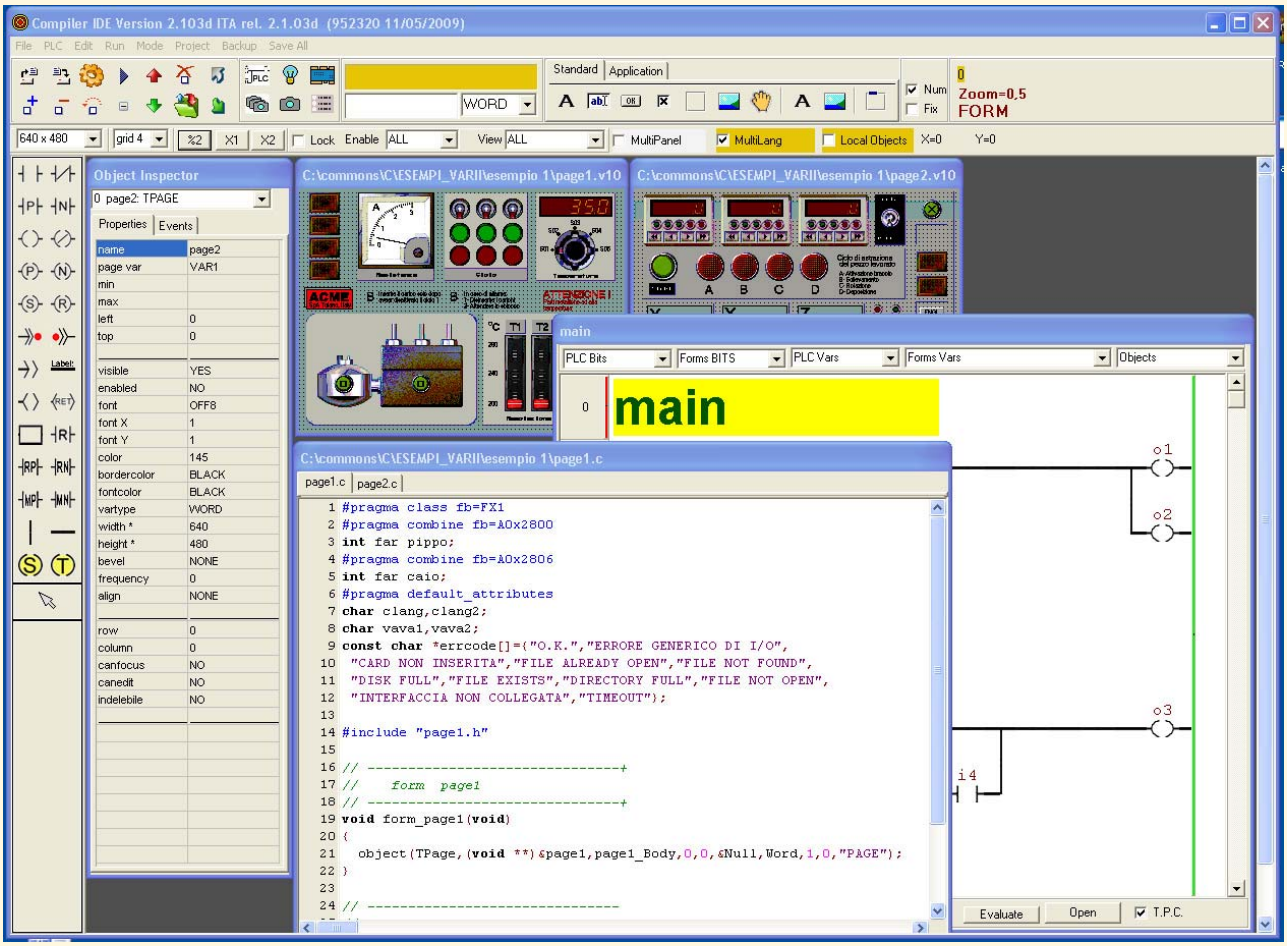
Proteus ES

Development Platform

Proteus

User Manual

Ver. 2009



PROTEUS
Divinity of Greek mythology, son of the Ocean and Teti; able to assume any form.

PROTEUS it is a complete system, composed from tool of development software and a series of modules hardware, able to allow to an express development and working realization of a finished product, also without a deepened acquaintance of programming, starting from a preconstituted modules. The Proteus system introduces some revolutionary aspects, regarding the classic way of development of an application.

- Developing an application PLC with graphical interface, the two members of project (the PLC and the graphical interface) come developed in a single program, avoiding protocols of talk, passages of parameters and other troubles, normally tied to the development of a PLC classic.
-
- A project created with Proteus, without to change nothing, but simply redefining the target on the menu of the project, it can to run on completely various configurations hardware, as an example: on a PLC S400 with monitor TFT, or display alphanumeric, or on a PC with operating system Windows Xp, or Windows CE or Linux, connected with the PLC remote S400.

- Proteus allows the development of applications for PLC without display, with display LCD 4 lines X 20 characters, with monitor TFT from 5”, 10”, 15”, with keyboard, mouse, touch-screen or on PC, in an only atmosphere of programming.

Introduction

With Proteus we can to create:

- simple programs of PLC written in **Ladder language**.
- simple programs of PLC written in **language C**.
- PLC programs using both languages over sayings. Containing programs with a graphical interface to objects and a program PLC

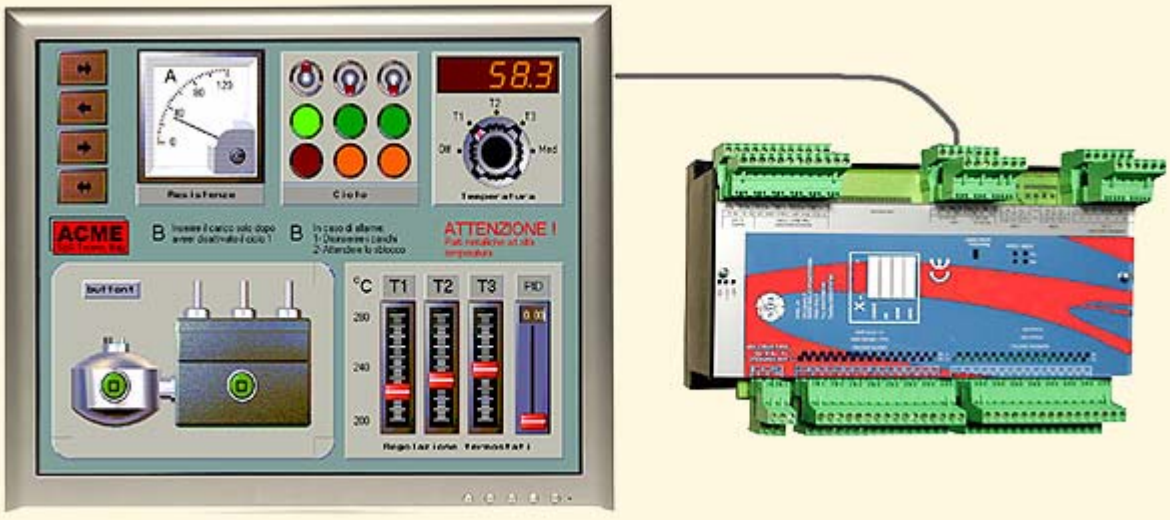
The main characteristic is the extreme simplicity of use and the rapidity with which it is possible to create and to test an application.
It is possible, in a short time and with some click of the mouse, beginning from to create a zero application with graphical interface, to try of the operation on the PC, and to send it in execution on the final hardware.
Proteus has been born for the implementation of electronic control panels with interface touch-screen, that its primary function remains, but can be used for the creation of any type of program for the S400 controls and bys-product, with and without monitor touch-screen.
It is possible moreover to make to execute application (graphical + PLC) on a PC, being used or a more PLC S400 for the physical interface with the machine.
In phase of test can to simulate the machine with an appropriate program on PC.
In a project developed with Proteus the programmed part in Ladder language is separable. This renders possible to compile the application, leaving to the electrician the possibility to modify on the field and to compile part PLC separately.
Contrarily to the philosophy current of the PLC, in which part PLC (resident on the PLC) and the graphical interface (of usual resident on a PC) is two very distinguished programs, created separately and that speak between they by means of serial protocol, the applications created with Proteus fuse in an only final object part PLC and the part graphical interface.
In this way:
- the graphical interface has a complete vision and control of the PLC.
- The two task they turn on the same hardware:
  - Both on the PLC (in this way for the graphical part a PC is not necessary but a simple monitor TFT is sufficient)
  - Entrambi sul PC (in questo modo viene sfruttata al massimo la velocita' di esecuzione del programma, che e' caricata in una **estensione Real-Time** indipendente da Windows, e l' hardware s400 contiene il programma standard di interfaccia input-output)

In any case, entire program (graphical and PLC) are contained in a single project, in which the two objects they are looked at mutual, everyone have the immediate control of the variable ones of the other, and talk protocols are not necessary, avoiding the troubles tied to the double programming and the relative delays and the other problems.

1– PROTEUS used to create an ELECTRONIC PANEL

- acquaintance of necessary programming: programming to contacts (Ladder, LD).

