UNI-PRO

DEVELOPMENT ENVIRONMENT FOR PROGRAMMABLE CONTROLLERS

SOFTWARE MANUAL

CODE 114UPROSWE26

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

1 GENERAL INFORMATION

1.1 About UNI-PRO

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:

- Pentium® processor @ 800 MHz
  (1.6 Ghz recommended)
- 512 MB RAM
  (1 GB recommended)
- n° 1 COM serial port
- n° 2 USB ports
- 300 MB hard disk space
- SVGA video card, 800x600 resolution 128 MB
  (1024x768 256MB or higher recommended)
- CD-ROM drive

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

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 (it is included in the UNI-PRO installation CD-ROM under the folder [CDROM]:\Acrobat Reader).

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

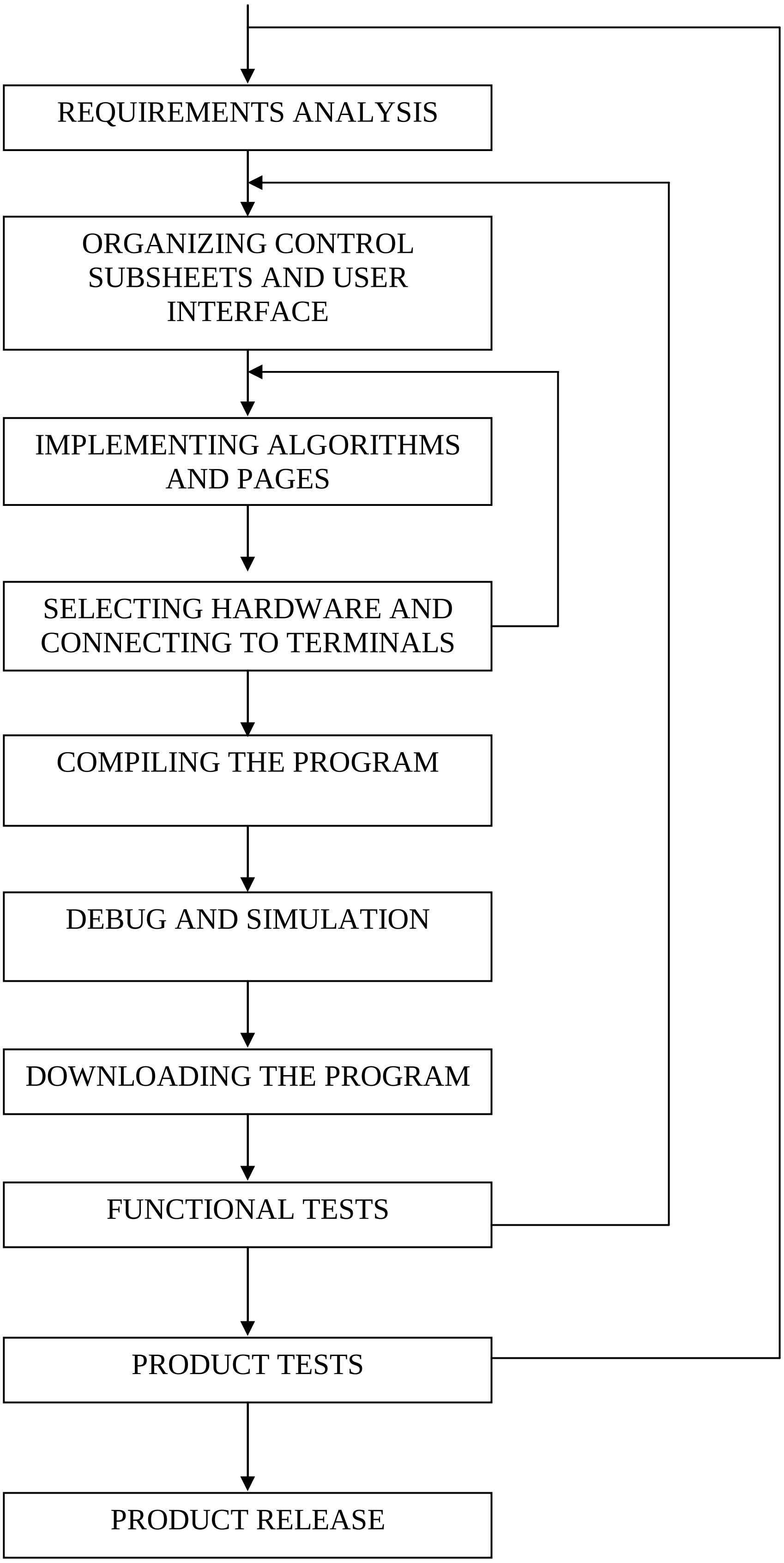
Through the UNI-PRO development environment you can create and manage distributed control projects, no matter how large they may be.

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.

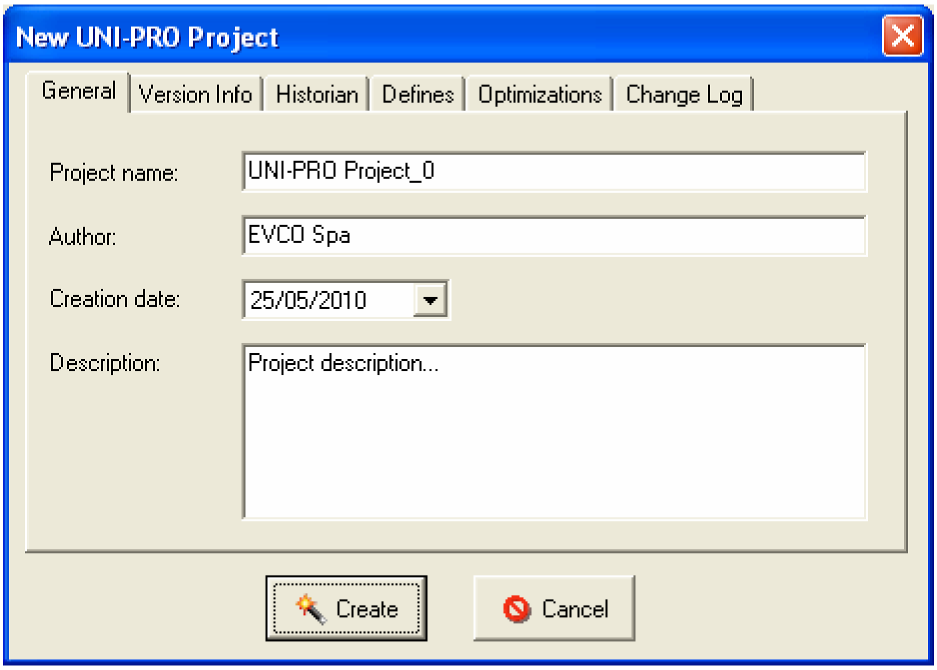
The use of UNI-PRO to develop a project should follow an iterative and incremental approach, according to the following diagram:



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.



Project name: UNI-PRO Project_0

Author: EVCO Spa

Creation date: 25/05/2010

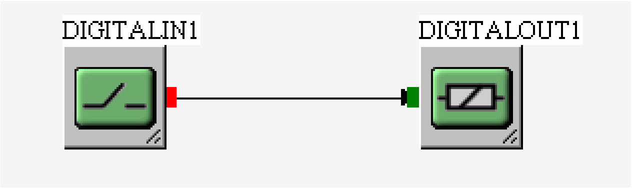
Description: Project description...

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.

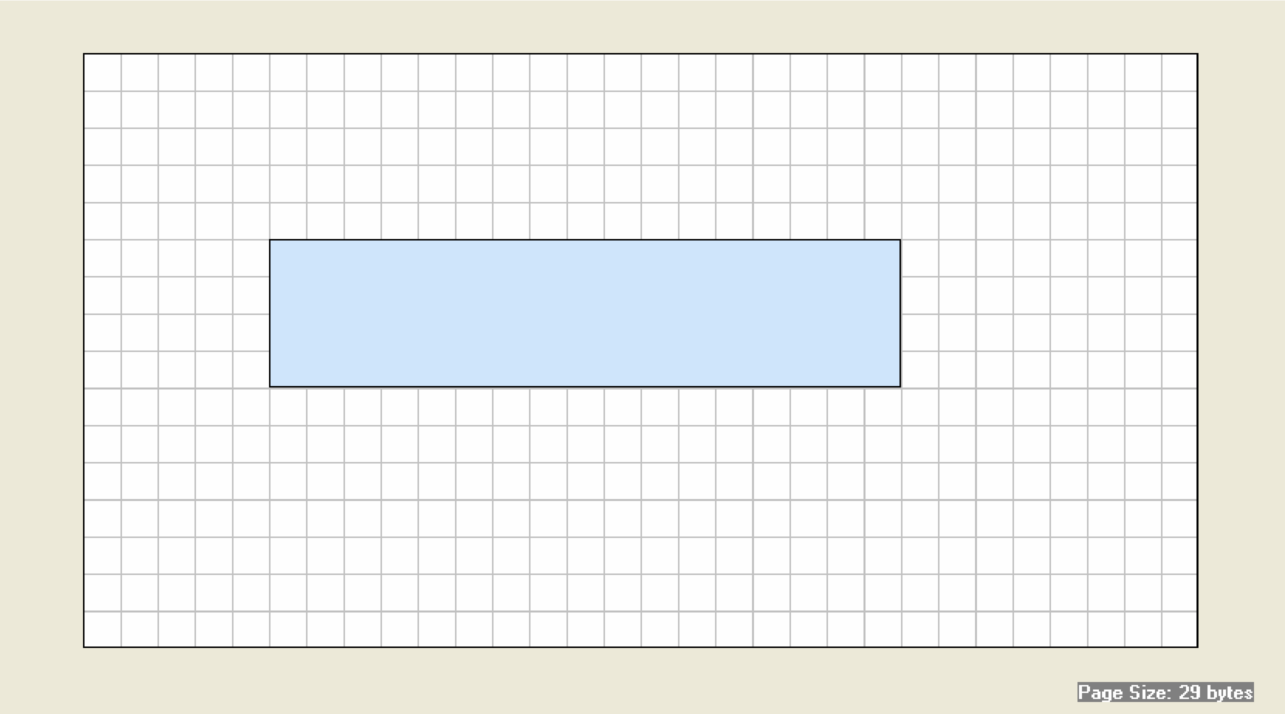
Once both entities have been added to the main worksheet of the application, we will need to join them. To do so, we place the mouse pointer over the digital input's output until it changes size. We click and notice that an arrow is drawn from the digital input's **output pin** to the pointer. We place 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.



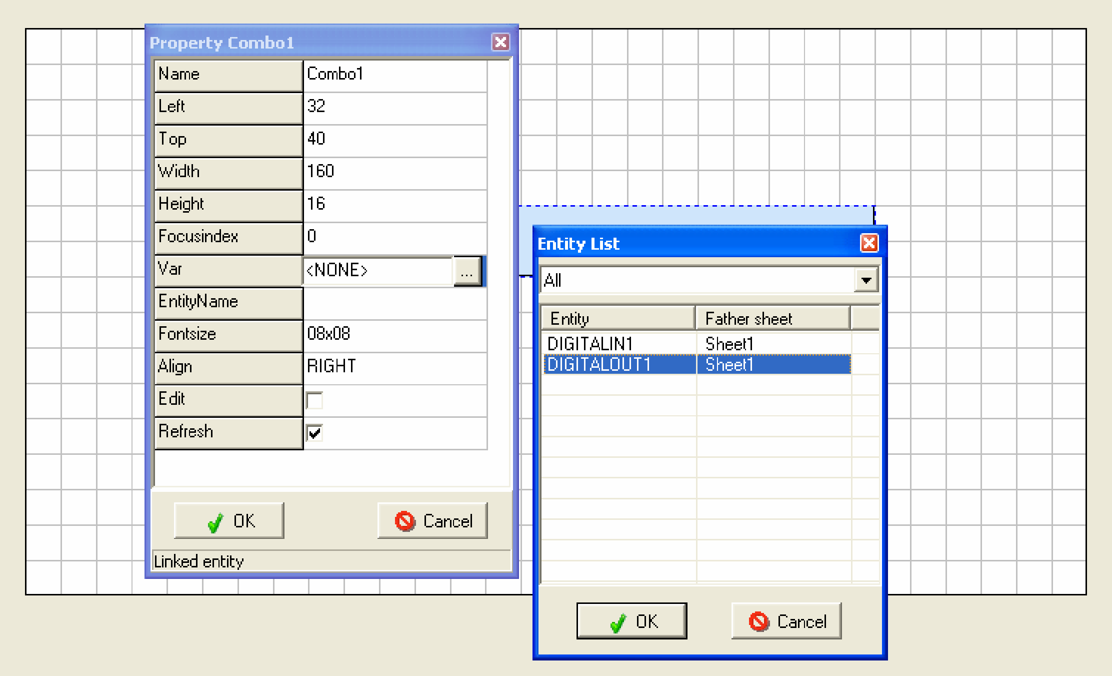
To save the project, choose *File / Save Project* from the menu, or click the corresponding icon in the toolbar.

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:



Property Combo1

Name	Combo1
Left	32
Top	40
Width	160
Height	16
Focusindex	0
Var	<NONE>
EntityName	
Fontsize	08x08
Align	RIGHT
Edit	□
Refresh	☑

Linked entity

Entity List

All

Entity	Father sheet
DIGITALIN1	Sheet1
DIGITALOUT1	Sheet1

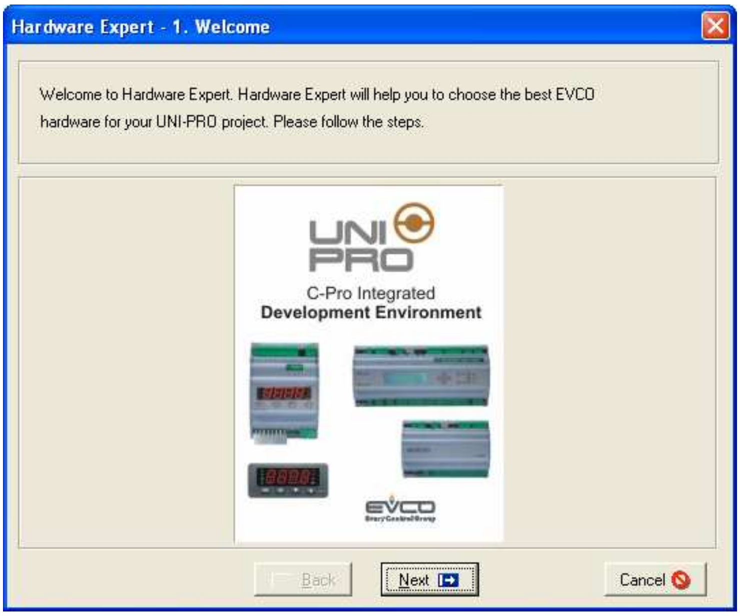
In this way we have created a join between the DigitalOut variable and the Combo1 graphic element that represents it.

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.

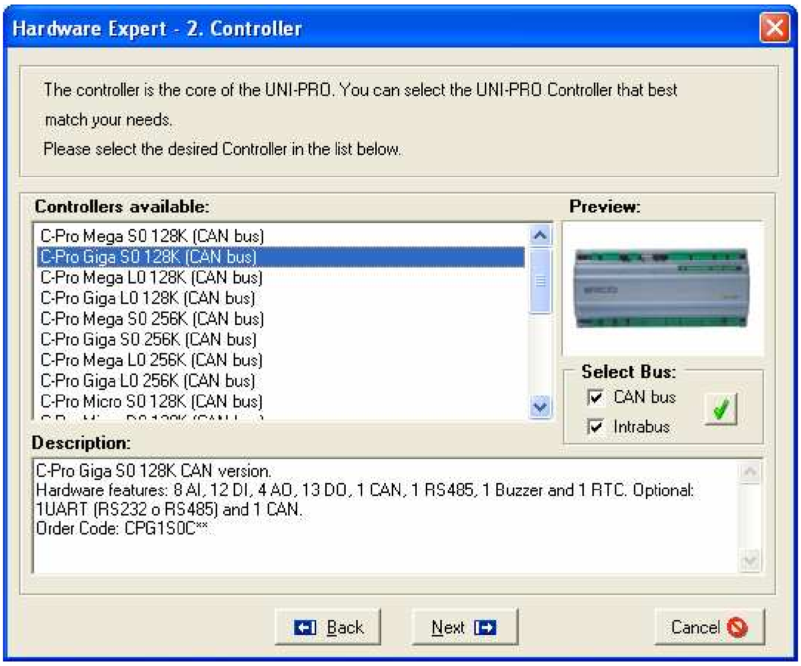
If we had not selected the hardware on which the project will be run yet, the **Hardware Expert** will appear.



Hardware Expert - 1. Welcome

Welcome to Hardware Expert. Hardware Expert will help you to choose the best EVCO hardware for your UNI-PRO project. Please follow the steps.

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.



Hardware Expert - 2. Controller

The controller is the core of the UNI-PRO. You can select the UNI-PRO Controller that best match your needs.
Please select the desired Controller in the list below.

Controllers available:

C-Pro Mega S0 128K (CAN bus)
C-Pro Giga S0 128K (CAN bus)
C-Pro Mega L0 128K (CAN bus)
C-Pro Giga L0 128K (CAN bus)
C-Pro Mega S0 256K (CAN bus)
C-Pro Giga S0 256K (CAN bus)
C-Pro Mega L0 256K (CAN bus)
C-Pro Giga L0 256K (CAN bus)
C-Pro Micro S0 128K (CAN bus)

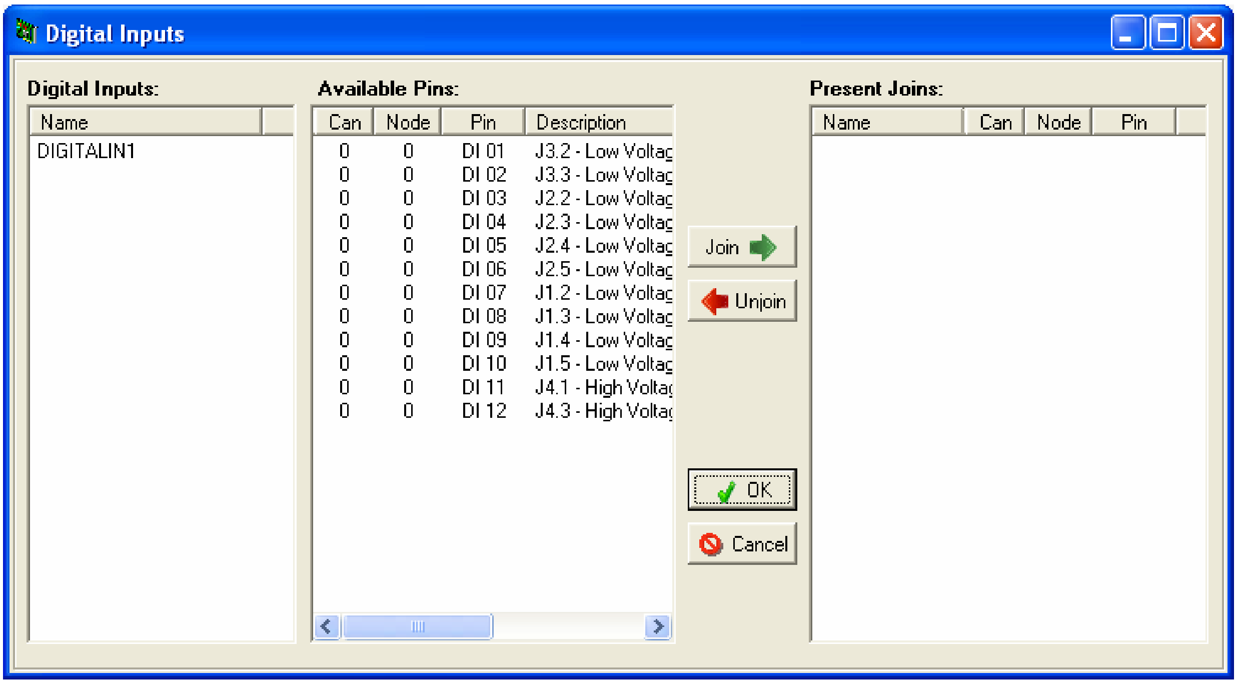
Preview:

Select Bus:
☑ CAN bus
☑ Intrabus

Description:

C-Pro Giga S0 128K CAN version.
Hardware features: 8 AI, 12 DI, 4 AO, 13 DO, 1 CAN, 1 RS485, 1 Buzzer and 1 RTC. Optional: 1UART (RS232 o RS485) and 1 CAN.
Order Code: CPG1S0C**

We will select the most suitable controller (e.g. CProPlus_S0) 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.



Digital Inputs:

Name
DIGITALIN1

Available Pins:

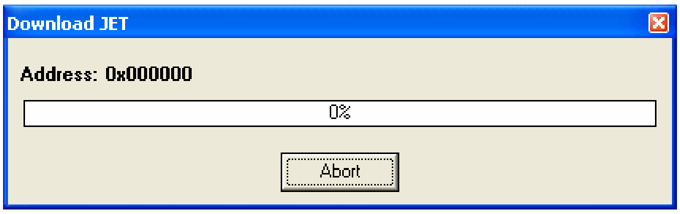
Can	Node	Pin	Description
0	0	DI 01	J3.2 - Low Voltag
0	0	DI 02	J3.3 - Low Voltag
0	0	DI 03	J2.2 - Low Voltag
0	0	DI 04	J2.3 - Low Voltag
0	0	DI 05	J2.4 - Low Voltag
0	0	DI 06	J2.5 - Low Voltag
0	0	DI 07	J1.2 - Low Voltag
0	0	DI 08	J1.3 - Low Voltag
0	0	DI 09	J1.4 - Low Voltag
0	0	DI 10	J1.5 - Low Voltag
0	0	DI 11	J4.1 - High Voltag
0	0	DI 12	J4.3 - High Voltag

Present Joins:

Name	Can	Node	Pin

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:



Address: 0x000000

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:

1. 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.
2. The serial cable must be connected both to the computer and to the controller

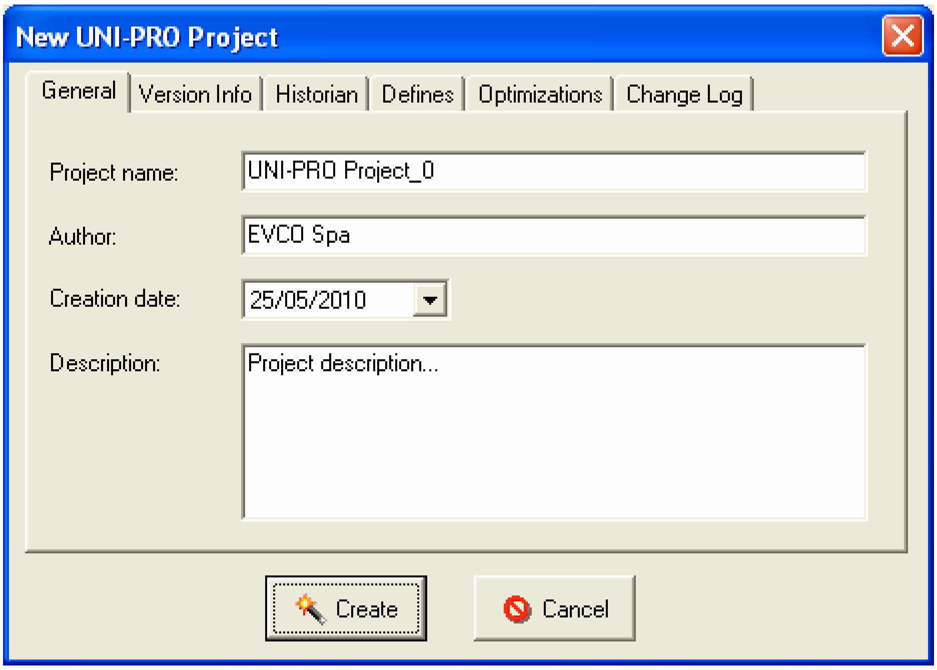
Having terminated the download operations, your first program should have been loaded in the CproPlus_S0: if you now close the contact where the DigitalIn has been connected, the relay corresponding to the DigitalOut should trigger and the words “Hello World!” should appear on the graphic viewer.

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.



Project name: UNI-PRO Project_0

Author: EVCO Spa

Creation date: 25/05/2010

Description: Project description...

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).

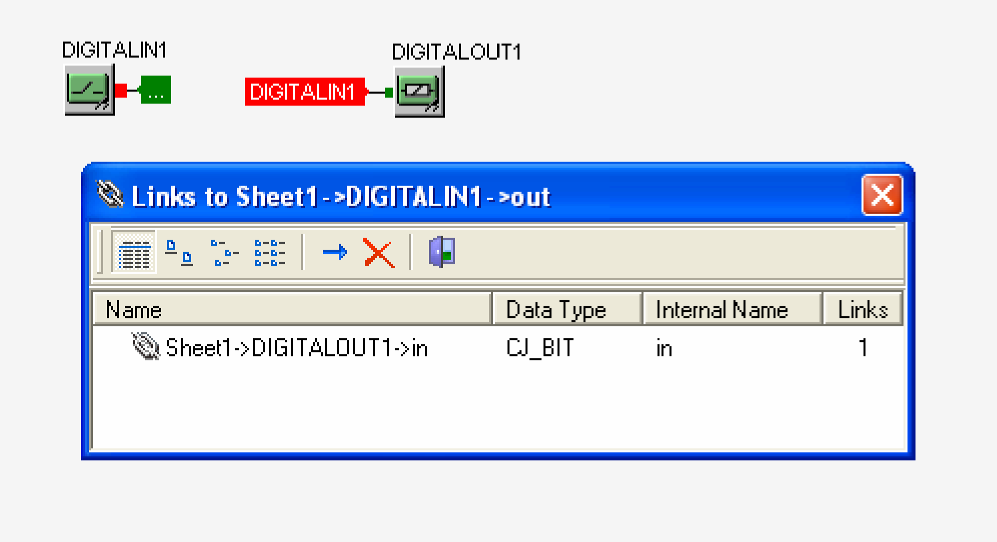
2.5 Joining control entities

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 in the Software palette, select the output you wish to join, and select the input. 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.



Links to Sheet1->DIGITALIN1->out

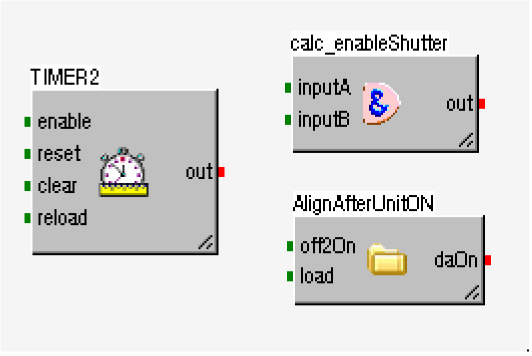
Name	Data Type	Internal Name	Links
Sheet1->DIGITALOUT1->in	CJ_BIT	in	1

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

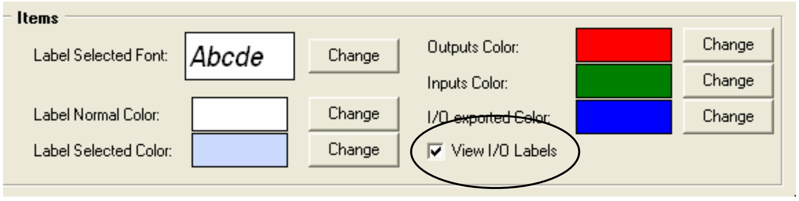
To unjoin two entities, you can proceed in two different ways according to the type of link:

- 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*



Label Selected Font: Abcde Change

Label Normal Color: Change

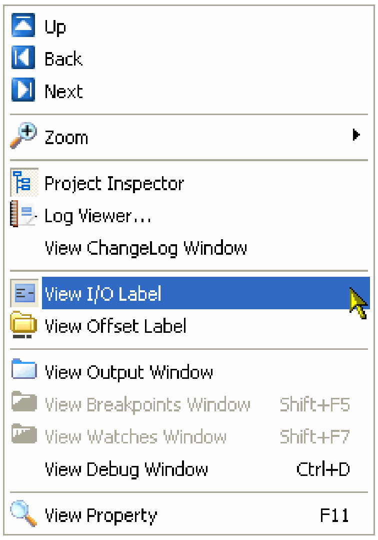
Label Selected Color: Change

Outputs Color: 🟥 Change

Inputs Color: 🟩 Change

I/O exported Color: 🟦 Change

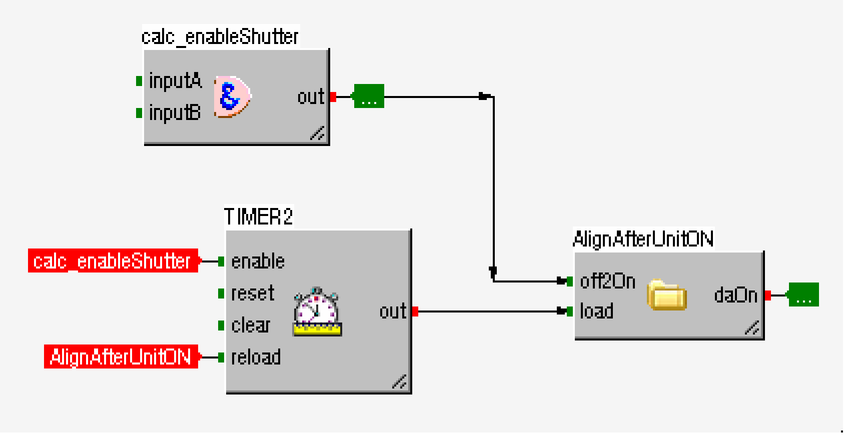
☑ View I/O Labels



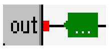
or in the Menu *View->View I/O Label*.

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

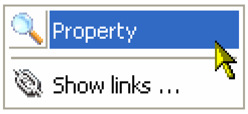
2.5.2 Names to the external links



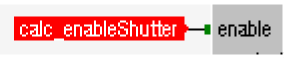
Link Out

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

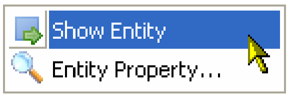
Drop-down menu
Property: link properties
Show Links: window with the connected links



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

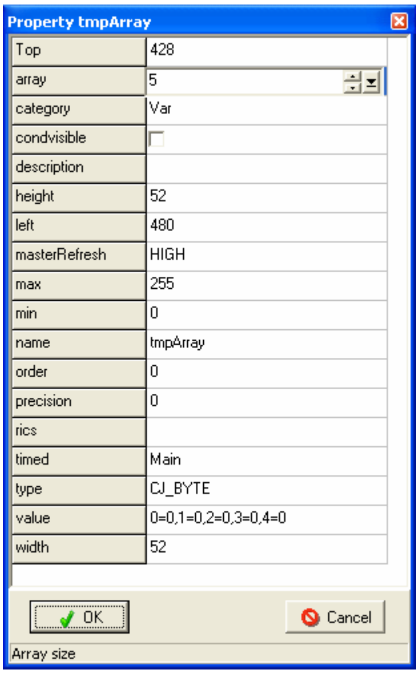


2.6 Using array entities

"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. 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.

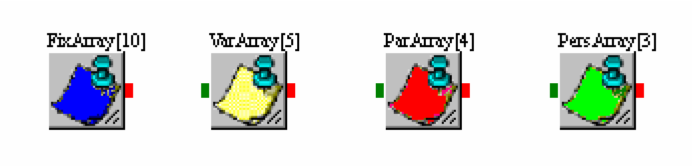
When the data type are different from the over reported data, the *array* is not modifiable, but fixed (read-only) with value 1. For any other entities different of PAR – FIX – PERS – VAR the propriety ***array*** is not available.



Property tmpArray

Top	428
array	5
category	Var
condvisible	□
description	
height	52
left	480
masterRefresh	HIGH
max	255
min	0
name	tmpArray
order	0
precision	0
rics	
timed	Main
type	CJ_BYTE
value	0=0,1=0,2=0,3=0,4=0
width	52

Array size



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

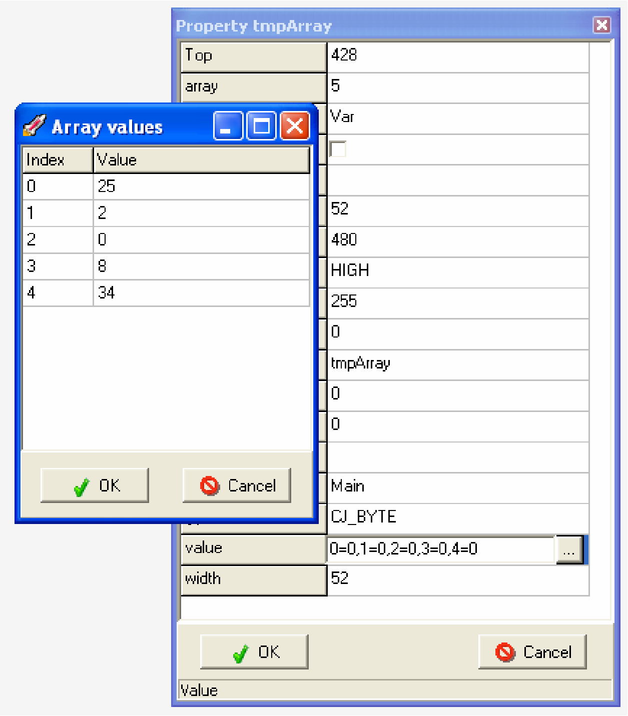
Array property is a numeric type that has 100 as maximum value and 1 as minimum value (mono-dimensional entity) : when array data value is greater than 1, the property value becomes a string to match array index with the correct array value. The syntax are following:

0=value[0], …, *array-1*=value[*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:

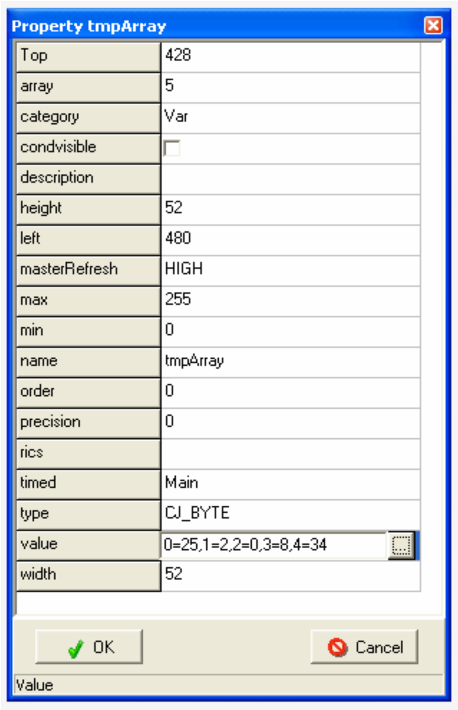


Array values

Index	Value
0	25
1	2
2	0
3	8
4	34

Property tmpArray

Top	428
array	5
	Var
	□
	52
	480
	HIGH
	255
	0
	tmpArray
	0
	0
	Main
	CJ_BYTE
value	0=0,1=0,2=0,3=0,4=0
width	52



Property tmpArray

Top	428
array	5
category	Var
condvisible	□
description	
height	52
left	480
masterRefresh	HIGH
max	255
min	0
name	tmpArray
order	0
precision	0
rics	
timed	Main
type	CJ_BYTE
value	0=25,1=2,2=0,3=8,4=34
width	52

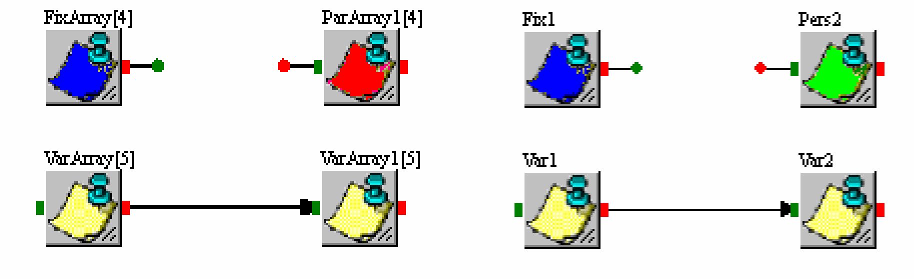
Clicking on **Ok** button, the edited values will be accepted and automatically set up the *value* property like showed in the picture.
Clicking on **Cancel** button, the edited values are not accepted and will be maintained the previous values.

2.6.1 Compatibility and link

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

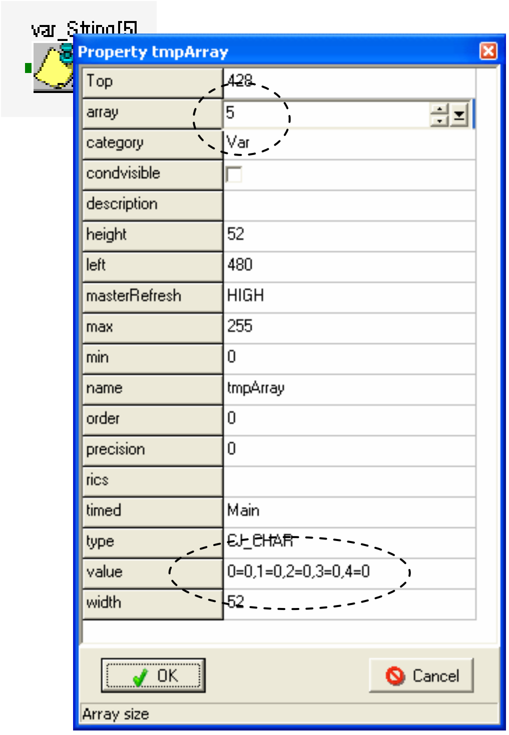
1. if both the entities has the same array size (same *array* property value)
2. if the entities are of the same data type, with the exception :
    a. when start entity data type is CJ_BIT and linked entity data type is CJ_S_BYTE
    b. when start entity data type is CJ_BIT and linked entity data type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

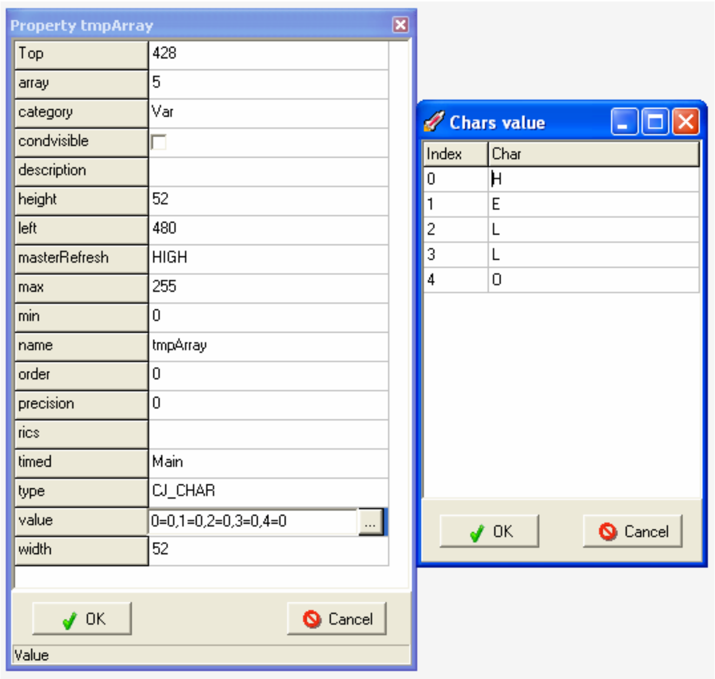
By using type CJ_CHAR and the array property, is possible to manage a maximum 100 characters string. To use this function is necessary to set as entity type CJ_CHAR and the string characters number setting array property.



