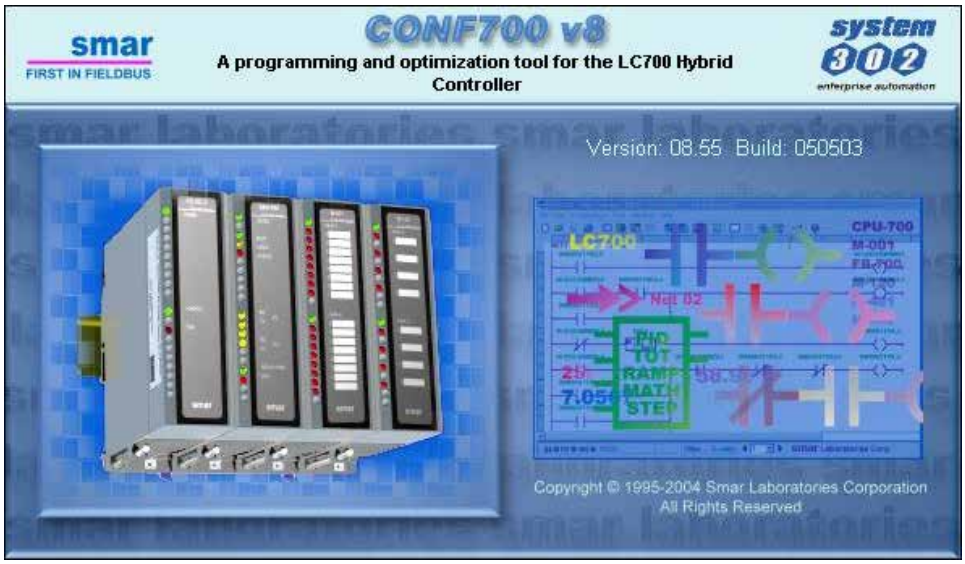
- LC700

smar
First in Fieldbus

CONFIGURATION MANUAL

LC700 Programable Controller



MAR / 07

LC700

FOUNDATION™

LC700SWME

smar

www.smar.com

Specifications and information are subject to change without notice.
Up-to-date address information is available on our website.

web: www.smar.com/contactus.asp

INTRODUCTION

This manual is divided in three parts:

- **Ladder Logic**: The control elements of a control strategy available in the CONF700 and used by the CPU-700 are described in this chapter.

- **Function Blocks:** We present detailed descriptions of all function blocks available in the CONF700 and used by the CPU-700.

- **CONF700**: We describe Smar's Software CONF700. This is the application used to configure the hardware of a control system (I/O Modules, Power Supplies, CPU, etc), mplement ladder logic (including ladder network elements and function blocks)

We suggest the user to read initially Chapters 1 and 2 and next go to Chapter 3 that describes clearly how to implement the elements described in the first two Chapters. However, the user is free to start reading from Chapter 3 prior to the other ones and consult Chapters 1 and 2 any time it is necessary while reading of Chapter 3. Chapter 3 deals with description of the Smar's CONF700 Software that is part of the LC700 System.

Chapter 4 **Troubleshooting** brings solutions to problems found by users by configure the LC700 system.

This manual has practical examples that describe step-by-step how to set strategies of control.

These examples were included in order to make easier the user's understanding of the system.

These applications are distributed along this manual.

NOTE

The CONF700 version 8 only reads and converts configurations made in previous versions of the CONF700. The CONF700 version 8 only supports CPU-700-D3, CPU-700-D3R, CPU-700-E3, CPU-700-E3R, RIO-700-D3 and RIO-700-E3.

The configuration file updates versions previous to PL4 will cause changes in the MODBUS Addresses, however the configuration files higher than PL4 will not have these changes.

NOTE

This document is a description of all function blocks and logic elements (ladder) implemented in the logic programmable controller (LC700). Besides this document presents a description of how to configure and edit ladder networks through Smar's CONF700. This document also describes details of this software.

Smar reserves it the right to change any part of this manual without prior notice.

Note that different versions of the LC700 have different types of data, function blocks and generic characteristics. The last version of the LC700 is always an update of the old manual without prior notice. It means it will have all characteristics included, old and new ones.

TABLE OF CONTENTS

Chapter 1

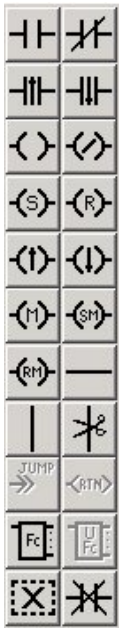
NETWORK ELEMENTS (LADDER ELEMENTS) AND TOOLS

This section will help the user understand the meaning of the network ladder elements and the network tools.

- The Network Elements
- The Label Dialog Box

Network Elements

As mentioned before, CONF700 uses symbols and notations defined in the standard IEC-61131-3.



Network Tool Box

Definitions for the Network Tool Box Elements

Normally Open Contact

The state of the left link is copied to the right link if the state of the Boolean variable is ON. Otherwise, the state of the right link is OFF.

Normally Closed Contact

The state of the left link is copied to the right link if the state of the Boolean variable is OFF. Otherwise, the state of the right link is OFF.

Positive Transition-Sensing Contact

The state of the right link is ON from one evaluation of this element to the next when a transition of the associated variable from OFF to ON is sensed at the same time that the state of the left link is ON. The state of the right link shall be OFF at all other times.

Negative Transition-Sensing Contact

The state of the right link is ON from one evaluation of this element to the next when a transition of the associated variable from ON to OFF is sensed at the same time the state of the left link is ON. The state of the right link shall be OFF at all other times.

Coil

The state of the left link is copied to the associated Boolean variable and to the right link.

Negated Coil

The state of the left link is copied to the right link. The inverse of the state of the left link is copied to the associated Boolean variable, that is, if the state of the left link is OFF, then the state of the associated variable is ON, and vice versa.

Set (Latch) Coil

The associated Boolean variable is set to the ON state when the left link is in the ON state, and remains set until reset by a RESET coil.

Reset (Unlatch) Coil

The associated Boolean variable is reset to the OFF state when the left link is in the ON state, and remains reset until set again by a SET coil.

Retentive (Memory) Coil

The associated Boolean Variable will be retentive to the memory.

Note that the action of this coil is identical to Coil, except that the associated Boolean variable is automatically declared to be in retentive memory without the explicit use of the VAR RETAIN declaration defined in the initialization of variables in IEC-61131-3 standard.

Set Retentive (Memory) Coil

The associated Boolean variable is set to the ON state when the left link is in the ON state, and remains set until reset by a RESET coil. The associated Boolean Variable will be retentive to the memory.

Note that the action of this coil is identical to Set (Latch) Coil, except that the associated Boolean variable is automatically declared to be in retentive memory without the explicit use of the VAR RETAIN declaration defined in the initialization of variables in IEC-1131-3 standard.

Reset Retentive (Memory) Coil

The associated Boolean variable is reset to the OFF state when the left link is in the ON state, and remains reset until set by a SET coil. The associated Boolean Variable will be retentive to the memory.

Note that the action of this coil is identical to RESET (Unlatch) Coil, except that the associated

Boolean variable is automatically declared to be in retentive memory without the explicit use of the VAR RETAIN declaration defined in the initialization of variables in IEC-61131-3 standard.

Positive Transition-Sensing Coil

The state of the associated Boolean variable is ON from one evaluation of this element to the next when a transition of the left link from OFF to ON is sensed. The state of the left link is always copied to the right link.

Negative Transition-Sensing Coil

The state of the associated Boolean variable is ON from one evaluation of this element to the next when a transition of the left link from ON to OFF is sensed. The state of the left link is always copied to the right link.

Horizontal Connecting Line

Use this tool to draw a connecting line from left to right in the marked cell.

Vertical Connecting Line

Use this tool to draw a connecting line from the right side of the marked cell downward.

Eliminate Vertical Connecting Line from the Focused Cell

This tool eliminates the vertical connection line. Place the selection box in the element that has the vertical line the user whishes to eliminate.

Function Blocks

Use this tool to open a Dialog Box for choosing the desired Built-in-Function.

User Function

Use this tool to open a Dialog Box for choosing available User-Functions.

Jump to a Network

If more than one Network is available, it will open a Dialog Box for choosing the desired Network.

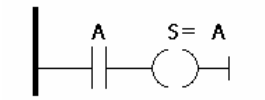
Return for the Last Jump

Use this tool to return to the next executable cell preceding the last Jump. If no Jump has being used, it will then be ignored.

Boolean Logic

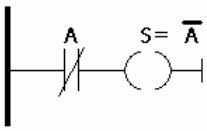
The association of relays and coils creates boolean functions. Below a brief summary of these functions and Boolean Algebra is presented.

Normally Open Relay

Diagram	State Table	
	A	S
	0	0
	1	1

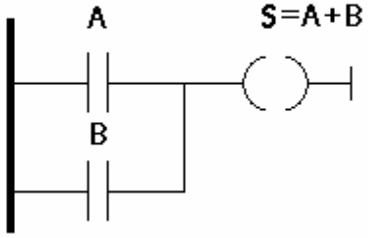
When the state of A changes from 0 to 1, the contact A closes and the flow goes from the power rail on the left to the right powering the coil S.

Normally Closed Relay

Diagram	State Table	
	A	S
	0	1
	1	0

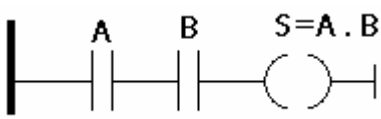
The operation of a Normally Closed Relay is the same to that of a Normally Open Relay, except backwards. That is, when the state of A changes from 0 to 1, the contact A opens and the current does not flow from the power rail to the right. (through the contact A circuit)

Logical Function OR

Diagram	State Table		
	A	B	S
	0	0	0
	0	1	1
	1	0	1
	1	1	1

Relays A and B are normally open. With the association of both we implement the OR function. The coil is powered when any of the two relays is closed.

Logical Function AND

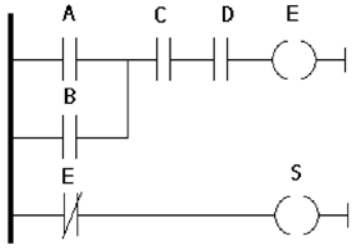
Diagram	State Table		
	A	B	S
	0	0	0
	0	1	0
	1	0	0
	1	1	1

Relays A and B are normally open. The coil S is powered when A and B are equal to 1 at the same

time. Otherwise, power will not flow from the left side (power rail) to the right side.

Boolean Equations

By using relays and coils it is possible to implement boolan functions. For example, consider the diagram below:



The S output depends on the state of the relays A, B, C, D and on the coil E. E depends on the values of A, B, C and D. So:

$$E = (A + B).(C).D$$
$$S = \overline{E}$$

Boolean Algebra

Boolean equations as shown above may become very complicated; however the result might be simplified using the boolean algebra. Below is a summary of properties of the Boolean Algebra.

1	A.1= A
2	A.0= 0
3a 3b	A.A= A $A\overline{A} = 0$
4a 4b	A+ $\overline{A}$ =1 A+A=A
5	A+1=1
6	A.B+A.C=A.(B+C)
7	A+A.B= A
8	A.(B+C)= A.B+A.C
9a 9b	$\overline{A + B} = \overline{A}.\overline{B}$ $\overline{A.B} = \overline{A} + \overline{B}$

When these expressions become too complex we suggest that you use the Karnaugh map in order to simplify them. This information is easily found on any Digital Electronics Book.

Chapter 2

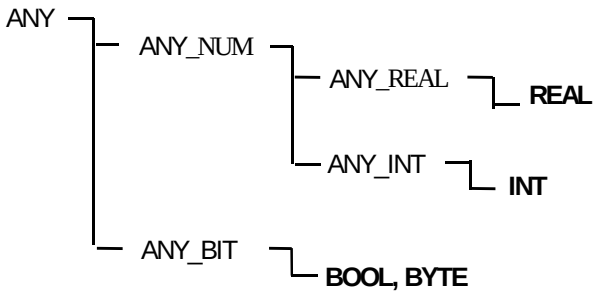
FUNCTION BLOCKS

Introduction

This is a complete and updated reference of the Function Blocks (FB) supported by the LC700 CPU. Here we present block diagrams showing inputs, outputs, configuration parameters and internal variables. It also includes detailed explanations of each block, how they work, how to configure each one of them and a few examples are presented in order to help with the user's understanding and utilization.

Many times, an input or output will be classified as ANY, ANY_NUM, ANY_BIT, ANY_REAL or ANY_INT. If input is ANY_NUM it means it might be connected to either a REAL or INT output. For a better explanation, see the table below:

Reference	Data Type	Number of bits	Version
BOOLEAN	Boolean	1	1.xx or superior
INT	Integer	16	2.xx or superior
REAL	Float	32	2.xx or superior
WORD	String	16	2.xx or superior



If the user tries to set two outputs of a function block having variables of different data types, for example, adding an integer to a float, the CONF700 will not allow this setting. As the first block variable is selected, it is expected that all other inputs have the same data type of this variable.

During this configuration of inputs and outputs CONF700 asks the user to inform variable data types that should be set when the manual describes it as ANY_XX.

Each function block has a table where all inputs, outputs, parameters and variables of each block are shown.

I - Inputs: It is a variable from another FB or from an I/O card
P - Parameter: User's configurations.
O - Outputs: Variables resulting from processing inside each block.
V - Variables: Auxiliary variables of block algorithms.

Information about using period "." and comma "," for the function blocks:
The format of writing the numeric data (using "." and ",") should be according to the local configuration of the computer.

EN Input And ENO Output

Every function has an EN input and an ENO output.

EN input is set to enable the function block that should be processed. If EN is false, all outputs change to zero and the FB is not executed.

ENO changes to true logic to indicate the function was successfully executed without troubles.

Available Function blocks in alphabetic order

The table below shows all the available function blocks.

FUNCTION NAME	DESCRIPTION
ABS	Absolute Value
ACC	Pulse Accumulator
ACC_N	Pulse Accumulator
ADD	Addition
ARAMP	Automatic Up And Down Ramp
BTB	BYTE_TO_BITS Conversion
BTI	BCD_TO_INT Conversion
BWL	Bit wise Logic
CTD	Counter Down
CTU	Counter Up
CTU1	Counter Up
DIV	Division
EQ	Equality
FIFO	First In First Out
GE	Decreasing Monotonic Sequence
GT	Decreasing Sequence
ICT	Integer Constants
ITB	INT-TO-BCD Conversion
ITR	Conversion Int To Real
LE	Increasing Monotonic Sequence
LIN	Linearization
LMT	Limiter
LT	Increasing Sequence
MATH1	Multivariable Equations
MAX	Maximum
MIN	Minimum
MOD	Modulo
MUL	Multiplication
MUX	Multiplexer
NE	Inequality
NOT	Bitwise NOT
OSEL	Output Selection
PID	PID Controller
RCT	Real Constants
RTA	Real Time Clock Alarm
RTI	Conversion Real To Int
SEL	Binary Selection
SMPL	Sample Hold With Up And Down
SQR	Square Root
STATUS	System Status
STP	Step Control
SUB	Subtraction
TOF	Timer Off-Delay
TOF1	Timer Off-Delay
TON	Timer On-Delay
TON1	Timer On-Delay
TOT	Totalization
TP	Timer Pulse
TP1	Timer Pulse
TRC	Truncation
XLIM	Cross Limit And Rate-Of-Change

Function Blocks listed by functional groups

Time and Pulse related Functions

TIME/PULSE FUNCTIONS	
MNEMONIC	**DESCRIPTION**
ACC	Pulse Accumulator
ACC_N	Pulse Accumulator
CTU1	Counter Up
TOF1	Timer Off-Delay
TON1	Timer On-Delay
TP1	Timer Pulse
CTD	Counter Down
CTU	Counter Up
TOF	Timer Off-Delay
TON	Timer On-Delay
TP	Timer Pulse
RTA	Real Time Clock Alarm

Data Manipulation Functions

DATA MANIPULATION FUNCTIONS	
MNEMONIC	**DESCRIPTION**
BTB	BYTE_TO_BITS Conversion
BTI	BCD_TO_INT Conversion
BTW	Bitwise Logic
FIFO	First In First Out
ICT	Integer Constants
ITB	INT-TO-BCD Conversion
ITR	Conversion Int To Real
MUX	Multiplexer
NOT	Bitwise NOT
OSEL	Output Selection
RCT	Real Constants
RTI	Conversion Real To Int
SEL	Binary Selection

Mathematical Functions

MATH FUNCTIONS	
MNEMONIC	**DESCRIPTION**
ABS	Absolute Value
ADD	Addition
DIV	Division
MOD	Modulo
MUL	Multiplication
SQR	Square Root
SUB	Subtraction
TRC	Truncate

Comparison Functions

COMPARISON FUNCTIONS	
MNEMONIC	**DESCRIPTION**
EQ	Equality
GE	Decreasing Monotonic Sequence
GT	Decreasing Sequence
LE	Increasing Monotonic Sequence
LMT	Limiter
LT	Increasing Sequence
MAX	Maximum
MIN	Minimum
NE	Inequality

Process Control Functions

PROCESS CONTROL FUNCTIONS	
MNEMONIC	**DESCRIPTION**
ARAMP	Automatic Up And Down Ramp
LIN	Linearization
MATH1	Multivariable Equation
PID	PID Controller
SMPL	Sample Hold With Up And Down
STATUS	System Status
STP	Step Control
TOT	Totalization
XLIM	Cross Limit And Rate-Of-Change

Time and Pulse related Functions

ACC - Pulse Accumulator

Description
This Pulse Accumulator block works with the M-302/M-303/M-304 modules and the main objective of the accumulating input pulses is coming from an external source. Typically one of the inputs from the module is linked to the IN input in the ACC block.

During the control cycle the pulse input module accumulates pulses in a local register in the circuit. At the end of every control cycle the LC700, CPU reads the accumulated amount and automatically clears the internal register for the next cycle (preventing an overflow). When the control logic executes, the ACC block gets the integer number of pulses in the IN input and adds it to an internal accumulator TOT_L and TOT_H and, as can be seen, this accumulator is shared as outputs of the ACC block.

Two actions take place when the CLRA input is high in the ACC block:

- The TOT_L and TOT_H accumulated values are moved to MEM_L and MEM_H register.
- The TOT_L and TOT_H contents are then cleared.

The Q output
This function block can also give the information of "pulse speed" (flow) in a time interval (MP) that can be configured by the user. The Q output will keep showing an updated value of accumulated pulses on each MP time interval.

The parameters TR_ON and TR_OFF are the hysteresis limits for the Q threshold. The THR output will go to high when Q is higher or equal to TR_ON and will go back to low when it is lower or equal to TR_OFF.

Accumulator Mode
The ACC function block can count pulses in the TOT_L and TOT_H registers in two different ways:

- Maximum counting in TOT_L is 32767 and TOT_H represents how many times TOT_L overflowed.This means that the total accumulated pulse is calculated by the following formula:

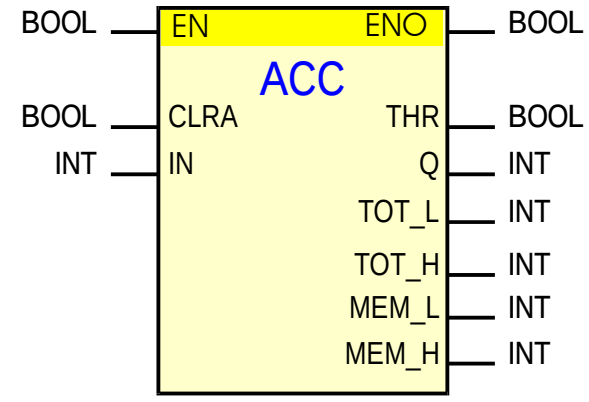
  (TOT_H * 32768) + TOT_L

- Maximum counting in TOT_L is 9999 and TOT_H represents how many times TOT_L overflowed.This means that the total accumulated pulse is calculated by the following formula:

  (TOT_H * 10000) + TOT_L

The accumulation mode is set during the ACC block configuration. The mode set for TOT_L and TOT_H will be extended to MEM_L and MEM_H.

ACC	PULSE ACCUMULATOR	




```
IF EN = 1 THEN
  ENO = 1
  TOT= TOT+ IN
  TACC = TACC + IN
  IF CLRA = 1 THEN
     MEM = TOT
     TOT = 0
  IF TIMER >= MP
    Q = TACC
    IF TACC >= TR_ON
       THR = 1
    IF TACC <= TR_OFF
       THR = 0
    TACC = 0
ELSE
  ENO = 0
```

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	CLRA	STORES TOT TO MEM AND ERASES THE ACCUMULATOR	BOOLEAN
	IN	INPUT PULSE (M-302/M-303/M-304/M-305)	INT
P	CTW	CONTROL WORLD	WORD
	TR_ON	THRESHOLD VALUE TO SET THR OUTPUT TO ON	INT
	TR_OFF	THRESHOLD VALUE TO SET THR OUTPUT TO OFF	INT
	MP	MEASURING PERIOD (THRESHOLD)	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	THR	OUTPUT THR	BOOLEAN
	Q	PULSES ACCUMULATED IN MP PERIOD (FLOW))	INT
	TOT_L	ACTUAL ACCUMULATOR VALUE (LOW WORD)	INT
	TOT_H	ACTUAL ACCUMULATOR VALUE (HIGH WORD)	INT
	MEM_L	MEMORY ACCUMULATOR VALUE (LOW WORD)	INT
	MEM_H	MEMORY ACCUMULATOR VALUE (HIGH WORD)	INT
V	TACC	PULSE ACCUMULATOR	INT
	TMAC	TIME ACCUMULATOR	INT

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

Bit sequence:

Only Configuration								Auxiliary and PRM Passing							
15					10	9	8	7	6	5	4	3	2	1	0

Auxiliary and PRM Passing
Status indication bits:
Bit 0 - Is the EN Boolean input status
Bit 1 - Is the CLRA Boolean input status
Bit 2 - Is the ENO Boolean input status
Bit 3 - Is the THR Boolean input status

Configuration Only
Select the TOTALIZATION MODE (LOWER WORD LIMIT):
Bit 8
0 = TOT ACCUMULATOR (LOW WORD) GOES FROM 0 TO 9999
1 = TOT ACCUMULATOR (LOW WORD) GOES FROM 0 TO 32767

ACC_N - Pulse accumulator

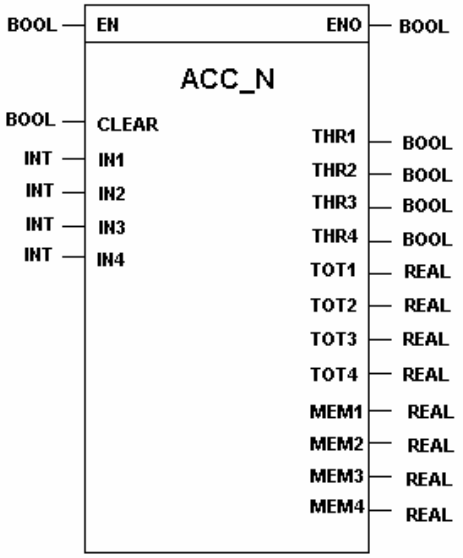
Description
This block accumulates pulses from IN1 and IN2 inputs and displays the totalized value in the TOT1 to TOT4 outputs. CLEAR input equals to 1 clears these counters and the totalized value is moved to the registers MEM1 and MEM2 and the totalization continues.

Factor of Multiplication of Scale
All inputs have an associated parameter to convert the inputs in Engineering Units. If the Factor is equal to 1 the output is given in the 0 to 10000 range.

Hysteresis and limits
Each input has two parameters to define the hysteresis of pulse totalization. Parameters TR_ON1 to TR_ON4 and TR_OFF1 to TR_OFF4 set this hysteresis. Outputs THR1 to THR3 go to a true logic state (1) when the flow is bigger than the values of TR_ON1 to TR_ON4 and go to the false logic state (0) when the flow is smaller or equal than values set on TR_OFF1 to TR_OFF4. Flow is defined as the pulse frequency in a time interval MP (user set).

CLEAR input
Every time there's a transition in the CLEAR input from zero to one, outputs TOT are reset and their respective values are passed to the outputs MEM.

ACC_N	PULSE ACCUMULATOR	



CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOL
	CLEAR	CLEARS TOTALIZATION AND SENDS ALL VALUES TO THE MEM OUTPUTS	BOOL
	IN1	PULSE INPUT 1	INT
	IN2	PULSE INPUT 2	INT
	IN3	PULSE INPUT 3	INT
	IN4	PULSE INPUT 4	INT
P	FACTOR1	SCALE CONVERSION FACTOR (EU) OF INPUT 1.	REAL
	FACTOR2	SCALE CONVERSION FACTOR (EU) OF INPUT 2	REAL
	FACTOR3	SCALE CONVERSION FACTOR (EU) OF INPUT 3.	REAL
	FACTOR4	SCALE CONVERSION FACTOR (EU) OF INPUT 4.	REAL
	TR_ON1	TR_ON HYSTERESIS UPPER LIMIT	INT
	TR_OFF1	TR_OFF HYSTERESIS LOWER LIMIT	
	TR_ON2	TR_ON HYSTERESIS UPPER LIMIT	INT
	TR_OFF2	TR_OFF HYSTERESIS LOWER LIMIT	
	TR_ON3	TR_ON HYSTERESIS UPPER LIMIT	INT
	TR_OFF3	TR_OFF HYSTERESIS LOWER LIMIT	
	TR_ON4	TR_ON HYSTERESIS UPPER LIMIT	INT
	TR_OFF4	TR_OFF HYSTERESIS LOWER LIMIT	
	MP	TIME WHERE PULSES ARE COUNTED	INT

CLASS	MNEM	DESCRIPTION	TYPE
O	ENO	ENABLE OUTPUT	BOOL
	MEM1	VALUE OF ACCUMULATED PULSES	REAL
	MEM2	VALUE OF ACCUMULATED PULSES	REAL
	MEM3	VALUE OF ACCUMULATED PULSES	REAL
	MEM4	VALUE OF ACCUMULATED PULSES	REAL
	THR1	INDICATES IF HYSTERESIS LIMITS WERE REACHED FOR INPUT 1	REAL
	THR2	INDICATES IF HYSTERESIS LIMITS WERE REACHED FOR INPUT 2	REAL
	THR3	INDICATES IF HYSTERESIS LIMITS WERE REACHED FOR INPUT 3	REAL
	THR4	INDICATES IF HYSTERESIS LIMITS WERE REACHED FOR INPUT 4	REAL
V	TACC1	PULSE ACCUMULATOR OF INPUT 1	INT
	TACC2	PULSE ACCUMULATOR OF INPUT 2	INT
	TACC3	PULSE ACCUMULATOR OF INPUT 3	INT
	TACC4	PULSE ACCUMULATOR OF INPUT 4	INT
	TAMC	TIME ACCUMULATOR (TIMER)	INT
	OVRFLW	OVERFLOW INDICATION OF TOT AND MEM	BYTE
	B_THR	BOOLEAN THRESHOLD AND STATUS	BYTE
	INCR1	INCREMENTS FOR CARRY OVER	REAL
	INCR2	INCREMENTS FOR CARRY OVER	REAL
	INCR3	INCREMENTS FOR CARRY OVER	REAL
	INCR4	INCREMENTS FOR CARRY OVER	REAL

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

CTD - Pulse Counter Down

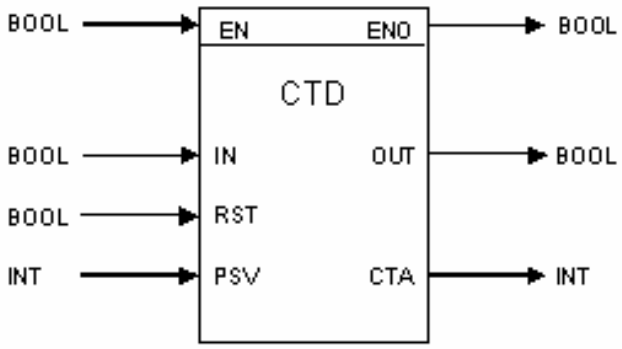
Description
CTD function counts 0 (false) to 1 (true) logic state transitions, for example, an ON- OFF button. While the button is not pressed, the ON operation is not done. When the button is pressed, the state changes to ON, so there was an OFF/ ON transition.

Internal Counter CTA
Every time an ascending transition occurs in the block input, the pulse accumulator (CTA) is decreased by one unit. When the internal counter reaches zero, OUT changes to its true state. The internal counter CTA may be accessed through an output of this function block.

RST (Reset)
If RST is equal to true, the internal clock will reset.

Setting the number of pulses to be counted
Internal parameter PST adjusts the number of pulses this block will count until OUT changes to the true logic state. On CONF700, the user must inform the value of the PST parameter. This value may also be set through the block input PSV. In this case the user should connect the PSV input to another FB output or to an I/O module.

CTD	PULSE COUNTER DOWN (CTD)

	IF PSV THEN
	PT:= PSV
	ELSE
	PT:= PST
	IF EN=1 THEN
	ENO := 1
	IF RST = 1 THEN
	CTA := PT
	ELSE
	IF IN = 1 AND CTA > MIN_INT
	THEN
	CTA := CTA - 1
	IF CTA <= 0 THEN
	OUT = 1
	ELSE
	OUT = 0
	ELSE
	ENO := 0
	OUT := 0
	CTA := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	PULSE INPUT	BOOLEAN
	RST	BLOCK RESET	BOOLEAN
	PSV	THIS INPUT IS CONNECTED TO ADJUST PST EXTERNALLY	INT
P	PST	COUNTER VALUE ADJUSTED THROUGH PARAMETER	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	COUNTER STATE CTA. 0 IF CTA <> 0, 1 IF CTA=0	BOOLEAN
	CTA	PULSE ACCUMULATOR	INT
V	STS	STATUS	WORD

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

CTU - Pulse counter up

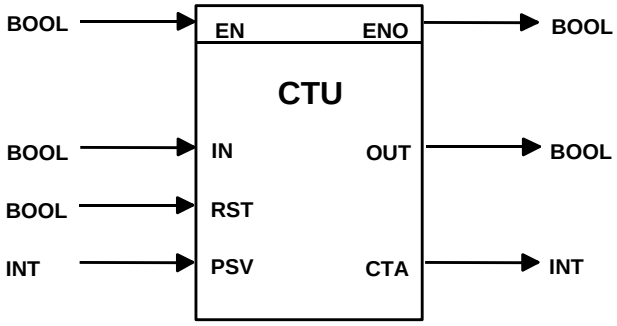
Description
The CTU function counts transitions from 0 (false) to 1 (true).

Internal Counter CTA
Every time an ascending transition occurs in the block input, the pulse accumulator (CTA) is increased by one unit. When the internal counter reaches zero, OUT changes to the true state. The internal counter CTA may be accessed through an output of this function block.

RST (Reset)
If the RST input is TRUE the counter will be cleared.

Setting the number of pulses to be counted
Internal parameter PST adjusts the number of pulses this block will count until OUT changes to the true logic state. On CONF700 the user must set the value of the PST parameter. This value may also be set through block input PSV. In this case the user should connect the PSV input to another FB output or to an I/O module.

CTU	PULSE COUNTER UP

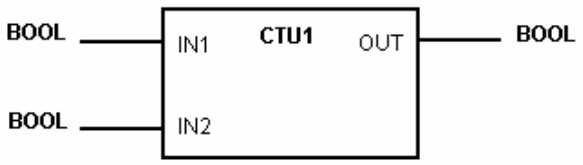
	IF PSV THEN
	PT:= PSV
	ELSE
	PT:= PST
	IF EN=1 THEN
	ENO := 1
	IF RST = 1 THEN
	CTA := 0
	ELSE
	IF IN = 1 AND CTA < MAX_INT THEN
	CTA := CTA + 1
	IF CTA >= PT THEN
	OUT = 1
	ELSE
	OUT = 0
	ELSE
	ENO := 0
	OUT := 0
	CTA := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	PULSE INPUT	BOOLEAN
	RST	BLOCK RESET	BOOLEAN
	PSV	THIS INPUT IS CONNECTED TO ADJUST PST EXTERNALLY	INT
P	PST	COUNTER VALUE ADJUSTED THROUGH PARAMETER	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	OUT=1 WHEN THE VALUE SET IN PST IS REACHED	BOOLEAN
	CTA	PULSE ACCUMULATOR	INT
V	STS	STATUS	WORD

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

CTU1 - Pulse counter up

This function works exactly like the CTU block but it only has two inputs and one output. OUT changes to true when the internal counter (not accessible) reaches the value set by the PST parameter.

CTU1	PULSE COUNTER UP
	

CLASS	MNEM	DESCRIPTION	TYPE
I	IN1	INPUT ENABLE	BOOLEAN
	IN2	PULSE INPUT	BOOLEAN
P	PST	COUNTER VALUE ADJUSTED THROUGH PARAMETER	INT
O	OUT	OUT=1 WHEN THE VALUE SET IN PST IS REACHED	BOOLEAN
V	STS	STATUS	WORD

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

RTA - Real time alarm

Description
This alarm is based on the LC700 CPU real time clock. When the alarm occurs the ALM output goes to true and remains in this condition. The alarm is turned ON according to the date and hour set inside the RTA block. The user should select the time to start the alarm and also the date. The user may select the day of the week, the day of the month, month and year.

