Visual Designer v7.0

Quick Start Guide

E·T·N

Powering Business Worldwide

Contents

Contents

Introduction

Visual Designer (or Visual Designer) is a powerful, integrated tool that exploits key features of Microsoft operating systems and enables you to build full-featured SCADA (Supervisory Control and Data Acquisition) or HMI (Human-Machine Interface) programs for your industrial automation business.

This *Visual Designer Quickstart Guide* is intended for individuals using Visual Designer for the first time. This publication will help you quickly familiarize yourself with the basic functions of Visual Designer.

Product Overview

Visual Designer projects run on microcomputers connected in real-time to machines or processors through programmable controllers, remote I/O devices, and other data-acquisition equipment.

These projects consist of animated operator-interface screens, configurable PLC (programmable logic controller) drivers and other controllable I/O devices, a project tags database, and optional modules such as alarm monitors, logic, trend charts, recipes, schedulers, and a security system. Visual Designer projects interface with industrial I/O systems and other Windows applications in the runtime environment using the following protocols:

- ODBC (*Open Database Connectivity*)
- DDE (*Dynamic Data Exchange*)
- NetDDE (*Network Dynamic Data Exchange*)
- OPC (*Open Connectivity*)
- TCP/IP (*Transmission Control Protocol/Internet Protocol*)

After developing a project, you can run it on your development workstation or download the project to a runtime workstation (using a serial or TCP/IP connection) and run it using Visual Designer or CEView runtime software. The workstation processes scan data from connected devices according to parameters defined in the project and then react to, display, store and upload the data.

The product consists of two parts:

- The development system software runs on a desktop, laptop, or industrial PC running a currently supported Microsoft Windows desktop or server operating system.
- The runtime system software runs on an operator interface workstation running a currently supported Microsoft Windows desktop operating system or Windows Embedded.

  Note: The runtime client for Windows Embedded operating systems (CEView) is often pre-loaded on the HMI. If necessary, you can update the CEView version of the development system software by downloading the current version to the HMI.

Product Features

The Visual Designer product provides the following features:

- Integrated Windows development environment with toolbars, dialogs, and menus:
  - Drop-down (pop-up) menus, which you activate by right-clicking on any area of the development environment (Options vary according to context.)
  - Customizable fly-over toolbars
  - Tasks, objects, and controls organized in a tree-view explorer

- Full-featured objects and animations (the ability to modify object properties, execute commands, or inset values to tags used to build screens on the fly at runtime):

  - Configurable objects such as buttons, rectangles, ellipse, polygons, lines, and text

  - Object animations such as bar graphs, color, resizing, position, hide/unhide, rotation, command, hyperlink, and text input/output

  - On-line and historical alarm list displays

  - On-line and historical trending

  - Alignment and distribution tools

  - Background bitmap layer creation and editing

  - Graphics importation

  - ActiveX object containers

- On-line remote management and configuration

- Microsoft DNA architecture compliance, with full OPC and XML support

- Web interface enabled, which exports project screens to a "thin" client through the Internet/intranet and by exchanging data on-line through the TCP/IP protocol

- Symbols library with more than 100 pre-made objects, such as pushbuttons, meters, sliders, switches, text and numeric displays, LED-style indicators, pipes, bumps, icons, vehicles, valves, frames, motors, gauges, and common controls

- Debugging tools:

  - Database Spy window to monitor/force tag values and execute functions

  - LogWin module to record OPC, DDE, and TCP/IP transactions, modules activation, trace tags, and so forth

  - Cross-referencing to locate tags throughout the project

  - On-line system and network diagnostics

- Powerful and flexible tag database (Boolean, Integer, Real, and String tags), array tags, classes, and indirect tag-pointers

- Open architecture with API exchanges and tag values with external software

- Translation editor, which enables you to translate a project into several different languages, and switch between them while the runtime system is online

- TCP/IP client and server modules to exchange tag values and configure redundancy systems

- More than 200 direct communication drivers for different devices (such as PLC) from several manufacturers; such as Allen-Bradley, Siemens, GE-Fanuc, as well as standard protocols such as MODBUS RTU/ASCII, DeviceNet, Profibus, Interbus, and so forth

- Full integration with PC-based control packages (imports tags database) such as ISaGRAF, SteepleChase, Think&Do, OpenControl, FP Control and ASAP.

- OPC Server and 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, UNICODE, and RTF formats) builders integrated into the product

- Event scheduler based on date, time, or data condition (100ms resolution)

- Multi-layer project, which means modular worksheets and screens can be merged easily to other projects

Introduction

- Dial-Up functions to trigger, monitor, and hang-up a dial-up connection with the RAS Server of remote stations
- Functions to send e-mail from Visual Designer (or CEView)
- Real-time project documentation
- Screen resolution converter

Note: Visual Designer provides different product types for each level of project responsibility. However, Visual Designer does not support some features in certain product types (such as CEView). You can review the `TargetVersions.pdf` document on the Visual Designer installation CD for detailed information about these the limitations of each product-type limitations.

Conventions used in this documentation

This documentation uses standardized formatting and terminology to make it easier for all users to understand.

Text conventions

This documentation uses special text formatting to help you quickly identify certain items:

- Titles, labels, new terms, and messages are indicated using **italic** text (for example, **Object Properties**).
- File names, screen text, and text you must enter are indicated using `monospace` text (for example, `D:\Setup.exe`).
- Buttons, menu options, and keyboard keys are indicated using a **bold** typeface (for example, **File** menu).

In addition, this documentation segregates some text into **Tip**, **Note**, and **Caution** boxes:

- **Tips** provide useful information to save development time or to improve the project performance.
- **Notes** provide extra information that may make it easier to understand the nearby text, usually the text just before the note.
- **Cautions** provide information necessary to prevent errors that can cause problems when running the project, and may result in damage.

Mouse and selection conventions

Because most PCs used for project development run a version of Microsoft Windows with a mouse, this documentation assumes you are using a mouse. Generally, a PC mouse is configured for right-handed use, so that the left mouse button is the primary button and the right mouse button is the secondary button.

This documentation uses the following mouse and selection conventions:

- **Click** and **Select** both mean to click once on an item with the left mouse button. In general, you *click* buttons and you *select* from menus and lists.
- **Double-click** means to quickly click twice on an item with the left mouse button.
- **Right-click** means to click once on an item with the right mouse button.
- **Select** also means you should use your pointing device to highlight or specify an item on the computer screen. Selecting an item with a touchscreen is usually the same as selecting with a mouse, except that you use your finger to touch (select) a screen object or section. To select items with your keyboard, you typically use the Tab key to move around options, the Enter key to open menus, and the Alt key with a letter key to select an object that has an underlined letter.
- **Drag** means to press down the appropriate mouse button and move the mouse before releasing the button. Usually an outline of the item will move with the mouse cursor.

Windows conventions

This documentation uses the following Windows conventions:

- **dialoges** (or *dialogs*) are windows that allow you to configure settings and enter information.

- **Text boxes** are areas in dialogs where you can type text.

- **Radio buttons** are white circles in which a black dot appears or disappears when you click on the button. Typically, the dot indicates the option or function is *enabled* (selected). No dot indicates the option or function is *disabled* (not selected).

- **Check boxes** are white squares in which a check (☑ Titlebar) appears or disappears when you click on it with the cursor. Typically, a check ☑ indicates the option or function is *enabled* (selected). No check ☐ indicates the option or function is *disabled* (not selected).

- **Buttons** are icons in boxes appear "pressed" when you click on them.

- **Lists** are panes (white boxes) in windows or dialogs containing two or more selectable options.

- **Combo boxes** have arrows that, when clicked, show part or all of an otherwise concealed list.

- **Dockable windows** are windows that you can drag to an edge of the interface and merge with that edge.

Installation

This section provides instructions for installing, starting, and uninstalling Visual Designer and CEView.

System requirements

These are the minimum system requirements to install and run the Visual Designer software.

Note: The requirements described below are based on typical projects. Depending on your specific project, the requirements may vary:

- "Windows Embedded and Windows Mobile-compatible devices" includes a wide variety of processors and feature sets, from smartphones to industrial displays. Consult your vendor for the specific hardware requirements to run your project on these devices.

- Some of the items listed as optional may be mandatory depending on your project. For instance, if you need to exchange data with a PLC via a serial interface, then the computer must have a serial COM port.

Development

To install and run the development application, you must have:

- A Windows-compatible computer with a standard keyboard, mouse, and SVGA display

- A Windows desktop or server operating system that is currently supported by Microsoft, which at this time includes:

  - Microsoft Windows XP Service Pack 3 or later

  - Microsoft Windows Vista Service Pack 1 or later

  - Microsoft Windows 7, all versions

  - Microsoft Windows Server 2003 Service Pack 2 or later

  - Microsoft Windows Server 2008, all versions

- Microsoft Internet Explorer 6.0 or later

- Minimum of 500MB free hard drive space

- Ethernet adapter or wireless networking

- CD-ROM drive (optional, to install the application; it can also be downloaded from our website)

- USB port (optional, to be used with hard key licensing)

- Serial COM ports and adapters (optional, to be used for direct communication with devices)

Note: Any station that has the development application installed can also run as a project server and/or a project client.

Project Server

To run as a project server, you must have:

- A Windows or Windows Embedded-compatible computer

- A Windows desktop, server, or embedded operating system that is currently supported by Microsoft, which at this time includes:

- Microsoft Windows XP Service Pack 3 or later

- Microsoft Windows Vista Service Pack 1 or later

- Microsoft Windows 7, all versions

- Microsoft Windows Server 2003 Service Pack 2 or later

- Microsoft Windows Server 2008, all versions

- Microsoft Windows XP Embedded Service Pack 3

- Microsoft Windows Embedded Standard 7 (2009)

- Microsoft Windows Embedded Compact (previously known as Windows CE), version 5.0 or later

  Note: We recommend "Professional" and "Ultimate" editions over "Home" and "Media Center" editions, because they include Internet Information Services (IIS) that can be used as your project's Web server.

- Minimum of 500MB free hard drive space

- Ethernet adapter or wireless networking

- USB port (optional, to be used with hard key licensing)

- Serial COM ports and adapters (optional, to be used for direct communication with devices)

Project Client – Embedded

To run as a project client using CEView, you must have:

- A Windows Embedded or Windows Mobile-compatible device with a mouse or touchscreen input

- A Windows embedded or mobile operating system that is currently supported by Microsoft, which at this time includes:

  - Microsoft Windows XP Embedded Service Pack 3

  - Microsoft Windows Embedded Standard 7 (2009)

  - Microsoft Windows Embedded Compact (previously known as Windows CE) or Windows Mobile, version 5.0 or later

- Ethernet adapter or wireless networking

Project Client – Thin

To run as a project client using the Secure Viewer program or the browser-based Thin Client, you must have:

- A Windows or Windows Embedded-compatible computer with a mouse or touchscreen input

- A Windows desktop, server, or embedded operating system that is currently supported by Microsoft, which at this time includes:

  - Microsoft Windows XP Service Pack 3 or later

  - Microsoft Windows Vista Service Pack 1 or later

  - Microsoft Windows 7, all versions

  - Microsoft Windows Server 2003 Service Pack 2 or later

  - Microsoft Windows Server 2008, all versions

  - Microsoft Windows XP Embedded Service Pack 3

  - Microsoft Windows Embedded Standard 7 (2009)

- • Microsoft Windows Embedded Compact (previously known as Windows CE) or Windows Mobile, version 5.0 or later

- • Microsoft Internet Explorer 6.0 or later

- • Ethernet adapter or wireless networking

Installing the Software

Visual Designer provides development tools for all Visual Designer projects, and it can be installed on a PC running Microsoft Windows XP, Windows Vista, or Windows 7 operating system. For more information, see System Requirements.

You can install the development application either from the web download or from the Visual Designer installation CD. For projects running on Windows Embedded target systems, you can use the development application to download CEView (the runtime engine) to the target system via serial or TCP/IP link.

The Visual Designer installation program creates directories as needed, copies files to your hard drive, and creates the Visual Designer icon on your Windows desktop.

Note:

- • You must have Administrator privileges on your PC in order to install or uninstall the development application.

- • You must uninstall an older version of the development application (or move it to a different directory) before installing a new version. Also, you cannot install the same version of the development application in two different paths on the same PC.

The instructions for installing Visual Designer and CEView are provided inthe following two sections.

Installing the Development Application on Your Windows PC

To install the Visual Designer development application from the installation CD:

1. Turn on your PC and be sure that no other programs are running.

2. Insert the installation CD into your PC's CD-ROM drive.

   Internet Explorer should run automatically and show the CD's welcome screen. If it does not — for example, if you have the Autorun feature turned off in your Windows settings — then you can manually show the screen by using Windows Explorer to locate and open the file D:\Eaton.htm.

3. In the welcome screen, select the product that you want to install.

   Internet Explorer will ask if you want to run or save the installer.

4. Click **Run**.

   The product's installation wizard will begin.

5. Follow the wizard's instructions to proceed with the installation.

6. When the installation is finished, select **Yes, I want to restart my computer now** and then click **OK**.

After your PC has restarted, you can run the development application. See Starting the Software for instructions.

Note: When you install the development application, Microsoft .NET Framework 2.0 and some other utilities are also installed to support the features of Visual Designer. Your PC may have later versions of the .NET Framework already installed, but there is no reason for concern because multiple versions of the .NET Framework should not conflict with each other. You can see which versions are installed on your PC by

opening the **Add or Remove Programs** control panel (**Start** > **Control Panel** > **Add or Remove Programs**).

For more information about Microsoft .NET Framework, see Database Appendix A: Using ODBC Databases.

Installing CEView on Your Windows Embedded Device

CEView is the runtime engine for Visual Designer projects on Windows Embedded devices. CEView must be installed on your device *before* you send your project to it.

