Reference manual for the configuration tool H-Designer for the H-series

altus

H-Designer Reference Manual

Foreword

This manual is a description of H-Designer, the configuration tool used to create applications for the operator terminals in the H-series.

The manual assumes that the most recent versions of the system program (firmware) and configuration tool are used.

The operator terminal can be connected to many types of automation equipment, such as PLCs, servo and drives. In this document the term “the controller” is used as a general term for the connected equipment.

Contents

1 Installation

The basic H-Designer system requirements are as follows:

– PC - CPU 80586 or higher;

– Memory - 64 MB RAM or more;

– Hardware - 60 MB or more available hard drive space;

– Display - VGA or SVGA. Microsoft Windows with 256 colors or higher, and a resolution of 800×600 or higher.

Since all the programs in the H-Designer suite have been compressed, you need to uncompress and install the software before using it.

The H-Designer software can be run under the following Windows operating systems:

– Windows 95

– Windows 98

– Windows ME

– Windows 2000

– Windows XP

The software is available from the following website or your local distributor.

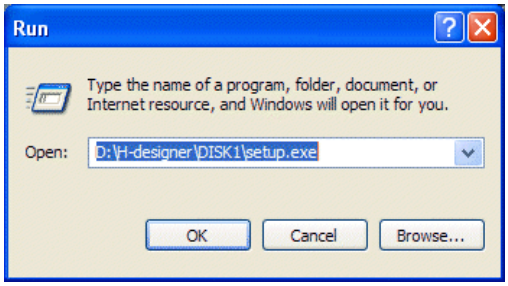
www.beijerelectronics.com

Note also that not all features provided by H-Designer are available for every operator terminal model.

For complete details of these H-Designer features and the applicable models, please refer to *Appendix A - H-Designer Features and Operator Terminal Models*.

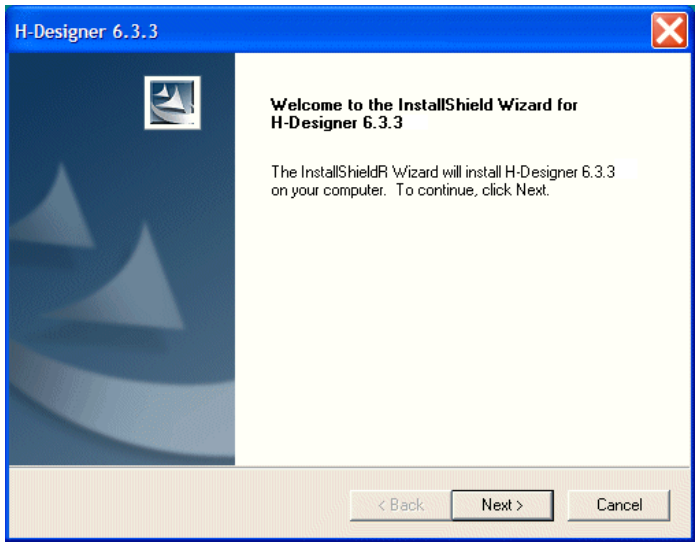
1.1 Installation Procedure

1. Start your computer in the Windows environment.
2. Click **Start** and select **Run**. When the Run dialog box appears on the screen, select **Browse** to locate the installation program **Setup.exe**.



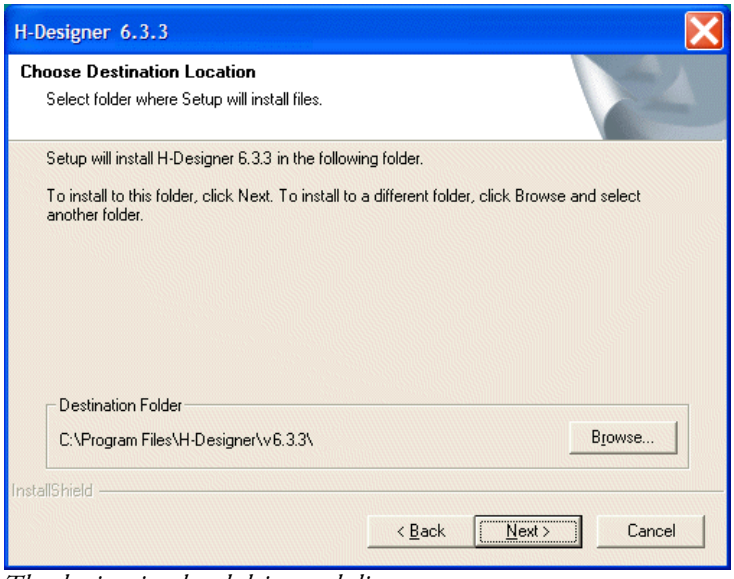
Running the H-Designer installation program "Setup.exe" in Windows

3. Click **OK** to start the installation.



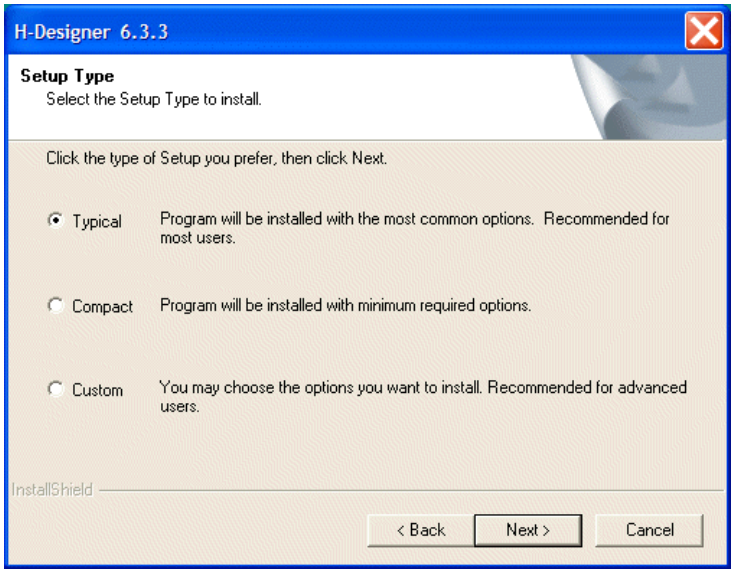
H-Designer is preparing to install

4. Follow the instructions and specify the hard drive and directory where H-Designer is to be installed.



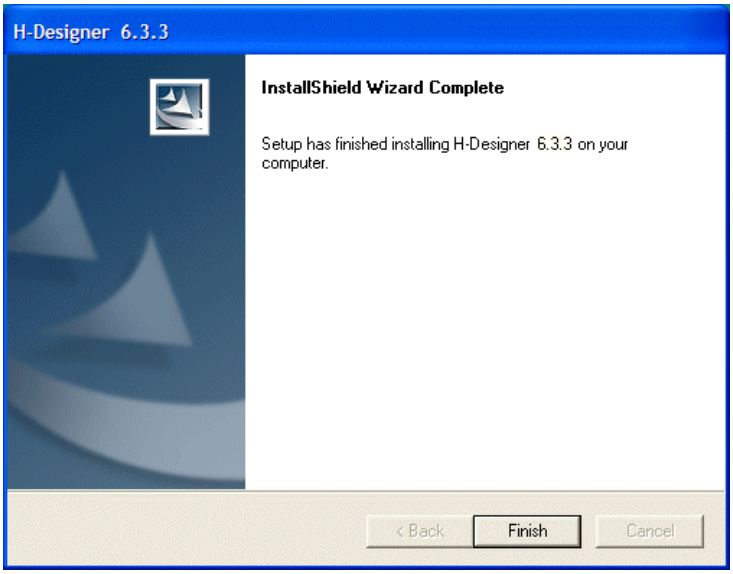
The destination hard drive and directory

5. Click **Next** to select the type of setup. **Typical** is recommended for most users. **Compact** installs the program with basic options. **Custom** allows you to individually select the options to install and this is recommended for advanced users.



Selecting the type of setup

6. Click **Next** to begin the installation. If the **Typical** option is selected, the following dialog box will appear on the screen.



H-Designer installation

7. After installation, the system will create the H-Designer icon automatically.
8. Once the installation is complete, the H-Designer software can be found in the specified directory. To launch the H-Designer program, simply click the H-Designer icon.

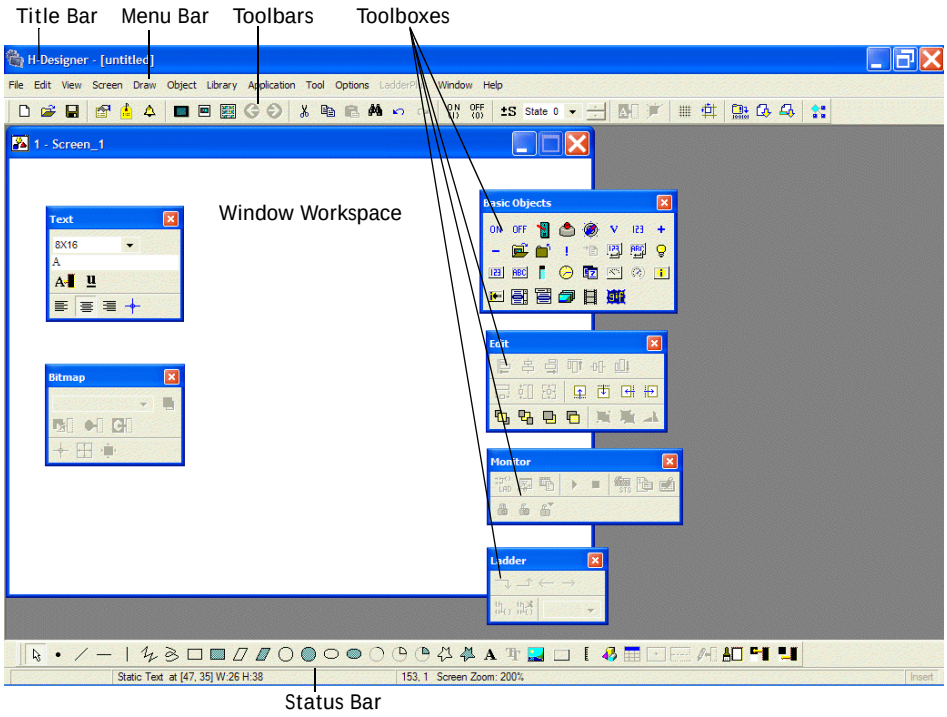
2 Instructions

Microsoft Windows™ is undeniably the predominant PC operating environment these days. H-Designer is designed to make full use of the Windows environment, using a "What You See is What You Get" Approach. You can immediately see the objects you create on a PC screen with their specified attributes, such as font size, color, object location, picture, scale, frame, and so on. What the you see on the PC screen will be the same as what is displayed on a operator terminal.

H-Designer also uses the principles of object-oriented design to implement drag-and-drop editing. You can use a mouse to conveniently drag objects to another location or change their shape and size as you wish.

2.1 H-Designer Programming Environment

The figure *H-Designer program environment overview* illustrates the main menu bars and tool boxes in the H-Designer program environment.



H-Designer program environment overview

Title Bar

The **Title Bar** shows the name of the window and the directory where the current application is found.
For example: **C:\Program Files\H-Designer\Project Files\Project1.V6F**.

If an application file has not been saved, it will be displayed as "Untitled" the Title Bar.

Menu Bar

The Menu Bar consists the following items:

File, **Edit**, **View**, **Screen**, **Draw**, **Object**, **Library**, **Application**, **Tool**, **Options**, **LadderPlus**, **Window**, **Help**.

Tool Bar

You can create an H-Designer project by clicking the icons on the Tool Bar. This feature also helps you to get familiarized with the software quickly and easily.

Window Workspace

This is the area where screens are designed. The objects or data created in this are will be displayed on the operator terminal screen.

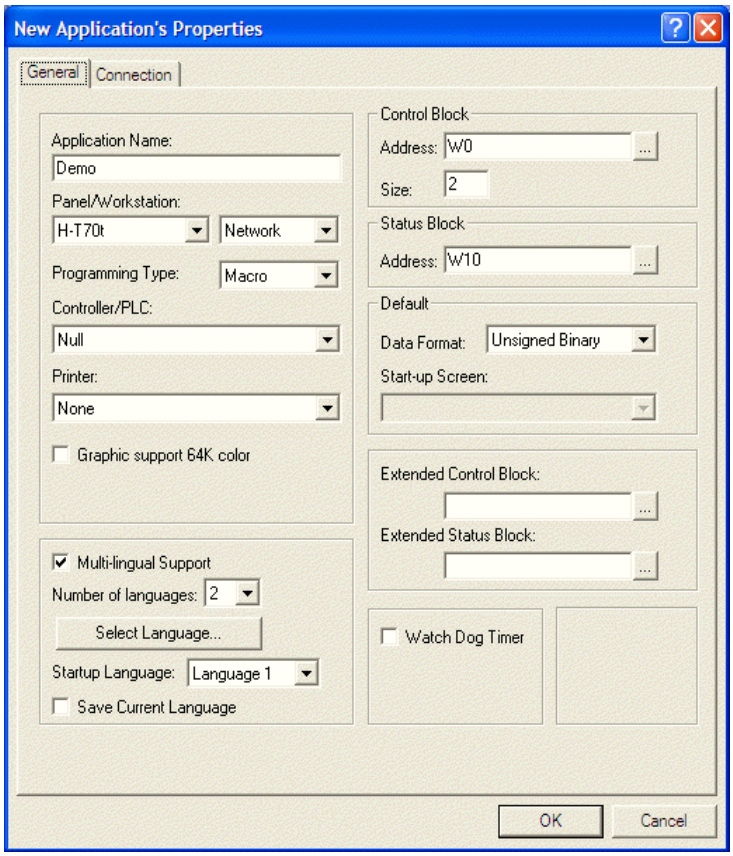
2.2 File Menu

In the **File** menu you can manage the files created by H-Designer.

2.2.1 New

This command allows you to create a new application and specify its properties: **Application Name**, **Panel/Workstation**, **Controller/PLC**, **Printer**, **Multilingual Support**, **Control Block**, **Status Block**.

To create a new application, select **File/New**. The **New Application's Properties** dialog box appears on the screen.



*The **New Application's Properties** dialog box*

The following are the basic properties you must set up for a new application:

– In the **Application Name** box, enter the application name.

– In the **Panel/Workstation** list, select an operator terminal model.

– In the **Controller/PLC** list, select the type of controller with which the operator terminal will communicate.

Please see the sections *Application Menu* and *Connection Tab* for more details.

2.2.2 Open and Close

Open opens an existing application.

Close closes the application.

2.2.3 Save

Selecting **Save** saves an existing application, replacing the previous copy with the new copy.

2.2.4 Save as

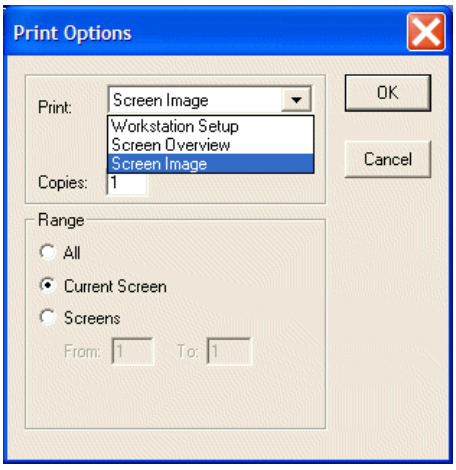
Selecting **Save As** saves a new or existing application with a new name.

2.2.5 Print

An application file can be printed for the planning, management or storage.

Note:
The **Print** option is not available for all operator terminal models: please refer to *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

Select **File/Print** and the **Print Options** dialog box appears on the screen.



*The **Print Options** dialog box*

There are three options available: **Workstation Setup**, **Screen Overview**, and **Screen Image**.

Workstation Setup prints the operator terminal data, such as the controller type, configuration setup, and the logging buffer details.

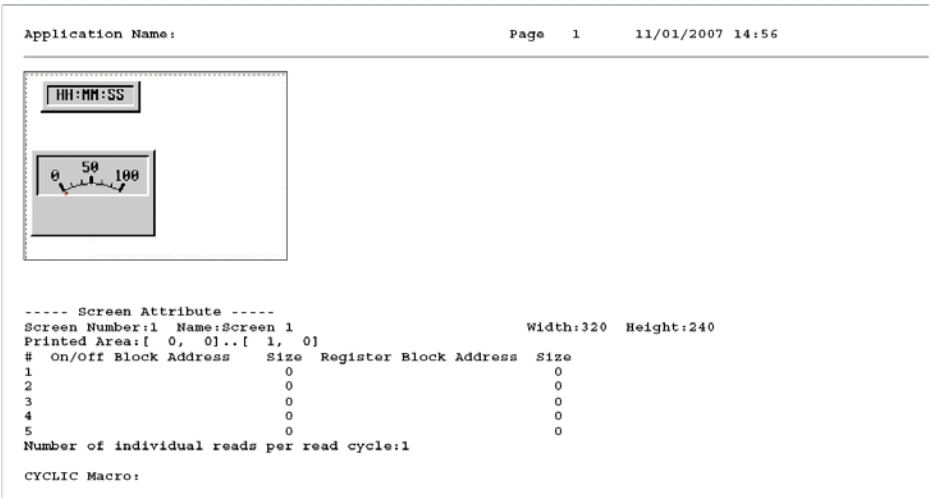
```
Application Name:                                    Page   1      11/01/2005 14:58

Workstation Type:
PLC Type:
Printer Type:None
Default Startup Screen:(None)
Default Data Format:Unsigned Binary
Control Block Address:W0   Size:2
Status Block Address:W10

Logging Buffers
                                 Record      Stamp    Auto Triggered     Time
 #  Source Address             Size Total Time Date Stop     By       Interval
 1  W30                           2  200   No   No   No    Timer          2
 2                                0    0   No   No   No    PLC            0
 3                                0    0   No   No   No    PLC            0
 4                                0    0   No   No   No    PLC            0
 5                                0    0   No   No   No    PLC            0
 6                                0    0   No   No   No    PLC            0
 7                                0    0   No   No   No    PLC            0
 8                                0    0   No   No   No    PLC            0
 9                                0    0   No   No   No    PLC            0
10                                0    0   No   No   No    PLC            0
11                                0    0   No   No   No    PLC            0
12                                0    0   No   No   No    PLC            0
```

*An example of the **Workstation Setup** print option*

Screen Overview prints a screen image with the controller location of each object.



*An example of the **Screen Overview** print option*

Screen Image prints a screen image without the controller locations.

*An example of the **Screen Image** print option*

Other options:

Copies: Specifies the number of copies to print.

Range: Only available for the **Screen Overview** and **Screen Image** options.

2.2.6 Upload Application

Upload Application uploads an application from a operator terminal to a PC and save the file as *.AF6.

2.2.7 Download Application From

Download Application From downloads a program from a PC to a operator terminal. The file format is *.AF6.

2.2.8 Upload Recipes

Upload Recipes is uploads recipes from the operator terminal to a PC. The file will be saved as *.RCP.

2.2.9 Download Recipes

Download Recipes downloads recipes from a PC to a operator terminal. The file format used is *.RCP.

Note:
These functions are not available for all operator terminal models: please refer to *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

2.2.10 Reconstruct Source

Reconstruct Source is used to reconstruct an uploaded application file from *.AA6 to *.V6F. The application will be displayed on a PC and the source file (*.V6F) can be saved for the future editing and application.

In H-Designer, an application downloaded to an operator terminal can be rebuilt; **Reconstruct Source** enables you to directly reconstruct a source file of an uploaded application on an operator terminal from *.AA6 to *.V6F.

Note:
This function is not available for all operator terminal models: please refer to *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

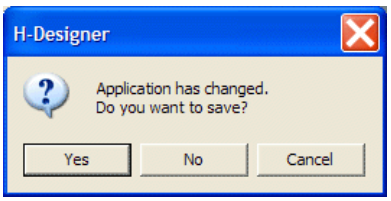
Steps to reconstruct a source file from *.AA6 to *V6F:

1. On the operator terminal, select **Upload Application**. In H-Designer, select **File/Upload Application**. The operator terminal will upload the application to the PC and the file will be saved as *.AF6.
2. Next, select **File/Reconstruct Source** and open the application file (*.C64 or *.AA6). The application appears on the PC and you can save the source file as *.V6F.

2.2.11 Exit

Exit is used to close H-Designer.

If any changes have been made, the following dialog box will appear on the screen, asking you to save the changes or exit.

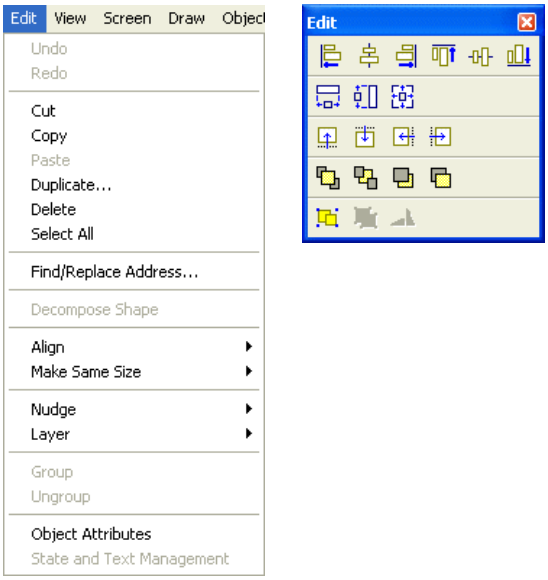


*The **Exit** dialog box*

2.3 Edit Menu

In the **Edit** menu, you can find common editing functions for H-Designer screen and objects.

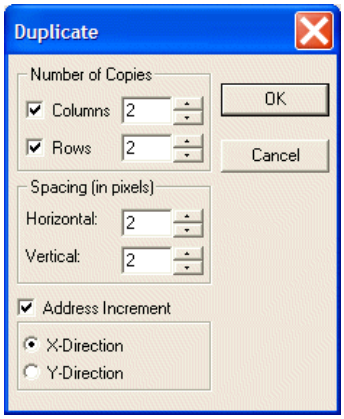
The following sections will explain the functions.



*The **Edit** menu and the **Edit** toolbar*

2.3.1 Duplicate

Allows you to make multiple copies of an object and simultaneously increase the corresponding addresses incrementally.



*The **Duplicate** dialog box*

Number of Copies

Columns: Specifies number of columns to duplicate.

Rows: Specifies the number of rows to duplicate.

Spacing

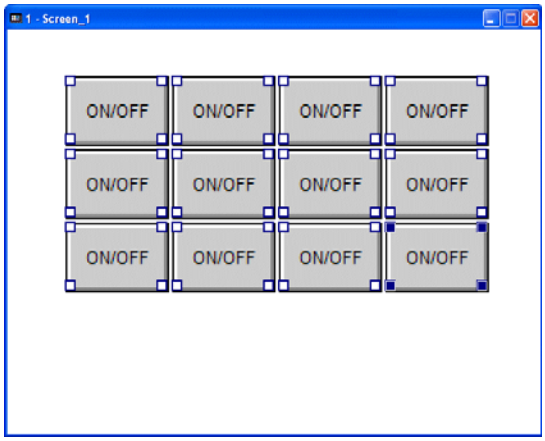
Horizontal: Specifies how many pixels to space duplicate ob-jects horizontally.

Vertical: Specifies how many pixels to space duplicate ob-jects vertically.

Address Increment

X-Direction: The address of the same dynamic objects in-creases from left to right.

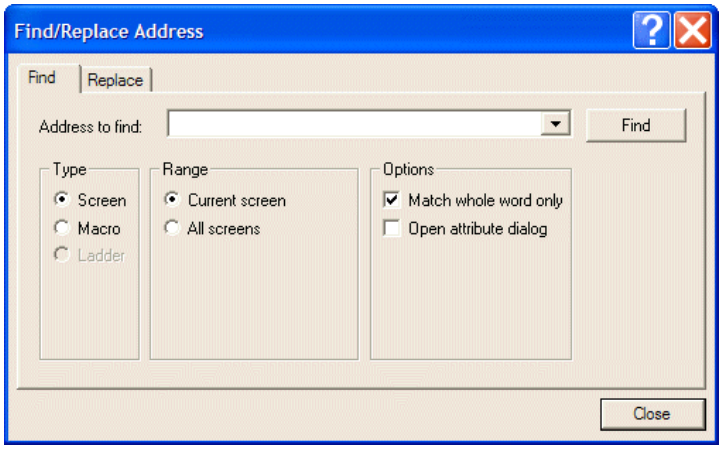
Y-Direction: The address of the same dynamic objects increases from top to bottom.



Duplicate

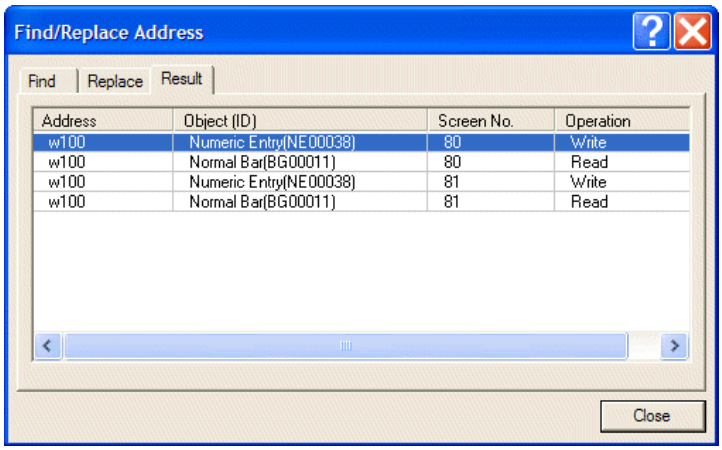
2.3.2 Find/Replace Address

The main function is to find or edit addresses in a created project and to replace the found addresses.



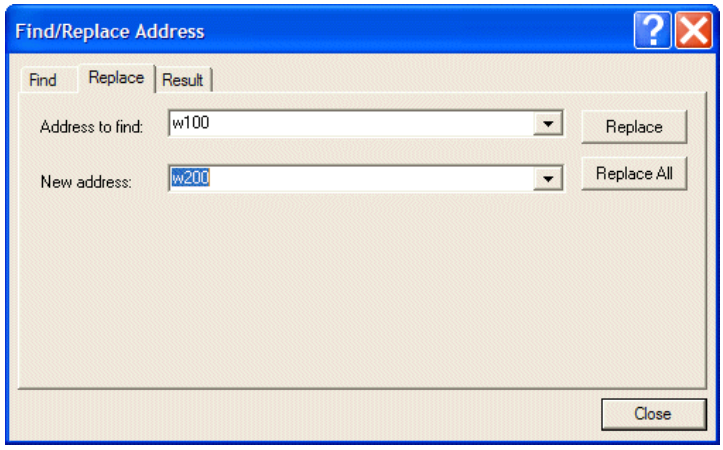
*The **Find/Replace Address** dialog box*

Enter the address in the blank space or select from the drop-down list directly (e.g. W100); then click the **Find** button to search. The **Result** tab displays a detailed list according to the designated range and address.



*The **Result** tab*

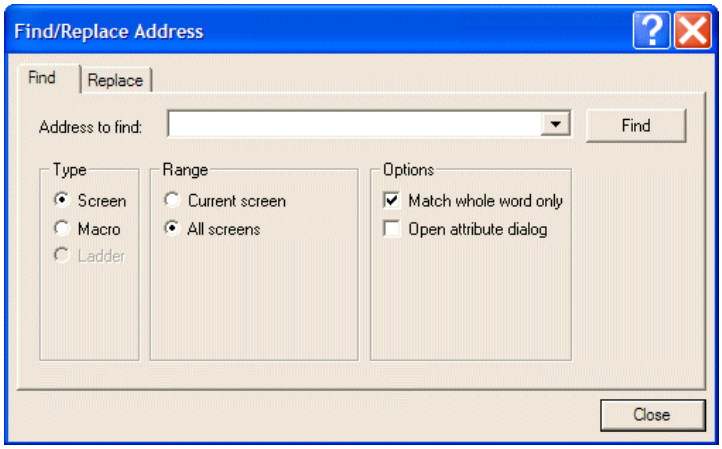
You can use the **Replace** button to replace the address with a new one. Use **Replace All** to replace all of the objects' address at one time.



*The **Replace** tab*

Example 1: Find Screen Address

Select **Type/Screen** to find the screen address.



*Find **Screen** address*

Range:

– **Current screen**: Searches only in the current open screen.

– **All screens**: Finds all the screens in the application program.

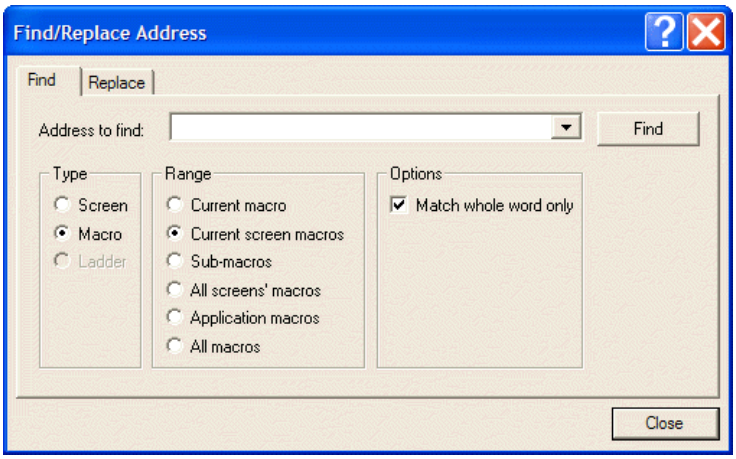
Option:

– **Match whole word only**: The findings match the entry address entirely. If this option is not selected, the initial findings match the entry address including partial and entire matches.

– **Open attribute dialog**: To display an object's attributes, double-click it in the list on the Result tab. If this option is not selected, the object attributes dialog box is unavailable.

Example 2: Find Macro Address

Select **Type/Macro** to search for macro addresses.



*Find **Macro** address*

Range

– **Current macro**: Finds the current macro.

– **Current screen macros**: Finds the current image/screen macros.

– **Sub-macros**: Finds all sub-macros.

– **All screens' macros**: Finds all image/screen macros.

– **Application macros**: Finds three types of macros in the **Application** menu.

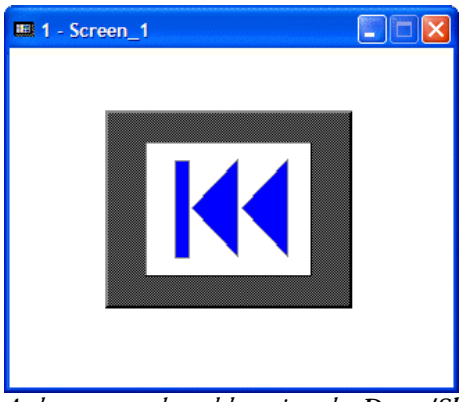
– **All macros**: Finds all macros.

Options

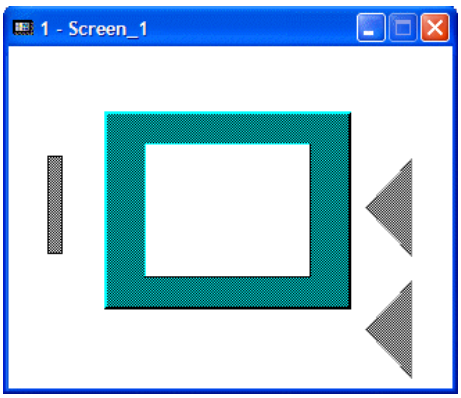
– **Match whole word only**: The findings match the full entry address. If this option is not selected, the initial findings match the entry address including partial and entire matches.

2.3.3 Decompose Shape

This function is primarily used to decompose the graphics created using **Shape** in the **Draw** command. Each decomposed graphic can be modified and edited.



*A shape was selected by using the **Draw/Shape** command*



*Selecting **Edit/Decompose** allows modification of each part of the decomposed object*

2.3.4 Align

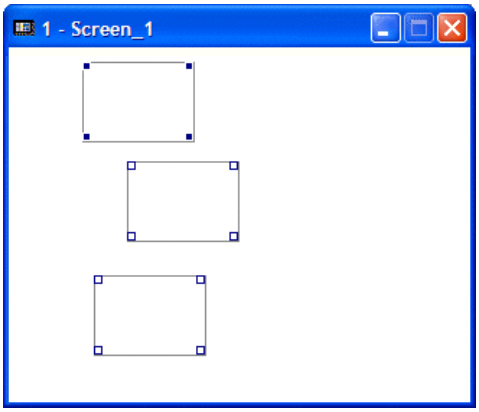
This function can be used to align created objects, for example indicators, moving signs, message displays and lines.

2.3.5 Make Same Size

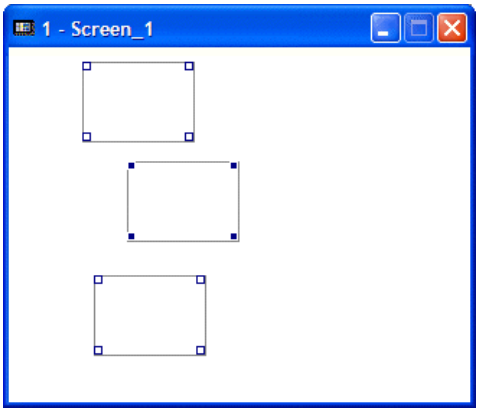
This function can be used to align created objects, for example indicators, moving signs, message displays and lines.

The following are the steps used for **Align** and **Make Same Size**:

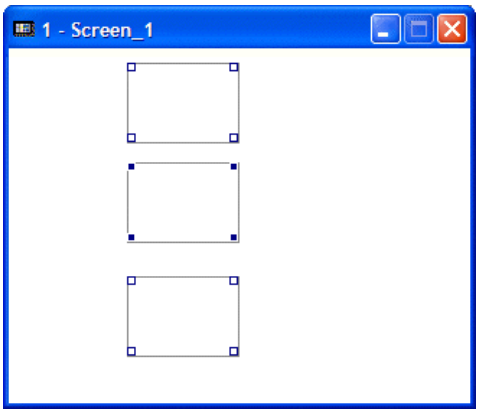
1. Select **Shift** and click on objects which you want to align or make the same size.



2. Click on the master object; the four corners of this master object becomes shaded.



3. Select the command **Align** or **Make Same Size** to make the other objects aligned with or made same size as the master object.

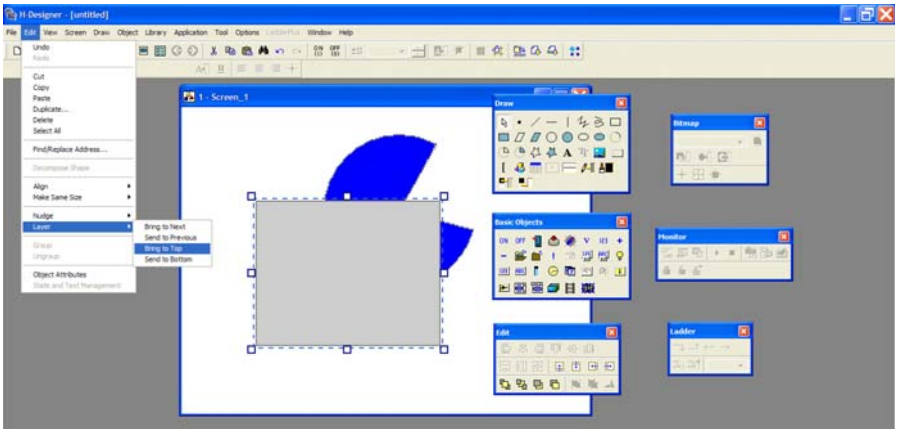


2.3.6 Nudge

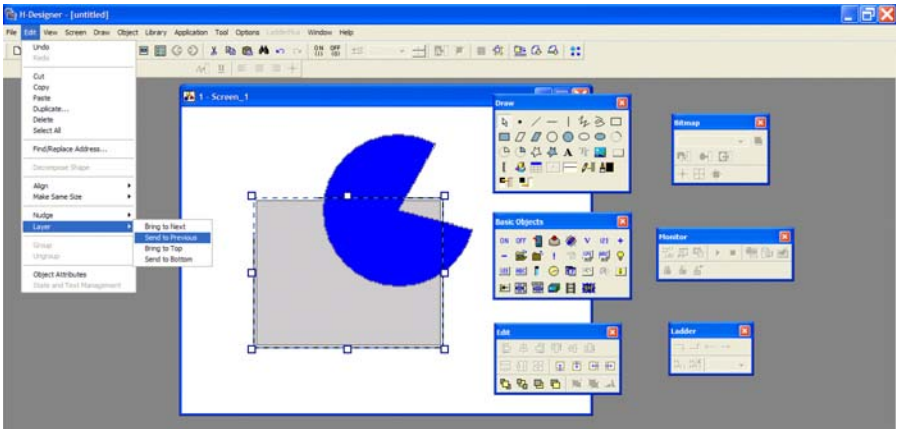
Nudge: Choose objects to slightly shift location and adjust towards the specified direction.

2.3.7 Layer

Layer: If there are more than two objects, you can shift the objects' layer positions up and down.



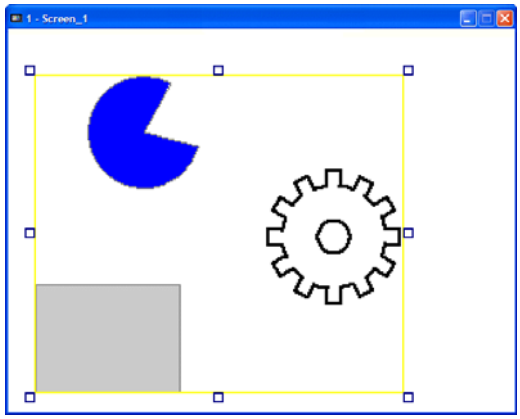
*The **Layer/Bring to Next** command sends the rectangle to the top.*



*The **Layer/Send to Previous** sends the rectangle to the bottom.*

2.3.8 Group

If there are more than two graphics or objects on the screen to be edited; frame (by using **Shift** + left-click) all objects to be grouped and then select **Group**. All of the framed objects will be moved to the appointed position as a single unit.



*Select **Edit/Group** to group the selected objects together.*

2.3.9 Ungroup

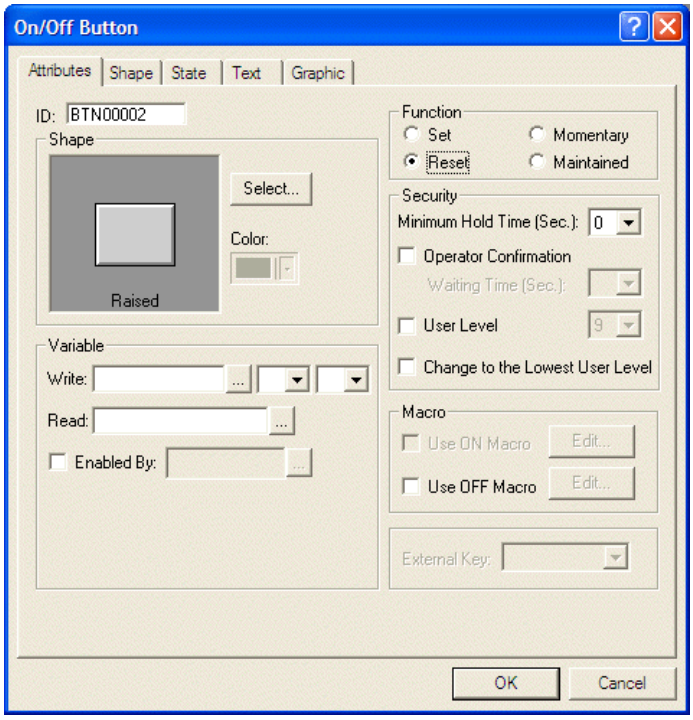
Ungroup is used to ungroup the selected group of objects.

2.3.10 Object Attributes

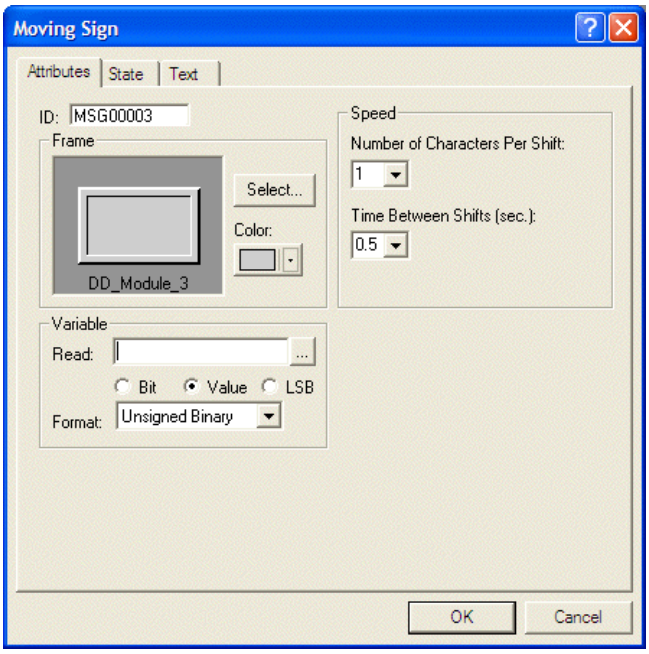
You can edit the content of the objects or modify the data location and formats associated with the controller.

Note that different objects (such as the push button, indicator, moving sign and message display) have their own object attributes.

Click on the object and then select **Edit/Object Attributes** and the dialog box will then be displayed on the screen. You can also double-click directly on the objects. For properties not explained in this section, please refer to section *Object Menu*.



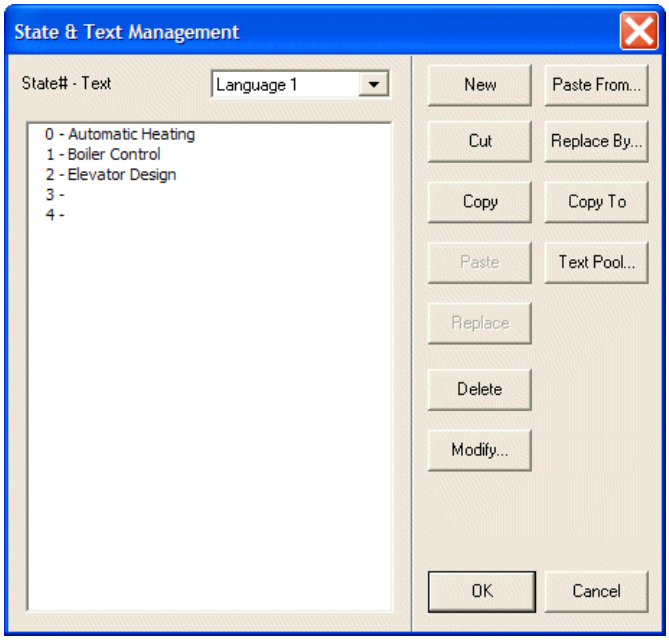
*The **ON/Off Button** object attributes dialog box*



*The **Moving Sign** object attributes dialog box*

2.3.11 State/Text Management

Using **State/Text Management** you can edit the text, colors, and typefaces for the object created. It also provides such functions as copy, modify, line feed.

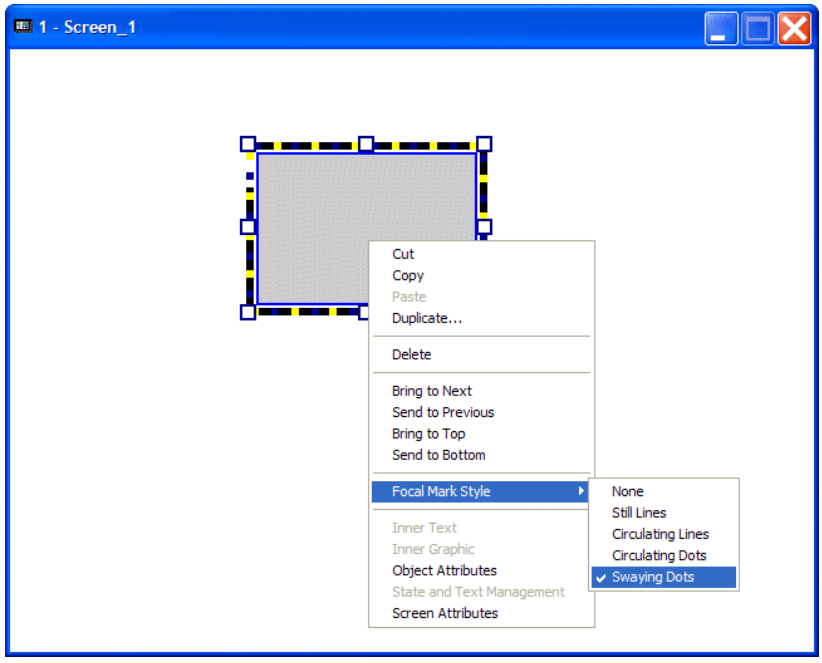


*The **State & Text Management** dialog box*

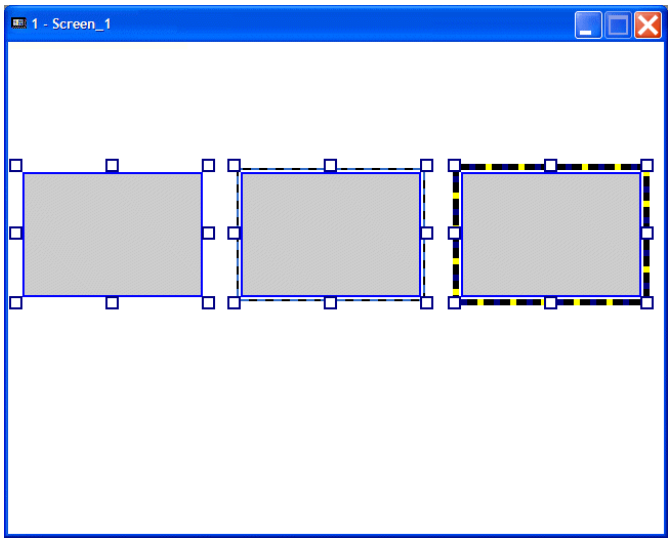
Focal Mark Style

There are five focal mark styles to choose from: **None**, **Still Lines**, **Circulating Lines**, **Circulating Dots** and **Swaying Dots**.

This function is not limited by any application images or files. It modifies the H-Designer editing environment.



*The **Focal Mark Style** list*

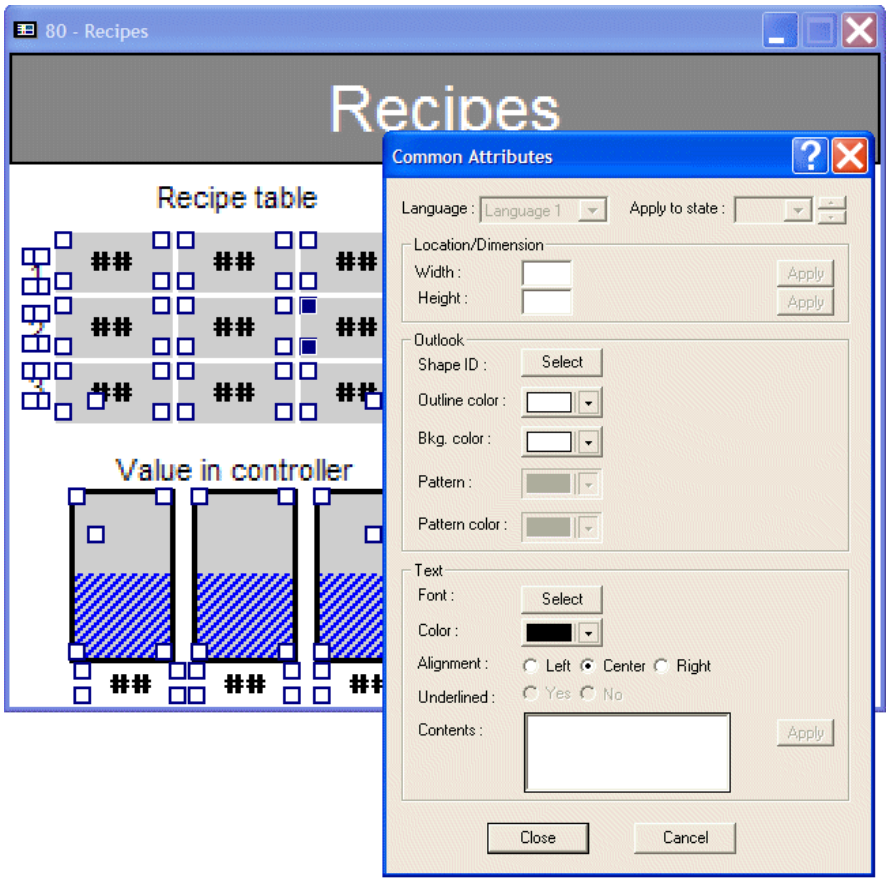


Objects with different focal marks

Common Attributes

This function makes it easier to edit objects with common attributes simultaneously. You can modify attributes of all objects which are on the screen at one time.

First, select the objects to be edited. Double-click one of the objects, and the **Common Attributes** dialog box will be displayed on the screen.



*Opening the **Common Attributes** dialog box*

You can edit the common attributes of selected objects, such as location/dimension, apperance and text.

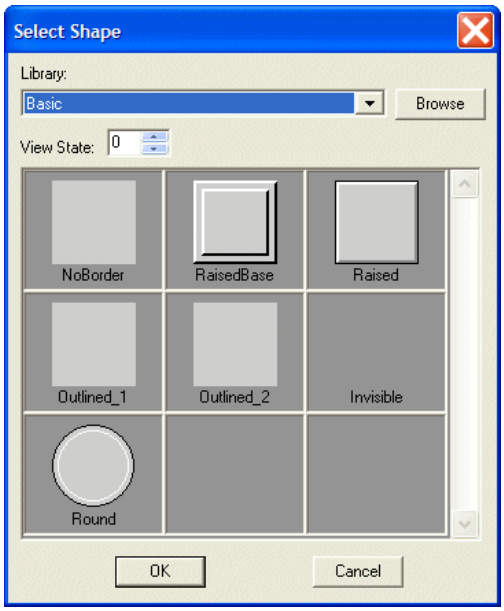
Language
Specifies language to be used for the selected objects.

Apply to State
Specifies the state of the selected objects. You can edit different states in this list.

Location/Dimension
Specifies the width and height of the objects.

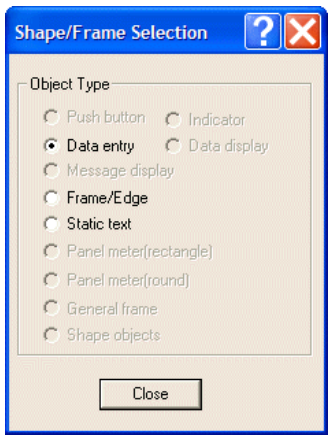
Appearance
Specifies the appearance of selected objects.

Shape ID: Specifies the shape of different objects. When clicking the **Select** button, a dialog box with a selection of shapes will appear on the screen.



Selection of shape of the object

When the selection has been made, a dialog box with a selection of functions for the selected shape will appear on the screen.



Selection of function of the object

You can then select the next object to create.

Outline Color: Specifies the outline color of all selected objects.

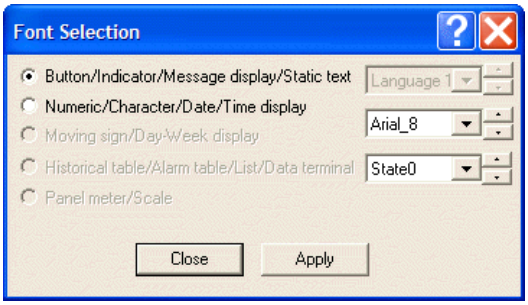
Background Color: Specifies the background color of the selected objects.

Pattern: Specifies the pattern of all selected objects which can be set.

Pattern Color: Specifies the pattern color of all selected objects which can be set.

Text

Font: Specifies the text font for different objects. You can specify the text font for different types of objects. When using the **Selection** button, the **Font Selection** dialog box will be displayed.



Specifies the text font for objects

Color: Specifies the text color for all selected objects.

Alignment: Specifies the text alignment type for all selected objects.

Underlined: Specifies whether the text of all selected objects is to be underlined.

Contents: Specifies the text contents for all selected objects.

Remember to click the **Apply** button to apply changes.

2.4 View Menu

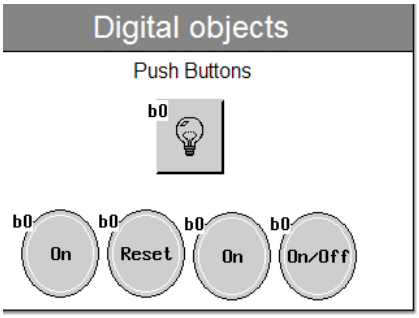
The main functions of the **View** menu include the managing functions of whole-screen, whole screen with I/O labels, language selections, zoom in/out, and toolbars.

2.4.1 Whole Screen

Whole Screen: Full-screen view showing all edited objects. You can return to the previous view by clicking on the screen.

2.4.2 Whole Screen With I/O Labels

Whole Screen With I/O Labels: To view the whole screen with dynamic objects with write/read addresses displayed on the top left side . You can return to the previous view by clicking on the screen.

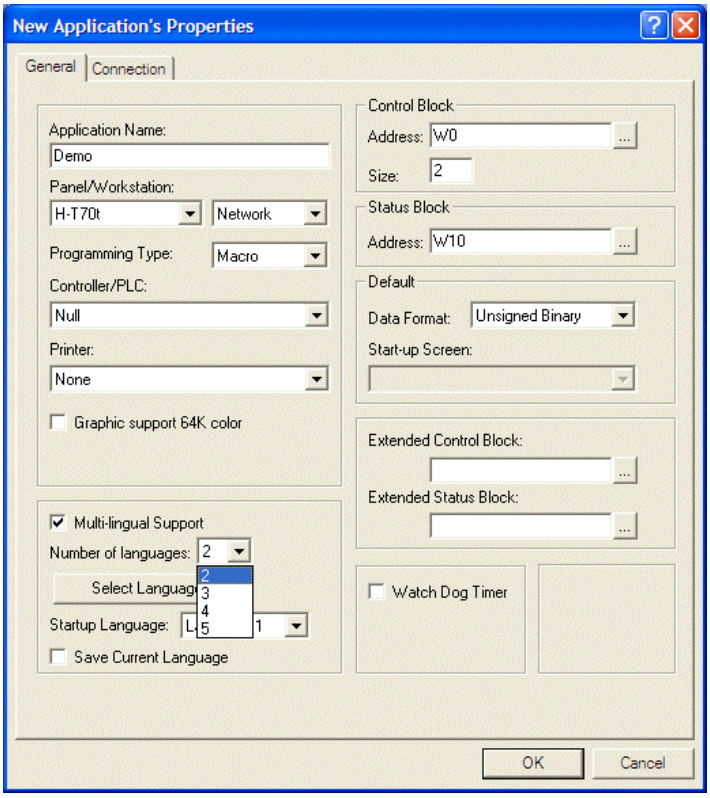


*Displaying **Whole Screen With I/O Labels***

2.4.3 Language 1-5

There are five languages available. Please follow the steps below to set up multiple languages:

1. Select **Application/Workstation Setup**. The **Application Properties** dialog box will appear on the screen.



*The **Application Properties** dialog box*

2. On the **General** tab, check the **Multi-lingual Support** box to begin setup.

   Number of Languages: Specify the number of languages required for the project. A maximum of 5 languages can be selected.

   Select Language: Specifies the languages to use.

   Startup Language: Specifies the language to display at startup. You can select the language directly from the **View** menu if the screen language will be changed later.

2.4.4 Zoom In

Zoom In: To enlarge the image size by a set percentage of 150%, 175%, 200%, or 250%.

2.4.5 Normal Screen

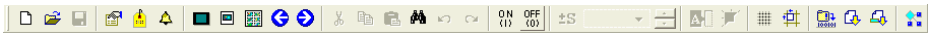
Normal Screen: To adjust the image size to the actual size of the screen.

2.4.6 Zoom Out

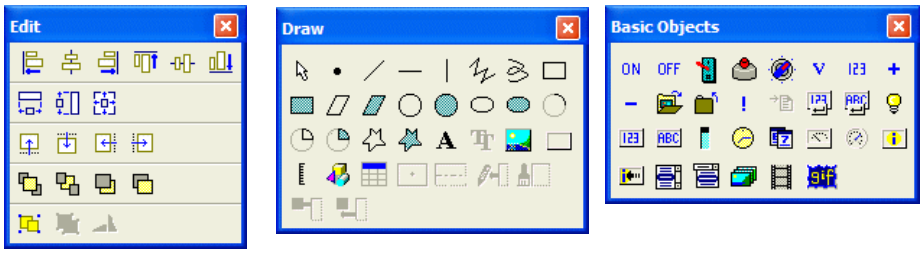
Zoom Out: To decrease the image size by a set percentage of 75%, 50%, or 25%.

2.4.7 Toolbars

There are eight different toolbars in the **View** menu. You can turn on and off the displaying of the toolbars.

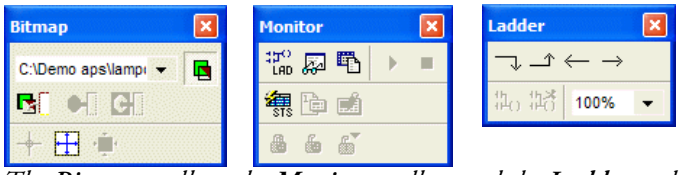


*The **Standard** toolbar*



*The **Edit** toolbar, the **Draw** toolbar and the **Basic Objects** toolbar*

*The **Text** toolbar*



*The **Bitmap** toolbar, the **Monitor** toolbar and the **Ladder** toolbar*

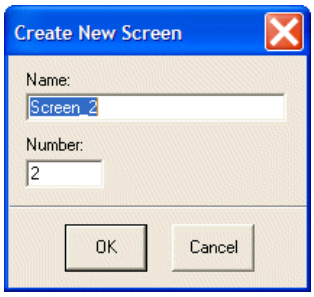
2.5 Screen Menu

The **Screen** menu can be used to name, number, edit, and manage screens. The following sections will explain these commands in detail.

2.5.1 New Screen

To create a new screen select the **New Screen** option.

In the **Name** field, enter the name for the new screen. In the **Number** field, enter the screen number.

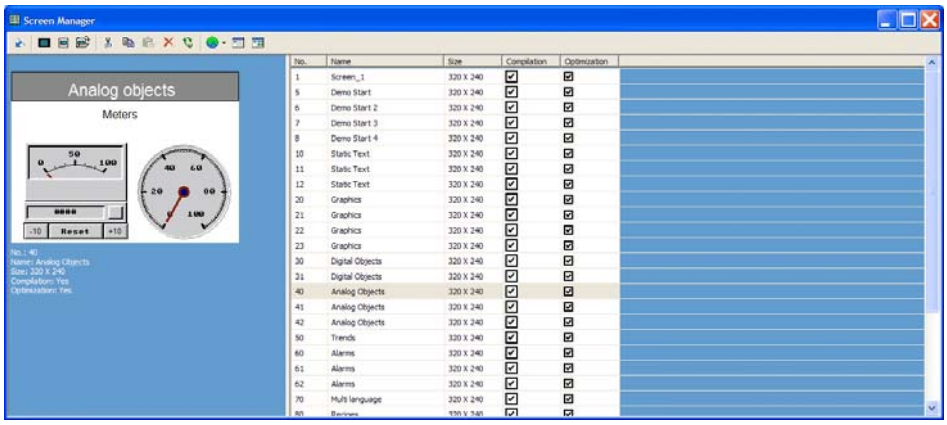


*The **Create New Screen** dialog box*

2.5.2 Screen Manager

The function of the Screen Manager command is to display all the application files in detail view or thumbnail view to make it easy to search, modify, edit, and so on.

Select **Screen/Screen Manager** or click  . The entire Screen Manager will be displayed in the middle of the window or minimized on the left side.



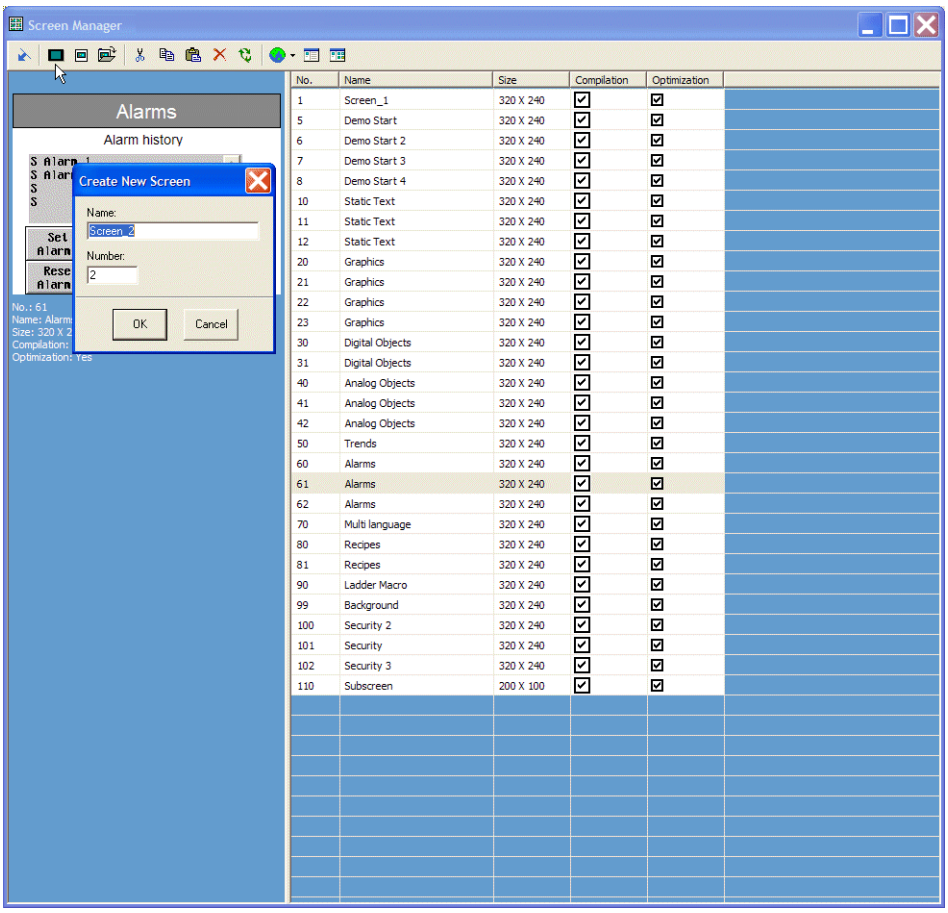
*The **Screen Manager***

In the **Screen Manager** window, click the number or name to display the selected screen. If you want to open the image, just double-click on the image.

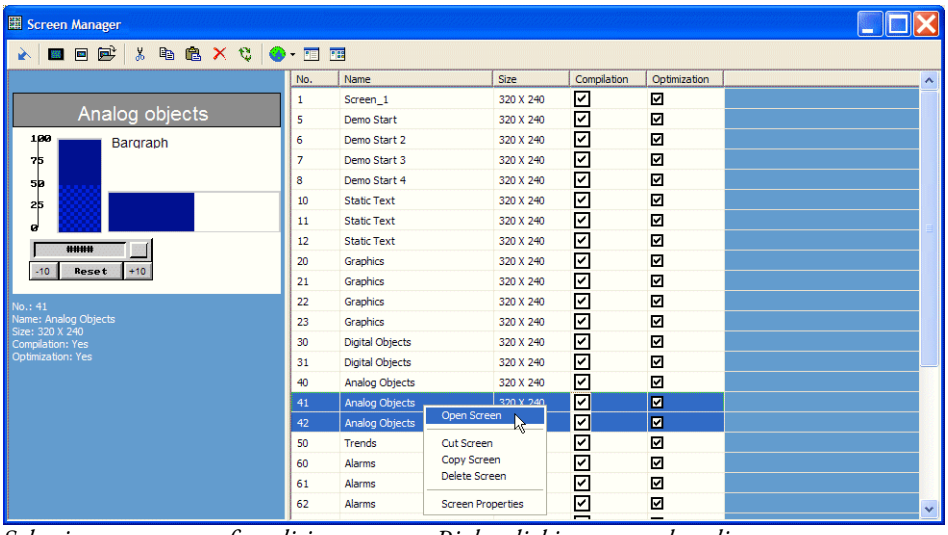
Screen Manager Icons

Screen Manager icons provide functions to manage screens and screen properties, as well as functions to increase ease of use of the **Screen Manager**, as explained below:

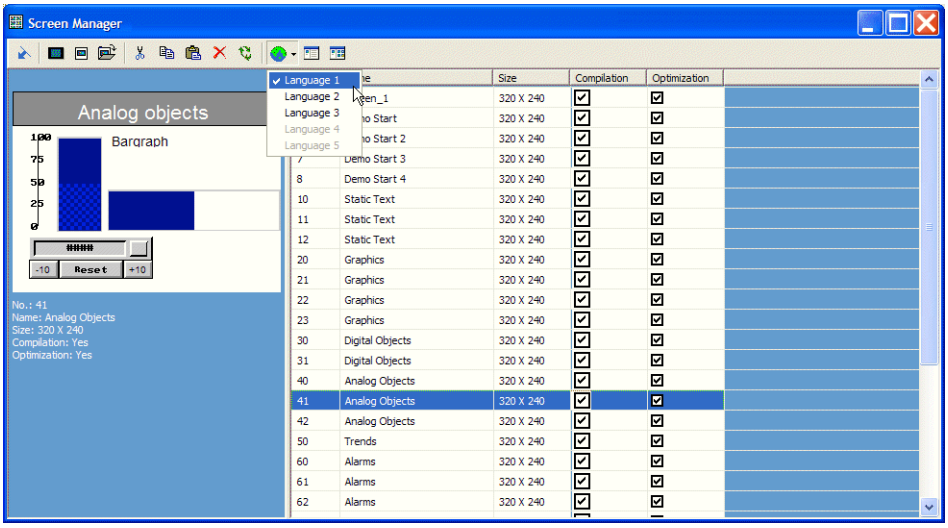
Icon	Name	Function
	Dock	The size of the **Screen Manager** window will be decreased, and the **Screen Manager** will be docked at the left side of the screen.
	Undock	The **Screen Manager** window is restored to full screen size.
	Close	Closes the **Screen Manager**.
	Detail View/ Thumbnail View	Displays screen data in different ways, either as detailed list data, or as thumbnails.
	New Screen	Adds a screen with **No.** and **Name**. A new screen appears in the **Screen Manager** list. Also described in image below.
	Screen Properties	Opens the Screen Properties dialog box. Please refer to section *Screen Properties* for complete details.
	Open Screen	Opens and displays the selected screen while the **Screen Manager** window will be minimized at the bottom. A screen can also be opened by double-clicking on it.
	Cut Screen/ Copy Screen/ Paste Screen/ Delete Screen	Edits the selected screens. A pop-up menu to select editing operations can also be displayed by by right-clicking on the screen. Use the **Ctrl**-key while left-clicking to select several screens for editing at once. Also described in image below.
	View Recycle Bin	Lists all of the deleted screens marked with red. Right-click on the screen list and then select **Restore** from the list. Also described in image below. Click the  again to return to the previous **Screen Manager** window.
	Select Language	Specifies the language displayed in the screen. Also described in image below.



Adding a new screen from the ***Screen Manager****.*



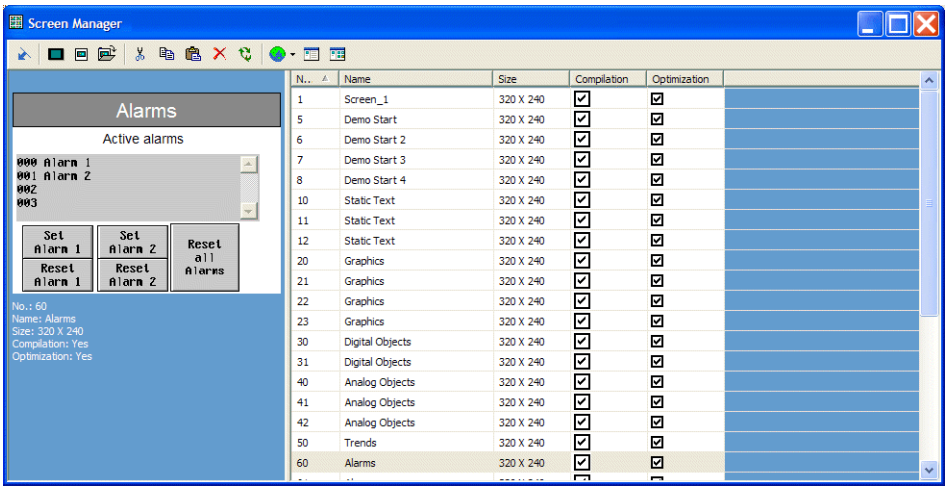
Selecting two screens for editing at once. Right-clicking opens the edit pop-up menu.



*Switching between **Language 1** and **Language 2***

Screen Table

In the **Screen Manager**, click the **Detail View** button. The detailed data will be listed in the table in five columns: **No.**, **Name**, **Size**, **Compilation** and **Optimization**. When you click the header, the data will be listed in increased or decreased order following numerical order, dimensional order, and so on.



*List sorted in numerical order after clicking **No.** in the table header.*

The **Screen Manager** can also help you manage and edit the screens quickly and easily. If the data in these columns is to be modified, you can click the **Screen Properties** button on the toolbar and modify the screen properties in the displayed dialog box. Alternatively, right-click for the **Screen Properties** selection.

The **Compilation** column is used to choose to check compilation or not. This can also be configured in the **Screen Properties** dialog box, on the **General** tab.

The **Optimization** column is used to choose to perform block optimization during compilation. This can also be configured in the **Screen Properties** dialog box, on the **Read Blocks** tab.

Please refer to section *Screen Properties* for complete details.

2.5.3 Close Screen

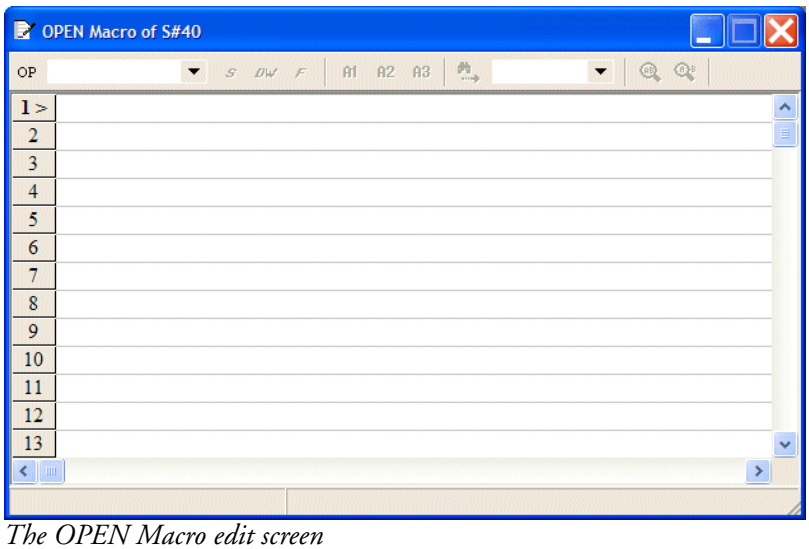
Closes the current screen.

2.5.4 Cut/Copy/Delete Current Screen and Paste Screen

Cuts, copies or deletes the current screen. A cut or copied screen can be pasted onto other screens.

2.5.5 OPEN Macro, CLOSE Macro and CYCLIC Macro

These three macros enable the operator terminal to execute data initialization, display control, and internal register or contact initialization. Once these commands are selected, the operator terminal will display the edit screen.



The OPEN Macro edit screen

OPEN Macro is executed when the screen open command is issued. A screen is not displayed until the **OPEN Macro** is completely executed. There is one **OPEN Macro** per screen.

CLOSE Macro is executed when the screen close command is issued. A screen is not closed until the **CLOSE Macro** is completely executed. There is one **CLOSE Macro** per screen.

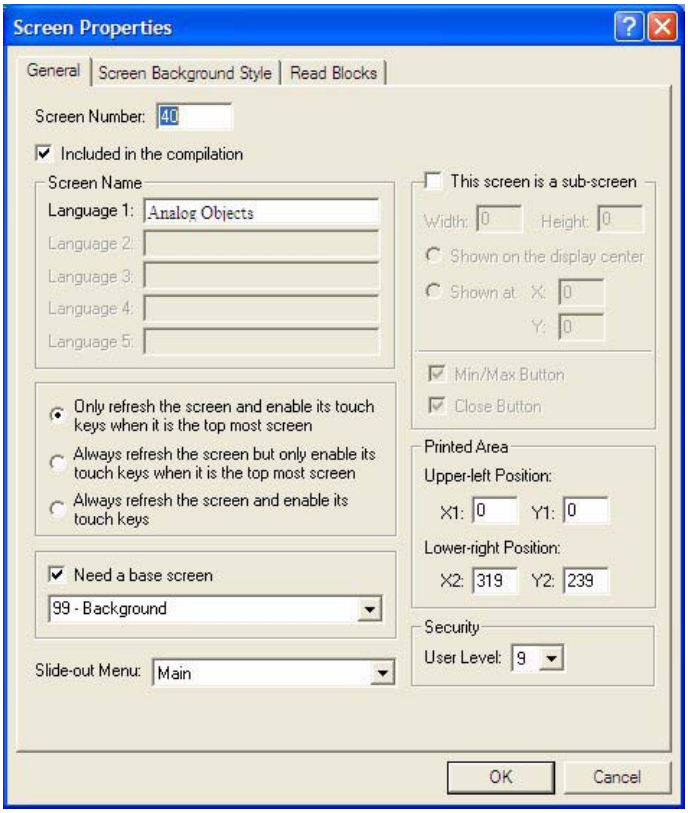
The **CYCLIC Macro** is executed cyclically when the screen is displayed. The operator terminal stops executing the macro when it encounters an **End** command or reaches the end of the macro.