RST (Reset)
If a RST (RESET) is applied, the ALM output will return false but not before it remains true for at least one second.

Time Parameter
The user must set the hour desired for the alarm to be active. This hour must be set in the format HR:MIN:SEC, where the parameters HR, MIN and SEC are respectively related to hours, minutes and seconds.

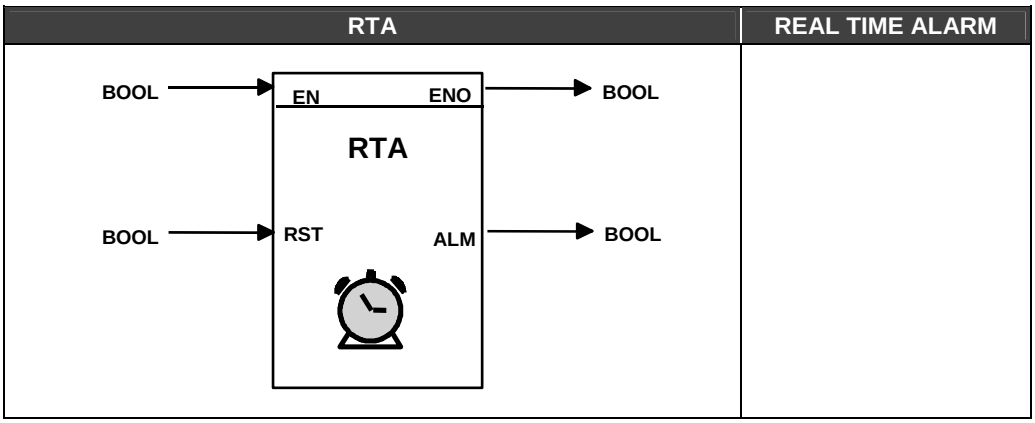
Day Parameter
The user may select this parameter, choosing a specific date. The Day parameter supports two options: Day of The Week and Day of The Month.

If the user sets the Day of The Week parameter, it will have to set by the named day: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday or Saturday.

If the user sets the parameter, Day of The Month, it will have to set in the 2 digits format.

Month Parameter
The user will have to set the month of the year for the alarm to be active in a 2-digit format

Year Parameter
The user will have to set the year for the alarm to be active in a 4-digit format. The desired year must be in the 1980 to 2079 range.



CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	RST	BLOCK RESET	BOOLEAN
P	SEC	SECOND	BYTE
	MIN	MINUTE	BYTE
	HR	HOUR	BYTE
	WD	DAY OF THE WEEK	BYTE
	DAY	DAY	BYTE
	MON	MONTH	BYTE
	YR	YEAR	BYTE
O	ENO	OUTPUT ENABLE	BOOLEAN
	ALM	ALARM OUTPUT	BOOLEAN
V	CTB	CONTROL BYTE	BYTE

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

TOF - Timer off-delay

Description
This FB holds the true state of the input IN for a time interval previously set.

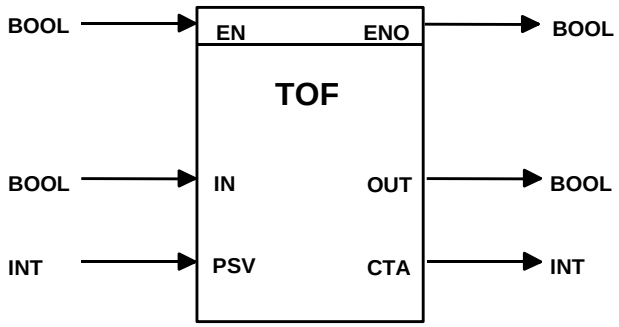
PST Parameter
PST defines the time interval during the true state which is hold. It is set through the PST parameter. This time is given by PST and is multiplied by 10 ms (0.01 s).

If IN changes to true, before OUT changes to false, OUT will stay on true state and the time period will restart when IN goes to false.

Internal Counter CTA
When a true to false logic state transition occurs in the block input, the CTA accumulator would be increased by one unit.

PSV Input
The user may set the PSV input to select the value of the PST parameter externally. In this case, the PSV input must be connected to an FB output or anI/O module.

TOF	TIMER OFF-DELAY

	IF PSV THEN
	PT:= PSV
	ELSE
	PT:= PST
	IF EN=1 THEN
	ENO := 1
	IF IN = 1 THEN
	OUT = 1
	CTA := 0
	ELSE
	IF CTA >= PT THEN
	OUT := 0
	ELSE
	OUT := 1
	CTA := CTA + 1
	ELSE
	ENO := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	PULSE INPUT	BOOLEAN
	PSV	THIS INPUT IS CONNECTED TO ADJUST PST EXTERNALLY	INT
P	PST	PARAMETER PRE ADJUSTED TIMER VALUE THROUGH PST	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	BLOCK OUTPUT	BOOLEAN
	CTA	TIMER PULSE ACCUMULATOR	INT
V	ICT	INITIAL COUNTER TIMER VALUE TO	INT
	STS	STATUS	WORD

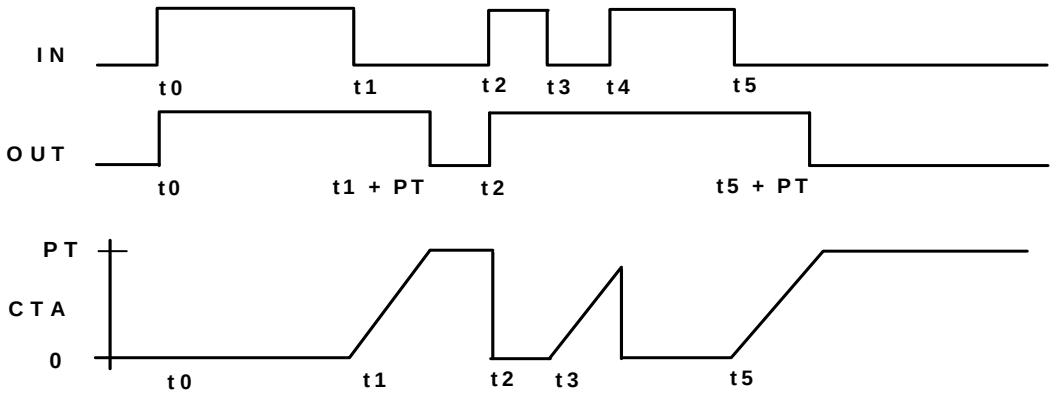
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

Bit sequence for the STS parameter

BIT 8
1= ON, ON-DELAY OCCUR;
0= OFF, OUT=0.

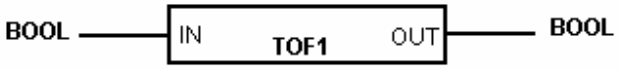
BIT 0
1= COUNTER IS RUNNING;
0 = IS NOT COUNTING.

Timer Off-Delay Function - Timing diagrams



TOF1 - Timer off-delay

It works as the TOF block except it has only one input and one output. The Timer value is only set through internal block parameters.

TOF1	TIMER OFF-DELAY
	

CLASS	MNEM	DESCRIPTION	TYPE
I	IN	PULSE INPUT	BOOLEAN
P	PST	TIMER PRESET VALUE adjusted through the timer parameter	INT
O	OUT	OUT=1 WHEN THE VALUE SET IN PST IS REACHED	BOOLEAN
V	STS	STATUS	WORD

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

TON - Timer on-delay

Description
This function causes a delay in the false to true transition on the OUT output for a specific time interval.

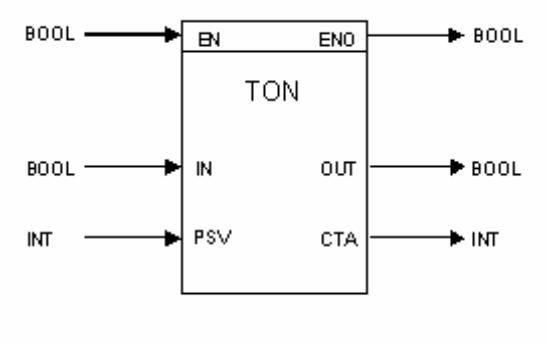
PST Parameter
PST defines the time interval during the true state which is hold. It is set through the PST parameter". PST multiplied by 10 ms (0.01 s) gives this time.

If IN changes to true, before OUT changes to false, OUT will stay on the true state and the time period will restart when IN goes to false.

Internal Counter CTA
When a true to false logic state transition occurs in the block input, the CTA accumulator would be increased by one unit.

PSV Input
The user may set the PSV input to select the value of the PST parameter externally. In this case, the PSV input must be connected to an FB output or an I/O module.

TON	TIMER ON-DELAY

	IF PSV THEN
	PT:= PSV
	ELSE
	PT:= PST
	IF EN=1 THEN
	ENO := 1
	IF IN = 1 AND CTA >= PT THEN
	OUT = 1
	ELSE
	OUT := 0
	IF IN = 0 THEN
	CTA := 0
	ELSE
	CTA := CTA + 1
	ELSE
	ENO := 0
	OUT := 0
	CTA := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	Pulse input	BOOLEAN
	PSV	THIS INPUT IS CONNECTED TO ADJUST PST EXTERNALLY	INT
P	PST	PARAMETER PRE ADJUSTED TIMER VALUE THROUGH PST	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	BLOCK OUTPUT	BOOLEAN
	CTA	TIMER PULSE ACCUMULATOR	INT
V	ICT	INITIAL TIMER VALUE TO THE COUNTER	INT
	STS	STATUS	WORD

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

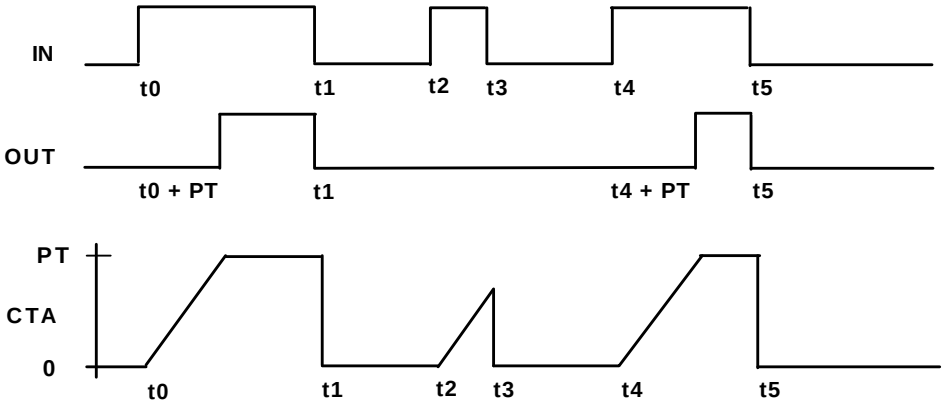
Bit sequence for the STS parameter

Result Bit:
BIT 8
1= ON, ON-DELAY OCCUR;
0= OFF, OUT=0.

Enable Bit
BIT 0
1= COUNTER IS RUNNING;

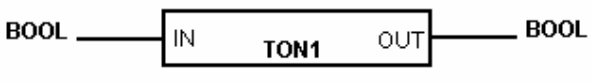
Timer On-Delay Function - Timing diagrams

***OBS.: Parameter BAS has no effect at moment!**



TON1 - Timer on-delay

It works as the TON block except it has only one input and one output. The Timer value is only set through internal block parameters.

TON1	TIMER ON-DELAY
	

CLASS	MNEM	DESCRIPTION	TYPE
I	IN	PULSE INPUT	BOOLEAN
P	PST	PARAMETER PRE ADJUSTED TIMER VALUE THROUGH PST	INT
O	OUT	OUT=1 WHEN THE VALUE SET IN PST IS REACHED	BOOLEAN
V	STS	STATUS	WORD

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

TP - Timer pulse

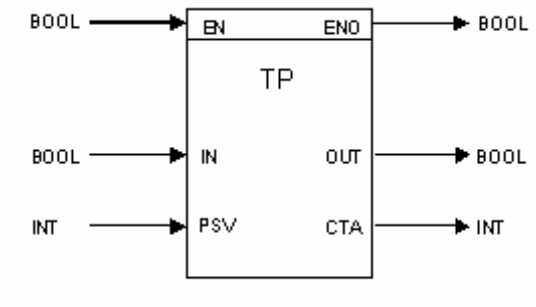
Description
This FB generates a pulse with fixed duration on the OUT output for each rising transition (false to true) in the IN input.

Setting Pulse Width
Internal parameter PST multiplied by 0.01 second (10ms) determines pulse width (or it is set externally through PSV, if this input is connected). Transitions on the IN input will be ignored while the output pulse is active. The CTA counter is available as an output (integer).

Internal Counter CTA
Every time a pulse is generated in the output the pulse accumulator CTA is increased by one unit.

PSV Input
User may set the PSV input to select the value of the PST parameter externally. In this case, the PSV input must be connected to an FB output or anI/O module.

TP	TIMER PULSE

	IF PSV THEN
	PT:= PSV
	ELSE
	PT:= PST
	IF EN=1 THEN
	ENO := 1
	IF CTA > 0 AND CTA < PSV
	THEN
	OUT = 1
	CTA := CTA + 1
	ELSE
	OUT := 0
	IF IN = 0 AND CTA >= PSV
	THEN
	CTA := 0
	IF IN = 1 AND CTA = 0 THEN
	CTA := CTA + 1
	ELSE
	ENO := 0
	OUT := 0
	CTA := 0

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	PULSE INPUT	BOOLEAN
	PSV	THIS INPUT IS CONNECTED TO ADJUST PST EXTERNALLY	INT
P	PST	PARAMETER PRE ADJUSTED TIMER VALUE THROUGH PST	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	BLOCK OUTPUT	BOOLEAN
	CTA	TIMER PULSE ACCUMULATOR	INT
V	ICT	INITIAL TIMER VALUE TO THE COUNTER	INT
	STS	STATUS	WORD

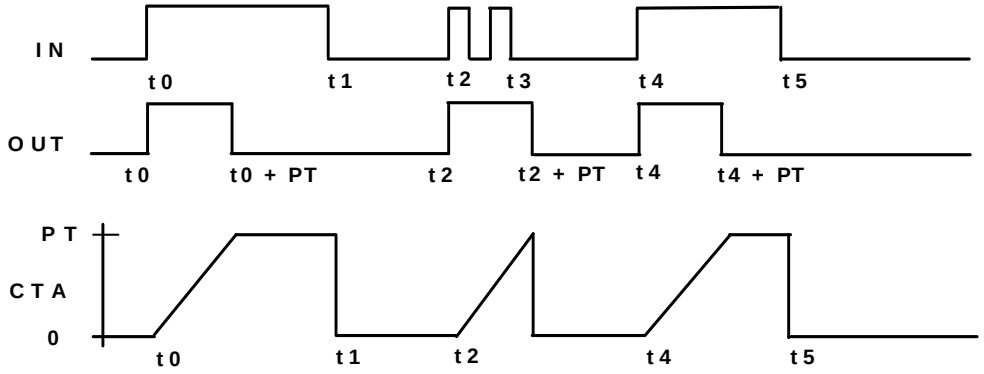
Bit sequence for the STS parameter

							8								0

Result Bit
BIT 8
1= ON, ON-DELAY OCCUR;
0= OFF, OUT=0.

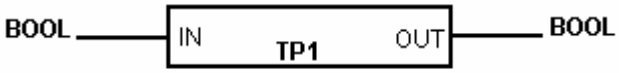
Enable Bit
BIT 0
1= COUNTER IS RUNNING;
0 = IS NOT COUNTING.

Timer Pulse Function - Timing diagrams



TP1 - Timer pulse

It works like the TP FB, however this block only has one pulse input and only one output OUT. Every time a rising transition occurs on the IN input, a pulse with width defined by the PST parameter times 0.01 seconds will be generated in the output.

TP1	TIMER PULSE
	

CLASS	MNEM	DESCRIPTION	TYPE
I	IN	PULSE INPUT	BOOLEAN
P	PST	PARAMETER PRE ADJUSTED TIMER VALUE THROUGH PST	INT
O	OUT	OUT=1 EVERY 0.01 SECONDS EVERY TIME THERE IS A RISING TRANSITION IN THE INPUT	BOOLEAN
V	STS	STATUS	WORD

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

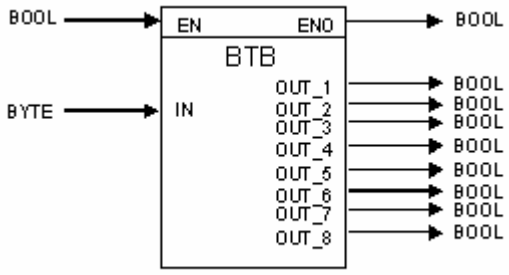
Data Manipulation Functions

BTB - Byte to bits conversion

Description
This FB converts 1 byte in 8 parallel outputs, each one of them representing one bit.

Conversion
One byte is the input of this block and the outputs (OUT_1 to OUT_8) compose the input byte in the parallel format. The least significant bit is OUT_1 and the most significant bit is OUT_8. The input byte may be from an I/O card, a virtual byte or another FB. Outputs can be connected to the BWL (Bit Wise Logic) block or the NOT block for example.

BTB	BYTE TO BITS CONVERSION

	IF EN=1 THEN
	ENO := 1
	OUT_1 := BIT_0 (IN)
	OUT_2 := BIT_1 (IN)
	OUT_3 := BIT_2 (IN)
	OUT_4 := BIT_3 (IN)
	OUT_5 := BIT_4 (IN)
	OUT_6 := BIT_5 (IN)
	OUT_7 := BIT_6 (IN)
	OUT_8 := BIT_7 (IN)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	BLOCK INPUT	BYTE
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT_1	BIT 0 (LSB)	BOOLEAN
	OUT_2	BIT 1	BOOLEAN
	OUT_3	BIT 2	BOOLEAN
	OUT_4	BIT 3	BOOLEAN
	OUT_5	BIT 4	BOOLEAN
	OUT_6	BIT 5	BOOLEAN
	OUT_7	BIT 6	BOOLEAN
	OUT_8	BIT 7 (MSB)	BOOLEAN

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

BTI - BCD to integer conversion

Description
This FB converts an input BCD value to an integer and puts the result in the OUT output.

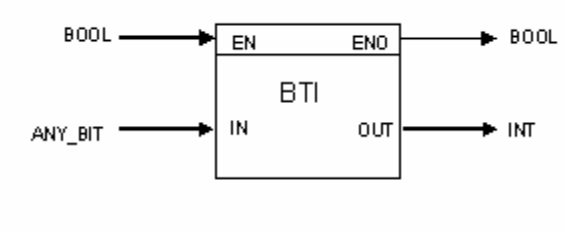
Conversion
A 2-digit number on BCD has the following format:

BIT7-BIT6-BIT5-BIT4 _____BIT3-BIT2-BIT1-BIT0

Each set of 4 bits composes a digit. For example: the number 10. If expressed in the BCD code it is written as 10. The first digit can be written in the binary form as 0001, and the second as 0000. So, 10BCD= 0001 000Binary.

It is common to confuse the BCD code with the binary representation. However, each group of 4 bits only represents one digit varying from 0 to 9. There can't be a representation on BCD like 12 9BCD, even though 12 can be expressed by 4 bits. The BCD code is typically used in 7 segment displays. Each segment represents a BCD digit. The above representation may be extended to N digits, always noting that each digit varies only from 0 to 9.

BTI	BCD TO INTEGER CONVERSION

	IF EN=1 THEN
	ENO := 1
	OUT = INTEGER(IN)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	BLOCK INPUT	ANY_BIT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	INPUT VALUE CONVERTED TO INTEGER	INT

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

FIFO - First In-First Out

Description
This block allows storing data with LC700. Every time this block is used, a non-volatile RAM area is reserved to the database first in first out.

FIFO size
The user may create this area directly in the SIZE parameter. The biggest size allowed will depend on the free RAM memory available in the CPU module when the setting is being done. Data in the FIFO and their correspondent sampling times may be accessed directly through communication with the CPU module via Modbus/RTU or Modbus/TCP.

Control Word (CTW) - FIFO MODE
- **Standard Mode:**

Data is stored in the FIFO until it is full. After that, any other data can't be stored until one or more variables are removed.

- **Moving Windows Mode:**

Data flow is always in the IN direction of the block. In this case if the FIFO is full, an automatic unload of the old variables is executed before the new variable can be stored. So, the FIFO always keeps the most updated samples.

- **Circular Queue Mode:**

Data flow is always in the IN direction of the block. If the FIFO is full, the new data will be stored in the oldest data's position. There is no change in the position of other data.

Control Word (CTW)- FIFO Memory
Data logging works in 3 different ways:
- Save Only Last Time
It saves variable and only time of the last sample.
- Do Not Save Time
It saves only the variable.
- Save Value and Time
It saves variable and sampling time for all samples.

LOAD, UNLOAD and CLEAR Inputs
Every time LOAD state changes to true, FIFO starts to store data in the IN input. At each CPU cycle of scan time, the FIFO block reads the input variable and increments the internal pointer to the next memory position. If UNLOAD changes to true, FIFO is unloaded. CLEAR input erases all memory area reserved for the FIFO block.

Trigger
If the trigger parameter is set to any N value, output will change to true when the FIFO stores the sample number N. Suppose Trigger is set to 9 and the FIFO size is 10 registers. When FIFO stores the ninth value the trigger output goes to true.

EMPTY and FULL outputs
EMPTY output indicates when FIFO is completely empty if its state is equal to true.
FULL output indicates when FIFO is completely full if its state is equal to true.

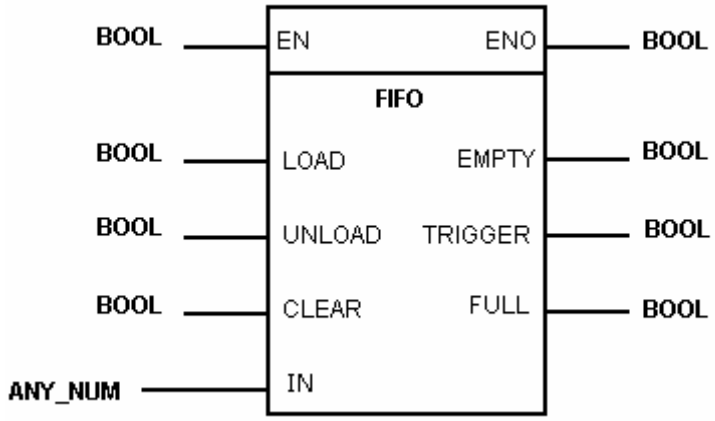
Data Type
The user must select two types of data to be stored: integer or real.
- 1 integer DATA has 2 bytes (1 Modbus Register)
- 1 real DATA has 4 bytes (2 Modbus Registers)

If the user chooses the data type as integer, each register will occupy one Modbus register. There must be a distinction between the number of registers set for FIFO and the actual number of Modbus registers. If the chosen data type is REAL, two Modbus registers are allocated.

Sampling Time Storing Mode
FIFO always allocates Modbus addresses to store time.
- Save Only Last Time: 6 bytes are allocated to register time for the last sample
- Do Not Save Time: FIFO reserves this Modbus area to internal time parameters, not storing sampling time.
- Save Value and Time: To each value stored it is reserved one register to this data plus 3 registers (6 bytes) to store each sampling time.

Allocated Modbus Addresses
FIFO is allocated by the CONF700 in a Modbus area 4xxxx (register). PTR is a pointer to the beginning of the Modbus address of the FIFO (relative Modbus Addresses). For example, if FIFO has 16 registers (words), Modbus registers 42501 to 42516 are addressed as 0 to 15.

FIFO	FIRST IN-FIRST OUT



Bit sequence for the FIFO Control Word

Configuration Only								Auxiliary and PRM Passing							
15				11	10	9	8	7	6	5	4	3	2	1	0

Auxiliary and PRM Passing

- Status indication bits:
  - Bit 0 - Is the EN boolean input status
  - Bit 1 - Is the LOAD boolean input status (1=LOAD; 0=NONE)
  - Bit 2 - Is the UNLOAD boolean input status (1=UNLOAD; 0=NONE)
  - Bit 3 - Is the CLEAR boolean input status (1=CLEAR; 0=NONE)
  - Bit 4 - Is the ENO boolean output status
  - Bit 5 - Is the EMPTY boolean output status
  - Bit 6 - Is the TRIGGER boolean output status (Trigger Quantity Matched)
  - Bit 7 - Is the FULL boolean output status

Configuration Only

Bit 11	Bit8	
0	0	- STANDARD MODE
0	1	- MOVING WINDOW MODE
1	0	- CIRCULAR QUEUE MODE
1	1	- CIRCULAR QUEUE MODE

NOTE
The CIRCULAR QUEUE mode is only implemented from the firmware version xx.55*.

* For further details about the equipment version, refer to the figure 3.2 of this manual

Select the ACQUISITION:

Bit 10	Bit 9	
0	0	Save data on FIFO and last time-stamp on the Control Table.
0	1	Save data to FIFO and no time-stamp at all.
1	0	Save data with time-stamp on every sample on the FIFO.

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	LOAD	LOAD N VALUES FROM FIFO	BOOLEAN
	UNLOAD	UNLOAD N VALUES FROM FIFO	BOOLEAN
	CLEAR	CLEAR FIFO DATA	BOOLEAN
	IN	DATA INPUT	ANY_NUM
	CTW	CONTROL WORLD	WORD
P	SIZE	SPECIFIES FIFO SIZE, REGISTER # MODBUS ADDRESS OF THE REGISTER (WORD)	INT
	TRIGGER	WHEN FIFO READS THE VALUE SET ON TRIGGER, OUTPUT TRIGGER WILL CHANGE TO TRUE.	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	EMPTY	FIFO IS EMPTY	BOOLEAN
	TRIGGER	INDICATES THAT THE AMOUNT OF REGISTERS SET ON THE TRIGGER PARAMETER WAS REACH	BOOLEAN
	FULL	FIFO IS FULL	BOOLEAN
V	PTR	POINTER FOR MEMORY ADDRESSING OF FIFO (MODBUS RELATIVE ADDRESS)	INT
	CTR	NUMBER COUNTER OF THE REGISTER USED BY FIFO	INT
	SEC	SECOND	BYTE
	MIN	MINUTE	BYTE
	HR	HOUR	BYTE
	DAY	DAY	BYTE
	MON	MONTH	BYTE
	YR	YEAR	BYTE

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

ICT - Integer constants

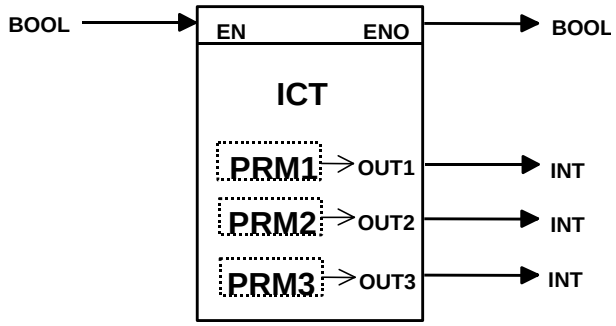
Description
This FB sends integer constant values to the outputs OUT1, OUT2 e OUT3. These constant values are set during the block configuration in the CONF700. These constants will only be sent to the outputs when the EN input is true and the outputs are necessarily integer variables.

PRM1, PRM2 e PRM3 Parameters
The user must type the values of the constants in these parameters. Each value will be passed to the respective block output. For example:

PRM1= 32
PRM2=346
PRM3= 456

When EN=1 true, OUT1, OUT2 e OUT 3 will indicate: 32, 346 and 456, respectively.

ICT	INTEGER CONSTANTS

	IF EN=1 THEN
	ENO := 1
	OUT1 := PRM1
	OUT2 := PRM2
	OUT3 := PRM3
	ELSE
	ENO := 0
	OUT1 := 0
	OUT2 := 0
	OUT3 := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
P	PRM1	VALUE OF CONSTANT 1	INT
	PRM2	VALUE OF CONSTANT 2	INT
	PRM3	VALUE OF CONSTANT 3	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT1	OUTPUT WITH VALUE SET ON PRM1	INT
	OUT2	OUTPUT WITH VALUE SET ON PRM2	INT
	OUT3	OUTPUT WITH VALUE SET ON PRM3	INT

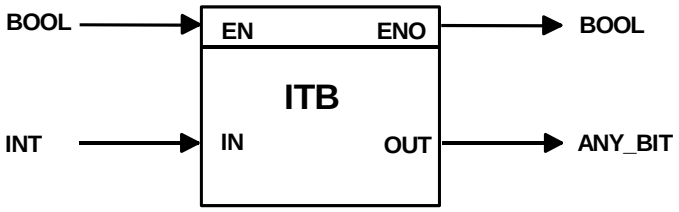
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

ITB - Integer to BCD conversion

Description
This function converts an integer to the BCD format and puts the result in the output OUT.

Conversion and Operation
If the output is a byte, the two less significant digits of the number are converted to BCD and if the output is a bit, it will represent the bit of less significant order of this conversion.
For example: IN= 112 and the output is a byte. In the output we have 12BCD or 0001 0010. If the output is one bit, it will indicate false.

ITB	INTEGER TO BCD CONVERSION

	IF EN=1 THEN
	ENO := 1
	OUT = BCD(IN)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	BLOCK INPUT	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	INPUT VALUE CONVERTED TO BCD	ANY_BIT

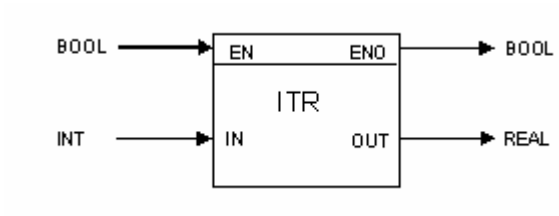
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

ITR - Integer to real conversion

Description
This function converts an integer to a real and puts the result in the output OUT.

Conversion and Operation
For example: Suppose we have 455 (integer) in the input of this block. The ITR block will convert this value to real allowing operations that require real data.

ITR	INTEGER TO REAL CONVERSION

	IF EN=1 THEN
	ENO := 1
	OUT = INT_TO_REAL(IN)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	BLOCK INPUT	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	INPUT VALUE CONVERTED TO REAL	REAL

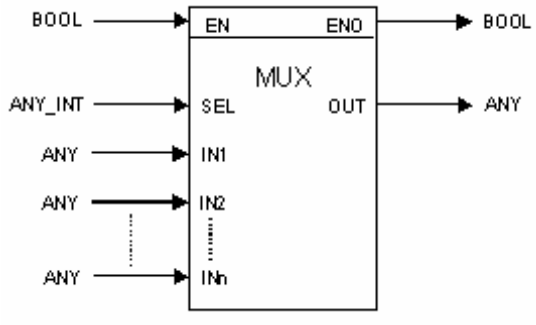
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

MUX - Multiplexer

Description
This function selects one of the inputs IN and puts the result in the OUT output. The selection is done according to the value in the SEL input.

Output Selection
If SEL is equal to 0, the selected output will be IN1. If SEL= 1 the selected output will be IN2 and so on. However, if the SEL input is negative, IN1 will be selected. If SEL is greater than the number of possible inputs (N-1) the INn output will be selected. In both exceptions ENO goes to false indicating the SEL input is out of range.

MUX	MULTIPLEXER

	IF EN=1 THEN
	ENO := 1
	SWITCH SEL
	CASE 0: OUT := IN1
	CASE 1: OUT := IN2
	.
	.
	.
	CASE n-1: OUT := INn
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	SEL	INPUT SELECTION	ANY_INT
	IN1	INPUT NUMBER 1	ANY
	IN2	INPUT NUMBER 2	ANY
	IN3	INPUT NUMBER 3	ANY
	…	…	
	…	…	
	INNn-1	INPUT NUMBER N-1	ANY
	INn	INPUT NUMBER N	ANY
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	OUTPUT SELECTED BY SEL INPUT	ANY

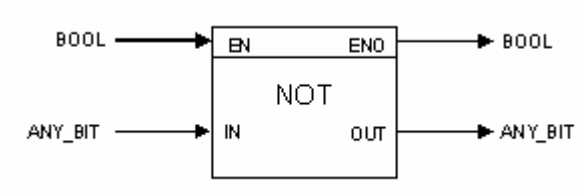
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

NOT - Bitwise Not

Description
It inverts the Logic State of the IN input. The output will be NOT (IN).

Operation
If the input is true (1), the output will be false and vice versa. This function allows both bit or byte inputs. One byte has all of its digits logically inverted. If the input is 0000000 the output will be 1111111.

NOT	BITWISE NOT

	IF EN=1 THEN
	ENO := 1
	OUT := NOT IN
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	BLOCK INPUT	ANY_BIT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	INPUT STATE IS LOGICALLY INVERTED	ANY_BIT

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

OSEL - Output binary selection

Description
This function allows the user to select one output to where the input value (IN) will be sent. If the input SEL is false (0), then the output OUT1 will be selected. Otherwise, OUT2 is selected.

