

Integrated Development Environment to design and debug the applications based on the DAT9000 series controllers.

SUMMARY

1. Introduction

- 1.1 – General Description
- 1.2 – Minimum system requirements
- 1.3 – Procedure of installation
- 1.4 – Dev9K window
- 1.5 - Terminology

2. Getting started

- 2.1 – Start a new Project
- 2.2 – Save and Load a Project
- 2.3 - Insert, Modify and Remove a Function Block
- 2.4 – Compile and Error Check

3. Internal Registers

- 3.1 – Internal Registers
- 3.2 – Type of Data Format
- 3.3 – Mapping Registers
- 3.4 – System Registers overview

4. Functions description

- 4.1 – Function list
- 4.2 – Functions description

5. Insertion of Tables

- 5.1 – Insertion of Tables

6. Controller operations

- 6.1 – Searching of the devices connected
- 6.2 – Connecting to the Controller
- 6.3 - Download the Program
- 6.4 – Debug Mode
- 6.5 – Release Mode
- 6.6 – INIT Mode
- 6.7 - Web Server

7. Tips and suggestions

- 7.1 – Ethernet connection

8. Error messages

- 8.1 – Error messages in the log window and in the status bar
- 8.2 – Error messages inside the popup windows

9. Troubleshooting

- 9.1 – Possible causes of fault

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. With Dev9K it is possible to set the DAT9000 series controllers to execute I/O read and write operations (DAT3000 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® 2000 / NT / ME / XP/ Vista
Available Hard Disk memory	2 MB

1.3 – PROCEDURE OF INSTALLATION

- Close eventual active or background applications.
- Insert the CD-ROM of installation in the driver.
- Wait for the Autorun window opening.

If the Autorun function is disabled, open the CD-ROM and execute the installation file at the path:

<CD Driver>:\doc\download.html

- Click on the Tools button.
- Click on the “download” button in the DAT9000 section.
- Follow the Installation Wizard.

1.4 – DEV9K WINDOW

In the Main Window of Dev9K the following components are shown:

- The Menu bar, the Tool bar and the Status bar (Fig.1.1-A)
- The Project tree (Fig.1.1-B)
- The Log window (Fig.1.1-C)
- The Main Program window (Fig.1.1-D)
- The Registers Table (Fig.1.1-E)

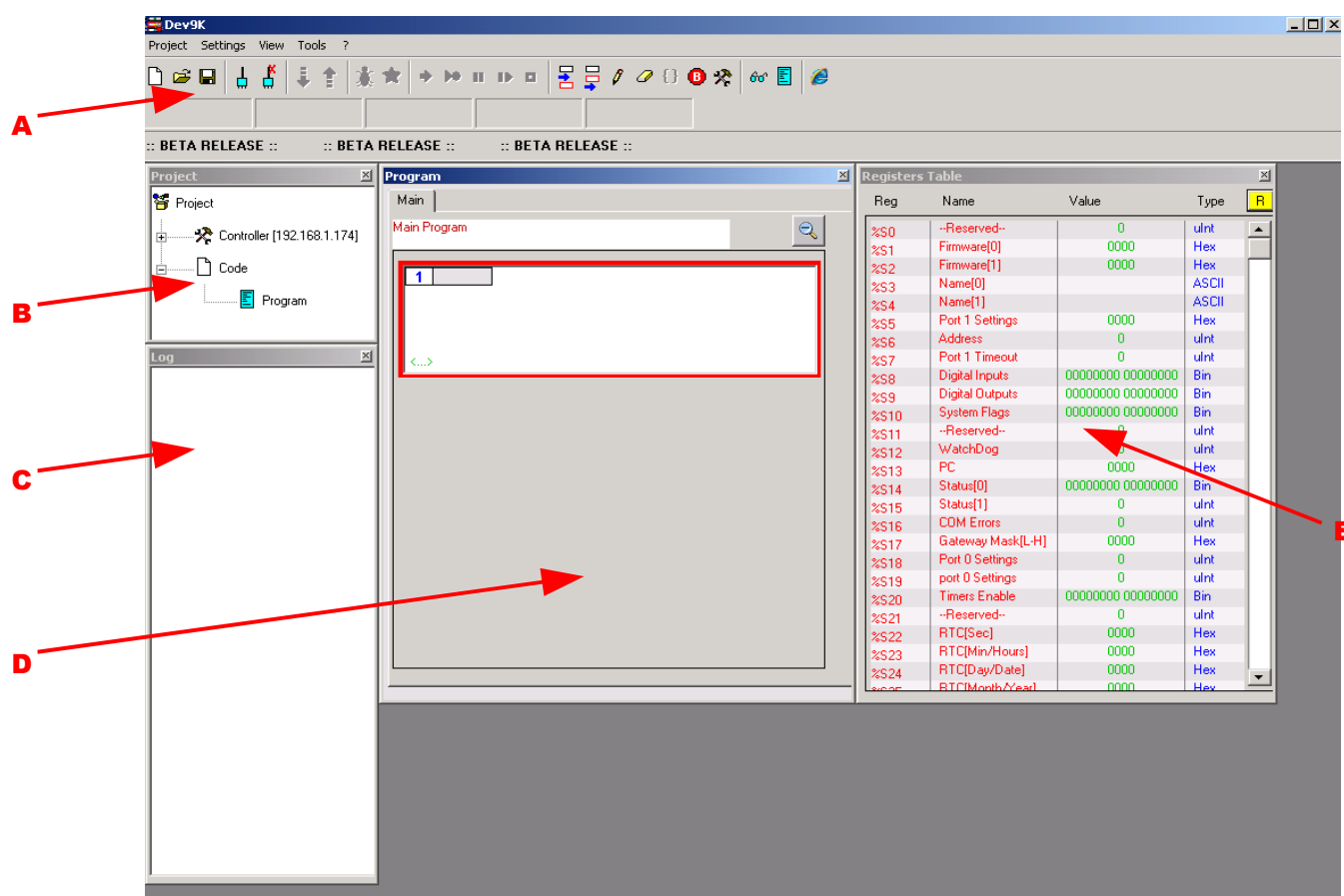


Fig. 1.1

1.5 - TERMINOLOGY

Description of terms and abbreviations used in Dev9K and in this manual.

Controller	Device of the DAT9000 series.
Integrated Development Environment IDE	Instructions set to create, to debug and to test the Program.
Program	List of functions executed from the Controller.
Function Block F.B.	Each block constituting the Main Program. Each Function Block can contain a function.
Function	Logical, mathematical or flow operation executed from the Controller.
Variable	Each parameter contained in a function.
Register	Position of a variable in the volatile memory of Controller.
Retentive Register	Position of a variable in the non-volatile memory of Controller.
Label	String of characters used to identify the name of an object (Function Block, Register, Table,...)

2.1 – START A NEW PROJECT

Click on the “New Project” button (Fig.2.1-A) or select “Project -> New” in the Menu bar.

The following windows will appear:

- “Project”: contains the list of the design properties (Fig.1.1-B).
- “Main Program”: in this window it is possible to insert the Function Blocks to generate the Program. For a new Project this window contains a blank Function Block. (Fig.1.1-D).
- “Registers Table”: contains the list of Controller's Registers (Fig.1.1-E).

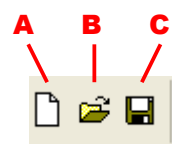
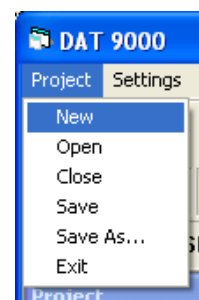


Fig. 2.1



2.2 – SAVE AND LOAD A PROJECT

To save the Project click on the “Save Project” button (Fig.2.1-C) or select “Project -> Save” or “Project -> Save As...” in the Menu bar.

When a Project is saved for the first time, Dev9K requires the file name that will be used as name for the Project.

The following files will be generated:

- *filename.prj* – Contains the information about the Project and the Program.
- *filename.prj.lst* – Contains the information about the Register Table.
- *filename.prj.tbl* – Contains the information about the Linearization Tables.

To load a Project previously saved click on the “Open Project” button (Fig.2.1-B) or select “Project -> Open” in the Menu bar. It will be loaded the Program, the existing Tables, the name and the types of the Registers.

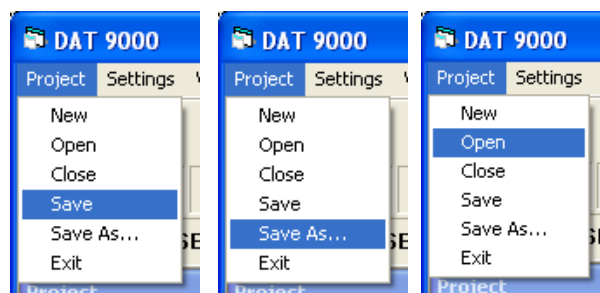


Fig. 2.2

2.3 – INSERT, MODIFY AND REMOVE A FUNCTION BLOCK

To set a Function Block it is necessary to activate it, clicking inside the block: the block is active when its border is highlighted in red.

