Via monte Nero, 40/B – 21049 TRADATE (VA) ITALY Phone: +39 (0)331841070 - e-mail:datexel@datexel.it - www.datexel.it

*User Guide **Dev9K** v. 2.1*

Integrated Development Environment to design and debug the applications based on the DAT9000 series controllers.
NOTE: this document is suitable for devices with these firmware versions:
9BB0, 9BB4, 9BB5, 9BA8, 9560, 9570, 9BD1, 9BD2

INDEX

1.1 - GENERAL DESCRIPTION

Dev9K is an Integrated Development Environment running under the Windows® Operative System that allows to design and debug the applications based on the DAT9000 series control devices. By Dev9K it is possible to make the DAT9000 series controllers to execute I/O read and write operations (DAT3000, DAT8000, DAT10000 series), mathematical and logic operations and timers. Moreover it is possible to read and write in real time the Internal Registers of the Controller or connect it directly to the slave devices connected to its Modbus Master Port.

1.2 – MINIMUM SYSTEM REQUIREMENTS

Operative System	Windows® 7 / 8 / 8.1 / 10 / 11
Available Hard Disk memory	50 MB

1.3 - PROCEDURE OF INSTALLATION

Download the installation file from the website www.datexel.it
Close eventual active or background applications.
Run the installation file.
Wait for the Windows Firewall's notice of security. Click OK when the system asks for security permission.
Follow the Installation Wizard.

1.4 - TERMINOLOGY

Description of terms and abbreviations used in the IDE and inside this manual:

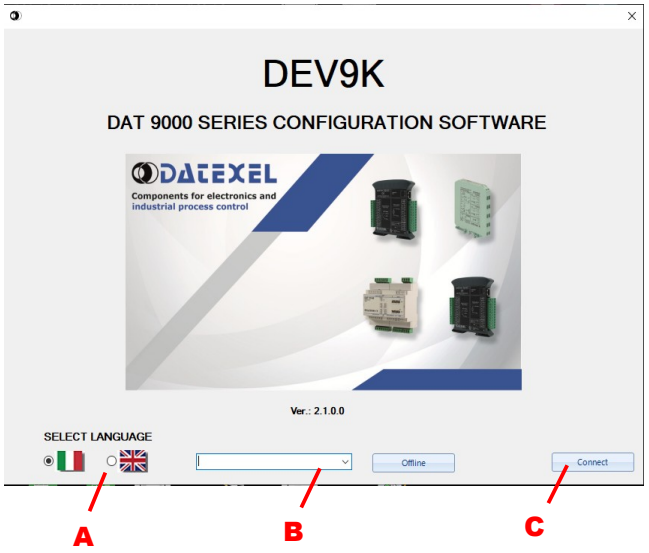
Controller	**Device of the DAT9000 series.**
Integrated Development Environment IDE	Instructions set to create, debug and test the Program.
Program	List of functions executed by the Controller.
Function Block F.B.	Each block constituting the flow chart of Program.
Function	Logical, mathematical or flow operation executed by the Controller.
Argument	Each parameter contained in a function.
Register	Position of a variable in the volatile memory of Controller accessible from the Register Table.
Retentive Register	Position of a variable in the non-volatile memory of Controller accessible from the Register Table.

2.1 – OPEN AND INITIALIZE THE PROGRAM

Once the software has been installed, double click on the program icon **(Pict.2.1)** and give the administrator permissions to run it .
The opening window of the program will appear **(Pict.2.2)**
It is possible to:
- set the language **(Pict.2.2-A);**
- enter in offline modality choosing a device from the drop down list and clicking on button "*Offline*" **(Pict.2.2-B)**;
- enter in the search window to connect directly to a device **(Fig.2.2-C)**

Pict. 2.1

Pict. 2.2



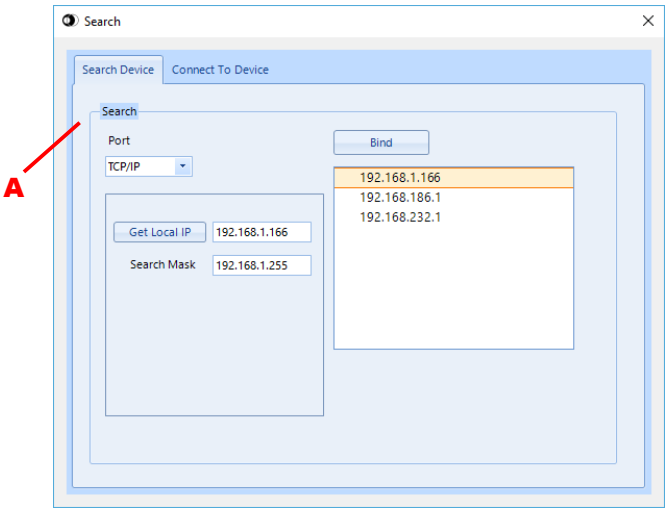
2.2 – CONNECT AND SEARCH FOR THE DEVICE

In the Search window there are two modes to connect to device:

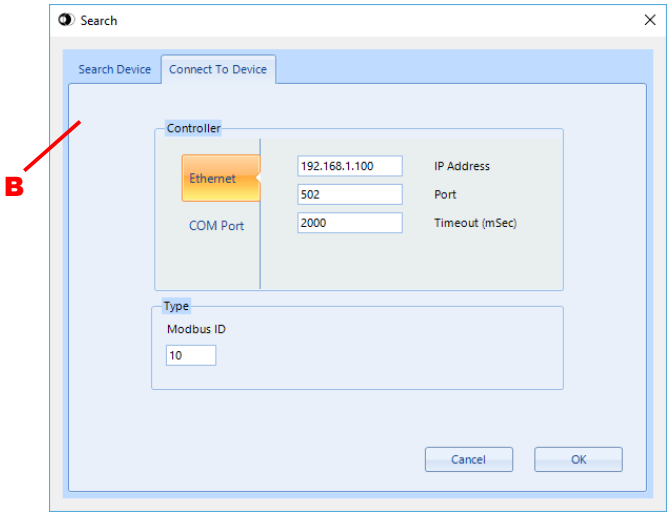
- Search a device over the Ethernet network and connect to it **(Pict.2.3-A)**

- Connect directly to the device by its parameters via Ethernet or via COM port (RS485/uUSB), inserting them in the fields and clicking the button "*OK*" **(Pict.2.3-B)**

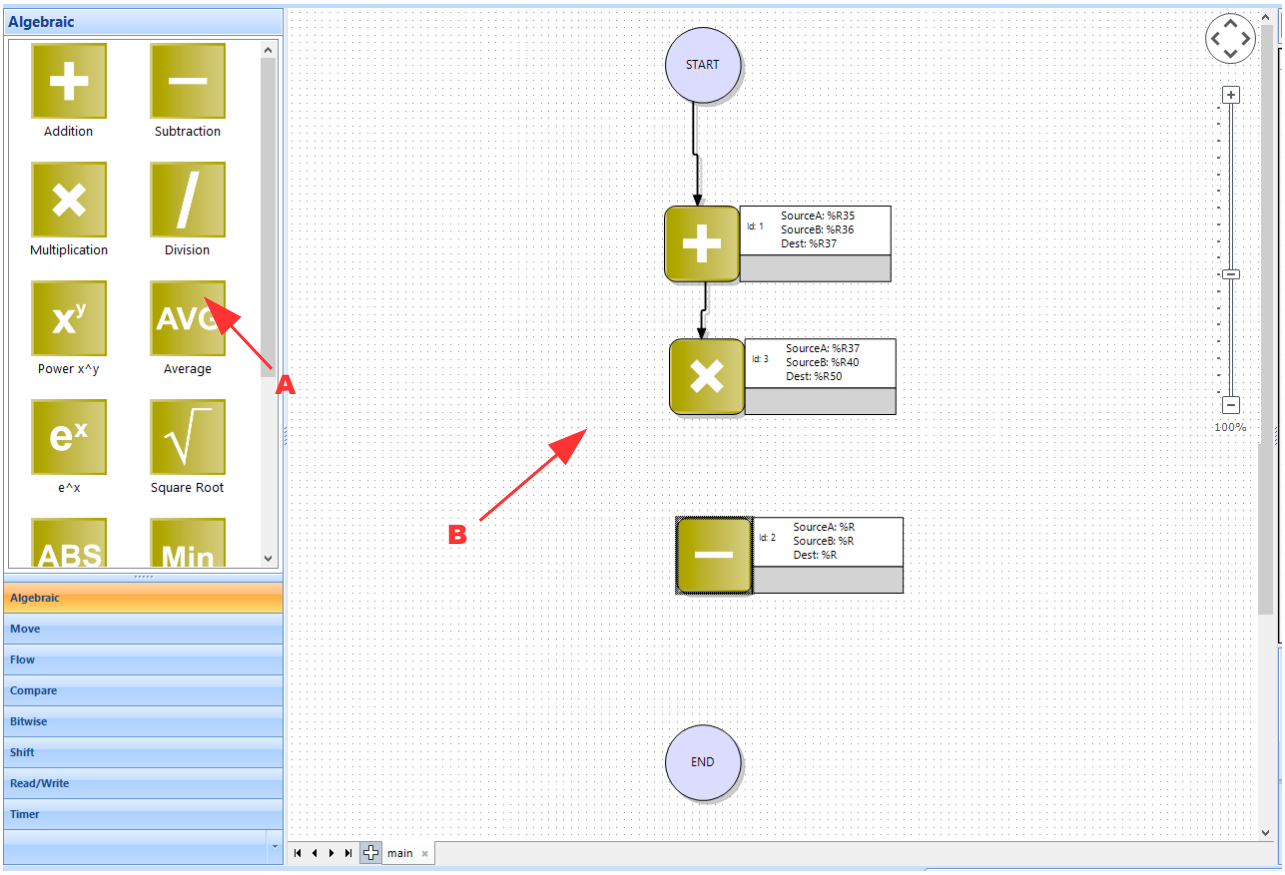
For details refer to the paragraph 6.1 and 6.2.



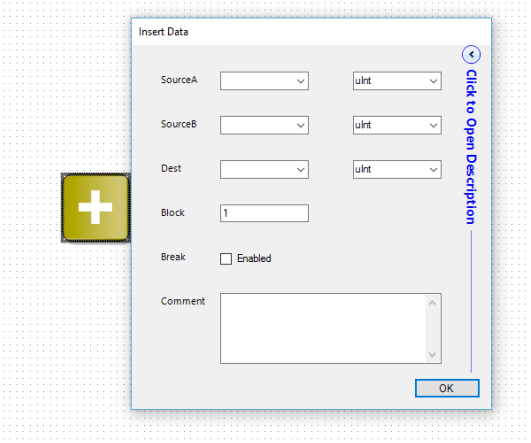
Pict. 2.3



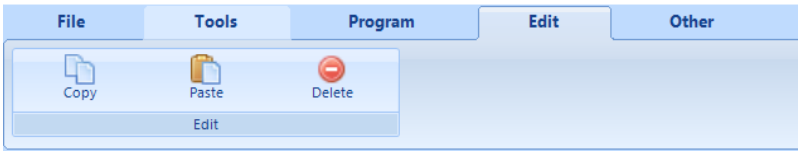
2.3 – CREATION OF A DIAGRAM



Pict. 2.4



Pict. 2.6



Pict. 2.5

- Insert a function block

It is possible to insert a new function into the diagram dragging it from function block list **(Pict.2.4-A)** to the working area of diagram **(Pict.2.4-B).** When a block is dropped down in the diagram, the software opens a window for the insertion of data that the function chosen requires **(Pict.2.6)**
The function blocks are grouped and listed into functional groups.

- Move the function blocks in the working area

To move one or more function block, select them (left click of the mouse) and then drag them in the desired position.

- Creation of the links between a block and another

The flow of the program is defined by the links created between the function blocks
To create a link between a block and another, the block from which the link starts from must be deselected. So click the left mouse button on the block and drag the link to the destination block.

- Modify the data of a function block

To modify data in a function block, double-click of the left mouse button on the desired function block. As result, the data entry window will open **(Pict.2.6).**

- Delete the function blocks

To delete one or more function blocks select the desired blocks and use one of following methods:
- press the button *CANC* of the keyboard
- click the right button of the mouse and then select *Delete*
- in the menu bar, in the section *Edit,* select the button *Delete* **(Pict.2.5)**

- Copy the function blocks

To copy one or more function blocks select the desired blocks and use one of following methods:
- press the buttons *CTRL+C* of the keyboard
- click the right button of the mouse and then select *Copy*
- in the menu bar, in the section *Edit,* select the button *Copy* **(Pict.2.5)**

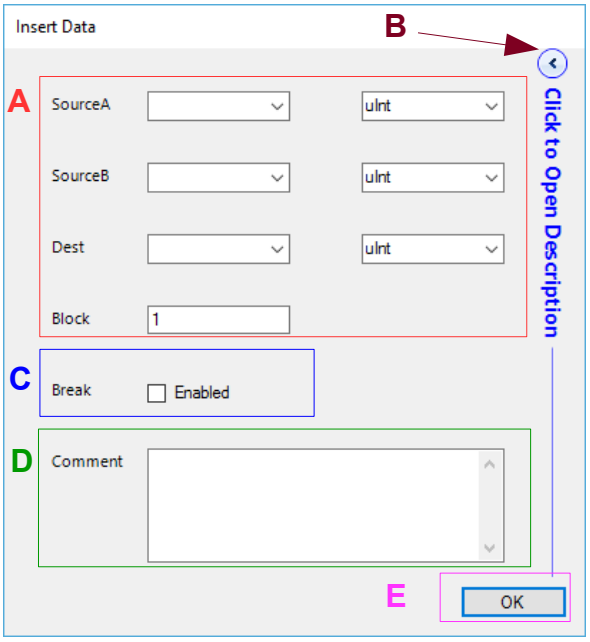
- Paste the function blocks

To paste one or more function blocks be sure to have copied them before and use one of following methods:
- press the buttons *CTRL+V* of the keyboard
- click the right button of the mouse and then select *Paste*
- in the menu bar, in the section *Edit,* select the button *Paste* **(Pict.2.5)**

2.4 INSERT DATA IN A FUNCTION

When the user modifies a Function Block or inserts another one, the data entry window will open **(Pict.2.7).** It allows to set the arguments required by the function selected.

Usually the arguments in the top part **(Pict.2.7-A)** of the window identify and set data of the function (number of register, type of register, values, etc.). The structure is the same for each function: on the left there are the labels that describe what is the specific argument, in the centre there are the values for the arguments and on the right a combo box with the drop down list that allows to choose the type of data. Some functions may have some additional buttons that allow to edit more specific data.
In general:
Source: is the input data of the function. This can be a register or a constant. When the source is selected as constant(*K_Flt*), the insertion field becomes green **(Pict.2.8). The decimal point to use depends on the setting of the language in Windows.**
Dest: is the register where the result of the function is written
Block: is the number of repetition of the operation in consecutive registers (Source and Dest)
Mask: is the desired mask to apply to the relative *Source* or *Dest*



Pict. 2.7

In the top right corner of the window there is a pop-up menu **(Pict.2.7-B)** that describes the arguments of the function.

All of the functions, in addition to the first arguments, have also:

Break **(Pict.2.7-C):** this flag allows to insert a breakpoint into the function. When the device connected is in Debug mode, to create a breakpoint set this flag, close the function and download the program. If the device is in debug and receives the command *Run to Break* the flow of program stops on the function with break enabled and highlights it up.

Comment **(Pict.2.7-D):** this argument allows to insert a comment into the function block. By this the user can describe the role of function block in the diagram.

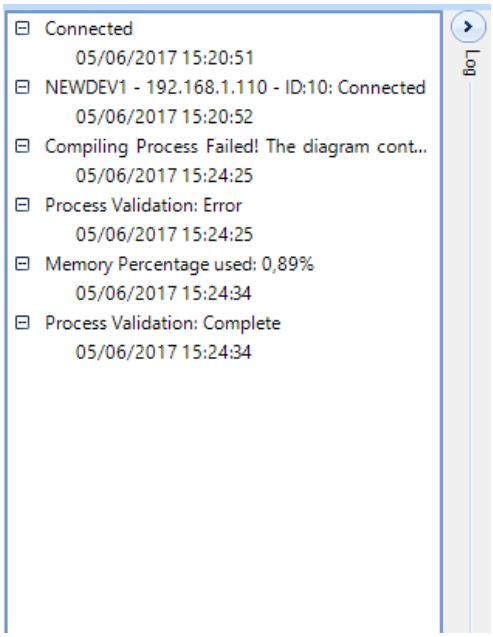
Pict. 2.8

After the insertion of all data required for the selected function click the button OK **(Pict.2.7-E).** If a field is empty, it will appear the message "*Found an empty field, continue?*"

Refer to the section "Function Description" for how to use the function blocks.

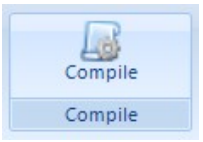
2.5 – LOG PANEL AND COMPILE

The Log Panel contains the information of connection and validation of the program **(Pict.2.9).**
After the insertion of function blocks, it is possible to compile the program clicking the button "*Compile* " in the menu bar **(Pict.2.10)**. The result of the validation will be appended to the log panel.
If there is an error in the validation process it will be displayed in the panel with the phrase "*Process Validation: Error*". If the validation ends correctly, in addition to the result ("*Process Validation: Complete"*), it will be displayed the percentage of Eeprom used by the program.



Pict. 2.9

NOTE:
Each function block takes a different quantity of Memory program. It is possible to insert function blocks until all of the available Eeprom Memory Space has been used (100 %).



Pict. 2.10

3.1 – REGISTER TABLE

The Internal Memory of the Controller is composed of 16 bit registers that are listed in the Register Table **(Pict.3.1)**

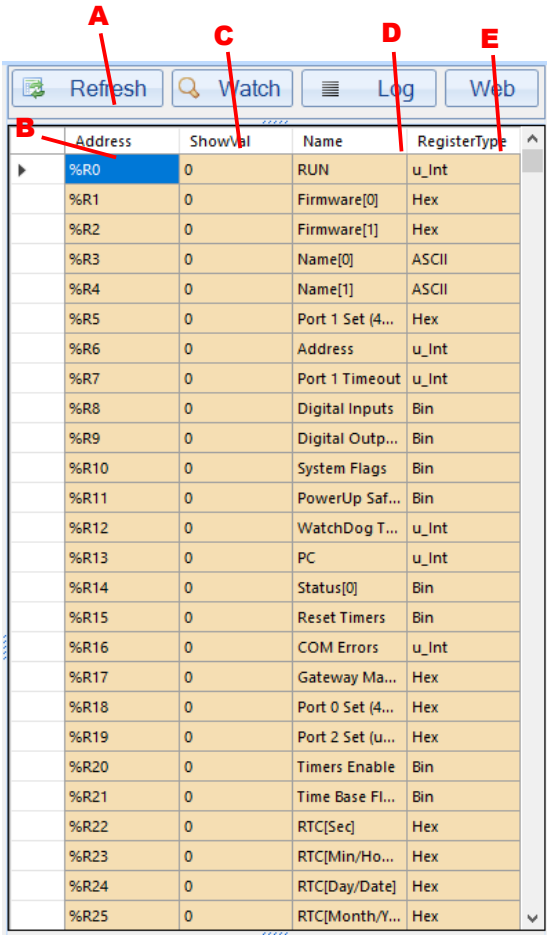
The Register Table is composed of:

System Registers: these registers contain the information about the status of the Controller. They change in function of the device connected and, available, allow to read the digital and analogue inputs and to set the digital and analogue outputs of the device. They are identified by orange rows.

General Purpose Registers: these registers can be used in the Program to move the data or to execute calculation functions. When the device is powered off the data in these register will be lost. They are identified by white rows.

Battery-Backed, General-Purpose Registers with Unlimited Writes: these registers can be used in the Program as source or destination of data for the function blocks that have to be saved when the controller is switched off such as value of counters.
They are identified by green rows.

