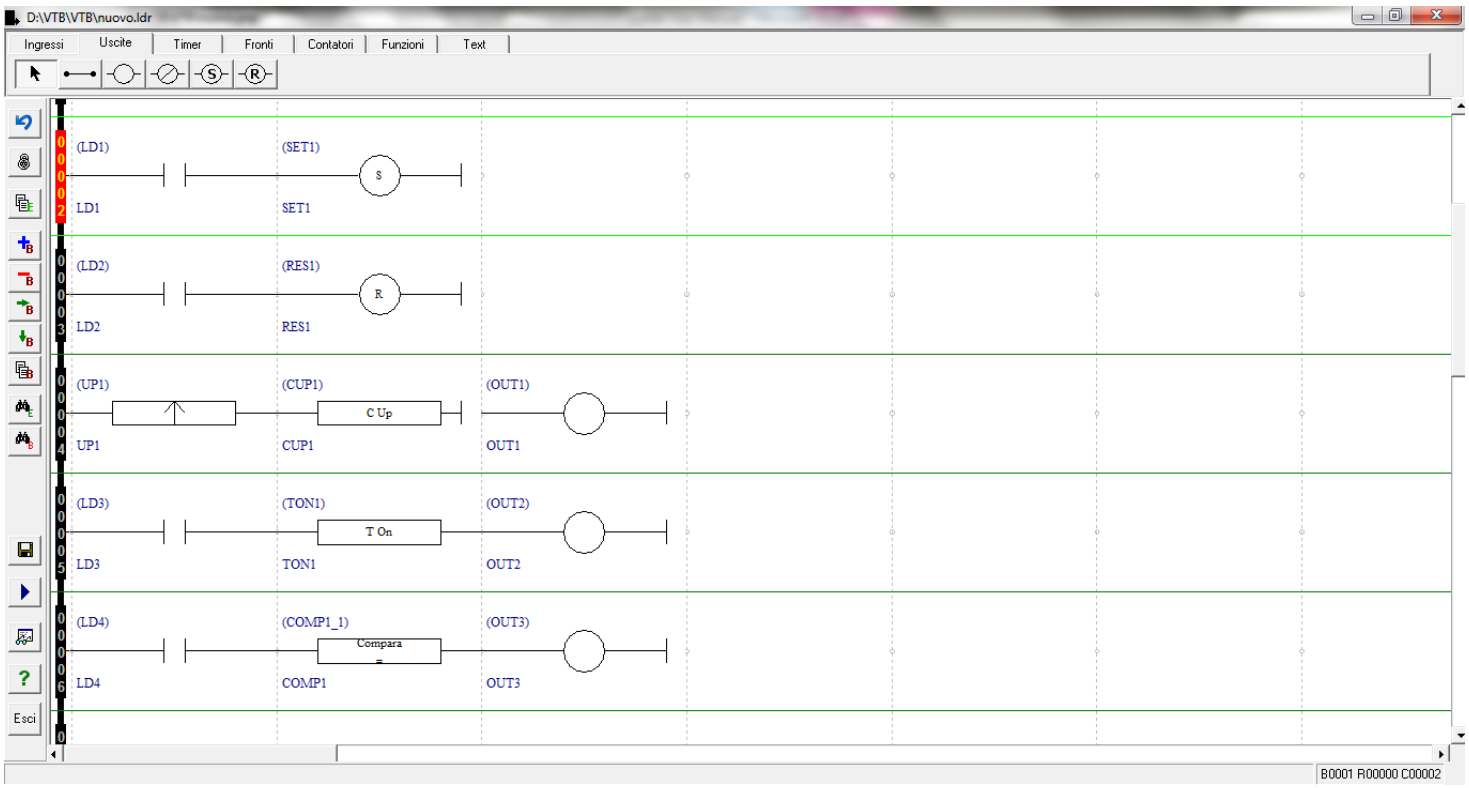
User Manual PROMAX Ladder's

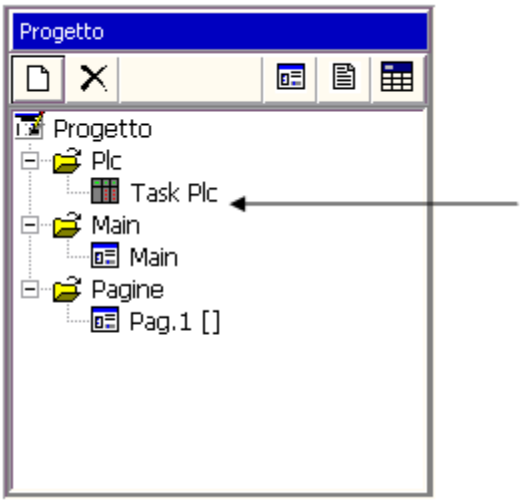


Use the Internal PLC LADDER

Using the CAN OPEN protocol is enabled TASK PLC to control fast I/O. This provides for the use of special instructions that are optimized for cycle management PLC, consequently any instructions belonging to the BASIC TASK can create system malfunction. Language the PLC used is very simple and practical and can also be generated by a compiler LADDER chart. Various bits of memory instructions operate, set / reset outputs, input management, timers etc that is needed to build a functional PLC cycle. It should be noted that the PLC is shared with the TASK PLC and hence the instructions to HIGH LEVEL are handled latter's, greatly reducing the load on the PLC. The TASK PLC has a high priority and is executed in a sampling time determined by the configuration of VTB .The Task PLC can be written with the EDITOR specific integrated in VTB or using the application PLC Ladder which consists of a graphics system contacts very simple and intuitive.