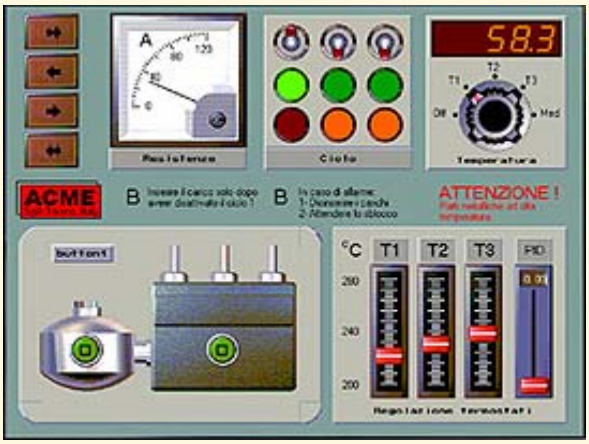
Proteus has been created primarily for the development of electronic panels. An electronic panel consists in an endowed monitor TFT of touch-screen, connected with a PLC, that it emulates the functions of the component classics electromechanical presents on the machine (interrupting, lamps spy, trimmers, ammeters etc…), It carries out moreover the functions of others members of usual presents on the machine (posizionatori, visual display units, thermometers, PID). Such members appear on the monitor as animated designs and are maneuverable with a simple touch of the fingers on the touch-screen.
A fundamental part of the project is the part PLC, programmabile in language LADDER or structured language (c). Such section concurs to connect between they and with the external world the members over sayings (interrupting, positionator etc…), carrying out moreover the function of the PLC of the machine. Part PLC is the only one that has need of programming. The development of the graphical part of the panel and the placing of the members, are carried out with of the simple operations of copy-glue like in a design program. The introduced members are completely pre-programmed and totally working, and it does not remain that to connect them between they through the program of the PLC.
In the library of the component included in the installation we can find:
-Switches, push-buttons, leds, lamps spy
-Potenziometers rotating and sliders

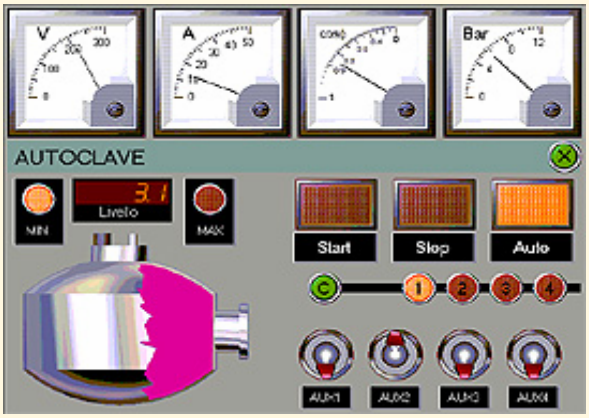


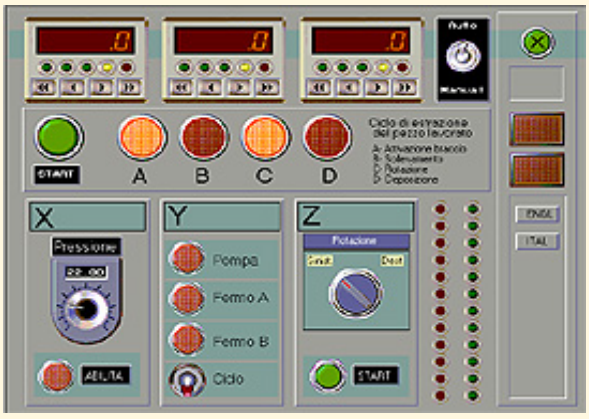
Possible configuration of an electronic panel: Monitor TFT 1024 x 768 15” + PLC S400. In this configuration the PLC has 32 inputs and 32 outputs on-off, 8 analogic inputs, 4 outputs type them, 4 encoders, 4 seriali lines, interface CAN-OPEN and puó to control 4 traditional positionings + others on line CAN-OPEN.

-Ammeters, Voltmeters, Electric power meters, Cosfimetri, gauges, pointers RPM and other instruments of visualization, or analogic that types them.
-Posizionators, interpolators, Quasar, Mesar and other blocks work evolven them.

Photos and designs (also animated) created with external programs of diagram can be imported.







Example of various pages for the electronic panel. It is passed from a page to the other page pressing a button on the panel.

At any moment of the development we can **test the developed part of program** and try the functions of the PLC, with a simple click of the mouse, emulating the panel and the PLC on the PC with which we are developing the application.
Once arranged all the components and written program PLC, with a click of the mouse, we can generate the final program and send it to the hardware.
The electronic panel is modifiable and upgradable at any moment, simply compiling the program. Being all the project contained in an only final program, the modernization operation is elementary, and can be made with the simple insertion of a pen USB, directly on the machine.
On the contrary of the traditional electromechanical panel instruments, the electronic panel can be organized being in more pages, and to visualize of time in time only the functions that interest, can show at call synoptic and pages of alarms, it concurs with extreme simplicity and with cost the zero addition of new instruments and the modification of the layout, and on hand allows to have an extensive range of members (push-buttons, switches, visual display units, ammeters…) holding to warehouse a single voice (the panel).
It is possible moreover to visualize the written ones on the panel in various languages, passing from a **language** to the other instantaneously.

2– Advanced Programmation

Proteus can be used to create programs for all product of the Syel series S.

The menu of the project concurs to specify esactly the target, than can be a blind material (without display), or with display alphanumeric or monitor TFT of several formats with keyboard or touch-screen.

The programming uses the language ANSI C, of which the functions are implemented all standard, a series of specific functions for the machine, in a position to managing the memory Flash, the serial modules of expansion and CAN, the peripheral CAN-OPEN, the task, the timers, the encoders, the inputs and the outputs type them and analogic, the serial doors, the memories of mass USB and other.

Applications multipage, using can be created the objects precast concrete slabs, the members (label, edit, bitmap, timer etc) also in form of carriers and matrices. It is possible the programming multitask, the personalized management of the interrupts, the serial protocols and the peripheral CAN-OPEN.

The project can be divided in two parts: one, written in language C, and one section separated, programmed in language to contacts, than can be modified and adapted from the installator of the machine, without to touch the rest of the program.
When a new project is created comes automatically generated a series of files, initially empty, that they constitute the project:

Common.h In which they come declared variable the total ones and the functions that will be used in the several pages and the rest of the program. In these rows we can also develop the body of the functions that do not make reference to objects of the pages (in this point of the project, the pages and the relative members still have not been declared).

Commonplc.h Here we can develop the body of the functions that use variable objects and of the pages.

Start.c This code makes part of the main of the program, and contains the initialization code, that is all this that it goes made before that the program sends in execution the first page.

Plc.c it contains the code in language C of task of the cycle the machine. It must contain an open code (than it is not closed in a loop) that will executed, with to the code generated from the compilation of the program written in language to contacts, in a task separated, timesharing with the interface code man-machine (the code of the graphical pages).

Lang2.h it contains the definition of the strings. It is generated automatically from the compiler, and can be edited and modified in order to make to appear the written ones in various languages.

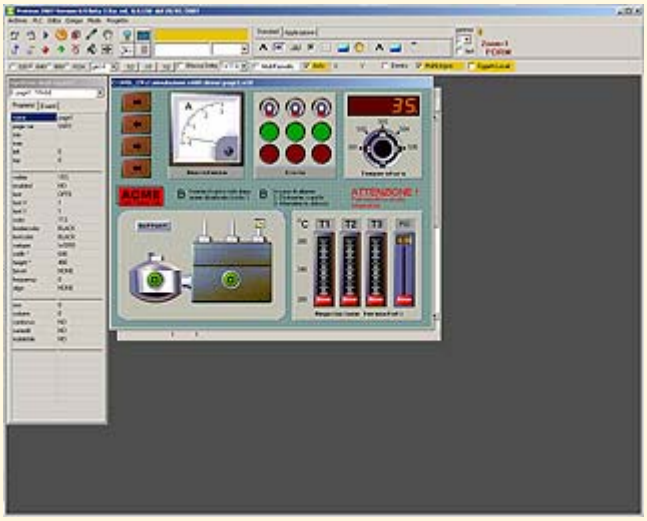
Pagexx.c It is the code of the pages, and comes shown in the programming window when the graphical page correspondent is selected. It concurs to write the code of the events, interactively with the graphical page.

Besides the bookcases standard, in order to develop the applicativo program we on hand have two modules of bookcase (whose headers they are x80.h and x80_obj.h) that they contain a series of created functions in order to use the peripheral ones of the PLC (encoders, seriali doors, can, usb etc.) and in order to manipulate the objects of the graphical pages.