It is possible to insert a new Function Block before or after the highlighted block (Fig.2.3-A).

Click on the “Insert Before” button (Fig.2.2-A) to insert a new block between the highlighted block and the previous.

Click on the “Insert After” button (Fig.2.2-B) to insert a new block between the highlighted block and the successive.

The new block inserted will be automatically highlighted.

To remove a Function Block, click on the block to select it and then click on the “Delete” button (Fig.2.2-D).

To modify a Function Block, click two times on the block or click on the block to select it and then click on the “Modify” button (Fig.2.2-C).

The operations described above are also accessible from the menu that appears clicking the right button of the mouse inside the highlighted Function block.

The “Zoom” button (Fig.2.3-B) allows to change the scale of visualization of the Main Program window reducing the dimensions of the Function Blocks (Zoom -) or turn back to the standard visualization (Zoom +).

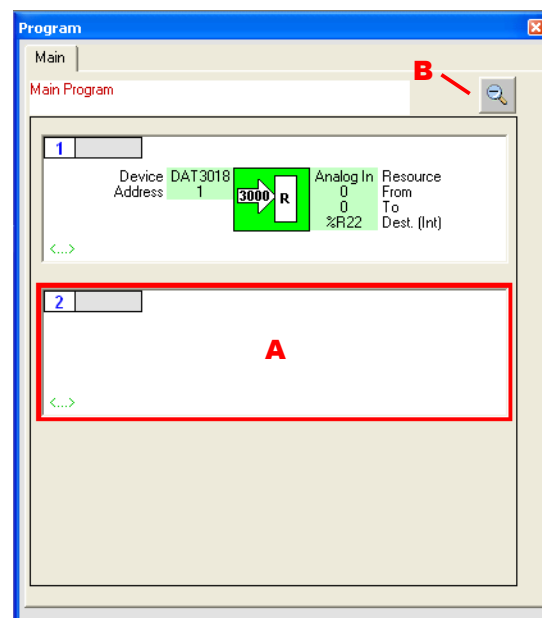


Fig. 2.3

When the user modifies a Function Block, the “*Function Block*” window (Fig.2.4) will be visualized; this allows to set the variables relative to the Function selected.

The Functions are gathered in specific functional groups: to visualize them, click on the button relative to a functional group (Fig.2.4-A) in order to visualize its specific functions (Fig.2.4-B).

Clicking on the button relative to the function to be inserted; it will appear a menu where it is possible to set the Label and the Variables (Fig.2.4-C) proper of the selected function.

To define particular variables like Masks or Tables, it is possible to use the “*Set*” button to open a window of guided insertion of the value.

At the end, click on the “*OK*” button (Fig.2.4-D) to insert the function inside the Main Program window.

In the Main Program window, the following parameters will be visualized:

- Index (Fig.2.5-A): unique number that identifies the position of the block inside the Program.
- Label (Fig.2.5-B): unique label that identifies the Function Block inside the Program (used for the functions of the “Flow” functional group, refer to section “Function description”).
- Symbol (Fig.2.5-C): icon relative to the function to be inserted.
- Variable (Fig.2.5-D): parameter of the Function Block.

Refer to the section “Function description” to know how to insert the functions.

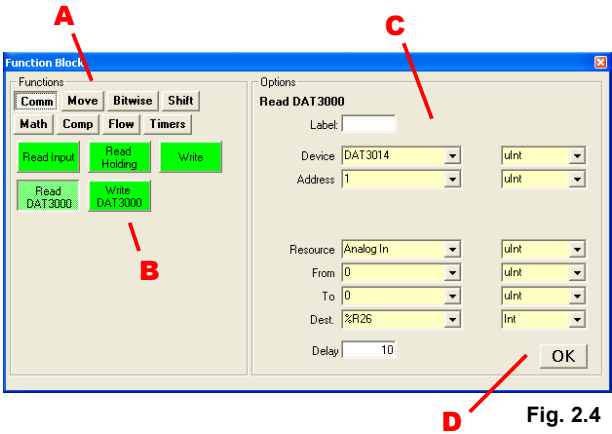


Fig. 2.4

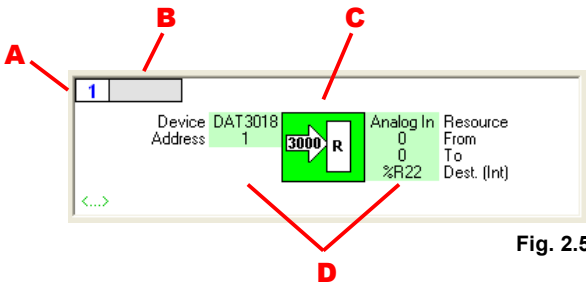


Fig. 2.5

2.4 – COMPILE AND ERROR CHECK

When the insertion of the Function Blocks is complete, it is possible to compile the Program clicking on the “*Compile*” button (Fig.2.6).

It will be created a report in the *Log* window (Fig.2.7) containing the eventual errors and/or anomalies encountered during the error checking procedure(Fig.2.7-A).

If the compiling process has a successful conclusion, the *Log* window will show the memory resources in use by the Program (Fig.2.7-B).



Fig. 2.6

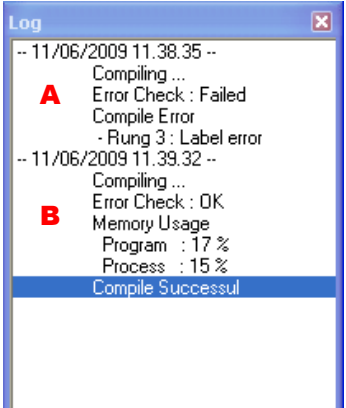


Fig. 2.7

3.1 – INTERNAL REGISTERS

The Internal Memory of the Controller is composed of a 16 bit Register series divided as follows:

- %S System Registers:** contain the information about the status Controller.
- %R General Purpose Registers:** can be used in the Program to move the data or to execute calculation functions.
- %M Retentive General Purpose Registers:** can be used from the Program to move the data or to execute calculation functions. These Registers are saved in Eprom each time their values change and they are uploaded when the Controller is powered-on.

Click on the “Watch” button (Fig.3.1) to visualize the Registers Table (Fig.3.2). To update the Register's value, click on the “Read” button (Fig.3.2-A).

For each Register it is visualized:

- The Register address (Fig.3.2-B)
- The Register name (Fig.3.2-C)
- The value contained into the Register (Fig.3.2-D)
- The Register data format (Fig.3.2-E)

To modify the name or the value of a Register, click two times on the row of table regarding the Register.

Inside the “Set” window (Fig.3.3) it is possible to set the name (Label) of the Register (only for the General Purpose Register), to force the value contained in the Register (only if the Controller is connected) and to set the type of data format of the Register.

3.2 – TYPE OF DATA FORMAT

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

- Uint 16 bit Unsigned Integer (0 ÷ 65535)
- Int 16 bit Signed Integer (-32768 ÷ +32767)
- Ulong 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
- K_Flt Floating Point Kostant (as decimal)
- K_Long Long Kostant (as binary code)

NOTE: the 32 bit Registers request the position of 2 Registers.

