

MET-TP 3rd Course

# A Guide to HIQUEL PLC

"It is possible to fly without motors, but not without knowledge and skill."



ALMAWARED  
ENGINEERING  
& TRADING SAE



1<sup>st</sup> edition 11/2011

© Almawared Engineering and Trading (MET) SAE, Cairo, Egypt

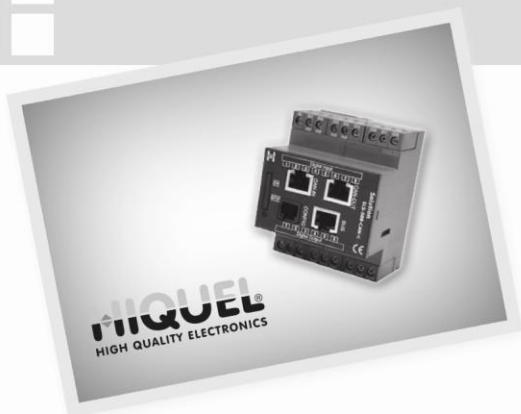
All rights reserved, including those of the translation.  
No part of this manual maybe reproduced in any form,  
processed or distributed by means of electronic systems  
without written permission of MET SAE

This material is subject to changing without notice.

*MET-TP 3rd Course*

# A Guide to HIQUEL PLC

"It is possible to fly without motors, but not without knowledge and skill."



ALMAWARED  
ENGINEERING  
& TRADING SAE







<b>Table of Contents</b>	<b>Introduction</b>	<b>6</b>
	<b>Company Overview</b>	<b>10</b>
	<b>SLS-500 Configurator Setup</b>	<b>14</b>
	<b>SLS-500 Configurator Menus</b>	<b>24</b>
	<b>CONFIG Menu</b>	<b>28</b>
	<b>Project Menu</b>	<b>32</b>
	<b>Page Menu</b>	<b>34</b>
	<b>Line</b>	<b>43</b>
	<b>Group Menu</b>	<b>45</b>
	<b>Flow Menu</b>	<b>47</b>
	Flow: Constants	48
	Flow: Special flags	51
	Flow: Memories	54
	Flow: Bit Handling	63
	Flow: Analog Handling	69
	Flow: Counter	79
	Flow: Conversion	84
	Flow: Tables + Interpolation	88
	Flow: State	95
	Flow: Comment	99
	Flow: System	99
	<b>I/O Menu</b>	<b>105</b>
	I/O: Digital Inputs	105
	I/O: Digital Outputs	106
	I/O: Analog Inputs	107
	I/O: Analog Outputs	108
	I/O: Potentiometer	109
	<b>Objects Menu</b>	<b>111</b>
	Objects: Timer Relays	111
	Objects: Real-time clock	116

Objects: Terminal	124
Objects: Memory Card	134
Objects: Closed Loop Control	138
<b>Debug Menu</b>	<b>141</b>
Debug: Add Symbol	141
Debug: Add Monitor	142
Debug: Write to Symbol	143
Debug: Show System Information	143
<b>Run Menu</b>	<b>145</b>
Run: Compile	145
Run: Simulate	145
Run: Download + Run	151
Run: Start PLC	151
Run: Stop PLC	151
Run: Erase PLC	151
Run: Show	152
<b>Applications and Exercises</b>	<b>158</b>

# 1. Introduction





**Introduction**

Welcome to the third course in the **MET-TP** series. In order to excel in this course and get the best advantage of it, you should be aware of classic control and how it works, so we highly recommend taking the first course in this training series.

Al-Mawared Engineering and Trading Training Program offers a training service so that you can plan over the time the growth of the device knowledge, from the frequency inverters to the soft starters for asynchronous motors up to the PLC “Programmable Logic Controller”; Touch screens and SCADA system. Courses are targeted to engineers, technicians, users and installers and to the service personal as well. MET suggests courses that are mainly oriented to the use of the drives and of the automation system.

Our highly trained engineers will guide you step by step through each training course, allowing you to perform each step by yourself from small examples to large applications to help you practice everything you learn during the training course.

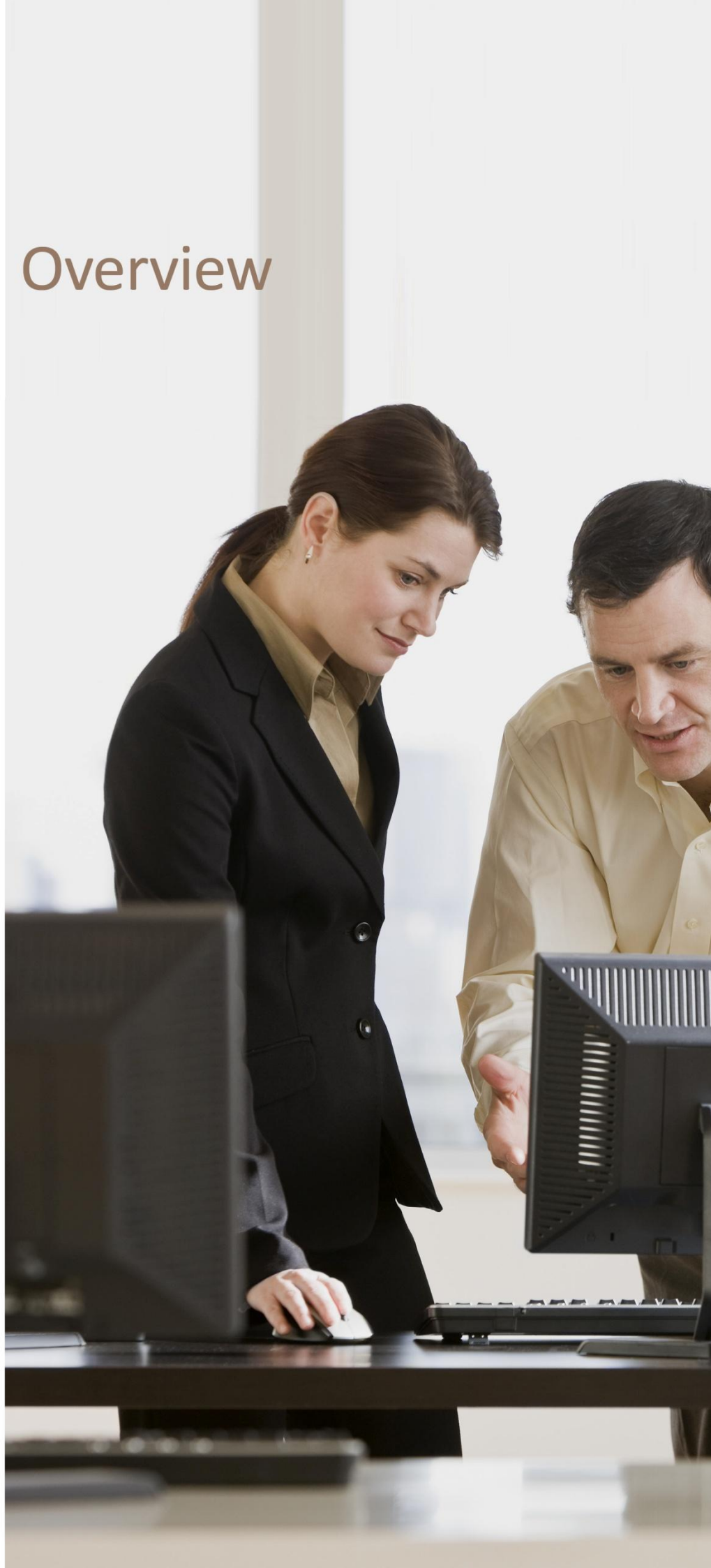
This course covers **HIQUEL PLC**. Upon completion of this course you will be able to:

- Open existing projects in HIQUEL Configurator.
- Develop new software or modify an old one.
- Download any software.
- Implement all the basic and advanced functions needed in software to build a fully functioned application.





## 2. Company Overview





**Experience and high responsiveness**

Year 2001 is not only a date for ALMAWARED ENGINEERING AND TRADING S.A.E (MET). It represents the starting of a company that is today specialized in the industrial automation, mechanical and electrical power transmission field, thanks to the entrepreneurial capabilities from foundation members Mr. Abdel Aziz Aboul Atta, Ms. Bahia Khairy, Eng. Mohamed Abdel Aziz, Eng. Khalid Abdel Aziz and Eng. Khalid Ateya, expertise and firm commitment of the promoting partners.

**MET** partners achieved the quality certificates, with the aim to grant a good quality system for the different market needs to satisfy most of their customers' needs by providing complete solutions or individual tailor-made solutions. Only a long experience allows a company to reach in a flexible and wide way to market demands, with a complete range of products and services.

**Flexibility and rapidity of product range**

Our structure is composed of real problem solvers who study the customer requirements and orient him towards simple and innovative solutions thanks to the structure of MET product range that gives our highly professional technical engineers a wide area of solutions. MET product range comprises a whole variety of automation, electrical and mechanical equipment such as PLCs, touch screens, SCADA systems, flow meters, density meters, BMS components, inverters, soft starters, AC/DC motors, Servo systems, gearboxes, clutches and brakes. Distributed Control System (DCS) is the latest product that MET provides to our customers.

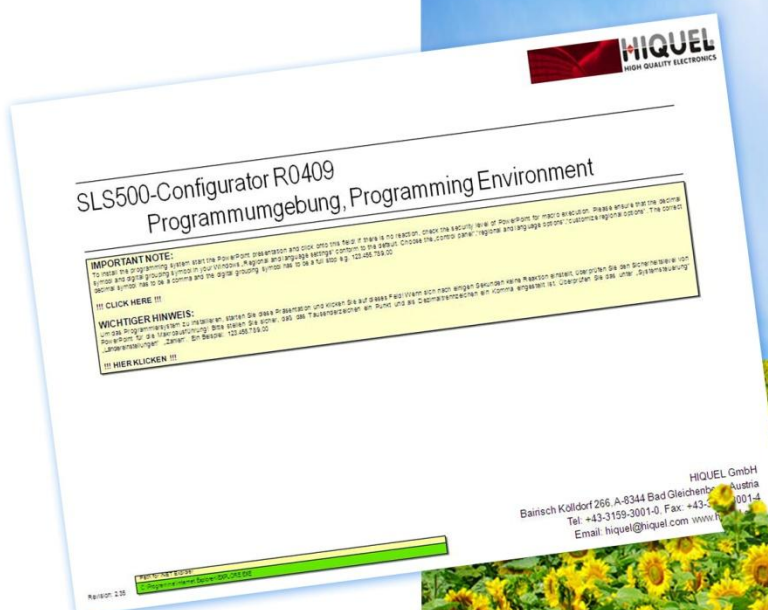
**A strong presence in the market**

Our experienced sales and marketing team is highly technical and customer-oriented. They combine premium customer support with years of industry experience to develop and sustain long-term relationships.

Our team ensures each customer is treated with professionalism and integrity as we work hard to understand our customers' needs and to provide solutions to meet those needs, so we serve a wide variety of market fields such as soup and fats, sugar industries, petroleum and refineries, cement factories, iron and steel mills, pharmaceuticals and cosmetics, textiles and dyeing, paints and chemicals, plastics and petrochemicals, food & beverages industries, paper and printing, packing and wrapping, pump & water treatment plants, sewage & water treatment plants and commercial HVAC.



# 3. SLS-500 Configurator Setup





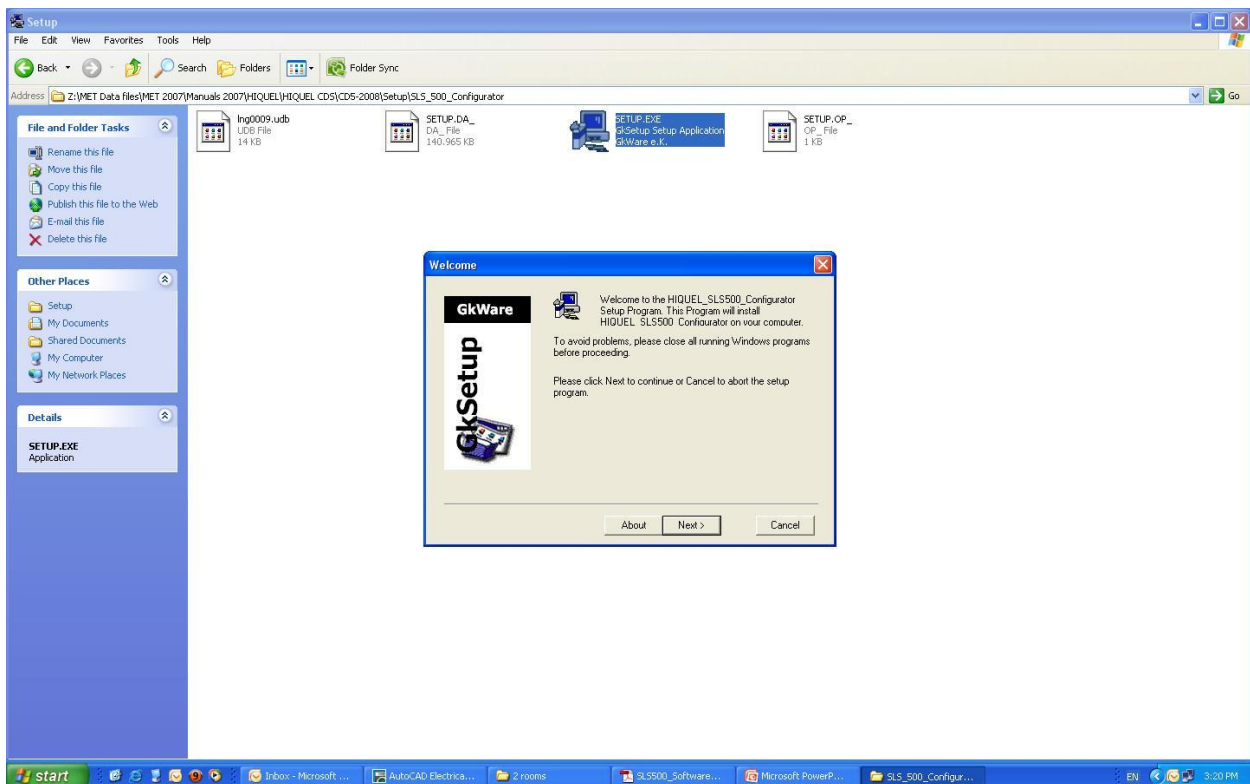


## SLS-500 Configurator Setup

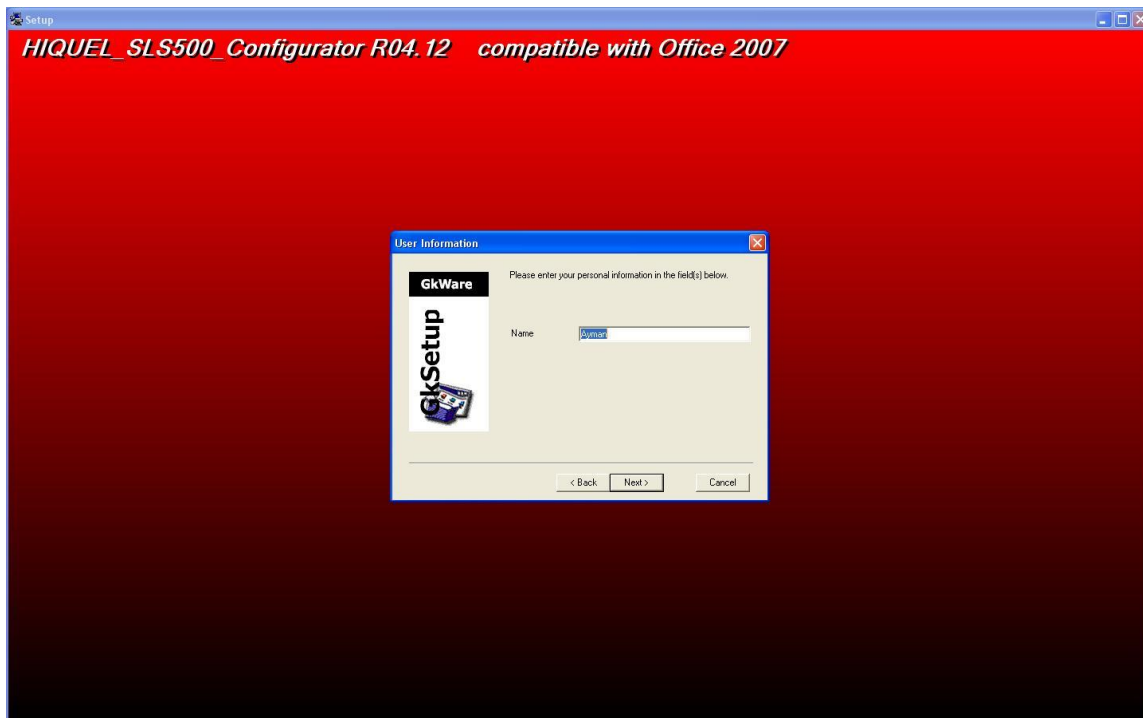
In this chapter, we will learn how to setup the software of HIQUEL PLC SLS-500 series (SLS-500 CONFIGURATOR).

Before we start the setup, the PC must have Microsoft Office PowerPoint installed on it, so SLS- 500 CONFIGURATOR is installed under Microsoft PowerPoint and it is preferred to be version 2003.

The 1<sup>st</sup> step is running setup as shown in the figure below:



When you press [Next], the next window will ask you to enter your personal information [Name]:



When you press [Next], the next window will ask you for the directory where the software will be installed, the default path is C:\HIQUEL\SLS500\_Configurator:



The next window will ask you about the language to be used inside the software, the default language is Deutsch (German) so it is preferred to change it to English to be easier for most users here in Egypt.



The next window will ask you about the name of the folder to be added to the program folders:



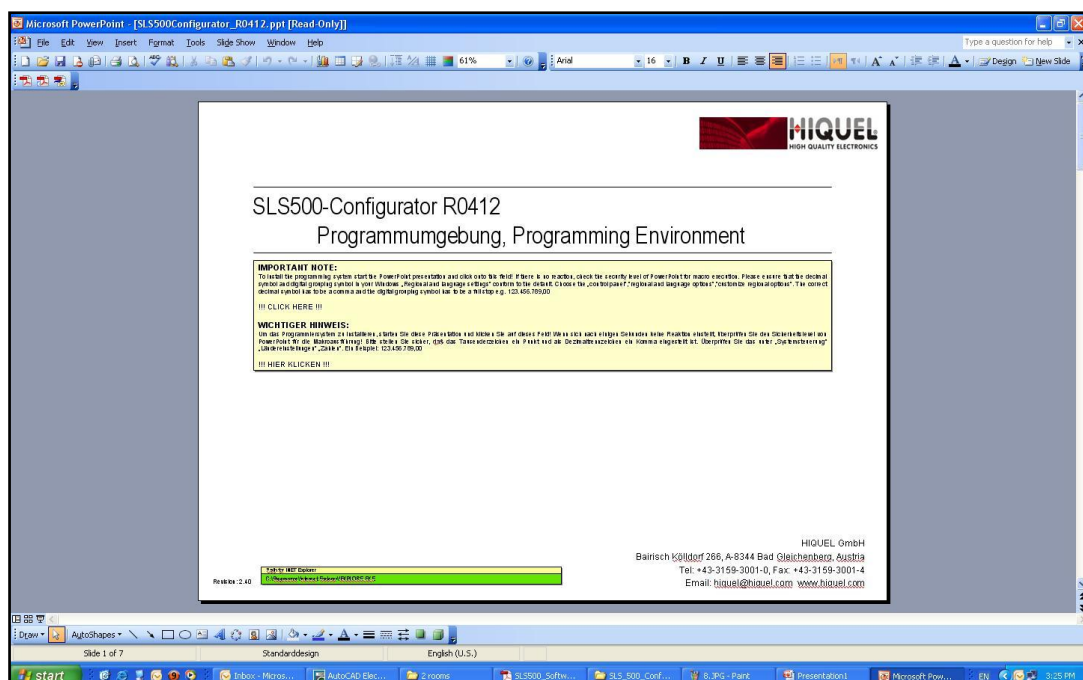
The next window will confirm with you the information to start the setup of the program so you should press [Next] if this information is right, otherwise press Back.



After you press [Next] the setup begins and after a while it will finish showing the following window:

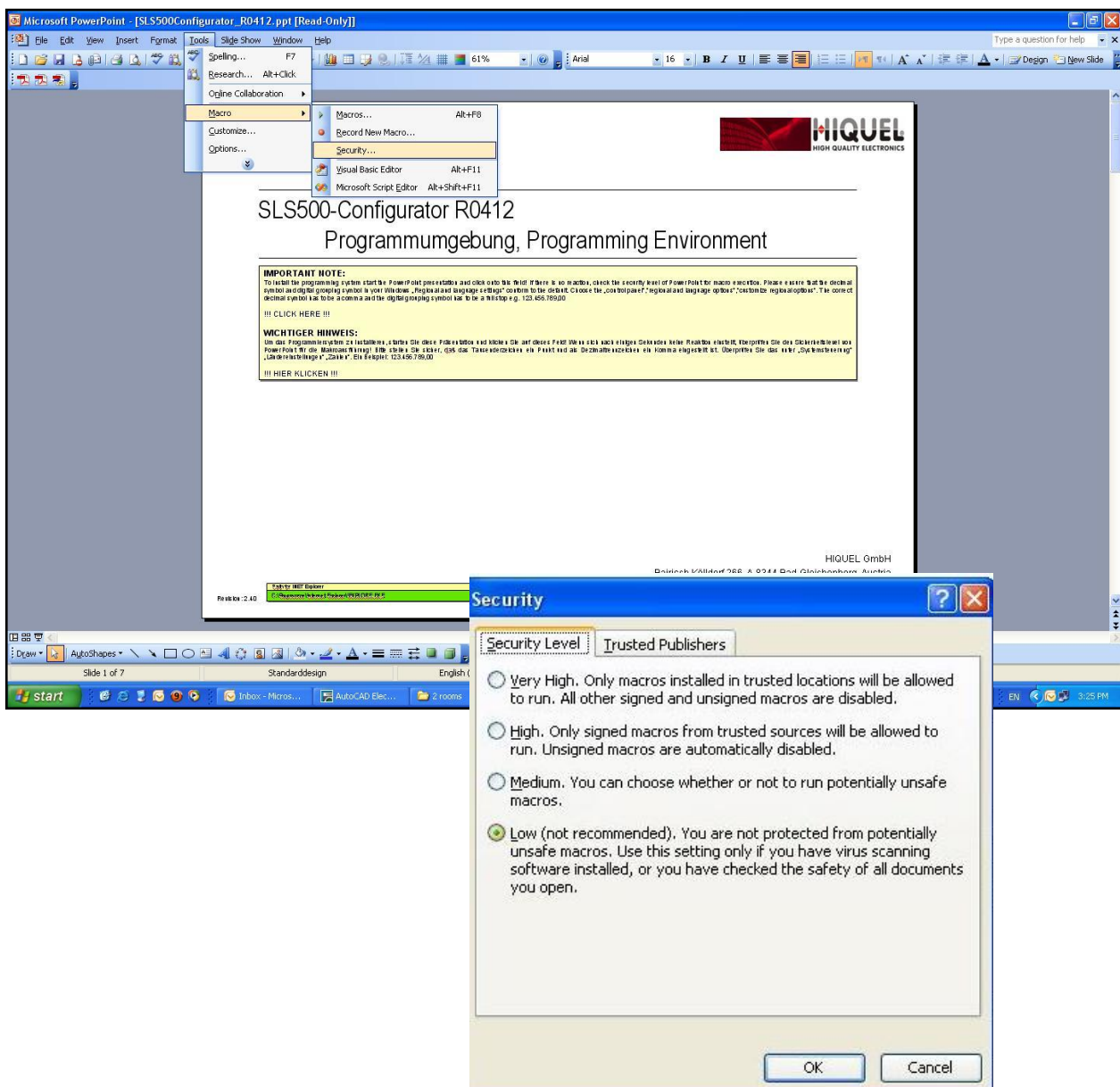


After you press [Finish] button, the setup finishes and you will find a power point shortcut icon on your desktop named "SLS500Configurator\_R0412". If you double click it, it will open a power point file as in the following figure:



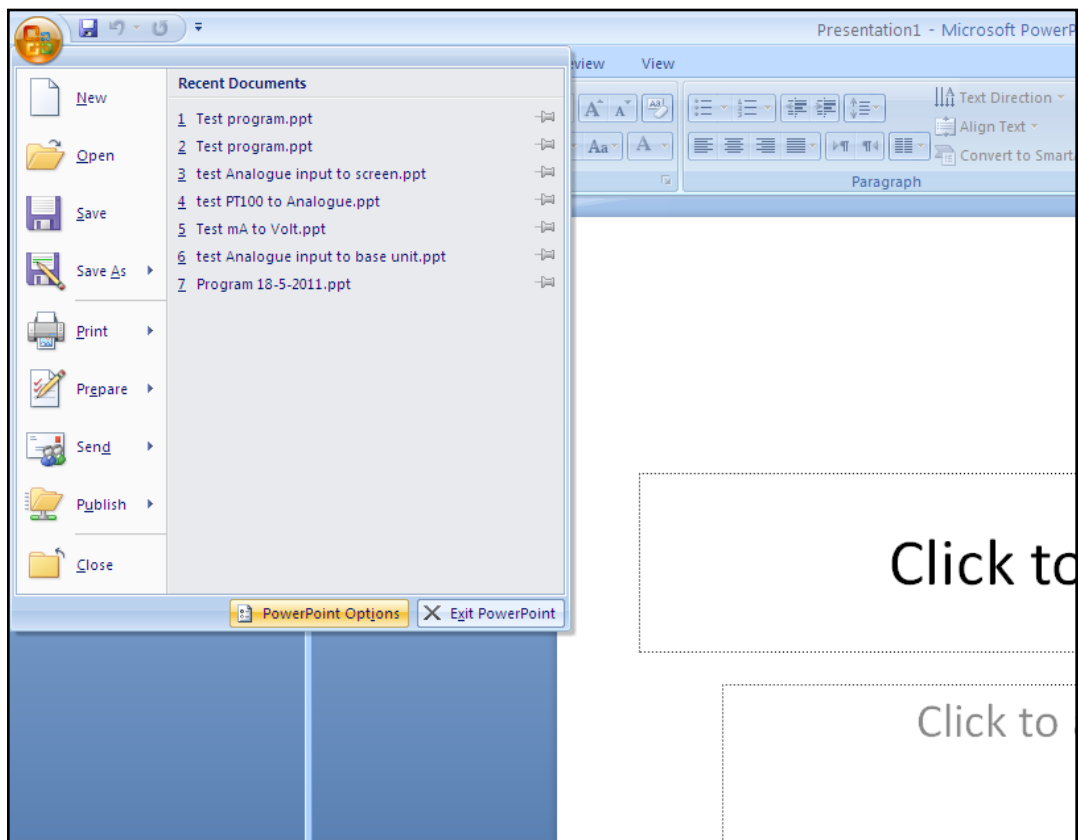
Before you start using the software, there are two steps you must do, the first step is to enable the macro inside power point, this is because the SLS-500-CONFIGURATOR is installed under Microsoft PowerPoint as a macro so you must enable it to let our software run correctly otherwise it will not run, and to do this at Microsoft office 2003, please follow the next instructions:

1. Open Microsoft PowerPoint program.
2. Open the Tools menu.
3. Choose Macro.
4. Choose Security from the sub menu (as shown in the figure).
5. In the Security window, choose "Security Level" tab, select Low then OK and close the program.



*Note:* To enable the macro at Microsoft office 2007, please follow the next instructions:

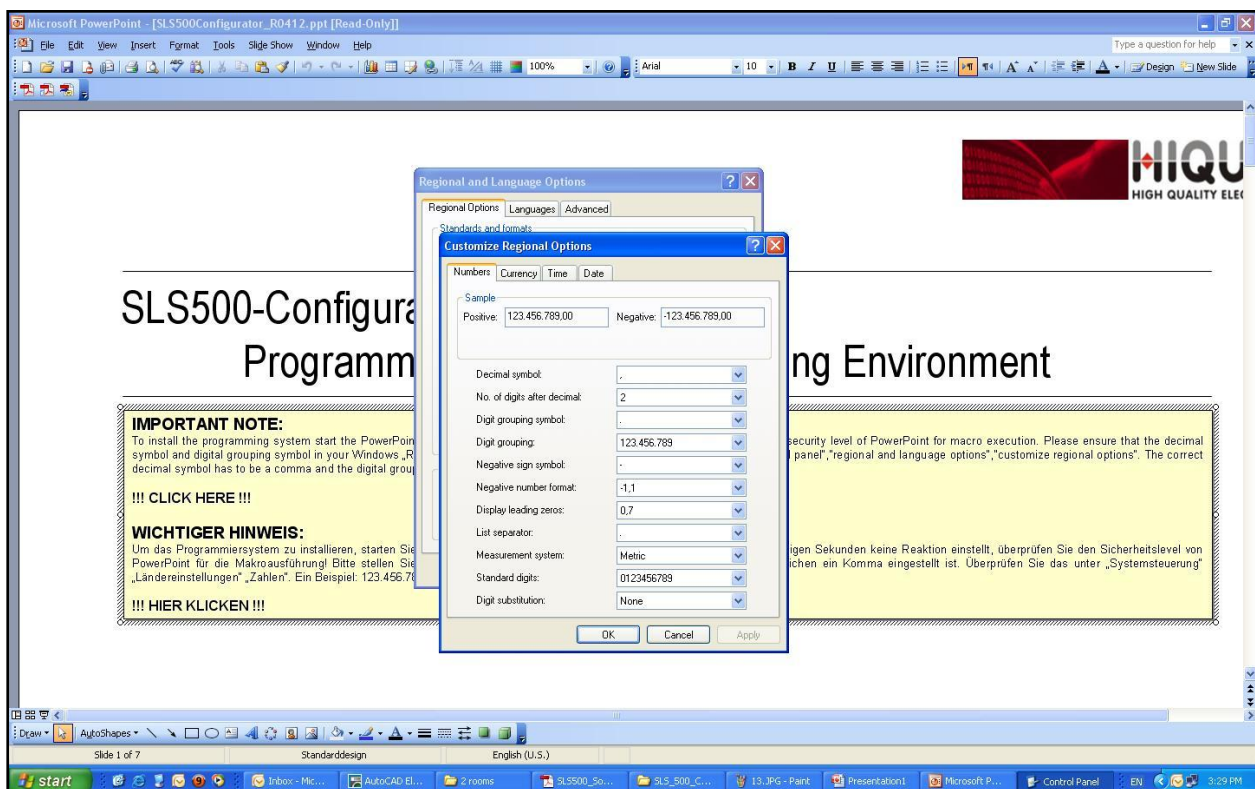
1. Open Microsoft PowerPoint program.
2. Click on the office button in the upper left corner, as in the figure.
3. Click on the PowerPoint Options button from the window (as shown also in the figure).
4. In the Security window, choose "Security Level" tab, select Low then OK and close the program.





The second step, please follow the next instructions:

1. In the control panel open Regional and Language Options.
2. In the Regional Options tab press on [Customize...] button.
3. In the [Customize Regional Options] window you should change the decimal symbol to (,) instead of (.) and the digit grouping symbol to (.) Instead of (,) as shown in the figure:



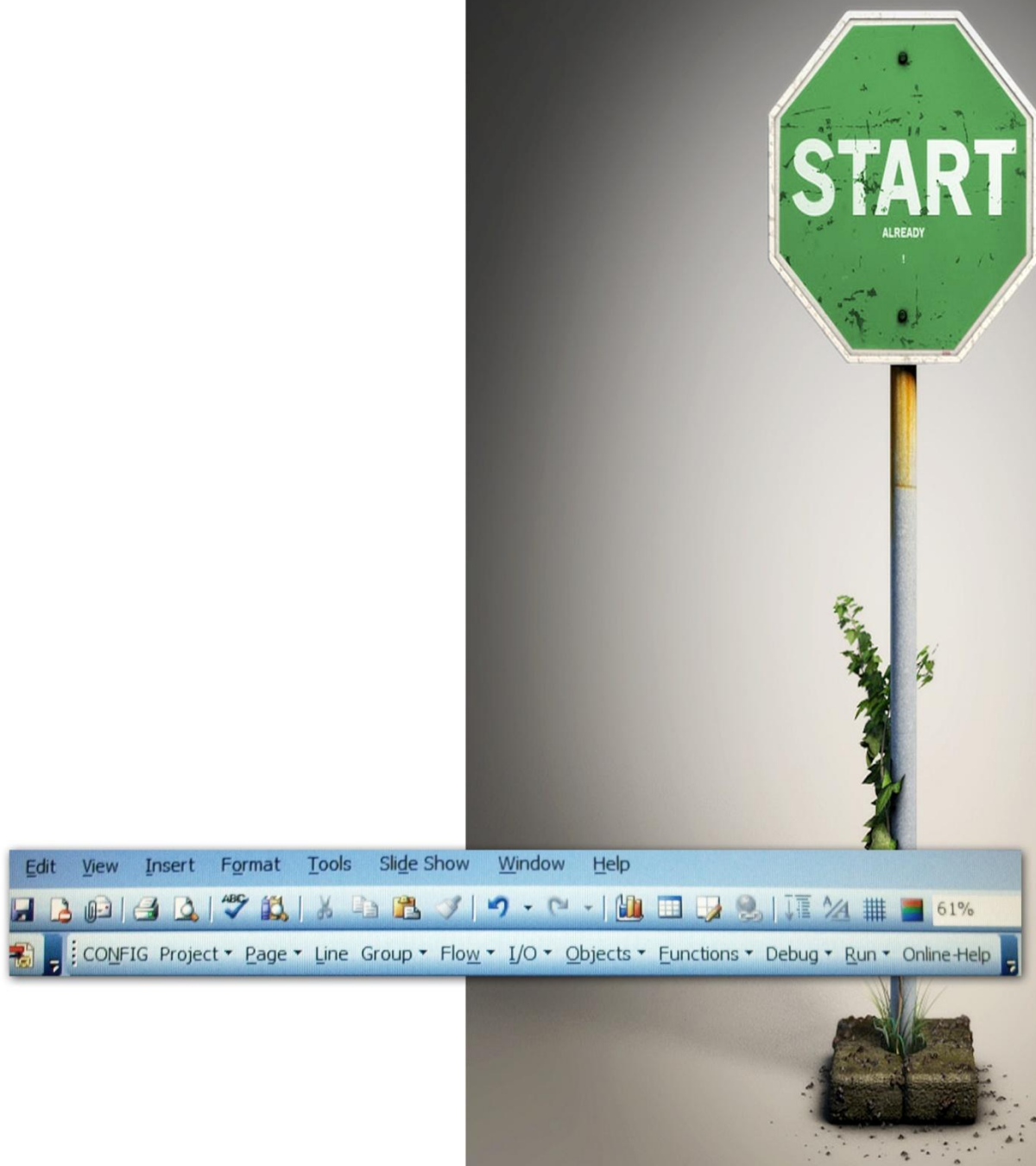
These two steps are done only before the first run of our software after installing it and there is no need to do it again.

After finishing these two steps our software is ready for using.

Now we will move to chapter 4 to know more about HIQUEL SLS-500 CONFIGURATOR.



## 4. SLS-500 Configurator Menus



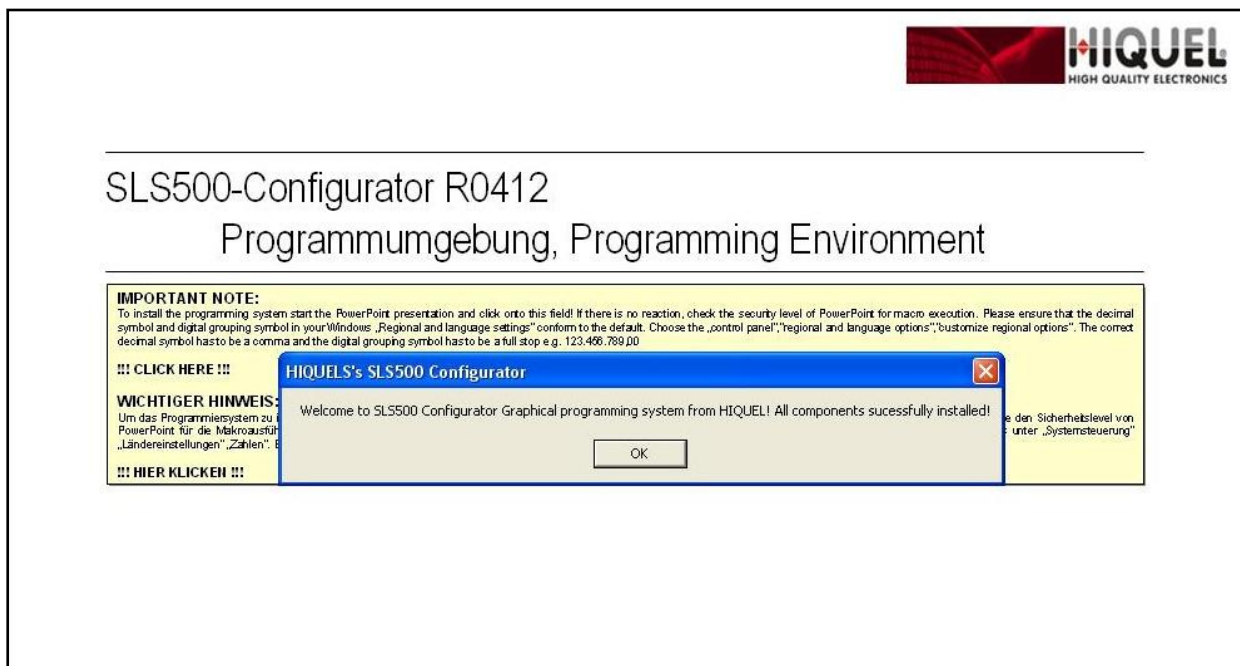


### SLS-500 Configurator Menus

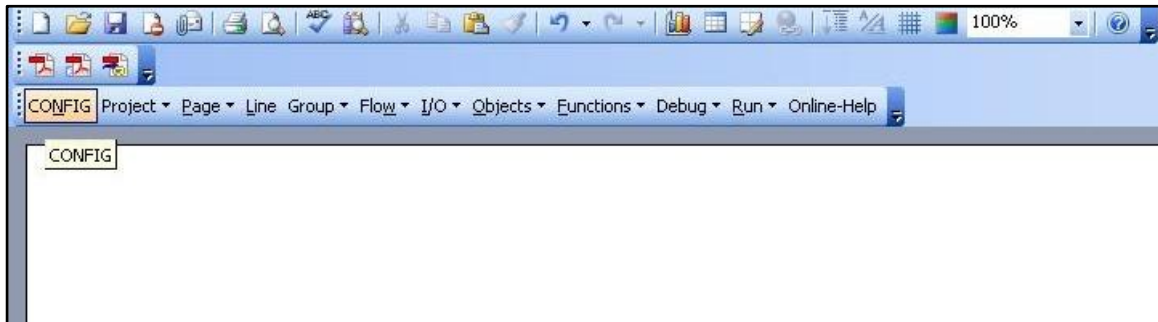
In this chapter we will learn about the software menus of HIQUEL PLC SLS-500 series (SLS-500 CONFIGURATOR).

Now we will open the power point icon named "SLS500Configurator\_R0412" then we should run the slide show to activate the macro and thus activate our software, by pressing F5 button on the keyboard then inside the yellow window shown in the 1<sup>st</sup> page, the following figure, we should click on the paragraph titled "IMPORTANT NOTE" which is written in English, then a window titled "HIQUEL's SLS500 Configurator" appears to inform us that the our program is installed successfully and ready for use.

After pressing OK, the window will disappear and the SLS-500 CONFIGURATOR is ready for use. This step should be done every time we start our program.



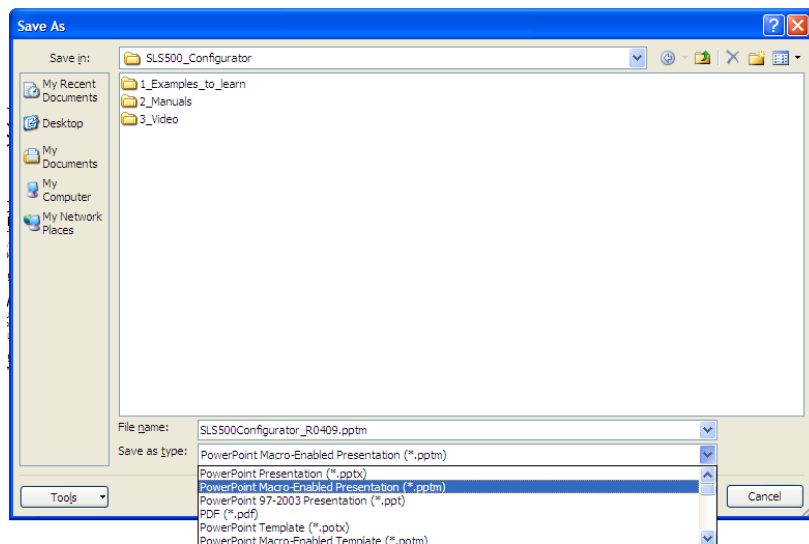
Now, we can see the menu bar of our program which consists of (CONFIG, Project, Page, Line, Group, Flow, I/O, Objects, Functions, Debug, Run and Online-Help) as shown in the figure.



The 1<sup>st</sup> step in our project is to save it in a new file by select “save as” to save the file with a new name instead of “SLS500Configurator\_R0412”.