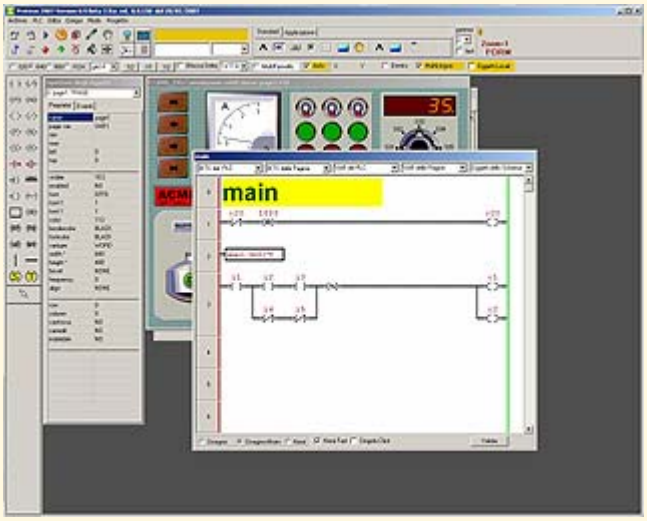
All the functions of these bookcases are described, with the relative examples, in the programming manual.

Proteus is currently the instrument used inner in SYEL to develop the new applications for the derived product S Series.

Phase of development of a project



1– The pages with the objects are created to visualize, by means of operations on hand copy-paste from the menu of the objects standard. They can be added also written, created panels and photos and personal designs with other applications. It is possible to recover entire pages or parts of they from developed applications previously, with copy-paste.



2– One from each other develops to the program of the PLC in Ladder language, connecting inputs and outputs of the plc with the objects of the graphical pages. The expert programmator can to make the same thing using language C, and eventually leaving a programmed part in Ladder language, modifiable from the installator, for the putting to final point on the machine.

3– it can test the operation of the entire program (graphical pages, PLC, inputs and outputs, motors etc) on the PC from which the application is being developed, interactively, without to exit from the development atmosphere. This allows to quickly test the entire program and without need of final hardware.

4– At the end, it is compiled for the target final and the program is sent to the PLC.

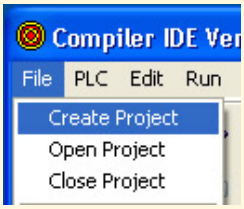
Practical example of development of a project

We see now like, very fasly, with some click of the mouse, is possible to install Proteus, to create a zero simple application beginning from, to test it on the PC and to send it in execution to the hardware.

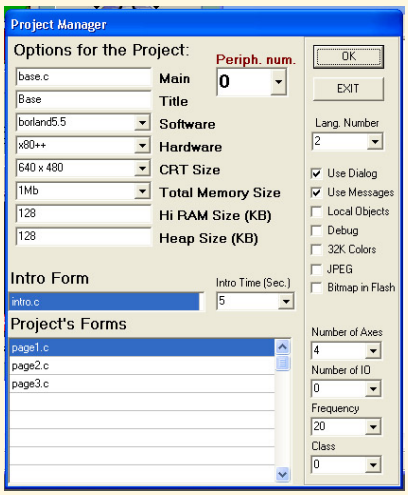
For to install program Proteus, run the program of installation **Install_proteus.exe.** So we can to find the **Proteus.exe** into the directory ***c:\Proteus*** and other files in ***c:\Commons\c***.

1– Creation of project

- To run Proteus.Exe, and so into main menu, selecte ***File->Create Project.*** It will open a window **Select a directory for the project.**



- To create a new folder, where it wants, to enter in the directory created and, leaving in the field **File Name** of window the name **page** , to click with mouse on key **Save**.
- It will appear a new window : **DIRECTORY of PROJECT**. To verify that the directory is that which we had created, we leave in the field **File Name** the name **project.c**, we click with mouse on key **Open**.
- To this time the window **Project Manager** will be opened. All the field can be left unchanged, or can change the name of main, il title etc. In order now we don't change nothing, except:
- In the window **Software** select **borland 5.5**, to test the application that we go to construct in the PC.
- In the window **Screen** set the resolution of screen (320 x 240 if we have a screen like **V5**, 640 x 480 if we have a screen **V10** etc…) **Now we push OK**.

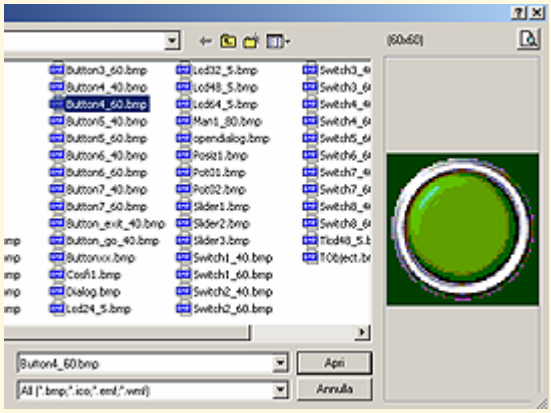


- If we have made all correctly, now there are in the screen three windows: **Inspector of the Objects, directory \ Page1.V10** and to **third window** that for the time being it is empty, except the number of column 1 up on the left.
- Click with mouse, we carry in first plan the second window (**directory\Page1.V10).** Now we begin the construction of our simple program.

2- Constraction of graphic interface

- With the key left of the mouse to make click on the button **object** of the bar standard (you see figure to side), therefore to click on the window centers (Page1.v10) in the place in which we want to insert the first object.

- A menu will be opened, in which we select the **Button4_60.bmp** object. The object will appear on the page in the position that we had chosen. If we want to move it, enough to click with the left key of mouse on the object and, fearing pressed the button, to move it where we want. If we want an other object in place of what we had chosen, a double click of mouse and it open the menu of choice.



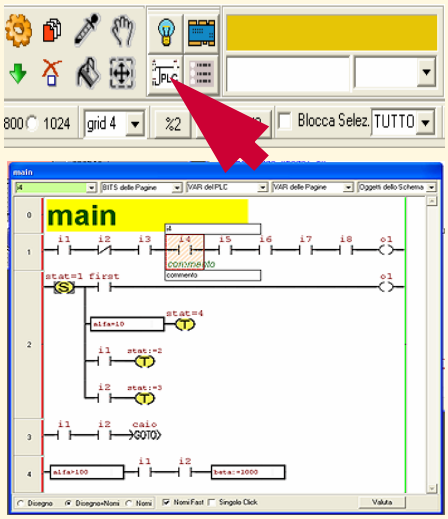
- In the same way, to place on the page a second object, of type **Switch4_60.bmp**. We now have a page with a button, than from the inspector of the objects we see that it is called **bu1** (field var) and a spy lamp **sw1**. (we can change such names, if we want, or from the inspector of the objects, or changing the name in the bar up.

- **The construction of the graphical interface of our simple object is finished.** Now it is necessary alone to virtually connect such objects between them and to the external world, and we make this it with part **PLC**.

3– THE PLC INTERFACE

- To click with the mouse on button PLC of the bar up. The window of programming in language LD will be opened, than for the time being it will be empty. Now we go to execute the connections that we want.

- In our example we decide that pressing the button bu1 active output 1 of the PLC and at the same time switch on the lamp sw1.



(to be continued to following page)

- To select the type up **Contact N.A.** from the bar on the left (the first up on the left), and therefore, in the sheet of the PLC to insert it **in line 1 of column 1** with a double click of the mouse.

- Two fields will be opened in correspondence of the contact. One **up** for the name and **low** for an eventual comment. In order to choose the name we have **3 various ways**:

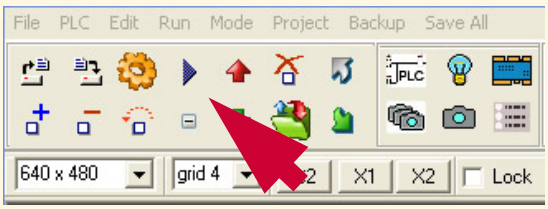
⇒ 1- In the **page of the graphical interface** to click with the mouse on the object to associate (in our case the green button **bu1**)
⇒ 2– Or : In the **Bits menu of the Pages** of the window of the PLC to select the voice **page1-> bu1**.
⇒ 3– Or: **To directly write** in the field the name of the object, in the format **`page_name-> object_name`** (in our case **page1-> bu1**).

- To select the type Coil N.A. from the bar on the left and inserting it in **line 1 column 2 or following**. Be a coil, it automatically will be shift in last column. To this point, to **select the field** arrow of the low bar on the left (the last in low), to positionar on the coil, to make a double click with the mouse and to give them a name, like approval over. In our example the name will be **sw1**.

