

NEWEL 3



PLC FUNCTIONS

HELP

Digitel reserves the right to modify the technical characteristics described without prior notice.

Non-contractual document

Digitel SA

All rights reserved

Table of contents

1.	DESCRIPTION OF PLC FUNCTIONS	4
2.	COMPATIBLE HARDWARE AND SOFTWARE	4
3.	EXAMPLE OF USE	4
4.	INPUTS AND OUTPUTS	5
4.1.	Possible Inputs	5
4.2.	OUTPUTS	5
5.	HOW TO CREATE A PLC FUNCTION	6
5.1.	Modification of parameters by a user	14
6.	PLC FUNCTIONS PROGRAMMING LANGUAGE REFERENCE	16
6.1.	Structures	16
6.1.1.	Comments	16
6.1.2.	Variable assignation	16
6.1.3.	Operators	16
6.1.4.	IF..THEN..ELSE..	18
6.2.	Predefined functions	19
6.2.1.	digAlarmSet	19
6.2.2.	digAlarmGetState	20
6.2.3.	digMessageSend	20
6.2.4.	digTrace	21
6.3.	System variables that can be used in PLC functions	23
6.4.	System constants that can be used in PLC functions	24
6.4.1.	CONTROLLER_OUTPUT	24
7.	HELP BUTTON	26
8.	IMPORTING AND DUPLICATING FUNCTIONS	27

9. SAVE AND RESTORE FUNCTIONS	28
10. EXAMPLES	30
10.1. Machine room ventilation	30
10.2. Thermostat	31
10.3. Cyclic timer	32
10.4. Using the timers predefined in the central unit	33

1. DESCRIPTION OF PLC FUNCTIONS

The PLC functions allow certain tasks to be automated in a similar way to that of a PLC (Programmable Logic Controller). These functions are programmed in TelesWin with a language that is easy to understand and write and are stored in the DC58 central unit and executed every two seconds. These functions make it easy to adapt the operation of the system to suit your needs.

The Digital solution makes it possible to use all the inputs and outputs connected to the network, whatever their use (regulation of cooling units, compressors, etc.) In comparison, traditional PLCs can only act on their own inputs and outputs and not those of other network elements.

PLC functions make it possible to create new functionalities or extend existing ones, even after commissioning and during normal operation of the installation, without affecting infrastructures such as cabling. A rich library of pre-programmed functions allows you to perform various and complex tasks.

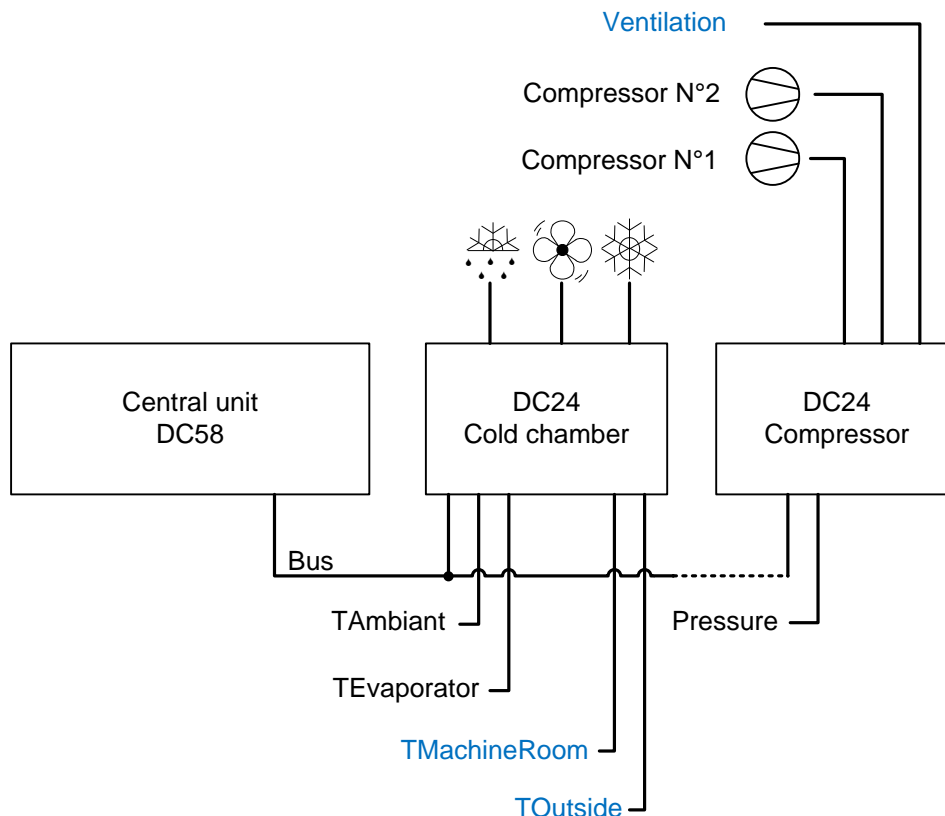
The programming language we use is based on the Windev Mobile language developed by PC Soft. This manual describes only the basic operators, the most frequently used. They are sufficient to create simple functions. To create advanced functions, we recommend that you study the Windev Mobile manual, which contains a detailed description of all operators and functions of this language.

2. COMPATIBLE HARDWARE AND SOFTWARE

- TelesWin with software revision 22.73-19.45.1 or higher.
- DC58 Central unit with software revision 19451 or higher.
- DC24 D/DE/E/EE regulator with software revision 19451 or higher.

3. EXAMPLE OF USE

- Control of machine room ventilation using 2 free cold room controller inputs and one free compressor controller output.



Setup of custom fonctions

Variable name	Description	Type	Value
TMachineRoom	Machine room temperature	I/O of a module	M 1.1/Temperature probe E (°C)
TOutside	Outside temperature	I/O of a module	M 1.0/Ambient temperature (probe A) (°C)
Ventilation		I/O of a module	M 1.0/Contact C2
Setpoint		Parameter	20
Delta		Parameter	1

Code

```
// Machine room ventilation
IF {{TMachineRoom}} > ( {{Setpoint}} + {{Delta}} ) AND {{TOutside}} < {{TMachineRoom}} THEN
  {{Ventilation}} = 1
ELSE
  IF {{TMachineRoom}} < {{Setpoint}} OR {{TOutside}} > {{TMachineRoom}} THEN
    {{Ventilation}} = 0
  END
END
digTrace({"Ventilation = "} + {{Ventilation}})
```

Cancel OK Apply Aide

4. INPUTS AND OUTPUTS

PLC functions allow actions to be performed on outputs (O) according to input values (I). All network I/Os can be used.

4.1. POSSIBLE INPUTS

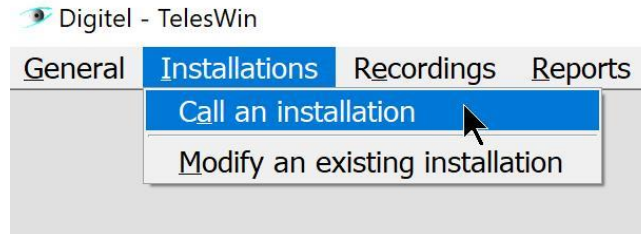
- Measurement of physical parameters (temperature, pressure, humidity, luminosity).
- Timers defined in the central unit.
- Fixed parameters, set by the programmers.
- Settings configured by users.

4.2. OUTPUTS

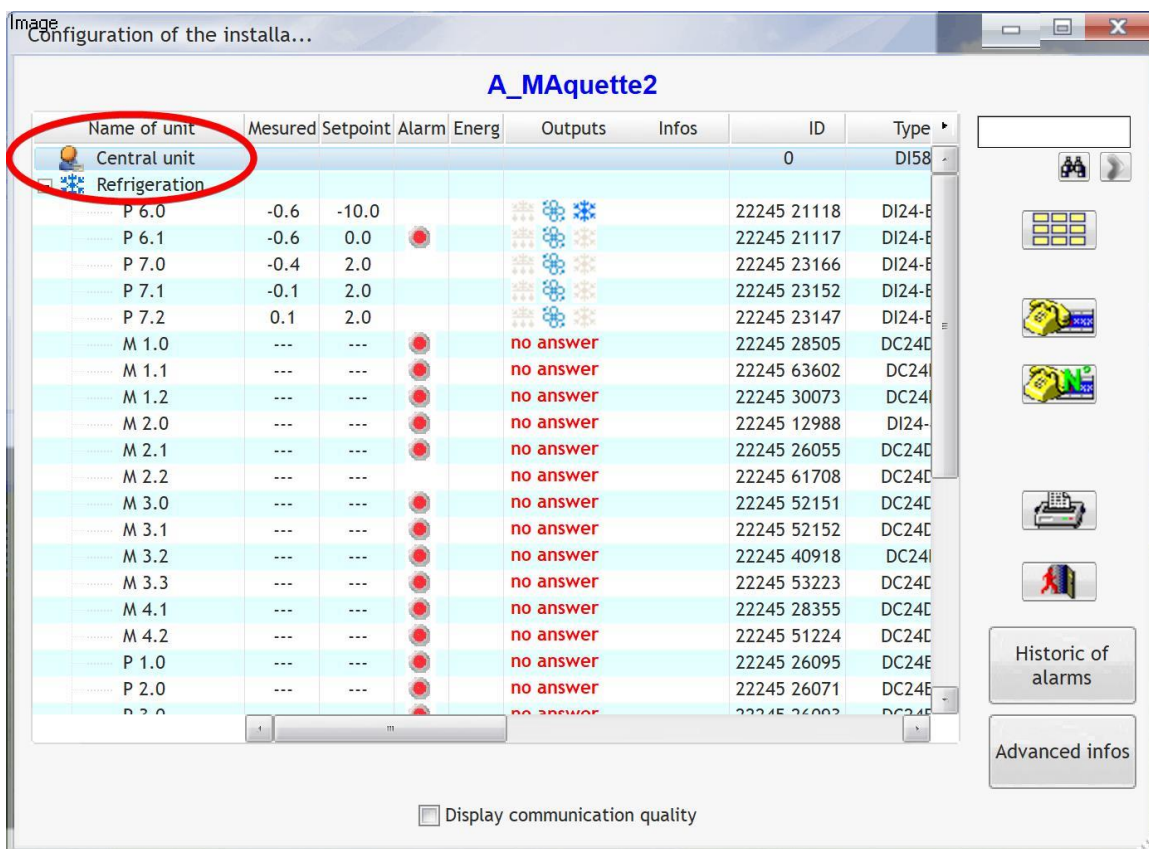
- Action on the digital outputs of satellite units (fans, compressors or relays on/off switching).
- Control of analog outputs.
- Activation of alarms defined in the central unit.
- Sending SMS messages.
- Sending mails.
- Displaying messages and values in a console on TelesWin.