Top	428
array	5
category	Var
condvisible	□
description	
height	52
left	480
masterRefresh	HIGH
max	255
min	0
name	tmpArray
order	0
precision	0
rics	
timed	Main
type	CJ_CHAR
value	0=0,1=0,2=0,3=0,4=0
width	52

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:



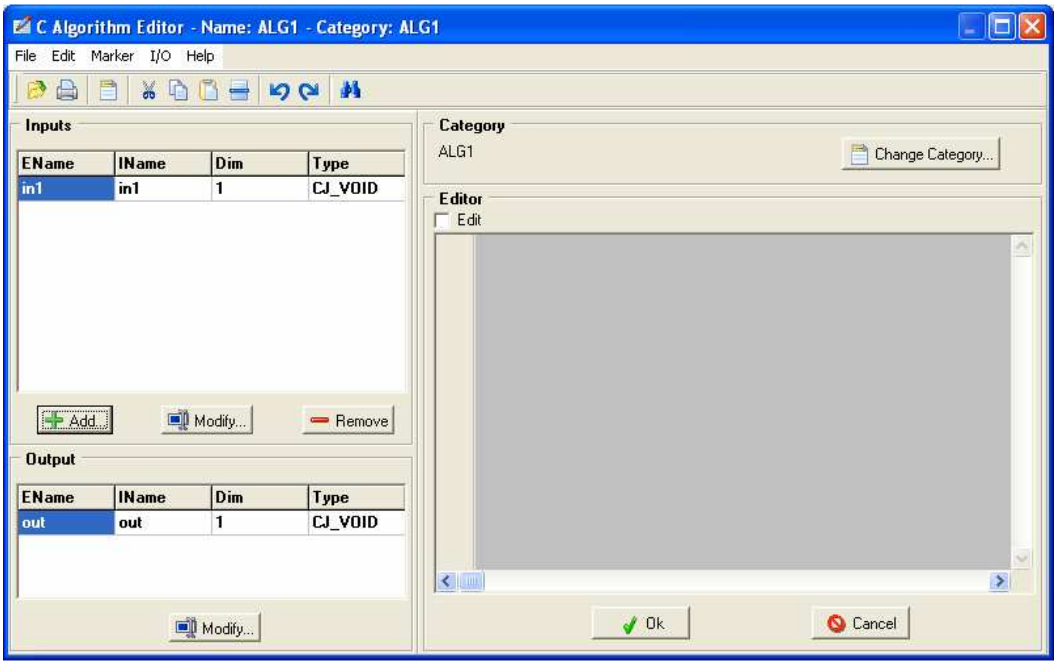
Top	428
array	5
category	Var
condvisible	□
description	
height	52
left	480
masterRefresh	HIGH
max	255
min	0
name	tmpArray
order	0
precision	0
rics	
timed	Main
type	CJ_CHAR
value	0=0,1=0,2=0,3=0,4=0 ...
width	52

Index	Char
0	H
1	E
2	L
3	L
4	O

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.



Inputs

EName	IName	Dim	Type
in1	in1	1	CJ_VOID

Output

EName	IName	Dim	Type
out	out	1	CJ_VOID

Category

ALG1

Editor

□ Edit

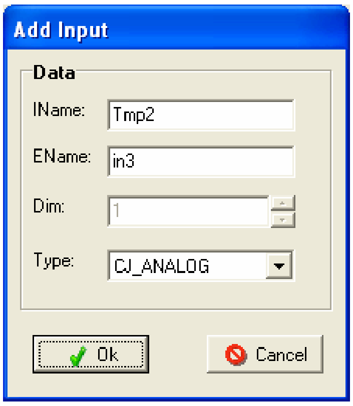
An algorithm is characterized by a series of inputs, a single output and a code that describes its behavior.

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.

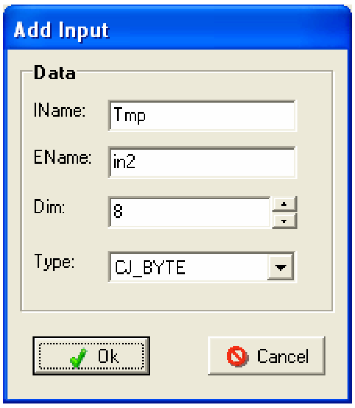
The input table shows their type and number; three properties are specified for each input:

- **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:



IName: Tmp2
EName: in3
Dim: 1
Type: CJ_ANALOG

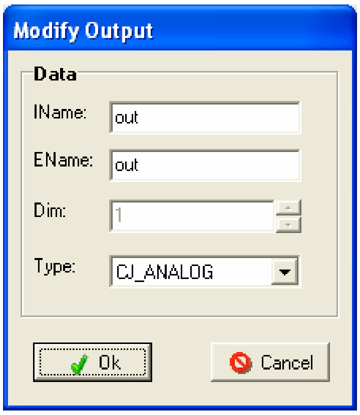


IName: Tmp
EName: in2
Dim: 8
Type: CJ_BYTE

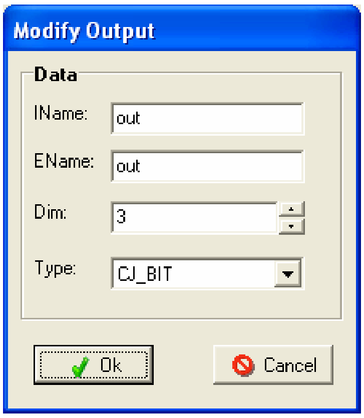
where you can enter the new input's name, size and type.

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.



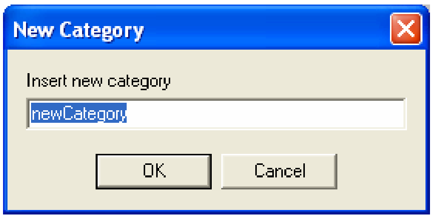
IName: out
EName: out
Dim: 1
Type: CJ_ANALOG



IName: out
EName: out
Dim: 3
Type: CJ_BIT

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**).

Press the *Change category…*, the following window will be displayed:



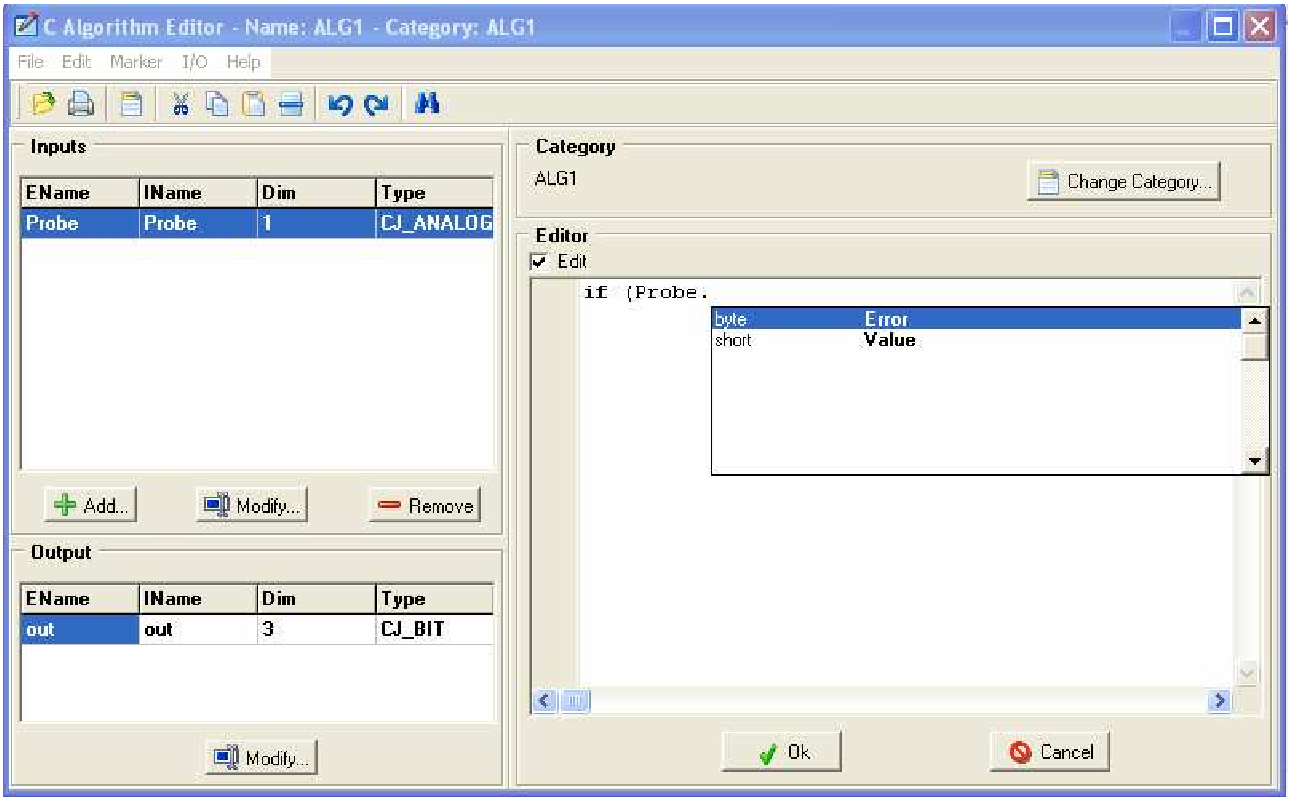
Insert new category
newCategory

where you can enter a new category.

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:



Inputs

EName	IName	Dim	Type
Probe	Probe	1	CJ_ANALOG

Output

EName	IName	Dim	Type
out	out	3	CJ_BIT

Category: ALG1

```
if (Probe.
```

byte	Error
short	Value

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

- 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!

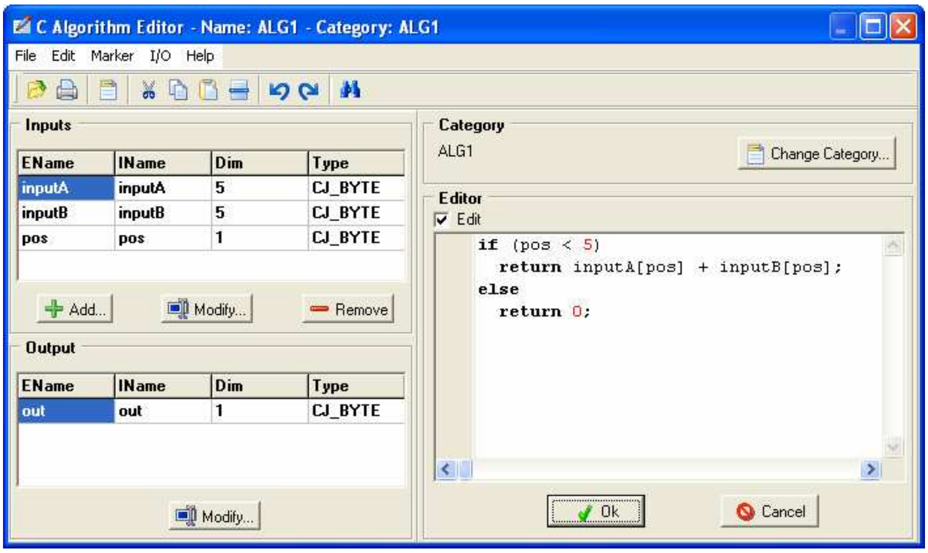
The algorithm code may be defined at any time, provided that it is done prior to compiling.

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.



Inputs

EName	IName	Dim	Type
inputA	inputA	5	CJ_BYTE
inputB	inputB	5	CJ_BYTE
pos	pos	1	CJ_BYTE

Output

EName	IName	Dim	Type
out	out	1	CJ_BYTE

Category

ALG1

Editor

☑ Edit

```
if (pos < 5)
  return inputA[pos] + inputB[pos];
else
  return 0;
```

In this example the two inputs have the same array size, but they could have different size.

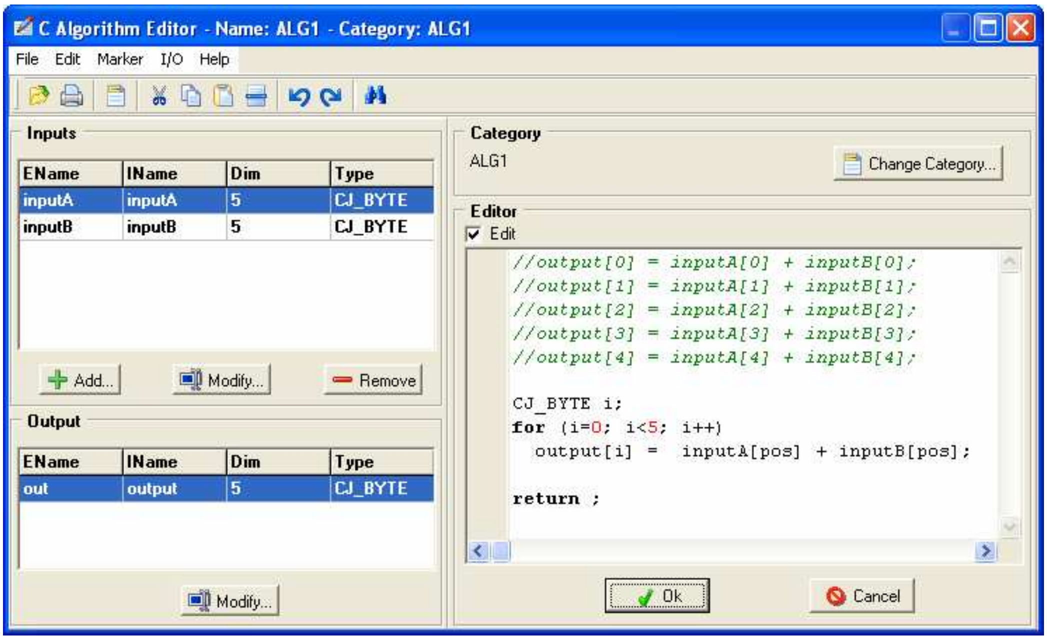
Both of the two inputs can contain five numeric information. For the access in every cell of the array use the following C syntax :

***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.



Inputs

EName	IName	Dim	Type
inputA	inputA	5	CJ_BYTE
inputB	inputB	5	CJ_BYTE

Output

EName	IName	Dim	Type
out	output	5	CJ_BYTE

```
//output[0] = inputA[0] + inputB[0];
//output[1] = inputA[1] + inputB[1];
//output[2] = inputA[2] + inputB[2];
//output[3] = inputA[3] + inputB[3];
//output[4] = inputA[4] + inputB[4];

CJ_BYTE i;
for (i=0; i<5; i++)
  output[i] =  inputA[pos] + inputB[pos];

return ;
```

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.

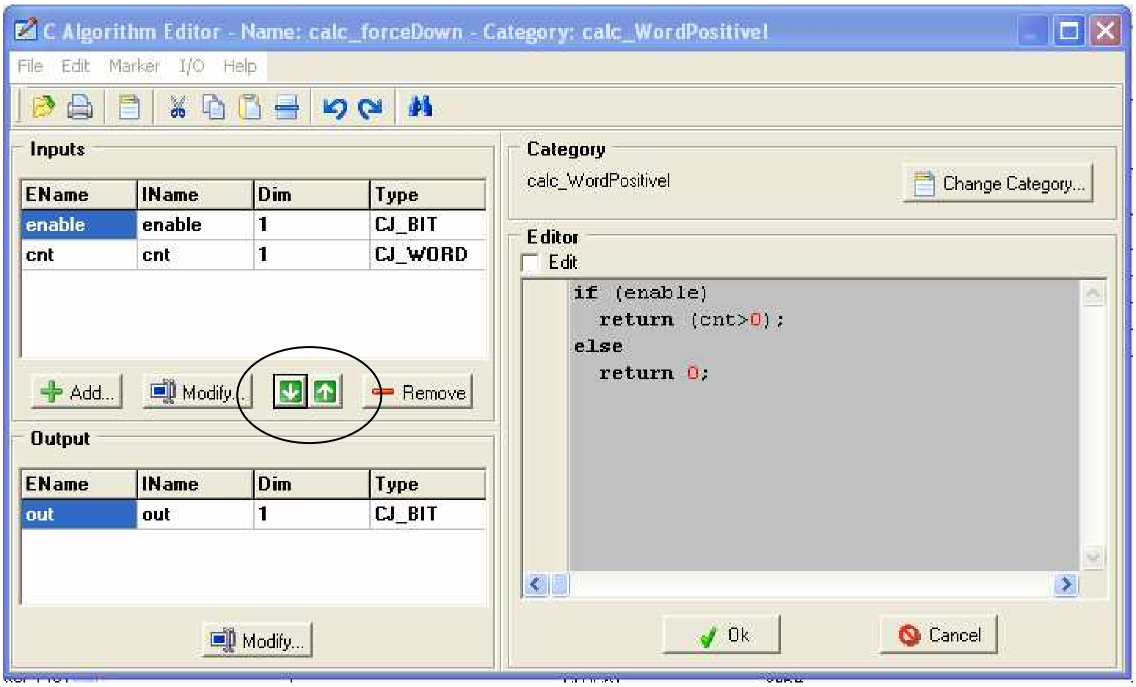
Attention to write the keyword **return** in the following way: ***"return;* "**. As told before it is not involved in the elaboration, but it has to be reported in it : if not the compiler will give you a *Warning* during the project compilation.

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”.



calc_WordPositiveI

Inputs

EName	IName	Dim	Type
enable	enable	1	CJ_BIT
cnt	cnt	1	CJ_WORD

Output

EName	IName	Dim	Type
out	out	1	CJ_BIT

```
if (enable)
  return (cnt>0);
else
  return 0;
```

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

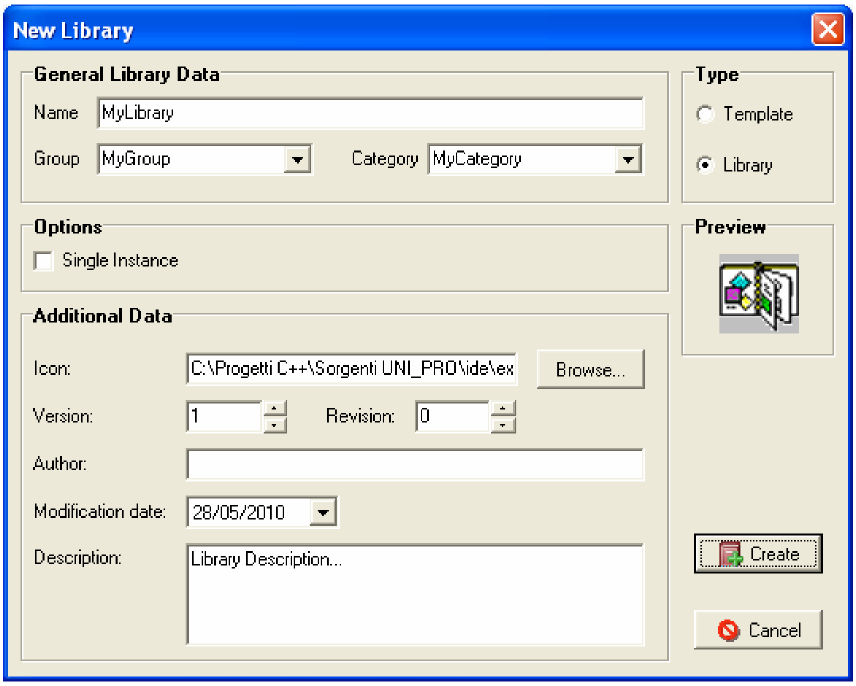
2.9 Creating libraries

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 in the toolbar

The following window will appear:



New Library

General Library Data

Name: MyLibrary

Group: MyGroup

Category: MyCategory

Type

○ Template

● Library

Options

□ Single Instance

Preview

Additional Data

Icon: C:\Progetti C++\Sorgenti UNI_PRO\ide\ex

Version: 1

Revision: 0

Author:

Modification date: 28/05/2010

Description: Library Description...

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.

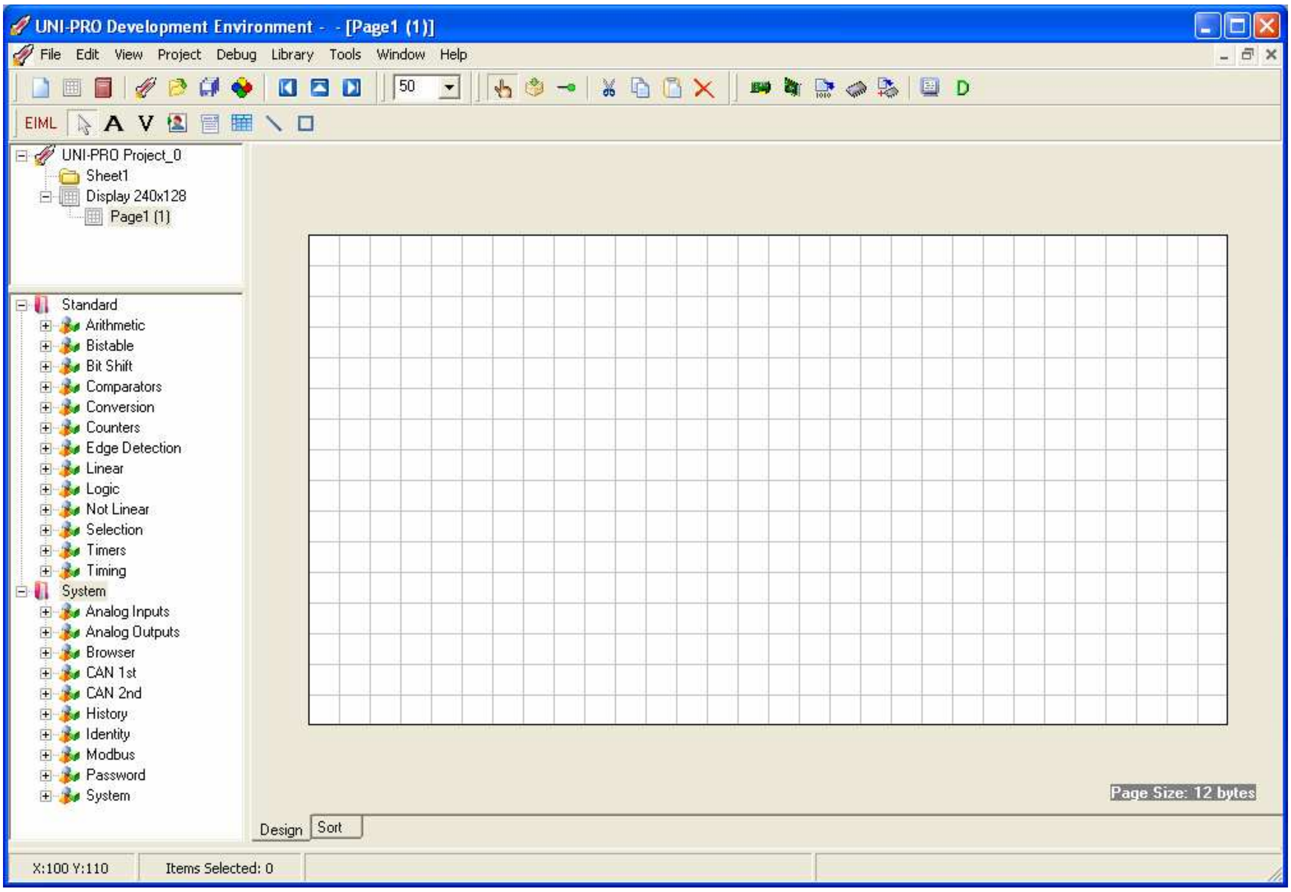
In the *Type* section you can choose the library's type: a *Template* is a group of reusable entities that can be browsed, i.e. once you have added it to a project you can double click its icon to display and change its contents; conversely, a *Library* is a group of entities that cannot be opened or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

2.10 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.

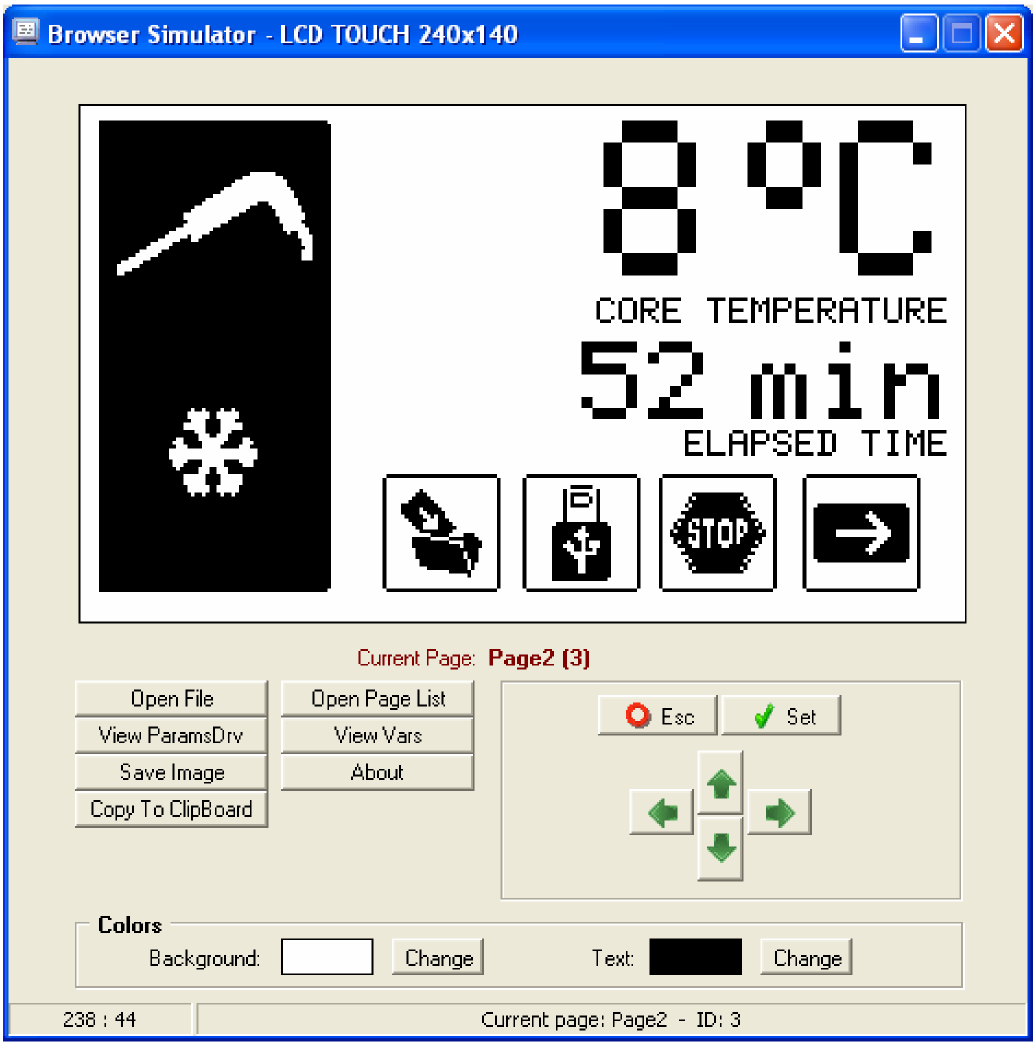


EIML pages may contain texts, variables, icons, combos, tables, lines, and rectangles, all of which will be described in section 3.4.

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.

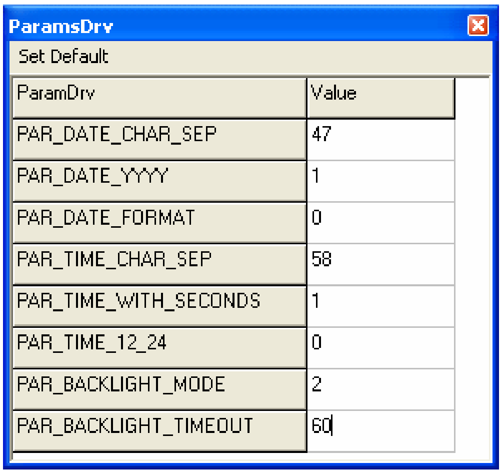
2.11 Simulating EIML pages

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 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.

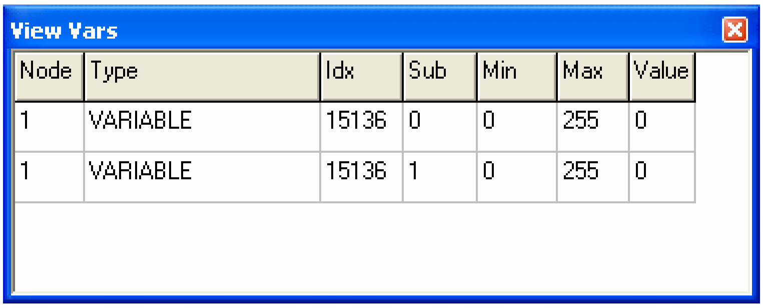
By pressing *View ParamsDrv* you can display and change certain simulation parameters, included in the firmware as well, that affect the way pages are displayed.



ParamDrv	Value
PAR_DATE_CHAR_SEP	47
PAR_DATE_YYYY	1
PAR_DATE_FORMAT	0
PAR_TIME_CHAR_SEP	58
PAR_TIME_WITH_SECONDS	1
PAR_TIME_12_24	0
PAR_BACKLIGHT_MODE	2
PAR_BACKLIGHT_TIMEOUT	60

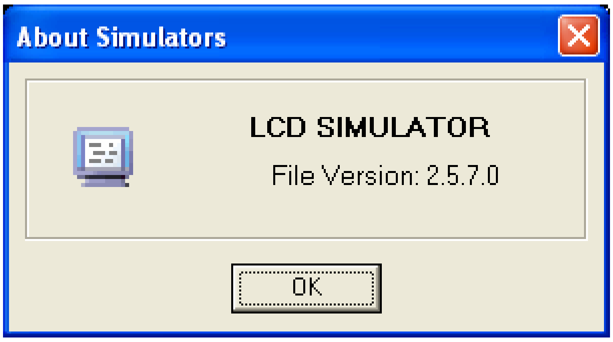
For example, you can decide which characters should be used as time and date separators, whether years should be displayed as two-digit or four-digit values, and set data format.

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.



Node	Type	Idx	Sub	Min	Max	Value
1	VARIABLE	15136	0	0	255	0
1	VARIABLE	15136	1	0	255	0

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.



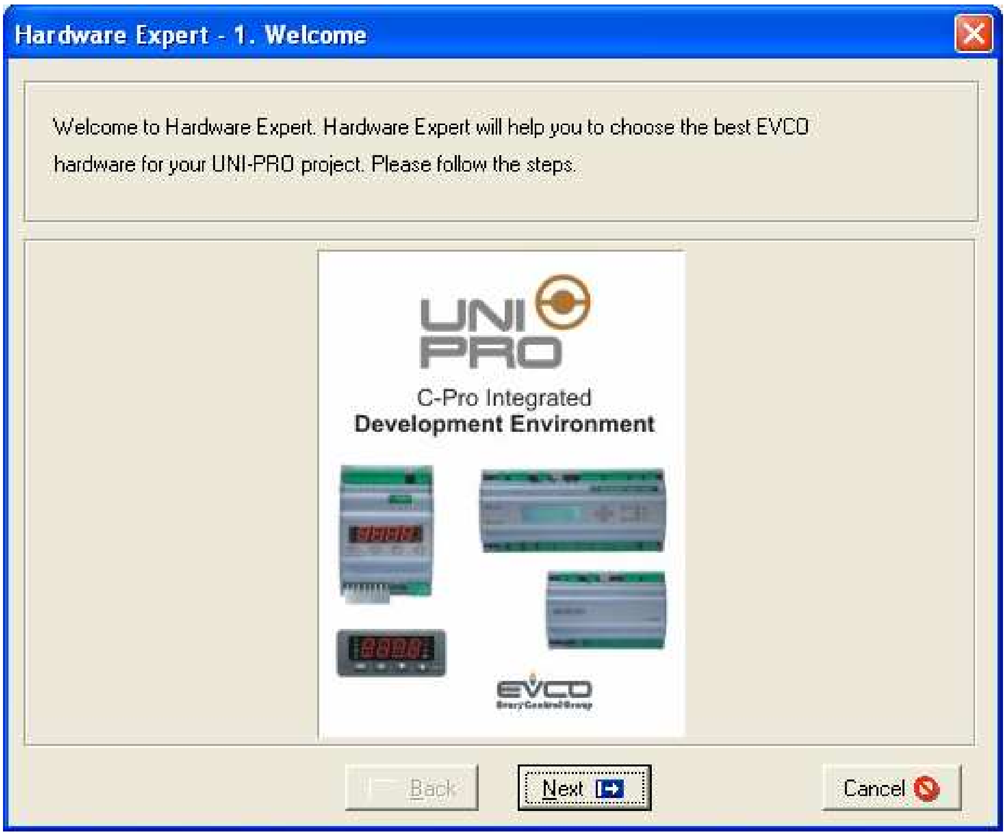
LCD SIMULATOR
File Version: 2.5.7.0

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).

2.12 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.

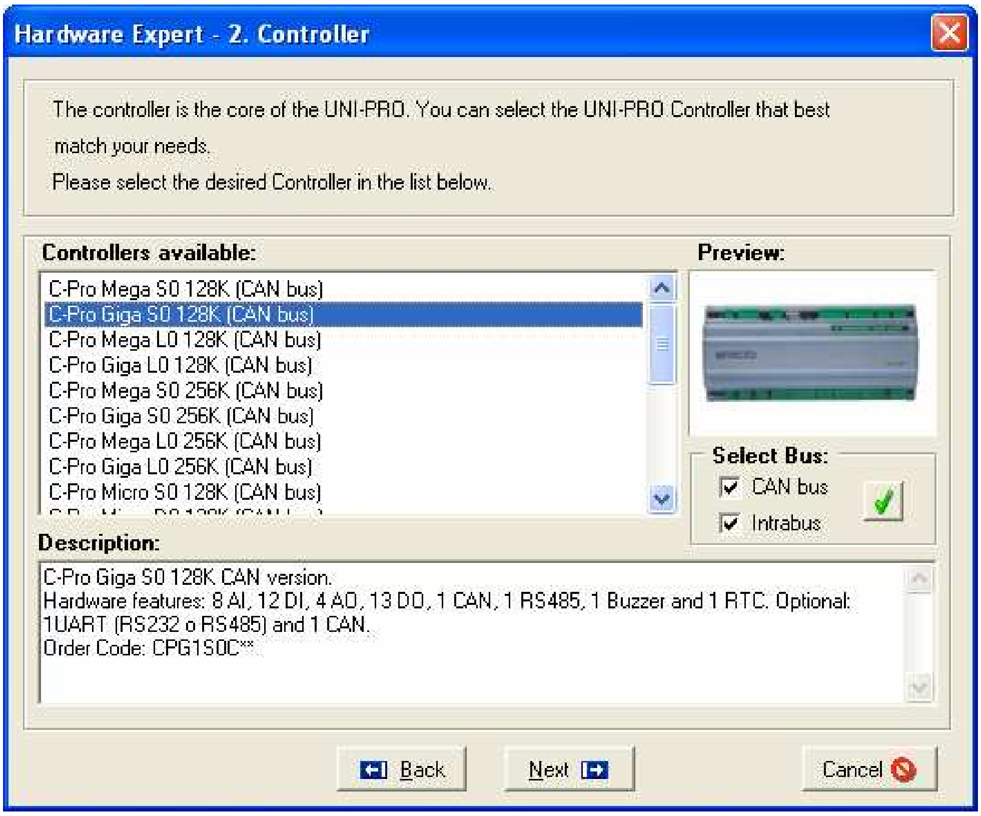
To select the hardware, launch the *Hardware Expert* Wizard by choosing *Project/Hardware Expert* from the menu or clicking the icon. The Wizard will start with a presentation window.



Hardware Expert - 1. Welcome

Welcome to Hardware Expert. Hardware Expert will help you to choose the best EVCO hardware for your UNI-PRO project. Please follow the steps.

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



Hardware Expert - 2. Controller

The controller is the core of the UNI-PRO. You can select the UNI-PRO Controller that best match your needs.
Please select the desired Controller in the list below.

Controllers available:

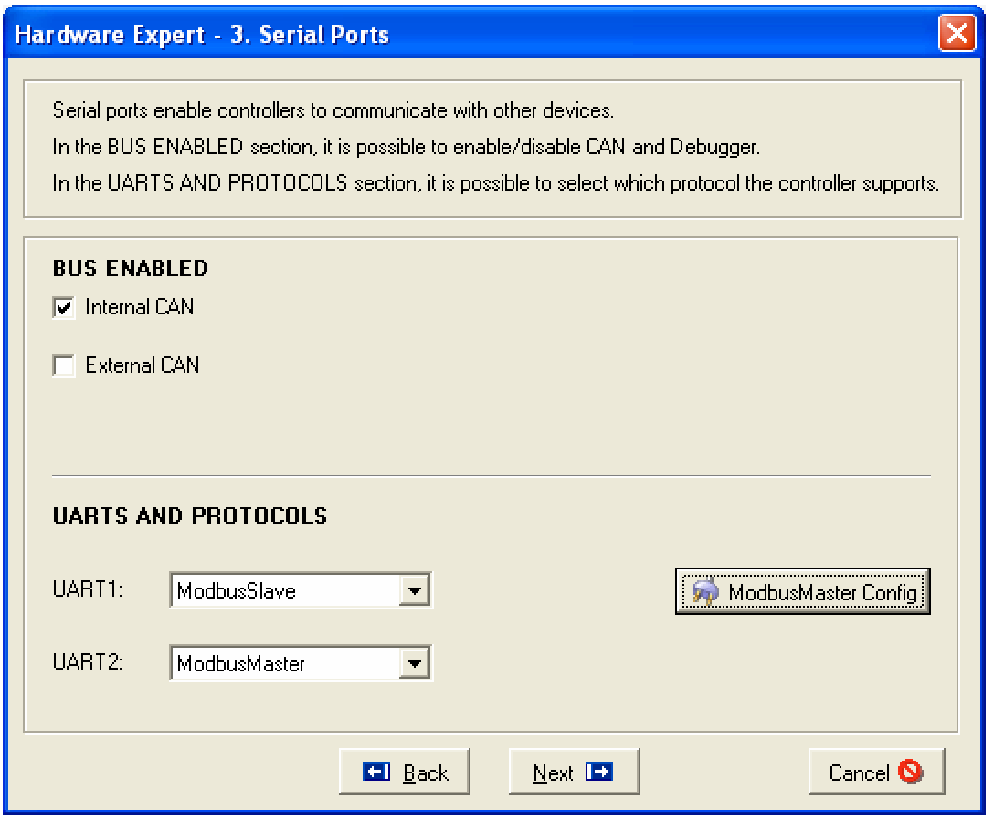
- C-Pro Mega S0 128K (CAN bus)
- C-Pro Giga S0 128K (CAN bus)
- C-Pro Mega L0 128K (CAN bus)
- C-Pro Giga L0 128K (CAN bus)
- C-Pro Mega S0 256K (CAN bus)
- C-Pro Giga S0 256K (CAN bus)
- C-Pro Mega L0 256K (CAN bus)
- C-Pro Giga L0 256K (CAN bus)
- C-Pro Micro S0 128K (CAN bus)

Preview:

Select Bus:
☑ CAN bus
☑ Intrabus

Description:
C-Pro Giga S0 128K CAN version.
Hardware features: 8 AI, 12 DI, 4 AO, 13 DO, 1 CAN, 1 RS485, 1 Buzzer and 1 RTC. Optional: 1UART (RS232 o RS485) and 1 CAN.
Order Code: CPG1S0C**

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).
Click on the appropriate controller and then click Next.