Control Word- Selection of value of non selected outputs

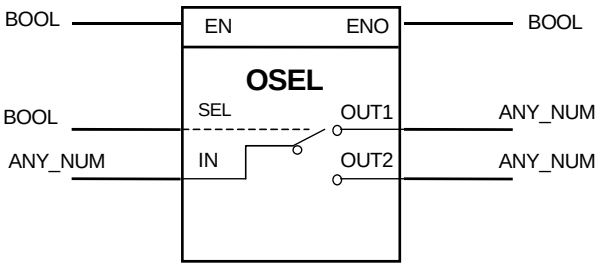
OUT1 Not Selected
When SEL is true, this will select OUT2. The user should set the desired value for the non-used output.

- Keep Last Value: Keep the last value of the output OUT1
- Set To Zero: Sends zero to the output OUT1

OUT2 Not Selected
When SEL is false, this will select OUT1. The user should set the desired value for the non-used output.

- Keep Last Value: Keep the last value of the output OUT2
- Set To Zero: Sends zero to the output OUT2

OSEL	OUTPUT BINARY SELECTION




```
IF EN=1 THEN
  ENO := 1
  IF SEL = 0 THEN
     OUT1 := IN
     IF m2 = 1 THEN
        OUT2 := 0
  ELSE
      OUT2 := IN
      IF m1 = 1 THEN
         OUT1  := 0
ELSE
  ENO := 0
```

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOL
	SEL	OUTPUT SELECTION	BOOL
	IN1	INPUT	ANY_NUM
P	CTW	Control word	WORD
O	ENO	OUTPUT ENABLE	BOOL
	OUT1	OUTPUT 1	ANY_NUM
	OUT2	OUTPUT 2	ANY_NUM

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

Bit sequence for the Control Word

Only Configuration								Auxiliary and parameter passage							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						m1	m2								

Bits below select the value of the output when it is not connected to the input IN.

Bit 8 **0 = Output OUT1 keeps last value**
1 = Output OUT1 goes to 0 (Set to ZERO)
Bit 9 **0 = Output OUT2 keeps last value**
1 = Output OUT2 goes to 0 (Set to ZERO)

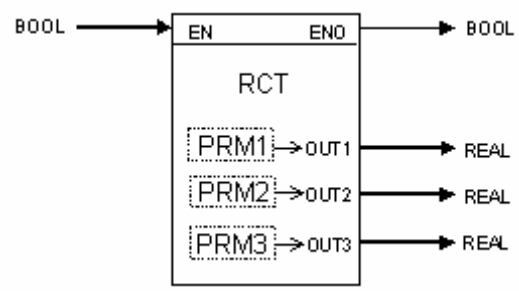
RCT - Real constants

Description
This FB sends real constant values to the outputs OUT1, OUT2 and OUT3. These constant values are set during the block configuration in the CONF700. These constants will only be sent to the outputs when the EN input is true and the outputs are necessarily real variables.

PRM1, PRM2 e PRM3 Parameters
The user must type the values of the constants in these parameters. Each value will be passed to the respective block output. For example:
PRM1= 32.78
PRM2=346.76
PRM3= 456.87

When EN=1 true, OUT1, OUT2 and OUT 3 will indicate: 32.78/346.76/456.87.

RCT	REAL CONSTANTS

	IF EN=1 THEN
	ENO := 1
	OUT1 := PRM1
	OUT2 := PRM2
	OUT3 := PRM3
	ELSE
	ENO := 0
	OUT1 := 0.0
	OUT2 := 0.0
	OUT3 := 0.0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
P	PRM1	CONSTANT VALUE 1	REAL
	PRM2	CONSTANT VALUE 2	REAL
	PRM3	CONSTANT VALUE 3	REAL
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT1	OUTPUT WITH VALUE SET ON PRM1	REAL
	OUT2	OUTPUT WITH VALUE SET ON PRM2	REAL
	OUT3	OUTPUT WITH VALUE SET ON PRM3	REAL

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

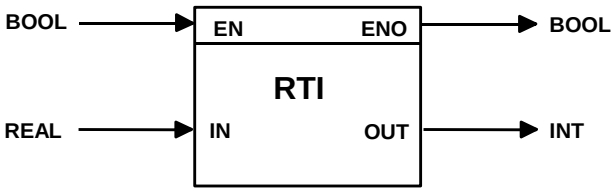
RTI - Real to integer conversion

Description
This function converts a real value to an integer and puts it in the OUT output.

Conversion and Operation
If the number we want to convert cannot be put in the integer format, OUT assumes the greatest (or smallest) possible integer value and ENO goes to false, indicating an exception in the function execution. The table below presents some of these conversions:

REAL	INTEGER
5,55	6
-4,954	-4
0,3	1
0,65	1
0,22	1
7,11	8
1001,1	1002
9050,7	9051
-0,25	0
-0,75	0
-0,55	0
1001,8	1002

RTI	REAL TO INTEGER CONVERSION

	IF EN=1 THEN
	ENO := 1
	IF IN > INT_H_LIMIT THEN
	OUT = INT_H_LIMIT
	ELSE
	IF IN < INT_L_LIMIT THEN
	OUT = INT_L_LIMIT
	ELSE
	OUT := REAL_TO_INT(ROUND_UP(IN
	))
	ELSE
	ENO := 0
	OUT := 0

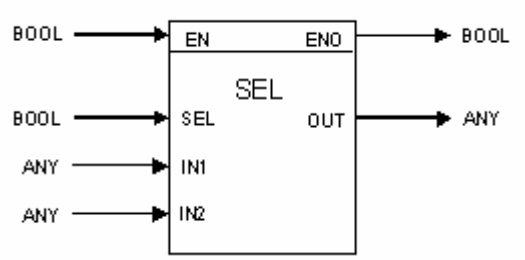
CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	BLOCK INPUT	REAL
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	INPUT VALUE CONVERTED TO INTEGER.	INT

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

SEL - Binary selection

Description
This FB is used to select between two inputs IN1 and IN2 and will re direct them to the OUT output. SEL works as a selection switch. If SEL is false, IN1 will be sent to OUT. Otherwise IN2 will be sent to the output OUT.

SEL	BINARY SELECTION

	IF EN=1 THEN
	ENO := 1
	IF SEL = 0 THEN
	OUT := IN1
	ELSE
	OUT := IN2
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	SEL	INPUT SELECTION	BOOLEAN
	IN1	INPUT 1	ANY
	IN2	INPUT 2	ANY
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	OUTPUT	ANY

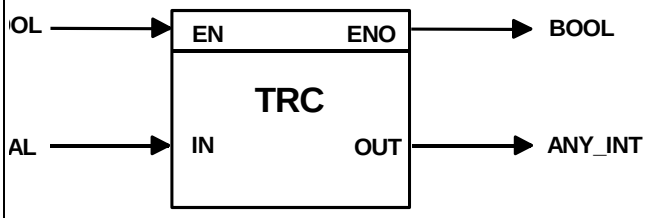
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

TRC - Truncate

Description
This function truncates a real number and the output will have only the integer part of the input number.

Conversion and Operation
Suppose the input is the format IN= X.Y then the output will be equal to OUT= X. For example, if IN= 1.34566 the output will be 1.

TRC	TRUNCRATE

	IF EN=1 THEN
	ENO := 1
	OUT = TRUNC(IN)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	BLOCK INPUT	REAL
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	TRUNCATED INPUT VALUE	ANY_INT

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

BWL - Bit Wise Logic

Description
This function allows implementation of the logic functions using an FB. Six different FBs can be set: AND, NAND, OR, NOR, XOR e NXOR. The user chooses the type of logic operation during the BWL block setting and this block will perform this logic function. It has extensions for more than 2 configurable inputs (max of 14 inputs). If the inputs are bytes, it will perform the logic operations bit-to-bit. If the inputs are bits, logic operations are done for each bit/input.

Control Word - AND Function
The logic function of the IN1 and IN2 inputs has an output given by the following Boolean expression:
OUT=IN1.IN2. This will result in a state table as shown below:

IN1	IN2	OUT
0	0	0
0	1	0
1	0	0
1	1	1

If the inputs are bytes, the AND function is done bit-to-bit, i.e.:

IN1= (BIT17)(BIT16)(BIT15)(BIT14)(BIT13)(BIT12)(BIT11)(BIT10)
IN2= (BIT27)(BIT26)(BIT25)(BIT24)(BIT23)(BIT22)(BIT21)(BIT20)
OUT= (BIT17ANDBIT27)……(BIT11ANDBIT21)

Ex: IN1= 00001111
IN2= 11110000
OUT= 00000000

Control Word - Function NAND
This function associates the AND and NOT functions. So, the logic output is the AND logic function inverted.

Control Word - Function OR
Logic function for the two inputs IN1 and IN2 have an output given by the expression:

OUT=IN1+IN2.

Transposed to state as in table below:

IN1	IN2	OUT
0	0	0
0	1	1
1	0	1
1	1	1

If the inputs are bytes, the OR function is done bit-to-bit,:

IN1= (BIT17)(BIT16)(BIT15)(BIT14)(BIT13)(BIT12)(BIT11)(BIT10)
IN2= (BIT27)(BIT26)(BIT25)(BIT24)(BIT23)(BIT22)(BIT21)(BIT20)
OUT= (BIT17ORBIT27)……(BIT11ORBIT21)

Ex: IN1= 00001111
IN2= 11110000
OUT= 11111111

Control Word - Function NOR
This function associates the OR and NOT functions. So, the logic output is the OR logic function inverted.

Control Word - Function XOR

Logic function for the two inputs IN1 and IN2 has an output given by the expression: OUT= $IN1\,\overline{IN2} + \overline{IN1}\,IN2$. Transposed to state as in the table below:

IN1	IN2	OUT
0	0	0
0	1	1
1	0	1
1	1	0

If the inputs are bytes, the XOR function is done bit-to-bit, i.e.:

IN1= (BIT17)(BIT16)(BIT15)(BIT14)(BIT13)(BIT12)(BIT11)(BIT10)
IN2= (BIT27)(BIT26)(BIT25)(BIT24)(BIT23)(BIT22)(BIT21)(BIT20)
OUT= (BIT17XORBIT27)……(BIT11XORBIT21)

Example: IN1= 01011100
IN2= 11110000
OUT= 10101100

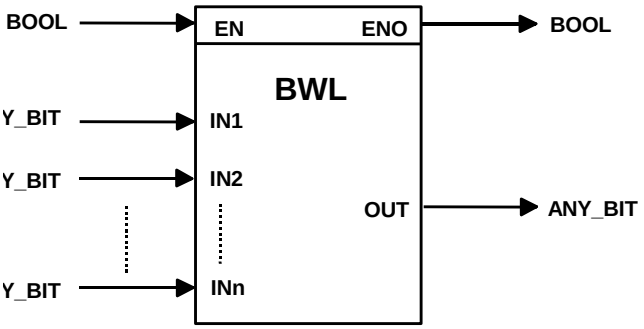
Control Word - Function NXOR

This function associates the AND and XOR functions. So, the logic output is the XOR logic function inverted.

The BWL block allows expansion to up to 14 inputs. In the table below we present logic functions for more than 2 inputs and their respective outputs.

INPUTS					OUTPUTS					
IN1	IN2	…	INn-1	INn	AND	NAND	OR	NOR	XOR	NXOR
0	0		0	0	0	1	0	1	0	1
0	0		0	1	0	1	1	0	1	0
0	0		1	0	0	1	1	0	1	0
0	0		1	1	0	1	1	0	0	1
		…					1	0		
1	1		1	0	0	1	1	0	1	0
1	1		1	1	1	0	1	0	0	1

BWL	BITWISE LOGIC

	IF EN=1 THEN
	ENO := 1
	OUT := IN1 BWL IN2 BWL ...
	BWL Inn
	IF CTW > 2 THEN
	OUT := NOT(OUT)
	ELSE
	ENO := 0
	OUT := 0
	; ----------
	BWL = IS THE LOGIC OPERATION
	CTW - PARAMETER defines the LOGIC OPER
	0 = AND ; 3 = NOT(AND)
	1 = OR ; 4 = NOT(OR)
	2 = XOR ; 5 = NOT(XOR)

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_BIT
	IN2	INPUT NUMBER 2	ANY_BIT
	IN3	INPUT NUMBER 3	ANY_BIT
	…	…	
	…	…	
	INNn-1	INPUT NUMBER N-1	ANY_BIT
	INn	INPUT NUMBER N	ANY_BIT
P	CTW	CONTROL WORD	WORD
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	OUTPUT	BOOLEAN/BYTE
V	OPR	LOGICAL OPERATION	WORD

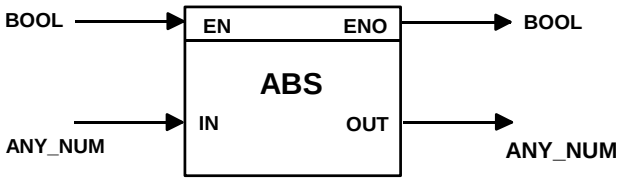
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

Mathematical Functions

ABS - Absolute Value

Description
This FB finds the absolute value of the input IN and puts the result in the output OUT.
For example, if IN= - 0.875 the output will be 0.875.

ABS	ABSOLUTE VALUE

	IF EN=1 THEN
	  ENO := 1
	  OUT := ABS(IN)
	ELSE
	  ENO := 0
	  OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	BLOCK INPUT	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	ABSOLUTE VALUE OF INPUT	ANY_NUM

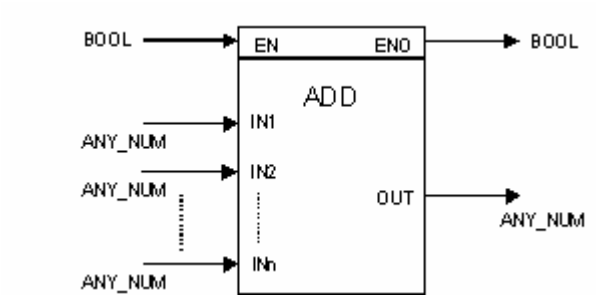
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

ADD - Addition

Description
This function adds all of the IN inputs and places the result in the OUT output.

Operation
If the result goes out of the limits of the data type that can be represented, OUT will be the greatest (or shortest) possible value represented according to its type. This would be indicated as ENO= false. The number of inputs (n) used in this operation is previously set during configuration. If the user tries to set more than two inputs with variables of different data types, for example, adding a real to an integer, the CONF700 will not allow this operation. As the first input is selected it is expected that all other inputs be of the same data type as in the first data type.

ADD	ADDITION

	IF EN=1 THEN
	ENO := 1
	OUT := IN1 + IN2 + ... + Inn
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_NUM
	IN2	INPUT NUMBER 2	ANY_NUM
	IN3	INPUT NUMBER 3	ANY_NUM
	…	…	
	…	…	
	INNn-1	INPUT NUMBER N-1	ANY_NUM
	INn	INPUT NUMBER N	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	ADDITION OPERATION RESULT	ANY_NUM

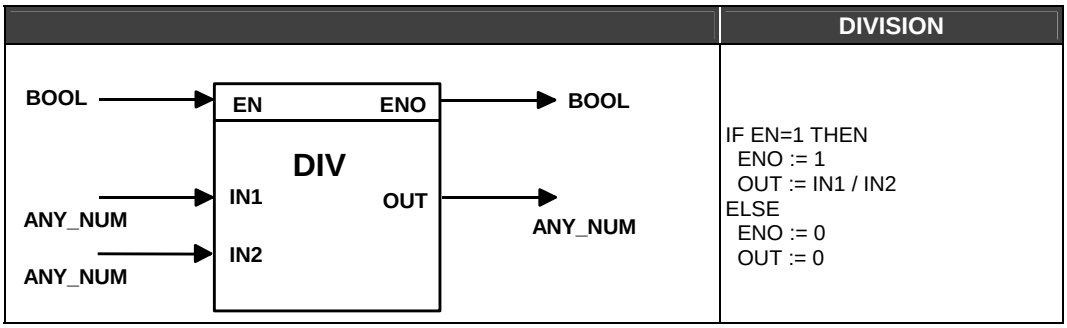
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

DIV - Division

Description
This function divides IN1 by IN2.

Operation
If the result goes out of the limits of the data type that can be represented, output will be the greatest (or shortest) possible value that can be represented according to its type. All exceptions are indicated by ENO equal to false.

If the user tries to set more than two inputs with variables of different data types, for example, dividing a real to an integer, CONF700 will not allow this operation. As the first input is selected it is expected that all other inputs will be of the same data type as the first data type. A division by zero will produce different results according to the data type. If the variables are integer the output will be –1. If the variables are real the result will be zero.



CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	DIVIDEND INPUT	ANY_NUM
	IN2	DIVISOR INPUT	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	DIVISION RESULT	ANY_NUM

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

MOD - Modulo

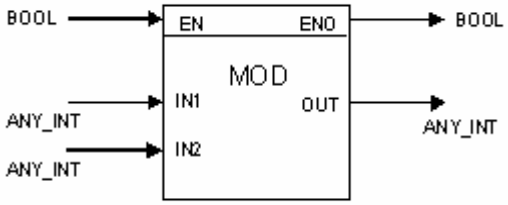
Description
This FB takes the rest of the division of IN1 by IN2 and places the result in the OUT output.

Operation
For example: IN1= 25 and IN2= 7, OUT= 4 because:

25 |_7_
4← 3

Both IN1 and IN2 must be integer variables.

MOD	MODULO

	IF EN=1 THEN
	ENO := 1
	/* OUT := IN1 MODULO IN2 */
	IF IN2 = 0 THEN
	OUT := 0
	ELSE
	OUT := IN1 - (IN1 / IN2) * IN2
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	DIVIDEND INPUT	ANY_INT
	IN2	DIVISOR INPUT	ANY_INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	REST OF DIVISION	ANY_INT

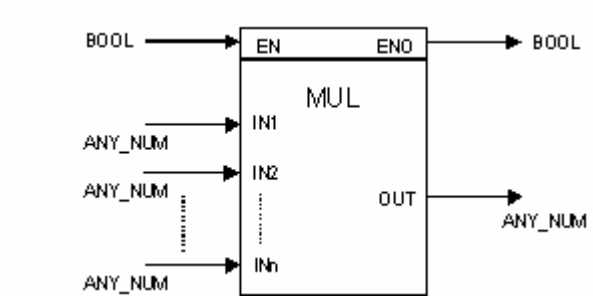
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

MUL - Multiplication

Description
This function multiplies all inputs and places the result in the OUT output.

Operation
If the result goes out of the limits of the data type that can be represented, OUT will be the greatest (or shortest) possible value represented according to its type. This would be indicated as ENO= false. The number of inputs (n) used in this operation is previously set during configuration. If the user tries to set more than two inputs with variables of different data types, for example, multiplying a real to an integer, CONF700 will not allow this operation. As the first input is selected it is expected that all other inputs will be the same data type as the first data type.

MUL	MULTIPLICATION

	IF EN=1 THEN
	ENO := 1
	OUT := IN1 * IN2 * ... * INn
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_NUM
	IN2	INPUT NUMBER 2	ANY_NUM
	IN3	INPUT NUMBER 3	ANY_NUM
	…	…	
	…	…	
	INNn-1	INPUT NUMBER N-1	ANY_NUM
	INn	INPUT NUMBER N	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	MULTIPLICATION RESULT	ANY_NUM

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

SQR - Square Root

Description
This FB will find the square root of the input and places the result in the OUT output. IF In is negative, OUT= 0 then the ENO output will indicate false.

Control World- Input/Output Type
The data type in the inputs and outputs may be set as “Regular” or “Percentages”.

If the Percentage option was chosen, then there are two modes of operation:

- If the input was set to integer:

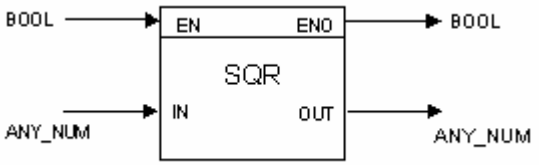
$$OUT = 100 * \sqrt{IN}$$

- If the input was set to real:

$$OUT = 10 * \sqrt{IN}$$

Option “Regular” will make the FB operate normally, that is calculating the SQR function directly.

SQR	SQUARE ROOT

	IF EN=1 THEN
	ENO := 1
	OUT := SQR(IN)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	INPUT	ANY_NUM
P	CTW	CONTROL WORLD	WORD
	CTO	LEVELING	REAL
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	OPERATION RESULT	ANY_NUM
V	PER	PERCENTAGE OPERATION	WORD

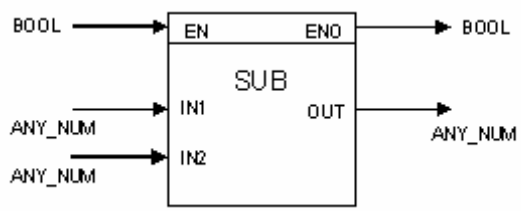
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

SUB - Subtraction

Description
This function subtracts IN2 from IN1 (IN1 - IN2).

Operation
If the result goes out of the limits of the data type that can be represented, OUT will be the greatest (or smallest) possible value represented according to its type. This situation is indicated as ENO= false. The number of inputs (n) used in this operation is previously set during configuration. If the user tries to set more than two inputs with variables of different data types, for example, adding a real to an integer, CONF700 will not allow this operation. As the first input is selected it is expected that all other inputs will be the same data type as the first data type.

SUB	SUBTRACTION

	IF EN=1 THEN
	ENO := 1
	OUT := IN1 - IN2
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	FIRST ELEMENT OF SUBTRACTION	ANY_NUM
	IN2	SECOND ELEMENT OF SUBTRACTION	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	SUBTRACTION RESULT	ANY_NUM

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

Comparison Functions

EQ - Equality

Description
This function will indicate true in the OUT output if the inputs do not have a deviation greater than the DBN value (Death Zone) of the input IN1. This block is indicated when the user desires to compare variables in terms of equality. The DBN parameter supplies a tool to determine how close each one of these measurements is to be considered equal.

DBN Parameter and Operation
In case only two inputs are used (IN1 and IN2) this FB performs as an equal-with-death-zone comparison, so OUT will be true only if ABS(IN1-IN2) <= DBN

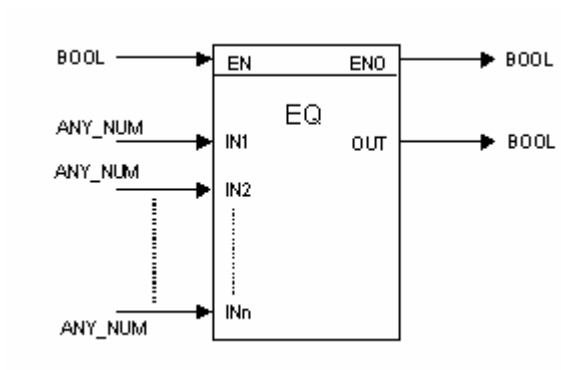
For example: We have 3 inputs and DBN is equal to 10. And IN1= 12, IN2=21 e IN3= 5.

So,

ABS(IN1-IN2)= 9 < 10
ABS(IN1-IN3)= 7 < 10

Thus, as DBN = 10, OUT is equal to true.

EQ	EQUALITY

	IF EN=1 THEN
	ENO := 1
	OUT := ABS (IN1 - IN2) <= DBN &
	ABS (IN1 - IN3) <= DBN &
	... &
	ABS (IN1 - INn) <= DBN
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_NUM
	IN2	INPUT NUMBER 2	ANY_NUM
	IN3	INPUT NUMBER 3	ANY_NUM
	…	…	
	…	…	
	INNn-1	INPUT NUMBER N-1	ANY_NUM
	INn	INPUT NUMBER N	ANY_NUM
P	DBN	DEAD ZONE	REAL
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	COMPARISION LOGIC RESULT	BOOLEAN

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

GE - Decreasing monotonic sequence

Description
This function will indicate true in the OUT output if the inputs (IN1 to INn) are disposed in a decreasing monotonic sequence, i.e., a sequence of numbers in two adjacent elements are related:

INn-1>=INn, i.e:

IN1,IN2,IN3…….INn-2,INn-1, INn

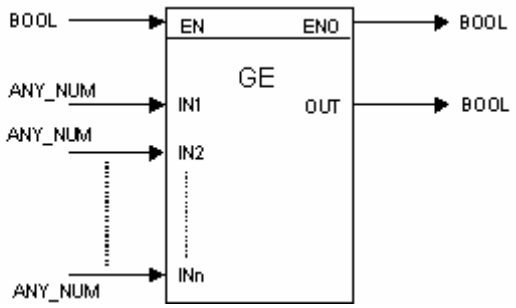
Where:
IN1>=IN2
IN2>=IN3
…
INn-2>=INn-1
INn-1>=INn

It is possible to use this expression to implement conditional blocks comparing two inputs and then, making a decision.

Operation:
An example: 12,8,8,5,3,1.

In a case where only two inputs are used this FB performs as a comparison of greater-or-equal to, making OUT= true if IN1>=IN2.

GE	DECREASING MONOTONIC SEQUENCE

	IF EN=1 THEN
	ENO := 1
	OUT := (IN1 >= IN2) & (IN2 >= IN3) & ...
	... & (INn-1 >= INn)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_NUM
	IN2	INPUT NUMBER 2	ANY_NUM
	IN3	INPUT NUMBER 3	ANY_NUM
	…	…	
	…	…	
	INNn-1	INPUT NUMBER N-1	ANY_NUM
	INn	INPUT NUMBER N	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	COMPARISION LOGIC RESULT	BOOLEAN

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

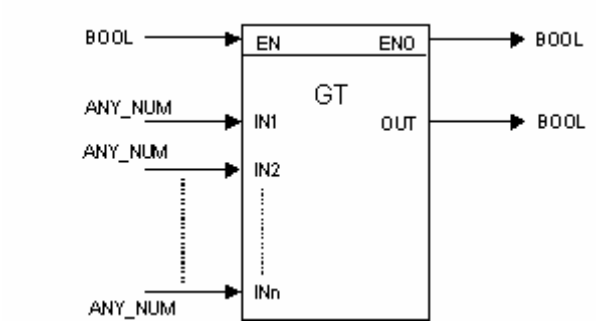
GT - Decreasing sequence

Description
This function will return true in the output OUT if the inputs (IN1 to INn) are in a decreasing order, i.e: IN1>IN2>IN3>IN4……..INn-1>INn.

In a case of the use of only 2 inputs IN's (IN1 and IN2) this FB performs as a comparison greater than, and OUT becomes true if IN1>In2.

It is possible to use this expression to implement conditional blocks that compare two inputs and then make a decision.

GT	DECREASING SEQUENCE

	IF EN=1 THEN
	ENO := 1
	OUT := (IN1 > IN2) & (IN2 > IN3) & ...
	& (INn-1 > INn)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_NUM
	IN2	INPUT NUMBER 2	ANY_NUM
	IN3	INPUT NUMBER 3	ANY_NUM
	…	…	
	…	…	
	INNn-1	INPUT NUMBER N-1	ANY_NUM
	INn	INPUT NUMBER N	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	COMPARISION LOGIC RESULT	BOOLEAN

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

LE - Increasing monotonic sequence

Description
This function will indicate true in the OUT output if the inputs (IN1 to INn) are disposed in an increasing monotonic sequence, that is., a sequence of numbers in two adjacent elements are related by INn-1 INn, so that:

IN1,IN2,IN3……..INn-2,INn-1, INn

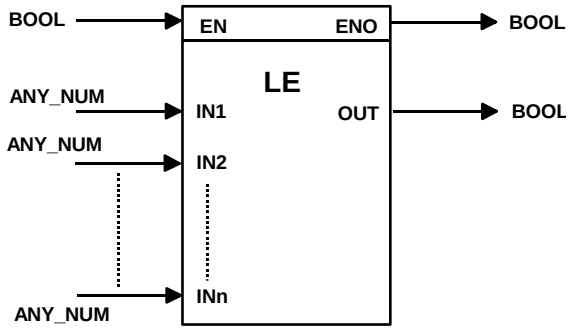
Where:
IN1 IN2
IN2 IN3
…
INn-2 INn-1
INn-1 INn.

For example: 1,1,3,3,4,5,6,78,78,8

In a case of the use of only 2 inputs IN's (IN1 and IN2) this FB performs as a comparison smaller or equal than, and OUT becomes true if IN1 IN2.

It is possible to use this expression to implement conditional blocks that compare two inputs and then make a decision.

LE	INCREASING MONOTONIC SEQUENCE

	IF EN=1 THEN
	ENO := 1
	OUT := (IN1 <= IN2) & (IN2 <= IN3) & ...
	... & (INn-1 <= INn)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_NUM
	IN2	INPUT NUMBER 2	ANY_NUM
	IN3	INPUT NUMBER 3	ANY_NUM
	…	…	
	…	…	
	INNn-1	INPUT NUMBER N-1	ANY_NUM
	INn	INPUT NUMBER N	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	COMPARISION LOGIC RESULT	BOOLEAN

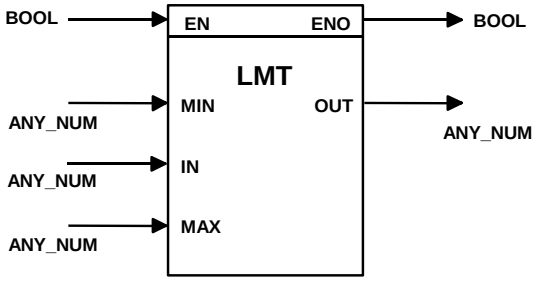
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

LMT - Limiter

Description
This FB limits the IN input between values of MAX and MIN inputs and places the result in the output OUT. If the limits are exceeded ENO will change to false. Suppose that we wish to limit the signal input between 1 and 10. In this case we may create two constants and connect them at the MIN and MAX inputs. When the upper limit is exceeded the output is equal to 10 and when the bottom limit is reached the output is 1.

MIN and MAX Parameters
The user will configure the maximum value (MAX) and minimum value (MIN) of the output. The output will be equal to MAX if the input exceeded the value set for MAX and it will be equal to MIN if the input is smaller than MIN.

LMT	LIMITER

	If en=1 then
	Out:= MIN(MAXM(IN,MIN),MAX)
	IF (OUT<MIN) OR (OUT>MAX) then
	ENO:=0
	Else
	ENO:=1
	Else
	Eno:=0
	Out:=0

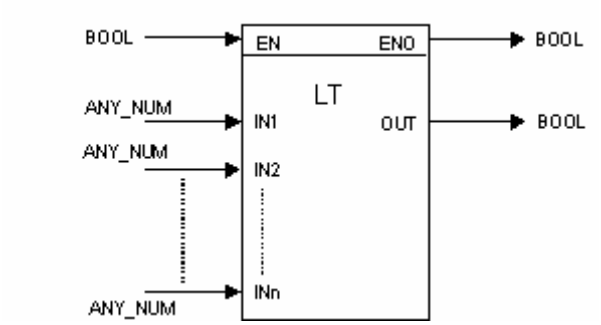
CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN	INPUT TO BE LIMITED	ANY_NUM
	MIN	LIMITER MINIMUM LIMIT	ANY_NUM
	MAX	LIMITER MAXIMUM LIMIT	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	LIMITED BLOCK OUTPUT	ANY_NUM

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

LT - Increasing sequence

Description
This function will return to true in the output OUT if the inputs (IN1 to INn) are in an increasing order, i.e: IN1<IN2<IN3<IN4……..INn-1<INn
In the case of the use of only2 inputs IN's (IN1 and IN2) this FB performs as a comparison smaller than, and OUT becomes true if IN1 < IN2.
It is possible to use this expression to implement conditional blocks that compare two inputs and then make a decision.

LT	INCREASING SEQUENCE

	IF EN=1 THEN
	ENO := 1
	OUT := (IN1 < IN2) & (IN2 < IN3) & ...
	& (INn-1 < INn)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_NUM
	IN2	INPUT NUMBER 2	ANY_NUM
	IN3	INPUT NUMBER 3	ANY_NUM
	…	…	
	…	…	
	INNn-1	INPUT NUMBER N-1	ANY_NUM
	INn	INPUT NUMBER N	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	COMPARISON LOGIC RESULT	BOOLEAN

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

MAX - Maximum

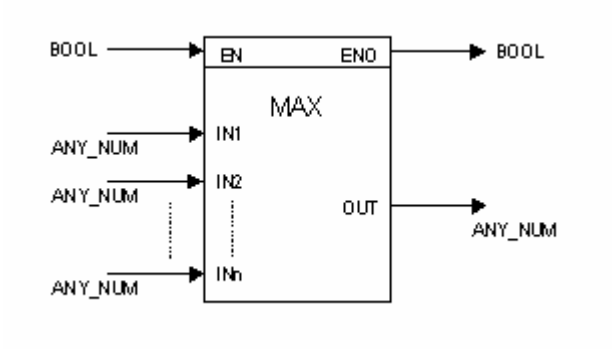
Description
This function selects the maximum value of the IN inputs and places it in OUT outputs.

Operation
The number of inputs is previously determined during setting. Suppose we have 4 inputs and their values are:

IN1= 5.899
IN2= 7.900
IN3= 10.899
IN4= 23.90

The output generated by the MAX FB will be IN4 or 23.90.