**Note:**

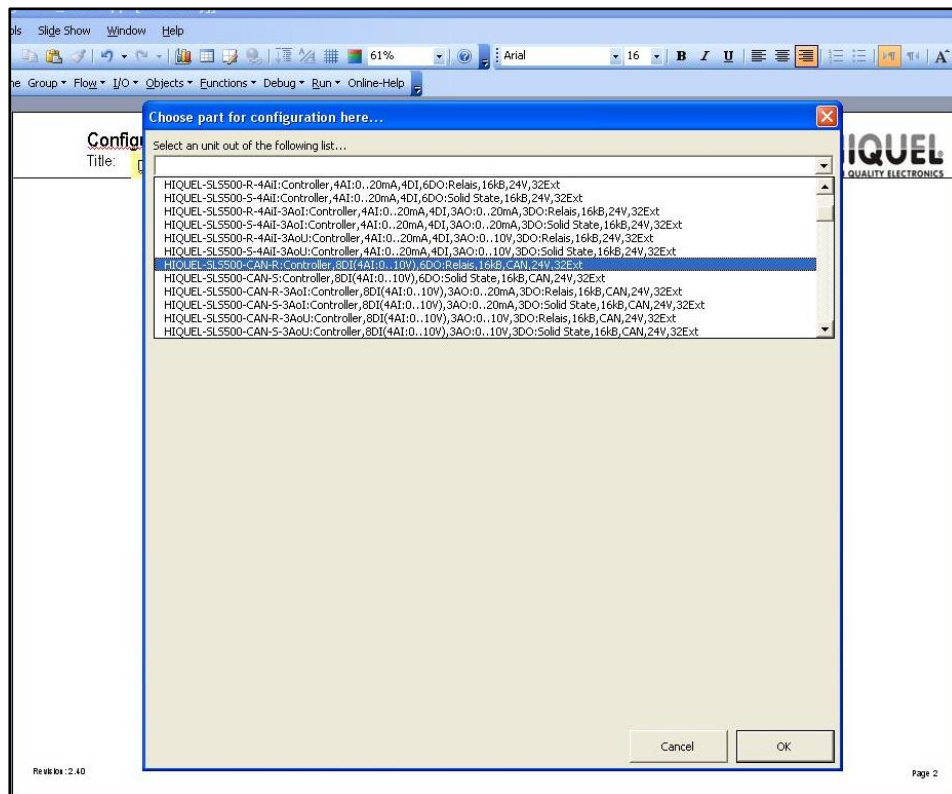
If you use Microsoft Office 2007, you should change the type of file you save, in the field “Save as type:” to be “PowerPoint Macro-Enabled Presentation (\*.pptm)”, as shown in the figure below:



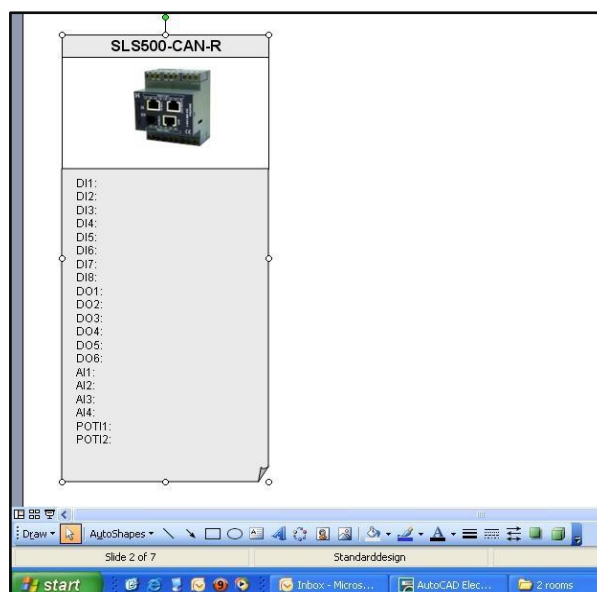


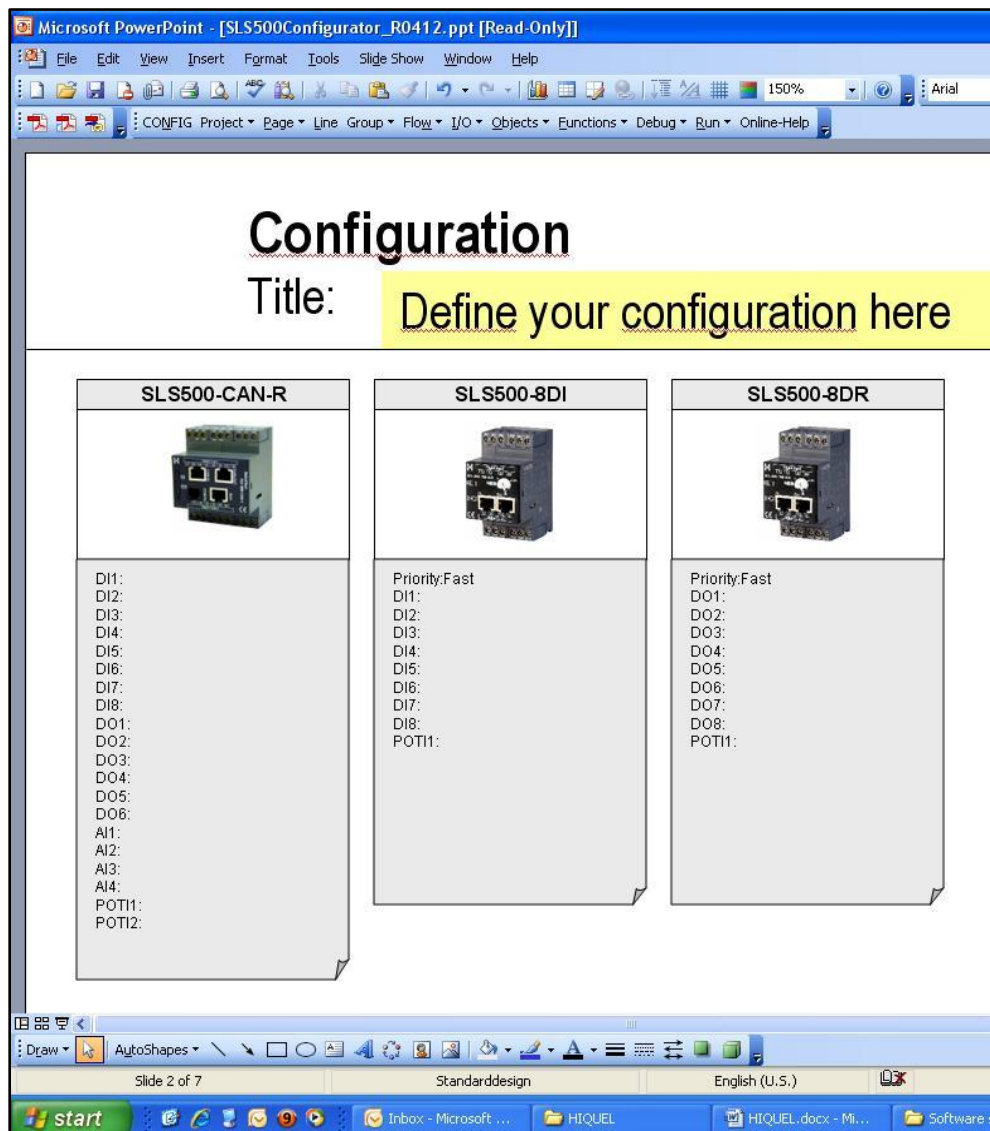
**CONFIG Menu**

When you press on it, it will open a window named [Choose part for configuration here...]. This window is to select the hardware configuration of your project, and it should be in the same order of your physical system in order to guarantee the correct function of your program.



If you need to add a new object, just click on CONFIGURATOR menu and select it from configuration window then press OK.





The above figure shows a system with 1 x SLS-500-CAN base module, 1 x 8 Digital input module and 1 x 8 Digital output module.

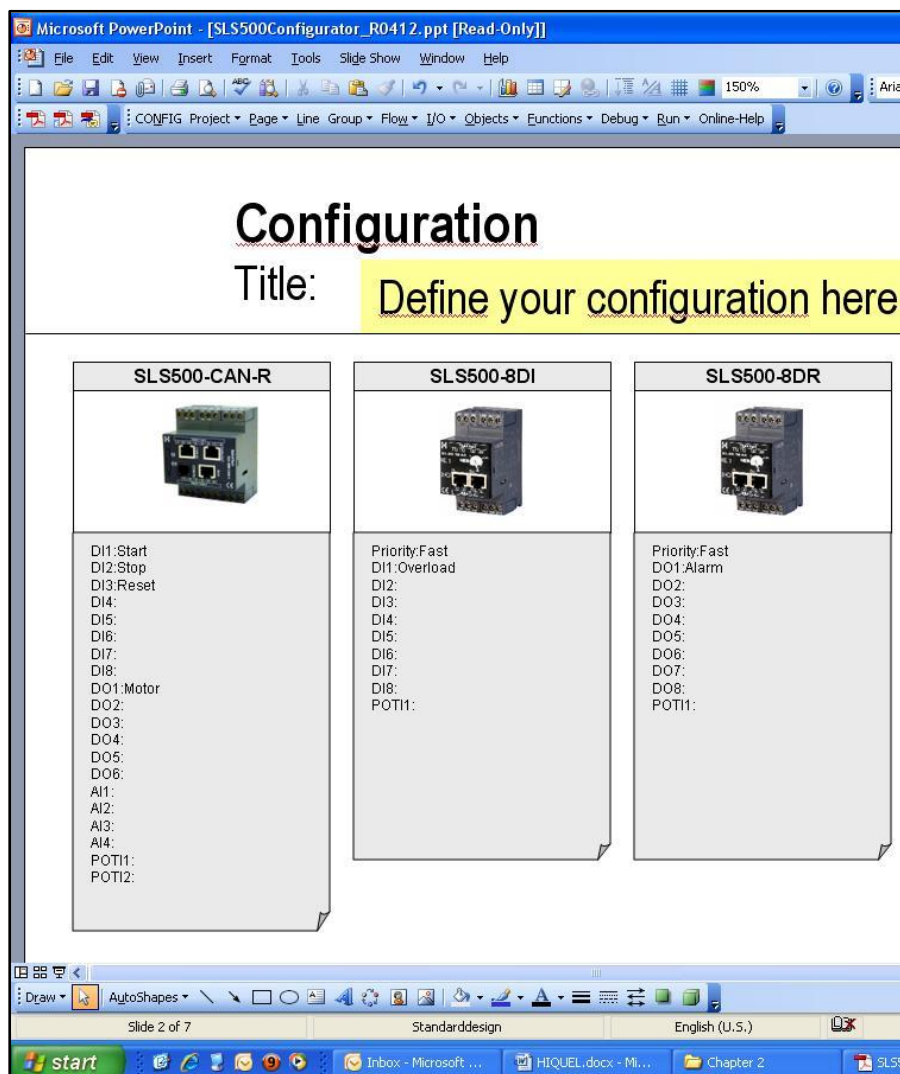
Every newly selected module will appear on the lower right of the page on right of the SLS-500-8DR extension module graphic. You must drag and drop the module into the position you require in order of priority within the program.

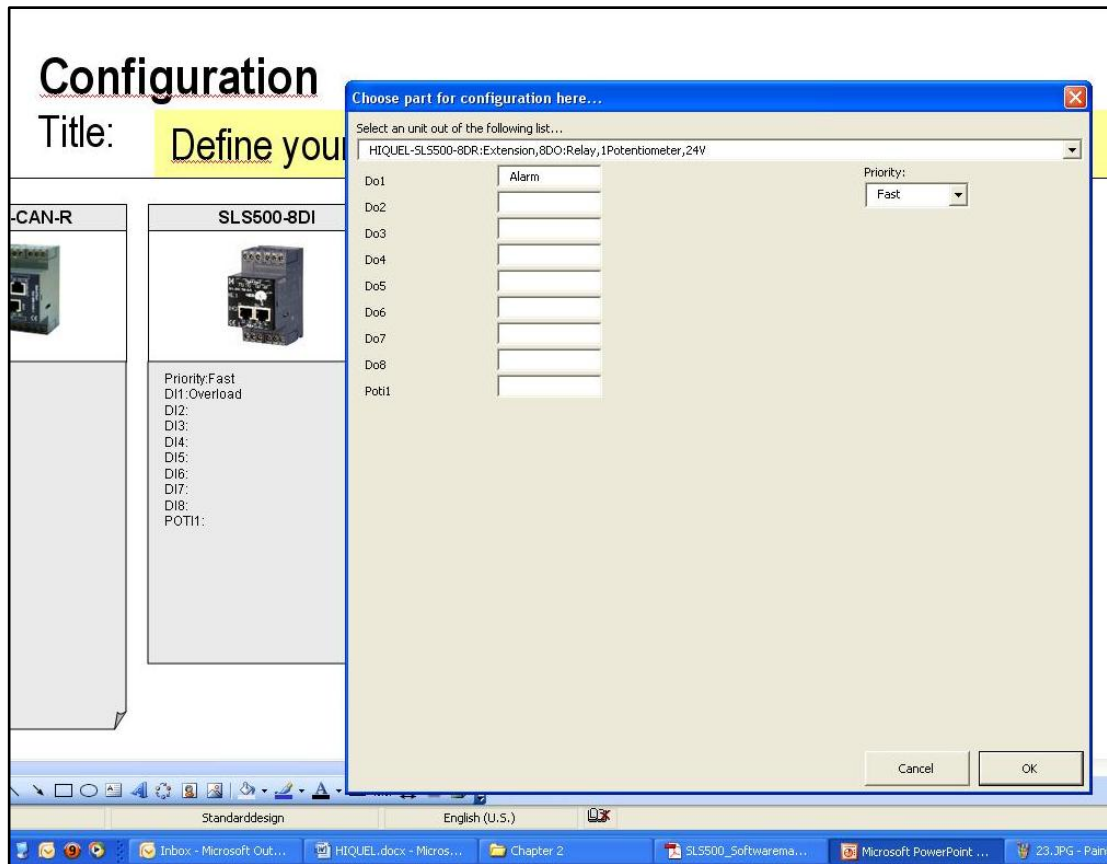
If you want to delete an object, select the desired module and delete it by pressing the [Delete] button on your keyboard.

**ADVICE:** The module will only be deleted in the configuration page. Any programmed object of the deleted module will not be deleted from your program! This will be detected when you attempt to compile your program. These objects must be deleted manually.

SLS-500-Configurator interprets the priority of the program objects from left to right and from top to bottom of the program page. The remote numbers are allocated exactly the same way. The base module has the definition [L1]. All expansion modules have the definition [Remote], beginning with a continuous numbering from 1. R1 is the first expansion; R2 is the second and so on.

If you add a module to the configuration, you can define a name for every digital or analog input and output, you can do this in the configuration page as shown in the figure, or when you select the object from “Choose part for configuration here...” window as shown in figure, when you use the input or output, the specified name will be displayed. This causes better understanding while programming.



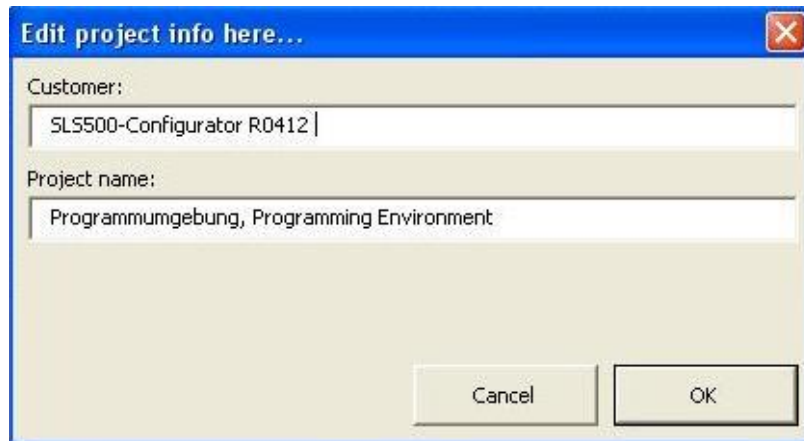


**Project Menu**

SLS-500-Configurator makes creating information and copies of projects easy, you have it all clearly on your start up page. Choose Project from the menu to get to all relevant program functions:

**Project: Info**

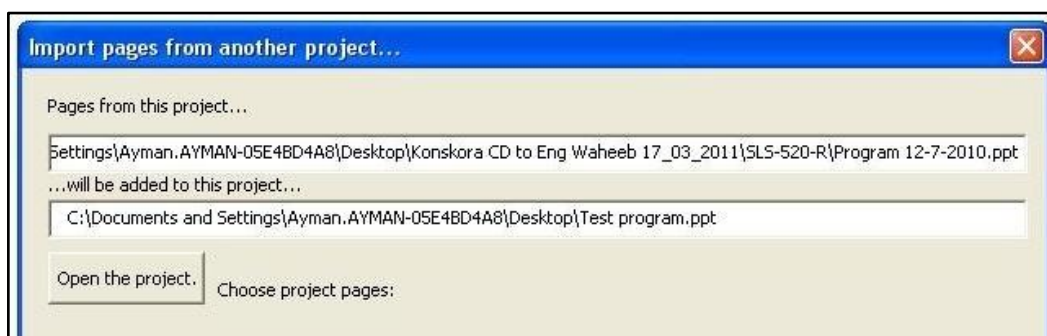
This window lets you edit the customer name and the project name at the starting page:

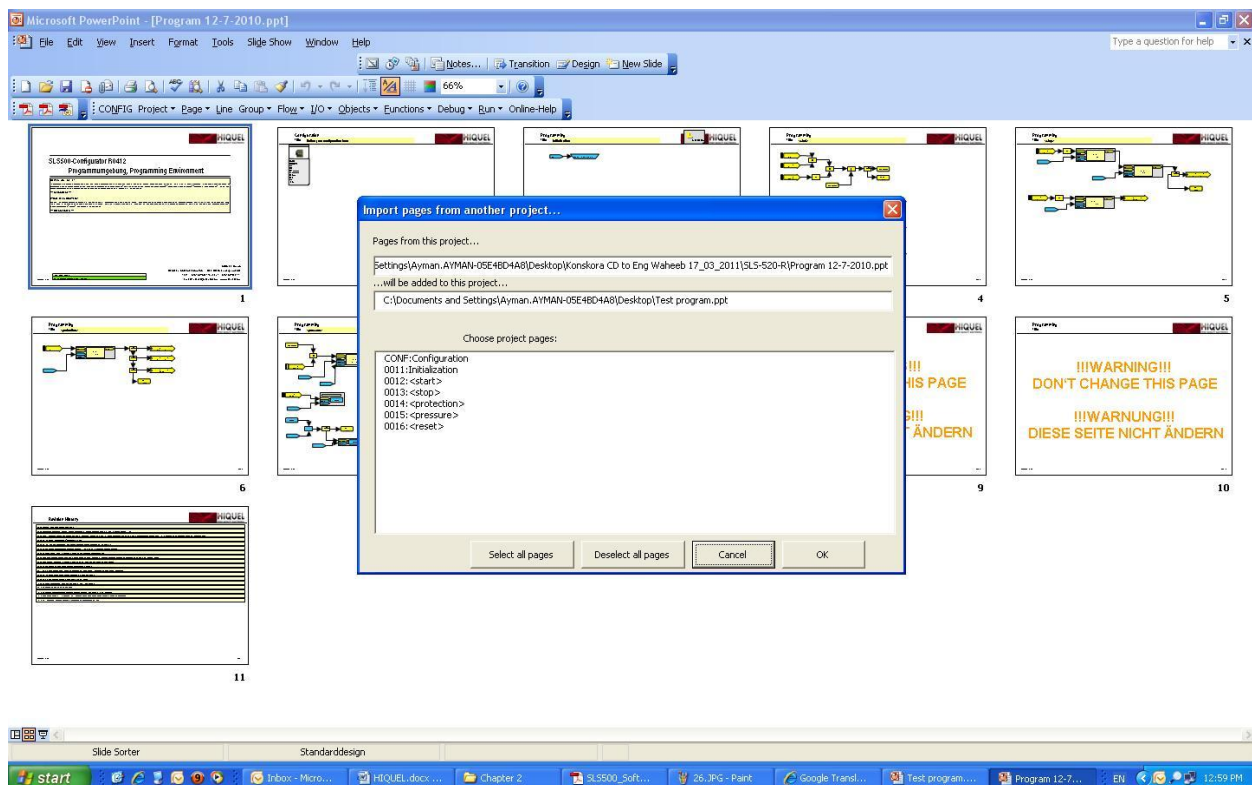
**Project: Import**

Choose this function to import a page or more from another open project into your current project, so you should open the two projects, the first project that you will import from and the second is our project that you will export the page to, but you should do this function from the other project that we want to import the page from.

For example I want to import a page from a project named "Program 12-7-2010" to our project "Test program" so you should follow the instructions:

1. Open the two projects.
2. In the project "Program 12-7-2010" open Project menu.
3. Select Import, a window will appear.
4. Click on "Open the project." Icon, the Following window will appear.





The previous window displays all the pages which the project “Program 12-7-2010” contains, and the window “Import pages from another project...” lets you to choose which page/pages you want to import (copy) to our project by page number or page name.

5. Select the pages you want to copy and then press OK.

Now the program that you imported from will close and the pages you chose to import are imported to your program “Test program” at the end of your existing pages.

### Project: Update I/Os

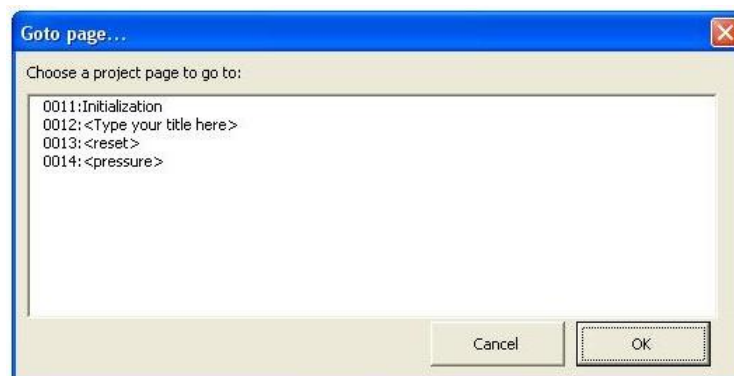
You can edit the already set names of the inputs and outputs with one click. Changing names has to be done on the configuration page. After changing the I/O names in the configuration page, choose [Update I/Os] from the menu to start the update, after finishing you will find the description of the inputs and outputs from the configuration page will correspond to this change within the whole project.

**Page Menu**

SLS-500-Configurator enables you add as many pages as you want and to zoom, copy, delete and other various options you can use to deal with the pages of your application. Click the [Page Menu] to get to the following options:

**Page: Goto**

With this command you can quickly jump to another page of the project. SLS-500-Configurator shows you a detailed overview of all pages with page numbers and titles. Just click onto the desired page and press OK. The page will display immediately!

**Page: Zoom all**

The active page will be displayed completely as full screen.

**Page: Zoom 100%**

The page will be displayed with a zoom factor of 100%.

**Page: Zoom 75%**

The page will be displayed with a zoom factor of 75%.

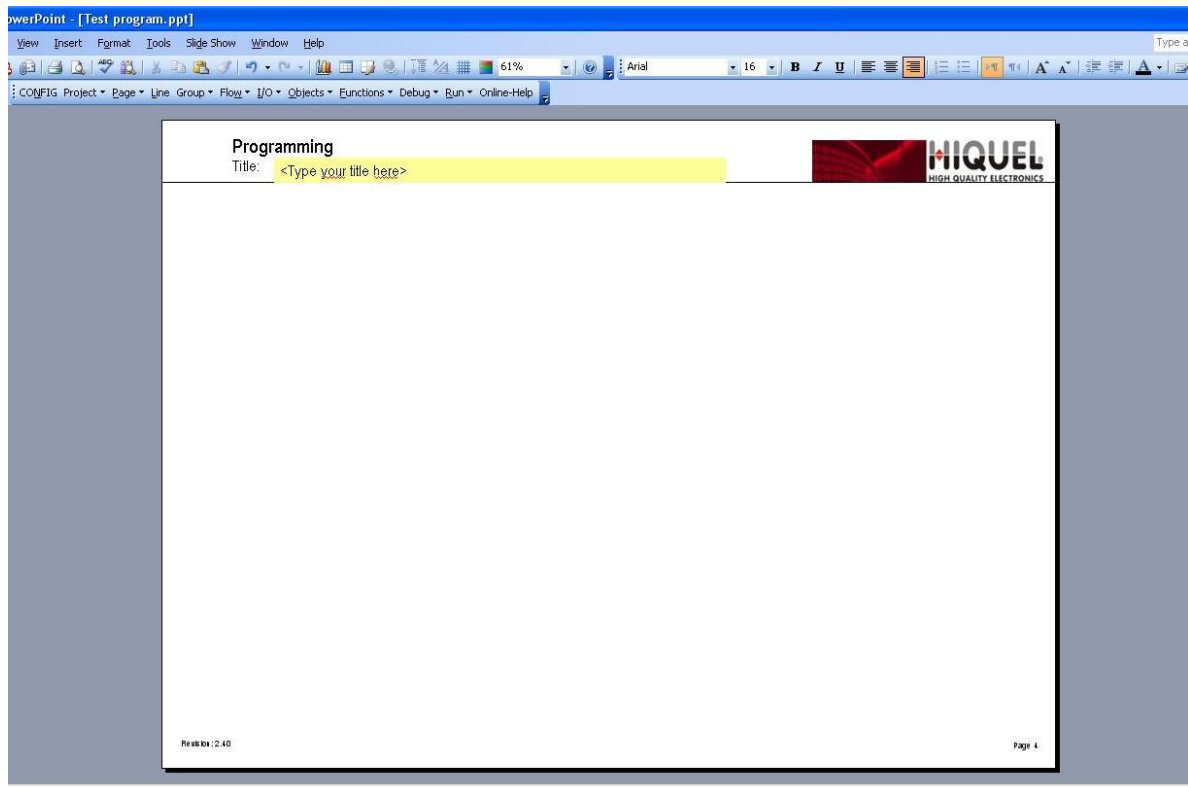
**Page: Zoom 60%**

The page will be displayed with a zoom factor of 60%.

**Page: New**

When you select this function, SLS-500-Configurator places a new programming page as in the following figure before the active page. Therefore if you want to insert a new page after the active page you must advance one page before inserting the new page.





Now define the title of the new programming page:

<b>Programming</b> Title: <Type your title here>
---

To do this you have to click into the text field and type in the text:

<b>Programming</b> Title: My first program page
--

**INFO:** You can spread your program over as many SLS-500-Configurator pages as desired!

**IMPORTANT:** SLS-500-Configurator programs can only be built on programming pages. All other PowerPoint pages will be left out during compilation.

#### **Page: Copy**

With this command the active page will be copied to the next page number.



**Page: Del**

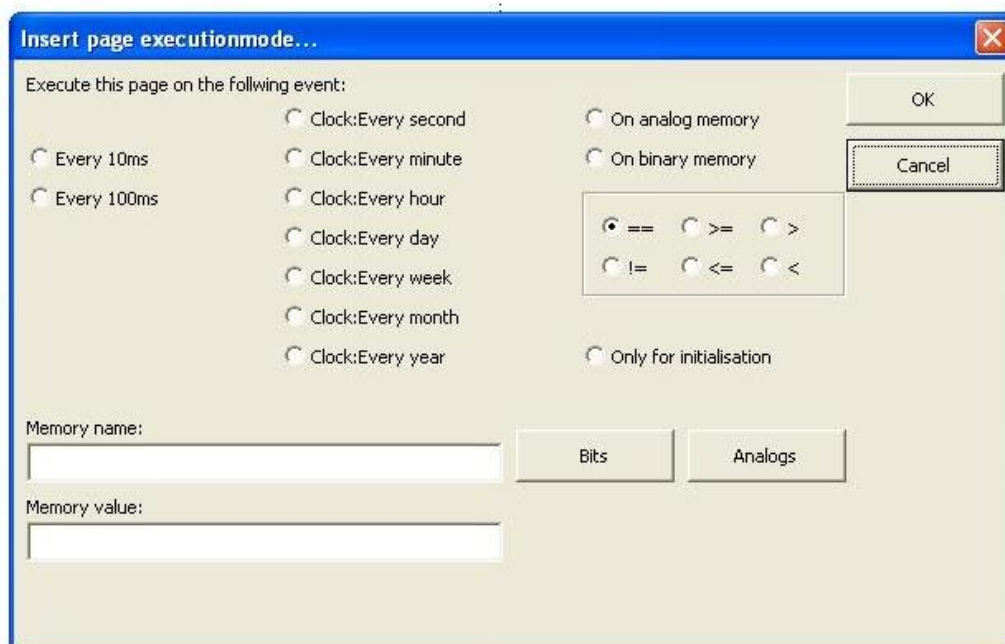
With this command you can delete the active SLS-500-Configurator programming page. After choosing this menu option the following message appears:



Press [Yes] to delete the page or [No] to cancel.

**Page: Execute**

You can select the execution rate or variable dependant operation of each SLS-500-configurator page with this menu option. The following window will occur:



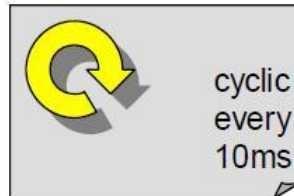
After choosing an execution type format the setting will be displayed on the top right of the page.

To delete the execution format, you just have to click the symbol on the top right and press the key [Del].

And now we will study each execution type in the window:

1. Every 10ms

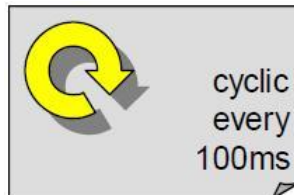
*Symbol:*



*Function:* The page will be executed every 10ms.

2. Every 100ms

*Symbol:*



*Function:* The page will be executed every 100ms.

3. Clock: Every second

*Symbol:*



*Function:* The page will be executed exactly every second. The function is only available with SLS500, which has a real time clock.

## 4. Clock: Every minute

*Symbol:*



*Function:* The page will be executed exactly every minute. The function is only available with SLS500, which has a real time clock.

## 5. Clock: Every hour

*Symbol:*



*Function:* The page will be executed exactly every hour. The function is only available with SLS500, which has a real time clock.

## 6. Clock: Every day

*Symbol:*



*Function:* The page will be executed exactly every day at exactly 00:00:00. The function is only available with SLS500, which has a real time clock.

## 7. Clock: Every week

*Symbol:*



*Function:* The page will be executed exactly every Monday at exactly 00:00:00. The function is only available with SLS500, which has a real time clock.

## 8. Clock: Every month

*Symbol:*



*Function:* The page will be executed exactly on the first of every month at 00:00:00. The function is only available with SLS500, which has a real time clock.

## 9. Clock: Every year

*Symbol:*



*Function:* The page will be executed exactly every year on the first of January at 00:00:00. The function is only available with SLS500, which has a real time clock.

## 10. Only for initialization

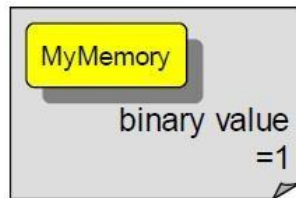
*Symbol:*



*Function:* The page will be executed with every program start up. Use this function for example to initialize the system.

## 11. On binary memory

*Symbol:*

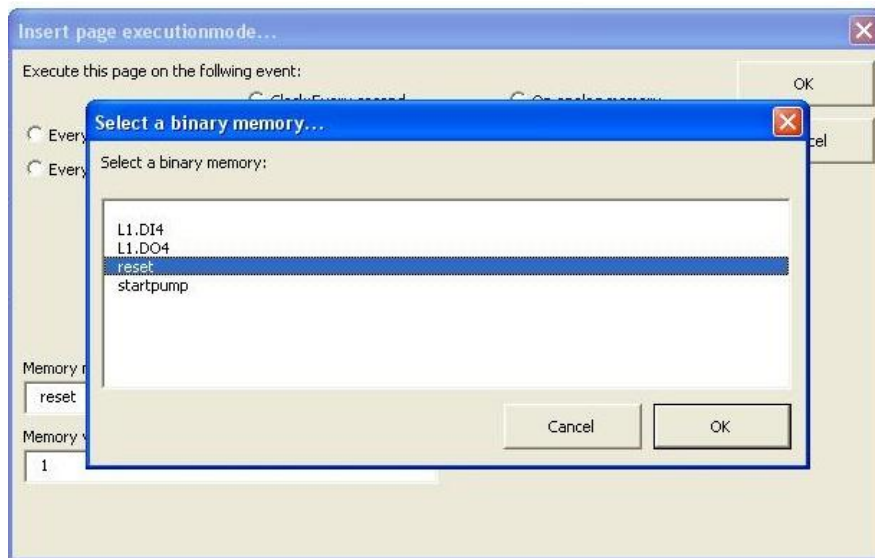


*Function:* This function defines that the page will only be executed if the binary memory value “MyMemory” is 1 otherwise it will not be executed.

To do this condition you should write the name of memory “MyMemory” in [Memory name] field in the [Page Execution Mode] window and the memory value “1” in the [Memory value] field and choose the function “=”.

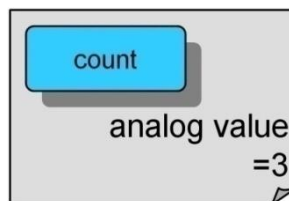
Notes:

- You have two choices for [Memory value] field (1 or 0).
- You can select an existing memory in your program from [Bits] button which will open a window that shows all binary memories in your application instead of writing it, as in the following figure, you can select the memory you want then press OK.



## 12. On analog memory

*Symbol:*

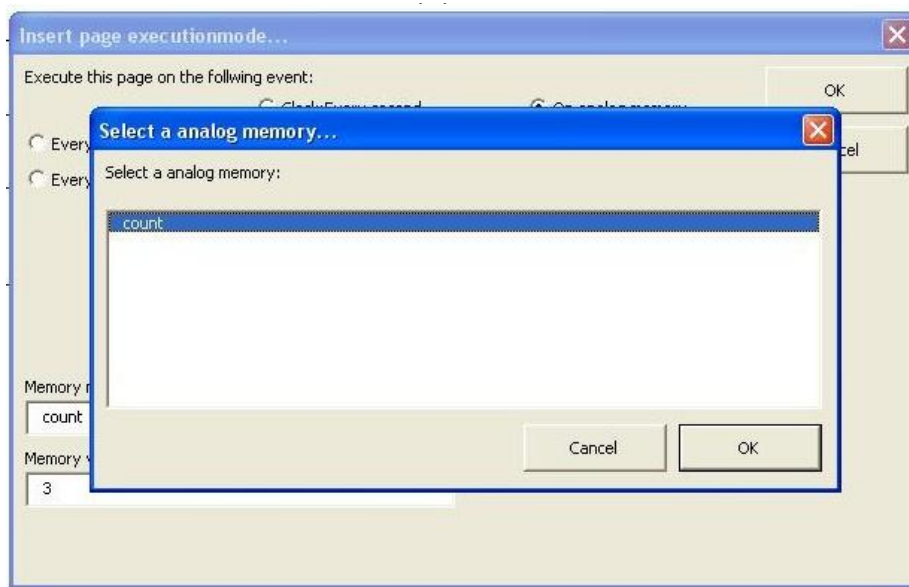


*Function:* This function defines that the page will only be executed if the analog memory value “count” is equal to 3, otherwise it will not be executed.

To do this condition you should write the name of memory “count” in [Memory name] field in the [Page Execution Mode] window and the memory value “3” in the [Memory value] field and choose the function “=”.

### Notes:

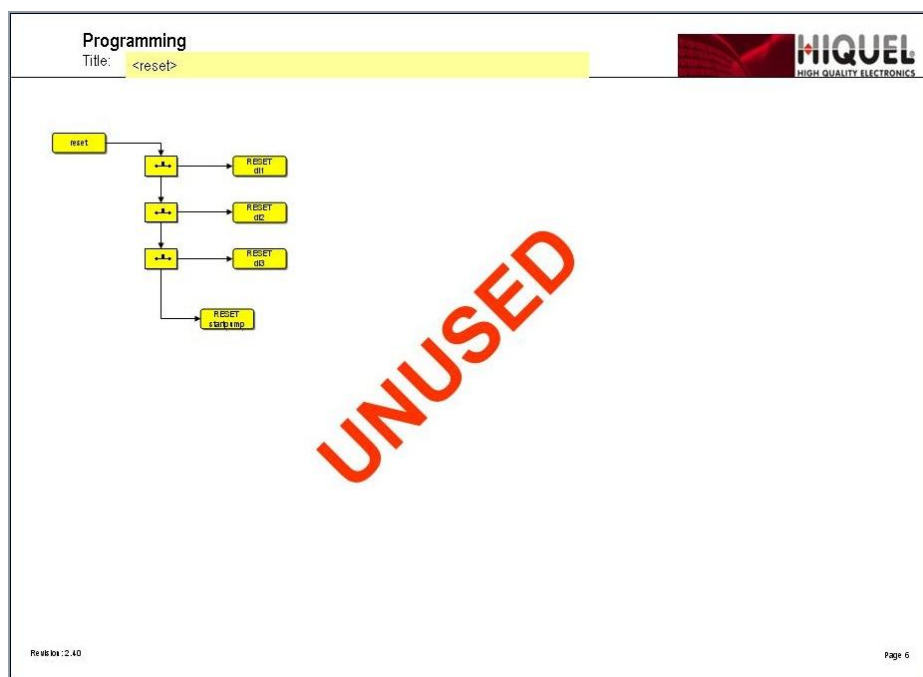
- You have the choice for “Memory value” field to put any value you want because it’s an analog memory.
- Because it is an analog memory you also have choices more than (Equal) such as (Not equal, greater than, greater than or equal, less than and less than or equal); you will find them in the same window.
- You can select an existing memory in your program from “Analog” button which will open a window that shows all analog memories in your program instead of writing it, as in the following figure, you can select the memory that you



### Page: Ignore

Use this command to leave out the whole content of the page during the next compilation.

To warn you of this, “**UNUSED**” will be written across the page, as in the following figure. Select this command a second time, “**UNUSED**” will disappear and the page will be included again with the next compilation.



**Line** Use the connections to connect the individual objects with each other.

*Symbol:*



*Data type:* Depending on the object, connections can operate with all data types.

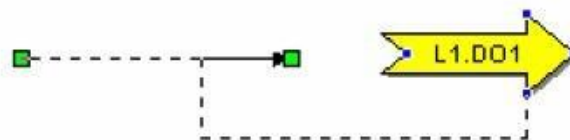
*Function:* The line connects the output of an object to the input of another object. It's important for the connection that you have the correct direction of the arrow, and to connect any object you create in your program.

#### Create connections:

You can add a new connection object by choosing Line from the menu, when you click on the connection object, you will see colored circles or rectangles on both ends:



If the circles are green, the ends are free and are not linked with an object. Now move the cursor to one of the ends and click the arrow. Hold the mouse key and drag the line to an object, as follows:



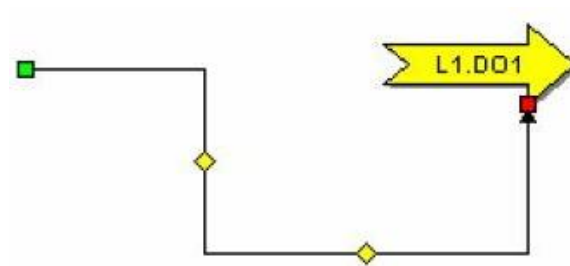
You will notice that small blue symbols appear on the object that you're dragging the line to and that the line snaps to the nearest blue symbol. Select a suitable blue symbol and release the mouse click. The line (arrow) terminal turns from green to red.

This is a confirmation of a correct connection.

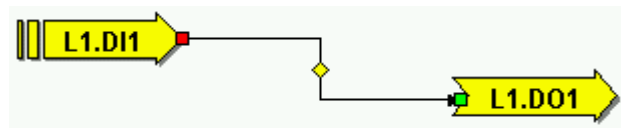




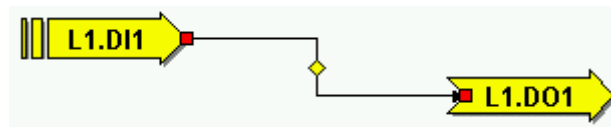
With the yellow symbol in the line, you can change the exact position of the connection:



Correct connections: Click on the connecting line. Both of rectangles must be red, a connection may seem visually to be correct, but if one of the rectangles is green then the right connection has not been made.



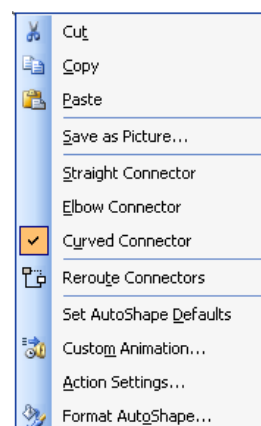
Connect the line with the object again to be as follows:



### Change the style of the line:

To change the line style, right-click the connection line and the following menu will appear:

You can choose from, straight Connector (default), Elbow Connector or Curved Connector.

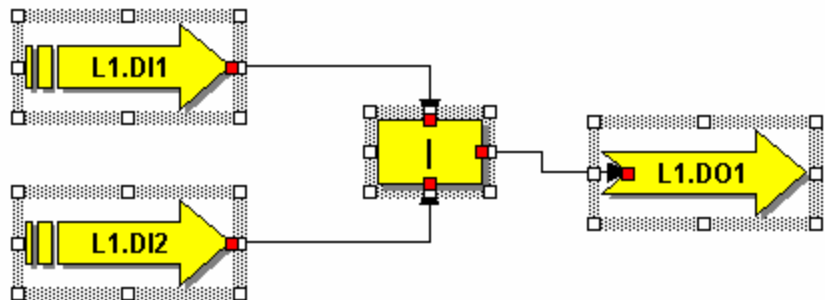


## Group Menu

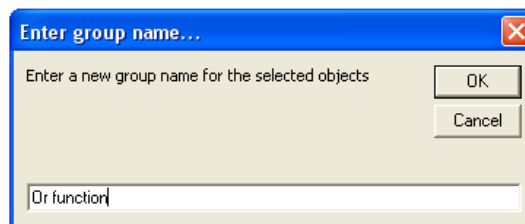
Groups are frequently used to combine SLS-500-Configurator objects or connections that perform a commonly used control function. You can define a name for each group. You can add groups to the current page by using this name at any time. Groups are only available within the same project. You cannot transmit groups of one project to another!

### Group: Export

Mark one or several objects that you want to save as a group:



Next choose Group/Export from the menu to get to the following window:

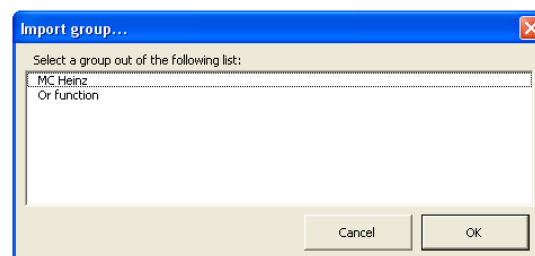


Now type a new name for the group and press OK. The objects will be exported as a group.