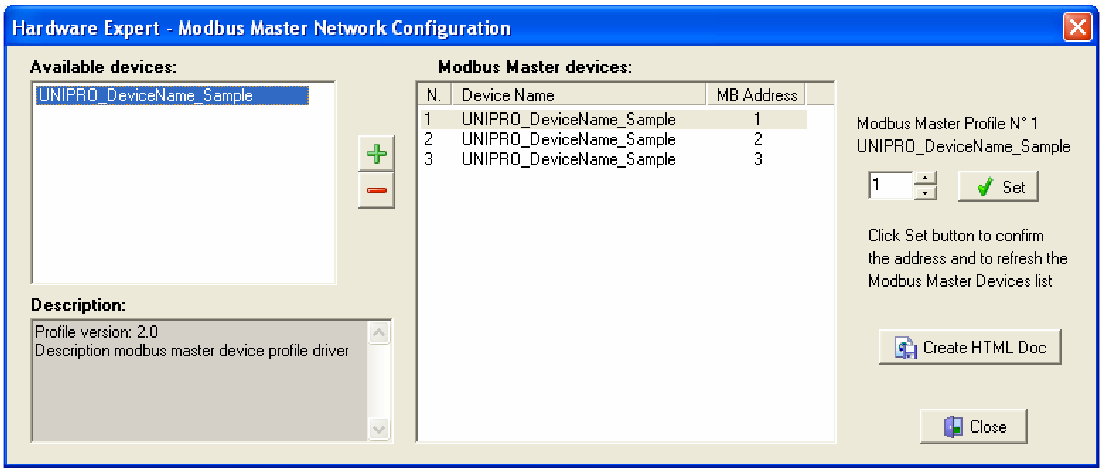
Hardware Expert - 3. Serial Ports

Serial ports enable controllers to communicate with other devices.
In the BUS ENABLED section, it is possible to enable/disable CAN and Debugger.
In the UARTS AND PROTOCOLS section, it is possible to select which protocol the controller supports.

BUS ENABLED
☑ Internal CAN
☐ External CAN

UARTS AND PROTOCOLS
UART1: ModbusSlave
UART2: ModbusMaster

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; the second one allows to select, for each available UART, the protocol to use (e.g. None, ModbusSlave, etc.).
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.



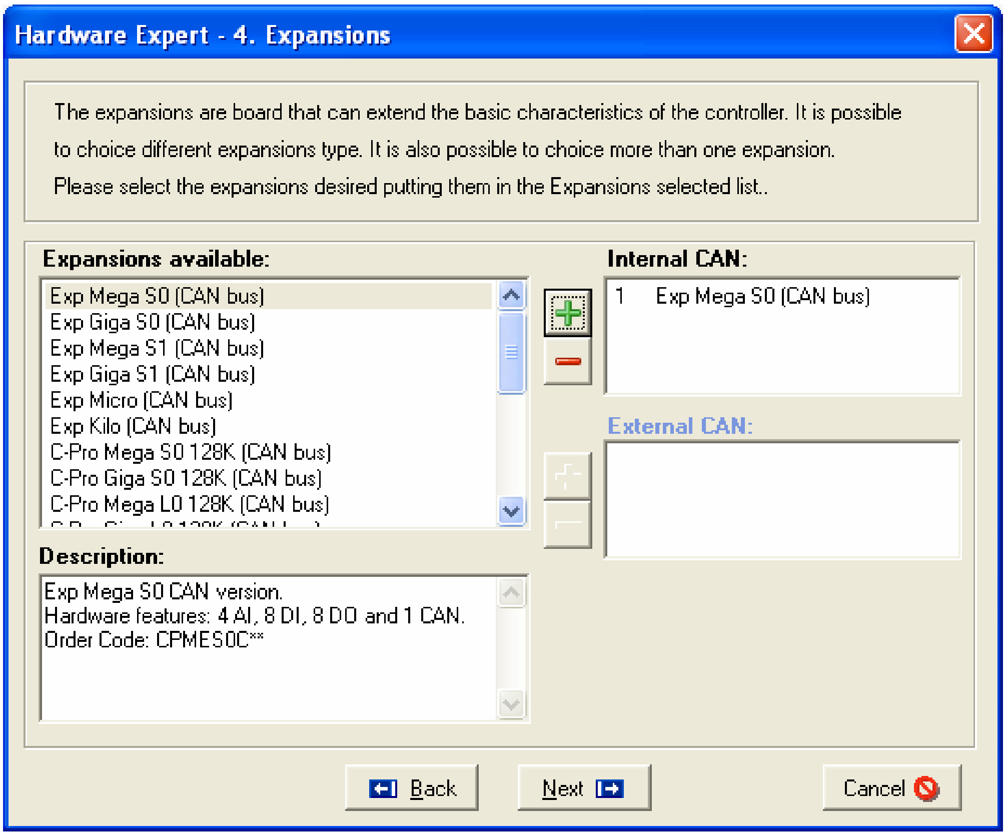
Profile version: 2.0
Description modbus master device profile driver

N.	Device Name	MB Address
1	UNIPRO_DeviceName_Sample	1
2	UNIPRO_DeviceName_Sample	2
3	UNIPRO_DeviceName_Sample	3

Modbus Master Profile N° 1
UNIPRO_DeviceName_Sample
1

Click Set button to confirm the address and to refresh the Modbus Master Devices list

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 + and − it is possible to add or delete the profiles for the ModbusMaster network. When the network are done it is required to set all the modbus addresses for the devices.
Click Close to continue with the Hardware Export in the expansions windows configuration.



The expansions are board that can extend the basic characteristics of the controller. It is possible to choice different expansions type. It is also possible to choice more than one expansion.
Please select the expansions desired putting them in the Expansions selected list..

Exp Mega S0 (CAN bus)
Exp Giga S0 (CAN bus)
Exp Mega S1 (CAN bus)
Exp Giga S1 (CAN bus)
Exp Micro (CAN bus)
Exp Kilo (CAN bus)
C-Pro Mega S0 128K (CAN bus)
C-Pro Giga S0 128K (CAN bus)
C-Pro Mega L0 128K (CAN bus)

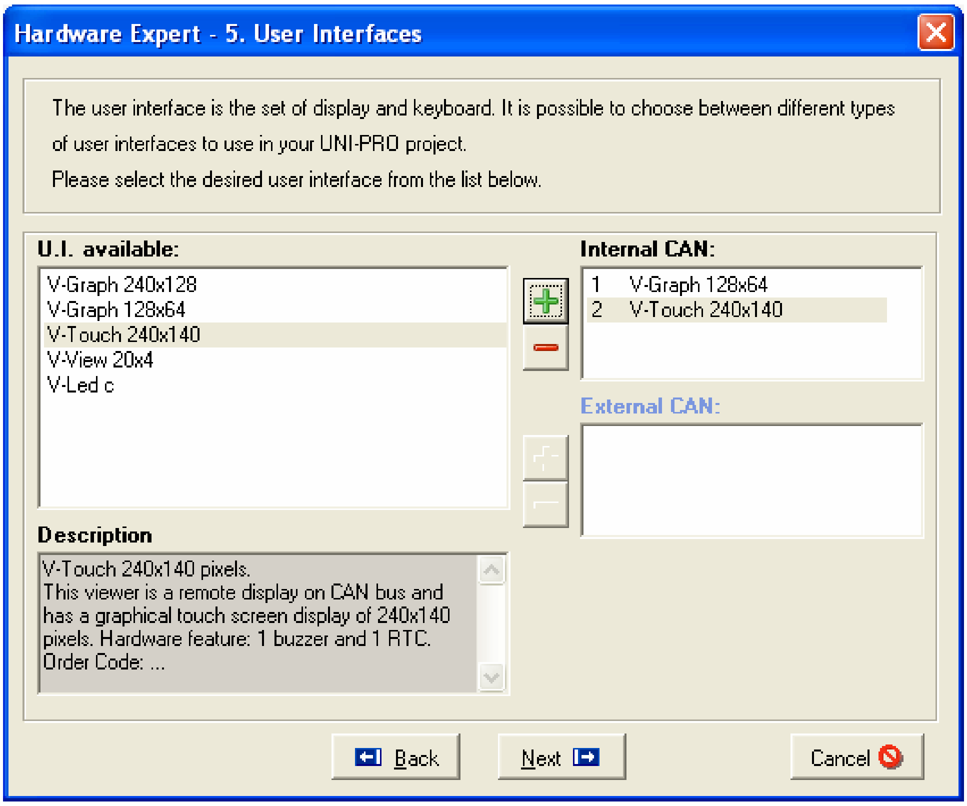
1 Exp Mega S0 (CAN bus)

Exp Mega S0 CAN version.
Hardware features: 4 AI, 8 DI, 8 DO and 1 CAN.
Order Code: CPMES0C**

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.



Hardware Expert - 5. User Interfaces

The user interface is the set of display and keyboard. It is possible to choose between different types of user interfaces to use in your UNI-PRO project.
Please select the desired user interface from the list below.

U.I. available:

- V-Graph 240x128
- V-Graph 128x64
- V-Touch 240x140
- V-View 20x4
- V-Led c

Internal CAN:

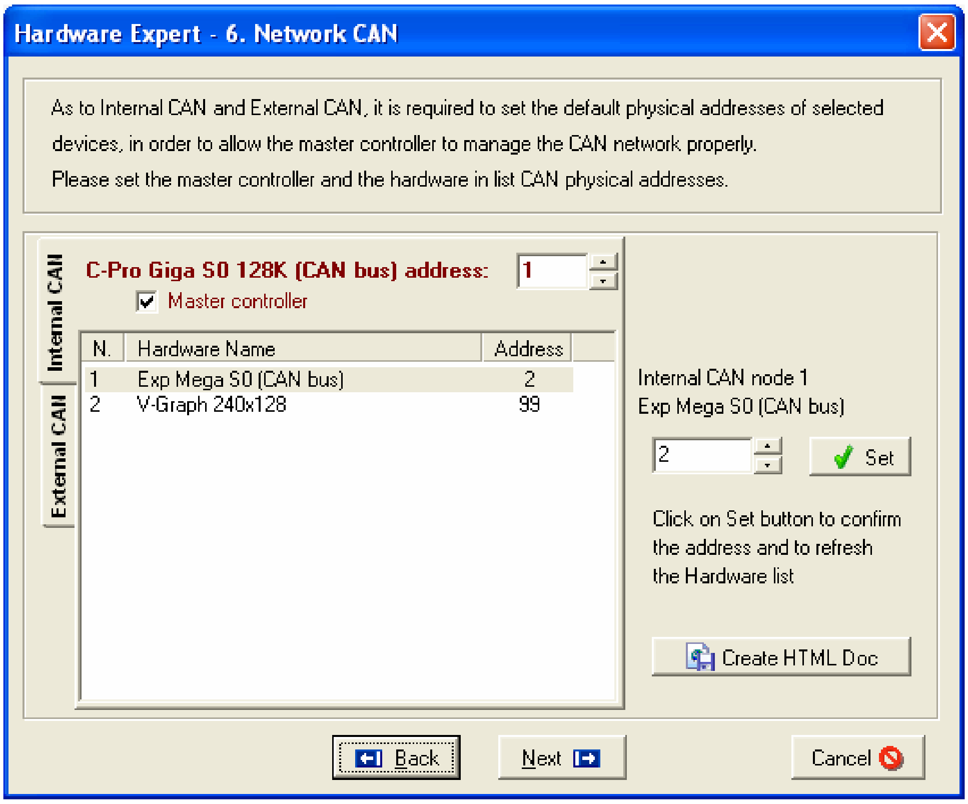
1	V-Graph 128x64
2	V-Touch 240x140

External CAN:

Description

V-Touch 240x140 pixels.
This viewer is a remote display on CAN bus and has a graphical touch screen display of 240x140 pixels. Hardware feature: 1 buzzer and 1 RTC.
Order Code: ...

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:



Hardware Expert - 6. Network CAN

As to Internal CAN and External CAN, it is required to set the default physical addresses of selected devices, in order to allow the master controller to manage the CAN network properly.
Please set the master controller and the hardware in list CAN physical addresses.

Internal CAN | External CAN

C-Pro Giga S0 128K (CAN bus) address: 1

☑ Master controller

N.	Hardware Name	Address
1	Exp Mega S0 (CAN bus)	2
2	V-Graph 240x128	99

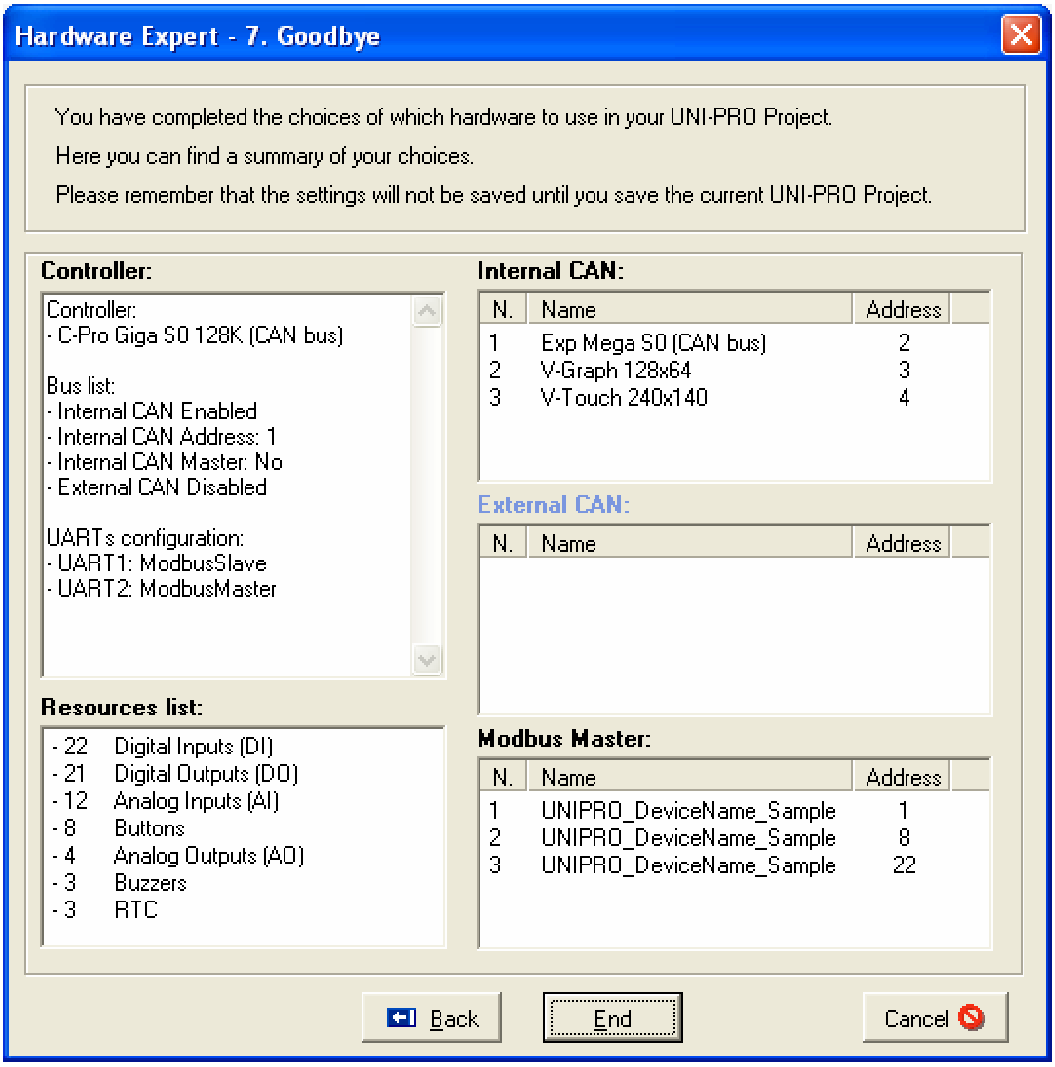
Internal CAN node 1
Exp Mega S0 (CAN bus)

2 ✓ Set

Click on Set button to confirm the address and to refresh the Hardware list

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.



You have completed the choices of which hardware to use in your UNI-PRO Project.
Here you can find a summary of your choices.
Please remember that the settings will not be saved until you save the current UNI-PRO Project.

Controller:

Controller:
- C-Pro Giga S0 128K (CAN bus)

Bus list:
- Internal CAN Enabled
- Internal CAN Address: 1
- Internal CAN Master: No
- External CAN Disabled

UARTs configuration:
- UART1: ModbusSlave
- UART2: ModbusMaster

Resources list:

- 22	Digital Inputs (DI)
- 21	Digital Outputs (DO)
- 12	Analog Inputs (AI)
- 8	Buttons
- 4	Analog Outputs (AO)
- 3	Buzzers
- 3	RTC

Internal CAN:

N.	Name	Address
1	Exp Mega S0 (CAN bus)	2
2	V-Graph 128x64	3
3	V-Touch 240x140	4

External CAN:

N.	Name	Address

Modbus Master:

N.	Name	Address
1	UNIPRO_DeviceName_Sample	1
2	UNIPRO_DeviceName_Sample	8
3	UNIPRO_DeviceName_Sample	22

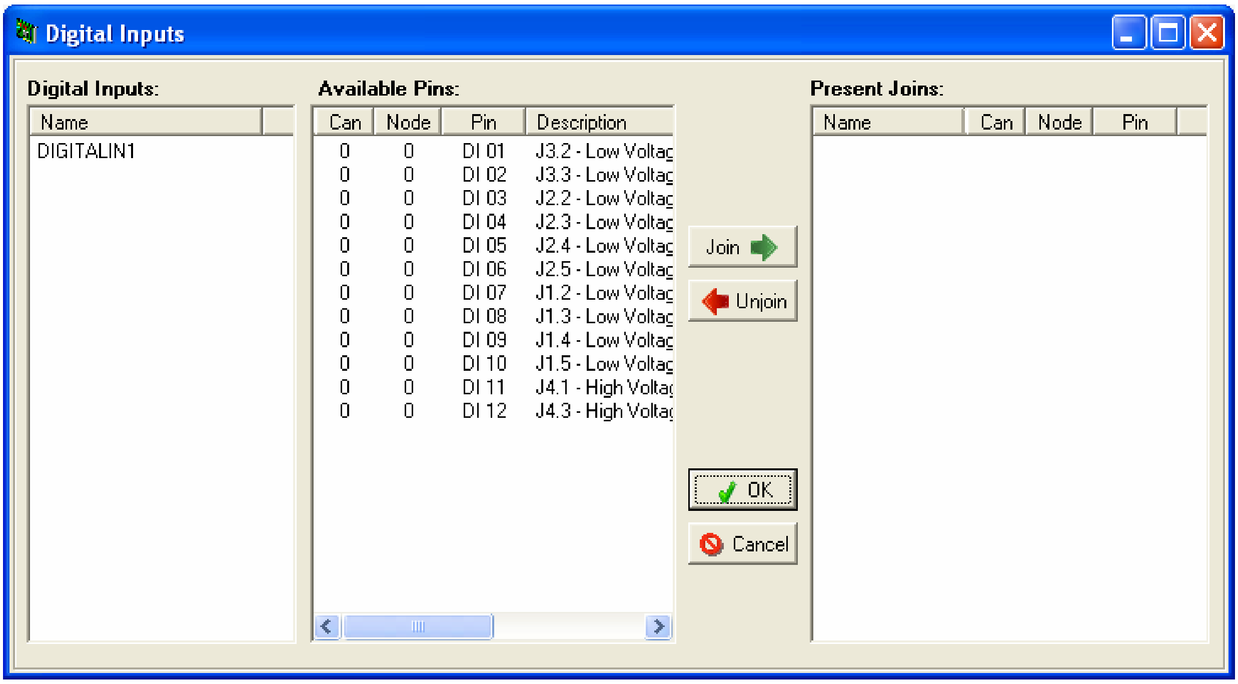
To terminate the Wizard, click *End*. To cancel all changes made, click *Cancel*.

2.13 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.

2.13.1 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.



Digital Inputs:

Name
DIGITALIN1

Available Pins:

Can	Node	Pin	Description
0	0	DI 01	J3.2 - Low Voltag
0	0	DI 02	J3.3 - Low Voltag
0	0	DI 03	J2.2 - Low Voltag
0	0	DI 04	J2.3 - Low Voltag
0	0	DI 05	J2.4 - Low Voltag
0	0	DI 06	J2.5 - Low Voltag
0	0	DI 07	J1.2 - Low Voltag
0	0	DI 08	J1.3 - Low Voltag
0	0	DI 09	J1.4 - Low Voltag
0	0	DI 10	J1.5 - Low Voltag
0	0	DI 11	J4.1 - High Voltag
0	0	DI 12	J4.3 - High Voltag

Present Joins:

Name	Can	Node	Pin

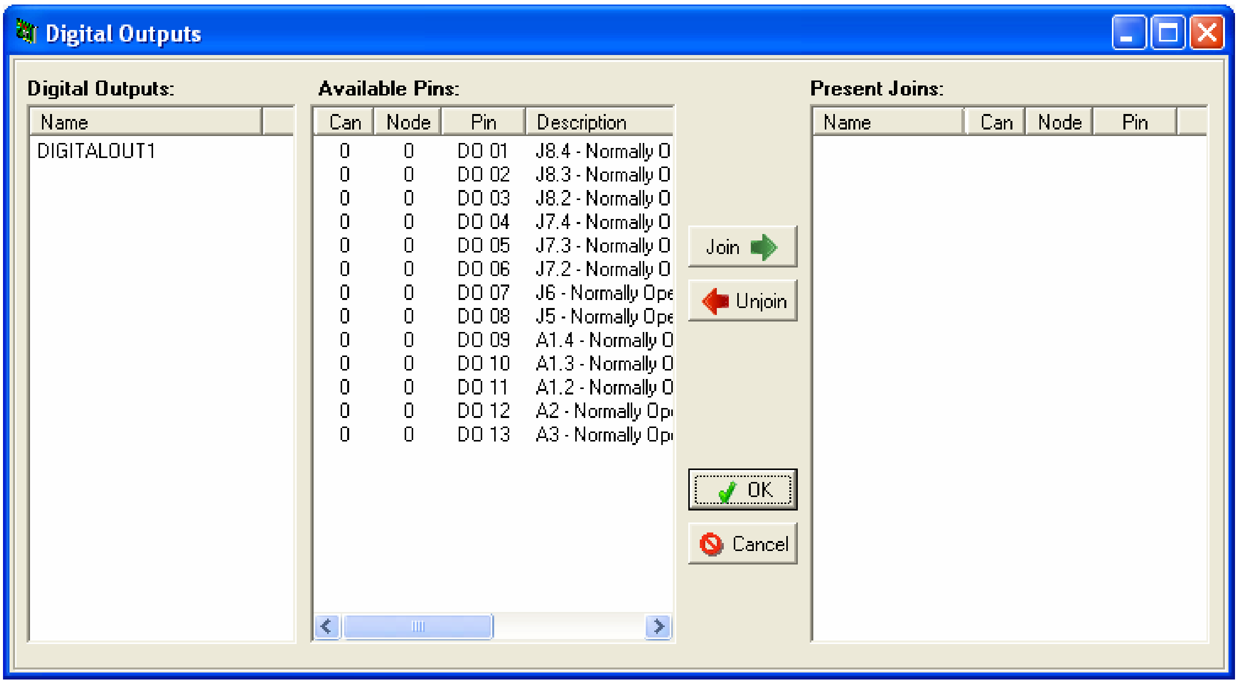
The window is divided into three lists: *Digital Inputs*, *Pins Available* and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the physical digital inputs that are still available, with the following characteristics:

- 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*.

2.13.2 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.



Digital Outputs:

Name
DIGITALOUT1

Available Pins:

Can	Node	Pin	Description
0	0	DO 01	J8.4 - Normally O
0	0	DO 02	J8.3 - Normally O
0	0	DO 03	J8.2 - Normally O
0	0	DO 04	J7.4 - Normally O
0	0	DO 05	J7.3 - Normally O
0	0	DO 06	J7.2 - Normally O
0	0	DO 07	J6 - Normally Ope
0	0	DO 08	J5 - Normally Ope
0	0	DO 09	A1.4 - Normally O
0	0	DO 10	A1.3 - Normally O
0	0	DO 11	A1.2 - Normally O
0	0	DO 12	A2 - Normally Op
0	0	DO 13	A3 - Normally Op

Present Joins:

Name	Can	Node	Pin

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:

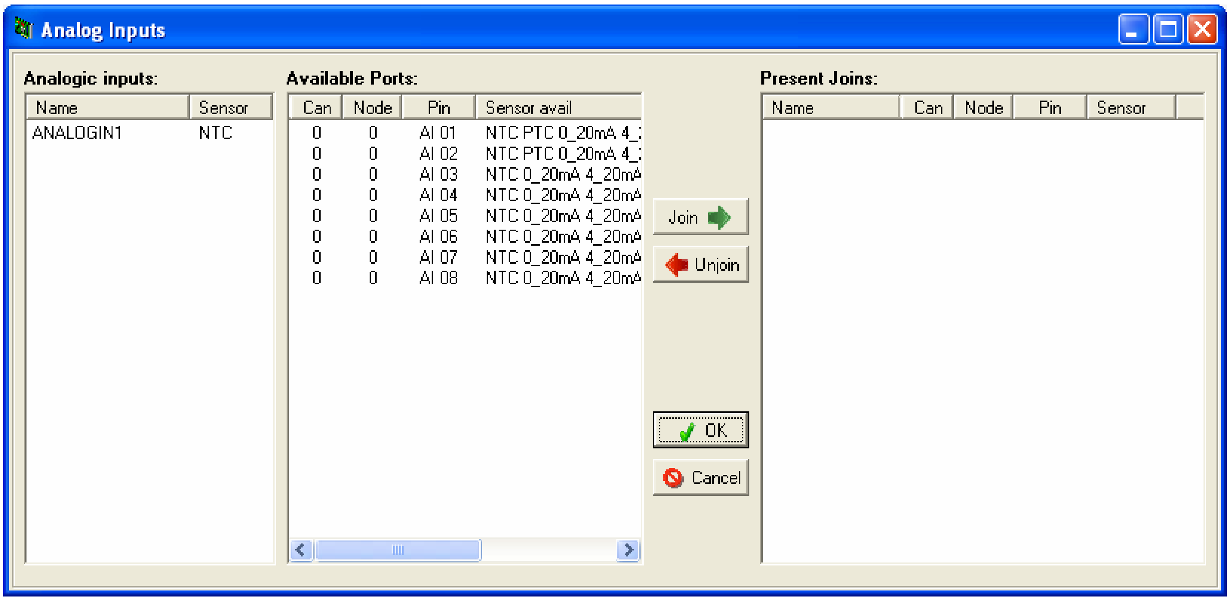
- Node : node of the controller where the resource is found
- Can : bus to which the controller is connected
- Pin : logical reference of the terminal
- Description : shows the description used in the hardware manual documentation of the concerned controller
- Hw name : name of the controller where the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

2.13.3 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.



Analogic inputs:

Name	Sensor
ANALOGIN1	NTC

Available Ports:

Can	Node	Pin	Sensor avail
0	0	AI 01	NTC PTC 0_20mA 4_
0	0	AI 02	NTC PTC 0_20mA 4_
0	0	AI 03	NTC 0_20mA 4_20m
0	0	AI 04	NTC 0_20mA 4_20m
0	0	AI 05	NTC 0_20mA 4_20m
0	0	AI 06	NTC 0_20mA 4_20m
0	0	AI 07	NTC 0_20mA 4_20m
0	0	AI 08	NTC 0_20mA 4_20m

Present Joins:

Name	Can	Node	Pin	Sensor

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:

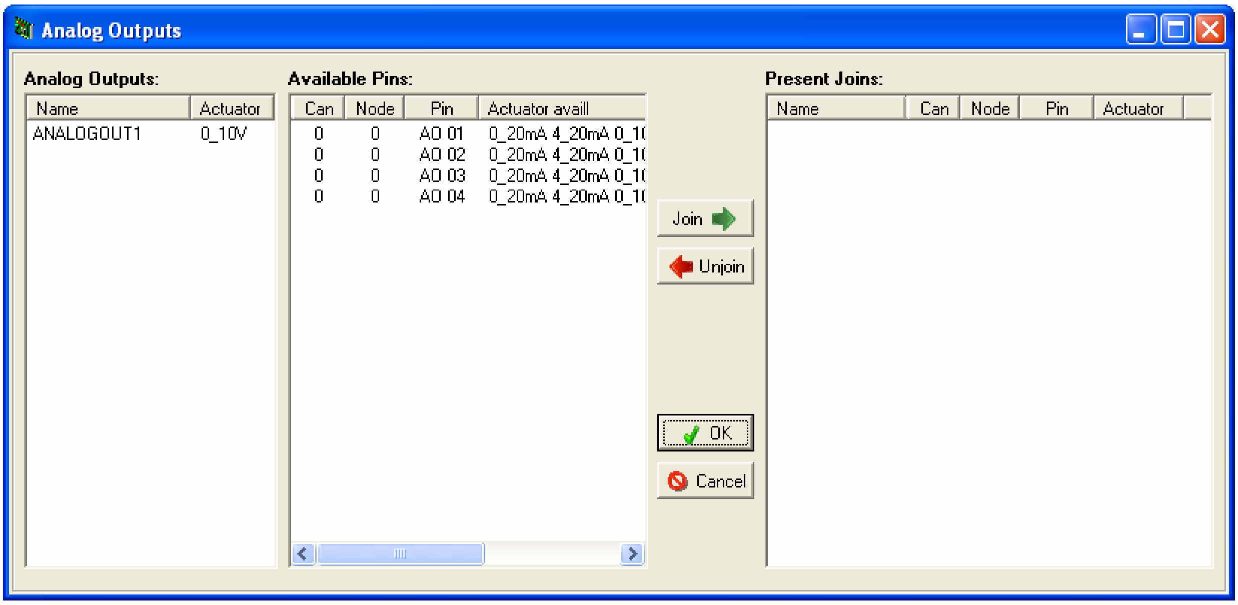
➢ Node	: node of the controller where the resource is found
➢ Can	: bus to which the controller is connected
➢ Pin	: logical reference of the terminal
➢ Sensor avail	: list of sensor types that may be joined to the physical resource
➢ Hw Name	: name of the controller where the resource is found

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*

2.13.4 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.



Analog Outputs:

Name	Actuator
ANALOGOUT1	0_10V

Available Pins:

Can	Node	Pin	Actuator avail
0	0	AO 01	0_20mA 4_20mA 0_1
0	0	AO 02	0_20mA 4_20mA 0_1
0	0	AO 03	0_20mA 4_20mA 0_1
0	0	AO 04	0_20mA 4_20mA 0_1

Present Joins:

Name	Can	Node	Pin	Actuator

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:

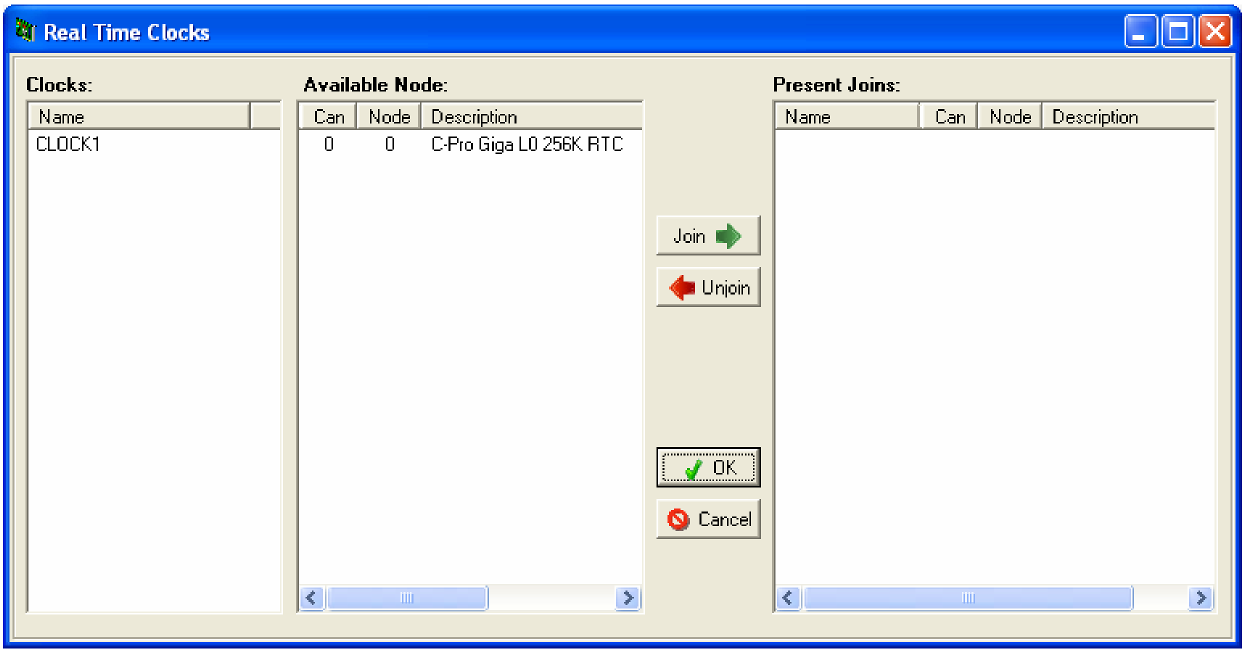
- Node : node of the controller where the resource is found
- Can : bus to which the controller is connected
- Pin : logical reference of the terminal
- Description : shows the description used in the hardware manual documentation of the concerned controller
- Hw name : name of the controller where the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

Finally, the third list contains the 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*

2.13.5 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.



Clocks:

Name
CLOCK1

Available Node:

Can	Node	Description
0	0	C-Pro Giga L0 256K RTC

Present Joins:

Name	Can	Node	Description

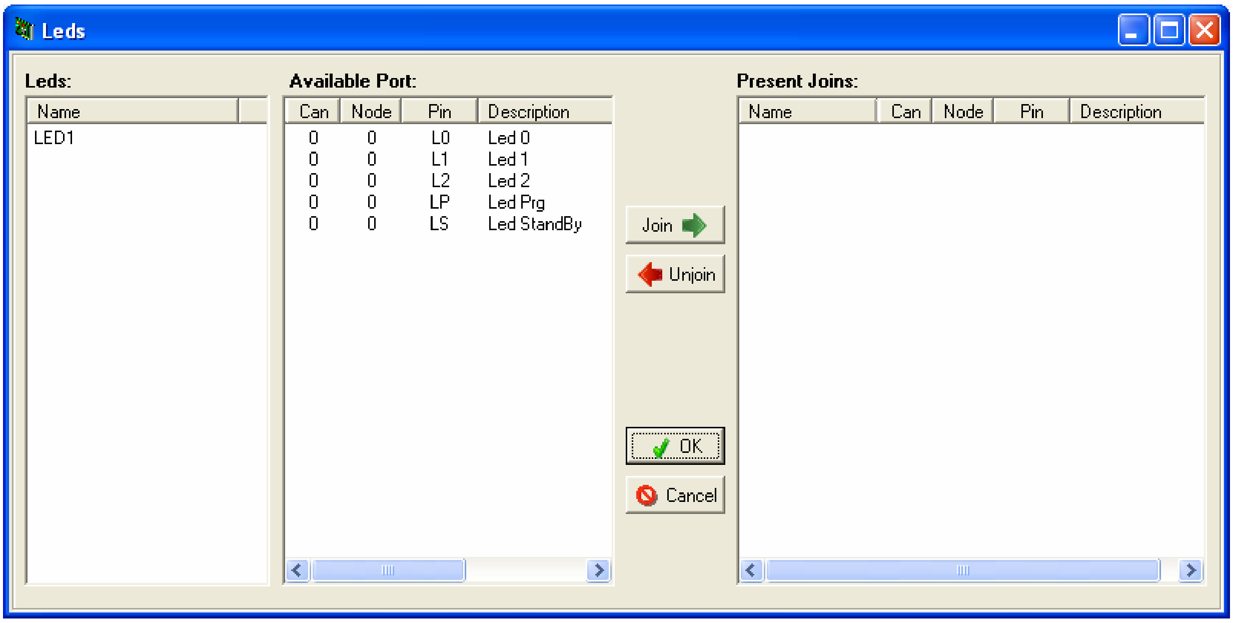
The window is divided into three lists: *Clocks*, *Node Available* and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the physical clocks that are still available, with the following characteristics:

- 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*

2.13.6 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.



Name
LED1

Can	Node	Pin	Description
0	0	L0	Led 0
0	0	L1	Led 1
0	0	L2	Led 2
0	0	LP	Led Prg
0	0	LS	Led StandBy

Name	Can	Node	Pin	Description

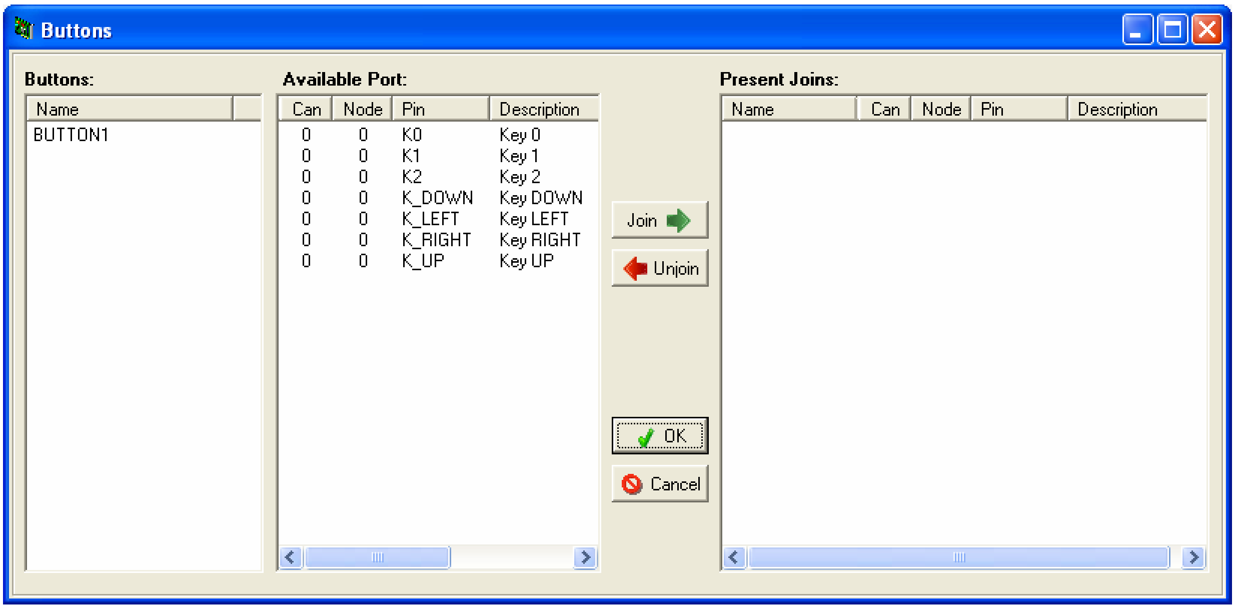
The window is divided into three lists: *LEDs*, *Port Available* and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the physical LEDs that are still available, with the following characteristics:

- ➢ 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*

2.13.7 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.



Buttons:

Name
BUTTON1

Available Port:

Can	Node	Pin	Description
0	0	K0	Key 0
0	0	K1	Key 1
0	0	K2	Key 2
0	0	K_DOWN	Key DOWN
0	0	K_LEFT	Key LEFT
0	0	K_RIGHT	Key RIGHT
0	0	K_UP	Key UP

Present Joins:

Name	Can	Node	Pin	Description

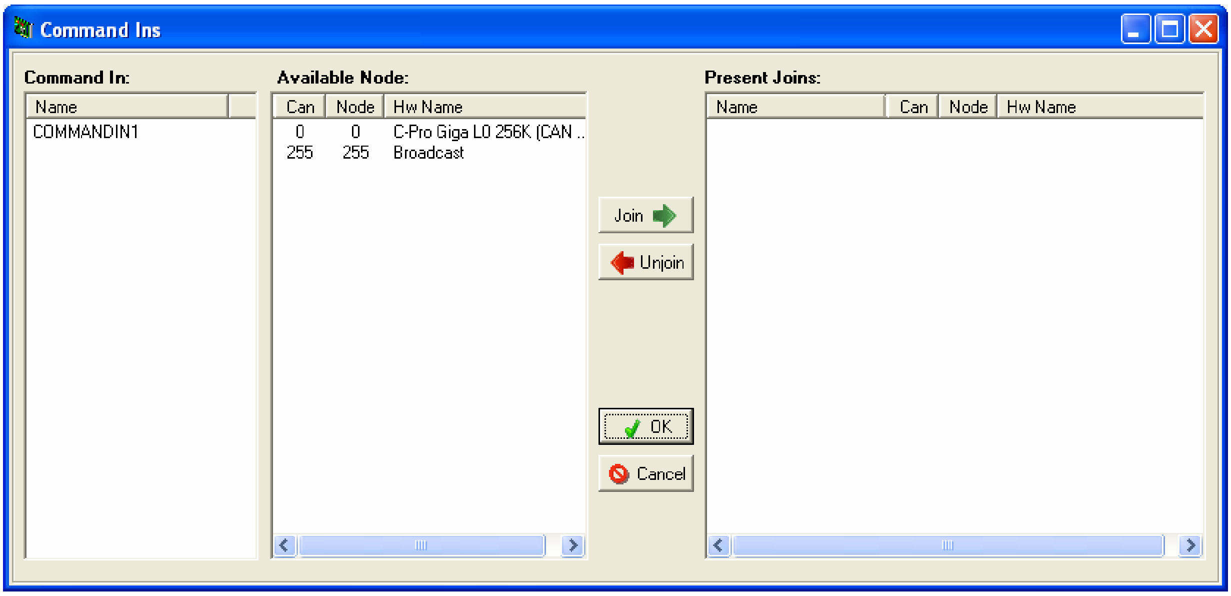
The window is divided into three lists: *Buttons*, *Port Available* and *Joins Present*.
The first list contains the project entities that have not been linked to a hardware resource yet.
The second list shows the buttons that are still available, with the following characteristics:

- 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*

2.13.8 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.