MAX	MÁXIMUM

	IF EN=1 THEN
	ENO := 1
	OUT := MAX(IN1, IN2, ..., INn)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_NUM
	IN2	INPUT NUMBER 2	ANY_NUM
	IN3	INPUT NUMBER 3	ANY_NUM
	…	…	
	…	…	
	INNn-1	INPUT NUMBER N-1	ANY_NUM
	INn	INPUT NUMBER N	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	MAXIMUM INPUT VALUE	BOOLEAN

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

MIN - Minimum

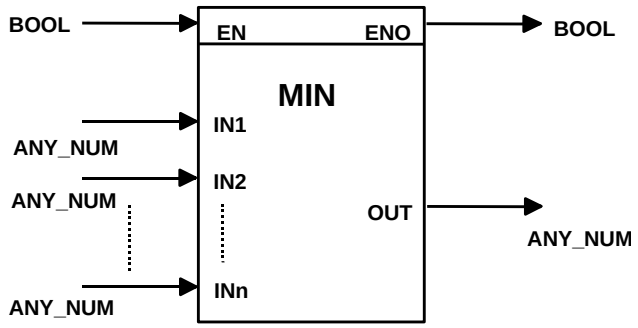
Description
This function selects the minimum value of the IN inputs and places it in OUT outputs.

Operation
The number of inputs is previously determined during setting. Suppose we have 4 inputs and their values are:

IN1= 5.899
IN2= 7.900
IN3= 10.899
IN4= 23.90

The output generated by the MAX FB will be IN4 or 5.899.

MIN	MÍNIMUM

	IF EN=1 THEN
	ENO := 1
	OUT := MIN(IN1, IN2, ..., INn)
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_NUM
	IN2	INPUT NUMBER 2	ANY_NUM
	IN3	INPUT NUMBER 3	ANY_NUM
	…	…	
	…	…	
	INn-1	INPUT NUMBER N-1	ANY_NUM
	INn	INPUT NUMBER N	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	MINIMUM INPUT VALUE	BOOLEAN

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

NE - Inequality

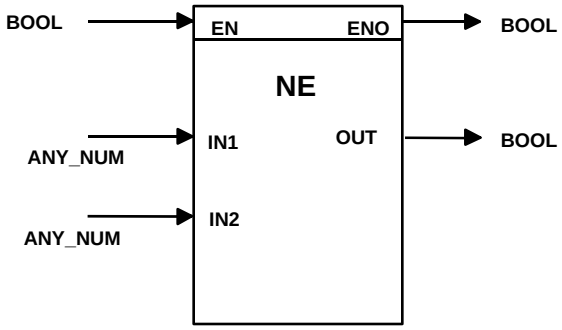
Description

The output will indicate true only if the difference IN1-IN is shorter than the DBN value, i.e., (IN1-IN2) > DBN. The user sets the DBN parameter.

Operation
Example:
IN1= 0.78
IN2= 0.70
IN1-N2=0.08
DBN= 0.05

In this case OUT is equal to 1 (true) because the DBN value (0.05) indicates that in this example IN1 is different of IN2. User may control the range where there is equality through the DBN parameter.

NE	INEQUALITY

	IF EN=1 THEN
	ENO := 1
	OUT := ABS (IN1 - IN2) > DBN
	ELSE
	ENO := 0
	OUT := 0

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	INPUT NUMBER 1	ANY_NUM
	IN2	INPUT NUMBER 2	ANY_NUM
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	LOGIC COMPARISON RESULT	BOOLEAN

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

Process Control Functions

XLIM- Cross Limit and Rate-Of-Change

Description
This function limits a signal between static and dynamic values and also controls the rate of change. OUT% is the filtered result of the A% input.

Static And Dynamic Limitation

- **Static**

  To limit statically, the input B is disconnected. Signal A is limited between BL and BH (user's settings).

- **Dynamic**

  If input B is connected it is possible to dynamically limit the input A through the B input. To achieve more flexibility, these limits are changeable with individual gain and bias.

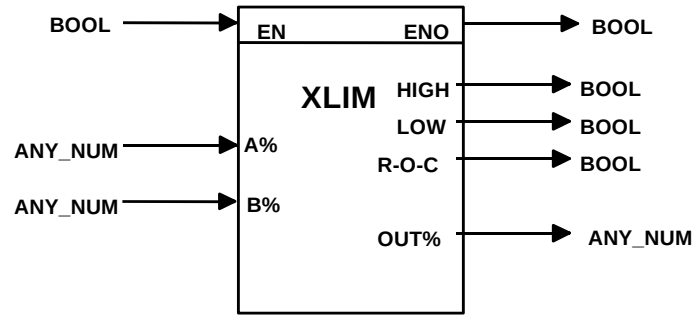
Control Word (CTW)- Rate of Change
Limit of rate of change may be applied in three ways, increasing, decreasing or in a specific direction. There are 4 types of rate of change available: Check both; Check Only Upper; only actuation rate; or none.

BL and BH parameters
If A BL output OUT is equal to BL.
If BL < A < BH output OUT is equal to A.
If A BH output OUT is equal to BH.

Parameters GH e GL
If A B.GL+BL output OUT is equal to B.GL + BL
If B.GL+BL <A < B.GH+BH output OUT is equal to a A
If A B.GH+BH output OUT is equal to B.GH + BH

DB Parameter and LOW and HIGH outputs
This FB has two outputs to indicate if the low (LOW) or high (HIGH) limits were reached. The DB parameter can be adjusted to generate a hysteresis, avoiding output oscillation while the variable is close to the limit value.

RAT Parameter and R-O-C output
ROC output goes to true when the signal rate of change reaches the value set in the parameter RAT. When the input A changes faster than RAT, the variation in the output is kept inside a value fixed by RAT until the input signal decreases to a value inferior to RAT. The ROC alarm in this interval is on HIGH.

XLIM	XLIM CROSS LIMIT AND RATE-OF-CHANGE
	

CLASS	PARAM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	A%	INPUT A	ANY_NUM
	B%	INPUT B	ANY_NUM
P	CTW	CHECK BOTH/JUST UPPER/ONLY ACTUATION RATE	WORD
	GL	BOTTOM LIMIT GAIN	I/1000
	BL	BOTTOM LIMIT BIAS	I/100
	GH	UPPER LIMIT GAIN	I/1000
	BH	UPPER LIMIT BIAS	I/100
	DB	DEAD ZONE (HYSTHERESIS) %	I/100
	RAT	SPEED OF MAXIMUM VARIATION % PER SECOND	I/100
O	ENO	OUTPUT ENABLE	BOOLEAN
	HIGH	UPPER LIMIT ALARM	BOOLEAN
	LOW	BOTTOM limit ALARM	BOOLEAN
	R-O-C	RATE OF CHANGE ALARM	BOOLEAN
	OUT%	OUTPUT	ANY_NUM

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

TOT - Totalization

Description
This block gives the totalization of the input. This totalization is the integral of the input times a scale factor FCF that allows the user to configure totalization in 3 different modes of operation. If your application requires the computing of instantaneous totalized volume, use the TOT FB to accomplish this task. The time basis of this calculation is seconds.

Flow is generally given in Engineering Units (EU) by units of time. For example: a 1 m^3/ second flow as input of the TOT FB will have as output volume in m^3.

For a given application that requires the value of energy for a eletric device, the TOT block allows the calculation for this value using the instantaneous power.

$$Energy = \int Pot\ (t)\, dt$$

and also Pot(t)= V(t).I(t), where V(t) is the instantaneous voltage and I(t) is the instantaneous current.

TOT Output and TU Parameter
The time interval while the output is totalized is according to the value set for TU. The integration (totalization) is kept in an internal register that goes up to 8000000 units. Output TOT is the totalization value.

dl Output
Maximum value of totalization is 8000000 and the minimum is – 8000000. Every time the FB output reaches these values the dl output changes from false to true during an interval of time. That particular Function Block has an internal counter. The user can reset this counting through a command that is represented by a pulse or a button. This dl output is a second counter that counts how many times this "CLEAR" operation was made.

FCF Parameter
The FCF parameter allows the TOT FB to operate in 4 different modes:

a) IN is real and represents flow in Engineering Units (EU):
FCF must be equal to 1 so that the totalization is done without any EU scale factor. (or adjust the factor you wish to use) For example:

Flow Q is measured in m^3/hours. 1 hour has 3600 seconds. So, the TU value must be equal to 3600. Suppose a constant flow of 60 m^3/hour. The totalization is given by the expression:

$$TOT(t) = \int_0^{t(s)} \frac{FCF}{TU} * IN(t)dt = \int_0^{t(s)} \frac{1}{3600} * 60dt = \int_0^{t(ss)} \frac{1}{60} dt[m^3]$$

So, after 1 minute or 1/60 hours or 60 seconds the TOTvalue is :

$$TOT[m^3] = \int_0^{60} \frac{1}{60} dt = 1m^3$$

Each 1/60 hours or each one 1 minute, the TOT block totalizes the input and displays the value in the output. So:

60 m^3_____ 1 hour
1 m^3_____ t (time interval when the totalization is displayed/updated)

So, t= 1/60 hours or 1 minute.

b) IN is REAL and represents flow in percentage values
In this case the input will be seen as a percentage represented by a REAL number in the range 0 to 100 %. FC must be equal to the maximum value in engineering units (flow at 100%) so the totalization is given in EUs. The TU parameter setting is similar to the previous item. The totalization will be displayed in the EU configured.

c) IN is INTEGER

In this case the input will be interpreted as an integer number in the range of 0 to 10000 (0% and 100%). FCF must be equal to the maximum flow in EUs divided by 10000. Suppose a maximum flow of 1 m3/s and a constant flow of 0.5 1 m3/s. The value for FCF is equal to the maximum flow divided by 10000, or 0.0001. TU in this case is 1 because the unit of totalization is m3. A 0.5 1 m3/second input is equal to 5000 or 50 % of the scale. Thus:

$$OUT = \int_0^t \frac{FCF}{TU} * IN\%(t)dt = \int_0^t 0.0001 * 5000 dt = 0.5t(m^3)$$

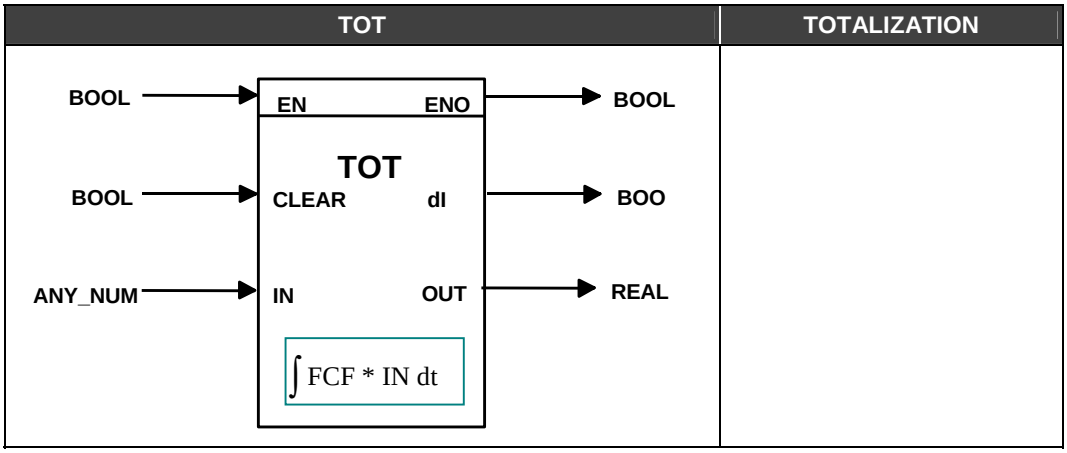
So, in 1 minute (or 60 seconds) the totalized value is 30 m3.

d) When FC is smaller than zero

When the block is totalizing a negative flow, the totalization is decreased while when the flow is positive the totalization is incremented. When FCF is greater than zero, positive, the TOT FB only accepts positive flows.

CLEAR Input

If the CLEAR input is changed to true, the totalization is re started and the internal registers of the TOT FB are cleared.

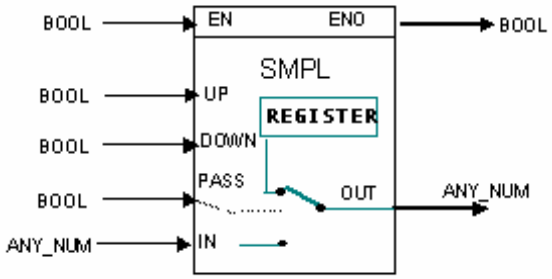


CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	CLEAR	CLEARS THE TOTALIZATION	BOOLEAN
	IN	BLOCK INPUT	ANY_NUM
P	CTW	CONTROL WORLD	WORD
	TU	TOTALIZATION VALUE FOR ONE UNITY OF COUNTING	REAL
	FCF	FACTOR OF FLOW RATE	REAL
O	ENO	OUTPUT ENABLE	BOOLEAN
	dI	ALARM THAT INDICATES WHEN THE TOTALIZATION REACHED 8000000 OR –800000. IN THIS CASE dI= 1.	BOOLEAN
	OUT	TOTALIZED OUTPUT	REAL
V	ACC	FRACTIONARY VALUE OF TOTALIZATION	REAL

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

SMPL - Sample Hold with Up and Down

This FB samples the value of the IN input and places it in a register when the PASS input moves from true to false. The register value may be increased or decreased using the UP and DOWN inputs. The speed of this increment or decrement is defined by the ASPD parameter. This block may be used with a PID block.

SMPL	SAMPLE HOLD WITH UP AND DOWN
	

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	UP	INCREMENT COUNTER	BOOLEAN
	DOWN	DECREMENT COUNTER	BOOLEAN
	PASS	PUTS REGISTER VALUE IN THE OUTPUT	BOOLEAN
	IN	INPUT	ANY_NUM
P	ACCEL	ACCELERATION FACTOR- INCREMENT OR DECREMENT	INT
	ASPD	SPEED OF ACTUATION IN % PER SECOND	REAL
	L_LMT	BOTTON LIMIT	REAL
	H_LMT	UPPER LIMIT	REAL
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT	Output	ANY_NUM

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

ARAMP - Automatic Up And Down Ramp

Description
This function increments or decrements the OUT output in a linear way based on a pre- established time interval. This FB may be used to create a database to an automatic setpoint generator when used together with the Linearization block or a simple ramp.
In a setpoint application this FB is prepared to generate a 0% to 100 % output in a time interval tracking the setpoint curve. The ARAMP output will be connected to the input of the LIN function (linearization) set with a profile curve of the setpoint.

Control Word- Time Selection
The time basis of this block can be selected in minutes, seconds or hours according to the requirements of the application. This selection affects directly the chosen value for the FTIME parameter.

FTIME, INC/DECR Parameter
FTIME is the time output it takes to change from 0% to 100 %. Direction of the change is given by the output INC/DECR. If this input is true, the output OUT will be gradually decreased with speed defined by the FTIME parameter, otherwise, the output will be incremented with the speed defined in the FTIME parameter.

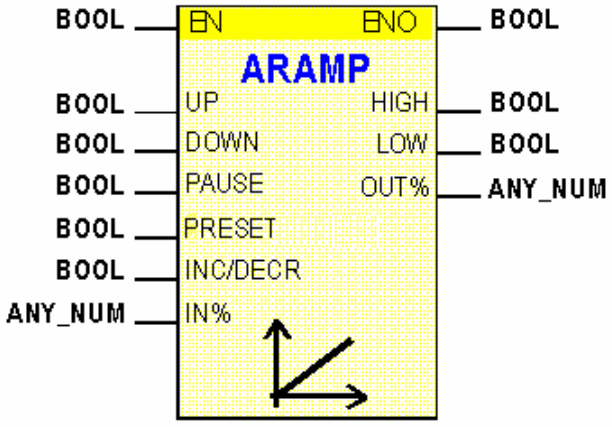
Pause Command (PAUSE)
PAUSE freezes the output OUT so that the output can be incremented or decrement right after pause is true through the UP and DOWN inputs.

UP and DOWN Commands, ASPD Parameter
The UP and DOWN inputs will advance or revert the output OUT to a desired value using the manual speed adjusted by the ASPD parameter. This parameter configures the speed of manual actuation.

LOW_L and HIGH_L Parameters
The parameter LOW_L configures the bottom limit of the ramp generated by the ARAMP FB while the HIGH_L parameter configures the upper limit of the output ramp. It starts from the value in the IN input and goes to the maximum value, set in the HIGH_L parameter. If the value is smaller than LOW_L, the initial value of the ramp will be equal to LOW_L.

HIGH and LOW Alarms
When the output ramp reaches the bottom limit (LOW_L) or upper limit (HIGH_L), the alarms LOW and HIGH will be turned on. LOW goes to high level if the bottom limit is reached. Similarly, if the upper limit is reached the output HIGH goes to true.

ACCEL Parameter
It is the manual acceleration of actuation. When the block output is a parabola, the ACCEL parameter allows better adjusting of the output, allowing more definition of the rate of change of the output.

ARAMP	AUTOMATIC UP AND DOWN RAMP
	

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	UP	MOVES OUTPUT FORWARD ACCORDING TO ASPD	BOOLEAN
	DOWN	REVERTS OUTPUT ACCORDING TO ASPD	BOOLEAN
	PAUSE	FREEZES OUTPUT	BOOLEAN
	PRESET	RAMP PRESET	BOOLEAN
	INC/DEC	OUTPUT WILL BE INCREMENTED OUTPUT WILL BE DECREASED	BOOLEAN
	IN%	BLOCK INPUT	ANY_NUM
P	CTW	HOURS, MINUTES OR SECONDS	WORD
	ASPD	MANUAL ACTUATION SPEED IN % PER SECOND	INT/100
	ACCEL	INITIAL MANUAL ACCELERATION OF ACTUATION	INT
	FTIME	TIME TO CHANGE THE OUTPUT FROM 0 TO 100 %	INT
	LOW_L	BOTTOM LIMIT OF REGISTER	INT/100
	HIGH_L	UPPER LIMIT OF REGISTER	INT/100
O	ENO	OUTPUT ENABLE	BOOLEAN
	HIGH	RAMP UPPER LIMIT ALARM	BOOLEAN
	LOW	RAMP BOTTOM LIMIT ALARM	BOOLEAN
	OUT%	OUTPUT RAMP	ANY_NUM

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

LIN - Linearization

Description
This block simulates a function using a table of points. Intermediate values are computed using the linear interpolation method. They can be arranged to implement curves of more than 10 points. The user should set a table of points, X and Y pair of points, that represent the function. A value in the X input corresponds to an output value Y, i.e., this block creates a function f(x). To each X coordinate there is a corresponding y.. The user must enter 10 pairs of points:

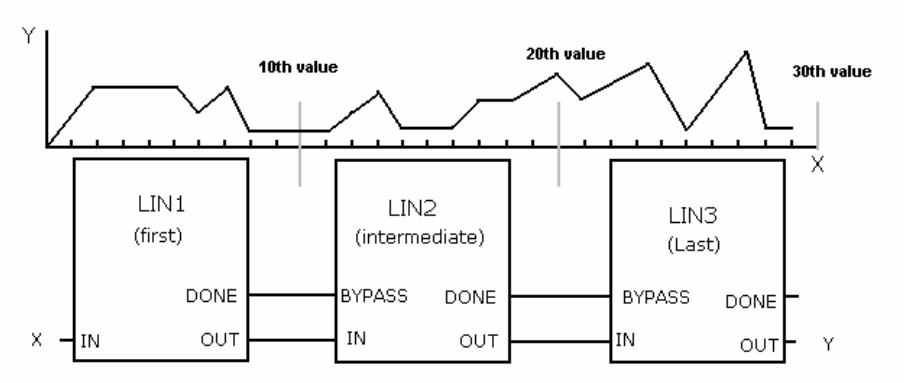
(x1,y1), (x2,y2), (x3,y3), (x4,y4), (x5,y5), (x6,y6),(x7,y7),(x8,y8),(x9,y9), (x10,y10)

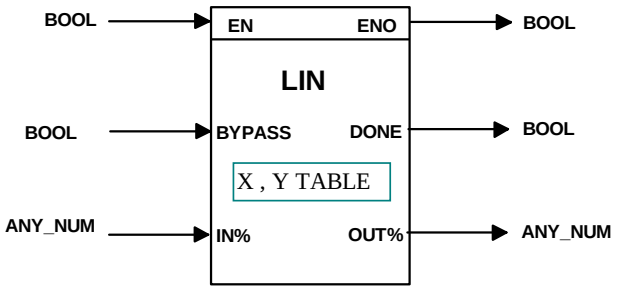
IN Input
This FB may work in two modes according to the input settings:
Input IN% is an integer:
In this case the FB input will be interpreted as a number in the range 0 to 10000.
Input IN% is a real:
In this case the FB input will be interpreted as a real percentage.

Bypass
If the Bypass input is on HIGH, the LIN block passes the input of the block without processing it.

Control Word (CTW)- Serial Behavior
When an application requires more than 10 points, LIN FB may be put in a series. The DONE signal must be connected to the bypass input of the next block. The first block must be set as FIRST and the intermediate blocks as INTERMEDIATE and the last block as LAST.

An application with 30 points will have the following configuration:



LIN	LINEARIZATION
	

CLASS	PARAM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	BYPASS	PASSES THE INPUT TO THE OUTPUT WITHOUT ANY PROCESSING	BOOLEAN
	IN%	BLOCK INPUT	ANY_NUM
P	CTW	CONTROL WORLD	WORD
	X1	X TO THE FIRST POINT	I/100
	Y1	Y TO THE FIRST POINT	I/100
	.		.
	.		.
	X10	X TO THE LAST POINT	I/100
	Y10	Y TO THE LAST POINT	I/100
	CTW	CONTROL WORLD	WORD
O	ENO	OUTPUT ENABLE	BOOLEAN
	DONE	ENABLES THE NEXT LIN BLOCK IN AN APPLICATION IN SERIES.	BOOLEAN
	OUT	OUTPUT	ANY_NUM

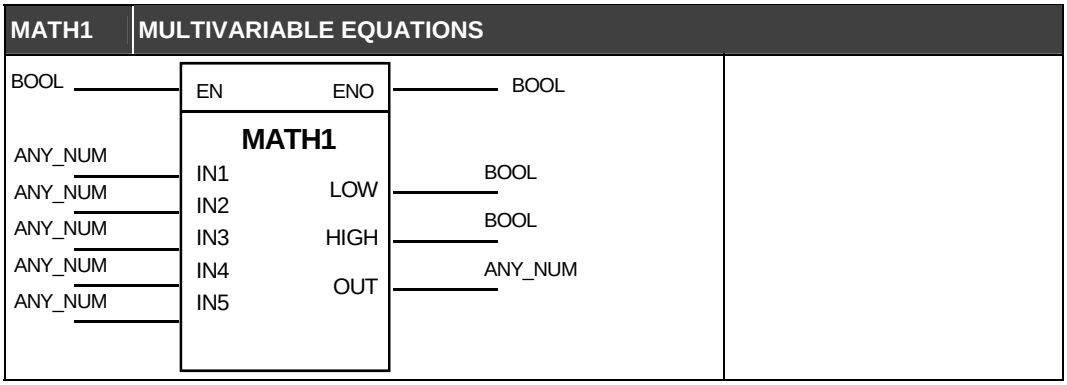
I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

MATH1 - Multivariable Equations

Description
It is possible to choose between 3 equations that make different mathematical operations. To each chosen type of equation, there are specific parameter settings. The 3 types of equation are:

EQUATION1 - RELATIVE HUMIDITY
EQUATION 2- API FUNCTION
EQUATION 3- SIGN PROCESSING

NOTE
In the block settings the CONF700 will present to each equation options to set many parameters. The user should only set the parameters indicated in this manual.



Equation 1 – RELATIVE HUMIDITY

The Relative Humidity output is relative humidity calculated with reference to two inputs. These consist of temperature inputs. One of them will read from a dry bulb and other read will from a humid bulb.

Humidity
Humidity is equal to the result of the function of moisture calculation, in the range of 0.000000 to 1000000, representing values from 0 to 100 %.

Scale Conversion to the outputs
OUT = A * Humidity + B
A and B parameters are set by the user. A is the scale GAIN of the output value OUT and B is the BIAS of the scale of output value OUT. Example: to obtain a 0 to 100 % output, A must be set to equal 100 and B to equal 0.

LOW and HIGH Parameters
LOW is the bottom LIMIT of the output OUT. If the output OUT is smaller than the bottom limit, it is displayed in the digital output LOW, that is., LOW= 1.

HIGH is the upper limit of the output OUT. If the output OUT is smaller than the bottom limit, it is displayed in the digital output HIGH, i.e., HIGH= 1.

Relative Humidity Equation Parameters
K1: It is a constant adjusted according to where the application is located. This value must be equal to the local atmospheric pressure and it is set in the block parameters configuration windows of the CONF700..

K2: It is the scale GAIN of input values on the inputs IN1 and IN2

K3: Scale BIAS of the values in the inputs IN1 and IN2.

Inputs are calculated using the following equations:

Tbdry = IN1 * K2 + K3
Tbhumid = IN2* K2 + K3

To obtain a 0 to 100 C input, where the values of IN1 and In2 are from 0 to 10000 K2 must be equal to 0.01 and K3 equal to 0.

K4: It shows the value of Tbdry = IN1 * K2 + K3 in Engineering Units (only for the supervision system using Modbus communication in the Modbus Address specified by the K4 parameter).

K5: It shows the value of TbHumid= IN2*K2 + K3 in EU Units (only for the supervision system using Modbus communication in the Modbus Address specified by the K5 parameter).

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	DRY BULB TEMPERATURE (BEFORE CONVERSION)	REAL
	IN2	HUMID BULB TEMPERATURE (BEFORE CONVERSION)	REAL
	IN3	NOT USED	REAL
	IN4	NOT USED	REAL
	IN5	NOT USED	REAL
P	K1	ATMOSPHERIC PRESSURE	REAL
	K2	GAIN OF VALUE SCALE OF Inputs IN1 AND IN2	REAL
	K3	BIAS OF VALUE SCALE OF Inputs IN1 AND IN2	REAL
	K4	DISPLAYS VALUE OF Tbdry (AFTER CONVERSION)	REAL
	K5	DISPLAYS VALUE OF TbHUMID (AFTER CONVERSION)	REAL
	A	OUTPUT SCALE (gAIN)	REAL
	B	OUTPUT SCALE (BIAS)	REAL
	LOW	OUTPUT BOTTOM LIMIT	REAL
	HIGH	OUTPUT UPPER LIMIT	REAL
	R_PTR	POINTER TO VIRTUAL MEMORY (ANAlog)	WORD
O	ENO	OUTPUT ENABLE	BOOLEAN
	LOW	BOTTOM LIMIT ALARM	BOOLEAN
	HIGH	UPPER LIMIT ALARM	BOOLEAN
	OUT	RELATIVE HUMIDITY OUTPUT	ANY_NUM

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

Equation 2- API FUNCTION

This block implements an equation according to the API standard whose specifications are presented in attached tables. These tables of petroleum measurement are used in calculations of amounts of crude oil and sub products of petroleum in reference conditions in any of the 3 systems of measurement widely used. These tables are supplied for standard calculations of measurements of petroleum fluids quantities in spite of the source point, destination or measurement units used by habit or laws.

A complete list of all new released ASTM-API-IP tables are the result of cooperation between the American Society for Testing And Materials, American Petroleum Institute and the Institute of Petroleum (London).

Control Word (CTW)- Select Inputs
This equation has four types of possible inputs where the user should select one of them.
Each option chooses a specific table.
ºAPI+ Temperature (ºF) ➔ see tables 5/6
Relative Humidity + Temperature (0F) ➔ see tables 23/24
Density + Temperature (ºC) ➔ see tables 53/54
Density + Temperature (ºC) ➔ see tables 59/60

Control Word (CTW)- Select Products
The type of product should be selected in this field. Available products are crude oil, generalized products, MTBE and lubricating oil. Once the inputs and product are selected, so the user can select the table that will be used in the calculations .

Control Word (CTW)- Select Output
The user should set the type of the output. There are 2 options:
- CCF
- VCF

Tables
Once the user has set inputs, product and output will have to be implicitly set a table of the API standard.

Input Scale Conversion Factors Conversion
The Input scale has two factors (parameters the user must set) to convert these parameters to engineering units (EU). IN1 input must be adjusted through the parameters K1 (GAIN) and K2 (BIAS). Similarly IN2 input must be set through parameters K3 (GAIN) and K4 (BIAS). After conversion they will be in the following format:

Density (EU) = IN1*K1+K2
Temperature (EU) = IN2*K3+K4
Pressure (EU) = IN3*LOW+HIGH

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	TYPE OF DENSITY	REAL
	IN2	TEMPERATURE	REAL
	IN3	PRESSURE	REAL
P	CTW	CONTROL WORLD	WORD
	K1	GAIN FOR DENSITY INPUT TO CONVERT TO EU AS INDICATED IN THE FOLLOWING TABLES	REAL
	K2	BIAS FOR DENSITY INPUT TO CONVERT TO EU AS INDICATED IN THE FOLLOWING TABLES	REAL
	K3	GAIN FOR TEMPERATURE INPUT TO CONVERT TO EU AS INDICATED IN THE FOLLOWING TABLES	REAL
	K4	BIAS FOR TEMPERATURE INPUT TO CONVERT TO EU AS INDICATED IN THE FOLLOWING TABLES	REAL
	K5	COEFFICIENT OF THERMAL EXPANSION AT 60^{0}F OR 15ºC (1/ºF OR 1/ºC) - (USED ONLY FOR MTBE)	REAL
	A	VCF	REAL
	B	F – COMPRESSIBILITY FACTOR	REAL
	LOW	GAIN FOR PRESSURE INPUT TO CONVERT TO EU AS INDICATED IN THE FOLLOWING TABLES	REAL
	HIGH	BIAS FOR PRESSURE INPUT TO CONVERT TO EU AS INDICATED IN THE FOLLOWING TABLES	REAL
	R_PTR	NOT USED	
O	ENO	OUTPUT ENABLE	BOOLEAN
	LOW	INPUT DATA WITHIN EXTRAPOLATION RANGE	BOOLEAN
	HIGH	INPUT DATA OUT OF EXTRAPOLATION RANGE	BOOLEAN
	OUT	OUTPUT: VCF VALUE OR INTERMEDIATE	REAL

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

Tables 5 & 6	Inputs		Outputs	
	API Gravity Range (ºAPI)	*Temperature Range(ºF)*	*Intermediate (table 5)*	*Final (table 6)*
			API at 60 ºF	*VCF at 60 ºF*
A-Crude Oil	0 up to 100	0 up to 300	0 up to 100 ºAPI	
B-Generalized products	0 up to 85	0 up to 300	0 up to 85 ºAPI	
C-MTBE	(*)	0 up to 300	0.00027 up to 0.00097($^{o}F^{-1}$)	
D-Lubricating oil	-10 up to 45	0 up to 300	-10 up to 45 ºAPI	

Tables 23 & 24	Inputs		Outputs	
	Relative Density Range	*Temperature Range(ºF)*	*Intermediate(table 23)*	*Final (table 24)*
			Relative Density at 60(ºF)	*VCF at 60 (ºF)*
A-Crude Oil	0.611 up to 1.076	0 up to 300	0.611 up to 1.076	
B-Generalized products	0.653 up to 1.076	0 up to 300	0.653 up to 1.076	
C-MTBE	(*)	0 up to 300	0.00027 up to 0.00097 ($^oF^{-1}$)	
D-Lubricating oil	0.800 up to 1.164	0 up to 300	0.800 up to 1.164	

Tables 53 & 54	Inputs		Outputs	
	Density Range (kg/m^3)	*Temperature Range(ºC)*	*Intermediate(table 53)*	*Final (table 54)*
			Density at 15 ºC (kg/m3)	*VCF at 15 º C*
A-Crude Oil	610 up to 1075	-18 up to 150	610 up to 1075	
B-Generalized products	653 up to 1075	-18 up to 150	653 up to 1075	
C-MTBE	(**)	-18 up to 150	0.000486 up to 0.001674 ($^oC^{-1}$)	
D-Lubricating oil	800 up to 1164	-20 up to 150	800 up to 1164	

Tables 59 & 60	Inputs		Outputs	
	Density Range (kg/m^3)	*Temperature Range(ºC)*	*Intermediate(table 59)*	*Final (table 60)*
			Density at 20 ºC (kg/m^3)	*VCF at 20 ºC*
A-Crude Oil	610 up to 1075	-18 up to 150	610 up to 1075	
B-Generalized products	653 up to 1075	-18 up to 150	653 up to 1075	
C-MTBE	(**)	-18 up to 150	0.000486 up to 0.001674 ($^oC^{-1}$)	
D-Lubricating oil	800 up to 1164	-20 up to 150	610 up to 1075	

(*) Coefficient of thermal expansion at 60 ºF
(**)Coefficient of thermal expansion at 15 ºC

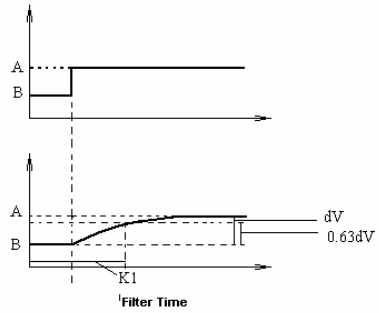
A few examples of calculation results:

Table	*D(IN1)*	*T(IN2)*	*Density at base*	*VCF*	*F*
5A/6A – Crude Oil – API+T(F) – **HYDRO CORRECTION**	30	200	21.2		
5A/6A – Crude Oil – API+T(F) –**NO Hydro**	40	35	42.2	1.0128	0.00000511
5A/6A – Crude Oil – API+T(F) –**NO Hydro**	30	150	23.9	0.96252	0.00000553
5A/6A – **Hydro Correction / No Correction**	30	80	28.6	0.9914	
5D/6D – Lubricating Oil - API+T(F)	30	80	28.8		
23B/24B – Generalized Products – Rel.Dens+T(F)	0.9	80	0.9075	0.9914	
53A/54A– Crude Oil – Dens+T(15C)	630	60	671.1	0.9377	
53A/54A– Crude Oil – Dens+T(15C) – **NO Hydro**	780	42	800.9	0.97395	0.00000103
53A/54A– Crude Oil – Dens+T(15C) – **NO Hydro**	830	50	855.3		0.00000096
53A/54A– Crude Oil – Dens+T(15C) – **NO Hydro**					
59A/60A– Crude Oil – Dens+T(20C)	630	42	650.5	0.9679	
59A/60A– Crude Oil – Dens+T(20C) - **Hydro**	920	50	939.0	0.97902	
59A/60A– Crude Oil – Dens+T(20C) –**Hydro**	905	-10	885.0	1.0233	
59A/60A– Crude Oil – Dens+T(20C) –**Hydro**	975	12	970.1	1.0052	
59D/60D– Lubricating Oil– Dens+T(20C) - **Hydro**	830	40	842.2	0.9850	
59A/60A– Crude Oil – Dens+T(20C) –**NO Hydro**	920	50	939.7	0.97908	
59A/60A– Crude Oil – Dens+T(20C) –**NO Hydro**	905	-10	884.4	1.02334	
59A/60A– Crude Oil – Dens+T(20C) –**NO Hydro**	975	12	970	1.0052	

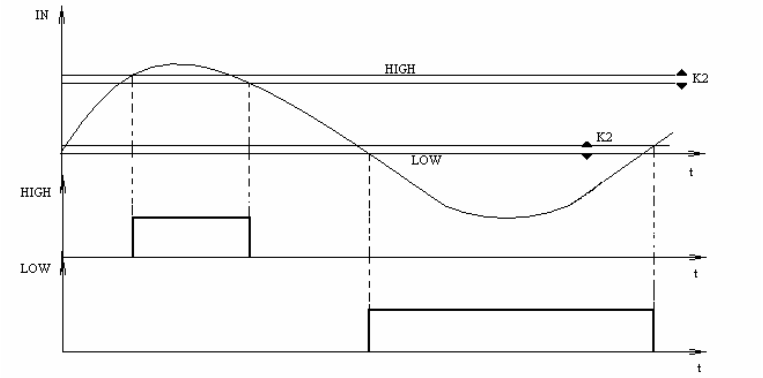
Equation 03 – SIGNAL PROCESSING

Description
Option Signal Processing uses an equation that filters the input signal. The filter used is a first order exponential filter. Input IN1 receives the signal.