ACTIVATION PLC EDITOR

PLC Editor is activated by the Project window by selecting the TASK PLC ICON. Then you go to an ASCII EDITOR where you can write a PLC program using the instructions explained below.



Icon for activation PLC EDITOR

The PLC application is saved in the PROJECT and charged accordingly without loss of information. Enter instructions in the task PLC than those plc , may cause a malfunction of the program.

NOTES ON THE TASK PLC

The TASK PLC runs in a INTERRUPT at regular intervals by stopping the TASK PLC. The two TASK can share all kinds of variables being careful only with the 32-bit variables (LONG, FLOAT). This is because since the two may happen that the asynchronous TASK PLC is INTERRUPTED by the PLC when upgrading a 32-bit variable (since this requires two instructions ASSEMBLER) if the TASK PLC uses the same variable can be an incorrect reading or writing (from TASK PLC) of the variable. If the two shared variables using TASK 32-bit is recommended to select the shared variable flag in the page of page variables, VTB will perform well in automatic control. This system, however, slows the operation of the application.

The next system is WRONG because the control of VAR1 type LONG made in Task PLC can generate ERRORS.

Example: variables used:

```
var1 long      'counter declared in the MAIN
var3 char      'declaration for variable bit
var2 var3.1    'bits declared in TASK PLC used by PLC
```

```
Main
If var1 > 1000
        ……
        ……
endif

TASK PLC
PLC_LD var2
PLC_CUP var1
```

If you do not want to use shared variables or you want additional control, this is the correct system:

Example: variables used:

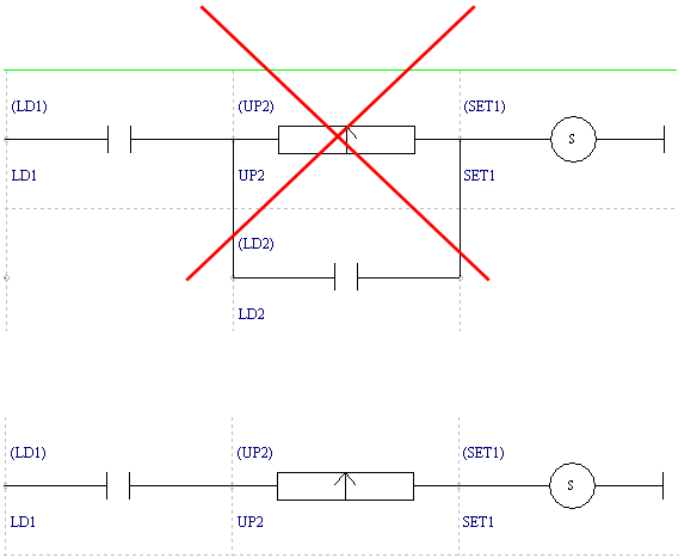
```
var1 long      'counter declared in the MAIN
var3 char      'declaration for variable bit
var2 var3.1    'bits declared in MAIN TASK used by PLC
var4 var3.2    'synchronism bit of TASK
```

```
Main
 If var4 = 1            'Check the status of bit synchronization
        ……
        ……
endif

TASK PLC
PLC_LD var2
PLC_CUP var1
PLC_CMP var1
PLC_GT 1000
PLC_OUT var4
```

NOTES ON THE LIMITATION OF LADDER

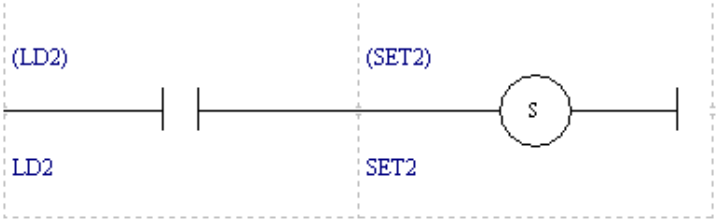
The ladder has limitations related to the construction of the circuit. These restrictions only concern the objects related to the family of TIMER, FACES, and FUNCTIONS .In practice these types of items shouldn't have no right to the knot of the Circuit functional.



Another limitation concerns the functions related to counters, These are considered output and therefore should always be there at the end of the block.

OPERATION OF COMPILER LADDER

The compiler allows you to manage LADDER PLC cycles using the classical graphical representation of electrical circuits, this greatly simplifies the management of PLC applications. In practice the PLC application is divided to blocks which contain an electrical circuit more or less complex. A block always begins with the elements of INPUT and OUTPUT ends with elements, can be inserted between these elements and processing conditioning (timers, counters, etc..).
A block can be represented in its minimum configuration in the following example:



In this example, the input which may be any bit of memory, determines the set of output that can be any bit of memory. The blocks may be endless, and may contain more complex networks of the previous example.

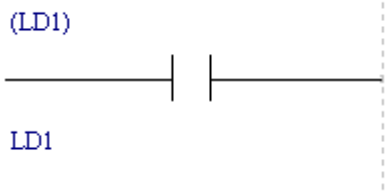
BLOCK MARKER

The MARKER block is used to identify the selected block. This turns solid RED when selecting

BLOCK MARKER

SETTING OF A FUNCTION PLC

A PLC function has parameters which must be set , this is done by clicking the right button on the grid that contains the PLC, and inserting in its own window pop up the various values . The parameters of each function will be described below.



Gird PLC Function of containment

Pop Up window parameters for inclusion

SELECTION OF ONE OR MORE ELEMENTS OF A BLOCK

The selection of one or more elements of a block is a very important feature of the compiler ladder, this because some functions operate only on selected items. An item is selected when the background of the grid that contains it becomes YELLOW. To select a single element of a block simply click with the left mouse button on element concerned, in addition to element is also selected the whole block (BLOCK MARKER the change of color). To select multiple items in a lock hold down the SHIFT key () on the keyboard and click with the left mouse button on the relevant elements, or select the various elements within the rectangular window designed by dragging the MOUSE. For UNCHECK one or more items, simply click the left mouse button in an empty grid.