Fig. 3.1



Fig. 3.2

Reg	Name	Value	Type
%S0	--Reserved--	0	uint
%S1	Firmware[0]	0000	Hex
%S2	Firmware[1]	0000	Hex
%S3	Name[0]		ASCII
%S4	Name[1]		ASCII
%S5	Port 1 Settings	0000	Hex
%S6	Address	0	uint
%S7	Port 1 Timeout	0	uint
%S8	Digital Inputs	00000000 00000000	Bin
%S9	Digital Outputs	00000000 00000000	Bin
%S10	System Flags	00000000 00000000	Bin
%S11	--Reserved--	0	uint
%S12	WatchDog	0	uint
%S13	PC	0000	Hex
%S14	Status[0]	00000000 00000000	Bin
%S15	Status[1]	0	uint
%S16	COM Errors	0	uint
%S17	Gateway Mask[L-H]	0000	Hex
%S18	Port 0 Settings	0	uint
%S19	port 0 Settings	0	uint
%S20	Timers Enable	00000000 00000000	Bin
%S21	--Reserved--	0	uint
%S22	RTC[Sec]	0000	Hex
%S23	RTC[Min/Hours]	0000	Hex
%S24	RTC[Day/Date]	0000	Hex
%S25	RTC[Month/Year]	0000	Hex
%R26		0	Int
%R27		0	Int
%R28		0	Int
%R29		0	Int
%R30		0	Int
%R31		0	Int
%R32		0	Int
%R33		0	Int
%R34		0	Int

Fig. 3.3

Set

%R24

Label : Type :

Value :

Cancel OK

3.3 – MAPPING Register

MODEL DAT9000-DL-IO – Ver.FW 9000

Register	Description	Access
%S0	--Reserved--	R/W
%S1	Firmware [0]	R
%S2	Firmware [1]	R
%S3	Name [0]	R/W
%S4	Name [1]	R/W
%S5	Port 1 [BaudRate]	R/W
%S6	Node ID	R/W
%S7	Port 1 [Timeout RX]	R/W
%S8	Digital Inputs	R/W
%S9	Digital Outputs	R/W
%S10	System Flags	R/W
%S11	--Reserved--	-
%S12	--Reserved--	-
%S13	PC	R
%S14	Status [0]	R
%S15	Status [1]	R
%S16	COM Errors	R/W
%S17	Gateway Mask [L-H]	R/W
%S18	Port 0 [Settings]	R/W
%S19	Port 0 [Settings]	R/W
%S20	Timers Enable	R/W
%S21	--Reserved--	-
%R22	RTC [0]	R/W
%R23	RTC [1]	R/W
%R24	RTC [2]	R/W
%R25	RTC [3]	R/W
%R26	General Purpose Registers	R/W
%R959	Retentive Registers	R/W
%R960		
%R1023		

3.4 – SYSTEM REGISTERS OVERVIEW

This paragraph describes the working of the System Registers. Refer to the Registers Mapping relative to the device in use to find out the position and the type of access to the Register.

FIRMWARE

Field of 2 read only Registers: contains the identifier firmware code provided from the manufacturer.

NAME

Field of 2 read/write Registers (4 bytes or 4 ASCII characters) at the user disposal: it can contain the name of Controller or a unique code that identifies its function in the plant. Each one of the byte can contain any value from 0 up to 255, the ASCII characters are included.

The default value of this field contains the identifier of the device in ASCII characters.

PORT 1 Baud Rate

Field of 1 read/write Register used to select the baud rate of the PORT 1 Modbus Master serial port. Set the value in function of the following table:

BaudRate	Value
1200	0
2400	1
4800	2
9600	3
19200	4
38400	5
57600	6
115200	7

NODE ID

Field of 1 read/write Register: contains the Modbus node address of the device; the addresses allowed are from 1 up to 247. Each device connected to the same net must have a unique address.

PORT 1 Timeout RX

Field of 1 read/write Register: contains the value of the delay time successive to the reception of the response on the PORT1 Modbus Master; the value is expressed as milliseconds.

DIGITAL INPUTS

Field of 1 read/write Register: contains the status of the Digital Inputs (0 = OFF , 1 = ON).

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Descr.					Input											
Channel	-	-	-	-	#3	#2	#1	#0	-	-	-	-	-	-	-	-

DIGITAL OUTPUTS

Field of 1 read/write Register: contains the status of the Digital Outputs and allows to drive directly the Output relays (0 = OFF , 1 = ON) .

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Descr.					Output											
Channel	-	-	-	-	-	-	#1	#0	-	-	-	-	-	-	-	-

SYSTEM FLAGS

Field of 1 read/write Register: contains the following system flags:

- PowerUp Event : this bit is forced to 1 at each Controller power-on. It is possible to set the value as 0 to monitor eventual Controller reset events .

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Coil	-	-	-	-	WE	PU	-	-	-	-	-	-	-	-	-	-

P	PowerUp Event
0	Normal Mode
1	Reset Occurred
WE	Watchdog Enable
0	Watchdog disabilitato
1	Watchdog abilitato

PC (Program Counter)

Field of 1 read only Register: shows the position of the instruction (Function Block) executed in the Main Program . The value 0 is the first instruction (Function Block 1).

STATUS

Field of 2 read only Registers reserved for diagnostic operations.

COM ERRORS

Field of 1 read/write Register: is a counter of communication errors on the PORT1 Modbus Master. The value of this Register is incremented each time that a query is sent on the Master Port and there is not response. This value can be resetted.

GATEWAY MASK

Field of 1 read/write Register: contains the range of Modbus addresses which the Controller can send to queries on the PORT1 Master Port. If is requested to the Controller to send queries to an address out of this range, the command won't be executed.

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Descr.	First address of the Mask								Last address of the Mask							

PORT 0 Settings

Field of 2 read/write Registers: contain the parameters “Baud-rate” and “delay time” (expressed as milliseconds) of the serial PORT0 Slave Port. Set the value of the first Register in function of the following table:

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Descr.	Baud Rate								Delay (0÷255)							

BaudRate	Value
1200	0
2400	1
4800	2
9600	3
19200	4
38400	5
57600	6
115200	7

TIMERS Enable

Field of 1 read/write Register. Each bit of this Register is associated to an internal Timer. The Timer starts to count when the associated bit is set as 0; when the parameter “time” of the Timer Function Block set for the Timer has passed, the Timer stops to count and the associated bit is automatically forced to 1.

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Descr.	Timers															
Channel	T7	T6	T5	T4	T3	T2	T1	T0	T15	T14	T13	T12	T11	T10	T9	T8

RTC (Real Time Clock/Calendar)

Field of 4 read/write Registers: contain the value of the internal clock.

The following information are available:

Seconds 00 ÷ 59
 Minutes 00 ÷ 59
 Hours 00 ÷ 23
 Day of week 01 ÷ 07 (01=Sunday, 02=Monday, 07=Saturday)
 Date 01 ÷ 31 (it depends on the month)
 Month 01 ÷ 12 (01=January, 02=February, 12=December)
 Year 00 ÷ 99 (00=2000, 99=2099)

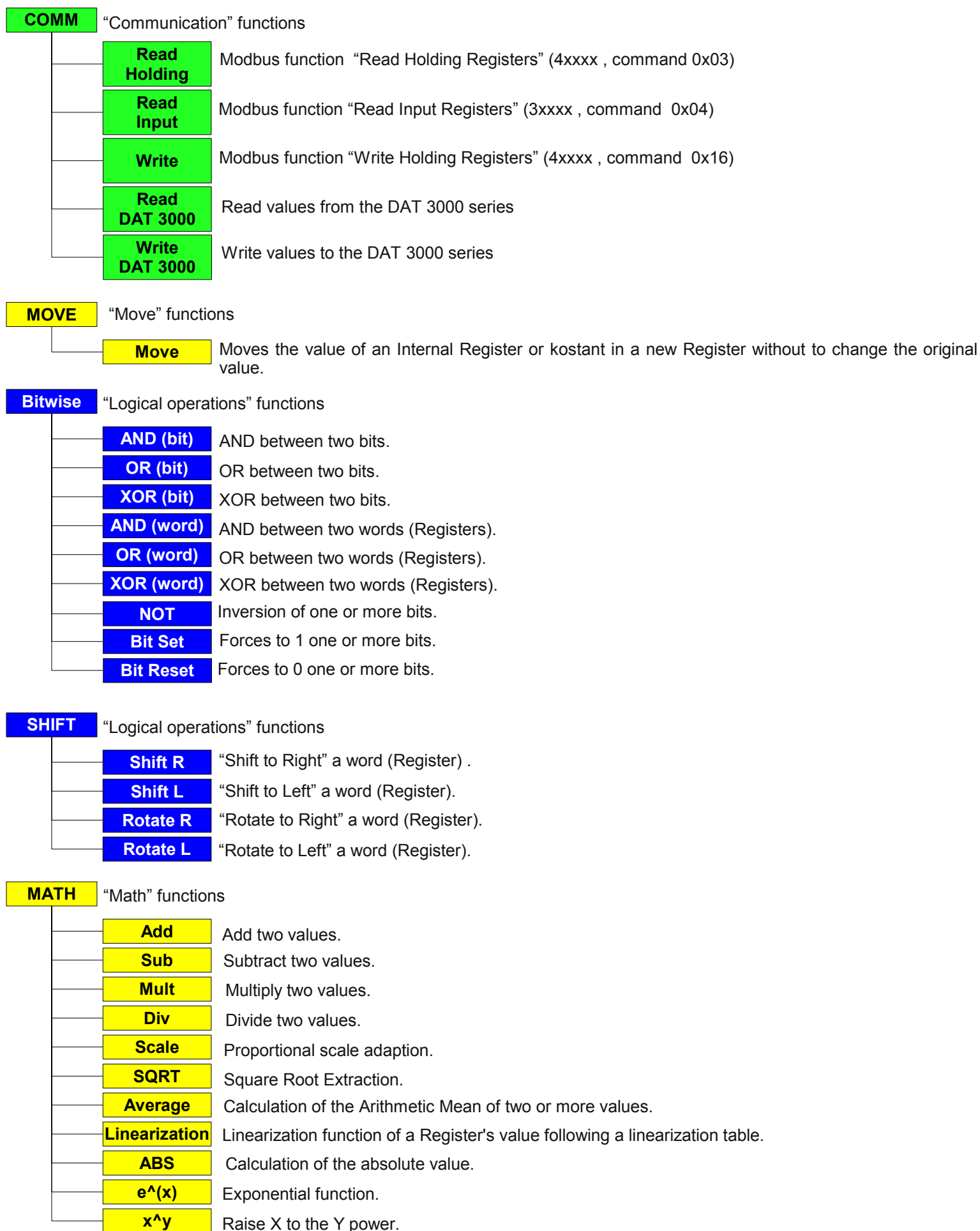
NOTE: all of the values are expressed as hexadecimal characters .

