

Win Studio/InduSoft Web Studio Technical Reference Manual

Version **101**

Win Studio/InduSoft Web Studio Technical Reference Manual

1070 072 272-101 (01.11) GB

© 2001

by Robert Bosch GmbH,
All rights reserved, including applications for protective rights.
Reproduction or handing over third parties are subject to our written
permission.

price 18,- €

1 Contents

page

| | | |
|--------|---|-------|
| 1 | Contents | I |
| 2 | Introduction | 2-1 |
| 2.1 | Related Manuals | 2-2 |
| 2.2 | Text Conventions Used in this Manual | 2-3 |
| 2.3 | Mouse and Selection Conventions | 2-4 |
| 2.4 | Windows Conventions | 2-5 |
| 2.5 | System Requirements | 2-6 |
| 2.6 | Main Features | 2-7 |
| 2.7 | InduSoft Web Studio Software Installation | 2-9 |
| 2.8 | Uninstalling InduSoft Web Studio | 2-11 |
| 2.9 | Starting InduSoft Web Studio | 2-13 |
| 3 | Development Environment | 3-1 |
| 3.1 | Titlebar | 3-2 |
| 3.2 | Status Bar | 3-3 |
| 3.3 | Menu Bar | 3-4 |
| 3.3.1 | File Menu | 3-4 |
| 3.3.2 | Edit Menu | 3-7 |
| 3.3.3 | View Menu | 3-9 |
| 3.3.4 | Insert Menu | 3-13 |
| 3.3.5 | Project Menu | 3-17 |
| 3.3.6 | Tools Menu | 3-21 |
| 3.3.7 | Window Menu | 3-23 |
| 3.3.8 | Help Menu | 3-24 |
| 3.4 | Toolbars | 3-25 |
| 3.4.1 | Standard Toolbar | 3-25 |
| 3.4.2 | Tag Properties Toolbar | 3-27 |
| 3.4.3 | Execution Control Toolbar | 3-28 |
| 3.4.4 | Web Toolbar | 3-30 |
| 3.4.5 | Align and Distribute Toolbar | 3-30 |
| 3.4.6 | Mode Toolbar | 3-34 |
| 3.4.7 | Bitmap Toolbar | 3-36 |
| 3.4.8 | Static Objects Toolbar | 3-37 |
| 3.4.9 | Dynamic Properties Toolbar | 3-41 |
| 3.4.10 | Active Objects Toolbar | 3-49 |
| 3.5 | Workspace | 3-61 |
| 3.5.1 | Database Tab | 3-62 |
| 3.5.2 | Graphics Tab | 3-86 |
| 3.5.3 | Task Tab | 3-93 |
| 3.5.4 | Communication Tab | 3-114 |
| 3.6 | Translation Tool | 3-126 |

| | page |
|--------|---|
| 3.7 | Functions List..... 3-129 |
| 3.7.1 | Send Message to the LogWin 3-135 |
| 3.7.2 | Arithmetic Functions 3-135 |
| 3.7.3 | Statistic Functions 3-140 |
| 3.7.4 | Logarithmic Functions 3-142 |
| 3.7.5 | Logic Functions 3-143 |
| 3.7.6 | Functions for Strings Manipulation 3-144 |
| 3.7.7 | Date and Time Manipulation..... 3-149 |
| 3.7.8 | Trigonometric Functions 3-151 |
| 3.7.9 | Functions for Opening and Closing Windows 3-153 |
| 3.7.10 | Security System 3-154 |
| 3.7.11 | Module Activation Functions..... 3-156 |
| 3.7.12 | File Manipulation Functions 3-168 |
| 3.7.13 | Functions for Graphics Screens Printing..... 3-172 |
| 3.7.14 | Functions for Text Translations 3-172 |
| 3.7.15 | Multimedia Functions..... 3-173 |
| 3.7.16 | System Information..... 3-173 |
| 3.7.17 | Database Access Functions..... 3-178 |
| 3.7.18 | Loops 3-178 |
| 3.7.19 | ODBC Functions..... 3-179 |
| 3.7.20 | MAIL Functions..... 3-191 |

2 Introduction

InduSoft Web Studio™ is a powerful tool for building full-featured SCADA (Supervisory Control And Data Acquisition) or HMI (Human-Machine Interface) applications for Industrial Automation that exploits the key features of Microsoft® Windows® NT/2000/CE.

⇒ **Please note that the Bosch Win Studio is based on the InduSoft Web Studio™ and comes with additional drivers for Bosch applications. All descriptions in this manual which refer to InduSoft Web Studio™ are valid for Bosch Win Studio as well.**

The application consists of animated operator-interface screens, drivers (configurable for PLCs or other I/O devices to be controlled), a database of application tags, and optional modules such as alarm logic, trend charts, recipes, schedulers, and a security system. The Web Studio application interfaces with industrial I/O systems and other Windows applications in the runtime environment using ODBC, DDE, NetDDE, OPC, or TCP/IP protocols.

The product consists of two parts:

- Development system: software running on a desktop, laptop, or industrial PC with Windows® NT/2000
- Runtime system: software running on an operator interface workstation with Windows® NT/2000 or Windows® CE. The runtime software (CEView) for the Windows® CE operating system is usually preloaded on the HMI. With the development system, you can update the CEView version, by downloading it to the HMI, when necessary.

This *Win Studio/InduSoft Web Studio Technical Reference manual* is designed for all InduSoft Web Studio users. The chapters are organized to help you quickly find information on any aspect of the software.

⇒ **This manual assumes that you are familiar with the Windows environment. If you are not, we suggest that you select Help from the Windows desktop Start menu before you continue to work through this guide.**

2.1 Related Manuals

Tutorial Manual

Describes how to build an application, step-by-step, with the main product features. You can use this document as a self-training manual. This tutorial is stored in the \Documentation folder on the Win Studio CD.

Drivers User Guides

Includes one Driver User Guide for each InduSoft driver. These User Guides describe the customized configuration of each driver, according with its protocol characteristics.

⇒ **The product manuals can be found in the *Documentation* folder on the Win Studio CD. The drivers User Guides are stored in the \DRV sub-directory of the InduSoft Web Studio folder just after its installation. You also can access technical information by selecting the Help menu option from the development environment.**

2.2 Text Conventions Used in this Manual

Throughout this manual the text of certain terms are formatted in ways to indicate the type of object being described. Also, some information is segregated from the main text to help you to read through this manual quickly.

- Titles, labels, and messages (such as *Object Properties*) are indicated using italic text.
- Computer filenames and text to be entered by you (such as ***d:\Setup.exe***) are indicated using bold italic text.
- Specific items that require operator input (such as the Start menu button), menu options, and keyboard keys (such as Enter) are indicated using a narrow bold typeface.
- Text requiring emphasis is bolded to **draw your attention** to the item.

Some text is segregated into ♦ **instructions for use**, ⇒ **Note** and ⚠ **Caution** boxes.

- **instructions for use** describe an activity which you will be required to perform.
- **Notes** contain extra and useful information that may make it easier to understand the nearby text, especially the text just before the note, to save development time or to improve the application performance.
- **Cautions** contain information necessary to prevent errors that can cause problems when running the application, and may result in damage.

2.3 Mouse and Selection Conventions

A mouse isn't the only pointing device; there are also keystrokes, touch-screens, etc. However, most PCs used for application development will be running a version of Windows with a mouse, so this manual is written assuming you are using a mouse. Generally, a PC mouse is configured so that the left mouse button is the primary button and the right mouse button is the secondary button. This manual uses the following mouse and selection conventions:

- *Double-click* means quickly click on an object twice with the left mouse button.
- *Right-click* means to click on an object with the right (secondary) mouse button.
- *Click* and *select* means to click on an object with the left (primary) mouse button.
- *Select* is also used when you should use your pointing device to highlight or specify an item on the computer screen.. Selection with a touch-screen is usually the same as selection with a mouse, except that you use your finger to touch a screen object or section. Selection with a keyboard usually requires you to use the Tab key to move around options, using the Enter key to open menus and to replace a Double-click, and using the Alt key with an underlined letter to select an object that has an underlined letter.
- *Dragging* means to press the appropriate mouse button after clicking and moving the mouse. Usually an outline of the object will move with the mouse pointer, or the outline of the shape will be defined by the movement of the mouse.

2.4 Windows Conventions

This manual uses the following Windows conventions:

- *Dialogs*, or *dialog boxes*, are windows that allow you to input information.
- *Text boxes* are spaces in windows where you can type in text.
- *Radio buttons* are white circles in which a black dot appears or disappears when clicked by the mouse.
- *Check boxes* are white squares in which a check appears or disappears when clicked by the mouse.
- *Buttons* are icons in boxes that appear to be pressed when selected.
- Lists are panes (white boxes) in windows that contain many selectable options.
- *Drop-down lists* have arrows that, when clicked, show part or all of an otherwise concealed list.
- In this manual, *interface* refers to the entire InduSoft Web Studio window.
- *Dockable windows* are windows that you can drag to an edge of the interface and merge with that edge.
- *Toolbars* are dockable windows that contain only *buttons* and text boxes.

2.5 System Requirements

To develop an InduSoft Web Studio application, we recommend the following hardware and software:

- IBM-compatible computer with an Intel® Pentium II-compatible processor
- Windows NT/2000 operating system

⇒ **The dialog box and procedure described in this manual are valid for Windows NT v4.0. Some terms may vary according to the operating system (type, language and version) you are using.**

- Minimum of 32 MB of random-access memory (RAM) - 64 MB or higher recommended
- MS Internet Explorer 4.0 or install 40comupd.exe version 4.71 or higher
- 150 MB of free hard disk space (required for the program without any application programs--more space is recommended)
- 3.5" floppy drive
- CD-ROM drive (drive can be on a different computer)
- Standard keyboard with function keys F1 through F12
- Parallel printer port (optional)
- 100% IBM-compatible VGA or SVGA display adapter with 2 MB Video RAM (VRAM)
- Microsoft-compatible pointing device (such as a mouse, trackball, joystick, or touch-screen)
- One or two COM ports and adapters for downloading applications (optional)
- Ethernet connection for downloading applications (optional)

2.6 Main Features

The following features are supported for InduSoft Web Studio product:

- Integrated Windows development environment with toolbars, dialogs, and menus:
 - Drop-down (pop-up) menus activated by a right-click on any area of the development environment. Includes options that will vary according to the context.
 - Flying toolbars that you can customize individually.
 - Tasks, objects, and controls organized in a "tree-view" explorer.
- Full-featured objects and dynamics used to build screens:
 - Configurable objects; such as buttons, rectangles, ellipse, polygons, lines, and texts.
 - Dynamic properties; such as bar graph, color, resize, position, hide/unhide, rotation, command, hyperlink, and text Input/Output.
 - On Line and Historical alarm list display
 - On Line and Historical trending
 - Alignment and distribution tools
 - Background bitmap layer creation and editing
 - Graphics importation
 - Active-X object containers
- On-line remote Management and Configuration.
- Microsoft DNA architecture compliant, with full support to OPC and XML.
- Web interface enabled: exports application screens to thin client by Internet/Intranet and exchanges data on-line by TCP/IP protocol.
- Object library with more than 100 symbols and dynamic objects, such as pushbuttons, meters, sliders, switches, text and numeric displays, LED-style indicators, pipes, bumps, icons, vehicles, valves, frames, motors, gauges, common controls, etc.
- Debugging tools:
 - Database Spy window to monitor and force tag values as well as execute functions.
 - LogWin module to record OPC, DDE, and TCP/IP transactions, modules activation, trace tags, etc.
 - Cross-referencing to locate tags throughout the project.
 - On-line system and network diagnostics.
- Powerful and flexible tags database (array tags, indirect tag-pointers, classes, Boolean, integer, real and string tags).
- Open architecture - API exchanges tag values with external software.
- Translation editor, which enables you to translate the same application into several different languages, even while the runtime is online.
- TCP/IP Client and Server modules to exchange tag values and configure redundancy systems.
- More than 200 drivers for different devices (such as PLC) from several manufactures, such as Allen-Bradley, Siemens, GE-Fanuc, as well as standard protocols like MODBUS RTU/ASCII, DeviceNet, Profibus, Interbus, etc.
- OPC Client with integrated OPC Browser.
- Screen and object password-protected runtime security (256 levels).
- Logical expressions and a scripting language with more than 200 functions.

- Recipe and Report (ASCII and RTF format) builders integrated in the product.
- Event scheduler can be based on date, time, or data condition (100ms resolution).
- Multi-layer application (modular worksheets and screens are merged easily to other applications).
- Full integration with PC-based control packages (imports tags database) - ISaGRAF, SteepleChase, Think&Do, ASAP, etc.
- Real time project documentation.
- Screens resolution converter.

⇒ **InduSoft Web Studio provides different product types for each level of application responsibility. Some features are not supported for several product types (such as CEView). Check the *TargetVersions.pdf* document on the InduSoft Web Studio CD-ROM for detailed information about the limitations of each product type.**

2.7 InduSoft Web Studio Software Installation

InduSoft Web Studio runs on the Microsoft Windows® NT/2000 operating system. The installation program creates directories, as needed, copies files to your hard drive, and creates the *InduSoft Web Studio* icons in a desktop folder.

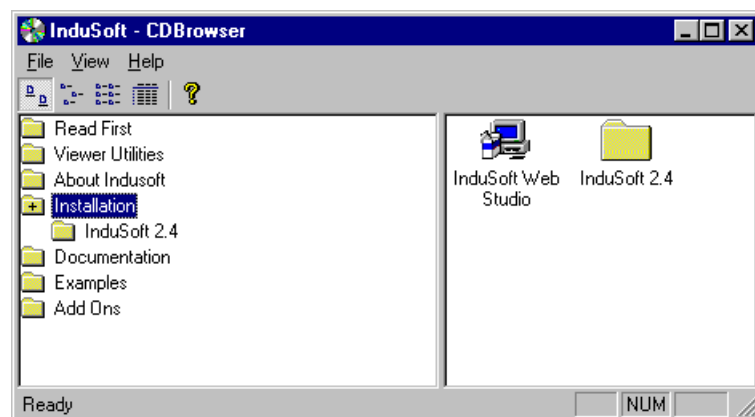
InduSoft Web Studio is packaged on a CD-ROM. You can install the program from this CD-ROM or create 3.5" installation floppy disks.

InduSoft Web Studio provides development tools for all InduSoft applications. For Windows CE applications, you can use InduSoft Web Studio to download CEView (runtime software) to the Windows CE HMI by serial or TCP/IP link.

⇒ **You must have Administrator privileges on a Windows NT workstation to install any software, including InduSoft Web Studio. You can install a newer version of InduSoft Web Studio over an older version. However, we recommend that you uninstall the older version first.**

Use the following procedure to install InduSoft Web Studio:

1. Power up the development computer (Windows NT/2000) and be sure that no programs are running.
2. Insert the installation CD-ROM into the CD-ROM drive or insert the first installation floppy disk into the 3.5" disk drive.
3. A **CD Browser** window displays. If the **CD Browser window** doesn't open automatically, you can start it manually in Windows Explorer. Execute the **Setup.exe** file from the **d:\Installation** directory (where *d* is your CD-ROM driver unit).



4. Select the *Installation* folder and double-click on the *InduSoft Web Studio* icon to launch the installation wizard.
5. A **Setup** dialog box will inform you that the **InstallShield® Wizard** is loading, then the first installation window will prompt you to follow instructions on the screen to proceed with the installation.

6. When prompted to restart Windows, select the **Yes, I want to restart my computer now** radio button and select OK.
7. After your computer restarts, go to *Starting InduSoft Web Studio*, chapter 2.9.

⇒ You can install InduSoft Web Studio from the CD-ROM or create installation 3.5" floppy disks. To create the 3.5" floppy disks, copy the contents from the *d:\Installation\Disk1* folder (where *d* is the CD-ROM drive unit) to floppy disk #1. The contents of *d:\Installation\Disk2* go on floppy disk #2 and so on. To install *InduSoft Web Studio* from these disks, insert floppy disk #1 in the floppy driver and execute the *Setup.exe* file. Follow the installation wizard instructions.

2.8 Uninstalling InduSoft Web Studio

If you find it necessary to remove *InduSoft Web Studio* from your system, follow these instructions:

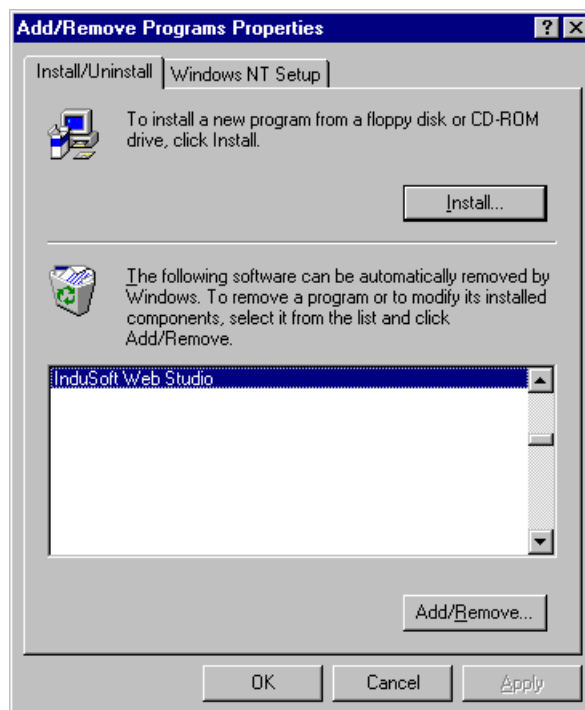
1. Select the Start button on the Taskbar, select Settings, and then select Control Panel to open the Control Panel window.



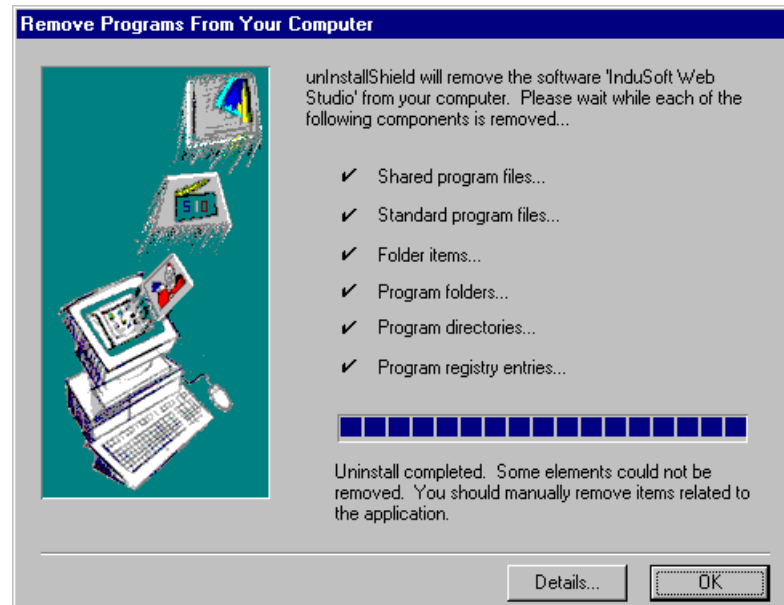
2. Double-click on the Add/Remove Programs icon in the *Control Panel* window.



3. In the Add/Remove Programs Properties window select *InduSoft Web Studio* in the lower pane list and then click on the Add/Remove... button.



4. At the Confirm File Deletion dialog, select the Yes button.
5. The Uninstall Shield Wizard and the Remove Programs From Your Computer dialog will open. When the Uninstall successfully completed message appears the OK button will become active. Select the OK button.



6. You will see that InduSoft Web Studio is no longer listed in the lower pane of the *Add/Remove Programs Properties* window. Close the window using the Cancel button or the close button (X), and then close the *Control Panel* window.
7. Open the Windows Explorer and browse to the directory that contained the InduSoft Web Studio directory.
8. Ensure that all of the InduSoft Web Studio files and folders were deleted. Manually delete any that are left.



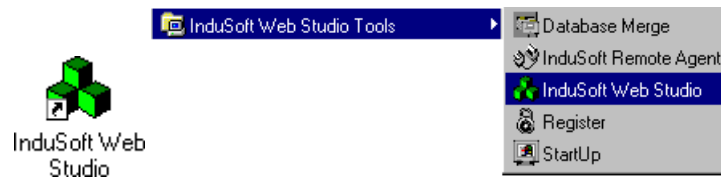
CAUTION

Before you start the uninstall procedure, be sure that you back-up all of the files in the *\InduSoft Web Studio* folder that may be useful in the future. Also, be sure that you have the InduSoft Web Studio installation CD-ROM (or floppy disks) so you can re-install the software (new or same version) if necessary and that they are not damaged.

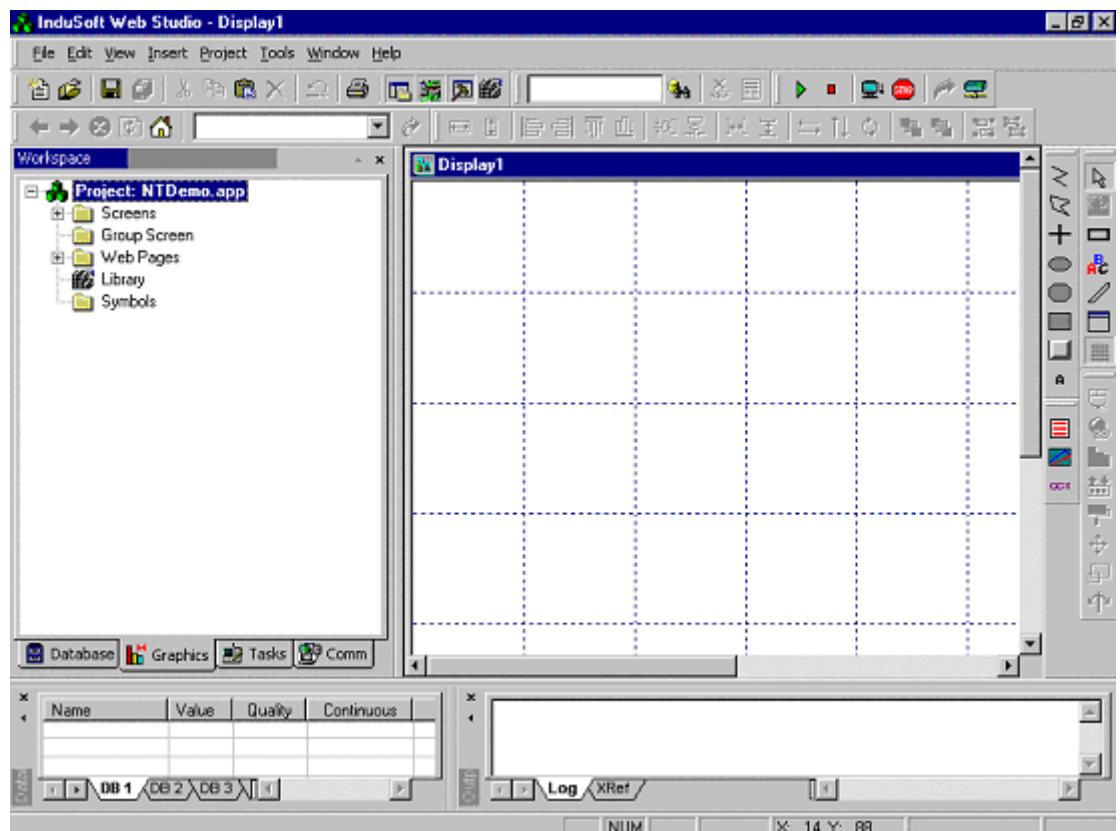
⇒ The files created or modified in the *\InduSoft Web Studio\Projects* folder are not deleted automatically by the uninstall tool.

2.9 Starting InduSoft Web Studio

1. Double-click the *InduSoft Web Studio* shortcut icon on the desktop or select Programs from the Start menu, then select InduSoft Web Studio from the InduSoft Web Studio Tools submenu.



2. InduSoft Web Studio launches.

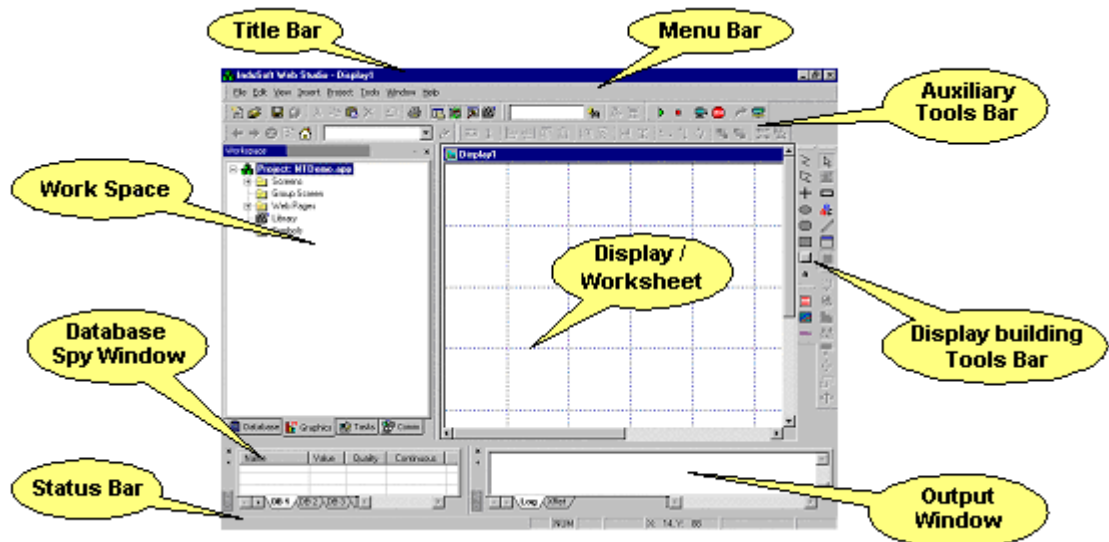


- ⇒ You can run the InduSoft Web Studio development environment under any video setting. However, it's recommended that you configure the video settings to resolution 800x600 (or higher) and more than 256 colors for a more pleasing environment. The application resolution (screen size) is independent of the operating system resolution.

3 Development Environment

InduSoft Web studio complies with the *Windows-like* view and adopts standard tools and interface to make the product user-friendly - even for new users.

The development environment is integrated and unique, for fast and easy access to any tools or information.



The development environment is composed of the following, basic areas:

- *Title bar*: Indicates the active display or worksheet.
- *Status Bar*: Provides quick access to actual information.
- *Menu bar*: Contains main product options and controls, which can be easily accessed by the pointer or by the keyboard.
- *Auxiliary Tool Bars*: Provides shortcuts to the main commands used in the development environment.
- *Displays Building Tool Bars*: Contains features and tools that you use to create and edit objects and dynamics in the application displays.
- *Workspace*: Provides tree-view control, from which you can access project worksheets and displays.
- *Database Spy Window*: Provides a debugging tool, which you can use to monitor/force tags and execute functions.
- *Output Window*: Window in which the debug messages are written.
- *Displays / Worksheets*: Provides an area where you can edit displays and worksheets.

⇒ **The previous picture shows the areas and windows in their default position. You can customize the development environment according to your needs by changing the position of each area described above can vary.**

3.1 Titlebar



The titlebar contains (from left to right):

- The InduSoft Web Studio or Bosch Win Studio icon and name.
- The name of the active, open screen or worksheet (if any).
- The Minimize button (☐): Press this button to minimize the InduSoft Web Studio window.
- The Resize/Maximize button (☐/☐): Press these buttons to toggle from one option to the other. The Resize button tiles the InduSoft Web Studio window and the Maximize button maximizes it.
- The Exit button (✕): Press this button to close , InduSoft Web Studio and automatically save the database. You will be prompted to save changed screens or worksheets. This button is similar to the Exit command in the File menu.

⇒ **Closing the Development System will not close the Runtime System.**

3.2 Status Bar



The status bar contains fields used to identify toolbar buttons and provide information about the active screen (if any). The fields are (from left to right):

- Hint field: Displays a short description of any toolbar button or display object touched by the cursor.
- Caps Lock field: Displays whether the keyboard Caps Lock is on (CAP) or off (empty).
- Num Lock field: Displays whether the keyboard Num Lock is on (NUM) or off (empty).
- Scroll Lock field: Displays whether the keyboard Scroll Lock is on (SCRL) or off (empty).
- ID field: Displays the ID number of a selected screen object.
- Screen Coordinate field: Displays the current location of the cursor on the active screen. Where X is the number of pixels from the left edge of the screen And Y is the number of pixels from the top of the screen.
- Object Size field: Displays the size of the selected object in pixels. Where W is the width and H is the height.
- No DRAG field: Displays whether dragging is disabled (No DRAG) or enabled (empty) in the active screen.

3.3 Menu Bar

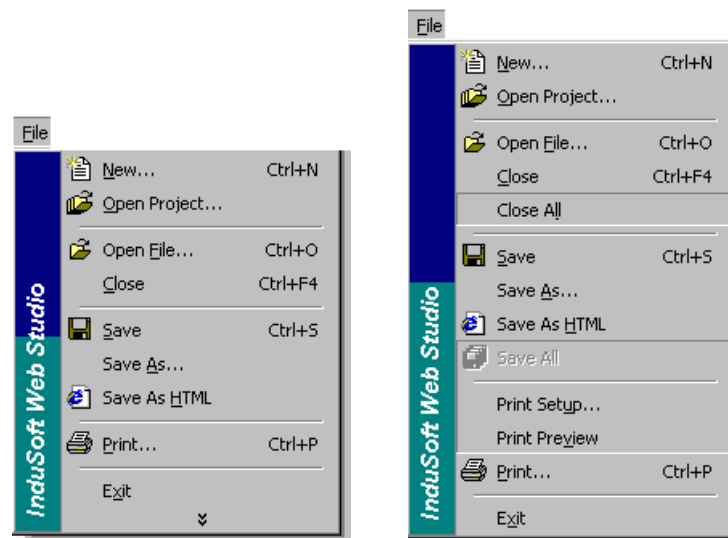



The menu bar contains File, Edit, View, Insert, Project, Tools, Window, and Help menus.

⇒ The menu bar is dockable. Right-click on the menu bar to open a pop-up menu. From this pop-up menu, you can make the menu bar visible or invisible.

3.3.1 File Menu

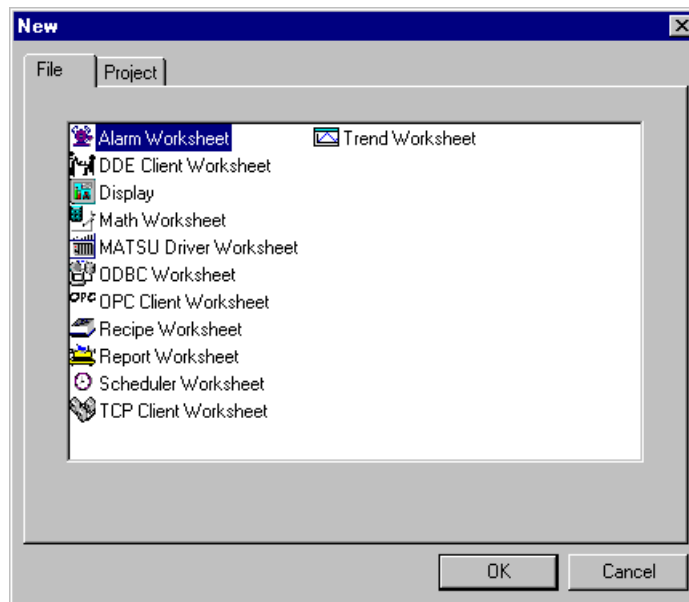
The File menu contains commands and tools to manage application files.







-  **New...**: Opens a New window containing File and Project tabs, which allow you to create a new application (project) or a new file that is part of your open application (Display, Math worksheet, etc). You also can open a New window by selecting the New button from the *Standard* toolbar or by choosing Document... from the Insert menu.

The **File** tab allows you to pick new *Alarm*, *DDE Client*, *Math*, *ODBC*, *OPC Client*, *Recipe*, *Report*, *Scheduler*, *TCP Client*, and *Trend* worksheets or a new *Display* screen. When you add an I/O driver to the application, there is an option that allows you to open a new driver worksheet. The **Project** tab allows you to create a new project.

⇒ **Worksheets for DDE Client and ODBC do not appear in Windows CE applications.**

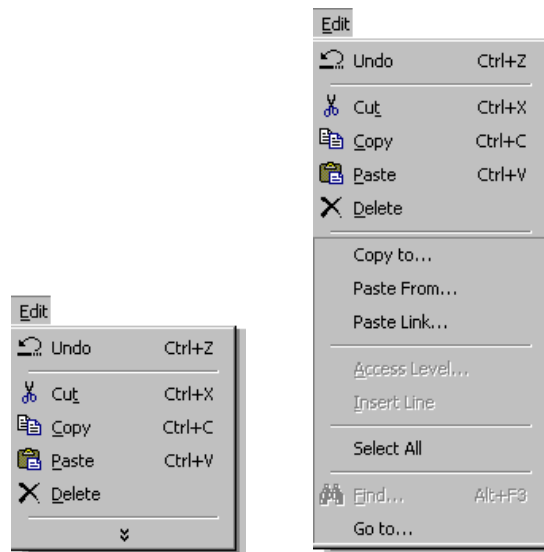







- **Open Project...**: Displays an **Open** window from which you can navigate to and open another InduSoft Web Studio application. You also can open projects from this window by double-clicking on a project name in a directory in Windows Explorer or by selecting the Open Project button on the *Standard* toolbar.
- **Open File**: Opens an InduSoft Web Studio application file. From the **Open** window, you can select a file type from the *Files of type* drop-down list and browse to the desired file.
- **Close**: Closes the active screen or worksheet. You will be prompted to save changes. This option is the same as the title bar close button ().
- **Close All**: Closes *all* active screens or worksheets. You will be prompted to save changes.
- **Save**: Saves any active and open worksheets or screens. You also can use the Save button on the *Standard* toolbar. The **Save** function is available only when you modify the active file.
- **Save As**: Saves the active worksheet or screen and allows you to choose the name and location of the file.

-  Save As HTML: Saves the active display in HTML format.
-  Save All: Saves all open worksheets or screens. You also can use the Save All button on the Standard toolbar. Save All is available only when something has been modified.
- Print Setup...: Allows you to configure the printing options. To set up a default printer, go to the Windows Start menu, select Settings, and then select Printers. Right-click the button on the printer you want to use as the default printer, and then select Set As Default on the pop-up menu. If a check displays next to Set As Default in the pop-up menu, then the selected printer is the default.
- Print Preview: This command is similar to the standard Windows *Print Preview* command. A Print Preview window opens in the workspace so you can see how the screen or worksheet will look when printed. The buttons along the top of the Print Preview window provide optional commands. You can Zoom In to check details and then Zoom Out to the default size. You can view the next page (Next Page), the previous page (Prev Page), or two pages at once (Two Page). At any time, you can Print... pages (which takes you to the same Print window as the Print... command) or you can Close the Print Preview window.
-  Print: Opens a **Print** window. You can print the display or worksheet in focus. In addition, you can specify the printer name, properties, and the number of copies you want to print. Also, you can print the current file by selecting the Print button on the *Standard* toolbar.
- Previous File List: Lists the four most recently opened files. Select the file to open it.
- Exit: Closes InduSoft Web Studio and automatically saves the database. You will be prompted to save any screens or worksheets with unsaved changes. This option is similar to the Exit button () in the titlebar.

3.3.2 Edit Menu

This menu contains commands and tools that allow you to manage screens and worksheet editions.



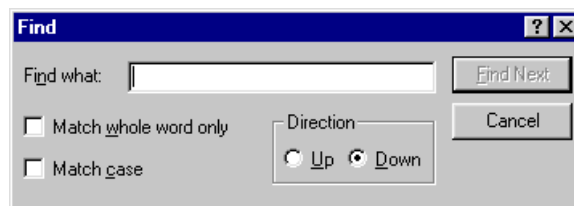
-  Undo: Cancels the last action performed while working on a screen. Cancels up to 20 actions taken prior to the current action. The actions in object properties do not increase Undo steps. You can use the Undo button on the *Standard* toolbar also.
-  Cut: Removes a selection and stores it on the clipboard, replacing any previously selections stored on the clipboard. You can use Cut to select an object and move it to another location on the screen or move it to another screen. You can use the Cut button on the *Standard* toolbar also.
-  Copy: Copies a selection to the clipboard and allows you to paste the selection to another location on the screen, paste it to a different screen, or make multiple copies of an object. You can use the Copy button on the *Standard* toolbar also.
-  Paste: Copies the contents of the Windows clipboard to the active screen. If the clipboard contains a selection, that selection is copied to the upper left corner of the screen. You can use the Paste button on the *Standard* toolbar also.
-  Delete: Deletes a selection. Use the Undo function to restore an object that is deleted accidentally. You can use the Delete button on the *Standard* toolbar also.
- Copy to...: Opens a **Save As** window and copies a selected element (an object or group of objects and its/their properties) to a file using an InduSoft Web Studio-specific format. These objects can have static and dynamic properties, as well as bitmap objects.

⇒ **The menu bar is dockable. Right-click on the menu bar to open a pop-up menu. From this pop-up, you can make the menu bar visible or invisible.**

- **Paste From...:** Imports a symbol, bitmap, or cut file to the current screen. The *.sym* files are objects with static and dynamic properties. The *.bmp* files are bitmaps (where the document was converted to a bitmap object). The *.cut* files are bitmaps (where the document was converted to a bitmap object).
- **Paste Link...:** Pastes a linked object into a screen, while maintaining a connection to the source. A linked object is information (the object) created in a source file (which can be another project or another screen). Automatically updates a linked screen object when you update the source. A linked object does not become part of the screen. Place the linked file in the application folder or a subfolder so that it can be downloaded with the application to the runtime workstation. This feature does not work in Windows CE applications.

⇒ **You can associate the linked picture (path and name) to a string tag value so it is possible to switch from one picture to another dynamically during runtime by modifying the string tag name.**

- **Access Level:** Allows you to set the security level for an active worksheet.
- **Insert Line:** Inserts a row into an active worksheet.
- **Select All:** Selects all objects on the active screen.
- **Find...:** Opens a **Find** window and allows you to find a word in the active worksheet.

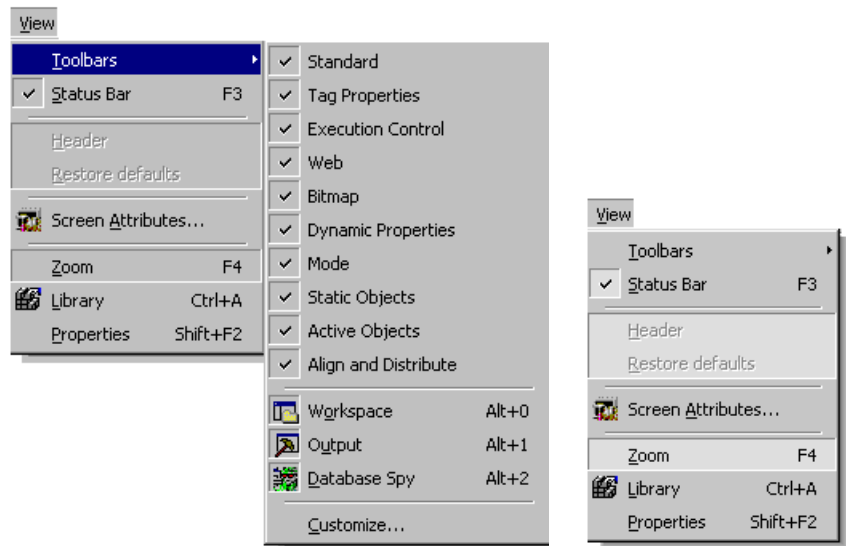


- **Go to...:** Jumps to a line in an open worksheet or selects an object on the screen by its ID number. InduSoft Web Studio applies a sequential identification number (ID) to each object created on the screen, starting with the number 0. When you select an object, the ID number displays in the **Status Bar**

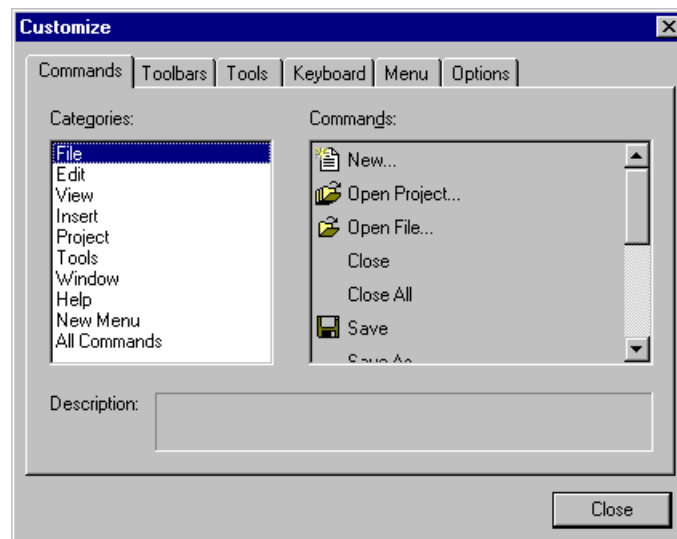
⇒ **If you have many superimposed objects, and it is not possible to select an object using the pointing device, you can use the Go to...: option to edit the properties of an object that is underneath other objects.**

3.3.3 View Menu

This menu contains commands that allow you to manage visible tools and it provides shortcuts to the dialog box that you open most frequently.

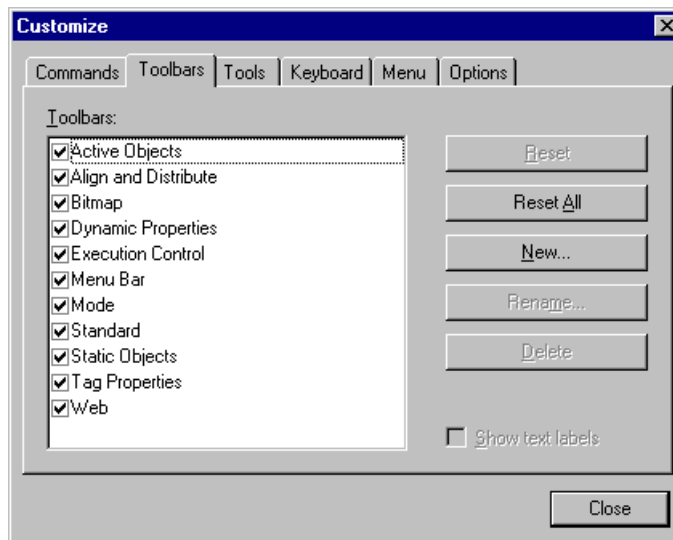






- **Toolbars:** Allows you to show/hide each tool bar, such as the *Workspace*, *Output* and *Database Spy* windows. Use the **Customize...** option from the **Toolbars** menu to customize the development environment appearance and open the *Customize* dialog box with the following tabs:
 - **Commands:** Customizes the menu options. You can select any command in the *Commands* list and drag it to any menu bar or to any tool bar in the development environment.