5. HOW TO CREATE A PLC FUNCTION

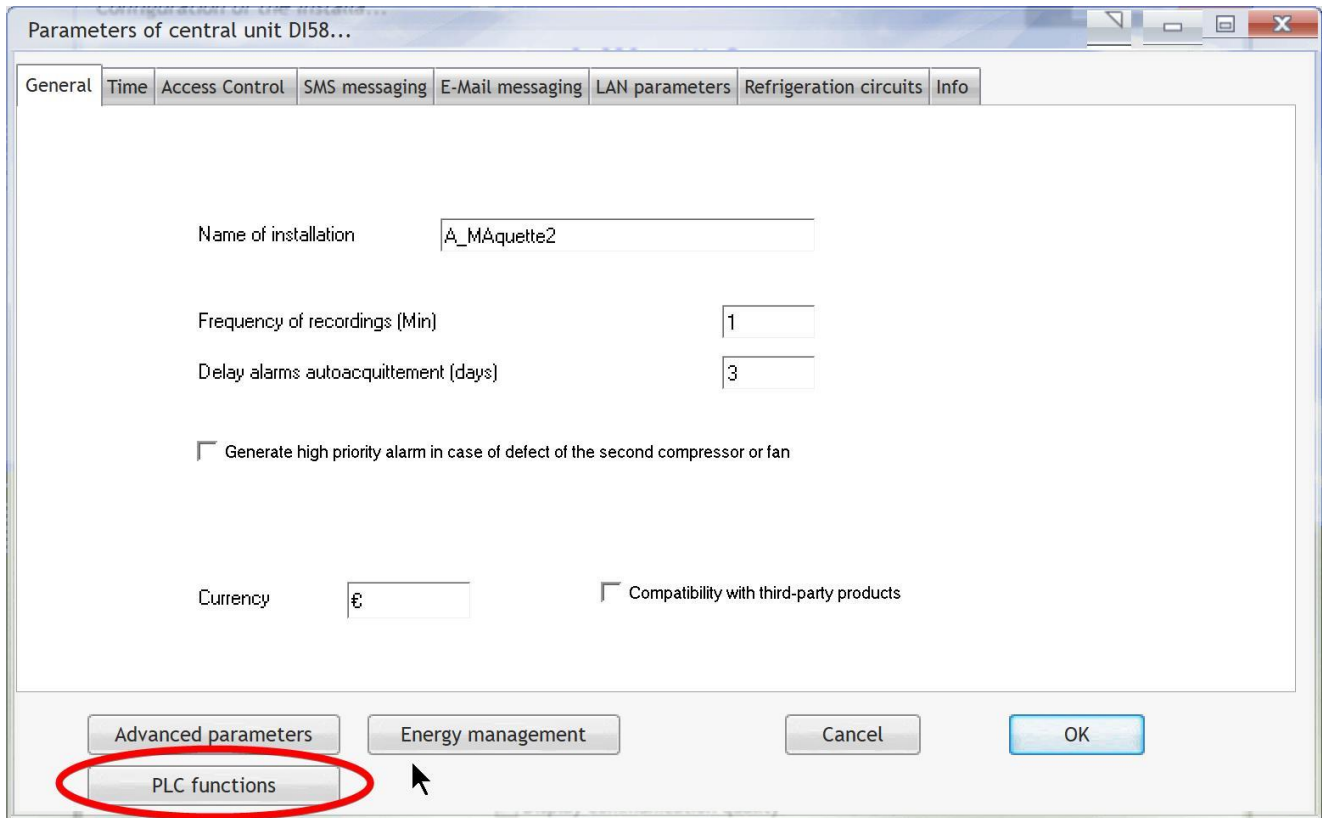
- Only the person with "Configuration" authorization for the installation (see on the central unit "Access control") can create and modify PLC functions. It must also have a dongle with the option "Configuration of plants" checked.
- Open the window **Configuration of the installation** (in the menu **Installations / Call an installation**).



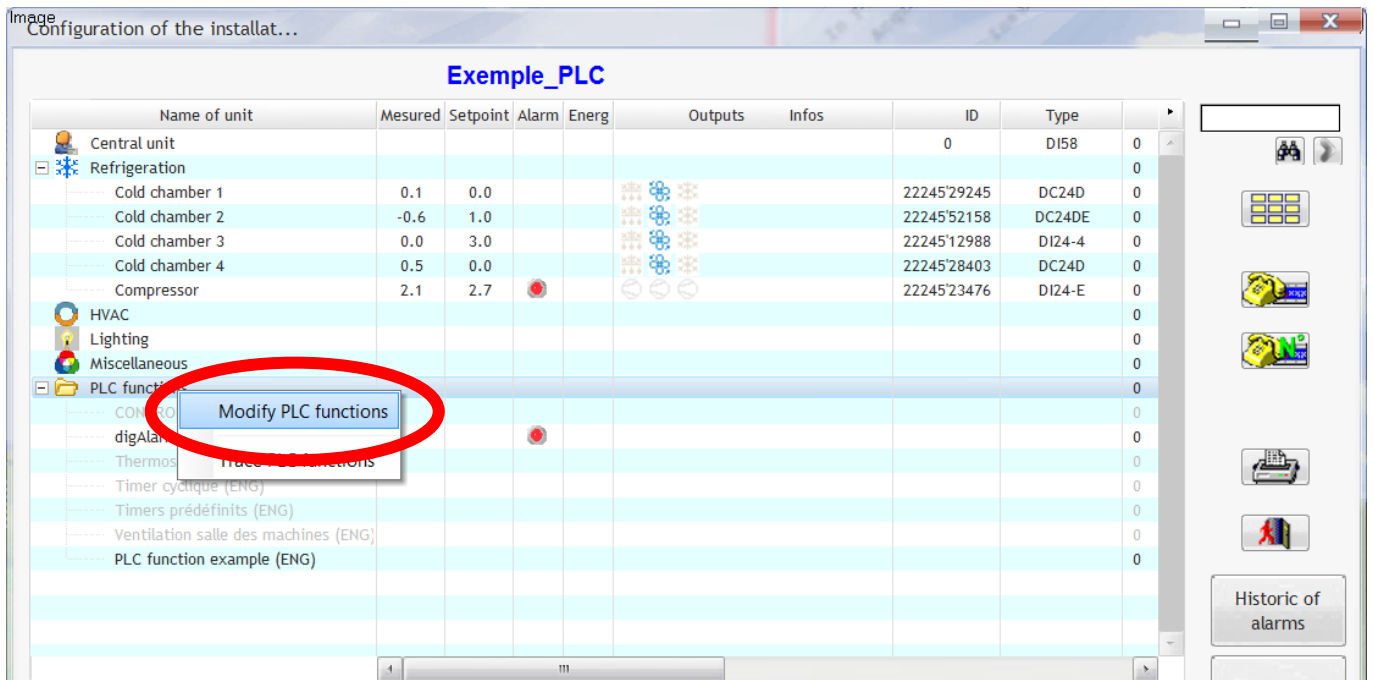
- Double click on **Central unit**.



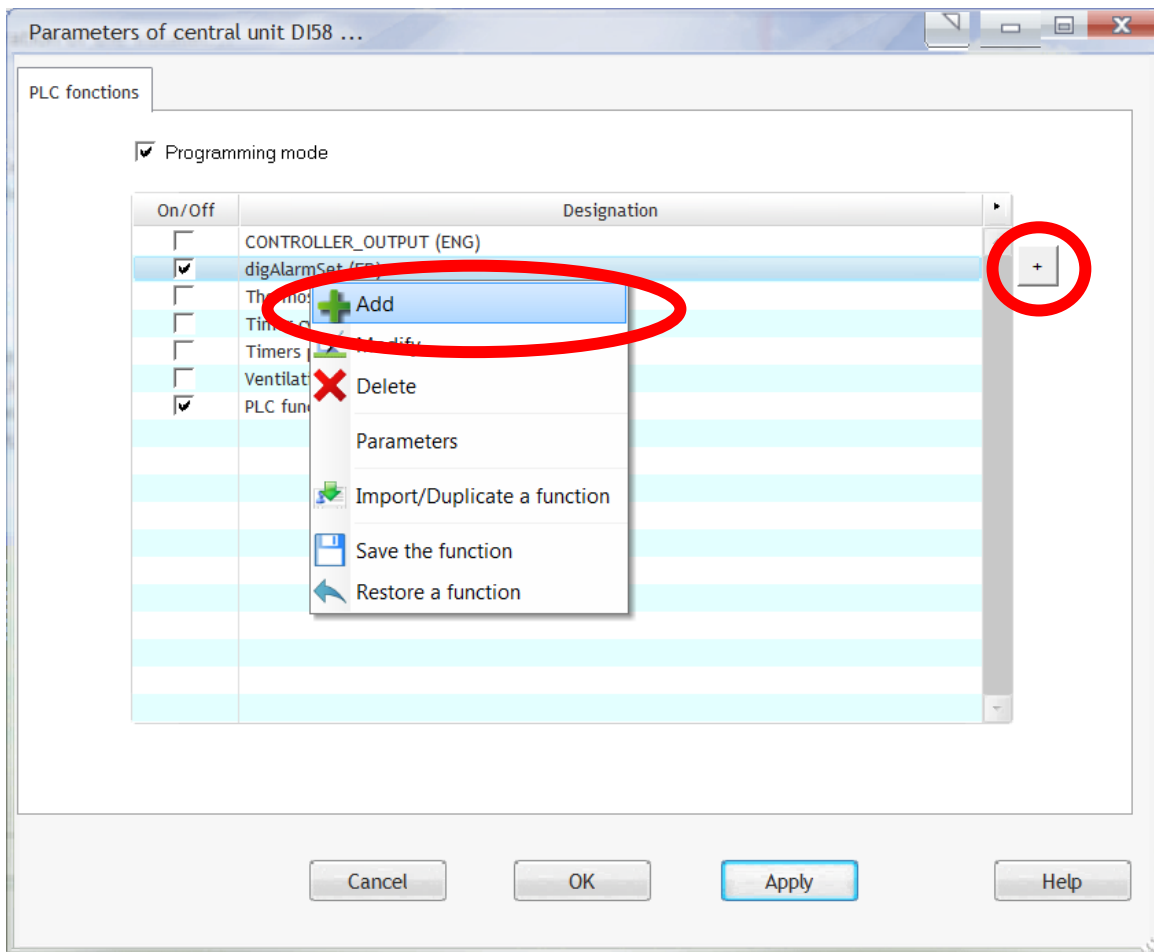
- The window **Parameters of central unit** opens.
- Click on the button **PLC functions**.