- Now we must to parallel to the coil of the spy that of output 1 of the PLC, therefore we select **the field Vertical line** of the bar on the left, we locate such line before the coil with a double click, therefore we still select the type **Coil N.A** and insert it in **line 2**, **last column**, giving them the **name o1**, that we will be able to strike directly or to take **from the Bits menu of the PLC.**

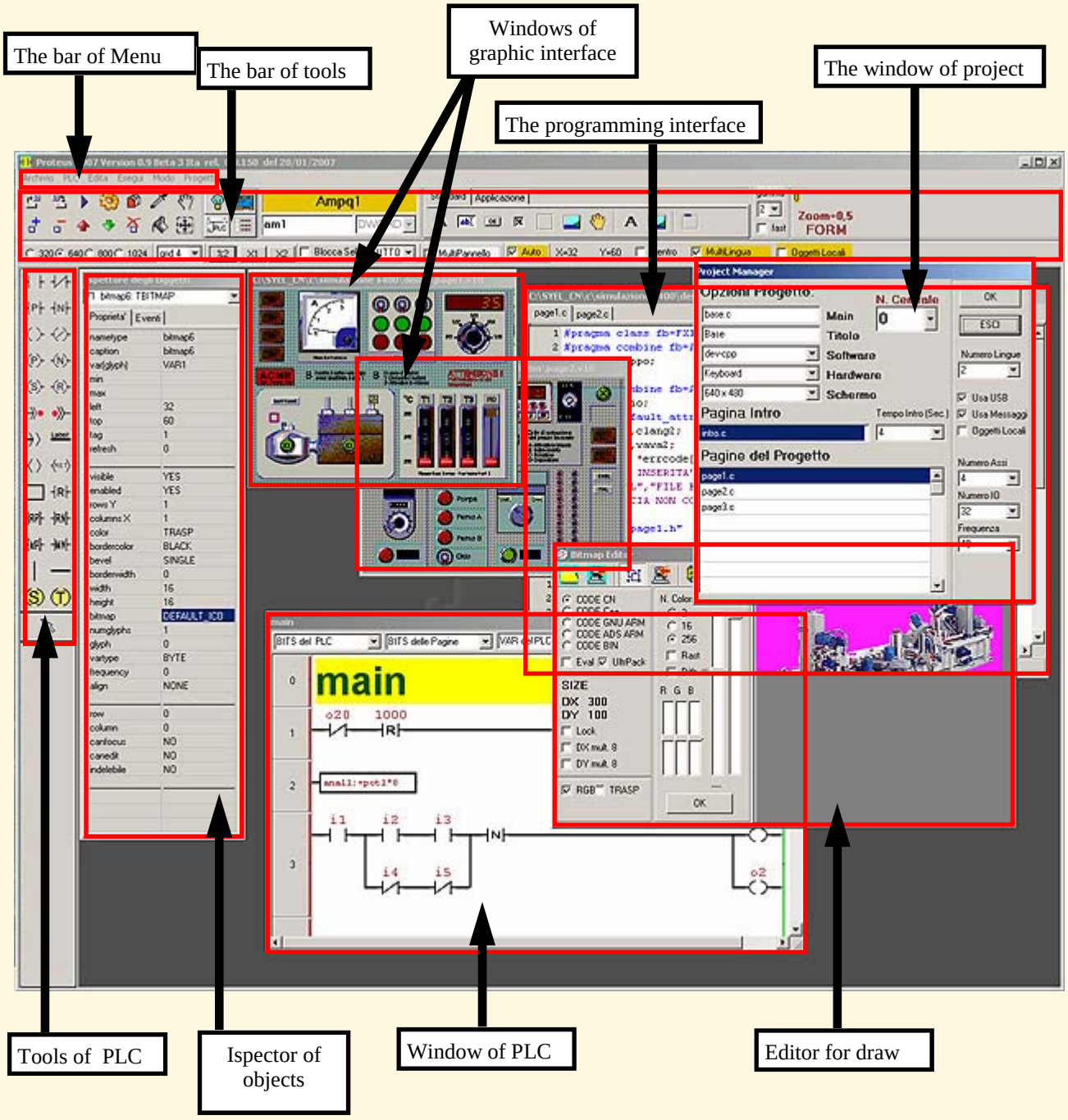
- We will have like result something that resembles to the design here over.
- **It is all. Our project is ended and ready for being tested and sent to the PLC for being executed.**

- In order to execute program on the PC, to press the key Compiles on the bar. The window of application will be opened (base.exe). Such program could be also launch later on once closed Proteus and it is found in the directory of the plan.



The Interface

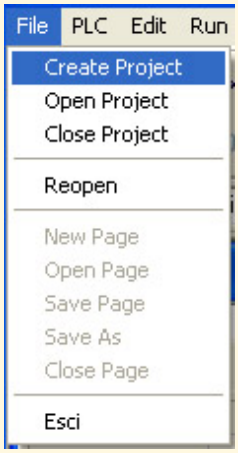
The development platform of Proteus is introduced with a window that encloses a series of elements:



Normally only some of these windows are opened at the same time.
We now see the function of the members of everyone of the indicated elements over.

MENU BAR

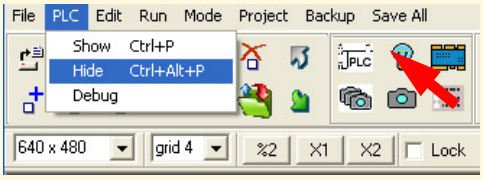
File Menu :



New project, **open project** e **close project** serve respective in order to create a new project, to open of an existing and to close it.
New Project: Selecting this menu, window is opened *Save new file with name*. Clicking on the button Create new folder, for default will come created a new folder of *name folder*, that we can rename with the name that we want. Pressing on the button OPEN we enter in the created folder, and write in the field File Name, the name that we want to give to the first page of the project. If we don't write nothing, it will come assigned the name of default *Page1*. Pressing the button SAVE, it opens the window Directory of the project. To conserve as name of the project Project.c and we press the button OPEN. The window of PROJECT MANAGER will be opened, in which we can assign the target of the project, but will see more ahead this. Now we press button OK. Two empty windows will be opened: one with gray background for the graphical interface, and one with background white for the programming of the events. If the simplified way is selected it opens only the window of the diagram.

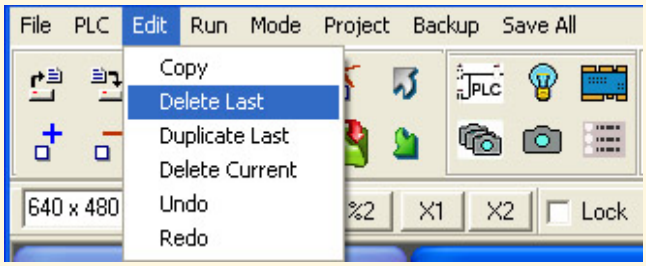
OPEN PROJECT: Selecting this menu a Loaded file window is opened. To choose the directory of an existing project, and to click on the name of a page of the same project.
REOPEN it makes to appear a menu, containing the last opened project, allowing to choose which to open in way express.
NEW PAGE it allows to add a graphical page to the project. Not specifying a name for the page, it comes given for default a progressive name Page1, Page2 etc.
OPEN PAGE it allows to open a graphical page of the project. More pages can be opened at the same time.
If we are in *complete way* with to the graphical page it will open also the relative page of programming of the events.
SAVE PAGE it saves the modifications brought to the page.
SAVE AS it concurs to save the actual page with a various name.
CLOSE PAGE it closes the visualization of the actual page.

PLC Menu :



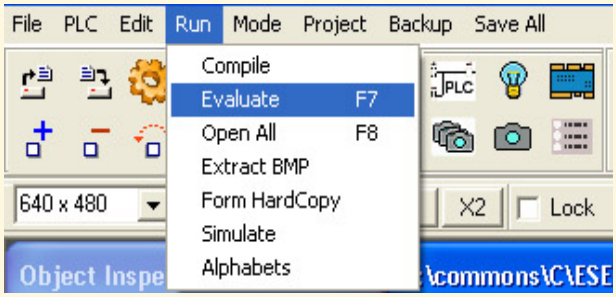
It allows to make to appear and to hide the window of the PLC. The same function is acquitted from the key PLC, or from keyboard pressing CTRL+P and CTRL+ALT+P respective.
The Debug optino open the Ondine Debugger window.

Edit Menu:



Copy: Selecting this menu, or with CTRL+C on the keyboard, we copy in buffer a currently selected graphical object. The arrow of the cursor will be placed side by side from a square, that it means that the buffer of the objects is not empty. Clicking with the key left of the mouse in a zone of the graphical window, we copy you the object in the buffer. Clicking instead on the right key of the mouse we empty the buffer. Copying a panel, we will copy also all the objects in it contained.
Cancel last: It corresponds to button 2 of the bar of the tools, and cancels the last created graphical object.
Duplicate last: It corresponds to button 1 of the bar of the tools, and duplicates the last created graphical object.
Cancel current: It corresponds to button 5, or to the key DEL. It cancels the graphical object running selected.
Annul: It corresponds to button 3, cancels the last action. It can be used recursively.
Restore: It corresponds to button 4, cancels the last one cancels. It can be used recursively.

Run Menu :



Compile: Compile all the project
Evaluate: only used in particolar cases (you see more ahead)
Open all: It opens in the programming window sources of all the pages of the project.
Extract BMP: Extrats all the images of the project from the resources to directory Bitmap in .BMP format.
Form Hardcopy: Copy the graphic image of the currnt form in the directory of the project in .BMP format.
Simulate: Open the simulator form (available only for PC compiled applications)
Alphabets: open the special alphabets and traduction window.