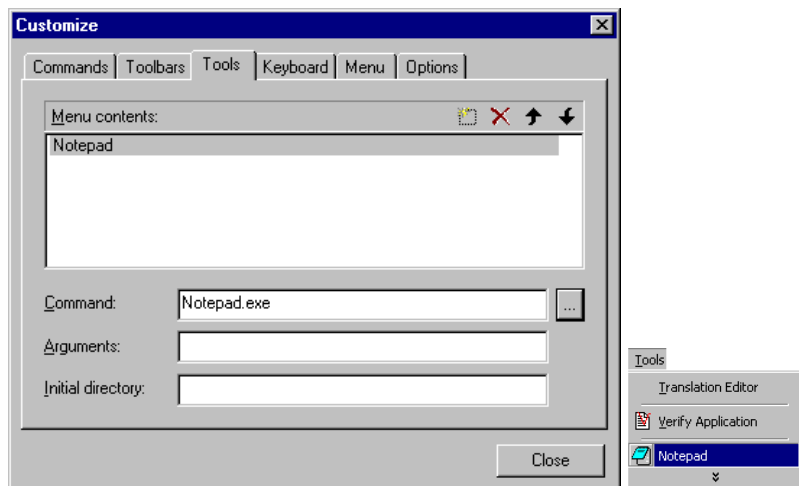


- **Toolbars:** Customizes the toolbars. You can configure any toolbar in the Toolbars list as visible or hidden. Use the **R**eset button to restore default settings for the selected toolbar. Use the **R**eset **A**ll button to restore the default settings for all toolbars. Use the **N**ew... button to create new toolbars.

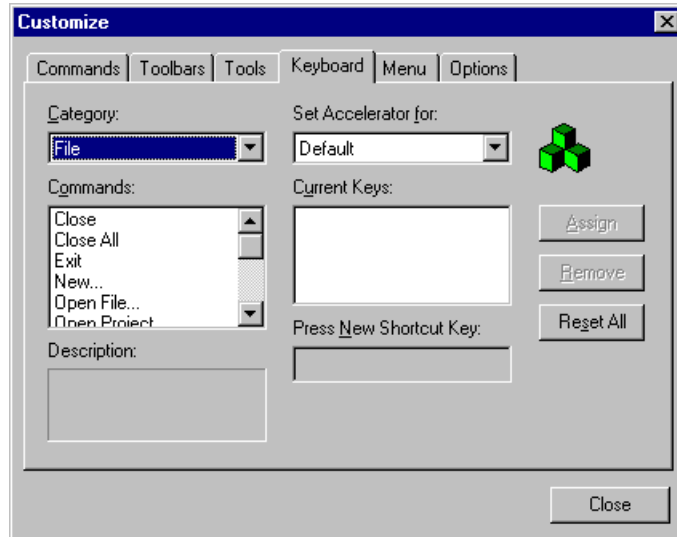
After creating a new toolbar, you can drag icons from the *Com*mands tab to the newly created toolbar. Use the **R**ename button to rename toolbars you created and you can use the **D**elete button to exclude toolbars you created. The **S**how text labels check box displays the icon's labels for the selected toolbar.



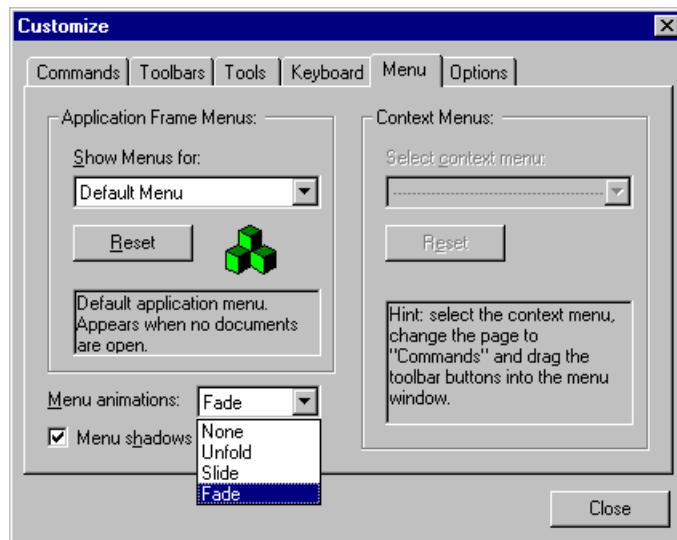
- **Tools:** Customizes the Tools menu options. You can create shortcuts to any external program and that program will be available in the Tools menu. To create a new shortcut, click on the  *New (Insert)* icon and configure the Command, Arguments and Initial directory for the shortcut. The  *Delete* icon button excludes the selected shortcut and the   *Move Item* icons shift the shortcut's position within the Tools menu.



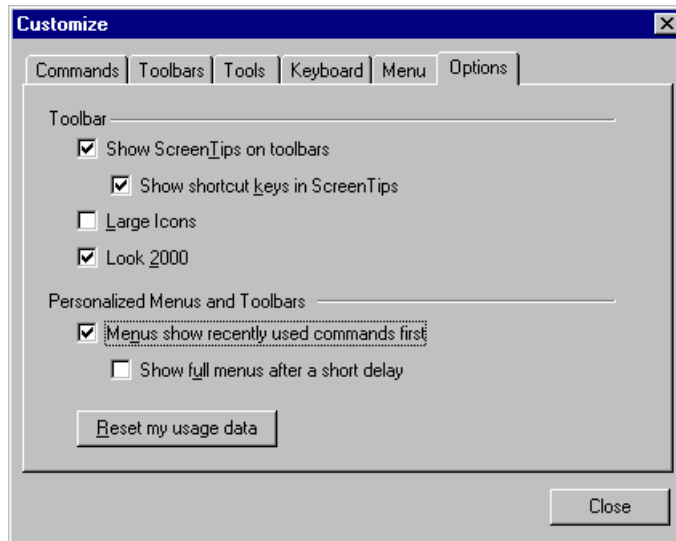
- Keyboard:** Customizes shortcut keys for menu commands. Use the *Categories* and *Commands* lists to select the menu option. The *Current Keys* field shows the shortcut assigned to the selected command. You can configure a new shortcut key for a selected command by typing the shortcut into the *Press New Shortcut Key* field and then pressing the *Assign* button. Use the *Remove* button to exclude a shortcut key from a selected command and use the *Reset All* button to restore the default settings.




- Menu:** Use *Menu animations* to set effects for the pop-up menus (*None*, *Unfold*, *Slide* or *Fade*). Use the *Menu shadows* check box to enable the shadow in the pop-up menus.




- **Options:** Customizes the general appearance of the *Toolbars* and *Menus*. Use the **Reset my usage data** button to restore the default settings for this tab.



- **Status Bar:** Displays the **Status Bar** at the bottom of the screen. When checked, the **Status Bar** opens in the interface.
- **Header:** Becomes enabled when a worksheet is active. A worksheet header displays when you check this. If you do not check this button, the header does not display.
- **Restore defaults:** Restores the default header size of the selected worksheet.
-  **Screen Attributes...:** Opens the *Screen Attributes* dialog box so you can configure general settings for a screen that is in focus within the development environment.
- **Zoom:** Provides a separate window, magnifying the image on which the pointer is pointing.

⇒ **Right-click twice to decrease the zoom scale down to 100%. Double-click to increase the zoom scale up to 3200%.**

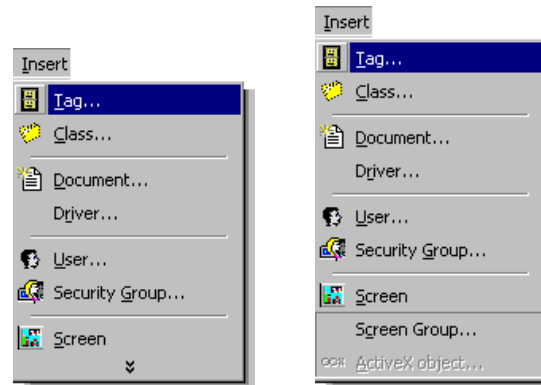
-  **Library:** Opens a library of objects previously configured. You also can open the library window using the library icon from the *Standard* toolbar.


⇒ **The objects library provides several objects with dynamics previously configured. You can use this library to add project screens and save time during application development. You also can upgrade the library with new objects by right-clicking on a screen icon (in the *Workspace*) and choosing the *Send to library* option. The application inserts the screen into the library with all its objects.**

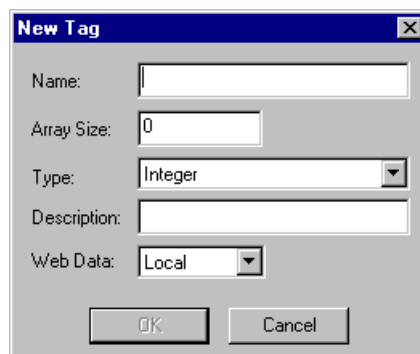
- **Properties:** Opens the *Object Properties* dialog box, which enables you to configure parameters and dynamics for an object selected from the opened screen.


3.3.4 Insert Menu

This menu contains commands that allow you to create and configure tags.




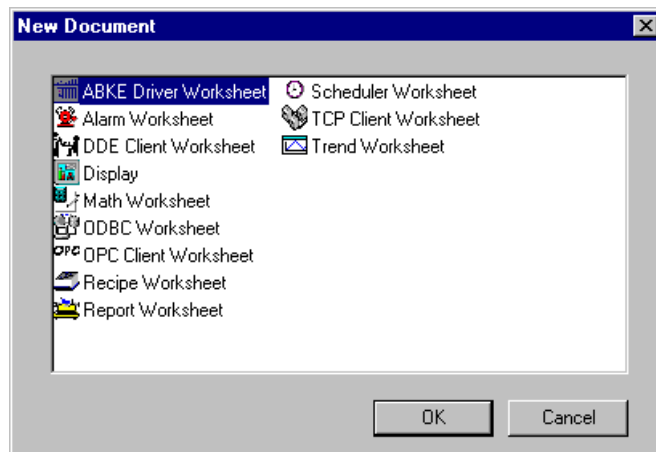
-  **Tag...**: Opens the *New Tag* dialog box where it's possible to create new tags and configure their main properties. You also can create a new tag, by right-clicking on the *Application Tags* folder located on the *Database* tab of the *Workspace* and selecting the *Insert Tag...* option from the pop-up menu.



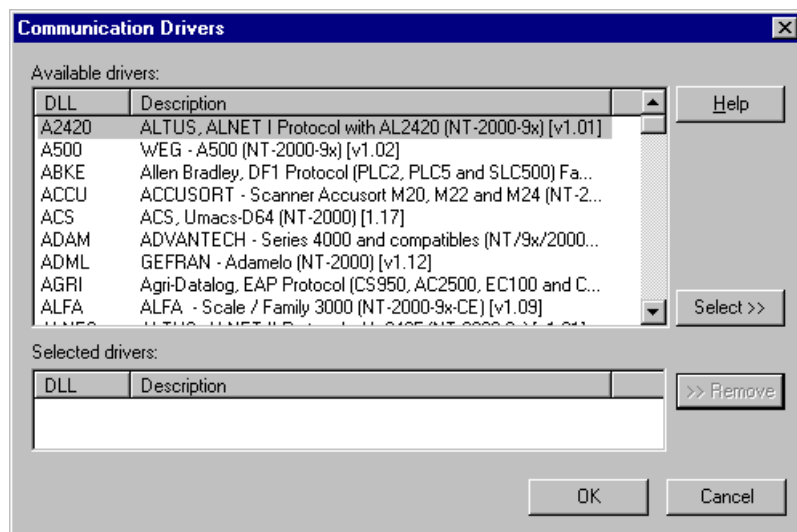
-  **Class...**: Opens the *Insert Class* dialog box where it's possible to create a new class of tags. You also can create a new tag, by right-clicking on the *Classes* folder located on the *Database* tab of the *Workspace* and selecting the *Insert Class* option from the pop-up menu.




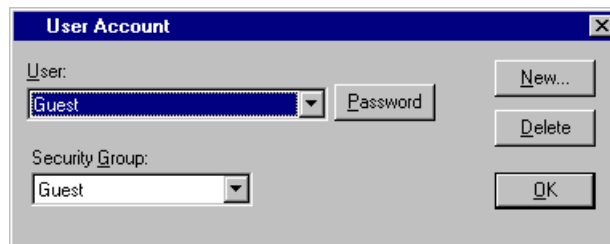
-  **Document...**: Opens the *New Document* dialog box where it's possible to create new Displays or new Worksheets. You also can create a document using the *File* tab or the **File-New** menu option.




- Driver...**: Opens the *Communication Drivers* dialog box, where it's possible to insert drivers for communicating with devices (such as PLCs) in the application. To insert a driver, select it from the *Available driver* list and press the **Select >>** button. You also can insert a driver in the application, by right-clicking on the *Drivers* folder located on the *Comm* tab of the *Workspace* and selecting the **Add/Remove drivers** option from the pop-up menu.




-  **User...**: Opens the *User Account* dialog box, where it's possible to create new users in the application security system. You also can insert a user by right-clicking on the *Users* folder located on the *Database* tab of the *Workspace* and selecting the *Insert user* option from the pop-up menu.

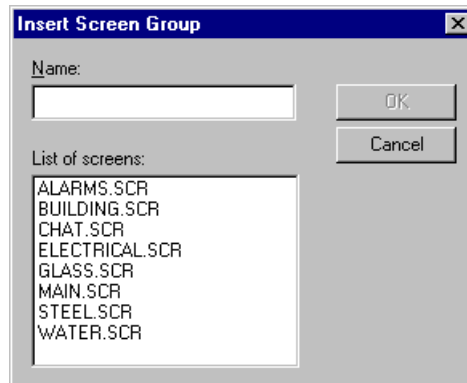


-  **Security Group...**: Opens the *Group Account* dialog box, where it's possible to create new user groups in the application security system. You also can insert a group account by right-clicking on the *Groups* folder located on the *Database* tab of the *Workspace* and selecting the *Insert group* option from the pop-up menu.

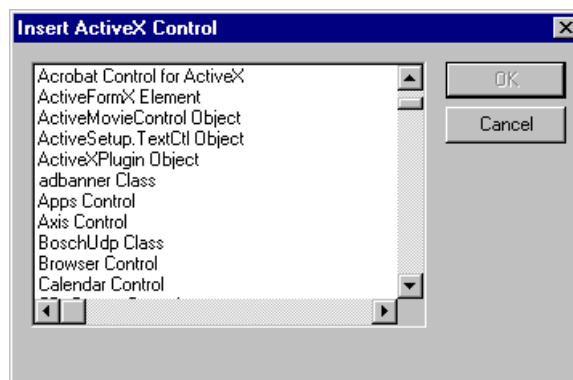


-  **Screen**: Inserts a new screen in the application. You also can insert a new screen by right-clicking on the *Screens* folder located on the *Graphics* tab of the *Workspace* and selecting the *Insert* option from the pop-up menu.

- **Screen Group...:** Opens the *Insert Screen Group* dialog box, where you can create a new group of screens in the application. You can also create a new screen group by right-clicking on the *Group Screen* folder located on the *Graphics* tab of the *Workspace* and selecting the *Insert screen group* option from the pop-up menu.



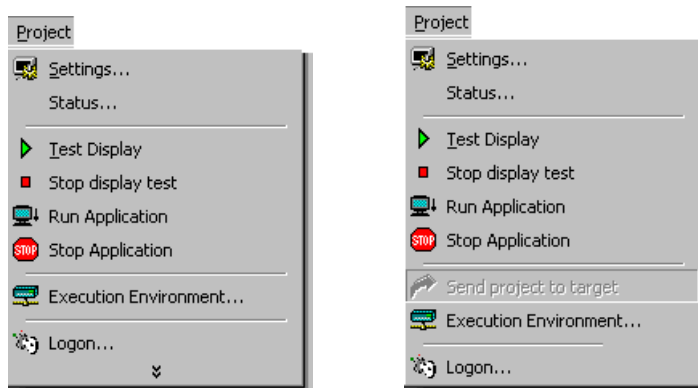
- **ActiveX object...:** Opens the *Insert ActiveX Control* dialog box, which lets you insert an ActiveX object into a screen. You also can insert an ActiveX object on an open screen by clicking on the *ActiveX Control* icon from the *Active Objects* toolbar.




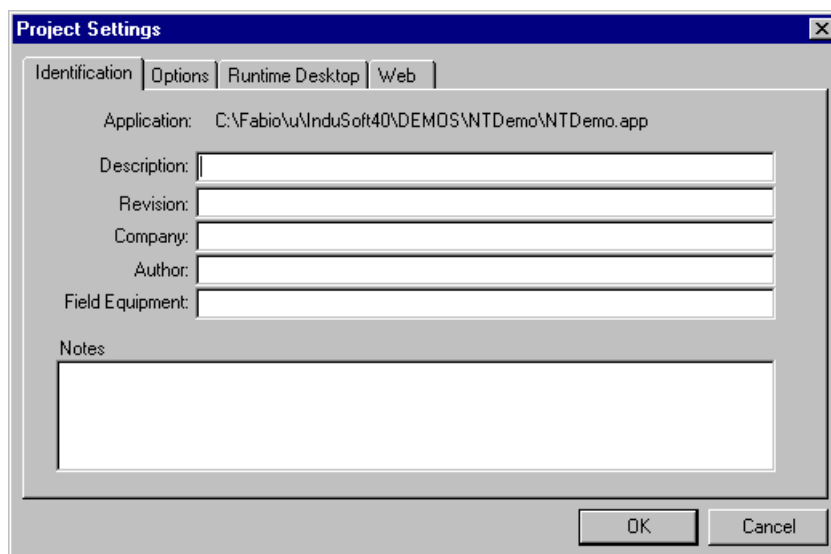
⇒ **The *Insert ActiveX Control* dialog box shows all the ActiveX controls previously registered in the computer. So, before inserting an ActiveX object in the application, you must register the object with the Windows command `regsvr32 <ControlName>`. For example: `regsvr32 e:\winnt\system32\ISSymbol.ocx`.**

3.3.5 Project Menu

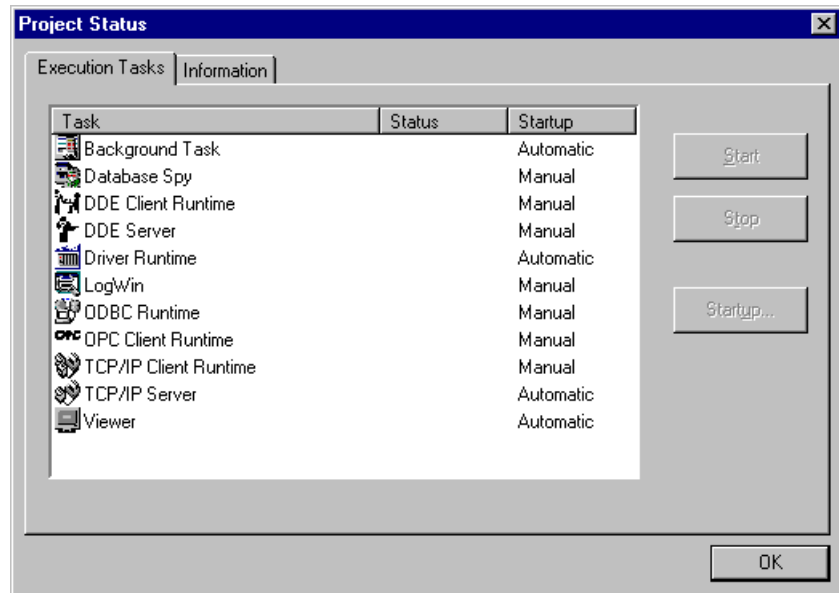
This menu contains commands and tools that you can use to manage the local/remote application execution and provide links to configure the general application settings.






-  Settings...: Opens the **Project Settings** window with four tabs: **Identification**, **Options**, **Runtime Desktop**, and **Web**. This dialog controls settings that affect the overall application.



- Status...: Opens the **Project Status** window with two tabs: **Execution Tasks** and **Information**. The **Execution Tasks** tab contains a list of **Tasks**, including their **Status** and **Startup** (automatic or manual) modes. You can use this tab to specify the tasks to be executed when the application is launched. You also can use this tab to start, or stop, any runtime task manually. The **Information** tab provides some general, view-only information about the development system and about the application. You cannot change the information on this screen.




-  **Test Display**: Activates test display mode. The Test Display button on the **Execution Control** toolbar also activates the test display mode. Test display mode allows you to configure the application while viewing graphical dynamics online in the development environment. The test display mode does not enable the **Command** or the input **Text I/O** dynamics or execute the worksheets.
-  **Stop display test**: Stops test display mode. You also can use the Stop Test Display button on the **Execution Control** toolbar to stop the test display mode.

-  Run Application: Launches the run-time modules set as *Automatic* on the *Execution Tasks* tab on the *Project Status* dialog box. You also can use the Run Application button on the **Execution Control** toolbar to launch the modules. When you start the *Viewer* module, it opens the screen(s) currently being edited. If there are no screens being edited on the development environment, it opens the screen configured in the field *Startup screen* on the *Runtime Desktop* tab located on the *Project Settings* dialog box.

⇒ If you do not set any tasks as *Automatic*, the tasks *Viewer* and *BGTask* are launched automatically when you execute the Run Application command.


**CAUTION**

This command affects the application from the *Target Station*, which is configured in the *Execution Environment* dialog box. Be sure you know which *Target Station* is configured (Local or Remote) before executing the Run Application command.

-
-  Stop Application: Stops all runtime tasks. You also can use the Stop Application button on the **Execution Control** toolbar to stop the run-time tasks.

**CAUTION**

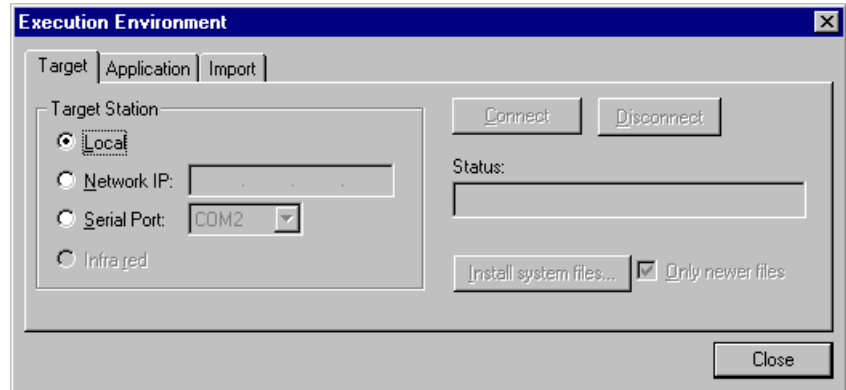
This command affects the application from the *Target Station*, which is configured in the *Execution Environment* dialog box. Be sure you know which *Target Station* is configured (Local or Remote) before executing the Stop Application command.

-
-  Send app to target: When active (and after the development computer is connected to the runtime workstation), you can use this command to send an application to the remote *Target Station*, configured in the *Execution Environment* dialog box. You also can use the Send app to target button on the **Execution Control** toolbar to send a project to a target station.

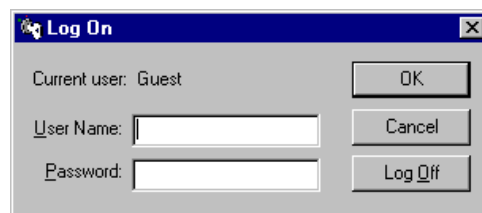
**CAUTION**

When you send an application to the remote target station, the changes will take effect online. In other words, once the application files are downloaded to the target station, they will replace the old ones automatically - even if the application was running previously. Also, if you uncheck the *Only newer files* check box on the *Application* tab located on the *Execution Environment* dialog box, all the previous files of the target application directory will be removed, before downloading the new ones.

- Execution Environment...: Opens an **Execution Environment** window with three tabs: **Target**, **Application**, and **Import**. The **Execution Environment** window also can be opened using the Execution Environment button found on the **Execution Control** toolbar. This dialog provides the interface that allows you to manage the remote stations (download/upload files and run/stop the remote application).

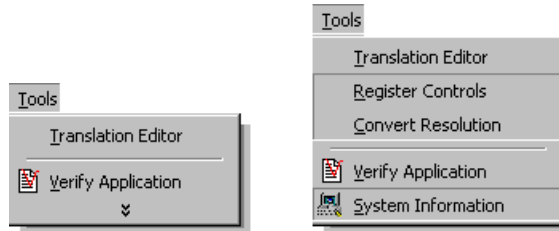


- Logon...: Opens a **Log On** window with **User Name** and **Password** text boxes. Use this dialog to log on or log off a user configured in the application **Security System**.

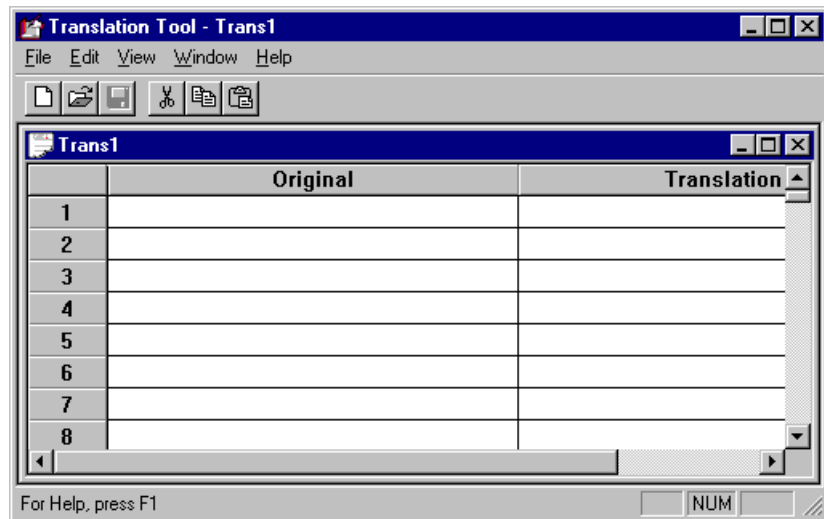


3.3.6 Tools Menu

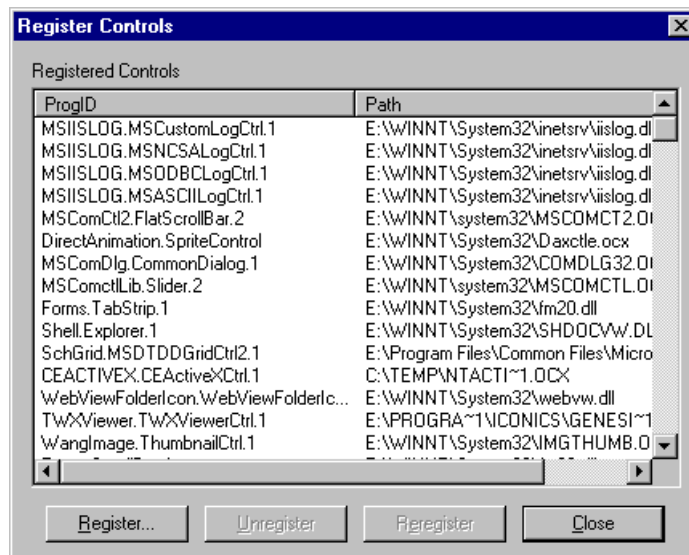
This menu provides links to auxiliary tools.



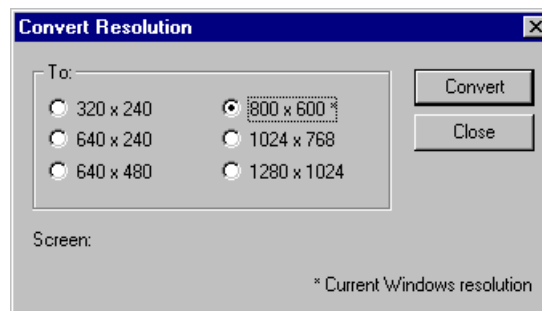
- **Translation Editor:** Opens a *Translation Tool* window that allows you to create translation worksheets.



- **Register Controls:** Opens a *Register Controls* window that allows you to register, un-register, or re-register ActiveX components.



- **Convert Resolution:** Opens a dialog that allows you to convert the screen size. It backs up the last screen size for all screens in a *Backup* folder, which is located in the *Screen* folder in your `\<application>` directory.



- **Verify Application:** Recompiles math worksheets and screen logic, and updates the HTML files with the settings that you configured using the *Web* tab on the *Project Settings* window.

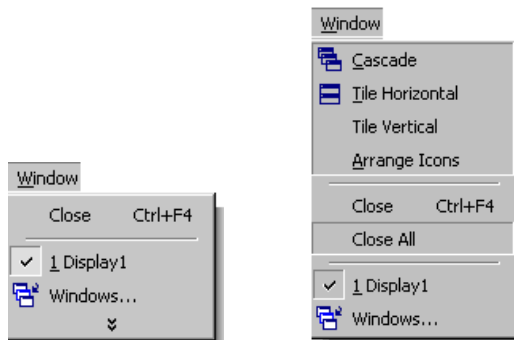
⇒ **When you save a screen or worksheet, it includes a pointer to the current database version. When you execute the application, the screen or worksheet database is compared to the current application database. If there is a mismatch, the expressions are recompiled. To avoid doing this during application runtime, you should perform the Verify Application function before downloading or finishing an application. You also should use this function when converting an application to a new version of the program.**


- **System Information:** Opens a **System Information** window, which contains information about the operating system, directories, the processor, discs, the display, and memory. **Network Resources** details the your computer's network. **Applications**, lists the applications that are currently running. **Processes**, shows all the Windows tasks that are currently running. **Services**, lists the Windows NT/2000 services used by the program (for Windows NT/2000 only).

⇒ **Although the System Information window is called from InduSoft Web Studio, it provides general information about the local station and about the network. It does not provide specific information about the application.**


3.3.7 Window Menu

This menu provides commands and tools that allow you to manage the displays and worksheets that are opened on the development environment.



-  **Cascade**: Arranges the opened worksheet and display windows in a cascade pattern.

⇒ **Screens with disabled titlebars are not affected by this function.**

-  **Tile Horizontal**: Arranges the opened worksheet windows in a tiled horizontal pattern.




⇒ **Screens with disabled titlebars are not affected by this function.**

- **Tile Vertical**: Arranges the opened worksheet and display windows in a tiled vertical pattern.

⇒ **Screens with disabled titlebars are not affected by this function.**

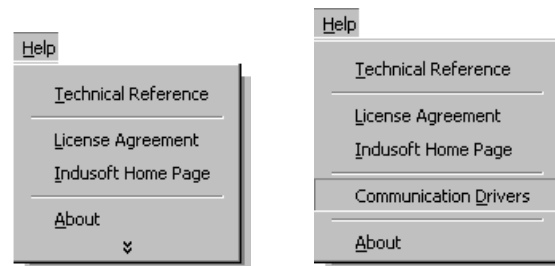
- **Arrange Icons**: Arranges minimized worksheets and display windows at the bottom of the workspace.

⇒ **Screens with disabled titlebars are not affected by this function.**

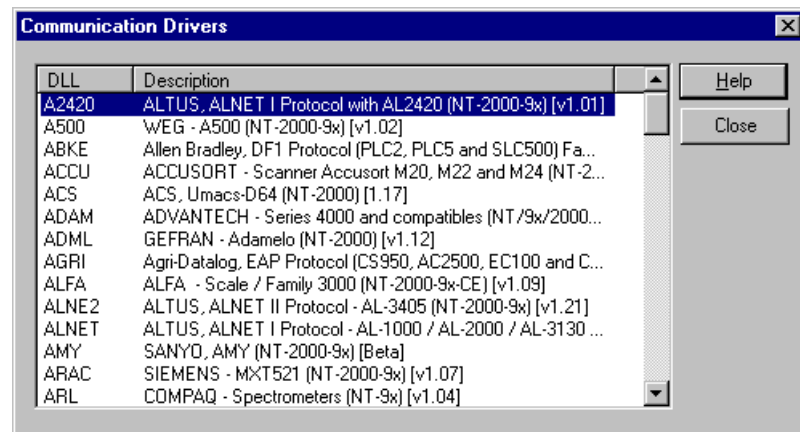
- **Close**: Closes the active screen or worksheet. You will be prompted to save changes. This command is the same as the **C**lose command in the **F**ile menu and the titlebar close button ().
- **Close All**: Closes all active screens or worksheets. You will be prompted to save changes for each file.
- **Window Listing**: Lists open files at the bottom of the **W**indow menu. The currently active file is indicated with a check (). Select a file to make it active.
-  **Windows...**: **Opens a list of all open worksheets and displays on the development environment.**

3.3.8 Help Menu

This menu provides links to information about the product and company.



- **T**echnical Reference: Opens the **main** help window.
- **L**icense Agreement: Displays the InduSoft Web Studio software license agreement, in Microsoft™ WinWord™ 97 format.
- **I**ndusoft Home Page: Opens the InduSoft web site using your web browser.
- **C**ommunication **D**rivers: Opens a Communication Drivers window from which you can select an installed driver and then open a help file for it with the Help button.



- **A**bout: Displays a window containing the copyright date, the **Version**, **Product Family**, and **Protection** type.

3.4 Toolbars

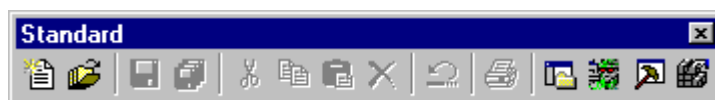
By default, the **Standard**, **Tag Properties**, **Execution Control**, **Web** and **Align and Distribute** toolbars display across the top of the workspace, just below the menu bar. By default, the **Mode**, **Static Objects**, **Active Objects** and **Dynamic Properties** toolbars, which contain screen editing tools display down to the right video side. By default, the **Bitmap** toolbar is hidden.


All toolbars are dockable screen objects. To move any toolbar to a different screen location, click on its title bar and drag it to the desired location.

⇒ To remind yourself about toolbar button functions, look at the left side of the status bar at the bottom of the InduSoft Web Studio interface. A brief description of the button currently highlighted by the mouse appears there.

3.4.1 Standard Toolbar

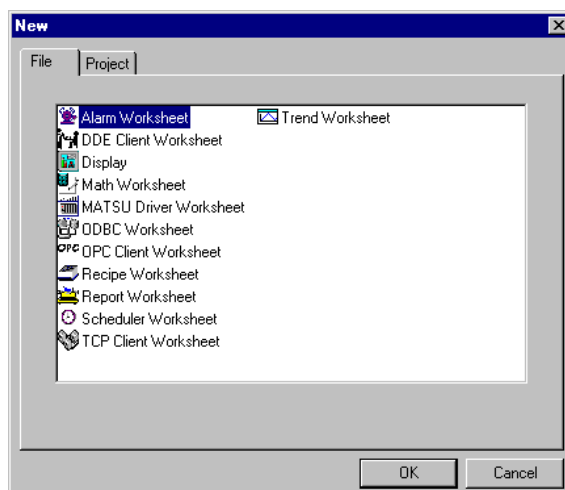
The **Standard** Toolbar provides icons, which allow you to execute general actions.
















-  **New:** Opens a New window containing File and Project tabs, which allow you to create a new application (project) or a new file that is part of your open application (Display, Math worksheet, etc.). You can also open the New window by selecting the New button from the **Standard** toolbar or using **D**ocument... in the **I**nsert menu.

The **File** tab allows you to pick new **Alarm**, **DDE Client**, **Math**, **ODBC**, **OPC Client**, **Recipe**, **Report**, **Scheduler**, **TCP Client**, and **Trend** worksheets or a new **Display** screen. When you add an I/O driver to the application, there is an option to open a new driver worksheet. The **Project** tab allows you to create a new project.

⇒ **The DDE Client and ODBC worksheets do not appear in Windows CE applications.**



-  **Open Project...:** Opens an **Open** window, which you can use to navigate to and open another InduSoft Web Studio application. You also can open a project by double-clicking on the project name in a directory in Windows Explorer or by selecting the **Open Project** in the **File** menu.
-  **Save:** Saves any active and open worksheets or screens. You also can select **Save** from the **File** menu. The **Save** function is available only when the active file has been modified.
-  **Save All:** Saves all open worksheets or screens. You also can select **Save All** from the **File** menu. **Save All** is available only when something has been modified.
-  **Cut:** Removes the selection and stores it onto the clipboard, replacing previously stored selections on the clipboard. You can use **Cut** to select an object and move it to another location on screen or move it to another screen. You also can select **Cut** from the **Edit** menu.
-  **Copy:** Copies a selection to the clipboard. **Copy** allows you to paste the selection to another location on the screen, paste it to another screen, or make multiple copies of an object. You can also select **Copy** from the **Edit** menu.
-  **Paste:** Copies the contents of the Windows clipboard to the active screen. If the clipboard contains a selection, it is copied to the upper left corner of the screen. You can also select **Paste** from the **Edit** menu.
-  **Delete:** Deletes the selection. If you accidentally delete an object, you can restore it using the **Undo** function. You also can select **Delete** from the **Edit** menu.
-  **Undo:** Cancels the last action performed while working on a screen. Cancels up to 20 actions taken prior to the current action. The actions in object properties do not increase **Undo** steps. You also can select **Undo** from the **Edit** menu.
-  **Print:** Opens a **Print** window. You can print the display or worksheet in focus. In addition, you can specify the printer name, properties, and the number of copies you would like printed. You also can print the current file by selecting **Print** from the **File** menu.
-  **Workspace:** Displays or removes the **Workspace** window. When you depress this button, the **Workspace** window opens. The **Workspace** toggle option is also available from the **View** menu.
-  **Database Spy:** Displays or removes the **Database Spy** window. When you depress this button, the **Database Spy** window opens. The **Database Spy** toggle option is also available from the **View** menu.
-  **Output:** Displays or removes the **Output** window. When you depress this button, the **Output** window opens. The **Output** toggle option is also available from the **View** menu.
-  **Library:** Opens the library of objects previously configured. The **Library** button is also available from the **View** menu and you can open the **Library** folder from the **Graphics** tab of the **Workspace** window.

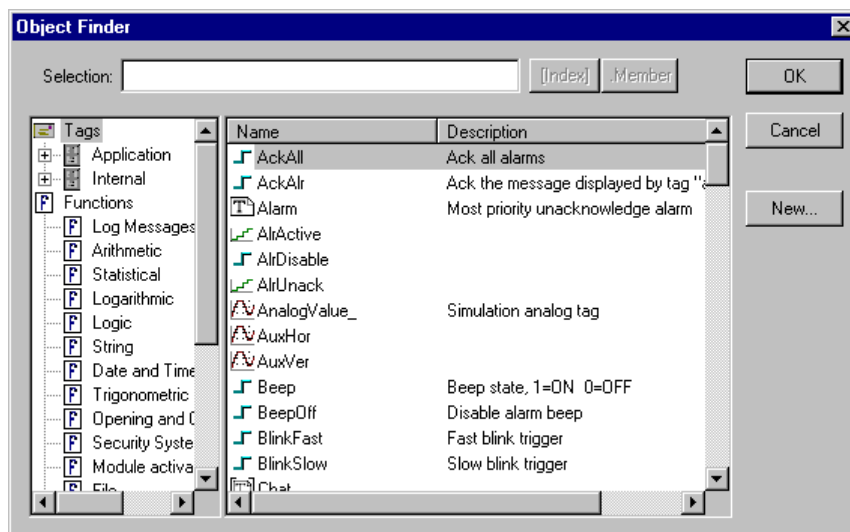
⇒ **The objects library provides several objects, with dynamics previously configured, which you can add to project screens to save application development time. You also can upgrade the library with new objects. To upgrade a library just right-click on a screen icon (in the *Workspace*) then chose the *Send to library* option. The screen will be inserted in the library with all its objects.**

3.4.2 Tag Properties Toolbar

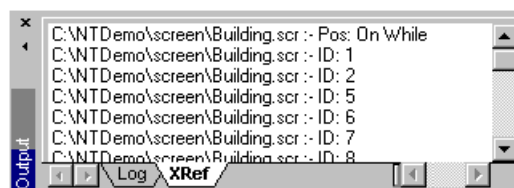
The **Tag Properties** toolbar contains special buttons for finding and accessing tags, functions, and tag properties.




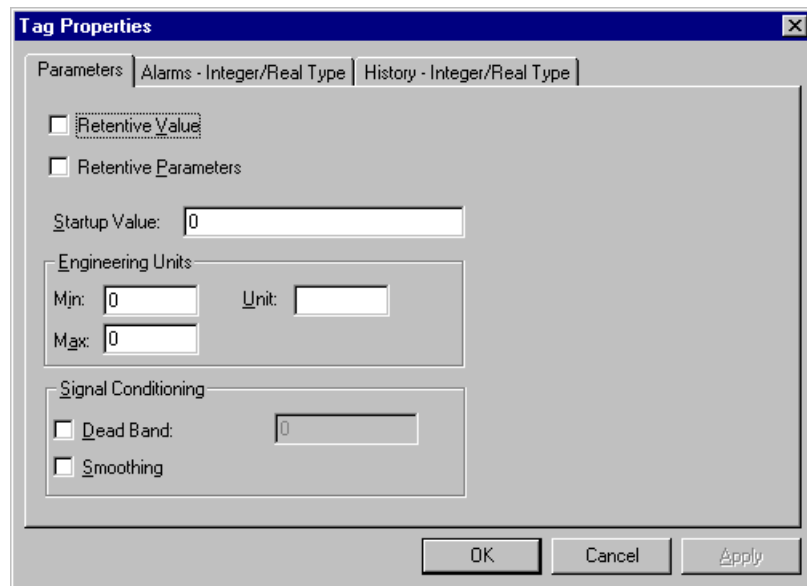
- **Tagname:** Provides a text box where you can type a tag name. The other icons (**Cross Reference** and **Tag Properties**) from the *Tag Properties* toolbar will use the tag in this field as reference for their actions.
- **Object Finder:** Opens an *Object Finder* window, which lists all functions and tags configured in the application. You can double-click on a tag to drop it into the **Tagname** text box.



- **Cross Reference:** Searches for the tag (from the **Tagname** text box) in all application screens and worksheets. Writes a log, with all the occurrences of the tag in the application to the *XRef* tab in the *Output* window.






-  **Tag Properties:** Opens a *Tag Properties* window, where you can configure parameters associated with each tag. There are different windows for Integer/Real, Boolean, and String tags.



3.4.3 Execution Control Toolbar

This toolbar provides tools, which allow you to manage the application execution (locally or remotely).




-  **Test Display:** Begins test display mode. You also can select **T**est Display in the **P**roject menu. This mode allows you to configure the application while executing the graphical dynamics online in the development environment. The test display mode does not enable the *Command* and *Text I/O* dynamic or execute the worksheets.
-  **Stop display test:** Stops test display mode. You also can select **S**top display test from the **P**roject menu.
-  **Run Application:** Launches runtime modules that were set as *Automatic* from the *Execution Tasks* tab from the *Project Status* dialog box. You also can select **R**un Application from the **P**roject menu. When you start the *Viewer* module, it opens the screen(s) to be edited. If there is no screen in the development environment, the command opens the screen configured by the field *Startup screen* on the *Runtime Desktop* tab of the *Project Settings* dialog box.

⇒ **If there are no tasks set as *Automatic*, the tasks *Viewer* and *BGTask* are launched automatically when you execute the Run Application command.**




CAUTION

This command affects the application from the *Target Station*, configured in the *Execution Environment* dialog box. Be sure you know which *Target Station* was configured (Local or Remote) before executing the Run Application command.