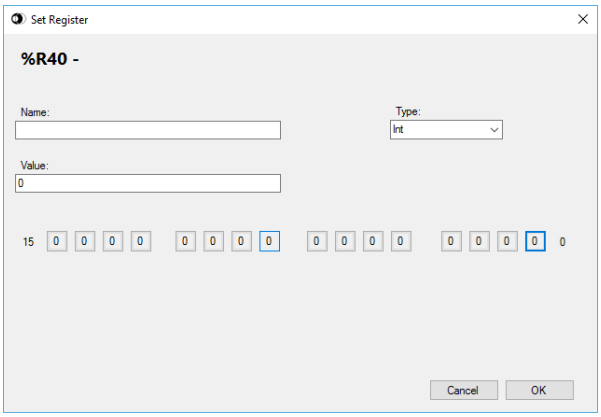
Retentive Registers: these registers can be used in the Program to contain fixed source of data for the function blocks such as constants. Unlike the General Purpose Registers, the values of Retentive Registers are saved in EEprom each time their value is written and they are uploaded when the Controller is powered-on.
Avoid to use these registers as destination of the function blocks because their cyclical writing may involve the permanent damaging of the EEprom of the device
They are identified by blue rows.



Pict. 3.1

To update the register's value, click the button “Refresh” **(Pict.3.1-A)**.
When a register is updated the font color changes to red.
Data visualized for each register:
Address: The register address **(Pict.3.1-B)**
ShowVal: The value contained into the Register **(Pict.3.1-C)**
Name: The register name **(Pict.3.1-D)**
Register Type: The register data format **(Pict.3.1-E)**

To modify the name or the value of a Register, double click on its row in the Register Table.
Inside the “*Set Register*” window **(Pict.3.2)** it is possible to set the name of the Register (only for the *General Purpose Register* or *Retentive Registers*), to modify the value contained in the Register (only if the Controller is connected) and to set the type of data format of the Register.



Pict. 3.2

3.2 - DATA FORMAT

Each *General Purpose or Retentive Register* can be read or written using one of the following data format :

u_Int	16 bit Unsigned Integer (0 ÷ 65535)
Int	16 bit Signed Integer (-32768 ÷ +32767)
u_Long	32 bit Unsigned Long (0 ÷ 4,294,967,295)
Long_	32 bit Signed Long (-2,147,483,648 ÷ +2,147,483,647)
Float	32 bit Floating Point
Hex	16 bit Unsigned Integer visualized as Hexadecimal characters (0000 ÷ FFFF)
ASCII	16 bit Unsigned Integer visualized as ASCII characters
Bin	16 bit Unsigned Integer visualized as binary code

NOTE: the 32 bit registers require the position of 2 registers and the second is identified as “*used*”
For the registers in 32 bit Floating point the format is Little Endian Bytes Swapped
For the registers in ASCII format, when more than 16 bit are occupied (more than a register that is more than 2 characters), starting from the second register the registers are identified as “*String_Ram*”.

4.1 – LIST OF FUNCTIONS

Functions selection tree, divided per functional group

4.2 – DESCRIPTION OF FUNCTION BLOCKS

Addition	**Calculates the sum of two values**
	Calculates the sum between an Internal Register and a constant or between two Internal Registers.

Arguments:

SourceA	Constant or Internal Register relative to the first operator
SourceB	Constant or Internal Register relative to the second operator
Dest	Internal Register wherein the result is written
Block	Number of repetition of the operation in consecutive registers (Source and Dest)

Subtraction	**Calculates the difference of two values**
	Calculates the difference between an Internal Register and a constant or between two Internal Registers.

Arguments:

SourceA	Constant or Internal Register relative to the first operator.
SourceB	Constant or Internal Register relative to the second operator.
Dest	Internal Register wherein the result is written.
Block	Number of repetition of the operation in consecutive registers (Source and Dest)

Multiplication	**Calculates the multiplication of two values**
	Calculates the multiplication between an Internal Register and a constant or between two Internal Registers

Arguments:

SourceA	Constant or Internal Register relative to the first operator.
SourceB	Constant or Internal Register relative to the second operator.
Dest	Internal Register wherein the result is written.
Block	Number of repetition of the operation in consecutive registers (Source and Dest)

Division	**Calculates the division between two values.**
	Calculates the division between an Internal Register and a constant or between two Internal Registers.

Arguments:

SourceA	Constant or Internal Register relative to the first operator.
SourceB	Constant or Internal Register relative to the second operator.
Dest	Internal Register wherein the result is written.
Block	Number of repetition of the operation in consecutive registers (Source and Dest)

Power x^y	**Performs exponentiation between two values.**
	Performs exponentiation between a register and a constant or between two registers.

Arguments:

SourceX	Constant or Internal Register relative to the base of the exponentiation
SourceY	Constant or Internal Register relative to the exponent of the exponentiation
Dest	Internal Register wherein the result is written
Block	Number of repetition of the operation in consecutive registers (Source and Dest)

Average	**Calculates the Arithmetic mean of N values.**
	Calculates the Arithmetic mean of N Internal Registers values starting from Source (sum of the values / N).

Arguments:

Source	Address of the Internal Register containing the first value
N	Number of Internal Registers of which calculate the mean.
Dest	Internal Register wherein the result is written.

e^x	Performs exponential function of a value

Performs the value of the exponential function with base e (Euler's number) and exponent contained in an Internal Register

Arguments:

SourceX	Internal Register relative to the exponent
Dest	Internal Register wherein the result is written.

Square Root	Calculates the square root of a value

Calculates the square root of a value contained in an Internal Register.

Arguments:

Source	Internal Register relative to the input
Dest	Internal Register wherein the result is written.

Absolute Value	Calculates the absolute value of a value

Calculates the absolute value of a value contained in an Internal Register.

Arguments:

Source	Internal Register relative to the input
Dest	Internal Register wherein the result is written.

Minimum	Calculates the minimum value of N registers

Calculates the minimum value of N Internal Registers values starting from Source

Arguments:

Source	Address of the Internal Register containing the first value
N	Number of Internal Register within find the minimum value
Dest	Internal Register wherein the result is written.

Maximum	Calculates the maximum value of N registers

Calculates the maximum value of N Internal Registers values starting from Source

Arguments:

Source	Address of the Internal Register containing the first value
N	Number of Internal Register within find the maximum value
Dest	Internal Register wherein the result is written.

Linearization	Calculates a value in function of a linearization table

Calculates the Linearization of a value in function of the linearization table selected. Refer to the section 5.1 “Insertion of Linearization Tables” for more information.

Arguments:

Source	Internal Register containing the value to linearize.
Function	Name of the Linearization table to be followed. The button *Table* opens the window “Tables”
Dest	Internal Register wherein the result is written.

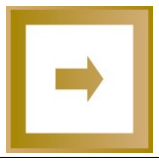
Scale	Executes the proportional scaling of a value of a register

Executes the proportional scaling of a value contained in an Internal Register referring to the input and output ranges. The input range is defined by the limits Zero In and Span In. The output range is defined by the limits Zero Out and Span Out

Arguments:

Source	Internal Register relative to the input
Span In	Maximum value of the input range
Zero In	Minimum value of the input range
Dest	Internal Register relative to the output
Span Out	Maximum value of the output range
Zero Out	Minimum value of the output range

Move	Moves the value of an Internal Register or a Constant to an Internal Register



Writes in an Internal Register the value of a Constant (pre-set) or the value of another Register (copy of a single register). The value will be converted to the format selected for the Register of destination.

Arguments:

Source	Constant or Internal Register from which the value is read
Dest	Internal Register wherein the value is written
Block	Number of repetition of the operation in consecutive registers (Source and Dest)

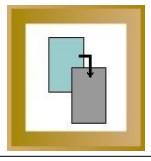
Block Move	Moves a block of registers

Moves a block of N registers starting from Source to another block. It is possible to swap the 8 bit for the uInt registers and the 16 bit for uLong registers.

Arguments:

Source	Address of the Internal Register containing the first value
Number	Number of Internal Register to move
Dest	First Register in which, for the quantity indicated in Number, are moved the registers starting from Source

Copy	Copies a block of registers



Copies the value of an Internal Register to one or more registers.

Arguments:

Source	Internal Register from which the value is read
Dest	First internal register wherein the value is copied
Block	Number of consecutive registers (Dest) to copy the value to

IFBITGOTO	Conditional jump executed in function of the value of one bit

If the value of the bit set in the Source Register is 1, the next function executed by the program is the one connected with the green arrow *true*. If the value of the bit is 0, the next function executed by the program is the one connected with the red arrow *false*.

Arguments:

Source	Internal Register relative to the input
Bit	Bit that determines the jump (0÷15)

IFBITCALL	Call to a subroutine in function of the value of one bit

If the value of the bit set in the Source Register is 1, the program executes a call to the page (subroutine) indicated in the field "*DestTrue*" and so to its first function after the *START.* If the value of the bit set in the Source Register is 0, the program does not execute any call and continues.
The called page must have an univocal name. It is possible to select one of the available pages from drop down list or to create another one writing a new name.

Arguments:

Source	Internal Register relative to the input
Bit	Bit that determines the call (0÷15)
DestTrue	Pointer to the page (subroutine)

CALL	Call to a subroutine

The program executes a call to the page (subroutine) indicated in the field "*Dest*" and so to its first function after the *START.* It executes a jump to the first function block of a subroutine. At the end of the subroutine (ended with the function Return), the program executes the function block after this. The called page must have an univocal name. It is possible to select one of the available pages from drop down list or to create another one writing a new name.

Arguments:

Dest	Pointer to the page (subroutine)

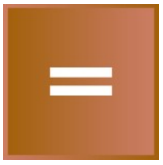
WHILECALL	Continuous call to a subroutine in function of value of a register

The program executes a continuous call to the page (subroutine) indicated in the field "*Dest*" until the value of the register *Source* is lower than a constant or another register indicated in the field *While<.*
If the register is equal or bigger, the program does not execute any continuous call and continues.
The called page must have an univocal name. It is possible to select one of the available pages from drop down list or to create another one writing a new name.

Arguments:

Source	Internal Register relative to the input
While<	Constant or Internal Register to compare to *Source*
Dest	Pointer to the page (subroutine)

A=B	Comparison of value (A=B) between two registers or between a register and a constant



Executes a comparison between InputA and InputB (registers or constants). If the values are equals sets the bits of a destination register *Dest* in function of the setting of the parameter *Mask*.

Arguments:

InputA	Constant or Internal Register relative to the first input to compare
InputB	Constant or Internal Register relative to the second input to compare
Dest	Internal Register wherein the bits will be set
Mask	Mask used to set the register Dest (the button on side opens the window for the setting)

A!=B	Comparison of value (A!=B) between two registers or between a register and a constant

Executes a comparison between InputA and InputB (registers or constants). If the values are different sets the bits of a destination register *Dest* in function of the setting of the parameter *Mask*.

Arguments:

InputA	Constant or Internal Register relative to the first input to compare
InputB	Constant or Internal Register relative to the second input to compare
Dest	Internal Register wherein the bits will be set
Mask	Mask used to set the register Dest (the button on side opens the window for the setting)

A>B	Comparison of value (A>B) between two registers or between a register and a constant

Executes a comparison between InputA and InputB (registers or constants). If InputA is greater than InputB sets the bits of a destination register *Dest* in function of the setting of the parameter *Mask*.

Arguments:

InputA	Constant or Internal Register relative to the first input to compare
InputB	Constant or Internal Register relative to the second input to compare
Dest	Internal Register wherein the bits will be set
Mask	Mask used to set the register Dest (the button on side opens the window for the setting)

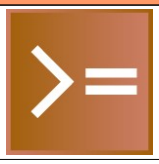
A<B	Comparison of value (A<B) between two registers or between a register and a constant

Executes a comparison between InputA and InputB (registers or constants). If InputA is lower than InputB sets the bits of a destination register *Dest* in function of the setting of the parameter *Mask*.

Arguments:

InputA	Constant or Internal Register relative to the first input to compare
InputB	Constant or Internal Register relative to the second input to compare
Dest	Internal Register wherein the bits will be set
Mask	Mask used to set the register Dest (the button on side opens the window for the setting)

A>=B	Comparison of value (A>=B) between two registers or between a register and a constant



Executes a comparison between InputA and InputB (registers or constants). If InputA is greater or equal than InputB sets the bits of a destination register *Dest* in function of the setting of the parameter *Mask*.

Arguments:

InputA	Constant or Internal Register relative to the first input to compare
InputB	Constant or Internal Register relative to the second input to compare
Dest	Internal Register wherein the bits will be set
Mask	Mask used to set the register Dest (the button on side opens the window for the setting)

A<=B	Comparison of value (A<=B) between two registers or between a register and a constant

Executes a comparison between InputA and InputB (registers or constants). If InputA is lower or equal than InputB sets the bits of a destination register Dest in function of the setting of the parameter *Mask*.

Arguments:

InputA	Constant or Internal Register relative to the first input to compare
InputB	Constant or Internal Register relative to the second input to compare
Dest	Internal Register wherein the bits will be set
Mask	Mask used to set the register Dest (the button on side opens the window for the setting)

Internal Limit — Comparison of value (Internal Limit) between a registers and two values: Maximum and Minimum

Executes a comparison between Input and two values: Maximum Value (MAX) and Minimum Value (MIN).
If the value of Input is between maximum and minimum, sets the bits of a destination register *Dest* in function of the setting of the parameter *Mask*.

Arguments:

Input	Constant or Internal Register relative to the input to compare
MAX	Constant or Internal Register relative to High Limit level
MIN	Constant or Internal Register relative to Low Limit level
Dest	Internal Register wherein the bits will be set
Mask	Mask used to set the register Dest (the button on side opens the window for the setting)

External Limit — Comparison of value (External Limit) between a registers and two values: Maximum and Minimum

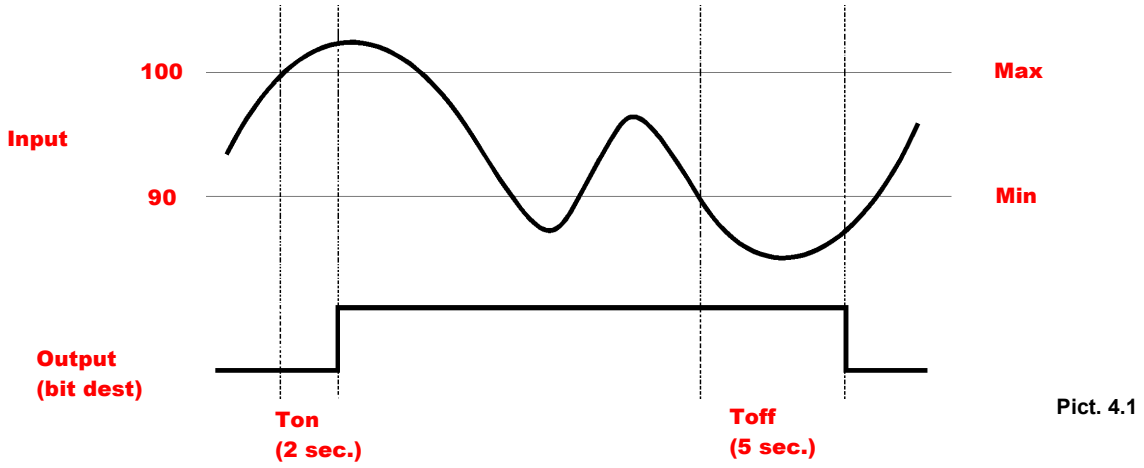
Executes a comparison between Input and two values: Maximum Value (MAX) and Minimum Value (MIN).
If the value of Input is external to maximum and minimum, sets the bits of a destination register *Dest* in function of the setting of the parameter *Mask*.

Arguments:

Input	Constant or Internal Register relative to the input to compare
MAX	Constant or Internal Register relative to High Limit level
MIN	Constant or Internal Register relative to Low Limit level
Dest	Internal Register wherein the bits will be set
Mask	Mask used to set the register Dest (the button on side opens the window for the setting)

Trip — Control of a Trip Alarm

Controls a Trip Alarm with setting of the Trip level, hysteresis and delay time for ON and OFF condition. If the input value is higher than the high trip level (Max) for a time longer than TimerOn, the bits selected in the output mask will be forced to 1. If the input value is lower than the low trip level (Min) for a time longer than TimerOff, the bits selected in the output mask will be forced to 0.
It is possible to use this Function Block to execute the operation “A>B”: set the trip levels with the same value and the delay times TimerOn and TimerOff = 0.
NOTE: the output status is updated at each execution of the Function after the end of the delay time: it is suggested to insert this Function Block in a zone of the Program continuously executed.
The graph **(Pict.4.1)** shows the working of a Trip alarm that goes on if the input signal is higher than 100°C for at least 2 seconds and goes off if the input signal is lower than 90°C for at least 5 seconds.

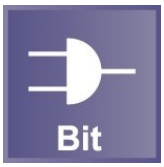


Pict. 4.1

Arguments:

Input	Internal Register containing the value to compare
Max	Constant or Internal Register relative to High Trip level
Min	Constant or Internal Register relative to Low Trip level
Dest	Internal Register wherein the bits will be set
Mask	Mask used to set the register Dest (the button on side opens the window for the setting)
NTimer	Number of Internal Timer to use (0÷15)
TimerOn	Delay time for Trip Alarm activation (ms)
TimerOff	Delay time for Trip Alarm deactivation (ms)

And (bit)	Executes the logical operation “AND” between two single bits.

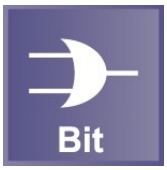


Executes the logical operation “AND” on a single bit between a Register and a Constant or between two Registers. The value will be converted to the format selected for the Register of destination. The address of the source Register and the address of the Register of destination can be the same. After the execution of the logical operation only the bit set in the register of destination will be forced. In case of a 32 bit source Register or Constant (long) and a 16 bit Register of destination (integer), the most significant bits will be ignored.

Arguments:

SourceA	Constant or Internal Register relative to the first operator.
BitA	Bit selected for the first operator
SourceB	Constant or Internal Register relative to the second operator.
BitB	Bit selected for the second operator
Dest	Internal Register wherein the result is written.
BitDest	Bit selected for the register Dest

Or (bit)	Executes the logical operation “OR” between two single bits.



Executes the logical operation “OR” on a single bit between a Register and a constant or between two Registers. The value will be converted to the format selected for the Register of destination. The address of the source Register and the address of the Register of destination can be the same. After the execution of the logical operation only the bit set in the register of destination will be forced. In case of a 32 bit source Register or constant (long) and a 16 bit Register of destination (integer), the most significant bits will be ignored.

Arguments:

SourceA	Constant or Internal Register relative to the first operator.
BitA	Bit selected for the first operator
SourceB	Constant or Internal Register relative to the second operator.
BitB	Bit selected for the second operator
Dest	Internal Register wherein the result is written.
BitDest	Bit selected for the register Dest

XOr (bit)	Executes the logical operation “XOR” (Exclusive Or) between two single bits.

Executes the logical operation “XOR” on a single bit between a Register and a constant or between two Registers. The value will be converted to the format selected for the Register of destination. The address of the source Register and the address of the Register of destination can be the same. After the execution of the logical operation only the bit set in the register of destination will be forced. In case of a 32 bit source Register or constant (long) and a 16 bit Register of destination (integer), the most significant bits will be ignored.

Arguments:

SourceA	Constant or Internal Register relative to the first operator.
BitA	Bit selected for the first operator
SourceB	Constant or Internal Register relative to the second operator.
BitB	Bit selected for the second operator
Dest	Internal Register wherein the result is written.
BitDest	Bit selected for the register Dest

And (word)	Executes the logical operation “AND” between two values.

Executes the logical operation “AND” between a Register and a constant (mask) or between two Registers. The value will be converted to the format selected for the Register of destination. The address of the source Register and the address of the Register of destination can be the same.
After the execution of the logical operation, only the bits set as 1 in the mask will be forced; the bits set as 0 won't be modified. In case of a 32 bit source Register or constant (long) and a 16 bit Register of destination (integer), the most significant bits will be ignored. It is possible to use this function to force one or more bits of a Register as 0 (in the mask set as 0 the bits to force, set as 1 the other bits).

Arguments:

SourceA	Constant or Internal Register relative to the first operator.
SourceB	Constant or Internal Register relative to the second operator.
Dest	Internal Register wherein the result is written.
MaskDest	Mask used to set the register Dest (the button on side opens the window for the setting)

Or (word)	Executes the logical operation ”OR” between two values.

Executes the logical operation “OR” between a Register and a Constant (mask) or between two Registers. The value will be converted to the format selected for the destination Register. The address of the source Register and the address of the Register of destination can be the same.
After the execution of the logical operation only the bits set as 1 in the mask will be forced; the bits set as 0 won't be modified. In case of a 32 bit source Register or constant (long) and a 16 bit Register of destination (integer), the most significant bits will be ignored. It is possible to use this function to force one or more bits of a Register as 1 (in the mask set as 1 the bits to force, set as 0 the other bits).