Mode Menu :

It allows to choose the wished way of programming:
-**Easy Mode** : Adapt to create applications type electronic panel, with programming in Ladder language. Not shows the window of programming of the events. On the bar of the tools show only the adapt components to being used on an electronic panel, and that they don't have need of programming structured for being used. It is the way most simple and express to develop a classic application PLC + graphical Pannel .
-**Complete Mode** : For the expert programmer, it puts all on hand the tools of programming of the Proteus, it allows to specify methods and events, and qualifies the programming in language C.

Project Menu:

It opens the window of the Project Manager. It is equivalent to the below button with the icona of the light bulb.

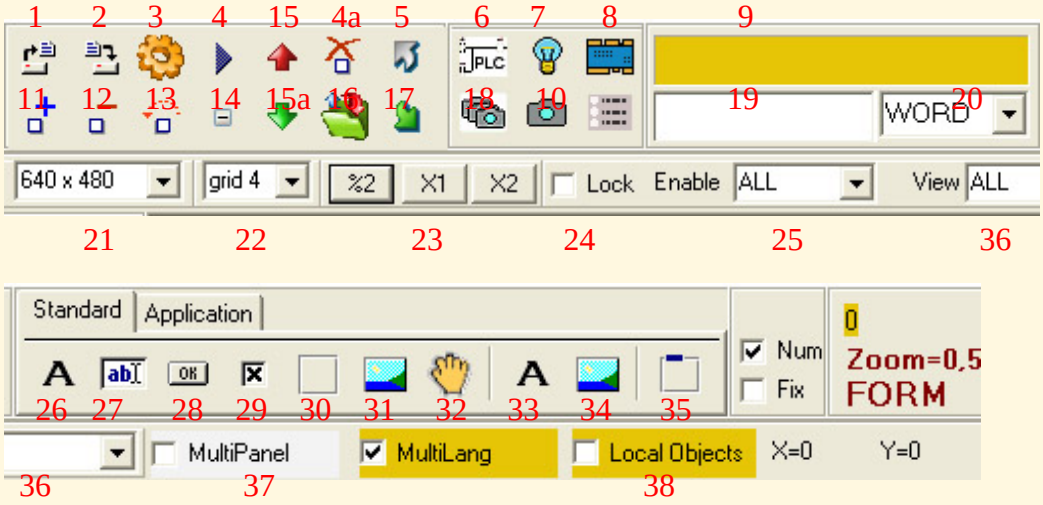
Backup Menu:

Perform a dated backup of the entire project in the Archivi subdirectory.

Save all:

Update all the files of the project (If the project is not saved the modifications are temporary and we are lost when the project was closet)

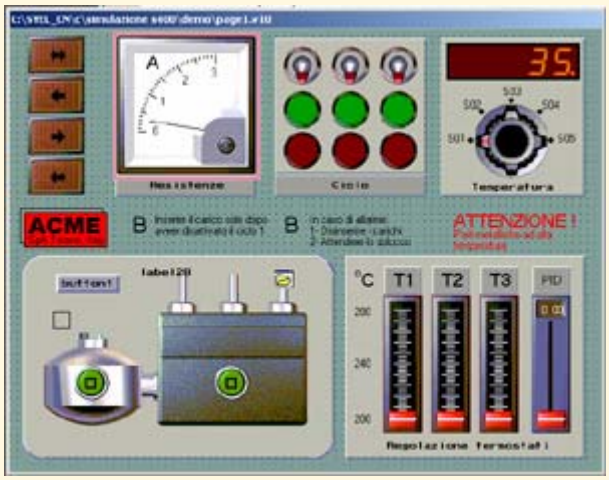
TOOL BAR



The bar of the tools has numerous buttons and fields, that they have the following functions:

1– it opens a graphical project or page (and the relative page of program if we are in complete way)
2– it saves all the project
3– it compiles the actual page.
4– it compiles the project, and it generates the application. If PC project, opens the emulation window.l
4a- it cancels the currently selected object
5– it get the selected object back to the background
6– it opens the window of programming in Ladder language
7– it open the window of project manager
8– it opens the serial interface window to send the program to the PLC.
9– name of the currently selected object.
10– It creates an image JPEG of the form in the directory of the project
11– It duplicates last created object
12– it cancels last created object
13- put the selected object first in the list of our group
14– put the selected object next in the list of our group
15/15a- undo (red) and redo (green) of the last operation.
16– The current selected object is the container (green flashing box)
17– it puts the object selected within the group.
18– it creates images bmp and JPEG of all the graphical objects of the form
19– name of the associated variable to the currently selected object
20– associated type of the variable to the currently selected object
21– dimension of the graphical window in pixels
22– step of the grill for the disposition of the
23– zoom of the graphical window
24– block of the objects (when this field is selected cannot move the objects with the mouse)
25– filter to choose which type of object is selectable with the mouse
26-35 buttons to choose the type of the member to place in the graphical window:
26– label
27– field of edit
28– button
29– checkbox
30– panel
31– bitmap (draw or photo of type BMP or JPEG)
32– object (to select in the library of the objects)
33– static label (deprecated)
34– static bitmap (deprecated)
35– multipanel.
36– if the selected panel contains other invisible panels it allows to choose which to visualize
37- it allows to choose, in a multipanel, if to select the entire file or the single card
38– if selected, the objects are associated to the page contains that them, and can use the same name for various objects in various pages, for against we will be obliged to indicate in the name of the object also the relative page (to es. page1-> label1). If selected the objects are not total, but we will have to make attention not to call with the same name objects in various pages.

Windows of graphic interface



In these windows, it is possible to place, to define and to move graphical members and objects that will make part of the application. Between the members we have:

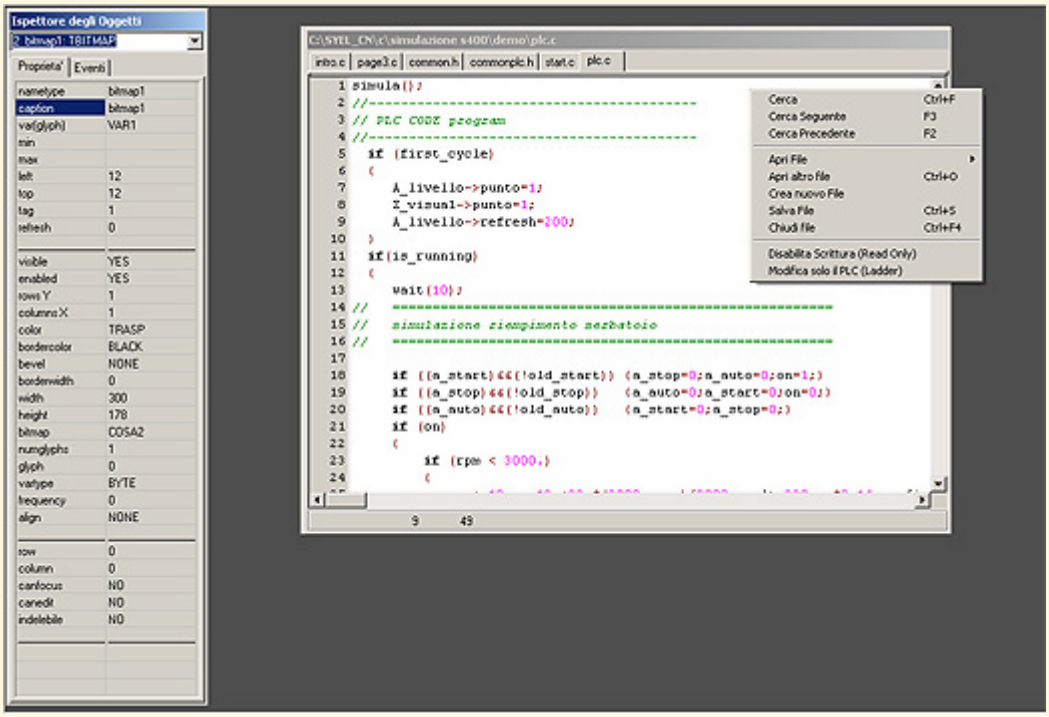
- Label
- Bitmap
- Field of edit
- Button
- Object

Generally an Object is a particular type of component that consists in a bitmap, animated, associated to a program of library, ready, than explain all the relative functions to the same object.

For example:
To an the switch object is associated bitmap that it show the switch in the condition in which is found (on or off) and a program that is taken care to manage the touch of the finger on the touch-screen to activate the switch, and that it associates the relative associated state to the variable one to it.
To an object ammeter the image of an ammeter is associated, with the number of divisions and the scale to adapt the chosen over the entire scale, and a program that draw the pointer designs in motion, reading the value from the analogic input or the variable one that we will have indicated.