-  Stop Application: Stops all runtime tasks. You also can select Stop Application from the **P**roject menu.


**CAUTION**

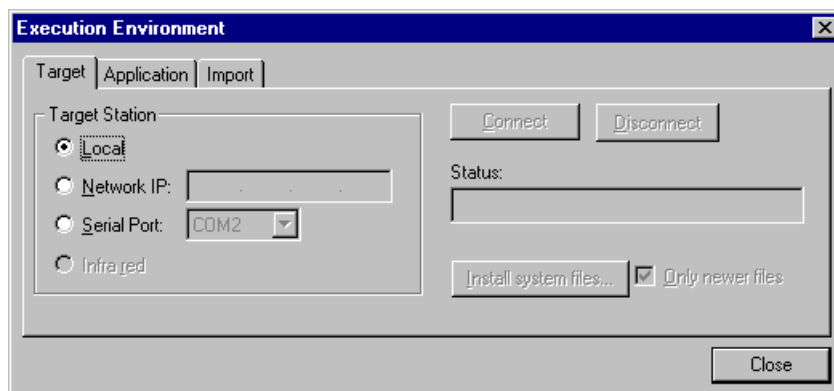
This command affects the application from the *Target Station*, configured in the *Execution Environment* dialog box. Be sure you know which *Target Station* was configured (Local or Remote) before executing the Stop Application command.

-  Send app to target: When active (and after the development computer is connected to the runtime workstation), this command can send the application to the remote *Target Station*, which is configured in the *Execution Environment* dialog box. You also can select Send project to target from the **P**roject menu.

**CAUTION**

When you send an application to the remote target station, the changes will take effect online. In other words, once the application files are downloaded to the target station, they will replace the old ones automatically - even if the application was running previously. Also, if you uncheck the *Only newer files* check box on the *Application* tab located on the *Execution Environment* dialog box, all the previous files of the target application directory will be removed, before downloading the new ones.

-  Execution Environment: Opens an *Execution Environment* window with three tabs: **T**arget, **A**pplication, and **I**mport. You also can select Execution Environment... from the **P**roject menu. This dialog provides an interface that allows you to manage the remote stations (download/upload files and run/stop the remote application).



3.4.4 Web Toolbar

This toolbar provides tools to open HTML files.

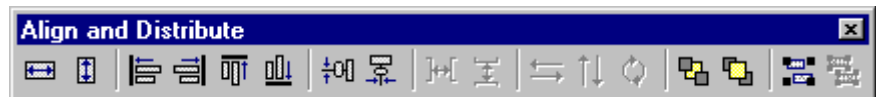


- Back: Calls the last URL address opened on the development environment.
- Forward: Calls the last URL address opened on the development environment.
- Stop: Cancels the file-downloading process from the specified URL address.
- Refresh: Reloads the URL address specified in the *Address* text box.
- Home: Calls the Home Page configured in your Internet Explorer Web browser.
- Address: Provides a text box where you can type the URL address of the page to download.
- Go: Starts downloading the pages specified in the Address text box.

⇒ **Internet Explorer v4.1 (or higher) must be installed before you use any tools from the *Web* toolbar.**

3.4.5 Align and Distribute Toolbar

This toolbar provides auxiliary drawing tools for editing the screen objects.





Resizing Objects

When you select an object or group of objects with the pointing device, eight selection handles (black squares) appear around its edge; one handle at each corner and one at the midpoint of each side. You can select a handle and drag it to elongate the object in the direction indicated by arrows that appear when you position the pointing device over the handle. For finer size control, you can select and hold a handle with the left mouse button and use the arrow keys to move the handle (and the corresponding side of the object) by one pixel at a time.

⇒ **All Group of Symbols objects – including most symbols and library objects – and all objects with dynamic properties added to them will have multiple Object Properties windows. You can access the different Object Properties windows, and the properties listed in them, from the Object Properties Selection drop-down list.**

If you resize a symbol or group of objects, all of the objects within the symbol or group are resized accordingly.


When you place the pointer cursor on a handle of an open or closed polygon, a boxed square displays at the base of the cursor. Drag this handle to move its position and change the shape of the polygon. To select and resize the whole polygon, draw a selection box around the polygon and group it.

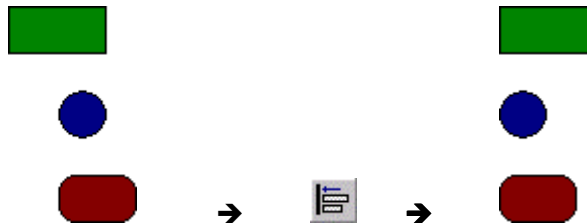
-  Resize width: Sets the width of all selected objects to the width of the last object selected (the object with the filled handles). You can use this command to resize one selected object to set its width equal to its height.
-  Resize height: Sets the height of all selected objects to the height of the last object selected (the object with the filled handles). You can use this command to resize one object to set its height equal to its width.


⇒ **You can use the Resize width and Resize height tools to create circles from an ellipse or squares from rectangles. Select just one object before using these tools.**

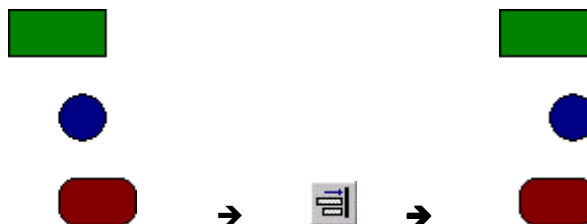
Object Alignment


The six alignment buttons are inactive until you select two or more objects. When you select two or more objects, you can use the alignment buttons to align objects according to the location of the last object selected. The last object selected has solid handles instead of empty box handles.

-  Align left: Aligns the left edges of all selected objects to the left edge of the last object selected.




-  Align right: Aligns the right edges of all selected objects to the right edge of the last object selected.




-  Align top: Aligns the top edges of all selected objects to the top edge of the last object selected.




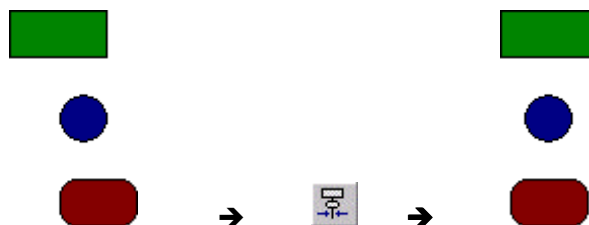
-  Align bottom: Aligns the bottom edges of all selected objects to the bottom edge of the last object selected.



-  Center Vertically: Aligns the vertical centers of all selected objects to the vertical center of the last object selected. See the figures below.




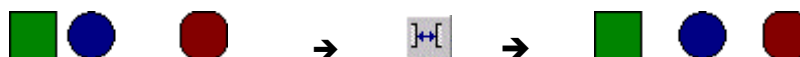
-  Center Horizontally: Aligns the horizontal centers of all selected objects to the horizontal center of the last object selected. See the figures below.




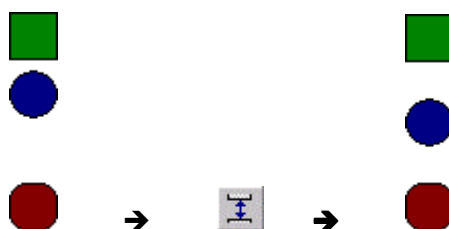
Spacing Objects Equally

The two spacing buttons are inactive until you select two or more objects. When you select two or more objects, you can use the spacing buttons to affect inter-object spacing.

-  Evenly space horizontally: Makes the horizontal space between selected objects the same.




-  Evenly space vertically: Makes the vertical space between selected objects the same.




⇒ The spacing functions may move the last object selected (the one with solid handles instead of empty box handles) by no more than a few pixels to make all of the inter-object spaces equal.

Object Orientation


The three object orientation buttons are inactive until you select a single object. The object can be a grouped object, but the buttons are not active if you select multiple objects.

-  Flip Horizontally: When you press this button, the selected object is inverted horizontally. It appears as if the object was rotated around an imaginary line through its horizontal center or as an image in a vertical mirror placed beside it and perpendicular to the screen.



-  Flip Vertically: When you press this button, the selected object is inverted vertically. It appears as if the object was rotated around an imaginary line through its vertical center or as an image in a horizontal mirror placed above or below it and perpendicular to the screen.





-  Rotate: When you press this button, the selected object rotates 90 degrees (a quarter turn) clockwise. See the figures below.





Changing Object Layers

Every object on the screen has an ID number that appears in the InduSoft Web Studio status bar when you select that object. The ID number determines whether an object appears to be behind or in front of another object on the screen. An object with a lower ID number will appear to be behind an object with a higher ID number. ID numbers always start at zero and range up to the total number of objects on the screen. No two objects have the same ID number. Even when you send a group of objects to the back or bring them to the front, the selected objects will still appear to be behind or in front of each other. The object layer buttons are available whenever you select an object or group of objects. Both the Move to back and Move to front functions also can be found on the object popup menu.

-  Move to back: When you press this button, any selected objects are given the lowest ID numbers and appear to be behind all other objects on the screen.
-  Move to front: When you press this button, any selected objects are given the highest ID numbers and appear to be in front of all other objects on the screen.

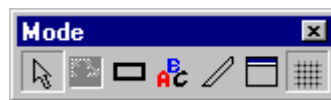
Object Grouping and Ungrouping



-  After selecting several items, you can group them using the Group button or the Group command on the object popup menu. Grouped objects are treated as a single object for the purposes of object selection and manipulation, but you can access each part of the group in the Object Properties window.
-  You can separate any grouped object back into its individual components using the Ungroup button or the Ungroup command on the object right-click menu. A grouped object can contain several individual groups of objects, so it may be necessary to select subgroups and ungroup repeatedly to completely ungroup a complex object.

⇒ **All Group of Symbols objects – including most symbols and library objects – and all objects with dynamic properties added to them will have multiple Object Properties windows. You can access these different Object Properties windows, and the properties listed in them, from the Object Properties Selection drop-down list.**


3.4.6 Mode Toolbar

This toolbar provides general tools for screen editing.




-  Selection: When you depress this button, you can use the pointer to select or move objects.
-  Bitmap Editor: Every screen has two, basic edition layers: the *Objects* layer and the *Background Picture* layer. The Bitmap Editor button allows you to switch between the two layers. Depress this button to use the *Background Picture* layer. When the *Background Picture* layer is active, the *Bitmap* toolbar displays automatically.


⇒ **The Bitmap Editor button remains grayed out (disabled) when the *Enable Background (BMP type only)* option from the *Screen Attributes* window is not checked.**

-  Fill Color: Sets the Fill Color for the selected objects. You also can use this color as the default color for newly filled objects created in the application. You can use this command on *Closed Polygon*, *Ellipse*, *Rounded Rectangle*, and *Rectangle* objects.



⇒ **You can select several objects (any type specified above) and change the Fill Color for all of them, using the Fill Color button to save development time.**

-  Fonts: Sets the font and color for selected *Text* objects. You also can use these settings as the default for new *Text* objects created in the application.

⇒ **You can select several *Text* objects and change their font and color settings, using the Fonts button to save development time. However, if you group the *Text* objects, this command does not work properly.**

-  Line Color: Sets the Line Color for selected objects. You also can use this color as the default color for new objects created in the application. Use this command for *Open Polygon*, *Closed Polygon*, *Line*, *Ellipse*, *Rounded Rectangle*, and *Rectangle* objects.

⇒ **You can select several objects (any type specified above) and change the Line Color for all them, using the Line Color button, to save development time.**




-  Background color: Sets the screen background color. This command is disabled automatically when you check the *Enable Background* option from the *Screen Attributes* window.
-  Grid: Shows/Hides the grid on the screen editor.

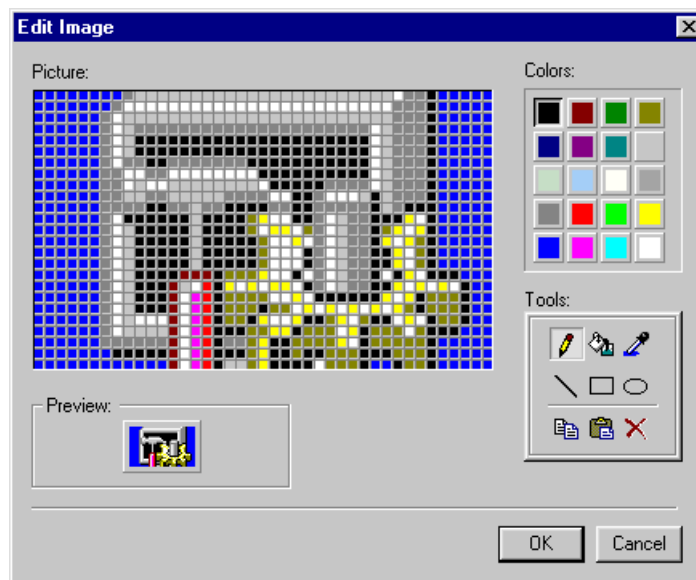
⇒ **You can configure the grid default settings from the *Grid* window. To open this window, right-click on the screen and choose the Grid Settings option from the popup menu.**





3.4.7 Bitmap Toolbar

This toolbar provides access to the main tools in the Bitmap editor. This toolbar is enabled only when the *Background Picture* layer is active.




-  **Select Area:** After clicking on this button, you can select an area from the bitmap screen editor.
-  **Flood Fill:** After selecting this button, you can click on the screen to paint the surrounding area, using the color previously selected by the Fill Color button.
-  **Pixel Editing:** Opens a zoom window, where it's possible draw detailed bitmaps, pixel by pixel.

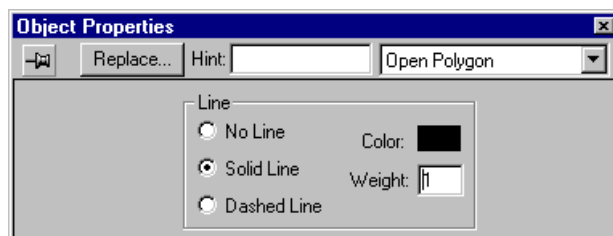


-  **Erase Area:** Fills a selected area with the color previously selected by the Fill Color button.
-  **Change colors:** Changes the Fill Color for the Transparent Color in the selected area. Before using this command, you should: Select the Fill Color using the Fill Color button; select the Transparent Color using the Select Transparent Color button; and then select the area which will be affected using the Select Area button.
-  **Select Transparent Color:** Sets the Transparent Color, used as reference for the Change Color command.
-  **Toggle Transparent Color:** When you check this button, the color selected using the Select Transparent Color button will become transparent for bitmaps selected on the bitmap editor.

⇒ You can exchange bitmap pictures between the InduSoft Web Studio bitmap editor and any other bitmap editor (*Paint Brush*, for example), using the *Copy* (Ctrl+C) and *Paste* (Ctrl+V) commands.

3.4.8 Static Objects Toolbar

-  **Open Polygon:** This draws an open polygon with the border in the foreground color. In the drawing area, click the left button to set the starting point of the polygon. Move the cursor and then lick the button again to place the second vertex. Repeat this process until you obtain the desired polygon. Double-click to stop drawing the polygon. To view the object properties, double-click on the object.




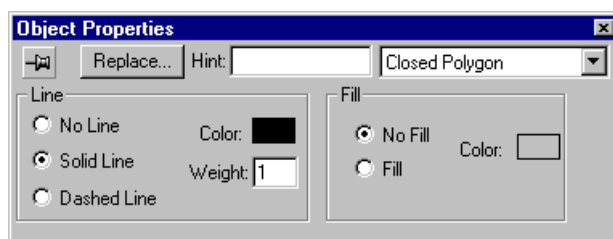
Line Group Box:

Line Radio Buttons To select a border style, click on **No Line**, **Solid Line**, or **Dashed Line**.

Color Rectangle Click the rectangle by the **Color** option to open a **Color** window. Double-click on the desired color or click the color and then the **OK** button. The object is now the desired color.

Weight Field Type a number corresponding to the desired pixel width of the line.

-  **Closed Polygon:** This draws a closed polygon with the border in the foreground color. In the drawing area, click the left mouse button to mark the first point; then continue clicking the left button until you obtain the desired polygon. To mark the last point, use a double-click or press the right mouse button. To view the object properties, double-click on the object.



Line Group Box:

Line Radio Buttons To select a border style, click on **No Line**, **Solid Line**, or **Dashed Line**.

Color Rectangle Click the rectangle by the **Color** option to open a **Color** window. Double-click on the desired color or click the color and then the **OK** button. The object is now the desired color.

Weight Field Type a number corresponding to the desired pixel width of the line.

Fill Group Box:

Fill Radio Buttons Select one of the options: **No Fill** or **Fill**.

Color Rectangle If **Fill** is selected, click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button. The object is now the desired color.

- **+ Line**: This draws an orthogonal line. Click the left mouse button on the desired point, drag it to adjust the line size, and click the button again to place the object. To view the object properties, double-click on the object.



Line Group Box:

Line Radio Buttons To select a border style, click on **No Line**, **Solid Line**, or **Dashed Line**.

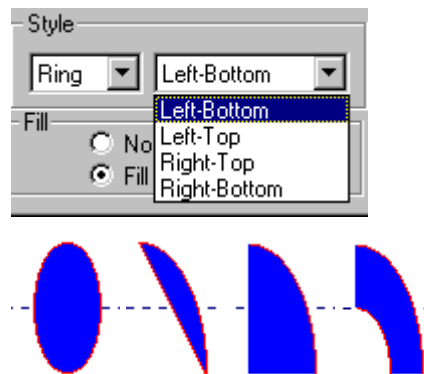
Color Rectangle Click the rectangle by the **Color** option to open a **Color** window. Double-click on the desired color or click the color and then the **OK** button. The object is now the desired color.

Weight Field Type a number corresponding to the desired pixel width of the line.

- **Ellipse**: This allows you to create ellipses, chords, arcs, and rings that are empty or filled. Click the button in the toolbar, then place the mouse in the draw area. Click and drag the mouse to create an oval shape. Change it to a chord, arc, or ring through the **Object Properties** window. To view the object properties, double-click on the object.



- **Style Group Box** - A drop-list allows you to select **Ellipse**, **Arc**, **Chord**, and **Ring** styles. When arc, chord, or ring styles are selected, a drop-list appears in the **Style** Group Box. Style selections include: Left-Bottom, Left-Top, Right-Bottom, Right-Top.



Ellipse, Chord, Arc, and Ring

⇒ **The Ring style is particularly useful in creating plumbing drawings.**

Line Group Box:

Line Radio Buttons

To select a border style, click on **No Line**, **Solid Line**, or **Dashed Line**.

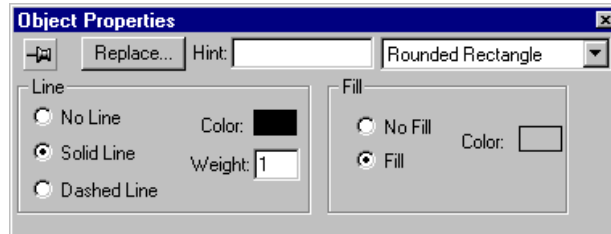
Color Rectangle

Click the rectangle by the **Color** option to open a **Color** window. Double-click on the desired color or click the color and then the **OK** button. The object is now the desired color.

Weight Field

Type a number corresponding to the desired pixel width of the line.

- Rounded Rectangle:** This creates rounded rectangles that are empty or filled. Click the button in the toolbar, then place the mouse in the draw area. Click and drag the mouse to adjust the shape. You cannot use a rounded rectangle shape to create a bar graph for WinCE applications. The Rounded Rectangle has one extra tracker in the lower left corner that allows you to modify the arc angle.



Line Group Box:

Line Radio Buttons

To select a border style, click on **No Line**, **Solid Line**, or **Dashed Line**.

Color Rectangle

Click the rectangle by the **Color** option to open a **Color** window. Double-click on the desired color or click the color and then the **OK** button. The object is now the desired color.

Weight Field

Type a number corresponding to the desired pixel width of the line.

Fill Group Box:

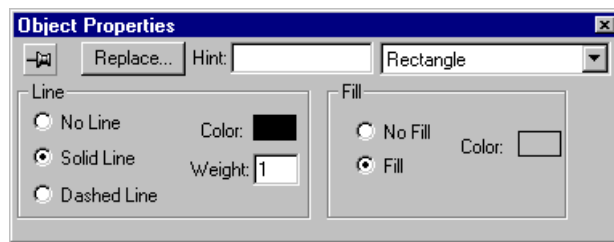
Fill Radio Buttons

Select one of the options: **No Fill** or **Fill**.

Color Rectangle

If **Fill** is selected, click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button. The object is now the desired color.

- Rectangle:** This creates rectangles that are empty or filled. Click the button in the toolbar, then place the mouse in the draw area. Click and drag the mouse to adjust the shape. To view the object properties, double-click on the object.



Line Group Box:

Line Radio Buttons

To select a border style, click on **No Line**, **Solid Line**, or **Dashed Line**.

Color Rectangle

Click the rectangle by the **Color** option to open a **Color** window. Double-click on the desired color or click the color and then the **OK** button. The object is now the desired color.

Weight Field

Type a number corresponding to the desired pixel width of the line.

Fill Group Box:

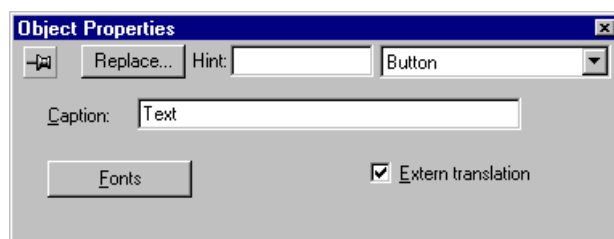
Fill Radio Buttons

Select one of the options: **No Fill** or **Fill**.

Color Rectangle

If **Fill** is selected, click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button. The object is now the desired color.

- Button:** Use this option to create custom size buttons. Click the button in the toolbar, then place the mouse in the draw area. Click and drag the mouse to adjust the shape. To view the object properties, double-click on the object.



Caption Field

Type in the text you want to appear on the button.

Fonts Button


Accesses the **Font** window, which allows you to define styles, sizes, colors, and font types.

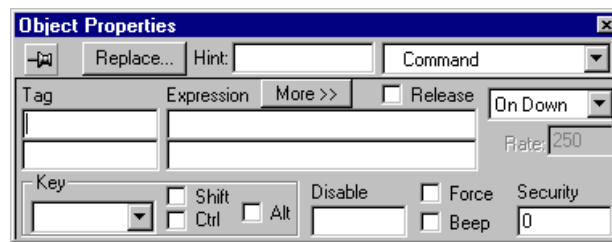
Extern Translation Check-box

Enables the use of an external translation file for the text on the button.

3.4.9 Dynamic Properties Toolbar


Dynamic properties must be applied to objects previously drawn and selected. You can apply several dynamics to each object or to each group of objects. Some dynamics can be applied to some objects types. The dynamics allows modify the objects properties on the fly (during the runtime) according to tags values. Some dynamics allow also that the user execute commands or insert values (set-points) to the tags.

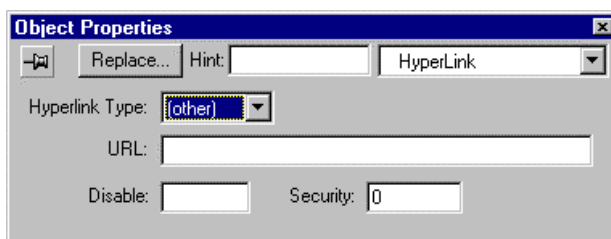
-  **Command:** This tool adds the command property to the object currently selected. During runtime execution, if the mouse is clicked on the object or the defined key is pressed, the command is executed. To view the object properties, double-click on the object.




| | |
|----------------------------|--|
| Tag Field | Tag that receives the result of the math expression. |
| Expression Field | Mathematical expression (command) that is executed when the key is pressed or the mouse command is triggered. |
| More>> Button | This opens more expression lines. |
| Release Check-box | When it's checked, the On Up event is executed when the pointer (mouse or finger) is dragged out the object area, does not matter if it was released or not. |
| Events Drop-List | You can create commands on the following events: <ul style="list-style-type: none"> On Down Expressions are executed when the mouse button (or key) moves down. On Up Expressions are executed when the mouse button (or key) moves up. On While Expressions are executed when the mouse button (or key) is pressed. |
| Rate Field | Defines the specified rate in milliseconds, for the On While event. |

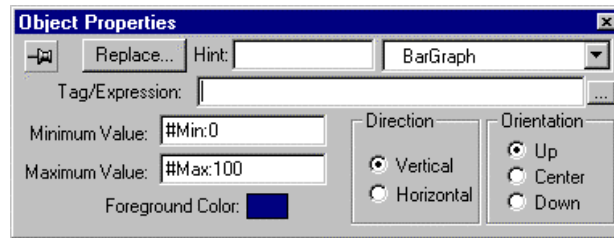
| | |
|------------------------|---|
| Key Drop-List | A key is associated with the object that triggers the execution of the command. |
| Key Drop-List | Selects from numerous keyboard keys listed. |
| Shift Check-box | This makes it necessary to press the SHIFT key with the selected trigger key. |
| Ctrl Check-box | This makes it necessary to press the CTRL key with the selected trigger key. |
| Alt Check-box | This makes it necessary to press the ALT key with the selected trigger key. |
| Disable Field | Disables the command property when the value from the tag typed in this field is greater than 0. |
| Beep Check-box | When selected, beeps when executing the command. |
| Security Field | Indicates the security level of the object, defined under Security . When the logged on user does not have this security level, the command is disabled. |

-  **Hyperlink:** This tool adds the hyperlink property to the currently selected object. During execution, if the mouse is clicked on the object or the defined key is pressed, the default browser is executed and link configured is opened. To view the object properties, double-click on the object.



| | |
|---------------------------------|---|
| Hyperlink Type Combo-box | Sets the hyperlink protocol type. When selected the protocol type, it's automatically inserted in the URL prefix. |
| URL Field | Link address (e.g. http://indusoft.com.br). |
| Disable Field | Disables the hyperlink command property when the value from the tag typed in this field is greater than 0. |
| Security Field | Indicates the security level of the object, defined under Security . When the logged on user does not have this security level, the command is disabled. |


-  **BarGraph:** This command adds bar graph properties to the currently selected object. To view the object properties, double-click on the object.

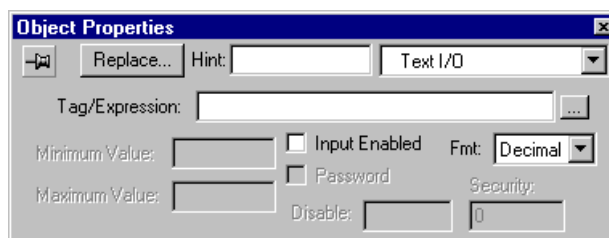


| | |
|--------------------------------|--|
| Tag/Expression Field | Tag or expression that evaluates the bar graph level. |
| Minimum Value Field | Defines the minimum value to calculate the height (if vertical) or width (if horizontal) of the bar. To define it, type a numeric constant or a tag in this field. |
| Maximum Value Field | Defines the maximum value to calculate the height (if vertical) or width (if horizontal) of the bar. To define it, type a numeric constant or a tag in this field. If the typed tag has not been created, a window displays and asks for the confirmation of the tag creation. |

⇒ In the fields where you enter a tag or a numeric value, you may also enter a constant. A constant (defined by the # character) is equivalent to a numeric value, except it appears in the Tag Replace window. It is useful for documentation and to create generic objects. *Example: #Name:100.* The number after the “:” is the constant value; the name is only a constant mnemonic that is not added to database.


| | |
|------------------------------|---|
| Foreground Color | Selects the color that will be used to fill the object. Click Foreground Color rectangle to display a Color window. Double-click on the desired color or click the color and then the OK button. |
| Direction Group Box | Determines if the bar graph will be Vertical or Horizontal . To select the direction, click the desired option. |
| Orientation Group Box | Determines the orientation used for the max and min values to draw the bar. Selections are Up , Center , and Down . To select one, click the desired option. |

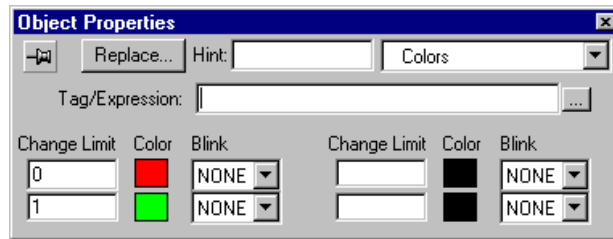
-  **Text I/O:** This option adds the dynamic input or output property of text to the currently selected text object. When running the application, using the keyboard, or on screen keypad if enabled, allows the user to insert the value of a tag and displays the value of a tag in real time. To view the object properties, double-click on the object.



⇒ **This dynamic can only be applied to text objects that contain the characters "#", each one representing one character.**

| | |
|----------------------------------|---|
| Tag/Expression Field | Holds a tag to the input or output operation or an expression only to the output operation. |
| Input Enable Check-box | Allows data entries; otherwise, this dynamic only executes the data outputs. |
| Minimum Value Field | Defines the minimum value of the tag associated with the object. The user is not allowed to input a number lower than this value. |
| Maximum Value Field | Defines the maximum value of the tag associated with the object. The user is not allowed to input a number greater than this value. |
| Password Check-box | Makes all text invisible to the user; text is replaced by asterisks (*). |
| Fmt Drop-list | From this list you can select the format for the I/O field. |
| Disable Field | Disables the data input property when the value from the tag typed in this field is greater than 0. |
| Security Field | Indicates the security level of the specific object for data input, defined under Security section. |

-  **Colors:** This tool adds the color change property to the selected object. The field should be filled with the tag that you want to monitor. This dynamic accepts up to four limits for the changing of colors. To view the object properties, double-click on the object.




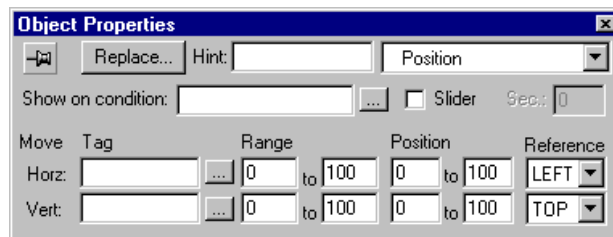
Tag/Expression Field Tag or expression associated with the object. The result of the expression will be compared with the change limits to determine the proper color for the object.

Change Limit Field Limit value for the color change. It must have a numeric constant or a tag.

Color Rectangle Defines the color associated with each limit for the color change. Click the rectangle and the **Color** window will appear. Double-click on the desired color or click the color and then the **OK** button.

Blink Drop-list Defines if the color change will blink or not, and if so how fast.

-  **Position:** This dynamic property lets you place the objects (or not) anywhere in the screen, according to values of the tags in database. It can be applied to any selected object. To view the object properties, double-click on the object.



Show on Condition Field May contain a math expression. When the expression is greater than zero, the object is visible; otherwise it is hidden. Leaving this field blank makes the object always visible.

Slider Check-box When checked, this object acts as a slider. It accepts mouse dragging, applying the corresponding values to tags.

Sec.: Field Security level of the object. When using the slider option, this defines the security level required to enable operator input by the slider option.


Tag Field Tag associated with the object, allowing it to move horizontally and vertically throughout the screen.

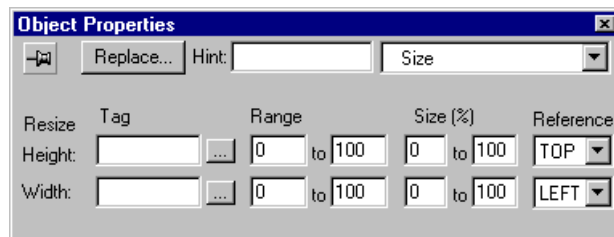
Range Field Defines the upper and lower limit for the tag values in order to make the object move throughout the screen according to the established condition.

Position Field Defines the change in position, in pixels, that the object moves through on the screen according to the established condition. The value in the second box (destination position) can be negative.

Reference Drop-list Defines the reference point in the object to move it throughout the screen. This option is only necessary if the object is being resized while it is moving.

- Left** Left corner of the object
- Right** Right corner of the object
- Center** Center of the object
- Top** Upper corner of the object
- Bottom** Lower corner of the object

-  **Resize:** This allows you to increase or decrease the size of an object or symbol according to application tags. Once clicked, the size property is added to the selected object. To view the object properties, double-click on the object.




Tag Field Tags associated with the increase or decrease the object's horizontal and vertical size.

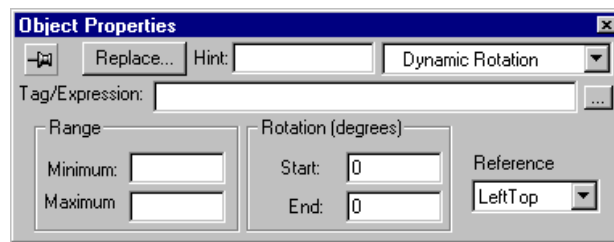
Range Field Defines the upper and lower limit of the tag values for increasing and decreasing the object size.

Size (%) Field Defines the percentage on which the system will be based to change the specified object size.

Reference Drop-list Tells how the object will increase its size horizontally and vertically.

- Left** From the left side
- Right** From the right side
- Center** Horizontally and vertically from the center of the object
- Top** From the upper side
- Bottom** From the lower side

-  **Rotation Property:** This option allows the movement of a line according to a specific pointer in the application. This property can be applied to the selected line objects. To view the object properties of a line, double-click on the object.




⇒ This dynamic can only be applied to line objects.

| | | | | | | | |
|-------------------------------------|---|-----------------|---------------------------------|---------------------|----------------------------------|---------------|----------------------|
| Tag/Expression Field | Tag or expression associated with the object. The value represented on the screen will be read from this variable or expression. | | | | | | |
| Range Group Box | Defines the upper and lower limit of the tag values in order to make the object move throughout the screen according to the established condition. | | | | | | |
| Rotation (degrees) Group Box | Defines the starting and ending degrees for the dynamic of the object rotation. On NT a line can rotate up to 360 degrees, but on CE it does not go past 90 degrees. | | | | | | |
| Reference Drop-list | Defines the reference point in the object to rotate it throughout the screen: <table border="0" style="margin-left: 20px;"> <tr> <td>Left Top</td> <td>Upper-left corner of the object</td> </tr> <tr> <td>Right Bottom</td> <td>Lower-right corner of the object</td> </tr> <tr> <td>Center</td> <td>Center of the object</td> </tr> </table> | Left Top | Upper-left corner of the object | Right Bottom | Lower-right corner of the object | Center | Center of the object |
| Left Top | Upper-left corner of the object | | | | | | |
| Right Bottom | Lower-right corner of the object | | | | | | |
| Center | Center of the object | | | | | | |

3.4.10 Active Objects Toolbar

These objects already cover some specific dynamics and require more parameters than a Static Object.

-  **Alarm:** This option selects an area in the screen to show the list of alarm messages. Click the icon in the Object Editing Toolbar; then place the mouse in the draw area. Click and drag the mouse to adjust the shape. To view the object properties, double-click on the object.



History Radio Button Sets the object to show alarm messages from the history files.

On Line Radio Button Sets the object to show on-line alarm messages.



CAUTION

It's necessary to set the option **Save to Disk** in the **Alarm worksheet** from the **Tasks** tab to save the alarm messages configured in that worksheet to history files.

Border Rectangle Defines the color of the alarm message border. Click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button.

Win Rectangle Defines the background color of the alarm message window. Click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button.

PgUp Field Scrolls up the alarm list. This should be associated with a tag name. A toggle to the tag commands a page up on the list.

PgDown Field Scrolls down the alarm list. This should be associated with a tag name. A toggle to the tag commands a page down on the list.

| | |
|----------------------------------|---|
| Message Format Group Box | Defines the alarm message format. The alarm can include the date, hour, name, tag, and message. |
| Font Button | Accesses the Fonts window, which defines styles, sizes, colors, and font types. |
| DD,MM,YY Check-boxes | Click the desired options to activate or deactivate the dates. |
| HH,MM,SS, MSS Check-boxes | Click the desired options to activate or deactivate the time. |
| * | If checked, displays an asterisks character between the alarm date/time and the alarm message. |
| Tag Field | Type a number for the tag name characters. |
| Message Field | Type a number for the messages characters. |
| Ack Check-box | Adds the acknowledged time. |
| End Check-box | Adds the normalization time. |
| Selection... Button | Opens an Alarm Filters window where you can specify filters for the listed alarm messages: |

| | |
|------------------------|--|
| Group Field | Allows you to select groups of alarms to be displayed in the alarm summary object. If the value in the field group is 0 (zero) all alarms are selected. If it is other than zero, a specific alarm group is selected. |
| Selection Field | Filters display alarms by matching the string that you declare in this field with the string declared in the selection column on the Alarm worksheet. |

⇒ **The string must be the exact string from the Selection column in the Alarm worksheet.**

⇒ **You can type a string tag name between curly brackets {} in this field and modify the tag value to modify the selection during the runtime.**


| | |
|------------------------------------|--|
| Priority Range Group Box | Filters display alarms by the priority assigned to the alarm in the priority column on the Alarm worksheet. It will group according to the priority assigned. For example, if an alarm is assigned 1 to 5 and you put a priority range from 0 to 4, then only alarm priorities 1 to 4 will be displayed and not alarm priority 5. |
| Sort Group Box | Contains Radio buttons to indicate sort by Time or Priority. Alarms to be displayed by either time that the alarm was received or by priority assigned to the alarm. |
| Print Tag Field | When a tag is inserted into this field, all alarms selected by the filters will be printed when this tag is changed. |
| Ack Tag Field | When the tag inserted into this field tag is changed, the current filtered active alarm (in the top of the alarm object list) will be acknowledged. |

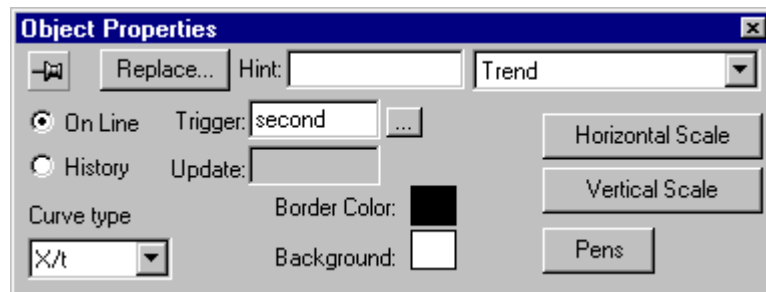
⇒ **You can use the internal tag AckAlr to acknowledge the last alarm from the application.**

| | |
|--------------------------|--|
| Ack All Tag Field | When the tag inserted into this field is changed, all the filtered active alarms will be acknowledged. |
|--------------------------|--|

⇒ **You can use the internal tag AckAll to acknowledge all alarms from the application.**

| | |
|---------------------|--|
| View % Field | The tag inserted in this field (string or integer) is given a value between 0 and 100 corresponding to the current alarm message location (in percentage) in the alarm list. |
|---------------------|--|

-  **Trend:** Selects an area on the screen for the exhibition of the Trend curves. The configuration fields specify the exhibition period, the values that will be exhibited, and the format of the graphic. You can expose up to eight curves simultaneously in the trend object. Click the button in the toolbar, then place the mouse in the draw area. Click and drag the mouse to adjust the shape. To view the object properties, double-click on the object.



On Line Radio Button Displays the online trend curves of the application.

History Radio Button Displays the historical trend curves of the application.

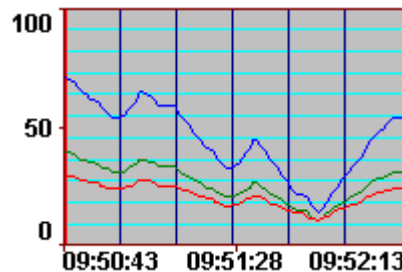


CAUTION

To use the history type graphics, you have to configure a Trend group through the Trend folder from the Tasks tab, create tags for this group and set the options **Save on Tag Change** or **Save on Trigger** from the trend group worksheet. These tags will have their samples stored on the hard disk.

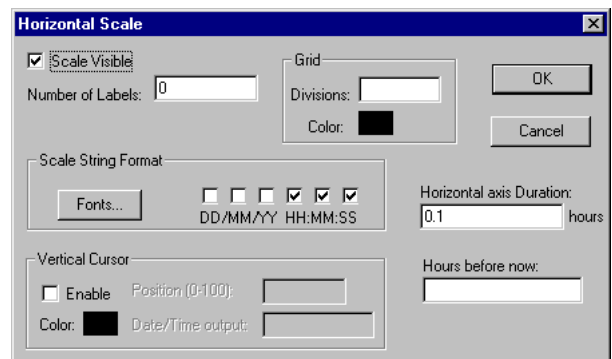
| | | | | | | | |
|-----------------------------|---|------------|---|------------|--|--------------|--|
| Trigger Field | A variable that defines the redraw of trend curves. Whenever there is a transition or the define tag, the curves are drawn. This field is obligatory in On Line trend; it is not used for History trends. | | | | | | |
| Update Field | When you enter a tag variable in this field, Studio refers to the tag to create a trend. It is used <u>only</u> in conjunction with Crisp trends. | | | | | | |
| Curve Type Drop-list | Defines the type of the curve used in the trend graphic. | | | | | | |
| | <table border="0"> <tr> <td style="padding-left: 20px;">X/t</td> <td>Plots the tag values according to time.</td> </tr> <tr> <td style="padding-left: 20px;">X-Y</td> <td>Plots curves from tag values according to the Tag X, another tag in the application.</td> </tr> <tr> <td style="padding-left: 20px;">Crisp</td> <td>Specific format for interface with VAX stations.</td> </tr> </table> | X/t | Plots the tag values according to time. | X-Y | Plots curves from tag values according to the Tag X, another tag in the application. | Crisp | Specific format for interface with VAX stations. |
| X/t | Plots the tag values according to time. | | | | | | |
| X-Y | Plots curves from tag values according to the Tag X, another tag in the application. | | | | | | |
| Crisp | Specific format for interface with VAX stations. | | | | | | |

- Border Color Rectangle** Defines the background color of the selected area for the trend graphic. Click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button.
- Background Color Rectangle** Defines the background color of the selected area for the trend graphic. Click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button.
- Horizontal Scale Button** Opens a window where you define the horizontal scale properties of the trend window.
- Vertical Scale Button** Opens a window where you define the vertical scale properties of the trend window.
- Pens Button** Opens a window that allows the configuration of the pen to draw each tag curve.



HORIZONTAL SCALE WINDOW- Curve Type X/t (ON-LINE)

This is the window when the On-line and Graph X/t options have been selected on the Object Properties Trend window.



- Scale Visible Check-box** Makes the trend graphic scale visible.
- Number of Labels Field** Specifies the quantity of labels to be used in the trend graphic scale.

Grid Group Box

Divisions Field Defines the number of division lines for the trend graphic grid.

Color Rectangle Defines the color of the trend graphic grid. Click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button. If this field is not filled, there will not be any grids.

Scale String Format Group Box

Specifies the format of the string to be used in the horizontal scale of the trend graphic.