- Or click on **Modify PLC functions** in the context menu accessible by right-clicking on **PLC functions** or on one of the PLC functions present.



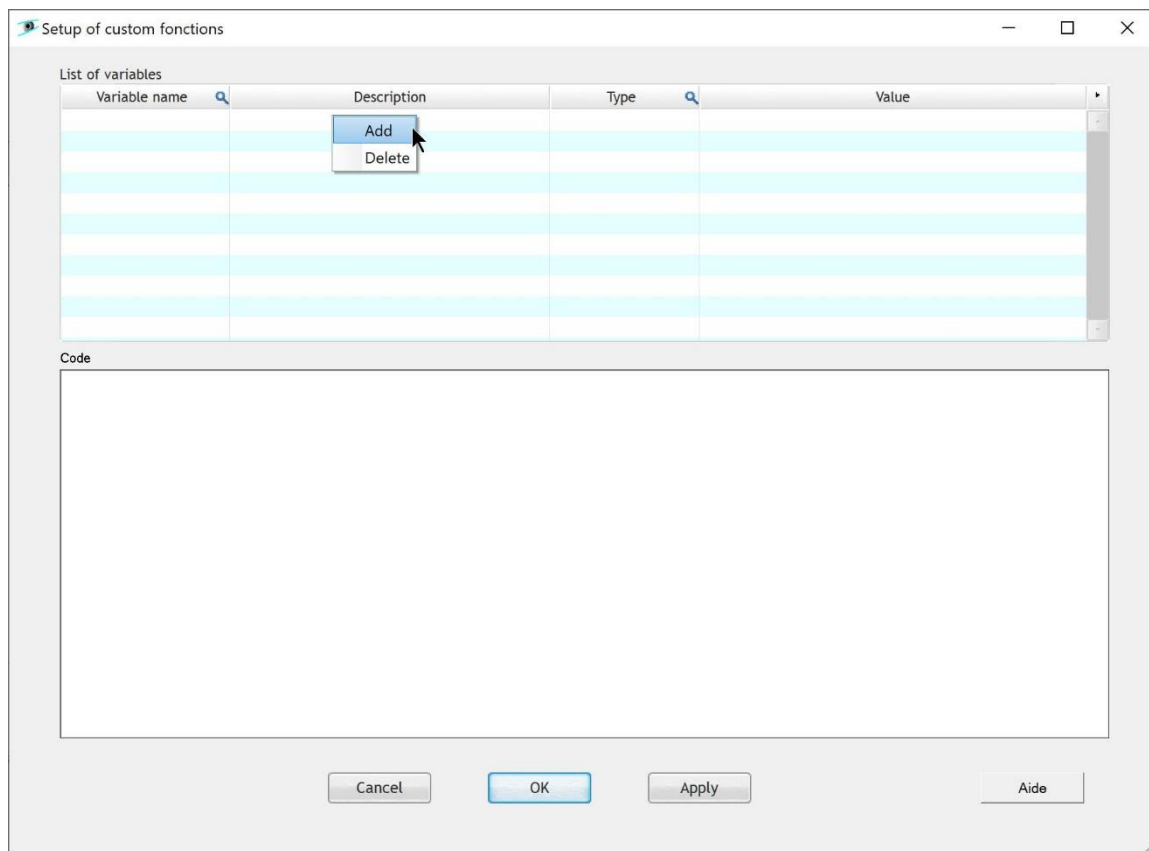
- The window **Parameters of central unit -3** opens.
- Click the **+** button to add a function. It is also possible to use the **Add** function of the context menu that can be shown by right clicking on the table.



- A popup pops out.
- Enter the name of the new PLC function. For example, **PLC function example**.
- Click OK.



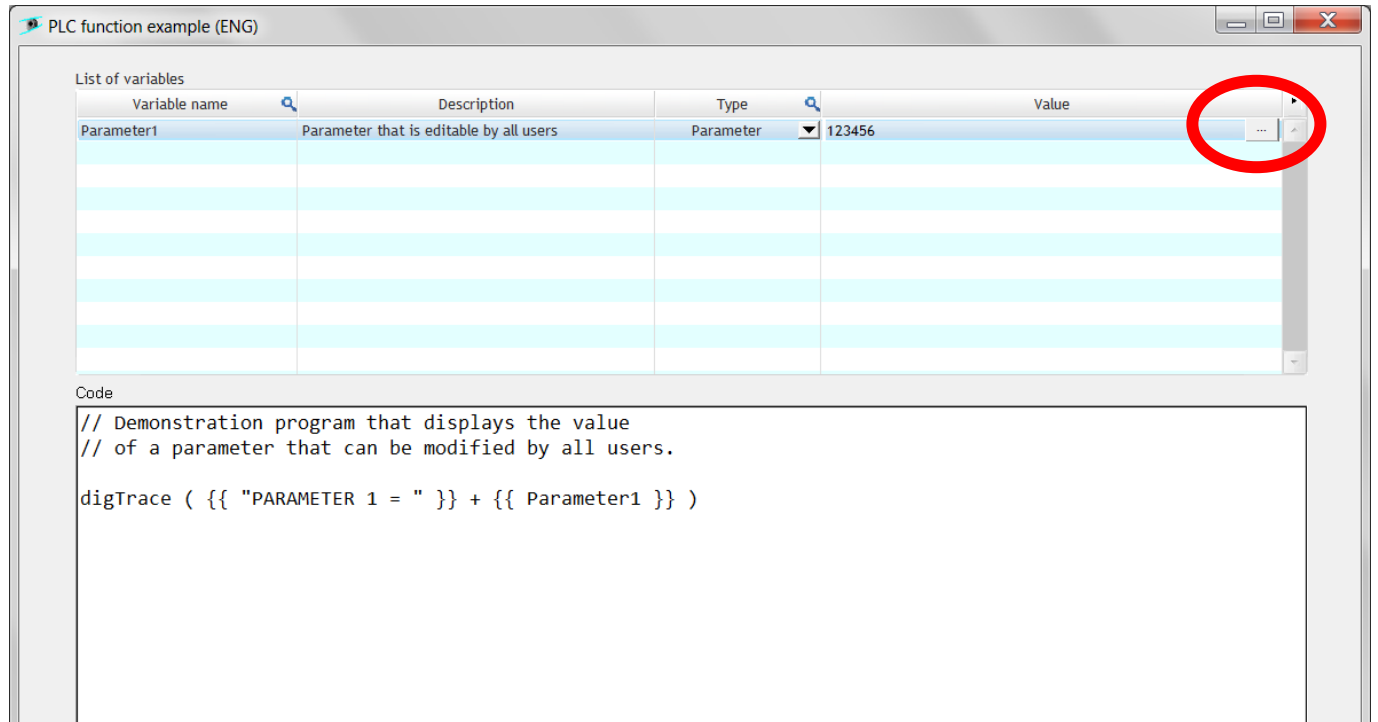
- The **Setup of custom functions** opens.
- This window allows you to create a PLC function.
- The **List of variables** table allows you to declare the variables that will be used by the PLC function.
- The **Code** field is used to enter the code.
- Right-click in the **List of variables** table and click on **Add**.



