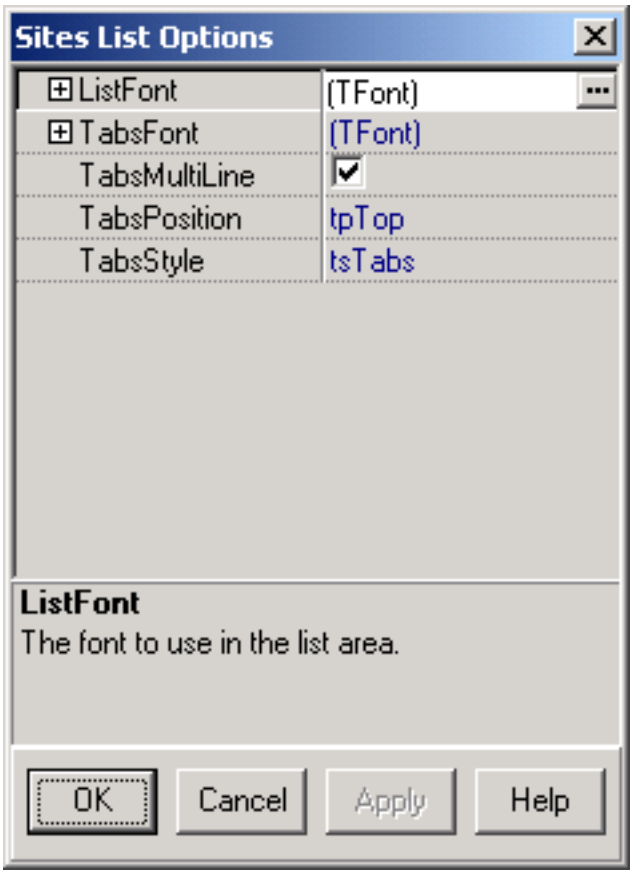
  Always select the Action first, then the ActionParameter. Once the Action has been selected, the program will know what type of parameter to ask you for.

- **Hint** - This is the text that will be displayed to the user if they pause the mouse cursor over the button for a few seconds.

- **Icon** - This allows you to change the icon that is displayed on the button. New icons may also be imported from .Bmp files using this option.

Database Options

These options allow you to change the look of the individual databases. To access these options, right-click in the list area of the database you wish to customize.



- **ListFont** - The font to use in the list area.
- **TabsFont** - The font to use for the tabs.
- **TabsMultiLine** - Display tabs on multiple lines if there are more tabs than will fit in the tabs area. If left unchecked, the user will be able to scroll through the list of tabs.
- **TabsPosition** - The position of the tabs. Some tab positions are incompatible with some tab styles.
- **TabsStyle** - The appearance of the tabs.

Panel Properties

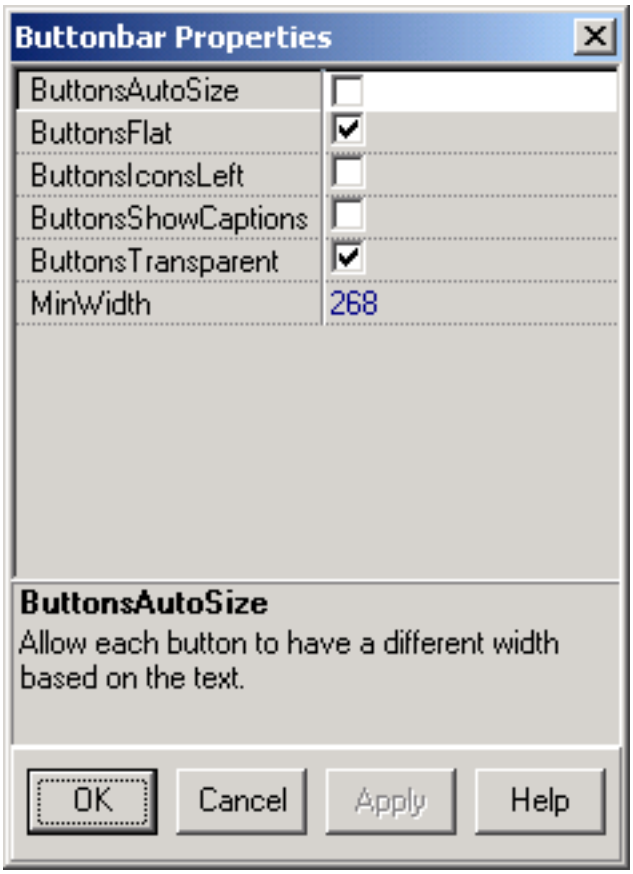
This feature allows you to customize the various panels on the toolbars (Site selection box, Alarms panel, etc.). Access this feature by right-click on the panel you wish to change.

- **Hide** - Hides (removes) the panel from the toolbar. It can only be restored to the toolbar by using the Main Form Properties.

- **MinWidth** - Minimum width of panel. Click "..." to set the minimum width to the current width of the panel. Setting the minimum width of the panels will keep them from obscuring each other when the main form is resized.

Buttonbar Properties

Use this feature to change the look of a buttonbar. Buttonbars may be added to either the toolbar at the top of the main form or to the status bar at the bottom of the main form. To access these options, right click on the buttonbar you wish to change and select "Properties".



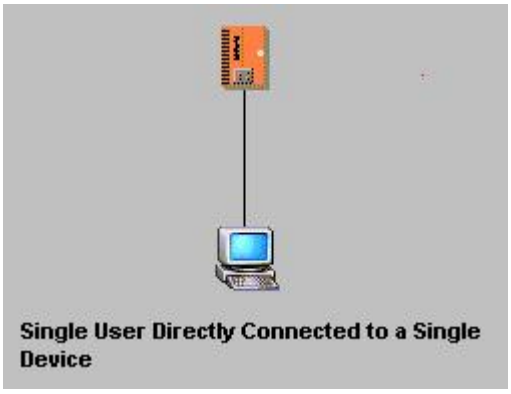
- **ButtonsAutoSize** - Allow each button to have a different width based on the text.
- **ButtonsFlat** - Give buttons a flat appearance rather than a 3D raised appearance.
- **ButtonsIconsLeft** - Show icons to the left of the text rather than above the text.
- **ButtonsShowCaptions** - Show the text on the buttons.
- **ButtonsTransparent** - Allow toolbar to show through flat buttons.
- **MinWidth** - Minimum width of buttonbar. Click "..." to set the minimum width to the current width of the button bar. Setting the minimum width of the buttonbars will keep them from obscuring each other if the main form is resized.

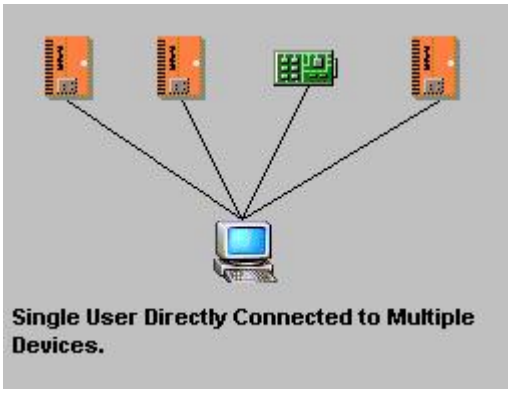
Chapter 12 – Networking and Remote Access

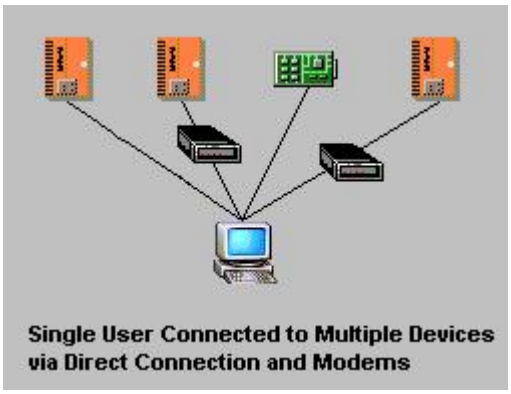
Overview

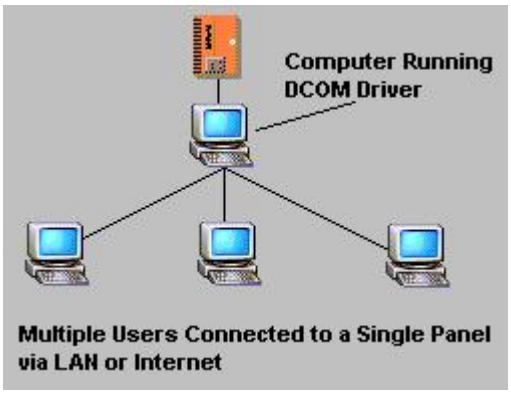
Catalyst Pro uses DCOM (OLE) drivers to communicate with the hardware. The drivers are separate programs that run independently of Catalyst Pro so they can be executed from a different computer, over a LAN or from a computer on the other side of the world via the Internet.

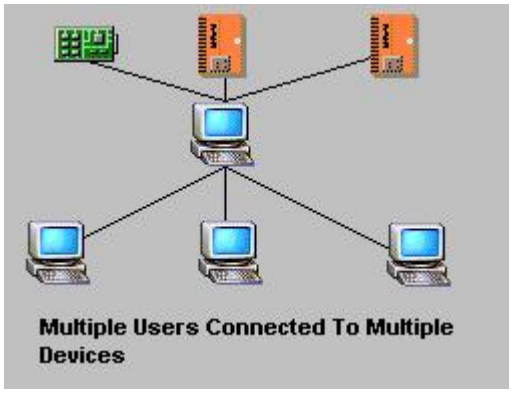
Below are a few of the possible configurations available.











Accessing Multiple Sites

Catalyst Pro has the built-in ability to communicate with multiple devices simultaneously. Graphic screens can be designed to display points from all systems and all other functionality is completely compatible with multiple connections.

No special configuration is needed to access multiple Sites simultaneously. As long as each Site is accessible by a given computer, you can connect to all of them at once assuming they all use different COM ports or are accessed via a network. See “Site Groups” in chapter 2 for more information.

Remote Access

Remote access allows you to connect to a Site through another computer that physically connects to the Site. It also allows multiple users to access the same Site simultaneously.

Example #1 – Remote Access via the local area network:

- Computer “A” is a computer at a Site running Catalyst Pro. It is connected to the panels via an RS-232 cable or modem. Computer “A” has been assigned the network name of “AutomationSystem” in the Windows networking options.

- Computer “A” can fully access the panels at any time using Catalyst Pro. The users of computer “A” can connect, disconnect, view displays and anything else they need to do without regard to remote access.

- Computer “B” is a computer three floors away that has a LAN connection to Computer “A”.

- To access the Site from computer “B”, create a new Site on computer “B”. The COM port and baud rate settings of this new Site on computer “B” must be set to reflect the COM port and baud rate used by computer “A” to access the Site. In other words, set up the Site on computer “B” exactly as you did on computer “A”.

- In the “RemoteName” option of the new Site on computer “B” put the network name of computer “A”, in this case it is “AutomationSystem”.

Computer “B” can now connect to the Site though computer “A”. The DCOM driver is executed on computer “A” every time computer “B” accesses the Site. If someone is currently accessing the Site on computer “A”, the Catalyst Pro driver will still allow computer “B” to connect and it will synchronize access to the panel between the two computers. Users at both computers can now communicate with the Site, view displays, change Schedules, etc. and disconnect when done.

Example #2 – Remote Access via the Internet:

- Computer “A” is a computer at a Site running Catalyst Pro. It is connected to the panels via an RS-232 cable or modem. Computer “A” has a full-time Internet connection and it is located at the IP address 111.222.333.444.

- Computer “A” can fully access the panels at any time using Catalyst Pro. The users of computer “A” can connect, disconnect, view displays and anything else they need to do without regard to remote access.

- Computer “B” is a computer in another state and has an Internet connection. It does not have to be a full-time connection.

- To access the Site from computer “B”, create a new Site on computer “B”. The COM port and baud rate settings of this new Site on computer “B” must be set to reflect the COM port and baud rate used by computer “A” to access the Site. In other words, set up the Site on computer “B” exactly as you did on computer “A”.

- In the “RemoteName” option of the new Site on computer “B” put the IP address of computer “A”, in this case it is “111.222.333.444”.