Please refer to chapter *Macros* for complete details.

2.5.6 Screen Properties

The function of the **Screen Properties** command is to display the properties of the current screen including **General**, **Screen Background Style** and **Read Block** tabs.

General Tab



*The **General** tab of the **Screen Properties** dialog box*

Screen Number: Specifies the number of the current screen.

Included in the compilation: Use this option to compile the selected screen.

Screen Name: You can enter the name for a current screen.

Screen Update and Key Function: Specifies the types of screen updates.

Need a base screen: Check this option for a base screen. A base screen can be the background for various screens.

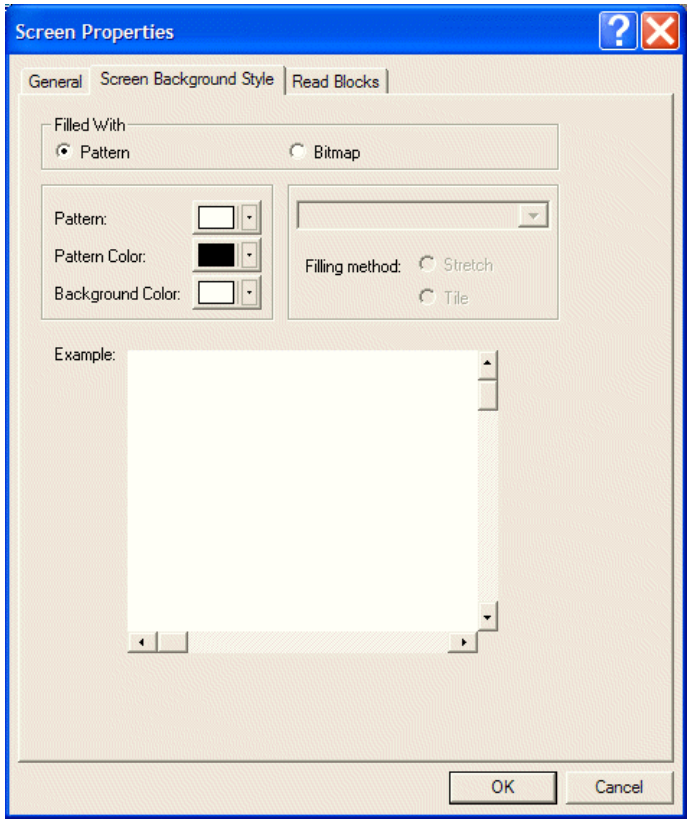
Slide-out Menu: Specifies the slide-out menu of the screen.

This screen is a sub-screen: Check this option to display the selected screen as a sub-screen. You can designate the width, height, location and buttons of a sub-screen in this block. (Maximum width = 180; height = 160). For further information, please refer to section *Sub-screen*.

Printed Area: Specifies the area to print. Note that this feature is available only on models with PRINTER PORT.

Security: Specifies the security level of the screen. A user level between 0 and 9 can be selected. The default value, 9, represents the lowest user level.

Screen Background Style Tab



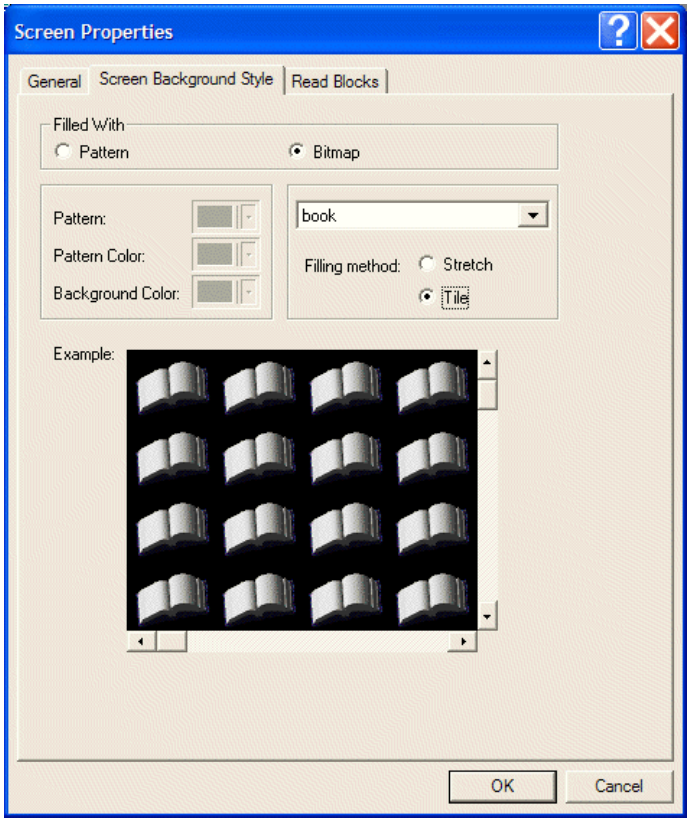
*The **Screen Background Style** tab of the **Screen Properties** dialog box*

Filled With:
When selecting **Pattern**, the options **Pattern**, **Pattern Color** and **Background Color** will be available for selection.

When selecting **Bitmap**, a bitmap can be selected from the drop-down list. There are two filling methods available:

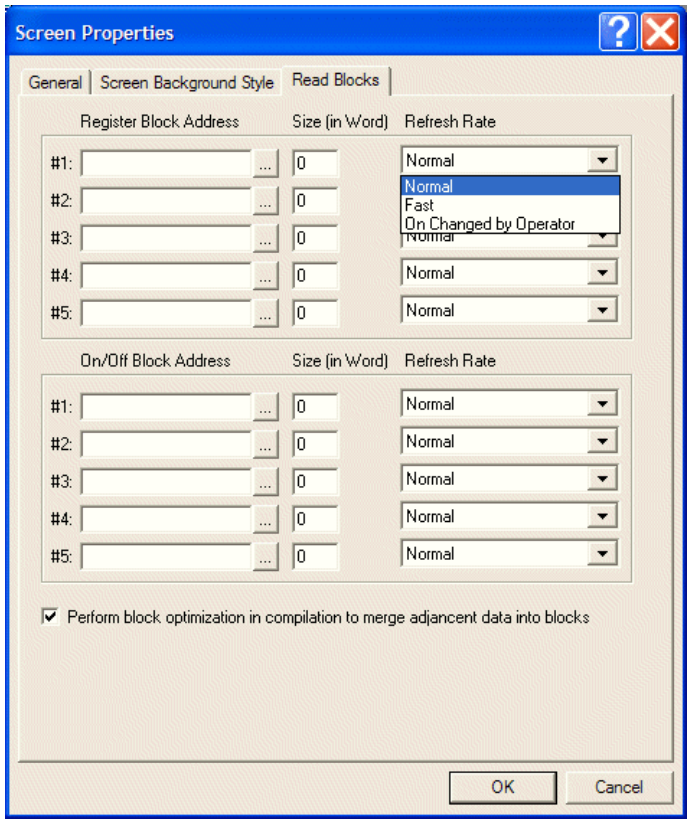
– **Stretch**: Displays the bitmap in full-screen.

– **Tile**: Displays the duplicate bitmaps lined up on the designed screen. Each bitmap keeps its original size.



*Selecting the **Tile** option*

Read Blocks Tab



*The **Read Blocks** tab of the **Screen Properties** dialog box*

The function of the **Read Block** tab is to specify register block addresses, on/off block addresses, size (in words), and refresh rate.

Register Block Address: Specifies the register block address according to the controller. A screen can have a maximum of five specified register addresses.

On/Off Block Address: Specifies the on/off block address. A screen can have a maximum of five specified register addresses.

Size: Specifies the size of block.

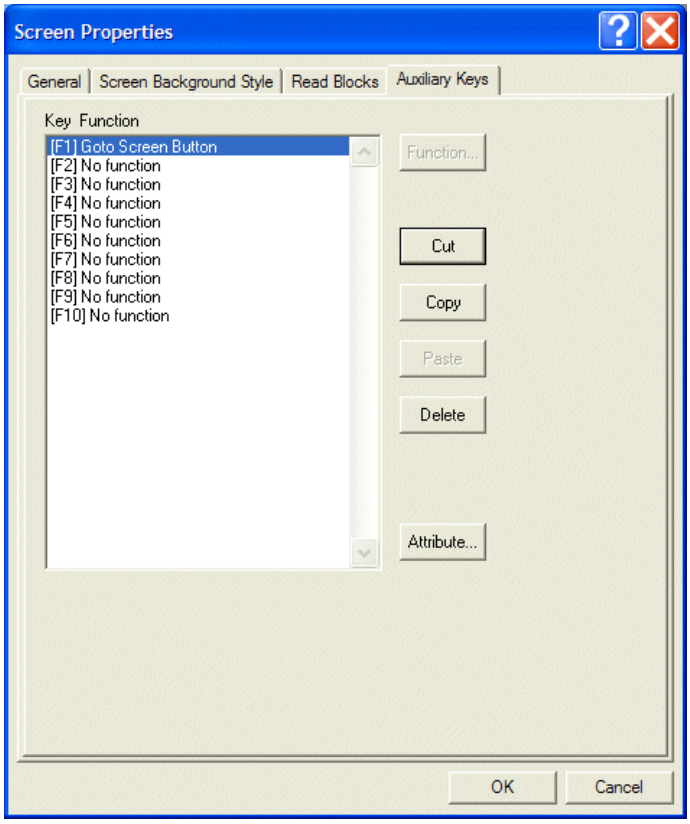
Refresh Rate:

- **Normal**: Reads data at normal speed for the controller.
- **Fast**: Reads data at twice the speed of **Normal**.
- **On Unchanged by Operator**: Enables you to change the value on the operator terminal but not change the value in the controller

Perform block optimization: Check this option to merge adjacent data into blocks.

Note:
It is recommended that the data addresses of the controller are continuous to ensure good communication with the controller.

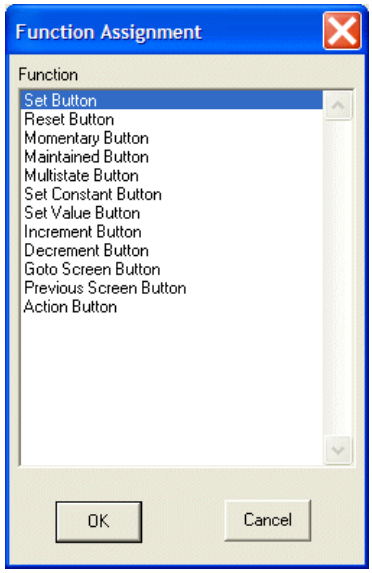
Auxiliary Keys Tab



*The **Auxiliary Keys** tab of the **Screen Properties** dialog box*

The main function of the **Auxiliary Keys** is to create the attributes of external buttons. The buttons defined on the **Auxiliary Keys** tab are only available for the current screen. If the **F1** key is defined in screen 5 as **Goto Screen 1**, this function is only available on screen 5.

Click the **Function** key to display the function assignment dialog box.



*The **Function Assignment** dialog box*

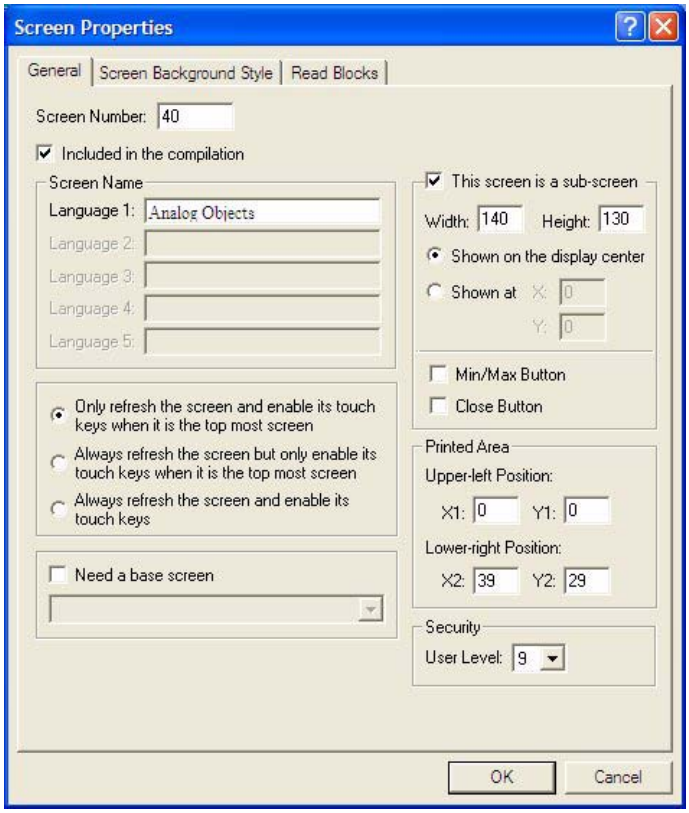
Note:
This function is not available for all operator terminal models: please refer to *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

Sub-screen

A sub-screen is a screen that is smaller than the normal screen. The operator terminal displays a sub-screen in the center of the screen without destroying the existing display and adds a raised frame to it automatically.

The following steps are required to create a sub-screen:

1. Create a new screen; enter the screen name and number.
2. Open the **Screen Properties** dialog box and check the **This screen is a sub-screen** box.
3. Enter the width and height of the sub-screen.
4. Specify the position of the sub-screen display (shown on the display center or designated specific position).



Modifying attributes of a sub-screen

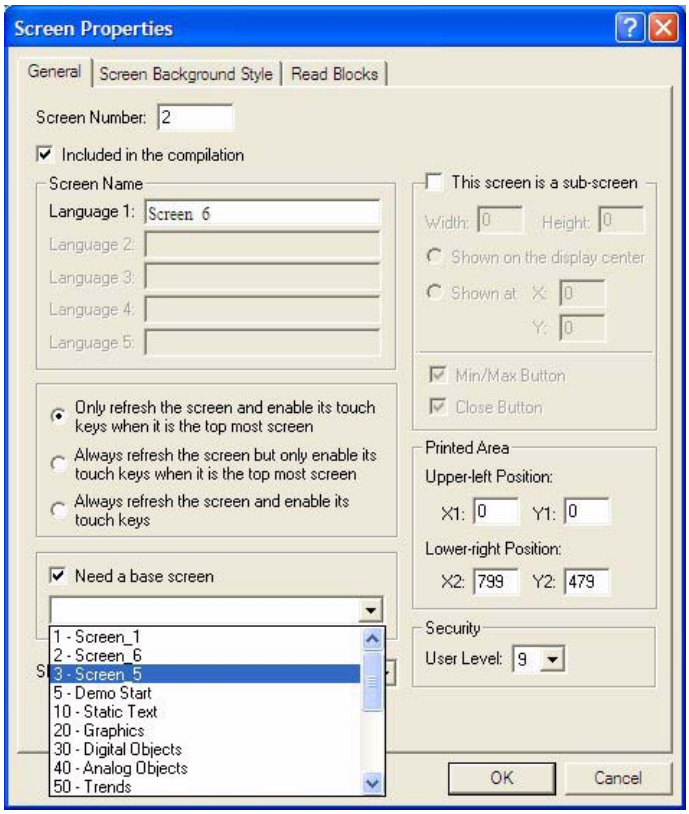
5. After clicking **Enter**, the screen will be minimized to the specified size.

Base Screen

A base screen is a screen which may be used as a template for many different screens. Once you have edited a base screen, all of the same base screens in the application will be changed at the same time.

The following are the steps required to create a base screen:

1. Create a base screen first, named for example **Screen 5**.
2. Create a new screen (**Screen 6**). Then check the **Need a base screen** box and specify the base screen (**Screen 5**) in the **Screen Properties** dialog box.



Specifying a base screen

2.6 Draw Menu

To strengthen the effectiveness of the display of the created objects, it is often helpful to draw a rectangle, a line, or a scale to label the object data. This will help you to read and take note of the data.

The geometric shapes are static, and are not influenced by dynamic controller data.

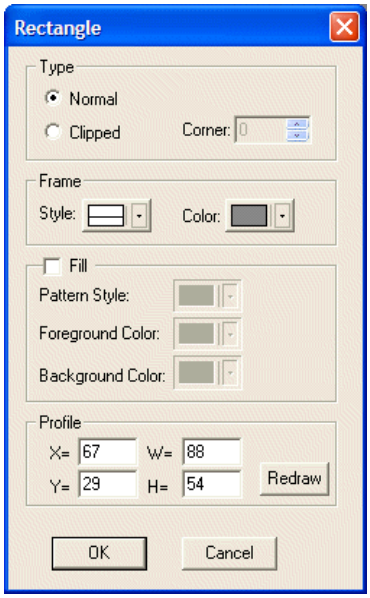
2.6.1 Geometric Shapes

The following geometric shapes are included:

Toolbar button	Function	Described in section
	Draws a dot	*Dot*
	Draws a line	*Line, Horizontal Line and Vertical Line*
	Draws a horizontal line	
	Draws a vertical line	
	Connects lines with mouse movement	*Connected Lines and Free Form*
	Connected lines and curves with mouse movement	
	Draws a rectangle	*Rectangle and Solid Rectangle*
	Draws a solid rectangle	
	Draws a parallelogram	*Parallelogram and Solid Parallelogram*
	Draws a solid parallelogram	
	Draws a circle	*Circle, Solid Circle, Ellipse and Solid Ellipse*
	Draws a solid circle	
	Draws an ellipse	
	Draws a solid ellipse	
	Draws an arc	*Arc, Pie and Solid Pie*
	Draws a pie segment	
	Draws a solid pie segment	
	Draws a polygon	*Polygon and Solid Polygon*
	Draws a solid polygon	
	Static text display and design	*Static Text*
	Displays a bitmap	*Bitmap*

Toolbar button	Function	Described in section
	Draws a frame/edge	*Frame/Edge*
	Draws a scale	*Scale*
	Draws a table	*Table*
	Displays a shape	*Shape*

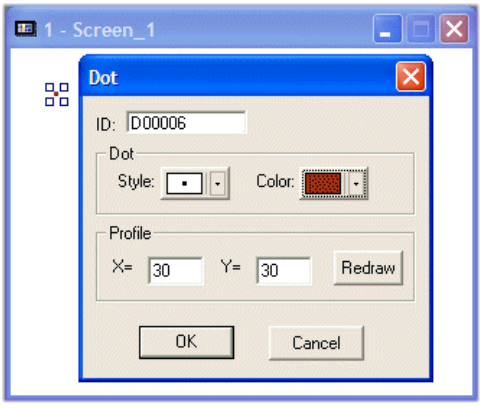
Double-click on the object, or right-click and select **Object Attributes**. The dialog box corresponding to the object will appear.



*The **Rectangle** dialog box*

Dot

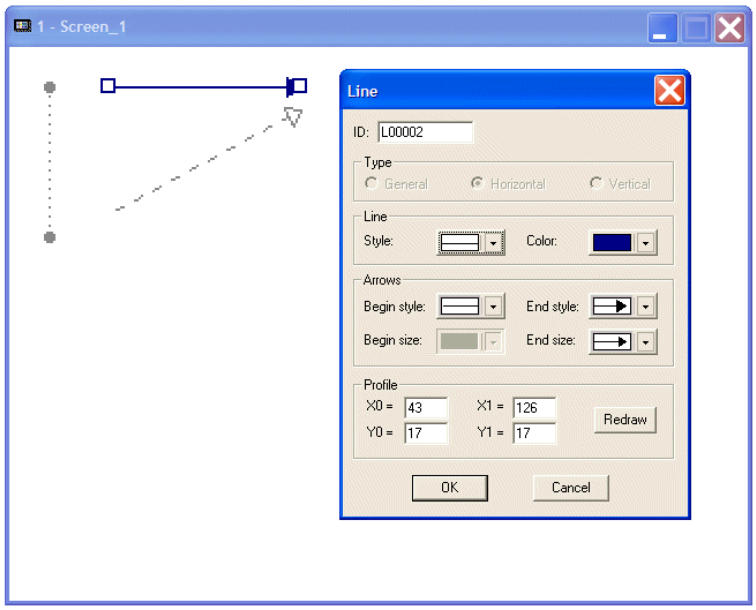
Style, **Color** and **Profile** can be specified in the **Dot** dialog box.



*Drawing a **Dot***

Line, Horizontal Line and Vertical Line

Type, **Color**, **Arrows** and **Profile** can be specified in the **Line**, **Horizontal Line** and **Vertical Line** dialog boxes.



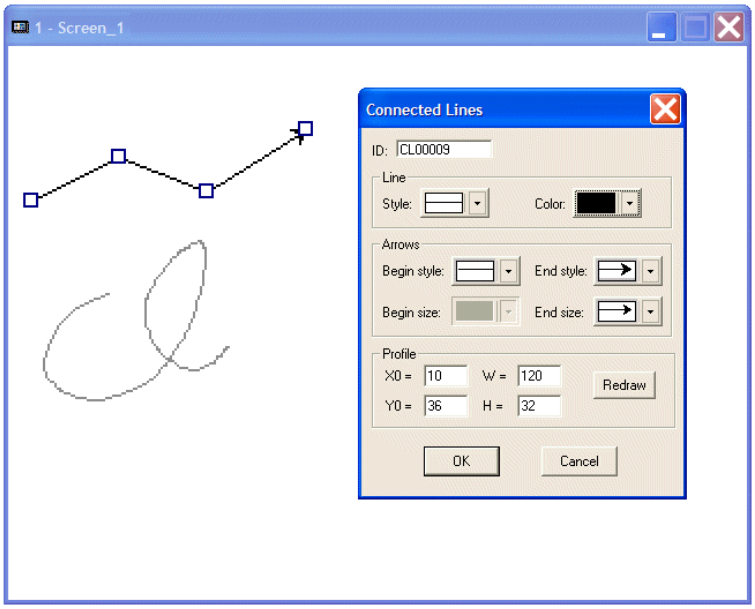
*Drawing a **Line** with arrows*

Connected Lines and Free Form

Connected Lines are used to connect lines with the movement of the mouse cursor. Click mouse button once, and then move the cursor somewhere else on the screen, to draw a straight line between the two points. This will continue until you right-click with the mouse.

Free Form is used to create lines and curves by holding down the left mouse button while moving the mouse cursor. This will continue until you right-click with the mouse.

Style, **Color**, **Arrows** and **Profile** can be specified in the **Connected Lines** and **Free Form** dialog boxes.

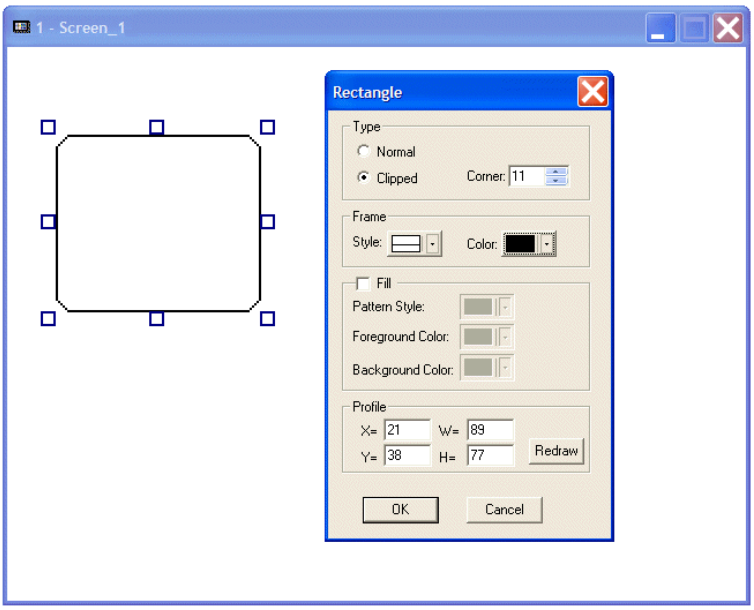


*Drawing **Connected Lines** with arrows*

Rectangle and Solid Rectangle

Rectangle and **Solid Rectangle** are used to draw rectangular shapes.

Type, **Frame** and **Profile** can be specified in the **Rectangle** and **Solid Rectangle** dialog boxes. **Fill** is used for the **Solid Rectangle**.

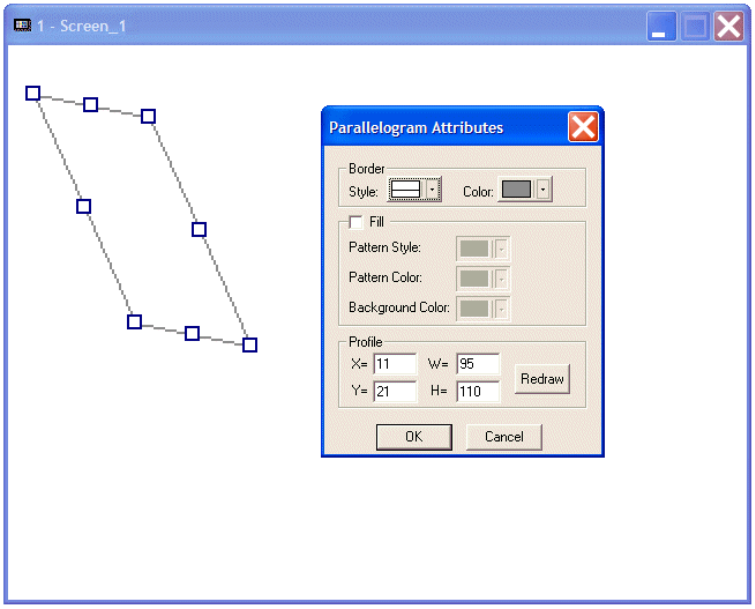


Rectangle with clipped corners

Parallelogram and Solid Parallelogram

Parallelogram and **Solid Parallelogram** are used to draw a parallelograms by holding down the left mouse button to draw a side. This side will continue until you left-click with the mouse. Then you drag this side to configure a rectangle until you right-click with the mouse.

Border and **Profile** can be specified in the **Parallelogram** and **Solid Parallelogram** dialog boxes. **Fill** is used for the **Solid Parallelogram**.

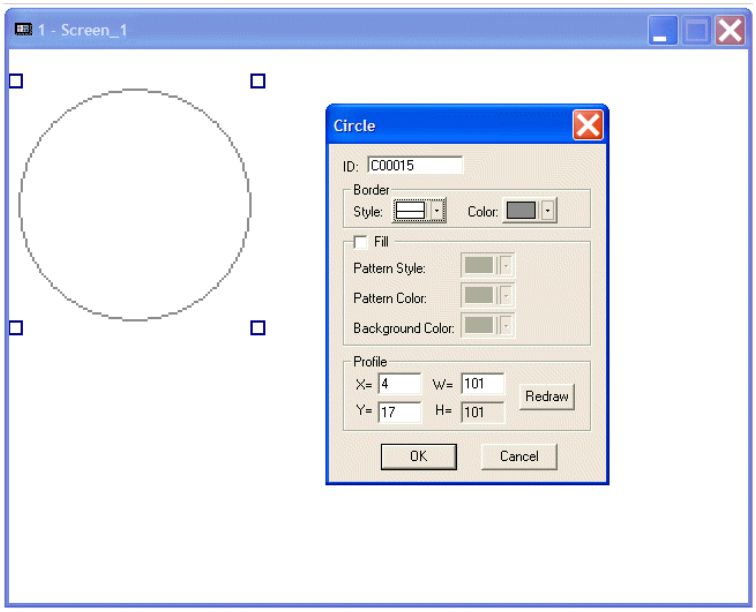


*Drawing a **Parallelogram***

Circle, Solid Circle, Ellipse and Solid Ellipse

Circle, **Solid Circle**, **Ellipse** and **Solid Ellipse** are used to draw circles and ellipses.

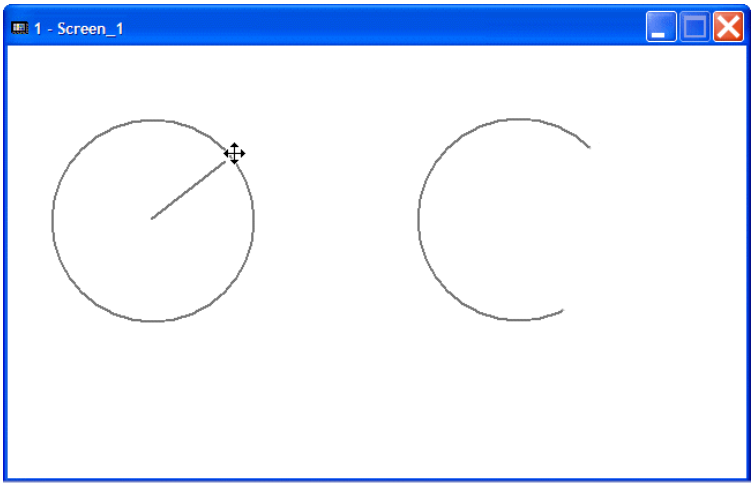
Border and **Profile** can be specified in the **Circle** and **Ellipse** dialog boxes. **Fill** is used for the **Solid Circle** and **Solid Ellipse**.



*The **Circle** dialog box*

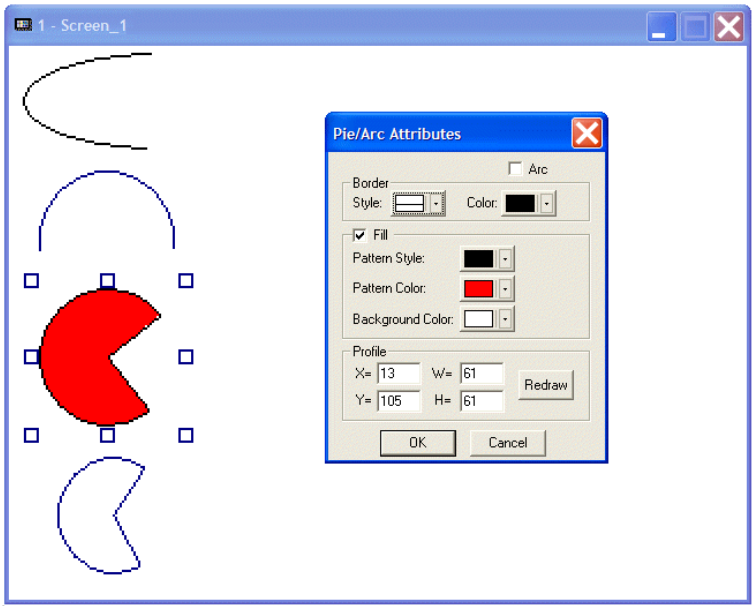
Arc, Pie and Solid Pie

Arc is used to draw a circle by holding down the left mouse button. Continue until you have configured the desired size, then right-click. Left-click for a radius display. You can drag the radius to configure a desired arc until left-clicking again.



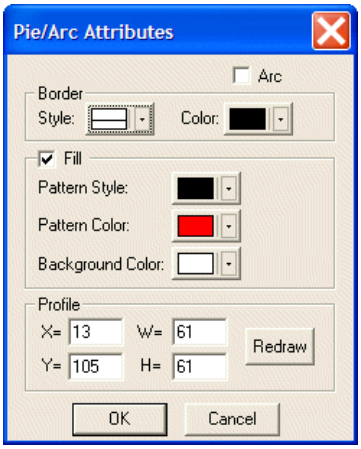
*Drawing an **Arc***

Drawing a pie segment or a solid pie segment is similar to drawing an arc. The difference between a pie and an arc is that two lines are connected between the two sides of an arc and the center.



Arc, Pie** and **Solid Pie

The **Arc**, **Pie** and **Solid Pie** dialog boxes are the same:



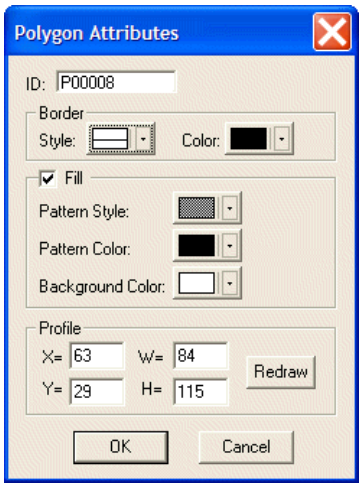
*The **Pie/Arc Attributes** dialog box*

Check **Arc** to draw an arc; check **Fill** to draw a solid arc and specify the **Pattern** there; check **Arc** and adjust the width and height under **Profile** to configure an arc from an ellipse.

Polygon and Solid Polygon

Polygon is used to draw sides of a polygon by moving the mouse cursor and to connect the lines between the starting point and the terminal point with the shortest distance. Draw a polygon by dragging the mouse cursor, then left-clicking on the turning point and right-clicking to form a polygon.

Border and **Profile** can be specified in the **Polygon** and **Solid Polygon** dialog boxes. **Fill** is used for the **Solid Polygon**.



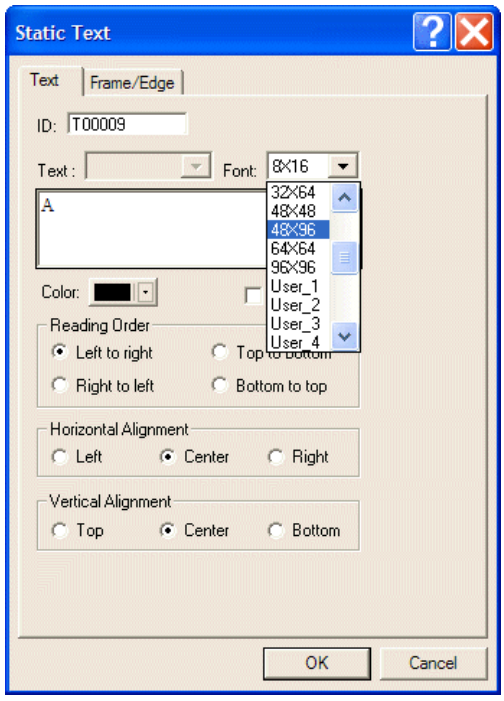
*The **Polygon Attributes** dialog box*

Static Text

The **Static Text** object allows you to select text, font, color, reading order, alignment, and frame/edge. Double-click on the object to display the **Static Text** dialog box on the screen.

Text tab

Enter text in the **Text** area. 16 different types of fonts can be selected.

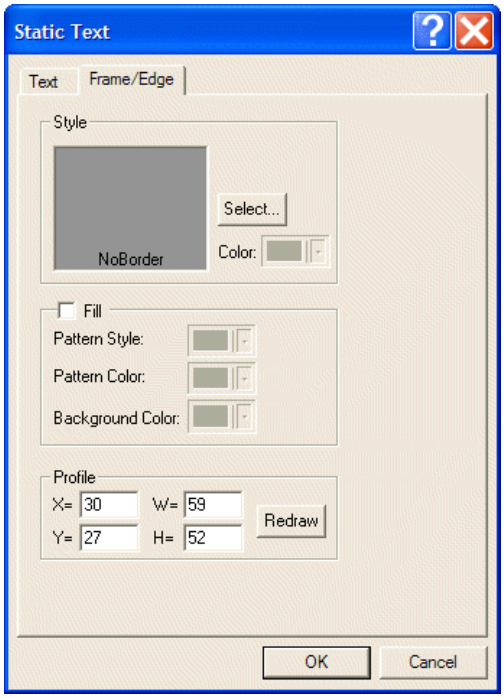


*The **Text** tab of the **Static Text** dialog box*

For properties not explained in this section, please refer to the section *Font Library*.

Frame/Edge tab

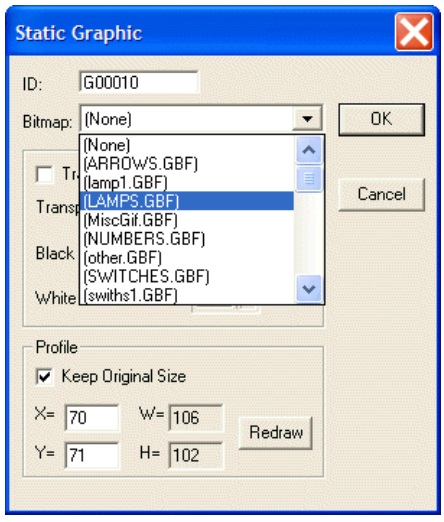
Click the **Select** button to specify the style of frame/edge.



*The **Frame/Edge** tab of the **Static Text** dialog box*

Bitmap

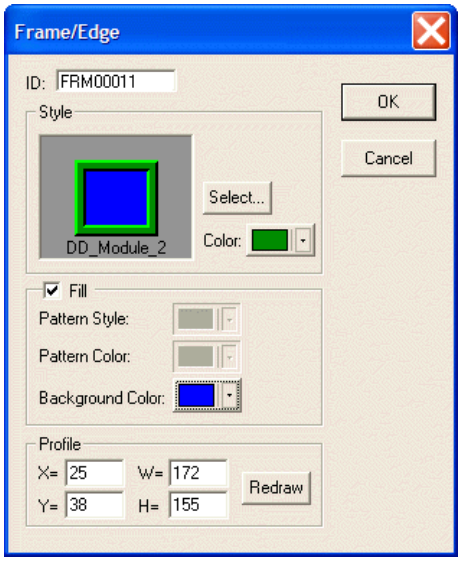
The purpose of **Bitmap** is to provide graphics for selection. Double-click on the object; you can then select the bitmap from the drop-down list in the **Static Graphic** dialog box.



*The **Static Graphic** dialog box*

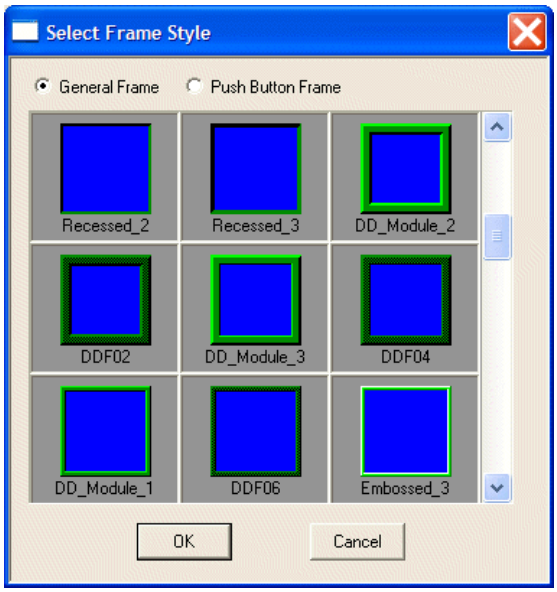
Frame/Edge

Frame/Edge enables you to choose style, pattern style, background and color.



*The **Frame/Edge** dialog box*

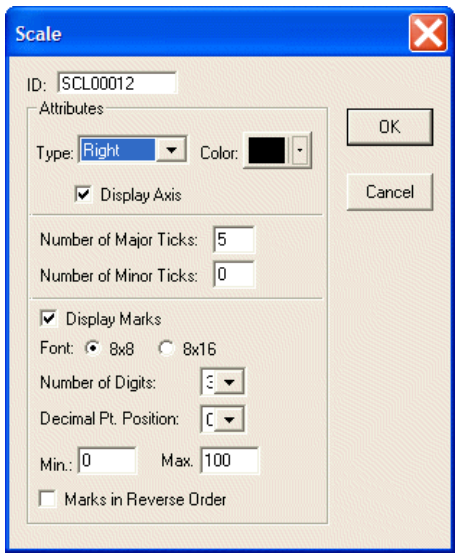
Clicking **Select** displays flow chart styles for selection.



*Selection of **Frame Style***

Scale

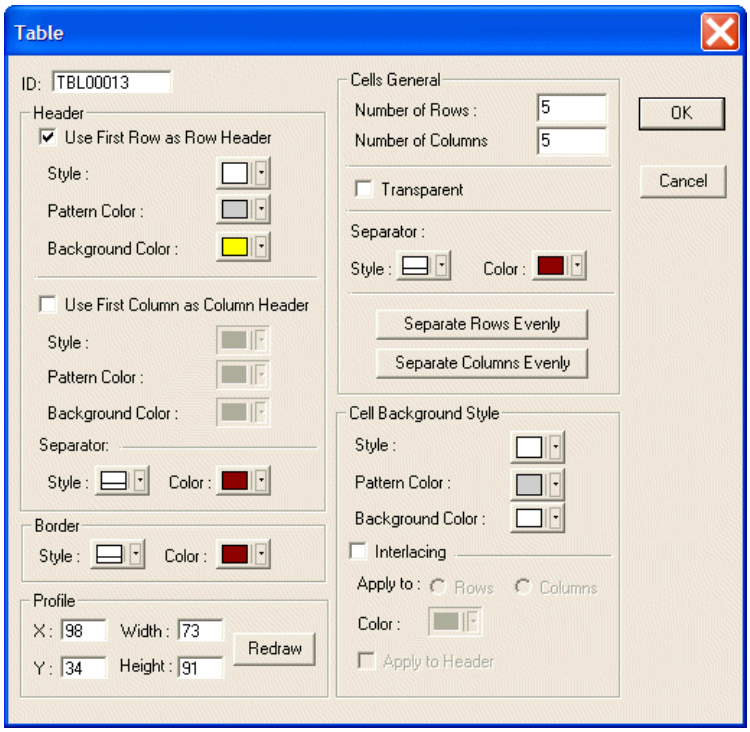
Scale provides scales directed left, right, up and down, as well as color, number of ticks, and display marks for scales.



*The **Scale** dialog box*

Table

Table is used to create tables.



*The **Table** dialog box*

Use First Row as Row Header: Specifies the pattern style and color of row header.

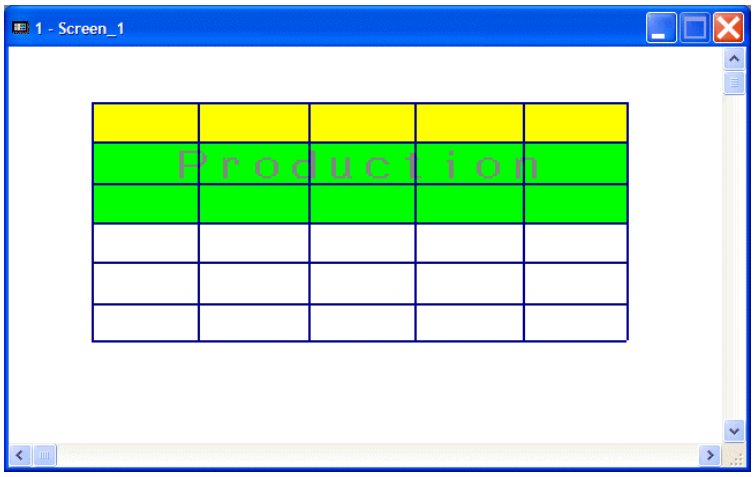
Use First Column as Column Header: Specifies the pattern style and color of column header.

Border: Specifies the style and color of border.

Profile: Specifies the location and size of a table.

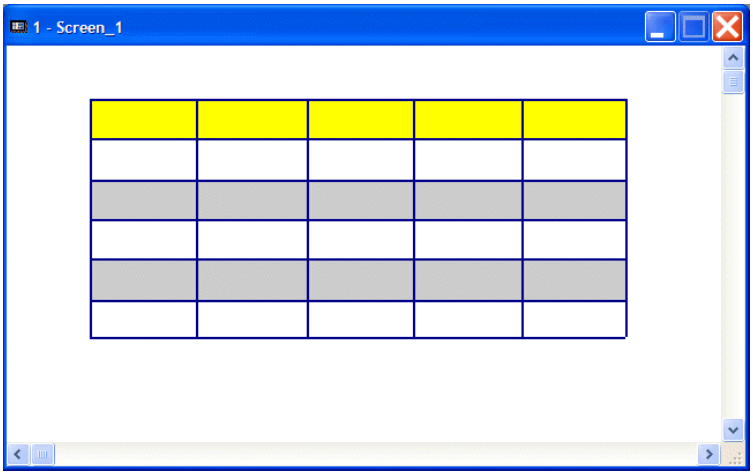
Cells General: Specifies the number of rows/columns, and style.

Transparent: Lets other object(s) be displayed under the table.



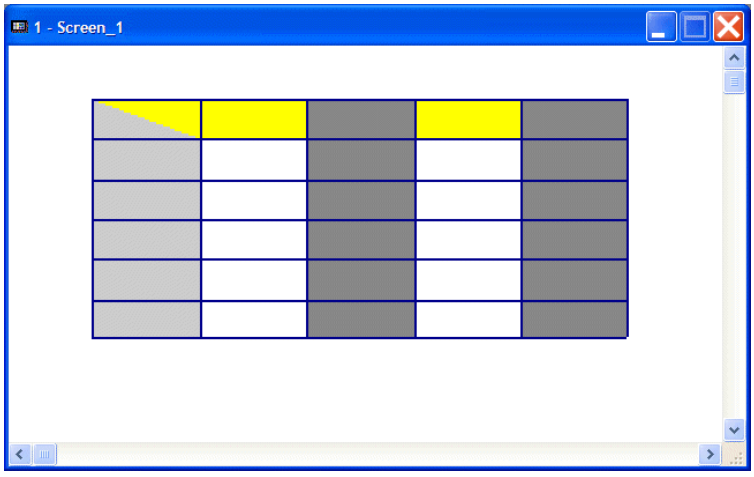
Example: Static text displayed under the table.

Interlacing: Interlaces rows or columns. Not available with transparent table.



Interlacing

Apply to Header: Interlace applied to header. Not available with transparent table.



Interlacing applied to header

Shape

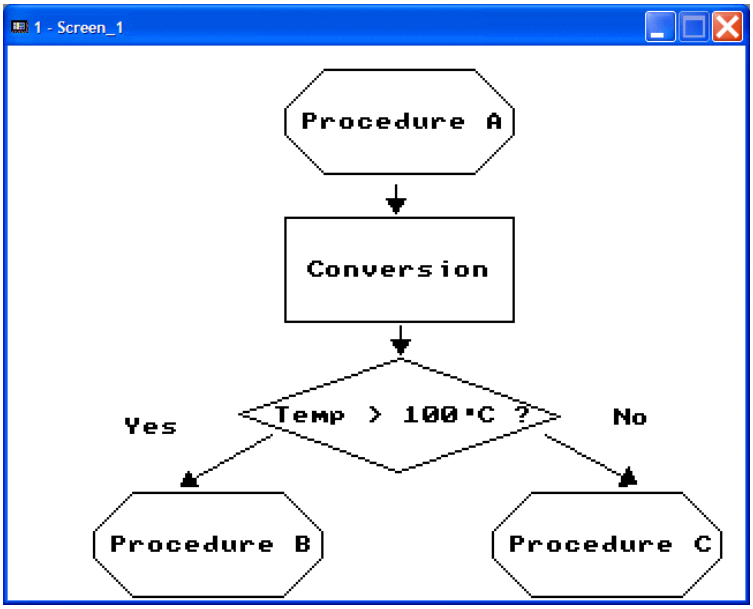
Shape provides graphics for selection. Double-click on the object, then click **Select** to access the shape library in the Shape dialog box.

Flow Chart

Flow chart is one of the applications in **Draw** used for lines, geometric graphs and frame/edge editing. You can illustrate an applied flow chart clearly to facilitate operations.

Example:

Convert the boiler temperature in **Procedure A** into centigrade (° C). If the temperature is less than 100° C, **Procedure B** will be entered; if the temperature is 100° C or more, **Procedure C** will be entered. The following flow chart is made up of polygons, rectangles, lines with arrows and static text:



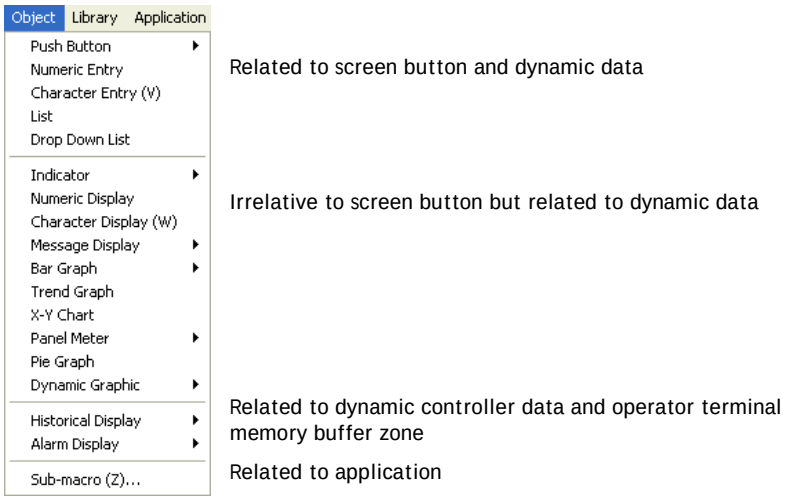
Flow chart example

2.7 Object Menu

A screen object is an item placed on the screen to perform a particular function. Each object has its unique user configurable properties and the object can be set to perform in exactly the method desired.

Objects are divided into four categories:

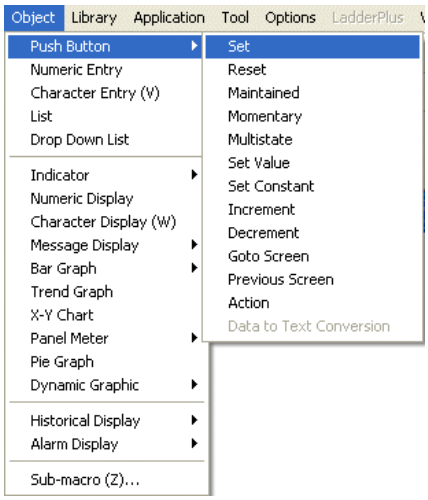
1. Related to screen buttons and dynamic data, e.g. **Push Button**, **Numeric Entry** and **List**
2. Unrelated to screen buttons but related to dynamic data, e.g. **Numeric Display** and **Bar Graph**
3. Related to dynamic controller data and operator terminal memory buffer zone; **Historical Display** and **Alarm Display**
4. Related to application; data contents are connected with the entire system. If one of the contents is modified, such as text display or controller data format, the other objects with the same application will be changed simultaneously as **Sub-macro**.



*The **Object** menu*

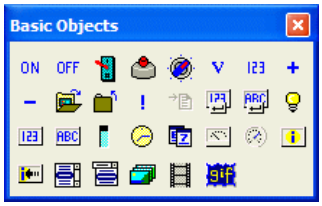
2.7.1 Creating Objects

You can select the object type from the **Object** menu for editing.



*The **Push Button** sub-commands*

Some of the objects are provided in the **Basic Objects** toolbar for editing.



*The **Basic Objects** toolbar*

Select a desired object from the list, for example **Push Button/Set Button**, and you will get a cursor (+) that enables you to drag the object to the desired size by clicking the left mouse button and then clicking again when finished.

Once created, the object can be resized by dragging one of the object's handlebars. If the object's handlebars are not visible, clicking anywhere on the object will display the handlebars). To move the object, click at the center of the object and then drag it.

2.7.2 Specifying Object Properties

There are three ways to specify the properties of an object:

1. You can select **Object Attributes** from the **Edit** menu.
2. You can double-click on the object.
3. You can right-click on the object and then select **Object Attributes** from the pop-up menu.

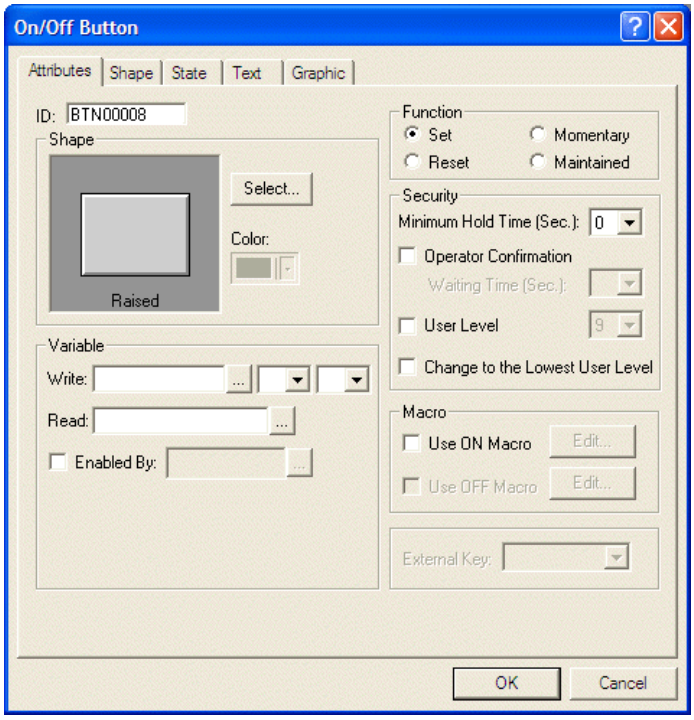
Each of the above methods will bring up the dialog box for the properties specified.

In the H-Designer software, each object has a corresponding dialog box. For example, there is the **On/Off Button** dialog box in the **Set Button** object; there is the **Numeric Entry** dialog box in the **Numeric Entry** object.

The following five tabs are available for most objects; certain specific properties will be explained later.

Attributes Tab

Major properties are specified on the **Attributes** tab.



*The **Attributes** tab of the **On/Off Button** dialog box*

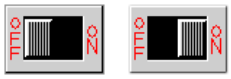
Shape
Select: Specifies shape from library.

Color: Specifies the color of the selected shape.

Variable
Write: Writes to the specified controller register.

Read: Reads the value from the specified controller register. If the location is not specified, the operator terminal reads from the **Write** location.

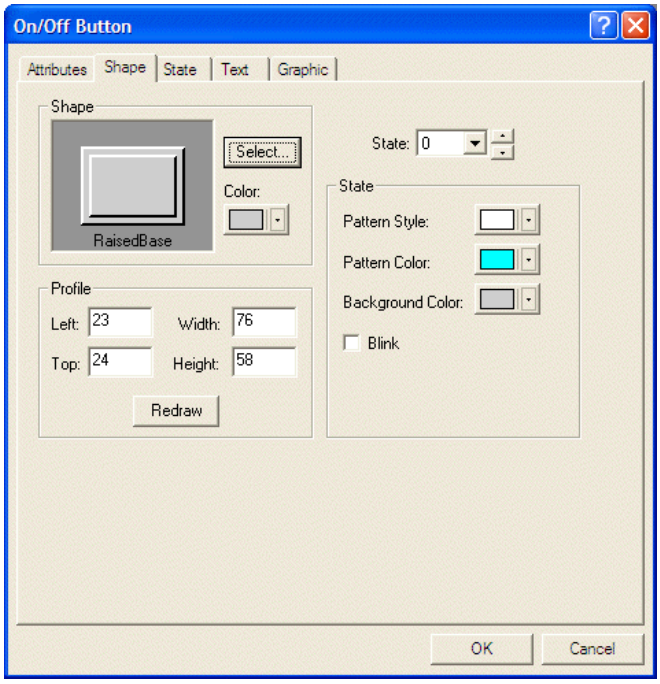
Enabled By: Specifies the controller register to the ON button. This is not available in OFF state labeled.



Specified State = OFF / Specified State =ON

This feature is only available on objects with input text/numeric or specific states.

Shape Tab
The shape style of a selected object is defined on the **Shape** tab.



*The **Shape** tab of the **On/Off Button** dialog box*

Shape
Select: Selects shape from library.

Color: Specifies the color of the shape.

Profile
Specifies the location, width, and height of the object.

State
Pattern Style: Specifies the pattern style for the object.

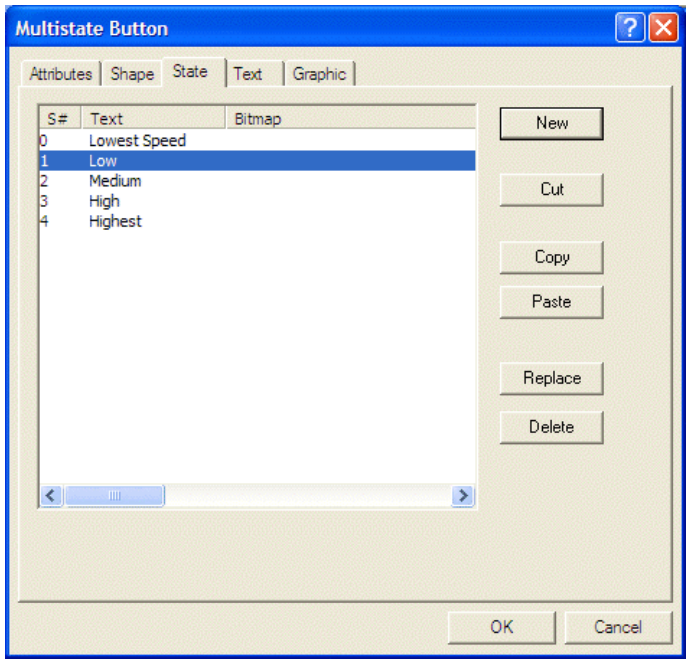
Pattern Color: Specifies the color of the pattern for the object.

Background Color: Specifies the background color of the object.

Blink: Specifies whether the object blinks.

State Tab

The states of the object are defined on the **State** tab.



*The **State** tab of the **Multistate Button** dialog box*

New: Adds a new state to the object.

Cut: Cuts the specified state to the clipboard.

Copy: Copies the specified state of the object and keep the original state.

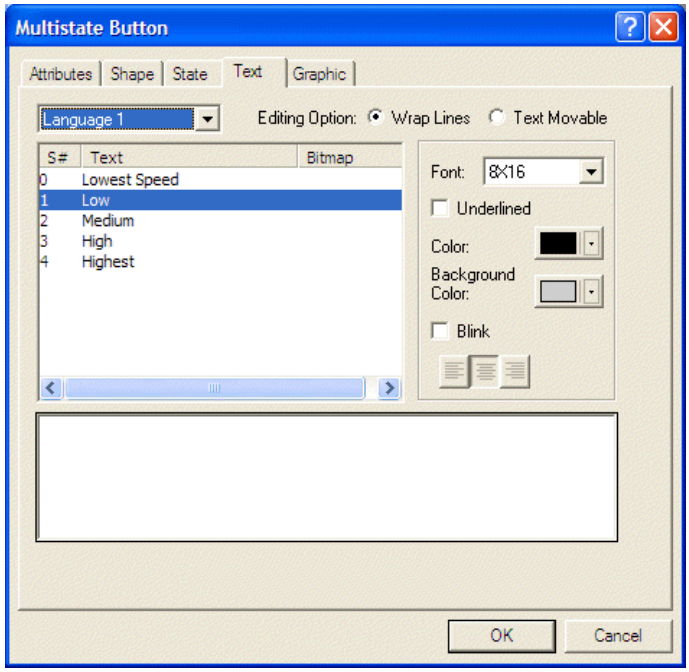
Paste: Pastes the state from the clipboard.

Replace: Replaces the current specified state from the clipboard.

Delete: Deletes the current specified state.

Text Tab

Font and apperarance of the text in the object is specified on the **Text** tab.



*The **Text** tab of the **Multistate Button** dialog box*

Editing Option

Wrap Lines: If the length of the text is longer than the width of the button, it will be executed in wrapped lines.

Text Movable: If the length of the text is longer than the width of the button, it will not be executed in wrapped lines. Click the text on the selected object, to make the text surrounded with the handlebars for dragging.

Appearance

Font: Specifies the size of the font.

Underlined: Specifies whether the text is to be underlined.

Color: Specifies the text color.

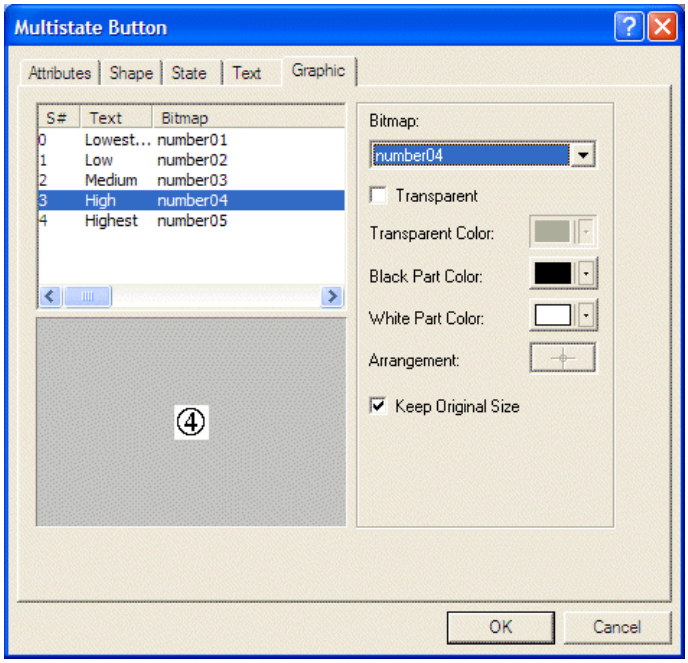
Background Color: Specifies the background text color.

Blink: Specifies whether the text blinks.

 Aligns the text to the left/center/to the right.

Graphic Tab

Bitmap style, color etc for each state is defined on the **Graphic** tab.



*The **Graphic** tab of the **Multistate Button** dialog box*

Bitmap: Specifies the bitmap to display.

Transparent: Specifies the transparency of the bitmap.

Transparent Color: Specifies the color of the bitmap when transparent.

Black Part Color: Replaces the black part color (only available for monochrome diplays).

White Part Color: Replaces the white part color (only available for monochrome displays).

Arrangement: Arranges the moveable bitmap to a previous location.

Keep Original Size: Keeps the bitmap's original size.

2.7.3 Buttons

There are 13 buttons in the sub-command list for the **Push Button** menu:

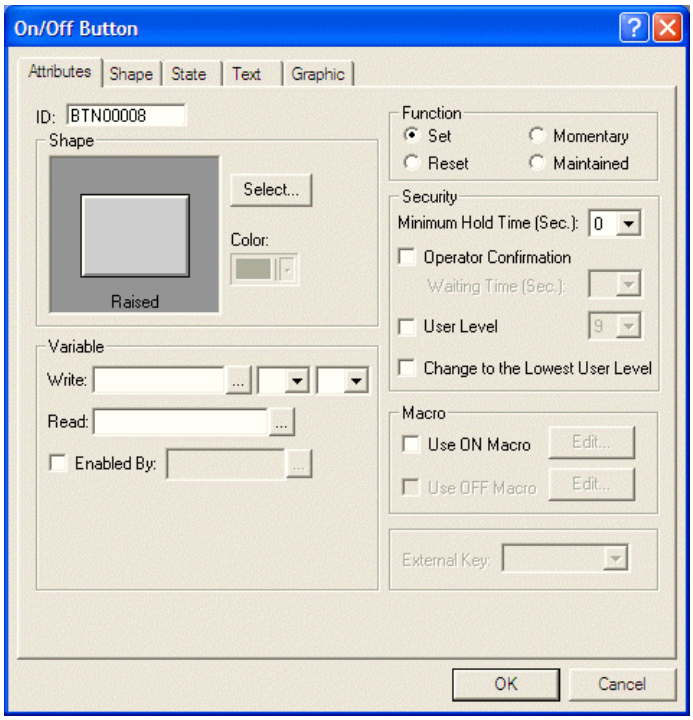
Icon	Function	Described in section
	Set Button: Click to set the contact as ON, release or re-click still to set ON.	*Set Button*
	Reset Button: Click to set the contact as OFF, release or re-click still to set OFF.	*Reset Button*
	Maintained Button: Click to set the contact as ON, released still on; re-click to set OFF.	*Maintained Button*
	Momentary Button: Click to set the contact as ON; release be OFF.	*Momentary Button*
	Multistate Button: Click to change a register to the next (previous) state of a referenced register. S0 → S1 → S2 → S3 → S4 → S0 (a straightforward cycle) or S0 → S4 → S3 → S2 → S1 → S0 (a reverse cycle).	*Multistate Button*
	Set Value Button: Click to a numeric keypad display. Click ENTER button to write a numeric entry to corresponding controller register.	*Set Value Button*
	Set Constant Button: Click to write a constant to a register.	*Set Constant Button*
	Increment/Decrement Button: Click to write the value obtained by adding/subtracting a constant to/from the corresponding register value corresponding controller register.	*Increment Button*, *Decrement Button*
	Goto Screen Button: Click to change the current screen to the specified screen.	*Goto Screen Button*
	Previous Screen Button: Click to change the current screen to the previous screen.	*Previous Screen Button*
	Action Button: Performs a built-in action.	*Action Button*

Set Button

When pressed, the operator terminal sets the controller's corresponding bit location to ON. A **Set Button** will be ON whether pressed or released.

Attributes Tab

Function: Select **Set**.

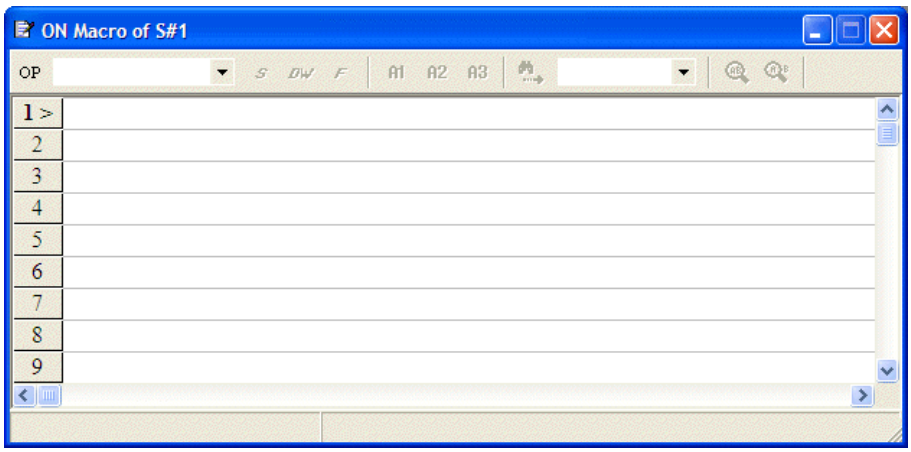


*Selecting the **Set** function for the **Set Button***

Security:

- **Minimum Hold Time (Sec.)**: Specifies how long to activate the button (0-10 seconds).
- **Operator Confirmation**: If any changes have been made, this dialog box will appear on the screen to ask the user to confirm the desired operation. A waiting time of 5-60 seconds can be selected.
- **User Level:** Select user level for the object.
- **Change to the Lowest User Level:** Changes to the lowest user level.

Macro: Check the **Use ON Macro** for a **Set Button**. Click the **Edit** button to display the **ON Macro** dialog box.



The ON Macro edit window

For properties not explained in this section, please see chapter *Macros*.

– **Use ON Macro**: When the **Set Button** is selected, the operator terminal will run the program createdas ON macro. This feature is for data control, screen display, controller register, bits initialized and so on.

For properties not explained in this section, please see the section *Specifying Object Properties*.

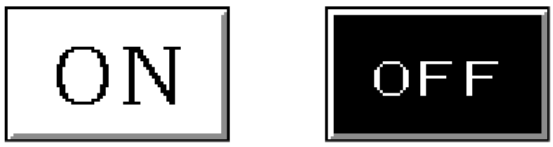
Example of designing a Set Button

On the **Attributes** tab of the **On/Off Button:**

1. **Shape**: Select **Raised**.
2. **Write**: Specify the controller register **B0** to write in.
   Read: Specify **B0**. (The controller model is NULL.)

On the **Text** tab of the **On/Off Button:**

3. Enter text **OFF** for state 0; the font **16 x 16**, the color **White** and the background color **Black**.
4. Enter text **ON** for state 1 ; the font **24 x 24**, the color **Black** and the background color **White**.



*The button is white displaying **ON** in state 1, and black displaying **OFF** in state 0.*

Reset Button

This button is contrary to the **Set Button**. The **Reset Button** sets a bit-location to OFF whenever pressed or released.

Attributes Tab

Function: Select **Reset**.

All other properties are the same as for the **Set Button**; please see section *Set Button*.

Maintained Button

The function of the **Maintained Button** is to change the button states by pressing. Click to be ON and when released will still be ON until re-clicked to be OFF.

Attributes Tab

Function: Select **Maintained**.

Macro: There are **Use ON Macro** and **Use OFF Macro** options for the **Maintained Button**. For properties not explained in this section, please see chapter *Macros*.

All other properties are the same as for the **Set Button**; please see section *Set Button*.

Momentary Button

The function of the **Momentary Button** is to change the state by clicking and releasing. When the button is pressed, the bit-location is ON; when it is released, it is OFF.

Attributes Tab

Function: Select **Momentary**.

Macro: There are **Use ON Macro** and **Use OFF Macro** options for the **Momentary Button**. For properties not explained in this section, please see chapter *Macros*.

To create two states as an ON button simultaneously, please see section *Set Button*.

Multistate Button

When the button is pressed, the operator terminal will write the command to a corresponding controller bit-location or register. The option **Change to Next State** to changes states in a straightforward cycle (S0 → S1 → S2 → S3 → S4 → S0); the option **Change to Previous State** is changes states in a reverse cycle (S0 → S4 → S3 → S2 → S1 → S0). Click to change a register to the next or previous state of a referenced register.

Attributes Tab

Variable:

Write: Writes the specified command to a corresponding controller bit-location and register.

- **Bit**: Only two states; enables you to enter multistated text but only two states can be displayed on the operator terminal.
- **Value**: 256 (0-255) states in all, 0 represents state 0; 1 represents state 1…etc.
- **LSB**: 16 states in all represented by bit. The operator terminal takes the bit number of the lowest bit that is on as the state number.

Format: Only available when **Value** has been selected. There are **BCD**, **Signed Binary** and **Unsigned Binary** options.

- **Read**: Specifies a register/bit location to read from; if the location is not specified, then the operator terminal reads from the **Write** location.

Function:

- **Change to Next State**: Changes the **Write** location to its next state in a forward cycle S0 → S1 → S2 → S3 → S4 → S0.
- **Change to Previous State**: Changes the **Write** location to its previous state in a reverse cycle S0 → S4 → S3 → S2 → S1 → S0.

Note:
The number of states can be edited on the **State** tab.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing a Multistate Button

On the **Attributes** tab of the **Multistate Button:**

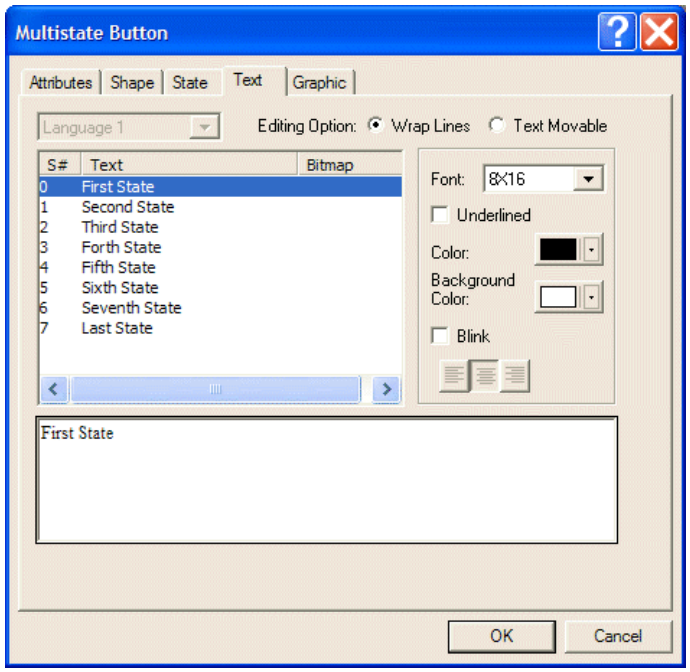
1. **Write**: Specify controller register **W50** to write in.
   Read: Specify **W50**. (The controller model is NULL.)
2. **Format**: **Value**.
3. **Function**: **Change to Previous State**.
4. **Shape**: Select **Outlined _1** and the color **Black**.

On the **State** tab of the **Multistate Button:**

5. Add **8 States**.

On the **Text** tab of the **Multistate Button:**

6. Enter the corresponding texts.



*Texts for the 8 different states of the **Multistate Button** in the example*

The above steps will create a **Multistate Button** that displays **First State** in state 0; **Second State** in state 1 etc.

Set Value Button

The function of the **Set Value Button** is to enable numeric entry. When pressed, the operator terminal displays a numeric keypad on the screen. When pressing **ENT**, the operator terminal will store the input value to the corresponding controller register.

Note that the corresponding controller value is not available in the **Set Value Button**.