CANCELLATION AND MOVE SELECTED

CANCELLATION OF SELECTED

Care for the cancellation of one or more elements selected, simply press the Delete key on the keyboard. Deleted items can still be recovered with the Undo button on the menu bar.

MOVE OF THE ELEMENTS SELECTED

This allows you to move the selected items in the grid. To activate the shift button is selected must LOCK / UNLOCK THE ELEMENTS, as it is sufficient to click with the left mouse button on the selected items, hold it down, drag the items to the desired location.

MENU BUTTONS

Following describes all the functionality of the toolbar buttons in the menus related to the compiler LADDER.

Undo 

Retrieves the last I/O element block erased.

LOCK / UNLOCK THE ELEMENTS 

It allows you to lock or unlock the displacement of selected items.

COPY SELECTED ELEMENTS 

The selected items are copied to memory, to the actual copy, simply click with the left mouse button in the desired grid point

ADD BLOCK 

Adds an empty block in the ladder diagram . The block is added after the last.

CLEAR THE BLOCK SELECTED 

The block with the MARKER selected is deleted. The Undo function makes it possible to block recovery.

INSERT A BLOCK AFTER THE SELECTED 

Is inserted into an empty block next to the one that has the MARKER selected.

ADD A ROW TO BLOCK SELECTED 

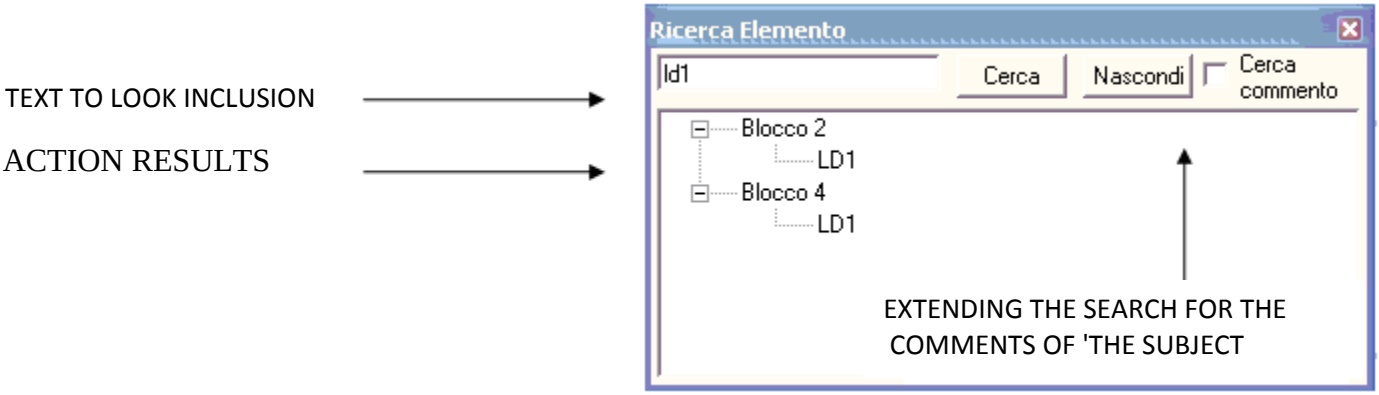
A row is added to the block that has the selected marker.

COPY ALL THE ELEMENTS OF SELECTED BLOCK 

All elements of the selected block are copied into memory. To make an actual copy in the grid, then click with the left mouse button in a grid point where it affects insert the new block that contains the copy of the selected one.

SEARCH FOR AN ITEM 

Search a symbol of an element in all the blocks of application. The results will be presented in the dialog box. Clicking on the result, the block will be brought to the foreground.



FIND A BLOCK

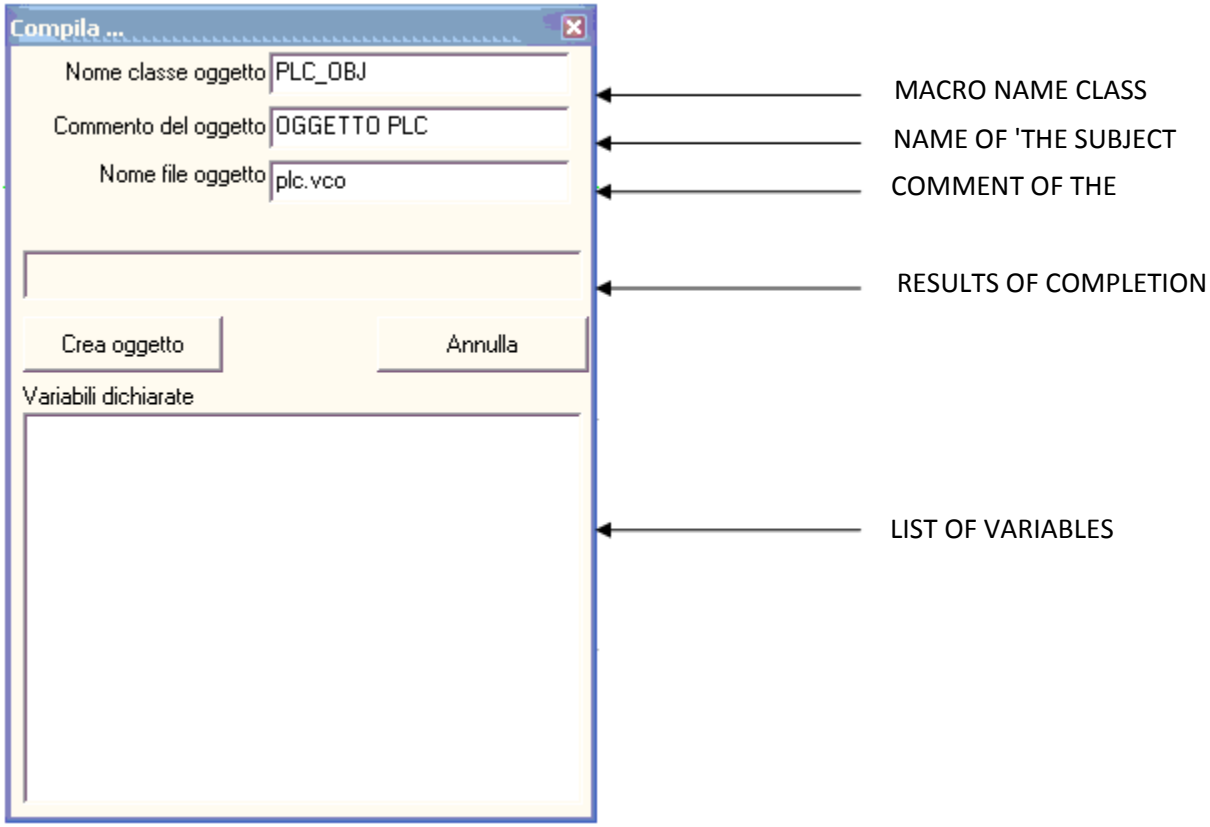
Search a name of a block, the results are similar to the function Search Element.