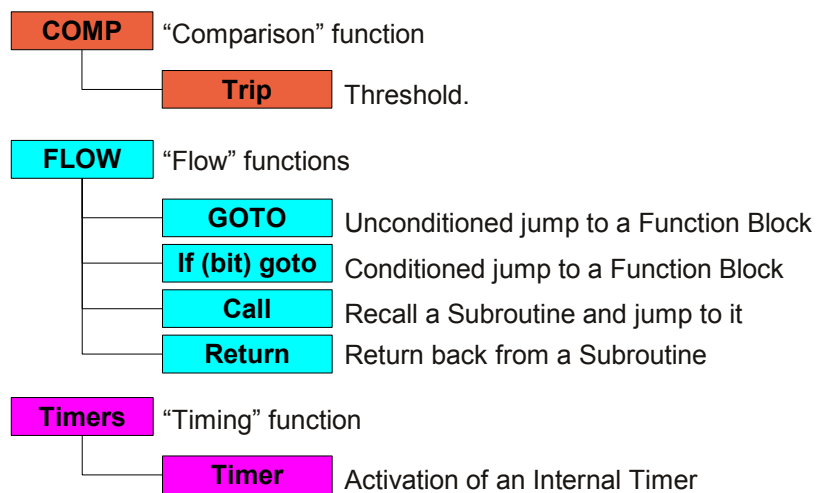
NOTICE! Writing these Registers will imply the variation of the clock and calendar settings.

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RTC[0]									Seconds [0÷59]							
RTC[1]	Minutes [0÷59]								Hours [0÷23]							
RTC[2]	Day of week [1÷7]								Date [1÷31]							
RTC[3]	Month [1÷12]								Year [0÷99]							

4.1 – FUNCTION LIST

Functions selection tree, divided per functional group





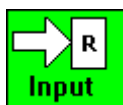
4.2 – FUNCTIONS DESCRIPTION

Read Holding Read the Holding Registers from a Modbus slave device

Reads the values of *N* holding *Registers* (Modbus function 0x03, Registers 4xxxx) from a Modbus slave device and writes the values in the selected Internal Registers. 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.

Variables:

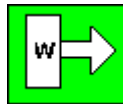
Address Modbus address of the slave device (1÷247)
Register Address of the first Register to read (the mapping Registers starts from 0)
Num Numbers of Registers to read (1÷16)
Dest Address of the first Internal Register wherein the read values are written to.
Delay Delay time between the reception of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

Read Input Read the Input Registers from a Modbus slave device

Reads the values of *N* input *Registers* (Modbus function 0x04, Registers 3xxxx) from a Modbus slave device and writes the values in the Internal Registers. 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.

Variables:

Address Modbus address of the slave device (1÷247)
Register Address of the first Register to read (the mapping Registers starts from 0)
Num 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 Write the Holding Registers of a Modbus slave device

Writes the values of *N* Internal *Registers* in *N* Holding *Registers* of a Modbus slave device (Modbus function 0x16, Registers 4xxxx). 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.

Variables:

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

Read3000 Read the I/O Registers from a Modbus slave DAT3000 series device

Reads the I/O values from a Modbus slave DAT3000 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 DAT3000 series device for the complete description of the I/O Registers.

Variables:

Device Selection of the DAT3000 series slave device.
Address Modbus address of the slave device (1÷247).
Resource Type of data 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.
Delay Delay time between the reception of the response and the execution of the next instruction.

Write3000 Write the I/O Registers of a Modbus slave DAT3000 series device

Writes the values of *N* Internal Registers in the I/O Register of a Modbus slave DAT3000 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 DAT3000 series device for the complete description of the I/O Registers.

Variables:

Device	Selection of the DAT3000 series device.
Address	Modbus address of the slave device (1÷247)
Resource	Type of data 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 withdraw from.
Delay	Delay time between the reception of the response and the execution of the next instruction.

Move Move the value of an Internal Register or kostant in a new Register without to change the original value.

Writes in an Internal Register the value of a Kostant (preset) or the value of another Register (copy). 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 (modify the format of the Register).

Variables:

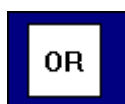
Source	Kostant or Internal Register from which the value is read.
Dest	Internal Register wherein the value is written.

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

Executes the logical operation "AND" between a Register and a kostant (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 (modify the format of the Register). 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 kostant (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).

Variables:


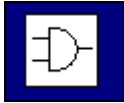
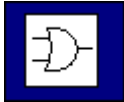
Source A	Kostant or Internal Register relative to the first operator.
Source B	Kostant or Internal Register relative to the second operator.
Mask Out	Mask applied to the result.
Dest	Internal Register wherein the result is written.

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

Executes the logical operation "OR" between a Register and a kostant (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 (modify the format of the Register). 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 kostant (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).

Variables:

Source A	Kostant or Internal Register relative to the first operator.
Source B	Kostant or Internal Register relative to the second operator.
Mask	Mask applied to the result.
Dest	Internal Register wherein the result is written.

Xor (word)	Executes the logical operation “XOR” (Exclusive Or) between two values.
	<p>Executes the logical operation “XOR” between a Register and a kostant (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 (modify the format of the Register). 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 kostant (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).</p>
<p>Variables:</p> <p>Source A Kostant or Internal Register relative to the first operator. Source B Kostant or Internal Register relative to the second operator. Mask Out Mask applied to the result. Dest Internal Register wherein the result is written.</p>	
And (bit)	Executes the logical operation “AND” between two bits.
	<p>Executes the logical operation “AND” (bit to bit) between a Register and a kostant (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 (modify the format of the Register). 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 kostant (long) and a 16 bit Register of destination (integer), the most significant bits will be ignored.</p>
<p>Variables:</p> <p>Source A Kostant or Internal Register relative to the first operator. Mask A Bits of the first operator Source B Kostant or Internal Register relative to the second operator. Mask A Bits of the second operator Dest Internal Register wherein the result is written. Mask Out Mask applied to the Register of destination.</p>	
Or (bit)	Executes the logical operation “OR” between two bits.
	<p>Executes the logical operation “OR” (bit to bit) between a Register and a kostant (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 (modify the format of the Register). 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 kostant (long) and a 16 bit Register of destination (integer), the most significant bits will be ignored.</p>
<p>Variables:</p> <p>Source A Kostant or Internal Register relative to the first operator. Mask A Bits of the first operator Source B Kostant or Internal Register relative to the second operator. Mask A Bits of the second operator Dest Internal Register wherein the result is written. Mask Out Mask applied to the Register of destination.</p>	

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


Executes the logical operation “XOR” (bit to bit) between a Register and a kostant (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 (modify the format of the Register). 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 kostant (long) and a 16 bit Register of destination (integer), the most significant bits will be ignored.

Variables:

Source A Kostant or Internal Register relative to the first operator.
Mask A Bits of the first operator
Source B Kostant or Internal Register relative to the second operator.
Mask A Bits of the second operator
Dest Internal Register wherein the result is written.
Mask Out Mask applied to the Register of destination .

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.

Variables:

Source Internal Register containing the value.
Dest Internal Register wherein the result is written.
Mask Out Mask applied to the Register of destination.

Shift R 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.

Variables:

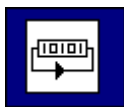
Source Internal Register containing the value.
N Number of shift to execute.
Dest Internal Register wherein the result is written.

Shift L 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.

Variables:

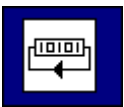
Source Internal Register containing the value.
N Number of shift to execute.
Dest Internal Register wherein the result is written.

Rotate R**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.