Computer “B” can now connect to the Site though computer “A”. The DCOM driver is executed on computer “A” every time computer “B” accesses the Site. If someone is currently accessing the Site on computer “A”, the Catalyst Pro driver will still allow computer “B” to connect and it will synchronize access to the panel between the two computers. Users at both computers can now communicate with the Site, view displays, change Schedules, etc. and disconnect when done.

✪ There is no limit to the number of users that can connect at one time. However, the more simultaneous users there are, the longer it will take each of them to receive data from the panel. This is more pronounced with a COM port connection to the panels.

✪ The above examples assume both computers are using Windows 2000/NT. Windows 95/98 will not automatically execute a DCOM driver. If you are using Windows 95/98, the driver must already be running before the remote computer can access the Site. Just execute the driver from Windows Explorer and leave it running.

✪ Windows 2000/NT must be properly configured to allow remote computers to execute DCOM drivers. On a LAN this is normally not a problem. For remote Internet access, see your Windows documentation or System Administrator for configuration information.

✪ Catalyst Pro does not do any type of file synchronization. The Graphic Displays, Schedule forms and Trend setup files must be manually copied to each computer that will access the Site. All Schedule times and dates are kept on the panels themselves so all users will have access to the current values, but the original files must be on each computer. In a LAN environment, each workstation may also execute Catalyst Pro from a main server or other workstation, thereby utilizing one set of files.

Chapter 13 – Custom Forms

Overview

Custom Forms allow you to add your own features and functionality to Catalyst Pro. Many of the forms that are used in Catalyst Pro itself are actually Custom Forms (the Backup and Restore memory, Control Point import, and Schedule forms for example.)

Once a Custom Form has been written, you can add a button to the toolbar and a menu selection to the main menu to execute it and it will be indistinguishable from a built-in function.

✪ Custom forms are an advanced topic. They can be used for simple functions that only require a few lines of script code, but can also get quite complex for more useful functions. In any case, there will definitely be script programming involved. Because a complete tutorial on programming, form use and Windows common controls would take several hundred pages, a few assumptions are made in this manual.

It is assumed that:

- You have read and understand most of this manual (especially chapters 1-4).
- You have at least a basic understanding of programming in general.
- You have access to a Visual Basic Scripting reference manual or the free help files available from Microsoft (or a Borland Delphi manual, see below.)
- You are comfortable working with Catalyst Pro and know your way around in it. To save time and space, we will say, for example, “Set the Width of the button to 32.” rather than “Select the Width property in the object inspector, click on it, type 32 and press [Enter].”

Scripting Languages

Visual Basic Scripting (VBS) is the default language for Custom Forms. Other languages can be used such as Delphi (Pascal), Perl, Python and Java. Catalyst Pro was written using Borland's Delphi compiler. Because of this, it includes a built-in Delphi script interpreter and all the Windows common controls are based on the Delphi Visual Control Library. Any of the other languages (including VBS) require a scripting engine. The VBS scripting engine is available free from the Microsoft web site but it is installed automatically along with newer versions of Microsoft Internet Explorer so most computers will already have it. Perl, Python and Java are included for those who wish to use them (with a third-party Active Scripting Engine), but their use will not be covered in this manual.

All forms that come with Catalyst Pro use Delphi scripting. There are several reasons for this:

- We did not want to require the VBS scripting engine to be installed for Catalyst Pro to operate properly.
- The Delphi script interpreter is generally faster than the VBS engine because it is built-in and not a DLL or ActiveX component.
- The Delphi Visual Control Library and the language itself are more robust than VBS.

With that said, there are several reasons for you to use VBS instead of Delphi:

- It is easier to program and the speed difference will usually be unnoticeable.
- There are far more books, sample code and help available for VBS than for Delphi.
- Even if you use VBS, you can still call most of the built-in Delphi functions and use most of the Delphi classes and objects.

This manual assumes that you will be using VBS. It is not difficult to translate the examples if you prefer Delphi to VBS.

Events and Event Handlers

Custom Form programming, like virtually all Windows programming, is "event based". This means that the program responds to events rather than continuously executing sequential code.

For example, a button has an "OnClick" event. You can write script code inside the OnClick event and that code will be executed every time the button is clicked. All available events for a component are listed in the object inspector under the "Events" tab.

The script code you write for an event is called an "Event Handler". To create an event handler, double click the event it in the object inspector.

Example:

1. Create a new form using the "Blank (VBScript)" template.

2. Place a Button on the form from the WinStandard tab.

3. Select the "Events" tab in the object inspector.

4. Double click the "OnClick" event in the object inspector.

   The script code editor (text editor) should now be visible and you should see the following event handler shell:

Sub Button1Click(Sender)

End Sub

Insert the following script code inside the event handler so it looks like this:

```
Sub Button1Click(Sender)
     MsgBox "Hello World!", 65, "Event Handler Example"
End Sub
```

(MsgBox is a standard VBS function. See your VBS documentation for more information.)

Now return to view mode and click the button. You should see a message pop up that says “Hello World!”. If you did not see it or received an error message, go back to edit mode and check your typing.

Notice that the program does not waste time sitting in a loop constantly checking to see if the button was clicked. Catalyst Pro handles all that for you and only executes your script code when the button is clicked.

All events for all components work like this. There are dozens of events for some components while some components only have a few events. Short of reading an exhaustive list of all events and their purpose, the only way to really learn them is to experiment. You can generally ignore most events until you have a need for them.

- Feel free to experiment with the different event handlers available for each component. However, keep in mind that you are working with Windows programming at a relatively low level and it is not possible for Catalyst Pro to catch logical errors in your program and they can sometimes cause problems with the form. For example, if you where to use the MsgBox function in a forms “MouseMove” event, a message would pop up every time you moved the mouse over the form. This could make it quite difficult to enter edit mode.

- One of the most important events is the “OnFormShow” event of the Custom Form itself. This event allows you to initialize the form every time it is viewed. Use this event to zero out global variables, reset label captions, clear memo components or any other initialization code you may need.

- Since the components used in Catalyst Pro are based on the Delphi Visual Component Library, a Delphi reference manual or the Delphi help files would contain a complete description of all events. Most events are very simple and usually don’t need an explanation, but know that the information is available if you need it for very advanced applications.

Accessing Component Properties

When you place a component on a form, it is given a default name (this name can be changed using the "Name" property of the component.) The default name is usually the type of the component followed by a number. For example, the first button you place on a form would be named "Button1". The second button would be named "Button2", etc.

Each of the properties of a component can be accessed in script code by using the components name followed by a period "." then followed by the property name.

Example:

1. Create a new form and place a button on the form.
2. Place a "Label" component on the form from the "WinStandard" tab.
3. Create an "OnClick" event handler for the button.
4. In the event handler, write:

   Label1.Caption="Hello World!"

Now return to view mode and click the button. The caption of the label should now read "Hello World!".

If you decide to change the name of a component, make sure to change your code as well. For example, if you changed the name of Label1 to "MyLabel", the script code would need to be changed to:

MyLabel.Caption="Hello World!"

For all but the simplest forms, you will probably want to rename the components to something more meaningful such as "Okbutton", "CancelButton" or "StatusLED"

A Simple Example

Let's put everything discussed so far in to one simple example program. This example is not especially useful, but it does demonstrate how events, properties and the naming of components all work together.

1. Create a new form.
2. Place a "Meter" component on the form and select "Simple Circular Meter" for the template. Do not select a Control Point for this meter.
3. Place a "Knob" component on the form and select "Default" for the template. Do not select a Control Point for this knob.
4. Place an LEDRectangle component on the form. Do not select a Control Point for this LED.
5. Rename the LED component to "StatusLED".
6. Set the "ActiveColor" property of the LED to Lime.
7. Set the "Active" property of the LED to TRUE (check mark ON)
8. Set the "FlashDelay" property of the LED to 100.

9. Select the Knob and insert the following code in its “OnPositionChange” event handler:

```
Meter1.Position = Knob1.Position
```

This code simply takes the current position of the knob and sets the meters “Position” property equal to it.

10. Select the Meter and insert the following code in its “OnPositionChange” event handler:

```
If Meter1.Position > 75 then
  StatusLED.ActiveColor=clRed
  StatusLED.Flash=TRUE
elseif Meter1.Position < 25 then
  StatusLED.ActiveColor=clYellow
  StatusLED.Flash=TRUE
else
  StatusLED.ActiveColor=clLime
  StatusLED.Flash=FALSE
End If
```

(Note that any color in the “ActiveColor” property list can be used by placing a “cl” in front of the color name.)

Now return to view mode and move the knob from one extreme to the other.

You could just as easily eliminate the knob and select a Control Point for the meter. The effect would be the same except the meters movement and the LED color would reflect the value of the Control Point rather than the knobs position.

Built-in Script Functions

Catalyst Pro includes a large selection of built-in script functions. These functions can be used regardless of the scripting language. They include general functions such as string manipulation, file access, and user interface functions, as well as Catalyst Pro specific functions such as Control Point, Site and Form functions.

- Since the Delphi script interpreter is built-in, you have access to all the Delphi objects and classes as well as all of the objects and classes of the scripting language you are using. Delphi classes such as TStringList, TList and TDateTime are all available. In addition, the entire Delphi run-time library of functions is available, even when using VBS.

- It is important to note that ALL variables and function parameters used in script files are of type "Variant". Even if you specifically declare a variable to be of a different type, it is actually still a variant. Declaring variables of different types is useful mainly for keeping it straight in your own mind how you will be using a variable.

The naming convention used for built-in Catalyst Pro functions is normally a noun that describes what category the function belongs to, followed by the action that will be performed. For example, "StringGetLeft" to return the leftmost part of a string, or "GUIError" to display an error message (GUI stands for Graphic User Interface.)

- Many of the functions built-in to Catalyst Pro duplicate the functions of VBS or the Delphi run-time library. For example, "StringGetLeft" has the exact same effect as the VBS function "Left". The only difference is that "StringGetLeft" does its own error checking and exception handling so passing an invalid string to "StringGetLeft" will not cause an error, it will simply return a blank string.

String Manipulation Functions

FloatToString (Val:Single) :String

Converts a single precision floating point number to a string with 2 decimal places.

Returns: The converted string.

Example: FloatToString(58.2+3.5)) 'Returns "61.70"

HMSToString (H,M,S:Integer) :String

Converts Hour, Minute and Second to a string in time format.

Returns: The time in string format.

Example: HMSToString(12,30,45) 'Returns '"12:30:45"

Notes: No range checking is performed on H, M and S.

IntDateToString (Value:Integer) :String

Converts an integer date value to a string.

Example: IntDateToString(101005) ' Returns "01/05/2001"

Note: The format for an integer date is ((Year-1900)*1000)+DayOfYear. This format is used in the Infinity schedule program.

IntTimeToString (Value:Integer, AmPm:Boolean) :String

Converts an integer time value to a string.

Example: IntTimeToString(1535,TRUE) ‘ Returns “03:35 pm”

Note: The format for an integer time is (Hour*100)+Minute. This format is used in the Infinity schedule program.

IsAlpha (Ch:Char) :Boolean

Returns TRUE if Ch is an alphabetic character (a-z or A-Z).

IsAlphaNum (Ch:Char) :Boolean

Returns TRUE if Ch is an alphabetic or numeric character (a-z, A-Z or 0-9).

IsNum (Ch:Char) :Boolean

Returns TRUE if Ch is a numeric character (0-9).

StringDateToDateTime (Src:String) :TDateTime

Converts a string containing a date to a Delphi compatible TDateTime.

Returns: the TDateTime value.

Note: The date string passed must be in the format “Jan 01, 2000, 12:00:00”.

StringDecrypt (Src:String, Key:Byte) :String

Decrypts a string that has been encrypted with StringEncrypt.

Returns: The decrypted string.

Note: “Key” should be the same value used to encrypt the string.

StringDelete (Src:String, Index:Integer, Count:Integer) :String

Deletes a portion of a string.

Example: StringDelete(“Hello World!”,2,3) ‘Returns “Ho World!”

StringDeleteSubStr (Src:String, DelStr:String) :String

Deletes a sub-string from a string.

Example: StringDeleteSubString(“Hello World!”,” ell”) ‘ Returns “Ho World!”

StringEncrypt (Src:String, Key:Byte) :String

Encrypts a string using a simple XOR of each character with the Key value.

Returns: The encrypted string.

StringGetDelIndexed (Src:String, Delim:Char, Index:Integer) :String

Returns a portion of a string starting at the “Index” occurrence of the Delim character, up to but not including the next occurrence of the Delim character.