Where the Files Are Located

Given the nature of Windows Embedded devices, each combination of OS version and device processor has its own build of CEView. All of these builds are located in the following directory:

```
[…]\EATON\Visual Designer\Redist\
```

The build for your specific device is located in the following directory:

```
[…]\EATON\Visual Designer\Redist\version\processor\
```

…where:

- `version` is the version of the operating system on the device where CEView will be installed:
  - The`\WinCE 5.0` folder contains the files for Windows CE and Windows Mobile 5.0 or later; and
  - The`\WinEmbedded` folder contains the files for Windows XP Embedded and Windows Embedded 7.

- `processor` is the processor used by your Windows CE device. We provide a CEView runtime for every processor that is currently supported by the Windows CE operating system (e.g., Pocket2003-ArmV4, ArmV4i, x86). For more information, consult the manufacturer's documentation for the device.

To install the files on your device, use the Remote Management tool in the development application.

Installing via TCP/IP (Ethernet)

Note: We recommend using TCP/IP whenever possible.

To install CEView on a Windows Embedded device via a TCP/IP (Ethernet) connection:

1. Make sure your Windows Embedded device is connected to your network.

2. Turn on the device.

   The **Remote Agent** dialog should open automatically. If it does not, then you must manually install the file CEServer.exe on the device. The file is located here:

   ```
   […]\EATON\Visual Designer\Redist\version\processor\CEServer.exe
   ```

   There are different ways to install the file — for example, you can use Microsoft ActiveSync (for Windows 2000 and Windows XP) or Windows Mobile Device Center (for Windows Vista) to communicate directly with the device or you can map the device as a shared folder on your PC. For more information about copying and executing files, consult the manufacturer's documentation for the device.

3. In the **Remote Agent** dialog, click **Setup** and then configure the communication settings for the device's network connection. In particular, note the IP address of the device.

4. Start the development application on your PC.

Installation

5. On the **Home** tab of the ribbon, in the **Remote Management** group, click **Connect**. The **Remote Management** dialog is displayed:
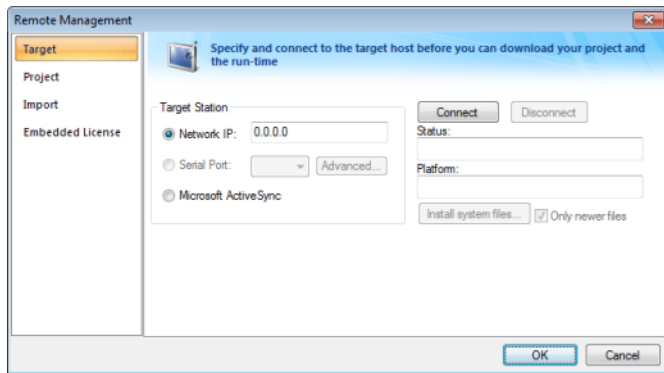
6. In the **Target System** group-box, select **Network IP** and then type the IP address of the device.

7. Click **Connect** to establish a connection between the development application and the device.

   If the conncection is successful, then the device's specifications will be displayed in the **Platform** text-box.

8. Click **Install System Files** to download the CEView files to the device.

9. When the installation is completed, click **Disconnect**.

For more information about downloading and running finished projects on the Windows Embedded device, please see Remote Management.

Installing via Microsoft ActiveSync

To install CEView on a Windows Embedded device via Microsoft ActiveSync:

1. Make sure that Microsoft ActiveSync (for Windows 2000 and Windows XP) or Windows Mobile Device Center (for Windows Vista) is installed on your PC.

2. Turn on the Windows Embedded device and connect it to your PC. Most devices should be able to connect via USB.

3. Start the development application on your PC.

4. On the **Home** tab of the ribbon, in the **Remote Management** group, click **Connect**. The **Remote Management** dialog is displayed.

5. In the **Target System** group-box, select **Microsoft ActiveSync**.

6. Click **Connect** to establish a connection between the development application and the device.

   If the conncection is successful, then the device's specifications will be displayed in the **Platform** text-box.

7. Click **Install System Files** to download the CEView files to the device.

8. When the installation is completed, click **Disconnect**.

Note: In some cases, the Remote Management tool may not be able to connect via Microsoft ActiveSync to a device running Windows CE 6.0 or later. This is because of a problem in the default configuration of Windows CE 6.0. You can fix the problem by using a small utility that is included with Visual Designer. The utility is located at:

```
[…]\EATON\Visual Designer\Redist\ActiveSyncUnlock.exe
```

Copy this file to the device using the **stand-alone version** of Microsoft ActiveSync and then execute the file on the device. It doesn't matter where on the device the

file is located. (For more information about copying and executing files, consult the manufacturer's documentation for the device.) When this is done, try again to use the Remote Management tool to connect to the device.

For more information about downloading and running finished projects on the Windows Embedded device, please see Remote Management.

Starting the Software

To run Visual Designer:

- Double-click the Visual Designer v7.0 icon on the desktop; or

- Choose **Start** > **All Programs** > **EATON** > **Visual Designer v7.0**.

Note: You can run the Visual Designer development environment under any video setting. However, we recommend that you configure the video settings to a resolution of 800x600 (or higher) and use more than 256 colors for a more pleasing environment. The project resolution (screen size) is independent of the operating system resolution.

Uninstalling the Software

⚠ CAUTION

Before starting the uninstall procedure, be sure to back-up any program files you may find useful later. Also, be certain that you have a current (or newer) version of the Visual Designer installation CD or diskettes so you can re-install the software later if necessary.

If you find it necessary to remove Visual Designer from your system, follow these instructions:

1. From the Windows task bar, select **Start** > **Settings** > **Control Panel** to open the Control Panel.

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

3. When the **Add/Remove Programs Properties** dialog displays, select Visual Designer from the list and click **Add/Remove**.

4. When the **Confirm File Deletion** dialog displays, click **Yes**.

   The **Uninstall Shield Wizard** and the **Remove Programs from Your Computer** dialogs display.

5. When the `Uninstall successfully completed` message displays and the **OK** button becomes active, click **OK**.

   Verify that Visual Designer is no longer listed in the **Add/Remove Programs Properties** dialog.

6. Click the **Cancel** button or the close button (), to close the **Add/Remove Programs Properties** dialog, then close the **Control Panel** window.

7. Open the **Windows Explorer** and browse to Visual Designer program directory.

8. Verify that all of the Visual Designer files and folders were deleted. (You must manually delete any that remain.)

Note: The uninstall tool cannot delete files you created or modified in your Visual Designer projects folder.

   You must have administrator privileges to uninstall (and install) Visual Designer.

The Development Environment

Visual Designer incorporates a modern, Ribbon-based Windows interface to provide an integrated and user-friendly development environment.
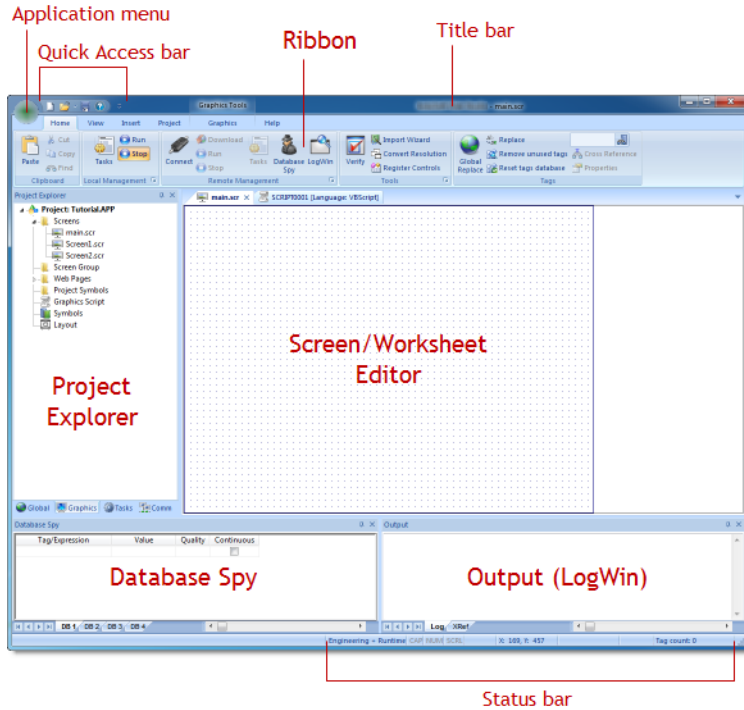
Title Bar

The Title Bar located along the top of the development environment displays the application name (e.g., Visual Designer) followed by the name of the active screen or worksheet (if any).
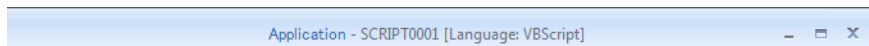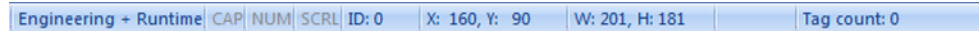

The Title Bar also provides the following buttons (from left to right):

- **Minimize** button : Click to minimize the development environment window to the Taskbar.

- **Restore Down** / **Maximize**: Click to toggle the development environment window between two sizes:

  - **Restore Down** button reduces the window to its original (default) size.

  - **Maximize** button enlarges the window to fill your computer screen.

- **Close** button : Click to save the database and then close the development environment. If you modified any screens or worksheets, the application prompts you to save your work. This button's function is similar to clicking **Exit Application** on the Application menu.

  Note: Closing the development environment does *not* close either the project viewer or the runtime system, if they are running.

Status Bar

The Status Bar located along the bottom of the development environment provides information about the active screen (if any) and the state of the application.


The Status Bar fields (from left to right) are described in the following table:

Field	Description
Execution Mode	The current execution mode of the application.
CAP	Indicates whether the keyboard **Caps Lock** is on (black) or off (grey).
NUM	Indicates whether the keyboard **Num Lock** is on (black) or off (grey).
SCRL	Indicates whether the keyboard **Scroll Lock** is on (black) or off (grey).
Object ID	The ID number of a selected screen object.
Cursor Position	The location of the cursor on the active screen or worksheet. If it's a screen, then the position of the *mouse* cursor is given as X,Y coordinates, where X is the number of pixels from the left edge of the screen and Y is the number of pixels from the top edge of the screen. If it's a worksheet, then the position of the *text* cursor is given as Line and Column.
Object Size	The size (in pixels) of a selected screen object, where W is the width and H is the height.
No DRAG	Indicates whether dragging is disabled (**No DRAG**) or enabled (empty) in the active screen.
Tag Count	The total number of tags used so far in the project.

Application button

The Application button opens a menu of standard Windows application commands like New, Open, Save, Print, and Close.
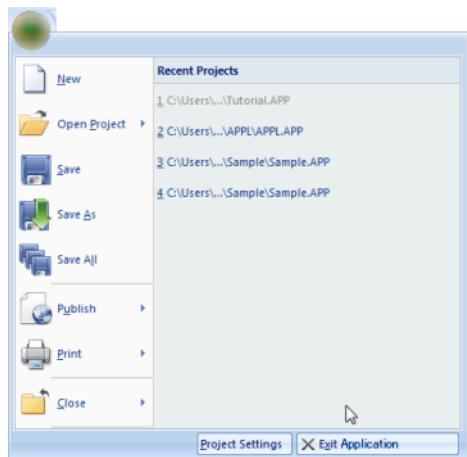
Quick Access Toolbar

The Quick Access Toolbar is a customizable toolbar that contains a set of commands that are independent of the ribbon tab that is currently displayed.

Move the Quick Access Toolbar

The Quick Access Toolbar can be located in one of two places:

- Upper-left corner next to the Application button (default location); or
- Below the ribbon, where it can run the full length of the application window.

If you don't want the Quick Access Toolbar to be displayed in its current location, you can move it to the other location:

1.
    Click **Customize Quick Access Toolbar** .

2. In the list, click **Show Below Ribbon** or **Show Above Ribbon**.

Add a command to the Quick Access Toolbar

You can add a command to the Quick Access Toolbar directly from commands that are displayed on the ribbon:

1. On the ribbon, click the appropriate tab or group to display the command that you want to add to the Quick Access Toolbar.

2. Right-click the command, and then click **Add to Quick Access Toolbar** on the shortcut menu.

You can also add and remove commands — as well as reset the toolbar to its default — using the **Customize** dialog:

1.
    Click **Customize Quick Access Toolbar** .

2. In the list, click **More Commands**. The **Customize** dialog is displayed.
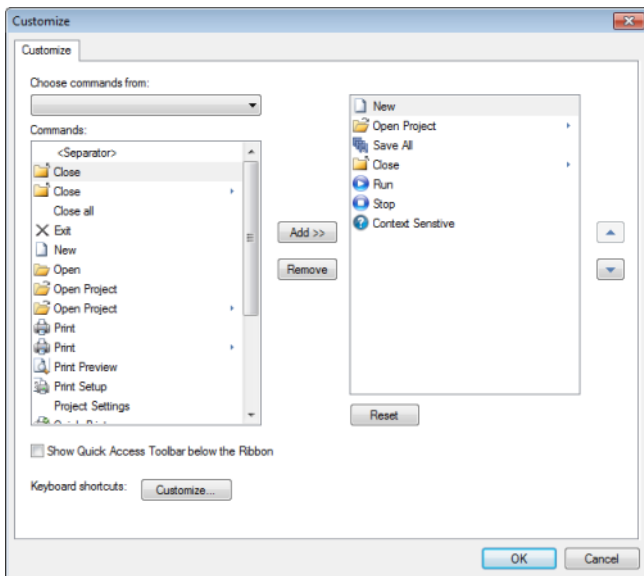
3. In the **Choose commands from** menu, select the appropriate Ribbon tab. The commands from that tab are displayed in the **Commands** list.

4. In the **Commands** list, select the command that you want to add to the Quick Access Toolbar.

5. Click **Add**.

Only commands can be added to the Quick Access Toolbar. The contents of most lists, such as indent and spacing values and individual styles, which also appear on the ribbon, cannot be added to the Quick Access Toolbar.

