Monitoriza

Acimut

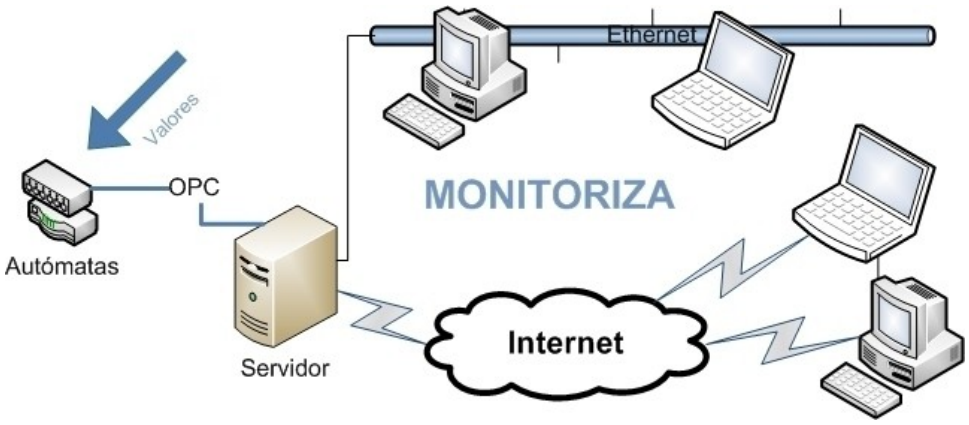
USER MANUAL

v. 3.2

http://monitoriza.acimut.com

2012

MONITORIZA
Supervisory Control and Data Acquisition (SCADA)



INDEX

INDEX **3**

MONITORIZA **5**

¿WHAT IS MONITORIZA? 5
INSTALL 7

EDITOR **10**

CREATE A PROJECT **16**

STARTING... 16
OPEN AND SAVING PROJECTS 19
CREATE VARIABLES 19
Variables Event *25*
ALARMS 27
Create Alarms *27*
Alarms Viewer *29*
Alarm Events *31*
CREATE FORMS 34

MONITORIZA CONTROLS **37**

SCADA TAB 39
Boton *39*
BotonEstado *40*
Etiqueta *41*
GrupoOpciones *43*
ImageList *44*
IndicadorAnalogico *44*
IndicadorLCD *45*
LedDisplay *47*
IndicadorNumerico *48*
Level *49*
LinkLabel *50*
PanelImagenes *50*
ProgressBar *52*
Recipe *53*
TecladoNumerico *54*
Temporizador *54*
Tendencia *55*
Text *57*
MaskedTextBox *58*
ToolTip *59*
TrackBar *59*
DESIGN TAB 59
Ellipse *60*
Image *60*
Line *62*
Rectangle *63*
Rotulo *64*
WINDOWS TAB 65

CheckBox *65*
CheckedListBox *66*
ComboBox *66*
DateTimerPicker *66*
GroupBox *66*
HScrollBar *66*
ListBox *67*
MonthCalendar *67*
NumericUpDown *67*
Panel *67*
PictureBox *67*
RadioButton *67*
TabControl *68*
RichTextBox *68*
VScrollBar *68*
WebBrowser *68*

TEST THE PROJECT **68**

IMAGE LIBRARY **71**

USERS & PERMISSIONS **74**

RECIPES & INTERFACE BATCH **77**

SAVING IN DATABASE **80**

DISPLAY HISTORIC DATA **83**

DISPLAY HISTORIC ALARMS **84**

CONFIGURE SERVER COMMUNICATIONS **85**

SERVER **86**

CLIENT **87**

EXTENSIBILITY AND PROGRAMMING **88**

INTERNAL CODE "CODEBINDINGS" 88
EXTERNAL CODE EVENTS 90
EVENT LIBRARY AND CONTROL LIBRARY *90*
THE PROGRAMMING TOOL *91*
FUNCTIONS *91*
ACCESSING MONITORIZA CONTROLS PROPERTIES *94*
ACCESING TO VARIABLES AND COMPONENTS *96*
CONTROL LIBRARY *97*
CROSS REFERENCES *99*

APPENDICE **101**

ALARMS TABLE 101
AUDIT TABLE 101
MODBUS REGISTERS 102

MONITORIZA

¿WHAT IS MONITORIZA?

Monitoriza is a monitoring and control system (SCADA Supervisory Control & Data Acquisition) which covers the requirements of any project, both basic and advanced.

Monitoriza allows us to create solutions for capturing information in industrial processes or any other field. With that information, the process gets a feedback and can be used for decision making.

It consists of three parts:

· A project editor for defining all the elements.

· A server that will execute the project and will ensure communication with the process (data acquisition, process benchmarking, etc.).

· A client that displays, visually, the information of the processes that are being monitored.

Monitoriza is a flexible system in their configuration because it can be run by multiple users simultaneously. It can operate on a stand alone or infrastructure, on a local network or through remote sites connected via the Internet.

Among the main features of Monitoriza can be highlighted:

- Simple and immediate product installation.
- Easy setup, even if it is a remote installation (WAN), because communications between client computers and server are based on Internet standards (HTTP protocol).
- Includes ModBUS native communications, Ethernet S7 for S7-300 and OPC connectivity.
- No programming required to create fully functional projects, simply "drag and drop" SCADA objects on the forms and set the corresponding properties for a working solution.
- If required advanced functionality, not covered in the objects defined in SCADA, there is no problem because Monitoriza can be extended by programming in C # or VB.Net. It is also possible to use third-party libraries developed for. NET Framework for Windows.
- The creation of the graphical user interface is based on technology Microsoft Windows Forms Designer © which greatly facilitates the design.
- At the project level, we can define users and permissions assigned to each of them. For example you can have only Read permissions or have full access to a form.
- Immediate definition of alarms.
- Effective operations monitoring.
- Increased instantaneous information.

- Easy tracking for variables.

- Information in several accessible formats. Monitoriza can store monitorized variables in standard databases on the market (Microsoft © SQL Server ™, Microsoft © Access ™, Oracle ®, etc.).

- Minimum investment and immediately redeemable.

- Definition of recipes using templates, with a user control to use recipes.

- Functions for loading batch recipes per event.

- Monitoriza provides server variables defined by OPC services and external applications that can connect to Monitoriza and access variables for use. (This service is only available in the Monitoriza Professional version and above).

INSTALL

Monitoriza has a very simple installation. It consists of two files, setup.exe and Installation Monitoriza.msi. Double clicking on setup.exe and installation begins.

Monitoriza require the. Net Framework ™ Windows ™ 3.5 SP1; if that is not installed you will get the following dialog window:

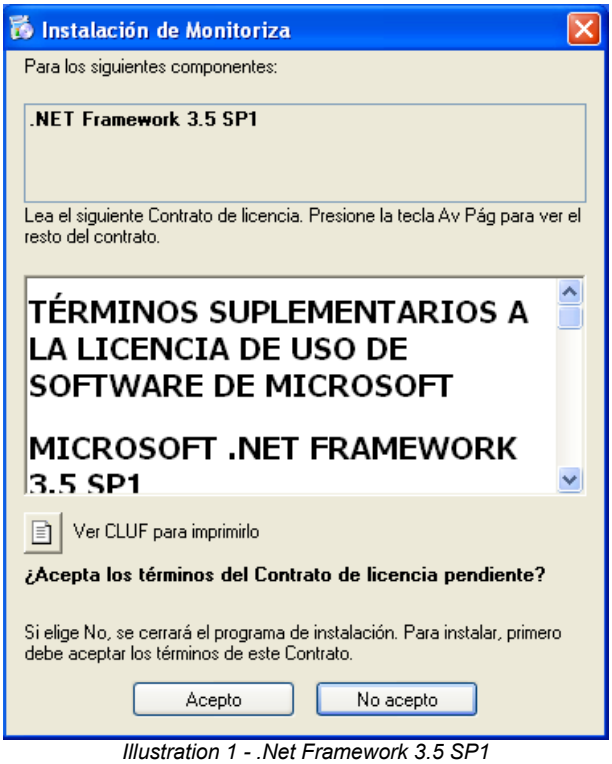


Illustration 1 - .Net Framework 3.5 SP1

In which we are asked for the License Agreement of .Net Framework from Microsoft.

Then, install proceeds to download from the Microsoft website and install the appropriate package

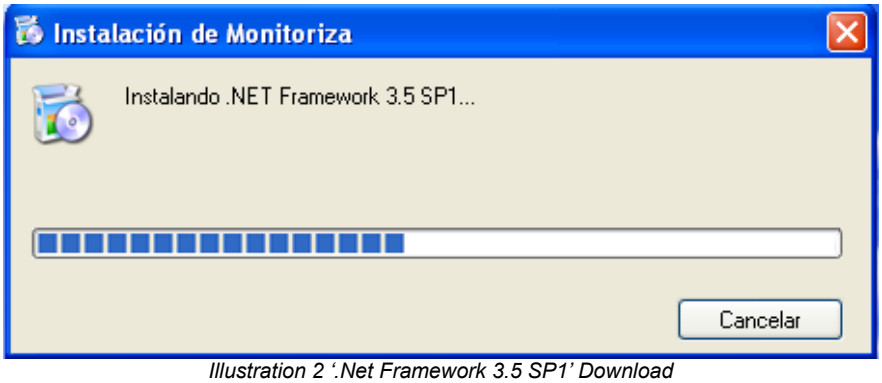


Illustration 2 '.Net Framework 3.5 SP1' Download

Once installed, start the installation of the Monitoriza main module:

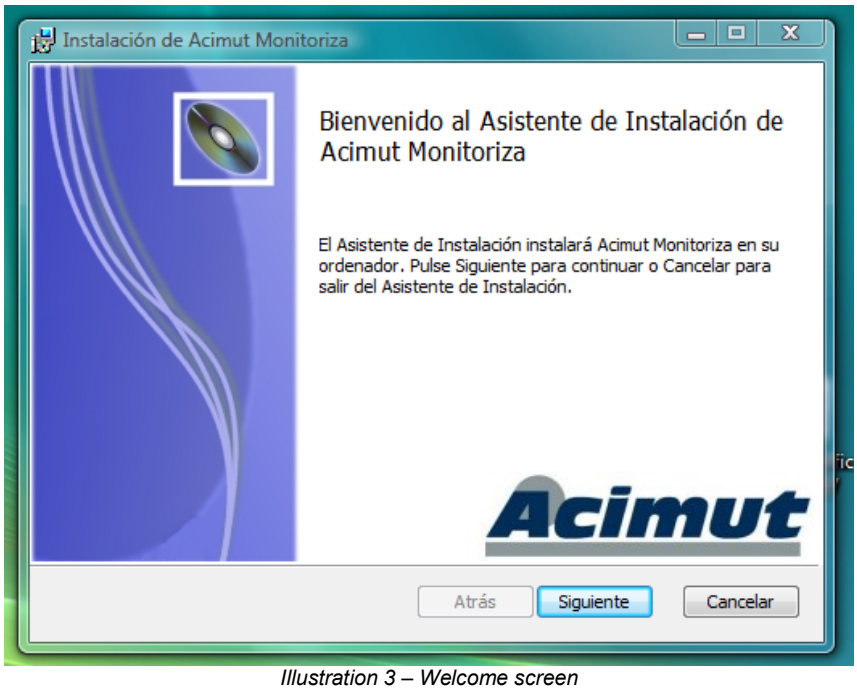


Illustration 3 – Welcome screen

Clicking on the Next button appears the custom installation screen, in which we can select the system elements that we want to install. These elements are the **Editor**, that lets you create and modify our Scada forms, the **Client**, which set the environment for project implementation and **SCADA Communication Server**, for establishing communications with both the OPC servers(and databases) and PLC.

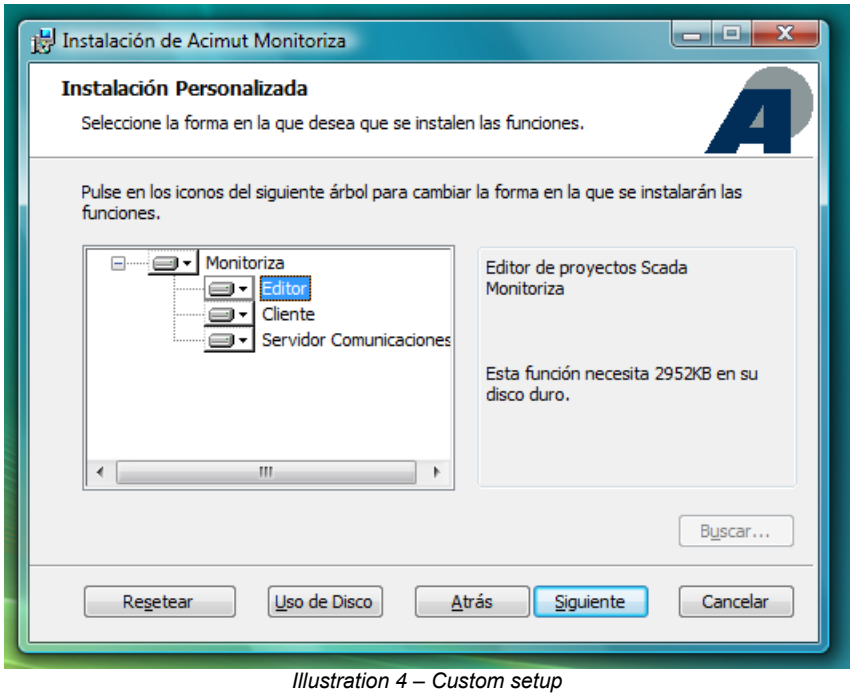


Illustration 4 – Custom setup

On the next screen we will click on the Install button to start the installation process itself:

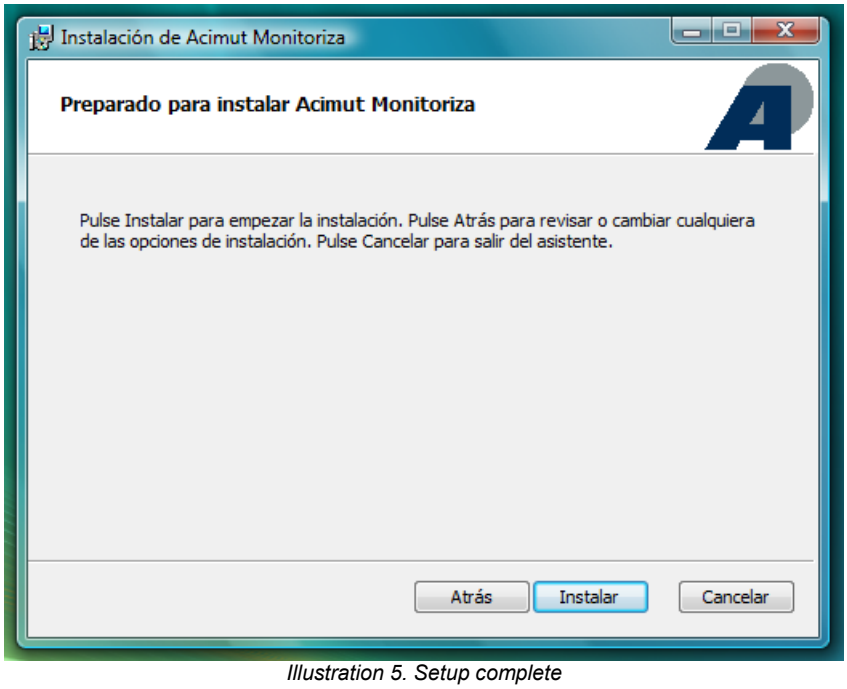


Illustration 5. Setup complete

Acimut Monitoriza is compatible with operating systems Windows ™ 7, Vista, XP, 2003 and 2008 the only requirement is to have installed .Net Framework 3.5 SP1 Microsoft.

EDITOR

Acimut Monitoriza Editor is one of the three main components of the system, with it we can create, design and modify our scada projects that are then executed through the **Communications Server** and **Client Scada**.

When you create or modify a scada project, using the publisher you can define variables and alarms, create forms to graphically display the values of the variables, store in a database the values of variables, keep a history of alarms, graphically show the variable values stored, write variables on a PLC (or other devices) and manage who can access the project resources.

The **Editor User interface** is what can be seen in the two following illustrations:

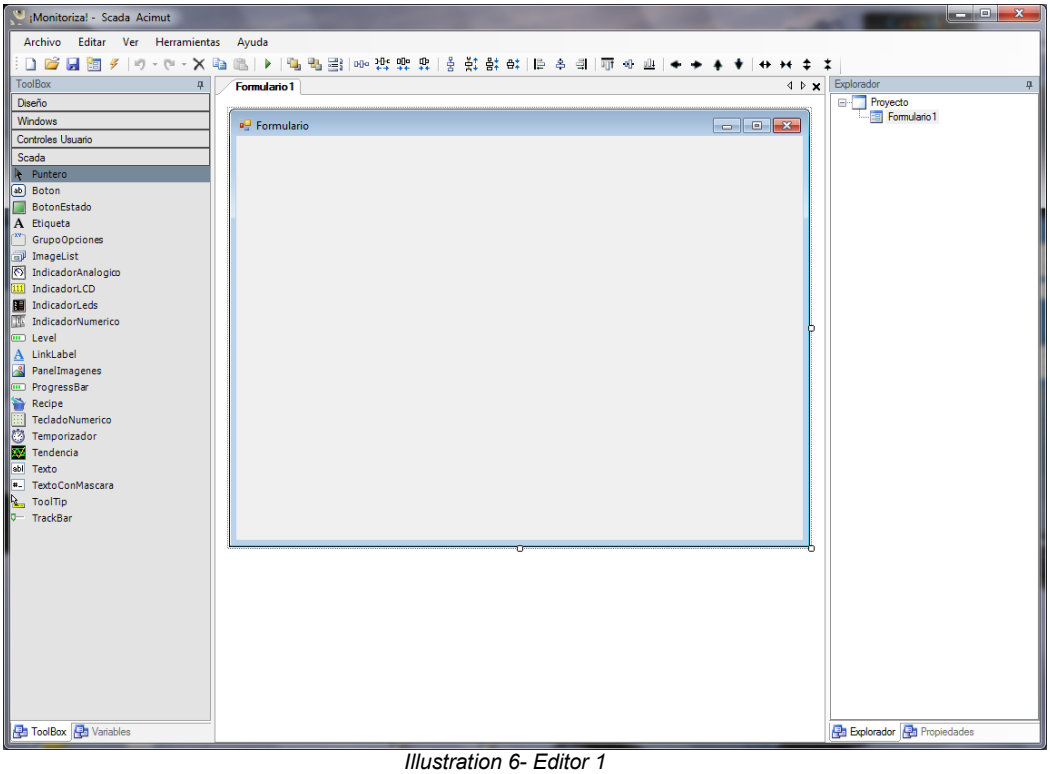


Illustration 6- Editor 1

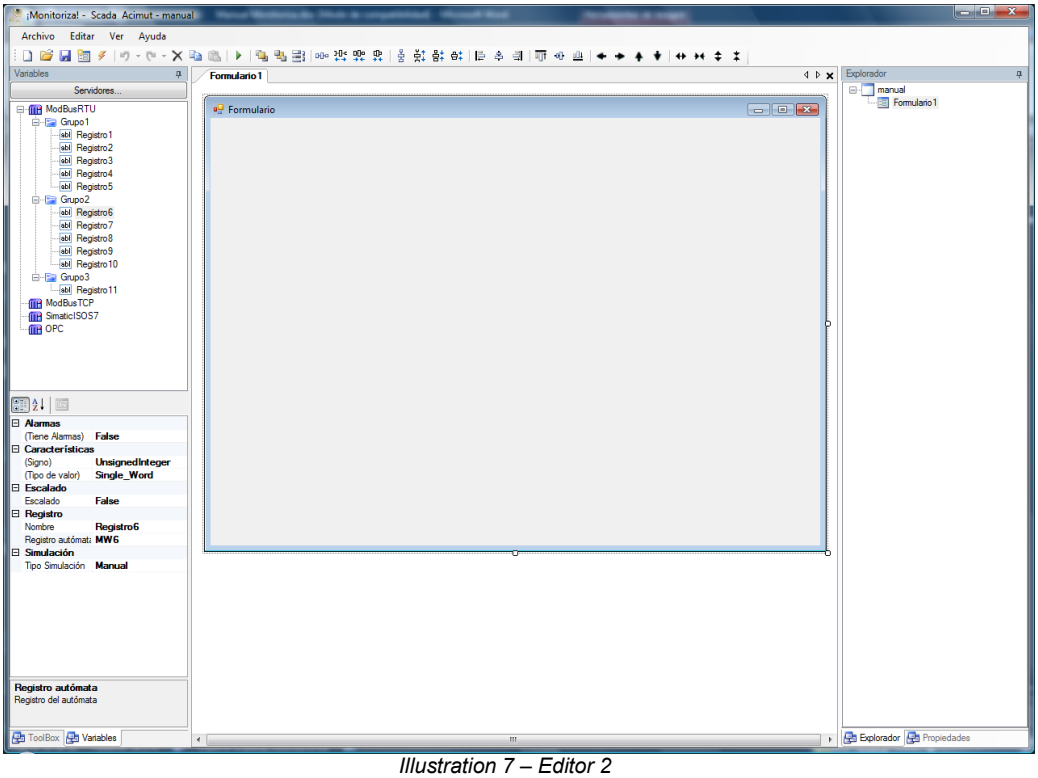


Illustration 7 – Editor 2

Let's see in some detail each part of the user interface.

At the top is the menu bar and toolbar with standard elements such as operations to open or save a project, editing operations (copy, cut, paste ...) and operations for formatting and alignment.

Illustration 8 – Menu bar and toolbar

On the sides are browser windows of the project, the toolbox, the server features and variables and the properties window.

The project explorer window shows the list of forms that conforms our project and with a double click we can display the form in question, we may also use the context menu to perform certain operations on the forms, as illustrated in next image.

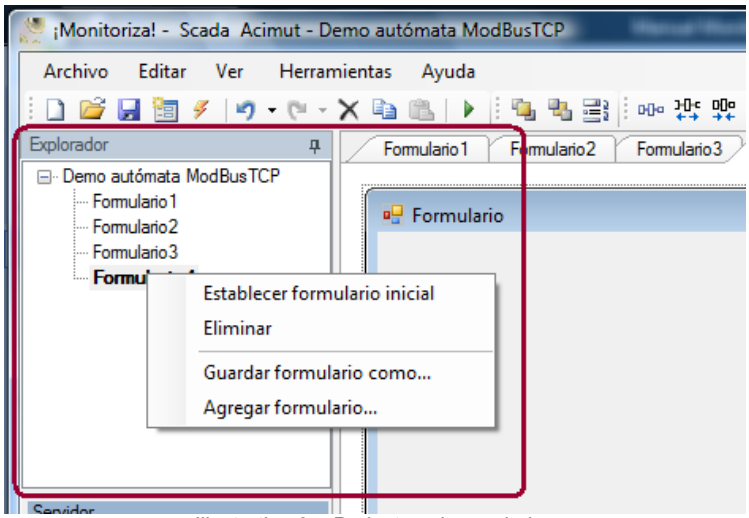


Illustration 9 – Project explorer window

The ToolBox window allows us to add components to forms, with these components we design the forms and give them the desired functionality, or use components that allow us to view or modify the variables or parameters that we want to monitor (Scada tab) or components used to control the flow of application and define its design or appearance (Layout and Windows tab).

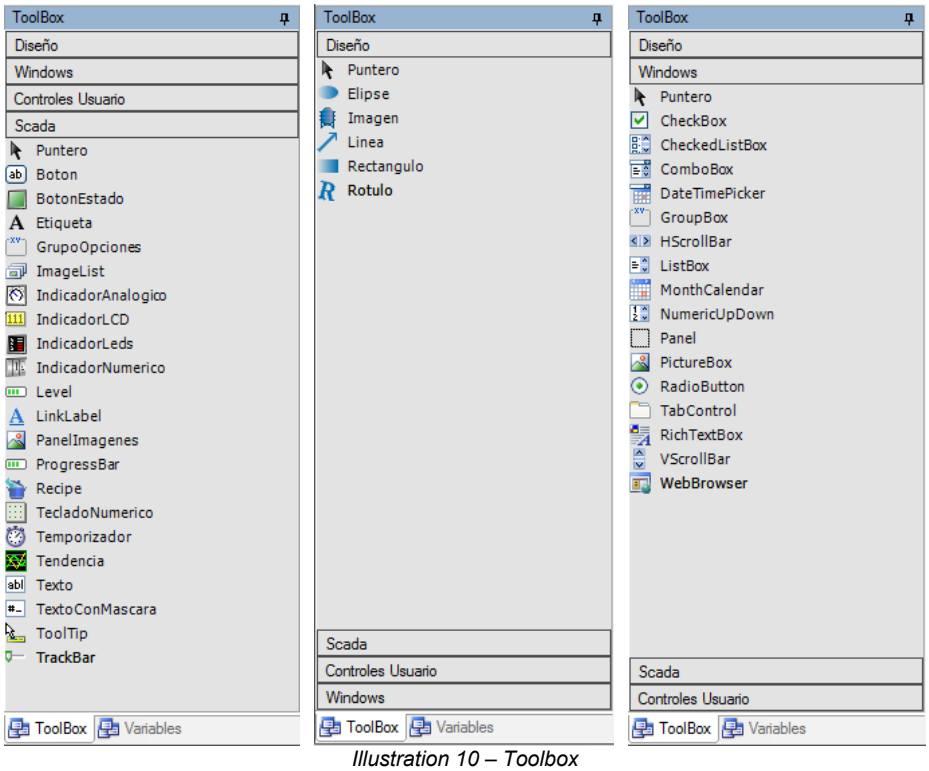


Illustration 10 – Toolbox

The window of features and server variables allows us to define parameters for each communication server that we define in the project, such as the IP address of the PLC with which we communicate and the characteristics of groups and variables defined on each server.

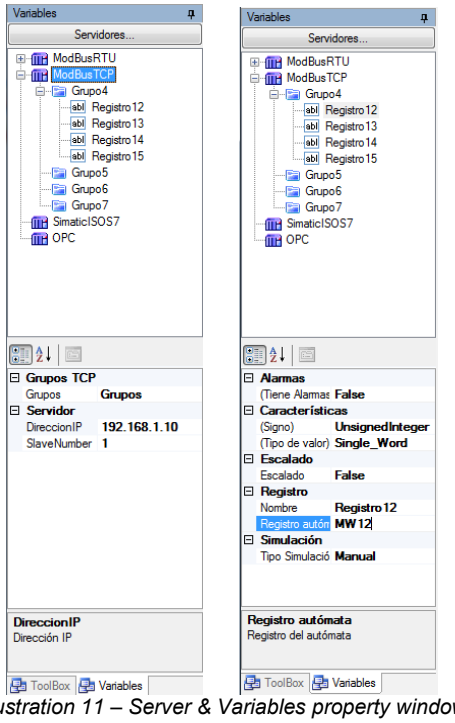


Illustration 11 – Server & Variables property window

In the Properties window we can define the characteristics of each of the components added to the forms, for example in the following figure we see that selecting the IndicadorNumerico component in the properties window will show the characteristics or properties of that component particularly

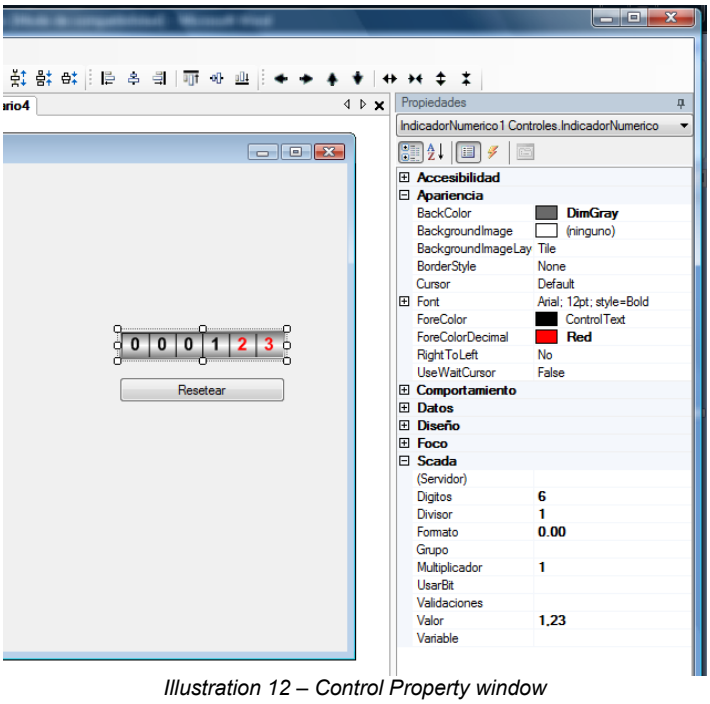


Illustration 12 – Control Property window

Keep in mind that although the windows are arranged on the sides, this layout is configurable, and simply dragging over the title of the window we can position them more conveniently for our work.

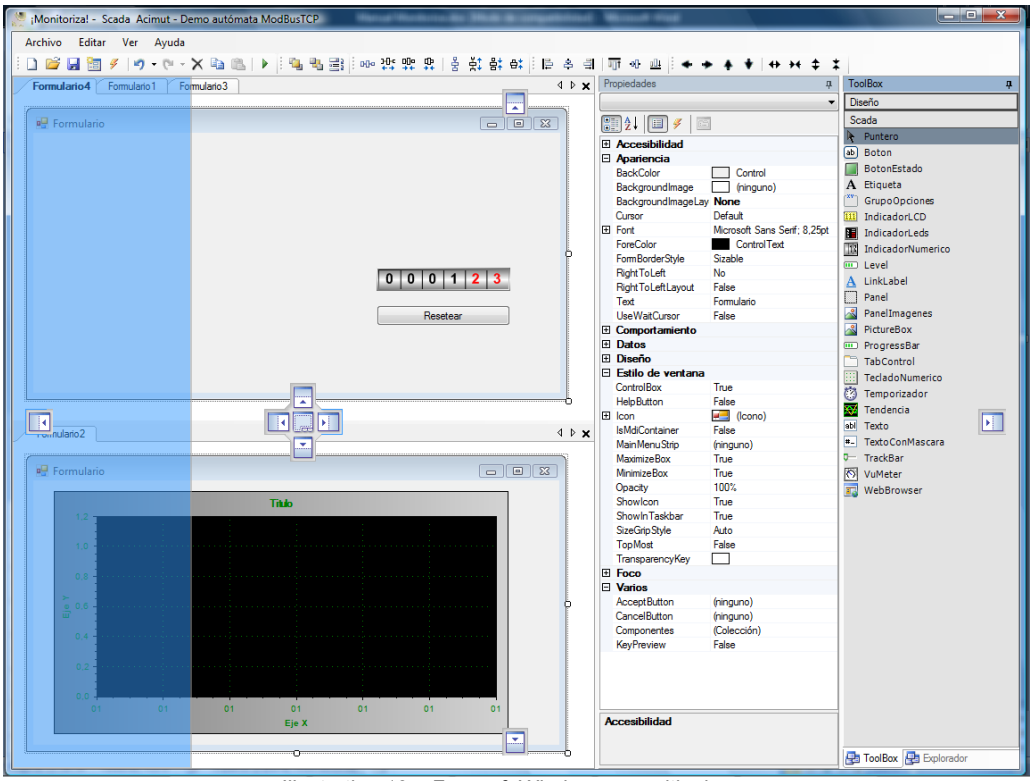


Illustration 13 – Forms & Windows repositioning

Demo: http://www.acimut.com/monitoriza/videos/editor1/default.html

Also, the windows for Variables, Properties, Toolbox and Explorer can set for automatically hide when they lost the focus making the work surface larger, simply clicking on the corresponding button on the window title (marked in red in the illustration below) the window hiddes when lost the focus. To re-view, move the mouse over the tab that represents the window is visible and if we want to stop hiding again press the same button to fix your position. Positioning settings of each window is stored with each project and when we reopen a particular project, the windows are positioned as they were when we saved the project.