Variables:

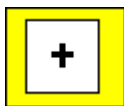
Source Internal Register containing the value.
N Number of shift to execute.
Dest Internal Register wherein the result is written.

Rotate L**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.

Variables:

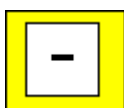
Source Internal Register containing the value
N Number of shift to execute.
Dest Internal Register wherein the result is written.

Add**Calculates the sum of two values**

Calculates the sum between an Internal Register and a Kostant or between two Internal Registers.

Variables:

Source A Kostant or Internal Register relative to the first operator.
Source B Kostant or Internal Register relative to the second operator.
Dest Internal Register wherein the result is written.

Sub**Calculates the difference of two values**

Calculates the difference between an Internal Register and a Kostant or between two Internal Registers.

Variables:

Source A Kostant or Internal Register relative to the first operator.
Source B Kostant or Internal Register relative to the second operator.
Dest Internal Register wherein the result is written.

Mult**Calculates the multiplication of two values**

Calculates the multiplication between an Internal Register and a Kostant or between two Internal Registers.

Variables:

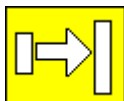
Source A Kostant or Internal Register relative to the first operator.
Source B Kostant or Internal Register relative to the second operator.
Dest Internal Register wherein the result is written.

Div Calculates the division between two values.

Calculates the division between an Internal Register and a Kostant or between two Internal Registers.

Variables:

Source A Kostant or Internal Register relative to the first operator.
Source B Kostant or Internal Register relative to the second operator.
Dest Internal Register wherein the result is written.

Scale Executes the proportional scaling of a value

Executes the proportional scaling between the input range values and the output range values, referring to the value of an Internal Register.

Variables:

Source Internal Register containing the value to scale.
Span In Maximum value of the input range
Zero In Minimum value of the input range
Dest Internal Register wherein the result is written.
Span Out Maximum value of the output range
Zero Out Minimum value of the output range

SQRT Calculates the Square Root of a value

Calculates the Square Root of a value contained in an Internal Register.

Variables:

Source Internal Register containing the value
Dest Internal Register wherein the result is written.

Average Calculates the Arithmetic mean of N values.

Calculates the Arithmetic mean of N Internal Registers values. (sum of the values / N).

Variables:

Source Address of the Internal Register containing the first value.
N Number of Internal Register of which calculate the mean.
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 "Table" for more information.

Variables:

Source Internal Register containing the value to linearize.
Function Name of the Linearization table to be followed.
Dest Internal Register wherein the result is written.

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 T_{on} , 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 T_{off} , 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 T_{on} and $T_{off} = 0$.

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 (Fig.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.

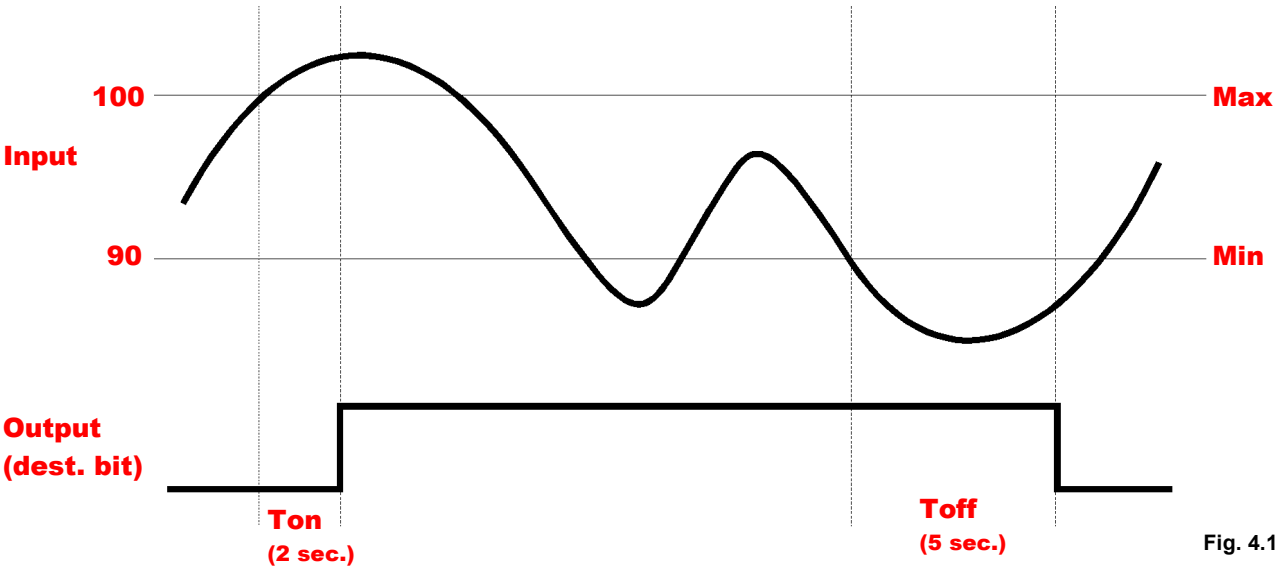







Fig. 4.1

Variables:

Input	Internal Register containing the value to compare
Max	High Trip level
Min	Low Trip level
Dest	Internal Register wherein the result is written.
Mask	Mask applied to the result.
Use Timer	Number of Internal Timer to use (0÷15)
Timer On	Delay time for Trip Alarm activation (ms)
Timer Off	Delay time for Trip Alarm de-activation (ms)

Goto	Unconditioned jump to a Function Block.
	Executes an unconditioned jump to a Function Block. The Function Block recalled must have a unique Label. It is possible to select the Label of the Function Block which jump to, sorting out it between those available in the listbox "Func.Block". The list is automatically updated each time that the user identifies a new Block; in the case of the list is empty the value 0 appears.
Variables:	
Func.Block	Pointer to the Function Block which jump to.
If (bit)	Conditioned jump to a Function Block.
	Executes a conditioned jump to the Function Block indicated in the variable "Goto if true" if the status of the bit selected is 1, while if the status of the bit is 0 jumps to the Function Block indicated in the variable "Goto if false". All of the Function Blocks called must have a unique Label. It is possible to select the Label of the Function Block which jump to, sorting out it between those available in the listbox "Goto if true" and "Goto if false". The lists are automatically updated each time that the user identifies a new Block; in the case of the list is empty the value 0 appears.
Variables:	
Source	Internal Register used as reference
Bit	Number of the bit to control (0÷15)
Goto if True	Pointer to the Function Block which jump to (if Bit=1)
Goto if False	Pointer to the Function Block which jump to (if Bit=0)
Call	Recall a Subroutine
	Executes a jump to the first Function Block of a Subroutine; at the end of the Subroutine (command "Return"), the Function Block successive to the block "Call" will be executed. The Function Block recalled must have a unique Label. It is possible to select the Label of the Function Block which jump to, sorting out it between those available in the listbox "Func.Block". The list is automatically updated each time that the user identifies a new Block; in the case of the list is empty the value 0 appears. NOTE: for each block "Call" must correspond a block "Return", otherwise it is possible to occur in a "Stack Overflow" error (refer to the section Troubleshooting).
Variables:	
Func.Block	Pointer to the Function Block which jump to
Return	Return from a Subroutine
	Indicates the end of a Subroutine. The Program will return to the Block following the Function Block "Call". NOTE: for each block "Return" must correspond a block "Call", otherwise it is possible to occur in a "Stack Overflow" error (refer to the section Troubleshooting).
Timers	Activation of an Internal Timer
	Sets an Internal Timer 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.
Variables:	
Timer mSec	Number of the Internal Timer to enable Timer Preset (milliseconds)

5.1 – INSERTION OF TABLES

To insert the linearization tables, it is necessary to open the proper window selecting “Tools -> Tables” in the Menu bar (Fig.5.1).

To upload the points of a table from a file, click on the “Load from File” button (Fig.5.1-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.

Each table must be associated to a unique name that identifies the table inside the Program when the user recall it: to modify the name, click on the text box “Name” (Fig.5.1-B) and write the new name or select one between those existing.

The number of points (32 max.) defines the steps of linearization applied to a variable.

To modify it click on the text box “N points” (Fig.5.1-C) and write the new value: the window will be automatically updated.

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 (Fig.5.1-D) shows how to create an 8 points table to linearize a RTD temperature sensor and to obtain the conversion Ohm/°C (Fig.5.2).