Attributes Tab

Variable:

- **Word**: The entered value is 16-bit data; maximum value 65,535.
- **Double Word**: The entered value is 32-bit data; maximum value is 4,294,967,295.
- **Format**: **BCD**, **Signed Binary**, **Unsigned Binary** and **Hexadecimal** can be selected.
- **Notification**: Specify a register/bit location to be notified; the operator terminal will set the bit to ON.
  Select **Before Writing** to make the operator terminal set the notification to ON when the numeric keypad appears, and set the location to OFF when the numeric keypad disappears.
  Select **After Writing** to make the operator terminal set the notification location to ON after writing the input value to the **Write** location.

Display Format:

- **Decimal Pt. Position**: Specifies the number of digits after the integral part of the number. The maximum is based on the specified format.
- **Intergral Digits**: The number of the integral part.
- **Fractional Digits**: The number of decimal digits.
- **Display Asterisk Instead of Number**: Displays an asterisk instead of input value for security reasons.

Validation and Security:

- **Input Min.**: Sets the minimum input value. A number less than the minimum input value will be rejected.
- **Input Max.**: Sets the maximum input value. A number greater than the maximum will be rejected.
- **User Level**: There are 9 user levels, the order is 1 > 2 … > 8 > 9.
- **Operation Confirmation**: Check this box to display a dialog box on the screen asking for **User Confirmation** during a waiting time of 5-60 seconds.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing a Set Value Button

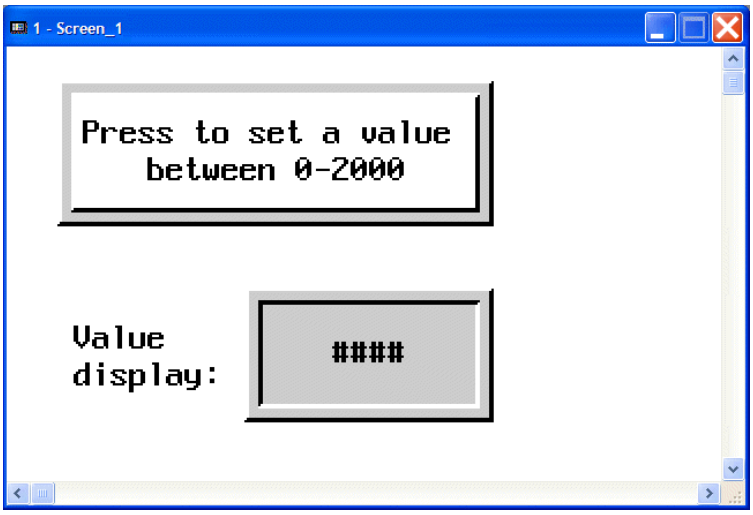
Set Value Button on the **Attributes** tab:

1. **Shape**: Select **RaisedBase**.
2. **Write**: Specify controller register **W100** to write in. **Numeric Entry**: **Word Notification**: Specify **B10** and **After Writing**. (The controller model is NULL.)
3. **Display Format**: Check **Display Asterisks Instead of Number**.
4. **Validation and Security**: Select **0** fot **Input Min.**, and **2000** for **Input Max.** Check **Operator Confirmation** to require a confirmation after numeric entry.

On the **Text** tab of the **Set Value Button:**

5. Enter the text, for example **Press to set a value between 0-2000**, and select **White** for background color.

The steps above will create a **Set Value Button**. When the button is pressed, the numeric keypad will be displayed on the screen. After numeric entry, the input value cannot be displayed directly on the **Set Value Button**. Instead, you can create a **Numeric Display** object to display the input value. When the value is changed, B10 is set.



*The **Set Value Button** and **Numeric Display** object*

Set Constant Button

The **Set Constant Button** is used to make the operator terminal write a constant to the corresponding controller register. The numeric keypad will not be displayed on the screen since the constant has been set in the controller.

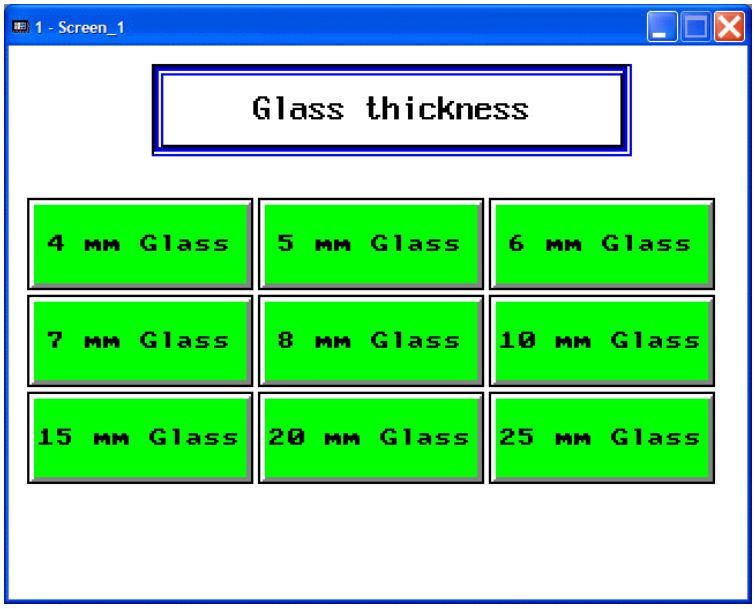
Attributes Tab

Value: Specifies the constant value.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing a Set Constant Button

Here we use a glass list as an example. When one of the **Set Constant Buttons** is pressed, the operator terminal will write the specified constant value to the corresponding controller register. In this example, **4 mm Glass** represents the constant value **400**, **5 mm Glass** represents the constant value **500** etc.



*The **Set Constant Buttons** - glass list example*

Set Constant Button on the **Attributes** tab:

1. **Variable:**
   Write: Specify controller register **W10**.
   Notification: Specify **B10**. (The controller model is NULL.)
2. **Set Value:** Select **Word** for numeric entry.
   Value: Specifiy the constant value **400** for the **4 mm Glass** button.

On the **Text** tab of the **Set Constant Button:**

3. Enter the text **4 mm Glass** and select **Green** for background color.

The above steps will create a **4 mm Glass** button. When clicked, the operator terminal will store the constant value **400** to the register **W10**.

Follow the same steps to create other glass buttons, using the corresponding constant values.

Increment Button

The **Increment Button** is used to make the operator terminal read a constant variable stored in a controller register. Then a specified constant will be added to the value, before writing it back to the controller register.

Attributes Tab

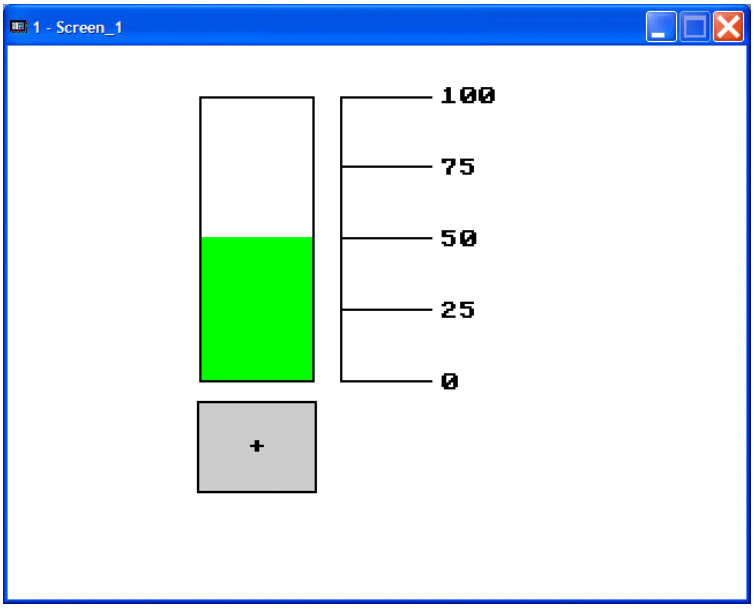
Function:

- **Increment**: Select **Increment** to create an **Increment Button**; one click increases a specified constant.
- **Jog Step**: The **Increment Button** is used to add a specified constant by clicking.
- **Limit**: Specifies the maximum value written to the register when using the **Increment Button**.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing an Increment Button

When clicking the **Increment Button**, the increased value stored in the controller register will be displayed in the bar graph. The **Bar Graph** is an object used to display the dynamic data.

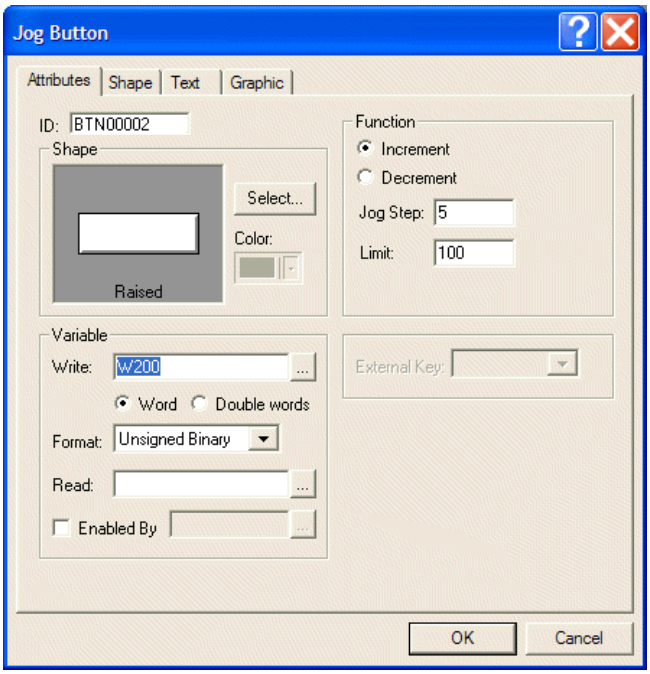


*When the **Increment Button** is clicked, the level in the **Bar Graph** is increased*

On the **Attributes** tab of the **Increment Button:**

1. **Variable:**
   Write: Specify controller register **W200** to write in.
   Read: Specify **W200**. (The controller model is NULL.)
2. **Shape**: Select **Raised**.

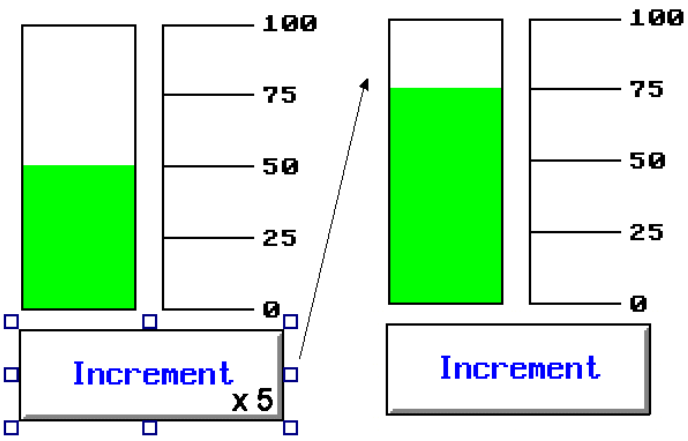
3. **Function**: Select **Increment**, **Jog Step**: 5 and **Limit: 100**.



*Attribute settings for the **Increment Button***

On the **Text** tab of the **Increment Button:**

4. Enter the desired text, for example **Increment**, and select **White** for background color.



*Clicking the **Increment Button** five times adds **25** to the controller constant **W200***

Decrement Button

The Decrement Button is used to make the operator terminal read a constant variable stored in a controller register. Then a specified constant will be subtracted from the value, before writing it back to the controller register.

Attributes Tab

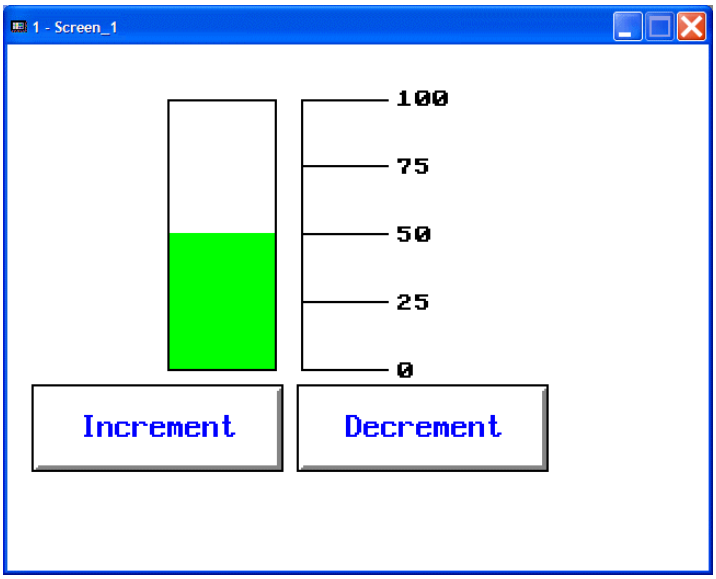
Function:

- **Decrement**: Select **Decrement** to create a **Decrement Button**; one click is one subtraction.
- **Jog Step**: The **Decrement Button** is used to subtract a specified constant by clicking.
- **Limit**: Specifies the minimum value written to the register when using the **Decrement Button**.

For properties not explained in this section, please see the section *Specifying Object Properties*.

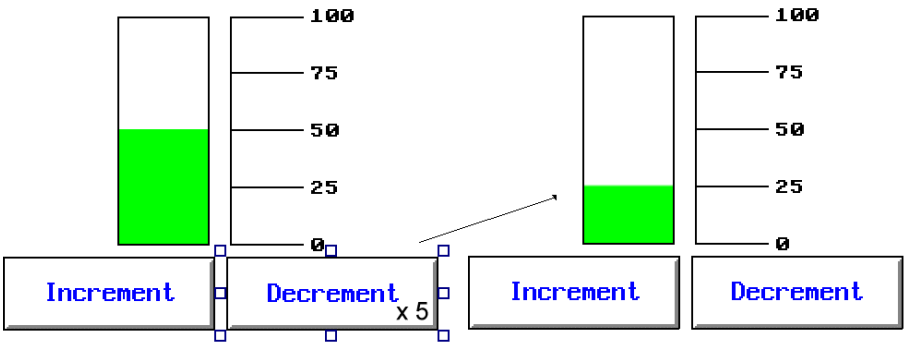
Example of designing a Decrement Button

When clicking the **Decrement Button**, the subtracted constant value stored in the controller register will be displayed in the bar graph. The **Bar Graph** is an object used to display the dynamic data..



*When the **Decrement Button** is clicked, the level in the **Bar Graph** is decreased*

The steps used to create a **Decrement Button** are the same as in section *Example of designing an Increment Button* , but remember to change **Limit** to **0**.



*Clicking the **Decrement Button** five times subtracts **25** from the controller constant*

Goto Screen Button

The **Goto Screen Button** is used to change the current screen to another screen.

Attributes Tab

Function:

– **Open/Go To**: Check this option to create a **Goto Screen Button**. You can select which screen to open from the drop-down list.

– **Enabled By**: Only change the screen when the controller register is ON.

Execution:

– **On Press**: Executes the command (changes the screen) when button is pressed.

– **On Release**: Executes the command (changes the screen)when button is released.

Appended Functions:

– **Change to the Lowest User Level**: Sets the current user level as the lowest level (User Level 9).

– **Acknowledge Alarm**: Acknowledges the current active alarm when the button is clicked.

– **Notify**: Notifies the specified bit-location after clicking the button.

Security:

– **User Level**: There are 9 user levels, the order is 1 > 2 … > 8 > 9.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing a Goto Screen Button

In this example, clicking the **Goto Screen Button** will open **Screen_3**, assuming that the **Screen_3** has been created in the project.

On the **Attributes** tab of the **Goto Screen Button:**

1. **Shape**: Select **Raised**.
2. **Function**: Select **Screen_3** from the drop-down list.
3. **Execution**: Select **On Press**.

On the **Text** tab of the **Goto Screen Button:**

4. Enter the desired text

Previous Screen Button

The **Previuos Screen Button** is used to return to the previous screen in the operator terminal.

Attributes Tab

Function:

– **Close/Previous**: Check this option to create a **Previous Screen Button**. You can select which screen to open from the drop-down list.

For properties not explained in this section, please see the sections *Goto Screen Button* and *Specifying Object Properties*.

Example of designing a Previous Screen Button

The steps used to create a **Previous Screen Button** are the same as in section *Example of designing a Goto Screen Button*, but remember to check the **Close/Previous** option for **Function**.

Action Button

The Action button is used to perform a built-in function.

The following actions are available:

Action	Description
Contrast Up	Increases the contrast or brightness of the display.
Contrast Down	Decreases the contrast or brightness of the display.
Save contrast	Saves the setting of contrast or brightness.
Password Table	Displays the password table.
Reenter Password	Displays the password table to reenter.
Set Lowest User Level	Changes to the lowest user level (level 9).
Print Screen	Prints the specified region (HARDCOPY) of current screen.
Goto System Menu	Changes to the system menu.
Turn off Backlight	Turns off the backlight.
Alarm Ack	Acknowledges the current active alarm to continue.
Set Time & Date	Sets the time and date.
Sub-connection table	Verify commnication states (Only for Modbus PLC)
Update Application Data from File	Update application data from CF Card/USB Memory Stick
Write Application Data to File	Write application data to CF Card/USB Memory Stick
Read Recipe Data from File	Read recipe data from CF Card/USB Memory Stick
Write Recipe Data to File	Write recipe data to CF Card/USB Memory Stick
Write Logging Data to File	Write logging buffer data to CF Card/USB Memory Stick
Write Alarm History to File	Write alarm history to CF Card/USB Memory Stick
Save Recipe Data to Flash ROM	Write recipe data to Flash ROM
Read Recipe Data from Flash ROM	Read recipe data from Flash ROM
Select Language #1-#5	Displays the screen in the specified language, 5 languages available for selection.

All features are not available on all operator terminal models; please refer to *Appendix A - H-Designer Features and Operator Terminal Models*.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing an Action Button

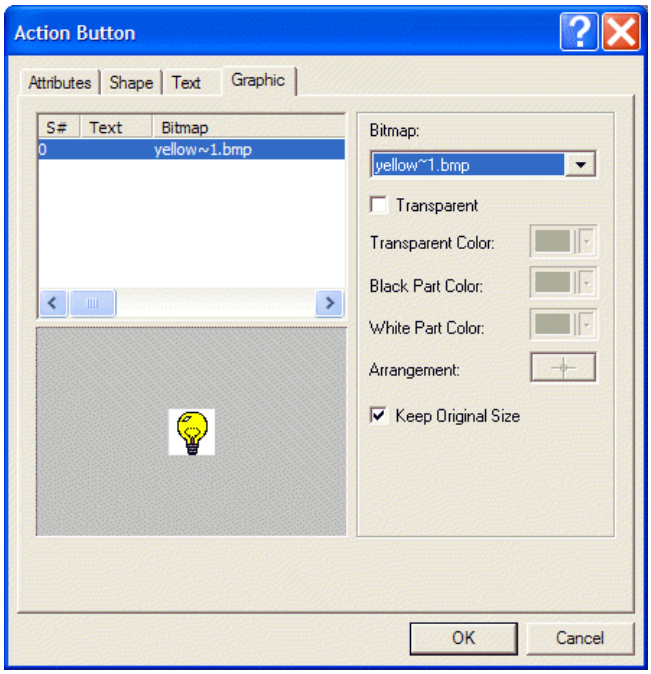
In this example, the **Action Button** will be used to change the contrast of the display.

On the **Attributes** tab of the **Action Button:**

1. **Action**: Select **Contrast Up** from the drop-down list.

On the **Graphic** tab of the **Action Button:**

2. **Bitmap**: Select a symbol.



*Selecting a symbol for the **Action Button***

2.7.4 Numeric Entry

The **Numeric Entry** is used to write an input value to a controller register, and to display the value on the operator terminal screen. When the button is clicked, a numeric keypad will be displayed on the screen. Enter a value and then press **ENT** on the keypad. The operator terminal will then write the input value to the specified controller register.

Variable

Write: Specifies the controller register.

Format: **BCD**, **Signed Binary**, **Unsigned Binary**, **Hexadecimal**, 32-**bit Floating-point** and **Octal** can be selected.

Notification: Specifies a register/bit location to be notified; the operator terminal will set the bit to ON.

- **Before Writing**: The operator terminal sets the notification location to ON when the numeric keypad appears and sets the location to OFF when the numeric pad disappears.
- **After Writing**: The operator terminal sets the notification location to ON after writing the input value to the write location.

Display Format

- **Font:** Select between a number of pre-defined fonts or 3 user-defined fonts. The user-defined fonts are set up in the *Font Library*.
- **Fill Leading Zeroes**: Select this option to add leading zeros; for example, **5902.1** is displayed as **005902.1**.
- **Decimal Pt. Position**: Specifies the number of digits after the integral part of the number. There are 0-10 digits for selection.
- **Fractional Digits**: The number of decimal digits.

  (Fractional Digits + Integral Digits or Decimal Pt. Position <= the maximum number of digits.)
- **Integral Digits**: The number of the integral part in a number.

  (Fractional Digits + Integral Digits or Decimal Pt. Position <= the maximum number of digits.)
- **Scaling**: The formula is **Y = aX+b**.
  Note that only the formats Signed Binary, Unsigned Binary, 32-bit Floating-point and BCD support this option.

  Gain: **Y = aX**, where **X** = the value stored in controller and **Y** = operator terminal displayed value.

  Offset: If the initial value is not zero, then set the Offset.

Validation and Security

- **Variable Input Limits**: Set the input limits as variable. The minimum is stored in the bit following the write location; the maximum is stored in the bit following the minimum input value.
  For example, if the write location is **W10**, then the minimum is stored in **W11**; the maximum is stored in **W12**.
- **Min**: Set the minimum input value. Values less than the minimum input value will be rejected.
- **Max**: Set the maximum input value. Values greater than the maximum will be rejected.

For properties not explained in this section, please see the sections *Set Value Button* and *Specifying Object Properties*.

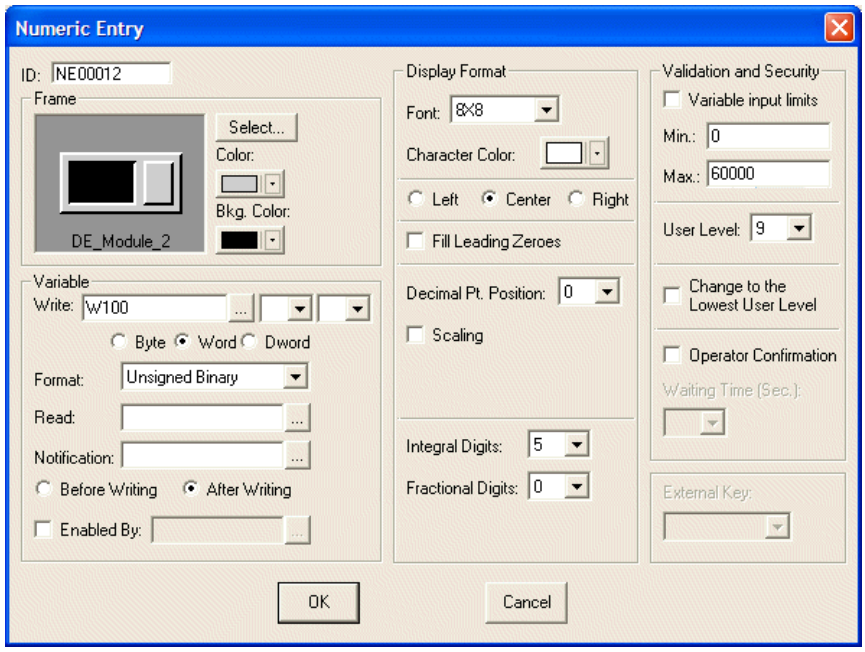
Example of designing a basic Numeric Entry button

Perform the following steps to create a basic **Numeric Entry** button:

*A basic **Numeric Entry** button*

1. **Frame**: Select **DE_Module_2** and **Black** for background color.
2. **Variable:** Specifiy the controller register **W100** for **Write** to store the value. (The controller model is NULL.) Select **Unsigned Binary** for **Format**.
3. **Display Format**: Select **White** for character color. Allow **5 Digits** in a number.

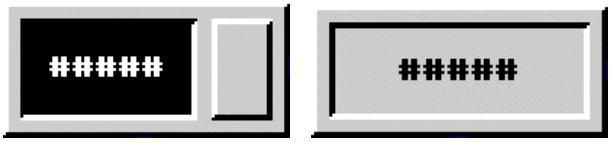
Validation and Security: Set the minimum input value to **0** and the maximum input value to **60000**.



***Numeric Entry** properties in this example*

The steps above will create a **Numeric Entry** button. When the button is clicked, a numeric keypad will be displayed on the screen. After entering a value, the operator terminal will show the input value on the button.

It is also possible to create a **Numeric Display** object to display the value stored in the controller. Therefore, for this basic numeric entry button, if you enter **10** on the operator terminal, then both the **Numeric Entry** button and the **Numeric Display** object will show 10.



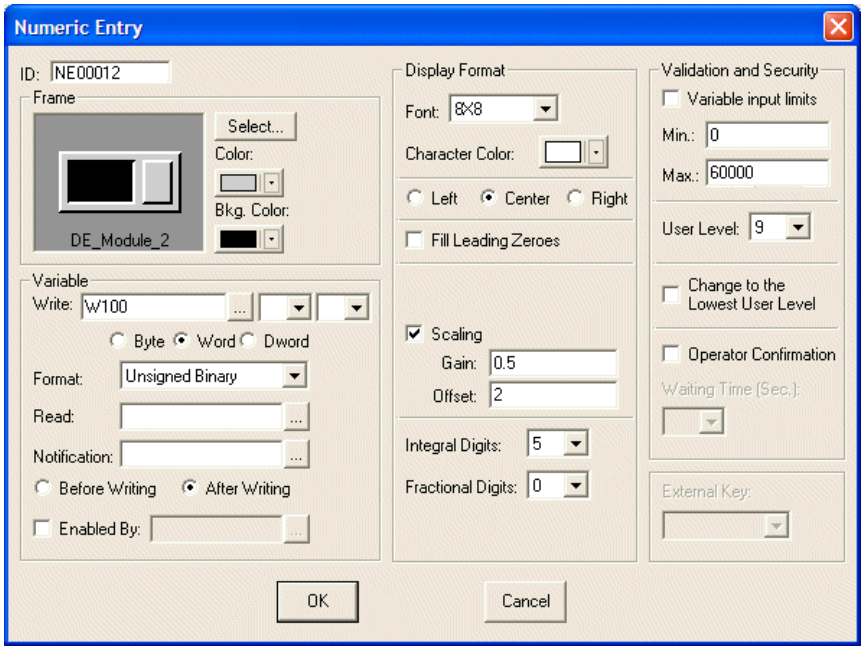
*A basic **Numeric Entry** button and a **Numeric Display** object*

Example of designing a Numeric Entry button with scaling feature

Perform the following steps to create a **Numeric Entry** button with the **Scaling** feature.

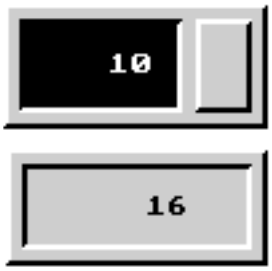
1. **Display Format**: Check **Scaling**. Select **Gain**: **0.5** and **Offset**: **2**.

All other properties are the same as in the previous example.



Numeric Entry properties in this example

After entering a value, the operator terminal will show the input value on the button. You can also create a **Numeric Display** object to display the value stored in the controller. If you enter **10** on the operator terminal, then the **Numeric Entry** button will show 10 and the **Numeric Display** object will show 16.



*A **Numeric Entry** button with **Scaling** feature and a **Numeric Display** button*

(Y = aX + b: X is the value stored in the controller, Y is the input value on an operator terminal; where a = 0.5 and b = 2 here)

2.7.5 Character Entry

The function of the **Character Entry** is used to provide you with alphabetic input and display.

When the button is clicked, an alphabetic keypad will be displayed on the screen. Enter character(s) and then press **ENT** on the keypad. The operator terminal will then write the input entry in ASCII to the specified controller register.

The object is not available on all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

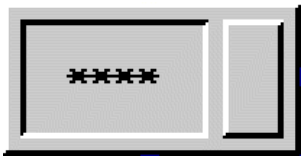
Variable

Number of Characters: Specifies the number of characters; the maximum is 28. (2 words in ASCII = 1 word in a controller register)

For properties not explained in this section, please see the sections *Set Value Button* and *Specifying Object Properties*.

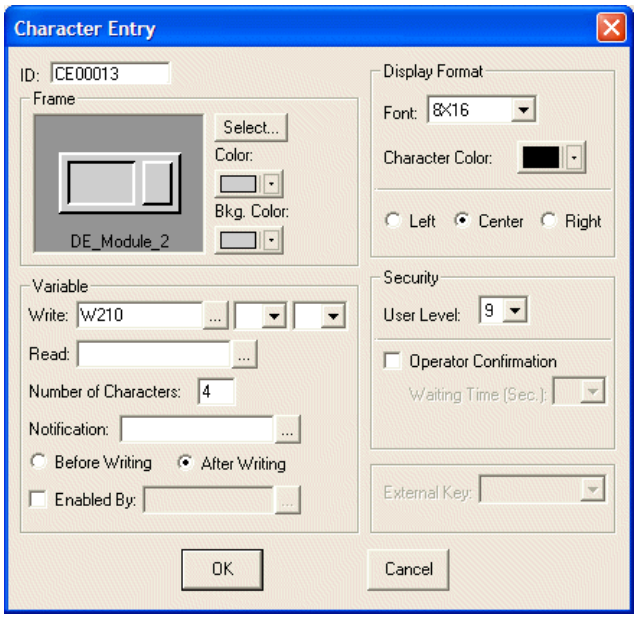
Example of designing a Character Entry button:

Perform the following steps to create a basic **Character Entry** button:



A Character Entry button

1. **Variable:** Specifiy the controller register **W210** for **Write** to store the value. (The controller model is NULL.) Select **4** for **Number of Characters**.
2. **Security**: Set the **Waiting Time (Sec.)** to **20** seconds.

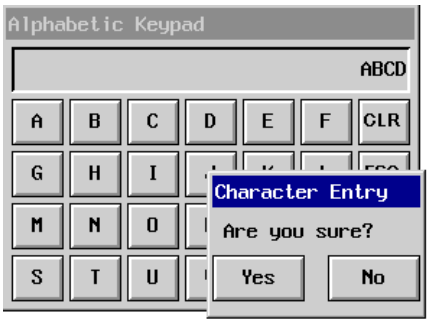


***Character Entry** properties in this example*

The steps above will create a **Character Entry** button. When the button is clicked, an alphabetic keypad will be displayed on the screen.

Note:
The **Alt**-key on the keypad is used for **Shift** functionality.

After entering the characters, press **ENT** on the keypad. A dialog box asking for user confirmation appears on the screen.



*Using the **Character Entry** button with user confirmation*

2.7.6 List

Each item in the **List** object corresponds to a designated register value in the controller. The first item represents the register value as 0. The second item represents the register value as 1, and so on. When the user chooses one of the items in the list, the operator terminal will store the corresponding value in the controller register.

The corresponding item will be highlighted in the **List** object. Furthermore, you can change the value of a controller register by making a selection from the **List** object.

The object is not available on all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

Variable

Read Only: For display purposes. The user is not able to make a selection from the list.

Write: Writes the value to the specified controller register.

Type of State:

- **Value**: There are 256 states (0-255). The value of 0 represents state 0; the value of 1 represents state 1, and so on.
- **LSB**: There are 16 states. If more than 2 bits are to be ON, the controller register will store the value of the lower bit.

Format: This is only available when the **Value** option is selected. There are three selections: **BCD**, **Signed Binary** and **Unsigned Binary**.

Read: Reads the value from the specified controller register. If the location is not specified, then the operator terminal reads from the **Write** location.

Scrollbar Size: Determine the size of the scrollbar; **Small**, **Medium** or **Large**.

For properties not explained in this section, please see the section *Specifying Object Properties*.

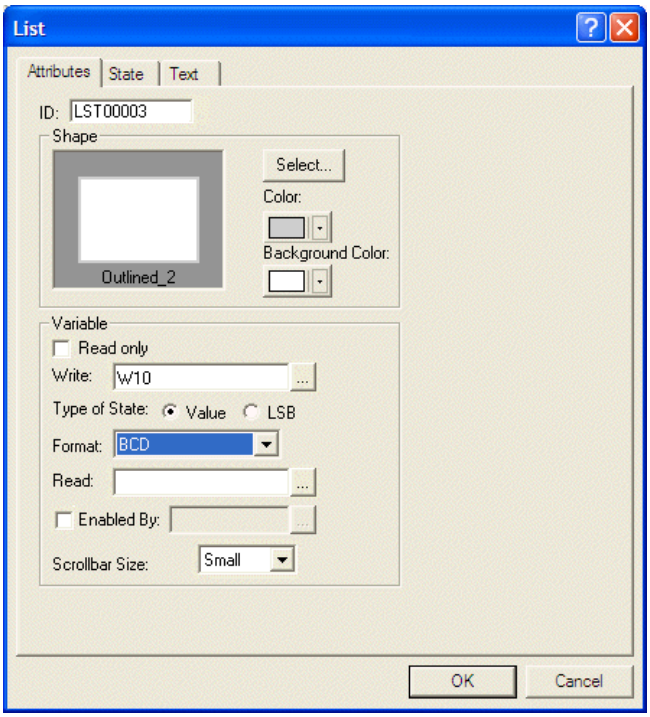
Example of designing a List object

Perform the following steps to create a **List** object:

*A **List** object*

On the **Attributes** tab of the **List** object:

1. **Shape:** Select **Outlined_2** and **White** for background color.
2. **Variable:** Specifiy the controller register **W10** for **Write** to store the value. (The controller model is NULL.) Select **Value** for **Type of State** and **BCD** for **Format**.



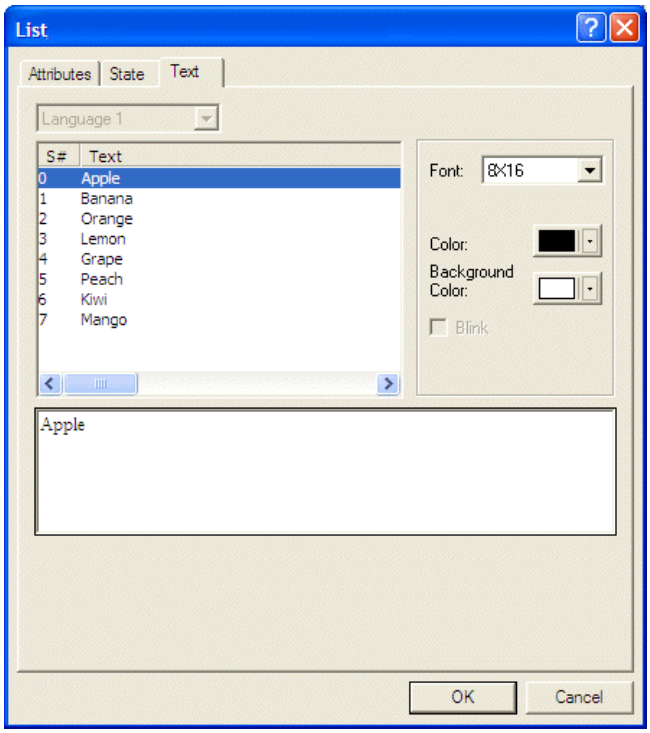
List properties in this example

On the **State** tab of the **List** object:

3. Add new states. There are 8 states in this object.

On the **Text** tab of the **List** object:

4. Enter the text and set up the format of the display.



*Adding one text for each state on the **Text** tab of the **List** object*

The steps above will create a **List** object. When an item from the **List** object is selected, the operator terminal will write the value associated with the item to the specified controller register. In this example, if the item **Peach** is selected, the value of the controller register will be 5.

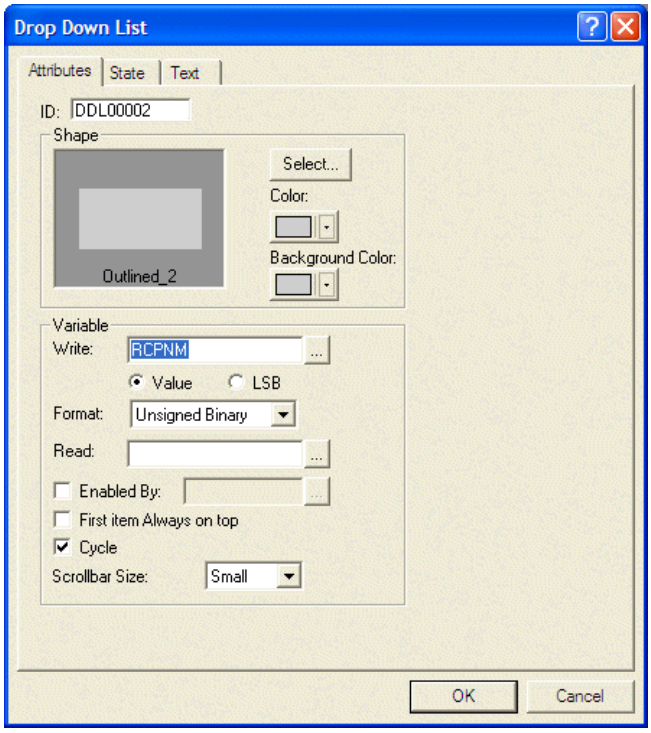
*The **List** object and the display of the controller register value of the item*

2.7.7 Drop Down List

Each of the items in the **Drop Down List** object corresponds to a value of a controller register. Therefore, for a **Drop Down List** object, the value associated with the displayed item is the current value of the controller register.

Click the object to display the list. A list of items to choose from is dropped down. You can also change the value of a controller register by making a selection from the **Drop Down List** object. Once a selection is made from the object, the list will disappear.

When selecting **RCPNM** as the **Write Variable**, the drop down list will display recipe names. You will then be able to select **First Item Always on Top** or **Cycle**. By default the drop down list includes 3 names (0 - 2). If you have more than 3 recipes in the project, you will need to add more states to show all the names.



RCPNM used as Write Variable

The object is not available on all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

The properties of the **Drop Down List** object are similar to those of the **List** object; please see the sections *List* and *Specifying Object Properties*.

Example of designing a Drop Down List object:

The steps used to create a **Drop Down List** object are similar to those for a **List** object. Please see the *Example of designing a List object* for the complete details. Remember to adjust the length of a list accordingly so that you are able to display all items of the list.

*A **Drop Down List** object*

The example below shows a **Drop Down List** object in the operator terminal. Click the object to display the list. The object will then drop down a list of items from which you can choose. When you select an item from the list, the operator terminal will write the value associated with the item to the controller register. Therefore, if the item **Peach** is selected, then the value of the controller register will be **5**.

Notice that once a selection is made from the object, the list will disappear.

*The **Drop Down List** object and the display of the register value of the item*

2.7.8 Indicators

There are two types of indicators; the **Multistate Indicator** and the **Range Indicator**.

Multistate Indicator

The **Multistate Indicator** is used to indicate which state exists with text and/or graphics. Therefore, as the operator terminal reads the contact status or the register value from the controller, it can automatically display the corresponding designed content on the operator terminal screen according to the indicator.

The number of states is as follows:

1. **Bit**: The maximum number of states is 2
2. **Value**: The maximum number of states is 256
3. **LSB**: The maximum number of states is 16

Variable:

Read: Specifies the register/bit location.

– **Bit**: Two states in all. (You can input more than two states but only two states will be displayed in the operator terminal.)

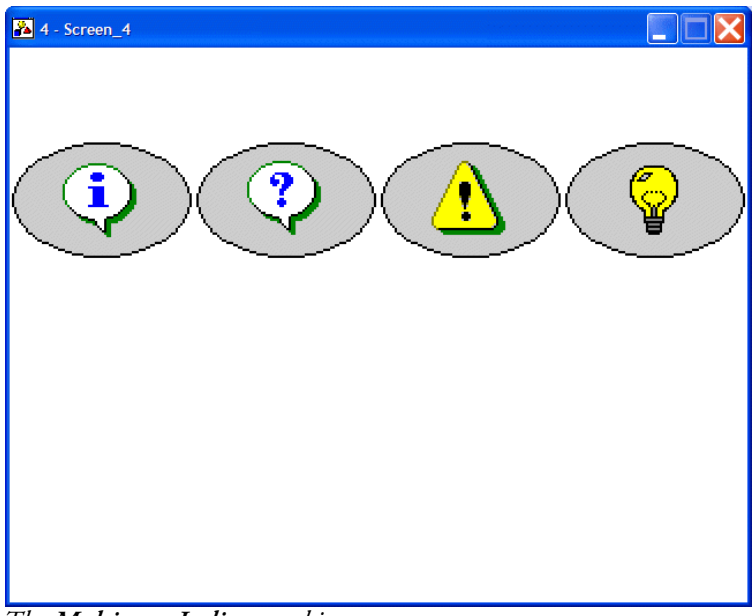
– **Value**: 256 states (0-255) in all ; value 0 represents state 0; value 1 represents state 1; value 2 represents state 2, and so on.

– **LSB**: 16 states in all; the operator terminal takes the bit number of the smallest bit that is on as the state number.

Format: Specifies the data format. There are three options: **BCD**, **Signed Binary**, and **Unsigned Binary**.

For properties not explained in this section, please see the section *Specifying Object Properties*.

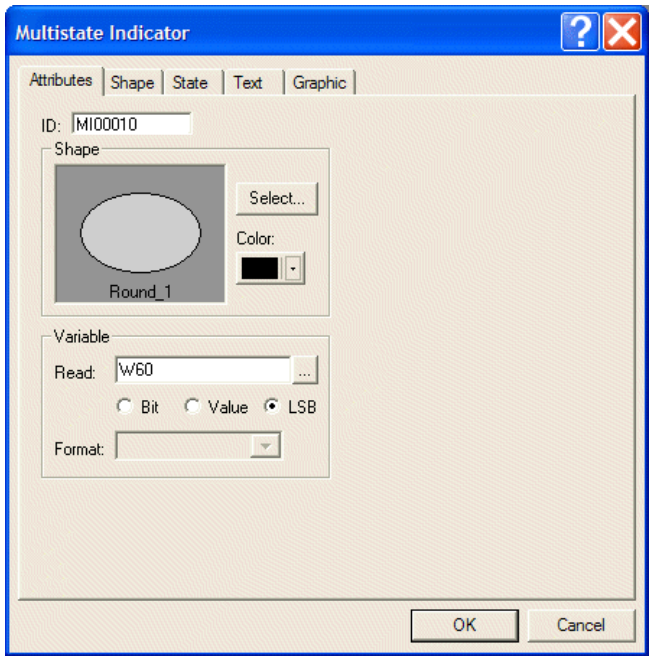
Example of designing a Multistate Indicator object



*The **Multistate Indicator** object*

On the **Attributes** tab of the **Multistate Indicator** object:

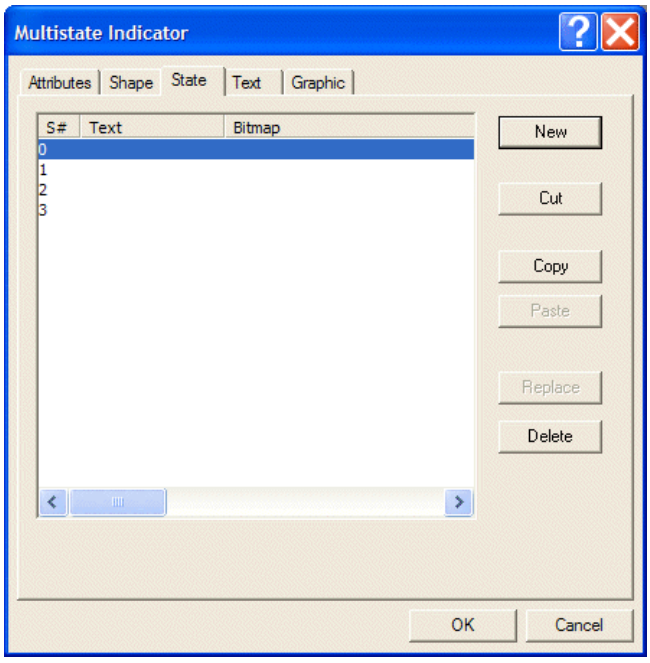
1. **Shape**: Select **Round_1** and **Black** for border color.
2. **Variable**: Specify controller register **W60** to read from. (The controller model is NULL). Select **LSB** for **Format**.



*The **Multistate Indicator** attributes in this example*

On the **State** tab of the **Multistate Indicator** object:

3. Add new states. There are 4 states in this object.



*Adding states for the **Multistate Indicator***

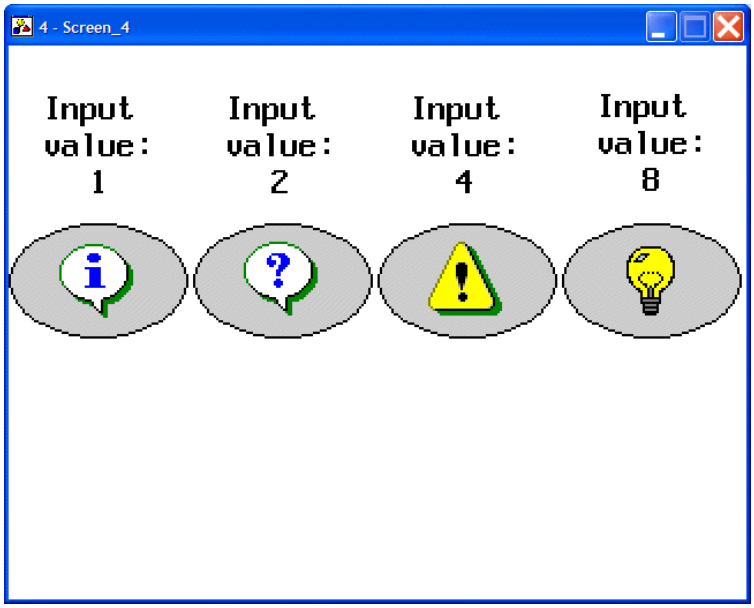
On the **Graphic** tab of the **Multistate Indicator** object:

4. Select bitmaps for the different states. This example does not display text, but graphics. These graphics are available in the SYMBOLS.GBF library.

On the **Text** tab of the **Multistate Indicator** object:

5. Select **White** for background color.

The above-mentioned steps will create a **Multistate Indicator** object. In this example, **Numeric Entry** buttons are created for the numeric entry written in the controller register. The **Multistate Indicator** objects will display the corresponding states according to the register value.



*The **Multistate Indicator** object displays the corresponding graphic*

Therefore, for this example of the object, if you enter **1** in the operator terminal, the **Multistate Indicator** object will show state 0; if you enter **4**, the object will show state 2 ; if you enter **8**, the object will show state 3.

The **Format** selected in this example is **LSB**; please refer to the following table:

Numeric Entry (LSB)	Bit State	Multistate Indicator	Graphics
1	0 bit is ON; the others are OFF	State 0	
2	1 bit is ON; the others are OFF	State 1	
4	2 bit is ON; the others are OFF	State 2	
8	3 bit is ON; the others are OFF	State 3	

Range Indicator

A **Range Indicator** displays one of several indicator labels depending on the register value. The operator terminal reads register values from the controller and automatically calculates the difference according to the boundary value of current states. Then the contents of current status are displayed on the operator terminal screen according to the calculated results.

Read value from controller ⟶ Calculated result ⟶ Display the corresponding states

Ranges:

Variable Limits: Specifies the minimum value of the ranges to be read from registers following the read location. If the **Read** address is **Wn**, the minimum value of Range # 0 is stored in **Wn+1**, the minimum of Range # 1 is stored in **Wn+2**, and so on.

Constant Limits: The minimum of the ranges is constant.

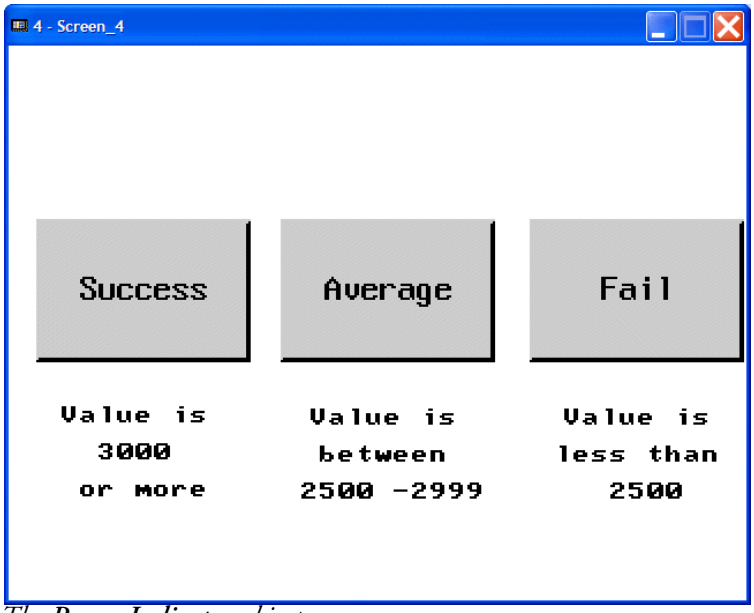
– **Range #**: The number of ranges, 15 ranges at most.

– **Minimum**: The minimum of ranges.

Note:
Number of ranges = number of states -1.

For properties not explained in this section, please see the section *Specifying Object Properties*.

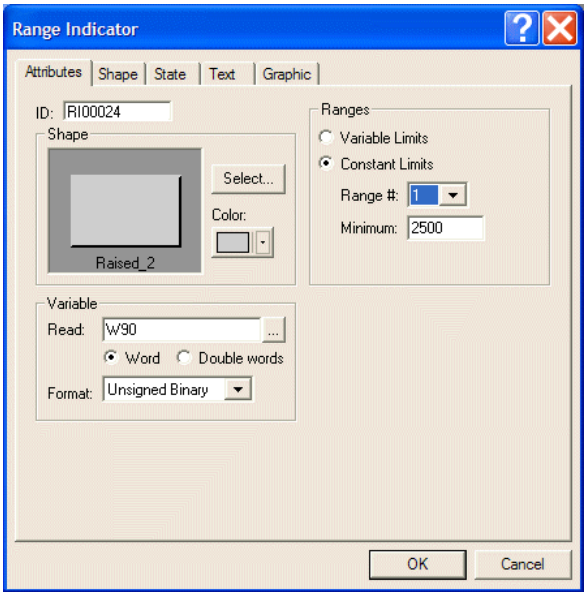
Example of designing a Range Indicator object



*The **Range Indicator** object*

On the **Attributes** tab of the **Range Indicator** object:

1. **Shape**: Select **Raised_2**.
2. **Variable**: Specify controller register **W90** to read from. (The controller model is NULL).
3. **Ranges**: Select **Constant Limits** and minimum **3000** of **Range # 0**; minimum **2500** of **Range # 1**.



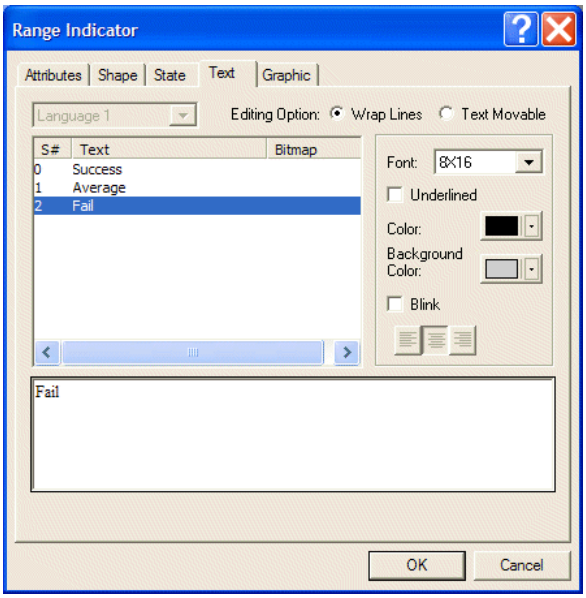
*The **Attributes** tab of the **Range Indicator** in this example*

On the **State** tab of the **Range Indicator** object:

4. Add new states. There are 3 states in this object.

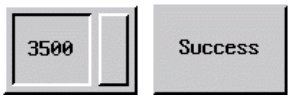
On the **Text** tab of the **Multistate Indicator** object:

5. Enter the texts for the states.



Entering texts for the different states

The previous steps will create a **Range Indicator** object. In this example, a **Numeric Entry** button is used to input value in the controller register, and a **Range Indicator** object is used to calculate the result and display its corresponding state associated with the specified range.



*The **Range Indicator** object displays the corresponding state*

If you enter **3500** as an input value, the corresponding range is **Range # 0**. Therefore, the **Range Indicator** object will show **Success**.

2.7.9 Numeric Display

The **Numeric Display** object is used to display the register value stored in controller. This object does not support click-button functionality.

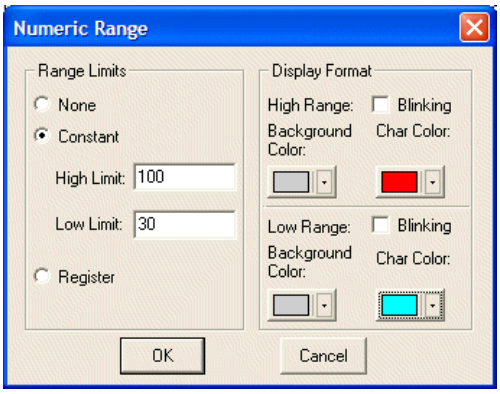
Variable

Read: Specifies a register/bit location.

Format: There are **BCD**, **Signed Binary**, **Unsigned Binary**, **Hexadecimal**, **32-bit Floating-point** and **Octal** formats.

Range

Edit button: When clicking this button, the displayed dialog box provides a display showing high/low range.



*Editing **Numeric Range***

– **None**: No high/low limit.

– **Constant**: Enter the constant variable of high/low limit.

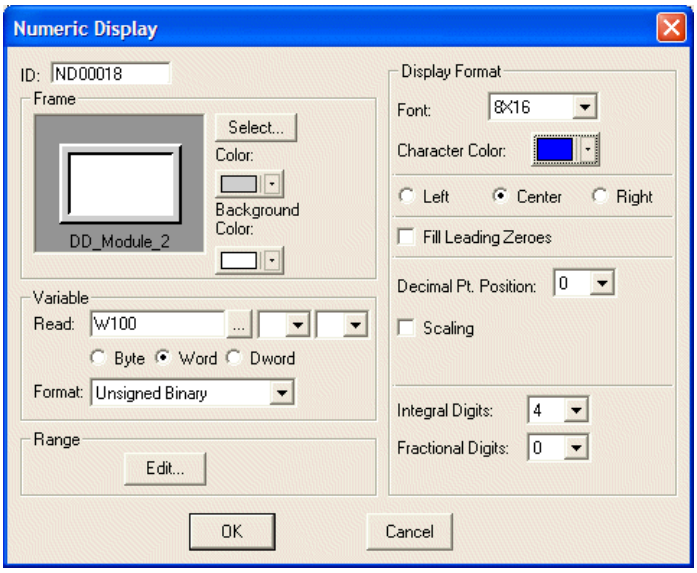
– **Register**: Read high/low limit from register. If the read location is **Wn**, the high limit is stored in **Wn+1** and the low limit is stored in **Wn+2**.

– **Display Format**: Specifies the format to display when the variable is equal or more/less than high/low limit.

For properties not explained in this section, please see the sections *Set Value Button* and *Specifying Object Properties*.

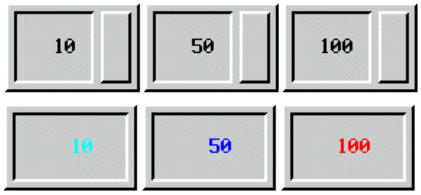
Example of designing a Numeric Display object

1. **Frame**: Select **DD_Module_2** and **White** for background.
2. **Variable**: Specify controller register **W100** to read from. (The controller model is NULL)
3. **Display Format**: Select **Blue** for character color.



Numeric Display properties in this example

4. **Range**: Click the **Edit** button and select **Constant**. Specify **100** for **High Limit** and **30** for **Low Limit**. Select **Red** for character color for **High Range**, and **Light Blue** for character color for **Low Range**.



Numeric Entry buttons are used to influence Numeric Display objects.

In this example, a **Numeric Entry** button is used to input a value in the controller register. The variable will display different text colors on the screen according to its range: If the variable is less than or equal to **30**, it shows **Light Blue** text; if the variable is greater than or equal to **100**, it shows **Red** text ; if the variable is between **30** and **100**, it shows the original setting **Blue** text.

2.7.10 Character Display

The **Character Display** object is used to provide an alphanumeric display for an ASCII variable in the controller register.

Note:
Click-button functionality is not supported.

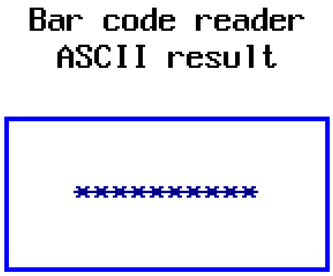
Variable

Number of Characters: Specifies the number of characters to display. It can have up to 28 characters, limited by the width of the object.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing a Character Display object

The following steps are used to create a **Character Display** object.



A Character Display object

1. **Frame**: Select **Outlined_2** , **Blue** for border color and **White** for background.
2. **Variable**: Specify controller register **W20** to read from. (The controller model is NULL). Specify **10** for **Number of Characters**.
3. **Display Format**: Select **Dark Blue** for character color.

2.7.11 Message Display Objects

There are six types of **Message Display** objects: **Prestored Message**, **Moving Sign**, **Data Terminal**, **Time Display**, **Data Display** and **Day-of-Week Display**.

Note:
Message Display objects contain only text; **Indicator Buttons** can have both text and graphic, however.

Prestored Message Display

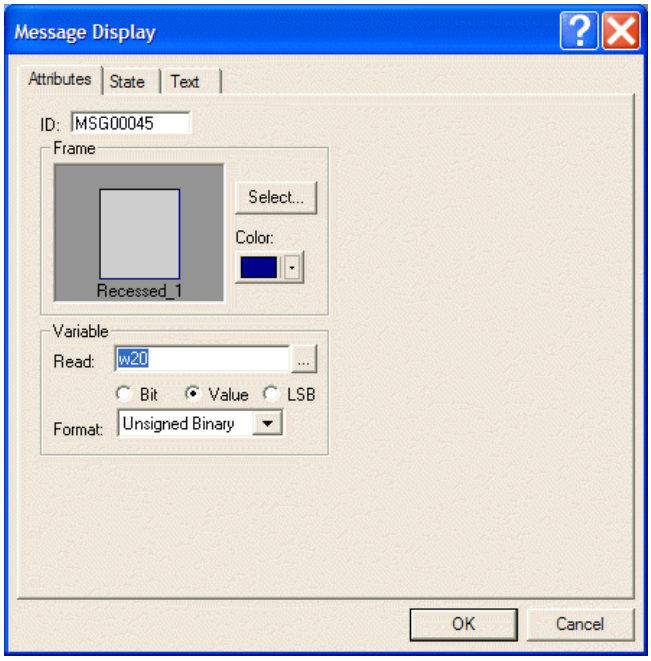
The **Prestored Message Display** object is used to make the operator terminal read the contact state (ON/OFF) or the register value from the controller and automatically display designed content on the operator terminal screen according to the state/value.

Please see the section *Multistate Indicator* for complete details.

Example of designing a Prestored Message Display object

On the **Attributes** tab of the **Message Display** object:

1. **Frame**: Select **Recessed_1** and **Dark Blue** for border color.
2. **Variable**: Specify **W20** to read from.



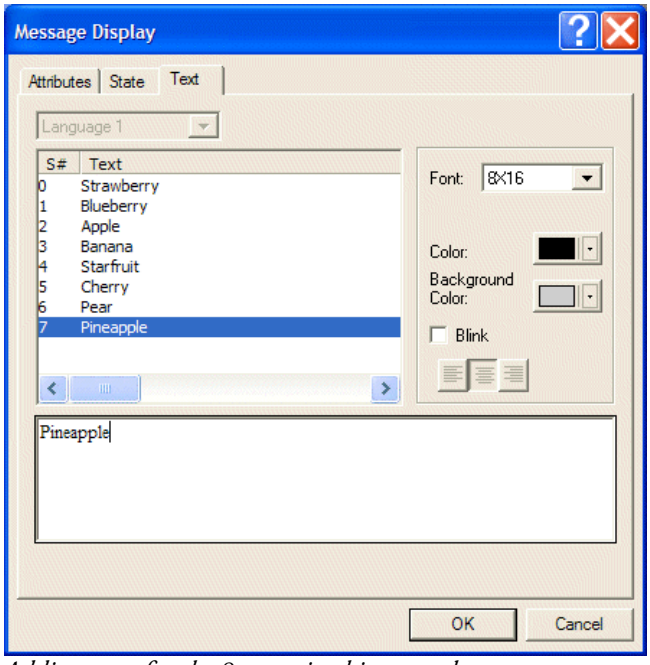
*Setting attributes for the **Message Display** object in this example*

On the **State** tab of the **Message Display** object:

3. Add 8 new states.

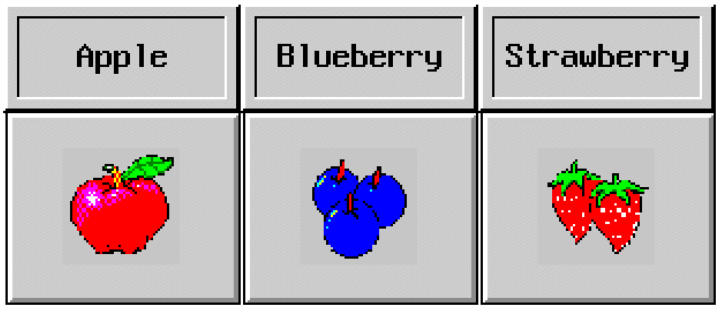
On the **Text** tab of the **Message Display** object:

4. Enter the desired text and specify the text format.



Adding texts for the 8 states in this example

The above-mentioned steps will create a **Prestored Message Display** object. When the **Multistate** button (under the **Prestored Message Display** object) is clicked once, the operator terminal writes the command to the controller for state change. When the state is changed, the **Prestored Message Display** will display the corresponding state.



***Prestored Message Display** objects used together with **Multistate** buttons*

For instance, when the state is **Blueberry**; the **Prestored Message Display** object will display the corresponding content **Blueberry**. When the state is **Starfruit**, the **Prestored Message Display** object will display the corresponding content **Starfruit**.

Moving Sign

The **Moving Sign** object is used to display content one by one, from right to left.

When the operator terminal reads the value from a bit-location (ON/OFF) or register in the controller, the **Moving Sign** object will display its contents or message according to the corresponding state on the screen.

Speed

Number of Characters Per Shift: Specifies the number of characters per shift.

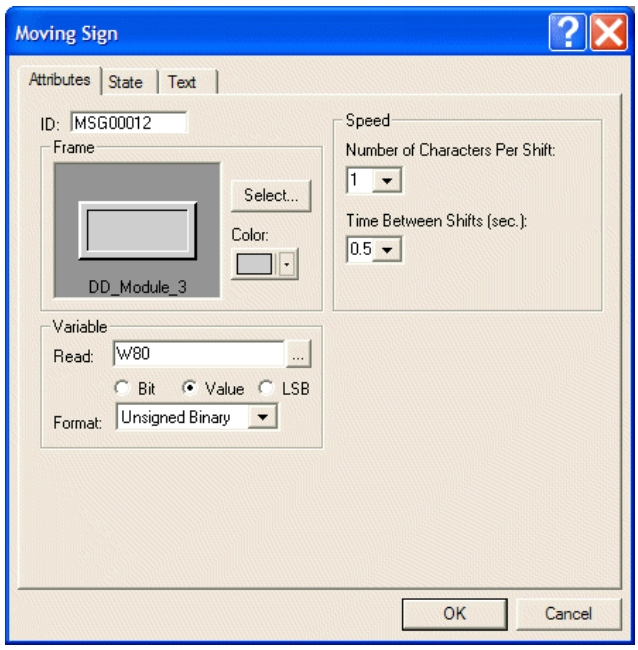
Time Between Shifts (sec.): Specifies the time between shifts in seconds.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing a Moving Sign object

On the **Attributes** tab of the **Moving Sign** object:

1. **Frame**: Select **DD_Module_3**.
2. **Variable**: Specify **W80** for **Read**. (The controller model is NULL). Select **Value** and **Unsigned Binary** format.
3. **Speed**: Select **1** for **Number of Characters Per Shift** and **0,5** seconds for **Time Between Shifts**.



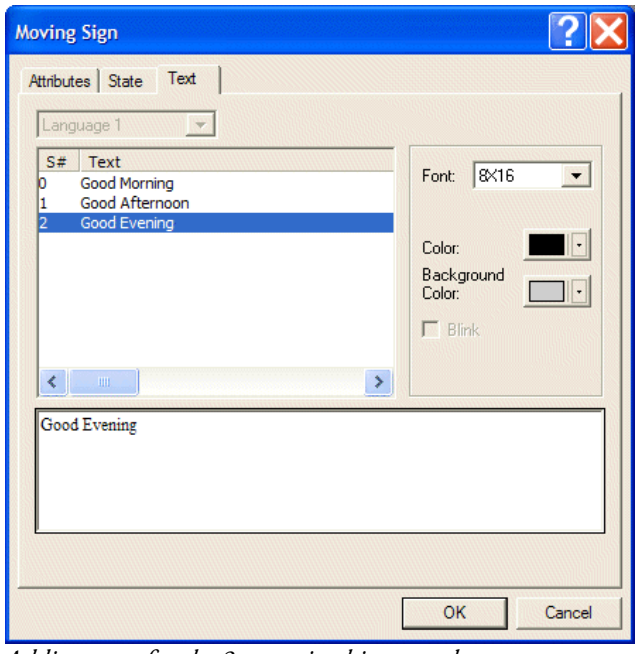
*Setting attributes for the **Moving Sign** object in this example*

On the **State** tab of the **Moving Sign** object:

4. 3 states are used in this example.

On the **Text** tab of the **Moving Sign** object:

5. Enter the desired text and specify the text format.



Adding texts for the 3 states in this example

The above-mentioned steps will create a **Moving Sign** object.

*A **Moving Sign** object used together with a **Multistate** button*

The **Multistate** button is designed to write the command to the controller when clicked. The **Moving Sign** object will display the corresponding content according the current state.

For example, if the state is **Morning**, the **Moving Sign** object will display **Good Morning Good Morning Good Morning**. This text is revolving, moving the characters from right to the left.

Data Terminal

The **Data Terminal** object is used to simulate an ASCII terminal. The operator terminal can be connected with another specified communication port and the specified communication parameters to the data terminal displayed using ASCII /HEX mode.

Remember to set the ASCII Device to communicate and specify the communication port.

The object is not available on all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

Variable

Read: Specifies a bit-location to read from. The ASCII Device provides **RX**, **RXSTS**, **TX** and **TXSTS** contacts.

Display

Mode: Displays the terminal data in ASCII/HEX mode.

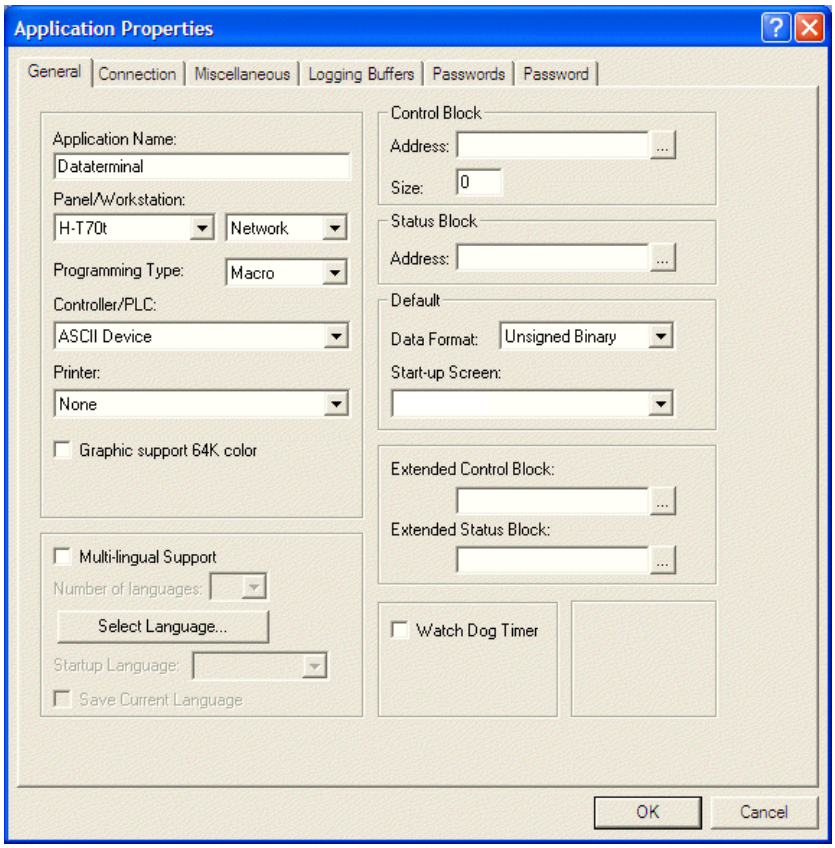
Data Buffer

Type: Selecting **Local** displays the current terminal data, but the last displayed data is not included when the screen is changed. Selecting **Global** displays the terminal data, including the last one when the screen is changed.

Size: Specifies the number of rows of terminal data.

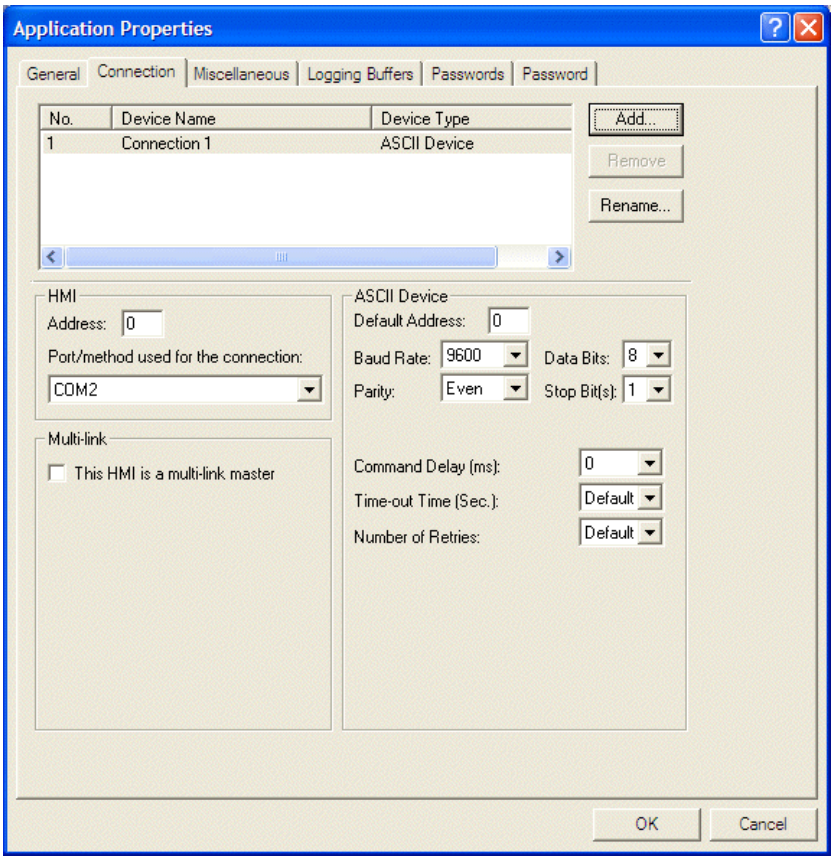
Example of designing a Terminal Data object

1. Click **Application/Workstation Setup** to set **ASCII Device** as **Controller/PLC** on the **General** tab of the **Application Properties** dialog box.



*The **General** tab of the **Application Properties** dialog box*

2. Specify the port/method used for the connection on the **Connection** tab.



*The **Connection** tab of the **Application Properties** dialog box*

Data Teminal attributes:

3. **Shape**: Select **Outlined_2** and **Black** for color.
4. **Variable**: Select **RX**.(The controller is an ASCII Device)
5. **Display**: Select **ASCII** mode to display terminal data.
6. **Data Buffer**: Select **Local**.

The above-mentioned steps will create a **Data Terminal** object that displays terminal data in ASCII mode.

Time Display

The **Time Display** object is used to make the operator terminal read the time value of the internal real time clock (RTC) and to display the content directly on the operator terminal screen.

Display Format

HH:MM:SS: Displays Hours:Minutes:Seconds

HH:MM: Displays Hours:Minutes

Example of a Time Display object

See section *Example of the Time, Date and Day-of-Week Display objects*.

Date Display

The **Date Display** object is used to make the operator terminal read the date value of the internal real time clock (RTC) and to display the content directly on the operator terminal screen.

Display Format

MM/DD/YY: The format is Month/Date/Year.

DD/MM/YY: The format is Date/Month/Year.

DD.MM.YY: The format is Date.Month.Year.