SAVE FILE PRESENT

Save the 'LADDER current application.

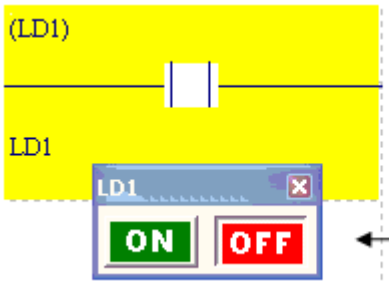
LADDER APPLICATION COMPILER

The ladder application is compiled by checking various errors. If the outcome is positive is generated in an OBJECT MACRO CLASS PLC OBJ with name chosen by the compilation window .To use the PLC application filled out, simply insert the object in the MAIN PAGE of application VTB.



DEBUG USE LADDER

In this mode, displays the status of the various circuits that make up the pattern LADDER. It is useful to make the DEBUG dell application, as you can see both the state will make a force on the variables that compose the circuit. A closed loop or other assets are represented by the color GREEN .To make the FORCING of the various bits or variables, simply click with the left mouse button on the circuit concerned ,and enter the desired value in special POP UP window.

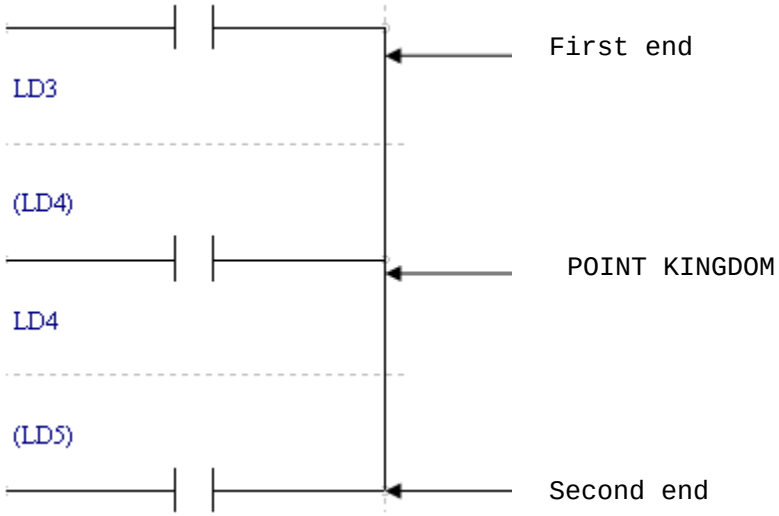


POP UP WINDOW FOR FORCING BIT

LIST INSTRUCTIONS PLC LADDER

CIRCUIT CONTACT

It allows you to combine one or more objects ladder. Click on one end of an object, then click on the other end of the object to merge . Set contact through multiple ends , these will all be united.



LOAD BIT

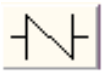
Operation logic block start. Upload the specified bit.

PARAMETERS:

Variable . Name of the variable associated bit (default LDnn). Enter the name of the variable to be loaded. The variable can be existing, in this case it is declared in application VTB

INITIALIZATION:

A Numeric Value: Enter 0 or 1 in the INITIALIZE
Do not initialize: the variable is not initialized with any value
A Variable: enter the name of a variable in the BIT field DEFAULTS

LOAD BIT DENIED 

Operation logic block start. Loads the bit indicated by denying the content.

PARAMETERS:

Variable . Name of the variable associated bit (default LDnn). Enter the name of the variable to be loaded. The variable can be existing, in this case it is declared in application VTB

INITIALIZATION:

A Numeric Value: Enter 0 or 1 in the INITIALIZE
Do not initialize: the variable is not initialized with any value
A Variable: enter the name of a variable in the BIT field DEFAULTS

Output 

Operation logic block end. The logical result of all the previous circuit is transferred to 'exit' s output remains active until the result is TRUE.

PARAMETERS:

Variable . Name of the variable associated bit (default OUTnn). Enter the name of the variable to set. The variable can be existing, in this case it is declared in application VTB

INITIALIZATION:

A Numeric Value: Enter 0 or 1 in the INITIALIZE
Do not initialize: the variable is not initialized with any value
A Variable: enter the name of a variable in the BIT field DEFAULTS

OUTPUT DENIED 

Operation logic block end. The logical result of all the previous circuit is transferred to release denying the content, the output remains active until the result is TRUE DENIED.