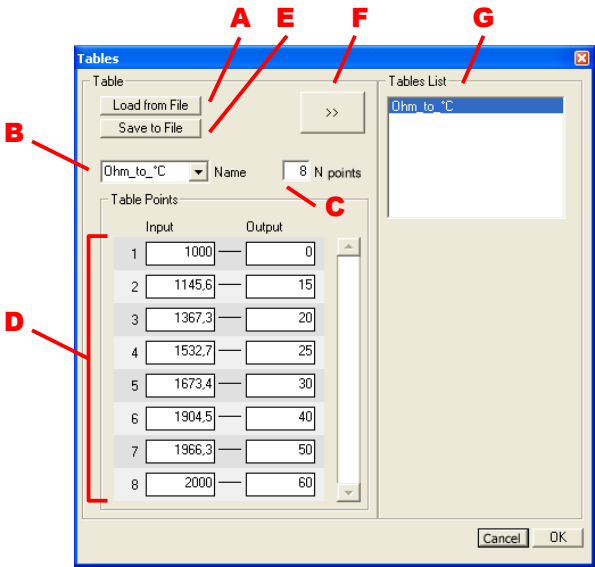
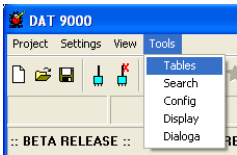


Fig. 5.1

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

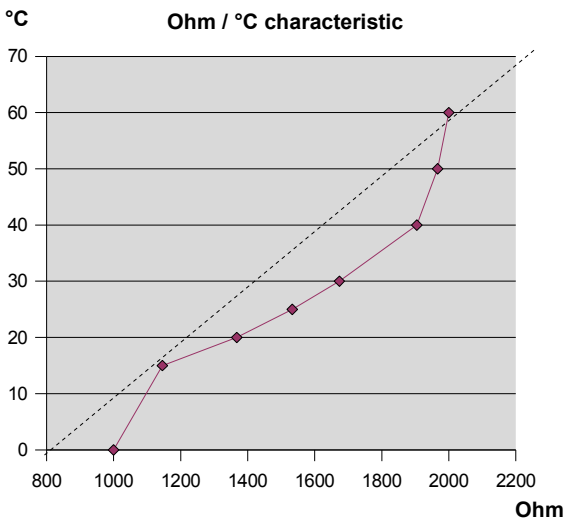


Fig. 5.2

When the insertion of points is complete, it is possible to save the table in a file, clicking on the “Save to File” button (Fig.5.1-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 (Fig.5.1-F). The “Table List” will be updated with the name of the table just inserted and shows the tables available for the Program. To turn back to the Function Block window, click on the “OK” button.

Select the Internal Register which the linearization curve will be associated to and click on the “OK” button to turn back to the Main Program window.

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 (Fig.5.1-D) for an input value of 1789 Ohm, will be calculated an output of 35 °C.

6.1 – SEARCHING OF THE DEVICES CONNECTED

Connect the Controller to the Ethernet network and power-on it (refer to the data-sheet).

Open the window “Search” (Fig.6.1) selecting “Tools -> Search” in the Menu bar. In this window it is possible to select the type of Controller to search (Fig.6.1-A) and to set the receiving Timeout (time over which the device is intended as not connected) - Fig.6.1-B. To search eventual devices connected to the serial Modbus-master port of the Controller (Sub-Nodes), the user must indicate the range of addresses to control (Fig.6.1-C); to use this function it is necessary that the Controller is set in “Debug” modality and in “Stop” condition.

When the options have been selected, click on the “Search” button (Fig.6.1-D) to start the search. If necessary it is possible to interrupt the search clicking on the “Stop” button.

When a Controller compatible with the type selected is recognized, it is visualized in the Ethernet list (Fig.6.1-E), wherein it is possible to read the IP address, the MAC address and the Modbus node of the device.

Afterwards, if required, the Modbus slave devices connected to the Controller will be searched. The recognized devices will be visualized in the Subnet list (Fig.6.1-F), wherein it will be possible to read the Modbus slave node address and, in case of the DAT3000 series devices, the Firmware code. At the end of the search, it is possible to select one of the Controllers detected clicking on its name and set it as default Controller, clicking on the name by the right button of the mouse and selecting the “Set as Controller” option; after this operation, the connection to the Controller will be automatically executed. Refer to the next paragraph to set the Controller manually.

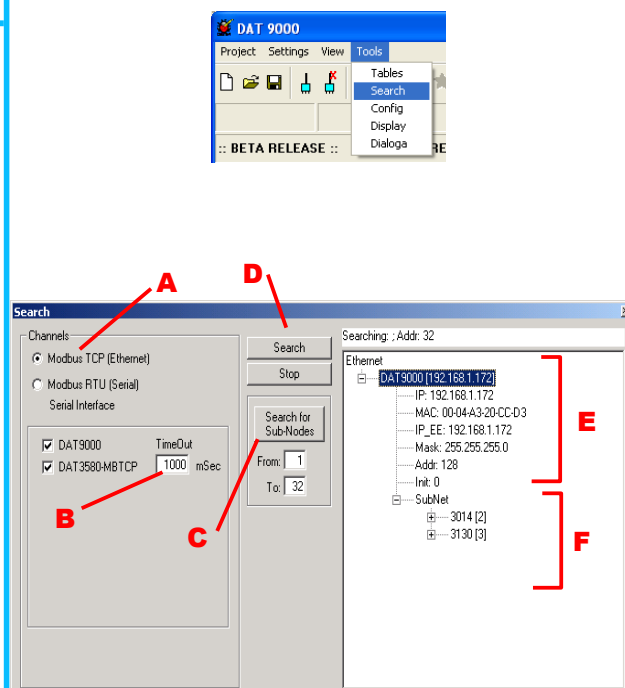


Fig. 6.1

6.2 – CONNECTING TO THE CONTROLLER

Open the window “Settings” (Fig.6.2) selecting “Settings -> Controller” in the Menu bar.

Set the following parameters:

Node ID : Modbus node address (1 ÷ 247)
Channel : Communication Interface (Ethernet or serial port)
IP Address : Controller IP address
Port : Modbus/TCP socket reserver port (502 for direct connection)
Timeout : Receiving Timeout for TCP commands

To confirm, click on the “OK” button.

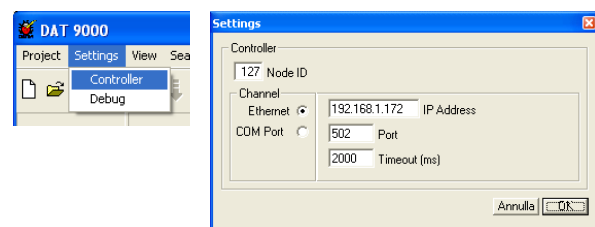


Fig. 6.2

Click on the “Connect” button (Fig.6.3-A): if the connection ends correctly, the message “Connected” will be visualized in the Status bar Fig.6.3-B) and in the Log window (Fig.6.4); 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 have to change the Controller, it is necessary to disconnect the Controller in use, click on the “Disconnect” button (Fig.6.3-C), modify the parameters in the “Settings” window and then click on the “Connect” button to communicate with the new Controller.

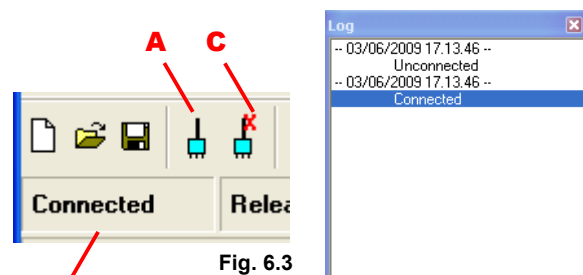


Fig. 6.3

Fig. 6.4

6.3 – DOWNLOAD THE PROGRAM

When the Program is complete and if the compiling ends correctly it is possible to download it in the RAM memory of Controller. To do it, open the “Download” window (Fig.6.6). clicking on the “Download” button (Fig.6.5-A).

The Download operations are allowed only in “Debug” modality (refer to the section “Debug Modality”).

Inside the “Download” window it is possible to set one or more options:

- “Download Program” (Fig.6.6-A) – Starts to download the Program in the RAM memory of Controller.
- “Verify” (Fig.6.6-B) – Compares the Program contained in the RAM memory with the compiled Program.
- “Save in Flash” (Fig.6.6-C) – Transfers the Program loaded in the RAM memory to the Internal Flash Memory of the Controller.
- “Clear Register Memory” (Fig.6.6-D) – Resets the value of the Controller's General Purpose Internal Registers.
- “Run After Download” (Fig.6.6-E) – At the end of the download, sets the Controller in “Run” modality (execution of the Program).