Command In:

Name
COMMANDIN1

Available Node:

Can	Node	Hw Name
0	0	C-Pro Giga L0 256K (CAN ..
255	255	Broadcast

Present Joins:

Name	Can	Node	Hw Name

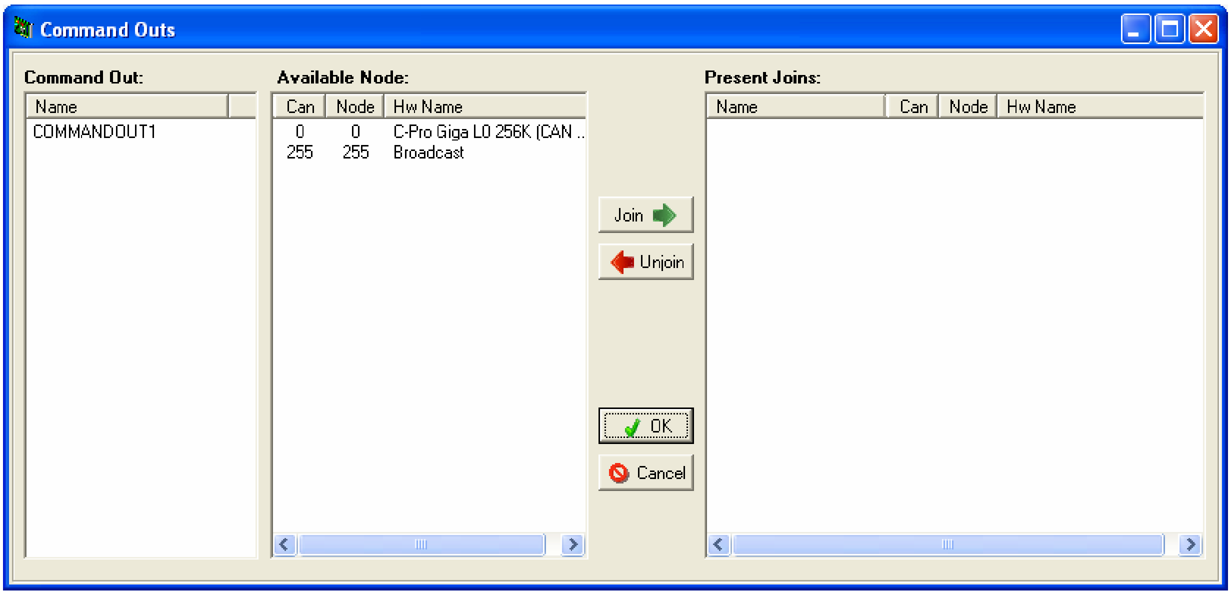
The window is divided into three lists: *Command In*, *Node Available*, and *Joins Present*.
The first list contains the project entities that have not yet been linked to a node from which to receive the command.
The second list contains all controllers (including expansions) that were added to the project 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*

2.13.9 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.



Command Outs

Command Out:

Name
COMMANDOUT1

Available Node:

Can	Node	Hw Name
0	0	C-Pro Giga L0 256K (CAN ..
255	255	Broadcast

Present Joins:

Name	Can	Node	Hw Name

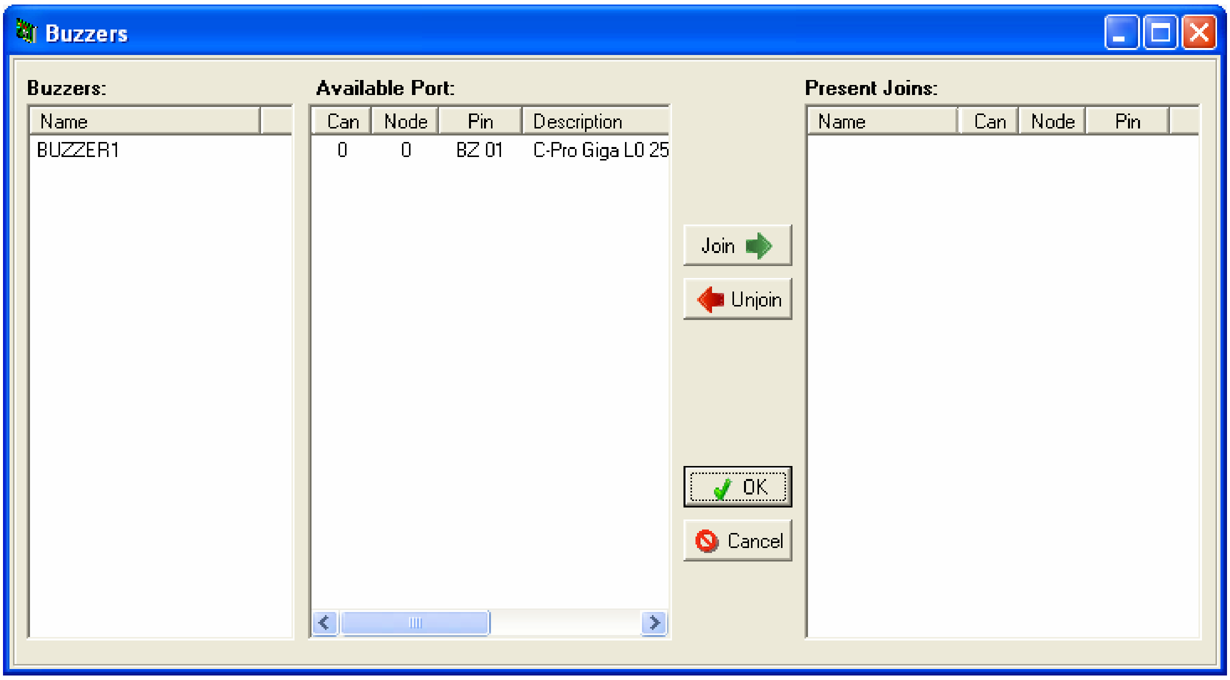
The window is divided into three lists: *Command Out*, *Node Available*, and *Joins Present*.
The first list contains the project entities that have not yet been linked to a node to which to send the command.
The second list contains all controllers (including expansions) that were added to the project 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*

2.13.10 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.



Buzzers:

Name
BUZZER1

Available Port:

Can	Node	Pin	Description
0	0	BZ 01	C-Pro Giga L0 25

Present Joins:

Name	Can	Node	Pin

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

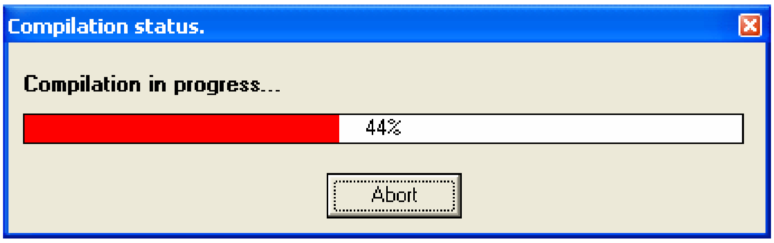
Finally, the third list contains the buzzers and physical resources that have already been joined.
To join an entity and a resource, select a buzzer from the first table and a physical resource from the Port Available table, and then click the *Join* button. Both the entity and the hardware terminal 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*

2.14 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

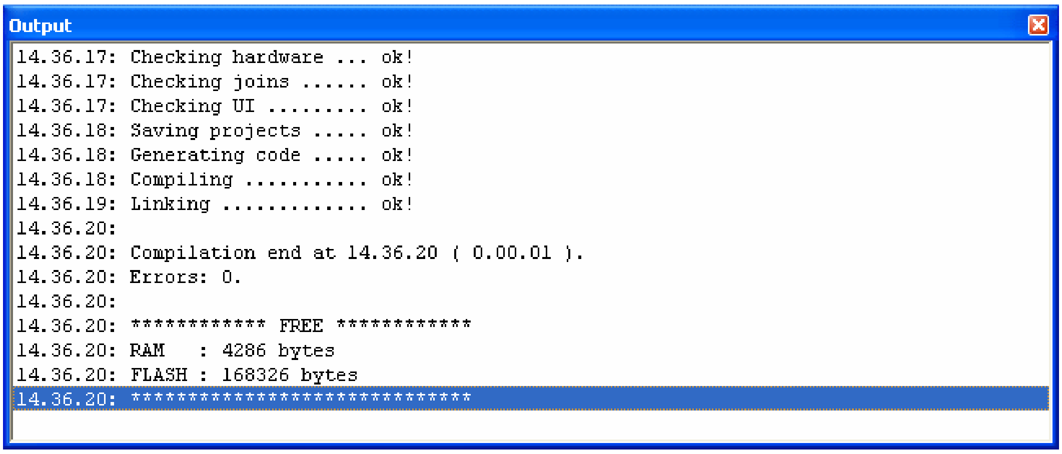
If the project had not been saved yet, you will be prompted to specify a file name to save it to. Then, the project is analyzed and the source files are created, which will be compiled and linked to generate the executable file that will be downloaded to the controller through the Download function. 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.



Compilation status.

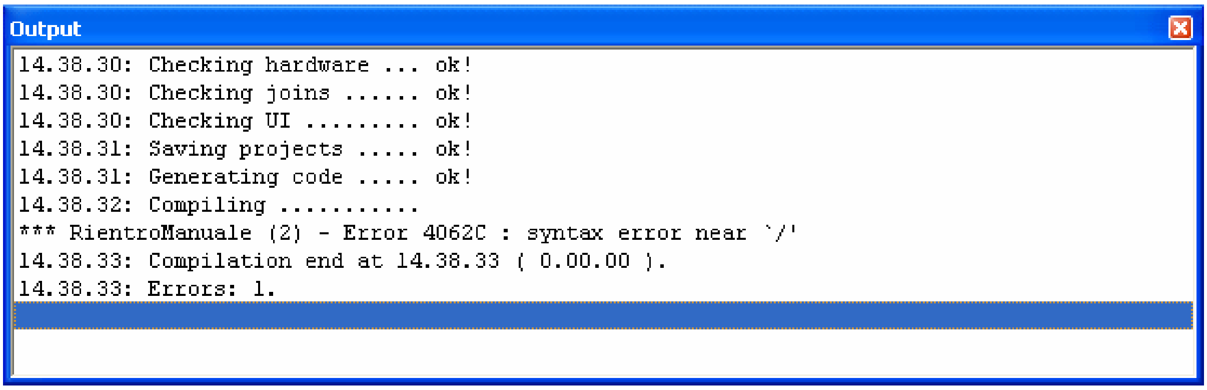
Compilation in progress...

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.



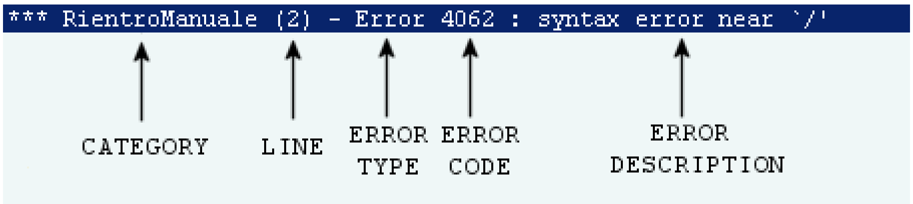
```
14.36.17: Checking hardware ... ok!
14.36.17: Checking joins ...... ok!
14.36.17: Checking UI ......... ok!
14.36.18: Saving projects ..... ok!
14.36.18: Generating code ..... ok!
14.36.18: Compiling ........... ok!
14.36.19: Linking ............. ok!
14.36.20:
14.36.20: Compilation end at 14.36.20 ( 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:



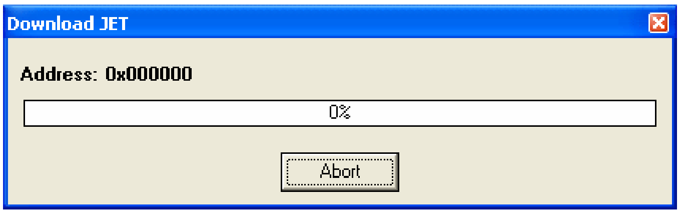
```
*** RientroManuale (2) - Error 4062 : syntax error near `/'
```

- ➢ 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 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 located on the toolbar. The following window will appear:



Address: 0x000000

0%

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.

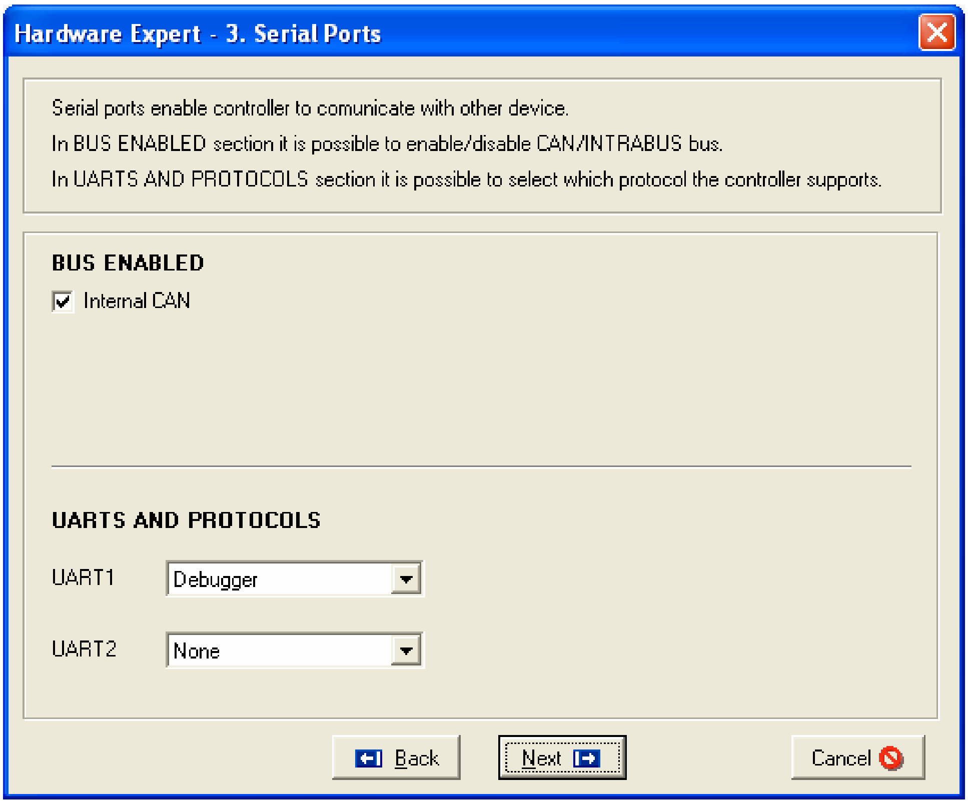
To change download options, see section **Environment configuration**.

2.16 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. This operation is performed in the Hardware Expert during the serial port protocol configuration phase (see the figure below).



Serial ports enable controller to comunicate with other device.

In BUS ENABLED section it is possible to enable/disable CAN/INTRABUS bus.

In UARTS AND PROTOCOLS section it is possible to select which protocol the controller supports.

BUS ENABLED

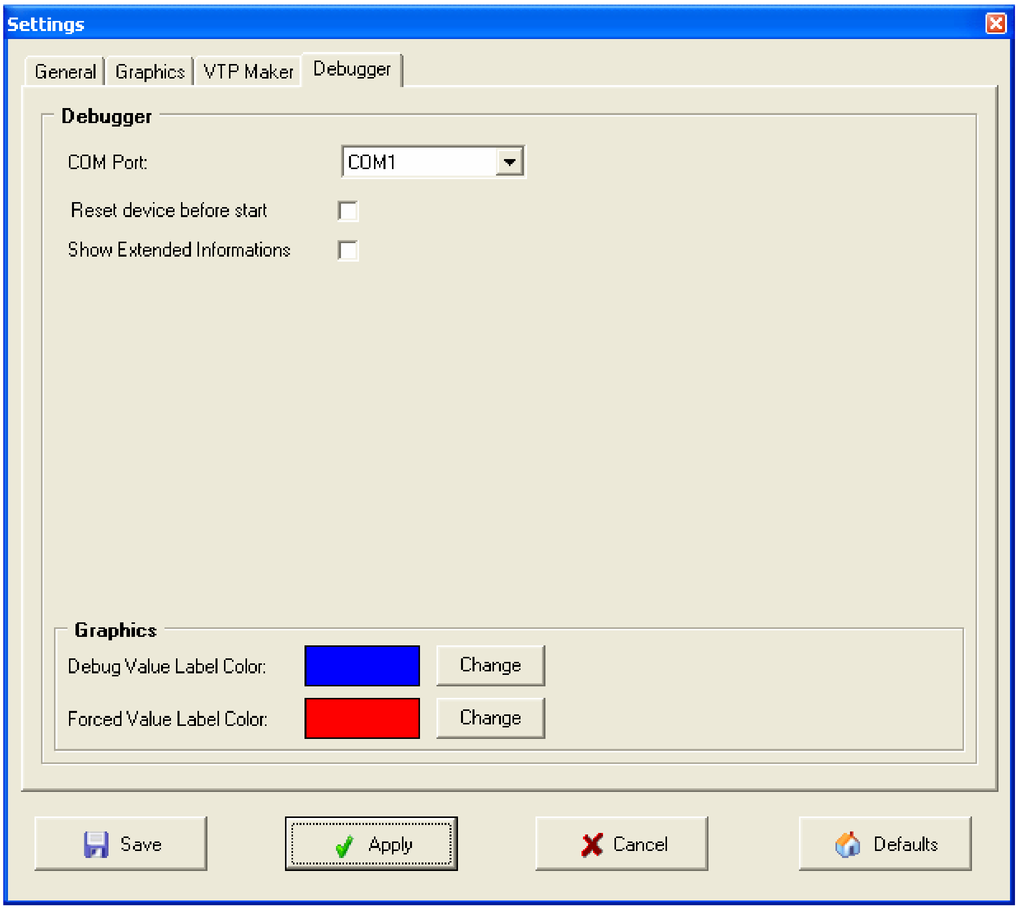
☑ Internal CAN

UARTS AND PROTOCOLS

UART1 Debugger

UART2 None

It is also necessary to configure the serial COM port of the PC to be used to communicate with the controller, using the following page that can be accessed from the Tools/Settings/Debugger menu:



Debugger

COM Port: COM1

Reset device before start □

Show Extended Informations □

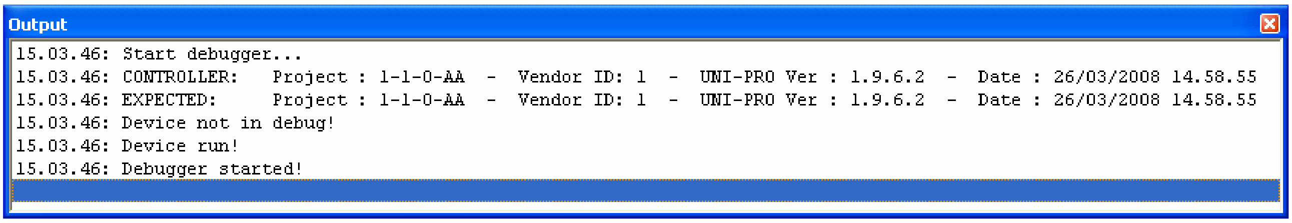
Graphics

Debug Value Label Color: 🟦 Change

Forced Value Label Color: 🟥 Change

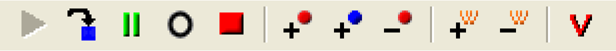
Finally, once the PC port is connected to the port of the controller via an RS232 or RS485 connection, everything is ready for a debug session to begin. To activate it, select *Debug/Start Debugger* from the menu, or the D icon located on the tool bar.

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.

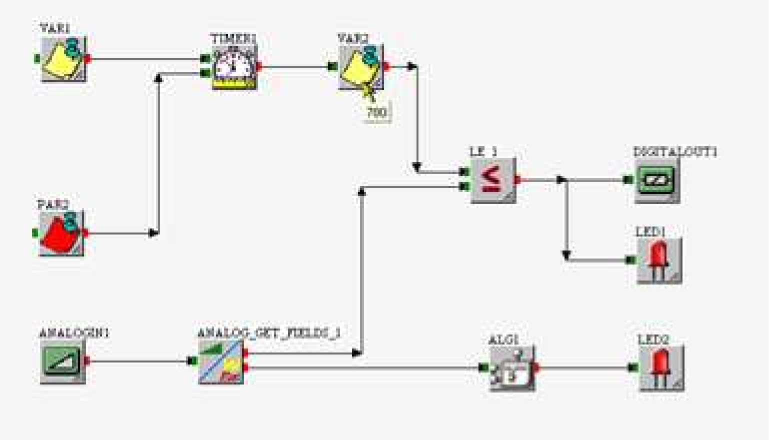