### Group: Import

To import a previously saved group, do as follows:

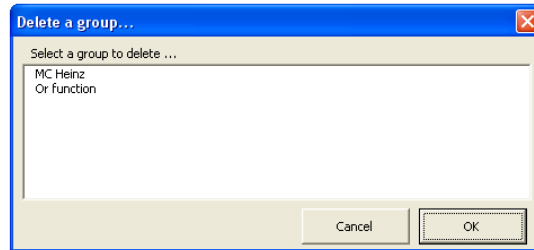
Choose Group/Import from the menu to get to the following list:



From the list, choose which group you want to import and press OK. All objects of the group will be added to the current page and are available again as single objects!

**Group: Delete**

In order to delete a group choose Group/Delete from the menu to get to a list containing the available groups:



Choose the group you want to delete and confirm by clicking OK.

**Flow Menu****Data types of SLS-500-Configurator:**

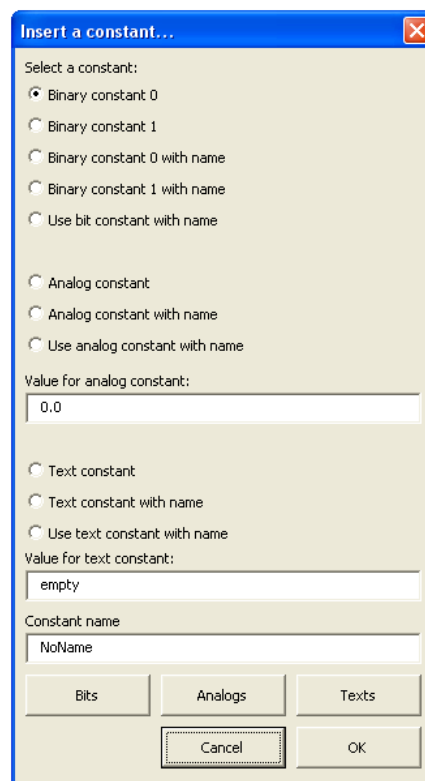
SLS-500-CONFIGURATOR supports three different data types:

1. Bit data: This data type can save exactly 1 Bit or the information 0 or 1.  
Examples for bit data are digital inputs or outputs or status markers.
2. Analog data: This data type has the ability to process signed analog values to three decimal places; the maximum numerical range is  $-2147483.647$  to  $+2147483.648$ .  
Examples for analog data are analog inputs or analog outputs, times, counting.
3. Text data: This data type can save text messages. Depending on the destination system, character strings with different lengths are supported. Maximum 20 characters per text.  
Examples for text data are messages for a text display or serial communication objects or SMS message.

**Flow: Constants**

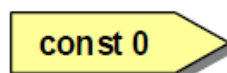
Constants define a fixed value. You can set constants for every data type of SLS-500-Configurator, as shown in the following window, using several constants with the same value, a “constant name” can be predefined and jointly changed.

Choose from the SLS-500-Configurator menu bar the command Flow/Constants. You will get the following window:

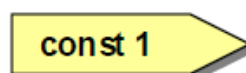
**Binary Constants:**

Define a value of 0 or 1, now choose binary constant 0, from the previous window, and confirm your choice with OK.

The active programming page will insert the following symbol:

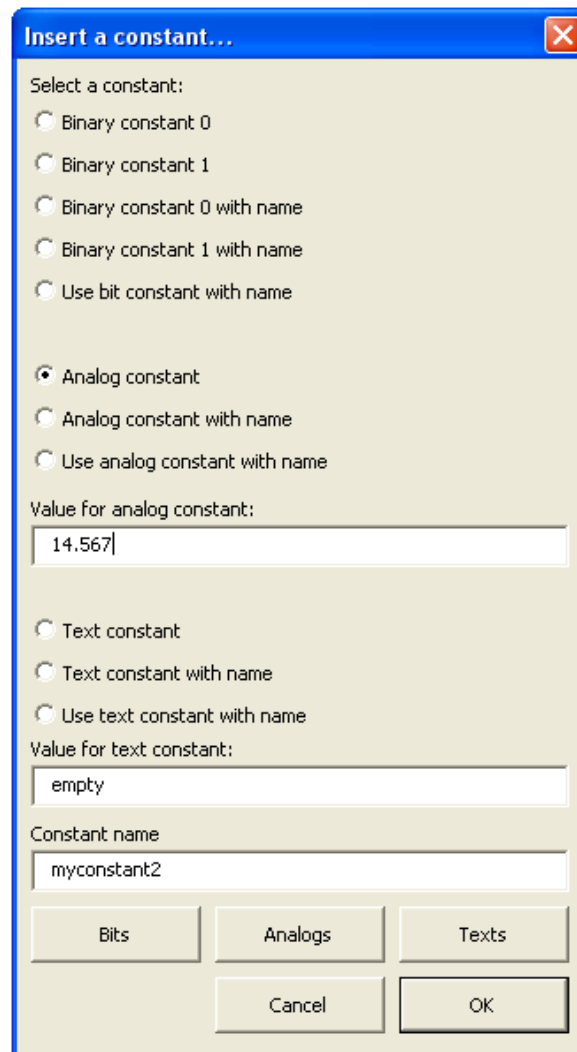


If you choose binary constant 1 the following symbol will be inserted:



*Analog Constants:*

Choose Analog constant, from the following window, and set a fixed value for the new constant in the field [Value for analog constant] as shown. Then confirm with OK.



**Insert a constant...**

Select a constant:

- ☐ Binary constant 0
- ☐ Binary constant 1
- ☐ Binary constant 0 with name
- ☐ Binary constant 1 with name
- ☐ Use bit constant with name
- ☒ Analog constant
- ☐ Analog constant with name
- ☐ Use analog constant with name

Value for analog constant:

14.567

- ☐ Text constant
- ☐ Text constant with name
- ☐ Use text constant with name

Value for text constant:

empty

Constant name

myconstant2

Bits    Analogs    Texts

Cancel    OK

The following symbol will be inserted:

**14.567**

For negative constants:

**-345.56**

*Text Constants:*

Define fixed character strings. Choose Text constant, from the following window, and type the following into the text fields of the dialogue in the field [Value for text constant], as shown then confirm with OK.

Insert a constant...

Select a constant:

- ☐ Binary constant 0
- ☐ Binary constant 1
- ☐ Binary constant 0 with name
- ☐ Binary constant 1 with name
- ☐ Use bit constant with name
- ☐ Analog constant
- ☐ Analog constant with name
- ☐ Use analog constant with name

Value for analog constant:

0.0

- ☒ Text constant
- ☐ Text constant with name
- ☐ Use text constant with name

Value for text constant:

Over temperature

Constant name

NoName

Bits Analog Texts

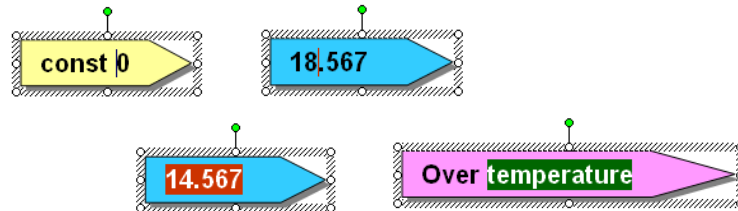
Cancel OK

The following symbol will be inserted:

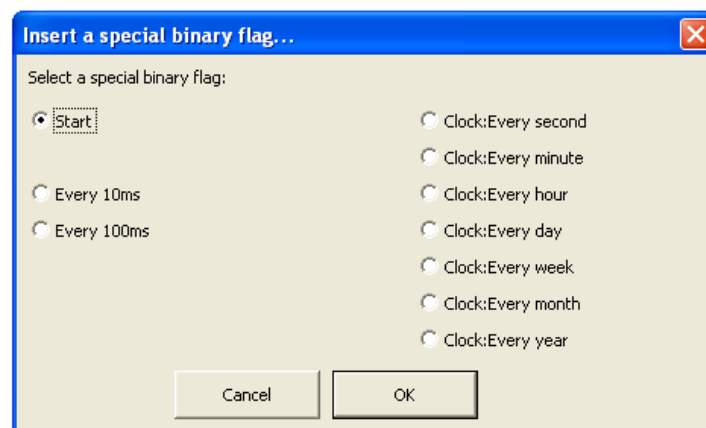
**Over temperature**

**NOTE:**

To change the value of any constant type “Binary, Analog, or Text constant” afterwards, click the value inside the symbol and edit the number/text!

**Flow: Special flags**

SLS-500-Configurator has a series of special flags, which display special signals. To insert a special flag choose Flow/Special flags. The following window will appear, select the desired flag and click OK.

**1. Start**

*Symbol:*



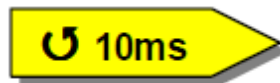
*Data type:* Bit

*Function:* This flag has the value 1 only during the first program cycle. Otherwise, this bit is always zero. Use this flag for example to initialize values.



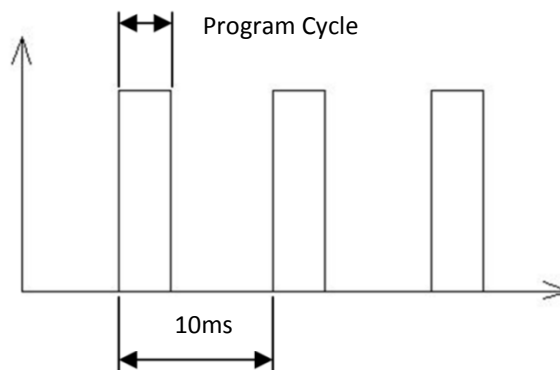
## 2. Every 10ms

*Symbol:*



*Data type:* Bit

*Function:* The flag has the value 1 for one program cycle at intervals of 10mS. Otherwise, the flag is always zero. Use this flag for example with signal time measuring.



## 3. Every 100ms

*Symbol:*

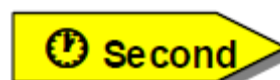


*Data type:* Bit

*Function:* The flag has the value 1 for one program cycle at intervals of 100mS. Otherwise, the flag is always zero. Use this flag for example with signal time measuring.

## 4. Clock: Every second

*Symbol:*



*Data type:* Bit

*Function:* An integrated real time clock creates this flag. The flag returns the value 1 every second for exactly one cycle, otherwise, it is zero. Use this flag for a flasher signal for example.

## 5. Clock: Every minute

*Symbol:*

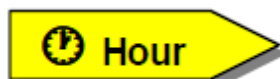


*Data type:* Bit

*Function:* An integrated real time clock creates this flag. The flag returns the value 1 every minute for exactly one cycle, otherwise, it is zero.

## 6. Clock: Every hour

*Symbol:*

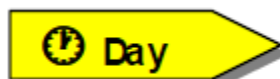


*Data type:* Bit

*Function:* An integrated real time clock creates this flag. The flag returns the value 1 every hour for exactly one cycle, otherwise, it is zero.

## 7. Clock: Every day

*Symbol:*

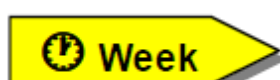


*Data type:* Bit

*Function:* An integrated real time clock creates this flag. The flag returns the value 1 every day at 00:00:00 for exactly one cycle, otherwise, it is zero.

## 8. Clock: Every week

*Symbol:*



*Data type:* Bit

*Function:* An integrated real time clock creates this flag. The flag returns the value 1 every week on Sunday at 00:00:00 for exactly one cycle, otherwise, it is zero.

## 9. Clock: Every month

*Symbol:*

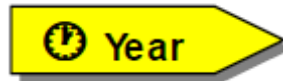


*Data type:* Bit

*Function:* An integrated real time clock creates this flag. The flag returns the value 1 on the first of every month at 00:00:00 for exactly one cycle, otherwise, it is zero.

## 10. Clock: Every year

*Symbol:*



*Data type:* Bit

*Function:* An integrated real time clock creates this flag. The flag returns the value 1 every year on the first of January at 00:00:00 for exactly one cycle, otherwise, it is zero.

**Flow: Memories**

Use the memory spaces to store values of one of the three data types (Bit, Analog, and Text). These values can be reloaded any time at another place within the program to continue processing. Every memory has a name.

NOTE: You can define the same name for a binary memory, an analog memory, and a text memory the graphic color tells you the difference, because the SLS-500 CONFIGURATOR is color coded, so the binary data is colored yellow, the analog data is colored blue, and the text data is colored violet.

However, take care which memory you are actually accessing! Choose Flow/Memories from the menu, the following window will appear, you can select the following memories:

Select a memory...

Select a memory:

- ☒ Binary memory
  - ☐ IF input is high SET binary memory
  - ☐ IF input is high RESET binary memory
  - ☐ IF input is high TOGGLE binary memory
- ☐ Analog memory
  - ☐ IF rising edge SET analog memory
  - ☐ IF falling edge SET analog memory
  - ☐ IF rising or falling edge SET analog memory
  - ☐ IF input is one SET analog memory
  - ☐ IF input is zero SET analog memory
- ☐ Text memory
  - ☐ IF rising edge SET text memory
  - ☐ IF falling edge SET text memory
  - ☐ IF rising or falling edge SET text memory
  - ☐ IF input is one SET text memory
  - ☐ IF input is zero SET text memory

Memory name:

Bits    Analog    Texts

Cancel    OK

### Binary memory

*Symbol:*

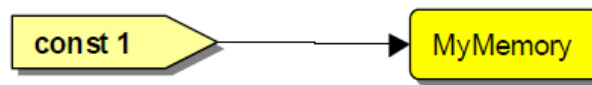


*Data type:* Bit

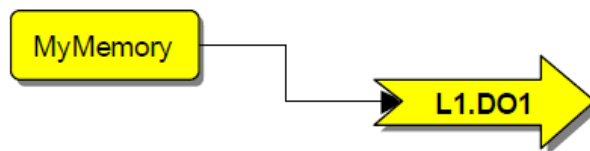
*Function:* Binary memory saves one Bit and transmits it, i.e. it can work as an input or output or both together.

In addition, as you see, its color is yellow, this indicates that is binary data, and all inputs to it and outputs from it should be binary also as we see in the next examples.

Examples:



- The constant value 1 will be transmitted to “MyMemory”



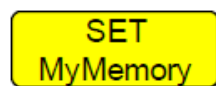
- The active value of my memory will be transmitted to digital output L1.DO1.



- The current value of digital input L1.DI1 will be saved to the “1stMemory” also to the 2ndMemory.

### IF input is high SET binary memory

Symbol:

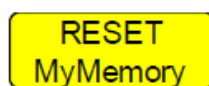


Data type: Bit

Function: IF input of this binary memory receives the value 1, “MyMemory” will be set to one, and will keep this value whatever its input becomes zero until it has been reset using the next function.

### IF input is high RESET binary memory

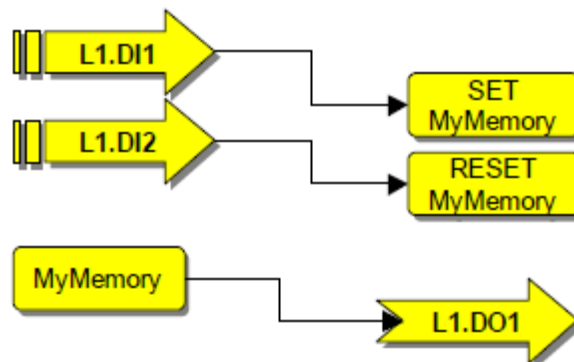
Symbol:



Data type: Bit

Function: If input of this binary memory receives the value 1, “MyMemory” will be reset to zero, and will keep this value whatever its input becomes one until it has been set again using previous function.

Example:



If digital input L1.DI1 is activated the variable memory “MyMemory” will be set to one. This status of “MyMemory” stays active until the input L1.DI2 is activated. “MyMemory” will then be reset to zero. The status of digital output L1.DO1 will be on when “MyMemory” is one, and off when “MyMemory” is zero.

#### IF input is high TOGGLE binary memory

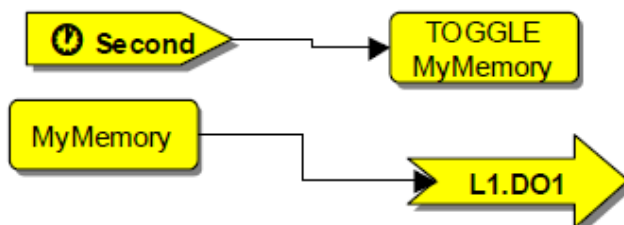
Symbol:



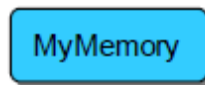
Data type: Bit

Function: If the input value of the bit memory is 1, the current content of “MyMemory” will be inverted.

Example:

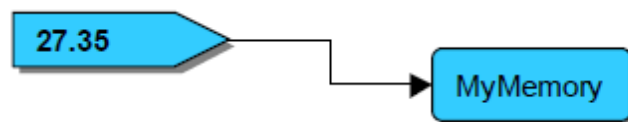


The value of “MyMemory” is inverted every second. Digital output L1.DO1 flashes every second.

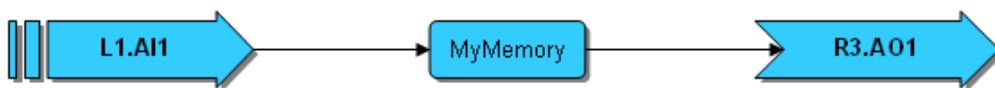
**Analog memory***Symbol:**Data type:* Analog

*Function:* The analog memory is able to store an analog value and to transmit it. This means it can work as an input, output, or both together.

In addition, as you see, its color is blue, this indicates that it is analog data, and all inputs and outputs connected to it should be analog as we see in the next examples.

*Examples:*

- The constant value 27.35 is transmitted to the analog memory “MyMemory”.

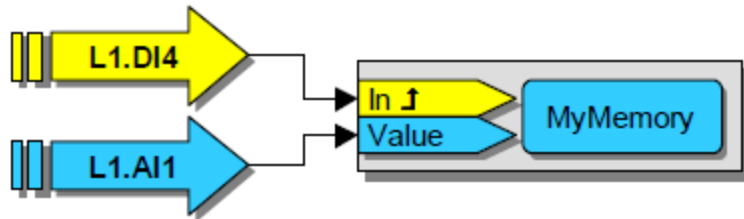


- The current value of analog input L1.AI1 is transferred to MyMemory. Also the value of “MyMemory” will be transmitted to analog output R3.AO1.

**IF rising edge SET analog memory***Symbol:**Data type:* In (Digital Input), Value (Analog Input)

*Function:* When digital input “In” reads a rising edge, the analog value of input “Value” will be saved to the analog memory “MyMemory”.

Example:



With every rising edge of digital input L1.DI4, the existing value of analog input L1.AI1 will be saved to the analog memory "MyMemory".

### IF falling edge SET analog memory

Symbol:



*Data type:* In ( Digital Input), Value (Analog Input)

*Function:* When digital input "In" reads a falling edge, the existing analog value of input "Value" will be saved to the analog memory "MyMemory".

### IF rising or falling edge SET analog memory

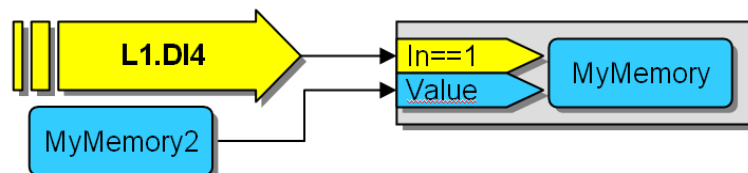
Symbol:



*Data type:* In ( Digital Input ), Value (Analog Input)

*Function:* When digital input "In" reads a rising or falling edge, the existing analog value of input "Value" will be saved to the analog memory "MyMemory".



**IF input is one SET analog memory***Symbol:**Data type:* In (Digital Input), Value (Analog Input)*Function:* All the time the digital input “In” reads the binary value 1, the analog value of input “Value” will be saved to the analog memory “MyMemory”.*Example:*

All the time the digital input L1.DI4 reads the binary value “one”, the value of analog memory “MyMemory2” will be transmitted to the analog memory “MyMemory”.

**IF input is zero SET analog memory***Symbol:**Data type:* In (Digital Input), Value (Analog Input)*Function:* All the time the digital input “In” reads the binary value “zero”, the analog value of input “Value” will be saved to the analog memory “MyMemory”.

**Text memory**

*Symbol:*

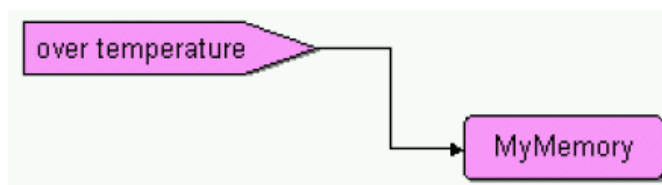


*Data type:* Text

*Function:* The Text memory is able to store a text value and to transmit it. This means it can work as an input, output, or both together.

In addition, as you see, its color is violet, this indicates that it is text data, and all inputs and outputs connected to it should be text as we see in the next example.

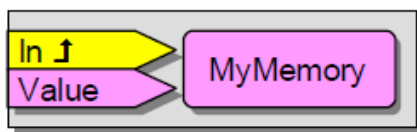
*Example:*



The constant text value "over temperature" will be transmitted and saved to text memory "MyMemory".

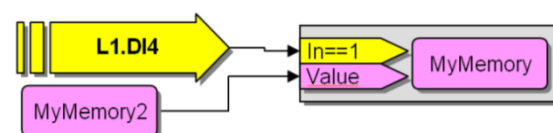
**IF rising edge SET text memory**

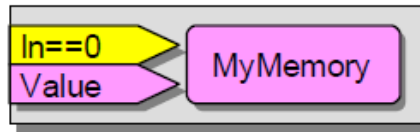
*Symbol:*



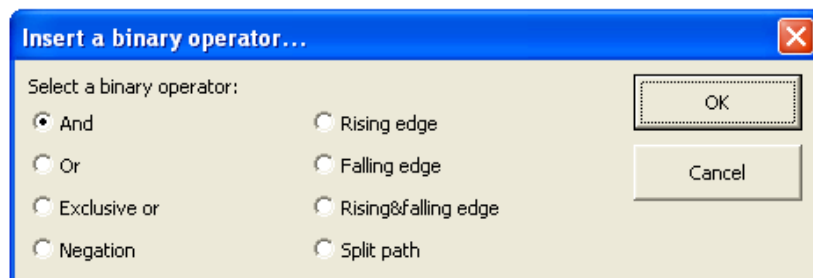
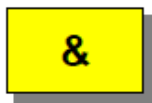
*Data type:* In (Digital Input), Value (Text Input)

*Function:* If digital input "In" reads a rising edge, the text value of input "Value" will be transmitted to the text memory "MyMemory".

**IF falling edge SET text memory***Symbol:**Data type:* In (Digital Input), Value (Text Input)*Function:* If digital input “In” reads a falling edge, the text value of input “Value” will be transmitted to the text memory “MyMemory”.**IF rising or falling edge SET text memory***Symbol:**Data type:* In (Digital Input), Value (Text Input)*Function:* If digital input “In” reads a rising edge or falling edge, the text value of input “Value” will be transmitted to the text memory “MyMemory”.**IF input is one SET analog memory***Symbol:**Data type:* In (Digital Input), Value (Text Input)*Function:* All the time the digital input “In” reads the binary value 1, the text value of input “Value” will be saved to the text memory “MyMemory”.*Example:**All the time the digital input L1.DI4 reads the binary value “one”, the value of the text memory “MyMemory2” will be transmitted to the text memory “MyMemory”.*

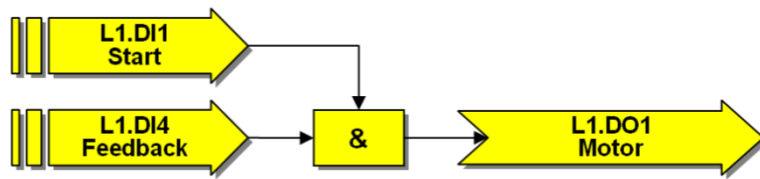
**IF input is zero SET analog memory***Symbol:**Data type:* In (Digital Input), Value (Text Input)*Function:* All the time the digital input “In” reads the binary value zero, the text value of input “Value” will be saved to the text memory “MyMemory”.**Flow: Bit handling**

There are a series of operators available for binary calculations. Choose Flow/Bit handling from the menu, and select one of the following operators:

**And***Symbol:**Data type:* In1, In2 (Digital Inputs), Out (Digital Output)*Function:* This function calculates the AND-connection by using two input signals and delivers the result to the output

In1	In2	Out
0	0	0
0	1	0
1	0	0
1	1	1

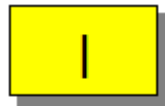
Example:



Digital output L1.DO1 (Motor) is active, only if **BOTH** digital inputs L1.DI1 **AND** L1.DI4 (Start and Feedback) are active (read binary value one).

Or

Symbol:

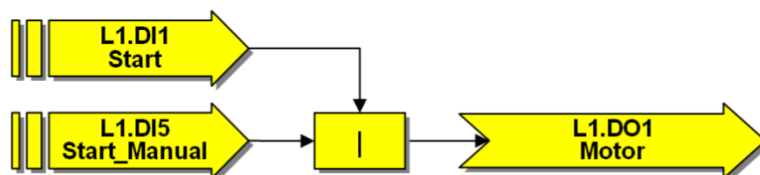


*Data type:* In1, In2 (Digital Inputs), Out (Digital Output)

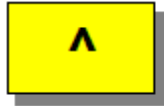
*Function:* This function calculates the OR-connection by using two input signals and delivers the result to the output.

In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	1

Example:

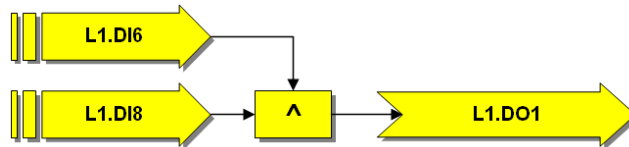


Digital output L1.DO1 (Motor) is active, only if one of the two digital inputs L1.DI1 (Start) **OR** L1.DI5 (Start\_Manual) at least is active (read binary value 1), if both are active then the output is active too.

**XOR***Symbol:**Data type:* In1, In2 (Digital Inputs), Out (Digital Output)

*Function:* This function calculates the EXCLUSIVE OR-connection by using two input signals and delivers the result to the output, according to the next truth table:

In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	1

*Example:*

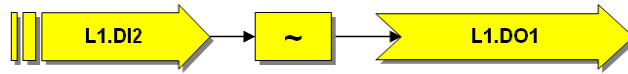
Digital output L1.DO1 is active, as long as the two digital inputs L1.DI6 and L1.DI8 have the same status. If both are active or non active, the digital output will be active.

**Negation***Symbol:**Data type:* In (Digital Input), Out (Digital Output)

*Function:* The current input binary (Digital) signal will be inverted and delivered to output binary (Digital) signal, according to the next truth table:

In	Out
0	1
1	0

Example:



Digital output L1.DO1 has the opposite signal of digital input L1.DI2.

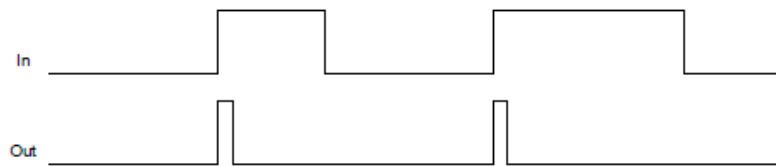
### Rising edge

Symbol:

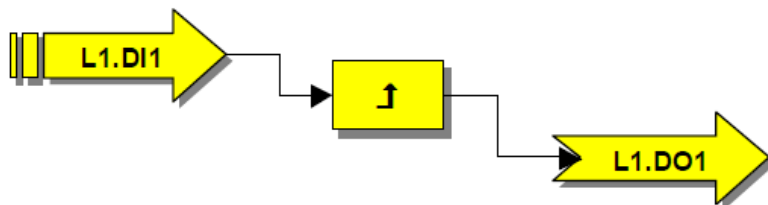


*Data type:* In (Digital Input ), Out (Digital Output)

*Function:* If the input signal has a rising edge, this function is high for exactly one cycle, as shown in the timing diagram below:



Example:



If digital input L1.DI1 reads a rising edge, digital output L1.DO1 will be high for exactly one cycle.

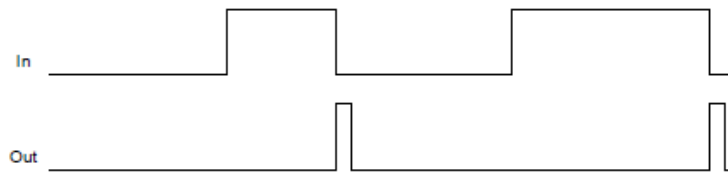
**Falling edge**

*Symbol:*

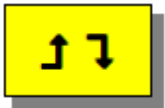


*Data type:* In (Digital Input), Out (Digital Output)

*Function:* If the input signal has a falling edge, this function is high for exactly one cycle, as shown in the timing diagram below:

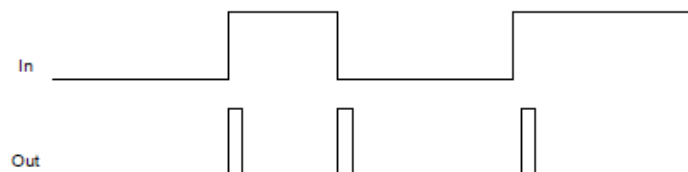
**Rising and Falling edge**

*Symbol:*

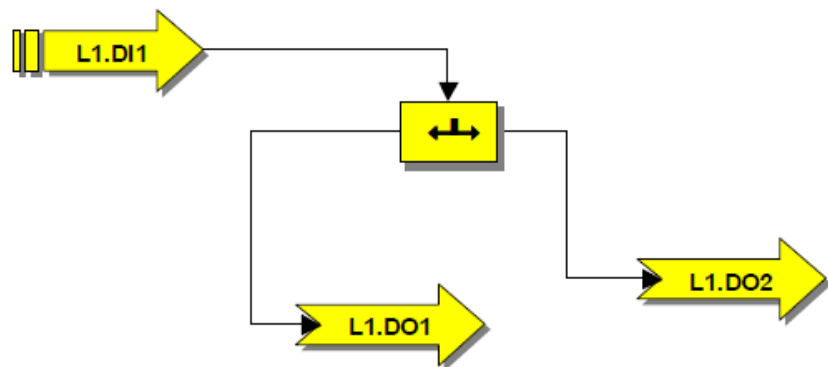


*Data type:* In (Digital Input), Out (Digital Output)

*Function:* If the input signal has a rising or falling edge, this function is high for exactly one cycle, as shown in the timing diagram below:



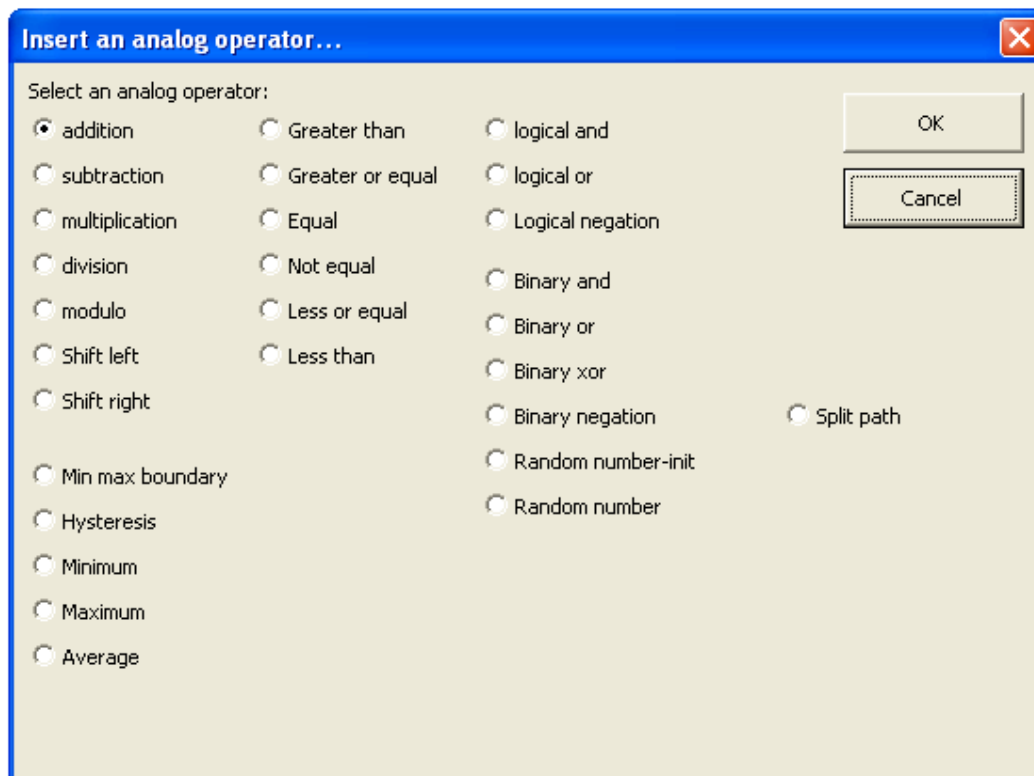


**Split***Symbol:**Data type:* In (Digital Input), Out1, Out2 (Digital Outputs)*Function:* This function splits the data into two paths. Both of outputs have the same signal as the input.*Example:*

The input signal L1.DI1 will be simultaneously transmitted to outputs L1.DO1 and L1.DO2.

### Flow: Analog handling

The following menu is specialized for analog signal operators and is available for processing the analog signals. Choose Flow/Analog handling from the menu to show the following window:



### Addition

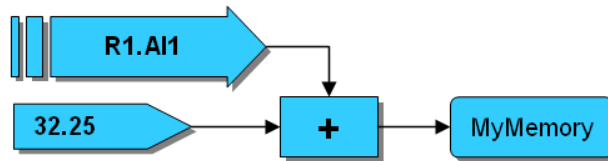
*Symbol:*



*Data type:* In1, In2 (Analog Inputs), Out (Analog Output)

*Function:* This function calculates the sum of the two analog signals In1 and In2 and delivers the result to output Out.

Example:



The value 32.25 will be added to the current value of analog input R1.AI1. The result will be saved to the analog memory "MyMemory".

### Subtraction

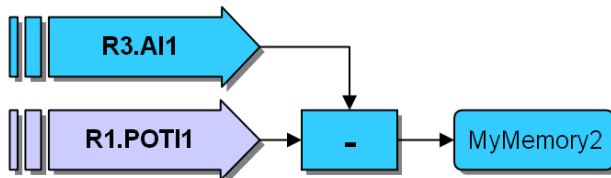
Symbol:



*Data type:* In1, In2 (Analog Inputs), Out (Analog Output)

*Function:* This function subtracts the value of one analog input "In2" from another analog input "In1" and delivers the result to analog output "Out".

Example:



The analog value of the first extension potentiometer will be subtracted from the current value of analog input R3.AI1. The result will be saved to the analog memory "MyMemory2".

### Multiplication

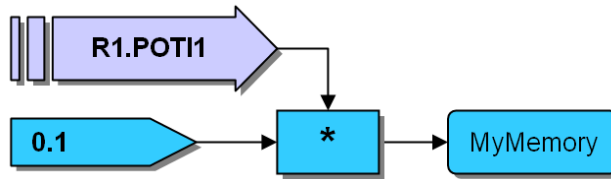
Symbol:



*Data type:* In1, In2 (Analog Inputs), Out (Analog Output)

*Function:* This function multiplies the two analog signals "In1" and "In2" and delivers the result to analog output "Out".

Example:



The value of the first extension potentiometer will be multiplied by factor 0.1, and the result will be saved to the analog memory “MyMemory”. In this way you can get a potentiometer value between 0 and 10, instead of 0 and 100 (Default value for potentiometer or any analog input).

### Division

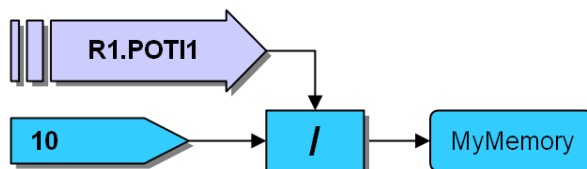
*Symbol:*



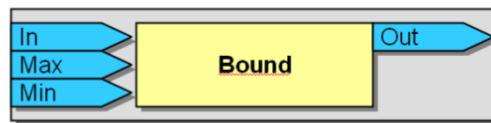
*Data type:* In1, In2 (Analog Inputs), Out (Analog Output)

*Function:* This function divides analog signal “In1” by “In2” and delivers the result to analog output “Out”.

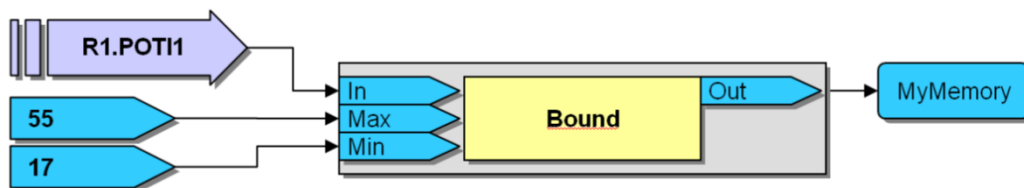
Example:



The current potentiometer value R1.POTI1 is divided by 10. The result will be saved to analog memory “MyMemory”. In this way you can get a potentiometer value between 0 and 10.

**Min-Max boundary***Symbol:**Data type:* In, Max, Min (Analog Inputs), Out (Analog Output)

*Function:* This function determines the maximum “Max” and minimum “Min” values for any set value “In” and delivers the set value with new boundary to analog output “Out”, which means that if the set value “In” is below the “Min” value, the “Out” will equal to the “Min” value, if the set value “In” is above the “Max” value, the “Out” will equal to the “Max” value, and if the set value “In” is between the “Min” and “Max” value, the “Out” will equal to the “In” value.

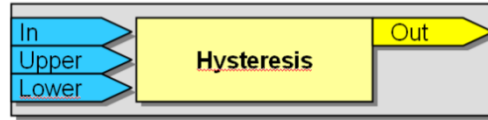
*Example:*

The value of the first extension potentiometer will be delivered to the analog memory “MyMemory” with maximum boundary “55” and minimum boundary “17”, for more clarification please see the next table:

In	Out
0	17
15.24	17
17.1	17.1
25.754	25.754
54.986	54.986
55.105	55
80	55
100	55

## Hysteresis

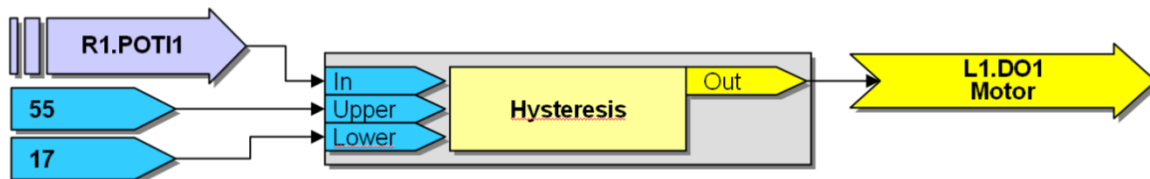
Symbol:



Data type: In, Upper, Lower (Analog Inputs), Out (Digital Output)

**Function:** This function determines a condition for run a digital output “Out” according to specific values for run and stop for an analog input “In”, which means that if the analog value “In” is above the “Upper” value, the “Out” will be activated as long as the analog value “In” is above “Lower” and if the analog value “In” is below the “Lower” value, the “Out” will be deactivated as long as the analog value “In” is below “Upper”.

Example:



The digital output L1.DO1 will be activated if the current value of R1.POTI1 is above the “Upper” value, and it will stay activated as long as R1.POTI1 is still above “Lower” value, when R1.POTI1 decreases below “Lower” value the L1.DO1 is deactivated and it will stay as it until R1.POTI1 increases above “Upper” value.

## Minimum

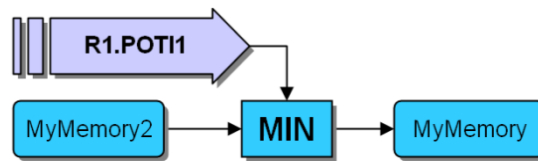
Symbol:



Data type: In1, In2 (Analog Inputs), Out (Analog Output)

**Function:** This function is used to compare between 2 analog inputs “In1” and “In2”, and delivers the smaller value of the 2 inputs to the output “Out”.

Example:



The analog memory “MyMemory” is the smaller value of the R1.POTI1 and “MyMemory2”, for more clarification please see the next table:

R1.POTI1	MyMemory2	MyMemory
0	0	0
15.24	0	0
17.15	8.45	8.45
17.15	17.16	17.15

### Maximum

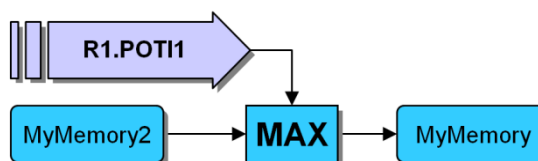
Symbol:



Data type: In1, In2 (Analog Inputs), Out (Analog Output)

Function: This function is used to compare between 2 analog inputs “In1” and “In2”, and delivers the bigger of the 2 inputs to the output “Out”.

Example:

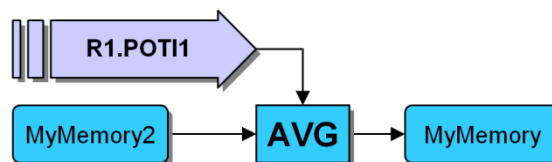


The analog memory “MyMemory” is the bigger of the R1.POTI1 and “MyMemory2”, for more clarification please see the next table:

R1.POTI1	MyMemory2	MyMemory
0	0	0
15.24	0	15.24
17.15	8.45	17.15
17.15	17.16	17.16
54.986	17.16	54.986

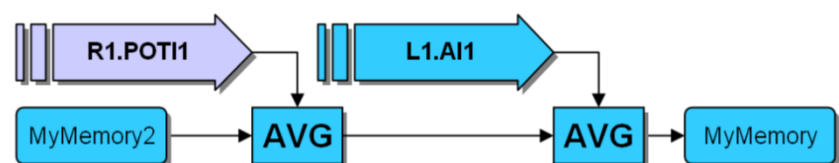
**Average***Symbol:**Data type:* In1, In2 (Analog Inputs), Out (Analog Output)

*Function:* This function is used to calculate the average value between 2 analog inputs “In1” and “In2”, and delivers the result to the output “Out”.

*Example:*

The function “AVG” calculates the average value between the two analog inputs R1.POTI1 and “MyMemory2”, and delivers the result to analog memory “MyMemory”.

