UNI-PRO 3

DEVELOPMENT ENVIRONMENT FOR PROGRAMMABLE CONTROLLERS

SOFTWARE MANUAL

CODE 114UP3E444

Important notice

This Instruction Manual should be read carefully before use, and all warnings should be observed; the Manual should then be kept for future reference.

Summary

GENERAL INFORMATION

1.1 About UNI-PRO 3

UNI-PRO 3 (hereinafter also called *UNI-PRO*) for Windows® is an integrated development environment for C-Pro programmable controllers that enables you to customize and manage complex system control projects.

It enables you to set up —in an easy and user-friendly way— a modular range of hardware that you can use to make the controller, as well as to define all functional and control features of the system.

The development environment is composed of a series of integrated and interacting tools that enable you to perform the following operations:

- ➢ creating and managing objects and control algorithms
- ➢ creating and managing user interfaces
- ➢ setting up controller networks
- ➢ choosing the hardware, compiling, verifying and downloading the code
- ➢ debugging controller operation
- ➢ assisted management of controller's user parameters (not yet available)
- ➢ creating and changing object libraries and managing the corresponding versions
- ➢ generating project documents automatically (not yet available)

1.2   System requirements

1.2.1   Hardware Requirements

Below are the minimum system requirements to run UNI-PRO:

- ➢ Microprocessor @ 2.0 GHz (or higher)
- ➢ 2 GB RAM (or higher)
- ➢ n° 2 USB ports
- ➢ 500 MB hard disk space
- ➢ SVGA video card, 1024x768 with 512 MB memory (or higher)

1.2.2   Software Requirements

One of the following operating systems needs to be installed in the PC to run UNI-PRO:

- ➢ Microsoft® Windows XP
- ➢ Microsoft® Windows VISTA
- ➢ Microsoft® Windows 7
- ➢ Microsoft® Windows 8/8.1
- ➢ Microsoft® Windows 10

Important note: Administrative privileges are required to install, run and uninstall UNI-Pro software.

Is requested a PDF viewer, for example Acrobat Reader v6.0 or higher, to the correct view of the help pages and the Hardware/Software documentation.

IMPORTANT. The default screen setup used by UNI-PRO is 96 DPI (Dot Per Inch). The change of this setup applies a resizing to the position and dimension of user interface elements. Therefore some windows may not correctly displayed. So is recommended to use UNI-PRO with 96 DPI setup.

1.3 Developing projects with UNI-PRO

Through the UNI-PRO development environment you can create and manage distributed control projects, no matter how large they may be.

UNI-PRO is not only a tool to develop controller code, but also a useful design tool to build and organize your own know-how. In fact, you can use it during the analysis stage to quickly decide which is the best solution to control a certain system, or to examine a control solution customized to meet customer's requests. The possibility of creating object libraries enables you to build your own collection of objects that you can reuse in several projects, thus protecting the Company's internal knowledge and minimizing development time.

In many cases, the method applied to develop a project is fundamental. With UNI-PRO, you can proceed according to either the *top-down approach* (i.e. defining the system layout first and then implementing the details) or the *bottom-up approach* (i.e. starting from the machine's basic, low-level elements and then building more and more complex blocks, one brick at a time, until you reach the final goal).

The project development cycle is divided into several steps. In the first one, graphic objects representing the main control components are entered in a worksheet. After customizing the properties of these objects, they are easily joined to create the controller's "logical" structure. During this step, the pages that make up the product's user interface will be developed as well.

The second step includes the connection with the "physical" part of the selected controller, i.e. with the inputs, outputs, and internal hardware resources.

In the following step, the graphical part of the program is compiled, including the corresponding algorithms and user interface pages.

In the final step, the code is downloaded and, finally, the program is run to check its operation and make any necessary changes.

The use of UNI-PRO to develop a project should follow an iterative and incremental approach, according to the following diagram:

```
                    ┌─────────────────────────────────┐
                    │      REQUIREMENTS ANALYSIS       │
                    └─────────────────────────────────┘
                                    │
                    ┌─────────────────────────────────┐
                    │      ORGANIZING CONTROL          │
                    │      SUBSHEETS AND USER          │
                    │          INTERFACE               │
                    └─────────────────────────────────┘
                                    │
                    ┌─────────────────────────────────┐
                    │   IMPLEMENTING ALGORITHMS        │
                    │          AND PAGES               │
                    └─────────────────────────────────┘
                                    │
                    ┌─────────────────────────────────┐
                    │   SELECTING HARDWARE AND         │
                    │   CONNECTING TO TERMINALS        │
                    └─────────────────────────────────┘
                                    │
                    ┌─────────────────────────────────┐
                    │      COMPILING THE PROGRAM       │
                    └─────────────────────────────────┘
                                    │
                    ┌─────────────────────────────────┐
                    │      DEBUG AND SIMULATION        │
                    └─────────────────────────────────┘
                                    │
                    ┌─────────────────────────────────┐
                    │    DOWNLOADING THE PROGRAM       │
                    └─────────────────────────────────┘
                                    │
                    ┌─────────────────────────────────┐
                    │        FUNCTIONAL TESTS          │
                    └─────────────────────────────────┘
                                    │
                    ┌─────────────────────────────────┐
                    │         PRODUCT TESTS            │
                    └─────────────────────────────────┘
                                    │
                    ┌─────────────────────────────────┐
                    │        PRODUCT RELEASE           │
                    └─────────────────────────────────┘
```

1.4 Your first UNI-PRO program

Below is an easy program to help you familiarize with the UNI-PRO development environment. The program uses a *Digital In* and a *Digital Out* that will be joined together. The effect we want to achieve is that when the digital input switches, the corresponding digital output switches too. In addition, we will also be creating an EIML page in which the digital output is linked to a combo object with two texts, so that a different message is displayed according to output state.

To create a new project, we can either choose *File / New Project* from the menu or click the *Create a new project* icon in the toolbar. The window called "New UNI-PRO Project" will appear. Enter the name "Sample1" in the Project Name box and click Create.

First of all, we need a digital type input and an output that corresponds to a relay. From the Hardware palette, we select a *Digital In* and click anywhere on the worksheet: the digital input icon with the default name DIGITALIN1 will appear. We can use the mouse to drag it to the desired position.
Following a similar procedure, we can position a *Digital Out* as well.
Once both entities have been added to the main worksheet of the application, we will need to join them. To do so, we place the mouse pointer over the digital input's output until it changes size. We click and notice that an arrow is drawn from the digital input's **output pin** to the pointer. We place the pointer over the digital output's input pin until it changes size and then click.
If we have followed all the steps in this procedure correctly, both entities should now be linked by an arrow.

To save the project, choose *File / Save Project* from the menu, or click the corresponding icon in the toolbar.

We will now discuss how to create the graphical interface: choose *File / New / New EIML Page (240x128)* from the main menu. A blank page will appear, corresponding to a 240x128 pixel LCD graphic display. From the palette of the page, we select the *Combo* icon, then we click on the blank page and drag it while holding the left mouse button down until we obtain a rectangle:

Once we have created it, we can move it to the desired position. If we wish to change its size, we just need to open the properties window (right click on the rectangle and select Properties) and set *Left*, *Top*, *Width* and *Height* as required. Otherwise, we can graphically resize the object by placing the mouse pointer at the ends of the element.

In addition, from the properties window, we will link the *Var* property to the DigitalOut variable, as shown in the figure below:

In this way we have created a join between the DigitalOut variable and the Combo1 graphic element that represents it.

Finally, we will set *Align* to CENTER, *Font* to 8x16 and we will check the *Refresh* property.

Now, we will right-click with the mouse on the rectangle and select the *Combo Wizard*. We will leave the *Type* property set to Text and type "Please Switch-On" in the *Name* window, then we will press Add. In this way, this text will remain linked to the value 0 of the DigitalOut variable (contact open). We will now repeat the operation but this time we will link the text "Hello World!" to the value 1 (contact closed).

Now let's try and launch the compilation by selecting *Project / Compile* from the menu or clicking the corresponding key in the toolbar.

If we had not selected the hardware on which the project will be run yet, the **Hardware Expert** will appear.

After reading the introduction, we will click *Next* to go on to the following screen that shows a list of all available controllers on the left, a brief hardware description in the center, and a preview on the right.

We will select the most suitable controller (e.g. C-pro 3 Mega Built-In) and click *Next*.

The third wizard step will show the serial port configurations (let's leave them out at present).

The fourth wizard step will show the available expansions complete with a brief description on the right, and will give us the possibility of joining them to the selected CAN bus. We won't add any expansions and we will just click *Next* (for a more detailed description, please refer to section **Selecting hardware**).

In the fifth step, we will be able to join a display (we will select the Control-Panel with 240x128 LCD graphic display).

The last step of the Hardware Expert summarizes the selected configuration. By clicking *End* we will close the Wizard and we will automatically go to a window where we can check the joins between the entities and the physical terminals.

In our case, we will need to join the DigitalIn and the DigitalOut in our project to the chosen digital input and digital output, respectively.

We will select DIGITALIN from the list on the left and the physical terminal we wish to join it to from the center list, for example the first one (for a more detailed description of physical terminals, please refer to the hardware's documentation); by clicking Join, the selected items will be removed and a new row will be generated in the list on the right, describing the join that has just been made. On closing this window, the system will automatically request us to join the DigitalOut as well. To join the DIGITALOUT, we will proceed in the same way as we did before. Once we have correctly joined all entities in the Hardware section, the project compilation will start (the result of compilation will be displayed in the output window).

Afterwards it is possible to download the program on the controller: to do that it is necessary to activate the *Project / Download* menu (or click on the corresponding toolbar icon). The following window appears:

If the connection with the board is successfully established, after a few seconds the progress bar begins to expand and once downloading is complete, a box will appear indicating the successful outcome of the operation. Otherwise an error message will appear; in this case we reattempt the operation after having verified some conditions:

> The serial port selected in the environment options (which can be activated from the *Tools/Settings* menu) must be the one where we want to perform the download from.

> The serial cable must be connected both to the computer and to the controller

Having terminated the download operations, your first program should have been loaded in the CproPlus_S0: if you now close the contact where the DigitalIn has been connected, the relay corresponding to the DigitalOut should trigger and the words "Hello World!" should appear on the graphic viewer.

2 BASIC OPERATIONS

In this chapter we will discuss the basic operations that allow you to develop a project using the UNI-PRO development environment. The user will be guided through all the operations required to develop a project (creation, algorithms implementation, compilation and download to the controller).

2.1 Creating a new project

The first thing you need to do to start working with UNI-PRO is create a new project: to do so, choose *File/New Project* from the menu, or click on the *Create New Project* icon in the toolbar. You will be shown the following window containing a list of all project properties.

When you create a new project, it is highly advisable to enter its name, a unique project number, the author's name, the creation date (the current date appears by default) and a brief description of the project. Each project can be identified by a project number, a version and a revision. To create the project, press the *Create* key or the *Cancel* key to cancel.

All project options may also be changed at a later time by choosing the *Project/Properties* menu or clicking the corresponding icon in the toolbar.

2.2 Saving and loading an existing project

Saving a project to disk and reloading it, you can develop it in several steps and store the different projects you have developed.

To save a project, do one of the following:

- ➢ choose File/Save Project from the menu
- ➢ click on the icon in the toolbar
- ➢ press Ctrl-S

The first time you save your project, you will be requested to enter a directory and a destination file name, subsequently the project will always be saved to the same file.

If you wish to save your project with a different name, choose *File/Save Project As...* from the menu; in this way, you will be able to specify a new file name.

To load a project you had saved to disk earlier, do one of the following:

➢ choose File/Open Project from the menu

➢ click on the icon in the toolbar

➢ press Ctrl-O

A window will be displayed where we can select the file to which we had saved our project.

A fast method to find and open a project we had just saved is to use the *Recently Opened Projects* list, which will show a list of projects opened recently sorted by date. To use this function, choose *File/Recently Opened Projects* from the menu and click on the desired project.

2.3   Entities: definition and use

Entities, a basic element in the UNI-PRO design method, are objects that are graphically represented by an icon on the development sheet. They share the following common features:

➢ a variable number of inputs, characterized by a type (see section 3.1)
➢ a variable number of outputs, characterized by a type (see section 3.1)
➢ the properties Left and Top that allow you to define their graphic position inside the worksheet
➢ the properties Width and Height that allow you to define the size of the individual entity
➢ a unique name that identifies the entity in the project
➢ a brief description

In addition, each one of them has specific properties that characterize its behavior and that will be discussed in detail in chapter 3.
Examples of entities are variables, digital inputs (DI), digital outputs (DO), algorithms, libraries, etc.

2.3.1   Name of the entity

Each entered entity needs to have an unique name that characterize it in the project; the name to give to the entity is preferable to follow these rules:

➢ The names of entities, algorithms, categories, inputs, outputs, etc.. must not exceed 80 characters.
➢ The compiler used to compile the project is case-sensitive type, then a name of two different entity CAN'T differ only in capital/lowercase letters. Eg. Two entities with associate names *requestFans* and *RequestFANS* will generate an error while the compilation.

➢ The names of entities must be C-like, then must not contain special characters like: spaces, commas, dots, apostrophes, accents, quotes and mathematical signs (+,-,*,/).

> ➢ The names of entities are case-sensitive

These three rules are advices to use while the develop to not encountering any errors in the end of the project while the compilation phase.

2.3.2   Entering entities

To enter an entity in a worksheet you need to select it with the mouse in the palette where it is, and then click on the desired position in the worksheet: the entity will appear in its default size, which you will be able to change afterwards.

2.3.3   Moving entities

To move an entity to a different place in a worksheet, do one of the following:
> ➢ left-click with the mouse on it and drag it to the required position (drag & drop method)
> ➢ select one or more entities and press the arrow keys (to move them faster, hold down Ctrl while pressing the arrow keys)

2.3.4   Removing entities

To remove an entity from a worksheet, do one of the following: select it and press *Del*, choose *Edit/Delete* from the menu, or the *Delete* item from the pop-up menu. A confirmation window will ask you to confirm if you wish to remove the entity. This operation may be performed with several selected entities as well.

If you remove a subsheet, all the entities contained in it will be removed too.

In addition, when an entity is removed, all the joins to other entities and any references found in the EIML pages will also be deleted.

2.3.5   Cutting, copying and pasting entities

You can perform all the usual editing operations (cut, copy, and paste) on the selected entities either by means of key combinations (*Ctrl-x*, *Ctrl-c*, *Ctrl-v* respectively) or by selecting the *Edit* menu, or the corresponding item (*Cut*, *Copy*, *Paste*) from the pop-up menu. Cutting an entity is similar to deleting it. Therefore, when an entity is cut, all the joins to other entities and any references found in the EIML pages will also be deleted.

2.4   Using control sheets

A control sheet is a special entity capable of containing other entities (included sheets). Thanks to the use of sheets and subsheets, you can implement a tree-structured project applying either a Top-Down or a Bottom-Up approach, or a mixture of both, according to your preferences and the needs of the project under examination.

Therefore, a control sheet allows you to group certain project entities in a logical macro-entity. If you save this sheet as a library, you will be able to reuse it in the future by taking it from the corresponding palette.

Each project is composed of at least one main sheet where you can add either subsheets or other entities.

There are two different approaches to add a subsheet to a project:

➢ you may add an empty subsheet by selecting the ▭ icon in the Software Palette and then insert the different entities in it

➢ or you may automatically move certain entities from the active sheet to a subsheet by selecting the entities you wish to group and then choosing *Library/Generate SubSheet,* from the menu or clicking on the ▭ icon in the toolbar.

In order to join the entities found in a subsheet to external entities, you will need to export the input and output pins of the involved entities (see section 2.5).

2.5   Joining control entities

To set up a project, you need to insert entities in the sheets and join them together according to the logical layout that establishes the controller's behavior.

Joining entities means to link each output to the respective inputs of other entities according to certain simple rules discussed below:

➢ a maximum of one output pin may be joined to each input pin

➢ the data type associated with the input pin has to be compatible with the data type associated with the output pin to which you intend to join it

To join two entities, do one of the following:

➢ if the entities are in the same sheet, you can join them by means of an arrow: place your mouse pointer over the output pin of the entity you want to join until it is highlighted; by clicking your mouse button and moving the mouse, an arrow will be drawn between the selected output and the pointer. If you want to draw a link composed of several broken lines, just click at the point where you want to break the arrow and go on moving the mouse in a different direction, towards the input pin. When you place your mouse pointer over the input pin, it will be highlighted: click to complete the link. Now, if the input had not been linked before and if the input and output types are compatible, an arrow will be drawn to confirm that both entities have been joined. If you want to cancel this operation at any time, just right-click once with your mouse.

➢ if the entities are placed in different sheets, you may decide to **export** the inputs/outputs. Each subsheet has the possibility of exporting certain input or output pins of the entities contained in it. To do so, place your mouse pointer over the pin you wish to export, then right-click and choose *Export IO* from the pop-up menu. The exported pin will change color (it will become blue by default) and will appear outside the subsheet. Let's say you want to join a Digital Input to a Var contained in a subsheet: first thing, you need to export the variable's input pin. Then you can link the Digital Input's output pin to the pin exported from the subsheet.

➢ or you can link inputs and outputs (even if placed in different sheets) without needing to make arrow connections, but just using the **external links**. To specify an external link you need to perform the following steps in order: click the ![button] button in the Software palette, select the output you wish to join, and select the input. If during the link you keep pressed button CTRL, the output terminal the link comes from is stored in order to make easier connecting another input, without loosing the original output and to avoid searching it. Once you have finished joining the entities, you can display the link you have just created by choosing **Show links…** from the pop-up menu of the input or output.

This function is very useful if links are particularly tangled or high in number, but it makes the project less readable.

To unjoin two entities, you can proceed in two different ways according to the type of link:
  ➢ if the entities are joined by an arrow, select the arrow and remove it by clicking Del or choosing *Edit/Delete* from the menu.
  ➢ if the entities were joined using an external link, you will have to display the **Show Links…** window, then select the link and remove it by clicking on the appropriate key.

2.5.1   Name of the input and output terminals of the entities

This function can be activated in the *Settings*

or in the Menu *View->View I/O Label*.

Note: The names are applied to TIMER, ALGORITHM, LIBRARY and SHEET in general.

2.5.2   Names to the external links

Link Out

The output terminals are designed as follows: . Double-click on the terminal will immediately open the Links window.

Drop-down menu
 Property: link properties
 Show Links: window with the connected links

Link In

The input terminals are designed as follows: , the colored box shows the name of the entity to which they are connected; in this case "calc_enableShutter". Double-click on the terminal will return the selection to the connected entity.

Drop-down menu:
 Show Entity: select the entity connected
 Entity Property: open the properties of the entity connected
 Show Links: open the links window
 Delete Link: delete the selected link
        (removal can also be performed with
          SHIFT + Left mouse button on the label input)

2.6   Using array entities

"ARRAY" is a data structure containing N numerical information of the same type. N **arrays** of N dimension are possible.   The propriety **"array"** is available only when the data types are: CJ_VOID, CJ_BIT, CJ_BYTE, CJ_S_BYTE, CJ_SHORT, CJ_WORD, CJ_DWORD, CJ_LONG and CJ_TIME. A*rray is* a modifiable propriety only for the following entities:

· 🔴 PAR - Parameters
· 🔵 FIX - Constants
· 🟢 PERS - Persistent
· 🟡 VAR - Variables
· Inputs and outputs of the algorithms.

When the data type are different from the over reported data, the *array* is not modifiable, but fixed (read-only) with value 1. For any other entities different of PAR – FIX – PERS – VAR the propriety *array* is not available.

The array name showed is modified (when array property is greater than 1) inserting between the square bracket the array size as under reported.

Array property is a numeric type that has 100 as maximum value and 1 as minimum value (mono-dimensional entity) : when array data value is greater than 1, the property value becomes a string to match array index with the correct array value. The syntax are following:

$$0=\textbf{value[0]}, …, \textit{array-1}=\textbf{value}[\textit{array-1}]$$

Opening the window property , this string is also helpful to show quickly each position of the array values index.
All "value" properties, as for example *min, max, precision,* will limit each array numeric value : if the value property has min=1, max=10 and precision=1, all the array values will have min=1 and max=10 and will be represented with one decimal places.

As the "mono-dimensional" entities also for the "array entities" it's possible to set default values, too. Clicking on [...] button near the *value* property, the following window for editing the default values will be proposed:

Clicking on **Ok** button, the edited values will be accepted and automatically set up the *value* property like showed in the picture.
Clicking on **Cancel** button, the edited values are not accepted and will be maintained the previous values.

2.6.1   Compatibility and link

Two entity with *array* property greater than one are compatible one another only for the two following conditions:

1. if both the entities has the same array size (same *array* property value)
2. if the entities are of the same data type, with the exception :
    a. when start entity data type is CJ_BIT and linked entity data type is CJ_S_BYTE
    b. when start entity data type is CJ_BIT and linked entity data type is CJ_BYTE

In all other cases, entities are incompatible. Linking two incompatible array entities error message will be showed and the link aborted.

A mono-dimensional entities link from an array entities link shall be discriminated modifying the *Array Segment Thickness* from the *Settings* windows (*Array Segment Thickness* default value = 2) : in this way the thickness of the lines connected to the entities will be different.

2.7   Type CJ_CHAR for string management

By using type CJ_CHAR and the array property, is possible to manage a maximum 100 characters string. To use this function is necessary to set as entity type CJ_CHAR and the string characters number setting array property.

Eg. This setup identify a 5 characters string.

To set the single characters default value, click on the button that appears when you selected the value property, appears a window similar to the array values manage one:

Once the fields are set up click OK and the "Value" will be updated.

2.8 Using CodeEditor

CodeEditor is a graphic tool that allows you to specify the behavior of each algorithm. To open it, either double-click on the algorithm or choose *Show* from the pop-up menu.

An algorithm is characterized by a series of inputs, a single output and a code that describes its behavior.

CodeEditor is divided into several sections so that you can easily set and change all these characteristics: on the left side of the CodeEditor window there are two tables that summarize the properties of the inputs and the output; at the top of the window there is the algorithm category and at the center the built-in editor.

The entire project can also be saved from the CodeEdit (File menu) except when the I/O is changed.

Edit Input/Output
The input table shows their type and number; three properties are specified for each input:

➢ **EName**, or external name, which allows you to identify the input inside the sheet

➢ **IName**, or internal name, i.e. the name used in the code; it can be easily entered by double-clicking on the row

➢ **Dim** specify the size of an input or the output (Min 1, Max 100). If input/output is a mono-dimensional entity the property value is set to one. When the input/output is an array entity *Dim* property is the number of the array element. **Dim** is a modifiable propriety only for the following input/output of data type: CJ_VOID, CJ_BIT, CJ_BYTE, CJ_S_BYTE, CJ_SHORT, CJ_WORD, CJ_DWORD, CJ_LONG, for each other data type entities *Dim* property is set to the value 1.

➢ **Type**, i.e. the data type to be processed

If you want to add a new input without exiting CodeEditor, just press the *Add* button; the following window will be displayed:

where you can enter the new input's name, size and type.

If you select an input and press the *Modify* button, a similar window will be displayed where you can change the selected input's names and type; finally, by pressing the *Remove* button, you can remove the selected input from the algorithm.

The *"Modify output"* table allows to display and modify the algorithm's output. Naturally, the algorithm's output have to be only one. All outputs have the same rules of the input : also an output can have *Dim* property . *Dim* output data dimension can be modified between 1 and 100 only for : CJ_VOID, CJ_BIT, CJ_BYTE, CJ_S_BYTE, CJ_SHORT, CJ_WORD, CJ_DWORD, CJ_LONG.

At the top of the window there is the algorithm **category**. The concept of category is very similar to the concept of class in object-programming, and it allows you to use the same executable code for all the algorithms belonging to the same category, thus optimizing resources. Two algorithms of the same category must necessarily have the same number and type of inputs and output and the same code (see section **Optimizations**).

Edit Category
Press the *Change category...*, the following window will be displayed:

where you can enter a new category.

It is allowed to change the type of algorithm.

- Generic: generic algorithm.
- Glogal Algorithm: they define functions that can be called several times in the code of other algorithms (see 2.8.4).
- Singe Instane: algorithm that can be used only once in the project (useful when it contains some elaborations that must not be repeated in any way).

Edit code

In the center of the window there is the Editor section, where you can specify the algorithm's code. To change or add a code, check the *Edit* box: when you want to change an algorithm, the system will check for other algorithms in the same category; if there are, it will ask you to change category or else make the same change to all the algorithms of the same category.

The editor is capable of recognizing the ANSI C syntax and provides some functions to manipulate the code of the algorithm:

> ➢ code autocompletion
> ➢ importing code prepared using a different editor from a file
> ➢ exporting the code to a file
> ➢ printing the code
> ➢ making searches using find & replace
> ➢ cutting, copying, and pasting code portions
> ➢ using bookmarks

The code autocompletion function is extremely helpful for developers, as it allows them to complete the syntax of predefined structures or functions without needing to check the documentation every time.

In the case of structures, the autocompletion window will pop-up when the dot that comes before the structure's field name is typed.

For example, if you are defining an analog input *Probe* (data type: *CJ_ANALOG*) and you want to use the value field inside the algorithm but don't remember the right syntax, you may type "*Probe.*" in the code and wait for the autocompletion window to pop-up, as shown in the following figure:

If you press the *Ctrl* and *Space* keys at the same time, a new window will appear showing useful code editing information, such as:
  ➢ functions and constant factors of the different drivers that may be used in the algorithms
  ➢ the global algorithm defined in the project (prototype)
  ➢ C language keywords
  ➢ data types that may be used
  ➢ the names of the inputs in the algorithm
  ➢ project defines

By pressing *Ctrl- Shift-Space*, all the parameters of a function will be displayed.

When entering the code, it is highly advisable to follow certain important rules:
  ➢ remember to always return a value using the C syntax keyword **return** (or the compiler will give you a Warning)
  ➢ you cannot use static variables (i.e. assign the attribute **static** to a variable)
  ➢ it is advisable to use the types included in the UNI-PRO system, as described under section 3.1
  ➢ refer to the structured types if, for example, you have structure-type inputs (such as CJ_ANALOG)
  ➢ be careful when using loops and avoid heavy computational loads or, even worse, infinite loops!

The algorithm code may be defined at any time, provided that it is done prior to compiling.

2.8.1 Algorithm with array inputs/outputs

Here two simple examples about the array inputs and output correct way to write C algorithms with the UNI-PRO Code Editor :

Array inputs

Having two CJ_BYTE array inputs, both bigger in value than 1, now we have to return the algorithm to the sum of the values contained in the position *pos* of both the arrays inputs.

In this example the two inputs have the same array size, but they could have different size.
Both of the two inputs can contain five numeric information. For the access in every cell of the array use the following C syntax :

$$nameVar\ [\ index\ ]$$

index is the access index to the array, it start always from 0; therefore **the first array element is at the** *index 0*, **and the last array element is** *at the index Dim-1*.

2.8.2 Array Inputs and array output

Here a more difficult example of the previous. We want now to sum each two input array element of the same index and give back the resulting value to an array output.

For the code optimization use a "**for** cycle" to access at all the array indexes , without specify for each index the necessary operation.

The variable output of the Algorithm called *output*, differently from the algorithms with mono-dimensional output, is directly used in the code. Array output is used as it was an input parameter passed as *reference* (using the rules of the computer programming); the modification and the access to the output are directly make on the same output and not on a copy of the temporary variable.

Attention to write the keyword **return** in the following way: *"return; ".* As told before it is not involved in the elaboration, but it has to be reported in it : if not the compiler will give you a *Warning* during the project compilation.

When the output of an algorithm is an array entity, remember:
· the output will have to be directly used in the C code editing
· the name of the using output will have to be the same of the name set inside the **IName** property
· to access at the index *i* of an array use the syntax *nameVar [i]*
· valid index of the array are comprised between *0 and Dim -1*
· use **return;** instruction at the end of the algorithm to avoid a *Warning* during the project compilation.

2.8.3   Change the order of input terminals of the algorithms

Without having to remove connections, the order of the terminal inputs of the algorithms can be edited using the special arrows in the "Code Editor".

Note: Currently this function cannot be used if there are exported algorithm inputs.

2.8.4   Use of global algorithms

A global algorithm can be used in other algorithms, joining it with the name of the category and transmitting it the list of parameters using the proper order and type.
Picture to have created a global algorithm which category is "*max3Word*" giving back the greater of the three inputs. If you want to use it in another algorithm, you have to join it using the category "*max3Word*" (not the name of the entity) and transmit it the three parameters *word* using this order.

To call correctly a global algorithm you can search it in the project or look for it easily with the list of the functions through buttons Ctrl+Space in the items *GLOBAL function*.

2.9 Algorithms simulator

The complexity of the algorithm depends on the length of his code, maybe there is a single statement, perhaps a mathematical expression, but very articulate. Predict the performance and the end result could not be simpler.

AlgoSim is a tool that helps in this task, simulating what the hardware would perform with arbitrary values of the input.

The most practical way to explore it is through an example.

We suppose to perform an algorithm, called ReverseVector, which accepts an arry input in1, reverses the position of the values from head to tail, saves the result in the back out variable having the same type and size.

Once confirmed, right click on the Algorithm symbol and choose AlgoSim from the pop-up context.

The window of the simulator is similar to CodeEditor: the inputs are, the return value and the code. Pressing the Run ⊙ button, the box corresponding to the value of in1 flashes red and a caption points out that it is empty.

 Double click on the box and fill the grid with the desired values.

For simulation purposes it is essential that all inputs have indicated a value.

Click on the button Run ⊚ the simulator processes all the instructions at once. The result appears in the *Output* pane.

While clicking Run ⊚ several times, the result does not change because the vector in1 is always the same.

We suppose want to reverse the array first one way and then viceversa.
If one of the inputs and the return value have the same type and size, you can link them together so that the output is available to the desired input to the next round.
We're talking about feedback.

Right-click on the input parameter. The action Link output to this input as feedback ⟳ is enabled because the input satisfies the two requirements set out: type and size.

Input and output are now link, highlighted in blue.

The simulator loads the specified values one by one, executes the C code, save the result in the *out* variable and finally copy this into the input connected to for the next run.
At first, the vector *in1* contains the number -56 -55 -54 etc, the algorithm reverses it in the series -36 -37 -38 copying it into the *out*. Finally, the vector *in1* is overwritten by *out*.
Everything is done in just one click.

At the second run, both in1 and out contain the original series.

To stop the feedback *in1* just right click on the parameter *in1* and choose Unlink output from the input 🔄.

Without feedback, while executing the algorithm several times the result does not change because the input is unchanged.

2.9.1 Absolute breakpoint

Indicates a line of code where the execution must stop regardless of any factor.

The quickest way to set an absolute breakpoint is to click in the left margin of the pane that contains the code, or to the left of the line numbers.

Alternatively, click on the desired line or move the blinking cursor inside the box and then click the button ⊘ on the toolbar.

2.9.2   Conditional breakpoint

We suppose to stop running when a certain condition is satisfied.

Clicking the button ◆ appears a pop-up through which we can manage all the conditions to be encountered during the execution dell'algortimo.

As the caption suggests, the condition is expressed in the C language as a classic statement if-then-else.

In this example, click on button *Run* ⊙ the program performs 8 iterations before stopping and waiting for action from the user.

Clicking again *Run* ⊙ the program completes 5 more iterations, as many as there are from 8 to 13, before stopping and waiting for action from the user.

Clicking the *Run* ⊙ for the third time, the program will complete the remaining iterations showing the result.

2.9.3 Setting value

We suppose to execute one instruction at a time through the Step ⚡ button on the toolbar. Suppose also that you want to change on fly, the value of a local variable to try a different code stream.
In the example below, the algorithm always returns 23747.

After the program has awarded 0 to the variable selection, click the button ▦ and change the value using the pop-up overlay.

The assignment is expressed in C language without the punctuation mark **;** at the end.

Click on *Step* ⚡ the simulator considers an executive branch different from the previous being able to try, so quickly and comprehensively, the behavior of the algorithm respect to multiple situations of use.

2.9.4 Use during Debug

When the project is running in the hardware connected to your PC, thanks to the Debugger, you can observe the intermediate states and the evolution of the outputs to changes in inputs.

Right-clicking on any algorithm and selecting *AlgoSim*, it opens the simulator with the inputs already preset with the values assumed at that precise moment.

2.10 Creating libraries

The possibility of storing a series of objects in library format allows you to build up your own collection of objects which you can reuse in other projects. This enables significant development time savings and, at the same time, ensures that the know-how is maintained inside the company.

To create a new library, select the chosen element (an algorithm or a subsheet) and then do one of the following:

➢ right-click with your mouse and choose *New library* from the pop-up menu.

➢ choose Library/New library from the menu

➢ click on the [icon] icon in the toolbar

The following window will appear:

To create a new library, you need to specify certain characteristics that can be divided into four sections: *General Library Data*, *Options*, *Additional Data,* and *Type*.

In the *General Library Data* section, you can specify the library's name, the *Group* and the sub-folder *Category* of the library. Each library must belong to a group and must be associated to one sub-folder Category. Not is allowed create libraries without *Group* or *Category*.

The *Single Instance* property (if selected) adds to the library the particularity to being able to be used once in the project in which it be added. This function is useful when you create libraries which the multiple presence in the project can create problems in the program working; for example if in the library are contained algorithms with static variables, or functions to call firmware drivers of the application.

In the *Additional Data* section you can specify all the options available to customize each individual library; in fact, you can link it to an icon, and enter a version number, the author's name, the creation/revision date and a brief description.

In the *Type* section you can choose the library's type: a *Template* is a group of reusable entities that can be browsed, i.e. once you have added it to a project you can double click its icon to display and change its contents; conversely, a *Library* is a group of entities that cannot be opened or changed and, therefore, once you have created it, you cannot display its contents.

Once you have finished entering all data, click *Create* to confirm. If a library with the same name does not exist yet, the new library will be immediately added to the tree view libraries with the chosen icon, otherwise the system will require a confirmation before overwriting.
If you place your mouse pointer over the new library in the toolbar, you will be able to display certain information about it, such as name, version, latest revision date, description, and type.
To add it to a new project, drag & drop the library from the tree view to the sheet.

2.11 Creating EIML pages

EIML pages allow you to describe the controller's graphical interface in graphical format. A project may include several interconnected pages to display texts, icons, and variables representing the internal states of the controller.

To add a page to the project, choose *File/New/New EIML Page* from the menu or click the
icon in the toolbar. Once you have chosen the page type you intend to create, a node representing the page you have just added will appear in the project's tree and the visual representation of the page will appear in the center of the development environment.
EIML pages are associated with a certain type of display; for example, the figure below shows a blank page of a 240x128 pixel LCD graphic display.

EIML pages may contain texts, variables, icons, combos, tables, lines, and rectangles, all of which will be described in section 3.4.

The pages are created following a WYSIWYG approach (What You See Is What You Get). This means that the positions occupied, the font sizes, and the static properties such as alignment or text are displayed exactly as they have been set. To display dynamic-type properties such as flashing or cursor movement, you will need to switch to EIML page simulation.

2.12 Simulating EIML pages

While designing the graphical interface it can be very useful to simulate the actual behavior of the pages you are creating, to check that they are being correctly implemented. UNI-PRO has a built-in tool that allows you to display the output of a graphical interface page as if it were being executed in the chosen display. To enable this tool, go to the page you wish to simulate and choose

Project/EIML Simulator from the menu or click the [icon] icon in the toolbar. Below is an example of a simulation of a 240x140 pixel graphic page:

The simulator allows you to check that all elements inside the page are properly displayed (texts, variables, icons, tables, etc.), that all actions have been correctly implemented (moving through fields, editing and sending commands), and that the pages can be properly browsed.

On the left of the simulator window, there is an *Open file* button: press it to load an EIML page saved in binary format. Instead, by pressing the *Open Page List* button, you can load a group of pages saved as files to check for proper browsing through them. When the simulator is launched from within the UNI-PRO, the list of pages contained in the current project is loaded and the page being edited is proposed as first page.

By pressing *View ParamsDrv* you can display and change certain simulation parameters, included in the firmware as well, that affect the way pages are displayed.

For example, you can decide which characters should be used as time and date separators, whether years should be displayed as two-digit or four-digit values, and set data format.

By pressing the *View Vars* button, you can display a list of the variables used in the pages and, if needed, change their value during simulation.

A row is added for each variable required by the page, which summarizes the main properties and can also be used to change their values.

The *Save Image* and *Copy to clipboard* buttons are respectively used to save the current display to a bitmap file or copy it to the system clipboard so that you can *Paste* it into another application.

The *About* button displays a window showing the simulator's version.

The remaining buttons, ESC, ENTER, UP, DOWN, LEFT and RIGHT, simulate the behavior of the corresponding keys found on the user interfaces, thus allowing you to perform operations such as moving the cursor (UP and DOWN), editing a value (ENTER, UP, DOWN, ENTER or ESC), or browsing through the pages (ENTER on an element to which a page load command is linked, ESC to return to the previous page).

2.13 Selecting hardware

A project developed using UNI-PRO may be executed on the family of controllers compatible with this development environment. That's why you should choose on which ones you wish to execute the project. In order to allow developers to work more freely, the controller may be chosen either at the beginning of the design phase to build the project around the controller, or at the time of compilation, depending on the resources used.

To select the hardware, launch the *Hardware Expert* Wizard by choosing *Project/Hardware Expert* from the menu or clicking the icon. The Wizard will start with a presentation window.

Click *Next* to go to the second step where all controller types are listed.

When you select a controller, a brief description of the resources available in it will appear in the center of the screen and a preview of the controller on the right. The controllers can be different for the local Bus, which can be IntraBus or CAN (Controlled Area Network). In the *Hw type* section is requested the selection of the used hardware type, generic or special (S-Type)
Click on the appropriate controller and then click Next.

The third wizard step is divided in two sections: the first one allows to enable the CAN channels there are in the controller, in order to adapt it to the network where the project is; the second one allows to select, for each available UART, the protocol to use (e.g. None, ModbusSlave, BACnet, Ethernet, ...).

2.13.1 Modbus Master network

If on this section ModbusMaster protocol is selected on the right side appears a configuration button, click it to open the tool for the ModbusMaster network configuration.

On the left side there is a list with all the ModbusMaster profiles available; to add other profiles it is required to create the configuration driver. With the key ![plus] and ![minus] it is possible to add or delete the profiles there are in the network to realize. When the network are done it is required to set all the modbus addresses. Click Close to continue.

2.13.2 BACnet MS/TP Configuration

Selecting the BACnet MS/TP on the UART you can configure the communication properties.

BACnet MS/TP protocol is an alternative to BACnet IP protocol, and viceversa.
NOTE: Please verify in the Hardware Manual if the controller is compatible for the BACnet protocol.

2.13.3 Configuration of protocols over Ethernet

If one chooses to use the Ethernet port it is possible to set the fundamental parameters for the communication of the selected protocols over IP.

BACnet IP protocol is an alternative to BACnet MS/TP protocol, and viceversa.
BACnet IP protocol is an alternative to Web Server protocol, and viceversa.

NOTE: Please verify in the Hardware Manual if the controller is compatible for the BACnet protocol.

Sending Mail and SMS
If you want to configure the controller to send e-mail and SMS, you must first enable the Ethernet port in the Hardware Expert and then select Mail / SMS in the options available on this port. The management is then handled to the user application by using the appropriate standard libraries (refer to the *Standard Library Manual*).

On afterwards the Hardware expert allows to gain access to section "Expansions".

On the left side there is a list of all available expansions which, depending on the controller, may be connected to the Internal Can or the External Can.
The frame and buttons for the CAN bus will be available if the controller has this feature and if they have been enabled in the previous configuration step.

If you want to add an expansion, just select it and click the ![+] button of the CAN communication bus by which you want to link it: a new row will appear.
Once you have selected all the expansions to be added, click Next to proceed.

It is now possible to set the parameters that define the local or remote CAN network, as its physical address, the fact that the controller manages as master the CAN network, all the physical addresses of the others network elements:

Note. If the controller have to manage expansions is important to flag the Master controller field.

The fourth step provides a list of all possible user interfaces with a brief description of each one on the right. To add a user interface to the selected bus, proceed just as you would to add an expansion. In the last step, the wizard summarizes the configuration selected including controller type, serial port configurations, expansions and interfaces linked to each corresponding communication bus.

To terminate the Wizard, click *End*. To cancel all changes made, click *Cancel*.

2.14 Connection to physical terminals

Since the entities used in the UNI-PRO are a logical representation unrelated to the controller type, prior to compiling you will need to specify the connections between the logical resources used in the project and the physical resources available in the selected controller.

That's why all entity types representing a controller I/O (Digital Inputs, Digital Outputs, Analog Inputs, Analog Outputs, Clocks, LEDs, Buttons, Command Out, Command In and Buzzer) need to be joined to the local or remote physical I/O by means of the *Join Tools*.

Prior to discussing each individual Join Tool, we would like to point out the *Check Joins* function: it automatically checks all joins and, if needed, opens up the required windows to join those entities that have not been assigned yet.

2.14.1 Digital Inputs

The digital input joining wizard will start automatically during the Check Joins function if there are still Digital Inputs that have not been joined yet; you can also start it manually by choosing *Project/Join Tools/Digital Inputs* from the menu.

The window is divided into three lists: *Digital Inputs*, *Pins Available* and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the physical digital inputs that are still available, with the following characteristics:

- ➢ Node   : node of the controller where the resource is found
- ➢ Can   : bus to which the controller is connected
- ➢ Pin   : logical reference of the terminal
- ➢ Description : shows the description used in the hardware manual documentation of the concerned controller
- ➢ Hw name   : name of the controller where the resource is found

Finally, the third list contains the links already made between digital inputs and physical resources.
To link an entity and a resource, select the input from the Digital Input table and the physical resource from the Pins Available table and click the *Join* button. Both the entity and the hardware terminal will disappear from the list and a new row will be created in the "Joins Present" table.
Conversely, to remove a join just select the row that represents it in the *Joins Present* table and then click *Unjoin*: the table row will be eliminated, the logical terminal will be restored to the first table and the physical resource to the second. To confirm the link just make click *OK* else to clear the operations click *Cancel*.

2.14.2 Digital Outputs

The digital output joining wizard will start automatically during the Check Join function if there are Digital Outputs that have not been joined yet; you can also start it manually by choosing *Project/Join Tools/Digital Outputs* from the menu.

The window is divided into three lists: *Digital Outputs*, *Pins Available* and *Joins Present*. The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the physical digital outputs that are still available, with the following characteristics:

> Node   : node of the controller where the resource is found
> Can   : bus to which the controller is connected
> Pin   : logical reference of the terminal
> Description : shows the description used in the hardware manual documentation of the concerned controller
> Hw name   : name of the controller where the resource is found

Finally, the third list contains the digital outputs and physical resources that have already been joined.

To join an entity and a resource, select the output from the Digital Output table and the physical resource from the Pins Available table and click the *Join* button. Both the entity and the hardware terminal will disappear from the list and a new row will be created in the "Joins Present" table.
Conversely, to remove a join just select the row that represents it in the *Joins Present* table and then click *Unjoin*: the table row will be eliminated and the logical terminal will be restored to the first table and the physical resource to the second. To confirm the link just make click *OK* else to clear the operations click *Cancel*

2.14.3 Analog Inputs

The analog input joining wizard will start automatically during the Check Join function if there are still Analog Inputs that have not been joined yet; you can also start it manually by choosing *Project/Join Tools/Analog Inputs* from the menu.

The window is divided into three lists: *Analog Inputs*, *Pins Available* and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the physical analog inputs that are still available, with the following characteristics:

  ➢ Node   : node of the controller where the resource is found
  ➢ Can   : bus to which the controller is connected
  ➢ Pin   : logical reference of the terminal
  ➢ Sensor avail : list of sensor types that may be joined to the physical resource
  ➢ Hw Name   : name of the controller where the resource is found

Finally, the third list contains the analog inputs and physical resources that have already been joined.
To join an entity and a resource, select an input from the Analog Input table and a physical resource with a compatible sensor type from the Pins Available table, then click the *Join* button. Both the entity and the hardware terminal will disappear from the list and a new row will be created in the "Joins Present" table.
Conversely, to remove a join just select the row that represents it in the *Joins Present* table and then click *Unjoin*: the table row will be eliminated and the logical terminal will be restored to the first

table and the physical resource to the second. To confirm the link just make click *OK* else to clear the operations click *Cancel*

2.14.4 Analog Outputs

The analog output joining wizard will start automatically during the Check Join function if there are still Analog Outputs that have not been joined yet; you can also start it manually by choosing *Project/Join Tools/Analog Outputs* from the menu.

The window is divided into three lists: *Analog Outputs*, *Pins Available* and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the physical analog outputs that are still available, with the following characteristics:

- ➢ Node   : node of the controller where the resource is found
- ➢ Can   : bus to which the controller is connected
- ➢ Pin   : logical reference of the terminal
- ➢ Actuator avail: analog output type list
- ➢ Hw name   : name of the controller where the resource is found

Finally, the third list contains the analog outputs and physical resources that have already been joined.
To join an entity and a resource, select an output from the Analog Output table and the physical resource from the Pins Available table and then click the *Join* button. Both the entity and the hardware terminal will disappear from the list and a new row will be created in the "Joins Present" table.
Conversely, to remove a join just select the row that represents it in the *Joins Present* table and then click *Unjoin*: the table row will be eliminated and the logical terminal will be restored to the first table and the physical resource to the second. To confirm the link just make click *OK* else to clear the operations click *Cancel*

2.14.5 Clocks

The clock joining wizard will start automatically during the Check Join function if there are still Clocks that have not been joined yet; you can also start it manually by choosing *Project/Join Tools/Clocks* from the menu.

The window is divided into three lists: *Clocks*, *Node Available* and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the physical clocks that are still available, with the following characteristics:

  ➢ Node   : node of the controller where the resource is found
  ➢ Can   : bus to which the controller is connected
  ➢ Description   : name of the controller type where the clock is found

Finally, the third list contains the analog outputs and physical resources that have already been joined.
To join an entity and a resource, select an RTC from the Clocks table and a physical resource from the Node Available table, and then click the *Join* button. Both the entity and the hardware terminal will disappear from the list and a new row will be created in the "Joins Present" table.
Conversely, to remove a join just select the row that represents it in the *Joins Present* table and then click *Unjoin*: the table row will be eliminated and the logical terminal will be restored to the first table and the physical resource to the second. To confirm the link just make click *OK* else to clear the operations click *Cancel*

2.14.6 Leds

The LED joining wizard will start automatically during the Check Join function if there is at least one LED that has not been joined yet; you can also start it manually by choosing *Project/Join Tools/LEDs* from the menu.

The window is divided into three lists: *LEDs*, *Port Available* and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the physical LEDs that are still available, with the following characteristics:

- ➢ Node : node of the controller where the resource is found
- ➢ Can : bus to which the controller is connected
- ➢ Pin : logical reference of the terminal
- ➢ Description : shows the description used in the hardware manual documentation of the concerned controller
- ➢ Hw name : name of the controller where the resource is found

Finally, the third list contains the links already made between LED entities and physical resources.
To join an entity and a resource, select a LED from the first table and a physical resource from the Port Available table, and then click the *Join* button. Both the entity and the hardware terminal will disappear from the list and a new row will be created in the "Joins Present" table.
Conversely, to remove a join just select the row that represents it in the *Joins Present* table and then click *Unjoin*: the table row will be eliminated and the logical terminal will be restored to the first table and the physical resource to the second. To confirm the link just make click *OK* else to clear the operations click *Cancel*

2.14.7 Buttons

The button joining wizard will start automatically during the Check Join function if there is at least one button that has not been joined yet; you can also start it manually by choosing *Project/Join Tools/Buttons* from the menu.

The window is divided into three lists: *Buttons*, *Port Available* and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the buttons that are still available, with the following characteristics:

- ➢ Node   : node of the controller where the resource is found
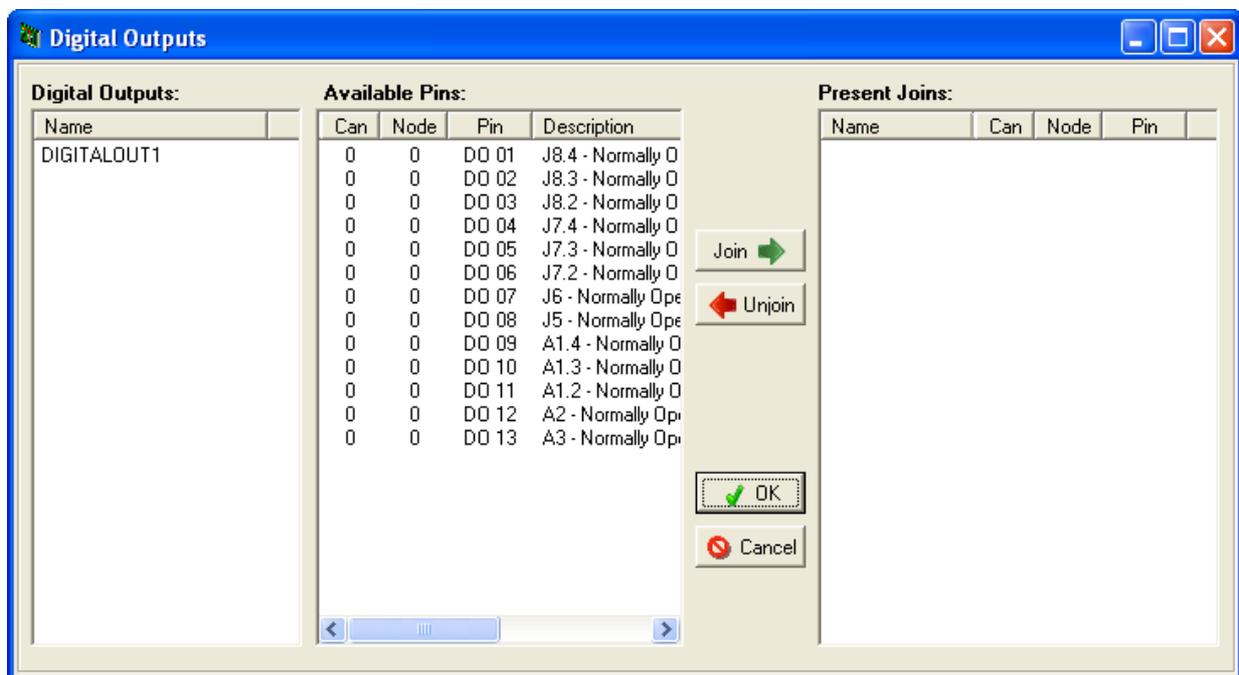- ➢ Can   : bus to which the controller is connected
- ➢ Pin   : logical reference of the terminal
- ➢ Description   : shows the description used in the hardware manual documentation of the concerned controller
- ➢ Hw name   : name of the controller where the resource is found

Finally, the third list contains the buttons and physical resources that have already been joined.
To join an entity and a resource, select a button from the first table and a physical resource from the Port Available table, and then click the *Join* button. Both the entity and the hardware terminal will disappear from the list and a new row will be created in the "Joins Present" table.
Conversely, to remove a join just select the row that represents it in the *Joins Present* table and then click *Unjoin*: the table row will be eliminated and the logical terminal will be restored to the first table and the physical resource to the second. To confirm the link just make click *OK* else to clear the operations click *Cancel*

2.14.8 Command In

The Command In joining wizard will start automatically during the Check Join function if there is at least one Command In that has not been joined yet; you can also start it manually by choosing *Project/Join Tools/Command In* from the menu.

The window is divided into three lists: *Command In*, *Node Available*, and *Joins Present*.
The first list contains the project entities that have not yet been linked to a node from which to receive the command.
The second list contains all controllers (including expansions) that were added to the project using the Hardware Expert wizard, from which you may choose to receive the selected command. Each element in this table has the following characteristics:

➢ Node   : controller node from which the command can be accepted
➢ Can   : bus to which the controller is connected
➢ Pin   : description of the controller node from which the command can be accepted
                      It can take two value types:
           • the value Broadcast, which means that the selected Command In can receive the command from any controller in the network;
           • the name of the controller type linked to the displayed Can-Node.
➢ Description : shows the description used in the hardware manual documentation of the concerned controller
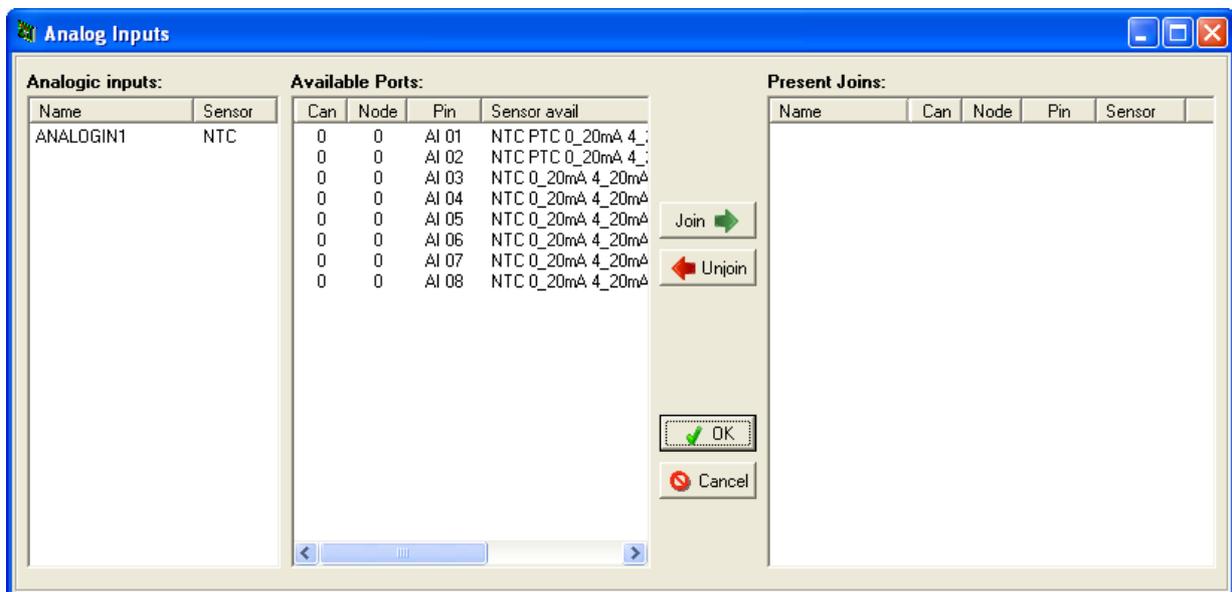➢ Hw name   : name of the controller where the resource is found

Finally, the third list contains the Command Ins and controllers that have already been associated.
To make a join, select a Command In from the first table, then select the node from which you wish to receive the command among those in the Node Available table and click the *Join* button. The selected Command In will disappear from the entity list and a new row will be created in the Joins Present table.
Conversely, to remove an association just select the row that represents it in the *Joins Present* table and then click *Unjoin*: the table row will be eliminated and the Command In will reappear. To confirm the link just make click *OK* else to clear the operations click *Cancel*

2.14.9 Command Out

The Command Out joining wizard will start automatically during the Check Join function if there is at least one Command Out that has not been joined yet; you can also start it manually by choosing *Project/Join Tools/Command Out* from the menu.