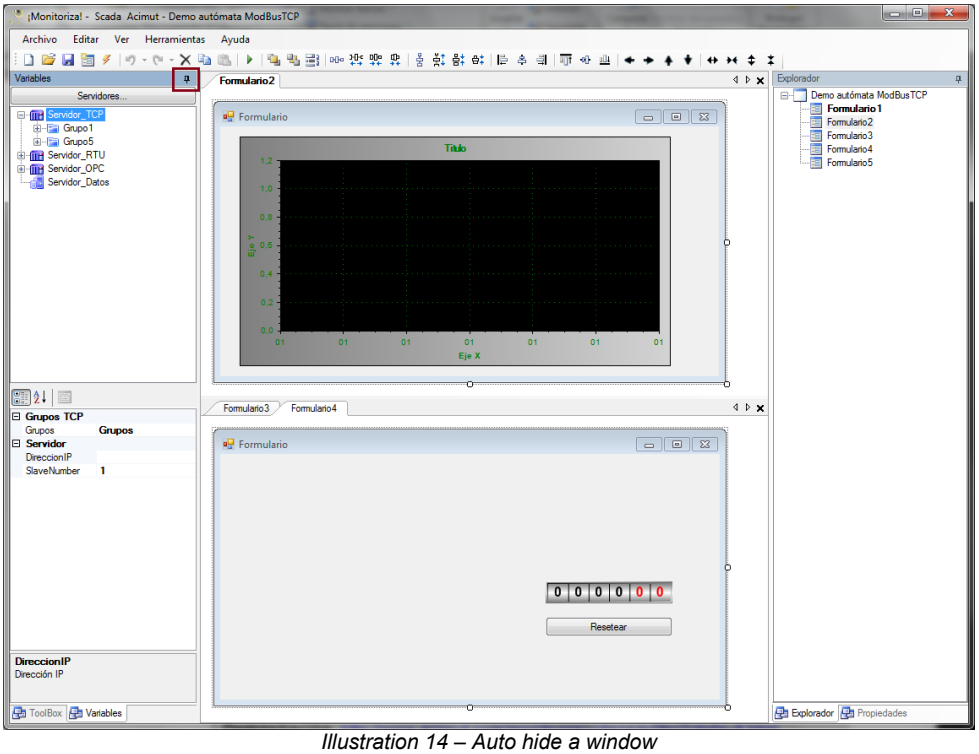
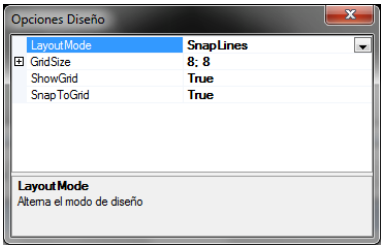


Illustration 14 – Auto hide a window

Demo: http://www.acimut.com/monitoriza/videos/editor2/default.html

In the Tools menu we have a couple of options that help us in design tasks

In this menu we have the *Options* item:



The options of this menu are:

Layout Mode: We can choose between Snaplines (default) that uses the advanced positioning features of Microsoft applications; the other possible value is Grid, in order to use the classical grid, allowing us to set the size between dots using GridSize property.

We can choose the options for the grid using the properties:
ShowGrid: Show or hide the grid in design mode.
SnapToGrid: If we want to snap the controls to the grid or not.

CREATE A PROJECT

STARTING...

To create a project using the Monitoriza editor, at least we must define a communications server with PLC, the variables that we want access to, that will join in groups, and at least one form for designing the user interface of the project.

Additionally we can set alarms that are triggered with the conditions that we define, users and permissions to establish form access to each of the defined user.

In the definition of servers we can also set the database server where we store permanently the values of the variables we monitored.

Here's how to do each task. To start the Monitoriza Editor we will have to start a new project by clicking on the **New Project** option from the **File menu**.

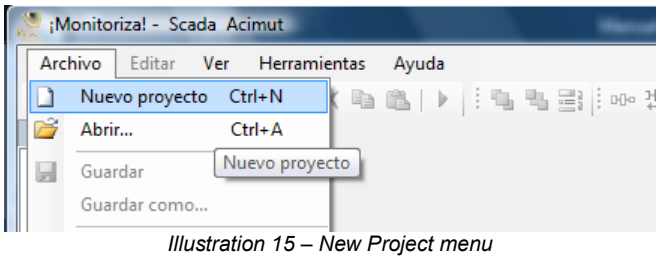


Illustration 15 – New Project menu

Or else clicking the **New Project** button in the toolbar.

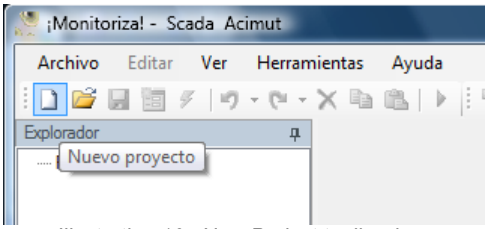


Illustration 16 –New Project toolbar icon

Then we define the communication servers with PLC and databases, either by clicking on the View menu option Servers

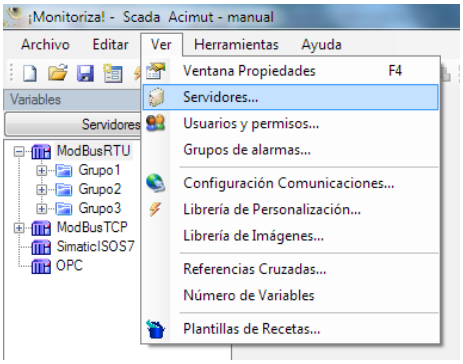


Illustration 17 – Servers Menu

Or Servers button in the Variables window

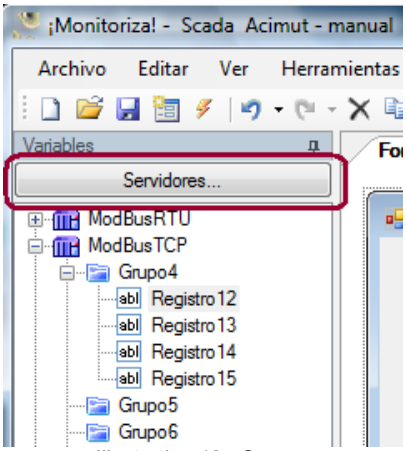


Illustration 18 - Servers

And we will get the dialog for the definition of the servers.

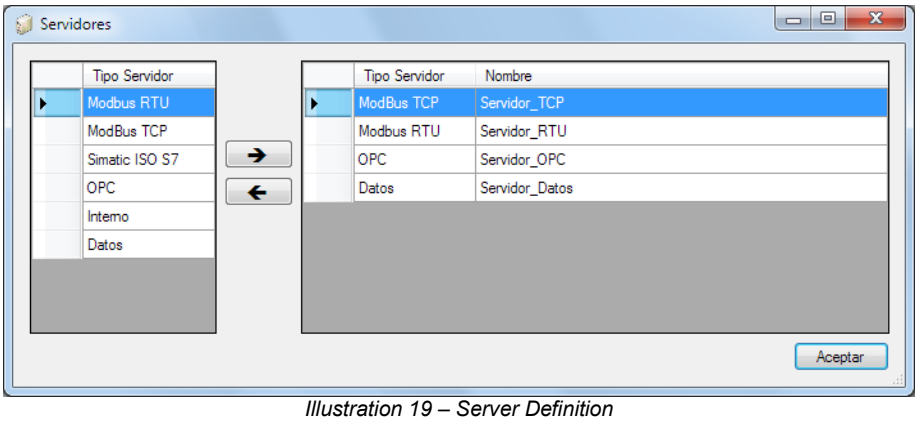


Illustration 19 – Server Definition

Demo: http://www.acimut.com/monitoriza/videos/definir_servidores/default.html

A new project needs at least one server, here is the list of different server types.

Monitoriza have native communication servers with PLCs (Modbus RTU, Modbus TCP and ISO Simatic S7) and third party servers as OPC, this allows you to connect a multitude of devices.

If you want to store a history of variable values and/or a history of alarms that occur in the system, we define one or more database servers in which we will establish the database used. The definition of data servers will be shown in detail in Section Save Database

Another possibility is to define an internal data server (Server Type: Interno), by this server can specify a set of variables that are independent of the variables of the PLC but can serve to store values such as configuration parameters or values calculated.

Each communication servers with the PLC will have different properties as discussed below.

ModBUS RTU

The different properties of a MODBUS RTU server type are shown below.

These properties set the parameters for serial communication and the number of slave to connect to a ModBUS environment. It should be noted that Monitoriza establishes connections ModBUS as a master.

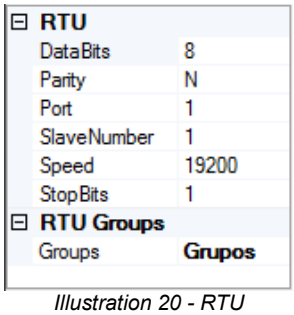


Illustration 20 - RTU

ModBUS TCP

The properties of a MODBUS TCP server type:

Are the IP address of Modbus devices, in an Ethernet environment is needed; and the number of slave to connect to a ModBUS environment. It should be noted that Monitoriza establishes connections ModBUS as a master. The use of slave number allows access gateways.

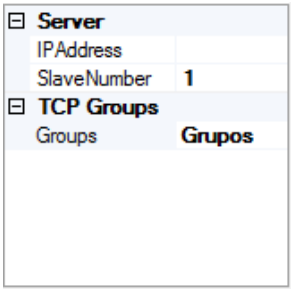


Illustration 21 - TCP

OPC

OPC stands for *OLE for Process Control*, it is a standard defined by Microsoft which aims to unify methods of communication with all types of devices because of its good acceptance by various manufacturers, we can say that there is an appropriate OPC server for any device that is in the market.

The properties that allow us to connect to an OPC server are as follows:

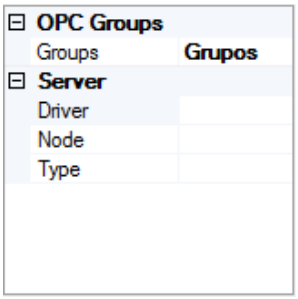


Illustration 22 - OPC

We begin by describing the last of these properties, the Type, this is the name that the manufacturer has given its OPC server, a dropdown will allow us to choose between the OPC server installed on your machine or in a computer in our network using Node (you must have the proper permissions, it is operating system dependent, there is abundant documentation on this subject that its outside the scope of this manual). The property is vendor-specific driver, we must consult the documentation for each OPC server, is a prefix that usually defines how we communicate with the PLC and its direction and characteristics.

In the next chart we can see some examples.

Fabricante	***Tipo***	***Driver***	Variable
Siemens S7-200	S7200.OPCServer	192.168.100.120:1200:1001,	VW1000,INT,RW
Telemecanique	Schneider-Aut.OFS.2	UNTLW01:0.254.0!	%MW2000
		MBS03:1/T!	%MW2000
		XIP01:192.168.1.2.10.2!	%MW2000

Simatic ISO S7

The properties of a server type Simatic S7 ISO are:

Are the Simatic device IP address, the number of CPU Rack (usually 0), and the slot in the Rack, in which lies the CPU (usually the 2).

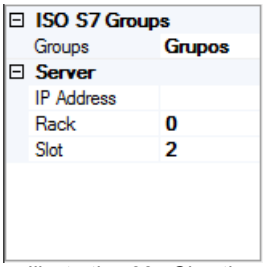
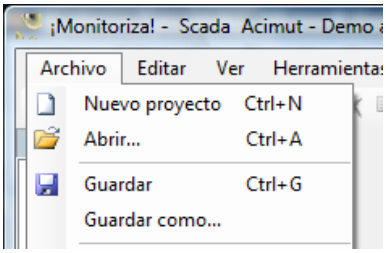


Illustration 23 - Simatic

OPEN AND SAVING PROJECTS

On the File menu options are typical Open ..., Save and Save As ... to allow us to open previously saved projects and save the changes that we make to our projects. These operations can also be done from the toolbar.



CREATE VARIABLES

The variables in Acimut Monitoriza are the basic element of information, with them we set the bond between the tags and records of the PLC and the various components of our project, enabling the visualization and modification of the different elements that we monitored.

The variables are ultimately what we will be measuring or monitoring, including temperature, pressure, voltage, open or closed status of any of the devices of our process.

Each Monitoriza variable is defined by three different elements:

The **communications server**, which refers to the PLC that manages and controls the device or devices on which we operate or control.

The **group**, which is a grouping of variables depending on the characteristics of these, for example by combining the variables that we are only interested to read in a group and others we want to read and write in another, or if we have a group of variables we must read with a different cadence from other variables grouped into different groups. It is also important to consider the communication protocols, so, when we read a set of records, such records in contiguous PLC memory positions, reading, either by an OPC server or a server ModBUS, is more efficient if we put the adjacent records in the same group and will improve the performance of our SCADA.

Name is the symbolic name we give to record the PLC.

Therefore to create variables once we have established communications servers as we saw in the previous chapter, we must create the groups to which the variables will belong

To do this, in the server window click on the three points of the Groups property

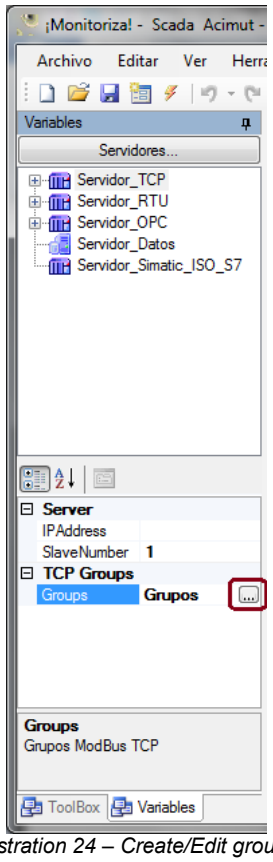


Illustration 24 – Create/Edit groups

Depending on the type of server, group characteristics are similar but with some difference, so we'll see each in detail.

Groups ModBUS TCP and ModBUS RTU

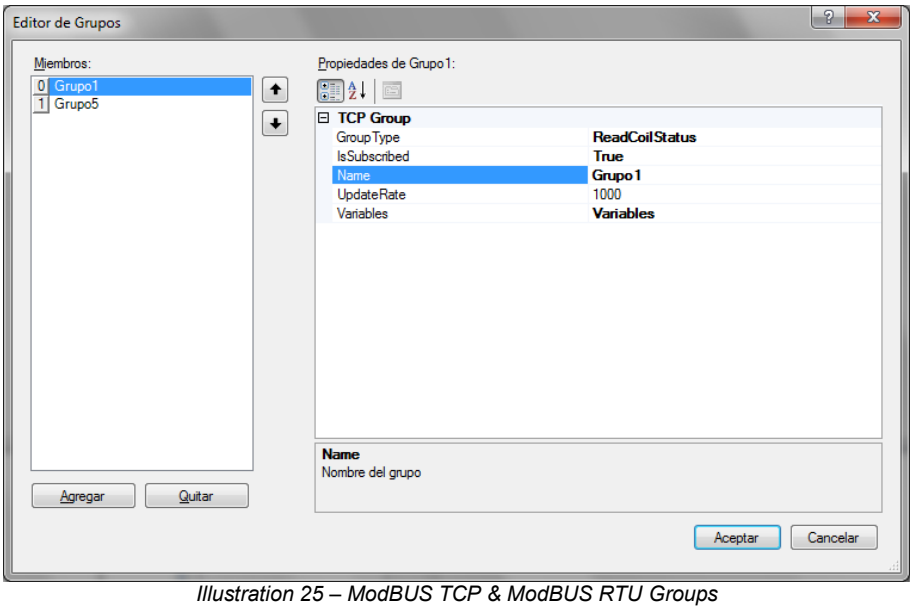


Illustration 25 – ModBUS TCP & ModBUS RTU Groups

The properties that define a group of a MODBUS TCP server are:

- **GroupType:** Group type, indicates whether the variables are of a register type or bit. Select ReadHoldingRegisters when the variables are read / write register (a word, 2 bytes), and ReadCoilStatus type for bit variables .
- **IsSubscribed:** Indicates that this group of variables is read continuously. If we define a group with only variables that will be written and not will be readed we can leave this property to False value.
- **Name:** Is a text describing the contens of the Group, for organization purposes is better to set a descriptive text.
- **UpdateRate:** In case of IsSuscribed is set to True this property indicates the rate in milliseconds for reading the contents of a group. Is appropriate to adapt the value of this property to the communication type and protocol.
- **Variables:** Is the property through which we can enlist the variables used. Will see this property in more detail later.

OPC Group

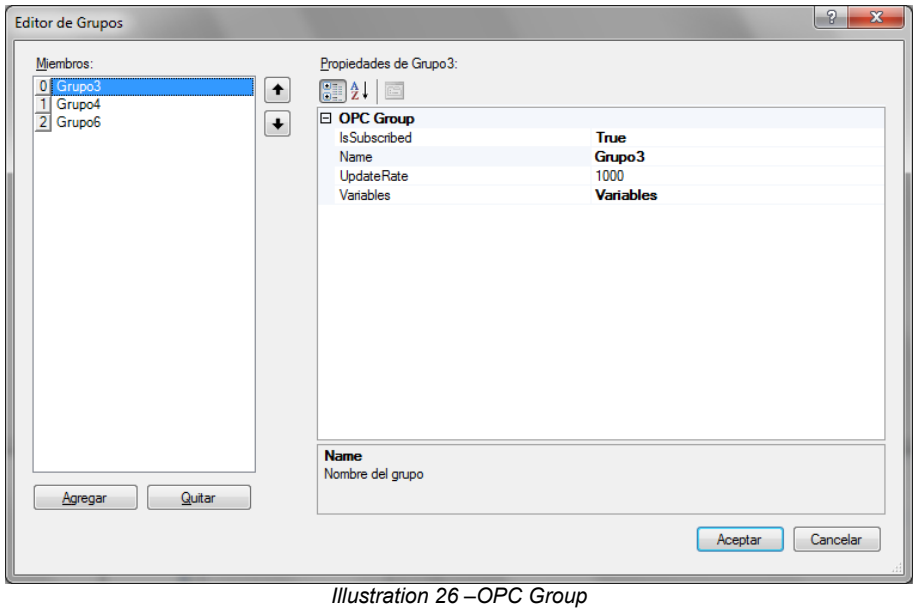


Illustration 26 –OPC Group

The properties that define a group of an OPC server are:

- **IsSubscribed:** Indicates that this group of variables will be read continuously. If we define a group with only variables that will be written and read is not needed, leave this property to False.
- **Name:** Name of the group and should be as descriptive as possible for organizational reasons.
- **UpdateRate:** If is suscribed property is True, will indicate the rate for reading in milliseconds for each group. It is advisable to adpat this value to the type of communication available.
- **Variables:** The property through which we can enlist the variables used.

Once we have defined the group we can define variables or records, to do it in the same window, we have the property **Variables** in which, if we click on the three-point button property, appears the definition window for variables or registers.

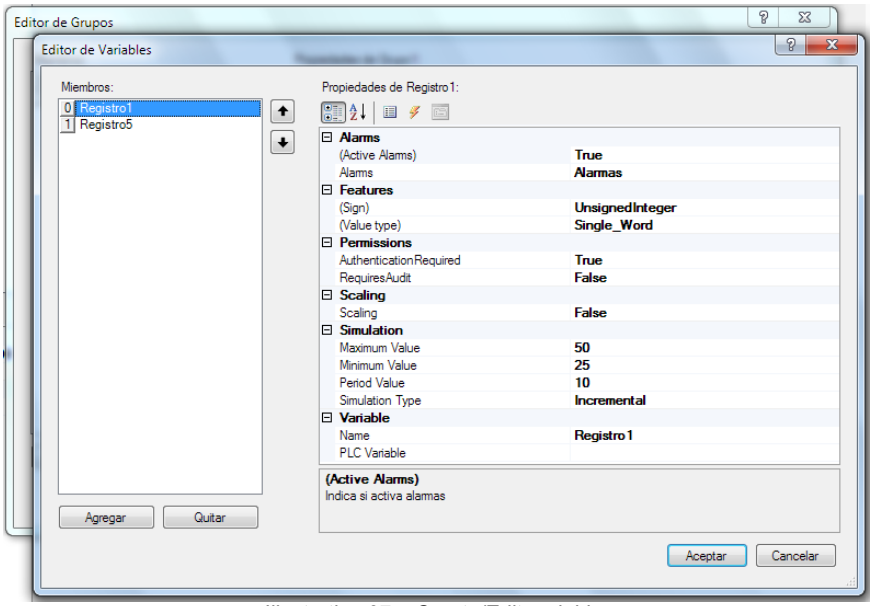


Illustration 27 – Create/Edit variables

This has the following properties:

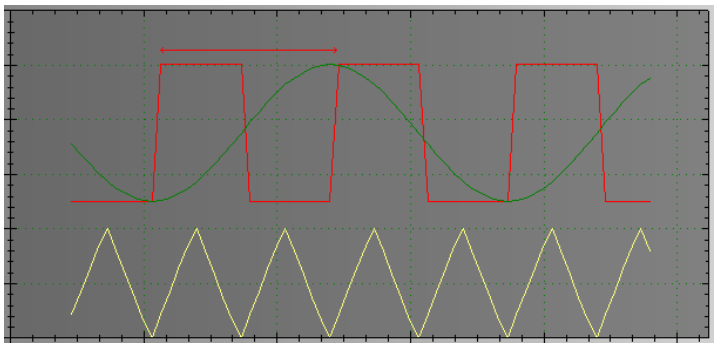
- **Active Alarms:** Indicates whether to activate alarms for that variable. We show in more detail in the next chapter.
- **Value Type:** Sets the size of the variable. The options are: Single_Word (2 bytes) and Double_Word (4 bytes)
- **Sign:** Defines whether the variables are considered signed or unsigned. The options are: UnsignedInteger - unsigned integer and SignedInteger - signed integer
- **Name:** The symbolic name we give to the variable. Must consist of alphanumeric characters and contain neither spaces nor symbols.
- **PLC Variable:** Identifies the PLC register and must follow the rules and conventions for the selected plc type.
- **Permissions:** Properties **RequiresAudit** and **AuthenticationRequired** specify whether a user must authenticate to be able to change a value and if you keep a record of changes respectively. These properties are detailed in the chapter Users and Permissions.
- **Scaling:** Indicates that the value of this register is scaled, i.e. we can specify the maximum and minimum values for scaling analog values, for example, if we have a record that will give us the input value of a probe, and on the one hand we have a signal of 12 bits and the other a probe 4-20 mA. measuring temperatures between -40 ° C to 60 ° C we should configure these properties this way:

Scaling	
Maximum Analogic Value	4095
Maximum Scaling Value	60
Minimum Analogic Value	0
Minimum Scaling Value	-40
Scaling	True

By this way, we have the value scaled, keep in mind that scaled value is used for alarms, recipes, etc. But values in simulation aren't scaled. A lineal scaling is used.

- **Simulation Type:** When 'running' a project in design mode, this option allows you to automatically change the value of the variables. If you choose any type that not is manual simulation we'll see the following three properties.
- **Maximum value:** Is the maximum value that the simulation system will give to the variable.
- **Minimum value:** Is the minimum value that the simulation system will give to the variable.
- **Period value:** It is the time, in seconds, of each cycle or state, will influence the changing speed of the value of the variable.

There are simulations for random, incremental, decremental, Square Wave, Sine Wave and Triangular Wave. We see in the picture below an example of waves generated by monitoriza and the signal period for the square wave.



Value Type property and **Sing** property only appear when the group is MODBUS TCP or MODBUS RTU and the kind of group can't be defined as ReadCoilStatus since this type involves a bit of a internal coil.

Demo: http://www.acimut.com/monitoriza/videos/crear_variables/default.html

Once we have created the groups and variables can also edit them by selecting the corresponding variable or group of variables in the tree at the top of the window variables and editing their properties in the lower pane of the window.

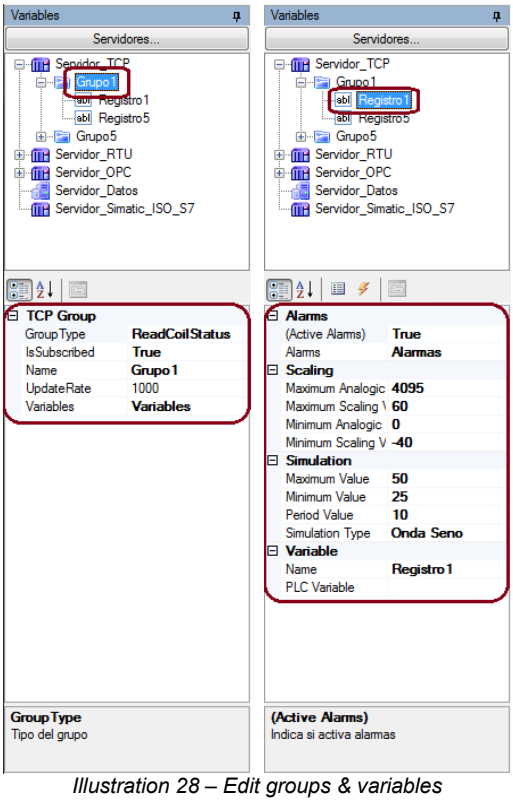


Illustration 28 – Edit groups & variables

Variables Event

Once we have defined a variable we can set the alarms associated with that variable as discussed in the next chapter (alarms) or control the *valueChanged* event of the variable.

By both methods we are able to perform actions when the value of a variable changes.

We will use the *alarm* method when we want to get a notification for a certain condition for the value of the variable and will use a *valueChanged* event when we want to take an action simply because the value has changed.

To set the *valueChanged* event we must click on the *Events button* properties window for the variable (marked in red in figure).

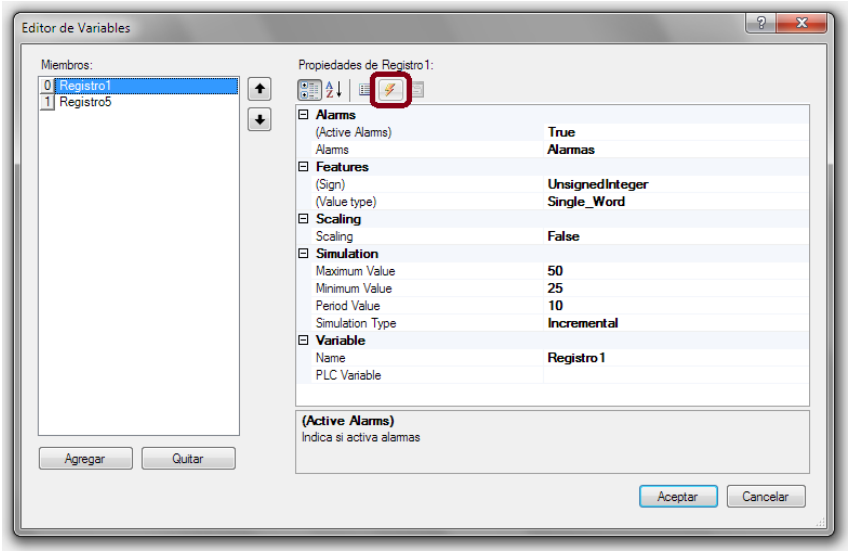


Illustration 29 – Activate event window

Once in the event window we must click the three dots button.

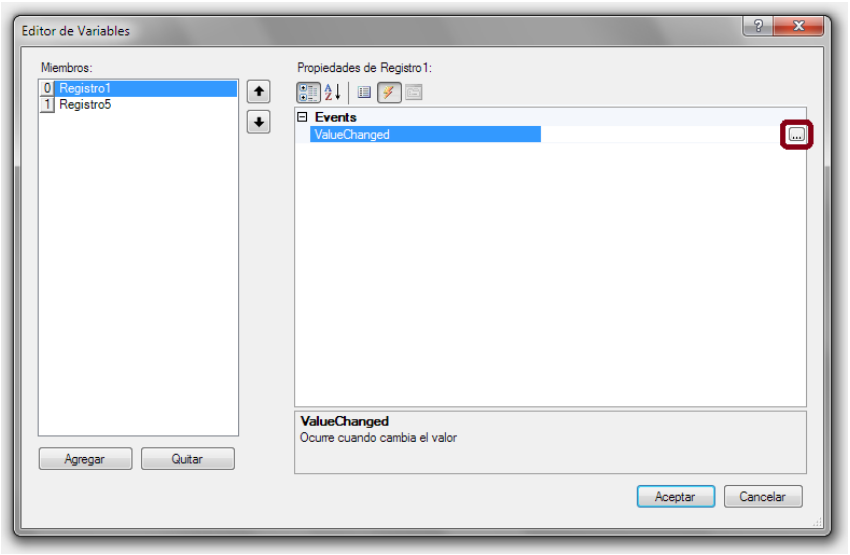


Illustration 30 – Variable event

A code editing Windows will be displayed in order to set the action to perform when the value of the variable changes.

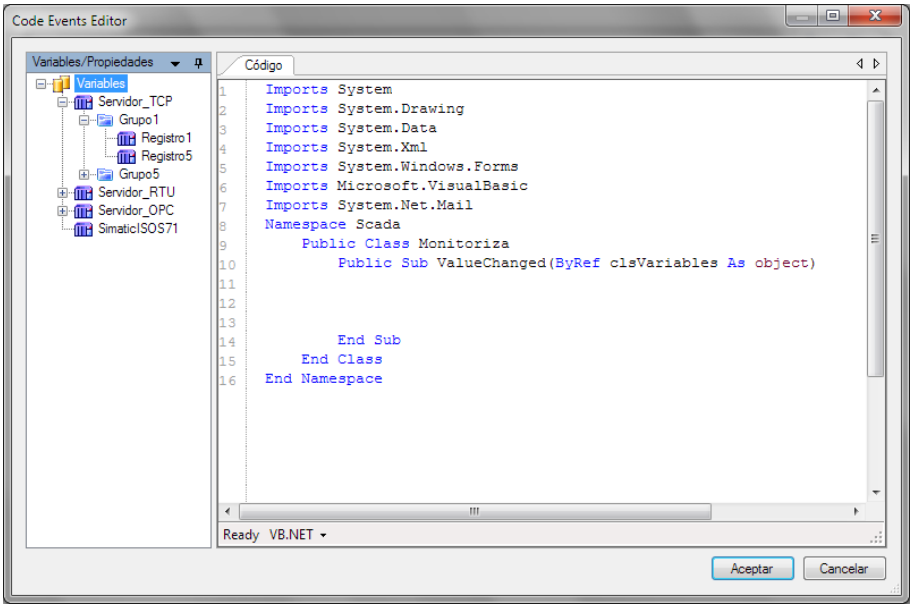


Illustration 31 – Code Events Editor

Our code will run on the communications server of Monitoriza therefore we have access to all resources on the server where is installed.

ALARMS

Create Alarms

Whenever we want the Scada system notifies us of a certain condition we must set an alarm. These alarms can check for example if a temperature is above or below a certain value, the pressure has reached a value or any other aspect that we want to consider.

Alarms are associated with variables and it is in the variable definition window where we establish if we bind a variable to an alarm. To do this, when creating or editing a variable we must set to True the property **Active Alarms**, then the **Alarms** property will appear and by hitting the button with three points we can define the alarms associated to the variable.

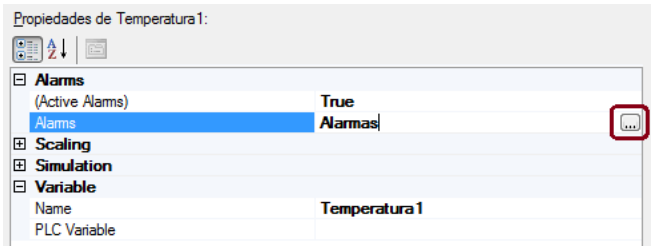


Illustration 32 – Alarm creation

Associated with a variable we can define one or more alarms depending on the conditions we want to control.

We must set the message we want to show when defining the alarm, if we store the event to keep track of which has occurred or if a user needs to validate the alarm, and the conditions in which an alarm should be fired.

We can make that the evaluation of the alarms should be by comparison of the variable in which the alarm is set with a constant value (eg the temperature is less than 40) or we may be interested to compare the variable that is defined in the Alarm system with another variable (eg if you are monitoring the production of various products can be stored in a variable that you want to fire the alarm temperatures depending on the product we are producing). Set the property to Constant or Variable TypeValue respectively for either case.

Additionally we can define a group of alarms using the AlarmGroup property, this group of alarms is used to specify user-level permissions, or define the users that will see that alarms or if they can validate or not as seen in the chapter Users and Permissions.