Example of a Date Display object

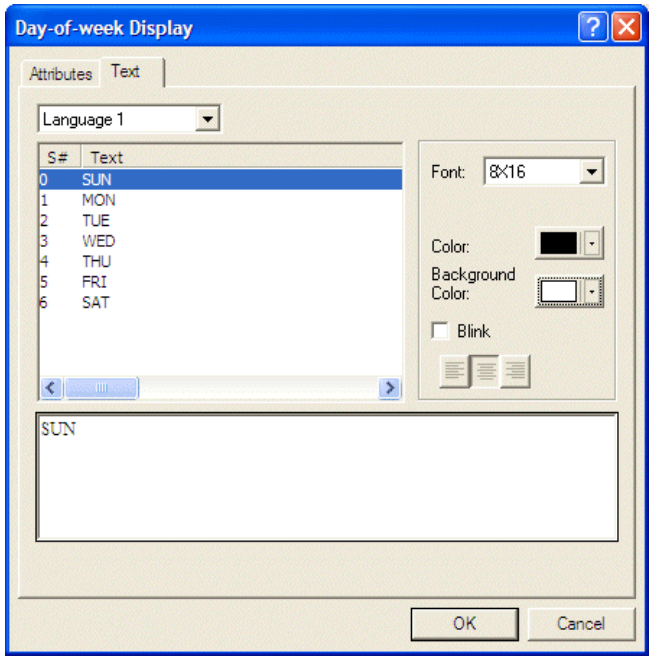
See section *Example of the Time, Date and Day-of-Week Display objects*.

Day-of-Week Display

The **Day-of-Week Display** object is used to make the operator terminal read the day of week value from the internal real time clock (RTC) and to display the content directly on the operator terminal screen.

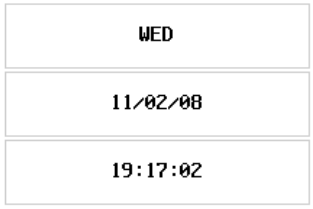
The object is not available on all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

The operator terminal will display the date of week automatically.



*The **Text** tab of the **Day-of-Week Display** object*

Example of the Time, Date and Day-of-Week Display objects



*The **Time Display**, **Date Display** and **Day-of-Week Display** objects*

2.7.12 Bar Graph

There are two types of **Bar Graphs**; **Normal** and **Deviation**.

Normal Bar Graph

The **Normal Bar Graph** is used to make the operator terminal read the value of the controller register, to convert its data into a bar graph, and then to display the bar graph in the operator terminal.

Variable

Min.: Specifies the minimum value the bar graph can display.

Max.: Specifies the maximum value the bar graph can display.

Variable target/range limits: Select this option if the target value and the range limits are read from the controller.

The **Target Variable** is stored in the word location that follows the **Read**. The **Low Limit** is stored in the word location that follows the **Target Variable**. The **High Limit** is stored in the word location that follows the **Low Limit**.

Register	X	X+1	X+2	X+3
	Read	Target Variable	Low Limit	High Limit
Example:	W10	W11	W12	W13

Display Format

Upward, **Downward**, **Rightward**, and **Leftward**: Selects the direction of the bar graph.

Color: Specifies the color of the bar graph.

Pattern: Specifies the pattern style to display.

Target: Sets the target to display.

– **Value**: Specifies the constant target value.

– **Color**: Specifies the color of target line.

Ranges: Select this option to fill the graph with a different color when the register value is beyond a normal range.

– **Low Range**: Specifies the color to fill the graph with if the register value is equal to or less than the low range limit.

– **Limit**: Specifies a constant for the low range limit.

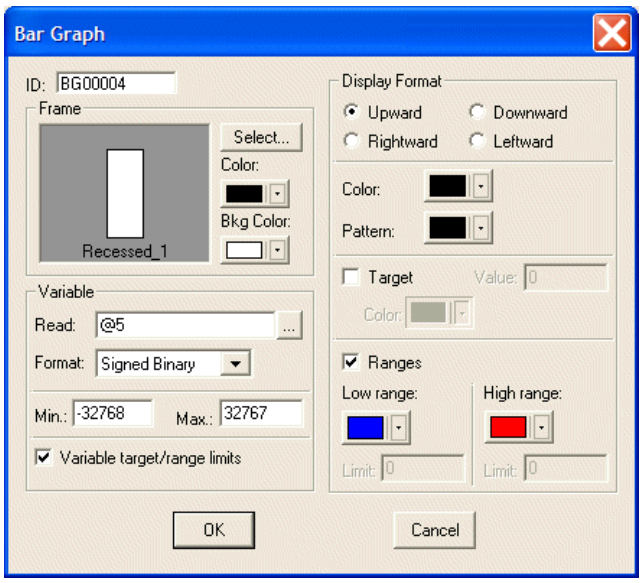
– **High Range**: Specifies the color to fill the graph with if the register value is greater than or equal to the high range limit.

– **Limit**: Specify a constant for the high range limit.

For properties not explained in this section, please see the section *Specifying Object Properties*.

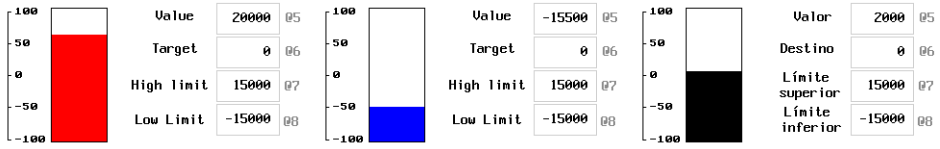
Example of designing a Bar Graph object

1. **Frame**: Select **Recessed_1** and **White** for background color.
2. **Variable**: Specify **@5** (Local Internal Memory) for **Read** and **Unsigned Binary** for **Format**.
3. Set **Min.** to **-32,768** and **Max.** to **32,767**.
4. Check the option **Variable target/range limits**.
5. Check **Ranges** and select **Blue** for **Low range** and **Red** for **High range**.



*The attributes of the **Bar Graph** object in this example*

When the register value is equal to or less than the low limit, the graph will be filled with blue color; when the register value is equal or greater than the high limit, the graph will be filled with red color.



***Bar Graphs** indicating different limits using **Numeric Entry** objects*

The **Numeric Entry** objects are used to set the high/low limits. There is a **Scale** to the left of the bar graphs. The **Low Limit** here is **-15000** and the **High Limit** is **15000** using the following configuration:

Variable ≤ -15000 fills graph with blue
Variable between -15000 and 15000 fills graph with black
Variable ≥15000 fills graph with red

Deviation Bar Graph

The **Deviation Bar Graph** is used to make the operator terminal read the values of the controller register and to compare them with the normal value. Then the operator terminal converts the difference and presents it on a **Deviation Bar Graph** in the operator terminal.

Variable

Variable Std Value/Deviation Limit: Select this option if the standard value and deviation limit are read from the controller. If **Read** address is **W10**, **Standard Value** will be stored in **W11**; **Deviation Limit** will be stored in **W12**.

Display Format

Vertical, Horizontal: Selects the direction of the bar graph.

Standard Value: Specifies the constant standard value. The standard value will be a date line on the bar graph.

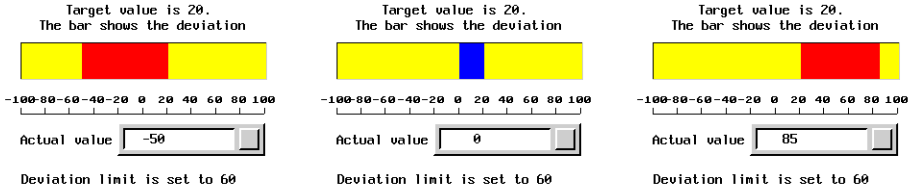
Display Deviation Limit: Select this option to fill the bar graph with selected color when the difference of the register value and the standard value is beyond the limit.

– **Limit**: Specifies the constant limit.
Difference Value: =¦Variable - Standard Value¦

– **Color**: Specifies the color to fill the graph with when the difference of the register value and the standard value is beyond the limit.

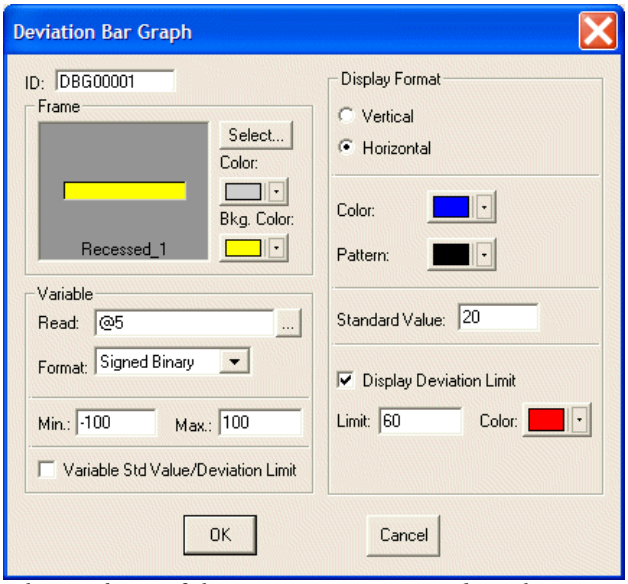
For properties not explained in this section, please see the sections *Normal Bar Graph* and *Specifying Object Properties*.

Example of designing a Deviation Bar Graph



*A **Deviation Bar Graph** shows the difference to the set standard value*

1. **Frame**: Select **Recessed_1** and **Yellow** for background color.
2. **Variable**: Specify **@5** (Local Internal Memory) for **Read** and **Signed Binary** for **Format**.
3. Set **Min.** to **-100** and **Max.** to **100**.
4. **Display Format**: Select horizontal direction for the **Deviation Bar Graph**.
5. The **Standard Value** is **20** on this deviation bar graph.
6. Specify that the **Red** color is to fill the graph when the difference between the register value and the standard value is beyond the limit **60**.



*The attributes of the **Deviation Bar Graph** in this example*

2.7.13 Trend Graph

The **Trend Graph** is used to read a series of values from the related controller register. The operator terminal converts these values and presents them in a trend graph in the operator terminal.

If the register is **Wn** to be read from, with three curves in total, the data will be read as follows format:

Value in **Wn** = **m** is the real sampling points;
Value in **Wn+1** is the first point of Y direction on the curve #1;
Value in **Wn+2** is the first point of Y direction on the curve #2;
Value in **Wn+3** is the first point of Y direction on the curve #3;
Value in **Wn+4** is the 2nd point of Y direction on the curve #1;
Value in **Wn+5** is the 2nd point of Y direction on the curve #2;
Value in **Wn+6** is the 2nd point of Y direction on the curve #3;
:
:
and so on. The operator terminal reads all controller registers from **Wn** to **Wn+3m**.

For example, if the value in **Wn** is **m** = 25 sampling points; the operator terminal will read data from 76 (=3 × 25 + 1) controller registers.

Control

To control the trend graph via the controller.

Trigger Flag #: The trigger flag number of the trend graph is 12-15 bits in CFR. The operator terminal reads data from the controller and displays the trend graph when the trigger flag turns on.

Clear Flag #: The clear flag number of the trend graph is bits 8-11 in CFR. The operator terminal clears the trend graph when the clear flag turns on.

Continuous update: A trend graph with many points that are updated frequently may be experienced as blinking in a distubing way on the operator panel display. Checking the box improves the user experience.

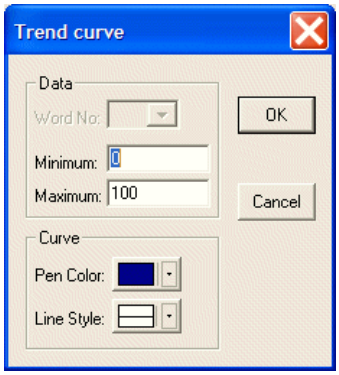
Display Format

Number of Points: Specifies the maximum number to display on the Y-direction.

Number of Grids: Specifies the number of evenly spaced horizontal grids to be displayed.

Grid Color: Specifies the color of the horizontal grids.

Curve # 1 - Curve # 4: Provides four curves for editing. When the **Edit** button is clicked, the dialog box below will appear:



*Editing **Display Format** of **Trend Curves***

Minimum: Specifies the value corresponding to the lowest point on the trend graph. When the data is equal to or less than the **Minimum**, the operator terminal places the dot at the bottom pixel of the drawing area of the trend graph.

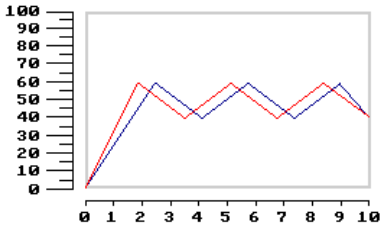
Maximum: Specifies the value corresponding to the highest point on the trend graph. When the register is equal to or greater than the **Maximum**, the operator terminal places the dot at the top pixel of the drawing area of the trend graph.

Pen Color: Specifies the color of the trend curve.

Line Style: Specifies the line style of the trend curve.

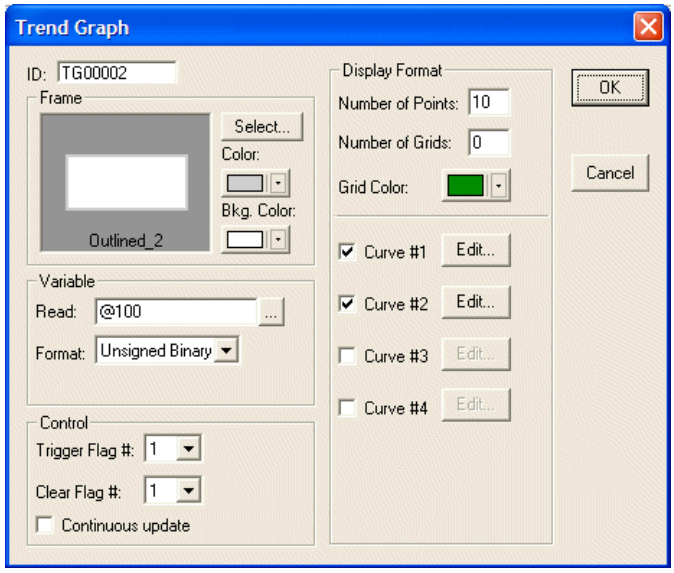
For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing a Trend Graph object



A Trend Graph

1. **Frame**: Select **Outlined_2** and **White** for background color.
2. **Variable**: Specify **@100** (Local Internal Memory) for **Read** and **Unsigned Binary** for **Format**.
3. **Control**: Select **1** for **Trigger Flag #** and for **Clear Flag #**.
4. **Display Format**: Specify **10** for **Number of Points**.



***Trend Graph** attributes in this example*

5. Check **Curve #1** and click **Edit**. Set **Minimum** to **0** and **Maximum** to **100**. Select **Blue** for **Pen Color**.
6. Check **Curve #2** and click **Edit**. Set **Minimum** to **0** and **Maximum** to **100**. Select **Red** for **Pen Color**.

2.7.14 XY Chart

The XY Chart is used to read a series of values from the related controller register. Then the operator terminal converts these values and presents them on an XY chart in the operator terminal.

Suppose that the register to read from is **Wn**, with two curves in all. The data will be read as follows:

Value in **Wn** = **m** are the real sampling points;
Value in **Wn+1** is the first point of the X-axis on curve #1;
Value in **Wn+2** is the first point of the Y-axis on curve #1;
Value in **Wn+3** is the first point of the X-axis on curve #2;
Value in **Wn+4** is the first point of the Y-axis on curve #2;
Value in **Wn+5** is the 2nd point of the X-axis on curve #1;
Value in **Wn+6** is the 2nd point of the Y-axis on curve #1;
Value in **Wn+7** is the 2nd point of the X-axis on curve #2;
Value in **Wn+8** is the 2nd point of the Y-axis on curve #2;
:
:
and so on. The operator terminal reads all controller registers from **Wn** to **Wn+2m**.

For example, if the value in **Wn** is **m** = 15 sampling points, the operator terminal will read data from 61 (=2 × 2 × 15 + 1) controller registers.

Control

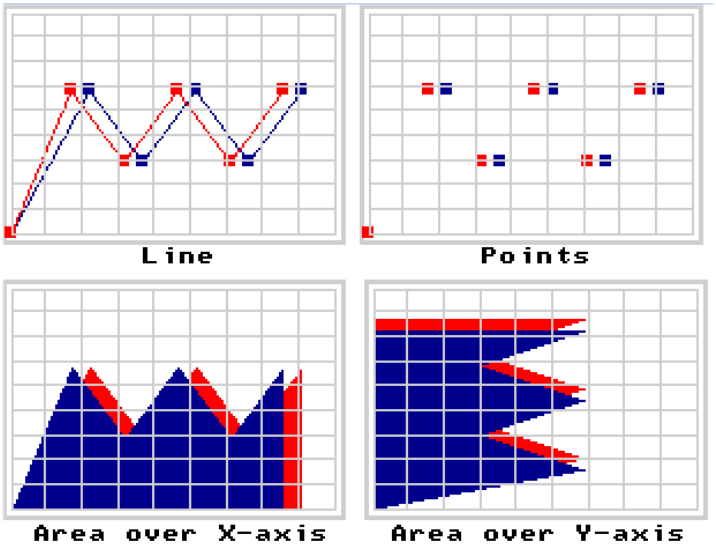
To control the trend graph by controller.

Trigger Flag #: The trigger flag number of the trend graph is bits 12-15 in CFR. The operator terminal reads data from controller and displays the XY chart when the trigger flag turns on.

Clear Flag #: The clear flag number of the trend graph is bits 8-11 in CFR. The operator terminal clears the trend graph when the clear flag turns on.

Display

Points, **Line**, **Area over X-axis** and **Area over Y-axis**: See the illustration below.



Maximum Number of Points: Specifies the maximum number of points to display on the XY chart.

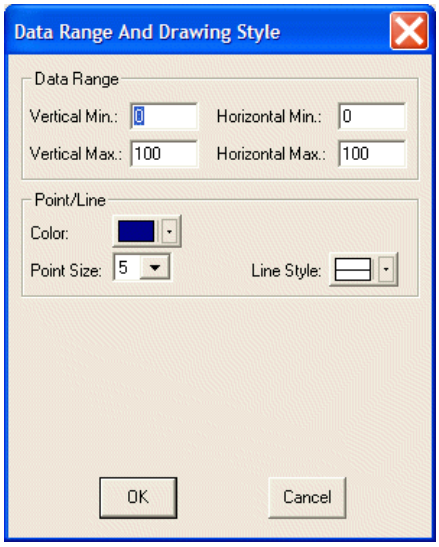
Grids:

– **Number of H Lines**: Specifies the number of horizontal lines.

– **Number of V Lines**: Specifies the number of vertical lines.

– **Color**: Specifies the color of the lines.

Data Set # 1 - Data Set # 2: When the **Edit** button is clicked, the dialog box below will appear:



– **Vertical Min.** and **Vertical Max.**: Specifies the minimum and maximum values for the Y-axis.

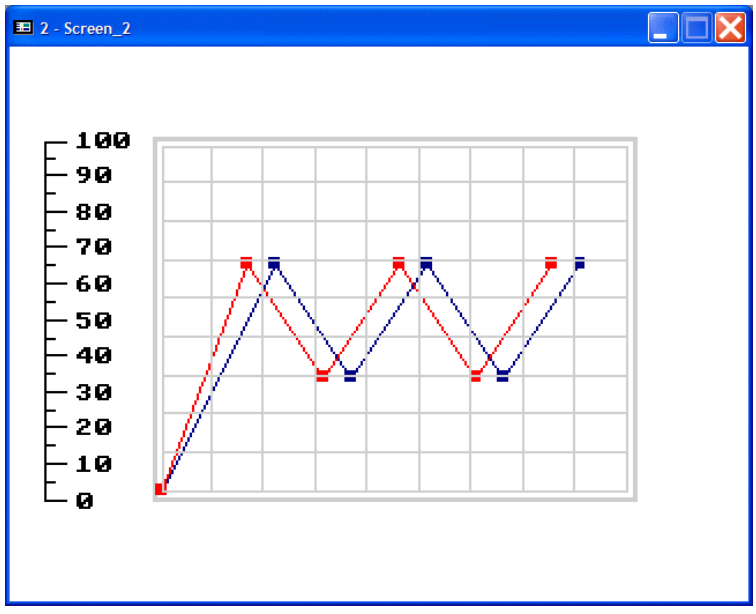
– **Horizontal Min.** and **Vertical Max.**: Specifies the minimum and maximum values for the X-axis.

– **Color**: Specifies the color for the point/line.

– **Point Size**: Specifies the size of the point to display.

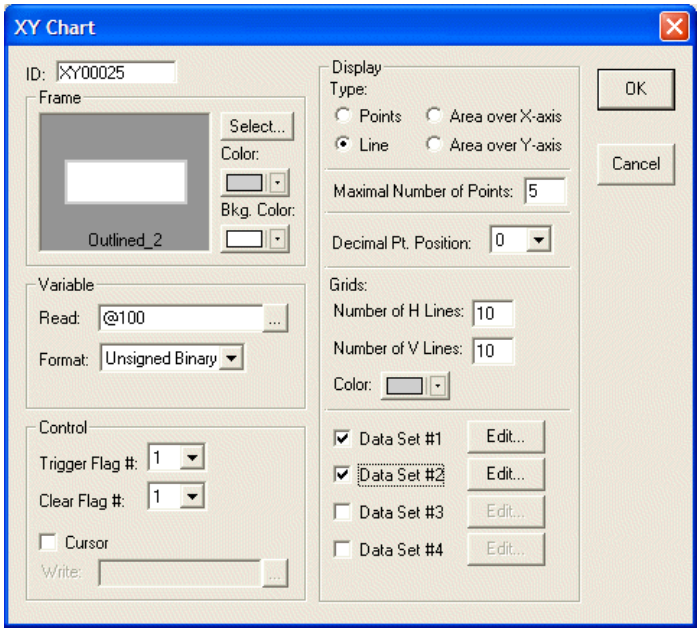
For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing an XY Chart object



An XY Chart

1. **Frame**: Select **Outlined_2** and **White** for background color.
2. **Variable**: Specify **@100** (Local Internal Memory) for **Read** and **Unsigned Binary** for **Format**.
3. **Control**: Select **1** for **Trigger Flag #** and **Clear Flag #**.
4. **Display**: Select **Line**.
5. **Maximum Number of Points**: 5.



***XY Chart** attributes in this example*

6. Check **Data Set #1** and click **Edit**. Set **Vertical Min.** and **Horizontal Min.** to **0**, and **Vertical Max.** and **Horizontal Max.** to **100**. Select **Blue** for **Point/Line Color**.
7. Check **Data Set #2** and click **Edit**. Set **Vertical Min.** and **Horizontal Min.** to **0**, and **Vertical Max.** and **Horizontal Max.** to **100**. Select **Red** for **Point/Line Color**.

2.7.15 Panel Meters

There are two types of panel meters - **Round** and **Rectangular**.

Round Panel Meter

The Round Panel Meter is used to make the operator terminal read the value from the controller register and to reflect the value on the **Round Panel Meter** object on the screen.

Needle

Color: Specifies the needle's color.

Start/End Angle (deg.): Specifies start and end of meter in degrees.

Direction of increment: Select if increment is to be clockwise or counterclockwise.

Scale

Color: Specifies the color used to display the scale.

Number of major ticks: Specifies the number of major ticks on the scale. If the number is less than 2, no ticks are displayed.

Number of minor ticks: Specifies the number of minor ticks on the scale.

Display axis: Check this option to display an arc as the axis of the scale.

Display mark: Check this option to display marks on the scale.

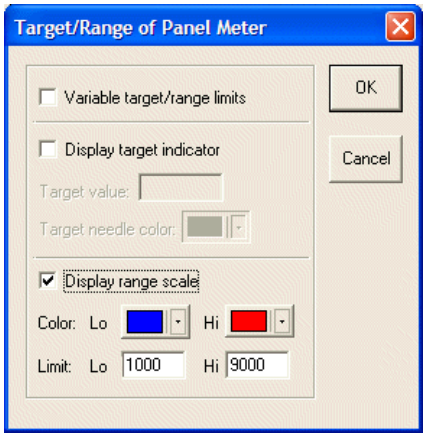
Font: **8 x 8** or **8 x 16** can be selected.

Number of digits: Specifies the number of digits, including precision and scale.

Decimal point position: Specifies the position of the mark's digit. If the number is 0, no decimal point is displayed.

Min. and **Max.**: Specifies the minimum and maximum number of marks.

Target/Range: Click the **Edit** button to display the dialog box below:



– **Variable target/range limits**: The target value and range limits are read from the controller. The target value is stored in a bit-location which is next to the **Read** location. The low limit is next to the target value. The low limit is next to the high limit. When the **Read** location is specified as **W10**, the target value is stored in **W11**; the low limit is stored in **W12**; the high limit is stored in **W13**.

– **Display target indicator**: Check this option to display target indicator.

  Target value: Specifies the target value.

  Target needle color: Specifies the color of the needle.

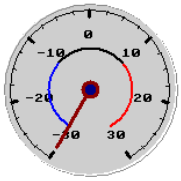
– **Display range scale**: Specifies the color of the range scale to display.

Low Range Color and **High Range Color**: Specifies the color to display on the scale when the value is less/greater than low/high range.

Low Limit and **High Limit**: Specifies the low limit and high limit constants.

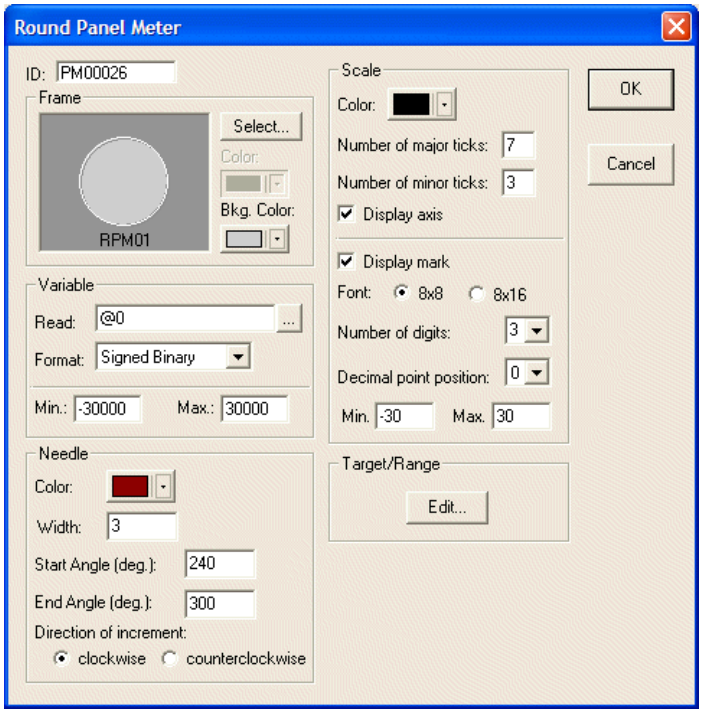
For properties not explained in this section, please see section *Specifying Object Properties*.

Example of designing a Round Panel Meter object



*A **Round Panel Meter***

1. **Variable**: Specify **@0** (Local Internal Memory) for **Read** and **Signed Binary** for **Format**.
2. Set **Min.** to **-30000** and **Max.** to **30000**.
3. **Needle**: Select **300** degrees for **Sweep Angle**.
4. **Scale**: Set **Number of Major Ticks** to 7 and **Number of Minor Ticks** to **3**.
5. Check **Display mark** and set **Min.** to **-30** and **Max.** to **30**.



*The attributes of the **Round Panel Meter** in this example*

6. Click **Edit** for **Target/Range** and check **Display range scale**. Set the **Low range limit** to **-10000** and **Blue** color; the **High range limit** to **10000** and **Red** color.

Rectangular Panel Meter

The properties of the **Rectangular Panel Meter** are the same as for the **Round Panel Meter**; please see section *Round Panel Meter*.

2.7.16 Pie Graph

The **Pie Graph** is used to make the operator terminal read the register values in the controller. Then it converts the values into a 360 degree pie graph and displays the graph on the operator terminal screen.

Display Format

Starting angle: Specifies the original angle of the pie graph.

Pie color: Specifies the color used to fill the pie graph.

Bkg color: Specifies the color of the unfilled part of the pie graph.

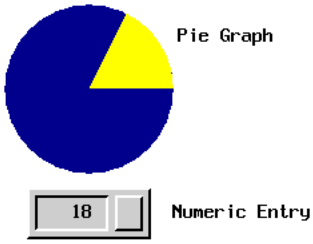
Style: Specifies the pattern style used for filling the pie graph.

Border: Check this option to display the pie graph with a border.

Color: Specifies the border color of the pie graph.

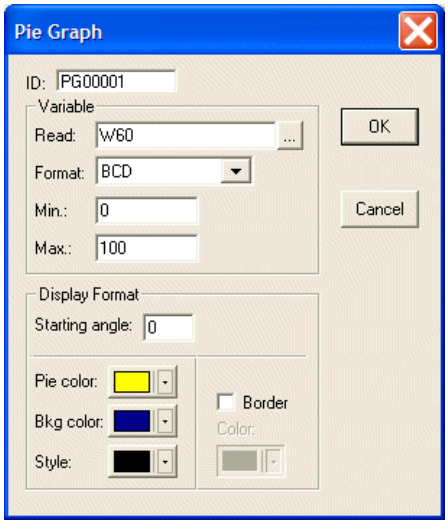
For properties not explained in this section, please see the sections *Bar Graph* and *Specifying Object Properties*.

Example of designing a Pie Graph object



*The **Pie Graph** object*

1. **Variable**: Specify **W60** for **Read** and **BCD** for **Format**.
2. Set **0** for **Min.** and **100** for **Max.**
3. **Display Format**: Set **Starting Angle** to **0**.
4. Select **Pie color**, **Bkg color**, **Style** and **Border color**.



*Attributes for the **Pie Graph** object in this example*

A **Pie Graph** object is drawn from (+) X-axis (starting angle = 0). When the input value is **20**, the area of the pie is one-fifth of a circle (= 20/100).

2.7.17 Dynamic Graphics

There are five types of dynamic graphics: **Animated Graphic**, **GIF Graphic**, **State Graphic**, **Dynamic Circle** and **Dynamic Rectangle**.

Animated Graphic

The **Animated Graphic** object enables you to control a graphic, including its position and moving-path on the operator terminal screen, whether via controller or not. For instance, the controller can control the graphic movement along the X-axis, Y-axis, or displaying different graphics.

Attributes Tab

Not Controlled by PLC: Check this option and the graph is not controlled by the controller.

Graphic State:

– **PLC controlled**: Displays the graphic states controlled by the controller.

– **Location dependent**: Displays the different states according to the location.

– **Auto change**: Changes the graphic state automatically.
Rate (Once per): Specifies the rate at which to change the graphic state.

Path:

– **PLC controlled**: Controls the object's movement path via the controller.

– **Horizontal line**: Moves the object along a horizontal line.

– **Horizontal marquee**: Moves the object along a horizontal line with marquee. Check **Duplication** to move duplicated graphics along a horizontal line.

– **Vertical Line**: Moves the object along a vertical line.

– **Vertical marquee**: Moves the object along a vertical line with marquee. Check **Duplication** to move duplicated objects along a vertical line.

– **Connected Lines**: Moves the object along the route of connected lines.

Path: Double-click the left key on the object to display the movement path.

Path Point: Right-click on the object to select **Add Path Point** or **Delete Path Point** from the drop-down list for the connected lines setup. Please see section *Example of creating horizontal/vertical line as a path* for complete details.

Note that a curve path can be configured with various path points.

– **Still**: Changes the state without movement.

Movement:

– **Rate (pixels/sec.)**: Specifies the rate of movement.

One-way: Moves the object in one-way mode.

Two-way: Moves the object in two-way mode.

Duplication: Select this option to move duplicate objects with marquee; the number of copies can be specified as well.

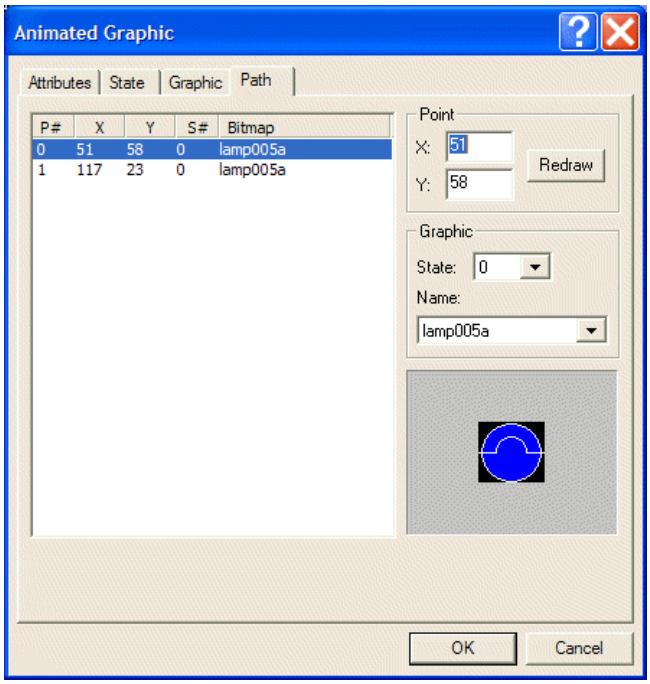
For properties not explained in this section, please see the section *Specifying Object Properties*.

State and Graphic Tab

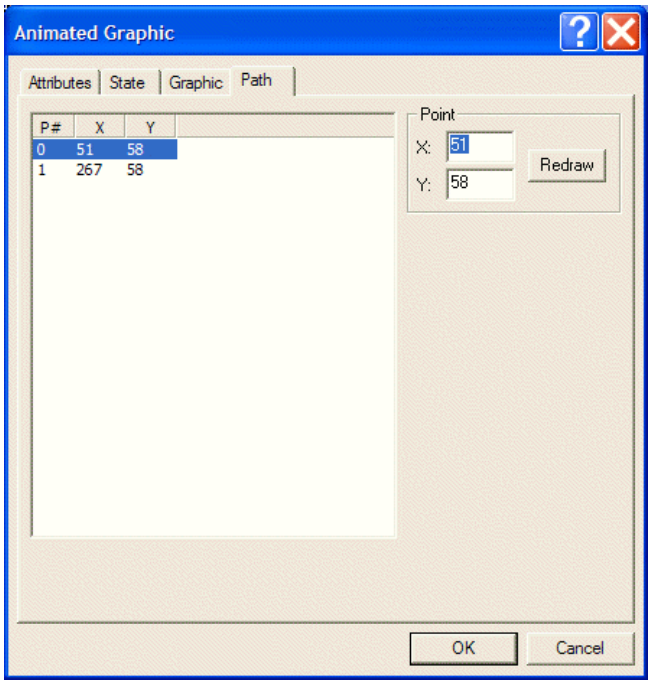
Please see section *Specifying Object Properties* for complete details.

Path Tab

– The **Graphic State** is **Location dependent**:
On the **Path** tab, this feature enables you to specify the graphic states to be changed according to different locations. For instance, the point # 0 displays the graphic state in state # 0; the point # 1 displays the graphic state in state # 1.

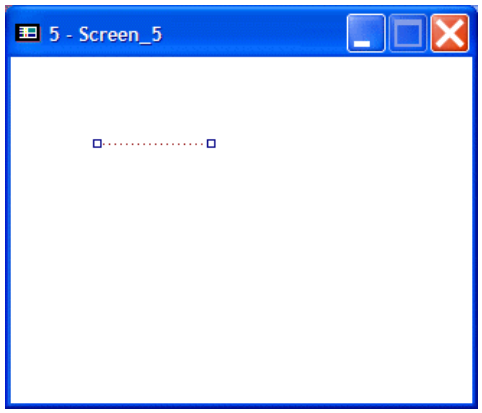


– The **Graphic State** is **Auto Change**:
On the **Path** tab, the object changes its state along the specified path. You can set the starting point and ending point of the path to be displayed on this tab.

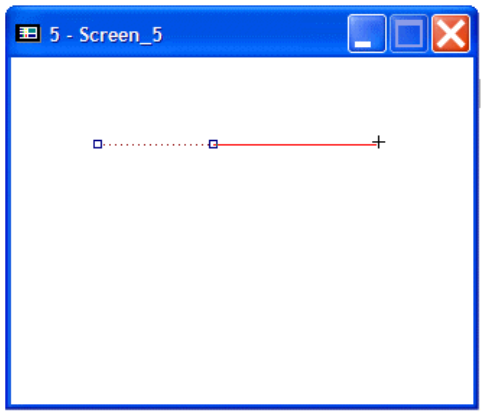


Example of creating horizontal/vertical line as a path

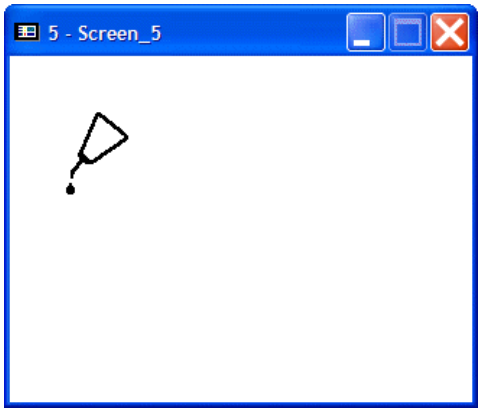
1. Double-click on the object to display the movement path.



2. Move the cursor to the point, then click to draw the movement path. The revised path is marked with a red line.

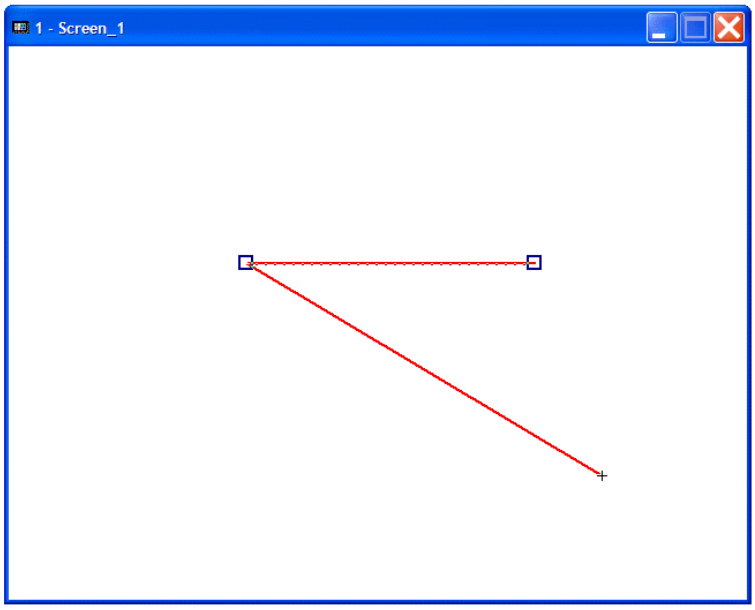


3. Click elsewhere on the screen to display the object's graphics.

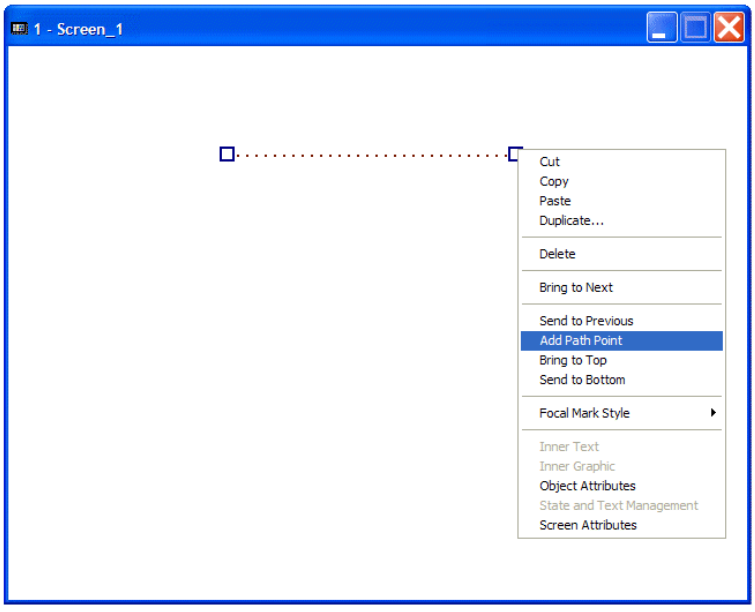


Example of creating connected lines as a path

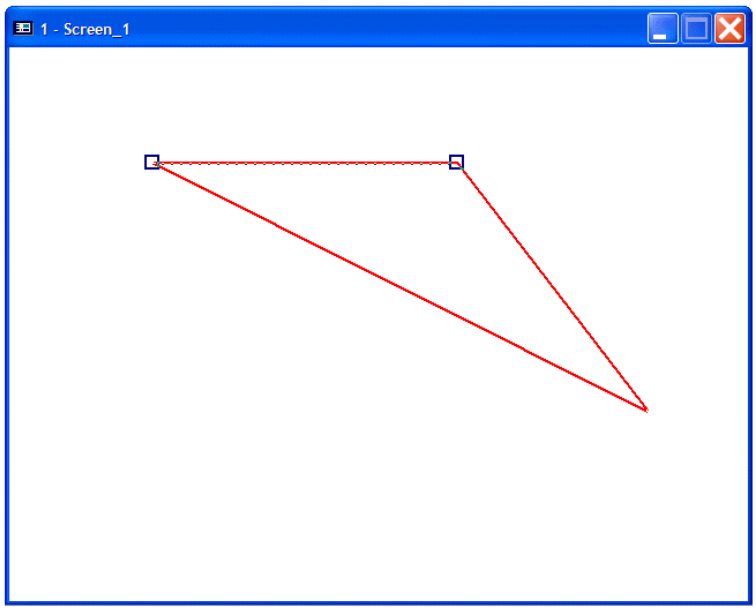
1. Double-click on the object to display the movement path.
2. Click on the points of the connected lines to draw the movement path. The revised path is marked with a red line.



3. Move the cursor to any one of the points and right-click to select **Add Path Point** or **Delete Path Point** from the drop-down list.



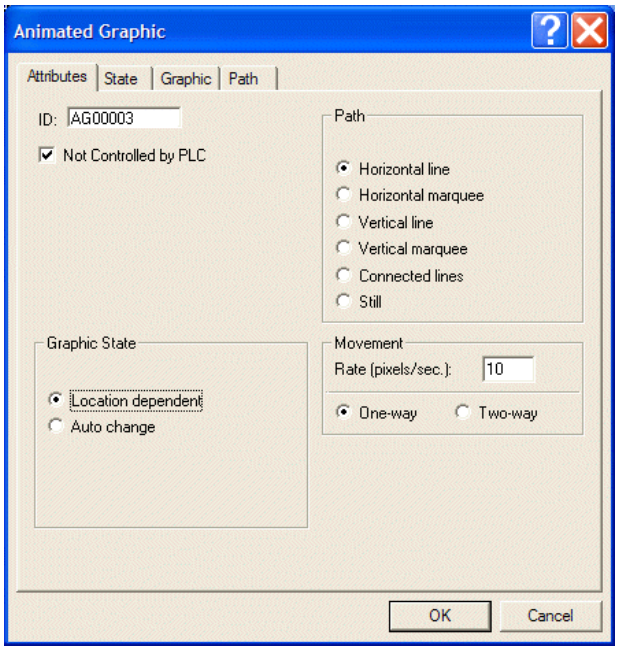
4. In the same fashion, the way to edit a new path is to click the added path point to drag a movement path, which is marked with red lines.



Example of Animated Graphic object Not Controlled by controller, One-way, Horizontal Line

On the **Attributes** tab of the **Animated Graphic** object:

1. Check the **Not Controlled by PLC** option.
2. The graphic state is **Location dependent**.
3. The path is **Horizontal line**.
4. The movement rate is **10** pixels/sec. and the direction is **One-way**.



On the **Path** tab of the **Animated Graphic** object:

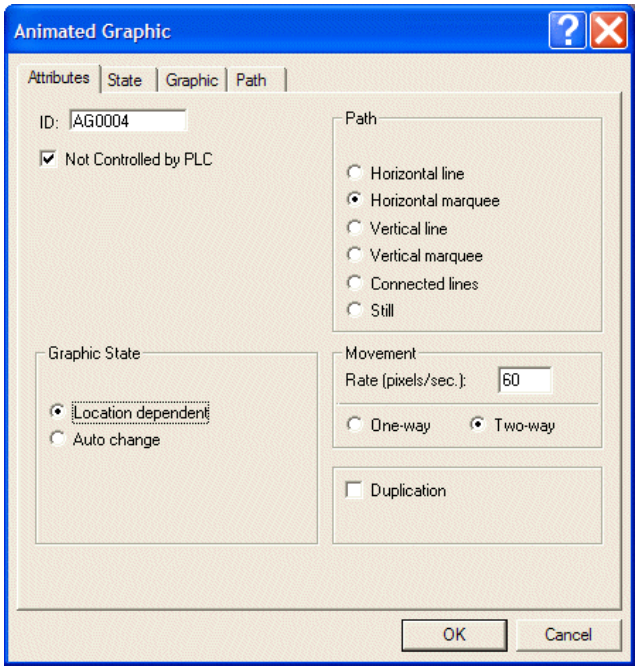
5. This object is one state; and the graphic is **oil.bmp**.

Therefore, this animated graphic is not controlled by the controller and moves along horizontal line at 10 pixels/sec. in one-way mode; the graphics are both **oil.bmp**.

Example of Animated Graphic object Not Controlled by controller, Two-way, Horizontal Marquee

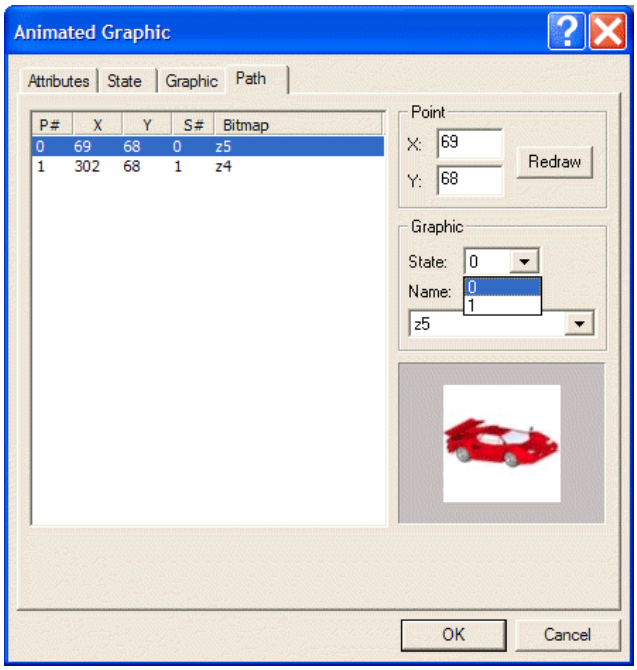
On the **Attributes** tab of the **Animated Graphic** object:

1. Check the **Not Controlled by PLC** option.
2. The graphic state is **Location dependent**.
3. The path is **Horizontal marquee**.
4. The movement rate is **60** pixels/sec. and the direction is **Two-way**.



On the **Path** tab of the **Animated Graphic** object:

5. Since the graphic state is location dependent, two states are set up.



Therefore, this object is not controlled by the controller and moves along a horizontal marquee at 60 pixels/sec. This object will move back and forth when it comes to the end point.

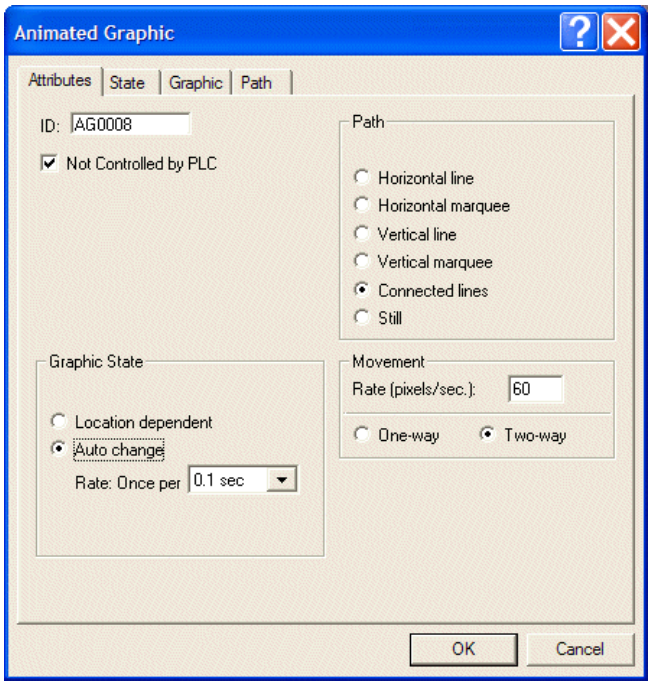
Moving to the right along a horizontal marquee

Moving to the left along a horizontal marquee

Example of Animated Graphic object Not Controlled by controller, Auto Change, Connected Line

On the **Attributes** tab of the **Animated Graphic** object:

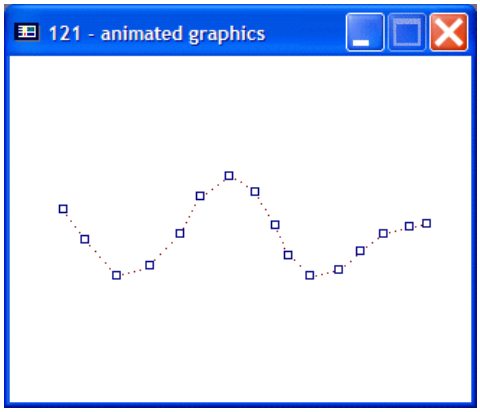
1. Check the **Not Controlled by PLC** option.
2. The graphic state is **Auto change** and the **Rate** is **0.1** sec.
3. The path is **Connected lines**.
4. The movement rate is **60** pixels/sec.; the direction is **Two-way**.



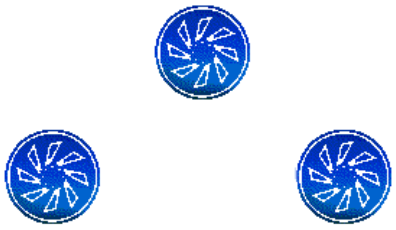
On the **Graphic** tab of the **Animated Graphic** object:

5. Setup 8 graphics for auto change. The example uses 8 wheels with different angles to create a rolling image when the graphic state is changed automatically.

6. Add more points to make the movement path smoother.



Therefore, this animated graphic object is not controlled by the controller and moves at 60 pixels/sec. back and forth along the curve. The graphic state is set to auto change to show a rolling effect.



Moving along the curve with auto change in two-way

GIF Graphic

The GIG Graphic displays GIF graphics, controlled by the controller or not.

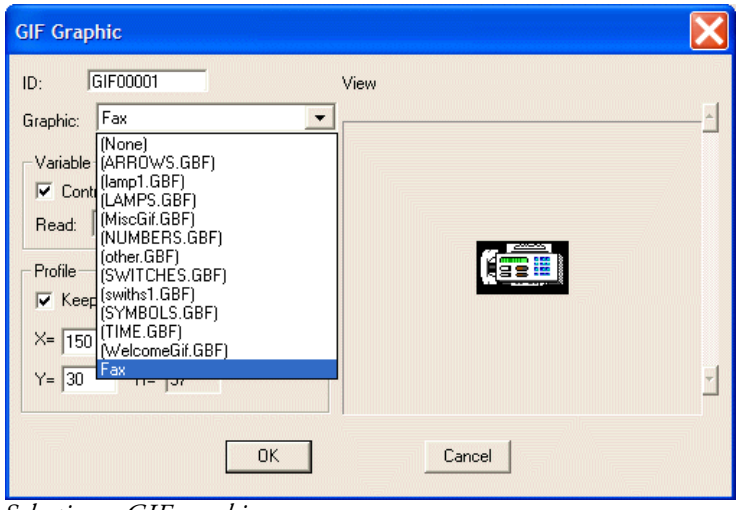
Graphic: Selects the graphic to display from the drop-down list. The graphic will appear in the **View** window.

Profile: Modifies the location and size of objects.

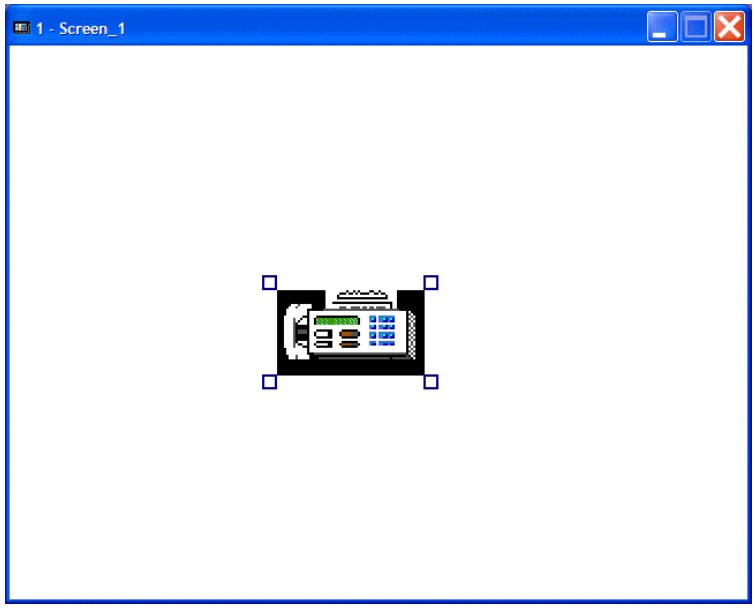
For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of designing a GIF Graphic object

1. Select a GIF graphic form the **Graphic** drop-down list; specify the controller register from which to read (if the option **Controlled by PLC** is checked) and modify its profile.



Selecting a GIF graphic



The GIF graphic appears on the object

State Graphic

The **State Graphic** constantly displays one of several bitmaps depending on the state of the controller register.

Attributes

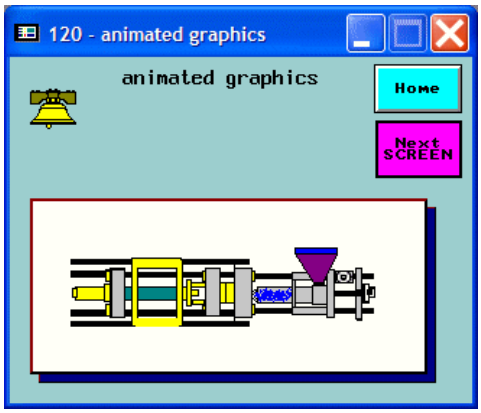
States:

– **Auto Change**: Check this option to change the graphic automatically.

Change Rate (Hz): Specifies the rate of change.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of creating a State Graphic object

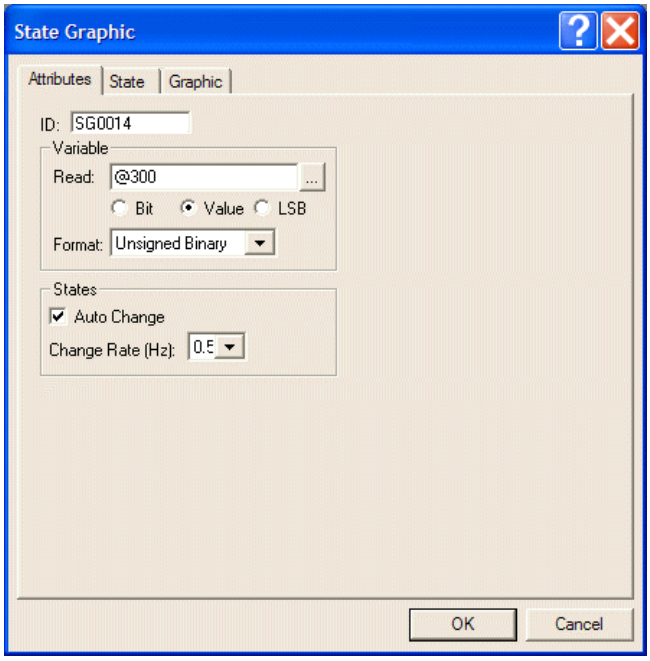


*A **State Graphic** object*

On the **Attributes** tab of the **State Graphic** object:

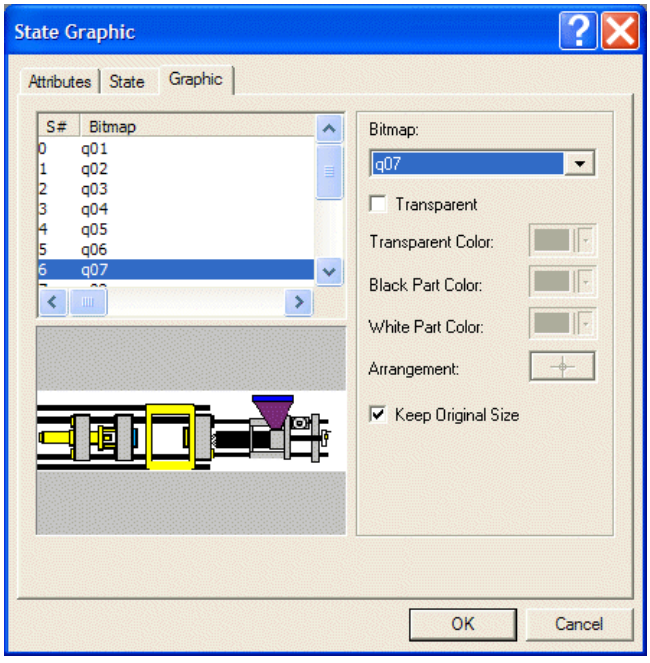
1. **Variable**: Specify **@300** (Local Internal Memory) for **Read** and **Value** to be displayed.
2. Select **Unsigned Binary** for **Format**.

3. Check the **Auto Change** option; **Change Rate** (Hz) is **0.5**.

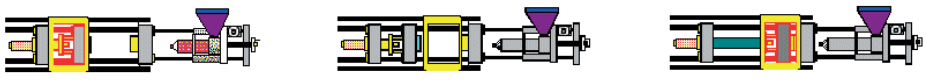


On the **State** and **Graphic** tabs of the **State Graphic** object:

4. There are 14 states altogether. Add states on the **State** tab and select specific graphic to display on the **Graphic** tab.



Therefore, the object changes its state repeatedly every 0.5 sec and is controlled by the controller. A pumping effect will be generated on the operator terminal screen.

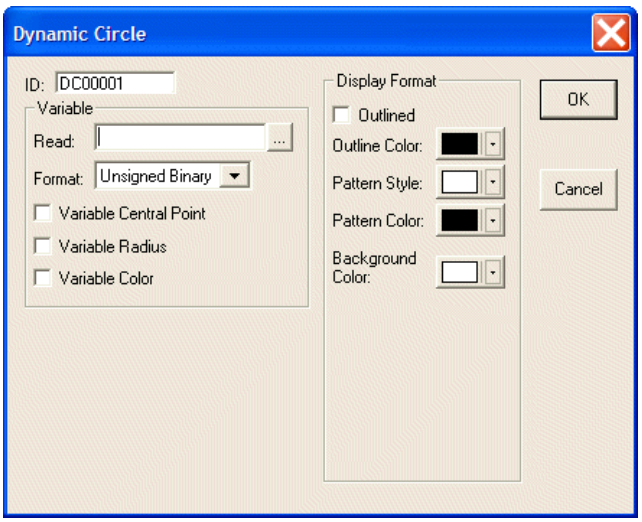


State 0 State 1 State 2

The object shows auto change (ex. state 0-2)

Dynamic Circles

The **Dynamic Circles** object changes its position, radius and color according to controller registers.



Variable Central Point: The position of the central point is controlled by the controller.

Variable Radius: The length of the radius is controlled by the controller.

Variable Color: The color of the object is controlled by the controller.

Display Format: Specifies the format of the object to display.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of using Dynamic Circle

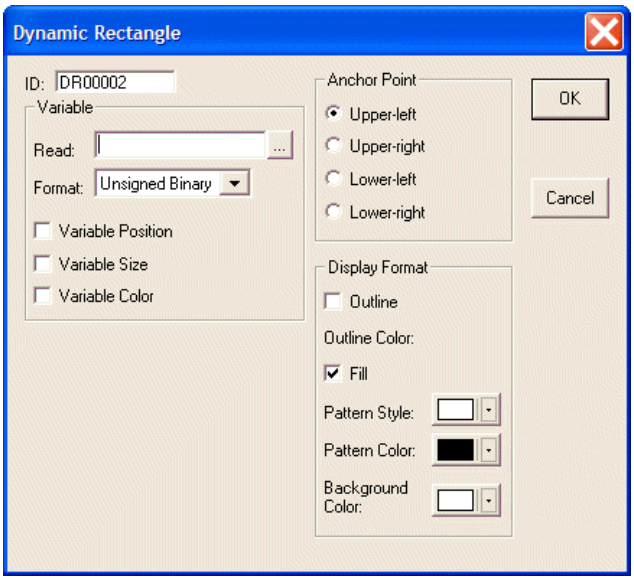
Suppose that the dynamic circle's central point, radius, and color are controlled by the controller. The **Read** address is **W430**.

The operator terminal can read four pieces of data simultaneously at most, and the read addresses here are W430, W431, W432, and W433. The following is the table for controller addresses and graphic properties.

Re-central Point Re-radius Fix Color Wn= Radius Wn+1= X Wn+2= Y Wn+3= Color	Re-central Point Re-radius Fix Color Wn= Radius Wn+1= X Wn+2= Y	Re-central Point Re-radius Re-Coloring Wn= Radius Wn+1= Color	Re-central Point Re-radius Fix Color Wn= Radius	Fix Central Point Re-radius Fix Color Wn= X Wn+1= Y Wn+2= Color	Re-central Point Fix Radius Fix Color Wn= X Wn+1= X	Fix Central Point Fix Radius Re-Coloring Wn= Color

Dynamic Rectangle

The **Dynamic Rectangle** object changes its position, radius, and color according to controller registers.



Variable Position: The position of the object is controlled by the controller.

Variable Size: The length of the object is controlled by the controller.

Variable Color: The color of the object is controlled by the controller.

Anchor Point: Specifies the anchor point for the dynamic rectangle whose position is variable and size is fixed.

Display Format: Specifies the format of the dynamic rectangle to display.

For properties not explained in this section, please see the section *Specifying Object Properties*.

Example of using Dynamic Rectangle

Suppose that the position, size, and color are variable and are controlled by the controller. The **Read** address is **W420**.

The operator terminal reads five simultaneous pieces of data from the controller at most. The read addresses here are W420, W421, W422, W423, and W424. The following is the table of controller addresses and graphic properties.

Re-position Re-size Re-Coloring W420=Width W421=Height W422=X W423=Y W424=Color	Re-position Re-size Fix Color W420=Width W421= Height W422=X W423=Y	Fix Position Re-size Re-Coloring W420= Width W421= Height W422=Color	Fix Color Re-size Re-Coloring W420=Width W421=Height	Fix position Fix Size Re-Coloring W420=X W421=Y W422=Color	Re-position Fix Size Fix Color W420=X W421=Y	Fix Position Fix Size Fix Color W420= Color

2.7.18 Historical Display

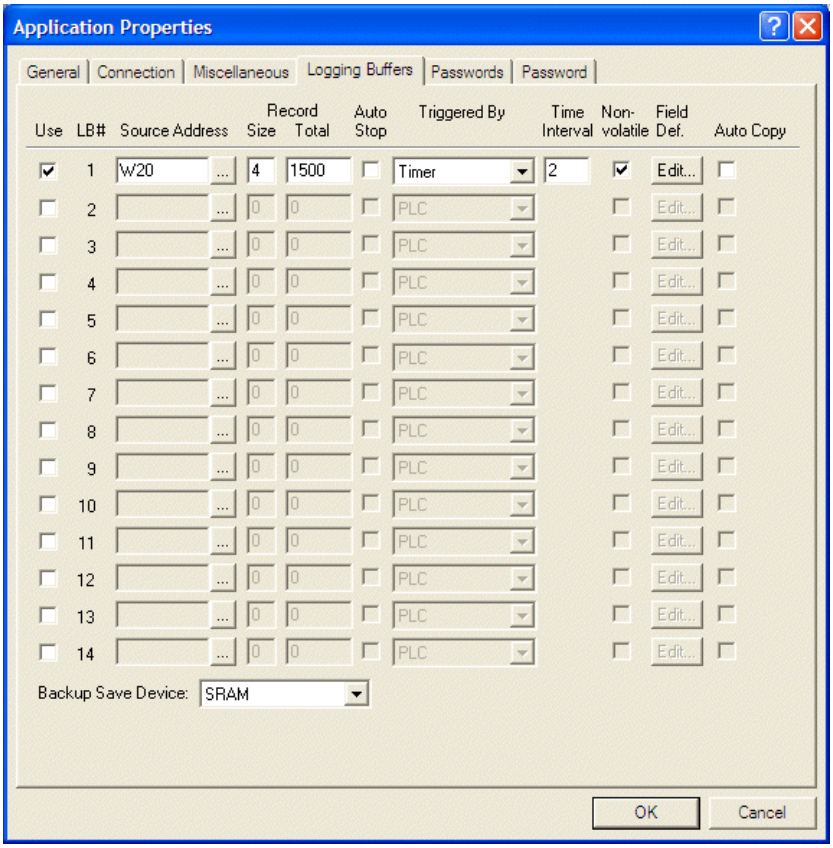
The **Historical Display** drop-down list includes the following: **Historical Trend Graph**, **Historical Data Table** and **Historical Event Table**.

The **Historical Display** data is stored in **logging buffers**, so you should assign its area and size first. The logging buffer is used to store the sampling data in the operator terminal battery backup RAM.

The object is not available on all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

Logging Buffer

Select the **Logging Buffers** tab from **Application/Workstation Setup**.



*The **Logging Buffers** tab*

Source Address: Specifies the starting address to read from, e.g. **W20** is starting address of a block of controller registers from which the logging buffer reads from.

Size: Specifies the size of a record to read from at one time, e.g. **Size = 4** represents 4 words = W20, W21, W22, W23.

Total: Specifies the total to store in, e.g. **1500** represents that the operator terminal reads 4 words each time, sampling 1500 times in total.

Time/Date: Check these boxes to record the **Time/Date** while sampling.

Auto Stop: Check this box to stop the sampling when it reaches the specified total = 1500. If this option is not selected, the first piece of data will be overwritten when the 1501st piece of data records is recorded.

Triggered By: Select **Timer** to trigger periods of fixed time or select controller to be triggered by the controller. If **PLC** is selected for triggering, it is triggered by the specified corresponding bit-locations, Wn+2, Wn+3, and Wn+4.

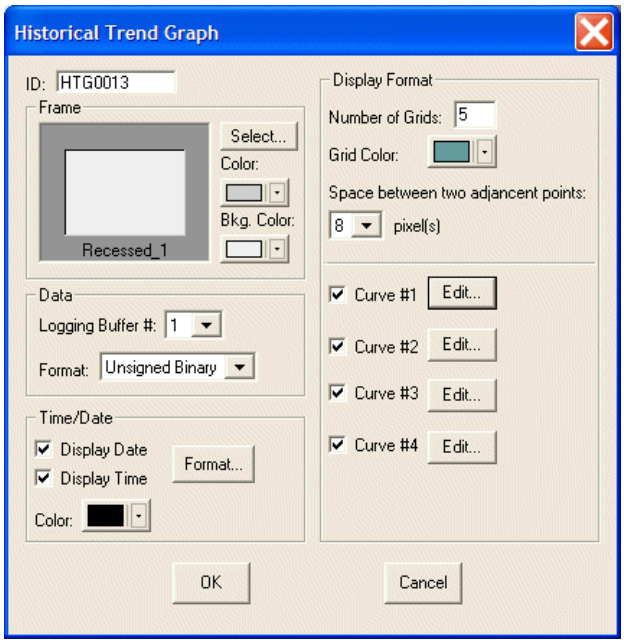
Time Interval: Specify how often the logging buffer gets a record of data from the timer. Unit: seconds.

Auto Copy: If checked, the logging buffer data will be copied and saved in the selected backup save device automatically. There are three **Backup Save Device** options: **SRAM**, **CF Card** and **USB Memory Stick**.

After completing the setup on the **Logging Buffers** tab , you can create three types of **Historical Display** objects:

Historical Trend Graph

The operator terminal may have a fixed sampling period or the controller may initiate that data be read from the specified registers and then store data in the logging buffers in the operator terminal memory. After a sampling period, data is converted to continuous curve(s) and displayed on the operator terminal.



Data

Logging Buffer #: Specifies the number of the logging buffer where the historical data is stored, numbered 1 to 12.

Format: **BCD**, **Signed Binary** or **Unsigned Binary**.

Time/Date

Display Date and **Display Time**: Checking the boxes displays date and time; click **Format** button to setup.

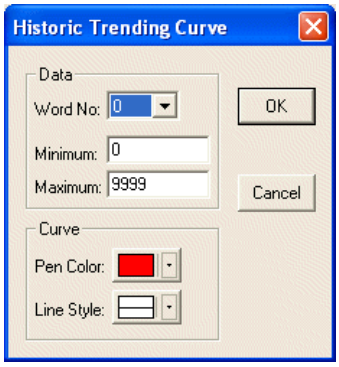
Color: Specifies the color of characters to be displayed.

Display Format

Number of Grids: Specifies how many evenly spaced horizontal lines shall be displayed.

Grid Color: Specifies the color of the horizontal grids.

Curve #1 - #4: There are four curves to be selected. Click the **Edit** button to display the following dialog box:



Word No: Specifies the number of the words to display on the historic trend curve.

Minimum and **Maximum**: Specifies the value corresponding to the lowest and highest point on the historical trend curve.

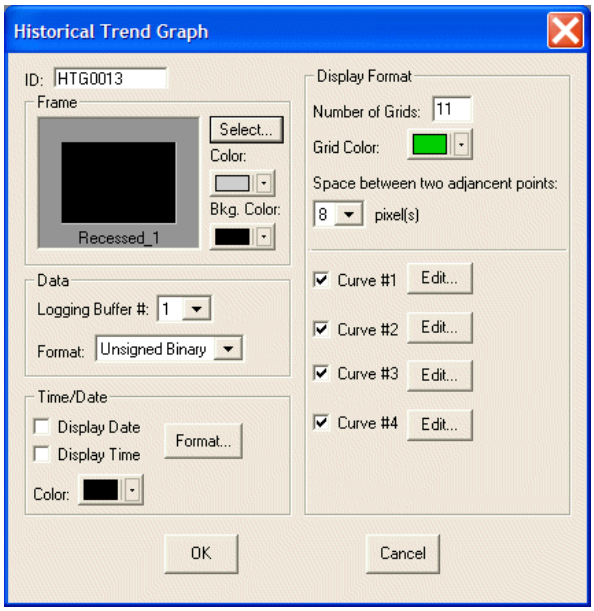
Pen Color: Specifies the color used to draw the trend curve.

Line Style: Specifies the line style of the trend curve.

Example of designing a Historical Trend Graph

1. First assign logging buffer size and area for **Logging Buffer #1** on the **Logging Buffers** tab in **Application/Workstation Setup**:
   Source Address: @20
   Size: 4
   Total: 3000
   Check **Time** and **Date**
   Triggered By: Time
   Time Interval: 1
   Check **Non-volatile**.
   Please see section *Logging Buffer*.

For the **Historical Trend Graph** object, select the following properties:

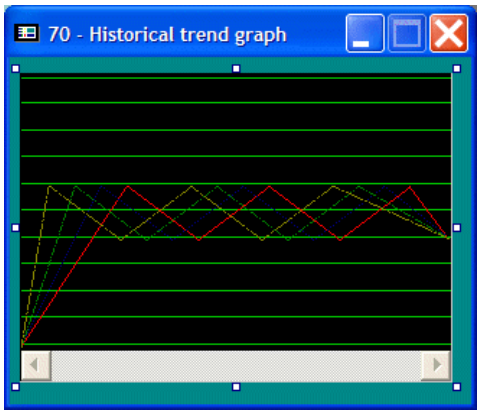


*The **Historical Trend Graph** properties*

2. **Frame**: Select **Recessed_1** and **Black** for background color.
3. **Data**: Select **Logging Buffer #1** and **Unsigned Binary** for **Format**.
4. **Display Format**: Specify **11** grids and **Green** for **Grid Color**.
5. Check four curves to display; Curve #1 displays the data record stored in the word 0, Curve #2 displays the data record stored in the word 1 etc.

*Properties for **Historical Trending Curve #1***

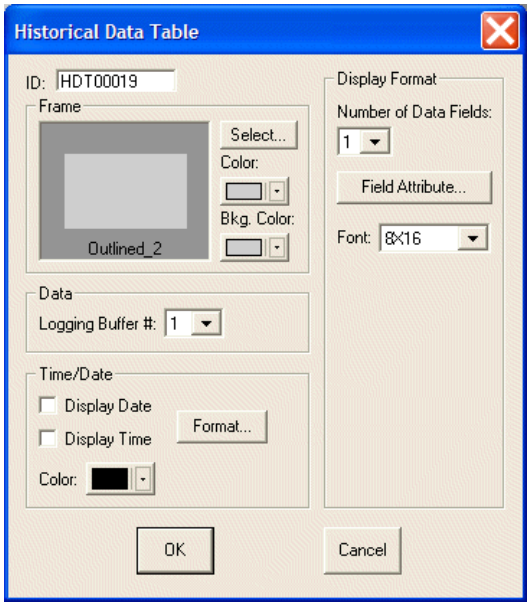
6. Set **Minimum** to **0** and **Maximum** to **65535**. Specify a different color for each curve.