The window is divided into three lists: *Command Out*, *Node Available*, and *Joins Present*.
The first list contains the project entities that have not yet been linked to a node to which to send the command.
The second list contains all controllers (including expansions) that were added to the project using the Hardware Expert wizard, to which you may choose to send the selected command. Each element in this table has the following characteristics:

- ➢ Node : controller node to which the command will be sent
- ➢ Can : bus to which the controller is connected
- ➢ Pin : description of the controller node to which the command will be sent
  It can take two value types:
  - the value Broadcast, which means that the selected Command Out can send the command to any controller in the network;
  - the name of the controller type linked to the displayed Can-Node.
- ➢ Description : shows the description used in the hardware manual documentation of the concerned controller
- ➢ Hw name : name of the controller where the resource is found

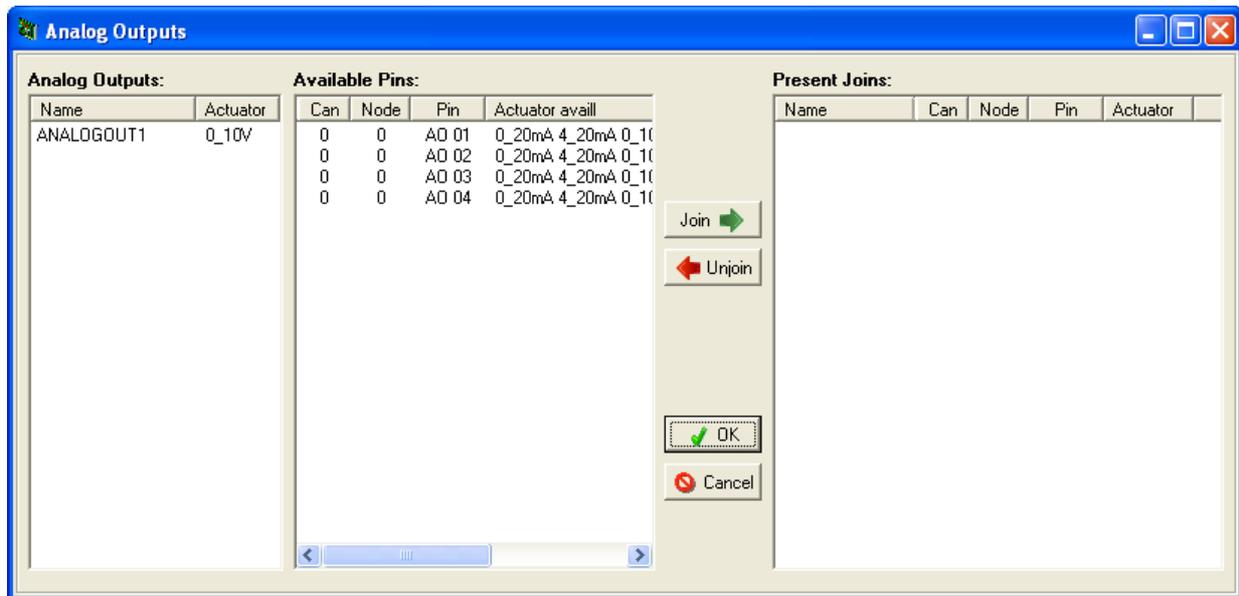Finally, the third list contains the Command Outs and controllers that have already been associated.
To make a join, select a Command Out from the first table, then select the node to which you wish to send the command among those in the Node Available table and click the *Join* button. The selected Command Out will disappear from the entity list and a new row will be created in the Joins Present table.
Conversely, to remove an association, just select the row that represents it in the *Joins Present* table and then click *Unjoin*: the table row will be eliminated and the Command Out will reappear. To confirm the link just make click *OK* else to clear the operations click *Cancel*

2.14.10 Buzzers

The buzzer joining wizard will start automatically during the Check Join function if there is at least one buzzer that has not been joined yet; you can also start it manually by choosing *Project/Join Tools/Buzzers* from the menu.

The window is divided into three lists: *Buzzers*, *Port Available* , and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the buzzers that are still available, with the following characteristics:

- ➢ Node : node of the controller where the resource is found
- ➢ Can : bus to which the controller is connected
- ➢ Pin : logical reference of the terminal
- ➢ Description : shows the description used in the hardware manual documentation of the concerned controller
- ➢ Hw name : name of the controller where the resource is found

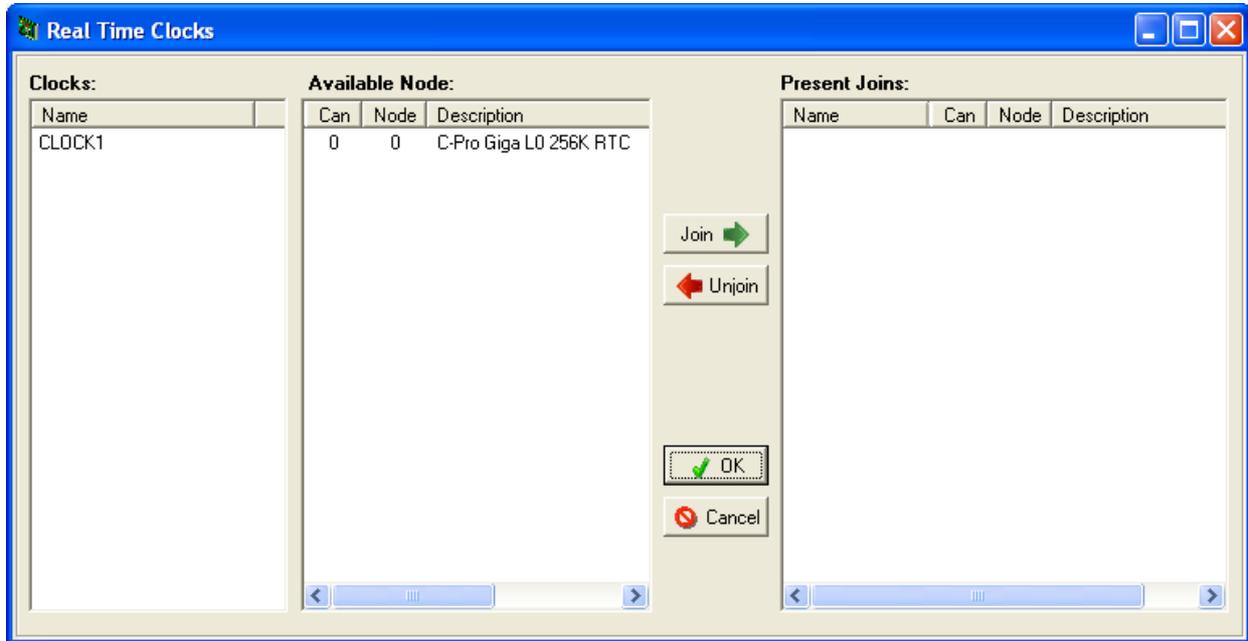Finally, the third list contains the buzzers and physical resources that have already been joined.
To join an entity and a resource, select a buzzer from the first table and a physical resource from the Port Available table, and then click the *Join* button. Both the entity and the hardware terminal will disappear from the list and a new row will be created in the "Joins Present" table.
Conversely, to remove a join just select the row that represents it in the *Joins Present* table and then click *Unjoin*: the table row will be eliminated and the logical terminal will be restored to the first table and the physical resource to the second. To confirm the link just make click *OK* else to clear the operations click *Cancel*

2.15 Compiling the program

Once you have saved your project, chosen the hardware to execute it on, and successfully run the Check Join wizard, you are ready to go on to the compilation phase by choosing the

Project/Compile menu or clicking the icon in the toolbar.
The compilation phase can be divided into five sub-phases:
- ➢ Project is automatically saved.
- ➢ The source file is generated.
- ➢ Compilation

> ➢ Linking
> ➢ Available resources are calculated

If the project had not been saved yet, you will be prompted to specify a file name to save it to. Then, the project is analyzed and the source files are created, which will be compiled and linked to generate the executable file that will be downloaded to the controller through the Download function. Finally, the available resources are calculated: the amount of FLASH and RAM memory still unused.

While source files are being compiled, a window is displayed in the foreground with a progress bar showing the completed process percentage and an Abort button, to interrupt the process.

In addition, a window that progressively displays the result of compilation appears at the bottom of the environment. If the compiling process ends successfully, the object files are linked, the executable file to be subsequently downloaded is generated, and a report showing all FLASH and RAM resources still available is displayed.


```
14.36.17: Checking hardware ... ok!
14.36.17: Checking joins ...... ok!
14.36.17: Checking UI ......... ok!
14.36.18: Saving projects ..... ok!
14.36.18: Generating code ..... ok!
14.36.18: Compiling .......... ok!
14.36.19: Linking ............ ok!
14.36.20:
14.36.20: Compilation end at 14.36.20 ( 0.00.01 ).
14.36.20: Errors: 0.
14.36.20:
14.36.20: ************ FREE ************
14.36.20: RAM   : 4286 bytes
14.36.20: FLASH : 168326 bytes
14.36.20: ****************************
```

If compilation does not end successfully, each detected error will cause a row to be displayed.


```
14.38.30: Checking hardware ... ok!
14.38.30: Checking joins ...... ok!
14.38.30: Checking UI ......... ok!
14.38.31: Saving projects ..... ok!
14.38.31: Generating code ..... ok!
14.38.32: Compiling ..........
*** RientroManuale (2) - Error 4062C : syntax error near `/'
14.38.33: Compilation end at 14.38.33 ( 0.00.00 ).
14.38.33: Errors: 1.
```

Each error row offers many pieces of information that can be interpreted as follows:

- ➢ Category : category of the algorithm where the error was detected
- ➢ Line : line where the error was detected
- ➢ Error Type : Information, Warning, Error, Fatal Error
- ➢ Error Code : code assigned to the error (see **Compilation Errors**)
- ➢ Error Description : a brief description of the error

Double-click on the row corresponding to the error you wish to correct; the CodeEditor of the algorithm in which the error was detected will open automatically and the cursor will be placed in the relevant line.

2.15.1 Using the cache

In the frame *Settings* it is possible to set the use of the cache for compiling the project files. For each project it is saved a cache of the file to compile and when compiling they will be recompiled only the files that have had modifications. This option makes quicker the process. To enable the cache select the flag *Enable cache* in *Settings*. If it is desired to cancel all the cache of the compiled projects, make *Clear Cache* from *Settings* of the environment. If it is desired to cancel only the cache of the open project, make eseguire *Clear Cache* from section *Options* on the project property.

In the main menu *Project* the *Clean* entry allows you to delete the cache of the open project and all temporary compilation files.
In the main menu *Project* the *Build All* entry allows you to compile the project completely by running a *Clean* operation first.

2.15.2 Compiling with browser EPJ-Color

If a browser Vcolor 3 is foreseen, i twill be necessary to make two operations of compiling to full compile the project. A compiling is for the controller and the second is for the browser since the EIML color pages and other projects must be compiled and downloaded directly in the browser.
The compiling project starts compiling the files for the controller; if this phase is successfully completes the seconds automatically runs (for the browser). Once the compiling finishes, the object files of the application are generated.
The object files (ucjb e ucje) of the browser are different fom those of the controller because of the name, which will finish with*" _Browser"* the classic project name.

For example, if the net of the project TestProject is made of a controller C-Pro 3 Micro Basic and a browser EPJ-Color 320x240, the result of the compiling will be the following:
Object file of the controller:
- TestProject.ucjb
- TestProject.ucje
Object file of the browser:
- TestProject_Browser.ucjb
- TestProject_Browser.ucje

2.16 Downloading the program

Once you have finished compiling, the executable code can be downloaded into the microcontroller. Before starting the download, check that the hardware is properly connected to the serial port and ready for programming (see the hardware manual). To start the program download, choose the

Project/Download menu or the icon [icon] located on the toolbar. The following window will appear:

Download JET

Address: 0x000000

0%

Abort

If the connection with the board is successfully established, after a few seconds the progress bar begins to expand and once downloading is complete, a box will appear indicating the successful outcome of the operation.

Otherwise an error message will appear; in this case reattempt the operation after having verified certain conditions:

➢ The serial port selected in the environment options (which can be activated from the *Tools/Settings* menu) must be that for which we wish to perform the download
➢ the serial cable must be connected both to the computer and to the controller

If the bar does not progress and an error message appears, check the connections, try to reset the controller again and restart downloading.

To change download options, see section **Environment configuration**.

2.16.1 Download and programming via USB cable

The procedure related here is valid only for the controllers c-pro 3 micro/kilo; these controllers can be connected to the Pesonal computer via USB and the binary generated when compiling is made of two files instead one.

    1.   *Project name.*ucjb: binary file of the application
    2.   *Project name.*ucje: binary file og the EIML pages

Since there are two files, the way to program changes; anyway the user will be asked for just one operation. The steps are:

-    binary download of EIML pages (*Project name.*ucje)
-    binary download of the application (*Project name.*ucjb)
-    restart-up of the controller
-    microcontroller memory programmation, using the binary that have been found
-    file generation with the programmino report.

For these controllers it is not necessary to set any COM port (for programming/debugging); it is automatically recognized by UNI-PRO. It is very important to verify there is just on e controller connected via USB because, otherwise, the wrong one could be programmed.

Once the procedure has finished, a text file is displayed; it contains a series of information on the programming procedure or a message relating the reason why it has not been possible to program the controller.

The visibility of this file depends on the status of the property *"Show Report After Download"* in the windows *Settings* of UNI-PRO.

It is necessary the COM port is automatically recognized because after a reset of the controller it runs in modality mass-storage (it means it is not a virtual COM); it is the development environment that, through commands, converts it from mass-storage to virtual COM, obtains the value and uses it in the various functions asking for the serial port.

2.16.2 Download with browser EPJ-Color

When in the CAN network there is a browser Vcolor, the compiling will create, in addition to the binary files for the controller, also the binary files for the browser; it means also the browser EPJColor must be programmed. Once the Download is selected, a window will appear to decide the hardware to use for the download.

To the end of the downlaod it is possible to repeat the procedure for the following devices.

2.16.3 Download and programming via USB key

For all the controllers with USB port it is possible to download the program using a standard USB flash drive (pendrive), following the below procedure.
- Rename the controller's binary files generated during the compilation:
    o TestProject.ucjb
    o TestProject.ucje
  Respectively in:
    o work.ucjb
    o work.ucje
- Copy them in the USB flash drive root.
- Insert the pendrive in the USB port with the controller power supplied.
- Wait the automatic copy of the files in the internal memory and then the reprogramming of the controller.
- This procedure is terminated when the controller flashes during the reset autotest.

The USB flash drive (pendrive) must be formatted as FAT32.

NOTE.
During the programming procedure the controller executes the following tests:
- If the files work.ucjb and work.ucje are equals to those in the controller memory, they are not copied again and, of course, the controller doesn't reprogram itself.
- The controller verify the congruency of the binary file after the copy.
- The controller verify if the hardware family code of the binary file and the controller are the same to avoid unrecoverable downloads.

In case of compilation using a VColor browser, it is possible to follow the same programming procedure using a USB flash drive also for the VColor display.

- Rename the display's binary files generated when compiling:
  - o TestProject_Browser.ucjb
  - o TestProject_Browser.ucje
  Respectively in:
  - o work.ucjb
  - o work.ucje
- Copy them in the USB flash drive root.
- Insert the pendrive in the USB port with the display power supplied.
- Wait the automatic copy of the files in the internal memory and then wait the reprogramming of the display.
- The procedure is terminated when the display shows the init page during the reset autotest.

2.17 Use of the Debugger

The debugger allows you to check the function of a program under execution by the controller. The debugger can be activated by a "reset" (for example right after the download of the program) or "live" (without stopping the application). The latter operation is very useful during testing and fine-tuning the machine control.

Using the debugger, the status of all internal (variables, timers, and parameters) and external (inputs and outputs) entities can be monitored directly from the graphic entity used for development. The parameter values and the statuses can be modified, as well as simulating analogical and digital inputs, interrupting the program via conditional breakpoints, performing the jump to execution operations for a specific calculation, or restarting from the initial condition. Entire debug sessions can be performed while investigating malfunctions or simply to accelerate the functional test phase for any adjustments made.

To activate the debug mode, the program must be compiled only after the Debugger protocol has been selected on one of the controllers free serial ports. Or enabling the *Enable DEBUGGERr*
Option on the *BUS ENABELD* section: This operation is performed in the Hardware Expert during the serial port protocol configuration phase (see the figure below).

It is also necessary to configure the serial COM port of the PC to be used to communicate with the controller. In the case of Debugger via Ethernet, you must also set the IP address and TCP port for communication, using the following page that can be accessed from the *Tools/Settings/Debugger* menu:

Finally, once the PC port is connected to the port of the controller via an RS232, RS485 or Ethernet connection, everything is ready for a debug session to begin. To activate it, select *Debug/Start Debugger* from the menu, or the ![D] icon located on the tool bar.

The program will connect to the controller and perform the controls per design. If the check has positive results, the *Debugger started!* message will appear in the output window.

And the following icons will appear on the command bar:

At this point the values of the program entity can be viewed simply by running the mouse pointer over the variables and waiting for the yellow prompt window to appear.

This window will only remain active fore a few seconds, to view it again you must move the mouse. If you want to view the value of a specific entity, you can enter the Watch window using the special

menu (right key of mouse) and select Add Watch, or use the ➕ icon located on the tool bar. A watch window will appear on top of the status window:

Up to 4 variables can be added to the Watch window. To remove them, remove each one individually using the special menu (right mouse key) and selecting Remove Watch, or using the

icon located on the tool bar. All of the variables in the watch window can be removed at the same time by selecting *Debug/Remove All Watch* from the menu.

When controlling congruency between the project on PC and that on the controller, the following information is compared:
   o Project number
   o Project version
   o Vendor ID
   o UNI-PRO software version
   o Compile date
   o Number of entities in the project
   o Number of tasks in the project

From these, 1, 6 and 7 must coincide otherwise the debugger will not start. When diagnosed, the others result in a warning message but allow the debug session to be activated.

To exit from the debug phase, select *Debug/Stop Debugger* from the menu, or the ⬛ icon located on the tool bar.

To stop program execution, select *Debug/Break* from the menu, or the ⏸ icon located on the tool bar. The program stops while executing an entity, and the index for the entity is indicated in the

status bar: Device break (1:4) This corresponds to the task required for calculation, as listed in the Call List:

Calls List

[Main] | Timed 5 Ms | Timed 100 Ms |

```
0: TIMER1
1: Identity_CJ_ANALOG
2: VAR2
3: LE_1
4: DIGITALOUT1
5: LED1
6: LED2
```

Close

In the sample shown, program execution stopped at the calculation of the DIGITALOUT1entity, corresponding to task number 4.

When the program is stopped all of the internal counters and timers are "frozen" but not the inputs or Real Time Clock. It is also possible to observe the value of the entity or to view a few in the watch window.

There are two possible methods to restart the program: The first is to execute a portion of the program until another entity is calculate using the *Debug/GoTo* command or the icon located on the tool bar. In the example shown, if you want the program to be performed until the LED2 entity, the calculation in tasks 5 and 6 will be performed after which the program will once again stop.

Alternatively, the program can be restarted in a continuative manner using the *Debug/Run* command or the icon located on the tool bar. In this case, the program will continue execution starting from the point where it was stopped.

If you want to restart the program from the beginning, it is necessary to perform a controller reset using the *Debug/Reset* command or the icon located on the tool bar. Thereafter it can be restarted with the Run command. The reset operation zeroes all of the internal variables, timers/counters, in exactly the same manner as a reset after a power outage.

It is possible to set the value of the entity selected, both when the program is running and when it is not, using the *Debug/Set Value* or the icon located on the tool bar. A window will appear where a new value can be set and to confirm it in writing.

Set Value

Entity: PAR2

Type: CJ_WORD

Value: 1234

✔ Set   🚫 Cancel

The values of the analogical and digital input values can also be set. In this case, the input, once set, will move into the "Input Simulation" mode and will not accept any values that are sent from the controller's conversion driver. This situation will be maintained until you exit the debugger mode.

To perform a debug session where it is necessary to investigate the causes of a possible malfunction, it is very important to be able to insert breakpoints in a number of points throughout the program. In this manner the software controls the occurrence of a few break conditions and immediately stops the controller program when these conditions occur.

The breakpoint can also be set without conditions, so that the program can stop exactly in the point of execution where the breakpoint was located. To insert and unconditional breakpoint for a specific entity, it must be selected and the *Debug/Set Breakpoint* command must be activated, or the icon located on the tool menu. This type is useful if you want to check the behavior or the program within a single main loop or between main loops, making the program stop at each entity where a breakpoint is set.

However, often it is more useful to investigate what happens when specific conditions occur. In these cases conditional breakpoints are needed, which can be activated using the *Debug/Set Conditional Breakpoint* command or the icon located on the tool bar. A window will open where you can select the condition to compare with the value of the controlled entity.

The conditions can be selected from the following list:

EXECUTE	No conditions (equivalent to an unconditional breakpoint)
EQUAL TO	The program will stop with the entity assumes the value set in the Value field.
DIFFERENT FROM	The program will stop with the entity assumes a different value than the one set in the Value field.
LESS THAN	The program will stop with the entity assumes a value less than the one set in the Value field.
LESS OR EQUAL	The program will stop with the entity assumes a value less than or equal to the one set in the Value field.
MORE THAN	The program will stop with the entity assumes a value greater than the one set in the Value field.
MORE OR EQUAL	The program will stop with the entity assumes a value greater than or equal to the one set in the Value field.

When a breakpoint is set, a window will appear on top of the status window:

Index	Entity Name	Condition	Value	Enable	BreakIdx	Idx	SubIdx	
1	VAR2	EQUAL TO	1	Enable	0	15136	0	

Output Breakpoints

Up to 4 breakpoint conditions can be set, which will appear in this window. The *Condition*, *Value*, and *Enable* properties can be modified in this window. To remove a breakpoint, remove each one individually using the special menu (right mouse key) and selecting Remove Breakpoint, or using the icon located on the tool bar. All of the breakpoints can be removed at the same time by selecting *Debug/Remove All Breakpoints* from the menu.

A conditional breakpoint can also be set for the occurrence of a specific event, such as the press of a button or the arrival of a command. In this case, the Value field assumes the following meaning:

Value = 0   No event
Value = 1   Button pressed or command intercepted

NOTE: In the event that the output of an algorithm or a library has more than one branch, such as the previous example of library LE_1, the development system will perform the intermediate calculation for this output. Consequently, during the debug phase a breakpoint can be set for this ;or it can be viewed using the mouse or in the watch window, exactly as if it were a "hidden" variable. However, it is not possible to perform a Set Value on it.

2.17.1 Present value in debug

While debugging a project, under the calculated entity, the present value that the entity assumes during the operating cycle appears. This makes the entire sequence of the values that change while running the program visible at all times.

If a digital input or analogical input is forced in debug at a determined value, disconnecting therefore from the physical acquisition, it is highlighted with a different colour.

The colour of the label can be changed with the DEBUG value, using the *Settings* window, in the *Debugger* tab.

2.18 Web Server

Some controllers featuring Ethernet interface can be connected from a browser that displays web pages residing inside.
The device executes a minimal web server that replies to HTTP clients connecting from TCP/IP network. The server publishes the files stored in the flash memory by means of the development environment, using the *Project/Download WebSite* menu.

Thanks to programming techniques such as AJAX and CGI, can be made interactive web applications through which view information and change values. The browser becomes an extension of the front panel where you perform some of the operations normally carried out with the latter. Warning: it will never replace it especially where safety regulations are involved.
For example, you might watch the temperature of a room and change the speed reference of an inverter controlling the air extraction.
The controller is theoretically accessible from any terminal connected to internet, even from a tablet while you are sitting comfortably at home.

Once you open the wizard *Hardware Expert*, choose a controller having an Ethernet interface.

In the next step, click on the combo box and enable *Ethernet*.
This makes visible a button immediately on the right.

Clicking on the button *Ethernet TCP/IP configuration*, the following window opens

Make sure the option *Enable web server* is ticked.
The default port number listening is 80; in most cases keep so.

The *Password* box contains the secret word to protect unauthorized write of the parameters. The *Password Level 1, Password Level 2 e Password Level 3* boxes permit to prevent unauthorized access to pages and functions using three authentication levels. If empty, the protection is turned off.

The option *Include File for Remote Download* active or not the possibility of including in the generation (and download to controller) of the WebServer the file to download remotely the application.

Html pages are written by the favorite authoring tools requesting, as usually happens, many small and incremental changes whose result should be checked every time. In order that work isn't tricky, you need to publish quickly the pages.
File transfer toward the controller is improper both because slow and because it engages the flash memory that, despite the high nominal performance, has a limited number of write cycles. To avoid these problems a Web Server simulator has been implemented.

2.18.1 Web Server Simulator

Uni-Pro embeds a web server running on the computer that simulates the behaviour of the one running in the controller.

The author saves the pages in a folder of hard disk modifying them at will. Uni-Pro draws here in real-time refreshing the browser. Being able to publish pages locally, their writing becomes easy. Only when it's advisable an ongoing test or when work is finished, you publish the pages in the controller.
Once this is plugged in, the browser interacts through the Debugger.

From the menu *Tools/Settings* choose the tab *Web server*.

The option *Active on debug* means whether the server runs at the start of Debugger, so when the controller is communicating.

The box *Document root* indicates the default top folder from which the tree of the website expands. The option *"Document root in current project folder"* generates instead the root of the website into the current project path.
Suppose it's set to C:\Jobs\Uni-Pro and there's the project C:\Jobs\Uni-Pro\chiller_alfa.ucjp .
If absent, when Debugger starts the folder C:\Jobs\Uni-Pro\www\chiller_alfa\ is created.
In other words, for the project XYZ will be created the folder www\XYZ\ attached to the one specified in the general settings. Here must be saved html files, JavaScript, images, and anything else that will be transferred in the controller.

2.18.2 Exported entities
The entities that can be read and written via CGI are first exported by clicking the menu *Tools/Export Entities*, checking "X" in the *BMS* column.

Using the automatic generation of the table for the web, from menu *Project/Generate Entities Web Table*, within the site folder a file named *XYZ (exported register var).txt* is created in json format containing the ticked properties of the list shown in the previous picture. This file could be renamed XYZ.js and this name will be called from inside the HTML page:
<center>*<script src='XYZ.js'></script>*</center>
Furthermore it is necessary to declare as variable the array of records so generated, for example:

> *var MyData =*
> *{*
> *"tab_prop": [*
> *{*
> *"cat": "Par",*
> *"webServer": "",*
> *"group": "PARAM",*
> *"descr": "",*
> *"type": "CJ_LONG",*
> *"prec": "0",*
> *"name": "PAR1",*
> *"addr": "10"*
> *},*
> *...*

}

The file isn't overwritten if existing, so any changes you make are retained.

We highlight the properties called *Group* and *WebServer*.
Unlike the others, *Group* doesn't exist as static property in Uni-Pro but it's dynamically generated. It may be helpful to group entities in arbitrary sets. Unless otherwise specified, the name is pre-filled according to the following fixed rule

> Entity's category: Var, Pers, Command In, Clock, Timer, Led, Buzzer e Button
> Reference group: STATUS
>
> Entity's category: Digital input
> Reference group: DIGITAL_IN
>
> Entity's category: Digital output
> Reference group: DIGITAL_OUT
>
> Entity's category: Analog input
> Reference group: ANALOG_IN
>
> Entity's category: Analog output
> Reference group: ANALOG_OUT
>
> Entity's category: Par
> Reference group: PARAM

WebServer is a property designed to exchange data of any kind between the project and the web application, so it is freely usable. It also allows you to overwrite the current values of the other published properties.
Suppose there's the entity VAR1_Long. If the WebServer property is empty, the following record could be generated:

```
{
  "cat": "Var",
  "webServer": "",
  "group": "STATUS",
  "descr": "",
  "type": "CJ_LONG",
  "prec": "0",
  "name": "VAR1_Long",
  "addr": "23"
}
```

Setting the above property to "Sierra; GrOUp=PUMP;06501; nAMe =Pump Hours;rtFoo;preC=1;" we would find in the created file

```
{
  "cat": "Var",
  "webServer": "Sierra;06501;rtFoo;",
  "group": "PUMP",
  "descr": "",
  "type": "CJ_LONG",
  "prec": "1",
  "name": "Pums Hours",
  "addr": "23"
}
```

If *WebServer* contains a pattern *key=value;* and *key* represents one of the published properties, then *value* replaces its current data.

2.18.3 Operation

A change in the general settings is applied when Debugger starts again.

The local web server connects with either localhost or 127.0.0.1 .
Typing `http://localhost` in the address bar of the browser, the server automatically searches for index.html as default home page. If the URL contains a path that doesn't exist, you will see the error code 404.

In order to transfer the site, click on the menu *Project/Download web site*.
The following preliminary tests are performed and have to be satisfied.
1) The folder hasn't to exceed the maximum size equal to the capacity of the controller (for example 2MB). Otherwise the browser displays the error code 412 against which the content needs to be cleaned up and optimized.
2) The file names cannot be longer than 24 characters.
3) The site must have one and only one level, i.e. subfolders aren't permitted.

The web page interacts with the controller via the embedded CGI program.
The call foresees some parameters we discover right now.
Warning: the syntax is case sensitive. While the name of CGI is uppercase, what follows the question mark is lowercase.

2.18.4 Reading entities

In order to get the actual status of all exported entities type

```
http://localhost/ODJET_CGI?pread=all
```

getting as answer an array formatted json

```
{
  "tab_param":[
    {"addr":"100","value":"698","min":"0","max":"65535","def":"0"},
    {"addr":"384","value":"0","min":"0","max":"65535","def":"0"},
    {"addr":"525","value":"-32767","min":"0","max":"0","def":"0"},
    {"addr":"1280","value":"0","min":"0","max":"0","def":"0"}
  ]
}
```

addr is the address of the Modbus register, while def represents the default value. Wanting to read a discontinuous set of entities, type an URL like

```
http://localhost/ODJET_CGI?pread=6,8-10,12
```

getting as answer an array

```
{
  "tab_param":[
    {"min":"-32768","max":"32767","def":"89","value":"89","addr":"6"},
    {"min":"0","max":"255","def":"121","value":"121","addr":"8"},
    {"min":"-32768","max":"32767","def":"1354","value":"1354","addr":"9"},
    {"min":"-32768","max":"32767","def":"30059","value":"30059","addr":"10"},
    {"min":"-32768","max":"32767","def":"-30060","value":"-30060","addr":"12"}
  ]
}
```

in which appear the addresses 6, 8, 9, 10 and finally 12.
Pread parameter accepts comma-separated elements each of which can be either a single address or a range defined by limits limA-limB. The latter represents the continuous sequence of integers ranging from limA to limB.

If there are no exported entities or the address is unknown, the answer is

```
{ "tab_param":[] }
```

2.18.5 Writing an entity

Suppose you write the capital letter A in a variable of type CJ_CHAR exported to the Modbus address 8. The instruction to launch is

```
http://localhost/ODJET_CGI?pwrite=8&value=65
```

where the value parameter can take exclusively numeric values.
The browser receives an array in json format containing the entity in question. The answer can be used to understand if the data was successfully written or an unexpected condition has occurred.

```
{
  "tab_param":[
    {"min":"0","max":"255","def":"121","value":"65","addr":"8"}
  ]
}
```

2.18.6 Protection to write

In order to know the status of content protection, launch the URL

```
http://localhost/ODJET_CGI?ppsw=status
```

If it is turned on, the answer is

```
{
  "tab_system": [
    {"ppsw": "lock"}
  ]
}
```

Conversely if it is turned off, the answer is

```
{
  "tab_system": [
    {"ppsw": "unlock"}
  ]
}
```

In order to activate the protection, type

```
http://localhost/ODJET_CGI?ppsw=lock&value=secret_word
```

where *secret_word* is the password set for the project in Hardware Expert.
In order to deactivate the protection, type

```
http://localhost/ODJET_CGI?ppsw=unlock&value=secret_word
```

For both lock and unlock, the answer equals to the one got by the status command.

Suppose protection is enabled and the password is 0$oZt6Q44.
In an attempt to remove the lock, suppose we send to the controller

```
http://localhost/ODJET_CGI?ppsw=unlock&value=SierraTango
```

It replies

```
{
  "tab_system": [
    {"ppsw": "lock"}
  ]
}
```

because SierraTango doesn't match 0$oZt6Q44; the lock is retained.

2.18.7 Web Pages Protection

There are two more CGI commands for the login and to verify the active user level.

Command to read the active user level:

/ ODJET_CGI?stslev=1

The answer is
```
{
    "tab_loguser":[
                    {"usrlog":"3","result":"1","err":"0"}
    ]
}
```

Where usrlog indicates the active user level (1 = level 1, 2 = level 2, 3 = level 3). The value 0 indicates no user logged.
The field result equal 1 indicates no errors.

Command for the login:

/ODJET_CGI?levpsw=Password&lvl= Level_Number

Where Password must be substituted with the password digited (case sensitive) and Level_Number is 1,2 o 3 depending on the authentication level requested.

The answer is
```
{
    "tab_loguser":[
                    {"usrlog":"3","result":"1","err":"0"}
    ]
}
```

Where the fields have the same meaning before.

In case of errors the returned value has the following meaning:
      1: requested level not supported
      2 – 3 : number of characters greater than the maximum value
      4: error wrong password inserted
      5: requested level not active (when you try to log to a not enabled level)

After a correct logon procedure, the navigation between web pages of thet level is allowed until a fixed timeout of 5 minutes. When expire it, a new page request is rejected showing the error code 401. This timeout is reload every page requested, not during CGI data exchange. So it is possible to remain within the same protected page for a time greater than the timeout.

2.18.8 Datalog remote management

Two CGI commands can be used to read the information about the Datalog status and to download the CSV file from the remote browser.

Command to read the Datalog status:

/ODJET_CGI?pdatalogf=1

Web server answer:

```
{
 "status_datalog":[
                    {"num_rec":"23","status":"ok_run"},
                    ]
}
```

Fields description:

"status":   uses the following return strings:

"no_link"
 "ok_stop"
"ok_run"
"err_compatible_mem"
"err_rtc"
"err_corrupted_mem"
"err_insufficent_mem"
"busy"   <-- in progress a USB data extarction

"num_rec": != 0   <- number of records at the moment filled
            0   <- no records

Command to download the CSV file:

/ODJET_CGI?pdatalogf=csv

Send the command to the browser with the file download request.

With this command there are no JSON answers.

NOTE: Before to send the file download command, it is important to be sure that there are no datalog errors and the number of records is greater than zero, otherwise no feedback in the browser could be bad intepreted.

NOTE: Is advisable to do the CSV download operation in safety mode (for example when the unit is stopped), to avoid the application runs slowly during the CSV file preparation before to transmit it.

2.18.9 Network configuration info
It is possible to ask the web server info: controller MAC Address and IP and network configurations. The command to use is the following:

/ODJET_CGI?pnet=all

Answer:

```
{
    "MAC_ADDRESS": "00:AC:DD:BB:CC:DD",
    "IP": "192.168.0.1",
    "SUBNET_MASK": "255.255.255.0",
    "GATEWAY": "192.168.0.1"
}
```

Possible parameters:

           all -> all information returned
           0 -> only MAC returned
           1 -> only IP address
           2 -> IP , SUBNET, GATEWAY returned

2.18.10 Automatic Website Generation

Enabling the option *Automatica Website Generation* it is possible to have a web page with a table where it is possible to group all the exported entity info in columns ordered by belonging type:

This generation is automatic, you have only to follow the next steps, without any knowledment of web programmino languages or tools. Of course the possibilità to personalize the page is limited to the logo choice, the name and number of the groups that will appear in the Tab and the entity names

as showed in the previous table. It will be possible to read the values continuosly refreshedand, protected by password, it will be also possible to write them.
The entities that could be showed in Web table are the only one exported in the *Register* list   with the option *BMS* checked.

Enabling the option *WebServer*, you can compile and download the program. Then you have to generate the table of the Web Entitities, directly from the menu *Project/Generate Entities Web Table* :

With this procedure, you can generate the info neeeded to create the table and manage the communication with the variables. Moreover all the files necessary to the Web site management, are copied from the folder *Uni-Pro3 \ Config \ Webserver files* to the target directory *www*.
The project configuration is possible through the use of special keywords to use in the property *WebServer*:

These keywords assign a different representation of the table than the default table. For example the previous entity Var, should be showed in the column STATUS with the name Pump_Status. Having assigned *GROUP=PUMPS; NAME=Water Pump;* it will be showed instead in the column PUMPS with the desidered name Water Pump.

GROUP=PUMPS

Name=Water Pump

The keywords to use are the following:

- GROUP=XXXX; →changes the group assignment or creates a new group XXXX. The GROUP field should not contain spaces.
- NAME=YYYY; → changes the default name (proprerty *Name*) into the desidered name.
- DESC=YYYY; → changes default description (property *Description*) into the desidered description.

3 BASIC ELEMENTS

3.1 Data types

The data types allowed in the development environment may be divided into two categories:

 ➢ simple
 ➢ structured.

The former are composed of a single value that can be directly used in the algorithms for processing purposes, while the latter are composed of several fields because they contain several pieces of information. For example, data type CJ_ANALOG contains two pieces of information: sensor value and sensor error code.

CJ_VOID, a newly created data type introduced by the UNI-PRO development system, belongs to the simple data category.

3.1.1 Simple data types

Simple data may, in turn, be divided into two logical categories: the first category is composed of all C-language data types, while the second one includes new unstructured data types created by the UNI-PRO development environment.

The summarizing table below shows the data types that belong to the first category, and gives the following information about each one of them:

 ➢ Minus sign   : whether or not it can represent negative numbers
 ➢ Repr.   : number of bits actually used by an object of this type
 ➢ Min   : minimum value it can take
 ➢ Max   : maximum value it can take
 ➢ Corr. ANSI C : corresponding ANSI C data type

Data type	Minus sign	Repr.	Min.	Max.	Corr. ANSI C
CJ_BIT	NO	1 bit	0	1	
CJ_S_BYTE	YES	8 bits	-128	127	Signed char
CJ_BYTE CJ_CHAR	NO	8 bits	0	255	Unsigned char
CJ_SHORT	YES	16 bits	–32768	32767	Signed short
CJ_WORD	NO	16 bits	0	65535	Unsigned short
CJ_LONG	YES	32 bits	-2147483648	2147483647	Signed long
CJ_DWORD	NO	32 bits	0	4294967295	Unsigned long

All operations allowed by the ANSI C language may be performed on the above data types.

The new data types (CJ_VOID, CJ_LED, CJ_BUZZ, CJ_DATE, CJ_TIME, CJ_DATETIME) created by the UNI-PRO environment need to be discussed individually.

CJ_VOID

The CJ_VOID data type is an innovative concept created by the UNI-PRO development environment that allows dramatic development time savings while at the same time providing a high level of flexibility.

Thanks to this new concept, you can define the data type of a generic object (such as a Var or the inputs of an algorithm) just by joining it to an object whose data type has already been set.

For example, if you add a variable to a project, it will be set to CJ_VOID by default. By joining this variable to a digital input (defined as CJ_BIT by default), the data type of the variable will be automatically set to CJ_BIT.

CJ_LED and CJ_BUZZ

Data types CJ_LED and CJ_BUZZ are very similar and that's why they are discussed together. They represent the possible values that a LED and a Buzzer object may respectively take.

These data types may take values ranging from 0 to 3, which correspond to the following states:
- 0 : OFF
- 1 : Continuously on
- 2 : Slow flash/beeps
- 3 : Fast flash/beeps

CJ_DATE

The CJ_DATE data type has been implemented to perform operations on dates; it represents the number of seconds elapsed since midnight, 1 January, 2000, and it can represent dates up to the year 2068. Using this data type may prove useful, for instance, if you want to control operations based on certain fixed dates.

If you decide to use this data type in an algorithm, you might find it easier to use the CJ_DATE_STRUCT structure.

CJ_TIME

The CJ_TIME data type has been implemented to perform operations on times; for instance, it can prove particularly useful to manage different controller time ranges and in multiple other cases. It represents the number of seconds elapsed since the beginning of the day (00.00) and it can be easily converted into the structure CJ_TIME_STRUCT through the conversion function especially provided.

CJ_DATETIME

The CJ_DATETIME data type has been implemented for all those cases where you may need to process times and dates together; it represents the number of seconds elapsed since midnight, 1 January, 2000, and it can represent dates up to the year 2068.

This data type may be used directly in the algorithms; otherwise, you can convert it into the structure CJ_DATETIME_STRUCT through the library functions (see the section on CJ_DATETIME_STRUCT), which should make your work easier.

3.1.2 Structured data types

Structured data types capable of holding multiple pieces of information have been implemented in the UNI-PRO development environment. They are actually C structures composed of n fields, that may be accessed using the usual C syntax:

structure.fieldname

Below we will discuss structured data types and their meaning.

CJ_ANALOG

The CJ_ANALOG data type represents the state of an analog input. The structure is composed of two fields:

  ➢ **Error** a byte type representing an error code. If this field equals zero, there are no sensor errors; otherwise, it will take the following values:
  - 1 : short-circuited sensor.
  - 2 : open or missing sensor.
  ➢ **Value**, a short data type representing the value read by the sensor

Some defines of project can be associated to the value field (using the algorithms) in order to manipulate the type of data CJ_ANALOG:

CJ_AI_MIN_RESERVED and CJ_AI_MAX_RESERVED: respectively minimum and maximum value above which the field value of the sensor is in error. If the value field of the sensor isn't included inside these limits, the sensor is considered in error status(sensor field status is positive). Having this information can reset the error field using the value field so without necessarily having to use a CJ_ANALOG structure. Useful when is necessary to exchange the sensors value in a CAN communication.

CJ_AI_DISABLED: value who define the sensor disabling status. Setting Value at this value, the sensor results to be disabled and the EIML pages visualization becomes dots "…".It is useful if want to condition the sensor action to a possible enabling parameter.

CJ_CMD

The CJ_CMD data type is a structure associated with the arrival of a command. It is composed of the following fields:

  ➢ **Valid** a boolean data type representing that a command has been notified. If this property is TRUE, it means that the command has been received and the desired action can therefore be carried out; otherwise, no command has been received.
  ➢ **Node** a byte type indicating the logical node of the controller that sent the command.
  ➢ **Param** a short type representing the parameter of the command.

CJ_BTN

The CJ_BTN data type is a structure associated with an action performed on a button: pressing, holding down or releasing.
It is composed of the following fields:

> ➤ **Valid** a boolean data type representing that an action has been performed on the button (i.e. it has been pressed, released, or held down). If it is TRUE, it means that the action indicated in the Btn object has been notified, otherwise the action has not taken place.
> ➤ **Node** a byte data type indicating the logical node where the action on the button has been verified
> ➤ **Param** a short data type indicating the number of seconds the button has been held down.

CJ_DATE_STRUCT

The CJ_DATE_STRUCT data type can prove very useful to perform operations on dates. Starting from the unstructured data type CJ_DATE, you can fill in the CJ_DATE_STRUCT structure with the help of the conversion function provided to that end.
It is composed of the following fields:

> ➤ **Day** a byte data type indicating the day [1..31]
> ➤ **Month** a byte data type indicating the month [1= January, 2=February, … 12=December]
> ➤ **Year** a byte data type indicating the last two digits of the year, starting from the year 2000. For example, if this field takes the value 12, it represents the year 2012.

This structure is usually filled in by the conversion function DateToStruct(), that has the following C syntax:

<div align="center">

CJ_DATE_STRUCT DateToStruct(CJ_DATE Value);

</div>

where *Value* parameter is a date encoded with second starting from midnight of the year 2000.
To convert the structure back to CJ_DATE type use the StructToDate() function, that has the following C syntax:

CJ_DATE StructToDate(CJ_DATE_STRUCT date);

CJ_TIME_STRUCT

The CJ_TIME_STRUCT data type can prove very useful to perform operations on times, for example to manage different time ranges. Starting from the unstructured data type CJ_TIME, you can fill in the CJ_TIME_STRUCT structure with the help of the conversion function provided to that end.
It is composed of the following fields:

> ➤ **Sec** a byte data type indicating the seconds [0..59]
> ➤ **Min** a byte data type indicating the minutes [0..59]
> ➤ **Hour** a byte data type indicating the hours [0..23]

This structure is usually filled in by the conversion function TimeToStruct(), that has the following C syntax:

<div align="center">

CJ_TIME_STRUCT TimeToStruct(CJ_TIME Value);

</div>

where *Value* parameter is a time encoded with second starting from the midnight of the same day.
To convert the structure back to CJ_TIME type use the StructToTime() function, that has the following C syntax:

CJ_TIME StructToTime(CJ_TIME_STRUCT time);

CJ_DATE_TIME_STRUCT

The CJ_DATETIME_STRUCT data type is used to convert the CJ_DATETIME data type (which represents a date/time coded in seconds) into a more user-friendly format.

This structure is usually filled in by the conversion function DateTimeToStruct() that has the following C syntax:

CJ_DATETIME_STRUCT DateTimeToStruct(CJ_DATETIME Value);

where *Value* parameter is a date-time encoded with second starting from midnight of the year 2000. Here is a brief description of its fields:

➢ **Sec** a byte data type indicating the seconds [0..59]
➢ **Min** a byte data type indicating the minutes [0..59]
➢ **Hour** a byte data type indicating the hours [0..23]
➢ **Day** a byte data type indicating the day of the month [1..31]
➢ **Weekday** a byte data type indicating the day of the week [0=Sunday, 1=Monday, … 6=Saturday]
➢ **Month** a byte data type indicating the month [1= January, 2=February, … 12=December]
➢ **Year** a byte data type indicating the last two digits of the year, starting from the year 2000. For example, if this field takes the value 12, it represents the year 2012.

To convert the structure back to a CJ_DATETIME data type, use the StructToDateTime function that has the following C syntax:

CJ_DATETIME StructToDateTime(CJ_DATETIME_STRUCT rtc);

3.2 Type compatibility table

When you attempt to join two entities, the system will check whether or not the data types of the entities you are about to join are compatible. The compatibility table below is used to ensure that the output's data type is compatible with the input's data type. To read the table, look for the data type of the output from which you started the link on the left side (text highlighted in red), and the data type of the input where you want to end the link in the upper part (text highlighted in green). If there is an X where both rows meet, it means that the two data types are compatible and they can be successfully linked; otherwise, an error message will be displayed and they won't be linked.

The above does not apply to the CJ_VOID data type: as you can see, this type is compatible with all other types because when you make a join in which the data type of the input or the output is CJ_VOID, the development environment will automatically convert it to match the data type to which it is being linked.

	CJ_VOID	CJ_BIT	CJ_BYTE CJ_CHAR	CJ_S_BYTE	CJ_LED	CJ_BUZZ	CJ_WORD	CJ_SHORT	CJ_ANALOG	CJ_DWORD	CJ_LONG	CJ_DATE	CJ_TIME	CJ_DATETIME	CJ_CMD	CJ_BTN
CJ_VOID	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
CJ_BIT	X	X	X	X	X	X	X	X		X	X					
CJ_BYTE CJ_CHAR	X		X				X	X		X	X					
CJ_S_BYTE	X			X			X	X		X	X					
CJ_LED	X		X	X	X		X	X		X	X					
CJ_BUZZ	X					X										
CJ_WORD	X						X			X	X					
CJ_SHORT	X							X		X	X					
CJ_ANALOG	X								X							
CJ_DWORD	X									X						
CJ_LONG	X										X	X	X	X		
CJ_DATE	X										X	X	X	X		
CJ_TIME	X										X	X	X	X		
CJ_DATETIME	X										X	X	X	X		
CJ_CMD	X														X	
CJ_BTN	X															X

This data type compatibility check is carried out only at the time when the link is made. Therefore, if you make any changes to the type of inputs or outputs after that, they won't be noticed and could consequently lead to inconsistencies in the operation of the program.

Two entity with *array* property greater than one are compatible one another only for the two following conditions:

3. if both the entities has the same array size (same *array* property value)
4. if the entities are of the same data type, with the exception :
     a.   when start entity data type is CJ_BIT and linked entity data type is CJ_S_BYTE
     b.   when start entity data type is CJ_BIT and linked entity data type is CJ_BYTE

In all other cases, entities are incompatible.

3.3 Firmware library functions to be used in the algorithms

3.3.1 CJ_DATETIME StructToDateTime(CJ_DATETIME_STRUCT rtc)

This function converts a data structure CJ_DATETIME_STRUCT rtc to a number that is encoded with second starting from midnight of the year 2000, that is a CJ_DATETIME type.

3.3.2 CJ_DATE StructToDate (CJ_DATE_STRUCT date)

This function converts a data structure CJ_DATE_STRUCT date to a number that is encoded with second starting from midnight of the year 2000, that is a CJ_DATE type.

3.3.3 CJ_TIME StructToTime (CJ_TIME_STRUCT time)

This function converts a data structure CJ_TIME_STRUCT time to a number that is encoded with second starting from midnight of the year 2000, that is a CJ_TIME type.

3.3.4 CJ_DATETIME_STRUCT DateTimeToStruct(CJ_DATETIME Value)

This function returns a data structure CJ_DATETIME_STRUCT created from the value of the *Value* parameters of CJ_DATETIME type used as input.

3.3.5 CJ_DATE_STRUCT DateToStruct (CJ_DATE Value)

This function returns a data structure CJ_DATE_STRUCT created from the value of the *Value* parameters of CJ_DATE type used as input.

3.3.6 CJ_TIME_STRUCT TimeToStruct (CJ_TIME Value)

This function returns a data structure CJ_TIME_STRUCT created from the value of the *Value* parameters of CJ_TIME type used as input.

3.3.7 CJ_BYTE CJ_GetSeconds(CJ_DATETIME dt)

This function returns the number of the seconds [0..59] contained in the parameter *dt* of CJ_DATETIME type.

3.3.8 CJ_BYTE CJ_GetMinutes(CJ_DATETIME dt)

This function returns the number of the minutes [0..59] contained in the parameter *dt* of CJ_DATETIME type.

3.3.9 CJ_BYTE CJ_GetHours(CJ_DATETIME dt)

This function returns the number of the hours [0..23] contained in the parameter *dt* of CJ_DATETIME type.

3.3.10 CJ_BYTE CJ_GetDay(CJ_DATETIME dt)

This function returns the number of the day [1..31] contained in the parameter *dt* of CJ_DATETIME type.

3.3.11 CJ_BYTE CJ_GetWeekDay(CJ_DATETIME dt)

This function returns the weekday [0 = Sunday, 1.., 6 = Saturday] contained in the parameter *dt* of CJ_DATETIME type.

3.3.12 CJ_BYTE CJ_GetMonth(CJ_DATETIME dt)

This function returns the number of the month [1..12] contained in the parameter *dt* of CJ_DATETIME type.

3.3.13 CJ_BYTE CJ_GetYear(CJ_DATETIME dt)

This function returns the year [00..68] contained in the parameter *dt* of CJ_DATETIME type.

3.3.14 CJ_TIME CJ_GetTime(CJ_DATETIME dt)

This function returns the time CJ_TIME derived to *dt* parameters, that is the number of seconds starting from midnight of the same day.

3.3.15 CJ_DATE CJ_GetDate(CJ_DATETIME dt)

This function returns the date CJ_DATE derived to *dt* parameters, that is the number of seconds starting from midnight of the year 2000 until the midnight of the same day.

3.3.16 CJ_BIT CJ_GetSecondTic (void) e CJ_BIT CJ_GetMinuteTic (void)

This two functions are managed under the system interrupt. The functions have the same rule of the corrispondent TIMER entity, but they don't use the controller memory because they are managed at firmware level and therefore they can be directly used in the algorithm.
The function *CJ_BIT CJ_GetSecondTic (void)* returns the logic value '1' for each elapsed second.
The function *CJ_BIT CJ_GetMinuteTic (void)* returns the logic value '1' for each elapsed minute.
It's advisable to use this function when a great number of TIMER are required in the project to not weight the application functionality.

3.3.17 CJ_BIT CJ_IsFirstMain(void)

Is one for the entire first loop of the main cycle, of applicative.

3.3.18 CJ_WORD CJ_SemaphoreRead(CJ_WORD i)

Reads and returns the values of the n-th semaphore. The available semaphors move from 0 to 9.

3.3.19 CJ_BYTE CJ_SemaphoreWrite(CJ_WORD i, CJ_WORD value)

Writes in the in the n-th semaphore the value *value*. The writing of the semaphore is allowed on condition that one of these two conditions is true:
1. The semaphore is free, or it contains value 0.
2. The value *value* to write is the value 0.
It returns 0 = Error code (index out of limits), 1 = Ok (Writing executed), 2 = Semaphore busy (Writing not allowed).

3.3.20 CJ_BIT CJ_FlagWrite (CJ_WORD i, CJ_BIT val) - CJ_BIT CJ_FlagRead (CJ_WORD i)

This function avails to manage of the *Semaphores* in the algorithm.
Supposing to manage a shared resource between anymore entities, each entity has to know the other entities status to correctly uses the shared resource. About UNI-PRO, this peculiarity involves to use several links between algorithms, or each algorithm has to have the status of the each other algorithms that would use the shared resources as input. This solution has the disadvantage to consume a lot of memory in the controller.
To solve this problem, it is proposed a typical solution of the concurrent computer programming: to the use some entities called *Semaphores*. A *Semaphore* is a structure able to manage the access to some shared resources to control and to assign it in the correct mode. Two functions to realizes this data management are:
➢ CJ_BIT CJ_FlagWrite (word i, byte val)
➢ CJ_BIT CJ_FlagRead (word i)
The function *CJ_BIT CJ_FlagWrite (word i, byte val)* avails to set the status of the i-th semaphore.
If *val=0* the semaphore is free therefore it's possible set its status as busy and then use the resources.

If *val=1* the semaphore is busy, therefore it isn't possible to use the resources and it's required to wait.

For example following function call:

CJ_FlagWrite (10, 1)

set the status of tenth semaphore as busy, therefore the resources managed from this semaphore will be accessible only by the entity that have set the status.

A busy semaphore can is freed only from the same entity that have set its status.

Function call:

CJ_FlagWrite (10, 0)

set the status of tenth semaphore as free, therefore the resources is free and usable.

The status of a semaphore is controllable using the follow function:

CJ_BIT CJ_FlagRead (CJ_WORD i)

that avails to ask the status of the i-th semaphore. It returns '1' if semaphore is busy, otherwise it returns '0'. If the semaphore is busy isn't possible to use the controlled resources until the resources will be freed.

3.3.21 CJ_SHORT CJ_WriteVarExpo(CJ_WORD add, CJ_LONG value)

This function allows to directly write from algorithm the *"value"* value of an exported variable at the *"add"* address on Modbus protocol. Thus the variable is modifiable from all project algorithm. This feature is analogous at the *global variable* concept used in computer programming.

To correctly use the variable is required that it's exported on Modbus protocol using *Export Entities* functionality activable from *Tools/Export Entities* menu of the programming environment.

3.3.22 CJ_LONG CJ_ReadVarExpo(CJ_WORD add)

This function allows to read the value of the exported variable at the "add" address on Modbus protocol. The function is the symmetric of *CJ_WriteVarExpo(word add, long value)*.

To correctly use the variable is required that it's exported on Modbus protocol using *Export Entities* functionality activable from *Tools/Export Entities* menu of the programming environment.

3.3.23 CJ_BYTE CJ_ModbusAskQueue(void)

Returns the free items number of the Modbus queue

3.3.24 CJ_BYTE CJ_SendCommand (CJ_BYTE channel, CJ_BYTE node, CJ_BYTE command, CJ_SHORT par1)

Allows to send a command from inside of an algorithm;

Returns 0= command sent, 1= full queue

Channel: 0= CAN local, 1= Can wide (if present)

Node: logic node of CAN channel

Command: command index

Par1: 16 bit parameter associated to the command.

3.3.25 CJ_WORD CJ_MaxMainTime(void)

Returns the maximum time of cycle of applicative main, showed in milliseconds.

3.3.26 CJ_WORD CJ_MinMainTime(void)

Returns the minimum time of cycle of applicative main, showed in milliseconds.

3.3.27 CJ_WORD CJ_RunMainTime(void)

Returns the current time of cycle of applicative main, showed in milliseconds.

3.3.28 CJ_BYTE CJ_MaxInterruptTime(void)

Returns the maximum time of cycle of applicative interrupt, showed in milliseconds.

3.3.29 CJ_BYTE CJ_MinInterruptTime(void)

Returns the minimum time of cycle of applicative interrupt, showed in milliseconds.

3.3.30 CJ_BYTE CJ_RunInterruptTime(void)

Returns the current time of cycle of applicative interrupt, showed in milliseconds.

3.3.31 CJ_BIT CJ_Stack_Error_Read(void)

When returns one, indicate there was a stack overflow of the program.

3.3.32 CJ_BIT CJ_Math_Error_Read(void)

Is one if there was an DivByZero, Overflow, Underflow or NaN error.

3.3.33 CJ_BIT CJ_DivByZero_Error_Read(void)

Is one if there was a division by zero.

3.3.34 CJ_BIT CJ_Overflow_Error_Read(void)

Is one if there was an overflow.

3.3.35 CJ_BIT CJ_Underflow_Error_Read(void)

Is one if there was an underflow.

3.3.36 CJ_BIT CJ_NaN_Error_Read(void)

Is one if there was a NaN (Not a Number) generic error.

3.3.37 void CJ_DivByZero_Error_Write(void)

Allows to set the mathematical error caused by a division by zero.

3.3.38 void CJ_Overflow_Error_Write(void)

Allows to set the mathematical error caused by an overflow.