Ribbon

The new ribbon combines the numerous menus and toolbars from the previous version of Visual Designer into a single, user-friendly interface. Almost all application commands are now on the ribbon, organized into tabs and groups according to general usage.
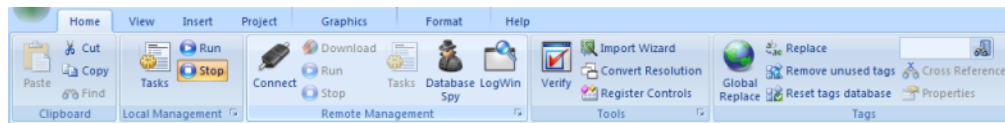
Home tab

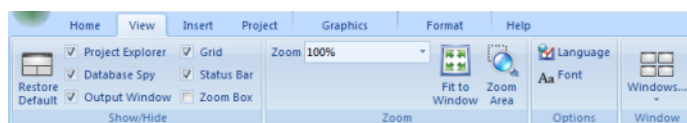The **Home** tab of the ribbon is used to manage your project within the development environment.
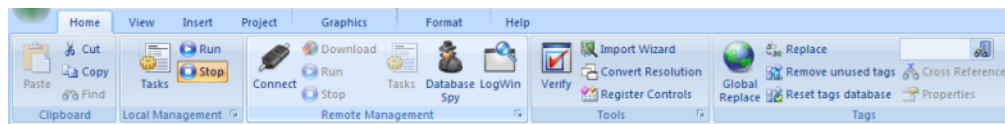
The tools are organized into the following groups:

• **Clipboard**: Cut, copy, paste, and find items in project screens and task worksheets.

• **Local Management**: Run and stop the project on the local station (i.e., where the development application is installed), as well as manage the execution tasks.

• **Remote Management**: Connect to a remote station (e.g., a Windows Embedded device) so that you can download the project to it, and then run, stop, and troubleshoot the project on that station.

• **Tools**: Miscellaneous tools to verify the project, import tags from other projects, convert screen resolutions, and register ActiveX and .NET controls.

• **Tags**: Manipulate tags and tag properties in the project database.

View tab

The **View** tab of the ribbon is used to customize the look of the development environment itself.

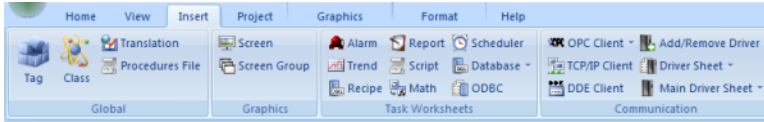The tools are organized into the following groups:

• **Show/Hide**: Show and hide the different parts of the development environment, as well as restore the default layout.

The Development Environment

- **Zoom**: Zoom in and out of the screen editor.
- **Options**: Change the language and font used in the development environment.
- **Window**: Arrange the windows in the development environment.

Insert tab

The **Insert** tab of the ribbon is used to insert new tags, screens, worksheets, and other components into your project.
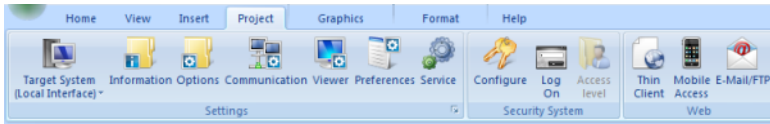

The tools are organized into the following groups:

- **Global**: Insert tags, classes, translations, and procedures into the Global tab of the Project Explorer.
- **Graphics**: Insert screens and screen groups into the Graphics tab of the Project Explorer.
- **Task Worksheets**: Insert task worksheets into the Tasks tab of the Project Explorer.
- **Communication**: Insert server configurations and communication worksheets into the Comm tab of the Project Explorer.

Project tab

The **Project** tab of the ribbon is used to configure your project settings.

The tools are organized into the following groups:

- **Settings**: Configure the general project settings, and also set the project to run as a Windows service.
- **Security System**: Enable and configure the project security system.
- **Web**: Configure the project to accept connections from thin clients and mobile devices, and also configure outgoing email and FTP.

Graphics tab

The **Graphics** tab of the ribbon is used to draw project screens.
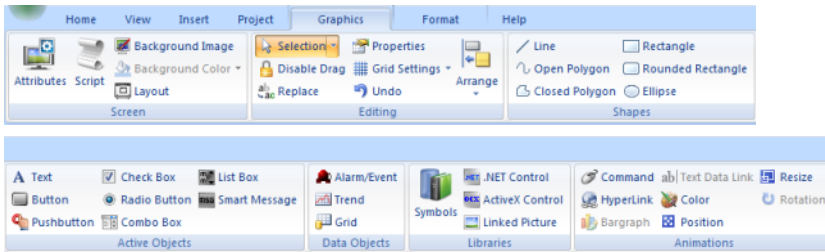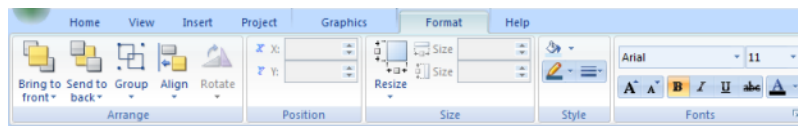

Note: This tab is available only when you have a project screen open for editing.

The tools are organized into the following groups:

- **Screen**: Configure settings for the project screen itself, such as its attributes, script, and background color or image.

- **Editing**: Select and edit objects in the project screen.

- **Shapes**: Draw static lines and shapes.

- **Active Objects**: Draw active objects, like buttons and check boxes.

- **Data Objects**: Draw objects that display historical data, like alarms, events, and trends.

- **Libraries**: Select from libraries of premade objects, such as symbols, .NET and ActiveX controls, and external picture files.

- **Animations**: Apply animations to other screen objects.

Format tab

The **Format** tab of the ribbon is used to format and arrange objects in a project screen.

Note: This tab is available only when you've selected one or more objects in a project screen.

The tools are organized into the following groups:

- **Arrange**: Arrange objects in a project screen, including bring to front and send to back, group, align, and rotate.

- **Position**: Precisely adjust the position of a screen object in a project screen.

- **Size**: Precisely adjust the size of a screen object.

- **Style**: Change the fill and line color of a screen object.

- **Fonts**: Change the caption font of a screen object.

Help tab

The **Help** tab of the ribbon provides additional help with using the software.
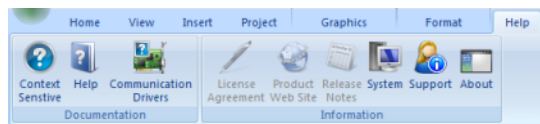
The tools are organized into the following groups:

- **Documentation**: Access the documentation for the development application, including this help file / technical reference and notes for the individual communication drivers.

- **Information**: Access other information about Visual Designer, including the license agreement, product website, and release notes, as well as system and support details that make it easier for Customer Support to assist you.

Project Explorer

The **Project Explorer** organizes all of the screens, worksheets, and other components that make up your project and presents them in an expandable tree-view.
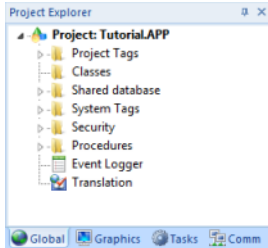
Click the Expand icon ▷ or double-click the folder to view the contents of the folder. Click the Collapse icon ◢ to close the folder.

If you right-click on any component in the **Project Explorer**, a shortcut menu is displayed with options for that component.

Global tab

The **Global** tab of the Project Explorer contains the project tags database, as well as other features that apply to the entire project such security and UI translation.
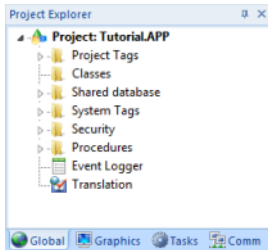
The folders on the **Global** tab are described on the following pages:

- **Project Tags** contains tags you create during project development (such as screen tags or tags that read from/write to field equipment).

- **Classes** contains compound tags, called *class* tags, created to associate a set of values (rather than a single value) with an object.

- **Shared Database** contains tags that were created in a PC-based control program and then imported into the project tags database.

    For example you can import SteepleChase tags into your project so that it can read/write data from a SteepleChase PC-based control product.

- **System Tags** contains predefined tags with predetermined functions that are used by the project for specific, supervisory tasks (for example, *Date* tags hold the current date in string format).

    All system tags are *read-only*, which means you cannot add, edit, or remove these tags from the database.

- **Security** contains all of the group and individual user security accounts configured for the current project.

- **Procedures** contains VBScript functions and sub-routines that can be called by any other script in the project.

- **Event Logger** contains logging and event-retrieval features.

- **Translation** contains the translation worksheet that defines how your project's user interface should be translated into another language.

Graphics tab

The **Graphics** tab of the Project Explorer contains all of the screens, screen groups, and symbols in your project.
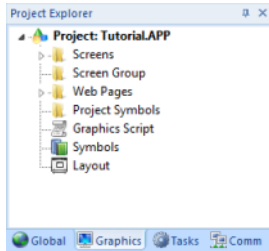
The folders on the **Graphics** tab are described on the following pages:

- **Screens** contains all of the screens created for the current project.

- **Screen Group** contains the entire screen groups (individual screens combined into manageable groups) created for the current project.

- **Web Pages** contains all of the Web pages (i.e., screens saved in HTML format) created for the project.

    - **Mobile Access** allows configuration of the mini-site that is targeted to cell phones, PDAs, and other mobile devices.

- **Project Symbols** contains all of the user-defined symbols, which can be groups of images and/or text. You can create custom symbols for the project and save them into this folder.

- **Graphics Script** contains predefined functions that are executed when certain screen actions occur, such as when the Thin Client is launched on a remote station.

- **Symbols** contains the library of common symbols and graphics provided with the project. Double-click the **Library** icon to open the **Symbol Library**.

- **Layout** displays all screens currently open in the Screen Editor and allows you to visualize how the screens fit together during runtime.

Tasks tab

The **Tasks** tab of the Project Explorer organizes the worksheets that are processed as background tasks during project runtime.
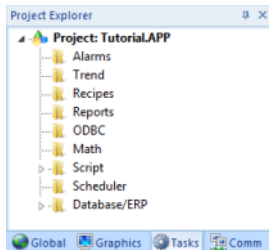
The folders on the Tasks tab are described on the following pages:
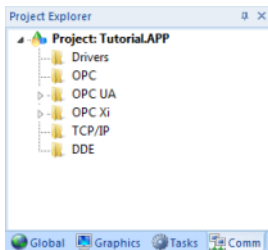
- **Alarms** contains the **Alarm** worksheets used to configure alarm groups and the tags related to each alarm group in the project. You also use the Alarm task to define the alarm messages generated during project runtime.

- **Trend** contains the **Trend** worksheets used to configure history groups that store trend curves for the project. You can use the Trend task to declare which tags must have their values stored on disk, and to create history files for trend graphs. Your project stores the samples in a binary history file (`*.hst`), and displays both history and on-line samples in a trend graph screen.

- **Recipes** contains the **Recipe** worksheets used to configure how data is exchanged between the project database and disk files in ASCII or DBF format, and how values are transferred between files and real-time memory.

- **Reports** contains the **Report** worksheets used to configure reports (text type) that are sent to a printer or a disk. Reports tasks allow you to configure text reports with system data, which makes report creation easier and more efficient.

- **ODBC** contains the **ODBC** worksheets used to configure how the ODBC interface runs in a network environment and uses standard Windows ODBC configuration. You configure ODBC tasks to exchange data between your project and any database supporting the ODBC interface.

- **Math** contains the **Math** worksheets used to configure and implement additional routines to work with different tasks. Your project executes **Math** worksheets as Background Tasks during runtime. You can configure **Math** worksheets to provide free environments for logical routines and mathematical calculations required by the project.

- **Script** contains the Startup Script and other Script Groups.

- **Scheduler** contains the **Scheduler** worksheets used to configure events using defined mathematical expressions, which are executed according to time, date, or other monitored event.

- **Database/ERP** contains the **Database** worksheets that communicate with external databases using the standard ADO.NET interface (as an alternative to ODBC).

Comm tab

The **Comm** tab of the Project Explorer organizes the worksheets that establish communication with another device or software using available protocols.

The folders on the Comm tab are described on the following pages.

- **Drivers** contains the **Driver** worksheets used to configure a communication interface (or interfaces) between the project and remote equipment (such as a PLC or transmitters).

  A *communication driver* is a `.DLL` file that contains specific information about the remote equipment and implements the communication protocol.

- **OPC** contains the **OPC** worksheets used to configure OPC interfaces between your project and an OPC Server. An OPC Client module enables your project to communicate with any device that acts as an OPC Server by implementing the OPC standard described in the *OLE for Process Control Data Access Standard Version 2.0* document published by the OPC Foundation.

- **OPC UA** contains the **OPC UA** worksheets that are used to connect to OPC Servers via the new OPC Unified Architecture protocol.

- **OPC Xi** contains the **OPC Xi** worksheets that are used to connect to OPC Servers via the new OPC Express Interface protocol.

- **TCP/IP** contains the **TCP/IP** worksheets used to configure TCP/IP Client interfaces for other Visual Designer stations.

  Visual Designer TCP/IP Client and Server modules enable two or more projects to keep their databases synchronized using the TCP/IP protocol.

- **DDE** contains the **DDE** worksheets used to configure a DDE Client for a DDE Server application (such as Microsoft Excel or any other Windows program that supports this interface).

  DDE (Dynamic Data Exchange) is a protocol that enables dynamic data exchange between Windows applications. A DDE conversation is an interaction between server and client programs. Visual Designer provides interfaces that run as clients or as servers.