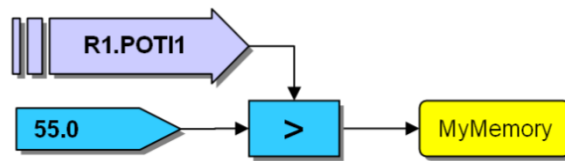
Note: if you want to calculate the average value of more than two analog values, use more than 1 “AVG” function for each new analog input, as in the next example:

**Greater than***Symbol:**Data type:* In1, In2 (Analog Inputs), Out (Digital Output)

*Function:* This function is used to compare between 2 analog values “In1” and “In2”, if “In1” is greater than “In2”, the digital output “Out” will be activated.



Example:



If R1.POTI1 is greater than “55” The digital memory “MyMemory” will be activated, otherwise it will be deactivated (Zero).

### Greater or equal

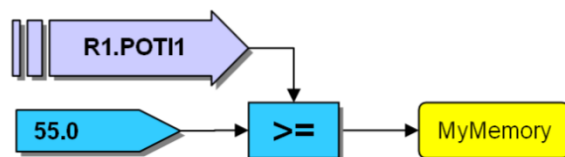
Symbol:



Data type: In1, In2 (Analog Inputs), Out (Digital Output)

Function: This function is used to compare between 2 analog values “In1” and “In2”, if “In1” is greater than or equal to “In2”, the digital output “Out” will be activated.

Example:



If R1.POTI1 is greater than or equal to “55” The digital memory “MyMemory” will be activated, otherwise it will be deactivated (Zero).

### Equal

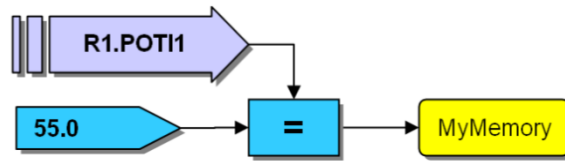
Symbol:



Data type: In1, In2 (Analog Inputs), Out (Digital Output)

Function: This function is used to compare between 2 analog values “In1” and “In2”, if “In1” is equal to “In2”, the digital output “Out” will be activated.

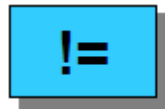
Example:



If R1.POTI1 is equal to “55” The digital memory “MyMemory” will be activated, otherwise it will be deactivated (Zero).

### Not equal

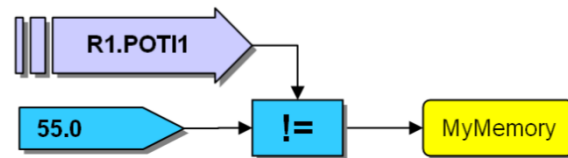
Symbol:



Data type: In1, In2 (Analog Inputs), Out (Digital Output)

Function: This function is used to compare between 2 analog values “In1” and “In2”, if “In1” is not equal to “In2”, the digital output “Out” will be activated.

Example:



If R1.POTI1 is not equal to “55” The digital memory “MyMemory” will be activated, otherwise it will be deactivated (Zero).

### Less or equal

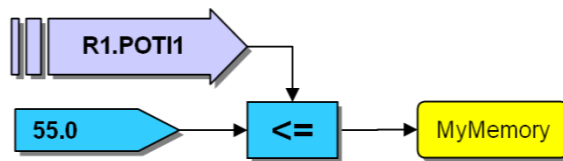
Symbol:



Data type: In1, In2 (Analog Inputs), Out (Digital Output)

Function: This function is used to compare between 2 analog values “In1” and “In2”, if “In1” is less than or equal to “In2”, the digital output “Out” will be activated.

Example:



If R1.POTI1 is less than or equal to “55” The digital memory “MyMemory” will be activated, otherwise it will be deactivated (Zero).

### Less than

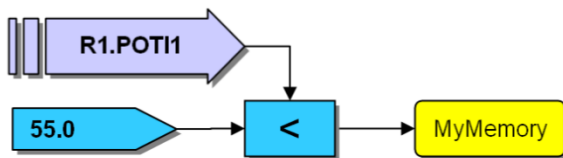
Symbol:



Data type: In1, In2 (Analog Inputs), Out (Digital Output)

Function: This function is used to compare between 2 analog values “In1” and “In2”, if “In1” is less than “In2”, the digital output “Out” will be activated.

Example:



If R1.POTI1 is less than “55” The digital memory “MyMemory” will be activated, otherwise it will be deactivated (Zero).

### Split

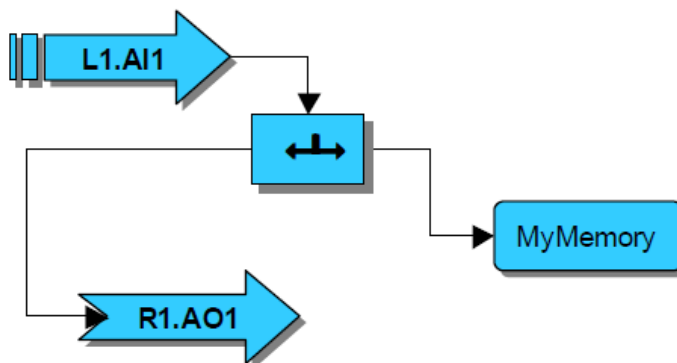
Symbol:



Data type: In (Analog Input), Out1, Out2 (Analog Outputs)

Function: This function splits the data into two paths. Both of the outputs have the same analog value as the input.

Example:



The input signal L1.AI1 will be simultaneously transmitted to outputs R1.AO1 and analog memory "MyMemory".

#### Flow: Counter

The following menu is used for counter functions. Choose Flow/Counter from the menu to show the following window:

The screenshot shows a dialog box titled "Insert a counter...". It contains a section "Select a counter function:" with five radio button options: "count up" (selected), "count down", "set counter value", "count up with limit", and "count down with limit". Below this is a text field labeled "Memory name, counter name:". At the bottom are three buttons: "Analog", "Cancel", and "OK".

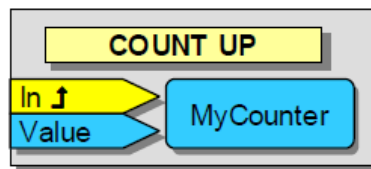
To select the analog memory that you will use in counter functions, you have two choices:

1. Write the name of the memory in the [Memory name, counter name] field if this is the first time you use this memory.

- Click on “Analog” button at the lower left corner to choose from existing analog memories in your program.  
After you choose your analog memory, select which function you will use.

**Count up**

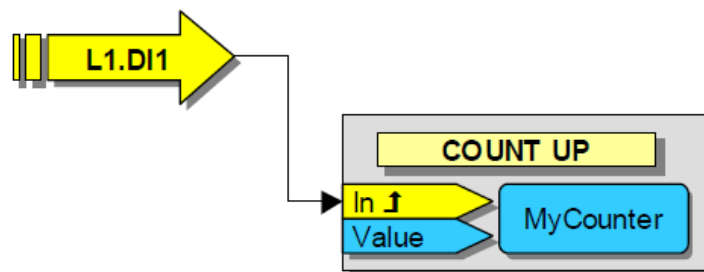
*Symbol:*



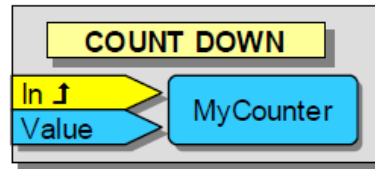
*Data type:* In (Digital Input), Value (Analog Input)

*Function:* When input “In” reads a rising edge, this function adds the value of input “Value” to the value of the analog memory “MyCounter”. The use of Input “Value” is optional. If the input “Value” remains unused, the value 1.0 will be added every time by default.

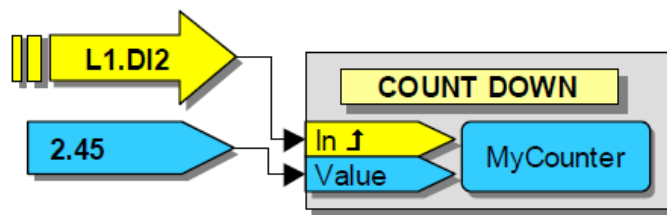
*Example:*



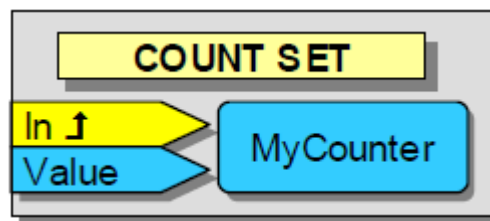
Every time digital input L1.DI1 detects a rising edge, the analog memory “MyCounter” will be increased by one.

**Count down***Symbol:**Data type:* In (Digital Input), Value (Analog Input)

*Function:* When input “In” reads a rising edge, this function subtracts the value of input “Value” from the value of the analog memory “MyCounter”. The use of Input “Value” is optional. If the input “Value” remains unused, the value 1.0 will be subtracted every time by default.

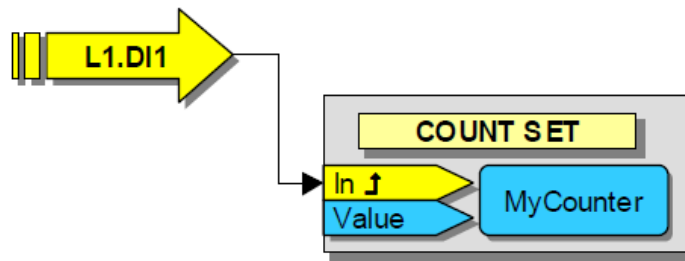
*Example:*

Every time digital input L1.DI2 detects a rising edge, the analog memory “MyCounter” will be decreased by 2.45.

**Set counter value***Symbol:**Data type:* In (Digital Input), Value (Analog Input)

*Function:* When input “In” reads a rising edge, the analog memory “MyCounter” will be set to the value of analog Input “Value” which is optional, If the input “Value” remains unused, the value 0.0 will be set.

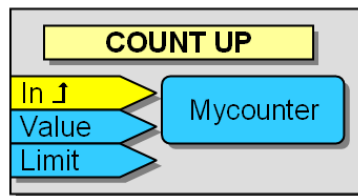
Example:



Every time digital input L1.DI1 detects a rising edge, the analog memory “MyCounter” will be reset to Zero.

### Count up with limit

Symbol:

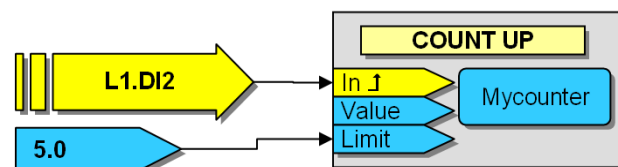


Data type: In (Digital Input), Value, Limit (Analog Inputs)

Function: When input “In” reads a rising edge, this function adds the value of input “Value” to the value of the analog memory “MyCounter”. The use of Input “Value” is optional. If the input “Value” remains unused, the value 1.0 will be added every time.

This process can be repeated as long as the “Limit” for the analog input is reached. If the Limit stays unused, no limit will be set.

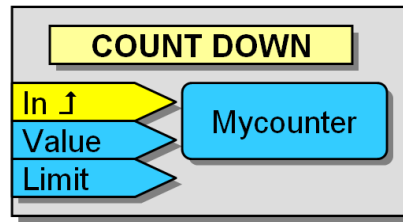
Example:



With every rising edge at digital input L1.DI2 the value 1 will be added to “MyCounter”, as long as “MyCounter” reaches the value 5.

### Count down with limit

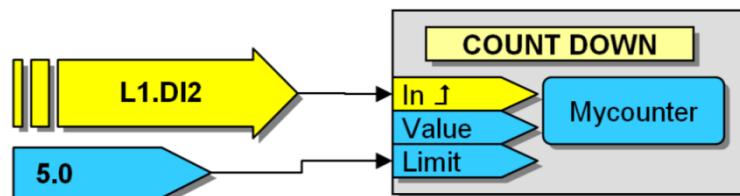
Symbol:



*Data type:* In (Digital Input), Value, Limit (Analog Inputs)

*Function:* When input “In” reads a rising edge, this function subtracts the value of input “Value” from the value of the analog memory “MyCounter”. The use of Input “Value” is optional. If the input “Value” remains unused, the value 1.0 will be subtracted every time. This process can be repeated as long as the “Limit” for the analog input is reached. If the Limit stays unused, no limit will be set.

Example:

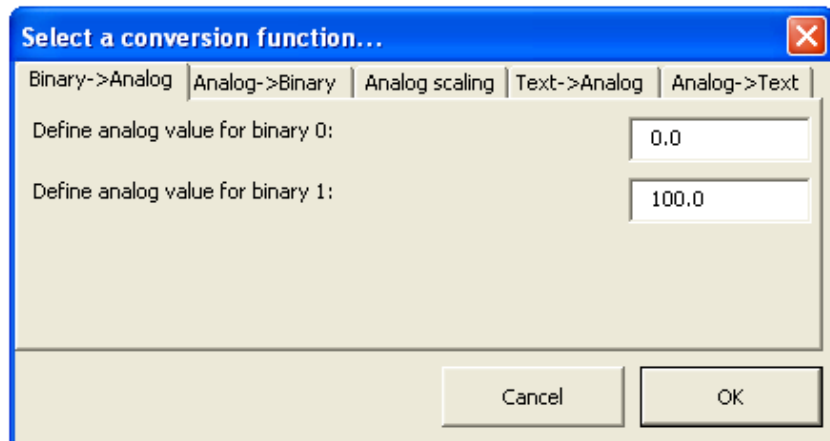


With every rising edge at digital input L1.DI2 the value 1 will be subtracted from analog memory “MyCounter”, as long as “MyCounter” reaches the value 5.

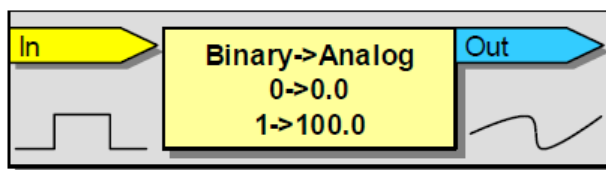


**Flow: Conversion**

This item deals with operations which can be used for the conversion of data. Choose Flow/Conversion from the menu, the following window will appear:

**Binary -> Analog**

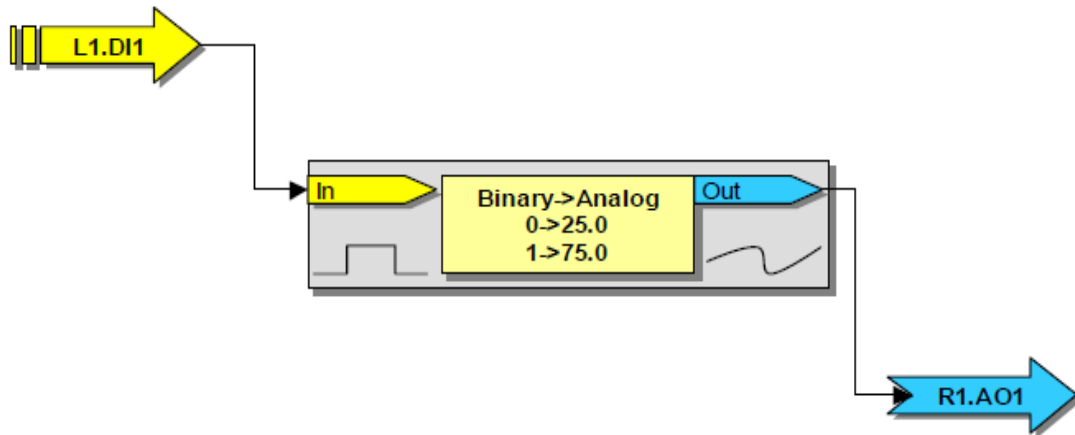
*Symbol:*



*Data type:* In (Digital Input), Value (Analog Output)

*Function:* This function converts a binary value to an analog value. For this you can select the analog values, which represent binary status 0 and 1, in the previous window write the analog value you want for a binary value zero in the first field, write the analog value you want for a binary value one in the second field, then press OK.

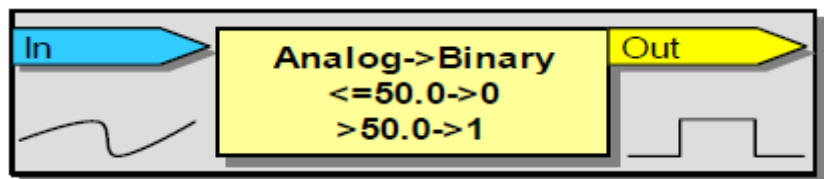
Example:



If digital input L1.DI1 receives no signal, analog output R1.AO1 is at 25. If the digital input receives a signal, the analog output will be at 75.

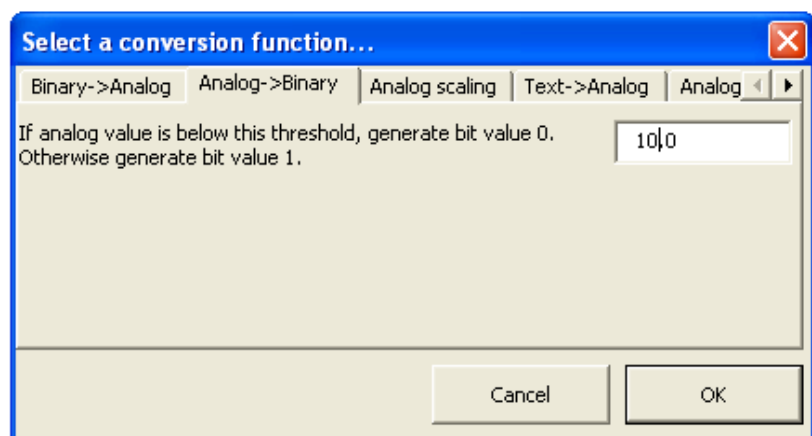
### Analog -> Binary

*Symbol*

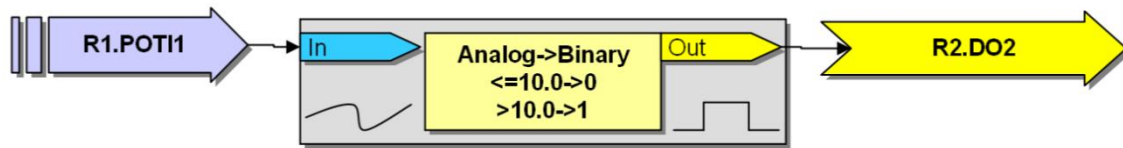


*Data type:* In (Analog Input), Value (Digital Output)

*Function:* This function converts an analog value to a binary value. For this you can select the analog value which represents binary status 0, in the next window write the analog value you want for a binary value 0 in the field, other analog value will be converted to a binary value 1.



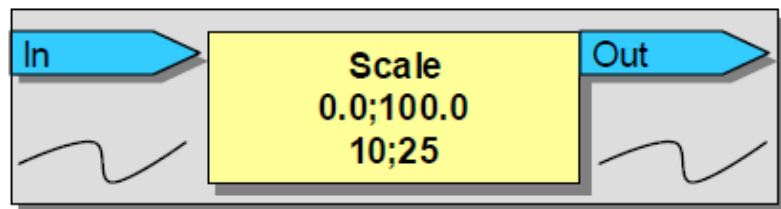
Example:



If the value set at potentiometer R1.POT1 is below or equal 10.0, digital output R2.DO2 will be switched off. If the value is higher than 10%, the digital output will be switched on.

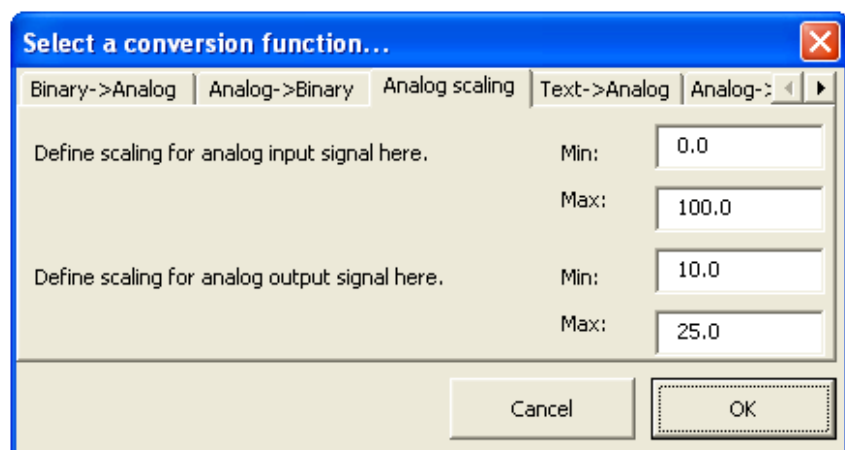
### Analog scaling

Symbol:

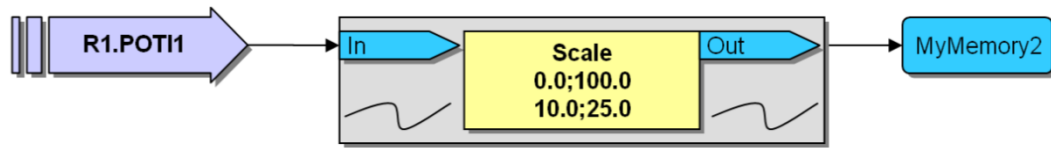


*Data type:* In (Analog Input), Out (Analog Output)

*Function:* This function converts the analog input signal to the analog output signal with a different output range, the first field "Min:" is for the minimum value for the input, the second "Max:" is for the maximum value for the input, the third is for the minimum value for the output, and the last field is for the maximum value for the output.



Example:



The output “MyMemory2” will be directly proportional to analog input R1.POTI1.

Note: analog input range is (0-100), “MyMemory2” range is (10-25).

**Flow: Tables + Interpolation**

This item deals with operations which can be used for the conversion of data in shape of tables plus processing of the second order polynomial equation. Choose [Flow/Tables + Interpolation] from the menu, the following window will appear:

Select a table or interpolation function...

Binary->analog table | Binary->text table | Analog->analog table | Analog->text table | Polynom 2nd order

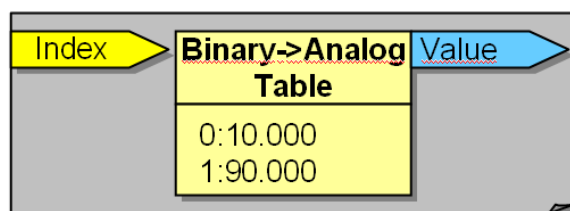
Define the analog table entry for binary index 0: 0.000

Define the analog table entry for binary index 1: 100.000

Cancel OK

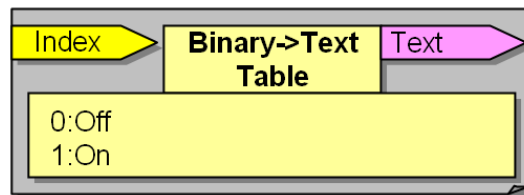
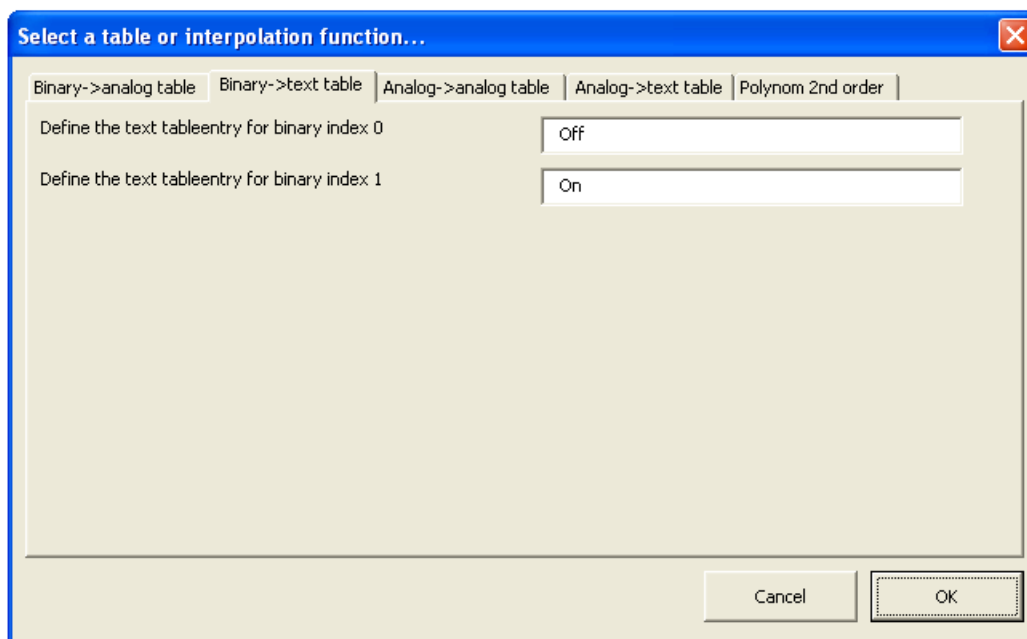
**Binary -> analog table**

Symbol:

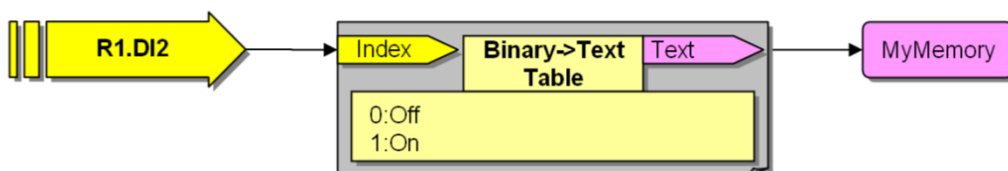


*Data type:* Index (Digital Input), Value (Analog Output)

*Function:* This function converts a binary index to an analog value. For this you can select the analog values, which represent binary index 0 and 1, in the previous window write the analog value you want for a binary index zero in the first field, write the analog value you want for a binary index one in the second field, then press OK.

**Binary -> text table***Symbol:**Data type:* Index (Digital Input), Text (Text Output)

*Function:* This function converts a binary index into text. For this you can select the text table entry, which represent binary index 0 and 1, in the previous window write the text you want for a binary index zero in the 1<sup>st</sup> field, write the text you want for a binary index one in the 2<sup>nd</sup> field, then press Ok.

*Example:*

If the digital input R1.DI2 has the index zero, the text string "Off" will be transmitted to text memory "MyMemory", If R1.DI2 has the index one, the text string "On" will be transmitted to text memory "MyMemory".

**Analog -> analog table***Symbol:*

Index	Analog->Analog Table	Value
	0:10.000	
	1:15.250	
	2:28.000	
	3:65.000	
	4:82.000	
	5:120.000	
	6:0.000	
	7:0.000	
	8:0.000	
	9:0.000	
	10:0.000	
	11:0.000	
	12:0.000	
	13:0.000	
	14:0.000	
	15:0.000	

*Data type:* Index (Analog Input), Value (Analog Output)

Select a table or interpolation function...

Binary->analog table | Binary->text table | Analog->analog table | Analog->text table | Polynom 2nd order

Define 16 analog table entries for the analog indices 0..15:

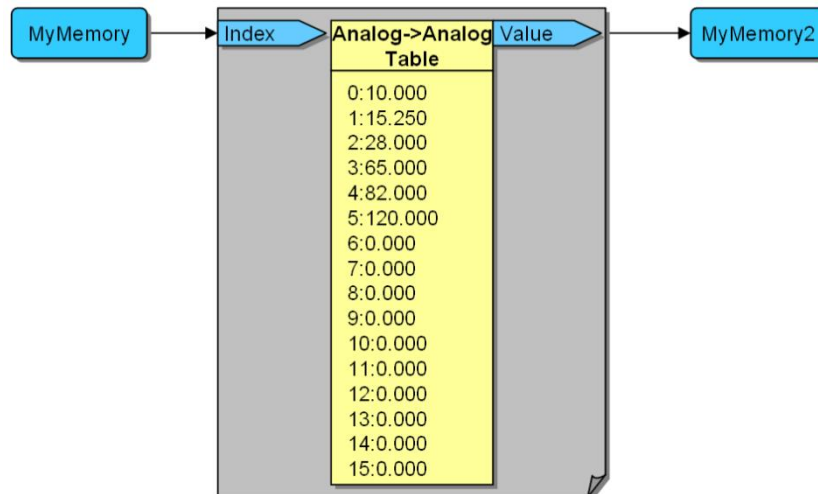
0: 10.000	4: 82.000	8: 0.000	12: 0.000
1: 15.250	5: 120.000	9: 0.000	13: 0.000
2: 28.000	6: 0.000	10: 0.000	14: 0.000
3: 65.000	7: 0.000	11: 0.000	15: 0.000

Cancel OK

*Function:* This function converts 16 analog indexes (From 0 to 15) to analog values in shape of table. For this you can write each table entry analog value, which represent each analog index input, in the previous window write the table entry analog value you want for an analog index zero in the first field, write

the table entry analog value you want for an analog index one in the second field and so on for many times you want up to 16 times, then press OK.

Example:



If the analog memory input “MyMemory” has the index zero, the analog value “10” will be transmitted to analog memory “MyMemory2”, If analog memory input “MyMemory” has the index one, the analog value “15.25” will be transmitted to analog memory “MyMemory2” ...etc.



**Analog -> text table***Symbol:*

The diagram shows a rectangular symbol with three ports at the top: 'Index' (blue arrow pointing right), 'Analog->Text Table' (yellow rectangle), and 'Text' (pink arrow pointing right). Below the ports is a large yellow rectangular area containing a list of 16 entries, each with an index number followed by a text description:

- 0: No Alarm
- 1: Alarm Fan1
- 2: Alarm Fan2
- 3: Alarm Fan3
- 4: Alarm Pump1
- 5: Alarm Pump2
- 6: Alarm Valve1
- 7: Alarm Valve2
- 8:
- 9:
- 10:
- 11:
- 12:
- 13:
- 14:
- 15:

*Data type:* Index (Analog Input), Text (Text Output)

The screenshot shows a dialog box titled "Select a table or interpolation function...". It has five tabs: "Binary->analog table", "Binary->text table", "Analog->analog table", "Analog->text table" (which is selected), and "Polynom 2nd order". Below the tabs, the text "Define 16 text table entries for the analog indices 0..15:" is displayed. There are two columns of input fields for indices 0 through 15. The first column contains the following text entries:

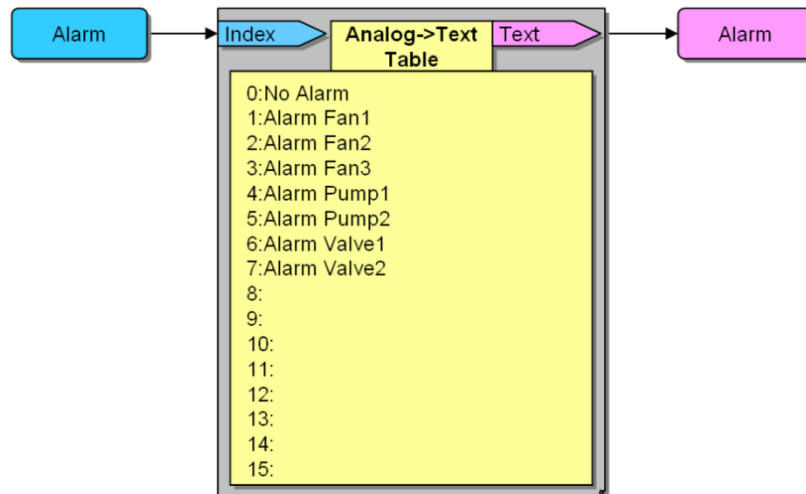
- 0: No Alarm
- 1: Alarm Fan1
- 2: Alarm Fan2
- 3: Alarm Fan3
- 4: Alarm Pump1
- 5: Alarm Pump2
- 6: Alarm Valve1
- 7: Alarm Valve2

The second column (indices 8-15) contains empty input fields. At the bottom right, there are "Cancel" and "OK" buttons.

*Function:* This function converts 16 analog indexes (From 0 to 15) to a text in shape of table. For this you can write each text table entry, which represent each analog index input, in the previous window write the text table entry you want for an analog index zero in the first field (No Alarm), write the text

table entry for an analog index one in the second field (Alarm Fan1) and so on for many times you want up to 16 times, then press OK.

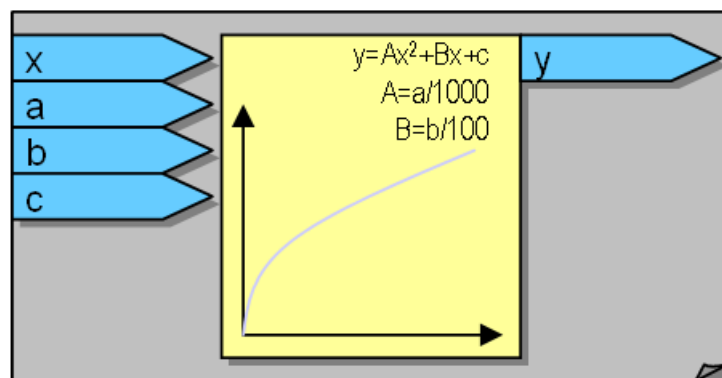
Example:



If the analog memory input “Alarm” has the index zero, the text “No Alarm” will be transmitted to a text memory “Alarm”, If analog memory input “Alarm” has the index one, the text “Alarm Fan1” will be transmitted to a text memory “Alarm” ...etc.

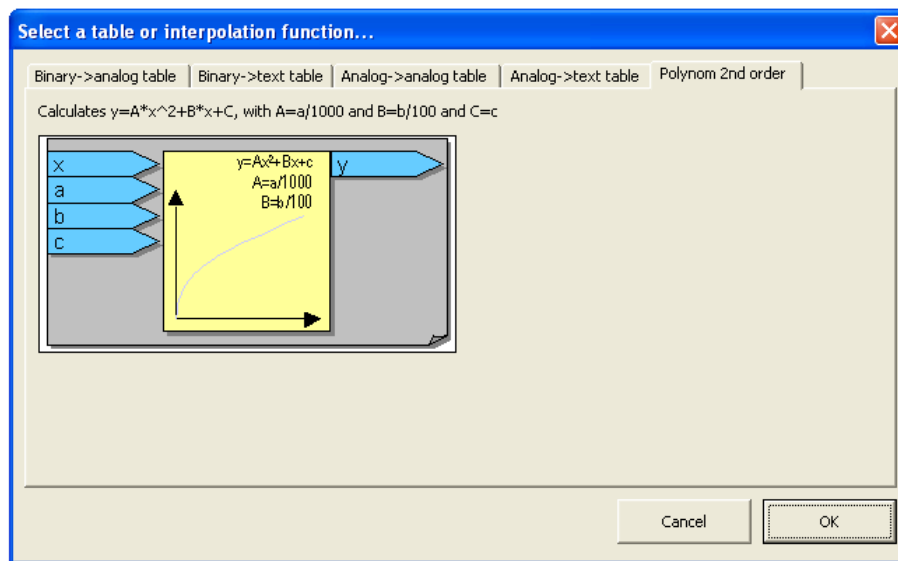
### Polynomial 2<sup>nd</sup> order

Symbol:



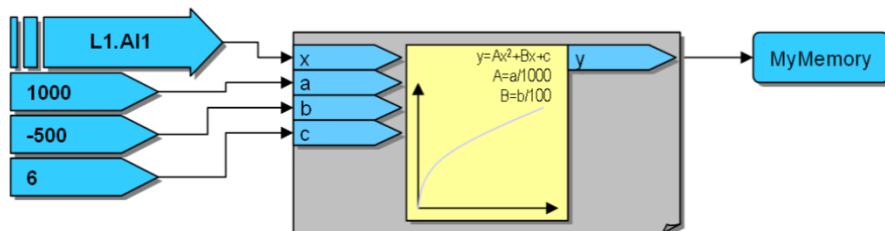
Data type:

- x Analog input
- a Analog input
- b Analog input
- c Analog input
- y Analog output



*Function:* This function is used to calculate the 2<sup>nd</sup> order polynomial equation which is like  $Y = Ax^2 + Bx + C$ , you should give the constants A, B, and C to equation and according to input X, the output Y calculates the equation.

Example:



In this example we will take an equation ( $Y = X^2 - 5X + 6$ ) as example, so from equation  $A = 1$ ,  $B = -5$ , and  $C = 6$ ; but we want to know the constant a, b, and c, so if you look at the symbol you will see that  $A = a/1000$ ,  $B = b/100$ , and  $C = c$ , for our equation  $a = 1000$ ,  $b = -500$ , and  $c = 6$ , so the analog output “MyMemory” will receive the result of equation according to the analog input x which represents L1.AI1.

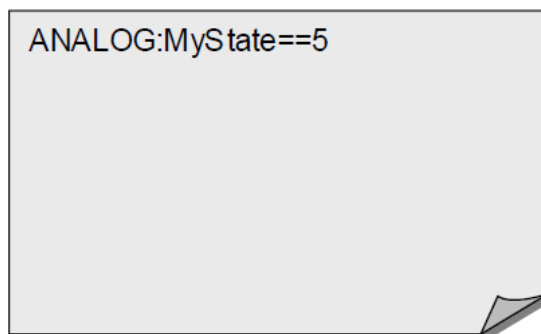
### Flow: State

This item deals with states and its use with the software. To add a new state, choose Flow/State. The following window will appear:

You can set an analog or digital memory name and define a constant value. Additionally you can choose whether you refer to an analog or to a binary memory.

### Analog state

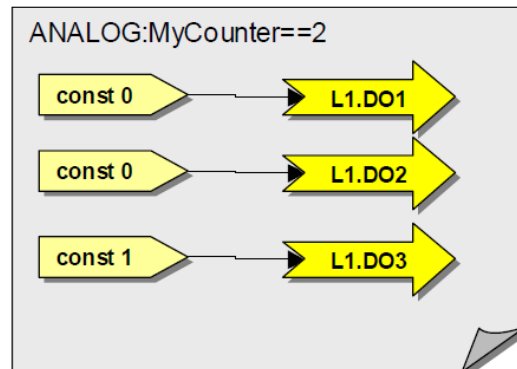
*Symbol:*



*Data type:* Analog

*Function:* Only objects, which are within this state frame, will be executed when required if the analog memory “MyState” has the value 5. The effect within SLS500 is that the program parts, which are not required at this time, will be bypassed. As a result the speed of the SLS500 program can be significantly increased!

Example:

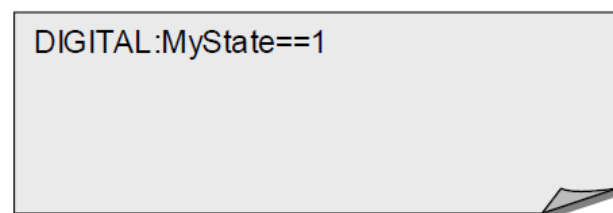


If the analog memory “MyCounter” has the value 2, digital outputs L1.DO1, L1.DO2 and L1.DO3 will be set to the values 0, 0, and 1. If “MyCounter” doesn’t equal 2, no command will be executed.

To do this, choose “Analog state” from previous window, then write “MyCounter” in the [Memory name] field or you can choose it from previously used memories in your application by clicking on “Analog” button, then write “2” in the [Memory value] field, then at the top right corner you will see some analog values comparison like “Equal, Not equal, Greater than, Greater than or equal, less than, and less than or equal” you can use any of them in case of analog state to compare between the value of the analog memory and the number in the [Memory value] field, in our example we use “Equal”.

### Binary state

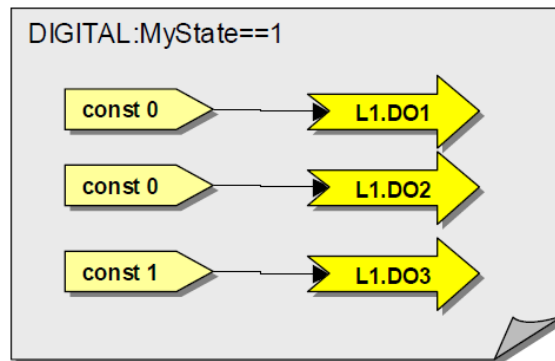
*Symbol:*



*Data type:* Binary

*Function:* Only objects, which are within the state frame, will be executed if the binary memory “MyState” has the value 0 or 1. The effect within SLS500 is that the parts of the program that are not required will be bypassed. As a result the speed of the SLS500 program can be significantly increased!

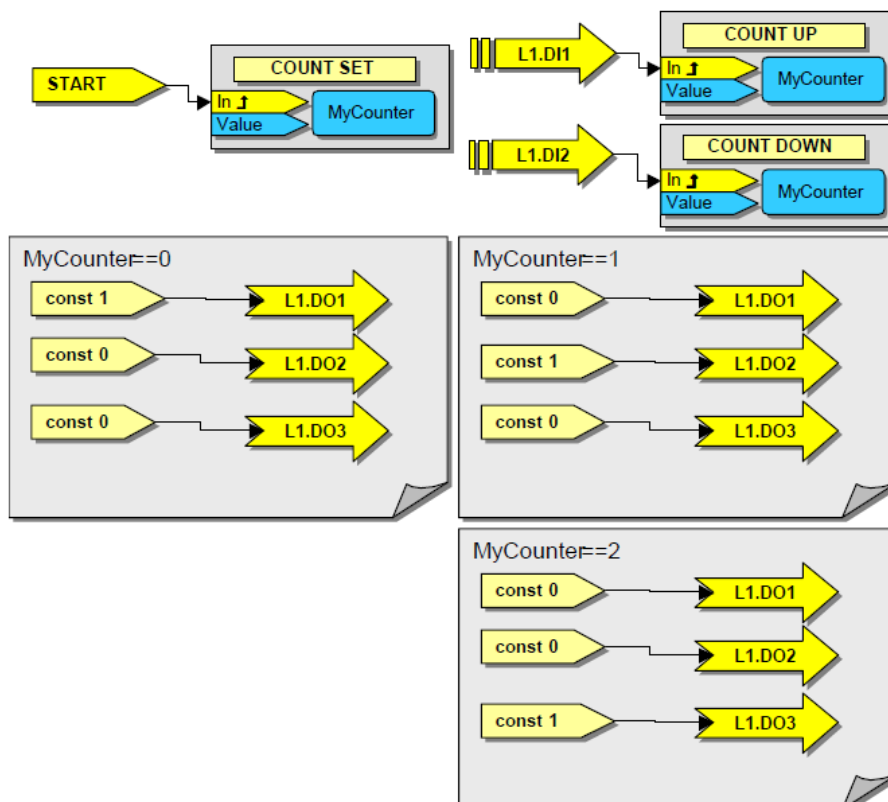
Example 1:



If the binary memory “MyState” has the value 1, digital outputs L1.DO1, L1.DO2 and L1.DO3 will be set to the values 0, 0, and 1. If “MyState” is equal 0, no command will be executed.