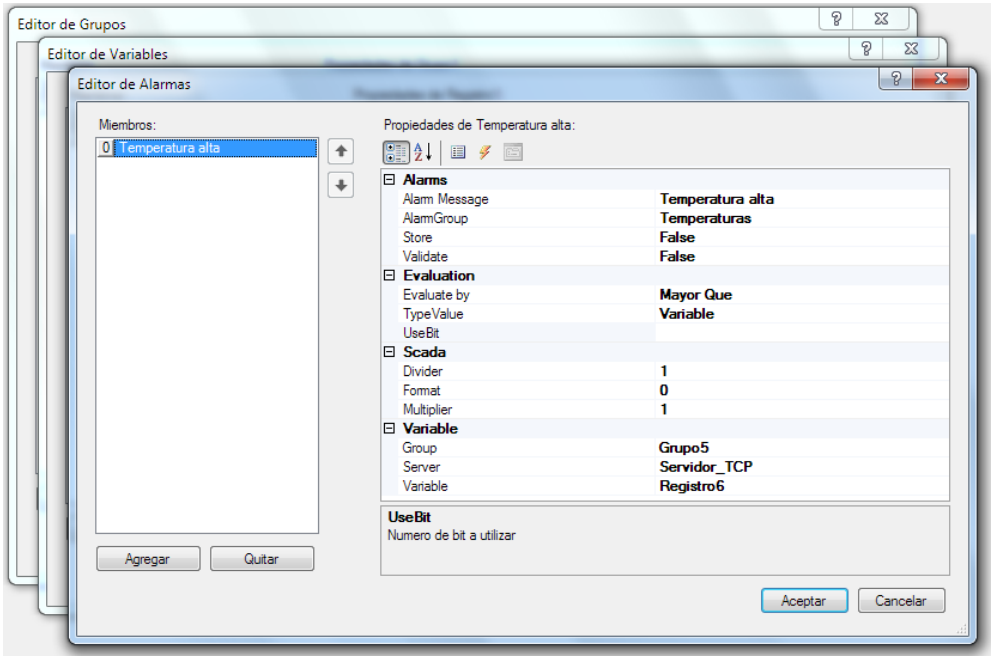


Illustration 33 – Alarms property window

Let's look in detail by examining each of the properties of the alarm.

- **Store:** Defines if the alarm will be stored. In order to store the alarms we must to define an additional database server with the property Alarms established.
- **Alarm Message:** Alarm message displayed when activated.
- **AlarmGroup:** Permission group linked to the alarm.
- **Validate:** If set to True, will require a user to validate in order to hide it in the window display when no alarm is active. For a properly work in validation it is necessary to store the alarm. We will see how to define the database, table and fields in the Save Database point.
- **Evaluate by:** Sets the comparison condition that determines when the alarm is active. The possible values are: *Equal, MoreThan, LessThan and Distinct.*
- **TypeValue:** The possible values are constant and variable. *Constant* is used if the EvaluateBy property is to compare a absolute value, this value is defined in the *value* property. If we want to fire the alarm comparing whith another variable in the system we must to set the TypeValue property to Variable and the properties Server, Variable and Group will be used to set the variable used in the comparation.
- **Value:** Set the value to apply the condition evaluation for the alarm. For example if we are monitoring the temperature and we define the property value to 40 and assesses property for a GreaterThan, we activate the alarm if the temperature value exceeds the value of 40.

- **UseBit:** It allows us to use in isolation of the bits of the variable in the assessment given (first bit is 0). Clearly, in this case, we can only compare with the values 0 and 1.
- **Server, Group and Variable:** Set the system variable that we will use for comparison when the TypeValue is set to Variable.

Scada property group allows us to specify the variable value for storage in the database and therefore for display in the Alarm Viewer.

- **Divider:** Display the variable value divided by the number indicated in the property.
- **Multiplier:** Display the variable value multiplied by the number indicated in the property.
- **Format:** Change the appearance of the value of the variable, for example a format equal to # # .00 variable will cause the display with two decimal places.

We can define alarms groups when we are defining the alarm using the option New of the **AlarmGroup** property or using the command Alarm Groups in the View menu, which shows us this form

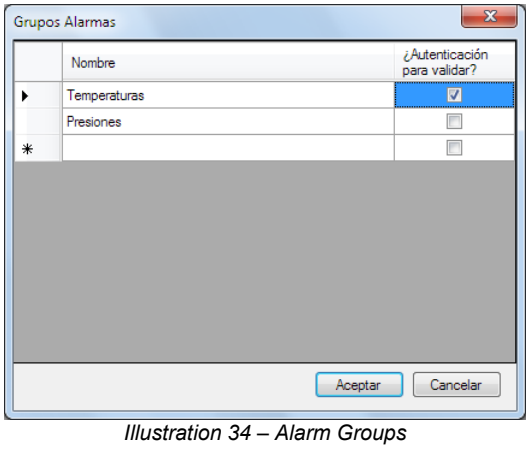


Illustration 34 – Alarm Groups

In which we can not only specify the name of the alarm group but change it if we have been mistaken at the creation, set if the user is to be required to authenticate entering username and password every time it wants to validate an alarm belonging to this group.

As you will see in the chapter USERS & PERMISSIONS there are project overall parameters that will affect this behaviour.

Demo: http://www.acimut.com/monitoriza/videos/crear_alarmas/default.html

Alarms Viewer

The alarm display is the centerpiece of notification and validation of alarms that occur in SCADA project implementation.

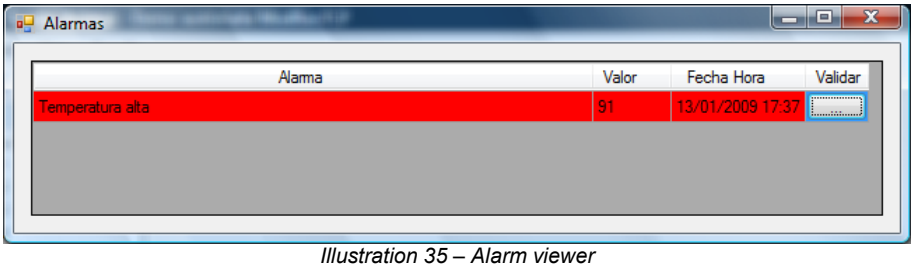
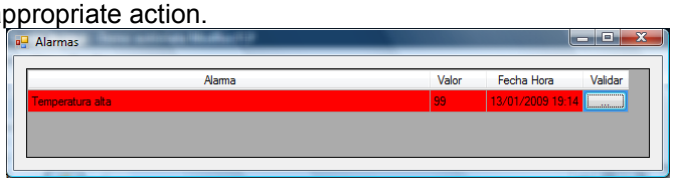


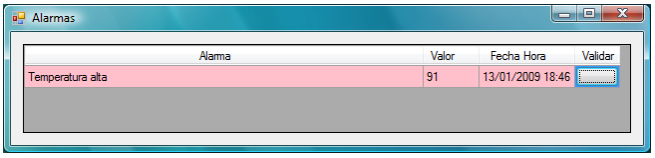
Illustration 35 – Alarm viewer

Whenever an alarm fires, because the conditions stablished in design have been fullfilled, the display shows alarm, the possible states of the alarm, depending on if you have set as you need validation or not and these states are represented by different colors.

- **Red**: Indicates that the alarm is active and requires validation, that is, we must click on the Validate button to indicate that a user has ignored the alarm and took the appropriate action.

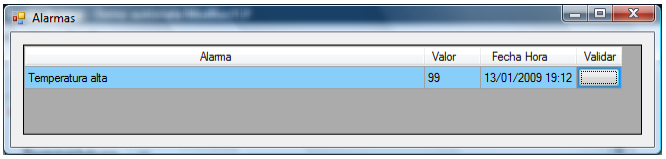


- **Pink**: Indicates that the alarm is active and has been validated, i.e. a user has clicked on the SCADA Validate button.



- **Yellow**: Indicates that the alarm is no longer active but has not yet been validated.

- **Blue**: Indicates that the alarm is active and does not require validation. In this case not requires validation and disappears at the moment the alarm is no longer active.



If an alarm requires validation, depending on the alarm group to which he belongs, may be necessary to re-authenticate, that is, you may be prompted again the username and password.

This will look in more detail in the chapter of Users and permissions. Its function is to reinforce security and that only authenticated users can validate the alarm.

Demo: http://www.acimut.com/monitoriza/videos/validar_alarma/default.html

Alarm Events

Alarm Events is one of the more important extensibility elements in Monitoriza.

Using the Alarm Viewer, described in the previous section, we have seen that we have the opportunity to be notified whenever an alarm occurs and we can validate it, if it is defined that way, so the alarm will appear until the alarm is validated even if the alarm condition no longer occurs.

The problem is that the Alarm Viewer notify us via a popup window, in the operation post, that the alarm has occurred, and therefore if you are not controlling the screen not we realize the alarm.

To solve this issue we have the alarm events, that can be defined as procedures that are triggered in the server of Monitoriza, whenever an alarm is initiated, completed or validated.

The typical use of these events is to send a notification that the alarm has entered into a certain state, for example by sending an email or SMS message. Thus, if we are controlling, for example, temperature, and this exceeds a certain value we can send a message to a particular phone or an email message indicating that the temperature has exceeded the preset value.

While this is a typical use, the truth is that we can do a wide range of actions when an alarm is set (launch a process in the server, or stop it, record in a database, invoke a service external to our system ...) so how Monitoriza allows you to define procedures to be fired when the alarm starts, ends or is validated is programming the procedure in C # or VB.NET.

Let's see how to do this:

From the Alarms Editor window, we must select the alarm to which we assign events

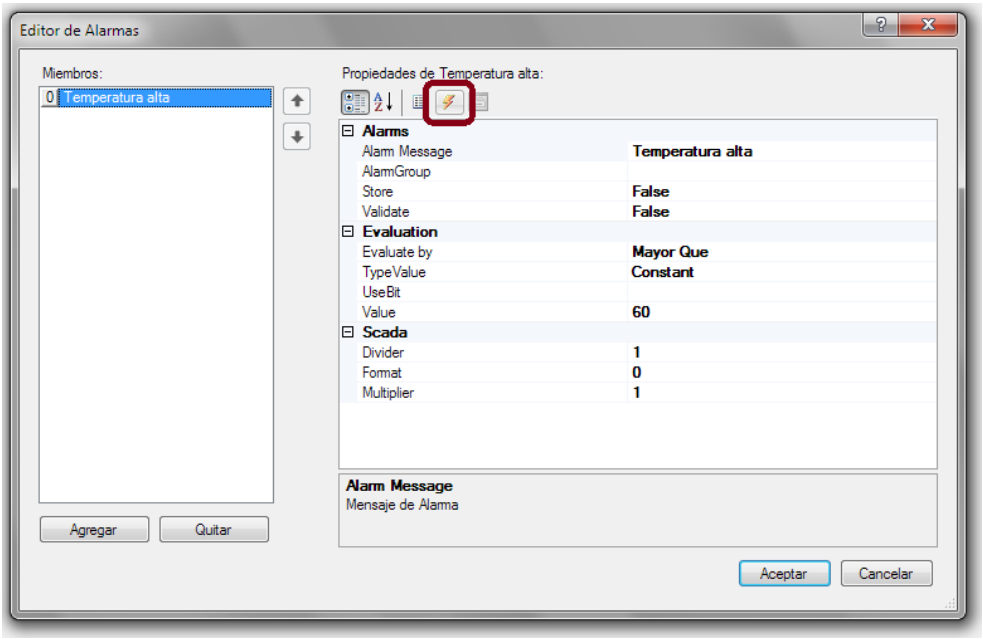


Illustration 36 – Alarms Editor

Clicking the Events button (in red in the picture) will show the next window

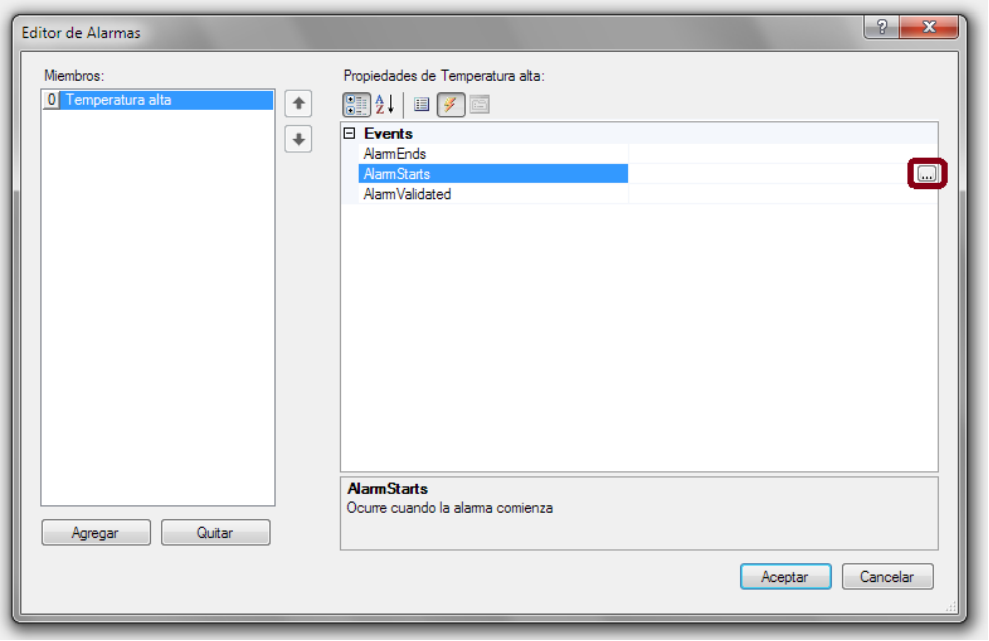


Illustration 37 – Alarms Events

In this Window are three events: AlarmStarts, AlarmEnd and AlarmValidated, that are the events that will be fired when an alarm situation is started, ended or validated.

We can bind code to one event or to all. For example, to bind code to the event AlarmStarts we must click over the three dots button (in red in the illustration) in order to open the code editor for the event.

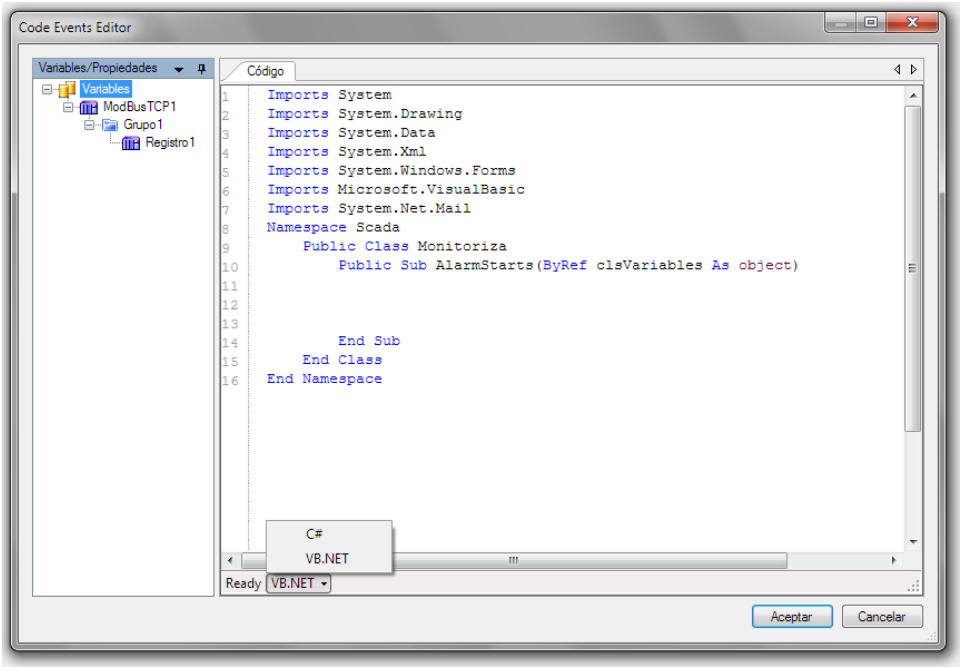


Illustration 38 – Code Events Editor

In the Code Editor of Events, we can write code in both C # and in VB.NET, by simply selecting the appropriate menu.

Left panel shows the list of servers, groups and variables that we have defined, having the ability to drag and drop any of the variables on the editor window in order to pick up their value or change it.

Here's an example of code used to send an email using a gmail.com account. In the example we can see that consists in creating a SmtpClient object to set the connection to the mail server, a MailMessage object that allows us to identify the sender, recipients, subject and message body. In the example we use the value of the variable `Registro1` to compose the message body in the property `MyMailMessage.Body`.

```
Imports System
Imports System.Drawing
Imports System.Data
Imports System.Xml
Imports System.Windows.Forms
Imports Microsoft.VisualBasic
Imports System.Net.Mail
Namespace Scada
    Public Class Monitoriza
        Public Sub AlarmStarts(ByRef clsVariables As object)
           Try
                'Start by creating a mail message object
                Dim MyMailMessage As New MailMessage()

                'From requires an instance of the MailAddress type
                MyMailMessage.From = New MailAddress("remitente@gmail.com")

                'To is a collection of MailAddress types
                MyMailMessage.To.Add("destinatario@dominio.com")

                MyMailMessage.Subject = " Start Alarm Grupo1, Registro1"

                Dim strBody As String
                strBody = String.Format("{0}. The variable Grupo1, Registro1 has
reached the value {1}", Now(), clsVariables.Variables( "ModBusTCP1", "Grupo1",
"Registro1", ""))

                MyMailMessage.Body = strBody

                'Create the SmtpClient object and specifies the user's credentials
                Dim SMTPServer As New SmtpClient("smtp.gmail.com")
                SMTPServer.Port = 587
                SMTPServer.Credentials = New
System.Net.NetworkCredential("usuario@gmail.com", "contraseña")
                SMTPServer.EnableSsl = True

                'Send the message
                Try
                  SMTPServer.Send(MyMailMessage)
                Catch ex As SmtpException
                   MessageBox.Show("Ex1=" & ex.Message)
                End Try
            Catch ex2 As SmtpException
               MessageBox.Show("Ex2=" & ex2.Message)
            End Try
        End Sub
    End Class
End Namespace
```

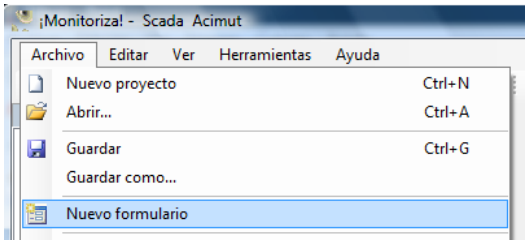
CREATE FORMS

In Acimut Monitoriza Scada to create a project the minimum needed is to define the **communication servers** with PLC and the **user interface.**

To create the **user interface** we can design as many forms as we need for our project and for this we will add any of the controls that are displayed in the Toolbox window, in any of the tabs: Scada, Design, Windows and User Controls. Also, we can add controls by dragging, directly from the Variables window, the desired variable as we can see in a later example.

To add a form to a project, click the button on the toolbar shown below.

Or select **New Form** from the **File Menu**



Now, the Monitoriza Editor looks like:

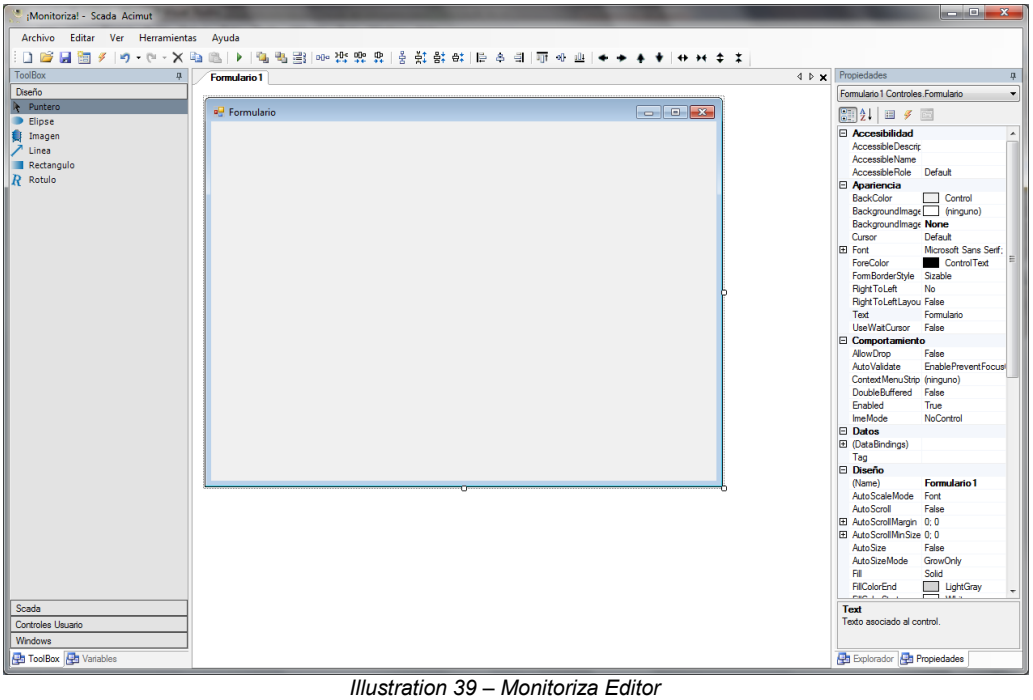


Illustration 39 – Monitoriza Editor

Now, we have a new form and its properties on the right.

Also note that now we have enabled the editing tools in the toolbar.

Additional forms will be created in new tabs, and also in the Explorer tree on the right. Closing a tab does not involve deleting a form.

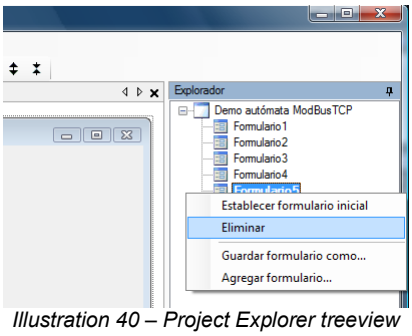


Illustration 40 – Project Explorer treeview

The Project Explorer allows, with double click, open a form and using a contextual menu Delete, Export and Import forms.

Demo: http://www.acimut.com/monitoriza/videos/crear_formularios/default.html

As mentioned above, to add controls to the form, we can drag them from the ToolBox window, as shown int the next illustration.

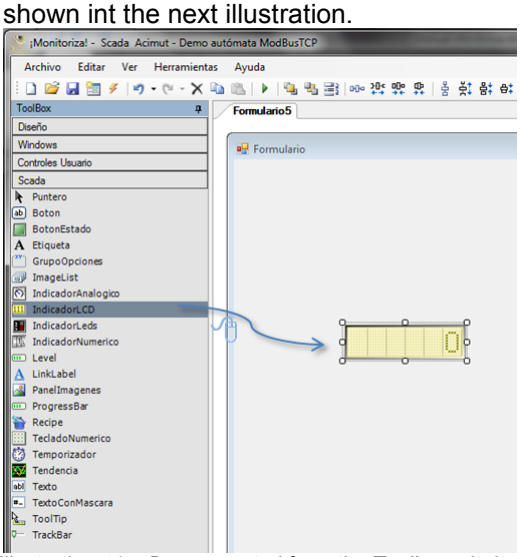


Illustration 41 – Drag a control from the Toolbox window

Or drag a variable from the Variables window, as shown in the illustration.

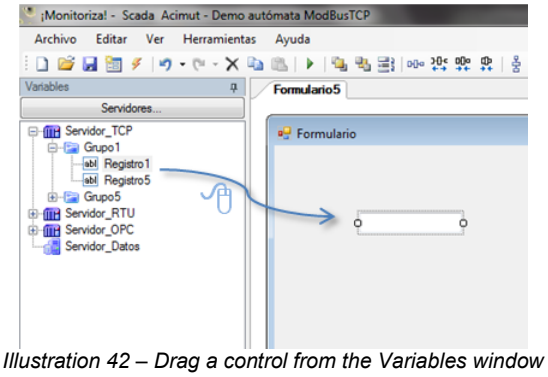
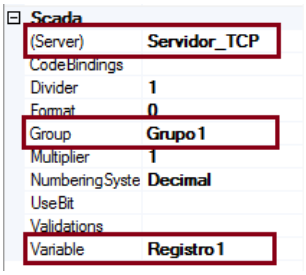


Illustration 42 – Drag a control from the Variables window

Dragging a variable has the advantage that in the same operation we are doing two actions, on the one hand we are designing the form by setting the necessary controls and on the other hand we are assigning directly the Server, Group and Variable properties.



By default, dragging a variable, creates a Text control, but clicking on the variable name we can change the control type associated to the variable, choosing the appropriate type from the dropdown list that appears.

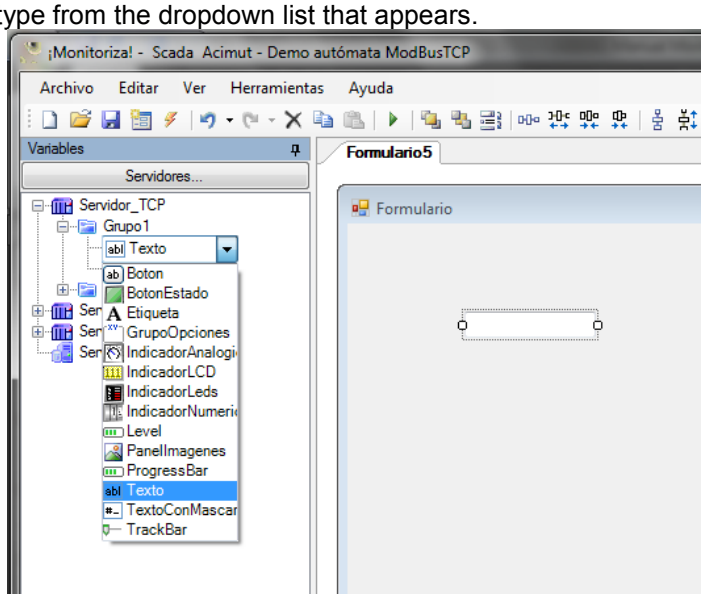
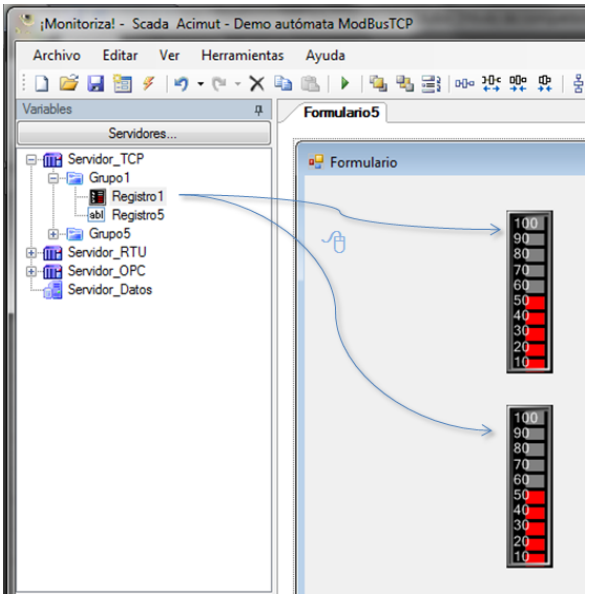


Illustration 43 – List of controls that can be used with variables.

The list of controls shown has all the Acimut's Monitoriza controls that can display or edit the value of a variable.

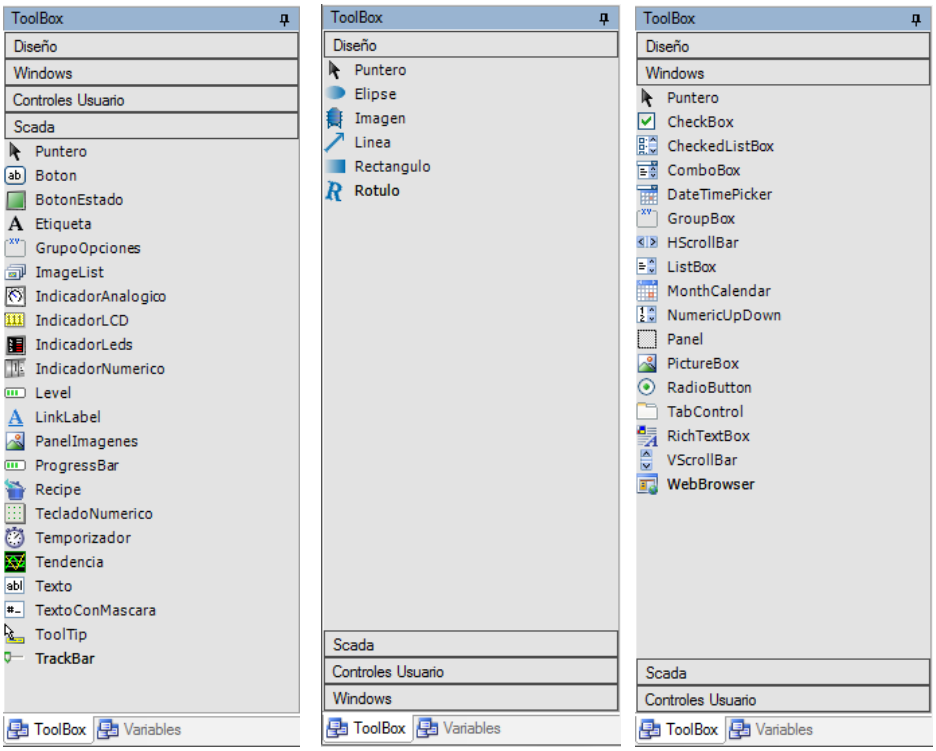
Once we have assigned a control type to the variable, each time we drag that variable, a control of this type is created in the desired form.



Another useful action we can do is to drag a variable, but dropping it on an existing control. This action will cause the target control, the one on the form, acquire the Server, Group and Variable properties from the dragged variable. If we drag the variable on a control type without the mentioned properties nothing occurs.

MONITORIZA CONTROLS

Controls that can be used are in the ToolBox in any of the tabs, Scada, Design, Windows or User Controls.

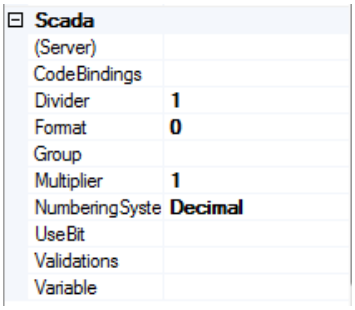


All these controls can be dragged to the surface of a form, once on the form their properties may be edited in order to give them the desired functionality.

There are two main groups of controls. One allows us to group functionalities, such as Panel and TabControl, and the other show the user the values of the variables.

These controls have properties to define how to display the value of a variable.

Let's see an example of these properties, used in most of the cases.



The properties (**Server), Group and Variable** refer to the variable to be represented. **UseBit** property, if used, refers to the bit between 0 and 15, which will be displayed (value 0 or 1).

Multiplier and **Divider** properties alter the value displayed. The property **NumberingSystem** allows you to display values in hexadecimal.

Format allows us, for example, display and limit the number of decimal places displayed.

Finally, **Validations** property allows us to establish conditions for writing and display on the control, when validations are true we will receive an incidence warning next to the control.

Let's see the validations editor:

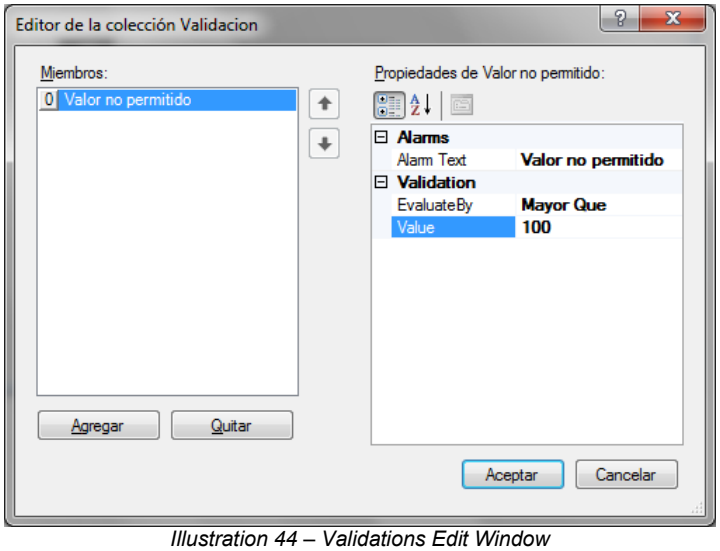
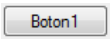


Illustration 44 – Validations Edit Window

The complete list of available controls is as follows:

Scada Tab



Boton

The functionality of this control is to implement actions by the user.