3.3.39 void CJ_Underflow_Error_Write(void)

Allows to set the mathematical error caused by an underflow.

3.3.40 void CJ_NaN_Error_Write(void)

Allows to set the mathematical error caused by an arithmetic generic error; NaN means Not a Number, Eg. The square root of a negative number.

3.3.41 void CJ_Math_Error_Reset(void)

Reset the mathematical error caused by the mathematical error functions

3.3.42 void CJ_DivByZero_Error_Reset(void)

Reset the mathematical error caused by a division by zero.

3.3.43 void CJ_Overflow_Error_Reset(void)

Reset the mathematical error caused by an overflow.

3.3.44 void CJ_Underflow_Error_Reset(void)

Reset the mathematical error caused by an underflow.

3.3.45 void CJ_NaN_Error_Reset(void)

Reset the mathematical error caused by an exeption caused by a NaN.

3.3.46 CJ_DWORD CJ_Read_DI_PulseCnt (CJ_BYTE in)

Return the number of impulse measured by the digital input *in* (note: in=0, DI=1).

3.3.47 void CJ_Clear_DI_PulseCnt (CJ_BYTE in)

Clear the number of impulse measured by the digital input *in* (note: in=0, DI=1).

3.3.48 CJ_WORD CJ_Read_DI_Frequency (CJ_BYTE in)

Return the frequency [Hertz] of the impulse counter connected to the digital input *in* (note: in=0, DI=1).

3.3.49 void CJ_ModBus_SetProfile(CJ_WORD numProfile, CJ_BIT currentStatus)

Allows you to enable / disable calls to a remote Modbus device configured via the Modbus Master profile inserted in the ModBus Master Network Config list within the UniPro Hardware Expert wizard.
- *numProfile* corresponds to the number of the profile you want to enable / disable, as entered in the list.
- *currentStatus* value to set to 1 to enable or 0 to disable.

3.3.50 CJ_DATETIME RTCLocalRead(void)

It allows to read the value expressed in seconds of the local RTC

3.3.51 CJ_SHORT RTCLocalWrite(CJ_DATETIME val)

Used to write the value expressed in seconds of the local RTC
Return 0 = successful writing, otherwise error

3.3.52 CJ_WORD CJ_GetTimer1ms(void)

Returns a value in milliseconds of a free running counter that resets to 10000

3.3.53 CJ_BIT EIMLinEdit(void)

Return 1 if the built-in controller is currently in edit.

3.3.54 CJ_BYTE USB_PenDriveIsConnect(void)

Returns 1 when the USB pen drive is recognized as inserted.

3.3.55 void CJ_AIxFilterWrite(CJ_BYTE value)

Allows you to change the filter of the analog input "x", with respect to the default value (normally *value* = 10). Applies only to local inputs, for expansions use the CAN Master tool.

3.3.56 void CJ_AIxEnaFilterWrite(CJ_BIT value)

Allows you to enable/disable the filter of the analog input "x".
value=1 enable (default),
value=0 disable

3.4 Control components

In the main screen of the UNI-PRO development environment there is a **toolbar** that includes all the basic objects divided into twol sections:

 ➢ Software
 ➢ Hardware

The basic elements of the development tool are in the Software and Hardware sections. By combining them, you can design projects and create libraries or templates that you can reuse in future projects.

3.4.1 Software

The Software section groups all the objects that allow you to define the machine's "behavior".
The most important element is, no doubt, the algorithm, which allows you to process input data as required with the purpose of returning an output value. This section also discusses Subsheets, Fixs, Pars, Pers, Vars, Timers, Command Ins and Command Outs.

Algorithm
Generic algorithms are the basic elements in a project, because they allow you to define functions using the C language. Each algorithm corresponds to a C function and that's why it can have a variable number of inputs with different data types, but only one output. Each algorithm belongs to a category. This concept is similar to the concept of class in object-programming, and it is useful to optimize resources, because you won't need to duplicate the code of an algorithm if you use it several times.
When you add an algorithm to a project, the environment will immediately prompt you to define its category which will, in turn, automatically set its name. By default, it has a CJ_VOID output and no inputs. You can add the required number of inputs by choosing **Add Input** from the pop-up menu. At the beginning, their data type will be CJ_VOID and, once they are linked, they will be automatically set to the right data type.

Otherwise, you may use the **CodeEditor**, which allows you:

 ➢ to define, or import from a file, the C function that characterizes the algorithm
 ➢ to change its category
 ➢ to define the number and data types of inputs
 ➢ to define the output type

(for more detailed information, please refer to section **Using CodeEditor**).

Global Algorithm

A global algorithm is a special algorithm it is possible to use in the code of the other algorithms; it is as a project function that can be joined in other parts of the code. The advantage coming from the use global algorithms is the fact they can be used from any other algorithm in the project; it means they can be used more times without writing code again, or without adding new ones. To add a global algorithm one can select the flag "Global Algorithm" in the CodeEditor or selecting the same option inserting a new algorithm.

A global algorithm must respect some rules:

- the name of the category must be in "C" format
- a global algorithm must not be a category algorithm; in other words it is not possible to have global which category is already used
- the terminals of the global algorithms must not be joined
- a global algorithm can be transformed in a generic one
- a global algorithm can be transformed in a global one on condition that it respects the previous rules.

all the global algorithms used in the project can be found through the prompt "Find" or the CodeEditor through the functions list (*Ctrl*+Space).

Single Instance Algorithms

Algorithms of this kind are algorithms that can only be used once in the project, for example algorithms that contain *static* statements or elaborations that must be used exclusively.

Subsheet

Subsheet objects allow you to logically group a series of entities, thus dividing the project into functional blocks.

Subsheets may contain all types of entities, including subsheets themselves, which allows you to organize your project into a multi-level structure. To join entities contained in the same subsheet, the usual procedure may be applied. To join internal and external entities, you will need to export the relevant inputs/outputs. Exported inputs/outputs will be displayed in a different color (black by default).

You may either add an empty subsheet to a project or select which entities to group first and then use the *Create Subsheet* command. Subsheets are the basis for creating templates and libraries (see chapter **Creating Libraries**).

Fix

In most projects, it is necessary to define constants; you can do this using the Fixs. They represent a flash memory area where you can define a constant value (thus without affecting the amount of RAM memory taken up by the project) that can be used in your project.

For each project the maximum use number of FIX is 1023.

Fix objects have the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Array	Entity array size. If value is greater than 1 entity is an array else it's a normal entity
Category	It is a read-only property and is set to Fix.
Show Out LIB	If is active, it indicates this entity as "public" when it is inserted in a library.
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
MasterRefresh	Entity refresh time on the master networks
Max	Maximum value that the constant can take. It depends on the selected Type.
Min	Minimum value that the constant can take. It depends on the selected Type.
Name	Unique name that identifies the element inside a project.
Precision	Indicates the number of decimal digits used to interpret the internal value of the element and to display the value in EIML pages.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more detailed information, please refer to option **Defining execution tasks**).
Type	Defines the constant's data type. The default value of this property is CJ_VOID. It may be changed either automatically by joining an input having a data type different from CJ_VOID to the entity's output, or manually by setting this property in the properties window. According to the selected data type, the constant will have different limits and will take up a different amount of Flash memory.
Value	Value taken by the constant. Always make sure that it is consistent with the selected data type, i.e. that it falls within the range set by properties Min and Max.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

Value under the fix (constants)

Under the FIX type entity (blue) a value has been added which assumes the constant.

FIX_Kostant_45

45

Par

Par objects represent the parameters of the application software that allow you to change the machine's behavior during its operation. Parameters are stored in the E2 memory, so that all set values are maintained even after the machine is switched off and back on; through parameters you can save and keep congruent the values in the memory.

For each project the maximum use number of PAR is 2047.

They have the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Array	Entity array size. If value is greater than 1 entity is an array else it's a normal entity
Category	It is a read-only property and is set to Par.
Condvisible	If selected allow to the parameter to be used with conditioned visibility.
Show Out LIB	If is active, it indicates this entity as "public" when it is inserted in a library.
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
MasterRefresh	Entity refresh time on the master networks
Max	Maximum value that the parameter can take. It depends on the selected Type.
Min	Minimum value that the parameter can take. It depends on the selected Type.
Name	Unique name that identifies the element inside a project.
Parmax	Limits the maximum value of the actual parameters at the value of another enitiy existent in the application. If set at <NONE> maximum limit is the value set in the Max property.
Parmin	Limits the minimum value of the actual parameters at the value of another enitiy existent in the application. If set at <NONE> minimum limit is the value set in the Min property.
Order	Execution order applied to calculate this element (see section "Setting the program's execution order").
Precision	Indicates the number of decimal digits used to interpret the internal value of the element and to display the value in EIML pages.
RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more detailed information, please refer to option **Defining execution tasks**).
Type	Defines the parameter's data type. The default value of this property is CJ_VOID. It may be changed either automatically by joining an element having a data type

	different from CJ_VOID to the entity's input/output, or manually by setting this property in the properties window. According to the selected data type, the parameter will have different limits and will take up a different amount of E2 memory.
Value	Initial value taken by the parameter. It should be consistent with the selected data type and therefore fall within the range set by the Min and Max properties (for more detailed information, please refer to option **Force Upload Parameters**).
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

To change a parameter value, do one of the following:
- ➢ join another entity (for instance, an algorithm) to the input pin
- ➢ join the parameter to the EIML pages and enable it for editing
- ➢ use one of the serial communication protocols to start a parameter writing request

Note: if the value of an entity linked to the maximum/minimum limit of a parameter slowed in a page changes, the new limit never will be refresh until the same page will be re-loaded from the application.

Pers

Pers objects represent all the states that are persistent, i.e. all those values that you may want to maintain, even after the machine has been reset (for example, compressor run-time or lighting status). Pers objects store status, working hours, ecc …

For each project the maximum use number of PERS is 1535.

They have the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Array	Entity array size. If value is greater than 1 entity is an array else it's a normal entity
Category	It is a read-only property and is set to Pers.
Condvisible	If selected allow to the persistent to be used with conditioned visibility.
Show Out LIB	If is active, it indicates this entity as "public" when it is inserted in a library.
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
MasterRefresh	Entity refresh time on the master networks
Max	Maximum value that the state can take. It depends on the selected Type.
Min	Minimum value that the parameter can take. It depends on the selected Type.
Name	Unique name that identifies the element inside a project.
Order	Execution order applied to calculate this (see section "Setting the program's execution order").
Precision	Indicates the number of decimal digits used to interpret the internal value of the element and to display the value in EIML pages.

RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more detailed information, please refer to option **Defining execution tasks**).
Type	Defines the parameter's data type. The default value of this property is CJ_VOID. It may be changed either automatically by joining an element having a data type different from CJ_VOID to the entity's input/output, or manually by setting this property in the properties window. According to the selected data type, the state will have different limits and will take up a different amount of E2 memory.
Value	Default value taken by the state. It should be consistent with the selected data type and therefore fall within the range set by the Min and Max properties (for more detailed information, please refer to option **Force Upload Parameters**).
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

To change a state following:

> ➢ join another entity (for instance, an algorithm) to the input pin
> ➢ join the parameter to the EIML pages and enable it for editing
> ➢ use one of the serial communication protocols to start a parameter writing request

Var

Vars represent RAM memory allocations that can be used to process data; since they are stored in the volatile memory, they will be set to their default values every time the machine is reset.
For each project the maximum use number of VAR is 2047.
They have the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Array	Entity array size. If value is greater than 1 entity is an array else it's a normal entity.
Category	It is a read-only property and is set to Var.
Condvisible	If selected allow to the variable to be used with conditioned visibility.
Show Out LIB	If is active, it indicates this entity as "public" when it is inserted in a library.
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
MasterRefresh	Entity refresh time on the master networks
Max	Maximum value that the variable can take. It depends on the selected Type.

Min	Minimum value that the variable can take. It depends on the selected Type.
Name	Unique name that identifies the element inside a project.
Order	Execution order applied to calculate this element (see section "Setting the program's execution order").
Precision	Indicates the number of decimal digits used to interpret the internal value of the element and to display the value in EIML pages.
RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more detailed information, please refer to option **Defining execution tasks**).
Type	Defines the variable's data type. The default value of this property is CJ_VOID. It may be changed either automatically by joining an element having a data type different from CJ_VOID to the entity's input/output, or manually by setting this property in the properties window. According to the selected data type, the variable will have different limits and will take up a different amount of RAM memory.
Value	Default value taken by the variable at controller power-up. Always make sure that it is consistent with the selected data type, i.e. that it falls within the range set by properties Min and Max.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

Timer

Timers are counters that are automatically increased/decreased by the system, under a time base. They have inputs on which we can act to reset and load the value and to enable or cancel the count, and one output that gives back the value of the counter. The only input always present is the reset one, while the enable, clear and reload inputs can be activated trough the context menu:

If the **Enable** input equals zero, timer count will stop, while if it equals 1, timer count will be enabled. The **Reset** input senses the transition from 0 to 1 to reload timer count. The **Clear** input realizes the transition from 0 to 1 to delete the current countdown. Connecting a parameter to the **reload** input means loading the timer count with the parameter value, instead of a default value (look at Max property).

You may conveniently set its properties to decide how often this counter should be changed and whether it should be increased or decreased.

Using these objects you can provide compressor short-cycle protection or timed lighting.

For each project the maximum use number of TIMER is 255.

They have the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Category	It is a read-only property and is set to Timer.
Description	Text-type description field where you can type in your own notes.
Direction	Direction towards which the value is changed when the enable input is activated. If this property is set to UP, the counter's value will be increased, under the pre-set time base, starting from zero (when timer is reset) until it reaches the value set in Max. Conversely, if set to DOWN, the value will be decreased starting from the value set in Max until it returns to zero.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Max	Indicates the maximum value the timer can take The meaning of this property depends on the value of the property Direction: if direction is set to UP, it indicates the value that has to be reached by the timer; otherwise, it indicates the starting value after a reset. If the reload input is present, this value will be the one loaded as start or end limit, while Max will represent only a superior control limit.
Name	Unique name that identifies the element inside a project.
Order	Execution order applied to calculate this element (see section "Setting the program's execution order").
RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Timed	Allows defining in which task calculate this element and the other elements linked to its inputs. (For more detailed information, please refer to option **Defining execution tasks**).
Type	Data type is set to CJ_WORD and cannot be changed. Therefore, the timer can take a maximum value equal to the limit of this data type (refer to section **Data types**).

Udm	Unit of time that causes Timer value to change. This property may take the following values:
	   ➢ 100 ms
	   ➢ sec
	   ➢ min
	   ➢ hour
	Depending on the value set here, the timer will be either increased or decreased every 100 milliseconds, one second, one minute, or one hour respectively.

Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can

either use the properties table or graphically resize the entity using your mouse pointer.

Udm and direction properties display under timer entity

Under the TIMER entity a string was added with the value of the *direction and udm* property.

Command In

The UNI-PRO environment allows you to specify generic, user-definable commands that will trigger certain actions. Command In objects indicate the reception of these commands and trigger whatever action may be linked to them. All objects of this type have an output that returns the parameter value of the command received and the node that sent it.

For each project the maximum use number of COMMAND IN is 255.

They have the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Category	It is a read-only property and is set to Command In.
Cmd	Indicates the name that identifies the command whose reception you want to monitor (refer to section **Commands**).
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
Type	Data type is set to CJ_CMD and cannot be changed.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

Command Out

Command Outs are used to send commands that will be interpreted by a Command In located in the same or a different controller.

This object type has two inputs: **param** and **trigger**.

The first input is used to retrieve the parameter when sending the command; the second one is a trigger input, which means that it will sense a change in signal edge connected with the entity that triggers the forwarding of a command when the value increases from zero to one.

For each project the maximum use number of COMMAND OUT is 255.

They the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Category	It is a read-only property and is set to Command Out.
Cmd	Indicates the name that identifies the command you want to send (refer to section **Commands**).
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
Order	Execution order applied to calculate this element (see section "Setting the program's execution order").
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more detailed information, please refer to option **Defining execution tasks**).
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

Note

It is possible adding in the project sheets some text Note to add comments useful in the drafting of the application.

The entity Note has the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Category	It is a read-only property and is set to Note.
Description	Text-type field where you can type the notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

Double-click the entity opens an editor for entering the text of the note.

The objects Notes are only visual and have no effect on the compilation of the project.

The background color and the font of the Notes can be changed in the Settings window of UNI-PRO environment.

3.4.2 Hardware

The hardware section includes all those entities that represent a physical I/O. The number and type of hardware entities needed in a project will depend on the type of controller and the number of connected expansions. Therefore, you can proceed in two different ways: add all the required entities to the project and then choose the most suitable hardware accordingly, or choose the hardware on which you have decided to develop your project first, and then add the required entities.

This category includes:

- ➢ Digital Inputs
- ➢ Digital Outputs
- ➢ Analog Inputs
- ➢ Analog Outputs
- ➢ Buttons
- ➢ LEDs
- ➢ Buzzers
- ➢ Clocks

Each entity in this section has to be joined to a physical resource (I/O, rtc, LED, etc.) by means of the **Join Tools**.

To each of these categories the maximum use number of entities for each project is 255.

Digital Input

Digital Input objects represent the digital inputs on the controller or the expansions, if any, and have the following properties:

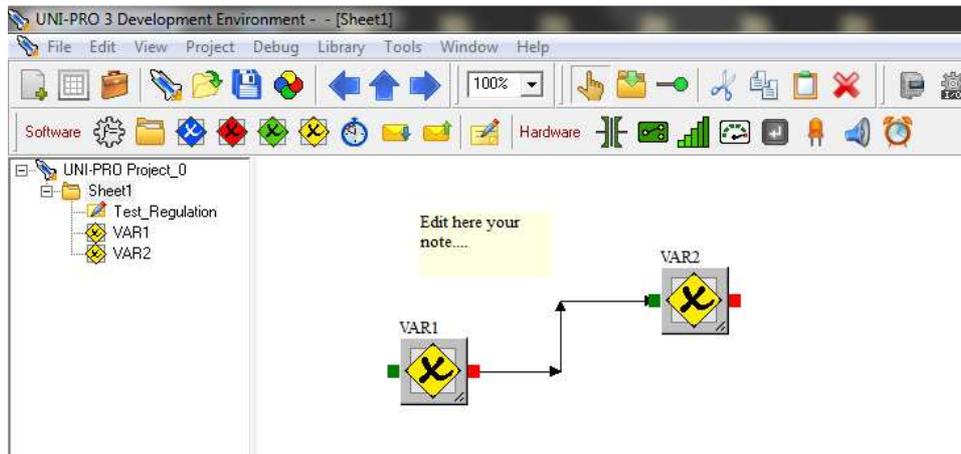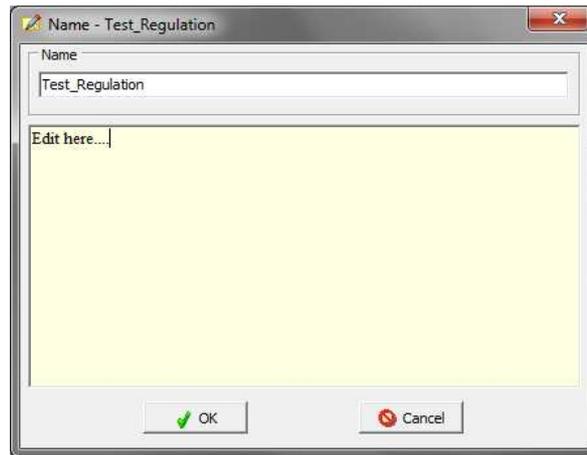|	
Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Category	Set to DigitalIn and cannot be changed.
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Type	Indicates data type, which for Digital Inputs is CJ_BIT and cannot be changed.

Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

To add a Digital In to a project, select the Hardware palette and click the Digital In icon; then click on the sheet where you wish to add the entity. To display its properties, double-click on Digital In (or else right-click and choose **Properties** from the menu).

Digital Inputs allow you to monitor the value taken by the physical digital input and use it as input for an algorithm, to perform more complex operations.

To find out how to join a Digital In to the corresponding physical digital input, please refer to the **Join Wizard** section.

Digital Ouput

Digital Out objects represent a digital output on the controller or the expansions, if any, and have the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Category	Set to DigitalOut and cannot be changed.
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Order	Execution order applied to calculate this element (see section "Setting the program's execution order").
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more detailed information, please refer to option **Defining execution tasks**).
Type	Indicates data type, which for Digital Outputs is CJ_BIT and cannot be changed.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

To add a Digital Output to a sheet, proceed as you would to add a Digital Input.

Analog Input

Analog Input objects are the abstract representation of any type of analog terminal. In fact, they can represent a physical input of the following types: NTC, PTC, 0-5 Volt, 0-10 Volt, 0-20 mA, 4-20mA, PT1000, TEMPERATURE, HUMIDITY. This means that this type of element may be used to read a probe or a generic temperature, pressure, or position sensor.

An Analog Input is characterized by a CJ_ANALOG output pin and the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Category	Set to AnalogIn and cannot be changed.
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
Precision	Indicates the number of decimal digits used to interpret the internal value of the element and to display the value in EIML pages.
RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Sensor	For every analog input you use in a project, you need to specify the type of sensor whose value you want to read. Possible choices are: ➢ 0-5 Volt ➢ 0-10 Volt ➢ 0-20 mA ➢ 4-20 mA ➢ NTC ➢ PTC ➢ PT1000 ➢ TEMPERATURE ➢ HUMIDITY ➢ NTC 10K-2 ➢ NTC 10K-3 ➢ RESISTANCE The selected sensor type has to be supported by the hardware.
Type	Indicates data type, which for Analog Inputs is CJ_ANALOG and cannot be changed.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

To read the state of an analog input added to a project, just join its output pin to the input of an Algo or a Var. As discussed earlier, CJ_ANALOG is a structured data type that contains several pieces of information: an Error field that indicates a possible error, and a Value field that shows its value.

Note. The sensor type 0-10 Volt and 0-5 Volt probe not detect the presence of broken or not connected probe, compared with the correct measurement of the value 0.

Analog Output

Analog outputs are the logical representation of a current analog output (0-20 mA) or a voltage analog output (0-10 V). These object types have a CJ_WORD input pin that expresses a percentage that can take values ranging from 0 to 100.00, and has the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Actuator	For each analog output used in a project is necessary to specify the actuator type connected to the output. Following below list: ➢ FAN ➢ 0-10 Volt ➢ 0-20 mA ➢ 4-20 mA ➢ PWM ➢ BELIMO ➢ EEV_U Sensor type has to be supported by the hardware.
Category	Set to AnalogOut and cannot be changed.
Description	Text-type description field where you can type in your own notes.
Frequency	Default value [HERTZ] of the cycle frequency when the actuator is PWM. [Min 10Hz – Max 2000Hz]
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
Order	Execution order applied to calculate this element under examination (see section "Setting the program's execution order").
Precision	Indicates the number of decimal digits used to interpret the internal value of the element and to display the value in EIML pages.
RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).

WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more detailed information, please refer to option **Defining execution tasks**).
Type	Indicates data type, which for AnalogOuts is obviously set to CJ_WORD and cannot be changed, because the only thing that has to be provided to set an analog output is its value.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

Button

Button elements represent an action on a button, because a button can generate the following events:

➢ press
➢ hold down
➢ release

That's why there may be up to three different Button objects referring to the same button, each one of which will capture a different event.
To choose the event you wish to analyze, set the Cmd property.

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Category	It is set to Button and cannot be changed.
Cmd	The Cmd property identifies the type of event you wish to intercept. It can take the following values: ➢ ON_PRESSED ➢ ON_CONTINUE ➢ ON_RELEASE
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
Type	Indicates data type, which for Buttons is CJ_BUZZER and cannot be changed.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

Led

LED objects are the logical representation of the LEDs found on the controller you are programming or on the expansions connected to it. This object type has a CJ_LED input pin and its

value range is 0-3. By conveniently setting this value, you can change the behavior of the selected LED as follows:

➤ 0 : off
➤ 1 : on
➤ 2 : slow frequency flash
➤ 3 : fast frequency flash

(Refer to section **Data types**).

LEDs can be used to notify to the user with information that you want to make always visible, such as a compressor status or a door open condition.

These objects have the following properties:

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Category	Set to LED and cannot be changed.
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
Order	Execution order applied to calculate this element (see section "Setting the program's execution order").
RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more detailed information, please refer to option **Defining execution tasks**).
Type	Indicates data type, which for LEDs is obviously set to CJ_LED and cannot be changed.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

Buzzer

Buzzer objects are extremely useful in all those situations that require an audible alarm to be sounded. For example, it could prove very convenient to enable a buzzer to warn about a potentially hazardous condition or in an emergency situation.

To do so, just add a Buzzer object to your project and join it to an algorithm having a CJ_BUZZ output.

Depending on the buzzer's input value, its state will be set to the following modes:

- ➢ 0 : off
- ➢ 1 : on
- ➢ 2 : slow frequency beep
- ➢ 3 : fast frequency beep

(Refer to section **Data types**).

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.
Category	Set to Buzzer and cannot be changed.
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Order	Execution order applied to calculate this element (see section "Setting the program's execution order").
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more detailed information, please refer to option **Defining execution tasks**).
Type	Indicates data type, which for LEDs is obviously set to CJ_LED and cannot be changed.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

Clock

Clock objects are the logical representation of the Real Time Clocks built in the controllers and expansions.

Through these objects you may get the current date and time, thus permitting all those operations that are somehow linked to the clock. For example, you may wish to program different controller behaviors depending on the time range in which it is operating.
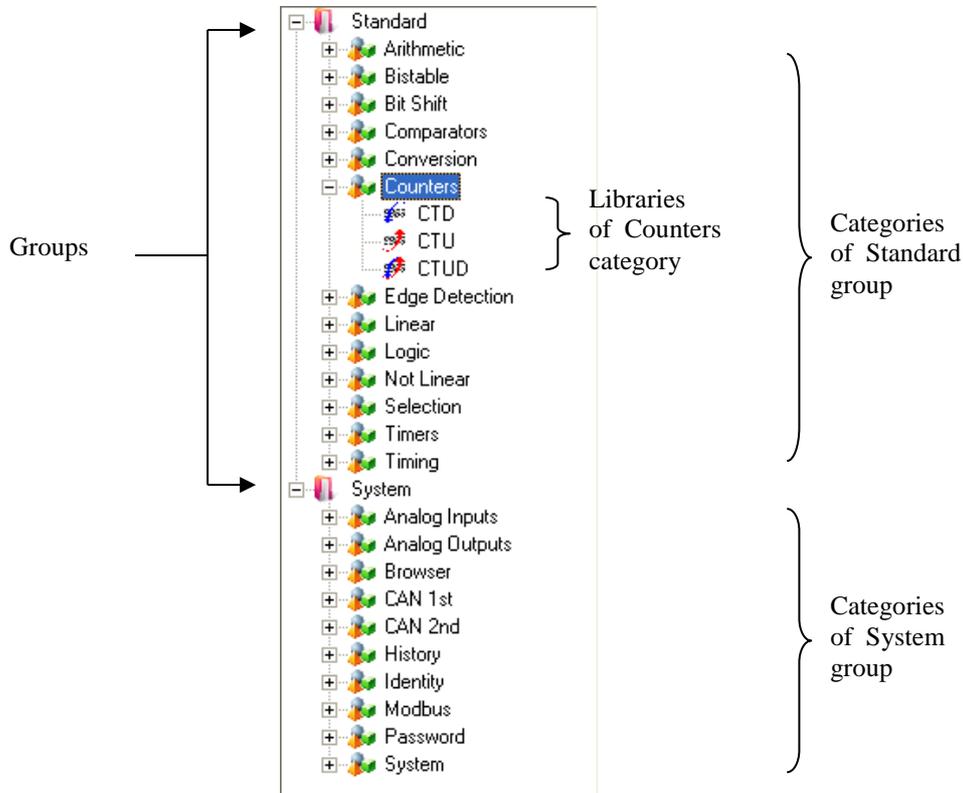
Clock objects return a CJ_DATETIME value that you can easily convert into the CJ_DATETIME_STRUCT structure, to perform more complex operations (please refer to section Data types - CJ_DATETIME_STRUCT).

Top	Sets the position where the element will be drawn, relative to the top edge of the sheet.

Category	It is set to Clock and cannot be changed.
Description	Text-type description field where you can type in your own notes.
Height	Vertical size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
RICS	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software and EVCO parameters management software *Params Manager*.
Mb Access	It indicates if the protection read/access via Modbus is active.
Mb Level	It indicates the level of write protection via Modbus (if level=0, no protection).
WebServer	Field to insert useful information for the representation of the entity in the website of the controller.
Type	Indicates data type, which for Clock objects is obviously set to CJ_DATETIME and cannot be changed.
Width	Horizontal size in pixels of the icon that represents the entity. To change it, you can either use the properties table or graphically resize the entity using your mouse pointer.

3.4.3 Libraries

The tree view libraries collects all the environment and users libraries with a structure that allows to organize the collection at most in two levels. The first level is the group's library, while the second level is the category, in the categories are listed the libraries.

Standard and *System* groups are the two default groups of the UNI-PRO libraries, you can add other groups and extends it with new categories creating new libraries.

3.4.4 Used libraries documentation

From menu *Project->Info Libraries Used* it is possible to create a document in HTML format containing the information of all the used libraries/templates (kind of libraries, entity name, version, etc) .

3.5 Elements in EIML pages

EIML pages are used to design the controller's graphical interface. They allow you to display texts, icons, variables, parameters, internal states, as well as to change values, enable commands, etc.

Through the EIML (Embedded Interface Markup Language) you can create interface screens to meet your specific needs and physically store them in the controller's memory so that they will be loaded on the display "on the fly".

By joining the objects and setting their properties conveniently, you can create user-friendly page navigation.

There are different types of pages available, according to the display for which they are designed: graphic (240x128 or 128x64 pixels), alphanumeric (20x4 or 16x4 characters), and 7-segment.

Choosing *File/New/New EIML Page* from the menu or clicking the [icon] icon in the toolbar will display a list of pages that can be created. Once you have selected one, a window will appear in the work area, where you can compose the page and set the desired display features.

The size of each page in bytes is shown in the lower right corner. This information is useful for you to know whether or not a terminal will be able to display that page.

For each project the maximum use number of EIML Pages is 767.

3.5.1 Toolbar

The EIML toolbar lists the elements that you can add to the page: Texts, Variables, Icons, Strings, Combos, Tables, Lines, Rectangles.

The first position in the toolbar is occupied by an icon representing an arrow, which allows you to select, resize and act on the elements found in the page.

To add a new element (text, variable, icon, etc.), select the desired element in the toolbar and draw the area where you wish to place it.

3.5.2 Embedding an element in an EIML page

To add an element to a page, select it from the toolbar and then move the mouse while holding down the left button to create a rectangle of the desired size. Once we have created it, we can move it to the desired position.

3.5.3 Page properties

EIML pages serve as containers for other EIML objects and have the following properties:

Name	Unique name that identifies the page inside a project.
Id	The Id property is a numeric index ranging from 1 to 240 that allows you to identify the page inside a project.
PreviousPage	If different from <NONE>, indicates the previous page that will be loaded when the LEFT key is pressed.
NextPage	If different from <NONE>, indicates the next page that will be loaded when the RIGHT key is pressed.
TimeoutPage	If different from <NONE>, indicates the page that will be loaded when

	timeout, set in Timeout property, expires. This page is loaded also when ESC key is pressed. If its value is <NONE>, the user interface's default page will be loaded.
Timeout	If different from zero, indicates the number of seconds that have to elapse since the last key was pressed before the previous page (set using the Previous Page property) is automatically recalled.
Language	Indicates the language used to write the page. The controller has a system variable that indicates the current language: every time a page with a specific Id is requested, a page in the current language will be searched first. If the page is multilanguage, it allows showing the page in the language it is set.
Level	Indicates the protection level of the page: the display that requests the page needs to have an authorization level equal to or higher than the level of the page. If it doesn't, there will be a password prompt requesting to enter a password with a higher authorization level.
Encoding	Indicates if encode the page. When variable are used in the page, selecting this property, the flash size of the page is reduced, recovering memory space. The encoded size of the page appears above the original size "Page Size".
Description	Description field (to write possible notes).
Color (only color pages)	Color used for the background of the page
Background (only color pages)	Name of the bitmap background of the page. If <NONE> the background of the page will be the color set in property "Color".
Show background (only color pages)	If active, it shows the background of the page, a design time (it is possible it slows the showing).
Navigation bar (only color pages)	It sets the showing of the navigation bar at the bottom of the page, with these values: - NO: Never visible - ALWAYS: Always vivible - ON TOUCH: Disappearing
Circular Focus	If checked avails the circular navigation of the elements with the Focusindex proprety selected. So, when the cursor is moved on the last element of a page the next element actived by the cursor will be the first of the same page.
Memo Focus Index	If selected allows to storage the Focusindex property during page navigation.
Visibility	If selected allows to use the Conditional Visibility on the EIML pages.
Multilanguage (only for page with multilanguage as 7-Segments)	If selected enables the use of the multilanguage in the page

3.5.4 Multilanguage pages

If the *Multilanguage* property for the EIML page is selected, this page can be created in several languages; thanks to this features one has not to add a new page (having the same *Id*) for each language one wants to use.
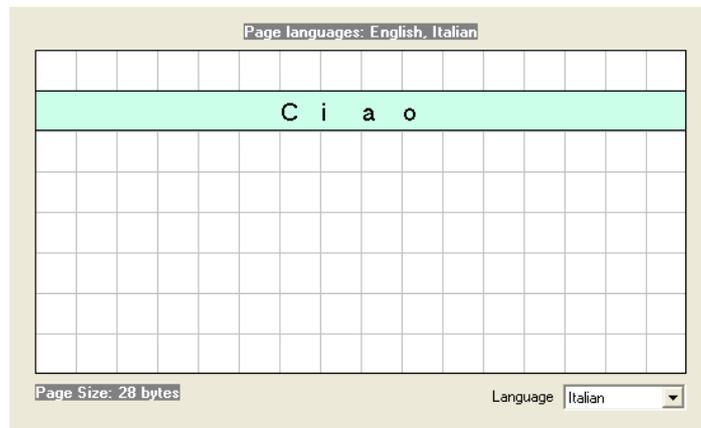
This features also allows being quicker in creating and in maintaining projects with user interfaces in several languages. A multilanguage page is shown as follows:

At the top of the page there is the list of the languages for which a text (at least) has been inserted; at the bottom, in addition to the information about the occupation of the page, there is the possibility to set the page preview for the language one has chosen.

If you modify the language, it will be possible to show the visual effect of the inserted texts, to verify if some text has not been inserted (or it is longer than the allowed space).

Example: if for the text "*Hello*" one has inserted the Italian translation "*Ciao*", modifying the language the following page will be shown as preview:
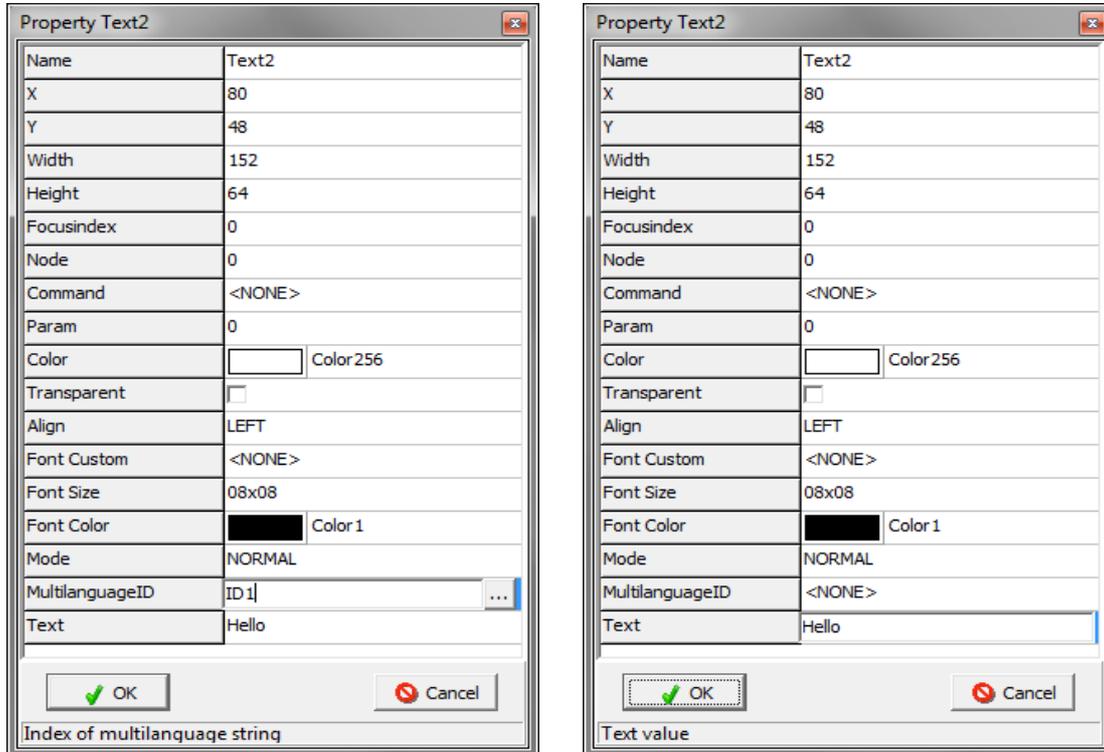

Possible empty texts will be the result of omitted translations or errors.

Note: for the texts in COMBO and CHARTS the preview can be shown only simulating the page with the EIML simulator and changing the languages using the parameters of the simulator.

3.5.5 Multilanguage text elements (with linguistic ID)

Each text element can be multilingual or be text-fixed for each language. To set up a multilingual text it is necessary to associate an ID from a linguistic database. If the *MultilanguageID* property is set to <NONE>, the text assumes for each language the value written in the Text property.

Property Text2			Property Text2	
Name	Text2		Name	Text2
X	80		X	80
Y	48		Y	48
Width	152		Width	152
Height	64		Height	64
Focusindex	0		Focusindex	0
Node	0		Node	0
Command	<NONE>		Command	<NONE>
Param	0		Param	0
Color	Color256		Color	Color256
Transparent	☐		Transparent	☐
Align	LEFT		Align	LEFT
Font Custom	<NONE>		Font Custom	<NONE>
Font Size	08x08		Font Size	08x08
Font Color	Color1		Font Color	Color1
Mode	NORMAL		Mode	NORMAL
MultilanguageID	ID1	...	MultilanguageID	<NONE>
Text	Hello		Text	Hello
✓ OK	✗ Cancel		✓ OK	✗ Cancel
Index of multilanguage string			Text value	

The first image shows the properties of a multilingual text with associated ID1, while the second one shows a text with the fixed text "Hello".

Each ID can be used in multiple text elements.

The display and preview of the pages is identical to the previous case (multilingual pages).
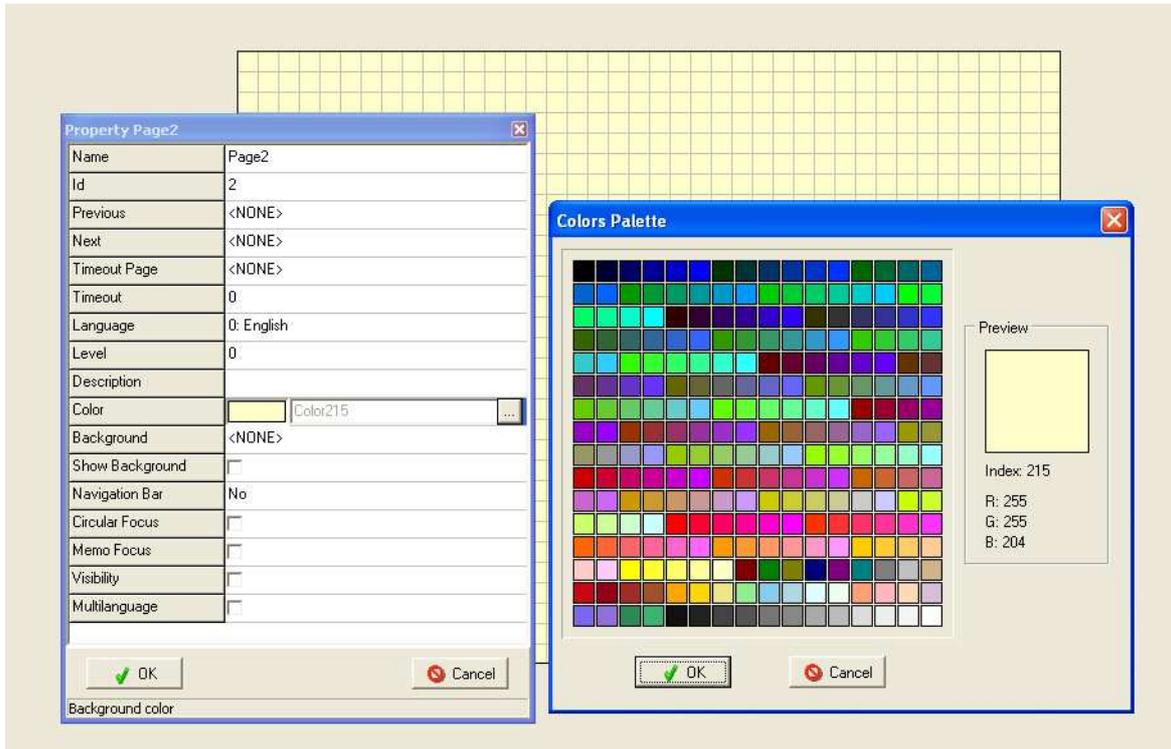
The database to manage the linguistic IDs is in *Tools->StringsTable*, it is however reachable from every page that uses texts with ID, in this case the database will show only a view of the IDs used in that page.

This feature improves the old multilanguage page (3.5.4) because it avoids the duplication of pages in memory, the page is always unique and it is only the texts on the page that are displayed according to the active language.

3.5.6 256 Color pages (EPJ Color)

The color pages can manage elements with color properties, backgrounds and Bitmap images made of 256 colors. The background of the page can be single colour or Bitmap, selectable among a maximum of 10 backgrounds (look at *EIML Backgrounds* tool).
To set the color properties of the EIMS elements it is necessary to use a frame having all the 256 colors of the palette.
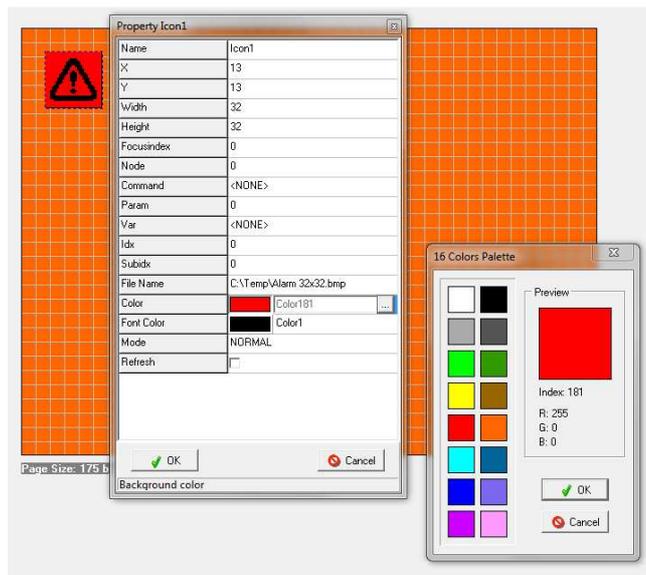
The palette of the UNI-PRO 3 colors is made of 256 not modificable colors.

3.5.7   16 Colors pages (EPJ Graph)

The 16 color pages (320x240 16C) can manage elements with color properties, and Bitmap images made of 2 colors. The background of the page can be single colour only.

To set the color properties of the EIMS elements it is necessary to use a frame having all the 16 colors of the palette.
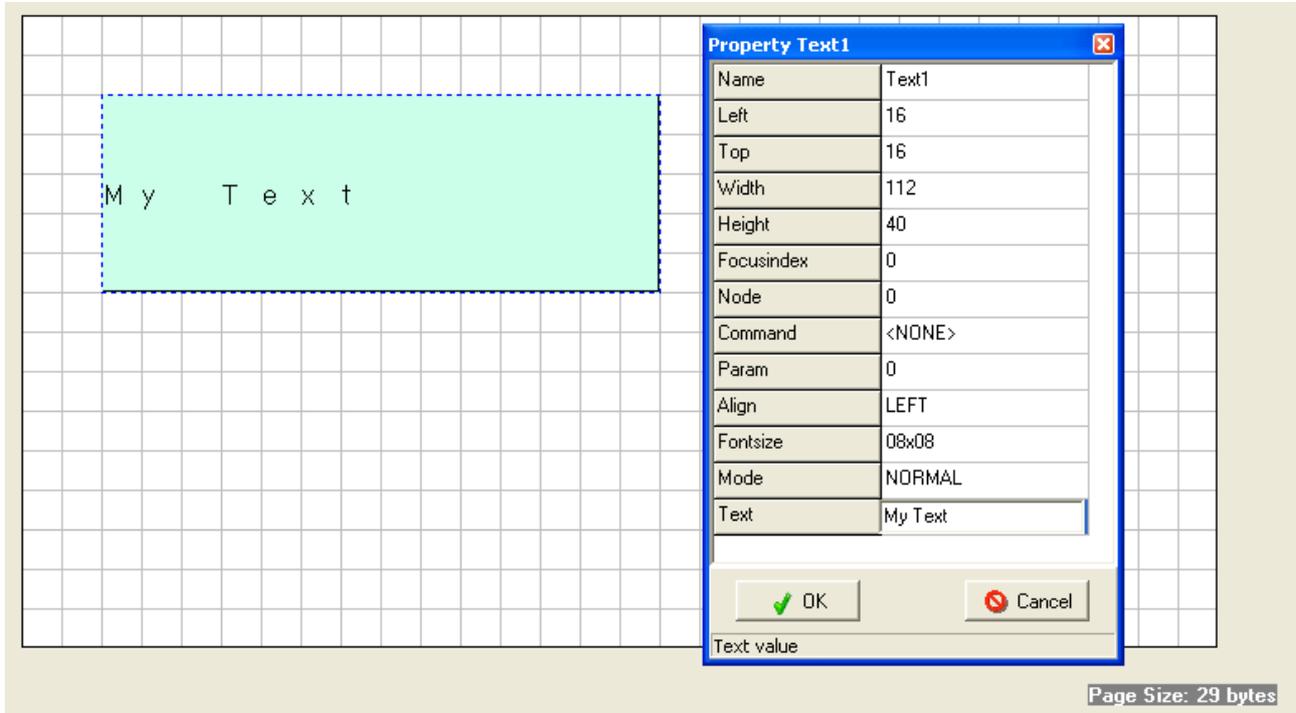

The custom fonts that can be used in these pages are a subset of all the fonts present in Uni-Pro and are not editable because they are preloaded in the viewers (EPJ Graph).
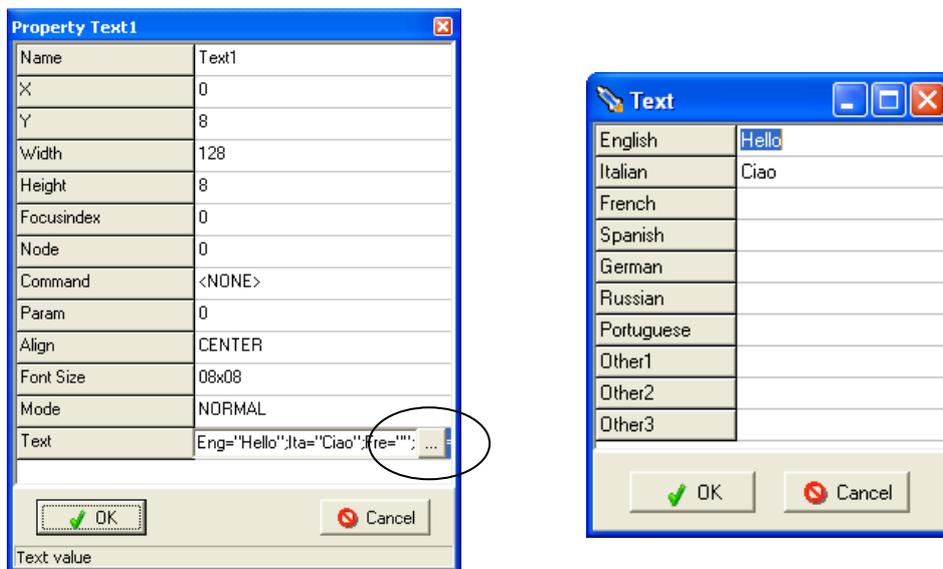
3.5.8 Element properties

Each EIML element has certain properties that allow you to specify its display format and behavior. To display and change the properties of an element, just select it, right-click and choose "*Properties*" from the pop-up menu, or double-click the element itself.
A window listing all the properties of the selected element will be displayed.

If the page is multilanguage the *Text* property of the elements Text , text in COMBO and text in the charts is set and shown in a different way.

To set the text in the desired languages click the button in the *Property* window of the element; on afterwards a further window (where to digit the strings) is shown.

If the elements are multilingual associated with an ID, the texts must be edited using the specific tool in *Tools-> StringsTable*.
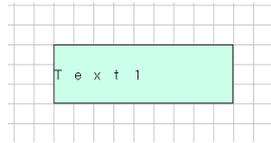
Text

The text element allows you to add descriptions, activate commands, and enable page browsing.
As described under section ***Basic Operations***, the text element has certain properties that allow you to graphically position it in the page ((Left, Top, Width and Height), while the remaining properties describe its "behavior".

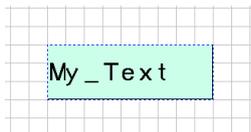|	
Name	Unique name that identifies the element inside a project.
Left	Offset from the left edge of the page, in pixels.
Top	Offset from the top edge of the page, in pixels.
Width	Element width in pixels.
Height	Element height in pixels.
Focusindex	The focusindex property defines the sequence in which the element will receive focus when moving the cursor. The permissible range is 0 to 255. If set to 0, the element will never be selected by the cursor. If set to a different value, defines after how many cursor movements it will be selected (for those who are familiar with Windows programming, it is the same concept as the TabIndex of controls).
Command and Param	These two properties allow a text to send a command to the controller. An action may be attributed to the text by setting a command different from <NONE>. When the cursor is over this element and the "Enter" key is pressed, it will be requested to the controller to execute such command with the parameter set in the param property.
Color (only color pages)	Color used for the text background
Trasparent (only color pages)	Enable the item transparency, the Text drawing is done without the background color.
Align	Permissible values are LEFT, CENTER, and RIGHT which indicate left, center or right alignment respectively.
Font Custom (only color pages)	It specifies wich project custom font use to draw the text. If it is set to <NONE> standard font is used (Fontsize property).
Fontsize	Describes the font size used to write text. For example, if you use an 8x8 font size, a character will be 8 pixels high and 8 pixels wide.
Font color (only color pages)	Color of the text font

Mode	A text may be displayed in four different modes:
	    ➢ NORMAL
	    ➢ NEGATIVE (only black and white pages)
	    ➢ NORMAL-BLINKING
	    ➢ NEGATIVE-BLINKING (only black and white pages)
	If the state is NORMAL, the text will be displayed in black on a white background; conversely, NEGATIVE will display a white text on a black background.
	If you select the "blinking" modes, the text will also blink in the respective modes.

Node	It is the physical node with which to associate any command.
MultilanguageID	ID in the language database. If set to <NONE> the text is the one in the Text property.
Text	Represents the text you wish to display.

To add text to a page, select the **Text** icon in the EIML toolbar and draw the area to be used by the text in the page by holding down the left mouse key and dragging the pointer.

On releasing the key, a text element set to the default values will be drawn.

Now, to enter the text you wish to display, open the properties window (for example by double-clicking the text element) and change the Text property as required. After that, you may go on setting the remaining properties (Fontsize, Align, etc.) and the element will be displayed with a new look:
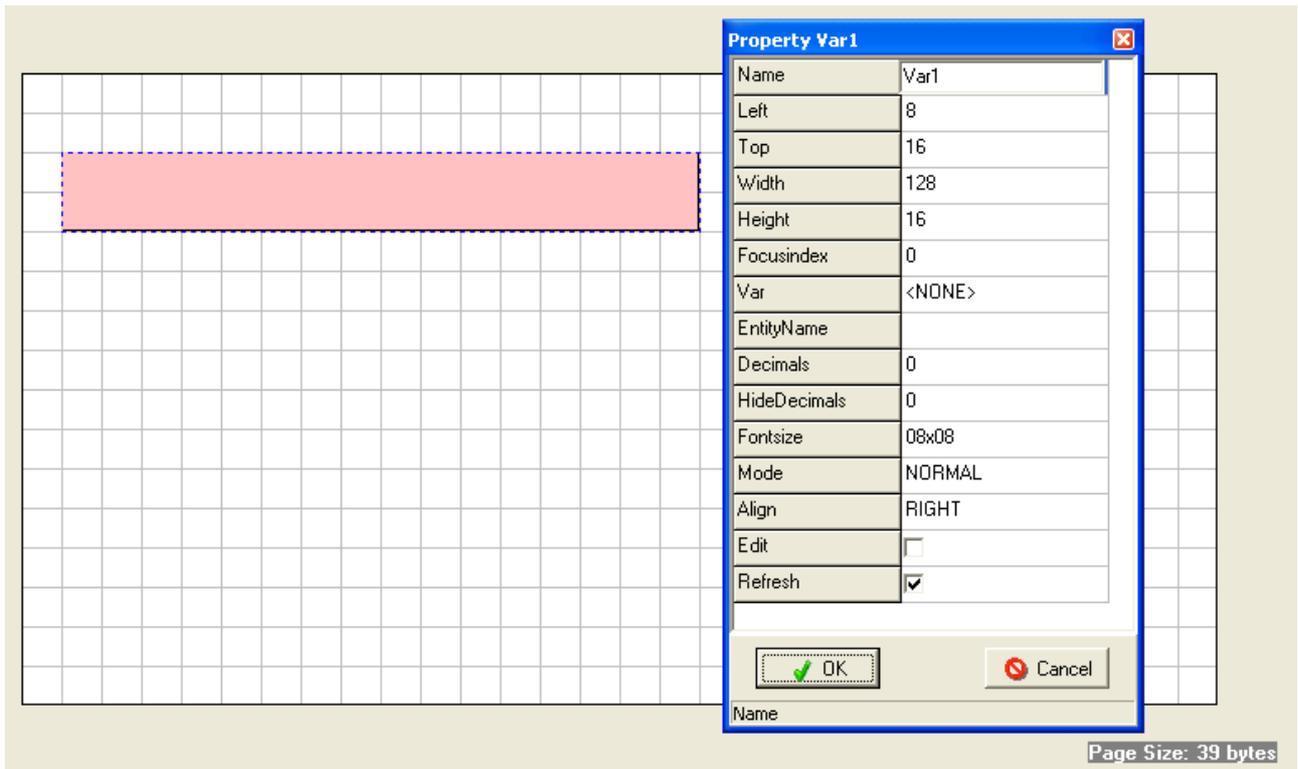
If the page is multilanguage, to modify the *Text* property it is necessary to use the editing window of the multilanguage strings.
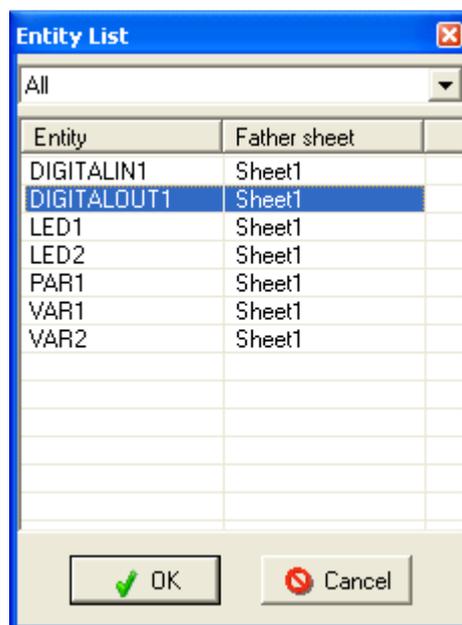

Variable
Variable objects allow you to display and, if required, set the values of internal states, inputs, outputs, etc.
To add a variable to a page, proceed just as you would to add a text.

The variable element may be linked to a project variable: to do this, select the *Var* property (set to
<NONE> by default) and click the button that appears. A window will open, where you can select
one of the project entities whose value you wish to display.