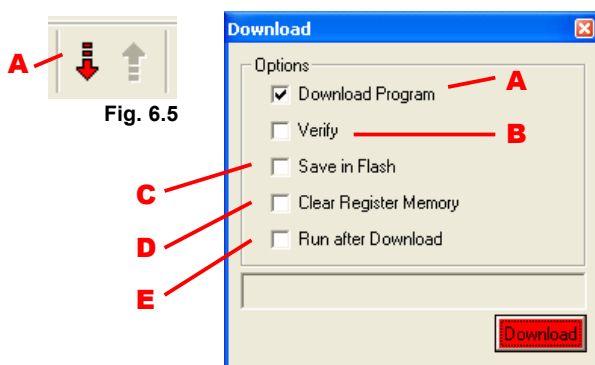


Fig. 6.6

6.4 – DEBUG MODALITY

By this modality it is possible to follow the Program flow and to monitor in real time the Controller's status and the value of the Internal Registers. When the Program is interrupted the Register Table is updated to the last reading.

During the development of the Program, if the Controller is connected, click on the “*Debug*” button (Fig.6.7-A) to activate the “Debug” modality.

In the Status bar the message “*Debug Mode*” (Fig.6.7-B) will be visualized and in the Tool bar will be activated the commands to execute the following debug operations:

- “*Run*” (Fig.6.7-C) – Executes the Program continuously.
- “*Run To Break*” (Fig.6.7-D) – Executes the Program up to the Break point .
- “*Pause*” (Fig.6.7-E) – Interrupts the execution of the Program (“Run” condition) / executes the Program step by step (“Stop” condition)
- “*Animate*” (Fig.6.7-F) – Simulates the evolution of the Program flow executing it step by step.
- “*Stop*” (Fig.6.7-G) – Blocks the Program and reset it to the first Function Block.

The Function Block in execution is identified by the Index parameter coloured in red and is updated the PC (“*Program Counter*”) value (Fig.6.8-A). Open the “*Settings*” window (Fig.6.9), selecting “*Settings -> Debug*” to set the following options for the Debug modality:

- “*Run-Time Register Update*” (Fig.6.9-A) – If active, in “Run” condition the table Register will be automatically updated (1 read per second)
- “*Animate Time*” (Fig.6.9-B) – Setting of the playing time between one step and the successive in the “Animate” condition.

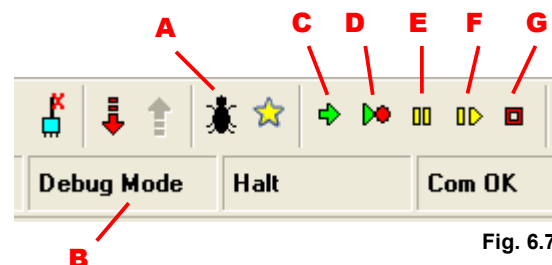


Fig. 6.7

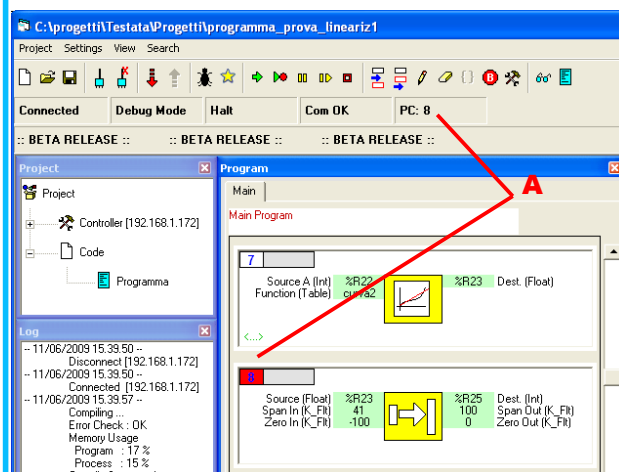


Fig. 6.8

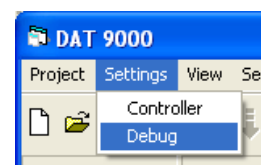


Fig. 6.9

6.5 – RELEASE MODALITY

After the phases of development and Debug it is possible to proceed with the “*Release*” modality, clicking on the “*Release*” button (Fig.6.10-A).

In the Status bar the message “*Release Mode*” (Fig.6.10 -B) will be visualized and in the Tool bar the commands relative to the operations of Debug and Download 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.

6.6 – INIT MODALITY

The “*INIT*” modality can be used in case of fortuitous loss of configuration to set the Controller in the default condition in order to recover the desired configuration. In this modality the value of the following parameters of the controller will be automatically set independently of the configuration saved in Eprom.

- IP Address Value assigned automatically from the network by a DHCP server
- Modbus Node 0x0Ah (10)
- PORT 0 (Slave) Baud-rate = 9600 bps

In this modality is not possible to execute the Program (Run) and the Debug functions are disabled but it is allowed to download the Program, to read and write the Internal Registers and the Gateway function is enabled.

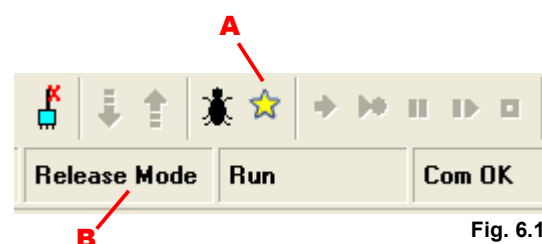


Fig. 6.10

IMPORTANT:

In “*Debug*” modality, at the power-on the Controller will be automatically set in “Stop” condition without load and execute the Program. For such reason, at the end of the Debug operations, set the Controller in “*Release*” modality.

In “*INIT*” modality is not possible to execute the Program (Run) and the Debug functions are disabled but it is allowed to download the Program, to read and write the Internal Registers and the Gateway function is enabled.

6.7 – WEB SERVER

By Web Browser it is possible to get the access to the Controller's Web Server in order to visualize the Web pages containing the data about the Ethernet configuration and the value of the Internal Registers.

To connect to the Web pages, it is necessary to write in the Address bar of the Browser in use the IP address of the controller which access to. The following example shows how to connect to the Web page *Index* of a Controller with IP address = 192.168.1.172

Address bar text:
<http://192.168.1.172/index.htm>

From the web page *Index* (Fig.6.11) it is possible to get the access to the following pages:

- **Configuration** (Fig.6.12) (<http://192.168.1.172/protect/config.htm>): allows to modify the TCP settings (IP address , Subnet mask, Gateway, etc...). Clicking on the “*Save Config*” button, the parameters set will be saved in Eprom and the Controller will be resetted. It is possible to protected this page by Password, modifying the fields “*User*” and “*Password*” and click on the “*Save Config*” button. WARNING!!in case of lost Password, the user won't be able to get back the information.
- **Dynamic Var** (Fig.6.13) (<http://192.168.1.172/dynvar.htm>): In this page are visualized the values of the Internal Registers from the %R26 up to the %R41. To update the value of the Registers it is necessary to use the F5 key.