The main properties of this control are: ***Action*** and ***Actions***, we will use the first when we want to make a single button action and property ***Actions*** should be used when we want a series of actions in rapid succession.

The possible options for the ***Action*** property and set of actions are:

- **NoAction:** Indicates that the button has no action to perform.
- **OpenForm:** Opens the form named in the property ***Form***.
- **CloseForm:** Closes current form.
- **ExitScada:** Closes the application.
- **ForceValue:** Force the value of the variable referred by **Server, Group and Value** and makes that the new value is the one in the **Value** property
- **ShowAlarmsHistory:** Opens the Alarms Storage form.
- **ShowGraphic:** Opens the Data Storage form.

To establish a set of actions for the button it's made using the **Actions Collection Editor** as shown in the image below:

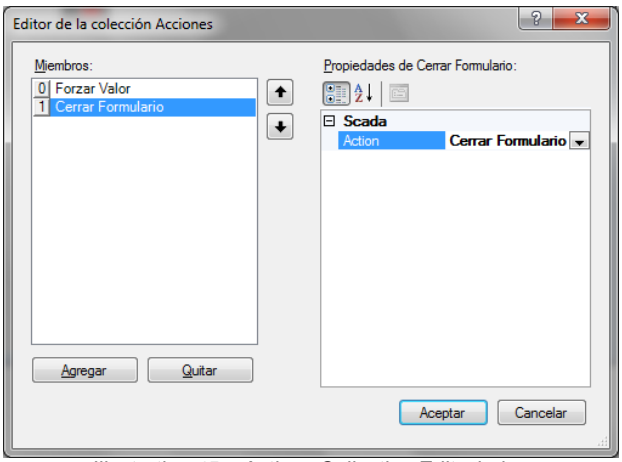


Illustration 45 – Actions Collection Edit window

BotonEstado

This control normally will be used to display and modify the status of a variable.

The default operation is to assign to StatusButton a variable defined in the communication servers and display a color or another depending on whether the variable is 0 or 1, and each time you click the button the value of the binded variable can change to 0 or 1.

We can display a set of images or a list of colors depending on the value of the variable changing through the collection of images or colors each time you press the button. Each of the images or colors is assigned to a value or range and show the image or color depending on the value of the variable.

The main properties of this control are:

- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Color v=0 y Color v=1:** Define the color that will show the button when the variable is 0 and 1 respectively. The default colors are red for the value 0 and Green for 1
- **Shape:** Possible values are Circle and Square, the button is represented respectively as seen in the following figures:

- **BorderWidth:** Defines width of the border line.
- **ReadOnly:** Indicates that the button will simply display the value of the variable, pressing the button will have no effect
- **Value:** Variable value associated with the properties Server, Group and Variable.
- **Text:** The text property allows us to establish a text that is displayed in the control.

- **Images:** When we make the button to display a set of images or a list of colors depending on the value of the variable we must use this property with the editor of the collection shown in the figure below:

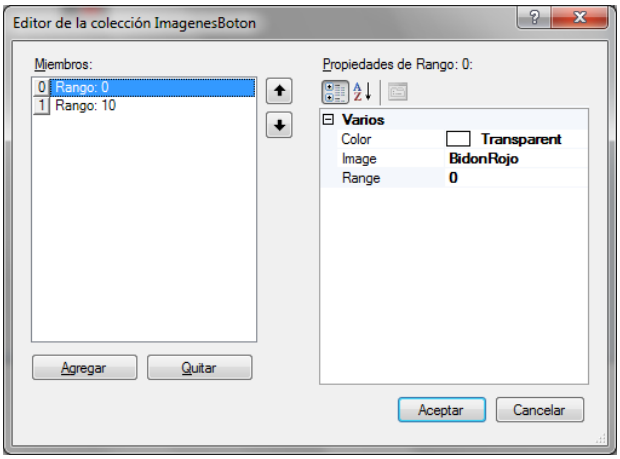


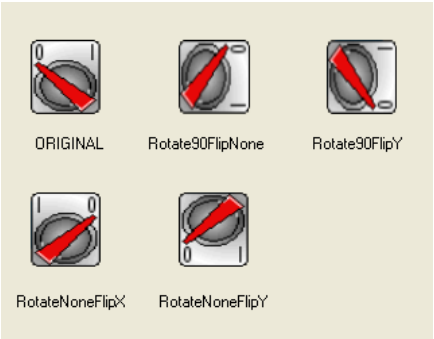
Illustration 46 – Images & Colors Collection Editor

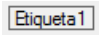
We should add value ranges as many states we want to have, for each state we assign a color to an image, if an image not is assigned to a range control will show the color corresponding to that range, if an image is assigned, control will display image instead the color.

Property Image will override all the color properties defined for the rank. That is, if a color is defined for v=0 and v=1 and images defined for these ranges, control will use the images instead the colors.

Also, the images are affected by the Rotate property, that modifies the aspect of the visualization, transforming, if we desire it, the way of showing it.

Rotate integrates in one property the possibility of rotate the image (90, 180 and 270 degrees) and the possibility of create a mirrored image (in the X axis, in the Y axis or in both axis).
Let's see several examples of it:





Etiqueta

The label control allows us to display fixed text, variable values and text variables based on the value of the variable

It is a read-only control, i.e. its functionality is to display text or values but we can't change a value with this control.

The default behavior of the control is display a fixed text. To show the value of a variable we must bind a variable to control through properties **Server, Group and Variable**

If we want to show various texts we must use the messages property.

Main properties of the control are:

- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.

  Setting these properties change the behavior of the control, so the control show the value of the variable referenced by Server, Group and Variable instead of text that is set to the Text property.

- **Text:** Text of the control. This property has no effect if we bind the Server, Group and Variable properties.
- **AutoEllipsis:** If set to *True*, the text that extend beyond the width of the control is replaced by ellipses.
- **AutoSize:** If set to *True* will enable automatic size of the control to fit the size of the text.
- **Divider:** When control is binded to a variable, shows the value of the variable divided by the value of the property.
- **Format:** When control is binded to a variable, displays the value using the format specified by the property, for example a format equal to # # .00 variable will cause the display of two decimal places.
- **Multiplier:** When control is binded to a variable, shows the value of the variable multiplied by the value of the property.
- **UseBit:** Bit number to use in the variable.Text may display 0 or 1, depending the value of the variable.
- **Value:** Value of the variable associated with Server, Group and Variable.
- **Messages:** Using this property we can make the control to display different messages depending on the value of the variable bound to it, using the Messages Collection Editor to establish the ranges that we want use to show each of the messages.

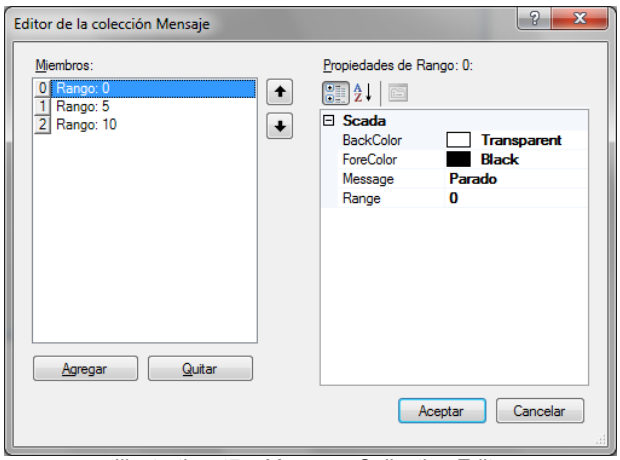
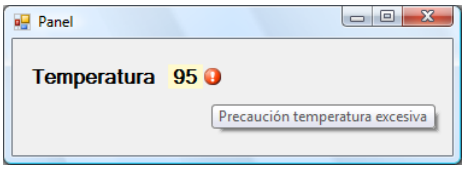


Illustration 47 – Message Collection Editor

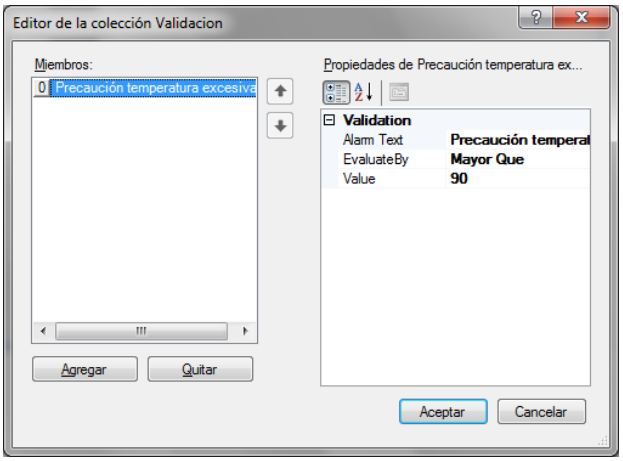
For each of the possible messages we must set the properties Range, Message, BackColor and ForeColor, being Range: the highest rank value below

which we want to show the message, Message: the text message will be displayed and BackColor and ForeColor colors of background for label and text respectively.

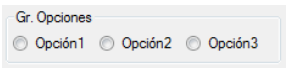
- **Validations:** Validations property allows us to define a set of conditions on the value of the variable that is bound to the control, so if running the project value met any of these conditions will display a notice at the control, for example we can design a form as the following figure:



Where two labels have been defined, one with a fixed text and one linked to a variable with a property defined in Validations Collection Editor as shown in the figure below:



In which is specified to display an alarm message associated with this control when the variable value is greater than 90.



GrupoOpciones

GrupoOpciones control is used to set or display the value of a variable that has a defined set of values.

The main properties of this control are:

- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Text:** Text shown as headline of the group
- **ReadOnly:** If set to true, control will display the value but we can't change this value.
- **Value:** Value of the variable associated with Server, Group and Variable.
- **Options:** With this property we can set the options associated with the control, this will use the Options Collection Editor shown in the figure below

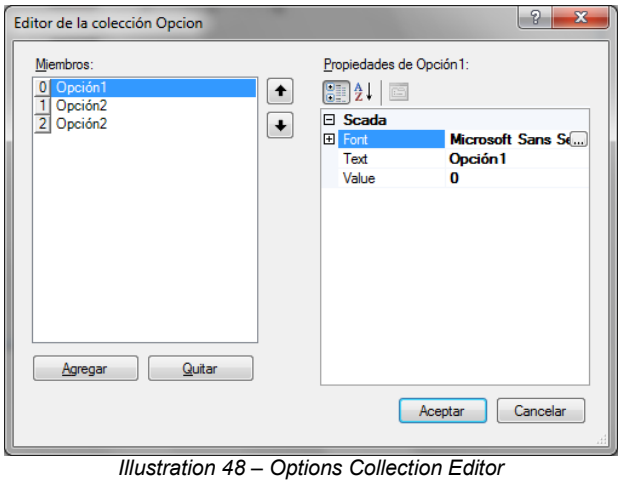
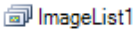


Illustration 48 – Options Collection Editor

In which, for each of the options, we set the Value, Text and Font properties, where Value is the value of the variable which is selected and will set the value to the variable when select the option, Font is the text font for the option.



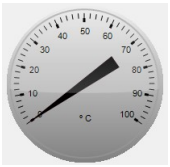
ImageList

The ImageList control is used to store images that can be displayed in other controls.

You can use an image list with any control that has a property ImageList

The main properties of this control are:

- **ColorDepth:** Gets or sets the number of colors used to render the image of the image list. This property must be established before the Images property.
- **ImageSize:** Image size in pixels. This property must be established before the Images property.
- **Images:** Collection of images stored in the ImageList.



IndicadorAnalogico

The function of this control is to show the value of a variable visually, having an analog representation of the variable, usually, when we know the range of values for a variable and want to make a graphical representation of it.

The control has the ability to store the maximum value it reaches in the corresponding session, or while the application is running, not permanently stored.

If you want, while running the application, you can restore the maximum by clicking on the control.

You can programmatically invoke the Reset method to reset the control maximum.

The main properties of this control are:

- **ArrowColor:** Gets or sets the display color of the arrow indicating the value of the variable.
- **Decimals:** Number of decimal digits shown in the numbers on the scale of values.
- **StoreMaximum:** Set to *True* if you want to display, during the session, the peak performance for the variable. For example in the figure below shows the AnalogGauge representing a value of 60, reaching a maximum of 80

- **StartingAngle:** Initial angle in degrees for the initial value of the scale.
- **BarsBetweenValues**: Number of sections or sub-marks to be drawn between each of the major established marks for the scale of values.
- **EspacingBetweenNumbers**: Spacing in degrees between the leading marks of the scale values.
- **Interval**: Number of units of the scale values among leading marks.
- **Maximum**: Maximum value of the scale values that may render the control. Any variable value that exceeds the value of the property is represented with the value of the property.
- **MaximumStored**: Peak achieved during the session. Do not store the high value of the variable.
- **Minimum**: The minimum value of the scale of values that can represent the control. Any value of the variable less than the value of the property is represented with the value of the property.
- **ClockWiseDirection**: Direction of rotation which represent the values of the scale. If set to True meaning rotate in a clockwise direction.
- **Text**: Text associated with the control. Normally we use this text to indicate what we are measuring or units of the variable.
- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Divider:** When control is binded to a variable, shows the value of the variable divided by the value of the property.
- **Multiplier:** When control is binded to a variable, shows the value of the variable multiplied by the value of the property.
- **Value:** Value of the variable associated with Server, Group and Variable.

IndicadorLCD

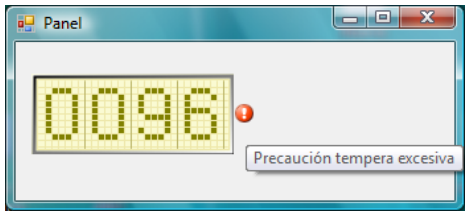
The primarily function for LCDDisplay control is design because it will be used to show the value of a variable in a way that simulates the representation by a dot matrix LCD panel.

Its function is read only, and this control only can view data but can not modify them.

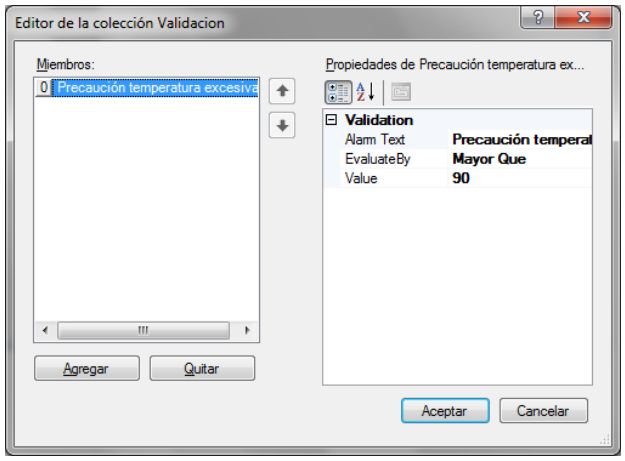
The main properties of this control are:

- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Divider:** When control is binded to a variable, shows the value of the variable divided by the value of the property.
- **Multiplier:** When control is binded to a variable, shows the value of the variable multiplied by the value of the property.
- **Value:** Value of the variable associated with Server, Group and Variable.
- **Format:** Displays the value in the format specified by the property, for example equal to 0000 format, will make the variable display box is filled with zeroes on the left of the value, as shown in Figure.

- **Validations:** Validations property allows us to define a set of conditions on the value of the variable that is bound to the control, so if running the project value met any of these conditions will display a notice at the control, for example we can design a form as the following figure:

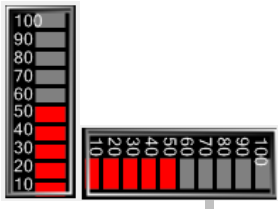


Which have placed LCDDisplay linked to a variable, this property has been set by Validations Collection Editor as shown in the figure below:



In which is specified to display an alarm message associated with this control when the variable value is greater than 90.

BorderWidth: Defines width of the border line.



LedDisplay

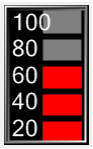
The LedDisplay belongs to the set of controls that serves to show in a graphic form the value of a variable.

Their representation is based on the pretense that depending on the value of the variable is turned on or off the "LED" shown in the control.

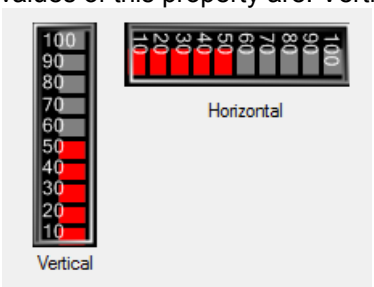
Its function is read only, and this control only can view data but can not modify them.

The main properties of this control are:

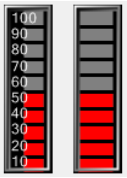
- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Divider:** When control is binded to a variable, shows the value of the variable divided by the value of the property.
- **Multiplier:** When control is binded to a variable, shows the value of the variable multiplied by the value of the property.
- **Value:** Value of the variable associated with Server, Group and Variable.
- **Maximum:** Maximum value of the scale of values. Any variable value greater than this value represents the maximum value.
- **LedsNumber**: Number of LEDs to display in the control. For example if we set its value to 5 with the maximum property 100 will be represented as shown in Fig.



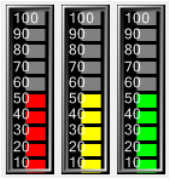
- **Orientation**: Possible values of this property are: Vertical and Horizontal.



- **ShowScale**: Determines whether to show the scale of control values.



- **ForeColorLeds**: Color of LEDs when illuminated.



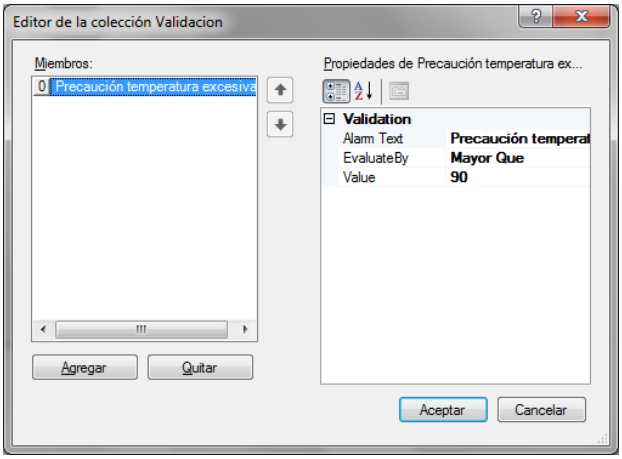
IndicadorNumerico

IndicadorNumerico control can show the value of a variable like an odometer or tachometer.

Its function is read only, and this control only can view data but can not modify them.

The main properties of this control are:

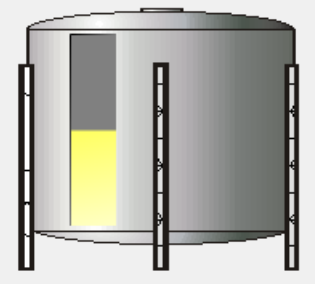
- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Divider:** When control is binded to a variable, shows the value of the variable divided by the value of the property.
- **Multiplier:** When control is binded to a variable, shows the value of the variable multiplied by the value of the property.
- **Value:** Value of the variable associated with Server, Group and Variable.
- **Format:** Displays the value in the format specified by the property, for example a format equal to ## .00 will cause the display to two decimal places using the color defined in ForeColorDecimal. 
- **Validations: :** Validations property allows us to define a set of conditions on the value of the variable that is bound to the control, so if running the project value met any of these conditions will display a notice at the control. In order to define the collection of valiodations we must use de validations Collection Editor.



- **Digits:** Number of digits used to display decimal places.
- **ForeColorDecimal:** Color for the decimal part of the number. For example, the value 30,56 will be displayed:  if we stablish ForeColorDecimal to Red and Format equal to ##.00.

Level

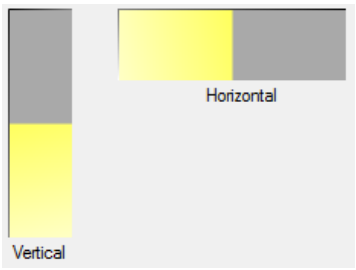
Level control is used to graphically represent a value of one variable. For example if we have a tank in wich we are controlling the volume we can put a volume control on Level associated with an image of the deposit and have a visual representation of the volume as we can see a figure:

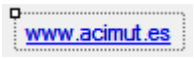


Its function is read only, and this control only can view data but can not modify them.

The main properties of this control are:

- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Divider:** When control is binded to a variable, shows the value of the variable divided by the value of the property.
- **Multiplier:** When control is binded to a variable, shows the value of the variable multiplied by the value of the property.
- **Value:** Value of the variable associated with Server, Group and Variable.
- **Maximum**: Maximum value of the scale values that may render the control. Any variable value that exceeds the value of the property is represented with the value of the property.
- **Minimum**: The minimum value of the scale of values that can represent the control. Any value of the variable less than the value of the property is represented with the value of the property.
- **ForeColorLevel:** Color for representing the value of the control.
- **Orientation**: Possible values of this property are: Vertical and Horizontal.

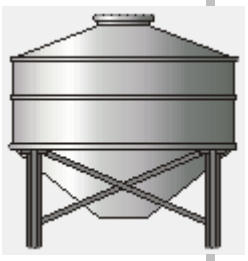




LinkLabel

LinkLabel control function has the ability to put links to Web sites on the Scada. So, when you click on a control LinkLabel opens Internet Explorer and navigate to the address specified by the URL property.

LinkLabel control is not associated with Scada variables and therefore can not represent any value.

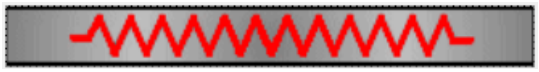


PanelImagenes

PanelImagenes control allows us to display different images depending on the value of a variable.

For example, we can make the control PanelImagenes to show next image if the value of the variable is less than 150

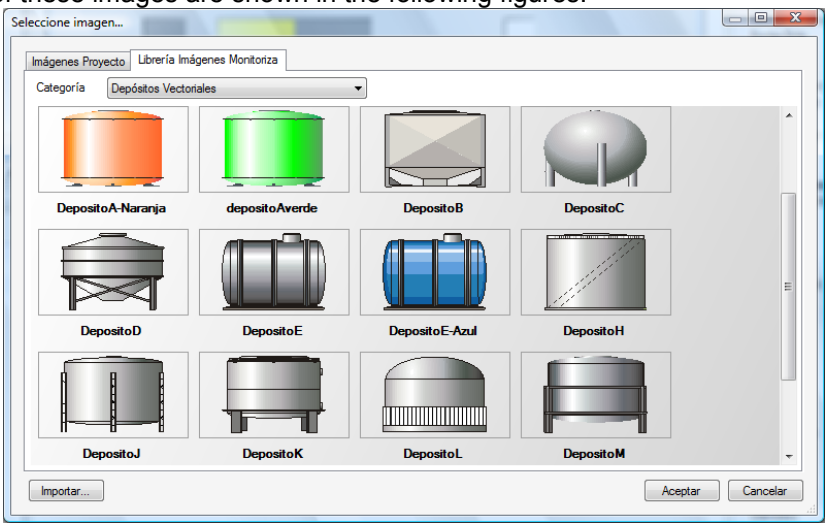
And the following image if is greater than or equal to 150

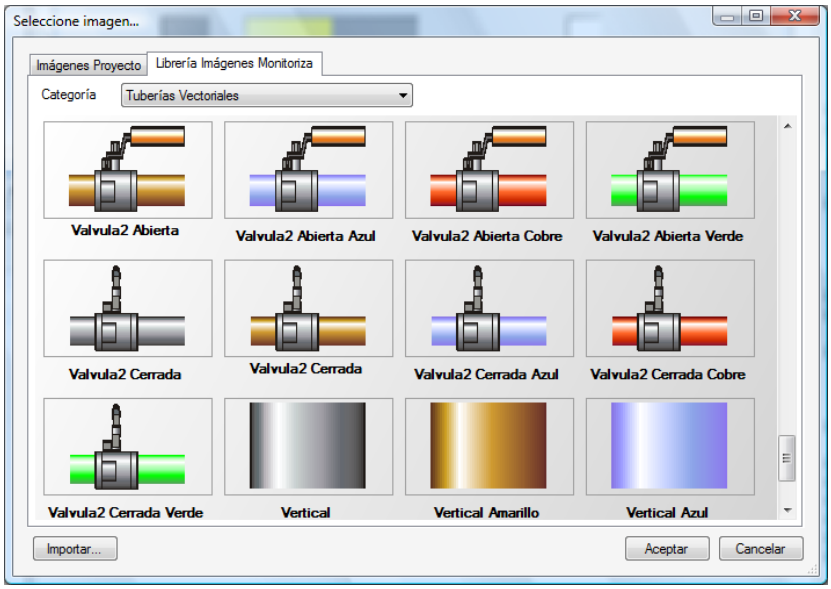


To assign the images we use the images property and using the Image Collection Manager we will define the images and values that we want to display.

Acimut Monitoriza features a wide variety of predefined images in the Image Library enabling us to make some attractive designs

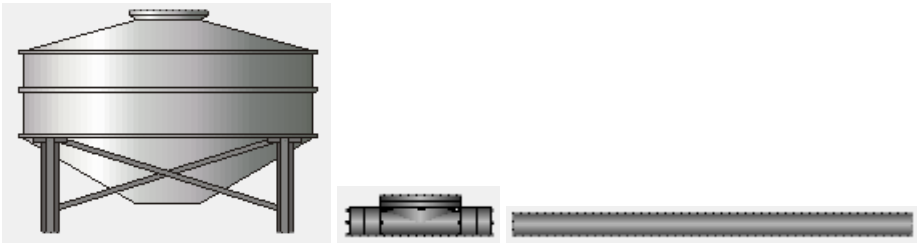
Some of these images are shown in the following figures:



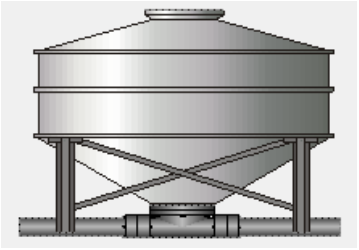


An important issue is that the images shown in PanelImagenes can have transparent areas, allowing us to put more controls ImagesPanel overlapping each other and to display the images that are provided under the above transparent.

For example suppose we have these three images of the Acimut Monitoriza Image Library:



We can combine putting a few before others to get this other figure, thanks to the parts that are transparent.



In this case transparency is achieved because the images used are created in vector 'wmf' format and is only a few solid areas concerned.

If we want a bitmap image have transparency, we define the image with the alpha channel with value 0.

The main properties of this control are:

- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Images:** Images property allows us to define the collection of images that we associate with the variable for this operation we will use in the editor of the collection of images, in which we will define the variable ranges and images that we associate with each value

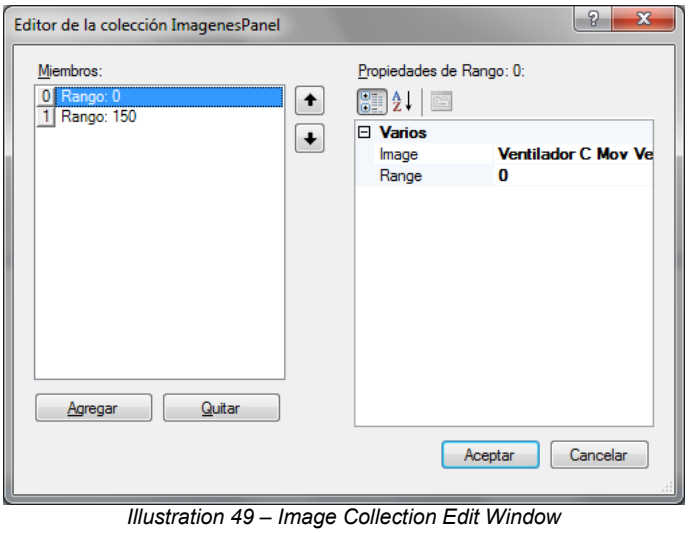


Illustration 49 – Image Collection Edit Window

Also, the images are affected by the Rotate Property, which modifies the aspect of the visualization, transforming, if we desire it, the way of showing it.

ProgressBar

ProgressBar control functionality is to represent the value of a variable, the size of the progress bar is proportional to the value of the variable.

The main properties of this control are:

- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Divider:** When control is binded to a variable, shows the value of the variable divided by the value of the property.
- **Multiplier:** When control is binded to a variable, shows the value of the variable multiplied by the value of the property.
- **Value:** Value of the variable associated with Server, Group and Variable.
- **Maximum**: Maximum value of the scale values that may render the control. Any variable value that exceeds the value of the property is represented with the value of the property.
- **Minimum**: The minimum value of the scale of values that can represent the control. Any value of the variable less than the value of the property is represented with the value of the property.

Recipe

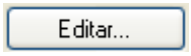
Recipe control is a control that allows users to view, edit and upload Recipes in the PLC, if they have permission to do it.

To understand how it works, it is appropriate to complement this information with the chapter related to recipes and batch control.

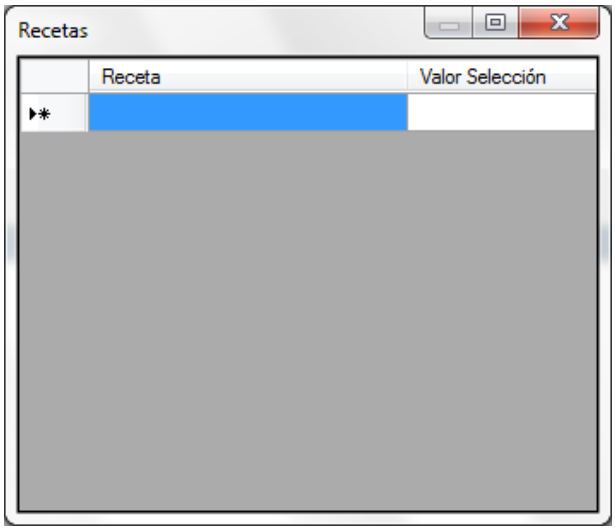
The main properties of this control are:

- **Template:** Here we specify what recipe template will use this control.
- **SendToPLC:** Indicates if, from this control can be uploaded a recipe in the PLC. During execution, you try to load a recipe; if any problems occur, or not comply with some requirements of the recipe, some elements of the grid will show in red, otherwise the color will be green.
- **ReadOnly:** Indicates if the control allow to edit recipes.

The control buttons have the following features:



Edit or add recipes. The following screen is displayed.



SelectionValue field also allows us to specify the load value of each batch recipe if necessary.


Save modifications to the recipes. **Important**: The recipes are stored by the server in a file, in the same location and with the same name that the project but with the extension '*.recipes*'.


Load selected recipe into PLC.

In the grid we have the records to; if you have permission, modify the value to load in each of the records. Moreover, when attempting to upload a recipe in the PLC the grid will be colored in green if it is successful and in red if there is a problem or not met the conditions specified in the recipe.

TecladoNumerico

TecladoNumerico control function is the ability to have an on-screen keyboard in cases where the running system has no physical keyboard, for security reasons or because they are designed for industrial environtment and there is only a touch screen.

The control works by sending the key is pressed to the form control that has focus.

Switching between controls can be controlled by pressing 

.

Keys 

and 

move the insertion point one character to the left and right respectively.

Pressing 

will clear the immediate key left of the insertion point

Key 

send press the Enter key depending on the control wich has the focus the functionality will be different.

The main properties of this control are those for the visual appearance of control and are as follows:

- **BackColor:** Background color of the control.
- **ForeColor:** Foreground color of the control. Used to display the text of the keys and the border of them.
- **Font:** Font of the key text.
- **Fill:** Determine how the background of each key is painted. The possible values are: None - 

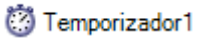
, Horizontal – 

, Vertical - 

, ForwardDiagonal - 

and BackwardDiagonal - 

.
- **FillColorStart:** Gradient start color of the background of the key.
- **FillColorEnd:** Gradient end color of the background of the key.