```
15.03.46: Start debugger...
15.03.46: CONTROLLER:    Project : 1-1-0-AA  -  Vendor ID: 1  -  UNI-PRO Ver : 1.9.6.2  -  Date : 26/03/2008 14.58.55
15.03.46: EXPECTED:      Project : 1-1-0-AA  -  Vendor ID: 1  -  UNI-PRO Ver : 1.9.6.2  -  Date : 26/03/2008 14.58.55
15.03.46: Device not in debug!
15.03.46: Device run!
15.03.46: Debugger started!
```

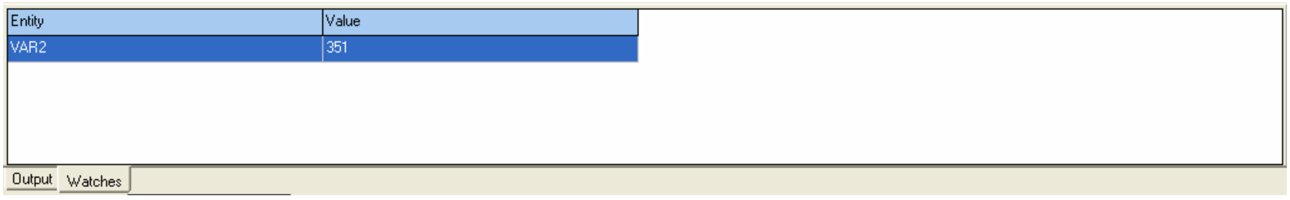
And the following icons will appear on the command bar:



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:



Entity	Value
VAR2	351

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.

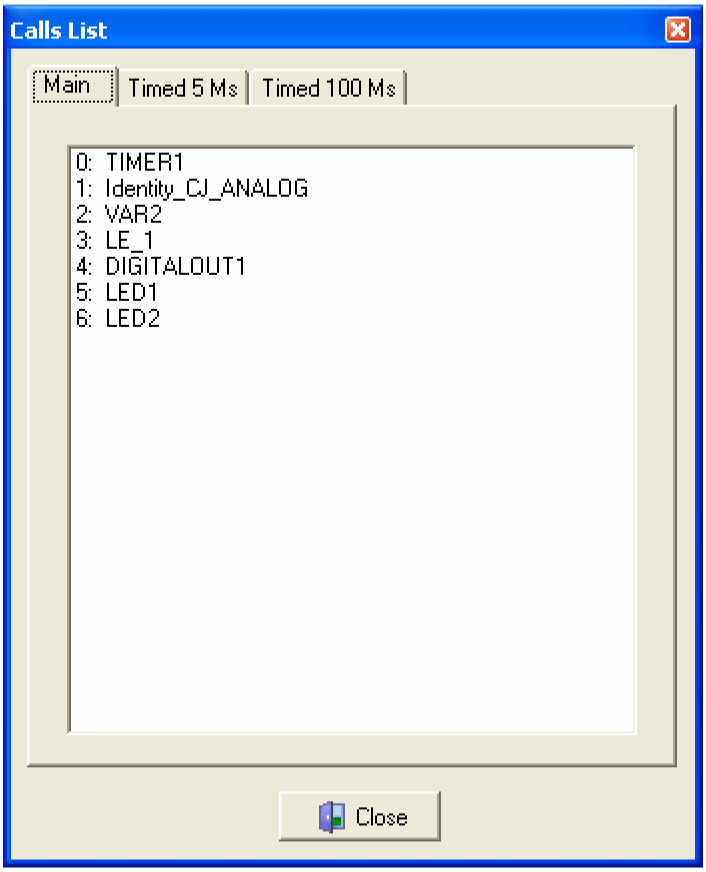
When controlling congruency between the project on PC and that on the controller, the following information is compared:

- Project number
- Project version
- Vendor ID
- UNI-PRO software version
- Compile date
- Number of entities in the project
- 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

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

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

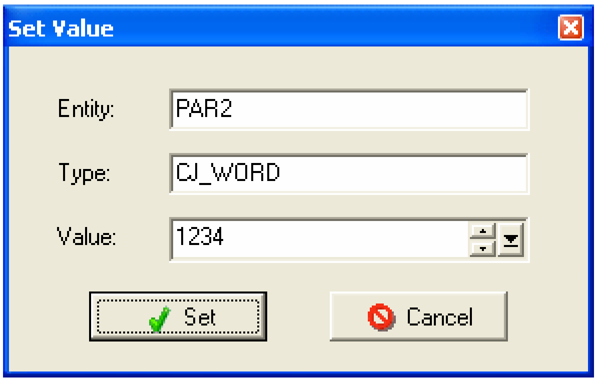
When the program is stopped all of the internal counters and timers are "frozen" but not the inputs or Real Time Clock. It is also possible to observe the value of the entity or to view a few in the watch window.

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 O 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 V 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

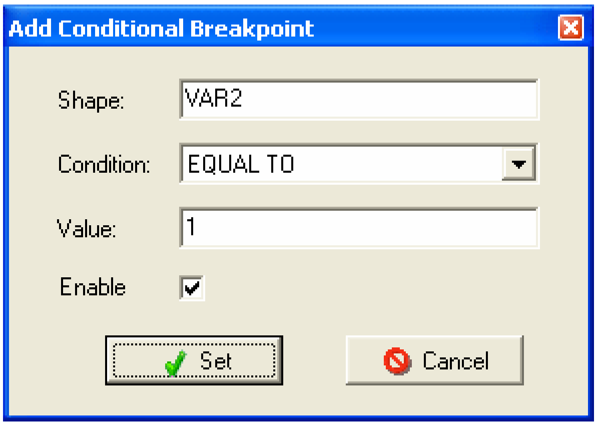
Entity: PAR2

Type: CJ_WORD

Value: 1234

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.


Add Conditional Breakpoint

Shape: VAR2

Condition: EQUAL TO

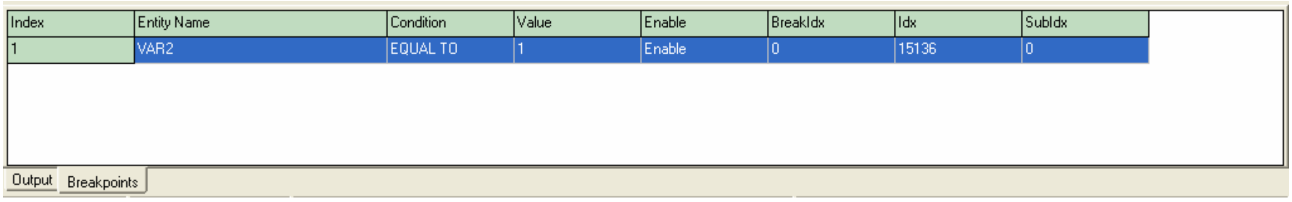
Value: 1

Enable ☑

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

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.

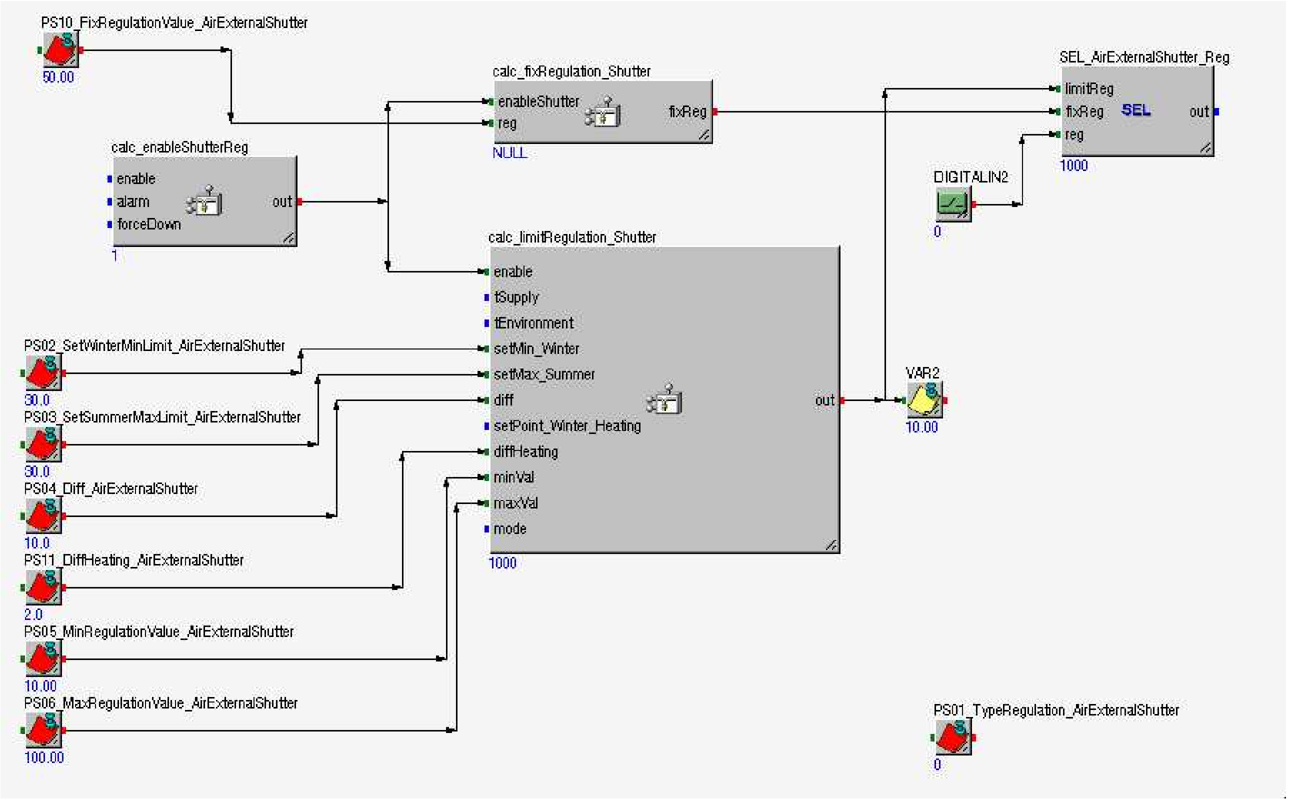
A conditional breakpoint can also be set for the occurrence of a specific event, such as the press of a button or the arrival of a command. In this case, the Value field assumes the following meaning:

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.

2.16.1 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.


PS10_FixRegulationValue_AirExternalShutter
50.00
calc_fixRegulation_Shutter
enableShutter
reg
fixReg
NULL
SEL_AirExternalShutter_Reg
limitReg
fixReg
SEL
out
reg
1000
calc_enableShutterReg
enable
alarm
out
forceDown
1
DIGITALIN2
0
calc_limitRegulation_Shutter
enable
tSupply
tEnvironment
setMin_Winter
setMax_Summer
diff
out
setPoint_Winter_Heating
diffHeating
minVal
maxVal
mode
1000
VAR2
10.00
PS02_SetWinterMinLimit_AirExternalShutter
30.0
PS03_SetSummerMaxLimit_AirExternalShutter
30.0
PS04_Diff_AirExternalShutter
10.0
PS11_DiffHeating_AirExternalShutter
2.0
PS05_MinRegulationValue_AirExternalShutter
10.00
PS06_MaxRegulationValue_AirExternalShutter
100.00
PS01_TypeRegulation_AirExternalShutter
0

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.

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:

CJ_DATE_STRUCT DateToStruct(CJ_DATE Value);

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:

CJ_TIME_STRUCT TimeToStruct(CJ_TIME Value);

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_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.

3.3.18 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.

3.3.19 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.

3.3.20 CJ_WORD CJ_MaxMainTime(void)

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

3.3.21 CJ_WORD CJ_MinMainTime(void)

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

3.3.22 CJ_WORD CJ_RunMainTime(void)

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

3.3.23 CJ_BYTE CJ_MaxInterruptTime(void)

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

3.3.24 CJ_BYTE CJ_MinInterruptTime(void)

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

3.3.25 CJ_BYTE CJ_RunInterruptTime(void)

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

3.3.26 CJ_BYTE CJ_ModbusAskQueue(void)

Returns the free items number of the Modbus queue

3.3.27 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.

3.3.28 CJ_BIT CJ_IsFirstMain(void)

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

3.3.29 CJ_BIT CJ_Stack_Error_Read(void)

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

3.3.30 CJ_BIT CJ_Math_Error_Read(void)

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

3.3.31 CJ_BIT CJ_DivByZero_Error_Read(void)

Is one if there was a division by zero.

3.3.32 CJ_BIT CJ_Overflow_Error_Read(void)

Is one if there was an overflow.

3.3.33 CJ_BIT CJ_Underflow_Error_Read(void)

Is one if there was an underflow.

3.3.34 CJ_BIT CJ_NaN_Error_Read(void)

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

3.3.35 void CJ_DivByZero_Error_Write(void)

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

3.3.36 void CJ_Overflow_Error_Write(void)

Allows to set the mathematical error caused by an overflow.

3.3.37 void CJ_Underflow_Error_Write(void)

Allows to set the mathematical error caused by an underflow.

3.3.38 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.

3.4 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:

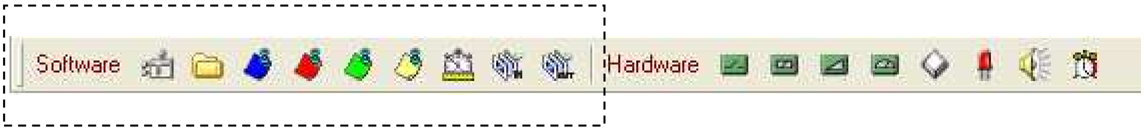
- Software
- Hardware

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.



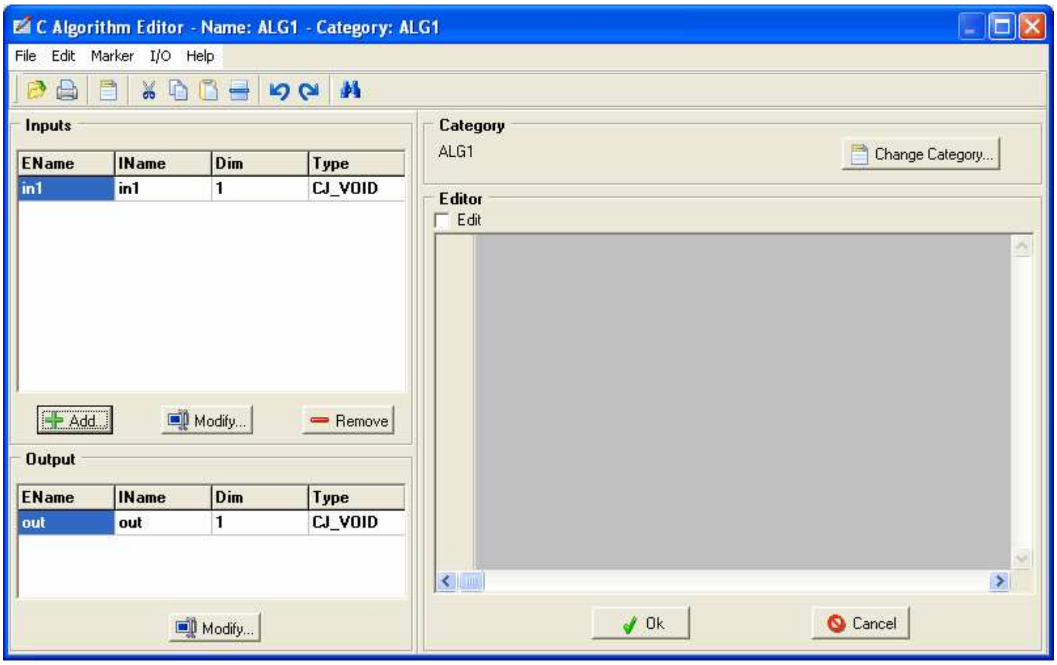
Algorithm

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


C Algorithm Editor - Name: ALG1 - Category: ALG1

Inputs

EName	IName	Dim	Type
in1	in1	1	CJ_VOID

Output

EName	IName	Dim	Type
out	out	1	CJ_VOID

Category: ALG1

☐ Edit

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

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

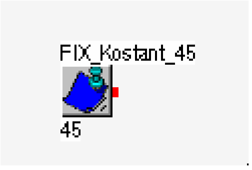
In most projects, it is necessary to define constants; you can do this using the Fixs. They represent a flash memory area where you can define a constant value (thus without affecting the amount of RAM memory taken up by the project) that can be used in your project.
For each project the maximum use number of FIX is 1000.
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.

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.
Rics	Field to insert useful information to VTP driver generation required for the communication with EVCO supervision software *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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.



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 2000.

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.
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 *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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 1000.
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.
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 *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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

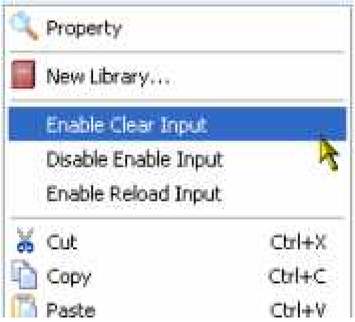
Vars represent RAM memory allocations that can be used to process data; since they are stored in the volatile memory, they will be set to their default values every time the machine is reset.
For each project the maximum use number of VAR is 2000.
They have the following properties:

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.
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 *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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 250.

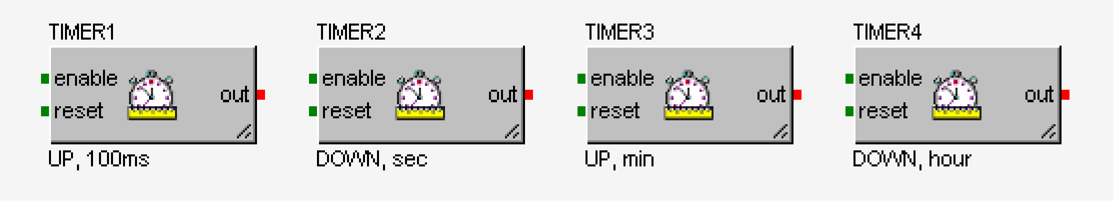
They have the following properties:

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 *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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 250.
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

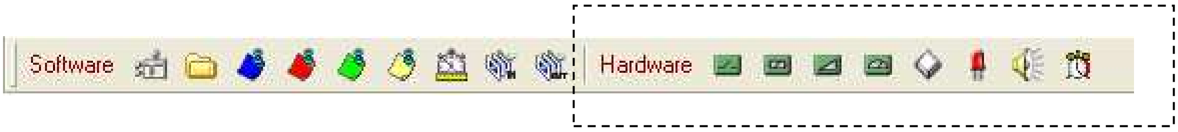
Command Outs are used to send commands that will be interpreted by a Command In located in the same or a different controller.
This object type has two inputs: **param** and **trigger**.
The first input is used to retrieve the parameter when sending the command; the second one is a trigger input, which 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 250.
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.

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 250.

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 *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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 *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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, and 4-20mA. 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 *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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 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.
Category	Set to AnalogOut 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 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 *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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 *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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.
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.

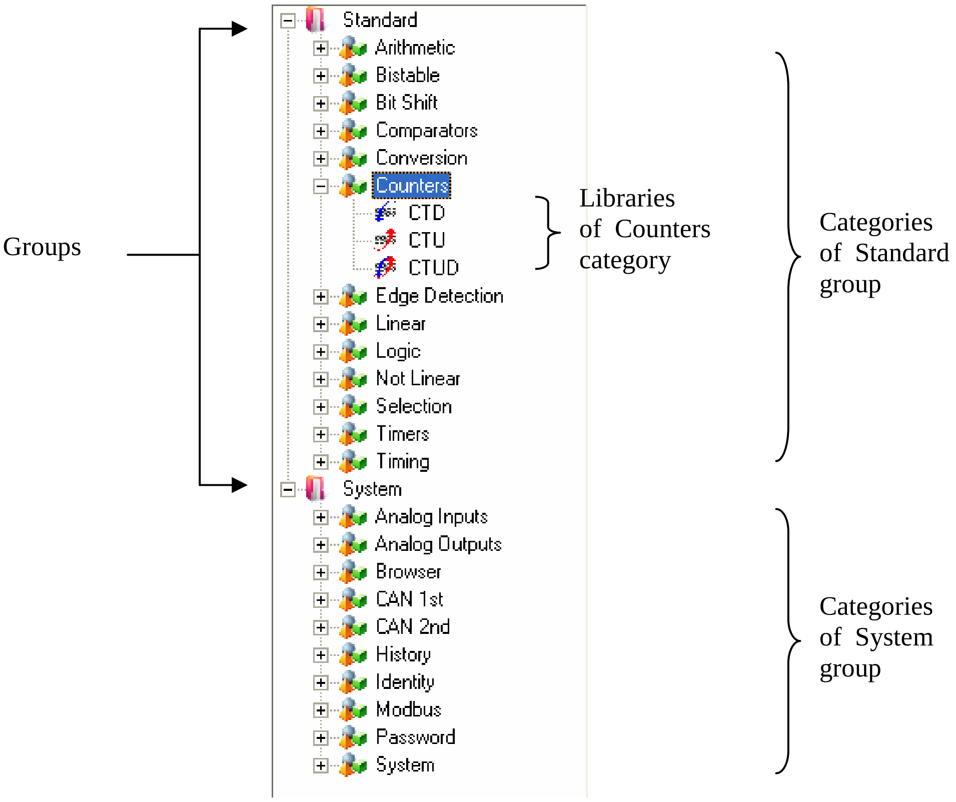
Clock objects return a CJ_DATETIME value that you can easily convert into the CJ_DATETIME_STRUCT structure, to perform more complex operations (please refer to section Data types - CJ_DATETIME_STRUCT).

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 *R.I.C.S.* and EVCO parameters management software *Params Manager*.
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
  - Arithmetic
  - Bistable
  - Bit Shift
  - Comparators
  - Conversion
  - Counters
    - CTD
    - CTU
    - CTUD
  - Edge Detection
  - Linear
  - Logic
  - Not Linear
  - Selection
  - Timers
  - Timing
- System
  - Analog Inputs
  - Analog Outputs
  - Browser
  - CAN 1st
  - CAN 2nd
  - History
  - Identity
  - Modbus
  - Password
  - System

Groups
Libraries of Counters category
Categories of Standard group
Categories of System group

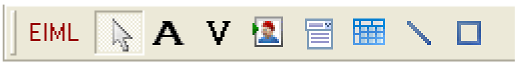
Standard and *System* groups are the two default groups of the UNI-PRO libraries, you can add other groups and extends it with new categories creating new libraries.

3.5 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 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.

3.5.1 Toolbar

The EIML toolbar lists the elements that you can add to the page: Texts, Variables, Icons, 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

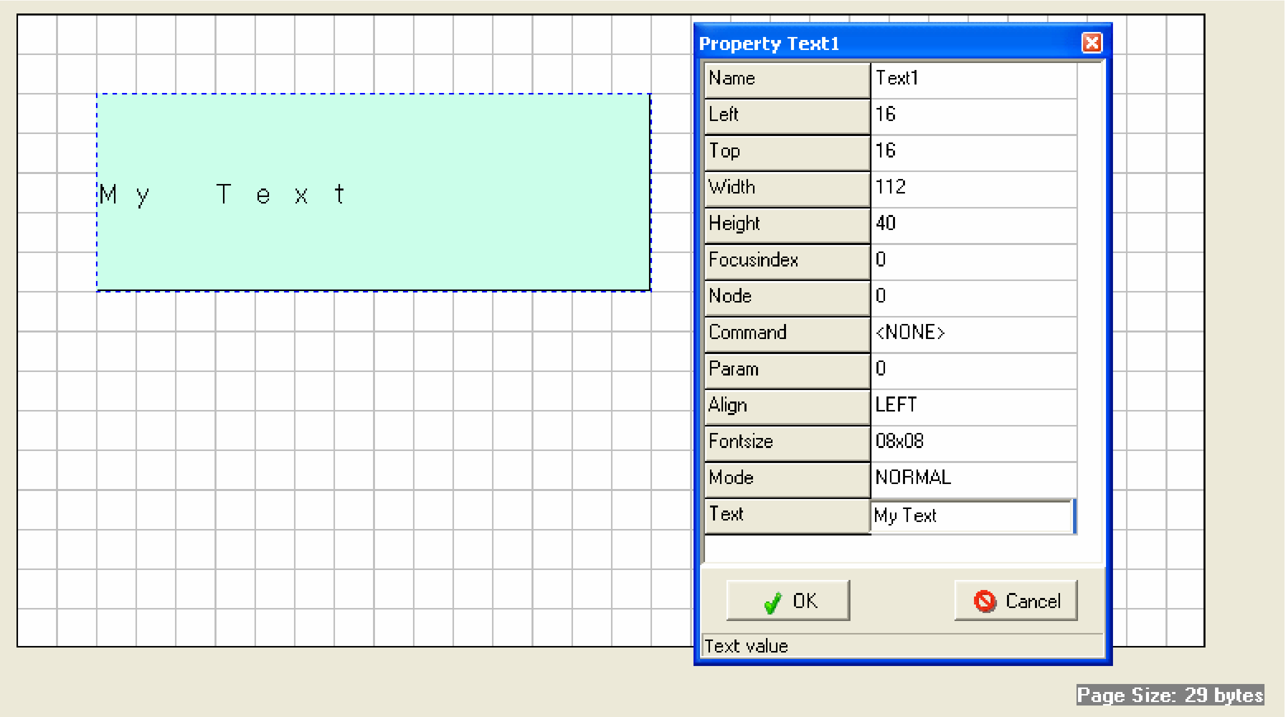
EIML pages serve as containers for other EIML objects and have the following properties:

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.
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).
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.
CondVis	If selected allows to use the Conditional Visibility on the EIML pages.

3.5.4 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.



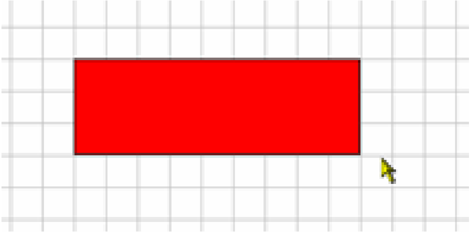
Name	Text1
Left	16
Top	16
Width	112
Height	40
Focusindex	0
Node	0
Command	<NONE>
Param	0
Align	LEFT
Fontsize	08x08
Mode	NORMAL
Text	My Text

Text

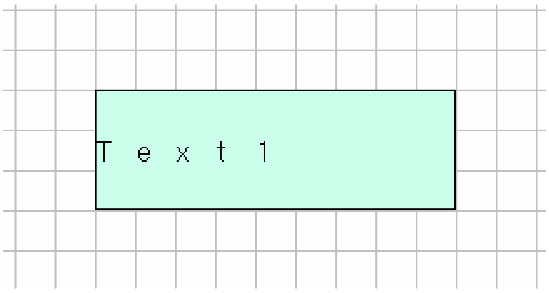
The text element allows you to add descriptions, activate commands, and enable page browsing.
As described under section ***Basic Operations***, the text element has certain properties that allow you to graphically position it in the page ((Left, Top, Width and Height), while the remaining properties 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.
Align	Permissible values are LEFT, CENTER, and RIGHT which indicate left, center or right alignment respectively.
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.
Mode	A text may be displayed in four different modes: ➢ NORMAL ➢ NEGATIVE ➢ NORMAL-BLINKING ➢ NEGATIVE-BLINKING 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.
Text	Represents the text you wish to display.

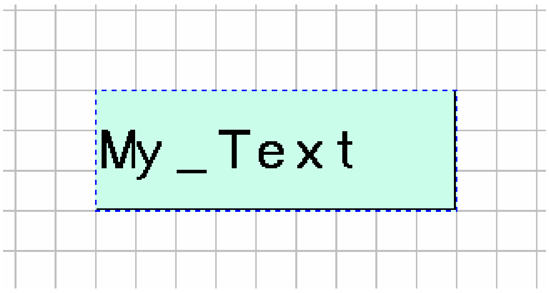
To add text to a page, select the **Text** icon in the EIML toolbar and draw the area to be used by the text in the page by holding down the left mouse key and dragging the pointer.



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



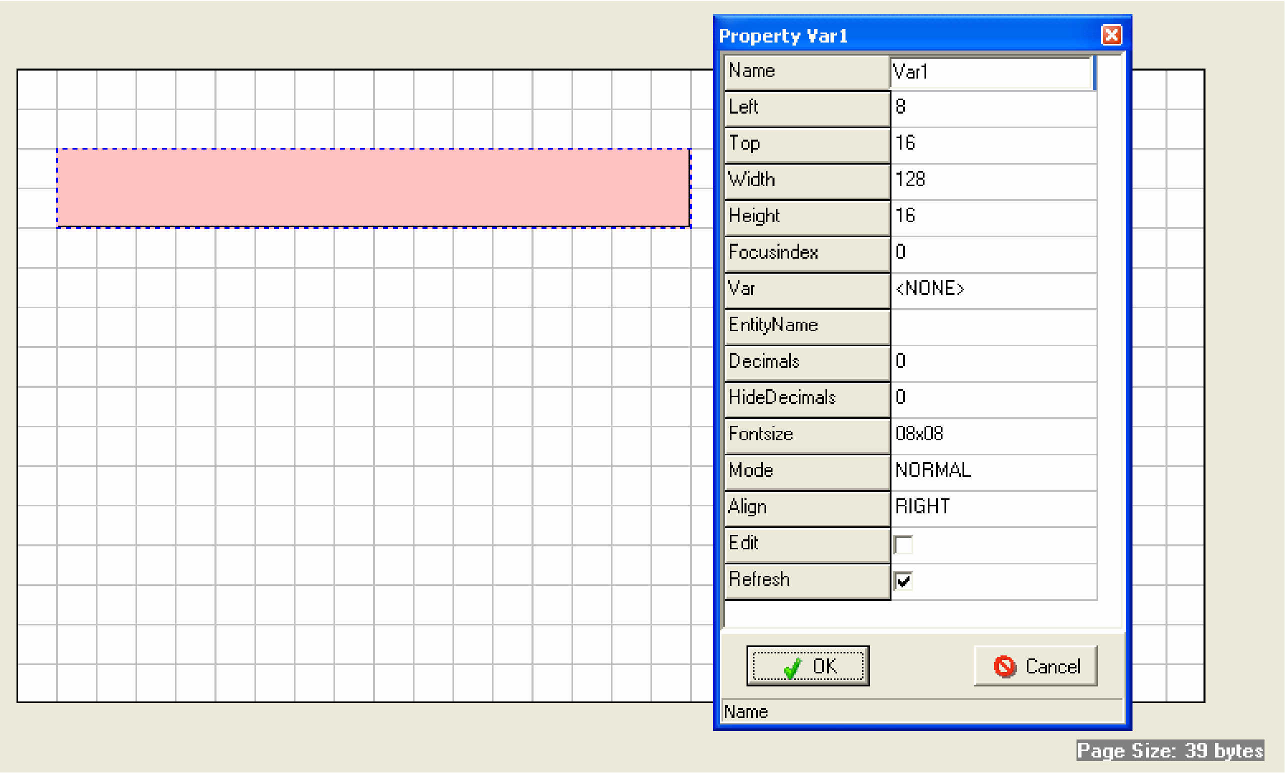
Now, to enter the text you wish to display, open the properties window (for example by double-clicking the text element) and change the Text property as required. After that, you may go on setting the remaining properties (Fontsize, Align, etc.) and the element will be displayed with a new look:



Variable V

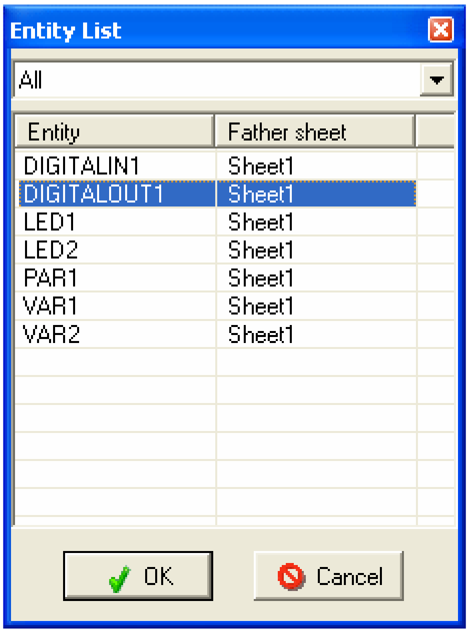
Variable objects allow you to display and, if required, set the values of internal states, inputs, outputs, etc.

To add a variable to a page, proceed just as you would to add a text.



Name	Var1
Left	8
Top	16
Width	128
Height	16
Focusindex	0
Var	<NONE>
EntityName	
Decimals	0
HideDecimals	0
Fontsize	08x08
Mode	NORMAL
Align	RIGHT
Edit	□
Refresh	☑

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.



Entity	Father sheet
DIGITALIN1	Sheet1
DIGITALOUT1	Sheet1
LED1	Sheet1
LED2	Sheet1
PAR1	Sheet1
VAR1	Sheet1
VAR2	Sheet1

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.
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.
Mode	A variable may be displayed in four different modes. ➢ NORMAL ➢ NEGATIVE ➢ NORMAL-BLINKING ➢ NEGATIVE-BLINKING 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.
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 so as to avoid overloading communication between 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 that will be loaded and converted to black/white.
Mode(*)	An icon may be displayed in four different modes: ➢ NORMAL ➢ NEGATIVE ➢ NORMAL-BLINKING ➢ NEGATIVE-BLINKING

	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.

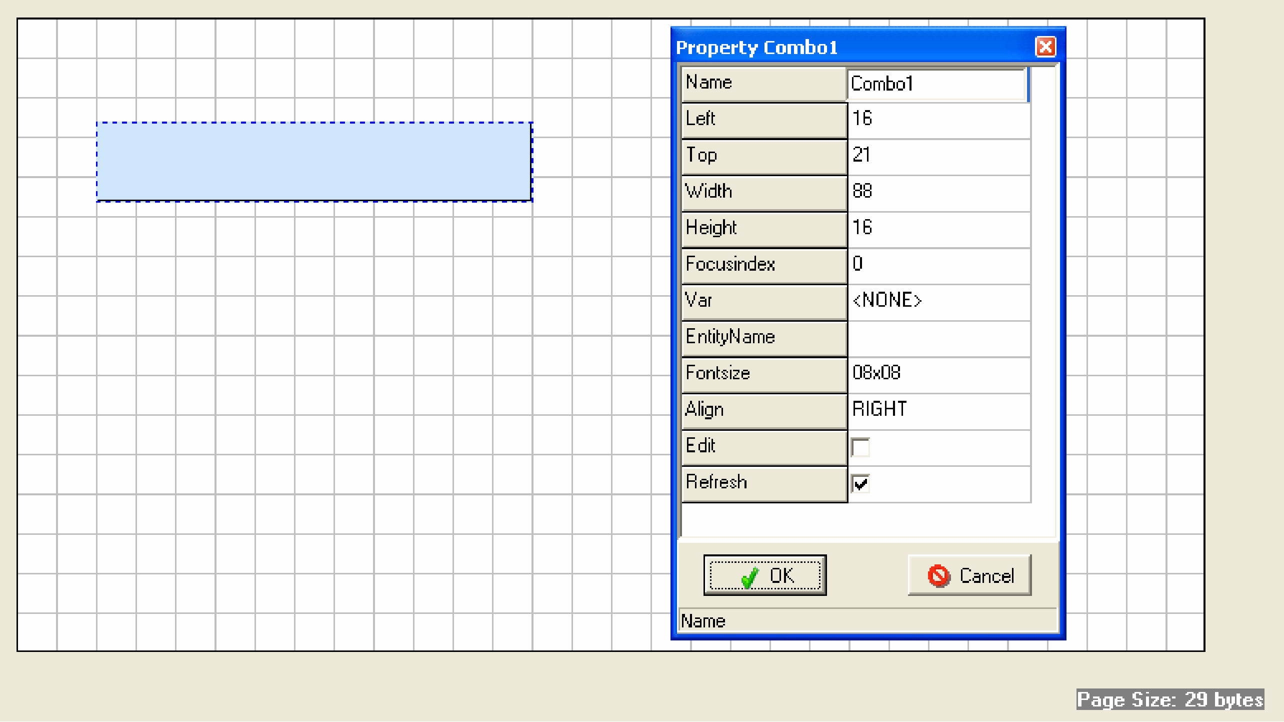
Note(*). When the **Var** property is set, the Mode is aligned with that property, whatever is the **Mode** property.

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.
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:

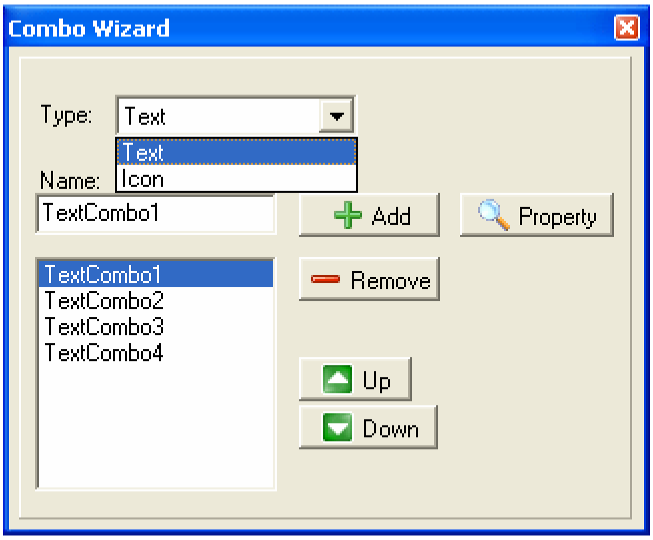


Property Combo1

Name	Combo1
Left	16
Top	21
Width	88
Height	16
Focusindex	0
Var	<NONE>
EntityName	
Fontsize	08x08
Align	RIGHT
Edit	□
Refresh	☑

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:



Combo Wizard

Type: Text

Name: TextCombo1

TextCombo1
TextCombo2
TextCombo3
TextCombo4

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.

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

Table

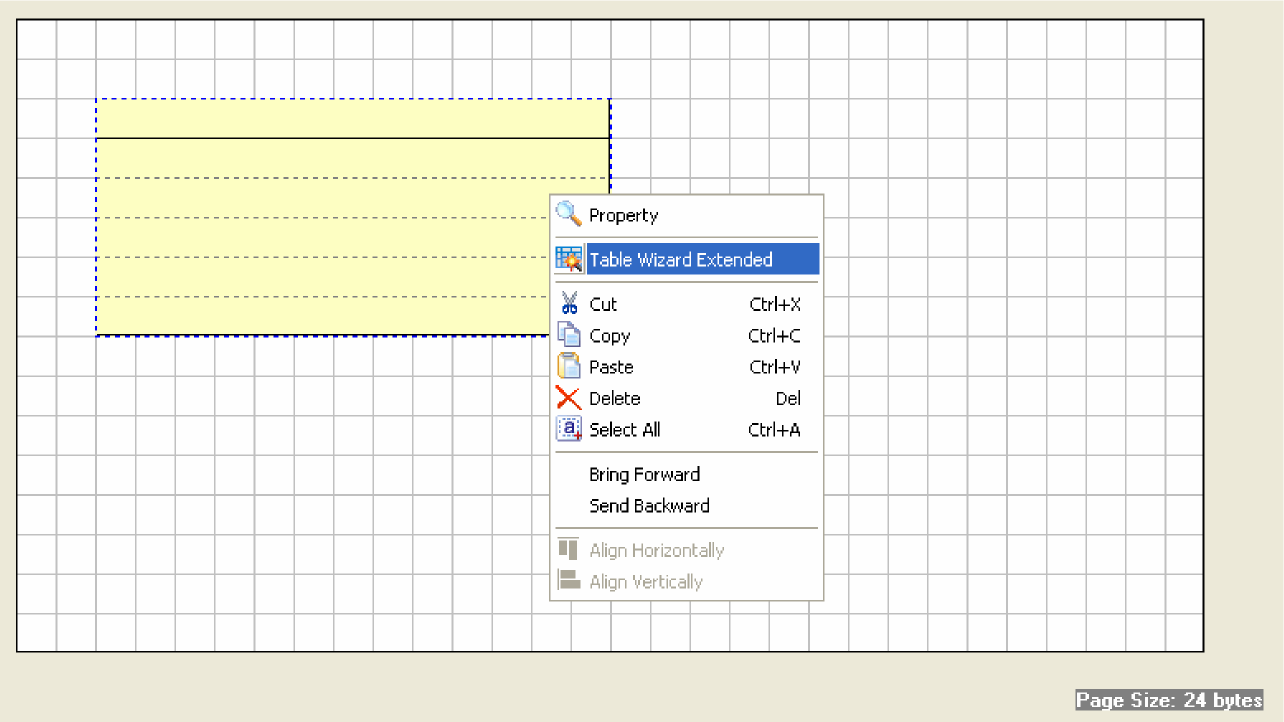
Tables are an additional innovative object type designed to help you develop graphical interfaces and characterized by being **scrollable**.

Therefore, using tables you will be able to display large amounts of data without needing to load more than one page.

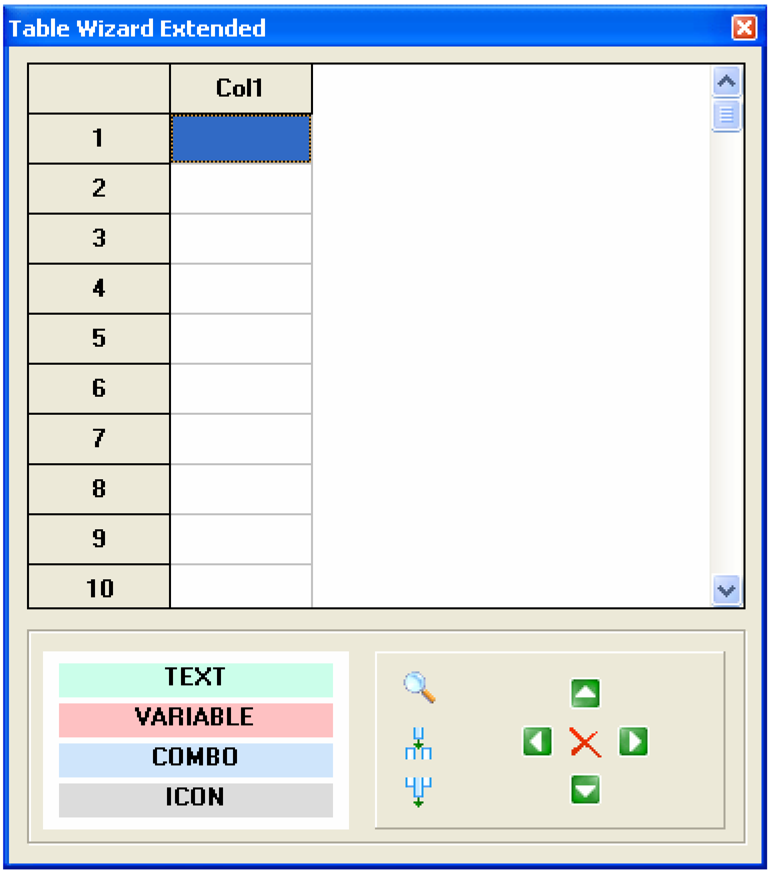
If we add a table to a page, we will be able to display its properties:

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.
Fontsize	This property defines the font size of all text elements displayed in the table.
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.

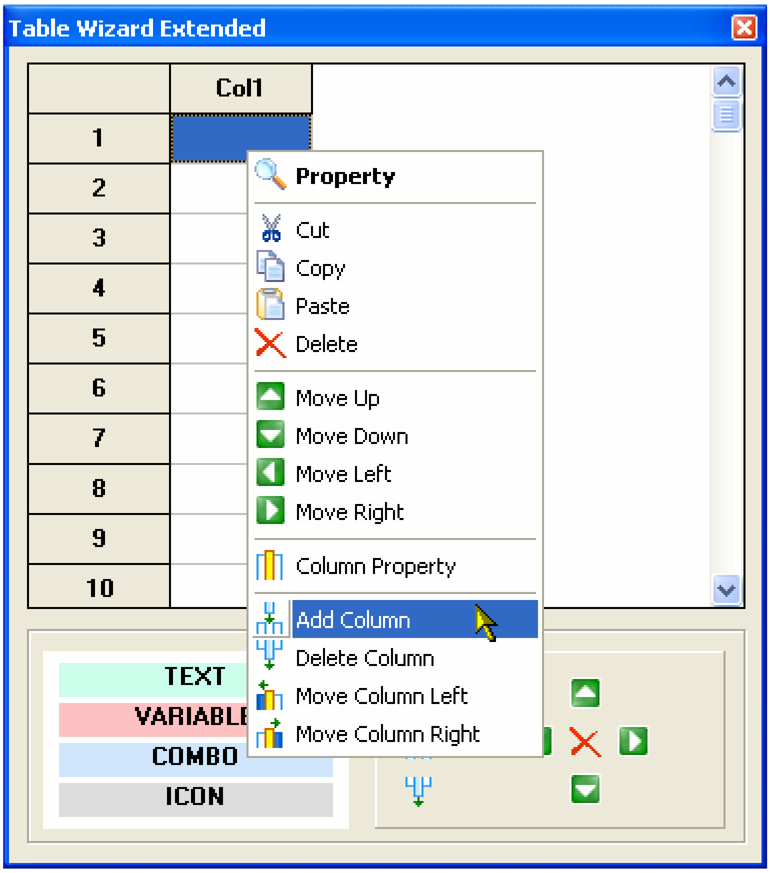
To add objects to a table, use the **Table Wizard Extended** (right-click the element and then choose Table Wizard Extended).



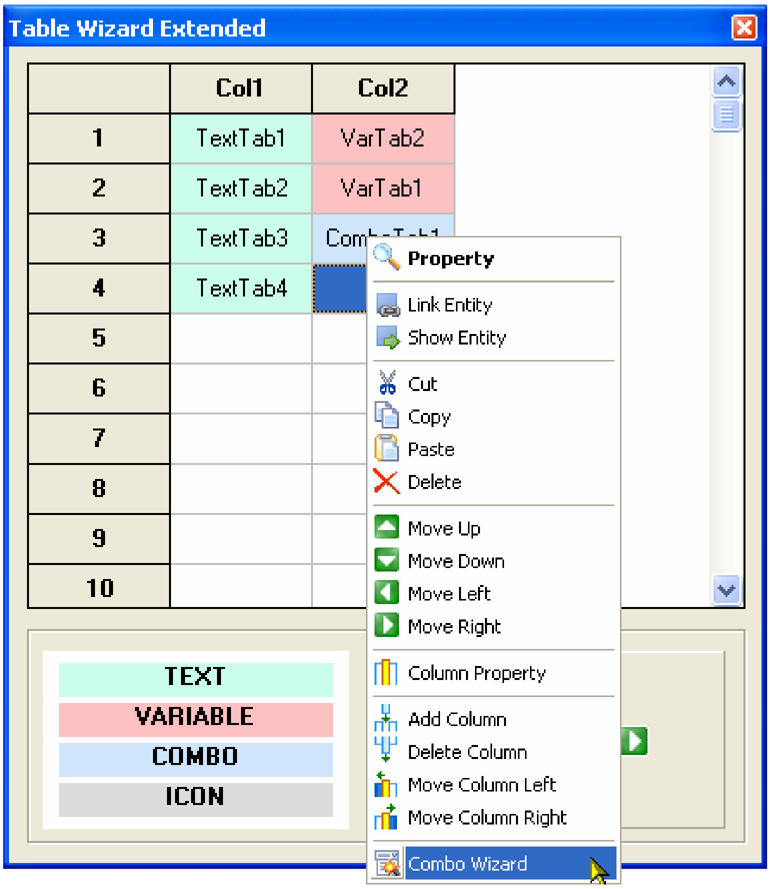
The following window will appear:



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*.
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:



	Col1	Col2
1	TextTab1	VarTab2
2	TextTab2	VarTab1
3	TextTab3	ComboTab1
4	TextTab4	
5		
6		
7		
8		
9		
10		

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.

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.

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.
Filled	Indicates whether or not the rectangle should be filled with the specified color.

4 ADVANCED OPERATIONS

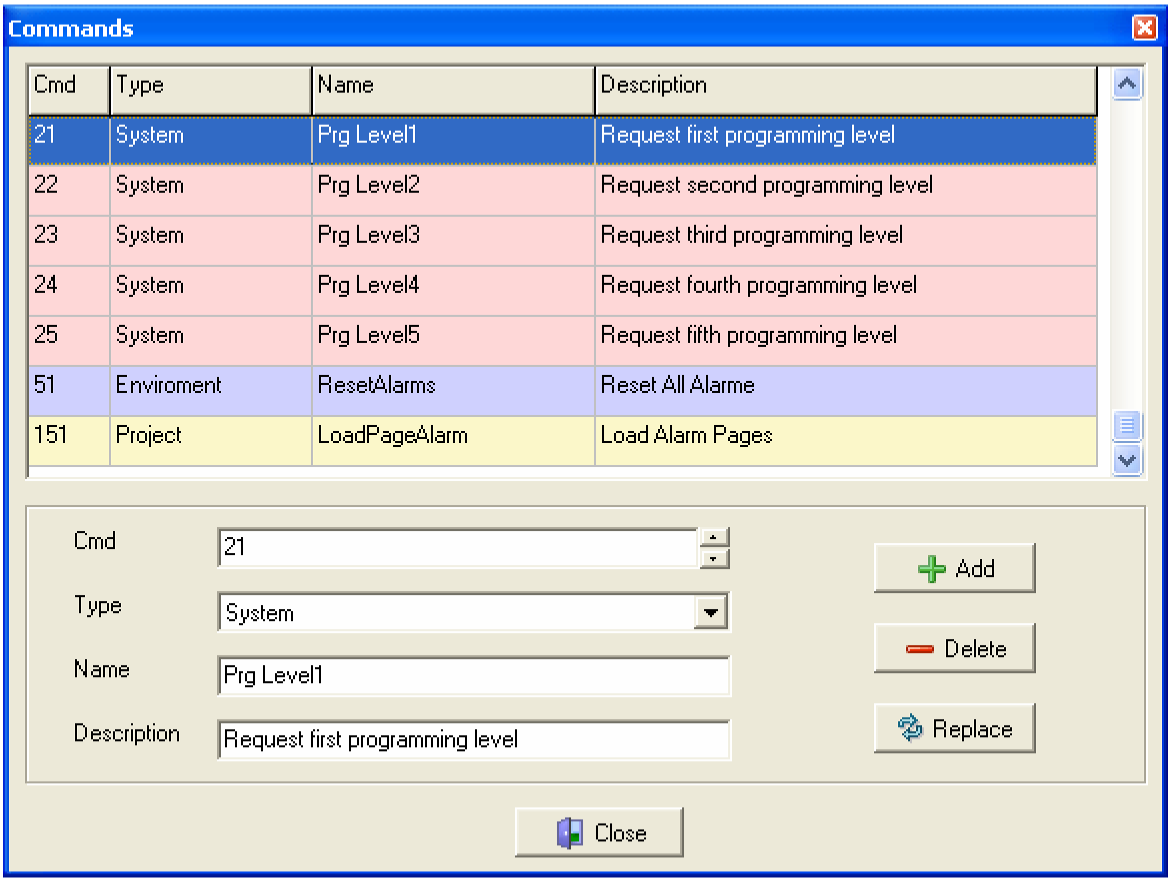
4.1 Commands

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.



Cmd	Type	Name	Description
21	System	Prg Level1	Request first programming level
22	System	Prg Level2	Request second programming level
23	System	Prg Level3	Request third programming level
24	System	Prg Level4	Request fourth programming level
25	System	Prg Level5	Request fifth programming level
51	Enviroment	ResetAlarms	Reset All Alarme
151	Project	LoadPageAlarm	Load Alarm Pages

Cmd: 21
Type: System
Name: Prg Level1
Description: Request first programming level

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).
- **Upload Par App**: Command to load all application parameters from a programming key (warning: this operation may take some time to complete).
- **Download Par App**: Command to save all application parameters to a programming key (warning: this operation may take some time to complete) .
- **Erase Historian**: Command to erase the Event Historian (warning: all data will be deleted) .
- **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*).

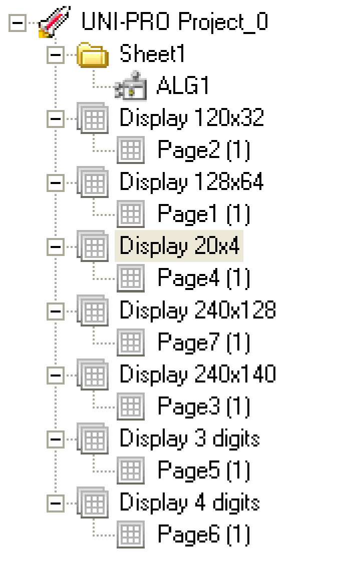
4.2 Navigating through the pages

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 3 Digit	= 0
DISPLAY 7-Segments 4 Digit	= 1
4X16 ALPHANUMERIC DISPLAY	= 2
4X20 ALPHANUMERIC DISPLAY	= 3
128x64 GRAPHIC DISPLAY	= 4
240x128 GRAPHIC DISPLAY	= 5
240x140 GRAPHIC DISPLAY	= 6
120x32 GRAPHIC DISPLAY	= 7

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.

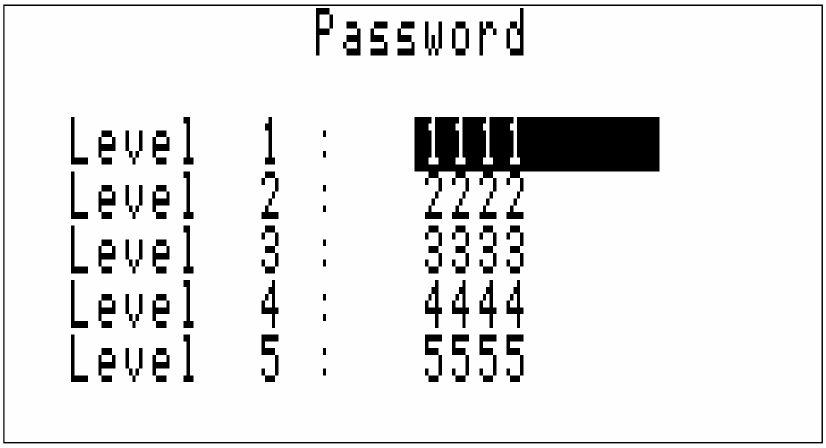
In addition, setting *Timeout* and *TimeoutPage* you can define the period of inactivity before a page is loaded.

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.

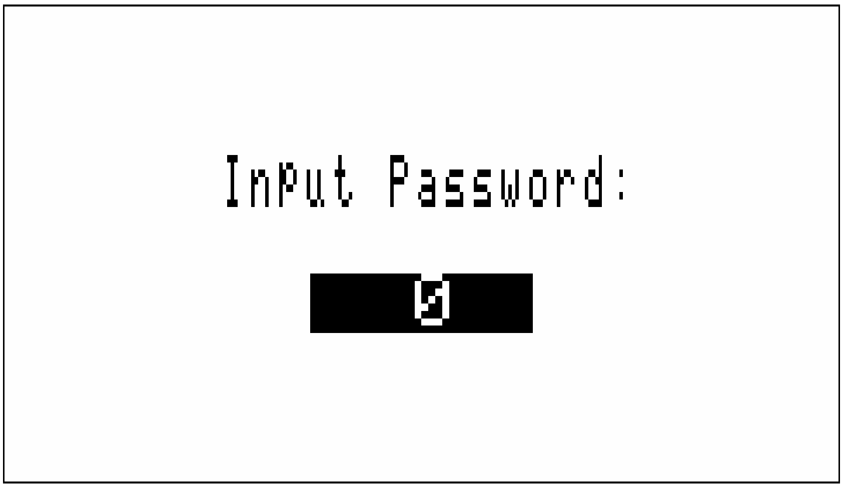
The protection system adopted is based on 5 protection levels that may be assigned to each EIML page by setting the *Level* property. Each level is linked to a different password in the controller. When a display requests a page whose Level property is set to a value higher than zero, the controller 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:



```
         Password

Level 1 :   1111
Level 2 :   2222
Level 3 :   3333
Level 4 :   4444
Level 5 :   5555
```

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.