Fonts Button Accesses the **Fonts** window, which defines styles, sizes, colors, and font types for the horizontal axis labels.

DD/MM/YY-HH:MM:SS Check-boxes You can configure the string with hour, minute, and second.

Vertical Cursor Group Box

Enable Check-box Enables the vertical cursor on the trend window.

Color Rectangle Sets the color of the vertical cursor.

Position (0-100) Field When using the vertical cursor, must be filled with a Real tag that will be updated with the cursor position.

Date/Time Output Field Selects a tag name to receive the string with the current time of the vertical cursor.

Horizontal Axis Duration Field

Specifies the value of the scale break of the trend graphic. This field can be a tag or a numeric value. *Example:* If the value of the Horizontal Axis Duration = 0.03333 (2 minutes).

Hours Before Now Field

Performs scrolling in the trend graphic by the use of a tag. *Example:* If Hours Before Now = HOUR, tag HOUR = 5, and current hour = 5 p.m., the system allows the display of the trend graphic in five hours before.

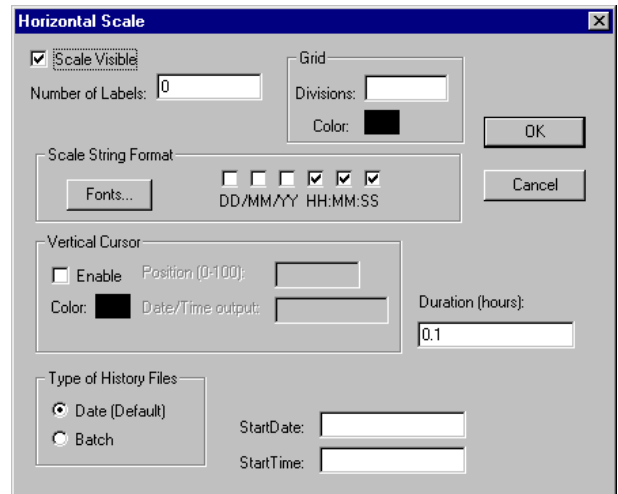
**CAUTION**

The maximum number of samples the trend can hold in the window is 16,000.

⇒ When you use Hours Before Now, you are handling historical data, so you must configure the pens in one trend group.

HORIZONTAL SCALE WINDOW- Curve Type X/t (HISTORY)

This is the window when the **History** and **Graph X/t** options have been selected on the **Object Properties Trend** window. The history graph should be used to handle past data that is more than a few hours old, as well as batch files.



Scale Visible
Check-box

Makes the trend graphic scale visible.

Number of Labels
Field

Specifies the quantity of labels to be used in the trend graphic scale.

Grid Group Box

Divisions
Field

Defines the number of division lines for the trend graphic grid.

Color Rectangle

Defines the color of the trend graphic grid. Click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button. If this field is not filled, there will not be any grids.

Scale String Format Group Box

Specifies the format of the string to be used in the horizontal scale of the trend graphic.

Fonts Button

Accesses the **Fonts** window, which defines styles, sizes, colors, and font types for the horizontal axis labels.

**DD/MM/YY-
HH:MM:SS**
Check-boxes

You can configure the string with hour, minute, and second.

Vertical Cursor Group Box

| | |
|---|---|
| Enable Check-box | Enables the vertical cursor on the trend window. |
| Color Rec- tangle | Sets the color of the vertical cursor. |
| Position (0-100) Field | When using the vertical cursor, must be filled with a Real tag that will be updated with the cursor position. |
| Date/Time Output Field | Selects a tag name to receive the string with the current time of the vertical cursor. |

Duration (hours) Field Specifies the value of the scale break of the trend graphic. This field can be a tag or a numeric value. *Example:* If the value of the **Horizontal Axis** Duration = 0.03333 (2 minutes).

Types of History Files Group Box

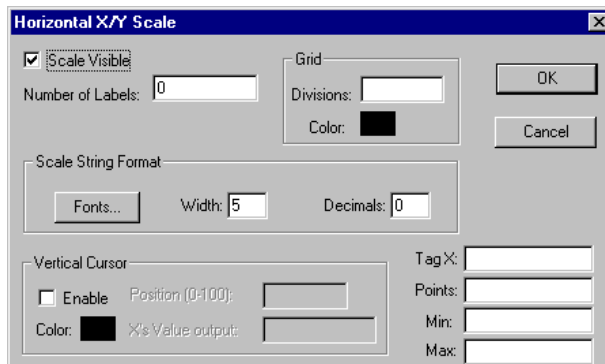
| | |
|--------------|---|
| Date | Handles files created in date format by trend group. |
| Batch | Handles files created in batch format by trend group. |

Start Date Field This sets the date to start the history curves. Usually filled with a string type tag. Its value should be in the date format DD/MM/YYYY.

Start Time Field Sets the time to start the history curves.

HORIZONTAL SCALE WINDOW- Curve Type X/Y

This is the window when the **X-Y** option has been selected on the **Object Properties** Trend window. This graph type plots curves from a set of tags according to a Tag X and also from the Application Database. On the configuration, you define the variable in the **Horizontal** window and the Y variables in the **Pens** window.



Scale Visible
Check-box

Makes the trend graphic scale visible.

Number of Labels
Field

Specifies the quantity of labels to be used in the trend graphic scale.

Grid Group Box

Divisions Field Defines the number of division lines for the trend graphic grid.

Color Rectangle Defines the color of the trend graphic grid. Click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button. If this field is not filled, there will not be any grids.

Scale String Format Group Box

Specifies the format of the string to be used in the horizontal scale of the trend graphic.

Fonts Button Accesses the **Fonts** window, which defines styles, sizes, colors, and font types for the horizontal axis labels.

DD/MM/YY-HH:MM:SS Check-boxes You can configure the string with hour, minute, and second.

Vertical Cursor Group Box

| | |
|-------------------------------|---|
| Enable Check-box | Enables the vertical cursor on the trend window. |
| Color Rec- tangle | Sets the color of the vertical cursor. |
| Position (0-100) Field | When using the vertical cursor, must be filled with a Real tag that will be updated with the cursor position. |
| Date/Time Output Field | Selects a tag name to receive the string with the current time of the vertical cursor. |

Tag X Field

Tag to be the X axis. This tag must be an array; when filling this field you must declare the position in which it is to start. (e.g. MyTagX[1]).

Points Field

Number of points (samples) in the graph window.

Min Field

Minimum value for the X variable.

Max Field

Maximum value for the X variable.

⇒ You can use Recipe feature from the Tasks tab to save and load history information for the X-Y trend.

VERTICAL SCALE WINDOW

Clicking on the **Vertical Scale** button of the **Trend** window opens a **Vertical Scale** window.

Scale Visible Check-box

Makes the trend graphic scale visible.

Number of Labels Field

Specifies the quantity of labels to be used in the trend graphic scale.

Grid Group Box

Divisions Field

Defines the number of division lines for the trend graphic grid.

Color Rec- tangle

Defines the color of the trend graphic grid. Click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button. If this field is not filled, there will not be any grids.

**Scale String
Format Group
Box**

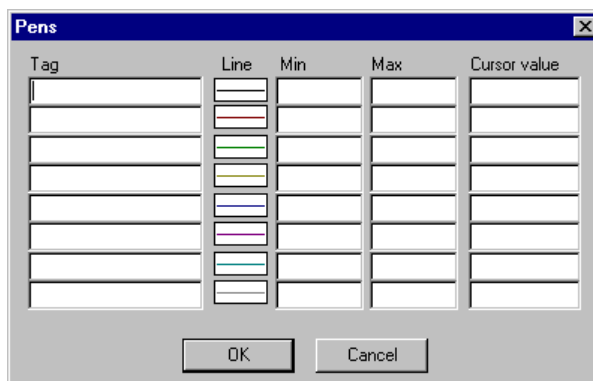
- Fonts Button** Accesses the **Fonts** window, which defines styles, sizes, colors, and font types. See **Fonts**.
- Width Field** Defines the number of digits of the string to be used in the vertical scale.
- Decimals Field** Defines the number of decimals of the string to be used in the vertical scale. *Example:* The values "Width = 3" and "Decimals = 2" indicate 3 digit numbers with 2 decimal places.

**Scale Range
Group Box**

- This value only displays the scale.
- Minimum Field** Specifies the minimum value of the trend graphic scale.
- Maximum Field** Specifies the maximum value of the trend graphic scale. The values to calculate the curves' positions are defined in the **Pen** window. You can use a numeric value or tag to dynamically change the vertical scale.

PENS WINDOW

Clicking on the Pens button of the Trend window opens a Pens window.




- Tag Field** The name of the tag that will be monitored in the trend. (e.g. MyTagY[1]) To do a generic trend window, you can use an indirect tag to define the tag to be monitored.
- Line Box** Defines the color of each trend curve. Click the **Color** rectangle to display a **Color** window. Double-click on the desired color or click the color and then the **OK** button. There can be up to eight differently colored lines on the Trend Graph.
- Min Field** The minimum value of the scale to draw the curve; it can be a numeric value or a tag.

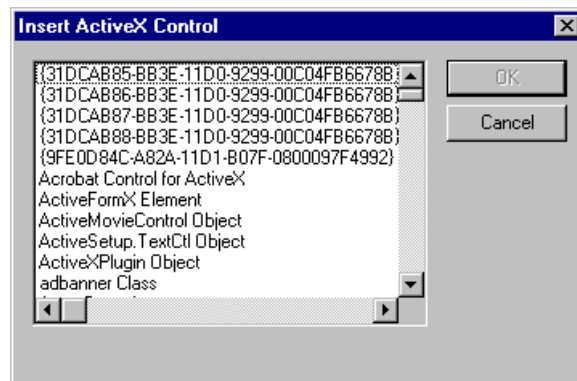
⇒ **The minimum value of each tag does not necessarily need to have the same minimum value as the trend graphic scale.**

Max Field The maximum value of scale to draw the curve; it can be a numeric value or a tag.

⇒ **The maximum value of each tag does not necessarily need to have the same maximum value as the trend graphic scale.**

Cursor Value Field Tag that receives the value of the intersection with the vertical cursor.

-  **ActiveX Control:** Opens a window with the list of all ActiveX components registered in your computer. You can select one of them and insert the object into the screen.

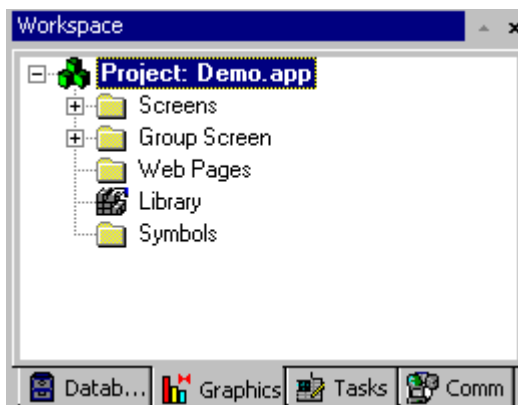


⇒ **You can use the function XGet(), XSet() and XRun() to read/write properties and to execute methods during the runtime.**

⇒ **This feature is not supported by Windows CE operating system.**

3.5 Workspace

The **Studio** Workspace is a user-friendly interface, which allows the user to quickly find a development module of the application (tags, screens, worksheets, etc). The application components are organized in a tree-view interface with each one having its own icon and customized description. This allows a quick association between the application component and its characteristics. The Workspace Window can be moved, resized, or hidden.



The **Workspace** window is divided into four tabs. Each tab, its folders, and component icons are described in length elsewhere in this manual. All folders and component icons are sensitive to right-clicking, which opens a menu with the principal actions of open, insert, delete, etc.

- **Database** Tab: Allows the user to access any available tag from the application and security system components. This tab has the following folders:
 - Application Tags
 - Classes
 - Shared Database
 - Internal Tags
 - Security
- **Graphics** Tab: Allows the user to access all screens and symbols in the application. This tab has the following folders and icon:
 - Screens
 - Group Screen
 - Web Pages
 - Library (icon)
 - Symbols
- **Tasks** Tab: Allows the user to access all tasks worksheets available in the application. This tab has the following folders:
 - Alarms
 - Trend
 - Recipes
 - Report
 - ODBC
 - Math
 - Scheduler

- **Communications Tab:** Allows the user to access all worksheets configured to establish communication with another device or software using available protocol. This tab has the following folders:
 - Drivers
 - OPC
 - TCP/IP
 - DDE

3.5.1 Database Tab



The **Database Tab** allows the user access to any available tag from the application and to the security system components. The **Database** tab has the following folders:

- **Application Tags:** This covers all tags created for the user and customized to the application.
- **Classes:** This covers all tags created for the user and customized to the application.
- **Shared Database:** This covers all tags shared between the **Studio** and the selected PC Based Control. If no PC Based Control is configured to share its database with the **Studio**, this folder will be empty.
- **Internal Tags:** This covers pre-defined tags, which have specific functions. These tags can not be edited for the user.
- **Security:** This covers Group Accounts and User Accounts, which comprise the application Security System.

In addition to presentations regarding the folders found on the **Database** tab, you should also refer to:

- Tag Syntax
- Tag Types and Tag Values
- Array Tags
- Indirect Tags
- Tag Properties

TAG SYNTAX

A tag name has the following restrictions:

- It can be composed of letters, numbers, and the character “_” (underscore).
- The following characters can not be used to compose a tag name
~`!@#\$%^&*()-=+\[\]{}<>?
- It must begin with a letter.
- Its maximum length is 32 characters (for a tag), or 16 characters (for a class member).
- You cannot have two tags with the same name.
- Tags are not case sensitive.

- The tag name must be different from internal tag names and math functions.
- For additional information, see *InduSoft* Scripting Language.


**CAUTION**

Studio does not differentiate between uppercase and lowercase characters. However, you should use both uppercase and lowercase characters to make names more clear (for example, *TankLevel* instead of *tanklevel*).

- ⇒ The character @ at the beginning of a tag name indicates that the tag will be used as an Indirect Tag in the application.

Tag Examples: temperature, pressure1, count, x

TAG FIELD SYNTAX

Fields are a set of parameters related to each tag in the database. Applications use these parameters at runtime as tag fields. Many of the parameters can be defined using the **Tag Properties** windows, accessed through the **Tag Properties**  icon on the Tag Properties Toolbar.

To access a tag field, use the following syntax: TagName->Field. You can access the following fields during runtime:

- **Min:** Minimum value for the tag in engineering units.
- **Max:** Maximum value for the tag in engineering units.

- ⇒ If the application tries to write a value outside of the specified range, the database does not accept it and a warning message is generated in the LogWin utility and in the OutPut window. If you do not wish to use these properties, simply enter 0 (zero) in the Min and Max fields.

- **Unit:** This field accepts any string up to 9 characters long related to the tag as a brief description or reference.
- **Description:** Tag description to help in application documentation.
- **Size:** Size of an array tag. If it is not an array, the size will be 0 (zero).
- **TimeStamp:** Last time/date when the tag value changed.
- **Quality:** Quality of the tag value. *Example:* GOOD:192 (C0 hex); Bad: 0. Used in communication protocols, which have algorithms to update this field (for example, OPC drivers).
- **B0-B31:** Bit 0 (zero) value of tag value through Bit 31 value of an integer tag value.

You can access the following alarm-related fields during runtime using the same syntax: TagName->Field:

- **Ack:** If greater than 0 (zero), an alarm associated with the tag is not acknowledged yet.
- **AlrDisable:** 1=disable, 0=enable
- **AlrStatus:** If greater than 0 (zero), at least one alarm associated with the tag is active. 0 means no active alarms.
- **Alarm Types:** HiHi, Hi, Lo, LoLo, Rate, Dev+, Dev-

All fields may be read by the application with the following syntax: Tag->field. *Examples:* level-Max, Temp->Unit, pv101->HiHiLimit.

**CAUTION**

The tag fields can not be used to configure Alarm worksheets nor Trend worksheets.

SUMMARY: You can access the following fields during runtime.

| Field Name | Boolean Tag | Integer Tag | Real Tag | String Tag | Allows change on the fly |
|-------------|-------------|-------------|----------|------------|--------------------------|
| Min | no | yes | yes | no | no |
| Max | no | yes | yes | no | no |
| Unit | yes | yes | yes | yes | no |
| Description | yes | yes | yes | yes | no |
| Size | yes | yes | yes | yes | no |
| TimeStamp | yes | yes | yes | yes | no* |
| Quality | yes | yes | yes | yes | no* |
| B0-B31 | no | yes | no | no | yes |
| Ack | yes | yes | yes | yes | no* |
| AlrDisable | yes | yes | yes | no | yes |
| AlrStatus | yes | yes | yes | no | no* |
| HiHiLimit | yes | yes | yes | no | yes |
| HiLimit | yes | yes | yes | no | yes |
| LoLimit | yes | yes | yes | no | yes |
| LoLoLimit | yes | yes | yes | no | yes |
| DevSetpoint | no | yes | yes | no | yes |
| Dev+Limit | no | yes | yes | no | yes |
| Dev-Limit | no | yes | yes | no | yes |
| RateLimit | no | yes | yes | no | yes |
| HiHi | no | yes | yes | no | no |
| Hi | yes | yes | yes | no | no |
| Lo | yes | yes | yes | no | no |
| LoLo | yes | yes | yes | no | no |
| Dev+ | no | yes | yes | no | no |
| Dev- | no | yes | yes | no | no |
| Rate | no | yes | yes | no | no |

**CAUTION**

Although the system allows the above fields marked with an asterisk (*) to be changed on the fly, it is not advisable to do so and should not be tried. This includes **AlrStatus**, **TimeStamp**, **Quality**, and **Ack**.






TAG TYPES

Tags can be communication points with field equipment, results of calculations, alarm points, and so forth. In **Studio**, all tags are organized on the **Database** tab in folders according to their origin: application, internal, or shared. There is also a folder for compound-tags called *classes*.

- **Application Tags:** In **Studio**, user-created tags are called *application tags*. These tags are created for displays, tags that read from and write to field equipment, tags used for control, auxiliary tags to perform mathematical calculations, and so forth.
- **Internal Tags:** Tags predefined by **Studio** are called *internal tags*. The internal tags have predetermined functions (time, date, acknowledge alarms, storage of the logged-on user name and so forth) and can not be deleted nor modified. However, their values can be accessed from any **Studio** task.
- **Shared Tags:** Tags created in a PC Based Control software and imported into the **Studio** environment are called *shared tags*. Shared tags can not be edited in the **Studio** environment, but they can be modified in the PC-based control software, used and updated to the **Studio** database. Thus they can be configured in any **Studio** task like any other tag.
- **Classes:** *Classes* are structures that allow for a high level of encapsulation in the application database. When a class-type tag is created, it does not contain just a single value, but a whole set of values. You can create class-type tags by grouping simple tags, called *members*. The maximum number of members for any class depends on product specification.

TAG VALUES

The value of a tag can be one of the following types. The icons given below can be found with their respective tag types in the folders on the **Database** tag.

-  **Boolean:** Boolean or digital variable (0 or 1).
-  **Integer** - Integer number (it may be positive, negative, or zero). Equivalent to "C" type long integer. *Examples:* 0, 5, -200.
-  **Real** - Real number internally stored as a double word. Equivalent to "C" type double.
-  **String** (ASCII text) - Character string up to 255 characters that holds letters, numbers, or special characters. *Examples:* Recipe product X123, 01/01/90, *** On ***.
-  **Class** – A user-defined, compound tag.

ARRAY TAGS

Studio tags can hold a single value or an array of values. An *array* tag is a set of tags with the same name; it is identified by *indexes* (a matrix of *n* lines and 1 column). The maximum array size depends on product specification. The syntax used to access an array tag is `<ArrayTagName>[ArrayIndex]`. *Example:* `tank[1]`, `tank[2]`, `tank[3]`, `tank[500]`.



CAUTION

The maximum index for each array tag is configured in the *size* column on any Datasheet. When size “*n*” is configured, it means that the array tag has positions from 0 to “*n*.” For example, if `TagA` size is 3, tags could be: `TagA[1]`, `TagA[2]`, and `TagA[3]`. It is not advisable to use the position [0] (zero) with any array tag because the system uses it with internal error configurations.

Use the array tag whenever possible because it optimizes memory use and simplifies the configuration task. Suppose, for example, that you want to have a display to monitor each tank. Using array tags makes it possible to configure a single display that contains tags linked to any tank.

Examples using the tag `tk` as an index that contains the number of the desired tank:

```
pressure[tk], temperature[tk], temperature[tk + 1].
```

An array index may be a tag, a numeric value, or an expression with the arithmetic operator “+”.

⇒ **To refer to an array that has an index with the arithmetic operation “+”, you must use the following syntax: `<ArrayTagName>[<NumValue1> + <NumValue2>]`, where `<NumValue1>` and `<NumValue2>` can be an integer tag or a numerical constant. *Examples:* `temperature[tk+2]`, `temperature[tk+6]`, `temperature[TagA + TagB]`.**

Using array tags in any **Studio** task can save a lot of application development time. Suppose that you need tag points related to the temperature of four tanks.

The conventional configuration method is:

```
temperature1    high temperature on tank 1
temperature2    high temperature on tank 2
temperature3    high temperature on tank 3
temperature4    high temperature on tank 4
```

Using array tags simplifies this task:

```
temperature[j]    high temperature on tank {j}
```

⇒ **When you create a four-position array tag, the system creates five positions (from 0 to 4). *Example:* `tag_example[15]`, `//start position=0`, `//end position=15`. Therefore, the `tag_example[15]` array has 16 elements.**

INDIRECT TAGS

Studio supports indirect access to tags in the database. For example, consider a tag X of the **string** type. This tag can hold the name of any other tag in the database (that is, it can provide a pointer to any other type of tag, including a class type). The syntax for an indirect tag is straightforward: **@<IndirectTagName>**. For example, assume that a tag named X holds a "TEMP" string. Reading and/or writing to **@X** provides access to the value of the *TEMP* variable.

⇒ **Any tag that is created as a string type is a potential indirect tag (pointer).**

To refer to a class-type tag, it is possible to declare a string-type tag, which will point to a class tag.

Examples:

- Class - TANK with members **Level**
- Tag - TK of the **class:TANK** type
- Tag - XCLASS of the string type

To access the **TK.Level** value, it is necessary to store within the XCLASS tag the value "TK.Level" and use the syntax **@XCLASS**. It is also possible to refer a member of a class-type tag directly, identifying a class-type that will point to a class member.

Examples:

- Class - TANK with members **Level**
- Tag - TK of the **class:TANK** type
- Tag - XCLASS of the string type

To access the **TK.Level** value, it is necessary to store within the XCLASS tag the value "TK" and use the syntax **@XCLASS.Level**.

When you create tags for indirect use, place an **@X** in the tag column rather than creating them as strings. For the type, write the type of tag for which a reference is being created. Follow the XCLASS example: **@Z Integer**, **@X Class:TANK**.

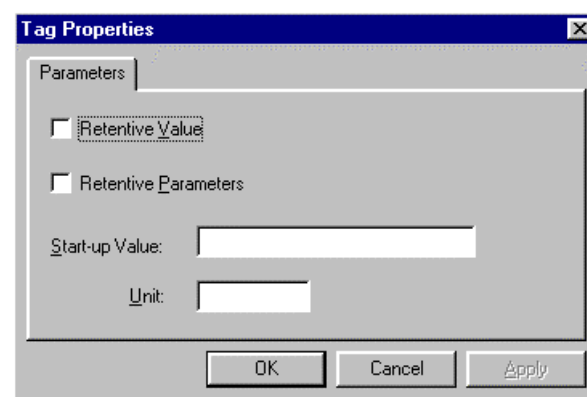
TAGS PROPERTIES

Each tag type has four properties (The tags properties can be configured by the **Tags Properties** icon from the **Tags Properties** Tool Bar):

- Parameters
- Alarms Properties
- History Properties

PARAMETERS

PARAMETERS OF THE STRING TYPE TAGS



- **Retentive Value Check-box:** Continually saves the tag value in case the system shuts down, so that it restarts from the last saved value.



CAUTION

Avoid the use of this option for tags with changing values; this causes frequent hard disk access, slowing performance.

- **Retentive Parameters Check-box:** Holds all runtime changes in the tag fields.
- **Start-up Value Field:** Tag value for the system load. The tag assumes this value if you disable the **Retentive Value** option.
- **Unit Field:** This field accepts any string (up to 9 characters) related to the tag as a brief description or reference. Accessible during runtime.



CAUTION

The system will not accept writing values outside the range defined in the Min and Max fields. Also, a message will be generated in the LOGWIN module, indicating that the system tried to write a value out of the defined range.

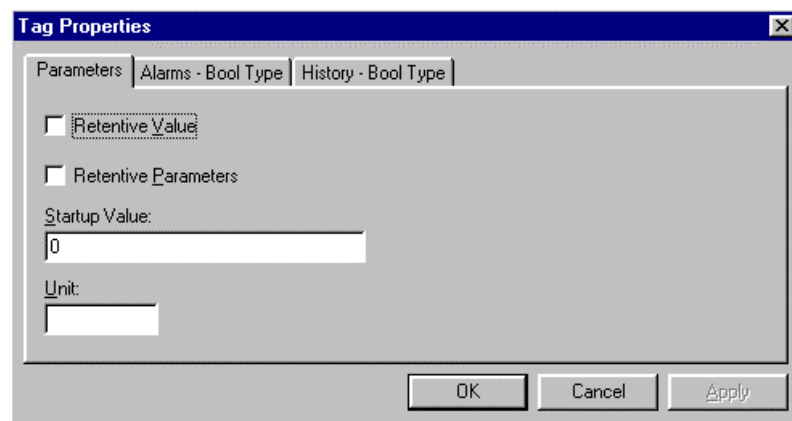
PARAMETERS OF THE INTEGER AND REAL TYPE TAGS

The screenshot shows the 'Tag Properties' dialog box with the 'Parameters' tab selected. The 'Retentive Value' checkbox is checked, while 'Retentive Parameters' is unchecked. The 'Start-up Value' is set to 0. In the 'Engineering Units' section, 'Min' and 'Max' are 0, and the 'Unit' is '1234567'. In the 'Signal Conditioning' section, both 'Dead Band' and 'Smoothing' are unchecked, and the 'Dead Band' value is 0.

Some of these field descriptions are the same as those described above. Those that are different are described below.

- **Engineering Units Group Box**
 - **Min. Field** - Minimum value for the tag in engineering units. Accessible during runtime.
 - **Max. Field** - Maximum value for the tag in engineering units. Accessible during runtime.
 - **Unit Field** - This field accepts any string (up to 9 characters) related to the tag as a brief description or reference. Accessible during runtime.
- **Signal Conditioning Group Box**
 - **Dead Band Check-box** - Inserts the *dead band* value of a tag. Dead band value is a variation around a central value of the tag, which is not recognized for alarms.
 - **Smoothing Check-box** - Reduces the rate of change of the tag's values and can be used only for integer and real tags. *Example:* The Smoothing option is selected for the LEVEL1 tag that contains the value = 50. If in the next search the LEVEL1 changes to 60, the system will store the average of 50 + 60 in the database, so the new value = 55.

PARAMETERS OF THE BOOLEAN TYPE TAGS



These check-box and field descriptions are the same as those described above.

ALARMS PROPERTIES

Through the **Tag Properties** window in the Tag Properties Toolbar, you can view the configured alarms for a selected tag. This command is disabled if there are open alarm worksheets. Before using these windows, you should have already created the alarm groups.

- **ALARM TYPES**

Alarms types are as follows:

- **HiHi** – A Very High alarm is present.
- **Hi** - A High alarm is present.
- **Lo** – A Low alarm is present.
- **LoLo** – A Very Low alarm is present.
- **Rate** – An alarm based on rate of change is present.
- **Deviation** – An alarm based on deviation from a given set point is present.

Example of a Deviation Alarm:

| | | |
|----------------------------|---|-----|
| SetPoint | = | 50 |
| Deviation + | = | 5 |
| Deviation - | = | 5 |
| Deviation Dead Band | = | 0.5 |

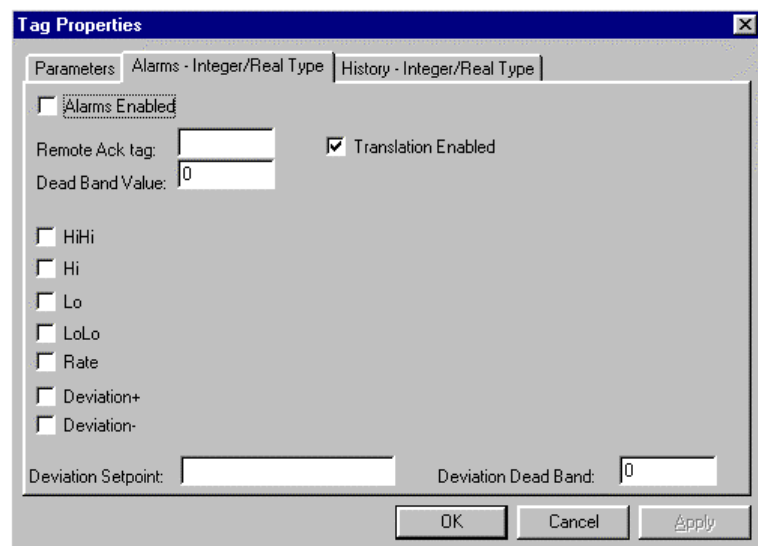
An alarm will be generated when **temp > 55.5** or **temp < 44.5**. The return to the normal will occur when **temp > 45** or **temp < 55**.

- **ALARM LIMITS**

Limits are as follows:

- **HiHiLimit** – When creating Very High alarms in the **Tag Properties** window, this field holds the limits. Accessible during runtime and it can be used during modifications on the fly.
- **HiLimit** – When creating High alarms in the **Tag Properties** window, this field holds the limits. Accessible during runtime and it can be used during modifications on the fly.
- **LoLimit** – When creating Low alarms in the **Tag Properties** window, this field holds the limits. Accessible during runtime and it can be used during modifications on the fly.
- **LoLoLimit** – When creating Very Low alarms in the **Tag Properties** window, this field holds the limits. Accessible during runtime and it can be used during modifications on the fly.
- **DevSetpoint** – Reference point for a tag value deviation that triggers an alarm. The alarm message is defined in the **Tag Properties** window or on an Alarm worksheet. Accessible during runtime.
- **Dev+Limit** - Limit deviation to a value higher than the **DevSetpoint** in tag value that triggers an alarm. The alarm message is defined in the **Tag Properties** window or on an **Alarm** worksheet. Accessible during runtime.
- **Dev-Limit** – Limit deviation to a value lower than the **DevSetpoint** in tag value that triggers an alarm. The alarm message is defined in the **Tag Properties** window or on an **Alarm** worksheet. Accessible during runtime.
- **RateLimit** – Limit of rate variation in tag value that triggers an alarm. The alarm message is defined in the **Tag Properties** window or on an **Alarm** worksheet. Accessible during runtime.

ALARMS FOR THE INTEGER AND REAL TYPE TAGS



- **Alarms Enabled Check-box:** Enables checking according to configuration.
- **Remote Ack tag Field:** The tag in this field acknowledges this alarm.
- **Dead Band Value Field:** Value of the filter for alarms generation.

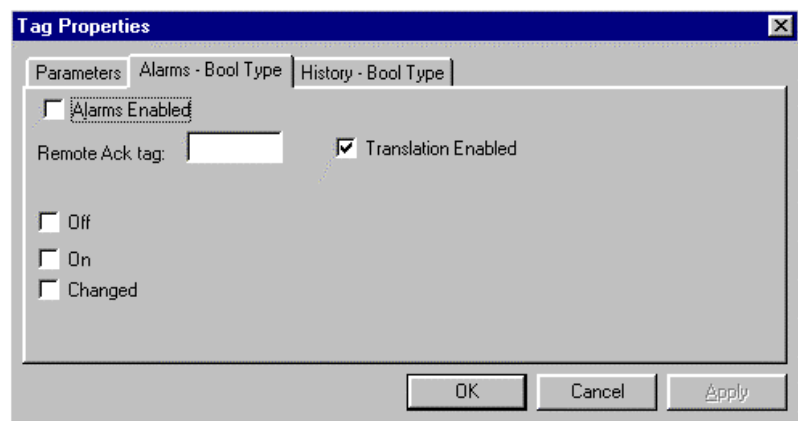
Example: If the TEMP1 tag is configured with an alarm with **Limit = 90** and **Dead Band = 5**, an alarm will be generated when $TEMP1 \geq 95$. The return to normal will occur when $TEMP1 < 90$.

- **Translation Enabled Check-box:** Enables the translation of messages if they were defined on the **Options** tab in the **Project Settings** window under **Project** on the Main Menu Bar. For additional information about translations, see **Translation Tools**.

⇒ **Alarm messages with the *Translation Enabled* attribute selected are saved in a file called “Alarm.TXT” in the \DATABASE\ directory of your application.**

- **HiHi Check-box:** If checked, a Very High alarm is present. Accessible during runtime.
- **Hi (HiLimit) Check-box:** If checked, a High alarm is present. Accessible during runtime.
- **Lo (LoLimit) Check-box:** If checked, a Low alarm is present. Accessible during runtime.
- **LoLo (LoLoLimit) Check-box:** If checked, a Very Low alarm is present. Accessible during runtime.
- **Rate (RateLimit) Check-box:** If checked, a Rate alarm is present. Accessible during runtime.
- **Deviation + Check-box:** If checked, a Deviation alarm is present. Accessible during runtime.
- **Deviation - Check-box:** If checked, a Deviation alarm is present. Accessible during runtime.
- **Deviation SetPoint Field:** Reference point for the deviation. Accessible during runtime.
- **Deviation Dead Band Field:** Reference value for the deviation.
- **Array Index:** Allows configuring the alarm for each position of the array tag. This field displays only for array tags.
- **Limit, Message, Group, Priority, Selection Fields:** Only enabled when a **Limit** field is selected (**HiHi, Hi, Lo, LoLo**, etc.).
 - **Limit Field** - Limit to trigger the alarm.
 - **Message Field** - Alarm message that displays.
 - **Group Field** - Number according to the alarm groups previously created.
 - **Priority Field** - Indicates the priority within a group; indicated by an integer (0 to 255). The tag with a higher priority must have a higher **Priority** value.
 - **Selection Field** - User-defined string that works as a filter in the alarm summary objects.

ALARMS FOR THE BOOLEAN TYPE TAGS



- **Alarms Enabled Check-box:** Enables checking according to configuration.
- **Translation Enabled Check-box:** Enables the translation of messages if they were defined on the **Options** tab in the **Project Settings** window under **Project** on the Main Menu Bar. For additional information about translations, see **Translation Tools**.

⇒ **Alarm messages with the *Translation Enabled* attribute selected are saved in a file called "Alarm.TXT" in the \DATABASE\ directory of your application.**

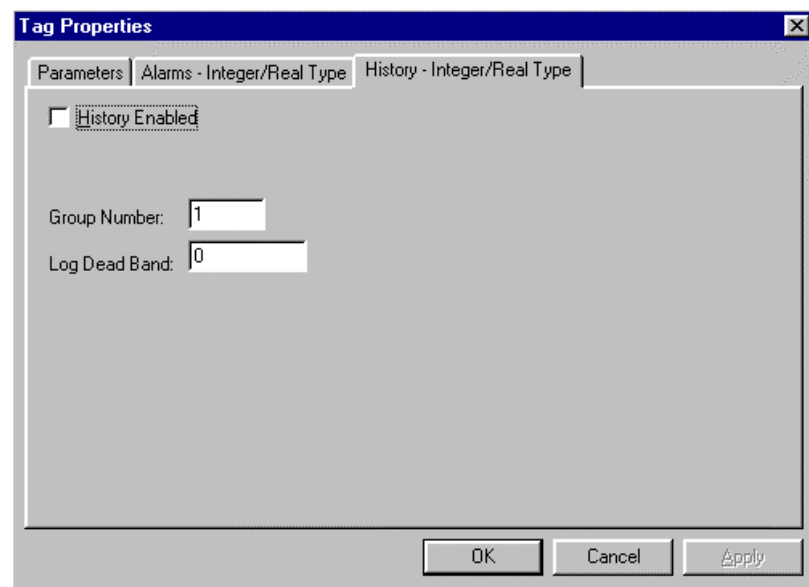
- **Off Check-box:** Always generates an alarm message when the tag value is **0** (zero).
- **On Check-box:** Always generates an alarm message when the tag value is **1**.
- **Changed Check-box:** Always generates an alarm message when the tag value has changed.

HISTORY PROPERTIES

Through the **Tag Properties** window in the Tag Properties Toolbar, you can view the history for a selected tag. This command is disabled if there are open trend worksheets. Before using these windows, you should have already created the trend groups.

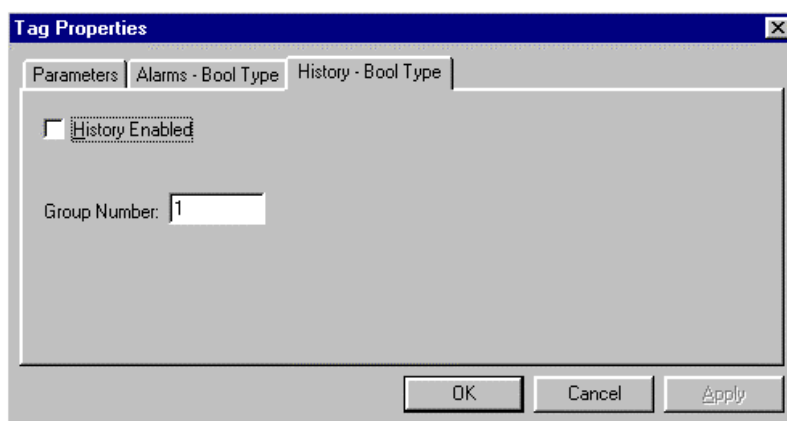
⇒ **String type tags are not supported by History. See Recipes to store string values.**

INTEGER AND REAL TYPE HISTORY



- **History Enabled Check-box:** Enables storage of the selected tag value samples.
- **Group Number Field:** Defines the group number to which this tag is associated.
- **Log Dead Band Field:** Value sample will be taken when the variation value is equal to or greater than the **Log Dead Band**.


BOOLEAN TYPE HISTORY




- **History Enabled Check-box:** Enables storage of the selected tag value samples.
- **Group Number Field:** Defines the group number to which this tag is associated.

APPLICATION TAGS


The **Application Tags** folder contains customized application tags created by the operator in the *Studio*. Application tags are tags created for displays, tags that read from and write to field equipment, tags used for control, auxiliary tags to perform mathematical calculations, and so forth.

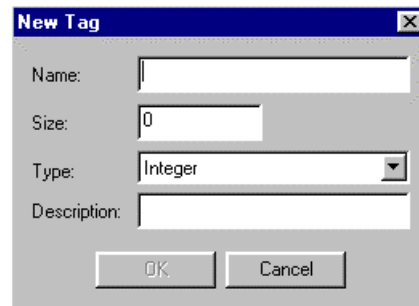
- ⇒ **Right-click on the Application Tags folder or Datasheet View ** and select the Refresh option to update your application tags database files. The tags are read from the system to the *Studio* environment. This option should be used to guarantee that viewed tags are the same as those in the internal file. However, it is not necessary to do this often.

**CAUTION**

Before deleting a tag, it is strongly advised to use the Object Finder  icon on the Tag Properties Toolbar to verify if the tag is being used in another application (screens, math sheets, etc.). If a tag is deleted from the application database and it is configured in another task, it will cause a compiling error and the application will function poorly.

CREATING NEW APPLICATION TAGS

Through the **Database** tab, you can select or create new Application Tags. To create a new Application Tag, right-click on the **Application Tags** folder, the **Tag List** sub-folder, or **Datasheet View** . Or you can select the **Tag** option under **Insert** on the Main Menu Bar.





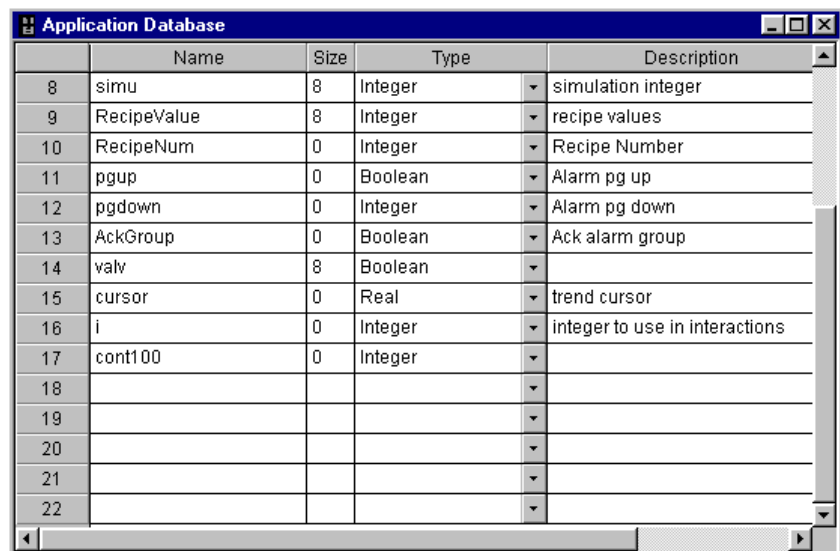
- **Name Field:** Type a tag name. The maximum name size is 32 characters; the first character must be a letter.
- **Size Field:** Type a tag size. It may have up to 256 positions (0-255); position 0 should not be used in the application. A size greater than 0 implies that the tag is an array.
- **Type Drop-list:** Select a tag type. Besides the standard tag types (Boolean, Integer, Real, String), you can define new types as structures formed by the standard types, that is, the classes.
- **Description Field:** Fill this field with a remark for documentation purposes.
- **Web Data:** This field has only two options, Local and Server. If you chose Server it means the information in this tag will be able to be shared over the net. If you chose Local the information can't be shared over the net. Neither selection affects an application that is not using the web capabilities. If your application is using the web capabilities then any object property, which uses a tag with Local in the Web Data field, will not work properly.

⇒ **You cannot create a tag with the same name of another tag that already exists.**

VIEWING AND EDITING APPLICATION TAG PROPERTIES

Application tags can be edited on the Tag Property window or the Application Tag Datasheet.

- **TAG PROPERTY WINDOW:** The **Tag Property** window can be accessed by clicking on the **Tag Properties**  icon on the Tag Properties Toolbar once the tag name appears in the **Tag name** field. Or access the **Tag Property** window by double-clicking on the **Tag Name** in the **Tag List** sub-folder in the **Application Tags** folder.
- **APPLICATION TAG DATASHEET:** To open the **Datasheet**, click the **Datasheet View**  in the **Application Tags** folder. The **Datasheet** is composed of four columns: name, size, type, and description.



| | Name | Size | Type | Description |
|----|-------------|------|---------|--------------------------------|
| 8 | simu | 8 | Integer | simulation integer |
| 9 | RecipeValue | 8 | Integer | recipe values |
| 10 | RecipeNum | 0 | Integer | Recipe Number |
| 11 | pgup | 0 | Boolean | Alarm pg up |
| 12 | pgdown | 0 | Integer | Alarm pg down |
| 13 | AckGroup | 0 | Boolean | Ack alarm group |
| 14 | valv | 8 | Boolean | |
| 15 | cursor | 0 | Real | trend cursor |
| 16 | i | 0 | Integer | integer to use in interactions |
| 17 | cont100 | 0 | Integer | |
| 18 | | | | |
| 19 | | | | |
| 20 | | | | |
| 21 | | | | |
| 22 | | | | |

The **Datasheet** allows you to create, modify, or delete any tag and its viewed properties (including its name). The table is sensitive to right-clicking, which allows the use of Windows default editing commands to Cut (CTRL+ X), Copy (CTRL+ C), and Paste (CTRL+ V) any tag and its properties. A typed selection can also be edited by double-clicking to highlight it and then right-clicking to get an options window with cut, copy, paste, delete. Additionally, the last modification in a field can be undone.