Example: StringGetDelIndexed(“@Item1@Item2@Item3@”,2,”@”) ‘Returns “Test2”

StringGetDelimited Src:String, Delim1:Char, Delim2:Char,Start:Integer) :String

Returns a portion of a string surrounded by Delim1 and Delim2, starting at character Index.

Example: StringGetDelimited(“The two values are [Val1] and [Val2]”, ”[“ , ”]” , 28) ‘Returns “Val2”

StringGetLeft (Src:String, Len:Integer) :String

Returns the leftmost Len characters of Src.

Example: StringGetLeft(“Hello World!”,5”) ‘Returns “Hello”

StringGetMid (Src:String, Pos:Integer, Len:Integer) :String

Returns Len characters of a string starting at Pos character.

Example: StringGetMid(“The value is 72.53 degrees”,14,5) ‘Returns “72.53”

StringGetNumber (Src:String,SPos:Integer) :Integer

Returns a number embedded in a string as an integer.

Example: StringGetNumber(”There are 31 days this month”,11) ‘Returns 31

StringGetRight (Src:String, Len:Integer) :String

Returns the rightmost Len characters of Src.

Example: StringGetRight(“Hello World!”,6”) ‘Returns “World!”

StringHexToInt (Str:String) :Integer

Converts a string containing a hexadecimal number to an integer.

Example: StringHexToInt(“FFFF”) ‘Returns 65535

Note: String can contain 1-4 hex characters.

StringInsertSubStr (Src:String, InsStr:String, Pos:Integer) :String

Inserts a sub-string in to a string.

Example: StringInsertSubString(“This is a test”, “nother”,10) ‘Returns “This is another test”

StringPadLeft (Src:String, PadChr:String, Len:Integer) :String

Pads the left side of a string with PadChr until it is at least Len characters long.

Example: StringPadLeft(“23”, “0”, 5) ‘Returns “00023”

StringPadRight (Src:String, PadChr:String, Len:Integer) :String

Pads the right side of a string with PadChr until it is at least Len characters long.

Example: StringPadRight(“23”, “0”, 5) ‘Returns “23000”

StringPadRightTrunc (Src:String, PadChr:String, Len:Integer) :String

Pads the right side of a string with PadChr until it is Len characters long or truncates the string if it is longer than Len. The returned string will always be exactly Len in length.

Example: StringPadRightTrunc(“23000”, “0”, 4) ‘Returns “2300”

StringParamsFromVA (VA:Variant) :String

Converts a variant array to a string of parameters.

Example:

```
Va[0]=”Hello World!”
Va[1]=123
Va[2]=TRUE
Va[3]=73.54
StringParamsFromVA(Va) ‘Returns “{Hello World!}{123}{TRUE}{73.54}”
```

Note: This function is useful for passing data to and from the communications drivers.

StringParamsToVA (Src:String) :Variant

Converts a string of parameters to a variant array.

Example:

```
Va=StringParamsToVA(“Hello World!}{123}{TRUE}{73.54}”)
Va[0]=”Hello World!”
Va[1]=”123”
Va[2]=”TRUE”
Va[3]=”73.54”
```

Note: All elements of the variant array will be strings. This function is useful for passing data to and from the communications drivers.

StringParseToList (ParmStr:String, List:TStringList, Delim1:Char,Delim2:Char)

Parse a string of parameters to a TStringList.

Example:

```
MyList=TStringList.Create
StringParseToList(“{Parm1}{Parm2}{Parm3}{Parm4}”,MyList, “{“ , “}”)

‘MyList now contains:
‘MyList[0]=”Parm1”
```

'MyList[1]="Parm2"
'MyList[2]="Parm3"
'MyList[3]="Parm4"

StringPos (Src:String, SubStr:String) : Integer

Returns the position of a sub-string within a string.

Example: StringPos("Hello World!","World") 'Returns 7

StringReplaceAllSubStr (Src:String, SubStr:String, NewStr:String) :String

Replaces all occurrences of a sub-string in a string.

Example: StringReplaceAllSubStr("Test1 TEST2 TeSt3", "test", "MyTest") 'Returns "MyTest1 MyTest2 MyTest3"

Note: The search for the sub-string is NOT case sensitive.

StringStripAllSubStr (Src:String, StripStr:String) :String

Strips all sub-strings out of a string.

Example: StringStripAllSubStr("Test1 Test2 TEST3", "Test") 'Returns "1 2 TEST3"

Note: The search for the sub-string IS case sensitive.

StringToBool (Src:String) :Boolean

Converts a string to a Boolean.

Example:
StringToBool("TRUE") 'Returns TRUE
StringToBool("0") 'Returns FALSE

Note: Valid values are "TRUE", "FALSE" or any number in string format. Any number that converts to non-zero value will return TRUE.

StringToDouble (Src:String) :Double

Converts a string to a double precision floating point number.

Example: StringToDouble("-358.456") 'Returns –385.456

StringToFloat (Src:String) :Single

Converts a string to a single precision floating point number.

Example: StringToFloat("58.37") 'Returns 58.37

StringToHMS (Str:String, var H:Integer, var M:Integer, var S:Integer)

Converts a string containing a time to Hours, Minutes and Seconds.

Example:

```
Dim H
Dim M
Dim S

StringToHMS("12:35:54,H,M,S)
'now H = 12,M = 35 and S = 54
```

StringToInt (Src:String) :Integer

Converts a string to an integer,

Example: StringToInt(-58) 'Returns -58

StringToIntDate (Text:String, var Value:Integer) :Boolean

Converts a string containing a date to an integer date.

Example:

```
StringtoIntDate("01/05/2001")
'If successful, returns TRUE and sets Value=101005
'If string is not valid integer date, returns FALSE and sets Value=999999
```

Note: The format for an integer date is ((Year-1900)*1000)+DayOfYear. This format is used in the Infinity schedule program.

StringToIntTime (Text:String, var Value:Integer) :Boolean

Converts a string containing a time to an integer time

Example:

```
StringtoIntTime("12:32")
'If successful, returns TRUE and sets Value=1232
'If string is not valid integer time, returns FALSE and sets Value=9999
```

Note: The format for an integer time is (Hour*100)+minute. This format is used in the Infinity schedule program.

StringTrimAfterSubStr (Src:String, TrimStr:String) :String

Trim all characters from the left of up to and including a sub-string.

Example: StringTrimAfterSubStr("Zone1 temperature = 58.35", "=") 'Returns "58.35"

StringTrimSpacesBoth (Src:String) :String

Trims the spaces from both sides of a string.

Example: StringTrimSpacesBoth(" Hello World! ") 'Returns "Hello World!"

StringTrimSpacesLeft (Src:String) :String

Trims the spaces from the left side of a string.

Example: StringTrimSpacesLeft(" Hello World! ") 'Returns "Hello World! "

StringTrimSpacesRight (Src:String) :String

Trims the spaces from the right side of a string.

Example: StringTrimSpacesLeft(“ Hello World! “) ‘Returns “ Hello World!”

StringTrimToSubStr (Src:String, TrimStr:String) :String

Trims a string up to but not including the first occurrence of a sub-string.

Example: StringTrimToSubStr(“This is a test string,” “test”) ‘Returns “test string”

StringTrimWSLeft (Src:String) :String

Trims all spaces and tabs (sometimes called “white space”) from the left side of a string.

Example: StringTrimWSLeft(“ Hello World! ”) ‘Returns “Hello World! ”

StringTrimWSRight (Src:String) :String

Trims all spaces and tabs (sometimes called “white space”) from the right side of a string.

Example: StringTrimWSRight(“ Hello World! ”) ‘Returns “ Hello World!”

StringTruncFloat (Src:String) :String

Truncates the decimal portion of a string containing a floating point number.

Example: StringTruncFloat(“123.45”) ‘Returns “123”

StringTruncSubStr (Src:String, TrunStr:String) :String

Truncates all characters from the right side of a string up to and including the first occurrence of a sub-string.

Example: StringTruncSubStr(“The is a test string”,”string”) ‘Returns “This is a test”

StringTruncSubStrRev (Src:String, TrunStr:String) :String,

Truncates all characters from the right side of a string up to and including the last occurrence of a sub-string.

Example: StringTruncSubStrRev(“The is a test string”,”t”) ‘Returns “This is a test s”

StringTruncZeros (Src:String) :String

Truncates all trailing zeros from a string.

Example: StringTruncZeros(“72.500”) ‘Returns “72.5”

StringWordGet (Src:String, StartIdx:Integer) :String

Returns a word from a sentence. SPos can point to any one of the characters in the word.

Example: StringGetWord(“The is a test sentence.”,12) ‘Returns “test”

Graphical User Interface (GUI) Functions

GuiAsk (Caption:string): Integer

Display a dialog box that prompts the user for a “Yes” or “No” response.

Example:

If GuiAsk(“Are you sure you want to quit?”) = mrYes then Close

Returns the built-in constants mrYes or mrNo.

GuiAskCancel (Caption:string): Integer

Display a dialog box that prompts the user for a “Yes”, “No” or “Cancel” response.

Example:

If GuiAskCancel(“Closing form…Save current data?”) = mrCancel then DontClose

Returns the built-in constants mrYes, mrNo or mrCancel.

GuiDialogCancelAsk :Boolean

Displays a dialog asking the user “'Cancel Changes...Are You Sure?'.

Example: if GuiCancelAsk = TRUE then CancelChanges

GuiDialogErrorInvalid

Displays the message “The information in this field is invalid.”.

GuiDialogErrorMissing

Displays the message “Information in this field is required”.

GuiError (Caption:string): Integer

Displays an error message.

Example: GuiError(“Cannot open the file.”)

GuiFileOpen (Filename:String, Path:String, DefaultExt:String,Filter:String, Caption:String) :Boolean

Displays a file selection dialog box that lets the user browse the hard drive.

Example:

Dim Filename

If GuiFileOpen (Filename,”C:\MyFolder”,”TXT”, ”Text Files (*.txt)|*.Txt” ,”Select a text file”) =TRUE then OpenFile

Note: If successful, “Filename” contains the full path to the selected file.

GuiFileSaveAs (Filename:String, Path:String, DefaultExt:String,Filter:String, Caption:String) :Boolean

Displays a file selection dialog box that lets the user browse the hard drive.

Example:

Dim Filename

If GuiFileSaveAs (Filename,"C:\MyFolder","TXT", "Text Files (*.txt)|*.Txt" ,"Select a text file") =TRUE then SaveFile

Note: If successful, "Filename" contains the full path to the selected file.

GuiFileSelect (Title:String, FileMask:String, Attrib:Integer): String

Displays a list of all matching files in a single directory and allows the user to select one.

Example: GuiFileSelect("Select a Form", "C:\MyForms*.Frm",faAnyFile)

Note: If successful, Filename contains only the filename portion of the path.

Attrib can be any of the following values:

faReadOnly	Read-only files
faHidden	Hidden files
faSysFile	System files
faVolumeID	Volume ID files
faDirectory	Directory files
faArchive	Archive files
faAnyFile	Any file

Attributes can be combined by adding their constants or values. For example, to search for read-only and hidden files in addition to normal files, pass (faReadOnly + faHidden) as the Attrib parameter.

GuiInform (ACaption:string): Integer

Displays a general informational message.

Example: GuiInform("The data was saved")

GuiInputBox (Title, Caption:String, Value:String, MaskStr:String, CharMax:Integer): Integer

Displays an edit box for the user to type text.

Example:

If GuiInputBox("Change Setpoint", "Enter a new value:", "####.##",7) = mrOK then ChangeValue

Returns the built-in constants mrOK or mrCancel

Note: If MaskStr is a blank string (""), the user can type any text they wish. CharMax is the maximum number of characters the user will be allowed to type.

GuiInputPassword (Title, Caption:string, Value:String, MaskStr:String, CharMax:Integer): Integer

Displays an edit box for the user to type a password or other sensitive information. The text they type will not appear on the screen.

Example:

```
If GuiInputPassword("View Log files", "Enter your password", "",32) = mrOK then ViewLogs
```

Returns the built-in constants mrOK or mrCancel

Note: If MaskStr is a blank string (""), the user can type any text they wish. CharMax is the maximum number of characters the user will be allowed to type.

GuiListPick (List:TStringList, Title:String, ItemIndex:Integer) :Integer

Displays all the strings in a TStringList and allows the user to select one of them.