In order to add to a component or an object to the page (than when it comes created it is initially empty), enough to select with mouse the type of component wished on the bar of the tools, and therefore, with an other click of the mouse on the window of the page, to place it where we want. At any moment we can move an object selecting it with the mouse and dragging it, holding pressed the button left of the mouse same. For the components that can be reorganized, like the panels, we can reorganize holding them pressed the right button of the mouse, or with the button left, after to have low carried the cursor in proximity of the angle to right of the same component.
Placing an object within a panel, it automatically will be associated to the same panel, for which, moving the panel, it will be moved with it.
The panels can be recursively placed within the other.
The component can be associate to a variable one in automatic way and biunivocal.
For example, if the field of edit Edit1 is associated to the variable one x1, if the value of x1 changes, the field of edit is brought up to date automatically, while if we enter in input of the field (touching with a finger on the same field) and are worth the visualized value, the variable one x1 will assume the manually set up value.
With a double quantity click of the mouse on the component, a window is opened that concurs to change in intuitive way the propertis of the same component. In alternative, in the inspector of the objects the list of all the properties of the selected object is brought back, that we will be able to change voluntad.
If we work in complete way of programming, beside the graphical window, it will be present a window of programming, and the inspector of the objects will have a card “Events”. In such case we will be able to associate to every relative action to the object, a code of management of the same event. This demands the acquaintance of language C and the programming to events (type Visual Basic, Delphi, Borland C++ Builder).

The inspector of the objects and the interface of programming



They are the two windows that, **in complete way of programming**, allow to write the code of the program and to assign *properties and events* to the single objects of every page.
In **simplified way of programming**, that to create simple applications of type PLC with graphical interface, it is necessity to use these windows not there.
The double spirit of Proteus is understood from this…, that it can be used by the installator of PLC without acquaintance of programming in structured languages, but can also be used from the expert programmer, that it can take advantage of without compromises all the potentiality offered from the hardware and the software.

Now we examine the fields of the inspector of the relative objects and the modalities of programming.

Property:

Nametype: it is the name of the component. If in a page there are more components with the usual name, the successive ones are of clone of the first one and of it they inherit property and events, *saves the individual properties, that they are Caption, Var, Left, Top, Tag and Refresh.* A component is characterized by:
`page_name->component_name`.
Inside of a page, the same page can be characterized from the Local pointer.
Inside of an event of an object, the same object can be characterized from the pointer **me**(or **This**).
As an example, if we are in page page1:

```
void Event(edit1_onentry)()
{
   page2->edit3->caption="abc"; //change the caption of edit3 in page2
   Local->edit2->caption="123"; //change the caption of edit2 in page1
   me->top=100;                 //change the position y of edit1 in page1
}
```

Using the Local and me pointer we can create an independent code from the page and the object, and therefore usable without modifications in various pages and for various objects.

Caption: it is a string associated to the component. For some component as Panel and Bitmap, this field is not visualized, but exist however, this being an individual property. In the field caption we can write a long text to the maximum 32 characters, that it will be the **text** of the relative string. In alternative we can write a text preceded from an asterisk, in such case this will be the **name** of the relative string, or a text followed from an index between parenthesis quadre, or a text followed from parenthesis quadre open-close, in the event of vector of components.

For example:

caption	it is equivalent:
string 123	me->caption="stringa 123"

*stringa2 me->caption=(char *)stringa2
stringa3[x] me->caption=(char *)stringa3[x]
stringa4[] me->caption=(char *)stringa4[_index] (only for vector of component)

Var: It is the name of the variable associated to the component. For default it comes assigned the variable VAR1, that it is predefined. When a variable to an object is assigned, is necessary that this variable is defined in the header of the project (common.h) and is necessary to set the propriety vartype of the component in compliance with the type of the variable.
The name of the variable can be the element of a vector, but not an expression (must be something that can be to the left of the equal in an allocation).
For vectors and matrices of components can be used variable continuations from parenthesis quadre open-close for the automatic indices.
For example, are valid names:
alfa alfa[2] beta[num1*3+5] varx[] (only for vector of component) vary[][] (only for matrices of components)
But they aren't valid names:
Alfa+1 alfa+beta

Left Top Height Width: They indicate the position of the component inside of the page, or of the panel if they are contained in a panel, and the dimensions of the component. Normally they are modified moving and reorganizing the object with the mouse, but for little alignments outside grill or for other reasons they can also be modified manually setting up a new value.

Tag: It is a generic numerical field, that it can be used by the programmer. In the event of clones objects it can be used for associate an index that distinguishes an object from the other. In the event of buttons and checkbox, if it comes put to 1 refresh the property, the tag identifies the button (or the checkbox) all buttons that have same the variable one (that is of the same group). (pressing the button the variable assumes the value of the tag, changing the value of the variable, the button whose tag it corresponds to such value, it will turn out pressed).
In the case of a invisibile panel the Tag is the index of visualization (1,2.). In such way (point 36 to pag 15) it will be possible by the menu to curtain () to choose which to visualize in the window of Edit. This is particularly useful in case of windows partially or completely overlapped).

Refresh: Normally to zero, such variable comes put to 1 for buttons and checkbox mutually exclusive (sees Tag), and for fields of edit if we want that the field comes redesigned periodically even if the variable of the relative value doesn't change.

Frequency: It indicates the minimal frequency (in milliseconds) of refresh of the field. For example, if for a field of edit we put the property frequency =50 the field comes refreshed every 50 milliseconds, even if the value of the variable changes more fastly. If put to zero, the field will come refreshed every time that changes the value of its variable.

Align: Valid property only for componet type panel, indicates the alignment of the panel regarding its container (the page or the panel in which it is contained). For example we can put property align=TOP for the bar of the title, and in such a way will place it always up, with the equal width to that of the panel contains that it.

Canfocus, Canedit: A field of edit is editable when both these properties are to 1. A panel with property canfocus to 1 it can contain fields of edit selectable with the tabulatore from keyboard.

Visible: An object is visible if it has this property to 1. An object that contains other objects (for example a panel) if it invisibile (visible=0) renders invisible also all the objects in it contained.

Enabled: An object with this property to 1 is sensitive to the touch of the finger on the touch-screen.

Rows, Columns: For panels, label and fields of edit, servants to create vectors or matrices of components. With columns > 1 it creates a horizontal vector, with rows > 1 vertical vector, if both fields are > 1 creates a matrix of components. The relative fields var and caption can be indexed with the index automatic rifle (parenthesis quadre open-close)

Color: It is the color of the background of the component. With a double click of the mouse on the field *color* in the inspector of the objects, a window of dialogue for the choice of the color is opened.
Fontcolor: It is the color of the text for components Edit, Label, Button. Like in the case of the property color, with a double click of the mouse on the field color in the inspector of the objects, a window of dialogue for the choice of the color is opened.

Bordercolor: It is the color of the edge for panels, label and bitmap, while for fields of Edit and Checkbox it is the color of the written (caption) right of the same field.

Bevel: It is the style of the edge, and it can be selected with the menu associated in the inspector of the objects.

Borderwidth: It is the thickness of the edge in pixel. It must be greater or equal to zero.

Bitmap: For components of type Bitmap and StaticBitmap, it is the name of the bitmap to design. With a double click on the field bitmap in the inspector of the objects or on the bitmap same in the graphical window, a menu is opened for research of file that contains the design (the file must be of type .BMP). Chosen the file, the window of Editor of the designs is opened, with which can to modified and to cut the image part that servants (you see chapter Editor of the designs). In alternative, if the design appears already in the project and already has been imported, we can write in the field, directly the name of the bitmap same. The image can contain or more designs, all of the same dimension, placed side by side in horizontal sense. In second case, with the **Numglyphs** property we will indicate the number of designs of which the bitmap it is composed, while the property **glyph** it indicates which of these designs it must be visualized.
When the program goes in execution, the number of the design that comes visualized is that one indicated from the variable associated the same member bitmap var (glyph).

Vartype: It is the type of the variable associated to the component. It must correspond to the effective type of the variable: for example if the property var (var (glyph) in the case of bitmap) is alfa1 of type short int, the relative Vartype field must be WORD.

The correspondence is:

char, unsigned char	BYTE
short int, unsigned short int	WORD
Long int, unsigned long int	DWORD
float, double	REAL
char *, char []	STRING

Other property: **min, max, row, column, indelebile, mobile**: they are used in particolar cases, than for the time being we omit to consider.

Methods (or events):

The second card of the inspector of the objects is dedicated to the management of the events.
Every object of the page, comprised the same page, that it is the object number zero, has associated a sure number of events, than, if associated to a function, they call it in case they are taken place.
In order to create a function for an event, enough to write the name in the relative field, or, with a double click on the field, to leave that Proteus kinds a name of default (raccomended choice).
The function will be created automatically in the programming window. With a double click on the name in the inspector of the objects, the cursor of the window of programming will be carried on the same function.
More events, also than various components, can be associate to the same function.