CLASSES

The **Classes** folder contains all the classes created with an application and allows the viewing and editing of the classes and their members. Classes are compound tags that are user-defined data type structures in addition to the standard data types of Integer, Real, Boolean, and String. Classes allow for a high level of encapsulation in the application database. A Class-type tag has not just one value, but a whole set of values about the class (group of members).

Defining a class means defining the group of members and their types. The **members** of a class are variables that can hold values of an object with particular characteristics. Thus, the definition of a class is very useful when you have an application with a repeating group of variables.

⇒ When a Class folder is made, a Class  icon also appears in the Tag List sub-folder in the Application Tags folder.

To access the members of a class tag, use a period (.) as a separator in the tag syntax <TagName>.<MemberName>. *Example:* tk.LEV and tk.TMP. If tag tk is an array, the syntax would be <ArrayTagName>[<ArrayIndex>].<MemberName>.

Example:

tk[1].LEV, tk[n].TMP.

This section about Classes is divided into the following topics:

- Creating New Classes
- Viewing and Editing Class Properties

CREATING NEW CLASSES

When a **class**-type tag is created, it does not contain a single value, but a whole set of values associated with the class. You can create **class**-type tags by grouping simple tags, called *members*. The maximum number of members for any class depends on product specification. Members of a class can hold standard values (Integer, Real, Boolean, String) as previously described.

To create a new class TANK, define its members as follows:

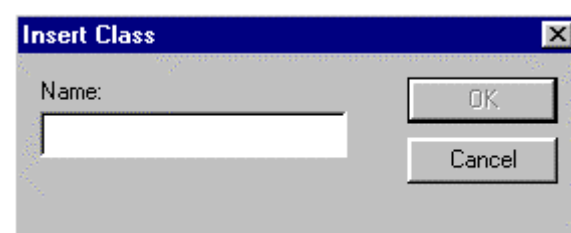
| | |
|-------------|---------|
| Level | Integer |
| Temperature | Integer |
| Pressure | Real |
| Valve | Boolean |

Through the **Database** tab, you can select or create new Classes.

- ◆ To create a new Class, right-click on the **Classes** folder, the **Members List** sub-folder, or **Datasheet View**  in the **Classes** folder.

These methods open an **Insert Class** window.

Or you can select the **Class** option under **Insert** on the Main Menu Bar. It is also possible to create a new class tag in the **Application Tags** folder.

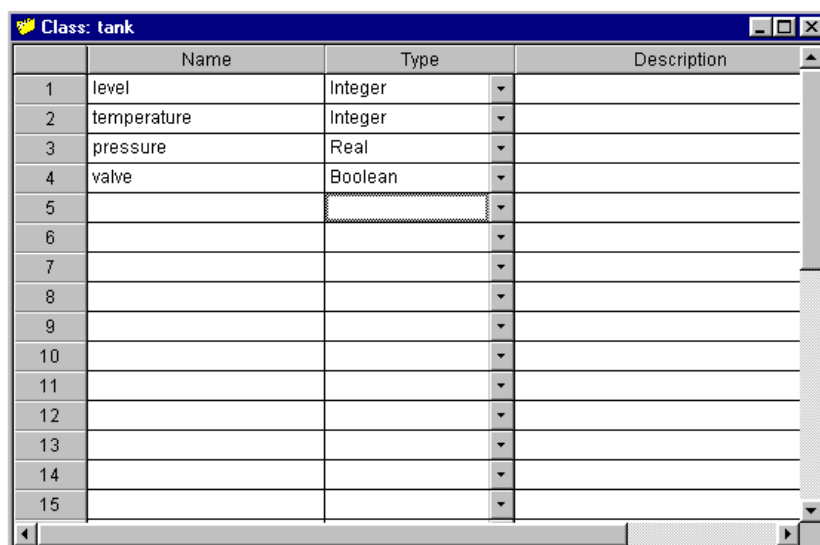


- **Name Field:** Type the name of the new class.

⇒ **Two classes cannot be created with the same name. Also, shared tags and internal tags can not be configured as class-types.**

CLASS DATASHEET

Once a new Class name is assigned in the **Insert Class** window, a **Class Datasheet** appears.



| | Name | Type | Description |
|----|-------------|---------|-------------|
| 1 | level | Integer | |
| 2 | temperature | Integer | |
| 3 | pressure | Real | |
| 4 | valve | Boolean | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| 10 | | | |
| 11 | | | |
| 12 | | | |
| 13 | | | |
| 14 | | | |
| 15 | | | |

- **Name Field:** Type a tag name with a maximum name size of 32 characters; the first character must be a letter.
- **Type Drop-list:** Select a tag type (Boolean, Integer, Real, String).
- **Description Field:** Fill this field with a remark for documentation purposes.



CAUTION

A class can hold up to 32 members.

- ⇒ **Members of a Class can not be of another class type.**
- ⇒ **If a class name already exists, it can not be used for the name for another class. However, it is possible to create members with the same name in different classes.**

VIEWING AND EDITING CLASS PROPERTIES

Classes can't be edited on the Tag Property window but they can be edited on the Classes Datasheet. Class folders, as they appear in the **Tag List** sub-folder in the **Tag Application** folder, can be edited as an Application Tag.

The **Class Datasheet** allows you to create, modify, or delete any Class members and its viewed properties. The table is sensitive to right-clicking, which allows the use of Windows default editing commands to Cut (CTRL+ X), Copy (CTRL+ C), and Paste (CTRL+ V) any tag and its properties. A typed selection can also be edited by double-clicking to highlight it and then right-clicking to get an options window with cut, copy, paste, delete. Additionally, the last modification in a field can be undone.

A delete option appears when right-clicking on a **Class** folder. If chosen, this deletes the class and all its members. This option is disabled if any Runtime Task is running. It will not delete a class which is associated to any tag.

SHARED DATABASE

The **Shared Database** folder contains tags shared between the **Studio** and the selected PC-based control software.

Shared tags are used when configuring the interface between **Studio** and PC-based control software. These tags must be created and modified in the PC-Based and Control Software and will automatically import in **Studio** under the following conditions:


- | | |
|---|---|
| 0 | You have started Studio . |
| 1 | You have right-clicked on the Shared Tags folder. This refreshes (updates) the database. |

⇒ **Each PC Based Control software has its own interface characteristics and conditions that allow Studio to import its tags. For example, in some cases it is required that the PC-based control software be running its application in order for the database tags to be imported by Studio.**

If a PC Based Control software is not associated with the application, the **Datasheet View** and the **Tag List** sub-folder in the **Shared Database** folder will be empty.


VIEWING AND EDITING SHARED TAGS


The shared tags can't be edited in the **Studio** environment, but they can be modified in the PC Based Control software used and updated to the **Studio** database. So, they can be configured in any **Studio** task like any other tag. Shared tags are view-only on the **Tag Property** window and the Shared Tag Datasheet.

- **TAG PROPERTY WINDOW:** The **Tag Property** window can be accessed by clicking on the **Tag Properties**  icon on the Tag Properties Toolbar once the tag name appears in the **Tag name** field. Or access the **Tag Property** window by double-clicking on the **Tag Name** in the **Tag List** sub-folder in the **Application Tags** folder.
- **SHARED TAG DATASHEET:** The Shared Tag Datasheet of four columns (Name, Size, Type, and Description) permits very little editing of tags and properties. This Shared Tag Datasheet does not allow the actions create, modify, or delete. It is used for viewing shared tags. It will be necessary to change tag characteristics with the specific PC-based control software editor.



CAUTION

Before deleting a tag, it is strongly advised to use the **Object Finder**  icon on Tag Properties Toolbar to verify if the tag is being used in another application (screens, math sheets, etc). If a tag is deleted from the application database and it is configured in another task, it will cause a compiling error and the application will function poorly.

- ⇒ Right-click on the Shared Database folder or Datasheet View  and select the Refresh option to update your last “version” of the PC-based control software tags database. In order to change the PC-based control tags database (create new tag, delete tags, change tag properties), this command must be activated to update the *Studio* Shared Database.

INTERNAL (PREDETERMINED) TAGS

Tags predetermined by *Studio* are called *internal tags*. Internal tags have predetermined functions (time, date, acknowledge alarms, storage of the logged user and so forth) and can not be deleted nor modified. However, their values can be accessed from any *Studio* task.

Examples:

| | |
|------|--|
| Date | Holds the current date in string format. |
| Time | Holds the current time in string format. |

A list of **Internal Tags** of the *InduSoft's* Scripting Language with their respective properties and descriptions can be found in the *InduSoft* Scripting Language chapter. Internal tags can not be edited, but they can be copied and used elsewhere.


INTERNAL TAGS DATASHEET

The **Internal Tags Datasheet** of four columns (Name, Size, Type, and Description) permits very little editing of tags and properties. This **Internal Tags Datasheet** does not allow the actions create, modify, or delete. It is used for viewing Internal Tags. These tags can't be edited (modified nor deleted), but they can be used as any other application tags to configure application tasks and objects.



CAUTION

Most internal tags are view-only. To change the time, for instance, use the proper math function to set the system time rather than writing to the internal time tag.

- ⇒ Right-click on the Internal Tags folder or Datasheet View  and select the Refresh option to update your *Studio* Shared Database with the Internal Tags files.

SECURITY SYSTEM

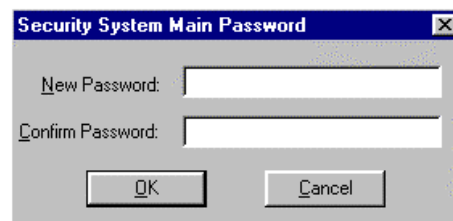
The **Security** folder allows you to define groups and users as well as their access privileges to **Studio** tools and to the application. Through the **Database** tab, you can select or create new groups and users. To access the **Security System** window right-click on the **Security** folder.



- **Enable Security System Check-box:** Enables the **Studio** Security System.
- **Main Password Button:** Opens the **Security System Main Password** window (see below).
- **Accounts Group Box**
 - **Groups Button** - Opens a **Groups** window (see below).
 - **Users Button** - Opens a **Users** window (see below).

PASSWORD

The **Main Password** button of the **Security System** window opens the **Password** window where you define a password for accessing the **Studio** Security System.



- **New Password Field:** Type a new password here to define it.
- **Confirm Password Field:** Confirm the password you typed in the **New Password** Field by typing it again and clicking on **OK**. If the password is different, the system asks you to type it again.



CAUTION

After you define your password, you will need to use it each time you access the Security System, so it is mandatory that you remember it.

GROUPS

The **Groups Account Button** of the **Security System** window opens the Group Account window in which you can create and maintain **user groups**. In this window, you enable/disable operations and set the range level. Groups can also be accessed by opening the **Groups** folder within the **Security** folder or by selecting the **Security Group** option under **Insert** on the Main Menu Bar. Select a specific group to view.



- **Group Account Drop-list** : Select the group to which the user belongs from the drop-list.
- **Security Level Development Group Box**: Defines the security level of each group (0 to 255). Any object for data input in the Display Screen (such as input commands, sliders, or screens) has a **Security Level** field. If the object level is not in the group security scale logged in at the moment, then the object is disabled. A level **0** (zero) means that the object is always enabled.
- **Security Level Runtime Group Box**: Defines the security level of each group (0 to 255). Any object for data input in the Display Screen (such as input commands, sliders, or screens) has a **Security Level** field. If the object level is not in the group security scale logged in at the moment, then the object is disabled. A level **0** (zero) means that the object is always enabled.
- **Engineering Access Group Box**: Lists Engineering (development) tasks that can be accessed when a user in this group is logged on. Includes check-boxes for **Create, modify tags**; **Project Settings**; **Drivers, Data Sources**; **Network Configuration**.



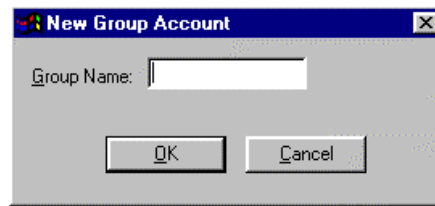
CAUTION

The security level can also be set to each document (worksheets and displays) to protect them in the development environment. This refers to the Engineering Access box.

- **Engineering Access Group Box**: Lists Engineering (development) tasks that can be accessed when a user in this group is logged on. Includes check-boxes for **Create, modify tags**; **Project Settings**; **Drivers, Data Sources**; **Network Configuration**.
- **Runtime Access Group Box**: Runtime modules that this user group can access. Includes check-boxes for Start App, Close App, Database Spy (white), Task switch enabled, CreateUser enabled.

⇒ You cannot delete the Guest group (the default logged group).


- **New Button:** Opens the **New Group Account** window, in which you can create a new group.



- **Delete Button:** Deletes the currently selected user group.

SECURITY ACCESS LEVEL

In the Group Account window, it is possible to set a range of access values in the Security Level- Development group box. Each group can be assigned its own range of values.

When any *InduSoft* worksheet is opened (Alarm, Math, Recipe, Report, Scheduler, TCP Client, Trend, and those not available on CE : DDE Client, OPC Client, and ODBC), it is possible to set an access range to THAT worksheet.

Click on any part of the worksheet body to activate the Access Level option under Edit on the Main Menu Bar. When Access Level is selected, a window opens in which an Access Level number can be assigned. This means that to edit the worksheet again, it would be necessary that the worksheet have an Access Level within the Security Level – Development group box range of the user logged onto the system.

For example, UserA of GroupA has a Security Access Level range of 0-10, UserB of GroupB has a Security Access Level range of 5-15.

To continue the example:

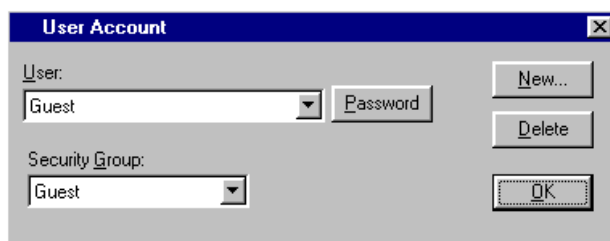
| | | |
|-------------------------------------|---|----|
| Math Worksheet 001 has Access Level | = | 1 |
| Math Worksheet 002 has Access Level | = | 7 |
| Math Worksheet 002 has Access Level | = | 12 |
| Math Worksheet 002 has Access Level | = | 20 |

In this situation, only UserA can access Math Worksheet 001, both UserA and UserB can access Math Worksheet 002, only UserB can access Math Worksheet 003, and neither UserA nor UserB can access Math Worksheet 004.

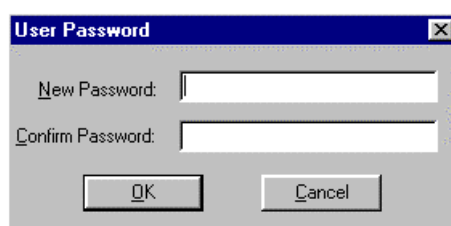
USERS

The **User Account Button** of the **Security System** window opens the **User Account** window in which you create and maintain accounts for application users. Define the application users that will be in each group in the **Group Account** list.

Users can also be accessed by opening the **Users** folder within the **Security** folder or by selecting the **User** option under **Insert** on the Main Menu Bar. Select a specific user to view.



- **User Drop-list:** Lists application users in a drop-list.
- **Security Group Drop-list:** Lists application groups.
- **New Button:** Opens the **New User Account** window to create a new user.
- **Delete Button:** Deletes the selected user.
- **Password Button:** Opens the **User Password** window, in which you can define a password for the user.



- **New Password** – Enter a password to define it.
- **Confirm Password** - Confirm the password you typed in the **New Password** field by typing it again and clicking on **OK**. If the password is different, the system asks you to retype it.

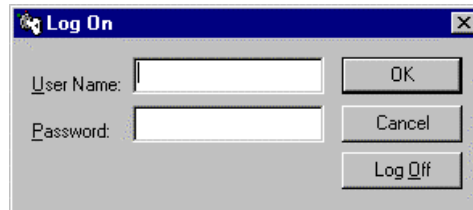
GUEST USER

After you initialize *Studio*, a default user is logged on the Guest user. If no user is logged on or the current user has logged off, Guest user is automatically logged on.

The Guest group has default privileges. Since the installation parameters of the Guest group leave all tasks enabled, you should change it and set as few privileges as you want for a start up procedure.

LOG ON/LOG OFF

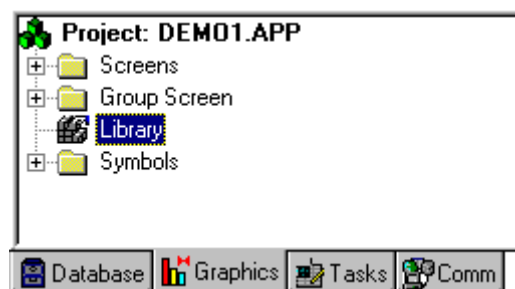
This utility is used to log users on and off. The user names and passwords are defined through the **Security folder** on the **Database** tab. You can also log on or off by using the **InduSoft** Scripting Language module activation functions **LOGON()** and **LOGOFF()** or by selecting Logon under **Project** on the Main Menu Bar.



- **User Name:** Name of the user to be logged in.
- **Password:** User password.
- **Log Off:** Logs off the current user.

⇒ When a Logoff is executed, the Guest user is automatically logged on.

3.5.2 Graphics Tab




Graphics Tab

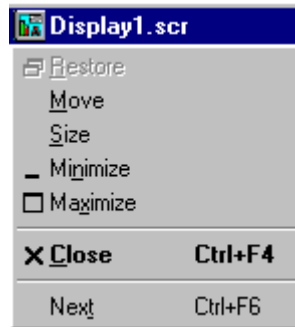
The **Graphics** tab has the following folders:

- **Screens:** This contains screens with finished graphic compilations as well as working drafts.
- **Group Screens:** This combines individual Display Screens from the Screens folder into more manageable groups. Available in NT only, it is not available in CE.
- **Library:** This is a library of symbols provided by **Studio**.
- **Symbols:** This is a collection of user-made symbols.

SCREENS

This accesses Display Screens containing finished graphic compilations or working drafts. When open, a screen is displayed to the right of the **Workspace** window.


To open an existing **Display Screen**, expand the **Screens** folder and double-click on the desired screen. Clicking the **Screen**  icon in the title-bar of the **Display Screen** opens a drop-down window with the options **Restore**, **Move**, **Size**, **Minimize**, **Maximize**, **Close**, and **Next**.



Display Screen Drop-down Window

Any object for data input in the **Display Screen** (such as input commands, sliders, or screens) has a Security Level Field.

Creating New Display Screens

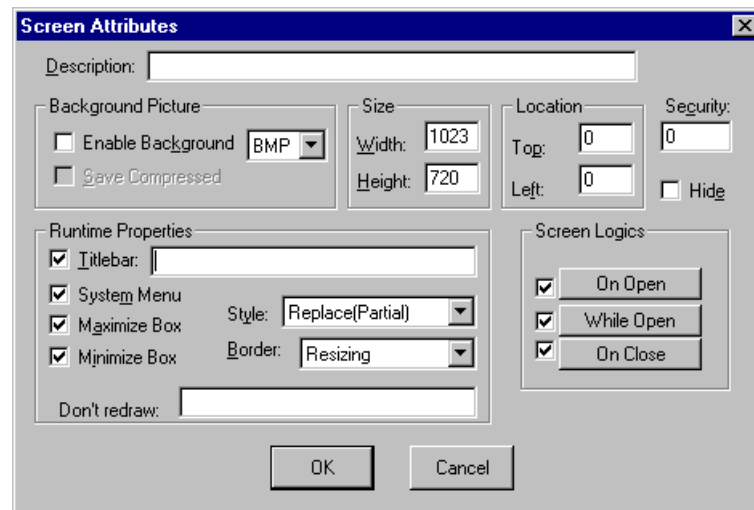
- ◆ Right-click on the **Screens** folder to insert a new Display Screen. Clicking the prompt opens a **Screen Attributes Window**. Or select **New** under **File** on the Main Menu Bar, click on the **New**  icon on the Standard Toolbar, or select the **Screen** option under **Insert** on the Main Menu Bar.

These methods open the **New Document** window.

- ◆ Select **Display** and then click on the **OK** button.

A **Screen Attributes** window appears.

SCREEN ATTRIBUTES WINDOW



Screen Attributes Window

- **Description:** This field is used for documentation. The text inserted in this field displays in the status bar (as the default value) at the lower left of the viewing screen during the **Run Application** mode.
- **Background Picture Group Box**
 - **Enable background Check-box** - Enables the use of background bitmaps.
 - **Enable Background Drop-List** - Allows you to enable or disable the background CE only bitmaps. The default value is disabled. In addition to BMP, background selections include TIF, DXF, EPS, WMF, IMG, JPG, WPG, PCD, PNG, FMF, FPX, FAX, and TGA.
 - **Save Compressed Check-box** - This option saves the **.BMP** file in a compressed form.

**CAUTION**

You cannot read the saved **.BMP** file in compressed format if the Windows *setup* values are modified or installed in an environment with a different number of colors. It is recommended that you save the screens in uncompressed format in case you want to switch among different configurations. In CE, Bitmaps must be 16-color.

- **Size Group Box:** Defines the window size with an integer number in the Width and/or Height boxes to define the pixel size of the selected window.
- **Location Group Box:** Defines the window location with an integer number in the **Top** and/or **Left** to define the number of pixels of the current window.

**CAUTION**

If you modify the window size (*Thin* or *Resizing* style) directly with the pointer device (mouse), and then click **Screen Attributes** from **View** on the Main Menu Bar, you will be prompted to update the current screen size and location.

- **Security Field:** Sets the window security level defined under **Security** on the **Database** tab. The default value is **0** (zero).
- **Hide Check-box:** Sets the screen to remain loaded in memory after being called for the first time. This enables fast loads when you open the screen. **Screen Logics** is executed normally. This feature causes a high use of GDI resources; during development you should monitor these resources by using the **InfoResources** function. The default value of this field is **disabled**.
- **Runtime Properties Group Box:** You can use this group box to define the window properties when running on Run Application.
 - **Titlebar Field** - Type the name that will appear on the title bar of the viewing screen during the Run Application mode. The check-box activates or deactivates the title bar.
 - **System Menu** - Enables the system menu.
 - **Minimize Check-box** - Activates or deactivates the **Minimize** button.
 - **Maximize Check-box** - Activates or deactivates the **Maximize** button.
 - **Style** - Defines the window style. The default window is the **Replace** style. The styles are:

| | |
|-------------------|--|
| Overlapped | The window is opened without closing any other. |
| Popup | The window is opened and remains in front of the others. The other windows are enabled. |
| Dialog | The window is opened and remains in front of the others. The other windows are disabled until the opened window is closed. |
| Replace | The window is opened, closing the Replace and Popup styles. |
- **Border** - Defines the window border. To select a border style, click on the desired option. The border default style is **Resizing**. The borders are:

| | |
|-----------------|--|
| None | No border. This does not allow a title bar or resizing. |
| Thin | Thin border window. This does not allow resizing in runtime. |
| Resizing | Normal border. It can be resized in runtime. |
- **Don't Redraw Field** - Receives a tg or value that controls refreshing the screen dynamics. When this value is higher than 0, all the screen dynamics are disabled.

- **Screen Logics Group Box** - The check-boxes and buttons allow you to execute mathematical functions in these events: **On Open**, **While Open**, **On Close**.

After you select an event, click on the corresponding button. This opens a window, allowing you to enter the following information:

| | |
|-------------------|---|
| Tag Name | Tag name to receive a return value from the Expression column. |
| Expression | Mathematical expression or function to be performed. The return value is applied to the Tag Name field. |
| Trigger | Only found on the While Open window. This holds a tag that works as a trigger (any value change) to execute this worksheet. When this field is left blank the worksheet is executed in the minimum time slice the system can perform. |

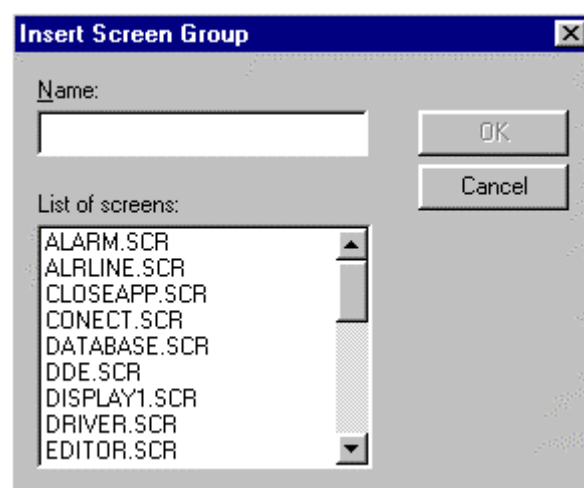
GROUP SCREEN

This combines individual Display Screens from the **Screen** folder into manageable Group Screens.

- ◆ To open a specific Screen Group, right-click on its sub-folder within the **Group Screen** folder.
- ◆ To remove a specific Screen Group, right-click on its sub-folder. Click the prompt to delete.

Creating New Screen Groups

Right-click on the Group Screen folder to insert a new Group Screen. Clicking the prompt opens an Insert Screen Group window, or selecting the Screen Group option under Insert on the Main Menu Bar.




Insert Screen Group Window

- **Name Field:** Assign a name for the folder that will contain the group of selected screens.
- **List of Screens:** Lists screens currently located in **Screen** folder. Select screens for a Screen Group by holding down the CTRL key while clicking on your selections.


Web Pages

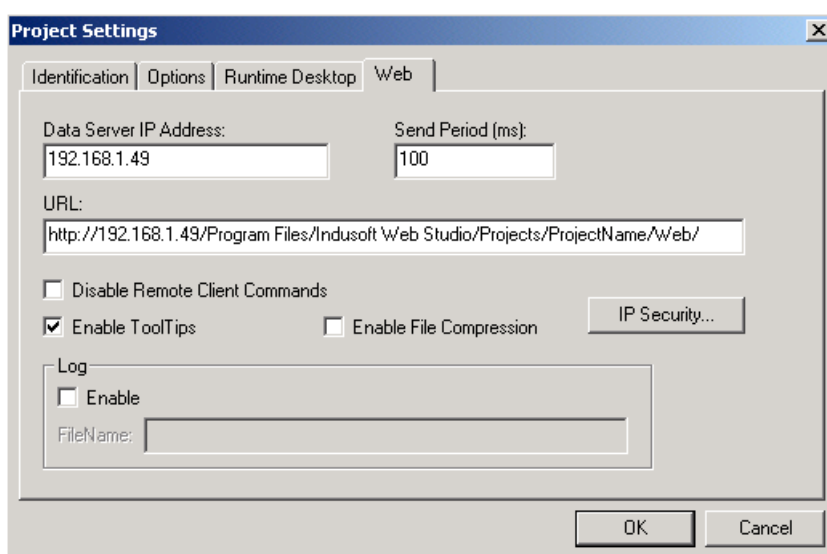
This is where the application screens are kept in HTML format. The HTML pages in this folder are not created directly; rather they are generated from preexisting display screens.

To create an HTML page you must first create a display screen. Configure the screen as usual, creating objects, adding properties, etc., but keep in mind that this screen will become a web page. Once the screen is complete save as usual. Finally, with the screen still open expand the **File Menu** from the **Menu Bar** and select  Save As **H**TML.



CAUTION

The Web Pages generated by the  Save As **H**TML function are independent of the screen file they were generated from. As such if you make a change to the Display Screen that change will not appear on the web page until you again  Save As **H**TML.



To be able to view your web pages you must first configure the web setting. These can be found in the **Project Settings** window under the **Web** tab.

- ◆ First you need to input the **Data Server IP Address**, this is the IP address where the application is running.
- ◆ Next you need to enter the **URL** in the following format: **http://<the IP address of the unit where the web server is running>/<path from the server to the web page directory>/**.
- ◆ Once these two fields are correct click the ok button.

- ◆ Then go to Tools on the menu bar and select Verify application (if you have any windows open in the development system Studio will demand you close them before verifying the application).



CAUTION

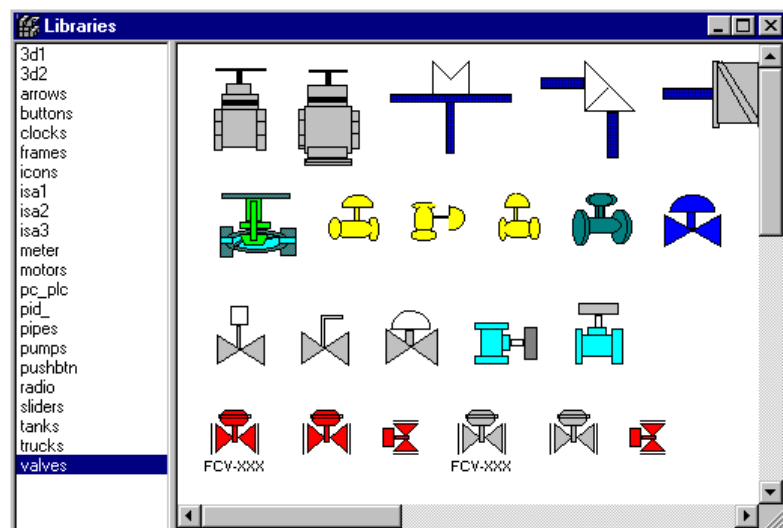
If you change any of the web information under the Project Settings you will need to Re-Verify the application for the new setting to take affect.

Because the Web Pages Display information from the application through the Web Server, the Runtime System, the Web Server and the TCP/IP Server need to be running to view the Web Pages.

LIBRARY

This is a library of symbols provided by **Studio**. The library is a set of common symbols grouped by meaning in one screen and stored in a specific directory.

- ◆ To open the **Studio Libraries** window, double-click **Library** on the **Graphics** tab or click the **Library** icon in the Standard Toolbar or under **View** on the Main Menu Bar.



Libraries Window

- ◆ Select a category from the left side of the screen to preview available images.
- ◆ To import a copy of an image to a Display Screen, double-click your selection. This keeps your image and closes the **Libraries** window. Click anywhere in the Display Screen to place the selected image.



CAUTION


Most of the symbols have predefined properties. To change the properties, use the Replace tab on the Object Properties window.



CAUTION

You can add a user screen to the Symbol library. Develop the screen as .scr and then copy it to the \LIB directory where **Studio** is installed.

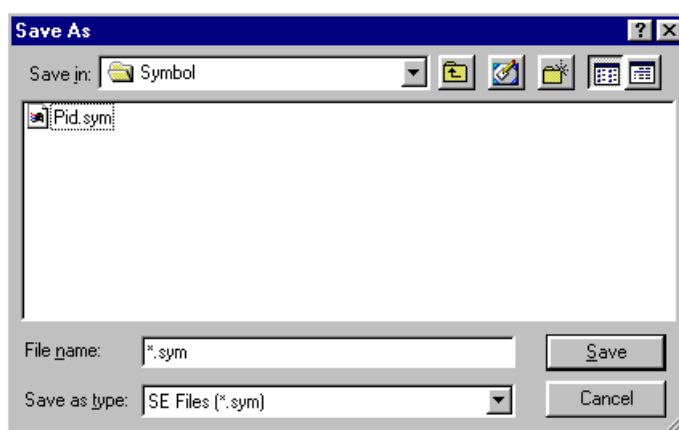
SYMBOLS

Symbols are groups of images and/or text. Symbols are created with the **Group**  icon on the Object Editing Toolbar.

You can create custom symbols in the Display Screen and save them into this folder.

- ◆ Select the symbol then select the **Copy to** option under **Edit** of the Main Menu Bar.

This will open a **Save As** window for saving into the **Symbol** folder.

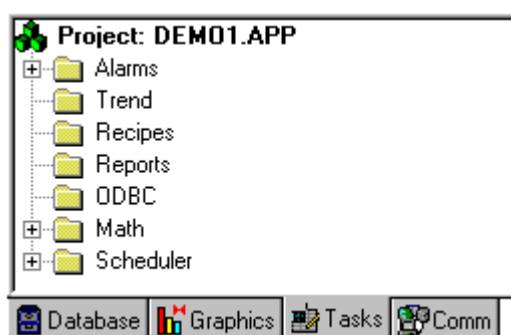


Save As Window

- ◆ To use a symbol from the **Symbol** folder, select the **Paste from** option under **Edit** of the Main Menu Bar.

This automatically imports a copy of a selected symbol to the active Display Screen.

3.5.3 Task Tab



Tasks Tab


The **Tasks** tab has the following folders:

- **Alarms:** Configuration of alarm groups and tags related to each group. The **Alarm** task defines the alarm messages that **Studio** will generate. Right-click on the folder to insert an alarm worksheet.
- **Trend:** Configuration of history groups that store the trend curves. The **Trend** task allows you to declare which tags must have their values stored on disk and creates history files for trend graphs. Right-click on the folder to insert a trend graph worksheet.

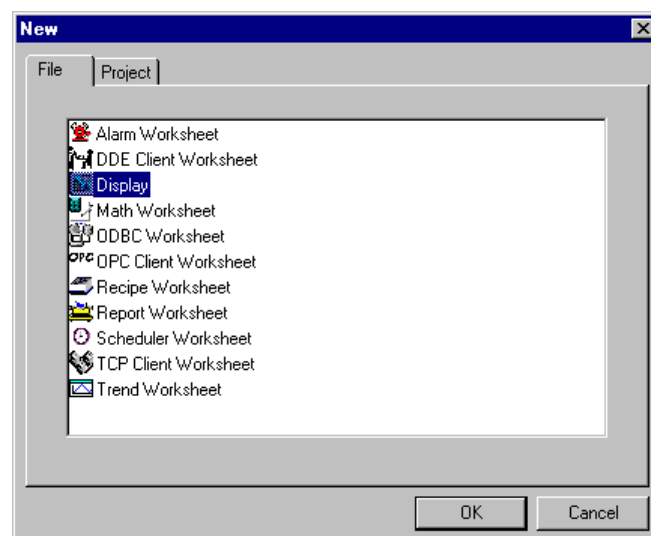
- **Recipes:** Configuration of recipe worksheets for data interchange between the application database and disk files in ASCII or DBF format. The **Recipe** task reads and writes tag values in files. This module transfers tag values from the application to a file or from a file to the application. Right-click on the folder to insert a recipe worksheet.
- **Reports:** Definition of reports (text type) by the user to be sent to the printer or disk. The **Report** task allows you to configure your own report (text type) with data from the system. Right-click on the folder to insert a report worksheet.
- **ODBC:** The **ODBC** interface runs in a network environment and also uses the Windows **ODBC** standard configuration. The ODBC task is capable of data interchange between **Studio** and any database that supports this interface. Right-click on the folder to insert an ODBC worksheet.
- **Math:** The **Math** task performs functions and calculations needed in the system. Worksheets use the mathematical functions and programming of the **InduSoft** Scripting Language. Right-click on the folder to insert a math worksheet.
- **Scheduler:** The **Scheduler** task generates events with definition of mathematical expressions to be executed according to the time, date or any monitored event. Right-click on the folder to insert a schedule worksheet.

TO ACCESS TASK WORKSHEETS

Task Worksheets are available for Alarm, Trend, Recipe, Report, Math, Scheduler and ODBC. A worksheet is displayed in the space to the right of the of the **Workspace** window.

- ◆ To open existing Task Worksheets, click on the **Tasks** tab, then expand the appropriate folder and double-click on the desired file.
- ◆ To create a new Task Worksheet, from the **File** menu select **New** or click on the **New**  icon in the Standard Toolbar.


This opens the **New** window with two tabs, select the **File** tab. Select the appropriate worksheet and click on the **OK** button.



New Window Displaying Worksheets Options


ALARMS

In the **Alarms** task, you define a group's characteristics and alarm messages that **Studio** reports in alarm conditions. The main purpose of alarms is to inform the operators about any problem or change of state during the process so that corrective action can be taken.

To show alarm messages on the screen, you must create an alarm object with the **Alarm**  icon on the Object Editing Toolbar. See also a description of alarms associated with tags in **Application Tags** on the **Database** tab.

⇒ **The number that identifies the Alarm Worksheet is sequentially incremented for each newly created worksheet.**

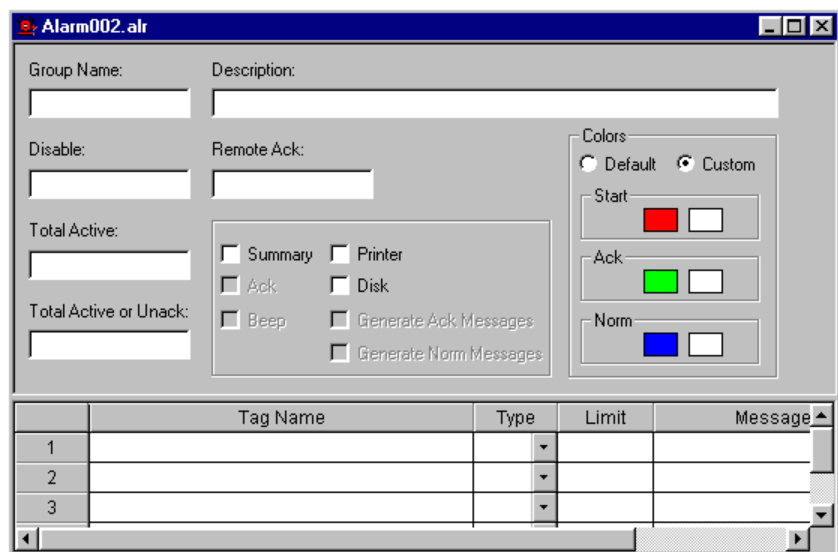
ALARM WORKSHEET

- ◆ Right-click on the **Alarm** folder to insert a new worksheet. Clicking the prompt opens an **Alarm Worksheet**. Or select **New** under **File** on the Main Menu Bar or click on the **New**  icon on the Standard Toolbar.

These two methods open the **New Document** window.

- ◆ Select **Alarm Worksheet** and then click on the **OK** button.

A new worksheet appears.



| | Tag Name | Type | Limit | Message |
|---|----------|------|-------|---------|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

Alarm Worksheet

The **Alarm** worksheet is divided into two parts:

- a *header* with information for the whole group,
- and a *body* where each tag of the group is defined.

ALARM WORKSHEET HEADER

This defines a group of common characteristics for all alarms of the group.

- **Group Name Field:** Name used to distinguish the alarm groups.



CAUTION

Before changing the Group Name field, save the alarm worksheet, because alarm settings in an unsaved worksheet can be lost.

- **Description Field:** Enter remarks here for documentation purposes.
- **Disable Field:** Disables all alarms in the group. You must fill this field with a tag. If the value of this tag is greater than zero, the group is disabled, and alarm messages are not generated. If the field is left blank, the group will be always enabled.
- **Remote Ack Field:** Tag for alarm acknowledgment. The acknowledgment occurs when there is a value change for this tag.
- **Total Active Field:** Holds the total number of active alarms in the group. The system always updates this value when one of the tags changes its alarm condition.
- **Total Active or Unack Field:** Holds the total number of active or unacknowledged alarms in the group. The system always updates this value when one of the tags changes its alarm condition.
- **Group Box**
 - **Summary Check-box** - When selected, sends alarm messages to an alarm object on the screen.



CAUTION

If you did not select the Summary option, the alarms of this group will not appear in the alarm objects in the screens and printer, during execution.

- **Ack Check-box** - Demands the acknowledgment of the alarm messages. Only available if the **Summary** field is enabled.
- **Beep Check-box** - Sounds the beep until the alarm is acknowledged. Only available if the **Ack** and **Summary** fields are enabled.
- **Printer Check-box** - Sends the each alarm messages of this group to the printer. This option can only be used with a dot matrix printer (or any other which prints line by line).
- **Disk Check-box** - Sends the alarm messages of this group to a file on the hard disk. You must select this option if you want to have history alarm objects.
- **Generate Ack Messages Check-box** - Generates messages whenever the alarms of this group are acknowledged. Only available if the **Disk** or **Printer** fields are enabled.
- **Generate Norm Message Check-box** - Generates messages whenever the alarms of this group return to their normal state. Only available if the **Disk** or **Printer** fields are enabled.

- **Colors Group Box:** Defines the colors of the alarm summaries to the alarm object. Each alarm message will be shown in the alarm object in the colors defined for its group.
 - **Default Radio Button**
 - **Custom Radio Button**
 - **Start Color Rectangle**
 - **Ack Color Rectangle**
 - **Norm Color Rectangle**
- ◆ Click on a Color rectangle to display a **Color Selection** window.
- ◆ Double-click on the desired color or click the color and then the **OK** button.



Color Selection Window

ALARM WORKSHEET BODY

The body of the Alarm worksheet defines the tags in this group, their alarm condition, and messages. It has six columns (only four are shown in the sample Alarm Worksheet).

- **Tag Name Field:** Defines the colors of the alarm summaries to the alarm object. Each alarm message will be shown in the alarm object in the colors defined for its group.
- **Type Drop-list:** Type of alarm: HiHi, Hi, Lo, LoLo, Rate, Dev +, Dev-. You can change any of these fields in the runtime module. For additional information see Application Tags.
 - **HiHi** - The too high alarm limit, generating an alarm message when the tag value is equal to or greater than the HiHi Limit value.
 - **Hi** - High limit, generating an alarm when the tag value is equal to or greater than the Hi Limit value.
 - **Lo** - Low limit, generating an alarm when the tag value is lower than or equal to the Lo Limit value.
 - **LoLo** - Too low limit, generating an alarm when the tag value is lower than or equal to the LoLo Limit value.
 - **Rate** - Determines the speed of the variation rate for a tag. If the variation speed is higher than the established one in this field, it generates an alarm. The speed can be determined per second, minute, or hour.
 - **Deviation +** - Deviation for a higher value, generating an alarm when an augmentation in the tag value equal to or higher than the established limit occurs.
 - **Deviation -** - Deviation for a lower value, generating an alarm when a diminution in the tag value equal to or higher than the established limit occurs.

- **Limit Field:** Value limit for the alarm generation.
- **Message Field:** Alarm message to be displayed.

**CAUTION**

The alarm messages can hold any system tag through the syntax: *message{tag_name}*.

- **Priority Field:** Indicates the priority within a group. This field can be filled with an integer number (0 to 255). The tag with a higher priority must have a higher **Priority** value.
- **Selection Field:** A user-defined string that works as a filter in the alarm summary objects.

**CAUTION**

The Selection field must have a string with a maximum of 7 characters (the other characters will not be considered).