Example:

```
Dim Selection

MyList=TStringList.Create
MyList.Add("Item1")
MyList.Add("Item2")
MyList.Add("Item3")
Selection=GuiListPick(MyList,"Pick an Item",2)
MyList.Free
```

Note: ItemIndex is the item number you want to be highlighted by default. The first item is 0. Use –1 for no default selection.

GuiWarn (Caption:string): Integer

Displays a warning message to the user.

Example: GuiWarn("Drive C: is almost full.")

File Functions

FilCopy (const FileName: String, DestName: String) : Boolean

Copy a file. Returns TRUE if successful.

Example:

```
If FilCopy("C:\MyFiles\MyDataFile.Dat", "D:\Backup\MyOldDataFile.Dat") = FALSE then ErrorMsg
```

Note: The full path and filename must be specified for both the source and destination. Wildcards are not allowed.

FilCreate (Filename:String) :Boolean

Create a new, empty file. Returns TRUE if successful.

Example:

```
If FilCreate("C:\MyFile\MyData.Dat")=FALSE then ErrorMsg
```

Note: The full path and filename must be specified.

FilDelete (Filename:String) : Boolean

Deletes a file. Returns TRUE if successful.

Example:
If FilDelete("C:\MyFiles\MyDataFile.Dat") = FALSE then ErrorMsg

Note: The full path and filename must be specified. Wildcards are not allowed.

✪ This function does not ask the user for confirmation before deleting the file. Use with caution.

FilDirExists (Path:String) :Boolean

Returns TRUE if the specified directory exists.

Example: if FilDirExists("C:\MyFolder") = FALSE then FilDirectoryCreate("C:\MyFolder")

Note: The full path must be specified. Wildcards are not allowed.

FilDirectoryCreate(Path:String) : Boolean

Create a disk directory.

Example:
If FilDirectoryCreate("C:\MyFolder") = FALSE the ErrorMsg

Note: The full path must be specified.

FilDirectoryRead (List:TStringList, FileMask:String, Attrib:word)

Read the contents of a disk directory in to a TStringList.

Example:

```
Dim MyList

MyList=TStringList.Create
FilDirectoryRead MyList,"C:\MyFolder\*.*",faAnyFile
If MyList.Count > 0 then GuiListPick MyList,"Pick a File",-1
MyList.Free
```

Attrib can be any of the following values:

faReadOnly	Read-only files
faHidden	Hidden files
faSysFile	System files
faVolumeID	Volume ID files
faDirectory	Directory files
faArchive	Archive files
faAnyFile	Any file

Attributes can be combined by adding their constants or values. For example, to search for read-only and hidden files in addition to normal files, pass (faReadOnly + faHidden) as the Attrib parameter.

FilExists (Filename:String) :Boolean

Returns TRUE if the specified file exists.

Example: If FilExists(“C:\MyFile\MyData.Dat”) = TRUE then GuiError(“File Already Exists.”)

Note: The full path must be specified. Wildcards are not allowed.

FilLength (Filename:String) :Integer

Returns the length of a file, in bytes.

Example: if FilLength(“C:\MyFile\MyData.Dat”) = 0 then GuiWarn(“No Data Available.”)

Note: The full path must be specified.

FilMove (const FileName:String, DestName:String) : Boolean

Moves a file from one folder to another. Returns TRUE if successful.

Example:

```
If FilMove(“C:\MyFiles\MyDataFile.Dat”, “D:\Backup\MyOldDataFile.Dat”) = FALSE then ErrorMsg
```

Note: The full path and filename must be specified for both the source and destination. Wildcards are not allowed.

FilNameLegal (Filename:String) :Boolean

Returns TRUE if the given filename is a legal Windows filename.

Note: This function only checks for illegal characters in the filename. It does not verify that a path or file exists.

FilOpenReading (Filename:String) :Integer

Opens an existing file in read mode.

Example:

```
Dim FileHandle

FileHandle=FilOpenReading(“C:\MyFiles\MyData.Dat”)
If FileHandle > 0 then
        ReadSomeData
        FilClose(FileHandle)
Endif
```

Note: If successful, an integer “file handle”: is returned. Use this file handle to access and close the file.

✪ A file MUST be closed when you are finished accessing it. If it is not closed the data could become corrupted.

FilOpenWriting (Filename:String) :Integer

Opens an existing file in write mode.

Example:

```
Dim FileHandle
```

```
FileHandle=FilOpenWriting("C:\MyFiles\MyData.Dat")
If FileHandle > 0 then
        WriteSomeData
        FilClose(FileHandle)
End if
```

Note: If successful, an integer "file handle": is returned. Use this file handle to access and close the file.

✪ A file MUST be closed when you are finished accessing it. If it is not closed the data could become corrupted.

FilReadVarData (FH:Integer, Buff:Variant, Size:Integer) : Boolean

Reads binary data from a file and stores it in a variant.

Example:

```
Dim VA
Dim FileHandle

FileHandle=FilOpenReading("C:\MyFiles\MyData.Dat")
If FileHandle > 0 then
        If FilReadVarData(FileHandle,VA,1000) = FALSE then GuiError("Cannot Read File.")
        FilClose(FileHandle)
End If
```

Note: If successful, the variant is a zero-based array of bytes "Size" bytes long.

FilRename (Filename:String, NewName:String) :Boolean

Renames a file.

Example:

```
If FilRename("C:\MyFiles\MyDataFile.Dat", "D:\MyFiles\MyOldDataFile.Dat") = FALSE then ErrorMsg
```

Note: The full path and filename must be specified for both the source and destination. Wildcards are not allowed.

FilUniqueFilename(Path:String, BaseName:String, Ext:String) :String

Returns a unique filename.

Example: Filename=FilUniqueFilename("C:\MyFiles","MyData",".Dat")

Note: This function checks for the filename specified by BaseName+Ext ("MyData.Dat" for the above example.) If that file already exists, it will check for "MyData2.Dat" through "MyData100.Dat". It will return the first filename it finds that does not already exist.

FilWriteVarData (FH:Integer, Buff:Variant) : Boolean

Writes a variant byte array to a binary file.

Example:

```
Dim VA
Dim FileHandle
```

```
FillAVariantWithSomeData(VA)
FileHandle=FilOpenWriting("C:\MyFiles\MyData.Dat")
If FileHandle > 0 then
        If FilWriteVarData(FileHandle,VA,1000) = FALSE then GuiError("Cannot Write to File.")
        FilClose(FileHandle)
End If
```

Control Point Functions

ControlPointDecodeString (Pnt:TControlPoint, StringMask:String) :String

Decodes a format string, inserting data from the Control Point where specified.

The format string may contain any of the following codes:

- @S for the site name
- @P for the point description
- @A for the point address
- @T for the type
- @C for the class
- @U for the units
- @V for the value
- @R for carriage return

Examples:

- ControlPointDecodeString(MyPoint,"@P = @V @U") 'Returns "Zone1 = 72.34 Deg. F".
- ControlPointDecodeString(MyPoint, "Average Zone Temp is @V @U") 'Returns "Average Zone Temp is 73.4 Deg. F"
- ControlPointDecodeString(MyPoint,"@S Economizer mode is @V") 'Returns "Central College Economizer mode is ON".

ControlPointDigitalUnitsToString (Pnt:TControlPoint, Value:Boolean) :String

Converts a Boolean TRUE/FALSE value to a string based on the "Units" of the point.

Example:

```
MyPoint.Units="Open/Closed"
ControlPointDigitalUnitsToString(MyPoint, FALSE) 'Returns "Open"
```

ControlPointFind (StringName:String) :TControlPoint

Finds a Control Point in the database.

Example:

```
MyPoint=ControlPointFind("{Central College}{Outside Air Temp}{Infinity1 OutTemp}")
If MyPoint <> NULL then Label1.Caption = MyPoint.PointName
```

ControlPointFindDynamic(StringName:String) :TControlPoint

Finds a Control Point in the internal list of points that are currently being updated (i.e. used on an active Graphic Display).

Example:

```
MyPoint=ControlPointFindDynamic("{Central College}{Outside Air Temp}{Infinity1 OutTemp}")
If MyPoint <> NULL then Label1.Caption = MyPoint.Value
```

ControlPointModify (Pnt:TControlPoint) :Boolean

Modifies the value of a Control Point on the panel.

Example:

```
MyPoint.Value="58.25"
If ControlPointModify(MyPoint) = FALSE then GuiError("Could Not Modify Point Value.")
```

ControlPointParsePoint (PntString:String, SiteStr:String, PntStr:String, AddStr:String) :Boolean

Parses a full Control Point ID in to a Site name, Description and address.

Example:

```
Dim SiteName
Dim Description
Dim Address

ControlPointParsePoint("{Central College}{Outside Air Temp}{OutTemp}", SiteName,
Description,Address)

'SiteName = "Central College", 'Description = "Outside Air Temp." and Address="OutTemp"
```

ControlPointSelect (CaptionText:String, DefaultSelection:String) :String

Displays the Control Point database and allows the user to select a point.

Example: ControlPointSelect("Please select a point", "{Central College}{Outside Air Temp}{OutTemp}")

Note: If DefaultSelection is an empty string (""), no point will be highlighted by default.

ControlPointTriStateUnitsToString (Pnt:TControlPoint, Value:Integer) :String

Converts a tri-state value to a string based on the "Units" of the point.

Example:

```
MyPoint.Units="Auto/OV Off/OV On"
ControlPointTriStateUnitsToString(MyPoint, -1) 'Returns "Auto"
ControlPointTriStateUnitsToString(MyPoint, 0) 'Returns "Off"
ControlPointTriStateUnitsToString(MyPoint, 1) 'Returns "On"
```

ControlPointValueToString (Pnt:TControlPoint) :String

Converts a Control Points value to a string based on "Units" for digital and tri-state points, and "Format" for analog points.

Example:

```
MyPoint.Format="###.#"
MyPoint.Value="273.745"
ControlPointValueToString(MyPoint) 'Returns "243.7"

MyPoint.Units="Closed/Open"
MyPoint.Value="OFF"
ControlPointValueToString(MyPoint) 'Returns "Closed"
```

Note: This is the exact same function that Catalyst Pro uses internally to display values on Graphic Displays. It will convert any point value to the correct string representation.

The TControlPoint Object

The TControlPoint object has the following properties:

- **PointName** – The description of the point.
- **Address** The address of the point.
- **PointType** – The type of the point. Possible values are:
  - **ptUnknown**
  - **ptDigital**
  - **ptAnalog**
  - **ptTristate**
  - **ptString**
  - **ptDate**
  - **ptTime**
  - **ptDateTime**
- **PointClass** – The class of the point. Possible values are:
  - **pcUnknown**
  - **pcInput**
  - **pcOutput**
  - **pcVariable**
- **Overridden** – The override state of the point. Possible values are:
  - **poUnknown**
  - **poOverridden**
  - **poAuto**
- **Disabled** – The disabled state of the point. Possible values are:
  - **pdUnknown**
  - **pdDisabled**
  - **pdEnabled**
- **Units** – the electrical units assigned to the point ("Deg.F", "Volts", etc.)
- **Format** – The display format of the point, for example, "####.##".
- **Value** – The raw value of the point in string format.
- **AllowDisable** – TRUE if the point can be disabled/enabled by a user.
- **AllowModify** – TRUE if the point can be modified by a user.

TControlPoint also has one method:

StringNameGet – Returns the full ID of the point in the format "{sitename}{description}{address}".

Example: Label1.Caption = MyPoint.StringNameGet

Note: All of the status values will be unknown until they are explicitly read using the Control Point functions below.

Site Functions

SiteCommand (SiteStr:String,InputArray:Variant, OutputArray:Variant,Title:String,Text:String) :Boolean
SiteCommandQue (Frm:TApp,SiteStr:String,InputArray:Variant,PacketID:Integer) :Boolean,
SiteCommandQueHigh (Frm:TApp,SiteStr:String,InputArray:Variant,PacketID:Integer) :Boolean,

These are very low-level function to send and receive raw data to and from a Sites panels. The parameters and the data returned are completely different from one type of panel to the next. Call technical support if you need more information about these commands.

SiteConnect (Path:String) :Boolean,

Connect to a Site or group of Sites.

Example: SiteConnect("C:\Catalyst\Sites\[My Site]") 'Returns TRUE if connect was successful

SiteConnectOffline (Path:String) :Boolean

Connect to a Site or group of Sites in offline mode.