Inside of the code of a function, we can use sure variable predefined:

`(char)_byte`	it is the variable associated if of type BYTE
`(short int)_word`	it is the variable associated if of type WORD
`(long int)_dword`	it is the variable associated if of type DWORD
`(_double)_real`	it is the variable associated if of type REAL
`(char *)_string`	it is the variable associated if of type STRING

These variable can be used in reading and writing: for example:

```
if(_byte > 2) _byte++;
```

`(short int)x_x`	it is the coordinate x absolute of the object (first point left)
`(short int)y_y`	it is the coordinate y absolute of the object (first point up)
`(OBJ *)me`	it is the pointer to object
`(TYPE *)Local`	it is the pointer to current page

`(unsigned char) page_end;`	If it is put to 1, exit from current page
`(unsigned char) page_init;`	It is worth 1 during the initialization of the page
`(char) __autorefresh;`	It is worth 1 if the object is in autorefresh
`(char) __clone;`	It is worth 1 if the object is a clone

`(short int) mouse_x;`	abscissa of the mouse
`(short int) mouse_y;`	Ordinate of the mouse
`(char) _numglyphs;`	Number of designs from which the bitmap is composed
`(char) _keyup;`	The touch screen has been as soon as released
`(char) _keydown;`	The touch screen has been as soon as pressed
`(char) _keypress;`	The touch-screen is pressed
`(char) _abilita_azione_draw;`	the design of the object has been modified (valid in on_exit)
`(char) _abilita_azione_repaint;`	the object has been completely redesigned (valid in on_exit)
`(char) _abilita_azione_click;`	it has been executed the action on_click (valid in on_exit)
`(char) _riga;`	actual line of field of edit
`(char) _colonna;`	actual column of field of edit
`(char) _changed;`	The object is changed

Events:

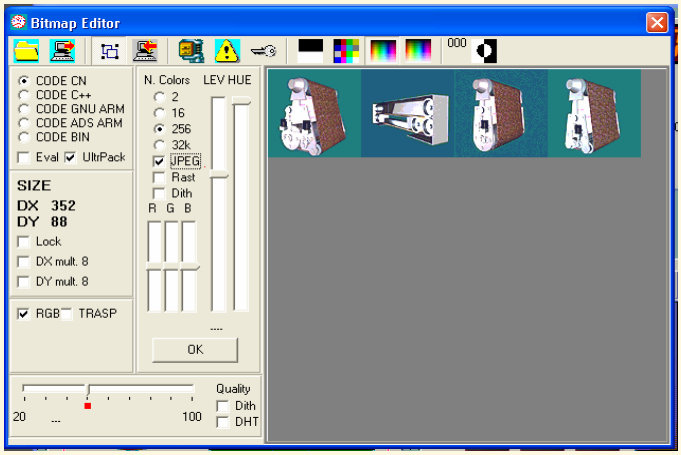
On_first	This event comes called, if defined, when a new page is opened
On_create	Called when the page is redesigned
On_entry	Called periodically, always, even if the object is not qualified and not visible
On_draw	Called when the object must be redesigned
On_click	Called when to click on the object
On_mousedown	Called when mouse (or the touch-screen) it comes pressed
On_mousepress	Called periodically until mouse (or the touch-screen) it remains pressed
On_mouseup	Called when mouse (or the touch-screen) it comes released
On_exit	Called periodically when it is exited from the object

Even if nobody of these functions is defined, the object works regularly (auto-upgrade when it changes the variable, a field of edit visualizes and concurs of modify the variable etc.). The events over listed, in case defined from the programmer, concur to make things in more regarding the normal management of the page. In case of programming simplified, in Ladder language, it is not necessary to define some event.

The programmer can inside of an event or elsewhere, to modify the properties of any object of any page.

`Pagina->oggetto->Left`	
`Pagina->oggetto->Top`	
`Pagina->oggetto->Width`	
`Pagina->oggetto->Height`	
`Pagina->oggetto->Caption`	
`Pagina->oggetto->Tag`	
`Pagina->oggetto->Color`	
`Pagina->oggetto->BorderColor`	
`Pagina->oggetto->FontColor`	
`Pagina->oggetto->BorderStyle`	
`Pagina->oggetto->Glyph`	
`Pagina->oggetto->DecimalPointPosition`	
`Pagina->oggetto->Digits`	
`*Pagina->oggetto->Visible`	(use like pointer)
`*Pagina->oggetto->Enabled`	(use like pointer)

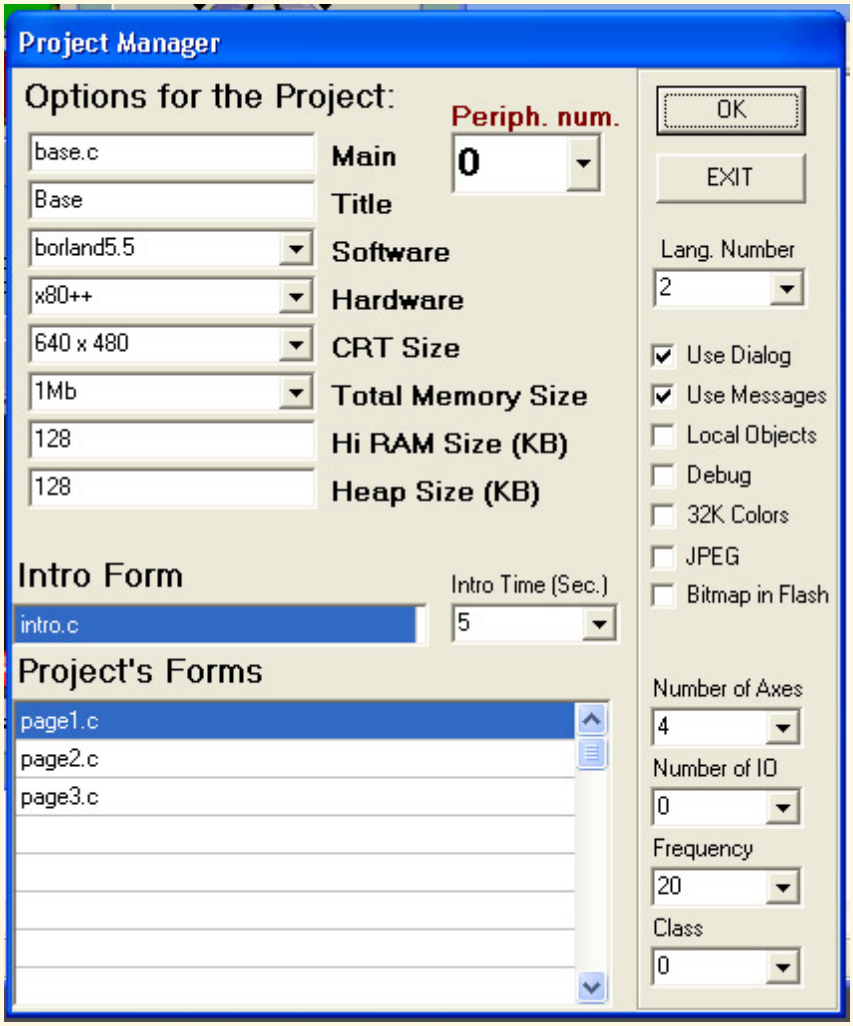
Bitmap Editor



In order to choose the image to visualize in a field of Bitmap type, the editor of the designs is used , that it comes recalled with a double click of the mouse on a object type bitmap on the graphical window, or with a double click on the relative one property in the inspector of the objects.
The file of type bitmap (BMP) come visualized in the right window, and can be cut, moving the angle up to right holding pressed the key left of the mouse, and the angle low on the left pressing the right key of the mouse.
Only the image part that is looked in the window will come visualized in the application.
We can use cursors LEV, HUE, R, G and B in order to change the ink of the design.
Selecting field TRASP we can render a color transparent, that we will choose clicking on the image with the mouse. The relative horizontal cursor allows to regulate the similitudine of color for the transparency. If the image composed from bitmap more placed side by side (in the example of the figure they are two bitmap) is necessary to make attention to cut a portion of image that has multiple width DX of the number of images (for example if they are 2 designs DX must be multiple of 2, if they are 3 designs, must be multiple of 3 etc.)
Pressing key OK the design comes converted in code and the window is closed.
The maximum depth of the color for the editated image is selected with the option N. Colors, for which, qualifying N. Colors=32K and Jpeg the possible modalities of color come sampled all.
In the window of the project we will choose the maximum depth of the color with which compiling the application.
For every image the effective depth of color will be the minimum between the depth of color of the project and that of the single image.
The field CODE CN, CODE C++ etc. aren't influentie.
For the maximum efficiency of the code, it is racommended to use N. Colors=256, not to use the Dith field, and however to always select modality the UltraPack.
The option JPEG concurs to generate more compact programs, but it is a greater wait to the start of application, due to the time of decodes of images.

The bitmap editor it is the single way to import of the designs in the application, as code C is not in a position to reading directly of the BMP or the JPEG, which must be transformed binary coded compressed .

Window of project



The window of the project, than can be opened anytime to modify the options, concurs to specify the target.

Main is the name of the final application, as an example if main=base.c, the final program calls base.com for the PLC or base.exe for the PC.

Title is the title it of the application (used only in the event of program for the PC).

Software can be gnu_arm (for Vk5 and V8), arm (for system applications on VK5 and V8), S400 (the module of expansion PLC), or dev-cpp (program emulation on PC) or still borland 5,5 (for PC, if we want to use this releasable compiler, liberations from the http://trial.borland.com/survey.aspx?sid=33). site, or DirectX anchors (the Program turns on PC using Directdraw for the visualization) .

Hardware specificates the human machine interface, which can be the touch-screen, keyboard, both, or no interface in the case of PLC without display (blind).