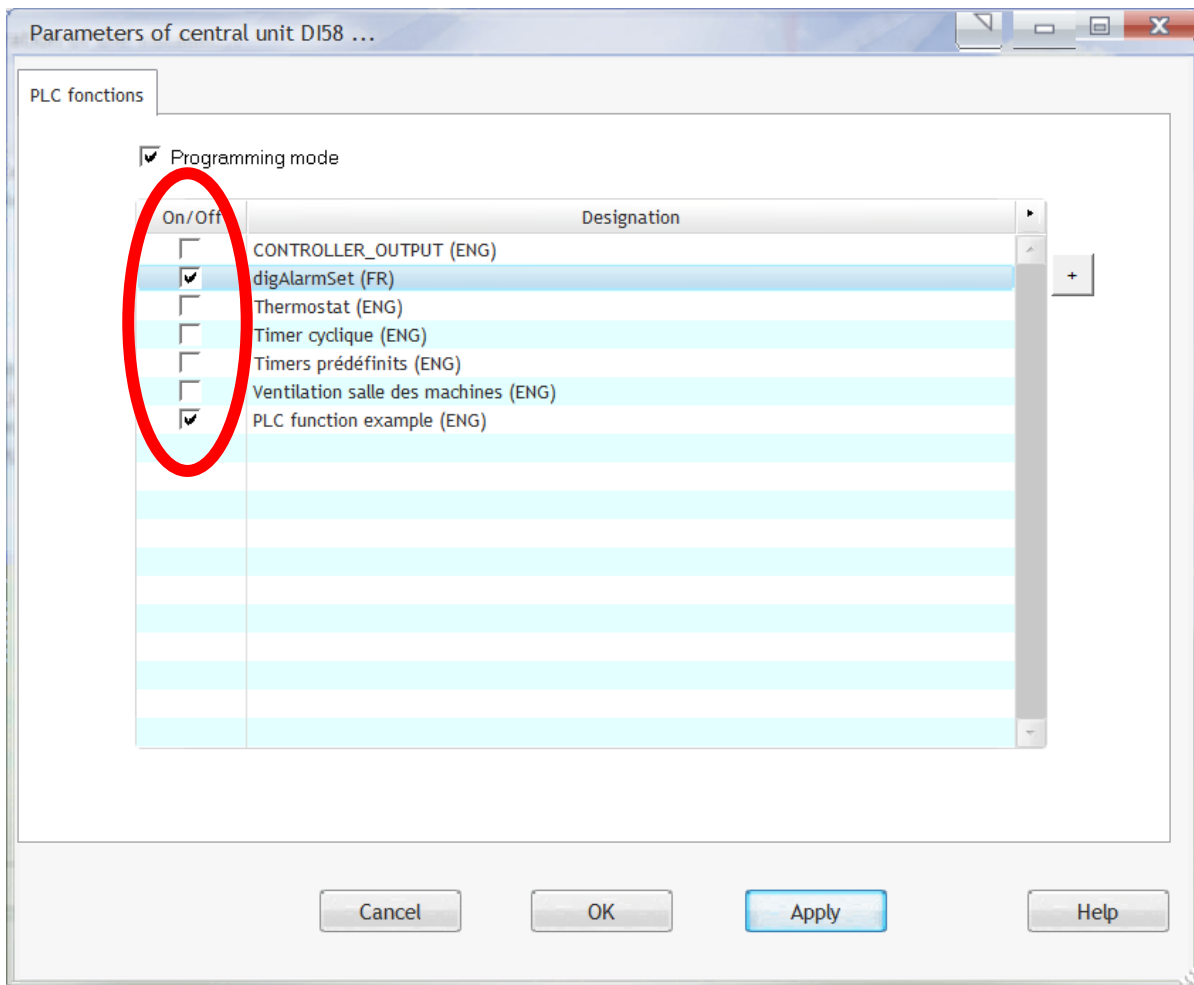
- Add variables in the **List of variables** table so that it looks like the image below.
- To fill in the field **value**, it is necessary to click on the button to the right of the field (see image below).
- Also fill in the **Code** field. You can copy and paste the example below.

```
// Demonstration program that displays the value
// of a parameter that can be modified by all users.

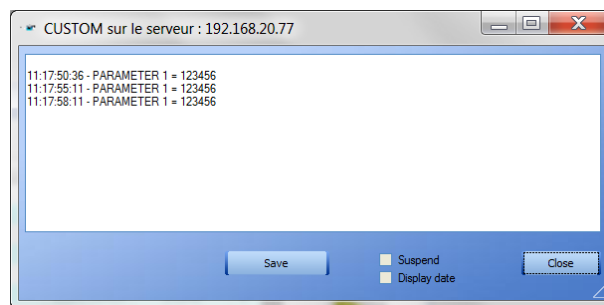
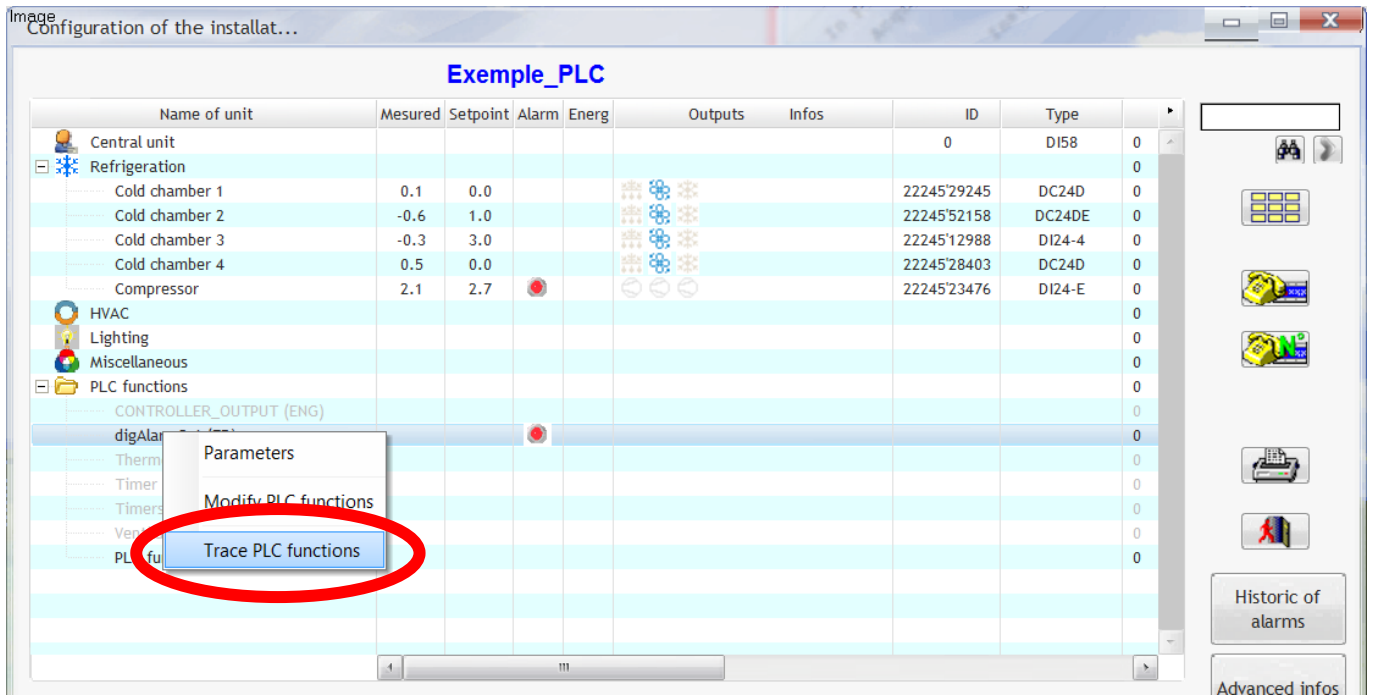
digTrace ( {{ "PARAMETER 1 = " }} + {{ Parameter1 }} )
```



- Once you have entered the parameters and code, click on OK.
- Click on the **ON/OFF** checkbox to the left of the function you just created to have it executed. The execution period is approximately two seconds.



- To follow the progress of the program execution, select **"Trace PLC functions"** from the PLC Functions context menu. A "Trace" window opens and displays the text entered in the digTrace function.
- We can see that the PLC function displays texts that do not change over time at the moment.
- We will later see more interesting examples, with measured values at the input and relays at the output.



- Once the function has been created and activated, it can be seen in the complete visualization of the installation. The active PLC functions are indicated in black and those that are disabled in grey. The red dot in the **Alarm** column indicates that this PLC function has generated an alarm.

Configuration of the installat...

Exemple_PLC

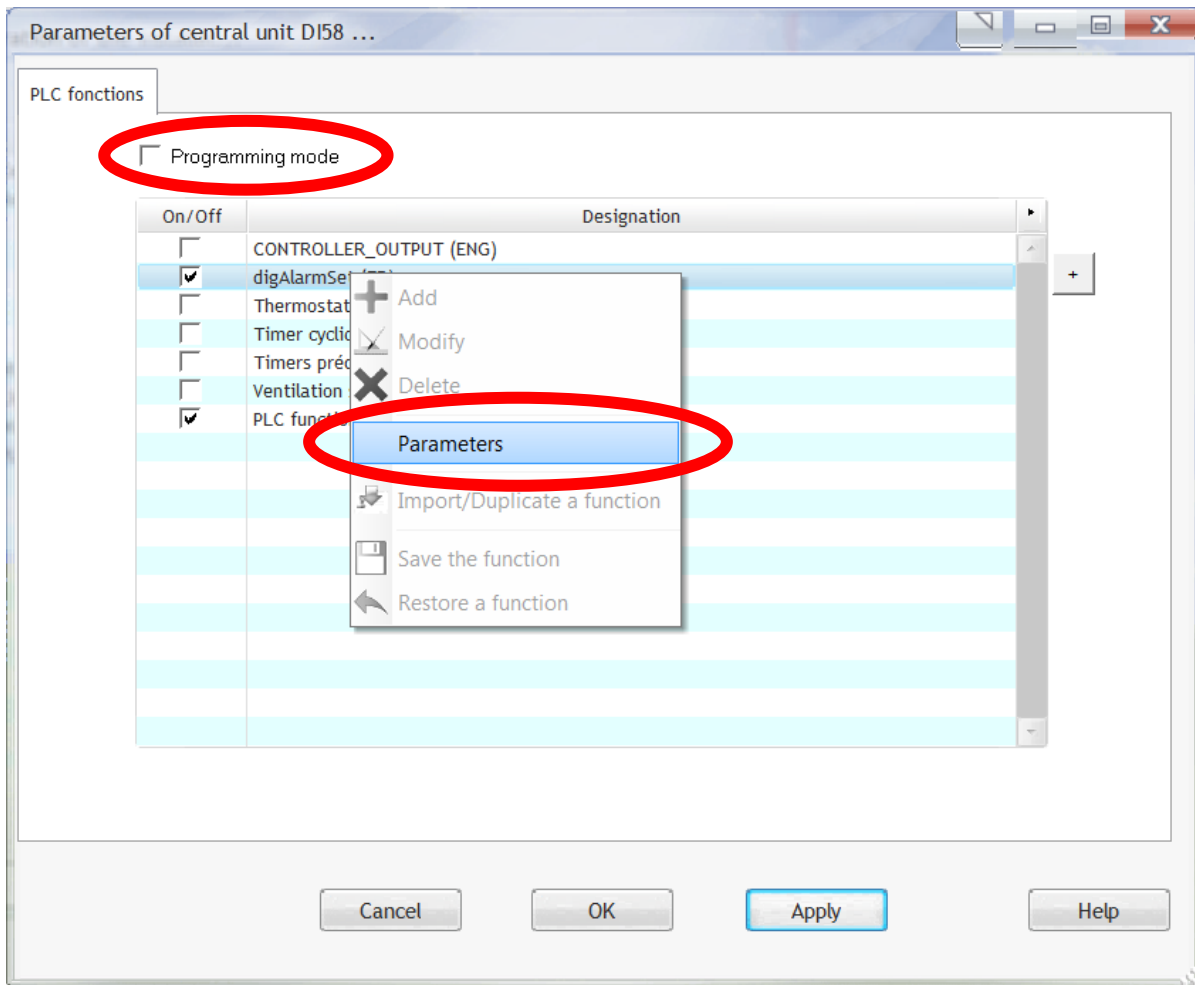
Name of unit	Mesured	Setpoint	Alarm	Energ	Outputs	Infos	ID	Type	
Central unit							0	DI58	0
Refrigeration									0
Cold chamber 1	0.1	0.0			❄️❄️❄️		22245'29245	DC24D	0
Cold chamber 2	-0.6	1.0			❄️❄️❄️		22245'52158	DC24DE	0
Cold chamber 3	0.0	3.0			❄️❄️❄️		22245'12988	DI24-4	0
Cold chamber 4	0.5	0.0			❄️❄️❄️		22245'28403	DC24D	0
Compressor	2.1	2.7	🔴		🌀🌀🌀		22245'23476	DI24-E	0
HVAC									0
Lighting									0
Miscellaneous									0
PLC functions									0
CONTROLLER_OUTPUT (ENG)									0
digAlarmSet (FR)			🔴						0
Thermostat (ENG)									0
Timer cyclique (ENG)									0
Timers prédéfinits (ENG)									0
Ventilation salle des machines (ENG)									0
PLC function example (ENG)									0