Arguments:

SourceA	Constant or Internal Register relative to the first operator.
SourceB	Constant or Internal Register relative to the second operator.
Dest	Internal Register wherein the result is written.
MaskDest	Mask used to set the register Dest (the button on side opens the window for the setting)

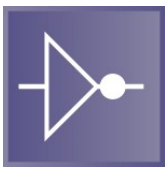
Xor (word)	Executes the logical operation “XOR” (Exclusive Or) between two values.

Executes the logical operation “XOR” between a Register and a constant (mask) or between two Registers. The value will be converted to the format selected for the Register of destination. The address of the source Register and the address of the Register of destination can be the same.
After the execution of the logical operation only the bits set as 1 in the mask will be forced ; the bits set as 0 won't be modified. In case of a 32 bit source Register or constant (long) and a 16 bit Register of destination (integer), the most significant bits will be ignored. It is possible to use this function to invert (NOT) one or more bits of a Register (in the mask set as 1 the bits to invert, set as 0 the other bits).

Arguments:

SourceA	Constant or Internal Register relative to the first operator.
SourceB	Constant or Internal Register relative to the second operator.
Dest	Internal Register wherein the result is written.
MaskDest	Mask used to set the register Dest (the button on side opens the window for the setting)

NOT	Executes the inversion of one or more bits of a Register



Executes the inversion of one or more bit of a Register. After the execution of the logical operation will be forced only the bits set as 1 in the mask; the bits set as 0 won't be modified.

Arguments:

Source	Internal Register containing the value
Dest	Internal Register wherein the result is written.
MaskDest	Mask used to set the register Dest (the button on side opens the window for the setting)

Bit Set	Set to logic state High the bits of a register

Set to logic state High one or more bits of a register *Dest* in function of the mask *MaskDest.* It will be set only the bits set to 1 in the mask, the other will not be modified.

Arguments:

Dest	Internal Register wherein the bits will be set
MaskDest	Mask used to set the Register (the button on side opens the window for the setting)

Bit Reset	Set to logic state Low the bits of a register

Set to logic state Low one or more bits of a register *Dest* in function of the mask *MaskDest.* It will be reset only the bits set to 1 in the mask, the other will not be modified.

Arguments:

Dest	Internal Register wherein the bits will be set
MaskDest	Mask used to reset the Register (the button on side opens the window for the setting)

Rising Edge	Intercepts the Rising Edge of one or more bits of a register

If the bits of the register *Source* have been selected in *MaskSource* and change state (from 0 to 1), then the corresponding bits of the register *Dest* will be forced to 1.
The register *Latch* is used as a temporary register and contains the previous value of the register *Source.*

Arguments:

Source	Internal Register containing the value.
MaskSource	Bit mask applied to the register (the button on side opens the window for the setting)
Latch	Temporary register (contains the previous value of the register *Source*)
Dest	Internal Register wherein the bits will be set

Falling Edge	Intercepts the Falling Edge of one or more bits of a register

If the bits of the register *Source* have been selected in *MaskSource* and change state (from 1 to 0), then the corresponding bits of the register *Dest* will be forced to 1.
The register *Latch* is used as a temporary register and contains the previous value of the register *Source.*

Arguments:

Source	Internal Register containing the value.
MaskSource	Bit mask applied to the register (the button on side opens the window for the setting)
Latch	Temporary register (contains the previous value of the register *Source*)
Dest	Internal Register wherein the bits will be set

Shift Right	Shift to right the bits of a Register

Executes the shift of a Register to right: all of the bits are shifted of N positions to right. The most significant bits will be forced to 0.

Arguments:

Source	Constant or Internal Register containing the value.
Number	Number of shift to execute.
Dest	Internal Register wherein the result is written.

Shift Left	Shift to left the bits of a Register.

Executes the shift of a Register to left: all of the bits are shifted of N positions to left. The least significant bits will be forced to 0.

Arguments:

Source	Constant or Internal Register containing the value.
Number	Number of shift to execute.
Dest	Internal Register wherein the result is written.

Rotate Right	Rotates to right the bits of a Register

Executes the rotation of a Register to right: all of the bits are shifted of N positions to right. At each shift the most significant bit receives the value of the least significant bit.

Arguments:

Source	Constant or Internal Register containing the value.
Number	Number of shift to execute.
Dest	Internal Register wherein the result is written.

Rotate Left	Rotates to left the bits of a Register

Executes the rotation of a Register to left: all of the bits are shifted of N positions to left. At each shift the least significant bit receives the value of the most significant bit.

Arguments:

Source	Constant or Internal Register containing the value.
Number	Number of shift to execute.
Dest	Internal Register wherein the result is written.

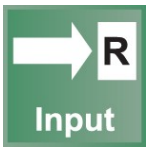
Read Register	Reads registers from a generic Modbus Slave device with function Modbus 03

Reads the value of N registers from a generic Modbus slave device and writes the read values in the selected Internal Registers. To read it uses the function Modbus 03. In case of missing response or wrong response by the slave device, the Registers of destination are not updated and the value of the System Register “COM Errors” is increased.

Arguments:

Address	Modbus address of the slave device (1÷247)
Register	Address of the first Register to read (the mapping Registers starts from 0)
Number	Numbers of Registers to read (1÷16)
Dest	Address of the first Internal Register wherein the read values are written to.
Delay	Delay time between the reception of the response and the execution of the next instruction

Read Input	Reads registers from a generic Modbus Slave device with function Modbus 04



Reads the value of N registers from a generic Modbus slave device and writes the read values in the selected Internal Registers. To read it uses the function Modbus 04. In case of missing response or wrong response by the slave device, the Registers of destination are not updated and the value of the System Register “COM Errors” is increased.

Arguments:

Address	Modbus address of the slave device (1÷247)
Register	Address of the first Register to read (the mapping Registers starts from 0)
Number	Numbers of Registers to read (1÷16)
Dest	Address of the first Internal Register wherein the read values are written to.
Delay	Delay time between the reception of the response and the execution of the next instruction

Write Single	Writes single register of a generic Modbus Slave device with function Modbus 06

Writes the value of an Internal Register in a register of a generic Modbus Slave device. To write it uses the function Modbus 06. In case of missing response or wrong response by the slave device, the Registers of destination are not updated and the value of the System Register “COM Errors” is increased.

Arguments:

Address	Modbus address of the slave device (1÷247)
Register	Address of the Register to write (the mapping Registers starts from 0)
Source	Address of the Internal Register where the value to write is contained
Delay	Delay time between the reception of the response and the execution of the next instruction

Write Multiple — Writes multiple registers of a generic Modbus Slave device with function Modbus 16

Writes the values of N Internal Registers in the registers of a generic Modbus Slave device. To write it uses the function Modbus 16. In case of missing response or wrong response by the slave device, the Registers of destination are not updated and the value of the System Register “COM Errors” is increased.

Arguments:

Address	Modbus address of the slave device (1÷247)
Register	Address of the Register to write (the mapping Registers starts from 0)
Number	Number of the registers to write (1÷16)
Source	Address of the first Internal Register from which the values to write are contained
Delay	Delay time between the reception of the response and the execution of the next instruction

Read Device — Read registers from a Modbus Slave DAT3000/DAT10000 series device

Reads the I/O values from a Modbus slave DAT3000/DAT10000 series device and writes the values in the Internal Registers. The function will generate the proper Modbus command and will process the response.
In case of missing response or wrong response by the slave device, the Registers of destination are not updated and the value of the System Register “COM Errors” is increased.
Refer to the technical documentation of the device for the complete description of the I/O Registers.

Arguments:

Delay	Delay time between the reception of the response and the execution of the next instruction
Device	Type of device to read
Remote Address	Modbus address of the slave device (1÷247)
Resource	Type of resource to read (analog inputs, digital inputs, etc...)
From	First resource to read
To	Last resource to read
Dest	Address of the first Internal Register wherein the read values are written to.

Write Device — Write registers of a Modbus Slave DAT3000/DAT10000 series device

Writes the values of *N* Internal Registers in the I/O Register of a Modbus slave DAT3000/DAT10000 series device. The function will generate the proper Modbus command and will process the response. In case of missing response or wrong response by the slave device, the Registers of destination are not updated and the value of the System Register “COM Errors” is increased.
Refer to the technical documentation of the device for the complete description of the I/O Registers.

Arguments:

Delay	Delay time between the reception of the response and the execution of the next instruction
Device	Type of device to write
Remote Address	Modbus address of the slave device (1÷247)
Resource	Type of resource to write (analog outputs, digital outputs, etc...)
From	First resource to write
To	Last resource to write
Source	Address of the first Internal Register from which the values to write are contained

Force Single Coil — Writes a single coil of a generic Modbus Slave device with function Modbus 05

Writes the value of a coil in a generic Modbus Slave device. To write it uses the function Modbus 05. In case of missing response or wrong response by the slave device, the coil of destination is not updated and the value of the System Register “COM Errors” is increased

Arguments:

Delay	Delay time between the reception of the response and the execution of the next instruction
Address	Modbus address of the slave device (1÷247)
Coil	Value to write to force the value of the coil. Write 0 to force the coil's value to 0, write 255 to force the coil's value to 1
Source	Address of the first Internal Register from which the values to write are contained

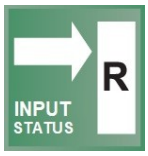
Read Coil Status — Reads the ON/OFF state of coils with mapping 0x from a slave.

Reads the ON/OFF state of coils with mapping 0x from a slave. To read, it uses the Modbus function 01. In case of missing response or wrong response by the slave device, the Registers of destination are not updated and the value of the System Register “COM Errors” is increased.

Arguments:

Delay	Delay time between the reception of the response and the execution of the next instruction
Address	Modbus address of the slave device (1÷247)
Coil	Address of the first coil to read (the mapping of Coils starts from 0)
Number	Number of coils to read.
Dest	Address of the first Internal Register wherein the read values are written to. The first coil is written to the bit 0 of register.

Read Input Status — Reads the ON/OFF state of coils with mapping 1x from a slave.



Reads the ON/OFF state of coils with mapping 1x from a slave. To read, it uses the Modbus function 02. In case of missing response or wrong response by the slave device, the Registers of destination are not updated and the value of the System Register “COM Errors” is increased.

Arguments:

Delay	Delay time between the reception of the response and the execution of the next instruction
Address	Modbus address of the slave device (1÷247)
Coil	Address of the first coil to read (the mapping of Coils starts from 0)
Number	Number of coils to read.
Dest	Address of the first Internal Register wherein the read values are written to. The first coil is written to the bit 0 of register.

Timer — Activation of an Internal Timer

Sets an Internal Timer (0÷15) and starts to count. During the count, the bit relative to the Timer selected in the System Register “Timers Enable”, will be forced to 0. At the end of the count, the bit will be forced to 1.
It is possible to check the status of the bit to determine the end of the time set. The duration of the timer is set by a constant *mSec* (until 65535) or if the constant is 0 by a register *Reg*.

Arguments:

Timer	Number of the Internal Timer to enable (0÷15)
mSec	Timer Pre-set (milliseconds)
Reg	Register used to preset the Timer if mSec is 0

Call Page — Call and refresh the page on the Display

Call the refresh of the page visualized on the display. Once the function is executed the program continues with the next function.

5.1 – INSERTION OF LINEARIZATION TABLES

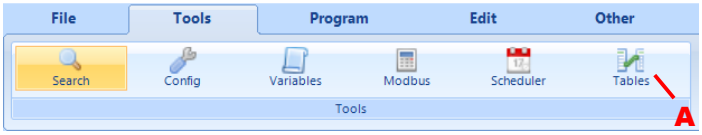
To insert or modify the Tables click the proper button in the menu bar **(Pict.5.1-A)** or the button *Table* of the function Linearization. Then it will be opened the window "*Tables*" **(Pict.5.2)**

To upload the points of a table from a file, click on the "*Load from File*" button **(Pict.5.2-A)**; the parameters relative to the table like the name, the number of points and the input and output values for each point will be loaded.
It is possible to modify the Name of the table **(Pict.5.2-B)** (the name must be unique, because it is used in the program to call that table). Set the number of points to insert in the table **(Pict.5.2-C)**.
It is possible to insert until 32 points. Each point is defined by the input and output values. The input values must be inserted in increasing order, while the output values can be inserted both in increasing and decreasing order. The example **(Pict.5.2-D)** shows how to create a 8 points table to linearize a RTD temperature sensor and to obtain the conversion Ohm/°C **(Pict.5.3)**.

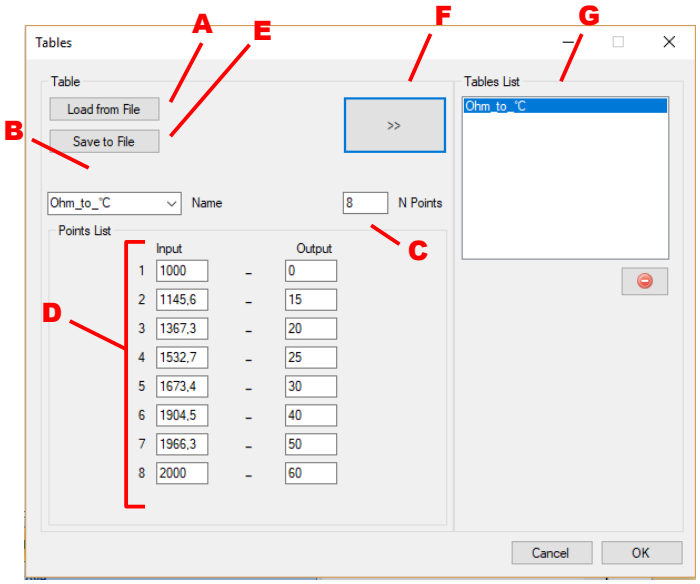
When the insertion of points is complete, it is possible to save the table in a file, clicking on the "*Save to File*" button **(Pict.5.2-E);** by this command the name of the table, the number of points and the input and output values per each points will be saved.

To insert the table inside the Program, click on the ">>" button **(Pict.5.2-F)**. The "Table List" **(Pict.5.2-G)** will be updated with the name of the table just inserted and shows the tables available for the Program.

In the Program, when the block function Linearization is inserted, it is possible to select one of the tables available.
When the Function Block is recalled by the Program, the Controller executes a control between the value contained into the Internal Register and the points of the selected table and calculates by interpolation the output value.
In the example for an input value of 1789 Ohm, will be calculated an output of 35 °C (without linearization function the output should be 47,3 °C).



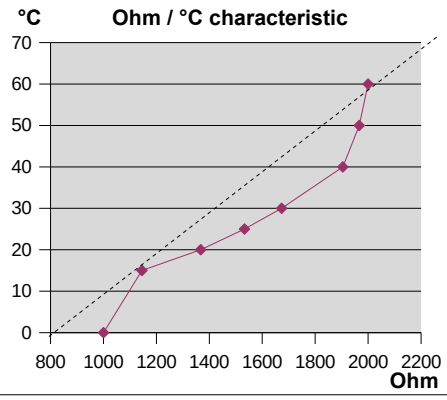
Pict. 5.1



Pict. 5.2

Example of table provided from the manufacturer of the sensor :

°C	Ohm
0	1000
15	1145,6
20	1367,3
25	1532,7
30	1673,4
40	1904,5
50	1966,3
60	2000



Pict. 5.3

5.2 – IP ADDRESSES TABLE "Server / Slave" TCP DEVICES

The DAT9000 series devices can operate as master not only on the RS485 serial port but also on the Ethernet interface using the Modbus TCP / IP protocol.

To due this it is necessary to set the IP addresses of the *"Server"* devices to which Modbus TCP commands of reading and writing are sent. The *"Server"* devices that can be inserted in the table are at most 8 **(Pict.5.4)**.

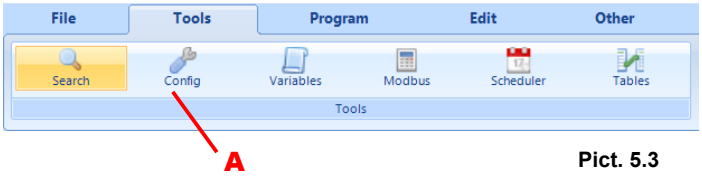
From the *Tools → Configuration* menu **(Pict.5.3)** click the frame IP Table to access it **(Pict.5.4)**

The IP table allows to associate to each "*IP Address*" **(Pict.5.4-B)** of the Server device, an "*RTU Node ID*" **(Pict.5.4-A)** that will be the equivalent of the Modbus address / node used in the standard reading and writing functions. In fact, in the construction of the project, the same function blocks of Read Holding, Read Input, etc. will be used, independently of the Modbus protocol.
Each server device inserted in the table has a physical address called "*Device Address*" **(Pict.5.4-C)** which, in most cases, is not significant (as per DAT8000 series). This address corresponds to the real Modbus address / node of the slave to be interrogated and is used by the controller to build the frame of the TCP / IP Modbus protocol. In the table it is necessary to insert the communication port that, in case of Modbus TCP, is 502 **(Pict.5.4-D)**

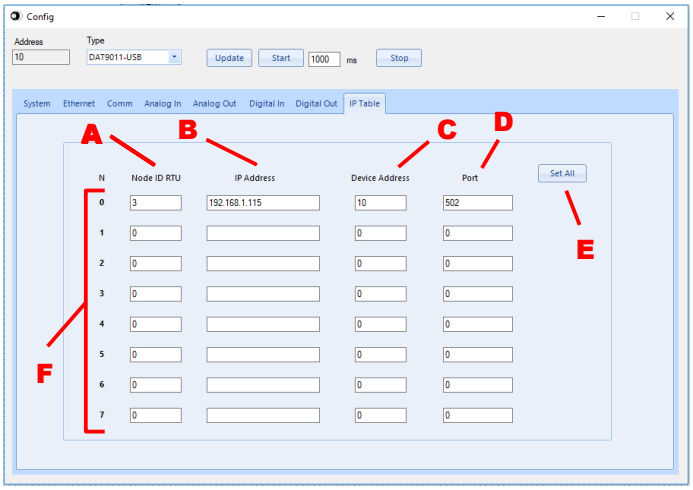
Once you have finished inserting / editing the IP table, click “Set All” button to save the data**(Pict.5.4-E)**.

In the diagram, in the read and write function blocks, the data inserted in the IP table in the column *RTU Node ID* **(Pict.5.4-A)** must be used as the Modbus address (Address) **(Pict.5.5)**.

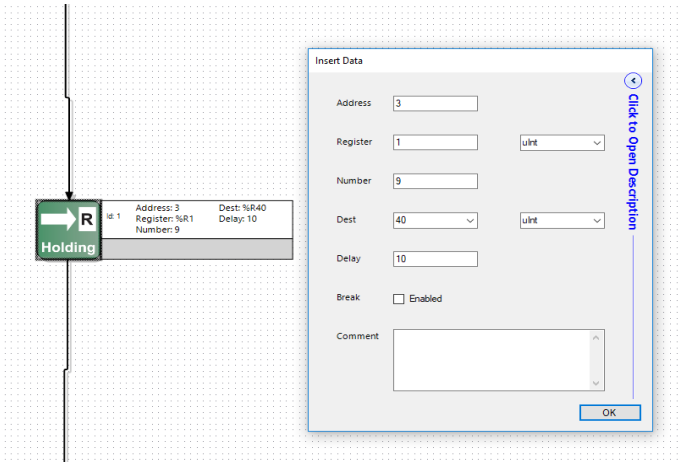
The IP table can also be edited via web pages in the page "IP Table Configuration" **(Pict.5.6)**. In this page it is also possible to test the connected devices when the controller is in STOP, by clicking on *"Devices Test"* **(Pict.5.6-A)**.