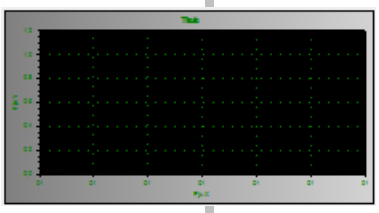
Temporizador

Timer control is used to trigger an event in the user-defined intervals.

Acimut Monitoriza supports extensibility of the framework through programming functions and libraries in the language *.Net (Visual Basic and C #)* and it is within this framework where the timer is helpful, since we can personalize the timer Tick event as seen in the Extensibility and Programming chapter and in this event perform related functions.

The main properties of this control are:

- **Interval:** Frequency of the Tick event in milliseconds.
- **Enabled:** Enables or disables the generation of Tick events.



Tendencia

The Tendencia control is used to display the value of one or more variables over time.

We must put this control on a form if we want the user of SCADA has a screen to see how evolve the set of variables has been defined in the design.

If we want that at some point the user can see a history by selecting the variable data and the period, we need a button with the Action property set to MostrarGrafica as shown in paragraph 'Show Historical Data'

An important thing to remember when we use a control trend is that, if we have a history of variable values we must define actions for save these variables as seen in paragraph 'Save to Database'.

In case of not saving the values of variables in the control database, trend can also show the changing values of variables but only displays the values collected during the current session, i.e., starting with the empty graph without any value represented and as time goes on will generate the corresponding graph, but closing the application these values will be lost and will be empty the next time you run the scada.

In the same graph we can mix both types of variables, AccionGuardado type variables when we have historic values in the database and dynamic type variables when only represent the values contained in the current session.

The main properties of this control are:

- **Frequency:** Frequency for updating the data in the graph, expressed in seconds.
- **Period:** Is set in minutes and corresponds to the time interval shown in the graph. For example if you set a period of 100 minutes on the graph to visualize the values of the variables corresponding to the last 100 minutes.
- **Series:** Set of variables that we want to represent. To define the series uses the Series Editor shown in the figure below:

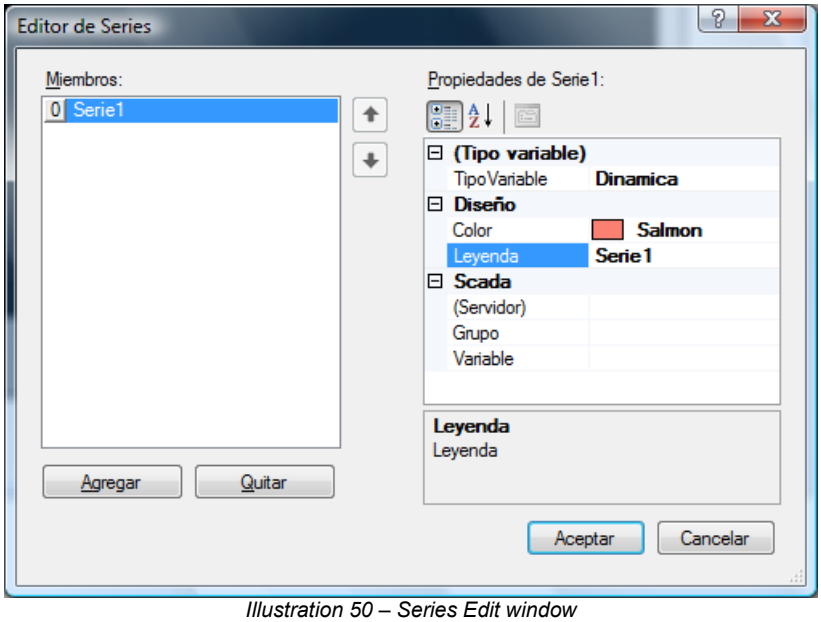


Illustration 50 – Series Edit window

For each series we have to set the following properties: *VariableType* determines whether the variable is saved in database or not; and accepts as possible values SaveAction and Dynamic, indicating respectively when we have to save in database and when not.

In case is *SaveAction* we must set the server properties, SaveAction and SaveVariable being respectively the database server where the variable to represent is defined and the particular variable that we represent.

In the case that the variable type is *dynamic* we have to set the Server properties, Group and Variable corresponding to server communications with the PLC, the group of variables where the variable is defined and the variable we wish to represent.

For both types of variables we must specify the color with which plot the variable using the Color property and the text that appears associated with the variable in the Legend property.

- **XAxisColor:** Color for drawing the horizontal axis.
- **XAxisText:** Text that appears next to the x-axis.
- **YAxisColor:** Color for drawing the vertical axis.
- **YAxisText:** Text that appears next to the y-axis.
- **GraphicStartColor:** Initial color for the chart gradient background.
- **GraphicEndColor:** Final color for the chart gradient background.
- **PanelStartColor:** Initial color for the chart area gradient background**.**
- **PanelEndColor:** Final color for the chart area gradient background**.**
- **Legend:** Set to True if you want to display the legends associated with each variable, else set to False.
- **TitleColor:** Text color for the title
- **FontTitle:** Font used to display the title.
- **TitleText:** Text for the title of the chart.

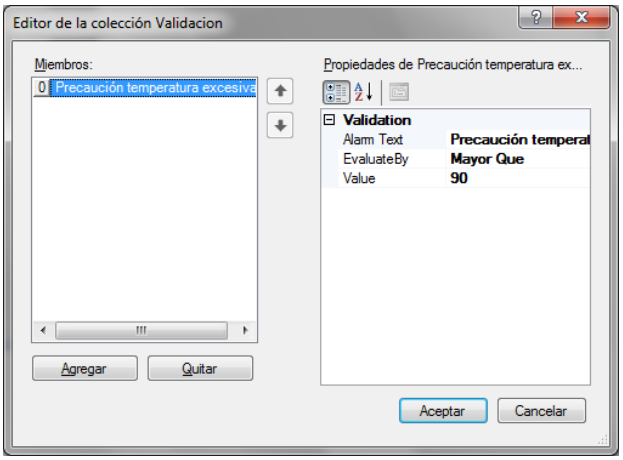
Text

The text control is possibly the most used control in any SCADA project, its goal is both to display values of variables and edit and modify these values.

The Value edition is completed by pressing the *Enter* key, until Enter is not pressed, data (new value) is not transmitted to the PLC.

The main properties of this control are:

- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Divider:** When control is binded to a variable, shows the value of the variable divided by the value of the property.
- **Multiplier:** When control is binded to a variable, shows the value of the variable multiplied by the value of the property.
- **Format:** Displays the value in the format specified by the property, for example a format equal to ## .00 will cause the display to two decimal places.
- **Text:** Variable value associated with the Server properties, Group and Variable.
- **ReadOnly:** Controls whether you can change the value in the edit control.
- **NumberingSystem:** Displays the value of the variable in the numbering system specified. Possible values are decimal and hexadecimal.
- **Validations:** Validations property allows us to define a set of conditions on the value of the variable that is bound to the control, so if running the project value met any of these conditions will display a notice at the control. To define validations we must use the Validations Collection Editor in which indicating to every condition that we define the following properties: Alarm Text, *EvaluateBy and Value*, where Alarm Text is the alarm message text to display if condition is met, *Evaluated by* the kind of condition we want to establish, with the possible values Same, GreaterThan, MinorThan and Different and *value* is the value we want to compare.



MaskedTextBox

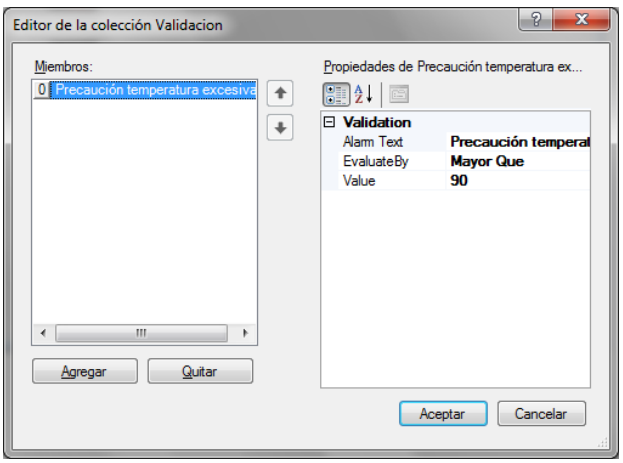
MaskedTextBox control is very similar to control text, the only difference is that we can define a mask for editing the text, for example, if we're showing a variable representing time information we can use a mask as __:__ in which separate the hours of the minutes.

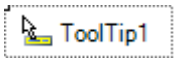
As in text control, we can view and edit values of variables using control MaskedTextBox.

The Value edition is completed by pressing the *Enter* key, until Enter is not pressed, data (new value) is not transmitted to the PLC.

The main properties of this control are:

- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Divider:** When control is binded to a variable, shows the value of the variable divided by the value of the property.
- **Multiplier:** When control is binded to a variable, shows the value of the variable multiplied by the value of the property.
- **Mask:** Sets the string that controls the entry allowed.
- **Format:** Displays the value in the format specified by the property, for example a format equal to ## .00 will cause the display to two decimal places.
- **Text:** Variable value associated with the Server properties, Group and Variable.
- **ReadOnly:** Controls whether you can change the value in the edit control.
- **NumberingSystem:** Displays the value of the variable in the numbering system specified. Possible values are decimal and hexadecimal.
- **Validations:** Validations property allows us to define a set of conditions on the value of the variable that is bound to the control, so if running the project value met any of these conditions will display a notice at the control. To define validations we must use the Validations Collection Editor in which we designate to every condition that we define the following properties: *Alarm Text*, *EvaluateBy and Value*, where *Alarm Text* is the alarm message text to display if condition is met, *EvaluateBy* the kind of condition we want to establish, with the possible values *Same, GreaterThan, MinorThan and Different* and *Value* is the value we want to compare.





ToolTip

ToolTip control is used to display an info window when you have your mouse over a particular control.

The main property of this control really is not assigned in the properties of this control, but simply by the presence of a ToolTip control on the form, all other controls contained in that form adquire the ToolTip property.

In this ToolTip property of each of the controls where we can introduce the text to show as help and information for the corresponding control.

TrackBar

TrackBar control represents the standard Windows control for tracking bars.

To configure the intervals between which the value of the Value property of a track bar will be moving, you must set the Minimum property to specify the lower end of the range and the Maximum property to specify the upper end of range.

- **Server, Group, Variable and UseBit:** Used to refer to the PLC variable depending on the communications server, the group that is defined and the symbolic name we give to the variable, additionally set the property UseBit if we want to refer to a bit in particular.
- **Divider:** When control is binded to a variable, shows the value of the variable divided by the value of the property.
- **Multiplier:** When control is binded to a variable, shows the value of the variable multiplied by the value of the property.
- **Value:** of the variable associated with Server, Group and Variable.
- **ReadOnly:** Controls whether you can change the value of the variable. If set to True you can display only values but may not be modified.
- **Minimum:** Lower end of the range.
- **Maximum:** Upper end of the range.

Design Tab

From it we can see controls belonging to Design Tab in Acimut Monitoriza.

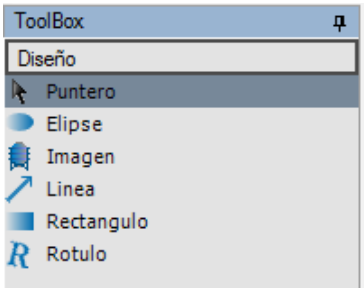


Illustration 51 –Design ToolBox Tab

These controls have primarily a function of design for our form and are not associated in any way with data display in Scada.

All controls in this tab are designed to have visual transparency and not to react to mouse clicks. That is, the controls aren't in a rectangular region of the form, as is normal in Windows, but in those regions where the control does not draw anything any control that is below will be visible, capturing mouse actions in these transparent zones. For example, we have a Rotulo control over a Boton control, as shown in the picture below.

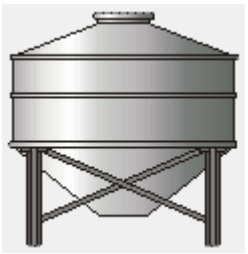
In this case Rotulo control, despite occupying the same rectangular area of the form, let us see the control button behind and if we click with the mouse for example in the center of the O in word Rotulo the click is made on Boton control.

Ellipse

The control allows us to draw Ellipses and Circles

The main properties of this control are:

- **BackColor:** Fill color of the ellipse if the Fill property is set to None.
- **ForeColor:** Color of the line forming the ellipse. If the Fill property is set to Solid color is used as the fill color of the ellipse.
- **Fill:** Fill Type of the ellipse. The possible values are: *None, Solid, Horizontal, Vertical, ForwardDiagonal and BackwardDiagonal*. *None* and *Solid* choices produce a solid color fill using respectively the BackColor and ForeColor to fill. Options *Landscape, Portrait, ForwardDiagonal and BackwardDiagonal* fill with a color gradient in the direction specified by the option.
- **FillColorStart:** Start color of the gradient filling.
- **FillColorEnd:** End color of the gradient filling.
- **Thickness:** Width of the line border.



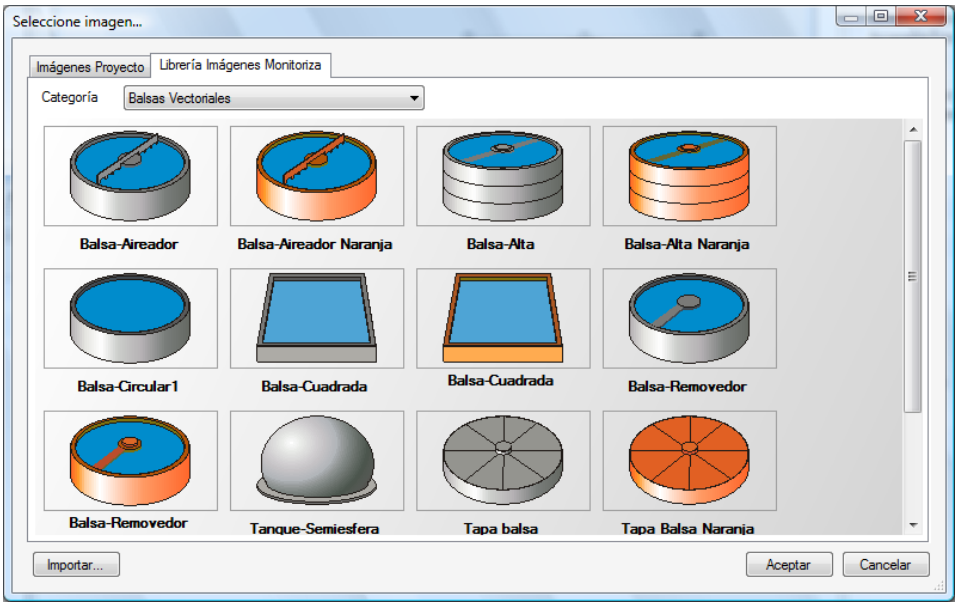
Image

The control **Image** is used to display an image in our form.

The main property in the Image control is the Picture property. By this property we establish the image displayed in the control.

Acimut Monitoriza features a wide variety of predefined images thanks to its Image Library that allows us to make some attractive designs.

Some of these images are shown in the figure below:

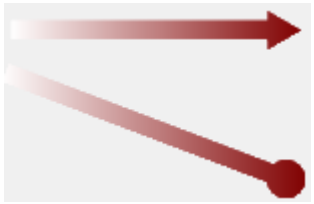


In the control Image, when you resize the control, the image is properly adjusted to the new size. This adjustment is more perfect if the image we have used is vector rather than bitmap. The Acimut Monitoriza Image Library provides images of the two types.

Also animated images are provided, for example in the following figure:

There are three versions of the fan; the first is an image that has movement while the last image corresponds to a stop. When selecting an image from the Image library images are displayed in animated movement to make clear its functionality.

Also, the images are affected by the Rotate property, that modifies the aspect of the visualization, transforming, if we desire it, the way of showing it.



Line

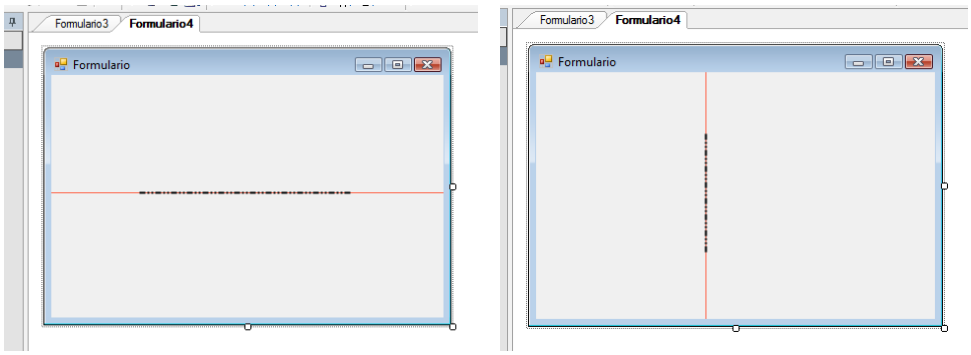
The control line serves to draw a line under the surface of the form or within any other container control.

Line is represented as shown in the figure below:

Besides the identifying framework indicating the control is selected, we see two red dots. These two dots are called handlers and serve to drag the start or end of the line to another position. To do this, simply select the handler with the mouse and drag it to another position, when we release the button we draw the line with the start or end in the final position of the mouse handler.

The same could have been done modifying in the properties window StartPoint property or EndPoint property.

Another useful tool when drawing a line are the vertical and horizontal indicators, these indicators are shown when moving the starting or end point of a line through their handlers and the line is parallel to one of the coordinate axes such and as seen in the following pictures.



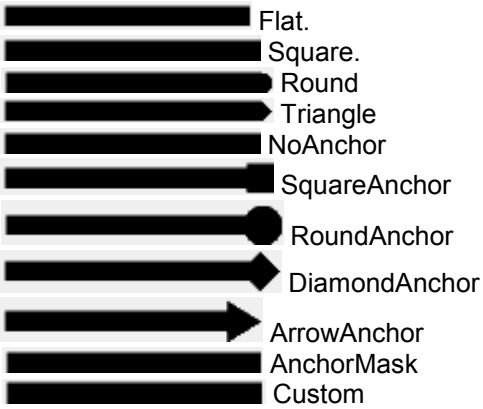
Demo: http://www.acimut.com/monitoriza/videos/mover_linea/default.html

The main properties of this control are:

- **ForeColor:** Color of the line forming the ellipse. If the Fill property is set to Solid color is used as the fill color of the ellipse.
- **Fill:** Fill Type of the ellipse. The possible values are: *None, Solid, Horizontal, Vertical, ForwardDiagonal and BackwardDiagonal*. *None* and *Solid* choices

produce a solid color fill using respectively the BackColor and ForeColor to fill. Options *Landscape, Portrait, ForwardDiagonal and BackwardDiagonal* fill with a color gradient in the direction specified by the option.

- **FillColorStart:** Start color of the gradient filling.
- **FillColorEnd:** End color of the gradient filling.
- **Thickness:** Width of the line border.
- **StartPoint:** Coordinates for starting point of the line.
- **EndPoint:** Coordinates for ending point of the line.
- **StartCap and EndCap:** Sets out the shape of the beginning and end of the line respectively. The possible values are:
-



Rectangle

The Rectangle control allows us to draw rectangles that we can fill with gradients.

The main properties of this control are:

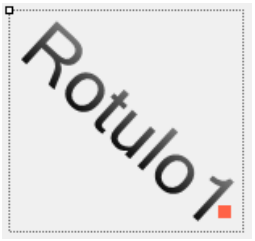
- **BackColor:** Fill color of the ellipse if the Fill property is set to None.
- **ForeColor:** Color of the line forming the ellipse. If the Fill property is set to Solid color is used as the fill color of the ellipse.
- **Fill:** Fill Type of the ellipse. The possible values are: *None, Solid, Horizontal, Vertical, ForwardDiagonal and BackwardDiagonal*. *None* and *Solid* choices produce a solid color fill using respectively the BackColor and ForeColor to fill. Options *Landscape, Portrait, ForwardDiagonal and BackwardDiagonal* fill with a color gradient in the direction specified by the option.
- **FillColorStart:** Start color of the gradient filling.
- **FillColorEnd:** End color of the gradient filling.
- **Thickness:** Width of the line border.
- **CornerRadius:** Radius for rounded corners.

Rotulo

Rotulo control provides the ability to display text in any direction and with a filling color that can be either solid color with a gradient.

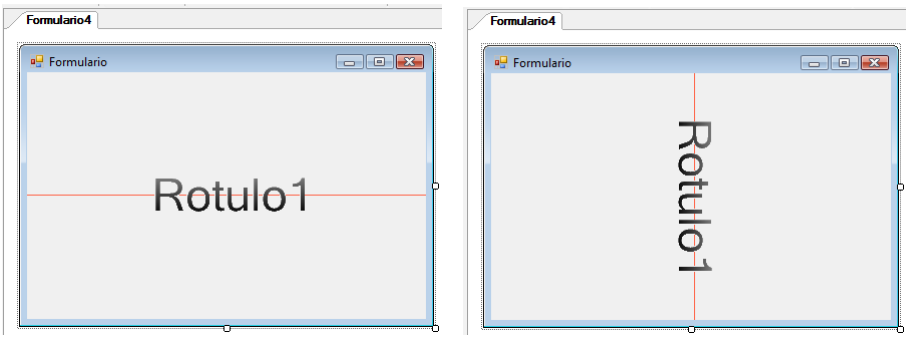
To rotate the label in any direction we have two options: set the Angle property to the value we want or use the Form Editor as with the control line.

If you drag this handle when we release the handler the label will be drawn in the direction specified.



The ability to turn the label using the front handle allows us to set the Angle property easily.

Another useful tool when designing a label are the vertical and horizontal indicators, these indicators are displayed when you are moving the label by its handle and the label is parallel to one of the coordinate axes as shown in the following images.



Demo: http://www.acimut.com/monitoriza/videos/mover_rotulo/default.html

Due to the transparency feature of the Rotulo control, for selecting the control we must to click in non transparent region of it.

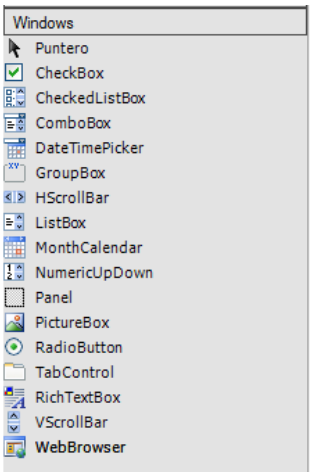
The main properties of this control are:

- **BackColor:** Fill color of the ellipse if the Fill property is set to None.

- **ForeColor:** Color of the line forming the ellipse. If the Fill property is set to Solid color is used as the fill color of the ellipse.
- **Fill:** Fill Type of the ellipse. The possible values are: *None, Solid, Horizontal, Vertical, ForwardDiagonal and BackwardDiagonal*. *None* and *Solid* choices produce a solid color fill using respectively the BackColor and ForeColor to fill. Options *Landscape, Portrait, ForwardDiagonal and BackwardDiagonal* fill with a color gradient in the direction specified by the option.
- **FillColorStart:** Start color of the gradient filling.
- **FillColorEnd:** End color of the gradient filling.
- **Thickness:** Width of the line border.
- **Angle:** Angle of the text in the control.

Windows Tab

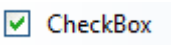
In this section are described the controls of the ***Windows Tab*** in the Monitoriza Toolbox.



This tab shows the standard Windows controls, and therefore can't be associated to the Monitoriza variables, because they don't have any of the scada properties like Server, Group and Variable, and therefore can't represent the data binded whith these variables.

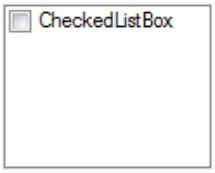
Their inclusión in the toolbox is due to the possibility of providing additional functionality to the Scada controls using Programming and Extensibility that is implemented in these windows controls.

A brief description of each control:



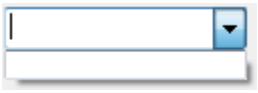
CheckBox

CheckBox control is used to make easy to the user choose between true and false values.



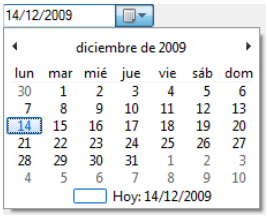
CheckedListBox

CheckedListBox control allows to show a list of ítems with a checkbox associated with each item in order to define the selected items.



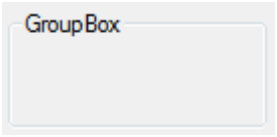
ComboBox

ComboBox control allows to show a dropdown list of ítems. ComboBox control has two parts, one part is always visible allowing the user to type like in a textbox, and a second part that is a, normally hidden, listbox that we can expand to select an item from the list.



DateTimerPicker

DateTimerPicker control allows the user to select a date or a time.



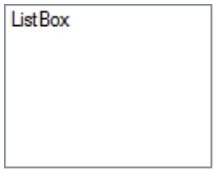
GroupBox

GroupBox control is used to Group other controls. There are two main reasons to group controls:

- Create a visual grouping of the controls in order to create a more useful interface.
- Create a grouping functionality, like including several RadioButton, but allowing only the selection of one of them.

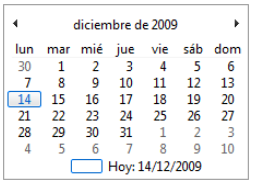
HScrollBar

HScrollBar Control allows performing a simple browsing when we have a large amount of information items to show.



ListBox

ListBox control is used to show a list of ítems which the user can select one or more.



MonthCalendar

MonthCalendar control allows user to select a date, showing a complete calendar on which the user can browse years and months.

NumericUpDown

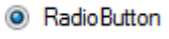
The NumericUpDown control is a combination of a textbox control and two controls that allows to increment or decrement the value of the control.

Panel

Panel control is used to visually group other controls and make a more clear and visual design for the SCADA. For example: setting the backcolor of the panel to different colors depending on the meaning for the controls.

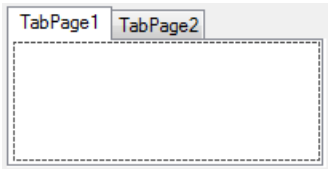
PictureBox

Picturebox Control allows us to display an image. By using the controls Image and PanelImagenes from Acimut Monitoriza we can perform more powerful functions because, from these controls, we can access to the integrated image library.



RadioButton

RadioButton control allows the user to select an option from a group of options when the radio button is matched with other radiobuttons.



TabControl

TabControl is used to show several tabs, like dividers on a notebook or labels of the folders in a filing cabinet. Tabs can contain images and other controls.

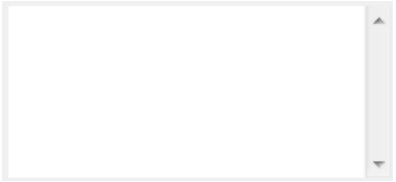
The most important property for the tabcontrol is **TabPages,** that contains the individual tabs. Each individual tab is a TabPage control.

RichTextBox

RichTextBoxcontrol allows displaying formatted text, that is, with different fonts and styles.

VScrollBar

VScrollBar control allows an easy browsing when we have a large set of items to show.



WebBrowser

WebBrowser control allows the user to browse web pages.

In order to use the control we must to set in the ***Url*** property a valid web address, for example a valid url can be: http://www.acimut.es

TEST THE PROJECT

Acimut Monitoriza has the option to simulate the execution of the project while we are developing it using two different ways.

If our development environment has no connection with PLC neither other devices we want to check, we can test the project using 'Simulation' mode.

Else, if we can access to the PLC (or the desired device) we can use the 'Server' mode to test.

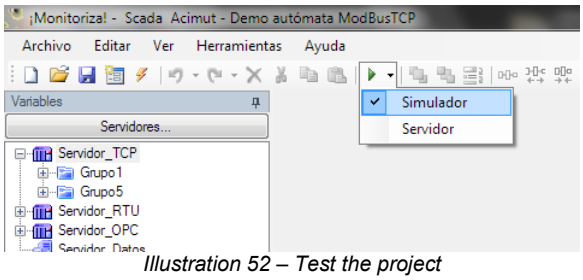


Illustration 52 – Test the project

By choosing the Server option, the application runs a Monitoriza Server that performs all communication tasks with PLC's and devices, and then it displays a Scada Client with our interface for the scada.

This way of testing is the most accurate to the production environment and the server connects to the real devices.The only difference between a real environment and this testing environment is that while in production we usually have the server and the client in different machines, in testing all applications, editor, server and client, are running in the same machine.

Once we have chosen an option (Server or Simulator) the project editor stores the preferred option and we can simply click on the Play button to run it again without having to display the options menu.

If we choose the simulation option is because we have no access to devices, or simply want to test for navigability between the forms in the user interface.

In the simulation there is no connection with PLC and therefore there are not real values of the variables.

This is a fairly situation, Acimut's Monitoriza gives us the option to simulate variable values using a form in which we can assign values to variables to see if our development is behaving as we expect (if an alarm is triggering while we had planned, for example).

While the definition of the variables we saw a section in the variable definition indicating the simulation type.

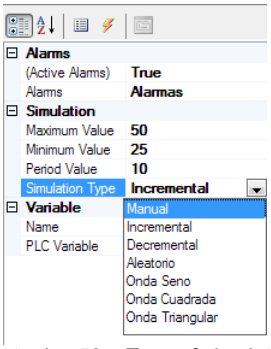


Illustration 53 – Type of simulation

If we choose a type of simulation different to manual, the value of the variable will change automatically.

We may also in the simulation, using input controls, such as control text, input values to variables and test the Scada like we're receiving data.

Through simulation we can test the permissions that are established for each defined user.

We are prompted to save the project if we have not done it previously and will launch the simulation.

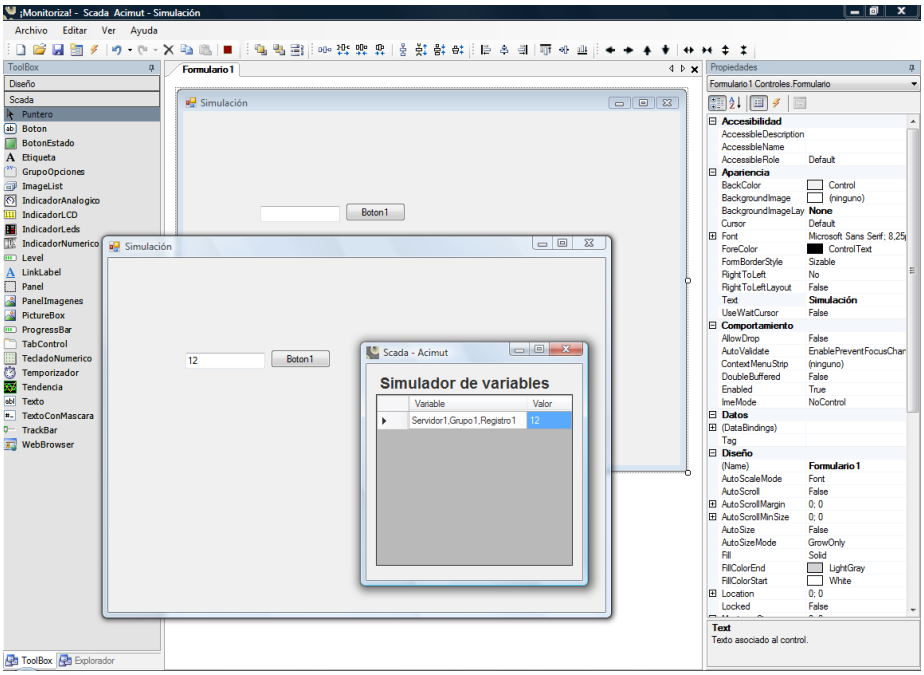


Illustration 54 – Simulation running

In the picture above you can see the Monitor Editor and the form you are designing running.

The form has added a text control that has been associated with variable Registro1 of Group1 and Server1, so when the window of the simulator variables enter a value in that variable that is immediately displayed in the associated text control.

To end the simulation we can either close the window of the simulator variables or close all forms in the project that we have opened or click on the 'play' button that in this moment is displaying a red square instead the green triangle indicating that we can stop the execution.

