SoHVAC

Software

User Guide

06/2014

EIO0000000537.03

www.schneider-electric.com

Schneider Electric

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Table of Contents

Safety Information 7

About the Book 8

1 General Information 10

1.1 About SoHVAC 10

1.1.1 Start-Up and Test 10

2 Developing Projects with SoHVAC 11

2.1 Quick Programming Example 11

3 Basic Operations 17

3.1 Creating a New Project 17

3.2 The Screen Organization 19

3.3 Entities: Definition and Use 20

3.3.1 Name of the Entity 20

3.3.2 Put an Entity on a Worksheet 20

3.3.3 Moving Entities 20

3.3.4 Removing Entities 21

3.3.5 Removing/Changing Entities and Saving Links 21

3.3.6 Cut and Paste of a Entity with EIML Connection 23

3.4 Control Sheets and Subsheets 24

3.4.1 Creating a New Sheet 24

3.4.2 Creating a New Subsheet 24

3.5 Navigation in Sheets 24

3.6 Linking Entities 25

3.6.1 Link Entities 25

3.6.2 External Links 25

3.6.3 Link Out 26

3.6.4 Link In 26

3.6.5 Show Link Context Menu 26

3.7 Creating EIML Pages 28

3.8 Selecting Hardware 29

3.9 Connection to Physical Terminals 36

3.9.1 Digital Inputs 36

3.9.2 Digital Outputs 37

3.9.3 Analog Inputs 38

3.9.4 Analog Outputs 39

3.9.5 Clocks 40

3.9.6 LEDs 41

3.9.7 Buttons 42
3.9.8 Command In 43
3.9.9 Command Out 44
3.9.10 Buzzers 45

3.10 Compiling the Program 46

3.11 Downloading the Program 48
3.11.1 COM Port Configuration 48

3.12 Use of the Debugger 53
3.12.1 Start Debug 54
3.12.2 Run/GoTo/Break/Reset 55
3.12.3 Set Value 57
3.12.4 Key-in Present Value 57
3.12.5 Breakpoints 58
3.12.6 Limitation during Debug 59

3.13 Algorithm Simulator 60
3.13.1 Feedback Output to an Input 63
3.13.2 Absolute Breakpoint 66
3.13.3 Conditional Breakpoint 68
3.13.4 Evaluate 71

4 Basic Elements Data Types 72

4.1 Control Components 72
4.1.1 Software 72
4.1.2 Hardware 84

4.2 Project Template Elements 95
4.2.1 Hardware Bits 95
4.2.2 Software Bits 96
4.2.3 ExpBUS Network 98
4.2.4 Analog Inputs 99

4.3 Simple Data Types 100
4.3.1 CJ_VOID 100
4.3.2 CJ_LED and CJ_BUZZ 101
4.3.3 CJ_DATE 101
4.3.4 CJ_TIME 101
4.3.5 CJ_DATETIME 101

4.4 Structured Data Types 102
4.4.1 CJ_ANALOG 102
4.4.2 CJ_CMD 102
4.4.3 CJ_BTN 102
4.4.4 CJ_DATE_STRUCT 103
4.4.5 CJ_TIME_STRUCT 103
4.4.6 CJ_DATE_TIME_STRUCT 104

4.5 Type Compatibility Table 105

4.6 Elements in EIML Pages 107
4.6.1 Toolbar 107
4.6.2 Embedding an Element in an EIML Page 107
4.6.3 Page Properties 108
4.6.4 Element Properties 109
4.6.5 Text A 109
4.6.6 Variable V 111

4.6.7 Strings 114
4.6.8 Icons 116
4.6.9 Combo 118
4.6.10 Table 120
4.6.11 Line 124
4.6.12 Rectangle 124

5 Advanced Operations 125

5.1 Project Settings 125
5.1.1 Version Info 125
5.1.2 Historian 126
5.1.3 Project Defines 127
5.1.4 Project Options 127
5.1.5 Project Change Log 129

5.2 Preference Menu: Configuration 130
5.2.1 General 130
5.2.2 Graphics 132
5.2.3 Debug Utility 134

5.3 Considerations in Project Development 135

5.4 Project Backup 136

5.5 Product Programming Tool (Download Manager) 137
5.5.1 Local Download 138

5.6 EIML Pages Advanced Topics 139
5.6.1 Navigating through the Pages 139
5.6.2 Password-Protected Information 140
5.6.3 Simulating EIML Pages 141
5.6.4 Conditional Visibility 144
5.6.5 Sorting Embedded Interface Markup Language (EIML) Elements 147
5.6.6 EIML Pages Import and Export 147

5.7 Commissioning on Expansion Bus (ExpBUS) 148
5.7.1 ExpBUS Node and Physical Address 148
5.7.2 Factory Configuration 149
5.7.3 ExpBUS Master…Window 149
5.7.4 Creating Networks 150
5.7.5 Configuration of One Expansion and One Display 150
5.7.6 Configuring Multiple Expansions 153
5.7.7 Configuring Data Exchanges between Controllers 157
5.7.8 Configuring One Remote Display with Multiple Controllers 159
5.7.9 Expansion Bus (ExpBus) Configuration 160
5.7.10 ExpBUS Master 161

5.8 Modbus Communication 162
5.8.1 Exporting Variables 162
5.8.2 Modbus Master 166

5.9 Using Array Entities 169
5.9.1 Array Property 170
5.9.2 Value Property 170
5.9.3 Min, Max, Precision Property 171
5.9.4 Compatibility and Link 171

5.10 Type CJ_CHAR for String Management 172

5.11 Using C Algorithm Editor 174
5.11.1 Input and Output Properties 174
5.11.2 Algorithm Category 175
5.11.3 Built-in Editor 176
5.11.4 Algorithm with Array Inputs/Outputs 177
5.11.5 Possibility to Change the Order of Input Terminals of the Algorithms Added 179
5.11.6 Operating Limits 179
5.11.7 Global Algorithms 179

5.12 Creating Libraries 182
5.12.1 General Data 182
5.12.2 Options 183
5.12.3 Additional Data 183
5.12.4 Type 183

5.13 Commands 183
Commands Window 184
5.13.1 System Command Description 185
5.13.2 Create a Command 187

5.14 Defining Execution Tasks 187

5.15 Setting the Execution Order of a Program 188
5.15.1 Order Property 190
5.15.2 Offset Property 191

6 Controller Operation 192

6.1 Cycle Time IO Management 192

6.2 Tasks 193

6.3 State Machine 194
6.3.1 Power OFF 195
6.3.2 Program State 195
6.3.3 BOOT State 195
6.3.4 Valid Application 195
6.3.5 RUN State 195
6.3.6 Watchdog and Applicative Errors 195
6.3.7 HALTED (only in DEBUG MODE) 195
6.3.8 Power On State 196
6.3.9 Watchdog Behavior 196

Appendix - Compilation Error Message Format 197

Glossary 198

Index 203

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

⚡ The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

⚠ This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, **can result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a potentially hazardous situation which, if not avoided, **can result in** minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book

At a Glance

Document Scope

This manual describes about the development environment of SoHVAC for Windows® that is used to create and manage distributed control projects. This manual gives you the easy-to-use program to familiarize with the SoHVAC development environment.

Validity Note

This document is valid for SoHVAC V3.0.

Related Documents

Title of Documentation	Reference Number
SoHVAC C Programming Language User Guide	EIO0000000536
SoHVAC Standard Library User Guide	EIO0000000538
Modicon M168 Controller Hardware Guide	EIO0000000533

You can download these technical publications and other technical information from our website at www.schneider-electric.com.

Product Related Information

⚠ WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.[1]
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

1 General Information

1.1 About SoHVAC

SoHVAC for Windows® is an integrated development environment for the M168, a programmable controller that enables you to customize and manage complex heating, ventilation and air conditioning applications.

It enables you to set up in an easy and user-friendly way a modular range of hardware.

The development environment is composed of a series of integrated and interacting tools:

- Creation and management of objects and control algorithms.
- Creation and architecture of user interfaces.
- Configuration and management of communication networks.
- Debug mode
- Creation of your own object libraries.

1.1.1 Start-Up and Test

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check be made and that enough time is allowed to perform complete and satisfactory testing.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

Failure to follow these instructions can result in injury or equipment damage.

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Test your application in both simulated test and real environments.

Verify that the completed system is free from all short circuits and unintended ground connections, except those grounds installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove ground from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

2 Developing Projects with SoHVAC

You can create and manage distributed control projects through the SoHVAC development environment.

The possibility of creating object libraries enables you to build your own collection of objects that you can reuse in multiple projects, minimizing development time.

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

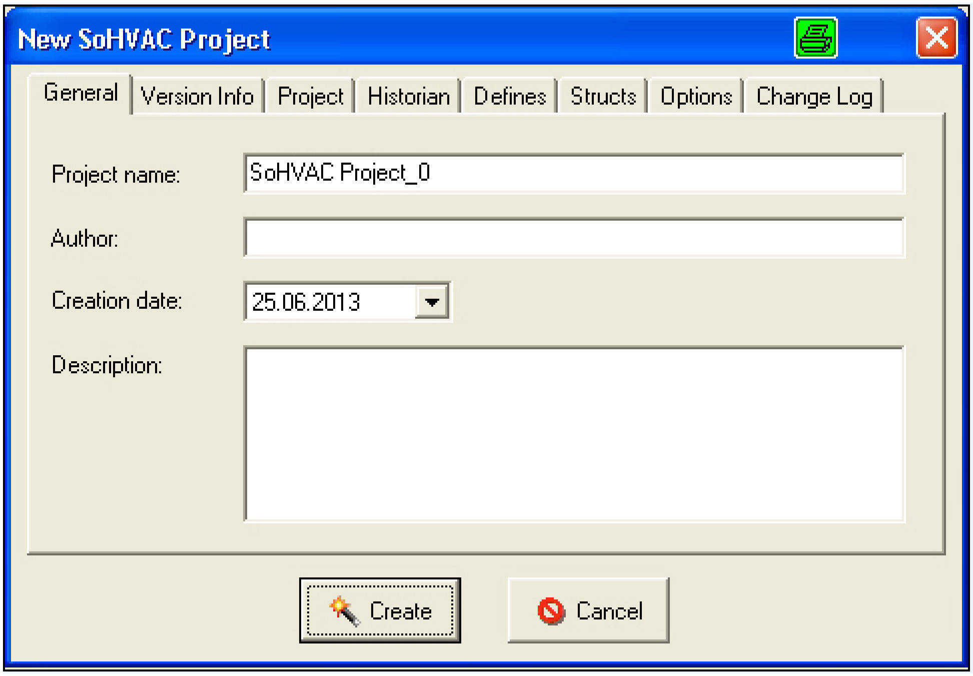
2.1 Quick Programming Example

Below is an easy program to help you familiarize with the SoHVAC development environment. The program uses a Digital In and a Digital Out that will be joined together. The effect you want to achieve is that when the digital input switches, the corresponding digital output switches too.

In addition, you can also create an EIML (Embedded Interface Markup Language) page in which the digital output is linked to a combo object with 2 texts, so that a different message is displayed according to output state.

To create a new project, you can either select **File→New Project** from the menu or click the **Create a new project** icon in the toolbar.

A window called **New SoHVAC Project** appears. Enter the name Sample1 in the **Project name** box and click **Create**.



Project name: SoHVAC Project_0

Author:

Creation date: 25.06.2013

Description:

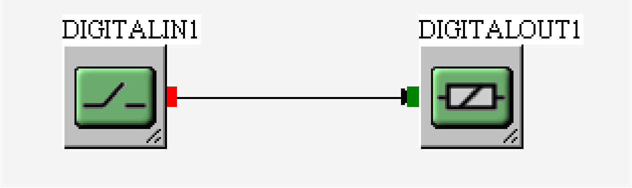
Initially, you need a digital type input and a digital type output. From the Hardware library, you can select a Digital In and drag-and-drop it anywhere on the worksheet. The digital input icon with the default name DIGITALIN1 appears. You can drag it to the desired position.

You can follow a similar procedure to position a Digital Out as well.

Once both entities have been added to the main worksheet of the application, you will need to link the entities.

To link the entities, place the mouse pointer over the output pin of the digital input entity and make a left mouse click. Move the mouse to the input pin of the digital output entity. During the mouse movement, you will notice that an arrow is drawn from the output pin of the digital input entity. Make a left mouse click again when the mouse reaches the input pin of the digital output entity.

If you have followed all the steps in this procedure correctly, both entities should now be linked by an arrow.

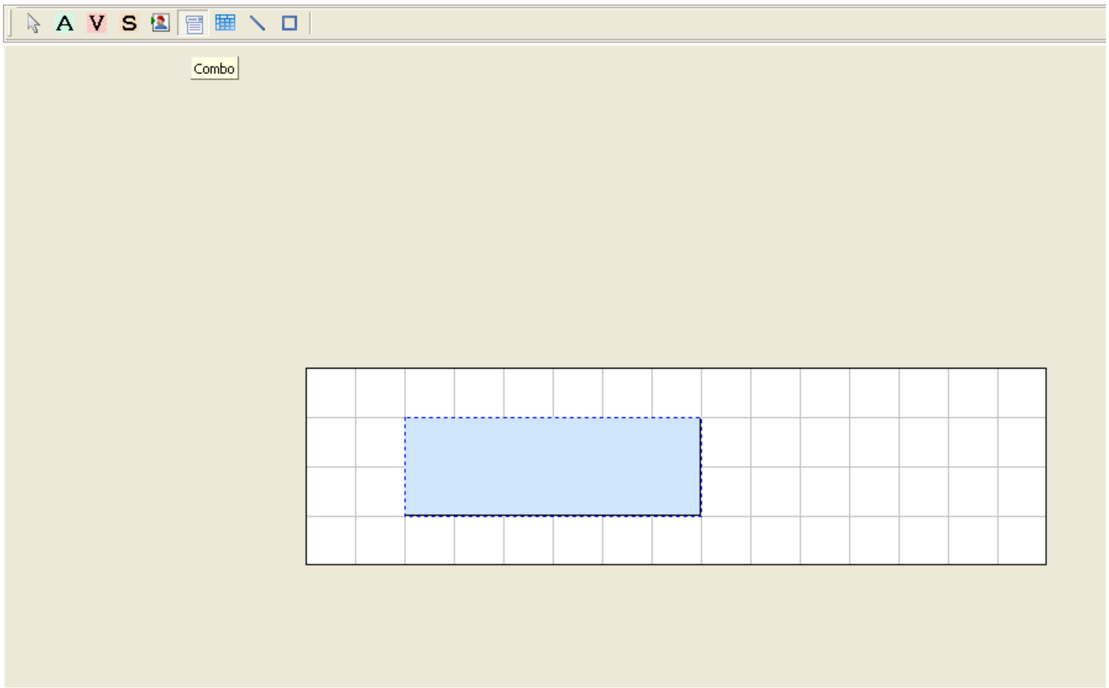


To save the project, select **File** →**Save Project** from the menu, or click the corresponding icon in the toolbar.

To create the graphical interface, select **File** → **New** → **New EIML Page (120 x 32)** from the main menu.

A blank page appears corresponding to a 120 x 32 pixel LCD graphic display.

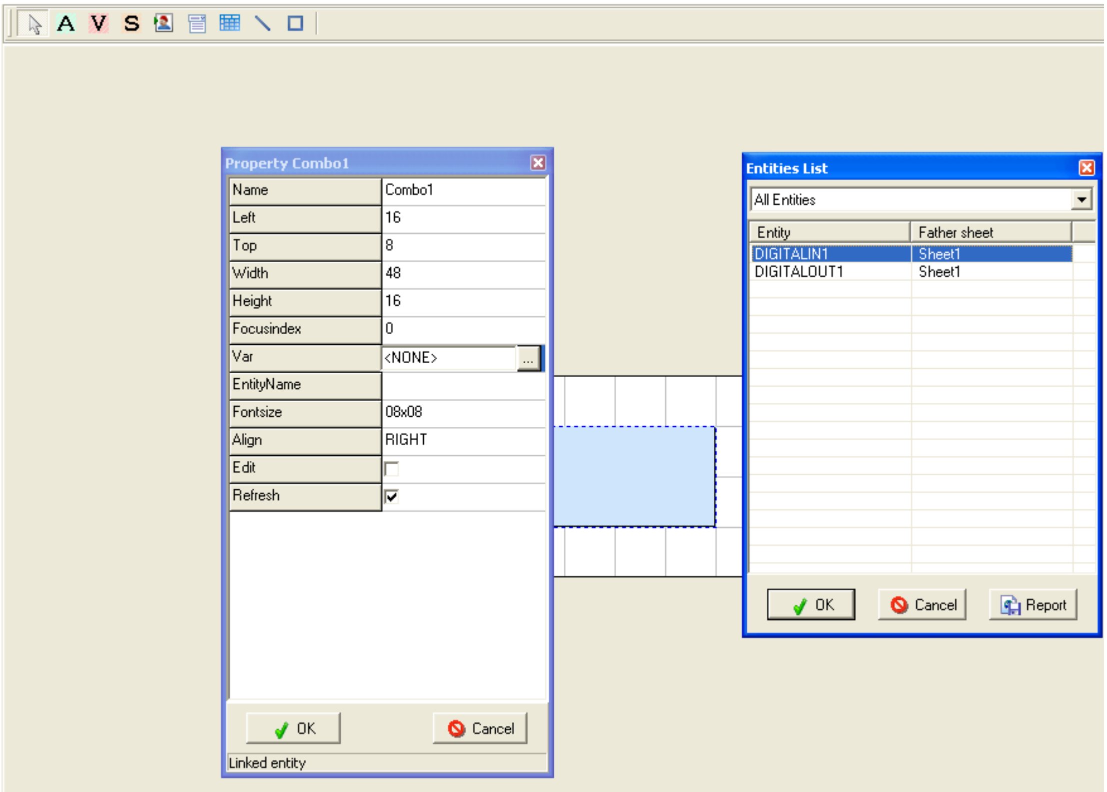
From the toolbar, select the **Combo** icon, click the blank page and drag it while holding the left mouse button down until you obtain a rectangle.



Once you have created it, you can move it to the desired position. If you wish to change its size, you need to open the properties window (right-click the rectangle and select Properties) and set Left, Top, Width and Height as required.

Otherwise, you can graphically resize the object by placing the mouse pointer at the right end or bottom end of the element.

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



Property Combo1

Name	Combo1
Left	16
Top	8
Width	48
Height	16
Focusindex	0
Var	<NONE>
EntityName	
Fontsize	08x08
Align	RIGHT
Edit	□
Refresh	☑

Linked entity

Entities List

All Entities

Entity	Father sheet
DIGITALIN1	Sheet1
DIGITALOUT1	Sheet1

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

Finally, set Align to Center, Font to 8x16 and check the **Refresh** property.

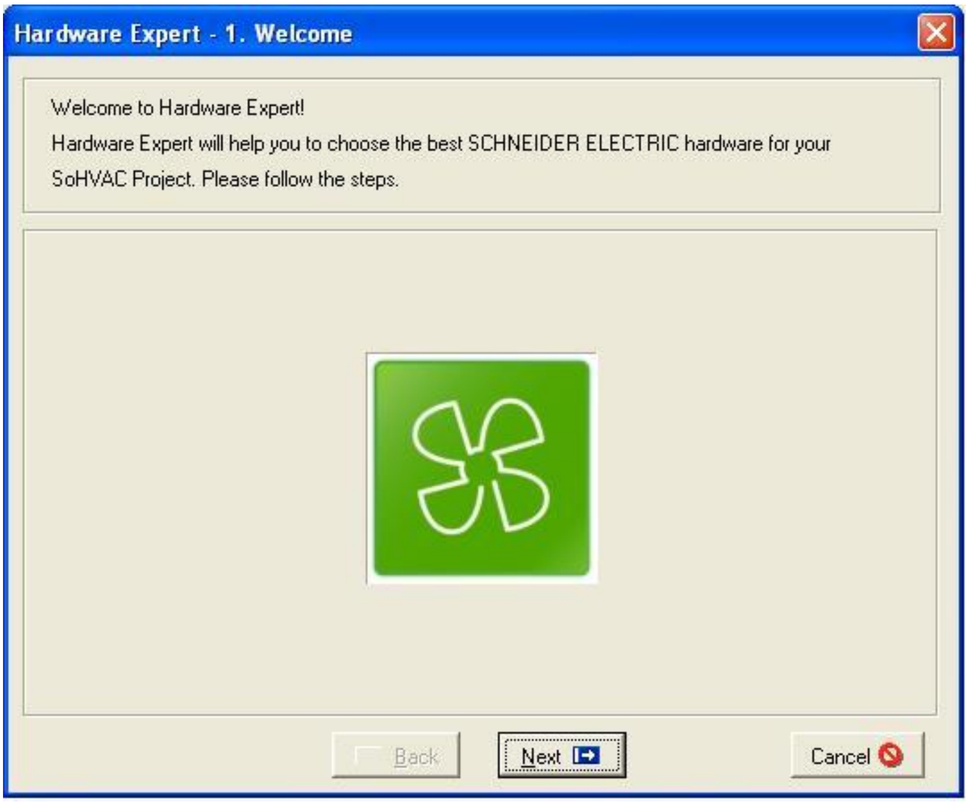
Now, right-click the rectangle and select the **Combo** wizard. Leave the **Type** property set to Text and type Switch-On in the Name window, and then click **Add**.

In this way, this text will remain linked to the value 0 of the DigitalOut variable (contact open). Now repeat the operation but this time link the text Hello World! to the value 1 (contact closed).

NOTE: In the **Combo** wizard list created, the first element from the top is linked to value 0, the second to value 1 and so on.

Launch the compilation by selecting **Project→Compile** from the menu or click the corresponding key in the toolbar.

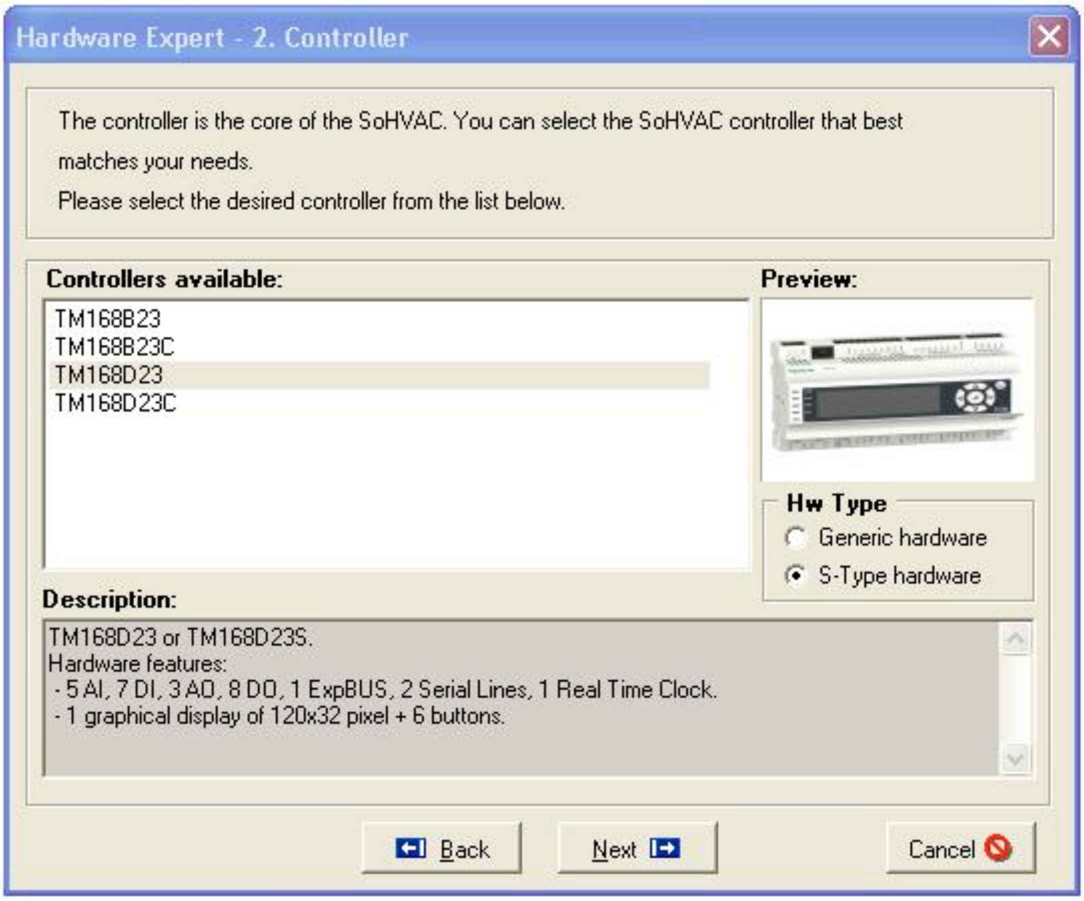
If you have not yet selected the hardware on which the project runs, the **Hardware Expert** appears.



Welcome to Hardware Expert!

Hardware Expert will help you to choose the best SCHNEIDER ELECTRIC hardware for your SoHVAC Project. Please follow the steps.

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



Hardware Expert - 2. Controller

The controller is the core of the SoHVAC. You can select the SoHVAC controller that best matches your needs.
Please select the desired controller from the list below.

Controllers available:

- TM168B23
- TM168B23C
- TM168D23
- TM168D23C

Preview:

Hw Type
○ Generic hardware
● S-Type hardware

Description:

TM168D23 or TM168D23S.
Hardware features:
- 5 AI, 7 DI, 3 AO, 8 DO, 1 ExpBUS, 2 Serial Lines, 1 Real Time Clock.
- 1 graphical display of 120x32 pixel + 6 buttons.

Select the controller (for example, TM168D23) and click **Next**.

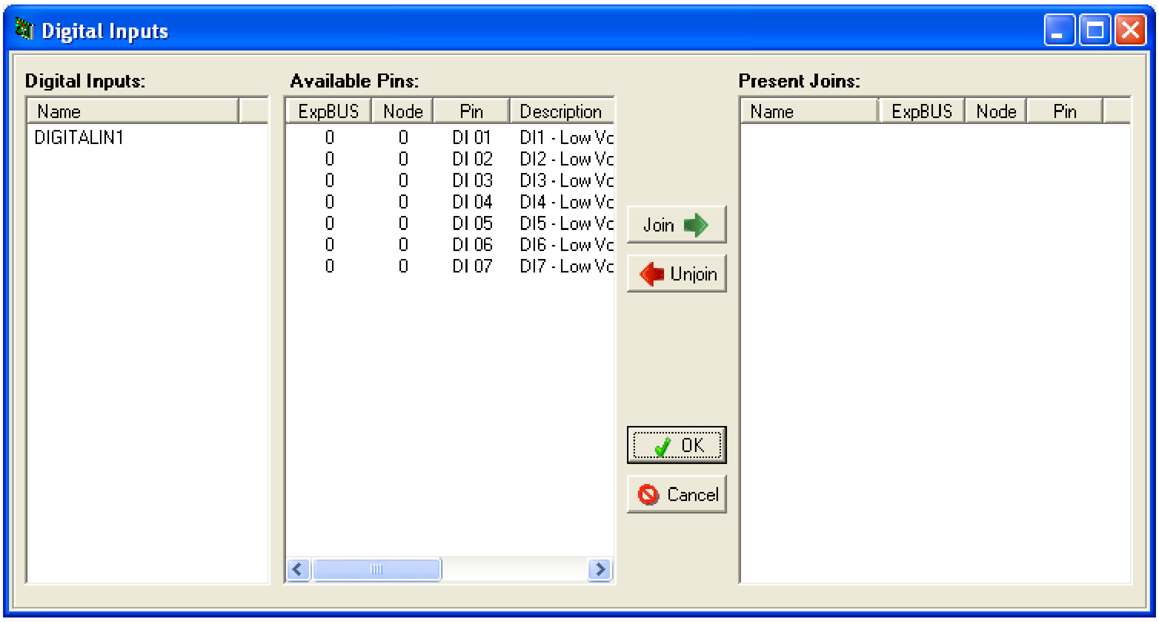
The third wizard step shows the serial port configurations (leave them out at present).

The fourth wizard step shows the available expansions with a brief description on the right, and gives the possibility of joining them to the ExpBUS. You need not add any expansions, click **Next**.

In the fifth step, you can join a display (select the TM168GDB graphic display with 128 x 64 resolutions).

The last step of the **Hardware Expert** summarizes the selected configuration. By clicking the End, the wizard closes and automatically goes to a window where you can verify the joins between the entities and the physical terminals.

In this case, you need to join the logical **DigitalIn** and the **DigitalOut** to a physical digital input and digital output, respectively.



Digital Inputs

Digital Inputs:

Name
DIGITALIN1

Available Pins:

ExpBUS	Node	Pin	Description
0	0	DI 01	DI1 - Low Vc
0	0	DI 02	DI2 - Low Vc
0	0	DI 03	DI3 - Low Vc
0	0	DI 04	DI4 - Low Vc
0	0	DI 05	DI5 - Low Vc
0	0	DI 06	DI6 - Low Vc
0	0	DI 07	DI7 - Low Vc

Present Joins:

Name	ExpBUS	Node	Pin

Select **DIGITALIN** from the list on the left and the physical terminal you wish to join it to from the center list, for example, Pin **DI 01**.

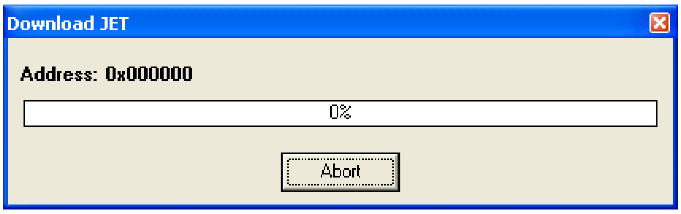
By clicking the Join, the selected items are removed and a new row is generated in the list on the right which describes the join that has been made.

When closing this window, the system automatically requests you to join the DigitalOut as well. To join the **DIGITALOUT**, proceed in the same way as you did in the previous example.

Once you have joined all entities correctly in the Hardware section, the project compilation starts (the result of compilation is displayed in the output window at the bottom of the screen).

Now you can download the program by selecting the **Project**→**Download** menu (or click the corresponding toolbar icon). For more connection information, refer to the *Modicon M168 Controller Hardware Guide*.

The following window appears:



Download JET

Address: 0x000000

0%

If you have selected a **TM168●23●●** controller in the hardware expert wizard, use the USB - RJ11 programming cable and select the correct COM port in the **Tools** →**Preference** menu.

If the connection with the controller is successfully established, after a few seconds the progress bar begins to expand. Once the download is complete, a box appears indicating the successful outcome of the operation. Otherwise an error message appears. In this case, see chapter COM port configuration.

Having terminated the download operations, your first program should have been loaded in the TM168.

If you close the contact where the DigitalIn has been connected, the relay corresponding to the DigitalOut should be switched ON and the message Hello World! should appear on controller built-in display and the remote graphic display.

3 Basic Operations

This chapter describes the basic operations that allow you to develop a project using the SoHVAC development environment.

The user is guided through the operations required to develop a project (creation, algorithms implementation, compilation and download to the controller).

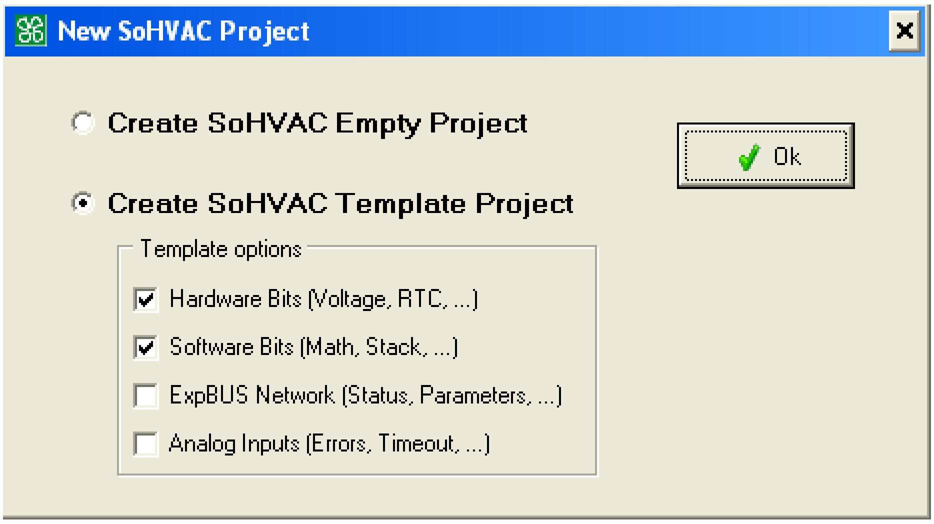
3.1 Creating a New Project

The first thing you need to do in SoHVAC is to create a new project.

Select **File**→**New Project** from the menu, or click the **Create New Project** icon in the toolbar.

A window will open to ask the type of project to be created:

- **Create SoHVAC Empty project**: creates a project without function blocks
- **Create SoHVAC Template Project**: creates a project including Alarms or Status Blocks



○ Create SoHVAC Empty Project

● Create SoHVAC Template Project

Template options

☑ Hardware Bits (Voltage, RTC, ...)

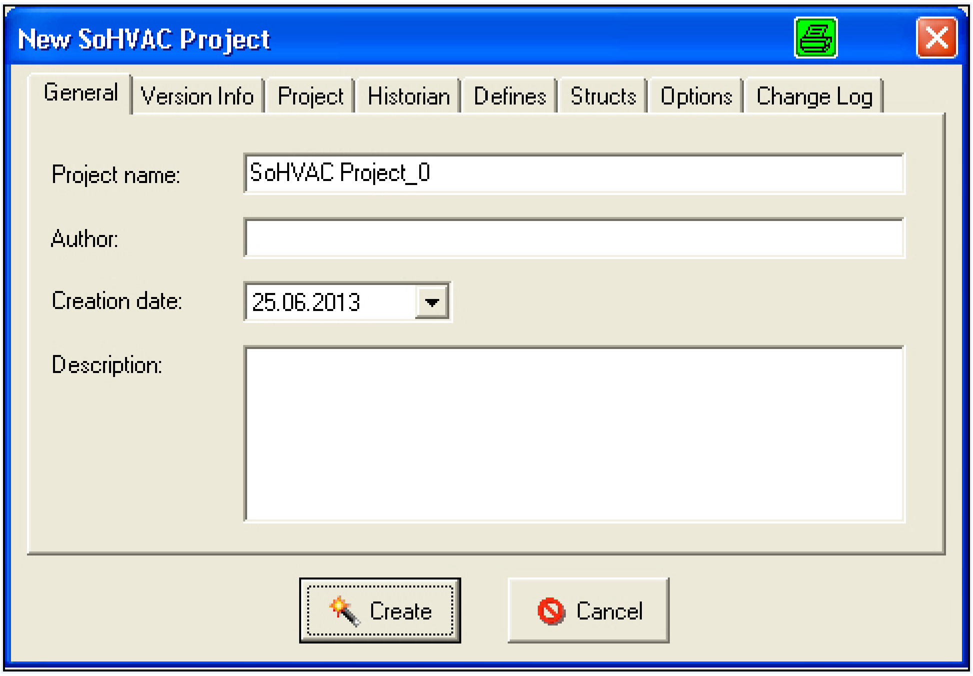
☑ Software Bits (Math, Stack, ...)

☐ ExpBUS Network (Status, Parameters, ...)

☐ Analog Inputs (Errors, Timeout, ...)

If you select Create SoHVAC Template Project, it is possible to select which information will be added to the project.

Once you have clicked **OK** to confirm the type of project to create, a window containing a list of all project properties appears:



Project name: SoHVAC Project_0

Author:

Creation date: 25.06.2013

Description:

When you create a new project, enter its name, the name of the author, the creation date (the current date appears by default) and a brief description of the project under the General tab.

Under the Version Info tab, the project can be identified by a project number, a version and a revision.

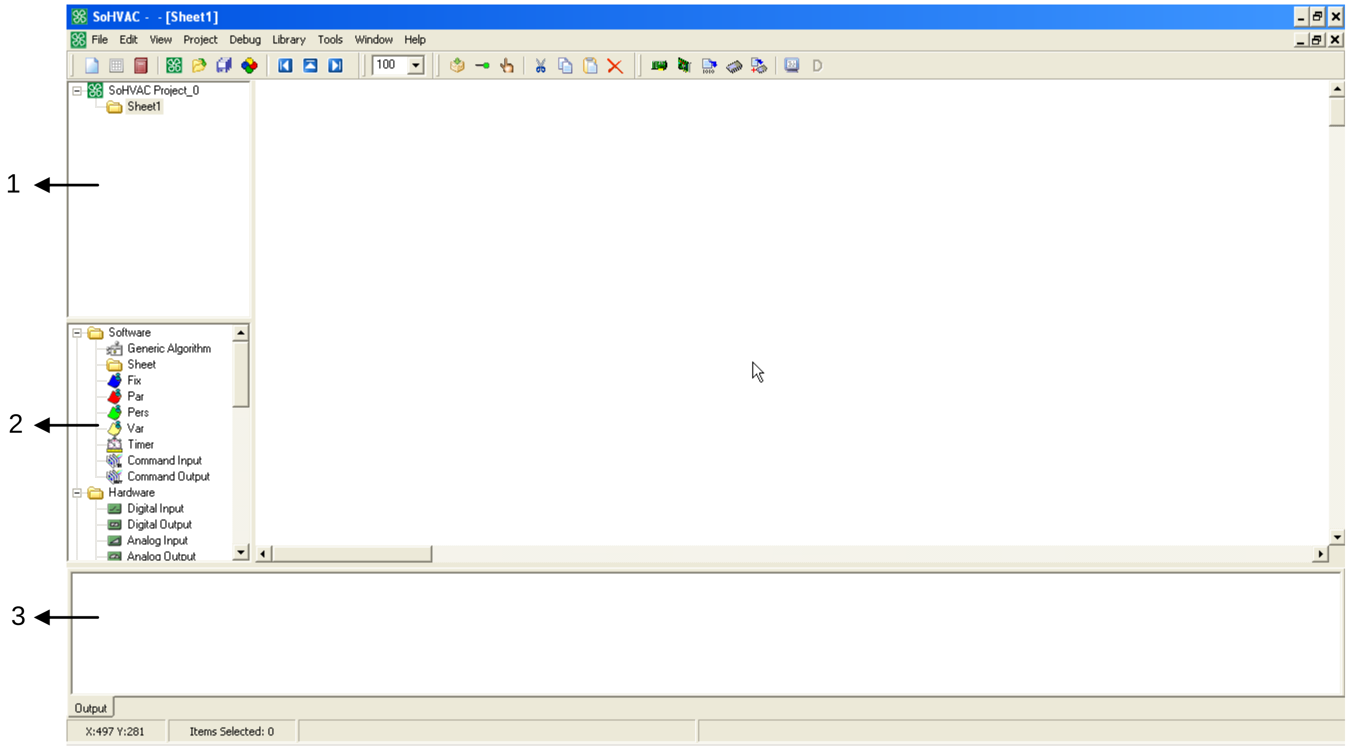
To create the project, click **Create**.

All project options can be changed later by choosing the **Project→Properties** menu or clicking the corresponding icon in the toolbar .

3.2 The Screen Organization

The program window is divided into several areas:

- menus on top
- project Inspector and Library Manager on the left
- programmable area in the center
- output window at the bottom



The window is divided into 3 lists:

1. **Project Inspector**
2. **Library Manager**
3. **Output Window**

You can display/hide the Project Inspector window through the **View→View Project Inspector** menu.

You can display/hide the corresponding windows with menu **View→View Library Manager** and **View→View Output Window**.

3.3 Entities: Definition and Use

Entities, a basic element in the **SoHVAC** engineering software tool, 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 (refer to Basic Elements Data Types).
- A variable number of outputs, characterized by a type (refer to Basic Elements Data Types).
- 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.

Examples of entities are variables, digital inputs (DI), digital outputs (DO), algorithms, libraries, and so on.

3.3.1 Name of the Entity

Each entity in the project needs to have a unique name that characterizes it.

Below are the rules to be followed when naming an entity:

- The names of entity, algorithms, categories, inputs, outputs, and so on must not exceed 80 characters.
- The compiler used to compile the project is not case-sensitive type. So, 2 different entity names should not differ only in capital/lowercase letters.
- For example, 2 entities with associate names requestFans and RequestFANS generate an error during compilation.
- The name of an entity should be C compliant and should not contain special characters like spaces, commas, dots, apostrophes, accents, quotes, and mathematical signs (+,-,*,/).

3.3.2 Put an Entity on a Worksheet

To put an entity in a worksheet, you need to select it from the library manager and drag-and-drop it.

The entity appears in its default size, which can be changed later.

3.3.3 Moving Entities

To move an entity to a different place in a worksheet, do one of the following:

- Drag-and-drop it to the required position.
- Select one or more entities and press the arrow keys (to move them faster, hold down CTRL while pressing the arrow keys).

3.3.4 Removing Entities

To remove an entity from a worksheet, do one of the following:

- Select and press **DEL** or select **Edit→Delete** from the menu
- Select an entity, right-click and select **Delete** from the context menu

A confirmation window asks you to confirm if you wish to remove the entity. This operation may be performed with multiple selected entities as well.

If you remove a subsheet, all the entities contained in it are also removed.

When an entity is removed, all the joins to other entities are removed. To save the links, refer to *Removing/Changing Entities and Saving Links*.

Any references found in the EIML pages are also deleted.

3.3.5 Removing/Changing Entities and Saving Links

To remove an entity and keep the links for further use, do the following:

- Select the entity to be removed, right-click and select **Delete** and **Save Links** from the context menu.

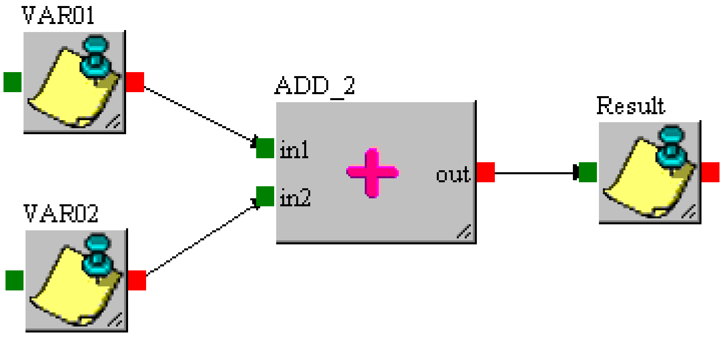
The entity is removed and connection entities are created. These entities disappear automatically when they are connected both the sides with other blocks.

To change an entity, do the following:

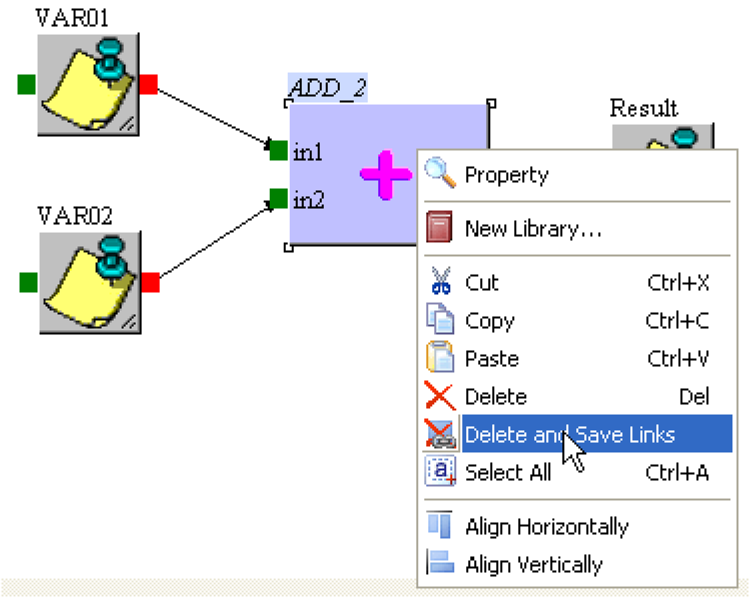
- Select the entity to be changed, right-click and select **Delete** and **Save Links** from the context menu. Connection entities are created.
- Create the new entity.
- Link the new entity to the connection entities. The connection entities disappear automatically. A direct link is made between the new entity and existing entities (vars…).

Changing entity example:

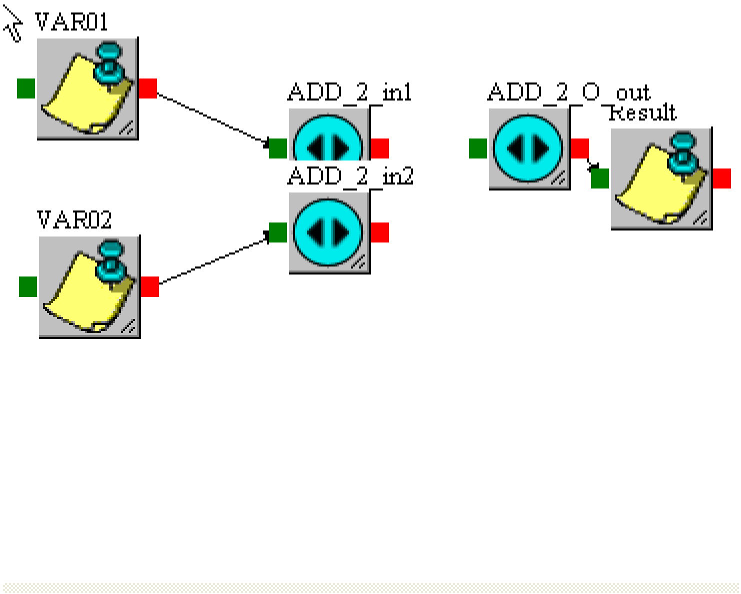
Create a program to add 2 variables and store the result in Result Variable.



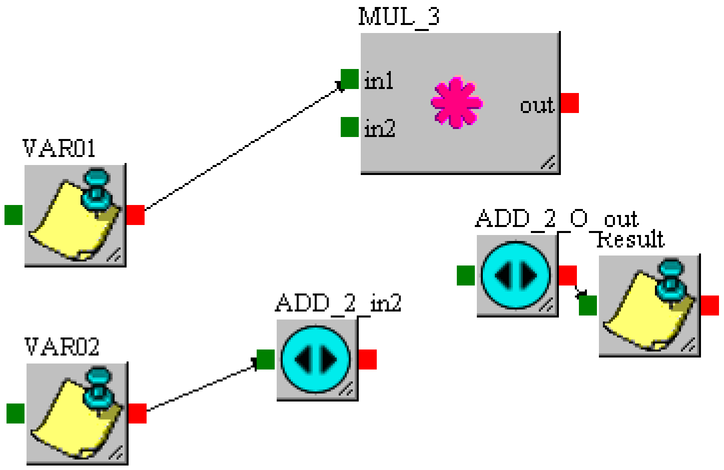
To replace the ADD entity by MUL entity, select **Delete and Save Links**.



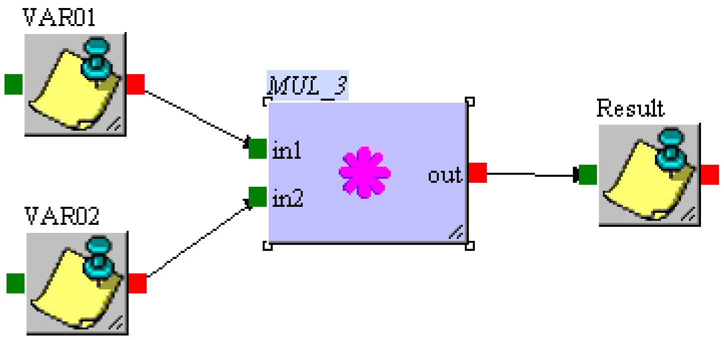
3 Connection entities are created.



Add the MUL entity and start to link ADD_2_in1 output to in1 input of MUL entity. The ADD_2_in1:



Finalize all the connections and move the MUL entity.



3.3.6 Cut and Paste of a Entity with EIML Connection

When you cut one or more entities, if those entities are linked on an EIML page, a confirmation message Do you want to save the selected entities links appears.

Click **Yes** to save the link between entities and the EIML page

Click **No** to cut the link between entities and the EIML page. The EIML Variable is not linked to a VAR any more.

Click **Cancel** to abort the cut operation.

3.4 Control Sheets and Subsheets

A control sheet is a special entity which has the ability to contain other entities (including sheets). You can implement a tree-structured project applying either a Top-Down approach, Bottom-Up approach or a mixture of both according to your preferences and the needs of the project.

Therefore, a control sheet allows you to group certain project entities in a logical macro-entity. If you save this sheet as a library function block, you can reuse it in the future by taking it from the library manager.

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

By default the program is created with one control sheet named Sheet1.

3.4.1 Creating a New Sheet

To create a new sheet, do one of the following:

- Select **File→New→New Sheet** from the menu.
- Click the 🗋 icon in the toolbar.
- Press CTRL+N.

3.4.2 Creating a New Subsheet

There are 2 approaches to add a subsheet to a project:

- Select the empty subsheet by selecting the 📁 icon in the Software library and insert the different entities in it. Subsheet creation is also available through menu **File→New→New Sub Sheet**.
- Select one or more entities from the active sheet by selecting the entities you wish to group, select **Library→Generate SubSheet** from the menu or click the 📦 icon in the toolbar. In this case, a new subsheet is created and the selected entities are moved to the new subsheet.

To join the entities found in a subsheet to external entities, you need to export the input and output pins of the involved entities (refer to Linking Entities).

3.5 Navigation in Sheets

The following arrows ◀ ▲ ▶ in the toolbar allow you to navigate in the different sheets and sub sheets.

When you click the:

- up arrow ▲, you can access the upper level of sheets
- left arrow ◀ and the right arrow ▶, you can navigate between the previous active pages

3.6 Linking Entities

To set up a project, you need to insert entities in the sheets and link them together in order to achieve the logical behavior of the controller.

Linking entities means to link each output to the respective inputs of other entities according to certain rules given:

- One output pin can be linked to multiple input pins.
- The data type associated with the input pin has to be compatible with the data type associated to the output pin.

3.6.1 Link Entities

To link 2 entities, do one of the following:

- If the entities are in the same sheet, you can link them by means of an arrow. Place mouse pointer over the output pin of the entity you want to link until it is highlighted. By clicking the left mouse button and moving the mouse, an arrow is drawn between the selected output and the pointer. If you want to draw a link composed of multiple angled lines, click at the point where you want to bend the arrow and go on moving the mouse in a different direction towards the input pin. When you place mouse pointer over the input pin, it is highlighted. Click to complete the link. If the input had not been linked before and if the input and output types are compatible, an arrow is drawn to confirm that both entities have been linked. If you want to cancel this operation at any time, click the right mouse button.
- If the entities are placed in different sheets, you can 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 mouse pointer over the pin you wish to export, right-click and select **Export IO** from the context menu. The exported pin changes color (it becomes blue by default) and appears outside the subsheet. Suppose you want to link a Digital Input to a Var contained in a subsheet, you need to export the variables input pin. Now you can link the output pin of the Digital Input to the pin exported from the subsheet.

Or, you can link inputs and outputs (even if placed in different sheets) without using arrow connections, but using the external links.

3.6.2 External Links

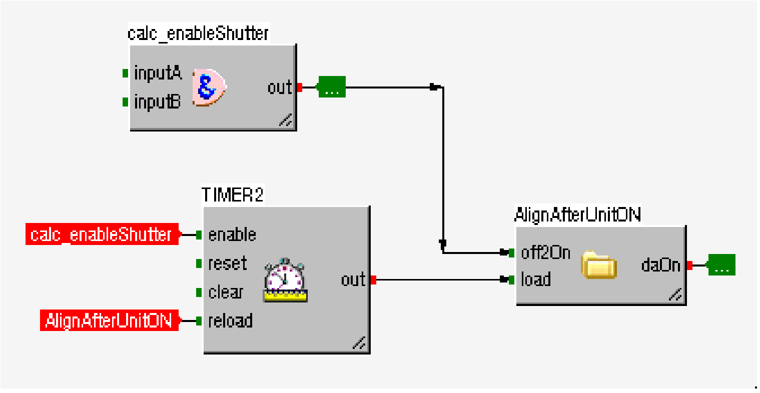
To specify an external link, you need to perform the following steps in order:

1. Click the icon from the toolbar, select the output you wish to link, and select the input.
2. This function is very useful if links are particularly tangled or high in number, but it makes the project less readable.

3. To unlink 2 entities, you can proceed in 2 different ways according to the type of link:

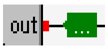
- If the entities are linked by an arrow, select the arrow and remove it by clicking **Del** or select **Edit→Delete** from the menu.
- If the entities were linked using an external link, you have to display the **Show links…** window, select the link and remove it by clicking the appropriate key.

The following program is an example for external links:



3.6.3 Link Out

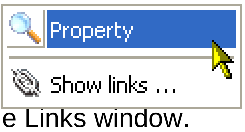
The output terminals are designed as follows:



Context menu:

Property: link properties. Refer to *Input and Output Properties*.

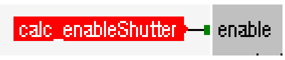
Show links: window with the connected links



A double-click on the terminal immediately opens the Links window.

3.6.4 Link In

The input terminals are designed as follows:

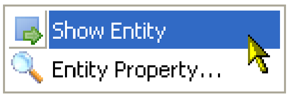


, the colored box display the name of the entity to which they are connected; in this case calc_enableShutter.

Context menu:

Show Entity: Select and display the entity connected (Output terminal).

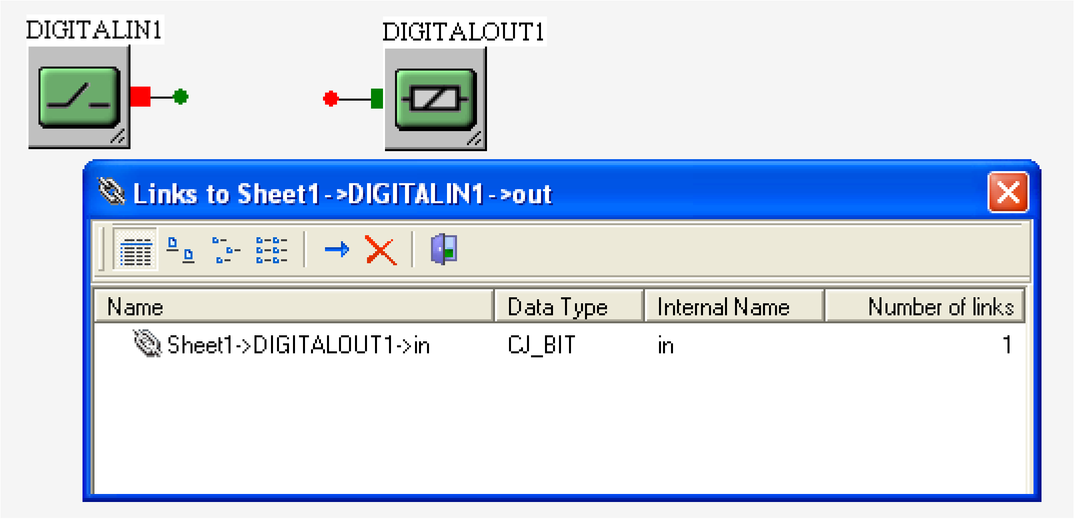
Entity Property: Open the properties of the entity connected (Output terminal).



A double-click on the terminal returns the selection to the connected entity.

3.6.5 Show Link Context Menu

Once you have finished joining the entities, you can display the link you have created by selecting Show links… from the context menu of the input or output.



Links to Sheet1->DIGITALIN1->out

Name	Data Type	Internal Name	Number of links
Sheet1->DIGITALOUT1->in	CJ_BIT	in	1

The menu helps to manage the different links of the terminal selected:

→ Go to the selected link.

✕ Remove selected link.

3.7 Creating EIML Pages

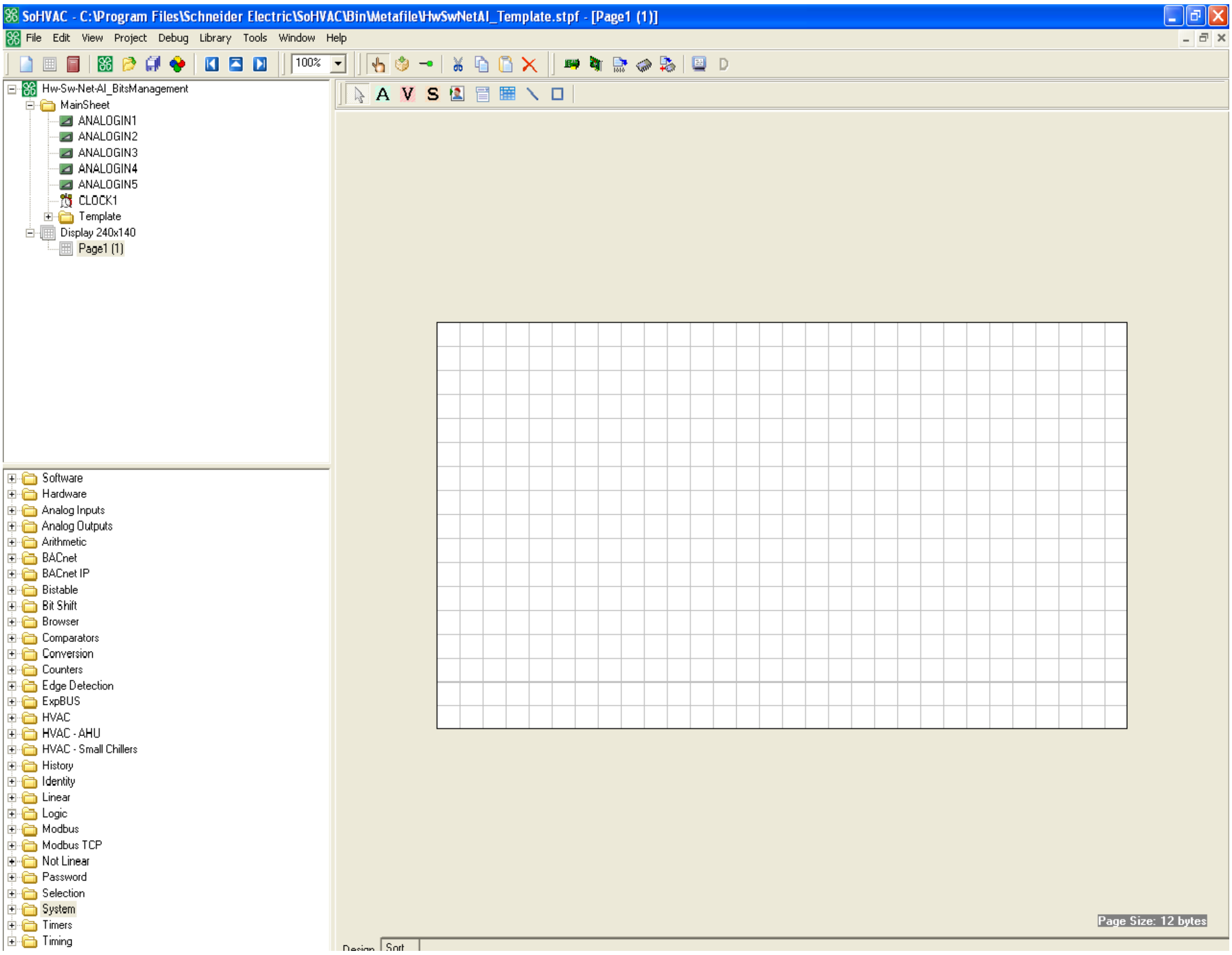
EIML pages allow you to describe the graphical interface of the controller in graphical format. A project may include multiple interconnected pages to display texts, icons, and variables representing the internal states of the controller.

To add a page to the project, select **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 appears in the project tree.

The visual representation of the page appears 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 240 x 140 pixel LCD graphic display.



EIML pages can contain texts, variables, icons, combos, tables, lines, and rectangles. Refer to *Entities: Definition and Use*.

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 need to switch to EIML page simulation.

3.8 Selecting Hardware

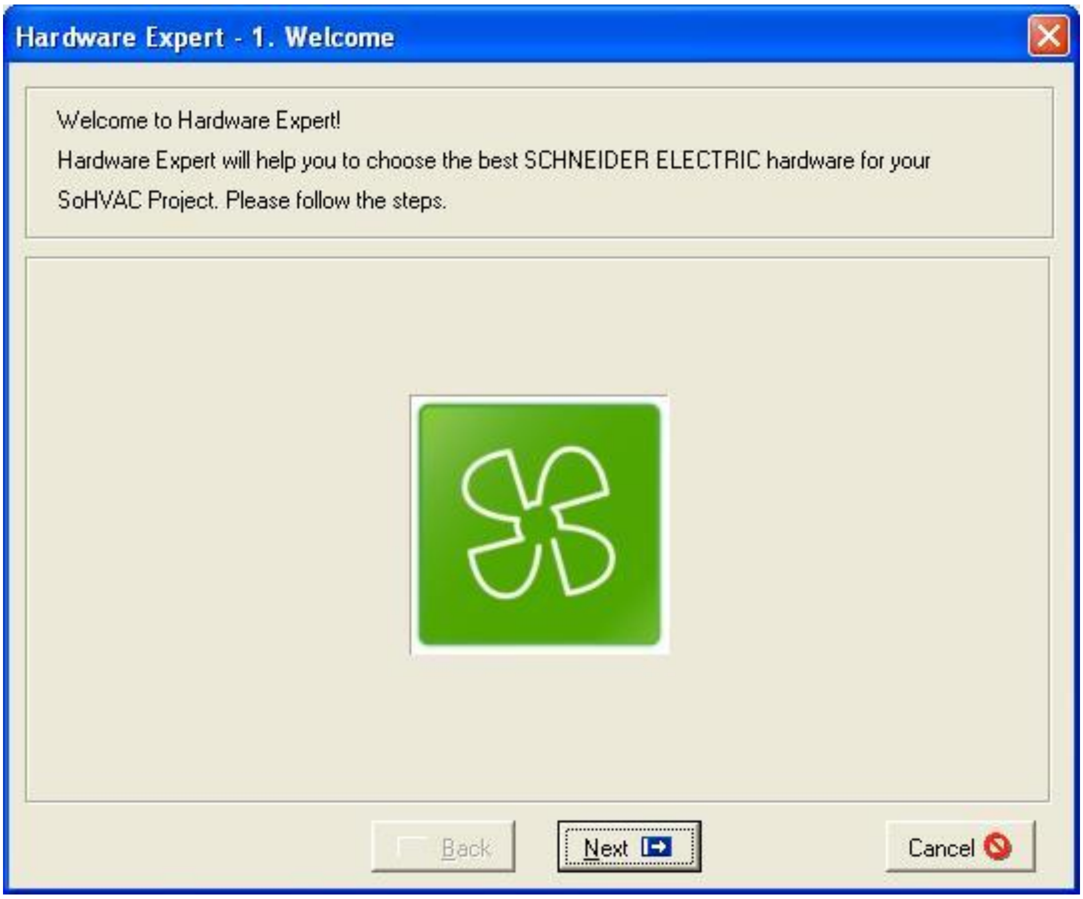
A project developed using SoHVAC can be executed on the family of controllers compatible with this development environment. So you should select the controller, on which you wish to execute the project.

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

- Selecting **Project→Hardware Expert** from the menu
- Clicking the **Hardware Expert** icon.

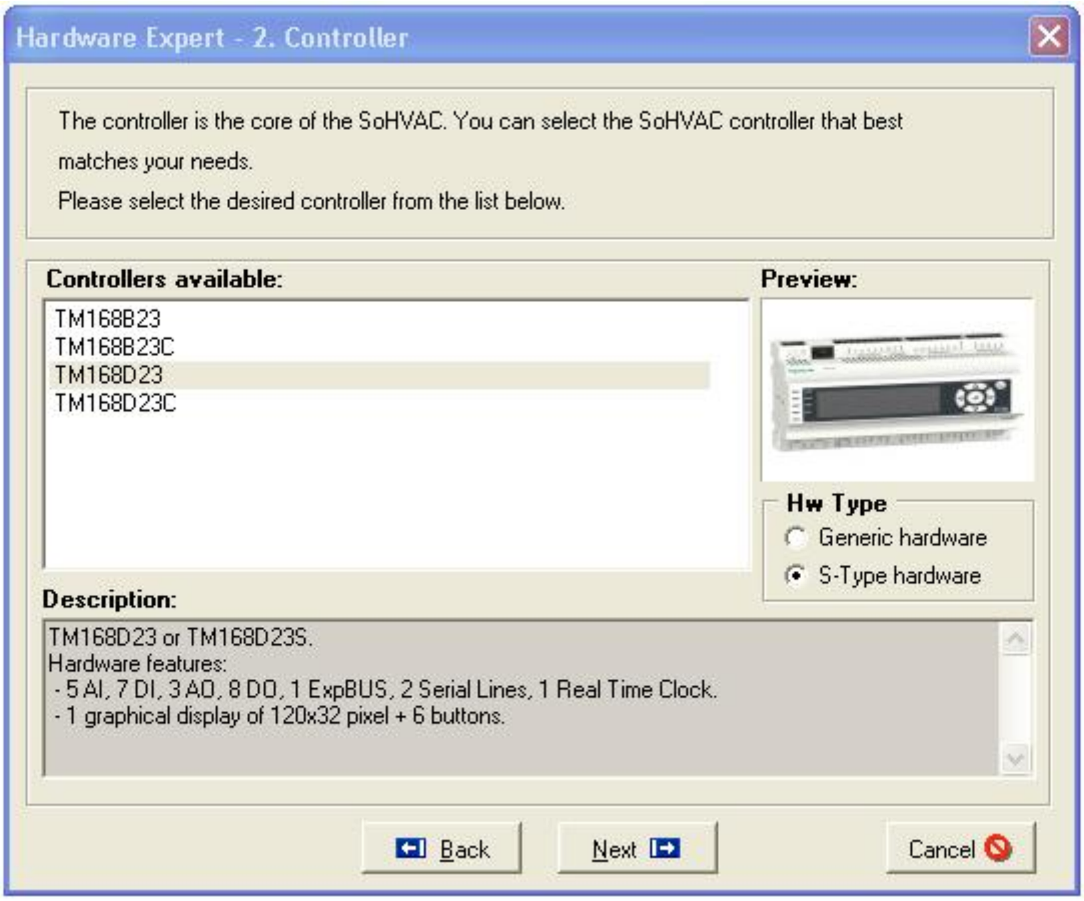
The wizard starts with a welcome window.



Hardware Expert - 1. Welcome

Welcome to Hardware Expert!
Hardware Expert will help you to choose the best SCHNEIDER ELECTRIC hardware for your SoHVAC 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 SoHVAC. You can select the SoHVAC controller that best matches your needs.
Please select the desired controller from the list below.

Controllers available:

- TM168B23
- TM168B23C
- TM168D23
- TM168D23C

Preview:

Hw Type

○ Generic hardware
● S-Type hardware

Description:

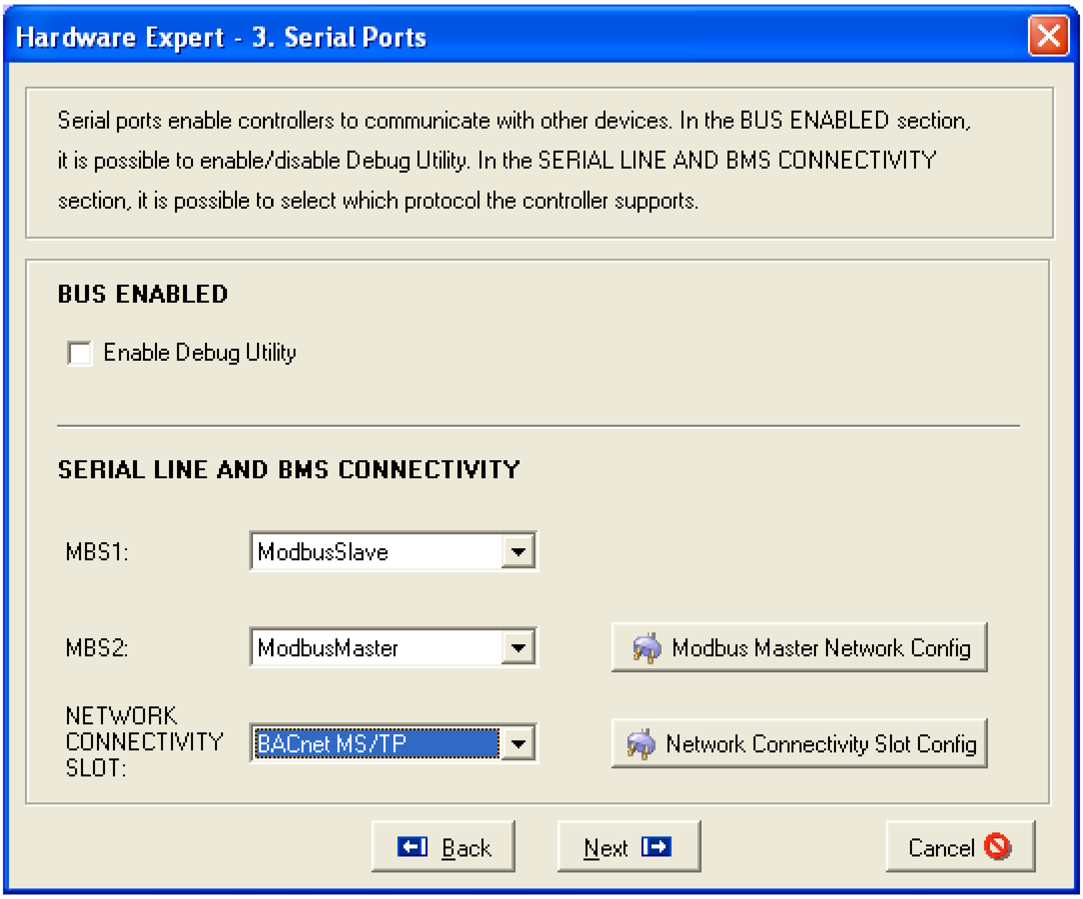
TM168D23 or TM168D23S.
Hardware features:
- 5 AI, 7 DI, 3 AO, 8 DO, 1 ExpBUS, 2 Serial Lines, 1 Real Time Clock.
- 1 graphical display of 120x32 pixel + 6 buttons.