To do this, choose “Binary state” from previous window, then write “MyState” in the [Memory name] field or you can choose it from previous used memories in your program by click on “Bits” button, then write “1” in the [Memory value] field, then at the top right corner you will see some analog values comparison like “Equal, Not equal, Greater than, Greater than or equal, less than, and less than or equal” you can use only “Equal and Not equal” in case of binary state, in our example we use “Equal”.

Example 2:



If digital input L1.DI1 has a rising edge, the analog memory "MyCounter" will be increased by one. If digital input L1.DI2 has a rising edge the analog memory "MyCounter" will be reduced by one.

If "MyCounter" has the value 0, only output L1.DO1 will be energized.

If "MyCounter"=1, the output L1.DO2 will be energized.

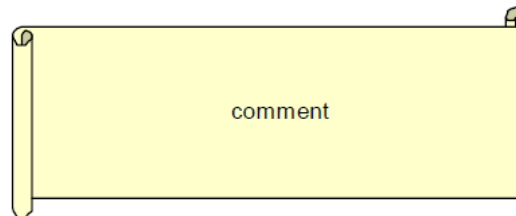
If "MyCounter"=2, the output L1.DO3 will be energized.

For all other values there will be no action.

**Flow: Comment**

You can use this operation to write anything as a comment anywhere in your program, however the SLS-500 compiler ignores this operation while compiling the program, also it doesn't download at the PLC.

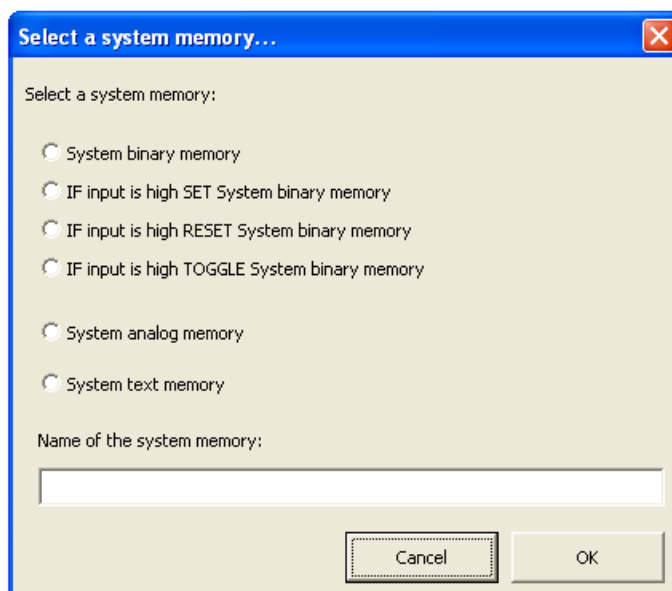
*Symbol:*



*Function:* Adds a comment to the current program page. Click on the text, and type in a desired comment! You can insert as many comments as desired. The comments will be ignored at program execution.

**Flow: System**

This operation deals with the system memory and its use in your software. The system memory can store values of the three data types. Every memory has an own name and is for a certain setting. To insert a system memory, select System from the menu. The following window will appear:



Type in a memory name corresponding to the system memory name list here to insert the memory into the program. Additionally you can choose whether to refer to an analog-, binary- or text-memory. To insert the memory click the OK button.



Example 1:



From the previous window choose “System analog memory”, then type the name of the memory “SIO\_RJ11\_BAUDRATE” in the field, then press OK button.

The constant value 19200 will be moved to the analog system memory “SIO\_RJ11\_BAUDRATE”.

Example 2:

### Incremental encoder

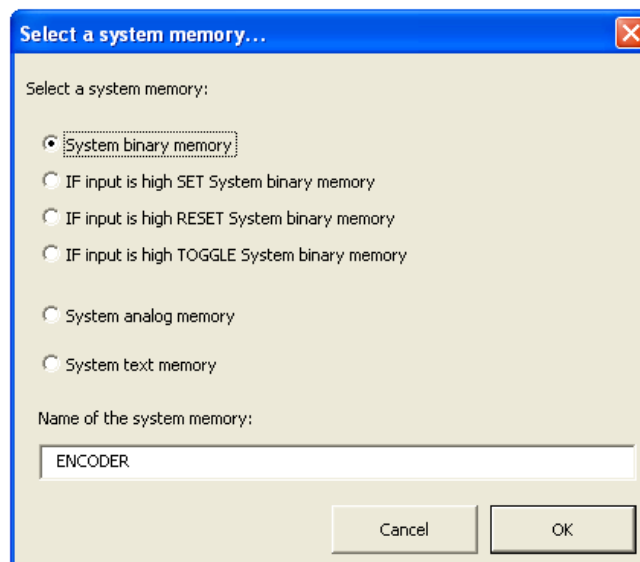
The Models HIQUEL-SLS500-R-24V and HIQUEL-SLS500-S-24V use digital inputs Di5-Di8 to receive the signal from the incremental encoders and they can each have two incremental encoders connected (Di5/6 – Di7/8), while Models HIQUEL-SLS500-CAN-R-24V and HIQUEL-SLS500-CAN-S-24V can each have one incremental encoder connected (Di5/6).

NOTE: The HIQUEL-SLS-500-Base controller can process 5000 edges per second maximum, combined Di's 5-8 (2 encoders), However, the extension SLS-500-ENC is used for connecting two incremental encoder with 500KHz maximum.

### Programming the incremental encoder for SLS-500-CAN series:

In SLS-500-Configurator, encoders are initialized on a binary system variable which is set to 1 using a bit-constant. This binary system-variable must only have the name "ENCODER" This is case sensitive.

Select System from the menu Flow > System. The following box will appear:



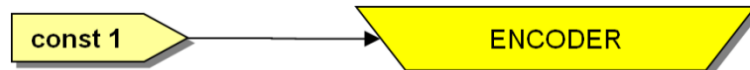
From the list, select "System binary memory" and in the field type "ENCODER" then click OK.

*Symbol:*

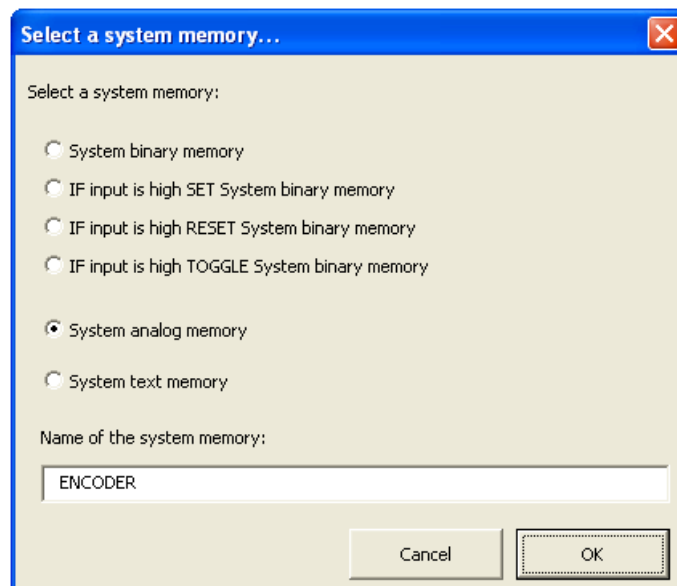


*Data type: Bit*

In the first, you must initialize the digital system memory "ENCODER" as in the next figure:



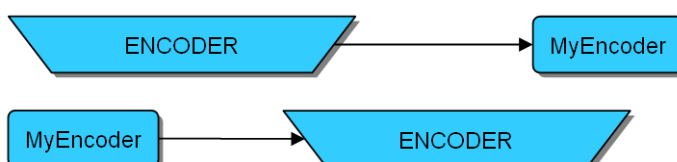
When the binary-system-variable is initialized, you must set the analog-system-variable "ENCODER" also to 0 to delete it. In contrast to "ENCODER" this is not a binary-system-variable but an analog-system-variable. The input is case sensitive. (ENCODER) Here they are set to 0:



From the list, select "System analog memory" and in the field type "ENCODER" then click OK.



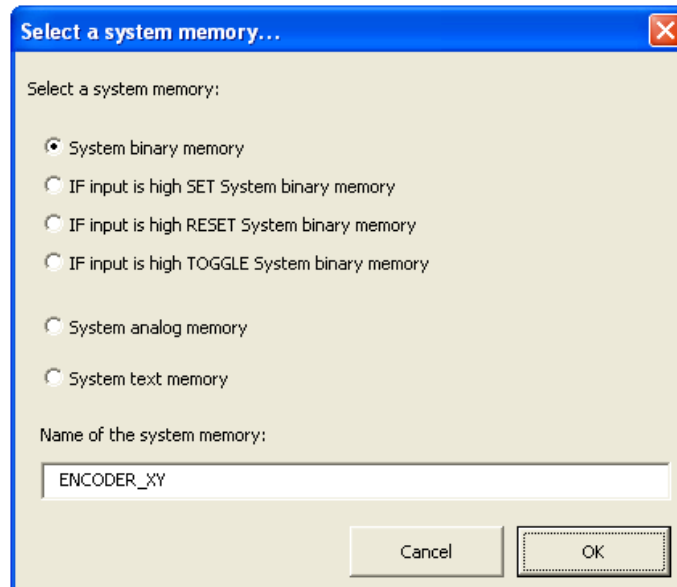
The value of incremental encoder could also be stored in an analog memory or can be set by analog memory.



### Programming the incremental encoder for SLS-500 series:

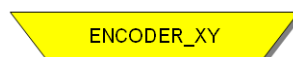
In SLS-500-Configurator, encoders are initialized on a binary system variable which is set to 1 using a bit-constant. This binary system-variable must only have the name "ENCODER\_XY" This is case sensitive.

Select System from the menu Flow > System. The following box will appear:



From the list select "System binary memory" and in the field type "ENCODER\_XY" then click OK.

*Symbol:*

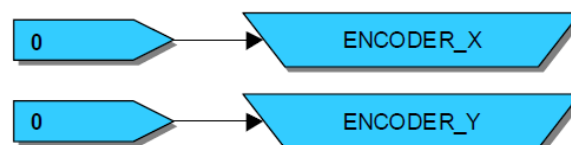


*Data type:* Bit

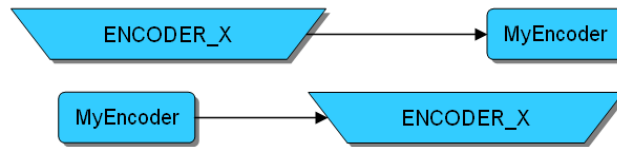
In the first, you must initialize the digital system memory "ENCODER\_XY" as in the next figure:



When the binary-system-variable is initialized, you must set the analog-system-variables "ENCODER\_X", and "ENCODER\_Y" also to 0 to delete them. The input is case sensitive. (ENCODER\_X), and (ENCODER\_Y) Here they are set to 0:



The value of incremental encoder could also be stored in an analog memory or can be set by analog memory.

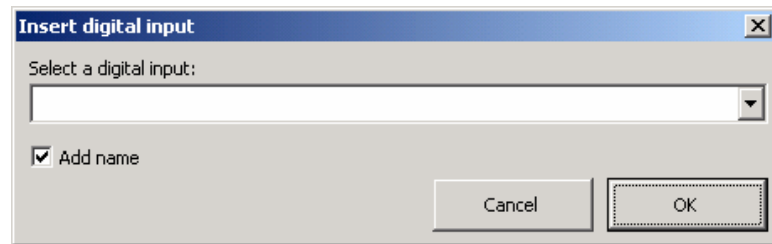


## I/O Menu

This menu contains all functions, which deals with the pasting of digital or analog inputs or outputs. The following subchapter will explain the use of inputs and outputs of SLS-500-Configurator.

### I/O: Digital inputs

Choose [I/O: Digital inputs] from the menu. The following window will be opened:



Choose a digital input by using the drop down menu and confirm by clicking OK. Check [Add name] to add the variable name to the symbol. If Add name is not active, just the digital input symbol with its address (L1.DI1) will be displayed.

Symbol without Add name:



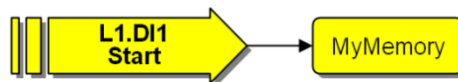
Symbol, Add name is active:



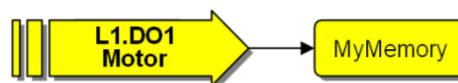
*Data type:* Bit

**IMPORTANT:** Before you can choose a digital input, you have to create a regular system configuration in page CONFIG first!

*Function:* This function adds a digital input. Digital inputs can be a starting point of connections. You will also find all digital outputs in the list to process the current status of the outputs, and in this case it is considered as an input.



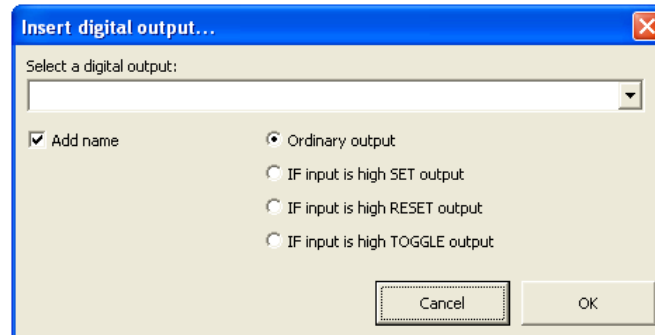
The current state of digital input L1.DI1 will be saved to the digital memory "MyMemory".



The current state of digital output L1.DO1 will be saved to the digital memory "MyMemory".

### I/O: Digital outputs

Choose [I/O: Digital outputs] from the menu. The following window will be opened:



Choose a digital output by using the drop down menu and confirm by clicking OK. Check [Add name] to add the variable name to the symbol. If Add name is not active, only the digital output symbol with its address (L1.DO1) will be displayed.

Additionally you can choose one of four functions for the output:

*Ordinary output:* The output always takes the value of the incoming connection.

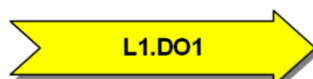
*SET output:* When the incoming connection is 1, the output will be set to High. When the connection returns to 0, the value of the output stays unchanged (High).

*RESET output:* When the incoming connection is 1, the output will be set to Low. If the connection is 0, the value of the output stays unchanged (Low).

*TOGGLE output:* When the incoming connection is 1 for the first time, the output will energize. When the incoming connection becomes 0, the output remains energized. When the incoming connection is 1 for the second time the output de-energizes. When the incoming connection becomes 0 for the second time the output remains de-energized, and so on..... (Can be used as a bi-stable function)

**IMPORTANT:** before you can choose a digital output, you have to create a regular system configuration in page CONFIG first!

Symbol without Add name:



Symbol, Add name is active:



Symbol for function SET:



Symbol for function RESET:



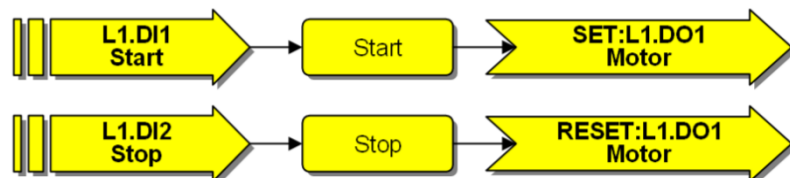
Symbol for function TOGGLE:



*Data type:* Bit

*Function:* This function adds a digital output. Digital outputs always display the end point of a connection. To get the current state of a digital output you have to choose the output from the list of the digital inputs.

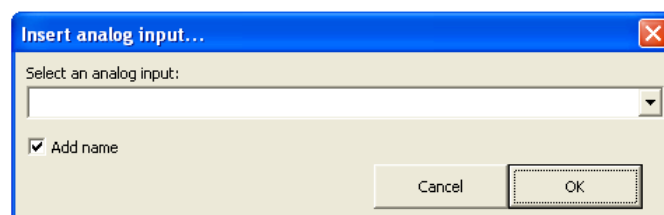
Example:



The current state of the digital input L1.DI1 is delivered to digital memory “Start” and then set the digital output L1.DO1 (Motor). The current state of the digital input L1.DI2 is delivered to digital memory “Stop” and then resets the digital output L1.DO1 (Motor).

### I/O: Analog inputs

Choose [I/O: Analog inputs] from the menu. The following window will be opened:





Choose an analog input by using the drop down menu and confirm by clicking OK. Check [Add name] to add the variable name to the symbol. If Add name is not active, just the analog input symbol with its address (L1.AI1) will be displayed.

**IMPORTANT** Before you can choose an analog input, you have to create a regular system configuration in page CONFIG first!

Symbol without Add name:



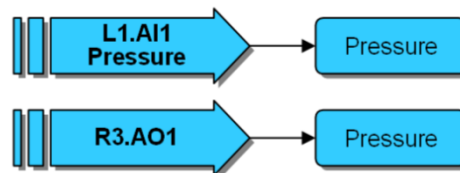
Symbol, Add name is active:



*Data type:* Analog

*Function:* This function adds an analog input. Analog inputs are always the beginning of a connection. All analog outputs are available as analog inputs as well. All analog signals are displayed between 0.000 and 100.000. As a result they display percentage value between 0% and 100%, independent of the true input/output value (0-10v/0-20mA) for example.

Examples:

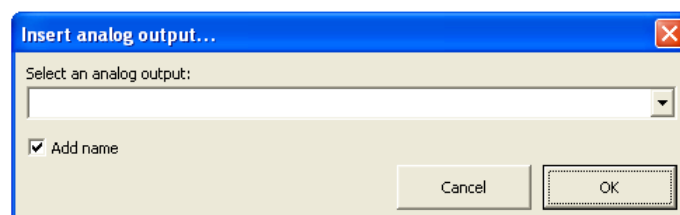


The current state of analog input L1.AI1 will be delivered to the analog memory "Pressure".

The current state of analog output R3.AO1 will be delivered to the analog memory "Pressure".

### I/O: Analog outputs

Choose [I/O: Analog outputs] from the menu. The following window will be opened:



Choose an analog output by using the drop down menu and confirm by clicking OK. Check [Add name] to add the variable

name to the symbol. If Add name is not active, just the analog output symbol with its address (R1.AO1) will be displayed.

**IMPORTANT:** Before you can choose an analog output, you have to create a regular system configuration in page CONFIG first!

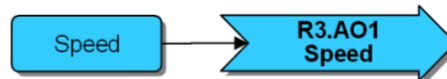
Symbol, Add name is active:



*Data type:* Analog

*Function:* This function adds an analog output. Analog outputs always represent the end of a connection. All analog outputs are available as analog inputs as well. All analog output signals are displayed between 0.000 and 100.000. As a result they display the percentage between 0% and 100%, independent of the true output value (0-10v/4-20mA) for example.

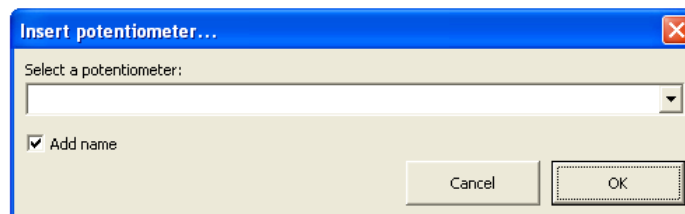
Example:



The current value of the analog memory “Speed” will be delivered to analog output R3.AO1.

### **I/O: Potentiometer:**

Choose [I/O: Potentiometer] from the menu. The following window will be opened:



Choose a potentiometer by using the drop down menu and confirm by clicking OK. Check [Add name] to add the variable name to the symbol. If Add name is not active, just the potentiometer symbol with its address (R1.POT11) will be displayed.

**IMPORTANT:** Before you can choose a potentiometer, you have to create a regular system configuration in page CONFIG first!

Symbol, Add name is active:



*Data type:* Analog

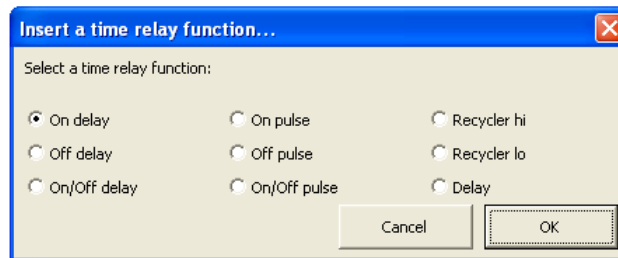
*Function:* This function adds a potentiometer. Potentiometers always represent the beginning of a connection. All analog potentiometer signals are displayed between 0.000 and 100.000. As a result they display the percentage between 0% and 100%.

## Objects Menu

In this menu you can find a series of extra functions that can be used with SLS-500-Configurator.

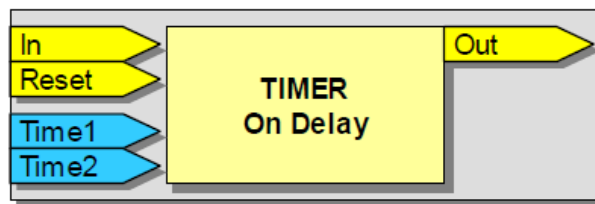
### Objects: timerelais

This item is special for inserting the different types of timers used in PLC such as on delay, off delay, on pulse, off pulse, and recycler timer, choose it from the menu to get the following window:



Choose your desired timer from the previous window, then click OK, you will get the following symbol according to the type of timer.

*Symbol:*



*Data type:*

In (Digital input), the chosen time function will be started if this input is active.

Reset (Digital input), as soon as this input is high, the output will be reset to 0.

Time1 (Analog input), the first time of the timer in seconds.

Time2 (Analog input), the second time of the timer in seconds.

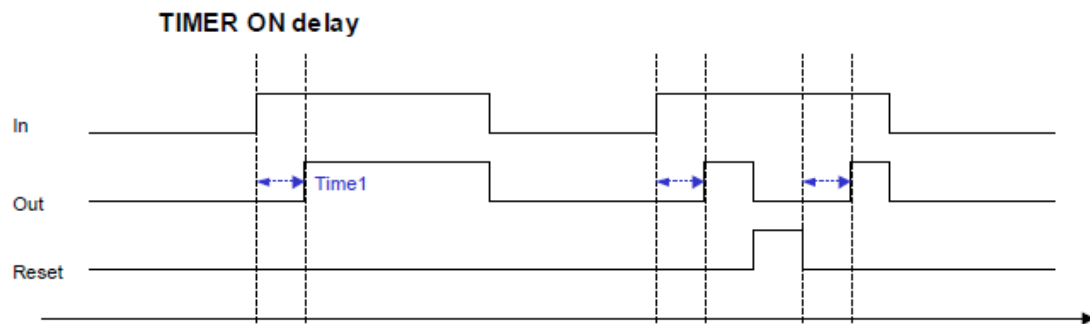
This time Input is only required with on/off delay, on/off pulse, Recycler high first and Recycler low first.

Out (Digital output), output of the time function. The output will change state in accordance with the time function

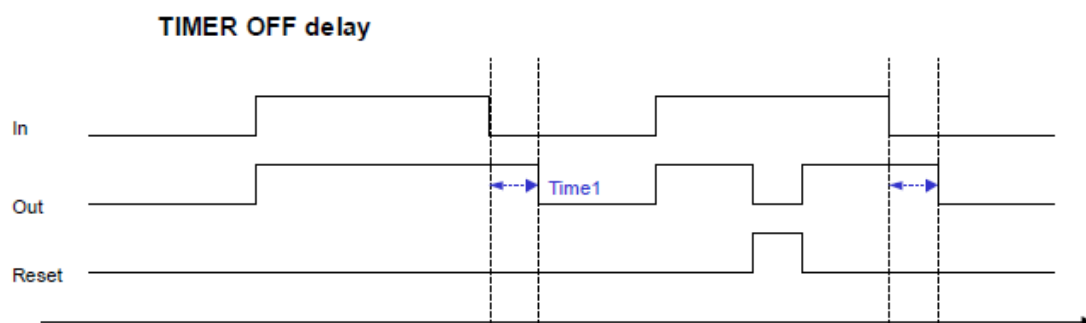
Time base: The time base of all timing functions is 100ms.

Here is the timing function diagram for each type of timer:

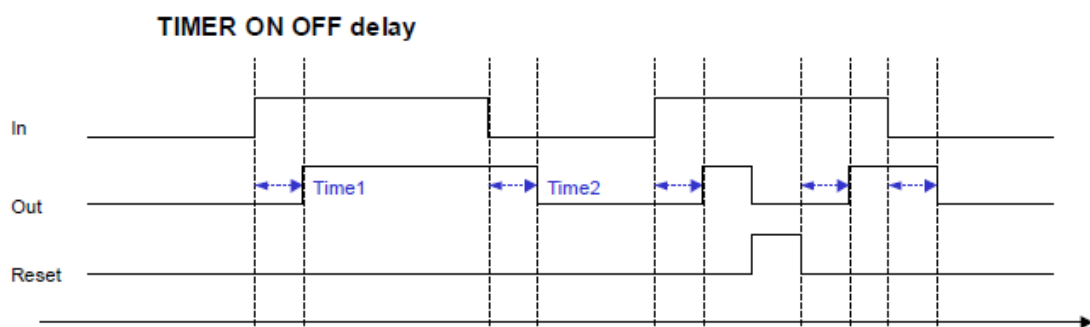
Timer on-delay:



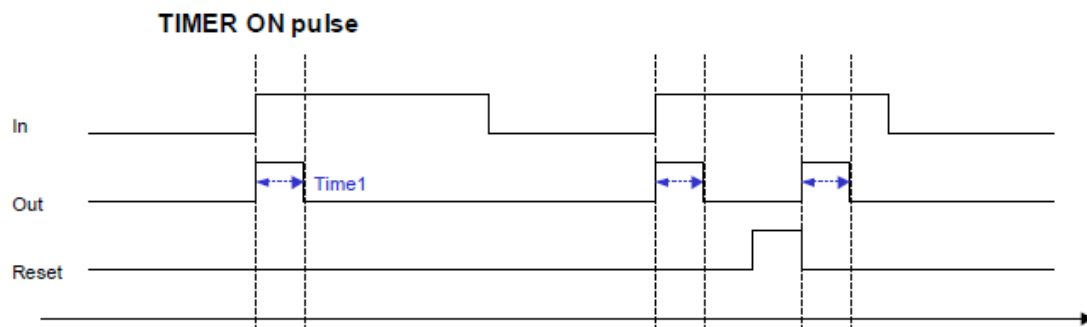
Timer off-delay:



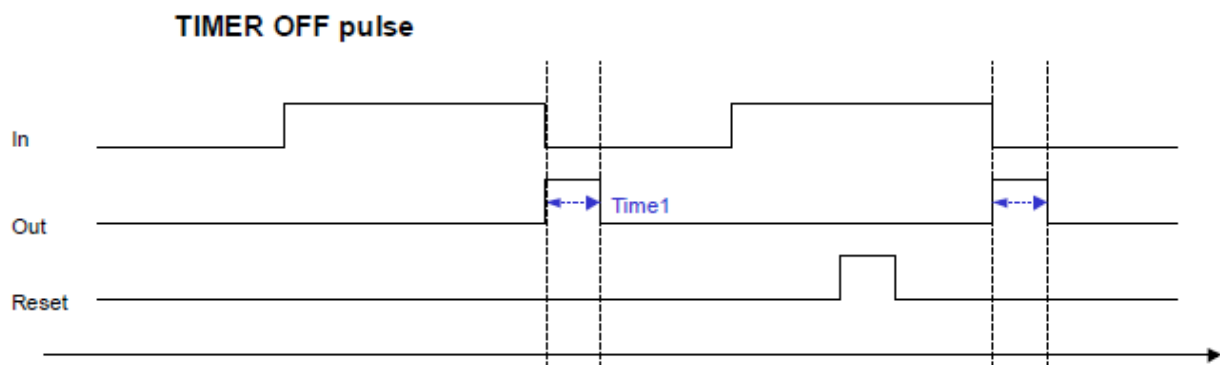
Timer on/off-delay:



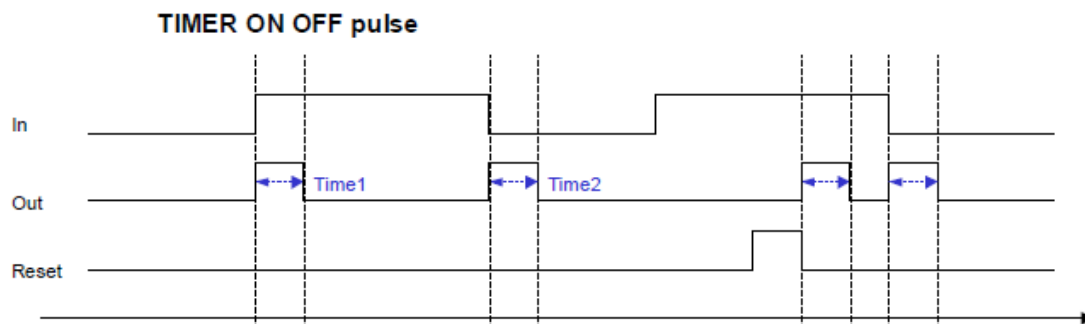
Timer on-pulse:



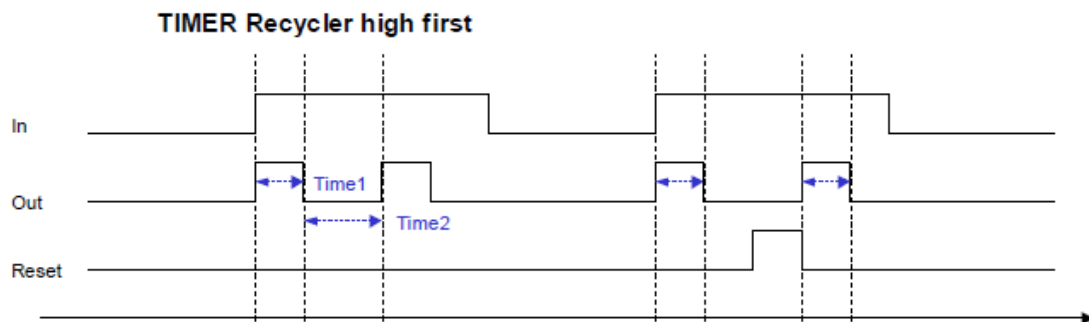
Timer off-pulse:



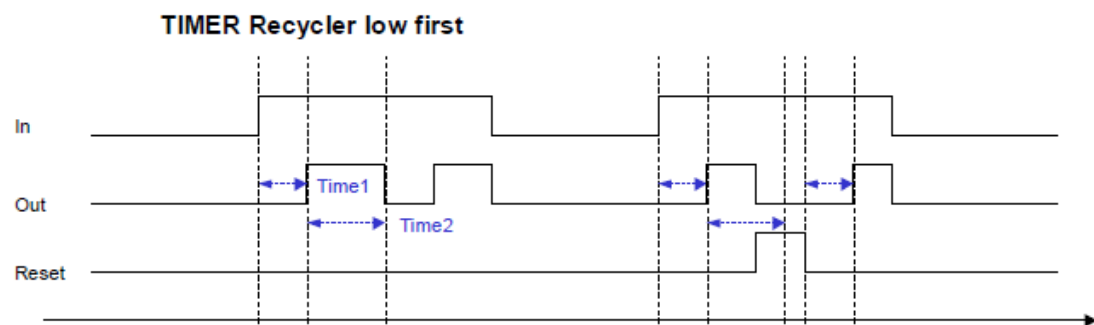
Timer on/off-pulse:



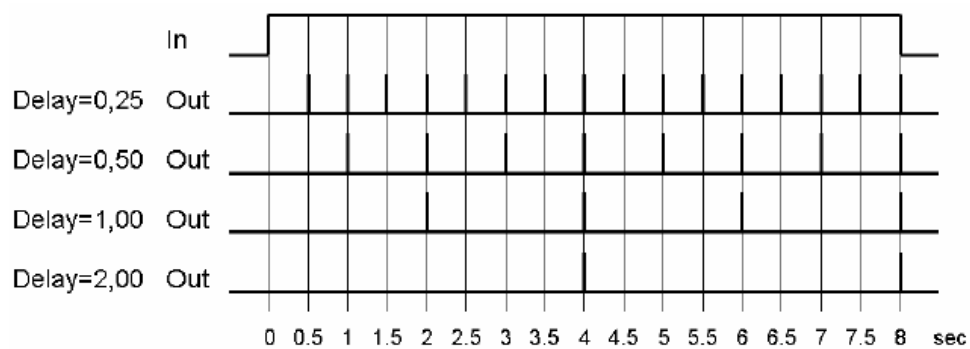
Timer recycler high:



Timer recycler low:



Timer delay:



Examples:



3 Sec after activation of digital input L1.DI4, digital output R2.DO1 will become active too.



Digital output R2.DO1 will be ON for 2 sec, and OFF for 1 sec, as long as digital input L1.DI4 is active.



**Objects: Realtime clock**

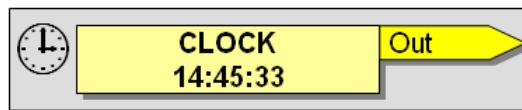
This item deals with the functions of the integrated real time clock used in PLC.

ADVICE: The real time clock is not available with all modules. Please take a look at the allocation dialog of the controller to see if the RTC is supported or not!

Choose it from the object menu to get the following window:

**Binary: Exact time HH:MM:SS**

*Symbol:*

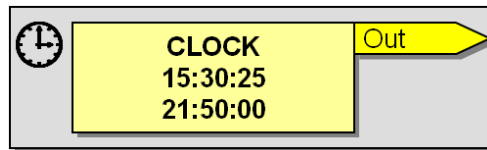


*Data type:* Out (Digital output)

*Input field:* Start time 24h Format HH:MM:SS

*Function:* This function compares the current time of the module with the selected time which you can insert in your program by writing it at “Start time:” field in the previous window. Every day when the selected time is equal to the real current time of the module, output “Out” will be activated for only 1 sec.

Note: there are other fields but this field “Start time:” and these all fields will be visible only according to your item selection. If you select another item, this field “Start time:” will be invisible and another one or two fields will be visible according to your selection as we will see in the next items.

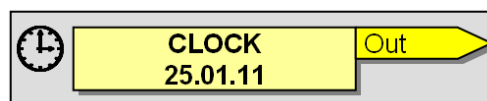
**Binary: Time range HH:MM:SS-HH:MM:SS***Symbol:**Data type:* Out (Digital output)*Input field:*

Start time 24h Format HH:MM:SS

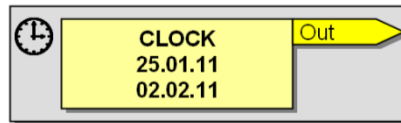
End time 24h Format HH:MM:SS

*Function:* This function compares the current time of the module with the selected time range which you can insert in your program by writing it at “Start time:”, and “End time:” fields in the previous window. Every day if the real current time of the module is between start time and end time, output “Out” will be activated, otherwise it will be zero, in the previous figure, digital output “Out” will be activated everyday from 03:30:25 pm until 09:50:00 pm.

Note: when you use items containing date or time or week range, be sure to make the end time / date / weekday is bigger than the start time / date / weekday, otherwise it will not work. Put in your consideration that MONDAY is the first day of the week and SUNDAY is the last day of the week.

**Binary: Exact date DD.MM.YY***Symbol:**Data type:* Out (Digital output)*Input field:* Start date day Format DD.MM.YY

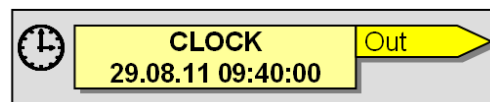
*Function:* This function compares the current date of the module with the selected date which you can insert in your program by writing it at “Start date:” field in the previous window. As long as the dates are equal, output “Out” will be activated for this day.

**Binary: Date range DD:MM:YY-DD.MM.YY***Symbol:**Data type:* Out (Digital output)*Input field:*

Start date day format DD.MM.YY

End date day format DD.MM.YY

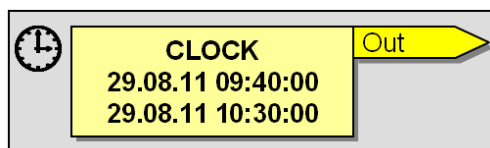
*Function:* This function compares the current date of the module with the selected date range which you can insert in your program by writing it at “Start date:”, and “End date:” fields in the previous window. As long as the current date is within the adjusted range, output “Out” will be activated; otherwise it will be zero, in the previous figure, digital output “Out” will be activated from day 25 Jan 2011 until day 2 Feb 2011.

**Binary: Exact date & time DD.MM.YY HH:MM:SS***Symbol:**Data type:* Out (Digital output)*Input field:*

Start date format DD.MM.YY

Start time 24h format HH:MM:SS

*Function:* This function compares the current date & time of the module with the selected date & time which you insert in your program. If both are the same, output “Out” will be activated.

**Binary: Date & time range DD.MM.YY HH:MM:SS-DD.MM.YY HH:MM:SS***Symbol:*

*Data type:* Out (Digital output)

*Input field:*

Start date day format DD.MM.YY

Start time 24h format HH:MM:SS

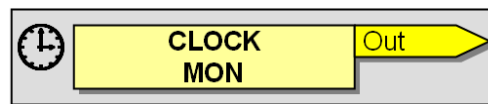
End date day format DD.MM.YY

End time 24h Format HH:MM:SS

*Function:* This function compares the current date & time of the module with the selected date & time range which you insert in your program. If the real current date and time is within the adjusted range, output “Out” will be activated.

**Binary: Exact day of week WWW**

*Symbol:*



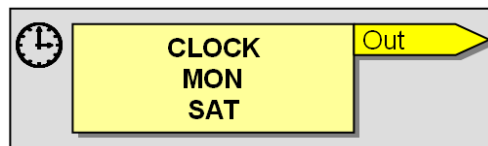
*Data type:* Out (Digital output)

*Input field:* Start day format WWW (SAT, SUN, MON, TUE, WED, THU, FRI), write it in the correct shown format with capital letters.

*Function:* This function compares the current day of week of the module with the selected day which you insert in your program. If the days match, output “Out” will be activated.

**Binary: Day of week range WWW-WWW**

*Symbol:*



*Data type:* Out (Digital output)

*Input field:*

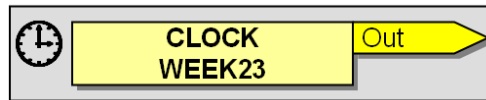
Start day format WWW (SAT, SUN, MON, TUE, WED, THU, FRI), write it in the correct shown format with capital letters.

End day format WWW (SAT, SUN, MON, TUE, WED, THU, FRI), write it in the correct shown format with capital letters.

*Function:* This function compares the current day of week of the module with the selected weekday range which you insert in your program. If the current day is within the specified range, output “Out” will be activated.

**Binary: Exact week WEEKxx**

*Symbol:*



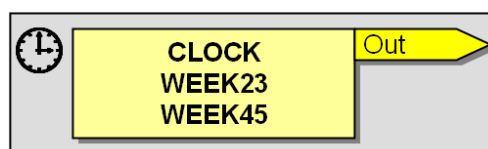
*Data type:* Out (Digital output)

*Input field:* Start week format WEEKxx (xx: No. of week in calendar).

*Function:* This function compares the current week number of the module with the selected week number which you insert in your program. If the weeks match, output “Out” will be activated.

**Binary: Week range WEEKxx-WEEKxx**

*Symbol:*



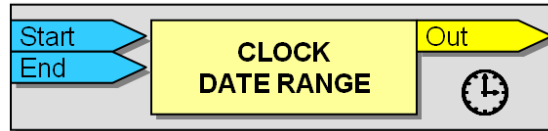
*Data type:* Out (Digital output)

*Input field:*

Start week format WEEKxx (xx: No. of week in calendar).

End week format WEEKxx (xx: No. of week in calendar).

*Function:* This function compares the current week number of the module with the selected week number range which you insert in your program. If the current week is within the specified range, output “Out” will be activated.

**Binary: Analog date range***Symbol:**Data type:*

Start Analog input, format DD.MM.YY

End Analog input, format DD.MM.YY

Out Digital output

*Function:* This function compares the current date of the module with the date range which inserting in the analog inputs “Start” and “End” using analog constants or variable analog memories. As long as the current date is within the adjusted range, output “Out” will be activated, otherwise it will be zero.

*Example:*

Digital output L1.DO2 will be activated from 29.08.2011 at 00:00 am until 31.08.2011 at 11:59:59 pm, otherwise it will be zero.

**Binary: Analog time range***Symbol:**Data type:*

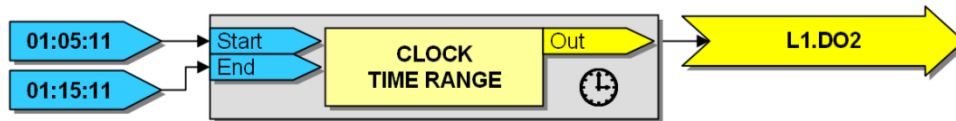
Start Analog input, format HH:MM:SS

End Analog input, format HH:MM:SS

Out Digital output

*Function:* This function compares the current time of the module with the time range which inserting in the analog inputs “Start” and “End” using analog constants or variable analog memories. As long as the current time is within the adjusted range, output “Out” will be activated, otherwise it will be zero.

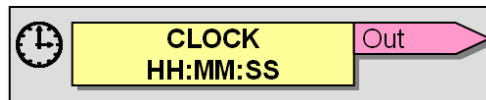
Example:



Digital output L1.DO2 will be activated everyday from 01:05:11 am to 01:15:11 am, otherwise it will be zero.

**Text: Time**

*Symbol:*

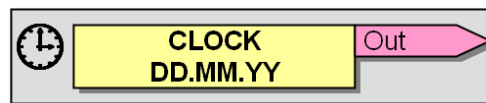


*Data type:* Out (Text output)

*Function:* This function delivers the current time as 8 characters text with a 24h format **HH:MM:SS**.

**Text: Date**

*Symbol:*

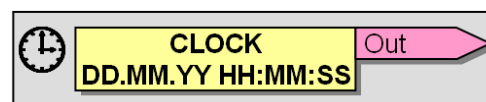


*Data type:* Out (Text output)

*Function:* This function delivers the current Date as 8 characters text with a **DD.MM.YY** format.