Characteristic Filter Time (K1)
This parameter is the characteristic time filter in seconds. Consider a step input. When the output signal reaches 63 % of the step value, the time measured until this moment is defined as characteristic time.



Hysteresis K2 and High e Low alarms



When the input reaches the values set in HIGH, output HIGH will change to true until the input goes beyond HIGH-K2. Similarly, when the input reaches LOW, output LOW will go to true until the input goes beyond LOW+K2.

CLASS	MNEM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	IN1	processing signal	ANY_NUM
P	K1	characteristic time in seconds. it is a first order EXPONENTIAL filter	REAL
	K2	hysteresisalarm processing high and low. it must be a non negative valve.	REAL
	LOW	BOTTOM limit for alarm processing after the digital filter.	REAL
	HIGH	upper limit for alarm processing after the digital filter..	REAL
O	ENO	OUTPUT ENABLE	BOOLEAN
	LOW	BOTTOM limit alarm	BOOLEAN
	HIGH	upper limit alarm	BOOLEAN
	OUT	output after filter	ANY_NUM

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

PID - PID Controller

Description
The acclaimed PID algorithm for continuous process control, associated with the flexibility of the configuration of the operation settings through parameterization, allow use of this block to a variety of applications and control strategies.

This block supplies several options of settings of the algorithm having as a basis the terms Proportional (P), Integral (I) and Derivative (D) that may be applied in error or just to the process variable (PV).

The user may set limits of anti-reset windup (only applied to the integral term).

Plus, the user might choose the type of PID algorithm: ISA or parallel, direct action or reverse, manual to automatic transference bumpless or hard.

Control Word (CTW) - Type of PID
PI.D: P and I actions actuate over the error and the D action over the process variable. In this way the output signal tracks setpoint changes according to proportional an integral action, but there is no undesired variation due to the derivative action. It is the most recommended type for most applications with setpoint adjustable by the user.

PID: P, I and D actuate over the error so that the output signal is changed when there are changes in the processes variable or setpoint. It is recommended for relation control or to cascade slave control.

I.PD: In this type only integral action actuates over the error. Setpoint changes produce soft output signal variations. It is recommended for a process that cannot have sudden changes in the variable due to the setpoint change. It is the case of heating process with high GAIN.

Control Word (CTW) - Type of algorithm

$$\text{PARALLEL}: \text{OUT} = \text{K}_\text{p}.e + \frac{e}{T_R.s} + \frac{T_D.s.e}{1+\alpha.T_D.s}$$

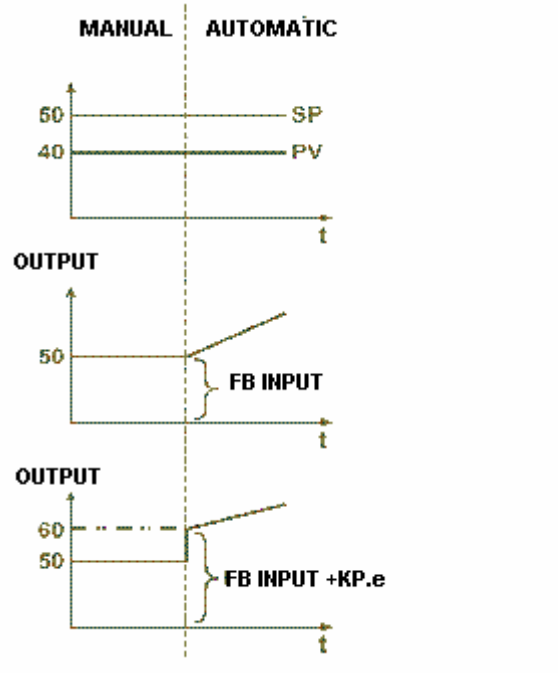
$$\text{ISA}: \text{OUT} = \text{K}_\text{p}\left[1 + \frac{1}{T_R.s} + \frac{T_D.s}{1+\alpha.T_D.s}\right].e$$

Control Word (CTW) - Type of action
Some processes require that the output signal (manipulated variable-MV) does not increase when the process variable increases, while most of the other applications require the opposite.

Type of action	Error	Effect
Reverse	e = SP – PV	Output decreases with the increase of PV
Direct	e = PV – SP	Output increases with the increase of PV

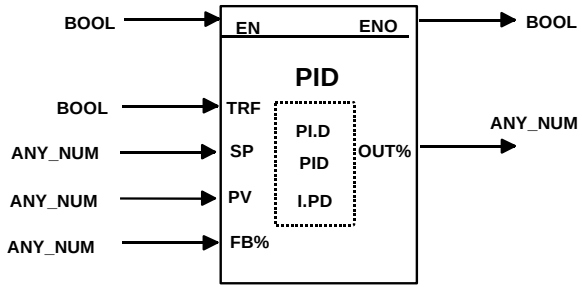
Control Word (CTW) - Type of transference from Manual to Automatic
Bumpless: During switching from manual to automatic, the PID block will start calculations from the last manual value, i.e., there is not a jump in the block output.

Hard: During switching from manual to automatic, the PID FB will supply as first value in automatic the last manual value plus the proportional term.



Anti Saturation by the integral term (AWL and AWU parameters)
Usually the control algorithm automatically stops the contribution of the integral mode when the output signal reaches the 0 or 100 % limits. Contributions of proportional and derivative are not affected.
A unique characteristic of the algorithm is the possibility to set these limits. For example in narrowing those limits through the AWL and AWU parameters, we can get faster answers while avoiding overshoot in the heating process.

PID Constants (KP, TR, TD and BIAS)
KP –Proportional Gain
TR – Integral time in minutes per repetition, so, the larger is this parameter the shorter is the integral action. It can be interpreted as the necessary time to the output being incremented or decreased of the error value (Parallel PID), keeping it constant.
TD –Derivative Time given in minutes. The derivative time is calculated using a false derivation, i.e., an action similar to a lead/lag controller, in which the lag constant is Alfa*TD. In this block implementation the Alfa factor is equal to 0.13.
BIAS –This parameter will allow the adjustment of the initial output value when the control is transferred from manual to automatic. This cannot be done if the FB output is connected.

PID	PID CONTROLER
	

CLASS	PARAM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	TRF	SELECTS MANUAL OR AUTOMATIC WORKING	BOOLEAN
	SP	SETPOINT	ANY_NUM
	PV	PROCESS VARIABLE	ANY_NUM
	FB%	IF TRF=1, INPUT CONNECTED TO FB IS PASSED TO THE OUTPUT	ANY_NUM
P	CTW	CONTROL WORLD	WORD
	KP	PROPORTIONAL GAIN	INT/100
	BIAS	BIAS	INT/100
	AWL	ANTI-RESET BOTTOM FINAL LIMIT	INT/100
	AWU	ANTI-RESET UPPER LIMIT	INT/100
	TR	INTEGRAL TIME (Min/Repetitions)	REAL
	TD	DERIVATIVE FACTOR (Min)	REAL
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT%	OUTPUT (MANIPULATED VARIABLE)	ANY_NUM
V	ER0	EXPECTED ERROR IN THE PROCESS	INT/100
	PV0	EXPECTED VARIABLE IN THE PROCESS	INT/100
	FB0	EXPECTED FEEDBACK VALUE	INT/100
	B0	EXPECTED BIAS VALUE	LONG
	IT0	EXPECTED INTEGRAL TIME (Min/Rep)	REAL
	DR0	EXPECTED DERIVATIVE FACTOR (Min)	REAL

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

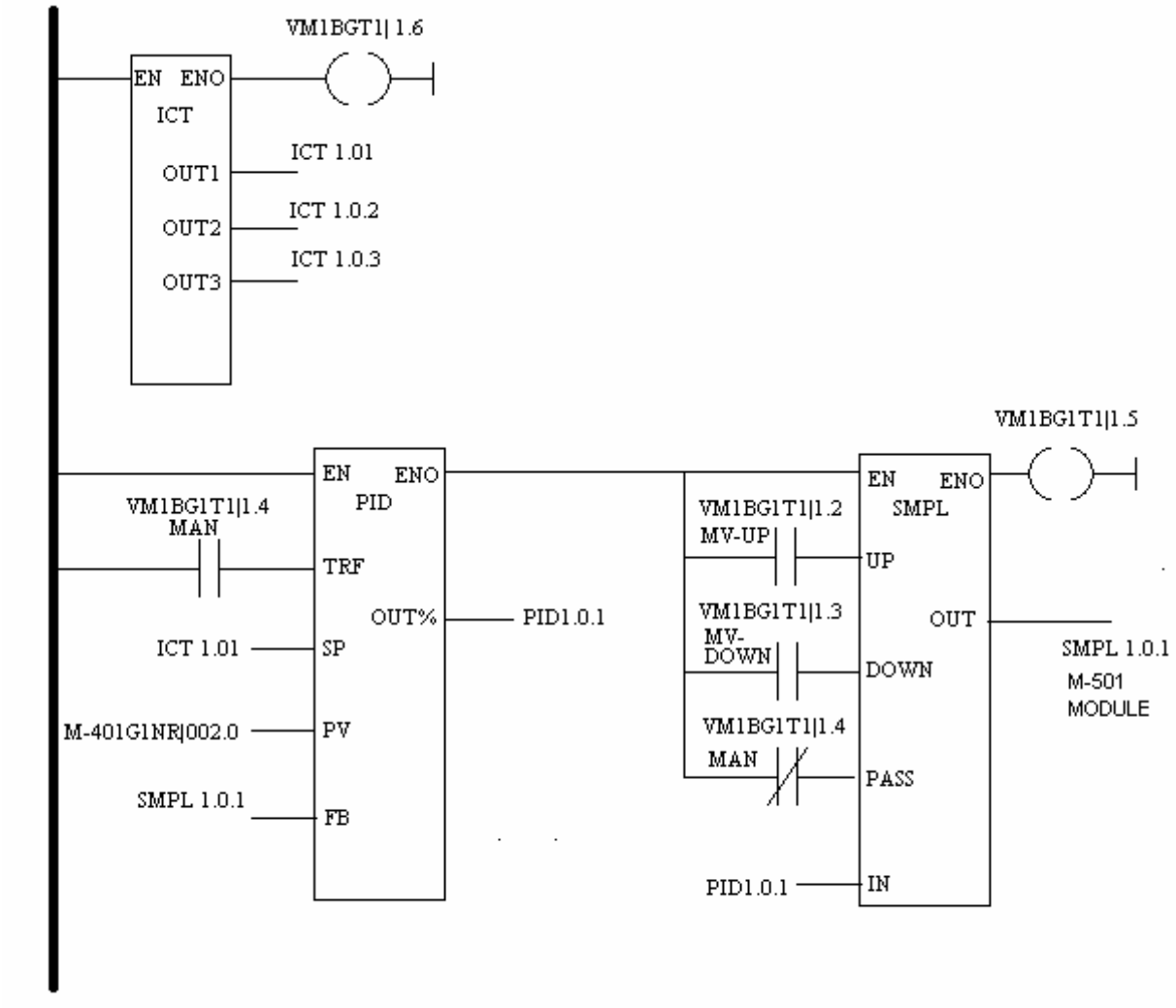
Bit sequences for the CTW Parameter

Only Settings								Auxiliary and parameters passing							
15			12	11	10	9	8			5	4		2	1	0

Auxiliary and parameter's passing

Bit 0 - State of Input Boolean EN
Bit 1 - State of Input Boolean TRF (0 = Auto; 1 = Manual, tracking)
Bit 2 - State of Input Boolean ENO
Bit 4- State of the feedback auxiliary variable (1 = tracking)
Bit 5 - Working State (0 = first time, 1 = running, not for the first time)

Control Loop with local setpoint and A/M station



Characteristics of the Configuration:

- Local setpoint adjustable through PRM1 parameter of the ICT block.
- Process variable obtained from the M-401-DR module, (see the diagram above), in the range of 0 a 10000.
- Mode control automatic/manual through virtual variable (MAN).
- SMPL plays the role of a manual/automatic station allowing increment or decrease (UP and DOWN inputs) of the output when in manual mode.

STATUS - System Status

Description
This block will allow the configuration of 8 boolean variables to inform status of an I/O module, a remote I/O module, or the communication ports of the CPU-C3. The user has 4 options of class to select.

Control Word (CTW)- Select Class
The user must select which is the class for the slave equipment between the options below:

- IO_Master: An I/O module connected in the same rack where the CPU is located.
- IO_RIO: It is an I/O module connected to a RIO-700-3.
- COMM_RIO: Communication status between CPU and RIO-700-3.
- Cpu_port: Communication status of CPU-C3 ports (P1, P2 and P3). Indicates activity in the communication port.

Control Word (CTW)- Select Sub Class and Select item
After choosing class, the user should select sub class and item. Class.sub_class.item

- IO_master.rack.slot: it must be informed rack and slot where the desired module is located.
- IO_RIO.rack.slot: It must be informed rack and slot where module is located.
- COMM_RIO.RIO: It must be informed which remote I/O.
- CPU_PORT.PORT: It must be informed which CPU port (P1, P2 or P3) it is required to monitory communication.

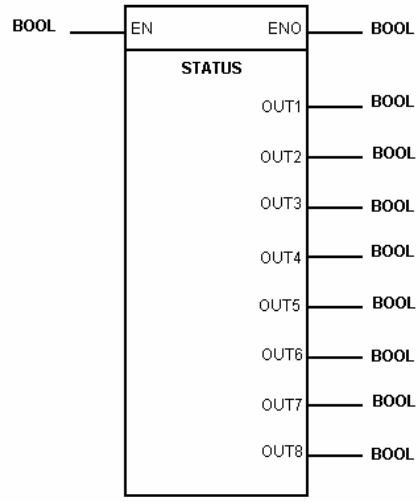
Status meaning and outputs
According to the choice, the meaning of the status block outputs is:

IO_Master e IO_RIO:
0: Status= module I/O “bad” .
1: Status= module de I/O “good”.

COMM_RIO:
0: “Communication failure”.
1: “Communication without errors”.

CPU_PORT:
0: Port not communicating..
1: Port Communicating.

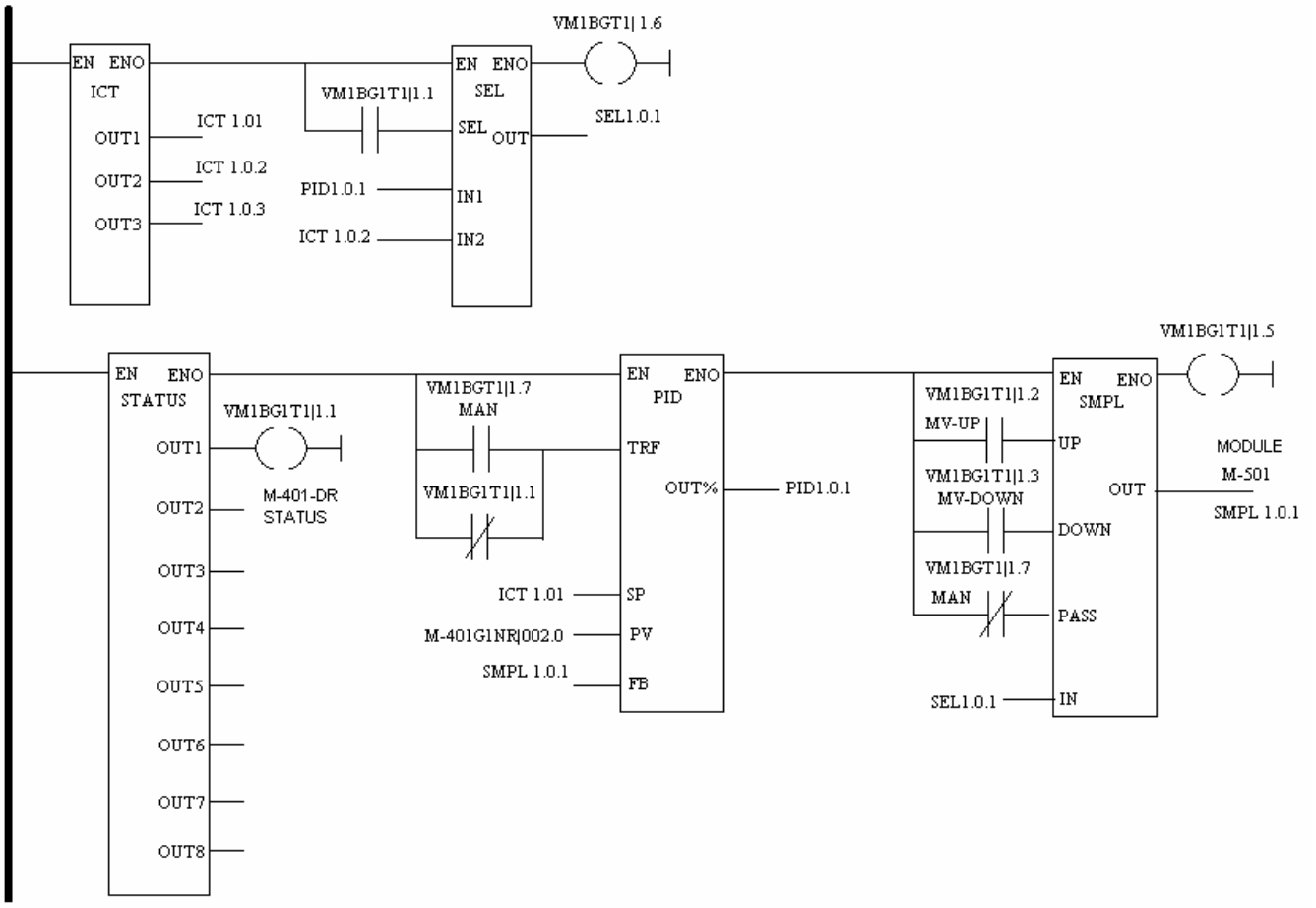
STATUS	SYSTEM STATUS



CLASS	MNEM	DESCRIPTION	TYPE
I	IN1	INPUT ENABLE	BOOLEAN
P	SC1	CLASS	BYTE
		SUB CLASS	BYTE
		ITEM	2 BYTES
	SC2	CLASS	BYTE
		SUB CLASS	BYTE
		ITEM	2 BYTES
	SC3	CLASS	BYTE
		SUB CLASS	BYTE
		ITEM	2 BYTES
	SC4	CLASS	BYTE
		SUB CLASS	BYTE
		ITEM	2 BYTES
	SC5	CLASS	BYTE
		SUB CLASS	BYTE
		ITEM	2 BYTES
	SC6	CLASS	BYTE
		SUB CLASS	BYTE
		ITEM	2 BYTES
	SC7	CLASS	BYTE
		SUB CLASS	BYTE
		ITEM	2 BYTES
	SC8	CLASS	BYTE
		SUB CLASS	BYTE
		ITEM	2 BYTES
O	ENO	OUTPUT ENABLE	BOOLEAN
	OUT1	STATUS OF SC1	BOOLEAN
	OUT2	STATUS OF SC2	BOOLEAN
	OUT3	STATUS OF SC3	BOOLEAN
	OUT4	STATUS OF SC4	BOOLEAN
	OUT5	STATUS OF SC5	BOOLEAN
	OUT6	STATUS OF SC6	BOOLEAN
	OUT7	STATUS OF SC7	BOOLEAN
	OUT8	STATUS OF SC8	BOOLEAN

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

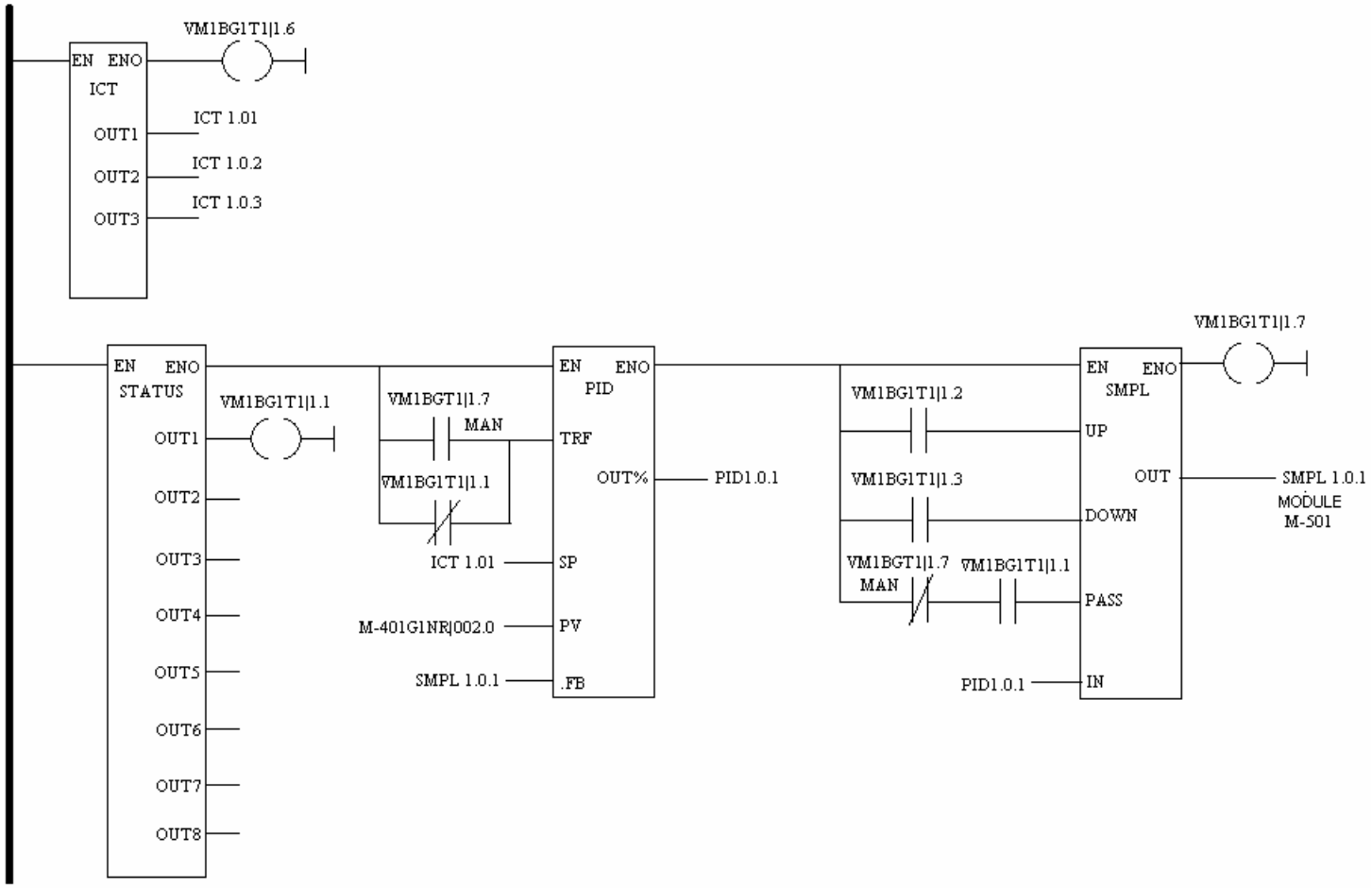
Control Loop with local setpoint, A/M station with safety value when the M-401-DR status is "bad"



Characteristics of the Configuration:

- Local setpoint adjustable through PRM1 parameter of the ICT block.
- Process variable obtained from the M-401-DR module, (see the diagram above), in the range of 0 a 10000.
- Mode control automatic/manual through virtual variable (MAN).
- SMPL plays the role of a manual/automatic station allowing increment or decrease (UP and DOWN inputs) of the output when in manual mode.
- M-401-DR module status is checked. In case it has a failure, the block output changes the status of the output OUT1 to "bad". A SEL block (binary selection) has as inputs the PID block output and a constant generated by the ICT block. So, when there is a failure a safety value is sent to the output.

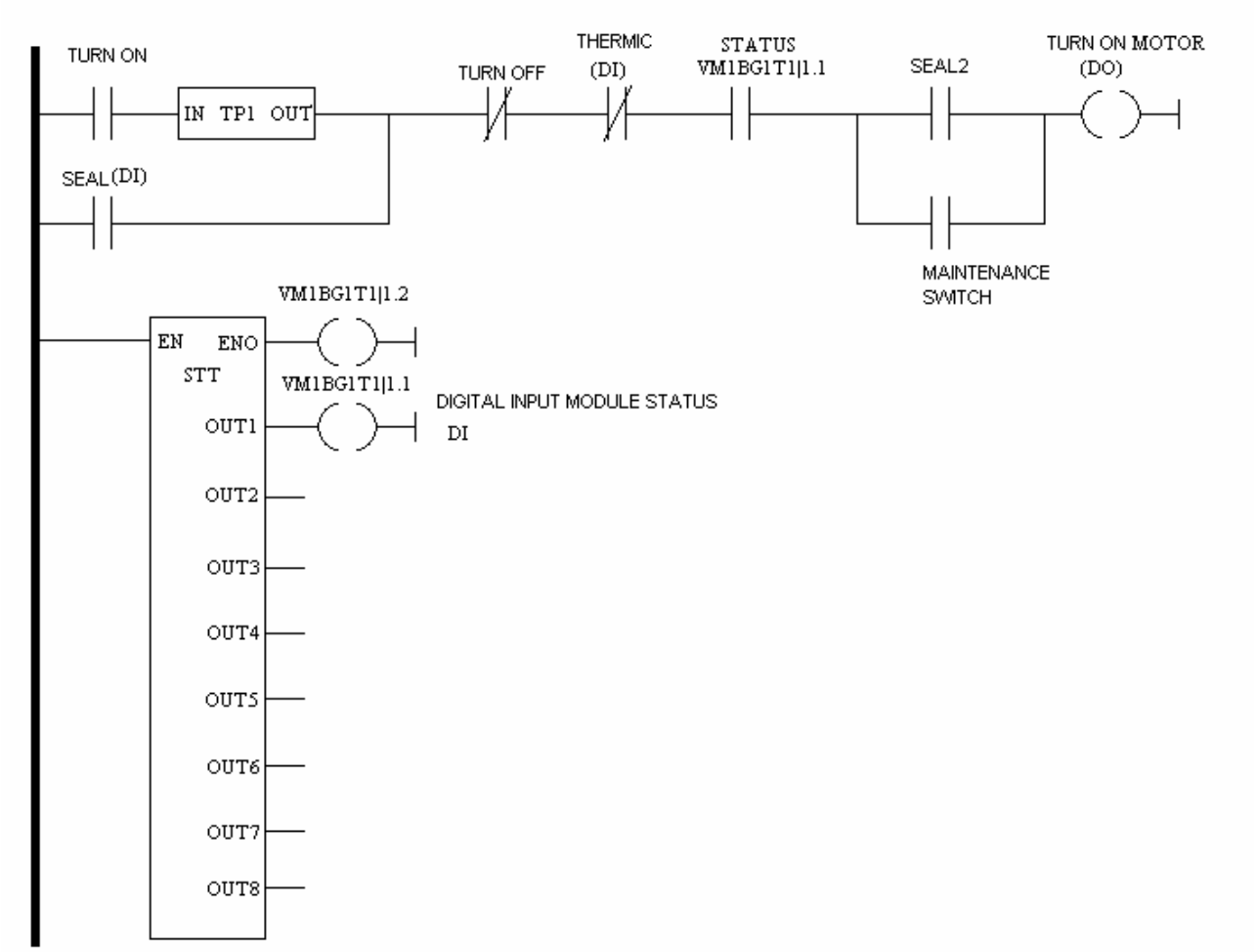
Control Loop with local setpoint, A/M station and status indication of M-401-DR module



Characteristics of the Configuration:

- Local setpoint adjustable through PRM1 parameter of the ICT block.
- Process variable obtained from the M-401-DR module, seen in the above picture, in the range of 0 a 10000.
- Mode control automatic/manual through virtual variable (MAN).
- SMPL plays the role of a manual/automatic station allowing increment or decrease (UP and DOWN inputs) of the output when in manual mode.
- M-401-DR module status is checked. In case it has a failure, the block output changes the status of the output OUT1 to “bad”. The SMPL block selects the control to manual and the output is frozen with the last value that had the “good” status.

Motor starting with TURN ON and TURN OFF commands and safety contacts including the status of the digital input module



TURN ON	*TURN OFF*	*SEAL(DI)*	*THERMAL (DI)*	*STATUS(A)*	*SEAL2*	*SWITCH/ MAINTENANCE*	*MOTOR (DO)*
Turns the MOTOR on through an auxiliary variable	Turns the MOTOR OFF through an auxiliary variable	Seal for turning ON the motor, it holds the state TURN MOTOR ON.	Alarm indicating the motor temperature reached a limit.	Module status of the digital input. Failures in this module turn the off the motor.	Auxiliary Control	Auxiliary Control	
1	0	1	0	1(**)	0/1(*)	0/1(*)	TURN ON
X	1	X	X	X	X	X	TURN OFF
X	X	X	1	X	X	X	TURN OFF
X	X	X	X	0(**)	X	X	TURN OFF

X- Redundant State
(*)- These controls are manual switches operated manually. Two contacts form an OR logic gate, so the output will be enabled if SEAL2 or Switch/Maintenance are TRUE (1).
(**) – STATUS = 1 means communication without failures
STATUS = 0 means incorrect ID or module not present.

STP - Step Control

Description
This functions use is combined with the PID block. Connect a PID block output to the DMV input to make an ON_OFF or ON_NONE_OFF control. ON_OFF establishes the OPEN and CLOSE control of valves during a particular time interval ON_NONE_OFF allows the OPEN and CLOSE control of valves according to the rate of variation in the PID output or DMV input.