When you select a controller, a brief description of the resources available in the controller appears in the center of the screen and a preview of the controller appears on the right side.

There are 2 different reference types of controller hardware: **Generic Hardware** and **S-Type Hardware**.

S-Type Hardware is required for the execution of certain function blocks. For more information, refer to *SoHVAC Function Block Help documentation* to determine the type of controller hardware you may need.

Click 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 Debug Utility. In the SERIAL LINE AND BMS CONNECTIVITY section, it is possible to select which protocol the controller supports.

BUS ENABLED

□ Enable Debug Utility

SERIAL LINE AND BMS CONNECTIVITY

MBS1: ModbusSlave

MBS2: ModbusMaster

NETWORK CONNECTIVITY SLOT: BACnet MS/TP

Modbus Master Network Config
Network Connectivity Slot Config
Back
Next
Cancel

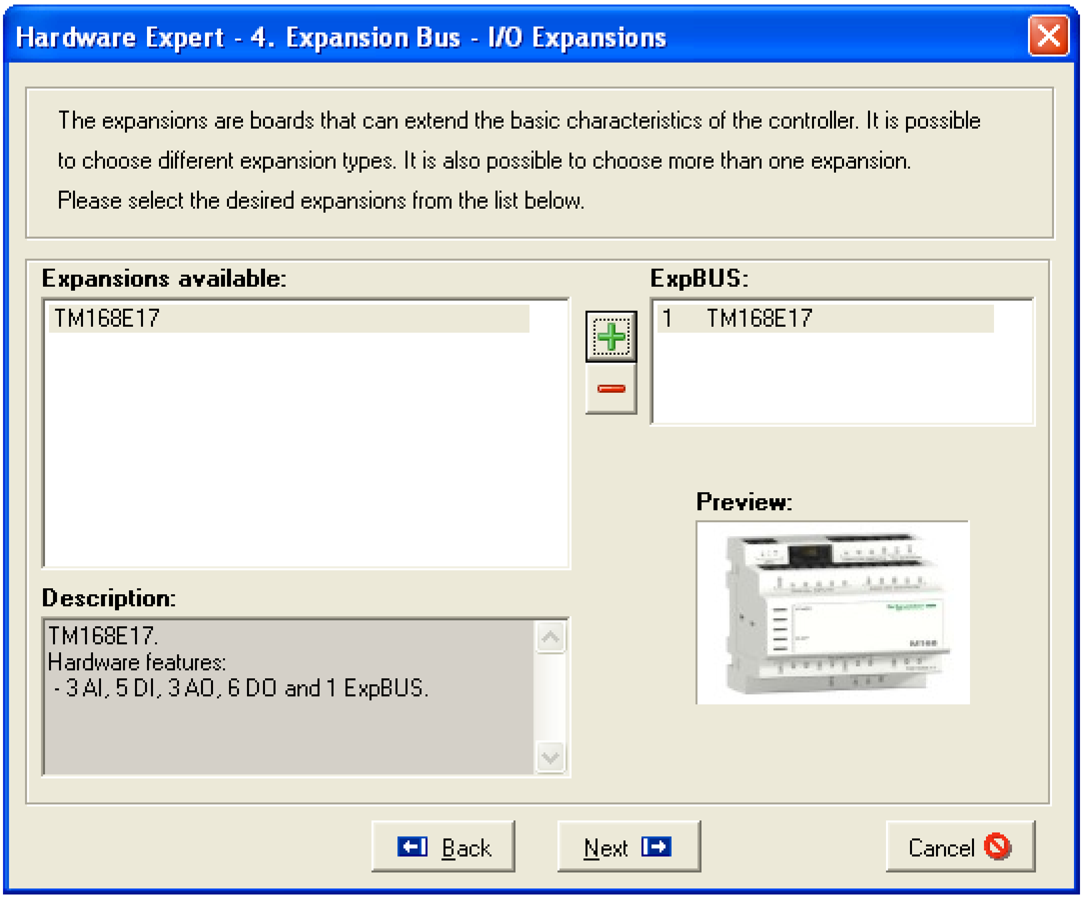
The third wizard step is divided into 2 sections.

1. The first one allows you to enable the debugger channel to debug the program.
2. The second one allows to select a protocol to use for each available communication port:

- **None**, **ModbusSlave** for the serial line **MBS1**.
- **None**, **ModbusSlave**, **ModbusMaster**, for the serial line **MBS2**.
- **None**, **BACnet MS/TP**, **BACnet IP**, **ModBus TCP** for the **Network Connectivity Slot**.

The **Modbus Master Network Config** button configures the serial line connectivity port and the **Network Connectivity Slot Config** button configures the BMS connectivity port.

If you click **Next**, you will gain access to configure the expansions on ExpBUS.



Hardware Expert - 4. Expansion Bus - I/O Expansions

The expansions are boards that can extend the basic characteristics of the controller. It is possible to choose different expansion types. It is also possible to choose more than one expansion.

Please select the desired expansions from the list below.

Expansions available:

TM168E17

ExpBUS:

1 TM168E17

Preview:

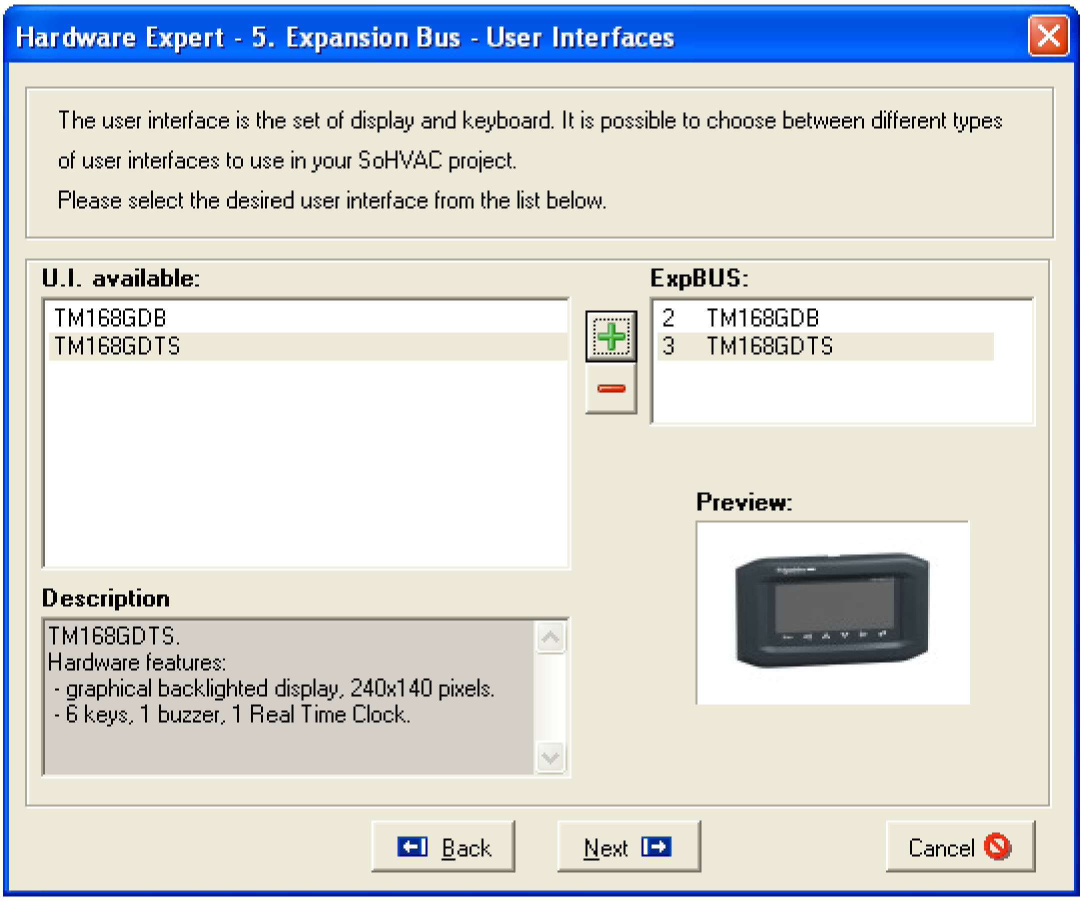
Description:

TM168E17.
Hardware features:
- 3 AI, 5 DI, 3 AO, 6 DO and 1 ExpBUS.

At the left side, there is a list of available expansions depending on the controller which can be connected to the ExpBUS.

To add an expansion, select an expansion and click the ➕ button. A new row appears at the right side.

Once you have selected all the expansions to be added, click **Next** to proceed.



Hardware Expert - 5. Expansion Bus - 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 SoHVAC project.

Please select the desired user interface from the list below.

U.I. available:

TM168GDB
TM168GDTS

ExpBUS:

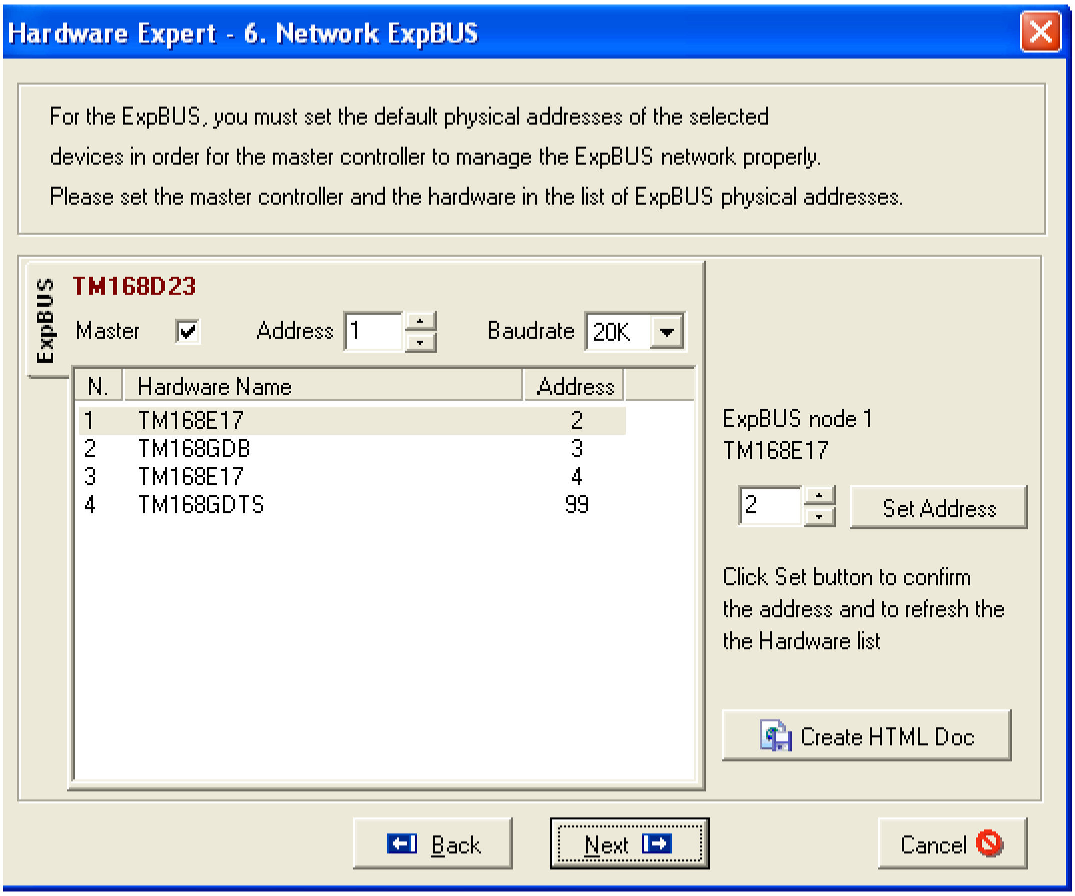
2	TM168GDB
3	TM168GDTS

Description

TM168GDTS.
Hardware features:
- graphical backlighted display, 240x140 pixels.
- 6 keys, 1 buzzer, 1 Real Time Clock.

The fifth wizard step is similar to the fourth one. It allows you to add **User Interfaces** to the ExpBUS.

Click **Next**.



Hardware Expert - 6. Network ExpBUS

For the ExpBUS, you must set the default physical addresses of the selected devices in order for the master controller to manage the ExpBUS network properly. Please set the master controller and the hardware in the list of ExpBUS physical addresses.

TM168D23

Master ☑ Address 1 Baudrate 20K

N.	Hardware Name	Address
1	TM168E17	2
2	TM168GDB	3
3	TM168E17	4
4	TM168GDTS	99

ExpBUS node 1
TM168E17

2 Set Address

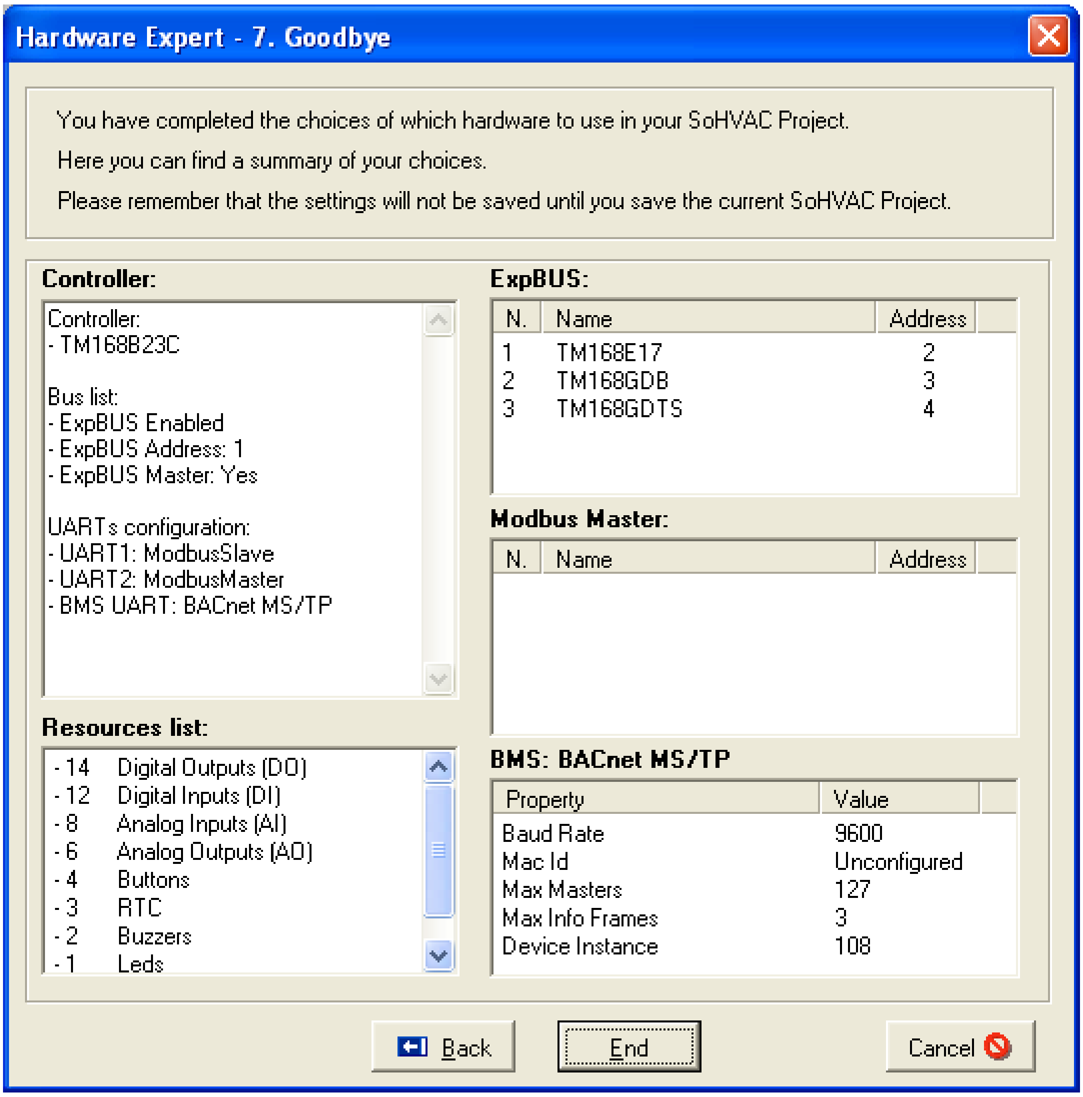
Click Set button to confirm the address and to refresh the the Hardware list

It is possible to set the parameters that define the ExpBUS network as its physical address, the fact that the controller manages as master the network, all the physical addresses of the other network elements.

It is also possible to create an html document with all network settings.

NOTE: If the controller has to manage the expansions, it is important to flag the **Master controller** field.

In the last step, the wizard summarizes the selected configuration 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 SoHVAC Project.

Here you can find a summary of your choices.

Please remember that the settings will not be saved until you save the current SoHVAC Project.

Controller:

Controller:
- TM168B23C

Bus list:
- ExpBUS Enabled
- ExpBUS Address: 1
- ExpBUS Master: Yes

UARTs configuration:
- UART1: ModbusSlave
- UART2: ModbusMaster
- BMS UART: BACnet MS/TP

Resources list:

- 14	Digital Outputs (DO)
- 12	Digital Inputs (DI)
- 8	Analog Inputs (AI)
- 6	Analog Outputs (AO)
- 4	Buttons
- 3	RTC
- 2	Buzzers
- 1	Leds

ExpBUS:

N.	Name	Address
1	TM168E17	2
2	TM168GDB	3
3	TM168GDTS	4

Modbus Master:

N.	Name	Address

BMS: BACnet MS/TP

Property	Value
Baud Rate	9600
Mac Id	Unconfigured
Max Masters	127
Max Info Frames	3
Device Instance	108

Click **End** to terminate the wizard.

Click **Cancel** to cancel all the changes made.

3.9 Connection to Physical Terminals

The entities used in the SoHVAC are a logical representation not related to the controller type.

Before compilation, you need to specify the connections between the logical resources used in the project and the physical resources available in the selected controller.

Therefore 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) needs to be joined to the local or remote physical I/O through the **Join Tools**.

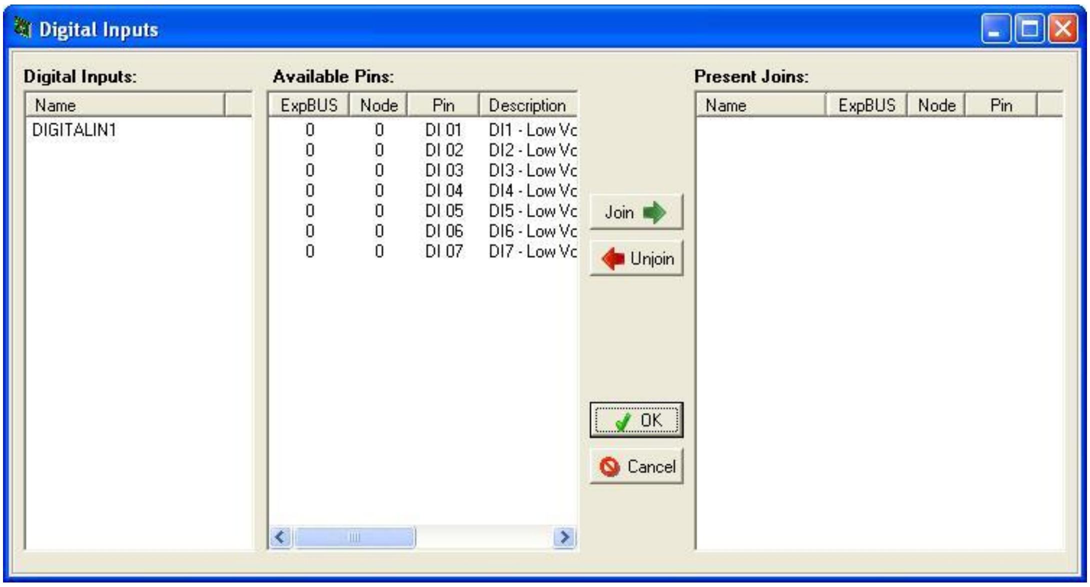
You can use the **Check Joins** function to verify all joins and if needed, open up the required windows to join those entities that have not yet been assigned:

- Click the icon on the toolbar menu.
- Select **Project→Join Tools→Check Joins** from the menu.

If you try to compile a program where all joins have not been made, the dedicated join windows open automatically.

3.9.1 Digital Inputs

The digital input joining wizard can be started manually by selecting **Project→Join Tools→Digital Inputs** from the menu.



Name
DIGITALIN1

ExpBUS	Node	Pin	Description
0	0	DI 01	DI1 - Low Vc
0	0	DI 02	DI2 - Low Vc
0	0	DI 03	DI3 - Low Vc
0	0	DI 04	DI4 - Low Vc
0	0	DI 05	DI5 - Low Vc
0	0	DI 06	DI6 - Low Vc
0	0	DI 07	DI7 - Low Vc

Name	ExpBUS	Node	Pin

The window is divided into 3 lists:

1. **Digital Inputs**
2. **Available Pins**
3. **Present Joins**

The first list contains the project entities that have not yet been linked to a hardware resource.

The second list shows the physical digital inputs that are still available with the following characteristics:

- **ExpBUS**: bus to which the controller is connected
- **Node**: node of the controller where the resource is found
- **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.

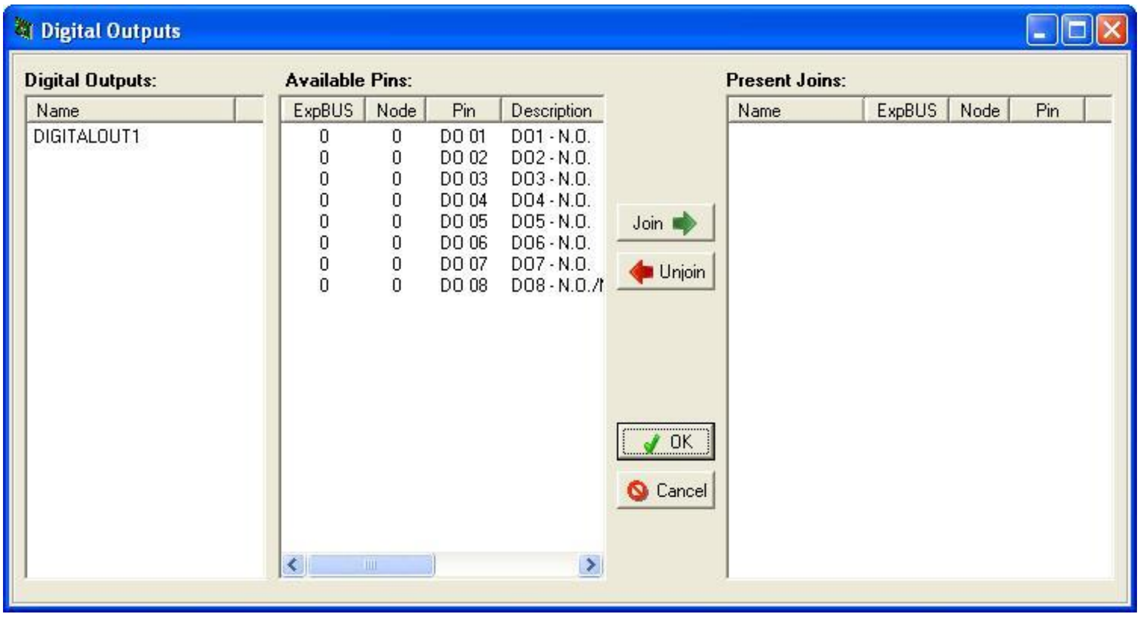
Both the entity and the hardware terminal disappear from the lists and a new row is created in the **Joins Present** table.

To remove a join, select the row that represents it in the **Joins Present** table and click **Unjoin**. The table row is eliminated. The logical terminal is restored to the first table and the physical resource to the second.

Click the **OK** or **Cancel** buttons to validate or cancel the changes.

3.9.2 Digital Outputs

The digital output joining wizard can be started manually by selecting **Project→Join Tools→Digital Outputs** from the menu.



Digital Outputs:

Name
DIGITALOUT1

Available Pins:

ExpBUS	Node	Pin	Description
0	0	DO 01	DO1 - N.O.
0	0	DO 02	DO2 - N.O.
0	0	DO 03	DO3 - N.O.
0	0	DO 04	DO4 - N.O.
0	0	DO 05	DO5 - N.O.
0	0	DO 06	DO6 - N.O.
0	0	DO 07	DO7 - N.O.
0	0	DO 08	DO8 - N.O./

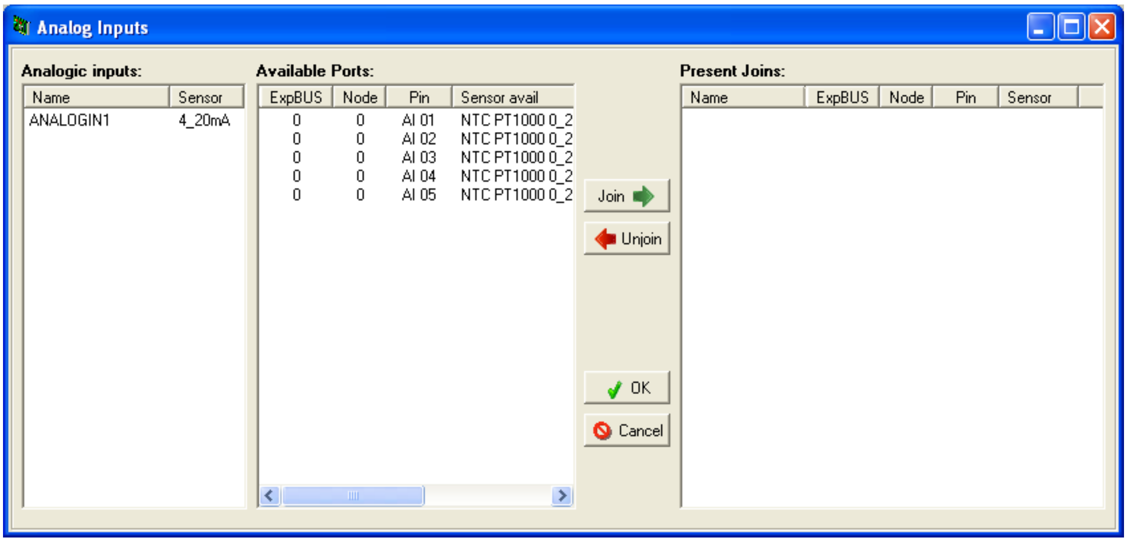
Present Joins:

Name	ExpBUS	Node	Pin

For more information, refer to Digital Inputs.

3.9.3 Analog Inputs

The Analog Input joining wizard can be started manually by selecting **Project→Join Tools→Analog Inputs** from the menu.



Analog Inputs

Analogic inputs:

Name	Sensor
ANALOGIN1	4_20mA

Available Ports:

ExpBUS	Node	Pin	Sensor avail
0	0	AI 01	NTC PT1000 0_2
0	0	AI 02	NTC PT1000 0_2
0	0	AI 03	NTC PT1000 0_2
0	0	AI 04	NTC PT1000 0_2
0	0	AI 05	NTC PT1000 0_2

Present Joins:

Name	ExpBUS	Node	Pin	Sensor

The window is divided into 3 lists:

1. **Analog Inputs**
2. **Available Ports**
3. **Present Joins**

The first list contains the project entities that have not yet been linked to a hardware resource with the following characteristics:

- **Name**: entity name in the program
- **Sensor**: sensor type selected for this entity

The second list shows the physical analog inputs that are still available with the following characteristics:

- **ExpBUS**: bus to which the controller is connected
- **Node**: node of the Bus where the resource is found
- **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, click the **Join** button.

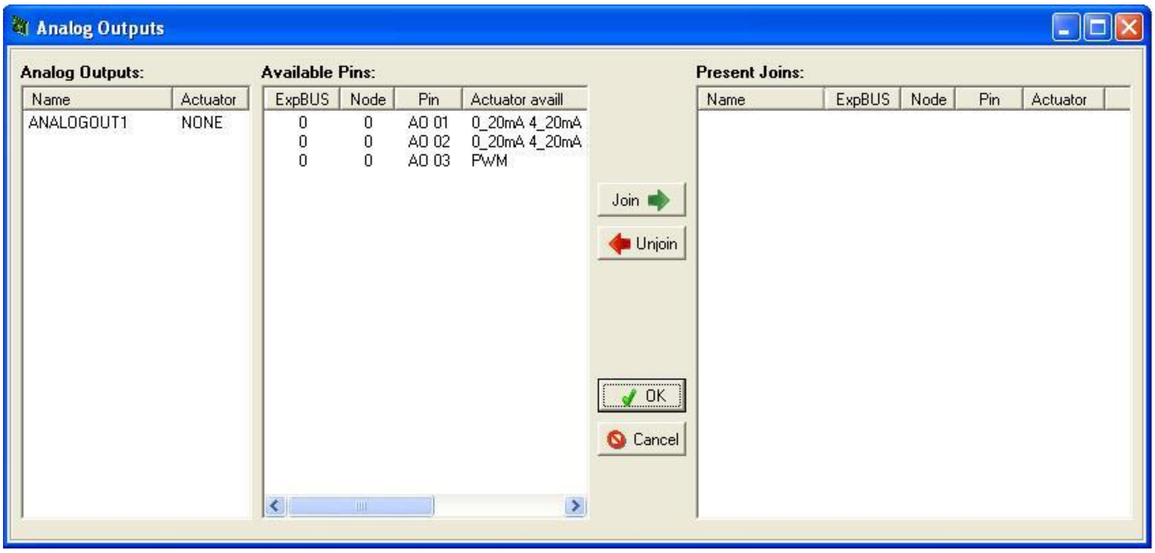
Both the entity and the hardware terminal disappear from the list and a new row is created in the **Joins Present** table.

To remove a join, select the row that represents it in the **Joins Present** table and click **Unjoin.** The table row is eliminated. The logical terminal is restored to the first table and the physical resource to the second.

Click the **OK** or **Cancel** buttons to validate or cancel the changes.

3.9.4 Analog Outputs

The **Analog Output** joining wizard can be started manually by selecting **Project→Join Tools→Analog Outputs** from the menu.



Analog Outputs:

Name	Actuator
ANALOGOUT1	NONE

Available Pins:

ExpBUS	Node	Pin	Actuator avail
0	0	AO 01	0_20mA 4_20mA
0	0	AO 02	0_20mA 4_20mA
0	0	AO 03	PWM

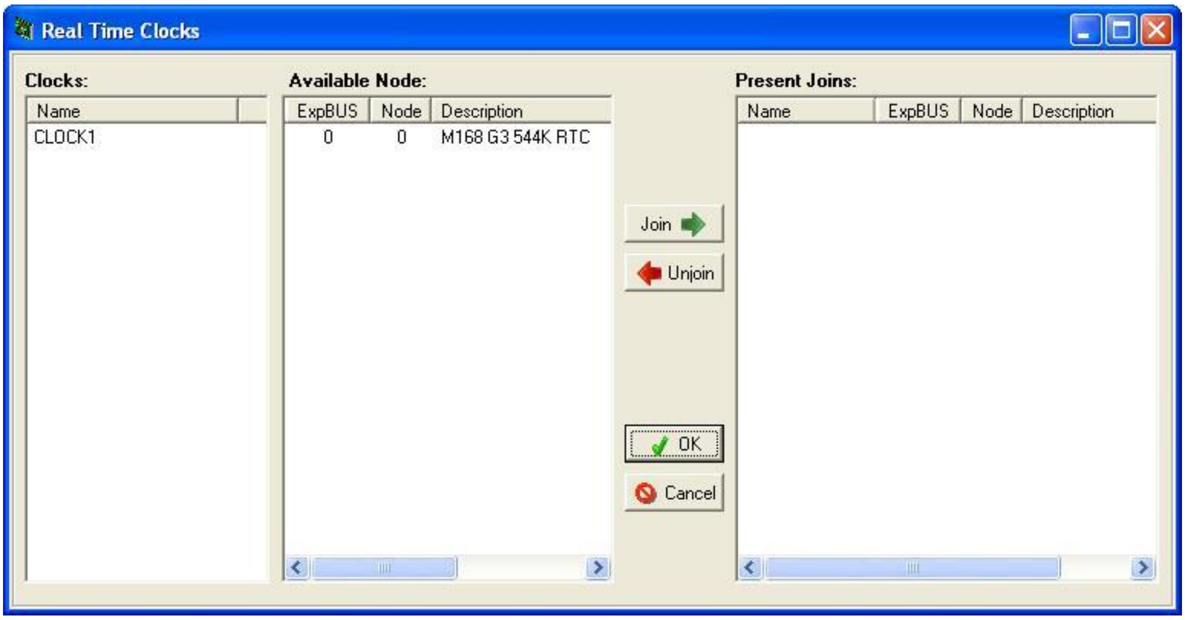
Present Joins:

Name	ExpBUS	Node	Pin	Actuator

For more information, refer to *Analog Inputs*.

3.9.5 Clocks

The **Clocks** joining wizard can be started manually by choosing **Project→Join Tools→Real Time Clocks** from the menu.



Clocks:

Name
CLOCK1

Available Node:

ExpBUS	Node	Description
0	0	M168 G3 544K RTC

Present Joins:

Name	ExpBUS	Node	Description

The window is divided into 3 lists:

1. **Clocks**
2. **Available Node**
3. **Present Joins**

The first list contains the project entities that have not yet been linked to a hardware resource.

The second list shows the physical clocks that are still available with the following characteristics:

- **ExpBUS**: bus to which the controller is connected
- **Node**: node of the controller where the resource is found
- **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 click the **Join** button.

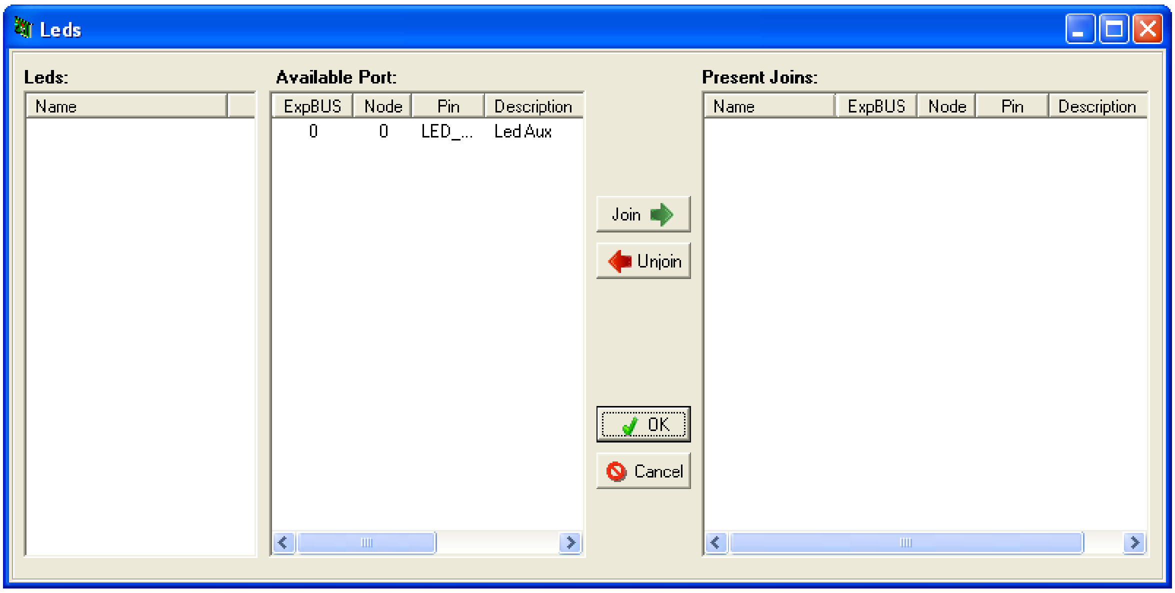
Both the entity and the hardware terminal will disappear from the list and a new row is created in the **Joins Present** table.

To remove a join, select the row that represents it in the **Joins Present** table and click **Unjoin**. The table row is eliminated. The logical terminal is restored to the first table and the physical resource to the second.

Click the **OK** or **Cancel** buttons to validate or cancel the changes.

3.9.6 LEDs

The LED joining wizard can be started manually by selecting **Project→Join Tools→LEDs** from the menu.



Leds:

Name

Available Port:

ExpBUS	Node	Pin	Description
0	0	LED_...	Led Aux

Present Joins:

Name	ExpBUS	Node	Pin	Description

The window is divided into 3 lists:

1. **LEDs**
2. **Available Port**
3. **Present Joins**

The first list contains the project entities that have not yet been linked to a hardware resource.

The second list shows the physical LEDs that are still available with the following characteristics:

- **ExpBUS**: bus to which the controller is connected
- **Node**: node of the controller where the resource is found
- **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 click the **Join** button.

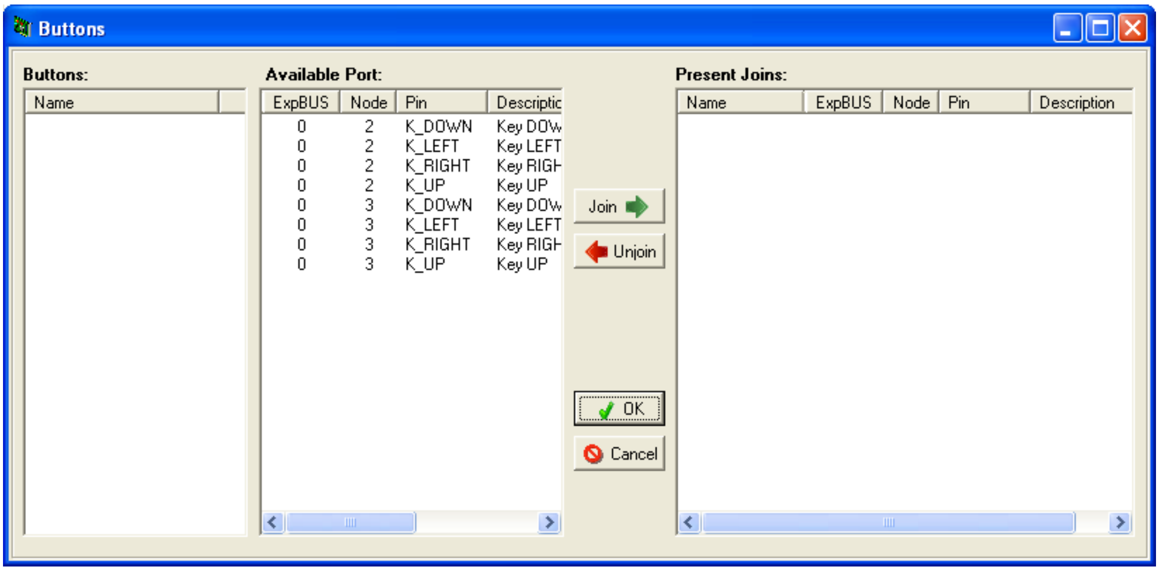
Both the entity and the hardware terminal disappear from the list and a new row is created in the **Joins Present** table.

To remove a join, select the row that represents it in the **Joins Present** table and click **Unjoin**. The table row is eliminated. The logical terminal is restored to the first table and the physical resource to the second.

Click the **OK** or **Cancel** buttons to validate or cancel the changes.

3.9.7 Buttons

The **Button** joining wizard can be started manually by selecting **Project→Join Tools →Buttons** from the menu.

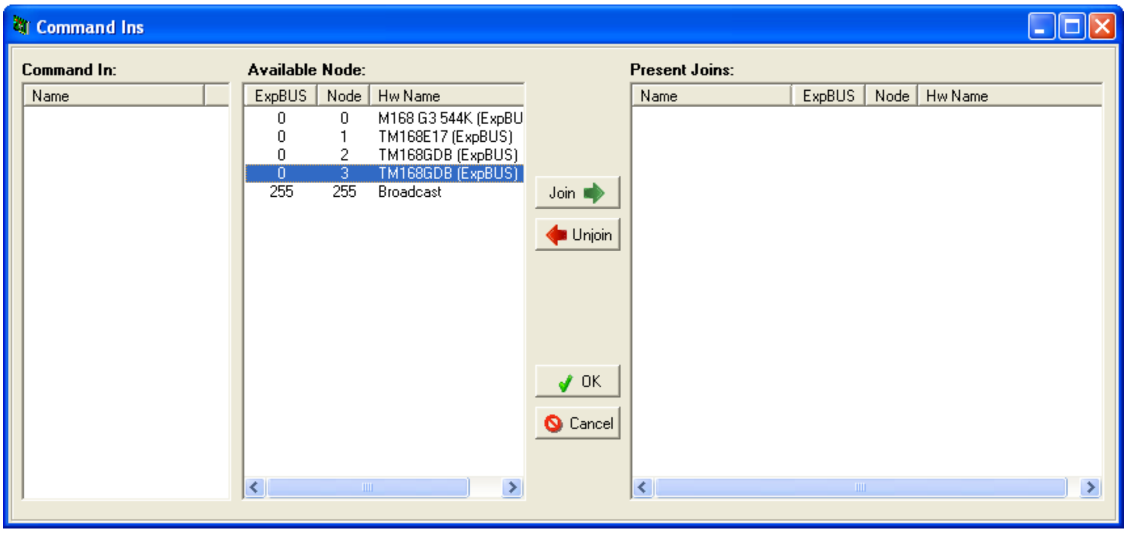


ExpBUS	Node	Pin	Descriptic
0	2	K_DOWN	Key DOW
0	2	K_LEFT	Key LEFT
0	2	K_RIGHT	Key RIGH
0	2	K_UP	Key UP
0	3	K_DOWN	Key DOW
0	3	K_LEFT	Key LEFT
0	3	K_RIGHT	Key RIGH
0	3	K_UP	Key UP

For more information, refer to *LEDs,*.

3.9.8 Command In

The **Command In** joining wizard can be started manually by selecting **Project→Join Tools→Command In** from the menu.



Command Ins

Command In:

Name

Available Node:

ExpBUS	Node	Hw Name
0	0	M168 G3 544K (ExpBU
0	1	TM168E17 (ExpBUS)
0	2	TM168GDB (ExpBUS)
0	3	TM168GDB (ExpBUS)
255	255	Broadcast

Present Joins:

Name	ExpBUS	Node	Hw Name

The window is divided into 3 lists:

1. **Command In**
2. **Available Node**
3. **Present Joins**

The first list contains the project entities that have not yet been linked to a node from which it receive the command.

The second list contains all the controllers (including expansions) that are added to the project using the **Hardware Expert** wizard, from which you can select to receive the selected command.

Each element in this table has the following characteristics:

- **ExpBUS**: bus to which the controller is connected
- **Node**: controller node from which the command can be accepted
- **Pin**: description of the controller node from which the command can be accepted
- It can take 2 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 **ExpBUS** 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, select the node in the **Node Available** table from which you wish to receive the command and click the **Join** button.

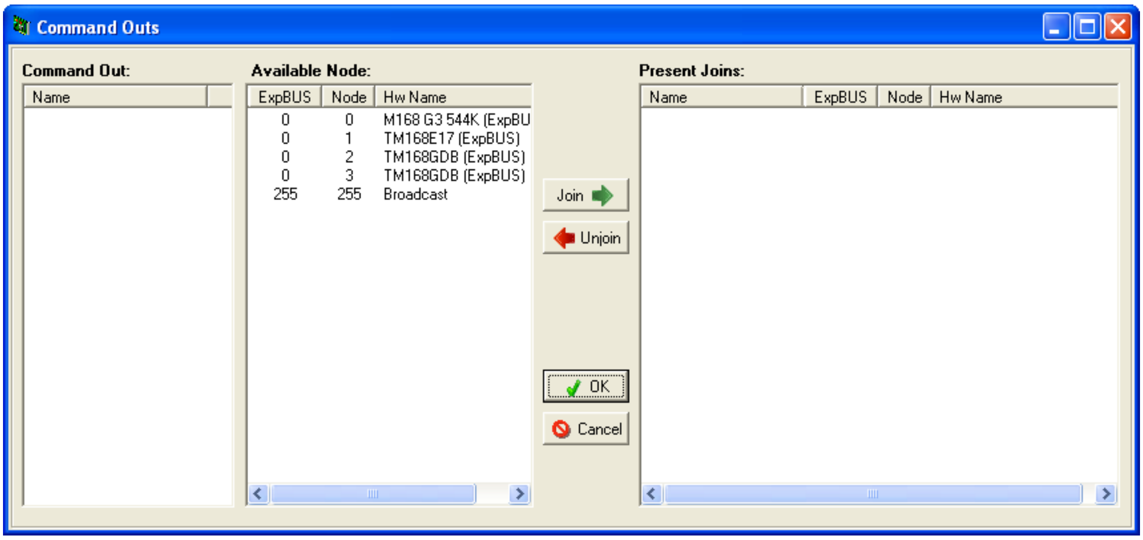
The selected **Command In** disappears from the entity list and a new row is created in the **Joins Present** table.

To remove an association, select the row that represents it in the **Joins Present** table and click **Unjoin**. The table row is eliminated and the **Command In** reappears.

Click the **OK** or **Cancel** buttons to validate or cancel the changes

3.9.9 Command Out

The **Command Out** joining wizard can be started manually by selecting **Project→Join Tools→Command Out** from the menu.



ExpBUS	Node	Hw Name
0	0	M168 G3 544K (ExpBU
0	1	TM168E17 (ExpBUS)
0	2	TM168GDB (ExpBUS)
0	3	TM168GDB (ExpBUS)
255	255	Broadcast

The window is divided into 3 lists:

1. **Command Out**
2. **Available Node**
3. **Present Joins**

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 the controllers (including expansions) that are added to the project using the **Hardware Expert** wizard, to which you may select to send the selected command.

Each element in this table has the following characteristics:

- **ExpBUS**: bus to which the controller is connected
- **Node**: controller node to which the command will be sent
- **Pin**: description of the controller node to which the command will be sent

It can take 2 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 **ExpBUS 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.

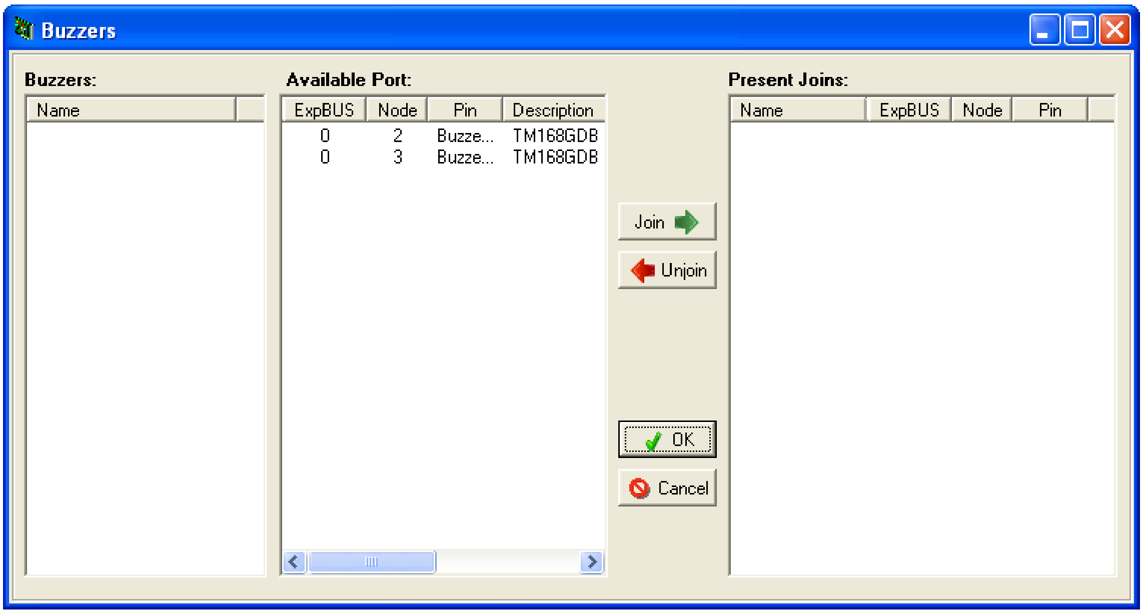
To make a join, select a **Command Out** from the first table, select the node in the Node Available table to which you wish to send the command and click the **Join** button.

The selected **Command Out** disappears from the entity list and a new row is created in the **Joins Present** table.

To remove an association, select the row that represents it in the **Joins Present** table and click **Unjoin**. The table row is eliminated and the **Command Out** reappears.

3.9.10 Buzzers

The buzzer joining wizard can be started manually by selecting **Project→Join Tools →Buzzers** from the menu.



ExpBUS	Node	Pin	Description
0	2	Buzze...	TM168GDB
0	3	Buzze...	TM168GDB

For more information, refer to *LEDs.*

3.10 Compiling the Program

Once you have saved a 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:

- Selecting the **Project→Compile** menu.
- Clicking the icon in the toolbar.

The compilation phase can be divided into 5 sub-phases:

1. Project is automatically saved.
2. The source file is generated.
3. Compilation.
4. Linking.
5. Available resources are calculated.

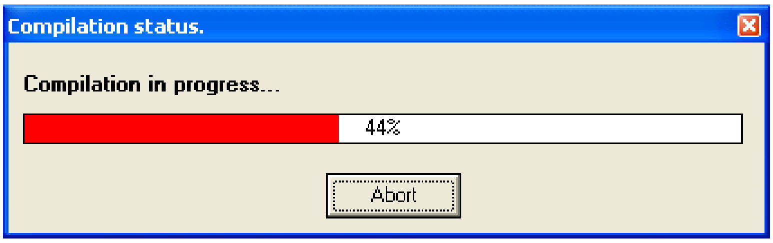
If the project had not been saved yet, you will be prompted to specify a file name to save it to.

The project is analyzed and the source files are created, which are compiled and linked to generate the executable file that is downloaded to the controller through the Download function.

Finally, the available resources like the amount of FLASH and RAM memory unused are calculated.

While source files are being compiled, a window is displayed in the foreground with a progress bar showing the completed process percentage.

An **Abort** button allows you to interrupt the process.

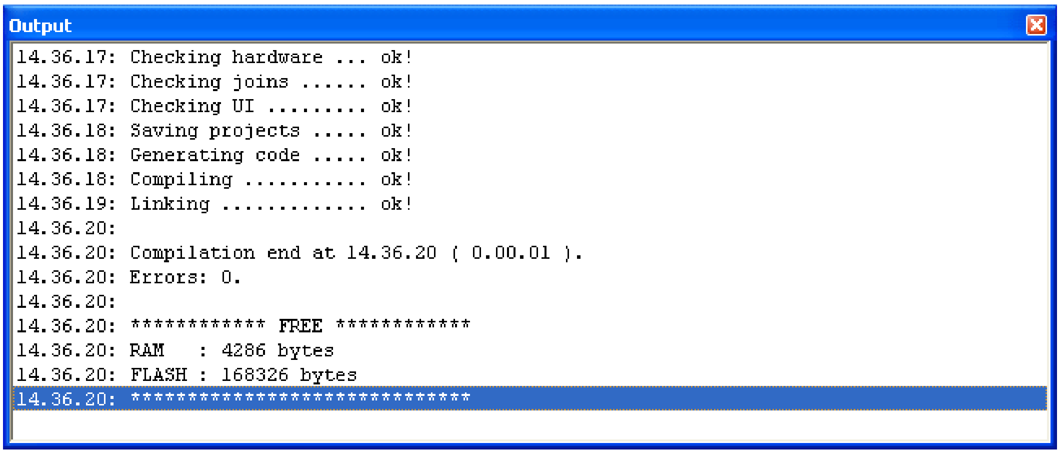


Compilation in progress...

In addition, a window that progressively displays the result of compilation appears at the bottom of the environment.

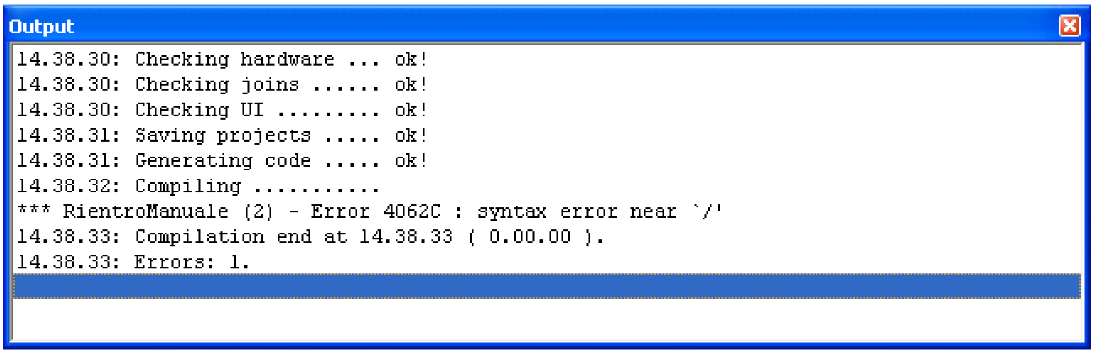
If the compiling process ends successfully, the object files are linked and the executable file to be subsequently downloaded is generated.

A report showing all FLASH and RAM resources available is displayed.



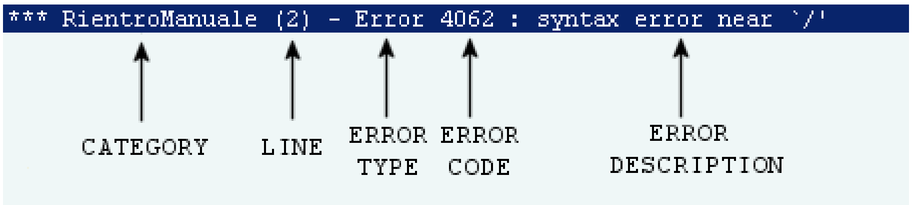
```
14.36.17: Checking hardware ... ok!
14.36.17: Checking joins ...... ok!
14.36.17: Checking UI ......... ok!
14.36.18: Saving projects ..... ok!
14.36.18: Generating code ..... ok!
14.36.18: Compiling ........... ok!
14.36.19: Linking ............. ok!
14.36.20:
14.36.20: Compilation end at 14.36.20 ( 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 error detected causes 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 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**: can be information, advisory note, error detected, unrecoverable error detected
- **ERROR CODE**: code assigned to the error, refer to *Appendix- Compilation Error Message Format*.
- **ERROR DESCRIPTION**: a brief description of the error

Double-click the row corresponding to the event you wish to correct, the **C Algorithm Editor** of the algorithm in which the error was detected opens automatically and the cursor is placed in the relevant line.

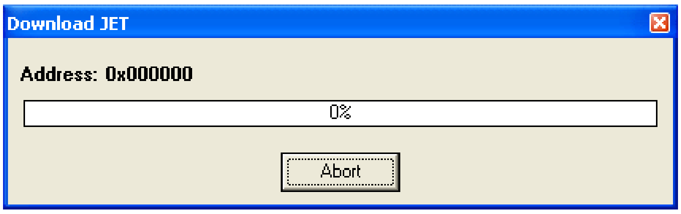
3.11 Downloading the Program

Once you have finished compiling, the executable code can be downloaded into the controller.

Before starting the download, verify that the hardware is properly connected to the serial port and ready for programming (refer to the *Modicon M168 Controller Hardware User Guide*).

To start the program download, select the **Project→Download** menu or the icon located on the toolbar.

The following window appears:



Address: 0x000000

If the connection with the controller is successfully established, after a few seconds the progress bar begins to expand. Once download is complete, a box appears indicating the successful outcome of the operation.

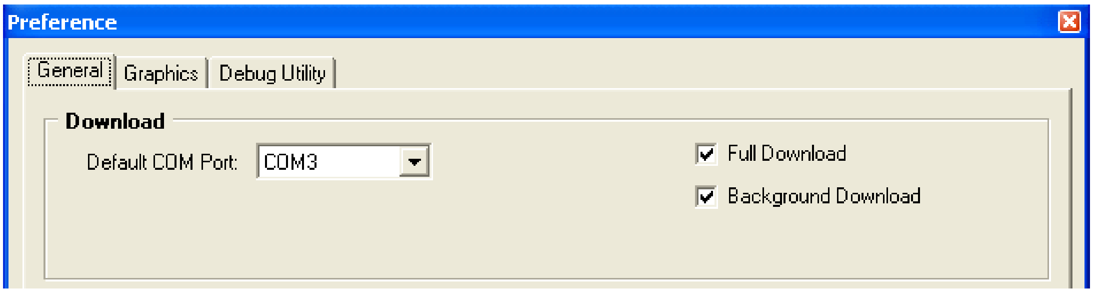
Otherwise an error message appears. In this case reattempt the operation after verifying conditions:

- The serial port selected in the general tab (which can be activated from the **Tools→Preference** menu) must be correct.
- The programming cable must be connected to both the computer and the controller.
- If the bar does not progress and an error message appears, verify the connections, try to reset the controller again and restart downloading.

To change download options, refer to *Preference Menu: Configuration*.

3.11.1 COM Port Configuration

The COM port used for application download and debugger is set on **Menu→Tools→Preference→General**.



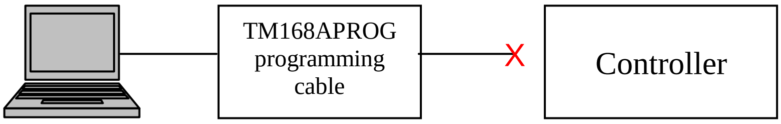
Default COM Port: COM3

☑ Full Download

☑ Background Download

If a controller is connected to this COM port, it will be reset the first time that the controller downloads the application or when the debugger is started. A notification message is shown, to inform the user, that a reset is required. After the reset the user will be notified to connect the programming cable again. To avoid this, the controller must be disconnected from the programming cable.

Connections to avoid controller reset on software start:



INITIALIZATION AFTER COM Port CHANGING

Each time the user changes the Default COM Port in **Tools→Preference→General**, the software makes a COM port initialization once you close the window.

If a controller is connected to the newly selected COM port at the time of the COM port initialization, that controller will be reset. That is, the controller stops executing its application, and all volatile data is re-initialized. In addition, all the outputs will fall back to their default values.

NOTICE

UNINTENDED EQUIPMENT OPERATION

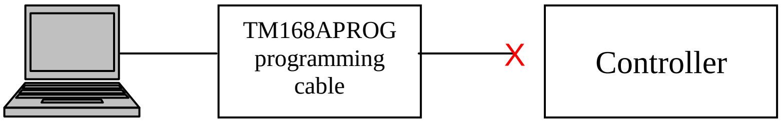
Disconnect the programming cable to your controller before changing the COM Port.

Failure to follow these instructions can result in equipment damage.

NOTE: This behavior occurs whenever there is a COM port initialization, such as when you reboot your PC and restart Windows.

In this case, an advisory message will appear.

Connections to avoid controller reset after COM port changing:

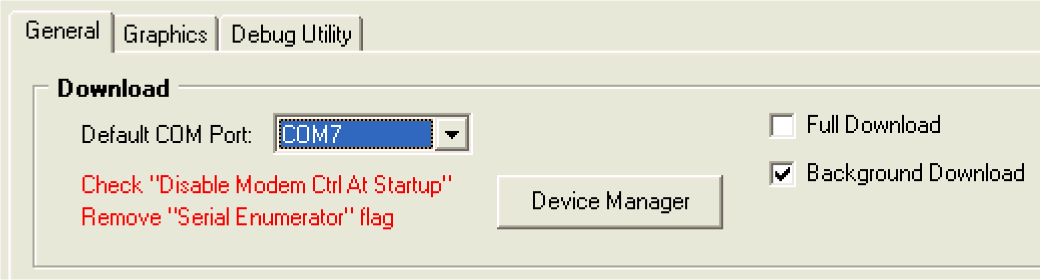


FTDI USB Serial Port Settings

To permit hot cable connection between PC-USB port and controller, the advanced property Disable modem Ctrl At Startup must be checked and the property **Serial Enumerator** must be unchecked.

Entering on **Tools→Preference**, if selected COM port has not this property checked, the following message appears:

If selected COM port has not the **Disable Modem Ctrl At Startup** checked, the following advisory message appears:



Download

Default COM Port: COM7

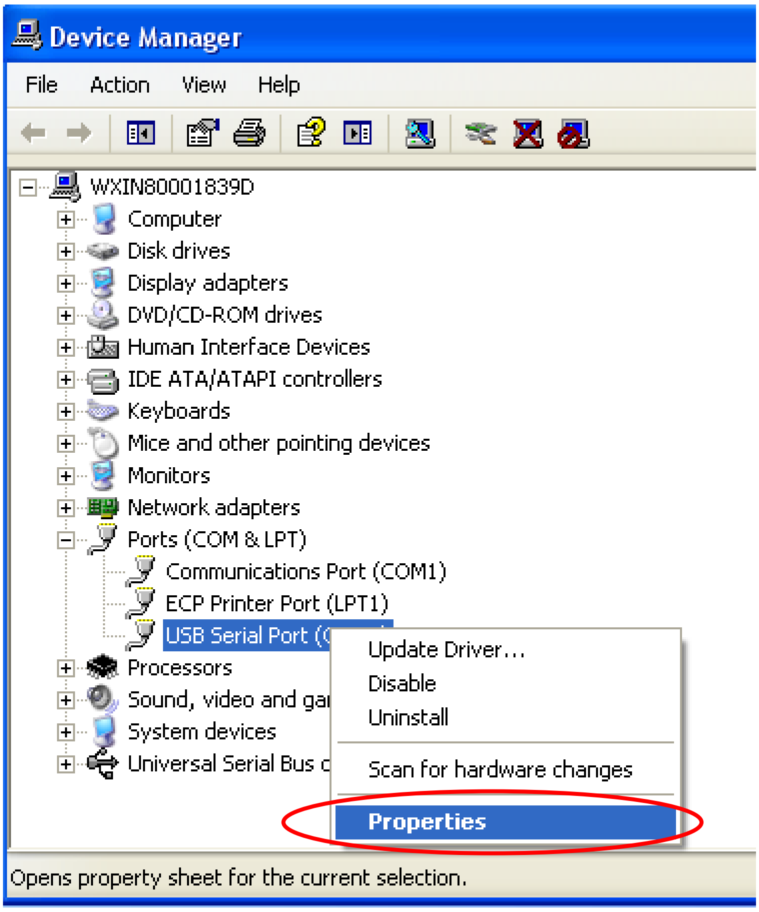
□ Full Download

☑ Background Download

Check "Disable Modem Ctrl At Startup"
Remove "Serial Enumerator" flag

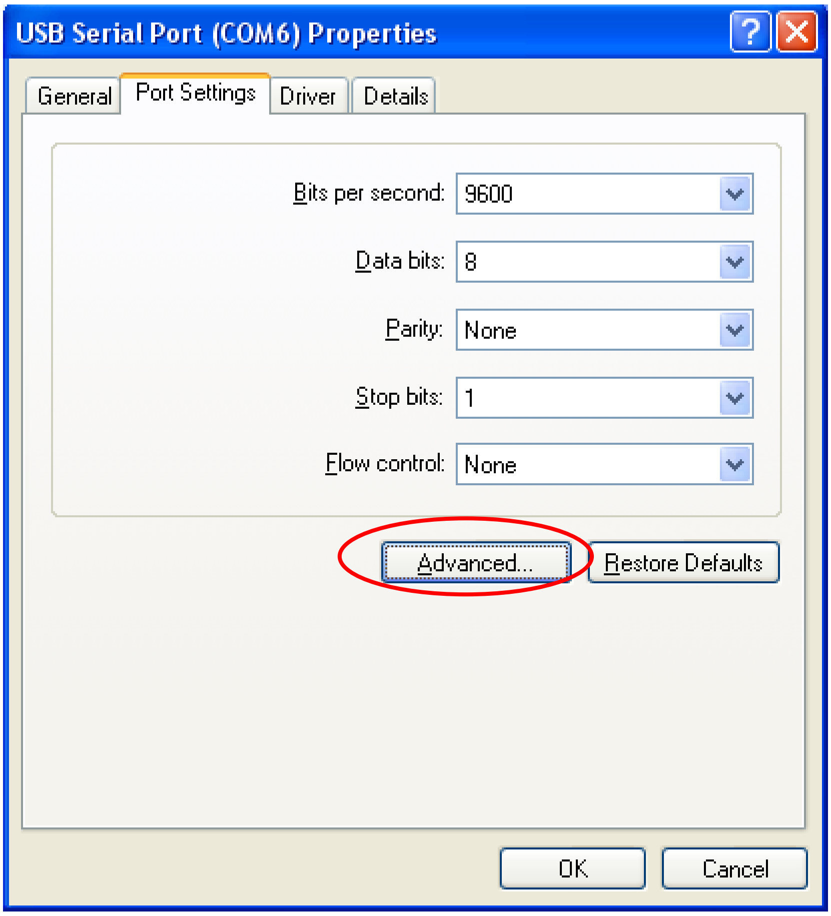
By clicking the **Device Manager** button, the windows **Device Manager** is launched.

Device Manger in Windows:



Here the COM port properties must be selected as shown in the above figure.

COM Port settings:



USB Serial Port (COM6) Properties

Bits per second: 9600

Data bits: 8

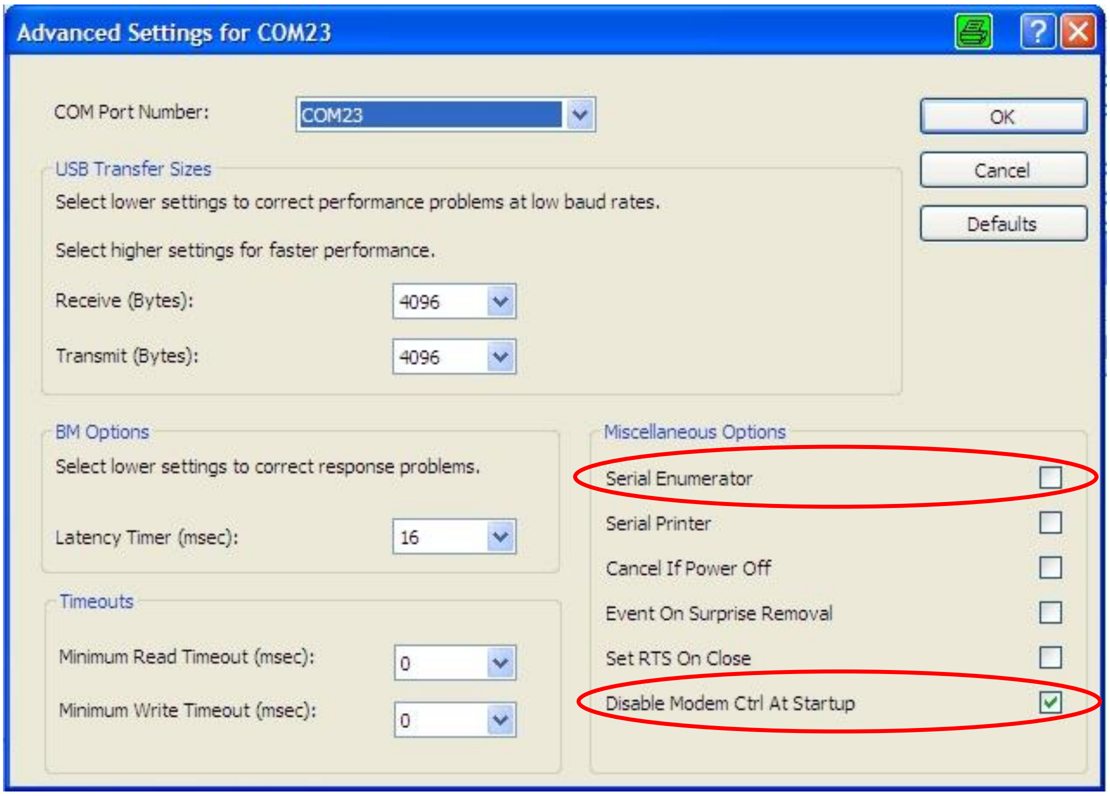
Parity: None

Stop bits: 1

Flow control: None

In COM **Port Settings** tab, **Advanced** button must be clicked.

Advanced COM Port settings:



Advanced Settings for COM23

COM Port Number: COM23

USB Transfer Sizes

Select lower settings to correct performance problems at low baud rates.

Select higher settings for faster performance.

Receive (Bytes): 4096

Transmit (Bytes): 4096

BM Options

Select lower settings to correct response problems.

Latency Timer (msec): 16

Timeouts

Minimum Read Timeout (msec): 0

Minimum Write Timeout (msec): 0

Miscellaneous Options

Serial Enumerator □

Serial Printer □

Cancel If Power Off □

Event On Surprise Removal □

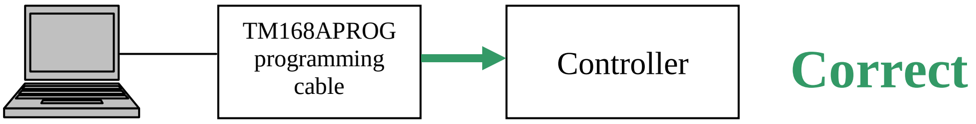
Set RTS On Close □

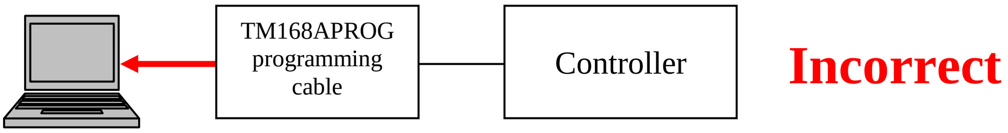
Disable Modem Ctrl At Startup ☑

In Advanced COM port settings, **Disable Modem Ctrl At Startup** must be checked.

NOTE: Each time the programming cable is connected to the USB port, the cable that connects the programmer to the controller must be disconnected.

Correct and Incorrect cable connection sequence:





NOTICE

INOPERABLE EQUIPMENT

Always connect the communication cable to the PC before connecting it to the controller.

Failure to follow these instructions can result in equipment damage.

3.12 Use of the Debugger

The debugger allows you to verify the behavior 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 later operation is used during testing and fine-tuning the machine control.

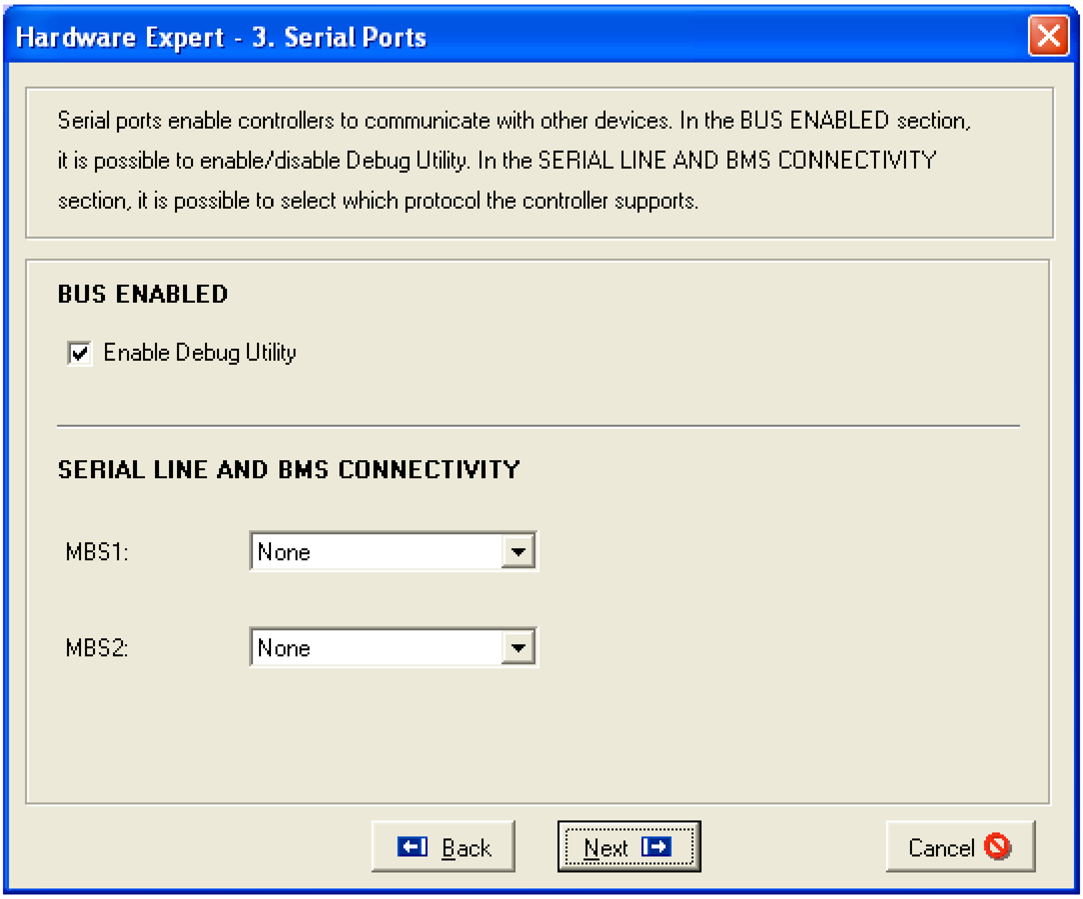
Using the debugger, the status of all internal (variables, timers, and parameters) and external (inputs and outputs) entities used for development can be monitored directly from the graphic entity.

The parameter values and the status can be modified, as well as simulating analog and digital inputs, interrupting the program through 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 is selected.

This operation is performed in the **Hardware Expert** during the serial port protocol configuration phase.



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 Debug Utility. In the SERIAL LINE AND BMS CONNECTIVITY section, it is possible to select which protocol the controller supports.

BUS ENABLED

☑ Enable Debug Utility

SERIAL LINE AND BMS CONNECTIVITY

MBS1: None

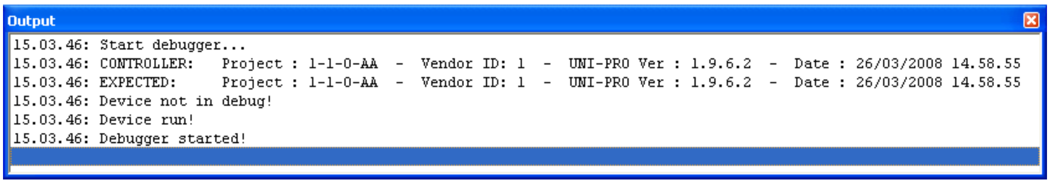
MBS2: None

To activate a debug session, select **Debug→Start Debugger** from the menu, or the D icon located on the tool bar.

3.12.1 Start Debug

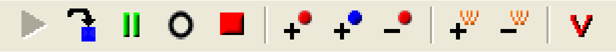
The program connects to the controller and control if the program in the controller is identical to the program in SoHVAC.

If the 2 programs are identical, the Debugger started! message appears in the **Output** window.

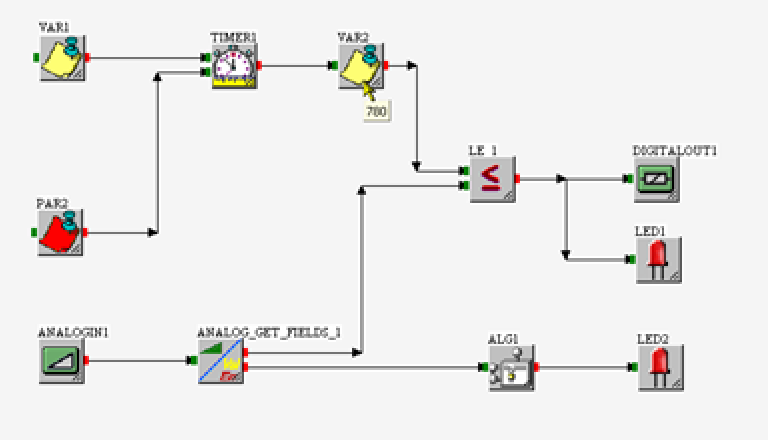


```
15.03.46: Start debugger...
15.03.46: CONTROLLER:   Project : 1-1-0-AA  -  Vendor ID: 1  -  UNI-PRO Ver : 1.9.6.2  -  Date : 26/03/2008 14.58.55
15.03.46: EXPECTED:     Project : 1-1-0-AA  -  Vendor ID: 1  -  UNI-PRO Ver : 1.9.6.2  -  Date : 26/03/2008 14.58.55
15.03.46: Device not in debug!
15.03.46: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

And the following icons appear on the command bar:

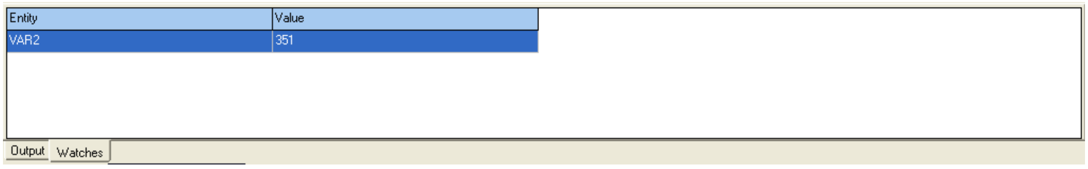


At this point, the values of the program entity can be viewed simply by running the mouse pointer over the variables and waiting for the yellow tool tip to appear.