Pict. 5.3



Pict. 5.4

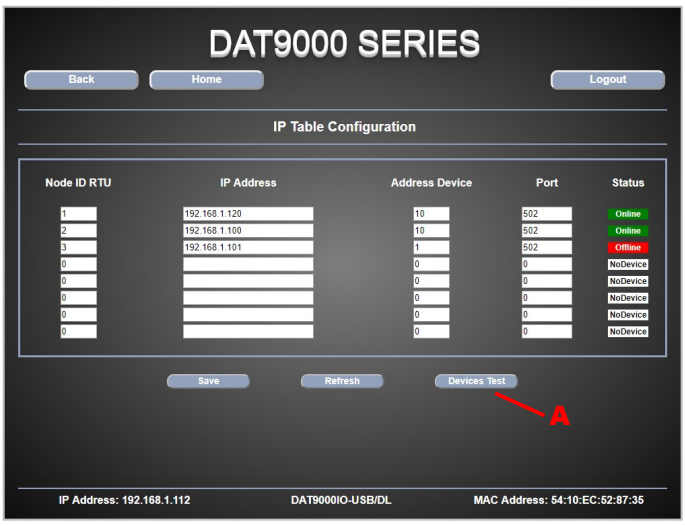


Pict. 5.5

NOTES:

1) the DAT9000 series can poll both devices connected to the Ethernet interface in Modbus TCP and slave devices connected to the RS485 Master serial port as long as the Modbus addresses of the devices on the RS485 are different from those associated with the IP devices of the server devices in the table (RTU Node ID).

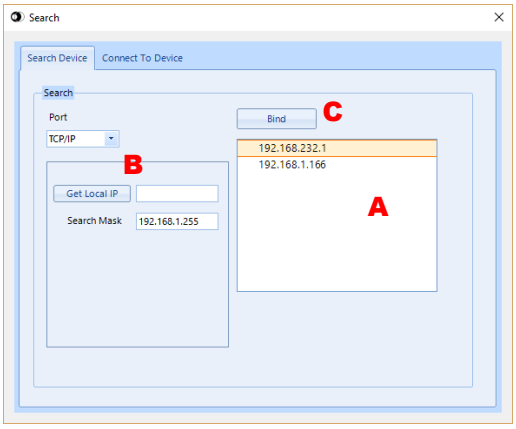
2) Before testing the devices via a web interface, make sure that the controller is in STOP mode.



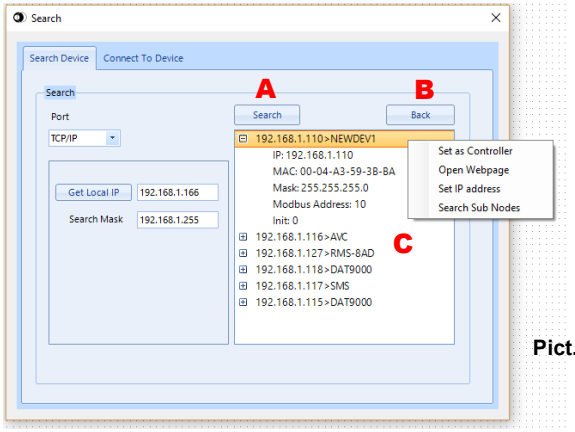
Pict. 5.6

6.1 – SEARCH FOR THE DEVICES CONNECTED OVER ETHERNET

Connect the Controller to the Ethernet network and power-on it (refer to the data-sheet).
Click "*Search*" in the menu bar to open the window "*Search"* **(Pict.6.1).**
In the first list **(Pict.6.1-A)** there are the available networks of the PC in use.
Click the button "*Get Local IP*" **(Fig.6.1-B)** to visualize the local IP of the Personal Computer in the network in use.
Once identified the correct network, select it and click the button "Bind" **(Pict.6.1-C)** to assign the parameters of search function.
Then click the button "*Search"* **(Pict.6.2-A)** to search devices in the Net. If the network is correct, and so there are one or more devices, the devices found will appear in the list. If the network is not correct and the user wants to bind another network, click the button "*Back*"**(Pict.6.2-B)**.
For each device found it is possible to read the relative IP address, the MAC number, the Mask, the Modbus Address and the Init status.
Click the right button of the mouse to open a drop down menu with some additional functions **(Pict.6.2-C)**.
Click *Set as Controller* to connect the device.
Click *Open Webpage* to open the web page of the device on the browser.
Click *Set IP address* to set a new IP address for the selected device.



Pict. 6.1



Pict. 6.2

6.2 – MANUAL CONNECTION TO THE DEVICE

In the window "*Search*" there is a section for manual connection through Ethernet or COM RS485/uUSB ports. Here it is possible to set communication parameters.

For the Ethernet **(Pict.6.3)**:
IP Address: IP address of the device
Port: Modbus/TCP socket reserved port (*502* for direct connection)
Timeout (mSec): Receiving Timeout for TCP commands
Modbus ID :Modbus node address (1 ÷ 247)

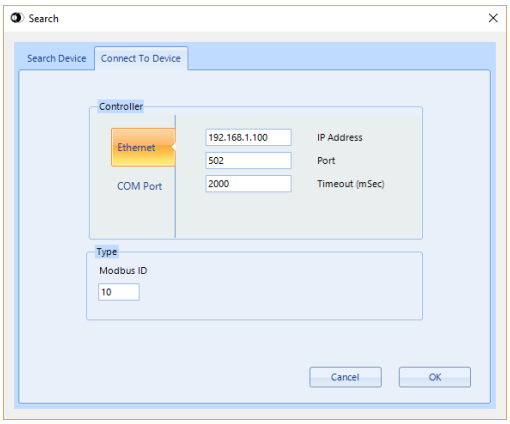
For COM port **(Pict.6.4)**:
Port Name: Name of the available COM ports
Baud Rate: Rate of serial transmission
Data Bits: Number of bits of transmission
Parity: Parity of the transmission
Stop Bits: Stop bit of the transmission
Handshake: Handshake Mode
Modbus ID : Modbus node address (1 ÷ 247)

To confirm the parameters, click on the "*OK*" button.

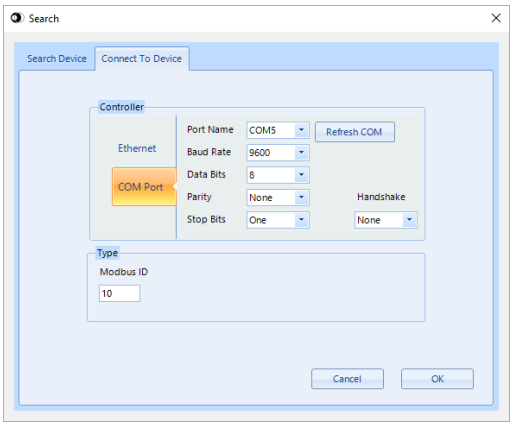
Click the button "Connect" **(Pict.6.5)** to connect automatically to the last device.
If the connection ends well, the message "*Connected*" will be visualized in the Status bar and in the Log panel; in case of error, refer to the section "*Troubleshooting*" to solve the problem.

From this moment all of the reading, writing, programming and debugging operations will be sent only to the selected Controller.
If the user has to change the Controller, it is necessary to disconnect the Controller in use and connect to another Controller following one of the modalities described above.

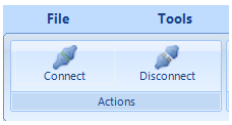
NOTE: the controller goes automatically offline after the socket timeout set into it has expired. In this case connect again to the device



Pict. 6.3



Pict. 6.4

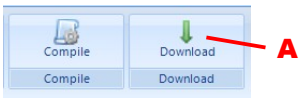


Pict. 6.5

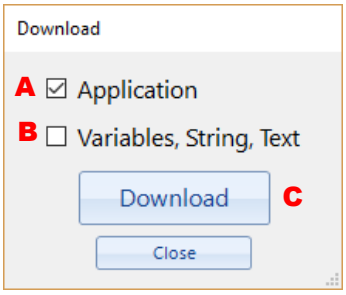
6.3 – DOWNLOAD THE PROGRAM

When the Program is complete and if the compiling ends correctly it is possible to download it in the internal memory of Controller.
Clicking the button "*Download*" in the menu bar **(Pict.6.6-A)**, the "*Download*" window will be opened **(Pict.6.7)**. The Download operations are allowed only in "Debug" mode (refer to the section "Debug Mode").
It is possible to set two options:
"*Application*" **(Pict.6.7-A)** – Download the program in the internal memory of the Controller.
"*Variables, String, Text*" **(Pict.6.7-B)** – Download the settings of the variables, strings and texts.

Click the button "*Download*" **(Pict.6.7-C)** to download the selected options in the Controller.



Pict. 6.6

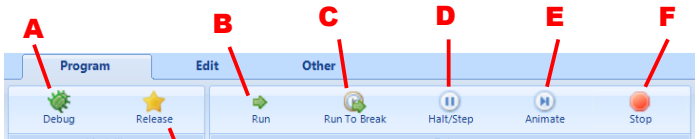


Pict. 6.7

6.4 - DEBUG MODE

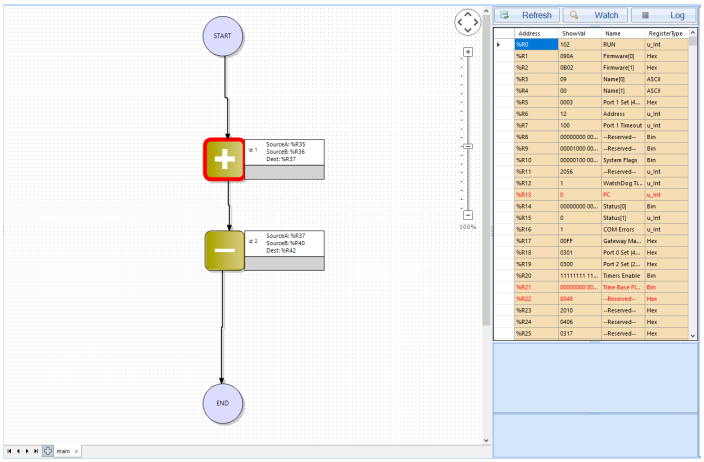
During the development of the Program, if the Controller is connected, click the button "*Debug*" **(Pict.6.8-A)** to activate the "Debug" mode.
In the Status bar the message "*Debug*" **(Pict.6.9)** will be visualized and in the menu bar the buttons to execute the following debug operations will be enabled:
"*Run*" **(Pict.6.8-B)** – Executes the Program continuously.
"*Run To Break*" **(Pict.6.8-C)** – Executes the Program up to the Break point
"*Halt/Step*" **(Pict.6.8-D)** – Interrupts the execution of the Program ("Run" condition) / executes the Program step by step ("Stop" condition)
"*Animate*" **(Pict.6.8-E)** – Simulates the evolution of the Program flow executing it step by step.
"*Stop*" **(Pict.6.8-F)** – Blocks the Program and reset it to the first Function Block.

Then it is possible to follow the flow of the program, reading the state of the Controller and input registers from the register table. When the Program is in Halt, Step or Animate the position of the function block to be executed is identified with the red border. In the same time in the register table the value of the Program Counter ("PC") is updated **(Pict.6.10)**.



Pict. 6.8

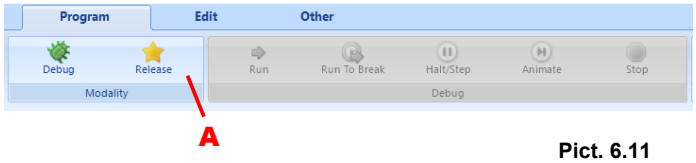
Pict. 6.9



Pict. 6.10

6.5 - RELEASE MODE

After the phases of development and Debug it is possible to proceed with the "*Release*" mode, clicking the button "*Release*" **(Pict.6.11-A)**.
In the Status bar the message "*Release*" **(Pict.6.12)** will be visualized and in the menu bar the buttons to execute the debug and download operations will be disabled.
In the "*Release*" Modality, at the power-on, the Controller will be automatically set in "Run" condition, loading in the RAM memory the Program saved in the Internal Flash memory.
In this modality it is possible to read and write the Internal Registers.



Pict. 6.11

Pict. 6.12

6.6 - INIT MODE

All DAT9000 series devices are equipped with the INIT mode. This is a mode to access the device with the default parameters regardless of the configuration stored in EEPROM.
Following is the procedure for using the INIT mode which is also indicated on the User Guide of each device in the ***"PROCEDURES"*** section:

Over Ethernet:
- IP Address: XXX.XXX.XXX.XXX provided by the DHCP if is enabled or 192.168.1.174 if DHCP is disabled (verify that the IP is not already used and that the PC belongs to the same subnet)
- Modbus address: 10

By these parameters it is possible to access the device in INIT mode to configure it or see the stored configuration.
To enter INIT, follow the procedure below:
1) Turn off the device;
2) Connect the INIT terminal to the -V terminal as shown in the technical datasheet of the device.
3) Turn on the device;
4) Connect to the device using the default parameters shown above.

When the user finishes working in INIT mode:
1) Turn off the device;
2) Remove the INIT connection;
3) Turn on the device and connect with the parameters known or configured in INIT mode.

Over RS485 slave or uUSB:
1) Switch off the device.
2) Connect only the device to be programmed to the RS485 slave or uUSB network.
3) Connect the INIT terminal to terminal V-.
4) Turn on the device.
5) Set the communication port with the following values
   - Mode = Modbus RTU
   - Baud-rate = 9600 bps
   - Parity = None
   - Bit number = 8
   - Stop bit = 1
   - Modbus address = 10

6) Read or program the desired settings in the registers using the Dev9k software.
7) Switch off the device.
8) Disconnect the INIT terminal from terminal V-.
9) Set the communication port with the programmed baud-rate
10) The module answers with the programmed address

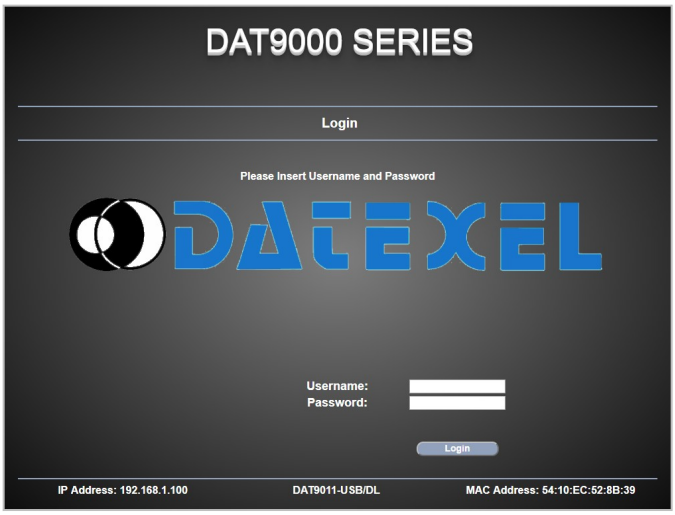
ATTENTION:

In INIT mode, the device's Debug / Stop mode is also forced. Therefore any program stored in EEPROM will not be executed.

6.7 – WEB INTERFACE

Through a Web Browser it is possible to access the Controller's Web Server to view the Web pages containing the configuration data and to download the Log files.
To access the integrated web server, open a browser on your PC and type the IP address of the device in the address bar of the browser.
- Factory IP Address: 192.168.1.100
Warning: make sure that the PC is in the same subnet as the device in use (see user guide of the device).
The factory / default access credentials that are requested on the "Login" page **(Pict. 6.13)** are:
- **Username:** Fact_user
- **Password:** Fact_pwd

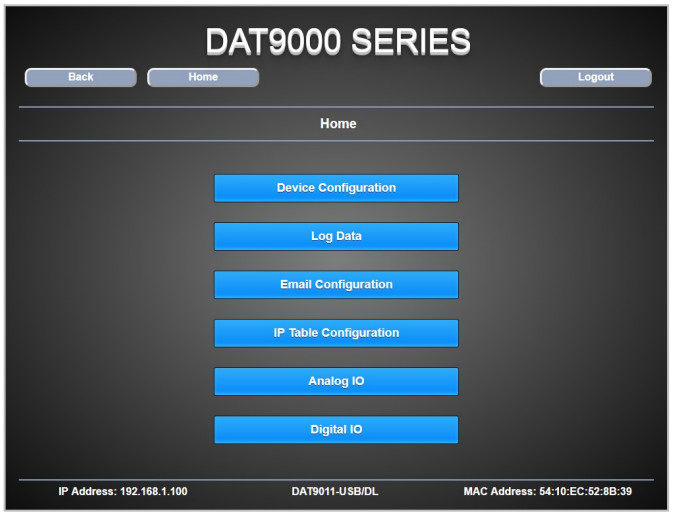
After the first login, it is possible to change the credentials in the "Username and Password" section and the network parameters in the "Network Settings" page.



Pict. 6.13

From the **Home** page **(Pict. 6.14)** it is possible to access the following pages:

Device Configuration: allows access to the device configuration menu.
From this menu it is possible to access the pages:
- System Configuration: configuration of system parameters (Netbios Name, Watchdog, modbus address, ...);
- Username and Passoword: modify access credentials;
- Date and Time: time and date setting;
- Network Settings: network parameters settings (IP address, Subnet mask, Gateway, DNS);
- DDNS Settings: setting connection parameters to the Dynamic DNS server, for remote connection with a dynamic IP (requires to subscribe to the service www.dyndns.com);
- Modbus Settings: setting of the communication parameters of the Master and Slave / uUSB serial ports;
- Software Update: allows the user to upload a new release of firmware or web pages;



Pict. 6.14

Log Data: through this page **(Pict. 6.15)** it is possible to read the contents of the USB stick or SD card (depending on the model) and download the .CSV files with the Data Logger recordings.

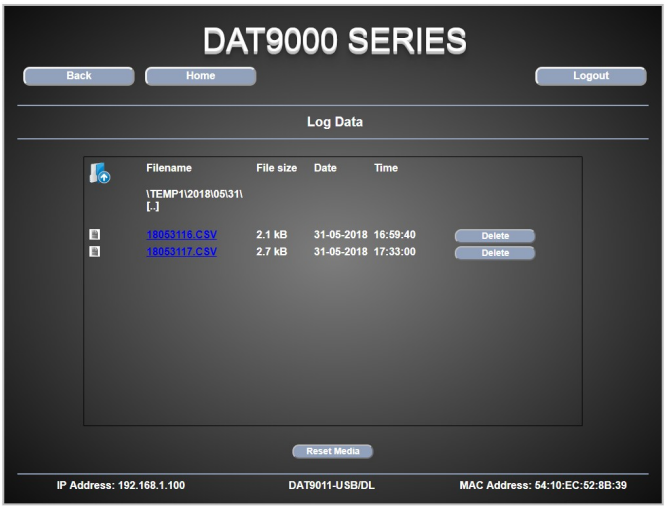
Email Configuration: it allows to modify the parameters of the sender and recipients' email, the message body and the parameters related to the outgoing mail server.

Warning:
1) it is possible to insert only body of the message
*2) the outgoing mail server (SMTP) must allow the use of port **25** (not encrypted)*

IP Table Configuration: in this page it is possible to enter the IP addresses of the "Server" devices to be queried by the controller with Modbus TCP / IP protocol. A maximum of 8 "Server" devices can be entered in the table.

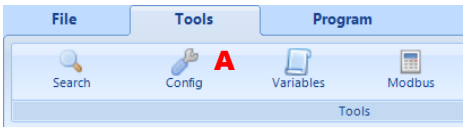
Analog IO: this page is only available for the DAT9011-USB and DAT9011-DL models as it allows the user to configure the analog inputs, display the measurements, write to the analog outputs and set the values of Safe and PowerUp.

Digital IO: this page is available for all models of the DAT9000-USB and DAT9000-DL models, as it allows to display the status of the digital inputs and outputs as well as the count of the pulse counters. It is also possible to change the status of the digital outputs, the Safe and PowerUp conditions.