Screen/Worksheet Editor

Use the powerful, object-oriented screen editor to create and edit a variety of screens and worksheets for your projects. You can input information using your mouse and keyboard, output control data to your processes, and automatically update screens based on data input from your processes.
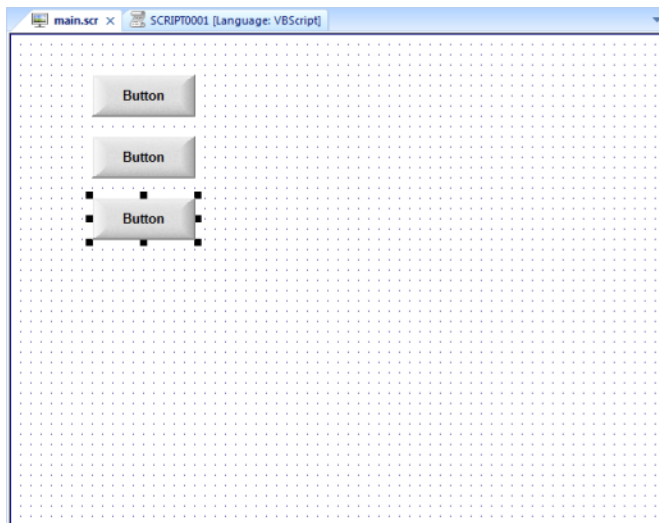
Other screen editor features include:

- Simple point-and-click, drag-and-drop interface
- Grouping objects to preserve the construction steps of individual objects
- Editing objects without having to ungroup internal object components or groups
- Handling bitmap objects and background bitmaps
- Status line support in project windows and dialogs

About Tags and the Project Database

Tags are a core component of any Visual Designer project. Simply put, tags are variables used by Visual Designer to receive and store data obtained from communication with plant floor devices, from the results of calculations and functions, and from user input. In turn, tags can be used to display information on screens (and Web pages), to manipulate screen objects, and to control runtime tasks.

But tags are more than simple variables. Visual Designer includes a real-time database manager that provides a number of sophisticated functions such as time-stamping of any value change, checking tag values against runtime minimum and maximum values, comparing tag values to alarming limits, and so on. A Visual Designer tag has both a value and various properties that can be accessed, some at development and others only at runtime.
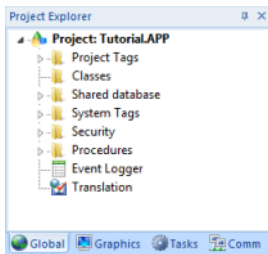
All tags are organized into one of the following categories, which are represented by folders on the **Global** tab of the **Project Explorer**:

- **Project Tags** are tags that you create during project development. Places where project tags are used include:

    - Screen tags

    - Tags that read from/write to field equipment

    - Control tags

    - Auxiliary tags used to perform mathematical calculations

- **Shared Database** tags are created in a PC-based control program and then imported into Visual Designer's tags database.

    For example you might create tags in SteepleChase and import them into Visual Designer so Visual Designer can read/write data from a SteepleChase PC-based control product.

    You cannot modify shared tags within Visual Designer — you must modify the tags in the original PC-based control program, and then re-import them into the Tags database.

- **System Tags** are predefined tags with predetermined functions that are used for Visual Designer supervisory tasks. For example,

    - Date tags hold the current date in string format

    - Time tags hold the current time in string format

    Most system tags are *read-only*, which means you cannot add, edit, or remove these tags from the database.

    To see a list of the system tags, select the **Global** tab in the **Project Explorer**, open the **System Tags** folder, and open the **Tag List** subfolder. The above figure shows a partial list of system tags.

After creating a tag, you can use it anywhere within the project, and you can use the same tag for more than one object or attribute.

Understanding the Tag Name Syntax

Observe the following guidelines when naming a tag:

- Your tag names **must be unique** — you cannot specify the same name for two different tags (or functions). If you type an existing tag name, Visual Designer recognizes that the name exists and will not create the new tag.

- You must begin each tag name with a *letter*. Otherwise, you can use letters, numbers, and the underscore character (_) in your tag name.

- You *cannot* use the following symbols in a tag name:

  ` ~ ! @ # $ % ^ & * ( ) – = \ + \ [ ] { } < > ?

- You can use a maximum of 255 characters for a tag name or a class member name.You can use uppercase and lowercase characters. Tag names are *not* case sensitive. Because Visual Designer does not differentiate between uppercase and lowercase characters, you can use both to make tag names more readable. (For example: `TankLevel` instead of `tanklevel`.)

- Tag names must be different from system tag names and math functions.

Note: Use the @ character at the beginning of a tag name to indicate that the tag will be used as an indirect tag in the project.

Some valid tag examples include:

- `Temperature`
- `pressure1`
- `count`
- `x`

Choosing the Tag Data Type

Another consideration when designing a tag is what type of data the tag will receive. Visual Designer recognizes the following, standard tag *data types*:

- **Boolean** (*one bit*): Simple boolean with the possible values of 0 (false) and 1 (true). Equivalent to the "bool" data type in C++. Typically used for turning objects off and on or for closing and opening objects.

- **Integer** (*four bytes*): Integer number (positive, negative, or zero) internally stored as a signed 32-bit. Equivalent to the "signed long int" data type in C++. Typically used for counting whole numbers or setting whole number values. Examples: `0, 5, #200`.

- **Real** (*floating point, eight bytes*): Real number that is stored internally as a signed 64-bit. Equivalent to the "double" data type in C++. Typically used for measurements or for decimal or fractional values.

- **String** (*alphanumeric data, up to 1024 characters*): Character string up to 1024 characters that holds letters, numbers, or special characters. Supports both ASCII and UNICODE characters. Examples: `Recipe product X123, 01/01/90, *** On ***`.

You can also make a tag into a compound tag by assigning it a Class. A Class is a template consisting of two or more tag definitions, each with its own data type. You can use Classes in projects that have items (e.g., tanks of liquid) with multiple attributes (e.g., fill level, temperature, pressure) to be monitored or controlled.

You can find these tag types (and their respective icons) in the **Global** tab of the **Project Explorer**.

See also: Understanding Tag Properties and Parameters

Changing How Boolean Tags Receive Numeric Values

By default, if any numeric value other than 0 (i.e., ≠0) is written to a Boolean tag, then the tag automatically assumes a value of 1. You can change this behavior, if necessary, by editing the *project_name*.app file to change the following setting:

```
[Options]
BooleanTrueAboveZero=value
```

If BooleanTrueAboveZero is set to the default 0, then the project will behave as described above. If BooleanTrueAboveZero is set to 1, then the project will behave as follows:

- When you write any numeric value less than or equal to 0 (i.e., ≤0) to a Boolean tag, the tag assumes a value of 0 (false).

- When you write any numeric value greater than 0 (i.e., >0) to a Boolean tag, the tag assumes a value of 1 (true).

⚠ CAUTION

This is a global runtime setting. If you only want to change how certain tags are handled, then you should not change this setting.

Using Array Tags

Visual Designer tags can consist of a single value or an array of values.

Note: The maximum array size is 16384 as long as it does not exceed the maximum number of tags supported by the license (Product Type) selected for the project. Each array position (including the position 0) counts as one tag for licensing restrictions, because each position has an independent value.

An array tag is a set of tags with the same name, which is identified by indexes (a matrix of *n* lines and 1 column). The maximum array size depends on the product specification. You can use the following syntax to access an array tag:

ArrayTagName[*ArrayIndex*]

For example: tank[0], tank[1], tank[2], and tank[500].

⚠ CAUTION

You must specify a maximum index for each array tag in the **size** column of any datasheet. You can specify *n* to indicate the array tag has positions from 0 to *n*. For example, if the size of TagA is 3, the tag elements could be TagA[0], TagA[1], TagA[2], and TagA[3].

Use the array tag whenever possible because it optimizes memory use and simplifies the configuration task. For example, if you want a display to monitor each tank, you could use array tags to configure a single display containing tags linked to any tank. For example (using the tk tag as an index containing the number of the tank): pressure[tk], temperature[tk], and temperature[tk+1].

An array index can be a tag, a numeric value, or an expression with the arithmetic operator "+".

Note: When you refer to an array with an index using 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. For example: `temperature[tk+2]` or `temperature[tk+6]`.

Using array tags in any Visual Designer task can save a signification amount of project development time. For example, if you needed tag points related to the temperature of four tanks. The conventional configuration method is the following:

- **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, as follows:

- **temperature[j]**: high temperature on tank {j}

Note: When you create a four-position array tag, the system creates five positions (from 0 to 4). For example:

`tag_example[15] //start position=0, end position=15`

Therefore, the `tag_example[15]` array has 16 elements.

When using another tag to reference the index of an array, if the value of the tag is outside the size of the array, then the following results are given:

- If *IndexTag* is greater than the size of the array, then `MyArray[IndexTag]` will point to the end position of the array; and

- If *IndexTag* is less than 0, then `MyArray[IndexTag]` will point to the start position of the array.

Array Tags

An *array* tag consists of a set of tags that all have the same name, but use unique array indexes (a matrix of *n* lines and one column) to differentiate between each tag. An *array index* can be a fixed value, another tag or an expression. Maximum array sizes are determined by product specifications.

You can use array tags to:

- Simplify configurations

- Enable multiplexing in screens, recipes, and communication interfaces

- Save development time during tag declaration

You specify array tags in one of two formats:

- For a simple array tag, type:

  ArrayTagName[*ArrayIndex*]

- For a complex array tag (where the array index is an expression consisting of a tag and an arithmetic operation), type:

  ArrayTagName[*ArrayIndex+c*]

  Where:

  - *ArrayTagName* is the tag name;

  - [*ArrayIndex*] is the unique index (fixed value or another tag);

  - + is an arithmetic operation; and

  - *c* is a numerical constant.

Note:

- You must specify a maximum index for each array tag by typing a value (*n*) in the Array Size column of an **Project Tags** datasheet or in the Array Size field on a **New Tag** dialog. (See "Creating project database Tags").

  When you create an *n*-position array tag, Visual Designer actually creates **n+1** positions (from 0 to *n*). For example, if you specify `ArrayTag[15]`, the array will have 16 elements, where 0 is the start position and 15 is the end position.

- You must not use spaces in an array tag.

  When Visual Designer reads a tag it begins with the first character and continues until it finds the first space or null character. Consequently, the system does not recognize any characters following the space as part of the array tag.

  For example, if you type `a[second + 1]`, Visual Designer regards `a[second` as the tag and considers it invalid because Visual Designer does not find (recognize) the closing bracket. However, if you type `a[second+1]`, this is a valid array tag.

You can specify an array tag wherever you would use a variable name. Also, because array tags greatly simplify configuration tasks and can save development time, we suggest using them whenever possible.

For example, suppose you want to monitor 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

You can use array tags to simplify this task as follows (where [*n*] represents the tank number):

- `temperature[n]` — high temperature on tank [*n*]

The following table contains some additional examples of an array tag:

Array Tag Examples

Array Tag Example	Description
`Tank[1]`, `Tank[2]`, `Tank[500]`	Simple arrays, where the array indexes (`1`, `2`, and `500`) are numerical constants. For example, tank numbers.
`Tank[tk]`	A simple array, where the array index (`tk`) is a tag. For example, a tag representing the tank number.
`Tank[tk+1]`	A complex array, where the array index (`tk+1`) is an expression. For example, the value of `tk` (tank number) plus 1.

Note: When using another tag to reference the index of an array, if the value of the tag is outside the size of the array, then the following results are given:

- If *IndexTag* is greater than the size of the array, then `MyArray[IndexTag]` will point to the end position of the array; and

- If *IndexTag* is less than 0, then `MyArray[IndexTag]` will point to the start position of the array (i.e., `MyArray[0]`).

About indirect tags

Visual Designer 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.

Note: Any tag created as a string-type tag is potentially an indirect tag (pointer).

To refer to a class-type tag, you can declare a string-type tag that points to a class tag. For example:

Class	TANK with members `Level`
Tag	TK of the class `TANK`
Tag	XCLASS of the String type

To access the `TK.Level` value, you must store the "`TK.Level`" value within the `XCLASS` tag and use the syntax, `@XCLASS`. You can also refer to a member of a class-type tag directly; identifying a class-type that points to a class member.

For example:

Class	TANK with members `Level`
Tag	TK of the class `TANK`
Tag	XCLASS of the class `TANK`

To access the `TK.Level` value, you must store the "`TK`" value within the `XCLASS` tag and use the syntax, `@XCLASS.Level`.

When creating 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 you are creating a reference. Follow the XCLASS example: `@Z Integer, @X Class:TANK`.

Indirect Tags

Indirect tags "point" to other database tags (including class-type tags). Using indirect tags can save development time because they keep you from having to create duplicate tags (and the logic built into them).

You create an indirect tag from any string-type tag simply by typing the `@` symbol in front of the tag name `@TagName`.

- To reference a simple tag, assume the `strX` tag (a string tag) holds the value "`Tank`", which is the name of another tag, then reading from or writing to `@strX` provides access to the value of the `Tank` tag.

- To reference a class-type tag and member, you simply create a string tag that points to the class tag and the member. For example, if a tag `strX` (a string tag) holds the value "`Tank.Level`", which is the name of the class tag, then reading from or writing to `@strX` provides access to the value of the `Tank.Level` member.

- You can also point directly to a class-type tag member; by identifying a class-type that points to a class member. For example: to access the `Tank.Level` member of the class, you must store the "`Tank`" value within the `strX` tag and use the syntax, `@strX.Level`.

List of Tag Properties

Tag properties (also known as "tag fields") are metadata attached to each tag in the database. Most of these properties can be set using the **Tag Properties** dialog, which you can open by clicking the **Tag Properties** button on the Tag Properties toolbar.