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 appears on top of the status window:



Entity	Value
VAR2	351

Up to 4 variables can be added to the **Watch** window. To remove the variables, 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
- SoHVAC software version
- Compile date
- Number of entities in the project
- Number of tasks in the project

Project number, Number of entities and Number of tasks must coincide. Otherwise the debugger cannot start.

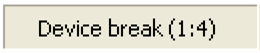
If the other items do not coincide, an advisory message is displayed and the debug session is activated.

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

3.12.2 Run/GoTo/Break/Reset

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**:

In the sample shown, program execution stopped at the calculation of the **DIGITALOUT1**entity, corresponding to task number 4.

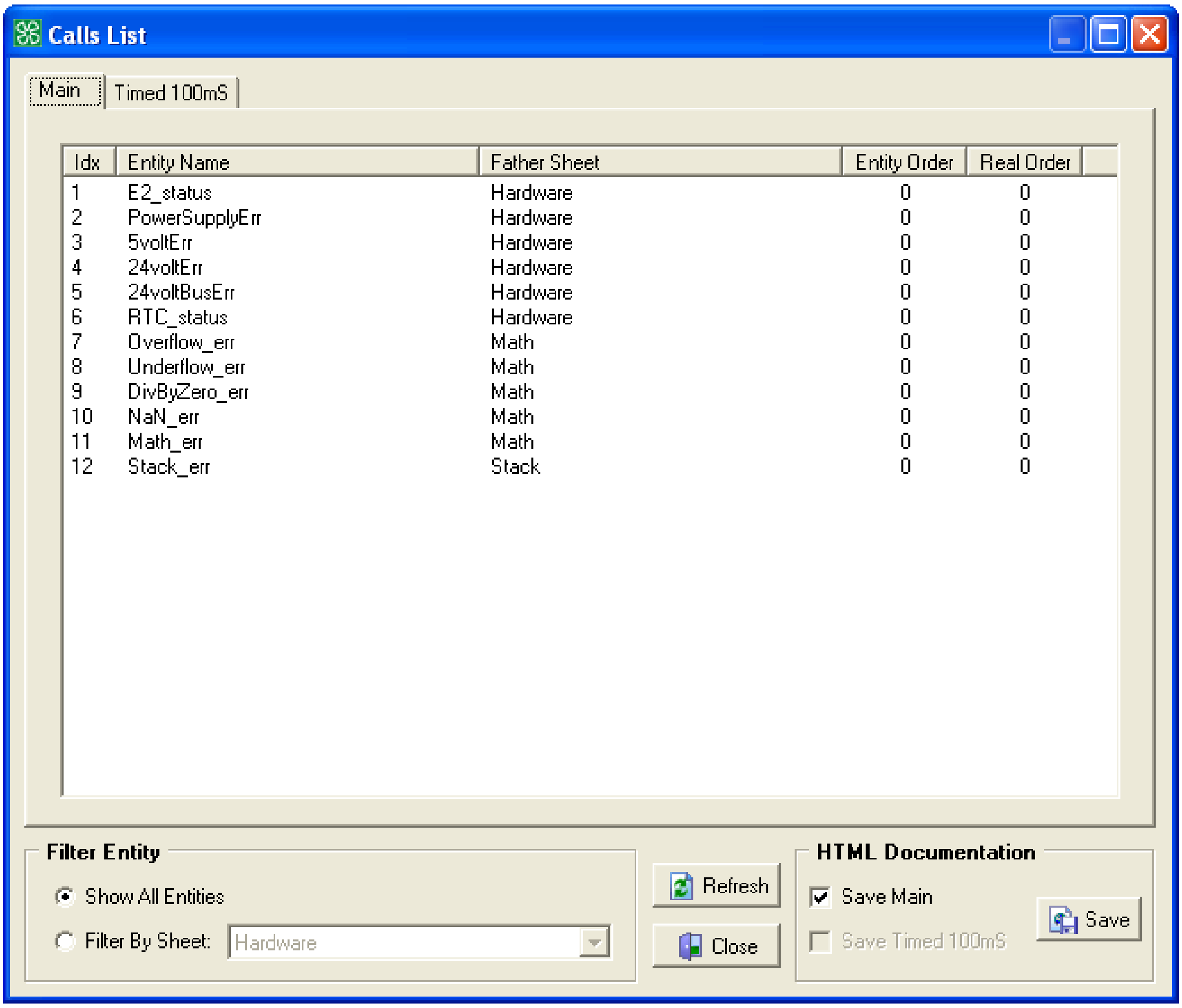
When the program is stopped, all the internal counters and timers are frozen but not the inputs or Real Time Clock. It is still possible to observe the value of the entity or to use the watch window.

There are 2 possible methods to restart the program:

1. The first is to execute a portion of the program until another entity is calculated 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 is performed after which the program once again stops.
2. 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 continues execution starting from the point where it was stopped.

If you want to restart the program from the beginning, it is necessary to perform a controller reset using the **Debug→Reset** command or the ○ icon located on the tool bar.

Thereafter it can be restarted with the **Run** command. The reset operation zeroes all of the internal variables and timers/counters in exactly the same manner as a reset after a power outage.



Idx	Entity Name	Father Sheet	Entity Order	Real Order
1	E2_status	Hardware	0	0
2	PowerSupplyErr	Hardware	0	0
3	5voltErr	Hardware	0	0
4	24voltErr	Hardware	0	0
5	24voltBusErr	Hardware	0	0
6	RTC_status	Hardware	0	0
7	Overflow_err	Math	0	0
8	Underflow_err	Math	0	0
9	DivByZero_err	Math	0	0
10	NaN_err	Math	0	0
11	Math_err	Math	0	0
12	Stack_err	Stack	0	0

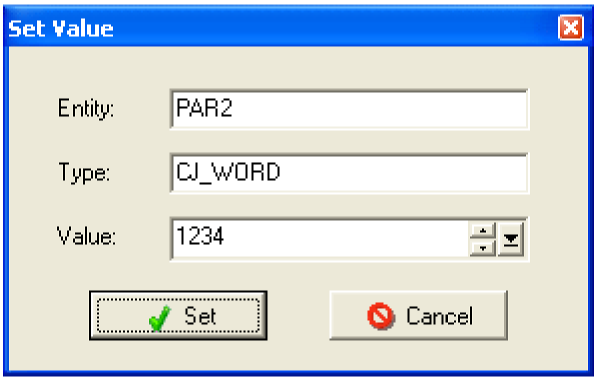
Filter Entity
● Show All Entities
○ Filter By Sheet: Hardware

HTML Documentation
☑ Save Main
□ Save Timed 100mS

3.12.3 Set Value

Both when the program is running and not running, it is possible to set the value of the entity selected using the **Debug→Set Value** or the V icon located on the tool bar.

A window appears where a new value can be set.



Entity: PAR2

Type: CJ_WORD

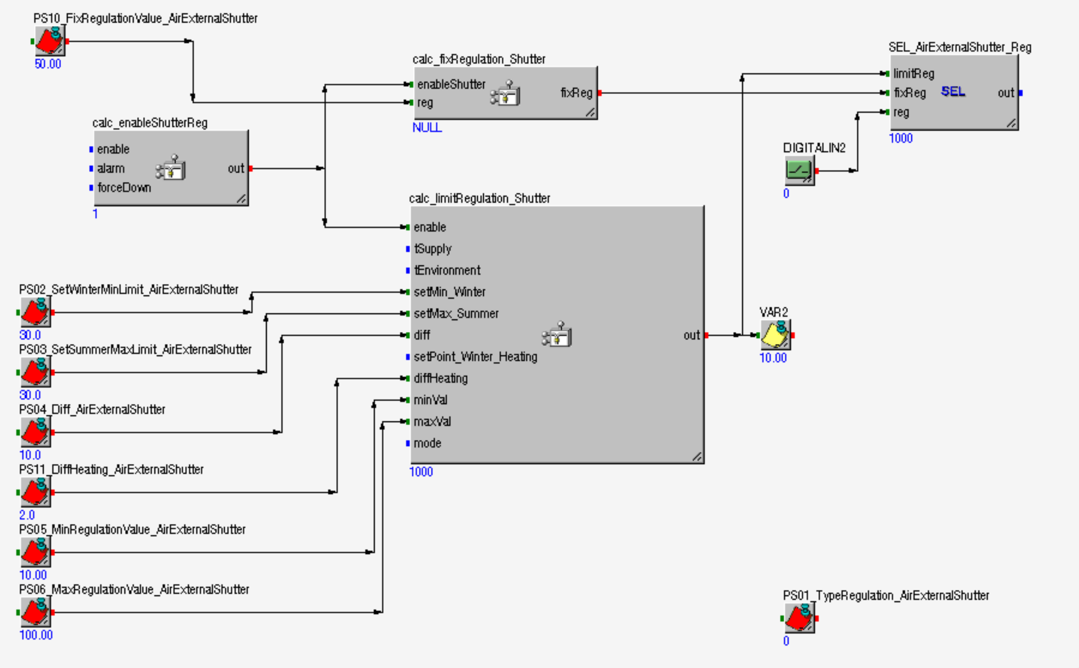
Value: 1234

The values of the analog and digital inputs can be set. In this case, once the input is set, the logical input values are no longer updated with the value of the physical inputs. Effectively, this is forcing the input value until you exit the debugger mode. After exiting, the logical input values are updated with their associated physical values.

3.12.4 Key-in Present Value

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
forceDown
out
1
DIGITALIN2
0
calc_limitRegulation_Shutter
enable
tSupply
tEnvironment
setMin_Winter
setMax_Summer
diff
setPoint_Winter_Heating
diffHeating
minVal
maxVal
mode
out
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 analog input is forced in debug at a determined value, disconnecting therefore from the physical acquisition, it is highlighted with a different color.

The color of the label can be changed in the Graphics section under the

Tools→Preference menu→**Debugger** tab.

3.12.5 Breakpoints

To perform a debug session where it is necessary to investigate the causes of a possible malfunction, it is very important to insert breakpoints in a number of points throughout the program.

When these conditions occur, it is also possible to create breakpoints with break conditions to immediately stop the controller program.

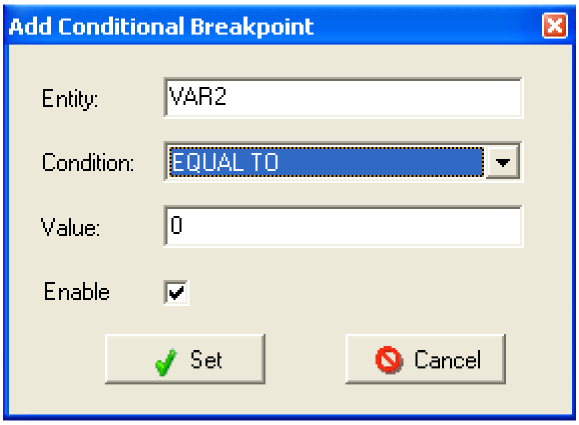
The breakpoint can also be set without conditions, so that the program can stop exactly in the point of execution where the breakpoint was located.

To insert an unconditional breakpoint for a specific entity, it must be selected and the **Debug→Set Breakpoint** command must be activated, or click the +● icon located on the tool menu.

This type is used to verify the behavior of the program within a single main loop or between main loops, making the program stop at each entity where a breakpoint is set.

However 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 opens where you can select the condition to compare with the value of the controlled entity.



Add Conditional Breakpoint

Entity: VAR2

Condition: EQUAL TO

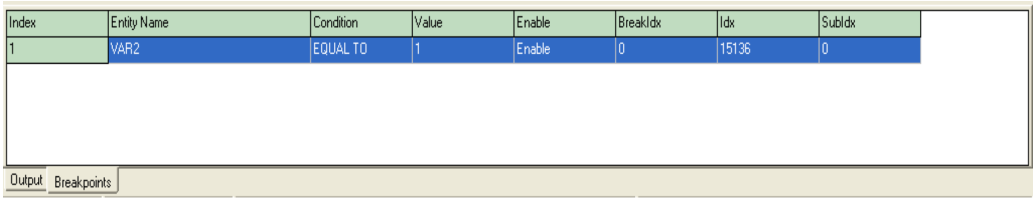
Value: 0

Enable ☑

The conditions can be selected from the following list:

Condition	Description
EXECUTE	No conditions (equivalent to an unconditional breakpoint)
EQUAL TO	The program stops with the entity assumes the value set in the Value field.
DIFFERENT FROM	The program stops with the entity assumes a different value than the one set in the Value field.
LESS THAN	The program stops with the entity assumes a value less than the one set in the Value field.
LESS OR EQUAL	The program stops with the entity assumes a value less than or equal to the one set in the Value field.
MORE THAN	The program stops with the entity assumes a value greater than the one set in the Value field.
MORE OR EQUAL	The program stops 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 appears on top of the status window.



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

In this window, up to 4 breakpoint conditions can be set. The **Condition**, **Value**, and **Enable** properties can be modified in this window.

To remove a breakpoint, remove each breakpoint individually using **Remove Breakpoint** command available from the special menu (right mouse key), 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	Meaning
0	No event
1	Button pressed or command intercepted

NOTE: In the event that the output of an algorithm or a library has more than one branch, the development system performs the intermediate calculation for this output. Consequently, during the debug phase a breakpoint can be set for this, exactly as if it is a hidden variable. However, it is not possible to perform a Set Value on it.

3.12.6 Limitation during Debug

It is normal to see a cycle time increase during debug.

3.13 Algorithm Simulator

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Simulation is not enough for verification/validation test. Commissioning test are needed.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

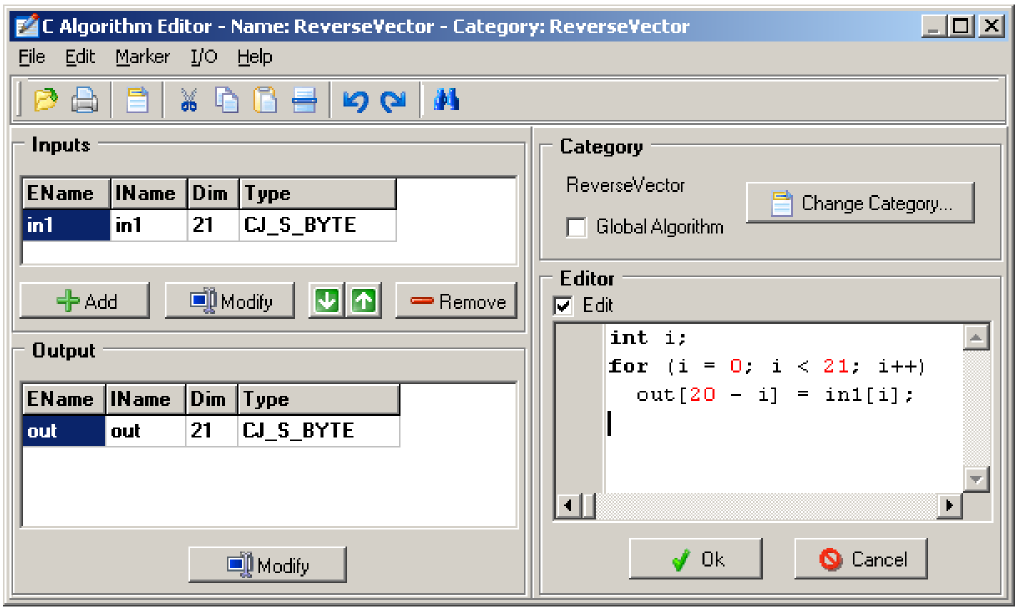
The algorithm simulator is a tool in SoHVAC to simulate the operation of a c-function. It allows the user to test the operation without having a M168 controller.

The main functions of the algorithm simulator are:

- Verify syntax
- Set input values for the simulator
- Execute or step over simulation
- Feedback output to an input
- Set breakpoints and conditional breakpoint
- Display local variables and function result

The following example is using an algorithm, which is called **ReverseVector**. An input array is copied to an output array and array values are in reverse order.

The algorithm will look in SoHVAC as displayed in the following figure:



Inputs

EName	IName	Dim	Type
in1	in1	21	CJ_S_BYTE

Output

EName	IName	Dim	Type
out	out	21	CJ_S_BYTE

Category

ReverseVector

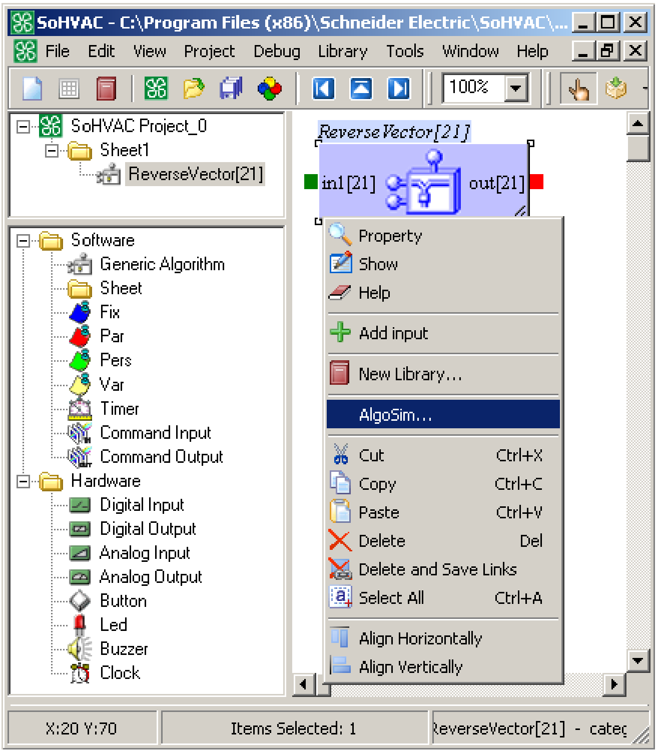
□ Global Algorithm

Editor

☑ Edit

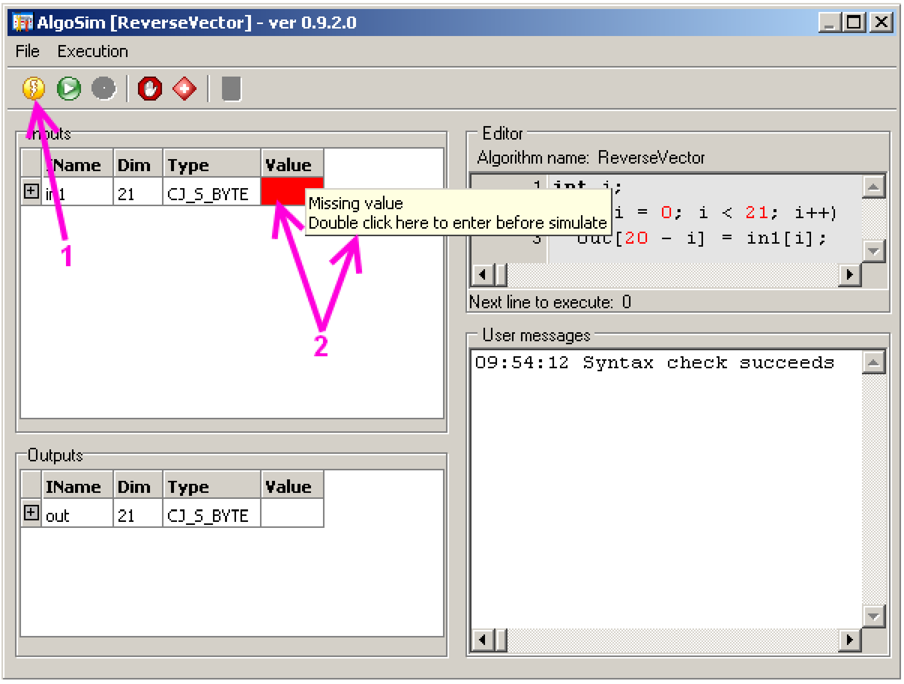
```
int i;
for (i = 0; i < 21; i++)
   out[20 - i] = in1[i];
```

To start the algorithm simulator, click on the algorithm and open the context menu using a right mouse click and select **AlgoSim...**



The algorithm simulator window opens displaying the same input and output declarations as well as the c-code.

As soon as the simulator is started, it performs a c-code syntax and the result is shown in the **User messages** window.


AlgoSim [ReverseVector] - ver 0.9.2.0
File Execution
Inputs
Outputs
Editor
1
2
Missing value
Double click here to enter before simulate

Inputs

Name	Dim	Type	Value
in1	21	CJ_S_BYTE	🟥

Outputs

IName	Dim	Type	Value
out	21	CJ_S_BYTE	

Algorithm name: ReverseVector

```
int i;
    i = 0; i < 21; i++)
  out[20 - i] = in1[i];
```

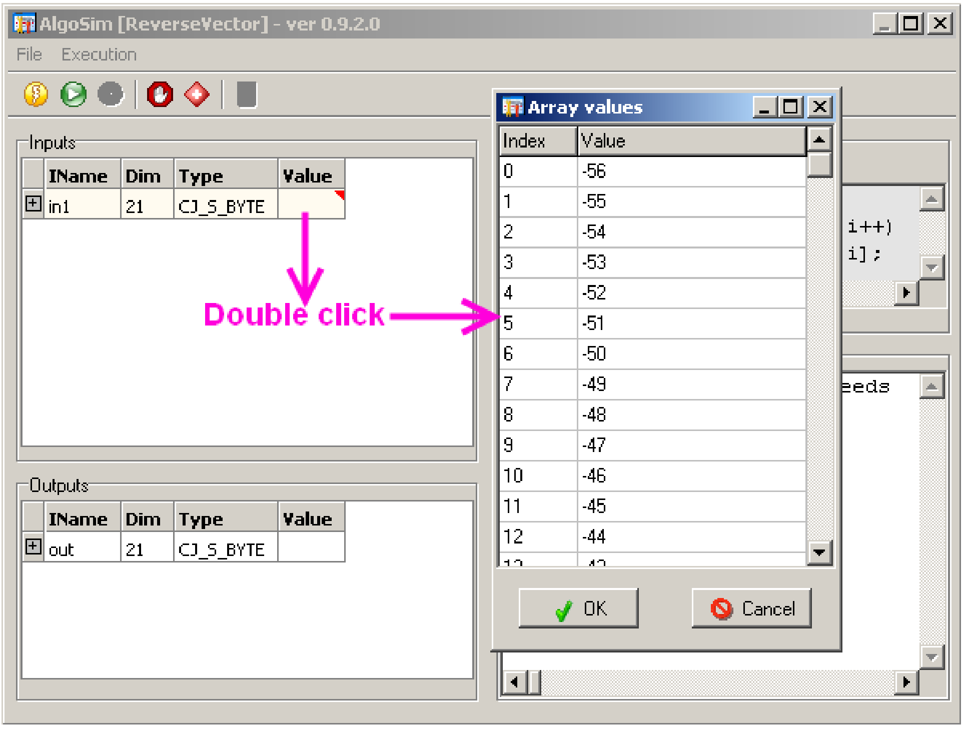
Next line to execute: 0

User messages

```
09:54:12 Syntax check succeeds
```

If the user clicks on the **Run** icon (1), the program highlights a row required to insert a value for each input. If the **Value** cell is empty, it indicates by red colour (2) and a hint suggests to specify a value before simulation can start.

Therefore, double-click on the **Value** cell and fill in the pop-up with desired values.



Inputs

	IName	Dim	Type	Value
⊞	in1	21	CJ_S_BYTE	

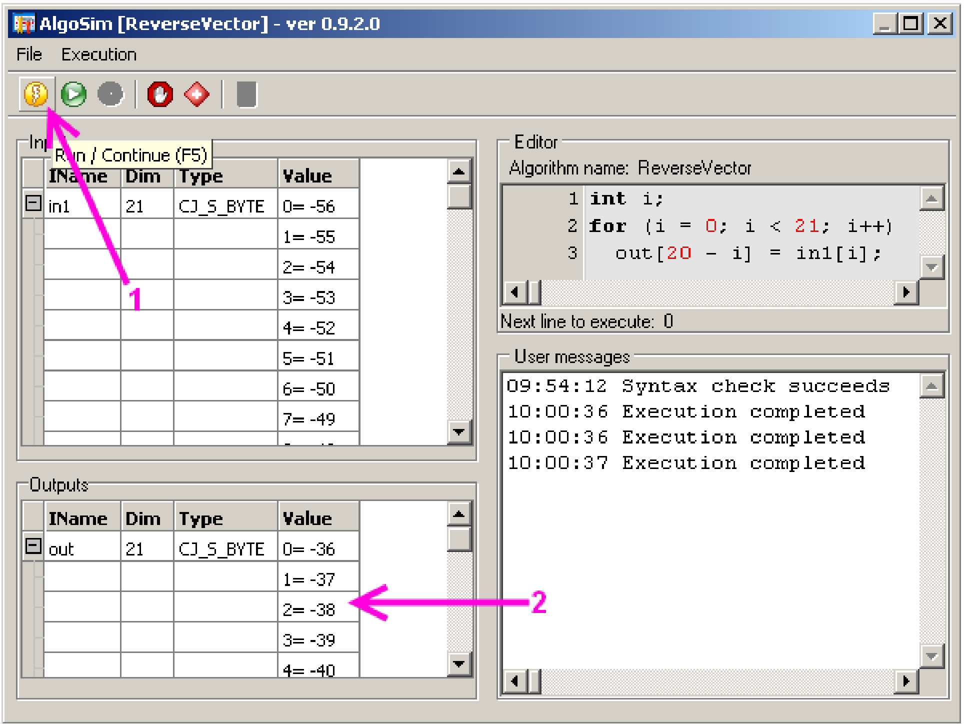
Outputs

	IName	Dim	Type	Value
⊞	out	21	CJ_S_BYTE	

Array values

Index	Value
0	-56
1	-55
2	-54
3	-53
4	-52
5	-51
6	-50
7	-49
8	-48
9	-47
10	-46
11	-45
12	-44

When all input values are defined, click on the **Run** icon (1) to start the simulator. The simulator executes all instructions and the result appears in the **Outputs** pane (2).



Inputs

	IName	Dim	Type	Value
⊟	in1	21	CJ_S_BYTE	0= -56
				1= -55
				2= -54
				3= -53
				4= -52
				5= -51
				6= -50
				7= -49

Algorithm name: ReverseVector

```
int i;
for (i = 0; i < 21; i++)
  out[20 - i] = in1[i];
```

Next line to execute: 0

```
09:54:12 Syntax check succeeds
10:00:36 Execution completed
10:00:36 Execution completed
10:00:37 Execution completed
```

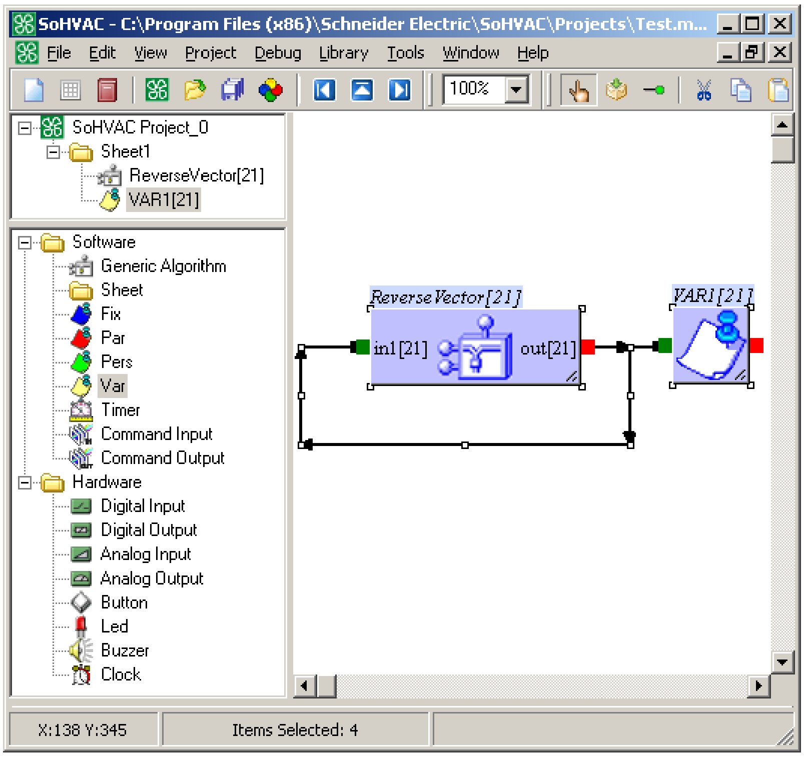
Outputs

	IName	Dim	Type	Value
⊟	out	21	CJ_S_BYTE	0= -36
				1= -37
				2= -38
				3= -39
				4= -40

3.13.1 Feedback Output to an Input

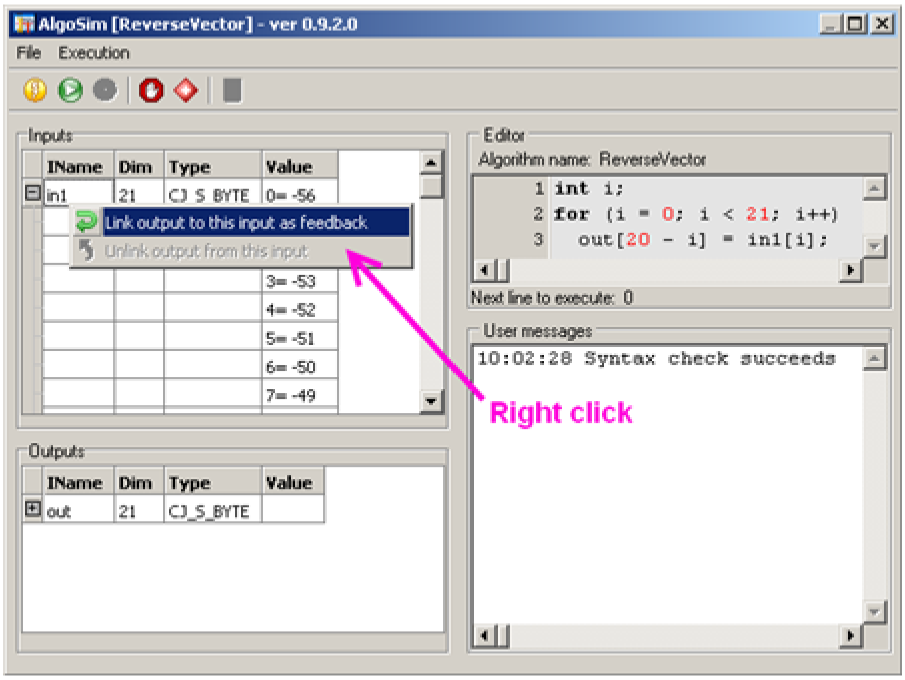
Often, the output of an algorithm is linked to an input an algorithm. This external feedback is supported by the algorithm simulator.

The figure shows the feedback:



To support the feedback in the algorithm simulator, select an input and open the context menu through a mouse right-click. Then, select **Link output to this input as feedback**.

NOTE: The feedback function is only allowed when the input type and output type are identical.


AlgoSim [ReverseVector] - ver 0.9.2.0
File Execution
Inputs
Link output to this input as feedback
Unlink output from this input
Right click
Outputs
Editor
User messages

Inputs

IName	Dim	Type	Value
in1	21	CJ_S_BYTE	0= -56
			3= -53
			4= -52
			5= -51
			6= -50
			7= -49

Outputs

IName	Dim	Type	Value
out	21	CJ_S_BYTE	

Algorithm name: ReverseVector

```
int i;
for (i = 0; i < 21; i++)
  out[20 - i] = in1[i];
```

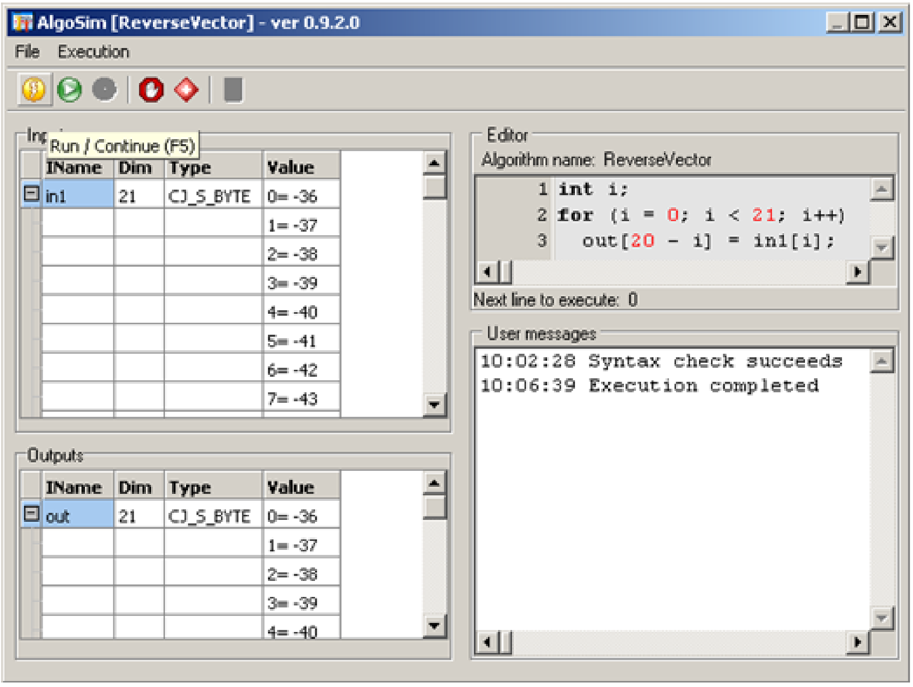
Next line to execute: 0

```
10:02:28 Syntax check succeeds
```

The blue colour indicates the input and output that are linked through the feedback.

The algorithm simulator loads the actual values of input, carries out the algorithm, saves the result in **out** variable, and finally copies it to the linked input parameters for the next execution.

The following figure shows the first execution in which the input **in1**, containing the initial values -56, -55, -54 etc, has been reversed in the output **out** array with the values -36, -37, -38 etc.



IName	Dim	Type	Value
in1	21	CJ_S_BYTE	0= -36
			1= -37
			2= -38
			3= -39
			4= -40
			5= -41
			6= -42
			7= -43

IName	Dim	Type	Value
out	21	CJ_S_BYTE	0= -36
			1= -37
			2= -38
			3= -39
			4= -40

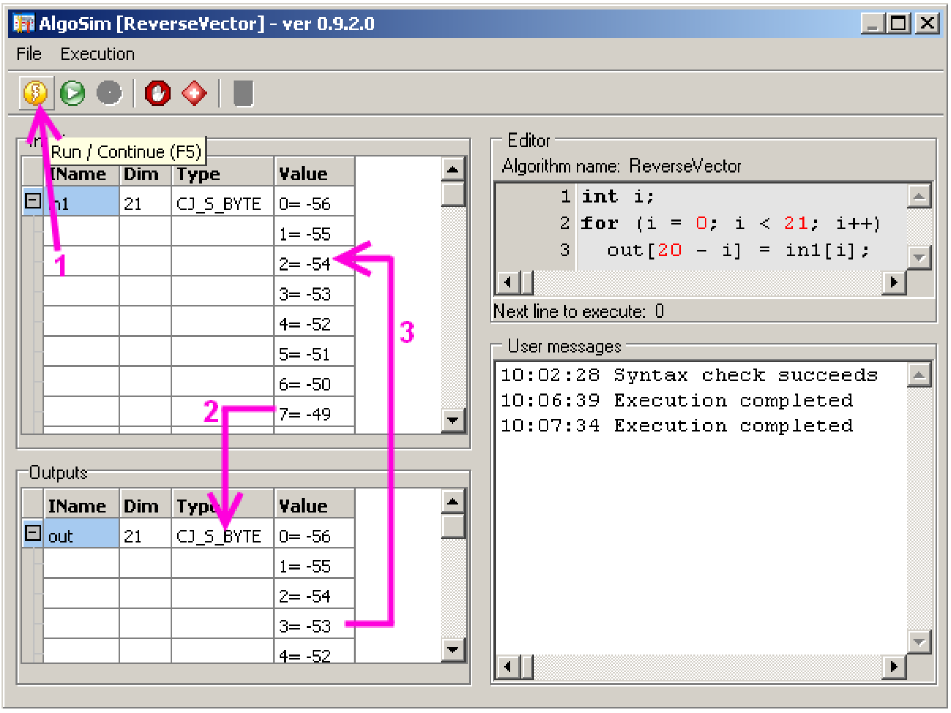
Algorithm name: ReverseVector

```
int i;
for (i = 0; i < 21; i++)
   out[20 - i] = in1[i];
```

Next line to execute: 0

```
10:02:28 Syntax check succeeds
10:06:39 Execution completed
```

Then, the input **in1** is overwritten by the output **out**.



IName	Dim	Type	Value
in1	21	CJ_S_BYTE	0= -56
			1= -55
			2= -54
			3= -53
			4= -52
			5= -51
			6= -50
			7= -49

IName	Dim	Type	Value
out	21	CJ_S_BYTE	0= -56
			1= -55
			2= -54
			3= -53
			4= -52

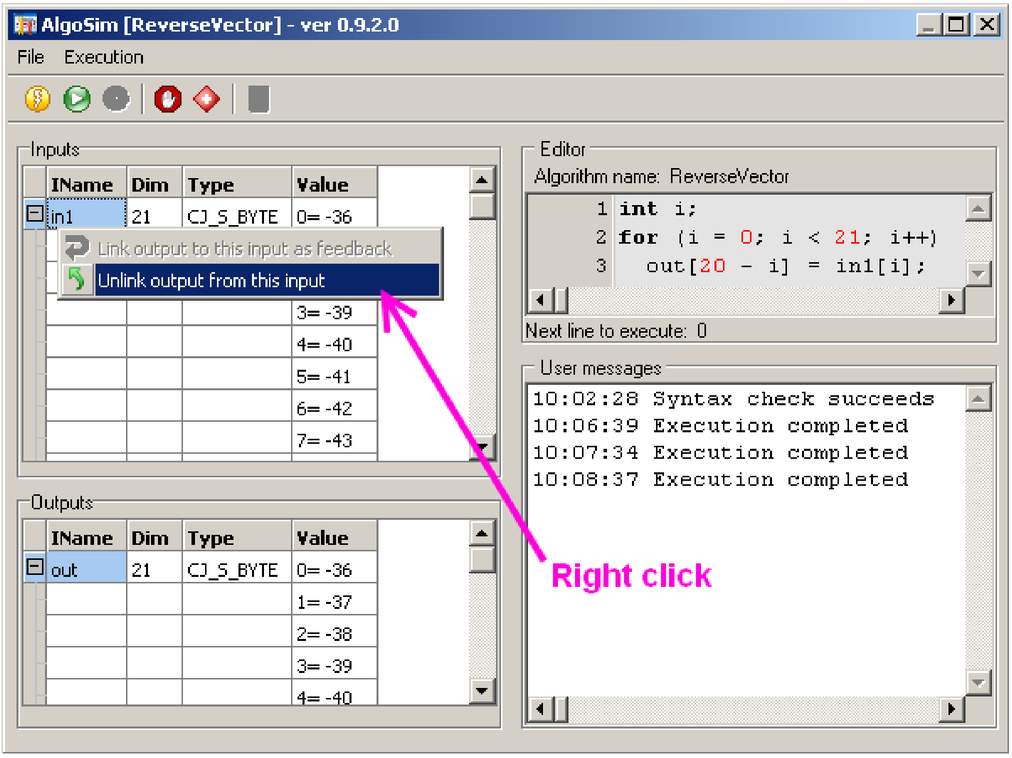
Algorithm name: ReverseVector

```
int i;
for (i = 0; i < 21; i++)
   out[20 - i] = in1[i];
```

Next line to execute: 0

```
10:02:28 Syntax check succeeds
10:06:39 Execution completed
10:07:34 Execution completed
```

To remove the feedback, select the input and open the context menu through a mouse right-click. Then, select **Unlink output from this input**.


AlgoSim [ReverseVector] - ver 0.9.2.0
File
Execution
Inputs
Link output to this input as feedback
Unlink output from this input
Editor
User messages
Outputs
Right click

IName	Dim	Type	Value
in1	21	CJ_S_BYTE	0= -36
			3= -39
			4= -40
			5= -41
			6= -42
			7= -43

Algorithm name: ReverseVector

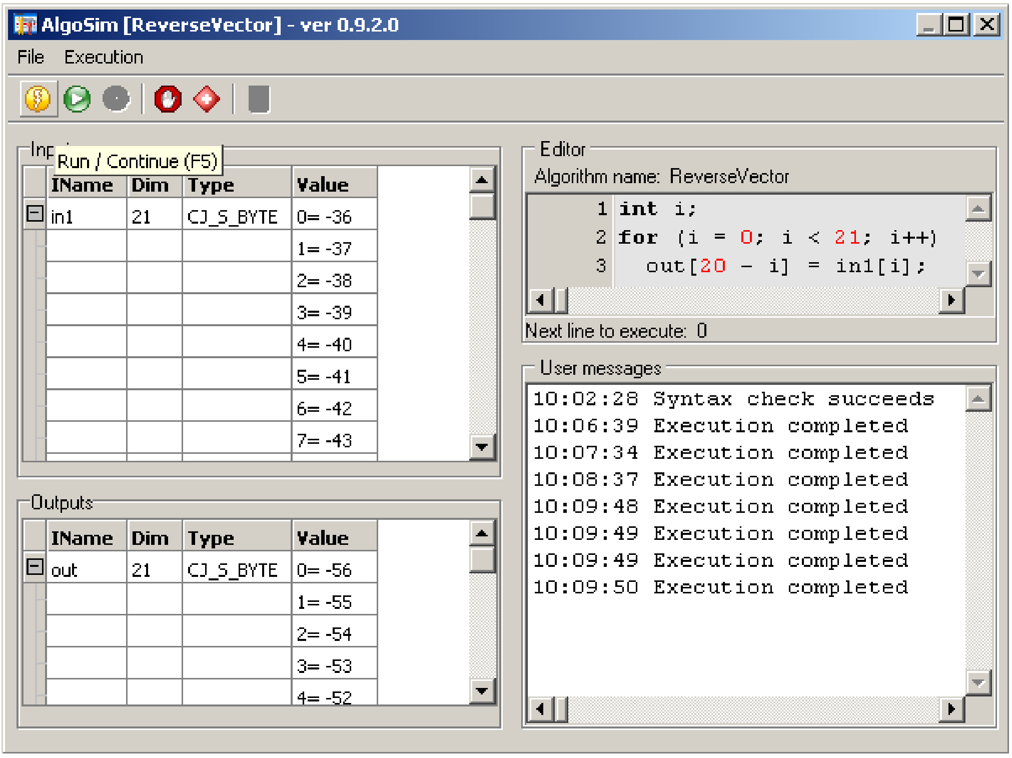
```
int i;
for (i = 0; i < 21; i++)
  out[20 - i] = in1[i];
```

Next line to execute: 0

```
10:02:28 Syntax check succeeds
10:06:39 Execution completed
10:07:34 Execution completed
10:08:37 Execution completed
```

IName	Dim	Type	Value
out	21	CJ_S_BYTE	0= -36
			1= -37
			2= -38
			3= -39
			4= -40

Without feedback, each run generates the same results because the input values remain the same.



IName	Dim	Type	Value
in1	21	CJ_S_BYTE	0= -36
			1= -37
			2= -38
			3= -39
			4= -40
			5= -41
			6= -42
			7= -43

Algorithm name: ReverseVector

```
int i;
for (i = 0; i < 21; i++)
  out[20 - i] = in1[i];
```

Next line to execute: 0

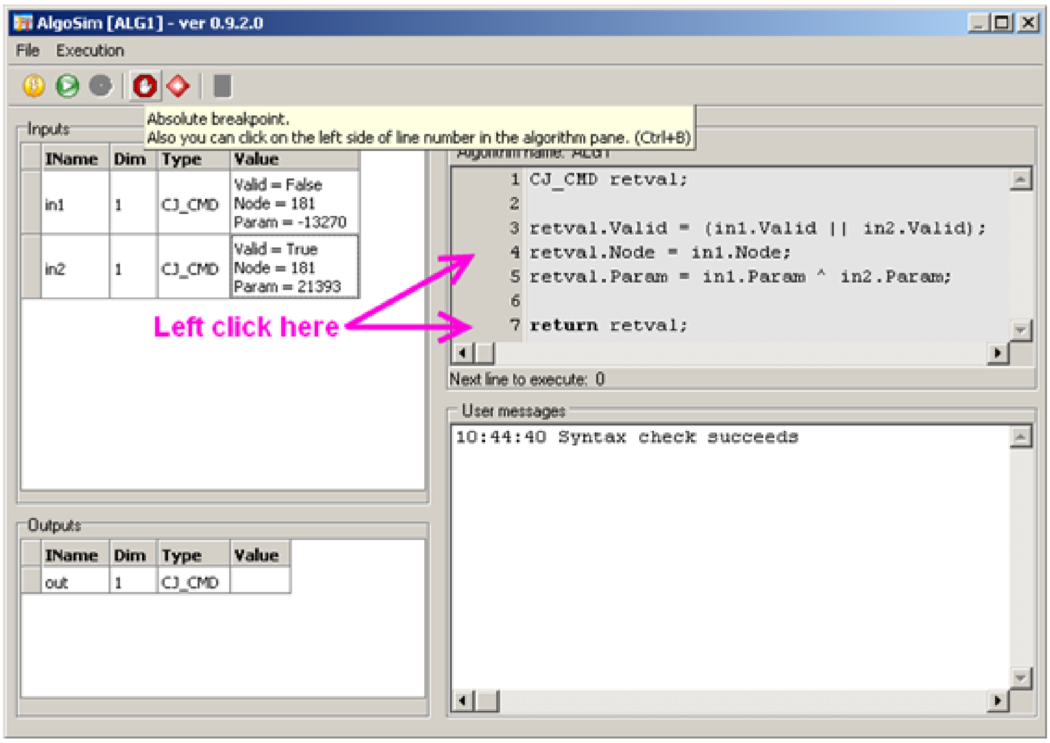
```
10:02:28 Syntax check succeeds
10:06:39 Execution completed
10:07:34 Execution completed
10:08:37 Execution completed
10:09:48 Execution completed
10:09:49 Execution completed
10:09:49 Execution completed
10:09:50 Execution completed
```

IName	Dim	Type	Value
out	21	CJ_S_BYTE	0= -56
			1= -55
			2= -54
			3= -53
			4= -52

3.13.2 Absolute Breakpoint

For testing purposes, it may be helpful to set breakpoints. The execution of the algorithm stops when the algorithm execution reaches the instruction where the breakpoint is set.

The fastest way to add/remove an absolute breakpoint is clicking on the left side of the line.



Absolute breakpoint.
Also you can click on the left side of line number in the algorithm pane. (Ctrl+B)

IName	Dim	Type	Value

in1	1	CJ_CMD	Valid = False
			Node = 181
			Param = -13270

in2	1	CJ_CMD	Valid = True
			Node = 181
			Param = 21393

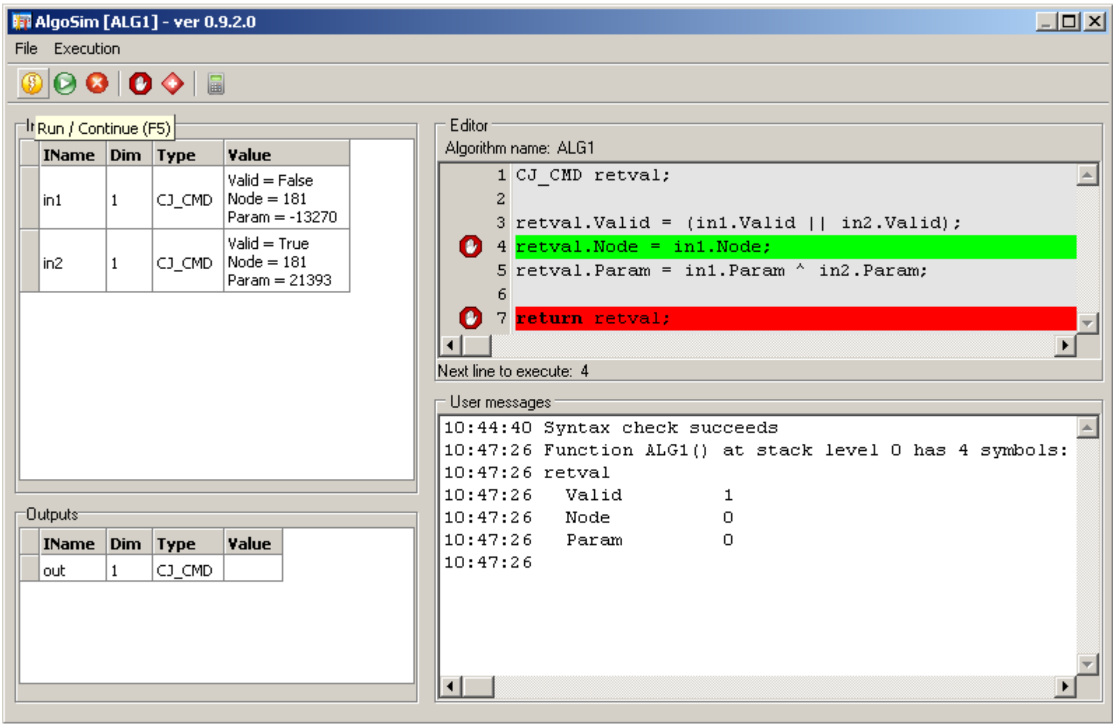
```
1 CJ_CMD retval;
2
3 retval.Valid = (in1.Valid || in2.Valid);
4 retval.Node = in1.Node;
5 retval.Param = in1.Param ^ in2.Param;
6
7 return retval;
```

Next line to execute: 0

```
10:44:40 Syntax check succeeds
```

IName	Dim	Type	Value
out	1	CJ_CMD	

When the breakpoint is reached, the line is highlighted in green.



Algorithm name: ALG1

IName	Dim	Type	Value

in1	1	CJ_CMD	Valid = False
			Node = 181
			Param = -13270

in2	1	CJ_CMD	Valid = True
			Node = 181
			Param = 21393

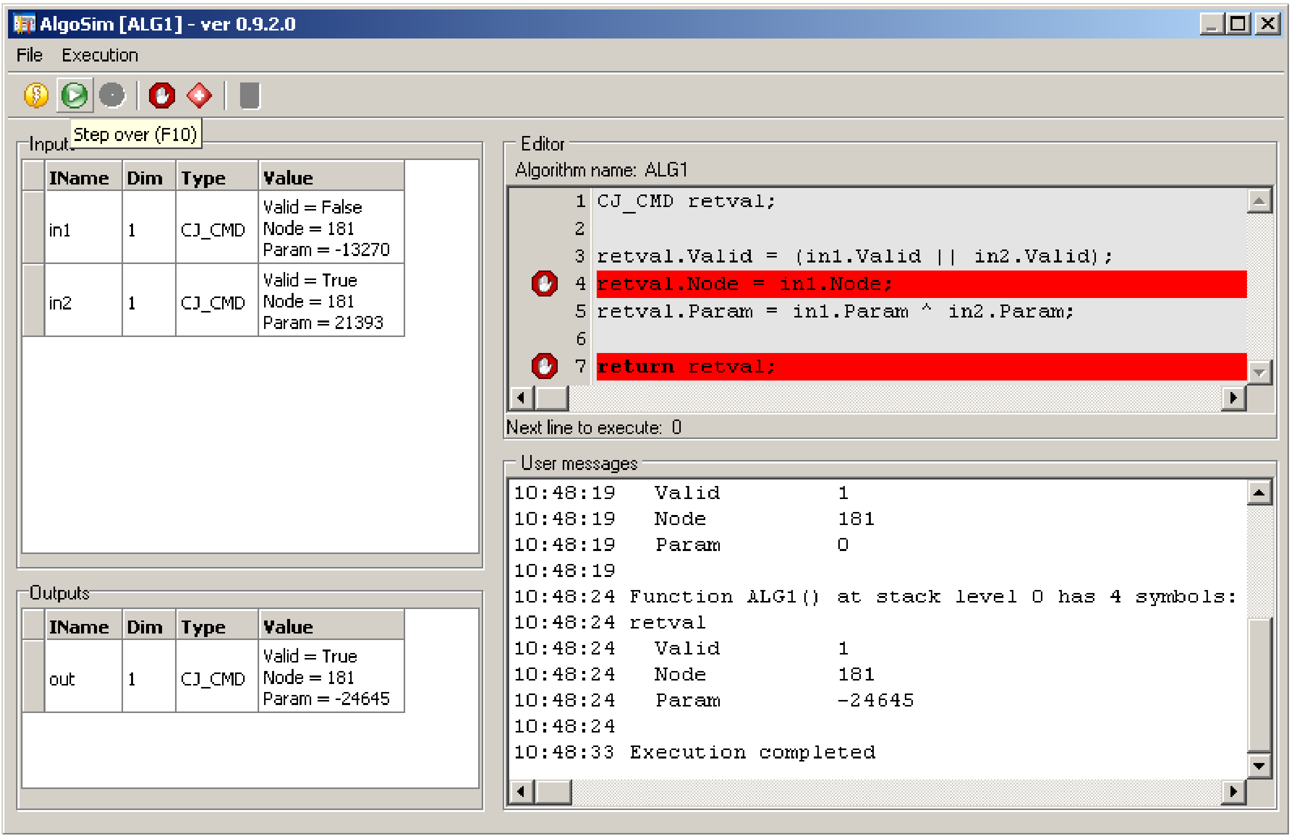
```
  1 CJ_CMD retval;
  2
  3 retval.Valid = (in1.Valid || in2.Valid);
● 4 retval.Node = in1.Node;
  5 retval.Param = in1.Param ^ in2.Param;
  6
● 7 return retval;
```

Next line to execute: 4

```
10:44:40 Syntax check succeeds
10:47:26 Function ALG1() at stack level 0 has 4 symbols:
10:47:26 retval
10:47:26   Valid        1
10:47:26   Node         0
10:47:26   Param        0
10:47:26
```

IName	Dim	Type	Value
out	1	CJ_CMD	

The **Step over** icon now allows executing the algorithm step by step. The results of each instruction are presented in the **User messages** window.



Inputs

IName	Dim	Type	Value

in1	1	CJ_CMD	Valid = False
			Node = 181
			Param = -13270

in2	1	CJ_CMD	Valid = True
			Node = 181
			Param = 21393

Outputs

IName	Dim	Type	Value

out	1	CJ_CMD	Valid = True
			Node = 181
			Param = -24645

Algorithm name: ALG1

```
CJ_CMD retval;

retval.Valid = (in1.Valid || in2.Valid);
retval.Node = in1.Node;
retval.Param = in1.Param ^ in2.Param;

return retval;
```

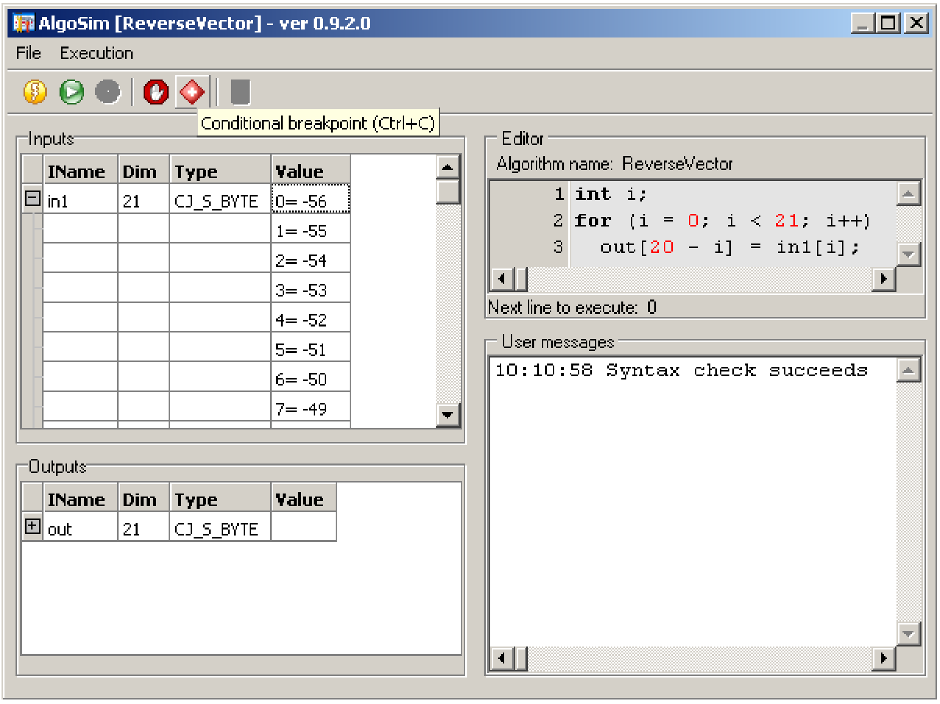
Next line to execute: 0

```
10:48:19   Valid          1
10:48:19   Node           181
10:48:19   Param          0
10:48:19
10:48:24 Function ALG1() at stack level 0 has 4 symbols:
10:48:24 retval
10:48:24   Valid          1
10:48:24   Node           181
10:48:24   Param          -24645
10:48:24
10:48:33 Execution completed
```

3.13.3 Conditional Breakpoint

For testing purposes, it may be helpful to set conditional breakpoints. The execution of the algorithm stops when the conditional breakpoint conditions are satisfied.

Click the **Conditional breakpoint** icon to create a condition breakpoint.


AlgoSim [ReverseVector] - ver 0.9.2.0
File
Execution
Conditional breakpoint (Ctrl+C)
Inputs
Outputs
Editor
User messages

Inputs

IName	Dim	Type	Value
in1	21	CJ_S_BYTE	0= -56
			1= -55
			2= -54
			3= -53
			4= -52
			5= -51
			6= -50
			7= -49

Outputs

IName	Dim	Type	Value
out	21	CJ_S_BYTE	

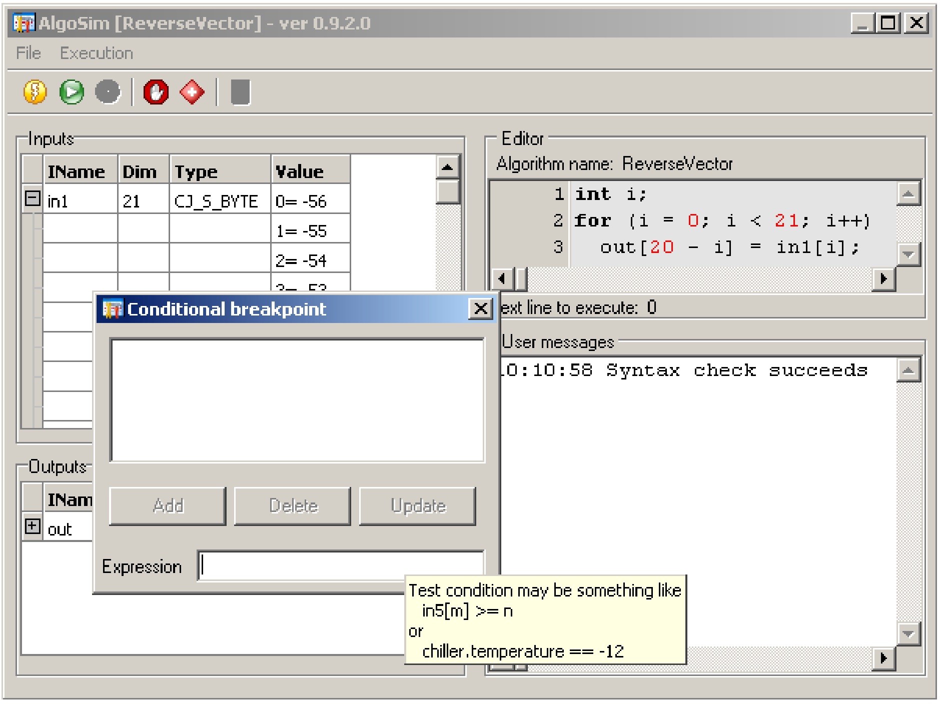
Algorithm name: ReverseVector

```
int i;
for (i = 0; i < 21; i++)
  out[20 - i] = in1[i];
```

Next line to execute: 0

```
10:10:58 Syntax check succeeds
```

The **Conditional breakpoint** window pops up.