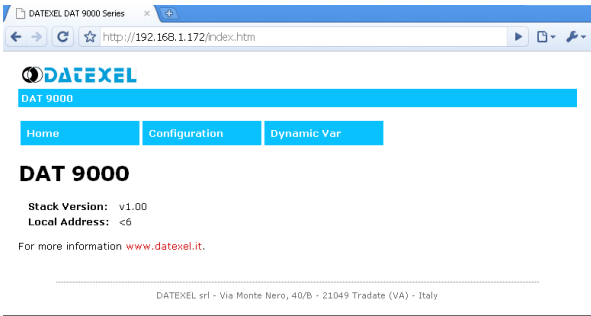


Fig. 6.11

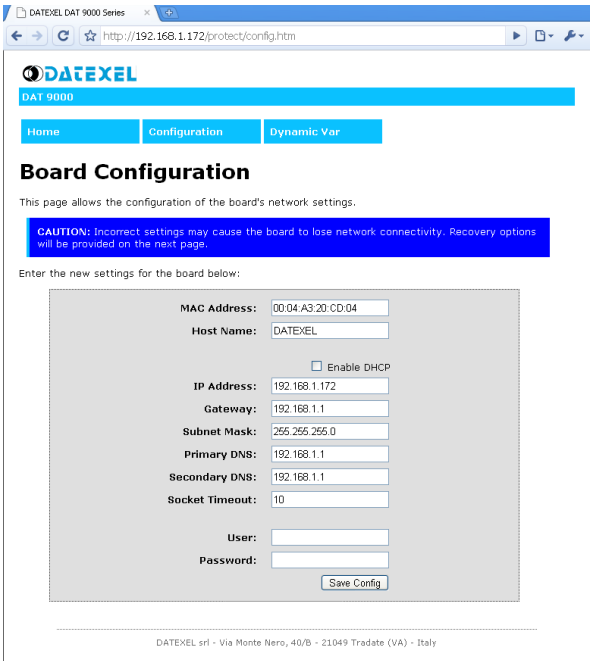


Fig. 6.12

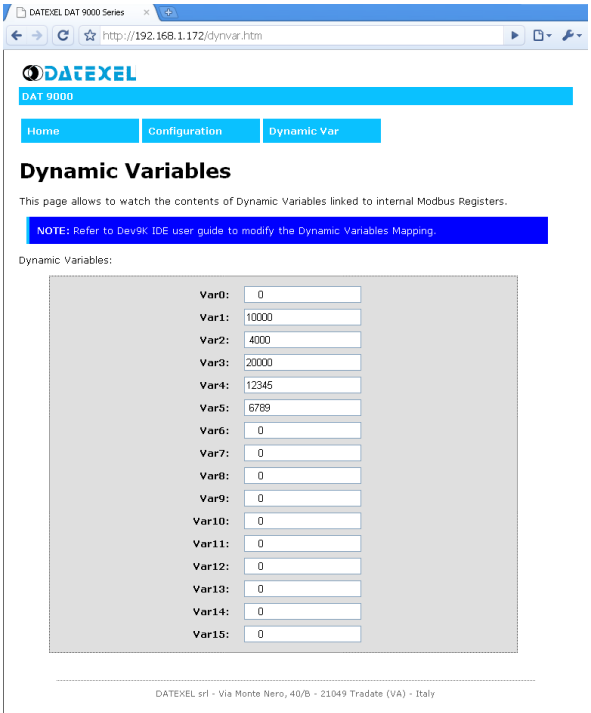


Fig. 6.13

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 are reported some practical tips to connect the Controller .

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 it 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 –ERROR MESSAGES IN THE LOG WINDOW AND IN THE STATUS BAR

EVENT	POSSIBLE CAUSES	POSSIBLE SOLUTIONS
“Not Connected”.	<ul style="list-style-type: none"> -The Controller is not connected. -The communication channel selected has not been enabled. 	<ul style="list-style-type: none"> -Verify in the “Settings” menu : <i>for Ethernet port:</i> -IP address -Port number reserved to the Modbus TCP socket (Port) -Time out value. -Modbus node ID of the device. <i>for serial port (COM):</i> -COM port number. -Baud rate. -Modbus node ID of the device.
“Com Error”.	<ul style="list-style-type: none"> -Wrong setting of the Ethernet port's communication parameters. -Wrong setting of the Slave port's communication parameters. -Wrong command addressing on the Master port. 	<ul style="list-style-type: none"> -Verify in the “Settings” menu : <i>for Ethernet port:</i> -IP address -Port number reserved to the Modbus TCP socket (Port) -Time out value. -Modbus node ID of the device. <i>for serial port (COM):</i> -COM port number. -Baud rate. -Modbus node ID of the device. -Verify in the “Config” menu : -Modbus node ID of the device. -Baud rate. - Delay of receiving (Slave port) <p>If the parameters have been correctly set check the connection of the device</p>
“Com Timeout”.	<ul style="list-style-type: none"> -With communication channel configured, indicates a missing reception of the response by the device. -Wrong communication parameters. 	<ul style="list-style-type: none"> -Verify in the “Settings” menu : <i>for Ethernet port:</i> -IP address -Port number reserved to the Modbus TCP socket (Port) -Time out value. -Modbus node ID of the device. <i>for serial port (COM):</i> -COM port number. -Baud rate. -Modbus node ID of the device.
“Label error”.	<ul style="list-style-type: none"> -One or more errors occur inside a Function Block when the Program is compiled,downloaded or verified. 	<ul style="list-style-type: none"> -Check the parameters of the Function Block identified by the index number visualized inside the log window. (refer to the sections 2.3 and 4.2).

8.2 – ERROR MESSAGES INSIDE THE POP-UP WINDOWS

EVENT	POSSIBLE CAUSES	POSSIBLE SOLUTIONS
<i>“Controller not connected”.</i>	<ul style="list-style-type: none"> -The Controller is not connected. -The communication channel selected has not been enabled. 	<ul style="list-style-type: none"> -Verify in the “Settings” menu : <i>for Ethernet port:</i> -IP address -Port number reserved to the Modbus TCP socket (Port) -Time out value. -Modbus node ID of the device. <i>for serial port (COM):</i> -COM port number. -Baud rate. -Modbus node ID of the device.
<i>“Timeout”.</i>	<ul style="list-style-type: none"> -The response provided by the Controller is not correct and the communication has been interrupted. 	<ul style="list-style-type: none"> -Verify in the “Settings” menu : <i>for Ethernet port:</i> -IP address -Port number reserved to the Modbus TCP socket (Port) -Time out value. -Modbus node ID of the device. <i>for serial port (COM):</i> -COM port number. -Baud rate. -Modbus node ID of the device. -Verify in the “Config” menu : -Modbus node ID of the device. -Baud rate. - Delay of receiving (Slave port) If the parameters have been correctly set check the connection of the device
<i>“Wrong response (function)”.</i>	<ul style="list-style-type: none"> - The Modbus slave device asked by the Controller doesn't provide a correct response. 	<ul style="list-style-type: none"> -Verify in the “Config” menu : -Modbus node ID of the device. -Baud rate set for the Slave device. - Delay of receiving (Slave port) - Stop bit, Parity type. - Number of the registers read or written. -Modbus function code transmitted. - Delay of receiving (Master port)
<i>“Check values”.</i>	<ul style="list-style-type: none"> - One or more parameters of a Function Block are not correct; this error occur when the user clicks on the “OK” at the moment to insert the Function Block in the Program. 	<ul style="list-style-type: none"> -Check the parameters of the Function Block: for the functions of external reading and writing, refer to the User Guide of the Slave device in use.

9.1 – POSSIBLE CAUSES OF FAULT

EVENT	POSSIBLE CAUSES	POSSIBLE SOLUTIONS
<i>Is not possible to power-on the Controller.</i>	<ul style="list-style-type: none"> -The Controller is not correctly powered. -The value of the power supply value is lower than the specifications limits. 	<ul style="list-style-type: none"> -Refer to the data-sheet of the Controller in use and verify the relative Technical Specifications.
<i>There is not communication between the Host PC and the Controller.</i>	<ul style="list-style-type: none"> -Ethernet port not correctly connected. -Modbus Slave port not correctly connected. -Eventual interface between PC and Controller not correctly connected. -Wrong communication parameters. 	<ul style="list-style-type: none"> -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 .
<i>There is not communication between the Controller and one or more Modbus slave devices.</i>	<ul style="list-style-type: none"> -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 %S17 (Gateway Mask) . 	<ul style="list-style-type: none"> -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.
<i>The Program is not correctly executed or it is impossible to execute the Program.</i>	<ul style="list-style-type: none"> -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. -Controller in Stack Overflow condition. -Parameters of the eventual Slave devices connected not correctly inserted. -The Program has not been downloaded -Controller in INIT modality. 	<ul style="list-style-type: none"> -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, in the Program, the correspondence between Call and Return. -Control the configuration of the Slave devices (type of input and output, etc..) -Download the Program. -Control if the INIT modality is active.
<i>The configuration of the Controller is unknown.</i>	-	<ul style="list-style-type: none"> -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 .
<i>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.</i>	<ul style="list-style-type: none"> -Controller not correctly connected. -Wrong port Baud Rate. 	<ul style="list-style-type: none"> -Connect the terminal INIT to GND. -Switch-off and then power-on the Controller after the connection of the terminal INIT to GND. -Set the Baud-rate of the Slave Port as 9600 bps.
<i>The functions Clock and Calendar (where foresee) don't work correctly.</i>	<ul style="list-style-type: none"> -Battery low or absent. -Clock and Calendar parameters not correctly set in the proper Registers. 	<ul style="list-style-type: none"> -Change or insert the battery. -Control the parameters of the System Registers (refer to the sections 3.3 and 3.4).
<i>The function "Search" doesn't find any Controller.</i>	<ul style="list-style-type: none"> -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 are active Firewall or Routers that block the access to the Controller. 	<ul style="list-style-type: none"> -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
<i>The function “Search” doesn’t find any Slave device.</i>	<ul style="list-style-type: none"> -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 %S17 (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. 	<ul style="list-style-type: none"> -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.
<i>The Web Pages haven’t been loaded.</i>	<ul style="list-style-type: none"> -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. 	<ul style="list-style-type: none"> -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.
<i>The data saved as Kostant in the Register table, are not saved when the Controller is switched off.</i>	<ul style="list-style-type: none"> -The data have been saved in General Purpose Registers instead Retentive Registers . 	<ul style="list-style-type: none"> -Save the kostant values in Retentive Registers.