Historic of alarms

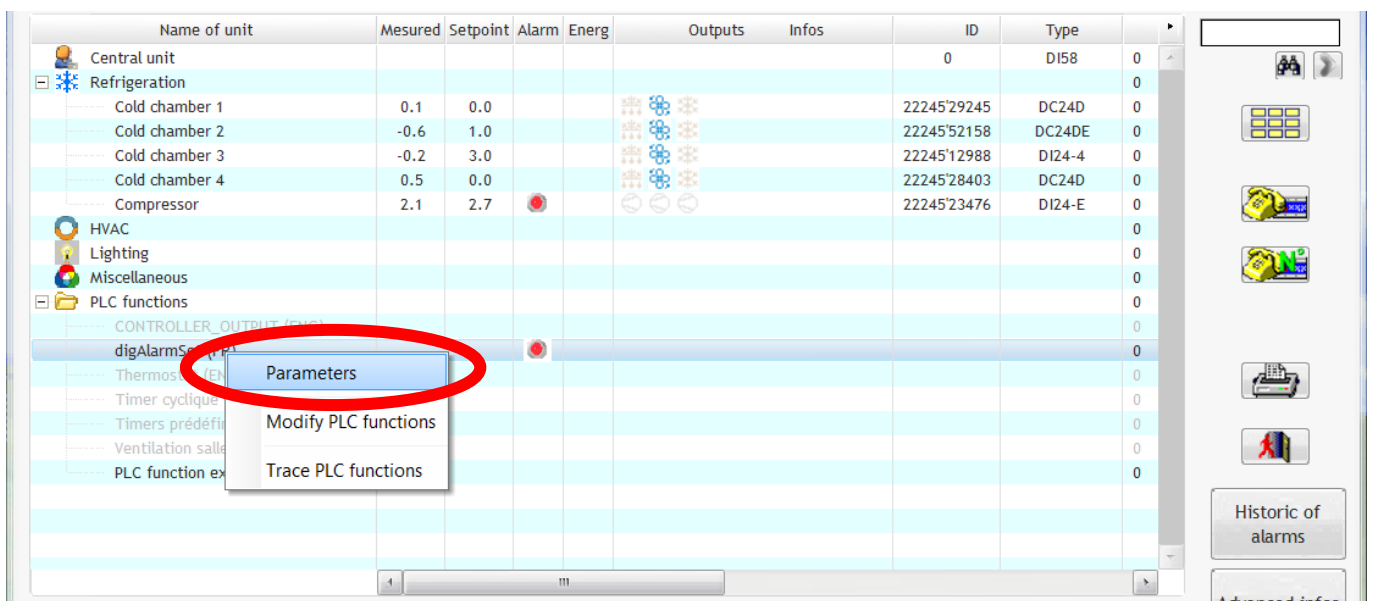
Advanced info

5.1. MODIFICATION OF PARAMETERS BY A USER

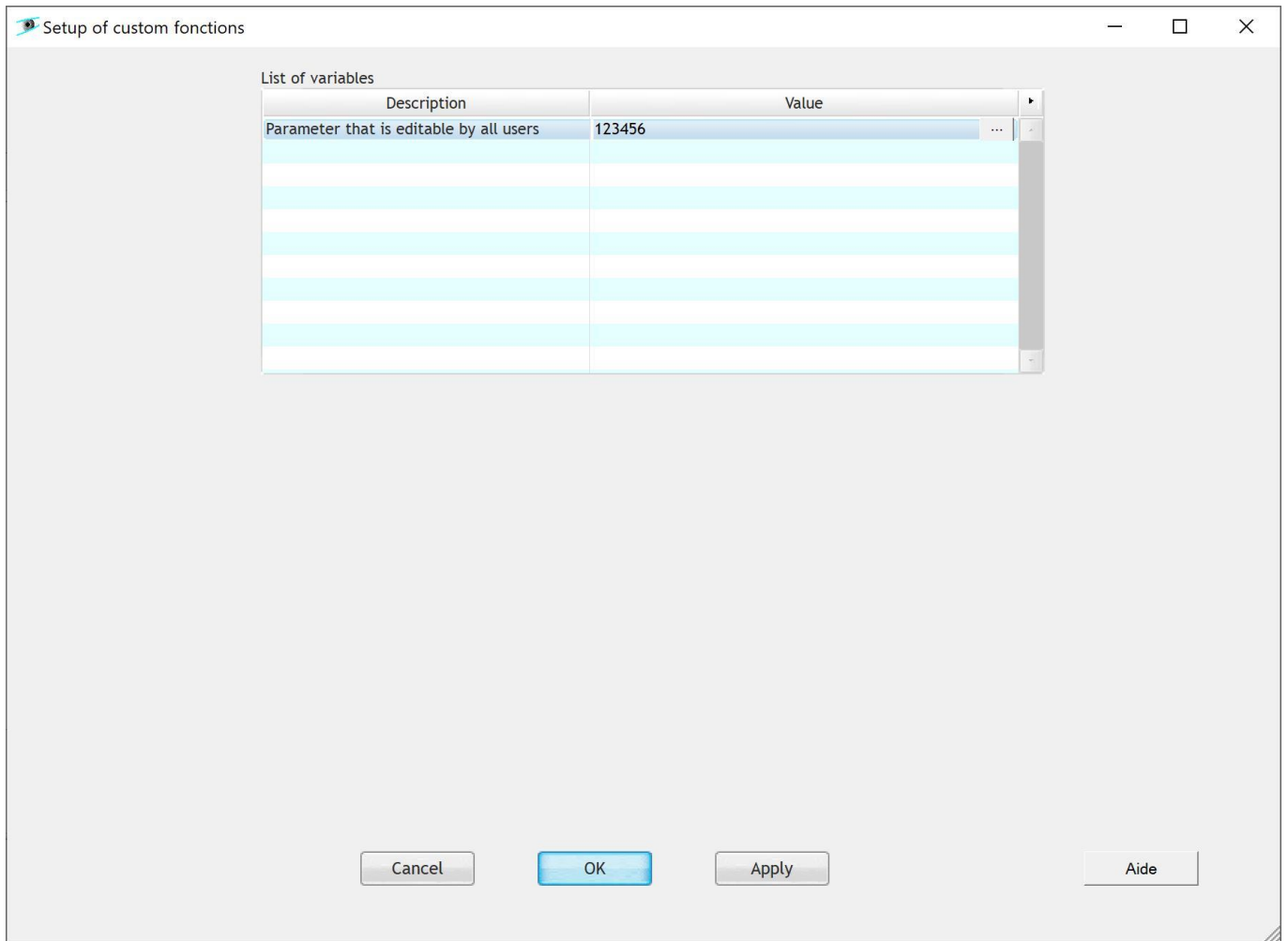
The variables declared with the `parameter` type can be modified by a user who does not have programming rights.



It is also possible to modify the parameters from the global view of the installation. Just click on the **Parameters** option of the context menu, or double-click on the function whose settings you want to change.



For users without programming rights, the **Programming mode** box is always unchecked. When they right-click to display the context menu, the only option they have access to is the **Parametres** option, which allows them to change the settings.



6. PLC FUNCTIONS PROGRAMMING LANGUAGE REFERENCE

The programming language we use is based on the Windev Mobile language developed by PC Soft. This manual describes only the basic operators, the most frequently used. They are sufficient to create simple functions. To create advanced functions, we recommend that you study the Windev Mobile manual, which contains a detailed description of all operators and functions of this language.

The names of the variables of the PLC functions are case sensitive, i. e. upper and lower case must be respected when using a variable.

Variable names should only contain letters, numbers and the underlined character (_). They cannot contain spaces.

6.1. STRUCTURES

6.1.1. Comments

Comments can be added to the code. They start with a double slash `//` and end at the end of the line. Comments are useful for documenting the code. They are not executed.

6.1.2. Variable assignation

Variables must be declared in the table **List of variables**. They must be surrounded by two braces `{{...}}` when used in the code, whether in reading or writing. The possible types of variables are:

I/O of a module: To assign the value of an input or output (I/O) to a variable in the PLC code.

Internal variable: To declare a new variable that will be defined and accessible in the PLC code.

Parameter: To assign a parameter value that can be modified by a user who does not have programmer rights.

Alarm : To generate an alarm.

Timer: To use the return value of a timer already created in the central unit (**Central unit settings / Advanced settings / Timers**).

6.1.3. Operators

CONCATENATION OF CHARACTER STRINGS

It is possible to concatenate strings between them and also to concatenate strings and variables. The result can then be used to display informative messages. The string concatenation operator is the `+` sign.

EXAMPLE

The code below will display in the **Trace** window the string `Temp = xx°C`, where `xx` is the temperature value, for example `Temp = -18°C`. The `Temp1` probe must be defined in the list of variables.

Constant character strings must be surrounded by double braces and quotation marks `{{"Temp = "}}`, while variables only need braces, but not quotation marks `{{Temp1}}`. The braces can optionally be followed and preceded by a space, but they cannot be separated by a space.

The function parameters are surrounded by parentheses that can optionally be preceded and followed by a space.

```
digTrace({"Temp = "} + {{Temp1}} + {"°C"})
```

ADDITION

The addition also uses the `+` sign, such as the concatenation of strings. If you want to use addition and concatenation in the same function, you must perform the operations on two separate lines.

```
{{Val}} = {{Temp1}} + 1.5 // The + sign performs an addition
digTrace ( {"Temp1 = "} + {{Temp1}} ) // The + sign performs a concatenation
digTrace ( {"Val = "} + {{Val}} ) // The + sign performs a concatenation
```

SUBTRACTION

```
{{Val}} = {{Temp1}} - 1.5
```

MULTIPLICATION

```
{{Val}} = {{Temp1}} * 1.5
```

DIVISION

```
{{Val}} = {{Temp1}} / 1.5
```

ABSOLUTE VALUE

```
{{Val}} = Abs ( {{Temp1}} )
```

COMPARISON OPERATORS

Comparison operators return Boolean values, i.e. True or False.

```
{{Val}} = {{Temp1}} > -20 // Greater
{{Val}} = {{Temp1}} >= -20 // Greater or equal

{{Val}} = {{Temp1}} < -20 // Smaller
{{Val}} = {{Temp1}} <= -20 // Smaller or equal

{{Val}} = {{Temp1}} = -20 // Absolutely equal
{{Val}} = {{Temp1}} <> -20 // Different
```

LOGICAL OPERATORS

```
{{Val}} = ( {{Temp1}} > -20 ) AND ( {{Temp2}} > -20 ) // Logical AND
{{Val}} = ( {{Temp1}} > -20 ) OR ( {{Temp2}} > -20 ) // Logical OR
{{Val}} = NOT ( {{Temp2}} > -20 ) // Logical NOT
```

LOGICAL VALUES

```
{{Val}} = True // All numerical values other than 0 are also evaluated as True.
{{Val}} = False // The numerical value 0 is also evaluated as False
```

MODULO

The modulo is the rest of the integer division. For example, 7 modulo 3 = 1, because $7 / 3 = 2$ remains 1, or in other words $3 \times 2 + 1 = 7$. The modulo can be used to easily create clocked signals. For example, if we want to generate a signal with a 10-minute period that is at 1 for 2 minutes and therefore at 0 for 8 minutes:

```
{{Val}} = modulo (minuteFrom2000, 10) < 2
```

6.1.4. IF..THEN..ELSE..

The **IF** conditional instruction allows you to choose to execute an action according to a condition.