ALARM HISTORY FILE

The alarm history file, when enabled in the group, is saved in the following format:

| Alarm Summary | | | | | | | | | | | | |
|---------------|------------|----------|---------|-----|-----|-------|-----|-----|------|------|-------|------|
| <1> | <2> | <3> | <4> | <5> | <6> | <7> | <8> | <9> | <10> | <11> | <...> | <15> |
| 000 | 16/06/1997 | 16:30:23 | simu[1] | 0 | 1 | 90.00 | 1 | 0 | 1 | 0 | | |
| (more lines) | | | | | | | | | | | | |

Where:

| | | |
|------|---|---|
| <1> | = | reserved (internal number of the document version, the actual is 000) |
| <2> | = | start date of the alarm* |
| <3> | = | start time of the alarm HH:MM:SS |
| <4> | = | tag name |
| <5> | = | 1: alarm is waiting ack, 0: other case |
| <6> | = | 1: alarm is active, 0: other case |
| <7> | = | tag value |
| <8> | = | number of the alarm group |
| <8> | = | alarm priority |
| <9> | = | selection field text |
| <10> | = | alarm type |
| <11> | = | 1: alarm type request ack, 0: other case |
| <12> | = | ending date of the alarm* |
| <13> | = | ending time of the alarm HH:MM:SS |
| <14> | = | acknowledge date of the alarm* |
| <15> | = | acknowledge hour of the alarm* HH:MM:SS |


This file is saved in the **ALARM** directory of the application with the following name:

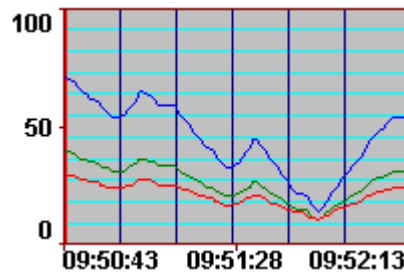
lapp\ALARM\ALyymmdd.ALH, where yymmdd refer to the year, month, and day the alarm file was created.

⇒ **This format varies according to the international date format.**

TREND

The **Trend** task keeps track of process variables behavior. You can store the samples in a history file and show both history and online samples in a screen trend graph.

To show a trend graph on the screen, you must create a trend object with the **Trend**  icon on the Object Editing Toolbar.




Sample Trend Graph Showing Three Histories

⇒ **The number that identifies the Trend Worksheet is sequentially incremented for each newly created worksheet. Four bytes save date and time information; eight bytes are used per variable in each sampling.**

For additional information, see **Converting Trend History Files**.

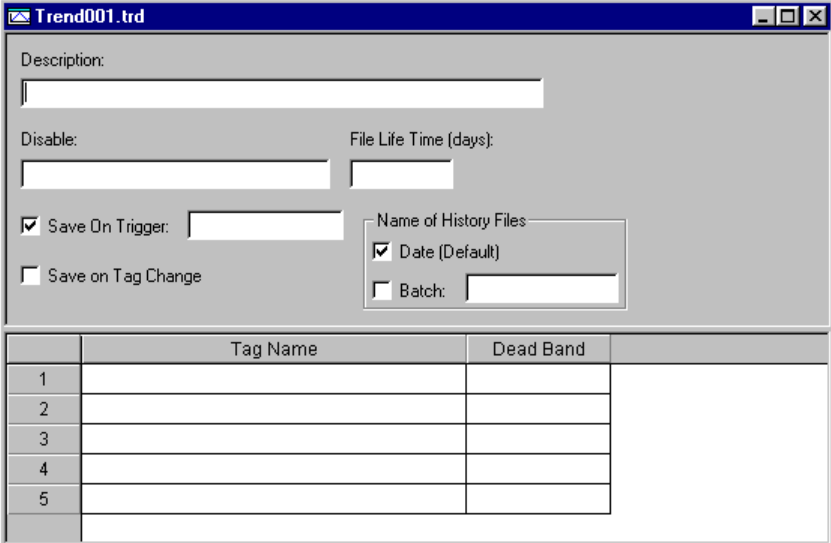
TREND WORKSHEET

- ◆ Right-click on the **Trend** folder to insert a new worksheet. Clicking the prompt opens a **Trend Worksheet**. Or select **New** under **File** on the Main Menu Bar or click on the **New**  icon on the Standard Toolbar.

These two methods open the **New Document** window.

- ◆ Select **Trend Worksheet** and then click on the **OK** button.

A new worksheet appears.



| | Tag Name | Dead Band | |
|---|----------|-----------|--|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

Trend Worksheet

The **Trend** worksheet is divided into two parts:

- a *header* with information for the whole group,
- and a *body* where each tag of the group is defined.

TREND WORKSHEET HEADER

- **Description Field:** You can fill this field with a tag to provide a temporary disable function when the tag value is greater than **0**.
- **Disable Field:** You can fill this field with a tag to provide a temporary disable function when the tag value is greater than **0**.
- **File Life Time (days) Field:** Determines how many days the history file will be kept on the disk. After the determined period, the file is automatically erased. This option is used only for files based on a date.
- **Save on Trigger Check-box/Field:** Always saves the trend samples when a change in the specified tag occurs. This tag change can be an event from Scheduler.
- **Save on Tag Change Field:** Always saves the trend sample when a value change occurs in any of the tags from that group.
- **Name of History Files Group Box:** Defines the history file name. Trend historical files can be generated in two forms: by date or batch (by events).

- **Date (Default) Check-box:** When selected, history files based in the date are generated. Use this option if you have a continuous process. In this case, the generated files are: `\app\HSTgggyyyymmdd.HST` where:
 - app = application directory
 - gg = historical group number (hexadecimal)
 - yyyy = year
 - mm = month
 - dd = day
- **Batch Check-box/Field:** When selected, creates history files, using the name indicated in the edition field. This field can have tag values. Use this option if you have a batch process.
Example: `c:\history\file{TagBatchNumber}.hst`

⇒ **To view online curves, the files based on date must be enabled. For historical curves, you can use the two kinds of files.**

TREND WORKSHEET BODY

- **Tag Name Field:** Tag to be saved in the history file.



CAUTION

Each Trend group can hold a maximum of 100 tags. It is recommended that you divide it into more groups if necessary.

- **Dead Band Field:** Value to filter acceptable changes when **Save on Tag Change** is used.

Example:

Dead Band has value = 5.

If the tag value is **50** and changes to **52**, the system will not register this variation in the database, because it is less than **5**.


If the change is equal to or greater than **5**, the new value will be shown in the trend graphic.

RECIPES

The **Recipes** task reads and writes files from and to the hard disk; it transfers values between files and real-time memory. Its typical use is to store process recipes, but these files can store any type of information such as operation logs, passwords, and so forth.

⇒ **The number that identifies the Recipe Worksheet is sequentially incremented for each newly created worksheet.**

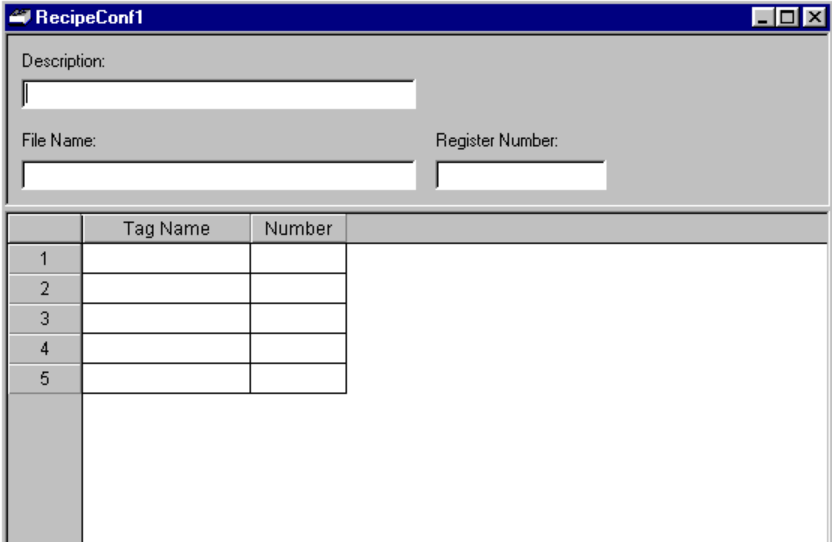
RECIPES WORKSHEET

- ◆ Right-click on the **Recipes** folder to insert a new worksheet. Clicking the prompt opens a **Recipe Worksheet**. Or select **New** under **File** on the Main Menu Bar or click on the **New**  icon on the Standard Toolbar.

These two methods open the **New Document** window.

- ◆ Select **Recipe Worksheet** and then click on the **OK** button.

A new worksheet appears.



The screenshot shows a window titled "RecipeConf1" with the following fields and table:

Description:

File Name: Register Number:

| | Tag Name | Number |
|---|----------------------|----------------------|
| 1 | <input type="text"/> | <input type="text"/> |
| 2 | <input type="text"/> | <input type="text"/> |
| 3 | <input type="text"/> | <input type="text"/> |
| 4 | <input type="text"/> | <input type="text"/> |
| 5 | <input type="text"/> | <input type="text"/> |

Recipe Worksheet

The **Recipe** worksheet is divided into two parts:

- a *header* with information for the whole group,
- and a *body* where each tag of the group is defined.

RECIPE WORKSHEET HEADER

- **Description Field:** Enter remarks for documentation purposes.
- **Save As XML:** If checked, indicates that the information will be saved in the XML format. If unchecked, information will be saved in standard DAT format.



CAUTION

While information in a .DAT file can be loaded into different tags using a second Recipe worksheet; information in a .XML file can only be loaded into tags with the same name as the tag the data originated from.

Like the HTML pages, the Web Server needs to be running in order to view the XML data from the web. Unlike the HTML Pages the Runtime System does not need to be running to view the XML data. (only Internet Explorer versions 5.0 and above can view XML data).

-
- **File Name Field:** Name of the file related to the recipe group. The file name can be static text (e.g. File1) or a dynamic tag value (e.g. {File-NameTag}).
 - **Register Number Field:** Tag that defines the register number to be read or written in a DBF file.

⇒ When you save your worksheet, you will be allowed to name it freely (it doesn't have a pre-defined file name). The configuration file with the default extension .RCP (.XSL if the Save As XML is selected) contains the recipe configuration, the File Name field has the data file name, which will be read or written.

RECIPE WORKSHEET BODY

- **Tag Name Field:** Tags to be updated with file contents or tags whose values will be written to a file. If the tag is an array, you must set the first position to be used.
- **Number Field:** Tag that defines the register number to be read or written in a DBF file.



CAUTION

When an array tag is defined, its initial position is 0 (zero), although it is used by the system in case of invalid position configuration. Avoid the use of the 0 (zero) position.

⇒ To read or write a recipe group, an *InduSoft* Scripting Language function is used.


REPORTS

The **Reports** task configures reports using system data. The main purpose of this module is to make report creation easier and more efficient.

⇒ **The number that identifies the Report Worksheet is sequentially incremented for each newly created worksheet.**

- ◆ To print a report, use an *InduSoft* Scripting Language function anywhere an expression is allowed.

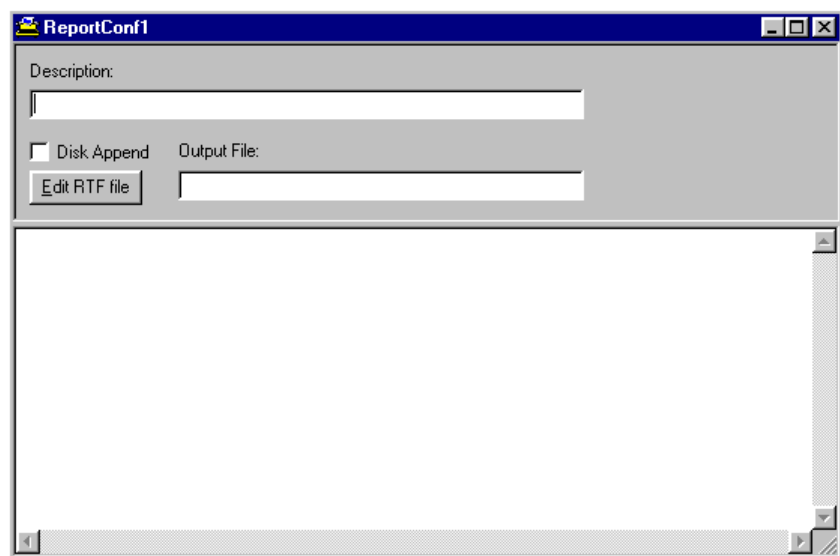
REPORT WORKSHEET

- ◆ Right-click on the **Reports** folder to insert a new worksheet. Clicking the prompt opens a **Report Worksheet**. Or select **New** under **File** on the Main Menu Bar or click on the **New**  icon on the Standard Toolbar.

These two methods open the **New Document** window.

- ◆ Select **Report Worksheet** and then click on the **OK** button.

A new worksheet appears.



Report Worksheet

The **Report** worksheet is divided into two parts:

- a *header* with information for the whole group,
- and a *body* where each tag of the group is defined.

REPORT WORKSHEET HEADER

- **Description Field:** Holds remarks for documentation purposes.
- **Disk Append Check-box:** When printing to file, this option adds (amends) the new report to the end of the existing file. If not selected, the new report will replace the previous report in that file.
- **Output File Field:** When printing to file, this is the name of the output file. The name of the output file follows the syntax *{tag}*, where *tag* value is part of the file name.

Example: report{day}.out.

In the previous example, the generated file could be report1.out, report2.out... and so on, according to the tag day value.

⇒ **The configuration file of a report has the default extension .REP. The Output File field is the file where data is stored.**

- **Edit RTF file Button:** Allows you to access the report as an RTF file for editing, such as layout modification, etc.

REPORT WORKSHEET BODY

This area is reserved for report formatting. You can configure your own report with the data in the system, indicating where the tag values are to be printed.

The name of each tag will replace the tag name: {tag_name}. If the tag is the **Real** type, use the following syntax: {tag_name *n*} where *n* is the number of decimal characters you want printed.

ODBC


The **ODBC** task is capable of data interchange between the Indusoft application and any database that supports this interface. The **ODBC** interface runs in a network environment and also uses the Windows **ODBC** standard configuration.

INSTALLATION

Besides **Studio ODBC** worksheet, you also need to configure the Windows **ODBC** standard driver. **Studio** refers to the **User DNS**, whose configuration is done through the **Control Panel**. For more information, refer to your Windows documentation.

⇒ **The number that identifies the ODBC Worksheet is sequentially incremented for each newly created worksheet.**

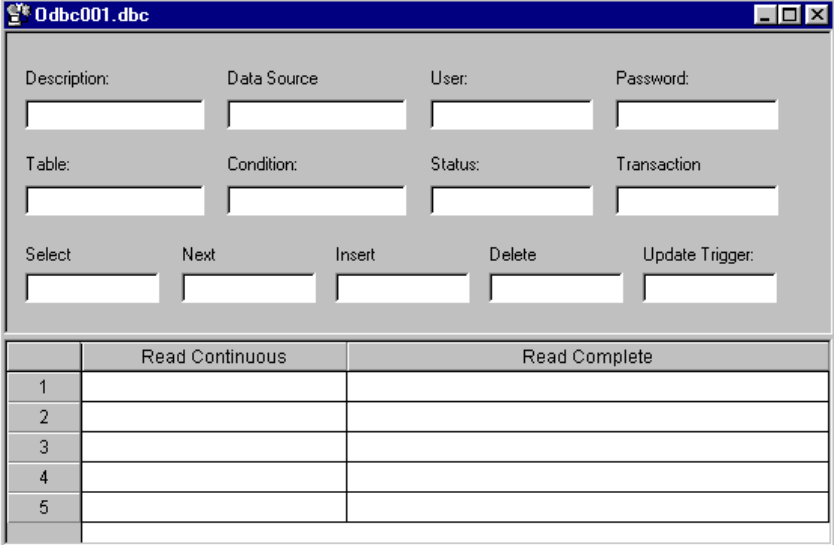
ODBC WORKSHEET

- ◆ Right-click on the **ODBC** folder to insert a new worksheet. Clicking the prompt opens an **ODBC Worksheet**. Or select **New** under **File** on the Main Menu Bar or click on the **New**  icon on the Standard Toolbar.

These two methods open the **New Document** window.

- ◆ Select **OBDC Worksheet** and then click on the **OK** button.

A new worksheet appears.



| | Read Continuous | Read Complete |
|---|-----------------|---------------|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |

ODBC Worksheet

The configuration worksheet is divided into two parts:

- a *header* with information for the whole group,
- and a *body* containing the tags and references to the values to be read or written in the database.

ODBC WORKSHEET HEADER

The header of the **ODBC Worksheet** lets you define tags to start read and write events, set return values, handle database access parameters, and so forth.

- **Description Field:** Enter remarks for documentation purposes.
- **Data Source Name Field:** The same **Data Source Name** configured in the **Windows Control Panel** that contains information on a specific database access.
- **User Field:** User name that has access to the database.
- **Password Field:** User password.
- **Table Field:** Table name in the database.
- **Condition Field:** Search condition or filter.
- **Status Field:** Return value (fill in with a tag name). The tag will report:
 - 0 - Success
 - Another value - Error code

- **Transaction Field:** Fill with a tag that will have a value change when the transaction is executed.
- **Select, Next, Insert, Delete, or Update Trigger fields:** Fill with a tag that works as a trigger. Each value change makes the system execute the command. At least one of the trigger fields is required.

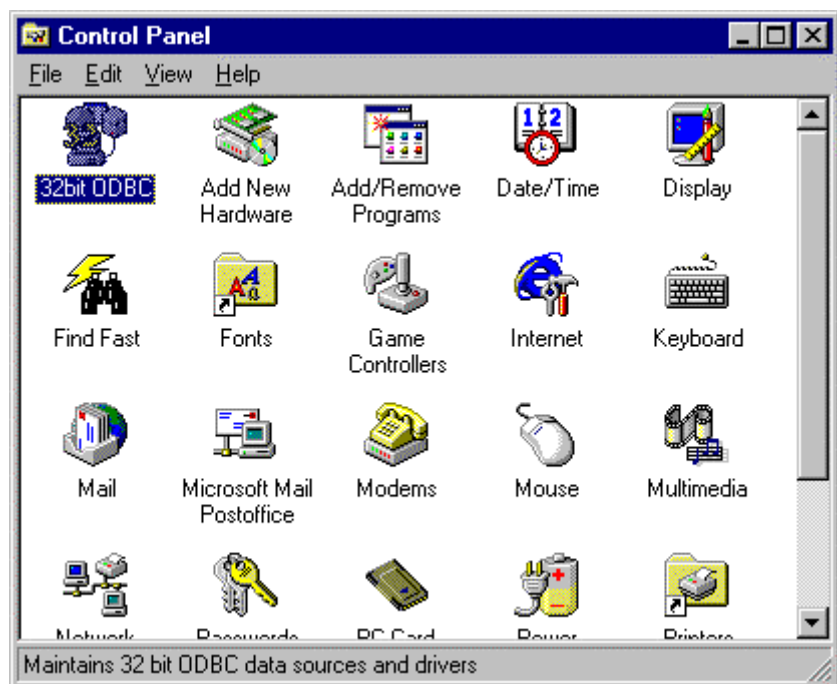
ODBC WORKSHEET BODY

In the **ODBC** Worksheet body, you relate tags to fields in the current register from the database table.

- **Tag Name :** Tags to be updated with file contents or tags whose values will be written to a file.
- **Column:** Holds the location where the data is to be found in the file. (e.g. R3CH corresponding to Row 3, Column H of an excel sheet).

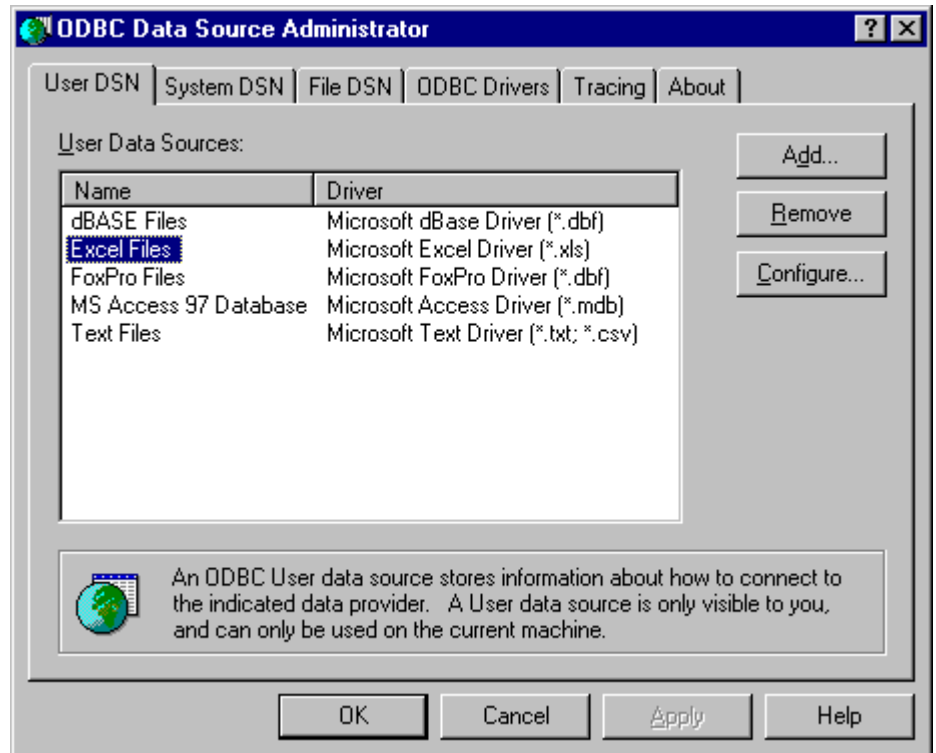
Setting up the ODBC Interface to Excel files WINDOWS CONTROL PANEL CONFIGURATION

- ◆ Access the Windows **Control Panel** from the **Start** button in the lower left of your screen.



Windows Control Panel

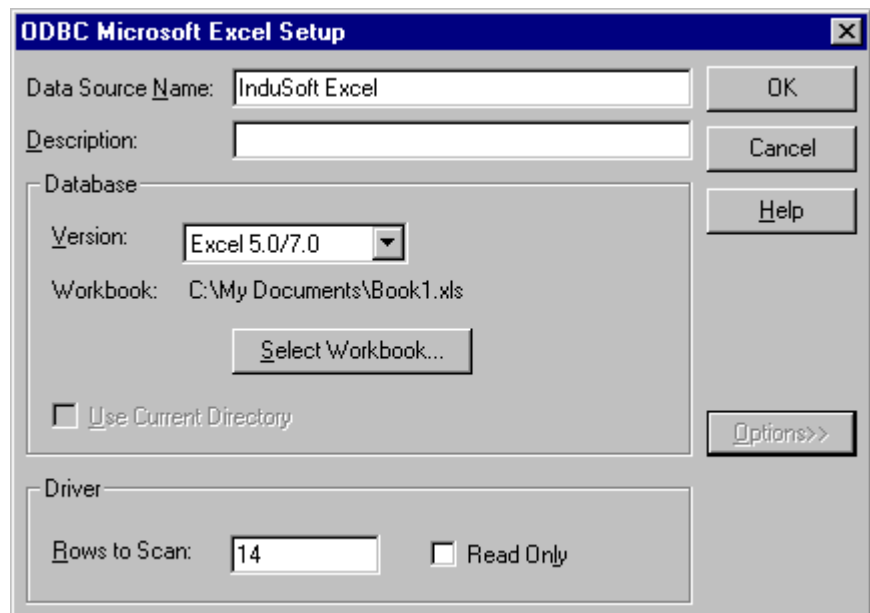
- ◆ Double-click on the **ODBC** icon in the **Windows Control Panel**, and then select *Excel Files*.



ODBC Data Source Administrator

- ◆ Click on the **Configure** button.

The ODBC Microsoft Excel Setup window is displayed.

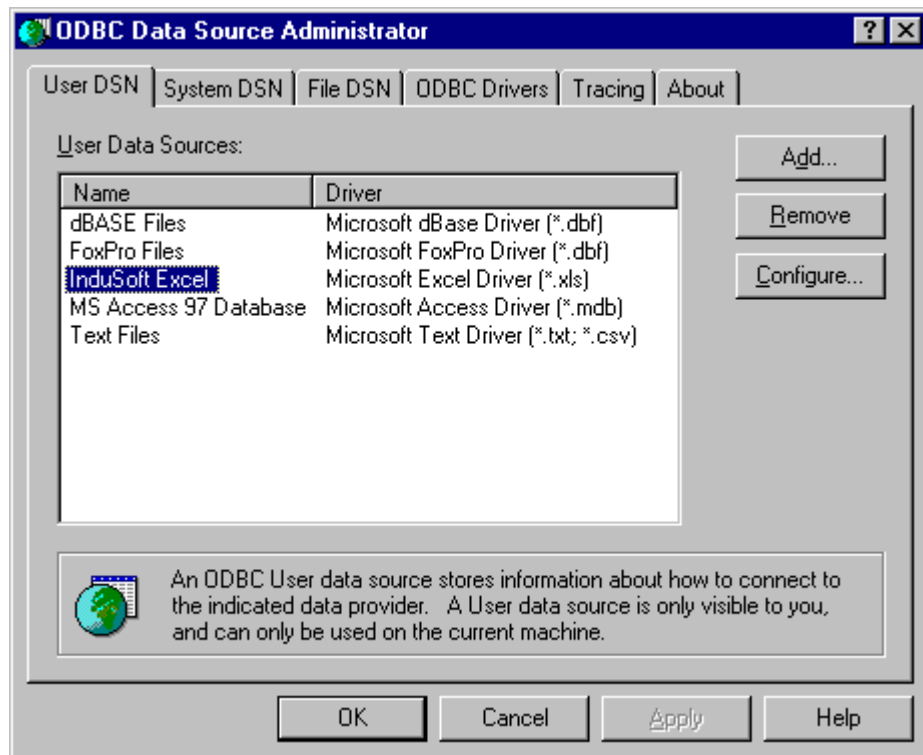


ODBC Microsoft Excel Setup

- ◆ In the **Data Source Name** field, enter the Windows configuration name to be used in the **ODBC** worksheet in the **DSN** field.
- ◆ Click on the **Select Workbook** button to configure the Excel file you will use.

- ◆ Return to the **ODBC Data Source Administrator** window.

Your **User DSN** displays in the list.



Updated List of User Data Sources

Studio ODBC WORKSHEETS

After you configure the ODBC Windows interface, you must configure the *Studio* ODBC worksheets.

- ◆ Through the Tasks tab, insert a new ODBC worksheet.

Be sure that the ODBC Runtime is set to startup on the Runtime Tasks tab in the Project Status option under Project on the Main Menu Bar. All you need to start this configuration is to run the project. Your application will be handling the Select, Next, Insert, Delete, and Update triggers to allow data exchange throughout rows in Excel and tags configured in the worksheet.

Error Codes

For the meaning of specific error codes, refer to your Windows documentation.

Select command:

- 1 Error in the ODBCPREPARE function.
- 2 Error in the ODBCINDCOL function.
- 3 Error in the ODBCEXECUTE function.
- 4 Error in the ODBCSETCH function.

Next command:

- 5 Error in the ODBCSETCH function.

Insert command:

- 6 Error in the ODBCPREPARE function.
- 7 Error in the ODBCEXECUTE function.
- 8 Error in the ODBCCommite function.

Update command:

- 9 Error in the ODBCPREPARE function.
- 10 Error in the ODBCEXECUTE function.
- 11 Error in the ODBCCommite function.

Delete command:

- 12 Error in the ODBCPREPARE function.
- 13 Error in the ODBCEXECUTE function.
- 14 Error in the ODBCCommite function.


MATH

The Math task allows you to implement additional routines to work with the basic functions of the **Studio** tasks.

A **Math Worksheet** is a group of programming lines that are executed as one of the Background Tasks during Runtime. You can configure the math worksheet to provide free environments for logical routines and mathematical calculations that the project may need. For these purposes, the **InduSoft** Scripting Language is very simple and easy to use.

⇒ **The number that identifies the Math Worksheet is sequentially incremented for each newly created worksheet.**

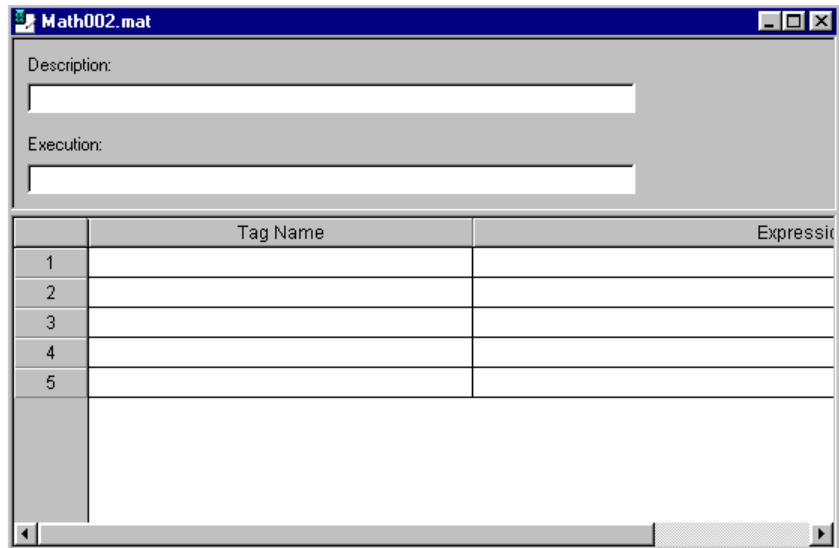
MATH WORKSHEET

- ◆ Right-click on the **Math** folder to insert a new worksheet. Clicking the prompt opens a **Math Worksheet**. Or select **New** under **File** on the Main Menu Bar or click on the **New**  icon on the Standard Toolbar.

These two methods open the **New Document** window.

- ◆ Select **Math Worksheet** and then click on the **OK** button.

A new worksheet appears.



| | Tag Name | Expression |
|---|----------|------------|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |

Math Worksheet

The **Math** worksheet is divided into two parts:

- a *header* with information for the whole group,
- and a *body* where each tag of the group is defined.

MATH WORKSHEET HEADER

- **Description Field:** Enter remarks for documentation purposes.
- **Execution Field:** Determines, with an expression, single tag value or constant value, when the worksheet should execute.



CAUTION

The worksheet is executed only when the result in the **Execution** field is not 0 (zero). If you want the worksheet to always execute, enter the value 1 (constant value).

MATH WORKSHEET BODY

The Mathsheet body defines the *Programming Lines* (logical routines and mathematical calculations through *functions* and *logical operations*).


- **Tag Name Field:** Tag that receives the return value of the configured calculation in the Expression column.
- **Expression Field:** Configuration whose return value is returned to the configured tag in the Tag Name column.

SCHEDULER

The **Scheduler** task generates time bases used in the application.

⇒ **The number that identifies the Scheduler Worksheet is sequentially incremented for each newly created worksheet. Different scheduler groups have only organizational purposes.**

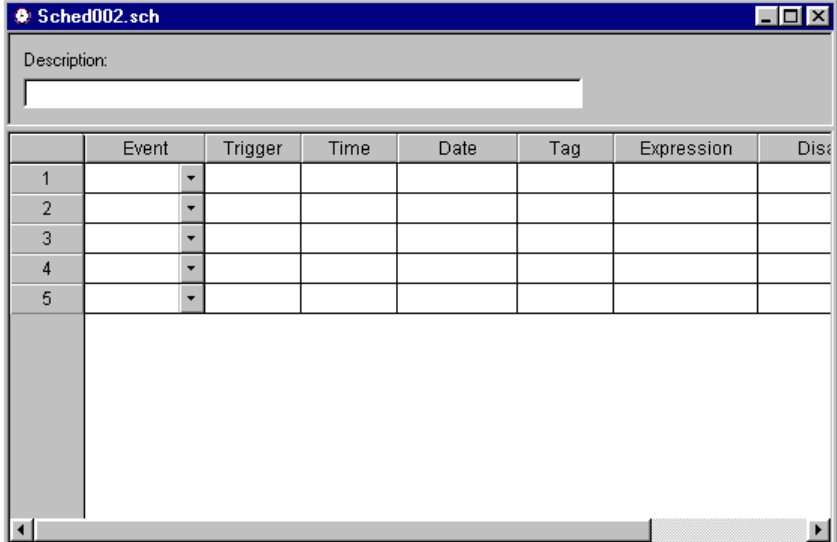
SCHEDULER WORKSHEET

- ◆ Right-click on the **Scheduler** folder to insert a new worksheet. Clicking the prompt opens a **Scheduler Worksheet**. Or select **New** under **File** on the Main Menu Bar or click on the **New**  icon on the Standard Toolbar.

These two methods open the **New Document** window.

- ◆ Select **Scheduler Worksheet** and then click on the **OK** button.

A new worksheet appears.



| | Event | Trigger | Time | Date | Tag | Expression | Dis |
|---|-------|---------|------|------|-----|------------|-----|
| 1 | ▼ | | | | | | |
| 2 | ▼ | | | | | | |
| 3 | ▼ | | | | | | |
| 4 | ▼ | | | | | | |
| 5 | ▼ | | | | | | |

Scheduler Worksheet

The **Scheduler** worksheet is divided into two parts:

- a *header* with information for the whole group,
- and a *body* where each tag of the group is defined.

SCHEDULER WORKSHEET HEADER

- **Description Field:** Enter remarks for documentation purposes.

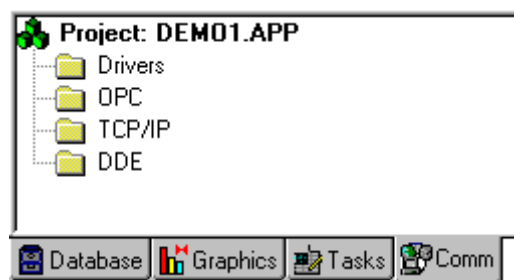
SCHEDULER WORKSHEET BODY

- **Event Drop-list:** Selects the type of the event (calendar, clock, change).
 - **Calendar** - Event that generates time bases greater than 24 hours.
Example: You can define an event that prints a report each Friday.

⇒ **Fill the Date field when you want a specific date for the event execution.**

- **Clock** - Event used to generate time bases smaller than 24 hours (intervals can be minutes or seconds). This function is frequently used with trend graphics. *Example:* Define a tag that is incremented each hour.
- **Change** - Event related to the change of a tag in the **Trigger** field.
- **Trigger Field:** Starts a **Change** event related to a tag value change. When a value change of the **Trigger** tag occurs, the value in the **Value** field is returned to the **Tag**. This field is used only by the **Change** event.
- **Time Field:** Sets the time interval in which the event must occur: hours (0 to 23), minutes (0 to 59), and seconds (0 to 59), when used by **Clock**. This also sets a specific time when used by **Calendar** events.
- **Date Field:** Sets the specific date when a Calendar event must occur: day (1 to 31), month (1 to 12), and year (1900 to 2099). If the field is blank, the event occurs daily. This field is only used by **Calendar** events.
- **Tag Field:** Tag that receives a new value or expression return in the event.
- **Expression Field:** Expression whose return value will be set to the tag. This field is used by all events.
- **Disable Field:** Holds a disable condition for the function. When it is left blank or the expression value is equal to zero, the function will be executed. If the expression value is = 1, the function will not execute (Disable = 1).

3.5.4 Communication Tab



The Communications Tab

The **Communications** tab has the following folders:

- **Drivers:** Allows you to define the communication interface (or interfaces) with remote equipments the project will handle.
- **OPC:** Allows you to configure OPC interfaces to an application through an OPC Server.
- **TCP/IP:** Allows you to configure TCP/IP Client interfaces to other *InduSoft* stations.
- **DDE:** Allows you to configure a DDE Client configuration to a DDE Server application like Excel and any other Windows program that supports this interface.

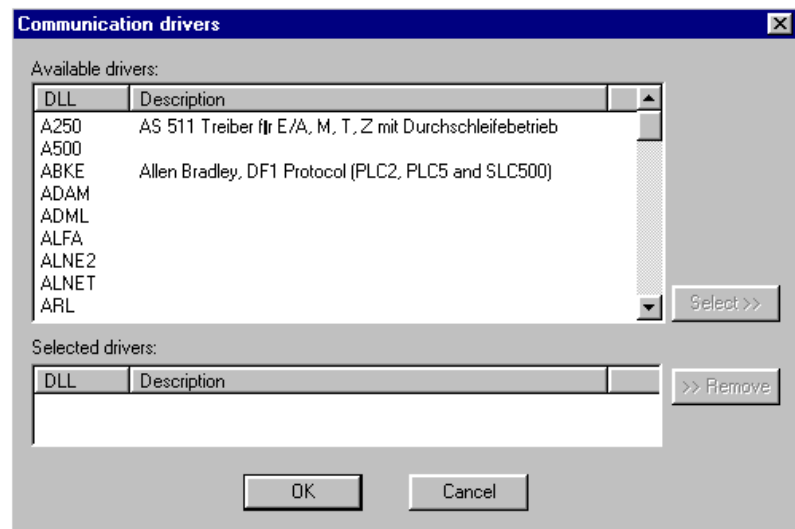
DRIVERS

Driver Configuration defines the communication interface with remote equipment such as PLC, single-loop, and transmitters. This help describes the functions and characteristics that are standard for all drivers. When developing an application, you should also refer to the specific documentation provided with each communication driver.

This documentation is usually located on the **DRV** directory. To configure a communication driver, first specify the interface parameters (for example, the station address and the baud rate). Then, specify the addresses in the equipment that is connecting to *InduSoft* tags.

- ◆ Right-click on the **Drivers** folder to add or remove a configured driver. Or select the **Drivers** option under **Insert** on the Main Menu Bar.

Both of these open a **Communication Drivers** window that displays a list of available drivers.



Communication Drivers Window

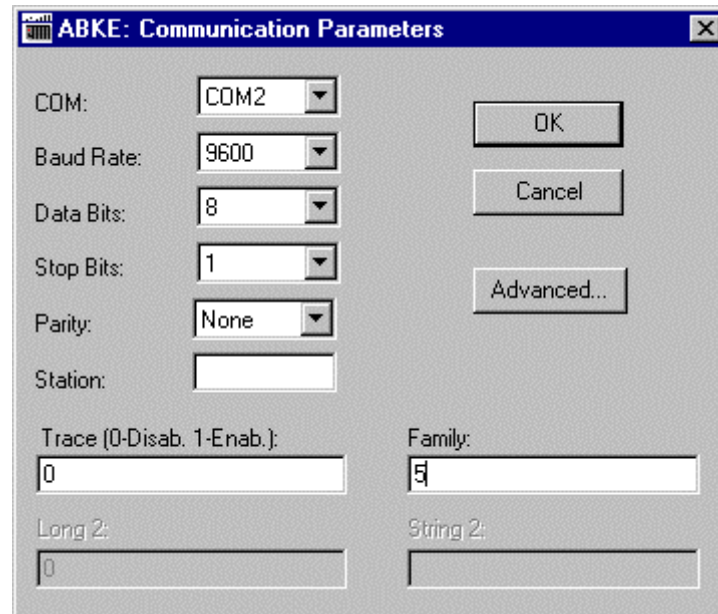
- **Available Drivers Field:** Lists names of drivers and a brief description.
- **Select Button:** To select a driver, highlight it from the **Available Drivers** field, then click on the **Select** button.
- **Selected Drivers Field:** Selected drivers appear in a list, along with description, if available.
- **Remove Button:** To remove a driver, highlight it from the **Selected Drivers** field, then click on the **Remove** button.

SETTINGS/COMMUNICATION PARAMETERS

Clicking on the **OK** button of the **Communications Driver** window creates a sub-folder for the selected driver in the **Drivers** folder on the **Communications** tab.

- ◆ Right-click on the selected driver sub-folder to access the **Settings** option.

This opens the **Communications Parameters** window.

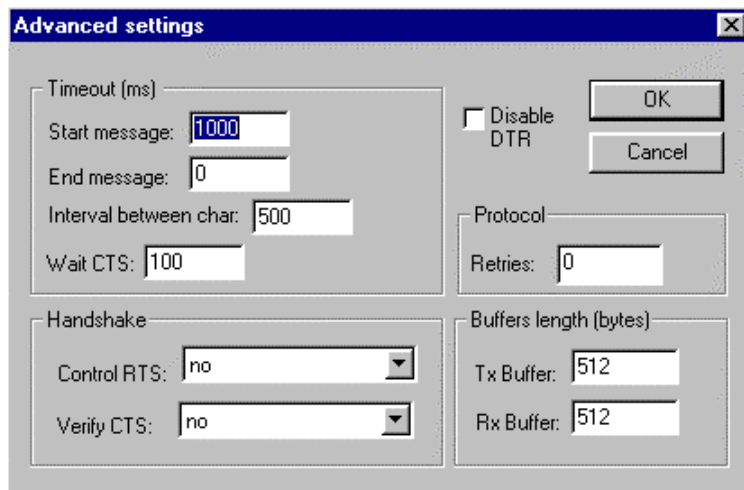


Communication Parameters Window

- **COM Field:** Serial communication port.
- **Baud Rate, Data Bits, Stop Bits, Parity Fields:** Serial port configuration.
- **Station Field:** Network station address.
- **Long1, Long2, String1, and String2 Fields:** These fields are automatically populated with information derived from your previous communication driver selection.
- **Advanced Button:** Opens the **Advanced Settings** window, in which you can change the default driver parameters.

ADVANCED SETTINGS WINDOW

Clicking on the **Advanced Settings** button of the **Communications Parameters** window opens the **Advanced Settings** window.



Advanced Settings Window

- **Timeout (ms) Group Box**
 - **Start Message Field** - Specifies the timeout for the message start.
 - **End Message Field** - Specifies the timeout for the message end.
 - **Interval between char Field**- This specifies the timeout between each character.
 - **Wait CTS Field** - Specifies the timeout for the Clear to Send wait.
- **Handshake Group Box**
 - **Control RTS Drop-list** - Select whether or not to use the "Request to Send" control.
 - **Verify CTS Drop-list** - Select whether or not to use the "Clear to Send" type of verification.
- **Disable DTR Check-box**: Allows you to disable the DTR function. If checked, the driver will not set the DTR signal before starting the communication.
- **Protocol Group Box**
 - **Retries Field** - Specifies the number of new communication attempts.
- **Buffers length (bytes) Group Box**
 - **Tx Buffer Field** - Specifies the transmission buffer length (in bytes).
 - **Rx Buffer Field** - Specifies the reception buffer length (in bytes).

DEVELOPING A COMMUNICATION DRIVER

A communication driver is a DLL that contains specific information about the remote equipment and implements the communication protocol. To develop a new communication driver, there is a driver toolkit available. Consult **InduSoft** for further information.

DRIVER WORKSHEET

- ◆ Right-click on the selected driver sub-folder to access the Insert option.

This opens the Driver Worksheet window.

| | Tag Name | Address | Div | Add |
|---|----------|---------|-----|-----|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |

Driver Worksheet

DRIVER WORKSHEET HEADER

The header of the Driver Worksheet allows you to define the tags that start read/write events, such as tags that contain the *status* of the communication.