Inputs

IName	Dim	Type	Value
in1	21	CJ_S_BYTE	0= -56
			1= -55
			2= -54

Algorithm name: ReverseVector

```
int i;
for (i = 0; i < 21; i++)
  out[20 - i] = in1[i];
```

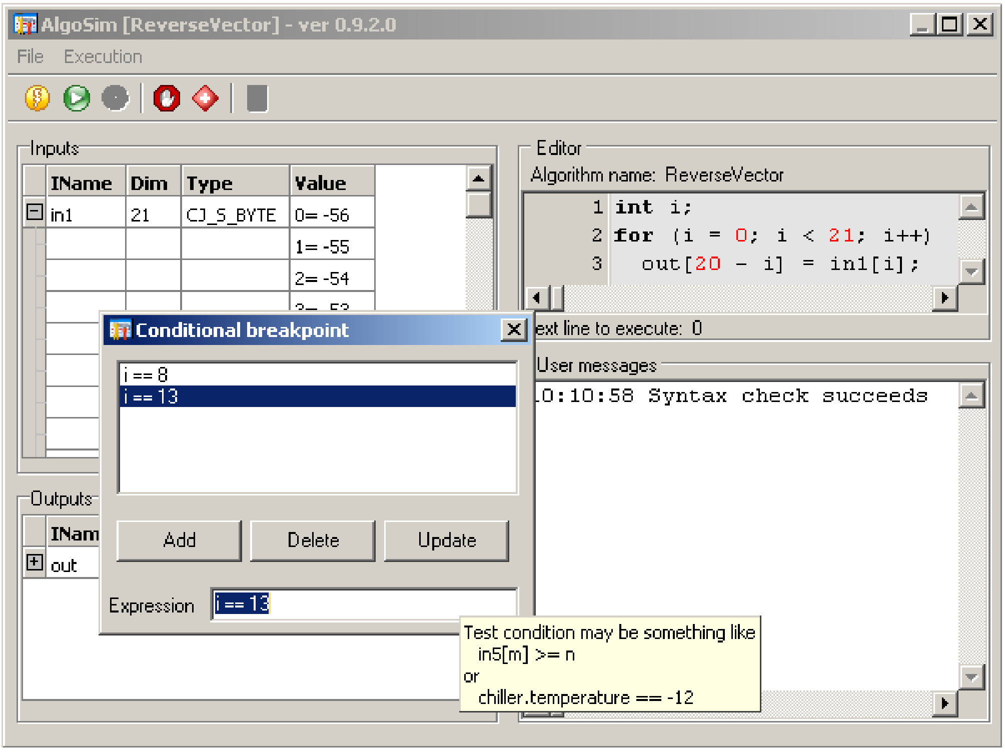
ext line to execute: 0

```
0:10:58 Syntax check succeeds
```

Expression

Test condition may be something like
in5[m] >= n
or
chiller.temperature == -12

As the hint suggests, the expression is a test condition written in C language without outer brackets.



IName	Dim	Type	Value
in1	21	CJ_S_BYTE	0= -56
			1= -55
			2= -54
			3= -53

IName
out

Algorithm name: ReverseVector

```
1 int i;
2 for (i = 0; i < 21; i++)
3   out[20 - i] = in1[i];
```

ext line to execute: 0

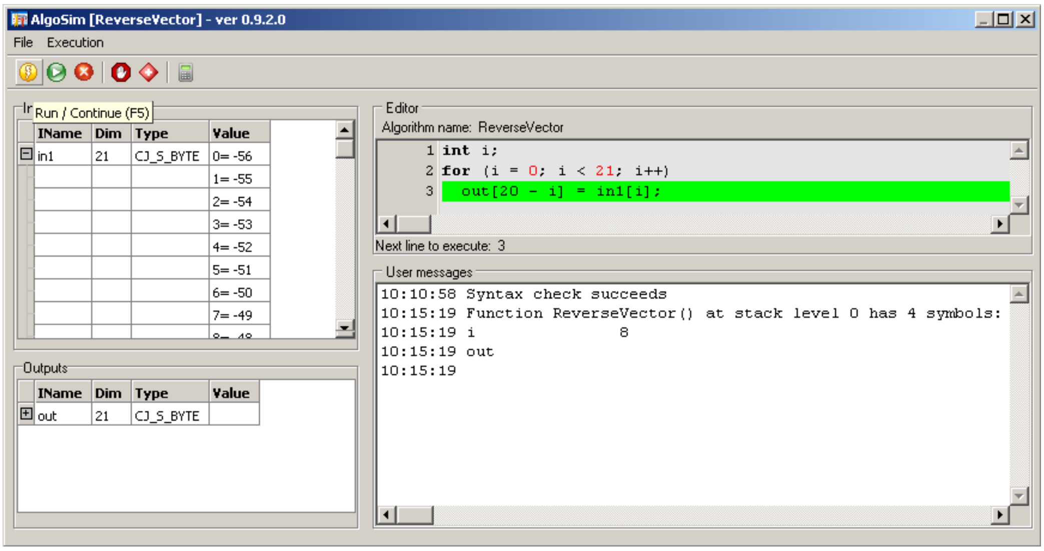
```
.0:10:58 Syntax check succeeds
```

i == 8
i == 13

Expression i == 13

Test condition may be something like
in5[m] >= n
or
chiller.temperature == -12

Click the **Run** icon, the program executes 8 iterations in this sample before prompting the user for an action.



IName	Dim	Type	Value
in1	21	CJ_S_BYTE	0= -56
			1= -55
			2= -54
			3= -53
			4= -52
			5= -51
			6= -50
			7= -49
			8= -48

IName	Dim	Type	Value
out	21	CJ_S_BYTE	

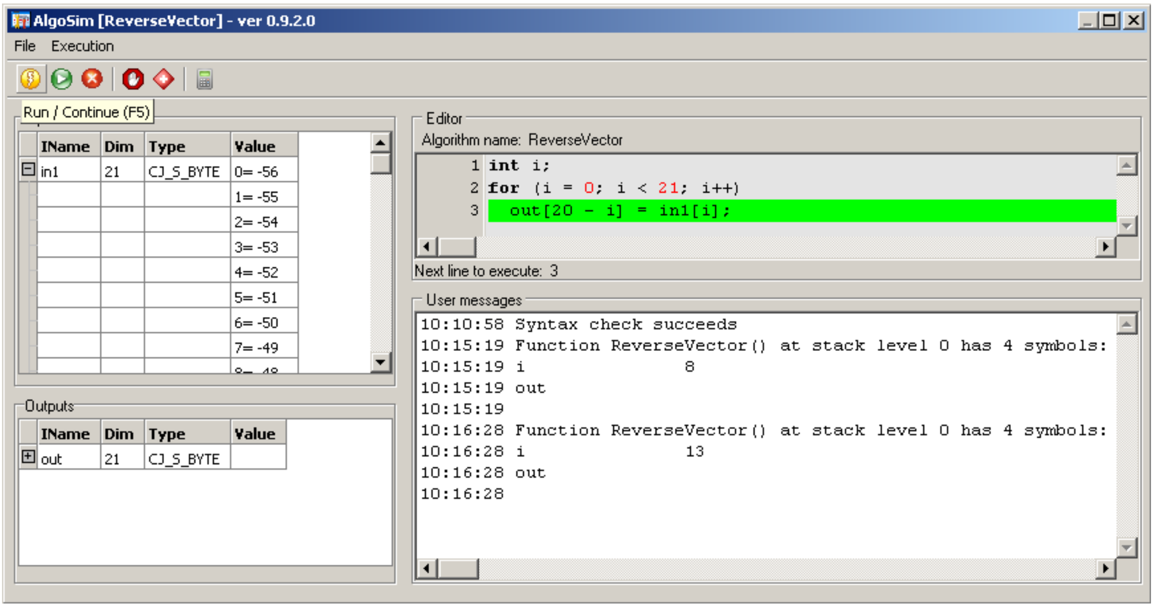
Algorithm name: ReverseVector

```
1 int i;
2 for (i = 0; i < 21; i++)
3   out[20 - i] = in1[i];
```

Next line to execute: 3

```
10:10:58 Syntax check succeeds
10:15:19 Function ReverseVector() at stack level 0 has 4 symbols:
10:15:19 i                    8
10:15:19 out
10:15:19
```

When the **Run** icon is clicked again, the program does further 5 iterations before prompting the user for an action.



	IName	Dim	Type	Value
⊟	in1	21	CJ_S_BYTE	0= -56
				1= -55
				2= -54
				3= -53
				4= -52
				5= -51
				6= -50
				7= -49
				8= -48

	IName	Dim	Type	Value
⊞	out	21	CJ_S_BYTE	

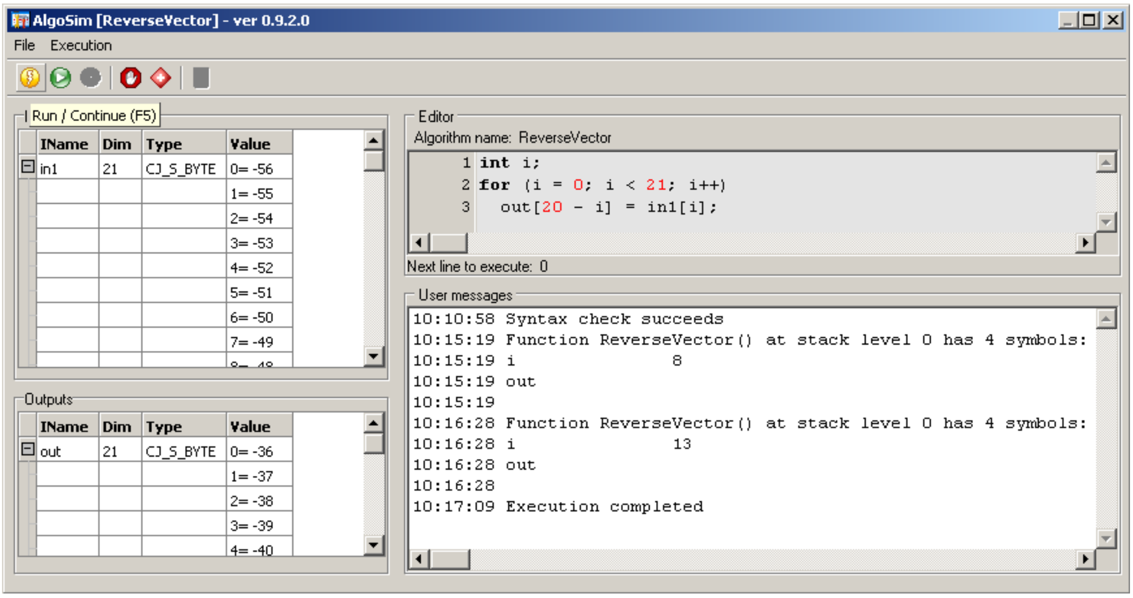
Algorithm name: ReverseVector

```
1 int i;
2 for (i = 0; i < 21; i++)
3   out[20 - i] = in1[i];
```

Next line to execute: 3

```
10:10:58 Syntax check succeeds
10:15:19 Function ReverseVector() at stack level 0 has 4 symbols:
10:15:19 i              8
10:15:19 out
10:15:19
10:16:28 Function ReverseVector() at stack level 0 has 4 symbols:
10:16:28 i              13
10:16:28 out
10:16:28
```

At the third click on the **Run** icon, the program does the remaining iterations and shows the result.



	IName	Dim	Type	Value
⊟	in1	21	CJ_S_BYTE	0= -56
				1= -55
				2= -54
				3= -53
				4= -52
				5= -51
				6= -50
				7= -49
				8= -48

	IName	Dim	Type	Value
⊟	out	21	CJ_S_BYTE	0= -36
				1= -37
				2= -38
				3= -39
				4= -40

Algorithm name: ReverseVector

```
1 int i;
2 for (i = 0; i < 21; i++)
3   out[20 - i] = in1[i];
```

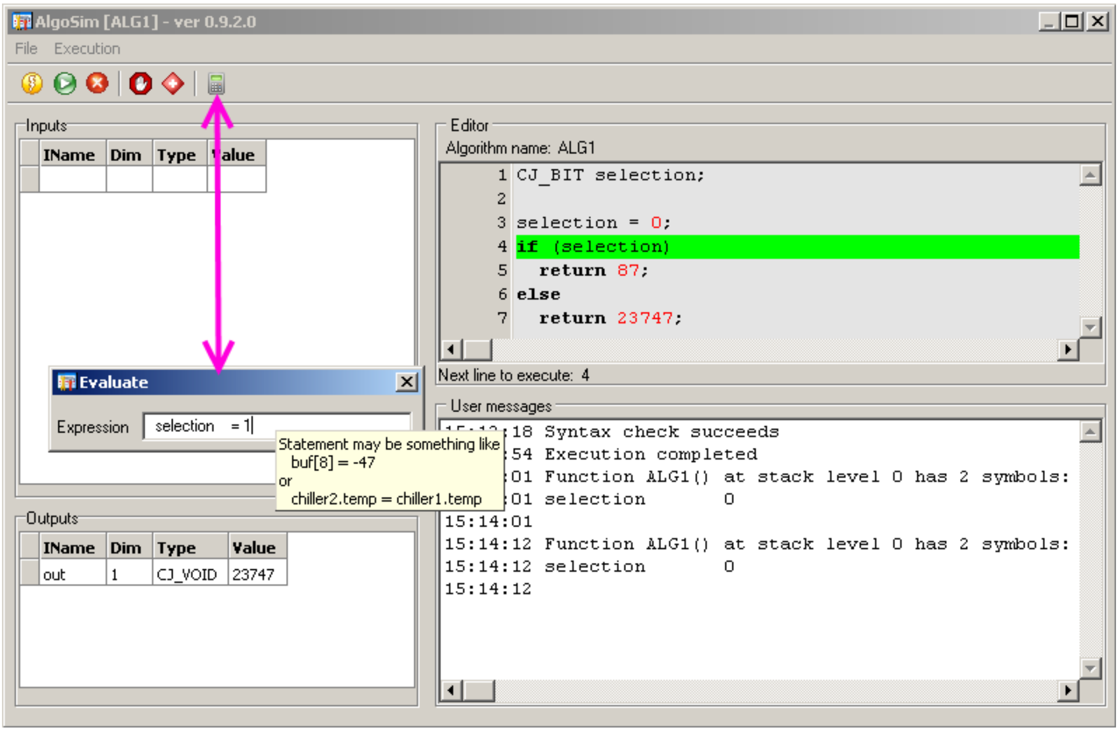
Next line to execute: 0

```
10:10:58 Syntax check succeeds
10:15:19 Function ReverseVector() at stack level 0 has 4 symbols:
10:15:19 i              8
10:15:19 out
10:15:19
10:16:28 Function ReverseVector() at stack level 0 has 4 symbols:
10:16:28 i              13
10:16:28 out
10:16:28
10:17:09 Execution completed
```

3.13.4 Evaluate

The evaluate function allows you to test different execution flows by setting local variables to a specific value. For example, you are debugging an algorithm step-by-step and you need to change the value of a local variable to test different execution flows with the same input.

The example below always returns 23747.



Inputs

IName	Dim	Type	Value

Editor

Algorithm name: ALG1

```
1 CJ_BIT selection;
2
3 selection = 0;
4 if (selection)
5   return 87;
6 else
7   return 23747;
```

Next line to execute: 4

Expression: selection = 1

Statement may be something like
buf[8] = -47
or
chiller2.temp = chiller1.temp

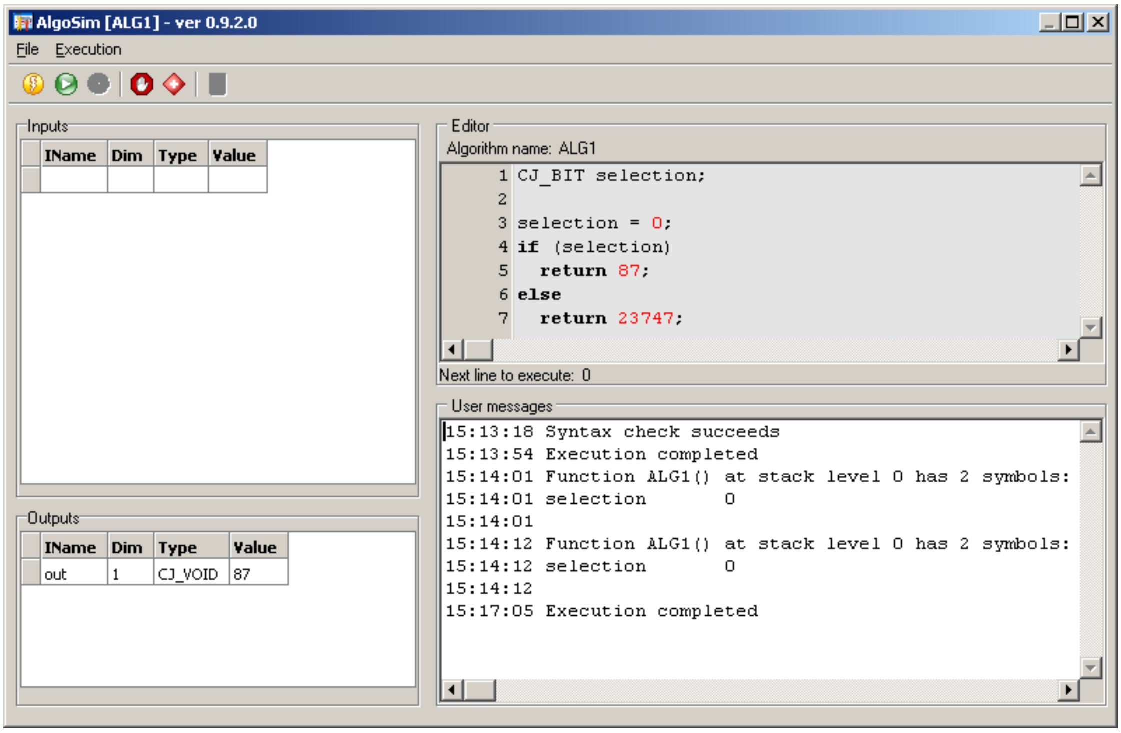
Outputs

IName	Dim	Type	Value
out	1	CJ_VOID	23747

User messages

```
15:1 :18 Syntax check succeeds
    :54 Execution completed
    :01 Function ALG1() at stack level 0 has 2 symbols:
    :01 selection        0
15:14:01
15:14:12 Function ALG1() at stack level 0 has 2 symbols:
15:14:12 selection        0
15:14:12
```

After the program has assigned zero to variable **selection**, the user can change its value through the **Evaluate** pop-up.



Inputs

IName	Dim	Type	Value

Editor

Algorithm name: ALG1

```
1 CJ_BIT selection;
2
3 selection = 0;
4 if (selection)
5   return 87;
6 else
7   return 23747;
```

Next line to execute: 0

Outputs

IName	Dim	Type	Value
out	1	CJ_VOID	87

User messages

```
15:13:18 Syntax check succeeds
15:13:54 Execution completed
15:14:01 Function ALG1() at stack level 0 has 2 symbols:
15:14:01 selection        0
15:14:01
15:14:12 Function ALG1() at stack level 0 has 2 symbols:
15:14:12 selection        0
15:14:12
15:17:05 Execution completed
```

4 Basic Elements Data Types

The data types allowed in the development environment can be divided into 2 categories:

1. simple
2. structured

The simple date types are composed of a single value that can be directly used in the algorithms for processing purposes, while the structured data types are composed of multiple fields since they contain multiple pieces of information.

For example, data type CJ_ANALOG contains 2 pieces of information namely sensor value and sensor error code.

CJ_VOID, introduced by the SoHVAC development system belongs to the simple data category.

4.1 Control Components

In the library manager of the SoHVAC development environment, there are multiple folders grouping the function blocks. There are 2 main folders; Software and Hardware, which are used to address hardware and software resources of the controller.

4.1.1 Software

The software folder contains following entities that are described below:

- Fix
- Par
- Pers
- Var
- Timer
- Command In
- Command Out
- Subsheet
- Algorithm

Fix

An entity of type Fix is a constant variable. These variables types are stored in flash memory (non-volatile memory) and it is not possible to change the value of a Fix during program execution.

Fix objects have the following properties:

Property	**Description**
Top	Sets the position where the element is 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 is 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 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 mouse pointer.
Left	Sets the position where the element is drawn, relative to the left edge of the sheet.
MasterRefresh	Automatic refresh time on networks. HIGH priority is processed in the message queue before the LOW priority. Useful only if the entity is shared through **Network Master** menus.
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.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more information, refer to *Defining Execution Tasks*).
Type	Defines the data type of a constant. The default value of this property is CJ_VOID. It can be changed either automatically by joining an input with a data type different from CJ_VOID to the output of the entity, or manually by setting this property in the properties window. According to the selected data type, the constant contains different limits and takes 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 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 mouse pointer.

NOTE: A maximum number of 1000 Fix objects can be created in a project.

Par

An entity of type Par is a parameter variable. These variables types are stored in non-volatile memory, so that all values are maintained even after the power is removed from the controller.

The parameter variables can be modified during program execution.

Parameter variables can be reset to the factory defaults, that means, to the original value when the machine was delivered.

NOTE: The parameter variables are stored in a specific non-volatile memory space, where the controller performs a CRC check. For program variables which changes frequently, use the persistent variable types instead of parameter variables types.

They have the following properties:

Property	Description
Top	Sets the position where the element is 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 is 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 (refer to *Conditional Visibility*).
Description	Text-type description field where you can type in 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 mouse pointer.
Left	Sets the position where the element is drawn, relative to the left edge of the sheet.
MasterRefresh	Automatic refresh time on networks. HIGH priority is processed in the message queue before the LOW priority. Useful only if the entity is shared through **Network Master** menus.
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.
Order	Execution order applied to calculate this element (refer to *Setting the Execution Order of a Program*).

Property	Description
Parmin	Limits the minimum value of the actual parameters at the value of another entity existent in the application. If set at <NONE> minimum limit is the value set in the Min property.
Precision	Indicates the number of decimal digits used to interpret the internal value of the element and to display the value in **EIML** pages.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more information, refer to *Defining Execution Tasks*).
Type	Defines the data type of the parameter. The default value of this property is CJ_VOID. It can be changed either automatically by joining an element with a data type different from CJ_VOID to the input/output of the entity, or manually by setting this property in the properties window. According to the selected data type, the parameter contains different limits and takes 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 information, refer to *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 mouse pointer.

NOTE: A maximum number of 4000 Par objects can be created in a project.

Pers

An entity of type Pers is a persistent variable. Persistent variables are similar to the parameter variables but with a few differences.

Persistent variables types are stored in non-volatile memory, so that all values are maintained even after the power is removed from the controller.

The persistent variables can be modified during program execution.

Persistent variables cannot be reset to the factory defaults.

NOTE: The persistent variables are ideal to store run time hours or alarm events as these variables may change frequently. There is no CRC check performed over the persistent variable memory space.

They have the following properties:

Property	**Description**
Top	Sets the position where the element is 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 is 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 (refer to *Conditional Visibility*).
Description	Text-type description field where you can type in 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 mouse pointer.
Left	Sets the position where the element is drawn, relative to the left edge of the sheet.
MasterRefresh	Automatic refresh time on networks. HIGH priority is processed in the message queue before the LOW priority. Useful only if the entity is shared through **Network Master** menus.
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 element (refer to *Setting the Execution Order of a Program*).
Precision	Indicates the number of decimal digits used to interpret the internal value of the element and to display the value in EIML pages.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more information, refer to *Defining Execution Tasks*).
Type	Defines the data type of parameter. The default value of this property is **CJ_VOID**. It can be changed either automatically by joining an element having a data type different from **CJ_VOID** to the input/output of the entity, or manually by setting this property in the properties window. According to the selected data type, the state contains different limits and takes 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 information, refer to *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 mouse pointer.

NOTE: A maximum number of 1000 Pers objects can be created in a project.

Var

An entity of type Var is a variable. These variables types are stored in volatile memory, so that all values are lost after the power is removed from the controller.

The variables value can be modified during program execution.

They have the following properties:

Property	**Description**
Top	Sets the position where the element is 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 is 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 (refer to *Conditional Visibility*).
Description	Text-type description field where you can type in 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 mouse pointer.
Left	Sets the position where the element is drawn, relative to the left edge of the sheet.
MasterRefresh	Automatic refresh time on networks. HIGH priority is processed in the message queue before the LOW priority. Useful only if the entity is shared through **Network Master** menus.
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 (refer to *Setting the Execution Order of a Program*).
Precision	Indicates the number of decimal digits used to interpret the internal value of the element and to display the value in EIML pages.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more information, refer to *Defining Execution Tasks*).

Property	Description
Type	Defines the data type of a variable. The default value of this property is **CJ_VOID**. It can be changed either automatically by joining an element having a data type different from **CJ_VOID** to the input/output of the entity, or manually by setting this property in the properties window. According to the selected data type, the variable contains different limits and takes 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 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 mouse pointer.

NOTE: A maximum number of 4000 **Var objects** can be created in a project.

The following table shows the comparison of various variable types:

Type	Description	Memory Type	Runtime Changes	Reset/ Power Cycle	Reset to Factory Default
VAR1	Variables	RAM (volatile)	Yes	Default value SoHVAC reloaded	No
PAR1	Parameter	Flash (non-volatile) (CRC-Check)	Yes	Remain	Yes
PERS1	Persistent	Flash (non-volatile)	Yes	Remain	No
FIX1 0	Constants	Flash (non-volatile)	No	Remain	No

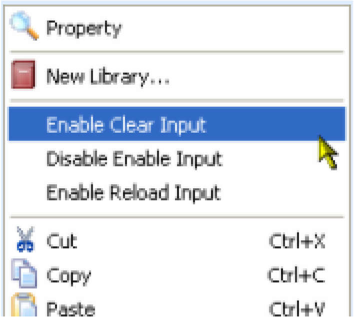
NOTE: Use parameter type for reading (configuration parameters). Use persistent type for writing (operation hours, alarms, and so on).

Timer

Timers are counters that are automatically increased/decreased by the system, under a time base.

They have inputs which you can reset, load the value, enable or cancel the count, and an 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 stops. If it equals 1, timer count is enabled.

Connecting a parameter to the reload input allows loading the timer count with the parameter value instead of a default value (look at **Max** property).

The Reset input senses the transition from 0 to 1 to reload timer count with Max value or reload pin value when connected.

The Clear input realizes the transition from 0 to 1 to delete the current countdown.

The Out Output gives the counter value.

Input	Type	Limits	Description
Enable	CJ_BIT	0-1	Enables or stops the timer during a 0 to 1 transition.
Reset	CJ_BIT	0-1	Enables the timer count-up or count-down during a 0 to 1 transition.
Clear	CJ_BIT	0-1	Clears the timer count during a 0 to 1 transition.
Reload	CJ_WORD	0...65535	Reload (Preset) Time. A 16 bit value which identifies the up/down delay time (hundreds of ms for UoM=100 msec or seconds for UoM=sec).

Output	Type	Limits	Description
Out	CJ_WORD	0...65535	Elapsed Time. Represents the current elapsed time value (hundreds of ms for UoM=100 msec or seconds for UoM=sec)

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

For example, you can provide compressor short-cycle protection or timed lighting using these objects.

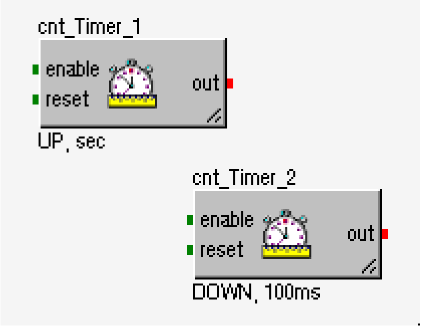
They have the following properties:

Property	**Description**
Top	Sets the position where the element is 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 notes.
Direction	Direction towards which the value is changed when the enable input is activated. If this property is set to UP, the counter value is increased, under the pre-set time base, starting from zero (when timer is reset) until it reaches the value set in Max. If the property is set to DOWN, the value is 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 mouse pointer.
Left	Sets the position where the element is 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 input value is the one loaded as start or end limit, while Max represents only a superior control limit.
Name	Unique name that identifies the element inside a project.
Order	Execution order applied to calculate this element (refer to *Setting the Execution Order of a Program*).
Timed	Allows defining in which task calculate this element and the other elements linked to its inputs. (For more information, refer to *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 *Basic Elements Data Types*).
Udm	Unit of time that causes Timer value to change. This property can take the following values: • 100 ms • sec • min • hour Depending on the value set here, the timer is 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 mouse pointer.

NOTE: A maximum number of 250 Timer objects can be created in a project.

Udm and Direction under Timer Display

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



Command In

The SoHVAC 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 a structure CJ_CMD including the parameter value of the command received and the node that sent it in. In addition, they have the following properties:

Property	Description
Top	Sets the position where the element is 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 *Commands*).
Description	Text-type description field where you can type in 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 mouse pointer.
Left	Sets the position where the element is 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 mouse pointer.

NOTE: A maximum number of 250 **Command In** objects can be created in a project.

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 2 inputs:

1. **param**
2. **trigger**

The param 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.

In addition, it has the following properties:

Property	Description
Top	Sets the position where the element is 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 *Commands*).
Description	Text-type description field where you can type in 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 mouse pointer.
Left	Sets the position where the element is drawn, relative to the left edge of the sheet.

Property	Description
Name	Unique name that identifies the element inside a project.
Order	Execution order applied to calculate this element (refer to *Setting the Execution Order of a Program*).
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more information, refer to *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 mouse pointer.

NOTE: A maximum number of 250 **Command Out** objects can be created in a project.

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 can be applied.

To join internal and external entities, you need to export the relevant inputs/outputs. Exported inputs/outputs are displayed in a different color (blue by default).

You can 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 (refer to *Creating Libraries*).

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 therefore 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.

It is useful to optimize resources, because you need not duplicate the code of an algorithm if you use it multiple times.

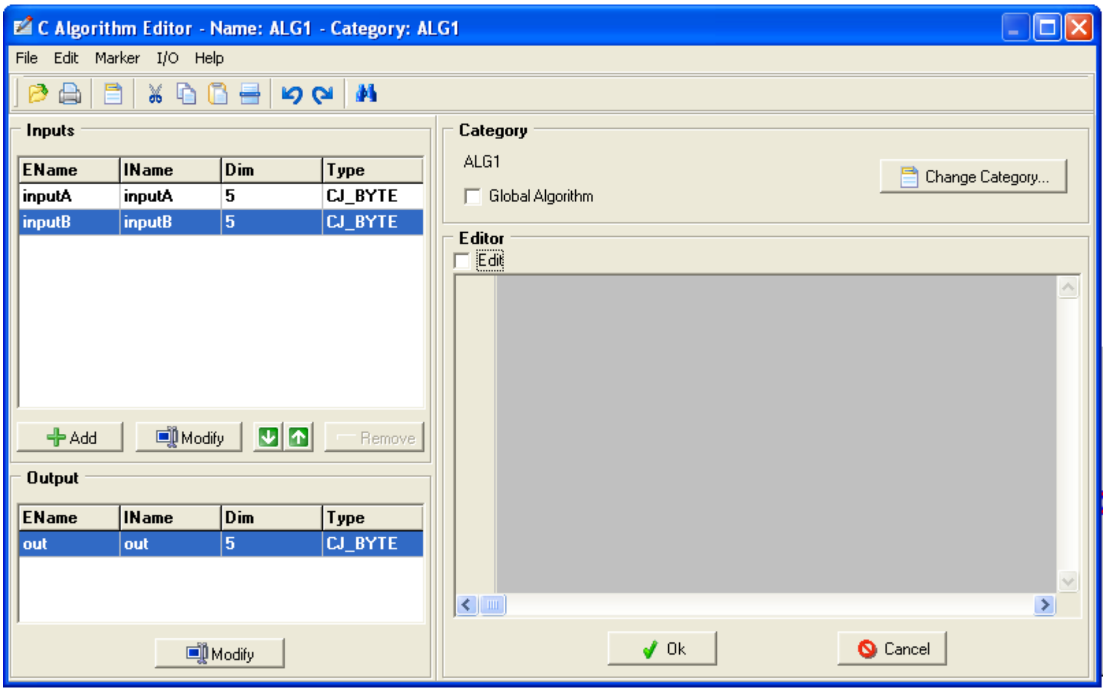
When you add an algorithm to a project, the environment immediately prompts you to define its category which, in turn, automatically set its name.

By default, it has a CJ_VOID output and no inputs. You can add the required number of inputs by selecting **Add Input** from the context menu.

At the beginning, their data type is CJ_VOID. Once they are linked, they can be set to the right data type automatically.

Otherwise, you can use the C Algorithm Editor, 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
inputA	inputA	5	CJ_BYTE
inputB	inputB	5	CJ_BYTE

Output

EName	IName	Dim	Type
out	out	5	CJ_BYTE

Category

ALG1

□ Global Algorithm

Editor

□ Edit

(For more information, refer to *Using C Algorithm Editor*).

4.1.2 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 depends on the type of controller and the number of connected expansions.

Therefore, you can proceed in 2 different ways:

- Add all the required entities to the project and then select the most suitable hardware accordingly, or
- Select 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, and so on.) through the Join Tools.

Digital Input

Digital Input objects represent the digital inputs on the controller or the expansions. They have the following properties:

Property	**Description**
Top	Sets the position where the element is 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 notes.
ExportInBMS	Indicates whether the object is accessible through the network connectivity port or not.
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 mouse pointer.
Left	Sets the position where the element is drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
Type	Indicates data type, for which the Digital Input 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 mouse pointer.

NOTE: A maximum number of 250 **Digital Input** objects can be created in a project.

To add a digital input to a project, select the **DigitalIn** element in the library manager and drag-and-drop it on the sheet where you wish to add the entity.

To display its properties, double-click **DigitalIn** (or else right-click and select **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 join a **Digital In** to the corresponding physical digital input, refer to *Linking Entities*.

Digital Output

Digital Out objects represent a digital output on the controller or the expansions. They have the following properties:

Property	**Description**
Top	Sets the position where the element is 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 notes.
ExportInBMS	Indicates whether the object is accessible through the network connectivity port or not.
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 mouse pointer.
Left	Sets the position where the element is 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 (refer to *Setting the Execution Order of a Program*).
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more information, refer to *Defining Execution Tasks*).
Type	Indicates data type, for which 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 mouse pointer.

NOTE: A maximum number of 250 **Digital Output** objects can be created in a project.

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

Analog Input

Analog Input objects are the abstract representation of any type of analog terminal. They can represent a physical input of the following types: NTC, PTC, PT1000, 0…5 Volt, 0…10 Volt, 0…20 mA, and 4…20 mA.

This means that this type of element can 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:

Property	**Description**
Top	Sets the position where the element is 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 notes.
ExportInBMS	Indicates whether the object is accessible through the network connectivity port or not.
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 mouse pointer.
Left	Sets the position where the element is 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. The number of decimals depends on the selected value in the sensor property. By definition, the temperature inputs have 1 decimal digit. The voltage and current inputs have 2 decimal digits.
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 • NTC 10K-2 • NTC 10K-3 • PTC • PT1000 • Resistance (reserved for future use) • Temperature (reserved for future use) • Humidity (reserved for future use) The selected sensor type has to be supported by the hardware.
Type	Indicates data type, for which 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 mouse pointer.

NOTE: A maximum number of 250 **Analog Input** objects can be created in a project.

To read the state of an analog input added to a project, join its output pin to the input of an Algo or a Var.

CJ_ANALOG is a structured data type that contains multiple pieces of information: an Error field that indicates a possible error, and a Value field that shows its value.

To add an **Analog Input** to a sheet, proceed as you add a **Digital Input**.

Range and precision of the analog inputs according to the sensor type:

Sensor	**Precision**	**Range**	**Unit**
0_10 V	2 digits	0…1000	0.01 V
0_20 mA	2 digits	0…2000	0.01 mA
0_5 V	2 digits	0…500	0.01 V
0_20 mA	2 digits	400…2000	0.01 mA
NTC	1 digit	Refer to the *Modicon M168 Controller Hardware Guide*.	0.1 °C or °F
NTC 10K-2	1 digit	Refer to the *Modicon M168 Controller Hardware Guide*.	0.1 °C or °F
NTC 10K-3	1 digit	Refer to the *Modicon M168 Controller Hardware Guide*.	0.1 °C or °F
Resistance	0 digit	Refer to the *Modicon M168 Controller Hardware Guide*.	1 Ohm
PT1000	1 digit	Refer to the *Modicon M168 Controller Hardware Guide*.	0.1 °C or °F
PTC	1 digit	Refer to the *Modicon M168 Controller Hardware Guide*.	0.1 °C or °F

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:

Property	Description
Top	Sets the position where the element is drawn, relative to the top edge of the sheet.
Actuator	For every analog output you use in a project, you need to specify the type of actuator you want to use. Possible choices are: • 0...10 Volt • 0...20 mA • 4...20 mA PWM: Pulse width modulation 0…100 % with frequency defined by property frequency. FAN: PWM Pulse width modulation 5...95 % synchronous to the AC power supply frequency. The selected actuator type has to be supported by the hardware.
Category	Set to AnalogOut and cannot be changed.
Description	Text-type description field where you can type in notes.
ExportInBMS	Indicates whether the object is accessible through the network connectivity port or not.
Frequency	Sets the frequency for the pulse width modulation output signal.
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 pointer.
Left	Sets the position where the element will be drawn, relative to the left edge of the sheet.
Max	Maximum value that the variable can take.
Min	Minimum value that the variable can take.

Property	**Description**
Name	Unique name that identifies the element inside a project.
Order	Execution order applied to calculate this element under examination (refer to *Setting the Execution Order of a Program*).
Precision	Indicates the number of decimal digits used to interpret the internal value of the element and to display the value in EIML pages. The number of decimals depends on the selected value in the actuator property. By definition, the range is 0…100% with 2 decimal digits.
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more information, refer to *Defining Execution Tasks*).
Type	Indicates data type, for which 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 mouse pointer.

NOTE: A maximum number of 250 **Analog Output** objects can be created in a project.

To add an **Analog Output** to a sheet, proceed as you add a **Digital Input**.

Range and precision of the analog outputs according to the actuator type:

Actuator	**Range**	**Unit**	**Comment**
0_10 V	0…10000	0.01% of full scale	0 is 0 V output 10000 is 10 V output
0_20 mA	0…10000	0.01% of full scale	0 is 0 mA output 10000 is 20 mA output
4_20 mA	0…10000	0.01% of full scale	0 is 4 mA output 10000 is 20 mA output
FAN	0…10000	0.01% of full scale	0…500: output is 0 continuously 9500…10000: output is 1 continuously
PWM	0…10000	0.01% of full scale	0 output is 0 continuously 10000 output is 1 continuously

Button

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

- press
- hold down
- release

Therefore, there can be up to 3 different Button objects referring to the same button, each one of which captures a different event.

To select the event you wish to analyze, set the Cmd property.

Property	**Description**
Top	Sets the position where the element is 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 notes.
ExportInBMS	Indicates whether the object is accessible through the network connectivity port or not.
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 mouse pointer.
Left	Sets the position where the element is drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
Type	Indicates data type, for which Buttons is **CJ_BTN** 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 mouse pointer.

NOTE: A maximum number of 250 **Button** objects can be created in a project.

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 with a value range of 0...3.

By 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 *Basic Elements Data Types*).

NOTE: All products LED are not customer programmable. Some are reserved to indicate product status.

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

These objects have the following properties:

Property	**Description**
Top	Sets the position where the element is 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 notes.
ExportInBMS	Indicates whether the object is accessible through the network connectivity port or not.
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 mouse pointer.
Left	Sets the position where the element is 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 (refer to *Setting the Execution Order of a Program*).
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more information, refer to *Defining Execution Tasks*).
Type	Indicates data type, for which 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 mouse pointer.

NOTE: A maximum number of 250 LED objects can be created in a project.

Buzzer

Buzzer objects are used in situations that require an audible alarm to be sounded in a potentially hazardous condition or in an emergency situation.

To do so, add a Buzzer object to the project and join it to an algorithm with a CJ_BUZZ output.

Depending on the input value of buzzer, its state is set to the following modes:

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

(Refer to *Basic Elements Data Types*).

Property	**Description**
Top	Sets the position where the element is 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 notes.
ExportInBMS	Indicates whether the object is accessible through the network connectivity port or not.
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 mouse pointer.
Left	Sets the position where the element is 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 (refer to *Setting the Execution Order of a Program*).
Timed	Allows you to define in which task this element and the other elements linked to its inputs are calculated. (For more information, refer to *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 mouse pointer.

NOTE: A maximum number of 250 **Buzzer** objects can be created in a project.

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.

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

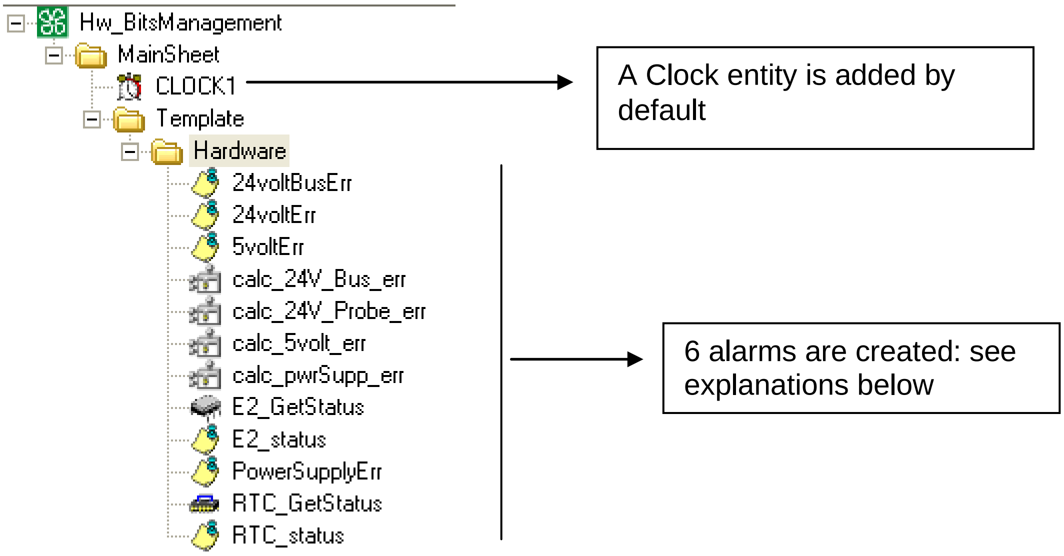
Property	**Description**
Top	Sets the position where the element is drawn, relative to the top edge of the sheet.
Category	It is set to *Clock* and cannot be changed.
Description	Text-type description field where you can type in notes.
ExportInBMS	Indicates whether the object is accessible through the network connectivity port or not.
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 mouse pointer.
Left	Sets the position where the element is drawn, relative to the left edge of the sheet.
Name	Unique name that identifies the element inside a project.
Type	Indicates data type, for which 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 mouse pointer.

NOTE: A maximum number of 250 **Clock** objects can be created in a project.

4.2 Project Template Elements

4.2.1 Hardware Bits

The following elements will be added to the project:

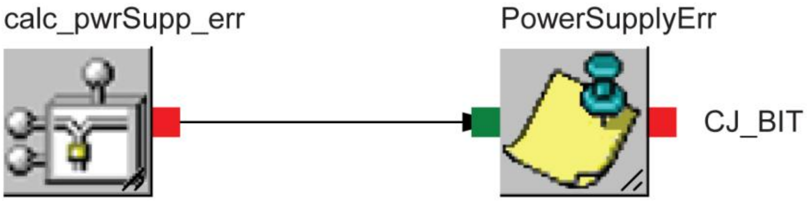


- Hw_BitsManagement
  - MainSheet
    - CLOCK1
  - Template
    - Hardware
      - 24voltBusErr
      - 24voltErr
      - 5voltErr
      - calc_24V_Bus_err
      - calc_24V_Probe_err
      - calc_5volt_err
      - calc_pwrSupp_err
      - E2_GetStatus
      - E2_status
      - PowerSupplyErr
      - RTC_GetStatus
      - RTC_status

A Clock entity is added by default

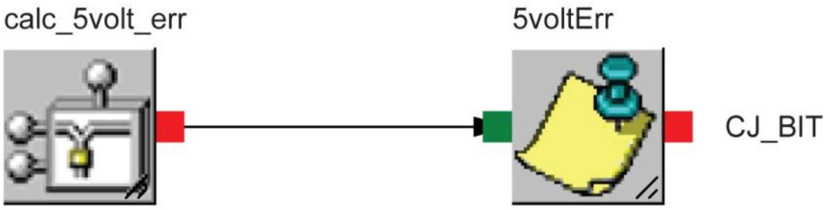
6 alarms are created: see explanations below

PowerSupplyErr



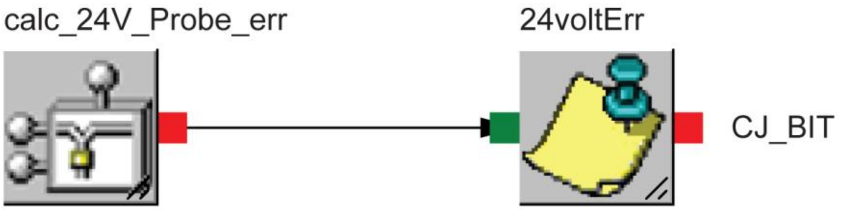
The output goes to 1, if the Input Power supply voltage is detected out of range.

5voltErr



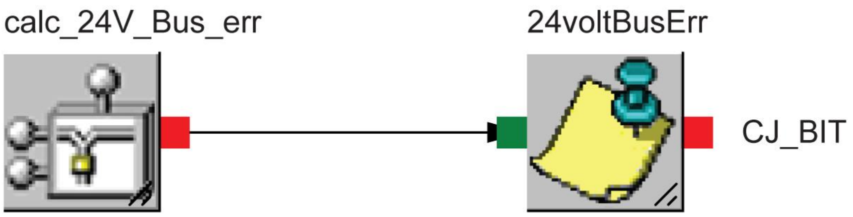
The output goes to 1, if the Analog Input 5 V auxiliary supply (5 V-S TO SENSOR) is detected out of range.

24voltErr



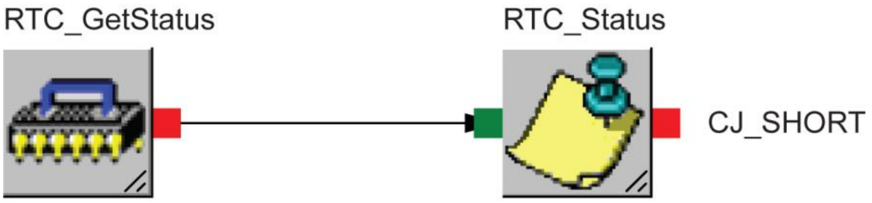
The output goes to 1, if the Analog Input 24 V auxiliary supply (24 V-S TO SENSOR) is detected out of range.

24voltBusErr



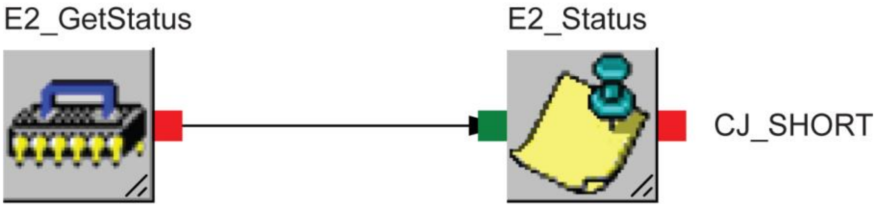
The output goes to 1, if the Display 24 V auxiliary supply (24 V-S TO DISPLAY) is detected out of range.

RTC_Status



Refer to the *RTC_GetStatus* function block description as described in the Standard Library User Guide.

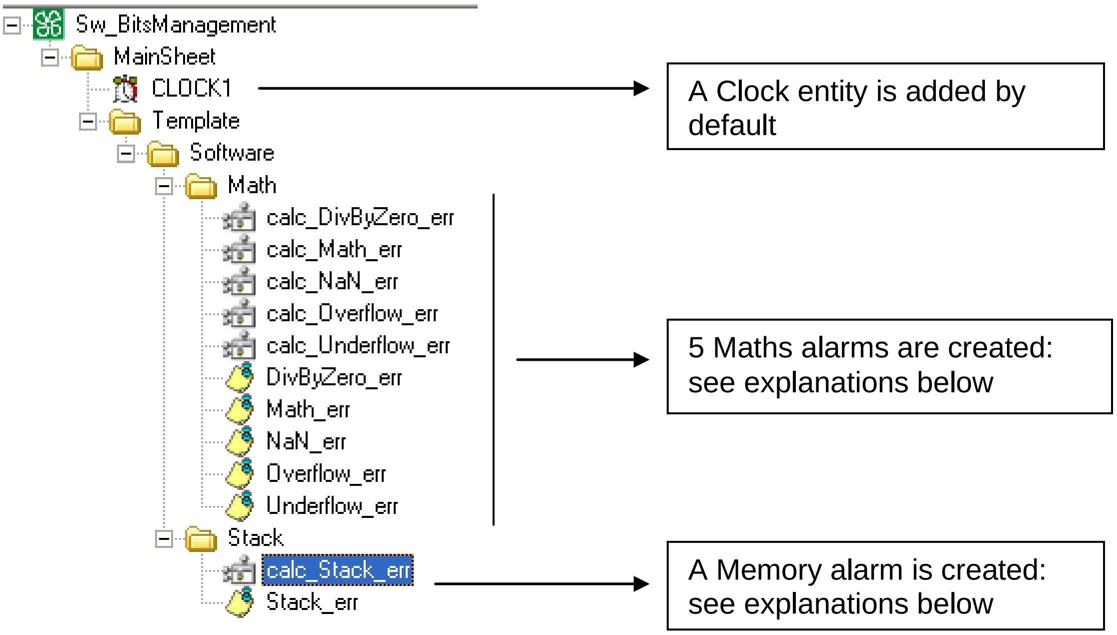
E2_GetStatus



Refer to the *E2_GetStatus* function block description as described in the Standard Library User Guide.

4.2.2 Software Bits

The following elements will be added to the project:



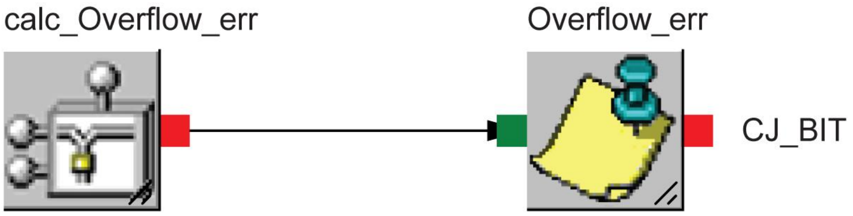
- Sw_BitsManagement
  - MainSheet
    - CLOCK1
    - Template
      - Software
        - Math
          - calc_DivByZero_err
          - calc_Math_err
          - calc_NaN_err
          - calc_Overflow_err
          - calc_Underflow_err
          - DivByZero_err
          - Math_err
          - NaN_err
          - Overflow_err
          - Underflow_err
        - Stack
          - calc_Stack_err
          - Stack_err

A Clock entity is added by default

5 Maths alarms are created: see explanations below

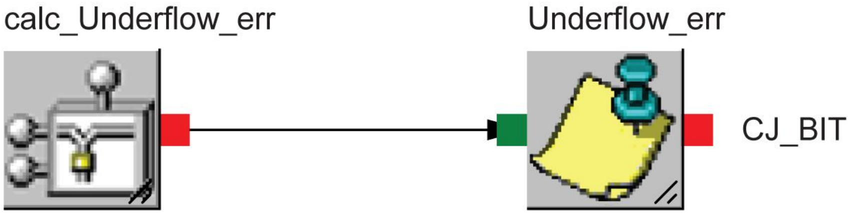
A Memory alarm is created: see explanations below

Overflow_err



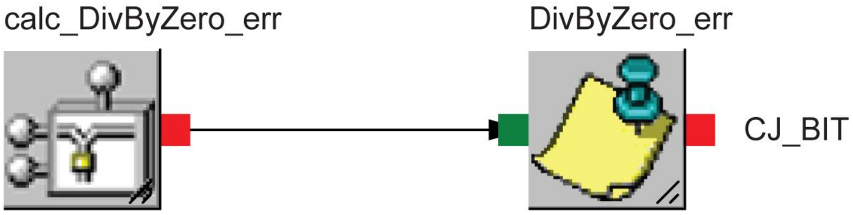
The output goes to 1 when a mathematical overflow is detected.

Underflow_err



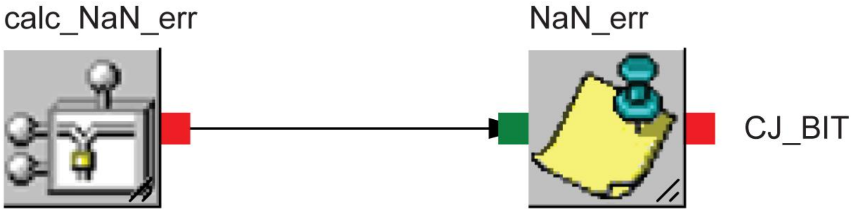
The output goes to 1 when a mathematical underflow is detected.

DivByZero_err



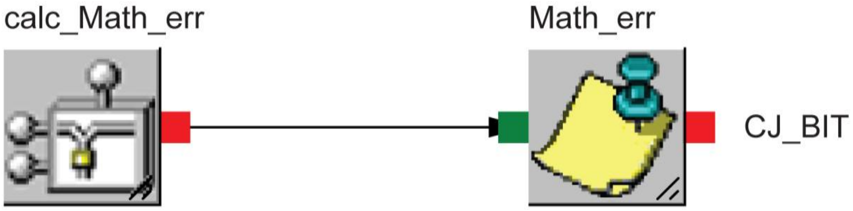
The output goes to 1 when a mathematical division by zero is detected.

NaN_err



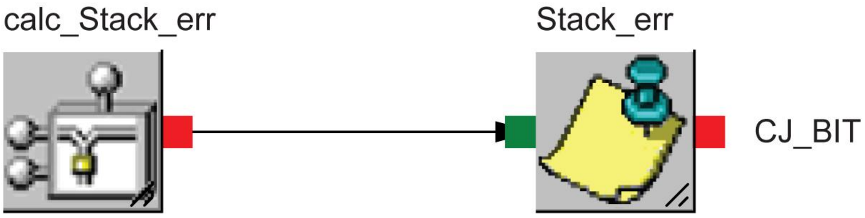
The output goes to 1 when a Not a Number mathematical error is detected (that is, square root of a negative number).

Math_err



The output goes to 1 when a mathematical error is detected (overflow, underflow, NaN, division by zero).

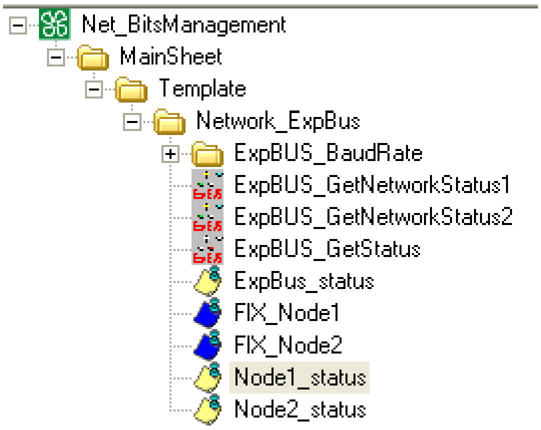
Stack_err



The output goes to 1 when a memory stack error is detected.

4.2.3 ExpBUS Network

The following template will be added to the project:



```
Net_BitsManagement
  MainSheet
    Template
      Network_ExpBus
        ExpBUS_BaudRate
        ExpBUS_GetNetworkStatus1
        ExpBUS_GetNetworkStatus2
        ExpBUS_GetStatus
        ExpBus_status
        FIX_Node1
        FIX_Node2
        Node1_status
        Node2_status
```

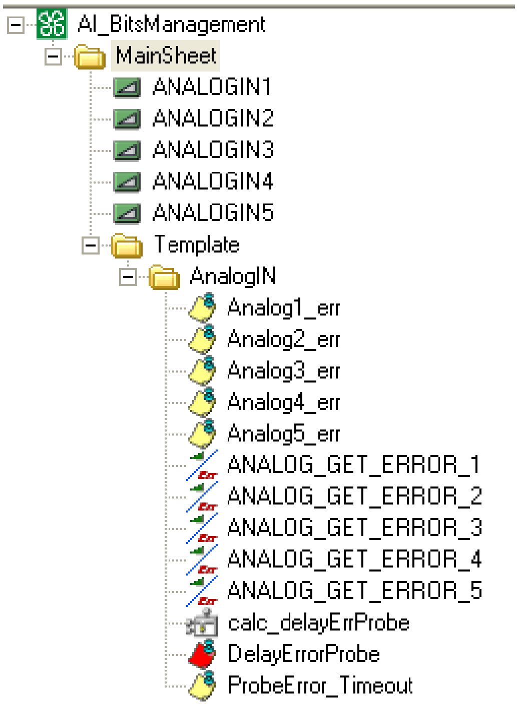
The template defines the minimum functions required to manage the system with 2 other devices connected.

- **ExpBUS_BaudRate**: minimum setting of the bus
- **ExpBUS_GetStatus**: monitoring of the equipment connection to the **ExpBUS**
- **ExpBUS_GetNetworkStatusx**: monitoring of the remote equipment status

The definition of the function blocks ExpBUS_BaudRate, ExpBUS_GetStatus, and ExpBUS_GetNetworkStatusx are available in the section *Expansion Bus Libraries in the Standard Function Blocks User Guide*.

4.2.4 Analog Inputs

The following template will be added to the project:



```
AI_BitsManagement
    MainSheet
        ANALOGIN1
        ANALOGIN2
        ANALOGIN3
        ANALOGIN4
        ANALOGIN5
        Template
            AnalogIN
                Analog1_err
                Analog2_err
                Analog3_err
                Analog4_err
                Analog5_err
                ANALOG_GET_ERROR_1
                ANALOG_GET_ERROR_2
                ANALOG_GET_ERROR_3
                ANALOG_GET_ERROR_4
                ANALOG_GET_ERROR_5
                calc_delayErrProbe
                DelayErrorProbe
                ProbeError_Timeout
```

The template defines the minimum functions required to manage the analog errors for 5 analog channels.

Through DelayErrorProbe parameter, it is possible to filter the errors and signal an error only when it is active for more than that specified by the Delay parameter. The ‘delay’ is in seconds.

4.3 Simple Data Types

Simple data may be divided into 2 logical categories:

- The first category is composed of all C-language data types.
- The second category includes new unstructured data types created by the SoHVAC development environment.

The summarizing table 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
- **Minimum**: minimum value it can take
- **Maximum**: maximum value it can take
- **Corr. ANSI C**: corresponding ANSI C data type

Data Type	**Sign**	**Repr.**	**Minimum**	**Maximum**	**Corr. ANSI C**
CJ_BIT	No	1 bit	0 (FALSE)	1 (TRUE)	(*)
CJ_S_BYTE	Yes	8 bit	-128	127	Signed char
CJ_BYTE	No	8 bit	0	255	Unsigned char
CJ_SHORT	Yes	16 bit	–32768	32767	Signed short
CJ_WORD	No	16 bit	0	65535	Unsigned short
CJ_LONG	Yes	32 bit	-2147483648	2147483647	Signed long
CJ_DWORD	No	32 bit	0	4294967295	Unsigned long

(*) ANSI-C does not declare a BOOL data type, as is the case with other program language types (for example, C++), but rather uses CHAR with the indications *different from 0* for **TRUE** and *equal to 0* for **FALSE**.

All operations allowed by the ANSI C language can 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 SoHVAC environment need to be discussed individually.

4.3.1 CJ_VOID

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

You can define the data type of a generic object (such as a Var or the inputs of an algorithm) by joining it to an object whose data type has already been set.

For example, if you add a variable to a project, it is set to CJ_VOID by default.

By joining this variable to a digital input (defined as CJ_BIT), the data type of the variable is set to CJ_BIT automatically.

4.3.2 CJ_LED and CJ_BUZZ

Data types CJ_LED and CJ_BUZZ are similar and represent the possible values that a LED and a Buzzer object may take respectively.

These data types may take values ranging from 0…3, which correspond to the following states:

Value	State
0	OFF
1	Continuously ON
2	Slow flash/beeps
3	Fast flash/beeps

4.3.3 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 is used 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.

4.3.4 CJ_TIME

The CJ_TIME data type has been implemented to perform operations on time; for instance, to manage different controller time ranges and in multiple other cases.

It represents the number of seconds elapsed since the beginning of the day (00:00:00) and it can be easily converted into the structure CJ_TIME_STRUCT through the conversion function especially provided.

4.3.5 CJ_DATETIME

The CJ_DATETIME data type has been implemented for all those cases where you 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 can be used directly in the algorithms. Otherwise, you can convert it into the structure CJ_DATETIME_STRUCT through the library functions (refer to *CJ_DATETIME_STRUCT*), which makes your work easier.

4.4 Structured Data Types

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

```
structure.fieldname
```

The structured data types and their meaning are described below.

4.4.1 CJ_ANALOG

The CJ_ANALOG data type represents an analog input. The structure is composed of 2 fields:

- Error, a byte type representing an error code. If this field equals zero, there are no sensor errors. Otherwise, it takes 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 data type CJ_ANALOG.

CJ_AI_MIN_RESERVED and CJ_AI_MAX_RESERVED are the minimum and maximum values above which the field value of the sensor is in error status.

If the field value of the sensor is not included inside these limits, the sensor is considered to be in error status (field Error is positive).

This can be used to exchange the sensors status in an ExpBUS communication without using the CJ_ANALOG complete structure.

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

4.4.2 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**, the command has been received and the intended action can 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.

4.4.3 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 (that is, it has been pressed, released, or held down).
- If it is **TRUE**, 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.

4.4.4 CJ_DATE_STRUCT

The CJ_DATE_STRUCT data type is used to perform operations on dates.

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 2 digits of the year, starting from the year 2000. For example, if this field takes the value 12, it represents the year 2012.

Starting from the unstructured data type CJ_DATE, you can fill in the CJ_DATE_STRUCT structure with the help of 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);
```

4.4.5 CJ_TIME_STRUCT

The CJ_TIME_STRUCT data type is used to perform operations on times, for example, to manage different time ranges.

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].

Starting from the unstructured data type CJ_TIME, you can fill in the CJ_TIME_STRUCT structure with the help of 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);
```

4.4.6 CJ_DATE_TIME_STRUCT

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 2 digits of the year, starting from the year 2000. For example, if this field takes the value 12, it represents the year 2012.

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.