EXAMPLE

```
IF {{Temp1}} > 0 THEN
    // Action 1
ELSE IF {{Temp1}} > -20 THEN
    // Action 2
ELSE
    // Action 3
END
```

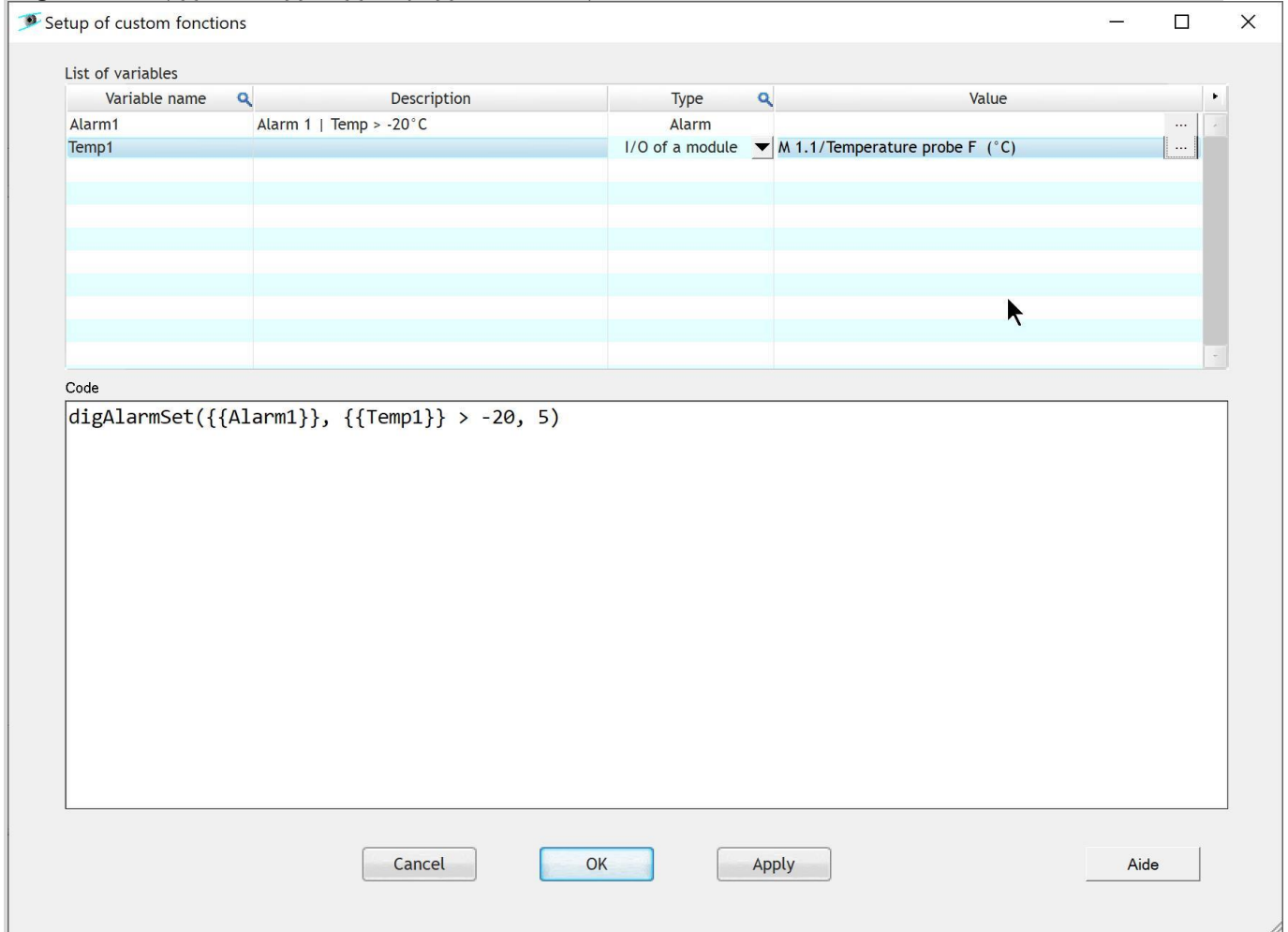
6.2. PREDEFINED FUNCTIONS

6.2.1. digAlarmSet

Triggers an alarm when a condition is true for a certain time. The alarm message must be configured in the **Description** field of the variable list.

EXAMPLE

```
digAlarmSet({{Alarm1}}, {{Temp1}} > -20, 5)
```



SYNTAX

```
digAlarmSet(name, condition, delay)
```

<name> : Character string

Alarm reference (Name of the variable in the table **List of variables**).

<condition> : Boolean

Condition that can take the value False (value equal to 0) or True (value other than 0). This condition is typically written as a test, for example: `{{Temp1}} > -20`.

<delay> : Integer or real

Delay in minutes before the alarm is triggered.

6.2.2. digAlarmGetState

Returns the current alarm status of a given alarm, 0 = no alarm, 1 = alarm in progress.

EXAMPLE

```
IF digAlarmGetState({{Alarme1}}) THEN
  // Action to execute
END
```

SYNTAX

```
digAlarmGetState(name)
```

<name> : Character string

Alarm reference (Name of the variable in the table [List of variables](#)).

6.2.3. digMessageSend

Send a message by email or SMS.

Attention! The "E-mail messaging" settings in the central unit must be configured correctly for sending emails to work. Similarly, the "SMS messaging" settings must be configured and the GSM modem must be connected for SMS sending to work.

EXAMPLE

```
{{Message1}} = {{ "Message to send" }}
digMessageSend({"test@example.com"}, {{Message1}}, {"EMail"})
digMessageSend({"+411234567"}, {{Message1}}, {"SMS"})
```

SYNTAX

```
digMessageSend(destination, message, type)
```

<destination> : Character string

E-mail address of the recipient when `type = "EMail"` or telephone number when `type = "SMS"`.

<message> : Character string

Message to send

<type>

Message type. `type = "EMail"` or `"SMS"`

6.2.4. digTrace

Send a character string to the Trace window.

SYNTAXE

```
digTrace(message)
```

< message > : Character string

The message to be displayed in the **Trace** window.

EXAMPLE

```
digTrace({"Temperature = "} + {{TempRead}} + {" °C. Heater = "} + {{HeaterOn}})
```

6.2.1. digSetpointShift

Can be used with controller firmware versions equal to or higher than 21011 and with DC24D, DC24DE, DC24E, DC24EE type controllers operating in modes 0, 1 or 2.

Shifts the setpoint with respect to the setpoint programmed in the controller parameters. The setpoint will be shifted for 10 minutes since the last `digSetpointShift()` command. After this time the setpoint programmed in the controller parameters will be automatically restored. The original setpoint can be restored immediately by sending the same command with the constant `CONTROLLER_SETPOINT` as parameter `rShift`.

SYNTAXE

```
digSetpointShift(unitID, rShift)
```

<unitID> : Integer or real

Controller identifier visible in the `unitID` column of the "System configuration" table.

<rShift> : Integer or real

Offset (positive or negative) from the setpoint programmed in the controller.

EXEMPLE

```
digSetpointShift(7, 5)
```

See chapter 6.4.1 for an additional example.

6.2.2. digSetpointSetTR

Can be used with controller firmware versions 21011 or higher and with software versions DC58 21011 or higher. Can only be used with DC24TR type controllers operating in modes 0 or 1.

Changes the controller setpoint to the value passed in the `rNewSetpoint` parameter. The setpoint will be changed for 10 minutes since the last `digSetpointSetTR()` command. After this time the setpoint programmed in the controller parameters will be restored automatically. The original setpoint can be restored immediately by sending the same command with the constant `CONTROLLER_SETPOINT` as parameter `rNewSetpoint`.

SYNTAXE

```
digSetpointSetTR(unitID, rNewSetpoint)
```

`<unitID>` : Integer or real

Controller identifier visible in the `unitID` column of the "System configuration" table.

`<rNewSetpoint>` : Integer or real

New controller setpoint (gascooler setpoint in mode 0 or HP setpoint in mode 1).

EXAMPLE

```
IF {{TempWarmWater}} > {{ValueLimit}} THEN
    digSetpointSetTR(7, 100)
ELSE
    digSetpointSetTR(7, CONTROLLER_SETPOINT)
END
```

6.2.3. digSetpointSetTR_MP

Can be used with controller firmware versions 21011 or higher and with software versions DC58 21011 or higher. Can only be used with DC24TR type controllers operating in mode 1.

Changes the medium pressure setpoint (MP) of the controller to the value passed in the `rNewSetpoint` parameter. The setpoint will be changed for 10 minutes since the last `digSetpointSetTR_MP()` command. After this time the setpoint programmed in the controller parameters will be automatically restored. The original setpoint can be restored immediately by sending the same command with the constant `CONTROLLER_SETPOINT` as parameter `rNewSetpoint`.

SYNTAXE

```
digSetpointSetTR_MP(unitID, rNewSetpoint)
```

`<unitID>` : Integer or real

Controller identifier visible in the `unitID` column of the "System configuration" table.

`<rNewSetpoint>` : Integer or real

New controller setpoint (MP setpoint in mode 1).

EXAMPLE

```
digSetpointSetTR_MP(7, 37)
```

6.3. SYSTEM VARIABLES THAT CAN BE USED IN PLC FUNCTIONS

minuteFrom2000

minutes from 1.01.2000 00:00

secondFrom2000

secondes from 1.01.2000 00:00

hourFrom2000

heures from 1.01.2000 00:00

minuteOfDay

minutes from midnight (00:00:00)

secondOfDay

seconds from midnight (00:00:00)

hourOfDay

heures from midnight (00:00:00)

6.4.1. CONTROLLER_OUTPUT

The `CONTROLLER_OUTPUT` system constant allows you to return the hand to a module after having forced the output to 1 or 0 for digital outputs or between 0 and 100% for analog outputs.

Explained in another way, 3 different values can be assigned to an output of a module:

- `1`: The output is forced to 1.
- `0`: The output is forced to 0.
- `CONTROLLER_OUTPUT`: The output is controlled by the module and no longer by the PLC function.