Example: SiteConnectOffline("C:\Catalyst\Sites\[My Site]") 'Returns TRUE if connect was successful

SiteCurrentDriver :String

Returns the driver name for the currently active Site (the Site that appears in the "Site Selection" box of the main form.)

SiteCurrentGet :String

Returns the description of the currently active Site (the Site that appears in the "Site Selection" box of the main form.)

SiteCurrentSet (SiteStr:String) :Boolean

Sets the currently active Site.

SiteDisconnect (SiteStr:String) :Boolean

Disconnect from a single Site.

SiteDisconnectAll :Boolean

Disconnect from all Sites.

SiteIsOffline (SiteStr:String) :Boolean

Returns TRUE if the specified Site is in offline mode.

SiteListGet (List:TStringList)

Fills a TStringList with the list of all connected Sites.

Example:

```
Dim SiteList

SiteList=TStringList.Create
SiteListGet(SiteList)
'SiteList[0]='Central College', SiteList[1]='My Test Site', etc.
SiteList.Free
```

SiteListGetByDriver (List:TStringList, DriverString:String)

Fills a TStringList with the list of all connected Sites that are of type "DriverString"
Example:

```
Dim SiteList

SiteList=TStringList.Create
SiteListGetByDriver(SiteList,"Andover Infinity")
'SiteList[0]='My Infinity Site1', SiteList[1]='My Infinity Site2', etc.
SiteList.Free
```

SitePathGet (SiteStr:String) :String

Returns the full path for the specified Site.

Example: SitePathGet("Central College") 'Returns "C:\Catalyst\Sites\[Central College]"

SitePointDisabledGet(SiteStr:String, PointStr:String, Disabled:Boolean) :Boolean

Sets the variable "Disabled" to the disabled state of a Control Point.

Example:

```
Dim Disabled

SitePointDisabledGet("Central College","OutdoorTemp",Disabled) 'Returns TRUE if state was read
'Disbaled now equals TRUE if the point is disabled.
```

Note: PointStr can be any valid point for the current type of equipment, or it can be a full Control Point ID ("{SiteName}{Decription}{Address}"). The point does not have to be in the Control Point database.

SitePointDisabledSet(SiteStr:String, PointStr:String, Disabled:Boolean) :Boolean

Enables or disables a Control Point on a panel.

Example:

```
SitePointDisabledSet("Central College","OutdoorTemp",TRUE) 'Returns TRUE if state was changed
```

Note: PointStr can be any valid point for the current type of equipment, or it can be a full Control Point ID ("{SiteName}{Decription}{Address}"). The point does not have to be in the Control Point database. A return value of TRUE does not mean the state was actually toggled, it only means that the panel accepted the command. If you disable a point that is already disabled a TRUE will still be returned.

SitePointValueGet (SiteStr:String, PointStr:String, Value:Variant) :Boolean

Sets the variable “Value” to the current value of a control Point in a panel.

Example:

```
Dim Value

SitePointValueGet(“Central College”,”OutdoorTemp”,Value) ‘Returns TRUE if value was read
‘Value now equals the value of the point in string format.
```

SitePointValueSet (SiteStr:String, PointStr:String, Value:Variant) :Boolean

Sets the value of a Control Point on a panel.

Example:

```
SitePointValueSet(“Central College”,”Setpoint1”,”78.5”) ‘Returns TRUE if point was modified.
```

Note: PointStr can be any valid point for the current type of equipment, or it can be a full Control Point ID (“{SiteName}{Decription}{Address}”). The point does not have to be in the Control Point database. A return value of TRUE does not mean the value was actually changed it only means that the panel accepted the command. If you change a point to the same value it already had a TRUE will still be returned.

SiteSendCommand (SiteStr:String, CmdStr:String, Title:String) :Boolean

Issues a command to a panel. The CmdStr parameter varies from one type of panel to the next.

Example:

```
SiteCommandSend(“My Infinity Site”,”PrintKWReport”,”Printing Report…Please Wait.”) ‘Executes the
user-written “PrintKWReport” function on the panel.
```

The TSite Object

It should not be necessary to create or use TSite objects. The following outline of the properties and methods is included for the rare application that may need this type of information.

The TSite object has the following properties.

```
RemoteName    :String   ‘These properties match those in the Site Database
PhoneNumber   :String   ‘
ComPort       :Integer  ‘
BaudRate      :Integer  ‘
Logon         :String   ‘
Password      :String   ‘
IntParm1      :Integer  ‘The remaining properties vary between different types of equipment
IntParm2      :Integer  ‘See the drivers .Ini file for more information
IntParm3      :Integer  ‘
FloatParm1    :Double   ‘
FloatParm2    :Double   ‘
FloatParm3    :Double   ‘
StringParm1   :String   ‘
StringParm2   :String   ‘
StringParm3   :String   ‘
StringParm4   :String   ‘
StringParm5   :String   ‘
BoolParm1     :Boolean’
BoolParm2     :Boolean’
BoolParm3     :Boolean’
```

BoolParm4 :Boolean'
BoolParm5 :Boolean'

TSite also has the following methods:

DataLoad(Path:String) :Boolean
DataSave(Path:String) :Boolean
PointsRead :Boolean
PointsWrite :Boolean
PointsSort
PointsFree
PointFind(PointName:String; Address:String) :Integer
PathName :String
FormTypeGet(Description:String) :Integer
FindDisplayByName(NameStr:String) :String
FindScheduleByName(NameStr:String) :String
FindAppByName(NameStr:String) :String
FormTypeGetByName(NameStr:String) :Integer
FormUniqueNameGet(Description:String) :String
DynamicPointFind(PointName:String; Address:String) :TControlPoint
DynamicPointAdd(ControlPoint:String; QLName:String; QLIndex:Integer)
DynamicPointDelete(ControlPoint:String; QLName:String; QLIndex:Integer)
DynamicPointsClear
PointUpdate(PntStr:String)

Form Functions

FormParametersGet (Frm:TApp) :Variant

The FormView functions allow a set of parameters to be sent to a form when it is viewed. The FormParametersGet function is used to retrieve those parameters.

Example:

Code in Form1:

```
Dim Parm

Parms=VarArrayCreate([0,1],VarVariant) 'Create a 2 element array
Parm[0]="MyData.Dat"
Parm[1]=350.5
FormView("Form2","Central College",Parms)
```

Code in Form2:

```
Sub AppFormShow(Sender)

    Dim Parms
    Dim Filename
    Dim DataValue

    Parms=FormParametersGet(Form2);
    if VarIsArray(Parms) then
      Filename=Parms[0] 'Filename now = "MyData.Dat"
      DataValue=Parms[1] 'Datavalue now = 350.5
    End if
End Sub
```

FormResultSet (Frm:TApp, Parm:Variant)

The FormResultSet function is used to return values to a calling form.

Example:

Code in Form1:

```
Dim ReturnVal

ReturnVal=FormViewModal("Form2","Central College","")
'ReturnVal = "Success" or "Failure"
```

Code in Form2:

```
If Successful then FormResultSet(Form2,"Success")
Else FormResultSet(Form2,"Failure")
```

Note: Only the "Modal" versions of the various FormView functions return useful values since the non-modal version do not wait for the called form to execute any code. The return value can be a full variant array as well as a simple string.

FormSiteGet (Frm:TApp) :String

Returns the description of the Site that the form belongs to.

Example:

```
Label1.Caption=FormSiteGet(MyForm) 'Caption='Central College'
```

FormSiteGetDriver (Frm:TApp) :String

Returns the driver description of the Site that the form belongs to.

Example:

```
Label1.Caption=FormSiteGetDriver(MyForm) 'Caption='Andover Infinity'
```

FormSiteSet (Frm:TApp, SiteStr:String)

Sets the Site description for a form. This is useful only for forms that do not belong to any particular Site. This is a temporary setting that is not saved with the form.

FormStringGet (Frm:TApp) :String

Every form contains one string that is saved along with the form every time it is closed. This allows you to automatically store a small amount of data that can be retrieved the next time the form is viewed.

Example:

```
'Displays the number of times the form has been viewed
Sub AppFormShow(Sender)
  Dim ViewCount

  ViewCount=FormStringGet(TestForm)
  if ViewCount=0 then ViewCount=1
  Label1.Caption="This form has been viewed "+ViewCount+" times."
```

```
    ViewCount=ViewCount+1
    FormStringSet TestForm,ViewCount

End Sub
```

Note: Use StringParamsToVA and StringParamsFromVA to store multiple parameters in the form string.

FormStringSet (Frm:TApp, Str:String)

Sets the built-in form string. See FormStringGet for more information.

FormView (FormName:String, SiteStr:String, Parms:Variant) :Variant

View a form from a Sites Custom Form database.

Example: FormView “MyCustomForm”, “MySite”, “”

Note: This function loads the Custom Form and then continues executing its script code. It does not wait for the user to close the called form.

FormViewByDriver (FormName:String, DriverStr:String, SiteStr:String, Parms:Variant) :Variant

View a driver specific form.

Example:

```
Dim DriverName

DriverName=FormSiteGetDriver(MyForm)
FormViewByDriver “ImportPoints”, DriverName, “MySite”, “”
‘The correct ImportPoints form will now be executed regardless of what type of devices are used.
```

Note: This function loads the form from the specified drivers \Form folder and then continues executing its script code. It does not wait for the user to close the called form.

FormViewByPath (Path:String, SiteStr:String, Parms:Variant) :Variant

View a form using a full path.

Example: FormViewByPath “C:\MyForms\MyForm1.Frm”

Note: Graphic Displays and Schedules may also be viewed with this function.

FormViewModal (FileName:String, SiteStr:String, Parms:Variant) :Variant
FormViewModalByDriver (Filename:String, DriverStr:String, SiteStr:String, Parms:Variant) :Variant
FormViewModalByPath (Path:String, SiteStr:String, Parms:Variant) :Variant

These are “Modal” versions of the ForrmView functions. When the modal versions are used, the called form will capture the focus of the entire program until it is closed. Once the form is closed, the calling form resumes execution.

Help Functions

HelpFileSearch (HelpFile:String, Keyword:String)

Search a custom help file for a keyword.

Example: HelpFileSearch “MyHelpFile”,”SomeKeyword”

HelpViewFile (FilePath:String,ContextID:Integer)

View a specified topic in a custom help file.

Example: HelpViewFile “MyHelpFile”, 1050 ‘view topic # 1050

Users Functions

UserAccessLevel :Integer

Returns the access level (0-255) of the current user or –1 if no user is currently logged in.

Example: if UserAccessLevel >100 then EnableSpecialFeature

UserAccessLevelCheck (Lvl:Integer) :Boolean

Returns TRUE if the current users access level is greater than or equal to “Lvl”.

Example: If UserAccessLevelCheck(50)=FALSE then CloseForm

UserAccessLevelCheckWarn (Lvl:Integer) :Boolean

Returns TRUE if the current users access level is greater than or equal to “Lvl”. Otherwise displays an “Access Denied” type message to the users and returns FALSE.

Example: If UserAccessLevelCheckWarn(50)=FALSE then CloseForm

UserHasAccess (Fnc:Integer) :Boolean

Returns TRUE if a user has access to a specific feature.

Example: If UserHasAccess(uaBackupMemory) then GoBackupTheMemory

Valid constants for the “Fnc” parameter are as follows:

uaAdvancedOptions
uaExitProgram
uaConnect
uaAckAlarms
uaEditUsers
uaEditPoints
uaEditSites
uaEditDisplays
uaEditSchedules
uaEditTrends
uaEditForms
uaModifySetpoints
uaModifySchedules
uaBackupMemory
uaRestoreMemory

UserHasAccessWarn (Fnc:Integer) :Boolean,

Returns TRUE if a user has access to a specific feature. Otherwise displays an "Access Denied" type message to the users and returns FALSE.

See "UserHasAccess" for more information.

UserLogin (Name:String,Password:String) :Boolean

Log a user in to the program.

Example: UserLogin ("Joe Smith","JoesPassword")

Note: Returns TRUE if successful.

UserLogout

Log the current user out of the program.

Variant Utility Functions

VarConvertFromData (Va:Variant, Data:Variant, Index:Integer, Key:String) :Integer

Converts a block of binary data to a variant array.

Since VBS and Delphi scripting do not allow the "pointer" type, this function is supplied to convert blocks of binary data to a variant array. Use VarConvertToData to convert the variant array back to binary data.