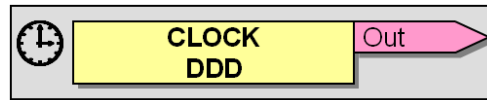
**Text: Date+Time**

*Symbol:*

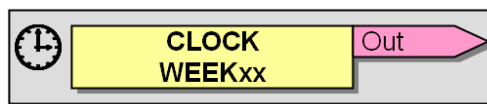


*Data type:* Out (Text output)

*Function:* This function delivers the current Date and time as a 17 character text with a **DD.MM.YY HH:MM:SS** format.

**Text: Day of week***Symbol:**Data type:* Out (Text output)

*Function:* This function delivers the current weekday as a 3 characters text, the weekdays have the English day abbreviation: MON, TUE, WED, THU, FRI, SAT, SUN.

**Text: Week of year***Symbol:**Data type:* Out (Text output)

*Function:* This function delivers the current calendar week as 6 characters text with a **WEEKxx** format, where **xx** represents the number of week in year.

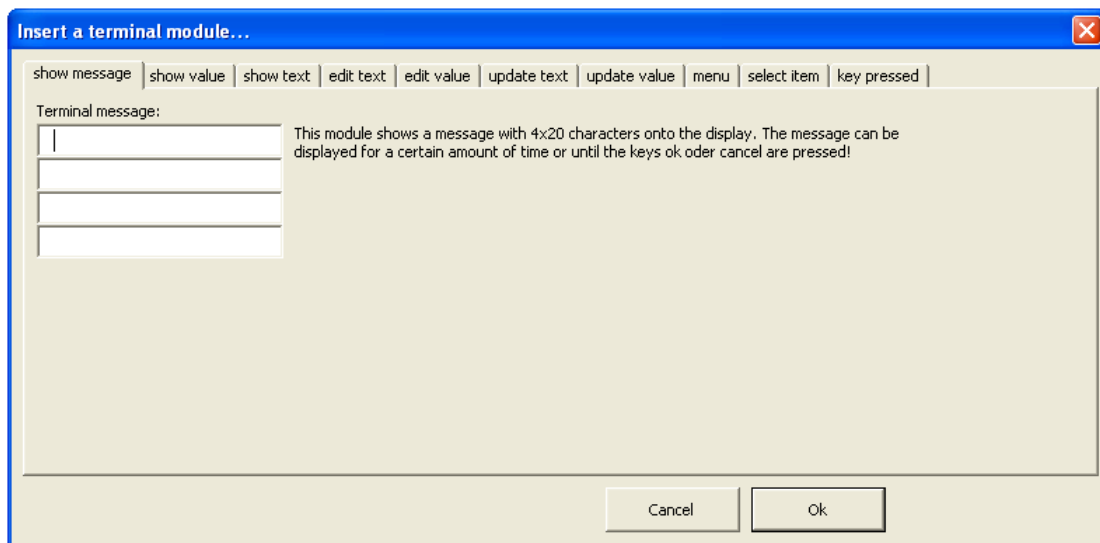


**Objects: Terminal**

This item deals with the functions of the HIQUEL text screen (TERM4).

SLS-500-Configurator supports HIQUEL-TERM4, the terminal is addressed to suit SLS500, but first you have to choose it for your configuration.

Choose Objects/Terminal from the menu to get to the following window:

**TAB: Show message**

This function is used to show a message on the TERM4 screen, and it contains 4 fields used to write your message, each field occupies a line in TERM4, and you can write up to 20 characters in each field (line).

*Symbol:*



*Data type:*

Show: Digital input

Time: Analog input (optional)

Ready: Digital output (optional)

Ok: Digital output (optional)

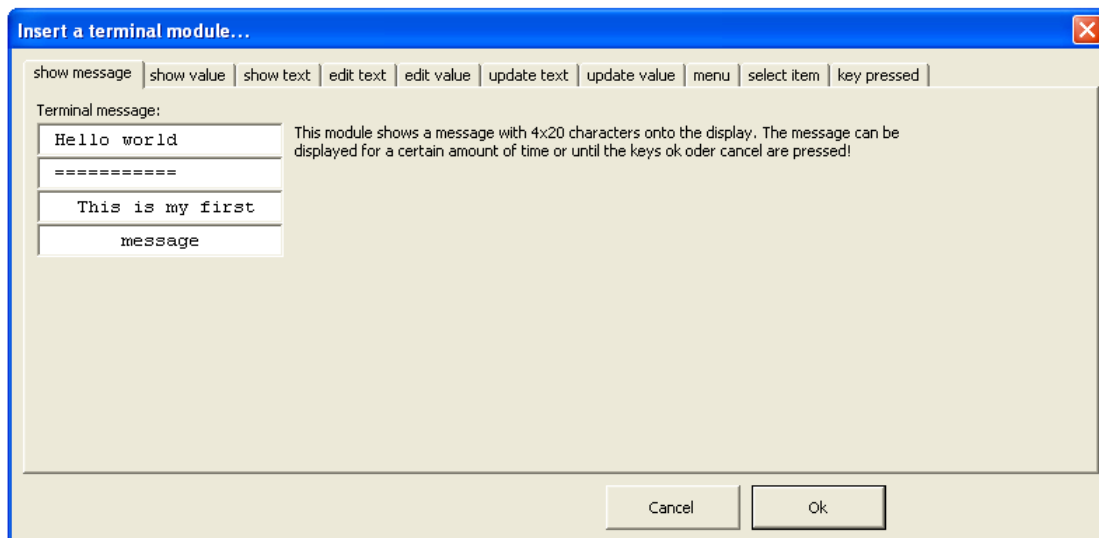
Cancel: Digital output (optional)

Time: Digital output (optional)

*Function:* With this function, if the digital input “Show” has a rising edge, the stored text will be displayed on the terminal. Additionally you can set a display time in seconds at the analog input “Time”. If this time runs out, digital output “Time” will be activated. If the input “Time” stays unconnected, the time function will be ignored. Output “Ready” will be active, when the complete screen set-up is finished. If the user presses the “OK” button that is existed on TERM4, output “OK” will be activated. If the user presses the “Cancel” button that is existed on TERM4, output “Cancel” will be activated.

*Example:*

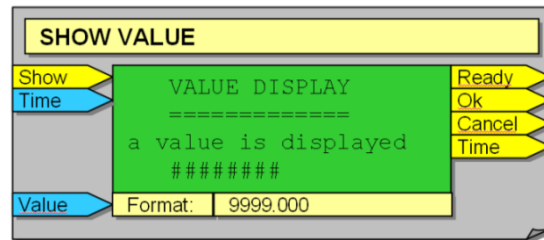
To show the previous message, you should write it in the 4 fields [Terminal message] in the “Show message” tab, as shown in the next figure, then press Ok:



**TAB: Show value**

This function is used to show a value on the TERM4 screen, and it contains 4 fields used to write your message (as in the previous function) and to define where the value should be shown by entering # characters within the text where the place for the displayed number, and there is another field [Value format] is used to define the format for this value.

*Symbol:*



*Data type:*

Show: Digital input

Time: Analog input (optional)

Value: Analog input

Ready: Digital output (optional)

Ok: Digital output (optional)

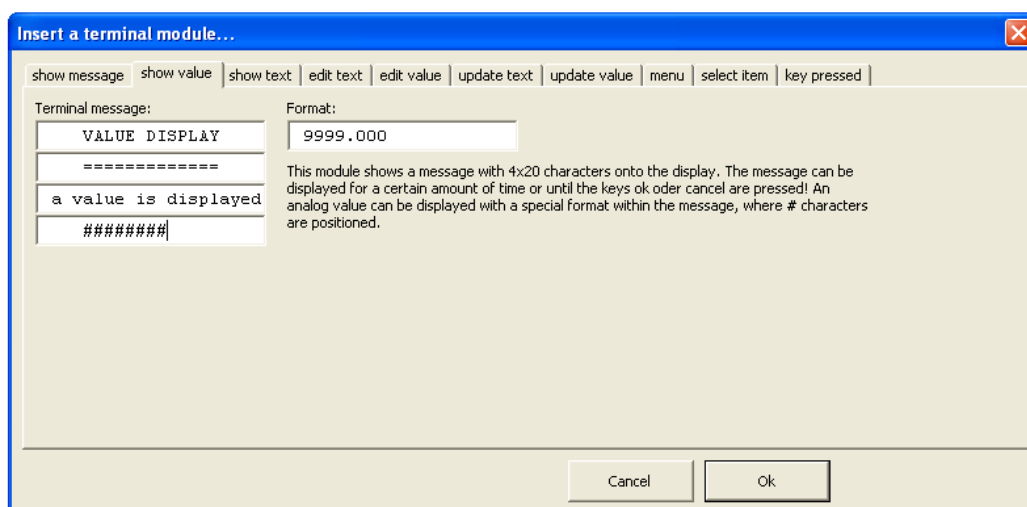
Cancel: Digital output (optional)

Time: Digital output (optional)

*Function:* With this function, if input “Show” has a rising edge the stored text will be displayed on the terminal. Additionally you can set a display time in seconds at input “Time”. If this time runs out, output “Time” will be activated. If the input stays unconnected, the time function will be ignored. Output “Ready” will be active when the complete screen set-up is finished. If the user presses the “OK” button, output “OK” will be activated. If the user presses the “Cancel” button, output “Cancel” will be activated. Input “Value” has the value which is displayed instead of the # characters.

*Example:*

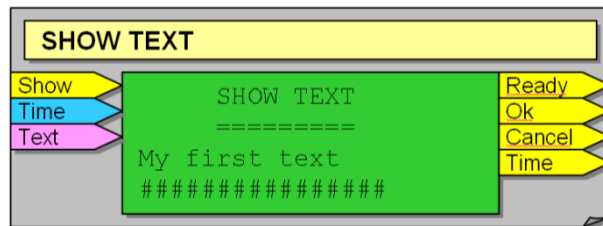
To show the previous message, you should write it in the “Show value” tab, as shown in the next figure, then press Ok:



**TAB: Show text**

This function is used to show a text on the TERM4 screen, and it contains 4 fields used to write your text, and to define where the text should be shown by entering # characters within the text where the place for the displayed text.

*Symbol:*



*Data type:*

Show: Digital input

Time: Analog input (optional)

Text: Text input

Ready: Digital output (optional)

Ok: Digital output (optional)

Cancel: Digital output (optional)

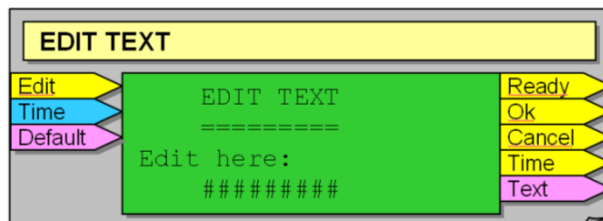
Time: Digital output (optional)

*Function:* With this function, if input “Show” has a rising edge the stored text will be displayed at the terminal. Additionally you can set a display time in seconds at input “Time”. If this time runs out, output “Time” will be activated. If the input stays unconnected, the time function will be ignored. Output “Ready” will be active when the complete screen set-up is finished. If the user presses the “OK” button, output “OK” will be activated. If the user presses the “Cancel” button, output “Cancel” will be activated. Input “Text” receives the character string which is displayed instead of the # characters.

**TAB: Edit text**

This function is used to edit a text on the TERM4 screen, and it contains 4 fields used to write your text, and to define where the text should be shown by entering # characters within the text where the place for the displayed text.

*Symbol:*



*Data type:*

Edit: Digital input

Time: Analog input (optional)

Default: Text input

Ready: Digital output (optional)

Ok: Digital output (optional)

Cancel: Digital output (optional)

Time: Digital output (optional)

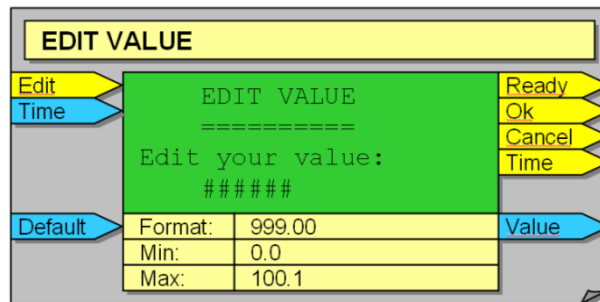
Text: Text output (optional)

*Function:* With this function, if input "Edit" has a rising edge the stored text will be displayed at the terminal. Additionally you can set a display time in seconds at input "Time". If this time runs out, output "Time" will be activated. If the input stays unconnected, the time function will be ignored. Output "Ready" will be active when the complete screen set-up is finished. If the user presses the "OK" button, output "OK" will be activated. If the user presses the "Cancel" button, output "Cancel" will be activated. Input "Default" defines the character string that is displayed at start up of the input. This string can be edited by the user at start up too. If the user presses "OK", output "Text" will receive the newly defined text. If the user presses "Cancel" the current input will be cancelled and the text "Default" will be delivered to output "Text".

**TAB: Edit value**

This function is used to edit a value on the TERM4 screen, and it contains 4 fields used to write your text, and to define where the value should be shown by entering # characters within the text where the place for the displayed value, and there is another field [Value format] is used to define the format for this value, and another two fields [Value minimum] and [Value maximum] are used to set a minimum and maximum value for the edited value.

*Symbol:*



*Data type:*

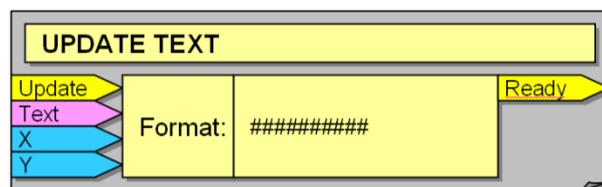
Edit	Digital input
Time	Analog input (optional)
Default	Analog input
Ready	Digital output (optional)
Ok	Digital output (optional)
Cancel	Digital output (optional)
Time	Digital output (optional)
Value	Analog output (optional)

*Function:* With this function, if input “Edit” has a rising edge the stored text will be displayed at the terminal. Additionally you can set a display time in seconds at input “Time”. If this time runs out, output “Time” will be activated. If the input stays unconnected, the time function will be ignored. Output “Ready” will be active when the complete screen set-up is finished. If the user presses the “OK” button, output “OK” will be activated. If the user presses the “Cancel” button, output “Cancel” will be activated. Input “Default” defines the number value that is displayed at start up of the number input. If the user presses “OK” after entering the number, the input will be converted and the analog value will be checked for the two limits ‘Value minimum’ and ‘Value maximum’. If the input value is within the range, output “OK” will be activated and the value will be delivered to output “Value”. If the input value is outside the range, the input cannot be continued with “OK”. By pressing “CANCEL” the value “Default” will be transmitted as a result of the input.

**TAB: Update text**

This function is used to enter one or more text on the same TERM4 screen by your program, and it contains just one field "Text format:" is used to define the number of character of your text by entering # characters.

*Symbol:*



*Data type:*

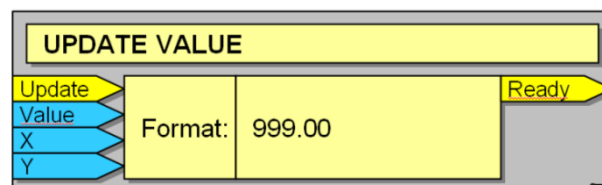
Update	Digital input
Text	Text input
X	Analog input
Y	Analog input
Ready	Digital output (optional)

*Function:* The current value of input "Text" with the specified length will be displayed from position "X" and "Y" in the terminal content. The coordinates will be counted beginning with (0,0) for the upper left character at the screen and ending with (19,3) for the lower right character at the screen. This action will be executed with every rising edge at input "Update". Output "Ready" will be activated immediately after the display of the text.

**TAB: Update value**

This function is used to enter one or more value on the same TERM4 screen by your program, and it contains just one field [value format] is used to define the format for this value.

*Symbol:*



*Data type:*

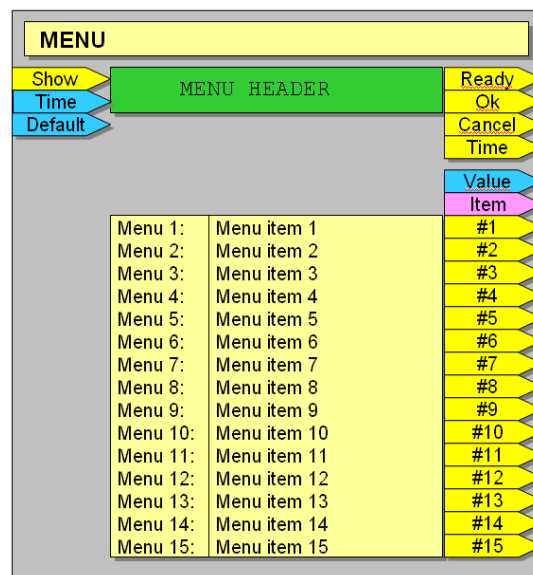
Update	Digital input
Value	Analog input
X	Analog input
Y	Analog input
Ready	Digital output (optional)

**Function:** The current value of input “Value” with the specified format will be displayed from position “X” and “Y” in the terminal content. The coordinates will be counted beginning with (0,0) for the upper left character at the screen and ending with (19,3) for the lower right character at the screen. This action will be executed with every rising edge at input “Update”. Output “Ready” will be activated immediately after the display of the value.

#### **TAB: Menu**

This function is used to enter a menu on TERM4 screen, and it contains one field for the menu header “Menu header:” and 15 fields “Menu items:” for up to 15 items you can insert in your menu.

**Symbol:**



**Data type:**

Show	Digital input
Time	Analog input (optional)
Default	Analog input (optional)
Ready	Digital output (optional)
Ok	Digital output (optional)
Cancel	Digital output (optional)
Time	Digital output (optional)
Value	Analog output (optional)
Item	Text output (optional)
#1 → #15	Digital output



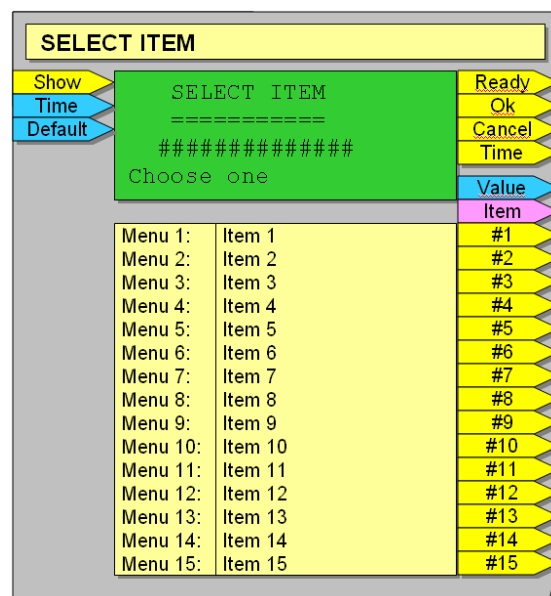
**Function:** With this function, if input “Show” has a rising edge, the stored text will be displayed at the terminal. Additionally you can set a display time in seconds at input “Time”. If this time runs out, output “Time” will be activated. If the input stays unconnected, the time function will be ignored. Output “Ready” will be active, if the complete screen set-up is finished. If the user presses the “OK” button, output “OK” will be activated. If the user presses the “Cancel” button, output “Cancel” will be activated.

A menu will be build up. The selected entry which has an order corresponds to the input “Default”. If this entry is unconnected, the first menu entry will always be selected. If you choose a menu entry and press “OK”, the number of the chosen menu entry will be delivered to output “Value”. The menu entry text will be delivered to output “Item” and the corresponding output “#1” to “#15” will be activated too. If you press “CANCEL” only output “Cancel” will be activated.

#### **TAB: Select item**

This function is used to show one item from up to 15 items on TERM4 screen; the position for the text entries has to be marked by # characters.

*Symbol:*



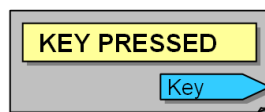
*Data type:*

Show Digital input  
 Time Analog input (optional)  
 Default Analog input (optional)  
 Ready Digital output (optional)  
 Ok Digital output (optional)  
 Cancel Digital output (optional)  
 Time Digital output (optional)  
 Value Analog output (optional)  
 Item Text output (optional)  
 #1 → #15 Digital output

*Function:* with this function, if input “Show” has a rising edge the terminal will display the stored text. Additionally you can set a display time in seconds at input “Time”. If this time runs out, output “Time” will be activated. If the input stays unconnected, the time function will be ignored. Output “Ready” will be active when the complete screen set-up is finished. If the user presses the “OK” button, output “OK” will be activated. If the user presses the “Cancel” button, output “Cancel” will be activated. The entered message will be build up. The displayed entry corresponds to input “Default”. If this input is unconnected, the first entry will always be displayed. If you choose an entry and press “OK”, output “Value” will output the number of the chosen entry. Output “Item” will output the entry text and the corresponding output “#1” to “#15” will be activated too. If you press “CANCEL” just output “Cancel” would be activated.










**TAB: Key pressed**

Symbol:



*Data type:* Key (Analog output)

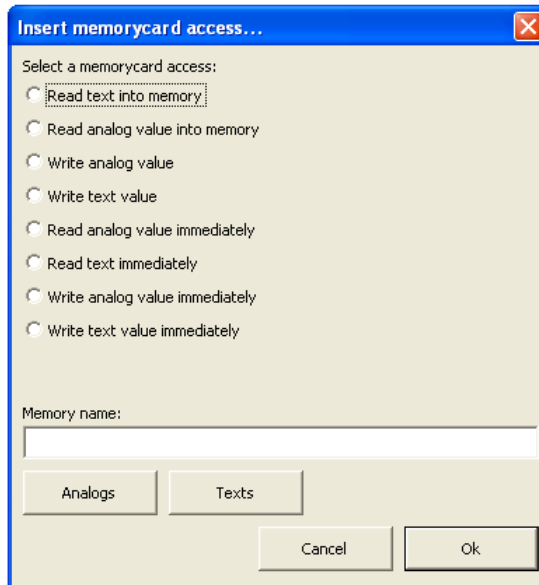
*Function:* The current value of the keys will be delivered from output “Key”. If no key is pressed, the value 0 will be returned. However the following code will be returned (Total number between 1 and 9):

		
2.000	5.000	9.000
		
3.000	7.000	4.000
		
8.000	6.000	1.000

### Objects: Memory card

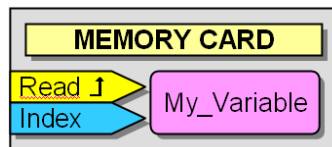
This chapter deals with all functions, which enables the saving of values to permanent memories like Memory Card.

Choose Objects/Memory card from the menu to get to the following window:



### Read text into memory

*Symbol:*

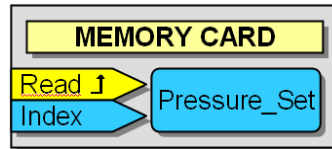


*Data type:*

Read     Digital input

Index    Analog input

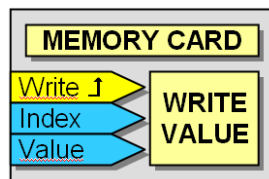
**Function:** If a rising edge is detected at input “Read” the text value stored at position “Index” will be read from the Memory Card and saved to the text variable “My\_Variable”.

**Read analog value into memory***Symbol:*

*Data type:* Read      Digital input  
                          Index      Analog input

*Function:* If a rising edge is detected at input "Read" the analog value stored at position "Index" will be read from the Memory Card and saved to the analog variable "Pressure\_Set".

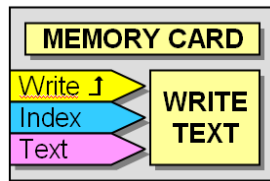
*ADVICE:* If the addresses 100000 to 100002 are used, the saved analog values will be taken from the real time clock.

**Write analog value***Symbol:*

*Data type:* Write      Digital input  
                          Index      Analog input  
                          Value      Analog input

*Function:* If a rising edge is detected at input "Write", the analog value of input "Value" will be saved to the Memory Card to position "Index".

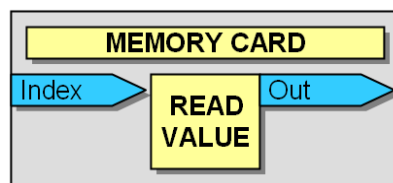
*ADVICE:* If the addresses 100000 to 100002 are used, the analog values will be saved to the real time clock.

**Write text value***Symbol:*

*Data type:*

Write	Digital input
Index	Analog input
Text	Text input

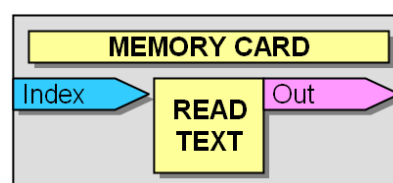
*Function:* If a rising edge is detected at input "Write", the text value of input "Text" will be saved to the Memory Card to position "Index".

**Read analog value immediately***Symbol:*

*Data type:*

Index	Analog input
Out	Analog output

*Function:* The current value of the analog register of index equal to the input "Index" will be saved to analog output "Out" as an analog variable.

**Read text value immediately***Symbol:*

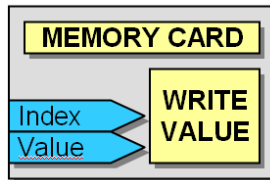
*Data type:*

Index	Analog input
Out	Text output

*Function:* The current value of the text register of index equal to the input “Index” will be saved to text output “Out” as a text variable.

### Write analog value immediately

*Symbol:*



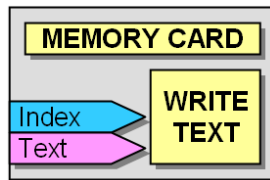
*Data type:* Index      Analog input  
Value      Analog input

*Function:* The current value of analog input “Value” will be saved to the Memory Card at register which has an index equal to analog input “Index”.

ADVICE: If the addresses 100000 to 100002 are used, the analog values will be saved to the real time clock.

### Write text value immediately

*Symbol:*

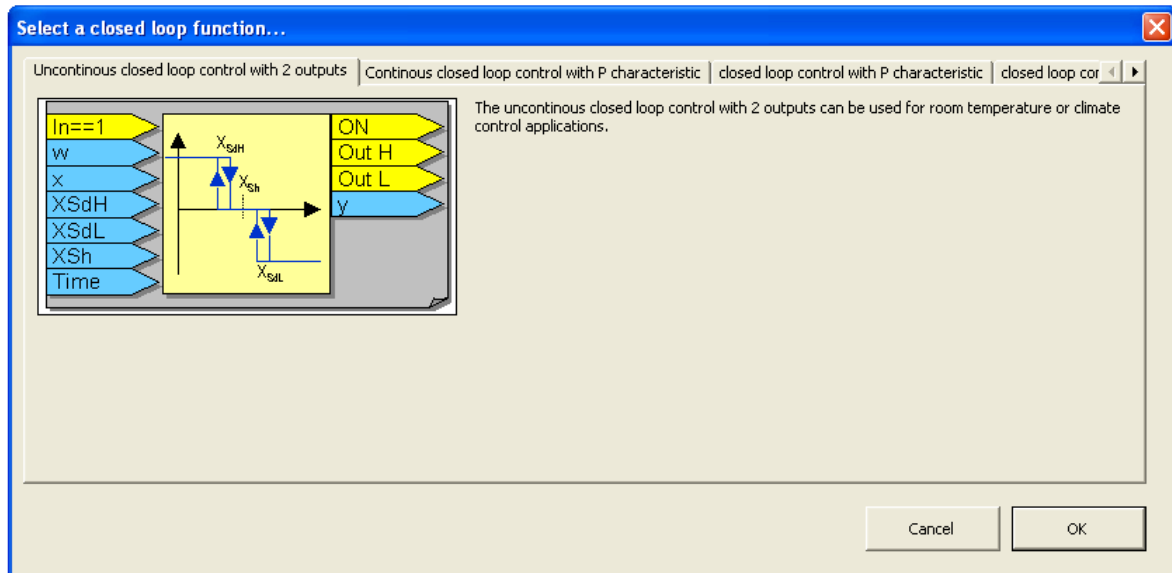


*Data type:* Index      Analog input  
Text      Text input

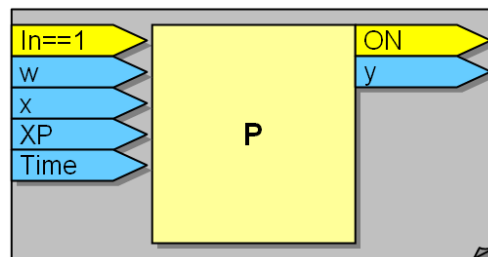
*Function:* The current value of text input “Text” will be saved to the Memory Card at register which has an index equal to analog input “Index”.

**Objects: Closed loop control**

This object deals with the closed loop control functions, such as P, PI, and PID control.

**Closed loop control with P characteristic**

*Symbol:*



*Data type:*

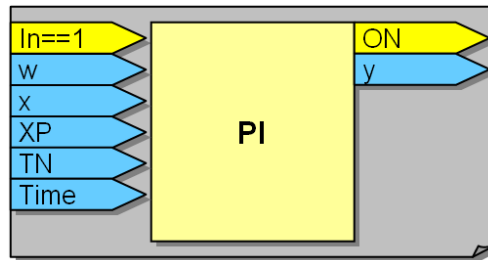
In	Digital input
W	Analog input
X	Analog input
XP	Analog input
Time	Analog input
ON	Digital output
Y	Analog output

*Function:* this function is used to control an analog value through proportional gain only, the digital input "In" must be activated to enable the function, the analog input "W" represents the set value, the analog input "X" represents the real analog value that you want to control (Feedback), the "XP" represents the proportional gain, the analog input "Time" is

used to specify a time in seconds which determines when the function updates, the analog output “Y” represents the analog value that have to be enabled to the physical instrument that controls your analog value (such as valve).

### Closed loop control with PI characteristic

*Symbol:*

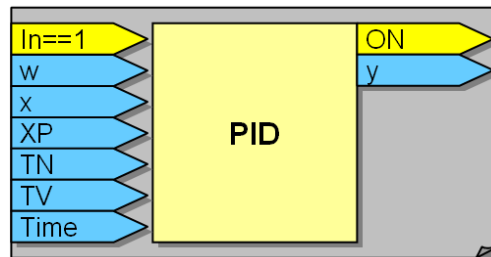


*Data type:*

In	Digital input
W	Analog input
X	Analog input
XP	Analog input
TN	Analog input
Time	Analog input
ON	Digital output
Y	Analog output

*Function:* this function is used to control an analog value through proportional, and integral gain, the digital input “In” must be activated to enable the function, the analog input “W” represents the set value, the analog input “X” represents the real analog value that you want to control (Feedback), the “XP” represents the proportional gain, the “TN” represents the integral gain, the analog input “Time” is used to specify a time in seconds which determines when the function updates, the analog output “Y” represents the analog value that have to be enabled to the physical instrument that controls your analog value.



**Closed loop control with PID characteristic***Symbol:**Data type:*

In	Digital input
W	Analog input
X	Analog input
XP	Analog input
TN	Analog input
TV	Analog input
Time	Analog input
ON	Digital output
Y	Analog output

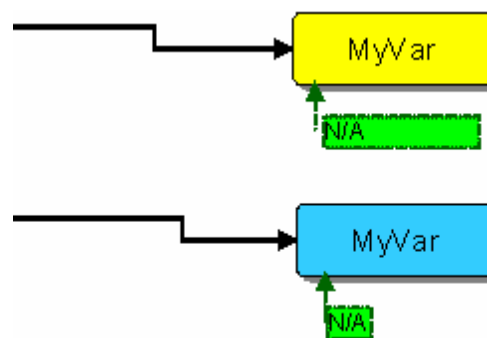
**Function:** this function is used to control an analog value through proportional, integral, and derivative gain, the digital input “In” must be activated to enable the function, the analog input “W” represents the set value, the analog input “X” represents the real analog value that you want to control (Feedback), the “XP” represents the proportional gain, the “TN” represents the integral gain, the “TV” represents the derivative gain, the analog input “Time” is used to specify a time in seconds which determines when the function updates, the analog output “Y” represents the analog value that have to be enabled to the physical instrument that controls your analog value (such as valve).

**Debug Menu**

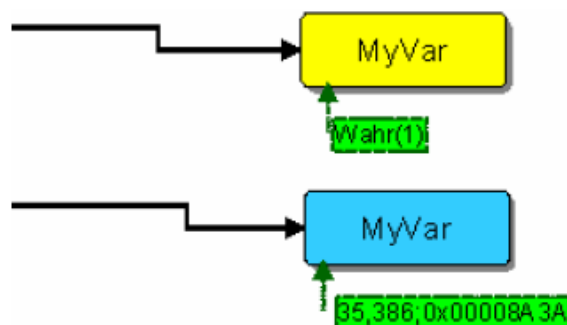
In this menu you can select from various functions for an online test with the hardware modules and the programming software. Choose Debug from the menu bar to get the functions listed below:

**Debug: Add symbol**

By choosing [Add Symbols] from the menu, the following green Symbol will be added to the selected Memory:



Now choose “Update all Symbols” from the menu to read the current state of the memory from the PC connected SLS-500 main module:

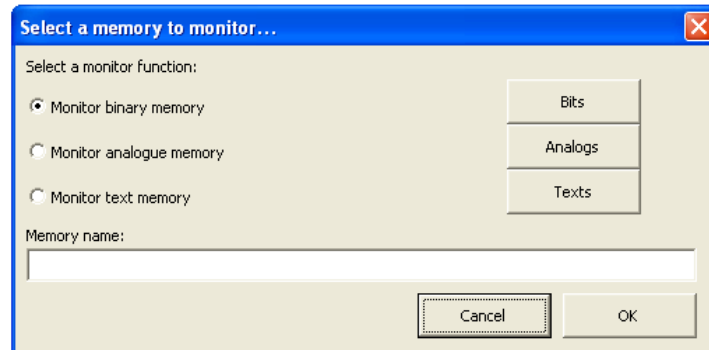


After the successful update the current states/values will be displayed within the green field.

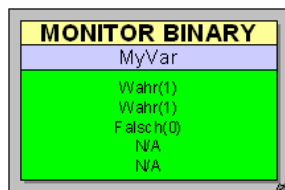
To delete all Symbols, choose “Remove all Symbols” from the menu.

**Debug: Add monitor**

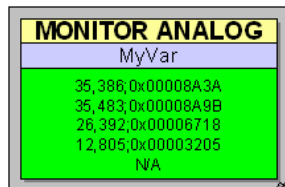
By choosing [Add monitor] from the menu, the following window will appear:



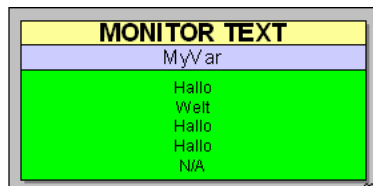
Now choose the memory type and write the memory name of the SLS-500 main module that you want to monitor on the PC, or choose it from one of the three buttons at right, then press OK. This symbol is for binary memory:



And this is for analog memory:



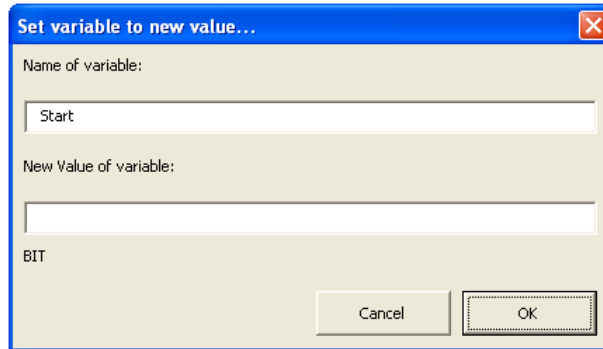
And this is for text memory:



Now choose [Update all Symbols] from the menu to read the current state of the binary/analog/text memory from the PC connected SLS-500 main module. The first line shows the most recent state.

**Debug: Write to symbol**

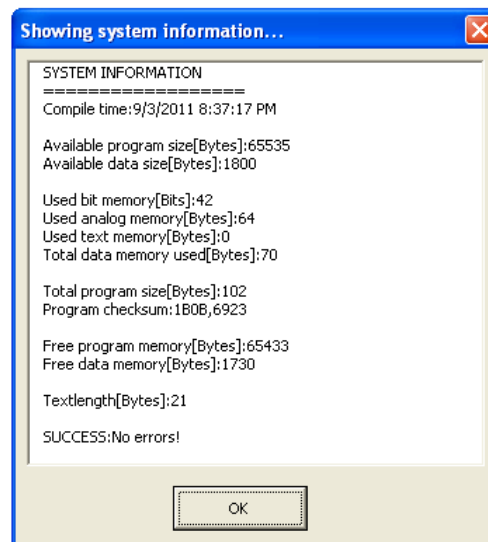
Use this function to force a value for a digital/analog/text memory by choosing “Write to symbol” from the menu, the following window will appear, but before choosing this function, you should select the memory that you want to write:



In this window you will find the name of the memory that you choose is written in the “Name of variable:” field, and the type of this memory is placed below the 2<sup>nd</sup> field “New Value of variable:” which you should write the new value for the memory, then press OK.

**Debug: Show system information**

Choose “Show System Information” from the menu to get to the following window:



This window is more useful in calculation of your remaining program memory, if you look at the 3<sup>rd</sup> last line you will see “Free data memory [Bytes]: 1730”, this represents the remaining program memory from the total program memory which specified at the 4<sup>th</sup> line “Available data size [Bytes]: 1800”, and it depending on the main module you use and if you use memory

card for your program or not, and also you will see more important details on each type of memory in the next lines “Used bit memory...”, “Used analog memory...”, “Used text memory...”, and “Total data memory used...”.

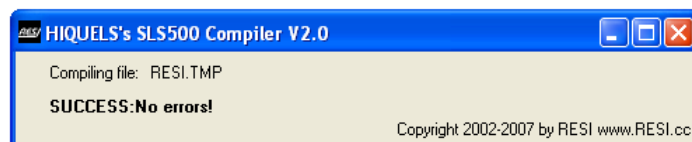
**Run Menu**

This menu option contains all actions of the compilation, simulation and the execution of the program on the SLS500. Choose Run from the menu bar to get the functions listed below:

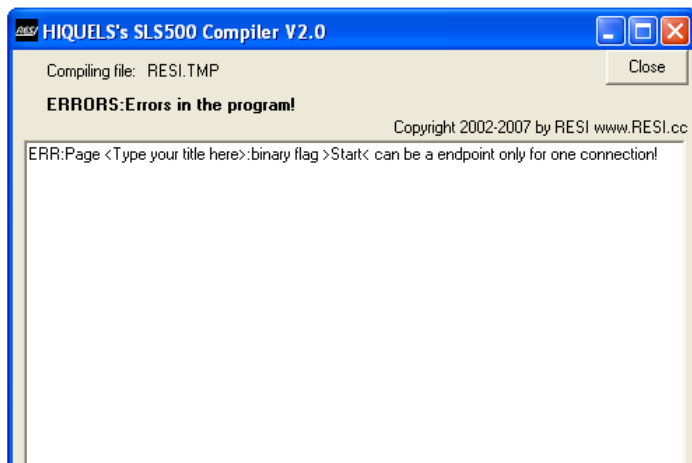
**Run: Compile**

This menu item starts the integrated compiler. The compiler creates an executable program out of the current graphic, and a window appears to show the status of the compiler [Wait: Compiling...] It also shows failures of your program if exist.

If there are no errors in your program the following window will appear for about 1 sec, Depending on size of the program, then it will disappear automatically and an executable program will be available.



If there is an error in your program, another window will appear specifying the title of page contains the error, and if possible the exact failure reason will be displayed too.

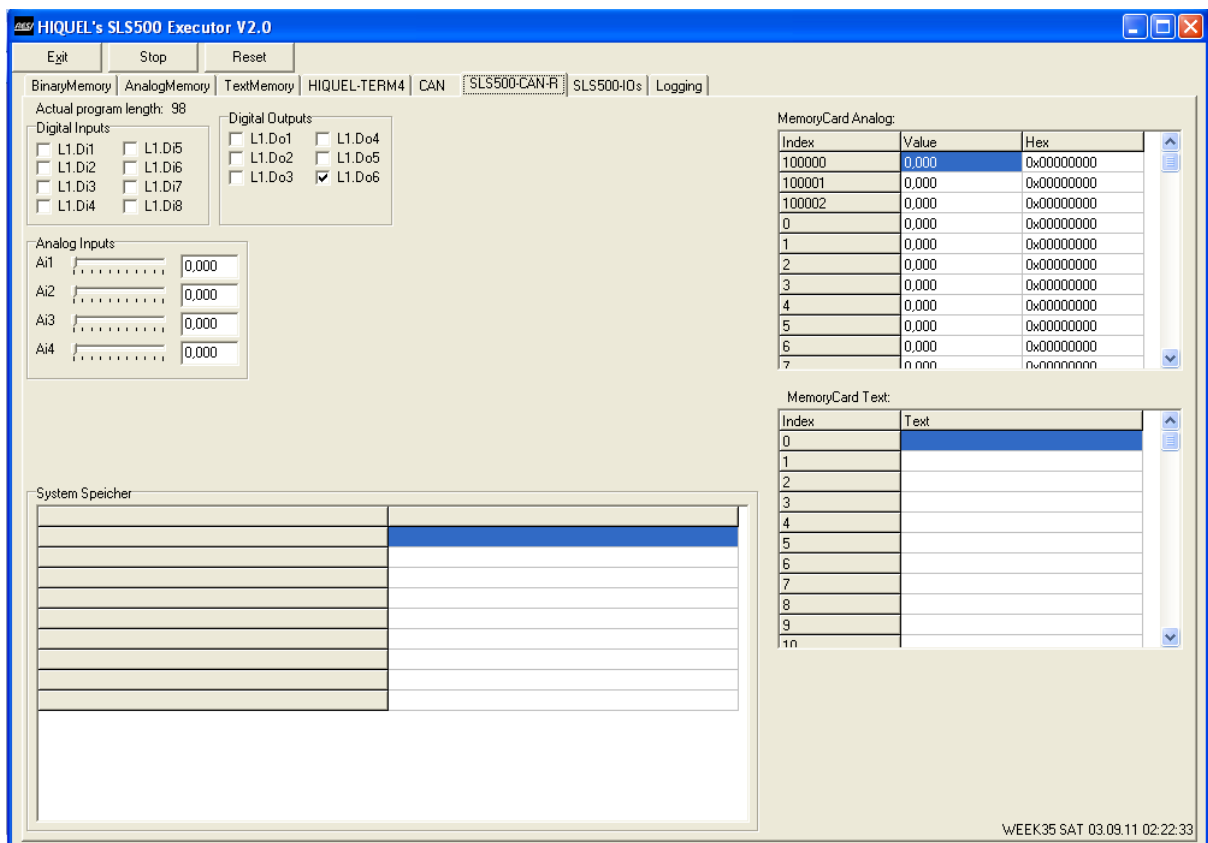


Click Close to finish the Compiler.