EXAMPLE

```
IF hourOfDay < 12 THEN
    {{OutputRL1}} = 1                // The exit is forced to 1
                                    // between midnight and noon.
ELSE IF hourOfDay < 13 THEN
    {{OutputRL1}} = 0                // The exit is forced to 0
                                    // between noon and 1:00 p. m.
ELSE
    {{OutputRL1}} = CONTROLLER_OUTPUT // The output is controlled by the module
END                                  // and not by the PLC function
                                    // between 13:00 and midnight.
```

Setup of custom functions

Variable name	Description	Type	Value
OutputRL1	CONTROLLER_OUTPUT Demo	I/O of a module	P 6.0/Contact C1

Code

```
IF hourOfDay < 12 THEN
    {{OutputRL1}} = 1                // The exit is forced to 1
                                    // between midnight and noon.
ELSE IF hourOfDay < 13 THEN
    {{OutputRL1}} = 0                // The exit is forced to 0
                                    // between noon and 1:00 p. m.
ELSE
    {{OutputRL1}} = CONTROLLER_OUTPUT // The output is controlled by the module
END                                  // and not by the PLC function
                                    // between 13:00 and midnight.
```

6.4.1. CONTROLLER_SETPOINT

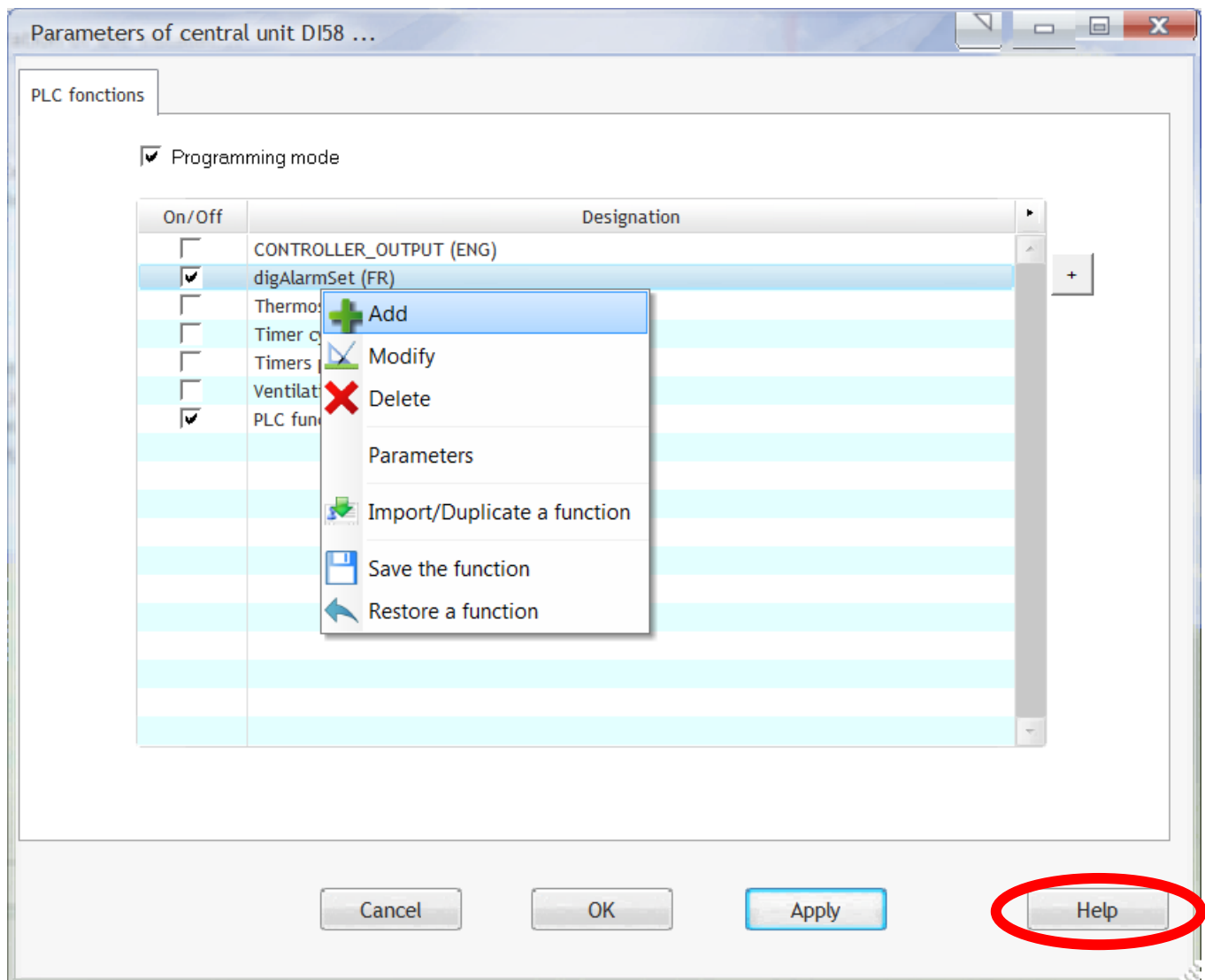
La constante système `CONTROLLER_SETPOINT` permet de rendre la main à un module après avoir forcé le décalage de la consigne (`digSetpointShift()`) ou une nouvelle consigne (`digSetpointSetTR()` ou `digSetpointSetTR_MP()`).

EXAMPLE

```
IF hourOfDay > 18 OR hourOfDay < 8 THEN
    digSetpointShift(10, 5)    // The set point is shifted by +5° between 6pm and 8am.
ELSE
    digSetpointShift(10, CONTROLLER_SETPOINT)// The output is controlled by the module.
END
```

7. HELP BUTTON

The help button opens the documentation of the PLC function.

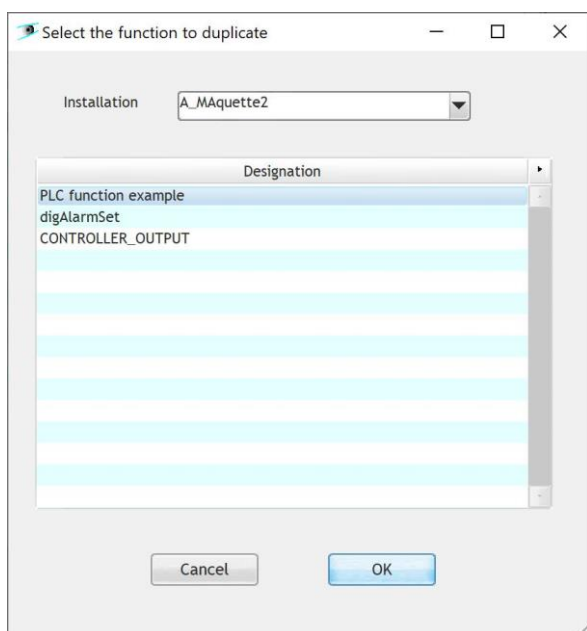
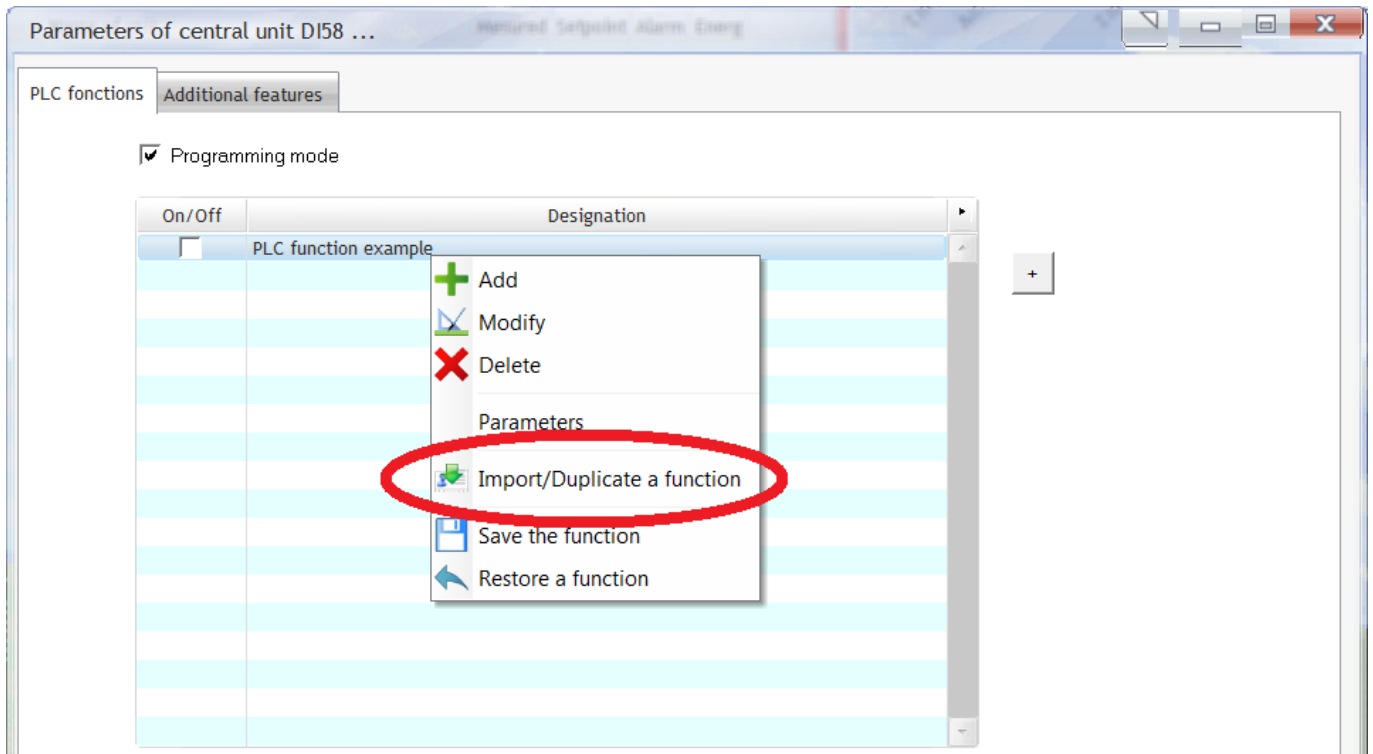


8. IMPORTING AND DUPLICATING FUNCTIONS

It is possible to import and duplicate functions from another installation. To do this, open the context menu by right-clicking on the mouse and choose **Import/Duplicate an existing function**. In the window that appears, you can select the installation on which the function is located. If this installation is different from the one you are connected to, the function will be imported. If, on the other hand, it is the same installation, then the function will be duplicated.