The result of the settings in the above example: Displaying data stored in logging buffer #1.

Historical Data Table

The operator terminal may have a fixed sampling period or the controller may initiate a data read from the specified registers and then store data in the logging buffers in the operator terminal memory. After a sampling period, data is then converted to numeric data tables and displayed in the operator terminal.

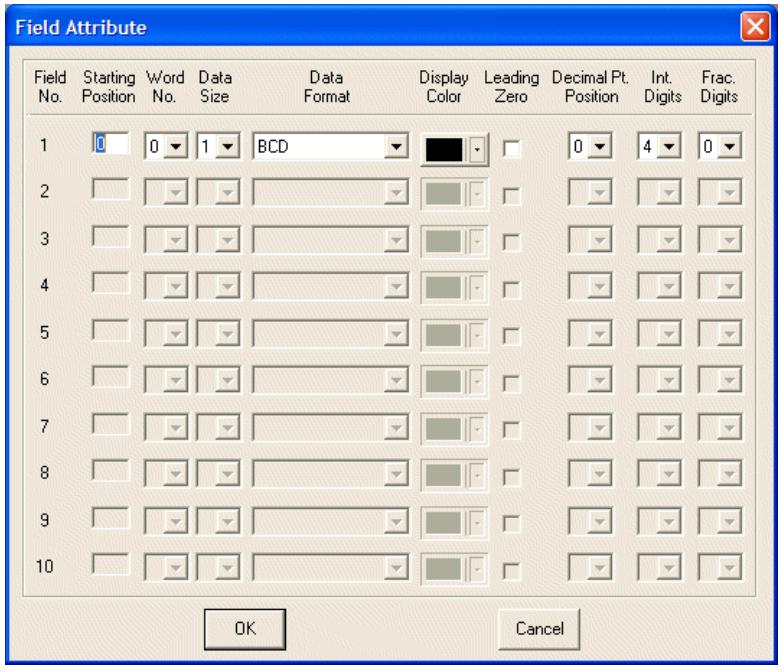


Historical Data Table properties

Display Format

Number of Data Field: Specify how many data fields to display; up to 10 data fields.

Clicking the **Field Attribute** button displays the dialog box below:



Starting Position: Specifies the position of a data field to display.

Note:
If the starting position is **0** for **Field No. 1**; the time will be displayed in **Field No. 1**, the date will be displayed in **Field No. 2**, and the first data field will be displayed in **Field No. 3**. If there is no time/date displayed, the first data field will be displayed in **Field No. 1**.

Word No.: 0-31 characters can be specified.

Data Size: 1 represents one-word; 2 represents double-word.

Data Format: Select **BCD**, **Signed Binary**, **Unsigned Binary** or **Hexadecimal**.

Display Color: Specifies the character color of a data field.

Leading Zeros: Checking this box to displays leading zeros.

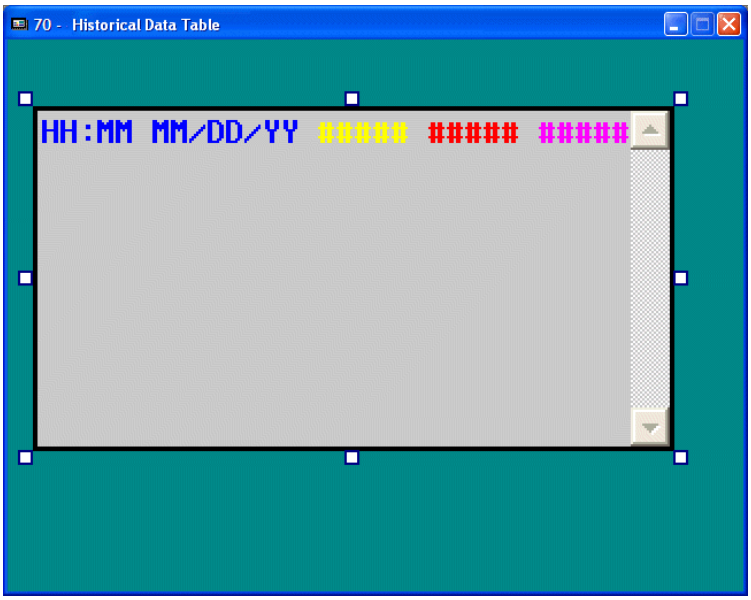
Decimal Pt. Position: Specifies the number of digits following the decimal point.

Int. Digits: Specifies how many digits to the left of the decimal point.

Frac. Digits: Specifies how many digits to the right of the decimal point.

For properties not explained in this section, please see sections *Historical Trend Graph* and *Specifying Object Properties*.

Example of designing a Historical Data Table



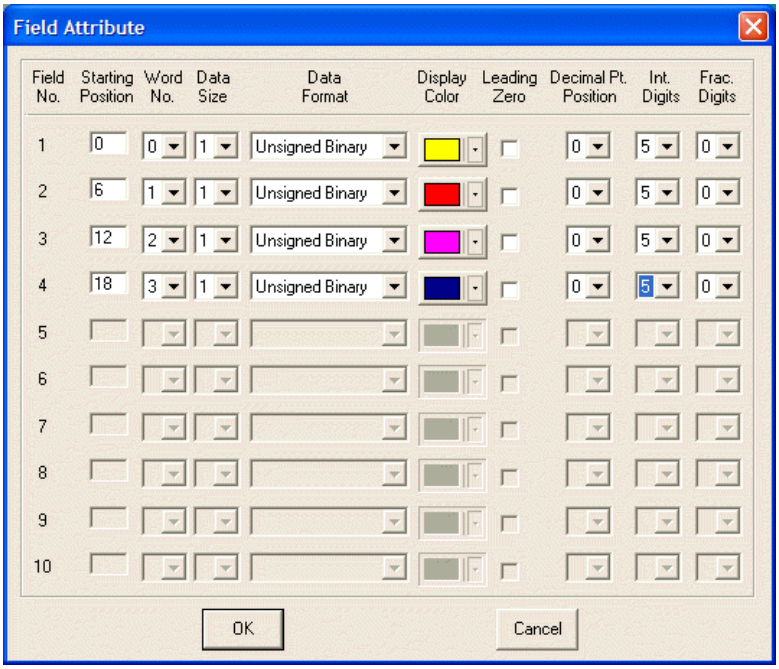
A Historical Data Table

Suppose that there is historical data stored in logging buffer #1.

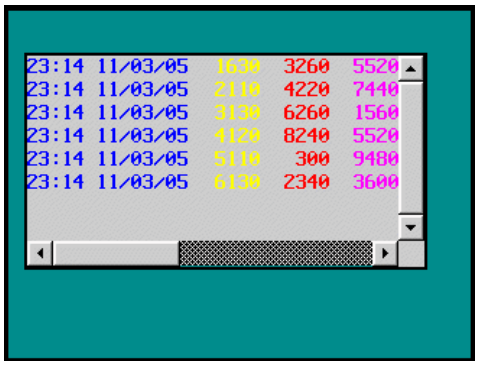
Select the following properties for the **Historical Data Table** object:

1. **Frame**: Select **Outlined_2** and **Black** for border color.
2. **Data**: Select **Logging Buffer #1** to read data from.
3. **Time/Date**: Check both boxes.

4. **Display Format**: Select 4 data fields. Click **Field Attribute**.



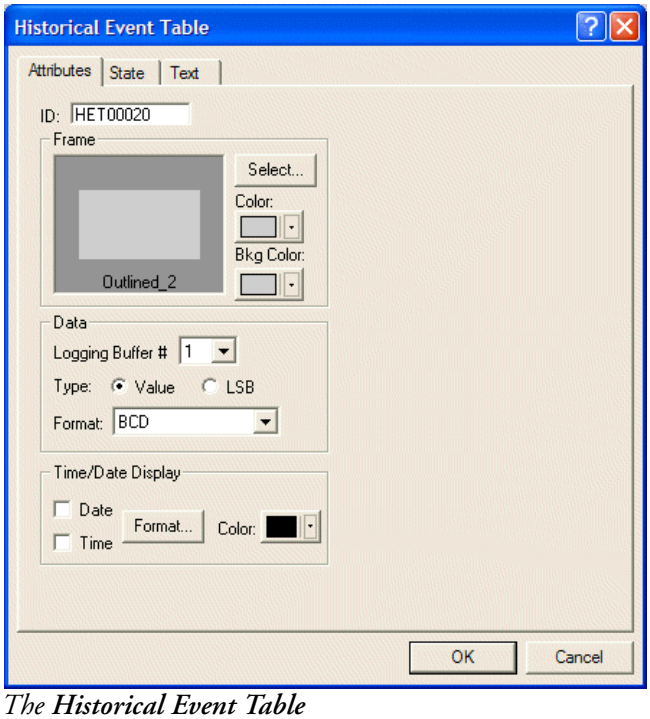
5. Make settings according to above.



The result of the settings in the above example: Displaying data stored in logging buffer #1.

Historical Event Table

The operator terminal may set a fixed sampling period or the controller may initiate a data read from the specified registers or the related bits in LSB. Thereafter, the data is converted into pre-defined message text(s) and then displayed line-by-line in the operator terminal.



*The **Historical Event Table***

Data

Logging Buffer #: Specifies the number of the logging buffer where the historical data is stored, numbered 1 to 12.

Type: Select **Value**; 256 states in all (0-255), where 0 represents state 0; 1 represents state 1 etc. or **LSB**; 16 states in all, the operator terminal takes the bit number of the lowest bit that is ON as the state number.

Format: Only available when **Value** is selected, and the formats are **BCD**, **Unsigned Binary** or **Signed Binary**.

For properties not explained in this section, please see sections *Historical Trend Graph* and *Specifying Object Properties*.

2.7.19 Alarm Display

There are four types of **Alarm Display**: **Alarm History Table**, **Active Alarm List**, **Alarm Frequency Table** and **Alarm Marquee**.

To use the **Alarm Display** objects, you must set up the address of the alarm block and its parameters. The operator terminal reads the value stored in the controller and displays its corresponding messages. Up to 512 messages can be set. Please see section *Alarm Setup* for information about setting up alarms.

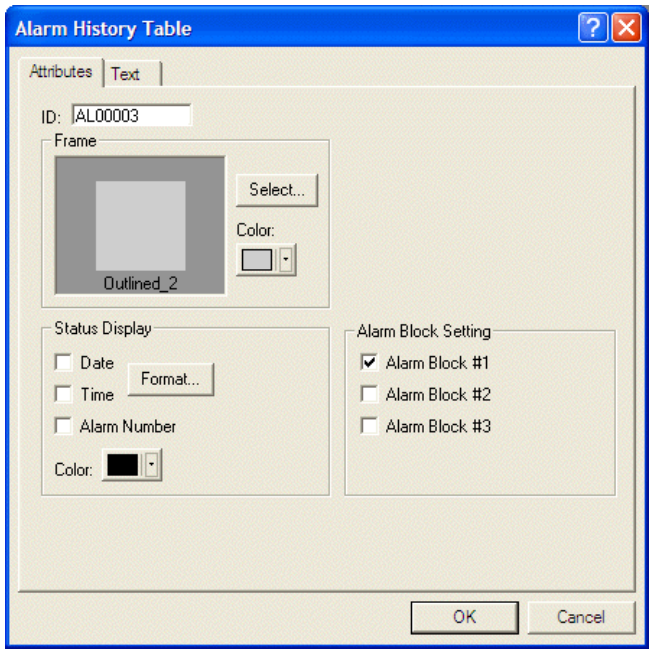
The objects are not available on all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

After completing alarm setup, the four types of **Alarm Display** objects can be used.

Alarm History Table

The operator terminal reads the reference bits in the controller in fixed periods and then activates the corresponding alarm messages. Thereafter, this the operator terminal can show the alarm history as an **Alarm History Table**.

Attributes Tab



Status Display

– **Date** and **Time**: Check the boxes to display the date and time and click the **Format** button to specify formats.

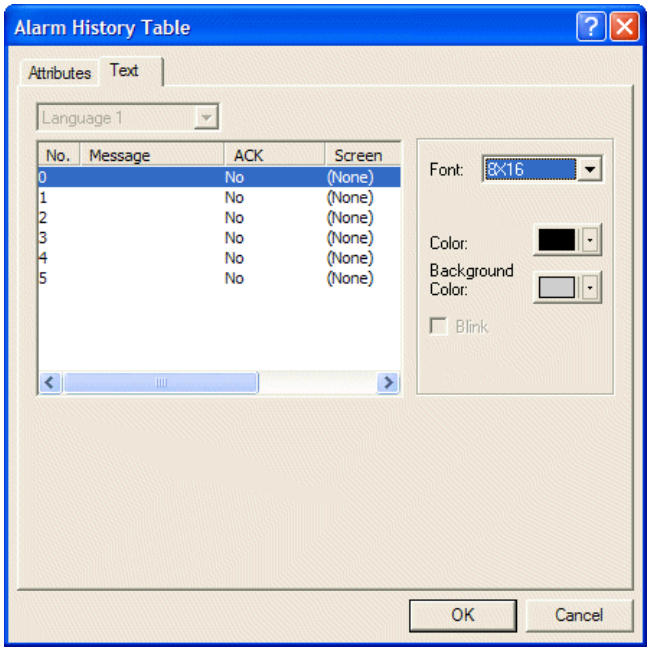
– **Alarm Number**: Check this box to display the alarm number.

– **Color**: Specifies the color of the message.

Alarm Block Setting

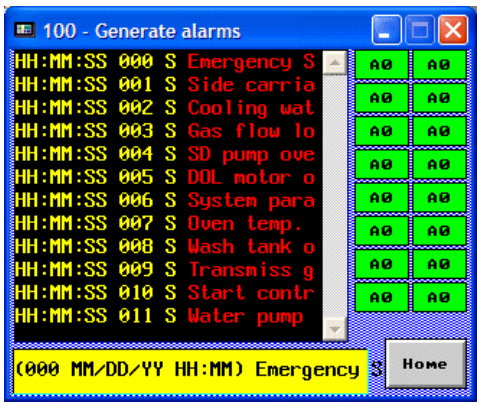
– Select which alarm blocks to collect alarms from. See section *Alarm Block Setting* for information about alarm blocks.

Text Tab



On this tab, **Font**, **Color** and **Background Color** for alarm messages can be selected.

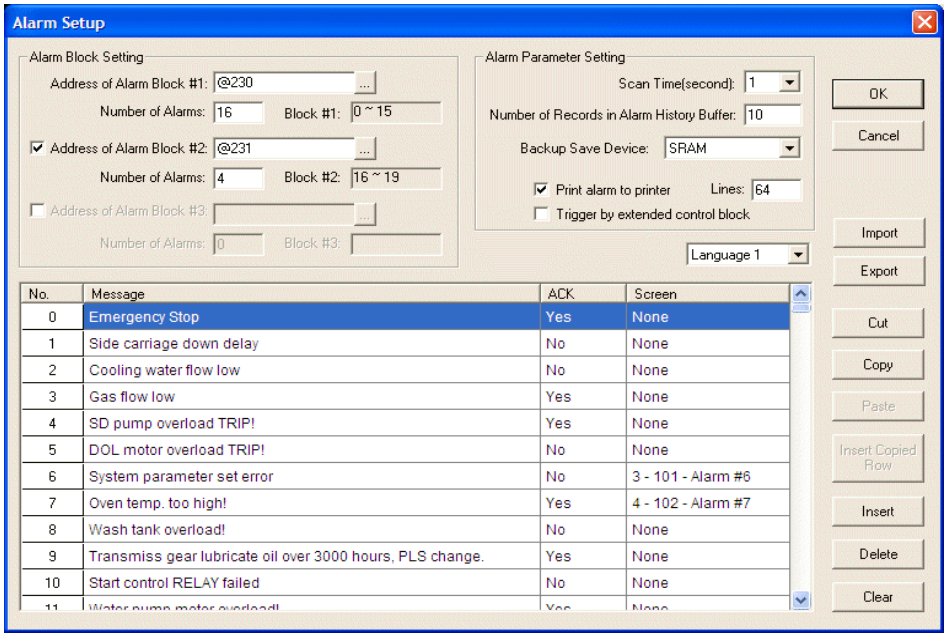
Example of designing an Alarm History Table



Alarm Setup:

1. Select **Application/Alarm Setup.**
2. Specify **@230** for **Address of Alarm Block** to and **16** as **Number of Alarms.**
3. Set the scan time to **1** second for sampling the controller data, and maximum number of records to **100**.

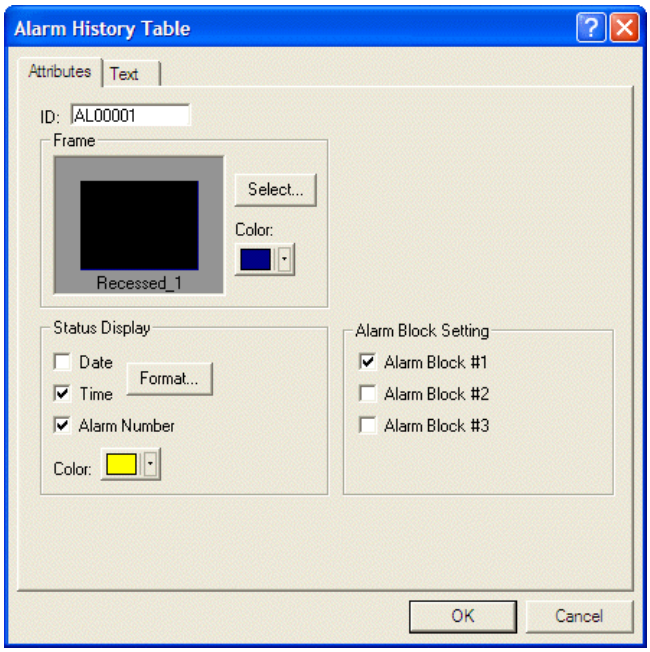
4. Enter texts in the message block, select to acknowledge the alarm and which screen to display.



*The **Alarm Setup** dialog box*

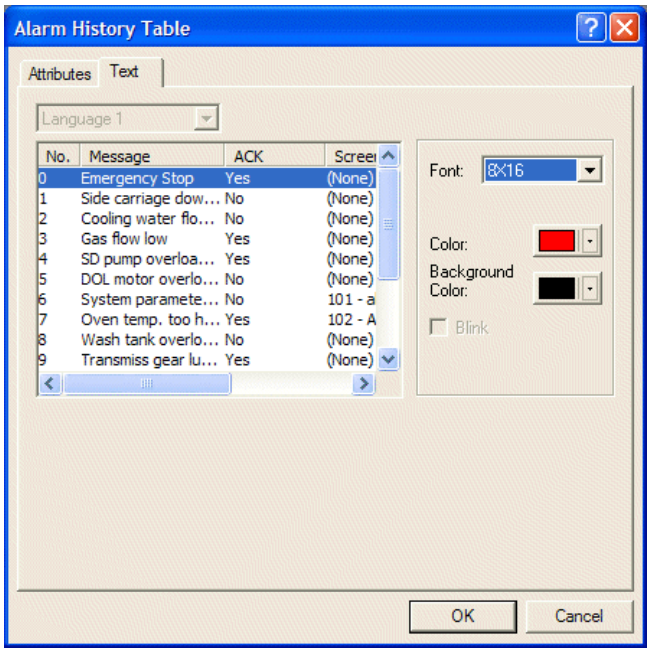
Setting up properties for the Alarm History Table:

5. **Frame**: Select **Recessed_1** and **Blue** for frame color.
6. **Status Display**: Check **Time** and **Alarm Number**, and select **Yellow** for color.



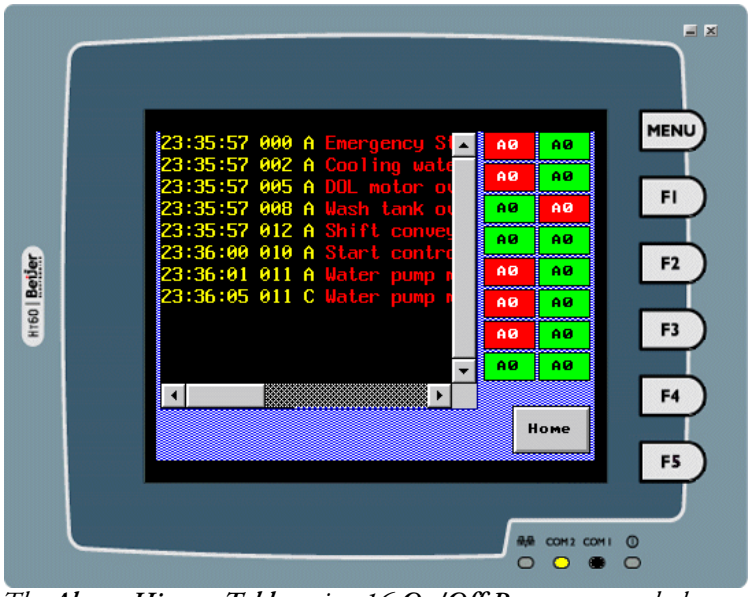
*The **Attributes** tab of the **Alarm History Table***

7. Note that the properties for **Message**, **ACK** and **Screen** set in the **Alarm Setup** dialog box will be shown on the **Text** tab. Select **Red** for message color and **Black** for background color.



*The **Text** tab of the **Alarm History Table***

The operator terminal will read the reference bits in the controller at fixed periods, then convert the data into its corresponding messages in sequence and display them on the screen.



*The **Alarm History Table**, using 16 **On/Off Buttons** to send alarm messages.*

Note that the **Alarm State A** represents **Activate**; **Alarm State C** represents **Clear**.

Active Alarm List

The operator terminal displays only the active alarms according to its reference bit in controller = ON and sorts the data according to the order of the state number.

Most attributes are the same as for the **Alarm History Table** object, please see the section *Alarm History Table*. **Alarm Number** and **Sort** are specific for the Active Alarm List:

Alarm Number

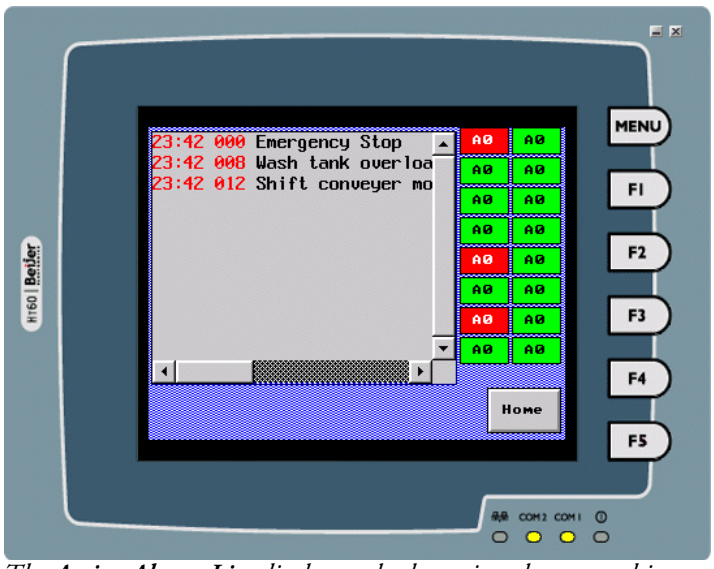
Checking the **Alarm Number** check box will display the alarm number for any active alarms (not shown by default).

Sort

Checking the **Sort** check box will sort the alarm list in chronological order; otherwise sorting will be done by alarm number.

Example of designing an Active Alarm List

The steps used to create an **Active Alarm List** are the same as for an **Alarm History Table** object. You must complete the alarm setup first, and then specify its properties. Please see section *Example of designing an Alarm History Table*.



*The **Active Alarm List** displays only the active alarms, and in numerical order.*

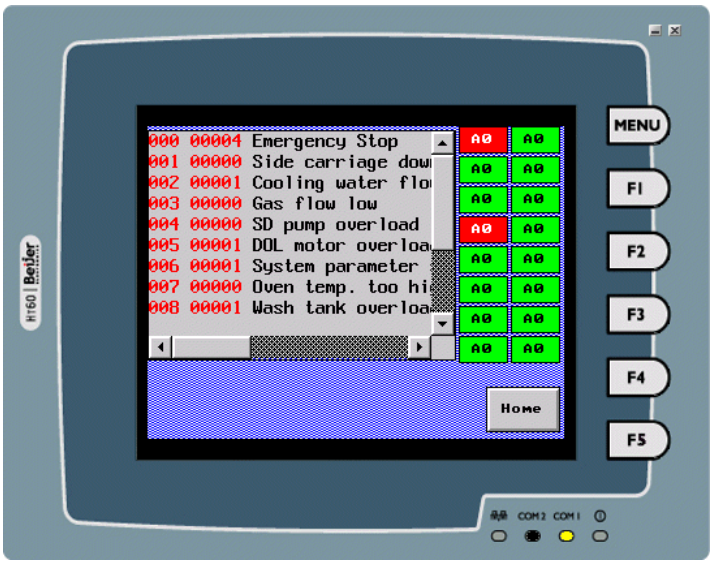
Alarm Frequency Table

The operator terminal summarizes the number of occurrences of each alarm which are to be monitored and displayed on the screen.

All the attributes are the same as for the **Alarm History Table** object, please see the section *Alarm History Table*.

Example of designing an Alarm Frequency Table

The steps used to create an **Alarm Frequency Table** are the same as for an **Alarm History Table** object. You must complete the alarm setup first, and then specify its properties. Please see section *Example of designing an Alarm History Table*.



*The **Alarm Frequency Table** displays the number of occurrences of each alarm*

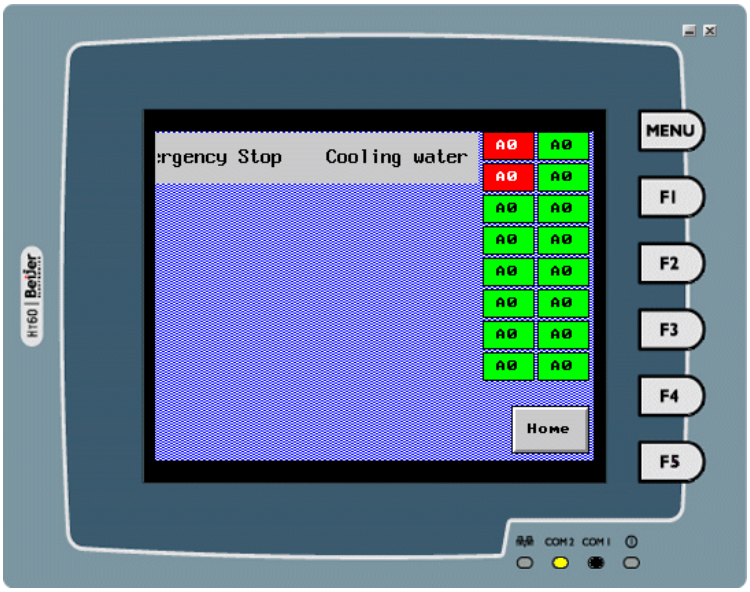
Alarm Marquee

The operator terminal displays alarm messages from active alarms as a moving sign.

All the attributes are the same as for the **Alarm History Table** object, please see section *Alarm History Table*.

Example of designing an Alarm Marquee

The steps used to create an **Alarm Marquee** are the same as for an **Alarm History Table** object. You must complete the alarm setup first, and then specify its properties. Please see section *Example of designing an Alarm History Table*.

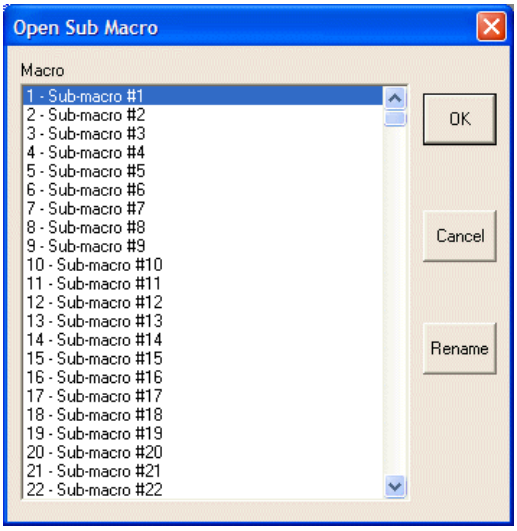


*The **Alarm Marquee** displays the active alarm message as a moving sign on the screen*

2.7.20 Sub Macro

A **Sub Macro** is the macro's sub-application. The main function is to call commands directly. Some common functions or operation commands which are used frequently can be edited and saved as sub macros for call commands.

There are 512 options for a Sub Macro, please see chapter *Macros* for complete details.



*The **Sub Macro** edit window*

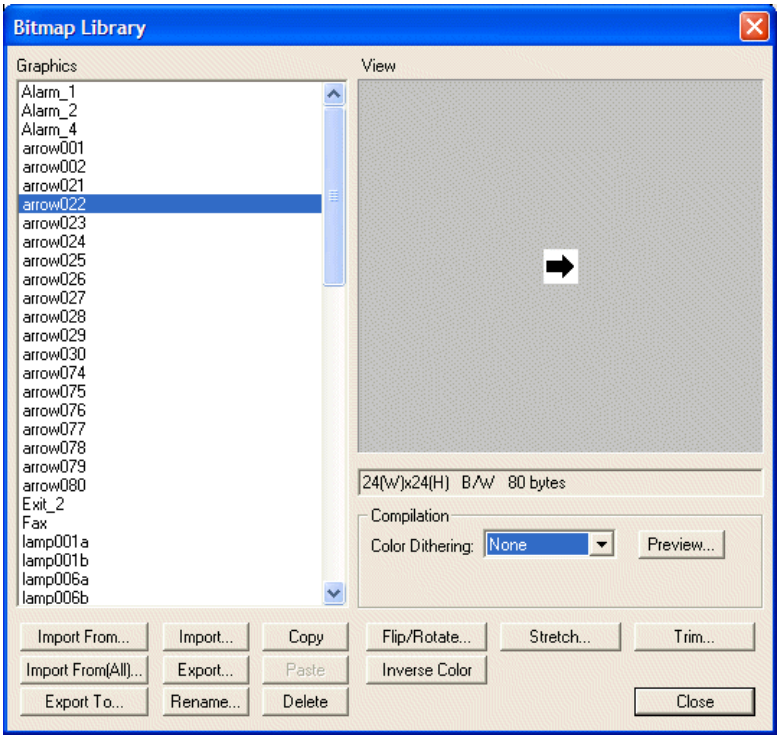
2.8 Library Menu

The following options are available from the **Library** menu: **Bitmap Library**, **Font Library**, **Save as Shape**, **Shape Library Manager**, **Message Library**, **Index Registers**, **Node Registers**, **Time Channels** and **Text Pool**. The main function is to edit, import and export bitmaps, shapes, fonts or text pools.

2.8.1 Bitmap Library

The **Bitmap Library** is used mainly to import, export, and edit bitmaps.

Select **Library/Bitmap Library**, to open the dialog box below:



*The **Bitmap Library** dialog box*

Graphics: Lists all the graphics available for selection.

View: Displays the selected graphic.

Compilation

Color Dithering: Processes the graphic (16-bit, 24-bit or JPEG) to display the image as vividly as the original on the screen. There are **8-color**, **16-color** and **256-color** options. The higher the color selected, the higher the contrast of the figure displayed.

Import From: Imports bitmaps one-by-one from a selected library, *.GBF or *.GIF graphics.

Import All: Imports all bitmaps from a selected library, without possibility to confirm each file that is to be imported, *.GBF or *.GIF graphics.

Export To: Exports bitmaps to a selected library, *.GBF.

Import: Imports the graphic into the bitmap library from a computer. The importable graphic formats include Bitmap Image (*.BMP), Jpeg Image Files (*.JPG), AutoCad Files (*.DWG or *.DXF) and GIF Files (*.GIF).

Export: Exports the graphic stored in the bitmap library to a computer.

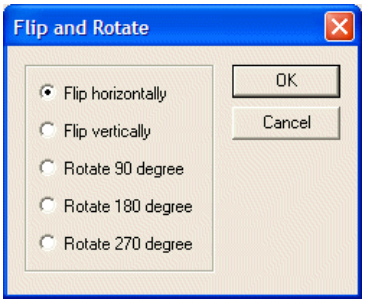
Rename: Modifies the name of the graphic.

Copy: Copies the selected bitmap to the clipboard.

Paste: Imports a bitmap from the clipboard. When clicked, a dialog box will appear and ask for the name of the imported graphic.

Delete: Deletes the selected bitmap.

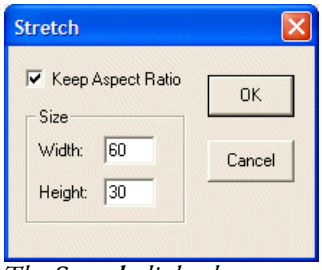
Flip/Rotate: Allows changing a bitmap's orientation. When clicked, a dialog box with flip or rotate degree options is displayed.



*The **Flip and Rotate** dialog box*

Inverse Color: Inverts the selected bitmap's colors.

Stretch: Adjusts the width and height of the bitmap.



*The **Stretch** dialog box*

Trim: Allows cutting unused area around a bitmap.

*The **Trim** dialog box*

2.8.2 Font Library

The **Font Library** supports all Windows fonts, and enables defining up to 16 types. You can define your preferred fonts to create a more attractive interface.

Times New Roman

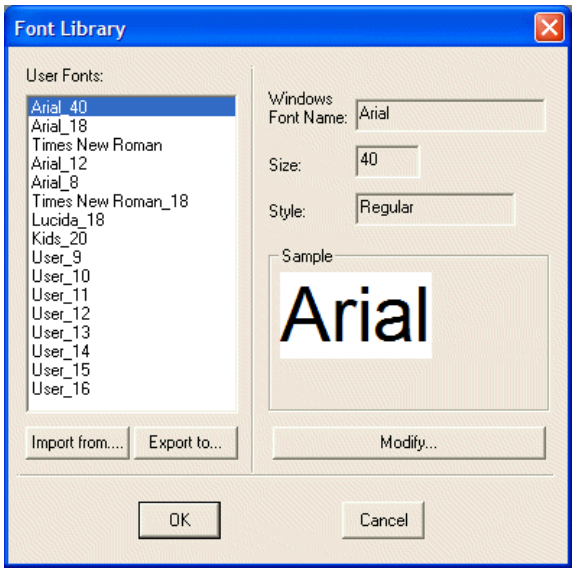
Arial

Kids Comic Sans MS

Lucida Handwriting

User-defined fonts

Select **Library/Font Library** to display the **Font Library** dialog box.

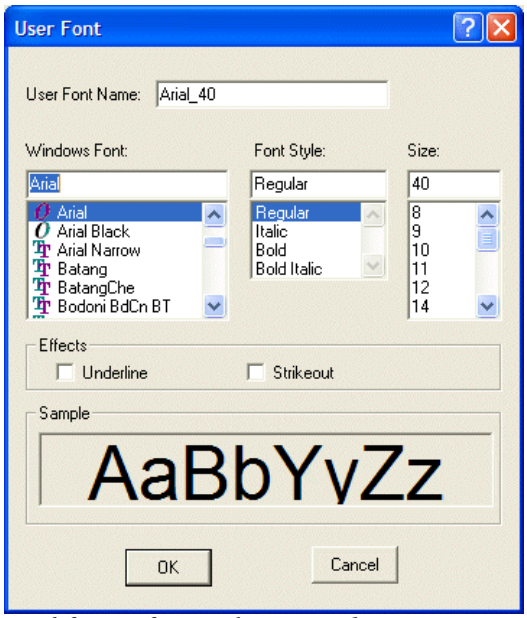


*The **Font Library** dialog box*

Import from: Click to import fonts into the font library.

Export to: Click to export fonts to the font library.

Modify: Click to modify the format of selected font.

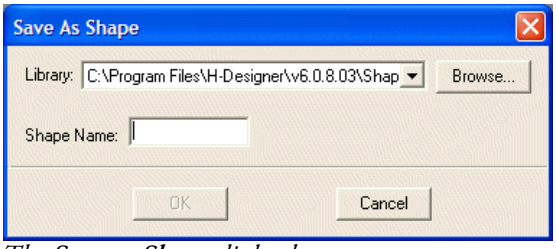


*Modifying a font in the **Font Library***

2.8.3 Save as Shape

The **Save as Shape** command allows you to save basic objects such as **Line**, **Rectangle**, **Circle**, **Polygon**, **Pie**, **Arc**, **Scale** or multiple shapes as a file in the **Library**. A shape must be selected before saving it as ashape. Multiple shapes may be selected simultaneously.

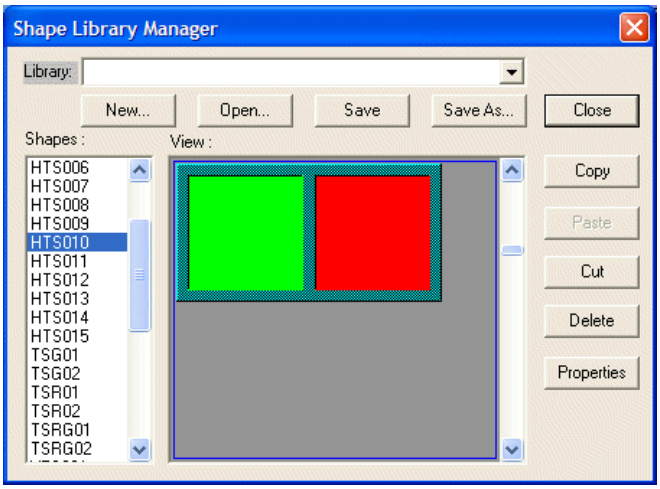
Select **Library/Save as Shape**. You can select the library to save in from the drop-down list and name the shape in the **Shape Name** block.



*The **Save as Shape** dialog box*

2.8.4 Shape Library Manager

The **Shape Library Manager** is used to manage shapes in the **Shape Library**.



*The **Shape Library Manager** dialog box*

New: Creates a new shape library.

Open: Opens an existing shape library.

Save: Saves the active shape library to a file.

Save As: Allows selection of which file to save to.

Close: Closes the **Shape Library Manager**. If changes have not been saved, a dialog box will be displayed, asking if you want to save.

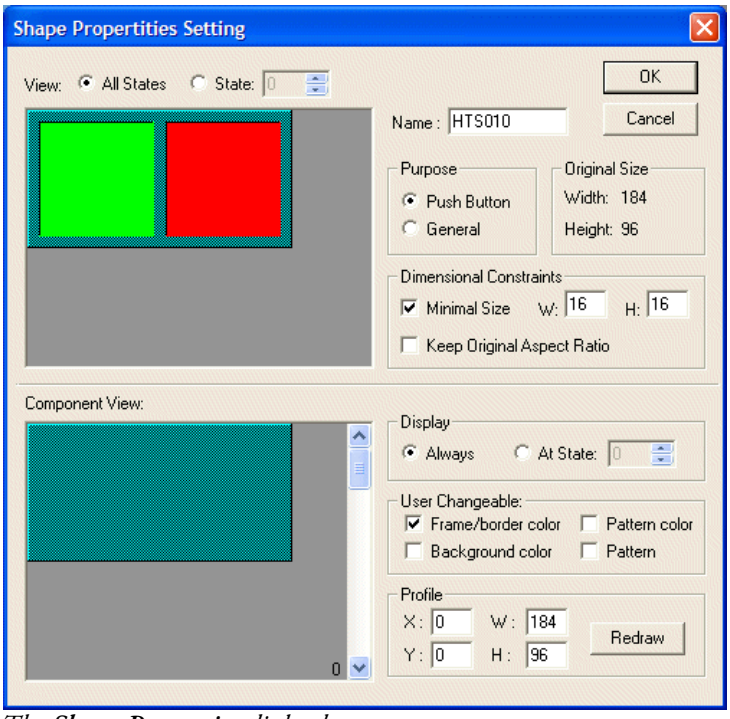
Copy: Copies the selected shape to the clipboard.

Paste: Imports the shape from the clipboard.

Cut: Exports the selected shape to the clipboard.

Delete: Deletes the selected shape.

Properties: Displays the properties of the selected shape.



*The **Shape Properties** dialog box*

– **View**:
All States: Displays all states of the selected shape.
State: Displays individual states of the selected shape.

– **Name**: Specifies the name of the selected shape.

– **Purpose**: The function of the selected shape; **Push Button** or **General**.

– **Original Size**: Displays the width and height of the original shape.

– **Minimum Size**: Specifies the minimum width and height of the selected shape.

– **Keep Original Aspect Ratio**: Selecting this option keeps the size of the shape in its original ratio.

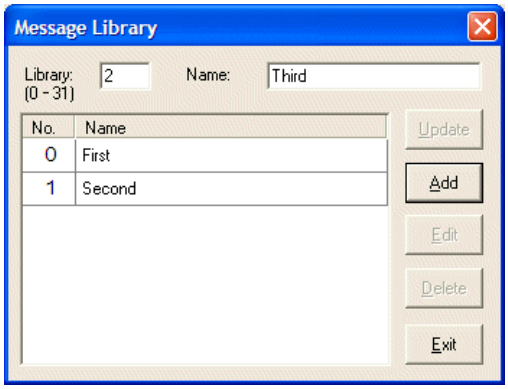
– **Display**: Sets the state of the selected graphic; **Always** or **At State**.

– **User Changeable**: Allows changing shape properties, including frame/border color, background color, pattern color, and pattern.

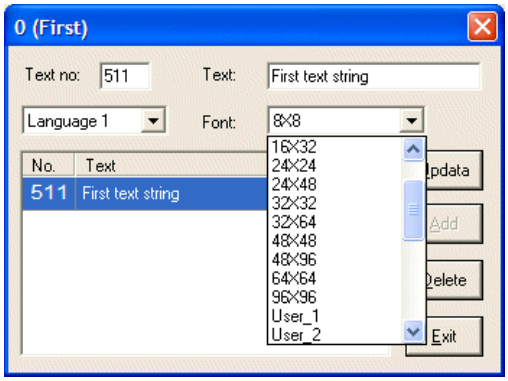
– **Profile**: Specifies the position of the component object and displays the specified view here.

2.8.5 Message Library

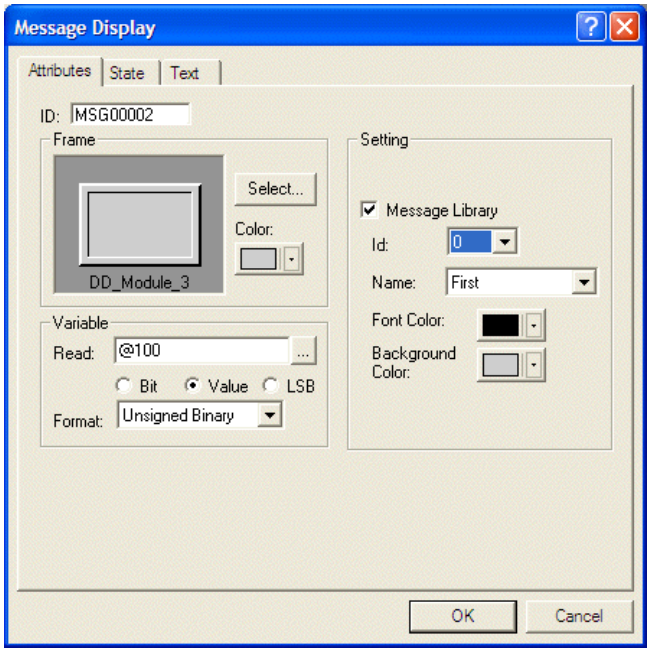
The **Message Library** allows you to define pre-stored message text. A maximum of 32 user-defined message library can be set up; 0 - 31. Each message library can be named differently.



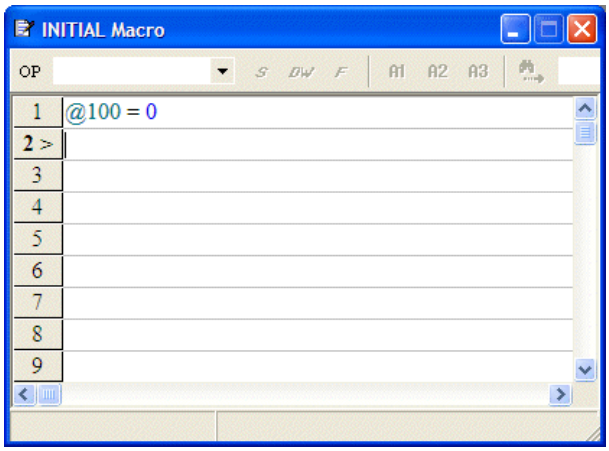
Selecting the message library, and clicking **Edit** makes it possible to edit up to 512 text numbers; 0 - 511. Each text number can contain up to 40 characters with user-defined languages and fonts.



After setting up the message libraries, the predefined messages can be displayed by using the prestored message (available by selecting **Message Display/Prestored Message** from the **Object** menu.



Checking the **Message Library** check box makes it possible to select the defined messages. The prestored message uses a controller register or a internal variable @ to read the value and display text number accordingly, e.g. if the text number is 0, you can define a controller register or @100 as 0 by a numerical entry or macro command.



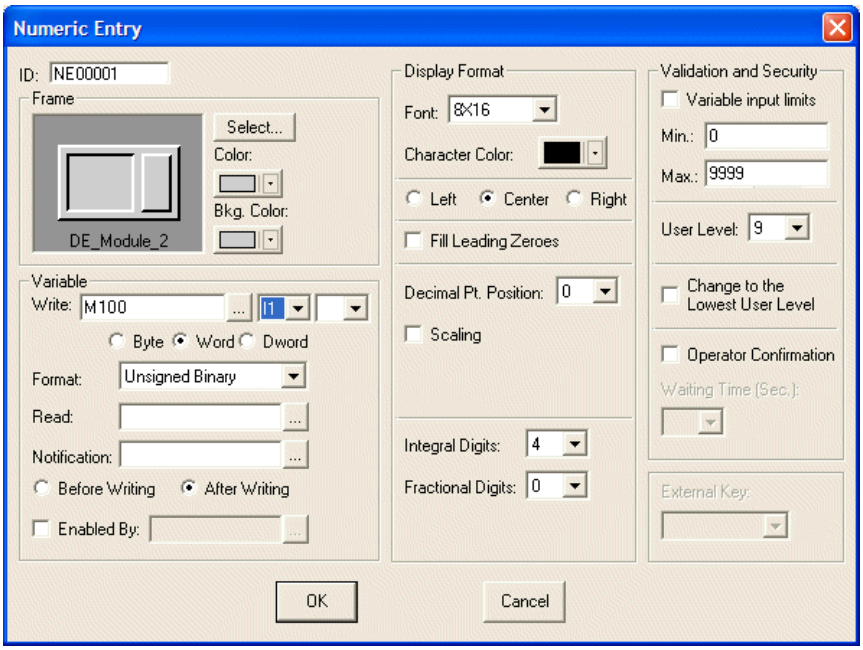
2.8.6 Index Registers

The **Index Registers** function allows you to define up to 8 controller registers, from I1 - I8, in order to add or subtract the value in the defined register to the object's address. Index registers can be used in numeric objects, on/off buttons, indicator objects and character objects.

Example

Index register1 is set as D10; the value in D10 is set as 2.

The numeric object address is set to write to M100 with index register as I1. This results in M100 + 2 = M102, to be written by the numeric object.



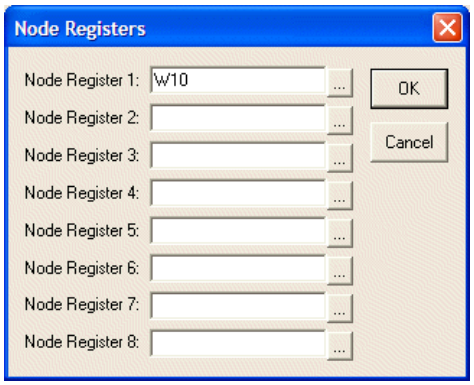
2.8.7 Node Registers

The Node Registers function makes it possible to define up to 8 controller registers; N1 - N8, that can be used in numeric objects, on/off buttons, indicators and character objects. The purpose is to define the controller station number for an object; the station number is the value in the defined controller register.

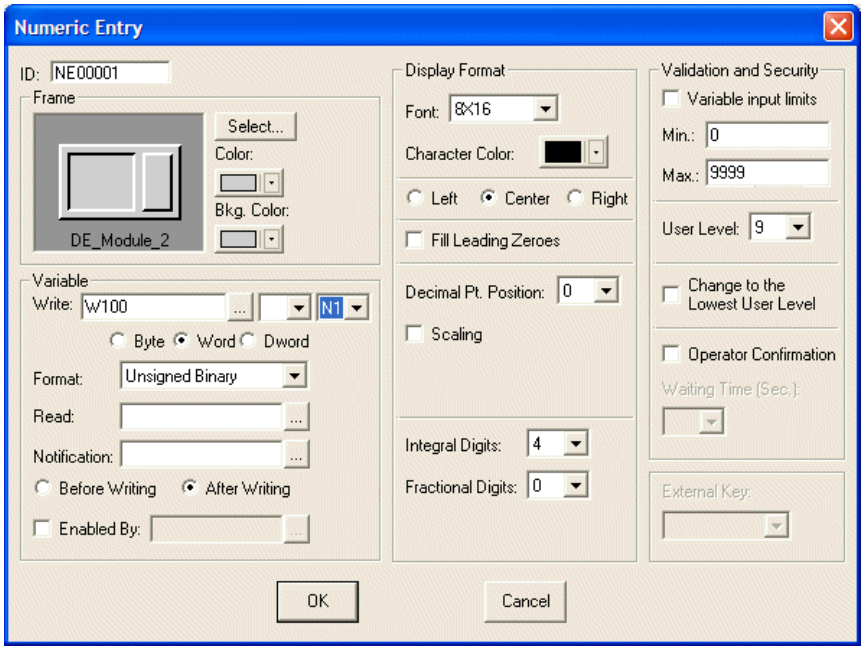
Note:
Only use this function when connecting to multiple controllers with the Modbus Slave driver.

Example

Node register 1 is set as W10; the value in W10 is set as 2.

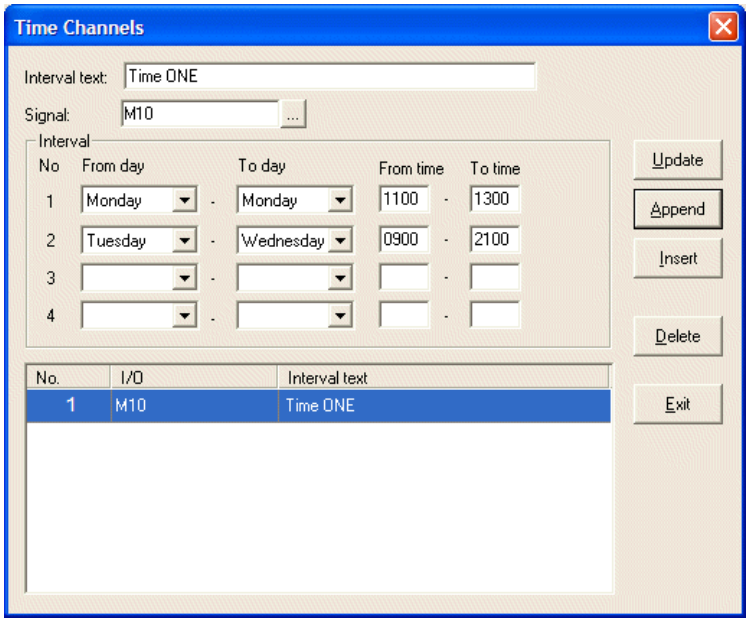


The numeric object address is set to write to W100 with node register as N1. This results in 2:W100 (where 2 is controller station 2).



2.8.8 Time Channels

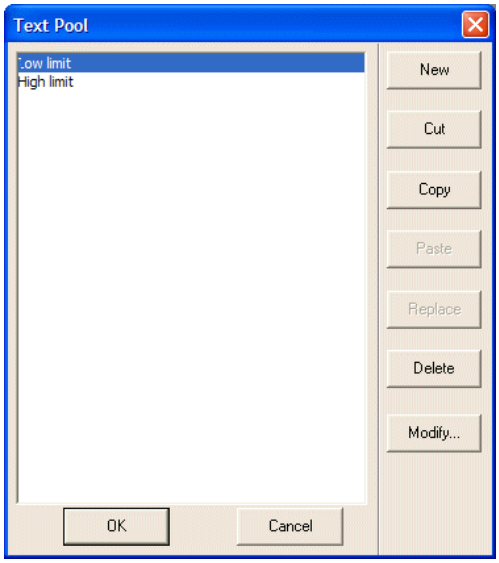
The **Time Channels** function makes it possible to set a defined bit in the controller to ON/OFF during an assigned time interval. A maximum of 4 time intervals can be defined; in any day of the week at any time.



2.8.9 Text Pool

The function of the **Text Pool** is mainly to provide common management and editing of the texts used in the application.

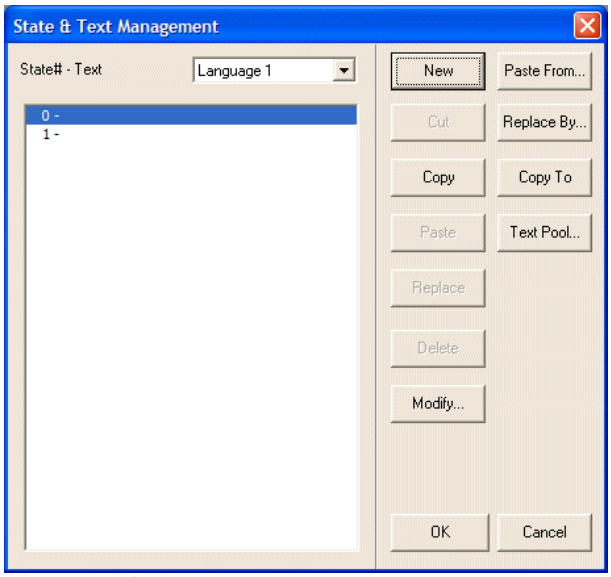
Select **Library/Text Pool**.



*The **Text Pool** dialog box*

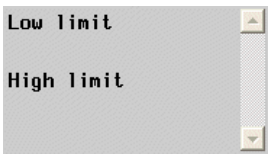
Perform the following steps to edit texts in the **Text Pool**:

1. Enter the desired texts in the **Text Pool** dialog box to save in.
2. Click on the object to edit an object which has texts, as in **Text Pool**, then select **Edit/State and Text Management**.



*The **State &Text Management** dialog box*

3. Select **Replace By** to edit. You can select the desired text which was edited in **Text Pool** to display.



*The following **List** object displays the text which was edited in **Text Pool**.*

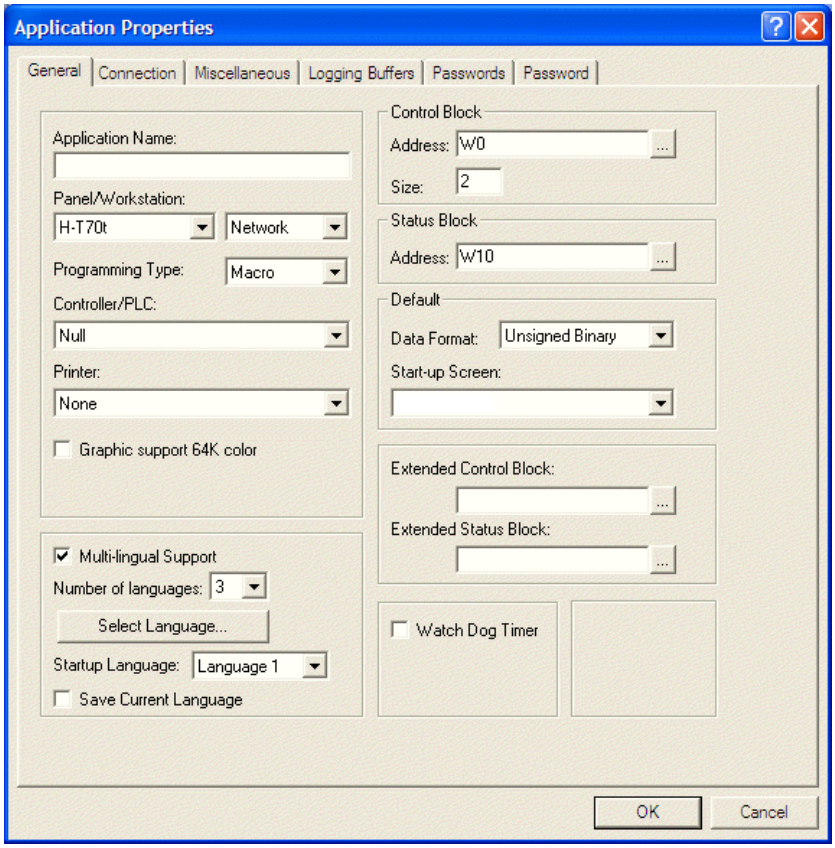
2.9 Application Menu

The **Application** menu is used for general management of the operator terminal and the software. You can set up configuration parameters for the operator terminal, such as controller type, operator terminal type, logging buffers and alarm setup. In addition, **Compile** and **Download** are also available in this menu.

2.9.1 Workstation Setup

Workstation Setup is used to set up the parameters of the operator terminal.

Select **Application/Workstation Setup** to display the following dialog box.



*The **General** tab of the **Application Properties** dialog box*

General Tab

On the **General** tab, you can set up the operator terminal and controller model, startup language and screen, and control block and status block.

Application Name: The name of the application.

Panel/Workstation: Specifies the model of operator terminal. Note the resolution, size, and color while selecting.

Programming Type: Selects **Macro** or **Ladder**.

Controller/PLC: Specifies the type of controller the operator terminal will communicate with.

Printer: Specifies the type of printer the operator terminal will print to.

Use external keys: Check this option if external keys are to be connected. Only available for H-T60.

Graphic support 64K color: Check this option to support 64K colors. Only available for H-T80 and H-T100.

Multi-lingual Support: Check this option to support multi-lingual use and specify the startup language. Supports up to 5 different languages including Arabic, Chinese Simplified, Chinese Traditional, Cyrillic, English, Greek, Japanese, Korean, Thai, Turkish and Western European. Please see section *Language 1-5* for setup. The **Multi-lingual Support** function allows that only one application file for a machine that can support up to 5 languages has to be maintained.

Control Block: Specifies the controller address to control, and size. The minimum size is 2 words, the maximum size is 32 words (the maximum size is 6 for recipe). The control block enables the controller to control actions in the operator terminal such as change screen, print, send recipes etc. Please see the chapter *Control and Status Block* for complete details.

Status Block: Specifies the starting address for the **Status Block**; the fixed size is 10 words. The **Status Block** provides communication between the operator terminal and the controller. The operator terminal will write a continuous block of data. Please see the chapter *Control and Status Block* for complete details.

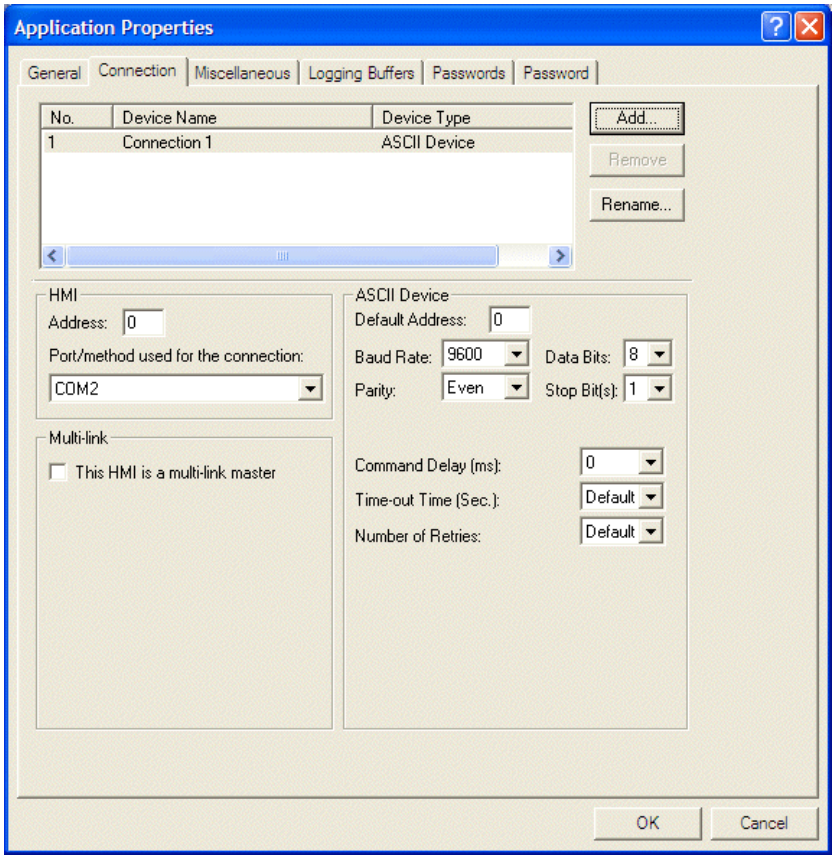
Data Format: Specifies the data format to be read.

Start-up Screen: Specifies the screen to display when the operator terminal starts up.

Extended Control Block and **Extended Status Block**: Please see sections *Extended Control Block* and *Extended Status Block* for details.

Connection Tab

On the **Connection Tab**, you can add or delete devices for connection and set up the parameters such as address, connection method and IP address.



*The **Connection** tab of the **Application Properties** dialog box; **Multi-link** is selected*

Add: Click to add new devices to connect with, including Muti-link, and select controller type to connect with. For the steps or methods related to setup, please see the chapter *Multi-Link: Normal Connection Port*.

Remove: Removes the connected device. The No. 1 device cannot be removed.

Rename: Modifies the device name and type, but the device type cannot be modified for the No. 1 device. This change has to be made on the **General** tab.

HMI:

Address: Sets up the operator terminal. Once the multi-link is made, the address can be repeated and the range is 0~255.

Port/method used for the connection: Specifies the port and method to connect with the controller or other operator terminal models including **COM1**, **COM2**, **Ethernet** (**Cross-link**), **COM1** (**Multi-link slave**), **COM2** (**Multi-link slave**) and **Ethernet** (**Multi-link slave**).

Multi-link:

– **This HMI is a multi-link master**: Check this option to specify the operator as a master.

  Master Port: Specifies the port which connects master with slaves; **COM1**, **COM2** or **Ethernet**.

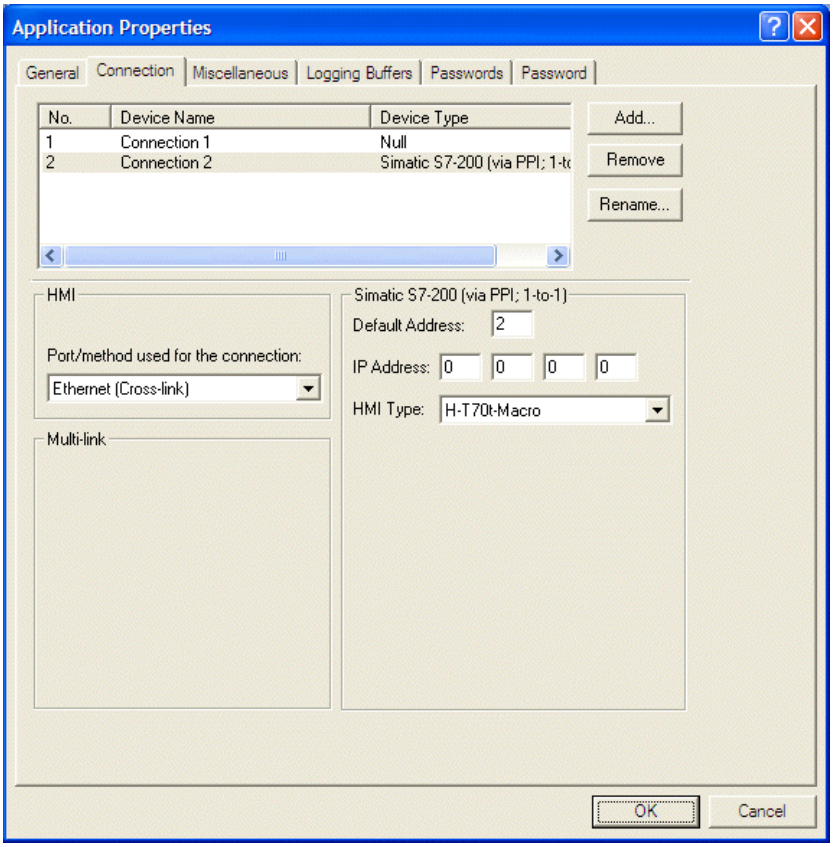
  Common Register Block: Specifies the starting location for the Common Register Block (CRB), which master and slaves use.

  CRB Size: Specifies the size of Common Register Block.

  Common On/Off Block: Specifies the starting location for the Common On/Off Block (COB).

  COB Size: Specifies the size of th Common On/Off Block.

Operator terminals arranged into a multi-link (one master; multi-slave) network is not available for all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.



***Cross-link** and connection to **Simatic S7-200 (via PPI; 1-to-1)** is selected*

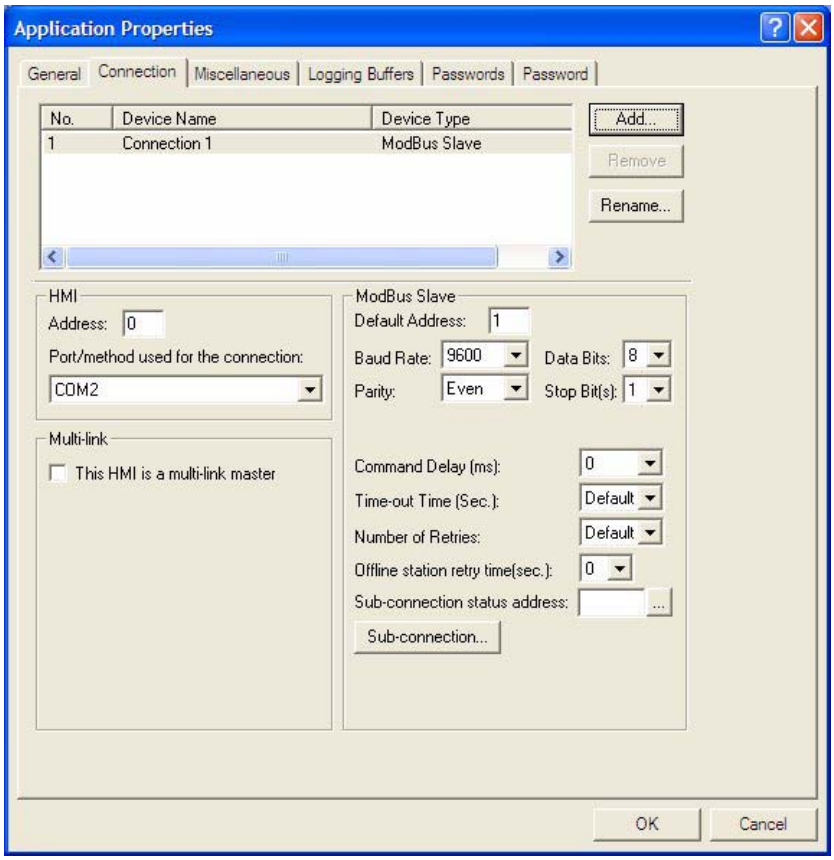
IP Address: Specifies the IP addresses of the other operator terminal. The specified operator terminal connects with the controller through the other operator terminal.

HMI Type: Specifies the model to connect with the controller (other operator terminals).

For the setup of Multi-link and Cross-link, please see the chapter *Ethernet Communication*.

Note:
The transmission parameters for the operator terminal and the controller must be identical as they are linked together. When the controller model is specified, H-Designer will set it up as the controller default, but you must ensure that this setup is identical in the operator terminal.

In addition to settings such as default address, baud rate and parity, some drivers, e.g. Modbus Slave, include special settings:



Offline station retry time (s): Number of seconds between retries when the station is offline.

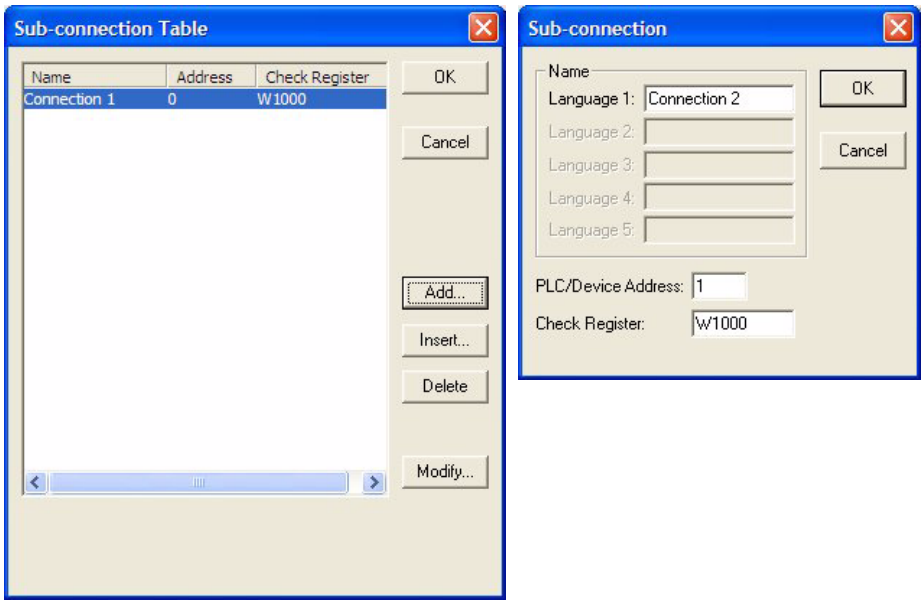
Sub-connection:

A sub-connection can be used when connecting multiple controllers; to allow the terminal to bypass a controller that does not respond, and continue without a communication error.

The terminal checks a defined controller and register for response in runtime, and if the register fails to response, this controller will be ignored.

The following example describes how to set up a sub-connection:

1. Click the **Sub-connection** button.
2. Click the **Add** button and state a name for the connection, the controller address and which register to use.

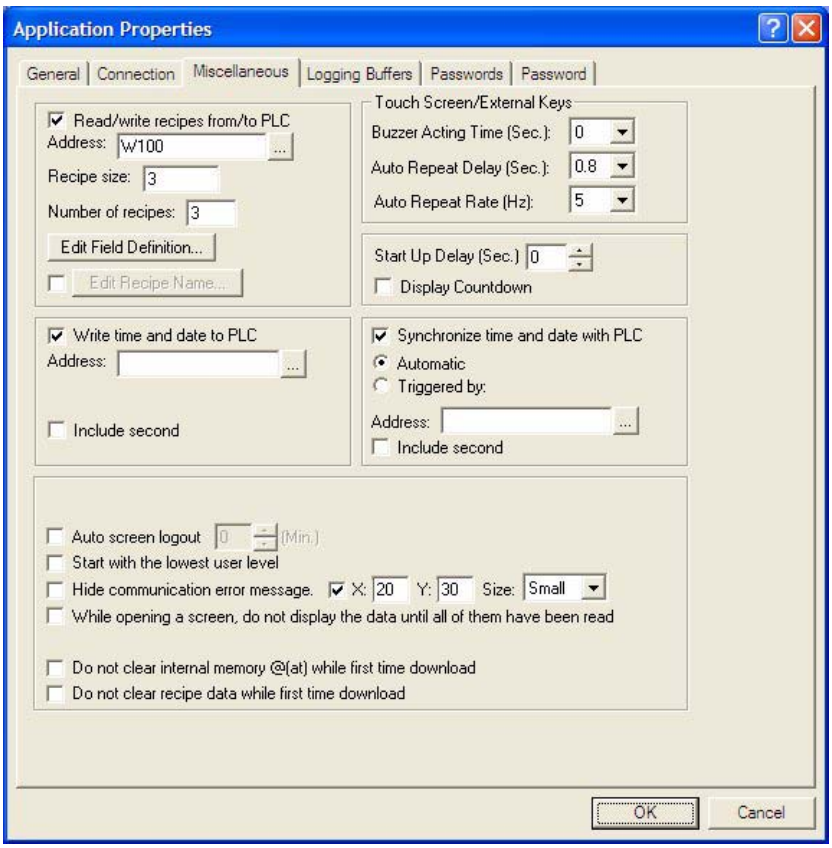


This sub-connection table is defined to check the controller register W1000.1 in station 0, and W1000.1 in station 1.

For the setup of each controller, please see the relevant controller manual or the driver help file.

Miscellaneous Tab

On the **Miscellaneous** tab, recipe functions and write time and data to the controller can be set up.



*The **Miscellaneous** tab of the **Application Properties** dialog box*

For setup of recipes, please see chapter *Recipes* and section *Recipe Register Block*.

The recipe function is not available for all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

Start up Delay (Sec.): Specifies the length of time before the screen start up.

Write time and date to PLC: Select this option to enable the operator terminal to write time and date to the real time clock in the controller. Please see section *Time Block* for details.