To access a tag property during runtime, use the following syntax (without spaces) anywhere that you would normally specify a tag:

tag_name->property_name

You can access the following tag properties during runtime:

Tag Property	Description	R or R/W	Data Type	Available on Data Type…				
				Bool	Int	Real	Str	Retain
Name	The name of the tag, as configured in the Project Tags database.	R	String, up to 32 chars	Y	Y	Y	Y	n/a
MemberName	The name of the class member, in a properly configured Class. NOTE: The syntax must be: *Class.Member->MemberName* Example: Tank.Lvl->MemberName = Lvl	R	String, up to 32 chars	Y	Y	Y	Y	n/a
Size	Array Size. If the tag is not an array tag, it returns the value 0	R	Integer	Y	Y	Y	Y	n/a
Index	The index number of an element in an Array. (An Array is any Tag of size greater than 0.) NOTE: The syntax must be: *Tag[Index]->Index* Example: Tag[1]->Index = 1	R	Integer	Y	Y	Y	Y	n/a
Description	The description of the tag, configured in the Tags datasheet.	R	String	Y	Y	Y	Y	Y
Quality	Tag quality (192=GOOD; 0=BAD). The project updates this field every time the tag receives the result of an expression or a value from a communication task (such as driver or OPC). If the expression is invalid (such as, division by zero) or if there is a reading communication error associated with the tag, then the project sets the quality to BAD.	R	Integer	Y	Y	Y	Y	N
TimeStamp	Time and date when the value of the tag last changed.	R	String	Y	Y	Y	Y	N
Blocked	This property can have two values: • 0: The tag is blocked and all runtime tasks will ignore it. It is effectively removed from the project database. • 1: The tag is unblocked and all runtime tasks can access it normally. This is useful when you want to dynamically disable all actions associated with a specific	R/W	Boolean	Y	Y	Y	Y	N

Tag Property	Description	R or R/W	Data Type	Available on Data Type…				
				Bool	**Int**	**Real**	**Str**	**Retain**
	tag. Even when a tag is blocked, however, it still counts towards the total number of tags used for licensing purposes.							
Unit	A brief description (up to 9 characters) of the Engineering Unit (i.e., the unit of measurement) for the Tag value. For example, Kg, BTU, psi.	R/W	String, up to 9 chars	Y	Y	Y	Y	Y
Max	The maximum value that can be written to the tag during runtime.	R/W	Real	N	Y	Y	N	Y
Min	The minimum value that can be written to the tag during runtime	R/W	Real	N	Y	Y	N	Y
B0 … B31	Value (0 or 1) of any of the 32 bits (b0, b1, b2, … b31) of an Integer tag. (B0: LSB B31: MSB)	R/W	Boolean	N	Y	N	N	N
DisplayValue	A converted Tag value that is only displayed on-screen: **DisplayValue** = (**Value** / **UnitDiv**) + **UnitAdd** This is used when the actual Tag values have one Engineering Unit (see **Unit** above) but need to be displayed on-screen in another Engineering Unit (see **DisplayUnit** below). For example, Celsius degrees and Farenheit degrees. If user input changes **DisplayValue** during runtime, then the conversion is reversed before the change is actually written to the Tag: **Value** = (**DisplayValue** – **UnitAdd**) * **UnitDiv**	R/W	Real	N	Y	Y	N	n/a
DisplayUnit	A brief description (up to 9 characters) of the Engineering Unit for **DisplayValue**. NOTE: This property can only be set by using the SetDisplayUnit and SetTagDisplayUnit functions.	R	String, up to 9 chars	N	Y	Y	N	N
UnitDiv	Number by which the Tag value is divided to get **DisplayValue**. To perform no division, **UnitDiv** should be 1. NOTE: This property can only be set by using the SetDisplayUnit and SetTagDisplayUnit functions.	R	Real	N	Y	Y	N	N
UnitAdd	Number added to the Tag value to get **DisplayValue**. To perform no addition, **UnitAdd** should be 0. NOTE: This property can only be set by using the SetDisplayUnit and SetTagDisplayUnit functions.	R	Real	N	Y	Y	N	N
DisplayMax	The maximum value that can be input to **DisplayValue** during runtime: **DisplayMax** = (**Max** / **UnitDiv**) + **UnitAdd**	R/W	Real	N	Y	Y	N	N

Tag Property	Description	R or R/W	Data Type	Available on Data Type…				
				Bool	Int	Real	Str	Retain
	If **DisplayMax** is changed during runtime, then **Max** is also changed as follows: **Max** = (**DisplayMax** – **UnitAdd**) * **UnitDiv**							
DisplayMin	The minimum value that can be input to **DisplayValue** during runtime: **DisplayMin** = (**Min** / **UnitDiv**) + **UnitAdd** If **DisplayMin** is changed during runtime, then **Min** is also changed as follows: **Min** = (**DisplayMin** – **UnitAdd**) * **UnitDiv**	R/W	Real	N	Y	Y	N	N
HiHiLimit	Limit value for the HiHi alarm.	R/W	Real	N	Y	Y	N	Y
HiLimit	Limit value for the Hi alarm.	R/W	Real	N	Y	Y	N	Y
LoLimit	Limit value for the Lo alarm.	R/W	Real	N	Y	Y	N	Y
LoLoLimit	Limit value for the LoLo alarm.	R/W	Real	N	Y	Y	N	Y
RateLimit	Limit value for the Rate alarm.	R/W	Real	N	Y	Y	N	Y
DevSetpoint	Setpoint value for Deviation alarms.	R/W	Real	N	Y	Y	N	n/a
DevPLimit	Limit value for the Deviation+ alarm.	R/W	Real	N	Y	Y	N	Y
DevMLimit	Limit value for the Deviation- alarm.	R/W	Real	N	Y	Y	N	Y
HiHi	If 0, the HiHi alarm is not active. If 1, the HiHi alarm is active.	R	Boolean	Y	Y	Y	N	n/a
Hi	If 0, the Hi alarm is not active. If 1, the Hi alarm is active.	R	Boolean	Y	Y	Y	N	n/a
Lo	If 0, the Lo alarm is not active. If 1, the Lo alarm is active.	R	Boolean	Y	Y	Y	N	n/a
LoLo	If 0, the LoLo alarm is not active. If 1, the LoLo alarm is active.	R	Boolean	Y	Y	Y	N	n/a
Rate	If 0, the Rate alarm is not active. If 1, the Rate alarm is active.	R	Boolean	Y	Y	Y	N	n/a
DevP	If 0, the Deviation+ alarm is not active. If 1, the DevP alarm is active.	R	Boolean	N	Y	Y	N	n/a
DevM	If 0, the Deviation- alarm is not active. If 1, the DevM alarm is active.	R	Boolean	N	Y	Y	N	n/a
AlrStatus	Integer value with the status of the current active alarms associated to the tag. Each bit of this integer value indicates a specific status: • Bit 0 (LSB): HiHi Alarm active • Bit 1: Hi Alarm active • Bit 2: Lo Alarm active • Bit 3: LoLo Alarm active • Bit 4: Rate Alarm active • Bit 5: Deviation+ Alarm active • Bit 6: Deviation- Alarm active	R	Integer	Y	Y	Y	N	N

Tag Property	Description	R or R/W	Data Type	Available on Data Type…				
				Bool	**Int**	**Real**	**Str**	**Retain**
	Examples: If Tag⊠AlrStatus returns the value 2, it means that "Hi" alarm is active. If it returns the value 3, it means that the "HiHi" and the "Hi" alarm are active simultaneously.							
	If this property returns the value 0, it means that there are no active alarms associated to this tag.							
	For Boolean tags, only the values 1 (bit 1), 4 (bit 2) or 16 (bit 4) can be returned.							
Ack	This property can have two values:	R	Boolean	Y	Y	Y	N	N
	• 0: There are no alarms associated with this tag that require acknowledgment.							
	• 1: There is at least one alarm associated with this tag that requires acknowledgment.							
	This works as a global acknowledge for the tag and goes to 0 only when all alarms for the tag have been acknowledged.							
UnAck	This property can have two values:	R/W	Boolean	Y	Y	Y	N	N
	• 0: There is at least one alarm associated with this tag that requires acknowledgment.							
	• 1: There are no alarms associated with this tag that require acknowledgment.							
	If you manually set this value to 1, then the active alarms (if any) are acknowledged. The value of this property is always the opposite of the **Ack** property.							
AlrAckValue	Text associated with the Acknowledged state of a Boolean tag. This text is displayed in the Value column of an Alarm/Event Control.	R/W	String, up to 32 chars	Y	N	N	N	Y
	You can also edit this text in the **Tag Properties** dialog (Alarms – Bool Type).							
AlrOffValue	Text associated with the Normalized state of a Boolean tag. This text is displayed in the Value column of an Alarm/Event Control.	R/W	String, up to 32 chars	Y	N	N	N	Y
	You can also edit this text in the **Tag Properties** dialog (Alarms – Bool Type).							
AlrOnValue	Text associated with the Active state of a Boolean tag. This text is displayed in the Value column of an Alarm/Event Control.	R/W	String, up to 32 chars	Y	N	N	N	Y
	You can also edit this text in the **Tag Properties** dialog (Alarms – Bool Type).							
AlrDisable	This property can have two values:	R/W	Boolean	Y	Y	Y	N	N
	• 0: The alarms associated with this tag are enabled. This means that when an alarm condition occurs, the alarm will become active.							

About Tags and the Project Database

Tag Property	Description	R or R/W	Data Type	Bool	Int	Real	Str	Retain
	• 1: The alarms associated to this tag are disabled. This means that even if an alarm condition occurs, the alarm will not become active.							

Note:

- If a property is marked "n/a" with regards to being retentive, it's because the property is inherent in the tag definition (e.g., Name, Size) or the value of the property is continuously derived during runtime (e.g., alarm activation, DisplayValue). To enable retention for a tag, select the **Retentive Parameters** option in the **Tag Properties** dialog.

- If the project attempts to write a value outside of the range specified in the **Min** and **Max** properties, the Tags Database will not accept the new value and a warning message is written in the **Output** window. If both **Min** and **Max** properties are configured with the value 0 (zero), it means that any value applied to the tag type will can be written to the tag.

- You cannot use tag properties (such as Bit fields) to configure **Alarm** or **Trend** worksheets.

- Although you can apply tag properties to System Tags, those properties will not persist when you download your project to a CE device.

Reset Tags Database

Select **Reset Tags Database** to "reload" the tags database on the local station. This command affects all tags stored in the **Project Tags** folder. This option is useful for resetting the project tags and restoring the values they had when the project was loaded for the first time. When you stop the project but leave the development environment open, the tags are not reset by default when the project is run again. Therefore, you can execute this command to reset them before the project runs again.

When this command is executed, the **Startup Value** configured for each tag (**Tags Properties** dialog) is written to the respective tag. If you did not configure any **Startup Value** for a numeric tag (**Boolean**, **Integer** or **Real**), the value 0 (zero) is written to the tag. If you did not configure any **Startup Value** for a string tag, the empty value ("") is written to the tag.

This command is disabled (in gray) if there is at least one runtime task running on the local station. You must close all runtime tasks (**Stop** on the Home tab of the ribbon) before this command can be executed.

Note: The tags stored in the **System Tags** folder and in the **Shared Tags** folder (if any) are not affected by this command.

Note: If you want to reset the project tags automatically whenever you run the project (**Run** on the Home tab of the ribbon), you can check the option **Reset Tags Database** when starting project on the **Preferences** tab of the **Project Settings** dialog.
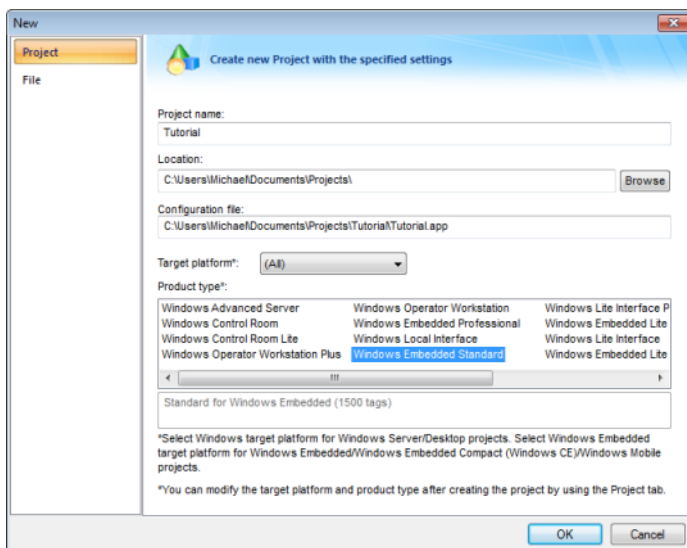
Tutorial: Building a Simple Project

This section explains, using a step-by-step tutorial, how to build a simple project, as well as how to select and configure an I/O driver.

Creating a new project

This part of the tutorial shows how to create a new project, including how to give it a name and select the target platform.

1. Click the Application button in the top-left corner of the development environment, and then click **New** on the Application menu..
   The **New** dialog is displayed.

2. Click the **Project** tab.

3. In the **Project name** box, type the name of your project.
   For this tutorial, type `Tutorial`.
   The development application automatically creates a new directory of the same name and assigns your project file to that directory. (Notice the **Configuration file** text box in the figure.) To put your project file somewhere other than in the default projects folder, click **Browse** and navigate to the preferred location.

4. In the **Product type** list, select the type of project that you want to build.
   For this example, select **Windows Embedded Standard**. This is a tag and feature-limited product type that can be safely deployed on Windows Embedded devices.

5. Click **OK**.
   The **New** dialog is closed and the **Project Wizard** dialog is displayed.

6. In the **Template** list, select **Empty Application**.

7. In the **Resolution** list, select **640 x 480**.
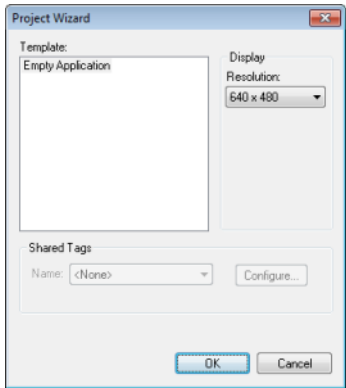
8. Click **OK**.
The **Project Wizard** dialog is closed and the new project is created in the development environment.