Below we will discuss the properties of variables.

Name	Unique name that identifies the element inside a project.
Left	Offset from the left edge of the page, in pixels.
Top	Offset from the top edge of the page, in pixels.

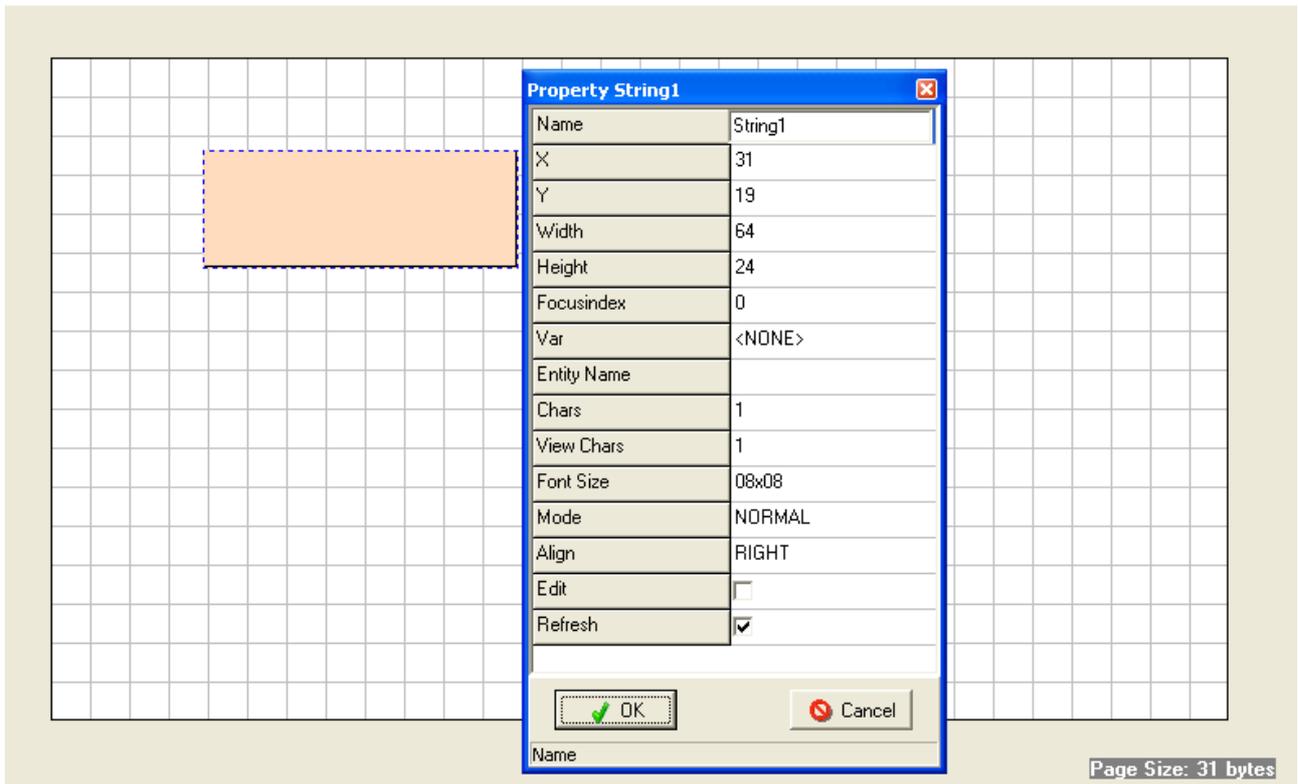|	
Width	Element width in pixels.
Height	Element height in pixels.
Focusindex	The focusindex property defines the sequence in which the element will receive focus when moving the cursor. The permissible range is 0 to 255. If set to 0, the element will never be selected by the cursor. If set to a different value, defines after how many cursor movements it will be selected (for those who are familiar with Windows programming, it is the same concept as the TabIndex of controls).
Var	Using this property you can link a project entity (such as Var, Par, Pers, DI, DO, etc.). Select this property and click the button that will appear: a window will open, where you can choose one of the project entities whose value you wish to display.
Decimals	Number of decimals used to represent the value [0..3]. This is a read-only property and is automatically set when the entity you wish to display is joined (see *Precision* under entity properties).
HideDecimals	Allows you to change the number of decimals that you wish to hide. This property can assume any value between 0 and the value of the Decimals property and has the purpose of hiding some decimal digits.
Color (only color pages)	Color used for the background of the variable.
Trasparent (only color pages)	Enable the item transparency, the variable drawing is done without the background color.
Align	Allows you to align the variable to the left (LEFT), at the center (CENTER) or to the right (RIGHT).
Font Custom (only color pages)	It specifies wich project custom font use to draw the text. If it is set to <NONE> standard font is used (Fontsize property).
Fontsize	Describes the font size used to write text. For example, if you use an 8x8 font size, a character will be 8 pixels high and 8 pixels wide.
Font Color (only color pages)	Color used for the text font.
Mode	A variable may be displayed in four different modes. ➢ NORMAL ➢ NEGATIVE (only black and white pages) ➢ NORMAL-BLINKING ➢ NEGATIVE-BLINKING (only black and white pages) If the state is NORMAL, the variable will be displayed in black on a white background; conversely, NEGATIVE will display a white variable on a black background. If you select the "blinking" modes, the variable will also blink in the respective modes.
Edit	Activate this property to enable variable editing mode and change the associated value (editing a variable requires the focusindex property to be set to a value different from zero).
Refresh	If the Refresh property is enabled, the variable will be continuously requested; otherwise, it will be refreshed slowly so as to avoid overloading communication between controllers.

Strings

String objects allow to show and set the string values, or the CJ_CHAR entities.

The association of the String elements in the EIML pages to the CJ_CHAR entities is possible in the same way of the Variables, using the tool Entity List (look at **Variables**).
Through the tool Entity list it is possible to set the first characters of the string from which the showing of the same will start; to do that it is enough to select from the list the desired position of the elements array CJ_CHAR.
The focusIndex of the element String always makes reference to the position of the first editable characters; the other ones can be selected using the keys of the user interface.
On afterwards the properties of the String elements:

Name	Univocal name identifying the element in the project.
Left	Offset in pixel from the left hand side of the page
Top	Offset in pixel from the top of the page
Width	Width of the element
Height	Height in pixel of the element
Focusindex	Specifying property focusindex it is possible to set with the cursor the order in the selection. The allowed range is from 0 to 255. In particular, setting to this property value 0, the element will never be selected from the cursor (otherwise the value will indicate the number of moves of the cursor after which it will be selected; for the experts in Windows this is the same concept of TabIndex of the controls).
Var	This property allows to join a project entity (for example Var, Par, Pers, Fix) on condition it is a CJ_CHAR one. Selecting the property and clicking the button, a box will open; the box allows selecting one of the project entities which value you want to be displayed.
Chars	It represents the number of characters the string is made of.

ViewChars	It sets the number of characters that will be displayed from the character at the beginning of the string.
Color (only color pages)	Color used for the background of the string.
Trasparent (only color pages)	Enable the item transparency, the string drawing is done without the background color.
Align	It allows aligning the string on the left (LEFT), in the middle (CENTER) or on the right (RIGHT).
Font Custom (only color pages)	It specifies wich project custom font use to draw the text. If it is set to \<NONE\> standard font is used (Fontsize property).
Fontsize	The fontsize sets the size of the font the text will be written. For example, using 8x8, a character will be eight pixel high and eight pixel wide.
Font color (only color pages)	Color of the text font.

Mode	A variable can be displayed in four different modalities:
	
	    ➢ NORMAL
	
	    ➢ NEGATIVE (only black and white pages)
	
	    ➢ NORMAL-BLINKING
	
	    ➢ NEGATIVE-BLINKING (only black and white pages)
	
	Status NORMAL allows showing the variable in black on white background; vice versa status NEGATIVE sets the variable in white on black background.
	Selecting the modalities blinking, the variable will flash using the respective status.

Edit	Activating this property the editing modality of the variable will be enabled, or it is allowed to modify the joined value (to edit a variable it is necessary the focusindex is different from zero).
Refresh	If property refresh is activated, the variable will continuously be refreshed, otherwise it will be refreshed with a lower frequency (which allows to avoid making more heavy the communication among the controllers).

Icons

Icons make EIML pages "better looking", allow you to display a state or activate a command. Below is a detailed description of all icon properties:

Name	Unique name that identifies the element inside a project.
Left	Offset from the left edge of the page, in pixels.
Top	Offset from the top edge of the page, in pixels.
Width	Element width in pixels.
Height	Element height in pixels.
Focusindex	The focusindex property defines the sequence in which the element will receive focus when using the cursor. The permissible range is 0 to 255. If set to 0, the element will never be selected by the cursor. If set to a different value, defines after how many cursor movements it will be

	selected.
Command and Param	These two properties allow a text to send a command to the controller. An action may be attributed to the text by setting a command different from <NONE>. When the cursor is over this element and the "Enter" key is pressed, it will be requested to the controller to execute such command with the parameter set in the param property.
Var(*)	The Var property allows you to join an entity to the icon just as you would do with variables. However, unlike variables, here you are only allowed to join CJ_BIT and CJ_BYTE type entities. Depending on the value read on the entity, the icon will be displayed in the following modes: ➢ 0 = NORMAL ➢ 1 = NEGATIVE ➢ 2 = NORMAL-BLINK ➢ 3 = NEGATIVE-BLINK
Filename	The filename property indicates the source image from which the bitmap was loaded. With this property you can change the icon after it has been added to the project. You can only select icons in Windows bitmap graphic format and multiple of 8x8, that will be loaded and converted to black/white (alternative to the Bitmap property).
Bitmap	Reference to the icon loaded in the EIML Bitmaps tool (alternative to the Filename property, only for 256 colors pages).
Color (only 16 colors pages)	Icon color 1 (only 16 colors pages).
Font color (only 16 colors pages)	Icon color 2 (only 16 colors pages).
Mode(*)	An icon may be displayed in four different modes: ➢ NORMAL ➢ NEGATIVE (only black and white pages) ➢ NORMAL-BLINKING ➢ NEGATIVE-BLINKING (only black and white pages) If the state is NORMAL, the icon will be displayed in black on a white background; conversely, NEGATIVE will display a white icon on a black background. If you select the "blinking" modes, the icon will also blink in the respective modes.
Refresh	The Refresh property allows you to specify how often the value of the entity linked to the icon will be requested. If the icon is not linked to any entity, this property will be ignored.

In the color pages the icons are shown using the color palette UNI-PRO; then if the bitmap to load is not made of the 256 colors of the palette, it will be necessary to wait some seconds the image is converted. The shown icon could be different from the original.

Note(*). When the **Var** property is set, the Mode is aligned with that property, whatever is the **Mode** property.
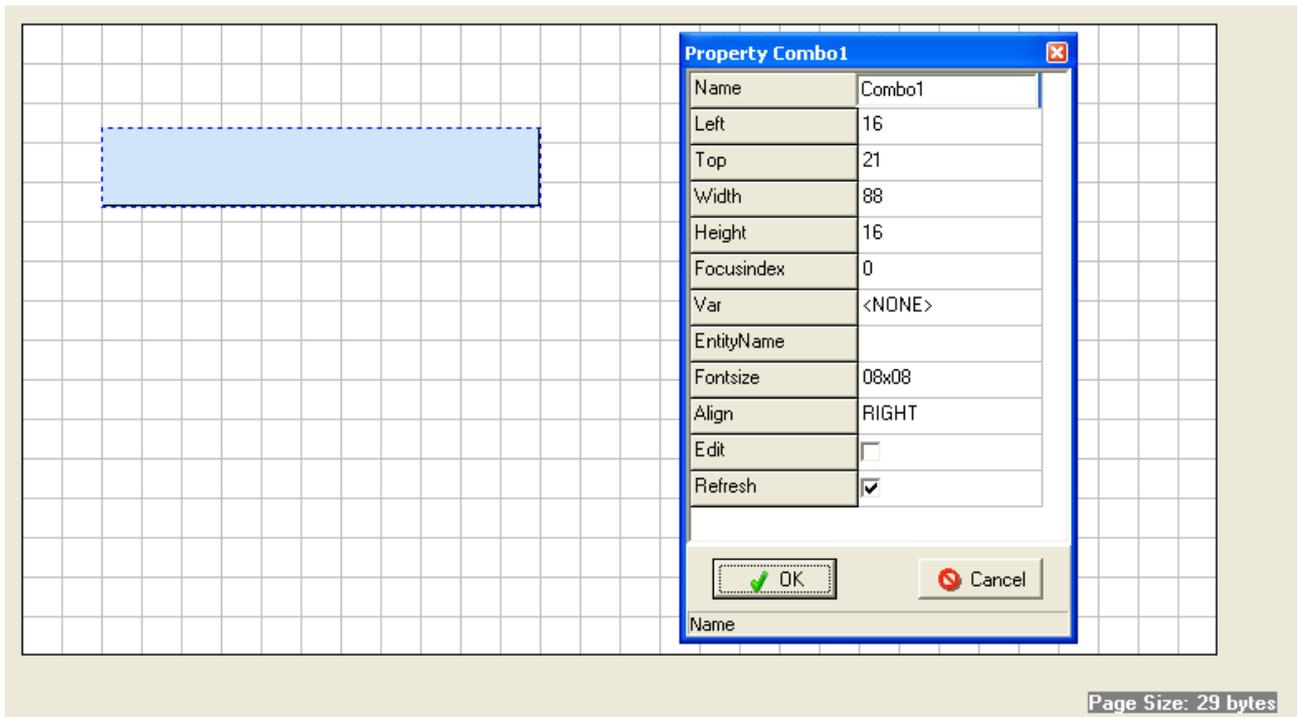
Combo

Combo objects are an innovative way to represent the information contents of an entity. Through these objects you may link a text or an icon to each value taken by an entity, thus providing a flexible and user-friendly way of displaying the entity's content.

Unlike variables, combo objects lack the following properties: **Mode**, **Decimals** and **HideDecimals**. In addition, only CJ_BIT or CJ_BYTE type entities can be joined to them.

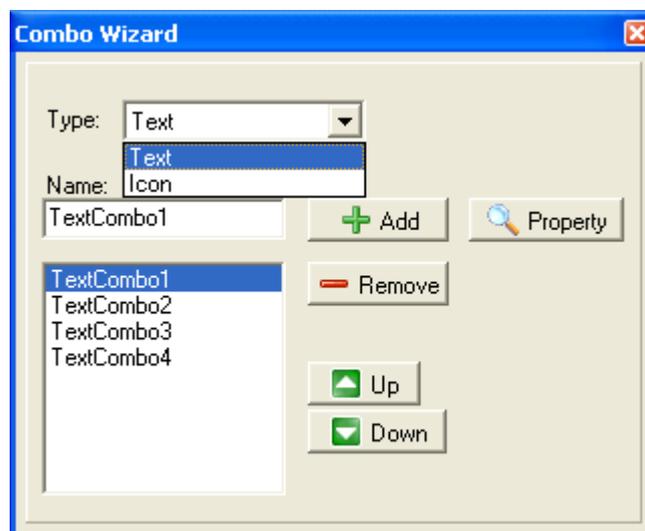|	
Name	Unique name that identifies the element inside a project.
Left	Offset from the left edge of the page, in pixels.
Top	Offset from the top edge of the page, in pixels.
Width	Element width in pixels.
Height	Element height in pixels.
Focusindex	The focusindex property defines the sequence in which the element will receive focus when moving the cursor. The permissible range is 0 to 255. If set to 0, the element will never be selected by the cursor. If set to a different value, defines after how many cursor movements it will be selected (for those who are familiar with Windows programming, it is the same concept as the TabIndex of controls).
Var	Using this property you can link a project entity (such as Var, Par, Pers, DI, DO, etc.). Select this property and click the button that will appear: a window will open, where you can choose one of the project entities to which you wish to link the value.
Font Custom (only color pages)	It specifies wich project custom font use to draw the combo text items. If it is set to <NONE> standard font is used (Fontsize property).
Fontsize	Describes the font size used to write text. For example, if you use an 8x8 font size, a character will be 8 pixels high and 8 pixels wide.
Align	Allows you to align the variable to the left (LEFT), at the center (CENTER) or to the right (RIGHT).
Edit	Activate this property to enable variable editing mode and change the associated value (editing a variable requires the focusindex property to be set to a value different from zero).
Refresh	If the Refresh property is enabled, the variable will be continuously requested; otherwise, it will be refreshed slowly as to avoid overloading communication between controllers.

The distinctive feature of Combo objects is the Combo Wizard, which allows you to display and edit the elements you wish to link to the values of the entities.

We will now add a Combo object to our page and open the properties window:

To join a Combo to an entity, select the *Var* property (set to <NONE> by default), then click the button that will appear to the right and select the desired variable from the Entity List.

Right-click on the Combo element and choose Combo Wizard from the pop-up menu. The following window will appear:

To link texts or icons to the values of the joined entity, select the type of object you want to add from the drop-down menu (Text/Icon) and assign a name to it, then click **Add**. The elements will be added to the list below in the order they were entered. The first element will be displayed when the linked entity takes value zero, the second when it takes value one, and so on.
To change the sequence, select the element you wish to move and press the **Move Up** or **Move Down** button as required.

Once you have added all the elements, you may proceed to their *configuration*: to display the properties of an element, select it from the list and press **Properties**. To find out the meaning, refer to the properties of Texts and Icons.

If the page is multilanguage to modify the *Text* property of the text elements it is necessary to use the multilanguage editing strings window. In this case the preview can be shown only through the EIML Simulator.

To see an example on how to use Combo objects, refer to the Icon sample included in the Samples folder.

Table

Tables are an additional innovative object type designed to help you develop graphical interfaces and characterized by being **scrollable**.
Therefore, using tables you will be able to display large amounts of data without needing to load more than one page.
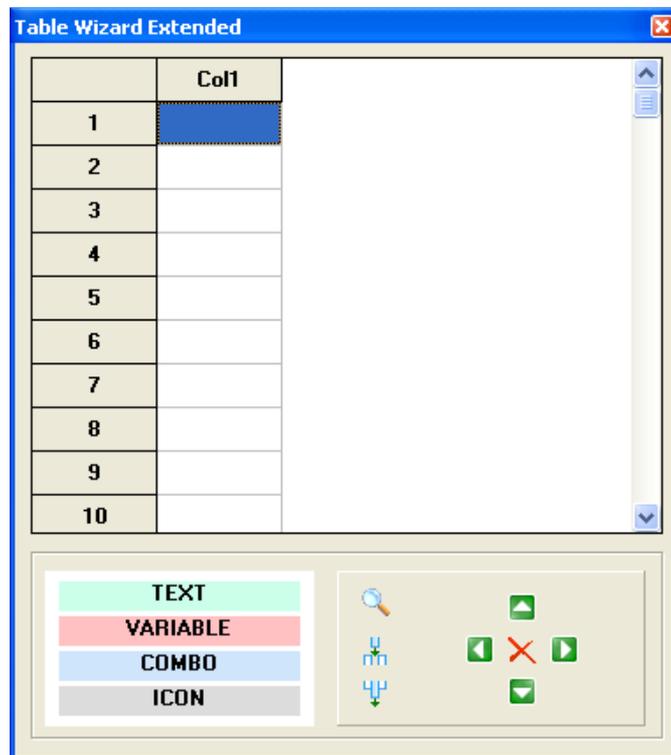If we add a table to a page, we will be able to display its properties:

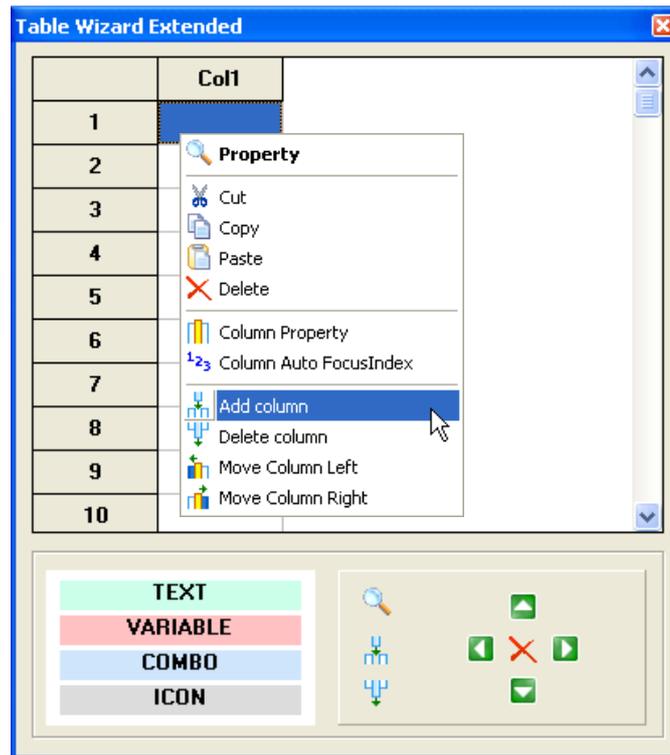|	
Name	Unique name that identifies the element inside a project.
Left	Offset from the left edge of the page, in pixels.
Top	Offset from the top edge of the page, in pixels.
Width	Element width in pixels.
Height	Element height in pixels.
Nrow	Indicates the number of visible rows in a table. This property is calculated automatically based on table height, row height, and whether or not the table has headers.
RowHeight	Defines row height in pixels. It is logically linked to table height, header display, and total number of table rows.
Color (only color pages)	Color used as background of the table.
Font Custom (only color pages)	It specifies wich project custom font use to draw the table text items. If it is set to <NONE> standard font is used (Fontsize property).
Fontsize	This property defines the font size of all text elements displayed in the table.
Header Color (only color pages)	Color of the background of the header
Header Font Color (only color pages)	Font color of the header
Header	Set this property to select whether or not to display column headers.
Borders	If enabled, table borders will be displayed, otherwise they won't.
Borders Colors (only color pages)	Colors of the lines of the edges of the table.

To add objects to a table, use the **Table Wizard Extended** (right-click the element and then choose Table Wizard Extended).
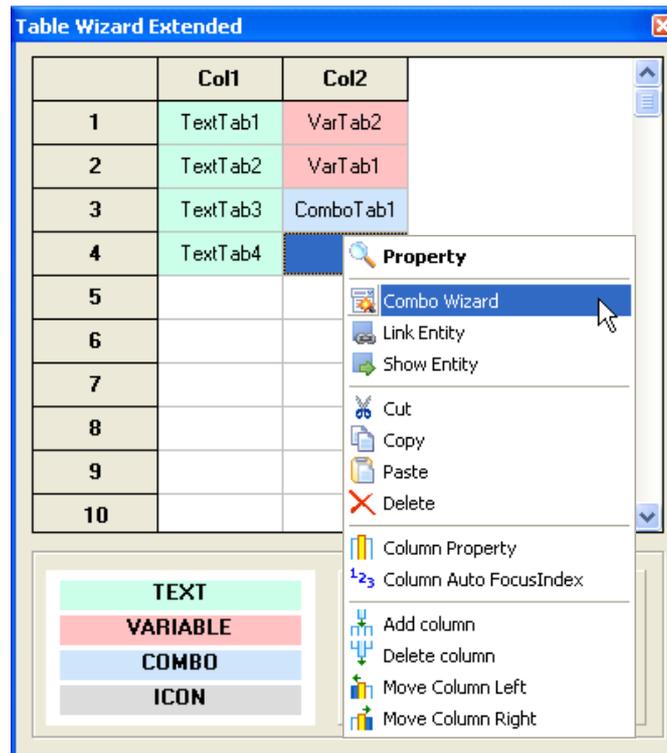
The following window will appear:

The first thing you need to do to use a table is create its columns. The Table object allows you to add from one to four columns. By default the table has one column active. It is possible to create new columns using the contextual menu (right-click the column and choose *Add Column)*:

For each column you may set column title, width, header and alignment selecting *Column Property*, or to change the column position with *Move Column Left* and *Move Column Right*.

If the page is multilanguage to modify the *Text* property of the text elements it is necessary to use the multilanguage editing strings window. In this case the preview can be shown only through the EIML Simulator.

To insert the required elements to each column it is necessary to select it from the list on the bottom and use your mouse to drag it to the desired position in column:

Using the positioning buttons on the right-bottom size it is possible to move the selected item into the table: to move it among rows and columns you have to click the corresponding arrows, instead to delete it you have to click on the X in the middle. To change the properties of an element, select it and select **Property** from the contextual menu, or double-click on the focused element.
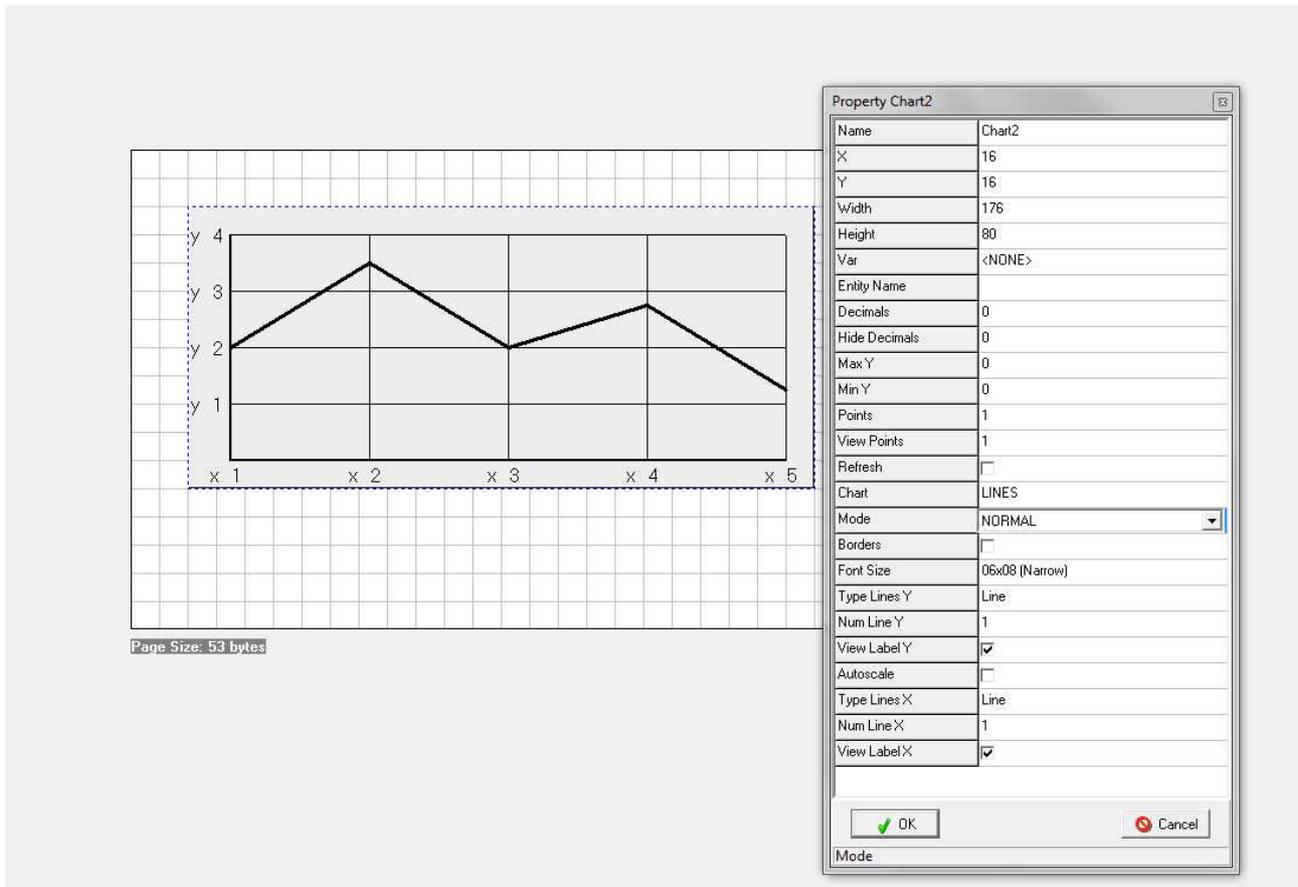
From the contextual menu (right-click on the selected item) it is possible to copy it or view its properties, for example it is possible to link a variable or a combo to a project entity. If you have added a Combo to the list, after setting its properties you will need to show the **Combo Wizard** to choose which elements you wish to link to it**.**

Some of these operations may be realized by key shortcuts. The following table summarizes the actions associated to the key shortcuts:

Key	Action
Ctrl+Up, Ctrl+Down, Ctrl+Left, Ctrl+Right	Move the selected item into the table.
Ctrl+C, Ctrl+Ins	Copy the selected item.
Ctrl+X	Cut the selected item.
Ctrl+V, Shift+Ins	Paste the selected item.
Canc	Delete the selected item.
Enter, F11	Show the properties of the selected item.
Ctrl+1, Ctrl+2, Ctrl+3, Ctrl+4	Insert into the column a text, a variable, a combo or an icon respectively.

Chart

Chart objects allow ro create simple graph, by connecting a variable as an array entity, it is possible to represent a range of up to 100 points. There are a number of other properties that allow you to configure the graph according to the needs (type, axes, color, values, ...).

The association of the series with the entity project is possible in the same way of the Variables, using the tool Entity List (look at **Variables**). Using the tool, you can also set the first point from which to start drawing the series, to do this you simply select from the list the desired position in the array. To start the design from the first point to select the location *EntityName[0]* array.

For sun color pages is possible to draw up to 3 different series, the reference variable will always be the same, will however filled ad-hoc deciding in advance what will be the indices of start for the second and third series, the first series will always be the reference.

On afterwards the properties of the Chart elements:

Name	Univocal name identifying the element in the project.
X	Offset in pixel from the left hand side of the page
Y	Offset in pixel from the top of the page
Width	Width of the element
Height	Height in pixel of the element
Var	Using this property you can link a project entity (such as Var, Par, Pers.). Select this property and click the button that will appear: a window will open, where you can choose one of the project entities whose value you wish to display the series.

Decimals	Number of decimals used to represent the value [0..3] on the Y axis. This is a read-only property and is automatically set when the entity you wish to display is joined (see *Precision* under entity properties).
HideDecimals	Allows you to change the number of decimals that you wish to hide. This property can assume any value between 0 and the value of the Decimals property and has the purpose of hiding some decimal digits.
Max Y	Maximum value of Y. The points over this value will be exluded from the drawing of the series.
Min Y	Minimum value of Y. The points under this value will be exluded from the drawing of the series.
View Points	Sets the number of points that will be drawn from the starting point of the graph.
Idx S2 **(only color pages)**	Array index that indicates the start of Series 2. If =0 zero, the series 2 is not drawn.
Idx S3 **(only color pages)**	Array index that indicates the start of Series 3. If =0 zero, the series 3 is not drawn.
Refresh	If the Refresh property is enabled, the variable will be continuously requested; otherwise, it will be refreshed slowly so as to avoid overloading communication between controllers.
Chart	It allows to decide the chart type: ➢ POINTS. ➢ LINES. ➢ HISTOGRAM.
Color **(only color pages)**	Color used for the background of the graph
Axis Color **(only color pages)**	Color used for the axis of the graph
Lines Color **(only color pages)**	Color used for the lines/tick of the graph
Mode **(only pages in black & white)**	Display mode of the graph, it can be of two types: ➢ NORMAL ➢ NEGATIVE (only page in balck and white) The NORMAL status allows to view the segments of the graph and text in black on a white background; vice versa NEGATIVE sets the axes and the white text on black background.
Borders **(only pages in black & white)**	If enabled, Chart area borders will be displayed, otherwise they won't.
Fontsize	Describes the font size used to write the label near the axes. For example, if you use an 8x8 font size, a character will be 8 pixels high and 8 pixels wide
Font Color **(only color pages)**	Color used for the label font.
Type Lines Y	It allows to display an horizontal grid under the chart: ➢ NO: no lines

	➢ LINE: horizontal lines
	➢ TICK: tick over the Y axis
Num Lines Y	Numbero of the lines/ticks on the Y axis [1..4].
View Label Y	It allows to display the value near the y- axis.
Autoscale	Se attiva, il grafico si scalerà automaticamente tra il punto massimo e minimo della serie 1.
Type Lines X	It allows to display a vertical grid under the chart:
	➢ NO: no lines
	➢ LINE: vertical lines
	➢ TICK: tick over the X axis
Num Lines X	Numbero of the lines/ticks on the X axis [1..ViewPoints].
View Label X	It allows to display the value near the x-axis.
S1 Color (only color pages)	Color used for the Series 1.
S2 Color (only color pages)	Color used for the Series 2.
S3 Color (only color pages)	Color used for the Series 3.

Note. For each EIML page it is allowed to use only a Chart object.

Line

To make an EIML page more graphically attractive, you may also add lines to it by pressing ⟍ .
A line has the following properties:

Name	Unique name that identifies the element inside a project.
X1	X coordinate of the point you clicked first
Y1	Y coordinate of the point you clicked first
X2	X coordinate of the point you clicked second
Y2	Y coordinate of the point you clicked second
Color	Color used to draw the element: it can be either black or white in the black and white pages or one of the 256 colors in the color pages.

Rectangle

To make an EIML page more graphically attractive, you may also add rectangles to it by pressing
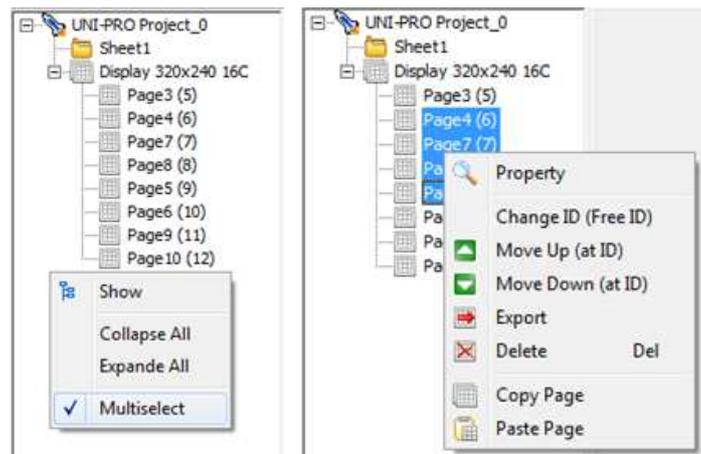☐ . A rectangle has the following properties:

Name	Unique name that identifies the element inside a project.
Left	Offset from the left edge of the page, in pixels.
Top	Offset from the top edge of the page, in pixels.
Right	Offset from the right edge of the page, in pixels.
Bottom	Offset from the bottom edge of the page, in pixels.
Color	Color used to draw the element: it can be either black or white in the black and white pages or one of the 256 colors in the color pages.
Filled	Indicates whether or not the rectangle should be filled with the specified color.

3.5.9 Multiselect from the page tree

From the page tree, by clicking with the right button, you can activate the Multiselect mode from the drop-down menu that appears.

This mode allows you to select multiple pages (with the Shift and Ctrl keys) and do some special operations, including:

Move Up / Down (at ID): moves the selected pages up/down and positions them consecutively at the ID that is indicated, if the indicated ID does not exist, no movement takes place.

Change ID (Free): changes the ID of the selected pages starting from what is indicated, occupies only the free IDs, if there is an overlapping of IDs the movement cannot be carried out.

Export: exports the selected pages.

Delete: delete the selected pages.

4 ADVANCED OPERATIONS

4.1   Commands

Commands represent a powerful development tool to create applications that are capable of managing events such as requesting a new page, resetting an Event Historian, resetting parameters with defaults, and informing another machine connected in the network about a certain state.

Commands can be managed in the sheets using the **CommandIn** and **CommandOut** entities or assigning a command different from *NONE* to the *Command* property of certain EIML page elements (Texts, Icons).

Commands are divided into three categories: **system, environment,** and **project** commands. System commands have a unique value assigned by the system and valid for all products (warning: certain products might not respond correctly to certain pre-set commands because they haven't the corresponding function; for instance, the *Erase Historian* command will not work on those devices that do not have the event logging feature). In the case of environment and project commands, their values may be assigned at the developer's discretion. The difference between them lies in the fact that environment commands may be used in all projects, while project commands are project-specific.

Choose *Tools/Commands...* from the menu to open the window below that shows the system commands available and allows you to add or change environment and project commands.

By convention, values are reserved as follows: from 1 to 50 for system commands (pink), from 51 to 150 for environment commands (violet) and from 151 to 250 for project commands (yellow).
The structure of a command is composed of three fields:

➢ COMMAND: indicates the numeric value of the command. A name is linked to each command, which is pre-set for system commands, valid for all projects for environment commands (51 to 150), and project-specific for project commands (151 to 250).

➢ PARAMETER: is a 16-bit value linked to the command. Depending on the command, it may be seen as a single field or as the combination of several fields.

➢ NODE: is the value of the network element sending (or receiving) the command. This value is assigned by means of the corresponding join window.

To send a command from an EIML page, place the cursor over an element (text or icon) that has a *Command* property different from zero and press *ENTER*.

A command embedded in a sheet will be activated when the *trigger input* goes from zero to one.

Below is a list of all available *system commands* and their meanings:

➢ **Load Page**: Command to request/send a page. The corresponding *parameter* indicates the index of the requested page.

➢ **Mute Buzzer**: Command to silence the buzzer at the card identified by the *node*. If the node field is set to broadcast, this command will silence all elements in the network.

➢ **Sync Clock**: Command to synchronize the clock of the destination node with the clock of the source node.

➢ **Save Par Drv**: Command to create a backup copy of all system parameters (warning: this operation may take some time to complete).

➢ **Save Par App**: Command to create a backup copy of all application parameters (warning: this operation may take some time to complete).

➢ **Restore Par Drv**: Command to restore all system parameters from the backup copy (warning: this operation may take some time to complete).

➢ **Restore Par App**: Command to restore all application parameters from the backup copy (warning: this operation may take some time to complete).

➢ **Load Default Par**: Command to load the default values of all parameters set during the design phase (warning: this operation may take some time to complete).

➢ **Erase Historian**: Command to erase the Event Historian (warning: all data will be deleted) .

➢ **Send Page**: Command to send from algorithm a particolar page to the display of a controller belonging to the CAN network.

➢ **Toggle ON/OFF**: On/Off device command. The management of this command must be implemented in the project.

➢ **Prg Level 1-5**: This command requires to send the page with the smallest index relative to the safety level 1-5 (to enable this features operate on the system parameter *EnablePrgLevel*).

➢ **Request Alarm**: Command to ask the first alarm

➢ **Request Historian**: Command to ask the first element in the historian

➢ **Key Save Par Drv**: Saves the hardware configuration parameters in the USB peripheral

➢ **Key Save Par App**: Saves the application configuration parameters in the USB peripheral

➢ **Key Restore Par Drv**: Loads the hardware configuration parameters in the USB peripheral

➢ **Key Restore Par App**: Loads the application configuration parameters in the USB peripheral

➢ **Send EVCM Commands**: Sends the commands to EVCM (*)

➢ **Send EVCM Manual Pos**: Sends the manual setpoint of the position to the EVCM (*)

➢ **Receive EVCM Current Pos**: Receives the running position of the EVCM (*)

➢ **Receive EVCM Status**: Receives the status of the EVCM (*)

➢ **Receive EVCM Alarms**: Receives the status of the alarms of the EVCM (*)

➢ (*) Look at the manual of EVDRIVE03.

4.2   Navigating through the pages

An extremely interesting aspect to be discussed is how to recall a page from another page and how the pages are interconnected. We already mentioned that pages reside in the controller and their creation coincides with the development of control algorithms. In the project tree, pages are displayed after the sheets, in one or more folders, depending on the number of displays the specific controller has.

This multiple-display management feature enables displaying pages created for a certain user interfaces with a more powerful "browser" (for example, you can actually display pages written for a 20x4 alphanumeric display on a 240x128 graphic display!).

For instance, referring to the above page tree, the pages with ID = 3 and ID = 4 could be alphanumeric pages designed to be displayed by the graphic display as well, thus enabling significant code savings: when the 240x128 display requests the page with ID = 3, if a first search in the list of graphic pages is unsuccessful, it will search the list of alphanumeric pages and display the page found there, but conveniently resized.

In general, navigation between pages is made possible through a mechanism that uses the **Load Page** command (refer to section *Commands*). For example, if you set the command and param properties of a text as follows: *Command=Load Page*, *Param=3*, the page with ID = 3 will be requested. In special cases, for example if you need to request the alarm page by pressing a certain key, it is possible to send a page to a user interface panel using the CommandOuts in response to a "key pressed" event. In this case, in addition to the page ID, you will also need to define the required browser type in the Param input, according to the following table:

```
DISPLAY 7-Segments 4 Digit        = 1
4X20  ALPHANUMERIC DISPLAY        = 3
128x64   GRAPHIC DISPLAY          = 4
240x140  GRAPHIC DISPLAY          = 6
120x32   GRAPHIC DISPLAY          = 7
DISPLAY 7-Segments 2x4 Digit      = 8
320x240 COLOR GRAPHIC DISPLAY     = 9
320x240 16 COLOR GRAPHIC DISPLAY = 10
```

Setting *PreviousPage*, *NextPage* and *TimeoutPage* properties it is possible to define navigation between pages. By assigning them values different from <NONE>, it is possible to define which pages load respectively when pressing LEFT, RIGHT or ESC key.

In addition, setting *Timeout* and *TimeoutPage* you can define the period of inactivity before a page is loaded.

4.3   Password-protected information

In many applications, access by unauthorized users to certain data needs to be prevented in order to ensure a high safety level. Changing improperly controller's set-up or operating parameters may lead malfunctions or shutdown of the whole system.

The protection system adopted is based on 5 protection levels that may be assigned to each EIML page by setting the *Level* property. Each level is linked to a different password in the controller. When a display requests a page whose Level property is set to a value higher than zero, the controller checks that the request is accompanied by an authorization level equal to or higher than the page level. If that is not the case, it prompts the user to enter a password. If the password entered belongs to one of the 5 levels the controller has, the authorization level for the specific display will be set to the level of the password.

For example, let us suppose that the controller has the following password set-up page:

If the display, which starts from an authorization level equal to 0, requests a Level=2 page, the controller will prompt the user to enter a password.

If the password entered is 1111, the user will be prompted to enter it again because the new level (1) is still lower than the requested one (2). If the password entered is 3333, the requested page will be sent to the user, who will have free access to all pages having a level equal to or lower than 3.

If no keys are pressed on the keypad during a pre-set period of time (system parameter, default 60 seconds), the authorization level will be reset.
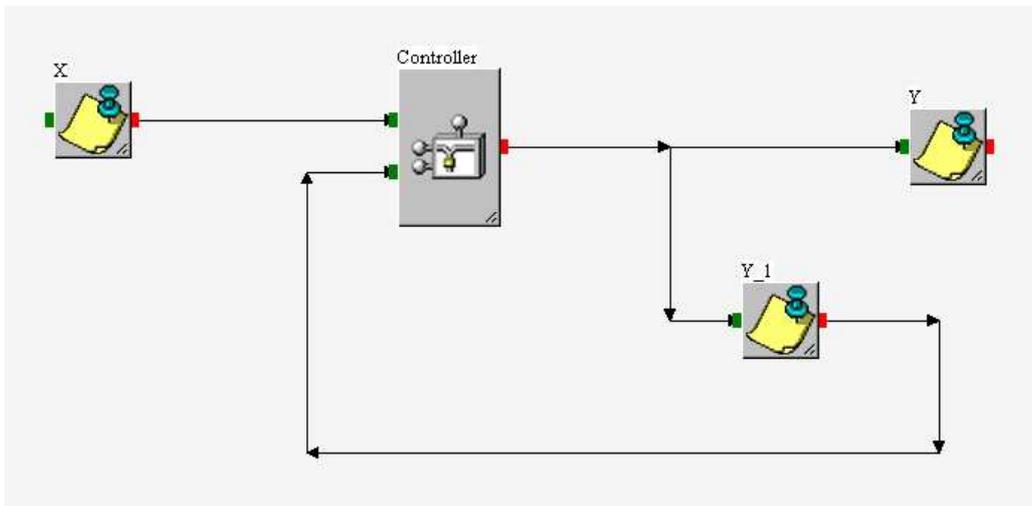
4.4   Defining execution tasks

All calculation operations defined through links of entities can be executed in two different tasks. Usually the greater part of operations are executed in the Main Task. Where it is necessary to calculate some critical operations it they can be executed in the high priority tasks Timed100ms, which execute under interrupts every 100 ms.

To notice that these last two tasks must be used only when it is absolutely necessary and their overload can introduce execution errors.

4.5   Setting the program's execution order

The program, whose graphical interface is designed by joining entities in the sheets, follows certain pre-set rules to assign an execution order to the calls to the various algorithms. Such order depends mainly on the path from outputs to inputs and the order in which they were added to the sheets, which is automatically assigned by the code generator. Quite often, though, the execution of certain parts of a program needs to be forced to a pre-set order. Feedback connections, like the one illustrated in the figure below, are a good example.

In this case, we want output Y to be calculated by the algorithm using input Y_1 (which represents the previous output) *before* the input is updated! To assign a specific order to the execution of this part of the program, we need to set the right sequence to the values of the *Order* property of all the involved variables. In the example above, the right operation will be ensured if the Order property of Y is higher than (or equal to) that of X, but lower than that of Y_1. Therefore, a possible combination is:

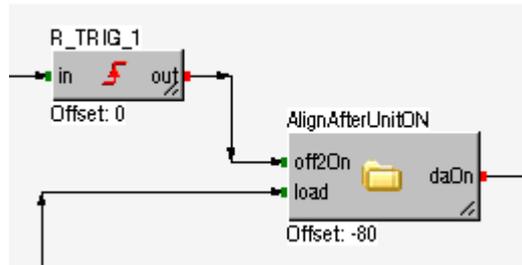- ➢ Order X   = 0
- ➢ Order Y   = 1
- ➢ Order Y_1 = 2

This means that, when executing the program, variable X will be calculated first, then Y and finally Y_1.

The entities affecting the program's execution order are all those that have the property Order, i.e.: Alg., Var, Par, Pers, DigitalOut, AnalogOut, LED, Buzzer, and CommandOut.
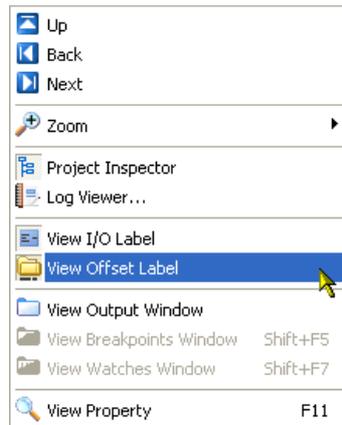
Operating on the property *Offset* on a subsheet, it is possible to move the execution order for all the entities it contains.

The OFFSET properties under the Sheet and libraries can be displayed.

To activate this function go to:
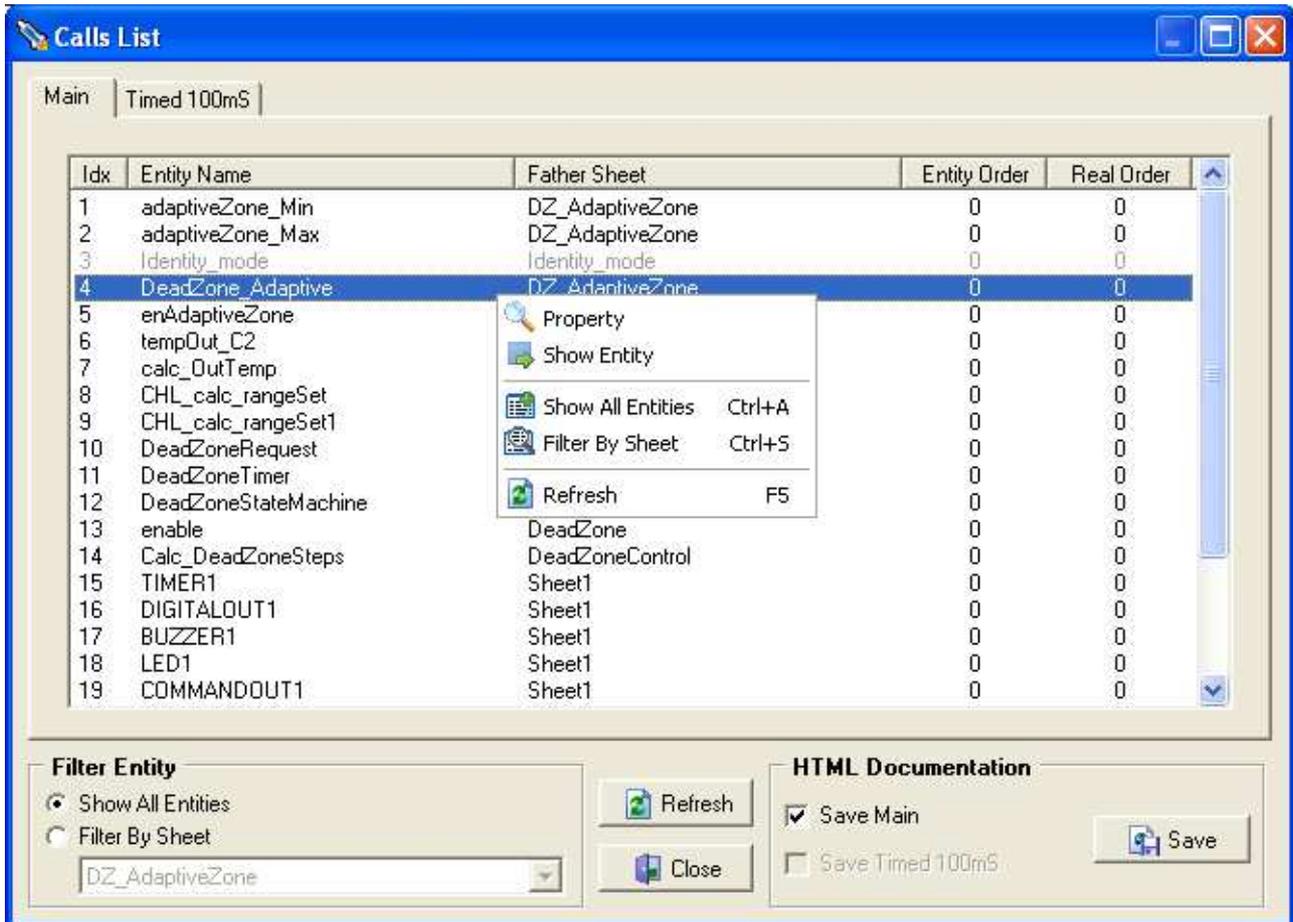1) Menù View->View Offset Label

Or

2) From *Settings*

4.5.1 Calls List

To view the ordered list with which portions of the program are calculated, the *Tools / Calls List* menu can be activated. As can be observed in the window below
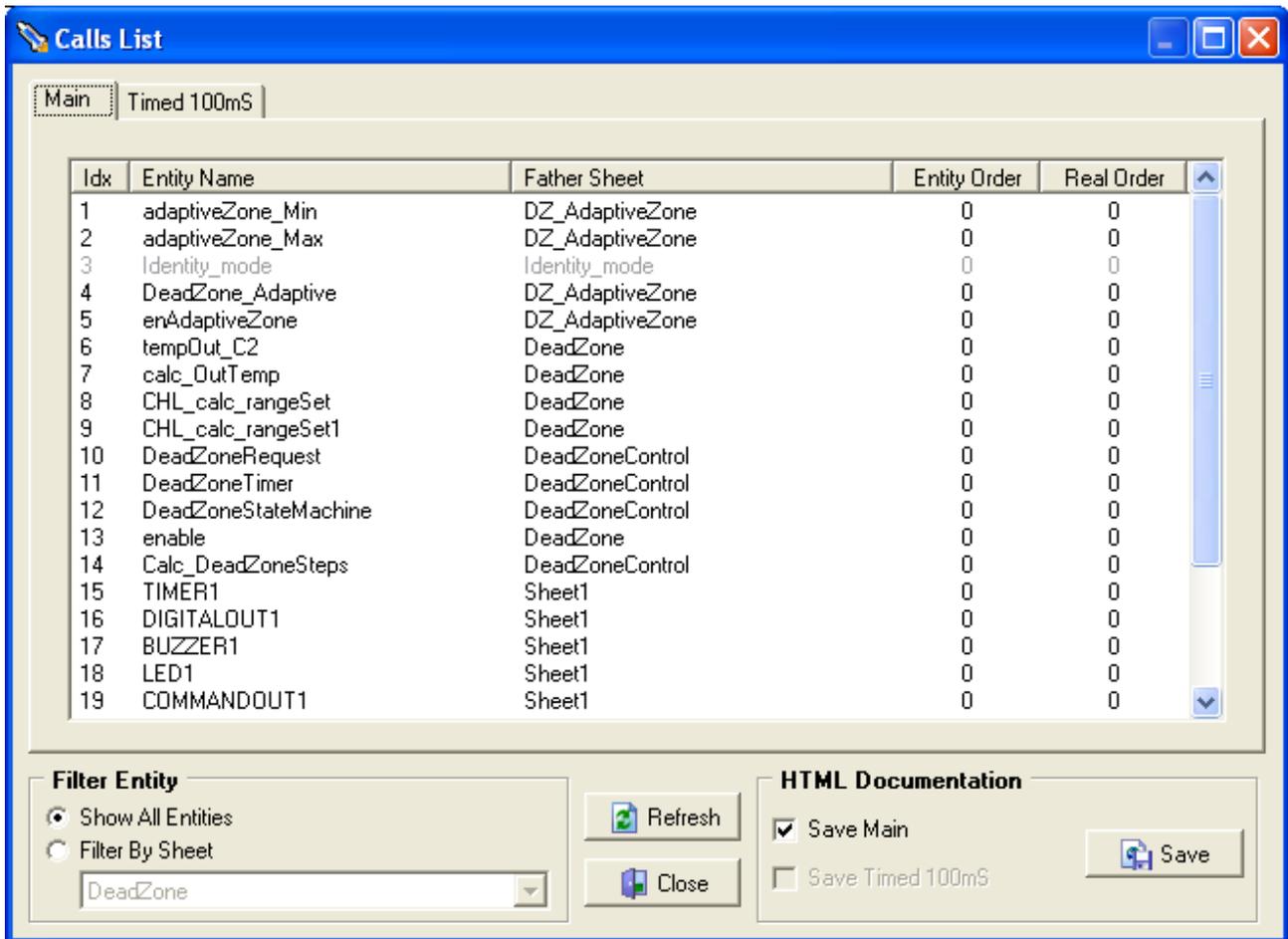

The calls are subdivided into three sections (one per task) and will be displayed in the order in which they will be executed. By double clicking on any of these rows, the entity will be searched in the project and viewed.

From the contextual menu (right-click on the selected item) it is possible to view and modify the selected entity properties. Modifying *Order* property it's possible change the tasks execution order and re-calculate "on fly" the new program's execution order by clicking on *Refresh* or by the contextual menu.

Selecting *Filter By Sheet* and the interested Sheet is possible to view only the entities of a particular sheet to have a most accurate view of that you want to control .

To select the desired sheet you can select it from the combo also clicking on the row's table that is referred at this one.
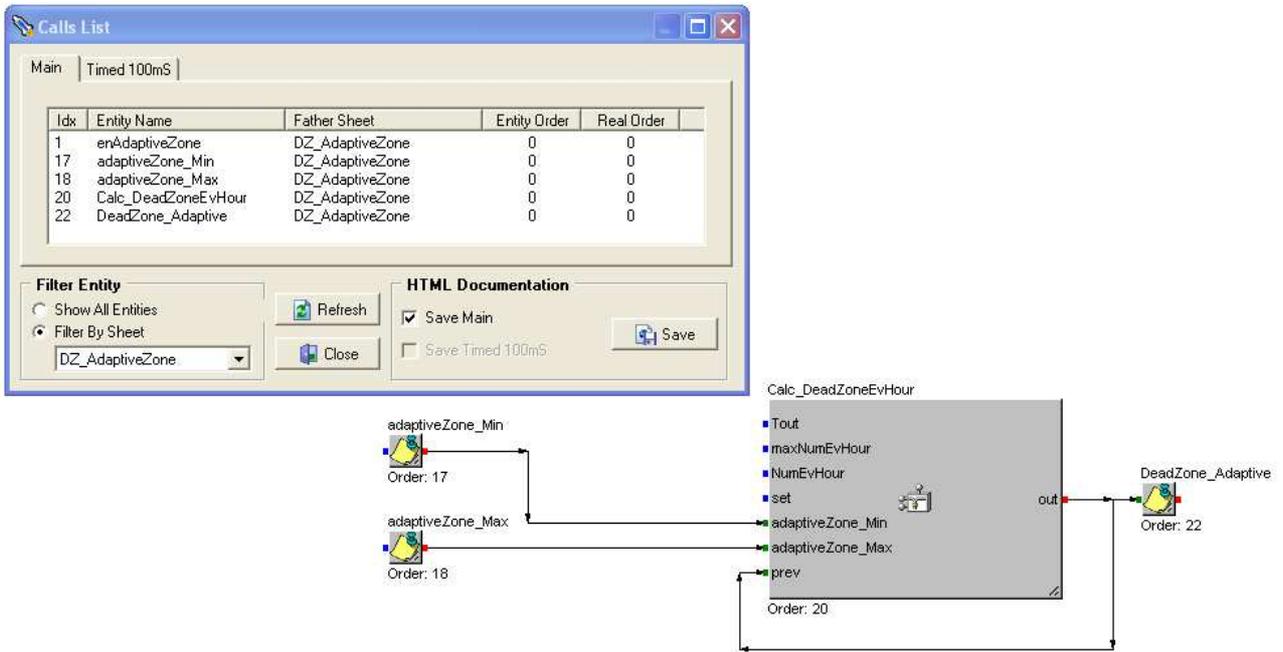
Anyway is possible to return at the general view selecting *Show All Entities* and clicking on *Refresh* button or selecting *Show All Entities* from the contextual menu.