**Run: Simulate**

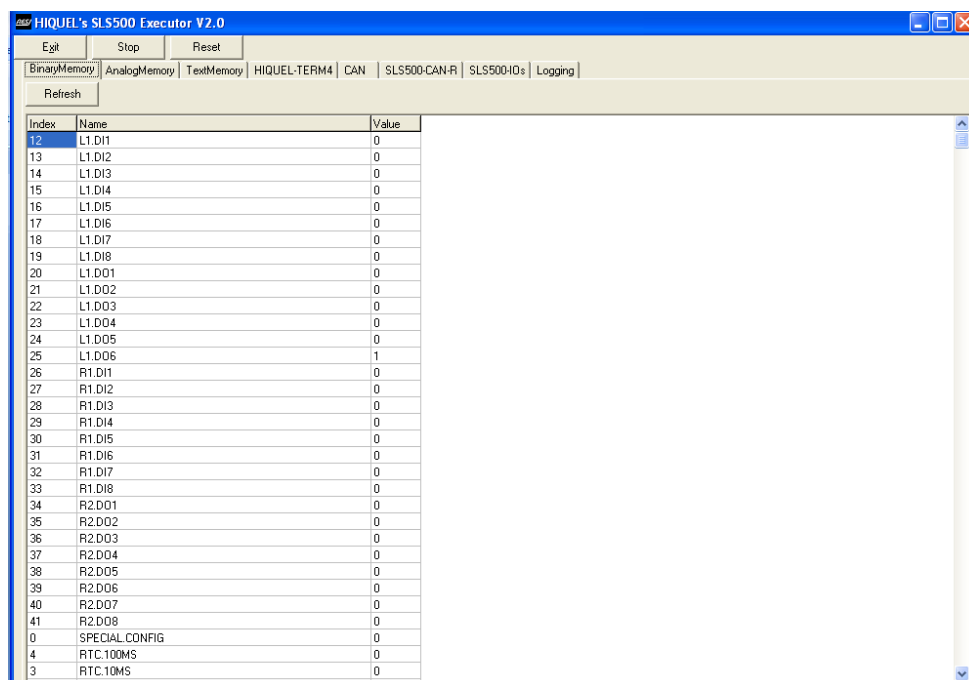
You can simulate a SLS-500-Configurator program on the screen. If the program was compiled successfully the simulator will be started automatically which enables you to test a complete application without external elements being connected.

Choose "Simulate" from the menu to get to the following window:



### Simulate Binary memory

All binary memories of SLS will be displayed here.

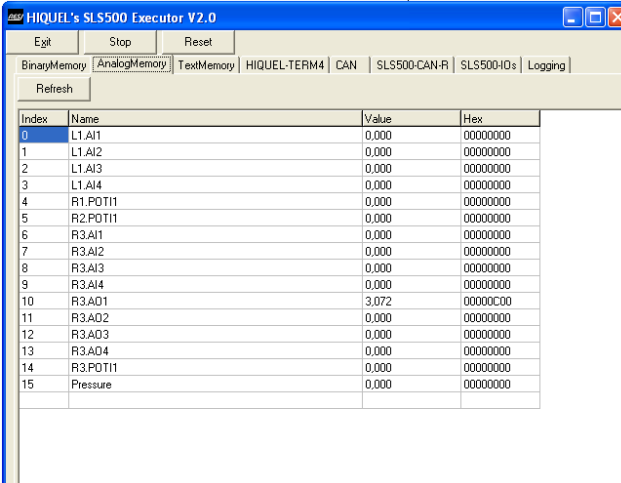


The column “Index” describes the internal memory space of the marker, “Name” describes the name, and “Value” shows you the current value.

To change the value of a memory, double-click the field “Index” and the value will be inverted!

### Simulate Analog memory

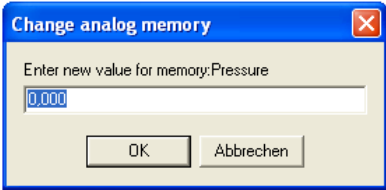
All analog memories of SLS will be displayed here.



Index	Name	Value	Hex
0	L1.AI1	0.000	00000000
1	L1.AI2	0.000	00000000
2	L1.AI3	0.000	00000000
3	L1.AI4	0.000	00000000
4	R1.POT11	0.000	00000000
5	R2.POT11	0.000	00000000
6	R3.AI1	0.000	00000000
7	R3.AI2	0.000	00000000
8	R3.AI3	0.000	00000000
9	R3.AI4	0.000	00000000
10	R3.A01	3.072	00000C00
11	R3.A02	0.000	00000000
12	R3.A03	0.000	00000000
13	R3.A04	0.000	00000000
14	R3.POT11	0.000	00000000
15	Pressure	0.000	00000000

All analog values are displayed in column “Value” and also a 32-Bit hexadecimal values in column “Hex”.

To change an analog value, click on the corresponding “Index” field. The following entry form opens up:



**Change analog memory**

Enter new value for memory:Pressure

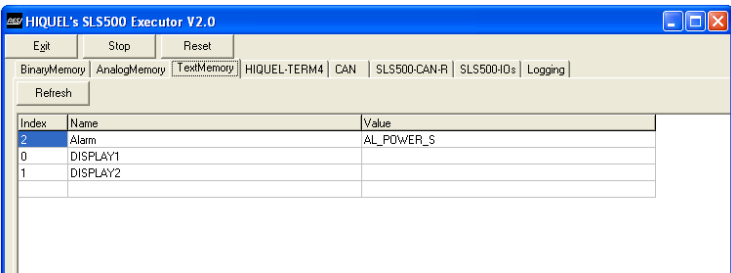
0.000

OK Abbrechen

Enter the new analog value and confirm by clicking OK.

### Simulate Text memory

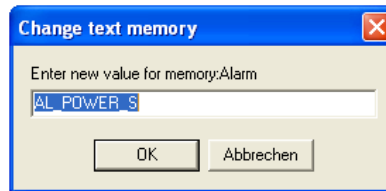
All text memories of SLS will be displayed here.



Index	Name	Value
2	Alarm	AL_POWER_S
0	DISPLAY1	
1	DISPLAY2	



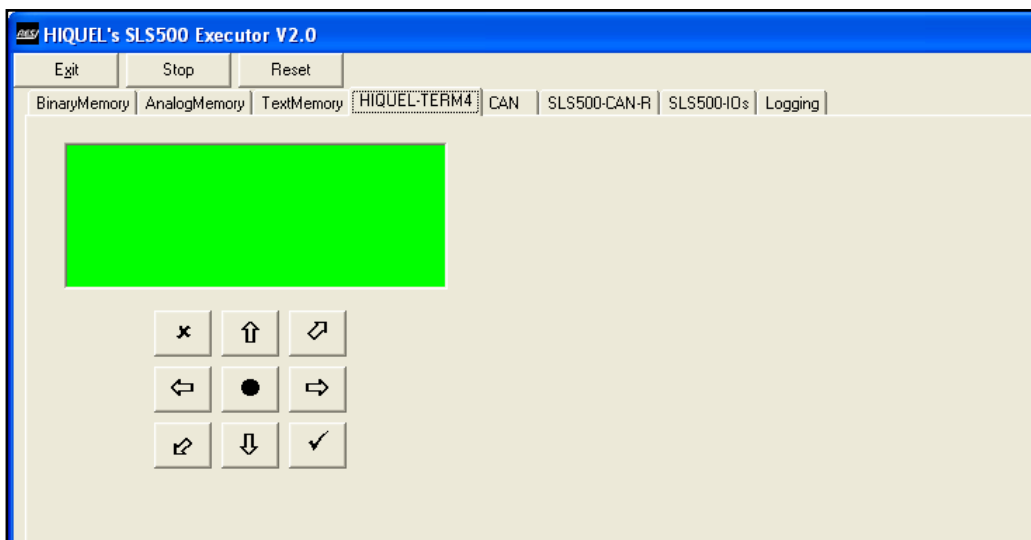
You can also change the text memory. Double-click the “Index” field and the following entry forms opens up:



Enter the new text value and confirm by clicking OK.

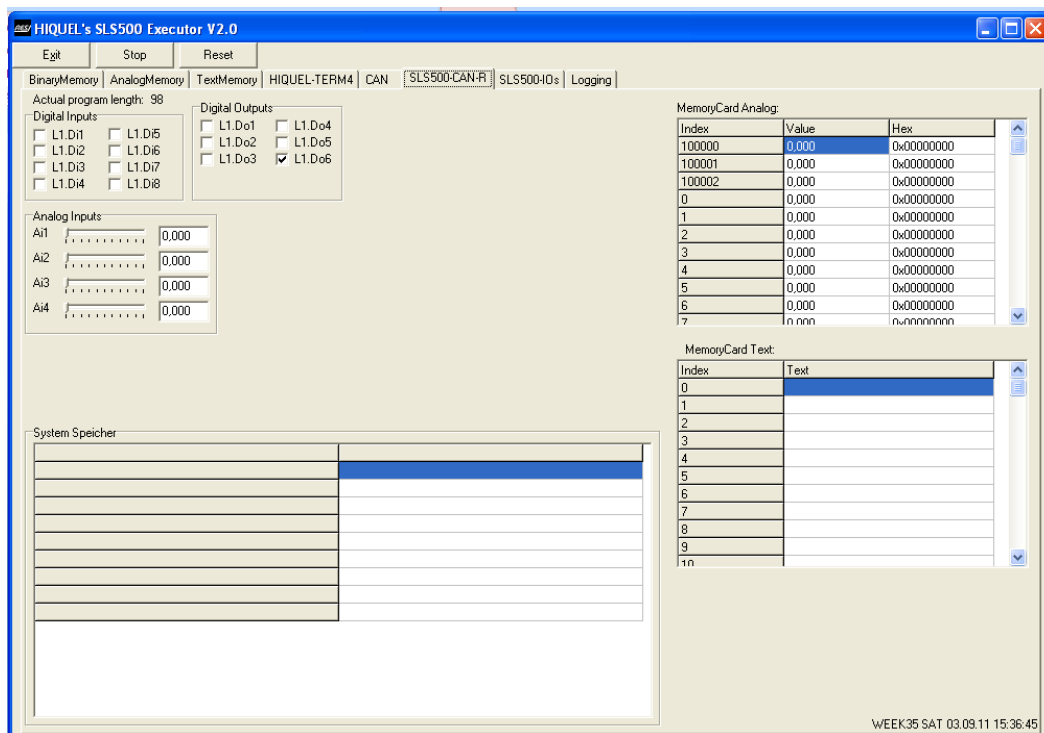
### Simulate: HIQUEL-TERM4

This tab appears only if you add text screen TERM4 to your configuration because it is specialized for it, and contains a simulation for all 9 buttons of TERM4 and its screen too.



### Simulate: SLS500-CAN-R

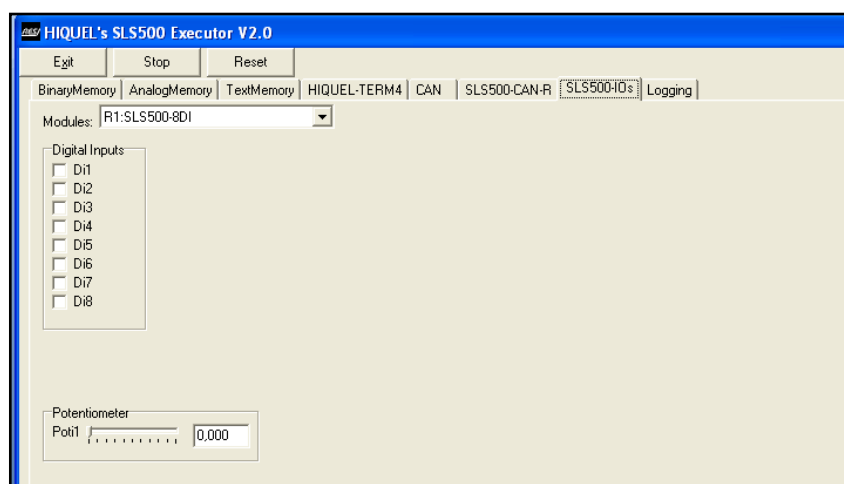
This tab is specialized for the base module you choose, and here we choose SLS500-CAN-R, and it contains a simulation for all inputs and outputs for this module, also it contains a simulation for the memory card registers (Analog and text), and in the lower right corner you will see the real time clock of your PC to use it in your simulation (See the next figure).



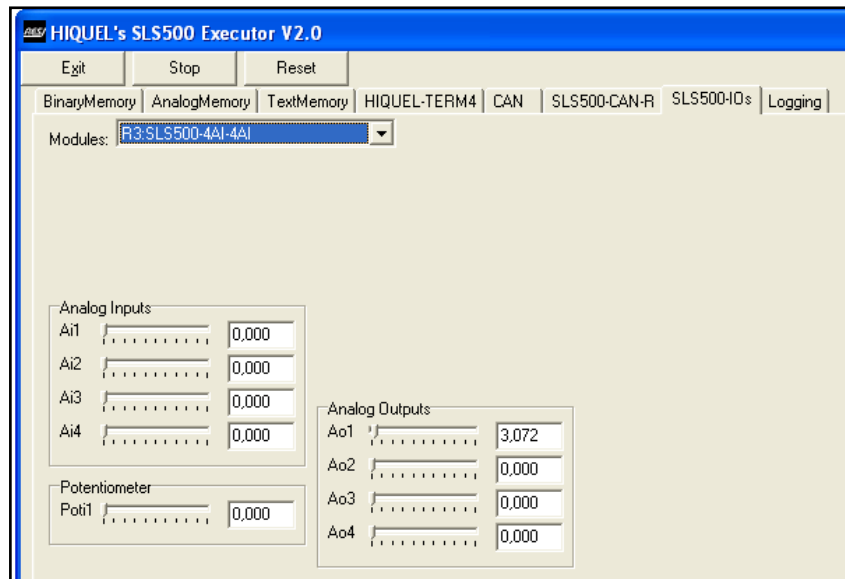
You will find the digital inputs and outputs is simulate as a check box, you can check box the digital inputs, and corresponding to your program the check box of digital outputs will be checked, also you will find the analog inputs for this base module as a slide bar from 0 to 100%.

### Simulate: SLS500-IOs

This tab is specialized for all the extension modules connected to the base module, and in the submenu you can choose which extension you want to simulate then there are all inputs, outputs, and potentiometer according to each extension, in the next figure the first extension to the base module is SLS500-8DI, it has 8 digital inputs and potentiometer.

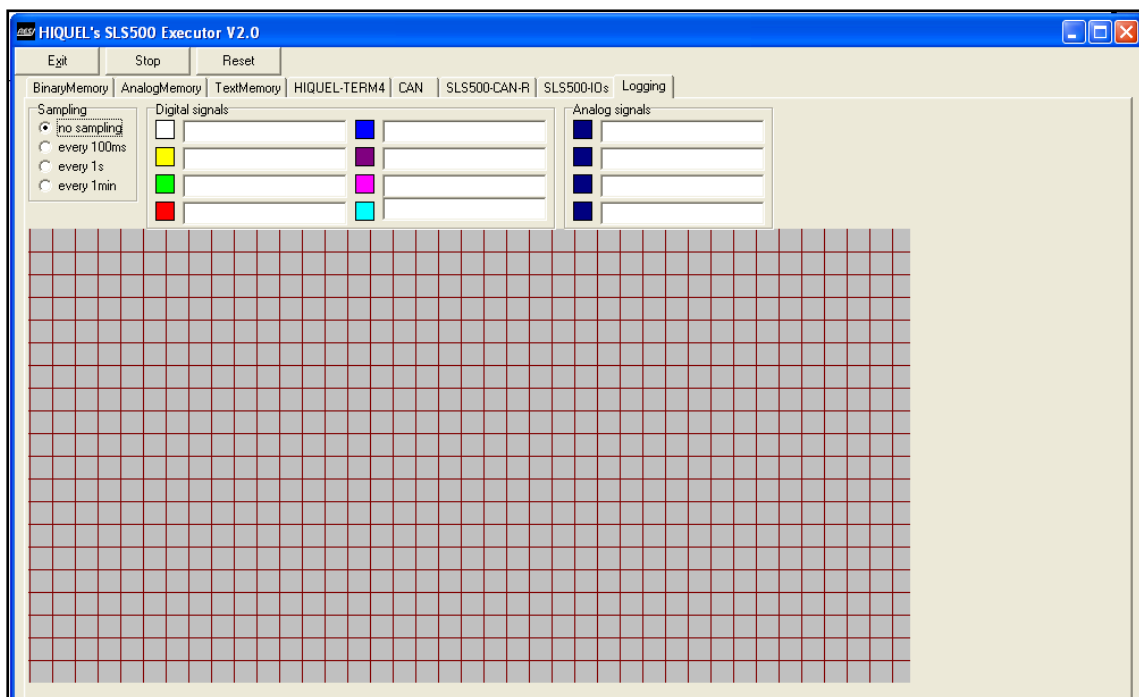


In the next figure you will see another extension SLS500-4AI-4AO, it contains 4 analog inputs, and 4 analog outputs.

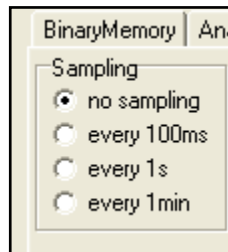


### Simulate: Logging

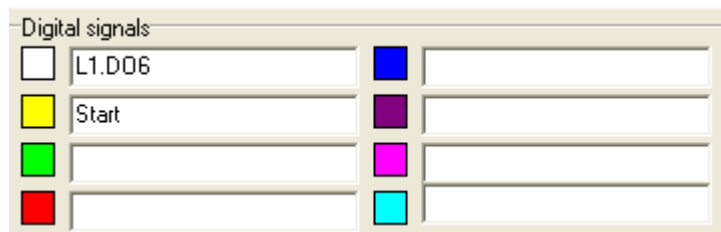
You can take a look at the rising and falling edges on this page, also the variation for analog memories.



You can set the update time of the edges here to be every 100msec, or 1 sec, or 1 min.

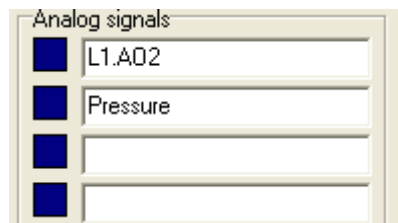


And here you can write the digital memories up to 8 digital memories each one with a unique color.



And here you can write the analog memories up to 4 analog memories.

To close the Simulator click "Exit" on the upper left.



#### **Run: Download+Run**

By choosing this menu item the program will be automatically compiled. If no failure occurs, the program will be loaded to the connected SLS500. Then the program will be started immediately.

#### **Run: Start PLC**

Choose this menu item to restart the current SLS500 program on the connected SLS500.

#### **Run: Stop PLC**

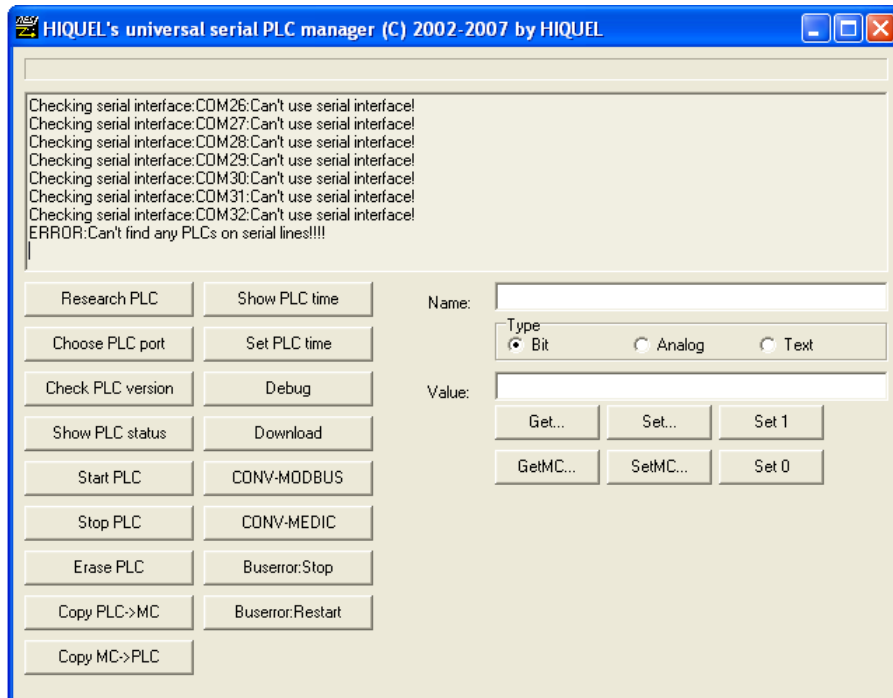
Choose this menu item to stop the current SLS500 program.

#### **Run: Erase PLC**

Choose this menu item to delete the current SLS500 program.

**Run: Show**

Activating this menu item the PLC manager will be activated with a screen interface that allows special actions as shown in the next window:



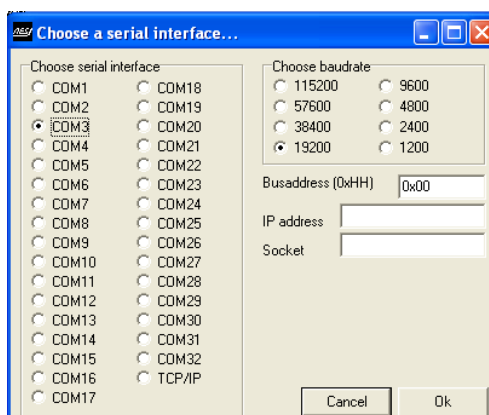
The previous screen is available only when the SLS500 doesn't connected to the PC or the communication parameters don't correct, if this use the second button "Choose PLC port" to modify these parameters as shown below.

**Run: Show: Research PLC**

Choose this button to scan through all ports for connected SLS500.

**Run: Show: Choose PLC port**

Choose this button to select the interface and the baud rate where the PLC is connected, using the next window:



Here in the left part, you can choose the COM port that's the programming cable is connected, and in the upper right part "Choose baudrate" you can choose the speed of transferring data, and finally the "Bus address" which consists of 2 hexadecimal or 4 bits you can choose the bus address according to the next table:

<b>Data bits</b>	Bit 0	0	7 Data bits
	Bit 0	1	8 Data bits
<b>Parity</b>	Bit 1+2	00	No parity
	Bit 1+2	01	Even parity
	Bit 1+2	10	Odd parity
<b>Stop bits</b>	Bit 3	0	1 Stop bits
	Bit 3	1	2 Stop bits

For example:

	Binary code				Hexa code
	Bit 0	Bit 1	Bit 2	Bit 3	
8 data bits, No parity, and 1 Stop bits	1	0	0	0	0x01
8 data bits, No parity, and 2 Stop bits	1	0	0	1	0x09
7 data bits, Even parity, and 1 Stop bits	0	1	0	0	0x02
7 data bits, Odd parity, and 1 Stop bits	0	0	1	0	0x04
8 data bits, Odd parity, and 1 Stop bits	1	0	1	0	0x05

#### Run: Show: Check PLC version

Choose this button to check the software version of the connected PLC.

#### Run: Show: Show PLC status

Choose this button to check the current status of SLS500. The current state of SLS500 (runs, does not run) or (communication parameter is right or not) and failures will be displayed. In addition you can see the current program length and the check sum of the current program.

#### Run: Show: Start PLC

You can start the program on the SLS500 by clicking Start.

#### Run: Show: Stop PLC

Stop the SLS500 program by clicking this button.

**Run: Show: Erase PLC**

You can completely delete the SLS500 program by clicking Erase.

**Run: Show: Copy PLC -> MC**

Choose this button to save the current program, which is already saved in the SLS500, to the Memory Card.

**Run: Show: Copy MC-> PLC**

Choose this button to save the program stored on the Memory Card to the SLS500.

If the Memory Card contains a valid program and SLS500 is restarted, the program will be loaded to SLS500 automatically. After that you can remove the Memory Card.

**Run: Show: Show PLC time**

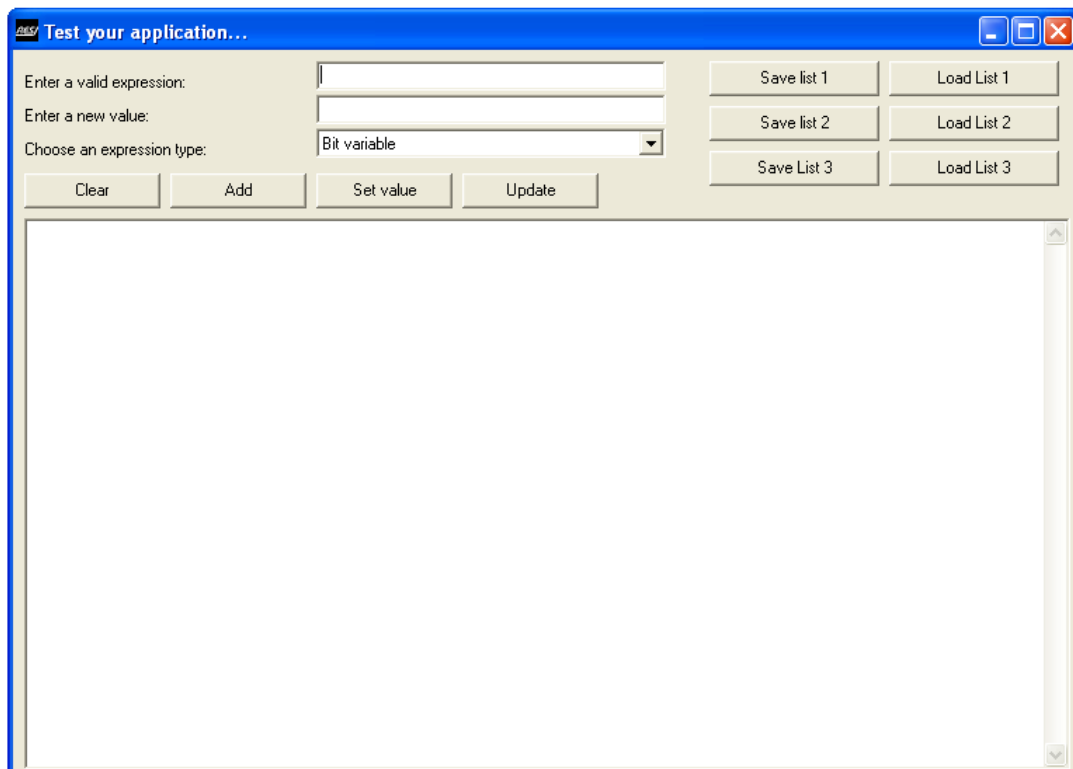
Display the current time of SLS500 by clicking this button.

**Run: Show: Set PLC time**

Click on this button to set the current time of the PC as new time for SLS500.

**Run: Show: Debug**

Choose this button the Test program will be activated with a screen interface that allows data exchange for many memories at the same time, as in the following window:



You can write the input/output signal in the “Enter a valid expression:” field, and choose its type using “Choose an expression type:” field then press “Add” button to add the signal into the white area down.

Repeat the previous instructions for another signal, as shown in the following window:

The screenshot shows a software window titled "Test your application...". It contains several input fields and buttons. The "Enter a valid expression:" field contains "L1.D05". The "Enter a new value:" field contains "N/A". The "Choose an expression type:" dropdown menu is set to "Bit variable". Below these fields are buttons for "Clear", "Add", "Set value", and "Update". To the right of the input fields are six buttons arranged in a 3x2 grid: "Save list 1", "Load List 1", "Save list 2", "Load List 2", "Save List 3", and "Load List 3". Below the buttons is a large text area containing a list of signals and their values:

```
L1.D11.#12,bv:False(0)
L1.A12.#1.av:0.000.0x00000000
L1.D13.#14,bv:False(0)
L1.D16.#17,bv:False(0)
L1.D02.#21,bv:False(0)
L1.D04.#23,bv:False(0)
L1.D05.#24,bv:False(0)
```

After you enter the signal, press the “Update” button to update the value for all signals you entered, plus you can force an output using the [Enter a new value] field additionally you can save the list you write and upload it again, you can save up to 3 lists.

#### Run: Show: Download

Choose this button the program which was compiled will be loaded to the connected SLS500. Then the program will be started immediately.

#### Run: Show: Read/Write binary memory

You can read Bits from the current program and set new bits with the PLC Manager. Use the right area of the PLC Manager. Choose Name to enter the name of the bit, the name corresponds to the names of the memory which were set in SLS-500-Configurator, or to set a memory number, do this by prefixing a # character before the number (for example #123). Type must be Bit.



Click the button Get to query the current state of the Bit. You can also set a new value by choosing “Set 1” or “Set 0” button.

**Run: Show: Read/Write analog memory**

Reading the analog values is done in the same way as reading binary values. You just have to change the Type to Analog.

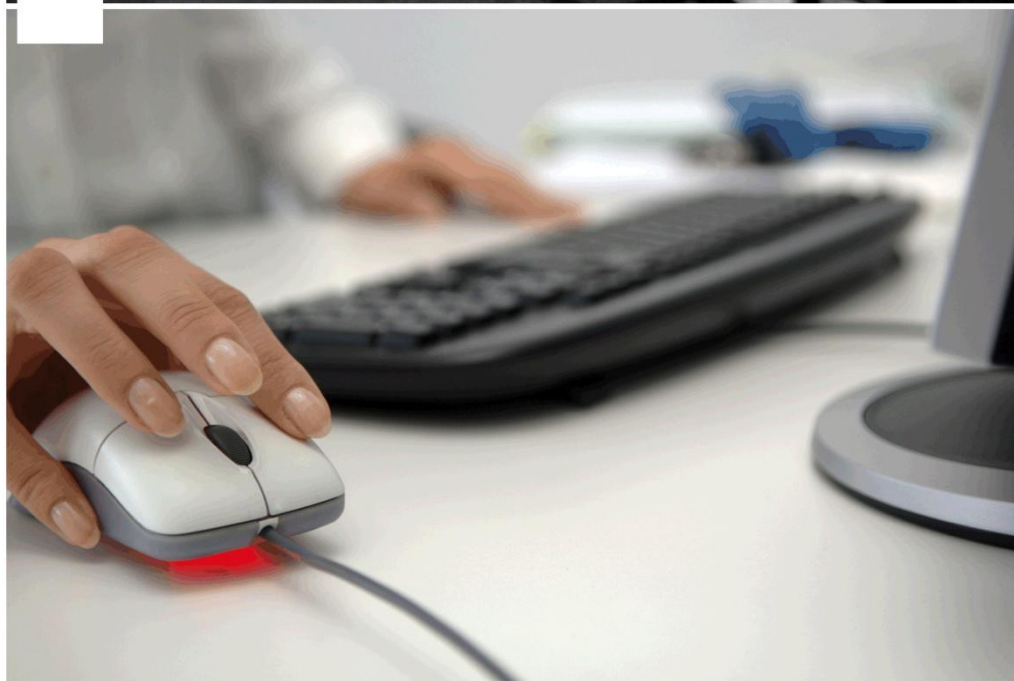
To preset an analog value, just enter a valid value into the field **Value**, and press “Set...”

**Run: Show: Read/Write text memory**

To read and write text memories you just have to change the Type to Text and enter a character string into the field **Value**, and press “Set...”



## 5. Applications and Exercises





**Applications**

1. Stairway or Corridor Lighting
2. Factory door control system
3. Motor run forward and reverse application
4. Air conditioning system
5. Chemical process plant control
6. Service water pump

### Application 1: Stairway or Corridor Lighting

#### Description:

If someone uses the stairway, the lighting should be switched on.

If nobody uses the stairway, the lights should be switched off in order to save energy.

#### Old solution:

The two conventional options for switching lights on or off:

- using a pulse relay
- using an automatic stairway light switch

The wiring of both lighting systems is identical.

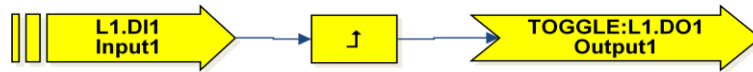
By using a SLS system, the automatic stairway light switch or the pulse relay can be replaced. Moreover, both functions (timed off-delay and pulse relay) can be implemented in one single unit. Another advantage is obvious: extra functions can be added without altering the wiring.

Here are some examples:

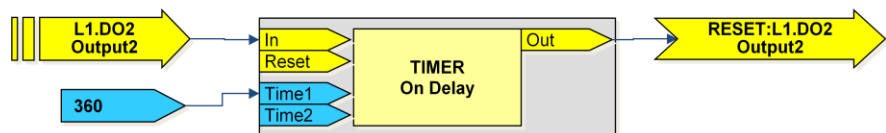
- 1 pulse relay with SLS
- 2 automatic stairway light switch with SLS
- 3 multiple function switch with SLS:
  - switch the light on
  - switch the light on permanently
  - switch the light off

**1 Pulse relay with SLS:**

Output L1.DO1 is toggled with a pulse signal at input L1.DI1.

**2 Automatic stairway light switch with SLS:**

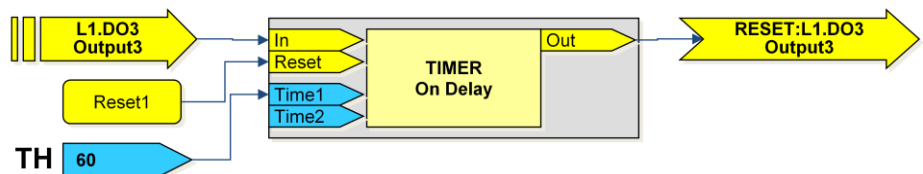
Output L1.DO2 is set for a period of 6 minutes with a pulse signal at input L1.DI2.

**3 Multiple function switch with SLS:**

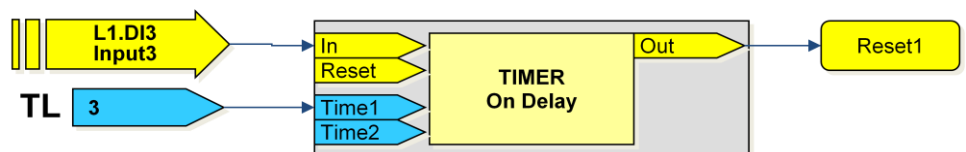
Output L1.DO3 is toggled with a pulse signal at input L1.DI3.



Output L1.DO3 is set for the period of a preset time "60 seconds".



The permanent lighting function is enabled by keeping the push button pressed for a specified time **TL** to reset the first time-relay.



Some examples of options to increase comfort or save energy:

- A flashing function indicating that the light is going to be switched off automatically can be used.
- Various central functions can be integrated :
  - Central off
  - Central on (panic button)
  - Control of all lamps or individual circuits by means of a daylight control switch
  - Controlling by means of an integrated timer (e.g. permanent lighting only until midnight; disabled at specific times)
  - Automatically switching off the permanent lighting on expiration of a preset period of time (e.g. after 3 hours)



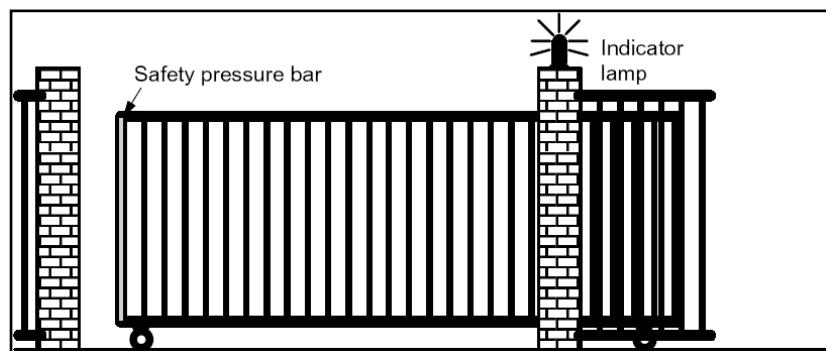
### Application 2: Factory door control system

The entrance to a company's premises is often closed with a gate. The gate is only opened to let vehicles pass and controlled by a porter.

#### Description:

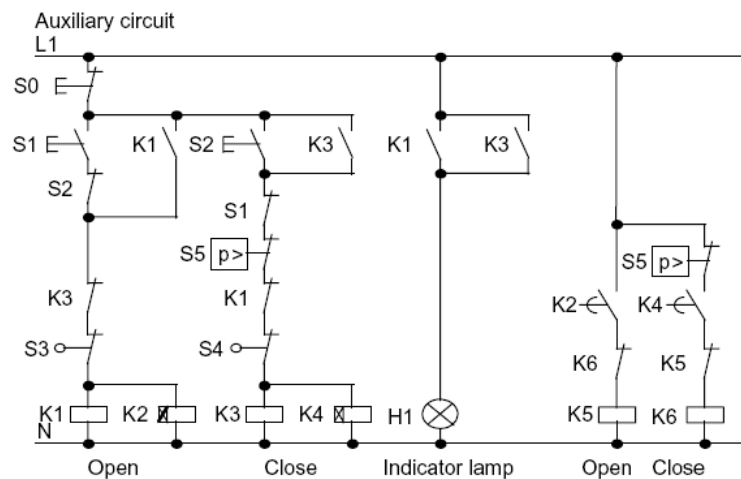
- The gate is opened and closed by means of pushbuttons in the gatehouse. At the same time the porter can monitor the operation of the gate.
- The gate is normally fully opened or fully closed. The gate motion can be interrupted at any time.
- An indicator lamp is switched on five seconds before the gate starts moving and when the gate is in motion.
- A safety pressure bar prevents harm to persons and objects from getting trapped or damaged when the gate is closing.
- Every gate is opened and closed by means of a control shaft. The gate will be totally opened or totally closed.

#### Drawing overview:



#### Conventional solution:

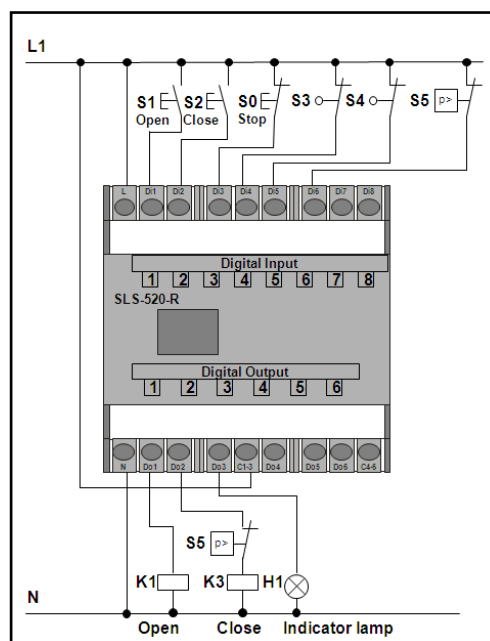
Various control systems are used to operate automatic gates. The circuit diagram below shows one of these options. A SLS can simplify this circuit considerably. The only thing that is necessary is to connect the Position sensors, limit switches and the contactor relays to the SLS-520.

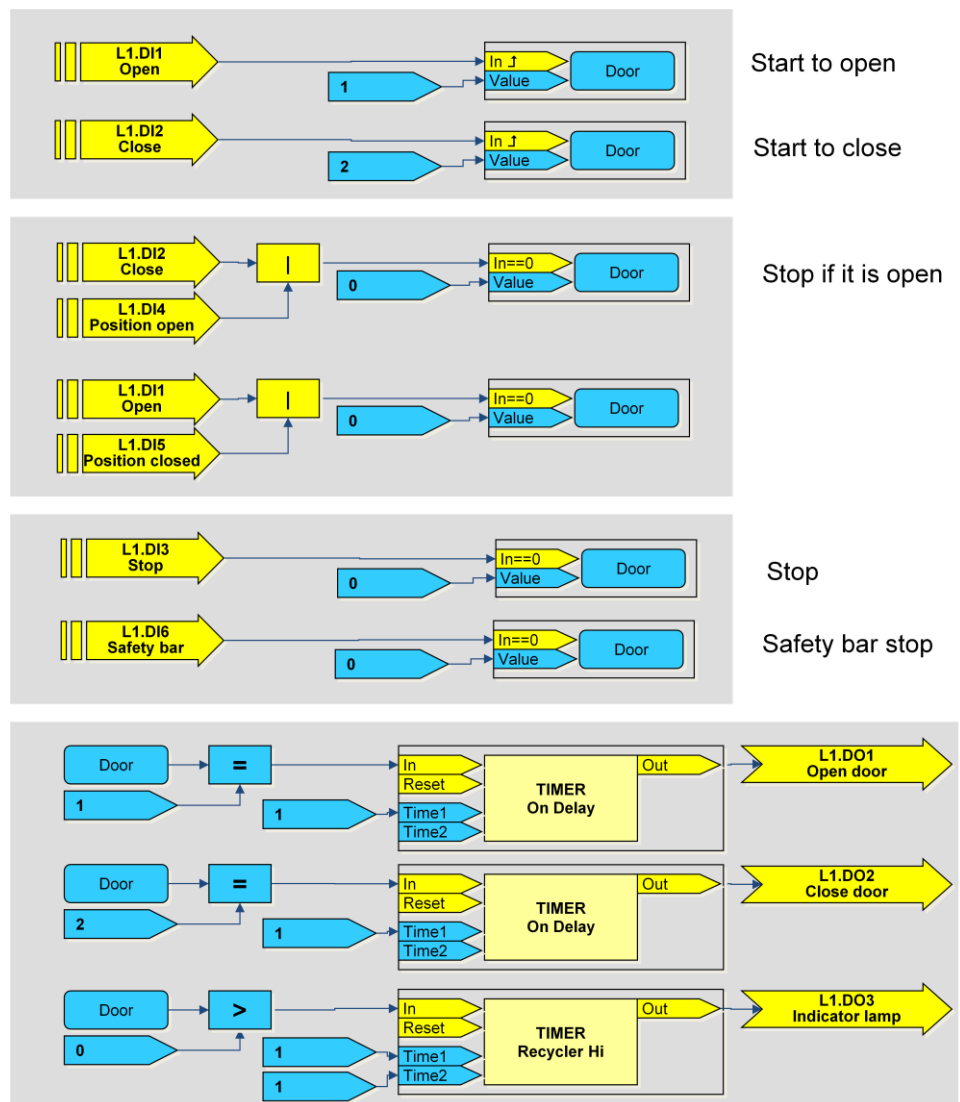


### Assignment:

Symbol	Description	State
K1	Contactor relay open	Output 1
K3	Contactor relay close	Output 2
S0	(Break contact) STOP Pushbutton	Input 3
S1	(Make contact) Open Pushbutton	Input 1
S2	(Make contact) Close Pushbutton	Input 2
S3	(Break contact) Position sensor open	Input 4
S4	(Break contact) Position sensor closed	Input 5
S5	(Break contact) Safety bar	Input 6
H1	Indicator Lamp	Output 3

### Hardware wiring:



**Solution:**

The OPEN or CLOSE pushbuttons initiate the gate motion. The gate is stopped by means of the STOP pushbutton or the corresponding limit switch. Furthermore a safety bar interrupts the closing motion of the gate if necessary.