Specifying the startup screen

This part of the tutorial shows how to open the project settings and then specify which screen should be displayed on startup.

- Use the **Information** tab to provide information that identifies the project (such as project description, revision number, Company name, Author's name, field equipment, and general notes).

- Use the **Options** tab to specify generic settings for the project, such as the Target System, Automatic Translation, Alarm history and Events, Default Database and Shared Tags.

- Use the **Viewer** tab to enable/disable the runtime desktop parameters.

- Use the **Communication** tab to specify communication parameters relating to the project in general.

- Use the **Web** tab to specify the Web Solution settings, such as the Data Server IP address.

- Use the **Preferences** tab to enable/disable warning messages when using the development application.

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Viewer**.
The **Project Settings** dialog is displayed with the Viewer tab selected.

2. In the **Startup screen** box, type `main.scr`.
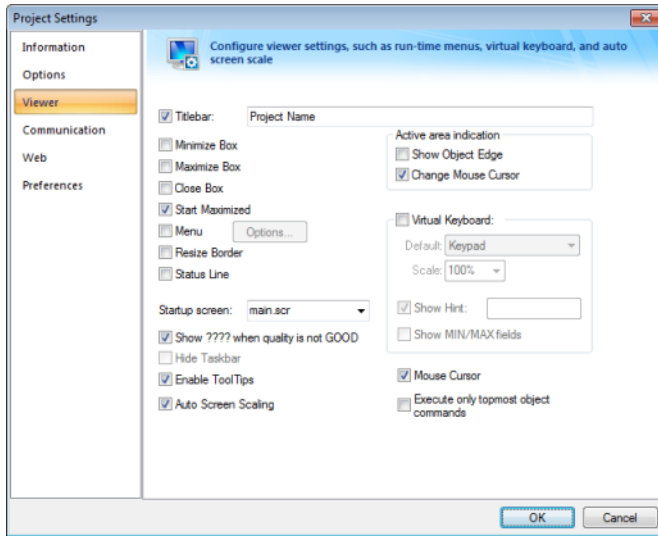
When you run the project, it will automatically display the main screen (or whichever screen you specify) first. You can specify a screen *before* you create it.

3. Click **OK**.

Creating tags

This part of the tutorial shows how to create new tags by adding them to the Project Tags datasheet.

A tag is any variable that holds a value. All tags created in an project are stored in the Project Tags folder, on the Global tab of the Project Explorer.
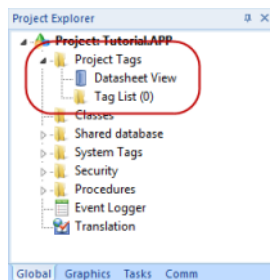
1. In the Project Explorer, click the **Global** tab.

2. Double-click **Project Tags** to expand the folder.

3. Double-click **Datasheet View** to open the **Project Tags** datasheet.

4. Use the following parameters to create a tag for the sample project.

   a) **Name**: Specify a unique tag name. For this tutorial, type `Level`.

   b) **Array**: Specify the top array index of the tag. (Simple tags have an Array of `0`.) For this tutorial, type `3`.

     Each array index relates to one of the three tanks:
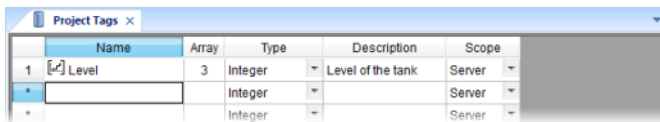
- `Level[1]` is the level of Tank #1
- `Level[2]` is the level of Tank #2
- `Level[3]` is the level of Tank #3

You will not use `Level[0]` in this tutorial, even though it is a valid tag.

c) **Type**: Specify the data type of the tag: Boolean, Integer, Real, String, or Class. For this tutorial, select **Integer**.

d) **Description** (optional): Type a description of the tag for documentation purposes only.

e) **Scope**: Specify how the tag is managed between the Server and the Thin Client stations.

- Select **Local** if you want the tag to have independent values on the Server and Client stations.

- Select **Server** if you want the tag to share the same value on the Server and Client stations.

For this tutorial, select **Server**.

	Name	Array	Type	Description	Scope	
1	Level	3	Integer	Level of the tank	Server	
*			Integer		Server	
*			Integer		Server	

5. Save and close the **Project Tags** datasheet.

You will create additional tags as you build the project.

Note: You can sort the data in the **Project Tags** datasheet or insert/remove additional columns by right-clicking on it and then choosing the applicable option from the pop-up menu.

Creating the startup screen

This part of the tutorial shows how to create your first screen, which will contain a single button that opens another screen.

1. In the **Project Explorer**, click the **Graphics** tab.

Project Explorer
- Project: Tutorial.APP
  - Screens
  - Screen Group
  - Web Pages
  - Project Symbols
  - Graphics Script
  - Symbols
  - Layout

Global | Graphics | Tasks | Comm
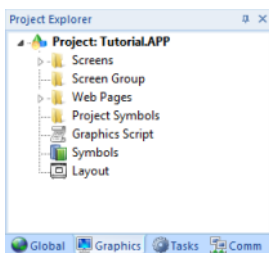
2. Right-click **Screens**, and then click **Insert** on the shortcut menu.
The development application stores all screens created for an project in this Screens folder.

The **Screen Attributes** dialog is displayed.
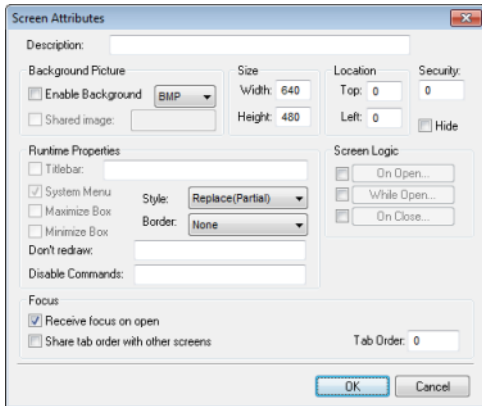
3. Use this dialog to set screen properties such as size and type.
   For this tutorial, click **OK** to accept the default settings.
   The **Screen Attributes** dialog is closed, and the new screen is opened in the workspace for editing.

4. On the **Graphics** tab of the ribbon, in the **Screen** group, click **Background Color**.
   A standard color picker is displayed.

5. In the color picker, select a light gray color.

That color is applied to the screen.

Drawing the startup screen's title

This part of the tutorial shows how to draw the startup screen's title using a Text object.

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Text**.
   Your mouse cursor changes from an arrow to a crosshair.

2. Click on the screen, type `Welcome to the Tutorial Application`, and then press Return.
   This creates a new Text object with the specified text.

3. Double-click the object to open its **Object Properties** dialog.
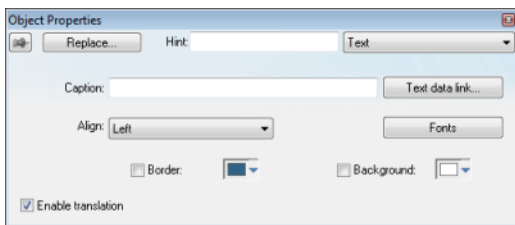
- Double-clicking on any screen object opens an **Object Properties** dialog containing the properties for that object. The properties shown in the dialog change depending on the type of object.

- The **Object Properties** dialog also contains a pin button that controls whether this dialog remains open. The button changes state (and function) each time you click on it, as follows:

  - When the pin button is released, the focus is passed to the object on the screen as soon as it is selected. It is recommended that this button is kept released when you want to manipulate the objects (Copy, Paste, Cut, or Delete). Although the **Object Properties** dialog is on the top, the keyboard commands (**Ctrl+C**, **Ctrl +V**, **Ctrl+X**, or **Del**) are sent directly to the objects.

  - When the pin button is pressed, the focus is kept on the **Object Properties** dialog, even when you click the objects on the screen. We recommend you keep this button pressed when you want to modify the settings of the objects. You can click an object and type the new property value directly in the **Object Properties** dialog (it is not necessary to click on the window to bring focus to it). Also, when the pin button is pressed, the **Object Properties** dialog does not automatically close when you click on the screen.

4. Click **Fonts** to open **Font** dialog, and then specify the font settings.
   For this tutorial…

   - Font is **Arial**

   - Font style is **Regular**

   - Size is **20**

   - Color is **Blue**
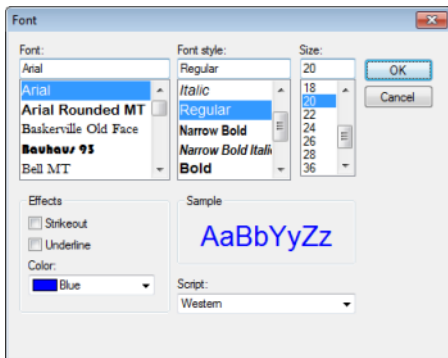
5. Click **OK** to close the **Font** dialog.
   The font settings are applied to the Text object.
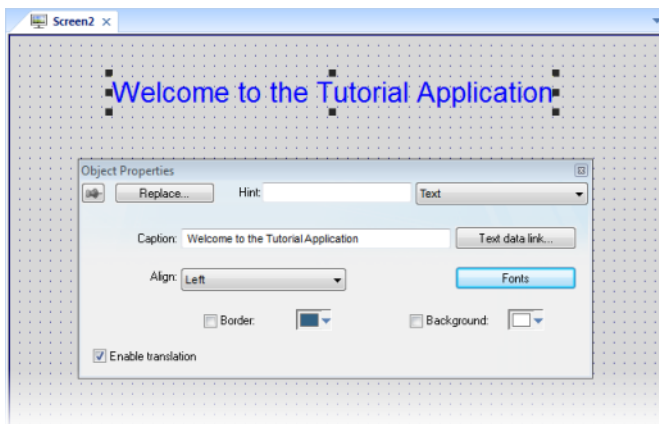
6. Close the **Object Properties** dialog (i.e., click the Close button in the dialog's top-right corner).

Drawing a button to open another screen

This part of the tutorial shows how to draw and configure a button that will open another screen.

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Button**.
   Your mouse cursor changes from an arrow to a crosshair.
2. Click and hold on the screen, and then drag the cursor to draw the Button object.
3. Double-click the object to open its **Object Properties** dialog.
4. In the **Caption** box, type the following text: `Click here to open the synoptic screen`.
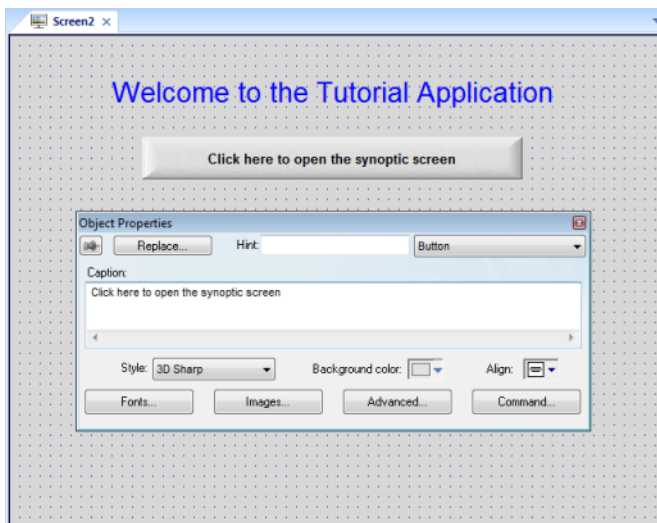
5. Click **Command**.
   The **Object Properties** dialog changes to show the properties for the Command animation.
6. In the **Type** list, select **Open Screen**.
7. In the **Open Screen** box, type `synoptic.scr`.
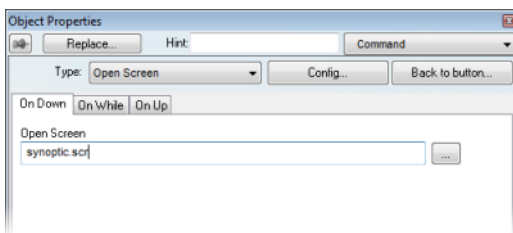
   You can specify a screen that you have not yet created.
8. Close the **Object Properties** dialog.

Saving and closing the startup screen

This part of the tutorial shows how to properly save and close a screen.

1. Click the Application button at the top-left of the development application, and then click **Save** on the Application menu.
   A standard Windows **Save** dialog is displayed.
2. In the **File name** box, type `main.scr`.
3. Click **Save**.

The file is saved in your project folder (*project_name*\Screen\main.scr), and the **Save** dialog is closed.

4. Click the Application button at the top-left of the development application, and then click **Close** on the Application menu.

Creating the synoptic screen

This part of the tutorial show how to create your second screen, which will include an animated tank of liquid and some basic controls for that tank.

1. In the **Project Explorer**, click the **Graphics** tab.
2. Right-click the **Screens** folder, and then click **Insert** on the shortcut menu.
   The **Screen Attributes** dialog is displayed.
3. Use this dialog to set attributes such as size and type.
   For this tutorial, click **OK** to accept the default settings.
4. Click the Application button at the top-left of the development application, and then click **Save** on the Application menu.
   A standard Windows **Save** dialog is displayed.
5. In the **File name** box, type synoptic.scr.
6. Click **Save**.
   The file is saved in your project folder (*project_name*\Screen\synoptic.scr), and the **Save** dialog is closed.

Drawing the synoptic screen's title

As in a previous part, this part of the tutorial shows how to draw the synoptic screen's title using a Text object.

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Text**.
2. Click on the screen, type Synoptic Screen, and then press Return.
3. Double-click the object to open its **Object Properties** dialog.
4. Click **Fonts** to open **Font** dialog, and then specify the font settings.
   For this tutorial…

   • Font is **Arial**

   • Font style is **Bold**

   • Size is **20**

   • Color is **Blue**
5. Close the **Object Properties** dialog.
6. Move the Text object to the top left corner of the screen.
7. Click the Application button at the top-left of the development application, and then click **Save** on the Application menu.