The parameters are:

- **Va** – The variant array to store the data in. This must already be dimensioned to the proper number of elements.
- **Data** – A variant set up as an array of bytes. This is the buffer that must contain the data to convert. See "FilReadVarData" for information on reading data from files.
- **Index** – The offset within the data to start converting.
- **Key** – A string that describes the data to convert. Valid key codes are:
  - "I" – A 4 byte long integer value.
  - "O" – A 1 byte Boolean value.
  - "N" – A 4 byte single precision floating point value.
  - "D" – An 8 byte double precision floating point value.
  - "M" – A 2 byte small integer value.
  - "T" – An 8 byte TDateTime value.
  - "Sxxx" – A string. "xxx" specifies the length.
  - "Bxxx" – A block of bytes. "xxx" specifies the length.

Example:

'For this example, we will read 54 bytes from a file, call a function that changes the data is some way and finally writes the data back to the file. The 54 bytes are a structure with the following format:

Integer (4 bytes)
Boolean (1 byte)
Single (4 bytes)
Single (4 bytes)

String (32 characters)
Boolean (1 byte)
Double (8 bytes)

Script Code:

```
Dim Data
Dim FileHandle
Dim Key
Dim Va(6) 'Va is 7 element array (0-6)


FileHandle=FilOpenReading("C:\MyFiles\MyData.Dat")
If FileHandle > 0 then
        If FilReadVarData(FileHandle,Data,54) = TRUE then
          Key="IONNS32OD" 'describe the data format
          VarConvertFromData(Va,Data,0,Key) 'Convert data to variant array
          'Va now contains something like the following values
          'Va(0)=35
          'Va(1)=TRUE
          'Va(2)=74.3
          'Va(3)=128.52
          'Va(4)="This is a test string"
          'Va(5)=FALSE
          'Va(6)=15834.9932
          ChangeTheData(Va)
          VarConvertToData(Va,Data,0,Key) 'Convert the data back to binary
          FileWriteVarData(FileHandle,Data) 'write the data back to the file

        End If
        FilClose(FileHandle)
End If
```

VarConvertToData (Va:Variant, Data:Variant, Index:Integer, Key:String) :Integer

Converts a variant array to a block of binary data. See VarConvertFromData for more information.

VarDataType (va:Variant) :Integer

Returns the type of the data held in a variant.

Valid return values are:

VarEmpty
varNull
varSmallint
varInteger
varSingle
varDouble
varCurrency
varDate
varOleStr
varDispatch
varError
varBoolean
varVariant

varUnknown
varByte
varStrArg
varString
varAny

Note: This function returns only the data type. An array of bytes passed to this function would return varByte. Use VarIsArray to determine if the variant is an array.

VarEditAsObject (Frm:TApp, Data:Variant, Options:Variant, Title:String, Sorted:Boolean, UseApply:Boolean, UseHelp:Boolean) :Boolean

This function allows the user to edit a list of properties stored in a variant array. It displays a dialog box with an OK and Cancel button, and optionally, Help and Apply buttons.

Note: This function returns TRUE if the OK button was clicked, FALSE if Cancel was clicked.

The parameters are as follows:

- **Frm** – The name of the current form.
- **Data** – A variant array that contains the current values of the properties (see below.)
- **Options** – A variant array that contains the options for each property (see below.)
- **Title** – The text to display in the caption of the dialog box.
- **Sorted** – Set to TRUE to display the properties alphabetically. FALSE will display them in the order they are defined in the array.
- **UseApply** – Shows the Apply button if TRUE.
- **UseHelp** – Shows the Help button if TRUE.

Data Array

The Data array must be a variant array containing the current values of the properties. You can use either method of array declaration to create this array:

Data=Array("StringVal", 123, 5.0, TRUE, 1, "Custom")

Or:

Dim Data

Data[0]="'StringVal"
Data[1]=123
Data[2]=5.0
Data[3]=TRUE
Data[4]=1
Data[5]="Custom"

Options Array

The options array holds the options for each of the properties. The meaning of each element varies between different data types.

- **Element 0** – Must be a one character long string that specifies the data type for this property. Valid values are:
  - **"S"** – String property
  - **"I"** – Integer Property

    - **"N"** – Single precision floating point property
    - **"O"** – Boolean property
    - **"L"** – A list of strings. The user can select one of the in a drop-down list.
    - **"C"** – A custom property. This options can be used to display another dialog box to edit advanced properties.
- **Element 1** – Specifies which element in the Data array (0-5) corresponds to this option.
- **Element 2** – The name of the property as it will be displayed in the dialog box.
- **Element 3 & 4**– The meaning of these elements vary between data type.
    - **String Properties** – The minimum and maximum number of characters allowed.
    - **Integer and Single Properties** – The minimum and maximum values allowed.
    - **Boolean, List and Custom** – Ignored.
- **Element 5** – Read only flag. If TRUE, the user will not be able to change the value.
- **Element 6** – The Hint or description of the property to display in the dialog box when it is selected.
- **Element 7 and up** – (List type only) The list of selectable items.

Example:

```
Dim Data
Dim Options(5)

Data =Array("StringVal",123,5.0,TRUE,1,"Custom")
Options(0)=Array("S",0,"MyString",2,10,FALSE,"String Hint")
Options(1)=Array("I",1,"MyInteger",10,300,FALSE,"Integer Hint")
Options(2)=Array("N",2,"MySingle",100,3000,FALSE,"Single Hint")
Options(3)=Array("O",3,"MyBoolean",0,0,FALSE,"Boolean Hint")
Options(4)=Array("L",4,"MyList",0,0,FALSE,"List Hint","Item1","Item2","Item3","Item4")
Options(5)=Array("C",5,"MyCustom",0,0,TRUE,"Custom Hint")
VarEditAsObject MyForm, Data,Options,"My Title",TRUE,TRUE,TRUE
```

This example will display the following dialog box:

Option Events

Four events are supplied for working with the options dialog box. Note that these are events of the form itself. Access them by left clicking on the background of the Custom Form and selecting them from the Events tab of the Object Inspector.

Sub AppOptionHelp(HelpIndex)

This event is fired when the Help button is clicked. HelpIndex is the property number for which the user is requesting help.

Sub AppOptionChange (Data, OptionIndex, Value, AllowChange)

This event is fired when the user changes any of the properties.

The parameters are as follows:

Data - The variant array currently being edited.
OptionIndex - The property number the user is attempting to change.
Value – The new value the user is attempting to enter.
AllowChange – Set to FALSE to prevent the change. The default is TRUE.

Sub AppOptionEdit(OptionIndex, Value)

This event is fired when the user clicks the "…" button of a custom property. OptionIndex is the property number the user is attempting to change. Value is the current value of the custom property.

Sub AppOptionsApply(Data)

This event is fired when the user clicks the Apply button. Data is the variant array currently being edited.

VarIsArray (va:Variant) :Boolean

Returns TRUE if the specified variant is an array. Use VarDataType to determine what type of array it contains.

VarIsByteArray (va:Variant) :Boolean

Returns TRUE if the specified variant is an array of bytes.

VarIsValid (va:Variant) :Boolean

Returns TRUE if the specified variant is valid (i.e. does not contain NULL or EMPTY.)

The following functions are used to pass parameters between the communications drivers and Custom Forms.

VarBoolGet (VArray:Variant, Name:String, Val:Boolean) :Boolean
VarBoolSet (Name:String, Val:Boolean) :String

VarFloatGet (VArray:Variant, Name:String, Val:Double) :Boolean
VarFloatSet (Name:String, Val:Double) :String

VarIntGet (VArray:Variant, Name:String, Val:Integer) :Boolean
VarIntSet (Name:String, Val:Integer) :String

VarStringGet (VArray:Variant, Name:String, Val:String) :Boolean
VarStringSet (Name:String, Val:String) :String

The driver expects variant arrays in the following format:

VA[0] = "COMMAND=SOMECOMMAND"
VA[1]="SOMESTRINGPARM=MYSTRING"
VA[2]="SOMEBOOLEANPARM=TRUE"
VA[3]="SOMEFLOATPARM=98.34"
VA[4]="SOMEINTEGERPARM=169"

These functions simplify setting and reading the parameters.

Example:

```
Dim VA

VA[0]="COMMAND=SOMECOMMAND"
VA[1]=VarFloatSet("VALUE", Meter1.Position) 'VA[1]="VALUE=75.74"
VA[2]=VarBoolSet("STATE", CheckBox1.Checked)  'VA[2]="STATE=TRUE"
VA[3]=VarIntSet("SELECTION",ListBox1.ItemIndex) 'VA[3]="SELECTION=14"
```

The same format is used when data is returned from the driver. The "Get" functions will search the variant array for the specified name and store the value in "Val". The functions return TRUE if the name was found.

Example:

```
Dim ReturnedValue
Dim ReturnedState
Dim ReturnedSelection

'VA is a variant array returned from the driver
If VarStringGet(VA,"COMMAND","SOMECOMMAND") then
  'We know this is the correct packet so get the expected parameters
  VarFloatGet(VA,"VALUE",ReturnedValue)
  VarBoolGet(VA,"STATE",ReturnedState)
  VarIntGet(VA,"SELECTION",ReturnedSelection)
End If
```

Miscellaneous Functions

AlarmMsgSend (Site:String, Text:String, PrintIt:Boolean, FileIt:Boolean, ShowIt:Boolean)

Manually registers an alarm message.

The parameters are as follows:

- **Site** – The description of the Site the alarm should be registered to.
- **Text** – The text of the alarm message.
- **PrintIt** – If TRUE, the alarm will be printed to the alarm printer.
- **FileIt** – If TRUE, the alarm will be stored in the alarm database.
- **ShowIt** – If TRUE, the number of alarms on the status bar will be raised.

Example: AlarmMsgSend("Central College", "An Alarm has occurred.",TRUE,TRUE,TRUE)

EmailSend (ToAddress:String, Subject:String, Body:TStringList) :Boolean

Sends and Email to a single recipient. Returns TRUE if the Email was successfully sent.

Example:

```
Dim Txt

Txt=TStringList.Create
Txt.Add("Email message line 1.")
Txt.Add("Email message line 2.")
Txt.Add("'Email message line 3.")
EmailSend "joe@somewhere.com", "Test Email Message",Txt
Txt.Free
```

Note: The Email options must be set up prior to using this function. See "Program Options" for more information. An Internet connection must be maintained or available for connection at all times.

InformationMsgSend (Site:String, Text:String, PrintIt:Boolean, FileIt:Boolean, ShowIt:Boolean)

Manually registers an informational message (Low priority alarm.) See AlarmMsgSend for more information.

LogWriteLine (Text:String)

Writes a line to the monthly log files. Logs must be enabled for the line to be written (see Program Options.) The line is time and date stamped and written to the log file for the current month.

MessageSet (Text:String)

Sets the text of the Message panel in the status bar of the main form.

Example: MessageSet “Here is an important message the user should see.”

SystemFormCount :Integer

Returns the number of MDI forms currently open. This includes all Custom Forms, Schedules, Graphic Displays and Trends.

WarningMsgSend (Site:String, Text:String, PrintIt:Boolean, FileIt:Boolean, ShowIt:Boolean)

Manually registers a warning message (medium priority alarm) See AlarmMsgSend for more information.

Custom Forms and Scheduled Events

When a Custom Form is executed from a Scheduled Event, it is very important that the script code never does anything that requires user interaction. If the form is waiting for the user to press the “OK” button of an error message, the program will effectively be locked up until someone presses the button.

To avoid this problem, each form has a global variable called **AutoMode**. Setting AutoMode to TRUE (the default is FALSE) affects the following functions:

AskUser (Prompt:String, DefaultAnswer:Boolean) :Boolean

If AutoMode=FALSE this function will display the Prompt in a standard dialog box with “Yes” and “No” buttons. It returns TRUE if the user clicks the “Yes” button and FALSE if the user clicks the “NO” button. DefaultAnswer is ignored.

If AutoMode = TRUE this function returns the DefaultAnswer. No dialog box is displayed.

Example:

```
If AskUser(“That file already exists…overwrite it?”,FALSE)=FALSE then CloseForm
‘If AutoMode = FALSE, the user will be asked the question.
‘If AutoMode = TRUE, the result will always be FALSE.
```

ShowError (Text:String)

If AutoMode=FALSE this function displays a standard error message dialog box and waits for the user to press the “OK” button.