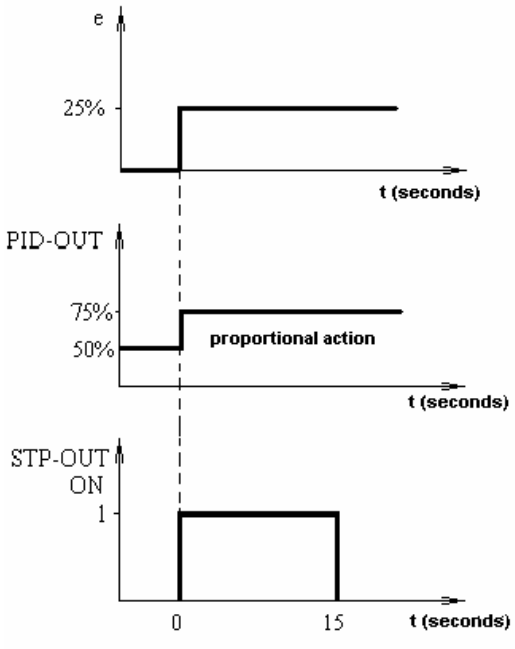
Valves opening time VOT
This parameter must be adjusted with the approximated time necessary to open the valve totally or close it totally.

Minimum Pulse Width
The user should configure minimum pulse width per 0.1 seconds through WPL parameter and time for total excursion of the control element.

Control Word (CTW)- Control Type
The user must choose the control type, i.e., ON_OFF or ON_NONE_OFF

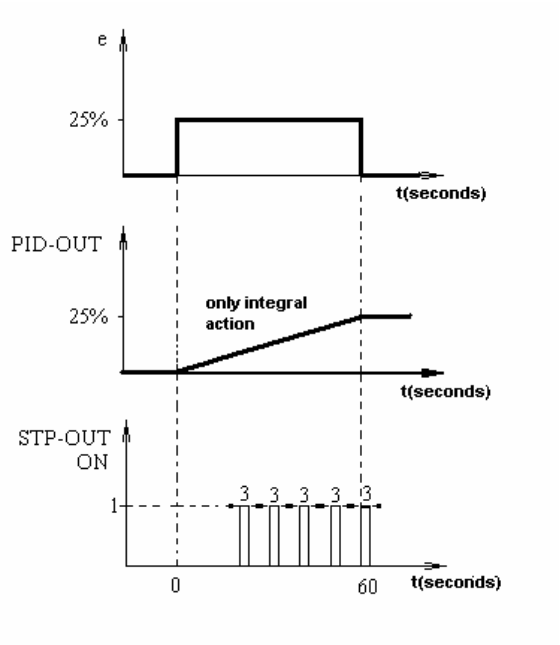
- **ON_OFF Control**
  Suppose the ON (OPEN) output starts a motor that opens a valve while OFF (CLOSE) output starts a motor that closes this valve. The STP block allows these control pulses to be generated. In this control mode, the block compares DMV with internal values. If DMV is greater than 55 % the ON output goes to TRUE state and the OFF output goes to FALSE state. If it's smaller than 45 % the OFF output goes to TRUE state and the ON output goes to FALSE state. Values between 45 % and 55 % make the ON and OFF outputs remain on the last state **.**

- **ON_NONE_OFF Control**
  A PID having only proportional action with gain KP=1 and VOT equal to 1 minute. Suppose that in the instant t= 0 a step with error 25% is applied. Thus, the valve opening is 25 % of 1 minute, or 0.25*TR= 15 seconds. Figure below shows this example:

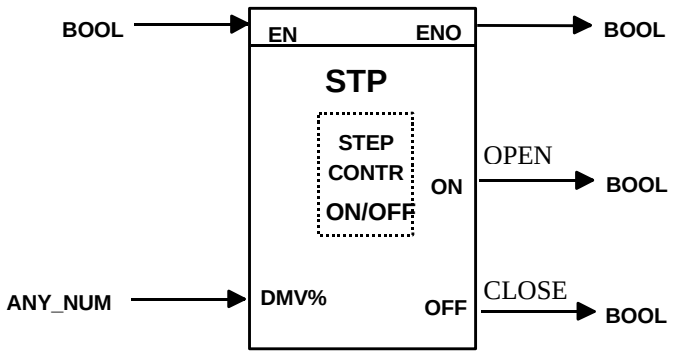


Integral action of a PID is equal to a series of pulses with size WPL and frequency determined by the integral time of the PID block (TR) and by the control deviation. Pulse frequency is given by the TR value. WPL is fixed and determined during the block setting. Suppose TR= 1 minute and WPL= 3 seconds and a step with error 25 % is applied in the input. A standard controller would increase or decrease output of 25 % on 1 minute (TR). To make the valve have an opening time (VOT) equal to 1 minute we need 15 seconds (25 % of 60 seconds), because WPL=3 seconds. So, 5 pulses with width equal to 3 seconds are required. Output remains in this mode of functioning while the PID

output keeps the same rate of change.



STP	STEP CONTROL



CLASS	PARAM	DESCRIPTION	TYPE
I	EN	INPUT ENABLE	BOOLEAN
	DMV%	BLOCK INPUT	ANY_NUM
P	CTW	CONTROL WORLD	WORD
	WPL	MINIMUM PULSE WIDTH PER 0.1 SECONDS	INT
	VOT	VALVE OPENING TIME IN 1	INT
O	ENO	OUTPUT ENABLE	BOOLEAN
	ON	HIGH LEVEL OUTPUT (OPEN)	BOOLEAN
	OFF	LOW LEVEL OUTPUT (CLOSE)	BOOLEAN
V	MVB	PREVIOUS MV	INT
	C_TIME	PULSE HOLD	INT
	DEBT	DEBT ACCUMULATED	INT

I	***Input***
P	***Parameter***
O	***Output***
V	***Variable***

Bit sequence for the Parameter CTW

Only Settings							Auxiliary and parameters passing								
15							8	7				3	2	1	0

Auxiliary and parameter passing

Bits that indicate status:

Bit 0 – State of Input Boolean EN
Bit 1 - State of Input Boolean ENO
Bit 2 - State of Output Boolean OPEN (1 = OPEN; 0 = NONE)
Bit 3 - State of Output Boolean CLOSE (1 = CLOSE; 0 = NONE)
Bit 7 – Last state of EN

Chapter 3

THE CONF700

Introduction

This chapter presents the essentials for the use of the CONF700 programming software for the advanced Smar LC700 Universal Hybrid Controller. It will show how to create, download and troubleshoot an LC700 configuration.

The user should read about the LC700 in chapter 1 “Ladder Logic". In chapter 2 "Function Blocks” to get familiar with the ladder elements and function blocks. Next the user will be able to design a control strategy for a specific application.

The CONF700 application software is based on 32 bits Microsoft Windows and is therefore operated in the same basic way as other Windows applications, i.e. through menus, browsing, cut and paste, buttons, drop down lists, etc. It is assumed that the user is already familiar with Windows interface.

This manual will also show how to generate and register the LC700 Tag List in the computer running the LC700 OPC Server.

The LC700 OPC Server operation and setup will also be reviewed here.

Installation

Operating System

The CONF700 runs on any 32bits Windows. Therefore, it is ready for Windows 2000 and Windows XP.

Before Installation Begins

Verify minimum resources as listed below. It is recommended (and sometimes mandatory), that the user stop any Anti-Virus and some display controller applications. It is best to close all applications before installing the new software.

Minimum Resources for the CONF700

- Pentium IV processor, or higher (or AMD, Athlon, Duron)
- RAM memory with 256 MB or more
- Hard disk with a minimum free space of 550 MB
- Microsoft Windows 2000 (Service Pack 2) or Windows XP
- A serial port or Ethernet Adapter Card to communicate with the LC700 controller

Installing

The installation should automatically start a few seconds after the CD-ROM is inserted in a CD-ROM driver. If after inserting the CD-ROM into the driver the installation does not start automatically, go to the directory containing the application and run the SETUP executable file. The installation program will run and guide the user through out the installation procedure.

Using the CONF700

Launching the application

To start the LC700 programming software click the START button. Go to PROGRAMS and then find the SMAR group while placing the mouse pointer over it the user will see one or more buttons with Smar applications, then click the CONF700 application icon.

First a registration window will appear, click OK to proceed. Later the user might come back to this screen by using the menu: Help/About Conf700.

Next select “New Configuration” on the Dialog Box shown below for a new configuration or “Open an Existing Configuration” to open an old file.

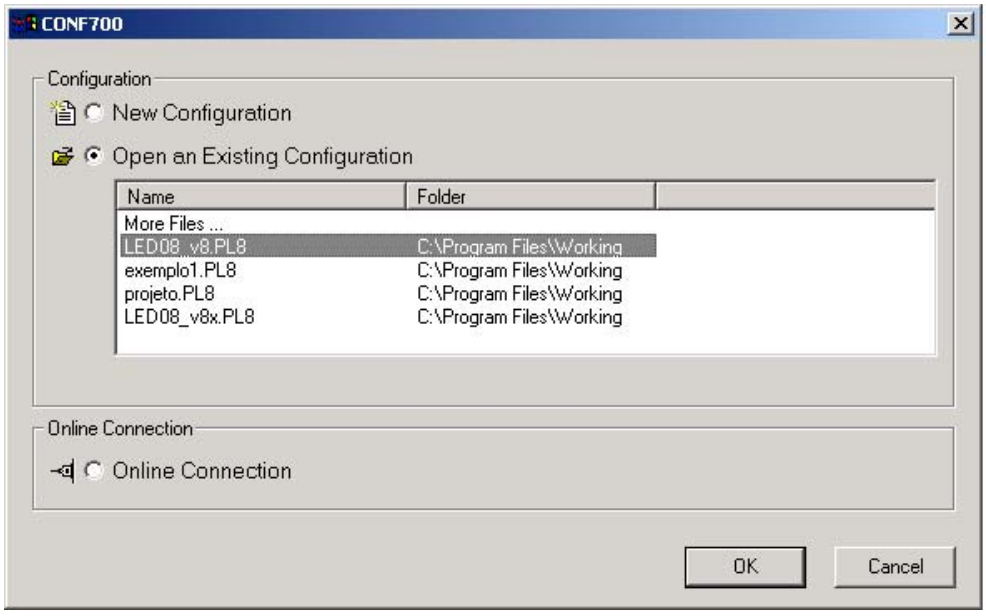


Figure 3.1- Launching CONF700

One configuration has to be created for each LC700 system. An LC700 system is composed of one CPU module, one or more I/O modules and 0 to 6 Remote I/O interfaces with respective I/O modules. This means, a project with many LC700 CPUs will have one configuration file for each CPU.

In case of redundant CPUs, both have to carry the same configuration.

Project Information

When launching the CONF700 “First Page” also referred to as, the “Documentation Page”, it will automatically appear. Fill the table with all of the pertinent information and the most important part here is to select the appropriate CPU version before starting configuration. Typically the CPU the user has received is the latest version available, but one way to make sure is to connect the LC700 CPU to a serial port of the PC and go to the Online mode. See Connecting to the LC700 topic for more details.
The user can return to the documentation page and make changes at any time using the menu Configuration/First Page or click  .

This information is valuable for project organization and documentation. Many of the printable reports generated by the CONF700 will contain this information.

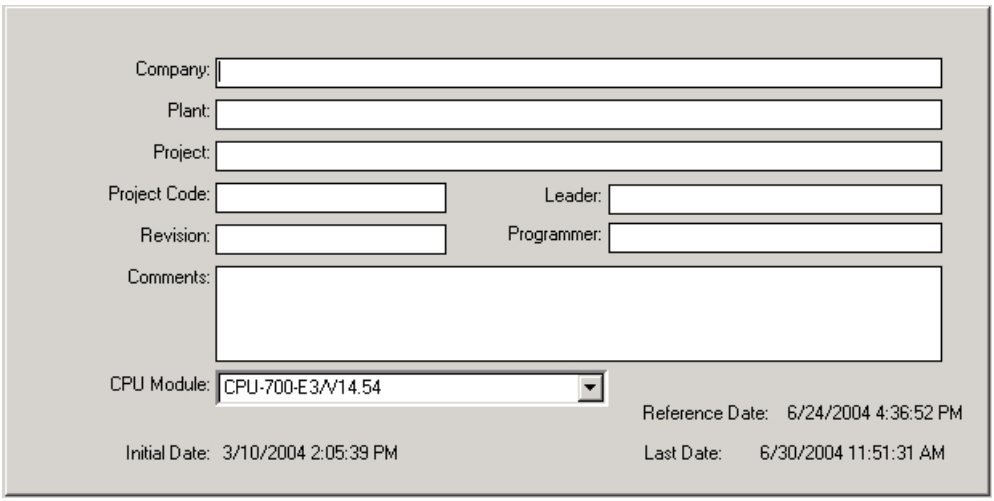


Figure 3.2- Project Information Window

Working Directory

CONF700 allows the user to set the working directory. This directory will be used as the default directory for saving configurations. Also when the user uploads a configuration from the CPU-700, CONF700 will automatically save the uploaded configuration in this directory.

To set the Working Directory click at Tools→Preferences and select the tab Directory, as shown in the Figure 3.3.

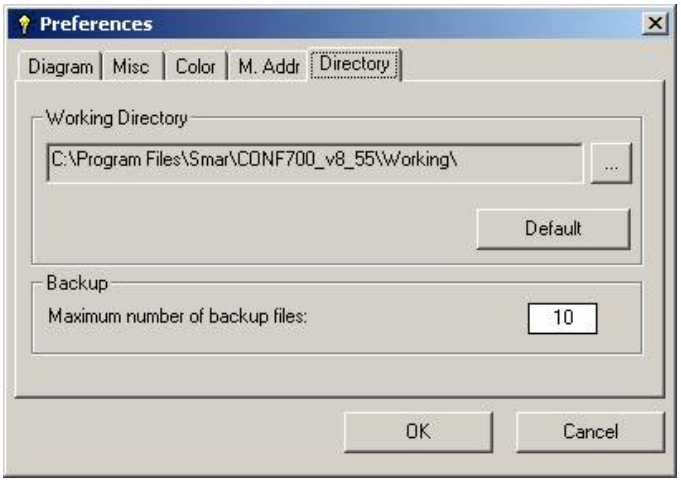


Figure 3.3-Selecting the Working Directory

Click the button "…" to change the directory.

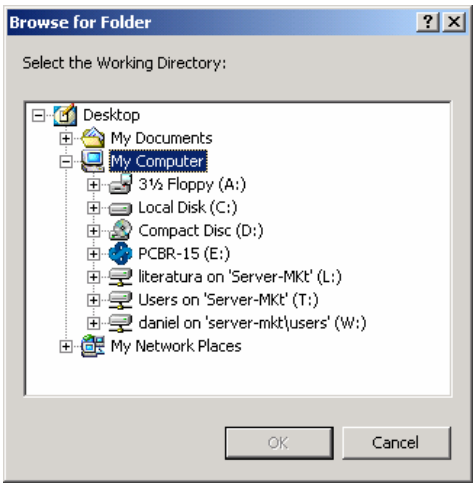


Fig 3.4- Selecting Window

The button “Default” will reset this path to the default path to C:\Program Files\Smar\CONF700_v8-54\Working\.

CONF700 generates backups for the configuration when the changes are saved. The extension for the backup file is “B.xx”, where “xx” means the sequential number of the generated backup files, for example: the first backup file has the B.01 extension, the second backup file has the B.02 extension, and so on. The quantity of backup files is limited in the field showed in the first picture of this item and also is limited by the physical space of the Hard Disk. For using the backup file, the user must rename the file for “PL.8” extension.

Setting up the I/O Modules

In the last step the user selected the CPU version to be used. Now the user will need to completely define the LC700 system hardware. To enter the “Module Page” the user may go to the menu Configuration/Module Page or click  .

For a new configuration this page will begin with a 4 slot Rack carrying a Power Supply on slot 0 (zero) and a CPU Module on slot 1 (one). Slots 2 and 3 are initially empty.

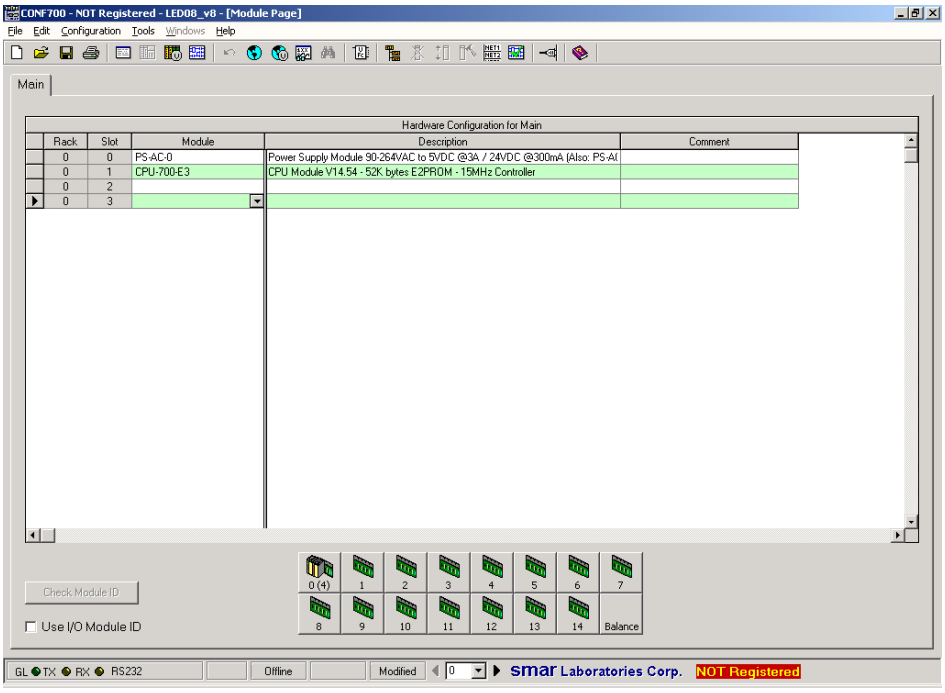


Figure 3.5-Setting the I/O Modules

Adding Modules

To add Modules go to an empty cell on the “Module” column and click on it. A drop down arrow will appear to the right of the cell. Now click on the down arrow and make a Module selection by clicking on it.

As soon as the Module is selected it will be added to the corresponding empty slot and also (very important), CONF700 automatically creates memory allocation for the I/O points. The user no longer needs to manage memory allocation as before on most of the old existing systems.

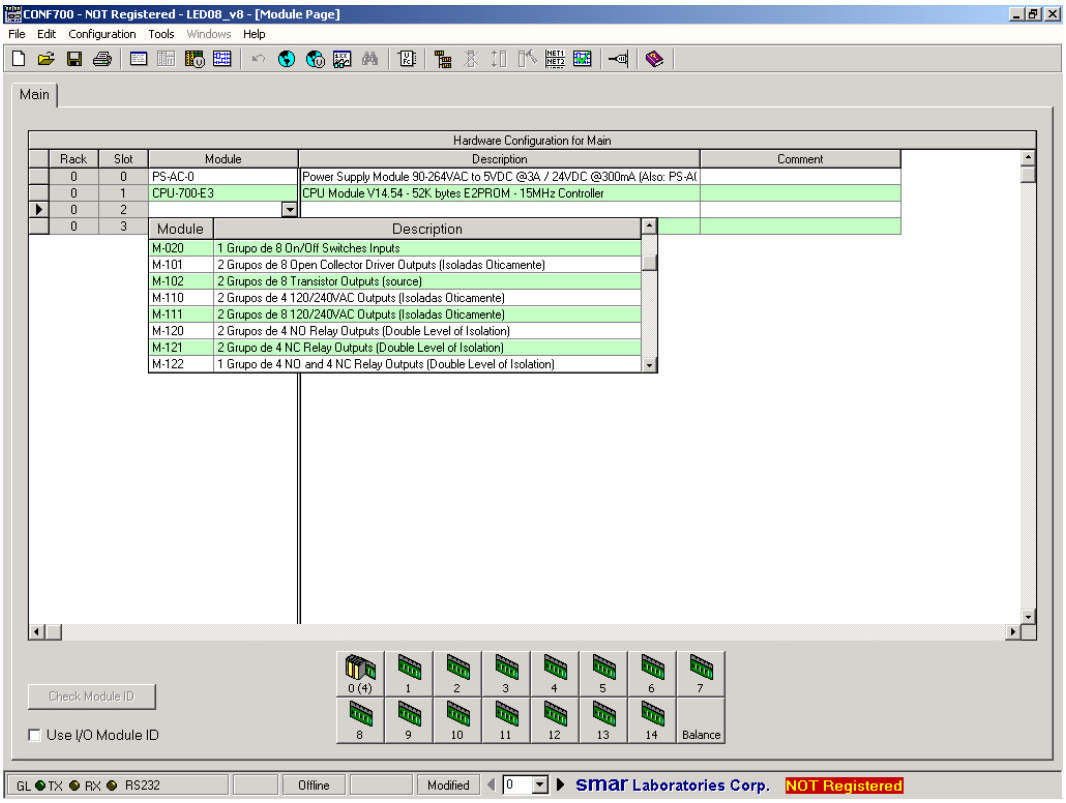


Figure 3.6- Adding Modules Window

Special Modules

The user will notice that some Modules require extra configuration and a special dialog box will automatically be launched as soon as the Module is selected. Some are:

- M-401DR - Analog Input
- M-501 - Analog Output
- M-402 - Temperature input
- FB-700 - Fieldbus Module

For example, the Temperature Module (M-402) will launch a configuration dialog box where the user can set how the inputs are going to work.

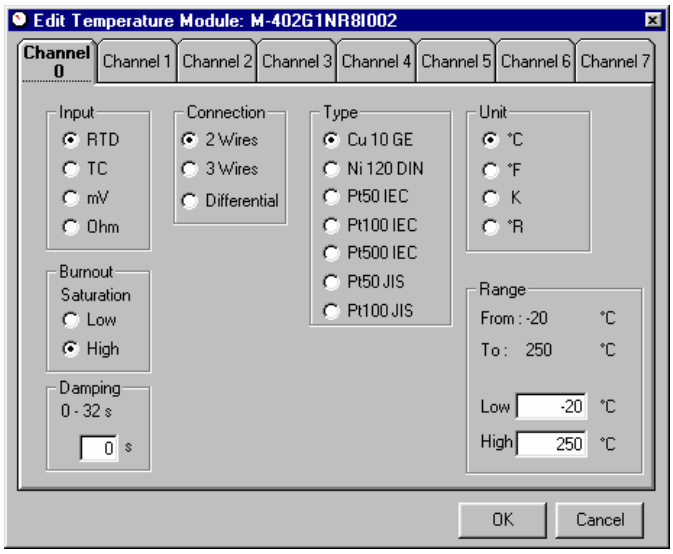


Figure 3.7- M-402 Temperature I/O Module Configuration on CONF700

Configuration and Hardware Consistency

It is extremely important that the Rack number and the position of each Module in the configuration match the actual hardware assembly. Many of the Modules on an LC700 system are not intelligent and the CPU cannot know if they are misplaced or non existent.

More elaborate modules, like the M-402, will cause warning messages from the FB700, if the actual positioning does not match the software configuration.

It is strongly recommended that you printout the hardware configuration and use it to check the actual installation.

It is necessary to go to menu File/Print and select options as indicated in the next figure then click the OK button for an initial printout of your project.

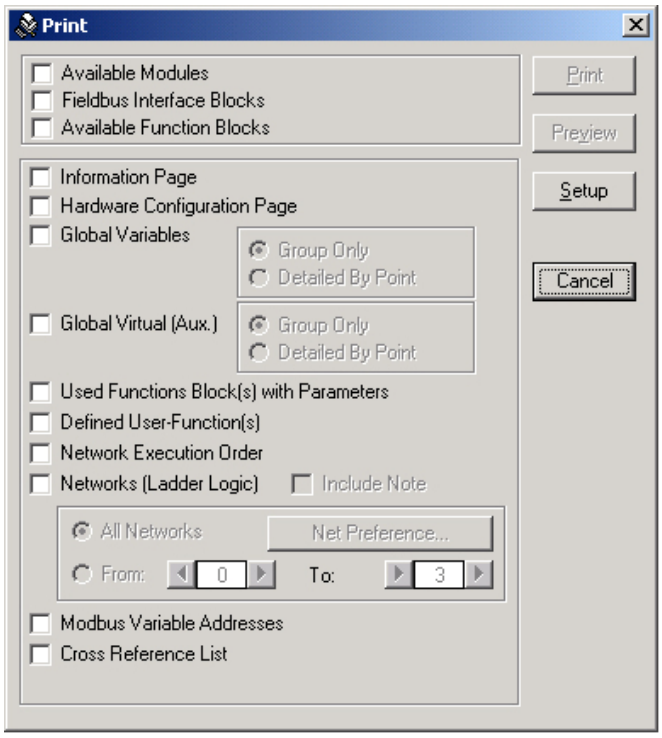


Figure 3.8- The Print Settings Window

Editing the I/O Modules

Modules can be Deleted, Replaced (deleted and placed) or simply moved to any other slot in the LC700 system even to the slot of a Remote I/O.
It is necessary to highlight the Module to be edited. Click the column to the left of the row where the Module is located. Note: some icons on the tool bar are now enabled for use.

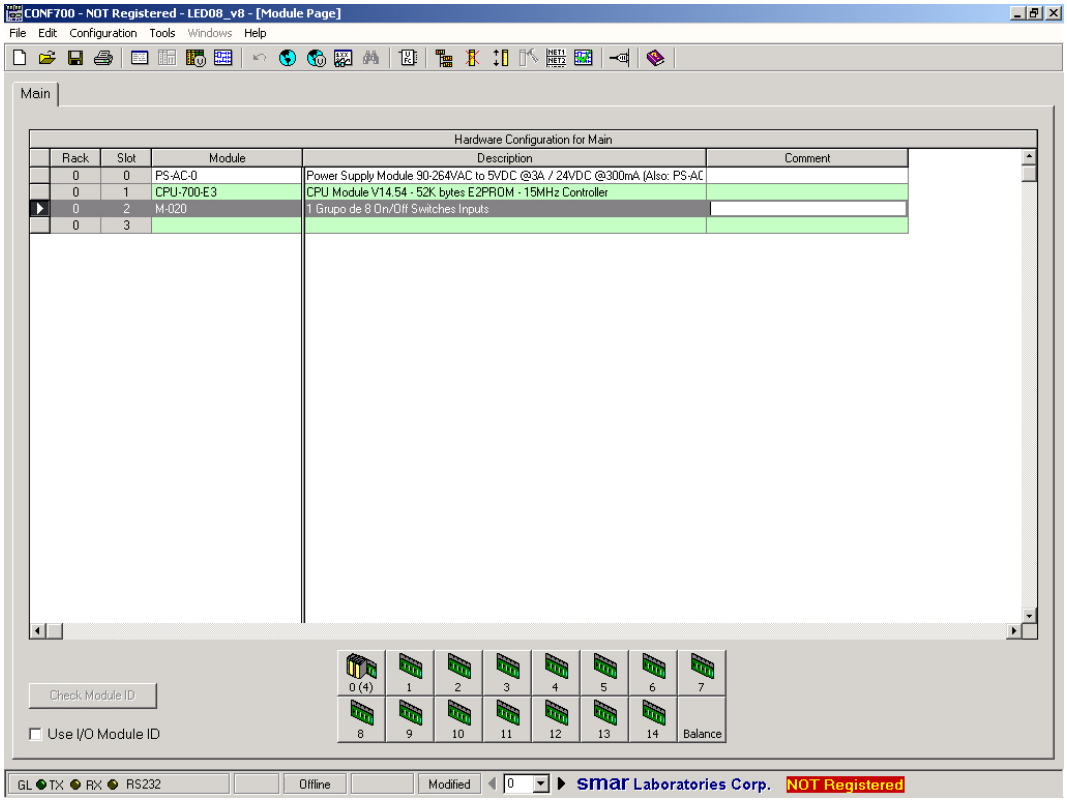


Figure 3.9- Editing I/O Modules Window

 To delete a Module

 To move a Module

 To characterize a Module

Special I/O Modules

Many of the LC700 I/O modules can be specially configured to be adapted to application requirements. These types of modules give the user an extra degree of flexibility. Typically modules that are related to analog signals, special sensors and those that acquire/send signals using popular protocols such as Modbus, DeviceNet, Profibus and Foundation Fieldbus have a dedicated level of configuration.

Configuring the M-401-DR Modules

This module provides 8 analog inputs and is typically designed to read signals coming from 2 or 4 wire analog transmitters.

The inputs are configured individually in order to read:
- 10 V, 5 V, 0 to 5 Vdc or 1-5 V, with the internal shunt resistor in the V position.
- 20 mA, 0-20 mA, 4-20 mA, with the internal sunt resistor in the I position.

As soon as the M-401-DR is added to an empty slot the following dialog box appears. Each channel must be independently configured. The CPU will receive the input signal already converted in a percentage on the input range selected.

This number comes in the range from 0 to 10,000. This number has the meaning of percentage with an imaginary fixed point to separate the last 2 decimal digits. For example 5000 represents 50.00 % while 10000 represents 100.00 % of the inputs selected range.

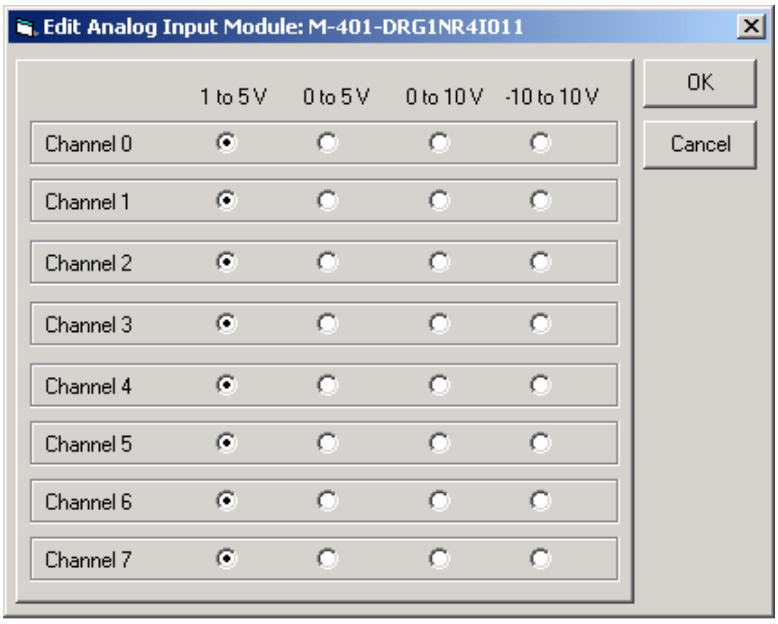


Figure 3.10- Setting the M-401-DR Inputs On CONF700

Configuring the M-402 Temperature Module

Each individual input in the M-402 has a particular configuration.
When the dialog box for the module configuration opens we can set the type of measurement, wiring connectivity, sensor type, engineering units to work and range that will be converted in percentage.

The M-402 module provides both a value (integer) and a status (Boolean). The status indicates that there is a sensor burnout, high or low as selected in the configuration. This status may be used to alert the operator and also be used for failure to make a decision in the interlock logic.

Two groups will be associated to this module:
- M-402G1B8Irrm.c, a group with 8 Boolean points where each one represents the “burn-out state” to the individual inputs.

- M-402G2NR8Irrm.c, a group with 8 Integer points representing the percentage of each individual input signal.

Each channel is composed by 3 terminals, they are identified by “A”, “B” and “C”.

- “A” input 1,
- “B” input 2,
- “C” common, it is shared by 2 inputs, as follows

  - Channels 1 and 2 use the common of the terminal 3A.
  - Channels 3 and 4 use the common of the terminal 8A.
  - Channels 5 and 6 use the common of the terminal 3B.
  - Channels 7 and 8 use the common of the terminal 8B.

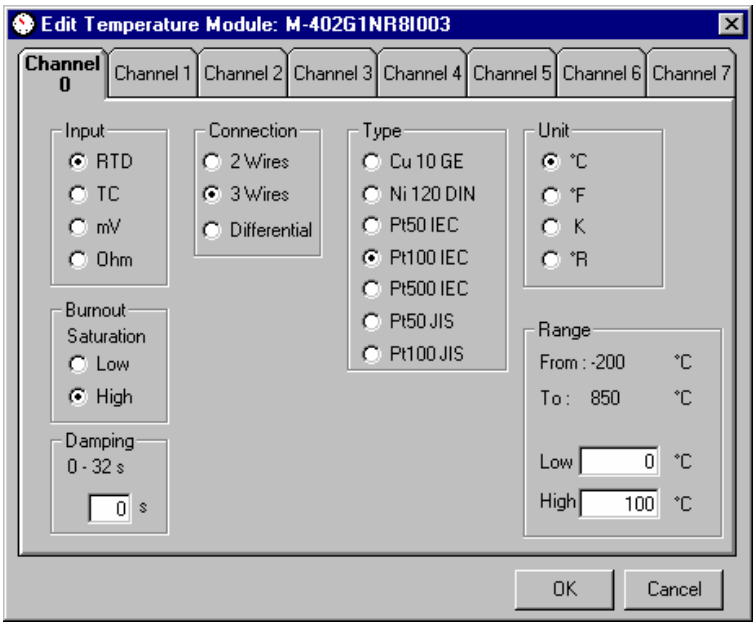


Figure 3.11- Setting the M-402 Inputs On CONF700

Configuring the M-501 Module

This is an analog output module with 4 current outputs and 4 voltage outputs representing the same signals. The module outputs 4 distinct signals in current and voltage mode at the same time.