This figure shows how your screen should look after you've created the date and time objects.


Drawing "Date" and "Time" displays

This part of the tutorial shows how to draw "Date" and "Time" displays by linking Text objects to system tags.

Date and Time are system tags that hold the current date and time of the local station. These tags are available to any project.

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Text**.

2. Click on the screen, type `Date: #########`, and then press Return.

3. Double-click the object to open its **Object Properties** dialog.

4. Click **Text Data Link**.
   The **Object Properties** dialog changes to show the properties for the Text Data Link animation.

5. In the **Tag/Expression** box, type `Date`.
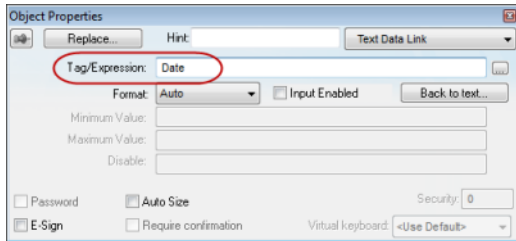
During runtime, the project replaces the `#########` characters of the Text object with the value of the system tag `Date`.

6. Close the **Object Properties** dialog.

7. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Text**.

8. Click on the screen, type `Time: #########`, and then press Return.

9. Double-click the object to open its **Object Properties** dialog.

10. Click **Text Data Link**.
   The **Object Properties** dialog changes to show the properties for the Text Data Link animation.

11. In the **Tag/Expression** box, type `Time`.
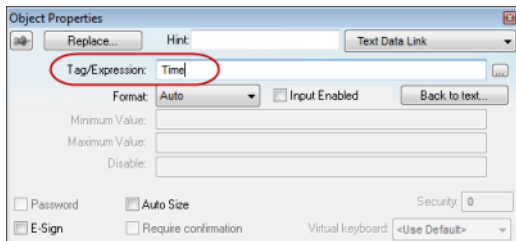
During runtime, the project replaces the `#########` characters of the Text object with the value of the system tag `Time`.

12. Close the **Object Properties** dialog.

13. Click the Application button at the top-left of the development application, and then click **Save** on the Application menu.

This figure shows how your screen should look after you've created the date and time objects.
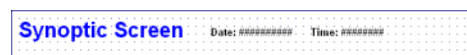

Placing an "Exit" icon

This part of the tutorial shows how to place an icon (by selecting and configuring a Linked Symbol) that allows the user to exit the project, .

1. On the **Graphics** tab of the ribbon, in the **Libraries** group, click **Symbols**.
   The symbols library is displayed.

2. In the Symbols menu tree, open the **System Symbols** folder and then open the **Icons** sub-folder.

3. In the Icons sub-folder, select **exit01**.
The symbol will be displayed in the symbol viewer to the right of the menu tree.
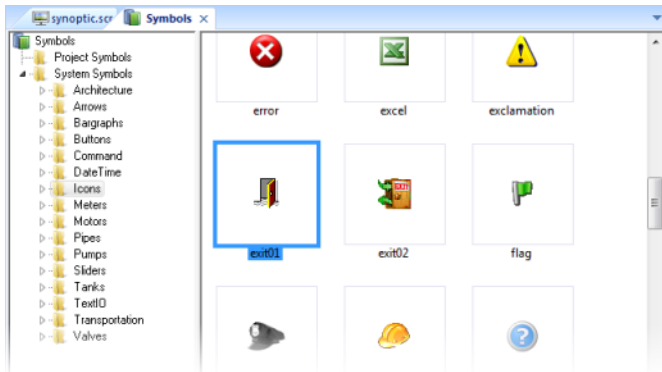
4. Click on the symbol.
The mouse cursor will change to show that the symbol is ready to be placed in a screen.
5. Switch back to the screen where you want to place the symbol and then click in it.
The symbol is placed as a Linked Symbol object.

6. With the object still selected, click **Command** (on the **Graphics** tab of the ribbon, in the **Animations** group) to apply this animation to the object.
7. Double-click the object to open its **Object Properties** dialog.
8. In the **Type** list, select **VBScript**.
9. In the **On Down** box, type `$Shutdown()`.
`Shutdown` is one of Visual Designer's built-in scripting functions, but it can be used within VBScript by prefacing it with a dollar sign ($).
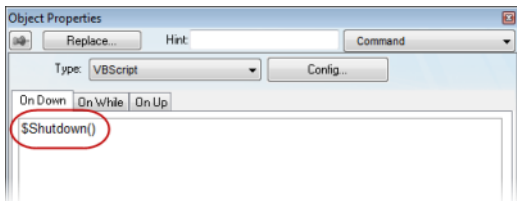
10. Close the **Object Properties** dialog.
11. Click the Application button at the top-left of the development application, and then click **Save** on the Application menu.

Now, when a user clicks this icon during runtime, the project will stop and exit to the station's desktop.

Testing the project

This part of the tutorial show how to test the project so far.

1. On the **Home** tab of the ribbon, in the **Local Management** group, click **Run**.
The project runs and the startup screen is displayed.
2. Click the button to open the synoptic screen.
The synoptic screen is displayed.
3. Click the exit icon to shut down the project.

If any part of the project doesn't work as expected, switch back to the development application (**ALT+TAB**) and then click **Stop** on the Home tab of the ribbon.

Placing an animated tank

This part of the tutorial shows how to select an animated tank from the Symbol Library and place it on the screen (similar to how you selected and placed the "Exit" icon), then associate some project tags with the tank's properties.

1. On the **Graphics** tab of the ribbon, in the **Libraries** group, click **Symbols**.
2. In the Symbols menu tree, open the **System Symbols** folder and then open the **Tanks** sub-folder.
3. In the Tanks sub-folder, select a tank symbol.

   You may select any tank you like; they all function basically the same way.
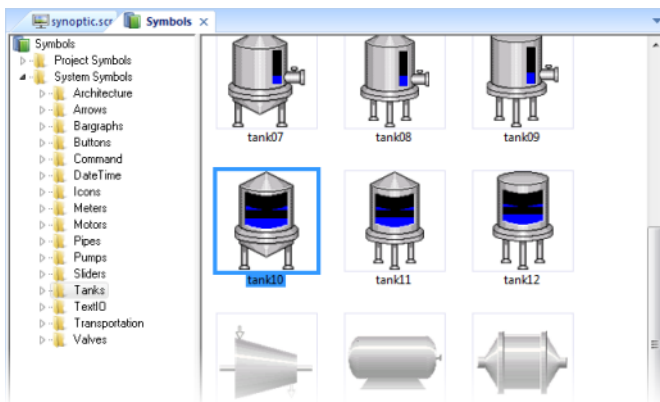
4. Click on the symbol.
   The mouse cursor will change to show that the symbol is ready to be placed in a screen.
5. Switch back to the screen where you want to place the symbol and click in it.
   The symbol is placed as a Linked Symbol object.
6. Double-click the object to open its **Object Properties** dialog.
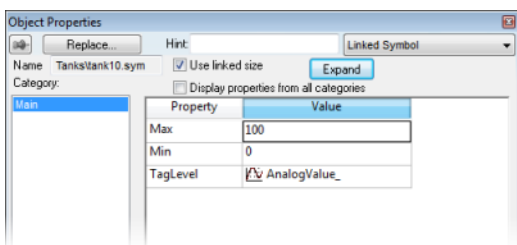
   A tank is an arrangement of different objects and animations (for example a rectangle, a bar graph, etc.), all combined together as a Linked Symbol. You can modify the properties of this symbol by editing the properties list. For this tutorial, you will modify the tag associated with the tank level.

7. For the property `TagLevel`, delete the existing value and then type `Level[Index]`.
   Note that you do not need to reopen the Project Tags datasheet to create tags as you develop the project.
   Because you have not previously created the tag `Index` in the Project Tags database, an alert message asks you if you would like to create it.
8. Click **Yes**.
   A **New Tag** dialog is displayed.

9. Configure the new tag with **Array** as 0, **Type** as Integer, and **Scope** as Local.
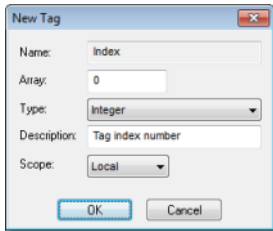
10. Click **OK** to close the **New Tag** dialog.

You can use the tag Index to set the array position of the tag Level, and show the level for any of the three tanks in the same object:

- When Index equals 1, the tank object shows the level of Tank #1 (i.e., Level[1]);
- When Index equals 2, the tank object shows the level of Tank #2 (i.e., Level[2]); and
- When Index equals 3, the tank object shows the level of Tank #3 (i.e., Level[3]).

Also, because the tag scope is local, the tag can have different values for the Server and Client stations at the same time. Consequently, the local user (i.e., the Server station) can be monitoring the level of Tank #1 while the remote user (i.e., the Client station) is monitoring the level of Tank #2.

11. Close the **Object Properties** dialog.

12. Click the Application button at the top-left of the development application, and then click **Save** on the Application menu.

This figure shows how your screen should look after you've created the tank object.

Placing a level slider

This part of the tutorial shows how to select a slider control from the Symbol Library and then connect it to the animated tank.

1. On the **Graphics** tab of the ribbon, in the **Libraries** group, click **Symbols**.

2. In the Symbols menu tree, open the **System Symbols** folder and then open the **Sliders** sub-folder.
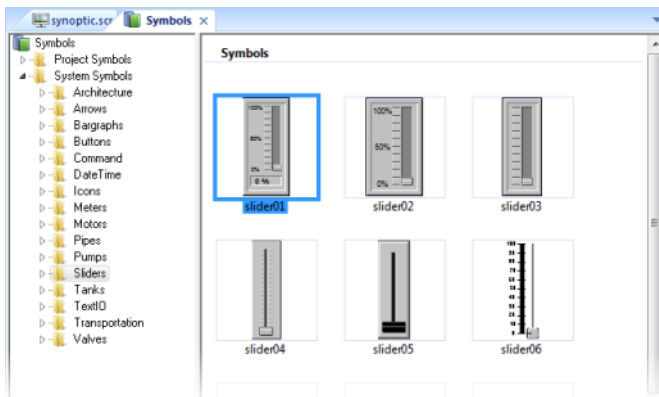

3. In the Sliders sub-folder, select a slider control.

You may select any slider you like; they all function basically the same way.

4. Click on the symbol.
The mouse cursor will change to show that the symbol is ready to be placed in a screen.

5. Switch back to the screen where you want to place the symbol and click in it.
The symbol is placed as a Linked Symbol object.

6. Double-click the object to open its **Object Properties** dialog.

7. For the property `TagName`, delete the existing value and then type `Level[Index]`.
Just as with the tank, you need to modify the symbol property associated with the slider level.

8. Close the **Object Properties** dialog.

9. Click the Application button at the top-left of the development application, and then click **Save** on the Application menu.
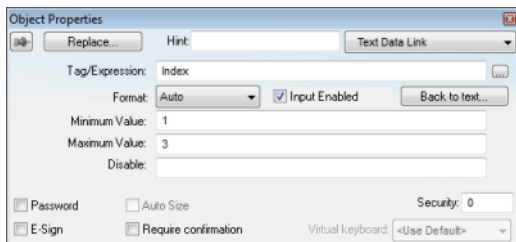
This figure shows how your screen should look after you've created the level slider object.
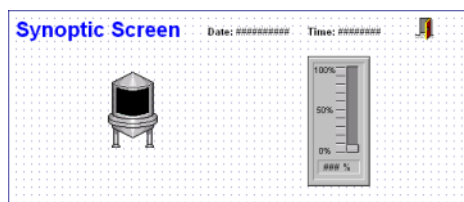
Drawing a tank selector

This part of the tutorial shows how to draw a text input box that can be used to change which real-world tank is represented by the animated tank on the screen.

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Text**.

2. Click on the screen, type `Tank: #`, and then press Return.

3. Double-click the object to open its **Object Properties** dialog.

4. Click **Text Data Link**.
The **Object Properties** dialog changes to show the properties for the Text Data Link animation.

5. In the **Tag/Expression** box, type `Index`.

6. Select the **Input Enabled** option.
This allows the operator to enter a new value for the tag during runtime.

7. In the **Minimum Value** box, type `1`.

8. In the **Maximum Value** box, type `3`.

9. Close the **Object Properties** dialog.

10. Click the Application button at the top-left of the development application, and then click **Save** on the Application menu.

Tutorial: Building a Simple Project

This figure shows how your screen should look after you've created the tank selector object.

Testing the project

This part of the tutorial show how to test the project again with the animated tank, the level slider, and the tank selector.

1. On the **Home** tab of the ribbon, in the **Local Management** group, click **Run**.
    The project runs and the startup screen is displayed.
2. Click the button to open the synoptic screen.
    The synoptic screen is displayed.
3. Type the tank number (1, 2, or 3) in the Tank label, and then use the slider to adjust the tank level.
    Note that you can view/adjust the level of each tank independently.
4. Click the exit icon to shut down the project.

If any part of the project doesn't work as expected, switch back to the development application (**ALT+TAB**) and then click **Stop** on the Home tab of the ribbon.

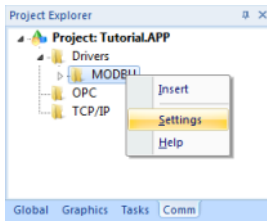## Configuring the communication driver

This part of the tutorial shows how to select and configure a driver to communicate with an external I/O device.