Using *Save* button you can save the selected table into an HTML file document.

4.5.2   Execution order during the calls list added

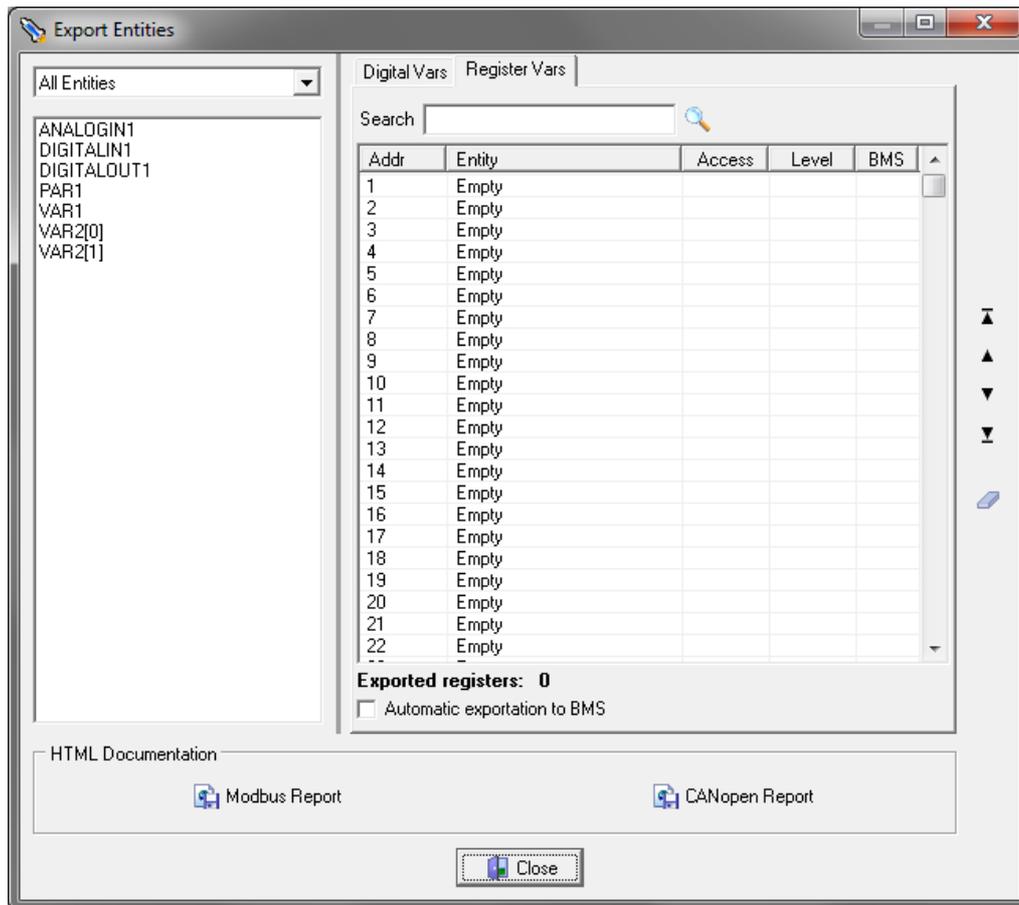When the Calls List is active you can see the execution order under that calculated entity; that is if and when it is used. It is helpful for understanding the correct main cycle of the program.

In this case it is immediate understood that the variable is calculated right after the algorithm.
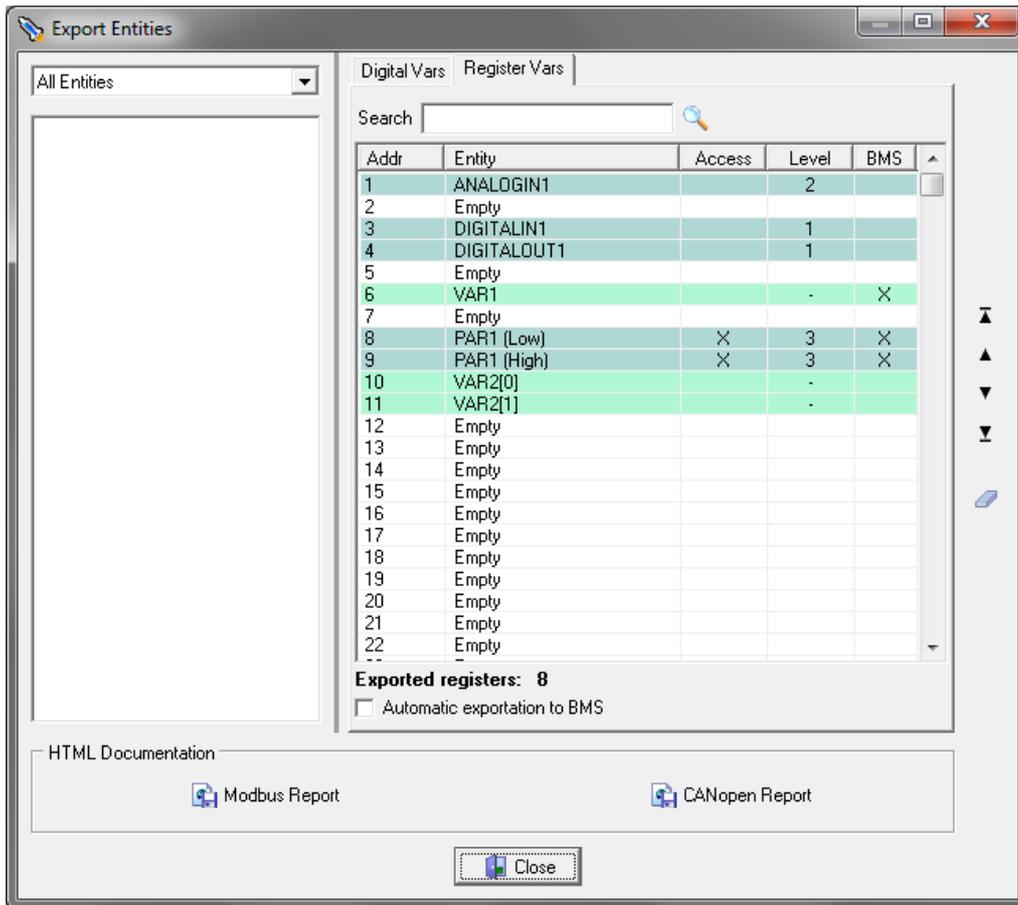
4.6 Exporting internal variables

In order to access the controller's internal variables through one of the available communication protocols, you need to define which variables you want to "export" (i.e. make visible outside) and sort them according to a criterion capable of maintaining the same sorting order even in the event that changes are subsequently made to the program.

To decide which variables to export and define their sorting order, you may use the following window (choose *Tools/Export Entities...* from the menu to open it):

On the left side there is a list of all project variables, while on the right side there are two additional lists that can be selected by clicking the corresponding tabs: *Digital Vars* and *Register Vars*. To make search easier, use the combo found at the top left corner of the window to filter the list by type.

To add a variable to the list, select it from the list on the left and use your mouse to drag it to the desired position in the list on the right.
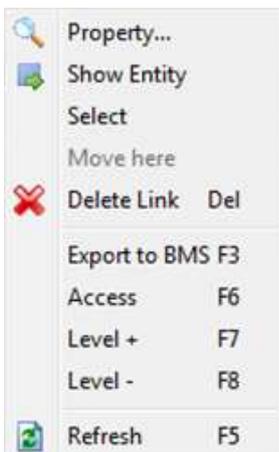
As a result of this operation, the name of the variable will disappear from the list on the left and will appear in the list on the right, at the position indicated in the ID column. Using this drag and drop method you can move the elements to the list of exported variables or bring them back to the list of variables that have not been exported yet.

The column *BMS* indicates if the exported entity is used (X) or not in the BMS (WebServer) protocol. Through the F3 key you can select/deselect this property.

The column *Access* indicates whether the exported entity is protected in reading (X) or not in the Modbus protocol. Through the F6 key you can select/deselect this property

The Level column indicates the level of write protection of the entity exported on Modbus, if there is no protection (level 0) is displayed "-". Through the F6 and F7 keys, you can increase / decrease the level of protection.

As alternative it is possible to use the contextual menu.

Using the contextual menu it is also possible to select and move a block of entities from a position to another one, without making the exportation again. Once the entities have been selected, choose "*Select*" from the contextual menu, to open it again from the new position, then "*Move Here*". In this way the entitied will be placed in the new position.

To manage in the project the R&W Modbus protection, it is possible to use some system libraries or other system function calls from inside the algorithms:

- To enable the access of an entity that is protected in reading (X), it is possible to use the library Modbus1_Protection_SetAccess with the input status=1 and setting to 1 the input enable

  o
  o Or calling the function **CJ_Modbus_SetAccess(1)**;

Setting to 1 the input enable with status=0 or calling the function **CJ_Modbus_SetAccess(1)**, the protection in reading is restored.

- To enable the access of an entity that is protected in writing with for example level 2, it is possible to use the library Modbus1_Protection_SetLevel with the input status=1, with the input level=2 and setting to 1 the input enable

  o
  o Or calling the function **CJ_Modbus_SetProtections (1, 2);**

Setting to 1 the input enable with status=0 and level=2 or calling the function **CJ_Modbus_SetAccess(0,2)**, the protection in writing for that level is restored.
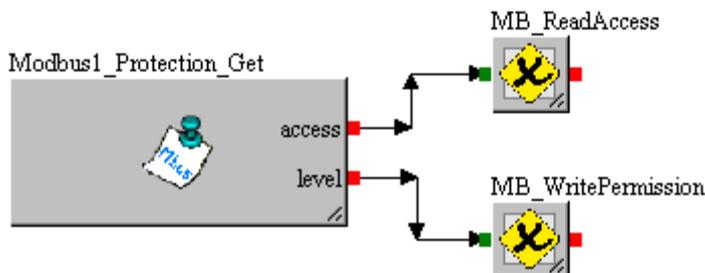
- The Modbus protection, both in reading and in writing, can be automatically restored once the timeout after the last received Modbus communication has expired, using the proper library to set this timeout:

  o
  o Or calling the funzction **CJ_Modbus_SetTimeOutProtections(value);**

A value equal to 0 in input of the library, or calling the function **CJ_Modbus_SetTimeOutProtections(0)** causes the unlimited unlock.

- To know at any time the protection status for both reading and writing protection, it is possible to use the library

  o

o Or calling the function CJ_BYTE **CJ_Modbus_GetStatusProtections()**, in the lower part of the byte is possible to read the current writing permission level, while in the bit 4 the reading access status, for example:
level = CJ_Modbus_GetStatusProtections()&0x0F;
access = CJ_Modbus_GetStatusProtections()& 0x10) > 0;

Depending on the list, only certain types of variables may be exported, as summarized in the tables below:

Digital Vars	
Entity	*Data type*
Digital In	
Digital Out	
Par	only CJ_BIT
Pers	only CJ_BIT
Var	only CJ_BIT
Fixs	only CJ_BIT

Register Vars	
Entity	*Data type*
Analog In	CJ_ANALOG: only the value
Analog Out	
Clock	
Timer	
Par	All
Pers	All
Var	All
Fixs	All

NOTE: 32-bit data types (CJ_LONG, CJ_DWORD, CJ_DATETIME, CJ_DATE and CJ_TIME) take up *two rows* in the Register Vars list: the low 16 bits are in the first row and the high 16 bits in the second one.

If in the *Settings* option "*Export Packed CJ_CHAR*" is selected, the types CJ_CHAR are exported in Modbus by groups (made of two elements) for each register.

4.6.1   Modbus and CANopen Documentation

It's possibile to show the export in a HTML document. This document will contain all the useful information for the exported entities both for Modbus and CANopen protocols.

To use this feature simply click on the bottons *Create ModBus Documentation* or *Create CANopen Documentation* placed in the underside of the *Export Entities* window.

The CANopen document contains the information to access to the project variable through index and sub-index of the *Object Dictionary*.

The Modbus document contains the information to access to the project variables through Modbus read/write commands of *coils* and *registers*.

4.7   Conditioned Visibility

The conditioned visibility allows to hide configuration parameters and status according to particular configurations. Some entities could be only visible if a particular configuration are made, in this way the number of the parameters and the status displayed by the user interface will be less than the total number that will be configurable. In other words, all not relevant configuration entities will not be showed by the user interface. For a user it's more simple to scroll the user interface and the search and the settings of a specified entities are faster. It's usefully also because the conditioned visibility could hide some particulars parameters and status that are dangerous for the application and so setting errors from an improper user could be induce a wrong operations.

The conditioned visibility can be using with variables, parameters and persistent linked at the EIML pages with variables, combos and tables. For can use this functionality is required follow some essential steps:

1. To select the *condvisible* property for all the interested entities
2. To export all the interested entities on Modbus protocol with *Export Entities* tool.
3. To define the algorithms that rule as display or hide the interested entities. Using firmware function   **void CJ_SetCondVisBit(word idx, bool value)**
4. To select the *CondVis* property in each page where there are variables with conditioned visibility

All the points must be correctly made else the conditioned visibility of the entities will be no effect.

In the alphanumeric 4x20 display and in the graphic 240x128 display all the entities hide for conditioned visibility will be substitute with some points "…." and will not be editable.

In the 7-segment display the hide entities will be excluded from the visualization.

4.7.1   Firmware function   void CJ_SetCondVisBit(word idx, bool value)

This function allows to directly rule from algorithm the visibility of an exported variable at the *"idx"* address on Modbus protocol. Function use two parameters:

·   *word idx*: are the Modbus address entity exported on Modbus protocol (tool *Export Entities*)
·   *bool value*: get the entity visibility. This parameters could be a boolean condition, too;
      If *value = 1* (or true condition) the entity will be hidden
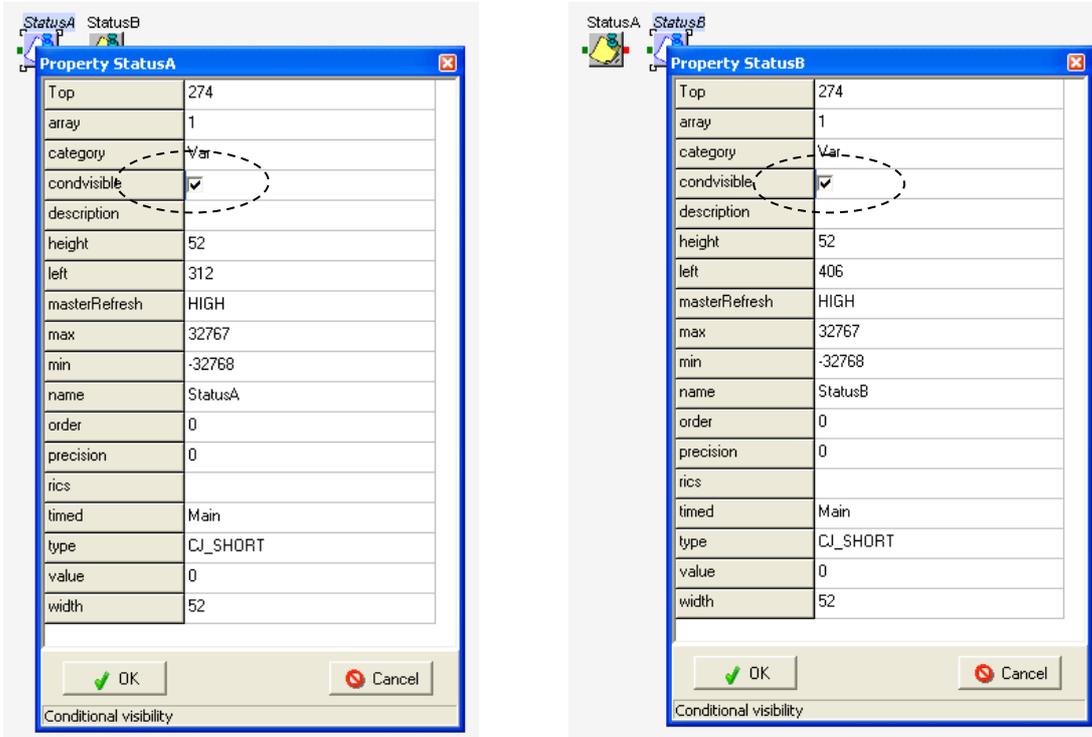   If *value = 0* (or false condition) the entity will be showed

A function call as **CJ_SetCondVisBit(idx, 1)** made the entity at the *idx* address always **hide**.

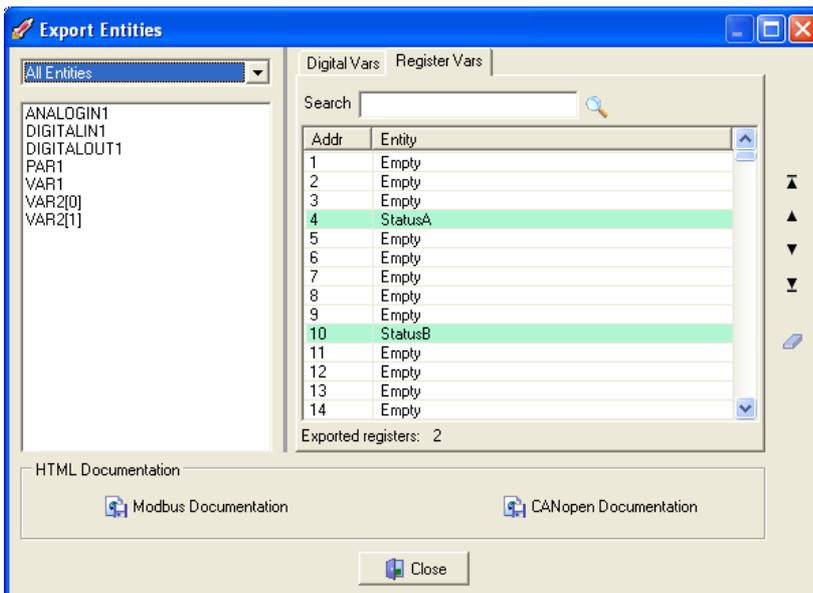A function call as **CJ_SetCondVisBit(idx, 0)** made the entity at the *idx* address always **visible**.

4.7.2   Example

With a simple example let's show the steps to follow to correctly use the conditioned visibility. Will suppose to want to condition the visibility of two variables, *StatusA* and *StautsB* with a CJ_BIT parameter P001.

As first step are required to select the *condvisible* property of both entities.

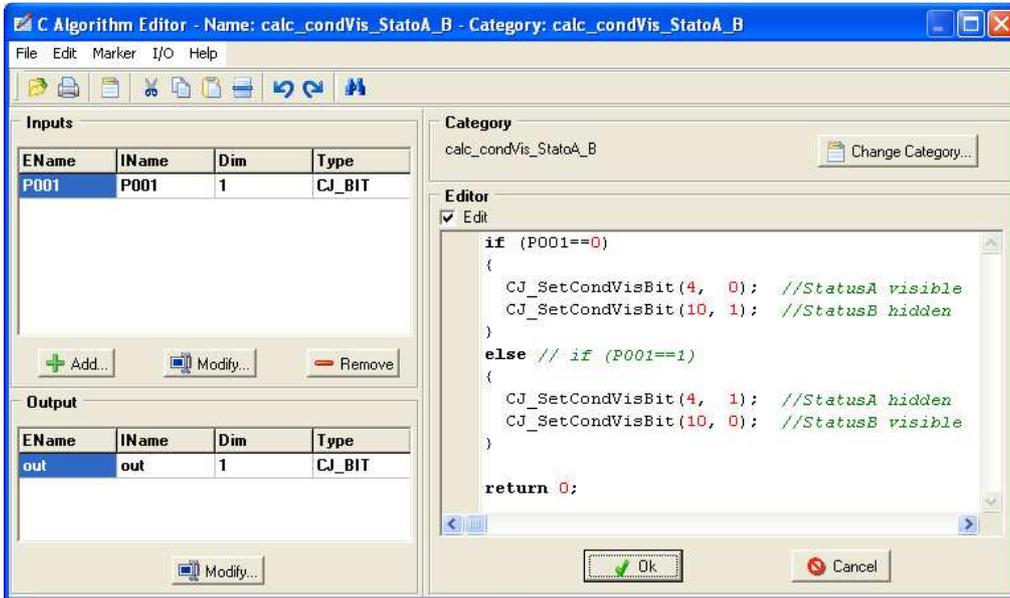As told before the two variables must be exported on Modbus table:

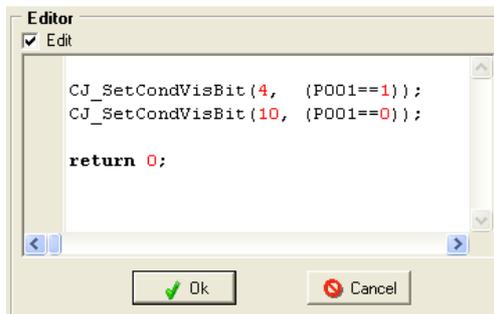With *Export Entities* tool export the variable *StatusA* at the address 4, and the variable *StatusB* at the address 10.

Now it's possible to write the algorithm to rule if display or hide the two variables. In the example we want to:

·   with *P001=0*: the *StatusA* is visible (with ON value), and the *StatusB* is hide
·   with *P001=1*: the *StatusA* is hide, and the *StatusB* is visible (with ON value)

The algorithm to rule the conditioned visibility are the following:

Same result we have as re-write the algorithm as a condition for the function *value* parameter:

The user interface effect on 4x20 (V-View) or on 240x128 (V-Graph) are the following:

With P001=0:   **With P001=1:**

The 7-segments display user interface effect are the following, the row with the *StatusA* or *StatusB* hidden will not be displayed in the table:

Without conditioned visibility: **With P001=0:**   **With P001=1:**

StatusX	1
StatusA	ON
StatusB	ON

StatusX	1
StatusA	ON

StatusX	1
StatusB	ON

4.8 Semaphores

Through the use of the functions *CJ_WORD CJ_SemaphoreRead(CJ_WORD i)* and *CJ_BYTE CJ_SemaphoreWrite(CJ_WORD i, CJ_WORD value)*, it is possible to use 10 semaphores to condition the access to a resource during the execution of a program.

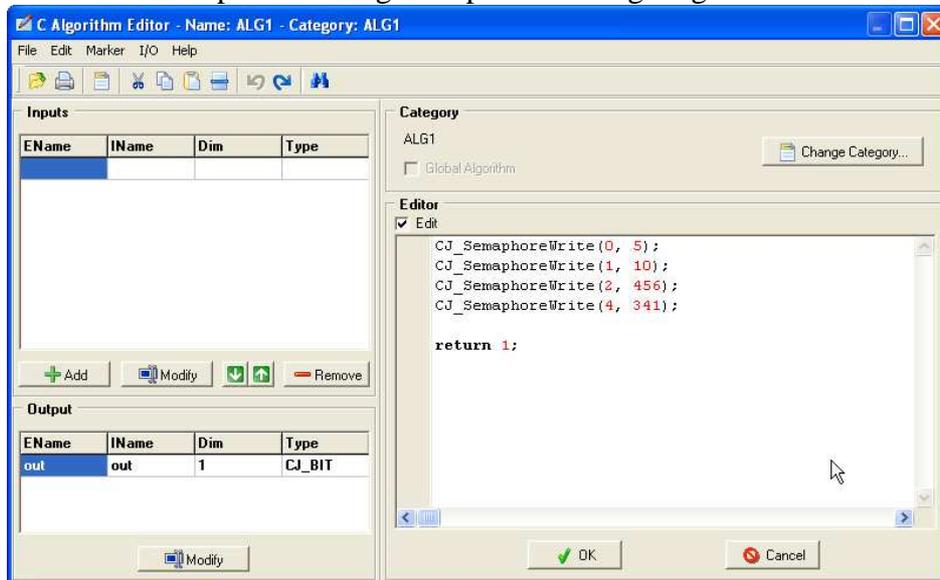Function *CJ_WORD CJ_SemaphoreRead(CJ_WORD i)* allows reading the value set by the the semaphore n-tn.

Function *CJ_WORD CJ_ SemaphoreWrite(CJ_WORD i, CJ_WORD value)*, allows writing the value *value* in the semaphore n-th. The writing of the semaphore is allowed on condition that one fo these conditions is true:

1. The semaphore is true, or contains value 0.
2. The value *value* to write is value 0.

A semaphore can be written again as long as it is free, writing 0.

Another use of the semaphores can be for debugging, or it is possible to use the semaphore to monitor the value of a particolar variable into the algorithms; during the debug of the application it is possible to show the list of the 10 semaphores using menu *View/reView Semaphores Window*.

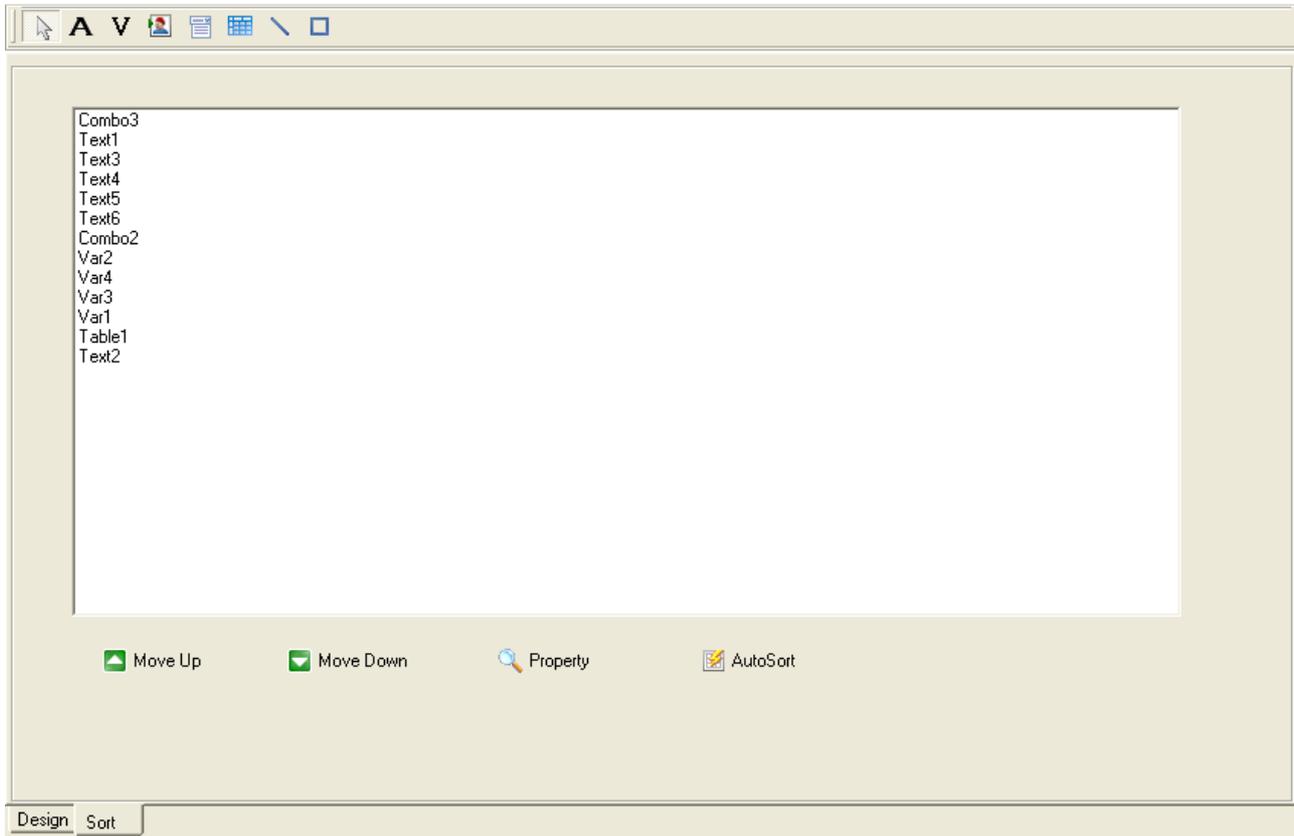Example of writing semaphores through algorithms.

Window of semaphores shown during debug.

Semaphore	Value
Semaphore_0	5
Semaphore_1	10
Semaphore_2	456
Semaphore_3	0
Semaphore_4	341
Semaphore_5	0
Semaphore_6	0
Semaphore_7	0
Semaphore_8	0
Semaphore_9	0

4.9 Sorting EIML elements

The elements added to an EIML page (texts, variables, icons, combos, tables, lines, and rectangles) are coded inside the page in the same order as they were entered by the developer when the page was created. When the page is loaded by the user interface panel, the elements in the page will appear in the same order as they were entered, which may not produce the desired optical effect. To customize the sorting order of the various graphic elements, and therefore the order in which they will appear in the page, click the *Sort* tab and open the following window:

This window lists the names of the various graphic objects embedded in the page, in the order in which they will be processed when the page is displayed on a user interface. Select one or more objects and drag them with the mouse to change the order in which they will appear. Otherwise, you may use the *Move Up* and *Move Down* buttons to move the element up or down the list.
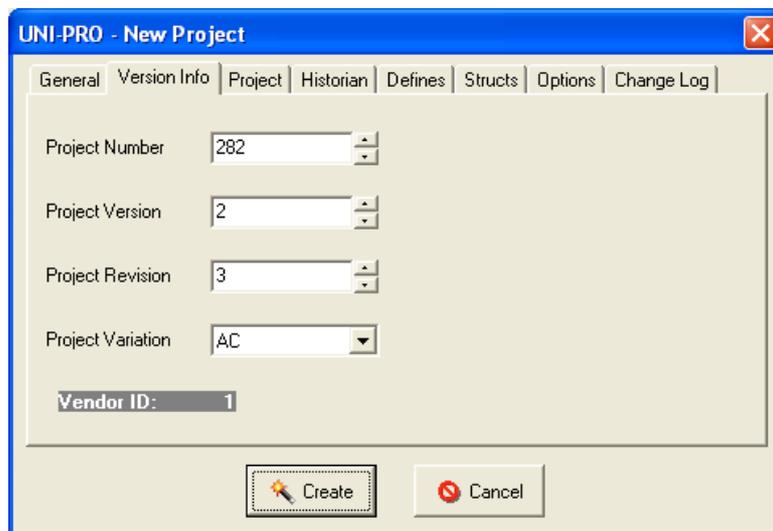
A very helpful utility function is the *AutoSort* function, that sorts the elements automatically depending on on their position in the page. They will be sorted starting from the element positioned on the top left side of the page and finishing with the element on the bottom right side.

4.10 Project settings

In addition to general data, such as name, author, creation date, project and version number, each project contains a certain number of settings that characterize the way it operates, such as serial protocols, availability of an Event Historian, and "defines" used in algorithms. To access these settings, choose *Project/Property...* from the menu or click and open the project properties window.

4.10.1 Version Info

In the Version Info window, you can set the project number, version, revision and variation for the user's project.

In the window there is the user's *Vendor ID* that is read-only, instead the modifiable information are the following:

Project Number:	Project number.
Project Version:	Project version.
Project Revision:	Project revision.
Project Variation:	Project variation. When two projects are equals but they are different for the EIML pages language or for the hardware could not be necessary singularize they with different number or version. It could be useful singularize they only with a variation so the projects target data will be the same.

This information is always modifiable.

4.10.2   Project

In this window it is possible to set a Project Code and the protection through password for the download of the project of the controller.

All the values shown in read only modality are obtained from the selected hardware and are required for its identification during the program download.

Project Code:	Project code
HW Type:	Hardware type
BMS:	Indicates the presence of a BMS protocol on the controller
HW Family:	Hardware family from which the controller comes from
HW Code:	Hardware ID of the controller
Enable Protection:	Enables the protection through password
Password:	Protection password for the program download

The protection through password is necessary to protect the application in order to download it only in particolar hardwares that have been previously programmed with the same password.
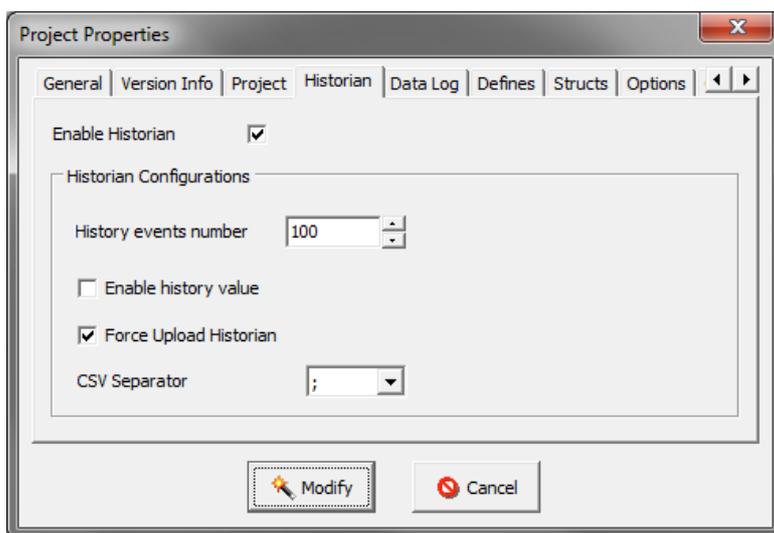
To program one or more hardwares in order that they accept application softwares only with that password, it is necessary to make two programmations at least, following this procedure:

- **First programmation**. Set *Enable Protection=NO* and digit the *Password* you want. Compile and make the download in the controller.
- **Second and following programmatins**. Set *Enable Protection=SI* and digit the *Password* you want. Compile and make the download in the controller. From this moment the controller will accept to be programmed only with application softwares containing that password (in the example, 56).

To make the controller free of the protection and make always the download, keep not selected (*Enable Protection=NO)*.

4.10.3 Historian

In the Historian window, you can enable and configure the event&alarm recording functions available in certain UNI-PRO controllers.

Enabling this function will cause the Historian Library to be linked to the controller's functions. If you do not wish to implement the event logging function, leave the *Enable Historian* check box unchecked.

If you do check it, then you will need to set the maximum number of events to be recorded in the memory in the *Events Number* field. The Event Historian queue is FIFO (First In First Out). For example, if you enter 30, the 30 most recent events will be stored in the controller's memory, and when that number is exceeded the oldest events will be overwritten.

Enabling the "*Enable History Value*" property is possible also store a value associated to the event of historical.

Disabling the "*Force Upload Historian*" property the historian will be retained in memory even after the new download of the program, otherwise it is always deleted.

The CSV Separator property allows you to customize the separator character in the CSV file generated by the controller (this property is unique and common with that for the Data Log).
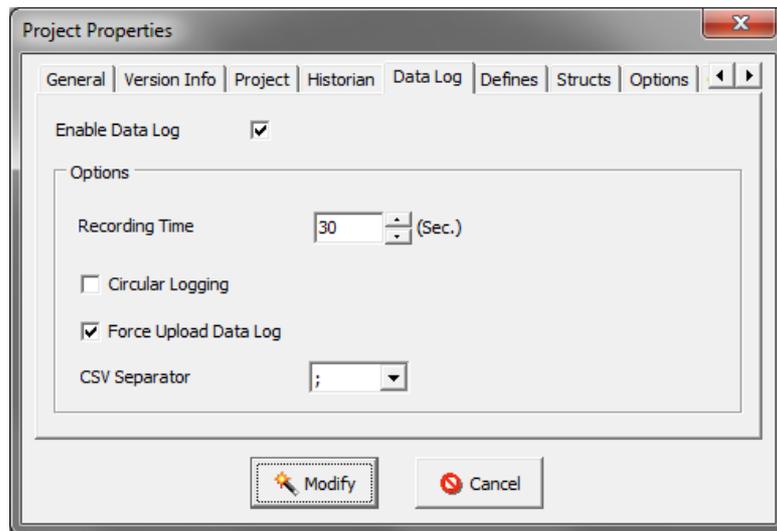
The following information is stored in the Event Historian:

Data	Type	Description
Date	CJ_DATETIME	Date and time when the event took place.
Code	CJ_WORD	Event identification code.
Progressive	CJ_LONG	Progressive event number.
Value	CJ_SHORT	Possible value associated to the event

To add an event to the historian, use the *HistoryWriteEvent* system library, to read an event from the historian use the *HistoryReadEvent* library (refer to document **UNI-PRO: Standard Libraries**).

4.10.4 Data Log

In the Data Log window, you can enable and configure the functionality for the project data logging available in certain UNI-PRO programmable controllers.

Enabling this function will cause the Data Log Library to be linked to the controller's functions. If you do not wish to implement the data logging function, leave the *Enable Data Log* check box unchecked.

Modifing the "*Recording Time*" value you set the logging time (in seconds) and conseguently and accordingly it determines the maximum number of recordable events in memory.

Enabling the property "*Circular Logging*" is enabled circular logging in memory, or when the dedicated memory is exceeded recording overwrites the oldest data. If the property is disabled, recording is linear and therefore when the memory becomes full is no longer recorded anything.

Disabling the "*Force Upload Data Log*" property the recorded data will be retained in memory even after the new download of the program, otherwise it is always deleted.
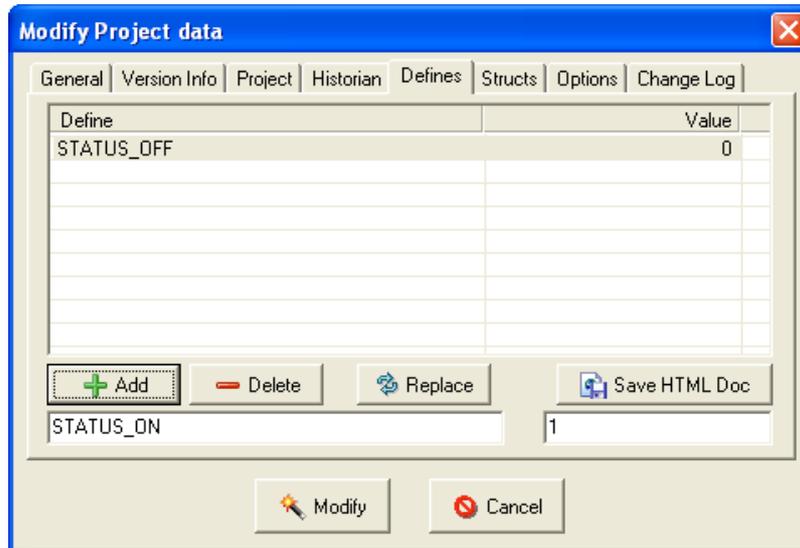
The CSV Separator property allows you to customize the separator character in the CSV file generated by the controller (this property is unique and common with that for the Historian).

To determine which entities to record, use the *Data Log* tool from the *Tools* menu.

The activation, deactivation, cancellation and other features of the Data Log can be configured through the system libraries (*Library\System\Data Log*) in the standard libraries of UNI-PRO (see document ***UNI-PRO: Librerie Standard***).

4.10.5 Project defines

The project defines window allows you to define constants and assign values to them so that you can use them several times in the algorithms that make up your project.
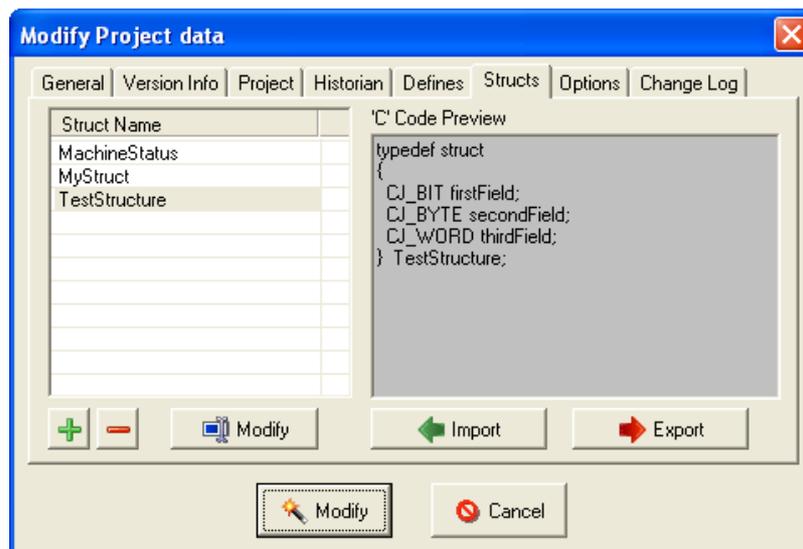

To add a define, type a unique name in the left box under the list and press *Add*. Doing so will cause the define to be added to the list. If you wish to assign a certain value to it, type it in the right box under the list.

To change a define, select it from the list, change it as required, and then press *Replace* to confirm. To delete it, press *Delete*.

4.10.6 Structs

In the window about the project structs, it is possible to decide the structures, in order to use them many times in the project into the algorithms. Each *Struct* must have an univocal name.

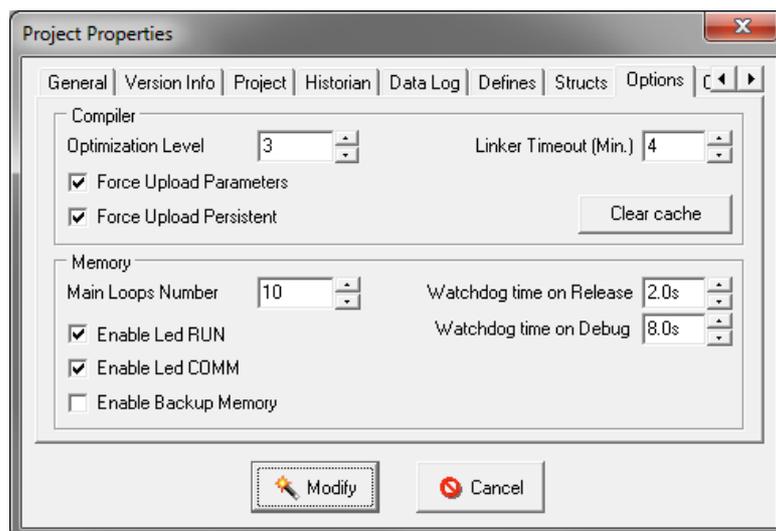In the box on the right there is the "C" version of the structure.

Action "Export", allows creating a file with all the structures that have been created for the project. Action "Import", allows importing a file containing others exported structures (through action "Export") from another project. Possible structures having an existing name will not be imported. Action "+" and "Modify" allow respectively to add and modify the structs, operating on the following window.

In the window it is possible to modify the name of the *Struct*, replace, add and cancel some structure fields. It is necessary each field ha san univocal name in the structure.

4.10.7 Options

Using the optimization window located in the project options, it is possible to define some parameters to optimize the program execution. Only expert users are advised to modify these options.

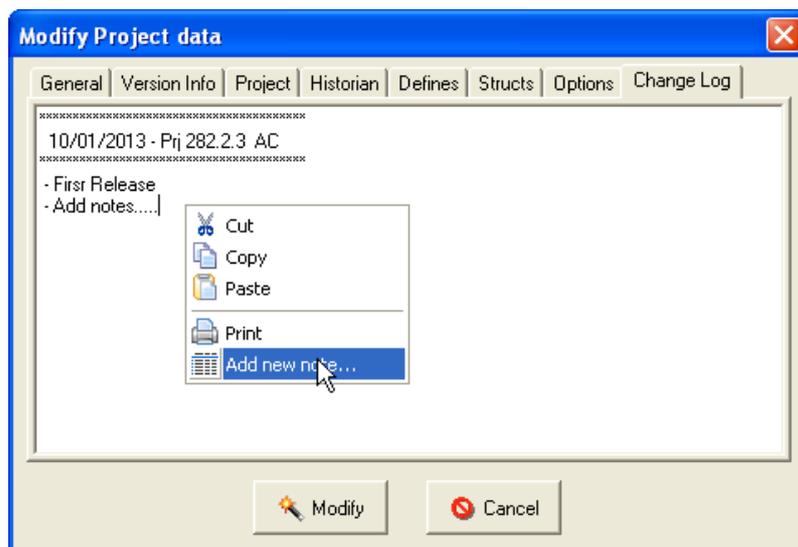| Optimizations Level:	Optimization level of the compiler.
Force upload	Forces the rewriting of the parameters in E2 during the initial execution after

parameters :	downloading.
Force upload persistent :	Forces the rewriting of the persistent in E2 during the initial execution after downloading.
Linker Timeout:	Waiting time (minutes) for completion of the linking process after compilation.For very large projects, you may need to increase this time.
Clear Cache:	It cancels the compiling cache of the projects
Main Loops Number:	Using this configuration it is possible to specify the number of *actions* calculated in a group. For very large projects, by increasing this number it is possible to speed up the main program.
Watchdog time on Release:	Watchdog time (*) for the projects compiled in Release (without Debugger protocol)
Watchdog time on Debug:	Watchdog time (*) for the project compiled in Debug (with Debugger protocol)
Enable Led RUN:	Enable/disable the management for the controller RUN led (if disabled it always remains OFF).
Enable Led COMM:	Enable/disable the management for the controller CAN led (if disabled it always remains OFF).
Enable Backup Memory:	It allows you to enable or inhibit the copying of the configuration and application parameters to E2 memory using system commands.

(*) Watchdog time = time required for the algorithms to complete the execution of the code; when the time elapses, the controller makes a hardware reset. This is a safety system to avoid endless loops or very long elaborations which would jeopardize the proper operation of the controller.

4.10.8 Project Change Log

To help the user keep track of changes made to the various projects, the development environment provides a history of the projects (Change Log). In this window it is possible to record modifications introduced and any other notes for each version.

Clicking with the right mouse button on the window it is possible to view the corresponding menu that, in addition to the classic Cut/Copy/Paste actions, allows:
  ➢ printing of the Change Log's contents
  ➢ adding, in the current cursor position, a new note (a few rows with headings containing the date, and the current project version are also added).

4.11 Data Log

From menù *Tools->Data Log* it is possible to gain access to a frame allowing selecting which entities to export and to configure for the log of the project.

For each entity exported, you can configure these properties:
- *Descrition*. Maximum 24 characters.
- *Unit of measurement*. Maximum 6 characters.
- *Type*. Registration type of the entity.
  - *TIME*. The registration is performed by time, using the logging time parameters.
  - *EVENT*. The recording is performed when the value change.
  - *ALL*. It includes both functionality TIME and EVENT.
- *Variation*. It determines when to register an EVENT type entity.
  - *ALL*. All the variaions
  - *0->1*. Variation from 0 to 1 (major of 0)
  - *1->0*. Variation from 1 to 0 (minor of 1)
- *Enable Log*. It determines if the exported entity is enabled to log by default or not.

You can configure a maximum of 32 TIME entities and 32 EVENT entities. If an entity is exported and it is of type ALL (TIME and EVENT), it occupies a place in both lists.

To add entities in the exported list just drag in the right list, the deletion may take place by dragging in the opposite direction, or using the *Delete* function associated with the *Del* key.

To change the properties of the entity exported, select it and edit the new properties in the Entity Log Options area and confirm with *Set button*, or use the drop-down menu that appears by clicking on the selection with the right button.

In the *Data Log Info* section are summarized some information including the time within which you are going to run out of recording space. You can also change he recording time here, to lengthen or shorten the total time of the log.

Through the *Report* it is generated an html documentation summarizing the configurations made.

For the management of the Data Log from the application you can refer to libraries in *System/Data Log*, with which you can:
- enable/disable logging data.
- enable/disable the recording of the individual entities configured.
- configure the recorcing time.
- extract the data from the individual records for printing on display.
- create a .CSV file on the USB port with all the recordings.
- check the status of memory dedicated to the Log.

4.12 Font custom per EIML

In EIML color pages, you can use custom fonts instead of the standard page font. All UNI-PRO fonts are loaded at program startup and via a configurator accessible from Tools-> EIML Fonts, you can select and edit which ones to use in the project.
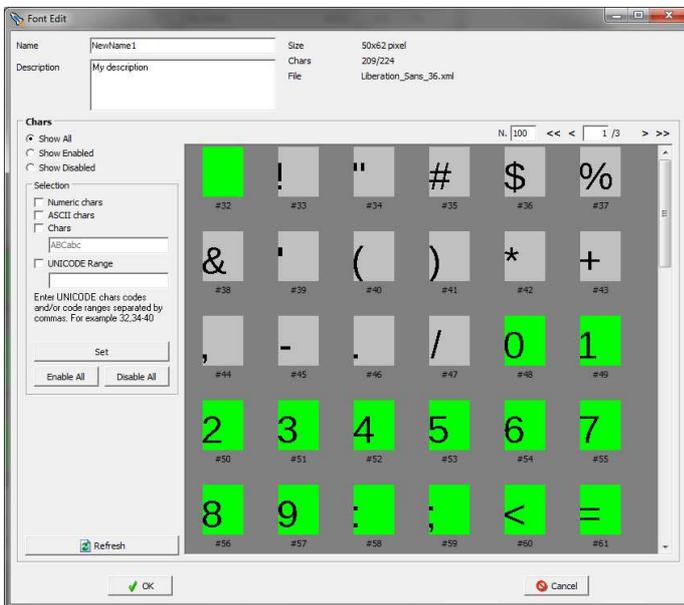

You can decide to use an entire font as it is, adding it to the project with the *Add* button

Add & Modify button allows you to add a new font by modifying it directly choosing only a few characters.

Modifying a project font can still be done later by using the *Modify* command from the pull-down menu.

Choosing a subset of characters may be necessary if you have insufficient memory problems when compiling your project, this is done through this window (from the *Modify* command):

There are a few options to make the character selection easier, otherwise you can manually select by enabling / disabling them with the F3 key.

An enabled character has the green background while a disabled character has the gray background.

Disabled characters will not be available on EIML pages.

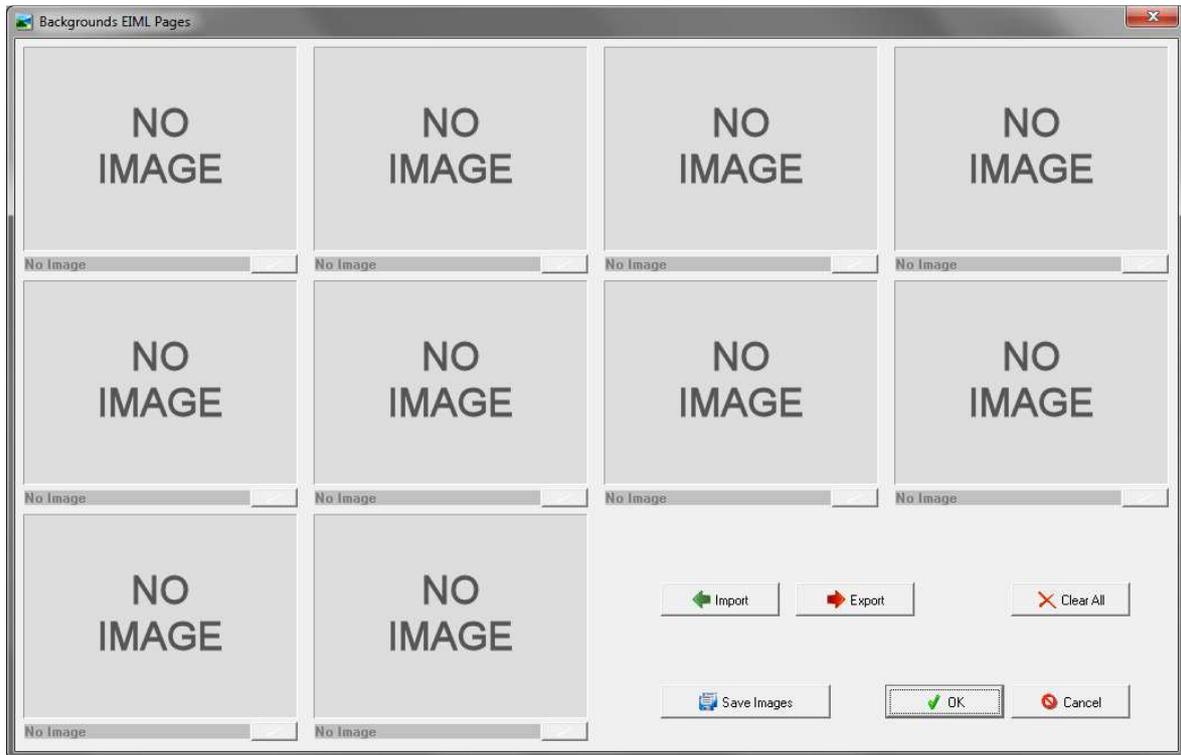### 4.12.1 Saving of the selected fonts

The project fonts used, originals or modified, will be saved as XML driver files with the project, they will be saved each time you save the project and will be stored at the same level of ucjp project file (eg "ProjectName_X.ucjp") in an ad-hoc created folder (eg "**Fonts**_ProjectName_X*.xml").

The name of the proposal drivers fiel as well as the name of the folder files should not be changed, if you rename the .ucjp project file, also need to be renamed the folder at the same way (do not change the .xml file names instead).

At the opening of the project, UNI-PRO will load the font from their driver file, if the file is not present an error is reported. To avoid unwatend data loss, if an error occurs during the driver loading close the project without saving and reposition the xml driver file in the specified search path.

4.13 EIML Backgrounds

From menù *Tools->EIML Backgrounds* it is possible to gain access to a frame allowing selecting up to 10 backgrounds for the EIML color pages of the project.

Ti insert/modify a background double click on the picture and select a bitmap having size 320x240 pixel. Once the image is loaded a preview will be shown and it will take a name similar to that of the loaded bitmap, in order to make easier to recognize it.
If the bitmap to be loaded is not made of the 256 colors of the palette, it will be necessary to wait some seconds it is converted (the image could be a little bit different from the original).

The *Import* and *Export* button allows to import and export the backgrounds with a XML file, that is portable for other UNI-PRO projects.

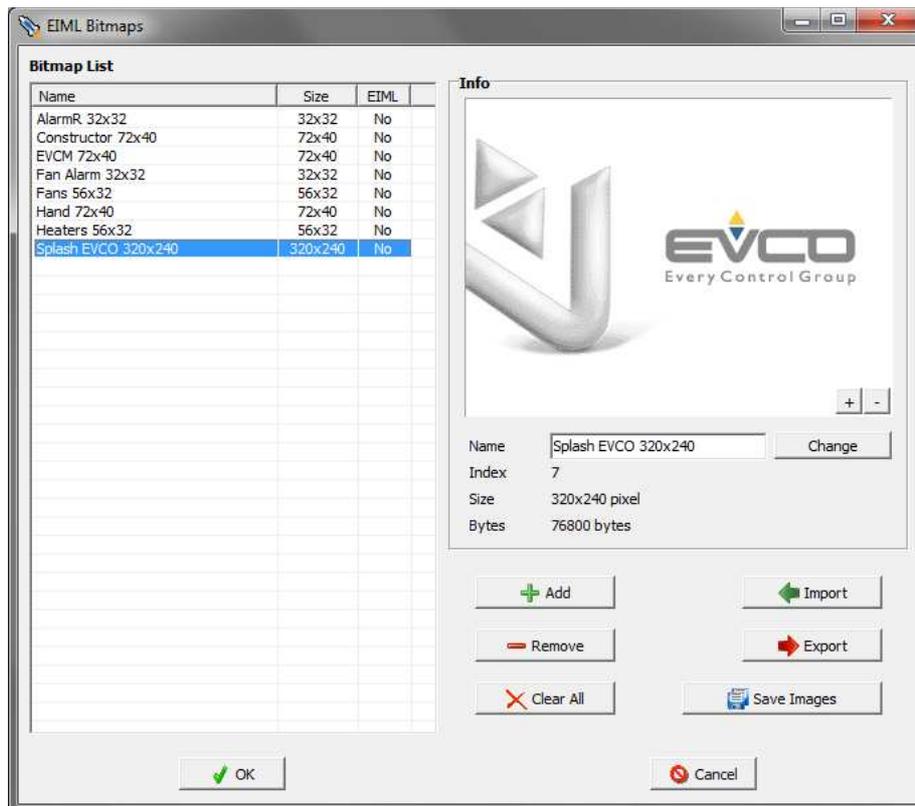To erase a background click on the button X in the bottom right hand corner of the image.
With the *Clear All* button it is possible clear all backgrounds in one shot.