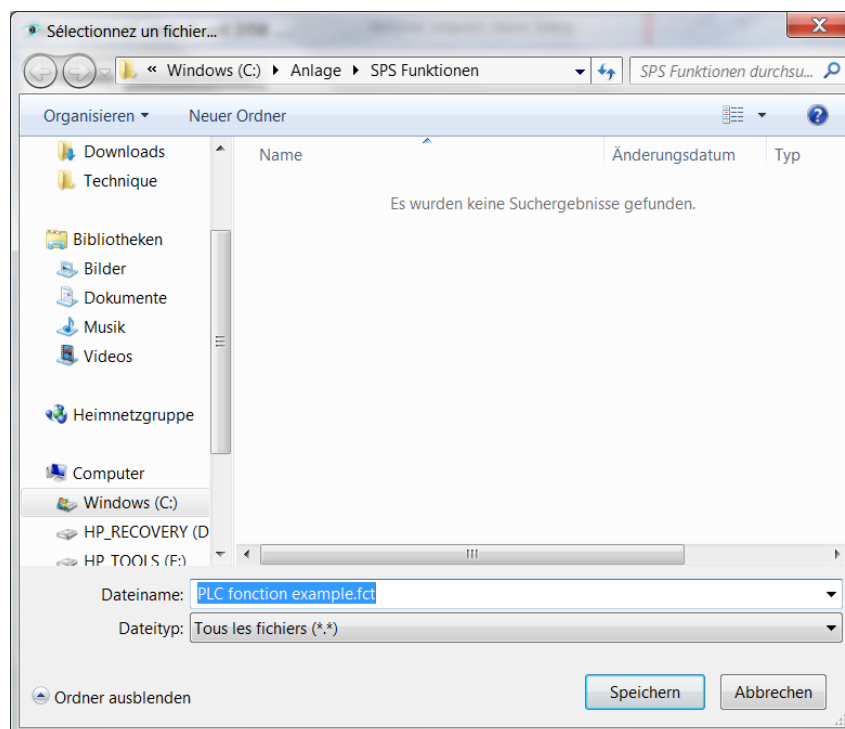
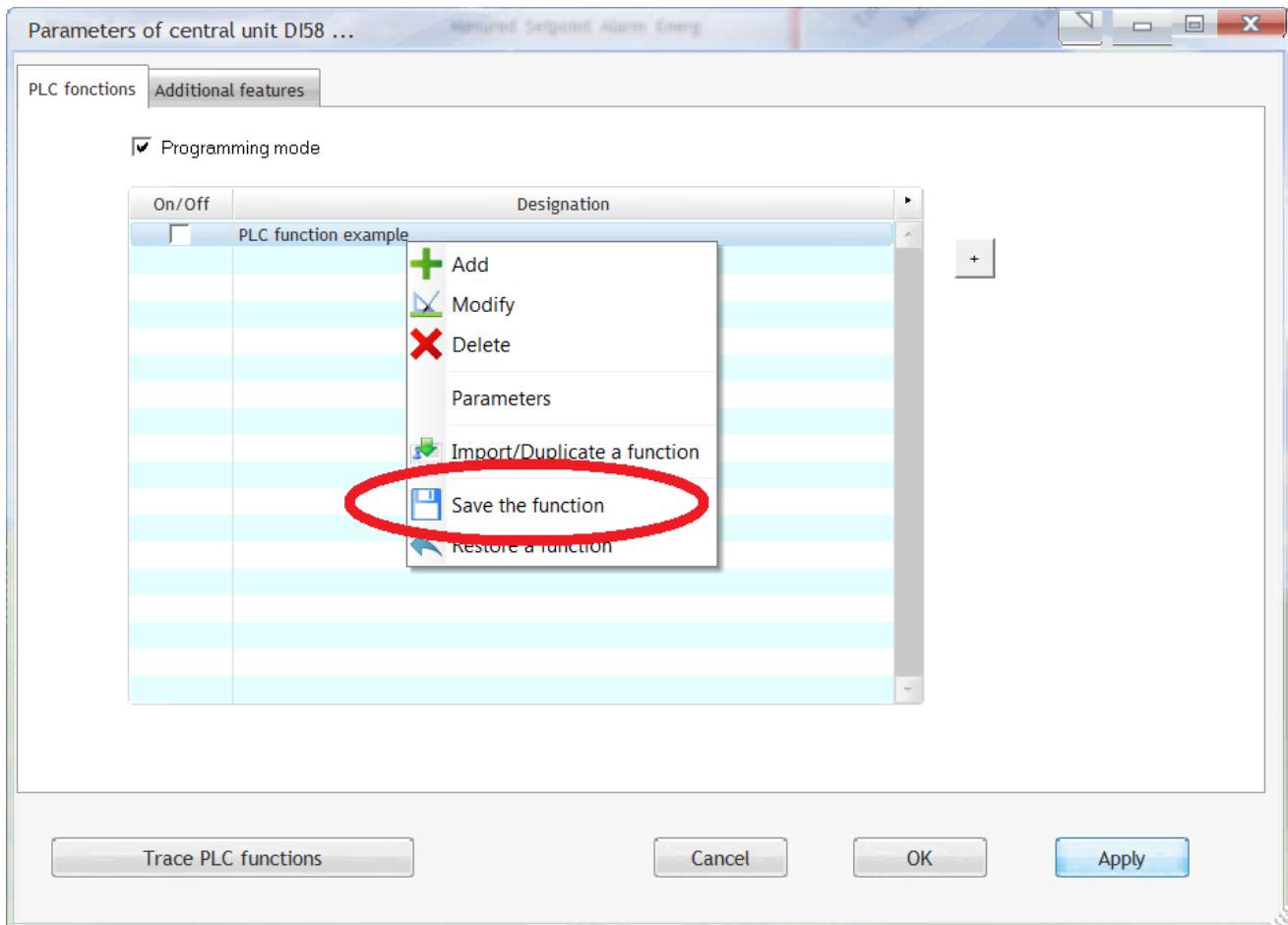
The links of variables of the type "I/O of a module" to the inputs and outputs of the modules ("Value" column) will not be copied because the new function will generally use other inputs and outputs. These links must be selected manually by clicking on the "..." button in the "Value" column.

It is possible to rename the functions by double-clicking on their name.



9. SAVE AND RESTORE FUNCTIONS

It is possible to save functions on your computer. Open the context menu by right-clicking on the function you want to save and choose **Save the function**. In the window that appears, you can select the folder and name under which you want to save the function.



10. EXAMPLES

10.1. MACHINE ROOM VENTILATION

```
// Machine room ventilation
IF {{TMachineRoom}} > ( {{Setpoint}} + {{Delta}} ) AND {{TOutside}} < {{TMachineRoom}}
THEN
    {{Ventilation}} = 1
ELSE
    IF {{TMachineRoom}} < {{Setpoint}} OR {{TOutside}} > {{TMachineRoom}} THEN
        {{Ventilation}} = 0
    END
END
digTrace({"Ventilation = "} + {{Ventilation}})
```

The screenshot shows a software window titled "Setup of custom fonctions". It contains a table of variables and a code editor.

Variable name	Description	Type	Value
TMachineRoom	Machine room temperature	I/O of a module	M 1.1/Temperature probe F (°C)
TOutside	Outside temperature	I/O of a module	M 4.2/Probe E (°C)
Setpoint	Setpoint	Parameter	25.0
Delta	Delta	Parameter	1.0
Ventilation	Ventilation control	I/O of a module	M 1.2/Output contact RL3

Code

```
// Machine room ventilation
IF {{TMachineRoom}} > ( {{Setpoint}} + {{Delta}} ) AND {{TOutside}} < {{TMachineRoom}} THEN
    {{Ventilation}} = 1
ELSE
    IF {{TMachineRoom}} < {{Setpoint}} OR {{TOutside}} >= {{TMachineRoom}} THEN
        {{Ventilation}} = 0
    END
END
digTrace({"Ventilation = "} + {{Ventilation}})
```

Buttons: Cancel, OK, Aide, ?

10.2. THERMOSTAT

```
// Thermostat
IF {{Temp}} > ( {{Setpoint}} + {{Delta}} ) THEN
    {{Heater}} = 0
ELSE
    IF ( {{Temp}} < {{Setpoint}} ) THEN
        {{Heater}} = 1
    END
END
digTrace({"Temperature = "} + {{Temp}} + {"°C"})
digTrace({"Heater = "} + {{Heater}})
```

Setup of custom fonctions

List of variables

Variable name	Description	Type	Value
Temp	Room temperature	I/O of a module	M 2.0/Ambient temperature (probe A) (°C)
Heater		I/O of a module	P 5.1/Output contact RL1
Setpoint		Parameter	20
Delta		Parameter	1

Code

```
// Thermostat
IF {{Temp}} > ( {{Setpoint}} + {{Delta}} ) THEN
    {{Heater}} = 0
ELSE
    IF ( {{Temp}} < {{Setpoint}} ) THEN
        {{Heater}} = 1
    END
END
digTrace({"Temperature = "} + {{Temp}} + {"°C"})
digTrace({"Heater = "} + {{Heater}})
```

Buttons: Cancel, OK, Apply, Aide

10.3. CYCLIC TIMER

```
// Activates the output for 2 minutes,  
// then turn it off for 8 minutes  
// for a total cycle duration of 10 minutes.
```

```
IF modulo (minuteFrom2000, 10) < 2 THEN  
    {{Output}} = 1  
ELSE  
    {{Output}} = 0  
END  
digTrace({"Output = "} + {{ Output }})
```

Setup of custom fonctions

List of variables

Variable name	Description	Type	Value
Output	Heater	I/O of a module	P 5.1/Output contact RL3

Code

```
// Activates the output for 2 minutes,  
// then triggers it for 8 minutes  
// for a total cycle duration of 10 minutes.  
  
IF modulo (minuteFrom2000, 10) < 2 THEN  
    {{Output}} = 1  
ELSE  
    {{Output}} = 0  
END  
digTrace({"Output = "} + {{ Output }})
```

Cancel OK Apply Aide

10.4. USING THE TIMERS PREDEFINED IN THE CENTRAL UNIT

```
// For timer configuration, see the manual  
// "Newel 3 - Complete - FR.pdf"  
// chapter 10.12.13
```

```
IF {{Timer_1}} THEN  
    digTrace({"Timer = 1"})  
ELSE  
    digTrace({"Timer = 0"})  
END
```

Setup of custom fonctions

List of variables

Variable name	Description	Type	Value
Timer_1	Daytime timer	Timer	Fonctionnement du Jour

Code

```
// For timer configuration, see the manual  
// "Newel 3 - Complete - FR.pdf"  
// chapter 10.12.13  
  
IF {{Timer_1}} THEN  
    digTrace({"Timer = 1"})  
ELSE  
    digTrace({"Timer = 0"})  
END
```

Cancel OK Apply Aide