Pict. 6.15

6.8 - CONFIGURATION WINDOW



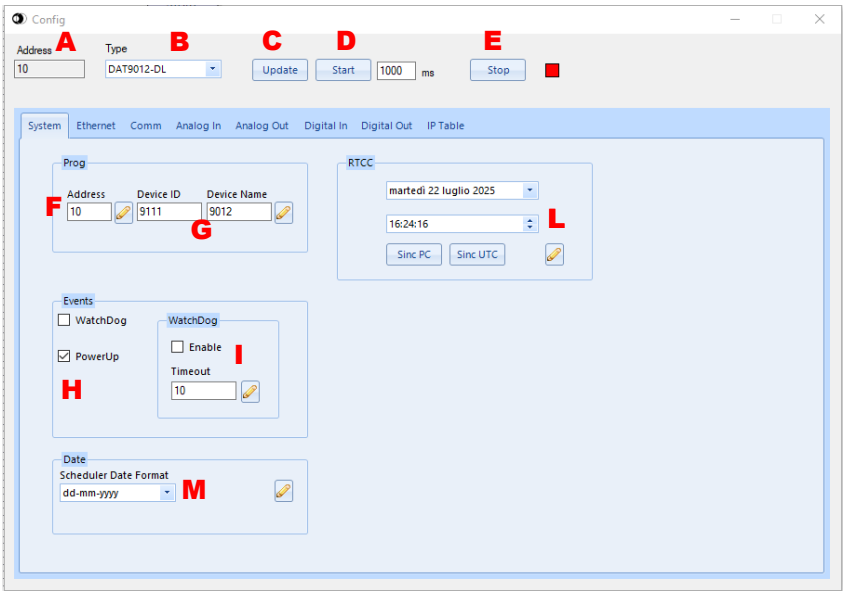
Pict. 6.16

Click in the menu bar the button "*Config*" to show the device configuration window **(Pict.6.16-A)**

In the configuration window **(Pict.6.17)** it is possible to visualize most of the editable parameters of a device.

The fields on the top of the window are:
- the Modbus address of the device to connect to **(Pict.6.17-A)**
- the model of the device connected **(Pict.6.17-B)**
- the button "*Update*" used to update the window and so the parameters available for the selected device **(Pict.6.17-C)**
- the button "*Start*" used to update cyclically the window and beside it there is the frequency of update **(Pict.6.17-D)**
- the button "*Stop*" to stop the cycling update **(Pict.6.17-E)**

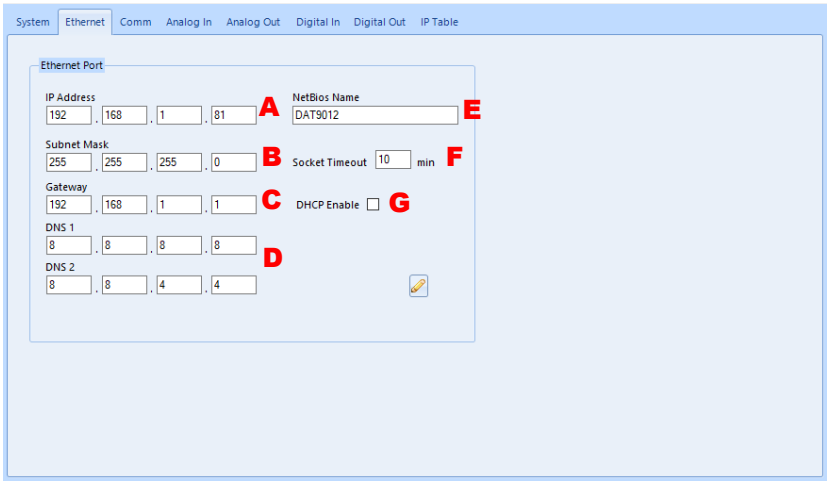
The window is divided in tabs and internal sections visible or not in function of the firmware of the selected device. To modify a parameter usually there is a button with pencil near it.



Pict. 6.17

System

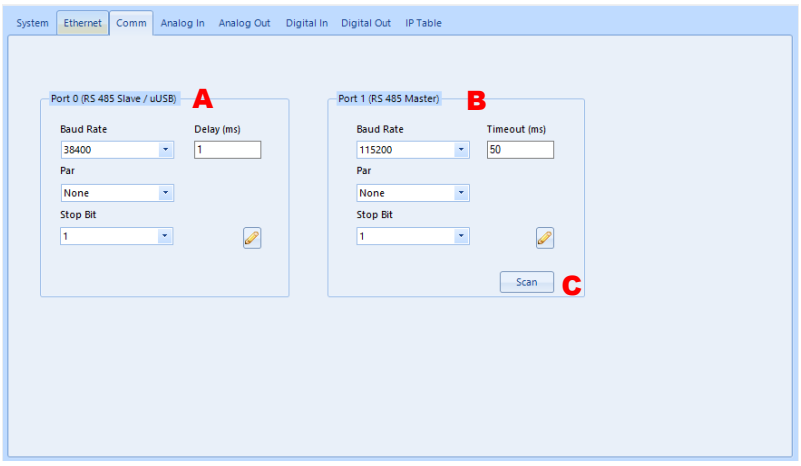
The System Tab contains the system data of the device:
- the Modbus address of the device **(Pict.6.17-F)**
- the ID and the name of the device **(Pict.6.17-G)**
- the WatchDog state and PowerUp state **(Pict.6.17-H)**
- the enable option of WatchDog with relative timeout time **(Pict.6.17-I)**
- the date and time measured by the device's RTC with sync button **(Pict.6.17-L)**
- the scheduler Date format (**Pict.6.17-M**) that allows the user to set the following formats for the date in the log file and web page:
dd-mm-yyyy
mm-dd-yyyy
- in the display there is a section to set brightness, contrast and negative/positive option of display.



Pict. 6.18

Ethernet

The Ethernet Tab contains the Ethernet configuration data of the device:
- IP address of the device **(Pict.6.18-A)**
- Subnet Mask of the device **(Pict.6.18-B)**
- Gateway of the device **(Pict.6.18-C)**
- DNS 1 and DNS 2 of the device **(Pict.6.18-D)**
- NetBios Name of the device **(Pict.6.18-E)**
- Socket Timeout of the device expressed as minutes **(Pict.6.18-F)**
- Enable of DHCP for the device **(Pict.6.18-G)**

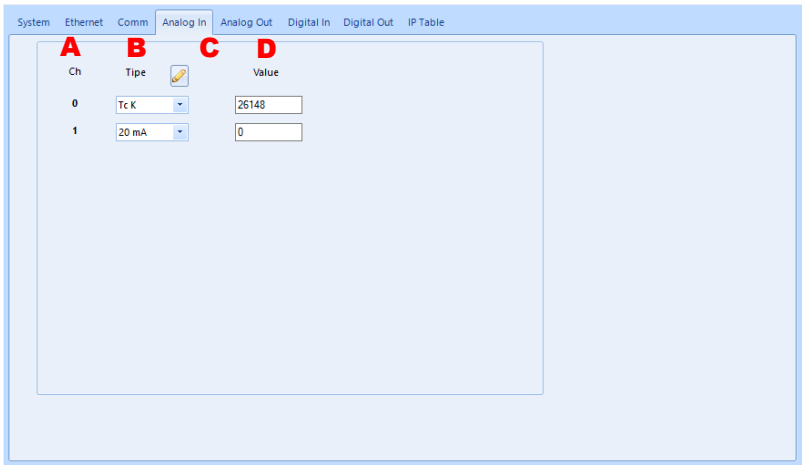


Pict. 6.19

Comm

The Comm Tab contains the configuration data of the Com Ports available on the device.
- Port 0 (RS485 Slave) with Baud Rate, Par, Stop Bit and Delay **(Pict.6.19-A)**
- Port 1 (RS485 Master) with Baud Rate, Par, Stop Bit and Timeout **(Pict.6.19-B)**

For RS485 Master is available the possibility to search for nodes over RS485. The searching will be performed at the communication settings configured for the RS485 Master.
The form of search can be recalled clicking the button "Scan" **(Pict.6.19-C).** For more information refer to section "Scan Network".

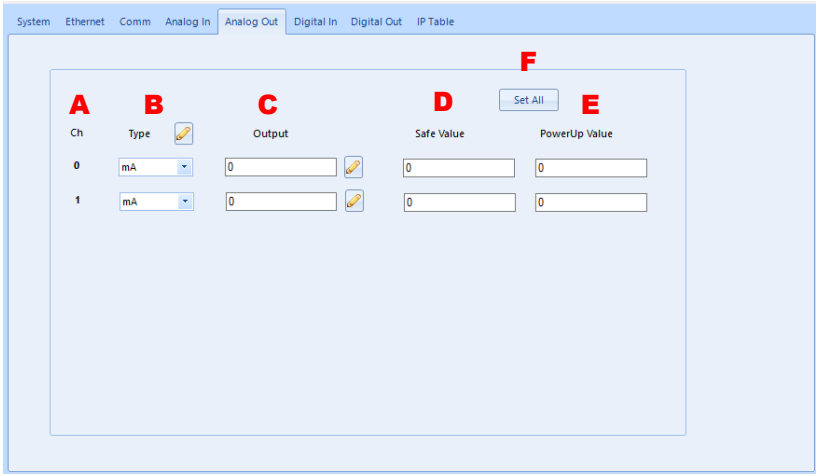


Pict. 6.20

Analog In

The Analog In Tab contains the configuration data of the analog input of the device. So in function of the selected device there is a different number of analog inputs with different settings.
In general there are:
- the number of the channel **(Pict.6.20-A)**
- the type of input for the channel and the relative button with pencil to set the input selected **(Pict.6.20-B)**
- the enable of the channel **(Pict.6.20-C)**
- the measured value of the input channel **(Pict.6.20-D)**

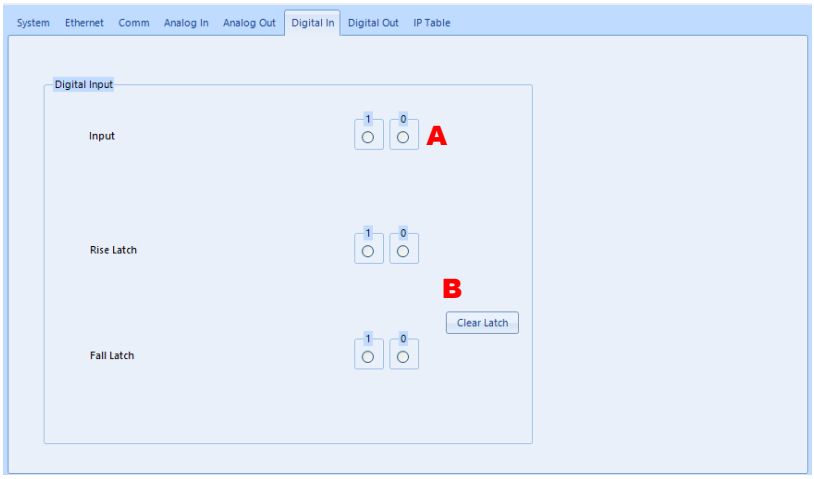
For the devices that foresee the following parameters there will be indication also for:
- the break status of the channel
- the sync value of the channel with the button to save the values



Pict. 6.21

Analog Out

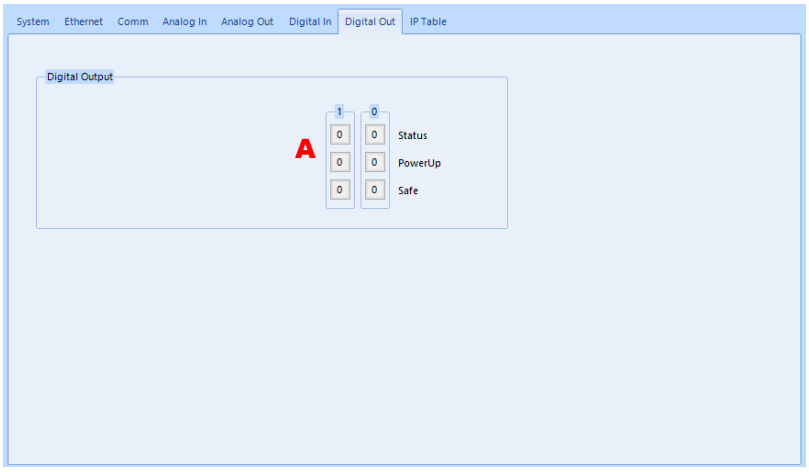
The Analog Out Tab contains the configuration data of the analog output of the device. In function of the selected device there is a different number of analog outputs with different settings.
In general there are:
- the number of the channel **(Pict.6.21-A)**
- the type of output for the channel and the relative button with pencil to set the output selected **(Pict.6.21-B)**
- the value of the output channel **(Pict.6.21-C)**
- the Safe value of the channel when the device is in WatchDog **(Pict.6.21-D)**
- the PowerUp value of the channel when the device is restarted **(Pict.6.21-E)**
- the button to save Safe Value and PowerUp Value **(Pict.6.21-F)**



Pict. 6.22

Digital In

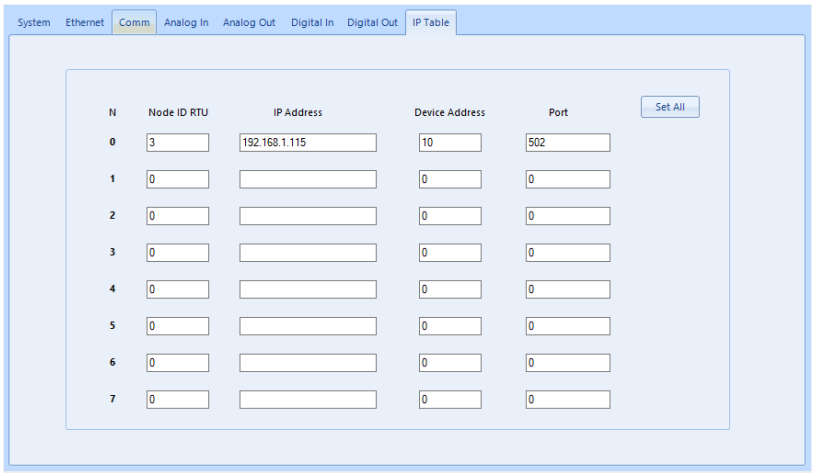
The Digital In Tab contains the states of digital inputs. In function of the device selected there are different numbers of inputs **(Pict.6.22-A).** If the digital input has logic state high the relative indicator is highlighted in green. There is a button to clear at the same time the states of all rising Latch and falling Latch of digital inputs.**(Pict.6.22-B)**.



Pict. 6.23

Digital Out

The Digital Out Tab contains the states of digital outputs and the PowerUp and Safe values of the relative outputs. In function of the selected device there are different numbers of outputs **(Pict.6.23-A).** It is possible to visualize and set the different states of the outputs clicking the proper buttons. When the output has logic state high the button is red.



Pict. 6.24

IP Table

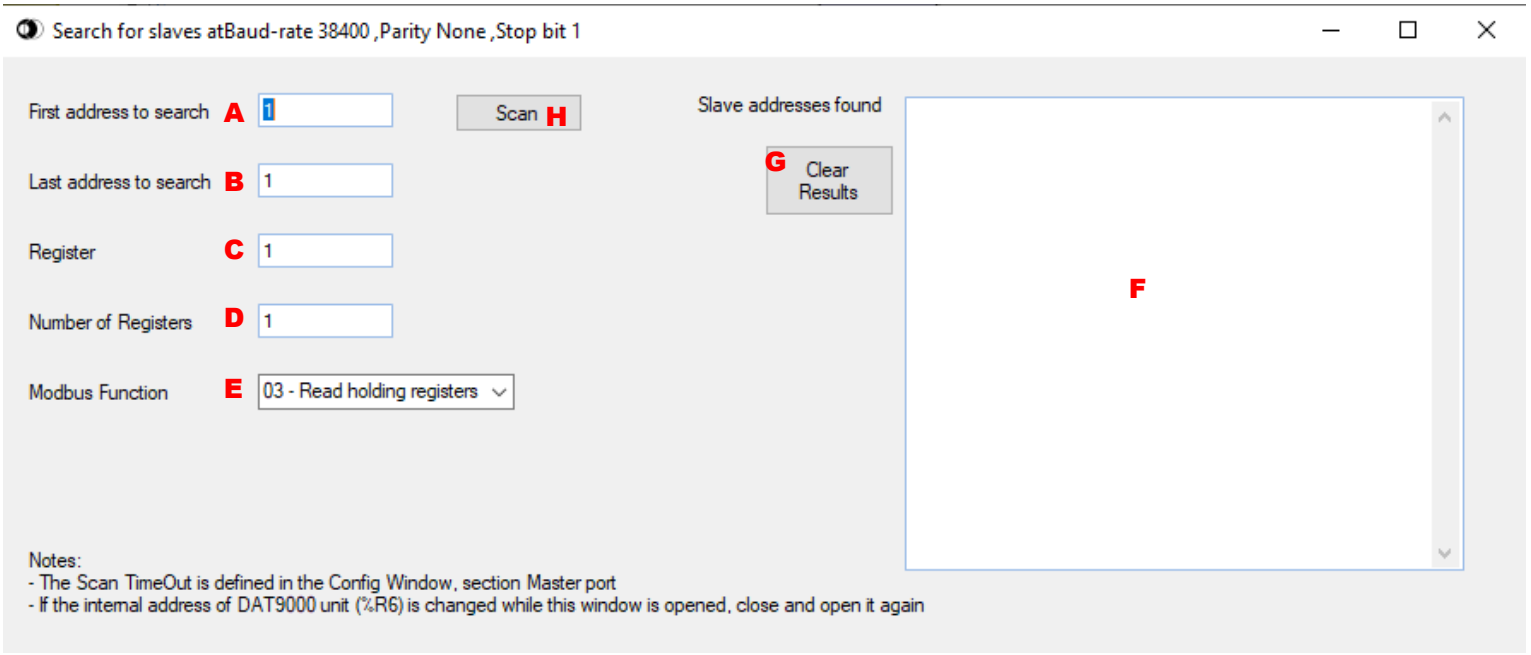
The IP table contains the IP addresses of the Slave / Server devices that can be queried on the Ethernet interface with the Modbus TCP / IP protocol **(Pict. 6.24).** A maximum of 8 IP addresses or a maximum of 8 server devices can be entered.

For more detailed information on the use and insertion of data in the IP table, refer to **paragraph 5.2** of this manual.