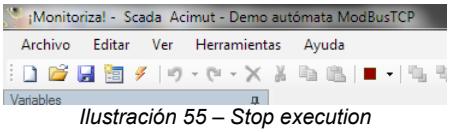


Ilustración 55 – Stop execution

Demo: http://www.acimut.com/monitoriza/videos/simulacion/default.html

IMAGE LIBRARY

One feature that Acimut Monitoriza that helps in the form design is the Image Library.

In one hand,the Image Library is a set of clip art by Acimut that can be used freely in any project that we develop and the in the other hand has the possibility of creating our own image library project having categorized and readily available images ready for use.

Another advantage of the Image Library is that if we use the images in the Library, the project stores only a single instance of the image, regardless of how often we use it, which makes the size of our projects are not unnecessarily increased .

Images of the Library Art by Acimut are part of the facility and therefore do not increase the size of your project, simply makes a reference to the image.

Controls in Acimut Monitoriza that are capable of using this feature are:

- ***BotonEstado***
- ***PanelImagenes***
- ***Imagen***

When setting the Picture property of the image collection and BotonEstado or property PanelImagenes Picture in Picture control us a window like this is shown:

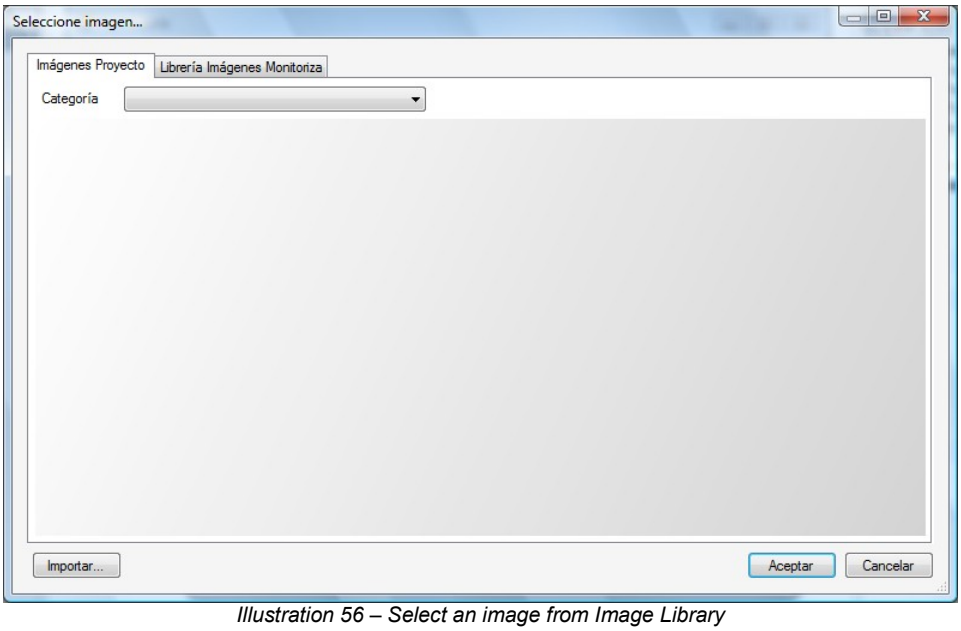


Illustration 56 – Select an image from Image Library

This has two tabs to determine the source of the image. The project images which contained all the images that we have added to the project and the Monitoriza Library tab that shows all the images of the library provided by Acimut.

In the same window there is also a button labeled Import used to assign an image control that we have but not is incorporated into the Library. The images that incorporate the control using the Import button increase the size of the project based on the number of times you use it.

To create the image library for the project select the option Image Library on the View menu as shown below.

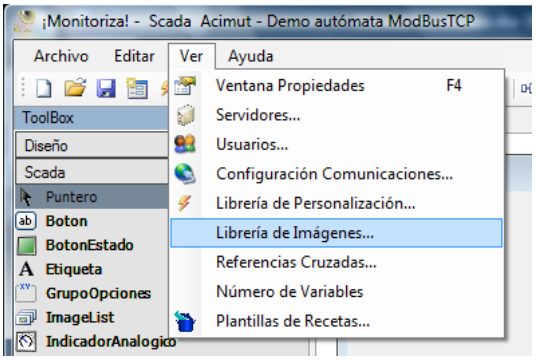


Illustration 57 – Image Library Menu

The following screen appears:

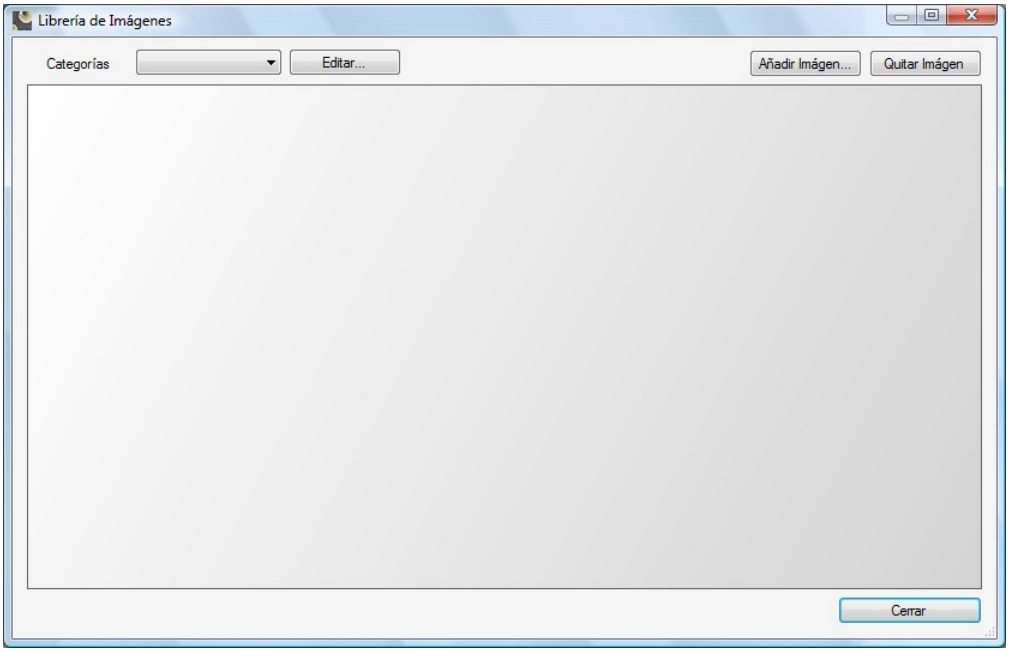


Illustration 58 – Image Library Creation

The first thing we must do is define the categories in which we classify our images. At least we define a category.

To set up categories and click the Edit button window appears the categories screen.

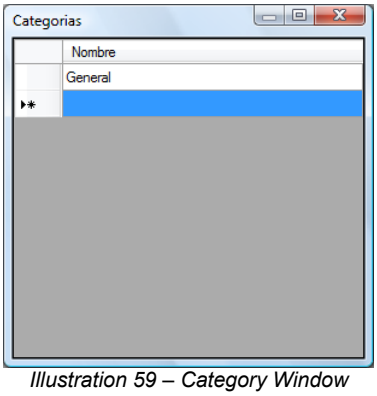
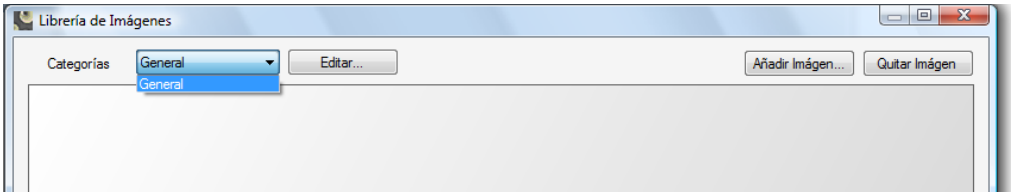


Illustration 59 – Category Window

In which we introduce each of the categories and press the button to close the window. From then on the category drop-down we will select the category to assign the image you want.

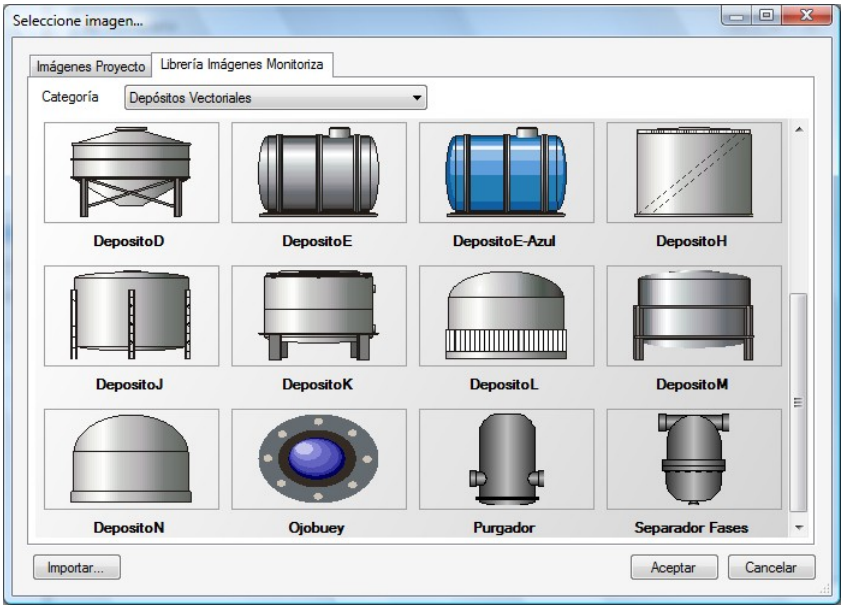


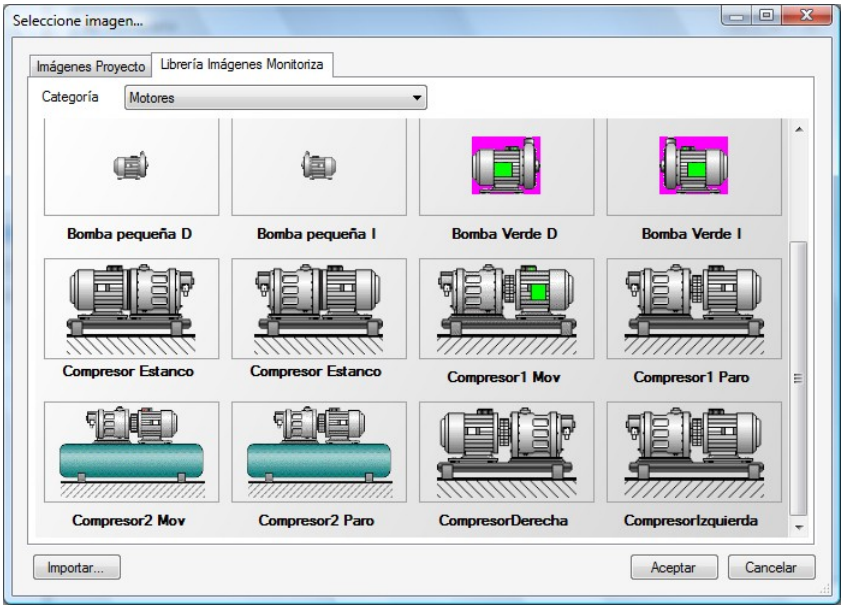
Once selected, you have the Add Image button to select the image you want to add and the Remove button to delete an image that we added to the Library Project.

The type of images that are accepted are: JPG and BMP raster images without animation, raster GIF with animation image and vector WMF image.

The Acimut Image Library is not modifiable, ie can not add or remove pictures from the collection of images that compose it.

Some examples of Acimut image library are:





USERS & PERMISSIONS

To edit application users there is a menu option *Users and permissions* from the *View* menu.

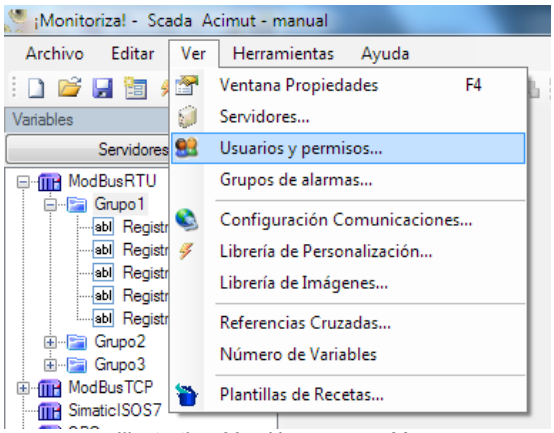


Illustration 60 – User access Menu

This menu leads us to the next screen

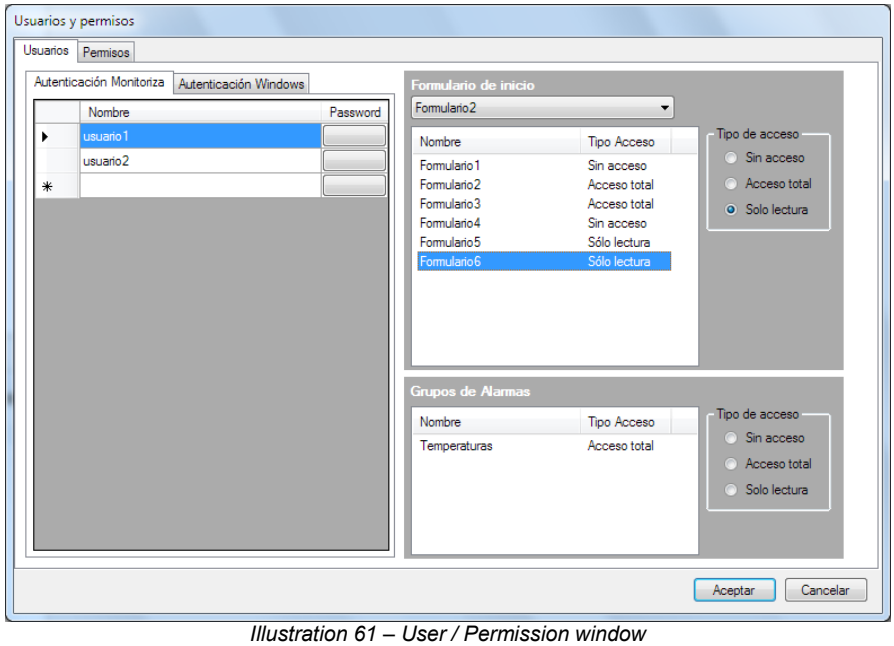


Illustration 61 – User / Permission window

Screen is divided in two tabs, the first one devoted to the definition of the Scada *Users* and the second one dedicated to the *Permisions* from a global to the project point of view.

In the *Users* tab we will able to register the necessary users, will specify their passwords and, as you can see, you can assign the startup form and permissions for each SCADA application form.

As you can see there are defined two types of users or, in other words, two types of user authentication. In one hand, there is the *Monitoriza Authentication*, which is characteristic of the Scada project itself and, in the other hand, there is the *Windows Authentication*, in which we can specify which users of the active directory (Domain) are allowed to use the Scada project.

Within a single project it is possible to create and use users with either Monitoriza or Windows authentication. Which one of both will be more accurated will depend on the security policies established in your environment.

When defining a user with Monitoriza authentication, a user name and a password must be provided, while defining a user with Windows authentication just needs to enter the Windows login in the format domain\user, as it appears in the active directory, and the password will be that one the user has defined in it, so when the user will need to login the application, he just need to enter the Windows user name and password as he does when login to Windows.

Whether you choose one type of authentication or the other, it is mandatory to specify the start form and the permissions for each form.

Keep in mind that if you assign a user *full access* permissions on a form, the user can modify variable values, unless it is prevented specifically by the control holding the variable; if we assign a *read-only* permission, the user will can not modify variable values regardless of the Read only property value of the controls.

You can also specify permissions for groups of alarms.

Alarm Groups are a way to group sets of alarms, based on the target users for those alarms. Por example, if we have two control points in a facility monitoring two different processes, for each alarm that we define, we must assign it to a group using the AlarmGroup property. Thus, in the permissions window we will set permissions for each user about whether the alarm is displayed or not and about whether the user can validate or not it.

By default, all users have full access to all alarm groups, that is, if we want to restrict access we must do it by specifying permissions.

Possible types of access are:

No Access: The user not is notified of the alarms.
Read Only: The user is notified of the alarm, but can't validate it.
Full Access: The user is notified of the alarm and can validate it.

The Permissions tab in the users form is as shown here:

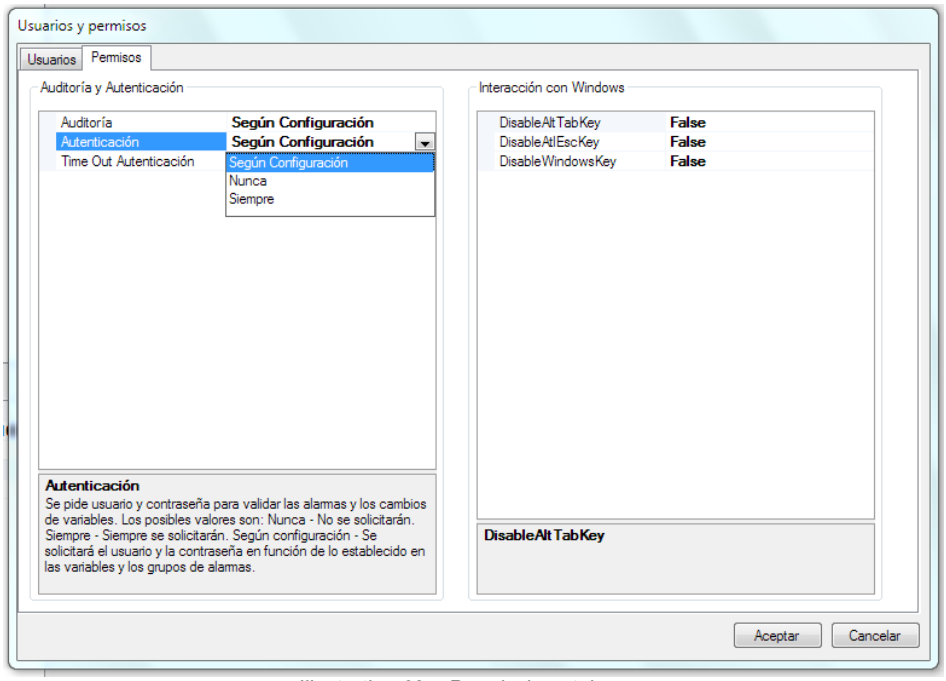


Illustration 62 – Permissions tab

This form is the place to establish general criteria for auditing events, changes authentication and the application interaction with Windows.

Auditing parameter

Through Auditing parameter it is set how to audit the use of the application, thus, it is possible to set a record of the events that occurs during the use of the application (when it is started, when it is stopped, when a variable value has changed and what is the new value, when a particular form is open, when it is closed…)

The possible values for this parameter are:
Never: Nothing will be audited.
Allways: Everything will be audited.
Depending on configuration: Changes of the variables will be audited depending on the **RequiresAudit** propertie value of each variable. All other events not corresponding to changes of the variables will be all audited.

Authentication parameter
Through Authentication parameter we can set whethe changes of a variable value or validation of an alarm will require a new authentication or not.

The possible values for this parameter are:
Never: The application will never require an authentification to the user before changing a variable or validating an alarm.
Allways: The pplication will allways require again user name and password before changing a variable value or validating an alarm.
Depending on configuration: User name and password will be required to the user depending on the **AuthenticationRequired** propertie value of the variable been changed or the alarm group that the alarm belongs to.

Time Out Authentication parameter
This parameter sets the time in seconds that an active authentication is valid, for example, if a Time Out of 30 seconds is established and a second change of a variable value is done before 30 seconds had passed after last authentication, the application will not require user name and password again.

A value of 0 seconds means that Time Out is infinite.

Interaction with Windows
The interaction with Windows parameters are used to set how we desire Windows to behave with respect with Monitoriza. These parameters allow us to define a higher level of security for a SCADA project because they can determine whether it is possible to use or not other applications simultaneously to SCADA, what might affect its performance.

The DisableAltTabKey property set to True means that the user cannot change to another application using de Alt+Tab combination.

The DisableAltEscKey property set to True means that the user cannot change to another application using de Alt+Esc combination.

The DisableWindowsKey property set to True means that the user cannot access the Start Menu and thus cannot start another applications or windows components.

RECIPES & INTERFACE BATCH

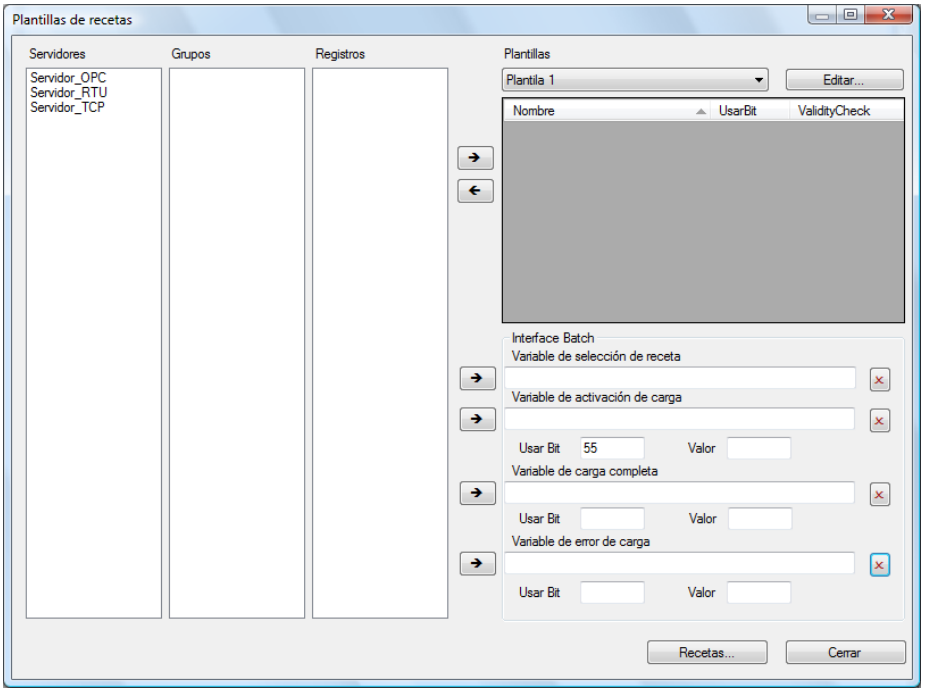
Let us first define what is a recipe. A recipe is a set of PLC records with a default value. It could serve as an example a mixture of several components in a manufacturing, depending on the percentage of these elements would have a different final product. The recipe would be, for example, the percentages of each element. It would mean different recipe depending on the final product that we wanted to obtain.

Monitoriza allows the loading of recipes either manually or automatically (Batch).

So, each set of recipes starts from a template, in which you define what we will use and whether such records shall comply with any condition.

Edit templates and recipes are available from the Templates menu option from the View menu recipes

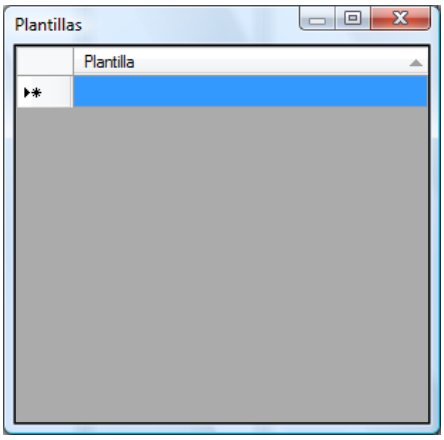
This option will lead to the following screen:



Each template will have an associated set of records that will be generic for all recipes and if so decided, a batch interface for automatic loading.

To get started we need to register at least one template, the Edit button ... will allow us to register, modify or delete templates recipes.

When we click on this button will display the following:



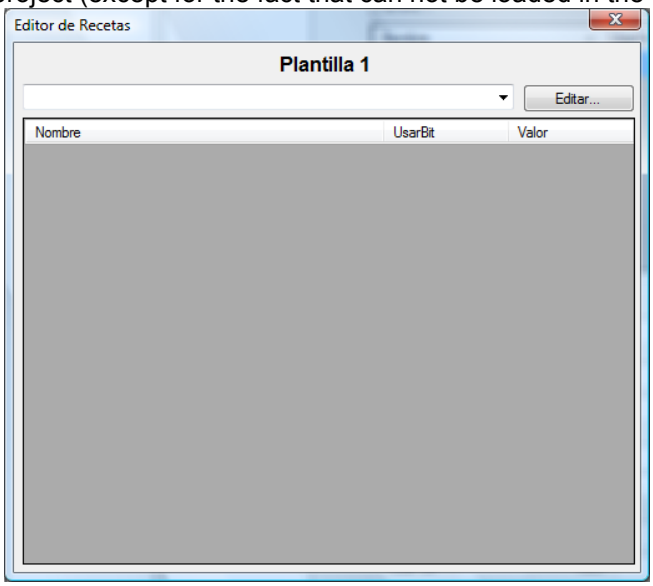
Once we have at least one template, we can add records to it, go for it by selecting the server, group and variable that we will use from the lists of the screen.

Each record added to the template has two additional fields:

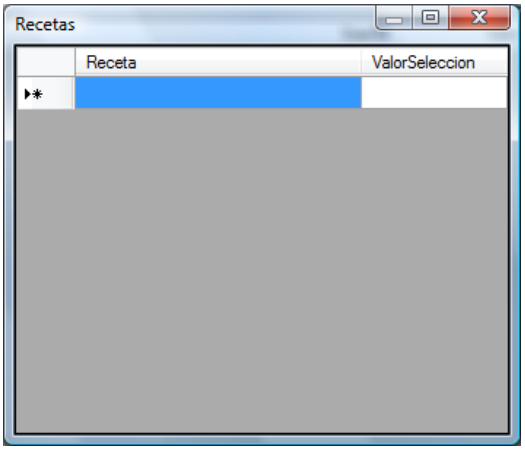
- **UseBit** allows us to use individual bits of the variable.
- **ValidityCheck** is a field that, if used, will be a requirement prior to loading of the recipe. In other words if the record hasn't that value will not load the recipe, nor manually, or by a batch procedure.

Once we have defined the template, we can add recipes using the recipes button (it is also possible in execution, by the user with the Recipe control).

This is the screen you will see, that has a behavior identical to the one while executing the project (except for the fact that can not be loaded in the PLC).



With the Edit button we add, edit or remove Recipes, ValorSeleccion field also allows us to specify the load value of each batch recipe if necessary. **Important:** The recipes are stored in a file in the same location and name the project has, but with the extension '.recipes'.



Every time we add a recipe, the system will populate the grid of the screen template registers, the user can simply fill in the values loaded in the PLC.

Each template has some registers to configure the load behavior in batch, ie, independently of the action of a user, the Monitoriza server checks a number of records to determine when to load a recipe in the PLC.

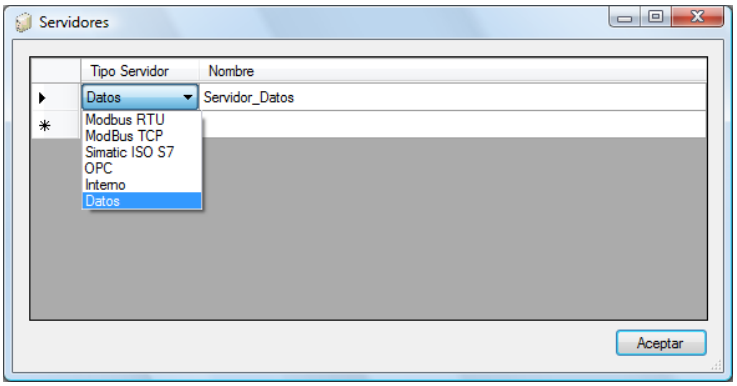
Batch settings for each template is in the Batch Interface section, this section we have four variables:

- **Variable for recipe selection,** this record is the one, with the value given adding prescription ValorSeleccion field, determine which recipe will be selected for loading in the PLC. If you want to configure a batch recipe this field is required.
- **Variable for load triggering, t**his record is the one that, when having the value specified, will trigger the load. If you want to configure a batch upload this field is required.

- **Variable for successful loading:** If the batch loading procedure is completed successfully, this register is written to the specified value in the PLC. There is a required field.
- **Variable for loading error:** if the batch loading procedure is not completed successfully, this register is written to the specified value of the PLC. There is a required field.

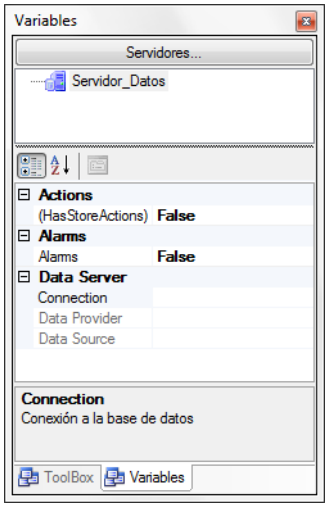
SAVING IN DATABASE

First we must create a server data type:

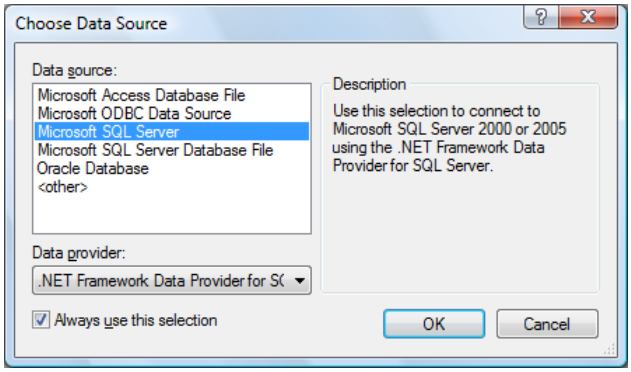


You can define as many data servers as you need, each of them will give us access to a particular database.

Editing **Connection** property:



see a wizard that will guide us in the connection process.



The alarms can be stored in a database, for this must be a table that is described in the appendices (Table Alarms). The data server in which we have this table should have **Alarms** property to True.

It will also be the data server that will store the audit records and Alarms property also must be set to True. The Audit table fields are described in Appendix Table Audit.

Also the data servers can be defined for saving, for it must be **HasStoreActions** property to True, and consequently define the property (collection) **StoreActions**, from which we can access the Store Actions Editor that see below.

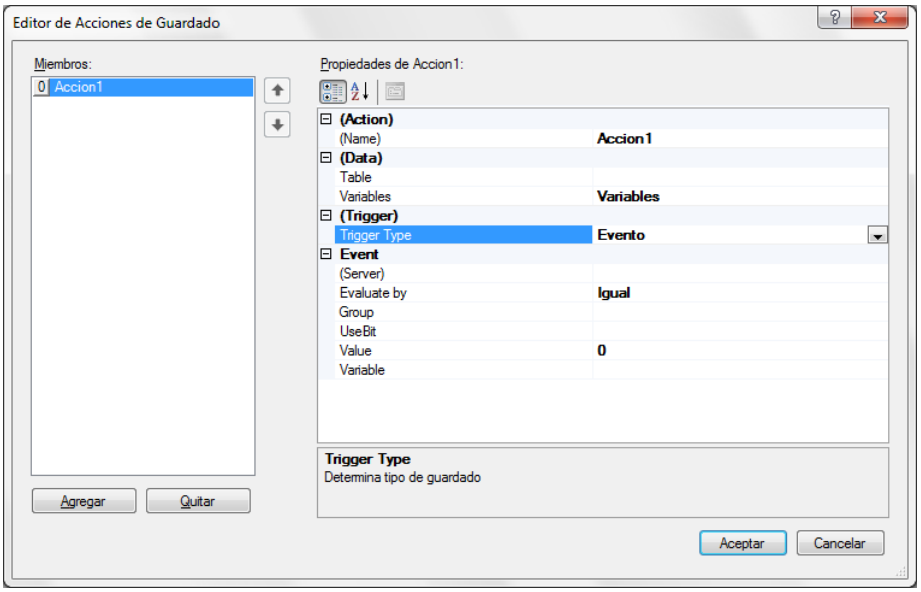


Illustration 63 – Storing Actions property window

In this editor we will define the name of the store action and the Trigger Type: Event or Cadence.

If Event is chosen, we can see that a variable is selected (with Server, Group and Variable) and as in the case of Alarms we should choose in Evaluate By, UseBit and Value the fact that it will trigger the save action.

If we choose Cadence, we have to specify how many seconds will pass between saving actions.