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);
```

4.5 Type Compatibility Table

When you attempt to join 2 entities, the system verifies whether the data types of the entities are compatible or not.

The compatibility table is used to ensure that the output data type is compatible with the input 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 in the first column), and the data type of the input where you want to end the link in the upper part (text in the top row).

If there is an X where both row and column meet, the 2 data types are compatible and can be successfully linked. Otherwise, an error message is displayed and cannot be linked.

FROM (OUTPUT) TO INPUT	CJ_VOID	CJ_BIT	CJ_BYTE	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	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

The data type CJ_VOID 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 automatically converts it to match the data type to which it is linked.

NOTE: This data type compatibility verification is carried out when the link is made. Therefore, if you make any changes to the type of inputs or outputs after that, it could lead to inconsistencies in the operation of the program.

Two entities with an array property greater than one are compatible with one another only if the two following conditions are true:

1. If both the entities have the same array size (same array property value)
2. If the entities are of the same data type, with the exception:

- when start entity data type is **CJ_BIT** and linked entity data type is **CJ_S_BYTE**
- when start entity data type is **CJ_BIT** and linked entity data type is **CJ_BYTE**

In all other cases, entities are incompatible.

4.6 Elements in EIML Pages

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

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

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

There are different types of pages available, according to the display for which they are designed:

- graphic (120 x 32 pixels for example)
- alphanumeric (20 x 4 characters)

Selecting **File→New→New EIML Page** from the menu or clicking the ▦ icon in the toolbar displays a list of pages that can be created.

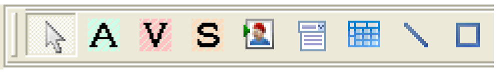
Once you have selected, a window appears in the work area, where you can compose the page and set the desired display features.

The size of each page in bytes is shown in the lower right corner. This information is used to determine whether or not a terminal is able to display that page.

4.6.1 Toolbar

The EIML toolbar (displayed on every EIML page) lists the elements that you can add to the page:

- Texts
- Variables
- Strings
- 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, and so on.), select the desired element in the toolbar and draw the area where you wish to place it.

4.6.2 Embedding an Element in an EIML Page

To add an element to a page, select it from the toolbar and move the mouse while holding down the left button to create a rectangle of the desired size.

Once you have created it, you can move it to the desired position.

4.6.3 Page Properties

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

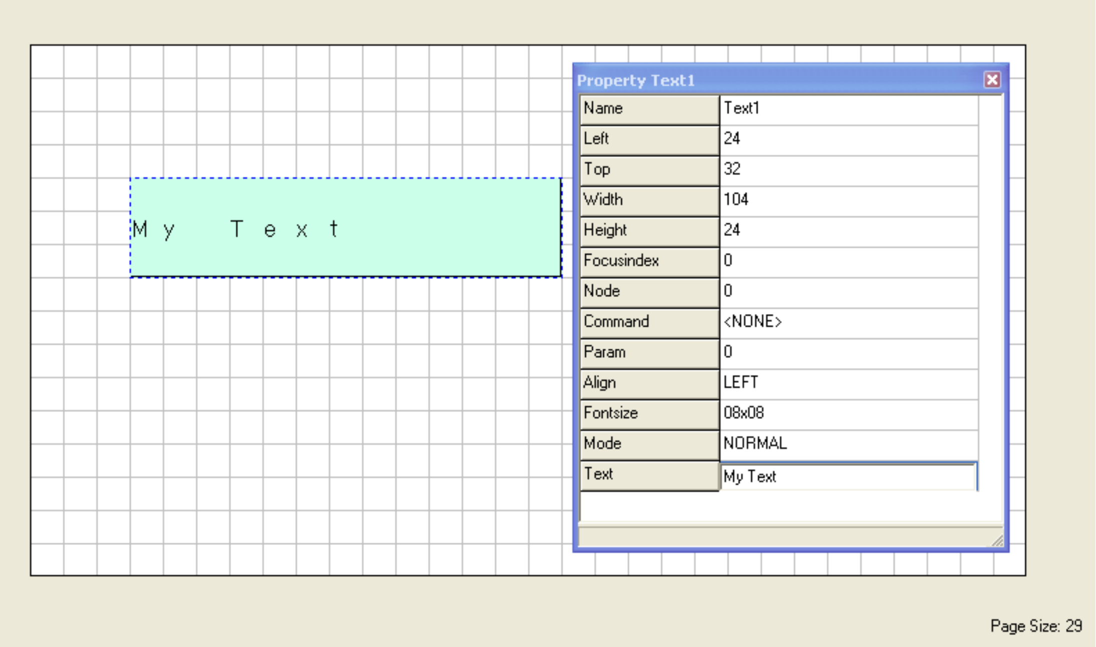
Property	**Description**
Name	Unique name that identifies the page inside a project.
Id	The Id property is a numeric index ranging from 1 to 240 that allows you to identify the page inside a project.
PreviousPage	If different from <NONE>, indicates the previous page that is loaded when the LEFT key is pressed.
NextPage	If different from <NONE>, indicates the next page that is loaded when the RIGHT key is pressed.
TimeoutPage	If different from <NONE>, indicates the page that is loaded when timeout, set in Timeout property, expires. This page is also loaded when ESC key is pressed. If its value is <NONE>, the default page of the user interface is 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. Each time a page with a specific Id is requested, a page in the current language is 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 does not, there is a password prompt requesting to enter a password with a higher authorization level.
Encoding	Indicates if you want to encode the page. When a variable is used in the page, selecting this property reduces the flash size of the page, 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* property selected. When the cursor is moved on the last element of a page, the next element active by the cursor is the first of the same page.
Memo Focus Index	If selected, allows you to store the *Focusindex* property during page navigation.
CondVis	The EIML page contains objects which are conditionally visible. This property must be selected to hide the entity values.

4.6.4 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, select it, right-click, and select **Properties** from the context menu, or double-click the element itself.

A window listing all the properties of the selected element is displayed.



Name	Text1
Left	24
Top	32
Width	104
Height	24
Focusindex	0
Node	0
Command	<NONE>
Param	0
Align	LEFT
Fontsize	08x08
Mode	NORMAL
Text	My Text

4.6.5 Text

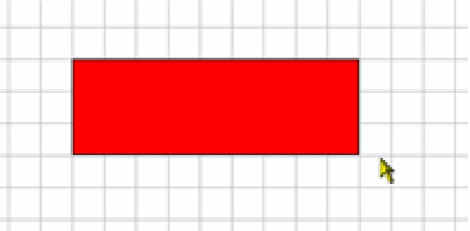
The text element allows you to add descriptions, activate commands, and enable page browsing.

As described under section *Basic Operations,* page *22*, 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.

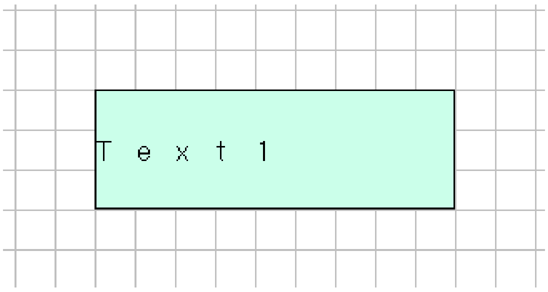
Property	Description
Name	Unique name that identifies the element inside a project.
Left	Offset from the left edge of the page, in pixels.
Top	Offset from the top edge of the page, in pixels.
Width	Element width in pixels.
Height	Element height in pixels.
Focusindex	The *Focusindex* property defines the sequence in which the element receives focus when moving the cursor. The permissible range is 0…255. If set to 0, the element is never selected by the cursor. If set to a different value, defines after how many cursor movements it is selected (for those who are familiar with Windows programming, it is the same concept as the TabIndex of controls).

Property	Description
Node	Node indicates the recipient node number. The properties Command and Param allow a text to send a command to the controller. An action can 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 requests the controller to execute such command with the parameter set in the *param* property.
Command	
Param	
Align	Permissible values are LEFT, CENTER, and RIGHT that indicate left, center and right alignment respectively.
Fontsize	Describes the font size used to write text. For example, if you use an 8x8 font size, a character is 8 pixels high and 8 pixels wide.
Mode	A text can be displayed in 4 different modes: • NORMAL • NEGATIVE • NORMAL-BLINK • NEGATIVE- BLINK If the state is NORMAL, the text is displayed in black on a white background. Conversely, if the state is NEGATIVE, the text is displayed in white on a black background. If you select the flashing modes, the text also flashes 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.

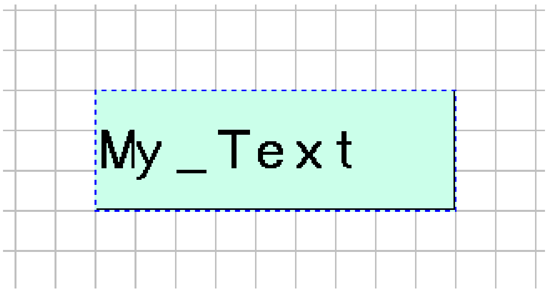


On releasing the key, a text element set to the default values are drawn.



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.

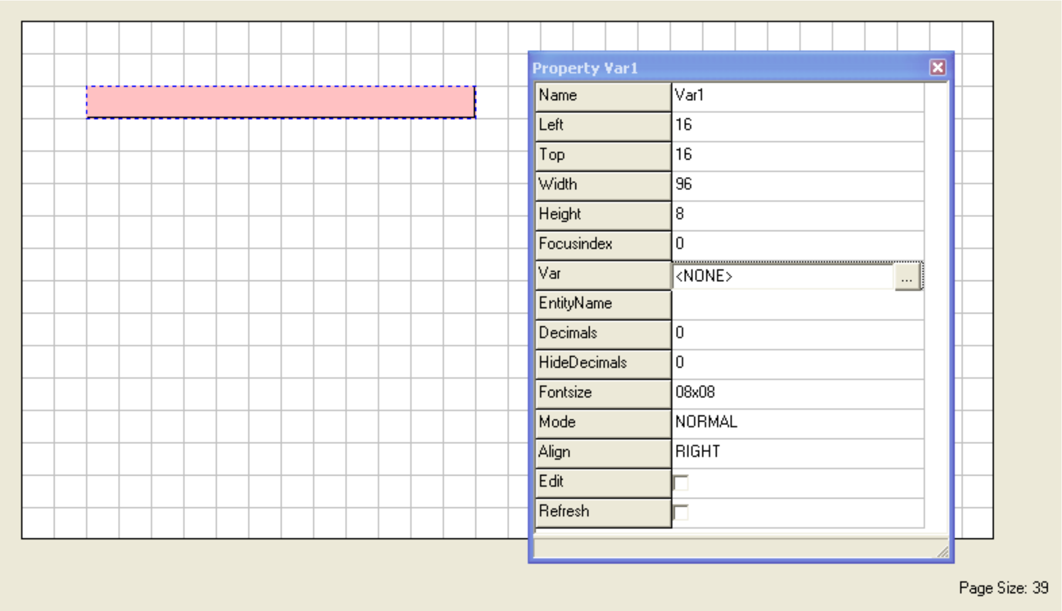
After that, you can go on setting the remaining properties (Font size, Align, and so on.) and the element is displayed with a new look:



4.6.6 Variable V

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

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



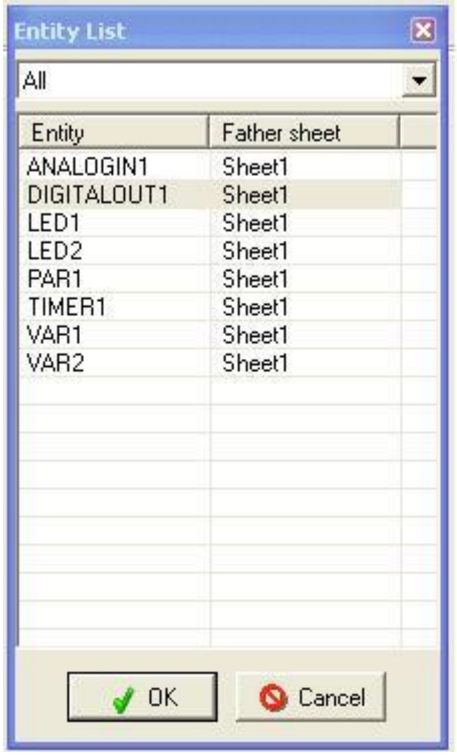
Property Var1

Name	Var1
Left	16
Top	16
Width	96
Height	8
Focusindex	0
Var	<NONE> ...
EntityName	
Decimals	0
HideDecimals	0
Fontsize	08x08
Mode	NORMAL
Align	RIGHT
Edit	□
Refresh	□

Page Size: 39

The variable element can be linked to a project variable. To link, select the **Var** property (set to <NONE> by default) and click the button that appears.

A window opens and allows you to select one of the project entities whose value you wish to display.



Entity	Father sheet
ANALOGIN1	Sheet1
DIGITALOUT1	Sheet1
LED1	Sheet1
LED2	Sheet1
PAR1	Sheet1
TIMER1	Sheet1
VAR1	Sheet1
VAR2	Sheet1

The table describes the properties of variables.

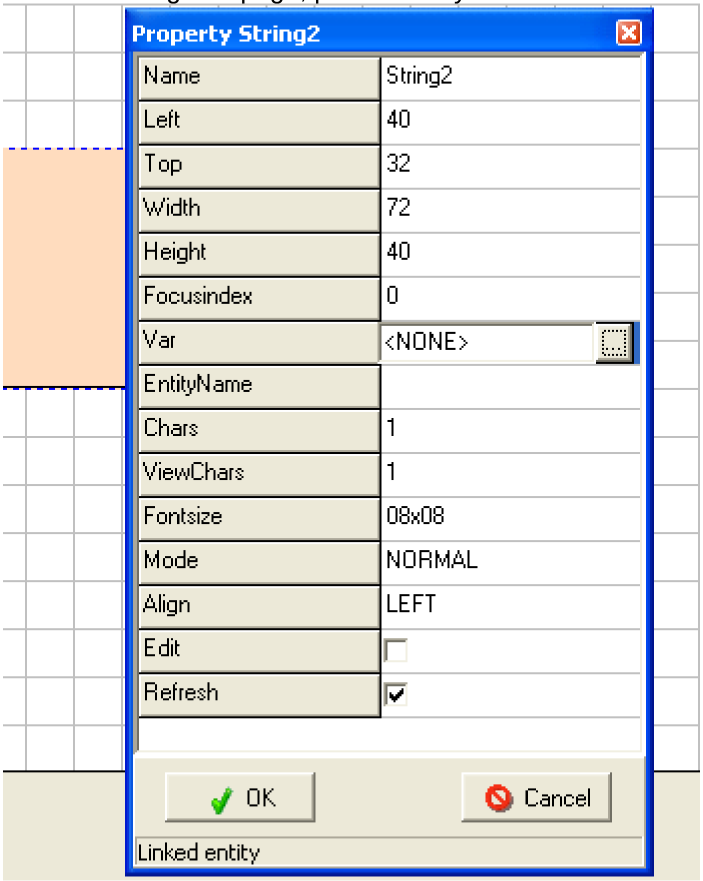
Property	Description
Name	Unique name that identifies the element inside a project.
Left	Offset from the left edge of the page, in pixels.
Top	Offset from the top edge of the page, in pixels.
Width	Element width in pixels.
Height	Element height in pixels.
Focusindex	The *Focusindex* property defines the sequence in which the element receives focus when moving the cursor. The permissible range is 0…255. If set to 0, the element is never selected by the cursor. If set to a different value, defines after how many cursor movements it is 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, and so on.). Select this property and click the button that appears. A window opens and allows you to select 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 (refer to *Precision property*).

Property	**Description**
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 8 x 8 font size, a character is 8 pixels high and 8 pixels wide.
Mode	A variable can be displayed in 4 different modes. • NORMAL • NEGATIVE • NORMAL-BLINK • NEGATIVE-BLINK If the state is NORMAL, the variable is displayed in black on a white background. Conversely, If the state is NEGATIVE, the variable is displayed in white on a black background. If you select the flashing modes, the variable also flashes 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 other than zero).
Refresh	If the *Refresh* property is enabled, the variable is continuously requested. Otherwise, it is refreshed slowly so as to avoid overloading communication between controllers.

4.6.7 Strings

String objects allow you to display and set the string values of the CJ_CHAR entities.

To add a string to a page, proceed as you add a text.

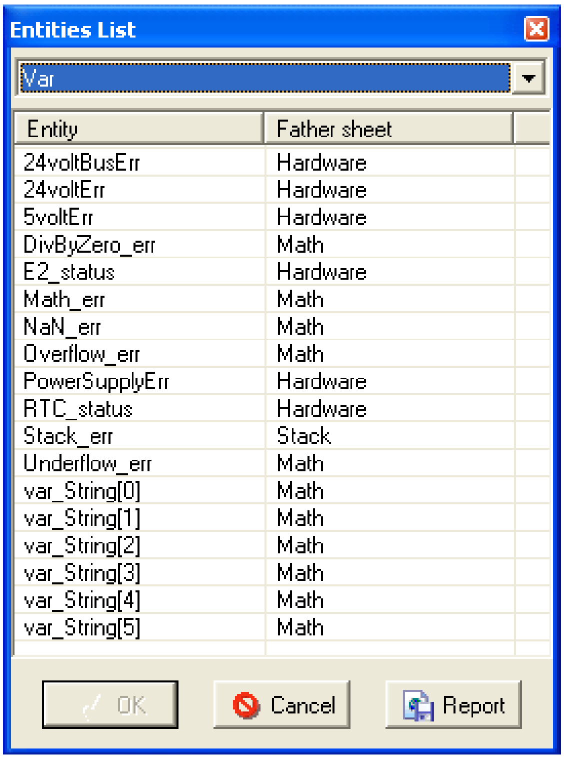


Property String2

Name	String2
Left	40
Top	32
Width	72
Height	40
Focusindex	0
Var	<NONE>
EntityName	
Chars	1
ViewChars	1
Fontsize	08x08
Mode	NORMAL
Align	LEFT
Edit	□
Refresh	☑

Linked entity

The string element can be linked to a project variable. To link, select the **Var** property (set to <NONE> by default) and click the button that appears. A window opens and allows you to select one of the project entities whose value you wish to display.



Entities List

Var

Entity	Father sheet
24voltBusErr	Hardware
24voltErr	Hardware
5voltErr	Hardware
DivByZero_err	Math
E2_status	Hardware
Math_err	Math
NaN_err	Math
Overflow_err	Math
PowerSupplyErr	Hardware
RTC_status	Hardware
Stack_err	Stack
Underflow_err	Math
var_String[0]	Math
var_String[1]	Math
var_String[2]	Math
var_String[3]	Math
var_String[4]	Math
var_String[5]	Math

Through the tool **Entity List** it is possible to set the first characters of the string from which the showing of the same will start; to do that it is enough to select from the list the desired position of the element array CJ_CHAR.

The **focusIndex** of the element String always makes reference to the position of the first editable characters; the other ones can be selected using the keys of the user interface.

The table describes the properties of variables.

Property	**Description**
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 receives focus when moving the cursor. The permissible range is 0…255. If set to 0, the element is never selected by the cursor. If set to a different value, defines after how many cursor movements it is 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, Fix.) on condition it is a **CJ_CHAR one**. Select this property and click the button that appears. A window opens and allows you to select one of the project entities whose value you wish to display.
Chars	Number of characters the string is made of. This is a read-only property and is automatically set when the entity you wish to display is joined (refer to *Precision property*).
ViewChars	Allows you to set the number of characters that will be displayed from the character at the beginning of the string.
Fontsize	Describes the font size used to write text. For example, if you use an 8 x 8 font size, a character is 8 pixels high and 8 pixels wide.
Mode	A variable can be displayed in 4 different modes. • NORMAL • NEGATIVE • NORMAL-BLINK • NEGATIVE-BLINK If the state is NORMAL, the variable is displayed in black on a white background. Conversely, If the state is NEGATIVE, the variable is displayed in white on a black background. If you select the flashing modes, the variable also flashes 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 other than zero).
Refresh	If the *Refresh* property is enabled, the variable is continuously requested. Otherwise, it is refreshed slowly so as to avoid overloading communication between controllers.

4.6.8 Icons

This function allows you to add icons to EIML pages, and to display a state or activate a command.

Icons have 4 different modes:

1. Normal
2. Negative
3. Flashing normal
4. Flashing negative

The mode can be changed by the program, to show an alarm for example.

After adding an Icon element to the EIML page, an open window opens automatically. Browse to the picture to add and select it.

Picture can have following format: *.bmp*.

The table describes of all icon properties:

Property	**Description**
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…255. If set to 0, the element is never selected by the cursor. If set to a different value, defines after how many cursor movements it will be selected.
Node	Node indicates the recipient node number.
Command	The properties Command and Param allow an icon to send a command to the controller. An action may be attributed to the icon by setting a command different from <NONE>. When the cursor is over this element and the ENTER key is pressed, it requests the controller to execute such command with the parameter set in the *param* property.
Param	
Var	The *Var* property allows you to join an entity to the icon as you 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 is 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 is loaded and converted to black/white.
Mode	An icon can be displayed in 4 different modes: 1 NORMAL 2 NEGATIVE 3 NORMAL-BLINK 4 NEGATIVE-BLINK If the state is NORMAL, the icon is displayed in black on a white background. Conversely, If the state is NEGATIVE, the icon is displayed in white on a black background. If you select the flashing modes, the icon also flashes in the respective modes. When the *Var* property is set, the Mode is aligned with that property, whatever is the mode property
Refresh	The *Refresh* property allows you to specify how often the value of the entity linked to the icon is requested. If the icon is not linked to any entity, this property is ignored. If the *Refresh* property is enabled, the variable is continuously requested. Otherwise, it is refreshed slowly so as to avoid overloading communication between controllers.

4.6.9 Combo

Combo objects are an innovative way to represent the information contents of an entity.

Through these objects, you can link a text or an icon to each value taken by an entity, thus providing a flexible and user-friendly way of displaying the content of entity.

Unlike variables, combo objects lack the following properties:

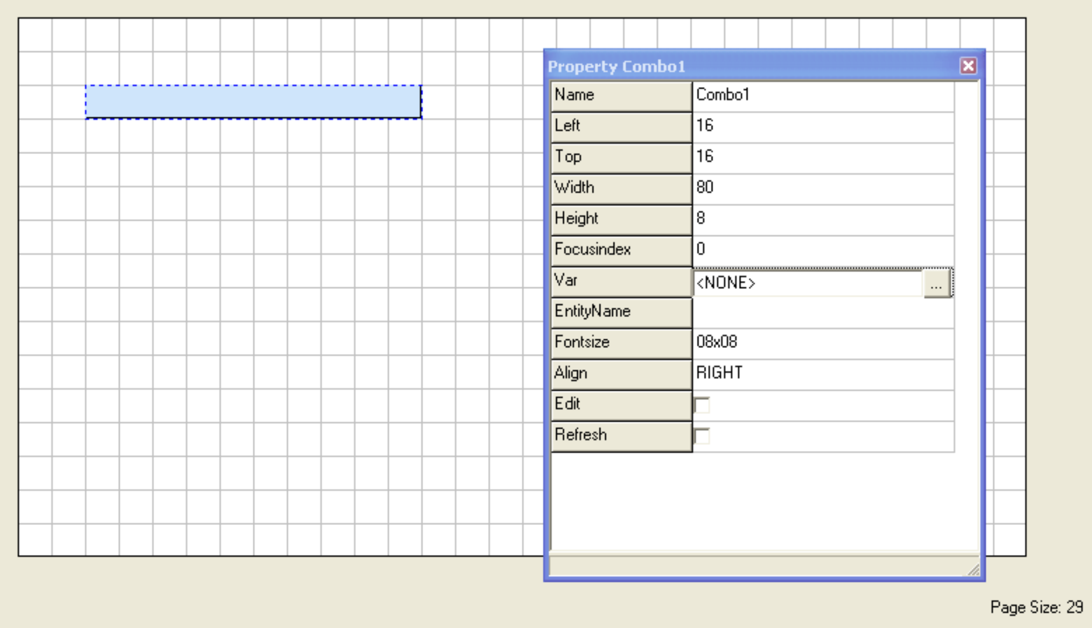
- Mode
- Decimals
- HideDecimals

In addition, only CJ_BIT or CJ_BYTE type entities can be joined to them.

Property	**Description**
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 receives focus when moving the cursor. The permissible range is 0…255. If set to 0, the element is never selected by the cursor. If set to a different value, defines after how many cursor movements it is 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, and so on.). Select this property and click the button that appears. A window opens and allows you to select one of the project entities to which you wish to link the value.
EntityName	Return the number of the element in case of an array. Otherwise is the same name of the Var.
Fontsize	Describes the font size used to write text. For example, if you use an 8 x 8 font size, a character is 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 other than zero).
Refresh	If the *Refresh* property is enabled, the variable is continuously requested. Otherwise, it is refreshed slowly as to avoid overloading communication between controllers.

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

Now add a **Combo** object to our page and open the properties window:

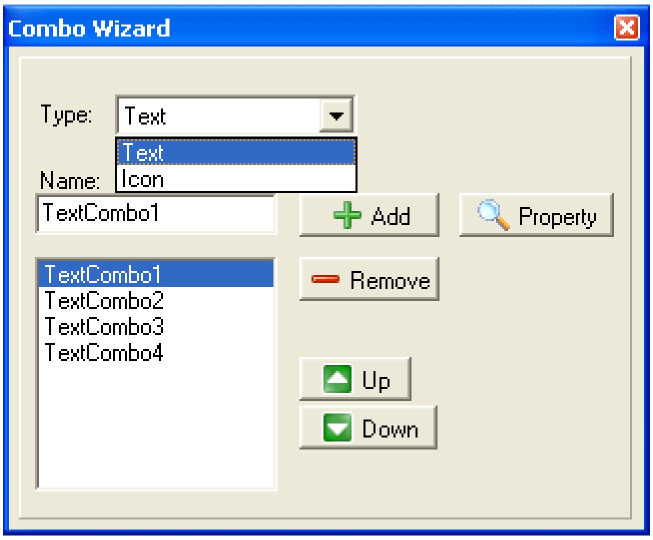


Property Combo1

Name	Combo1
Left	16
Top	16
Width	80
Height	8
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), click the button that appears to the right and select the desired variable from the Entity List.

Right-click the **Combo** element and select **Combo Wizard** from the context menu. The following window appears:



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, and click **Add**.

NOTE: The **Combo** element name must be C compliant. The characters like ° and % are not allowed. The displayed information can be different from its name. Use the **Property** button on the top right to change the element displayed text (including the characters ° and %).

The elements are added to the list in the order they were entered.

The first element is displayed when the linked entity takes value 0, the second element is displayed when it takes value 1, and so on.

To change the sequence, select the element you wish to move and click the **Up** or **Down** button as required.

Once you have added all the elements, you can proceed to their configuration.
To display the properties of an element, select it from the list and click **Property**.

To find out the meaning, refer to the properties of *Texts* and *Icons*.

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

4.6.10 Table

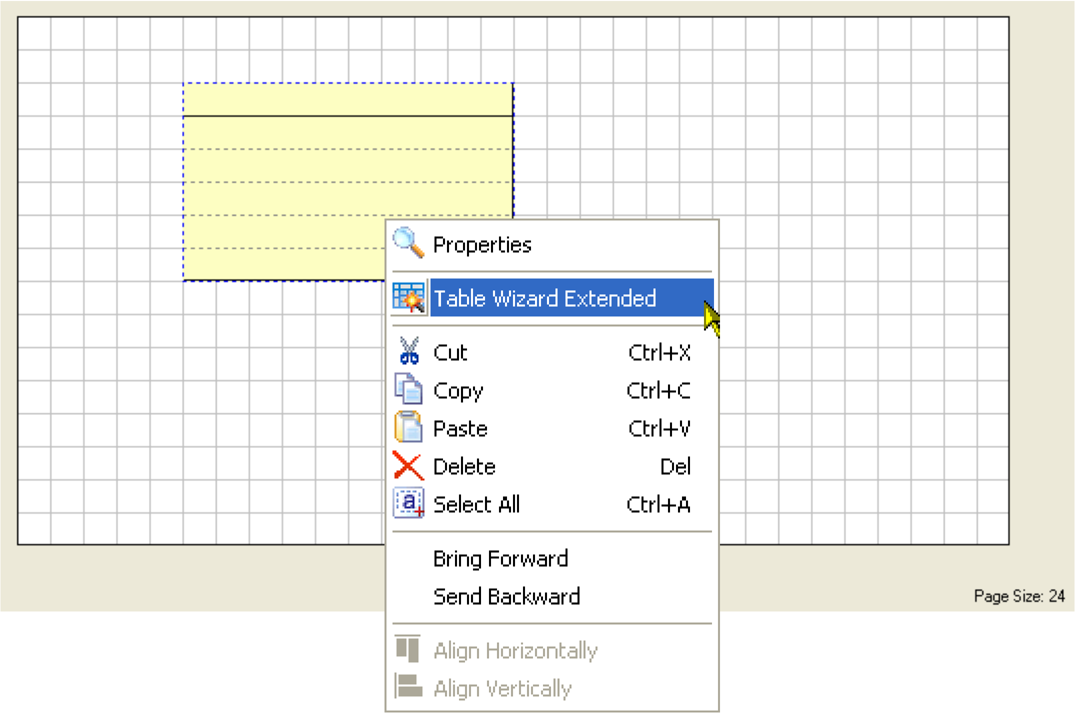
Tables are an object type designed to help you develop graphical interfaces and characterized by being scrollable.

Therefore, using tables, you can display large amounts of data without loading more than 1 page.

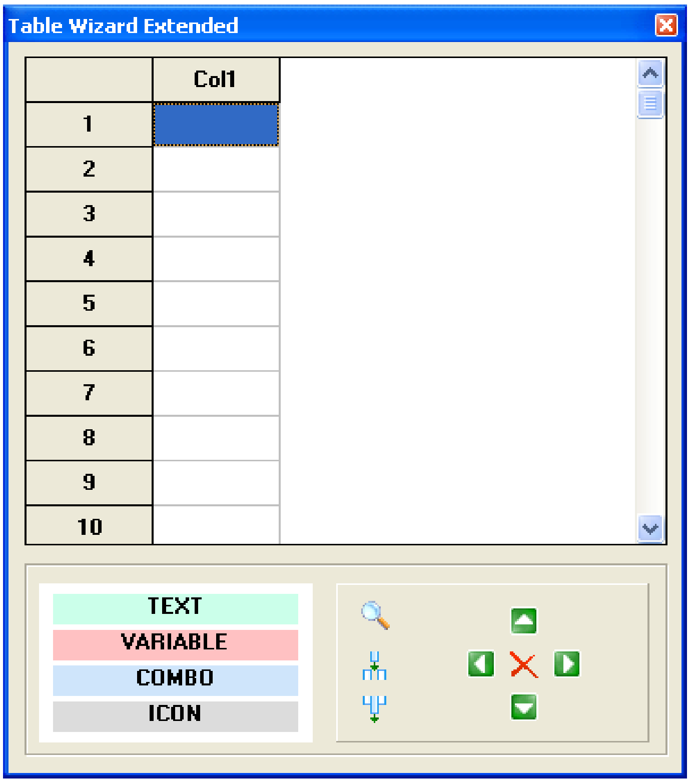
If you add a table to a page, you can display its properties:

Property	**Description**
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 are displayed, otherwise no borders are displayed.

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



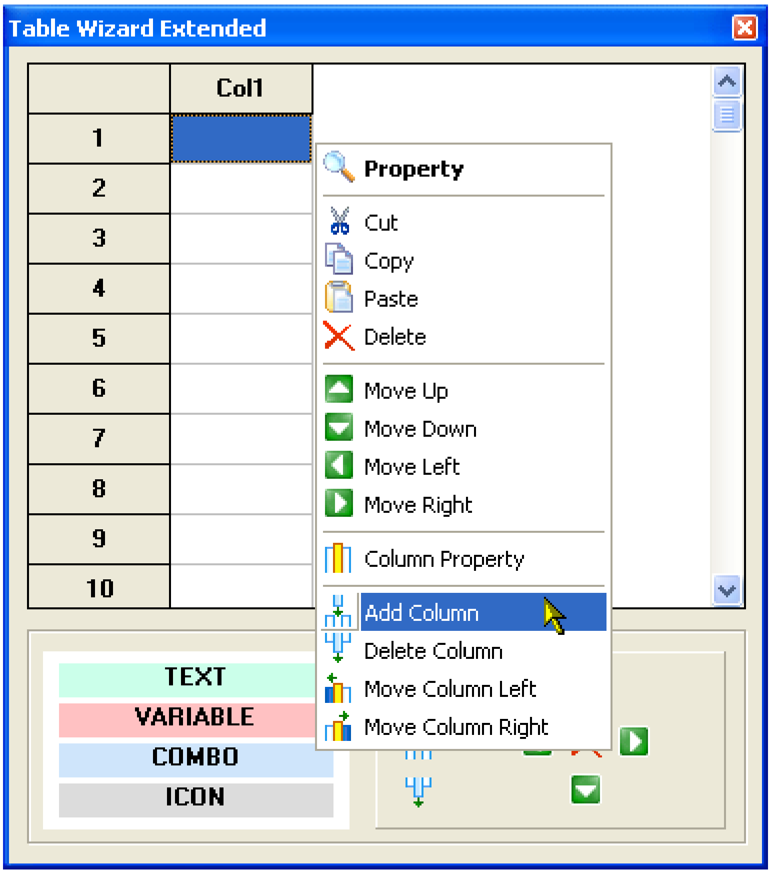
The following window appears:



The first thing you need to do to use a table is create its columns.

The **Table** object allows you to add from 1 to 4 columns. By default the table has 1 column active.

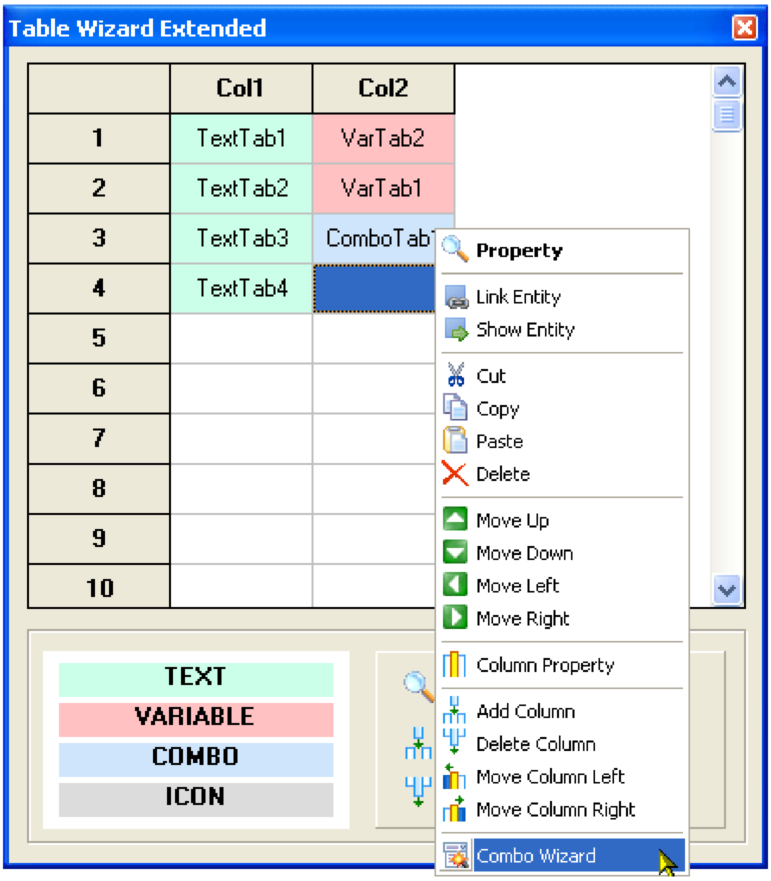
It is possible to create new columns using the contextual menu (right-click the column and select **Add Column**):



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

To insert the required elements to each column, select it from the list at the bottom and drag and drop it to the desired position in column.

The table do not allow empty cell. The requirement element is added at the first empty cell of the column.



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

It is possible to move the selected item into the table using the position buttons on the right-bottom of the window.

To move the selected item among rows and columns, you need to click the corresponding arrows.

To delete the selected item, you need to click the X in the middle.

To change the properties of an element, select it and select **Property** from the contextual menu, or double-click the focused element.

From the contextual menu (right-click 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 need to show the **Combo** Wizard to select which elements you wish to link to it.

Some of these operations can 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.

4.6.11 Line ╲

To make an EIML page more graphically attractive, you can also add lines to it by clicking ╲.

A line has the following properties:

Property	Description
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.

4.6.12 Rectangle □

To make an EIML page more graphically attractive, you can also add rectangles by clicking □ icon.

A rectangle has the following properties:

Property	Description
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.

5 Advanced Operations

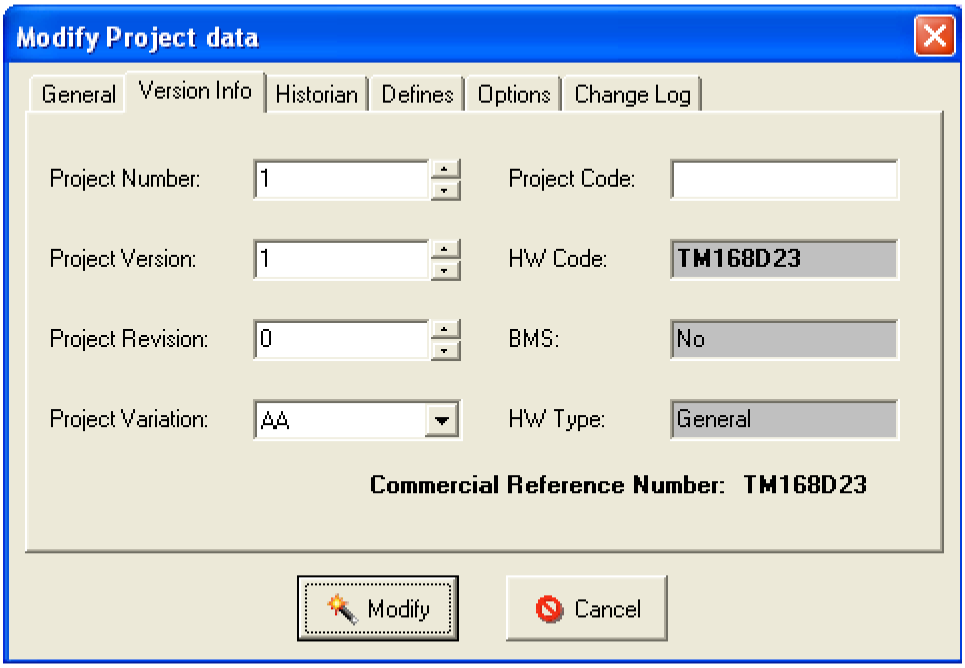
5.1 Project Settings

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 and size of an Event Historian, Global defines Entities used in algorithms.

To access these settings, select **Project→Property…** from the menu or click and open the project properties window.

5.1.1 Version Info

Version info windows:



Modify Project data

Project Number: 1

Project Code:

Project Version: 1

HW Code: TM168D23

Project Revision: 0

BMS: No

Project Variation: AA

HW Type: General

Commercial Reference Number: TM168D23

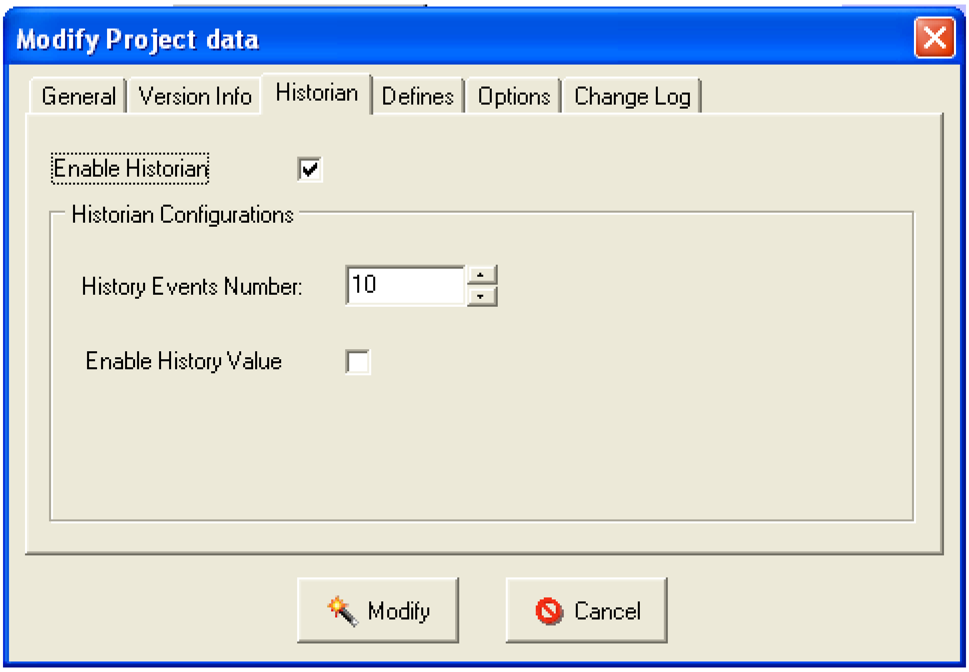
In the Version Info window, you can set the project number, version, revision, variation, and the project code for the user project.

The project Variation was created to identify when 2 projects code are equals, but when they have different EIML pages language or different hardware.

The HW Code, BMS selection and HW Type are read only values.

5.1.2 Historian

In the **Historian** window, you can enable and configure the event and alarm recording functions available in M168 controllers.



Enable Historian ☑

Historian Configurations

History Events Number: 10

Enable History Value □

Enable Historian check box links the Historian Library to the program.

If you do not wish to implement the event logging function, leave the **Enable Historian** check box unchecked, it saves the Flash memory usage.

If you wish to implement the event logging function, you also 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 are stored in the memory of a controller, and when the number is exceeded, the oldest events are overwritten.

History Events Number is limited to 100.

The following information is stored in the Event Historian:

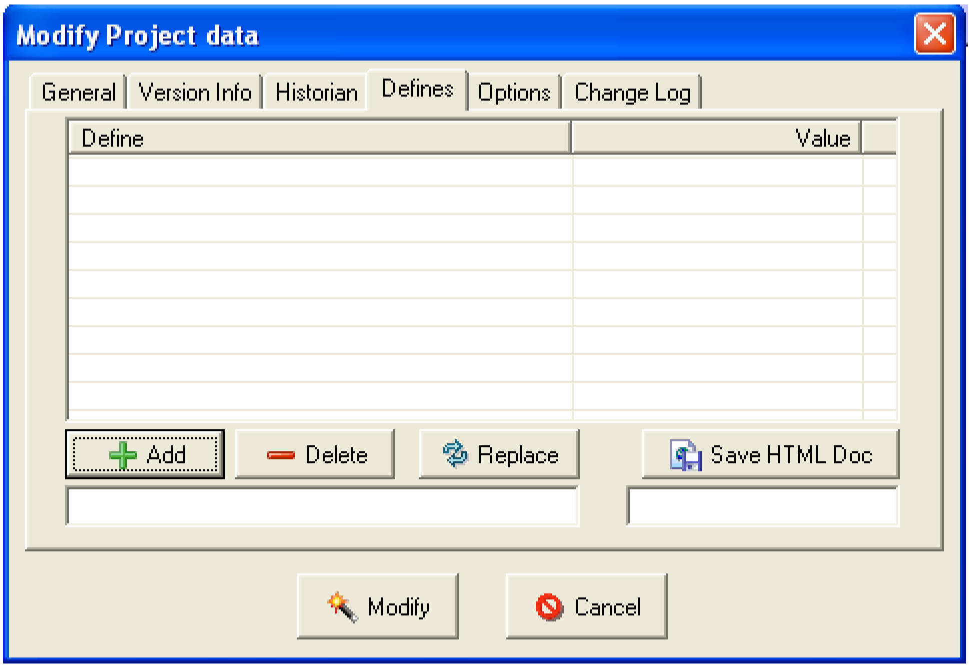
Data	**Type**	**Description**
Date	**CJ_DATETIME**	Date and time when the event took place.
Code	**CJ_WORD**	Event identification code.
Progressive	**CJ_LONG**	Progressive event number.
Value	**CJ_SHORT**	Possible value associated to the event.

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 *SoHVAC Standard Library User Guide*).

5.1.3 Project Defines

The project **Defines** window allows you to define constants and assign values to them.



To add a **Define**, type a unique name in the left box under the list and click **Add**. The **Define** is added to the list.

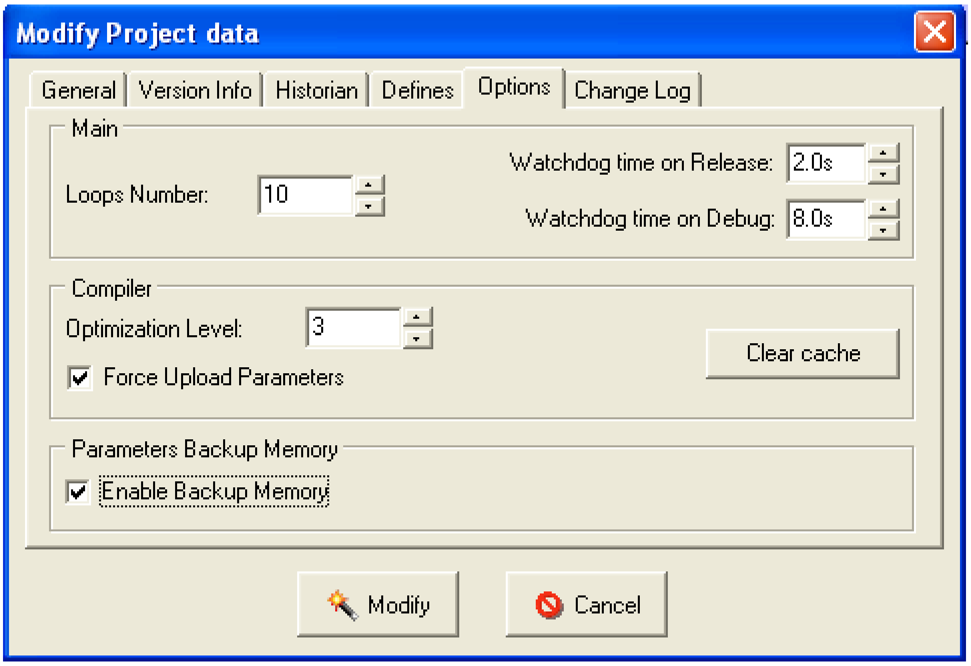
If you wish to assign a certain value to it, type it in the right box under the list.

To change a **Define**, select it from the list, change it as required in the left and right boxes under the list, and click **Replace** to confirm. To delete it, click **Delete**.

5.1.4 Project Options

The project **Options** window allows you to define various settings for program execution, compiler optimization level, watchdogs and backup memory management.

Only expert users are advised to modify these options.



Loops Number: 10

Watchdog time on Release: 2.0s

Watchdog time on Debug: 8.0s

Optimization Level: 3

☑ Force Upload Parameters

☑ Enable Backup Memory

Property	Description
Loops Number	Specify the number of *actions* calculated in a group. For large projects, by increasing this number it is possible to speed up the main program.
Watchdog time on Release	Watchdog time for the projects compiled in Release (without Debugger protocol).
Watchdog time on Debug	Watchdog time for the projects compiled in Debug (with Debugger protocol).
Optimization Level	Optimization level of the compiler. The bigger is this parameter the more the code is optimized.
Force Upload Parameters	Forces the use of the downloaded program parameters upon the first execution after the download. Otherwise, the existing parameters in the controller are used. Force Upload Parameter unchecked: parameter values of controller are used during the initial state. Force Upload Parameter checked: parameter values of SoHVAC software are used during the initial state.
Clear cache	It cancels the compiling cache of the projects.
Enable Backup Memory	The flag Enable Backup Memory allows you to enable or inhibit the copy of the configuration and application parameters to non-volatile memory using system commands.

If the project you download has had the number of parameters modified (parameters added or deleted) relative to the original program in the controller, the values of the parameters in the controller may not correspond to the program logic. This may lead to unintended consequences if you do not select the Force Upload Parameter option.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

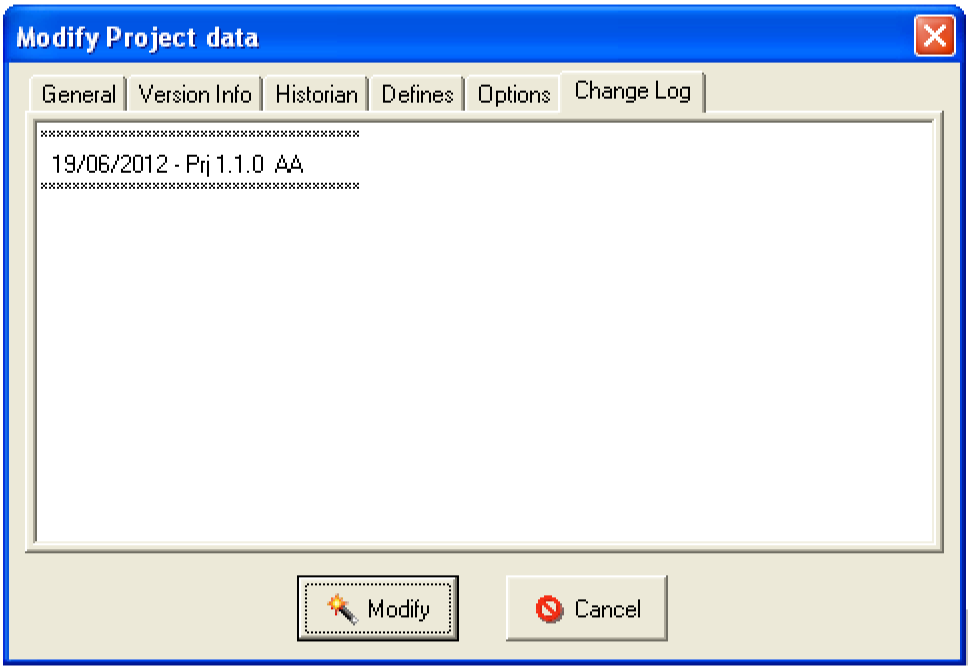
Select the Force Upload Parameter option if you have changed the number of parameters used in your project.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

5.1.5 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.



```
****************************************************
 19/06/2012 - Prj 1.1.0  AA
****************************************************
```

Clicking the right mouse button on the window, it is possible to view the corresponding menu in addition to the classic Cut/Copy/Paste actions.

This allows:

- **Printing** the content of the **Change Log**.
- **Adding**, at the current cursor position, a new note (a few rows with headings containing the date and the current project version).

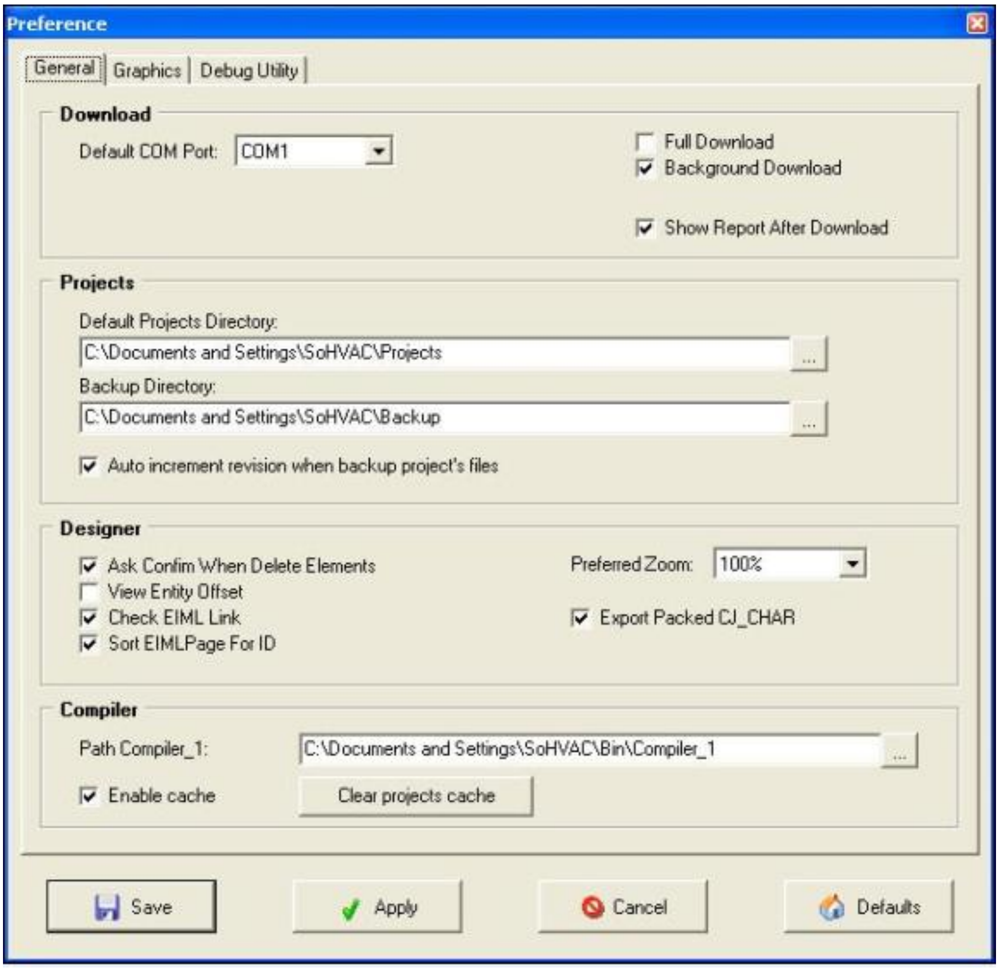
5.2 Preference Menu: Configuration

With the **Tools→Preference** menu, it is possible to access to the following configuration tabs:

- **General**
- **Graphics**
- **Debug Utility**

5.2.1 General

The General tab contains the **Download**, **Projects**, **Designer** and **Compiler** sections:



Download

Default COM Port: COM1

□ Full Download
☑ Background Download
☑ Show Report After Download

Projects

Default Projects Directory: C:\Documents and Settings\SoHVAC\Projects

Backup Directory: C:\Documents and Settings\SoHVAC\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%

☑ Export Packed CJ_CHAR

Compiler

Path Compiler_1: C:\Documents and Settings\SoHVAC\Bin\Compiler_1

☑ Enable cache

Clear projects cache

Download

Options	Description
Default COM port	Serial port used to connect to the controller.
Full download	Forces the full downloading of the entire executable file. To speed up the downloading options, it is possible to unselect this option.
Background download	Allows you to work on the project during the download phase.
Messages: Check "Disable Modem Ctrl At Startup" and Remove "Serial Enumerator" flag	If one of the above messages appears, then the properties for the port must be changed. Open the device manager and select the COM port which is for communication to the M168 controller. Select advanced properties and check the Disable Modem Ctrl At Startup and uncheck Serial Enumerator checkbox.
Show report after download	If this option is enabled, a report appears displaying previous application and new application. Applies only for the **TM168●21●●** controllers.

Projects

Options	Description
Default Projects Directory	Default directory prompted for project saving.
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 the backup operations are performed on project files and the revision number increases automatically. Otherwise the user has to perform manually this operation.

Designer

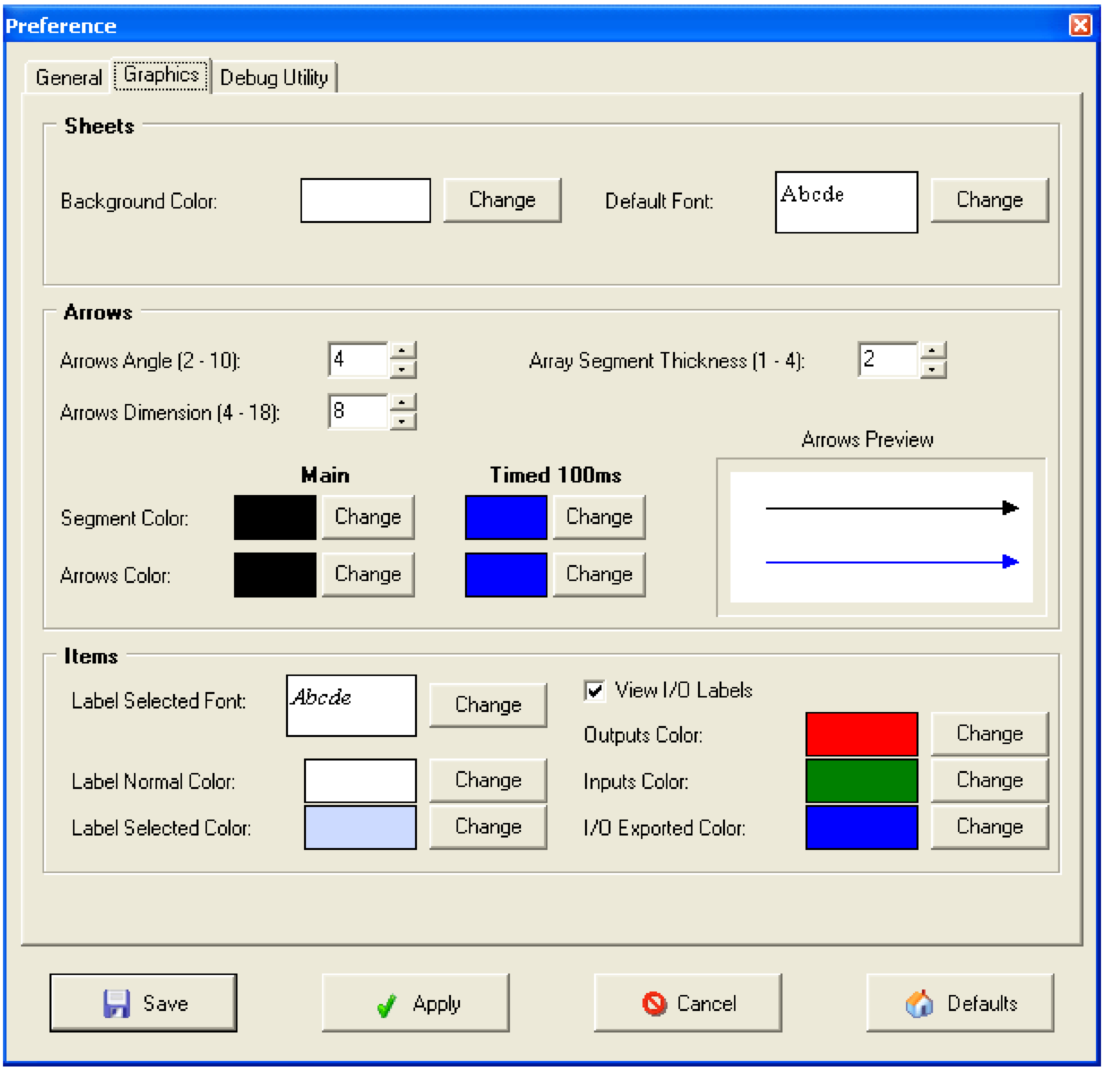
Options	Description
Ask confirm when delete elements	Activates or deactivates the request to confirm the deletion of entities or EIML components.
View Entity Offset	If enabled, allow to display the offset property below the entity.
Check EIML link	Verifies if all the connectable elements in the graphic interface section are properly connected to entities. The verification is made 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 are displayed in alphabetical order.
Preferred zoom	Specify the default zoom upon loading of a project. Afterwards the zoom level can be changed easily using the shortcut menu located in the toolbar.
Export Packed CJ_CHAR	If enabled, the Modbus exportation of the string type is done by groups (made of 2 byte for each register).

Compiler

Option	Description
Path Compiler_1	Specifies the path where the compiler program is located.
Enable cache	It enables the use of the compiling Cache.
Clear projects cache	It cancels all the Cache of the used projects.

5.2.2 Graphics

The **Graphics** tab contains the **Sheets**, **Arrows**, and **Items** sections:



Background Color: ⬜

Default Font: Abcde

Arrows Angle (2 - 10): 4

Array Segment Thickness (1 - 4): 2

Arrows Dimension (4 - 18): 8

	Main	Timed 100ms
Segment Color:	⬛	🟦
Arrows Color:	⬛	🟦

Label Selected Font: *Abcde*

☑ View I/O Labels

Outputs Color: 🟥

Label Normal Color: ⬜

Inputs Color: 🟩

Label Selected Color: ▨

I/O Exported Color: 🟦

Sheets

Options	Description
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

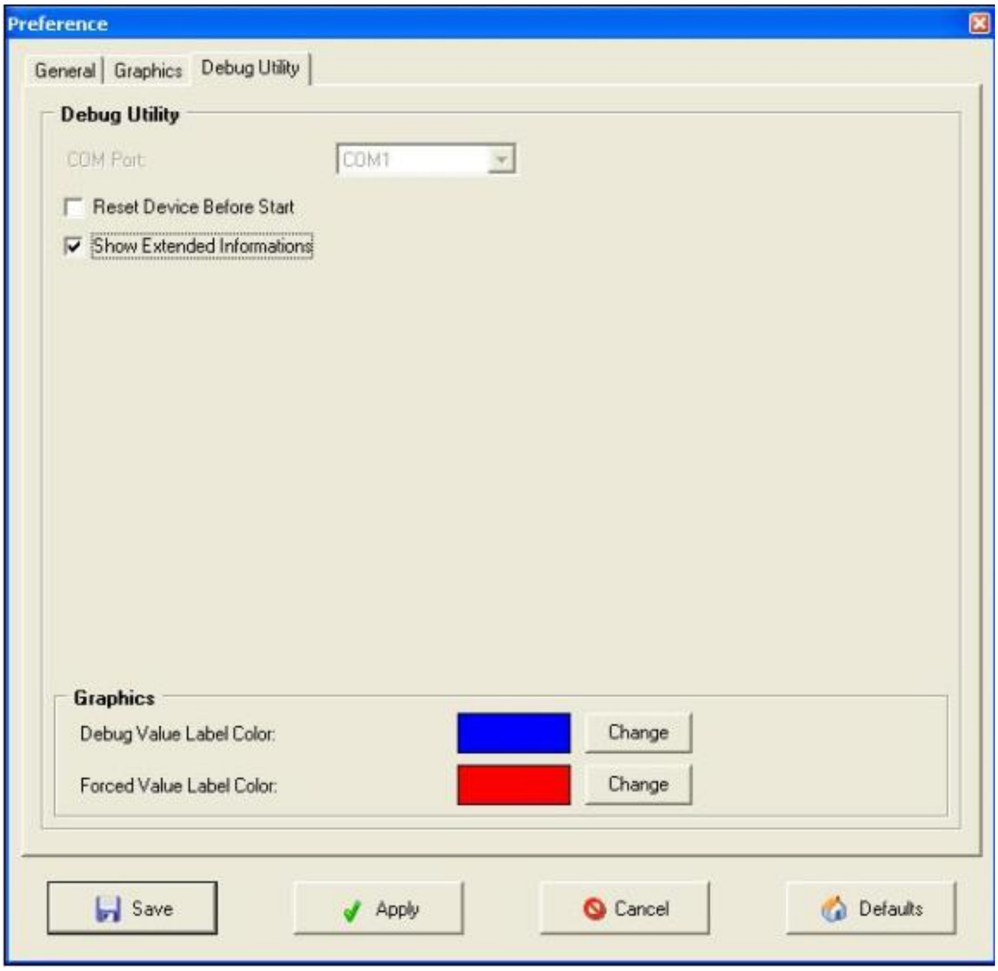
Options	Description
Arrows Angle	Modifies the aperture angle of the arrow tip connected with the entity.
Arrows Dimension	Modifies the dimension of the arrow tip connected with the entity.
Array Segment Thickness	Modifies the thickness of the lines connected with the array entity (entities with the array 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 possible to specify the color to be used if the path is to be calculated during the main task, or every 100 ms (refer to the *execution task* paragraph).
Arrows Preview	Preview window where examples of arrows settings are viewed.

Items

Options	Description
Label Selected Font	Defines the font to be applied to the selected entities.
Label Normal Color	Defines the background color of the text for entities not selected.
Label Selected Color	Defines the background color of the text for the selected entities.
View I/O Labels	Shows/hides the label names of the input/output pins for the sheets, algorithms and libraries.
Outputs Color	Defines the color of the entities output pin.
Inputs Color	Defines the color of the entities input pins.
I/O Exported Color	Defines the color of the entities inputs/outputs pins which are exported.

5.2.3 Debug Utility

The Debug Utility tab contains the configuration options used to debug.



Debug Utility

COM Port: COM1

☐ Reset Device Before Start

☑ Show Extended Informations

Graphics

Debug Value Label Color: 🟦

Forced Value Label Color: 🟥

Options are the following:

Debug Utility

Options	Description
COM Port	Serial port used with the controller during debug operation. This value is read only. It is the same as the Download Port.
Reset device before start	Reset the controller at each start of the debugger.
Show Extended Informations	If selected, allows to show more info on the state bar application like task index where program break (refer to *Breakpoints*).

Graphics

Options	Description
Debug Value Label Color	Defines the value label color of values calculated in debug mode.
Forced Value Label Color	Defines the value label color of values forced in debug mode.

5.3 Considerations in Project Development

1. Resources are finite

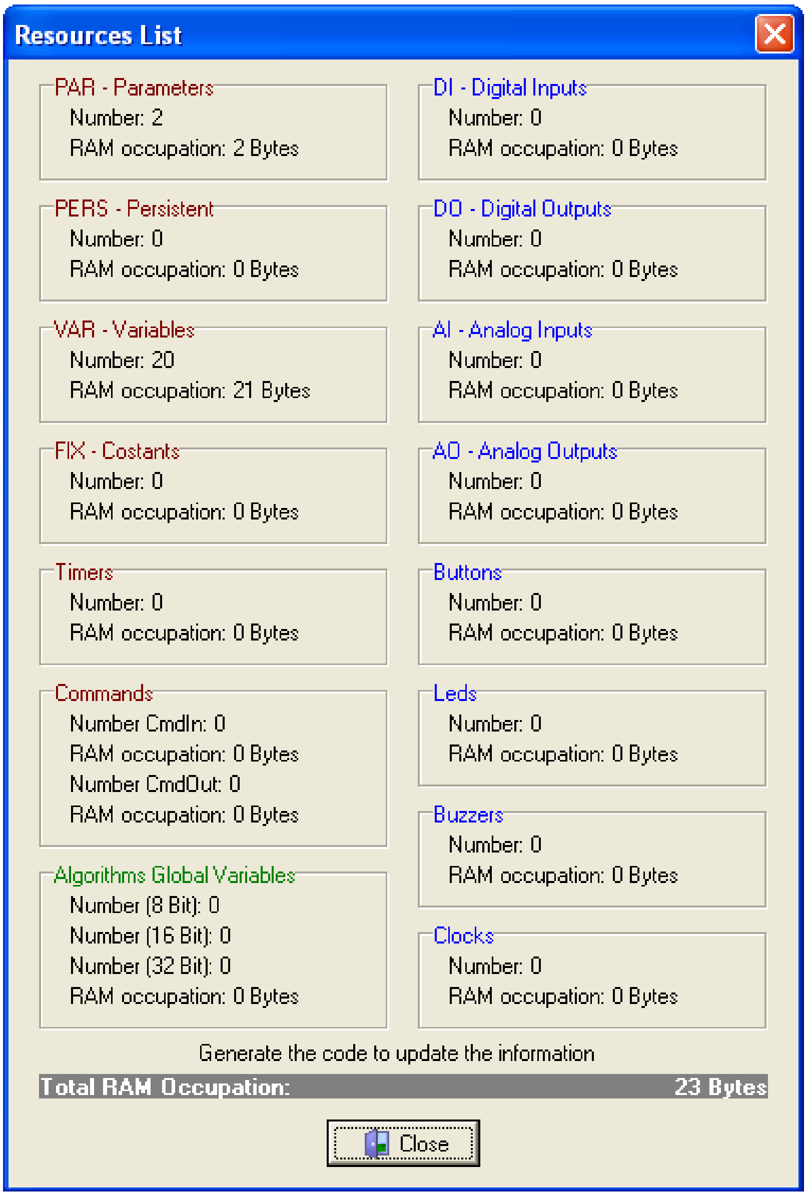
The controllers have finite RAM and Stack memory resources. This information is available in the *Modicon M168 Controller Hardware Guide*.

This table below indicates the other finite resources available in the controller.

Resource limitation:

Par	4000
Var	4000
Pers	1535
Fix	1023
Pages	767
All the others	255 (DI, DO, AI, AO, buttons, clocks)

To view the current resource utilization, select **Tools->Resources List** from the menu:



Resources List

PAR - Parameters
Number: 2
RAM occupation: 2 Bytes

PERS - Persistent
Number: 0
RAM occupation: 0 Bytes

VAR - Variables
Number: 20
RAM occupation: 21 Bytes

FIX - Costants
Number: 0
RAM occupation: 0 Bytes

Timers
Number: 0
RAM occupation: 0 Bytes

Commands
Number CmdIn: 0
RAM occupation: 0 Bytes
Number CmdOut: 0
RAM occupation: 0 Bytes

Algorithms Global Variables
Number (8 Bit): 0
Number (16 Bit): 0
Number (32 Bit): 0
RAM occupation: 0 Bytes

DI - Digital Inputs
Number: 0
RAM occupation: 0 Bytes

DO - Digital Outputs
Number: 0
RAM occupation: 0 Bytes

AI - Analog Inputs
Number: 0
RAM occupation: 0 Bytes

AO - Analog Outputs
Number: 0
RAM occupation: 0 Bytes

Buttons
Number: 0
RAM occupation: 0 Bytes

Leds
Number: 0
RAM occupation: 0 Bytes

Buzzers
Number: 0
RAM occupation: 0 Bytes

Clocks
Number: 0
RAM occupation: 0 Bytes

Generate the code to update the information

Total RAM Occupation: 23 Bytes

For example, an algorithm with 2 array inputs of 100 CJ_BYTE and an output of 100 CJ_BYTE uses 300 bytes in the stack.

The M168 has a 4 kbyte stack.

If the stack overflows, the controller is reset.

Function CJ_BIT CJ_Stack_Error_Read(void) can be used to monitor stack overflow, as can the `MemoryStack_Error` function block.

For more information, refer to the SoHVAC C Programming Language User Guide or the SoHVAC Standard Library User Guide.

2. Using fixed point and floating point mathematics

Floating point mathematical functions are only available inside the C language algorithms. Floating point is not available within function blocks.

For more important information about mathematical operations, refer to SoHVAC C Programming Language User Guide and/or the SoHVAC Standard Library User Guide.

5.4 Project Backup

The SoHVAC development environment has an automatic backup function that allows you to copy the project files in the special directory shown in the **General** tab.

Backup options are configured in the **Tools**→**Preference** window, general tab

The directory where the backup is performed is the following:

<Default directory>\Prj_<project number >\Ver._<project version>.<project revision>

To make project backup, use menu **File**→**Backup Project**.

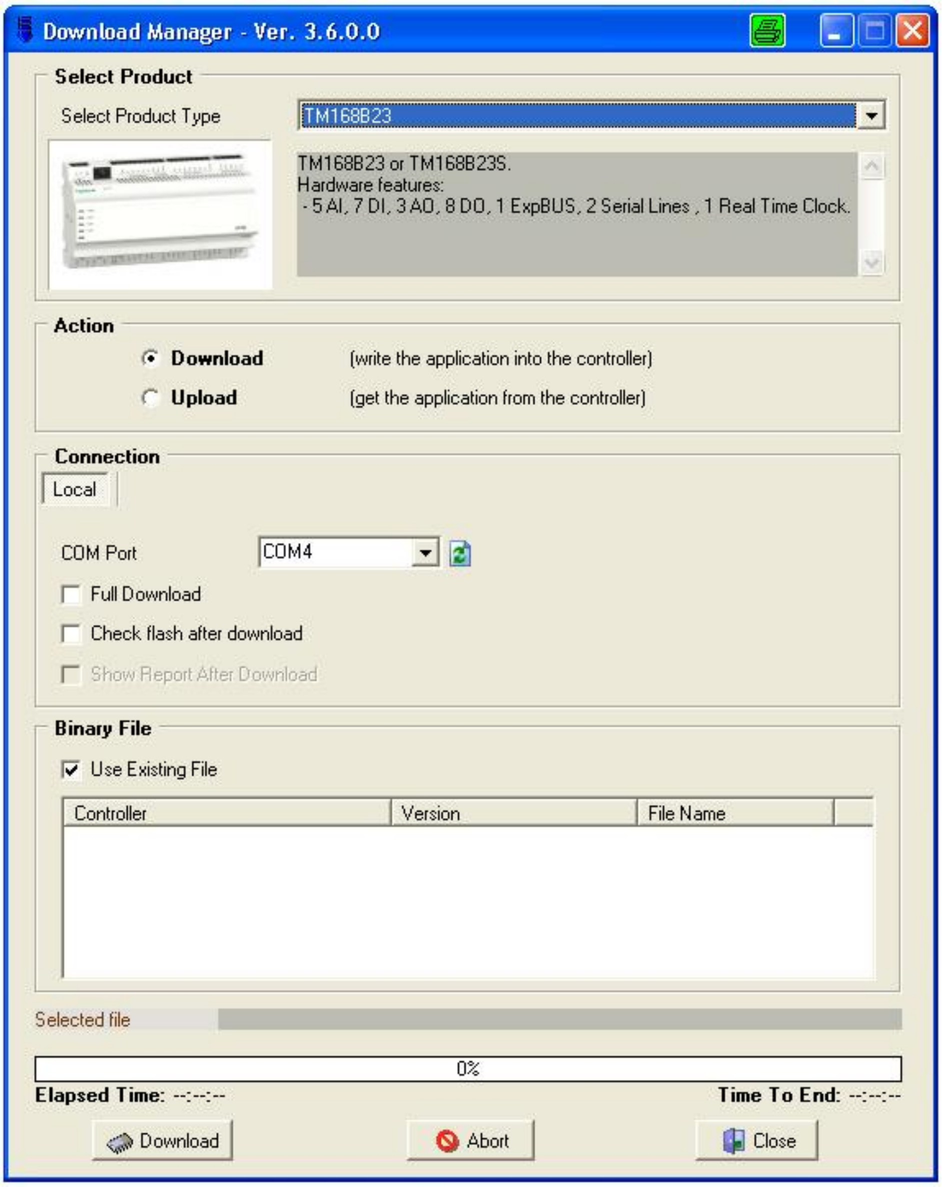
5.5 Product Programming Tool (Download Manager)

The download manager tool is used to download application programs into the M168 controller. You can also use the download manager tool to update the firmware in the M168 expansion modules and in the M168 remote displays.

NOTE: It is also possible to install the download manager separately from the SoHVAC software. This is used in the production process, where no changes are made to the application programs.

NOTE: The download manager only updates the firmware of the M168 expansion modules and M168 remote displays. The firmware of the controller is downloaded with the application program.

To start the **Download Manager** from SoHVAC, select the **Tools→Download Manager** menu.



Download Manager - Ver. 3.6.0.0

Select Product

Select Product Type: TM168B23

TM168B23 or TM168B23S.
Hardware features:
- 5 AI, 7 DI, 3 AO, 8 DO, 1 ExpBUS, 2 Serial Lines , 1 Real Time Clock.

Action

● **Download** (write the application into the controller)

○ **Upload** (get the application from the controller)

Connection

Local

COM Port: COM4

□ Full Download

□ Check flash after download

□ Show Report After Download

Binary File

☑ Use Existing File

Controller	Version	File Name

Selected file

0%

Elapsed Time: --:--:--

Time To End: --:--:--

5.5.1 Local Download

To program a product, first select a product from the drop-down menu. A preview and a brief description of the product selected are shown in the window.

In the **Action** section, select **Download** to program the application in the controller.

In the **Connection** section (**Local** tab), you can select the serial port where the programming cable is connected.

The **Full Download** option forces the full downloading of the entire executable file. To speed up the downloading options, you can unselect this option.

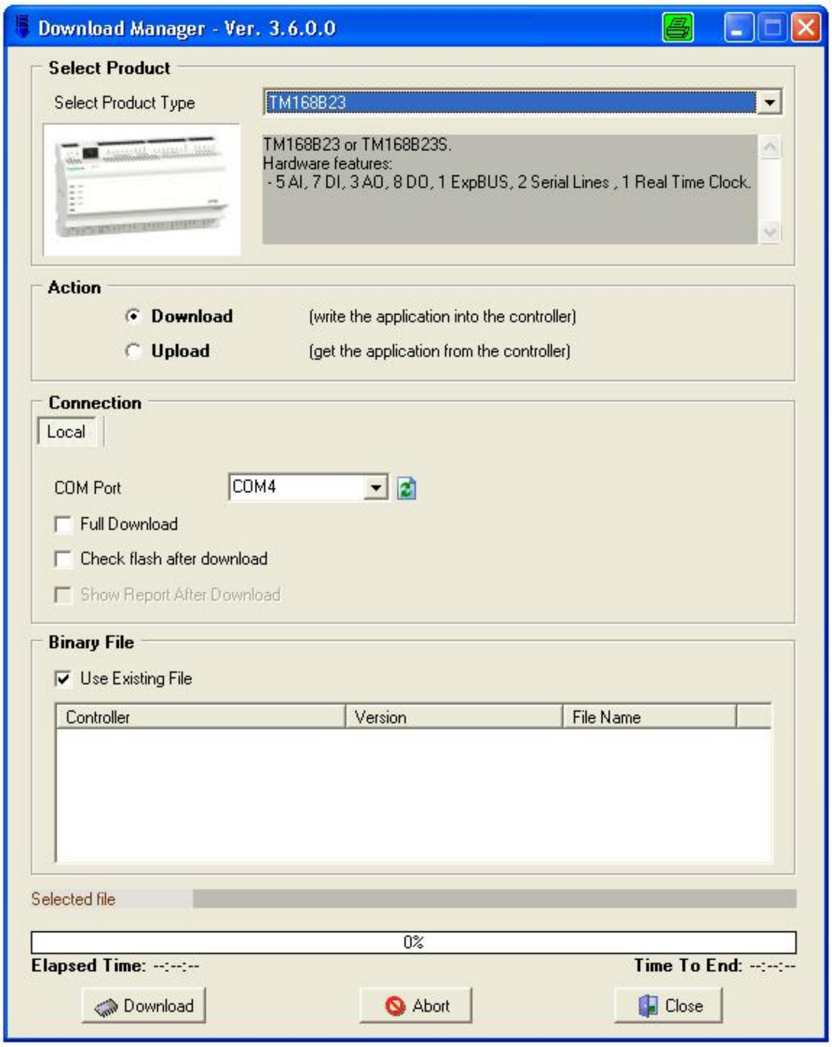
The **Check Flash after Download** option reads and compares the downloaded application program with the original application program. A message appears if an error is detected.

In the **Binary File** section, the list of the available compatible binary files is displayed.

By clearing the **Use Existing File** checkbox, you can browse the binary file to be downloaded.

To begin download, click the **Download** button. In the lower part of the window, you can see the process; in particular, the **Elapsed Time**, the **Time To End**, and percentage downloaded are shown.

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



Download Manager - Ver. 3.6.0.0

Select Product

Select Product Type: TM168B23

TM168B23 or TM168B23S.
Hardware features:
- 5 AI, 7 DI, 3 AO, 8 DO, 1 ExpBUS, 2 Serial Lines , 1 Real Time Clock.

Action

● **Download** (write the application into the controller)
○ **Upload** (get the application from the controller)

Connection

Local

COM Port: COM4

□ Full Download
□ Check flash after download
□ Show Report After Download

Binary File

☑ Use Existing File

Controller	Version	File Name

Selected file

0%

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

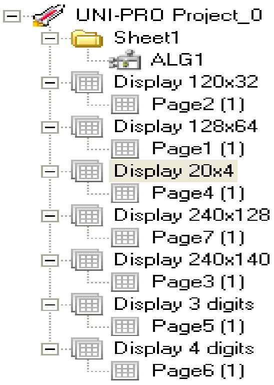
5.6 EIML Pages Advanced Topics

5.6.1 Navigating through the Pages

This chapter explains how to recall a page from another page and how the pages are interconnected.

Pages information is stored in the controller and their creation coincides with the development of control algorithms.

In the project tree, pages are displayed below the sheets in one or more folders depending on the number of displays the specific controller has.