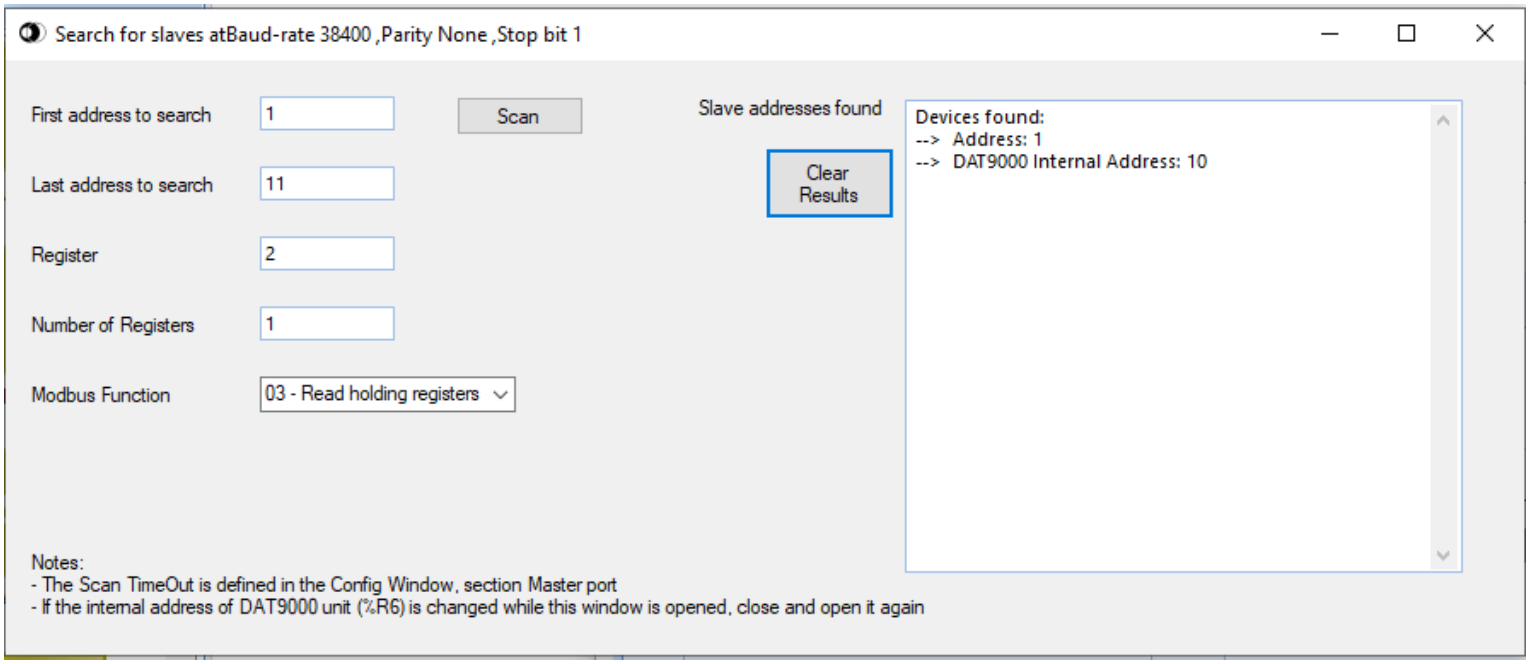
6.9 – SCAN RS485 NETWORK

This window allows the user to search for the devices connected to the RS485 Master port.
The communication parameters such as baud-rate, parity and stop bit are those set for the Master port.
The search for devices is performed sending a query by Modbus function 03 or 04 to a range of slave addresses included between the first and the last address to search.
To be created, both the function foresee a starting register and the number of register to read.
The query can be created by the user using the elements of this window.
If a slave doesn't respond to the command the time taken to move to the next address is the Timeout set for the Master port.
When a slave is found, its address will be listed in the right part of the window **(Pict.6.25-F)**.
The result will always show the Internal Modbus address of the Controller.
The window is composed of the following elements
First address to search (**Pict.6.25-A)**. It is the first address of the range to search for.
Last address to search **(Pict.6.25-B)**. It is the final address of the range to search for.
Modbus function **(Pict.6.25-E)** is the command identifier that will be sent to the serial line.
Register **(Pict.6.25-C)** is the first register of the reading command.
Number of Registers **(Pict.6.25-D)** is the number of registers of the reading command.
To clear the list of results click "Clear Results"**(Pict.6.25-G)**
To perform the search for devices click "Scan"**(Pict.6.25-H)**

Pict. 6.25



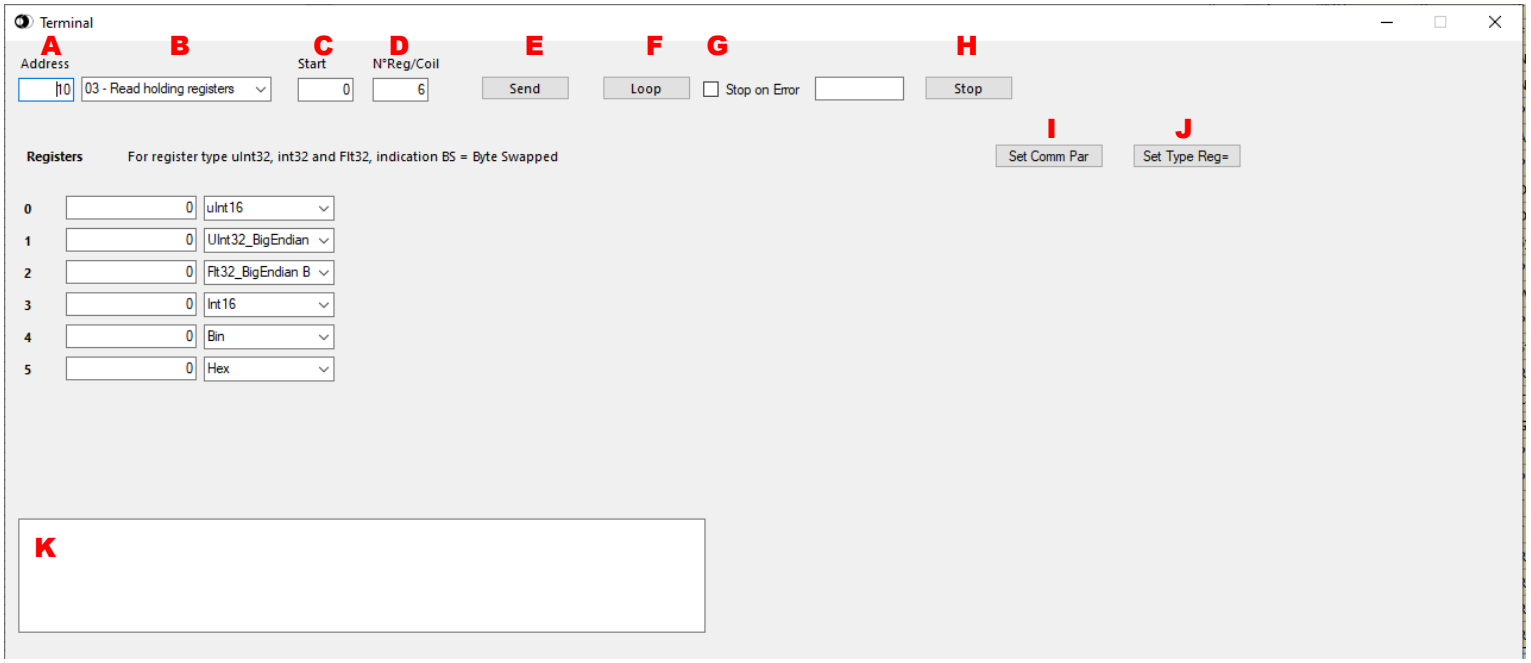
Example of search



6.10 – MODBUS TERMINAL

This window allows to create Modbus commands in Modbus RTU. If the command is sent via serial line, it will be implemented the CRC error control. The Modbus functions listed are those supported by DATEXEL devices. It is possible to poll slave manufactured by other companies too.
The supported functions are:
01 - Read coil status
02 - Read input status
03 - Read holding registers
04 - Read input register
05 - Write single coil
06 - Write single register
15 - Write multiple coil
16 - Write multiple registers

Pict. 6.26



"**Address"** (**Pict.6.26-A)**: insert in this field the address of the slave device to which send commands.
"Function" (**Pict.6.26-B)**: allows to choose the Modbus function between those listed above.
"Start" (**Pict.6.26-C)**: allows to edit the starting register of the Modbus function.
"N°Reg/Coil" (**Pict.6.26-D)**: allows to edit the parameter number of registers/coil of the Modbus function.
"Send" (**Pict.6.26-E):** sends a command to the device.
"Loop" (**Pict.6.26-F):** starts a communication cycle. To stop it click the **"Stop"**(**Pict.6.26-H)** button.
"Stop on Error" (**Pict.6.26-G):** if this flag is enabled, if a polling in loop has been started, the software stops to poll if an error occurs.
The response of the device will be printed in the textbox (**Pict.6.26-K)** in hexadecimal format**.**
For functions 03 and 04 after the starting register and the number of registers have been set, it is necessary to use the button "Set Comm Par" in order to edit the data type of the registers read. (**Pict.6.26-I)**
By the combo-box placed beside each register, the format data-type can be chosen from the followings:

- Binary "Bin"
- Hexadecimal "Hex"
- Unsigned Integer 16 bit "uInt16"
- Signed Integer 16 bit "Int16"
- Unsigned Integer 32 bit in Big Endian byte order "UInt32_BigEndian"
- Unsigned Integer 32 bit in Little Endian byte order "UInt32_LittleEndian"
- Unsigned Integer 32 bit in Big Endian byte swapped order "UInt32_BigEndian BS"
- Unsigned Integer 32 bit in Little Endian byte swapped order "UInt32_LittleEndian BS"
- Signed Integer 32 bit in Big Endian byte order "Int32_BigEndian"
- Signed Integer 32 bit in Little Endian byte order "Int32_LittleEndian"
- Signed Integer 32 bit in Big Endian byte swapped order "Int32_BigEndian BS"
- Signed Integer 32 bit in Little Endian byte swapped order "Int32_LittleEndian BS"
- Floating Point 32 bit in Big Endian byte order "Flt32_BigEndian"
- Floating Point 32 bit in Little Endian byte order "Flt32_LittleEndian"
- Floating Point 32 bit in Big Endian byte swapped order "Flt32_BigEndian BS"
- Floating Point 32 bit in Little Endian byte swapped order Flt32_LittleEndian BS"
- ASCII "ASCII"

At the next reading command, the value will be updated in the textbox near the registers

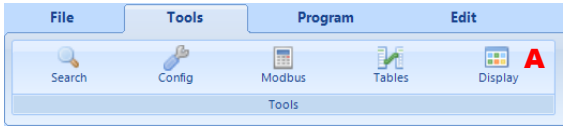
To set all of the registers with the same data type of the first click the button "Set Type Reg="

Note: for other Function codes, the parameters must be set referring to the Modbus standard about to how to set the data for the function code selected.

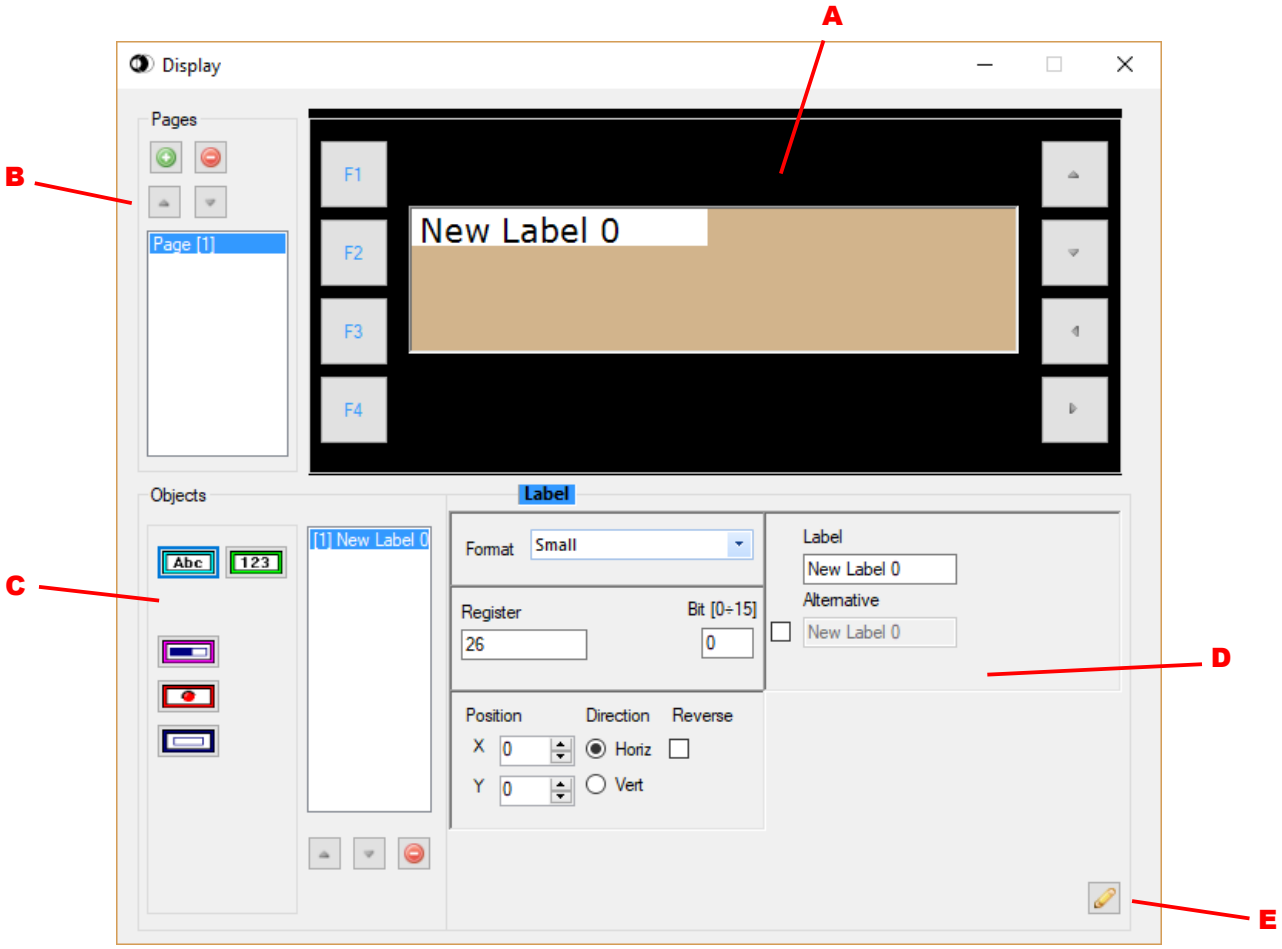
6.11 – GRAPHIC OBJECTS EDITOR AND DISPLAY WINDOW

6.11A: WINDOW STRUCTURE

This menu of configuration must be used only to program the graphic display.
To open the *"Display"* window select in the Menu Bar "*Tools → Display*" **(Pict.6.31-A)**
The window is composed of:
- a graphic preview of the display **(Pict.6.32-A)** that allows the user to visualize the position and the structure of the objects inserted;
- a group of operational buttons to work on the structure of the graphic pages **(Pict.6.32-B)**
- a group of operational buttons to work on the graphic objects **(Pict.6.32-C)**
- a window to show the properties of the single graphic object **(Pict.6.32-D)**
- a button to save the properties of the selected object **(Pict.6.32-E)**



Pict. 6.31



Pict. 6.32

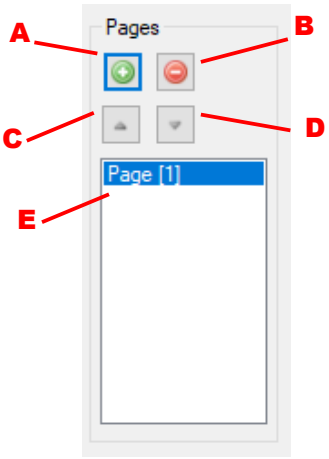
6.11B: GRAPHIC OBJECTS

By the *"Display"* window it is possible to create and set the visualization of the following objects:

-*page*: area of visualization of the objects.
-*label*: strings of 11 alphanumerical characters length; it is possible to visualize the characters that belong to the standard ASCII table (not extended). Note: for the character "°", use the character "^" (example: °C → ^C)
-*numeric value of a register:* visualization, in a format defined by the user, of the value of an Internal register.
-*progress bar:* progress bar with value of filling proportional to the value of an internal register;
-*rectangle:* creation and visualization of simple geometric shapes (rectangle);
-*picture:* creation and visualization of pictures where the filling condition depends of the logic state of the bit of a single register.

6.11C: CREATION OF THE DISPLAY WINDOW

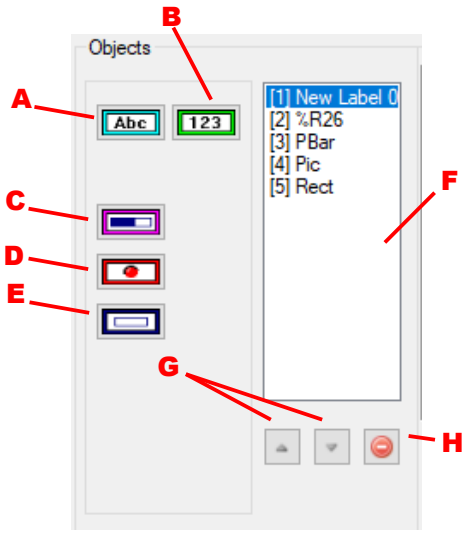
Work on the graphic pages (Reference Pict.6.33)
To insert a graphic page click the button "*Insert page*" **(Pict.6.33A)**; to the page created will be assigned a progressive identification number and the page will appear in the list of the existing pages. In the list the page selected will be highlighted in blue **(Pict.6.33E).**In case of creation of several pages it is possible to use the scrolling buttons "*Page Up*" **(Pict.6.33C)** and "*PageDown*" **(Pict.6.33D)** to move into the list and select the desired page; the page preview will be updated in function of the page selected. To delete a page click the button "*Delete page*" **(Fig.6.33B).**



Pict. 6.33

Work on the graphic object (Reference Pict.6.34)
- Label.
To insert the object "Label", click the button *"Label"* **(Pict.6.34A).** Inside the "*List Object*" **(Pict.6.34F)** the text "New Labe 0" will appear.
- Numeric value of a register.
To insert the object "Numeric value", click the button *"Number"* **(Pict.6.34B).** Inside the *"List Object "* **(Pict.6.34F)** the text "%R26" will appear.
- Progress bar.
To insert the object "Progress bar", click the button *"Progress bar"* **(Pict.6.34C).** Inside the *"List Object "* **(Pict.6.34F)** the text "Pbar" will appear.
- Picture.
To insert the object "Picture", click the button *"Picture"* **(Pict.6.34D).** Inside the *"List Object "* **(Pict.6.34F)** the text "Pic" will appear.
- Rectangle.
To insert the object "Rectangle", click on the button *"Rectangle"* **(Pict.6.34E).** Inside the *"List Object "* **(Pict.6.34F)** the text "Rect" will appear.

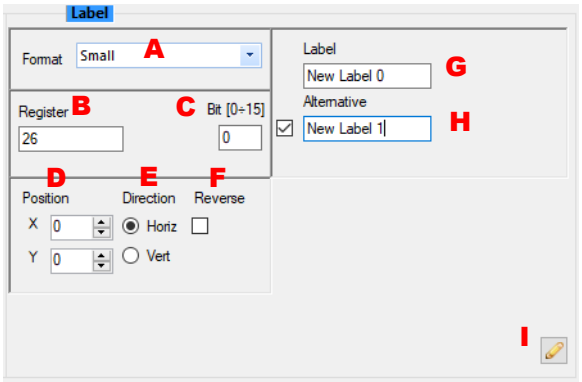
The object inserted will be highlighted in blue inside the *"List Object ".*
Use the scrolling buttons *"Scroll Up"* and *"Scroll Down"* **(Pict.6.34G)** to move inside the *"List Object "* and to select the desired object.
To delete an object, select the object inside the "*List Object*" and click the button "*Delete Object*" **(Pict.6.34H).**



Pict. 6.34

Properties of the graphic objects Window (Reference Pict.6.35)
Inside the Display window, each object is defined by specific properties.
The confirm button with the pencil **(Pict.6.35I)** saves the modifies applied to the selected graphic object and updates the preview of the display. In case of modify of an object and missed click of the confirm button, the modify won't be saved.

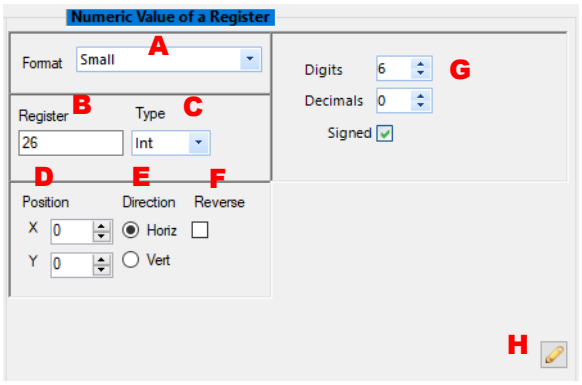
Label.
Format **(Pict.6.35A):**Allows to set the size of visualization of the character in two sizes:: Small(6x8p.) and Medium(12x16p.).
Register **(Pict.6.35B):** Allows to set the Internal Register that contains the bit to which the visualization of the dynamic text is connected to.
Bit [0÷15] **(Pict.6.35C):** Allows to set the bit of the Internal register to which the visualization of the dynamic text is connected to.
Position **(Pict.6.35D):** Defines the coordinates, expressed as pixel, of the dynamic text's position inside the graphic pages (X→horizontal: 0 up to 131; Y→vertical: 0 up to 31).
Direction **(Pict.6.35E):** Defines the orientation (horizontal or vertical) of the dynamic text inside the graphic page.
Reverse **(Pict.6.35F):** Defines the visualization of the object as direct or reverse referred to the background of the graphic page.
Label **(Pict.6.35G):** Allows to insert the string of alphanumerical characters (11 max.) that will be visualized if the logic state of the reference bit is 0.
Alternative **(Pict.6.35H):** Allows to insert the string of alphanumerical characters (11 max.) that will be visualized if the logic state of the reference bit is 1. If the flag alternative is not enabled, this parameter has not effect and is equal to Label.
Button *"Confirm"* **(Pict.6.35I).**



Pict. 6.35

Numeric value of a register.