Current signals are more immune to noise and other interference then is recommended for long distance and industrial environments. While voltage signals are for connectivity with devices in close proximity such as controllers, indicators, paper registers and recorders.

Note in the configuration dialog box that current and voltage ranges are connected. When one is configured the other is automatically determined.

For voltage ranges, the module hardware comes with an internal DIP Switch already configured for the 5V range (row one in the dialog box) and it is up to the user to change the DIP Switch position to work on the 10V range.

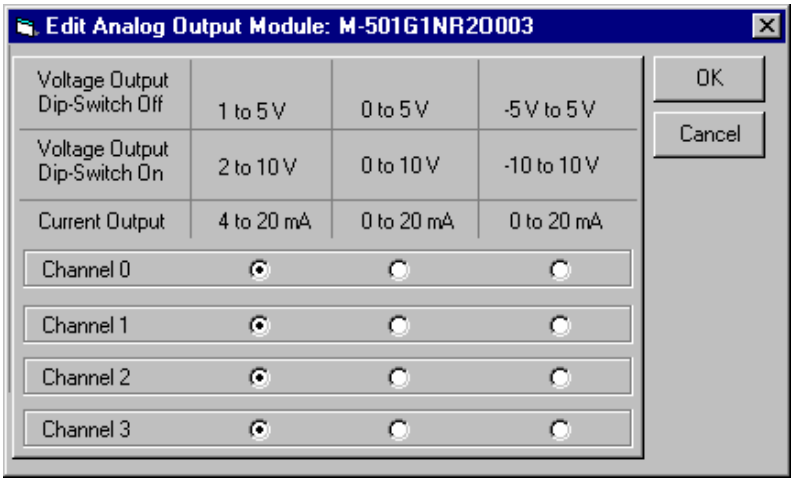


Figure 3.12- Setting the M-501 Inputs On CONF700

Configuring the FB-700 Module

Since the beginning, Smar has been the leader in Foundation Fieldbus technology and the LC700 was the first logic controller type of device to support a module that can be completely integrated in a Foundation Fieldbus network.

The FB-700 module is a standard Fieldbus Foundation device that can be integrated into a control strategy throughout a relationship with other function blocks located inside other devices in the network.

Smar has developed a couple block on the FB-700 allowing data transactions between the Logic Ladder network and FF devices. These function blocks act as links between two worlds. For discrete signal interface we use the CIDD or CODD while for analog signal interface we use the CIAD or COAD blocks.

Block Diagram	Description	Quantity	Memory
	ALARM - The input signal will be examined and an appropriate output will be set in order to reflect its condition.	8	0
	Comm. Input Digital Data - 8 Digital signals from the input will be available for the PLC.	4	16 Bits in the Digital I/O area. (8 Bits for data + 8 Bits for status)
	Comm. Input Analog Data - 8 Analog Signals from the input will be available for the PLC.	2	Take the place of 16 Analog Signals**+** 8 Bits in the I/O digital area (status)
	Comm. Output Digital Data - 8 Digital signals from the PLC can be sent to the network.	3	16 Bits in the Digital I/O area (8 Bits for data + 8 Bits for status)
	Comm. Output Analog Data - 8 Analog signals from the PLC can be sent to the network.	2	Take the place of 16 Analog Signals+ 8 Bits in the I/O digital area (status)

For the FB-700 module makes sure to enter the same physical device tag, (e.g. MIO-123 etc.) for the module as you have entered in the SYSCON software.

SYSCON is a dedicated tool to configure, download, troubleshoot and monitor a Foundation Fieldbus network. It is where the FF control strategy is set connecting function blocks from different devices. From the SYSCON each FB-700 is seen as a regular Fieldbus Foundation device and consequently the user has to associate the unique “Device Tag” to it. The SYSCON is also where the user defines the number of each I/O function block (MDO, MAO, MDI, MAI) necessary to interact with the FB-700.

The table below shows the direct relationship between function blocks seen in the SYSCON and how they are represented on the CONF700.

CONF700	SYSCON	Remark
CIDD	MDO	Fieldbus to Ladder Function Block, Discrete Type
CIAD	MAO	Fieldbus to Ladder Function Block, Analogue Type
CODD	MDI	Ladder to Fieldbus Function Block, Discrete Type
COAD	MAI	Ladder to Fieldbus Function Block, Analogue Type

The following dialog box, in the CONF700, must be filled in to be compatible with the SYSCON.

First the “Device Tag” in both SYSCON and CONF700 must match/ must be the same and this must also happen with the amount of each type of I/O function blocks (CIDD, CIAD, CODD, COAD).

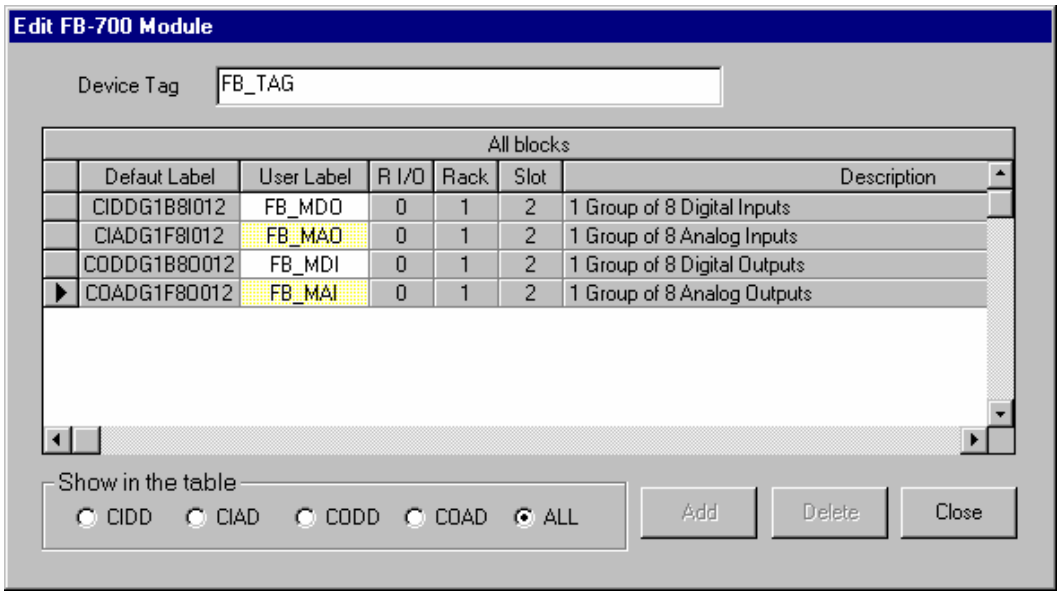


Figure 3.13- Setting Fieldbus Variables on CONF700

Any configuration mismatch between CONF700 and SYSCON will cause the "SAVING" LED to blink, it is located in front of the FB700 panel.

A Typical Foundation Fieldbus Application

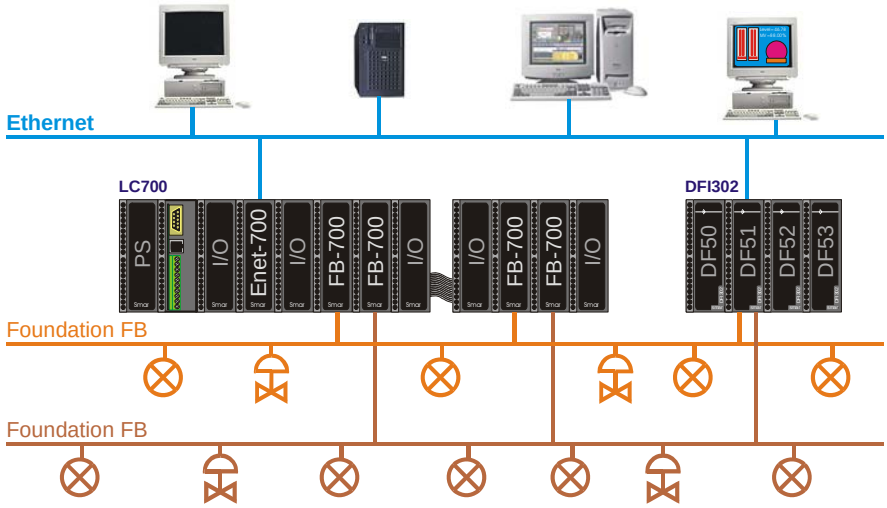


Figure 3.14- A Typical Fieldbus Foundation Architecture Using LC700 And FB700

In this example the LC700 and the DFI share the Ethernet line (hub/switch are not shown). A Foundation Fieldbus configuration can be sent to any of the Fieldbus networks from the PC. A configuration from the PC will pass through the DFI and reach all field devices including the FB700.

How to calculate the memory used for each Fieldbus Channel
Consider the Fieldbus module blocks connected and continue adding the necessary space per block. This information can be obtained from the last column of the Fieldbus blocks table.

The Balance Sheet

At the bottom of the “Hardware Page” there is a button with “Balance” written on it.

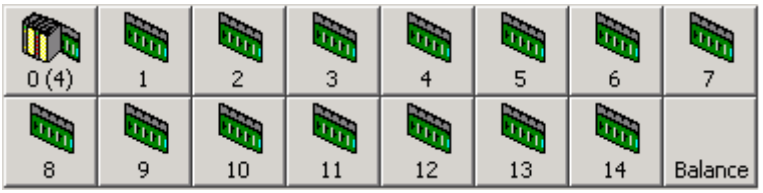


Figure 3.15- The Balance Function

Click this button, the CONF700 will present a complete balance sheet with a list of modules, racks, flat cables, terminals and an estimated current consumption. See example below.

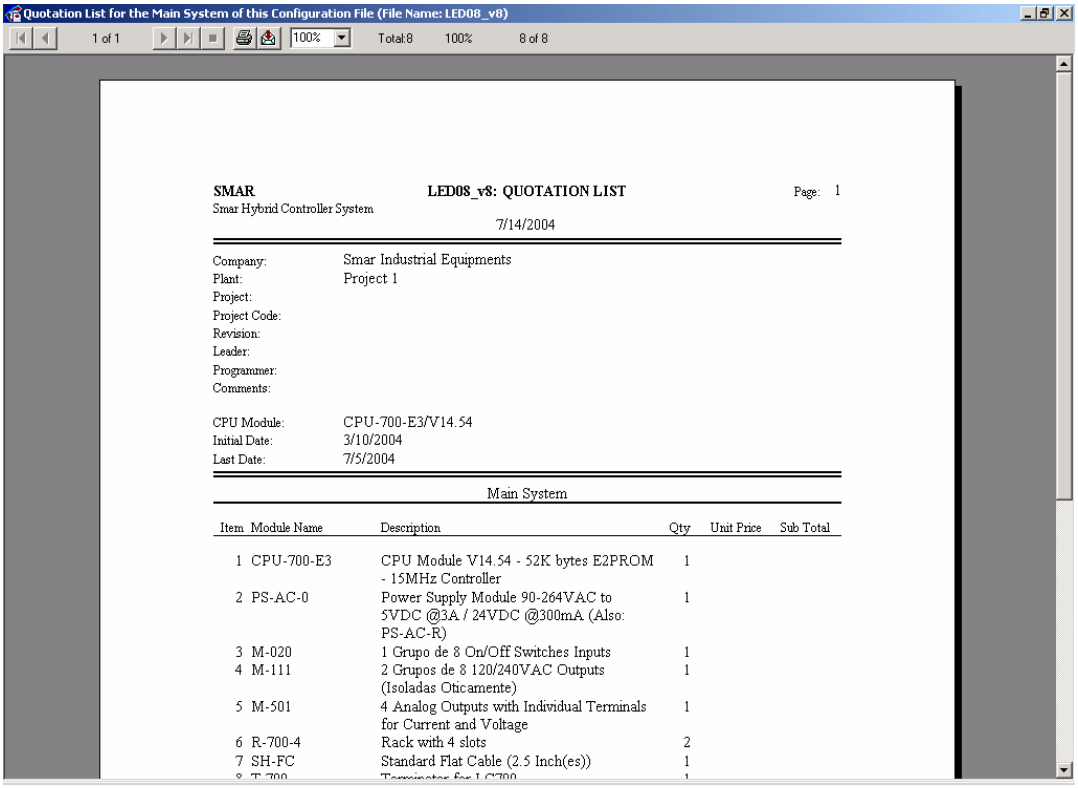


Figure 3.16- A Report Generated With the Balance Sheet Feature

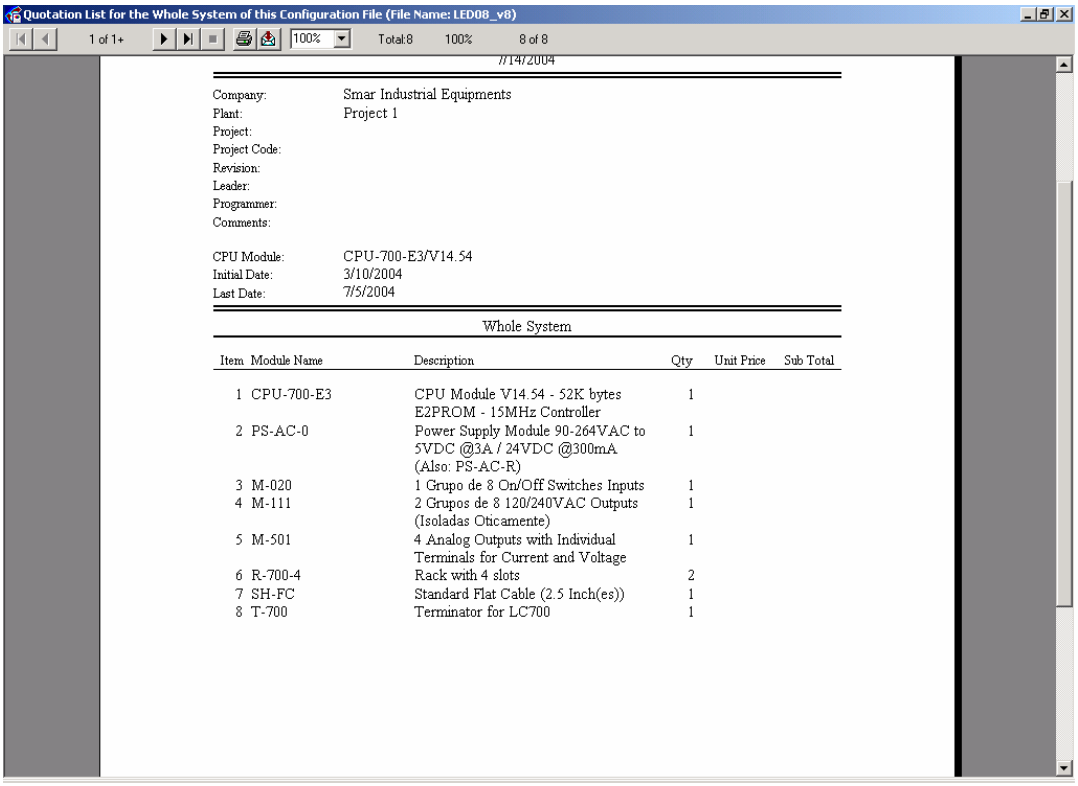


Figure 3.17- A Report Generated With the Balance Sheet Feature

This information can be printed or exported in many different formats for customization.

To export click on  and select the format and destination file.

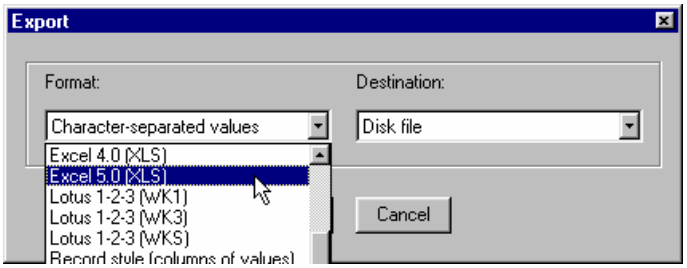


Figure 3.18- Exporting Modbus Variables

ID and the Modules

This extra circuit allows the CPU to identify modules, through CONF700, that have not been configured in the Hardware Page of the CONF700. During control the CPU checks for the existence of these modules.

Through the status block the user must keep updating and running the modules defined on the hardware page. To check the user must add the racks –on the hardware page of the CONF700, select I/O Module ID and click on the button at the lower left of the screen Check Module ID. In this way, CONF700 informs to the user which modules are plugged and their respective racks".

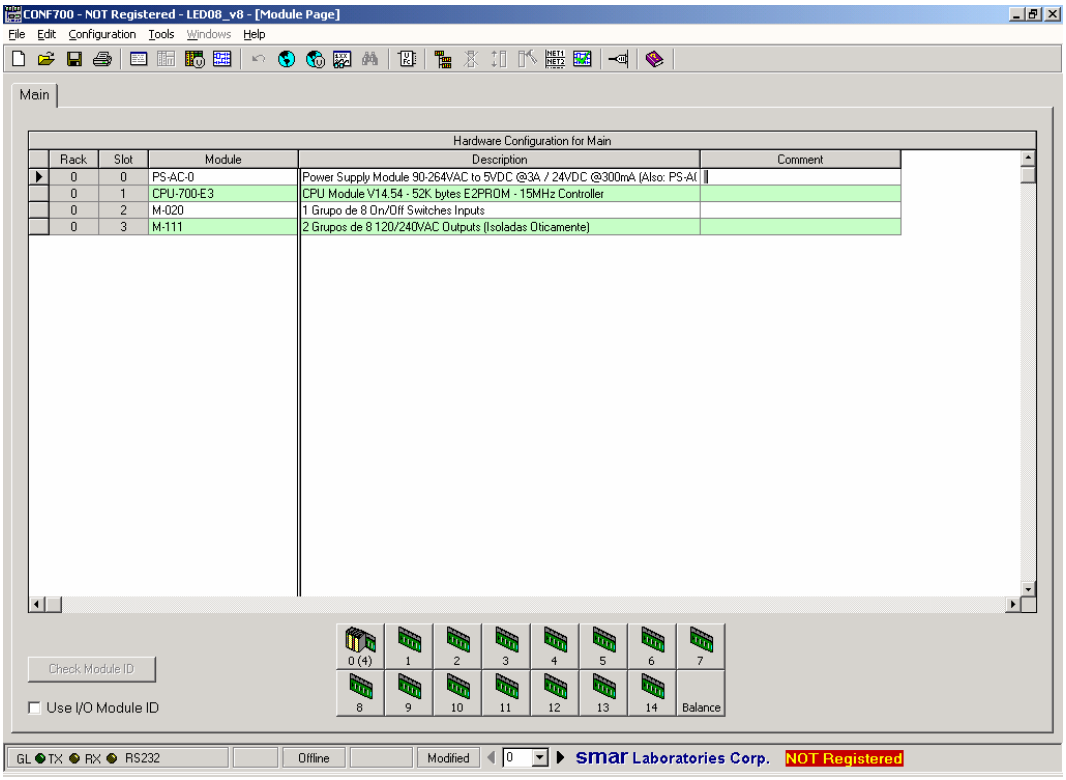


Figure 3.19- The Module ID Feature: The Check Box Enables This Feature

NOTE
The CPU-700 can read all I/O modules that do not support Module ID, ESD and Hot Swap since the option Use Module I/O with ID in the CONF700 is disabled. Thus, in systems having modules that do not have these features or systems that combine modules with these features it is necessary to disable flag in the CONF700.

A note about the cut, paste, and move tools

Cut and paste

These standard editing tools are available in the CONF700. They help the user to edit the control strategy, ladder diagrams and other functions.

Within a control strategy that requires several loops the user can use these tools to save time.

In the diagram below there's a ladder configuration. If it is required to repeat this block in another part of this logic network, these tools can be used.

Left click the mouse on the target area within the ladder logic and select the whole area. CONF700 generates a rectangle surrounding the target area.

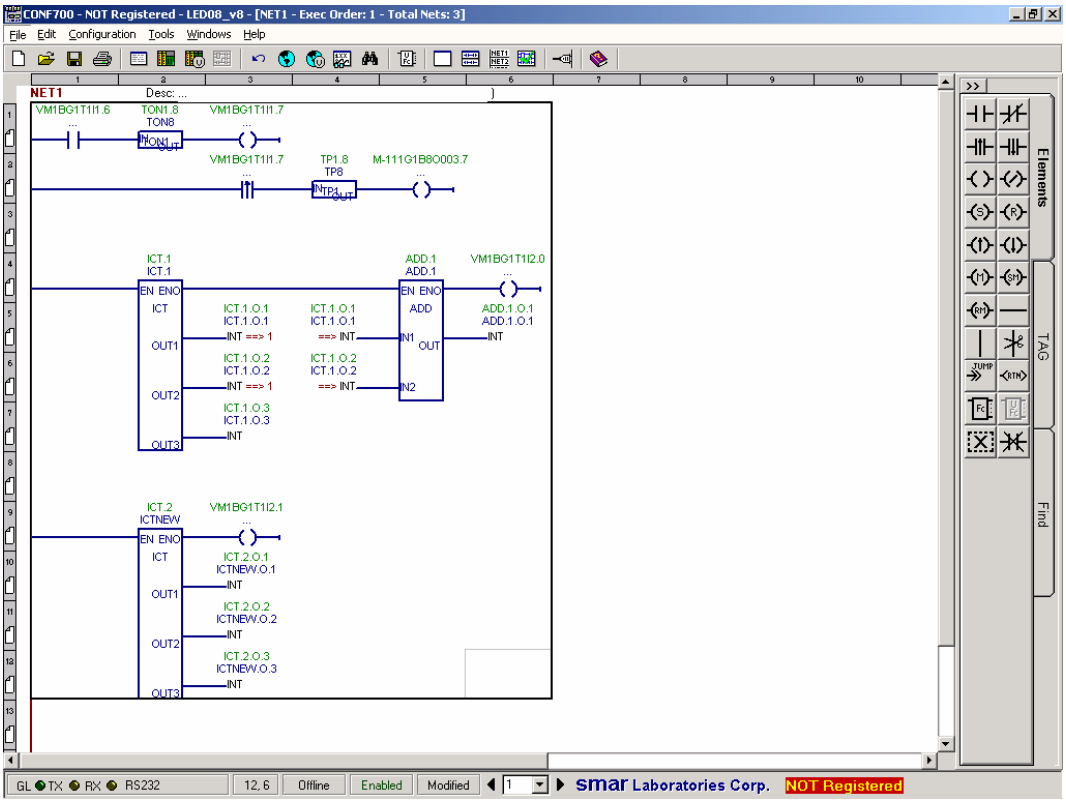


Figure 3.20- A Ladder Configuration Being Cut&Paste

Through the menu Edit→Cut we cut this selected region. After, it is necessary to go to the desired region within the ladder control strategy and using the Paste (Edit→Paste) tool. This might also be done through the CTRL+X and CTRL+V shortcut keys.

However during the Cut and Paste process links and labels are lost. CONF700 copies only the drawing. The user should add the new links (CONF700 does not duplicate links and labels).

Move

If the user tries to insert a function block where there is no space, the CONF700 will automatically enable the Move tool so the user must select another region to insert the element to be moved.
It is also possible to use the Move tool (menu Edit→Move) to move cells inside a ladder logic network. In this case CONF700 will keep links and labels because they have not been duplicated.

Undo

Every operation in the network can be undone through the undo button.

Figure 3.21- The Undo Button/Feature

In CONF700 it is possible to undo the last 20 actions. This is done through the Undo feature. The user can set the number of Undo actions in the menu Tools->Preferences. The number of operations that can be undone range from 0 to 20. The default value is 20 operations.

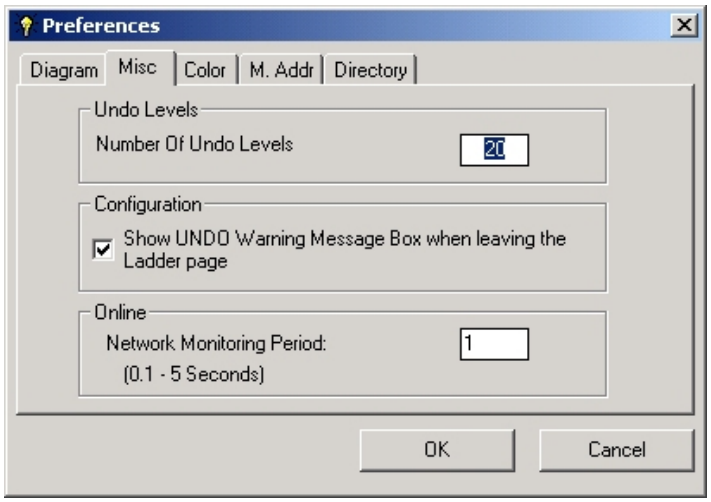
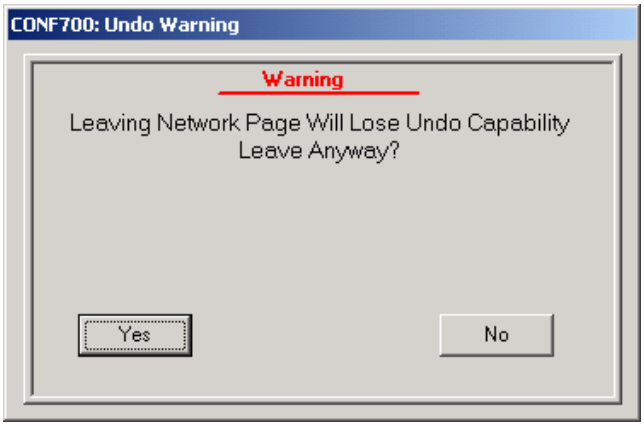


Figure 3.22- The Preferences Window

The Undo feature only works in the Network page. The user can undo operations made in this window according to the number of operations set. So, the user can revert operations like the following ones:

- Inserting an element (coils, relays, function blocks, etc.)
- Deleting elements
- Undo a move operation
- Undo a copy and paste operation
- Undo a replace operation

If the user leaves the network page all data available for undo operations will be lost. The following message will appear to the user:



Memory Allocation

CONF700 automatically generates a default tag for every I/O point and automatically allocates each module in memory.

The tag default label is based on the module type, local or remote, rack number and slot position. It means the default tag labels are based on the physical location of the I/O point.

The user can also set tags for every point according to his criteria. This is done through the global tables.

The default tag is constructed as indicated by the key below:

mmmm**c**g**d**ntxrs.c

Code	Function	Some Typical Values
mmmm	Module mnemonic	001, 101, 303, 401, CIDD, CIAD, …
c	Class	G (for regular I/O), S (for status), ...
g	Class number	1, 2, …
d	Data type	B (for Bit), I (for Integer), R (for Real), …
n	Number of points per group	1, 4, 8, …
t	Type of signal	I (for Input), O (for Output)
x	Local or remote location	0 (for Local), 1 to 6 (for Remote)
r	Rack identification number	0 to 14
s	Slot identification number	0 to 3
c	Channel number	0 to 7

Adding Modules

To add new modules go to an empty cell in the column Modules and click on it. A drop down arrow will appear at the left of the cell. Now click the drop down arrow and select the module clicking on it.

As soon as the module is selected it will be added to the corresponding empty slot and the CONF700 will automatically allocate memory for the I/O points. The user will no longer need to manage memory allocation.

Adding a new Rack

Click on an empty rack button icon to extend the backplane with more free slots. In the figure below only Rack 0 and 3 are used, the user can click on any other available rack.

Figure 3.23- Adding a New Rack On CONF700

When the next dialog box opens the user can indicate if the left slot (slot 0) needs a power supply, then a flat cable is used to connect this rack to the next rack. If this is the last rack of your back plane, select the Terminator instead.

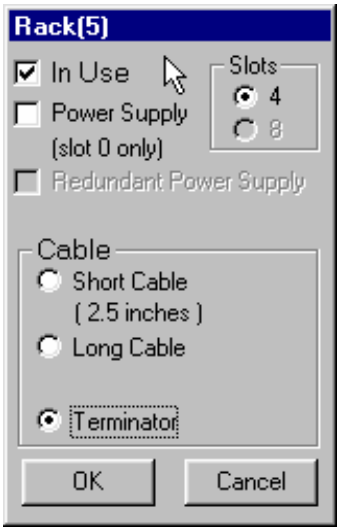


Figure 3.24- Enabling A Rack

Remote-I/O subsystem

Remote I/O allows distributing racks with I/O modules at various locations in the field and connecting them to the CPU via a high-speed industrial RS485 Remote I/O network. Distributing the Racks will save in wiring and installation costs and remote signals can still be part of the same configuration.

If the CPU model used is capable of working as a remote-I/O master, you can include remote-I/O interfaces to expand I/O modules on remotely located racks. An LC700 system can use up to 6 remote I/O subsystems.

To add remote I/O subsystems go to menu: Tools/Add-Delete Remote Interface or click on  on the tool bar.

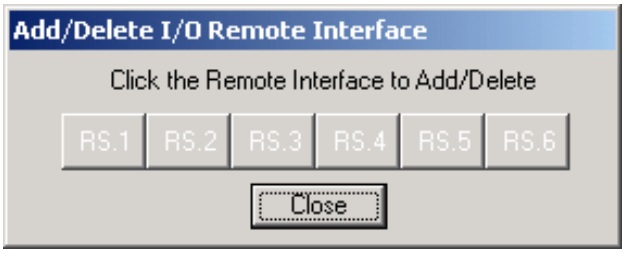


Figure 3.25- Enable Remote I/O Window

In the figure above select the subsystem by clicking on the remote I/O number(s) desired. Within the remote-I/O subsystem racks and modules can be configured exactly as the local I/O described for the Main system. New tabs, which are related to the remote I/O subsystems, will appear on the "Hardware Page"

RIO Limits

To follow has a utilization descriptive of remote and redundant CPU's in LC700. For each master CPU is possible to have up to 6 RIO's slave, and to each RIO had:

120 Words (240 bytes) of analog inputs, for example:

120 inputs for PT100 (15 X M-402) or;
120 inputs 4 to 20 mA (15 X M-401R) or;
60 inputs through FB700.

120 Words (240 bytes) of analog outputs, for example:

120 outputs 4 to 20 mA (30 X M-501) or;
60 outputs through FB700.

The digital inputs and outputs are limited for the total number in the CPU.

For the updating of redundant CPU's, where it is necessary to use remotes, prevail the same rule. Whenever the necessity goes bigger than these limits, the user will need to divide in more remotes.

Global Table

In the global I/O table you can configure a user-friendly symbolic identifier (user tag) for each I/O group or individual point (channel).

To start the Global Table go to the menu: Configuration/Global Table or click on  .

Channel user tags in particular, but also the description and the wire color (or alternatively the wire/terminal numbers) are extremely helpful when you configure the control strategy and they simplify your work. The more work done here, the less it will be done later in other parts. Most of all, it makes it easier for somebody else to understand the configuration in the future.

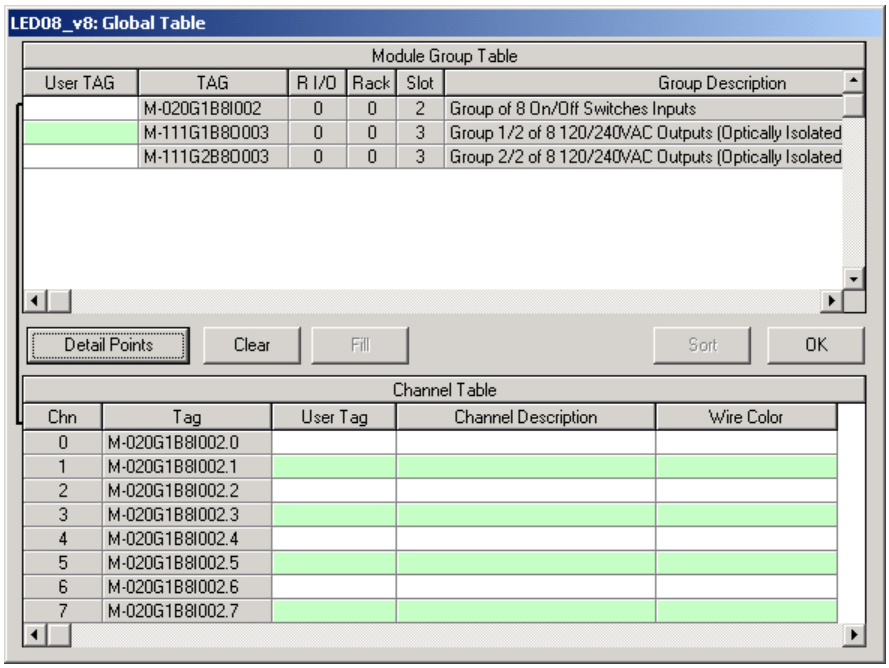


Figure 3.26- The CONF700 Global Table Window

The module group user label is also very helpful when locating the tags for the elements. It is therefore recommended that you enter these as well. It is recommended that the names are consistent and include loop tag and parameter names. For example, for the analogue I/O you can give a name in the format loop_tag.PV.