It is possible to save each image right clicking the mouse on the image and selecting "*Save Image*".
The button *Save Images* is usefull to save all the images into project folder.

4.14 EIML Bitmaps

From the *Tools-> EIML Bitmaps* menu you access a window that allows you to select the icons that can be used in the project, all the bitmap included in this list can be used, even more, in the EIML pages that support this feature.

Using icons from this list saves some code in the controller flash; in fact, the page will no longer allocate the space for the icon but only a reference to it, it will be a task of the controller to recognize this reference and to draw the associated bitmap. Therefore, if a given icon is to be used multiple times in the project, it will not be necessary to insert it several times in the pages, but it will be enough to insert it in the list above and then indicate it in the "Bitmap" property of the icon type elements in the EIML pages.

If the bitmap to be loaded is not made of the 256 colors of the palette, it will be necessary to wait a few it is converted. The image converted could be a little bit different from the original.

Once selected the image it is possible change it own own name by *Change* button.
With the *Clear All* button it is possible clear all bitmap in one shot.
The *Import* and *Export* button allows to import and export the bitmap with a XML file, that is portable for other UNI-PRO projects.
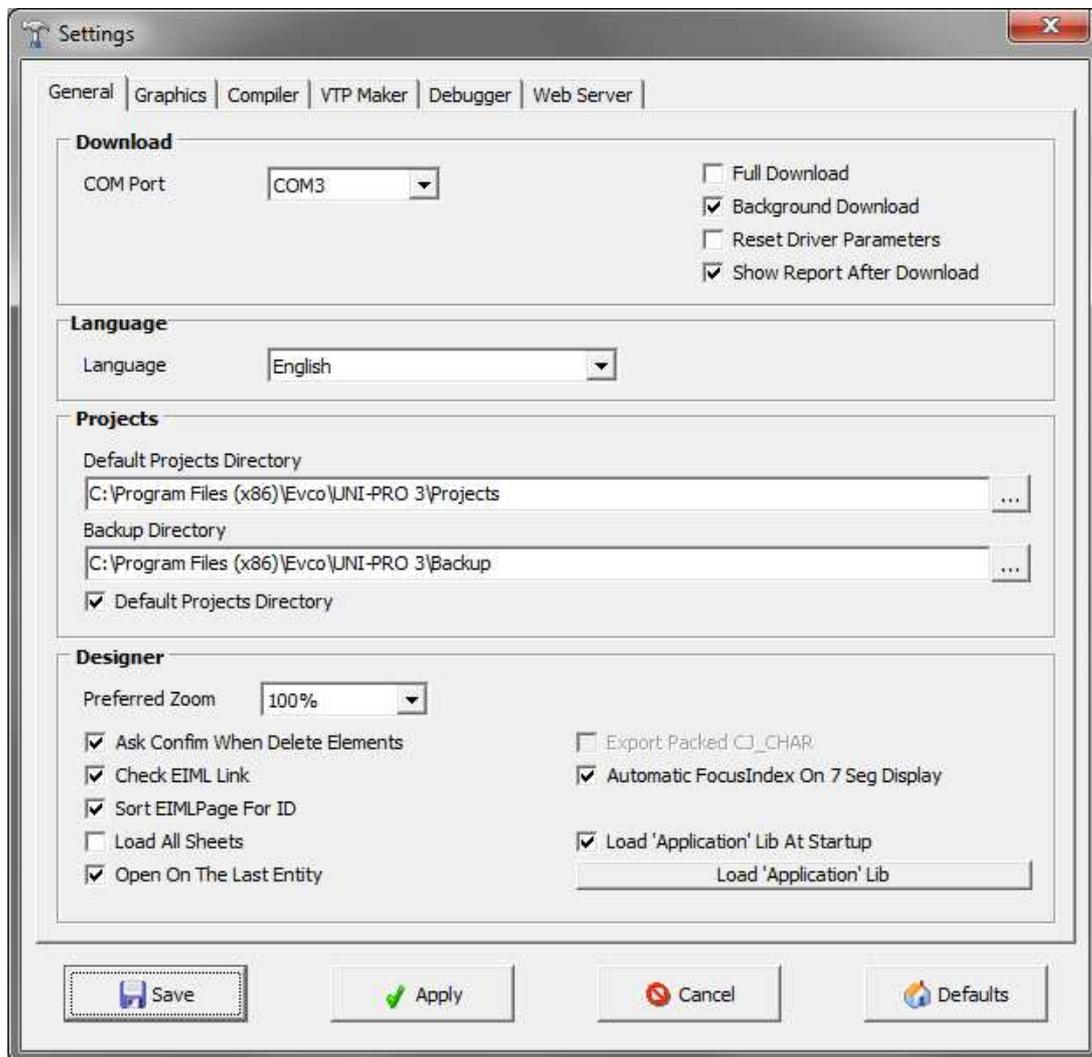
It is possible to save each image right clicking the mouse on the image and selecting "*Save Image*".
The button *Save Images* is usefull to save all the images into project folder.

4.15 Environment configuration

The UNI-PRO development environment is very flexible since it can easily be configured as desired by the user. Activating the *Tools / Settings* menu it is possible to gain access to the following configuration screens:

> General
> Graphics
> Compiler
> VTP Maker
> Debugger
> Web Server

The *General* screen contains the *Download, Language, Projects* and *Designer* sections. All the options of which will be analyzed here below:

Default COM port :	Serial port used for downloading operations. If the connected controllers are c-pro 3 micrio/kilo the COM porti s automatically recognized and the value you have set is ignored
Reset Driver Parameters:	It forces the restoring of the parameters about the driver of the

	controller after the download of the application software.
Full download :	Forces the full downloading of the entire executable file. To speed up the downloading options, it is possible to unselect this option and, after having correctly deleted the flash memory present in the microcontroller, proceed to download only the flash memory used.

Background download:	Allows to operate on the project during the download phase.
Show Report After Download:	Ifi t is selected, after each download in the controllers c-pro 3 micro/kilo a report file will be shown.
Language:	It modifies the language of the development environment. It is necessary to run UNI-PRO again to apply the modification.
Default projects directory :	Default directory prompted for saving project files.
Backup directory :	Directory used to backup the project files (refer to the *Project backup* paragraph).
Auto increment revision when backup project's files :	By activating this option, each time that backup operations are performed on project files, the revision number will automatically increased, otherwise the user will have to perform this operation.

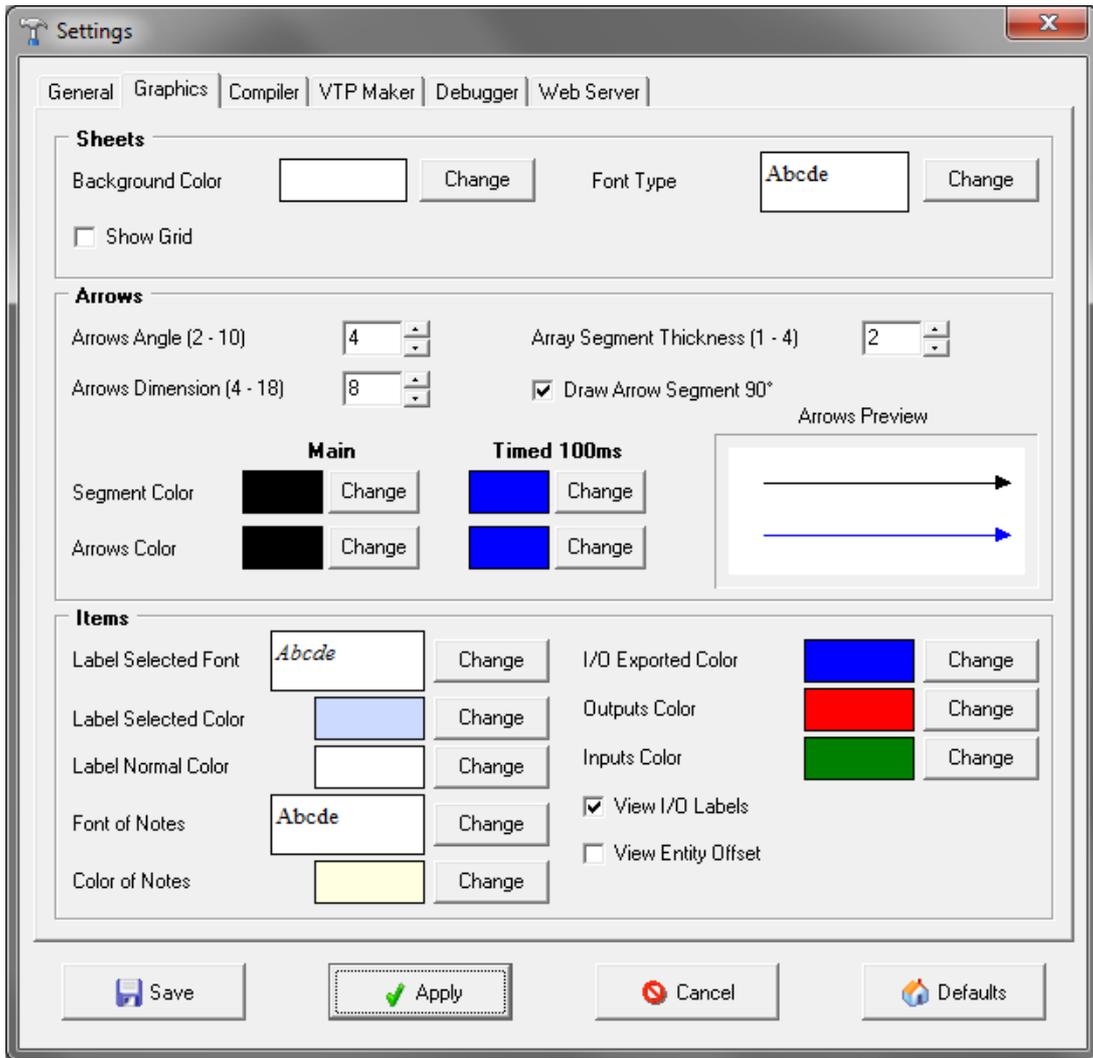|	
Ask confirm when delete elements :	Activates or deactivates the request to confirm the erasing of entities and the EIML components.
Preferred zoom :	Allows the specification of the zoom to be applied upon loading the project. Afterwards the zoom level can be changed easily using the shortcut menu located in the toolbar.
Check EIML link :	Checks that all the elements which can be linked with the entities present in the graphic interface section are properly connected. Check this during the graphic interface simulation phase.
Sort EIML Page for ID:	When enabled, it causes the EIML pages to be sorted according to the *Id* properties; otherwise they will be displayed in alphabetical order.
Load All Sheet:	When enabled, the project will be open completly When disabled the project will be open partially; only the EIML pages and main sheets are loaded.
Open On The Last Entity:	When enabled, the reopening of a recent project will be

	completed by selecting the entity on which the project is closed the previous time
Export Packed CJ_CHAR:	If it is enabled, the Modbus exportation of the string type (tipo CJ_CHAR) is done by groups (made of 2 Byte for each register).
Automatic FocusIndex On7 Seg Display:	If enabled, activate the automatic management of focu-index for pages with 7-Segment Display
Load 'Application' Libs at Startup	If enabled, load the Application libraries when starting UNI-PRO. With the **Load 'Application' Lib button**, you can load the Application libraries at any time, without changing the property value.

4.15.1 "Load All Sheet" option

When this option is enabled , it falls in the normal use of UNI- PRO , that the project is fully open. In this mode , however, the memory usage is very intense and the opening of the project , depending on the size , It may take several minutes to complete. To lighten the environment is recommended to keep this option turned off, so the memory used is less and also the opening of the project is much faster, it will be loaded only pages EIML and the main sheets. Access to the entity is not yet loaded graphically can be done either manually, browsing and opening the sheets needed, or automatically, when you try to access the entity in the manner provided by UNI-PRO.
The opening step-by-step sheet takes a long time much less than the time it would take to open the whole project, once a sheet is opened it remains open throughout the session for the project.

Using the option*"Open On The Last Entity"*, (when de opening is completed), you can activate the sheet on which you were working at the last time when you close the current project.

The *Graphics* screen contains the *Sheets, Arrows and Items* sections, which options will next be fully described:

Sheets background color :	Defines the background color of the project sheets.
Sheets Default font :	Defines the font for all components existing in the project sheets.
Show Grid :	Show a graphic grid in the work sheets; if enabled is also possible to modify the grid colour.

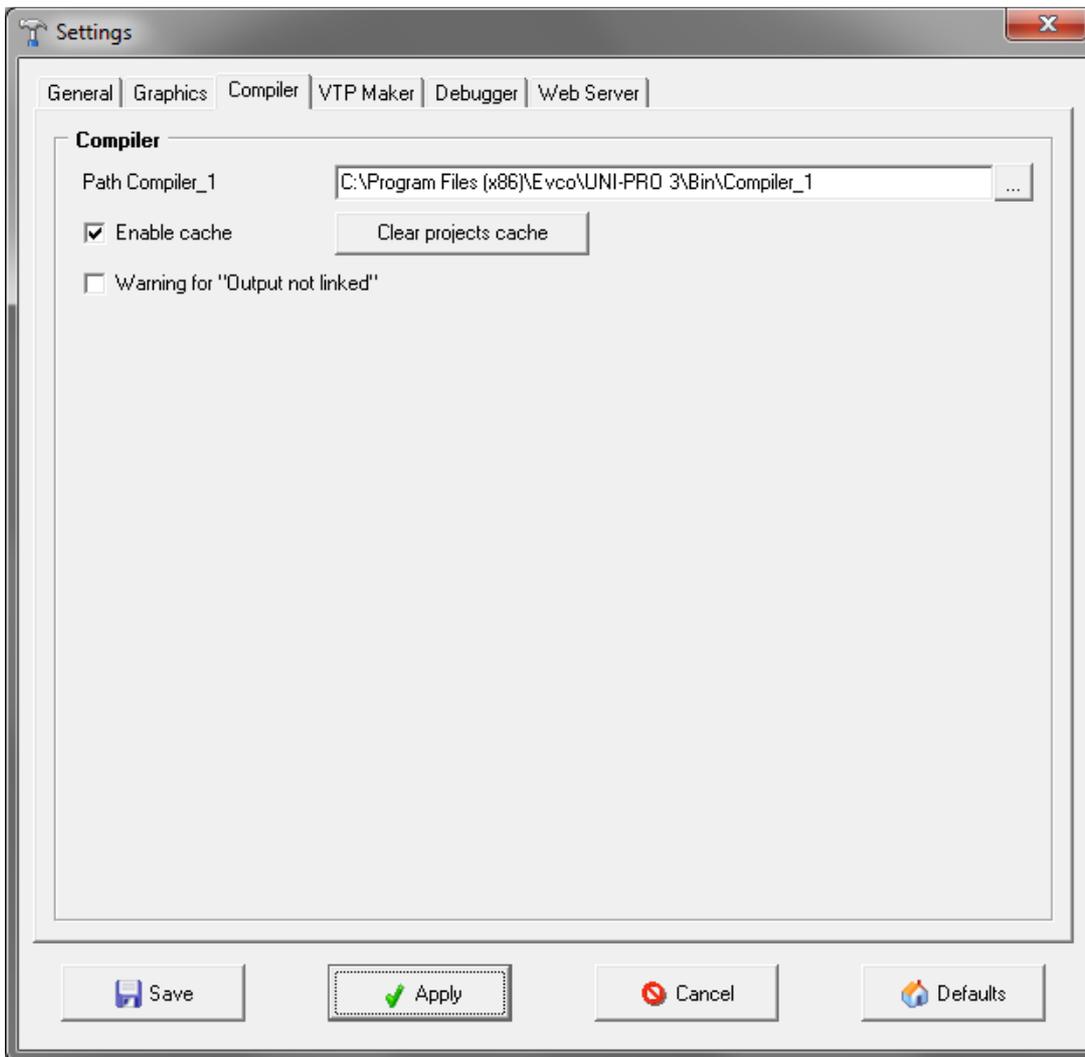|	
Arrows angle :	Modifies the aperture angle of the arrow tip connected with the entity.
Arrows dimension :	Modifies the dimension of the arrow tip connected with the entity.
Array Segment thickness :	Modifies the thickness of the lines connected with the array entity (entities with the array property set greater

	than one)
Draw Arrow Segment 90°:	If enabled, it forces the use of 90° segments on arrows drawing
Segment color and Arrows color :	Using these options it is possible to modify the color of the arrows. For both options it is necessary to specify the color to be used if the path is to be calculated during the main task or every 100 ms (refer to the **Defining the execution task** paragraph).
Preview :	Preview window where examples of arrows settings are viewed.

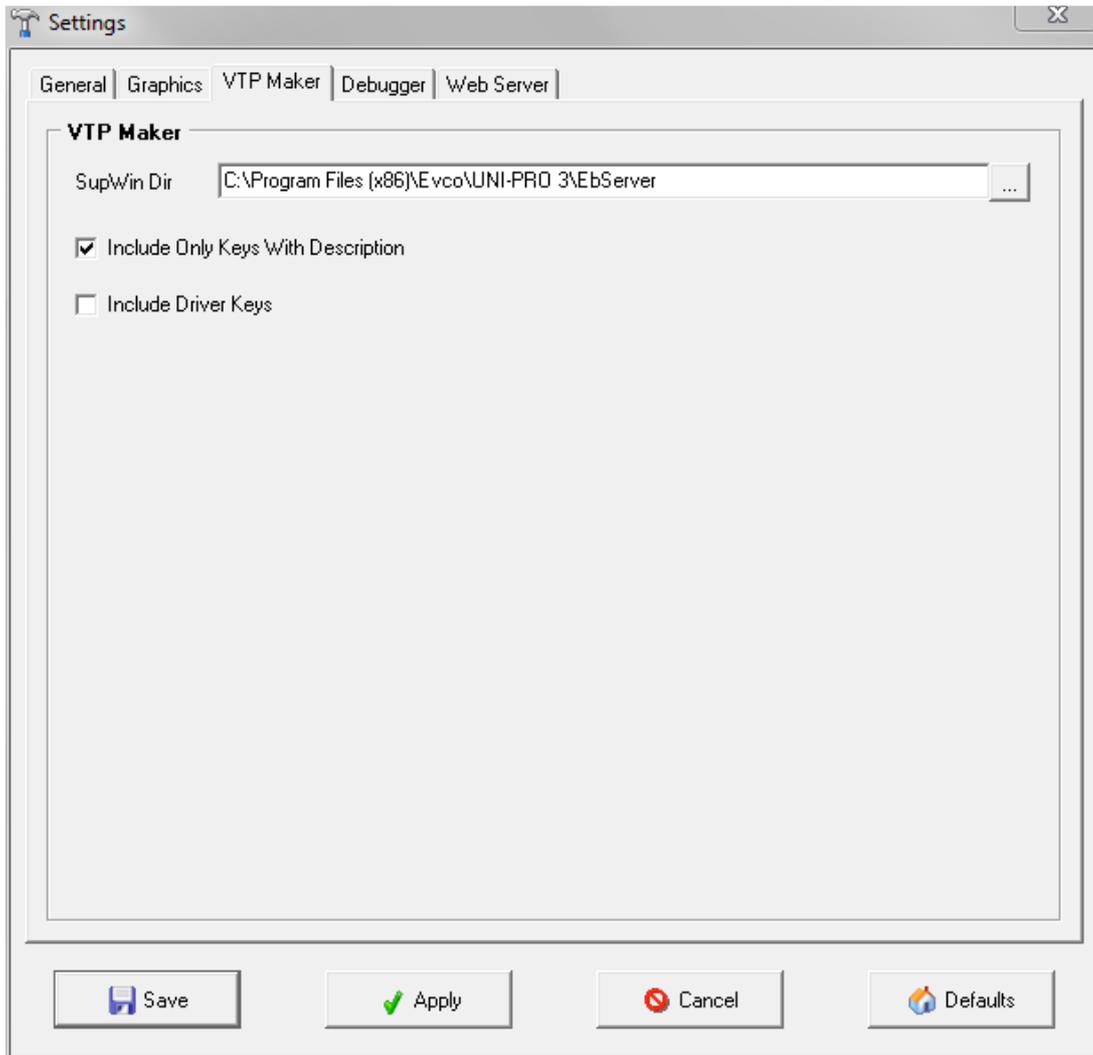|	
Label selected font :	Defines the font to be applied to the entities selected.
Label selected color :	Defines the background color of the text for the entities selected.
Label normal color :	Defines the background color of the text for entities not selected.
Font of Notes :	Defines the font to be applied to the Note type entities.
Color of Notes :	Defines the background color of the text for the Note type entities.
I/O exported color :	Defines the color to be attributed to the inputs/outputs of the entities exported in the subsheets which they contain.
Outputs color :	Defines the color to be attributed to the output of the entities.
Inputs color :	Defines the color to be attributed to the inputs of the entities.
View I/O Labels:	Show/Hide the label with the name of the clamp for the sheet, algorithm, library,…
View Entity Offset:	If enabled, allow to display the offset property below the entity.

The *Compiler* screen contains the configuration relative to the C compilers.

Options are the following:

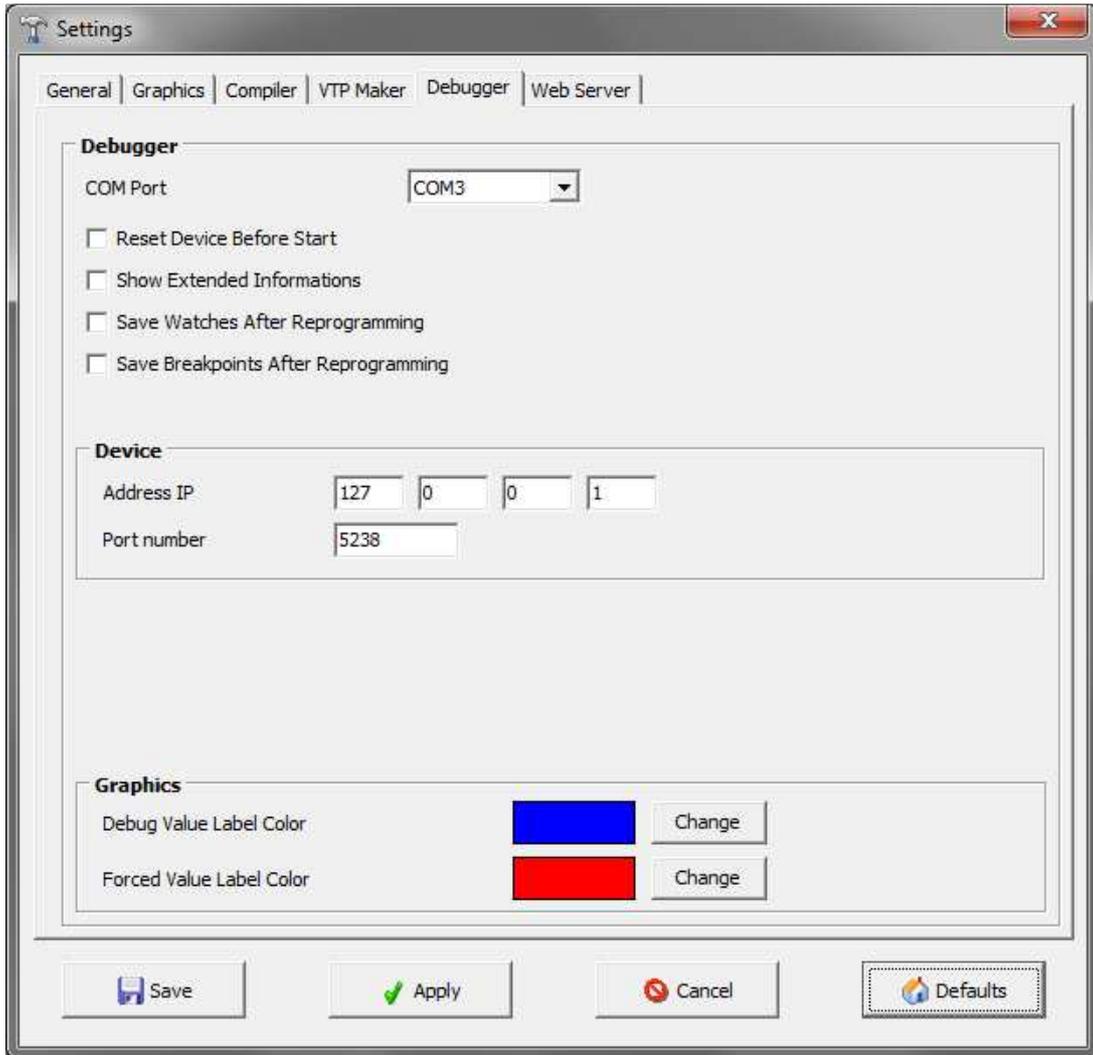|	
Path Compiler_1:	Specify the complete path where are located the source codes of Compiler_1 for projects compilation.
Enable cache:	It enables the use of the compiling Cache.
Clear projects cache:	It cancels all the Cache of the projects.
Warning "Output not linked":	Enable/Disable the compilation warning when there are algorithms not elaborated (because they have output linked to nothing)

The *VTP Maker* screen contains the configuration options used by VTP Maker tool to create VTP driver file required to communicate with EVCO software.

Options are the following:

SupWin Dir :	Defines the destination directory to the generated VTP Maker files. Default is: *..\UNI-PRO 3\EbServer*
Include Only Keys With Description :	Used to the dedicated Evcobus protocol. Generates VTP keys driver only for entities with description property set.
Include Driver Keys :	If selected include on VTP file driver the used controller hardware driver.

The *Debugger* screen contains the configuration options used to debug tool, for example that required to the comunication.
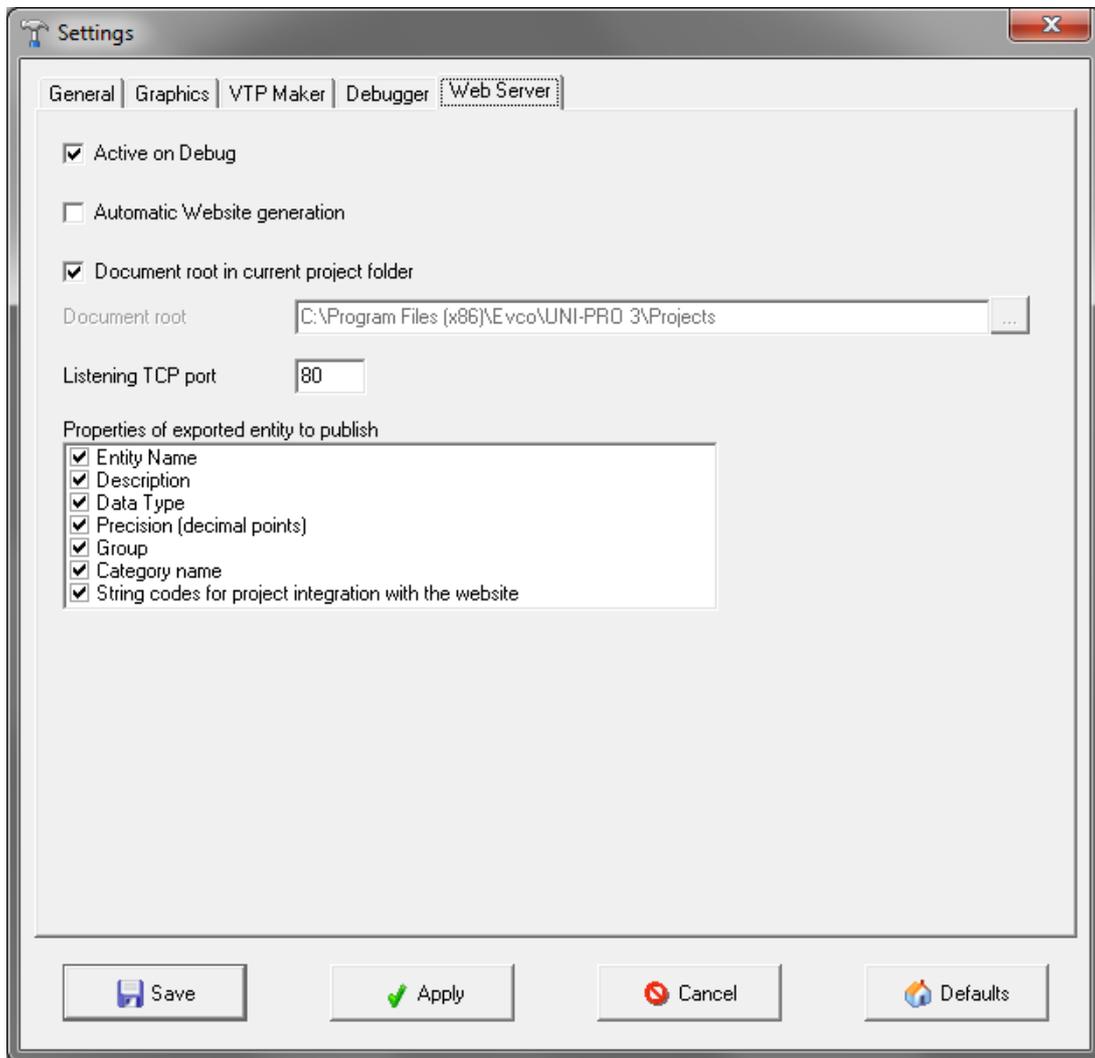

Options are the following:

COM Port :	Serial port used to the comunication with the controller during debug operation. If the controller has only the programmino port and not the debugging one, this property must coincide withe the COM port you have selected for the download. If the connected controllers are c-pro 3 micro/kilo the COM por twill automatically be recognized and the value you have set will be ignored.
Reset device before start:	Reset the controller at each start of the debugger
Show Extended Informations :	If selected allow to show more extended info on the state bar application, as task index where program break for a breakpoint setting
IP Address:	IP address of the connected controller to use Debugger vai Ethernet.

Port number:	TCP port dedicated for the Debugger via Ethernet.
Show watches after reprogramming:	Saved and restore watches after a controller reprogramming.
Show breakpoints after reprogramming:	Saved and restore breakpoints after a controller reprogramming.
Debug Value Label Color:	Color of the debugger values under the entity
Forced Value Label Color:	Color of the forced values by debugger (for Analog Input and Digital Input)

Nella schermata *Web Server* sono presenti le opzioni per le operazioni di simulazione del web server quando il debug è attivo.

Options are the following:

Active On Debug :	It determines whether to use the simulator for Web Server when you run the Debugger. If you deselected the Web Server simulator is not active.

Automatic Website Generation:	When activated, generates and automatically copy all files to test standard website.
Documet root in current project folder:	When activated, generates the root folder of the website into the current project folder (at the same level of "\exe" folder).
Documet root:	It specifies the directory from which expands the tree of the website. It is enabled only if the property *"Documet root in current project folder"* is disabled.
Listening TCP port:	Listening TCP port for the Web Server simulator.
Properties of exported entity to publish:	It determines which properties of entities to export to the Web Server will be generated and managed on the website.

4.16 Exporting / Importing

Using the import and export menus, it is possible to copy an EIML page from one project to another. From menu *File->Export->Export EIML Pages* it is possible to choose (among the pages of the project) those to export, creating a file that can be imported in a second moment in an UNI-PRO project. From menù *Edit->Remove EIML Pages* it is possible to show the same box with the same filters, to cancel pages (through their selection).

From the menu *File->Import->Import EIML Languages* and *File->Export->Export EIML Languages* it is possible to import and export a CSV multilanguage file containing all the multilanguage strings of the EIML pages. The saved files can be elaborated with the tool *EIML Translator*.

4.17 Export of internal variables of libraries/macroblocks

All entities of type VAR, PAR, PERS and FIX closed in libraries and macroblocks can be made public even if they are normally inaccessible, in order to export these entities it is necessary to activate the property "*Show Out LIB*".

To use this feature, before creating the library, select the entities that you want to "export" and set the "Show Out LIB" property for each one, then create the new library. From now on, when the new library will be used, the "public" entities will be visible in the Entities List of the tools and in the EIML pages, so it will be possible to modify some properties (name, description, default value, ...) and link them in the EIML pages and in the tools of the network protocols (datalog, modbus, bacnet, ...).

If you open the contextual menu on a library that has exported entities and if you click the "*Show Entities* ..." item, a list of all public entities of the library is opened, by double-clicking on the elements the relevant properties window is opened. The non-editable properties are presented as read-only.
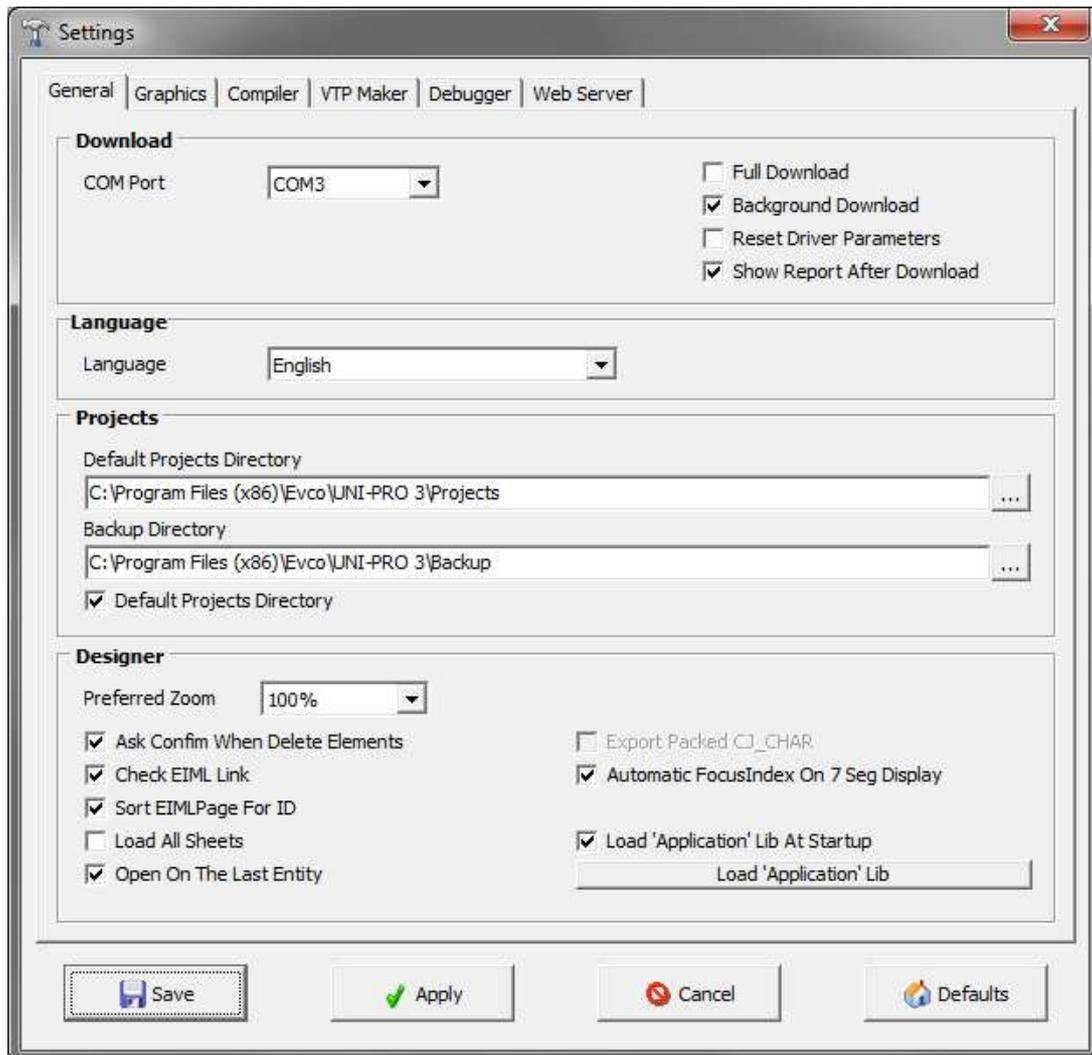
4.18 Project backup

The UNI-PRO development environment has an automatic backup function that allows you to copy the project files in the special directory shown in the environment options.

Setting some of the options located in the *Settings* window (using the *Tools/Settings* menu this last will appear) it is possible to backup to a preferred directory and choose whether to automatically increase the revision value once having performed the operation.
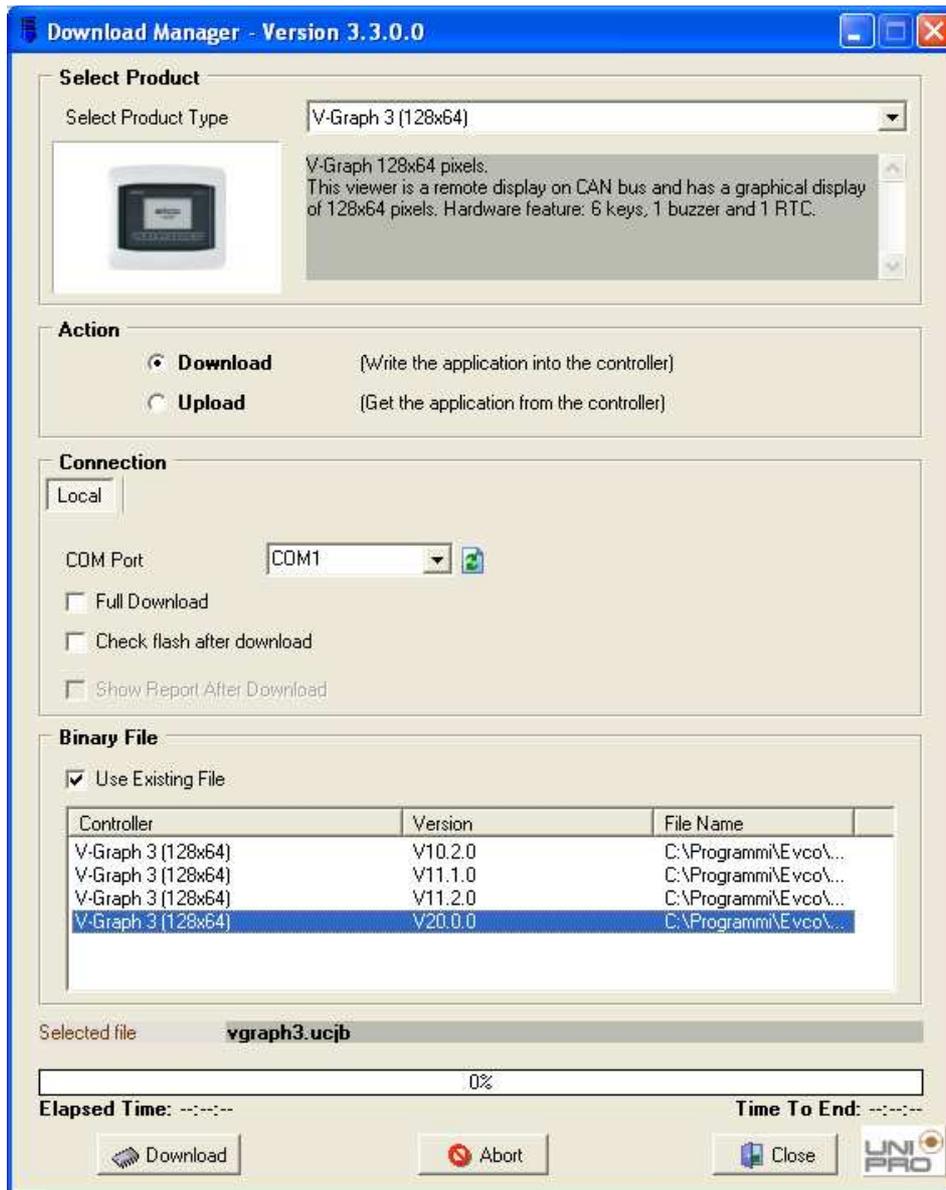
The directory where the backup is performed is the following:

<Default directory>\Prj_<project number >\Ver._<project version>.<project revision>

4.19 Product programming tool (Download Manager)

The development environment has a tool for programming and updating all the products (controllers and viewers). To view it, select the *Tools / Download products* menù.

Download Manager allows both the programmation of the controllers (download), and the writing and extraction of the firmware, in binary format, there is in the connected controller (upload).

The program is made of one window divided by the following four sections:
-   Select product: selection of the controller to program or from which it is destre to extract the firmware (this must coincide with the connected controller);
-   Action: select the action it is desired, or download (programming the device) or upload (extracting the firmware from the device);
-   Connection: selection of the serial port where the device is connected; if the controller is c-pro 3 micro/kilo the recognizing of the COM port is automatic and the value you have set is ignored.

- <u>Binary File</u>: in case of download it is possible to decide if using an existing firmware or specify the file to download; in case of upload it is possible to define the file name where to save the firmware read form the device.

Before beginning the download/upload phase it is necessary to verify the device to program is correctly connected to the serial port selected in the section *Connctions*.

In case of controllers c-pro 3 micro/kilo the files downloaded in the controller or to extract from the controller are always two: the application binary file (.ucjb) and the binary file of the EIML page (.ucje). In any case it is necessary to specify, in the program, only the application binary file (.ucjb); The Download Manager will also download the other file (in case of download) or extract (in case of upload). The file of the EIML pages has always the same name as that of the application (it just changes the extension).

At last, to start the download press button *Download*; press button *Upload* if the selected action is that of writing of the formware. In the lower part of the window it is possible to read the status of the process, in particular it is shown the elapsed time (Elapsed Time), the esteemed time (Time To End) and the download percentage (in the progression bar).

In case the download/upload does not happen correctly, verify the connection with the device, the adopted configurations and try again the operation.
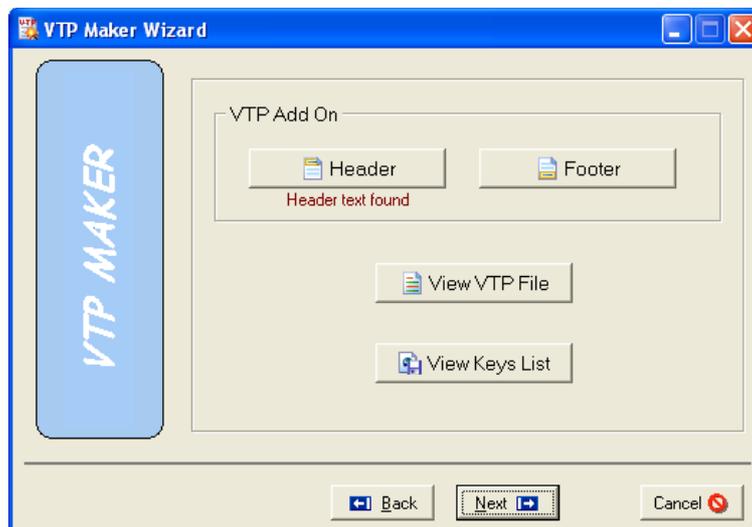
4.20 VTP Maker

The development environment has a tool that automatically generate the drivers file for the communication with supervision software *R.I.C.S.* and with parameters management software *Params Manager*. It's required that the interested entities will be exported on Modbus protocol using the *Export Entities* activable from *Tools/Export Entities* menu.
Select the *Tools / VTP Maker* menu.

Before beginning the creation of VTP file driver, make sure that are correctly set the options for the automatic maker from *Settings / VTP Maker* menu.

In the next window you can select three other functionality:

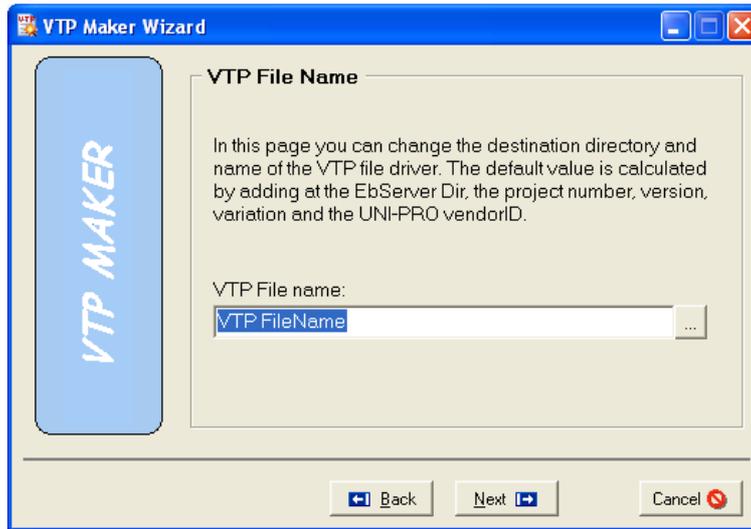|	
Header:	Open a text editor to insert a comment at the begin of VTP file. If header text exists the message "Header text found" will be shown below the button.
Footer :	Open a text editor to insert a comment on the end of VTP file. If footer text exists the message "Footer text found" will

	be shown below the button.
View VTP File :	It generates a simple VTP file preview, this is not manually modifiable.
View Keys List :	It generates and ask to save an HTML file with all the driver keys and some current project information.

Click button *Next >* to show a window with the path and the name where VTP driver file will be saved.

If *Parameters Manager* is installed, the generated VTP file is also copied in the directory of the *Parameters Manager* drivers. If the driver file is already present, it will be asked the user to make a copy or a manual comparison between the new and the existing one, in order that possible modifications of the driver in use are not overwritten.

Last showed window, less than errors during file generation, is that one indicated the successfully creation of VTP file driver.

At any time it is possible to click button *< Back* to modify your selection. Click button *Close* to finish the VTP Make wizard.

4.20.1 RICS property

Before to launch VTP Maker tool, it is possible to do some settings to adjust the driver properties during the VTP automatic generation. This project setting is possible using some keywords to write in the property *RICS*:

These keywords assign a different table representation than the default behaviour. For example the previous entity, that is a Var, should be put in the column STATUS with the name AI7_Pressure_HP_C2. After the assignment *GROUP=ANALOG_IN; DESC=High Pressure Circuit2;* and *Mu="bar";* ParametersManager will show it in the column I/O with the name High Pressure Circuit2 and the unit of measure *bar*.

> *R_key=kyAI7_Pressure_HP_C2;*
> *Group=ANALOG_IN;*
> *R_var=MODBUS_HLDREG; R_addr=15;*
> *R_for=SIGN; R_dec=1; R_do=RWW;*
> *R_inf=-32768; R_sup=32767;*
> *Desc="High Pressure Circuit2";*
> *Mu="bar";*
> *<<<*

The most important keywords are:

- **GROUP**=XXXX; →Changes the default group association.
- **DESC**=YYYY; →Changes the description (by default, the property *Name*) into the desired description.
- **Mu**="ZZZ"; → Set the unit of measure. For example Mu="°C";
- **DEC**=X; → Changes the position of the decimal point
- **INF**=XXX, **SUP**=YYY; → Change the inf and sup limits (fixed point representation)
- **EDIT**=RRR; → Protects the entity write (Read Only)

- **R_flag**="OFF", "ON"; → Allows the text representation OFF and ON instead of the numeric representation 0 and 1 respectively. In general, it permits the association among texts and numeric values.
- **hidden**=1; → Skips that record, that will be not shown in Parameters Manager

4.21 Parameters Manager & VTPEditor

It is possible, from the development environment UNI-PRO, to generate and on afterwards to modify (through the proper editor called *VTPEditor*), the VTP drivers used by the software *Parameters Manager*, to communicate with the controller using the Modbus communication protocol.

In order that the UNI-PRO can work properly with Parameters Manager, it is necessary that UNI-PRO finds the program installed; please note that even the installing file of Parameters Manager is present in the CD-ROM of UNI-PRO, Parameters Manager is not automatically installed when installing UNI-PRO but must be installed separately.
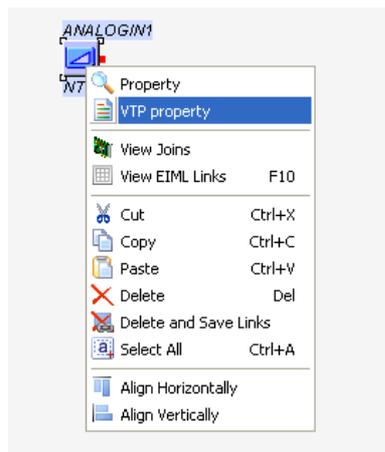
4.21.1 Generating and modifying a VTP driver file

To generate a VTP driver (a VTP driver that has never been generated before) the tool *VTPMaker* (*Tools/VTPMaker Wizard*) must be used; look at the proper section in the manual.
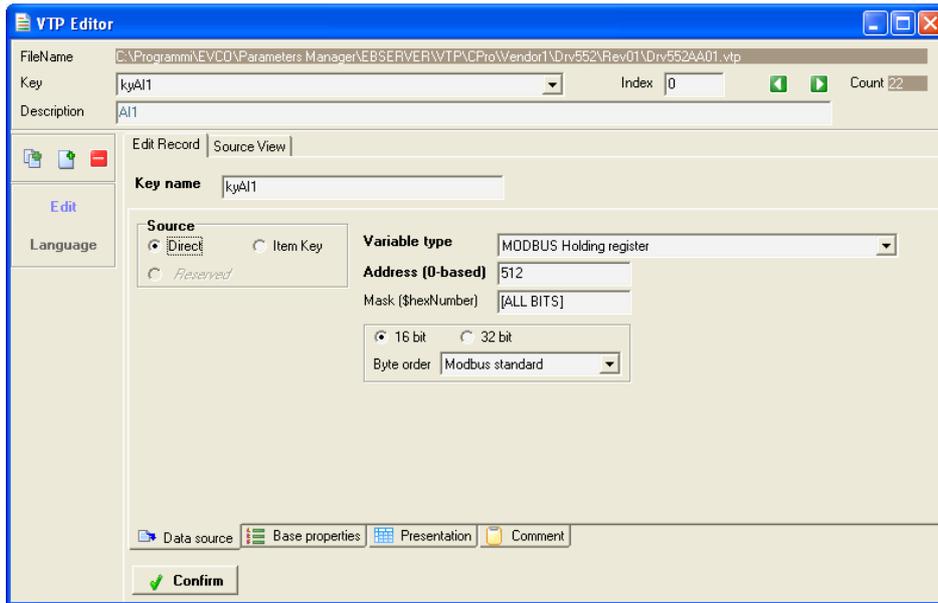
If parameters Manager is installed in the system, the generated driver is also copied in the folder of the drivers used by the program. The modifications that through the VTPEditor are applied to the properties of the variables there are in the driver, are saved in the operating driver used by Parameters Manager.

Once the VTP driver has been generated and is present in the drivers folder of Parameters Manager, in the pop-up menu relative to an entity, a new voice ("VTP Property") will appear.

Selecting this voice a new window opens; through this window it is possible to edit the properties there are in the driver and relative to the entity.

Selecting VTP Property, it is possible to gain access the tool *VTP Editor*.
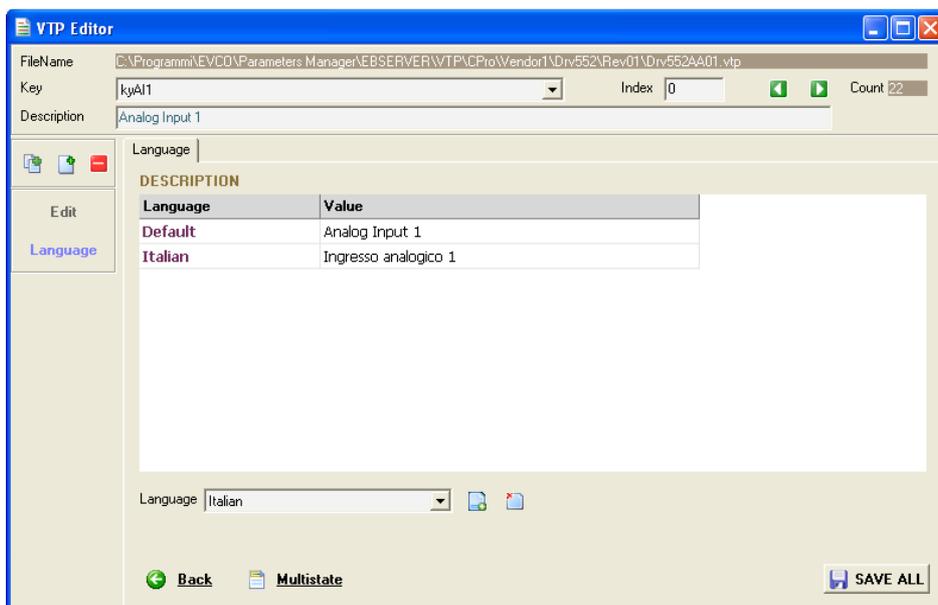
The editor window about the properties of an entity in the VTP is divided in two sections: *Edit* and *Language*.

Section *Edit* is divided in four further sections:

- *Data Source*: it specifies the sources of the entity (Modbus address, bit number required for the exportation, the order these are exported in Modbus)
- *Base properties*: it specifies the properties identifying the entity (kind of variable, minimum and maximum values, unit of measurements; in case that the variable represents a status it also specifies those values the status can assume and its description)
- *Presentation*: it specifies the representation of the entity in Parameters Manager (belonging group, style, level of visibility and so on)
- *Comment*: comments the user can insert about the entity.

In the section *Language* it is possible to insert the description of the entity (in its default value or specified in various languages).
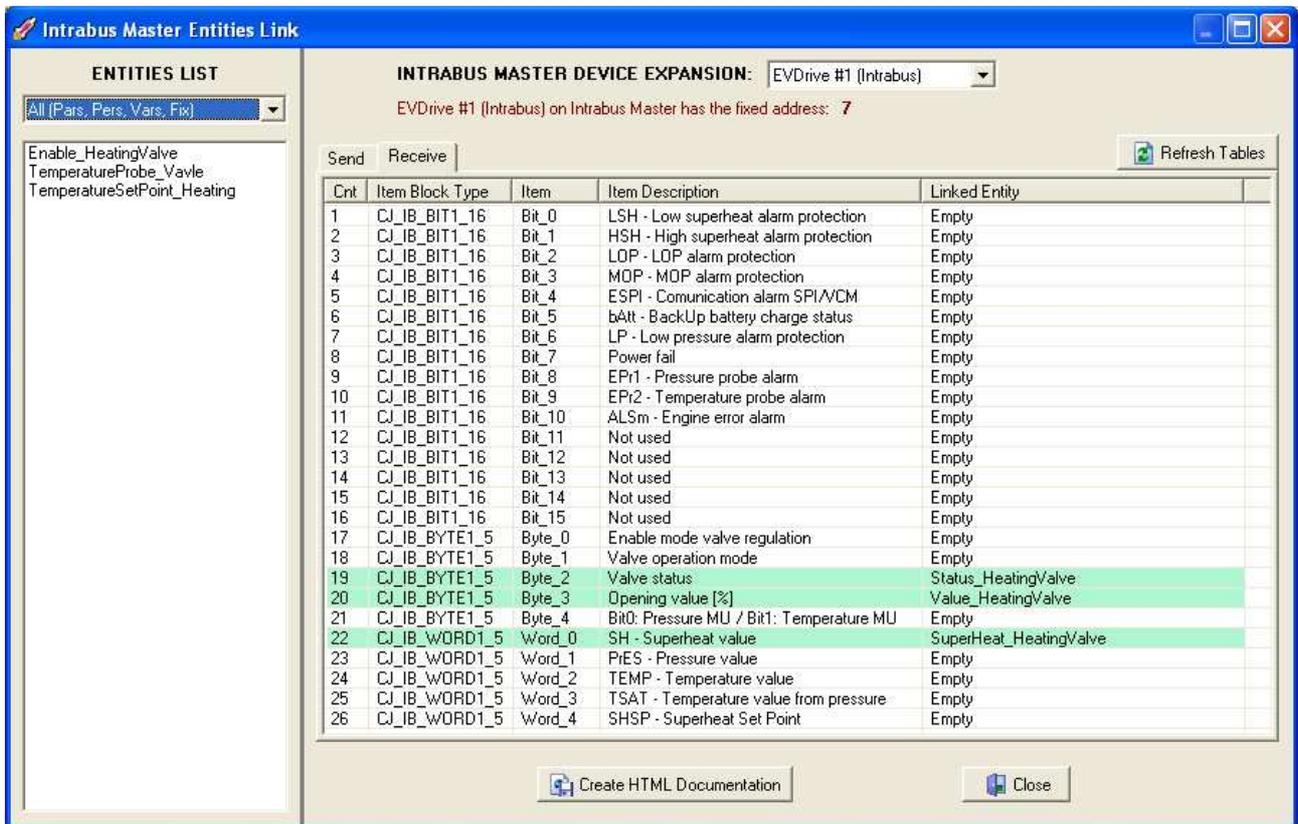
4.22 Expansions and remote terminals configuration

To realize the application software we can use expansions to increase the I/O, specific drivers for expansions valves and remote terminals. All this elements can be managed by the main controller, defining, in the project, the links to configuration parameters or to internal status with copies in the applicative project, trough appropriate tools. According to the chosen local bus (Intrabus or CAN) we can call the relative configuration tool.