### Application 3: Motor run forward and reverse application

#### Description:

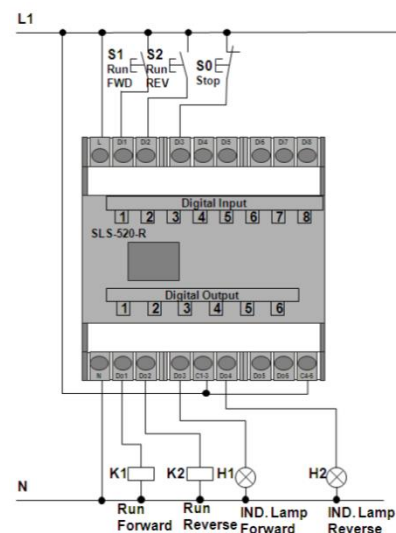
This is a forward and reverse circuit. Each direction's coil has its own start button and one stop button to stop any coil. At this circuit, interlocks are provided so that both outputs cannot be energized at the same time.

Directional pilot light indicators are present; each direction has separate light to know the motor direction.

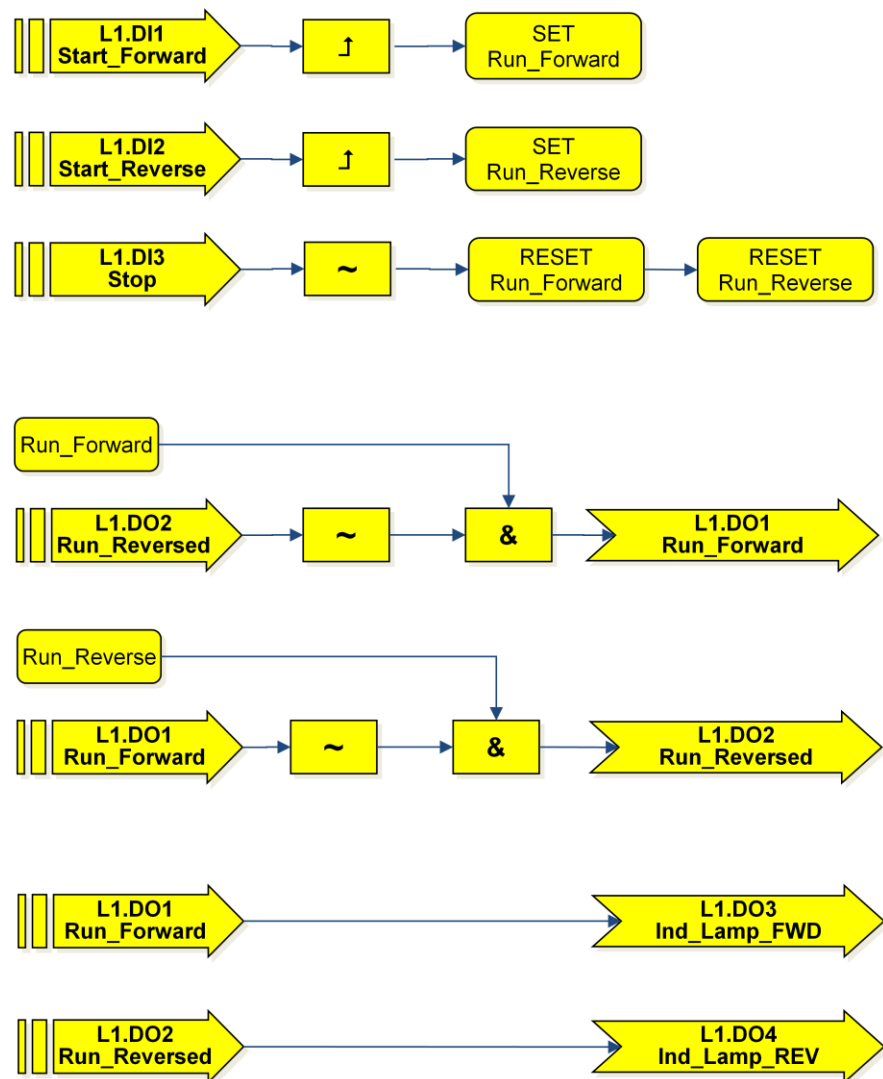
#### Assignment:

Symbol	Description	State
K1	Contactor relay Forward direction	Output 1
K2	Contactor relay Reverse direction	Output 2
S1	(Make contact) Start Forward Pushbutton	Input 1
S2	(Make contact) Start Reverse Pushbutton	Input 2
S0	(Break contact) STOP Pushbutton	Input 3
H1	Indicator lamp Forward direction	Output 3
H2	Indicator lamp Reverse direction	Output 4

#### Hardware wiring:



## Solution:



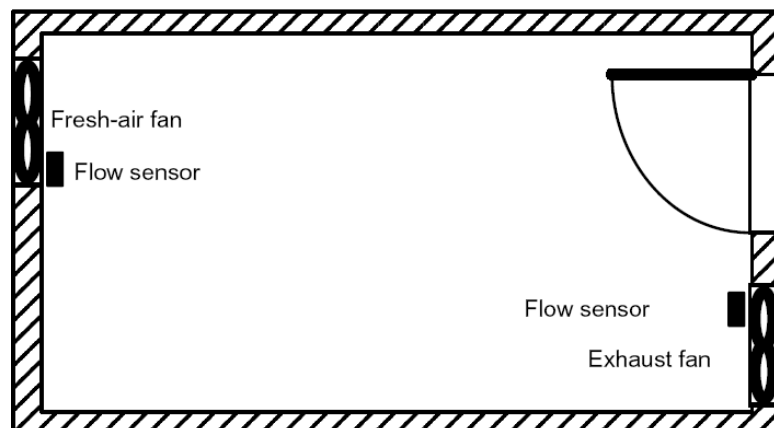
#### Application 4: Air conditioning system

Air conditioning includes the cooling and heating of air, cleaning it and controlling its moisture level: conditioning it to provide maximum indoor comfort. Air-conditioning systems supply a room with fresh air and at the same time exhaust the used air.

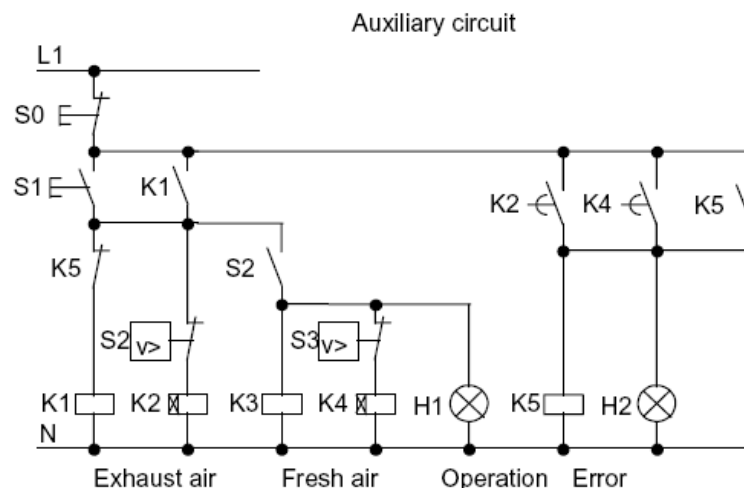
##### Description:

- The room is equipped with an exhaust fan and a fresh air fan.
- Both fans are monitored by means of a flow sensor.
- The pressure in the room never exceeds the atmospheric pressure.
- It must be guaranteed that the fresh-air fan is only switched on if the flow sensor signals no malfunction of the exhaust fan.
- A warning lamp indicates failure of a fan.

##### Drawing overview:



##### Conventional solution:



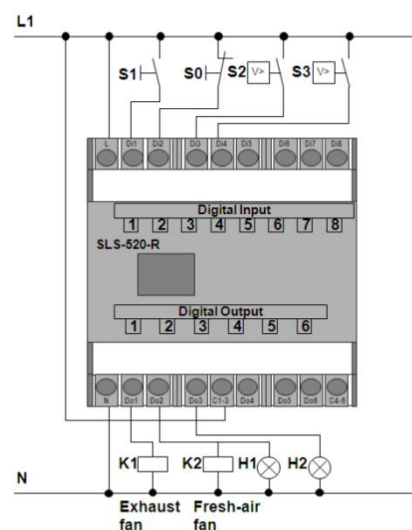
The fans are monitored by means of flow sensors. If no air flow is registered within a short waiting time, the system is switched off and the H2 output is an error message. This message can be reset by pressing the S0 button.

In addition to the flow sensors the fan monitoring system also requires an auxiliary circuit with several switching devices. This auxiliary circuit can be replaced by a single SLS unit.

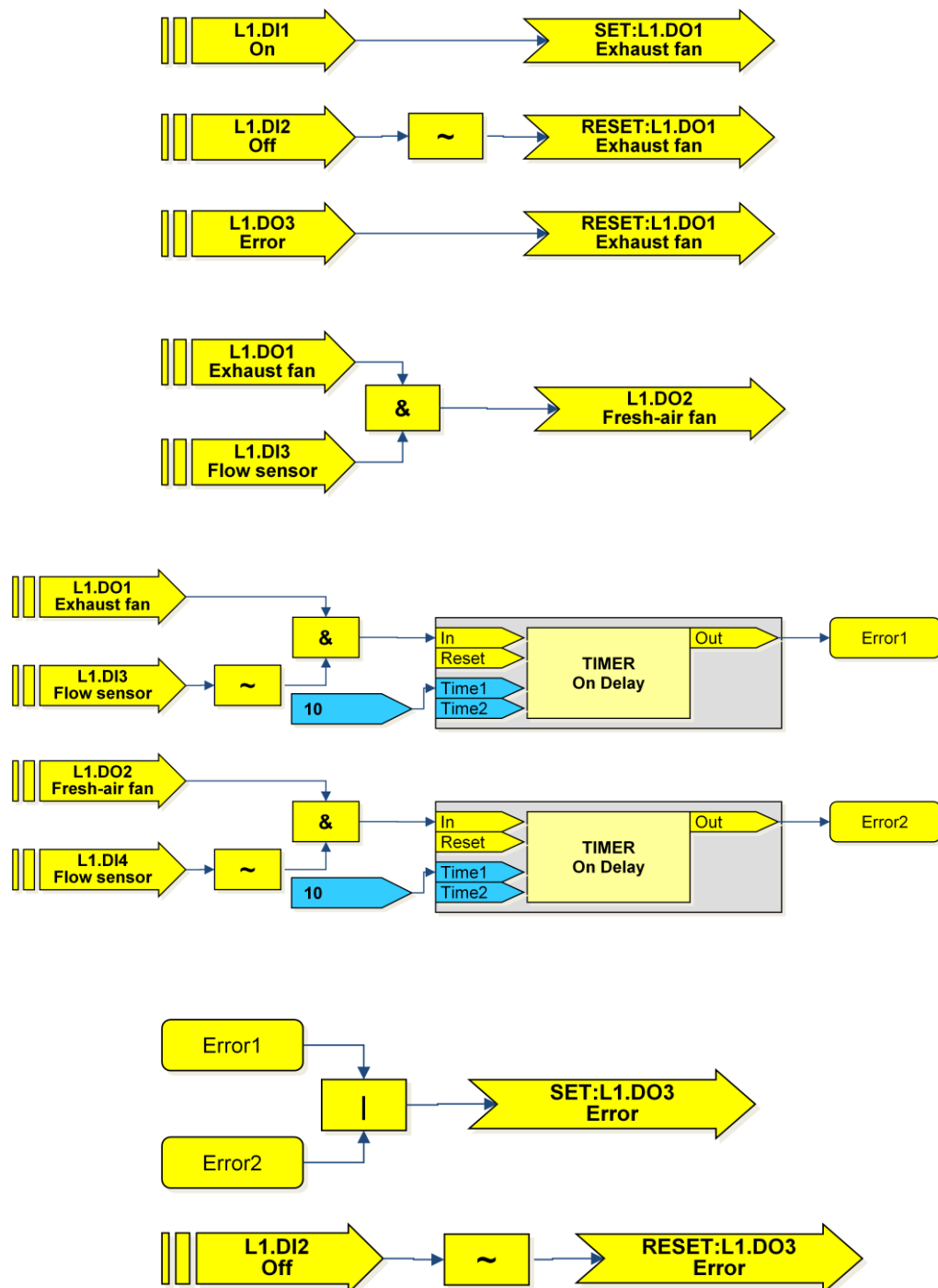
#### Assignment:

Symbol	Description	State
K1	Contactor relay Exhaust fan	Output 1
K2	Contactor relay Fresh air fan	Output 2
S0	(Break contact) STOP Pushbutton	Input 2
S1	(Make contact) Start Pushbutton	Input 1
S2	(Make contact) Flow sensor	Input 3
S3	(Make contact) Flow sensor	Input 4
H1	Indicator Lamp	Output 2
H2	Indicator Lamp	Output 3

#### Hardware wiring:



## Solution:





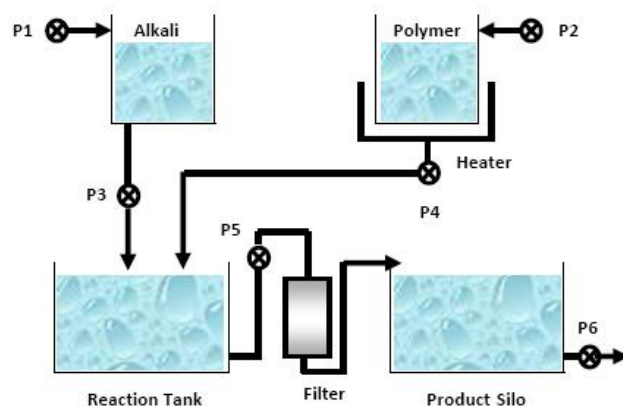
### Application 5: Chemical process plant control

#### Description:

The plant consists of four tanks with pumps to transfer the liquid contents through the system, each tank is fitted with sensors to detect “Empty” and “Full” states. Tank 2 has an integral heating element with associated temperature sensor. Tank 3 is equipped with a stirring arm to mix the two liquids when they are pumped from tank 1 and Tank 2, the lower tanks 3 and 4 have twice the capacity of Tanks 1 and 2, and will therefore be filled by the contents of tanks 1 and 2.

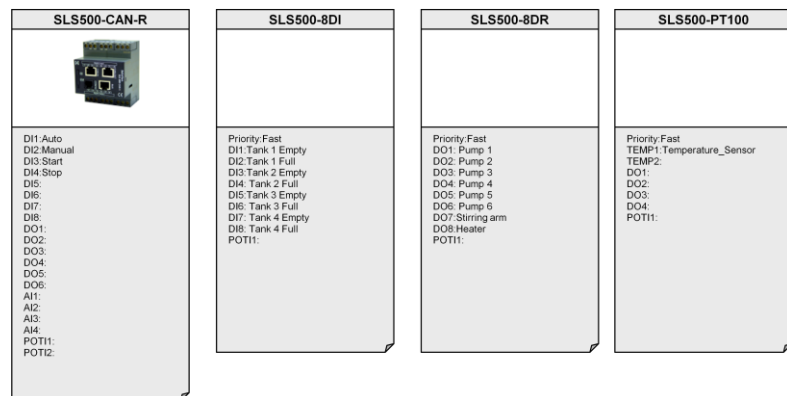
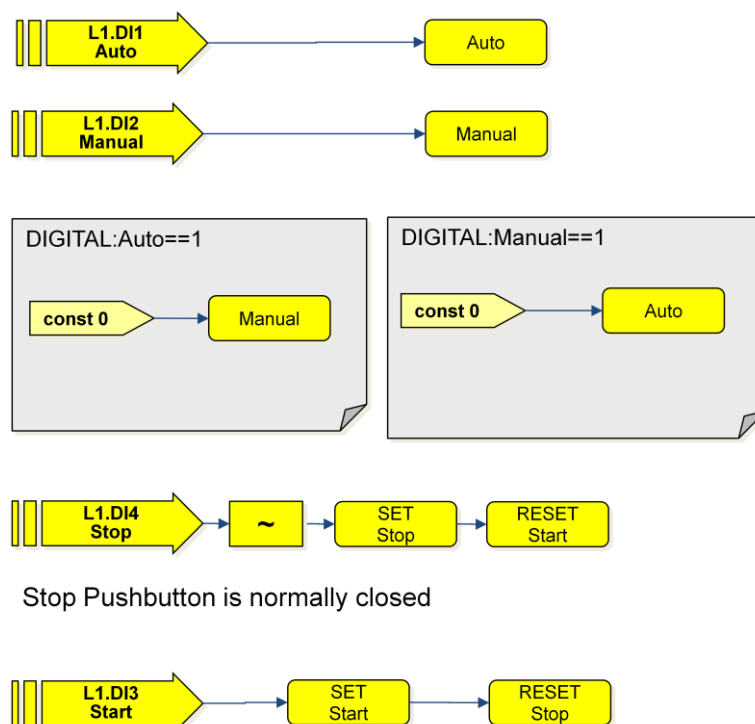
- Tanks 1 and 2 are filled from the supply by Alkali and Polymer respectively via pumps 1 and 2. Pumps 1 and 2 turns off when high level sensor indicates ON.
- The heating element in tank 2 is activated, increasing the polymer temperature up to 60 °C, and there will be temperature sensor type PT100 to measure the temperature, then should turn Off the heater and turn ON pumps 3 and 4 to transfer the liquids into the reaction vessel, Tank 3
- The stirring arm must also run when this tank is being used for 60 seconds. Once Tank 3 is full, Pumps 3 and 4 are stopped. If the stirring time is greater than 60 seconds, Pump 5 is working to transfer the mixture to Tank 4 (product silo) via a filter unit.
- Pump 5 is stopped once Tank 4 is full or Tank 3 is Empty. Finally the product is run off into storage using Pump 6.
- Then the cycle has to be start again.

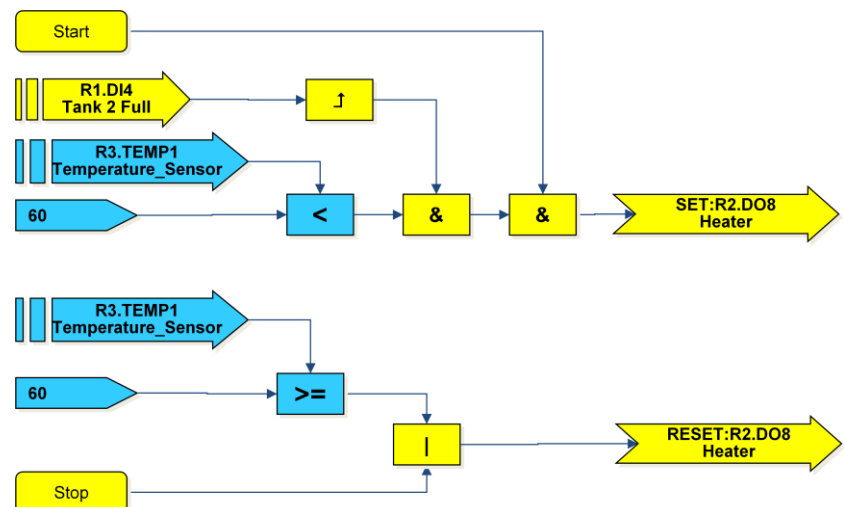
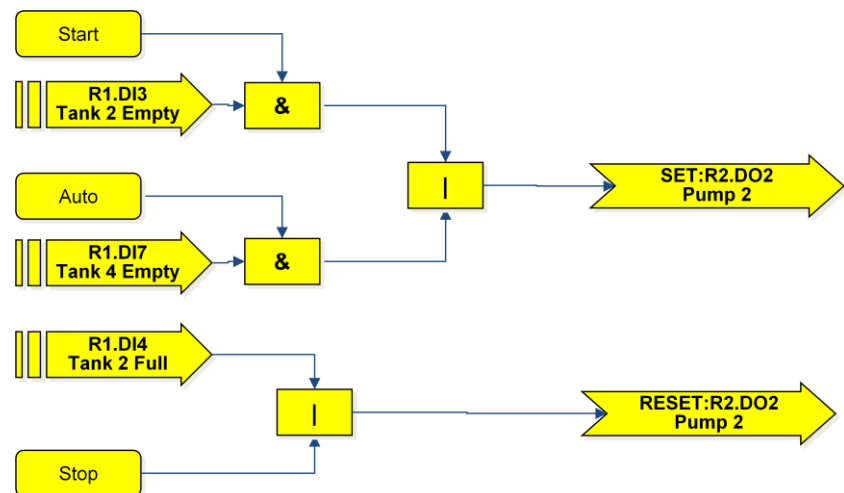
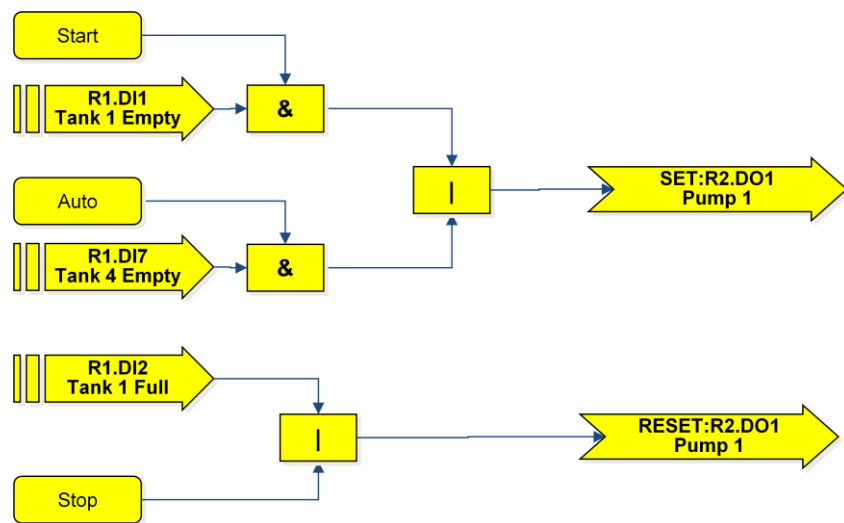
#### Drawing overview:

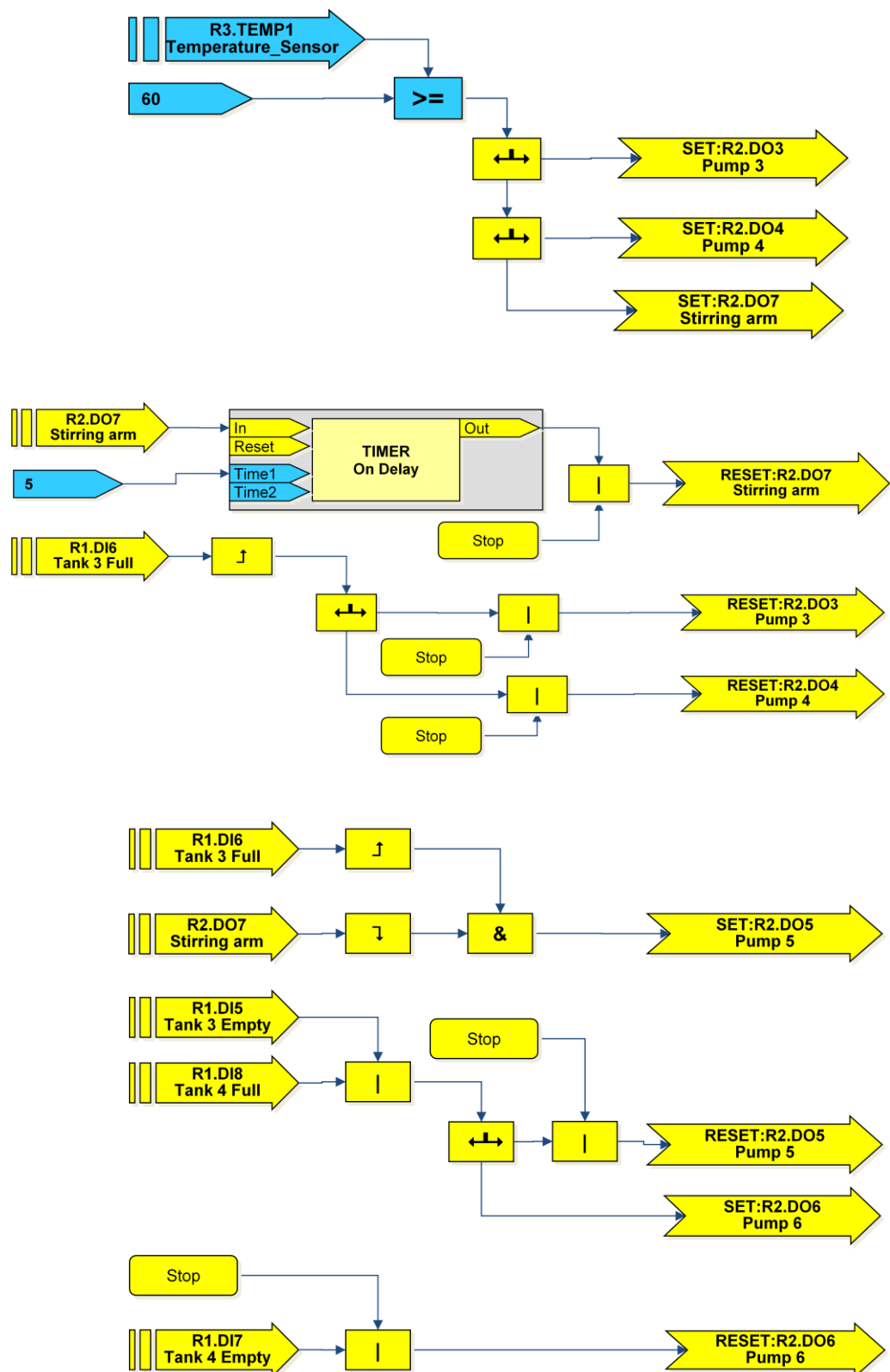


**Assignment:**

Symbol	Description	State
P1	Pump 1	Output 1
P2	Pump 2	Output 2
P3	Pump 3	Output 3
P4	Pump 4	Output 4
P5	Pump 5	Output 5
P6	Pump 6	Output 6

**Solution:****1 Hardware configuration:****2 Program:**





### Application 6: Service water pump

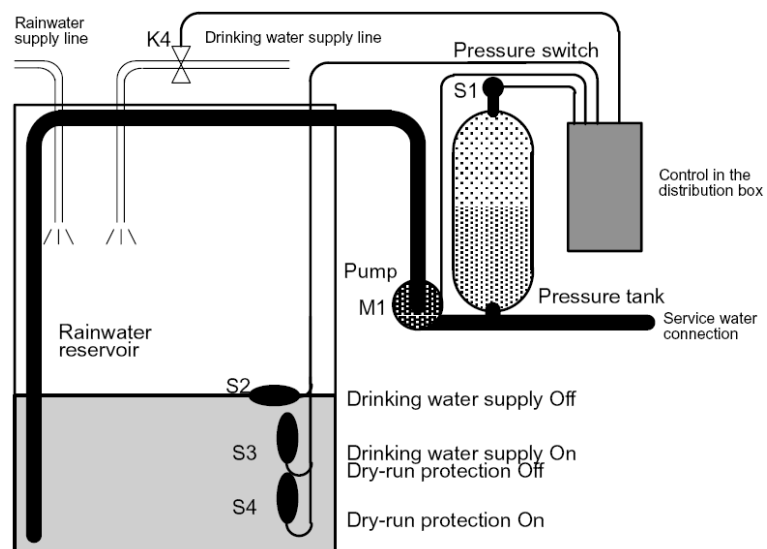
#### Description:

The use of rainwater, in addition to drinking water, is gaining importance in the domestic housing area as money is saved by reducing the demand for piped water and the environment is protected by reducing the volume and peak flows of storm water by providing onsite storage and temporary water holding capacity during storms.

Properly maintained rainwater tanks can supply good quality clean water for all in-house and garden uses including:

- washing clothes
- irrigation systems for gardens
- watering indoor plants
- car washing
- toilet flushing installations
- Bathing

The sketch illustrates how such a rainwater utilization system is operated:



In case of rain, the rainwater is collected in a reservoir. There is a pumping station supplies a local water pipe line system (Service water connection) from this tank. The water then can be tapped in the same way as normal drinking water. In case that the reservoir runs dry it can be topped up with drinking water by opening the solenoid valve (K4).

Service water must be available at any time. In case of emergency, the control system must automatically switch over to drinking water supply.

When switching over to the drinking water supply, the ingress of rainwater into the drinking water system must be prevented.

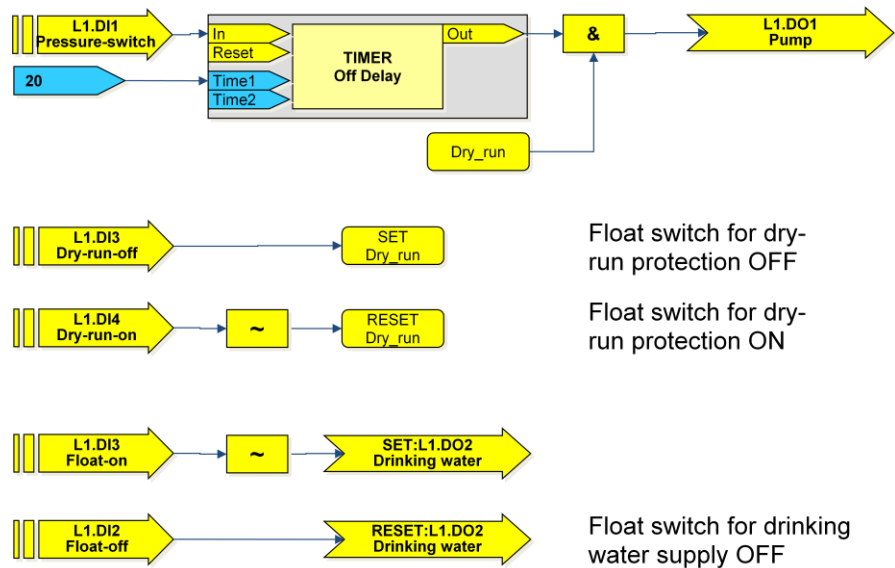
The service water pump may not be switched on if the reservoir has run low of rainwater (dry-run protection).

What is needed for this solution?

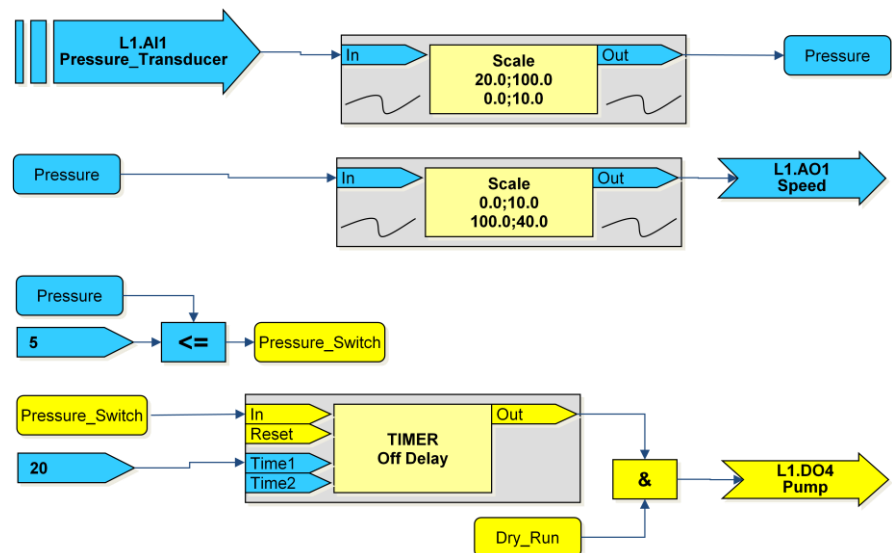
Apart from the SLS, all you need is a pressure Transducer to maintain a constant pressure and a frequency inverter to control the speed of pump and float switches to control the pump.

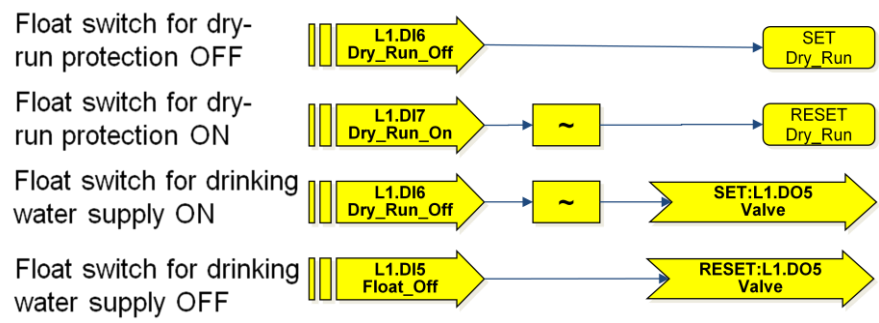
Solution:

1. In case of using just pressure switch and control the pump with a constant speed:



2. In case of using pressure transducer with frequency inverter:

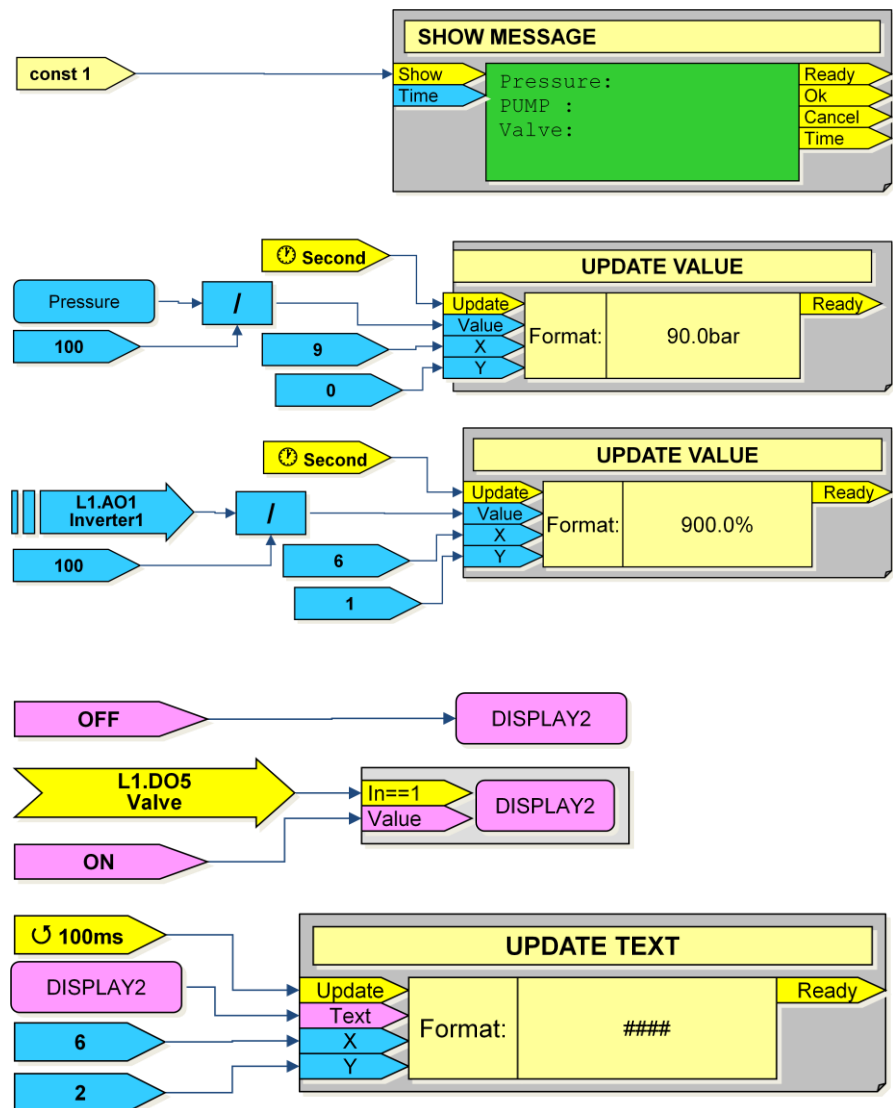






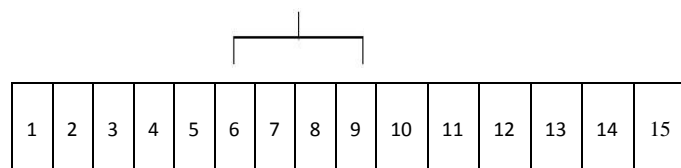
## 3. With text screen TERM 4:

In this case you can monitor if the pump is running or not and to see the speed of motor and also the valve is on or off.



**Exercises**

1. There are four outputs R, S, T and U. R starts immediately when an input is energized. S starts 4 seconds later. T starts 5 seconds later than S. U goes on 2 seconds after S. One switch turns all outputs off.
2. A machine, M, is to be turned on either when count A goes up to 11 or when count B goes up to 16, one stop button or switch resets the entire process.
3. A bottling process for 12 bottles operates as follows:
  - a. Bottles are counted until 12 are in position for filling.
  - b. When in position in the carton, the 12 bottles are filled simultaneously for 6 seconds.
  - c. After filling, there is a pause of 3 seconds for foam to subside.
  - d. The 12 caps are then put on and counted as they are installed.
  - e. A solenoid then pushes the completed carton of 12 on to a conveyer. The system is reset for a new group (to be restarted manually) of 12 bottles by a limit switch that indicates that the carton is out of the "fill" position and on the conveyer.
4. Devise a system to control the assembly line shown in figure. All 15 stations are to function as set up by one of two registers. Short and tall parts are sent down the line. Short parts get all 15 operations, if specified. Lines 6 through 9 are omitted for the tall parts only. Therefore, have operations 6 through 9 turned off when tall parts go by. Tall parts are detected by a limit switch just after station 5. After station 9, another limit switch unseals the MCR function, so the next part, whether larger or small, is again set up for all operations.



## 5. Compressor control

The example program implements a compressor control to maintain the pressure in a compressor air tank at the required level, the compressor motor is started with the star-delta starting and providing with an oil pressure switch to monitor the operating pressure of the oil lubrication.

When the compressor is switched on the correct operating pressure of the oil must be reached after a specified delay, so that the oil pressure sends a “1” signal to the PLC. When the compressor is switched off again, the oil pressure drops again after a delay, resulting in a “0” signal being sent to the PLC. If the oil pressure is not correct, the PLC switches on the fault indicator lamp. The reaction to the fault depends on whether the compressor is being operated in manual or automatic modes. The fault indicator lamp can be reset by pressing a reset button; this causes the fault indicator lamp to go out however the oil pressure fault itself is not remedied simply by pressing the reset button.

A rotary switch is used to choose between automatic and manual operation. When the compressor is running in automatic mode and the rotary switch is turned to manual mode, the compressor switches off automatically.

### Manual mode

The manual mode is used to test the compressor during commissioning, and it allows you to bring the compressor up to nominal operating pressure for testing purposes and to stop it again, this is done with two pushbuttons.

If the 1<sup>st</sup> pushbutton is pressed at this mode, the compressor is started using star-delta starting method. As soon as the nominal operating pressure has been reached, contact 2 closes on contact manometer which switches the compressor off again. If an oil pressure fault occurs the fault indicator lamp is switched on, but the PLC does not switch off the compressor, the fault indicator lamp can be cleared by pressing the reset button.

### Automatic mode

In this mode the PLC regulates the operating pressure with the help of the contact manometer. The contact manometer has two contacts, as follows:

Contact 1 closes when the pressure drops below the nominal operating pressure.

Contact 2 closes when the nominal operating pressure has been reached.

If the compressor is switched on, the motor runs until the pressure in the compressed air tank has reached the nominal operating pressure, causes contact 2 of the contact manometer to close which switches off the compressor again.

If the pressure drops below the nominal operating pressure, this closes contact 1 of the contact manometer which causes the compressor to be switched on again.

If the OFF pushbutton is pressed in this mode, the compressor is switched off as long as the pushbutton remains pressed.

If an oil pressure fault occurs, the PLC switches off the compressor and switches on the fault indicator lamp. If the motor protective circuit breaker has tripped a signal is sent to the PLC which also switches off the compressor.

## 6. Green house temperature and ventilation control:

Task: to automatically open and close the roof lights of a greenhouse in order to adjust the ventilation & temperature warm air should be blown in via the heating system when the temperature drops below a certain level. The drive motors for the fans and roof lights must be monitored for faults which should also be signaled by a flashing light.

Functional description:

The greenhouse is also used as a display and sales area. The roof lights are opened for ventilation and are closed again depending on the temperature.

### Ventilation control

All the roof lights are actuated by a three phased AC motor M1 with a reversing contactor circuit. The end positions are detected by limit switch S2 "OPEN" and S3 "closed". The motor switches off when the limit switch is reached.

### Warm air supply

When the temperature in the greenhouse falls below a certain level, the fan motor M2 is automatically activated to blow in warm air. The motor is switched off again when the temperature returns to the desired level.

### Failure of a motor

If M1 or M2 fails, the contact of the corresponding trip-indicating auxiliary contact Q1 or Q2 opens. The fault is signaled via the flashing light H1 for both motors.

### Continuous ventilation

Key switch S1 is used switch off the automatic temperature control and select "continuous ventilation". It may be necessary to first close the roof lights and then open them again in order to use this function.

It should be possible to enter the motor run time T2 which determines how far the roof light is opened.

The roof lights can be opened as far as the end position. The default for T2 is 4 seconds.

### Manual operation

For maintenance and repairs, the windows can be opened via the P2 button "Up arrow" and closed via P4 "Down arrow".

The P buttons are activated in the Special menu.

**Notes:**

[illegible]

[illegible]







ALMAWARED  
ENGINEERING  
& TRADING SAE

2 Ahmed Taysser St., Heliopolis, Cairo  
Tel. : 24192480 | Fax : 24192483  
almawared@link.net www.met-eg.com