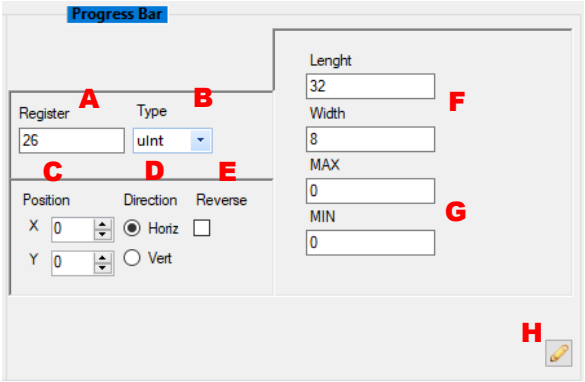
Format **(Pict.6.36A):** Allows to set the size of visualization of the object in three sizes: Small(6x8p.), Medium(12x16p.) and Large (24x32p.).
Register **(Pict.6.36B):** Allows to set the Internal Register of which the value is visualized .
Type **(Pict.6.36C):** Allows to set the format of the Internal Register's value to visualize.
Position **(Pict.6.36D):** Defines the coordinates, expressed as pixel, of the object's position inside the graphic page (X→horizontal: 0 up to 131; Y→vertical: 0 up to 31).
Direction **(Pict.6.36E):** Defines the orientation (horizontal or vertical) of the object inside the graphic page.
Reverse **(Pict.6.36F):** Defines the visualization of the object as direct or reverse referred to the background of the graphic page.
Digits, Decimals, Signed **(Pict.6.36G):** Defines the format of visualization of the value: digits is the number of integers before the commas, decimals is the number of decimals after the commas, signed is checked if the sign is used.
Button *"Confirm"* **(Pict.6.36H).**



Pict. 6.36

Progress bar.

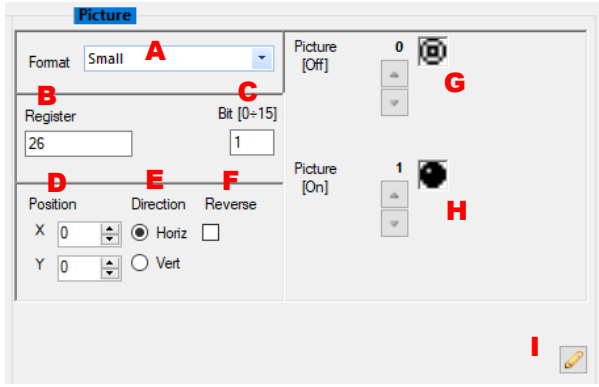
Register **(Pict.6.37A):** Allows to set the Internal Register to which the filling ratio of the bar is connected.
Type **(Pict.6.37B):** Allows to set the format of the Internal Register's value
Position **(Pict.6.37C):** Defines the coordinates, expressed as pixel, of the object's position inside the graphic page (X→horizontal: 0 up to 131; Y→vertical: 0 up to 31).
Direction **(Pict.6.37D):** Defines the orientation (horizontal or vertical) of the object inside the graphic page.
Reverse **(Pict.6.37E):** Defines the visualization of the object as direct or reverse referred to the background of the graphic page.
Dimensions **(Pict.6.37F):** Defines the dimensions (length x width), expressed as pixel, of the object.
Filling constants **(Pict.6.37G):** Defines the values of the Internal Register selected to which the events of Start (MIN) and Stop (MAX) of filling of the bar are connected.
Button *"Confirm"* **(Pict.6.37H).**



Pict. 6.37

Picture.

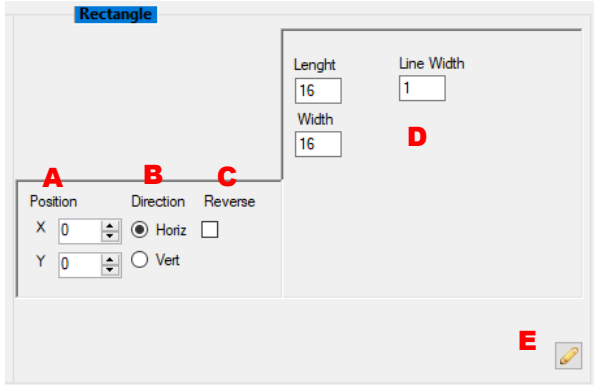
Format **(Pict.6.38A):** Allows to set the size of visualization of the object in three sizes: Small(6x8p.), Medium(12x16p.) and Large (24x32p.).
Register **(Pict.6.38B):** Allows to set the Internal Register to which the visualization of the picture is connected.
Bit **(Pict.6.38C):** Allows to set the bit of the Internal register to which the visualization of the picture is connected to.
Position **(Pict.6.38D):** Defines the coordinates, expressed as pixel, of the object's position inside the graphic pages (X→horizontal: 0 up to 131; Y→vertical: 0 up to 31).
Direction **(Pict.6.38E):** Defines the orientation (horizontal or vertical) of the object inside the graphic page.
Reverse **(Pict.6.38F):** Defines the visualization of the object as direct or reverse referred to the background of the graphic page.
Picture [Off] **(Pict.6.38G):** Indicates how the picture will be visualized if the logic state of the reference bit is 0.
Picture [On] **(Pict.6.38H):** Indicates how the picture will be visualized if the logic state of the reference bit is 1.
Button *"Confirm"* **(Pict.6.38I).**



Pict. 6.38

Rectangle.

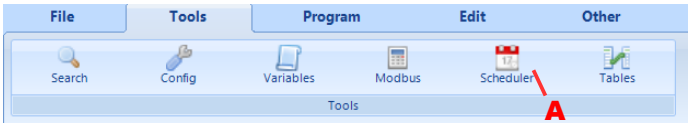
Position **(Pict.6.39A):** Defines the coordinates, expressed as pixel, of the object's position inside the graphic pages (X→horizontal: 0 up to 131; Y→vertical: 0 up to 31).
Direction **(Pict.6.39B):** Defines the orientation (horizontal or vertical) of the object inside the graphic page.
Reverse **(Pict.6.39C):** Defines the visualization of the object as direct or reverse referred to the background of the graphic page.
Dimensions **(Pict.6.39D):** Defines the dimensions (length, width and line thickness), expressed as pixel, of the object.
Button *"Confirm"* **(Pict.6.39E).**



Pict. 6.39

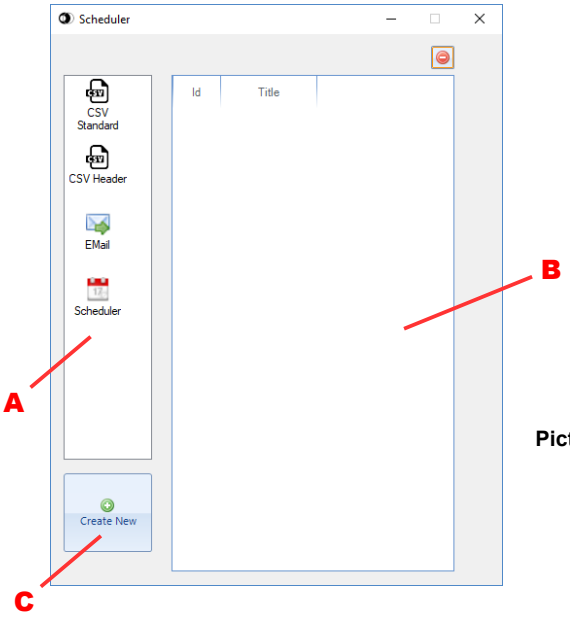
6.12 – SCHEDULER EDITOR

Click the button "*Scheduler*" in the menu bar to open the Scheduler window.**(Pict.6.24-A)**
The Scheduler window **(Pict.6.25)** allows to insert and modify the settings of data recordings, the emails and the synchronized scheduler.



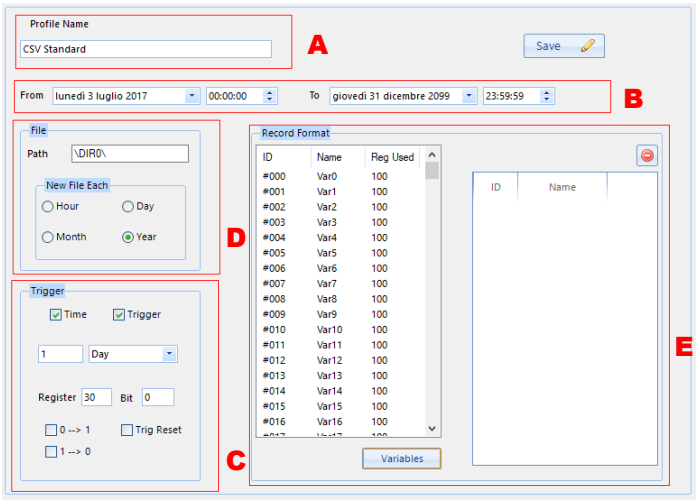
Pict. 6.24

To insert a new element drag it from left list **(Pict.6.25-A)** to right list **(Pict.6.25-B)**.
Click the button "*Create New*" (Pict.6.17-C) to open a wizard window that guides the user to create element without manual insertion.
The available elements are:
- *CSV Standard* : allows to execute a variables recording. The data are stored on USB pen in .CSV format; all files saved in the USB are accessible from the page "Download" of the Web pages.
- *CSV Header* : allows to execute a texts recording. The data are stored on USB pen in .CSV format; all files saved in the USB are accessible from the page "Download" of the Web pages.
- *EMail* : allows the device to send email. Server, Receivers, Body of email are settable from Web pages.
- *Scheduler* : allows to set a mask of bits of a register in order to control the logging events or the emails.



Pict. 6.25

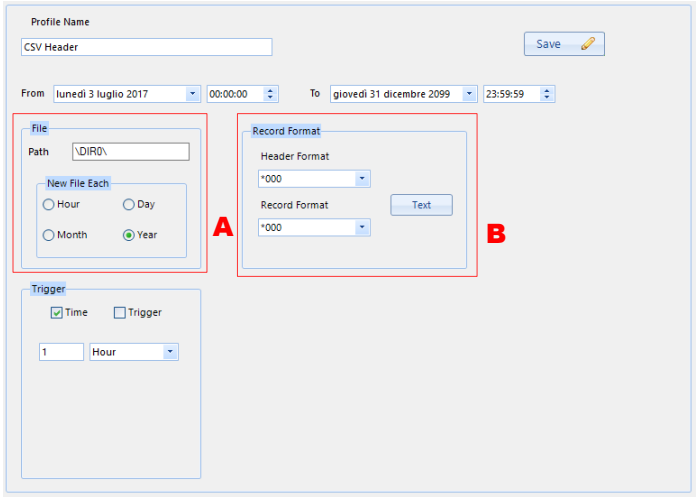
For each element it is possible to set:
- Profile Name **(Pict.6.26-A)**: name used to identify the element.
- Date and time of start and end **(Pict.6.26-B)**: indicates the period of validity of the element that works only between these two dates.
- Event that triggers the action **(Pict.6.26-C)**:
by time, the action is activated at regular intervals of time based on the time set;
by trigger, the action is activated at each variation of the bit's state of a register (rising edge or falling edge).



Pict. 6.26

For CSV Standard, in addition to the common parameters, it is possible to set:
- Destination directory / Frequency to create file **(Pict.6.26-D)**: the destination directory indicates the directory where all the files of the current task are created. If the directory written does not exist, it will be created automatically. The parameter about the time set (hour, day, month, year) indicates the frequency for which a new file containing the data recordings is created and opened for writing. Each time that the period of time set expires the file previously created is closed. The file name is assigned in function of period of time chosen.
- List of variables to save **(Pict.6.26-E)**: dragging the variables from the list on the left to the list on the right the fields composing the records will be created(columns of CSV).
To modify the variables click the button "*Variables*" (refer to section "Insertion of Variables, Strings, Texts").

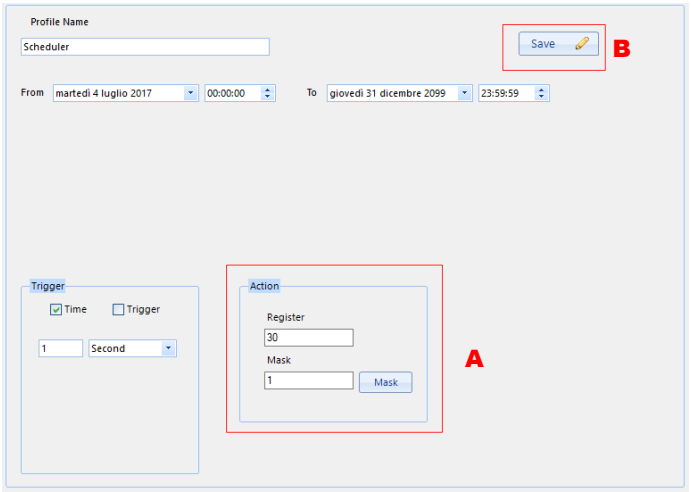
For CSV Header, in addition to the common parameters, it is possible to set:
- Destination directory / Frequency to create file **(Pict.6.27-A)**: the destination directory indicates the directory where all the files of the current task are created. If the directory written does not exist, it will be created automatically. The parameter about the time set (hour, day, month, year) indicates the frequency for which a new file containing the data recordings is created and opened for writing. Each time that the period of time set expires the file previously created is closed. The file name is assigned in function of period of time chosen.
- Record format **(Pict.6.27-B)**: Header Format is the text used for the header of the CSV file, Record Format is the text used for each record of the CSV file.
To modify the texts click the button "*Text*" (refer to section "Insertion of Variables, Strings, Texts").



Pict. 6.27

For the Emails it is possible to set only the common parameters: Profile Name, Date and time of start and end, event that triggers the Email.
The parameters of the Email message are settable only from Web pages.

For the synchronized scheduler (Scheduler), in addition to the common parameters, it is possible to set the register in which to set the bits in function of a Mask. **(Pict.6.28-A)**



Pict. 6.28

After insertion or modifying the parameters of an element, click the button "*Save*" to save it **(Pict.6.28-B)**.

IMPORTANT:
The device does NOT manage the DST.

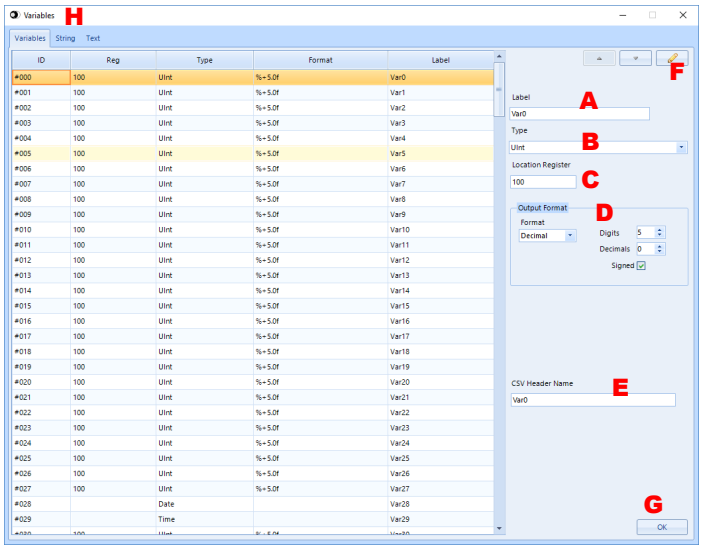
6.13 – INSERTION OF VARIABLES, STRINGS, TEXTS

The Variables window **(Pict.6.29)** allows to modify Variables, Strings and Texts to use in the CSV Data Recordings.

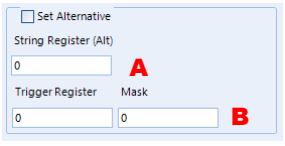
For each variable it is possible to set:
Label **(Pict.6.29-A)**: name of the variable (used only in the program as identifier)
Type **(Pict.6.29-B)**: type of variable. It can be UInt, Int, Ulong, Long, Float, String(Ram), String(Eprom), Date (variable to record the date), Time (variable to record the time)
Location Register **(Pict.6.29-C):** ram register of the device which the variable refers to. For the types Date and Time this field does not exist because the device automatically gets it.
Output Format **(Pict.6.29-D)**: format of the variable saved in the .CSV file. It can be set as Decimal, Exponential, or Hexadecimal, with sign, and the number of digits and decimals. For the types Date and Time this field does not exist.
Set Alternative **(Pict.6.30):** available only for types String(Ram) and String(Eprom). If selected, it allows to set a register that contains an alternative string**(Pict.6.30-A)** used when the bits of mask of a trigger register are 1 **(Pict.6.30-B)**
CSV Header Name **(Pict.6.29-E):** name of the header of the variable in the CSV file.

To confirm and save the parameters of a modified variable click the button with pencil **(Pict.6.29-F)** or the button "OK" that closes the window**(Pict.6.29-G)**.

For strings and texts in their tabs **(Pict.6.29-H)** it is possible to modify the content inserting a maximum of 32 characters for the strings and a maximum of 512 characters for the texts. To confirm click the relative button with pencil.



Pict. 6.29



Pict. 6.30

7.1 – ETHERNET CONNECTION

On the Ethernet side, the Controller works like a Server, therefore for the connection to the LAN network it is necessary to follow the standards for the Ethernet connections. Hereafter some practical tips to connect the Controller are reported.

To connect the Controller directly to a PC, use a crossover cable. To connect the Controller to an Hub, Switch or Router, use a direct cable.

Due to their settings, it could happen that some Firewalls won't allow the communication with the Controller; this kind of problem could happen particularly in phase of Search: in case of communication problems it is suggested, if possible, to disable eventual active Firewalls on the Client PC or Router.

If the DHCP service (Dynamic Host Communication Protocol) is not in use, be sure that the IP, the Subnet Mask and the Gateway address of the Controller will be compatible with the settings of the LAN network which the Controller is connected to.

8.1 – IMPORTANT MESSAGES IN THE LOG PANEL OR IN POP-UP WINDOW

MESSAGE	POSSIBLE CAUSES	POSSIBLE SOLUTIONS
"Connected" ***"... - Connected"***	- The device is just connected using the correct parameters	
"Disconnected" ***"... - Disconnected"***	-The device previously connected has just been disconnected or it is not possible to communicate with it	If it is not possible to communicate with the device verify the physical connections and the parameters and reconnect to it *(Par 6.1, Par 6.2)*
"Timeout Connection"	- It is not possible to communicate with the device over Ethernet, because it is in Socket Timeout or it has been physical disconnected.	-The device is in Socket Timeout for no actions: reconnect it *(Par 6.1, Par 6.2)* -The device is physically disconnected from the Net: verify the connections and reconnect.
"Read/Write COM Error"	- An error occurs during the communication over Serial Port	- The device is physically disconnected: verify the connections and reconnect.
"Socket Timeout"	- An error occurs during the communication over Ethernet. The device is not in the Net or the inserted IP doesn't exist.	- The device is not in the Net or the inserted IP doesn't exist: verify the connection parameters, the connections and try to connect.
"Timeout Error"	- An error occurs during the communication over Ethernet. The device has a Modbus Address different from the one selected	-The device has a Modbus Address different from the one selected: verify the connection parameters, the connections and try to connect.
"Com Error"	- An error occurs during the communication over Serial Port. The device is not connected to the PC via Com port or wrong parameters have been inserted.	- The device is not connected to the PC via Com port: verify connections and reconnect. - Wrong parameters are inserted: verify Serial connection parameters (Com port, Baud Rate, Modbus address, etc...), and try to connect.
"Process Validation: Complete"	- The diagram realized doesn't contain structural errors and so it is possible to download it in the Eprom of the device	
"Process Validation: Error"	- The diagram realized contains structural errors. It is not possible to download it in the Eprom of the device	- Check the diagram looking for the errors
"Download Complete"	- Download Eprom successful.	
"No Device Connected".	- No connection to a device has been executed or the device have been previously disconnected from the program	- If no connection to a device has been executed and the program is in Offline mode, connect to a device. - If the device have been previously disconnected or a communication error occurs, verify communication parameters, the connections and try to connect.