Synchronize time and date with PLC: Select this option to set up synchronization of time and date with the controller, automatically or triggered by a signal.

Touch Screen/External Keys: Specifies the format of the buzzer.

Auto screen logout: Enables automatic logout from the current screen after the specified number of minutes.

Start with the lowest user level: Checking this box overrides the dip switch setting **SW8 = ON** (user level 1). For a description of the dip switches, please see the installation and operation manual for the current operator terminal.

Hide communication error message: Checking the left-hand side box disables display of communication error messages in the operator terminal. Checking the right-hand side box makes it possible to modify size and location of communication error messages.

Do not clear internal memory (@) while downloading for the first time: Checking the box protects the original data in the terminal's internal memory from being overwritten when a new project is downloaded.

Do not clear recipe data while downloading for the first time: Checking the box protects the original recipe data in the terminal from being overwritten when a new project is downloaded.

Logging Buffers Tab

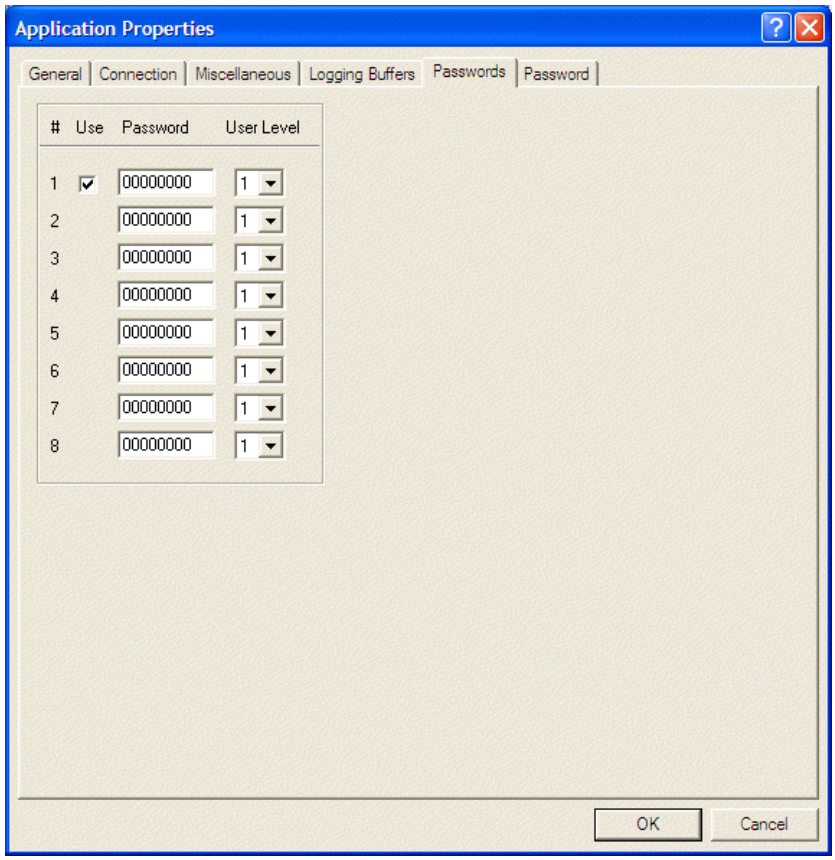
Logging Buffers are used to collect data from the battery backup RAM. It is a continuous data block and there are 12 buffers available.

The object is not available on all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

When creating a **Historical Display** object, the logging buffer's area and size has to be set up first. Please see section *Historical Display* for information.

Passwords Tab

You can set up pre-defined passwords with the settings on the **Passwords** tab. It is possible to configure up to 8 sets of passwords and user levels, where 1 is the highest user level. Objects in the operator terminal can be assigned corresponding user levels.

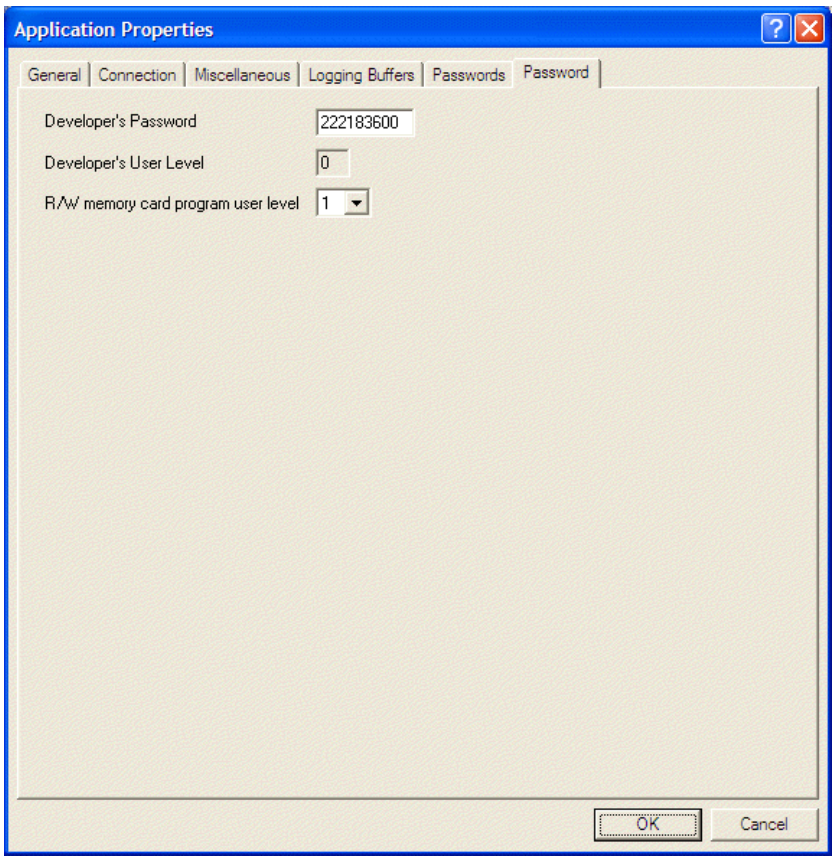


*The **Passwords** tab of the **Application Properties** dialog box*

This table can be used as a pre-defined password table in the operator panel.

Password Tab

When you want to copy the screen or upload the application to H-Designer, the operator terminal will ask for the password.



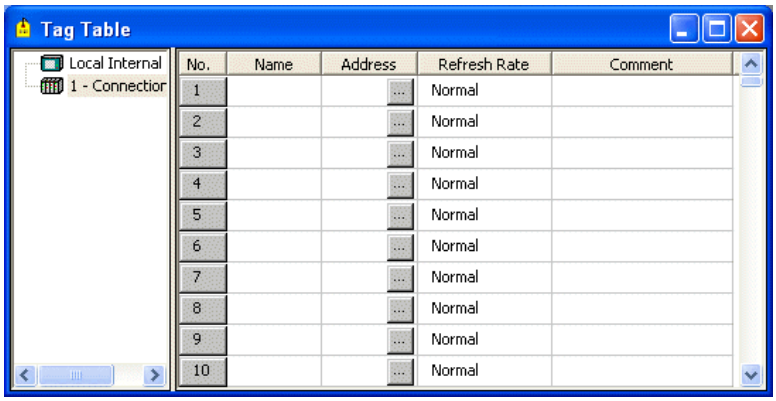
*The **Password** tab of the **Application Properties** dialog box*

R/W memory card program user level: Allows setting a certain user level required for access of a connected USB memory stick or Compact Flash memory card.

2.9.2 Tag Table

The **Tag Table** enables you to provide a name for the controller address and specify the refresh rate.

Select **Application/Tag Table**.

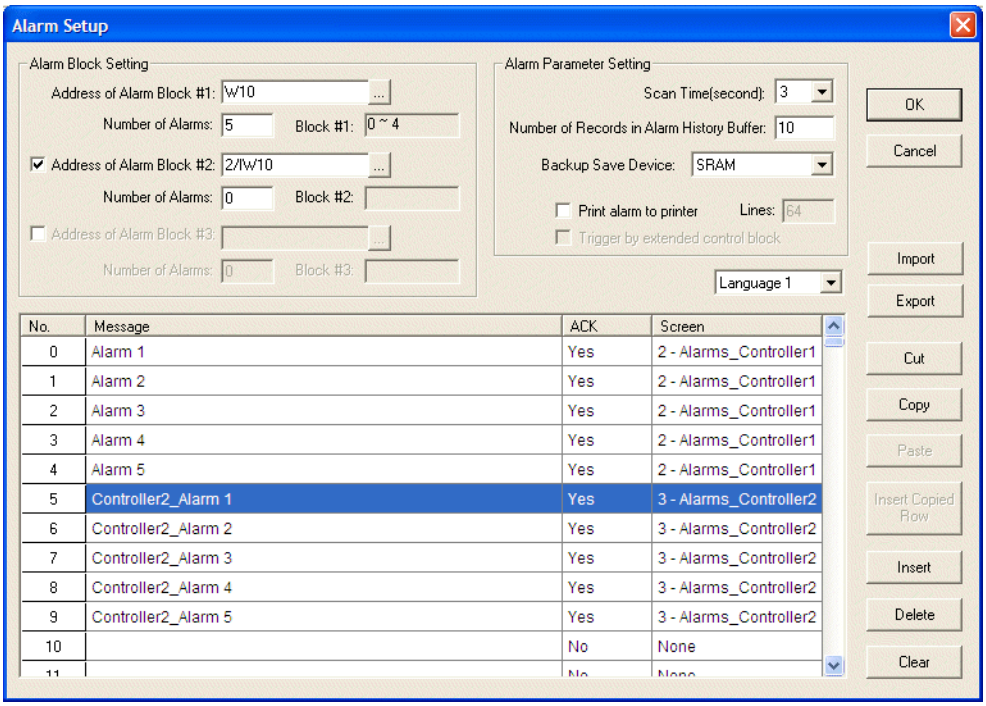


*The **Tag Table** window*

2.9.3 Alarm Setup

To use the **Historical Display** object, you must set up its address and parameters first. Then the operator terminal will display the corresponding messages after reading the controller value. Up to 512 messages can be specified.

Select **Application/Alarm Setup**.



*The **Alarm Setup** dialog box*

Alarm Block Setting

It is possible to set up three different alarm groups and each alarm group can be connected to a different controller.

Address of Alarm Block: To use a bit (LSB) as a corresponding alarm address. If **W130** is the starting position and the number of the alarm is set 160, the operator terminal will monitor 160 bits = 10 words, and this corresponds to W130, W131, W132……W139. When bit **W130** turns on, the operator terminal will sample and record an alarm message.

Number of Alarms: Specifies the number of alarms.

Alarm Parameter Setting

Scan Time: Specifies the sampling time for monitoring the controller data, 1-10 seconds.

Number of Records in Alarm History Buffer: Specifies the maximum number of events stored in the alarm buffer. For example, **100** means that when the 101st alarm event occurs, the first alarm message will be overwritten.

Backup Save Device: Select where to save Alarm history data; on **SRAM**, **CF Card** or **USB Memory Stick**.

Print alarm to printer: Set up if alarms that become active are to be printed (to the printer specified under **File/Printer Setup**). The printout will contain the alarm text of the active alarm. Also select number of lines to be used for each alarm. The default value, 64, uses one A4 sheet per alarm, but multiple alarms can be printed on the same page by reducing number of lines.

This function can also be triggered by the extend control block.

Table

No.: The number of the alarm.

Message: Enter the text to the alarm message. The format can be modified in its dialog box. Up to 512 alarm messages can be set.

ACK: Acknowledge the message which has been received to conceal the alarm.

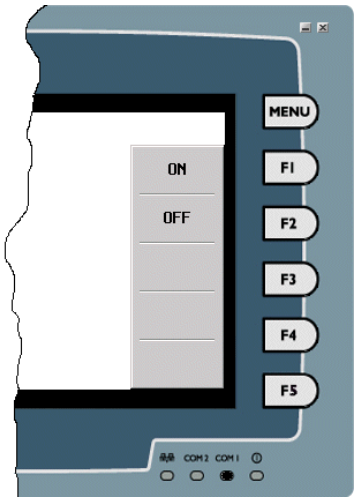
Screen: Specifies the screen to display when the alarm occurs.

Alarms can be imported from and exported to a .csv file using the **Import** and **Export** buttons.

2.9.4 Slide-out Menu

The function of the **Slide-out Menu** is to operate the functional keys, such as the **Set Button**, **Reset Button** and **Momentary Button**, in a convenient way in the operator terminal. The number of functional keys depends on the selected operator terminal model.

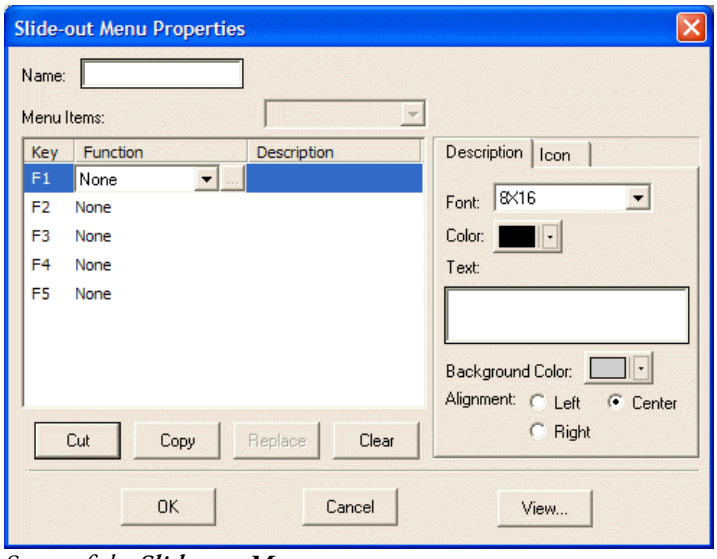
When the **Menu** button is clicked, the operator terminal will display the slide-out menu with its specified functional keys.



*The **Slide-out Menu***

The object is not available on all operator terminal models; please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

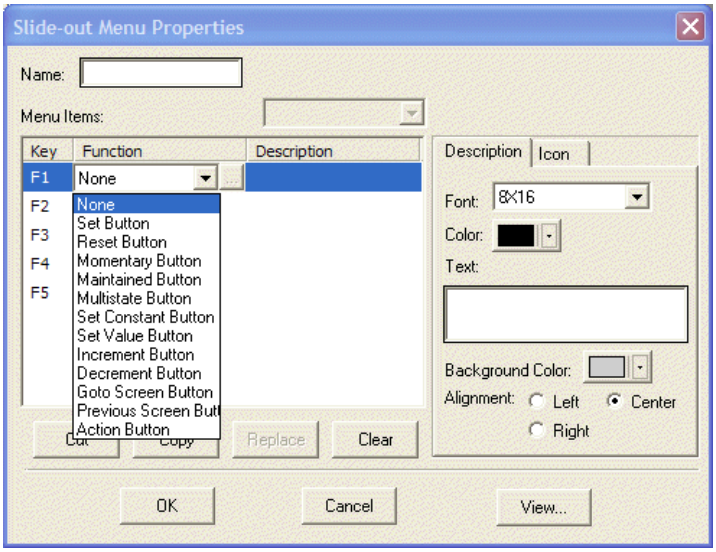
Select **Application/Slide-out Menu**. Click **New** to display the following dialog:



*Setup of the **Slide-out Menu***

Name: Enter the name for the slide-out menu.

Function: Select a function from the drop-down list.



*Selecting a function for a button on the **Slide-out Menu***

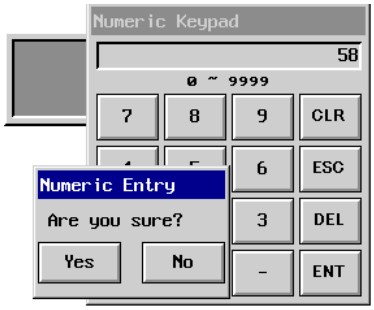
Cut, **Copy**, **Replace**, **Clear**: Use the buttons to cut, copy, replace and clear the button's content.

Description: Enter the name for the button in the **Text** block.

View: Click the View button to view the created slide-out menu.

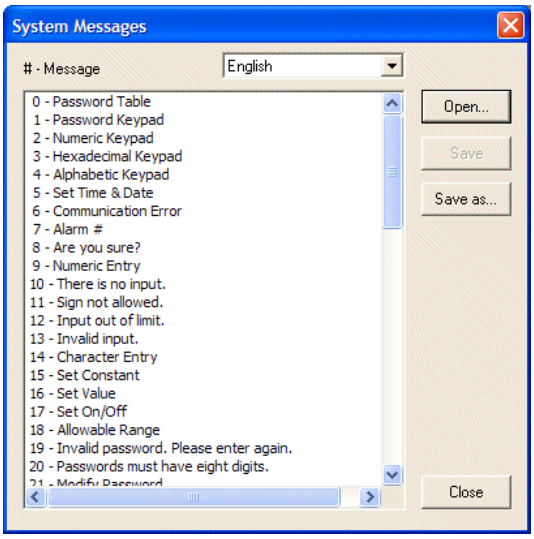
2.9.5 System Message

The **System Message** is used to edit messages for the operator terminal system. When the **Operator Confirmation** option is selected, executing the object will display its system message. For example, the system message **Are you sure?** can be displayed after the numeric entry.



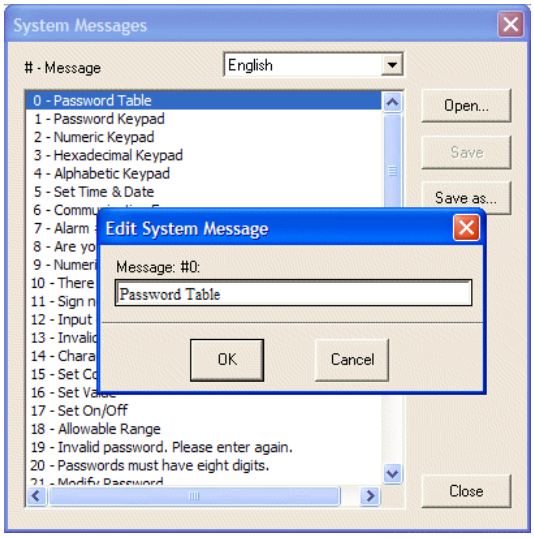
*An example of a **System Message***

Select **Tool/System Messages**.



*The **System Messages** edit window*

Double-click on the message block; the message can be modified in its dialog box as shown below.



*Editing **System Message***

The **Save as Default** button is to save the system messages as a default file (*.PSM).

2.9.6 Macros

Macros enable the operator terminal to execute a number of tasks including flow control, data transfer, conversion, counter, system service instructions, etc. Using macros can not only help you communicate to the controller but also connect to other devices. This feature provides an efficient integration system as well as an economical structure for hardware application. In addition, using macros can also significantly reduce program size and optimize controller efficiency.

Please see the chapter *Macros* for complete details.

There are three macro options in the **Application** menu; **INITIAL Macro**, **BACKGROUND Macro** and **CLOCK Macro**.

INITIAL Macro

When the operator terminal runs the application for the first time (this means the first time the application is executed after power off), this macro is executed once. The purpose of **INITIAL Macro** is for example, data initialization or communication parameters declaration.

BACKGROUND Macro

When the operator terminal runs the application, the command will be executed cyclically. A maximum 30 lines of macro commands can be executed at once. Whatever the screen is, the macro commands will be executed. The purposes of the **BACKGROUND Macro** include communication control, data conversion etc.

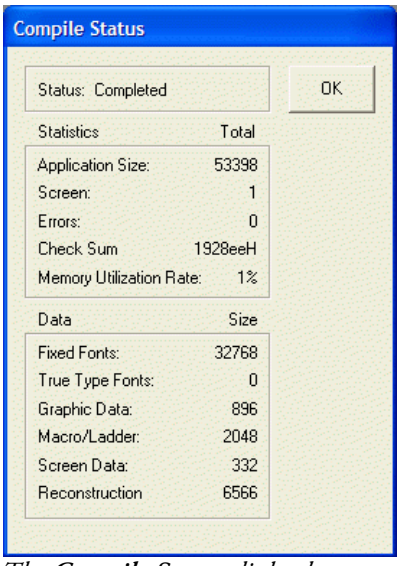
CLOCK Macro

When the operator terminal runs the application, the entire macro will be executed once every 500 ms. The purpose of the **CLOCK Macro** is screen control, bit setting, command control, data transfer etc.

2.9.7 Compile

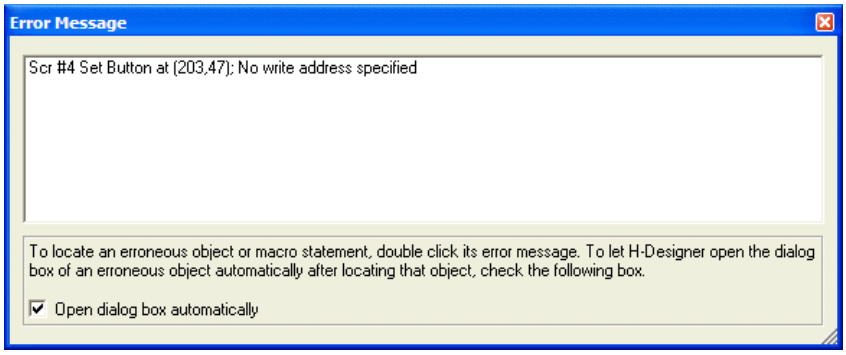
The **Compile** command is used to test the application to see if any errors happen before the application is executed. After correcting the errors, the application can be executed.

Select **Application/Compile**.



*The **Compile Status** dialog box*

After clicking **OK** the **Error Message** dialog box will be displayed on the screen.



*The **Error Message** dialog box*

Double-click the error message to display the incorrect object or macro on the screen. Alternatively, check **Open dialog box automatically**, to automatically display the incorrect object.

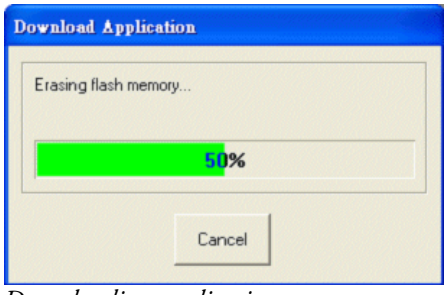
2.9.8 Compile (USB/CF)

The **Compile** (**USB/CF**) function makes it possible to update a running terminal project without connection to a PC (via download cable).

The function will generate two application files, *app_main.c64* and *app_main.fw6*, that you can copy to a USB memory stick or Compact Flash memory card. When the USB memory stick or Compact Flash memory card is connected to the terminal during runtime, the terminal will detect the files and ask if you wish to update the application.

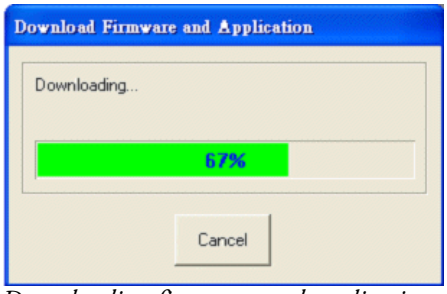
2.9.9 Download Application and Download Firmware and Application

Download Application is used to download the updated application and screen to the operator terminal.



Downloading application

Download Firmware and Application is used to download the firmware and application to the operator terminal. This option has to be selected the first time you download the application.



Downloading firmware and application

If the connected operator terminal is not the same as the model specified in the project, a message box will appear on the screen.

Note:
It is possible to perform compile automatically before download, according to section *2.11.10 Editing Options*. Otherwise, remember to execute **Compile** before downloading.

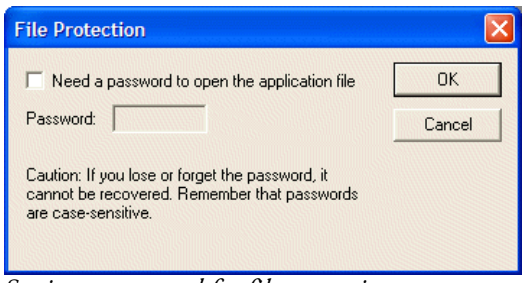
2.9.10 Ethernet Download

The **Ethernet Download** function makes it possible to download a project to a running terminal without confirmation on the terminal side. The function is supported only by terminal models with Ethernet, and you will be prompted to enter the developer password on the PC. See section *Password Tab* for details about the developer password.

2.9.11 File Protection

File Protection is used to protect the application; you must enter the password to open the application file.

Select **Application/File Protection**.



Setting a password for file protection

Note:
This password is used to protect the application file from being modified by unauthorized users, and is different from the password set on the *Password Tab*. That password provides security against copying and uploading.

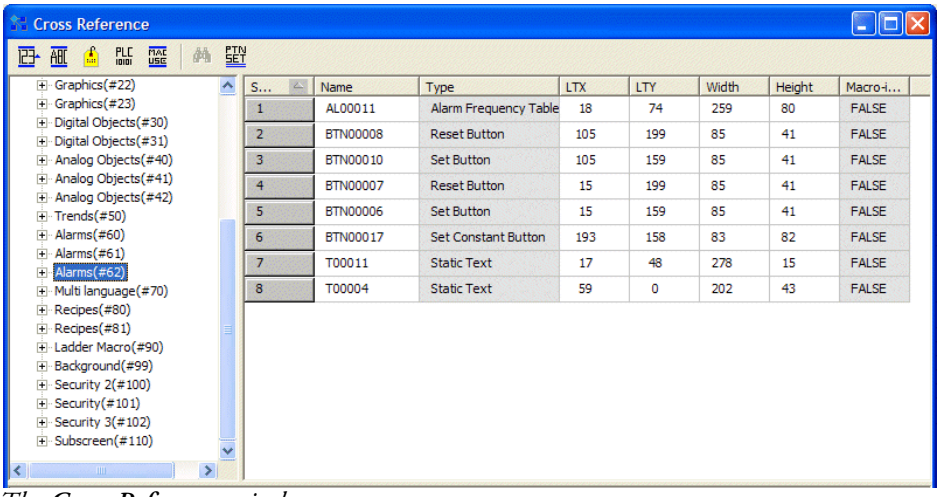
2.10 Tool Menu

The **Tool** menu is used to manage and simulate the application, and to edit recipes.

The following options are available for selection from the **Tool** menu: **Cross Reference**, **Off-line Simulation**, **On-line Simulation, View/Edit Recipes** and **System Messages**.

2.10.1 Cross Reference

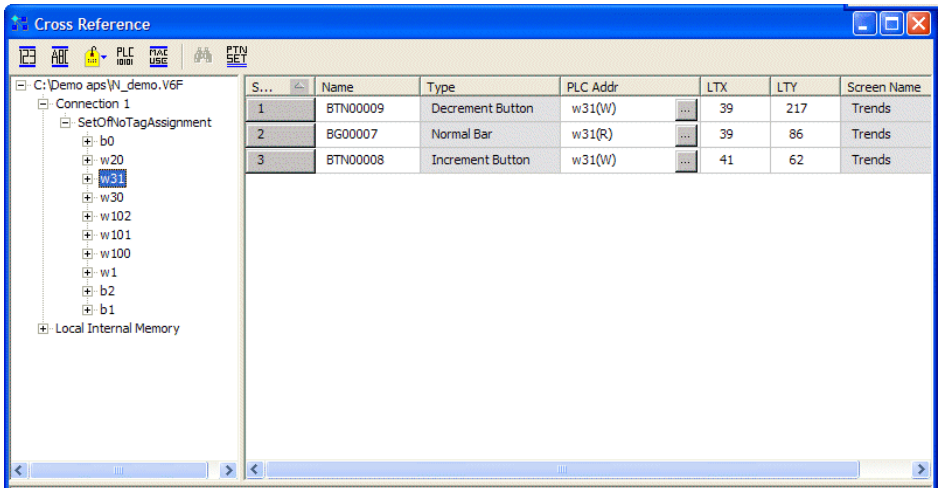
The **Cross Reference** tool helps you identify the screen name, screen number, controller address, tag address or macro-in use quickly.



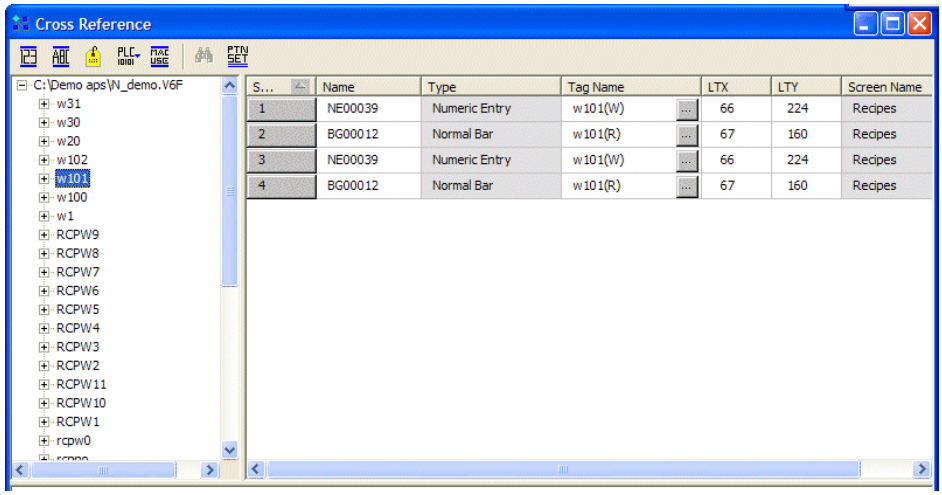
*The **Cross Reference** window*

The icons can be used to sort according to screen name, screen number, controller address, tag address or macro-in use. The right-hand table will list its objects and properties. The preview window below will display the selected object. Click on a heading of a column in the table (such as **Serial No**, **Name** or **PLC Addr**) to sort the properties in ascending or descending order.

Icon	Description
	Select this icon to sort by screen number.
	Select this icon to sort by screen name.
	Select this icon to sort by tag name.
	Select this icon to sort by controller address in ascending order; click the address to list the object.
	Select this icon to sort by Macro-in use objects.
	Select this icon to sort by search pattern. This feature is only available when searching controller addresses or tag names.
	Set the desired pattern for search. Select **by PLC Addr** or by **Tag Name**, **Exact match** or **Partial match**.

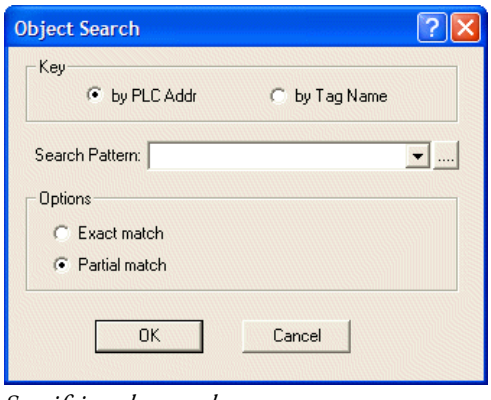


*Sorting by **Tag Name***



*Sorting by **Address**; clicking the address displays the object*

When searching controller addresses or tag names, the seach pattern can be specified. If the search pattern is **W**, and **Partial match** is selected, the cross reference will display objects with the controller addresses W - W0, W1, W20, W60.



Specifying the search pattern

2.10.2 Off-line and On-line Simulation

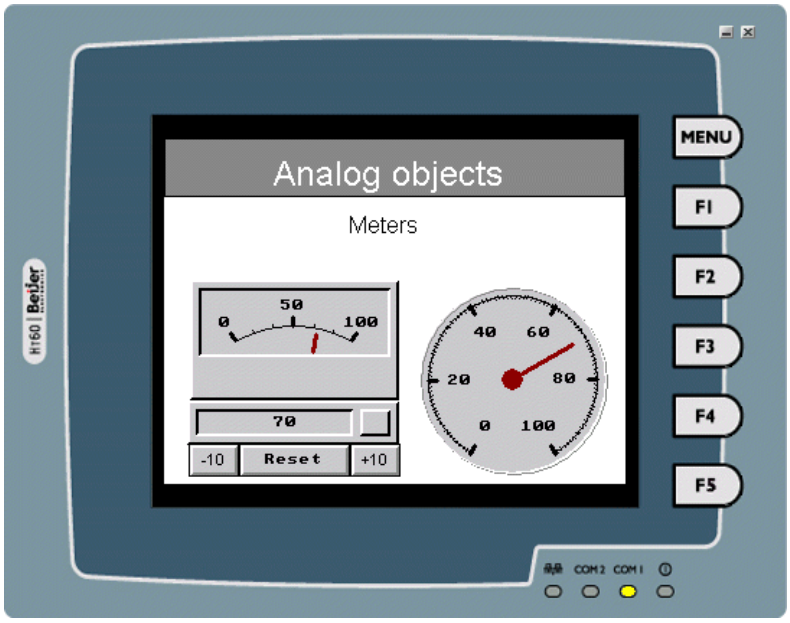
H-Designer supports two types of simulation; **Off-line Simulation** and **On-line Simulation**. They both offer simulation of the operator terminal in the PC.

Note:
The application must compile before simulation.

Off-line simulation is available with all controllers which H-Designer offers, but on-line simulation is only available on some controllers.

Off-line Simulation

Off-line Simulation can be used to present results on a PC with the same operation mode as between the operator terminal and the controller.



H-T60 in off-line simulation

Off-line simulation is without communication with the controller, and provides the following benefits:

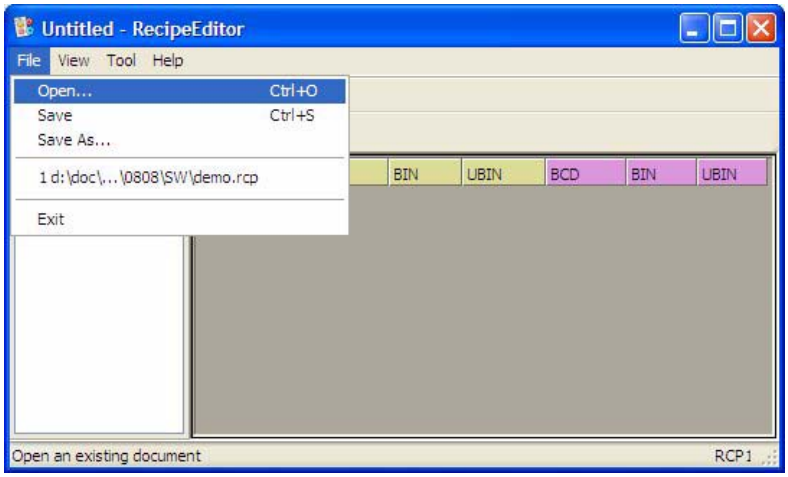
1. Before purchasing, you can simulate operation and recognize the operator terminal functions sufficiently.
2. Before downloading, you can simulate on the PC to test the application, including screen change, button functions and display etc.
3. Before the completion of the controller program, the operator terminal application can be presented to the customer.

On-line Simulation

H-Designer offers user **On-line Simulation** for connection between the PC and the controller communication ports. If there is only one RS232C serial port in the PC, you need to add an adapter for transferring the signal from RS232C to RS422 or RS485 in order to connect with the RS422 or RS485 port in the controller. Note that the communication time between H-Designer and controller is 60 minutes. If you need to connect again, please close the H-Designer and restart it.

2.10.3 View/Edit Recipes

Select **Tool/View/Edit Recipes** to display the recipe editor window.



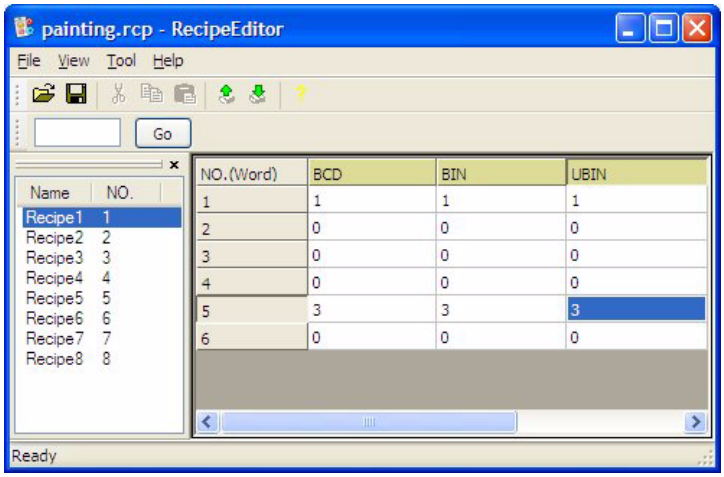
*The **Recipe Editor** window*

Note that the recipe file must be uploaded from the operator terminal and saved in the PC. The recipe size and total number of of recipes cannot be modified.

Recipe functions are not available for all operator terminal models; see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

For setup procedures, please see section *Application Menu* and chapter *Recipes*.

Select **File/Open** to open the selected recipe file. The recipe is displayed on the screen for you to edit.



Editing Recipe #1

The function of the recipe editor is the same as a common edit tool; it includes open, save, print, view the recipe file, and window arrangement.

Recipe files can be exported and imported in .csv format using the buttons in the Recipe Editor. The Recipe Editor also includes a help file, that describes Recipe Editor functions closer.

2.11 Options Menu

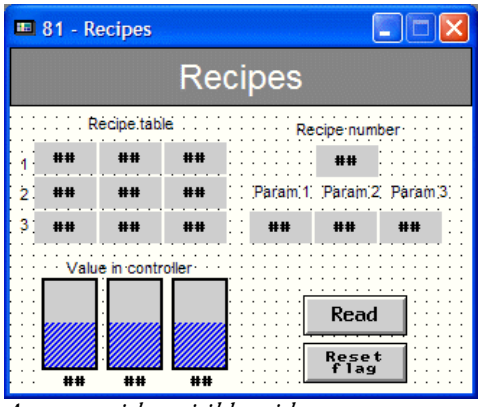
The **Options** menu provides options for editing and transmission.

2.11.1 Snap to Grid

If you select **Snap to Grid** on the edit screen, the edited objects will align to the nearby grid (see section *Display Grid*). This command is convenient for aligning objects.

2.11.2 Display Grid

When selecting this option, the edit screen will display a grid to allow easy alignment.



A screen with a visible grid

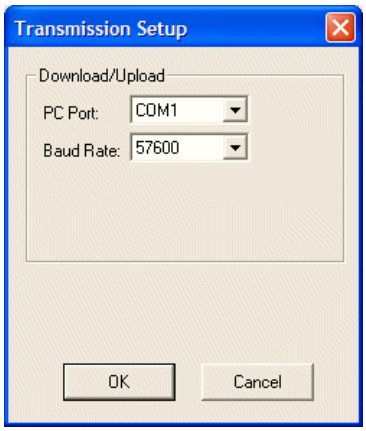
2.11.3 Grid Attributes

Select this option to specify the grid size. The bigger the grid size is, the longer the distance between points will be.

2.11.4 Transmission Setup

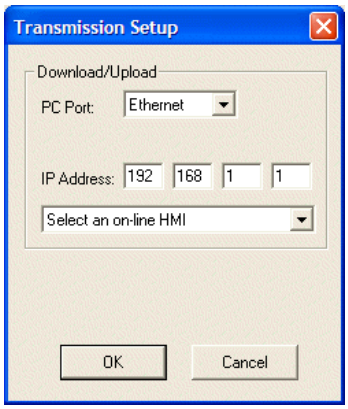
Select **Options/ Transmission Setup** to specify the download/upload port and baud rate between the PC and the operator terminal. The PC port options are **Ethernet**, **COM1…. COM16** and **USB**. A baud rate of **115200** is recommended.

Note:
These settings are not the same as the ones set on the **Connection** tab in **Application/ Workstation Setup**. The former is the setting between the PC and the operator terminal, the latter is the setting between the operator terminal and the controller.



*The **Transmission Setup** dialog box*

If **Ethernet** is selected for **PC Port**, you need to input the PC address or select from the drop-down list.



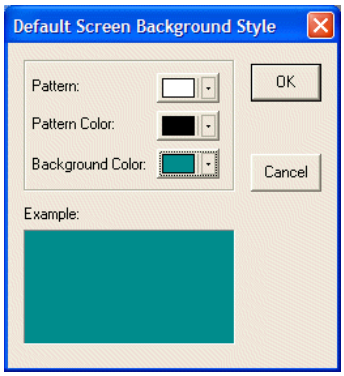
*Selecting **Ethernet** for **PC Port***

2.11.5 Language Selection

Select **Options/Language Selection** to select another language for H-Designer. H-Designer has to be restarted in order for the new language selection to take effect.

2.11.6 Default Screen Background Style

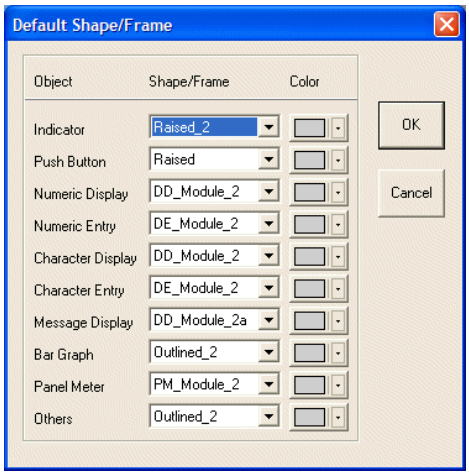
Select **Options/Default Screen Background Style** to specify the pattern, pattern color, and background color for all screens. The default setting will be displayed in all screens except for the special edit screen.



*The **Default Screen Background Style** dialog box*

2.11.7 Default Frame Styles

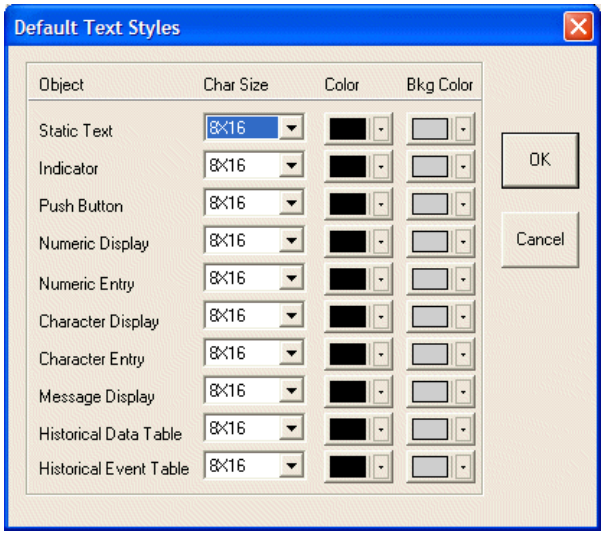
Select **Options/Default Frame Styles** to specify the different object types with their own shape/frame. Then those objects will be shown as specified on the screen.



*The **Default Frame Styles** dialog box*

2.11.8 Default Text Styles

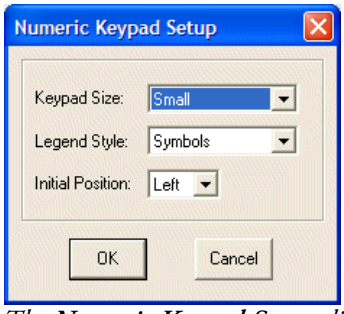
Select **Options/Default Text Styles** to specify character size, color and background color for each type of object. The objects will be displayed with their settings on the screen.



*The **Default Text Styles** dialog box*

2.11.9 Numeric Keypad Setup

Select **Options/Numeric Keypad Setup** to set up the numeric keypad on the screen (displayed for example when clicking the **Numeric Entry** object).



*The **Numeric Keypad Setup** dialog box*

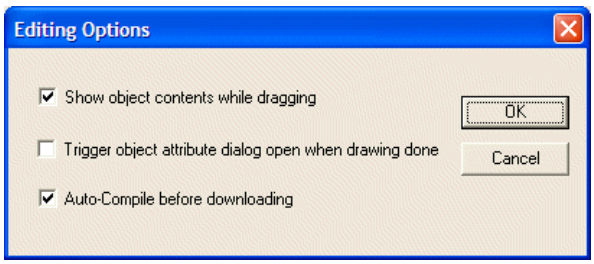
Keypad Size: **Small** or **Large** can be selected.

Legend Style: **Symbols** or **Chinese Characters** can be selected.

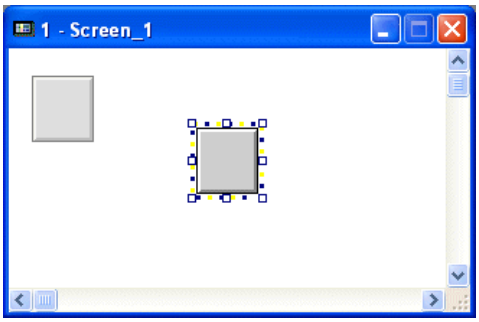
Initial Position: **Left** or **Right** can be selected.

2.11.10 Editing Options

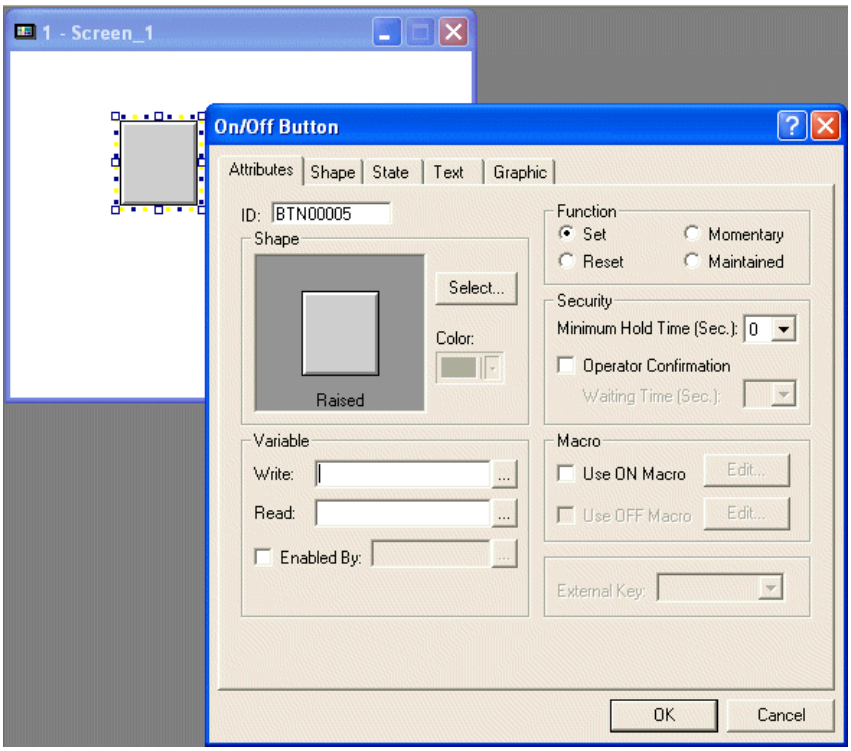
Select **Options/Editing Options**, the dialog box will be as the following on the screen. You can setup the edit environment here.



If **Show object contents while dragging** is selected, the object shadow will show while dragging.



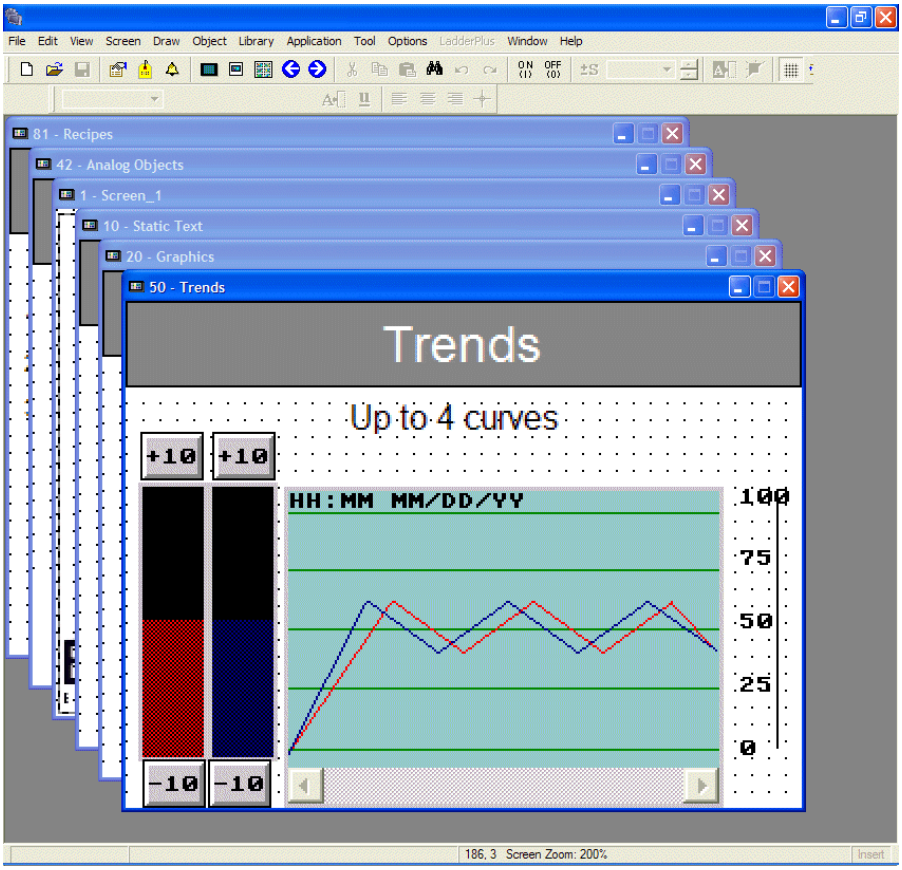
If **Trigger object attribute dialog open when drawing done** is selected, the object attributes dialog box will pop up directly when drawing an object or geometric shape is done.



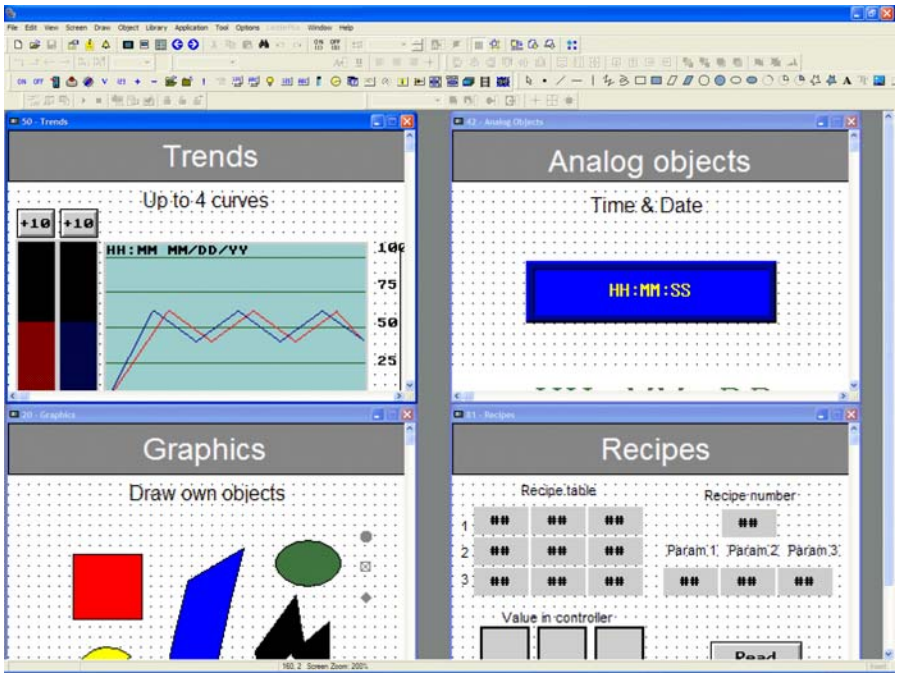
The **Auto-Compile before downloading** option automatically checks and compiles the project before downloading to the operator terminal. If errors are found, these are displayed and must be corrected before downloading the project. See also section *2.9.7 Compile*.

2.12 Window Menu

The **Cascade**, **Tile** and **Close All** options available in the **Window** menu and all open screens are listed.



Cascade is selected for screen display



Tile can be used when copying between screens and for an overview of screens

Close All is used to close all open screens at once; the screens will not be saved. The application is not closed.

2.13 Help Menu

The **Help** menu offers complete details and instructions about **Macros** and **Ladder-Plus**.

Selecting **Help/About**, displays the version number and copyright information of H-Designer.

3 Recipes

Recipes include blocks of similar systematic data. You can edit them as a group of recipes for convenient transmission, sending data efficiently and accurately.

Note:
Recipes are not available for all operator terminal models: please refer to *Appendix A - H-Designer Features and Operator Terminal Models* for more details.

3.1 Example

The coating equipment described below is used for spraying paint on different parts. The colors are limited to white, red, blue, dark, and mixed color (1 = spray, 0 = no spray). Recipes can be used to present and save data to simplify matters.

Coating Equipment

Color	White	Red	Blue	Black	Time
Top	1	0	0	0	3
Bottom	0	1	0	0	2
Left	0	1	1	0	1
Right	0	0	0	1	1

Five variables of the recipe data: **White**, **Red**, **Blue**, **Black** and **Time**.

The following recipes are to be created:

– **Recipe 1** paints top white in 3 minutes

– **Recipe 2** paints bottom red in 2 minutes

– **Recipe 3** paints left side purple (red+blue) in 1 minute

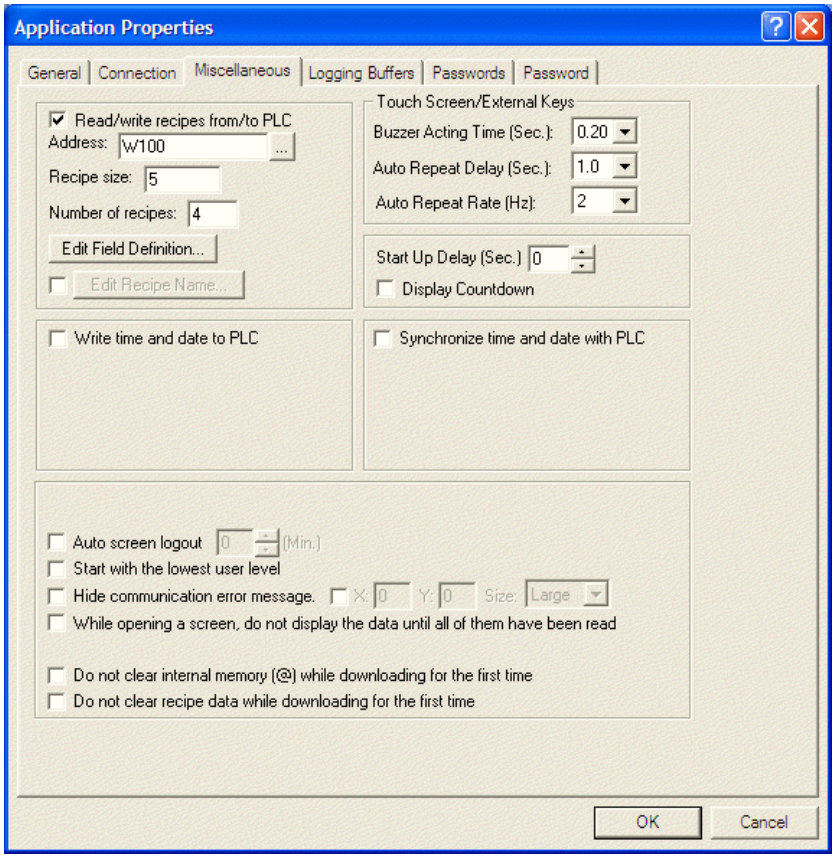
– **Recipe 4** paints right side black in 1 minute

A variable represents a word, the recipe size is five, and the number of recipes is four.

3.2 Recipe Operation Steps

This section illustrates the operation and application of recipes.

1. First, a **Recipe Register** has to be defined. Select **Application/Workstation Setup** and enter the **PLC address**, **Recipe size**, and **Number of recipes** on the **Miscellaneous** tab.



*Setting **Recipe size** = 5 and **Number of recipes** = 4 in this example*

The starting address of the controller recipe register is **W100** and its size is five words. The starting address of the current recipe in the operator terminal is **RCPW0** and its size is five words. The starting address of the operator terminal RAM is **RCPW5** and its size is 5×4 words.

If the recipe write flag is ON, the operator terminal writes the current recipe from **RCPW0**-**RCPW4** in the operator terminal to **W100**-**W104** in the controller. If the recipe read flag is ON, the operator terminal writes the recipe from controller **W100**-**W104** to **RCPW0**-**RCPW4** in the operator terminal. If you want the operator terminal to read/write the recipe data from/to the controller, **RNR** (Wn+5) must be defined.

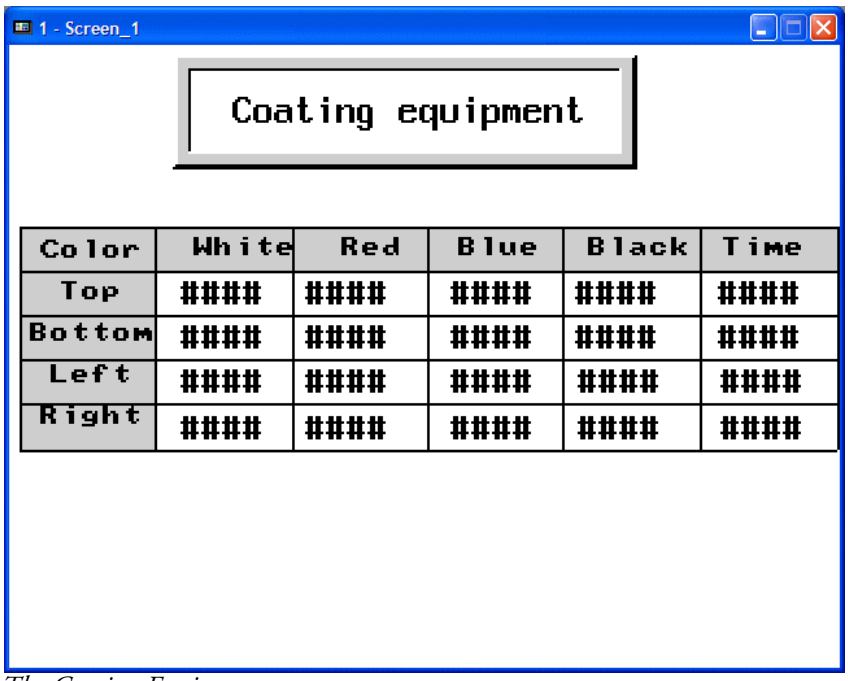
Edit Field Definition: Edits the context, such as format, offset and size, of a recipe.

Edit Recipe Name: Allows entering alphabetical characters in user-defined recipe names.

For instructions about operator terminal recipe registers, please see section *Recipe Register Block*.

2. The application can be edited and saved as a *.V6F file. In the example, a **Numeric Entry** object is used to display the coating equipment data in the operator terminal.

– **RCPW5-RCPW9** represent top recipe data
– **RCPW10-RCPW14** represent button recipe data
– **RCPW15-RCPW19** represent left side recipe data
– **RCPW20-RCPW24** represent right side recipe data

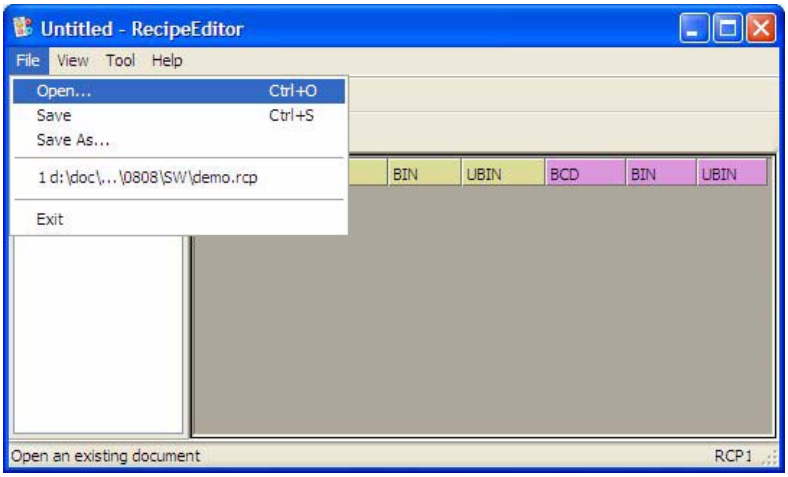


The Coating Equipment screen

3. Download the H-Designer file to the operator terminal. Select **Download Application** in the operator terminal, then select **Application/Download Firmware and Application** in H-Designer.

4. Upload the recipe from the operator terminal to H-Designer. Select **Upload Recipes** in the operator terminal, then select **File/Upload Recipes** in H-Designer. The recipe will be uploaded to H-Designer and saved as *.rcp.

5. The recipe file can be opened for editing. Select **Tool/View/Edit Recipes** to display the H-Designer **Recipe Editor** dialog box.

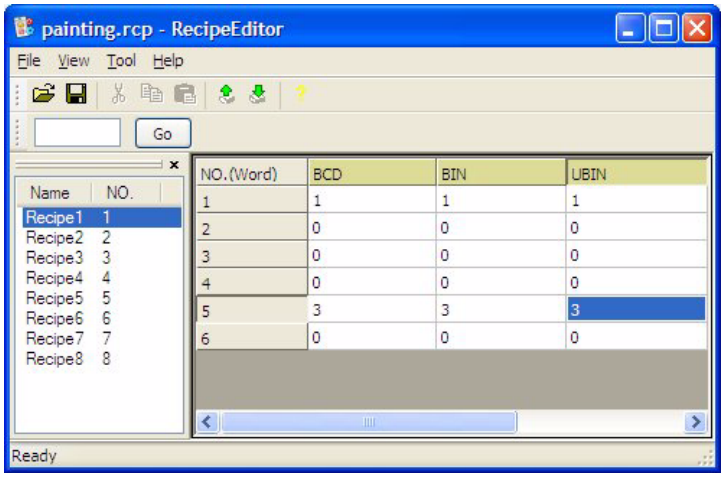


*The **Recipe Editor** dialog box*

6. Select **File/Open** and select the desired recipe file (e.g. **painting.rcp**).

Note:
The recipe file must be uploaded from the operator terminal and saved in the PC first. The recipe size and total number of of recipes cannot be modified.

7. The dialog box appears on the screen. The data can be edited in the dialog box. Note that the count for editable data is contained in data size. After editing, select **File/Save** to save the data. For example, Recipe 1 is (1,0,0,0,3), Recipe 2 is (0,1,0,0,2), etc.



Editing recipe data in the Recipe Editor

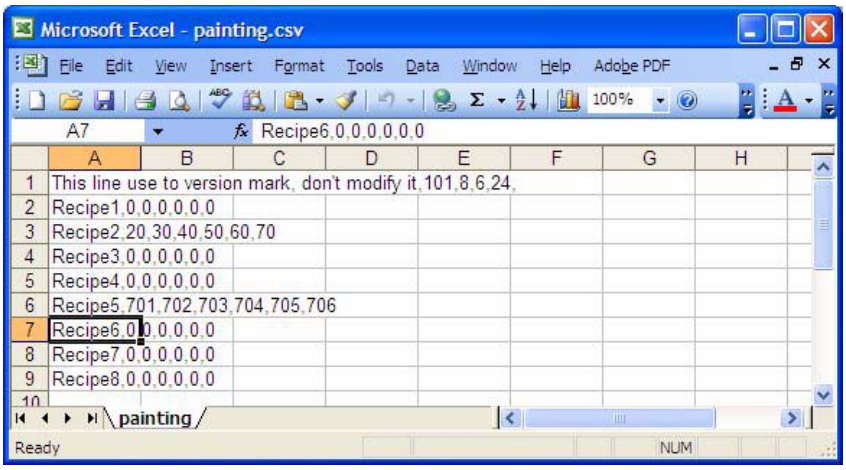
8. Select **Download Recipes** in the operator terminal, and then select **File/Download Recipes** to download the recipe file.
9. Select **Run** in the operator terminal. The operator terminal displays the recipe, filled with the same data as displayed on the screen in H-Designer. The coating equipment example includes painting methods and time.

Coating Equipment

Color	White	Red	Blue	Black	Time
Top	1	0	0	0	3
Bottom	0	1	0	0	2
Left	0	1	1	0	1
Right	0	0	0	1	1

The recipe data in the operator terminal

Recipe data can also be exported in .csv format, to be opened and edited in e.g. Excel.



Editing recipe data in Excel

The data can then be imported to the recipe editor again.

The Recipe Editor also includes a help file, that describes Recipe Editor functions closer.

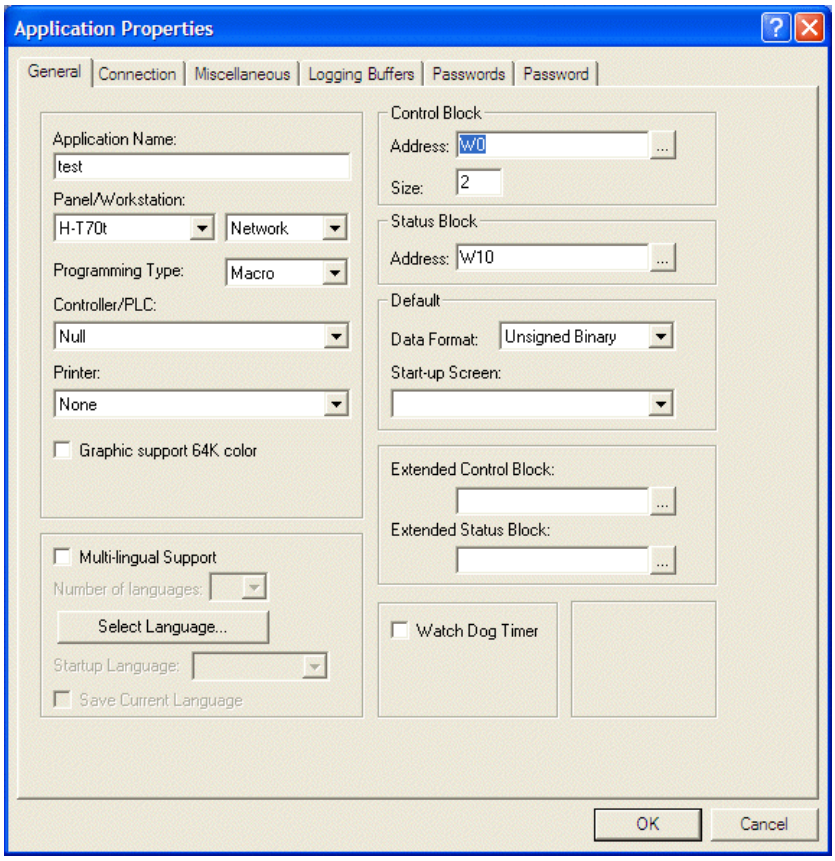
3.3 Recipe Controlled by Controller

This section introduces the controller setup and corresponding execution related to the operator terminal. For an introduction to communication between the controller and the operator terminal, please see the chapter *Control and Status Block* for complete details.

After completing the recipe, the controller is assigned to control the recipe through the communication link between the controller and the operator terminal. The controller can read/write the recipe from/to the operator terminal.

Steps to read a recipe from the controller to the operator terminal:

1. Set up two continuous blocks; one is the control block, the other is the status block. Select **Application/Workstation Setup** in H-Designer, enter the controller address and its size for **Control Block** and **Status Block** on the **General** tab. The control block size must be set to six words.
2. Example: coating equipment. The control block for the NULL controller W0-W5, the size is six, and the status block is W10-W15.



Setting up control and status blocks

For the properties not explained in this section, please see sections *Control Block* and *Status Block*.

The recipe register is defined in section *Recipe Operation Steps*, the starting address is W100 and the size is five.

In the table below words and their corresponding names in the controller are described. The words related to read/write recipes are in the blue background. Please see chapter *Control and Status Block* for more details.

Controller Internal Data Block

	Word	Bit																Member
Control Block	W0	0	1	2	.	.	.	.	.	13	14	15						SNR
	W1	**.**	**.**	**.**	**.**	**.**	**4**	**5**	**6**	**.**	**.**	**.**	**.**	**.**				**CFR**
	W2	.																LBCR#1
	W3	.																LBCR#2
	W4	.																LBCR#3
	W5	**0**	**1**	**2**	**.**	**.**	**.**	**.**	**.**	**13**	**14**	**15**						**RNR**
		.																
		.																
		.																
Status Block	W10	0	1	2	.	.	.	.	.	13	14	15						SSR
	W11	**.**	**.**	**.**	**.**	**.**	**4**	**5**	**6**	**.**	**.**	**.**	**.**	**.**				**GSR**
	W12	.																LBCR#1
	W13	.																LBCR#2
	W14	.																LBCR#3
	W15	**0**	**1**	**2**	**.**	**.**	**.**	**.**	**.**	**13**	**14**	**15**						**RIR**
		.																
		.																
		.																
		.																
		.																
		.																
Recipe Register	W100	1																1st recipe, 1st word
	W101	0																1st recipe, 2nd word
	W102	0																1st recipe, 3rd word
	W103	0																1st recipe, 4th word
	W104	3																1st recipe, 5th word
		.																
		.																
		.																

CFR bit # 4 is the Recipe Write Flag; bit # 5 is the RCPNO Change Flag; bit # 6 is the Recipe Read Flag.

GSR bit # 4 is the Recipe Write Status; bit # 5 is the RCPNO Change Status; bit # 6 is the Recipe Read Status.

3. Set up **RNR** to read recipe # N. W5 in the controller is assigned to read recipe # N from the operator terminal. For example, 1st recipe N = 1.
4. Set RCPNO Change Flag ON for about 1 second. The operator terminal's internal RCPNO and Current Recipe change to read recipe # N. Remember to set the RCPNO Change Flag OFF before re-triggering.

Status Block: When the value of RCPNO is changed, the value of RIR (W15) will also be changed. The current recipe # N can be checked on the controller. If the RCP-NO Change Flag is set ON, the GSR bit (W11 bit # 5) also sets ON. The status bit turns OFF automatically after RCPNO is changed.

For details on addressing recipe data, please see the chapter *Addressing Recipe Data - Enhanced Operator Terminals*.

Operator Terminal Data Register

Word	Recipe Data	Member
RCPW0	1	Current Recipe
RCPW**1**	**0**	
RCPW2	0	
RCPW3	0	
RCPW4	3	
RCPW**5**	**1**	**Recipe # 1**
RCPW**6**	0	
.	0	
.	0	
RCPW**9**	3	
RCPW10	0	**Recipe # 2**
RCPW**11**	**1**	
.	0	
.	0	
RCPW14	2	
.	.	.
.	.	.
.	.	.
.	.	.
RCPW20	0	**Recipe # 4**
RCPW21	0	
.	0	
.	1	
RCPW24	1	
.	.	.
.	.	.
RCPNO	1	Specified Recipe # N
.	.	.
.	.	.

5. Set the Recipe Write Flag ON. The operator terminal writes the Current Recipe to the controller. The recipe data will be saved in the designated **Recipe Register Block**. Remember to set the Recipe Write Flag OFF before re-triggering. In this example, set W1 bit # 4 ON for about 1 second. The operator terminal writes the Current Recipe to controller W100-W104.

Status Block: When the operator terminal writes a recipe, the GSR bit (W11 bit # 4) sets ON automatically. If the Recipe Write Flag is set OFF, the GSR bit also sets OFF.

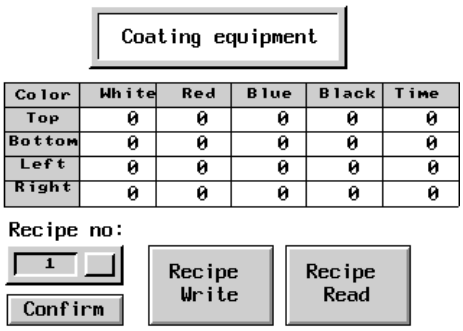
After completing these steps the controller can read one cycle of a recipe from the operator terminal. Remember to reset the flag OFF each time in order to trigger the flag.

Follow the steps above to set the RNR, RCPNO Change Flag and Recipe Read Flag values and to read a recipe from the controller to the operator terminal.

3.4 Recipe Controlled by Operator Terminal

Using objects on the operator terminal display makes it convenient to control read and write actions on recipe data, performed in the controller. Please see the chapter *Control and Status Block*.

This section uses coating equipment as an example of controlling recipe data in the controller. The following is an illustration of the coating equipment setup on the operator terminal.