4.22.1 Intrabus Master

In case the local Bus is Intrabus, for standard configuration the controller is master and can manage up to two I/O expansions and up to two drivers for expansion valve. Trough the *Tools/Intrabus master...* menu we can activate the following configuration window:
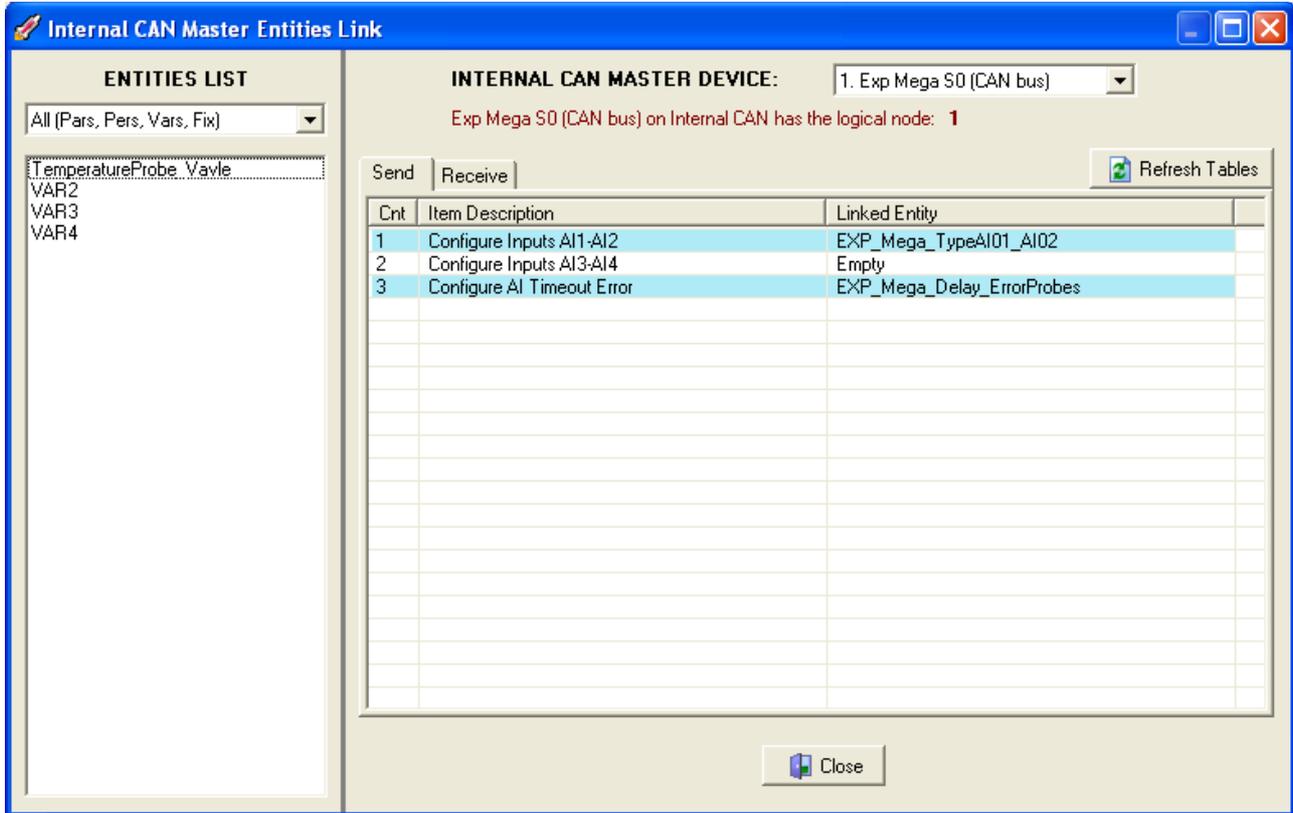

In the left-hand side of the window there are all the project variables, while in the right-hand side there are, for each configurable element in the network, two lists (selectable trough the relatives Tab): the list of the configurations to force as *Send* and the one of the configurations to read as *Receive*. To make easier the research, operating on the combo in the top left corner it is possible to filter the list according to the type.
To connect a parameter or project variable to a parameter or variable on the network you have to select it in the left list and drag it with the mouse in the desired position of the right list.

4.22.2 CAN master

In case the local Bus is CAN, the controller is master for that Bus only if the relative configuration parameter is enabled (look at Hardware Expert - 6 - Network CAN). Some controllers can manage up to two CAN channels (internal and extaernal). Trough the *Tools/CAN master…* menu and chosen the relative channel one activates the following configuration window:

In the left-hand side there are all the project variables, while in the right-hand side there are, for each element present in the network, two lists (selectable trough the relative Tab): the list of the configurations to force as *Send* and the one of the configurations to read as *Receive*. To make easier the research, operating on the combo in the top left corner it is possible to filter the list according to the type.

To connect a parameter or project variable to a parameter or variable on the network you have to select it in the left list and drag it with the mouse in the desired position of the right list.

4.23 **Modbus Master Entities Link**

The project entities, status and controls for the Modbus network can be sent or received using this tool to allow to the master controller the correctly communicate with the connected devices.
Trough the *Tools/Modbus Master...* menu we can activate the following configuration window:

In the left-hand side of the window there are all the project variables, while in the right-hand side there are, for each configurable element in the network, two lists (selectable trough the relatives Tab): the list of the configurations to force as *Send* and the one of the configurations to read as *Receive*. To make easier the research, operating on the combo in the top left corner it is possible to filter the list according to the type.
To connect a parameter or project variable to a parameter or variable on the network you have to select it in the left list and drag it with the mouse in the desired position of the right list.
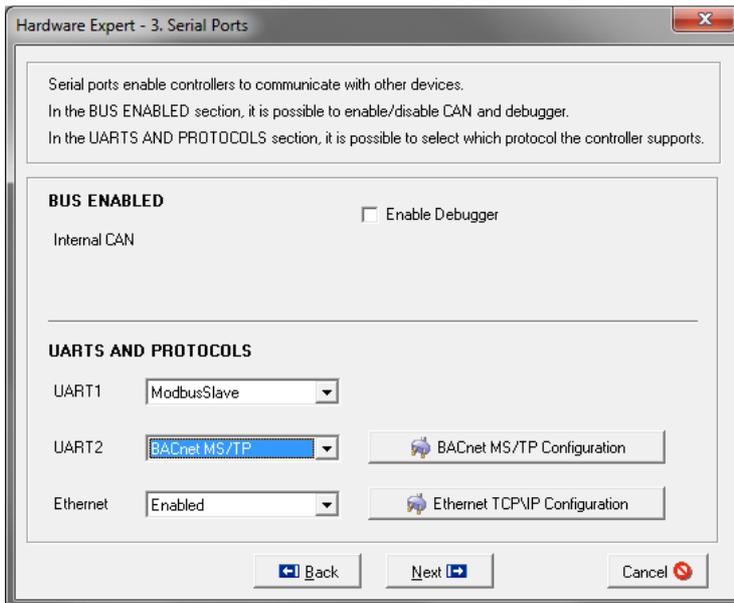*Priority* column refers to the *masterRefresh* property of the entities connected.
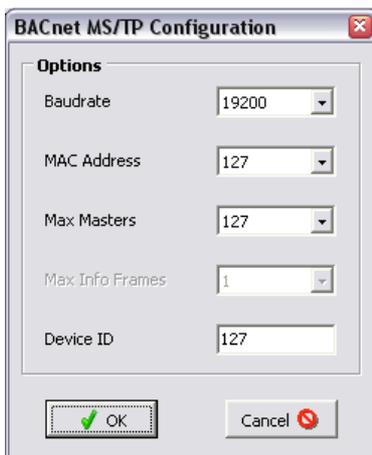
4.24 BACnet

In projects where the BACnet protocol is selected, you can define the objects for which you want to do monitoring in a BACnet MS/TP network, or IP network, depending on the choice in the Hardware Expert. The profile will be created with a configurator in the UNI-PRO and saved to an external file of the project.

4.24.1 BACnet MS/TP Configuration

If you want to configure the controller with BACnet MS/TP you must choose this protocol in the Hardware Expert from the UART dropdown list.

Clicking the button *BACnet MS/TP Configuration* it is possible to configure the main communication properties of the protocol:
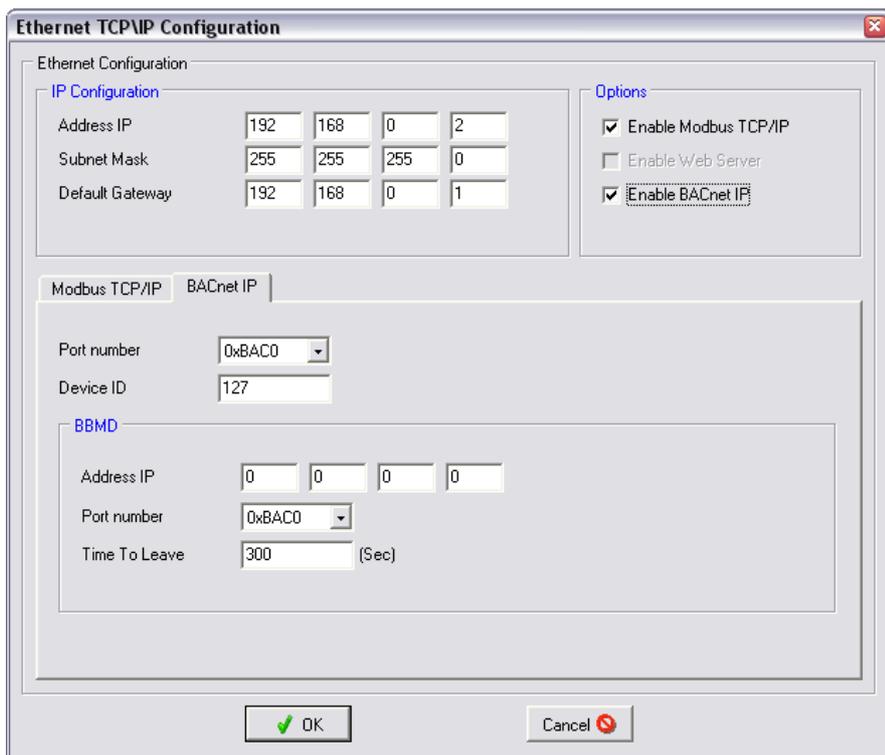

Options:
- **BaudRate** → set the communication speed using the dropdown menu. Supported baud rate for BACnet MS/TP are 9600, 19200, 38400 and 76800. Default = 19200.
- **MAC Address** → defines the physical address (1..127) within the BACnet local network. Default = 127.
- **Max Master** → specifies the highest allowable address for master nodes in the network, to optimize the token passing procedure. Default = 127.

- **Max Info Frames** → indicates the maximum number of frames for each communication session. Default = 1.
- **Device ID** → enter a Device Instance value (1..4194303). This property is a numeric code that is used to identify the Device object: the object identifier shall be unique internetwork-wide. Default = 127.

4.24.2 BACnet IP Configuration

If you want to configure the controller with BACnet IP you first have to enable the Ethernet port in the Hardware Expert wizard and then selsect this protocol in the options available in this port:

In the tab *BACnet IP* it is possible to configure the main communication properties of the protocol:
- **Port Number** → indicates the port the device will be listening on. The default BACnet port is 47808 (or BAC0 in hexadecimal)
- **Device ID** → enter a Device Instance value (1..4194303). This property is a numeric code that is used to identify the Device object: the object identifier shall be unique internetwork-wide. Default = 127.
- **BBMD Address IP** → indicates the BACnet Broadcast Management Device (BBMD) IP address. Local and foreign BACnet devices both use this address to register with the BBMD to be able to send and receive BACnet broadcast communications.
- **BBMD Port Number** → indicates the port the device will be broadcasting on. The default BACnet port is 47808 (or BAC0 in hexadecimal).
- **BBMD Time To Live** → specifies the length of time in seconds during which the BBMD forwards broadcast messages. Default = 300s
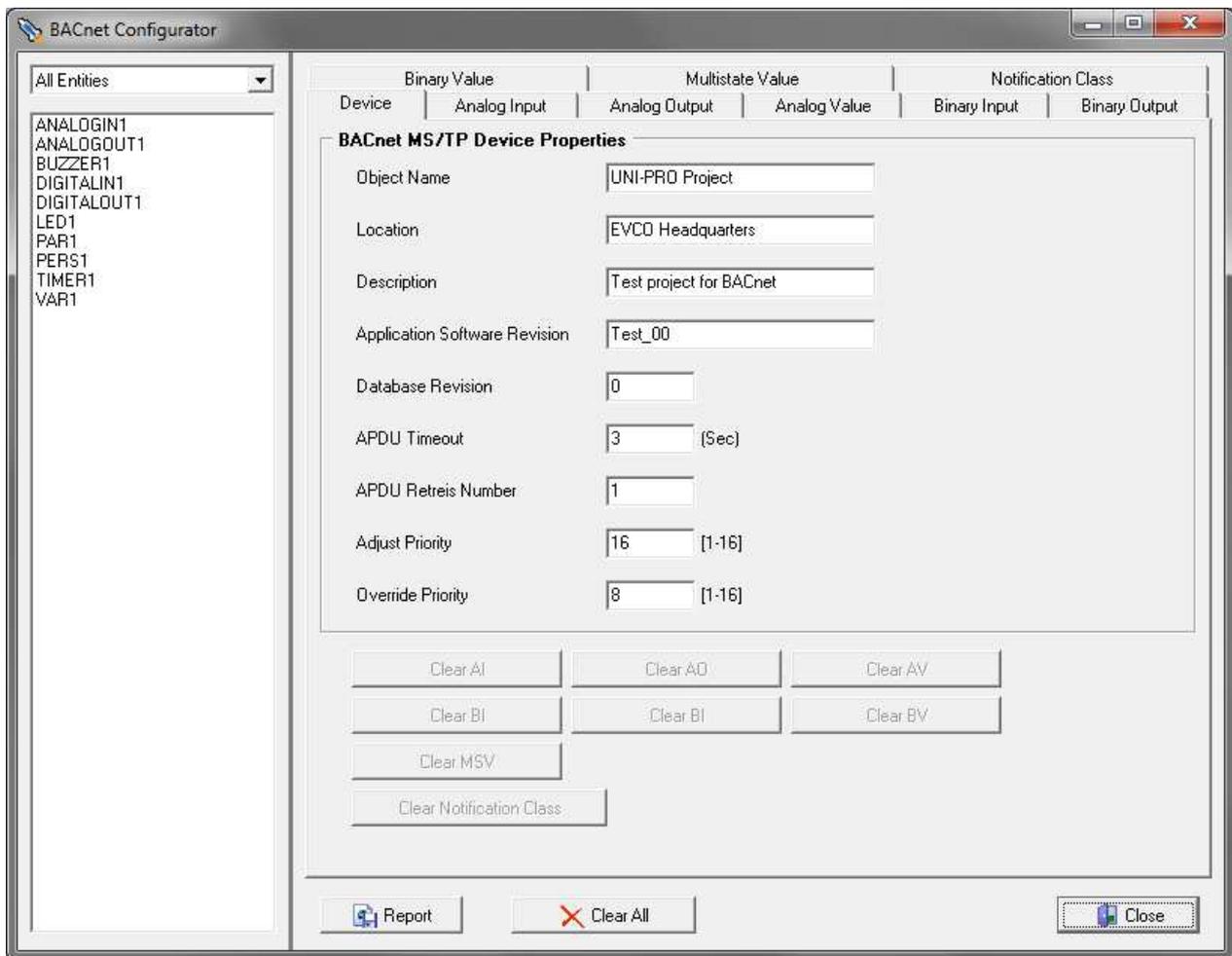
4.24.3 BACnet profile file

For projects that use the BACnet protocol will be saved a private files with the extension *.upbp* containing all the profile information for the created objects. The file will be saved each time you save the project and will be stored at the same level of ucjp project file (eg "ProjectName_X.ucjp") in an ad-hoc created folder (eg "**BACnet**_ProjectName_X\\ProjectName_X_**Profile.upbp**").

The name of the proposal profile as well as the name of the folder files should not be changed, if you rename the ucjp project file, also need to be renamed the folder and the profile file the same way.

At the opening of the project, UNI-PRO will load the information from the BACnet profile file, if the file is not present an error is reported. To avoid unwatend data loss, if an error occurs during the profile loading close the project without saving and reposition the BACnet profile file in the specified search path.

4.24.4 BACnet Configurator

Through the *Tools* menu by selecting *BACnet Configurator* item, we can open the configurator for the BACnet objects; each type of object is represented in its own section.

In the **Device** section you can set the identifying properties of the controller in the network.

The basic objects such as Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value and Multi State Value is created by combining a UNI-PRO entity using the drag&drop of the entity in the appropriate object list. The association <entities, BACnet object> is permitted only if the entity is "represented" by the relative object BACnet, according to these rules:

Analog Input (AI), entities:
- Analog Input
- Entites of type CJ_ANALOG (excluded Fix).

Analog Output (AO), entities:
- Analog Output
- Timer
- Entites of type CJ_WORD (excluded Fix).

Analog Value (AV), entities:
- Timer
- Entites of type CJ_ANALOG, CJ_LONG, CJ_DWORD, CJ_SHORT, CJ_WORD, CJ_CHAR, CJ_S_BYTE e CJ_BYTE (excluded Fix).

Binary Input (BI), entities:
- Digital Input
- Entites of type CJ_BIT (excluded Fix).

Binary Output (BO), entities:
- Digital Output
- Entites of type CJ_BIT (excluded Fix).

Binary Value (BV), entities:
- Entites of type CJ_BIT (excluded Fix).

Multi State Value (MSV), for this objects the maximum number of state are 16. Entities:
- Buzzer
- Led
- Entites of type CJ_BUZZER, CJ_LED, CJ_BIT (excluded Fix).
- Entites of type CJ_BYTE with max properties <=15.

Once you have added the items you can modify its properties; to apply the changes you have to confirm by clicking *the Set Properties* button. To delete an item just delete the link with a reverse drag&drop or removal action (by button or menu).

The **Notification Class** section defines objects for event notification, these objects are not connected to any entity thus the creation and removal is bound by the expliciti action of adding and removing by button (or by menu). Once you create objects, you can modify its properties; to apply the changes you have to confirm by clicking the *Set Properties* button.

Following the explanation of the properties for each BACnet object implemented.

4.24.5 Device object properties

Object Name → It defines the BACnet Device Object Name. This property is a string of printable characters that shall represent a name for the object that is unique internetwork-wide.

Location → This property is a string of printable characters that indicates the physical location of the BACnet device.

Description → It defines the BACnet Device Object Description. This property is a string of printable characters that may be used to describe the application being carried out by the BACnet device or other locally desired descriptive information.

Application Software Revision → This property identifies the version of application software installed in the machine. The content of this string is a local matter, but it could be a date-and-time stamp, a programmer's name, a host file version number, etc.

Database Revision → This property, of type Unsigned, is a logical revision number for the device's database.

APDU Timeout → This parameter represents how much time a client waits for a response from a Device. Default is 3s (3000ms).
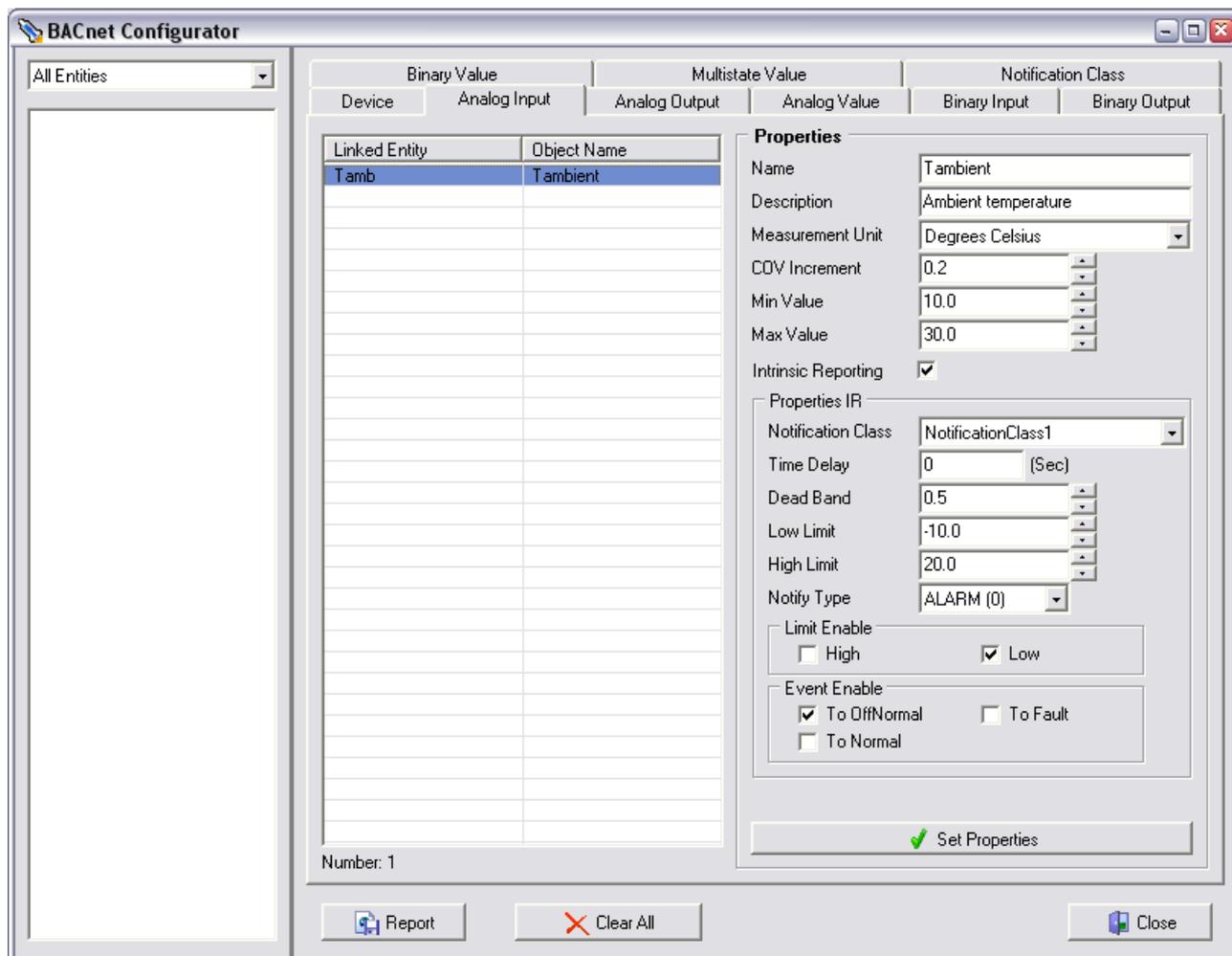
APDU Retries Number → This parameter represents the value of the Number of APDU Retries property of the node's Device object.

Reboot Password → The ReinitializeDevice service is used by a client BACnet-user to instruct a remote device to reboot itself (cold start), reset itself to some predefined initial state (warm start). Due to the sensitive nature of this service, a password is required from the client BACnet-user prior to executing the service. This password has to match with the Reboot Password in the controller.

Adjust Priority → The BACnet Priority Array is a specific property type described by the standard as one of the command prioritization mechanisms. The Adjust priority parameter defines with which priority the controller will command AV / BV / MV values. Default = 16.

Override Priority → The BACnet Priority Array is a specific property type described by the standard as one of the command prioritization mechanisms. The Override priority parameter defines with which priority the controller will command AO / BO / MO values. Default = 8.

4.24.6 Analog Input object properties

Object Name → It defines the BACnet Object Name. This property is a string of printable characters that shall represent a name for the object that is unique within the BACnet device that maintains it.

Description → It defines the BACnet Object Description. This property is a string of printable characters whose content is not restricted.

Measurement Unit → This property indicates the measurement units of this object. You can select it from a dropdown list of engineering units defined by this standard.

COV Increment → This property shall specify the minimum change in Present_Value that will cause a COVNotification to be issued to subscriber COV-clients.

Min Value → This property indicates the lowest number in engineering units that can be reliably obtained for the Present_Value property of this object.

Max Value → This property indicates the highest number in engineering units that can be reliably obtained for the Present_Value property of this object.

Intrinsic Reporting → This property specifies if the object supports intrinsic reporting and controls whether (TRUE) or not (FALSE) the object will be considered by event and alarm services.

Notification Class (*) → This property shall specify the instance of the Notification Class object to use for event-notification distribution.

Time Delay (*) → This property is the time delay in seconds for the object's event algorithm.

Dead Band (*) → This property is the dead band value for the object's event algorithm.

Low Limit (*) → This property is the low limit value for the object's event algorithm.

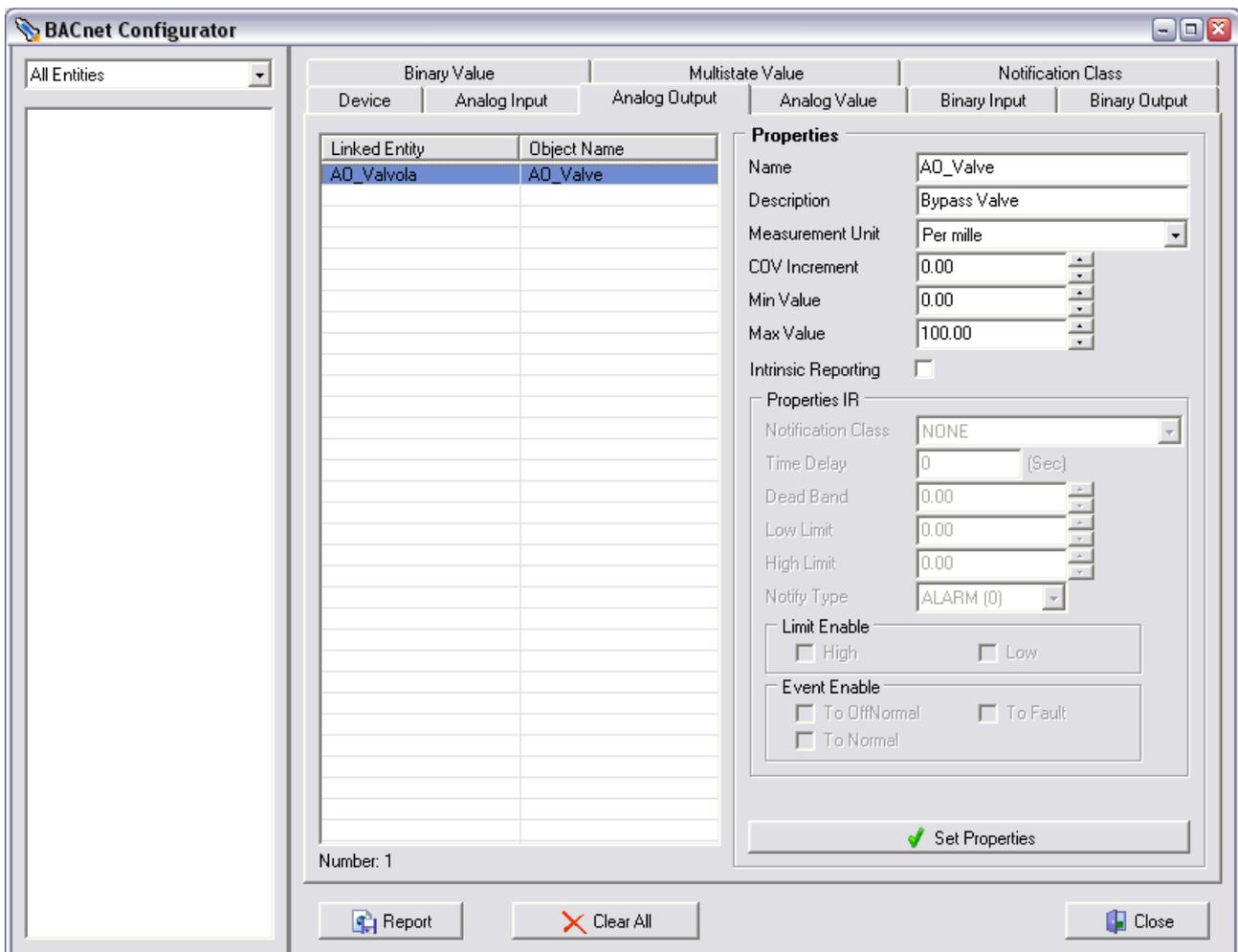**High Limit (*)** → This property is the high limit value for the object's event algorithm.

Notify Type (*) → This property shall convey whether the notifications generated by the object should be Events or Alarms. The value of the property, ALARM(0) or EVENT(1), is used as the value of the 'Notify Type' service parameter in event notifications generated by the object.

Limit Enable (*) → This property indicates what limits take in consideration for the object's event algorithm.

Event Enable (*) → This property shall convey three flags that separately enable and disable the distribution of TO_OFFNORMAL, TO_FAULT and TO_NORMAL notifications.

(*) The property is active only if **Intrinsic Reporting** is set TRUE.

4.24.7 Analog Output object properties

Object Name → It defines the BACnet Object Name. This property is a string of printable characters that shall represent a name for the object that is unique within the BACnet device that maintains it.

Description → It defines the BACnet Object Description. This property is a string of printable characters whose content is not restricted.

Measurement Unit → This property indicates the measurement units of this object. You can select it from a dropdown list of engineering units defined by this standard.

COV Increment → This property shall specify the minimum change in Present_Value that will cause a COVNotification to be issued to subscriber COV-clients.

Min Value → This property indicates the lowest number in engineering units that can be reliably obtained for the Present_Value property of this object.

Max Value → This property indicates the highest number in engineering units that can be reliably obtained for the Present_Value property of this object.

Intrinsic Reporting → This property specifies if the object supports intrinsic reporting and controls whether (TRUE) or not (FALSE) the object will be considered by event and alarm services.

Notification Class (*) → This property shall specify the instance of the Notification Class object to use for event-notification distribution.

Time Delay (*) → This property is the time delay in seconds for the object's event algorithm.

Dead Band (*) → This property is the dead band value for the object's event algorithm.

Low Limit (*) → This property is the low limit value for the object's event algorithm.

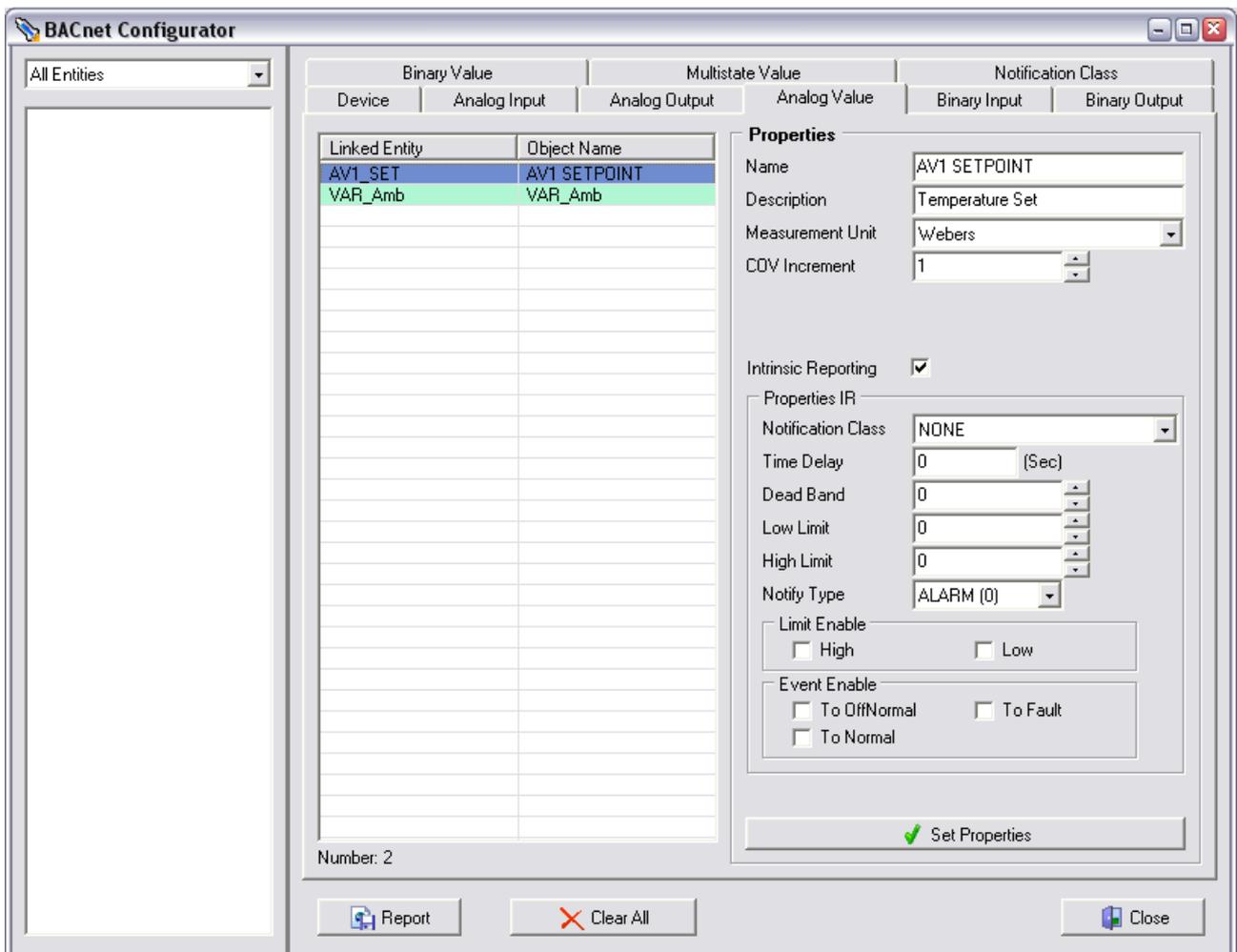**High Limit (*)** → This property is the high limit value for the object's event algorithm.

Notify Type (*) → This property shall convey whether the notifications generated by the object should be Events or Alarms. The value of the property, ALARM(0) or EVENT(1), is used as the value of the 'Notify Type' service parameter in event notifications generated by the object.

Limit Enable (*) → This property indicates what limits take in consideration for the object's event algorithm.

Event Enable (*) → This property shall convey three flags that separately enable and disable the distribution of TO_OFFNORMAL, TO_FAULT and TO_NORMAL notifications.

(*) The property is active only if **Intrinsic Reporting** is set TRUE.

4.24.8 Analog Value object properties

Object Name → It defines the BACnet Object Name. This property is a string of printable characters that shall represent a name for the object that is unique within the BACnet device that maintains it.

Description → It defines the BACnet Object Description. This property is a string of printable characters whose content is not restricted.

Measurement Unit → This property indicates the measurement units of this object. You can select it from a dropdown list of engineering units defined by this standard.

COV Increment → This property shall specify the minimum change in Present_Value that will cause a COVNotification to be issued to subscriber COV-clients.

Intrinsic Reporting → This property specifies if the object supports intrinsic reporting and controls whether (TRUE) or not (FALSE) the object will be considered by event and alarm services.

Notification Class (*) → This property shall specify the instance of the Notification Class object to use for event-notification distribution.

Time Delay (*) → This property is the time delay in seconds for the object's event algorithm.

Dead Band (*) → This property is the dead band value for the object's event algorithm.

Low Limit (*) → This property is the low limit value for the object's event algorithm.

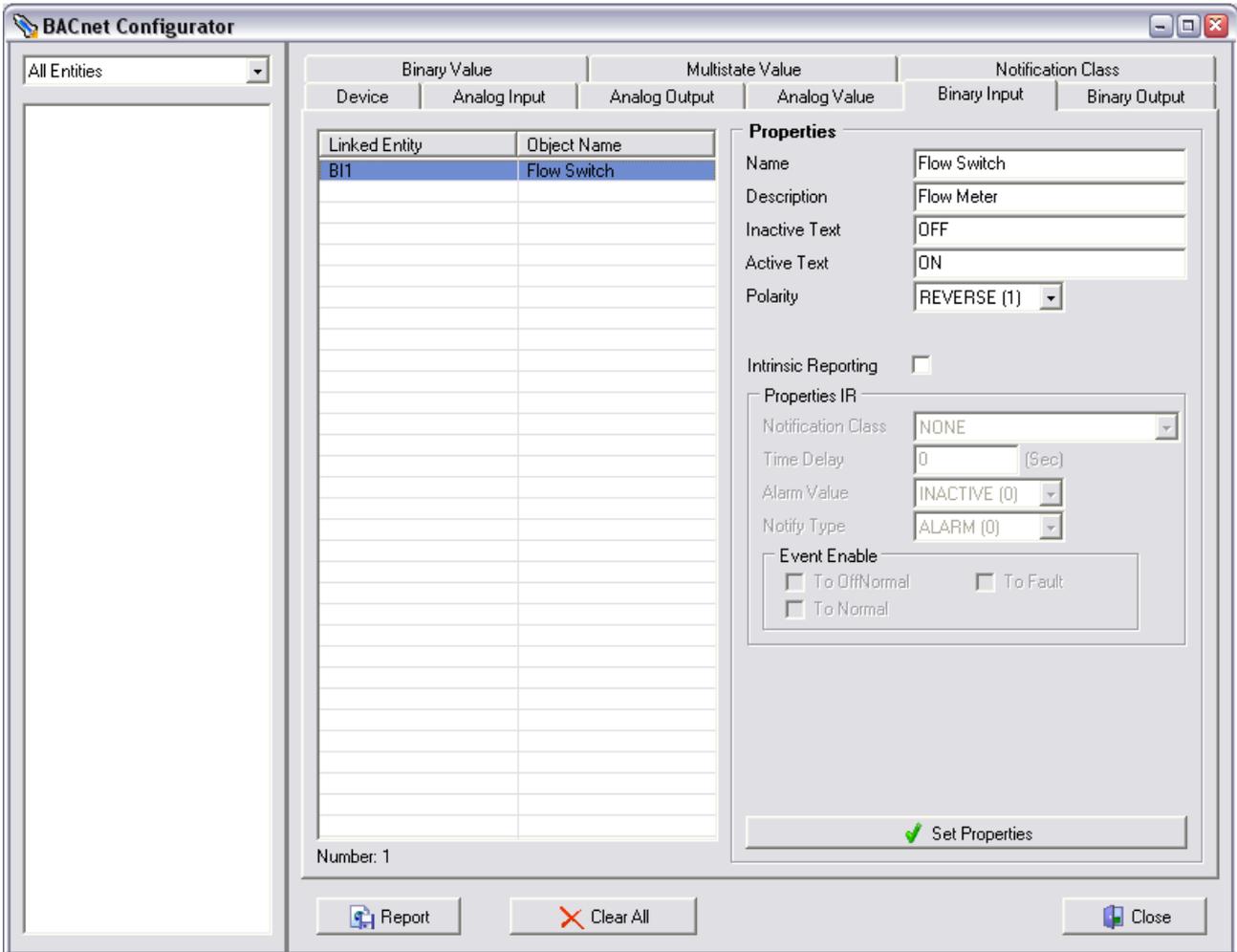**High Limit** (*) → This property is the high limit value for the object's event algorithm.

Notify Type (*) → This property shall convey whether the notifications generated by the object should be Events or Alarms. The value of the property, ALARM(0) or EVENT(1), is used as the value of the 'Notify Type' service parameter in event notifications generated by the object.

Limit Enable (*) → This property indicates what limits take in consideration for the object's event algorithm.

Event Enable (*) → This property shall convey three flags that separately enable and disable the distribution of TO_OFFNORMAL, TO_FAULT and TO_NORMAL notifications.

(*) The property is active only if **Intrinsic Reporting** is set TRUE.

4.24.9 Binary Input object properties

Object Name → It defines the BACnet Object Name. This property is a string of printable characters that shall represent a name for the object that is unique within the BACnet device that maintains it.

Description → It defines the BACnet Object Description. This property is a string of printable characters whose content is not restricted.

Inactive Text → This property characterizes the intended effect of the INACTIVE state of the Present_Value property from the human operator's viewpoint. The content of this string is a local matter, but it is intended to represent a human-readable description of the INACTIVE state. If either the Inactive_Text property or the Active_Text property is present, then both of them shall be present.

Active Text → This property characterizes the intended effect of the ACTIVE state of the Present_Value property from the human operator's viewpoint. The content of this string is a local

matter, but it is intended to represent a human-readable description of the ACTIVE state. If either the Inactive_Text property or the Active_Text property is present, then both of them shall be present.

Polarity → This property indicates the relationship between the physical state of the Input and the logical state represented by the Present_Value property. If the Polarity property is NORMAL, then the ACTIVE state of the Present_Value property is the ON state of the physical Input. If the Polarity property is REVERSE, then the ACTIVE state of the Present_Value property is the OFF state of the physical Input.

Intrinsic Reporting → This property specifies if the object supports intrinsic reporting and controls whether (TRUE) or not (FALSE) the object will be considered by event and alarm services.

Notification Class (*) → This property shall specify the instance of the Notification Class object to use for event-notification distribution.

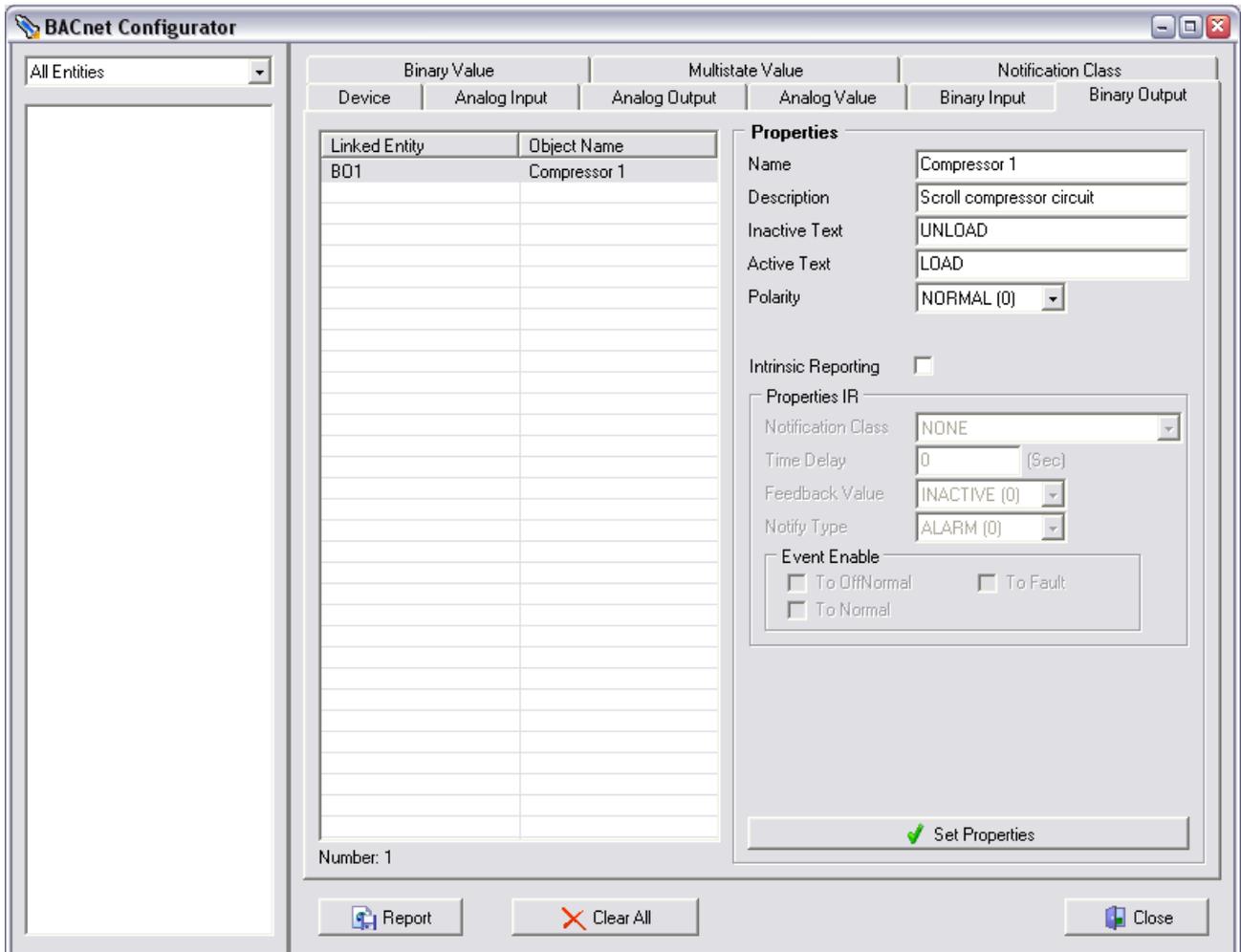**Time Delay (*)** → This property is the time delay in seconds for the object's event algorithm.

Alarm Value (*) → This property is the value (ACTIVE or INACTIVE) for the object's event algorithm.

Notify Type (*) → This property shall convey whether the notifications generated by the object should be Events or Alarms. The value of the property, ALARM(0) or EVENT(1), is used as the value of the 'Notify Type' service parameter in event notifications generated by the object.

Event Enable (*) → This property shall convey three flags that separately enable and disable the distribution of TO_OFFNORMAL, TO_FAULT and TO_NORMAL notifications.

(*) The property is active only if **Intrinsic Reporting** is set TRUE.

4.24.10   Binary Output object properties

Object Name → It defines the BACnet Object Name. This property is a string of printable characters that shall represent a name for the object that is unique within the BACnet device that maintains it.

Description → It defines the BACnet Object Description. This property is a string of printable characters whose content is not restricted.

Inactive Text → This property characterizes the intended effect of the INACTIVE state of the Present_Value property from the human operator's viewpoint. The content of this string is a local matter, but it is intended to represent a human-readable description of the INACTIVE state. If either the Inactive_Text property or the Active_Text property is present, then both of them shall be present.

Active Text → This property characterizes the intended effect of the ACTIVE state of the Present_Value property from the human operator's viewpoint. The content of this string is a local matter, but it is intended to represent a human-readable description of the ACTIVE state. If either the Inactive_Text property or the Active_Text property is present, then both of them shall be present.

Polarity → This property indicates the relationship between the physical state of the Output and the logical state represented by the Present_Value property. If the Polarity property is NORMAL, then the ACTIVE state of the Present_Value property is the ON state of the physical Output. If the Polarity property is REVERSE, then the ACTIVE state of the Present_Value property is the OFF state of the physical Output.

Intrinsic Reporting → This property specifies if the object supports intrinsic reporting and controls whether (TRUE) or not (FALSE) the object will be considered by event and alarm services.
Notification Class (*) → This property shall specify the instance of the Notification Class object to use for event-notification distribution.
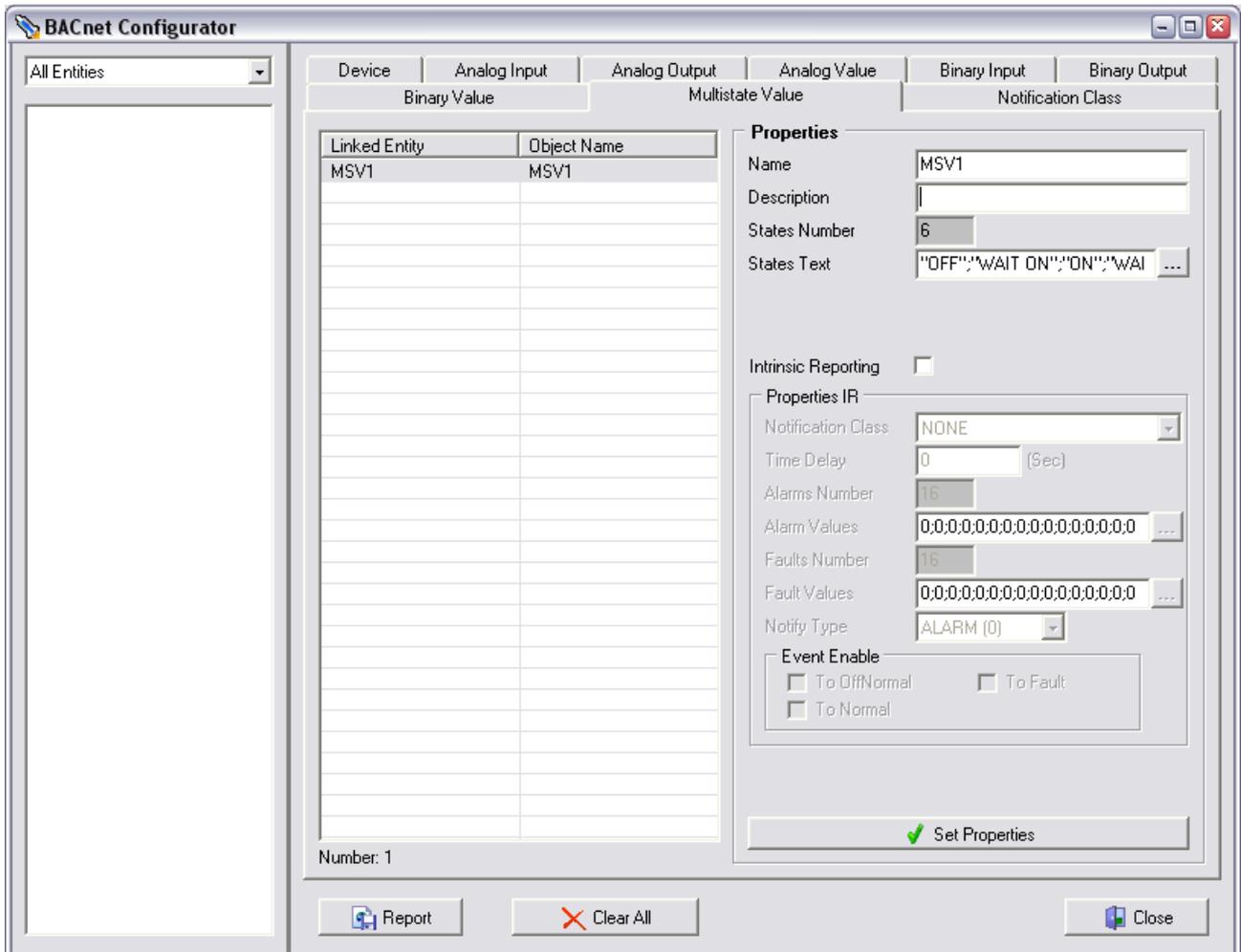**Time Delay (*)** → This property is the time delay in seconds for the object's event algorithm.
Feedback Value (*) → This property is an indication of the actual value of the entity controlled by Present_Value used for the object's event algorithm, if the object supports event reporting.
Notify Type (*) → This property shall convey whether the notifications generated by the object should be Events or Alarms. The value of the property, ALARM(0) or EVENT(1), is used as the value of the 'Notify Type' service parameter in event notifications generated by the object.
Event Enable (*) → This property shall convey three flags that separately enable and disable the distribution of TO_OFFNORMAL, TO_FAULT and TO_NORMAL notifications.

(*) The property is active only if **Intrinsic Reporting** is set TRUE.

4.24.11   Binary Value object properties

Object Name → It defines the BACnet Object Name. This property is a string of printable characters that shall represent a name for the object that is unique within the BACnet device that maintains it.
Description → It defines the BACnet Object Description. This property is a string of printable characters whose content is not restricted.

Inactive Text → This property characterizes the intended effect of the INACTIVE state of the Present_Value property from the human operator's viewpoint. The content of this string is a local matter, but it is intended to represent a human-readable description of the INACTIVE state. If either the Inactive_Text property or the Active_Text property is present, then both of them shall be present.

Active Text → This property characterizes the intended effect of the ACTIVE state of the Present_Value property from the human operator's viewpoint. The content of this string is a local matter, but it is intended to represent a human-readable description of the ACTIVE state. If either the Inactive_Text property or the Active_Text property is present, then both of them shall be present.

Intrinsic Reporting → This property specifies if the object supports intrinsic reporting and controls whether (TRUE) or not (FALSE) the object will be considered by event and alarm services.

Notification Class (*) → This property shall specify the instance of the Notification Class object to use for event-notification distribution.

Time Delay (*) → This property is the time delay in seconds for the object's event algorithm.

Alarm Value (*) → This property is the value (ACTIVE or INACTIVE) for the object's event algorithm.

Notify Type (*) → This property shall convey whether the notifications generated by the object should be Events or Alarms. The value of the property, ALARM(0) or EVENT(1), is used as the value of the 'Notify Type' service parameter in event notifications generated by the object.

Event Enable (*) → This property shall convey three flags that separately enable and disable the distribution of TO_OFFNORMAL, TO_FAULT and TO_NORMAL notifications.

(*) The property is active only if **Intrinsic Reporting** is set TRUE.

4.24.12   Multistate Value object properties

Object Name → It defines the BACnet Object Name. This property is a string of printable characters that shall represent a name for the object that is unique within the BACnet device that maintains it.

Description → It defines the BACnet Object Description. This property is a string of printable characters whose content is not restricted.

States Number → This property defines the number of states the Present_Value may have. The property shall always have a value greater than zero.

States Text → This property is a list of strings of printable characters representing descriptions of all possible states of the Present_Value.

Intrinsic Reporting → This property specifies if the object supports intrinsic reporting and controls whether (TRUE) or not (FALSE) the object will be considered by event and alarm services.

Notification Class (*) → This property shall specify the instance of the Notification Class object to use for event-notification distribution.

Time Delay (*) → This property is the time delay in seconds for the object's event algorithm.

Alarm Values (*) → This property is a index list of of possible states representing alarm condition for the object's event algorithm.

Alarm Number → This read only value is automatically calculated from the Alarm Values list.

Fault Values (*) → This property is a index list of of possible states representing fault condition for the object's event algorithm.

Fault Number → This read only value is automatically calculated from the Fault Values list.

Notify Type (*) → This property shall convey whether the notifications generated by the object should be Events or Alarms. The value of the property, ALARM(0) or EVENT(1), is used as the value of the 'Notify Type' service parameter in event notifications generated by the object.
Event Enable (*) → This property shall convey three flags that separately enable and disable the distribution of TO_OFFNORMAL, TO_FAULT and TO_NORMAL notifications.

(*) The property is active only if **Intrinsic Reporting** is set TRUE.

4.24.13 Notification Class object properties

Using this form it is possible to setup until 5 Notification Classes, each one with a Recipient List limited to maximum 5 Destinations.

Object Name → It defines the BACnet Object Name. This property is a string of printable characters that shall represent a name for the object that is unique within the BACnet device that maintains it.
Description → It defines the BACnet Object Description. This property is a string of printable characters whose content is not restricted.
NC Idx → It represent the progressive index of the Notification Class Object.
Priority → This property, an unsigned value for each of the three possible events, shall convey the priority to be used for event notifications for TO_OFFNORMAL, TO_FAULT, and TO_NORMAL

events, respectively. Priorities shall range from 0 - 255 inclusive. A lower number indicates a higher priority.

Ack required → This property, one boolean condition for each of the three possible events, shall convey three separate flags that represent whether acknowledgment shall be required in notifications generated for TO_OFFNORMAL, TO_FAULT, and TO_NORMAL event transitions, respectively.