8.2 – POSSIBLE CAUSES OF FAULT

EVENT	POSSIBLE CAUSES	POSSIBLE SOLUTIONS
It is not possible to power-on the Controller.	-The Controller is not correctly powered. -The value of the power supply value is lower than the specifications limits.	-Refer to the data-sheet of the Controller in use and verify the relative Technical Specifications.
There is not communication between the Host PC and the Controller.	-Ethernet port not correctly connected. -Modbus Slave port not correctly Connected or set. -Eventual interface between PC and Controller not correctly connected. -Wrong communication parameters.	-Refer to the section 7.1 -Refer to the data-sheets of the Controller and the Interface device in use. -Refer to the section 8.1
There is not communication between the Controller and one or more Modbus slave devices.	-Modbus Master port not correctly connected. -The slave device is not correctly powered -The slave device is not correctly connected on the RS-485 serial line. -Wrong communication parameters. -The Modbus addresses of the slave devices connected are not included in the range set in the System Register %R17 (Gateway Mask) .	-Refer to the section 8.1 -Refer to the data-sheets of the Controller and the Slave devices in use. -Slave device in INIT condition and Baud-rate of communication different of 9600 bps. -Verify the values of Gateway Mask.
The Program is not correctly executed or it is impossible to execute the Program.	-Wrong communication parameters. -The Controller is in “Debug” modality and in Halt, Stop or Break Point condition. -Wrong data-format of the Registers. -Wrong parameters of the Function Block. -Parameters of the eventual Slave devices connected not correctly inserted. -The Program has not been downloaded -Controller in INIT modality.	-Refer to the section 8.1 -Set the Controller in “Debug” modality and in “Run” condition or in “Release” modality. -Remove eventual Break Points. -Set the correct data-format of Registers. -Verify the parameters of Function Block (data-format, masks, tables, etc..). -Control the configuration of the Slave devices (type of input and output, etc..) -Download the Program. -Control if the INIT modality is active.
The configuration of the Controller is unknown.	-	-Set the Controller in “INIT” modality; the parameters of configuration of the Controller will be forced to the default values listed in section 6.6 .
The Controller is connected in “INIT” modality but is not executed (where foresee the LED “STS” doesn't blink) or there is not communication between the Host PC and the Controller.	-Controller not correctly connected. -Wrong port Baud Rate.	-Connect the terminal INIT to GND. -Switch-off and than power-on the Controller after the connection of the terminal INIT to GND. -Set the Baud-rate of the Slave Port as 9600 bps and Address 10.
The function “Search” doesn't find any Controller.	-There are not Controllers connected. -Controllers not correctly connected. -The Controllers connected by Ethernet port has been set with communication parameters not compatible with the Ethernet interface of the Host PC in use. -On the network there are active Firewall or routers that block the access to the Controller.	-Refer to the data-sheet of the Controller in use and verify the relative Technical Specifications. -Verify the parameters of the Ethernet interface of the Host PC. -Call the System Administrator in order to connect the controller to the network.

EVENT	POSSIBLE CAUSES	POSSIBLE SOLUTIONS
The functions Clock and Calendar (where foresee) don't work correctly.	-Battery low or absent. -Clock and Calendar parameters not correctly set in the proper Registers	-Change or insert the battery. -Control the parameters of the System Registers (refer to the sections 3.3 and 3.4).
The function "Search" doesn't find any Slave device.	-There are not slave devices connected . -The slave devices are not correctly connected. -The Controller which the slave devices are connected to has not been selected. -The Modbus addresses of the slave devices connected are not included in the range set in the System Register %R17 (Gateway Mask) or in the range set in the menu "Search". - The baud-rate of the Slave devices connected is not the same of that set for the Master port of the Controller. -The Timeout values are not correct.	-Refer to the data-sheets of the slave devices in use and verify the relative Technical Specifications. -Verify that the Controller selected is the same which the slave devices are connected to. - Verify the correspondence between settings and Modbus addresses of the slave devices. -Verify the values of Gateway Mask. -Control the baud-rate and delay time of the slave devices connected.
The Web Pages haven't been loaded.	-The IP address written in the address bar of the Internet browser is not the same of Controller's IP address. -The Controllers connected by Ethernet port has been set with communication parameters not compatible with the Ethernet interface of the Host PC in use. -On the network are active Firewall or Routers that block the access to the Controller	-Verify the IP address written in the address bar. -Verify the parameters of the Ethernet interface of the Host PC. -Call the System Administrator in order to connect the controller to the network.
The data saved as constant in the Register table, are not saved when the Controller is switched off.	- The data have been saved in General Purpose Registers instead of Retentive Registers .	-Save the constant values in Retentive Registers.

9.1 – E-MAIL CONFIGURATION

1. Introduction

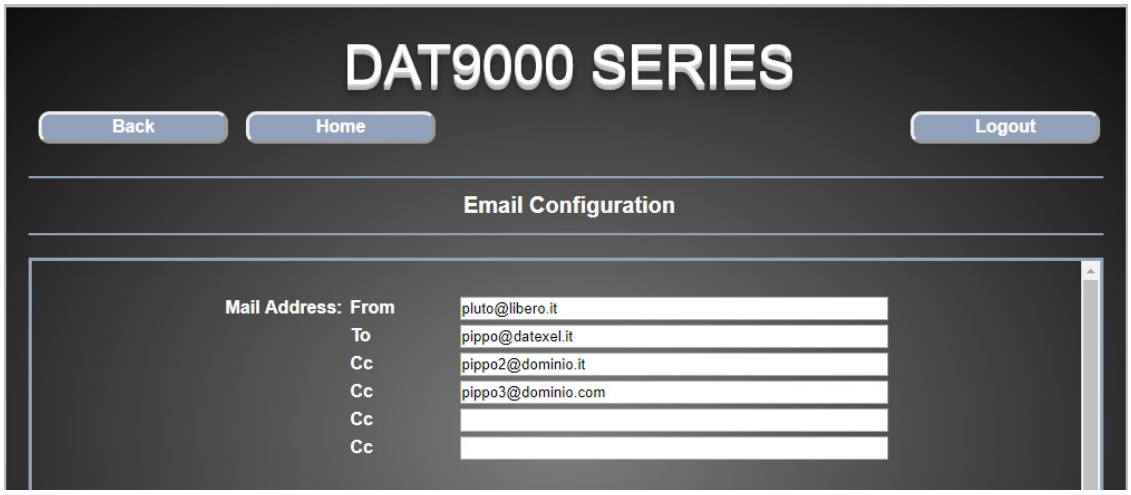
This Application Note shows how to send an e-mail on event or within a specific interval of time. When the e-mail is sent the system uses **port 25 without encryption**. It is possible to send one and only one e-mail. The message is written in the body of the email and can contain a fixed string of characters, numerical variables or the composition of one or more strings together with the variables.
Furthermore, it is also possible to add blocks of text.

2. Configure the E-mail using DAT9000 web server

By a web browser, recall the IP address of the device.
Enter the correct *username and password* so that the DAT9000 Home Page appears.
Click the button **"Email Configuration"** to access to the setup page for e-mail .

- Mail Address

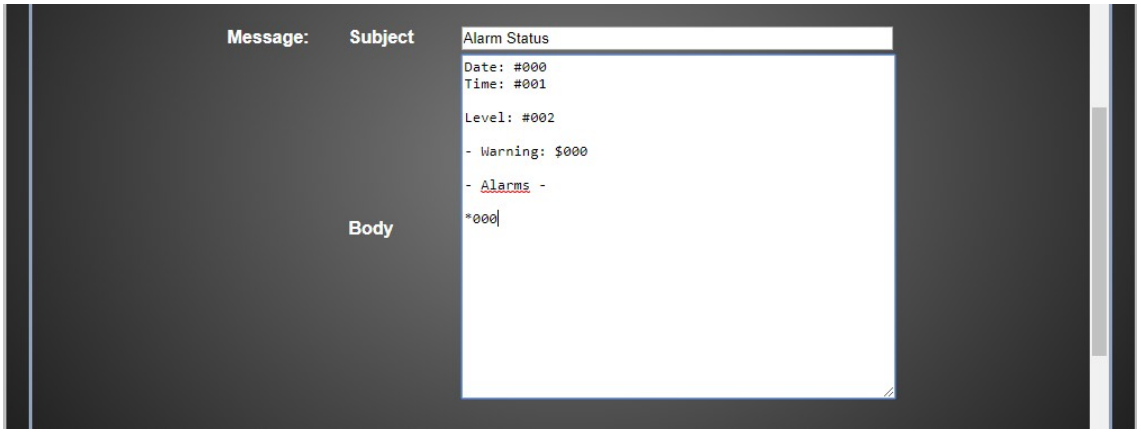
From: e-mail address that uses port 25 without encryption (if not already available, it is necessary to create a compatible account).
To: final destination e-mail address (can be any type of account).
Cc: e-mail address of recipients in "Knowledge Copy" (can be any account).



- Message

Subject: subject of the e-mail.

Body: body or message to send. The variables #XXX, the strings $YYY and the blocks of text *ZZZ are defined through the Dev9k respectively in the **"Variables"**, **"String"** and **"Text"** windows. In the *"Variables"* window an internal register of the DAT9000 is associated with a format variable #XXX. In the "String" window it is possible to associate a string of characters with the format $ YYY. In the *"Text"* window it is possible to associate to the *ZZZ format a text longer than a simple string.



Server Config.
In this area configure the outgoing mail server that uses port 25 without encryption.

SMTP Server: is the outgoing mail server. It depends on the outgoing mail account's domain.
This data can be easily found on the net (in the example, the outgoing mail account is on "*Libero*").

SMTP Port: is the port that is used by the outgoing mail server (in our case, port 25).

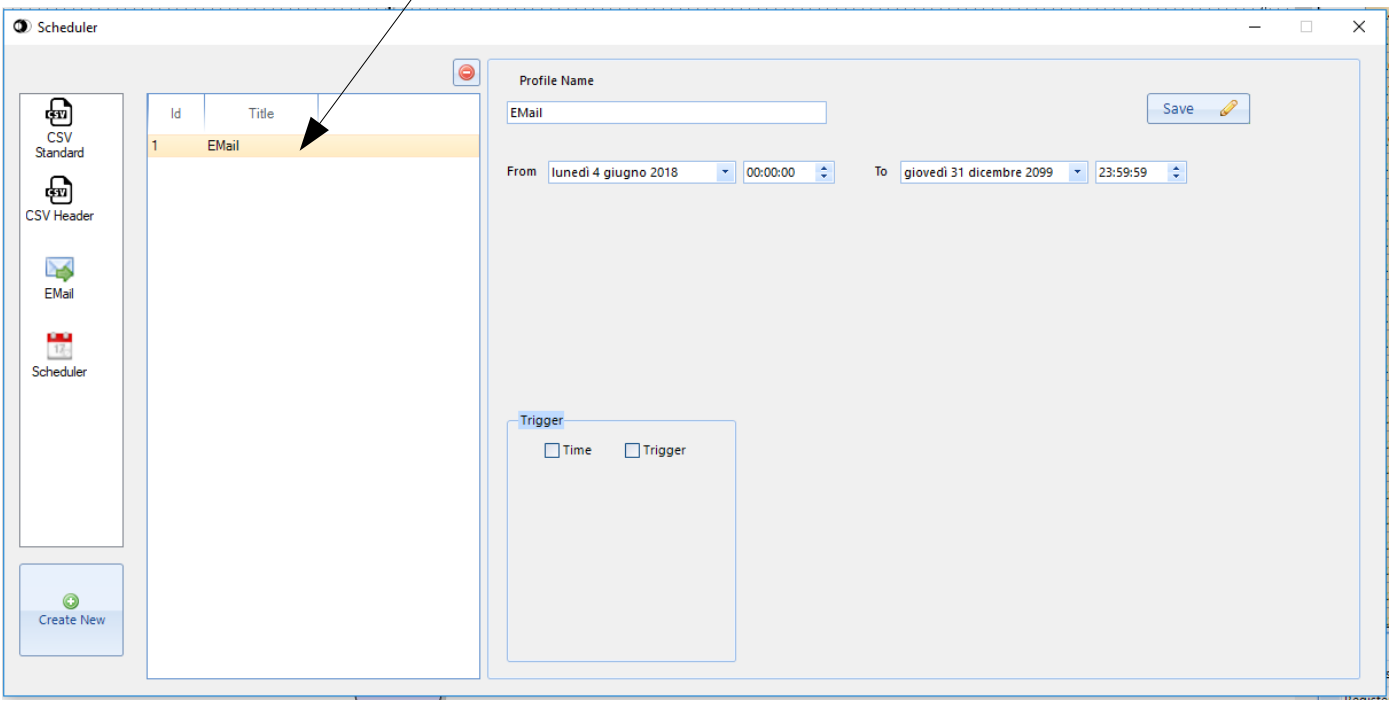
User: outgoing mail account.

Password: password of the outgoing mail account.

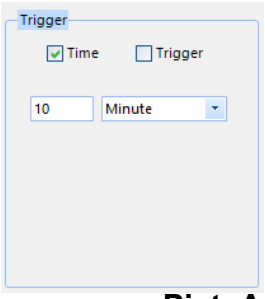
Click on **"Save"** to save the changes of the entire page.

It is possible to send a test email after saving the changes by clicking on the **"Test"** button at the bottom left.

3. Configuration of e-mail timestamp in the Dev9k software
The email configured via the web page can be sent on an event (the event is associated to a bit) or with a well-defined timestamp (every hour, every day every 10 minutes, etc.).
To define this it is necessary to insert the "Mail" field through the Dev9k by accessing the *Tools* → *Scheduler* and clicking first on the symbol to select the "Mail" and after dragging the symbol in the white area closing to the right.
At this point the following window will appear:



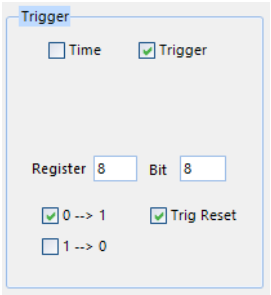
In the *"Trigger"* tag it is possible to set the event or the timestamp to send the e-mail. Therefore there are two possibilities to send an e-mail, *"by Time"* or *"on Event"*.



Pict. A

- By selecting the *"Time"* checkbox, the e-mail will be sent *"by Time"* ie periodically regardless of the status of any bit.
The timestamp is set through the drop-down menu and the relative input field (Pict.. A).
In the example, the e-mail will be sent automatically every 10 minutes.

- By selecting *"Trigger"*, the e-mail is sent in relation to an "Event". That can be related to the change of the status of a given bit, that belongs to a specific register (Pict. B). It is possible to select which status change generates the sending of the e-mail, ie sending on the rising edge of the bit (from 0 → 1) or on the falling edge of the bit (from 1 → 0).
If *"Trig reset"* is flagged when the rising edge, when the e-mail is sent, the bit that has generated the event of sending e-mail is automatically resetted.
In the example, the e-mail will be sent every time the bit 8 of the register 8 changes from 0 to 1 (on the rising edge). Having also selected the *"Trig Reset"* option, bit 8 of register 8 automatically resetted as soon as the e-mail is sent.



Pict. B

At the end of the parameter setting, click **"Save"** to save.

*It is also possible to use the **"Create New"** button which allows the user to enter the e-mail through a wizard.*

4. Setting variables, strings and text blocks in the Dev9k software

Variables #XXX, strings $YYY and text blocks *ZZZ can be inserted in the body of the mail and be defined in the Dev9k software.

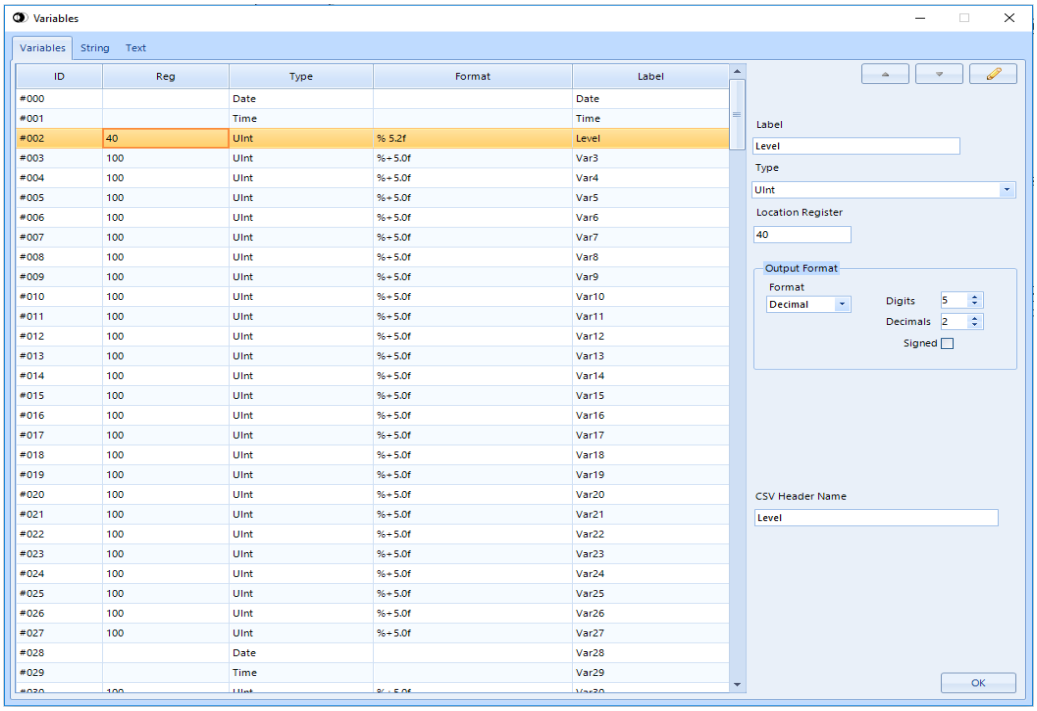
To access the variable definition window, click on *Tools* on the menu and then on *Variables*:

- Variables

Through this window it is possible to associate the ID (#XXX) that will be written in the body of the mail to a predefined variable (Date and Time) or an internal register of the DAT9000. To enter the variables, select the line and modify the parameters in the table. Finally click on  to edit the changes.

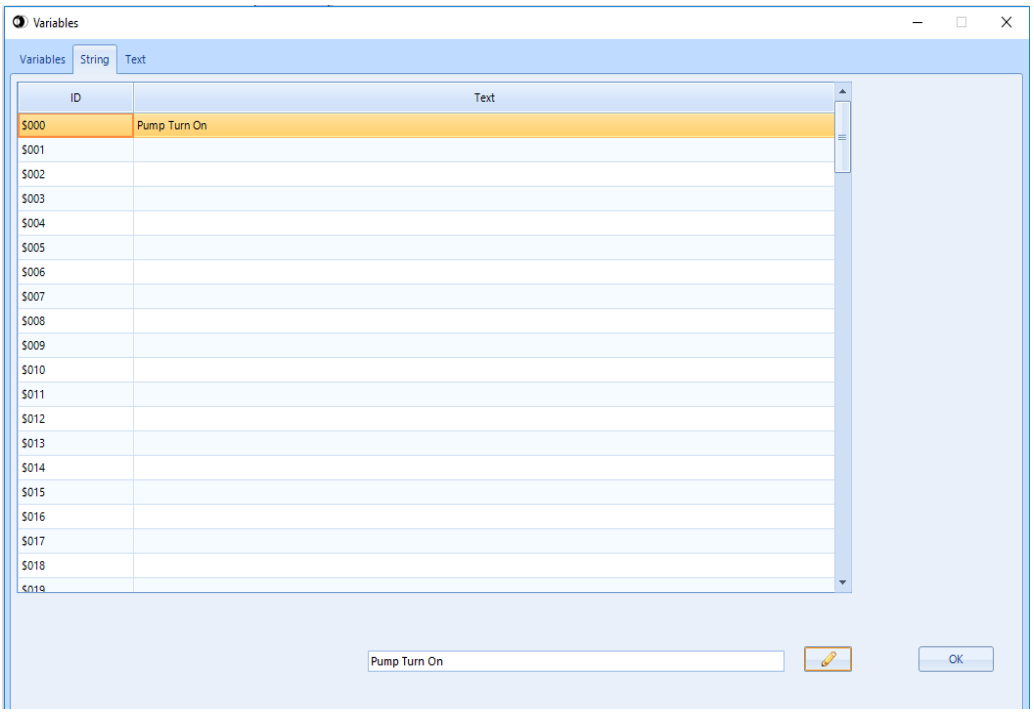
Repeat the operation for all the desired variables.

After entering and editing the variables with  , press **OK** to confirm.



- Strings

Through this window it is possible to associate the ID ($YYY) that will be written in the body of the email to a string of characters that can identify an alarm or any event. To enter the string, select the ID, write in the dedicated space and click on  to edit the changes.
Repeat the operation for all the desired strings.
After entering and editing the strings with  press **OK** to confirm.



- Text Blocks
Through this window it is possible to associate the ID (*ZZZ) that will be written in the body of the email to a text. Enter the desired text in the appropriate area and click on  to edit the changes.

It is possible to insert variables and strings either by writing the ID directly in the body of the text or by choosing from the drop-down menus relative to the IDs.

Repeat the operation for all the desired text blocks.

After entering and editing the text blocks with  press **OK** to confirm.