Screen indicates the resolution of the screen, which can be 320 X 200.640 X 480.800 X 600.1024 X 768, or LCD (4 lines X 20 characters), or not in the case of PLC blind.

Intro Form is a possible Introduction page to list the power-up of the machine for the time specified in the field **intro time**.

Project's Forms is sorted list of pages that are part of the project. This list is created and updated automatically every time you add a page to the project, but can also be amended manually. Clicking on a field, opens the menu of the pages of the project. The pages can be inserted and removed by hand, writing directly in the fields of the list.

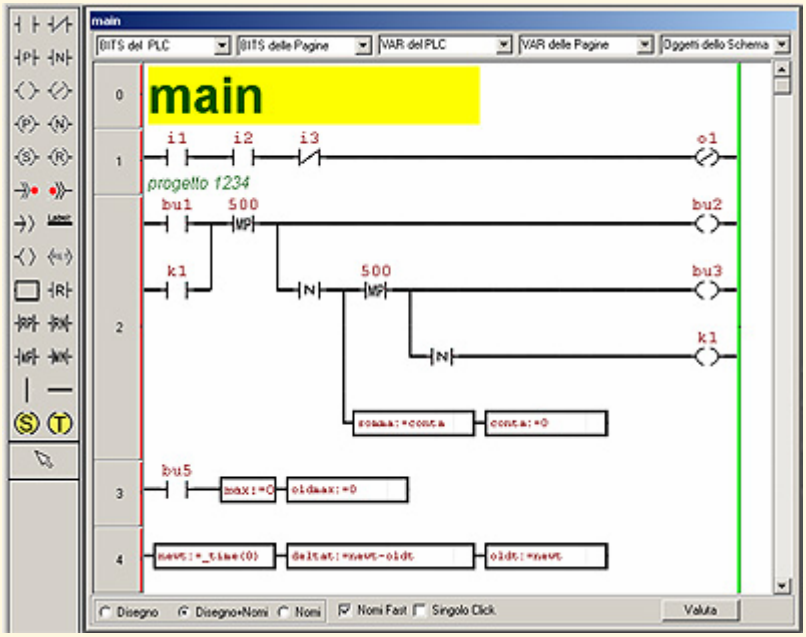
Languages number indicates the number of languages in which we see the pages of the project. For each string (Caption etc.) will be automatically generated a number of strings in the file of vocabulary of translation, corresponding to the number of languages chosen. For this is well decide from the beginning, to avoid manually the strings in the dictionary, how many languages we want to use.

For example, if for the field label1 we use the Caption= "string of proof", we will find in the dictionary, or in the file lang2.h in the directory of the project, the following line:

```
const char *L_string_of_proof[]={"string of proof","string of proof","stringa of proof"};
```

The string is repeated many times as specified in the field number languages. By default is repeated n times the original string. It is our manually edit the file **lang2.h** and translate the various occurrances of the string in the languages wanted (is not necessary that the strings have the same number of characters) if the variable Lang worth 1 will be displayed the first string, if it is worth 2 the second, etc. If we change the value of the variable Lang all the strings displayed change instantly.

Window and tools of PLC



Opening the window of the PLC also appears the toolbar. With this bar we can select the component to insert in the scheme. When the cursor is inside the window of the plc, a square grey indicates the position of edit current. If it is not selected the single click on the field we can place the component. With a double-click the mouse, otherwise you need only a single click (deprecated). In square around the cursor appears, in a corner, the symbol of the component chosen.

If the latter is barred in red means that cannot be placed at the current position. To cancel a component, position the cursor on it, and then press the button. If the component is a vertical line, to cancel must select the component vertical line, otherwise we will not succeed in placing the cursor on the line itself, given that lies between two adjacent cells.

If we are in modalities **design**, we can place quickly components, without assigning the name.

If we are **picture+name**, once placed a component, appears the edit window to give a name (top) and possibly a comment (below). If you do not want to assign suffered the name, we can leave the edit with a mouse click. We can assign it when we want, if you select as component the arrow, and clicking on the box of the scheme in which want to assign or change the name. So names the Toolbar disappears, and we can assign or change the Change the names and comments to the components. The name may be beaten directly from the keyboard, or searched in drop-down on the high in the window, or can be allocated clicking in the window graphics on the component on. To insert a new blank line between two existing need a double-click on the left on gray bar with the numbers of cells. To cancel a line, select it all from left to right (the red bar to the green) and then press the key DEL.

On the page of PLC we can select More cells, while holding down the left mouse button, and we can carry out copy and paste also of whole areas, with the keys control+c and control+v. With the button Evaluate we can assess the correctness of the key, if it detects an error (a cell without a name, or a connection missing) is reported the error and the cursor, outlined in red, it leads to the cell where there is the error.

In the case in which the error to be in a sub, the displayed page will be one of the Sub in which it was found the error. The evaluation is firm to the first error found. To continue, we must correct the error, close the evaluation by pressing the button currency, and then press the button currency for a new scanning files of the project. The evaluation part always from the beginning of the program main, independently from the page displayed.

For a correct editing, it is better Close the window of evaluation of the errors, pressing still on the button evaluate. to note that a single component (a wire horizontal or a contact) in the first column, not connected to another, not generates error. This is to allow certain special functions of debug, that we will see detail in the Manual of debug.

To scroll vertically the layout, use the scroll bar on the right. The bar on the left shows the number of the cell. A cell can be made of 1 to 8 lines. A line can contain up to 8 contacts more the coil. For lines more long use the symbol of the continuation, that allows you to continue the line on another cell.

The symbols used are those standard of programming to contacts (contacts, reels, single, derivatives, delays, logic blocks, vertical and horizontal lines, Goto, sub, labels, return) more the symbols of state and transition that allow you to implement machines to state. For a detailed explanation of the use of components of the PLC is refers to the Manual of programming.

Some of the keys have special functions:

DEL or Ctrl+Z replace it with another (you can place objects only in boxes empty or containing a wire horizontal)

Ctrl+X as DEL but the component current becomes that deleted (equivalent to Type A DEL, and after going to choose in the bar of the components of the PLC that just deleted)

Ctrl+C the component current becomes the one found under the cursor.

Ctrl+V inserted in box of the cursor the component chosen (equivalent to a double click).

Space rotates to right the line from current cursor.

Insert rotates in the lower the lines of the cell current from cursor.

Backspace cancel last operation (Undo).

F3 search a box that has the same name as that under the cursor. If the cursor is on a Sub opens the scheme that contains the Sub. If the cursor is on a return, or if it is on a white box and we are in a sub, back to previous scheme. The easiest way to create a new scheme (i.e. a new page of plc, or a Sub), is to add the existing scheme a function sub, give it a name and then beat the key f3. The new scheme is and automatically. The sub can be nested recursivamente one inside the other, up to a level of 100. We remind you that one function sub must be the last of its line, EVEN IF IT IS NOT IN COLUMN 8. To switch from a scheme to another we can also use the key open.

The keys Ctrl+X, ctrl+z, Ctrl+C and DEL may be used to cancel AND TO MOVE whole areas of the key, select by holding the left mouse button.

The library of the objects comprises:

Keys, switches, commutators, selectors.
Rotating Potentiometers and sliders
Ammeters, Voltmeters, Electric power meters AC and DC, Cosfimetri.
Visual display units of quota
Positioners for motors AC and DC
Controller PID of temperature.

Other objects upon request :

Module of Interpolation linear-circular.
Interpreter of interpolation ISO language .
Module of sanding.
Module of management Trifase (Volts RMS, Ampere RMS, Watt, Volt-ampere, Var, Cosfi for asymmetric terns unbalanced with and without earth).
Module DSP (filters Fir, Iir, Adattivi, direct and inverse FFT).

Technical Data of realized programs with Proteus

Program on S400 with monitor V5/V10:

Max Number of page of a project:	50-200
Max Number of object for page:	1000
Time of loop PLC	1 mSec/1200 lines
N. max. timers PLC	65000
N. max. task	3
N. max. routines to time	32
Resolution of display	320X240 - 1024X768
Dimensions max. of bitmap	Screen size
Numbers colors of bitmaps	256,32K,Jpeg

Program on VK5/V8 Arm:

Max Number of page of a project:	1000
Max Number of object for page:	1000
Time of loop PLC	1 mSec/5000 lines
N. max. timers PLC	65000
N. max. task	32
N. max. routines to time	100
Resolution of display	320X240-800X600
Dimensions max. of bitmap	Screen size
Numbers colors of bitmaps	256,32K,Jpeg

Program on PC with S400 without display:
(PC Pentium4 2GHz, 512MB Ram, OS Windows XP)

Max Number of page of a project:	65000
Max Number of object for page:	1000
Time of loop PLC 1	1 mSec/150000 lines
N. max. timers PLC	65000
N. max. task	16
N. max. routines to time	32
Resolution of display	320X200-1024X768
Dimensions max. of bitmap	Screen size
Numbers colors of bitmaps	256,Jpeg

For the programs that it run on PC, the task of PLC run out of the task switching of Windows, guaranteeing an operation Hard RealTime, with the maximum time of guaranteed latency inferior to 1 mSec