PARAMETERS:

Variable . Name of the variable associated bit (default OUTnn). Enter the name of the variable to set. The variable can be existing, in this case it is declared in application VTB

INITIALIZATION:

A Numeric Value: Enter 0 or 1 in the INITIALIZE
Do not initialize: the variable is not initialized with any value
A Variable: enter the name of a variable in the BIT field DEFAULTS

Set Output 

Operation logic block end. Set the relay latching according to the result of the entire circuit prior-logical output remains active until it is operated RESET.

PARAMETERS:

Variable . Name of the variable associated bit (default SETnn). Enter the name of the variable to set. The variable can be existing, in this case it is declared in application VTB

INITIALIZATION:

None

Reset Output 

Operation logic block end. Resets the relay in latching according to the logical result of the entire circuit prior-output remains reset until it is operated in a set.

PARAMETERS:

Variable . Name of the variable associated bit (default RESnn). Enter the name of the variable to set. The variable can be existing, in this case it is declared in application VTB

INITIALIZATION:

None

Ondelay Timer 

Delay to ignition switch output, the output is activated with delay TON to ignition pulse.

PARAMETERS:

Time. Delay time in milliseconds

Variable . Name of the variable associated timer (default TONnn). Enter the name of the variable that contains the time. The variable can be existing, in this case it is declared in application VTB

Type Variable Variable type INT (max 32768 Ms) or LONG. INT variables are faster as management

INITIALIZATION:

A Numeric Value: enter the value in the field in Ms TIME

Do not initialize: the variable is not initialized with any value

A Variable: enter the name of a variable in the TIME

Off Delay Timer 

Delay shutdowns of 'exit' s output is switched off with the TOFF to delay pulse off.

PARAMETERS:

Time.	Delay time in milliseconds
Variable .	Name of the variable associated timer (default TOFFnn). Enter the name of the variable that contains the time. The variable can be existing, in this case it is declared in application VTB
Type Variable	Variable type INT (max 32768 Ms) or LONG. INT variables are faster as management

INITIALIZATION:

A Numeric Value:	enter the value in the field in Ms TIME
Do not initialize:	the variable is not initialized with any value
A Variable:	enter the name of a variable in the TIME

Tp Timer 

Activate output for the time TP, the output is activated on the rising edge of the pulse, for the time indicated.

PARAMETERS:

Time.	Activation time in milliseconds
Variable .	Name of the variable associated timer (default TPnn). Enter the name of the variable that contains the time. The variable can be existing, in this case it is declared in application VTB
Type Variable	Variable type INT (max 32768 Ms) or LONG. INT variables are faster as management

INITIALIZATION:

A Numeric Value:	enter the value in the field in Ms TIME
Do not initialize:	the variable is not initialized with any value
A Variable:	enter the name of a variable in the TIME

Rising Edge 

Activate output on the rising edge of the pulse.

PARAMETERS:

Variable .	Name of the variable associated bit (default UPnn). Enter the name of the variable support. The variable can be existing, in this case it is declared in application VTB

INITIALIZATION:

A Numeric Value:	Enter 0 or 1 in the INITIALIZE
Do not initialize:	the variable is not initialized with any value
A Variable:	enter the name of a variable in the BIT field DEFAULTS

Falling Edge 

Activate output on the falling edge of the pulse.

PARAMETERS:

Variable . Name of the variable associated bit (default UPnn). Enter the name of the variable support. The variable can be existing, in this case it is declared in application VTB

INITIALIZATION:

A Numeric Value: Enter 0 or 1 in the INITIALIZE
Do not initialize: the variable is not initialized with any value
A Variable: enter the name of a variable in the BIT field DEFAULTS

Up Counter 

Increment the counter.

PARAMETERS:

Variable . Name of the variable associated counter variable (default CUPnn). Enter the name of the variable Support.The variable can be existing, in this case it is declared in application VTB

INITIALIZATION:

None

Down Counter 

Decrements the counter.

PARAMETERS:

Variable . Name of the variable associated counter variable (default CDNnn). Enter the name of the variable Support .The variable can be existing, in this case it is declared in application VTB

Type Variable Type of the variable CHAR INT etc.

INITIALIZATION:

None

Reset Counter 

Reset the counter.

PARAMETERS:

Variable . Name of the variable associated counter variable (default CRESnn). Enter the name of the variable to reset. The variable can be existing, in this case it is declared in application VTB

Type Variable Type of the variable CHAR INT etc.

INITIALIZATION:

None

COMPARISON OF TWO VARIABLES

Compare the content of two variables.

PARAMETERS

Variable 1 Name of the first variable to be compared (default COMP_1nn). The variable can be any, in this case it is declared in application VTB

Type var1 type variable1 CHAR or INT etc..

Variable 2 The name of the second variable to be compared (default COMP_2nn). The variable can be any,in this case it is declared in application VTB

Type var2 type variable2 CHAR or INT etc..

Type Comparison Conditions for comparison.