Moreover we need to define what values are stored in which table and what fields the table will be written in the table property, while the rest will be in the Saved

Variables Editor which we accessed from the property (collection) variables, we see then this editor.

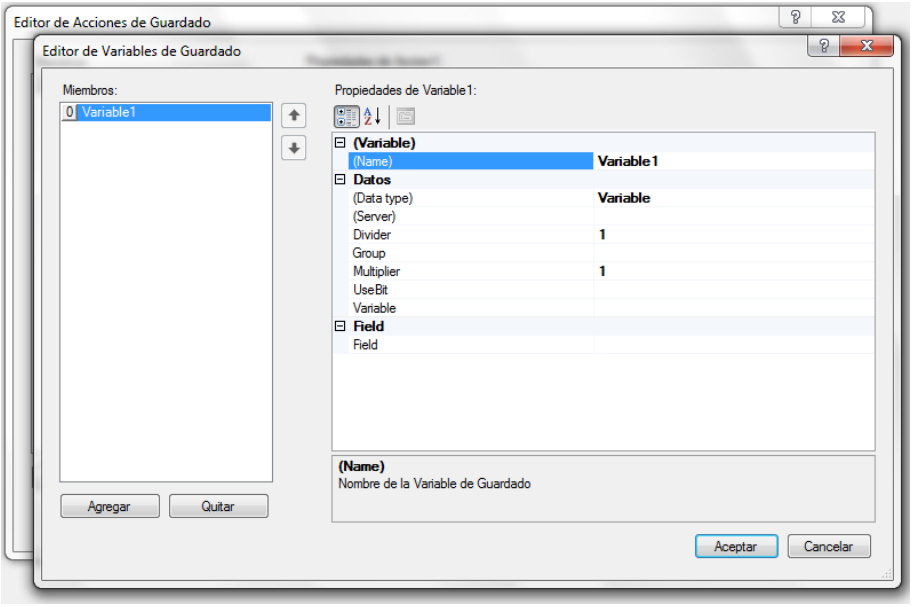


Illustration 64 – Storing Variables Editor

Here we define the fields you wish to write to the table and the type of data to be recorded in the property **field** will have the field name, the **Data type** property will allow us to save Data Variable values Scada, Date/Time field or constant, the rest of properties change depending on the data type.

It should be noted that if you later want to display or plot these data from the application Monitoriza it is necessary to have a field of type Date/Time.

DISPLAY HISTORIC DATA

Acimut Monitoriza allows storage of values in database and can easily show it to the user. It is only necessary to add a button on a form and with his Action property set to MostrarGrafica. During execution of the application, this will show us the form below.

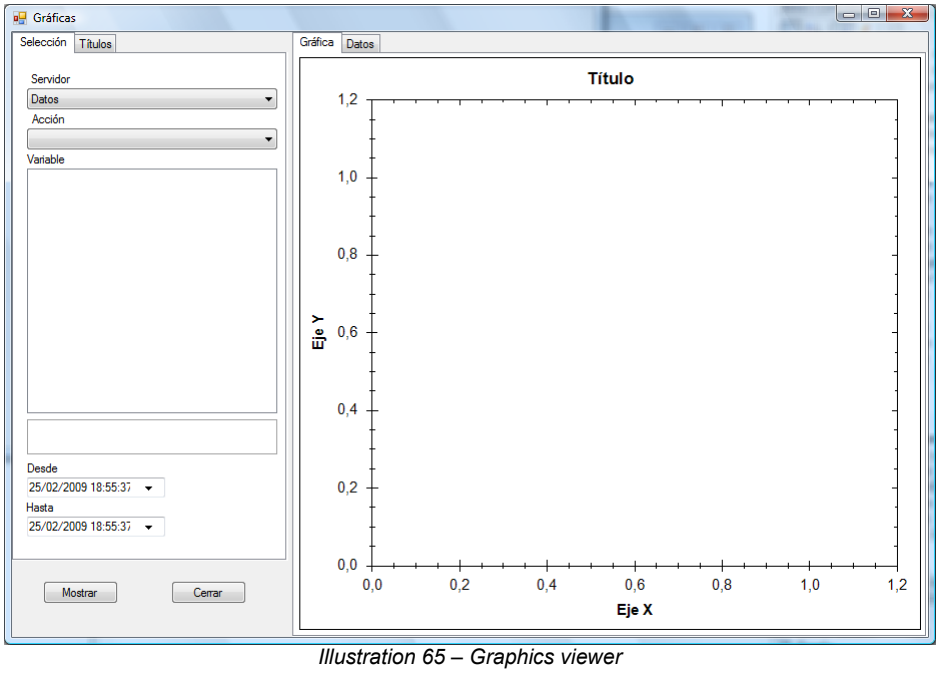


Illustration 65 – Graphics viewer

This form will allow us to see the values and plot them of the Store Action we choose.

DISPLAY HISTORIC ALARMS

Alarms stored in the database can be accessed from an application monitoriza with ease, for it is only necessary to add a button on a form in which his Action property is set to MostrarHistoricoAlarmas. During execution of the application, this will show us the form below.

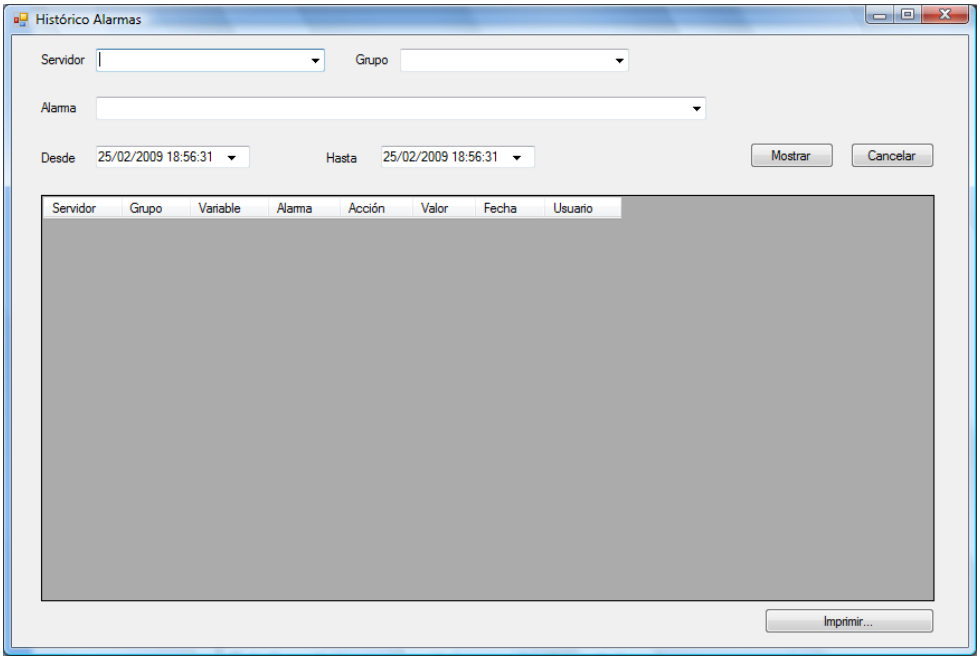


Illustration 66 – Alarms briefing

This form will allow us to see a history of alarms that we choose.

CONFIGURE SERVER COMMUNICATIONS

Monitoriza is a application that uses client / server communications technology, so it is necessary to properly configure the server, for this issue we have the next menu.

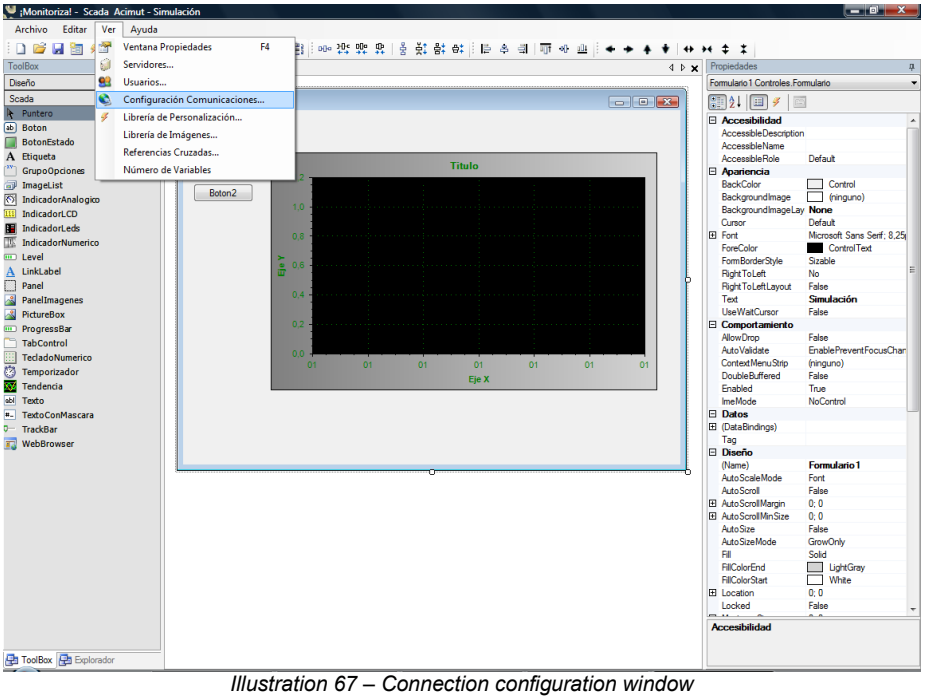
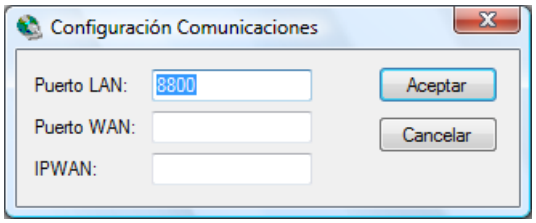


Illustration 67 – Connection configuration window

This option shows the following form.



On the one hand the Monitoriza server can be accessed from a LAN, the default port is the 8800 as shown in the LAN Port field, on the other hand, normally is necessary to access from a network like the Internet, for this, must choose a port in Port WAN and monitoriza must be informed of the public IP address to be on the Internet in the IPWAN field.

The required configuration of firewalls and routers is beyond the scope of this manual. It should be noted that in the case of WAN communications, HTTP protocol is used for convenience when configuring firewalls.

The server supports both LAN and WAN.

SERVER

Acimut's Monitoriza server is responsible for establishing and coordinating communications with both, devices and PLC's and also databases.

The server monitors the changes in the variables defined in real time and reports to clients, ie, the checkpoints, so they can present information to system user´s.

It is also responsible for evaluating the values of the variables to see if either of the conditions to trigger an alarm is true.

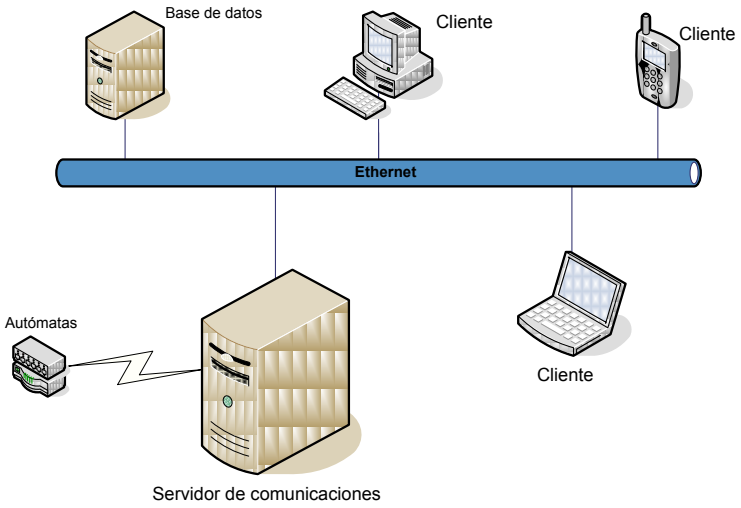


Ilustración 68 - Communications scheme

When server is started, application asks us to create a shortcut to the application.

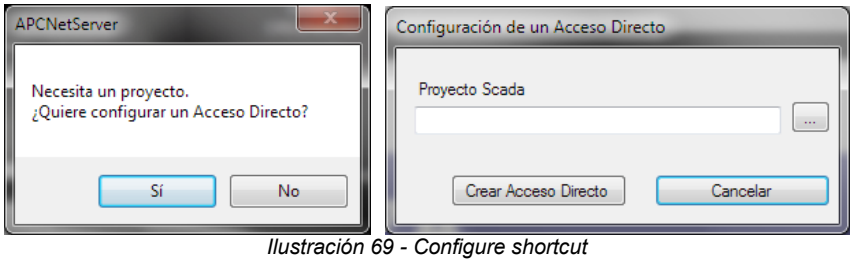


Ilustración 69 - Configure shortcut

This is because the server needs to know the project we want to run.

We must to specify a project '.*Scada*', created with the Monitoriza's Editor and clicking the 'Create Shortcut' on the desktop, we will create a link to the server that will remain as such:

"C: \ Program Files \ Acimut\ APCNetServer.exe" C: \ Documents and Settings \ User \ My Documents \ ServidorModBUSTCP.scada

To end the server application we have to click with right mouse button on the server icon located on the desktop notification bar (marked in red in the figure below)

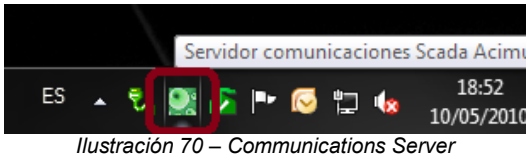


Ilustración 70 – Communications Server

And then, select the Exit option.

Using the icon marked in red in the figure below:

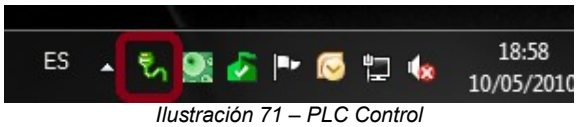


Ilustración 71 – PLC Control

We can see the status of communications of the Server with PLC's to check if it's correctly accessing each of the devices.

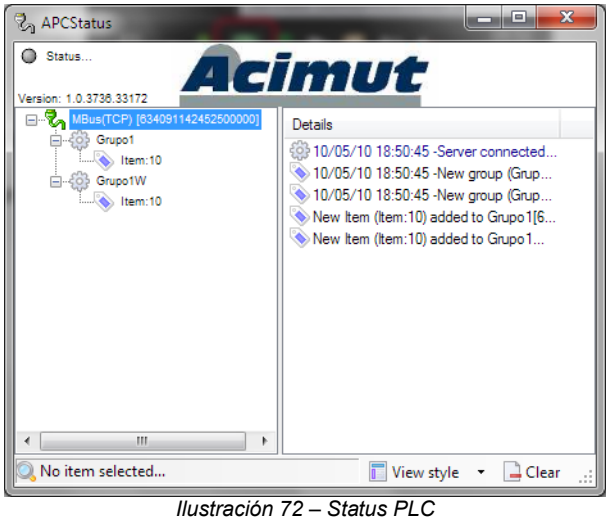


Ilustración 72 – Status PLC

CLIENT

Monitoriza client must refer to a server to show the execution of a project. The simplest case is when the server is on the same computer, does not require parameters, by default Monitoriza looks for a server on the local machine.

However we show here the parameters supported by the monitoriza client.

ServerName, Net and Port

ServerName is the name or IP address of the computer running the server
Port is the TCP port that runs the Monitoriza server, 8800 is the default.

Net can have two values:

WAN Communications used over the Internet (for example), use an HTTP protocol.
LAN Indicates communications using a LAN, it uses a TCP protocol.

EXTENSIBILITY AND PROGRAMMING

Although in most of the Monitoriza projects is not necessary to add programming, occasionally, by the complexity of what is intended, or because the developer feels more comfortable, it's necessary to have a tool that supports more advanced programming, in that case, we can say that Monitoriza is programmable in '.Net', that is, with all the power of C# and Visual Basic '.Net' from Microsoft, allowing both simple and elaborate user functions.

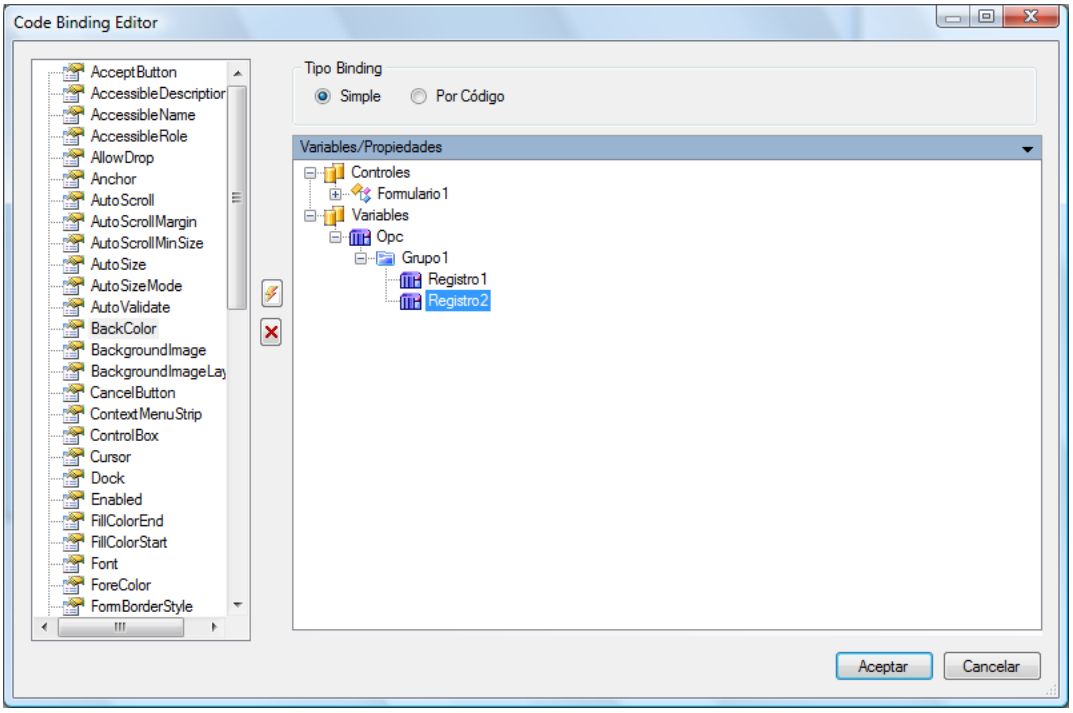
On the one hand we can code the assign of properties and events of the controls inside the own monitoriza editor and on the other hand we can assign external libraries developed in '.NET'.

Keep in mind that these functions are running on the client side of Monitoriza.

INTERNAL CODE "CodeBindings"

Monitoriza controls and forms have a number of properties, which can be allocated at design time. But sometimes we wish this allocation is dynamic depending on what is happening.

That's why Monitoriza controls have CodeBindings property, using this property we reach the next screen

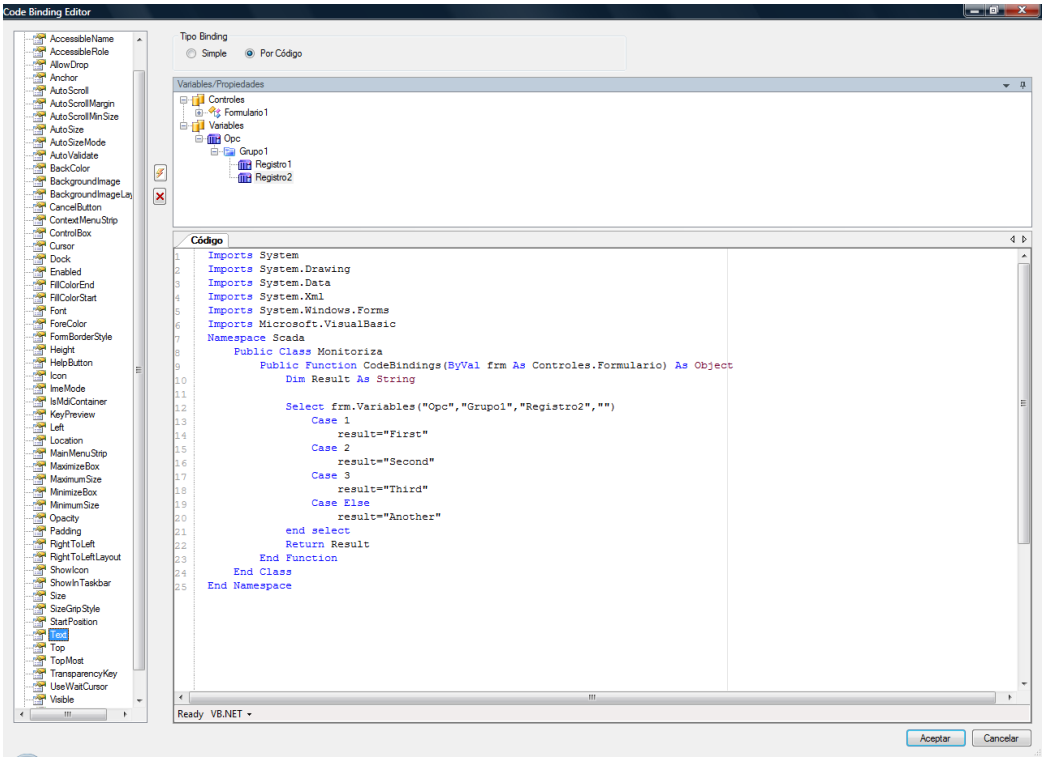


We can see that we have a list on the left with the properties of the selected control, these properties are those that can be allocated dynamically. In the picture we see that in this case, the BackColor is assigned, in execution time, the value of the

variable Registro2, this is a simple type binding and is associated with the button , to remove the associaton we have the button .

As well as values of variables, property values of other controls can also be associated.

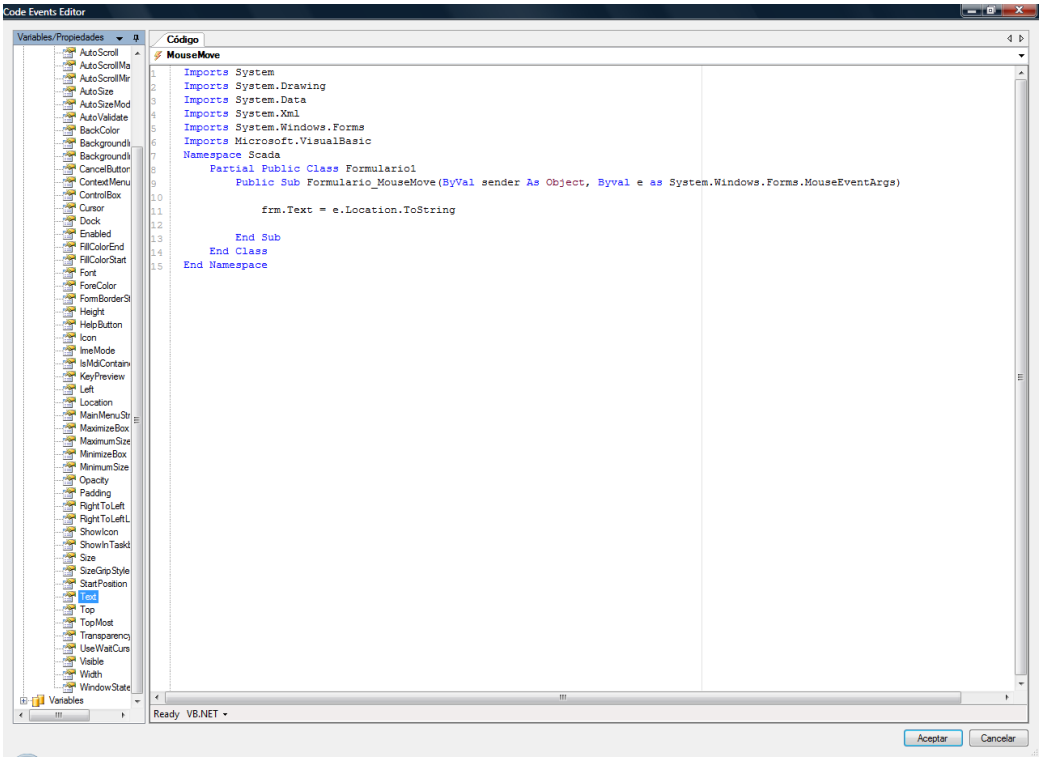
As the direct use of values of variables or properties do not always meet our expectations and we need something more elaborate, this is where we can choose a binding's type code, when we choose this type, display becomes as follows:



We will continue having the same functionality in the list on the left, but now we have a new code editing window in which we can develop a function, the value returned by the function is dynamically assigned to the chosen property. We can use the list of the top to drag and drop on the new code window and not have to write or properties or variables.

When making the association with the button , code will be compiled and reported for syntax errors. As we see in the picture above, we have created a procedure based on the value of a variable that returns us different values assigned to the Text property.

As mentioned, besides being able to dynamically assign values to control properties, we can execute code on events of the controls. Writing the code for these events can be achieved by double clicking on a control or by double clicking on the chosen event in the properties / events window and then Monitoriza will show the following window.



In this case the list on the left side helps us to drag and drop properties and variables on the code, and in the part of the code, we can navigate between the different events through the list of the top of the window. During execution of the event run this code.

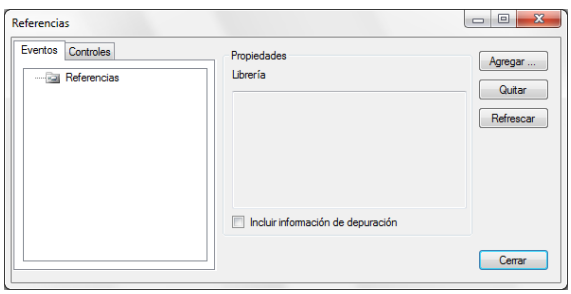
EXTERNAL CODE EVENTS

EVENT LIBRARY AND CONTROL LIBRARY

The libraries allow you to associate event of an external library code to an event Monitoriza control

Control Libraries, instead of events libraries, allow to incluye User Controls directly in the Monitoriza project, and we can use them like other native control in the application.

We can include referentes to DLL using the Customization Library in the view menu.



THE PROGRAMMING TOOL

The programming tool is Visual Studio ®, 2005 or later version, the programming language is anyone who is supported by Visual Studio ® and pleases the developer, Visual Basic, C #, etc..

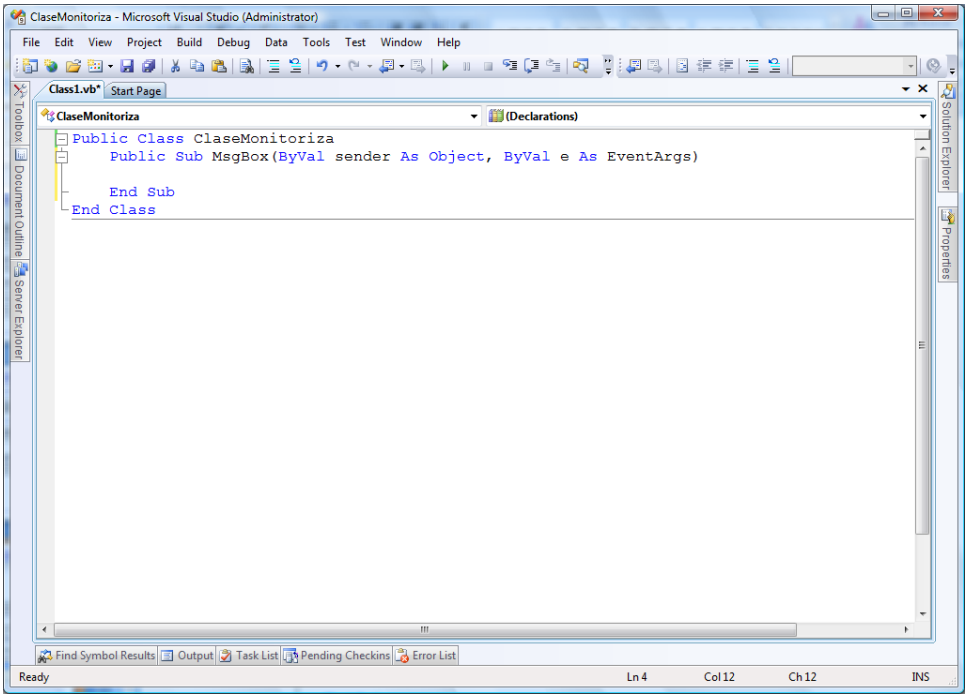
It should be noted that Microsoft provided free Express version of Visual Studio ® and is perfectly valid for use with Monitoriza.

In this document we will see examples developed in Visual Basic.

FUNCTIONS

Monitoriza uses .NET DLL projects to run functions; to create a project of this kind, from Visual Studio ® choose the option File-New Project menu and select a project type and Class Library, for example we will give it the name: *ClaseMonitoriza.*

There are a number of conditions that a library must meet to be used from Monitoriza: first, however the library can have any number of classes, Monitoriza can see only the **public** ones, so it must also be **public functions**. We see here an easy and usable code from Monitoriza.



An important observation is, that the functions we write, will be used from events of different Monitoriza objects, so they must contain the two arguments of the form:

```
Public Sub MsgBox(ByVal sender As Object, ByVal e As EventArgs)

End Sub
```

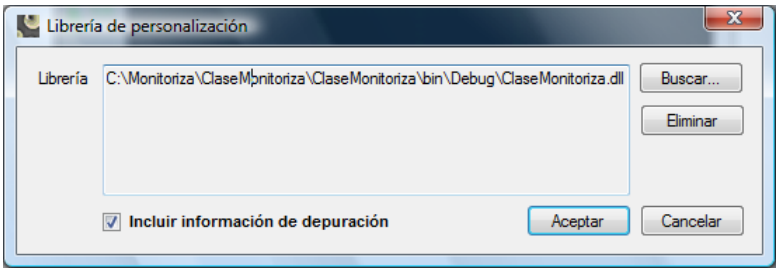
sender is the control, form, timer, etc. that triggers the event.
e is the parameter associated with the corresponding event.

Let's add some **code** to the function:

```
Public Class Monitoriza
    Public Sub MsgBox(ByVal sender As Object, ByVal e As EventArgs)
        Microsoft.VisualBasic.MsgBox("Hello I am " & sender.name.ToString)
    End Sub
End Class
```

To use this function we have written, we must generate the library and add it to the Monitoriza project, for this issue, in Visual Studio ® choose the menu option "Generate" ClaseMonitoriza.

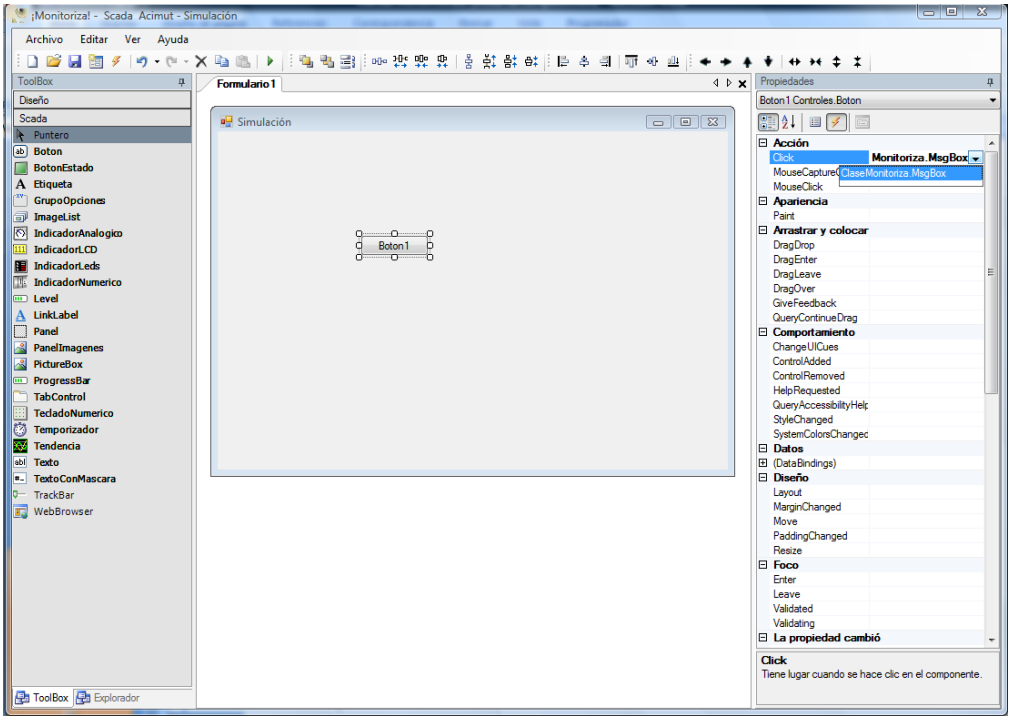
In the Monitoriza Project Editor, we must generate a new project and choose the menu option View- Customization Library; in the next window we look for and assign the library to the project.