- **Description Field:** Text for the application documentation.
- **Increase Read Priority Check-box:** Makes a Read Command be treated as a Write Command. This will cause this read message to be the next communication message to be treated.
- **Read Trigger Field:** A field for inserting a tag that allows the execution of a worksheet reading. When you change this tag's value, a worksheet read is performed.
- **Enable Read when Idle Field:** A field for inserting a tag that allows the execution of a worksheet reading. When you change this tag's value, a worksheet read is performed.



CAUTION

Using a constant value not 0, be sure that continuous reading is needed in you application, because this will place a reading request in every communication scan.

- **Read Completed Field:** The communication driver toggles the tag in this field when a read command is completed.
- **Read Status Field:** The tag in this field is updated with the last read command status.

⇒ **If the status value is a negative number, its Description is listed in the UNICOMM.MSG file in the *InduSoft* \BIN\ directory.**

- **Write Trigger Field:** Activates a group reading. Any time its value changes, the program writes an equipment worksheet.
- **Enable Write on Tag Change Field:** Accepts a tag or constant value. Whenever the value is not 0, the communication driver continuously checks for a change in a tag value in the worksheet. If a change occurs, this value will be written in the address field equipment.
- **Write Completed Field:** The communication driver toggles the tag in this field when a write command is completed.
- **Write Status Field:** The tag in this field is updated with the last write command status.

⇒ **If the status value is a negative number, its Description is listed in the UNICOMM.MSG file in the *InduSoft* \BIN\ directory.**

- **Station Field:** Equipment station number in the network. The syntax in this field varies depending on the communication driver.
- **Header Field:** Identifies the kind of data type and/or initial address to be read or written in the equipment. The syntax in this field varies depending on the communication driver.

Example:

Station: {tagStation}, Header: MEMORY {tagAddress}.

⇒ **The Station and Header fields can contain text with tag values using the syntax: text{tag}.**

- **Check-box:** Selecting the check box allows you to set minimum and maximum values for data from the field equipment.
- **Min. and Max. Fields:** These fields are only enabled if the check box to the left is selected. When selected, it enables a range of values, which can be converted into an engineering format. These fields determine the minimum and maximum range of values. Ex.: memory holds values from 0 to 4095 meaning 0% to 100% in the user interface. This setting takes effect for all tags in the worksheet. In the above example the tag parameters min and max must be set 0 to 100.

DRIVER WORKSHEET BODY

The body of the Driver Worksheet allows you to configure the relationship between tags in the application and their field equipment address. Please refer to specific protocol documentation.

⇒ **The maximum number of tags in each communication worksheet of the driver is 512. For some drivers, this number may be smaller. (see driver documentation).**

- **Tag Name Field:** Tag name to be used by the communication driver.
- **Address Field:** Field equipment address related to the application tag. The syntax varies depending on the communication driver.
- **Div Field:** Specifies the division constant when scale adjustment is required. This value will be a division factor in a reading operation and a multiplication factor in a write operation. Do not use this field if you are already using the Min, Max in the configuration body.
- **Add Field:** Specifies the addition constant when scale adjustment is required. This value will be an addition factor in a reading operation and a subtraction factor in a write operation. Do not use this field if you are already using the Min, Max in the configuration body.

⇒ **For read operations: $\langle tag \rangle = (\langle value \text{ in the equipment} \rangle) / Div + Add$. For write operations: $\langle value \text{ in the equipment} \rangle = (\langle tag \rangle - Add) * Div$. If you leave the cells empty in the Div and Add fields, this function is ignored.**



CAUTION

There are *Studio* versions with different limits in the number of communication points. This limit is the total of configured tags in all communication worksheets of the Driver Configuration.

OPC

Contents

Information about the OPC client module. **This file should be printed if you intend to use the OPC module.**

The InduSoft OPC Client module enables the InduSoft system to communicate with any device that implements an OPC Server. This module implements the OPC standard as described in the document "OLE for Process Control Data Access Standard Version 1.0A", available at the site <http://www.opcfoundation.com>.

Before using the InduSoft OPC Client module, you need to install and configure the OPC server in the machines your application will run it.

OPC Client Configuration

In the client machine, you need to use the OPC Client Configuration program to configure the Server Identifier, communication parameters and the items you want to connect.

- ◆ To access the client configuration insert a new OPC Client document at the “COMM” table.

The configuration table for OPC has the following entries:

- **Description:** this field is used for documentation only. The OPC Client module ignores it.
- **Server Identifier:** this field should contain the name of the server you want to connect. If the server is installed in the computer, its name can be selected through the list box.
- **Disable:** this field should contain the a tag or a constant. If its value is different of zero, the communication with the OPC server is disabled.
- **Update Rate:** this field indicates how often the server will update this group in milliseconds. If it is zero indicates the server should use the fastest practical rate.
- **Percent Deadband:** this field indicates the percent change in an item value that will cause a notification by the server. It's only valid for analog items.
- **Tag Name:** this field indicates the percent change in an item value that will cause a notification by the server. It's only valid for analog items.
- **Item:** these fields should contain the name of the server's items. Once you have selected an OPC Server, you can select items from the Server using the OPC Browser. Right-click in the Item field and select the OPC Browser option.

Running the OPC Client Module

- ◆ Run the program InduSoft OPC Client Runtime module automatically or by the menu “Project->Status”.

After running this program, a small icon will appear in your system tray.

- ◆ To close the InduSoft OPC Client module, right-click its icon in the system tray, and select Exit.



TCP/IP

Contents

Information about the use of the TCP/IP client/ server modules.

The InduSoft TCP/IP Client/Server modules enable two or more applications to keep their databases synchronized. These modules use TCP/IP protocol to make the communication between the applications.

Before using the InduSoft TCP/IP Client/Server modules, you need to install and configure the TCP/IP protocol in the machines you will run these modules.

Server Configuration

In the server machine, you don't need to configure anything. You just need to run the module InduSoft TCP/IP Server. You can choose running it automatically on the Start up, or manually, on the Menu Project->Status.

After running this program, a small icon will appear in your system tray.

- ◆ To close the InduSoft TCP/IP Server runtime, right-click its icon in the system tray, and select Exit.



Client Configuration

In the client machine, you need to use the TCP/IP Client Configuration to configure the Server IP address and the tags you want to share with the server.

How to Use the TCP/IP Client Configuration

The TCP/IP Client Configuration program is located on the "Comm" table and it has the same commands of the Driver Configuration program.

TCP/IP Client Parameters Description

Here is a description of the fields you need to fill in the TCP/IP Client Configuration:

- **Description:** this field is used for documentation only. The TCP/IP Client module ignores it.
- **Connection Status:** this field should contain a tag name. The TCP/IP Client Configuration module will update this tag according to the connection status. If the tag value is 0 (zero), then the connection is OK. Otherwise, it's the error code returned by the Windows Socket library.
- **Server IP Address:** this field should contain the IP Address of the server. It may be a string, or you may use a tag enclosed by brackets. For example, if you fill this field with `{tag_name}`, the TCP/IP Client module will try to connect to the server indicated by the tag `tag_name`.
- **Tag Name:** these fields should contain the tags you want to share with the server. If the tag is an array or a class (or both), every element and member is shared. You should only put the tag name in this field, without specifying the index or class member. If you specify an index or a class, the TCP/IP Client module will ignore it.
- **Remote Tag:** these fields should contain the name of the tag that will be linked with the tag specified in the field Tag Name. This field is optional. If you leave it in blank, the same tag name will be used in the client and in the server.



CAUTION

If you need to share an array, then the tag in the server should contain the same number of elements of the tag in the client. If the tag is a class, then the class definition should be the same in both server and client applications. If you don't follow these rules, unpredictable results may happen.

Running the TCP/IP Client Module

- ◆ Run the program InduSoft TCP/IP Client Runtime automatically or manually using the menu “Project->Status”.

After running this program, a small icon will appear in your system tray.



Custom Parameters

There are three parameters you can configure in the Application Configuration (.app) file:

[TCP]

| | | | |
|-----------------------|--|-----------|--------|
| Port=< | TCP/IP port number. | Default = | 1234 > |
| SendPeriod=< | Time in milliseconds; the client/server module will update the tag values of the other machine | Default = | 250 > |
| ConnectRetryTimeout=< | Time in milliseconds; the client/server module will update the tag values of the other machine | Default = | 30 > |

The Port parameter should be the same in both the client and server machines.

Only the client module uses the ConnectionRetryTimeout.

DDE and NetDDE

Dynamic Data Exchange (DDE) is a protocol for dynamic data exchange between Windows applications like Excel and any other Windows program that supports this interface. A DDE conversation is an interaction between server and client applications. **Studio** provides interfaces to run as a client or as a server. See **DDE Client Runtime** and **DDE Server** in the **Runtime Tasks** under the **Status** option of **Project** on the Main Menu Bar. To run as a server, simply start the DDE or NetDDE server task described in **Runtime Tasks**. To run as a DDE client, configure the DDE interface worksheet in the **Communication** tab.

Network Dynamic Data Exchange (NetDDE) is an extension of DDE that works across computers on a network. To run **Studio** as a server to a NetDDE connection, you need to start the application **DDE Server**. To run **Studio** as a client to a NetDDE connection, use the same **DDE** interface worksheets with the proper configuration to address a **Studio** application.

- ⇒ **When running NetDDE, only the WRITE triggers are accepted. To read data, configure a write command on the server computer.**

DDE WORKSHEET

- ◆ Right-click on the **DDE** folder to insert a new worksheet. Clicking the prompt opens a **DDE Worksheet**.

| | Tag Name | Item |
|---|----------|------|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |

DDE Client Configuration Worksheet

The **DDE Worksheet** is divided in two parts:

- a *header* with information for the whole group,
- and a *body* with tags and items related to the DDE server application.

Every DDE interface is based on addressing an application by three structures, namely Application Name, Topic, and Item. The first task is to find these identifiers in the DDE Server application.

DDE WORKSHEET HEADER

The header of the DDE Client allows you to define the tags that will start the reading and writing, as well as the tags that receive the connection status.

- **Description Field:** Worksheet description for documentation purposes.
- **Application Name Field:** DDE server application name.
- **Topic Field:** Topic in the server application.
- **Connect Field:** Tag that controls the connection of **Studio** DDE client and the DDE server application. When this tag is set to 1, it requests a connection to the server. If the connection is not possible or if it fails, **Studio** sets the tag to 0 (zero) again. If the connection is OK, this value remains set to 1.
- **Read Trigger Field:** Tag that commands the reading of the table. When this tag changes value, a polling to the DDE server is generated. This option can be used only with local DDE, not with NetDDE servers.
- **Enable Read when Idle Field:** When the tag inserted in this field is higher than 0, a reading of the equipment is enabled.
- **Read Status Field:** Status of the reading command.
- **Write Trigger Field:** Tag that enables poke commands to be generated to the server.

- **Enable Write on Tag Change Field:** Whenever the inserted tag in this field is higher than 0 (zero), the communication driver continuously checks for a change in a tag value in the worksheet. If a change occurs, the changed tag is written on the equipment along with its address.
- **Write Status Field:** Status of the writing command.

DDE WORKSHEET BODY

The *body* of the DDE client worksheet allows you to configure that each tag is related to each **ITEM** part of the DDE server address.

- **Tag Name Field:** Tag of the *Studio* database to be read or written from the DDE server application.
- **Item Field:** The **ITEM** part of the DDE address on the server. Please refer to the server software documentation for information about the proper syntax of **APP**, **TOPIC**, and **ITEM**. You can configure the **Topic** and **Item** fields with tags on the address using the syntax: **text{tag}**. The value of {tag} is evaluated to a string and used on the address.

Examples:

Topic: topic_{tag_topic_name}_example;

Item: {tag_item_name} or A{tag_number}.

Configuration of the NetDDE Client to the NetDDE Server

Configuring a **NetDDE** connection is like configuring a DDE connection, except for the Header Application name and topic. Before you start your tests verify that DDE Server is enabled on the station with which you want to exchange data.

⇒ **When connecting to servers other than *Studio*, please refer to the server documentation for information about the proper syntax of APP, TOPIC, and ITEM.**

DDE WORKSHEET HEADER TO NETDDE SERVER

- **Application Name Field:** <computer name>\NDDE\$, <computer name>: Must be a valid name for a network computer.
- **Topic Field:** To connect to another *Studio* station, use the topic name: UNISOFT\$.

DDE WORKSHEET BODY TO NETDDE SERVER

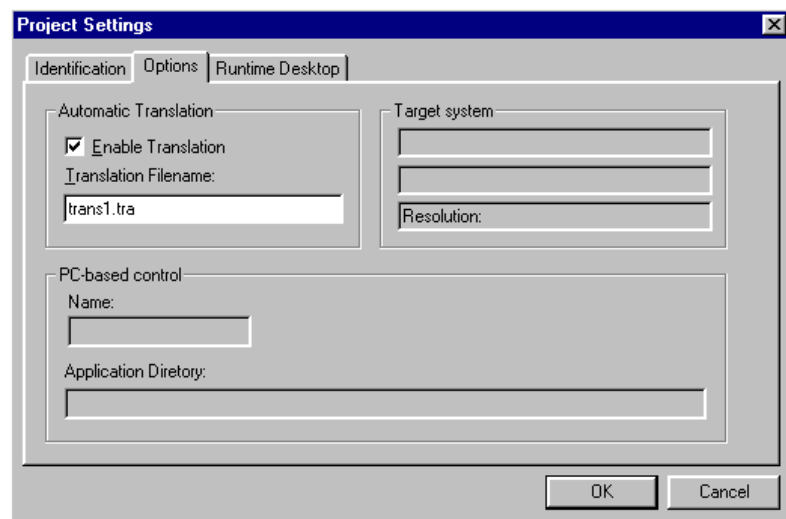
- **Tag Name Field:** *Studio* local database tagname, related to some remote tagname.
- **Item Field:** Remote tagname, related to the local tagname.

3.6 Translation Tool

When developing an application, you can translate it to another language without recreating display or alarm messages. Simply create a file with the translation strings.



When you execute the application, this information will be presented in the selected language.

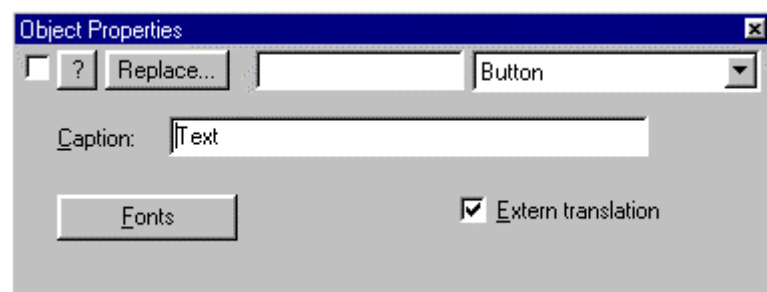
The **Translation Tool** utility creates the text file with the original texts and the translations. In addition to creating the translation file, this feature must be enabled and the translation file must be specified on the **Project Settings Options Tab** under **Project** of the Main Menu Bar. This defines the translation file name and enables automatic translation.



Enable Translation Check-box on the Options Tab of the Project Settings Window

OBJECT PROPERTIES WINDOWS

- ◆ Create a text or button object with their respective **Text**  icon or **Button**  icon on the Object Editing Toolbar.
- ◆ Double-click on the objects to access their **Object Property** windows and select the **Extern Translation** check-box.

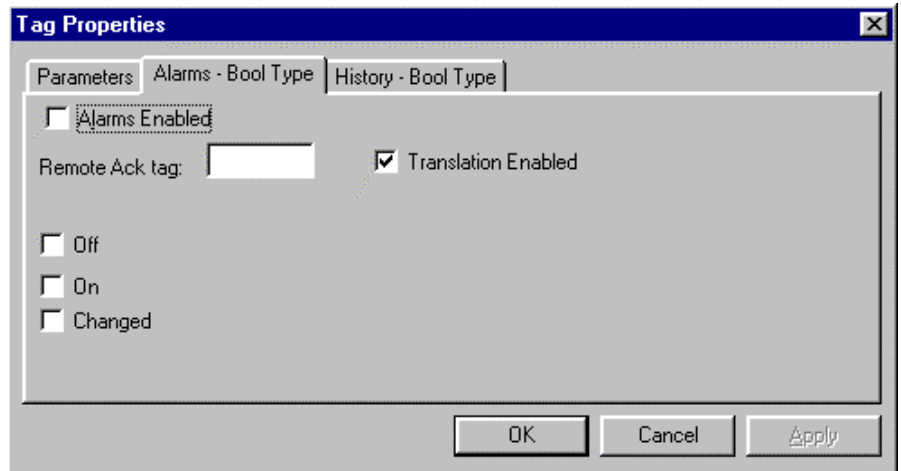


Translation Enabled Check-box on an Object Properties Window

TAG PROPERTIES WINDOW

- ◆ Select an alarm tag and click the **Tag Properties**  icon on the Tag Properties Toolbar.

This opens a **Tag Properties** alarm window that has a **Translation Enabled** check-box.



Translation Enabled Check-box on the Tag Properties Window

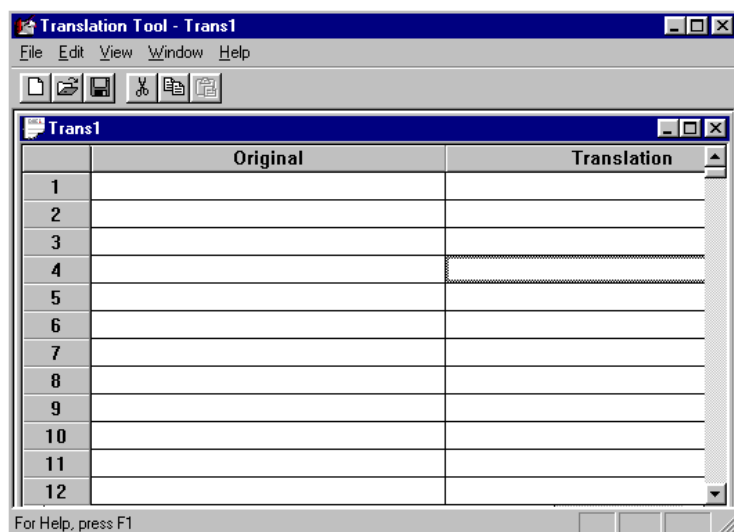
MATH EXPRESSIONS

On worksheets with math expressions, use the **InduSoft** Scripting Language function for translation **EXT()**.

Translation File

To create a Translation File, you need to open the **Translation Editor** under **Tools** in the Main Menu Bar. This opens the **Translation Tool Worksheet**.

If you want to translate into a third language, open another worksheet by selecting the **File/New** menu option on the **Translation Tool** worksheet.

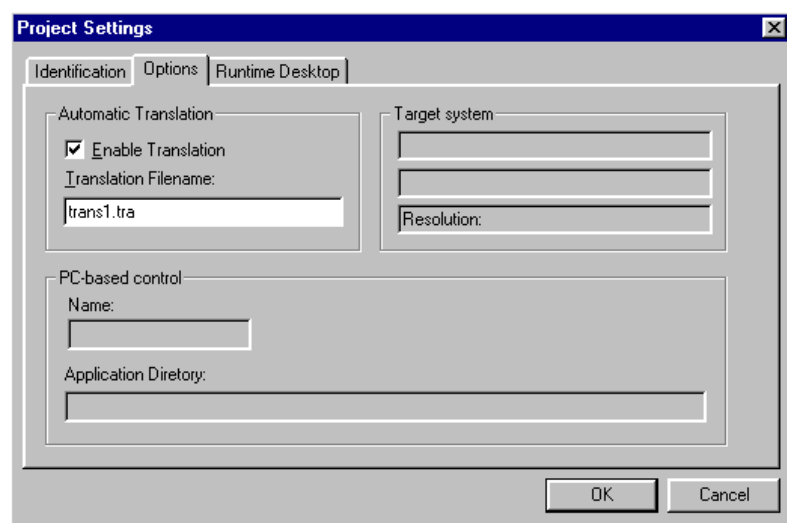


Test Translation Worksheet

Translation Tool Menus:

- **File:** New, Open, Close, Save, Save As, Recent File, Exit
- **Edit:** Cut, Copy, Paste, Find, Insert Line
- **View:** Line, Toolbar, Status Bar, Restore Defaults
- **Window:** New Window, Cascade, Tile, Arrange Icons, Currently open windows
- **Help:** About Translation Tool

⇒ You must select the name of the translation file that the application will use on the Option tab of Project Settings under Project on the Main Menu Bar. But *Studio* can dynamically change text when running an application by using the *InduSoft* Scripting Language function for translation `SetTranslationFile()`.



Translation Filename on the Options Tab of the Project Settings Window

3.7 Functions List

InduSoft Scripting Language has more than one hundred functions ready for use.

SEND MESSAGES TO THE LOGWIN

TRACE(strOutputMessage)

ARITHMETIC FUNCTIONS

ABS(numValue)

DIV(numDivisor, numDividend)

FORMAT(strFormatFlag, numValue)

GETBIT(strTagName, strBitNumber)

MOD(numDivisor, numDividend)

POW(numBase, numExponent)

RESETBIT(strTagName, strBitNumber)

ROUND(numValue)

SETBIT(strTagName, strBitNumber)

SQRT(numValue)

SWAP16(strTagName)

SWAP32(strTagName)

TRUNC(numValue)

STATISTIC FUNCTIONS

AVG(numValue1, numValue2, ... , numValueN)

MAX(numValue1, numValue2, ... , numValueN)

MIN(numValue1, numValue2, ... , numValueN)

RAND()

LOGARITIMIC FUNCTIONS

EXP(numExponent)

LOG(numLogArg)

LOG10(numLogArg)

LOGIC FUNCTIONS

IF(numCondition, numThen, numElse)

TRUE(numExpression)

FALSE(numExpression)

FUNCTIONS FOR STRINGS MANIPULATION

ASC2STR(strChar1, strChar2, ... , strCharN)

CHARTOVALUE("strTagName", "numArray")

CHARTOVALUEW("strTagName", "numArray")

NCOPY(strSource, numStartChar, numQtdChar)

NUM(strValue)

STR(numValue)

STR2ASC(strChar)

STRLEFT(strSource, numQtdChars)

STRLEN(strSource)

STRLOWER(strSource)

STRRCHR(strSource, strCharSequence)

STRRIGHT(strSource, numQdeChars)

STRSTR(strSource, strSequence)

STRSTRPOS(strSource, strCharSequence)

STRTRIM(strReference, numOptionalFlag)

STRUPPER(strValue)

VALUETOCHAR("numArray", numQdeChar)

VALUEWTOCHAR("numArray", numQdeChar)

DATE AND TIME MANIPULATION

CLOCKGETDATE(numSeconds)

CLOCKGETDAYOFWEEK(numSeconds)

CLOCKGETTIME(numSeconds)

DATETIME2CLOCK(strDate, strTime)

GETCLOCK()

HOURL2CLOCK(strTime)

SETSYSTEMDATE(strDate)

SETSYSTEMTIME(strTime)

TRIGONOMETRIC FUNCTIONS

ACOS(numValue)

ASIN(numValue)

ATAN(numValue)

COS(numAngle)

COT(numAngle)

PI()

SIN(numAngle)

TAN(numAngle)

FUNCTIONS FOR OPENING AND CLOSING WINDOWS

OPEN(strScrFile, numOptionalX1, numOptionalY1, numOptionalX2, numOptionalY2)

CLOSE(strScrFile)

SECURITY SYSTEM

CREATEUSER(strUserName, strGroup, strPassw)

REMOVEUSER(strUserName)

MODULE ACTIVATION FUNCTIONS

SHUTDOWN()

APPACTIVATE(strAppTitle, numOptionalActiv)

APPISRUNNING(strAppTitle)

APPPOSTMESSAGE(strAppTitle, numwParam, numlParam)

APPSSENDKEYS(strKeys1, strKeys2, ... , strKeysN)

CLEANREADQUEUE()

CLOSESPASHWINDOW()

DISABLEMATH()

ENABLEMATH()

EXITWINDOWS(numExitCode)

ISSCREENOPEN(strScrName)

```

ISVIEWERINFOCUS()
LOGOFF()
LOGON(strOptionalUser, numOptionalPassw)
MATH(numMathWorksheet)
NOINPUTTIME()
RECIPE(strOperation&File)
REPORT(strOperation&File)
SETAPPPATH(strDirPath)
SETVIEWERINFOCUS()
VIEWERPOSTMESSAGE(strScrTitle, numwParam, numlParam)
WAIT(numMilliseconds)

```

**CAUTION**

The wait function may only be used in Math worksheets. However, IT IS DANGEROUS to use this function anywhere in your application. Wait() pauses the application, any information coming into the application during a wait is ignored.

```

WINEXEC(StrFilePath, numOptionalState)
SENDKEYOBJECT(numEvent, strMainKey, numShift, numCtrl, numAlt, strTargetScreen)
SETVIEWERPOS(numLeft, numTop, numOptionalWidth, numOptionalHeighth)
STARTTASK(strTaskName)
ISTASKRUNNING(strTaskName)
ENDTASK(strTaskName)

```

FILE MANIPULATION FUNCTIONS

```

FILECOPY(strSourceFile , strTargetFile)
FILEDELETE(strFilePath)
FILELENGTH(strFileName)
FILERENAME(strOldName , strNewName)
FINDFILE(strFileMask)
PRINT(strFilePath)
RDFILEN(strSelectedFile, strSearchPath, strMask, numChangeDir)
GETFILEATTRIBUTES("strFileName")
GETFILETIME("strFileName", strNumber)

```

GETLINE(strFileName, strSeqChar, "strStoreTag", numOptCase, numOptOverflowTag)

FUNCTIONS FOR GRAPHICS SCREENS PRINTING

PRINTWINDOW(strScrName)

FUNCTIONS FOR TEXT TRANSLATIONS

EXT(strText)

SETTRANSLATIONFILE(strFileName)

MULTIMEDIA FUNCTIONS

PLAY(strWavFile)

SYSTEM INFORMATIONS

DBVERSION()

GETAPPHORIZONTALRESOLUTION()

GETAPPVERTICALRESOLUTION()

GETCOMPUTERNAME()

GETHARDKEYMODEL()

GETHARDKEYSN()

GETPRODUCTPATH()

GETOS()

GETPRIVATEPROFILESTRING(str_Section, str_Name, str_Default, str_FileName)

GETTICKCOUNT()

INFOAPPALRDIR()

INFOAPPDIR()

INFOAPPHSTDIR()

INFODISKFREE(strDiskUnit)

INFORESOURCES(numResourceCode)

NOINPUTTIME()

PRODUCTVERSION()

SETAPPALARMPATH (strPath)

SETAPPHSTPATH(strPath)

SETDATEFORMAT(strSeparator, strDateFormat)

DATABASE ACCESS FUNCTIONS

FORCETAGCHANGE(strTagName, numValue)

LOOPS

FOR(numInitialValue, numFinalValue, numStep)

NEXT

MAIL FUNCTIONS

CNFEMAIL (strSmtip, strFrom)

SENDKEYOBJECT(numEvent, strMainKey, numShift, numCtrl, numAlt, strTargetScreen)

3.7.1 Send Message to the LogWin

TRACE(arg)

Description: Shows the contents of *arg* in the LogWin screen. Arg is a string constant or a string tag.

Examples:

```
TRACE("The value of the count has changed")
TRACE(DATE)
```

3.7.2 Arithmetic Functions

ABS(arg)

Description: Returns the absolute value of argument.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------|
| Level | -20.153 |
| Temp | abs(level) // temp=20.153 |

DIV(arg1, arg2)

Description: Truncates and returns the division quotient of *arg1* by *arg2*.

Examples:

```
Div (temp, level)
Div (temp,4)
Div (4,level)
```

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------------|
| Level | 5.648 |
| Temp | 2 |
| Result | Div (level, temp) // result = 2 |

FORMAT(arg1, arg2)

Description: Creates a formatted *string* from a number.

Parameters: *arg1* must be the mask: “%[0*n*] [flag]”, and *arg2* is the number to be formatted.

flags:

| | |
|----------|---------------------------------------|
| d, D | decimal |
| x, X | hexadecimal |
| o, O | octal |
| b, B | binary |
| f, F | real |
| e, E | scientific notation |
| g, G | the same as F and E, but more compact |
| s, S | string |
| c, C | ASCII character |
| h, H | hours |
| <i>n</i> | The number of digits to be shown |

Examples:

| <u>Tag Name</u> | <u>Expression</u> | <u>Result</u> |
|-----------------|------------------------------|---------------|
| Output[1] | format("%b", 8) | 1000 |
| Output[2] | format("%x", 255) | ff |
| Output[3] | format("%02X", 15) | 0F |
| Output[4] | format("%o", 8) | 10 |
| Output[5] | format("%x", 17) | 10 |
| Output[6] | format("%f", 237.8) | 237.800000 |
| Output[7] | format("%d", level) | 97 |
| Output[8] | format(string_format, level) | 97 |
| Output[9] | format("%c", 38) | & |
| Output[10] | format("%c", 49) | 1 |
| Output[11] | format("%h", 37230) | 10:20:30 |

⇒ **This function accepts the same *flags* after the symbol “%” that are used in “C” standard function *printf()*, but only one value can be formatted in each cell.**

GetBit (strTagName, strBitNumber)**Description:** Gets a bit from the tag.

| | |
|----------------------|--------------------|
| Return Value: | Error codes: |
| Bit value | No error |
| -1 | Invalid parameter |
| -2 | Tag does not exist |

Example:

GetBit("minute",2)

MOD (arg1, arg2)**Description:** Returns the remainder of *arg1* by *arg2*.**Examples:**

Mod (level, temp)

Mod (level, 4)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------------|
| Level | 20 |
| Temp | 7 |
| Result | Mod (level, temp) // result = 6 |

POW(arg1, arg2)**Description:** Returns the value of *arg1*(base) raised to a power *arg2*(exponent).**Examples:**

pow(base,exponent)

pow(base,7)

pow (5,exponent)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|------------------------------------|
| Base | 2 |
| Exponent | 3 |
| Result | Pow (base, exponent) // result = 8 |

ResetBit (strTagName, strBitNumber)**Description:** Resets a bit from the tag.

Return Value: Error codes:

| | |
|---|--------------------|
| 0 | No Error |
| 1 | Invalid parameter |
| 2 | Tag does not exist |

Example:

ResetBit("hour",1)

ROUND(arg)**Description:** Rounds the value of the *arg* argument to the next greater integer value.**Examples:**

Round(level)

Round(-23.44)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|------------------------------|
| Level | 21.67 |
| Result | Round (level) // result = 22 |

SetBit (strTagName , strBitNumber)**Description:** Sets a bit from the tag.

Return Value: Error codes:

| | |
|---|--------------------|
| 0 | No Error |
| 1 | Invalid parameter |
| 2 | Tag does not exist |

Example:

SetBit("second",0)

SQRT(arg)**Description:** Returns the square root value of the *arg* argument.**Examples:**

Sqrt(level)

Sqrt(24)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------------------|
| Level | 24 |
| Result | Sqrt (level) // result = 4.898979 |

Swap16

Description: Swaps the two lower bytes from the tag. Returns an integer with the binary value correspondent to the swap of the two lower bytes from the tag.

Example:

Swap16(test16)

⇒ If the binary value of test16 is 1001111100000110, the function Swap16 returns the binary value 0000011010011111.

Swap32

Description: Swaps the two words from the tag. Returns an integer with the binary value correspondent to the swap of the two words from the tag.

Example:

Swap32(test32)

⇒ If the binary value of test32 is 10011111000001100000111111110000, the function Swap32 returns the binary value 00001111111100001001111100000110.

TRUNC(arg)

Description: Returns the integer part of the *arg* argument.

Examples:

Trunc(level)

Trunc(-23.44)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|------------------------------|
| Level | 15.2345 |
| Result | Trunc (level) // result = 15 |

3.7.3 Statistic Functions

AVG(arg1, arg2, ... argN)

Description: Returns the arithmetic average of the defined arguments.

Examples:

Avg(level,temp)

Avg(-23.44,level,temp)

Avg(12,24,32,88)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--------------------------------------|
| Level | 20 |
| Temp | 40 |
| Result | Avg (level, temp) // result = 30 |
| Result | Avg(10,level,30,temp) // result = 25 |

MAX(arg1, arg2, ... argN)

Description: Returns the highest value among the defined arguments.

Examples:

Max(level, temp)

Max(-23.44, level, temp)

Max(12,24,32,88)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Level | 20 |
| Temp | 40 |
| Result | Max (level, temp) // result = 40 |
| Result | Max(10,level,30,temp,100) // result = 10 |

MIN(arg1, arg2, ... argN)

Description: Returns the lowest value among the defined arguments.

Examples:

Min(level,temp)

Min(-23.44,level,temp)

Min(12,24,32,88)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--------------------------------------|
| Level | 20 |
| Temp | 40 |
| Result | Min (level, temp) // result = 20 |
| Result | Min(10,level,30,temp) // result = 10 |

RAND()

Description: Generates a random number in floating point between 0 and 1.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------------|
| Result | Rand() // result = 0.104892 |

3.7.4 Logarithmic Functions

EXP(arg):

Description: Calculates the *arg* argument exponential ($e = 2.71828\dots$).

Examples:

Exp(LEVEL)

Exp(22)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Level | 22 |
| Result | Exp(level) // result = 3584912846.131592 |

LOG(arg)

Description: Calculates the *arg* argument logarithm ($e = 2.71828\dots$). Also known as natural logarithm.

Examples:

Log(level)

Log(22)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------------|
| Level | 22 |
| Result | LOG(level) // result = 3.091042 |

LOG10(arg)

Description: *arg* logarithm calculated in the base 10.

Examples:

Log10(level)

Log10(22)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------------------|
| Level | 22 |
| Result | Log10(level) // result = 1.342423 |

3.7.5 Logic Functions

IF (condition, true, false)

Description: Conditional execution

Parameters:

| | |
|-----------|---|
| condition | Expression to be tested |
| true | Result expression in case of true condition |
| false | Result expression in case of false condition (optional parameter) |

Return Value: If the expression in the *condition* parameter is true (or more than zero), the *true* expression result returns, otherwise, the *false* expression. If the condition result is false (or equal to zero), or if the parameter was not declared, the tag of the Tag Name column remains unchanged.

Example:

if (tag > 20, tag/2, abs(count))

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| Account | if (account=10, 0, account+1) |
| Comment | If the value of the tag account = 10, account receives the value 0 (zero), otherwise, 1 will be added to its actual value |

⇒ **The Database Spy utility was not projected to evaluate this function in a direct way.**

TRUE(arg)

Description: Verifies if the expression *arg* is true.

Return Value: Error Codes:

| | |
|---|----------------------------|
| 1 | If the expression is true |
| 0 | If the expression is false |

Examples:

true (newtag)

true (a > b)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Cond | TRUE(cond=10) |
| Comment: | If the value of tag cond = 10, the tag cond will receive the value 1; otherwise, it will receive 0 |

FALSE(arg)

Description: Verifies if the *arg* expression is false.

Return Value:

| | |
|---|----------------------------|
| 1 | If the expression is false |
| 0 | If the expression is true |

Examples:

false (newtag)

false (a > b)

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| Cond | FALSE(cond=10) |

Comment: If the value of tag **cond** = 10, **cond** will receive the value 0; otherwise, it will add 1 to its actual value

3.7.6 Functions for Strings Manipulation**Asc2Str(arg1, arg2, ... argN)**

Description: Enchains characters in ASCII code to form a *string*.

Example:

49 is the ASCII code value for the character "1" and the number 50 is of the character "2".

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Name | "test" |
| New_string | Asc2Str(test, 49, 50) // new_string = "test12" |

CharToValue("strTagName" , "numArray")

Description: Converts a string to integer array. Returns the number of chars. The trigger defines when the values must be updated.

Examples:

| | |
|-----------|--|
| Tagstring | = "ABC" |
| Tagnum | ValueToChar("tagstring", "vet[1]") => tagnum=3 |
| Vet[1] | = 65 // char 'A' |
| Vet[2] | = 66 // char 'B' |
| Vet[3] | = 67 // char 'C' |

CharToValueW()

Description: Same as CharToValue but using words instead of bytes.

NCOPY(str, n1, n2)

Description: Returns a substring, starting with the n1 and n2 characters.

Parameters:

| | |
|-----|---|
| str | String or tag type string from which you want to extract a sub-string |
| n1 | Initial position of the sub-string |
| n2 | Number of the sub-string characters |

Return Value: String that starts in the n1 of str characters and has the n2 size.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Name | "System" |
| New_string | NCOPY(name, 3, 4) // new_string = "stem" |

NUM(string)

Description: Converts a *string* (or tag type *string*) to a numeric value.

Example:

| num ("4") | |
|-----------------|------------------------------|
| <u>Tag Name</u> | <u>Expression</u> |
| new_tag | "4" |
| New_int | NUM (new_tag) // New_int = 4 |

STR(val_num)

Description: Converts a numeric value (tag or value) to a *string*.

Example:

| str (3) | |
|-----------------|------------------------------|
| <u>Tag Name</u> | <u>Expression</u> |
| New_tag | "5" |
| Str_n | STR (new_tag) // str_n = "5" |

Str2Asc(arg1)

Description: Returns the ASCII code of a character.

Parameter: *arg1* is a string.

Return Value: Integer

Example:

49 is the ASCII code value for the character "1"

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------|
| Name | "1" |
| Num | Str2Asc(name) // num = 49 |

StrLeft (arg1, arg2)

Description: Returns the bytes to the left of the *string arg1* (tag or value).

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--------------------------------------|
| Strin | StrLeft("test",2) // strin = "te" |
| Strin | "TESTING" |
| New_str | StrLeft(strin,4) // new_str = "test" |

StrLen(arg)

Description: Returns the length in bytes of the *string arg*.

Parameters: *string* tag type *string*

Return Value: Integer numeric

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|----------------------------|
| Size | StrLen("test") // size = 4 |
| Strin | "test" |
| Size2 | StrLen(strin) // size2 = 6 |

⇒ **Don't forget that for variables of the string type, the double quotation marks (") are considered characters.**

StrLower (arg)

Description: Converts a *string* to lowercase letters.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Strin | StrLower("Test") // strin = "TEST" |
| Strin | "TESTING" |
| New_str | StrLower(strin) // new_str = "testing" |

StrRChr("string, "char")

Description: Find a character ("char") in a string ("string"). It returns a string to the first occurrence of "char" in "string", or "" (NUL string) if "char" is not found.

StrRight (arg1, arg2)

Description: Returns the bytes to the right of the *string arg1*.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------------------|
| Strin | StrRight("test",2) // strin = "st" |
| Strin | "TESTING" |
| New_str | StrRight(strin,4) // new_str = "ting" |

StrStr(arg1, arg2)

Description: Searches the first occurrence of the *string arg2* in the *string arg1*.

Parameters:

| | |
|------|--|
| arg1 | String or tag type string that performs the search |
| arg2 | Sequence of characters to be searched |

Return Values:

| | |
|--------------|---|
| String arg1 | starting where the sequence arg2 is found |
| String empty | if it was not found |

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Name | "test" |
| New_string | StrStr(name, "s") // new_string = "st" |
| New_string | StrStr("test", "s") // new_string = "st" |

StrStrPos(arg1, arg2)

Description: Searches the first occurrence of the *string* *arg2* in the *string* *arg1*.

Parameters:

arg1 String or tag type string that performs the search
arg2 Sequence of characters to be searched

Return Values:

Integer number with the start position, or
-1 if the string was not found

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Position | StrStrPos("test", "s") // position = 2 |

⇒ **The first position of the string is considered to be the number zero position.**

StrTrim (arg1, arg2)

Description: Removes the white spaces of the *string* *arg1*.

Parameters:

arg1 String or tag type string
arg2 Optional parameter
0 Remove from the right to the left (default)
1 Remove from the left
2 Remove from the right

Return Value: string

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|----------------------------------|
| Strin | "test" |
| Strin | StrTrim(strin) // strin = "test" |

StrUpper(arg)

Description: Converts a *string* for uppercase letters.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Strin | StrUpper("test") // strin = "TEST" |
| Strin | "testing" |
| New_str | StrUpper(strin) // new_str = "TESTING" |

ValueToChar()

Description: Converts an integer array to string. Returns the string. The trigger defines when the values must be updated.

Example:

```
Tagstring ValueToChar("vet[1]",3) => tagstring="ABC"
Vet[1]=65 // char'A'
Vet[2]=66 // char'B'
Vet[3]=67 // char'C'
```

ValueWToChar()

Description: Same as ValueToChar but using words instead of bytes.

3.7.7 Date and Time Manipulation**ClockGetDate(arg)**

Description: Returns the related date with the number of elapsed seconds as a parameter. The base date is 31/12/1969.

Parameter: *arg* is a long integer that contains the date in seconds.

Return Value: *String* in the DD/MM/AA format.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| Date | ClockGetDate(633000000) // data = "22/01/1990" |

ClockGetDayOfWeek (arg)

Description: Returns the day of the week according to the number of elapsed seconds as a parameter.

Parameter: *arg* is a long integer that contains the hour in seconds.

Return Value: Integer numeric:

| | |
|---|-----------|
| 0 | Sunday |
| 1 | Monday |
| 2 | Tuesday |
| 3 | Wednesday |
| 4 | Thursday |
| 5 | Friday |
| 6 | Saturday |

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Temp | ClockGetDayOfWeek (2999999) // temp = 3 |

ClockGetTime (arg)

Description: Returns *hours:minutes:seconds* related with the number of elapsed seconds as a parameter.

Parameter: *arg* is a long integer that contains the data in seconds.

Return Value: *String* in the *HH:MM:SS* format.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| Temp | ClockGetTime(633000000) // temp = "01:20:00" |

DateTime2Clock(arg1, arg2)

Description: Returns the number of seconds, beginning on 31/12/1969 until the *arg1* date and *arg2* hour.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| num_of_sec | DateTime2Clock("10/10/1990","11:02:30") // num_of_sec = 655581750 |

GetClock(arg)

Description: Returns the number of seconds counted, beginning on 31/12/1969 up to the current date and time.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------------------|
| num_of_sec | GetClock(0) // num_of_sec = 862252573 |

Hour2Clock (arg)

Description: Converts a time in number of seconds.

Parameter: *arg* is a *string* (or tag type *string*) with a time (*HH:MM:SS*) to be converted.

Return Value: Integer numeric, value of the space of time in seconds.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| s[1] | Hour2Clock("00:01:00") // s[1] = 60 |
| S[2] | Hour2Clock("10:00:00") // s[2] = 36000 |
| new_time | "10:20:30" |
| s[3] | Hour2Clock(new_time) // s[3] = 37230 |

SetSystemDate (arg)

Description: Modifies the system date in your computer.

Parameters: *arg* is a *string* (or tag type *string*) that contains the desired date.

Return Value: None

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------------|
| | SetSystemDate("22/09/1995") |
| new_date | "23/09/1996" |
| | SetSystemDate(new_date) |

SetSystemTime (arg)

Description: Modifies the system time in your computer.

Parameters: *arg* is a tag or constant of the type *string* with the desired hour.