= True condition if variable1 equal to variable2

<> True condition if variable1 to variable2 different

> True condition if variable1 variable2 more

<True condition if less than variable1 variable2

> = True condition if greater than or equal to variable2 variable1

<= True condition if less than or equal to variable2 variable1

INITIALIZATION:

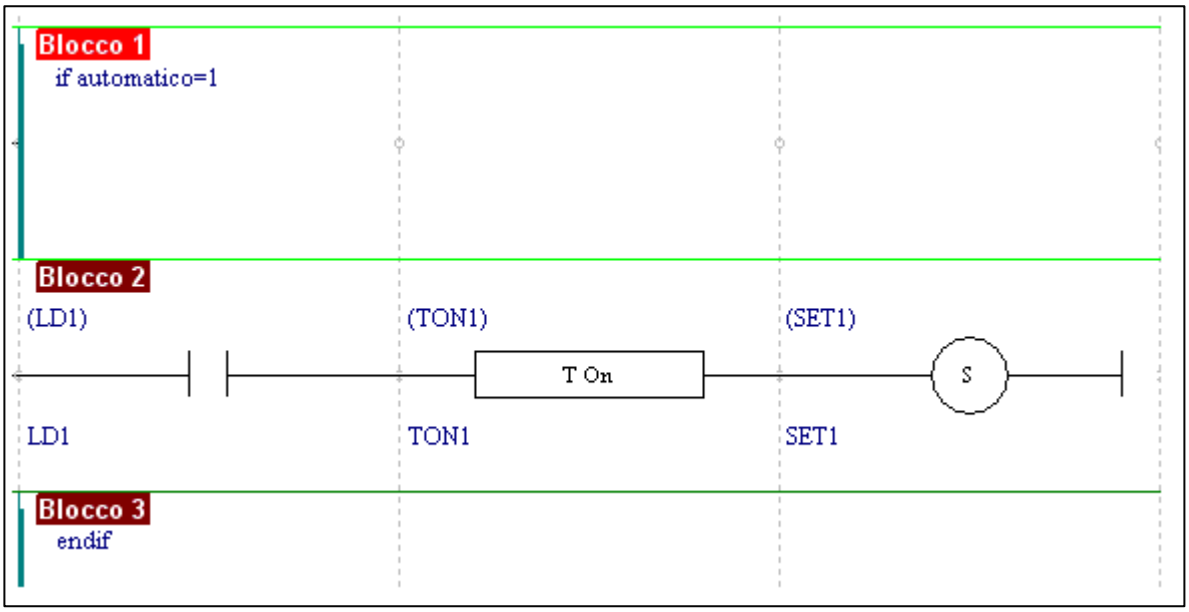
A Numeric Value: enter the value in the field DEFAULTS
Do not initialize: the variables are not initialized with any value
A Variable: enter the name of a variable in the INITIALIZE

WRITING CODE VTB TXT

The TXT element allows you to write code to VTB internal application LADDER. Therefore they can be used all the instructions that recognizes VTB .This function is very powerful because it allows to optimize the LADDER application, for example by inserting cycles ENDIF IF that affect one or more blocks LADDER. In example shown below, the block 2 is scanned only when the variable AUTOMATIC is equal to 1.This in effect creates an optimization of the speed of execution of the PLC cycle, as it may exclude entire sections that are related to a condition.

PARAMETERS
No one

INITIALIZATION
None



COMMENTS ALL INSIDE A BLOCK REM

The REM element allows you to write comments all inside of a block. Since this is an OBJECT must necessarily be was added at the beginning or end of a block. Comments may contain text that is ignored, however, at the time of compilation.
To enter the text you just click with the right mouse button on REM inserted object, and then enter the title and description.

PARAMETERS
No one

INITIALIZATION
None

TEXT LIST INSTRUCTIONS PLC

PLC_LD

Syntax	Operation	Function	Example
PLC_LD Var	Variable Bit declared in MAIN. In the case of the compiler LADDER he thinks the statement	Logical operation start	BIT type variable declaration Var1 → var.2 (original variable) **PLC_LD var1**

PLC_LDN

Syntax	Operation	Function	Example
PLC_LDN Var	Variable Bit declared in MAIN. In the case of the compiler LADDER he thinks the statement	Start logical operation denied	BIT type variable declaration Var1 → var.2 (original variable) **PLC_LDN var1**

PLC_AND

Syntax	Operation	Function	Example
PLC_AND Var	Variable Bit declared in MAIN. In the case of the compiler LADDER he thinks the statement	Logical AND operation	Variable declaration Var1 → var.2 Var2 → var.1 PLC_LD Var2 PLC_AND Var1

PLC_ANDN

Syntax	Operation	Function	Example
PLC_ANDN Var	Variable Bit declared in MAIN. In the case of the compiler LADDER he thinks the statement	Logical AND DENIED	Variable declaration Var1 → var.2 Var2 → var.1 PLC_LD Var2 PLC_ANDN Var1

PLC_OR

Syntax	Operation	Function	Example
PLC_OR Var	Variable Bit declared in MAIN. In the case of the compiler LADDER he thinks the statement	Logical OR operation	Variable declaration Var1 → var.2 Var2 → var.1 PLC_LD Var2 PLC_OR Var1

PLC_ORN

Syntax	Operation	Function	Example
PLC_ORN Var	Variable Bit declared in MAIN. In the case of the compiler LADDER he thinks the statement	Logical operation OR DENIED	Variable declaration Var1 → var.2 Var2 → var.1 PLC_LD Var2 PLC_ORN Var1

PLC_PUSH

Syntax	Operation	Function	Example
PLC_PUSH	ACC	PUSH dell ACC in STACK	Variable declaration Var1 → var.2 Var2 → var.1 PLC_LD Var2 PLC_OR Var1 PLC_PUSH

PLC_ANDS

Syntax	Operation	Function	Example
PLC_ANDS	ACC	Logical AND STACK with the ACC	Variable declaration Var1 → var.2 Var2 → var.1 Var3 → var.3 PLC_LD var2 PLC_OR var1 PLC_PUSH PLC_LD var3 PLC_ANDS