```
UNI-PRO Project_0
    Sheet1
        ALG1
    Display 120x32
        Page2 (1)
    Display 128x64
        Page1 (1)
    Display 20x4
        Page4 (1)
    Display 240x128
        Page7 (1)
    Display 240x140
        Page3 (1)
    Display 3 digits
        Page5 (1)
    Display 4 digits
        Page6 (1)
```

This multiple-display management feature enables displaying pages created for a certain user interfaces with more than one browser type (for example, you can actually display pages written for a 20 x 4 alphanumeric display on a 240 x 140 graphic display).

In general, navigation between pages is made possible through a mechanism that uses the **Load Page** command (refer to *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 is 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 also need to define the required browser type in the Param input, according to the following:

- 4 x 20 ALPHANUMERIC DISPLAY = 3
- 128 x 64 GRAPHIC DISPLAY = 4
- 240 x 140 GRAPHIC DISPLAY = 6
- 120 x 32 GRAPHIC DISPLAY = 7

Setting **PreviousPage**, **NextPage** and **TimeoutPage** properties, it is possible to define navigation between pages.

By assigning those 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 default page is loaded.

5.6.2 Password-Protected Information

In many applications, access to certain data by all users needs to be prevented in order to ensure a good machine control.

Changing improperly set-up of a controller or operating parameters can lead malfunctions or shutdown of the whole system.

The protection system adopted is based on 5 protection levels that can be assigned to each **EIML** page by setting the **Level** property.

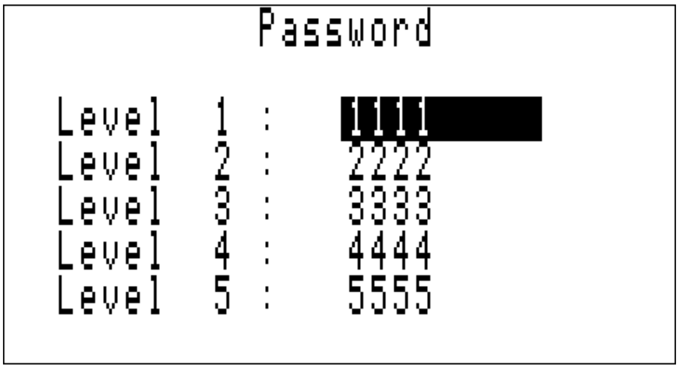
Each level is linked to a different password in the controller.

When display requests a page whose **Level** property is set to a value higher than 0, the controller verifies 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 no keys are pressed on the keypad during a pre-set period of time (system parameter, default 60 seconds), the authorization level is reset.

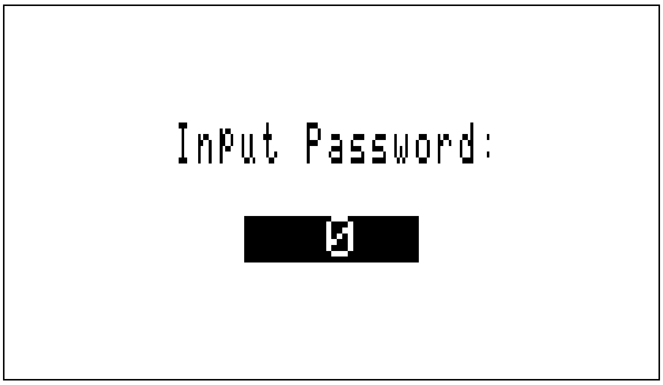
For example, the controller has the following password set-up page:



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 prompts the user to enter a password.



Input Password:

0

If the password entered is 1111, the user is 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 is displayed to the user, who will have free access to all pages having a level equal to or lower than 3.

After a pre-set period of time (system parameter, default 60 seconds), the authorization level is reset.

5.6.3 Simulating EIML Pages

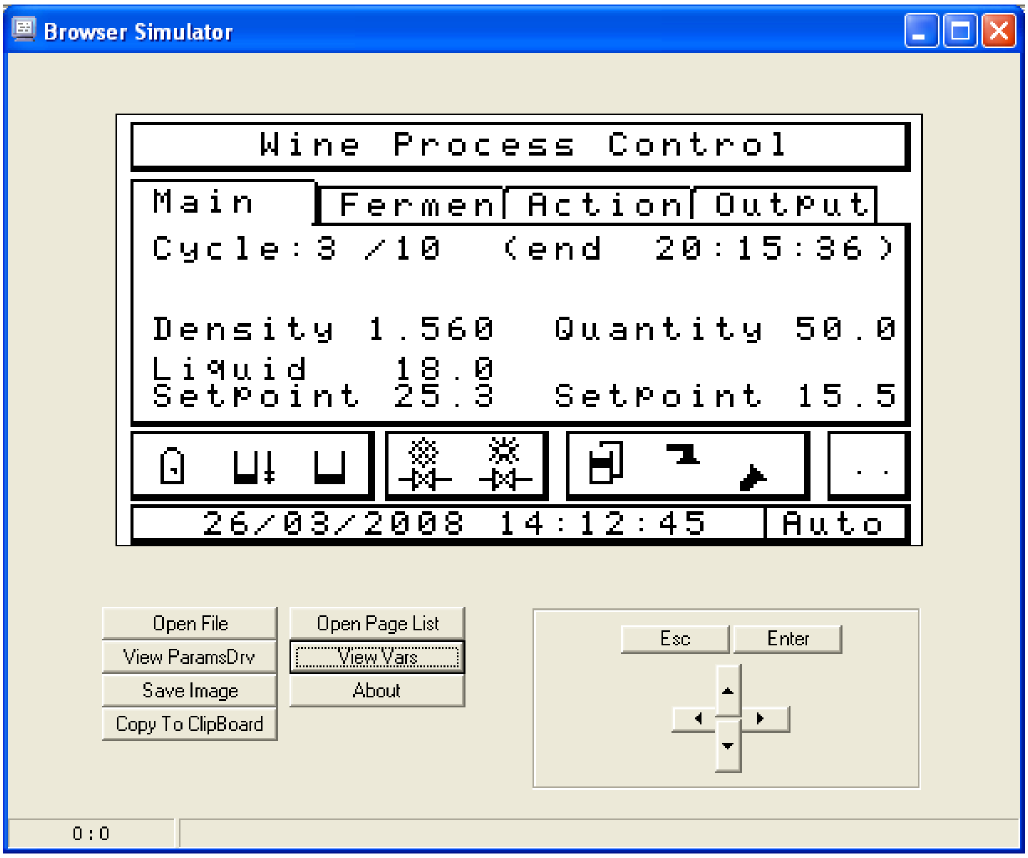
While designing the graphical interface, you can simulate the actual behavior of the pages you are creating to verify that they are correctly implemented.

SoHVAC has a built-in tool that allows you to display the output of a graphical interface page as if it is being executed in the chosen display.

To enable this tool:

- Go to the page you wish to simulate and select **Project→EIML Simulator** from the menu, or
- Click the icon in the toolbar.

Below is an example of a simulation of a 240 x 140 pixel graphic page:



Wine Process Control

Main | Fermen | Action | Output

Cycle:3 /10 (end 20:15:36)

Density 1.560 Quantity 50.0
Liquid 18.0
Setpoint 25.3 Setpoint 15.5

26/03/2008 14:12:45 Auto

0:0

The simulator allows you to verify:

- All elements inside the page are properly displayed (texts, variables, icons, tables, and so on).
- All actions have been correctly implemented (moving through fields, editing and sending commands).
- The pages are properly identified.

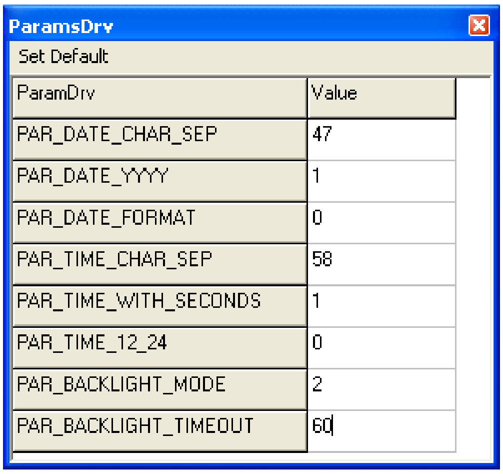
Open File/Open Page List

On the left side of the simulator window, there is an **Open file** button. Click this button to load an **EIML** page saved in binary format. Instead, by clicking the **Open Page List** button, you can load a group of pages saved.

When the simulator is launched from within the SoHVAC, the group of pages in the current project is loaded, and the page being edited is proposed as the first page.

View ParamsDrv

By clicking **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

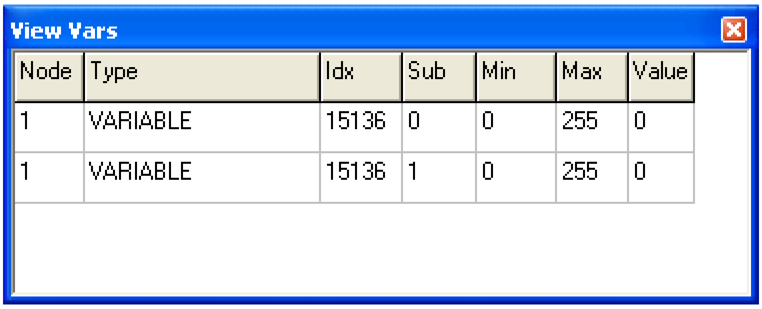
For example, you can set date format based on which characters can be used as time and date separators, whether a year can be displayed as 2-digit or 4-digit values.

ParDrv	Description	Min.	Max.	Default Value
PAR_DATE_CHAR_SEP	Character used for the date separator (ASCII code)	32 (space)	126 (~)	47 (/)
PAR_DATE_YY	Year with 2 or 4 digit	0 (yy)	1 (yyyy)	0 (yy)
PAR_DATE_FORMAT	Date format: 0 = dmy 1 = mdy 2 = ymd	0	2	0
PAR_TIME_CHAR_SEP	Character used for the time separator (ASCII code)	32 (space)	126 (~)	58 (:)
PAR_TIME_WITH_SECONDS	Hide or view the seconds	0 (hide)	1 (view)	-
PAR_TIME_12_24	Time mode: 0 = 24 h 1 = 12 h AM/PM	0	1	0
PAR_BACKLIGHT_MODE	Backlight mode: 0 = OFF continously 1 = ON continuously, 2 = Time out	0	2	2
PAR_BACKLIGHT_TIMEOUT	Number of seconds after the last key pressed that caused the backlight turn off (if backlight mode = 2)	0	240	60

View Vars

By clicking the **View Vars** button, you can:

- Display a list of variables used in the pages.
- Change their value during simulation, if needed.



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

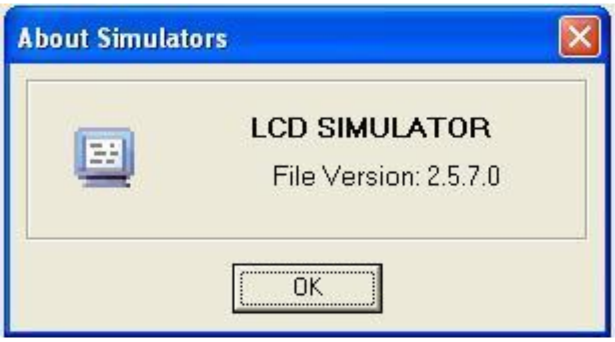
A row is added for each variable required by the page, which summarizes the main properties and can also be used to change their values.

Save Image/Copy to Clipboard

The Save Image and Copy to clipboard buttons are 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.

About

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



LCD SIMULATOR

File Version: 2.5.7.0

Navigation Buttons

The buttons **ESC**, **ENTER**, **UP**, **DOWN**, **LEFT**, and **RIGHT** simulate the behavior of the corresponding keys found on the user interfaces and allows you to perform operations such as:

- moving the cursor (**UP** and **DOWN**)
- editing a value (**ENTER**, **UP**, **DOWN**, **ENTER**, or **ESC**)
- browsing through the pages (**ENTER** an element to which a page load command is linked, **ESC** to return to the previous page)

5.6.4 Conditional Visibility

The conditional visibility allows you to hide the visualisation of entities. Those entities are only visible if a particular configuration is made, or a particular status reached.

In other words, configuration entities that are not relevant to the machine state are not shown by the user interface.

The user can focus on remaining displayed entities.

The conditioned visibility can be used with variables, parameters and persistent linked to the EIML pages (with variables, combos and tables).

To use this functionality, it is required to follow some essential steps.

1. To select the **CondVisible** property for all the interested entities
2. To export all the interested entities with **Export Entities** tool (enable to index the value).
3. To create the algorithms to rule the interested entities display or hide.
4. Use firmware function `void CJ_SetCondVisBit(word idx, bool value)`

In the alphanumeric displays and in the graphic displays, all the entities hidden with conditional visibility are substituted with points … and are not editable.

Firmware Function void CJ_SetCondVisBit (word idx, bool value)

This function allows to rule the visibility of an exported variable at the idx address on Export table from the algorithm.

Function uses 2 parameters:

1. word idx: is the ID of the entity exported on **Export** table (**Tool→Export Entities**).
2. bool value: get the entity visibility (this parameters can be a boolean condition result):
   - If value = 1 (or true condition), the entity is hidden.
   - If value = 0 (or false condition), the entity is shown.

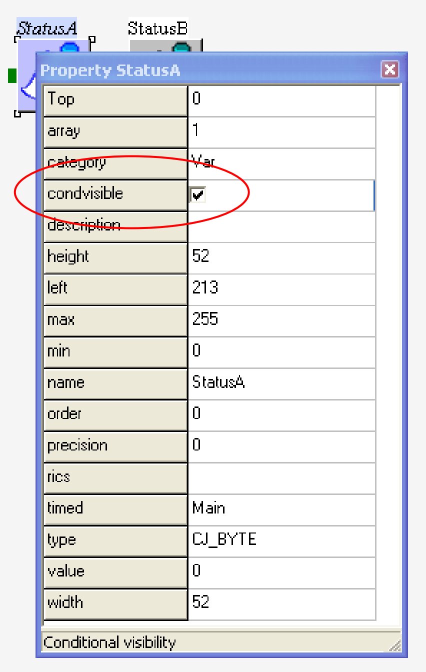
A function call as `CJ_SetCondVisBit(idx, 1)` made the entity at the idx address hidden in EIML.

A function call as `CJ_SetCondVisBit(idx, 0)` made the entity at the idx address visible in EIML.

Example

To condition the visibility of 2 variables, StatusA and StautsB with a CJ_BIT parameter P001, follow the steps:

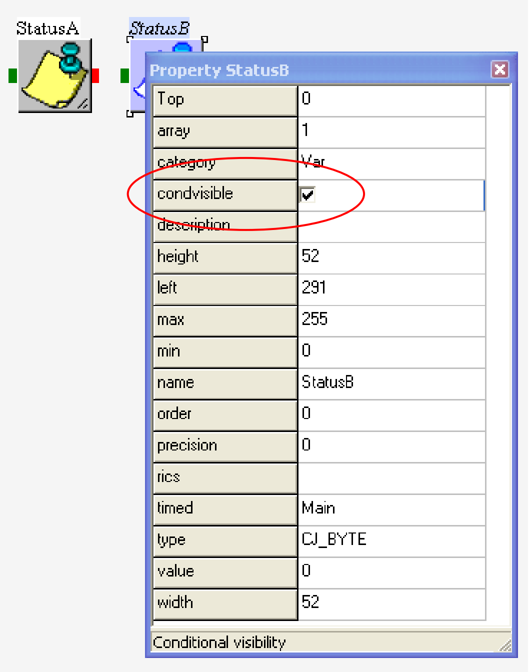
Step 1: Select the condvisible property of both entities.



Property StatusA

Top	0
array	1
category	Var
condvisible	☑
description	
height	52
left	213
max	255
min	0
name	StatusA
order	0
precision	0
rics	
timed	Main
type	CJ_BYTE
value	0
width	52

Conditional visibility

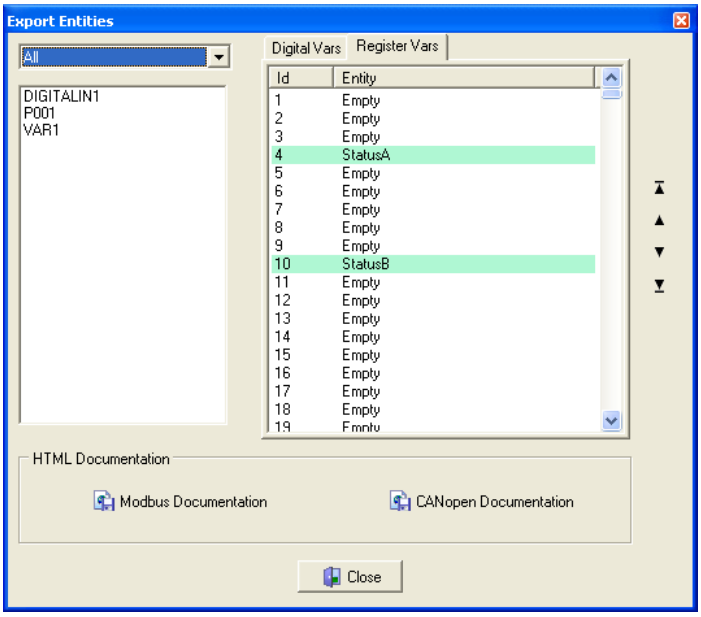


Property StatusB

Top	0
array	1
category	Var
condvisible	☑
description	
height	52
left	291
max	255
min	0
name	StatusB
order	0
precision	0
rics	
timed	Main
type	CJ_BYTE
value	0
width	52

Conditional visibility

Step 2: Export the 2 variables on Export table:



DIGITALIN1
P001
VAR1

Id	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
15	Empty
16	Empty
17	Empty
18	Empty
19	Empty

With **Export Entities** tool, export the variable StatusA at the address 4, and the variable StatusB at the address 10.

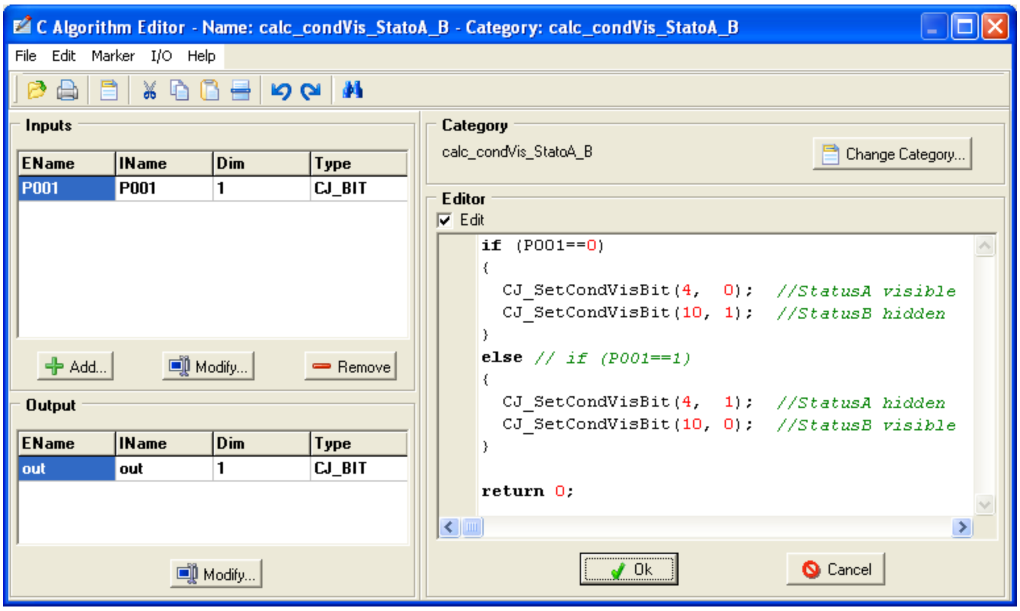
Step 3: Write the algorithm to rule the display or hiding of the 2 variables.

In the example:

If P001=0: the StatusA is visible (with ON value), and the StatusB is hided.

If P001=1: the StatusA is hide, and the StatusB is visible (with ON value).

The algorithm code:



Inputs

EName	IName	Dim	Type
P001	P001	1	CJ_BIT

Output

EName	IName	Dim	Type
out	out	1	CJ_BIT

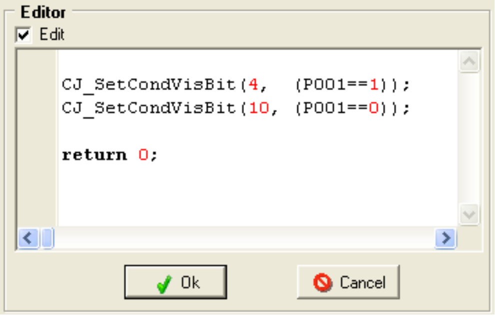
Category: calc_condVis_StatoA_B

☑ 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 can be achieved including the condition result in the value parameter:



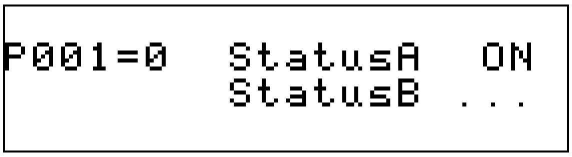
☑ Edit

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

return 0;
```

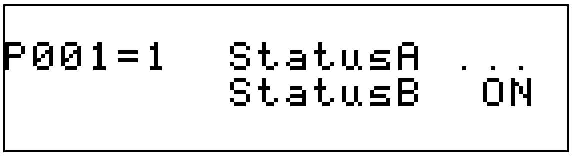
The user interface effect on 4 x 20 alphanumeric or on 240 x 128 graphic is the following:

With P001=0:



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

With P001=1:



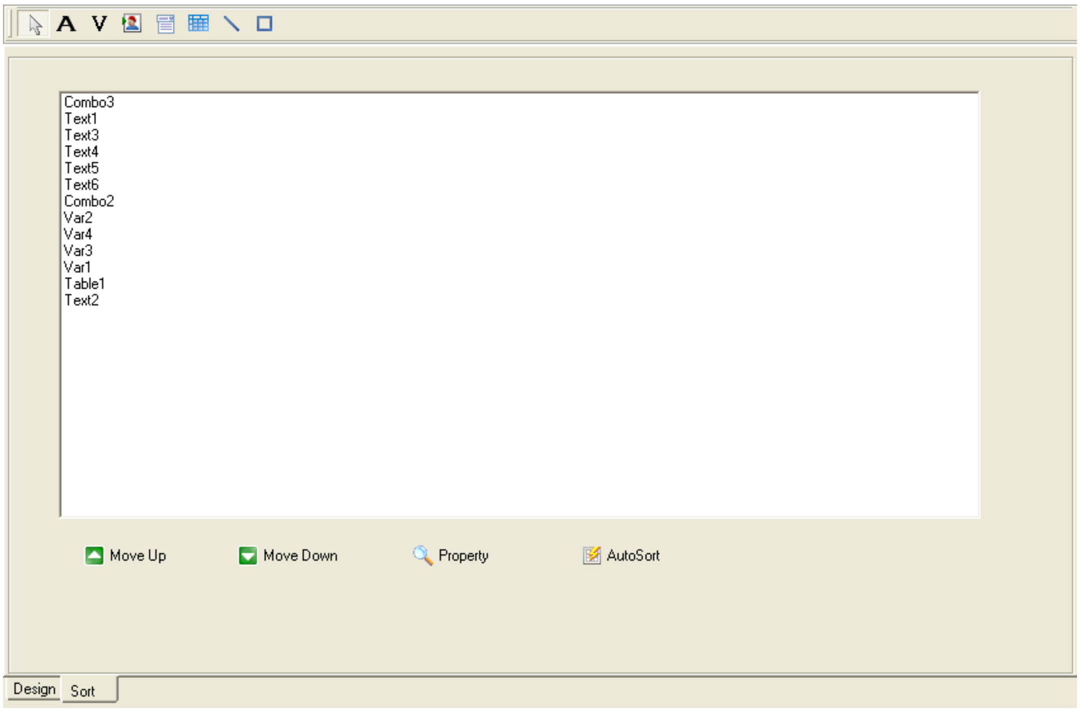
```
P001=1   StatusA ...
         StatusB  ON
```

5.6.5 Sorting Embedded Interface Markup Language (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 are entered by the developer when the page was created.

When the page is loaded by the user interface panel, the elements in the page appear in the same order as they were entered.

To customize the sorting order of the various graphic elements, and therefore the order in which they appear in the page, On the EIML definition page, click the **Sort** sheet 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 are 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 appear. Otherwise, you can use the **Move Up** and **Move Down** buttons to move the element up or down the list.

The AutoSort function sorts the elements automatically depending on their position in the page.

They are sorted starting from the element positioned on the top left side of the page and finishing with the element on the bottom right side.

5.6.6 EIML Pages Import and Export

The EIML pages of the current projects can be exported and reused in other projects. Each EIML page is exported individually.

1. Select the EIM**L** page to be exported.
2. Select **File→Export→Export EIML Page** to export the selected **EIML** page.
3. The **EIML** page is saved with .eiml extension.
4. To import **EIML** pages in a project, select **File→Import→Import EIML Page**.
5. The **EIML** page is added to the project inspector Window.

5.7 Commissioning on Expansion Bus (ExpBUS)

ExpBUS can be used for:

- Adding expansions to a controller (IO Expansions).
- Connecting a remote terminal (**TM168GDB**, **TM168GDTS**).
- Making communication between controllers.

5.7.1 ExpBUS Node and Physical Address

ExpBUS is configured in the **Hardware Expert**.

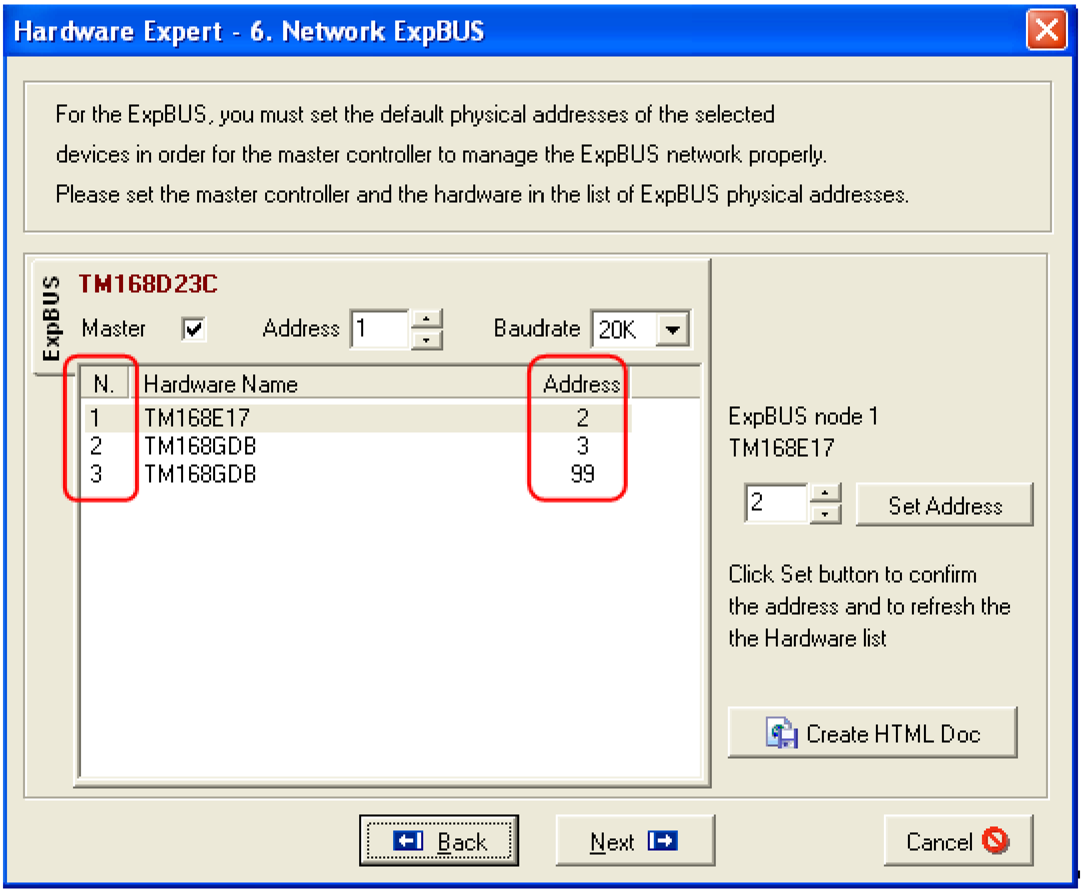
The ExpBUS Expansions are selected in the **Hardware Expert – 4. Expansions** window.

The ExpBUS Displays are selected in the **Hardware Expert – 5. Displays** window.

Addressing of the ExpBUS uses a logical node number (unique inside a project) and a physical address (unique in the entire network):

- The logical node number is assigned by the **Hardware Expert**.
- Logical node numbers are assigned starting with 1 and increasing for each additional device added.
- The controller is always node 0.
- A default physical address is assigned by the **Hardware Expert**, which can be changed by the programmer.

The figure shows the **Hardware Expert – 6. Network ExpBUS**.



Hardware Expert - 6. Network ExpBUS

For the ExpBUS, you must set the default physical addresses of the selected devices in order for the master controller to manage the ExpBUS network properly.
Please set the master controller and the hardware in the list of ExpBUS physical addresses.

ExpBUS

TM168D23C

Master ☑ Address 1 Baudrate 20K

N.	Hardware Name	Address
1	TM168E17	2
2	TM168GDB	3
3	TM168GDB	99

ExpBUS node 1
TM168E17

2 Set Address

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

The devices are sorted by their address or node number (left column) with the physical address shown in the right column.

5.7.2 Factory Configuration

Every equipment is delivered with a default physical address and default baud rate.

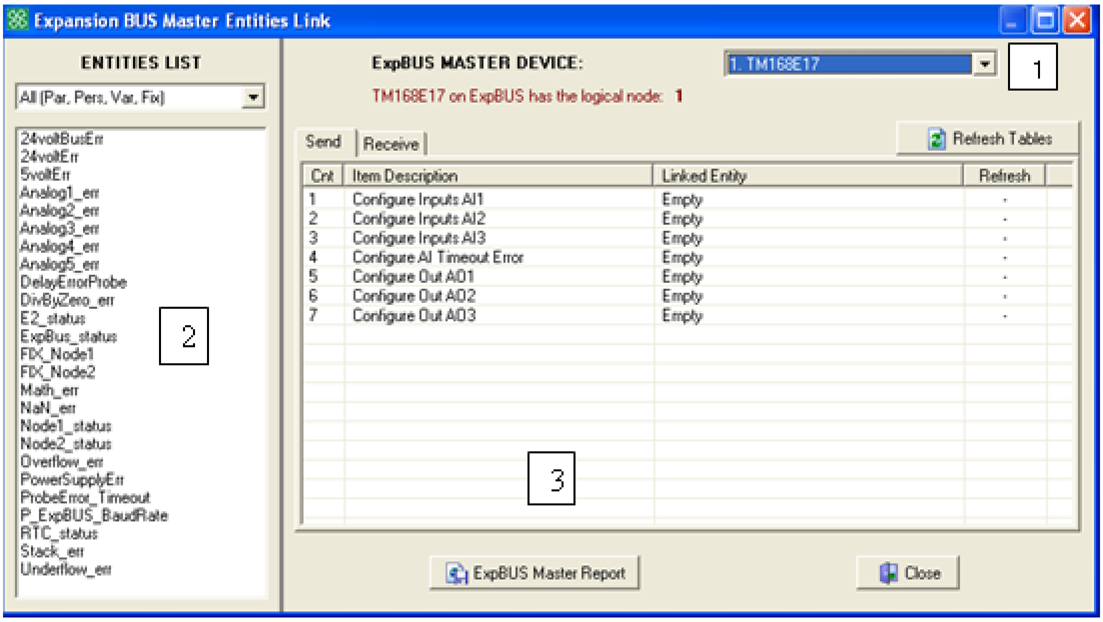
- **TM168●23●●** :
  - Physical address: 1
  - Baud rate: 20 K
- **TM168E17**:
  - Physical address: 2
  - Baud rate: 20 K
- **TM168GD●●**:
  - Physical Address: 99
  - Baud rate: AUTO

5.7.3 ExpBUS Master…Window

This menu is accessible through **Tool→ExpansionBUS Master….**

It allows to link the program entities with the variables of the ExpBUS declared equipments.

The figure shows the **ExpBUS Master…** window:



TM168E17 on ExpBUS has the logical node: 1

24voltBusErr
24voltErr
5voltErr
Analog1_err
Analog2_err
Analog3_err
Analog4_err
Analog5_err
DelayErrorProbe
DivByZero_err
E2_status
ExpBus_status
FIX_Node1
FIX_Node2
Math_err
NaN_err
Node1_status
Node2_status
Overflow_err
PowerSupplyErr
ProbeError_Timeout
P_ExpBUS_BaudRate
RTC_status
Stack_err
Underflow_err

Cnt	Item Description	Linked Entity	Refresh
1	Configure Inputs AI1	Empty	-
2	Configure Inputs AI2	Empty	-
3	Configure Inputs AI3	Empty	-
4	Configure AI Timeout Error	Empty	-
5	Configure Out AO1	Empty	-
6	Configure Out AO2	Empty	-
7	Configure Out AO3	Empty	-

This menu is divided as following:

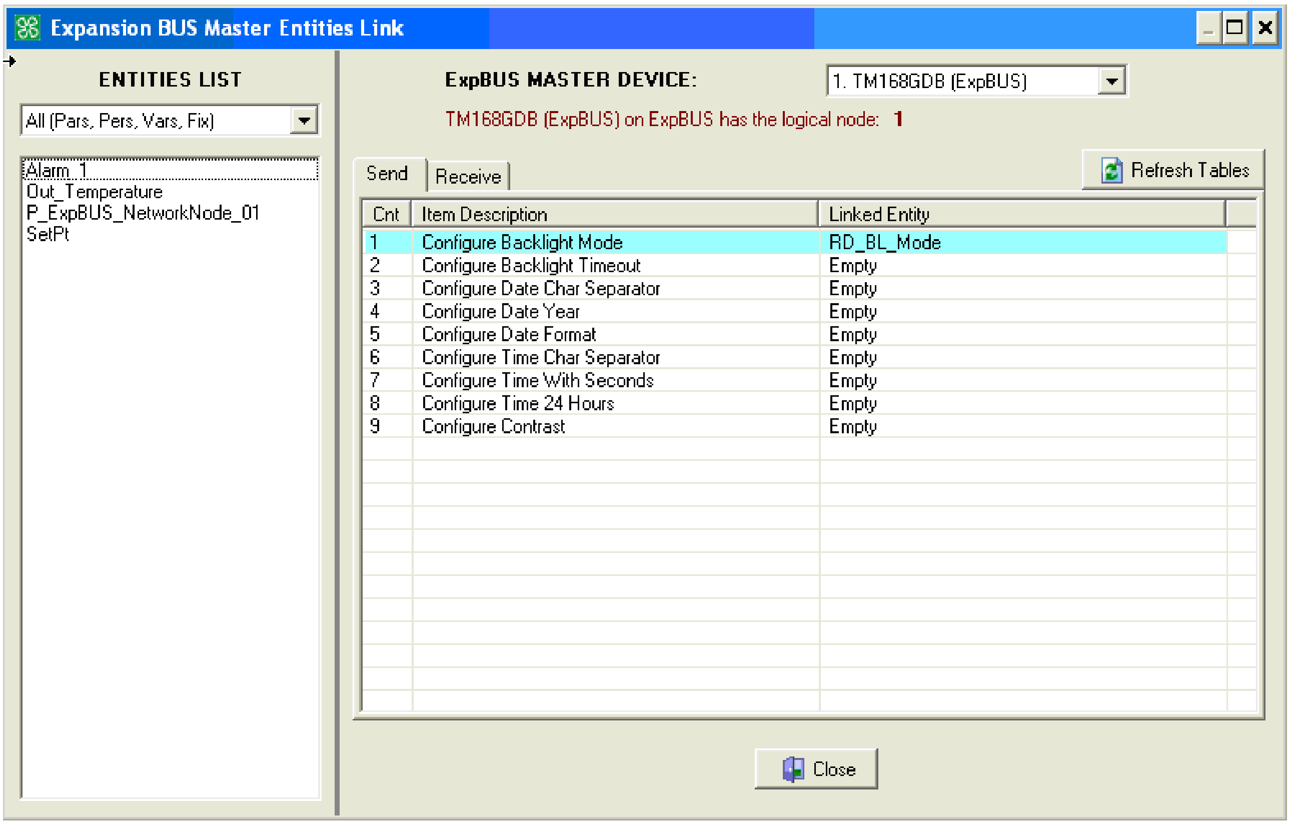
1. On top right, a combo box to select which device to configure. The logical node of the device is given.
2. On the left, the program entities.
3. On the right, the configuration variables of the entity.

They are divided in 2 windows:

- Send variable is the variable sent by the controller to the **ExpBUS** equipment.
- Receive variable is the variable read by the controller from the **ExpBUS** equipment.

Drag-and-drop entities in the configuration variables line to link them together. The ExpBUS equipment variable can be modified through the program.

Example:



TM168GDB (ExpBUS) on ExpBUS has the logical node: 1

Cnt	Item Description	Linked Entity
1	Configure Backlight Mode	RD_BL_Mode
2	Configure Backlight Timeout	Empty
3	Configure Date Char Separator	Empty
4	Configure Date Year	Empty
5	Configure Date Format	Empty
6	Configure Time Char Separator	Empty
7	Configure Time With Seconds	Empty
8	Configure Time 24 Hours	Empty
9	Configure Contrast	Empty

In this example, the **RD_BL_Mode** entity is linked to the TM168GDB sent parameter Configure Backlight Mode.

When **RD_BL_Mode** is changed in the program, the modification is sent to the TM168GDB and its Backlight mode is modified.

5.7.4 Creating Networks

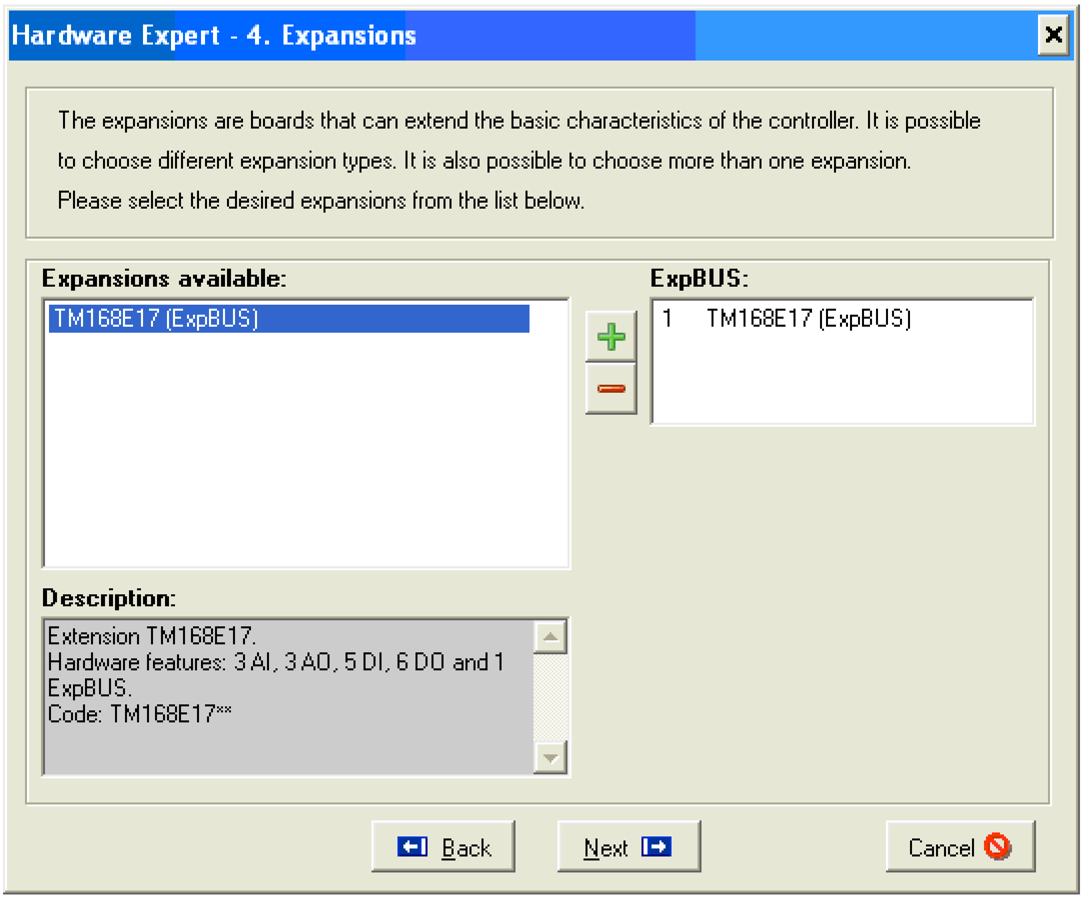
This section explains how to configure networks from the simplest to more complex.

5.7.5 Configuration of One Expansion and One Display

The configuration can be completely made through the **Hardware Expert**.

You need to select the expansion to be used. This will be done through the **Hardware Expert – 4. Expansions window**.

The figure shows the **Hardware Expert – 4. Expansions**.



The expansions are boards that can extend the basic characteristics of the controller. It is possible to choose different expansion types. It is also possible to choose more than one expansion.
Please select the desired expansions from the list below.

Expansions available:
TM168E17 (ExpBUS)

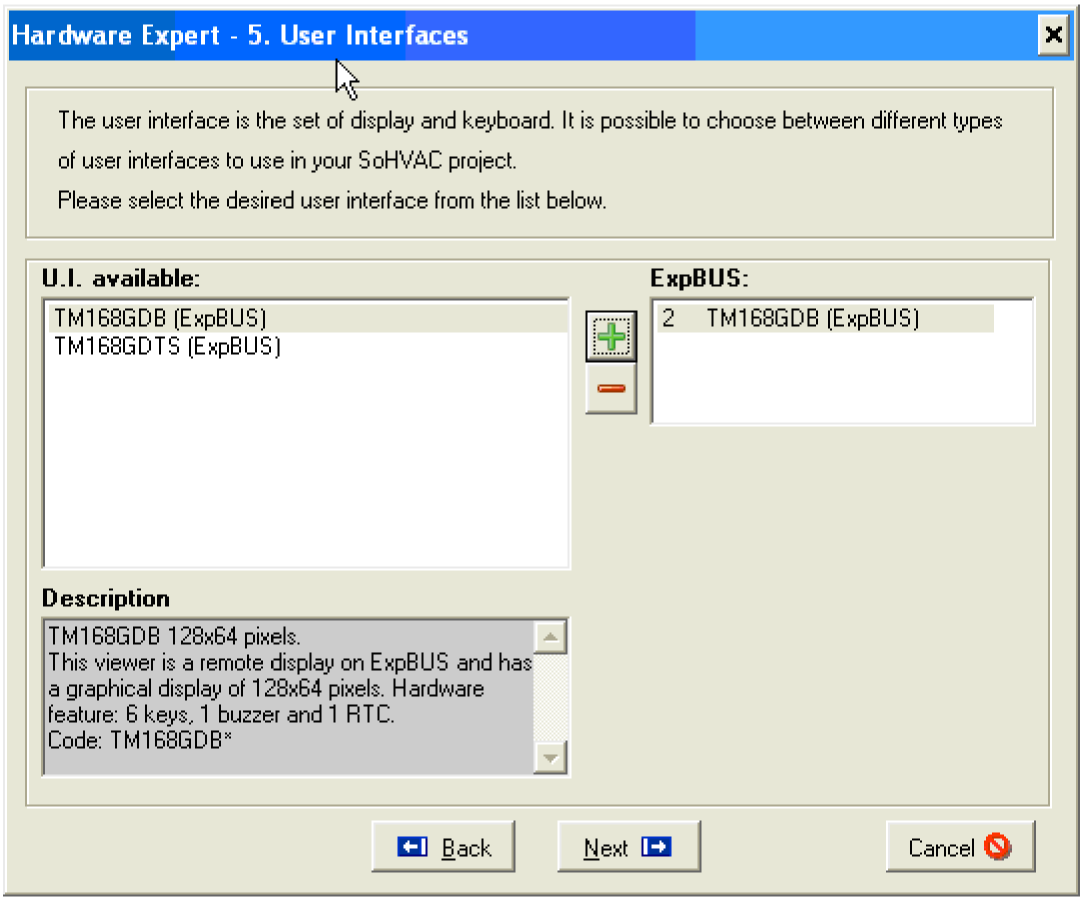
ExpBUS:
1 TM168E17 (ExpBUS)

Description:
Extension TM168E17.
Hardware features: 3 AI, 3 AO, 5 DI, 6 DO and 1 ExpBUS.
Code: TM168E17**

In this example, add one TM168E17 expansion to the project.

Then you need to select the display to be used. This will be done through the **Hardware Expert – 5. User Interfaces** window.

The figure shows the **Hardware Expert – 5. User Interfaces**.



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 SoHVAC project.

Please select the desired user interface from the list below.

U.I. available:

- TM168GDB (ExpBUS)
- TM168GDTS (ExpBUS)

ExpBUS:

2 TM168GDB (ExpBUS)

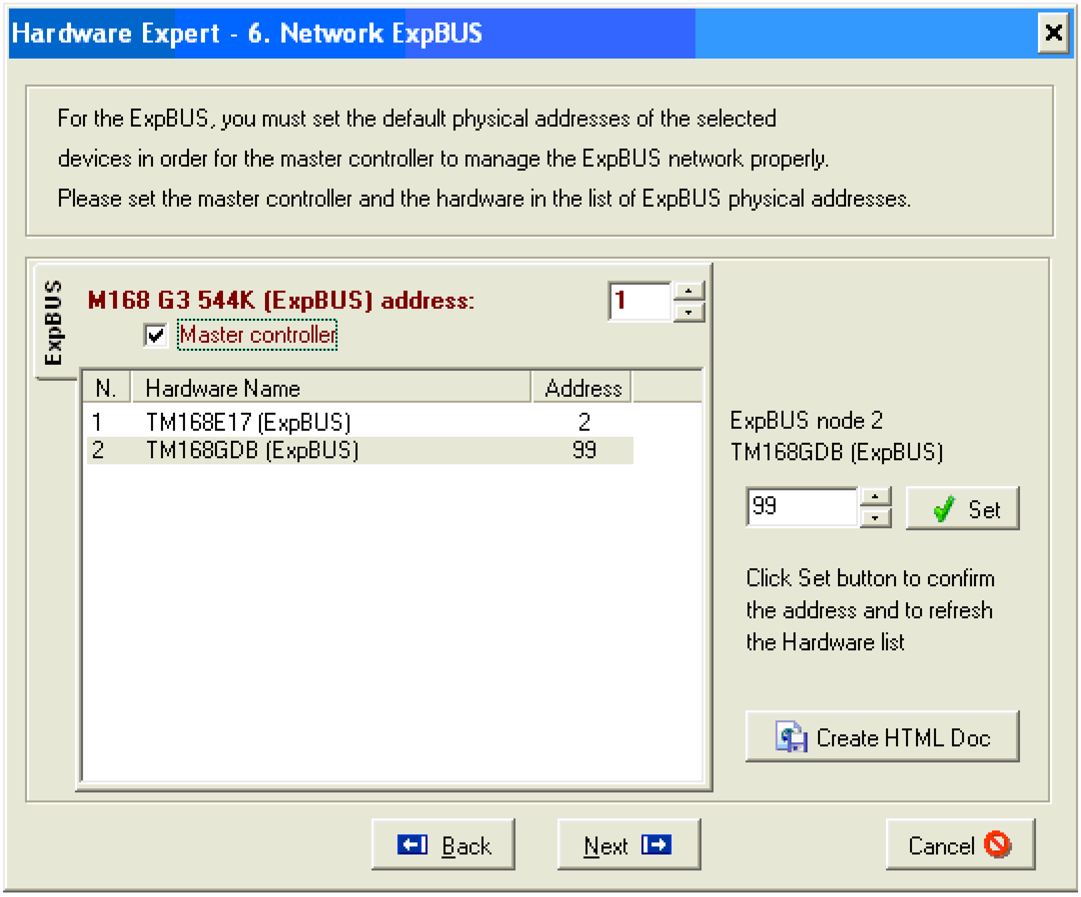
Description

TM168GDB 128x64 pixels.
This viewer is a remote display on ExpBUS and has a graphical display of 128x64 pixels. Hardware feature: 6 keys, 1 buzzer and 1 RTC.
Code: TM168GDB*

In this example, add one TM168GDB display to the project.

In the next window **Hardware Expert – 6. User Interfaces**, you need to:

- Configure the address to the equipment default address.
  - **TM168E17**: Address 2
  - **TM168GDB**: Address 99
- Make the controller a Master by selecting the **Master controller** check box



Hardware Expert - 6. Network ExpBUS

For the ExpBUS, you must set the default physical addresses of the selected devices in order for the master controller to manage the ExpBUS network properly.
Please set the master controller and the hardware in the list of ExpBUS physical addresses.

M168 G3 544K (ExpBUS) address: 1

☑ Master controller

N.	Hardware Name	Address
1	TM168E17 (ExpBUS)	2
2	TM168GDB (ExpBUS)	99

ExpBUS node 2
TM168GDB (ExpBUS)

99

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

The network is operational.

5.7.6 Configuring Multiple Expansions

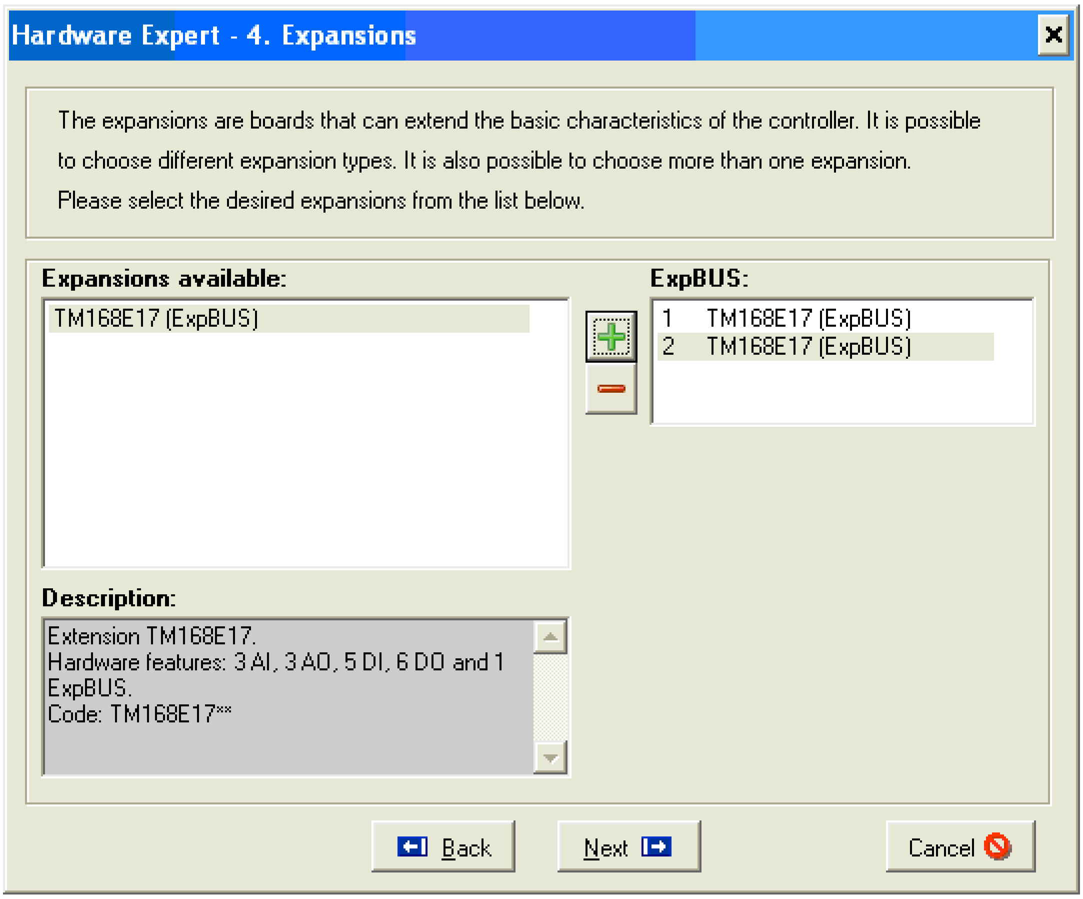
Adding expansions will increase the number of Inputs/Outputs of the controller. The controller will be master in the communication with the expansions.

The first steps are similar to section 5.7.4.1. Then the expansion address must be configured.

The following example explains how to configure a network with 2 TM168E17 expansions.

1. Expansions declaration

You need to select the expansions to be used. This will be done through the **Hardware Expert – 4. Expansions** window.



Hardware Expert - 4. Expansions

The expansions are boards that can extend the basic characteristics of the controller. It is possible to choose different expansion types. It is also possible to choose more than one expansion. Please select the desired expansions from the list below.

Expansions available:

TM168E17 (ExpBUS)

ExpBUS:

1 TM168E17 (ExpBUS)
2 TM168E17 (ExpBUS)

Description:

Extension TM168E17.
Hardware features: 3 AI, 3 AO, 5 DI, 6 DO and 1 ExpBUS.
Code: TM168E17**

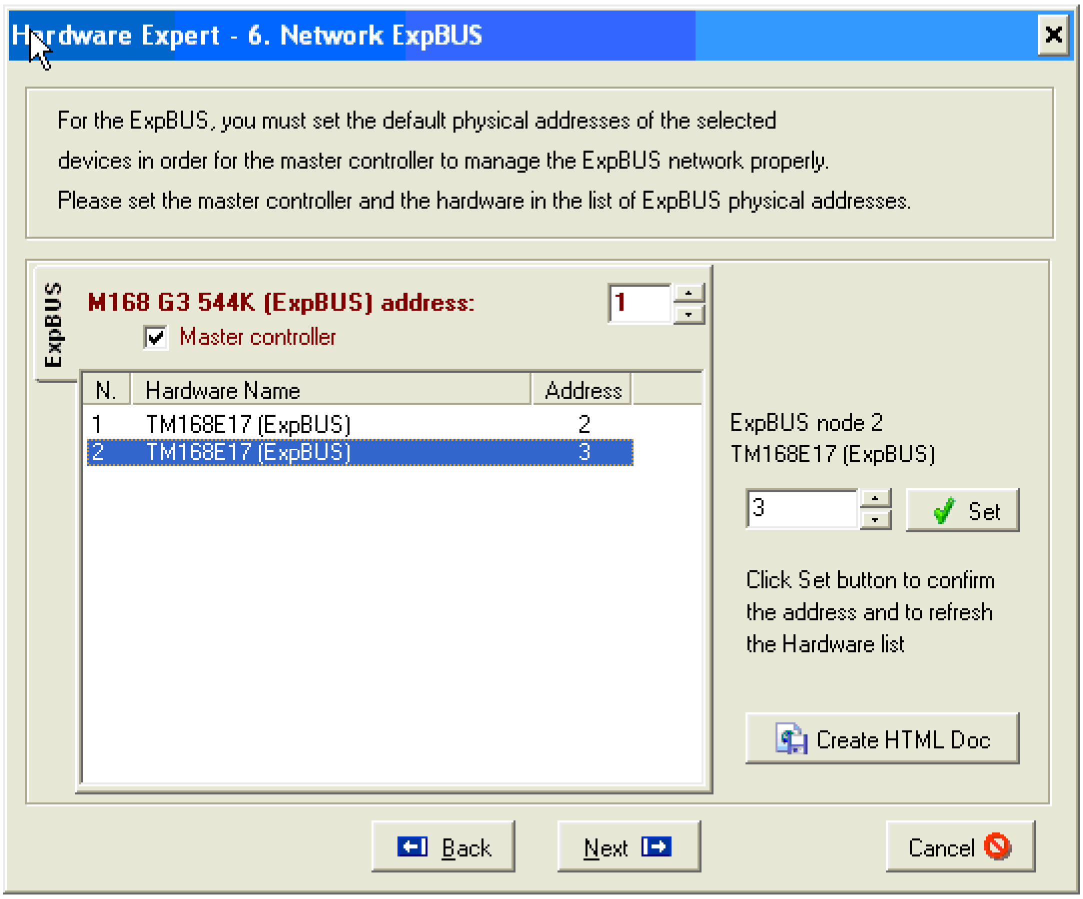
In this example, add 2 TM168E17 expansions to the project.

2. Address configuration and Master selection

In the **Hardware Expert – 6. Network ExpBUS**, you must:

- Select the Master controller check box to configure the controller as a master. The controller handles the expansion communication and therefore must be a master on the network.
- Set the 2 expansions address as required with the right box and **Set** Button.

By default the first expansion has the address 2 and the following expansion 2 has address 3….



Hardware Expert - 6. Network ExpBUS

For the ExpBUS, you must set the default physical addresses of the selected devices in order for the master controller to manage the ExpBUS network properly.
Please set the master controller and the hardware in the list of ExpBUS physical addresses.

ExpBUS

M168 G3 544K (ExpBUS) address: 1

☑ Master controller

N.	Hardware Name	Address
1	TM168E17 (ExpBUS)	2
2	TM168E17 (ExpBUS)	3

ExpBUS node 2
TM168E17 (ExpBUS)

3 Set

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

In this example, the two Expansions address are 2 and 3 and the Controller is Master with address 1.

3. Changing address of an expansion

By default the expansion is delivered with address 2. The third activity is to configure one of the expansions with the address 3.

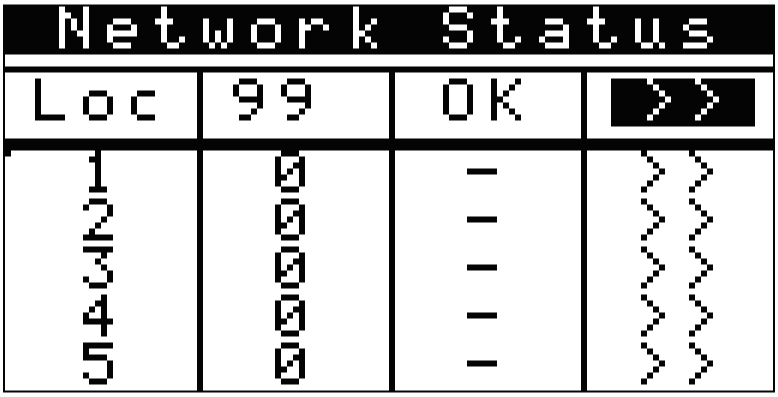
To perform this, you need:

- The expansion **TM168E17** to reconfigure,
- One remote display (**TM168GDB** or **TM168GDTS**), 24 Vdc / Vac to power the equipments, and
- Cabling between the two **ExpBUS** connectors.

Once both the equipments are powered up and connected on the ExpBUS, make the following actions on the display:

Press the **LEFT+ENTER** buttons for 3 seconds.

The following Network status page appears:



Network Status

Loc	99	OK	>>
1	0	-	>>
2	0	-	>>
3	0	-	>>
4	0	-	>>
5	0	-	>>

First column corresponds to the location Node on the network.

Second column corresponds to the physical address.

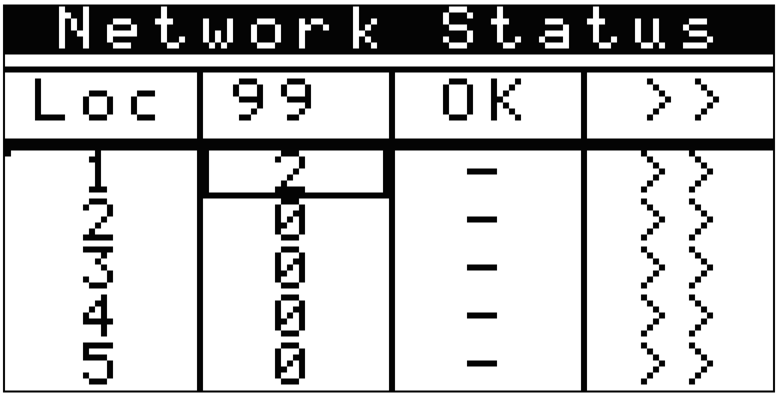
Third column gives communication status between the equipment and the display: OK means connection OK, -- means connection not established.

Fourth column is a double arrow programmed to enter the configuration pages of each element.

To connect the TM168E17, move the arrows all the way down the fourth column until the second column is activated.

Move the active element, to the line corresponding to Loc 1.

Press ENTER and change the address value to 2. Press ENTER.

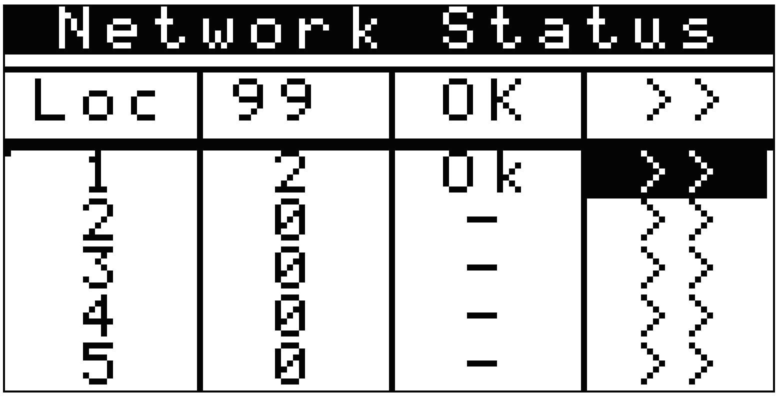


Network Status

Loc	99	OK	>>
1	2	-	>>
2	0	-	>>
3	0	-	>>
4	0	-	>>
5	0	-	>>

When the equipment is detected, the third line shows Ok.

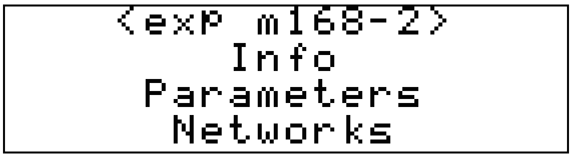
Move activation item to the >> in the loc 1 line:



Network Status

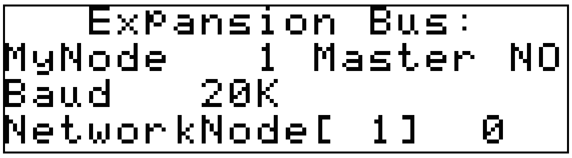
Loc	99	OK	>>
1	2	Ok	>>
2	0	-	>>
3	0	-	>>
4	0	-	>>
5	0	-	>>

Press ENTER. You can access to the Expansion 17 configuration pages.



```
  <exp m168-2>
     Info
  Parameters
   Networks
```

Navigate in the menu: Networks …ExpBUS… to display the **Expansion Bus** menu:



```
   Expansion Bus:
MyNode    1 Master  NO
Baud    20K
NetworkNode[ 1]   0
```

Select the MyNode value and change it to the correct address: 3.

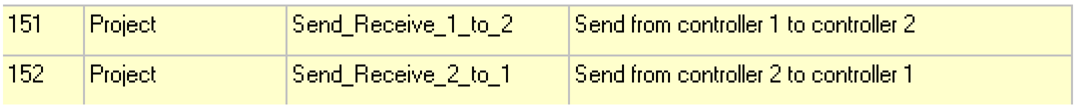
This expansion is configured at physical address 3 of ExpBUS network.

5.7.7 Configuring Data Exchanges between Controllers

The goal is to connect exchange information between 2 controllers, on the same network.

To handle communication between 2 controllers, we need to declare 2 Commands in the project.