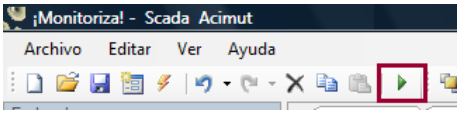
We have chosen to include **debugging information**, we will see in the next step how to debug a function.

Keep in mind that we will not need to distribute the library as an additional file, library is embedded in the Monitoriza project, this is why every time you change the library you need to refresh it, using the previous screen and accepting the changes.

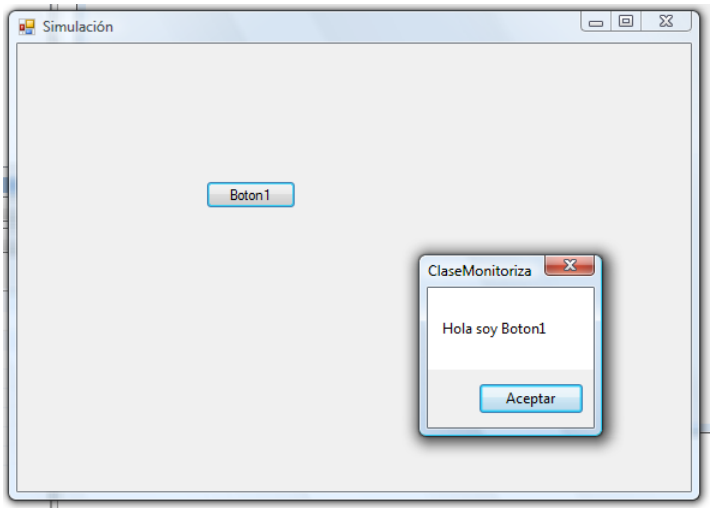
Once accepted the previous option and on a new form (declared as default) we will place a button and assign the Click event ClaseMonitoriza.MsgBox, let's see how it looks.



Now we will implement the project with the Play button

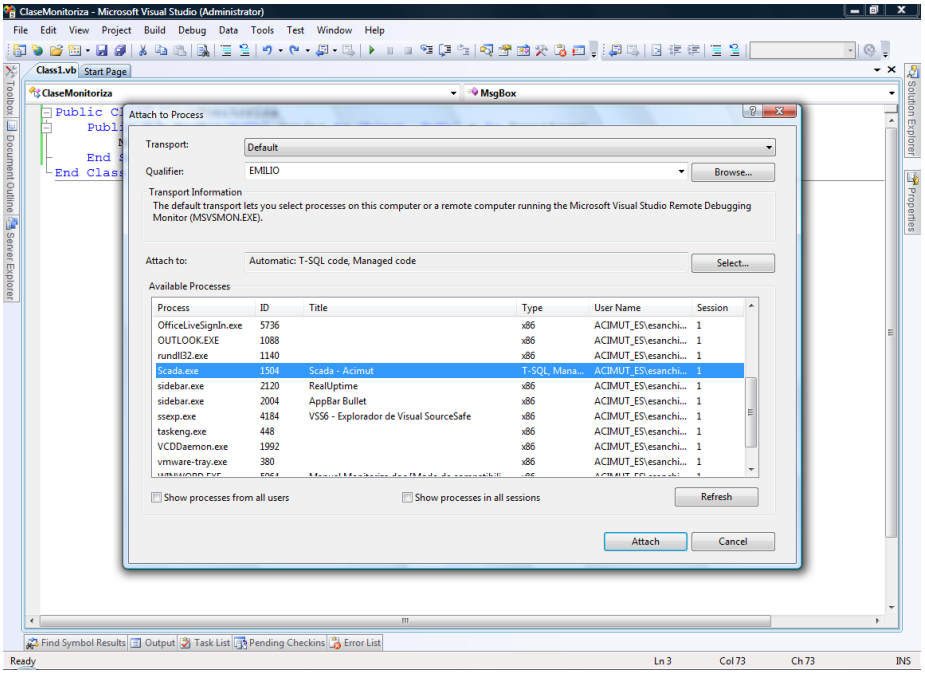


It will bring up the project in simulation mode and click on the button to get the following

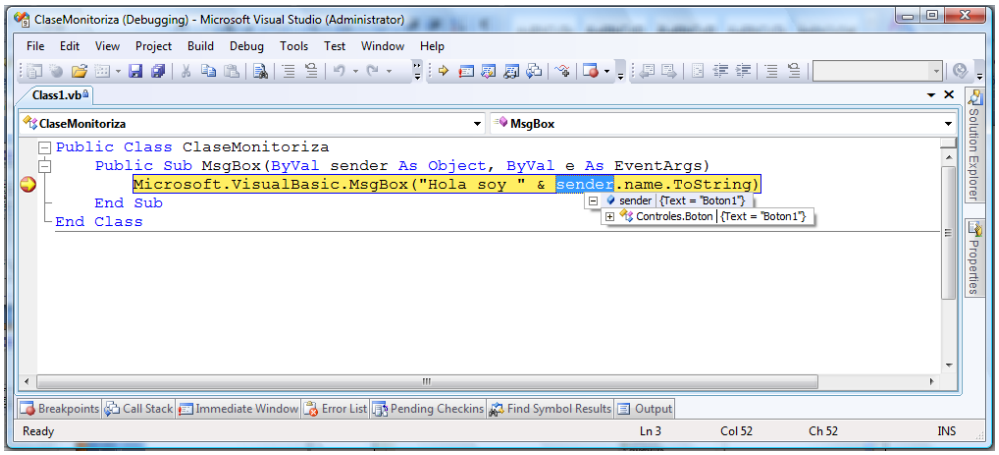


As previously said, you can debug the library (must be generated with debug information and include debugging information in the project).

The first step is to open with Visual Studio ® project the library, then we start the project and again on Visual Studio ® choose the menu option Tools-Attach to Process and in a screen like this choose Scada.exe process.

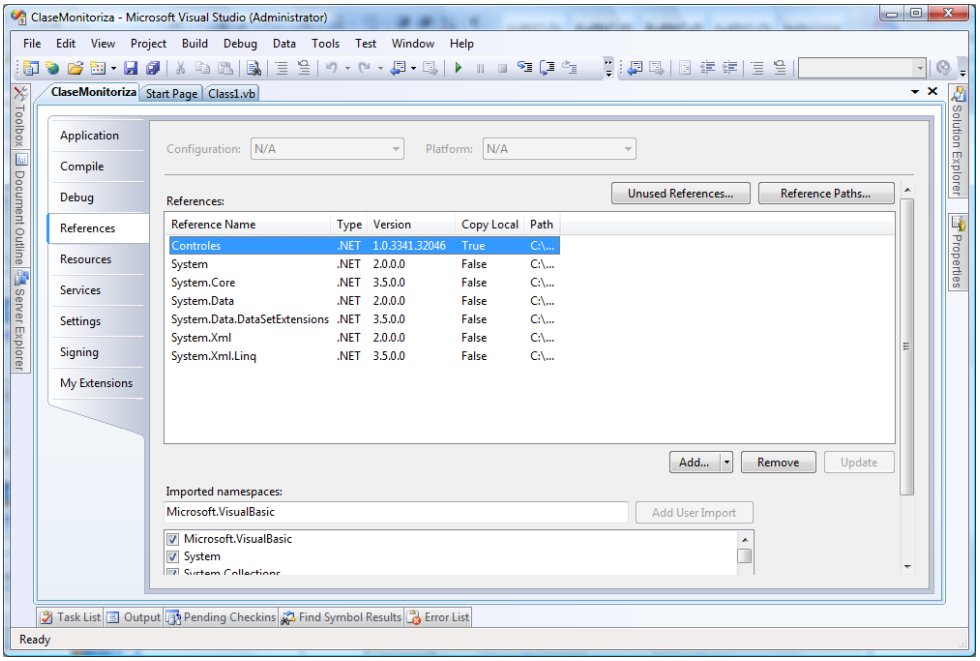


We also have set a breakpoint in our function; when we click the button code stops and will show in Visual Studio ® where we can examine the contents of variables, step through and any other functionality that give us Visual Studio ®.



ACCESSING MONITORIZA CONTROLS PROPERTIES

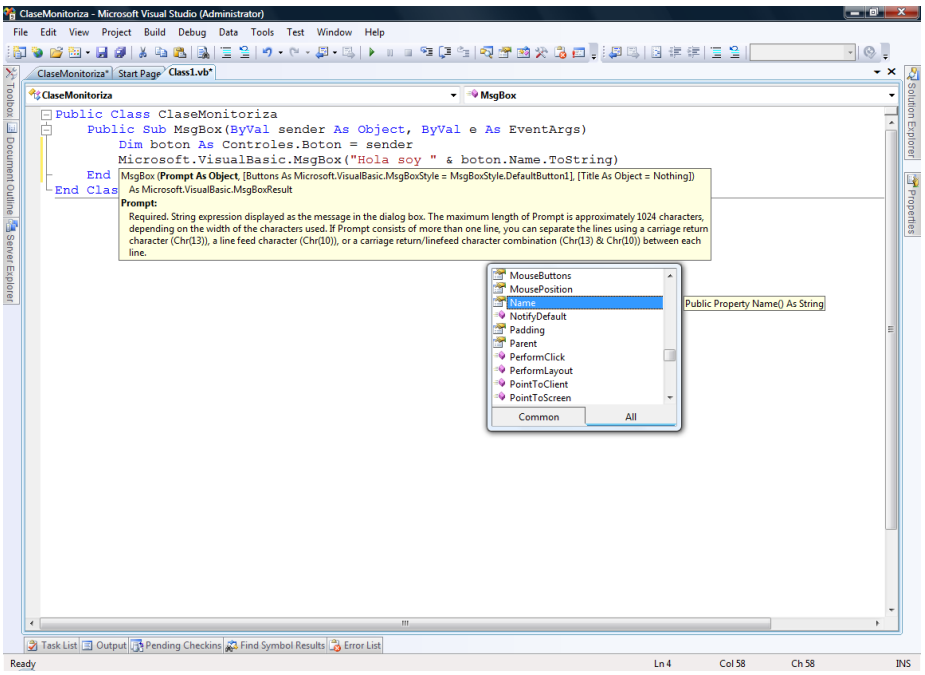
As seen in the example above, when we have to get or change properties of a control, we do not have Intellisense, but it is also easy to fix and have all the help provided by Visual Studio ®.
To do this, in the project we will add a reference to the library Controles.dll that is in the directory where you installed the Client Monitor (Scada.exe).



Now a small code change will provide the IntelliSense. Following is the new code and its effect on the editor.

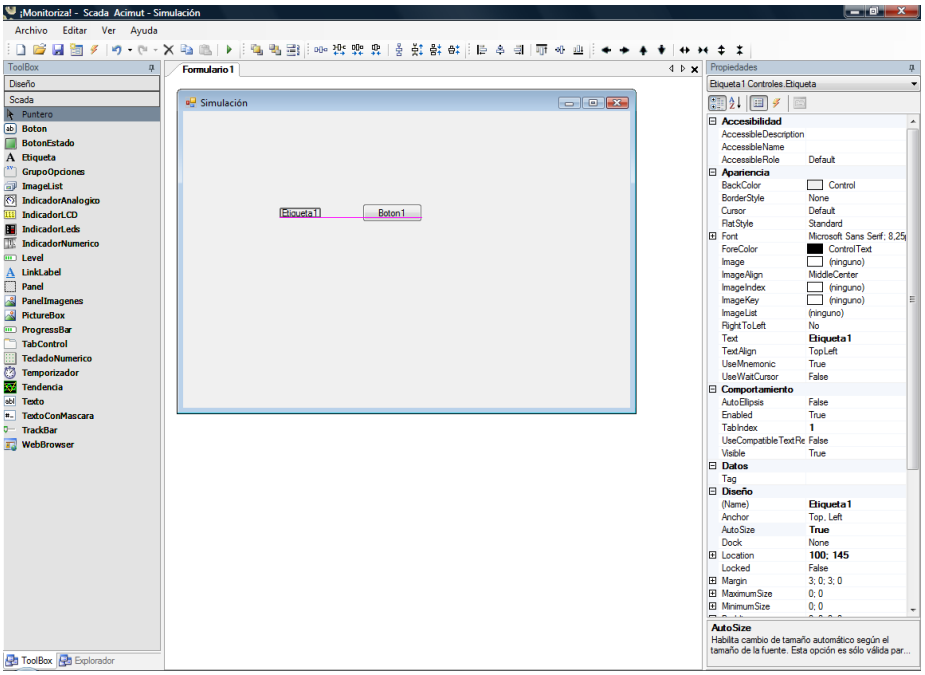
```
Public Class Monitoriza
    Public Sub MsgBox(ByVal sender As Object, ByVal e As EventArgs)
        Dim boton As Controles.Boton = sender
        Microsoft.VisualBasic.MsgBox("Hola soy " & boton.Name.ToString)
```

```
    End Sub
End Class
```



Another peculiarity is the possibility, from a control, to achieve the form or other control contained herein.

To show this, add a Label to the form of the project and modify the code. The form would be as follows:



We must modify the code this way:

```
Public Class Monitoriza
    Public Sub MsgBox(ByVal sender As Object, ByVal e As EventArgs)
        Dim boton As Controles.Boton = sender

        'le cambiamos el título al formulario
        boton.TopLevelControl.Text = "Hemos hecho clic"

        'accedemos a otro control en el formulario
        Dim etiqueta As Windows.Forms.Label
        etiqueta = boton.TopLevelControl.Controls("Label1")
        etiqueta.Text = "Aqui también cambiamos el texto"

    End Sub
End Class
```

For these changes to take effect; first we must generate the new library and then reload the project before running.

ACCESING TO VARIABLES AND COMPONENTS

An important element to highlight is ability for accessing from the code to the values of variables, these are found in a Dictionary property of the Form, also allows access to components (such as the timer) on a property similar to Controls called Components, we see an example how easy access to these two properties of the form.

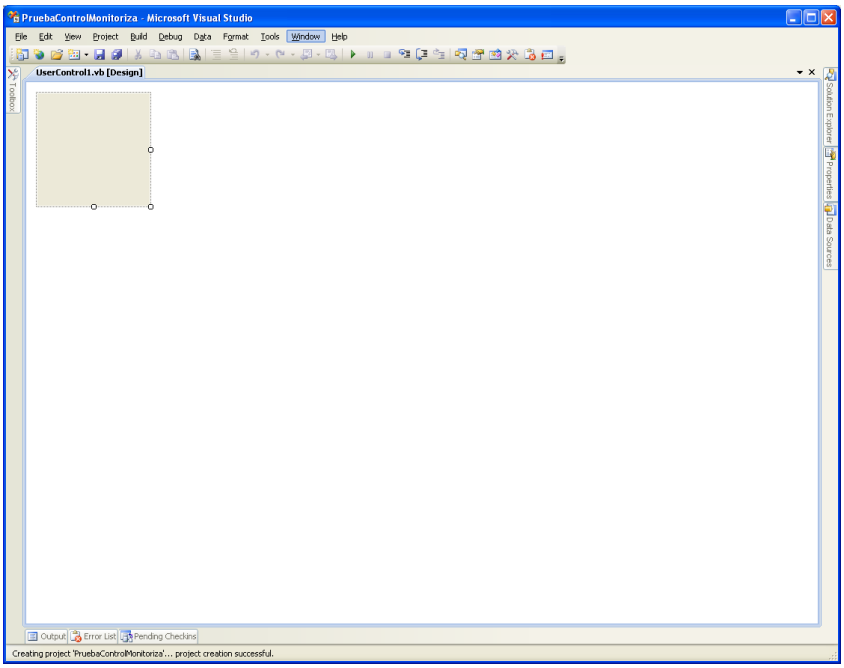
When we access the property variables, we indicate what variable we get the value from with the arguments: server name, group name and the name of the variable, separated by commas.

```
Public Class Monitoriza
    Public Sub MsgBox(ByVal sender As Object, ByVal e As EventArgs)
        Dim boton As Controles.Boton = sender

 'le cambiamos el título al formulario
        boton.TopLevelControl.Text = "Hemos hecho clic"

        'accedemos a otro control en el formulario
        Dim etiqueta As Windows.Forms.Label
        etiqueta = boton.TopLevelControl.Controls("Label1")
        etiqueta.Text = "Aqui también cambiamos el texto"

        'accedemos a las variables y componentes

        'definimos el formulario
        Dim formul As Controles.Formulario
        formul = CType(boton.TopLevelControl, Controles.Formulario)

        'accedemos a las variables
        Microsoft.VisualBasic.MsgBox("Variables: " & formul.Variables.Count)
        Microsoft.VisualBasic.MsgBox("Variable(""Servidor,Grupo1,Registro1"")=
" & _
        formul.Variables("Servidor,Grupo1,Registro1"))

        'accedemos a los componentes
        Microsoft.VisualBasic.MsgBox("Componentes:              "             &
formul.Componentes.Count)
        Dim temp As Controles.Temporizador
        temp = formul.Componentes("Temporizador1")
        Microsoft.VisualBasic.MsgBox("Interval   de   Temporizador1:    "    &
temp.Interval)
    End Sub
End Class
```

CONTROL LIBRARY

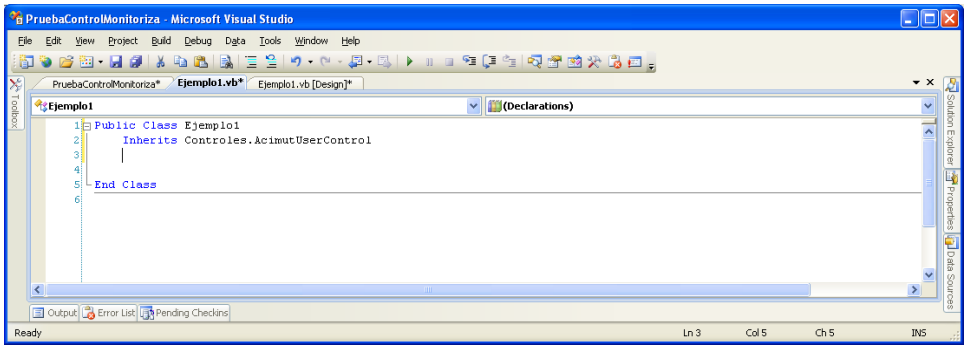
In the same way that we can bind external code to the events in the monitoriza controls, also is possible to include controls designed in an external application inside Monitoriza itself. These are **UserControls**.

In order to use these controls, we must create a Windows Forms Control Library in Visual Studio.



In this library we must add the controls we want to use later in Monitoriza.

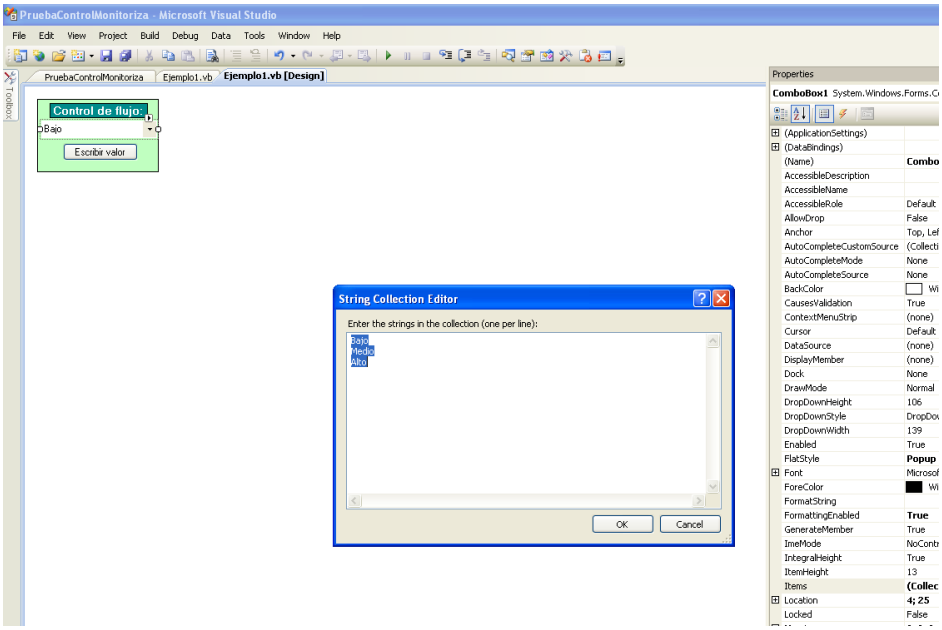
In order to use these controls in Monitoriza we must to add a reference to Monitoriza's Controles.dll (Project – Add reference...) and in each control we must to indicate that the controls must inherit the main class from Controles.AcimutUserControl.



Visual Studio throw a warning indicating that is not possible to inherit AcimutUserControl and UserControl at the same time, then we must indicate to inherit only AcimutUserControl.

Let's see a simple example of UserControl creation and the latar using in Monitoriza.

Once created the Project, we add a control named Ejemplo1. In this control we add a label, a button and a combobox with 3 items:



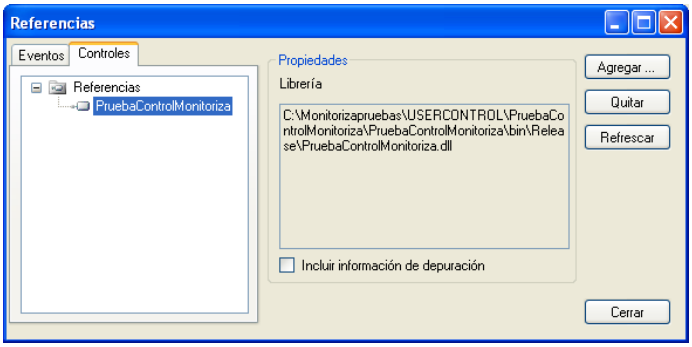
Let's add some code to the Project:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim ValorAEscribir As Integer = 0
        'Seleccionamos el valor a escribir en la variable en funcion de lo que
se escoja en el combo
        'los 3 valores permitidos son BAJO,MEDIO y ALTO.
        Select Case Me.ComboBox1.Text
            Case "Bajo"
                ValorAEscribir = 10
                 Me.BackColor = Color.Coral
            Case "Medio"
                ValorAEscribir = 50
                 Me.BackColor = Color.Yellow
            Case "Alto"
                ValorAEscribir = 100
                 Me.BackColor = Color.Green
            Case Else
                ValorAEscribir = 0
        End Select
        'Asignamos el objeto frm al formulario padre, que en este caso es el
formulario de Monitoriza
        'que esté usando este usercontrol.
        Dim frm As Controles.Formulario = Me.TopLevelControl
        frm.Variables(Me.Servidor, Me.Grupo, Me.Variable) = ValorAEscribir
End Sub
```

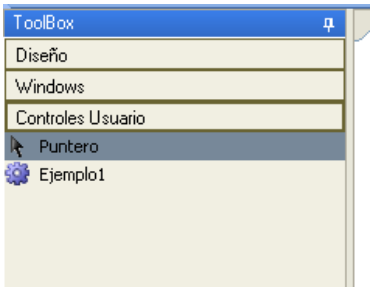
By this way, when we click the button, a value is binded to the variable asigned to the control in the Monitoriza Project. That is:

```
        frm.Variables(Me.Servidor, Me.Grupo, Me.Variable) = ValorAEscribir
```

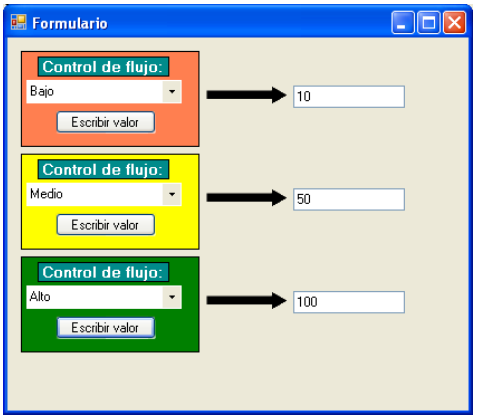
Once we have the control the way we want and the project is builded (Project – Build), we must include the DLL in Monitoriza, for this we must use the References form:



Validating the form, in the toolbox appears a new control in the User Controls Group:



This control can be used like each other native in Monitoriza. Once included in the desired form, we must compliment the Scada properties (Server, Group, Variable, etc...) and the control is ready to use.

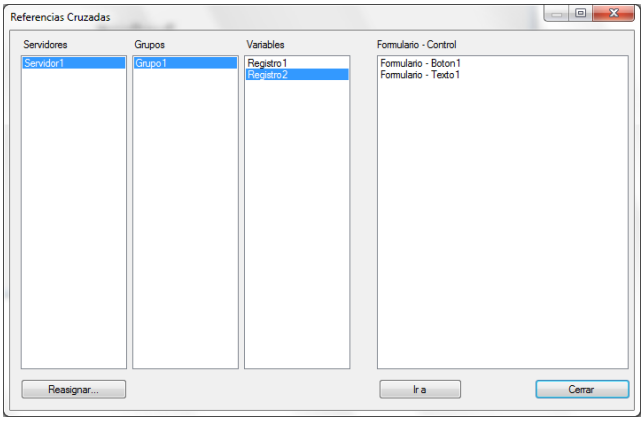


UserControls can access to variables and components from Monitoriza like Events Dll do. To do this we must use the same method indicated in Accesing variables and components.

CROSS REFERENCES

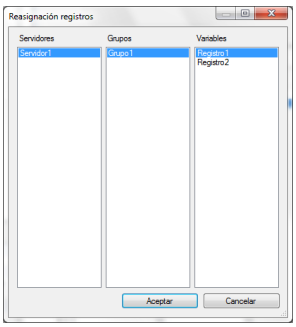
Acimut Monitoriza includes a useful application named Cross References, in wich is possible to check where are used the variables of the project and if is necessary reasign the variables binded to the controls.

If we want to know where is used a variable (in wich controls is used, across the entire application) we can use this application, that is located in the View menu.



First we must to select a **server** in the first column, when we select it, in the second column appears all the groups related to this server, once selected the **group** we want, in the next column appears the different **variables** that conform this group, and once again if we select the register we want to inspect, in the last column appears all the **controls** (form and control) that use the selected variable. We can see the form itself using the *'Go to'* button.

In case we want to change the variable that is assigned to these controls we can use the Reasign option. In order to reasign variables we must select the desired controls in the last column (the controls to change), and click on *'Reasign...'* button, a new window appears:



In this window, we must select the new variable to use in the selected controls. This may cause that all previously selected controls will change the bindings associated with them.

APPENDICE

ALARMS TABLE

Following is the table definition to storage alarms as defined in SQL Server.

```
CREATE TABLE [Alarmas] (
    [ID] [int] IDENTITY (1, 1) NOT NULL,
    [Servidor] [varchar] (50) NULL,
    [Grupo] [varchar] (50) NULL ,
    [Variable] [varchar] (50) NULL ,
    [TextoAlarma] [varchar] (250) NULL ,
    [Accion] [varchar] (50) NULL ,
    [Valor] [varchar] (50) NULL ,
    [Usuario] [varchar] (50) NULL ,
    [FechaHora] [datetime] NULL ,
    [Numerico] [int] NULL ,
    [Sencillo] [float] NULL
) ON [PRIMARY]
GO

ALTER TABLE [Alarmas] WITH NOCHECK ADD
    CONSTRAINT [PK_Alarmas] PRIMARY KEY CLUSTERED
    (
          [ID]
    )  ON [PRIMARY]
GO

 CREATE INDEX [IX_Alarmas] ON [Alarmas]([Servidor], [Grupo],
[Variable], [TextoAlarma]) ON [PRIMARY]
GO

 CREATE INDEX [IX_Alarmas_1] ON [Alarmas]([FechaHora]) ON [PRIMARY]
GO
```

AUDIT TABLE

Following is the table definition to storage audit records as defined in SQL Server.

```
CREATE TABLE [dbo].[Auditoria] (
            [ID] [int] IDENTITY (1, 1) NOT NULL ,
            [Computer] [varchar] (50) NULL ,
            [UserName] [varchar] (50) NULL ,
            [ActionAudited] [varchar] (50) NULL ,
            [Parameter1] [varchar] (250) NULL ,
            [Parameter2] [varchar] (250) NULL ,
            [Parameter3] [varchar] (250) NULL ,
            [DateTime] [datetime] NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Auditoria] ADD
            CONSTRAINT [PK_Auditoria] PRIMARY KEY  CLUSTERED
            (
                        [ID]
            )  ON [PRIMARY]
GO
```

MODBUS Registers

Modbus Registers Range

PLCs and Modbus devices store information in registers.
The registers are divided into blocks of 65536 units. Monitoriza uses ranges between 0 and 65535. Different types of data can be stored in each block.

If we want to read or write to a Modbus register, we must have specific information about the module or PLC, to find out which group of Modbus data is involved and therefore what type of group we must use in Monitoriza.

The Modbus registers have 16 bits (one word), while marks (coils) are 1-bit data. Monitoriza provide the ability to read directly double words and single (float) registers.

What follows next is a table of the ranges of Modbus registers and the group type to be used in Monitoriza:

Modbus Register Range	Modbus	Start Digit	Monitoriza Modbus Function Code
000001 – 065536	**Output Coils (Rango de 0 - 65535)** (For example, Digital input points and digital output points.)	0	ReadCoilStatus WriteCoil
100001 – 165536	**Input Discretes (Rango de 0 – 65535)** (For example, Digital input points.)	1	ReadInputStatus (Solo lectura)
300001 – 365536	**Input Registers (Rango de 0 - 65535)** (For example, Digital input points and analog input points.)	3	ReadInputRegisters (Solo lectura)
400001 – 465536	**Holding Registers (Rango de 0 - 65535)** (For example, Digital input points, digital output points, analog inputs, and analog output points.)	4	ReadHoldingRegisters WriteMultipleRegisters

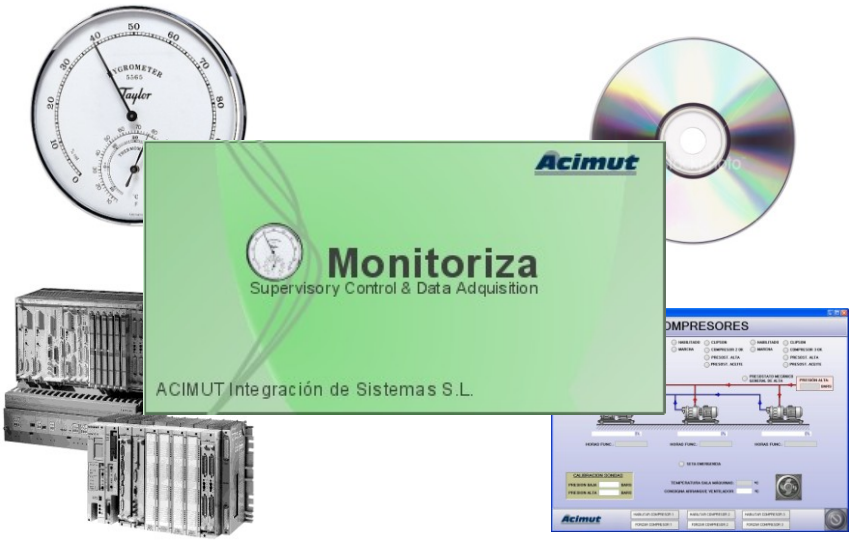
ModBUS Functions used by Monitoriza

The Modbus protocol uses different functions for each record type, functions used in Monitoriza are in the table below.

Modbus Function Code	Funcionalidad
FC1	Read Coil Status
FC2	Read Input Status
FC3	Read Holding Registers
FC4	Read Input Registers
FC5	Write Coil
FC16	Write Multiple Registers

Functions not used are:

FC6	Write Single Register
FC15	Write Multiple Coils



Review October 2012
Acimut Integración de Sistemas