PLC_ORS

Syntax	Operation	Function	Example
PLC_ORS	ACC	Logical OR STACK with the ACC	Variable declaration Var1 → var.2 Var2 → var.1 Var3 → var.3 PLC_LD var2 PLC_OR var1 PLC_PUSH PLC_LD var3 PLC_ORS

PLC_OUT

Syntax	Operation	Function	Example
PLC_OUT	Variable Bit declared in MAIN. In the case of the compiler LADDER he thinks the statement	Sends the contents of the accumulator in the BIT output	Variable declaration Var1 → var.2 PLC_LD var1 PLC_OUT

PLC_OUTN

Syntax	Operation	Function	Example
PLC_OUTN	Variable Bit declared in MAIN. In the case of the compiler LADDER he thinks the statement	Send the contents of DENIED in BIT accumulator output	Variable declaration Var1 → var.2 PLC_LD var1 PLC_OUTN

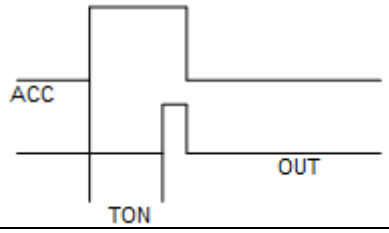
PLC_SET

Syntax	Operation	Function	Example
PLC_SET	Variable Bit declared in MAIN. In the case of the compiler LADDER he thinks the statement	Set the relay in SELF based on the content of ACCUMULATOR	Variable declaration Var1 → var.2 PLC_LD var1 PLC_SET

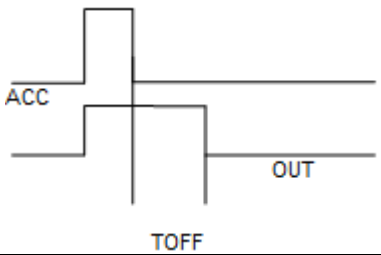
PLC_RES

Syntax	Operation	Function	Example
PLC_RES	Variable Bit declared in MAIN. In the case of the compiler LADDER he thinks the statement	Resets the relay in SELF based on the content of ACCUMULATOR	Variable declaration Var1 → var.2 PLC_LD var1 PLC_RES

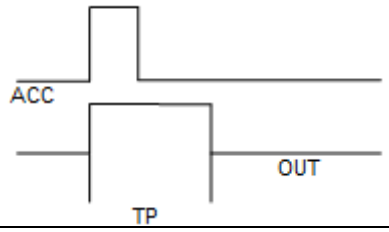
PLC_TON

Syntax	Operation	Function	Example
PLC_TON	Variable vector of positions 2 or UINT or LONG declared in MAIN. In the case of the compiler LADDER he thinks the declaration and all INITIALIZATION. NOTES: Not using the compiler LADDER need to initialize the variable in the INIT MAIN. Where VARIABLE (0) contains the millisecond value of the TARGET TIMER, VARIABLE (1) contains the starting value and the present value	Delay to ignition switch output. The output is activated with delay TON to ignition pulse. The output can be any one of the blocks PLC_OUT PLC_OUTN PLC_SET PLC_RES	Variable declaration Var1 → var.2 Var2 → var.1 Tempo(2) as long Initialize TIMER Tempo(0) = 1000 ‘ 1 sec Tempo(1)=1000 ‘ 1 sec PLC_LD var1 PLC_TON tempo() PLC_SET var2 

PLC_TOFF

Syntax	Operation	Function	Example
PLC_TOFF	Variable vector of positions 2 or UINT or LONG declared in MAIN. In the case of the compiler LADDER he thinks the declaration and all INITIALIZATION. NOTES: Not using the compiler LADDER need to initialize the variable in the INIT MAIN. Where VARIABLE (0) contains the millisecond value of the TARGET TIMER, VARIABLE (1) contains the starting value and the present value	Off Delay of output. The output is switched off with the TOFF to delay pulse off. The output can be any one of the blocks PLC_OUT PLC_OUTN PLC_SET PLC_RES	Variable declaration Var1 → var.2 Var2 → var.1 Tempo(2) as long Initialize TIMER Tempo(0) = 1000 ‘ 1 sec Tempo(1)=0 PLC_LD var1 PLC_TOFF tempo() PLC_SET var2 

PLC_TP

Syntax	Operation	Function	Example
PLC_TP	Variable vector of positions 2 or UINT or LONG declared in MAIN. In the case of the compiler LADDER he thinks the declaration and all INITIALIZATION. NOTES: Not using the compiler LADDER need to initialize the variable in the INIT MAIN. Where VARIABLE (0) contains the millisecond value of the TARGET TIMER, VARIABLE (1) must be CLEARED	Activate output for the time TP. The output is activated on the rising edge of the accumulator. The output can be any one of the blocks PLC_OUT PLC_OUTN PLC_SET PLC_RES	Variable declaration Var1 → var.2 Var2 → var.1 Tempo(2) as long Initialize TIMER Tempo(0) = 1000 ‘ 1 sec Tempo(1)=0 ‘ reset PLC_LD var1 PLC_TP tempo() PLC_SET var2 

PLC_UP

Syntax	Operation	Function	Example
PLC_UP	CHAR or UCHAR variable declared in MAIN. In the case of the compiler LADDER he thinks the statement. The variable must be initialized with the INIT MAIN DEFAULT value of the STATE	Turning on the FRONT OUTPUT RISE of BATTERY. The output can be any one of the blocks PLC_OUT PLC_OUTN PLC_SET PLC_RES	Var1 → var.2 Var2 → var.1 Old_stato as char Initialization STATE Old_stato=0 PLC_LD var1 PLC_UP old_stato PLC_OUT