Return Value: None

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------|
| | SetSystemTime("12:00:00") |
| new_time | "23:09:19" |
| | SetSystemDate(new_time) |

3.7.8 Trigonometric Functions**ACOS(arg)**

Description: Returns the *arg* arc-cosine value.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| value_acos | ACOS(0.997495) // value_acos = 0.070796 |
| value | 0.707107 |
| value_acos | ACOS(value) // value_acos = 0.785398 |

ASIN(arg)

Description: Returns the *arg* arc-sine value.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| value_asin | ASIN(0.997495) // value_asin = 1.5000000 |
| value | 0.707107 |
| value_asin | ASIN(value) // value_asin = 0.785398 |

ATAN(arg)

Description: Returns the *arg* arc-tangent value.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| value_atan | ATAN(14.101420) // value_atan = 1.500000 |
| value | 2 |
| value_atan | ATAN(value) // value_atan = 1.107149 |

COS(arg)

Description: Returns the *arg* cosine (*arg* in radians).

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|------------------------------------|
| value_cos | COS(1.5) // value_cos = 0.070737 |
| Angle | PI()/4 |
| value_cos | COS(angle) // value_cos = 0.707107 |

COT(arg)

Description: Returns the *arg* co-tangent (*arg* in radians).

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------------------|
| value_cotan | ATAN(1.5) // value_cotan = 0.982794 |
| Angle | PI()/4 |
| value_cotan | ATAN(angle) // value_cotan = 0.665774 |

PI()

Description: Returns the value of the PI numeric constant.

Return Value: (= 3.141593) with seven decimal places.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|------------------------------|
| value_pi | PI() // value_pi = 3.141593 |

SIN(arg)

Description: Returns the *arg* sine (*arg* in radians).

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|------------------------------------|
| value_sin | SIN(1.5) // value_sin = 0.997495 |
| Angle | PI()/4 |
| value_sin | SIN(angle) // value_sin = 0.707107 |

TAN(arg)

Description: Returns the *arg* tangent (*arg* in radians).

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|------------------------------------|
| value_tan | TAN(1.5) // value_tan = 14.101420 |
| Angle | PI()/4 |
| value_tan | TAN(angle) // value_tan = 1.000000 |

3.7.9 Functions for Opening and Closing Windows**OPEN(arg, x1, y1, x2, y2)**

Description: Opens a screen or group of screens of an application during the execution.

Parameters:

Tag or constant value of the string type, with the name of the screen (default extension is .SCR) or a group of screens (extension .SG) to be opened.

Arg Name of the screen file or group of screen files. It can be a tag or constant of the string type.

x1, y1, x2, y2 Optional parameters that define the initial coordinates of the window to be opened.

Return Value:

| | |
|---|--------------------------------|
| 0 | Function executed successfully |
| 1 | Function cannot be executed |

Examples:

"screenlb.scr" is the name of a screen created in the Graphical Interface, so:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Status | OPEN("screenlb") // it is the same as OPEN("screenlb.scr") |

CLOSE(arg)

Description: Closes a window in the execution.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| | CLOSE("screenlb") |

**CAUTION**

When you open a window of the Replace style, it automatically closes the windows with Replace and Popup attributes that intercept the new window. In this case, it is not necessary to call the CLOSE(arg) function.

3.7.10 Security System

CreateUser(Username, Group, Password)

Description: Adds a user in the Security System.

Parameters:

| | |
|----------|---|
| UserName | Tag or value of the string type with the name of the user to be inserted in a group of the Security System. |
| Group | Tag or value of the string type with the name of the group in the Security System. |
| Password | Tag or value of the string type with the password for the user. |

Return Value:

| | |
|---|---|
| 0 | Success |
| 1 | Invalid number of parameters |
| 2 | Wrong parameter type |
| 3 | User already exists |
| 4 | Group does not exist |
| 5 | It is not possible to safely write the data |
| 6 | It is not possible to use the CreateUser function |

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Status | CreateUser ("John","Projects","8763") |
| UserName | "John" |
| Group | "Projects" |
| Password | "8763" |
| Status | CreateUser (UserName, Group, Password) |

RemoveUser(Username)

Description: Removes an user in the Security System.

Parameter:

| | |
|----------|---|
| UserName | Tag or value of the string type with the name of the user to be removed from the Security system. |
|----------|---|

Return Value:

| | |
|---|---|
| 0 | Success |
| 1 | Invalid number of parameters |
| 2 | Wrong parameter type |
| 3 | User does not exist |
| 4 | It is not possible to safely write the data |

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------|
| | RemoveUser ("John") |
| UserName | "John" |
| | RemoveUser (UserName) |

3.7.11 Module Activation Functions

ShutDown()

Description: Function for finalization of the system. It closes all of the active runtime programs of **Studio**.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| | ShutDown() |



CAUTION

This function does not close the configuration application, the Database, nor LogWin.

AppActivate (arg1, arg2)

Description: Activates an application.

Parameters:

| | |
|------|--|
| arg1 | String with the application title |
| arg2 | Integer optional parameter of the command activation. See the Windows documentation about the following options: |
| 0 | SW_HIDE |
| 1 | SW_SHOWNORMAL |
| 2 | SW_SHOWMINIMIZED |
| 3 | SW_SHOWMAXIMIZED |
| 4 | SW_SHOWNOACTIVATE |
| 5 | SW_SHOW |
| 6 | SW_MINIMIZE |
| 7 | SW_SHOWMINNOACTIVE |
| 8 | SW_SHOWNA |
| 9 | SW_RESTORE (default) Must be 9 |

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------------------------|
| Status | AppActivate("notepad - (untitled)") |

AppIsRunning (arg)

Description: Verifies if an application is being executed.

Parameters: Tag or string type with the application title.

Return Value:

| | |
|---|---------------------------|
| 0 | Application is running |
| 1 | Application was not found |

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Status | AppIsRunning ("Microsoft Word - test.doc") |

AppPostMessage (arg1, arg2, arg3)

Description: Sends a message to an application.

Parameters:

| | |
|------|--|
| arg1 | Tag or string type value with the application title |
| arg2 | Integer with the Windows message wParam, or tag (or value) of the string type with the following values: "MINIMIZE" "MAXIMIZE" "RESTORE" "CLOSE" |
| arg3 | IParam of the Windows message |

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Status | AppPostMessage("Calculator", "CLOSE", 0) |

AppSendKeys (arg1, arg2, ... argN)

Description: Sends keyboard codes to the *foreground* Windows application.

Parameters:

| | |
|------|---|
| arg1 | Tag or string type value with the commands to be sent or a tag or integer value with the keyboard codes to be sent. |
| arg2 | The same as <i>arg1</i> , but it has a delay of 200 ms between the sending of <i>arg1</i> and <i>arg2</i> . |

**CAUTION**

To send a code equal as the keyboard commands ALT, CTRL, or SHIFT, use <CTRL> or <SHIFT> in the text. To send the < character, send <<.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| Status | WinExec("write.exe c:\windows\README.WRI") AppActivate("Write - README.WRI") AppSendKeys("<ALT>FP") //for Windows version in English Wait(1000) AppSendKeys("<ENTER>") Wait(1000) AppSendKeys("<ESC>") AppPostMessage("Write - README.WRI", "CLOSE") |

Special Keyboard Commands:

The following special keyboard commands are disposable. To use a command, put the text between the brackets (<>).

BACKSPACE

BREAK

CAPSLOCK

DELETE

DEL

DOWN

END

ENTER

ESCAPE

ESC

F1F12

HOME

INSERT

LEFT

NUMLOCK

PGDN

PGUP

PRTSC

RIGHT

TAB

UP

CleanReadQueue()

Description: Removes all messages of reading in the communication drivers.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| | CleanReadQueue() |

CloseSplashWindow()

Description: Closes **Studio** Splash window.

DisableMath()

Description: Stops the execution of the mathematical task until the call of the **EnableMath()**.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| | DisableMath() |

EnableMath()

Description: Enables the execution of the mathematical task after the use of the **DisableMath()**.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| | EnableMath() |

ExitWindows (arg)

Description: Exits Windows.

Parameters: *arg* is an integer from 0 to 2

| | |
|---|----------------|
| 0 | Reboot Windows |
| 1 | Log off |
| 2 | Shut down |

Force (unsaved data is lost)

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| | ExitWindows (1) |

IsScreenOpen (arg)

Description: Verifies if an **Studio** screen is opened in the execution.

Parameters: arg is a tag or value of the string type with the name of the screen.

Return Value:

| | |
|---|---------------------------|
| 0 | If the screen is not open |
| 1 | If the screen is open |

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------------|
| | IsScreenOpen("menu.scr") |
| actual_screen | "menu" |
| Status | IsScreenOpen(actual_screen) |

IsViewerInFocus

Description: Verifies if Viewer task is in the focus.

| | |
|----------------------|--------------------------------|
| Return Value: | Error codes: |
| 1 | Viewer has the focus |
| 0 | Viewer does not have the focus |

LOGOFF()

Description: Disables the **Studio** Log On/Off utility. When a user of a determined group effectuates a *logoff* operation, the *Guest* group is always activated.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| Status | LOGOFF() |

⇒ **Status receives the Return Value.**

LOGON(arg1, arg2)

Description: Activates the **Studio** Log On/Off utility.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| status[1] | LOGON("Smith") LOGON("Smith","senha") |
| | LOGON() |

Math (arg)

Description: Executes the math worksheet. (**Background Task** on the **Runtime Tasks** tab in the **Project Status** window needs to be running.)

Parameters: *arg* is an integer with math worksheet number.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------------|
| | Math (5) // executes math 5 |

NoInputTime()

Description: Returns the time from the last keyboard action.

Return Value: Integer

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| Number | NoInputTime() |

RECIPE (arg)

Description: Activates the recipe functions.

Parameters:

Tag or string type value with a specific format, depending on the operation to be accomplished.

The string format is: "operation:configuration_file".

The valid values of the operation are:

| | |
|--------|---------------------------|
| save | Save values operation |
| load | Load values operation |
| delete | Delete file operation |
| init | Initialize file operation |

Return Value: Error codes:

| | |
|---|-------------------------------------|
| 0 | No errors |
| 1 | The tag is numeric |
| 2 | Expression doesn't contain ":" |
| 3 | Previous command to the invalid ":" |
| 4 | Task not found by the system |
| 5 | Disk error |

**CAUTION**

The recipes work with two kinds of files: *configuration files* and *data files*. The configuration file contains the tag declarations that form a part of a recipe worksheet (or group) that has the .RCP extension. The name of the data file is defined in the report worksheet in the Output File camp; that will contain the report (with the .OUT extension).

**CAUTION**

Background Task on the Runtime Tasks tab in the Project Status window needs to be running in order to execute the recipe functions.

REPORT(arg)

Description: Activates the report functions.

Parameters:

Tag or string type value with specific format that contains the command for a report task.

The string format is "operation:configuration_file".

The valid values of the operation are:

| | |
|------|-----------------------------|
| disk | Disk load operation |
| prn | Sends values to the printer |

**CAUTION**

The reports work with two kinds of files: *Configuration files* and *data files*. The configuration file contains the tag declarations and strings that form a report model (with the .RPT extension). The name of the data file is defined in the report worksheet in the Output File camp; that will contain the report (with the .OUT extension).

| | |
|----------------------|-------------------------------------|
| Return Value: | Error codes: |
| 0 | No errors |
| 1 | The tag is numeric |
| 2 | Expression doesn't contain ":" |
| 3 | Previous command to the invalid ":" |
| 4 | Task not found by the system |
| 5 | Disk error |

**CAUTION**

Background Task on the Runtime Tasks tab in the Project Status window needs to be running in order to execute the report functions.

SetAppPath (arg)

Description: Points the subdirectories **HST** and **ALARM** for the current application.

Parameters: *arg* or a value of the string type with the name of the new application directory.

Return Value:

| | |
|---|---------|
| 0 | Failure |
| 1 | Success |

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|----------------------------|
| | SetAppPath ("C:\INDUSOFT") |
| actual_dir | "C:\INDUSOFT" |
| | SetAppPath (actual_dir) |

SetViewerInFocus

Description: Set focus to Viewer task.

ViewerPostMessage (arg1, arg2, arg3)

Description: Sends an internal message to Viewer.

Parameters:

| | |
|------|--|
| arg1 | Tag or string type value with the screen title |
| arg2 | Integer with the wParam of the Windows message |
| arg3 | lParam of the Windows message |

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Status | ViewerPostMessage("demo.scr", "CLOSE",0) |

Wait (arg)

Description: Interrupts the execution for *arg* milliseconds.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| | Wait (200) // interrupts the execution for 200 ms |

WinExec (arg1, arg2)

Description: Activates an external program to *Studio*.

Parameters:

| | |
|------|--|
| arg1 | Tag or constant of the string type that contains the path of the executable file |
| arg2 | Optional numeric value with the initial state of the new application |
| 0 | Hides the application and gives control to another one |
| 1 | Activates and displays the application (default) |
| 2 | Activates the application and displays it as an icon |
| 3 | Activates the application and maximizes it |
| 4 | Task not found by the system |
| 5 | Shows the application with its recent size. The application is still active |

Return Value:

| | |
|---|--|
| 0 | The operation was not correctly executed |
| 1 | The operation was successfully executed |

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| Status[1] | WinExec("write.exe mytext.wri") // edits the text file mytext.wri WinExec("\\INDUSOFT\\BIN\\pserver.exe myprint.txt") // prints the in disk text file myprint.txt |
| Status[2] | WinExec("\\INDUSOFT\\BIN\\logon.exe /OFF") // deactivates the LOGON utility of <i>Studio</i> |

SendKeyObject (numEvent, strMainKey, numShift, numCtrl, numAlt, strTargetScreen)

Description: It sends keys codes to objects on the opened screens. The "Command" dynamics from the objects can be triggered by this function.

Parameters:

| | |
|-----------------|---|
| numEvent: | code of the key event: |
| 0 | On Down |
| 1 | While Down |
| 2 | |
| strMainKey | tag or string with of the key to be sent to the object |
| numShift | flag, which indicates that the "Shift" key code will be sent too. It is an optional parameter |
| numCtrl | flag, which indicates that the "Ctrl" key code will be sent too. It is an optional parameter |
| numAlt | flag, which indicates that the "Alt" key code will be sent too. It is an optional parameter |
| strTargetScreen | specifies the screen, which will receive the keys code |

Return Value:

| | |
|----|-------------------------|
| -2 | memory allocation error |
| -1 | Viewer is not running |
| 0 | Invalid parameter(s) |
| 1 | Success |

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Status | SendKeyObject(0, "F1") // Sends the "F1" key code Event = "On Down" |
| Status | SendKeyObject(2, "V", 1, 0, 0) // Sends the "Shift+V" key code Event = "On Up" |

⇒ The "numEvent" argument defines if the function will execute the expressions configured in the "On Down", "On While" or "On Up" of the objects "Command" dynamic. It requires special attention to the "On While" event. Each time that the "SendKeyObject" function is executed, it executes the expressions configured in the "On While" sheet (from the objects "Command" dynamic) just at once.

⇒ The "strMainKey" Parameter can be filled with the following values: "F1" .. "F20", "+", "-", "/", "*", "HOME", "END", "INSERT", "DELETE", "DOWN", "UP", "LEFT", "RIGHT", "PAGEUP", "PAGE-DOWN", "SPACE", "RETURN", "BACKSPACE", "ESCAPE", "A" .. "Z".

⇒ The Parameters "numShift" , "numCtrl" and "numAlt" and "strTargetScreen" are optional. However, if one of them will be used, the other will must to be configured too.

⇒ Note: The "numMainKey" argument is not case sensitive.

SetViewerPos(numLeft, numTop, numOptionalWidth, numOptionalHeight)

Description: Sets the Viewer window position and/or size. This function is especially useful when using dual monitor feature from operating system.

Parameters:

| | |
|---|---|
| numLeft | Horizontal coordinate, in pixels, of the left border from the Viewer window |
| numTop | Vertical coordinate, in pixels, of the top border from the Viewer window |
| numOptionalWidth (Optional parameter) | Screen window width, in pixels |
| numOptionalHeight (Optional parameter) | Screen window height, in pixels |

⇒ **Note:** When the optional Parameters (numOptionalWidth and numOptionalHeight) are omitted, the Viewer window will get the size from the application resolution.

Return Value:

| | |
|---|---------|
| 0 | Error |
| 1 | Success |

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| ErrorCode | SetViewerPos (TagLeft, TagRight, TagWidth, TagHeight) |
| ErrorCode | SetViewerPos (0, 0, 800, 600) |
| ErrorCode | SetViewerPos (0, 0) |

StartTask(strTaskName)

Description: Call this function to start an Indusoft Task

Parameters:

| | | |
|-------------|---|------------------|
| strTaskName | String or string tag with the name of the task to be started. It may be one of the following tasks: | |
| | BGTASK | Background Tasks |
| | VIEWER | Viewer |
| | DBSPY | Database Spy |
| | LOGWIN | Logwin |
| | DRIVER | Driver |
| | UNIDDECL | DDE client |
| | UNINDDE | DDE server |
| | UNIODBC | ODBC |
| | TCPSERVER | TCP/IP Server |
| | TCPCLIENT | TCP/IP Client |
| | OPCCLIENT | OPC |

Examples:

```
StartTask ("BGTASK")
StartTask ("VIEWER")
```

IsTaskRunning(strTaskName)

Description: Call this function to verify if an Indusoft Task is running

Parameter:

| | |
|-------------|---|
| strTaskName | String or string tag with the name of the task to be started It may be one of the tasks used in StartTask function |
|-------------|---|

Return Value:

| | |
|---|---------|
| 0 | Error |
| 1 | Success |

Examples:

```
IsTaskRunning ("BGTASK")
IsTaskRunning ("VIEWER")
```

EndTask(strTaskName)

Description: Call this function to stop an Indusoft Task

Parameter:

| | |
|-------------|---|
| strTaskName | String or string tag with the name of the task to be stopped It may be one of the tasks used in StartTask function |
|-------------|---|

Examples:

```
EndTask ("BGTASK")
EndTask ("VIEWER")
EndTask ("DRIVER")
```

3.7.12 File Manipulation Functions**FileCopy(arg1, arg2)**

Description: Copies the file *arg1* to *arg2*.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|----------------------------------|
| Status | FileCopy("file.txt", "file.bak") |

FileDelete (arg)

Description: Deletes the file expressed on *arg*.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|------------------------|
| Status | FileDelete("file.txt") |

FileLength (filename)

Description: Returns the file size indicated on *arg*.

Return Value:

| | |
|---|------|
| 0 | Fail |
|---|------|

Otherwise, returns the size of the file (float).

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|------------------------|
| Length | FindLength("Text.txt") |
| Filename | "Text.txt" |
| Length | FindLength(filename) |

FileRename (arg1, arg2)

Description: Renames the file *arg1* with a new name expressed on *arg2*.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------------------|
| Status | FileRename("file.txt","new_file.txt") |

FindFile (arg)

Description: Verifies that the indicated *arg* files exist.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| n_files | FindFile("*.hst") |

PRINT(arg)

Description: Prints an ASCII file.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| Status | PRINT("file.txt") |

RDFilen ("filename", "path", "mask", "ChangeDir")

Description: Returns a user-selected filename.

Parameter:

| | |
|----------|---|
| Filename | Tag of the string type This tag receives the filename the user chooses |
|----------|---|

⇒ **Note:** There is a difference between: *filename* and *Filename* tags. The first is *filename* tag of the string type that will receive the name of the opened file. The second is *Filename* tag of the string type that will contain a valid tag name (also of the string type) to receive the name of the chosen file.

| | |
|-----------|---|
| Path | Path (including subdirectories) for search and file selection |
| Mask | Tag or string that contains the mask (options or filters) of the search It can contain the characters "*" and "?" for generic searches |
| ChangeDir | Tag or value of the integer type. If 0 does not allow change directory, of other value does |

Return Value:

| | |
|---|--|
| 0 | Success |
| 1 | One of the parameters is not a string |
| 2 | Parameter 1 contains an invalid tag name |
| 3 | The user canceled the operation Must be 3 |

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Status | RDFilen (filename, "C:\InduSoft", "*.txt") |
| Path | "C:\INDUSOFT" |
| Masc | "*.txt" |
| Status | RDFilen (filename, path, masc) |

GetFileAttributes("strFileName")

Description: Returns attributes for a specified file.

The attributes can be one or more of the following values (in hexadecimal):

| | | |
|-----------|---|--------------------|
| error | = | -1 (decimal) |
| READONLY | = | 0x00000001 [bit 0] |
| HIDDEN | = | 0x00000002 [bit 1] |
| SYSTEM | = | 0x00000004 [bit 2] |
| DIRECTORY | = | 0x00000010 [bit 4] |
| ARCHIVE | = | 0x00000020 [bit 5] |
| NORMAL | = | 0x00000080 [bit 7] |
| TEMPORARY | = | 0x00000100 [bit 8] |

GetFileTime("strFileName" , strNumber)

Description: Returns a string with date and/or time of a file.

| | |
|------------|---|
| StrNumber: | identifies the return of the function: |
| 0 | returns the date and time from the file |
| 1 | returns only the file date |
| 2 | returns only the file time |

GetLine(strFileName , strSeqChar, "strStoreTag", numOptCase, "numOptOverflowTag")

Description: Searches a sequence of characters (string) in a ASCII file and stores (in a string tag) the contents of the whole line where the sequence of characters has been found. The function searches just the first occurrence of the string in the ASCII file.

Parameters:

| | |
|---|--|
| strFileName | string or tag string with the path and name of the ASCII file where the sequence of chars is going to be searched |
| strSeqChar | string or tag string with the sequence of chars to be found in the ASCII file |
| strStoreTag | name of the tag (string type), which will receive the contents of the whole line, where the sequence of chars has been found. If the string is not found in the file, this tag will receive null value |
| numOptCase (Optional parameter) | numerical value or numerical tag with case sensitive settings (0 = No case sensitive [Default]; 1 = Case sensitive) |
| numOptOverflowTag (Optional parameter) | name of the tag (integer type), which will receive the result of overflow verification - line has more than 255 chars (0 = OK, 1 = Overflow) This flag tag checks just the first occurrence of the string in the ASCII file |

Return Value:

| | |
|-----|--|
| -7 | Invalid Number of parameters (This function should have more than 2 parameters and less than 6) |
| -6 | Invalid numOptOverflowTag parameter |
| -5 | Invalid numOptCase parameter |
| -4 | Invalid strStoreTag parameter |
| -3 | Invalid strSeqChar parameter |
| -2 | Invalid strFileName parameter |
| -1 | ASCII File has not been found |
| 0 | String has not been found in the target ASCII file |
| <N> | Amount of lines where the "sequence of characters" has been found from the target ASCII file |

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| ErrorCode | GetLine ("c:\Settings.txt", "Studio", "TagLine") |
| ErrorCode | GetLine (TagPath, TagSeqChr, "TagLine2") |
| ErrorCode | GetLine ("c:\Settings.txt", TagSeqChr, "TagLine2", 0) |
| ErrorCode | GetLine (TagPath, "Studio", "T+C52agLine2", 0, "TagOverflow") |

3.7.13 Functions for Graphics Screens Printing

PrintWindow(arg)

Description: Prints any application screen (.SCR extension). This screen can be utilized and opened by the Viewer or not. If it is in a disk screen, it will be loaded to the memory, actualized with the tag's values and curves, and printed. This operation does not interfere with the screens in use in the Viewer.

Parameters: Tag or value with the screen name to be printed (with or without extension).

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------|
| Status | PrintWindow("screen.scr") |
| Status | PrintWindow("screen") |

⇒ You can use the **PrintWindow()** function to print reports in graphical format.

3.7.14 Functions for Text Translations

EXT(arg)

Description: Function for text translation in the application.

Parameter: arg is a tag or value of the string type that contains the text to be translated.

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Output | EXT("Text") // output is a string tag that will receive the translation of "Text", according to the translation file |

SetTranslationFile(filename)

Description: Function for files translation. The system uses the translation file and changes all objects with text outputs of the application for its translation.

Parameter:

| | |
|-----------------|--|
| <i>filename</i> | Tag or value of the string type that contains the translation filename |
|-----------------|--|



CAUTION

You must have a translation file in the translation file utility.

Return Value:

| | |
|---|---|
| 0 | Success |
| 1 | Invalid number of parameters |
| 2 | Wrong parameter type |
| 3 | Translation file could not be opened or localized |

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------------------|
| Status | SetTranslationFile ("trans1.TRA") |
| Filename | "trans1.TRA" |
| Status | SetTranslationFile (filename) |

3.7.15 Multimedia Functions**Play(arg)**

Description: Plays the .WAV file passed as a parameter.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| StatusPlay | ("songs.wav") |

3.7.16 System Information**DbVersion ()**

Description: Returns the database version number of the current application.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------------------|
| Version_db | DbVersion () // version_db = 173 |

GetAppHorizontalResolution()

Description: Function gets the value from the .app file, section [Info]. Returns the value on [Info], it does not test the Windows configuration.

Example:

```
[Info]
AppResolution=640 480
```


GetAppVerticalResolution()

Description: Function gets the value from the .app file, section [Info]. Returns the value on [Info], it does not test the Windows configuration.

Example:

```
[Info]
AppResolution=640 480
```

GetComputerName()

Description: Returns the local computer name.

GetHardkeyModel()

Description: Returns the name of your hardkey model.

Return Value: Returns a string with the hardkey model name:
 None Hardkey not installed or not found
 Otherwise, returns a string with the hardkey model name

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Hardkey_mod | GetHardkeyModel () // hardkey_mod = InduSoft Full Version |

**CAUTION**

For the correct execution of this function, you must do the hardkey installation first.

GetHardkeySN ()

Description: Returns the serial number of the hardkey.

Return Value: Returns a string with the serial number of the hardkey:
 0 Hardkey not installed or not found
 Otherwise, returns a string with the hardkey serial number

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| Hardkey_num | GetHardkeysn() // hardkey_num= 120.745 |

**CAUTION**

For the correct execution of this function, you must do the hardkey installation first.

GetProductPath()

Description: Returns the path to the *Studio* directory.

GetOS()

Description: Returns operating system:

| | |
|---|--------------|
| 0 | Windows 3.1x |
| 1 | Windows 95 |
| 2 | Windows NT |

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------------|
| Os_version | GetOS () // Os_version = 2 |

GetPrivateProfileString

Description: Reads .ini files.

GetTickCount()

Description: Returns the current value of the clock ticks counter.

Parameters: None

Return Value: Integer with the milliseconds counted by the clock for each initialization of the operational system.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| num_of_ms | GetClickCount() // num_of_ms will receive the // counted milliseconds since the computer initialization |

InfoAppAlrDir()

Description: Returns the alarm directory of the current application.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Alr_hst_path | InfoAppAlrDir() // alr_hst_path = "D:\INDUSOFT\TEST\alarm\" |

InfoAppDir()

Description: Returns the application's current directory.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Curr_appl | InfoAppDir () // curr_appl = "D:\INDUSOFT\TEST" |

InfoAppHstDir()

Description: Returns the data historic directory of the current application.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| Hst_path | InfoAppHstDir () // hst_path = "D:\INDUSOFT\TEST\hst\" |

InfoDiskFree (arg)

Description: Returns disposable free space in the disk.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|--|
| free_bytes | InfoDiskFree ("C") // free_bytes = 60604416.000000 |

InfoResources (arg)

Description: Returns the Window's disposable resources.

Not for NT: On NT, only valid parameter with value "3" (available memory). Others are not used.

Parameters: *arg1* is an integer from 0 to 3:

| | |
|---|--------------------|
| 0 | System functions |
| 1 | GDI functions |
| 2 | USER functions |
| 3 | Memory. Must be 3. |

Examples:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| System | InfoResources (0) // system = 76.000000 % |
| GDI | InfoResources (1) // GDI = 76.000000 % |
| USER | InfoResources (2) // USER = 80.000000 % |
| Memory | InfoResources (3) // memory = 16150528.000000 bytes |

NoInputTime()

Description: Returns the time from the last keyboard action.

Return Value: Integer.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-------------------|
| Number | NoInputTime () |

ProductVersion()

Description: Returns the *Studio* number version.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---|
| Version | ProductVersion() // version = 1.130000 |

SetAppAlarmPath

Description: Set current alarm application path.

SetAppHstPath

Description: Set current hst application path.

SetDateFormat

Description: Sets the separator and date format (DMY, DYM, MDY, MYD, YDM, or YMD).

Return Value: Error codes:

| | |
|---|-------------------|
| 0 | No error |
| 1 | Invalid parameter |

3.7.17 Database Access Functions

ForceTagChange(arg1, arg2)

Description: In some cases, you may want to write a value in a tag, forcing the system to act as if it were a new value. This function forces the value *arg2* on the tag *arg1*.

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|---------------------------------|
| | ForceTagChange("tagcount", 100) |

3.7.18 Loops

FOR(first_value, last_value, step)

Description: Implements execution steps.

Parameters:

| | |
|-------------|--|
| first_value | Tag, numerical value, or expression with the first step of the variable) |
| last_value | Tag, numerical value, or expression with the last step for the variable value |
| step | Tag, numerical value, or expression with the incremental step for the variable value |

Return Value: Numerical value

Example:

| <u>Tag Name</u> | <u>Expression</u> |
|-----------------|-----------------------------|
| J | FOR (1, tag_test, 1) |
| Temperat[j] | count / j |

Next

NEXT

Description: An internal tag that points to the next increment.

⇒ **Note:** Any FOR function must have its corresponding NEXT function.

3.7.19 ODBC Functions

ODBCOpen(DSN, User, Password, TableName, Filter, Sort)

Description: Use this function to open a connection to a database. This function returns a handler that should be used in subsequent calls to the ODBC functions. After calling this function, no register has been read from the database yet. You need to bind the columns and call the ODBCQuery function to retrieve the first register.

Parameters: The ODBCOpen function accepts the following arguments:

| | |
|-----------|------------------------------|
| DSN | Data Source Name (string) |
| User | User name (string) |
| Password | Password (string) |
| TableName | Database table name (string) |
| Filter | SQL WHERE clause (string) |
| Sort | SQL ORDER BY clause (string) |

Returns: On success, this function returns a handler greater than zero that identifies the database. Each database opened by this function receives a different handler.

On fail, this function returns:

| | |
|----|--|
| -1 | Invalid parameter Every parameter should be string. |
| -2 | DSN or TableName contain an empty string |

⇒ **Note:** This function doesn't open the database itself. It just creates a handle to manipulate the database. To open the database, you need to bind the columns, and call the function ODBCQuery.

ODBCClose(Handler)

Description: Close a connection to a database.

Parameters: The ODBCclose function accepts the following argument:

| | |
|---------|---|
| Handler | Handler returned by the ODBCOpen function (integer) |
|---------|---|

Returns:

| | |
|-------------|-------------------------|
| On success, | this function returns 0 |
| On fail, | this function returns: |
| | 1 Invalid handler |

ODBCBindCol(Handler, ColName, ColType, TagName)

Description: Binds a column to a tag. Every time you finish making the columns binding, you need to call the ODBCQuery function.

Parameters: The ODBCBindCol function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

ColName Database column name (string)

ColType SQL data type (string). It may be one of the following types:

SQL_BIT

SQL_TINYINT

SQL_LONGVARCHAR

SQL_CHAR

SQL_VARCHAR

SQL_DECIMAL

SQL_NUMERIC

SQL_DATE

SQL_TIME

SQL_TIMESTAMP

SQL_DOUBLE

SQL_REAL

SQL_SMALLINT

SQL_INTEGER

TagName Name of the tag bound to the column (string).

Returns:

On success, this function returns 0

On fail, this function returns:

1 Invalid handler

2 Invalid parameter type

3 One of the parameters has an empty string

4 ColType contains an invalid type

ODBCUnbindCol(Handler, ColName)

Description: Unbinds a column from a tag.

Parameters: The ODBCUnbindCol function accepts the following arguments:

| | |
|---------|---|
| Handler | Handler returned by the ODBCOpen function (integer) |
| ColName | Database column name (string) |

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Invalid parameter type
- 3 Column not bound

ODBCSetFilter(Handler, Filter)

Description: Use this function to constrain or filter the records InduSoft selects. This is useful for selecting a subset of records, such as "all salespersons based in California" ("state = 'CA'"). Remember to call ODBCQuery after calling this function.

Parameters: The ODBCSetFilter function accepts the following arguments:

| | |
|---------|---|
| Handler | Handler returned by the ODBCOpen function (integer) |
| Filter | SQL WHERE clause (string) |

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Invalid parameter type

ODBCSetSort(Handler, Filter)

Description: Use this function to sort the records InduSoft selects. You can use this feature to sort the records on one or more columns. Remember to call ODBCQuery after calling this function.

Parameters: The ODBCSetSort function accepts the following arguments:

| | |
|---------|---|
| Handler | Handler returned by the ODBCOpen function (integer) |
| Sort | SQL ORDER BY clause (string) |

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Invalid parameter type

ODBCQuery(Handler)

Description: Use this function after opening and binding the columns to retrieve the first register. If you modify the column binding, or modify the filter and sort, you need to call this function again.

Parameters: The ODBCQuery function accepts the following arguments:

| | |
|---------|---|
| Handler | Handler returned by the ODBCOpen function (integer) |
|---------|---|

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 No columns bound
- 3 Couldn't open database
- 4 Can't restart database
- 5 Query error

ODBCInsert(Handler)

Description: Use this function to insert a new register to the database. InduSoft will use the values of the tags bound by the ODBCBindCol function to create the new register.

Parameters: The ODBCInsert function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Database not open
- 3 Insert error

ODBCDelete(Handler)

Description: Use this function to delete the current register. After a successful deletion, you need to explicitly call one of the Move functions in order to move off the deleted record.

Parameters: The ODBCDelete function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Database not open
- 3 Delete error

ODBCUpdate(Handler)

Description: Use this function to update the current register. InduSoft will use the values of the tags bound by the ODBCBindCol function to update the current register.

Parameters: The ODBCUpdate function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Database not open
- 3 Update error

ODBCMove(Handler, Offset)

Description: Call this function to move the current record pointer within the record set, either forward or backward. If you pass a value of 0 for Offset, ODBCMove refreshes the current record.

Parameters: The ODBCMove function accepts the following arguments:

| | |
|---------|---|
| Handler | Handler returned by the ODBCOpen function (integer) |
| Offset | The number of rows to move forward or backward Positive values move forward, toward the end of the record set Negative values move backward, toward the beginning (integer) |

Returns:

| | |
|-------------|-------------------------|
| On success, | this function returns 0 |
| On fail, | this function returns: |
| | 1 Invalid handler |
| | 2 Database not open |
| | 3 Move error |

ODBCMoveFirst(Handler)

Description: Call this function to move to the first record within the record set.

Parameters: The ODBCMoveFirst function accepts the following arguments:

| | |
|---------|---|
| Handler | Handler returned by the ODBCOpen function (integer) |
|---------|---|

Returns:

| | |
|-------------|-------------------------|
| On success, | this function returns 0 |
| On fail, | this function returns: |
| | 1 Invalid handler |
| | 2 Database not open |
| | 3 Move error |

ODBCMoveLast(Handler)

Description: Call this function to move to the last record within the record set.

Parameters: The ODBCMoveLast function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Database not open
- 3 Move error

ODBCMoveNext(Handler)

Description: Call this function to move to the next record within the record set.

Parameters: The ODBCMoveNext function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Database not open
- 3 End of record set reached
- 4 Move error

ODBCMovePrev(Handler)

Description: Call this function to move to the next record within the record set.

Parameters: The ODBCMovePrev function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Database not open
- 3 Begin of record set reached
- 4 Move error

ODBCCanAppend(Handler)

Description: Call this function to determine whether the database allows you to add new records.

Parameters: The ODBCCanAppend function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns: Nonzero if the database allows adding new records; otherwise 0.

ODBCCanTransact(Handler)

Description: Call this function to determine whether the database allows transactions

Parameters: The ODBCCanTransact function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns: Nonzero if the database allows transactions; otherwise 0.

ODBCCanUpdate(Handler)

Description: Call this function to determine whether the database can be updated.

Parameters: The ODBCCanUpdate function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns: Nonzero if the database can be updated; otherwise 0.

ODBCIsBOF(Handler)

Description: Call this function before you scroll from record to record to learn whether you have gone before the first record of the record set. You can also use ODBCIsBOF along with ODBCIsEOF to determine whether the record set contains any records or is empty. Immediately after you call ODBCQuery, if the record set contains no records, ODBCIsBOF returns nonzero. When you open a record set that has at least one record, the first record is the current record and ODBCIsBOF returns 0.

If the first record is the current record and you call ODBCMovePrev, ODBCIsBOF will subsequently return nonzero.

Parameters: The ODBCIsBOF function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns: Nonzero if the record set contains no records or if you have scrolled backward before the first record; otherwise 0.

ODBCIsEOF(Handler)

Description: Call this function as you scroll from record to record to learn whether you have gone beyond the last record of the record set. You can also use ODBCIsEOF to determine whether the record set contains any records or is empty. Immediately after you call ODBCQuery, if the record set contains no records, ODBCIsEOF returns nonzero. When you open a record set that has at least one record, the first record is the current record and ODBCIsEOF returns 0.

If the last record is the current record when you call ODBCMoveNext, ODBCIsEOF will subsequently return nonzero.

Parameters: The ODBCIsEOF function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns: Nonzero if the record set contains no records or if you have scrolled beyond the last record; otherwise 0.

ODBCIsDeleted(Handler)

Description: Call this function to determine whether the current record has been deleted. If you scroll to a record and ODBCIsDeleted returns nonzero, then you must scroll to another record before you can perform any other operations.

Parameters: The ODBCIsDeleted function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns: Nonzero if the record set is positioned on a deleted record; otherwise 0.

ODBCBeginTrans(Handler)

Description: Call this function to begin a transaction with the connected data source.

Parameters: The ODBCBeginTrans function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns:

On success, this function returns 0

On fail, this function returns:

1 Invalid handler

2 Database not open

3 Error beginning transaction

ODBCCommitTrans(Handler)

Description: Call this function upon completing transactions.

Parameters: The ODBCCommitTrans function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns:

On success, this function returns 0

On fail, this function returns:

1 Invalid handler

2 Database not open

3 Error committing transaction

ODBCRollback(Handler)

Description: Call this function to reverse the changes made during a transaction

Parameters: The ODBCRollback function accepts the following arguments:

Handler Handler returned by the ODBCOpen function (integer)

Returns:

On success, this function returns 0

On fail, this function returns:

1 Invalid handler

2 Database not open

3 Error rolling back transaction

ODBCExecuteSQL(Handler, SQLCommand)

Description: Call this function when you need to execute an SQL command directly. ODBCExecuteSQL does not return data records.

Parameters: The ODBCExecuteSQL function accepts the following arguments:

| | |
|------------|---|
| Handler | Handler returned by the ODBCOpen function (integer) |
| SQLCommand | A valid SQL command (string) |

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Database not open
- 3 Invalid parameter
- 4 Error executing SQL command

ODBCIsFieldNull(Handler, ColName)

Description: Call this function to determine whether the specified field of a record set has been flagged as Null.

Parameters: The ODBCIsFieldNull function accepts the following arguments:

| | |
|---------|---|
| Handler | Handler returned by the ODBCOpen function (integer) |
| ColName | Column name (string) |

Returns: Nonzero if the specified field is flagged as Null; otherwise 0.

ODBCIsFieldNullable(Handler, ColName)

Description: Call this function to determine whether the specified field is "null able" (can be set to a Null value).

Parameters: The ODBCIsFieldNullable function accepts the following arguments:

| | |
|---------|---|
| Handler | Handler returned by the ODBCOpen function (integer) |
| ColName | Column name (string) |

Returns: Nonzero if the specified field is flagged as Null; otherwise 0.

ODBCSetFieldNull(Handler, ColName, Value)

Description: Call this member function to flag a field data member of the record set as Null (specifically having no value) or as non-Null.

Parameters: The ODBCIsFieldNullable function accepts the following argument:

| | |
|---------|--|
| Handler | Handler returned by the ODBCOpen function (integer) |
| ColName | Column name (string) |
| Value | Nonzero if the field data member is to be flagged as having no value (Null). Otherwise 0 if the field data member is to be flagged as non-Null |

Returns:

On success, this function returns 0

On fail, this function returns:

- 1 Invalid handler
- 2 Database not open
- 3 Invalid parameter
- 4 Invalid column name

3.7.20 MAIL Functions

CnfEmail (strSmtp, strFrom, strPOP3, strUser, strPassword, numOptionalTimeOut)

Description: Set SMTP parameters. This function must be executed to configure these parameters before sending emails with the SendEmail() function.

Parameters:

| | |
|---------------------|--|
| strSMTP | String or string tag with the SMTP (Simple Mail Transfer Protocol) server name or with the SMTP server IP Address. For CEView application it's JUST allowed to use the SMTP IP Address |
| strFrom | String or one string tag with the sender address |
| strPOP3 | POP3 name from the sender |
| strUser | User account name from the sender |
| strPassword | Password for the user account name from the sender |
| numOptional-TimeOut | Timeout limit (in seconds) used when sending messages. This parameter is optional. When it's not configured, the default timeout from operating system is used – recommended. |

Return Values:

| | |
|----|---|
| 0 | Success |
| 1 | Invalid format for parameter 1 (strSMTP) |
| 2 | Invalid format for parameter 2 (strFrom) |
| 3 | Invalid format for parameter 3 (strPOP3) |
| 4 | Invalid format for parameter 4 (strUser) |
| 5 | Invalid format for parameter 5 (strPassword) |
| 6 | Invalid format for parameter 6 (numOptionalTimeOut) |
| 7 | Wrong amount of parameters |
| 8 | Error getting host IP Address (invalid POP3 server) |
| 9 | Error Connecting POP3 server |
| 10 | Error sending UserName |
| 11 | Error sending Password |

Examples:

| | |
|----------|--|
| CNFEMail | ("smtp.test.com.br", "factoryaddress@machine.com.br", "pop3.mail.com", "MyUserName", "MyPassword") |
| CNFEMail | (TagString1, TagString2, TagString3, TagString4, TagString5) |

SendEmail(strSubject, strMessage, strTo)

Description: Send e-mail messages. Before executing this function, it's necessary to set some parameters with the CnfEmail() function.

Parameters:

| | |
|------------|---|
| strSubject | String or string tag with the e-mail subject |
| strMessage | String or string tag with the e-mail message (up to 255 characters) |
| strTO | String or string tag with recipient address (target) |

Return Values:

| | |
|----|---|
| 0 | Success |
| 1 | Invalid format for parameter 1 (strSubject) |
| 2 | Invalid format for parameter 1 (strMessage) |
| 3 | Invalid format for parameter 3 (strTo) |
| 4 | Wrong amount of parameters |
| 5 | Start Socket error |
| 6 | Error getting host IP Address (invalid SMTP server) |
| 7 | Error Connecting SMTP server |
| 8 | Error sending HELO command (initialization) |
| 9 | Error sending MAIL command (sending FROM address) |
| 10 | Error sending RCPT command (sending TO address) |
| 11 | Error sending DATA (sending message) |

Examples:

SendEMail ("Factory 1", "Error to start process", MyAddress@HostName.com)

SendEMail (TagSubject, TagMessage, TagMyAddress)