```
Input Password:

       0
```

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.

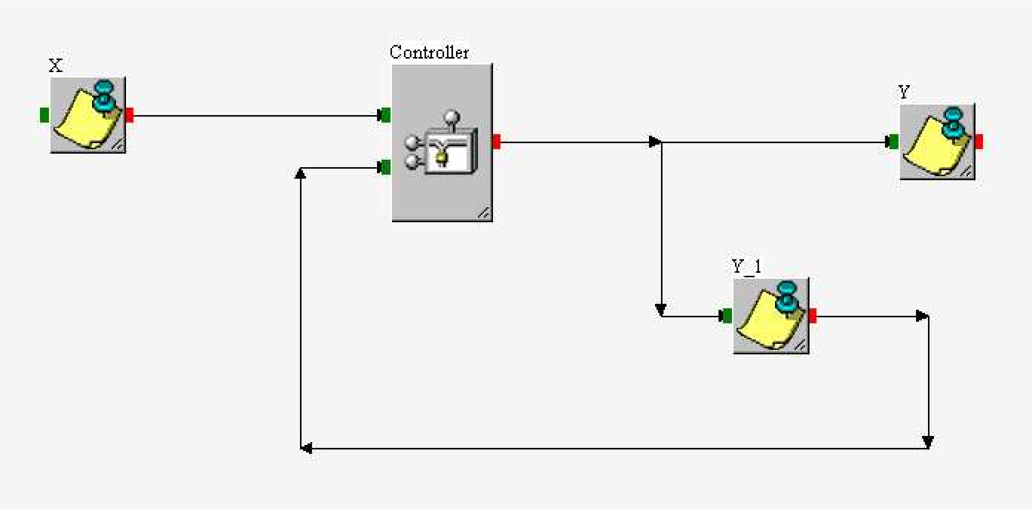
If no keys are pressed on the keypad during a pre-set period of time (system parameter, default 60 seconds), the authorization level will be reset.

4.4 Defining execution tasks

All calculation operations defined through links of entities can be executed in three 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 one of the high priority tasks (Timed100ms and Timed5ms), which execute under interrupts every 100 and 5 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.

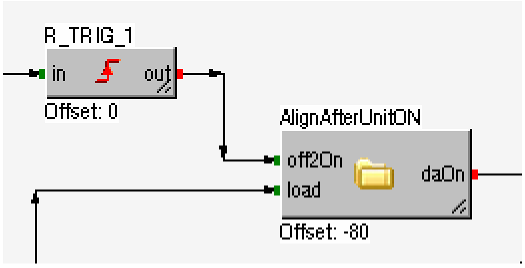


In this case, we want output Y to be calculated by the algorithm using input Y_1 (which represents the previous output) *before* the input is updated! To assign a specific order to the execution of this part of the program, we need to set the right sequence to the values of the *Order* property of all 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:

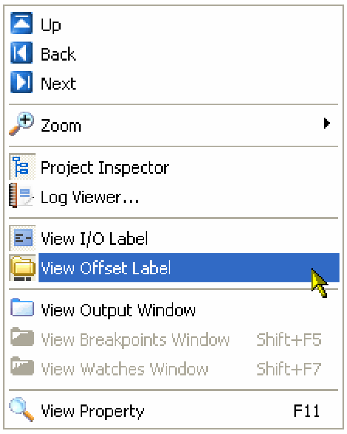
- Order X = 0
- Order Y = 1
- Order Y_1 = 2

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.

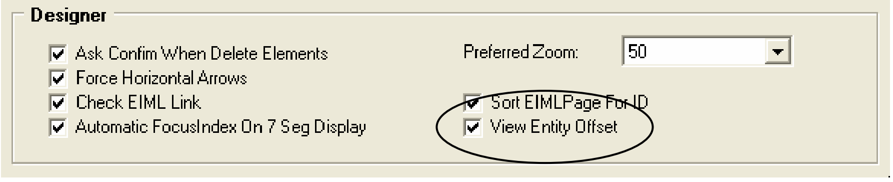


To activate this function go to:
1) Menù View->View Offset Label



Or

2) From *Settings*



Designer

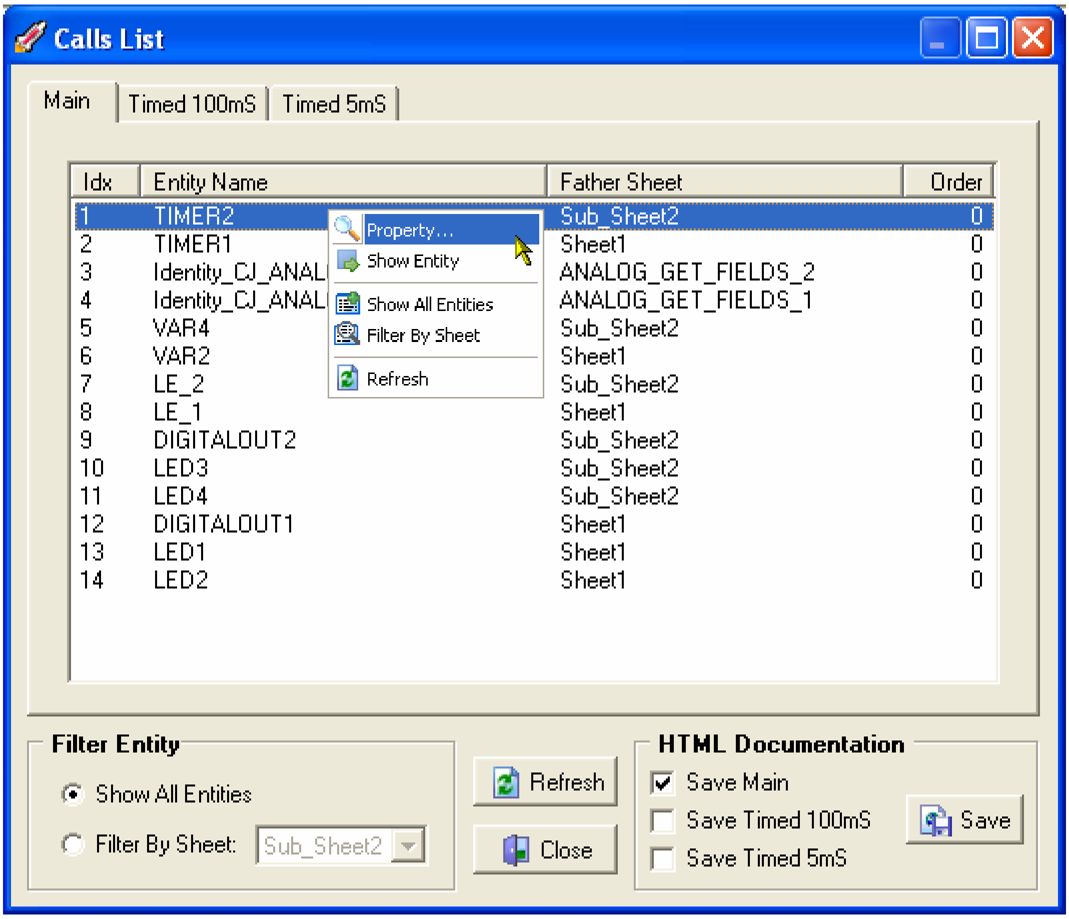
☑ Ask Confim When Delete Elements
☑ Force Horizontal Arrows
☑ Check EIML Link
☑ Automatic FocusIndex On 7 Seg Display

Preferred Zoom: 50

☑ Sort EIMLPage For ID
☑ View Entity Offset

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



Idx	Entity Name	Father Sheet	Order
1	TIMER2	Sub_Sheet2	0
2	TIMER1	Sheet1	0
3	Identity_CJ_ANAL	ANALOG_GET_FIELDS_2	0
4	Identity_CJ_ANAL	ANALOG_GET_FIELDS_1	0
5	VAR4	Sub_Sheet2	0
6	VAR2	Sheet1	0
7	LE_2	Sub_Sheet2	0
8	LE_1	Sheet1	0
9	DIGITALOUT2	Sub_Sheet2	0
10	LED3	Sub_Sheet2	0
11	LED4	Sub_Sheet2	0
12	DIGITALOUT1	Sheet1	0
13	LED1	Sheet1	0
14	LED2	Sheet1	0

● Show All Entities
○ Filter By Sheet: Sub_Sheet2

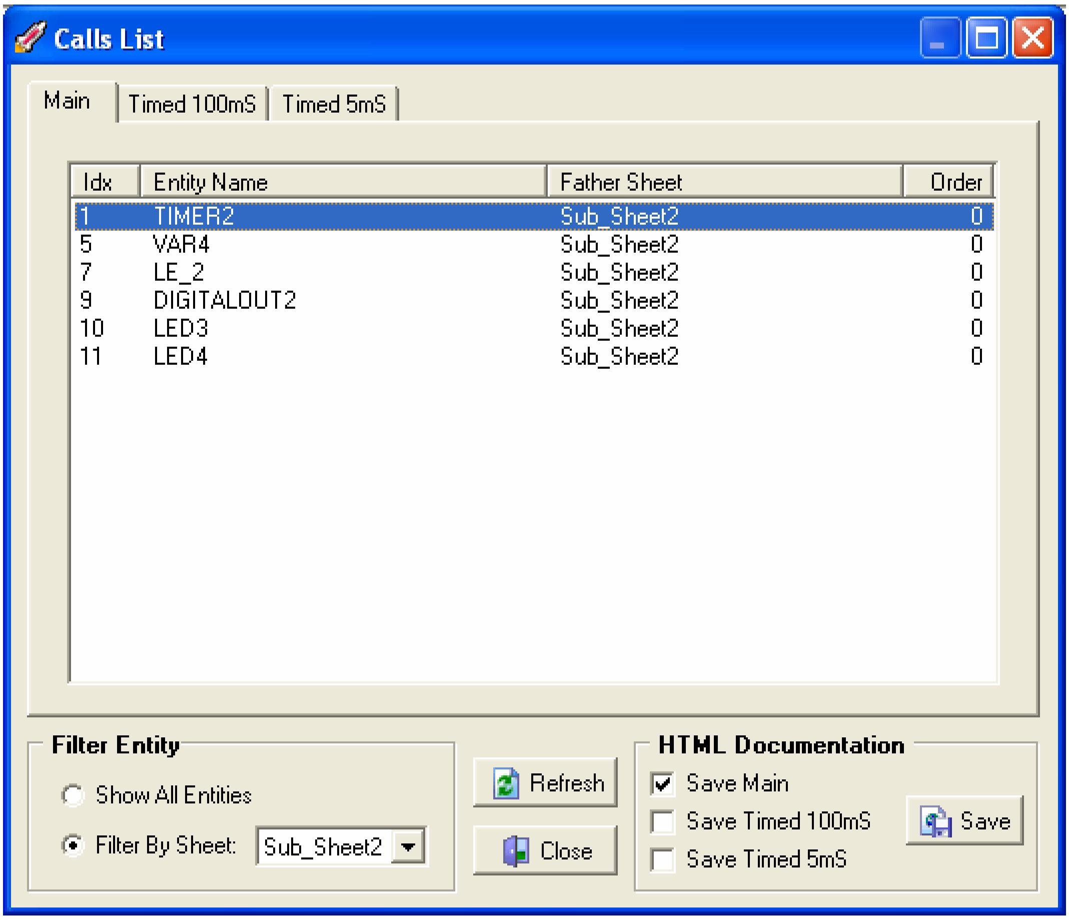
☑ Save Main
□ Save Timed 100mS
□ Save Timed 5mS

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.



Idx	Entity Name	Father Sheet	Order
1	TIMER2	Sub_Sheet2	0
5	VAR4	Sub_Sheet2	0
7	LE_2	Sub_Sheet2	0
9	DIGITALOUT2	Sub_Sheet2	0
10	LED3	Sub_Sheet2	0
11	LED4	Sub_Sheet2	0

○ Show All Entities
● Filter By Sheet: Sub_Sheet2

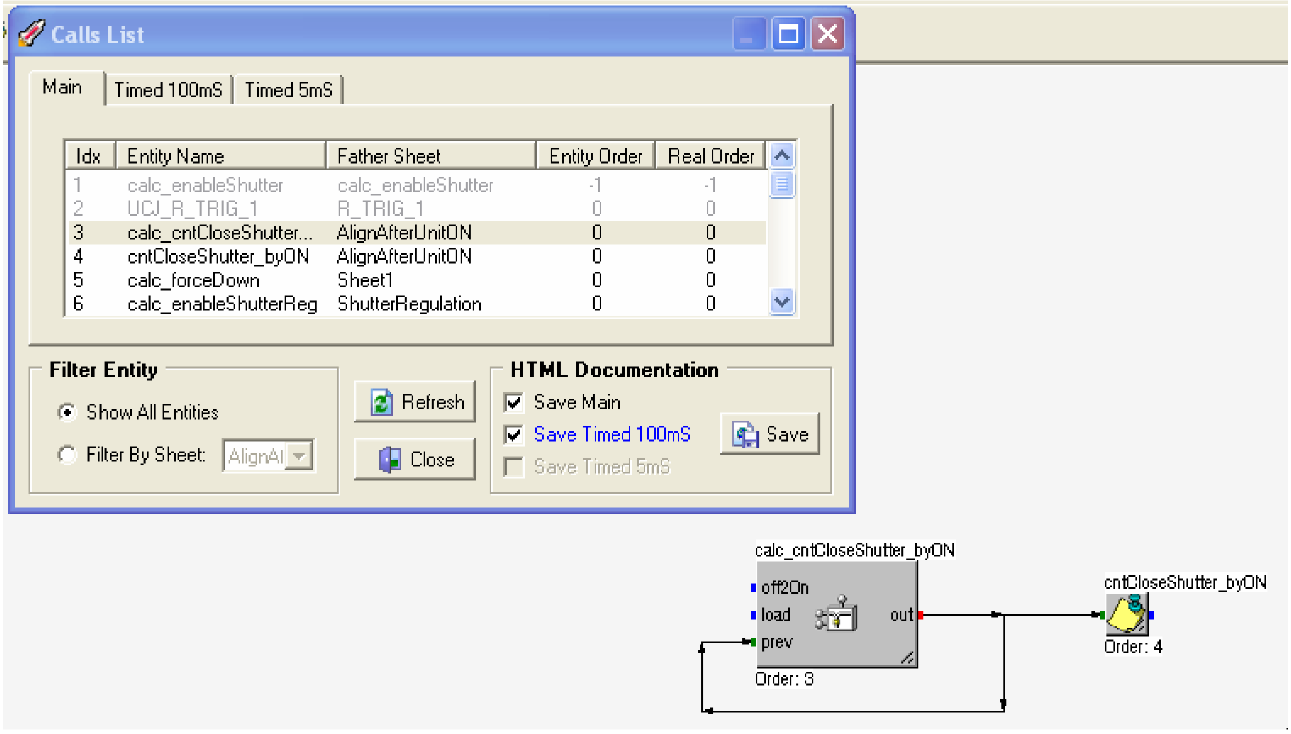
☑ Save Main
□ Save Timed 100mS
□ Save Timed 5mS

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.



Calls List

Main | Timed 100mS | Timed 5mS

Idx	Entity Name	Father Sheet	Entity Order	Real Order
1	calc_enableShutter	calc_enableShutter	-1	-1
2	UCJ_R_TRIG_1	R_TRIG_1	0	0
3	calc_cntCloseShutter...	AlignAfterUnitON	0	0
4	cntCloseShutter_byON	AlignAfterUnitON	0	0
5	calc_forceDown	Sheet1	0	0
6	calc_enableShutterReg	ShutterRegulation	0	0

Filter Entity

● Show All Entities

○ Filter By Sheet: AlignAl

HTML Documentation

☑ Save Main

☑ Save Timed 100mS

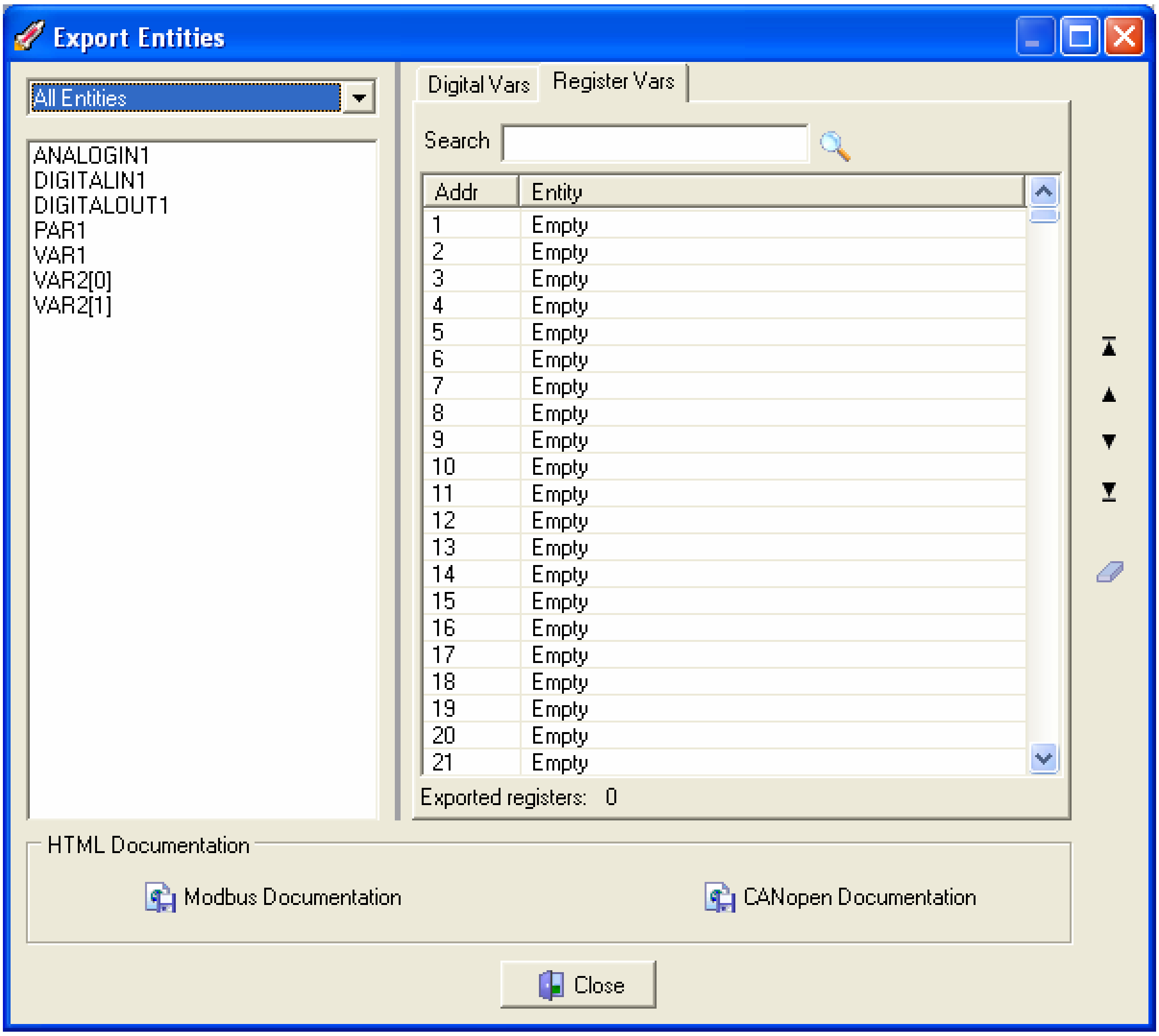
□ Save Timed 5mS

In this case it is immediate understood that the variable is calculated right after the algorithm.

4.6 Sorting 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/Position…* from the menu to open it):



All Entities

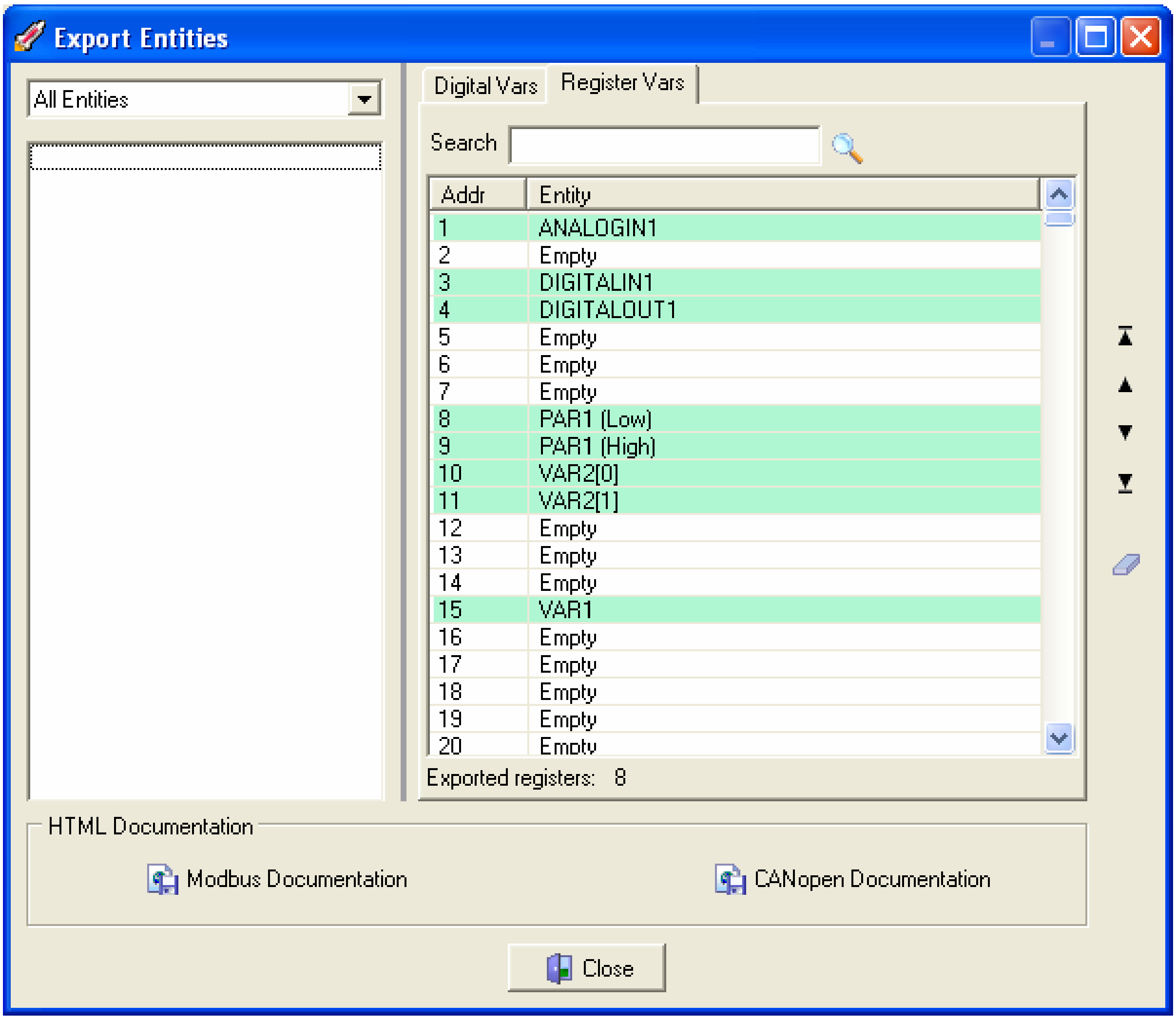
ANALOGIN1
DIGITALIN1
DIGITALOUT1
PAR1
VAR1
VAR2[0]
VAR2[1]

Addr	Entity
1	Empty
2	Empty
3	Empty
4	Empty
5	Empty
6	Empty
7	Empty
8	Empty
9	Empty
10	Empty
11	Empty
12	Empty
13	Empty
14	Empty
15	Empty
16	Empty
17	Empty
18	Empty
19	Empty
20	Empty
21	Empty

Exported registers: 0

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.



Addr	Entity
1	ANALOGIN1
2	Empty
3	DIGITALIN1
4	DIGITALOUT1
5	Empty
6	Empty
7	Empty
8	PAR1 (Low)
9	PAR1 (High)
10	VAR2[0]
11	VAR2[1]
12	Empty
13	Empty
14	Empty
15	VAR1
16	Empty
17	Empty
18	Empty
19	Empty
20	Empty

Exported registers: 8

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.
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

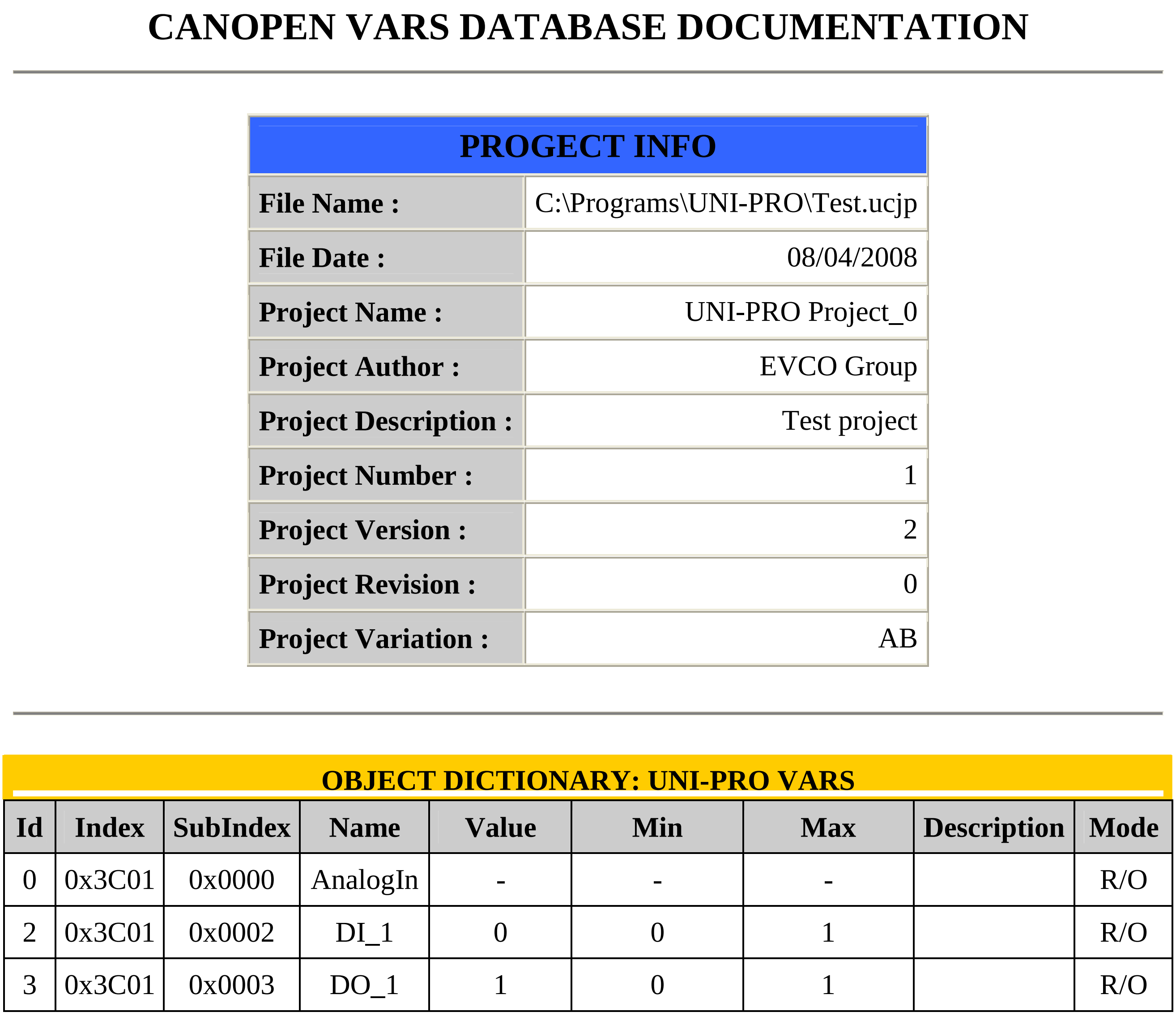
Register Vars	
Entity	*Data type*
Analog In	CJ_ANALOG: only the value
Analog Out	
Clock	

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.

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*.
Following there is an example of the created documentation for the CANopen protocol.



CANOPEN VARS DATABASE DOCUMENTATION

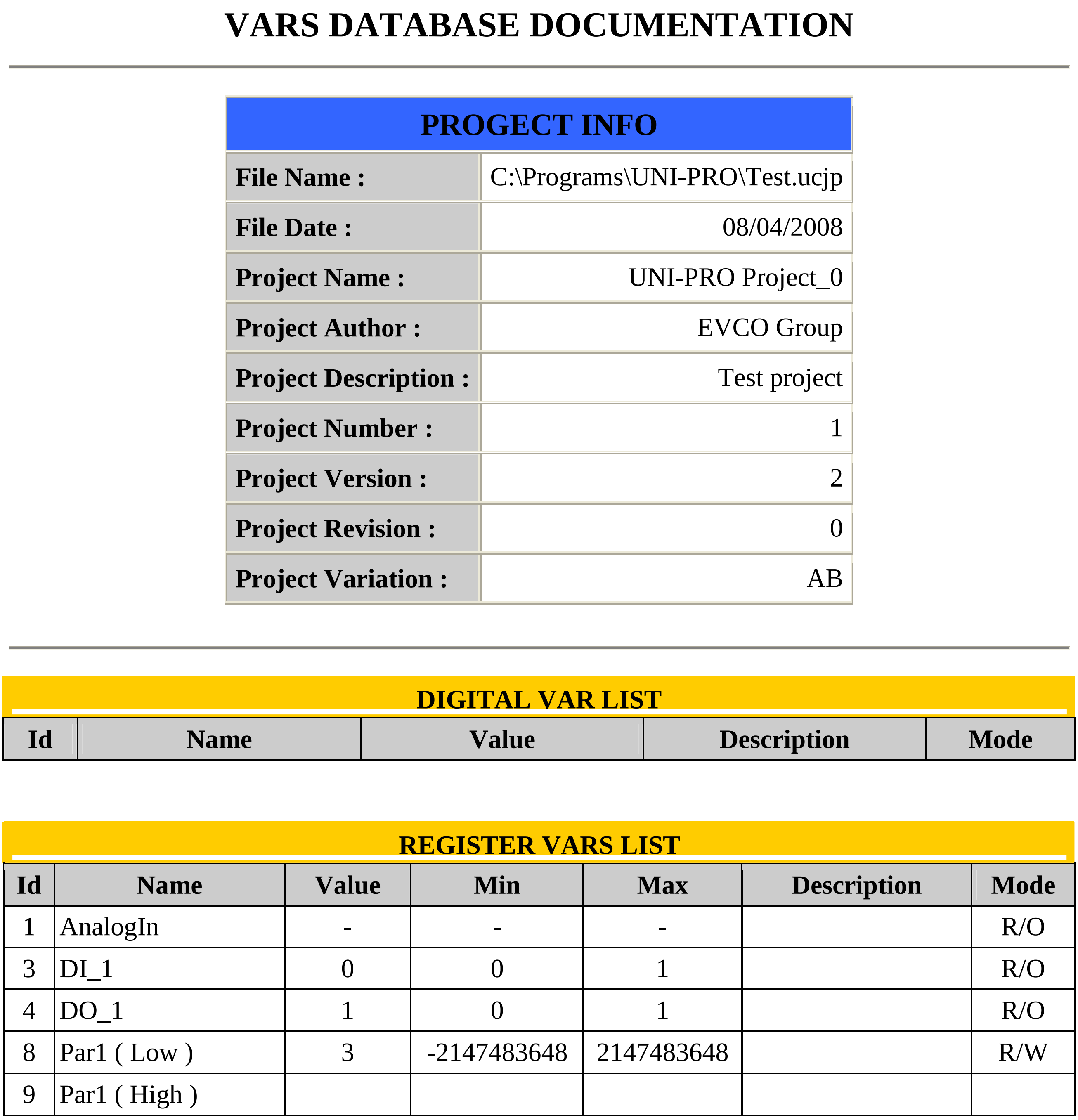
PROGECT INFO	
File Name :	C:\Programs\UNI-PRO\Test.ucjp
File Date :	08/04/2008
Project Name :	UNI-PRO Project_0
Project Author :	EVCO Group
Project Description :	Test project
Project Number :	1
Project Version :	2
Project Revision :	0
Project Variation :	AB

OBJECT DICTIONARY: UNI-PRO VARS

Id	Index	SubIndex	Name	Value	Min	Max	Description	Mode
0	0x3C01	0x0000	AnalogIn	-	-	-		R/O
2	0x3C01	0x0002	DI_1	0	0	1		R/O
3	0x3C01	0x0003	DO_1	1	0	1		R/O

7	0x3C01	0x0007	Par1	3	-2147483648	2147483648		R/W
9	0x3C01	0x0009	Var2[0]	0	0	1		R/W
10	0x3C01	0x000A	Var2[1]	0	0	1		R/W
14	0x3C01	0x000E	Var1	-12	-32768	32767		R/W

The Modbus document contains the information to access to the project variables through Modbus read/write commands of *coils* and *registers*.
Following there is an example of the created documentation for the Modbus protocol.



VARS DATABASE DOCUMENTATION

PROGECT INFO	
File Name :	C:\Programs\UNI-PRO\Test.ucjp
File Date :	08/04/2008
Project Name :	UNI-PRO Project_0
Project Author :	EVCO Group
Project Description :	Test project
Project Number :	1
Project Version :	2
Project Revision :	0
Project Variation :	AB

DIGITAL VAR LIST

Id	Name	Value	Description	Mode

REGISTER VARS LIST

Id	Name	Value	Min	Max	Description	Mode
1	AnalogIn	-	-	-		R/O
3	DI_1	0	0	1		R/O
4	DO_1	1	0	1		R/O
8	Par1 (Low)	3	-2147483648	2147483648		R/W
9	Par1 (High)					

10	Var2[0]	0	0	1		R/W
11	Var2[1]	0	0	1		R/W
15	Var1	-12	-32768	32767		R/W

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)**

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

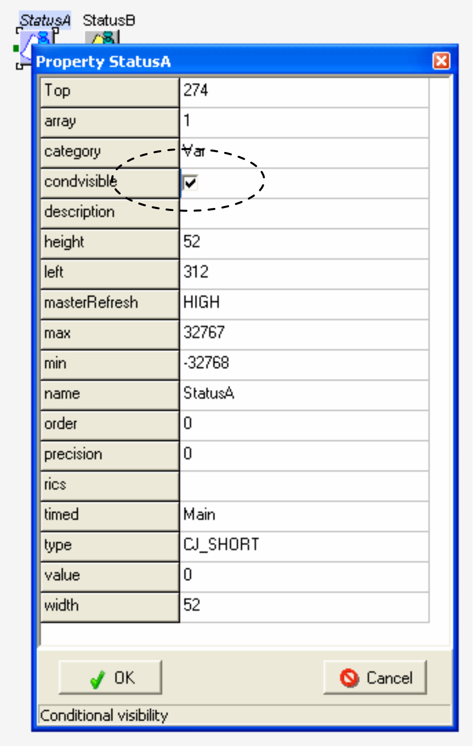
A function call as **CJ_SetCondVisBit(idx, 1)** made the entity at the *idx* address always **hide**.

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.

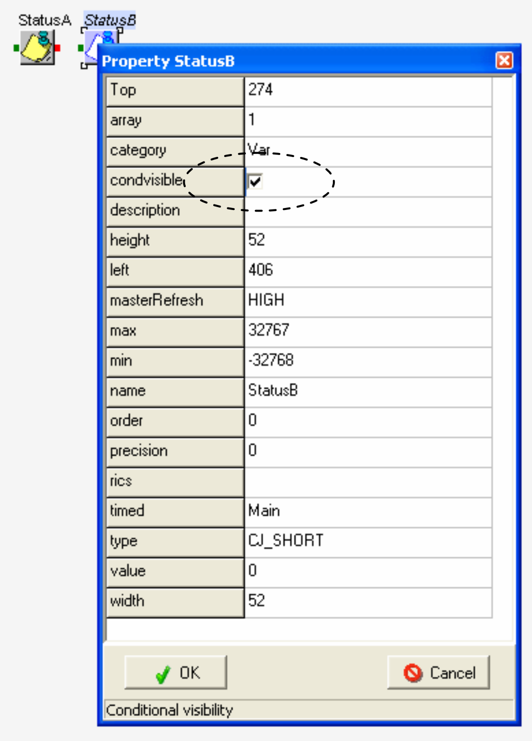
As first step are required to select the *condvisible* property of both entities.



Property StatusA

Top	274
array	1
category	Var
condvisible	☑
description	
height	52
left	312
masterRefresh	HIGH
max	32767
min	-32768
name	StatusA
order	0
precision	0
rics	
timed	Main
type	CJ_SHORT
value	0
width	52

Conditional visibility

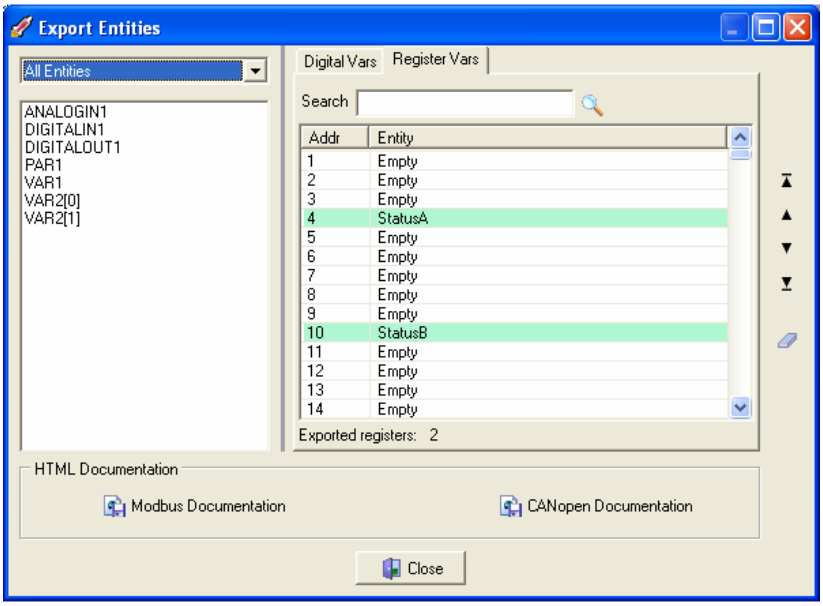


Property StatusB

Top	274
array	1
category	Var
condvisible	☑
description	
height	52
left	406
masterRefresh	HIGH
max	32767
min	-32768
name	StatusB
order	0
precision	0
rics	
timed	Main
type	CJ_SHORT
value	0
width	52

Conditional visibility

As told before the two variables must be exported on Modbus table:



ANALOGIN1
DIGITALIN1
DIGITALOUT1
PAR1
VAR1
VAR2[0]
VAR2[1]

Addr	Entity
1	Empty
2	Empty
3	Empty
4	StatusA
5	Empty
6	Empty
7	Empty
8	Empty
9	Empty
10	StatusB
11	Empty
12	Empty
13	Empty
14	Empty

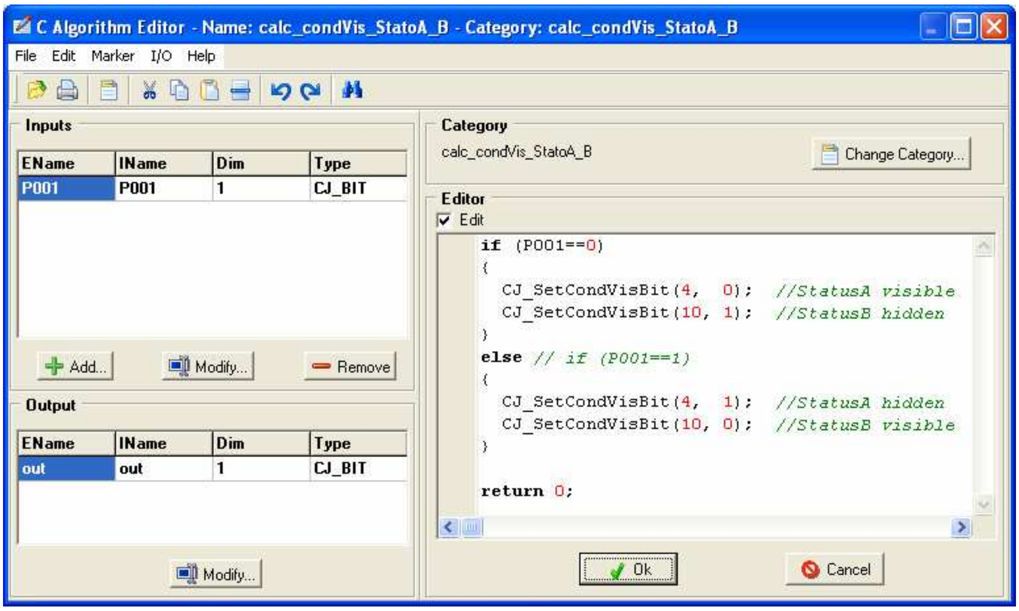
Exported registers: 2

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:



Category: calc_condVis_StatoA_B

Inputs

EName	IName	Dim	Type
P001	P001	1	CJ_BIT

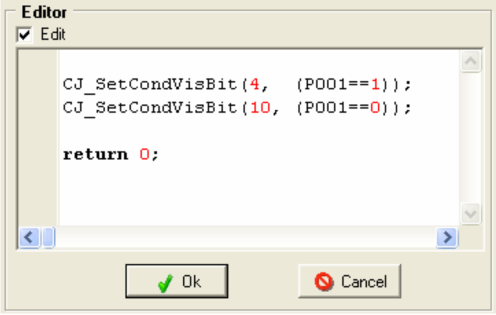
Output

EName	IName	Dim	Type
out	out	1	CJ_BIT

Editor ☑ Edit