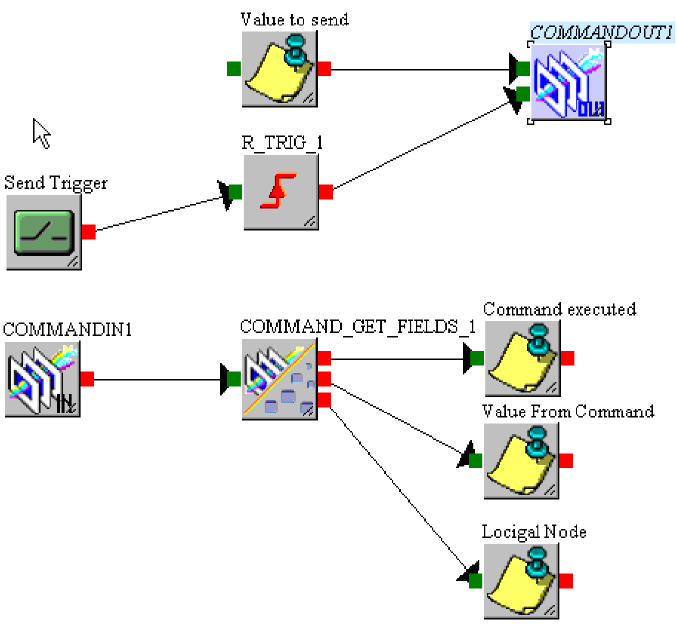
See Commands chapter and create the 2 following commands in the project area:

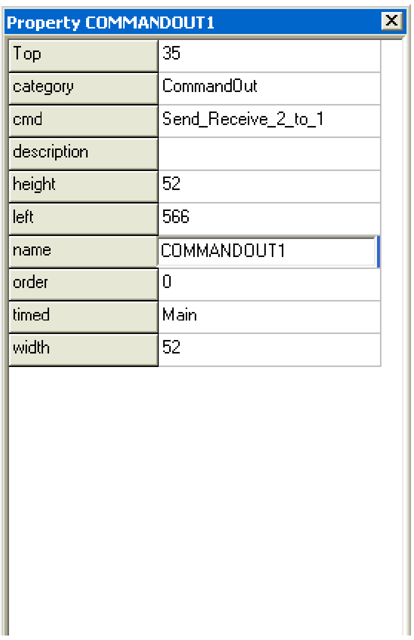


151	Project	Send_Receive_1_to_2	Send from controller 1 to controller 2
152	Project	Send_Receive_2_to_1	Send from controller 2 to controller 1

Those commands must be used with the **Command In** and **Command Out** entities.

For more information about command entities, refer to *Commands*.





Property COMMANDOUT1

Top	35
category	CommandOut
cmd	Send_Receive_2_to_1
description	
height	52
left	566
name	COMMANDOUT1
order	0
timed	Main
width	52

In this example, in the controller 2:

- The **CommandOut** is linked to the declared cmd Send_Receive_2_to_1.
- The **CommandIn** is linked to the declared cmd Send_Receive_1_to_2.

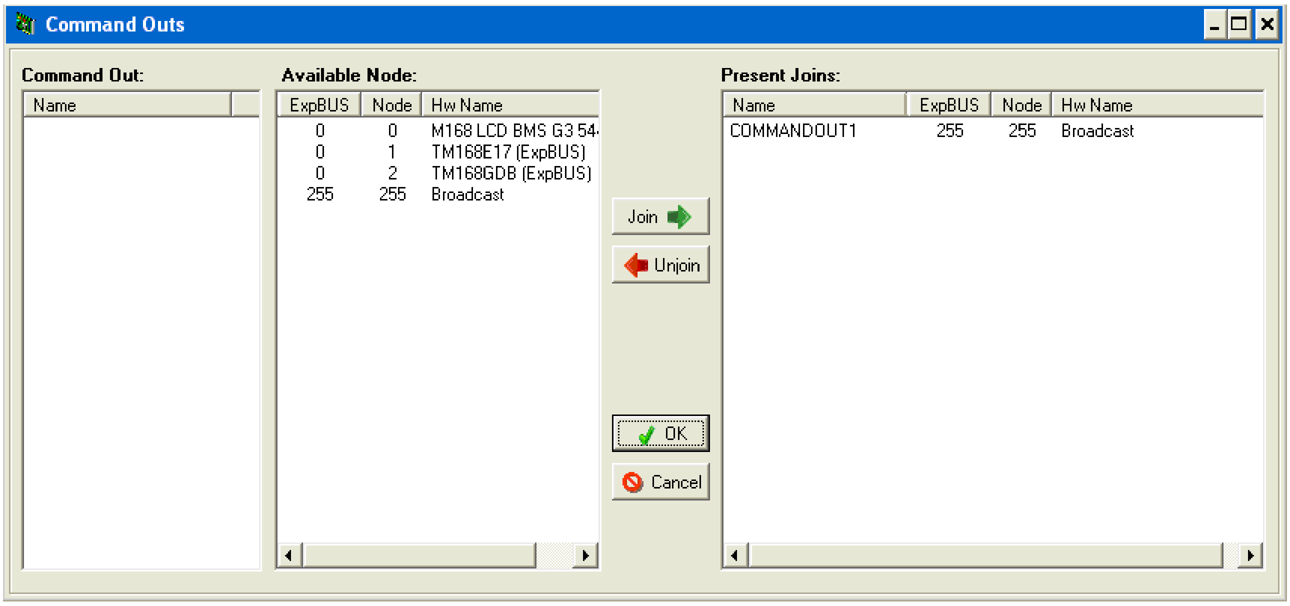
In the controller 1, you must have:

- The **CommandOut** linked to the declared cmd Send_Receive_1_to_2.
- The **CommandIn** linked to the declared cmd Send_Receive_2_to_1.

After the 2 created commands must be directed on the network:

This is possible through **Project→Join Tools→ Commands In** and **Commands Out**.

Project→Join Tools→Commands Out.



Command Outs

Command Out:

Name

Available Node:

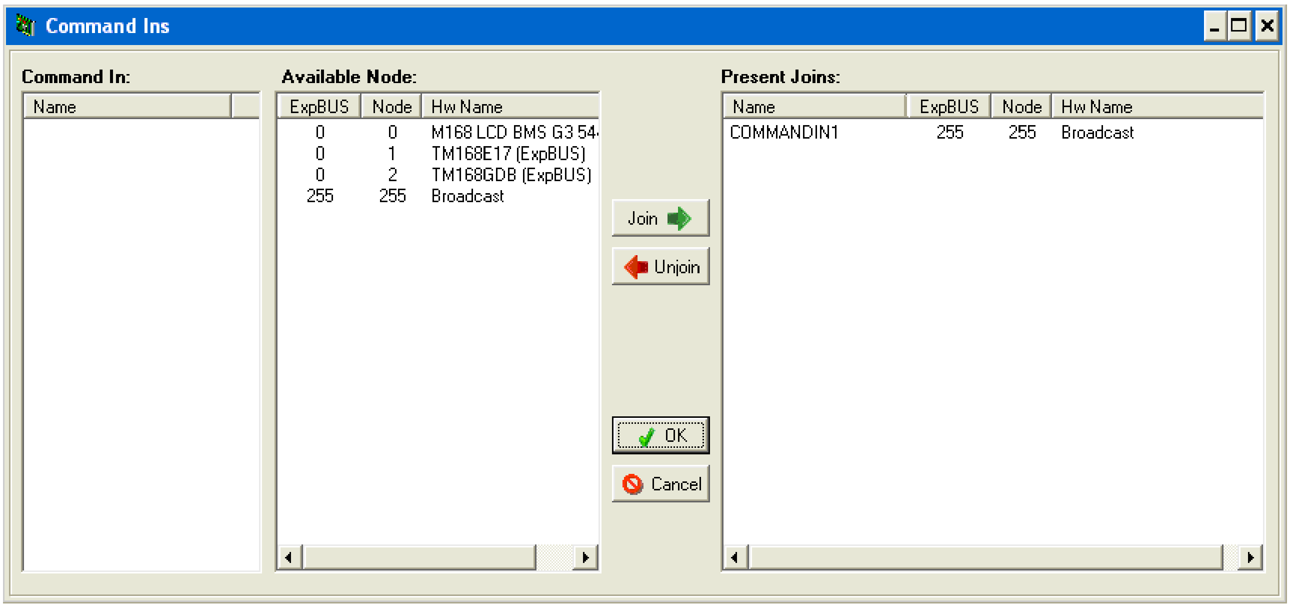
ExpBUS	Node	Hw Name
0	0	M168 LCD BMS G3 54
0	1	TM168E17 (ExpBUS)
0	2	TM168GDB (ExpBUS)
255	255	Broadcast

Present Joins:

Name	ExpBUS	Node	Hw Name
COMMANDOUT1	255	255	Broadcast

The message is broadcasted in the network.

Project→Join Tools→Commands In.



Command Ins

Command In:

Name

Available Node:

ExpBUS	Node	Hw Name
0	0	M168 LCD BMS G3 54
0	1	TM168E17 (ExpBUS)
0	2	TM168GDB (ExpBUS)
255	255	Broadcast

Present Joins:

Name	ExpBUS	Node	Hw Name
COMMANDIN1	255	255	Broadcast

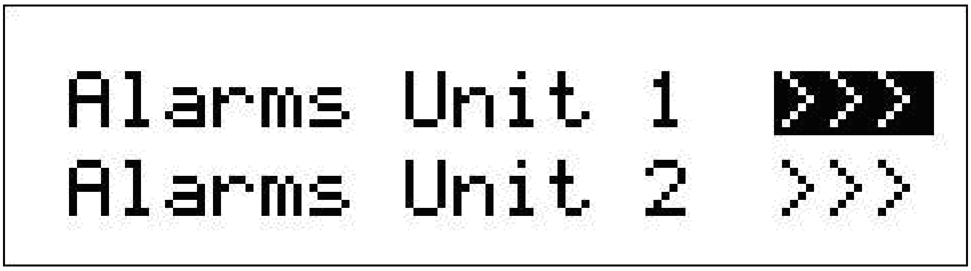
The message can be received from any slave in the network.

5.7.8 Configuring One Remote Display with Multiple Controllers

The remote display can only communicate to one controller at a time. By default, the display attempts to load the start up page (ID1) of Node 1. If there is no controller configured at Node 1, enter the network status page to select a different node where a controller is present. After the communication is established with either the default node 1 or with another node that you have manually selected, the start up page of the controller will load into the display.

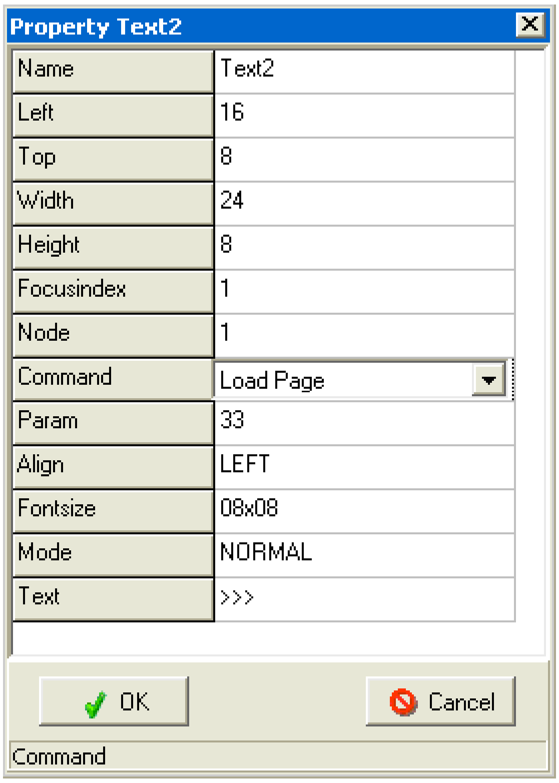
To navigate to other controllers, the easiest solution is to use the **LoadPage** function in each of the controllers with which you wish to communicate.

For example, the >>> can be used to access specific controller pages in other controllers than the one in with which you are currently communicating.



Alarms Unit 1 >>>
Alarms Unit 2 >>>

To load Page 33 from Node 1, for example, the property of the >>> text must be configured as follows:



Property Text2

Name	Text2
Left	16
Top	8
Width	24
Height	8
Focusindex	1
Node	1
Command	Load Page
Param	33
Align	LEFT
Fontsize	08x08
Mode	NORMAL
Text	>>>

The following property must be configured:

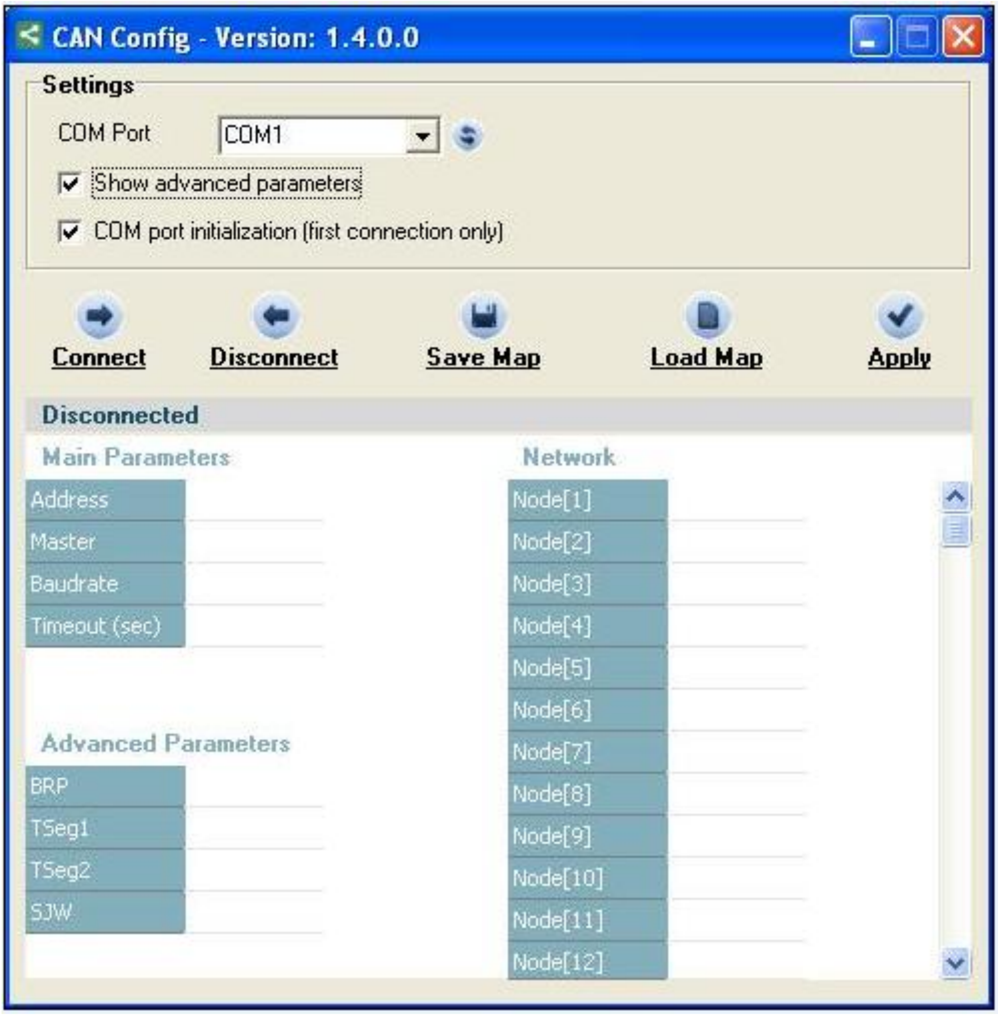
- **Command: LoadPage**
- **Node: Node** (from Display configuration) where to request the page
- **Param: ID** of the page

NOTE: The **Focusindex** property must be different from 0 to be able to click on the text.

5.7.9 Expansion Bus (ExpBus) Configuration

To configure the ExpBus you must use the **CAN Config** tool in SoHVAC Software. The ExpBus is based on the CAN network. The **CAN Config** tool allows you to configure the parameters for the devices that you wish to connect to the ExpBus.

Choosing menu tools/**CAN Config**, the following window is displayed:



CAN Config - Version: 1.4.0.0

Settings

COM Port: COM1

☑ Show advanced parameters

☑ COM port initialization (first connection only)

Disconnected

- Values are in grey if the communication is not active.
- Values are in red in case of communication error.
- Values underlined are in red where the written values are different from the read ones.

The configuration parameters are written to the controller once you click on **Apply**.

Settings:

- **COM Port**: Serial port of the PC that is connected to the controller.
- **Show advanced parameters**: When selected, bit timing parameters are shown and can be set (they are shown in the window as **Advanced Parameters**).
- **COM port initialization**: The interface integrated in the programming cable must be initialized when used for the first time. To accomplish the first initialization, if the interface has been already initialized, do not check the check box.

Prompts:

- **Connect**: It starts the communication with the controller. Since that moment it is possible to show and modify the CANbus communication parameters.
- **Disconnect**: It stops the communication with the controller.

- **Save Map**: It saves the values there are in the windows in a file (map; Main Parameters, Network, Advanced Parameters.
- **Load Map**: It reads a map file and writes the loaded values in the controller.
- **Apply**: It sends the controller the prompt to make valid the parameters that have been modified.

To modify the parameters when the connection is active, it is enough to write the values in the window. The program refreshes the values every second and provides an immediate feedback (on condition the writing has successfully been completed). To make valid the values just written and in order that the controller starts again the CAN communication using the new parameters, it is necessary to press prompt **APPLY**. If after the validation of the parameters some of these are overwritten by the application (for example using the libraries belonging to the group System/CAN 1st) the incongruity is underlined in red.

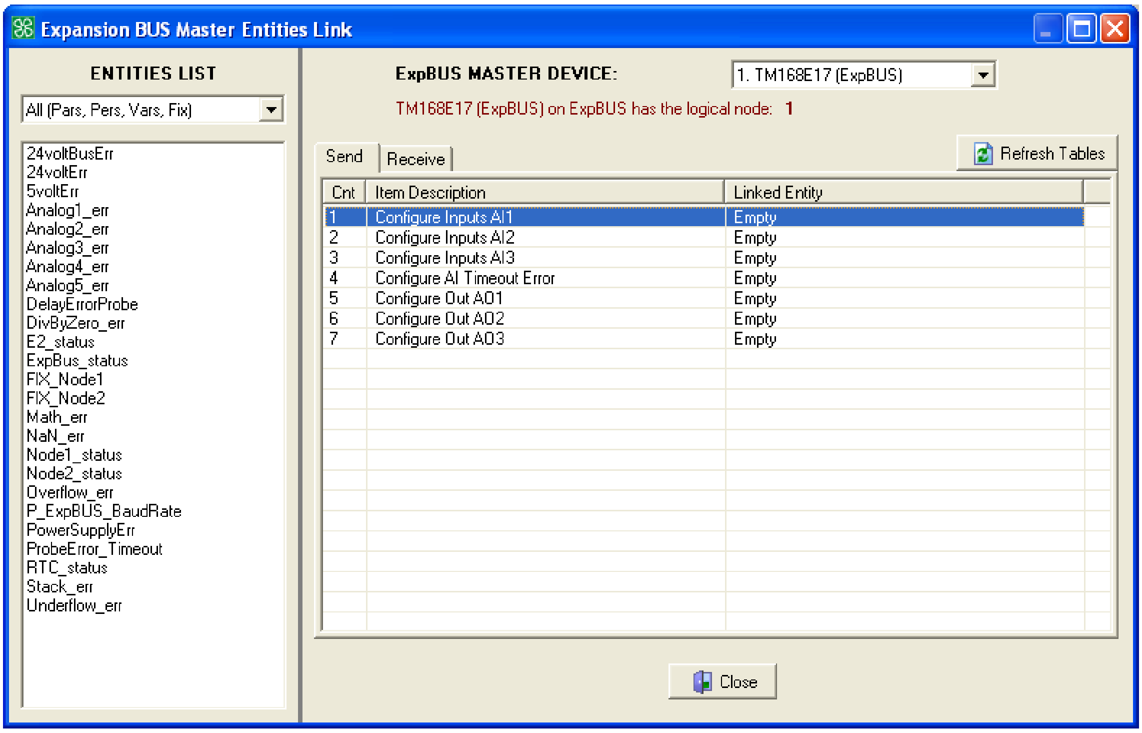
To save and load maps it is necessary the communication is active; no modification is allowed when offline.

5.7.10 ExpBUS Master

The expansion bus master function allows the M168 controller to send configuration data to an expansion bus device or to receive data from an expansion bus device.

Expansion bus modules must be defined in the **Hardware Expert Wizard** and the M168 controller must be enabled as Master in the **Hardware Expert -6- Network ExpBUS**.

To link entities to an expansion bus module, select the menu: **Tools→ExpansionBUS Master…**



TM168E17 (ExpBUS) on ExpBUS has the logical node: 1

24voltBusErr
24voltErr
5voltErr
Analog1_err
Analog2_err
Analog3_err
Analog4_err
Analog5_err
DelayErrorProbe
DivByZero_err
E2_status
ExpBus_status
FIX_Node1
FIX_Node2
Math_err
NaN_err
Node1_status
Node2_status
Overflow_err
P_ExpBUS_BaudRate
PowerSupplyErr
ProbeError_Timeout
RTC_status
Stack_err
Underflow_err

Cnt	Item Description	Linked Entity
1	Configure Inputs AI1	Empty
2	Configure Inputs AI2	Empty
3	Configure Inputs AI3	Empty
4	Configure AI Timeout Error	Empty
5	Configure Out AO1	Empty
6	Configure Out AO2	Empty
7	Configure Out AO3	Empty

The left side of the window shows all the project variables, while the right side of the window shows expansion bus master device with the registers. The registers are listed in 2 sheets: Send and Receive.

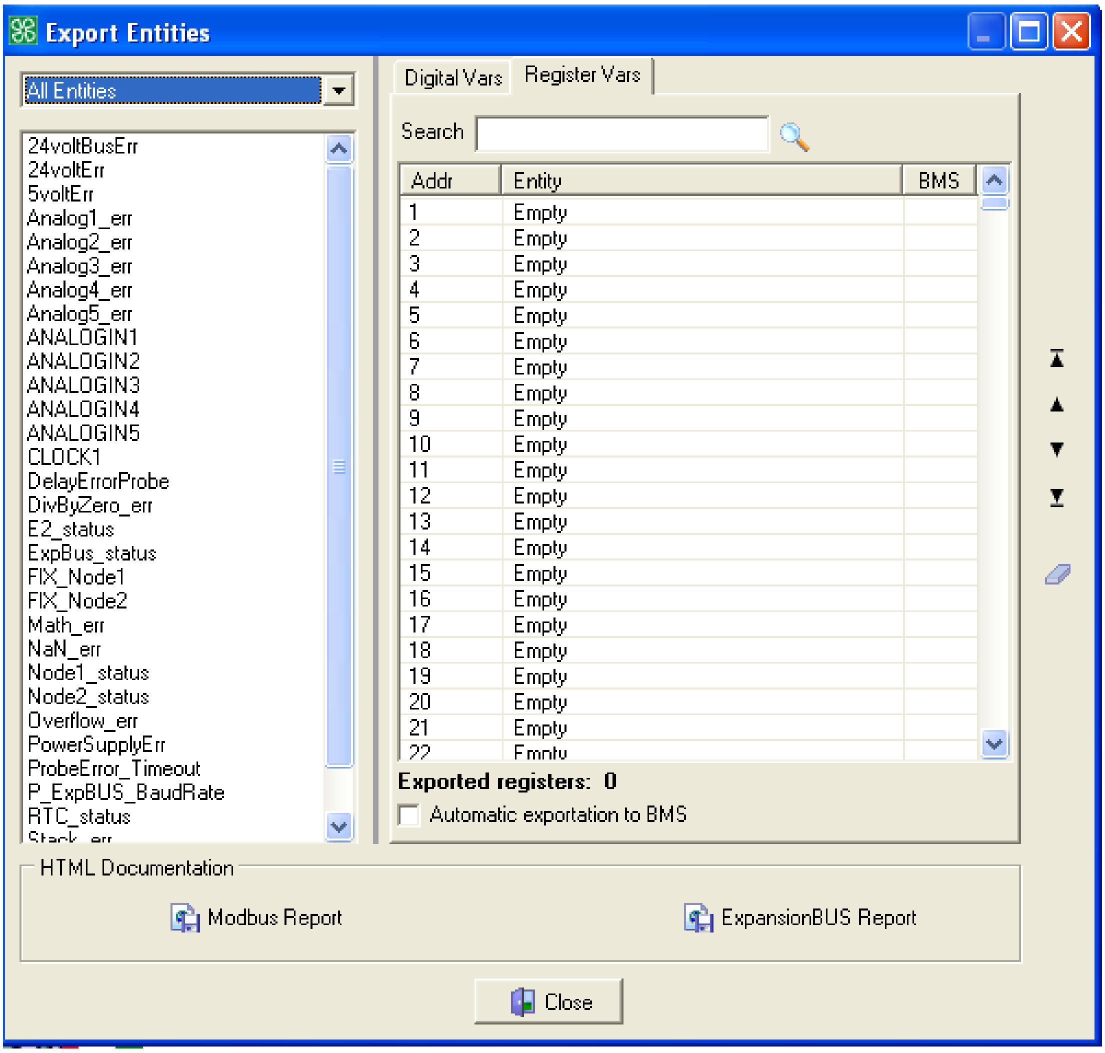
To connect a parameter or project variable to a parameter or variable on the network, you need to select it in the left list and drag and drop it to the desired position in the right list.

5.8 Modbus Communication

5.8.1 Exporting Variables

To access internal variables of the controller through one of the available communication protocols, you need to define which variables you want to export (that is, making visible outside) and how to sort them.

To open the **Export Entities** window, use the menu **Tools→Export Entities…**



24voltBusErr
24voltErr
5voltErr
Analog1_err
Analog2_err
Analog3_err
Analog4_err
Analog5_err
ANALOGIN1
ANALOGIN2
ANALOGIN3
ANALOGIN4
ANALOGIN5
CLOCK1
DelayErrorProbe
DivByZero_err
E2_status
ExpBus_status
FIX_Node1
FIX_Node2
Math_err
NaN_err
Node1_status
Node2_status
Overflow_err
PowerSupplyErr
ProbeError_Timeout
P_ExpBUS_BaudRate
RTC_status
Stack_err

Addr	Entity	BMS
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	
22	Empty	

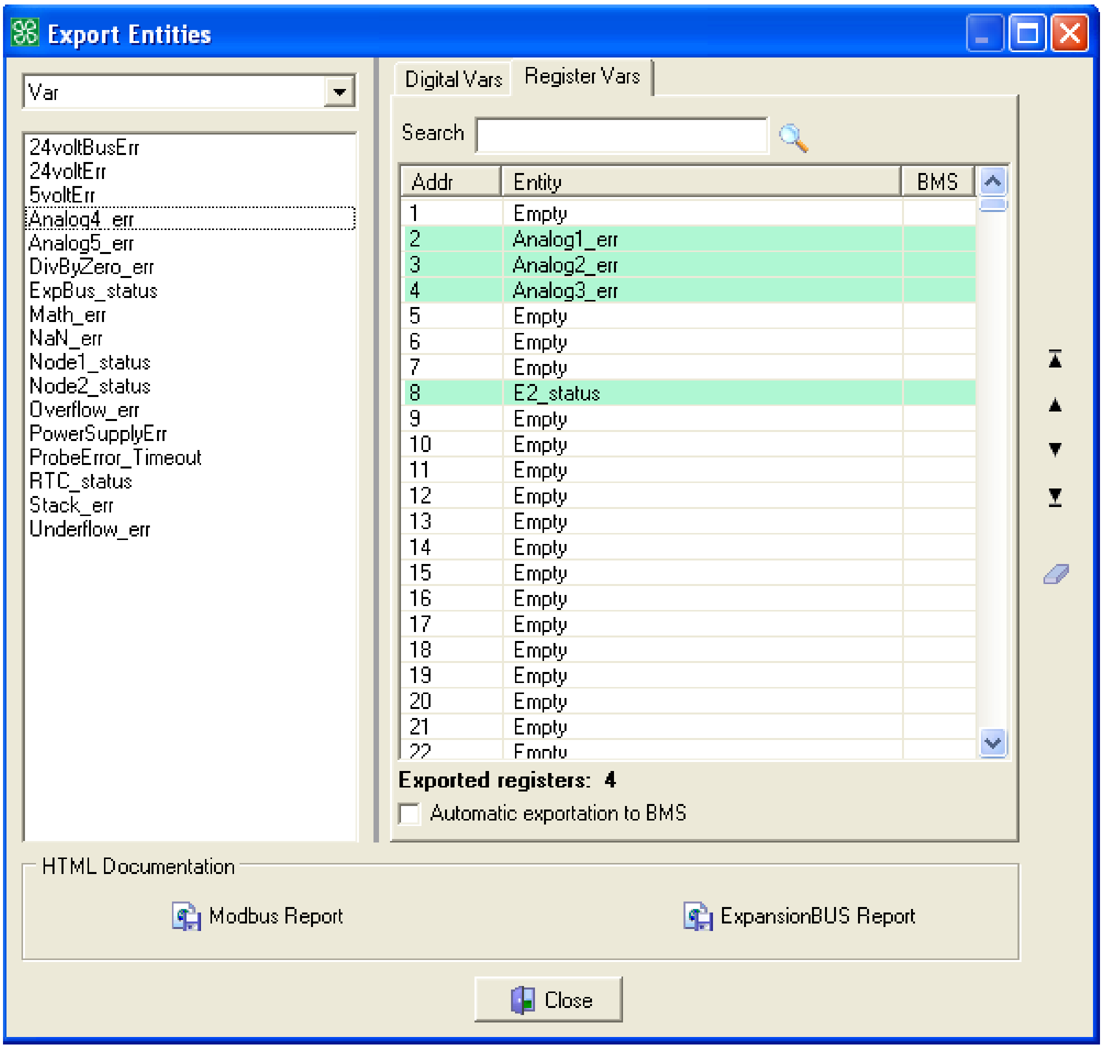
Exported registers: 0

☐ Automatic exportation to BMS

On the left side there is a list of all project variables, while on the right side there are 2 table lists that can be selected by clicking the corresponding tabs: **Digital Vars** and **Register Vars**.

The combo box at the left side allows you to filter the list by type.

To add a variable to the export table, select it from the list on the left and drag and drop it to the desired position in the table on the right.



24voltBusErr
24voltErr
5voltErr
Analog4_err
Analog5_err
DivByZero_err
ExpBus_status
Math_err
NaN_err
Node1_status
Node2_status
Overflow_err
PowerSupplyErr
ProbeError_Timeout
RTC_status
Stack_err
Underflow_err

Addr	Entity	BMS
1	Empty	
2	Analog1_err	
3	Analog2_err	
4	Analog3_err	
5	Empty	
6	Empty	
7	Empty	
8	E2_status	
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	
22	Empty	

Exported registers: 4

□ Automatic exportation to BMS

As a result of this operation, the name of the variable disappears from the list on the left and appears in the table on the right at the position indicated in the ID column.

Using this drag-and-drop method, you can move the elements to the table of exported variables or bring them back to the list of variables that have not been exported yet.

Digital Vars and Register Vars Tables

The **Digital Vars** table only exports binary values. Only certain types of variables can be exported as summarized in the table:

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**

The following Modbus function code received from Master access to the **Digital Vars** Table:

FC 01: Read coils

FC 02: Read discrete inputs

FC 05: Write single coil

FC 15: Write multiple Coils

The **Register Vars** table export 1, 8, 16 or 32 bit values.

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 2 rows in the Register Vars list. The low 16 bits are in the first row and the high 16 bits are in the second one.

The following Modbus function code received from Master access to the **Register Vars** Table:

FC 03: Read multiple registers

FC 04: Read input registers

FC 06: Write single register

FC 16: Write multiple registers

FC 23: Read Write multiple register (Max 10 registers)

Modbus and ExpBUS Documentation

The **Modbus Documentation** and **ExpBUS Documentation** buttons are exported in HTML format. The entities exported include all the useful information (including addresses).

NOTE: First address start by 1. For Master equipment starting at 0, the Modbus HTML documentation gives both addresses.

The ExpBUS Documentation document contains the information to access the project variable through index and sub-index of the **Object Dictionary**.

The Modbus document contains the information to access the project variables through Modbus read/write commands of coils and registers.

Example of the report for the Modbus protocol:

VARS DATABASE DOCUMENTATION

PROJECT INFO	
File Name	C:\Programs\SoHVAC\Test.M168p
File Date	08/04/2010
Project Name	SoHVAC Var Test
Project Author	SCHNEIDER ELECTRIC
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 VAR 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

5.8.2 Modbus Master

The Modbus master function allows the M168 controller to send data to a Modbus device or to receive data from a Modbus device.

The **Tools→Modbus Master…** menu allows you to easily connect the predefined simple Modbus slave to the controller.

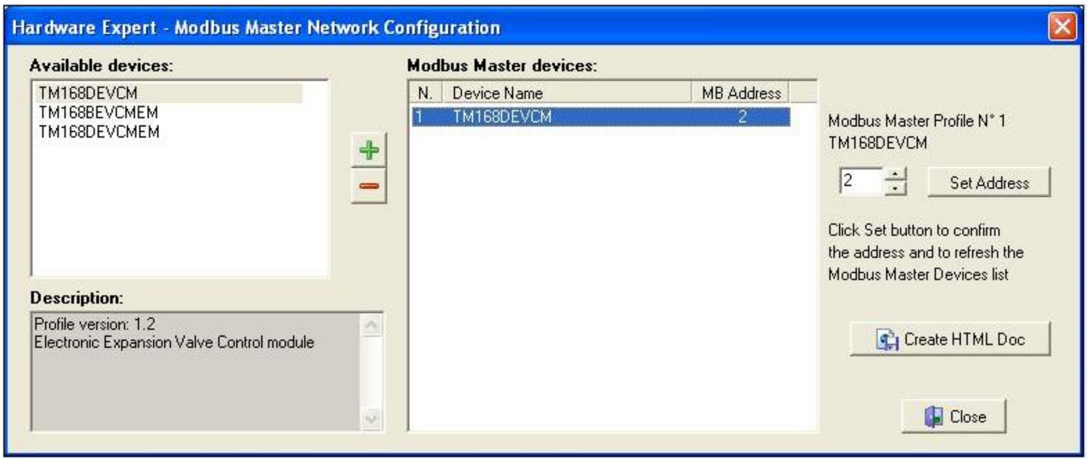
The list of Modbus slave includes the TM168DEVCM, TM168BEVCM, TM168BEVCME, TM168BEVCMEM, and TM168DEVCMEM.

When using this menu, the controller automatically manages the exchange with the slave.

NOTE: For fine tuning of the Modbus network, it is advised to use the Modbus function blocks.

To make the controller a Modbus master on MBS2 port:

1. Enable the relative configuration parameter for MBS2 in the Hardware expert-3-Serial Port.
2. Declare the slaves managed by this Modbus port through the **Modbus Master Config** menu. To do this, click **Modbus Maste Config** button in Hardware expert-3-Serial Port page.



Hardware Expert - Modbus Master Network Configuration

Available devices:

- TM168DEVCM
- TM168BEVCMEM
- TM168DEVCMEM

Description:

Profile version: 1.2
Electronic Expansion Valve Control module

Modbus Master devices:

N.	Device Name	MB Address
1	TM168DEVCM	2

Modbus Master Profile N° 1
TM168DEVCM

2

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

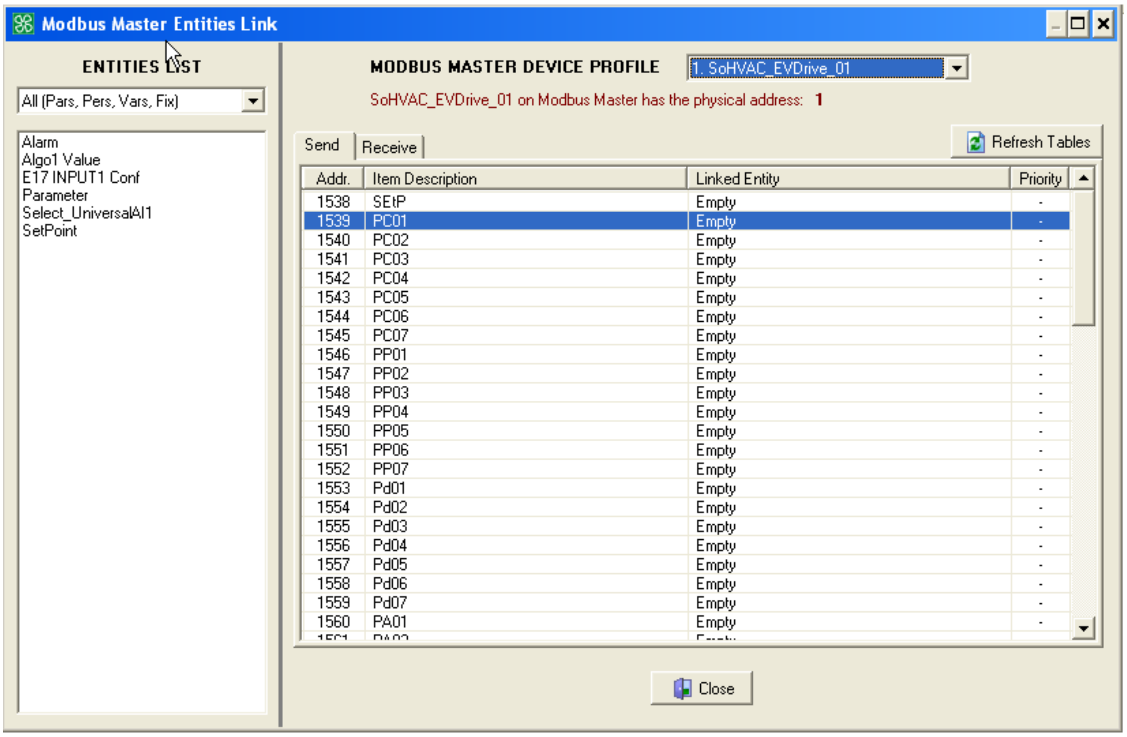
The left part shows the available Modbus profiles with a short description of the device.

You can add or remove the selected profiles to the Modbus master network definition (center windows) by using the + and − buttons.

On the right hand side you can adjust the default Modbus address for each device declared on the network.

To document the project, the **Create HTML Doc** button allows you to create an HTML document of the network.

The **Tools→Modbus Master…** menu opens the window to link the slave data to the controller program data.



Modbus Master Entities Link

ENTITIES LIST

All (Pars, Pers, Vars, Fix)

Alarm
Algo1 Value
E17 INPUT1 Conf
Parameter
Select_UniversalAI1
SetPoint

MODBUS MASTER DEVICE PROFILE: 1. SoHVAC_EVDrive_01

SoHVAC_EVDrive_01 on Modbus Master has the physical address: 1

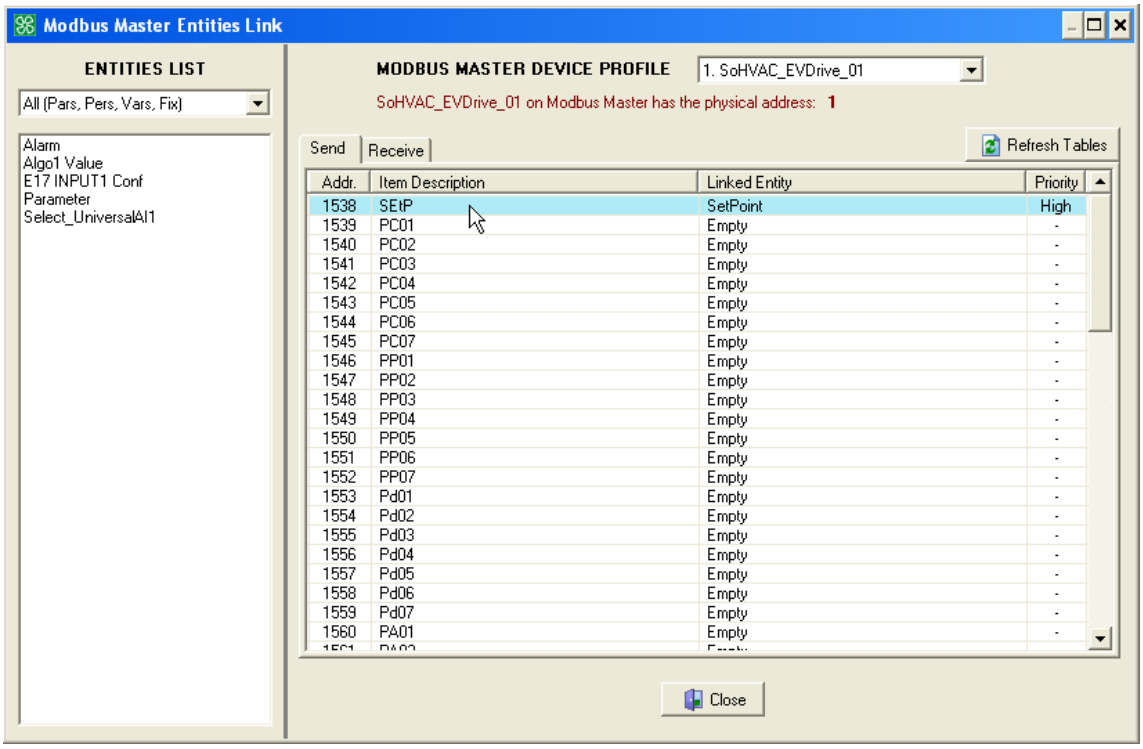
Addr.	Item Description	Linked Entity	Priority
1538	SEtP	Empty	-
1539	PC01	Empty	-
1540	PC02	Empty	-
1541	PC03	Empty	-
1542	PC04	Empty	-
1543	PC05	Empty	-
1544	PC06	Empty	-
1545	PC07	Empty	-
1546	PP01	Empty	-
1547	PP02	Empty	-
1548	PP03	Empty	-
1549	PP04	Empty	-
1550	PP05	Empty	-
1551	PP06	Empty	-
1552	PP07	Empty	-
1553	Pd01	Empty	-
1554	Pd02	Empty	-
1555	Pd03	Empty	-
1556	Pd04	Empty	-
1557	Pd05	Empty	-
1558	Pd06	Empty	-
1559	Pd07	Empty	-
1560	PA01	Empty	-

In the combo box on top, select the Modbus slave you want to access.

The main window shows the writeable (send) and readable (receive) data declared for this slave.

To link a program entity with the available variable on the slave, select the entity and drag-and-drop it in the correct line of the main window.

In the following example, the **SetPoint** entity is linked to the variable 1538 SEtP of slave 1: SoHVAC_EVDrive_01.



Modbus Master Entities Link

ENTITIES LIST

All (Pars, Pers, Vars, Fix)

Alarm
Algo1 Value
E17 INPUT1 Conf
Parameter
Select_UniversalAI1

MODBUS MASTER DEVICE PROFILE: 1. SoHVAC_EVDrive_01

SoHVAC_EVDrive_01 on Modbus Master has the physical address: 1

Addr.	Item Description	Linked Entity	Priority
1538	SEtP	SetPoint	High
1539	PC01	Empty	-
1540	PC02	Empty	-
1541	PC03	Empty	-
1542	PC04	Empty	-
1543	PC05	Empty	-
1544	PC06	Empty	-
1545	PC07	Empty	-
1546	PP01	Empty	-
1547	PP02	Empty	-
1548	PP03	Empty	-
1549	PP04	Empty	-
1550	PP05	Empty	-
1551	PP06	Empty	-
1552	PP07	Empty	-
1553	Pd01	Empty	-
1554	Pd02	Empty	-
1555	Pd03	Empty	-
1556	Pd04	Empty	-
1557	Pd05	Empty	-
1558	Pd06	Empty	-
1559	Pd07	Empty	-
1560	PA01	Empty	-

The program automatically sends the value of the **SetPoint** entity to the register 1538 of the slave at address 1.

The main window shows the following information:

- **Addr**: Slave register address
- **Item Description**: Description of the slave register.
- **Linked Entity**: **SoHVAC** entity linked to this register (the program automatically manages the transmission).
- **Priority**: Configured priority of the synchronisation. The Priority can be changed in the entity property, field MasterRefresh.

The controller refreshes the send and receive item linked to the entities. The items not linked to the entities are ignored during the Modbus exchanges.

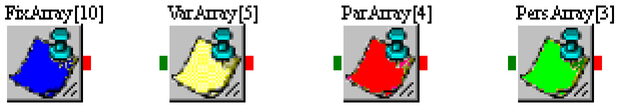
NOTE: Variables of array type can be assigned only to the same Modbus device.

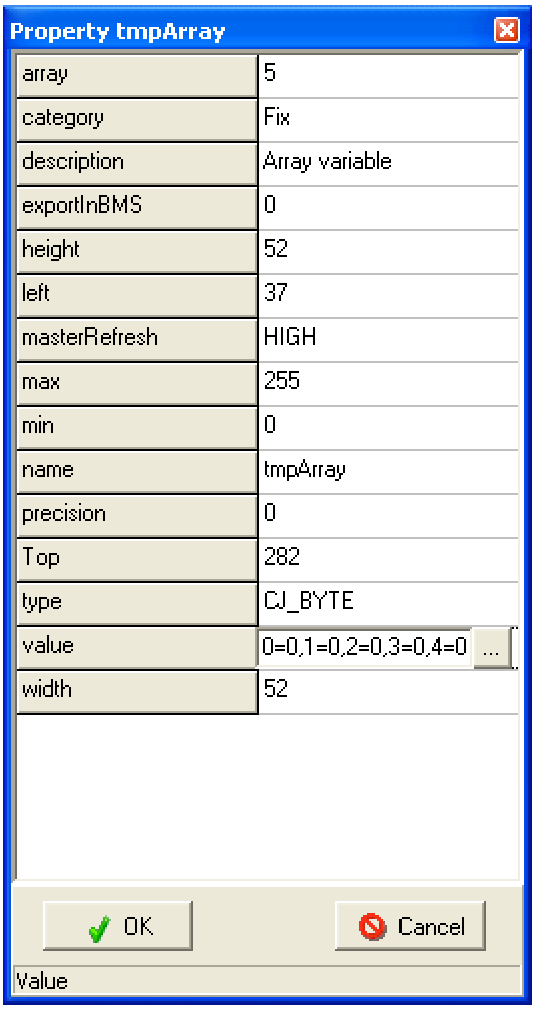
5.9 Using Array Entities

Array is a data structure containing N numerical information of the same type. The property array is available only when the data types are: CJ_VOID, CJ_BIT, CJ_BYTE, CJ_S_BYTE, CJ_SHORT, CJ_WORD, CJ_DWORD, CJ_LONG and CJ_CHAR.

Array is a modifiable property only for the following entities:

- PAR - parameters
- FIX - constants
- PERS - persistent
- VAR - variables
- Input and outputs of the algorithms.




Property tmpArray

array	5
category	Fix
description	Array variable
exportInBMS	0
height	52
left	37
masterRefresh	HIGH
max	255
min	0
name	tmpArray
precision	0
Top	282
type	CJ_BYTE
value	0=0,1=0,2=0,3=0,4=0
width	52

Value

The array name is modified (when array property is greater than 1) with the array size inserted between square brackets.

5.9.1 Array Property

Array property is a numeric type that has a maximum value of 100 and a minimum value of 1 (mono-dimensional entity) when array data value is greater than 1.

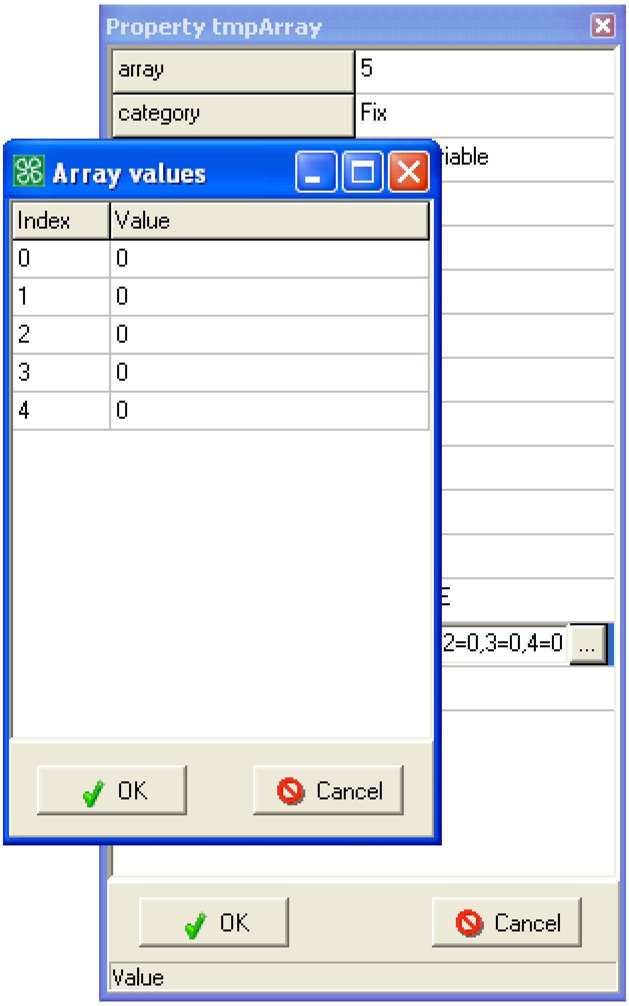
5.9.2 Value Property

Value property becomes a string to match array index with the correct array value. The syntax is as follows:

`0=value[0], …, array-1=value[array-1]`

As the mono-dimensional entities also for the array entities, it is possible to set default values also.

By clicking the ... button near the value property, the following window for editing the default values is proposed:



array	5
category	Fix
	iable
	E
	2=0,3=0,4=0 ...

Index	Value
0	0
1	0
2	0
3	0
4	0

By clicking the **Ok** button, the edited values are accepted and automatically set up the value property as shown in the picture.

By clicking the **Cancel** button, the edited values are not accepted and maintain the previous values.

5.9.3 Min, Max, Precision Property

All value properties, for example, min, max and precision, limits 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 are represented with one decimal place.

5.9.4 Compatibility and Link

Two entities (with array property greater than 1) are compatible one another only for the 2 following conditions:

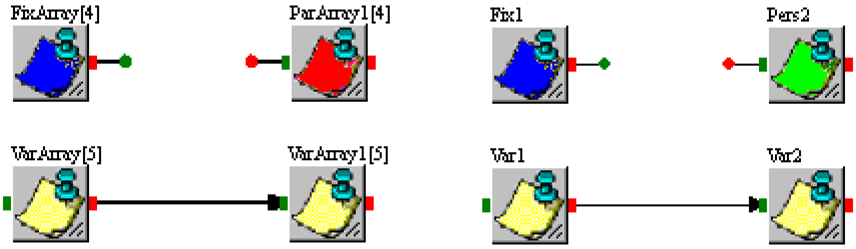
1. If both the entities have the same array size (same array property value).
2. If the entities are of the same data type with the exception:

- When start entity data type is **CJ_BIT** and linked entity data type is **CJ_S_BYTE**.
- When start entity data type is **CJ_BIT** and linked entity data type is **CJ_BYTE**.

In all other cases, entities are incompatible. Linking 2 incompatible array entities shows error message and the link is aborted.

A single-dimensional entities link can be visualized different from an array entities link using the Array Segment Thickness setting from the **Preference** menu (Array Segment Thickness default value = 2).

In this way the thickness of the lines connected to the entities are different.

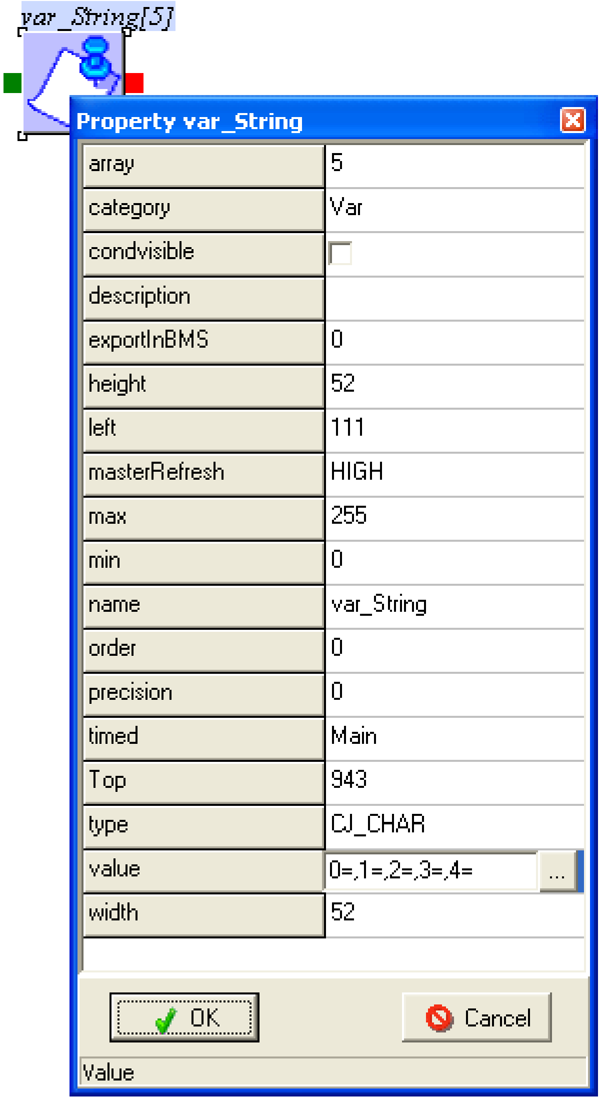


5.10 Type CJ_CHAR for String Management

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

Example:

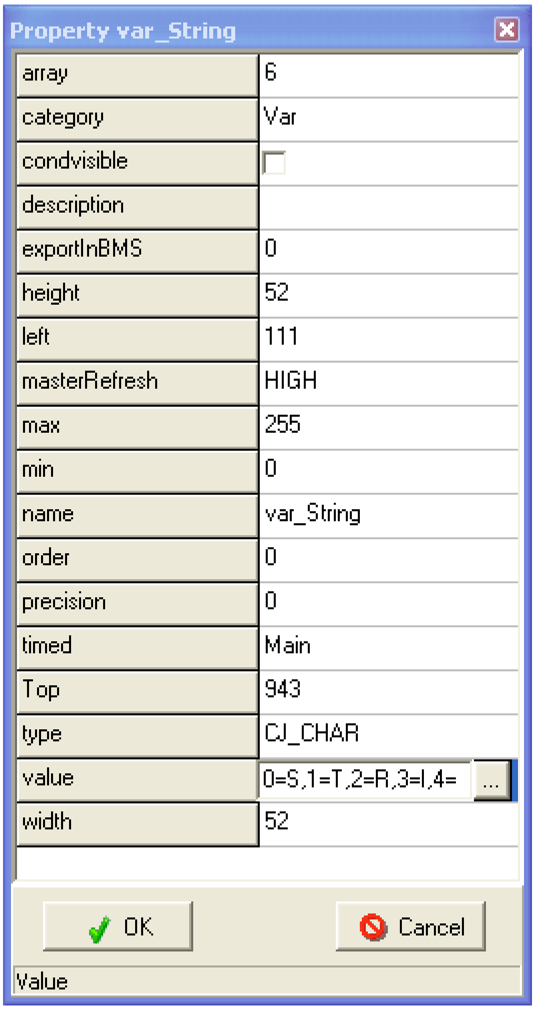
The figure shows a sample of a string using 5 characters:



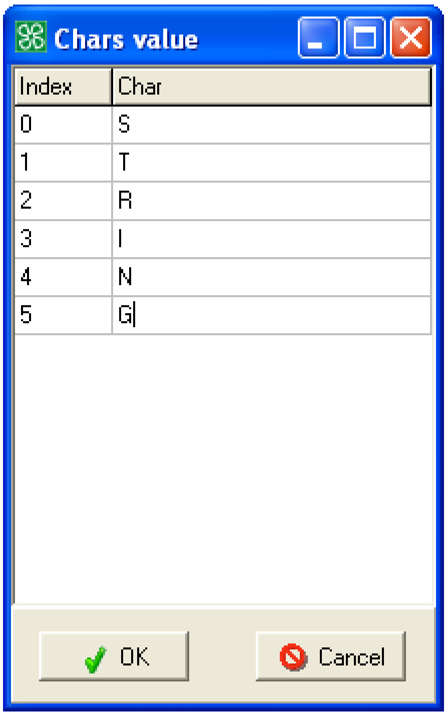
array	5
category	Var
condvisible	□
description	
exportInBMS	0
height	52
left	111
masterRefresh	HIGH
max	255
min	0
name	var_String
order	0
precision	0
timed	Main
Top	943
type	CJ_CHAR
value	0=,1=,2=,3=,4= ...
width	52

To set the default value of the single characters, click the button that appears when you select the value property.

A window appears similar to the array values manage.



array	6
category	Var
condvisible	□
description	
exportInBMS	0
height	52
left	111
masterRefresh	HIGH
max	255
min	0
name	var_String
order	0
precision	0
timed	Main
Top	943
type	CJ_CHAR
value	0=S,1=T,2=R,3=I,4= ...
width	52



Index	Char
0	S
1	T
2	R
3	I
4	N
5	G

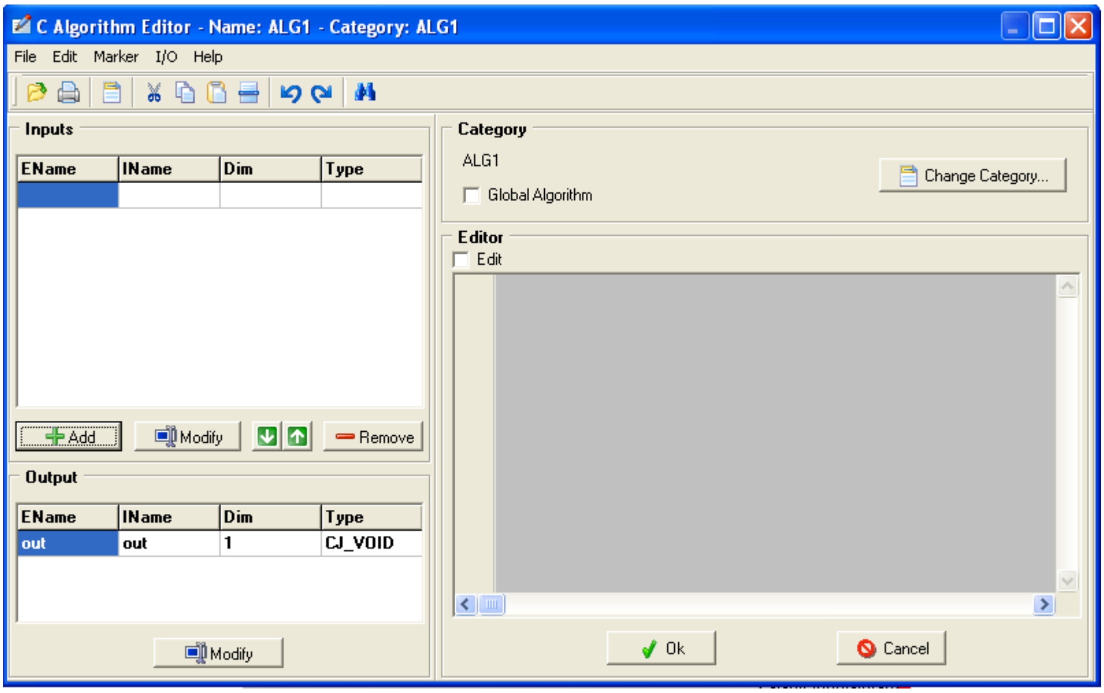
Once the fields are set up, click **OK** and the value is updated.

NOTE:

- To enter the NULL character, leave the char value empty.
- When displaying **CJ_CHAR** on **EIML** page, as the first elements 0…31 of the ASCII table are control codes, a right ended arrow is displayed.

5.11 Using C Algorithm Editor

The C Algorithm Editor is a graphic tool that allows you to specify the behavior of each algorithm. To open it, either double-click the algorithm or select **Show** from the context menu.



Inputs

EName	IName	Dim	Type

Output

EName	IName	Dim	Type
out	out	1	CJ_VOID

Category: ALG1

□ Global Algorithm

□ Edit

An algorithm is characterized by a series of inputs, a single output and a code that describes its behavior.
C Algorithm Editor is divided into multiple sections:

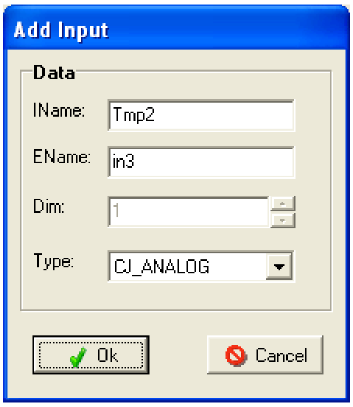
- On the left side of the **C Algorithm Editor** window, there are 2 tables that summarize the properties of the input and the outputs.
- At the top of the window, there is an algorithm category.
- At the center, the built-in editor.

5.11.1 Input and Output Properties

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

- **EName**, or external name, which allows you to identify the input terminal in the sheet.
- **IName**, or internal name, that is the name used in the code. It can be easily entered by double-clicking 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. refer to *Using Array Entities*.
- **Type**, data type to be processed.

To add a new input without exiting C Algorithm Editor, click the **Add** button. The following window appears where you can enter the name, size and type of the new input:



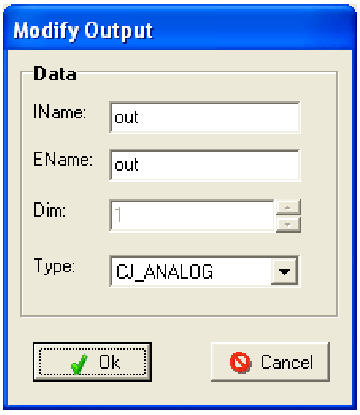
IName: Tmp2
EName: in3
Dim: 1
Type: CJ_ANALOG

If you select an input and click the **Modify** button, a similar window appears where you can change the name and type of the selected input.

By clicking the **Remove** button, you can remove the selected input from the algorithm.

The **Modify Output** table allows displaying and modifying the output of an algorithm. The algorithm can have only one output.

The output has the same properties as the input.



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

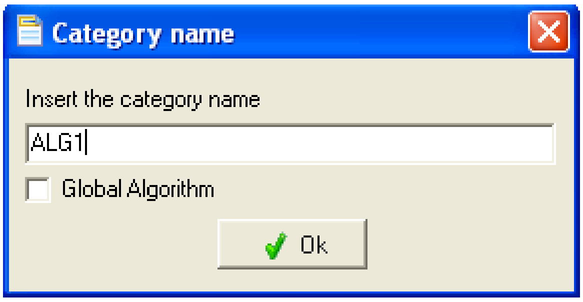
5.11.2 Algorithm Category

At the top of the window there is an algorithm category. The concept of category is similar to the concept of class in object-programming. It allows you to use the same executable code for all the algorithms belonging to the same category, thus optimizing resources.

2 algorithms of the same category must have the same number and type of inputs and output and the same code (refer to *Optimizations*).

Click the **Change category…** to enter a new category.

The following window appears where you can enter a new category:



Insert the category name
ALG1
□ Global Algorithm

5.11.3 Built-in Editor

In the center of the window there is the Editor section, where you can specify the algorithm code. To change or add a code, check the Edit box.

To change an algorithm, the system verifies other algorithms in the same category. If algorithms are there, it asks you to change category or else make the same change to all the algorithms of the same category.

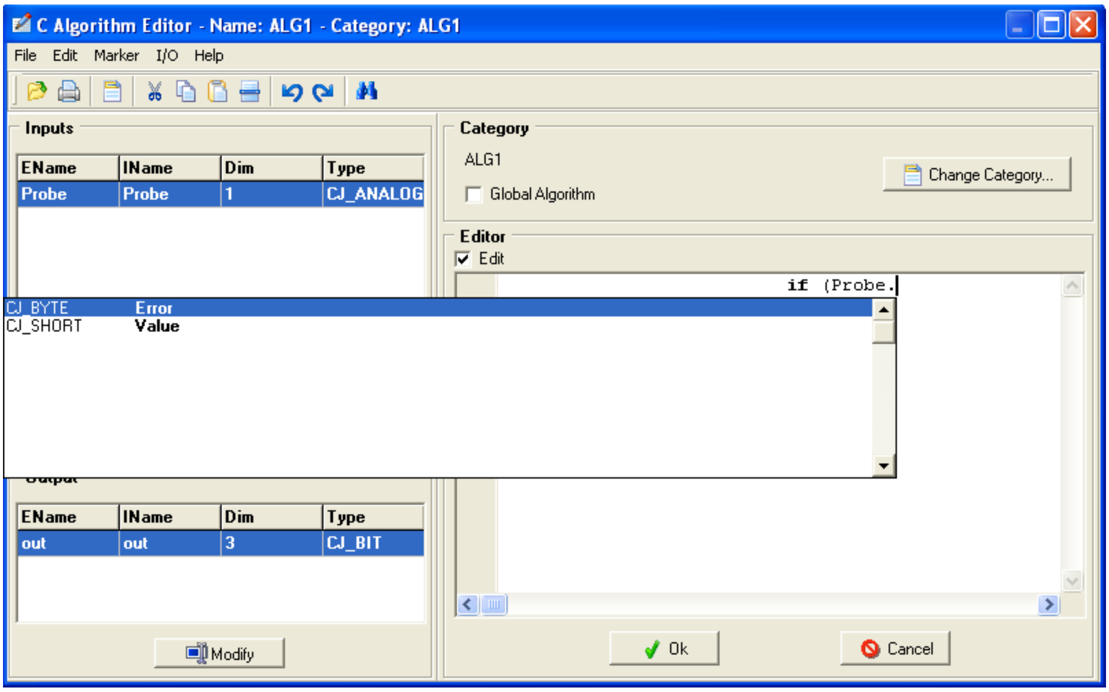
The editor is capable of recognizing the ANSI C syntax and provides functions to manipulate the code of the algorithm:

- code auto-completion
- importing code prepared using a different editor from a file
- exporting the code to a file
- printing the code
- find and replace
- cut, copy, and paste code portions
- use markers

The code auto-completion function is helpful for developers, as it allows them to complete the syntax of predefined structures, or functions without verifying the documentation every time.

In the case of structures, the auto-completion window appears when you type the dot before the structure field name.

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 do not remember the right syntax, you can type “Probe.” in the code and wait for the auto-completion window to appear as shown in the following figure:



Inputs

EName	IName	Dim	Type
Probe	Probe	1	CJ_ANALOG

Category

ALG1

□ Global Algorithm

Editor

☑ Edit

```
if (Probe.
```

CJ_BYTE	Error
CJ_SHORT	Value

Output

EName	IName	Dim	Type
out	out	3	CJ_BIT

If you press the CTRL and SPACE keys at the same time, a new window appears showing code editing information such as:

1. Functions and constant factors of the different drivers that may be used in the algorithms
2. C language keywords
3. Data types that may be used
4. The name of the inputs in the algorithm
5. Project defines

By pressing CTRL+SHIFT+SPACE, all the parameters of a function are displayed.

When entering the code:

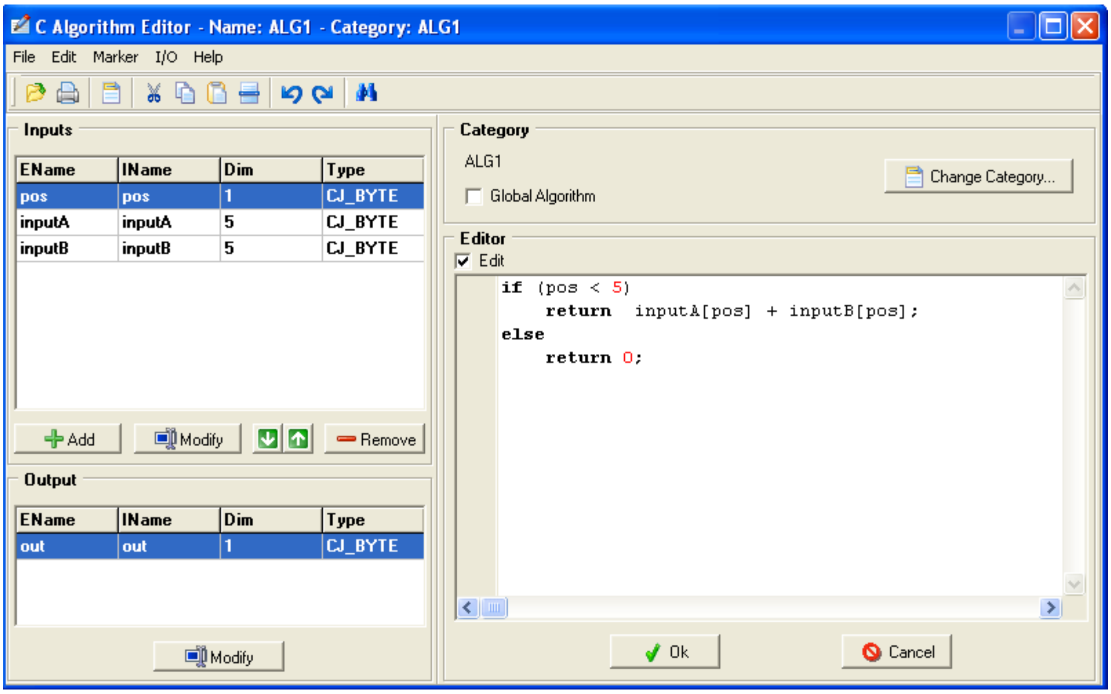
- Make sure that there is always a return value using the C syntax keyword return (or the compiler gives you an advisory message).
- Avoid the use of static variables (that is, assign the attribute static to a variable). If used, the algorithm created must be used only once in the project.
- Use the types included in the SoHVAC system, refer to *Creating a New Project*.
- Refer to the *Structured Data Types*, if you have structure-type inputs (such as **CJ_ANALOG**).
- When using loops, avoid heavy computational loads and infinite loops.