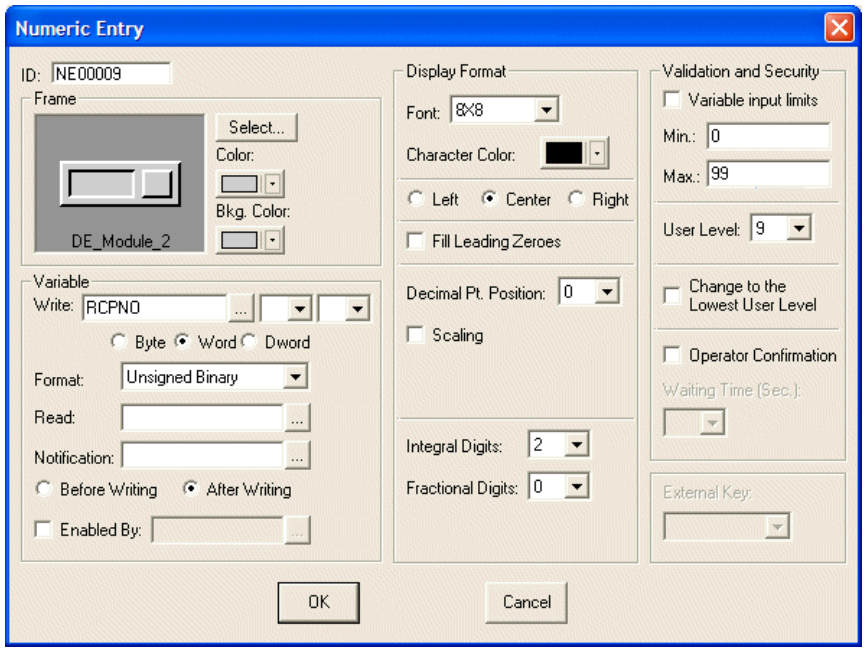
An illustration of a coating equipment screen in the operator terminal

Object Design steps:

To design an object for the user to enter as recipe N and write recipe N to controller RNR:

1. In H-Designer, select **Object/Numeric Entry** and enter the address of RNR in the **Write** box.

   Using the coating equipment as an example, the address of RNR is W5; so the controller recipe N will write to W5. It is also possible to enter RCPNO directly into a Numeric Entry object. This can only be changed locally in the operator terminal.



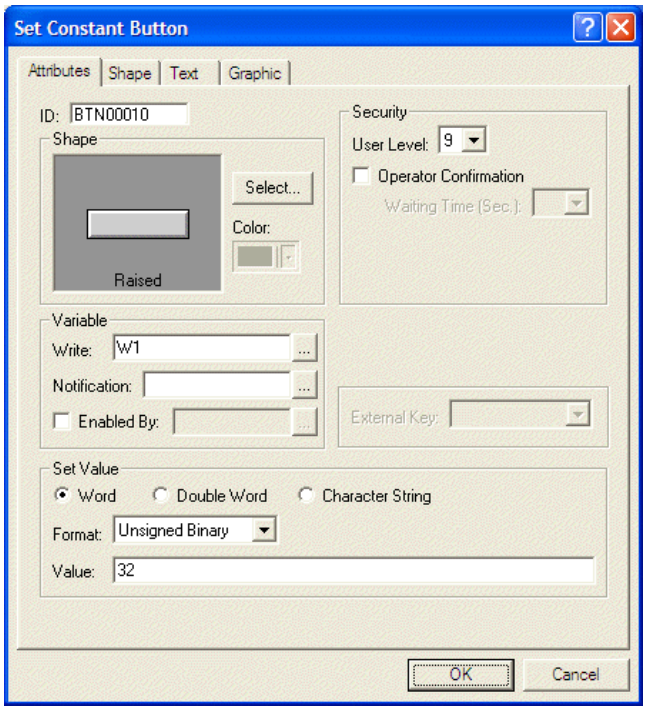
Settting up the address to write recipe N to controller RCPNO

To design an object that can confirm the designated recipe and write the commands to controller CFR bit # 5 RCPNO Change Flag; set the flag ON:

2. In H-Designer, select **Object/Push Button/Set Constant**. Enter the address of CFR in the **Write** box. Then enter the constant value in the **Value** box to set its register bit.

   Using the coating equipment as an example, the address of CFR is W1. The RCPNO Change Flag is located in CFR bit # 5.

   The constant is set to 32 (2^5 = 32), so the RCPNO Change Flag will be set ON.

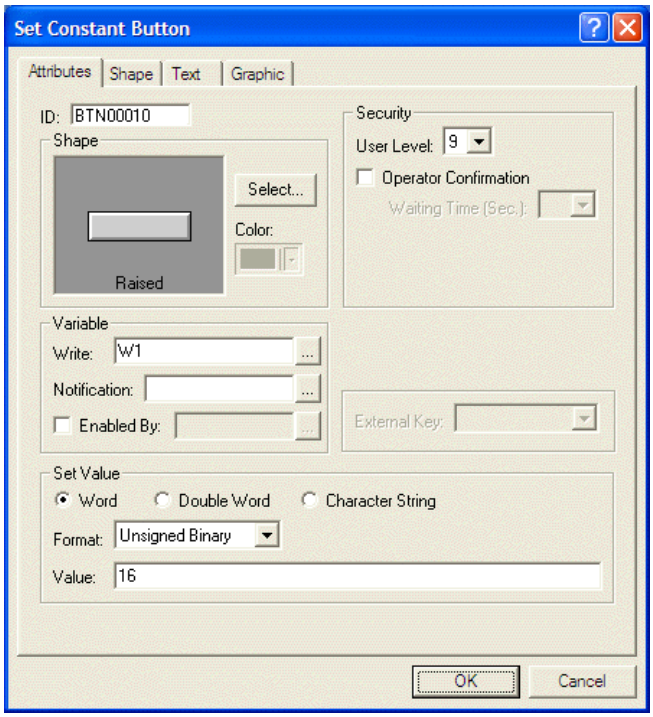


Setting RCPNO Change Flag to ON

To design an object which can set the CFR bit # 4 Recipe Write Flag ON and write the Current Recipe to the controller:

3. Using the coating equipment as an example, the address of CFR is W1.

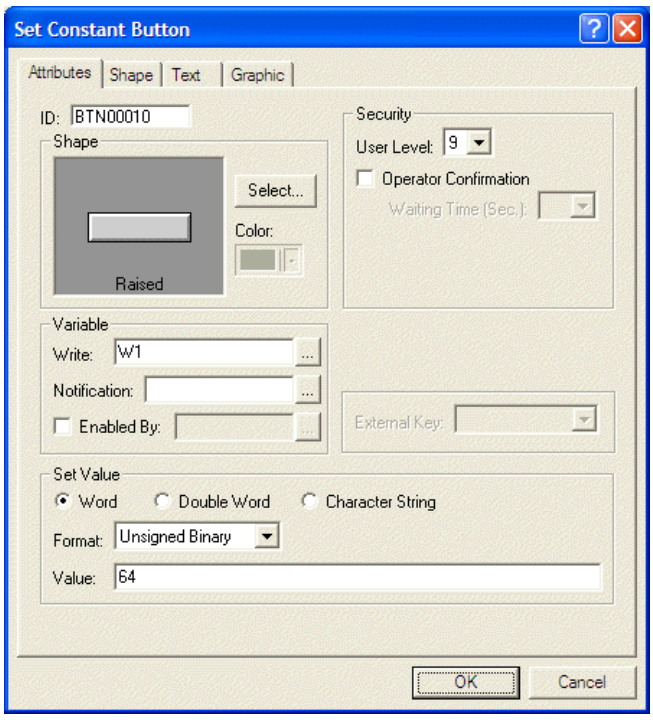
   The Recipe Write Flag is located in CFR bit # 4, so enter the value 16 (2^4 = 16). The Recipe Write Flag located in W1 bit # 4 will then be set ON.



Setting up Recipe Write Flag ON

4. If you want to read a recipe from the controller to the operator terminal, the Recipe Read Flag located in CFR bit # 6 must be set ON.

   Using the coating equipment as an example, the address of CFR is W1.
   The Recipe Flag is located in CFR bit # 6 (2^6 = 64).



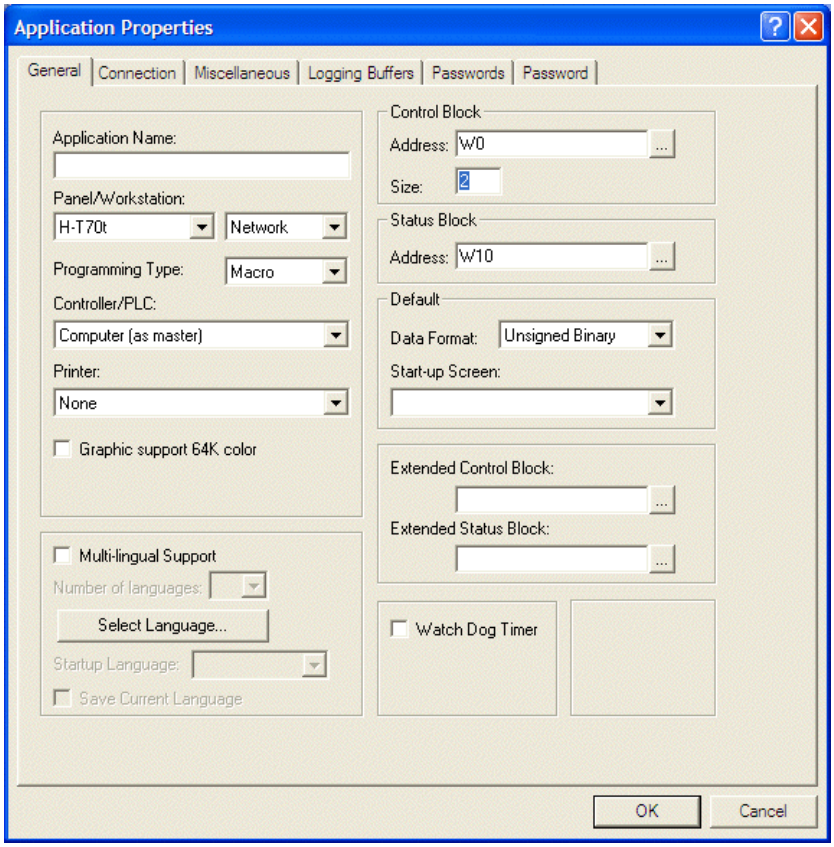
Setting up Recipe Read Flag ON

After completing these steps, you can execute the actions in the operator terminal.

4 Control and Status Block

This chapter describes the general information that you need for programming the controller to communicate with the operator terminal. Detailed information about connecting specific controllers to the operator terminal is presented in this chapter.

To set up the **Control Block Address**, **Size** and **Status Block Address**, select **Application/Workstation Setup**.



Setting up control and status block parameters

4.1 Control Block

The **Control Block** is a block of continuos registers in the controller. The most important function in the operator terminal is the control block.

The control block enables the controller to control actions in the operator terminal through the controller program. The minimum number of words used in the control block is 2. The maximum number of words used is 32. The size of the control block varies according to the functionality required (if recipe functionality is used, then the minimum length is 6 words). The members of the Control Block are shown in the following table:

Word #	Member	Example: S7-200	Example: FX2
Wn	Screen Number Register (SNR)	VW0	D0
Wn+1	Command Flag Register (CFR)	VW2	D1
Wn+2	Logging Buffer Control Register #1	VW4	D2
Wn+3	Logging Buffer Control Register #2	VW6	D3
Wn+4	Logging Buffer Control Register #3	VW8	D4
Wn+5	RCPNO Number Register (RNR)	VW10	D5
Wn+6 and above	General User Area Register (GUAR) User's application registers CBn, n must not exceed 31.	VW12=cb6 VW14=cb7 … VW18=cb9 …	D6=cb6 D7=cb7 … D9=cb9 …

For example, the starting address is W0 (the starting address can be specified; the members in the above table will shift according to the starting address). Size 10 means that the operator terminal can read data from the W0-W9 (10 words) controller registers and store data in the CB0-CB9 internal control block.

The functions of the words Wn through Wn+m (word n+m) in the control block will be discussed in the following sections.

4.1.1 Screen Number Register

A controller can request an operator terminal to display a specific screen by setting its **Screen Number Register (SNR)** to the number of that screen.

SNR (Wn) enables the controller to control the operator terminal screen or print the screen. For example, a controller can request a operator terminal to display a specific screen by setting its SNR to the number of that screen.

The operator terminal cannot reset the SNR (Wn) to 0 automatically. However, it does reset the SNR (Wn) to 0 before changing a screen. If the screen specified by the SNR does not exist, the operator terminal does nothing but resetting the SNR (Wn=0).

The value in the SNR can be BCD or binary.

Example

A controller can control the switch of screen 001

The value of the SNR data register (Wn) and the functions (bit 0-bit 05) are as follows:

Wn 16-bit #(00-15)	Function
Bit 9-..bit 0	The first 10 bits store the screen number to be changed to.
Bit 10	Reserved
Bit 13=off, 12=off, 11=off	No language was selected
Bit 13=off, 12=off, 11=on	Language 1
Bit 13=off, 12=on, 11=off	Language 2
Bit 13=off, 12=on, 11=on	Language 3
Bit 13=on, 12=off, 11=off	Language 4
Bit 13=on, 12=off, 11=on	Language 5
Bit 13=on, 12=on, 11=off	Reserved
Bit 13=on, 12=on, 11=on	Reserved
Bit 14	Backlight turned off when set to 1
Bit 15	Backlight turned on when set to 1

The register (bit 0-9) is used to control the screen change and the other bits (bit 10-15) are not related. It is not necessary to control the backlight or language when changing the screen. It is also not necessary to assign the screen number when setting up the backlight or selecting a language.

4.1.2 Command Flag Register

The functions of the bits in the CFR are summarized in the following table:

Wn+1 16-bit #(00-15)	Function
Bit 0	Alarm History Buffer Clear Flag
1	Alarm Frequency Buffer Clear Flag
2	Print Change Paper Flag/Form Feed Flag
3	Hardcopy Flag
4	Recipe Write Flag - Data sent from operator terminal to controller
5	RCPNO Change Flag
6	Recipe Read Flag - Data sent from controller to operator terminal
7	Buzzer action control
8	Clear Flag #1
9	Clear Flag #2
10	Clear Flag #3
11	Clear Flag #4
12	Trigger Flag #1
13	Trigger Flag #2
14	Trigger Flag #3
15	Trigger Flag #4

The bits of the CFR (Wn+1) are described in more detail below.

Bit 0: Alarm History Buffer Clear Flag

This controller bit is used to clear the data in the alarm history buffer.

If bit 1 is set to clear the data in the alarm history buffer, the operator terminal will clear its data when bit 0 is set to 1.

The controller requires that the bit be reset if the operator terminal is re-assigned to clear the data and it needs enough time for operator terminal detection. The “handshake” function can be used to reset the bit as well. For more information about the “handshake” function, please see the section *General Status Register*.

Bit 1: Alarm Frequency Buffer Clear Flag

This controller bit is used to clear the Alarm Frequency Buffer.

If bit 1 represents to clear the data of alarm frequency buffer, the operator terminal will clear its data when bit 0 sets to bit 1.

The controller requires that the bit be reset if the operator terminal is re-assigned to clear the data and it needs enough time for operator terminal detection. The “handshake” function can be used to reset the bit as well. For more information on the “handshake” function, please see the section *General Status Register*.

Bit 2: Print Change Paper Flag

This controller bit is used to control form feed on the printer connected to the operator terminal.

Set the bit ON, and the printer will change paper.

The controller requires that the bit be reset if the operator terminal is re-assigned to form feed and it needs enough time for operator terminal detection.

Bit 3: Hard Copy Flag

This controller bit is used to control the hard copy function for the printer connected to the operator terminal.

Set the bit to ON and the printer will print the current screen.

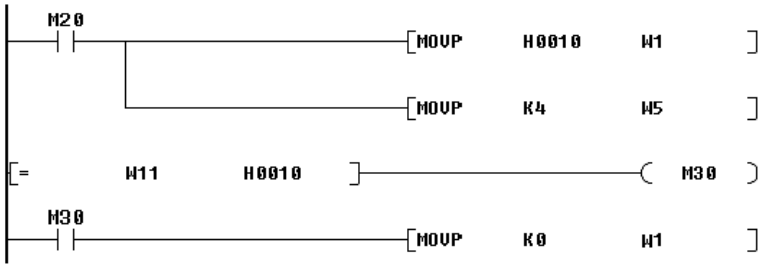
The controller requires that the bit be reset if the operator terminal is re-assigned to print a hard copy and it needs enough time for operator terminal detection.

Bit 4: Recipe Write Flag - from operator terminal to controller

This bit is used to write the recipe from RAM to the controller. This is supported only by operator terminals with a recipe function.

Set the RNR (Wn+5) to write the recipe, set the bit to ON, and the recipe will be written to the controller.

The controller requires that the bit be reset if the operator terminal is re-assigned to write another recipe and it needs enough time for operator terminal detection.



Controller M20 writes the data from terminal 4th recipe to the controller. W11 bit 4 is Recipe Write status bit.

Bit 5: RCPNO Change Flag

This controller bit is used to change the content value of RCPNO. RCPNO is an internal operator terminal register used to control the recipe data. This is only supported by operator terminals with a recipe function.

Set the RNR (Wn+5) to write the recipe, set the bit to ON, RCPNO can be modified.

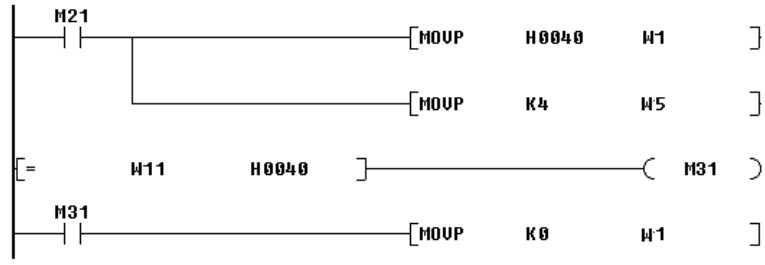
The controller requires that the bit be reset if the operator terminal is re-assigned to modify RCPNO and it needs enough time for operator terminal detection.

Bit 6: Recipe Read Flag - from controller to operator terminal

This controller bit is used to read the recipe data from the controller to the operator terminal and save it in the RAM block.

Set the RNR (Dn+5) to the recipe number to be updated. Then set the bit to ON and the operator terminal will update the corresponding recipe.

The controller requires that the bit be reset if the operator terminal is re-assigned to update and it needs enough time for operator terminal detection.



Controller M21 reads the data from the controller to the 4th recipe. W11 bit 6 is Recipe Read status bit.

Bit 7: Buzzer Flag

This controller bit is used to control the operator terminal's buzzer.

Set the bit to ON (about 1 sec.) to start the buzzer.

The controller requires that the bit be reset if the operator terminal is re-assigned to start the buzzer.

Bit 8-11: Clear Flag #1 - #4

This controller bit is used to clear curves in the operator terminal. There are four clear flags, and you can set the corresponding signal to clear the desired curve.

Set the bit to ON/OFF once to clear the values of a trend graph or X-Y chart.

The controller requires that the bit be reset if the operator terminal is re-assigned to modify RCPNO and it needs enough time for operator terminal detection.

Bit 12-15: Trigger Flag #1 - #4

This controller bit is used to sample the trend graph data. There are four trigger flags in all.

Once the controller bit is set to ON/OFF, the operator terminal will read the continuous data and covert it into a continuous curve which is displayed as a trend graph or X-Y chart objects.

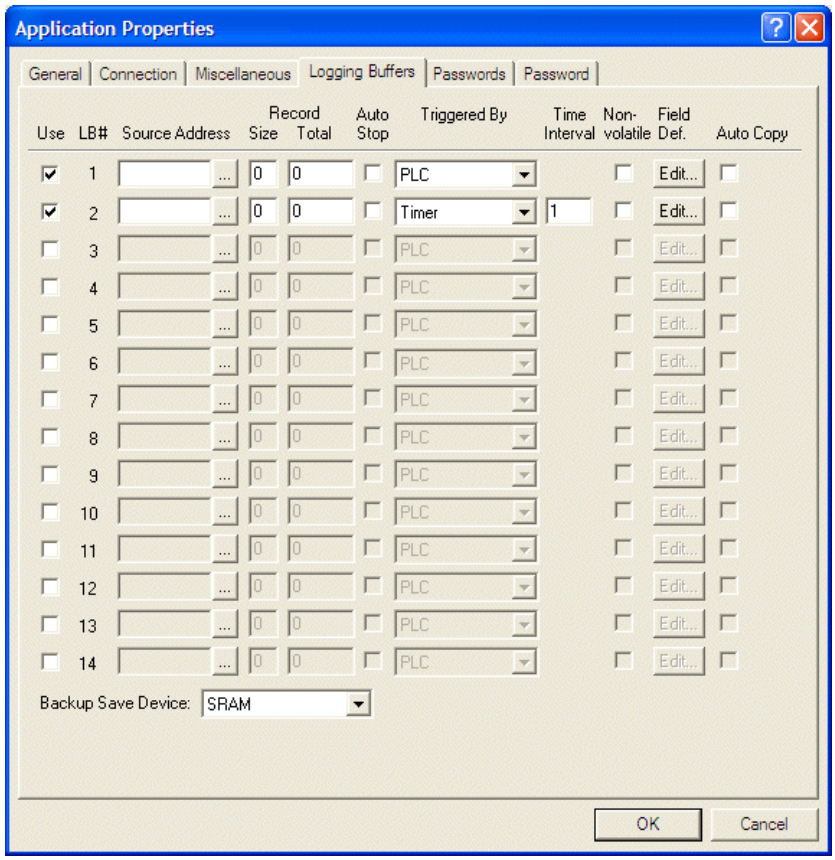
The controller requires that the bit be reset if the operator terminal is re-assigned to sample the data and it needs enough time for operator terminal detection.

4.1.3 Logging Buffer Control Registers: LBCRs

The other type of trend graph in the operator terminal is called the **Historical Trend Graph**. The operator terminal reads the data from the corresponding logging buffer according to the specific signal. The logging buffer is used to save the sample data in battery backup RAM. Remember to specify the logging buffer to read from and its size.

In LBCRs, **Trigger Bits** are used to request logging buffers to sample the data from controllers. **Clear Bits** are used to clear logging buffers and **Size Bits** are used to determine the size of the data to be read. Consequently, you can use the LBCRs to clear logging buffers or to request logging buffers to sample the data from the controllers.

There are twelve logging buffers here, and operator terminals can be set to sample the data automatically at fixed periods, or to sample or clear the historical trend graph controlled by the controllers.



Setting up logging buffers

Note that LBCR1 controls logging buffer #1 through #4. LBCR2 controls logging buffer #5 through #8. LBCR3 controls logging buffer #9 through #12.

The positions of the trigger bit, clear bit and size bit for each logging buffer are shown in the following table:

Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VW4 LBCR1	0	SB4	CB4	TB4	0	SB3	CB3	TB3	0	SB2	CB2	TB2	0	SB1	CB1	TB1
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VW6 LBCR2	0	SB8	CB8	TB8	0	SB7	CB7	TB7	0	SB6	CB6	TB6	0	SB5	CB5	TB5
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VW8 LBCR3	0	SB12	CB12	TB12	0	SB11	CB11	TB11	0	SB10	CB10	TB10	0	SB9	CB9	TB9

SB: Size Bit; CB: Clear Bit; TB: Trigger Bit:

LBCR1	Buffer #4	Buffer #3	Buffer #2	Buffer #1
LBCR2	Buffer #8	Buffer #7	Buffer #6	Buffer #5
LBCR3	Buffer #12	Buffer #11	Buffer #10	Buffer #9

Trigger Bit #1- #12: Sampling Control

The operator terminal can not only sample the historical trend graph at a particular time interval, it can also sample the historical trend graph under the control of the trigger bit in the controller. When the trigger bit (TB #1 - TB #12) is set to ON/OFF (about 1 sec.), the operator terminal will execute its sampling function. Remember to set it to OFF before re-triggering.

Clear Bit #1- #12: Clear Control

By triggering the trend graph clear bit (CB #1 - CB #12) ON/OFF once (about 1 sec.), the trend graph can be erased. The flag needs to be set to OFF if it is to be triggered again.

Size Bit #1-#12: Multiple Sampling Control

By triggering the size bit (SB #1 - SB #12) to ON, the operator terminal can sample single or multiple data values. When the trigger bit (TB #1 - TB #12) is set to ON/OFF (about 1 sec.), the operator terminal will execute its sampling function.

Logging Buffer

In setting up the logging buffer, the first step is to specify the **Source Address**, which is used to specify the controller address to read the data from.

After setting up the source address, the size bit is set to OFF and the trigger bit is changed from 0 to 1. Then the logging buffer will read a data value from the controller.

To force the logging buffer to read multiple data values from the controller, set the size of the source address to read from. Then set the size bit to ON and change the trigger bit from 0 to 1. Note that the size cannot exceed 1,022 words.

Change the clear bit from 0 to 1 to clear the logging buffer.

To force logging buffer recording, the controller must reset the trigger and clear bits. Sufficient time must be provided for operator terminal detection

Example: FX2 controller

Assumptions:

1. Control block starts from D0 with a size = 6
2. Source address of logging buffer #11 is D200
3. The record size of logging buffer #11 is 3 words

To request logging buffer #11 to read only one data record from the controller, first write the data to be read in D200-D202. Set D4's size bit (10) to OFF and change its trigger bit (8) from 0 to 1. The operator terminal reads D200-D202 into logging buffer #11 after it detects the trigger bit (8) of LBCR3 has changed from 0 to 1.

To request logging buffer #11 to read 50 data records from the controller, set D200 to 150 (=50 x 3). Write the data to be read in D201-D350. Set D4's size bit (10) to ON and change its trigger bit (8) from 0 to 1. The operator terminal first reads D200-D202 to get the actual size of the data to be read after it detects the trigger bit (8) of LBCR3 has changed from 0 to 1. Then the operator terminal reads D200-D350 and stores the data in battery backup RAM.

To request logging buffer #11 to clear the data records, change D4's clear bit from 0 to 1.

4.1.4 RCPNO Number Register: RNR

RCPNO is an internal register of the operator terminal that specifies the current recipe number. To change the RCPNO, the controller first sets the RNR to the recipe number and then sets the RCPNO write flag or recipe read flag.

To change RCPNO by the controller, the controller has to set RNR to the recipe number and set the RCPNO change flag, which is the CFR 5 bit. If the RNR is 0 or greater than the maximum recipe number, the operator terminal will ignore the request.

To request the operator terminal to change RCPNO, the controller must reset the RCPNO change flag, or use the RCPNO change status flag, which is the GSR 5 bit.

Be sure to set this flag long enough for the operator terminal to detect it.

4.1.5 General User Area Register

For high-speed display, the operator terminal only reads data from the internal register (cannot write to it) when editing in H-Designer. (The maximum size is 32; the size depends on the length of the control block). The format is shown in the following table:

Format	Description
CB n	n represents the word data of the nth register, where n is a decimal number; n ≥ 0 but smaller than the specified size.
CBn b	n.b represents the bit data corresponding to the nth word register, where b is in hexadecimal nth b = 0-f.

For example, if the address of the control block is D0, the **Numeric Display** object can be selected to display the recipe number register by configuring it to display CB5 instead of displaying D5.

The internal buffer of the control block is read-only. This means, for example, that you can configure a **Numeric Display** object to show the value of CB2, but you cannot configure the object to allow the operator to change the value of CB2.

For example, you want to achieve the effect of the RNR numeric display object using H-Designer. You can specify that D5 be read from (writeable) or that CB5 be read from (non-writeable).

4.1.6 Determine the Control Block Size

As every application needs a **Screen Number Register** (**SNR**) and a **Control Flag Register** (**CFR**), you can refer to the following rules to determine the size of the control block:

1. If the operator terminal reads/writes a recipe from/to a controller, the minimum size is six.
2. If Item 1. is not true and the operator terminal uses LBCR3 to control logging buffer #9 - 12, the minimum size is five.
3. If either Item 1. or Item 2. is not true and the operator terminal uses LBCR2 to control logging buffer #5-8, the minimum size is four.
4. If none of Item 1. through Item 3. is true and the operator terminal uses LBCR1 to control logging buffer #1-4, the minimum size is three.
5. If none of the above is true, the minimum size is two.
6. The size of the control block is the minimum size plus the size of the user area.

4.2 Status Block

The Status Block is a block of contiguous registers in your controller that display status information from the operator terminal. For example, you can get the current screen number from the first word of the Status Block. The members of Status Block are shown in the following table:

Word #	Member	Example: S7-200	Example: FX2
Wn	Screen Status Register (SNR)	VW20	D10
Wn+1	General Status Register (GSR)	VW22	D11
Wn+2	Logging Buffer Status Register #1 (LBSR1)	VW24	D12
Wn+3	Logging Buffer Status Register #2 (LBSR2)	VW26	D13
Wn+4	Logging Buffer Status Register #3 (LBSR3)	VW28	D14
Wn+5	RCPNO Image Register (RIR)	VW30	D15
Wn+6	Reserved	VW32	D16

For example, if the status block is W10 and the size is 6 words, the operator terminal will write the status data of the current screen to W10-W15.

4.2.1 Screen Status Register

When a screen is changed in the operator terminal, the controller sets its **Screen Status Register** (**SSR**) to the number of the new screen. Consequently, the controller can identify the current screen by reading the SSR.

The value of the SSR can be in BCD or binary format.

4.2.2 General Status Register

The components of the **General Status Register** (**GSR**) are shown in the following table (bit 0 - bit 15):

Wn11 16-bit #(00-15)	Function
Bit 0	Password Level Status (not available for applications configured to monitor alarms)
1	Password Level Status (not available for applications configured to monitor alarms)
Bit 0	Alarm History Buffer Clear Status
1	Alarm Frequency Buffer Clear Status
2	Form Feed Status
3	Hardcopy Status
4	Recipe Write Status
5	RCPNO Change Status
6	Recipe Read Status
7	Battery Status
8	Clear Status Flag #1
9	Clear Status Flag #2
10	Clear Status Flag #3
11	Clear Status Flag #4
12	Trigger Status Flag #1
13	Trigger Status Flag #2
14	Trigger Status Flag #3
15	Trigger Status Flag #4

Bit 0, 1: Password Level Status (not available for applications configured to monitor alarms)

Once connected to the operator terminal, the password level status bit 0 - bit 3 represent the current user level.

Level 0 ==>Bit 0 = off, Bit 1 = off

Level 1 ==>Bit 0 = on, Bit 1 = off

Level 2 ==>Bit 0 = off, Bit 1 = on

Level 3 ==>Bit 0 = on, Bit 1 = on

Level 4 - 9 ==>Bit 0 = on, Bit 1 = on

Bit 0: Alarm History Buffer Clear Status

The operator terminal will turn ON this status bit when it detects the alarm history buffer clear flag being turned ON. When the operator terminal finishes clearing the alarm history buffer, it will turn OFF this status bit.

Bit 1: Alarm Frequency Buffer Clear Status

The operator terminal will turn ON this status bit when it detects the alarm frequency buffer clear flag being turned ON. When the operator terminal finishes clearing the alarm fre-quency buffer, it will turn OFF this status bit.

Bit 2: Form Feed Status

The operator terminal will turn ON this status bit when it detects the form feed flag being turned ON. When the operator terminal finishes sending the form feed character to a printer, it will turn OFF this status bit.

Bit 3: Hardcopy Status

The operator terminal will turn ON this status bit when it detects the hardcopy flag being turned ON. When the operator terminal finishes printing the current screen, it will turn OFF this status bit.

Bit 4: Recipe Write Status

The operator terminal will turn ON this status bit when it finishes sending a recipe from the operator terminal's RAM block to the controller. The operator terminal will turn OFF this status bit when it detects the recipe write flag being turned OFF. This bit can be used as a handshake signal to switch the recipe write flag.

Note:
This function is only supported by operator terminals with a recipe function.

Bit 5: RCPNO Change Status

The operator terminal will turn ON this status bit when it detects the RCPNO change flag being turned ON. When the operator terminal finishes changing the RCPNO, it will turn OFF this status bit.

Bit 6: Recipe Read Status

The operator terminal will turn ON this status bit when it finishes reading a recipe from the controller. The operator terminal will turn OFF this status bit when it detects the recipe read flag being turned OFF. You can use this bit as a handshake signal to switch the recipe read flag..

Note:
This function is only supported by operator terminals with a recipe function.

Bit 7: Battery Status

The operator terminal will turn ON the battery status bit if it detects a low battery before running an application.

Bit 8-11: Clear Status Flag #1 - #4
The operator terminal will turn ON one of the clear status bits when it finishes the clearing task requested by the corresponding clear flag controlled by the controller. The operator terminal will turn OFF the same status bit when it detects the corresponding clear flag being turned OFF. You can use clear status bits as handshake signals to switch the clear flags.

Bit 12-15: Trigger Status Flag #1 - #4
The operator terminal will turn ON one of the trigger status bits when it finishes the task triggered by the corresponding trigger flag. The operator terminal will turn OFF the same status bit as it detects the corresponding trigger flag being turned OFF. You can use trigger status bits as handshake signals to switch the trigger flags.

4.2.3 Logging Buffer Status Registers (LBSRs)

LBSR1 saves the status of logging buffer #1 - #4. LBSR2 saves the status of logging buffer # 5 - #8. LBSR3 saves the status of logging buffer #9 - #12.

The status bit's position for each of the logging buffers is shown in the table below:

Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VW4 LBSR1	AB4	FB4	CB4	TB4	FB3	AB3	CB3	TB3	AB2	FB2	CB2	TB2	AB1	FB1	CB1	TB1
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VW6 LBSR2	AB8	FB8	CB8	TB8	AB7	FB7	CB7	TB7	AB6	FB6	CB6	TB6	AB5	FB5	CB5	TB5
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VW8 LBSR3	AB12	FB12	CB12	TB12	AB11	FB11	CB11	TB11	AB10	FB10	CB10	TB10	AB9	FB9	CB9	TB9

AB: Almost Full Bit - indicates that the buffer is 90% or more full.
FB: Full Bit - indicates that the buffer is full.
CB: Clear Status Bit indicates the clear command was received.
TB: Trigger Status Bit indicates the trigger command was received.

LBSR1	Buffer #4	Buffer #3	Buffer #2	Buffer #1
LBSR2	Buffer #8	Buffer #7	Buffer #6	Buffer #5
LBSR3	Buffer #12	Buffer #11	Buffer #10	Buffer #9

The operator terminal will turn ON one of the trigger status bits when it finishes collecting one data record for the logging buffer. The operator terminal will turn OFF the same status bit as it detects the corresponding trigger flag being turned OFF. You can use the trigger status bits as handshake signals to switch the trigger flag.

4.2.4 RCPNO Image Register

The operator terminal sets the RCPNO Image Register (RIR) to the new value of the RCPNO as this internal register is changed by the user or a controller. Consequently, the controller is able to identify the current value of the RCPNO. The operator terminal reports the value of the RCPNO to the controller by writing the value to the RCPNO Image Register. The RCPNO Image Register is word #5 of the status block and you can keep track of the current recipe with this register.

4.3 Recipe Register Block

The recipe block is located in the controller register. To make the operator terminal read/write the recipe data from/to the controller, a recipe block needs to be defined for the application. Please see the chapter *Recipes* for complete details.

The maximum recipe memory block is 524,288 16-bit (word) for the operator terminal with recipe function. For the applied operator terminal models, please see *Appendix A - H-Designer Features and Operator Terminal Models* for complete details.

4.3.1 Recipe Register Number - Enhanced Operator Terminals

H-Designer provides internal recipe register number for use in the operator terminal application using the formats shown below:

Format	Description
RCPNO	Recipe Register Number (1-N) RCPNO is an internal register of the operator terminal that specifies the current recipe number; No≥1.
RCPWnnnnn	Recipe Register #nnnnn is current recipe where nnnnn is a decimal number and n≥0.
RCPWnnnnn.b	Recipe Register Bit nnnnn is decimal number, n≥0; b is a hexadecimal number, b=0-F.

RCPNO is an internal register of the operator terminal used to display the specified recipe on the screen. The operator terminal changes the RCPNO number to display its corresponding recipe data.

There are two methods to change the RCPNO number:

One way is for the user to change the RCPNO number directly through the numeric entry object.

The other way is for the controller to change the RCPNO constant. To change the RCPNO constant, the user must write the specified number N to RCPNO number register Dn+5, then set the RCPNO change flag Dn+1 bit 5 as ON (about 1 sec.). The operator terminal will change the RCPNO constant to N and display the recipe data RCPW0 - RCPWm corresponding to the Nth recipe.

4.3.2 Addressing Recipe Data - Enhanced Operator Terminals

Suppose that the number of recipe N=20, a recipe size m=100 words.

To edit an address, you need to set up the current recipe N =RCPNO. The operator terminal will display the corresponding recipe data.

1. Enter the recipe number N in RCPNO or change RCPNO using the controller. The operator terminal will display the corresponding recipe data.

   For example, if RCONO N=5, RCPW0-RCPW99 displays the data corresponding to the fifth recipe; if RCPNO N=7, RCPW0-RCPW99 displays the data corresponding to the seventh recipe.

2. Another way to edit the corresponding recipe register data is to use the absolute address.

   Suppose that an address is greater than RCPW100, the corresponding recipe address will display the Nth recipe data.

   RCPW100-RCPW199 represents the first recipe data.

   RCPW200-RCPW299 represents the second recipe data.

   RCPW2000-RCPW2099 represents the twentieth recipe data.

   Addresses greater than RCPW2099 are invalid.

   So, RCP234 represents the second recipe data, 35 words and RPCW 34 in RCPNO = 2.

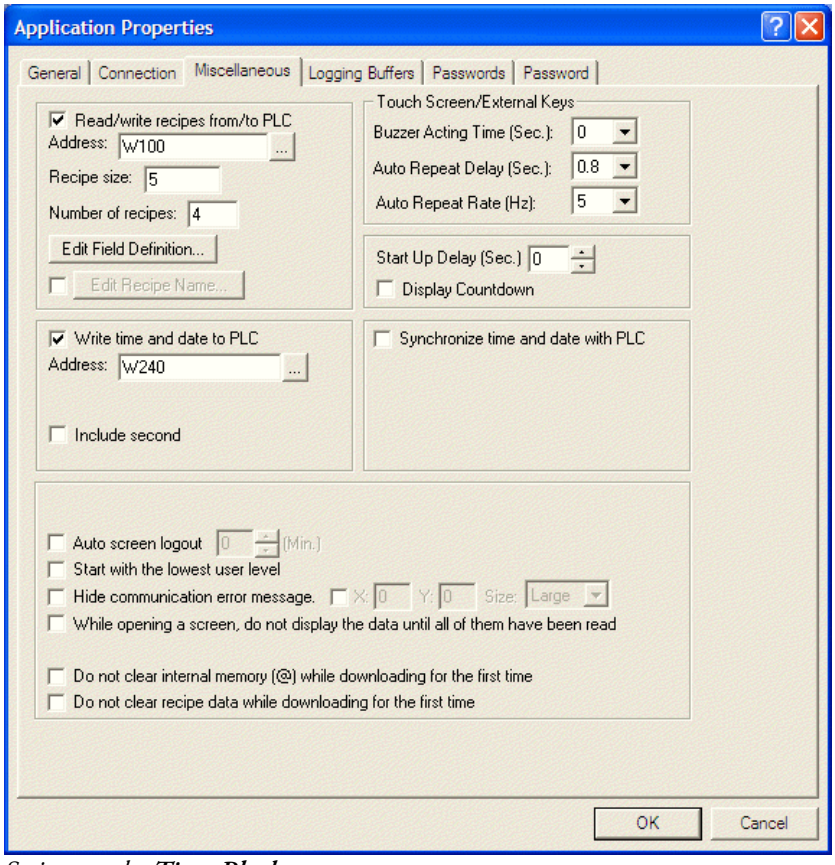
4.4 Time Block

4.4.1 The Operator Terminal Writes to the Controller

To make the operator terminal write the current time and date to the controller, the **Time Block** has to be defined for the application. The time block is a block of three words in the controller and its format is BCD. The operator terminal updates the time block every minute with the time data. The format of time block is shown in the following:

Low byte of word 0 (07-00)	Minute BCD 00-59
High byte of word 0 (15-08)	Hour BCD 00-23
Low byte of word 1 (07-00)	Day BCD 00-31
High byte of word 1 (15-08)	Month BCD 01-12
Low byte of word 2 (07-00)	00-99
High byte of word 2 (15-08)	Day of week 0 = Sunday 1 = Monday 2 = Tuesday 3 = Wednesday 4 = Thursday 5 = Friday 6 = Saturday

The steps to set up the time block follow. Select **Application/Workstation Setup** in H-Designer and you can set up the time block on the **Miscellaneous** tab. The starting address is W240 and the size is 3 words, so the data will be saved in the W240, W241, and W242 16-bit registers. The operator terminal updates the time block every minute with the time data.



*Seting up the **Time Block***

4.4.2 The Controller Writes to the Operator Terminal

The operator terminal can read time and date from the internal real time clock of the controller. Then the operator terminal can modify the corresponding data for the time/date/week read from the real time clock and display the content in the operator terminal. The operator terminal updates the time block every minute with the time data.

4.5 Read Cycle

The operator terminal does the following steps to accomplish one read cycle and it will repeat these steps continuously. You need to know this read cycle in order to configure an operator terminal to communicate with the controller efficiently.

Steps of the cycle:

1. Reads control block of the controller.
2. Reads specified register blocks for the current screen.
3. Reads specified On/Off blocks for the current screen.
4. Reads specified alarm register regularly (3-10 sec.).
5. Reads a number of controller locations which: (1) are shown on the current screen; and (2) do not appear in the current screen's register blocks or On/Off blocks and have not been read recently.

   The number of controller locations to be read in this step is specified by the "number of individual reads" per read cycle of the current screen.

This read cycle is repeated continuously from Step 1. to Step 5.

4.6 Extended Control Block

Extended Control Block (ECB) is a 5-word register block in the controller. The first word of the ECB is the command word and is named CW. The other 4 words of the ECB are the parameter words and are named PW1, PW2, PW3 and PW4. To make the terminal do a job, the controller stores parameter values in the parameter words first, and then sets the command word to the job number. The controller is responsible to reset the command word to zero after the job is done, or the job is terminated due to errors in the parameters. The terminal will not be ready to perform another job until it sees the command word becomes zero. The controller can get the status of the current job from the Extended Status Block (ESB).

Job number: 64

Job description: Saving records of unsaved logging buffer to a file on a Compact Flash memory card or USB memory stick.

Job parameters:

Parameter word	Description
PW1	Logging buffer number
PW2	Not used
PW3	Not used
PW4	Not used

Job number: 65

Job description: Saving unsaved alarm history buffer to a file on a Compact Flash memory card or USB memory stick.

Job parameters:

Parameter word	Description
PW1	Not used
PW2	Not used
PW3	Not used
PW4	Not used

Job number: 66

Job description: Saving recipe data to a file on a Compact Flash memory card or USB memory stick.

Job parameters:

Parameter word	Description
PW1	File number
PW2	Device: When 1, save to USB memory stick When 0, save to Compact Flash memory card
PW3	Not used
PW4	Not used

Job number: 67

Job description: Read recipe data from a file on a Compact Flash memory card or USB memory stick.

Job parameters:

Parameter word	Description
PW1	File number
PW2	Device: When 1, read from USB memory stick When 0, read from Compact Flash memory card
PW3	Not used
PW4	Not used

Job number: 128

Job description: Write recipe data to flash ROM.

Job parameters:

Parameter word	Description
PW1	Not used
PW2	Not used
PW3	Not used
PW4	Not used

Job number: 129

Job description: Read recipe data from flash ROM.

Job parameters:

Parameter word	Description
PW1	Not used
PW2	Not used
PW3	Not used
PW4	Not used

Job number: 130

Job description: Write alarm data to flash ROM.

Job parameters:

Parameter word	Description
PW1	Not used
PW2	Not used
PW3	Not used
PW4	Not used

Job number: 131

Job description: Read alarm data from flash ROM.

Job parameters:

Parameter word	Description
PW1	Not used
PW2	Not used
PW3	Not used
PW4	Not used

To allow performing more than one task involving USB/CF, four new commands are added.

Job number: 0x0001

Job description: Saving records of unsaved logging buffer to a file on a Compact Flash memory card or USB memory stick.

Job parameters:

Parameter word	Description
PW1	Logging buffer number
PW2	Not used
PW3	Not used
PW4	Not used

Job number: 0x0002

Job description: Saving unsaved alarm history buffer to a file on a Compact Flash memory card or USB memory stick.

Job parameters:

Parameter word	Description
PW1	Not used
PW2	Not used
PW3	Not used
PW4	Not used

Job number: 0x0004

Job description: Saving recipe data to a file on a Compact Flash memory card or USB memory stick.

Job parameters:

Parameter word	Description
PW1	File number
PW2	Device: When 1, save to USB memory stick When 0, save to Compact Flash memory card
PW3	Not used
PW4	Not used

Job number: 0x0008

Job description: Read recipe data from a file on a Compact Flash memory card or USB memory stick.

Job parameters:

Parameter word	Description
PW1	File number
PW2	Device: When 1, read from USB memory stick When 0, read from Compact Flash memory card
PW3	Not used
PW4	Not used

If job number = 0x0003, both logging buffer and alarm history buffer will be saved.

Text Mode Printing Control:

PW3

Bit	Description
0	Immediately Printing Flag 0 to 1: Action Accept, after printing, you have to set to 0 yourself
1	Suspend cache Alarm Buffer Flag: 0: False, continue cache 1: True, suspend cache
2-15	Reserved

4.7 Extended Status Block

Extended Status Block (ESB) is a 5-word register block in the controller. The five status words are named SW0, SW1, SW2, SW3, and SW4.

The terminal updates the ESB every second.

SW0

Bit	Description
0	0: Auto Saving Logging Buffer Okay 1: Auto Saving Logging Buffer Error

Most of the time, it will be set to 0. It will only be changed when there is a logging buffer saving in progress. If the saving is unsuccessful, it will be set to be 1. If it is successful, it will be remain 0.

Examples of unsuccessful saving:

1. No Compact Flash memory card or USB memory stick
2. Not enough space on Compact Flash memory card or USB memory stick
3. Others that make the saving unsuccessful

0 to 1: Saving Unsuccessful

1 to 0: Compact Flash memory card or USB memory stick is inserted;
Saving Successful

Bit	Description
1	0: Auto Saving Alarm Okay 1: Auto Saving Alarm Error

Most of the time, it will be set to 0. It will only be changed when there is a alarm saving in progress. When the saving is unsuccessful, it will be set to be 1. If it is successful, it will be remain 0.

Examples of unsuccessful saving:

1. No Compact Flash memory card or USB memory stick
2. Not enough space on Compact Flash memory card or USB memory stick
3. Others that make the saving unsuccessful

0 to 1: Saving Unsuccessful

1 to 0: Compact Flash memory card or USB memory stick is inserted;
Saving Successful

Bit	Description
2	Back Light Status 0: off 1: on

Bit	Description
3-5	Current Language (001-101)

Bit	Description
6	Printing Screen Hard Copy Status 0: Printer Idle 1: Printing

Bit	Description
7	Printing Text Mode Status 0: Printer Idle 1: Printer Action

Bit	Description
11-8	Current User Level (0000-1001)

Bit	Description
12	0: Auto Saving Okay 1: Auto Saving Error

Bit	Description
14-13	Job Processor (00: Ready, 01: Busy, 10: Complete, 11: Error) 10: Saving or Reading Recipe, Alarm or Logging through Extended Control Block - Okay 11: Saving or Reading Recipe, Alarm or Logging Buffer when using Extended Control Block - Error

Bit	Description
15	0: No Action when Saving through Extended Control Block 1: Action

SW1

Bit	Description
1-0	01:Error when Saving Logging Buffer to Compact Flash memory card or USB memory stick by using Extended Control Block 10: OK when Saving Logging Buffer to Compact Flash memory card or USB memory stick by using Extended Control Block

Bit	Description
4-5	01:Error when Saving Alarm History Buffer to Compact Flash memory card or USB memory stick by using Extended Control Block 10: OK when Saving Alarm History Buffer to Compact Flash memory card or USB memory stick by using Extended Control Block

Bit	Description
8-9	01:Error when Reading Recipe from Compact Flash memory card or USB memory stick by using Extended Control Block 10: OK when Reading Recipe from Compact Flash memory card or USB memory stick by using Extended Control Block

Bit	Description
12-13	01:Error when Saving Recipe to Compact Flash memory card or USB memory stick by using Extended Control Block 10: OK when Saving Recipe to Compact Flash memory card or USB memory stick by using Extended Control Block

SW2

Bit	Description
15-0	Free Space in Compact Flash memory card

SW3

Bit	Description
15-0	Free Space in USB memory stick

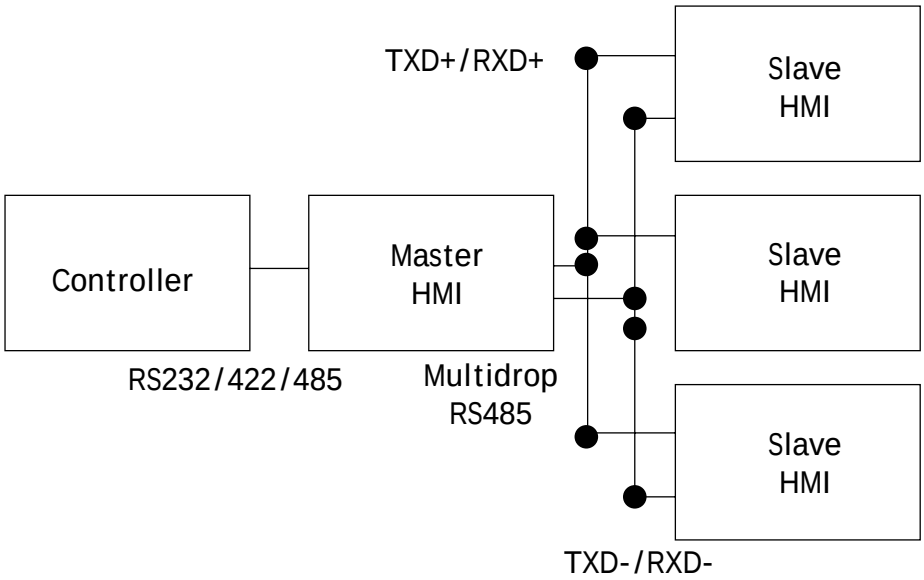
SW4

Bit	Description
7-0	Backlight Brightness Percentage (not supported for all terminal models)
15-8	Reserved

5 Multi-Link: Normal Connection Port

The **Multi-Link** function provides an economical and convenient way to link several operator terminals and communicate with a single controller connection port. One operator terminal is master and the others are slaves. The master is the only operator terminal that is physically connected to a controller and this operator terminal is responsible for data exchange between the controller and the slaves. Each of the slave operator terminals must be assigned a unique address so that the master operator terminal is able to identify which slave to send the data to.

The following illustration shows the setup for four operator terminals with one controller. Note that the RS485 cable must be used for the connection between the master and the slaves and each of the slaves must be assigned a unique address.



A Multi-Link network

The cable and the connection between the master and the controller is the same as for the normal 1-to-1 application. The RS485 cable must be used for connecting the master and the slaves. Additionally, each of the slaves must be assigned a unique address. All the controller models in H-Designer support this function.

Multi-link can also be connected through Ethernet. Please see chapter *Ethernet Communication* for more details.

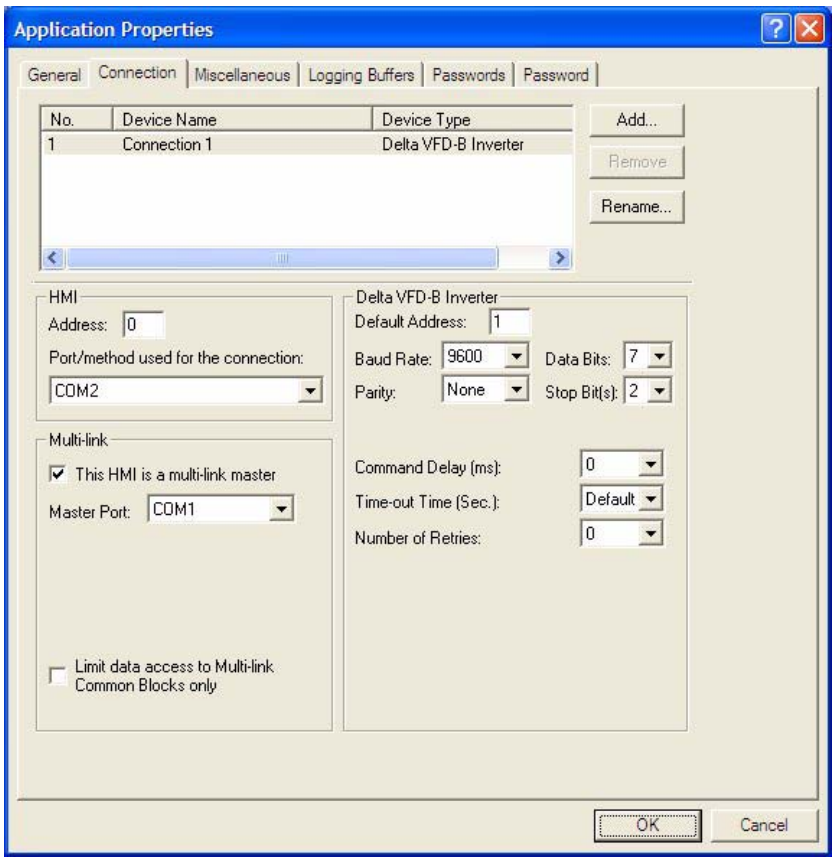
5.1 Communication Parameters

Perform the steps to set up the communication parameters.

Setting up the master

The master is the operator terminal that communicates with the controller.

1. Select **Application/Workstation Setup** and check the **This HMI is a multi-link master** box.



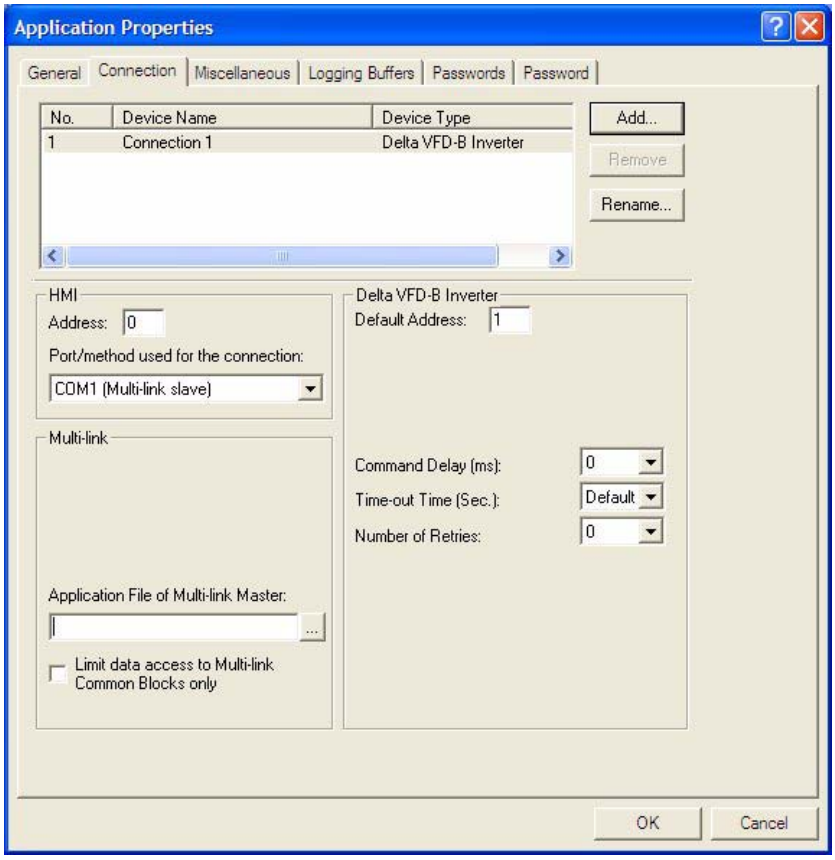
Setting up the multi-link master

2. Download the edited application to the master operator terminal.

Setting up the slaves

The slave operator terminals do not communicate with the controller

3. Select **Application/Workstation Setup**. On the **Connection** tab, select **COM1** port for the connection.
4. To set up the **Default Address**:
   If the slave operator terminal dip switch #5 is set OFF, the operator terminal reads the communication parameters from H-Designer. The unique address (2-10) must be entered in **Default Address**.



Setting up slave operator terminal parameters

Remember to compile and download applications each time after making any changes to the address.

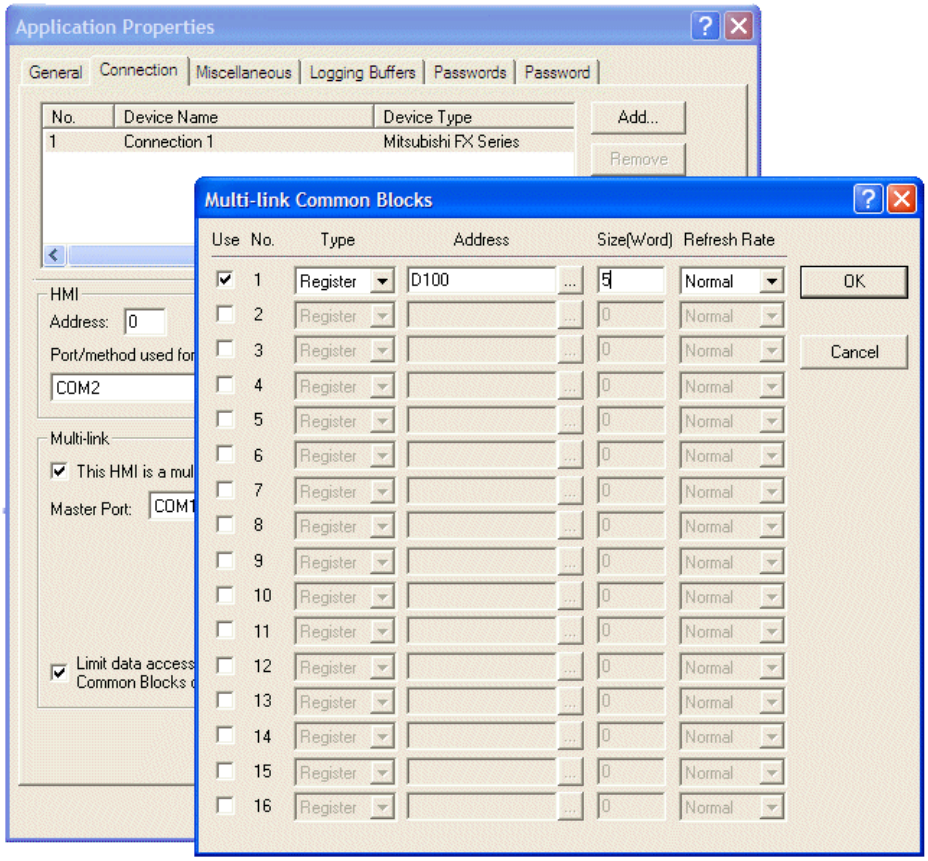
If the slave operator terminal dip switch #5 is set ON, the operator terminal reads the parameters from the hardware. You must enter the address (1-15) in **HMI Node Address**.

5. Download the edited application to the slave operator terminals.

Note:
The address number of the master operator terminal will not affect communication with the slave operator terminals. It is not necessary to set up the baud rate or data type for the slave operator terminals. The purpose of setting up the slave operator terminals is to make sure the connection port for the master operator terminal is correctly set.

5.2 Communication Efficiency

H-Designer allows you to specify a **Common Register Block** (**CRB**) and a **Common On/Off Block** (**COB**) for the operator terminals. Select **Limit data access to Multi-link Common Blocks only**.



The CRB is a block of registers and the COB is a block of On/Off locations in the controller. In every read cycle, the master operator terminal reads the data from both the CRB and the COB. Then the master sends the CRB and COB data to all the slaves.

The CRB and the COB allows a maximum of 128 words and 256 words respectively. In multi-link, the CRB and the COB have to be specified with the same size and format for each of the operator terminals. The slaves are not requested to read the data from the CRB or the COB directly. The slaves read the data from the buffer containing the data sent by the master. The CRB and the COB play important roles in terms of communication efficiency since they can reduce traffic in the multi-link as well as in the link between the master and the controller

For example, arranging control blocks for the operator terminal in the CRB and the COB is one of the most effective ways to improve performance. Arranging the variables common to some of the operator terminals in the CRB or the COB will also improve performance. Doing so will result in a high refresh rate for the variables held in the CRB and the COB since the variables are refreshed every read cycle.

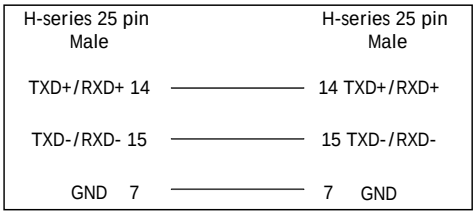
In addition to the CRB and the COB, remember to make use of the register blocks and on/off blocks for screens, since these too lessen the burden of the operator terminal by reducing the number of read commands.

It is recommended to specify the CRB and the COB with continuous locations when designing screens.

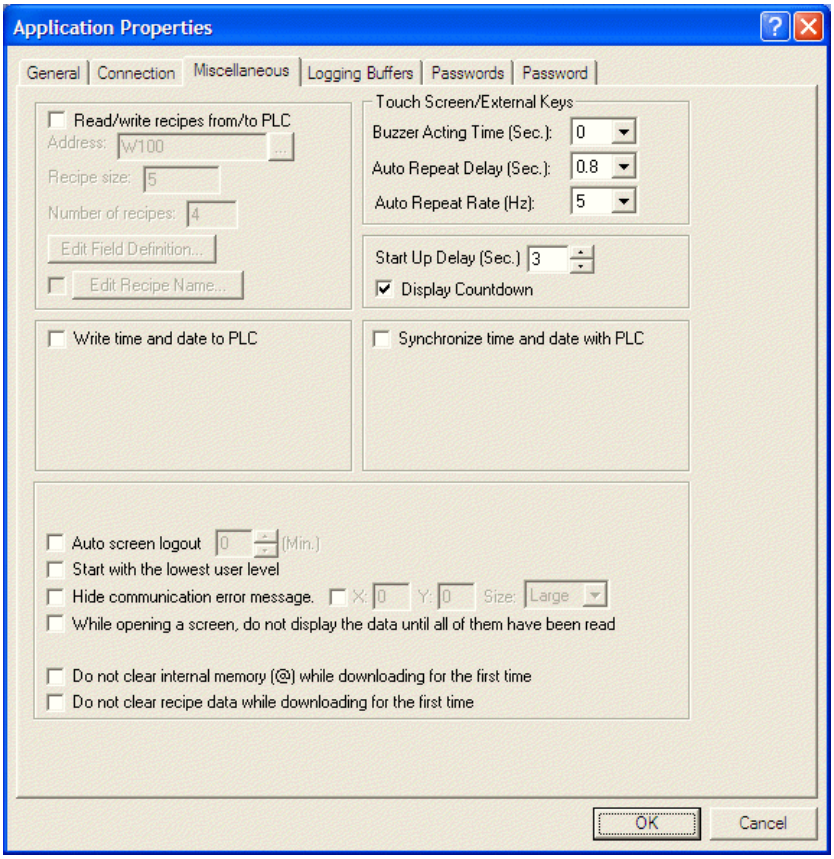
5.3 Important Notes

When using multi-link, please note the following points:

1. The RS485 connection method is suggested for the multi-link. The RS232 connection method is suggested for a single slave operator terminal.



2. Each slave must have its own unique address.
3. The operator terminals on the same multi-link must have the same CRB and COB.
4. The master operator terminal should only start after all the slaves have displayed their first screens. To delay the start-up of the master, select the **Miscellaneous** tab in the **Application Properties** dialog box. Then set the time for the **Start Up Delay**.



Setting up delay of start-up

6 Ethernet Communication

The following chapter will introduce communication setup, application upload/download and Ethernet communication with Ethernet-enabled controllers and operator terminals. There are two methods for Ethernet-enabled operator terminals to communicate with multi-operator terminal/controllers over Ethernet: multi-link and cross-link.

Note:
Ethernet communication is not available for all operator terminal models: please refer to *Appendix A - H-Designer Features and Operator Terminal Models* for details.

6.1 Connection

There are two methods to set up the link; using a RJ45 straight through cable or using a RJ45 crossover cable. The RJ45 crossover cable requires a HUB for connection.

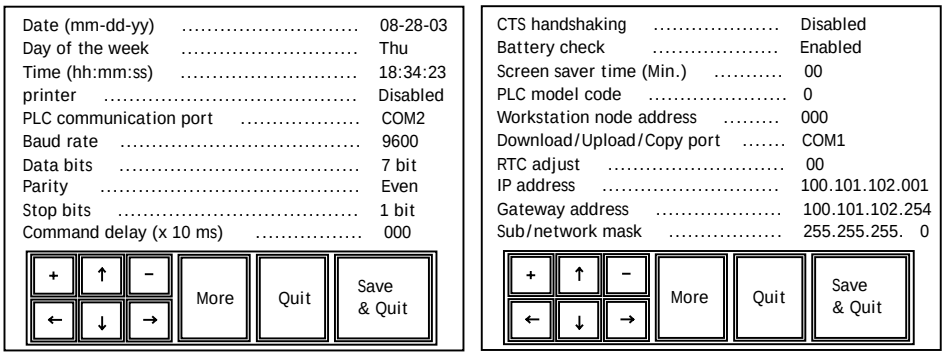
The choice between these two methods depends on your needs and available equipment. The following table describes differences between these two methods.

RJ45 crossover cable	RJ45 straight through cable
Requires no HUB; links to operator terminal directly	HUB required
1-to-1 only	Multi-link

6.2 IP Address Setup

To read or send data from an operator terminal over Ethernet, correct IP addresses have to be set up.

The IP address can be set under **Configure** in the operator terminal system menu.

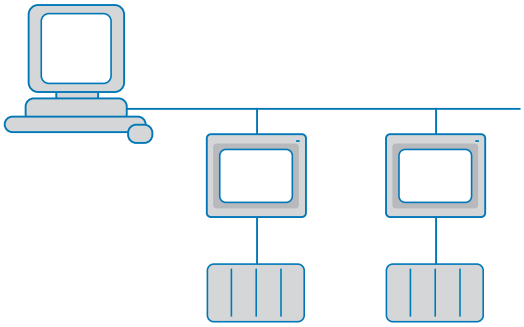


Configuration table of operator terminal with support for networking

Note:
If uploading/downloading over Ethernet, the first three segments of the PC IP address must be the same as the first three segments of the operator terminal IP address. **Example**: PC IP address=192.168.1.10 and operator terminal IP address=192.168.1.XXX. The IP address of the operator terminal should not be shared with other units in the network.

6.3 Application Upload/Download over Ethernet

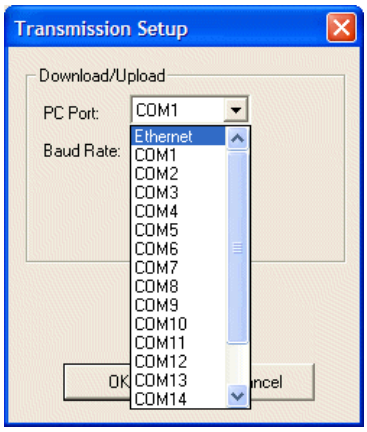
The Ethernet-enabled operator terminals together with H-Designer provide upload/download over Ethernet for application, firmware, recipes and source code.



Uploading/downloading over Ethernet

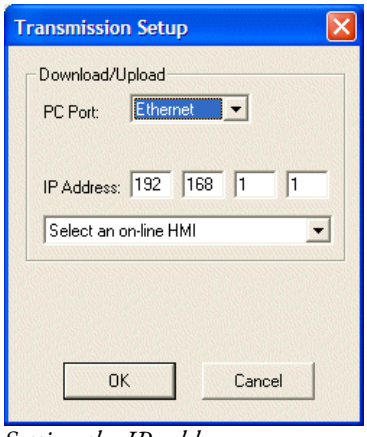
Preform the following steps to upload/download an H-Designer application over Ethernet:

1. Set the IP address, gateway address etc in the **Configuration Table**. Please see the section *IP Address Setup* for details.
2. In H-Designer, select **Options/Transmission Setup** and select **Ethernet** from the **PC Port** list.



Selecting PC Port

3. Enter the address under **IP Address** or select from the drop-down list.

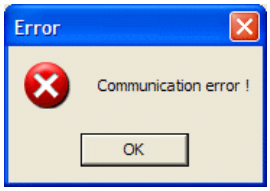


Setting the IP address

4. Select **Application/Download Application** or **Download Firmware and Application** to download the application.

Follow the same steps to **Upload Application**, **Upload Recipes**, **Download Recipes** or **Reconstruct Source over Ethernet**. For **Upload Application**, the steps above must be changed to select **File/Upload Application.**

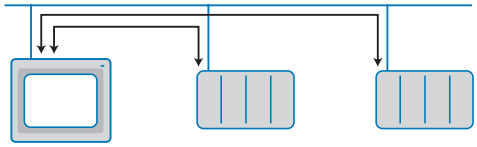
If the link is not set up properly, H-Designer will display an error message.



Communication error message

6.4 Communication with Ethernet-enabled Controllers

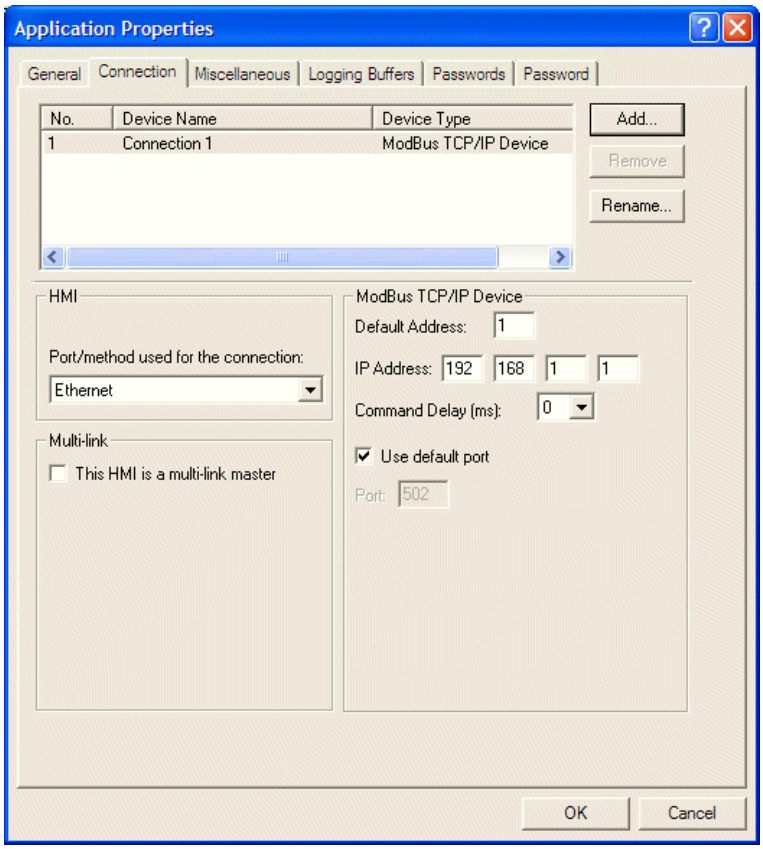
H-Designer supports operator terminals communicating with Ethernet-enabled controllers using Modbus TCP/IP. Consequently, the operator terminal can control or read data from the controllers.



Connection to Ethernet-enabled controllers

Perform the following steps to set up the connection:

1. Select **Application/Workstation Setup**. On the **General** tab, select the type of controller or **Modbus TCP/IP Device** from the **Controller/PLC** list.
2. On the **Connection** tab, select **Ethernet** for **Port/method used for the connection**. Enter the address in the **Default Address** and **IP Address** boxes.

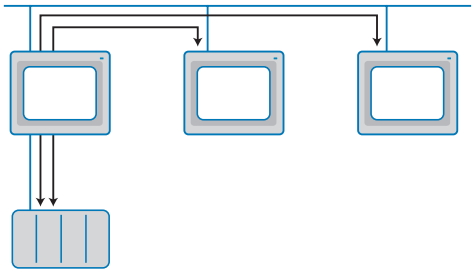


Selecting communication method and setting the IP address

3. Download the H-Designer application file to the operator terminal and set up the link to connect.

6.5 Multi-Link - One Master and Multiple Slaves

The Multi-Link over Ethernet function allows the linking of several operator terminals (one master and multiple slaves). This speeds up communication between the operator terminals.