```
if (P001==0)
{
   CJ_SetCondVisBit(4,  0);  //StatusA visible
   CJ_SetCondVisBit(10, 1);  //StatusB hidden
}
else // if (P001==1)
{
   CJ_SetCondVisBit(4,  1);  //StatusA hidden
   CJ_SetCondVisBit(10, 0);  //StatusB visible
}

return 0;
```

Same result we have as re-write the algorithm as a condition for the function *value* parameter:

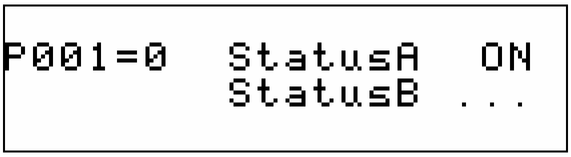


☑ Edit

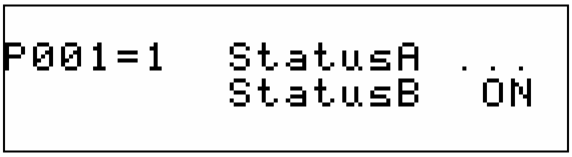
```

CJ_SetCondVisBit(4,  (P001==1));
CJ_SetCondVisBit(10, (P001==0));

return 0;
```

The user interface effect on 4x20 (V-View) or on 240x128 (V-Graph) are the following:

With P001=0: **With P001=1:**



```
P001=0   StatusA  ON
         StatusB ...
```



```
P001=1   StatusA ...
         StatusB  ON
```

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:

StatusX	1
StatusA	ON
StatusB	ON

With P001=0:

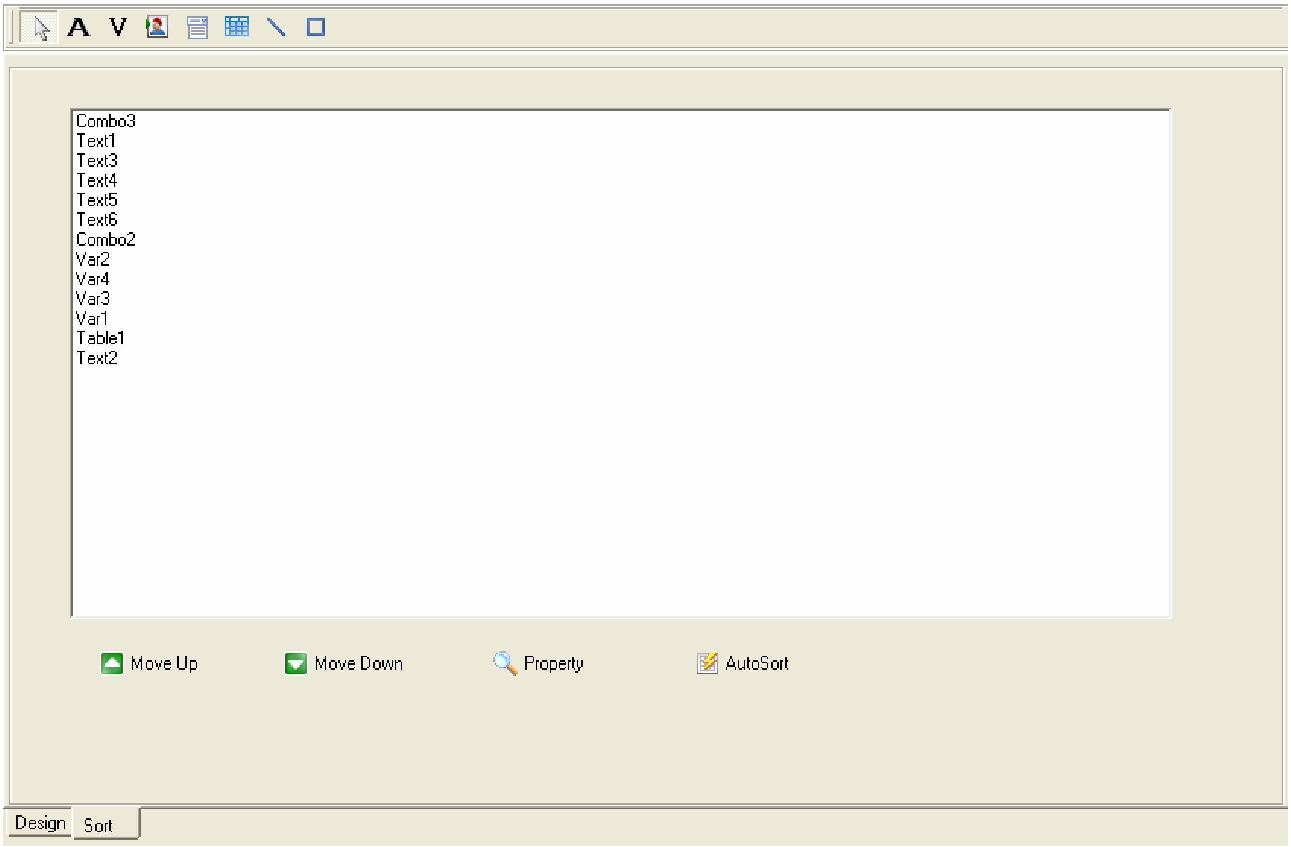
StatusX	1
StatusA	ON

With P001=1:

StatusX	1
StatusB	ON

4.8 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:



Combo3
Text1
Text3
Text4
Text5
Text6
Combo2
Var2
Var4
Var3
Var1
Table1
Text2

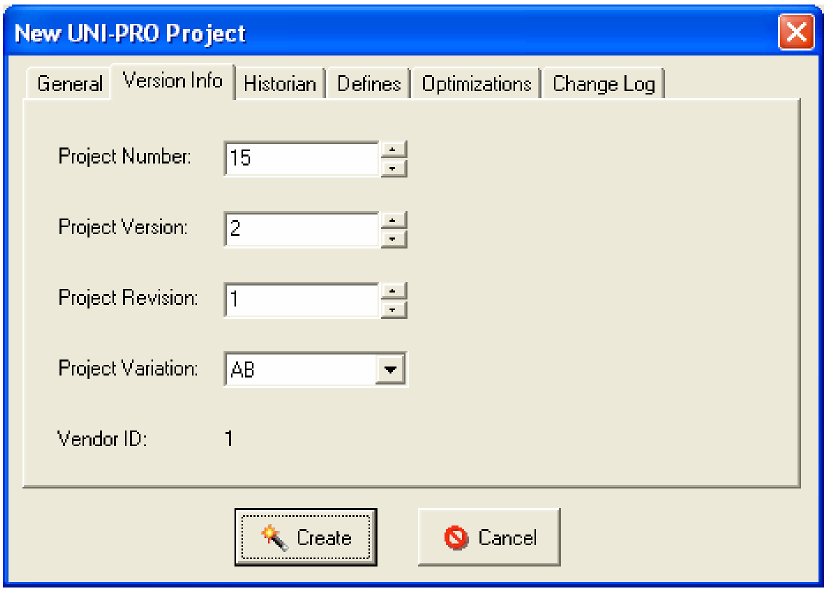
This window lists the names of the various graphic objects embedded in the page, in the order in which they will be processed when the page is displayed on a user interface. Select one or more objects and drag them with the mouse to change the order in which they will appear. Otherwise, you may use the *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.

4.9 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.

4.9.1 Version Info

In the Version Info window, you can set the project number, version, revision and variation for the user’s project.



Project Number: 15

Project Version: 2

Project Revision: 1

Project Variation: AB

Vendor ID: 1

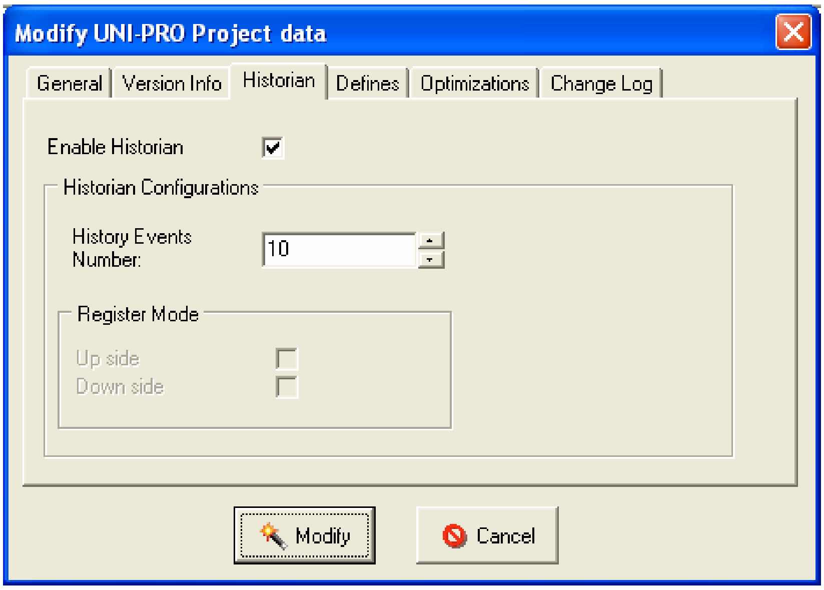
In the window there is the user’s *Vendor ID* that is read-only, instead the modifiable information are the following:

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.

4.9.2 Historian

In the Historian window, you can enable and configure the event&alarm recording functions available in certain UNI-PRO controllers.



Enable Historian ☑

History Events Number: 10

Up side □
Down side □

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.

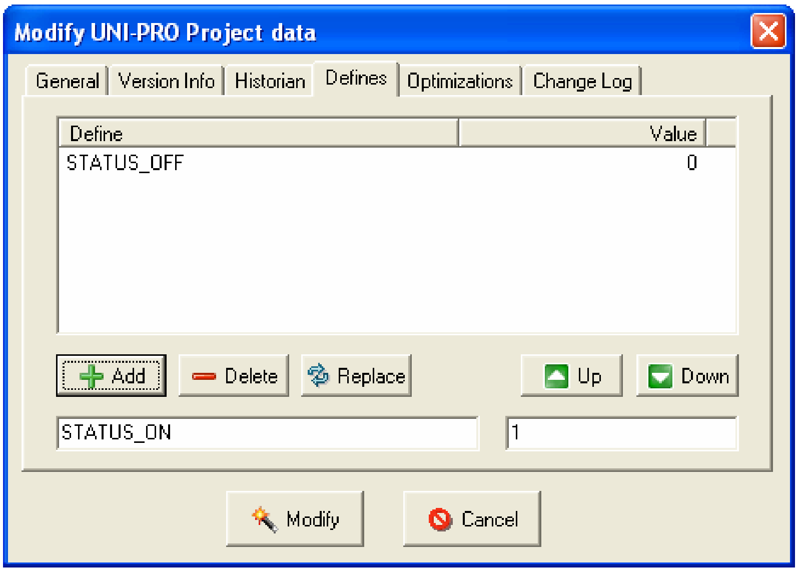
The following information is stored in the Event Historian:

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***).

4.9.3 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.



Define	Value
STATUS_OFF	0

STATUS_ON

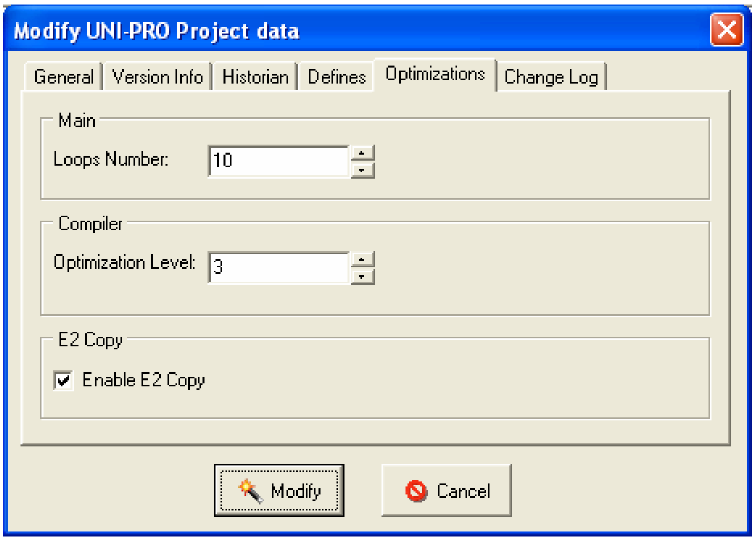
1

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*.

4.9.4 Optimizations

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.



Main

Loops Number: 10

Compiler

Optimization Level: 3

E2 Copy

☑ Enable E2 Copy

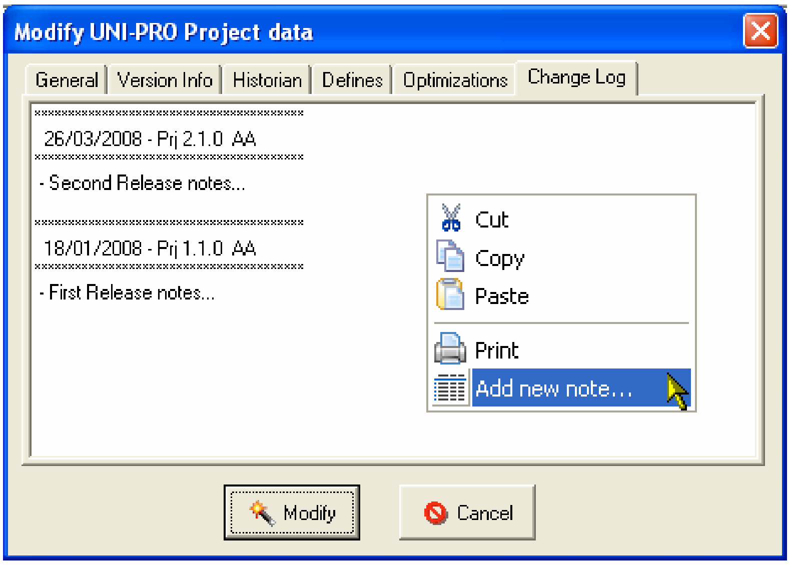
Loops Number : Using this configuration it is possible to specify the number of *actions* calculated in a group. For very large projects, by increasing this number it is possible to speed up the main program.

Optimizations Level: Optimization level of the compiler.

E2 Copy : The flag *Enable E2 Copy* allows you to enable or inhibit the copying of the configuration and application parameters to E2 memory using system commands.

4.9.5 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.



```
****************************************
 26/03/2008 - Prj 2.1.0  AA
****************************************
- Second Release notes...