5.11.4 Algorithm with Array Inputs/Outputs

Below are 2 simple examples about the array inputs and outputs which shows the correct way to write C algorithms with the SoHVAC Code Editor.

Array Inputs

Having 2 CJ_BYTE array inputs both bigger in value than 1, you have to return in the algorithm the sum of the values contained in the position pos of both the array inputs.



Inputs

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

Output

EName	IName	Dim	Type
out	out	1	CJ_BYTE

Category

ALG1

□ Global Algorithm

Editor

☑ Edit

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

In this example, the 2 inputs have the same array size, but they can have different size.

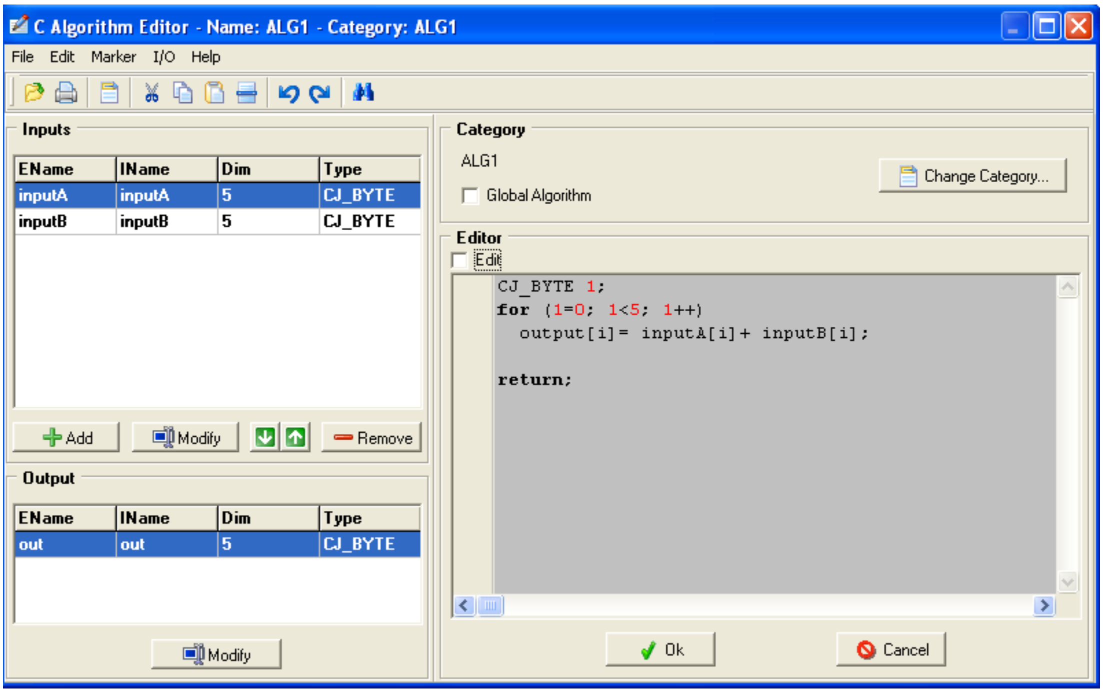
Both of the 2 inputs can contain 5 numeric information. For the access in every cell of the array, use the following C syntax:

```
nameVar [ index ]
```

Index is the access index to the array, it always starts from 0. Therefore the first array element is at the index 0, and the last array element is at the index Dim-1.

Array Inputs and Array Outputs

To sum each 2 input array elements 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	out	5	CJ_BYTE

ALG1

□ Global Algorithm

□ Edit

```
CJ_BYTE 1;
for (1=0; 1<5; 1++)
  output[i]= inputA[i]+ inputB[i];

return;
```

For the code optimization, use a **for** cycle in order to access all the array indexes.

The variable output of the algorithm called output, different 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 computer programming).

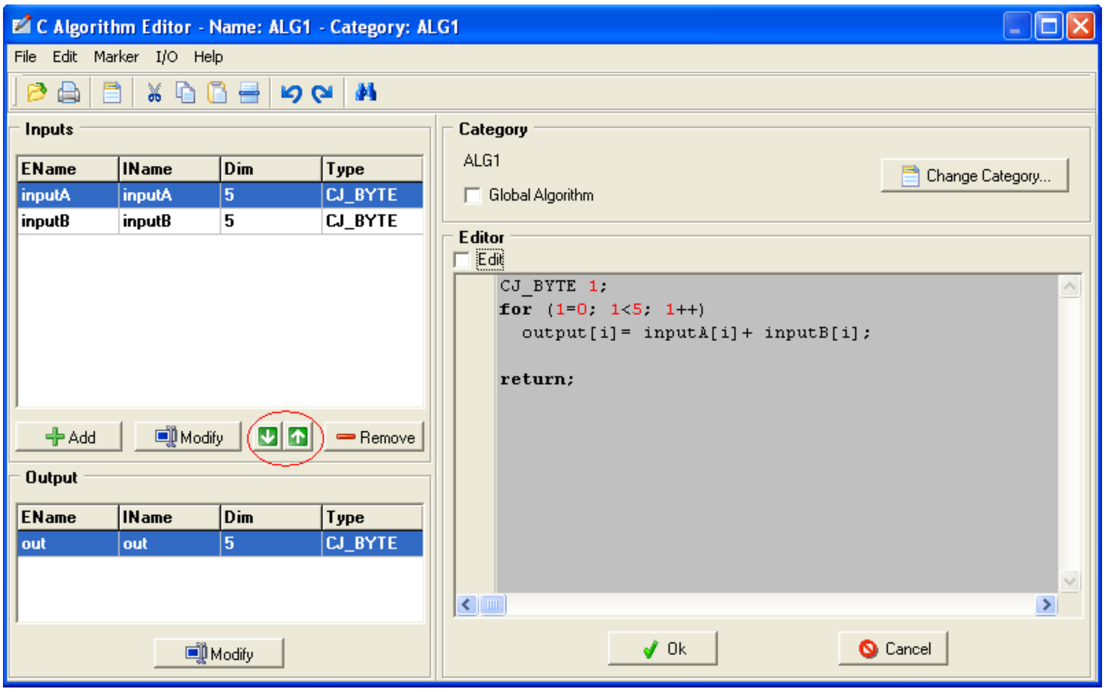
The modification and the access to the output are directly made on the same output and not on a copy of the temporary variable.

When the output of an algorithm is an array entity:

- the output has to be directly used in the C code editing
- the name of the output must be the same as 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 an advisory message during the project compilation

5.11.5 Possibility to Change the Order of Input Terminals of the Algorithms Added

Without removing connections, the order of the terminal inputs of the algorithms can be edited using the special arrows in the Code Editor.



C Algorithm Editor - Name: ALG1 - Category: ALG1

Inputs

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

Output

EName	IName	Dim	Type
out	out	5	CJ_BYTE

□ Global Algorithm

□ Edit

```
CJ_BYTE 1;
for (1=0; 1<5; 1++)
  output[i]= inputA[i]+ inputB[i];

return;
```

NOTE: This function cannot be used if there are exported algorithm inputs.

5.11.6 Operating Limits

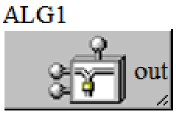
For more information about the operating limits and memory management when using C language in the editor, refer to the *SoHVAC C Programming Language User Guide*.

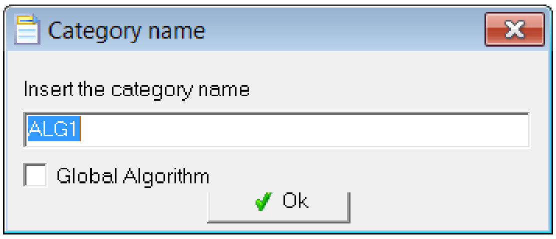
5.11.7 Global Algorithms

A global algorithm is a function, sub-algorithm, which can be used in one or more other algorithms.

Global algorithms can be used only in an application program. You cannot save or use global algorithms in a library function block.

To create a global algorithm, select the **Generic Algorithm** function from the software library and drag and drop it in the working sheet. The following message box appears enabling you to define the category name:





Category name

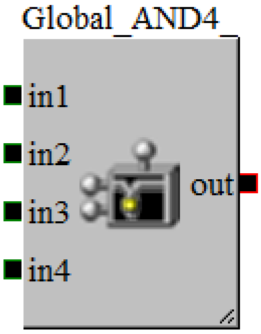
Insert the category name

ALG1

□ Global Algorithm

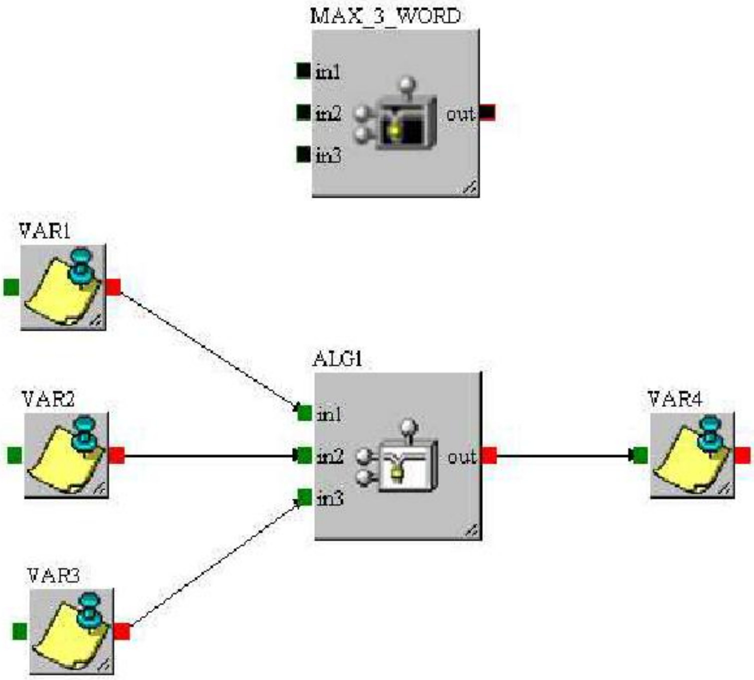
Select the **Global Algorithm** check box to define the algorithm as global algorithm.

The **C Algorithm Editor** window opens to define the inputs and output and to create the function in c-language. After the **C Algorithm Editor** window is closed, the icon of the algorithm and also the input and output pins appears in black color. This action does not allow you to connect other function blocks to the input or output pins.

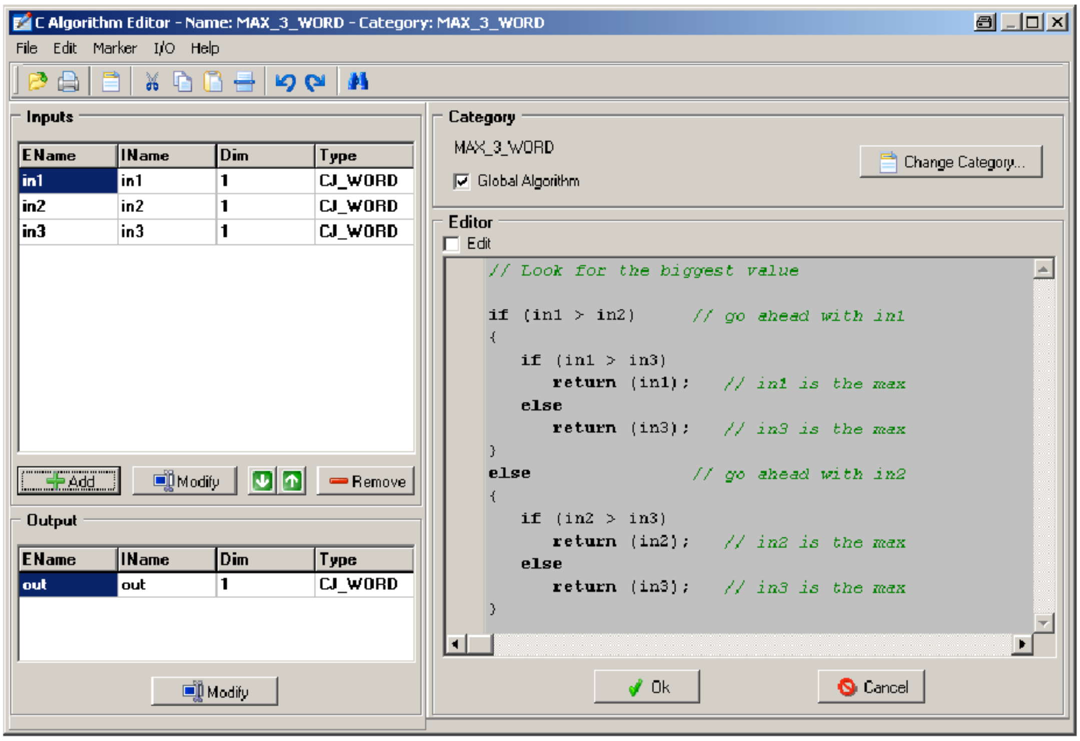


You can now use the global algorithm in other algorithms. Open the **C Algorithm Editor** and press CTRL+Spacebar in the editor to see a list of built-in functions, including the global functions defined in the SoHVAC project.

The following figure shows an example for usage of global algorithm:



The global algorithm **MAX_3_WORD** window is shown in the figure:



Inputs

EName	IName	Dim	Type
in1	in1	1	CJ_WORD
in2	in2	1	CJ_WORD
in3	in3	1	CJ_WORD

Output

EName	IName	Dim	Type
out	out	1	CJ_WORD

Category

MAX_3_WORD

☑ Global Algorithm

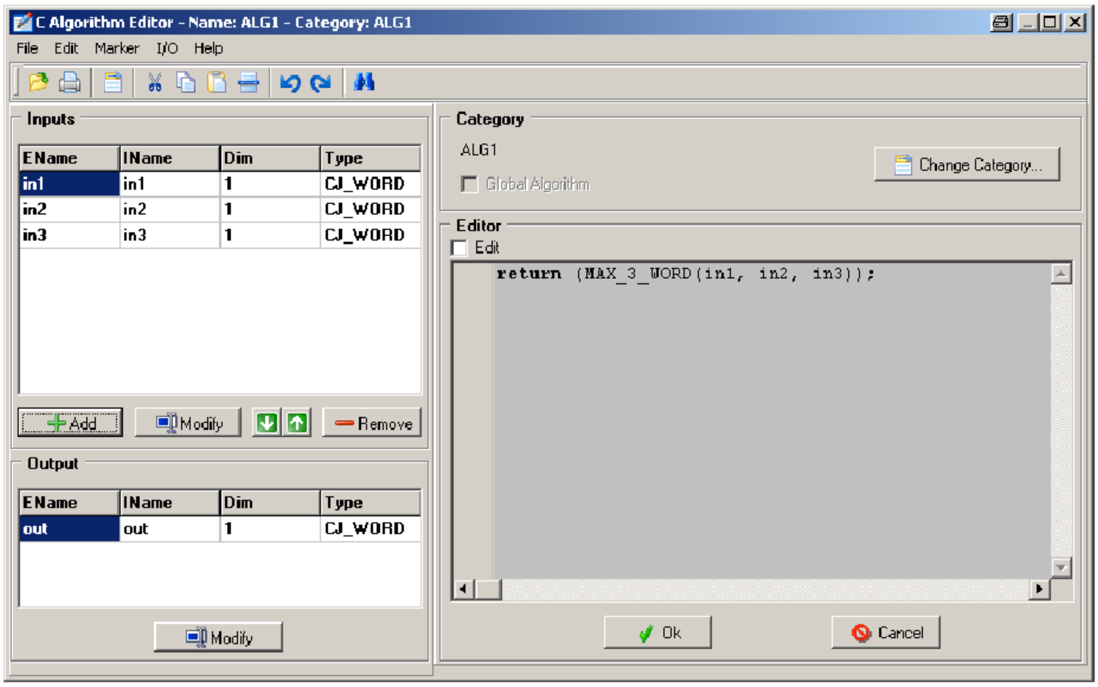
Editor

□ Edit

```
// Look for the biggest value

if (in1 > in2)      // go ahead with in1
{
   if (in1 > in3)
      return (in1);   // in1 is the max
   else
      return (in3);   // in3 is the max
}
else                // go ahead with in2
{
   if (in2 > in3)
      return (in2);   // in2 is the max
   else
      return (in3);   // in3 is the max
}
```

The algorithm ALG1 with call of the global algorithm **MAX_3_WORD** is shown in the figure:



Inputs

EName	IName	Dim	Type
in1	in1	1	CJ_WORD
in2	in2	1	CJ_WORD
in3	in3	1	CJ_WORD

Output

EName	IName	Dim	Type
out	out	1	CJ_WORD

Category

ALG1

□ Global Algorithm

Editor

□ Edit

```
return (MAX_3_WORD(in1, in2, in3));
```

5.12 Creating Libraries

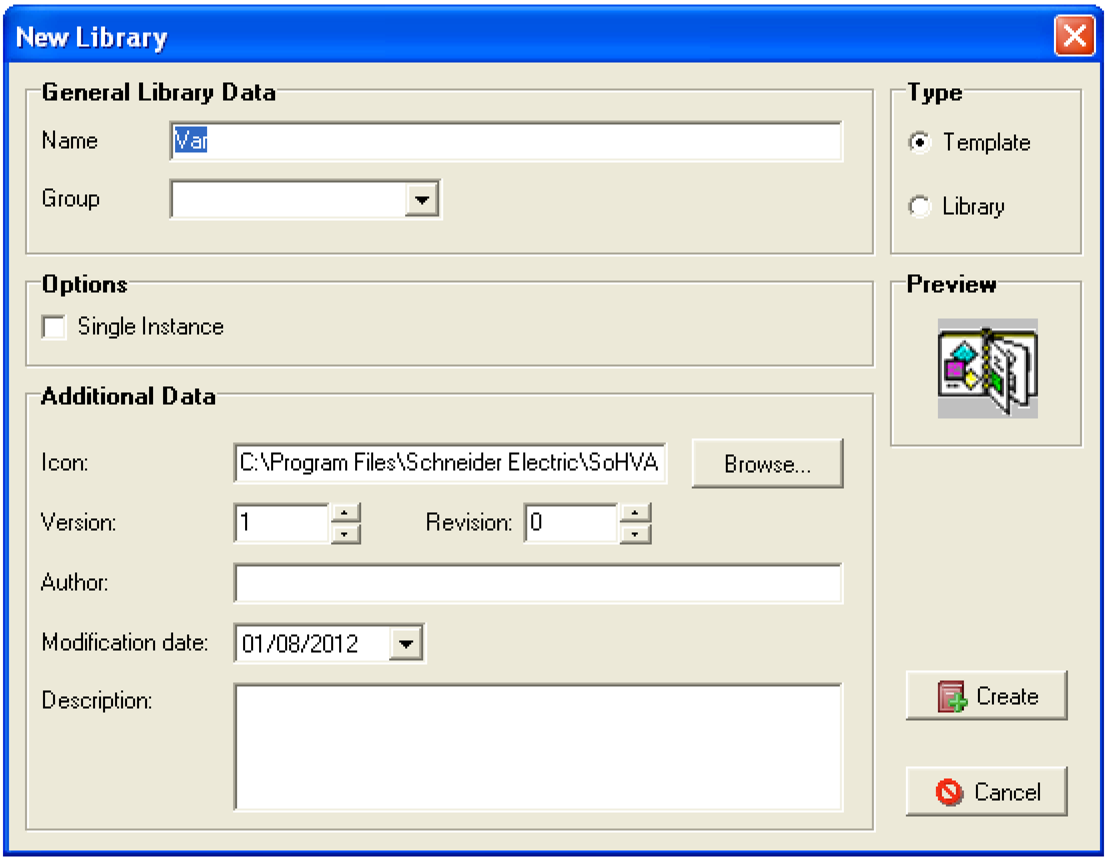
The possibility of storing a series of objects in library format allows you to build up collection of objects which you can reuse in other projects.

This enables significant time savings in development, 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 and select **New library** from the context menu.
- Select **Library→New library** from the menu.
- Click the icon in the toolbar.

The following window appears:



New Library

General Library Data

Name: Var

Group:

Type

● Template

○ Library

Options

□ Single Instance

Preview

Additional Data

Icon: C:\Program Files\Schneider Electric\SoHVA

Version: 1

Revision: 0

Author:

Modification date: 01/08/2012

Description:

To create a new library, you need to specify certain characteristics that can be divided into 4 sections:

- **General Data**
- **Options**
- **Additional Data**
- **Type**

5.12.1 General Data

In the General Data section, you can specify the library name and the group to which it belongs.

You can select the group in which you want to store the library. You can also create a new group by typing a new name.

5.12.2 Options

The Single Instance property (if selected) adds particularity to the library so that it can be used once in the project in which it has been added.

This function is used when you create libraries in which the multiple presence of the function creates problem in the working of the program.

For example, the library containing algorithms with static variables or functions to call the firmware drivers of the application.

5.12.3 Additional Data

In the Additional Data section, you can specify all the available options to customize each individual library.

You can link it to an icon. Enter a version number, the author name, the creation/revision date and a brief description.

5.12.4 Type

In the Type section, you can select the library type:

- A Template is a group of reusable entities that can be modified. After you have added it to a project, you can double-click its icon to display and change its contents.
- A Library is a group of entities that cannot be opened or changed. 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 function block with the same group/name not yet exists, the new library function block is immediately added to the library manager with the chosen icon. Otherwise the system requires a confirmation before overwriting.

If you place mouse pointer over the new library in the toolbar, you can display information such as name, version, latest revision date, description, and type.

To add it to a new project, select it and click inside the sheet, as you add any other element.

5.13 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).

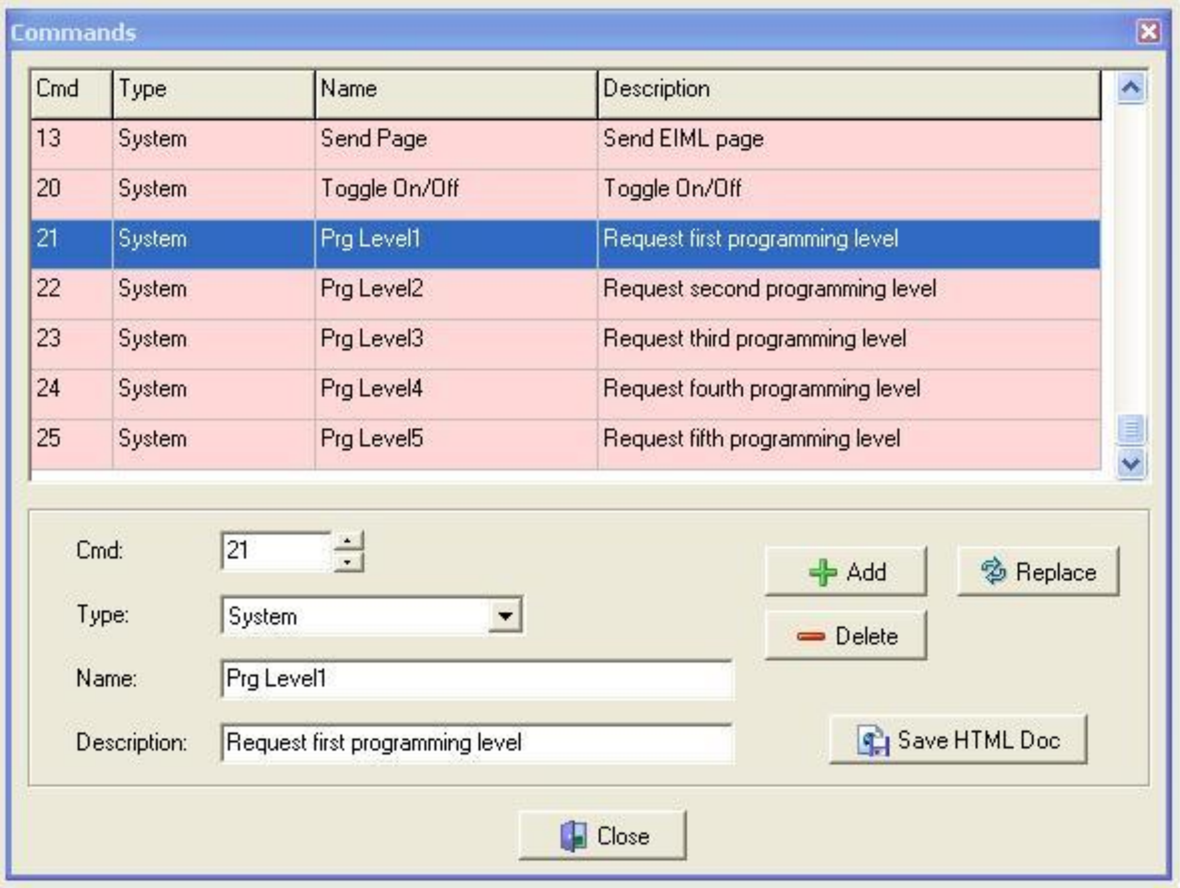
Commands are divided into 3 categories: system, environment, and project commands.

System commands have a unique value assigned by the system and valid for all products. In the case of environment and project commands, their values can be assigned at the discretion of the developer.

Environment commands can be used in all projects, while project commands are project-specific.

Commands Window

Select **Tools→Commands…** from the menu to open the window that shows the system commands available and allows you to add or change environment and project commands.



Cmd	Type	Name	Description
13	System	Send Page	Send EIML page
20	System	Toggle On/Off	Toggle On/Off
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

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)

With **Add** button, it is possible to add commands with value from 51 to 250.

With **Delete** button, it is possible to remove environment and project commands.

To modify command information, select the **Command In** the top windows, change the **Name** or **Description** and click the **Replace** button.

Save HTML Doc button allows the commands to be exported in an HTML format.

5.13.1 System Command Description

List of all available system commands and their meanings:

- Cmd 1: **Load Page**: Command to request/send a page. The corresponding parameter indicates the index of the requested page. The **LoadPage** command cannot be used in broadcast mode. For each target device, an individual **LoadPage** command is needed.
- Cmd 2: **Mute Buzzer**: Command to silence the buzzer at the device, controller or display which is identified by the expansion bus node. If the node field is set to broadcast, this command silences all devices on the network. The parameter is not used in this command.
- Cmd 3: **Sync Clock**: Command to synchronize the clock of the destination node with the clock of the source node. The parameter is not used in this command.
- Cmd 4: **Save Par Drv**: Command to create a backup copy of all system parameters. The parameter is not used in this command.

NOTE: This operation may take some time to complete.

System parameters are meant for the configuration of controllers, that is, for the configuration of **ExpBUS**, Modbus, AI sensor type, AO actuator type, Password, Backlight, Date&Time visualization, and Language.

- Cmd 5: **Save Par App**: Command to create a backup copy of all application parameters. The parameter is not used in this command.

NOTE: This operation may take some time to complete.

Application parameters are user defined parameters.

- Cmd 6: **Restore Par Drv**: Command to restore all system parameters from the backup copy. The parameter is not used in this command.

NOTE: This operation may take some time to complete.

- Cmd 7: **Restore Par App**: Command to restore all application parameters from the backup copy. The parameter is not used in this command.

NOTE: This operation may take some time to complete.

- Cmd 8: **Load Default Par**: Command to load the default values of all parameters set during the design phase. The parameter is not used in this command.

NOTE: This operation may take some time to complete.

- Cmd 11: **Erase Historian**: Command to erase the Event Historian. The parameter is not used in this command.

NOTE: All data will be deleted.

- Cmd 13: **SendPage**: Display the required Page on the target display (refer to Navigating through the Pages). The corresponding parameter indicates the index of the requested page. The **SendPage** command cannot be used in broadcast mode. For each target device, an individual **SendPage** command is needed.
- Cmd 20: **Toggle ON/OFF**: On/Off command. Allows you to send a ON/OFF signal. The management of this command must be implemented in the project. The Toggle ON/OFF command is triggered when the **ESC** key is pressed for more than 3 seconds. The **ESC** key can be used to switch on or off of the machine.
- Cmd 21 to 25: **Prg Level 1...5**: Command to send the page relative to the password level with the smallest index number to the target display. The parameter is not used in this command (to enable this feature, the function block: **EnablePrgLevel** must be used in the application program).
- Cmd 32: **Key Save Par Drv**: Command to save all system parameters on the parameter key. The parameter is not used in this command.

- Cmd 33: **Key Save Par App**: Command to save all application parameters on the parameter key. The parameter is not used in this command.
- Cmd 34: **Key Restore Par Drv**: Command to restore all system parameters from the parameter key. The parameter is not used in this command.
- Cmd 35: **Key Restore Par App**: Command to restore all application parameters from the parameter key. The parameter is not used in this command.
- Cmd 38: **Send EVCM Command**: Send the Enable or disable command to the electronic expansion valve driver. The Node indicates the target electronic expansion valve driver. The parameter is a set of bits and indicates the command value:
  - Bit 0 = Enable valve request: 1=Enable, 0=Disable
  - Bit 1 = Resynchronization request: 1=Request
  - Bit 2 = Functioning mode: 0=Auto, 1=Manual
  - Bit 3 = SH Control parameters selection: 0=Set #1, 1=Set #2
  - Bit 4...7 = Reserved
  - Bit 8...15 = Bit 0...7 Mask
- Cmd 39: **Send EVCM Manual Position**: Send the manual valve position to the electronic expansion valve driver. The Node indicates the target electronic expansion valve driver. The parameter indicates the manual valve position value, value range 0...10000 (0…100.00%).
- Cmd 40: **Receive EVCM Current Pos**: Receive the current valve position of the electronic expansion valve driver. The Node indicates the target electronic expansion valve driver. This event is sent by the electronic expansion valve driver every 5 sec and also when the valve position falls below 5%.
- Cmd 41: **Receive EVCM Status**: Receive the current status of the electronic expansion valve driver. The Node indicates the target electronic expansion valve driver. The parameter is a set of bits and indicates the status value:
  - Bit 0...7 = FSM status
  - Bit 8 = Enable valve status
  - Bit 9 = Resynchronization request status
  - Bit 10 = Used SH control parameters St: 0=Set #1, 1=Set #2
  - Bit 11...15 = Reserved

  This event is sent every 5 sec or by a change of value by the electronic expansion valve driver.
- Cmd 42: **Receive EVCM Alarms**: Receive the current alarm status of the electronic expansion valve driver. The Node indicates the target electronic expansion valve driver. The parameter is the alarm status (see the *Electronic Expansion Valve Driver User Manual*). This event is sent every 5 sec or by a change of value by the electronic expansion valve driver.

5.13.2 Create a Command

The structure of a command is composed of 3 fields:

1. **COMMAND**: indicates the numeric value of the command. A name is linked to each command, which is preset for system commands, valid for all projects for environment commands (51 to 150), and project-specific for project commands (151 to 250).
2. **PARAMETER**: is a 16-bit value linked to the command. Depending on the command, it can be seen as a single field or as the combination of multiple fields.
3. **NODE**: is the value of the network element sending (or receiving) the command. This value is assigned by means of the corresponding join window.

A command is emitted linked to Human action on the Displays or through the program.

It is then sent to a particular Node of the ExpBUS, and it proceeds as follows:

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

   A **Command Out** entity embedded in a sheet is activated when the trigger input goes from 0 to 1.
2. The Command Out needs to be linked to one HW through the Project→Join Tool →Command Out menu.
3. The command linked program is activated with the help of the Command In output. The Command In entity needs to be linked through the Project→Join Tool→Command In menu.

5.14 Defining Execution Tasks

All calculation operations defined through links of entities can be executed in 2 different tasks.

Usually the greater part of operations is executed in the Main Task, where it is necessary to calculate some critical operations.

It can be executed in the Timed100ms high priority tasks, which execute under interrupts every 100 ms.

Notice that these last 2 tasks must be used only when it is necessary and their overload can introduce execution errors.

Main task and 100 ms execution time can be monitored through functions:

```
CJ_WORD CJ_MaxMainTime(void)
CJ_WORD CJ_MinMainTime(void)
CJ_WORD CJ_RunMainTime(void)
CJ_BYTE CJ_MaxInterruptTime(void)
CJ_BYTE CJ_MinInterruptTime(void)
CJ_BYTE CJ_RunInterruptTime(void)
```

For more information, refer to SoHAV C Programming.Language User Guide.

5.15 Setting the Execution Order of a Program

The execution order of an application program defines the order in which the entities are processed. SoHVAC calculates the execution order when entities are linked. This order is based on the logical flow of the application program.

In some cases there is a need to modify the execution order for an application program to have a proper operation. This happens especially when loops are programmed in a graphical environment.

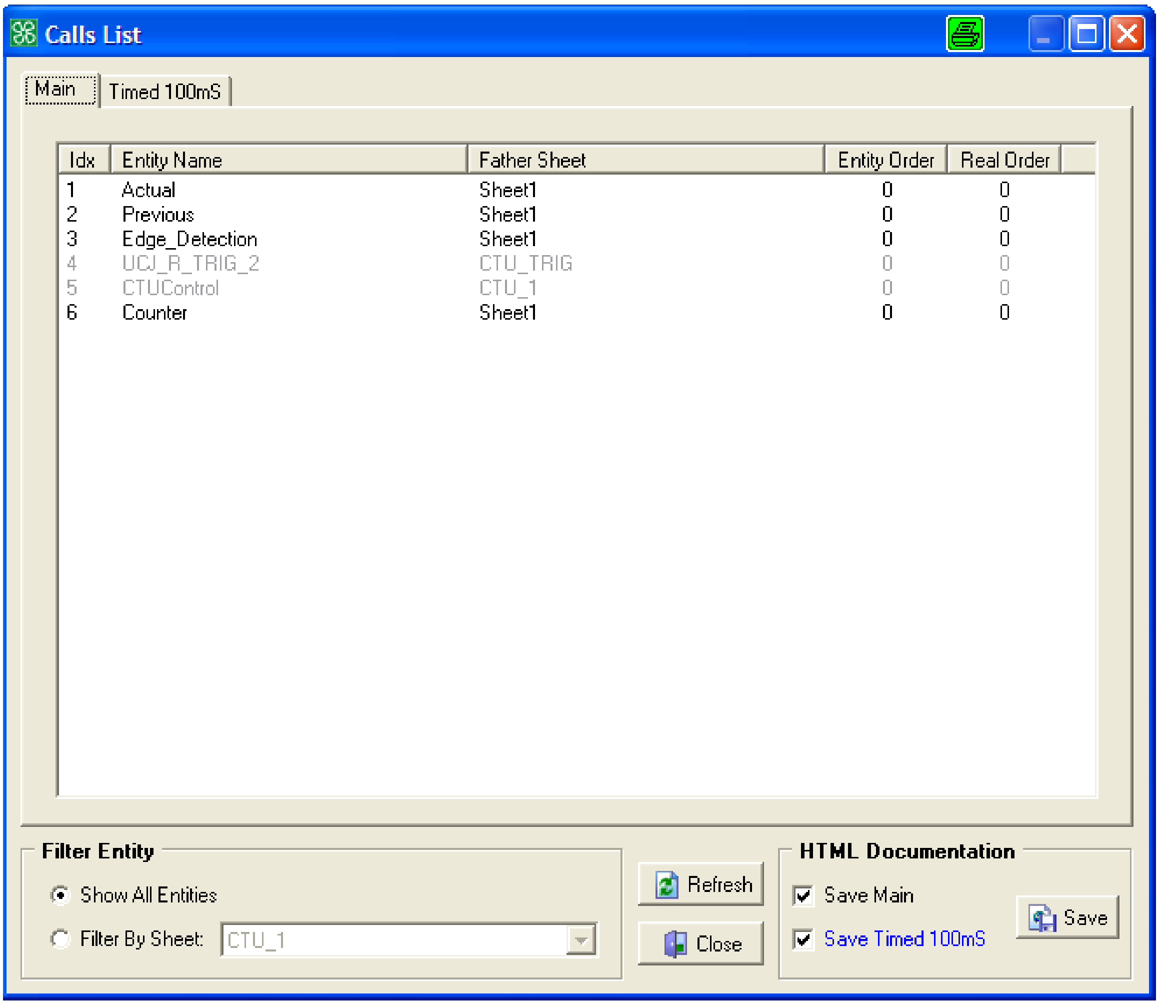
The execution order can be adjusted using 2 properties:

- Property order of an entity object
- Property offset of a sheet

The execution order is based on the real order which is the sum of the entity order and sheet offset. Entities with the lowest real order are executed first. Entities with the highest real order are executed last.

The execution order can be displayed and modified using the **Calls List** from the menu **Tools→Calls List…**

The **Calls List** shows only entities which require processing. Entities which have only default values do not appear in the list.



Calls List

Main | Timed 100mS

Idx	Entity Name	Father Sheet	Entity Order	Real Order
1	Actual	Sheet1	0	0
2	Previous	Sheet1	0	0
3	Edge_Detection	Sheet1	0	0
4	UCJ_R_TRIG_2	CTU_TRIG	0	0
5	CTUControl	CTU_1	0	0
6	Counter	Sheet1	0	0

Filter Entity

● Show All Entities

○ Filter By Sheet: CTU_1

Refresh

Close

HTML Documentation

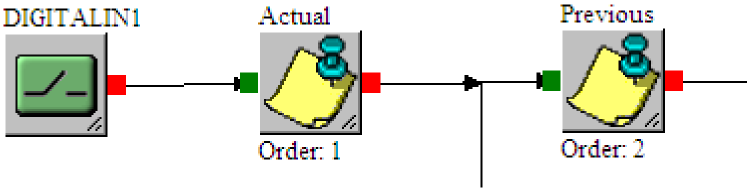
☑ Save Main

☑ Save Timed 100mS

Save

NOTE: Entities which are grayed are not accessible. They are used in library type function blocks.

Once the **Calls List** window is opened, the order is displayed each entity. The figure shows that the variable Actual is executed first followed by the variable Previous.

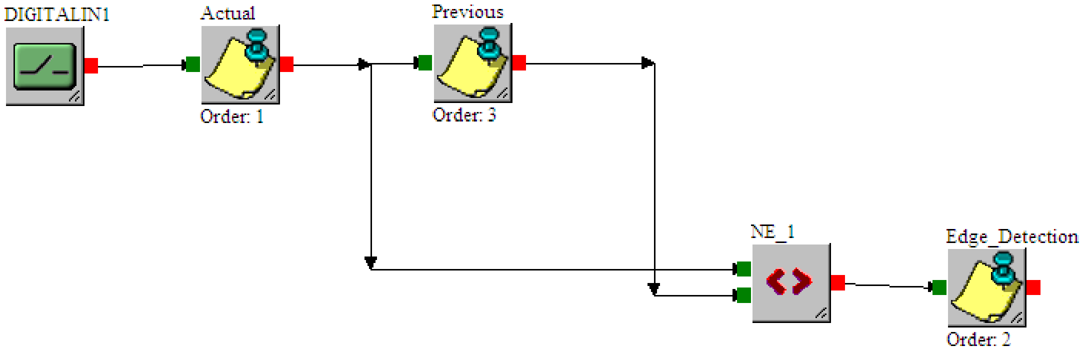


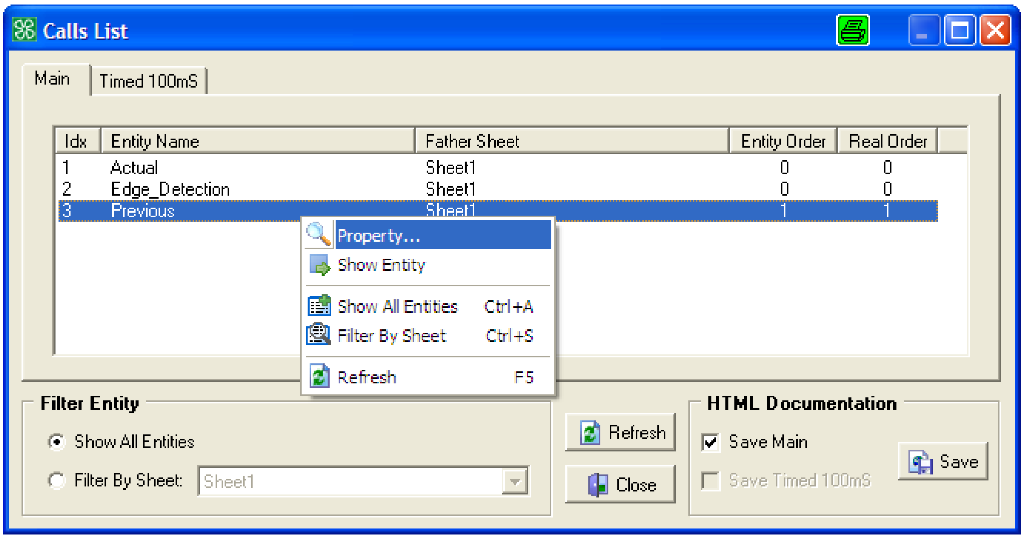
The following example demonstrates where changing the execution order can be useful.

The application program monitors the digital input DIGITALIN1 and is indicated by the variable Edge_Detection when the input DIGITALIN1 has changed state.

The function compares the actual state of the digital input with the previous state of the digital input. In case the states are different, the variable Edge_Detection is set to **TRUE**.

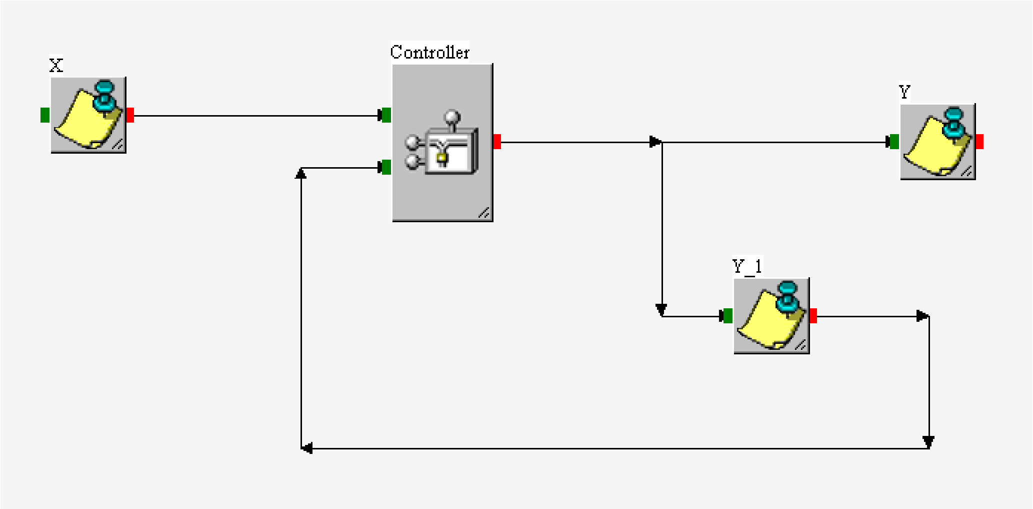
To execute this function, the variable Previous needs to be updated with the actual value after the comparison. To achieve the execution order, you need to increase the entity order property of the entity Previous. In this sample the entity order is set to 1.





Idx	Entity Name	Father Sheet	Entity Order	Real Order
1	Actual	Sheet1	0	0
2	Edge_Detection	Sheet1	0	0
3	Previous	Sheet1	1	1

5.15.1 Order Property



In this case, the output Y needs to be calculated by the algorithm using input Y_1 (which represents the previous output) before the input Y_1 is updated.

To assign a specific order to the execution of this part of the program, you need to set the right sequence to the values of the **Order** property of all the involved variables.

In the example above, the correct operation is established 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

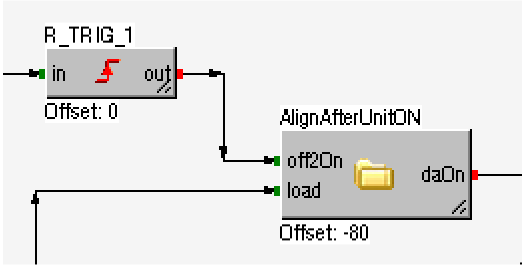
This means when executing the program, variable X is calculated first, then Y and finally Y_1.

The entities affecting the execution order of a program are all those that have the property order, that is Alg., Var, Par, Pers, DigitalOut, AnalogOut, LED, Buzzer, and CommandOut.

5.15.2 Offset Property

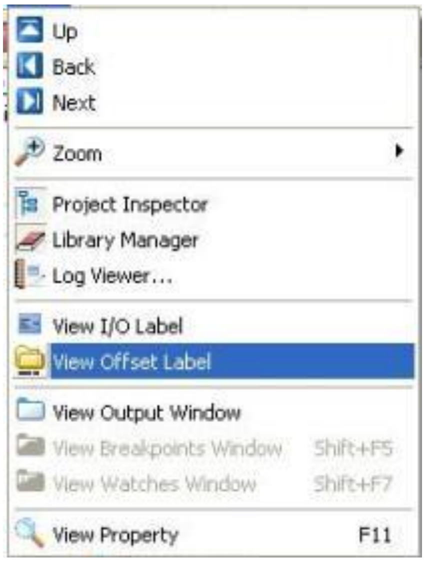
Operating on the property **Offset** on a subsheet, it is possible to move the execution order for all the entities it contains.

The **Offset** properties under the Sheet and libraries can be displayed.



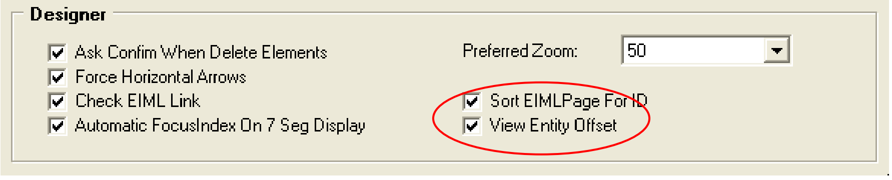
To activate this function go to:

1) Menu **View→View Offset Label**



Or

2) From **Tools→Preference→General** sheet



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

6 Controller Operation

6.1 Cycle Time IO Management

Physical Input and Output Management differs if they are managed in the Main task or the 100ms Task.

An application is split into a number of execution blocks depending on the number of instructions contained in your program. The following diagram assumes that your application has been split into 5 separate execution blocks (referred to as ALGOs in the chart), which is done automatically by the firmware of the controller.

⓪ At the beginning of the main cycle, physical inputs are copied into their assigned logical memory locations, referred to as Input Process.

① Each execution block of the main cycle uses the values captured during Input Process.

② At the end of the processing of each execution block, outputs are written to logical outputs, but not to the physical outputs.

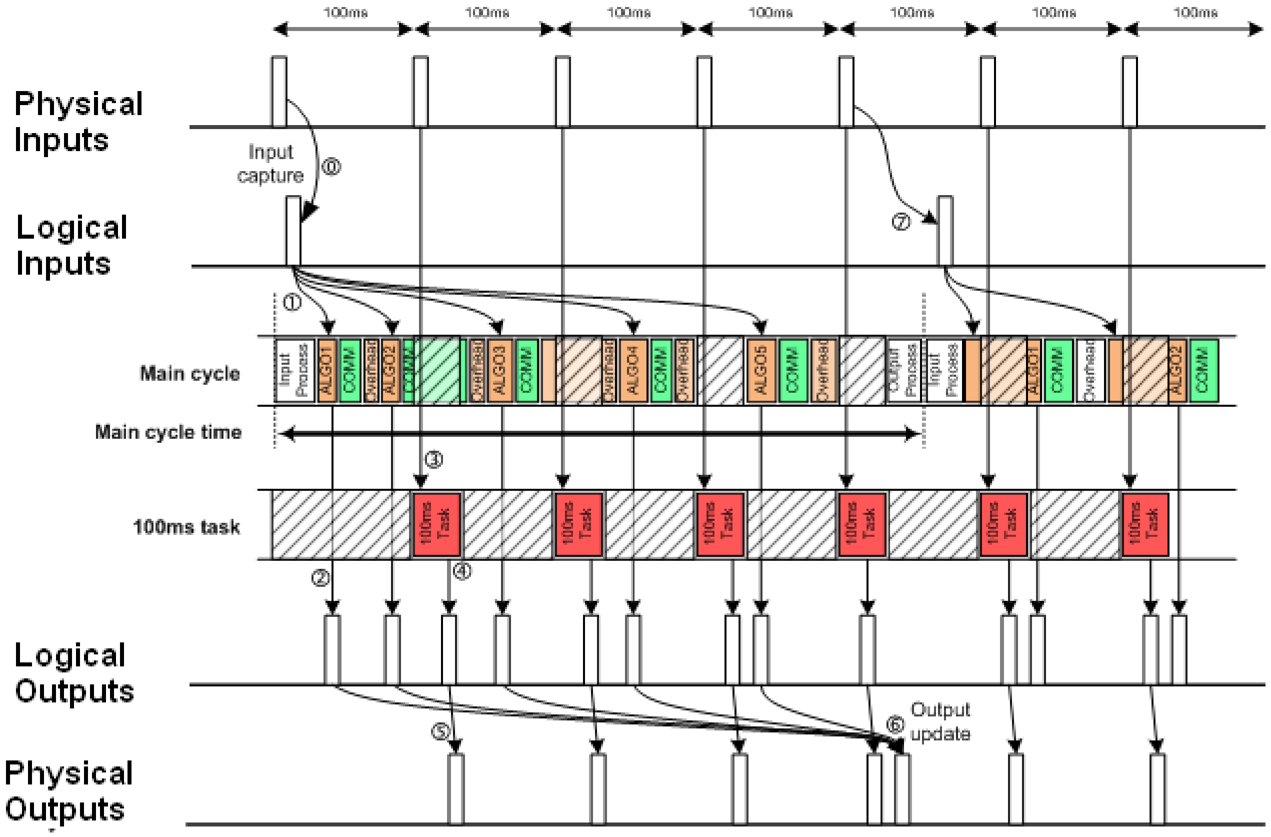
③ When the 100 ms task begins execution, the main cycle is interrupted and the physical inputs used within the 100 ms task are read directly into the variables used in the task. However, this does not affect the value of the same logical variable used in the main cycle.

④ At the end of 100 ms task, the value of the logical outputs used in the 100 ms task are updated.

⑤ After the update of the logical outputs, the assigned physical outputs used in the 100 ms task are also updated.

⑥ At the end of main task, during the output process, the physical outputs used in the main cycle are assigned their logical value as a result of the processing of the execution blocks.

⑦ The processing of the main cycle restarts ⓪.



NOTE: The main cycle is split into many execution blocks to improve the performance and response of the CPU for the internal processing and communication tasks. The number of execution blocks (ALGO blocks) depends on the complexity of the program. During the execution of the ALGO block, the controller can buffer up to 10 ExpBUS requests and 6 responses. It can also buffer up to 10 Modbus requests.

6.2 Tasks

The user writes application in several sheets by placing functions blocks on it, and selects operations to be executed in the main cycle task or the 100 ms task.

The system creates 2 calls list, one for the main cycle and the other one for the 100ms task.

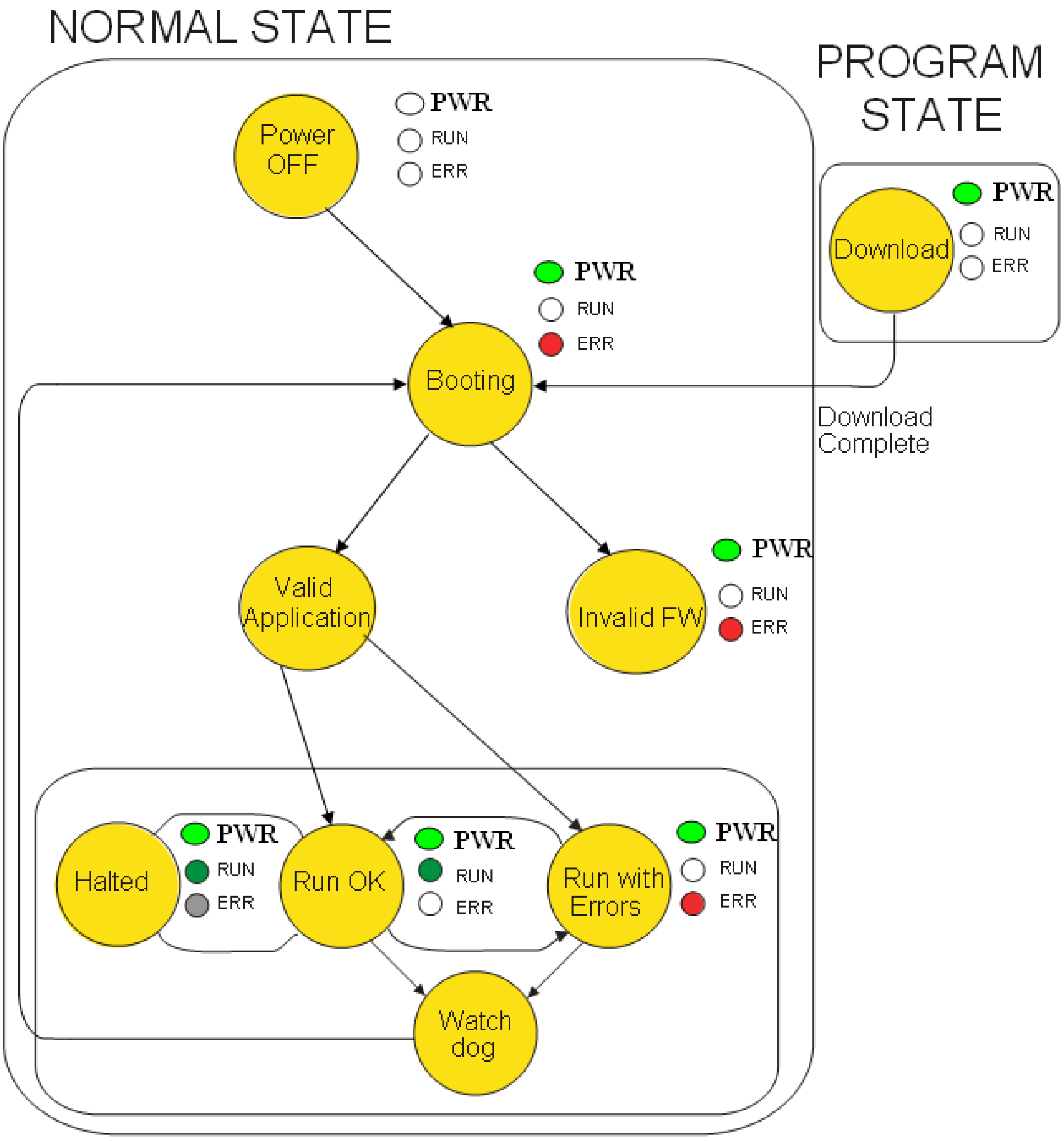
The user can edit this list to modify the execution order of operations.

The lists are split into several execution blocks. The number of operations to execute in a block can be changed by the user in the **Project→Property...→Optimizations** tab.

6.3 State Machine

The controller state machine is shown in the following diagram:

Diagram State:



6.3.1 Power OFF

Controller not powered.

6.3.2 Program State

The controller receives a new program through Programming cable.

In this state all outputs are OFF.

6.3.3 BOOT State

The controller verifies the compatibility with the hardware. If the verification is OK, the controller starts, else the controller goes in Invalid FW state where only a firmware upgrade can be performed.

6.3.4 Valid Application

Valid Application is the state after a booting. The application waits to start, auto-test is performed and the data context is reinitialized.

6.3.5 RUN State

Application is processed, and all IOs are managed. Errors like RTC LOW VOLTAGE and EEPROM error are signalled without stopping the application. These situations have to be managed by the application.

Without errors detected, the controller is in **RUN OK** mode

With errors detected, the controller is in **RUN with ERRORS** mode, the **ERR LED** is flashing.

6.3.6 Watchdog and Applicative Errors

This state is reached when a task watchdog elapses.

The controller passes into the BOOTING state.

6.3.7 HALTED (only in DEBUG MODE)

The application is valid but the application task is stopped on breakpoint.

Outputs are frozen in their current state (PWM still pulsing).

Event task can be processed, and its IOs are applied.

RUN LED is slowly flashing, instead of being ON.

6.3.8 Power On State

A power on reinitializes the controller context, except the parameter data, and persistent data that were saved.

All intermediate internal data are reset.

All the ongoing network transactions are reset.

Communication stacks are reinitialized.

Embedded IOs are reconfigured.

IO Modules are reconfigured.

Error conditions are reset.

Forced IOs are un-forced.

The values of the retain data are restored, after computing of their checksum.

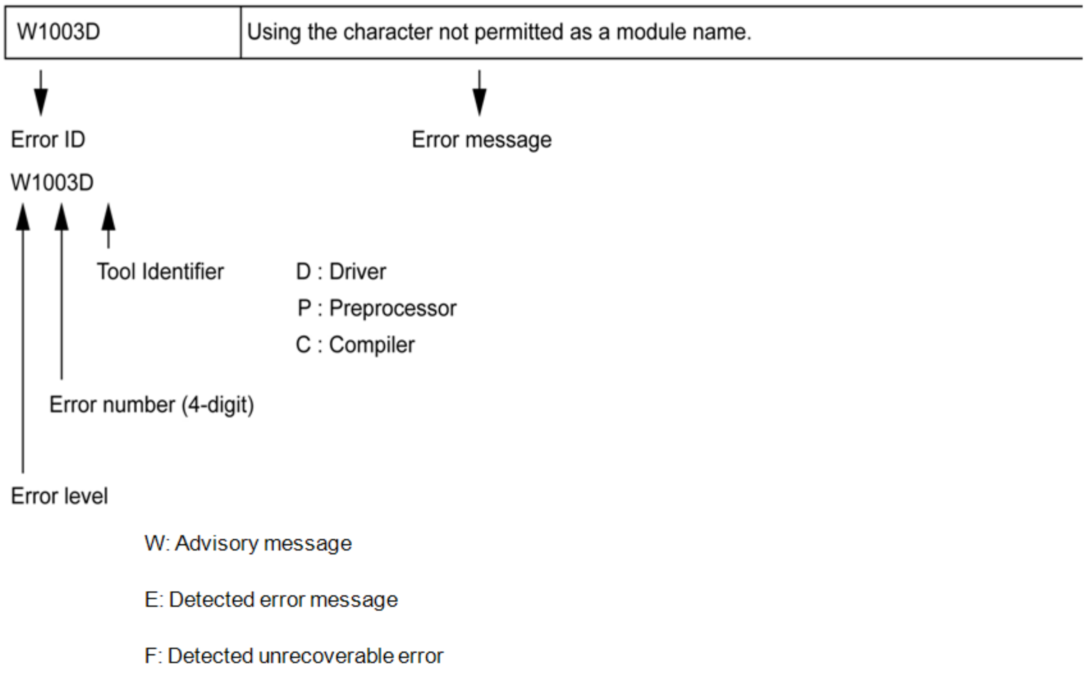
If the checksum is wrong, they are reinitialized in their default value (Cold start).

6.3.9 Watchdog Behavior

When a watchdog occurs, the controller is automatically reset.

Appendix - Compilation Error Message Format

The figure shows the compiler error message format:



Glossary

A

algo

Algorithm

algorithm

A calculation method that produces a control output by operating on an error signal or a time series of error signals.

ANSI-C

Standard version of the C program language, as directed by the American National Standards Institute.

ASCII

American Standard Code for Information Interchange is a communication protocol for representing alphanumeric characters (letters, numbers, and certain graphic, and control characters).

B

BACnet

Building Automation and Control Network

Building Automation and Control Network (BACnet) Protocol

A BMCS communication protocol developed by the American Society of Heating, Refrigerating, and Air Conditioning Engineers (ASHRAE).

Building Control System (BCS)

A system that controls the comfort and safety of building assets and environment.

Building Management and Control System (BMCS)

An integrated BMS and BCS

Building Management System (BMS)

A system which centralizes the monitoring, operations, and the management of a building to achieve more efficient operations.

BOOL

Boolean type.

This is the basic data type in computing. A `BOOL` variable can have either of the following 2 values: 0 (`FALSE`) or 1 (`TRUE`).

A bit extracted from a word is of type `BOOL`.

BYTE

When 8 bits are grouped together, they are called a `BYTE`. You can enter a `BYTE` either in binary mode or in base 8.

The `BYTE` type is encoded in an 8 bit format which, in hexadecimal format, ranges from `00`...`FF` hex.

bottom-up approach

Project development method that starts from the machines basic, low level elements and then building more and more complex blocks, one brick at a time until the final goal is reached.

C

C++

An extension of the C program language which enables object-oriented programming.

CondVisible

Conditional Visibility

controller

A *controller* (or ‘programmable logic controller,” or “programmable controller”) is used to automate industrial processes.

D

DI

Digital input

DO

Digital output

DWORD

Double Word.

The `DWORD` type is encoded in a 32 bit format.

This table shows the upper/lower limits of each of the base that can be used:

Base	**Lower limit**	**Upper limit**
Hexadecimal	16#0	16#FFFFFFFF hex
Octal	8#0	8#37777777777
Binary	2#0	2#11111111111111111111111111111111

Examples of Representation:

Data	Representation in one of the bases
00000000000010101101110011011110	16#ADCDE
00000000000000010000000000000000	8#200000
00000000000010101011110011011110	2#10101011110011011110

E

EIML

Embedded Interface Markup Language

Ename

External name

entity

An element of the SoHVAC development environment that features at least one input or one output represented by a data type. An application program can be created by combining several entities.

ExpBUS

An electronic communication bus between expansion modules and a CPU.

expansion I/O modules

A digital or an analog module that adds additional I/O to the base controller.

F

Firmware

A type of software stored in the memory and is usually read-only. Within SoHVAC, it represents the operating system executed by the controller.

I

Iname

Internal name

L

LED

Light emitting diode.

It is an indicator that lights up when electricity passes through it. It can be used to indicate the operational status of a communications electronic module.

M

Modbus

Communication protocol allowing communication between many devices connected to the same network.

N

node

An addressable device on a communication network.

NTC

Negative temperature coefficient

P

palette

A portable platform used for storing or moving goods.

PAR

Parameter

PERS

Persistent

persistent data

Value of persistent data is used during next application change or cold start. It is re-initialized only at a reboot of the controller or reset origin. The value is maintained after a download.

PTC

Positive temperature coefficient

PT100/PT1000

Platinum resistance thermometer are characterized by their nominal resistance R0 at a temperature of 0° C.

- Pt100 (R0 = 100 Ohm)
- Pt1000 (R0 = 1 kOhm)

R

Real time clock

An option that keeps the time for a limited amount of time even when the controller is not powered.

S

software

Generic term used to indicate all non-tangible components of an information system, such as the programs and the data being processed.

SoHVAC

Graphic development environment which enables the creation of controller programs in an assisted and simplified way.

T

top-down approach

Project development method that defines the system layout first and then implementing the details.

V

VAR

Variable

Variable

Memory entity of type `BOOL`, `WORD`, `DWORD`, and so on, whose contents can be modified by the program currently running.

W

WORD

The type `WORD` is encoded in a 16 bit format and is used to perform processing on series of bits.

This table shows the upper/lower limits of each of the bases that can be used:

Base	Lower limit	Upper limit
Hexadecimal	16#0	16#FFFF
Octal	8#0	8#177777
Binary	2#0	2#1111111111111111

Examples of representation:

Data	Representation in one of the bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Index

1

139BCAN Config, *160*

A

About
- Simulating EIML Pages, *143*

About SoHVAC, *10*
Advanced Operations, *125*
Algorithm, *83*
Algorithm Category, *175*
Analog Input
- Hardware, *87*

Analog Inputs, *99*
- Physical Terminals, *38*

Analog Output
- Hardware, *89*

Analog Outputs
- Physical Terminals, *39*

Array Inputs
- C Algorithm Editor, *177*

Array Inputs and Array Output
- C Algorithm Editor, *178*

Arrows, *133*

B

Basic Elements Data Types, *72*
Basic Operations, *17*
Break
- Debugger, *55*

Breakpoints, *58*
Built-in Editor
- C Algorithm Editor, *176*

Button
- Hardware, *91*

Buttons
- Physical Terminals, *42*

Buzzer
- Hardware, *93*

Buzzers
- Physical Terminals, *45*

C

Changing the Order of Input Terminals
- C Algorithm Editor, *179*

CJ_ANALOG, *102*
CJ_BTN, *102*
CJ_BUZZ, *101*
CJ_CMD, *102*
CJ_DATE, *101*
CJ_DATE_STRUCT, *103*
CJ_DATE_TIME_STRUCT, *104*
CJ_DATETIME, *101*
CJ_LED, *101*
CJ_TIME, *101*
CJ_TIME_STRUCT, *103*
CJ_VOID, *100*
Clock
- Hardware, *94*

Clocks
- Physical Terminals, *40*

COM Port Initialization, *48*
Combo
- EIML Pages, *118*

Command In, *81*
- Physical Terminals, *43*

Command Out, *81*
- Physical Terminals, *44*

Commands, *183*
Commands Window, *184*
Commissioning on Expansion Bus (ExpBUS), *148*
Compiler, *132*
Compiling the Program, *46*
Conditional Visibility, *144*
Configuration of One Expansion and One Display, *150*
Configuring Data Exchanges between Controllers, *157*
Configuring Multiple Expansions, *153*
Configuring One Remote Display with Multiple Controllers, *159*
Connection to Physical Terminals, *36*
Considerations in Project Development, *135*
Control Components, *72*
Control sheets and Subsheets, *24*
Controller Operation, *192*
Copy to Clipboard
- EIML Pages, *143*

Create a Command, *187*
Creating a New Project, *17*
Creating a New Sheet, *24*
Creating a New Subsheet, *24*
Creating EIML Pages, *28*
Creating Libraries, *182*
Creating Network, *150*
Cut and Paste of a Entity with EIML Connection, *23*
Cycle time IO Management, *192*, *193*

D

Defining Execution Tasks, *187*
Designer, *131*
Developing projects with SoHVAC, *11*
Digital Input
- Hardware, *85*

Digital Inputs
- Physical Terminals, *36*

Digital Output
- Hardware, *86*

Digital Outputs
- Physical Terminals, *37*

Download
- Project Settings, *131*

Downloading the Program, *48*

E

EIML Pages Import and Export, *147*
Element Properties
- EIML Pages, *109*

Elements in EIML Pages, *107*
Embedding an Element in an EIML Page, *107*
Entities
  Definition and Use, *20*
ExpBUS Master…Window, *149*
ExpBUS Network, *98*
ExpBUS Node and Physical Address, 148
Exporting Variables, *162*
External Links, *25*

F

Factory Configuration, *149*
Firmware Function, *144*
Fix, *73*

G

General Information, *10*
General Tab
Glossary, *198*
GoTo
  Debugger, *55*
Graphics, *134*
Graphics Tab, *132*

H

Hardware, *84*
Hardware Bits, *95*
Historian, *126*

I

Icons
  EIML Pages, *116*
Input and Output Properties
  C Algorithm Editor, *174*
Items, *133*

J

Join Entities, *25*
Joining Entities, *25*

K

Key-in Present Value
  Debugger, *57*

L

LED
  Hardware, *92*
LEDs
  Physical Terminals, *41*
Line
  EIML Pages, *124*
Link In, *26*
Link Out, *26*

M

Moving Entities, *20*

N

Navigating through the Pages, *139*
Navigation in Sheets, *24*

O

OFFSET Property, *191*
Open File/Open Page List
  EIML Pages, *141*
ORDER Property, *190*

P

Page Properties
  EIML Pages, *108*
Par, *74*
Password-Protected Information, *140*, *141*
Pers, *75*
Preference Menu, *130*
Product Programming Tool
  Download Manager, *137*
Project Backup, *136*
Project Change Log, *129*
Project Defines, *127*
Project Options, *127*
Project Settings, *125*
Projects, *131*
Put an Entity on a Worksheet, *20*

R

Rectangle
  EIML Pages, *124*
Removing Entities, *21*
Removing/Changing Entities and Saving Links, *21*
Reset
  Debugger, *55*
Run
  Debugger, *55*

S

Save Image
  EIML Pages, *143*
Selecting Hardware, *29*
Set Value
  Debugger, *57*
Setting the Execution Order of a Program, *188*
Sheets, *132*
Show Link Context Menu, *26*
Simple Data Types, *100*
Simulating EIML Pages, *141*
Software, *72*
Software Bits, *96*
Start Debug, *54*
State Machine, *194*
Strings
  EIML Pages, *114*
Structured Data Types, *102*
Subsheet, *82*
System Command Description, *185*

T

Table
  EIML Pages, *120*
Tasks, *193*
Text
  EIML Pages, *109*
The Screen Organization, *19*
Timer, *79*
Timer Display, *81*
Toolbar
  EIML Pages, *107*
Type CJ_CHAR for String Management, *172*
Type Compatibility Table, *105*

U

Use of the Debugger, *53*
Using Array Entities, *169*
Using C Algorithm Editor, *174*

V

Var, *77*
Variable
  EIML Pages, *111*
Version Info, *125*
View ParamsDrv
  EIML Pages, *142*
View Vars
  EIML Pages, *143*