PLC_DOWN

Syntax	Operation	Function	Example
PLC_DOWN	CHAR or UCHAR variable declared in MAIN. In the case of the compiler LADDER he thinks the statement. The variable must be initialized with the INIT MAIN DEFAULT value of the STATE	OUTPUT FACE DOWN on the activation of the ACC. The output can be any one of the blocks PLC_OUT PLC_OUTN PLC_SET PLC_RES	Variable declaration Var1 → var.2 Var2 → var.1 Old_stato as char Initialization STATE Old_stato=0 PLC_LD var1 PLC_DOWN old_stato PLC_OUT

PLC_CUP

Syntax	Operation	Function	Example
PLC_CUP	Variable LONG - INT - UINT - CHAR - UCHAR declared in MAIN. In the case of the compiler LADDER he thinks the statement. The variable can be pre-set at any TASK VTB.	UP counter. The count is INCREASED if the ACC is ON	Variable declaration Var1 → var.2 Contad long PLC_LD var1 PLC_CUP contad

PLC_CDOWN

Syntax	Operation	Function	Example
PLC_CDOWN	Variable LONG - INT - UINT - CHAR - UCHAR declared in MAIN. In the case of the compiler LADDER he thinks the statement. The variable can be pre-set at any TASK VTB.	UP counter. The count is INCREASED if the ACC is ON	Variable declaration Var1 → var.2 Contad long PLC_LD var1 PLC_CDOWN contad

PLC_CRES

Syntax	Operation	Function	Example
PLC_CRES	Variable LONG - INT - UINT - CHAR - UCHAR declared in MAIN. In the case of the compiler LADDER he thinks the statement.	Reset based on the contents of the accumulator counter the UP or DOWN	Variable declaration Var1 → var.2 Conta long PLC_LD var1 PLC_CRES conta

PLC_CMP

Syntax	Operation	Function	Example
PLC_CMP	Variable to compare LONG - INT - UINT - CHAR - UCHAR declared in the MAIN or CONSTANT inserted directly. In the case of the compiler LADDER he thinks the statement. NOTES: PLC_CMP phase begins for comparison (in LADDER is not present) must be followed by one of the following instructions: PLC_EQ PLC_NE PLC_GT PLC_GE PLC_LT PLC_LE	Comparison of two variables	Variable declaration Var1 int Var2 int PLC_CMP var1 PLC_EQ var2 (see var1=var2 accumulatore=1)

PLC_EQ

Syntax	Operation	Function	Example
PLC_EQ	Variable Comparison LONG - INT - UINT - CHAR - UCHAR declared in the MAIN or CONSTANT inserted directly. In the case of the compiler LADDER he thinks the statement. NOTES: The education PLC_EQ must always be preceded by: PLC_CMP Var1	Comparison if EQUAL. If the two variables are the SAME content of accumulator is set to 1 (TRUE)	Variable declaration Var1 int Var2 int PLC_CMP var1 PLC_EQ var2 (see var1=var2 accumulatore=1)

PLC_NE

Syntax	Operation	Function	Example
PLC_NE	Variable Comparison LONG - INT - UINT - CHAR - UCHAR declared in the MAIN or CONSTANT inserted directly. In the case of the compiler LADDER he thinks the statement. NOTES: The education PLC_NE must always be preceded by: PLC_CMP Var1	Comparison if DIFFERENT. If the two variables are DIFFERENT the content of accumulator is set to 1 (TRUE)	Variable declaration Var1 int Var2 int PLC_CMP var1 PLC_NE var2 (se var1=var2 accumulatore=1)

PLC_GT

Syntax	Operation	Function	Example
PLC_GT	Variable Comparison LONG - INT - UINT - CHAR - UCHAR declared in the MAIN or CONSTANT inserted directly. In the case of the compiler LADDER he thinks the statement. NOTES: The education PLC_GT must always be preceded by: PLC_CMP Var1	Comparison if AJEURE. If the variable VAR1 VAR2 is GREATER than the content of accumulator is set to 1 (TRUE)	Variable declaration Var1 int Var2 int **PLC_CMP var1** **PLC_GT var2**

PLC_GE

Syntax	Operation	Function	Example
PLC_GE	Variable Comparison LONG - INT - UINT - CHAR - UCHAR declared in the MAIN or CONSTANT inserted directly. In the case of the compiler LADDER he thinks the statement. NOTES: The education PLC_GE must always be preceded by: PLC_CMP Var1	Comparison if Greater than. If the variable VAR1 to VAR2 is greater than the content of accumulator is set to 1 (TRUE)	Variable declaration Var1 int Var2 int **PLC_CMP var1** **PLC_GE var2**

PLC_LT

Syntax	Operation	Function	Example
PLC_LT	Variable Comparison LONG - INT - UINT - CHAR - UCHAR declared in the MAIN or CONSTANT inserted directly. In the case of the compiler LADDER he thinks the statement. NOTES: The education PLC_LT must always be preceded by: PLC_CMP Var1	Comparison if LESS. If the variable VAR1 VAR2 is LESS than the content of accumulator is set to 1 (TRUE)	Variable declaration Var1 int Var2 int **PLC_CMP var1** **PLC_LT var2**

PLC_LE

Syntax	Operation	Function	Example
PLC_LE	Variable Comparison LONG - INT - UINT - CHAR - UCHAR declared in the MAIN or CONSTANT inserted directly. In the case of the compiler LADDER he thinks the statement. NOTES: The education PLC_LE must always be preceded by: PLC_CMP Var1	Comparison Less than though. If the variable VAR1 VAR2 is less than or equal to the contents of the accumulator is set to 1 (TRUE)	Variable declaration Var1 int Var2 int **PLC_CMP var1** **PLC_LE var2**