****************************************
 18/01/2008 - Prj 1.1.0  AA
****************************************
- First Release notes...
```

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:

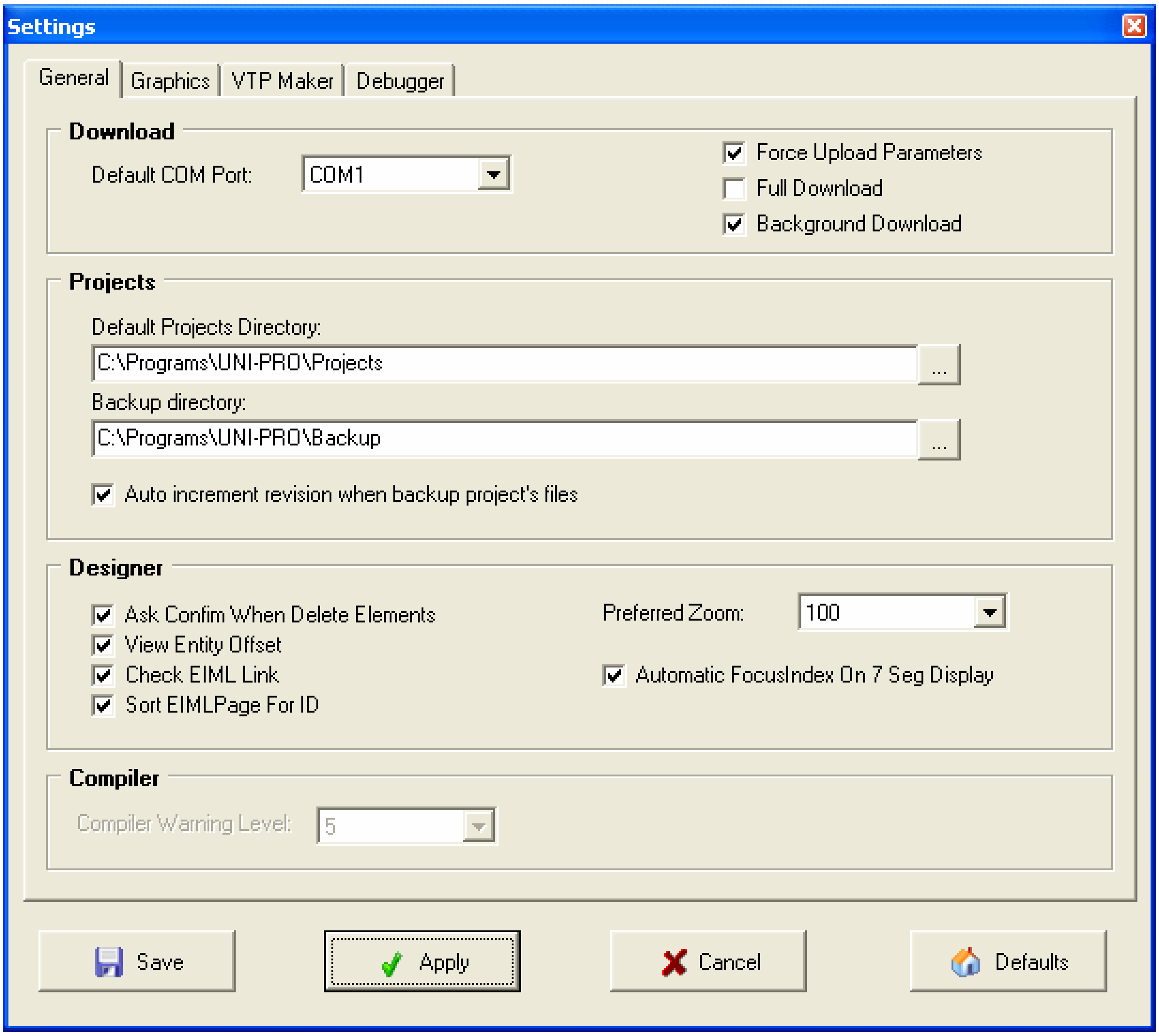
- printing of the Change Log's contents
- adding, in the current cursor position, a new note (a few rows with headings containing the date, and the current project version are also added).

4.10 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
- VTP Maker

The *General* screen contains the *Download*, *Projects*, *Designer* and *Compiler* sections. All the options of which will be analyzed here below:



Download

Default COM Port: COM1

☑ Force Upload Parameters

□ Full Download

☑ Background Download

Projects

Default Projects Directory: C:\Programs\UNI-PRO\Projects

Backup directory: C:\Programs\UNI-PRO\Backup

☑ Auto increment revision when backup project's files

Designer

☑ Ask Confim When Delete Elements

☑ View Entity Offset

☑ Check EIML Link

☑ Sort EIMLPage For ID

Preferred Zoom: 100

☑ Automatic FocusIndex On 7 Seg Display

Compiler

Compiler Warning Level: 5

Default COM port : Serial port used for downloading operations.

Force upload parameters : Forces the rewriting of the parameters in E2 during the initial execution after downloading.

Full download : Forces the full downloading of the entire executable file. To speed up the downloading options, it is possible to unselect this

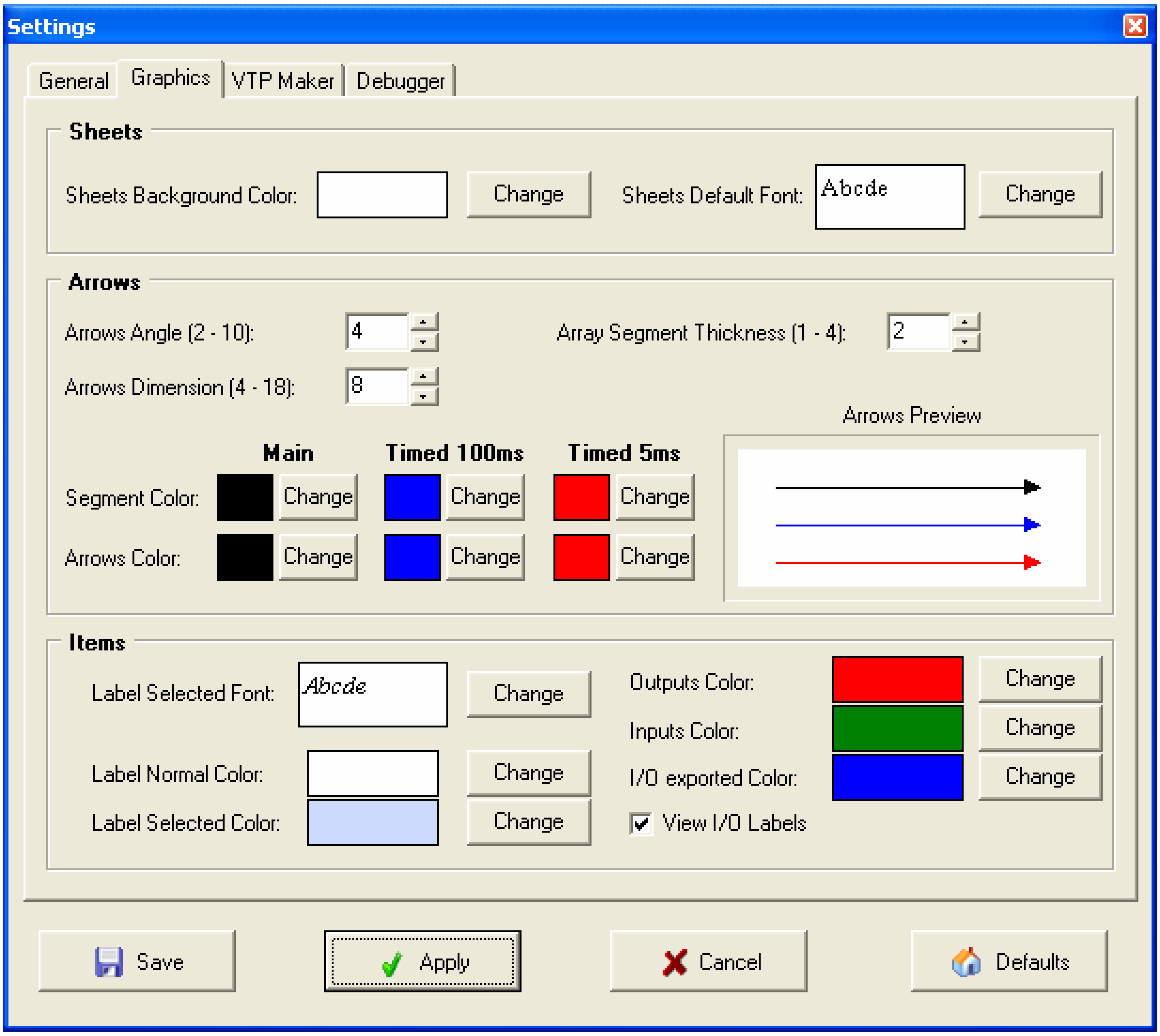
	option and, after having correctly deleted the flash memory present in the microcontroller, proceed to download only the flash memory used.

Background download:	Allows to operate on the project during the download phase.
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.
View Entity Offset:	If enabled, allow to display the offset property below the entity.

Compiler warning level :	Specifies the warning signal level used during the project compilation process. It is advisable not to alter the predefined value (5) unless otherwise indicated.

The *Graphics* screen contains the *Sheets, Arrows and Items* sections, which options will next be fully described:



Settings

General | Graphics | VTP Maker | Debugger

Sheets

Sheets Background Color: ⬜ Change

Sheets Default Font: Abcde Change

Arrows

Arrows Angle (2 - 10): 4

Array Segment Thickness (1 - 4): 2

Arrows Dimension (4 - 18): 8

Arrows Preview

	Main		Timed 100ms		Timed 5ms	
Segment Color:	⬛	Change	🟦	Change	🟥	Change
Arrows Color:	⬛	Change	🟦	Change	🟥	Change

Items

Label Selected Font: *Abcde* Change

Label Normal Color: ⬜ Change

Label Selected Color: 🟦 Change

Outputs Color: 🟥 Change

Inputs Color: 🟩 Change

I/O exported Color: 🟦 Change

☑ View I/O Labels

Save | Apply | Cancel | Defaults

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.

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.

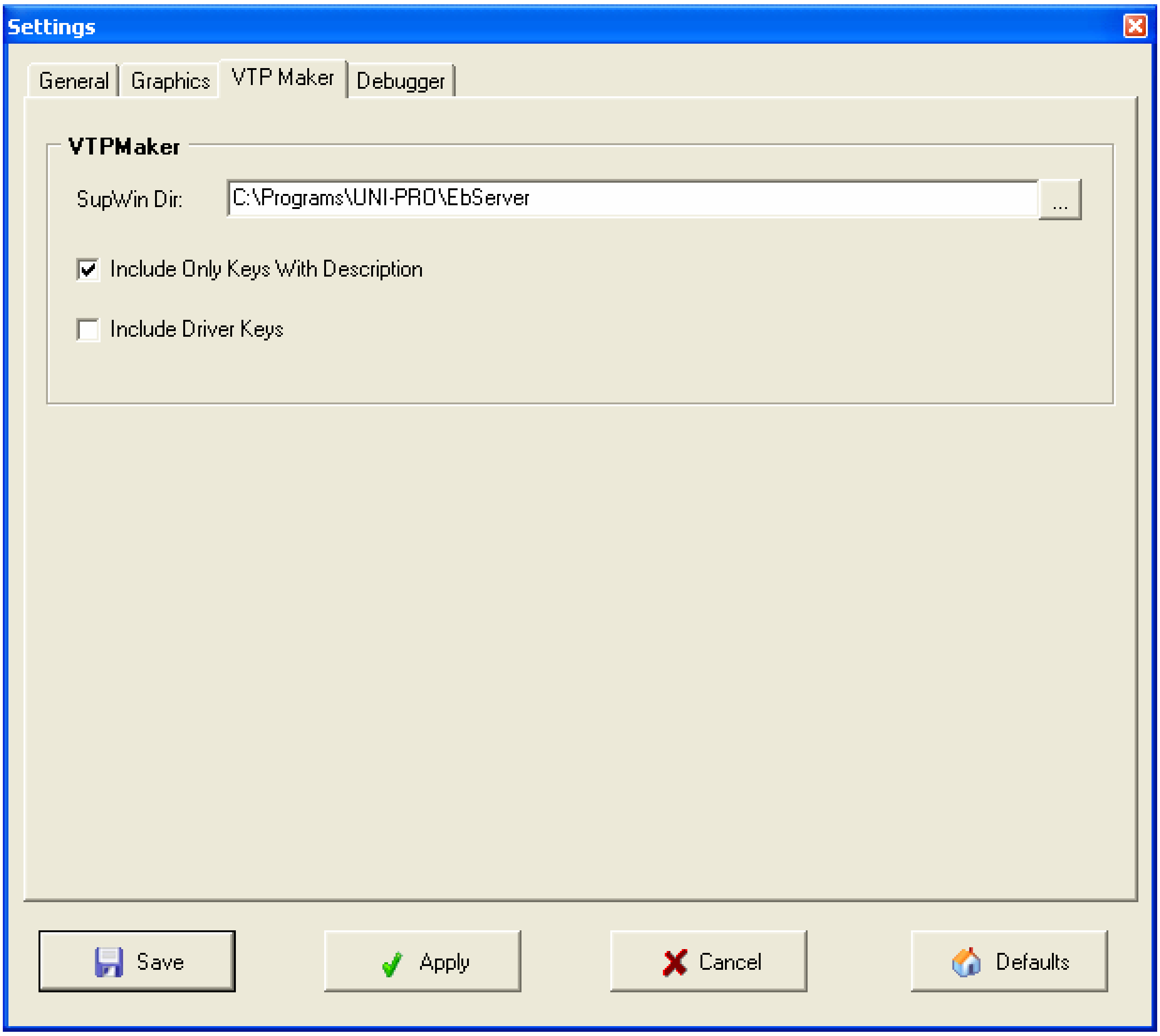
Array Segment thickness : Modifies the thickness of the lines connected with the array entity (entities with the array property set greater than one)

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, every 100 ms or 5 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.
Inputs color :	Defines the color to be attributed to the inputs of the entities.
Outputs color :	Defines the color to be attributed to the output of the 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.
View I/O Labels:	Show/Hide the label with the name of the clamp for the sheet, algorithm, library,…

The *VTP Maker* screen contains the configuration options used by VTP Maker tool to create VTP driver file required to communicate with EVCO software.



VTPMaker

SupWin Dir: C:\Programs\UNI-PRO\EbServer

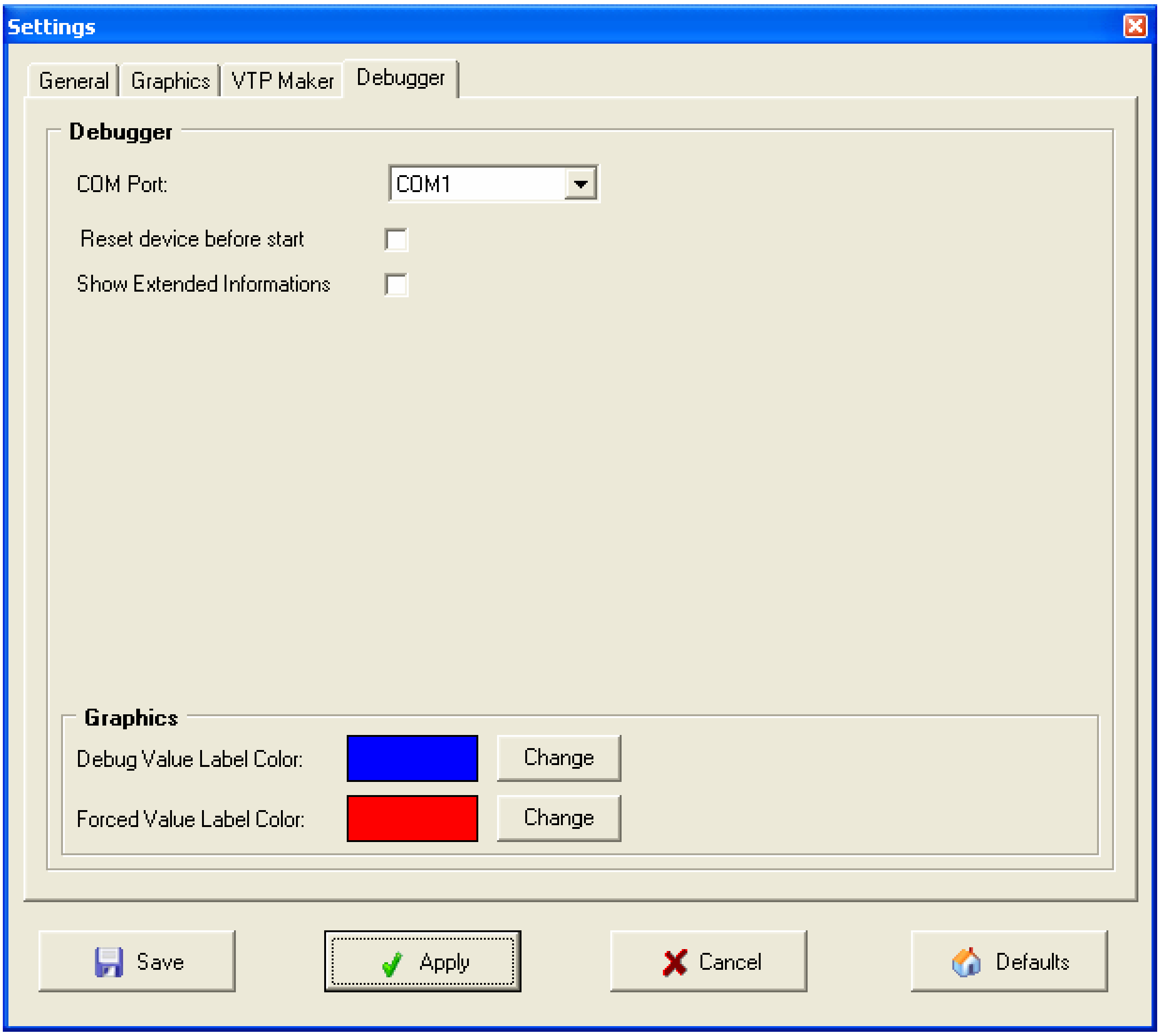
☑ Include Only Keys With Description

☐ Include Driver Keys

Options are the following:

SupWin Dir :	Defines the destination directory to the generated VTP Maker files. Default is: *..\UNI-PRO\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.



Debugger

COM Port: COM1

Reset device before start □

Show Extended Informations □

Graphics

Debug Value Label Color: 🟦 Change

Forced Value Label Color: 🟥 Change

Options are the following:

COM Port :	Serial port used to the comunication with the controller during debug operation
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
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)

4.11 Exporting / Importing

Using the import and export menus, it is possible to copy an EIML page from one project to another.

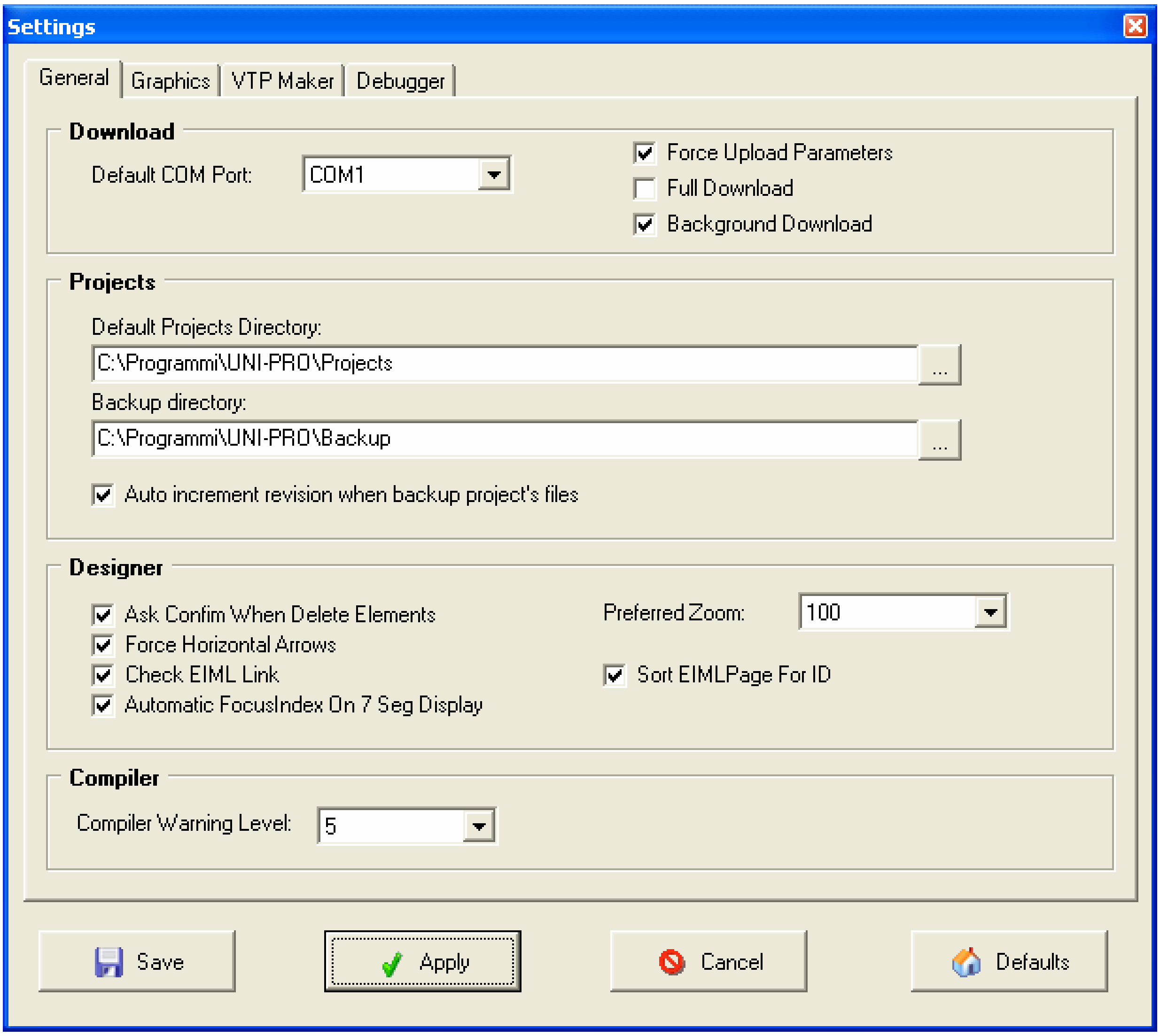
4.12 Project backup

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>



Settings

Download

Default COM Port: COM1

☑ Force Upload Parameters
□ Full Download
☑ Background Download

Projects

Default Projects Directory:
C:\Programmi\UNI-PRO\Projects

Backup directory:
C:\Programmi\UNI-PRO\Backup

☑ Auto increment revision when backup project's files

Designer

☑ Ask Confirm When Delete Elements
☑ Force Horizontal Arrows
☑ Check EIML Link
☑ Automatic FocusIndex On 7 Seg Display

Preferred Zoom: 100

☑ Sort EIMLPage For ID

Compiler

Compiler Warning Level: 5

4.13 Product programming tool (Download Manager)

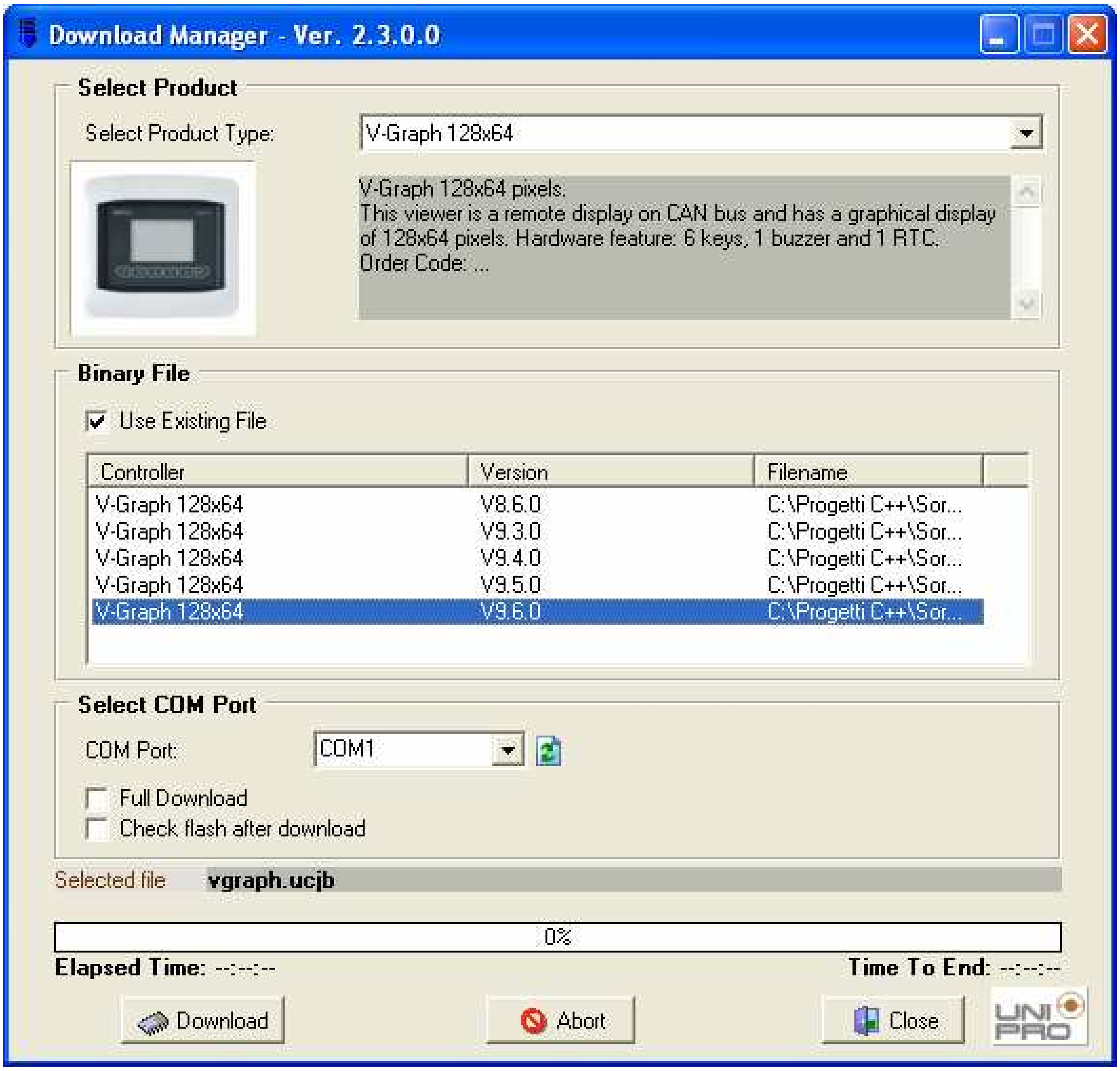
The development environment has a tool for programming and updating all the products (controllers and viewers). To view it, select the *Tools / Download products* menù.

To proceed with the programming of a product, it is necessary to select an item from the drop-down menu; a preview and a brief description of the product selected will be shown in the window below and, in the *Binary File* section, the list of the binary files present will be viewed. Removing the *Use Existing File* checkbox, it is instead possible to freely choose the file to be downloaded.

Before starting the download phase it is advisable to ensure that the system to be programmed is correctly connected to the serial port selected in the *Select COM Port* section. The *Full Download* option instead allows the complete downloading of the binary file if so desired.

To begin downloading, press the *Download* button. In the lower part of the window, the process status can be seen, in particular the Elapsed Time, the Time To End and percentage downloaded (in the progress bar) are shown.

If the download is not completed properly, check the connection with the device, the configurations used and re-attempt the operation.



Select Product Type: V-Graph 128x64

V-Graph 128x64 pixels.
This viewer is a remote display on CAN bus and has a graphical display of 128x64 pixels. Hardware feature: 6 keys, 1 buzzer and 1 RTC.
Order Code: ...

☑ Use Existing File

Controller	Version	Filename
V-Graph 128x64	V8.6.0	C:\Progetti C++\Sor...
V-Graph 128x64	V9.3.0	C:\Progetti C++\Sor...
V-Graph 128x64	V9.4.0	C:\Progetti C++\Sor...
V-Graph 128x64	V9.5.0	C:\Progetti C++\Sor...
V-Graph 128x64	V9.6.0	C:\Progetti C++\Sor...

COM Port: COM1

☐ Full Download
☐ Check flash after download

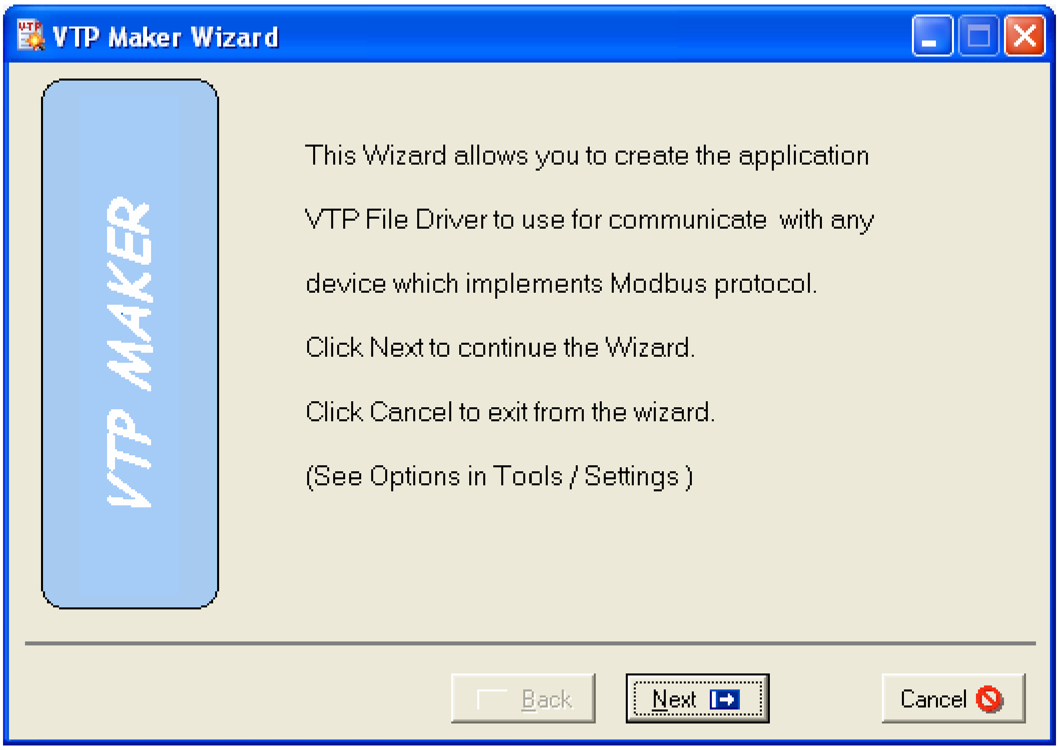
Selected file **vgraph.ucjb**

0%

Elapsed Time: --:--:-- **Time To End:** --:--:--

4.14 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.



This Wizard allows you to create the application

VTP File Driver to use for communicate with any

device which implements Modbus protocol.

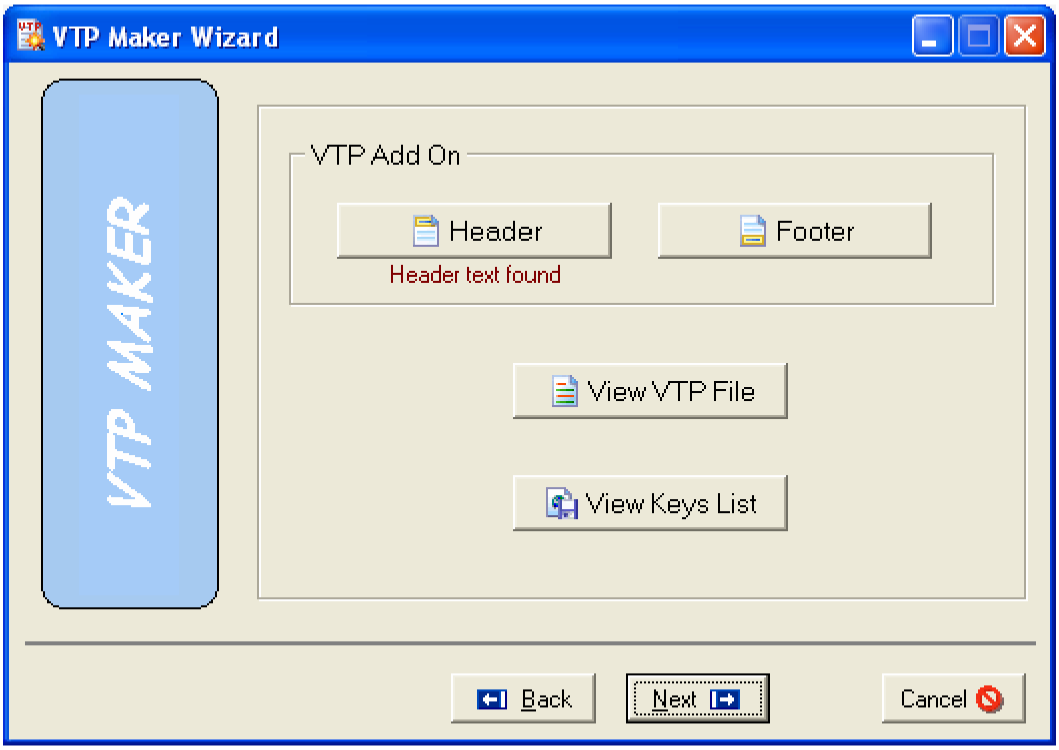
Click Next to continue the Wizard.

Click Cancel to exit from the wizard.

(See Options in Tools / Settings)

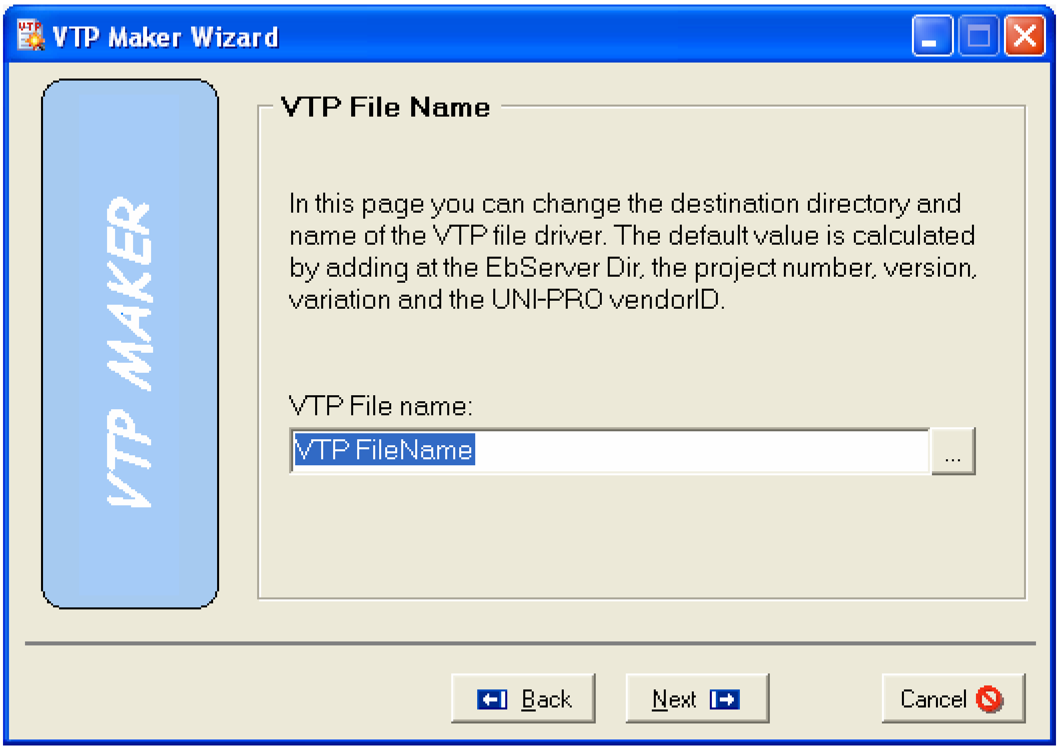
Before beginning the creation of VTP file driver, make sure that are correctly set the options for the automatic maker from *Settings / VTP Maker* menu.

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.

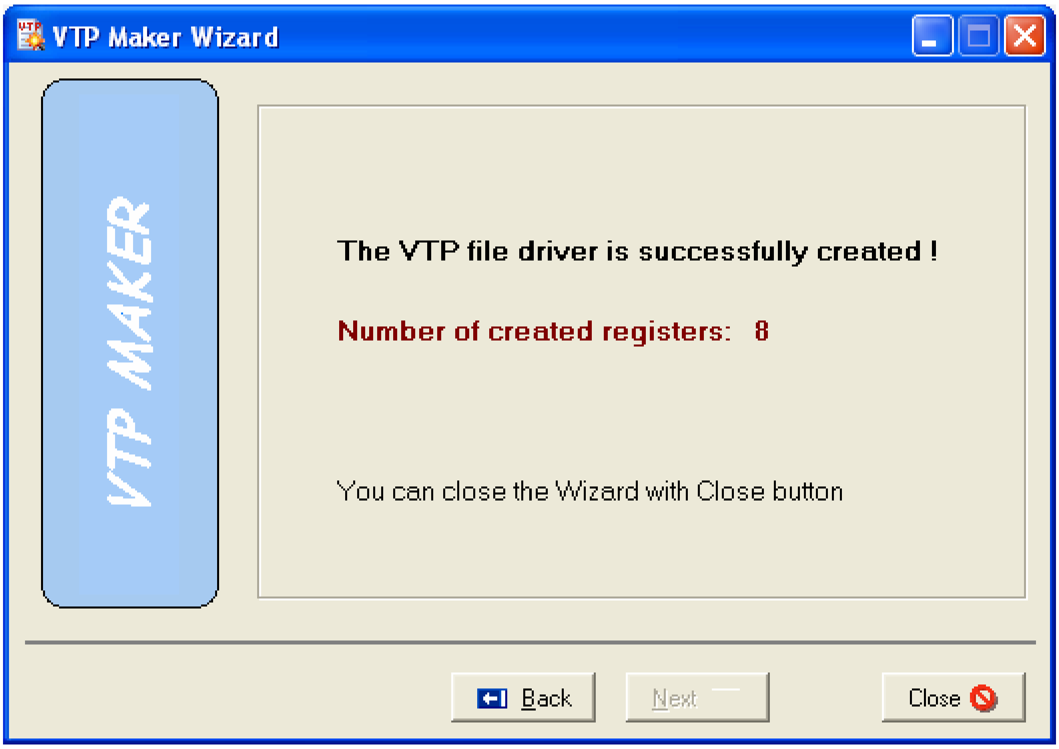


VTP File Name

In this page you can change the destination directory and name of the VTP file driver. The default value is calculated by adding at the EbServer Dir, the project number, version, variation and the UNI-PRO vendorID.

VTP File name: VTP FileName

Last showed window, less than errors during file generation, is that one indicated the successfully creation of VTP file driver.



The VTP file driver is successfully created !

Number of created registers: 8

You can close the Wizard with Close button

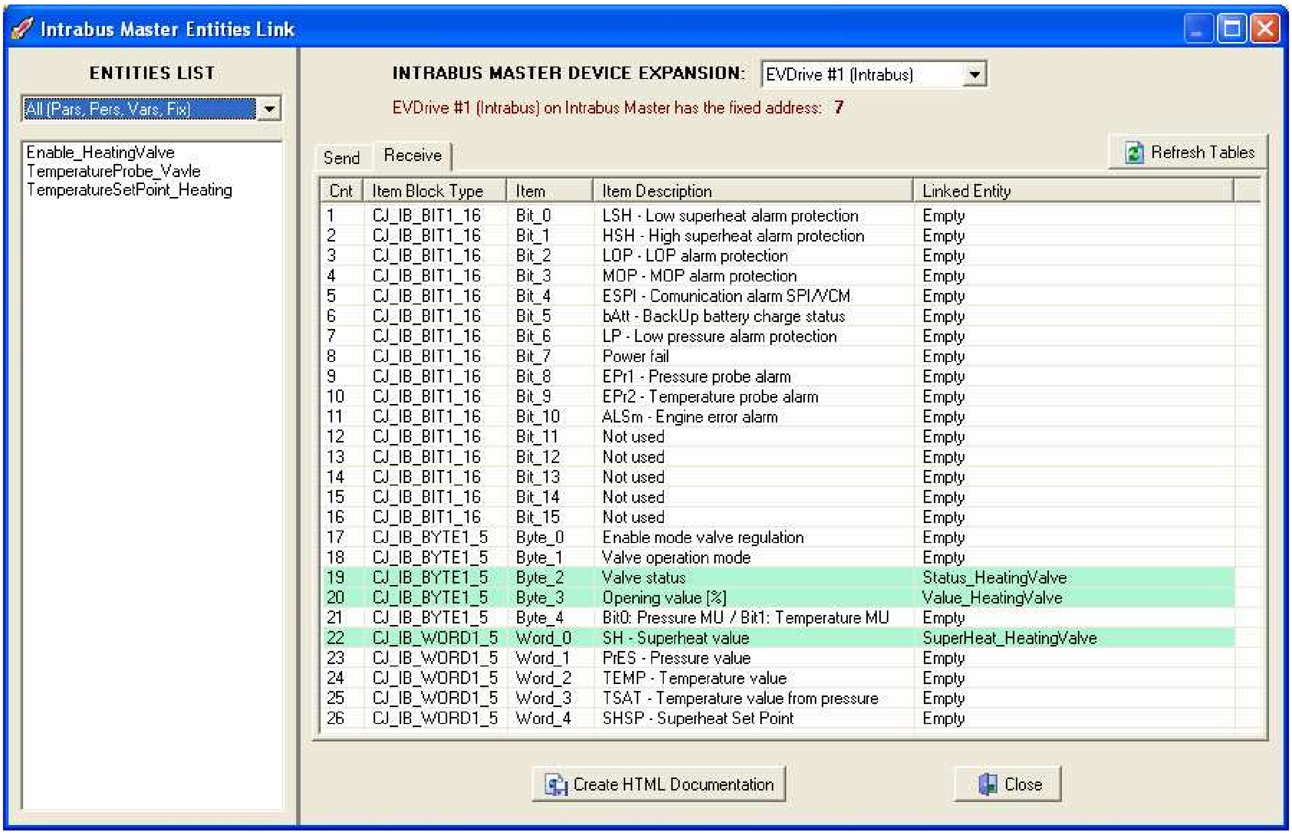
At any time it is possible to click button < *Back* to modify your selection. Click button *Close* to finish the VTP Make wizard.

4.15 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.

4.15.1 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:



All (Pars, Pers, Vars, Fix)

Enable_HeatingValve
TemperatureProbe_Vavle
TemperatureSetPoint_Heating

INTRABUS MASTER DEVICE EXPANSION: EVDrive #1 (Intrabus)

EVDrive #1 (Intrabus) on Intrabus Master has the fixed address: 7

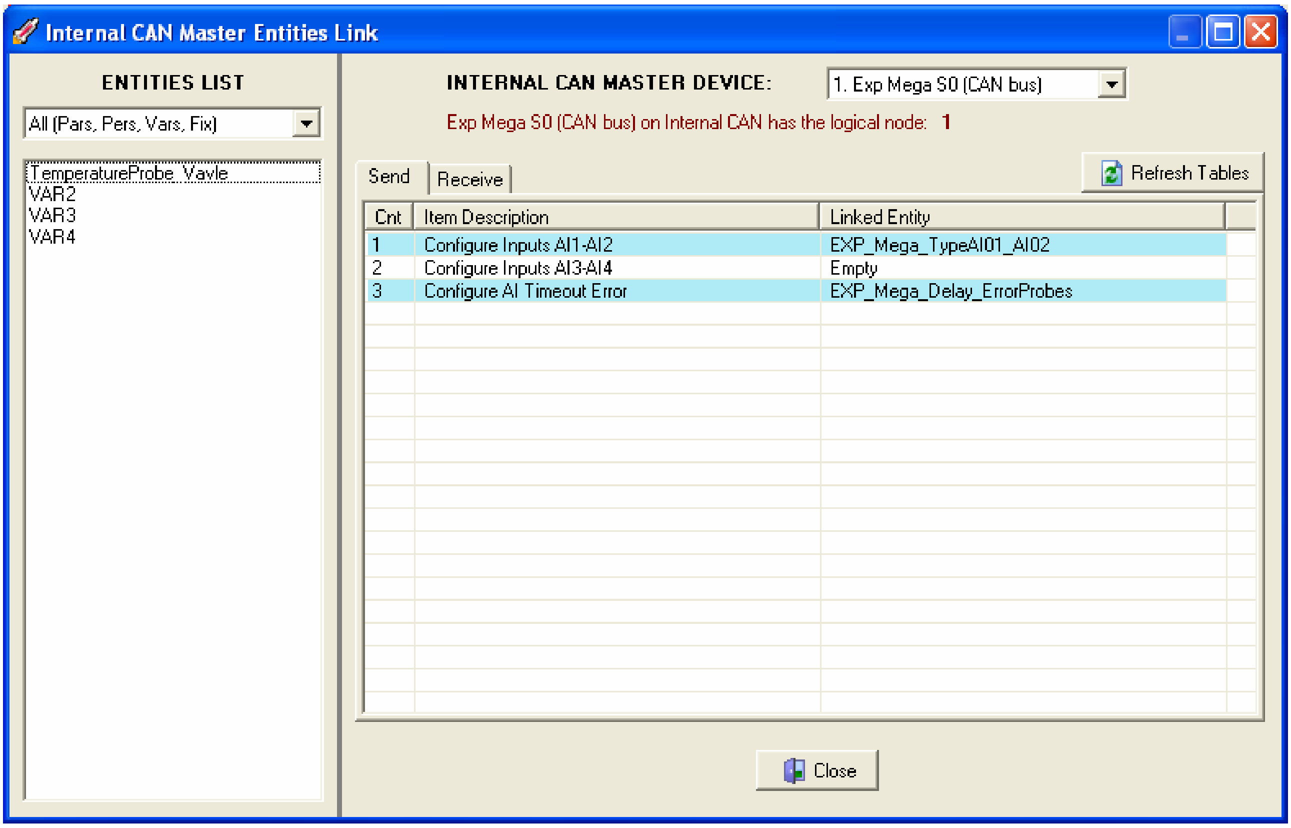
Cnt	Item Block Type	Item	Item Description	Linked Entity
1	CJ_IB_BIT1_16	Bit_0	LSH - Low superheat alarm protection	Empty
2	CJ_IB_BIT1_16	Bit_1	HSH - High superheat alarm protection	Empty
3	CJ_IB_BIT1_16	Bit_2	LOP - LOP alarm protection	Empty
4	CJ_IB_BIT1_16	Bit_3	MOP - MOP alarm protection	Empty
5	CJ_IB_BIT1_16	Bit_4	ESPI - Comunication alarm SPI/VCM	Empty
6	CJ_IB_BIT1_16	Bit_5	bAtt - BackUp battery charge status	Empty
7	CJ_IB_BIT1_16	Bit_6	LP - Low pressure alarm protection	Empty
8	CJ_IB_BIT1_16	Bit_7	Power fail	Empty
9	CJ_IB_BIT1_16	Bit_8	EPr1 - Pressure probe alarm	Empty
10	CJ_IB_BIT1_16	Bit_9	EPr2 - Temperature probe alarm	Empty
11	CJ_IB_BIT1_16	Bit_10	ALSm - Engine error alarm	Empty
12	CJ_IB_BIT1_16	Bit_11	Not used	Empty
13	CJ_IB_BIT1_16	Bit_12	Not used	Empty
14	CJ_IB_BIT1_16	Bit_13	Not used	Empty
15	CJ_IB_BIT1_16	Bit_14	Not used	Empty
16	CJ_IB_BIT1_16	Bit_15	Not used	Empty
17	CJ_IB_BYTE1_5	Byte_0	Enable mode valve regulation	Empty
18	CJ_IB_BYTE1_5	Byte_1	Valve operation mode	Empty
19	CJ_IB_BYTE1_5	Byte_2	Valve status	Status_HeatingValve
20	CJ_IB_BYTE1_5	Byte_3	Opening value [%]	Value_HeatingValve
21	CJ_IB_BYTE1_5	Byte_4	Bit0: Pressure MU / Bit1: Temperature MU	Empty
22	CJ_IB_WORD1_5	Word_0	SH - Superheat value	SuperHeat_HeatingValve
23	CJ_IB_WORD1_5	Word_1	PrES - Pressure value	Empty
24	CJ_IB_WORD1_5	Word_2	TEMP - Temperature value	Empty
25	CJ_IB_WORD1_5	Word_3	TSAT - Temperature value from pressure	Empty
26	CJ_IB_WORD1_5	Word_4	SHSP - Superheat Set Point	Empty

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.

4.15.2 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:



All (Pars, Pers, Vars, Fix)

TemperatureProbe_Vavle
VAR2
VAR3
VAR4

INTERNAL CAN MASTER DEVICE: 1. Exp Mega S0 (CAN bus)

Exp Mega S0 (CAN bus) on Internal CAN has the logical node: 1

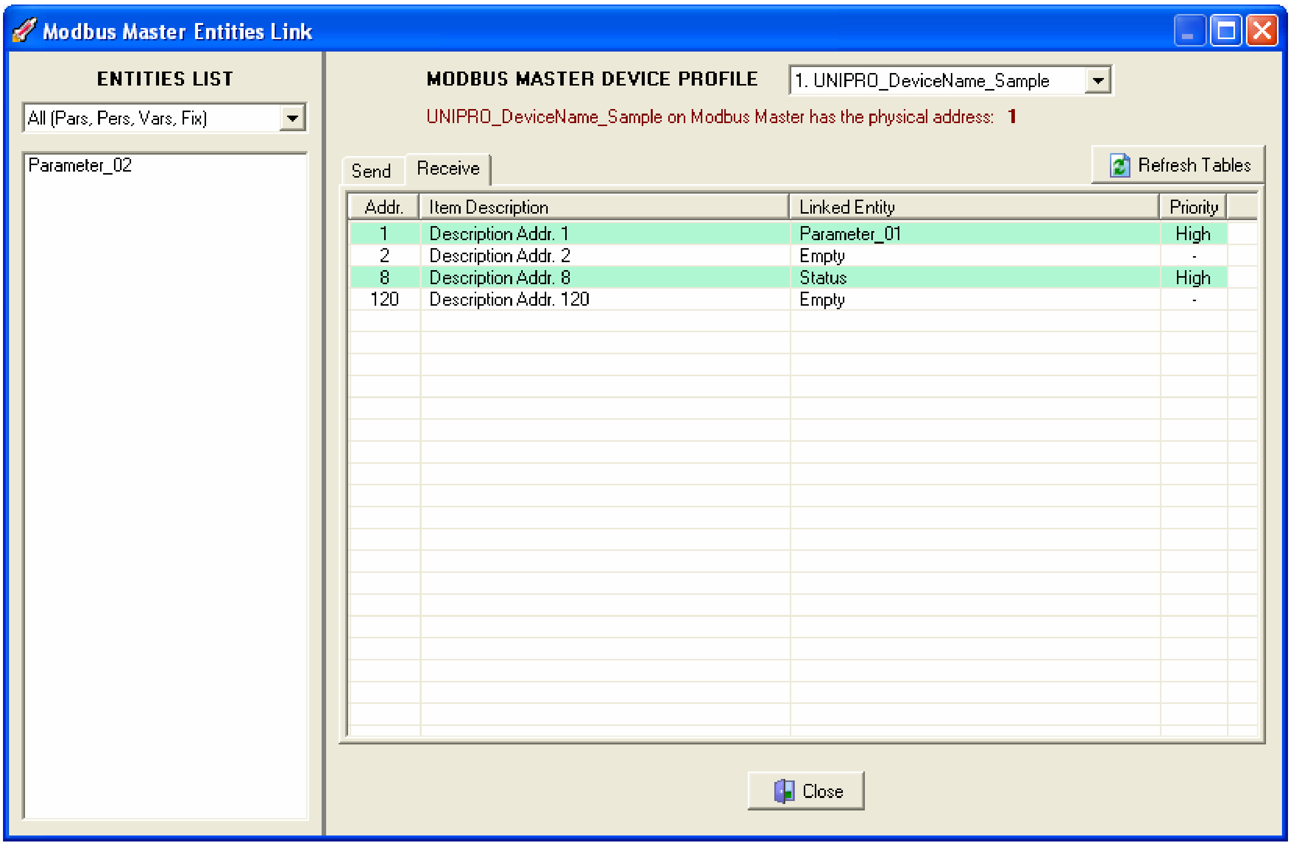
Cnt	Item Description	Linked Entity
1	Configure Inputs AI1-AI2	EXP_Mega_TypeAI01_AI02
2	Configure Inputs AI3-AI4	Empty
3	Configure AI Timeout Error	EXP_Mega_Delay_ErrorProbes

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.

4.16 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:


Modbus Master Entities Link
ENTITIES LIST
All (Pars, Pers, Vars, Fix)
Parameter_02
MODBUS MASTER DEVICE PROFILE
1. UNIPRO_DeviceName_Sample
Send
Receive
Refresh Tables
Close

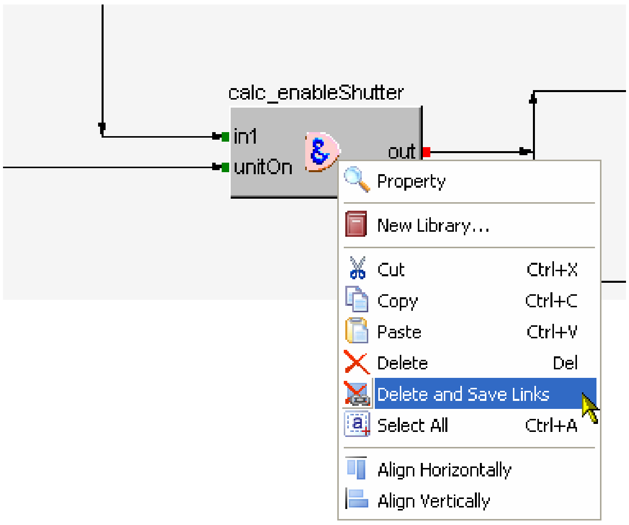
UNIPRO_DeviceName_Sample on Modbus Master has the physical address: 1

Addr.	Item Description	Linked Entity	Priority
1	Description Addr. 1	Parameter_01	High
2	Description Addr. 2	Empty	-
8	Description Addr. 8	Status	High
120	Description Addr. 120	Empty	-

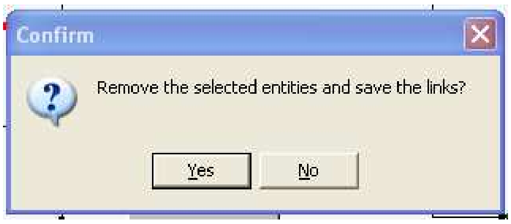
In the left-hand side of the window there are all the project variables, while in the right-hand side there are, for each configurable element in the network, two lists (selectable trough the relatives Tab): the list of the configurations to force as *Send* and the one of the configurations to read as *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.

4.17 "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.

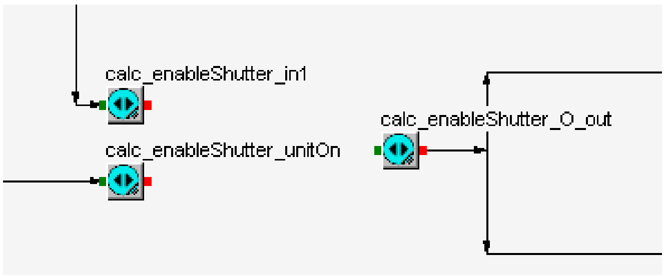


Click…


Confirm

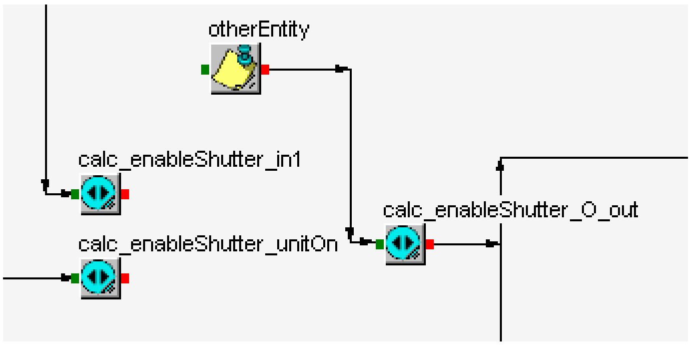
Remove the selected entities and save the links?

Click Yes ….

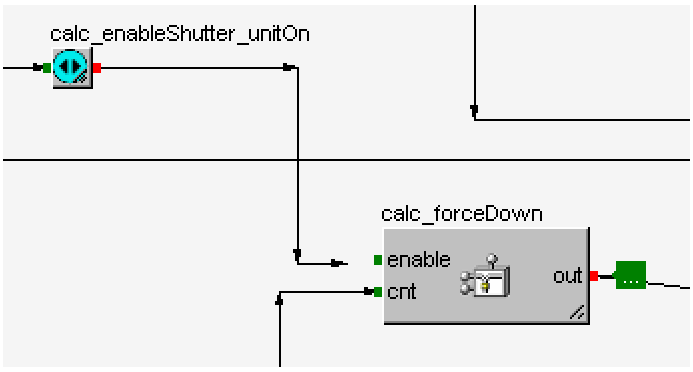


The entity *calc_enableShutter* has been removed and three variables were created (one for each input and output terminal) which are connected to the entity to which *calc_enableShutter* was connected. In this way the inputs/outputs have been "Secured" and can be elaborated and reused.

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.

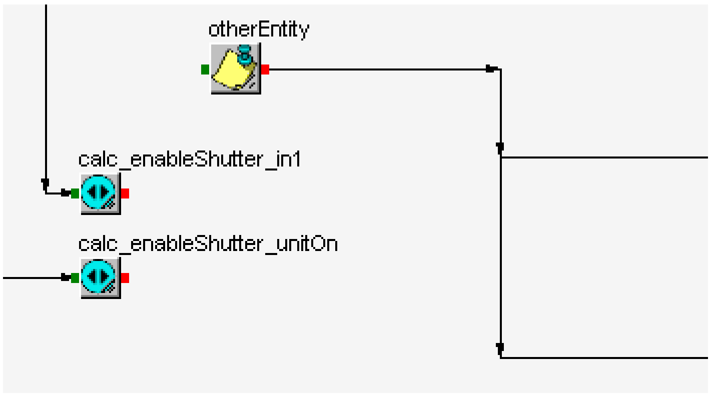


A)
Linking of *calc_enableShutter_O_out* with the new entity “otherEntity”

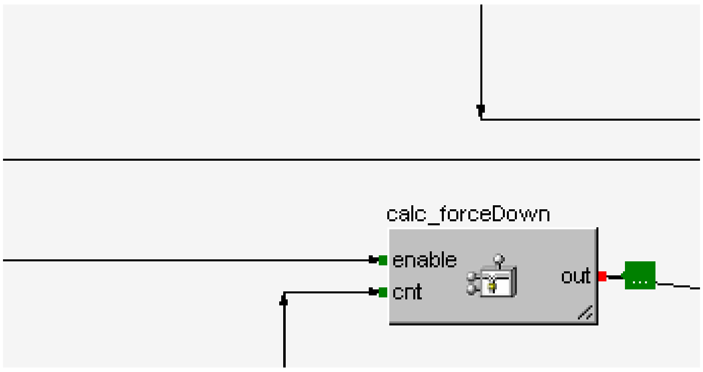


B)
Linking of *calc_enableShutter_unitON* to the “enable” input of the algorithm “calc_forceDown”

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

4.18 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 the 5ms task
The task under interrupt of the 5ms includes the following limitations:
- it does not work if the application is compiled in "debugger"
- The use of more entities calculated under this time base slows down the performance of the application and could impair other activities under the time base such as the serial communications

3) 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.

4.19 "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 cycles (under interrupt of 5ms and 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.

5 WHAT’S NEW IN VERSION 2.6.0.0

5.1 New functionality

Added CondVis property for conditional visibility of EIML elements in the pages.

Added MasterRefresh property for refresh speed of the entities in the ModbusMaster networks.

Improved Link Fixer entities to fix the link on entities deleting.

Improved *OK* and *Cancel* button on the Join Tools windows.

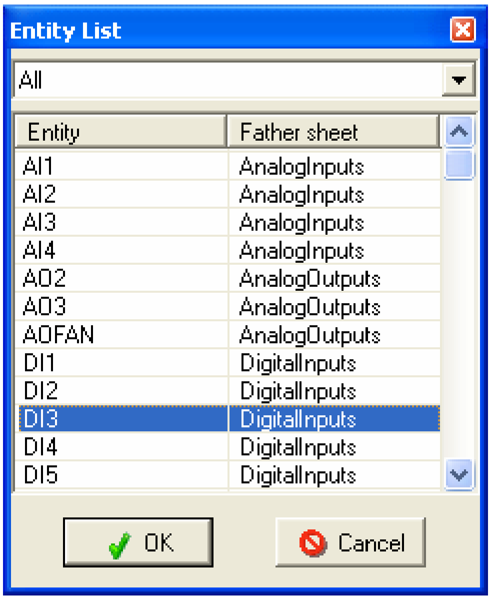
Improved *OK* and *Cancel* button on the Entity and EIML property windows to confirm or delete the modifies.

Added configuration and management for the Modbus Master network.

Modified management toolbar Hardware, Software ed EIML.

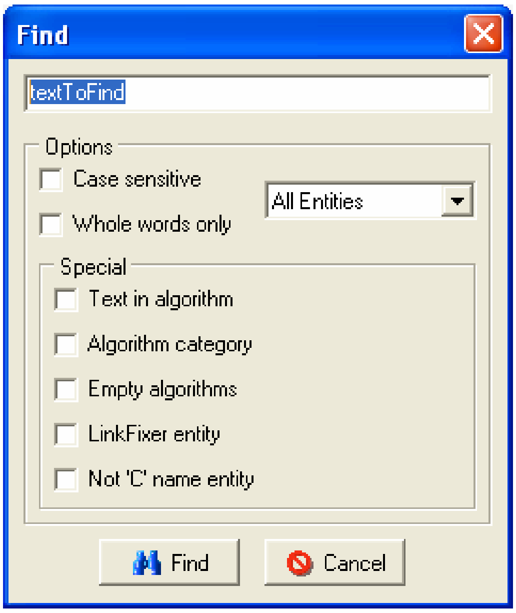
Modified libraries list with a tree view visualization; added *Category* concept, each library must have a Category (sub-folder) into the *Groups*.

Modified Entity List window, added a column with the father sheet of the listed entities.



Entity	Father sheet
AI1	AnalogInputs
AI2	AnalogInputs
AI3	AnalogInputs
AI4	AnalogInputs
AO2	AnalogOutputs
AO3	AnalogOutputs
AOFAN	AnalogOutputs
DI1	DigitalInputs
DI2	DigitalInputs
DI3	DigitalInputs
DI4	DigitalInputs
DI5	DigitalInputs

Improved new FIND function that allows to made more extensive search in code algorithm too.



Find

textToFind

Options

☐ Case sensitive

☐ Whole words only

All Entities

Special

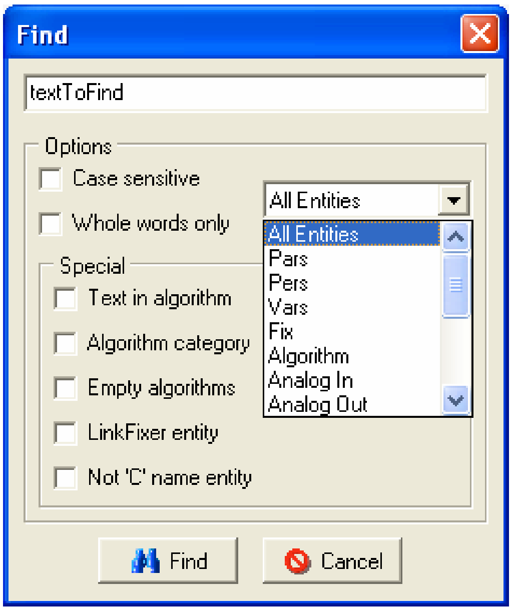
☐ Text in algorithm

☐ Algorithm category

☐ Empty algorithms

☐ LinkFixer entity

☐ Not 'C' name entity



Find

textToFind

Options

☐ Case sensitive

☐ Whole words only

All Entities

- All Entities
- Pars
- Pers
- Vars
- Fix
- Algorithm
- Analog In
- Analog Out

Special

☐ Text in algorithm

☐ Algorithm category

☐ Empty algorithms

☐ LinkFixer entity

☐ Not 'C' name entity

Text in algorithm	Search the text to find in the code of all the algorithms in the projects
Algorithm category	Search the text to find in the category of all the algorithms in the projects
Empty algorithms	Search all algorithms with empy text-code.
LinkFixer entity	Find all the LinkFixer entities in the projects
Not ‘C’ name entity	Search all entities that have got a name that not is in ‘C’ format, for example that contain son invalid character as “, ? - …

UNI-PRO – Software manual.
Version 2.6 - June 2010.
Code 114UPROSWE26.
File 114UPROSWE26.pdf.

Evco declines any responsibility regarding characteristics, technical data or any mistakes contained in this publication or consequential from usage of the same. Evco cannot be held responsible for any damages caused by non-compliance with warnings. Evco reserves the right to make any changes without previous notice and at any time, without prejudice to essential characteristics of functionality and safety.

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