Destination List → This property shall convey a list of zero or more recipient destinations to which notifications shall be sent when event-initiating objects using this class detect the occurrence of an event. These recipient destinations are intended to be relatively permanent, do not expire, and shall be maintained through a power failure or device "restart."

For each element of the list it is possible to configure:
- The name
- The valid days
- The window of time during which the destination is viable on the days of the week specified as valid days.
- The recipient type NETWORK (0) or ADDRESS(1)
- The Device ID in case of NETWORK (0)
- The Network ID and the MAC Address (BACnet MS/TP) or IP Address (BACnet IP) in case of ADDRESS(1)
- The flag Confirmed Notification: (TRUE) if confirmed notifications are to be sent and (FALSE) if unconfirmed notifications are to be sent
- Transitions: a set of three flags that indicate those transitions {TO_OFFNORMAL, TO_FAULT, TO_NORMAL} for which this recipient is suitable
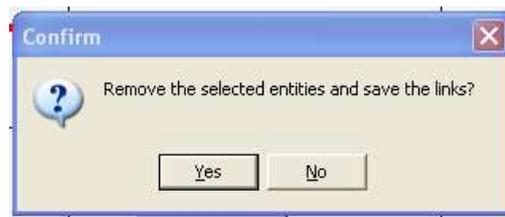
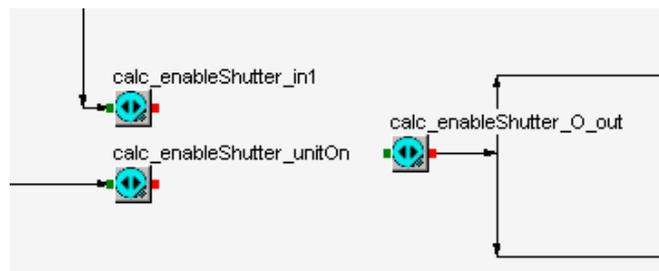## 4.25 "Delete and save links" function and LinkFixer

This function can be used to remove the entities selected and to save the connections, introducing the variables which "secure" the connections (of the deleted entity) with other entities in the project.
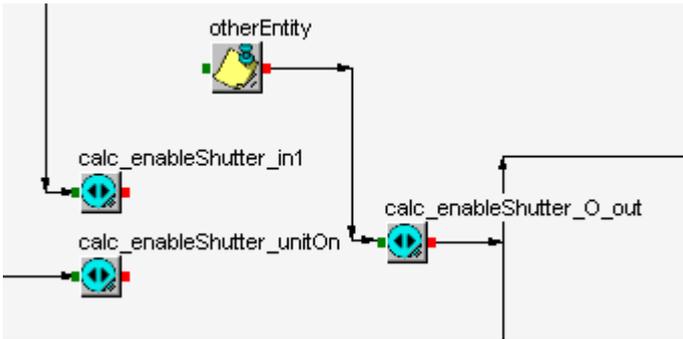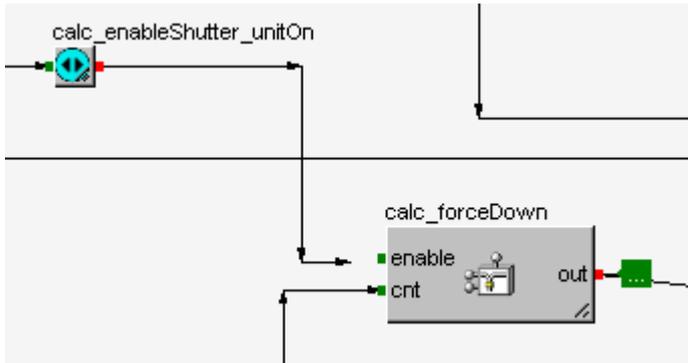

Click…

Click Yes ….

The entity *calc_enableShutter* has been removed and three variables were created (one for each input and output terminal) which are connected to the entity to which *calc_enableShutter* was connected. In this way the inputs/outputs have been "Secured" and can be elaborated and reused.

To fix the I/O of the deleted entities are created "Link Fixer" special entities. This object allow to memorize the links and they are simplified; if connect the free input or output to other entity the Link Fixer will be automatically deleted and the link will be automatically connected to the new entity.
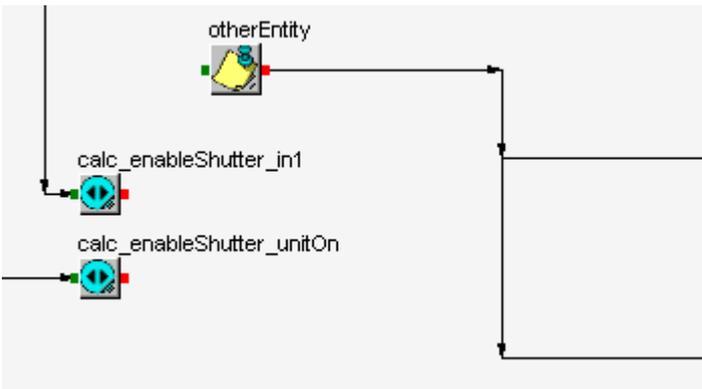
A)
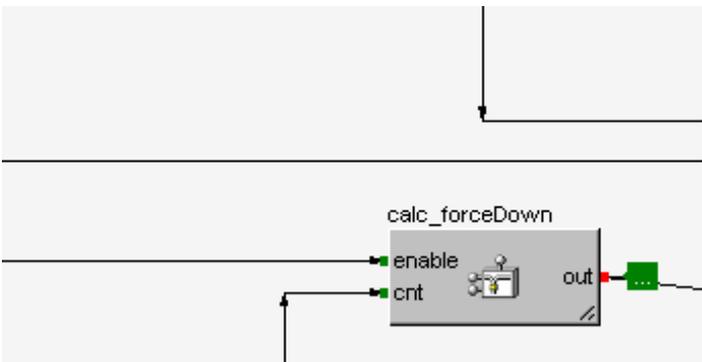Linking of *calc_enableShutter_O_out* with the new entity "otherEntity"

B)
Linking of *calc_enableShutter_unitON* to the "enable" input of the algorithm "calc_forceDown"

Solve of the Link Fixer

A)
Semplification of the Link Fixer *calc_enableShutter_O_out*; entity "otherEntity" will be automatically connected to the clamps connected with the Link fixer that will be removed

B)
Semplification of the Link Fixer *calc_enableShutter_unitOn*; input "enable" of the algorithm will be automatically connected to the entity connected with the Link Fixer that will be removed
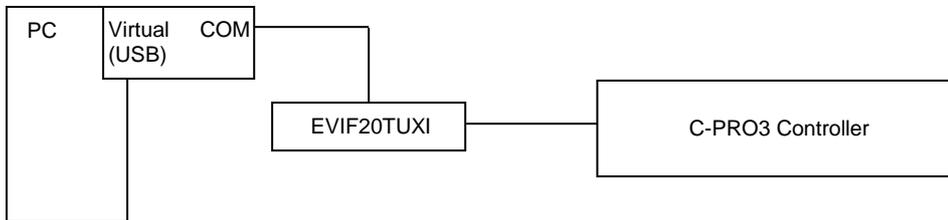
4.25.1 Function "Fix links"

This function allows you to "detach and attach" connections for individual I/O terminals of entities. Similarly previous function, selecting the "Fix links" function from the right-click menu on the terminal, it is created a Link Fixer and the links are deleted from the clamp and transferred to the new Link Fixer entity just generated.
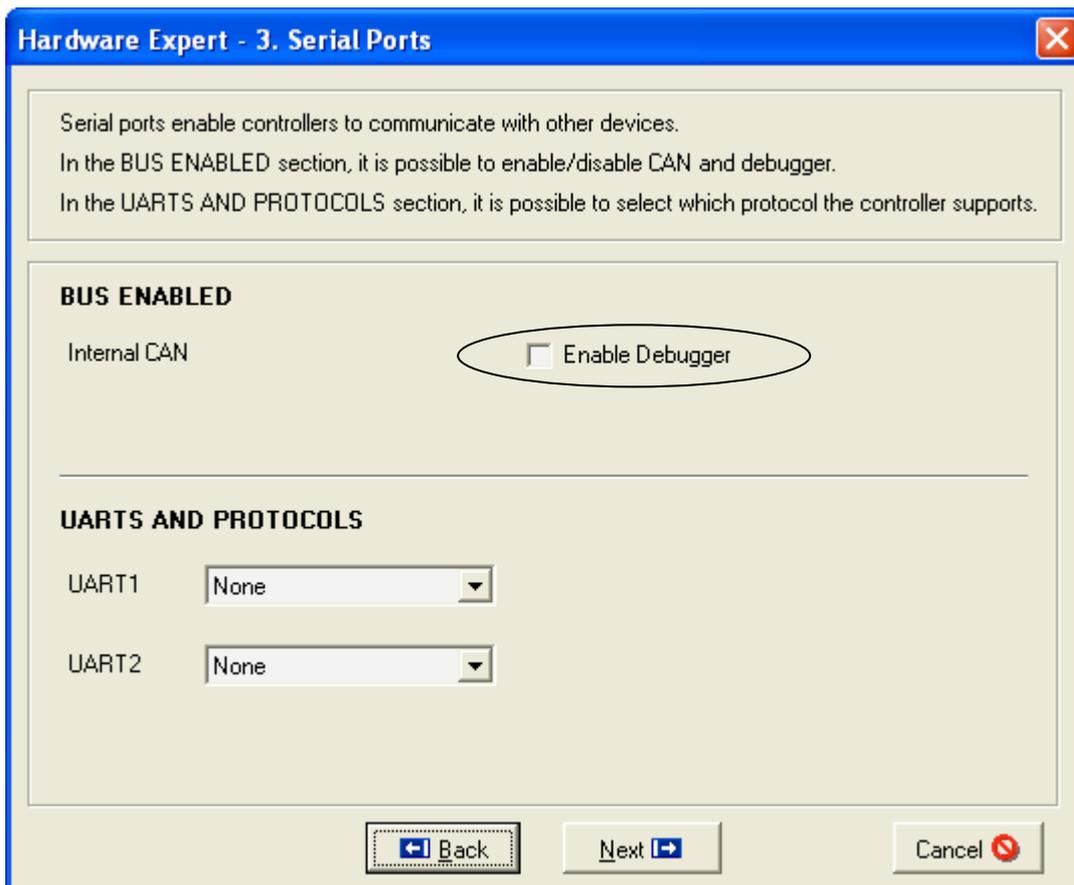
4.26 CAN Config

CAN Config is a tool that allows the programming of the CANbus parameters of the controllers, of the expansions and of the user interfaces belonging to the c-pro 3.

To use *CAN Config* no particular hardware is required but the interface EVIF20TUXI, also required for the programming of the devices.
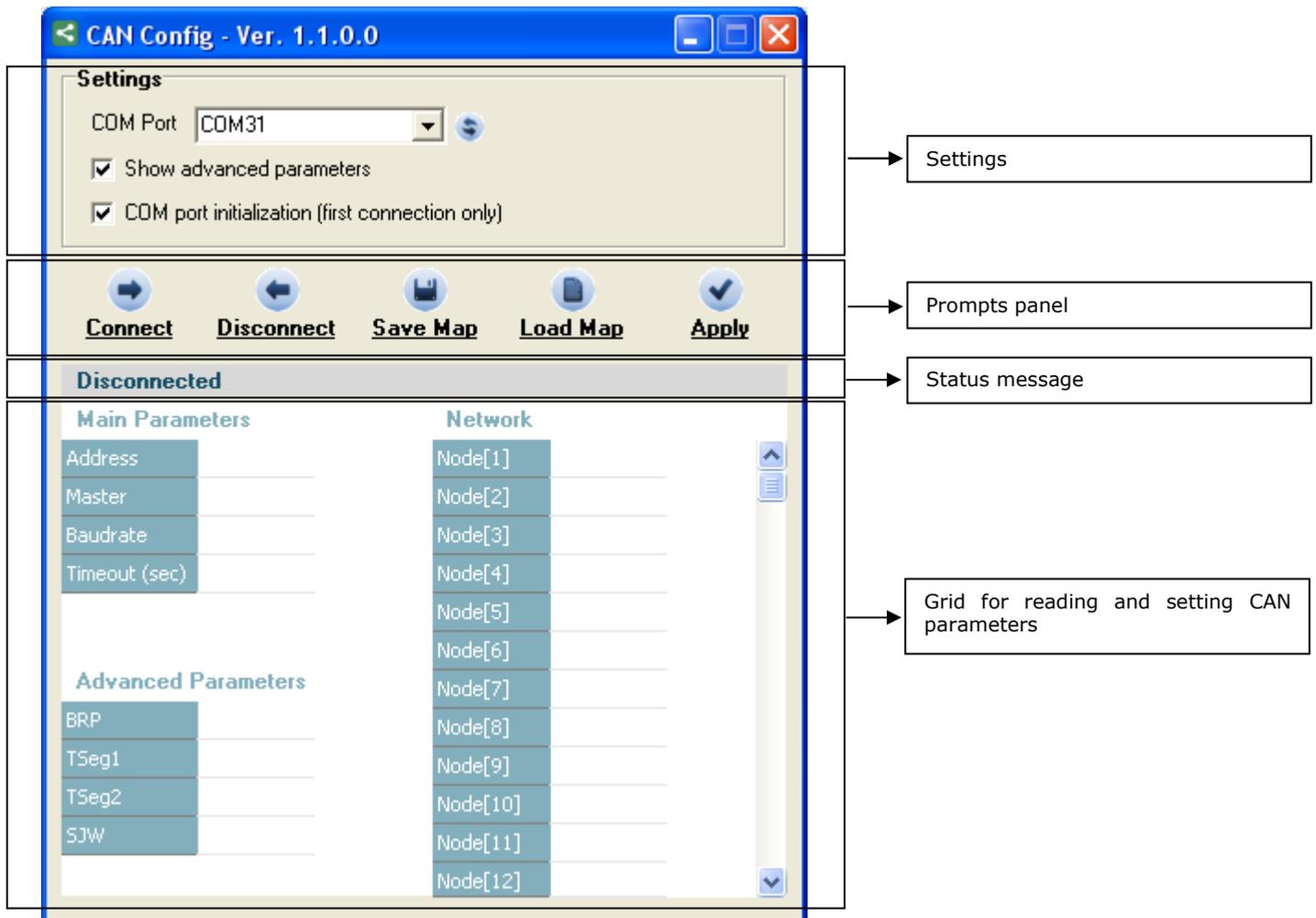
Previous versions of UNI-PRO did not allow the use of CAN Config if the utility Debugger was activated. From version 3.3 this problem has been solved and the two things can work.

Just pay attention: if it is wanted to use CAN Config with old applications compiled with previous versions to the 3.3 and with the use of Debugger activated, it is necessary these are compiled with the new version of UNI-PRO (from ver. 3.3); on afterwards make the download into the controller again.

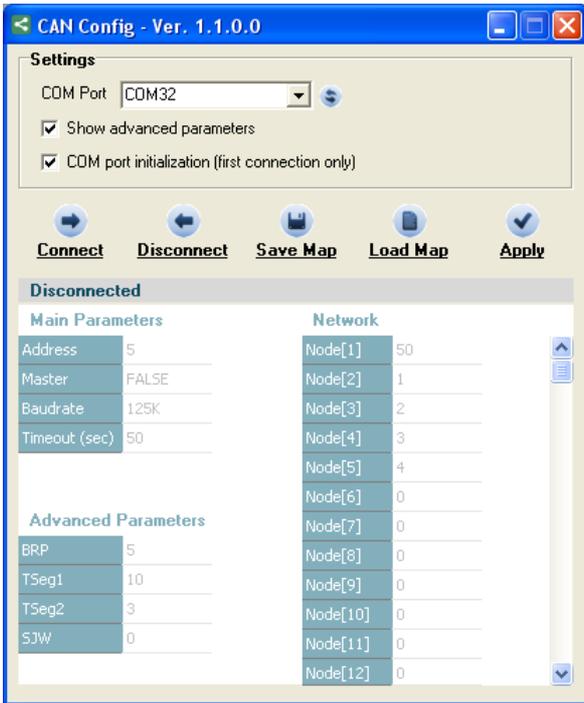Choosing menu Tools/CANconfig, the following window is displayed:

Settings
- **COM Port**: serial port where the interface EVIF20TUXI is connected; if the connected controller is c-pro 3 micro/kilo (or directly to the USB port without using the interface EVIF20TUXI), the COM port is automatically recognized and this setting is ignored.
- **Show advanced parameters**: when selected, *bit* timing parameters are shown and can be set (they are shown in the grid as *Advanced Parameters*)
- **COM port initialization**: when selected, an initialization of the serial port will be done (the first time the program joins to the controller using the *COM Port*, to make sure the communication with the controller does not belong to an unknows status the first time there is the connection). If the controller remains connected to the serial port, this will provoke its reset.
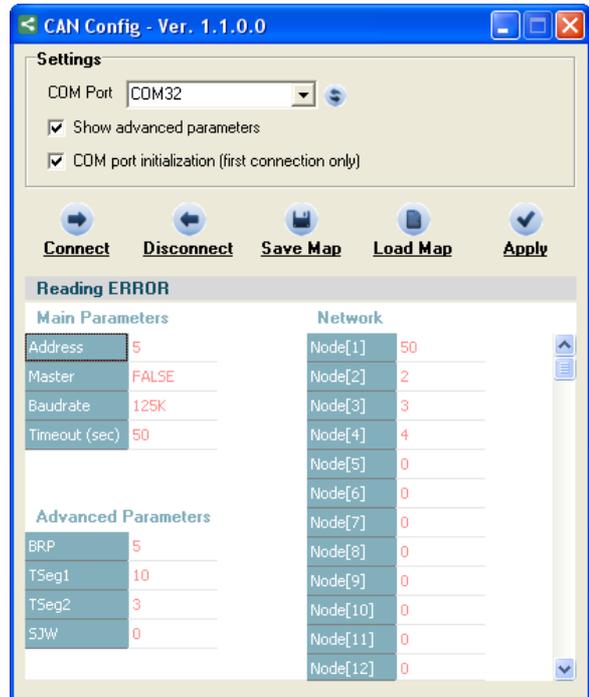-

Prompts
- **Connect**: it starts the communication with the controller. Since that moment it is possible to show and modify the CANbus communication parameters.
- **Disconnect**: it stops the communication with the controller
- **Save Map**: it saves the values there are in the grids in a file (map; Main Parameters, Network, Advanced Parameters)
- **Load Map**: it reads a "map" file and writes the loaded values in the controller
- **Apply**: it sends the controller the prompt to make valid the parameters that have been modified.
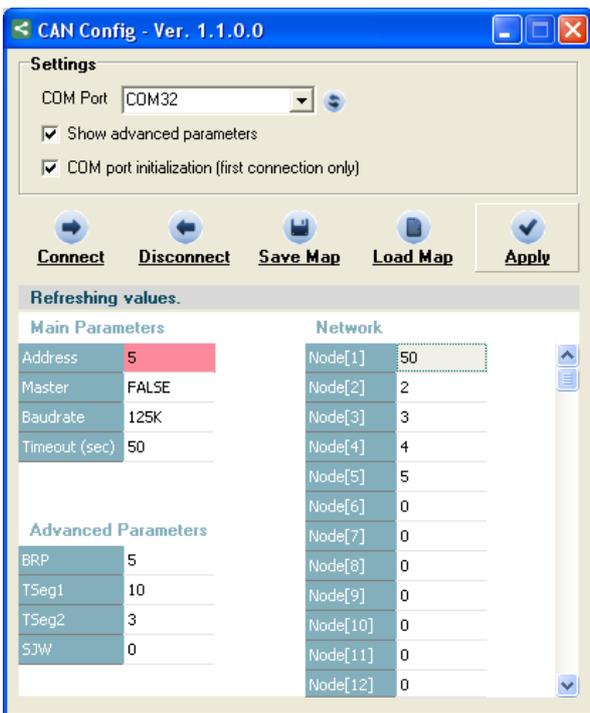
To modify the parameters when the connection is active, it is enough to write the values in the grid. The program refreshes the values every second and provides an immediate feedback (on condition the writing has succesfully been completed). To make valid the values just written and in order that the controller starts again the CAN communication using the new parameters, it is necessary to press prompt APPLY. If after the validation of the parameters some of these are overwritten by the application (for example using the libraries belonging to the group *System/CAN 1st*) the incongruity is underlined in red. To save and load maps it is necessary the communication is active; no modification is allowed when offline.

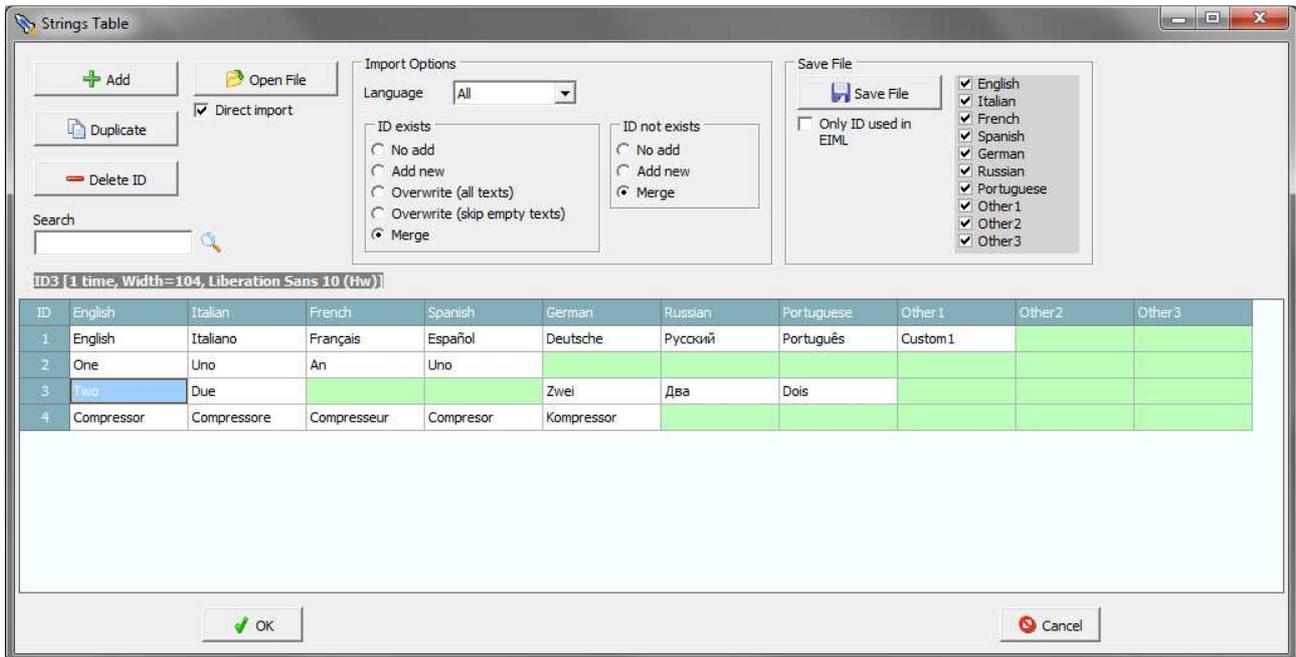Values in grey if the communication is not active.

Values in red in case of communication error.

Values underlined in red where the written values are different from the read ones. The control is done after the validation of the modifications through prompt APPLY.

4.27 Strings Table

From the menu *Tools-> StringsTable*, a tool is used that allows you to create and manipulate the language strings of the language database to be used in the EIML pages.

Above the table, a string with useful ID information is displayed:
- Number of instances of the ID on all EIML pages.
- The maximum number of characters that can be displayed when the text is associated with only EIML elements that don't use custom fonts, or the maximum size in pixels when the ID is associated with at least one element that uses a custom font.
- Custom font with greater width to which the ID is associated.

If a text exceeds the maximum number of characters that can be displayed by the EIML element on the page, it is highlighted in red. If an element has empty text it is highlighted with the color Green, otherwise it has a white background.

Add. Add a newer ID.

Duplicate. Duplicate selected ID.

Delete ID. Erase selected ID.

Open File. It allows you to open a previously saved linguistic file and to be able to modify it, in this mode the text **FILE OPEN MODE** appears to indicate that you are working on an external file and not on the real linguistic content of the project, it will be possible to import this file into that of the project.

The *Import Options* section indicates the import options:
- **Language**: only imports the selected language
- **ID exists**: action to be taken when an ID to be imported is already present in the project
    - Does not add ID.
    - Add a new ID (first free)

- Overwrites all texts with all the texts of the ID to be imported.
- Overwrites the texts with the non-empty texts of the ID to be imported, leaves the original texts unchanged if the relative text of the new ID is null.
- Merge. It adds the new IDs trying to standardize them with the existing ones, for each new ID he looks for if there is a compatible one, if it finds it it adds the new translations, otherwise a new ID is added as it is.
  - ID not exists: action to be taken when an ID to be imported is not present in the project
- Does not add ID.
- Add the new ID.
- Merge. It adds the new IDs trying to standardize them with the existing ones, for each new ID he looks for a compatible one, if it finds it it adds the new translations, otherwise the ID is added as it is.

Direct Import. It allows you to open a file with Open File and import it directly following the Import Options without first viewing the file to be imported (you do not go through the FILE OPEN MODE).

Save File. It allows you to save a file in CSV format with all the IDs and languages selected (possibly only the IDs used in the EIML by ticking the relevant option). The file can subsequently be imported into another project.
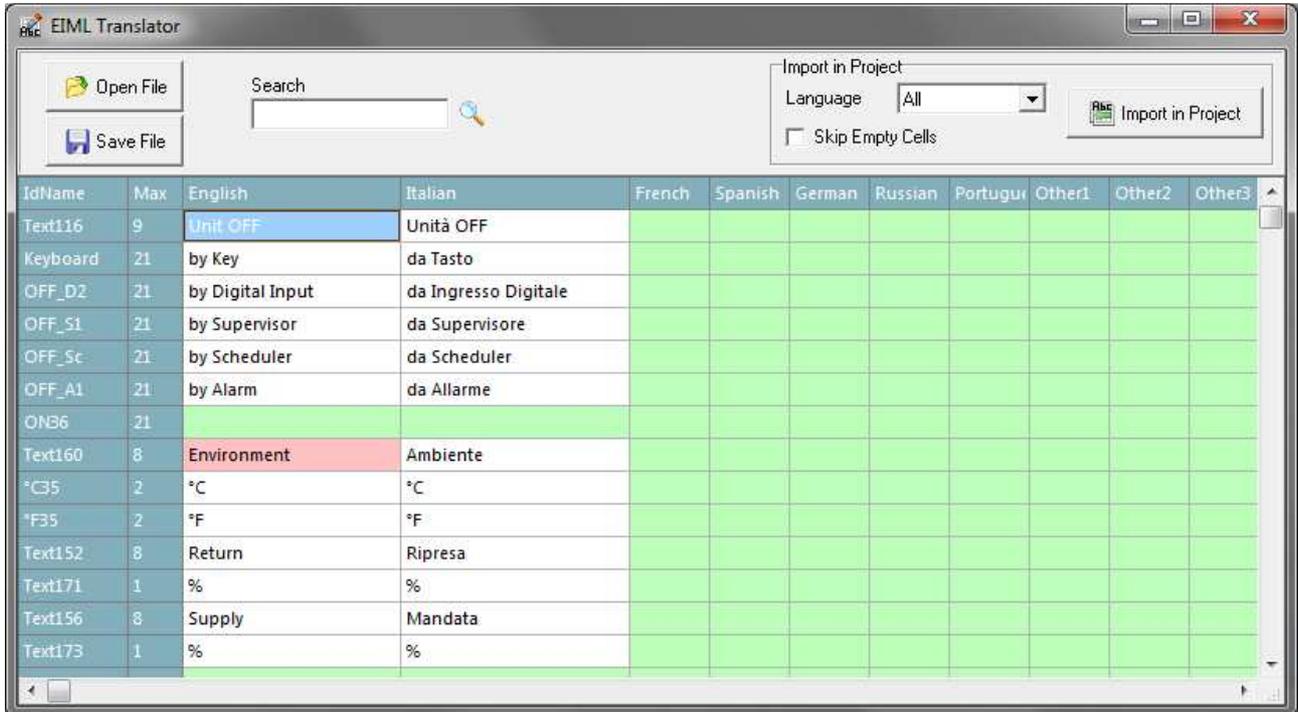
Through the drop-down menu with the right mouse button, you can perform these operations:
- **Change ID**: change the number of the ID selected in the table.
- **Delete language**: delete all strings of the selected language.
- **Delete ID not used**: delete all the IDs not used in the EIML pages.
- **Delete all**: delete all language IDs.
- **Replace ID in EIML**: Replaces in the EIML pages that use the ID selected in the table, another ID in the table (this operation is definitive, once performed it is not possible to go back).
- **Remove ID from EIML**: Permanently deletes the association of the selected ID from all the EIML pages that use it (this operation is definitive, once performed it is not possible to go back).
- **Remove all ID from EIML**: Permanently deletes the association of all the ID from all the EIML pages that use them (this operation is definitive, once performed it is not possible to go back).

It is also possible to launch the tool from the single EIML page (from the drop-down menu from the page), in this way only the linguistic IDs used on the page will be listed. The actions allowed in this mode are limited.

4.28 EIML Translator

From the menu *Tools->EIML Translator* one can access to a tool that allows to complete easily the multilanguage strings of the EIML pages. A list of all the inserted texts is shown; this tool also allows to insert the respective translations in the desired languages.

The first column indicates the name of the EIML elements, the second one shows the maximum showing size the element has in the page and on afterwards there are the translations in the languages supported by the EIML language.

If an element has an empty text, this will be underlined in green.

If a text is longer than the maximum number of characters that can be shown by the EIML element of the page, it is underlined in red.

Open File	It opens a CSV multilanguage file previously saved through the language exportation of the project (it is also possible to open files belonging to other projects)
Save File	It saves a CSV multilanguage file with all the elements there are in the multilanguage charts
Import in Project	It tries to import the elements in the running project; if an error occours during the importation an alert is shown and it is saved into a Log file (in the directory of the project)
Language	During import of the strings in the project, are imported only the elements of selected language (or ALL).
Skip Blank Cell	If selected, when the import is done in the project, the empty cells are skipped, is maintained in the project the previous string.

4.29 Entities Editor

From the menu *Tools->Entities Enditor*, you can access to a tool to view and modify some properties of the project entities.

It is a collection in table format of all the selected entities, through which it is possible to view and modify some properties without having to search for the entity in the project. It is useful above all during the verification phase when checking the correctness of the values assigned during development and / or to modify the default configurations of the application.

The properties can be edited individually by modifying the cell, some must be modified with special windows, by clicking on the "..." button that appears when editing the cell.

The gray cells indicate that the property cannot be activated for the specific entity.

The application of the changes to the project takes place only once the closing of the tool is confirmed using the **OK** button. Pressing the **Cancel** button will abort any changes made.

The managed properties are as follows:

- **Name** indicates the name of the entity. Any change to this property reloads the table, because its uniqueness must be verified (even with the entities not listed in the table).
- **Value**, **Min**, **Max** respectively represent the default value, the minimum limit and the maximum limit of the entity.
- **Par Min** and **Par Max**, indicate whether the entity (only the parameters) is limited below and above by other entities; NONE means no limitation.
- **VTP** contains the string for generating additional properties on the VTP driver file.
- **Web Server** contains the string for the correct generation of additional information for displaying the entity on the controller's web server.
- **Visibility** indicates whether the entity is subject to conditional visibility on the EIML pages.
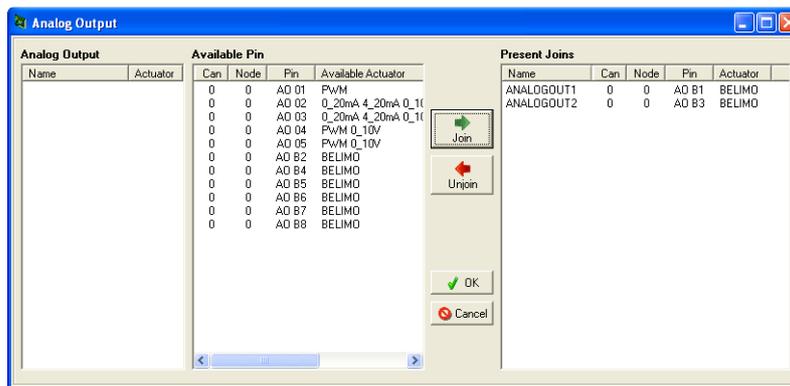
4.30 Configuration MP-Bus Belimo protocol

To the MP-Bus port can be connected the Belimo's devices that manage correctly the commands *MP_Get_SeriesNo*, *MP_Set_MP_Address* (that allow the configuration of the network) and *MP_Set_Relative* (that allows to drive the position of the actuator).

If you want to connect a probe (or a relay) to the actuator, the device must also manage the commands *MP_AD_Convert* (that allows to read the probe value) and/or *MP_Get_Forced_Control* (that allows to read the relay status).

The controllers that support the MP-Bus communication protocol allow realizing a network made of 8 (at most) actuators Belimo. Each node must be configured; the configuration can be done through a template *MPBus_SerialDeviceN* for each device. To monitor the status of each device one has to use the libraries *MPBus_GetStatusDeviceN*. To monitor the global status of the MP-Bus network one has to use libraries *MPBus_TxCounter* e *MPBus_RxCounter*.

All the libraries mentioned are related in the system libraries of the development environment UNI-PRO.

Each device connected to the MP-Bus network is used as an analog output, which signal moves from 0.00% and 100.00%; it must be inserted in the project an *Analog Output* having property "*actuator*" Belimo for each device. Using the *Join Tool* these *Analog Output* must be connected to the dedicated pins.

Each device Belimo offers the reading of a probe or, as alternative, of a relay status. If one wants to monitor the value read by the probe, one has to insert in the project an *Analog Input*, to be connected in the joining tool to the dedicated pins. The probes can be configured as Pt 1000, NTC, 0-10 V. The relay status works as a *Digital Input*, to be connected to the relative pin.

For each device Belimo it is possible to read one probe value or, as alternative, one relay status.

Please note that if one wants to use two actuators Belimo and they are configured using the libraries **MPBus_SerialDevice1** and **MPBus_SerialDevice3,** the two *Analog Output* must be connected to

pin 1 and pin 3 Belimo. In the same way, possible analog/digital inputs must be connected to the pins representing the first and the third node of the MP-Bus network.

4.31 Compatibility control for browser EPJ-Color

When in the CAN network there is a browser EPJ-Color, it is necessary to compile and download the object file (prjName_*Browser.ucjb* and prjName_*Browser.ucje*) also for the interface.

The color EIML pages, the backgrounds and some information of the project will remain in Vcolor and this is the reason why it is necessary to update also this hardware.

When the CAN communication between controller and browser begins, a compatibility control (handshake) is made. The control is required to guarantee the data in the controller and those in the browser are updated among them. The result of the handshake can be checked through the page "Checking" of the browser at the beginning of the communication.

The compatibilità test can give back three results:

- **No Compatibility**: There is no compatibility, the number of the EIML pages or the number of the project that has been downloaded and in the browser are different. In this status the browser can not communicate with the controller.
- **Partial Compatibility**: There is a partial compatibility, the EIML pages could be different or ssome entities could be modified; the reason is the modification of the data about version/project revision or the re-compiling and update of one of the parts of the project. The browser can communicate with the controller ; it is the user who knows if the communication is still valid.
- **Compatibility OK**: Full compatibility.

At the end of the handshake the interface will start the communication or will remain in "Checking" according to the result of the verification.

4.32 Precautions in project development

1) Resources may be limited

The controllers have limited RAM and STACK memory resources. They are varied for the type to control (it is advisable to check the hardware manual for the CPU characteristics of each model). Using array consumes those resources: e.g. an algorithm with two array inputs from 100 CJ_BYTE and an output from 100 CJ_BYTE uses 300 byte in the stack, in some cases it may be sufficient to break down the stack with consequent reset of the controller! The same is true if these arrays are declared locally within an algorithm.

Example:

To give a size order of the stack for the controllers used, the Mega and Giga have 1,000 byte of stack while the Nano, Micro, and Kilo are only 300 byte.

2) Using fixed point and floating point mathematics

The use of arithmetic (addition, multiplication, division ...) inside the algorithms in C language is the responsibility of the developer.

Therefore, please take care when entering the algorithm codes and check the result of the operations for the range of admissible values beforehand.

The possibility to automatically control that the result of any operation does not exceed the permitted limits (overflow) or if the divisor assumes a zero value (division by zero) does not exist. Furthermore, the 16 bit architecture of the controllers ensures, if not otherwise specified through the "casting" operations, that the temporary result of an intermediate operation are saved on a 16 bit register, with possible truncation of the value.

4.33 "Run-time" variables

While or after the develop of an application is useful to verify controller system variables, that allow to evaluate the performances and the critical points of the program, while is in function. These system variables allow to measure the cycle time of the program part under main, of the part under interrupt and to know how many byte of stack are still free. Is possible to access to this information from the configuration menu of each controller: from the local user interface (built-in) or from the remote one, must press UP+DOWN for 3 seconds to access to this menu, next select the "debug" item. It is possible to use functions from inside the algorithms to read these variables to allow a custom diagnostic. Following are described in detail these variables.

Main cycle time

Represents time in milliseconds it takes the controller to perform the main cycle, since the moment when acquire and lock all the inputs to the moment when unlock and update the outputs. Represents the time it takes to run all calls of main part call list. The variable possible to read in the "debug" menu of the first pages of configuration, represents the maximum value calculated from the last reset of controller. Editing this time to lower values is possible to determine the average time. In the algorithms is possible to use the following functions to calculate the maximum value, minimum and instantaneous of main time:

- ➢ CJ_WORD CJ_MaxMainTime(void)
- ➢ CJ_WORD CJ_MinMainTime(void)
- ➢ CJ_WORD CJ_RunMainTime(void)

The main cycle time depends on many factors, like the processor of controller used, the program dimension, the use of serial communication protocols, the presence of expansions and remote user interfaces, fixed and floating points calculations, … the measure of this time may give an indication about the program performances about the controller magnitudes. Anyway a time who exceeds 1000ms (one second) can be excessive for the execution of some program parts. Is recommended to keep an average value under 500ms.

Timed (or interrupt) Cycle time

Represents the time in milliseconds it takes the controller to run the timed cycle (under interrupt of 100ms), from it acquire and lock their timed inputs to the moment when it unlock and update their timed outputs. It represent the time it takes to run all the call list calls relative to the 5ms part and one twentieth of 100ms one. The variable can be viewed in the "debug" menu on the first configuration pages represents the maximum value calculated from the last controller reset. Editing this time to lower values is possible to assign the average time. In the algorithms can be used the following functions to calculate the maximum, minimum and instantaneous value of timed time:

- ➢ CJ_BYTE CJ_MaxInterruptTime(void)
- ➢ CJ_BYTE CJ_MinInterruptTime(void)
- ➢ CJ_BYTE CJ_RunInterruptTime(void)

The interrupt cycle time depends on many factors, like the processor of controller used, the program dimension, the use of serial communication protocols, the presence of expansions and remote user interfaces, fixed and floating points calculations, …the measure of this time may give an indication about the program performances about the controller magnitudes. Anyway a time who exceeds 5ms can be excessive for the execution of some program parts. Is recommended to keep an average value under 3ms.

Measure of free stack

The stack is the memory that the controller uses dynamically to allocate the variables inside the algorithms and function calls. each controller has a number of starting stack in proportion to the total RAM, therefore the most powerful controllers can benefit initially 2000 byte while the lowers only 300 byte. While the functioning of the controller this area will be used execution of drivers and algorithms and will be stored continuously the maximum value reached by that occupation. The variable that you can read in the "debug" menu of the configuration pages represents the minimum value of free stack calculated from the last controller reset. In the algorithms is possible to use the following functions to read the minimum value of free stack:

> CJ_WORD CJ_GetStackFree(void)

For example if this value was 0, means that there was been a memory overflow, with disastrous consequences on the stability of the program. It is recommended during the development to check this value and leave a free security zone (20-30 byte) to avoid possible inconveniences.

4.34 License registration

From the *Help* menu, select *"Regiser"* action, and the following screen appears (from which you get the license registration) :

First you need to generate the *activation code*, then you have to provide it to EVCO (via website or e-mail).
EVCO will return you the corresponding *serial code*.
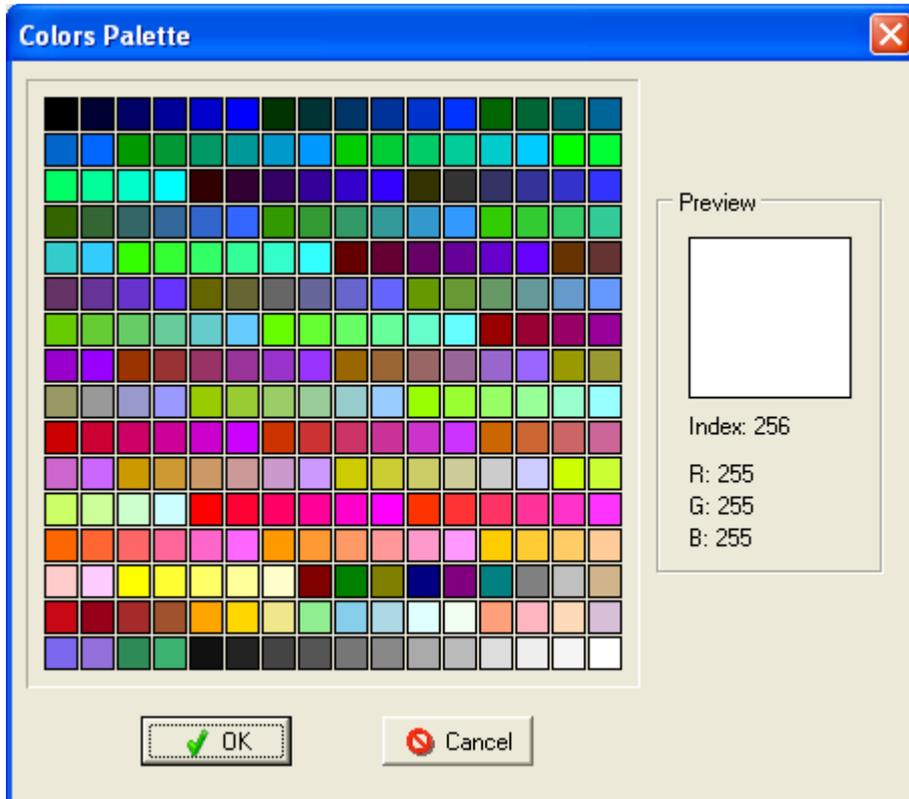
To activate the license, enter the activation code generated, the serial code received and press "Register".
Depending on the validity of the data it will display the result of the operation.

If everything is correct license will be immediately activated, otherwise you receive the error found.

To check the license status, from the *Help* menu select "*About License*".

4.35 Palette of the 256 colors of UNI-PRO

Hereinafter the full listo f the 256 colors used by UNI-PRO.

Color 1: RGB (0, 0, 0)	Color 31: RGB (0, 255, 0)	Color 61: RGB (51, 204, 0)
Color 2: RGB (0, 0, 51)	Color 32: RGB (0, 255, 51)	Color 62: RGB (51, 204, 51)
Color 3: RGB (0, 0, 102)	Color 33: RGB (0, 255, 102)	Color 63: RGB (51, 204, 102)
Color 4: RGB (0, 0, 153)	Color 34: RGB (0, 255, 153)	Color 64: RGB (51, 204, 153)
Color 5: RGB (0, 0, 204)	Color 35: RGB (0, 255, 204)	Color 65: RGB (51, 204, 204)
Color 6: RGB (0, 0, 255)	Color 36: RGB (0, 255, 255)	Color 66: RGB (51, 204, 255)
Color 7: RGB (0, 51, 0)	Color 37: RGB (51, 0, 0)	Color 67: RGB (51, 255, 0)
Color 8: RGB (0, 51, 51)	Color 38: RGB (51, 0, 51)	Color 68: RGB (51, 255, 51)
Color 9: RGB (0, 51, 102)	Color 39: RGB (51, 0, 102)	Color 69: RGB (51, 255, 102)
Color 10: RGB (0, 51, 153)	Color 40: RGB (51, 0, 153)	Color 70: RGB (51, 255, 153)
Color 11: RGB (0, 51, 204)	Color 41: RGB (51, 0, 204)	Color 71: RGB (51, 255, 204)
Color 12: RGB (0, 51, 255)	Color 42: RGB (51, 0, 255)	Color 72: RGB (51, 255, 255)
Color 13: RGB (0, 102, 0)	Color 43: RGB (51, 51, 0)	Color 73: RGB (102, 0, 0)
Color 14: RGB (0, 102, 51)	Color 44: RGB (51, 51, 51)	Color 74: RGB (102, 0, 51)
Color 15: RGB (0, 102, 102)	Color 45: RGB (51, 51, 102)	Color 75: RGB (102, 0, 102)
Color 16: RGB (0, 102, 153)	Color 46: RGB (51, 51, 153)	Color 76: RGB (102, 0, 153)
Color 17: RGB (0, 102, 204)	Color 47: RGB (51, 51, 204)	Color 77: RGB (102, 0, 204)
Color 18: RGB (0, 102, 255)	Color 48: RGB (51, 51, 255)	Color 78: RGB (102, 0, 255)
Color 19: RGB (0, 153, 0)	Color 49: RGB (51, 102, 0)	Color 79: RGB (102, 51, 0)
Color 20: RGB (0, 153, 51)	Color 50: RGB (51, 102, 51)	Color 80: RGB (102, 51, 51)
Color 21: RGB (0, 153, 102)	Color 51: RGB (51, 102, 102)	Color 81: RGB (102, 51, 102)
Color 22: RGB (0, 153, 153)	Color 52: RGB (51, 102, 153)	Color 82: RGB (102, 51, 153)
Color 23: RGB (0, 153, 204)	Color 53: RGB (51, 102, 204)	Color 83: RGB (102, 51, 204)
Color 24: RGB (0, 153, 255)	Color 54: RGB (51, 102, 255)	Color 84: RGB (102, 51, 255)
Color 25: RGB (0, 204, 0)	Color 55: RGB (51, 153, 0)	Color 85: RGB (102, 102, 0)
Color 26: RGB (0, 204, 51)	Color 56: RGB (51, 153, 51)	Color 86: RGB (102, 102, 51)
Color 27: RGB (0, 204, 102)	Color 57: RGB (51, 153, 102)	Color 87: RGB (102, 102, 102)
Color 28: RGB (0, 204, 153)	Color 58: RGB (51, 153, 153)	Color 88: RGB (102, 102, 153)
Color 29: RGB (0, 204, 204)	Color 59: RGB (51, 153, 204)	Color 89: RGB (102, 102, 204)
Color 30: RGB (0, 204, 255)	Color 60: RGB (51, 153, 255)	Color 90: RGB (102, 102, 255)

Color 91: RGB (102, 153, 0)
Color 92: RGB (102, 153, 51)
Color 93: RGB (102, 153, 102)
Color 94: RGB (102, 153, 153)
Color 95: RGB (102, 153, 204)
Color 96: RGB (102, 153, 255)
Color 97: RGB (102, 204, 0)
Color 98: RGB (102, 204, 51)
Color 99: RGB (102, 204, 102)
Color100: RGB (102, 204, 153)
Color101: RGB (102, 204, 204)
Color102: RGB (102, 204, 255)
Color103: RGB (102, 255, 0)
Color104: RGB (102, 255, 51)
Color104: RGB (102, 255, 102)
Color105: RGB (102, 255, 153)
Color106: RGB (102, 255, 204)
Color107: RGB (102, 255, 255)
Color108: RGB (153, 0, 0)
Color110: RGB (153, 0, 51)
Color111: RGB (153, 0, 102)
Color112: RGB (153, 0, 153)
Color113: RGB (153, 0, 204)
Color114: RGB (153, 0, 255)
Color115: RGB (153, 51, 0)
Color116: RGB (153, 51, 51)
Color117: RGB (153, 51, 102)
Color118: RGB (153, 51, 153)
Color119: RGB (153, 51, 204)
Color120: RGB (153, 51, 255)
Color121: RGB (153, 102, 0)
Color122: RGB (153, 102, 51)
Color123: RGB (153, 102, 102)
Color124: RGB (153, 102, 153)
Color125: RGB (153, 102, 204)
Color126: RGB (153, 102, 255)
Color127: RGB (153, 153, 0)
Color128: RGB (153, 153, 51)
Color129: RGB (153, 153, 102)
Color130: RGB (153, 153, 153)
Color131: RGB (153, 153, 204)
Color132: RGB (153, 153, 255)
Color133: RGB (153, 204, 0)
Color134: RGB (153, 204, 51)
Color135: RGB (153, 204, 102)
Color136: RGB (153, 204, 153)
Color137: RGB (153, 204, 204)
Color138: RGB (153, 204, 255)
Color139: RGB (153, 255, 0)
Color140: RGB (153, 255, 51)
Color141: RGB (153, 255, 102)
Color142: RGB (153, 255, 153)
Color143: RGB (153, 255, 204)
Color144: RGB (153, 255, 255)
Color145: RGB (204, 0, 0)
Color146: RGB (204, 0, 51)
Color147: RGB (204, 0, 102)
Color148: RGB (204, 0, 153)
Color149: RGB (204, 0, 204)
Color150: RGB (204, 0, 255)
Color151: RGB (204, 51, 0)

Color152: RGB (204, 51, 51)
Color153: RGB (204, 51, 102)
Color154: RGB (204, 51, 153)
Color155: RGB (204, 51, 204)
Color156: RGB (204, 51, 255)
Color157: RGB (204, 102, 0)
Color158: RGB (204, 102, 51)
Color159: RGB (204, 102, 102)
Color160: RGB (204, 102, 153)
Color161: RGB (204, 102, 204)
Color162: RGB (204, 102, 255)
Color163: RGB (204, 153, 0)
Color164: RGB (204, 153, 51)
Color165: RGB (204, 153, 102)
Color166: RGB (204, 153, 153)
Color167: RGB (204, 153, 204)
Color168: RGB (204, 153, 255)
Color169: RGB (204, 204, 0)
Color170: RGB (204, 204, 51)
Color171: RGB (204, 204, 102)
Color172: RGB (204, 204, 153)
Color173: RGB (204, 204, 204)
Color174: RGB (204, 204, 255)
Color175: RGB (204, 255, 0)
Color176: RGB (204, 255, 51)
Color177: RGB (204, 255, 102)
Color178: RGB (204, 255, 153)
Color179: RGB (204, 255, 204)
Color180: RGB (204, 255, 255)
Color181: RGB (255, 0, 0)
Color182: RGB (255, 0, 51)
Color183: RGB (255, 0, 102)
Color184: RGB (255, 0, 153)
Color185: RGB (255, 0, 204)
Color186: RGB (255, 0, 255)
Color187: RGB (255, 51, 0)
Color188: RGB (255, 51, 51)
Color189: RGB (255, 51, 102)
Color190: RGB (255, 51, 153)
Color191: RGB (255, 51, 204)
Color192: RGB (255, 51, 255)
Color193: RGB (255, 102, 0)
Color194: RGB (255, 102, 51)
Color195: RGB (255, 102, 102)
Color196: RGB (255, 102, 153)
Color197: RGB (255, 102, 204)
Color198: RGB (255, 102, 255)
Color199: RGB (255, 153, 0)
Color200: RGB (255, 153, 51)
Color201: RGB (255, 153, 102)
Color202: RGB (255, 153, 153)
Color203: RGB (255, 153, 204)
Color204: RGB (255, 153, 255)
Color205: RGB (255, 204, 0)
Color206: RGB (255, 204, 51)
Color207: RGB (255, 204, 102)
Color208: RGB (255, 204, 153)
Color209: RGB (255, 204, 204)
Color210: RGB (255, 204, 255)
Color211: RGB (255, 255, 0)
Color212: RGB (255, 255, 51)

Color213: RGB (255, 255, 102)
Color214: RGB (255, 255, 153)
Color215: RGB (255, 255, 204)
Color216: RGB (128, 0, 0)
Color217: RGB (0, 128, 0)
Color218: RGB (128, 128, 0)
Color219: RGB (0, 0, 128)
Color220: RGB (128, 0, 128)
Color221: RGB (0, 128, 128)
Color222: RGB (128, 128, 128)
Color223: RGB (192, 192, 192)
Color224: RGB (210, 180, 140)
Color225: RGB (200, 8, 21)
Color226: RGB (150, 0, 24)
Color227: RGB (165, 42, 42)
Color228: RGB (160, 82, 45)
Color229: RGB (255, 165, 0)
Color230: RGB (255, 215, 0)
Color231: RGB (240, 230, 140)
Color232: RGB (144, 238, 144)
Color233: RGB (135, 206, 235)
Color234: RGB (173, 216, 230)
Color235: RGB (224, 255, 255)
Color236: RGB (240, 255, 240)
Color237: RGB (255, 160, 122)
Color238: RGB (255, 182, 193)
Color239: RGB (255, 218, 185)
Color240: RGB (216, 191, 216)
Color241: RGB (123, 104, 238)
Color242: RGB (147, 112, 219)
Color243: RGB (46, 139, 87)
Color244: RGB (60, 179, 113)
Color245: RGB (17, 17, 17)
Color246: RGB (34, 34, 34)
Color247: RGB (68, 68, 68)
Color248: RGB (85, 85, 85)
Color249: RGB (119, 119, 119)
Color250: RGB (136, 136, 136)
Color251: RGB (170, 170, 170)
Color252: RGB (187, 187, 187)
Color253: RGB (224, 224, 224)
Color254: RGB (238, 238, 238)
Color255: RGB (245, 245, 245)
Color256: RGB (255, 255, 255)

5 WHAT'S NEW IN VERSION 3.17.0.0

5.1 New features

Added C-Pro 3 Giga and C-Pro 3 Giga+ family controllers.

Modified multilingual in EIML pages, new method where each text is associated with an ID in a linguistic database.

Added String Table tool for the management of linguistic strings.

Added Entities Editor tool to quickly edit some properties of entities.

Added option in the settings to load or not load the Application libraries at UNI-Pro startup.

Added option in the settings to save/restore watches and breakpoints after a controller reprogramming.

Increased to 50 the maximum number of watches for the debugger

Add array for CJ_TIME entities.

Added Multiselect mode for multiple operations on EIML pages.

The maximum number of columns in the EIML 320x240 and 320x240 16C pages has been extended to 8.

Allowed to save the project from the algorithm editor.

UNI-PRO 3 - Software manual.

Version 4.4 – July 2020.

Code 114UP3E444.

File 114UP3E444.pdf.

HEADQUARTERS
Evco
Via Mezzaterra 6, 32036 Sedico Belluno ITALY
Tel. +39 0437-852468
Fax +39 0437-83648
info@evco.it
www.evco.it

OVERSEAS OFFICES
Control France
155 Rue Roger Salengro, 92370 Chaville Paris FRANCE
Tel. 0033-1-41159740
Fax 0033-1-41159739
control.france@wanadoo.fr

Evco Latina
Larrea, 390 San Isidoro, 1609 Buenos Aires ARGENTINA
Tel. 0054-11-47351031
Fax 0054-11-47351031
evcolatina@anykasrl.com.ar

Evco Pacific
59 Premier Drive Campbellfield, 3061, Victoria Melbourne, AUSTRALIA
Tel. 0061-3-9357-0788
Fax 0061-3-9357-7638
everycontrol@pacific.com.au

Evco Russia
111141 Russia Moscow 2-oy Proezd Perova Polya 9
Tel. 007-495-3055884
Fax 007-495-3055884
info@evco.ru

Every Control do Brasil
Rua Marino Félix 256, 02515-030 Casa Verde São Paulo SÃO PAULO BRAZIL
Tel. 0055-11-38588732
Fax 0055-11-39659890
info@everycontrol.com.br

Every Control Norden
Cementvägen 8, 136 50 Haninge SWEDEN
Tel. 0046-8-940470
Fax 0046-8-6053148
mail2@unilec.se

Every Control Shangai
B 302, Yinhai Building, 250 Cao Xi Road, 200235 Shangai CHINA
Tel. 0086-21-64824650
Fax 0086-21-64824649
evcosh@online.sh.cn

Every Control United Kingdom
Unit 19, Monument Business Park, OX44 7RW Chalgrowe, Oxford, UNITED KINGDOM
Tel. 0044-1865-400514
Fax 0044-1865-400419
info@everycontrol.co.uk