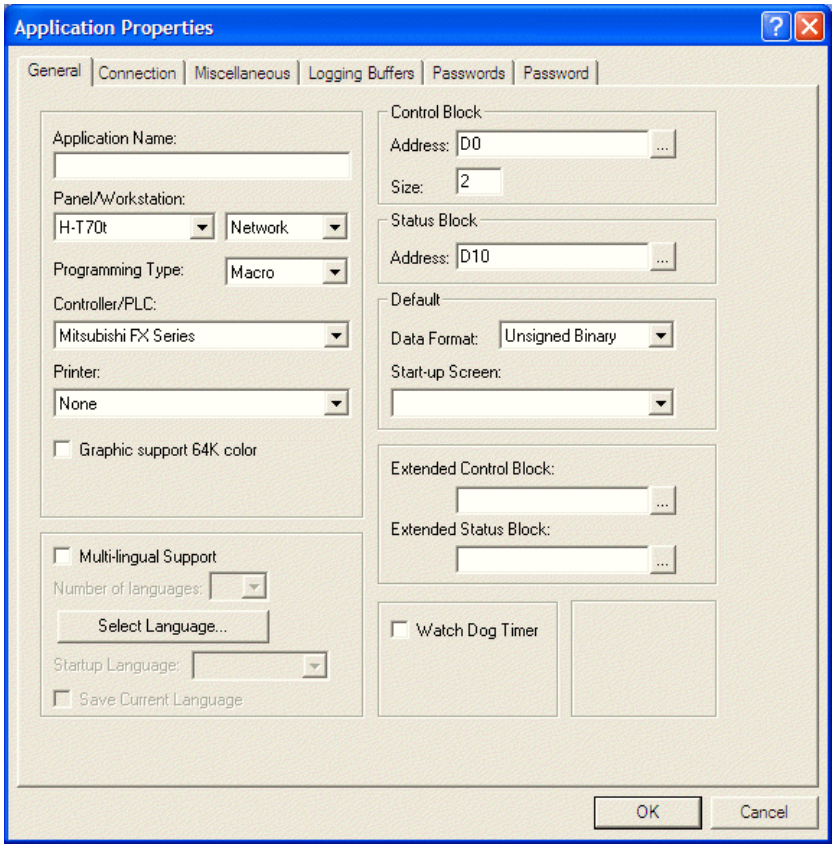
Multi-Link over Ethernet

Perform the following steps to set up the communication:

Setting up the master

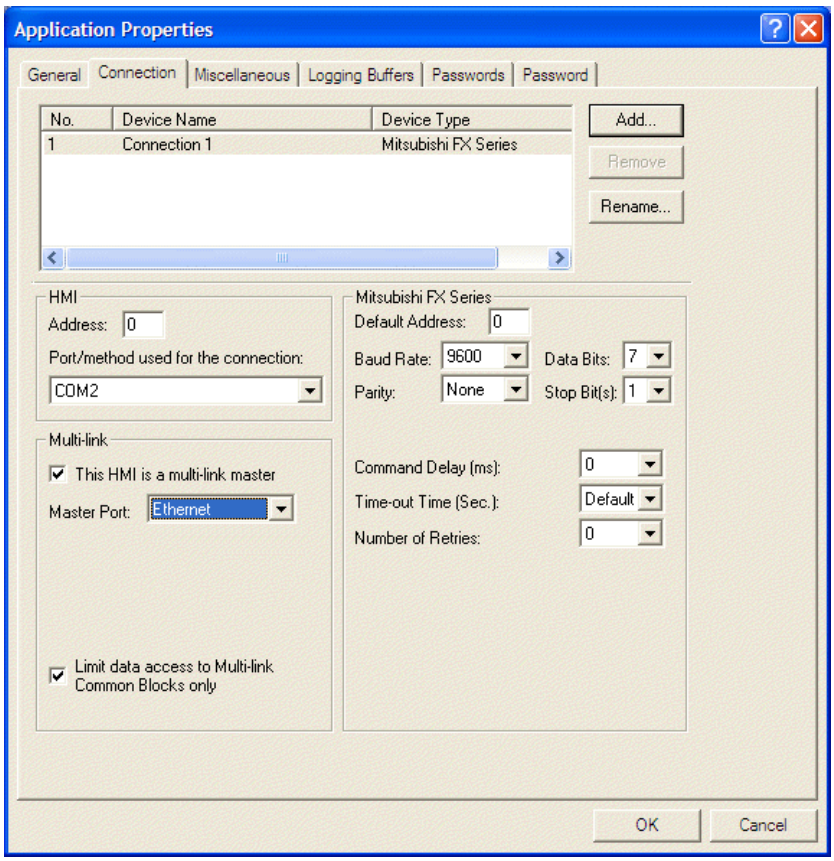
The master is the operator terminal that communicates with the controller.

1. Select **Application/Workstation Setup**. On the **General** tab, select the operator terminal model from the **Panel/Workstation** list and the type of controller from the **Controller/PLC** list.



Selecting operator terminal model and controller model

2. On the **Connection** tab, check the **This HMI is a multi-link master** box and select **Ethernet** from the **Master Port** list. Next, check **Limit data access to Multi-link Common Blocks only**.



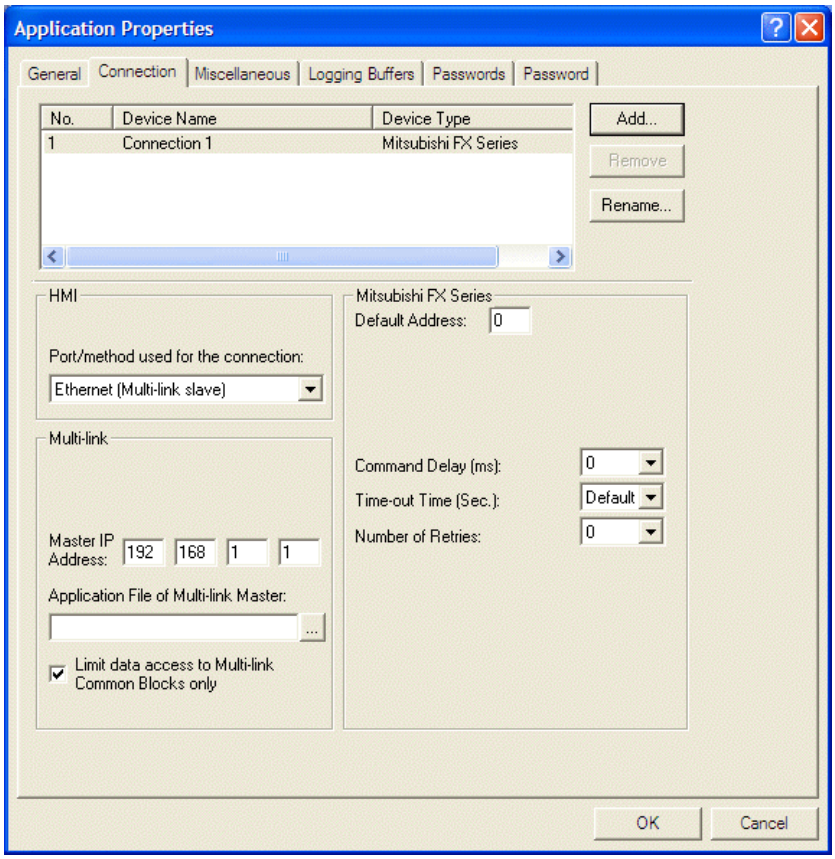
Setting up the multi-link master

3. Download the edited application to the master operator terminal.

Setting up the slaves

The slave operator terminals do not communicate with the controller

4. Select **Application/Workstation Setup**. On the **Connection** tab, select **Ethernet (Multi-link slave)** port for the connection.
5. Set up **Master IP Address**, **Common Register Block**, **Common On/Off Block**, **CRB Size** and **COB Size**.

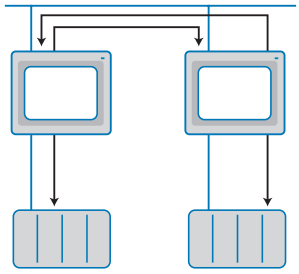


Setting up connection method, master IP address etc.

6. Download the edited application to the slave operator terminals.

6.6 Cross-Link over Ethernet (Data Sharing)

The Cross-Link over Ethernet function allows you to link several operator terminals and controllers and share data between them.

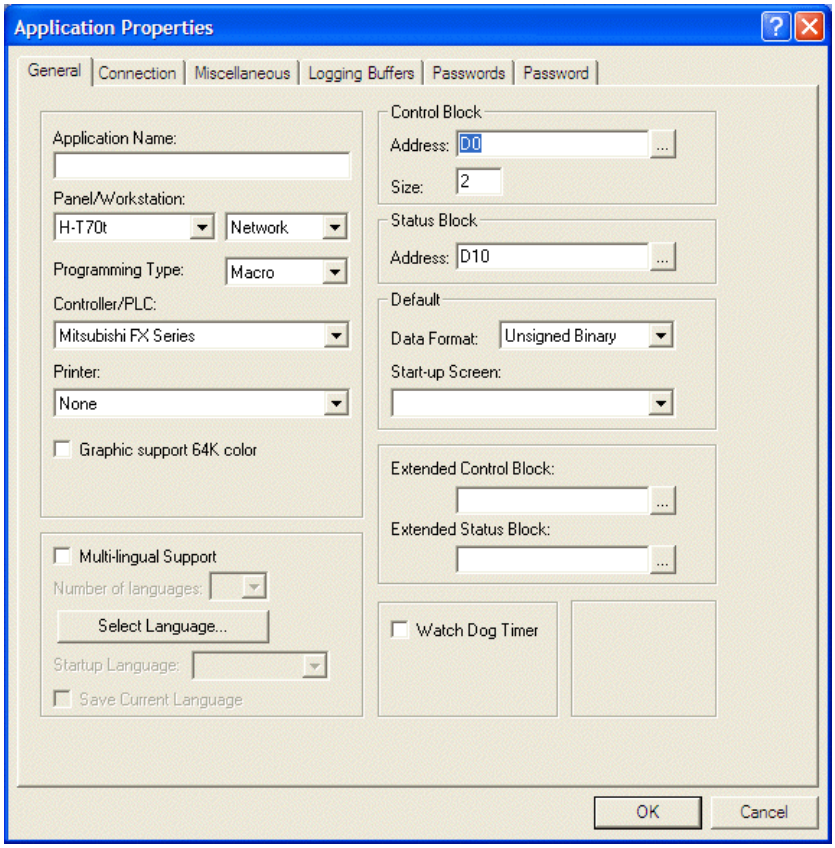


Cross-Link over Ethernet function

The following steps are performed to set up the connection:

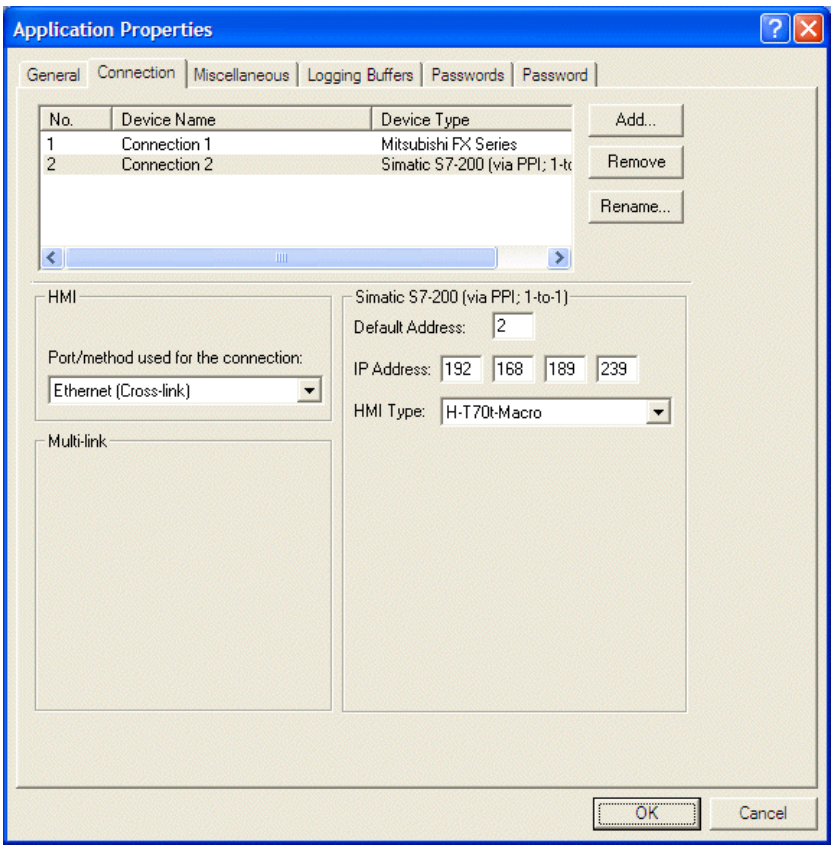
Setup in the first operator terminal

1. Select **Application/Workstation Setup**. On the **General** tab, select the operator terminal model from the **Panel/Workstation** list and the controller type from the **Controller/PLC** list.



Selecting operator terminal model and controller model

2. On the **Connection** tab, click **Add** to add Connection 2 for the Cross-link. Select **Ethernet (Cross-link)** from the **Port/method used for the connection** list. Enter the **Default Address**, **IP Address** and **HMI Type** for the desired device.



Adding the Cross-link device

Note that **Connection 1** is linked to the controller by COM Port while **Connection 2** is linked to the controller by Cross-Link Ethernet.

If the operator terminal using Connection 1 is to access data from the controller linked to Connection 2, follow these steps:

Controller register address setup:

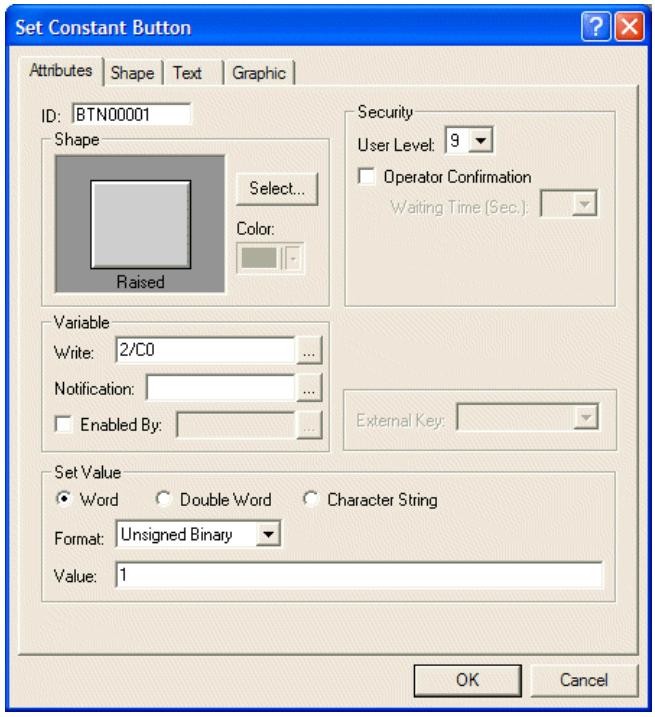
Example: Specify the address of the controller register, **2/C0** for Siemens S7-200. Note that controller register **2/C0** refers to **Connection 2** on the **Connection** tab. **/** denotes the separation from the register address.

3. There are three ways to set this up:

   a) Double-click on the object. Select the **Attributes** tab. For **Write** under **Variable** enter the location **2/Q0.0**.

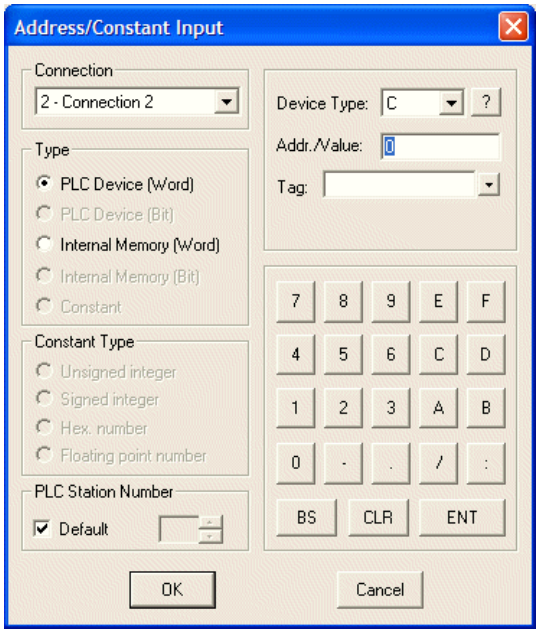
   b) Enter the address of the controller register in the object attributes dialog box.

In the example the address is **2/C0**.



c) Click ... to display the **Address/Constant Input** dialog box.

Select **2-Connection 2** from the **Connection** list. Enter the address in the **Device Type** and **Addr./Value** boxes. In the example the address is **C0**.

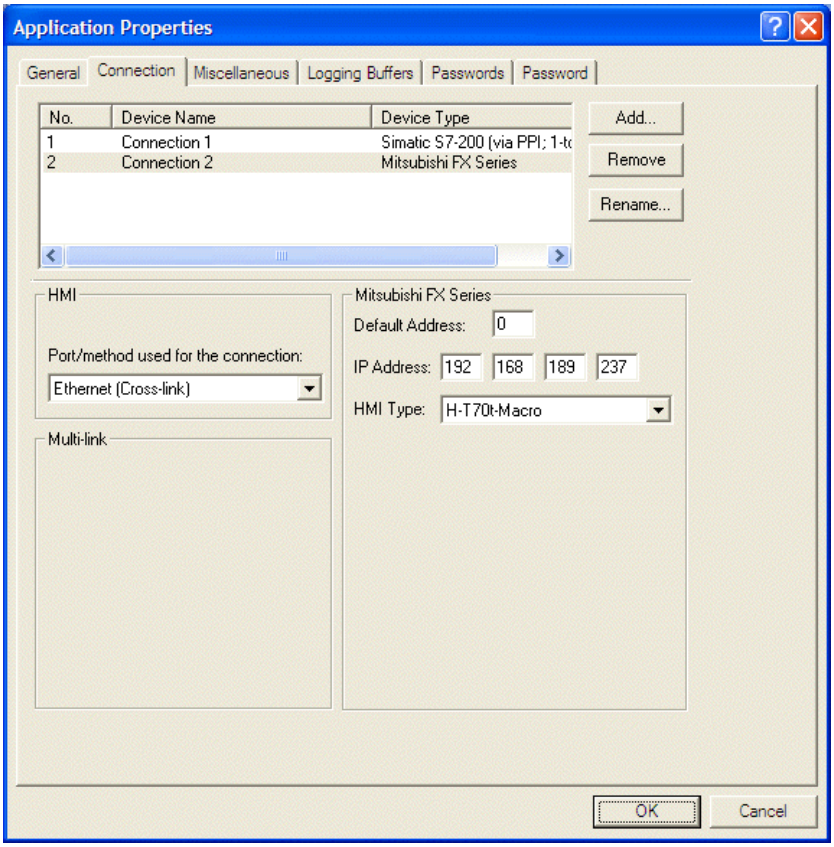


Click **OK** and **2/C0** will appear in the object attribute dialog box.

4. Download the edited application to the first operator terminal.

Setup in the second operator terminal

5. The setup steps are the same as for the first operator terminal, with the difference that the controller device in **Connection 1** of the second operator terminal is the controller device in **Connection 2** of the first operator terminal. Note that the **Addr./Value** of the controller device **Connection 2** must be changed.



The connections in operator terminal 2

6. Download the H-Designer application file to operator terminal 1 and operator terminal 2. Connect the link to the controllers and network.

7 Multi-Channel Communication

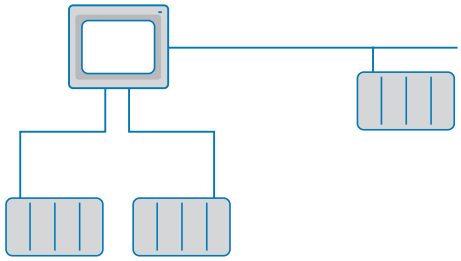
The operator terminal model with 2 COM Ports and an Ethernet connection can be used to connect controllers or other equipment (such as temperature controllers, servers, inverters etc.) from different vendors in order to integrate and collect data using an operator terminal or PC.

Note:
This feature is not available for all operator terminal models: please refer to *Appendix A - H-Designer Features and Operator Terminal Models* for details.

7.1 Connection

COM1, COM2 or the Ethernet Port can be used to link the Multi-Channel connection.

The link can be set up via RS232, RS422 or RS485, with the connection method based on needs and available equipment. For Ethernet, RJ45 has to be used to set up the link. The controller must also be Ethernet-enabled.



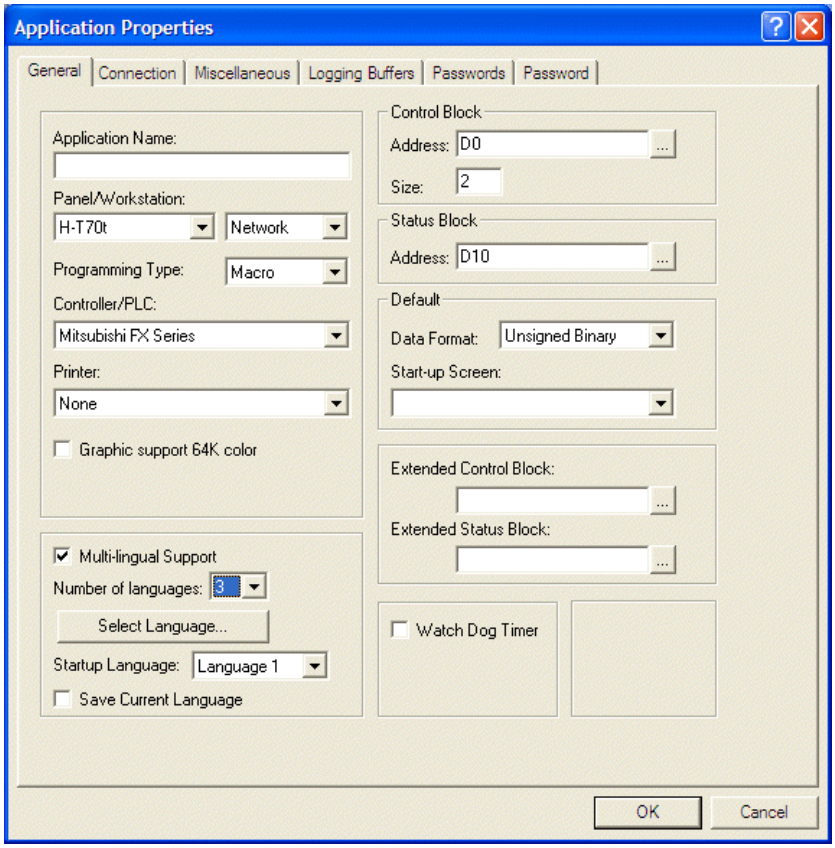
Multi-Channel Connection

7.2 Connection Setup

The Multi-Channel connection setup includes the controller type and its parameters.

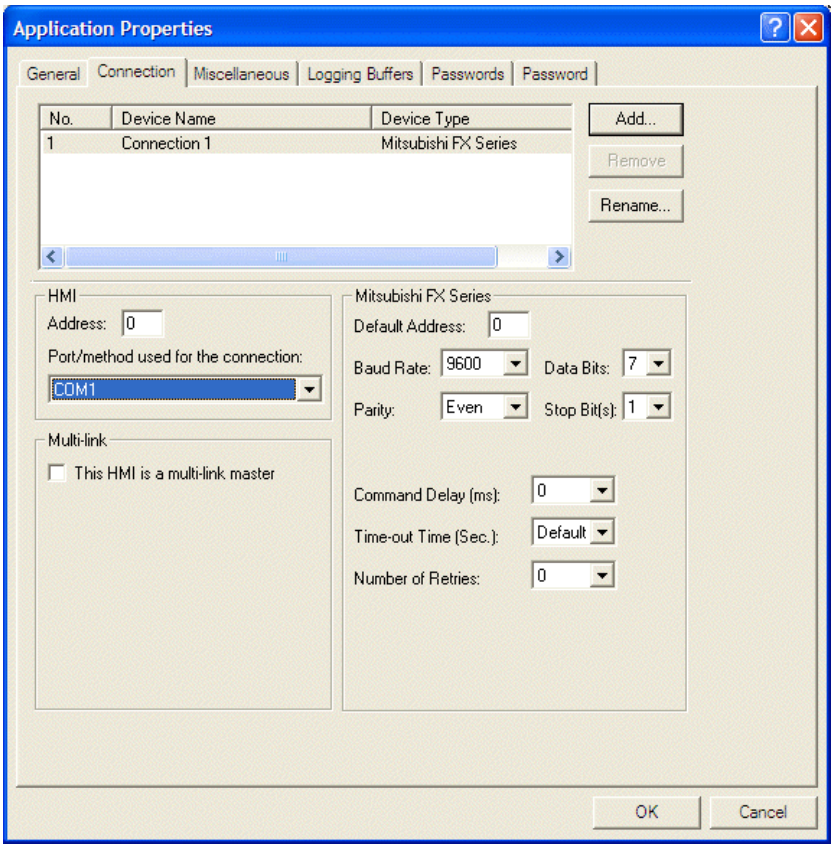
Follow the steps below to set up the connection:

1. In H-Designer, select **File/New**. The **Applications Properties** dialog box is displayed. On the **General** tab, select the type of the first controller from the **Controller/PLC** list, for example, **Mitsubishi FX Series**.



Selecting the type of the first controller

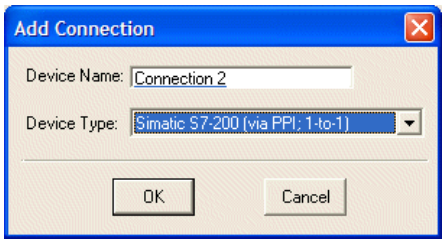
2. On the **Connection** tab, select the method of connection for the first controller from the **Port/method used for the connection** list. Enter the addresses in the **HMI Address** box and controller's **Default Address** box and make the appropriate selections for **Baud Rate**, **Data Bits**, **Parity** and **Stop Bits**.



Setup of the connection with the first controller

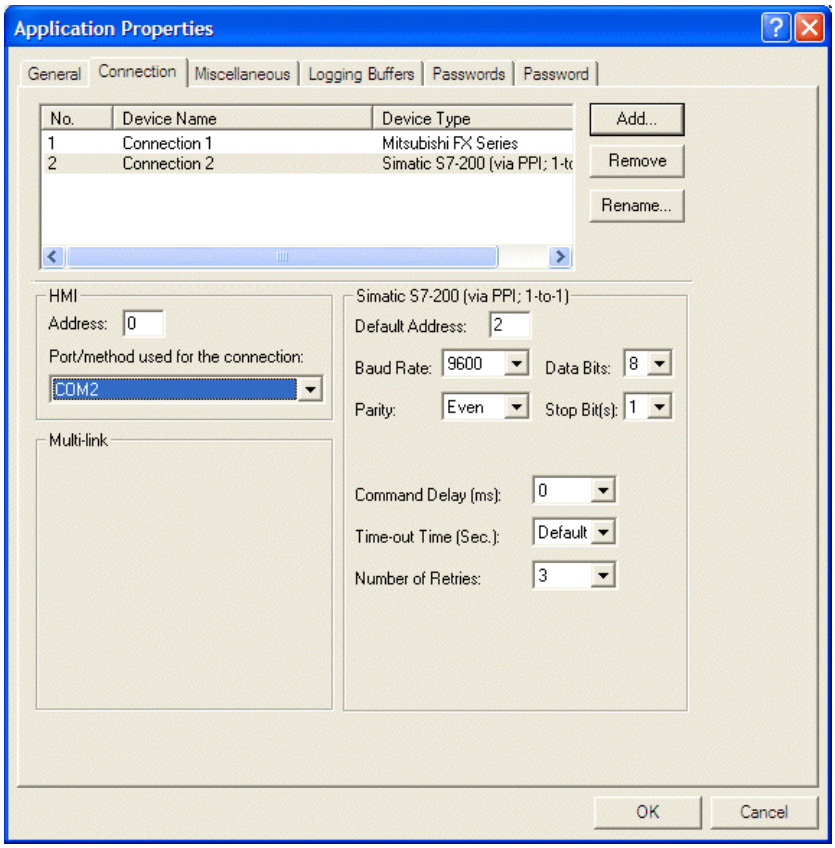
Note:
If **SW5 = ON** on the operator terminal, the parameters of the first linked controller must be set up according to the **Configuration Table** in the operator terminal's **System Menu**. If **SW5 = OFF**, the parameters of the first linked controller must be set up according to the **Connection tab** in **Application/Workstation Setup** in H-Designer. The switches are described in the Installation and Operation Manual for the operator terminal.

3. To add a second controller, click **Add** on the **Connection** tab, and select, for example, **Simatic S7-200 via PPI; 1-to-1**.



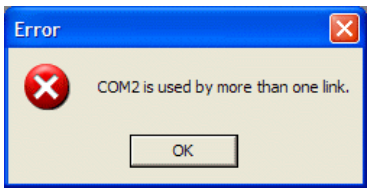
Adding a second linked controller

4. Enter the addresses in the **HMI Address** box and controller's **Default Address** box and make the appropriate selections for **Baud Rate**, **Data Bits**, **Parity** and **Stop Bits**.



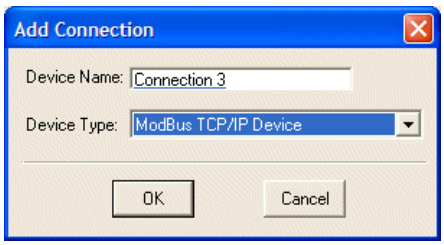
Setup of the connection with the second controller

5. If the communication port is already being used, the following error message will be displayed.



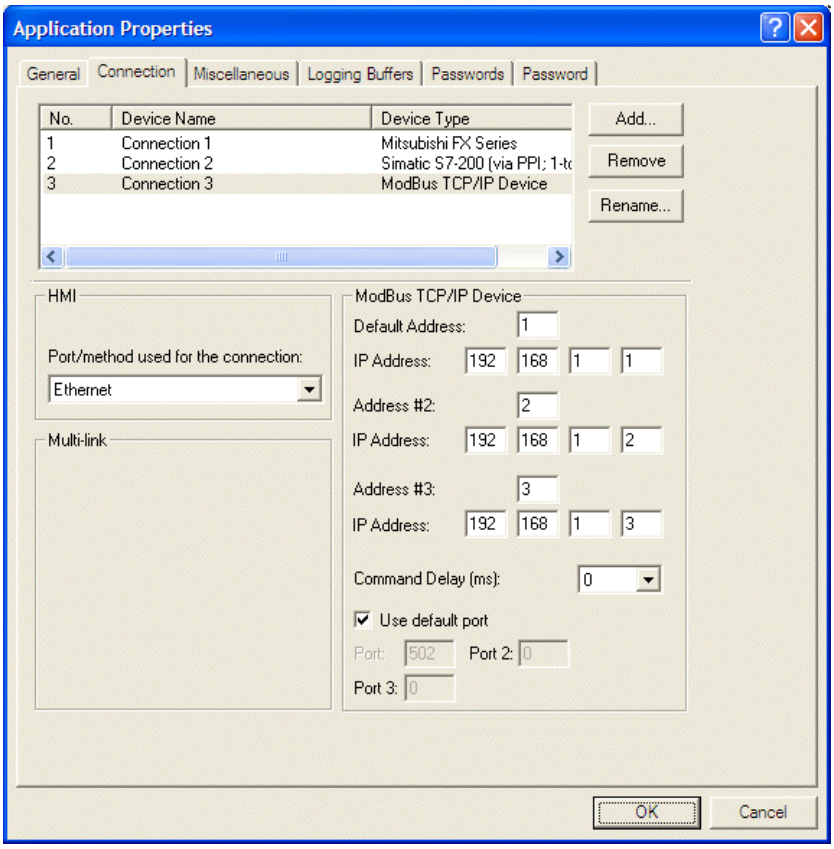
Error message when the port is already being used

6. To add an Ethernet-enabled controller, click **Add** again, to add a third controller, for example, **ModBus TCP/IP Device**.



Adding a third, Ethernet-enabled controller

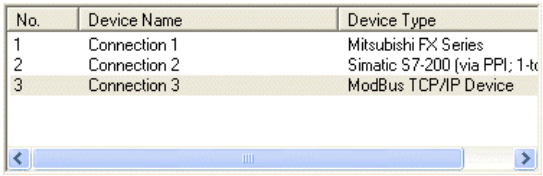
7. Enter the addresses in the **Default Address** and **IP Address** boxes.



Setup of the connection with the third, Ethernet-enabled controller

8. Click **OK** to finish the setup. To change the setup later, simply select **Application/Workstation Setup.**

Description of the Devices block on the Connection tab



Item	Description
Add	Click to add a new connection device/controller.
Remove	Click to delete a connection device. The first connection (Connection 1) cannot be deleted.
Rename	Click to change **Device Name** or **Device Type**. For **Connection 1**, only the **Device Name** can be changed. The **Device Type** can be changed on the **General** tab.
No. column	Numbered according to the order in which devices are added and cannot be changed.

7.3 Read/Write Address Setup

Since there is more than one type of controller, a read/write address for each controller has to be assigned. The symbol **/** denotes the separation of the connection number from the register address.

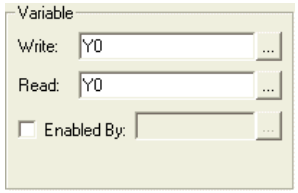
Note:
This function is not available for all operator terminal models: please refer to *Appendix A - H-Designer Features and Operator Terminal Models* for details.

The connection in section *7.2 Connection Setup* is used in this example.

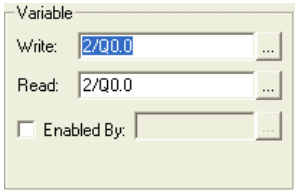
No.	Device Name	Device Type
1	Connection 1	Mitsubishi FX Series
2	Connection 2	Simatic S7-200 (via PPI;1 to 1)
3	Connection 3	Modbus TCP/IP Device

Perform the following steps:

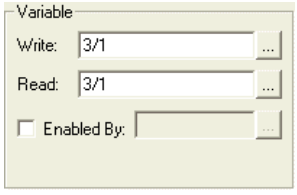
1. For **Connection 1**, in the **Write** and **Read** boxes, enter **1/Y0** for the bit address and **1/D100** for the register address. **1** is for **column No.1**, and can be omitted, so enter **Y0**.



2. For **Connection 2**, in the **Write** and **Read** boxes, enter **2/Q0.0** for the bit address. Note that **2** refers to **Connection No. 2** and **/** denotes the separation from its register address.



3. For **Connection 3**, in the **Write** and **Read** boxes, enter **3/1** for the bit address. Note that **3** refers to **Connection No. 3** and **/** denotes the separation from its register address.



8 Macros

8.1 Macro Function

H-Designer offers user a convenient and powerful macro application. It enables the operator terminal to execute a number of tasks: Arithmetic, Logic, Flow Control, Data Transfer, Comparison, Conversion and System Service Instructions, for example. Using macros can also significantly reduce program size and optimize controller efficiency. Macros not only allow the operator terminal to communicate with the controller, but can also connect it to other devices. Macros provide an efficient integration system as well as an economical hardware application structure.

8.2 Macro Classifications

Macros offer a number of functions for different situations and applications. The relevant macro window can be used to define an application according to needs. The operator terminal will execute the macro commands in accordance with different modes.

Macros can be categorized into **Application Macros**, **Screen Macros**, **ON/OFF Macros** and **Sub-Macros**.

8.2.1 Application Macros

There are three types of macro commands in the **Application** menu.

1. **INITIAL Macro**: The **INITIAL Macro** is used for data initialization and communication parameter declarations. This command is executed only once when an application is started, and the start-up screen does not appear until the command is executed. There is one **INITIAL Macro** in an application.
2. **BACKGROUND Macro**: When the operator terminal runs the application, these macros will be executed cyclically. The maximum size of macro commands are 30 rows. The macro commands will execute whatever the current screen is. Common uses for the **BACKGROUND Macro** are communication control and controller sample data conversion.
3. **CLOCK Macro**: When the operator terminal runs this application, these macros will be executed every 500 ms. Common uses for the **CLOCK Macro** are display control, controller bit monitor, timer control and data timer conversion.

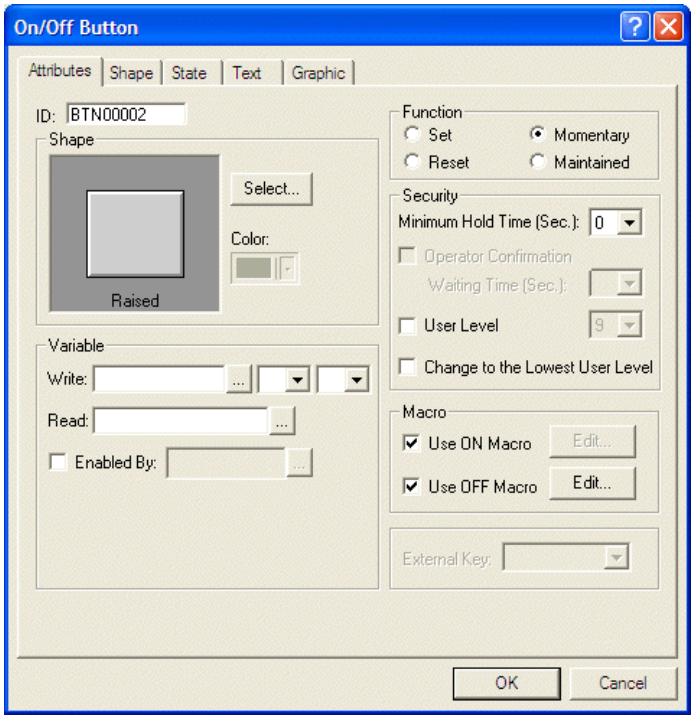
8.2.2 Screen Macros

There are three types of macro commands in the **Screen** menu.

1. **OPEN Macro**: The **OPEN Macro** is executed when the screen is commanded to be opened. Common uses for the **OPEN Macro** are screen initialization, display control, internal register or bit initialization.
2. **CLOSE Macro**: The **CLOSE Macro** is executed when the screen is commanded to be closed. The **CLOSE Macro** will execute only once.
3. **CYCLIC Macro**: The **CYCLIC Macro** is executed cyclically when the screen is displayed. The operator terminal will execute the **BACKGROUND Macro** and **CLOCK Macro** periodically.

8.2.3 ON/OFF Macros

There are two **ON/OFF Macro** commands for push-button objects.



ON/OFF macros are available for the button object

ON Macro: The **ON Macro** is executed when the button is clicked and sets a bit to ON. Common uses for the **ON Macro** are push-button actions, chain process control, initial screen display and controller register and bit initialization.

OFF Macro: The **OFF Macro** is executed when the button is clicked and released, sets a bit to OFF. The operator terminal will execute the **OFF Macro** commands once. Common uses for the **OFF Macro** are push-button actions, sequence process control and displaying the close screen.

8.2.4 Sub-Macros

The **Sub-Macro** can be selected from the **Object** menu.

Sub-Macro: The **Sub-Macro** is a sub-command of **Macro**. The **Sub-Macro** is executed by the operator terminal with the CALL command. Common uses for the **Sub-Macro** are to edit and save some basic functions and macro arithmetic commands.

8.3 Macro Commands

The following chart details the macro commands and their formats. For the set up procedure, please see the next section.

Operation	Format	A1*	A2*	A3*	Data Format	Controller Data
ADD	A1 = ADD (A2, A3)	2	2, 4	2, 4	DW/Signed	-
SUB	A1 = SUB (A2, A3)	2	2, 4	2, 4	DW/Signed	-
MUL	A1 = MUL (A2, A3)	2	2, 4	2, 4	DW/Signed	-
DIV	A1 = DIV (A2, A3)	2	2, 4	2, 4	DW/Signed	-
MOD	A1 = MOD (A2, A3)	2	2, 4	2, 4	DW/Signed	-
OR	A1 = A2 | A3	2	2, 4	2, 4	DW	-
AND	A1 = A2 & A3	2	2, 4	2, 4	DW	-
XOR	A1 = A2^A3	2	2, 4	2, 4	DW	-
SHL	A1 = A2<<A3	2	2, 4	2, 4	DW	-
SHR	A1 = A2>>A3	2	2, 4	2, 4	DW	-
MOV	A1 = A2	0, 2	0, 2, 4	-	DW	Yes
BMOV	BMOV (A1, A2, A3)	0, 2	0, 2	2, 4		Yes
FILL	FILL (A1, A2, A3)	2	2, 4	2, 4		-
CHR	CHR (A1, "A2")	2	5	-	-	-
GETX	A1=@X	2,4				X
SETY	@Y=A1	2,4				X
X2Y	X2Y(A1,A2)	2,4	2,4			X
IF==	IF A1 == A2 GOTO A3	2, 4	2, 4	4	DW/Signed	-
IF!=	IF A1! = A2 GOTO A3	2, 4	2, 4	4	DW/Signed	-
IF>	IF A1>A2 GOTO A3	2, 4	2, 4	4	DW/Signed	-
IF>=	IF A1>=A2 GOTO A3	2, 4	2, 4	4	DW/Signed	-
IF<	IF A1<A2 GOTO A3	2, 4	2, 4	4	DW/Signed	-
IF<=	IF A1<=A2 GOTO A3	2, 4	2, 4	4	DW/Signed	-
IF AND ==0	IF A1 AND A2==0 THEN GOTO A3	2, 4	2, 4	4	DW	-
IF AND !=0	IF A1 AND A2!=0 THEN GOTO A3	2, 4	2, 4	4	DW	-
IF==ON	IF A1 = ON GOTO	3	4	-	Bit	-
IF==OFF	IF A1 = OFF GOTO	3	4	-	Bit	-
IF-THEN	IF condition ** THEN DO ENDIF	2, 4	2, 4	-	Condition **	-

IF-THEN-ELSE	IF condition ** THEN DO
	ELSE DO
	ENDIF
		2, 4	2, 4	-	Condition **	-

Nest IF-THEN-ELSE	IF condition ** THEN DO
	IF-THEN-ELSE
	ELSE DO
	IF-THEN-ELSE
	ENDIF
		2, 4	2, 4	-	Condition **	-

* The usable range of memory will be identified according to the commands. The numbers in the table represent: 0 = Controller Device (word), 1 = Controller Device (bit), 2 = Internal Memory (word), 3 = Internal Memory (bit), 4 = Constant, 5 = ASCII Character

Operation	Format	A1*	A2*	A3*	Data Format	Controller Data

ELIF	IF condition 1** THEN DO
	ELIF condition 2** THEN DO
	ELIF condition 3** THEN DO
	ENDIF
		2, 4	2, 4	-	Condition **	-

** Condition includes A1==A2, A1!=A2, A1>A2, A1>=A2, A1<A2, A1<=A2, (A1&A2)==0, (A1&A2)!=0, A1==ON and A1==OFF. A1 and A2 are only for internal memory and constant.						
GOTO	Goto label A1	4	-	-		-
LABEL	Label A1	4	-	-		-
CALL	Call A1	2, 4	-	-		-
RET	Return	-	-	-		-
FOR	For A1	2, 4	-	-		-
NEXT	Next	-	-	-		-
SETB	Bit setting	1, 3	-	-	Bit	Yes
CLRB	Bit resetting	1, 3	-	-	Bit	Yes
INVB	Bit inversion	1, 3	-	-	Bit	Yes
BCD	A1 = BCD (A2)	2	2	-	DW	-
BIN	A1 = BIN (A2)	2	2	-	DW	-
W2D	A1 = W2D (A2)	2	2	-	Signed	-
B2W	A1 = B2W (A2, A3)	2	2	2, 4		-
W2B	A1 = W2B (A2, A3)	2	2	2, 4		-
SWAP	SWAP (A1, A2)	2	2, 4	-		-
MAX	A1 = MAX (A2, A3)	2	2, 4	2, 4	DW/Signed	-
MIN	A1 = MIN (A2, A3)	2	2, 4	2, 4	DW/Signed	-
A2H	A1 = A2H	2	2			-
H2A	A1 = H2A (A2)	2	2			-
TIMETICK	A1 = TIMETICK	2	-	-	DW	-
COMMENT	#A1 = "Chars"	5	-	-		-
SYS	SYS (A1, A2)					-
	SYS (SET_TIMER,N)		4			-
	SYS (STOP_TIMER,N)		4			-
	SYS (STOP_COUNTER,N)		4			-
	SYS (WAIT_TIMER,N)		4			-
	SYS (INI_COM,N)		4			-
	SYS (GET_CHAR,N)		4			-
	SYS (GET_CHARS,N)		4			-
	SYS (PUT_CHAR,N)		4			-
	SYS (PUT_CHARS,N)		4			-
	SYS (READ_WORDS,N)		4			-
	SYS (READ_BITS,N)		4			-
	SYS (WRITE_WORDS,N)		4			-
	SYS (WRITE_BIT,N)		4			-
	SYS (SUM_ADD,N)		4			-
	SYS (SUM_XOR,N)		4			-

* The usable range of memory will be identified according to the commands. The numbers in the table represent: 0 = Controller Device (word), 1 = Controller Device (bit), 2 = Internal Memory (word), 3 = Internal Memory (bit), 4 = Constant, 5 = ASCII Character

8.3.1 Arithmetic

Note:
Only internal memory can be used in these commands. The internal memory includes @, RCPW, CB, RCPNO and *@ (indirect internal memory). The data format is word, double-word, signed binary and unsigned binary.

Format: A1 = A2 + A3

Description: Adds A2 and A3 and saves the result in A1.

ADD → Format: A1 = A2+A3. Adds A2 and A3 and saves the result in A1.

SUB → Format: A1 = A2-A3. Subtracts A3 from A2 and saves the result in A1.

MUL → Format: A1 = A2×A3.

DIV → Format: A1 = A2/A3. A1 is the quotient and A3 cannot be zero.

MOD → Format: A1 = A2%A3. A1 is the remainder and A3 cannot be zero.

8.3.2 Logical

Note:
Only internal memory can be used in these commands. The internal memory includes @, RCPW, CB, RCPNO and *@ (indirect internal memory). The data format is word, double-word etc. (no signed binary, floating point number arithmetic).

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

*Truth tables for **OR** (to the left) and **AND** (to the right) logical operations*

OR → Format: A1=A2 | A3.

Performs the bit-wise **OR** operation on A2 (word) and A3 (word) and saves the result in A1 (word), or on A2 (double-word) and A3 (double-word) and saves the result in A1 (double-word).

AND → Format: A1 = A2 & A3. Performs the bit-wise **AND** operation on A2 (word) and A3 (word) and saves the result in A1 (word) or on A2 (double-word) and A3 (double-word) and saves the result in A1 (double-word).

XOR → Format: A1=A2 ^ A3. Performs the bit-wise exclusive **OR** operation on A2 (word) and A3 (word) and saves the result in A1 (word) or on A2 (double-word) and A3 (double-word) and saves the result in A1 (double-word).

SHL → Format: A1=A2 << A3. Shifts A2 (word) left by A3 bits and saves the result in A1 (word). The left shift command puts 0 into bit 0 and the last bit will shift out. If the displacement (A3) is greater than 16, then 16 will be the most shiftable amount.
Alternatively, shifts A2 (double-word) left by A3 bits and saves the result in A1 (double-word). The left shift command puts 0 into bit 0 and the last bit will shift out. If the displacement (A3) is greater than 32, then 32 will be the most shiftable amount.

SHR → Format: A1=A2 >> A3. Shifts A2 (word) right by A3 bits and saves the result in A1 (word). The right shift command puts 0 into bit 15 and the first bit will shift out. If the displacement (A3) is greater than 16, then 16 will be the most shiftable amount.
Alternatively, shifts A2 (double-word) right by A3 bits and saves the result in A1 (double-word). The right shift command puts 0 into bit 31 and the first bit will shift out. If the displacement (A3) is greater than 32, then 32 will be the most shiftable amount.

8.3.3 Data transfer

Note:
Both the MOV and BMOV commands can be located in the controller memory or internal memory. These include @, RCPW, CB, RCPNO and *@ (indirect internal memory).
The data format for this command is word.

MOV → Format: A1 (word) = A2 (word), A1 (double-word) = A2 (double-word). The **MOV** command copies the value of A2 to A1 and the value of A2 is unchanged. If A1 is located in the controller, it represents the A2 data in the operator terminal internal register in the controller. If A2 is located in the controller, it will represent the data read and copied from the operator terminal internal register A1.

BMOV → Format: **BMOV** (A1, A2, A3). Copies a block of data starting at A2 to the memory block starting at A1. A3 specifies the number of words to be copied. The data format is word. The **BMOV** command copies a block of length A3 starting at A2 to the A3 long block starting at A1. The A2 data is unchanged. A3 must be between 2 and 524. Format: BMOV (A1, A2, A3).

FILL → Format: **FILL** (A1, A2, A3). Fill a block of memory starting at A1 with the value of A2. A3 specifies the number of words to be filled. The data format is word. The **FILL** command fills a block of A3 words starting at A1 with the A2 data. The A2 data is unchanged. A3 must be between 2 and 524.

CHR → Format: **CHR** (A1, "A2"). Copies the character string A2 to A1. The A1 data is in ASCII format.

GETX → Format: **A1** = **@X**. Convert @X input signal to A1. The A1 data has numeric value.

SETY → Format: **@Y** = **A1**. Convert A1 to output signal. The A1 data has numeric value.

X2Y → Format: X2Y(A1, A2). Convert input to output.

If A2 value is not 0, the specified internal bit A1 is enabled. When A2 value is 0, the action of the specified internal A1 bit is disabled. An internal bit can assign the corresponding input point and output point which the sensitive time needs to be greater than 20 msec. When cancelling the action internal point, the relative digital output point will be reset.

8.3.4 Comparison

Note:
Only internal memory can be used in these commands. The internal memory includes @, RCPW, CB, RCPNO and *@ (indirect internal memory).

IF == → Format: IF A1 == A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is equal to A2.

IF!= → Format: IF A1!=A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is not equal to A2.

IF > → Format: IF A1>A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is greater than A2.

IF >= → Format: IF A1>=A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is greater than or equal to A2.

IF < → Format: IF A1<A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is less than A2.

IF <= → Format: IF A1<=A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is less than or equal to A2.

IF AND == 0 → Format: IF (A1&A2) == 0 THEN GOTO LABEL A3. Goes to LABEL A3 if the result of an AND operation on A1 and A2 is 0.

IF AND! = 0 → Format: IF (A1&A2)! = 0 THEN GOTO LABEL A3. Goes to LABEL A3 if the result of an AND operation on A1 and A2 is not 0.

IF == ON → Format: IF A1==ON THEN GOTO LABEL A2. If bit A1 is ON (1), goes to LABEL A2.

IF ==OFF → Format: IF A1==OFF THEN GOTO LABEL A2. If bit A1 is OFF (0), goes to LABEL A2.

Example

Command: IF . . . DO; ELIF . . . DO; ELSE DO; ENDIF

Description: Use an IF statement when you want your macro to choose between two or more options. An IF statement consists of the keyword IF, a condition to be evaluated, the keyword THEN, the keyword DO, and the keyword ENDIF, as shown below:

IF condition THEN DO

    # statements to be executed if condition is true

ENDIF

The condition can be one of the following:

A1 == A2
A1 != A2
A1 > A2
A1 >= A2
A1 < A2
A1 <= A2
(A1 & A2) == 0
(A1 & A2) != 0
A1 == ON
A1 == OFF

The following IF statement structures are provided:

IF-THEN structure

The simplest IF statement evaluates a condition and performs a specified action if the condition is true. If the condition is not true, the entire statement is ignored. For example:

```
IF @100 == 50 THEN DO
        CALL 50
ENDIF
```

If @100 equals 50, sub-macro 50 is called. If @100 contains anything else, the entire statement is ignored.

IF-THEN-ELSE structure

An IF statement can also specify one or more statements to be executed if the condition is false. This option is indicated with the keyword ELSE, as shown below.

```
IF condition THEN DO
        # statements to be executed if condition is true
ELSE DO
        # statements to be executed if condition is false
ENDIF;
```

In the example below, if @100 equals 50, the sub-macro 50 is called. If @100 is not equal to 50, sub-macro 1 is called following the else statement.

```
IF @100 == 50 THEN DO
        CALL 50
ELSE DO
        CALL 1
ENDIF
```

Nested IF-THEN-ELSE structure

You can create nested IF statements, in which one IF statement is embedded in another:

```
IF first condition THEN DO
        IF second condition THEN DO
            # statements to be executed if second condition is true
        ELSE DO
            # statements to be executed if second condition is false
        ENDIF
ELSE DO
        IF third condition THEN DO
            # statements to be executed if third condition is true
        ELSE DO
            # statements to be executed if third condition is false
        ENDIF
ENDIF
```

In the following example, if the value of @100 is 50, sub-macro 50 is called. If the value of @100 is 100, sub-macro 100 is called. If @100 is not equal to either of those values, the sub-macro 1 is called.

```
IF @100 == 50 THEN DO
        CALL 50
ELSE DO
        IF @100 == 100 THEN DO
        CALL 100
        ELSE DO
        CALL 1
        ENDIF
ENDIF
```

ELIF structure

The ELIF statement is provided to create IF structures in which the ELIF branch of one IF statement leads to another option:

```
IF first condition THEN DO
        # statements to be executed if condition is true
ELIF second condition THEN DO
        # statements to be executed if condition is false
ELIF third condition THEN DO
        # statements to be executed if third condition is false
ENDIF
```

In the following example, if @100 equals 50, sub-macro 50 is called. If @100 is not equal to 50, the program continues to the ELIF statement to test if @100 is equal to 100. If @100 equals 100, sub-macro 100 is called. If @100 is not equal to 100, the program moves to the next ELIF, and so on.

```
IF @100 == 50 THEN DO
        CALL 50
ELIF @100 == 100 THEN DO
        CALL 100
ELIF @100 == 150 THEN DO
        CALL 150
ENDIF
```

You cannot define a label inside an IF statement.

8.3.5 Flow Control

Note:
Only internal memory can be used in these commands.

GOTO → Format: GOTO LABEL A1. Goes to LABEL A1 unconditionally. The GOTO command will cause a branch to the specified label (Label A1). LABEL A1 must be in the macro.

LABEL →Format: LABEL A1. Note that no two labels are allowed to have the same number in one macro but the same number in different macros is acceptable.

CALL → Call Sub-macro. Format: CALL A1. The **CALL** command can assign control to a sub-macro. Common uses of sub-macros are to execute some specific functions, to pass the parameter table and complex instruction sets. Note that the specified sub-macro must exist and return via the **RET** command when the end of the sub-macro is reached. Then the next macro will be executed. The number of sub-macros is from 001 to 512, and they can be named. A sub-macro can also be assigned to **CALL** another sub-macro.

RET → Return to macro. The **RET** command only exists in sub-macro, although **CALL** exists in macro. Each **RET** command must have a corresponding **CALL** command.

FOR..NEXT → Loop, **FOR** is the start of a loop and **NEXT** is the end of a loop. Note that the maximum number of **FOR** loops is 3, for example, the FOR A1. .NEXT. FOR loop is formed by the set of **FOR** and **NEXT** commands and executes the macro instructions within the **FOR** loop A1 times. A1 can be a variable or a constant. When A1 is 0, the macro will skip the **FOR** loop and execute the line of code following the **NEXT** command. When A1 is greater than 0, the macro will execute the loop continuously until the end of the **FOR** loop. The value of A1 can be changed within the **FOR** commands. Note that if A1 is too great, the CPU will overload and malfunction.

The **FOR/NEXT** loop command can execute the program repeatedly. Each **FOR** command must have one corresponding **NEXT** command. You are allowed to have up 3 nested **FOR** loops, such as FOR @1…, FOR @2…, FOR@3... NEXT, NEXT, NEXT.

END → End the macro. The **END** command represents the end of the macro. The macro will not execute the next line of code after the **END** command but will start at the first line of code next time the program is run.

Note:
The **END** command represents the end of the macro and is invalid in sub-macro. Sub-macro must use the **RET** command, otherwise, the program will cause errors.

8.3.6 Data Conversion

Note:
Only internal memory can be used in these commands. The internal memory includes @, RCPW, CB, RCPNO and *@ (indirect internal memory).

BCD → Convert BIN to BCD. Format: A1 = BCD (A2). This command is used to convert A2 (integer, word or double-word) from a binary number to a BCD number and saves the result in A1. Valid integer values of A2 are between 0 and 9999 (word) or 0 and 99999999 (double-word).

BIN → Convert BCD to BIN. Format: A1 = BIN (A2). This command is used to convert A2 from a BCD number (word or double-word) to a binary number and saves the result in A1 (integer, word or double-word). Valid BCD numbers are between 0 and 9999 (word) or 0 and 99999999 (double-word).

W2D → Convert WORD to DOUBLE WORD. Format: A1 = W2D (A2). The W2D command is used to convert A2 from a WORD number (integer) to a DOUBLE WORD (integer) and saves the result in A1 (double-word, signed, or unsigned). Valid integer values of A2 are between 0 and 65535 (word, unsigned) or -32768 and 32767 (word, signed). This function can extend the size of a 16-bit signed integer (word) to a 32-bit integer (double-word).

B2W → Convert BYTE to WORD. Format: A1 = B2W (A2, A3). The byte array starting at A2 with the size A3 and the result is saved in the memory starting at A1 (word). The high bytes of the word array are set to 0.

W2B → Convert WORD to BYTE. Format: A1 = W2B (A2, A3). The word array starting at A2 with the size A3. The result is saved in memory starting at A1. The conversion will discard the high bytes of the A2 word array.

SWAP → Swap the Bytes, Format: SWAP (A1, A2). The SWAP command is used to swap the low byte and high byte of each word of a memory block starting at A1. A2 specifies the size of the memory block in words. After execution, the Al data will be changed.

MAX → Maximum. Format: A1 = MAX (A2, A3). Sets A1 to the larger of A2 and A3. (The data format can be word, dword, signed binary, or unsigned binary.)

MIN → Minimum. Format: A1 = MAX (A2, A3). Sets A1 to the smaller one of A2 and A3. (The data format can be word, dword, signed binary, or unsigned binary.)

A2H → Convert 4-digit hex number in ASCII character form into a binary number. Format: A1 = A2H (A2). The character of the fourth digit is in word A2 and the characters of the other digits are in the words following A2 in sequence. The result will be saved in A1. For example, suppose A2 is @200 and the data in @210=9538H. After the conversion, the result will be saved in A1=@210 and will be @200=0039H, @201=0033H, @202=0035H, and @203=0038H. (The data format is word only.)

H2A → Convert a 16-bit binary number into a 4-digit hex number in ASCII character form. Format: A1 = A2H (A2). The number to be converted is in A2. The character of the fourth digit will be saved in A1 and the characters of the other digits will be saved in the words following A1 in sequence. For example, suppose A2 is @100 and the data in @100=1234H. After the conversion, the result will be saved in A1=@110 and will be @110=0031H, @111=0032H, @112=0033H, and @113=0034H. (The data format is word only.)

8.3.7 Bit Setting

Note:
Both internal memory and controller bit can be used in these commands, including @nnn.b and RCPWnnn.b.

SETB → Set bit to ON. Format: SETB A1.

CLRB → Set bit to OFF. Format: CLRB A1.

INVB → Inverse bit state. Format: INVB A1.

8.3.8 Others

There are three special commands to use.

TIMETICK → Get the current system time tick (CPU internal clock time). Format: A1= TIMETICK (). The system time tick is increased by 1 in every 100 ms.

COMMENT → This is a non-executable instruction and it is used to comment macros.

SYS → There are a number of system services which can be used in the **SYS** command. Please see below for full details:

1. **SET_TIMER** → Specify the internal timer.
   Format: SYS (SET_TIMER,N).

   @N: Time number. N is between 0 and 7.
   @N+1: Current Timer Value.
   @N+2: Timer Limit.
   @N+3:Time-up Flag.
   @N+4: Type of Operation as below:

   0 Timer will stop when it reaches the default setting, the flag will be set to 1.

   1 Timer resets to 0 automatically when the flag is changed to 0 or 1. When the flag is 1, the timer resets to 0 automatically. When the flag is 0, the timer resets to 0 automatically.

   STOP_TIMER → Stops the internal timer.
   Format: SYS (STOP_TIMER,N).
2. **STOP_COUNTER** → Stop the internal counter.
   Format: SYS(STOP_COUNTER,N).
3. **WAIT_TIMER** → Wait for the time-up event in the internal timer.
   Format: SYS (WAIT_TIMER,N).
   The macro instruction following this command will not be executed until the timer reaches the Timer Limit. Remember that the corresponding timer must be activated by the **SET_TIMER** service before requesting this service.

4. **INIT_COM** → Select and initialize a COM port.
   Format: SYS (INIT_COM,N).
   The word @n specifies the communication setting of the COM port. The format of the setting is shown below:

   Bit 1, Bit 0 → DATA Bit S 10:7 Bit S, 11:8 Bit S.

   Bit 2 → STOP Bit S 0:1 Bit,1:2 Bit S.

   Bit 4, Bit 3 → PARITY.> 00:NONE, 01:ODD, 11:EVEN.

   Bit 6, Bit 5 → COM PORT > 00: COM1, 01: COM2, 10: COM3, 11: COM4.

   Bit 7 → Not used.

   Bit11, Bit 10, Bit 9, Bit 8 → 0001: 115200, 0010: 57600, 0011: 38400, 0110: 19200, 1100: 9600, Others: 4800.

   Bit 15 → Computer Protocol Driver; 0: Disable, 1: Enable

   If this service is successful, the word @n+1 will be set to 1; otherwise, it will be set to 0.
   Some models provide a Computer Protocol slave driver for the second COM port. This function provides communication between PC/another operator terminal on the second COM port. The operator terminal can communicate with the controller over the first COM port and the connection steps are the same as the normal steps. The P PC can read from the internal registers @0-@8191 (the data in W0-W8191 corresponds to @0-@8191).

   When using the Computer Protocol driver, this function is unavailable to GET_CHAR, GET_CHARS, PUT_CHAR, and PUT_CHARS.

5. **GET_CHAR** → Gets a character from the COM port.
   Format: SYS (GETCHAR,N).
   The character will be saved in the low byte of the word @n. If there is no input, the word @n will be set to be -1(ffffH).

6. **GET_CHARS** → Gets a number of characters from the COM port.
   Format: SYS (GETCHARS,N).
   The word @n specifies the maximum number of characters to receive. The actual number of characters received is saved in word @n+1. The characters received will be saved in the low bytes of the words @n+2, @n+3, @n+4, and so on.

7. **PUT_CHAR** → Sends a character in the low byte of the word @n to the COM port.
   Format: SYS (PUTCHAR,N).
   If this service is successful, the word @n+1 will be set to 1; otherwise, it will be set to another value.

8. **PUT_CHARS** → Sends the characters in the low bytes of the words starting from @n+2 to the COM port.
   Format: SYS (PUTCHARS,N).
   The word @n specifies the number of characters to be sent and the actual number of characters sent is saved in the word @n+1.

9. **SUM_ADD** → Calculates the sum of a block of words by normal arithmetic addition.
   Format: SYS (SUM_ADD,N).
   The output data is saved in "@N+3". This feature offers a more convenient application for macros. For example, the command SYS(SUM_ADD,30) (here N = 30) will calculate the sum of the @30, @31, @32, @33 internal registers.

   @N=30 represents the pointer parameter, and the internal value of @30 must be 0.
   @N+1(@31) represents the starting address of the block.
   @N+2(@32) represents the size of the WORDS block.
   @N+3(@33) represents the initial value of the summand and the sum will be saved in this address automatically. The command must be set before execution. Most communication protocols regulate the initial value of the summand = 00H or FFH, so please refer to initial value assigned by the vendor.

10. **SUM_XOR** → Calculate the sum of a block of words by the bit-wise logical exclusive-or operation and save the result in the specified address.
    Format: SYS(SUM_XOR,N).
    The output data will be saved in "@N+3". This function is convenient for macro communication applications. For example, SYS(SUM_XOR,50) (here N = 50) will calculate the sum of the @50, @51, @52, @53 internal registers. Execution of this command requires the internal values of @50, @51, @52 and @53.

    @N=50 represents controller station number and the internal value of @50must be 0 if no controller station is required.
    @N+1(@51) represents the starting address of the block.
    @N+2(@52) represents the size of the WORDS block.
    @N+3(@53) represents the initial value of the summand and the sum will be saved in this address automatically. The command must be set before execution. Most communication protocols regulate the initial value of the summand = 00H or FFH, so please refer to initial value assigned by the vendor.

11. **READ_WORDS** → Read a number of words from controller word devices or internal memory and save the result in the specified address.
    Format: SYS (READ_WORDS,N).
    The data will be saved in "@+5". This command is powerful for use in communication with any controller registers and can be used for setting and monitoring controller data. Take, for example, SYS(READ_WORDS,80) (here N = 80). Execution of this command requires the internal values of @80, @81, @82, @83, @84, @85 and @86.

    @N (@80) represents the controller station number, and the internal value of @80 must be 0 if no controller station is required.
    @N+1(@81) represents the device type setting. For the device type of controller, please see the driver help file for full details.
    @N+2(@82) represents the low word of the device address.
    @N+3(@83) represents the high word of the device address.
    @N+4(@84) represents the auxiliary address if required else set to 0.
    @N+5(@85) represents the address of the internal memory to receive the data and the size of data is specified by N+6(@86).
    @N+6(@86) represents the number of words to be read.

12. **READ_Bit** → Read a controller bit device or internal bit and save the data in the specified address.
Format: SYS (READ_Bit,N).
The data will be saved in "@+5". This command is powerful for use in communication with any controller bit-state and can be used to set and monitor controller data. For example, SYS(READ_bit,80) (here N = 80). Execution of this command requires the internal values of @80, @81, @82, @83, @84 and @85.

    @N(@80) represents the controller station number, and the internal value of @80 must be 0 if no controller station is required.
    @N+1(@81) represents the device type. For controller device types, please see the driver help file for full details.
    @N+2(@82) represents the low word of the device address.
    @N+3(@83) represents the high word of the device address.
    @N+4(@84) represents the auxiliary address if required else set to 0.
    @N+5(@85) represents the address of the internal memory to receive the data.
    N+5(@85) DATA = 1 if the bit is ON; DATA = 0 if the bit is OFF.

13. **WRITE_WORDS** → Writes a block of data in internal memory to controller word devices or internal memory.
Format: SYS(WRITE_WORDS,N).
The data will be saved in "@N+5". This command is powerful for the random modification of controller data and can be used to set and monitor controller data. For example, SYS(WRITE_WORDS,90) (here N = 90). Execution of this command requires the internal values of @90, @91, @92, @93, @94, @95 and @96.

    @N(@90) represents the controller station number, and the internal value of @90 must be 0 if no controller station is required.
    @N+1(@91) represents the device type. For controller device types, please see the driver help file for full details.
    @N+2(@92) represents the low word of the device address.
    @N+3(@93) represents the high word of the device address.
    @N+4(@94) represents the auxiliary address if required else set to 0.
    @N+5(@95) represents the source address while the size of the continuous block of data is assigned by N+6 (@96).
    @N+6(@96) represents the number of words in the data block.

14. **WRITE_Bit** → Set a controller bit device or internal bit to the state of an internal word.
Format: SYS(WRITE_Bit,N).
The source address is"@+5". This command is powerful for the random modification of controller data and can be used to set and monitor controller data. For example, SYS(WRITE_Bit,90) (here N = 90). Execution of this command requires the internal values of @90, @91, @92, @93, @94 and @95.

    @N(@90) represents the controller station number, and the internal value of @90 must be 0 if no controller station is required.
    @N+1(@91) represents the device type. For controller device types, please see the driver help file for full details.
    @N+2(@92) represents the low word of the device address.
    @N+3(@93) represents the high word of the device address.
    @N+4(@94) represents the auxiliary address if required else set to 0.
    @N+5(@95) represents the address of the internal memory to receive the data.
    N+5(@95) DATA = 1 if the bit is ON; DATA = 0 if the bit is OFF.

8.4 Cautions

The last line of code must be the **RET** command, otherwise an error will occur when you compile.

Except in sub-macro, the **END** command marks the end of the macro.

The CPU will execute other programs after the execution of INITIAL Macro, CLOCK Macro, ON/OFF Macro, OPEN Macro and CLOSE Macro.

For BACKGROUND Macro, CYCLIC Macro and sub-macro, the CPU executes the 30 command lines once. The CPU will then execute other programs. The CPU will execute the 30 command lines following the last executed command until the next cycle.

To use the macro communication function, you must define the related communication format for INICOM. This command is only used once, so it is usually entered in INITIAL Macro.

8.5 Internal Memory

The operator terminal provides some internal registers that can be read from/written to. By using these internal registers, you can make more efficient and convenient use of the macro function. The internal registers can not only enhance macro with infinite functions, but can also store a great deal of arithmetic source data and results. Note that this system provides the internal registers divided into RAM and ROM.

Details of the four types of internal memory follow:

Words	Device Type	Size	Address	Aux. Address	R/W
RCPNO	0x81	W	0 (only one word)	0	R/W
RCPWn	0x82	W	0 - 8191	0	R/W
CBn	0x83	W	0 - 31	0	R
@n	0x85	W	8191	0	R/W

Bits	Device Type	Address	Aux. Address	R/W
CBn.b (b=0-f)	0x83	0 -31	0 - 15	R/W
RCPWn.b (b=0-f)	0x84	0 - 8191	0 - 15	R
@n.b (b=0-f)	0x86	8191	0 - 15	R/W

1. RCPNO.
2. The n value of RCPWn is based on the size of the recipe and the maximum number. The data register can be used as bit.
3. The n value of CBn is based on the size of the control block. The current size is 2-32. This data register can be used as bit.
4. @n: Internal Register. The size is 8191 WORDS (n=0-8191). This data register can be used by bit.

9 Ladder Programming

Ladder logic is a method of drawing electrical logic schematics. It can also be used for programming controllers. A program in ladder logic, also called a ladder diagram, is similar to a schematic for a set of relay circuits.

H-series operator terminals and H-Designer support ladder programming and functions in a continuous, cyclical series of tasks called scan. When the operator terminal is running, your program should be in one of the following status: RUN, STOP, PAUSE, and ERROR, and you can control the status changing by using debug commands and inserting instructions in your program.

Detailed information about using ladder programming as well as examples are available by selecting **Help/LadderPlus** in H-Designer.

10 Appendix A - H-Designer Features and Operator Terminal Models

The following table summarizes the H-Designer features and operator terminal models.

H-Designer Feature	H-K30m-S	H-T40m-S	H-T40m-PA	H-T50b-S	H-T60b-S	H-T60b-Pe	H-T60b-Ne
Ethernet	No	No	No	No	No	No	Yes
Printer	No	No	No	No	No	Yes	Yes
Upload Application	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Upload/Download Recipes	No	No	Yes	No	No	Yes	Yes
Reconstruct Source	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Contrast Adjustment	No	Yes	Yes	Yes	Yes	Yes	Yes
Turn off Backlight	No	Yes	Yes	Yes	Yes	Yes	Yes
Set Time & Date	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Character Entry	No	No	No	Yes	Yes	Yes	Yes
List and Drop-down List	No	Yes	Yes	Yes	Yes	Yes	Yes
Numeric Display	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Time, Date, and Day of Week Display	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Historical Display	No	No	No	Yes	Yes	Yes	Yes
Alarm Display	No	Yes	Yes	Yes	Yes	Yes	Yes
Macro	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Ladder	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Multi-Link (One master, Slaves)	Slave	Slave	Slave	Yes	Yes	Yes	Yes
Cross-Link (Mutual Read)	No	No	No	No	No	No	Yes
Multi-Channel Communication	No	No	No	Yes	Yes	Yes	Yes
Logging Buffer	No	No	Yes	Yes	Yes	Yes	Yes
Common Key	Yes	No	No	No	Yes	Yes	Yes
Slide-out Menu	No	No	No	No	Yes	Yes	Yes
System Messages	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Off-line/On-line Simulation	Yes	Yes	Yes	Yes	Yes	Yes	Yes
View/Edit Recipes	No	No	Yes	No	Yes	Yes	Yes

H-Designer Feature	H-T60t-S	H-T60t-Pe	H-T60t-Ne	H-T70t-Ne	H-T80c-Ne	H-T100t-Ne
Ethernet	No	No	Yes	Yes	Yes	Yes
Printer	No	Yes	Yes	Yes	Yes	Yes
Upload Application	Yes	Yes	Yes	Yes	Yes	Yes
Upload/Download Recipes	Yes	Yes	Yes	Yes	Yes	Yes
Reconstruct Source	Yes	Yes	Yes	Yes	Yes	Yes
Contrast Adjustment	No	No	No	No	Yes	No
Turn off Backlight	Yes	Yes	Yes	Yes	Yes	Yes
Set Time & Date	Yes	Yes	Yes	Yes	Yes	Yes
Character Entry	Yes	Yes	Yes	Yes	Yes	Yes
List and Drop-down List	Yes	Yes	Yes	Yes	Yes	Yes
Numeric Display	Yes	Yes	Yes	Yes	Yes	Yes
Time, Date, and Day of Week Display	Yes	Yes	Yes	Yes	Yes	Yes
Historical Display	Yes	Yes	Yes	Yes	Yes	Yes
Alarm Display	Yes	Yes	Yes	Yes	Yes	Yes
Macro	Yes	Yes	Yes	Yes	Yes	Yes
Ladder	Yes	Yes	Yes	Yes	Yes	Yes
Multi-Link (One master, Slaves)	Yes	Yes	Yes	Yes	Yes	Yes
Cross-Link (Mutual Read)	No	No	Yes	Yes	Yes	Yes
Multi-Channel Communication	Yes	Yes	Yes	Yes	Yes	Yes
Logging Buffer	Yes	Yes	Yes	Yes	Yes	Yes
Common Key	Yes	Yes	Yes	Yes	Yes	Yes
Slide-out Menu	Yes	Yes	Yes	Yes	Yes	Yes
System Messages	Yes	Yes	Yes	Yes	Yes	Yes
Off-line/On-line Simulation	Yes	Yes	Yes	Yes	Yes	Yes
View/Edit Recipes	Yes	Yes	Yes	Yes	Yes	Yes