If AutoMode is TRUE this function writes the specified text to the Alarm database as a high priority alarm message.

ShowInfo (Text:String)

If AutoMode=FALSE this function displays a standard informational message dialog box and waits for the user to press the “OK” button.

If AutoMode is TRUE this function writes the specified text to the Alarm database as a low priority alarm message (Informational message).

ShowMsg (Text:String)

If AutoMode=FALSE this function displays a standard information message dialog box and waits for the user to press the “OK” button.

If AutoMode is TRUE this function does nothing.

ShowWarning (Text:String)

If AutoMode=FALSE this function displays a standard warnign message dialog box and waits for the user to press the “OK” button.

If AutoMode is TRUE this function writes the specified text to the Alarm database as a medium priority alarm message (Warning message.)

Appendix A –Andover Infinity

This section contains information specific to using Catalyst Pro with the Andover Infinity.

Site Options

The following is a detailed description of the Andover Infinity Site options:

- **AlarmPolling** - Allow this computer to poll the Site for alarms. If enabled, Catalyst Pro will ask the driver once every minute for any alarms it has received. This should normally stay enabled. In a multi-user or networked environment, it can be disabled in all but one users computer to force all alarms to register in that users database.

- **BaudRate** - The baud rate or speed of the COM port connection.

- **ComPort** - The communications port to use. There is a separate selection for modem or direct cable connection on each COM port.

- **ConnectString** - String of characters to send out COM port to establish connection (optional). This would normally be used only with smart switching devices. The following codes may be embedded in the string:

    - **$** - This will force a ¼ second pause.
    - **^** - Makes the next character a “Control” character. For example, ^M transmits a carriage return.
    - # - Makes the next two characters a hexadecimal number. For example, #1B transmits an [Esc] (hexadecimal 1B = ASCII 27.)

    Example: **#1B^{M}Port5^M** will transmit an [Esc], wait ¼ second, transmit a carriage return, wait ¼ second, transmit “Port5” and finally transmit another carriage return.

    Note: These codes may be used with all of the “string” Site options.

- **DeviceType** - The type of devices the system contains (read only). This is included in the options for reference only. It cannot be changed once the Site has been created.

- **DisconnectString** - String of characters to send out COM port to break connection (optional). This would normally be used only with smart switching devices.

- **DisplayForm** - Specifies a form to display after connecting. This can be used to display a main floor graphic or other top-level display for simplifying user navigation through the program.

- **LogonCode** - The logon code used to log in to the system. This should be set to the logon code of the panel that the computer or modem is physically connected to.

- **ModemInitialization** - String of characters to send out to modem before attempting to dial (optional). If left blank, “ATV1Q0&C1&D2” is used as a default. This should work for most modems.

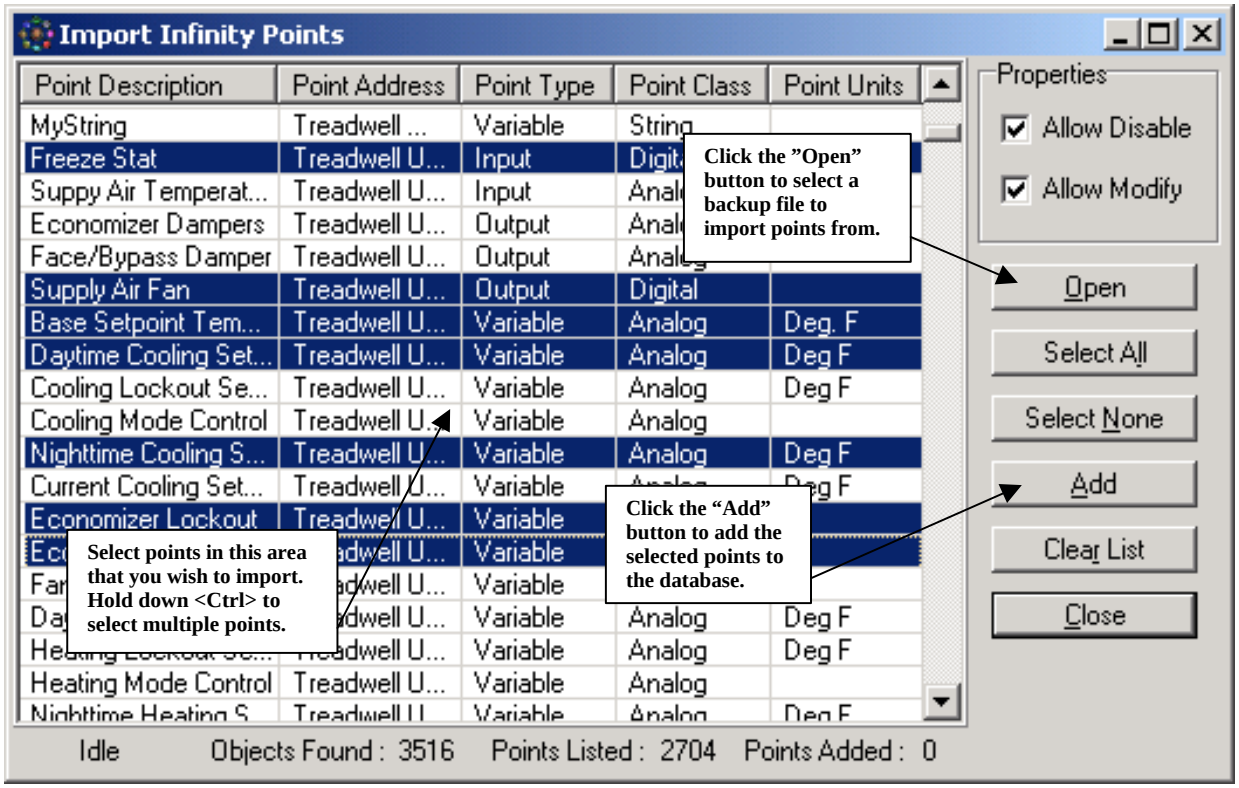
- **Password** - The password used to log in to the system. This should be set to the logon code of the panel that the computer or modem is physically connected to.

- **PhoneNumber** - The phone number to dial if using a modem.
- **PostLogonString** - String of characters to send out COM port after logging on (optional). This can be used to execute a program on the panel after the logon process is complete.
- **PreLogoffString** - String of characters to send out COM port before logging off (optional). This can be used to execute a program on the panel just before the logoff process begins.
- **RemoteName** - The name, URL or IP Address of the PC the driver resides on (optional). This is only used for remote access and should normally be left blank.

Importing Control Points

Catalyst Pro provides a complete point importing feature for the Andover Infinity. Right click in the Control Point Database and select "Import Points" to access this feature.

✪ The import feature requires a backup file of the panel or Site from which you wish to import points. Use an existing backup file or make a new backup using the built-in "Backup Memory" function.



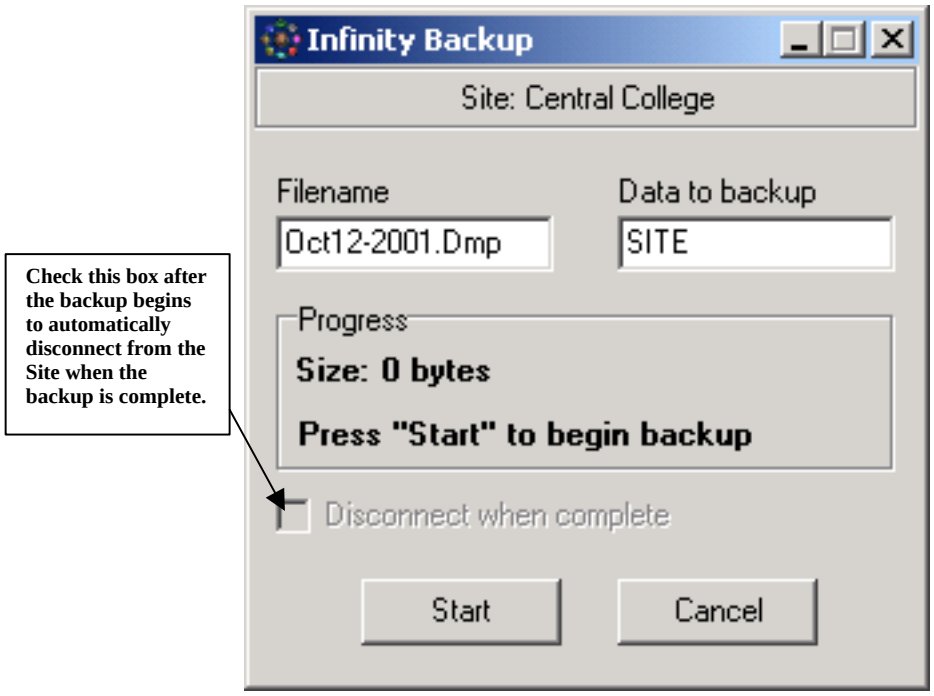
🕒 If you wish to import all of the points at once, click the "Select All" button, then click the "Add" button.

✪ While there is no limit on the number of points in the Control Point database, adding unneeded points will make it harder to find the point you want when browsing the database and it also wastes memory. This can become a real issue if you plan to connect to several Sites at once.

Backup Memory

The Backup Memory feature is used to make a copy of a panels or Sites memory on disk. The Restore Memory feature can be used to reload the memory in case of a major power outage or other problem.

To backup a Site to disk, select “Backup” from the “Site” menu. You should see a form similar to this:



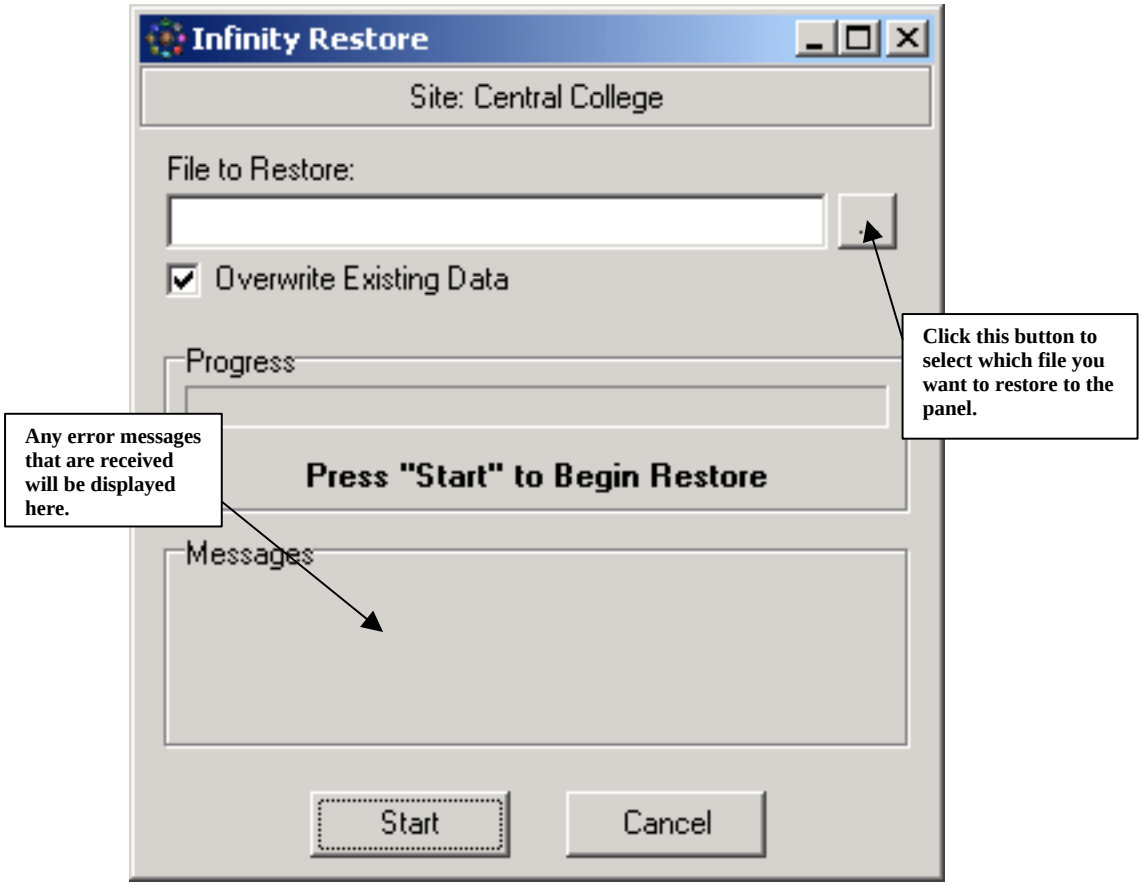
To backup the entire Site, just click the “Start” button. The default filename and data settings are appropriate for most users.