1. In the **Project Explorer**, click the **Comm** tab.
2. Right-click the **Drivers** folder, and the click **Add/Remove Drivers** on the shortcut menu.
    The **Communication Drivers** dialog is displayed.
3. Select a driver from the **Available drivers** list, and then click **Select**.
    For this tutorial, select MODBU.
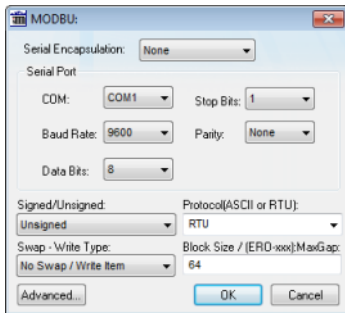   The driver is moved to the **Selected drivers** list.
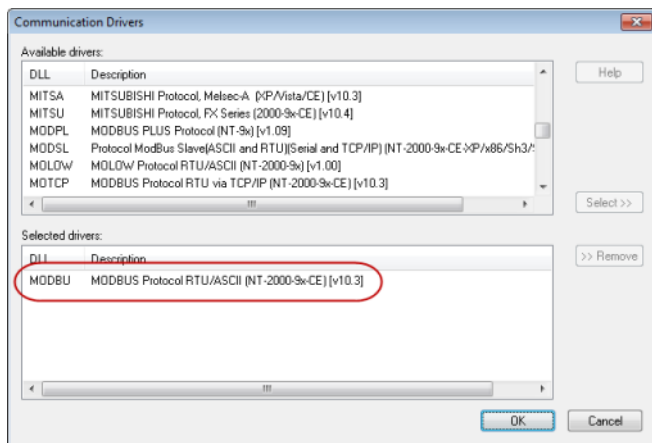
4. Click **OK**.
    The **Communication Drivers** dialog is closed, and the driver is added to the Drivers folder in the **Project Explorer**.

5. In the Project Explorer, right-click the **MODBU** folder, and then click **Settings** on the shortcut menu.

The **Communication Settings** dialog is displayed.

6. Configure the communication settings as needed for the target device.
For this tutorial, accept the default settings.

> **Note:** For more information about a specific driver, click **Communication Drivers** on the **Help** tab of the ribbon.

7. Click **OK** to close the dialog.

8. In the Project Explorer, right-click the **MODBU** folder and then click **Insert** on the shortcut menu.
A new driver worksheet named MODBU001.drv is created and opened for editing.

9. Configure the worksheet header:

a) In the **Description** box, type Tutorial Modbus.
This setting is for documentation only; it does not affect the runtime project in any way.

b) In the **Enable Read When Idle** box, type 1.
This setting is a trigger that takes a Boolean value. A value of 1 — either entered manually as above or evaluated from a tag/expression — forces your project to continue reading tag values from the target device even when there are no changes in value.

c) In the **Enable Write On Tag Change** box, type 1.
This setting is also a trigger. A value of 1 forces your project to write tag values to the target device only when those values change, rather than continuously. This saves system resources and improves performance during rutime.

d) In the **Station** box, type 1.
This indicates the I/O device number to be accessed by this driver. Typically, the PLC is specified as Device #1.
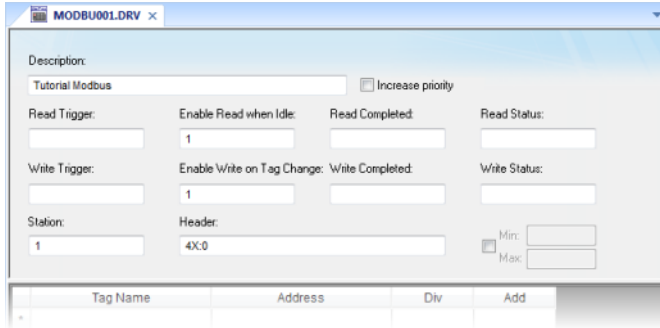
e) In the **Header** box, type 4X:0.

You must use a driver-specific format. The format for the MODBU driver is:

register_type:*initial_offset*

Register Type	Description
0X	Coil Status

Register Type	Description
1X	Input Status
3X	Input Register
4X	Holding Register
ID	Slave ID Number

10. In the worksheet body, enter the tags and their associated device addresses — for each tag:
  a) In the **Tag Name** field, type the name of the project tag.
  b) In the **Address** field, type the value to be added to the header to form the complete device address.

Tag Name	Address	Complete Device Address
Level[1]	1	4X:1 (Holding Register 1)
Level[2]	2	4X:2 (Holding Register 2)
Level[3]	3	4X:3 (Holding Register 3)
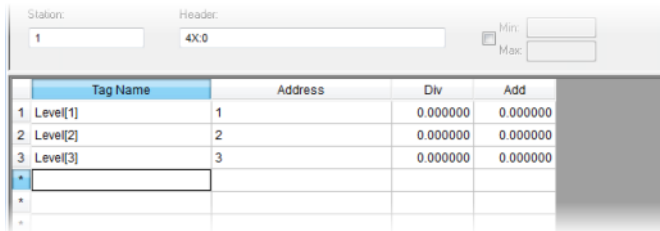
11. Click the Application button at the top-left of the development application, and then click **Save** on the Application menu.

12. When prompted to choose the driver sheet number, type 1 and then click **OK**.

Monitoring device I/O during runtime

This part of the tutorial shows how to monitor device I/O during runtime by using the **Log** window.

1. On the **Home** tab of the ribbon, in the **Local Management** group, click **Run**.
The project runs and the startup screen is displayed.

2. Press **ALT+TAB** to switch back to the development application.

3. Right-click in the **Output** window, and then click **Settings**.
The **Log Settings** dialog is displayed.

4. Select the **Field Read Commands**, **Field Write Commands**, and **Protocol Analyzer** options.

5. Click **OK** to close the **Log Settings** dialog.

You can now monitor the device I/O during runtime.

Downloading your project to a Windows Embedded device

This part of the tutorial shows how to download your project to a Windows Embedded device, such as a plant-floor HMI panel.

After configuring a project and testing it locally (on the development station), you can download it to a remote station — either a Windows PC that is running Visual Designer or a Windows Embedded device that is running CEView.
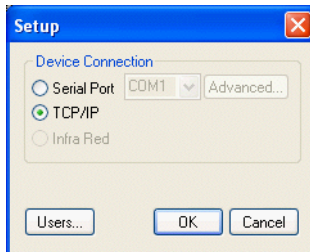
1. On the desktop of the remote station, click **Start** > **All Programs** > **EATON** > **Remote Agent** .
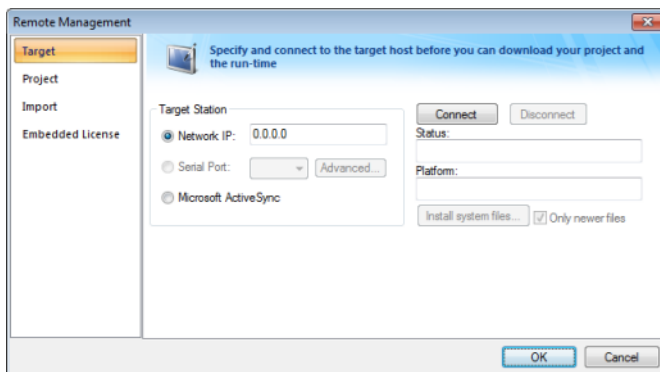   The Remote Agent utility runs.
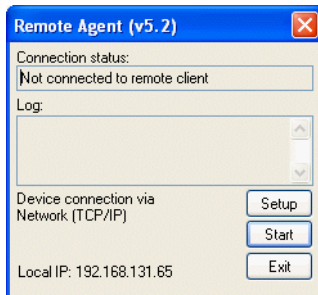
2. Click **Setup**.
    The **Setup** dialog is displayed.
3. Select the type of connection — **Serial**, **TCP/IP**, or **Infrared** — between the remote station and the development station.

    Note: For better performance, we recommend that you use TCP/IP whenever possible.

4. Click **OK** to close the **Setup** dialog, but leave the Remote Agent utility running on the remote station.
5. In the development application, click **Connect** on the **Home** tab of the ribbon.
    The **Remote Management** dialog is displayed.

6. Select the type of connection to the target (remote) station.

This selection should match the selection you previously made in the Remote Agent utility on the remote station.

7. If you selected **Network IP**, type the IP address of the remote station.

8. Click **Connect**.
    If you successfully connect to the remote station, then information about that station is displayed in the **Status** and **Platform** boxes.

9. If the remote station is a Windows Embedded device, click **Install system files**.
    The system files are installed on the remote station.

10. Click the **Project** tab.

11. Click **Download**.
    The project files are downloaded to the remote station.

12. Click **Run**.
    Your Visual Designer project is run on the remote station.

Deploying your project as a web application

This part of the tutorial shows how to deploy your project as a web application, to which remote users can connect with Internet Explorer.

For Internet Explorer to work as a web thin client, it must install an ActiveX control that "plays" Visual Designer project screens. If your computer is connected to the Internet, then IE will automatically download the control from Eaton's public server when you access a runtime project for the first time.

1. Configure the IP address of the data server.

    a) On the **Project** tab of the ribbon, in the **Web** group, click **Thin Client**.
        The **Project Settings** dialog is displayed with the **Web** tab selected.
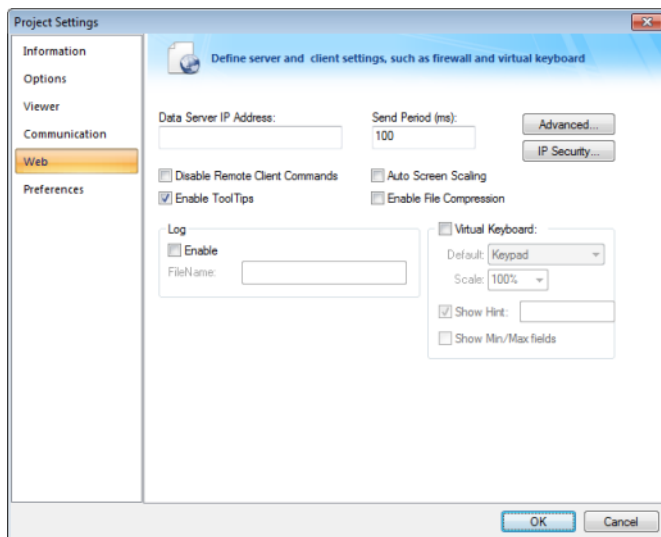

    b) In the **Data Server IP Address** box, type the IP address of the web server.
        For this tutorial, type `127.0.0.1`, which is the standard loopback address (a.k.a. "localhost").

    c) Click **OK** to close the dialog.

2. Make sure the data server is set to start up when you run your project.

    a) On the **Home** tab of the ribbon, in the **Local Management** group, click **Tasks**.

The **Execution Tasks** dialog is displayed.
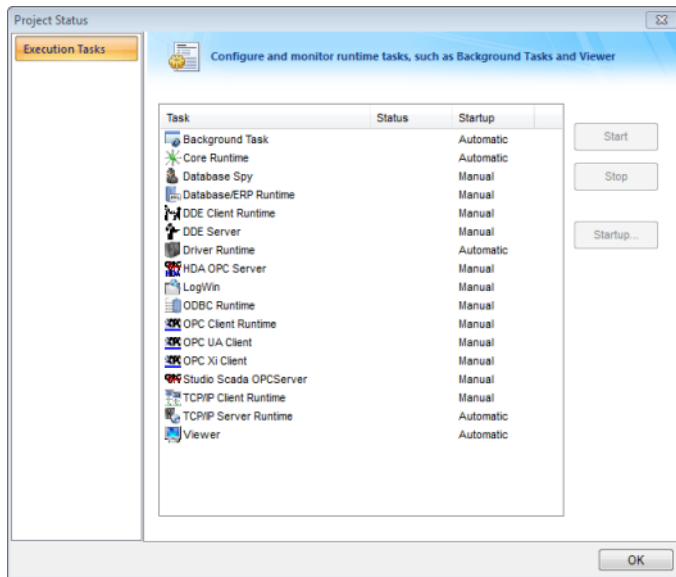
b) In the list of tasks, select **TCP/IP Server Runtime**, and then click **Startup**.
   The **Startup** dialog is displayed.
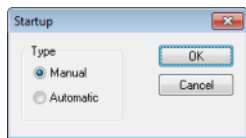
c) Select **Automatic**, and then click **OK**.
d) Click **OK** to close the **Execution Tasks** dialog.

3. Save and close all open screens and worksheets.

4. Click the Application button at the top-left of the development application, and then click
   Publish > **Save All As HTML** on the Application menu.
   Your project screens are saved as HTML files in the Web sub-folder of your project folder
   (i.e., *project_name*\Web).

5. Configure a web server to make the Web sub-folder available to web browsers.
   For this tutorial, copy […]\EATON\Visual Designer\Bin\NTWebServer.exe (a
   free, lightweight web server) to the Web sub-folder, and then run it.
   You can also configure the "root" or "home" directory of some other web server software
   (e.g., Microsoft IIS) to point to the Web sub-folder.

   Note: The web server (which makes the web pages available to clients on the network)
   and the data server (which actually runs your Visual Designer project and
   exchanges data with the clients) do not need to be the same computer.

6. On the **Home** tab of the ribbon, in the **Local Management** group, click **Run**.

7. Open a web browser (e.g., Microsoft Internet Explorer), and then enter the URL address
   of the synoptic screen on the web server.
   For this tutorial, type http://127.0.0.1/synoptic.html.
   After a few moments, during which the browser downloads and installs the ActiveX
   control, the synoptic screen is displayed in the browser.

Notice that you can modify the level of any tank either locally using the project viewer or
remotely using the web browser, and changes on one client appear immediately on the
other. They work equally well.

Eaton's Electrical Sector is a global leader in power distribution, power quality, control and automation, and monitoring products. When combined with Eaton's full-scale engineering services, these products provide customer-driven PowerChain Management® solutions to serve the power system needs of the data center, industrial, institutional, public sector, utility, commercial, residential, IT, mission critical, alternative energy and OEM markets worldwide.

PowerChain Management solutions help enterprises achieve sustainable and competitive advantages through proactive management of the power system as a strategic, integrated asset throughout its life cycle, resulting in enhanced safety, greater reliability and energy efficiency. For more information, visit www.eaton.com/electrical.

E·T·N
Powering Business Worldwide

PowerChain
Management®

PowerChain Management is a registered trademark of Eaton Corporation.

All other trademarks are property of their respective owners.