If you wish to backup a single panel or program rather than the entire Site, change the “Data to Backup”. It has the same format as the Infinity’s “Save” command.

Restore Memory

The Restore Memory feature is used to restore the memory of a panel or an entire Site after a major power outage or other problem.

To restore a Sites memory, select “Restore” from the “Site” menu. You should see a form similar to this:

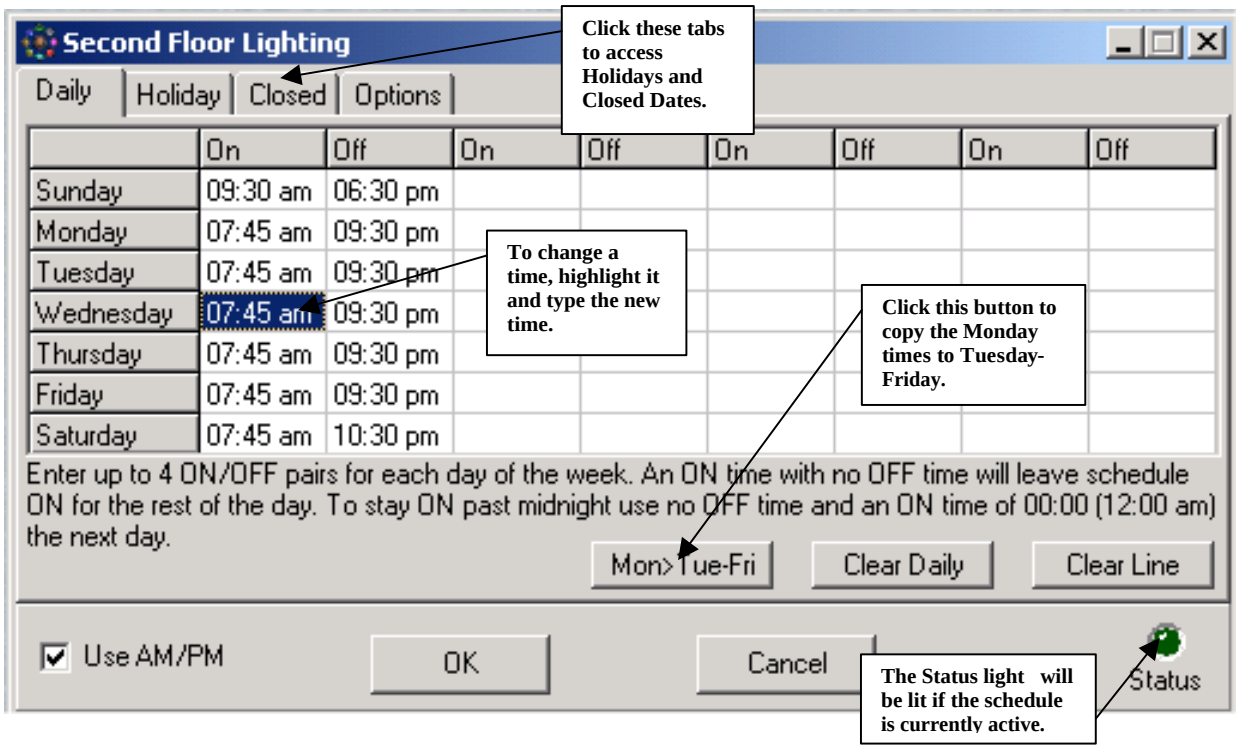


To restore a backup file to a panel or Site, select the file by clicking the “…” button and then click the “Start” button.

As the backup progresses, any error messages received from the Site are displayed in the “Messages” area. These messages are all generated by the panels themselves and some messages may require action on your part. See the documentation that came with your equipment if you need more information.

Changing Schedules

Infinity Schedules allow for four pairs of ON/OFF times for each day of the week, twelve special holiday periods and eight closed dates.



To change a time or date, highlight it and type the new value. Times may be entered in either 24-hour format or am/pm format.

To enter Holiday periods or closed dates, click the tabs at the top of the form. Dates must be entered as MM/DD/YYYY or MM/DD/YY.

When you are finished making changes to the Schedule, click the OK button and the changes will be sent to the panel. It may take a few moments to save the new schedule data.

✪ The information on the Options page is set up by the programmers of the Site. It should not be changed unless you are sure it needs to be changed.

Schedule Programming

✪ This section describes the technical details of schedule programming and can be ignored by most users.

Infinity schedule data is stored in the panel in a manual array with 168 elements and a single panel program is used to process all schedules.

This has several advantages over storing the schedule data on disk or using separate panel programs for each schedule:

- The times and dates are stored on the panel and not on disk. This allows multiple users to see the times and dates as they are in the panel and not as they were the last time the schedule was changed on their particular computer.
- The 168-element array takes up less panel memory than individual panel programs.
- The arrays can be created in any panel or sub-panel on the network.
- The manual array is quicker to reload in to the panel when changes are made than panel programs .
- Because of the way the schedules are processed, they are much more flexible as arrays (see below).

The one disadvantage is a brief delay (5 seconds or so) when you view a Schedule because it has to read the manual array from the panel. This slight disadvantage is heavily outweighed by the numerous advantages.

Capabilities

Schedules include 4 pairs of ON/OFF times for each day of the week, 12 holiday periods with 3 pairs of ON/OFF times, 8 separate "closed" dates and optional placeholders for your optimum start/stop variables.

It is important to note that the schedule processing function that is supplied does not process the optimum start/stop variables. If you create an optimum start/stop schedule, there will be places where the user can adjust the parameters of the optimum start/stop but it is up to you to implement it in your panel programs.

Schedule Set-up

Once you have created a Schedule form, you need to go to the options page of the form and enter the panel name and a variable name. The variable name must be a valid, unused Infinity variable name. It is not possible to directly control an output with the Schedule form.

After entering the panel name and variable name, click the "Create Variable" button. This will create the manual array in the panel. This only has to be done once. The Schedule is now ready to be used normally.

Enter some times and dates in to the Schedule form and press "OK". There will be a short delay as the schedule data is written to the panel.

Using Schedules in Infinity Programs

What we have now is a numeric variable; we'll call it SCHD for this example, that is a manual array with 168 elements. Assuming you have loaded the schedule processing function in to the panel (via the "prepare site" menu selection) all you need to do now is put the following line in Infinity program:

SchCalc(SCHD)

The line can be put in the Infinity just about anywhere. You could have a single looping program that processes all of your schedules:

```
SchCalc(SCHD)
SchCalc(Ahu1Sch)
SchCalc(Lights2Sch)
etc...
```

Or you could call the function only when needed:

```
//====================

'Fan 1 Control Program

  if (Zone1 > Setpoint1) and (SchCalc(SCHD)) then
    Fan1=ON
  EndIf

//====================
```

The SchCalc function will determine if the current date and time fall within any of the closed dates, holiday periods or daily schedules and set element zero to ON or OFF. It will also return ON or OFF as the result of the function. Lastly, it will set element 168 to 0,1,2,3 or 4 to indicate which ON/OFF pair is currently active (daily schedules have 4 ON/OFF pairs, holidays have 3 pairs.)

So all of the following are possible uses of the variable:

- Fan1=SchCalc(SCHD) - Not recommended, put possible
- if SchCalc(SCHD) then TurnSomethingON
- SchCalc(SCHD)
- if SCHD[0]=OFF then TurnSomethingOff
- SchCalc(SCHD)
- if SCHD=OFF then TurnSomethingOff 'specifying no element is same as SCHD[0]
- SchCalc(SCHD)
- if SCHD[168]=2 then StartSecondStageCooling 'we're in the second ON/OFF pair

There is no reason at all why you could not create your own customized schedule processing function and use it instead:

```
MyOptStartCalc(SCHD)
```

That is all you normally need to know to effectively use the Schedules.

Schedule Array Format

The following is a description of the manual array format for those who want to customize the function or implement optimum start/stop.

The format of the array is as follows:

- All times are in the format (hour*100)+min (1:30pm = 1330)
- All dates are in the format ((year-1900)*1000)+DayOfYear (Jan. 1st, 2002 = 102001)
- Times that are blank in the Schedule form will be set to 9999.
- Dates that are blank in the Schedule form will be set to 999999.
- Element 0 holds the current state of the schedule (ON/OFF)
- Element 1-56 are the daily times. (Sun. - Sat. On1,Off1,On2,Off2,ont,Off3,On4,Off4)
- Element 57-128 are the holiday times (H1-H12, On1,Off1,On2,Off2,ont,Off3)
- Element 129-152 are holiday dates. (BeginH1, EndH1,BeginH2, EndH2,etc.)
- Element 153-160 are closed dates. (Closed1,Closed2,Closed3, etc.)
- Element 161 is the Optimum Start Minutes
- Element 162 is the Optimum Stop Minutes
- Element 163 is the Optimum Start Setpoint
- Element 164-167 are undefined. You can use them if you wish to customize the Schedule form to include, for example, a night setback setpoint.
- Element 168 holds the ON/OFF pair (0-4) that is currently active.

So if the user moves the Optimum Setpoint slider and clicks "OK", element 163 would contain the value he selected and you can use it like this:

if Zone1<SCHD[163] then goto DoOptStart

Alarm Programming

✪ This section describes the technical details of alarm programming and can be ignored by most users.

Alarm Message Format

Alarms must be generated by the panel. Usually this is done in the panels programming (See below). When the panel detects an alarm condition, it should display a message that contains the following codes:

{A#TEXT<FORM>}

Where:

{ = start of message marker

A,W,I or **a,w,i** = Use A or a for alarm messages, W or w for warning messages and I or i for informational messages. Lowercase letters will not cause the alarm box on the status line to flash or increment the number of alarms. Use this if you only wish to print something to the printer, but don't want to make anyone think there is an alarm.

= Destination of message
- 0 = none (useful for executing a Custom Form without generating a message)
- 1 = printer
- 2 = disk file (add to alarm database)
- 3 = printer and disk file

TEXT = The actual text of the message.

<FORM> = The name of a Custom Form to execute (optional).

} = end of message marker

This example will send an alarm to the printer:

{A1Zone 1 Temperature is Out of Range (78.5 Deg.f)}

This example will print a warning on the printer and store it in the alarm database:

{W3Airhandler #1 has Exceeded 300 Hours of Runtime}

The maximum length of an alarm message is 255 characters. The actual text of the message may be no longer than 120 characters.

Generating Alarm Messages

Programming alarms for the Infinity panel presents a few problems:

- If the panel simply prints the message to the screen, the program will see the alarm up to 8 more times as the alarm message scrolls up the screen.

- If the alarm message is displayed on the status bar, the program will see the alarm every time the screen redraw (<Ctrl>-Z>) command is used (as is done occasionally by the program itself as it reads data.)

- If an Infinity comport is opened in "raw" mode and the messages are printed, it can cause problems with any displays that are being viewed, trend data collection, etc.

The simplest (but not ideal) solution is to create a string variable in the panel named "AlarmMsg" (any name will work) and place a program similar to the following in the panel:

```
Line Checking
        If Length(AlarmMsg) > 0 then
           Print "{";AlarmMsg;"}" to Comm3 StatusBar 'change Comm3 if neccessary
           AlarmMsg=""
           Goto Clearing
        Endif

Line Clearing
        Print "" to Comm3 StatusBar
        Goto Checking
```

Any panel program may then set AlarmMsg = "A2Zone Temperature is Out of Range" and the alarm will be displayed on the status bar and erased on the next panel scan automatically. If this program is set at the end of the firing order, AlarmMessage will only contain a string for a maximum of one scan.

Using this method, any program that prints on the status bar would have to check to make sure an alarm wasn't active before printing:

```
        If Length(AlarmMsg) = 0 then Print "YourInformation" to Comm3 StatusBar
```

If you aren't concerned about getting an occasional duplicate alarm message, you could simply:

```
        Print "|123A2Zone Temperature is Out of Range|125" to Comm3 StatusBar
```

(The |123 and |125 are equivalent to the '{' and '}' characters.)