Fail/Safe Outputs

In the global I/O table you can also configure fail-safe value, i.e. the output in case of failure or at the downloading of a configuration to the LC700.

The user can type the fail/safe values in the grid displayed (see picture below) at the bottom of the window. The user can resize the fields (User Tag, Safe Value, etc.) if the text or values do not fit the field.

NOTE
Only Digital and Analog Output Modules support this feature. The Safe Values for the Digital Output Module are expressed in “0” and “1”, while the Safe Values for the Analog Output Module are displayed in percentage.

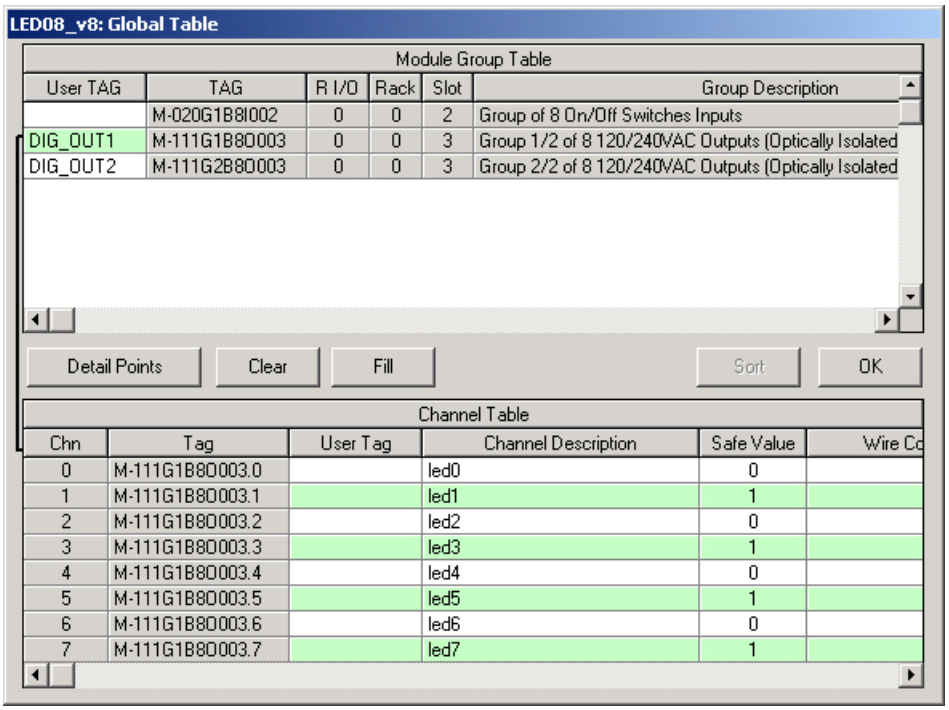


Figure 3.27- The Fail/Safe Outputs

Configuring Virtual Modules (Discrete Memory Locations)

Often your logic strategy requires storage of temporary variables that you may use later in one or more places in the ladder logic set of networks.

You may also need variables that have no physical I/O, but need access via Modbus for display and operation from the workstation. Virtual modules are used to create such points. An example of this is the auto/manual mode control of a PID block.

Use menu: Configuration/Virtual Module Page or click on  **to insert and configure as many virtual modules as required.**

Each virtual module gives 8 discrete memory locations that is the same of saying that extra 8 auxiliary Boolean variables were created. Add/create as many virtual modules as required but remember that 2000 points is the limit for the total number of discrete supported by one CPU-700-E3.

It is recommended that you separate different group of Virtual Variables for different parts of the plant strategy. This simple rule can make for a late track of the strategy and logic debugging.

Another set of virtual modules may also be reserved for miscellaneous application as unused ENO outputs of function blocks and a false constant for unused inputs of the function blocks is also useful.

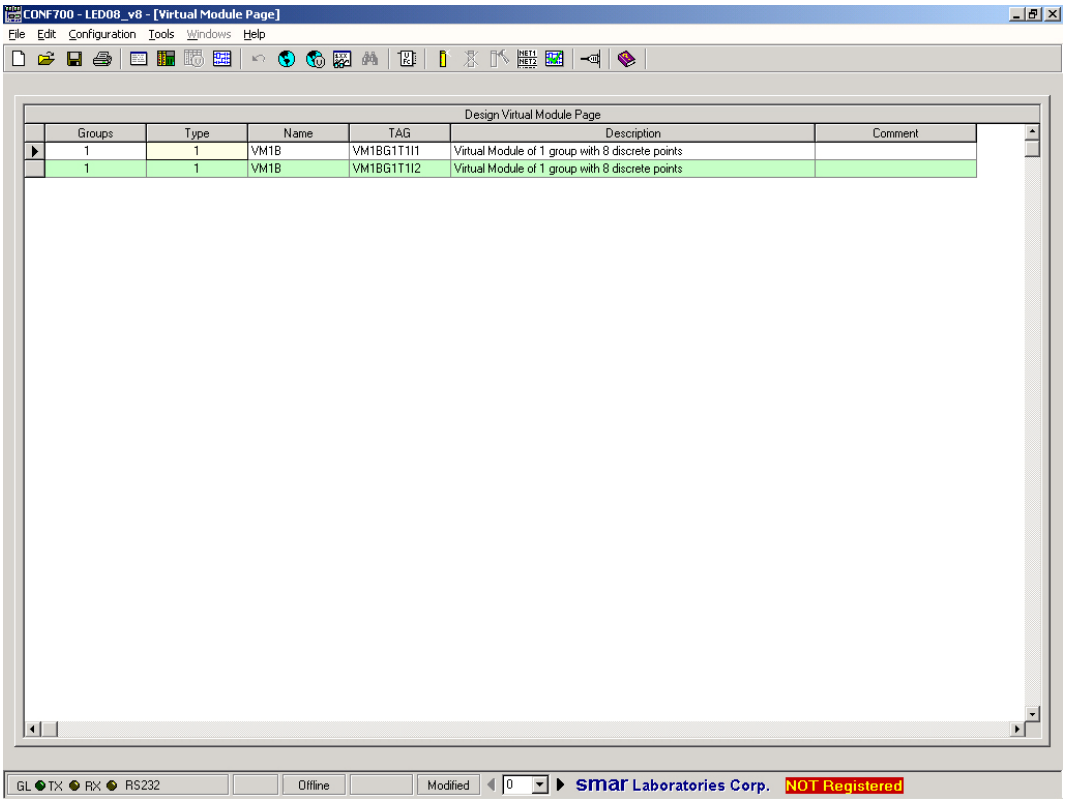


Figure 3.28- Configuring the Virtual Variables

To make it easier to find a specific memory point, it is recommended to configure “user label” for every point in a virtual module.

User Tags and Description for Virtual Points

User tags may also be configured for the virtual points in the memory.

Go to menu: Configuration/Virtual Global Table or click on  .

It is recommended that naming is consistent, e.g. using loop tag and a parameter name. E.g. loop_tag*.MODE for all points used to control the auto/manual mode of PID blocks.

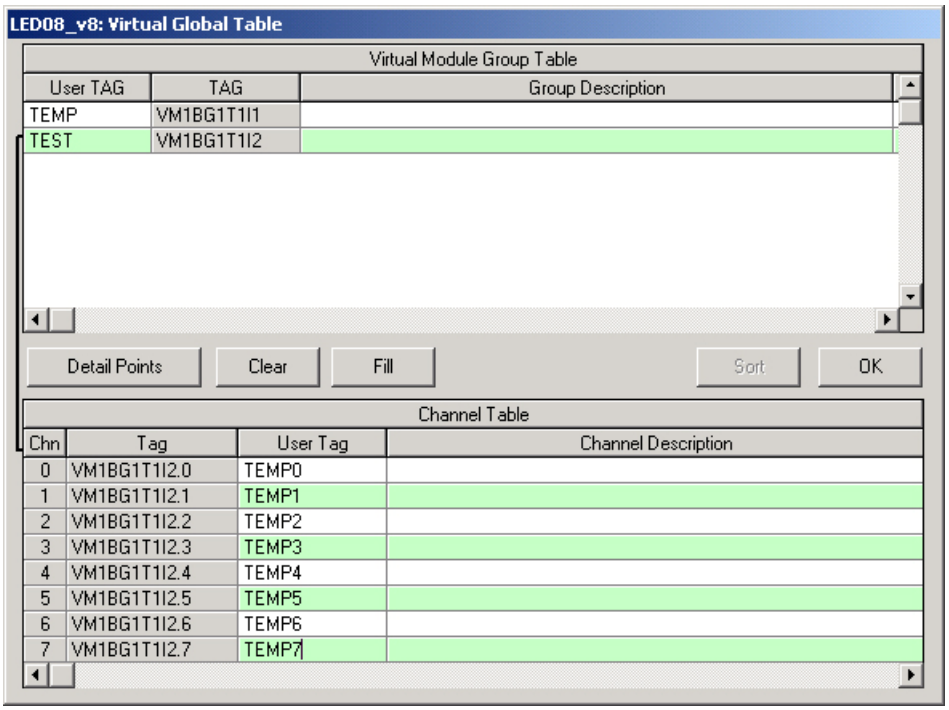


Figure 3.29- Setting User Tags

To make it easier to find a specific group of memory points, it is recommended that the user label be configured for the virtual module group.

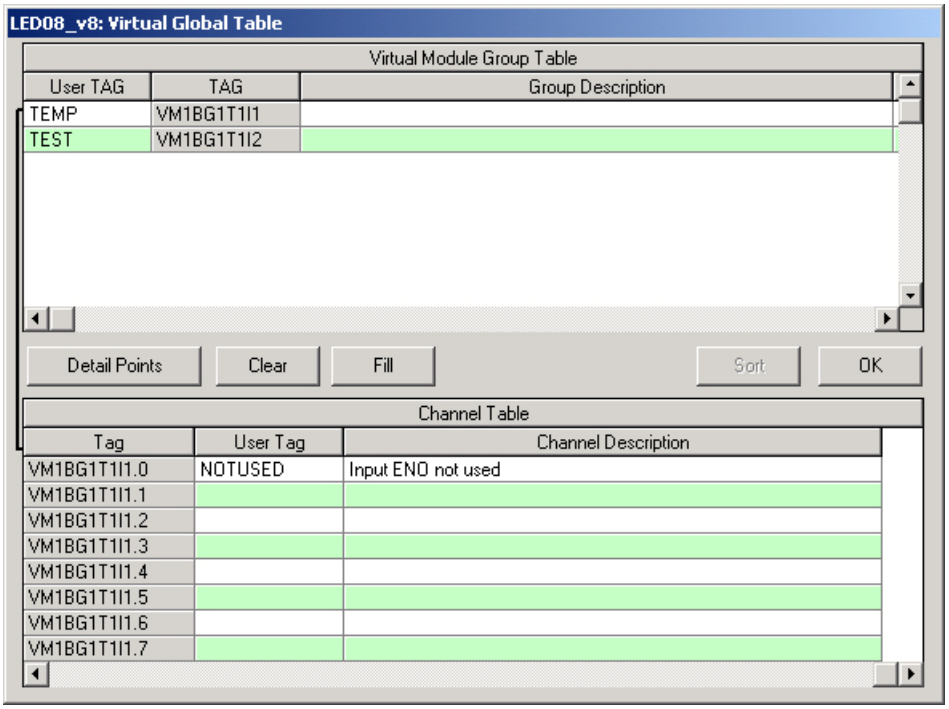


Figure 3.30- Setting User Tags

Only a single UNUSED point is required for all unused ENO points in the CPU. It can serve all function blocks necessary. It is used for the ENO output of all function blocks not used to link to other function blocks. This will ensure that there are no given disturbing error messages when the configuration is validated. There will be in most configurations unused inputs that have to be associated with some point in order to avoid error messages. For this purpose it is also a good idea to create a point as an unused input, e.g. called FALSE.

Configuring the control strategy

After the hardware is defined and the control strategy is ready the user can begin to assemble the operating logic for the application connecting a set of Logic Ladder Networks to accommodate functionality of the plant application.

It is strongly recommended to follow the basic steps explained before in order to continue to the logic network preparation. User might go back at anytime to make changes for optimization or to expand the application itself. This implies in number and type of modules as well as the logic of the control strategy that can be edited.

To enter the control strategy (also called “Network Page”) go to menu: Configuration→Network Page or click on the icon  .

Ladder Diagrams (Ladder Networks)

Control strategy may be divided into several ladder diagrams (ladder networks). Don't confuse the ladder networks with the Modbus communication network.

The LC700 follows the IEC-61131-3 standard for the ladder language and it supports logic ladder elements. It also supports a powerful set of function blocks, from simple ones to more the sophisticated ones.

The LC700 logic networks can accomplish a wide range of applications. It is very easy to write these logic networks.

The LC700 is described as a Hybrid Universal Controller. Due to the vast number of function blocks that can be mixed in the same diagram with discrete type of elements to cover each application not only oriented for discrete control but also for the more complex applications in process control.

Ladder flow and blocks can be linked together. The user can use as many ladder networks as is necessary in the project if there is enough memory (the demo version of CONF700 is limited to only two networks).

For example, the user may want to have one network for each control loop in the system so it is easy to find all the blocks and logic associated with that one loop, i.e. similar to the ISA S88.01 "control module" concept. For sequence and batch control, one network can be configured for each step, and one main network to control the transition between the steps.

The Logic Network

Each network is a matrix of 15 rows by 16 columns totaling 240 available cells. These cells are used for entering logic ladder elements and function blocks. A “Power Rail” for ladder is at the left of the matrix and a power flow will always go from left to right, consequently “coils” tend to go to the right following a logic that includes “relays” and function blocks.

It is also possible to create a one-time-use Boolean function to be added later while editing in any logic network. This is basically a customized function block where the user determines the number of outputs and all of the Boolean equations to be internally solved.

The Complete Cycle of the LC700

Sometimes it is important to know exactly how the execution sequence used by the CPU of a LC700 system is to solve the Ladder Logic.

Everything begins when the CPU processor reads the inputs coming from all kinds of I/O modules (local or remote). In the next step the CPU verifies the execution order in the list of Logic Network included in the configuration and begins to execute one by one. When the last Logic Network is finished the CPU processor sends the results to the output modules (local and remotes). Following it also responds to any communication request pending.

Synchronized Ladder Logic Execution and Communication

The local and remote modules work in synchronization. It is important to note that the CPU can receive a communication request any time but it will only reply after all Logic Networks are completely solved. In this way you will never read an intermediate result during the execution of the Logic Networks.

Execution Sequence of a Logic Network

For many applications, the specific order for each individual cell in the logic could make the difference .The CPU processor begins with the cell in the first row and column, Cell (1,1) of the matrix. It continues to the end of the first column, before it moves to the first element of the second column and proceeds this way until all cells in a specific network are read. It will then proceed to the next logic network until the last one in the execution list.

Logic Network Editing Preferences

To make it easier the user can display the rulers (row and column grid numbering) by selecting Tools followed by Preferences. To start go to menu: Tools/Preferences

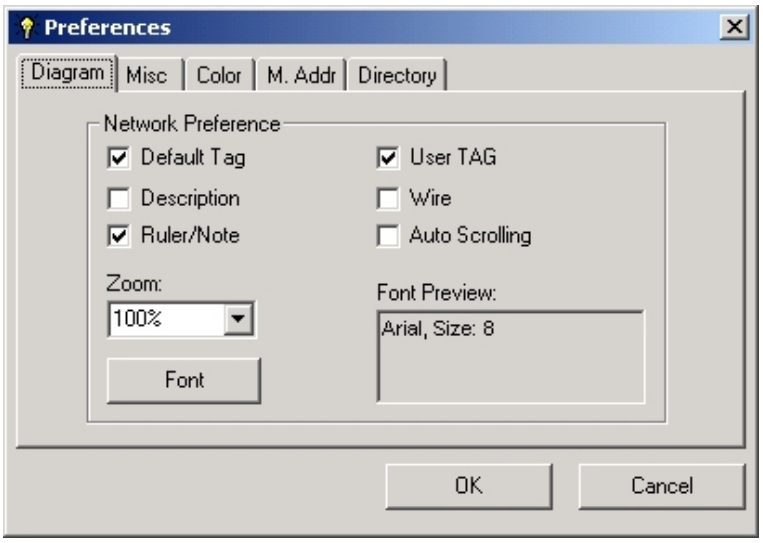


Figure 3.31- The CONF700 Preferences Window

Managing Multiple Logic Networks

To make quick reference easy, each diagram (ladder network) can be named and described. Additionally the execution order can be configured. It is also possible to disable the execution of one or more Logic Networks.

To access the Network Manager go to Menu: Edit→Network Management or click on 

.

A listing of all Logic Networks (ladder diagrams) will appear in a dialog box.

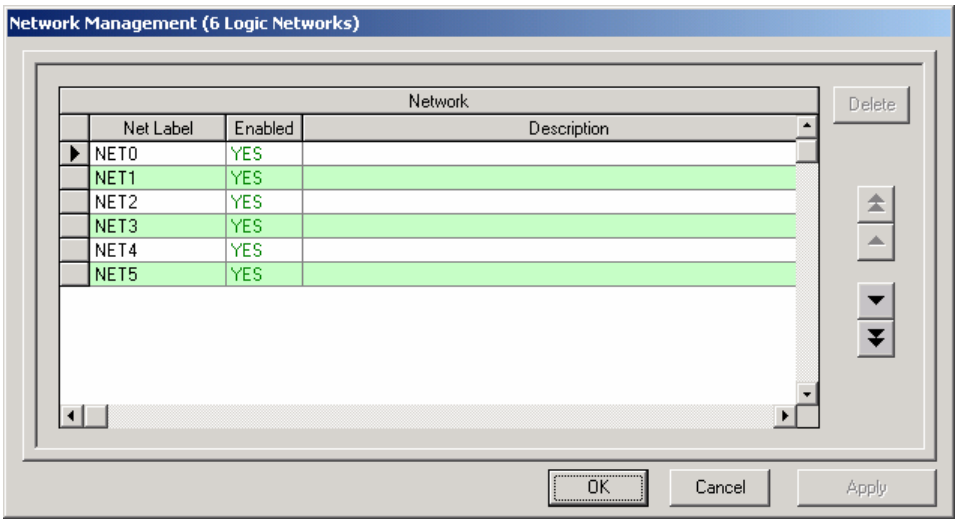


Figure 3.32- The Network Management Window

As in the list above, it is recommended that each ladder diagram network be named and given a description.

Moving from one Logic Network to the other

Select the ladder network that you want to work with by clicking on the back and forth arrows at the bottom of the screen, or select directly by name from the drop down list that appears when you click on drop down arrow in white box between the arrows.

Inserting Ladder Diagram Elements

There is a specific tool bar to insert/delete ladder elements. It is available on the right side of the CONF700 interface. The user can create and edit ladder network programs via the “Elements” tab.

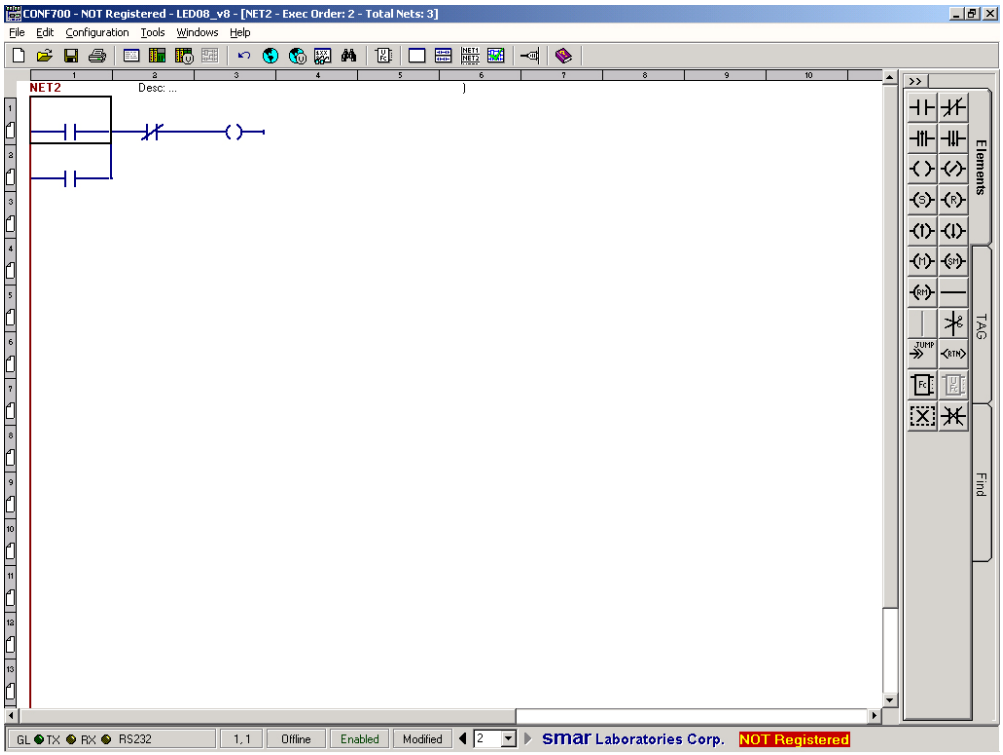


Figure 3.33- Inserting Contacts And Coils

Select contacts and coils from the “Elements” tab. Horizontal and vertical lines may also be selected.

To add a ladder element to the network, just select the element such as coil, relay, horizontal or vertical lines in the “Elements” tab on the right side. Click the element to add and place it to the desired area in the network. CONF700 will automatically insert this element.

CONF700 has a check-as-you-go feature that prevents the user from inserting elements that are not applicable to a particular cell. In this case an “unfit” message will appear. Many times restrictions can be overcome by inserting horizontal or vertical lines in the adjacent cells.

Once you place a logic element you can associate it to a point referred by its default or user tag.

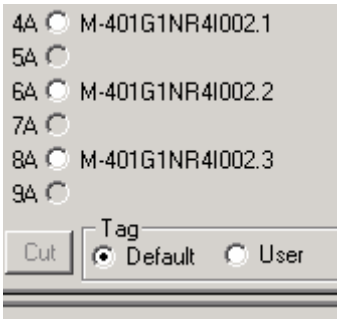


Figure 3.34- CONF700 Default Tags

After inserting an element, it is required to be associated with an user TAG.

This is done by using the TAG tab utility located on the right of the CONF700 interface. See the picture at the left side of this page.

In this tool bar, there is a drop down list where the user should choose the type of the element or link.

The type of the element or link may be:
Relays
Coils
Analog Input Link
Analog Output Link
Byte Input Link
Byte Output Link

If the user selects Relay then CON700 will show two options: I/O groups and Virtual Groups.

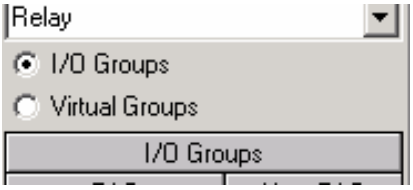


Figure 3.35- I/O Groups and Virtual Modules

If the user selects Coils then CONF700 will show two options: Output Groups and Virtual Groups.

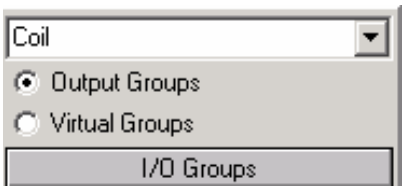


Figure 3.36- Output Groups And Virtual Groups

Relays and coils may be also associated with digital modules inputs or outputs.

Next, to give a TAG to the element, select the target element and CONF700 will automatically show the available virtual modules, I/O modules (input/outputs). This is all done in the TAG tab.

After selecting a virtual point, input or output by clicking on it, place it to the element which will be assigned with the TAG. All this TAG attribution is done through a simple click (drag) and drop operation.

Note: Do not forget to select the element type in the drop down list. If the selection is “coil” and the element that will be assigned with the tag is a relay, CONF700 will prompt an error message .

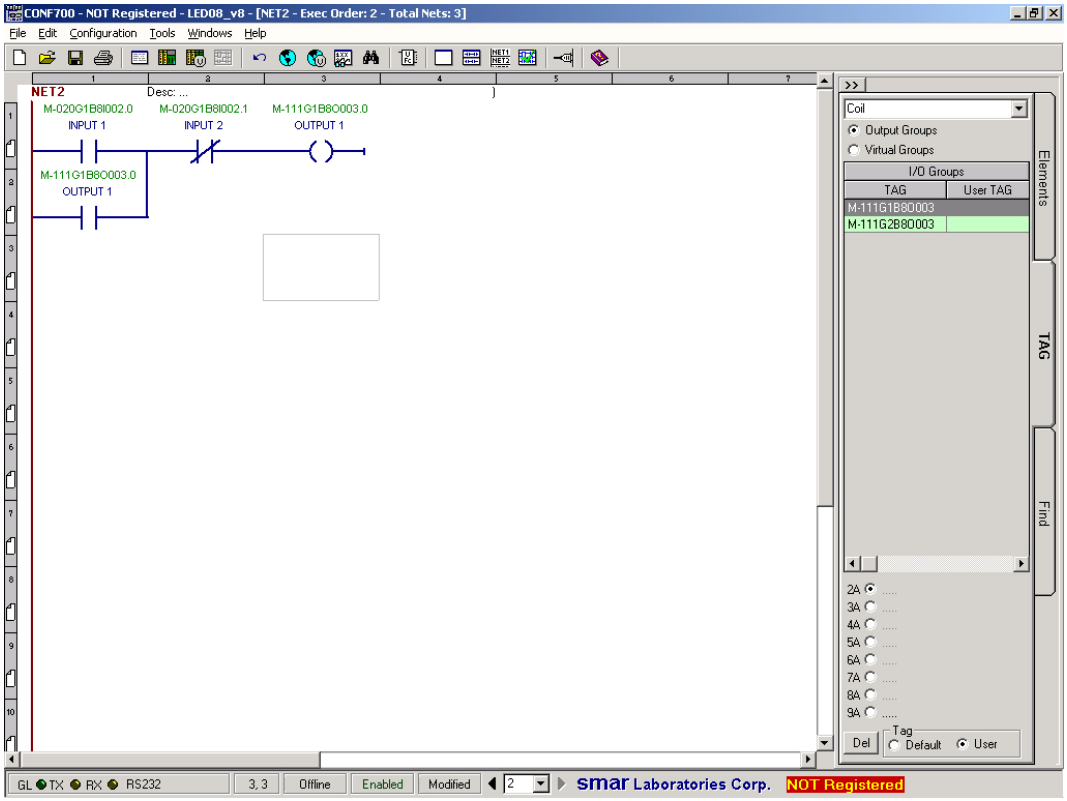


Figure 3.37- Contacts And Coils And Their Respective User Tags

Note that contacts (inputs) also can be associated to outputs (coils) creating interlocking logic often used to hold/latch a state. See the MOTOR1 point as both output coil (at R1C3) and input contact (at R2C1) in the above diagram indicating that they have been “linked”.

The user can also delete an assigned TAG. There is a ‘Del” command button at the bottom of the TAG tab:

Fig 3.38- The Del Button

Click at this button and place the mouse over any TAG necessary to be deleted.

Inserting Function Blocks

Click on the Elements tab and select “Fc” icon in the toolbar. Move the mouse on the Network. mouse icon will change to “Fc” alike. Click on any cell where the user wants to insert a function block. A “Function Block” form will pop out. Select the block type from the drop-down list. There are restrictions where the block can be inserted. This is related to its size and the elements on its neighborhood.

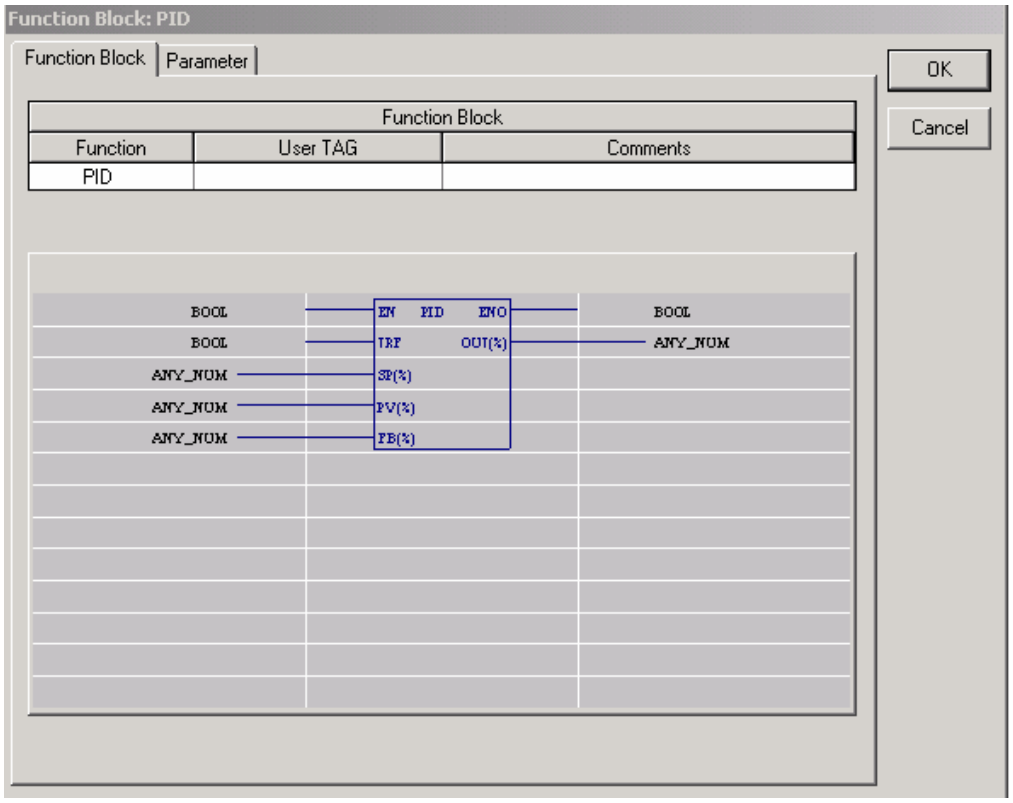


Figure 3.39- The PID Block: Selecting A Function Block

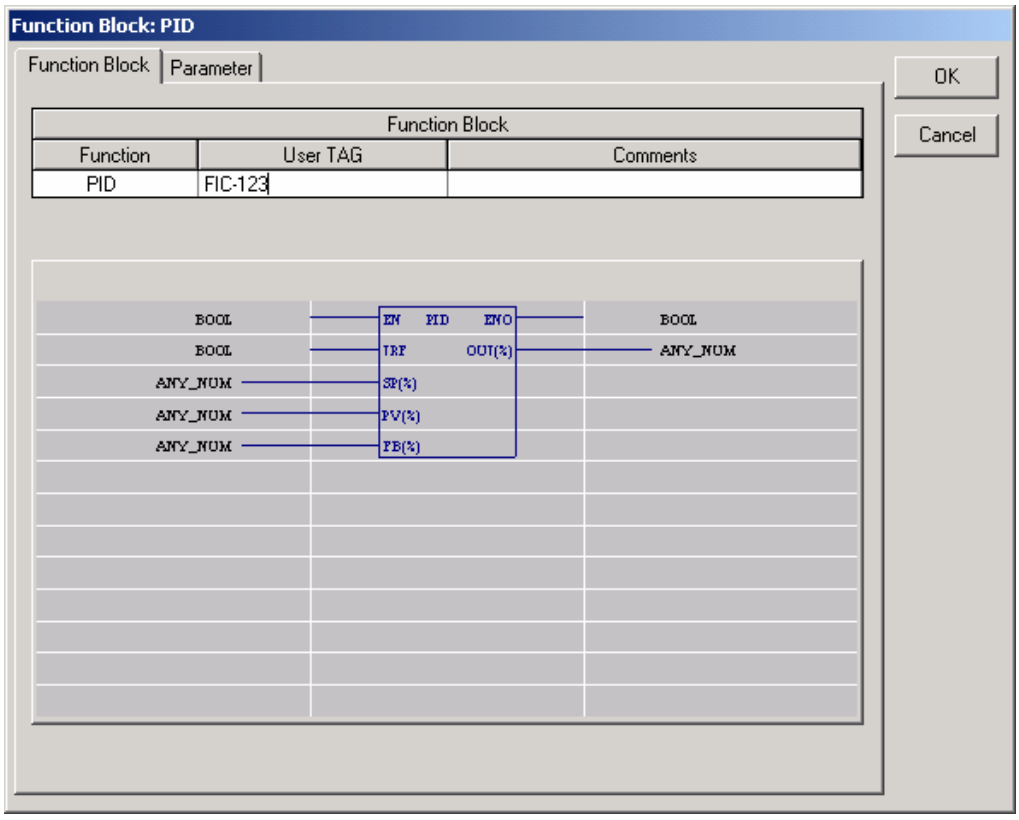


Figure 3.40-The PID Block: Creating An User Tag For The Function Block

A tag must be assigned to the function block in the "User Tag" column. CONF700 prevents tag duplication. The user may need to go to the Parameter tab to complete the function block settings.

Each parameter has a default value and specific value range.

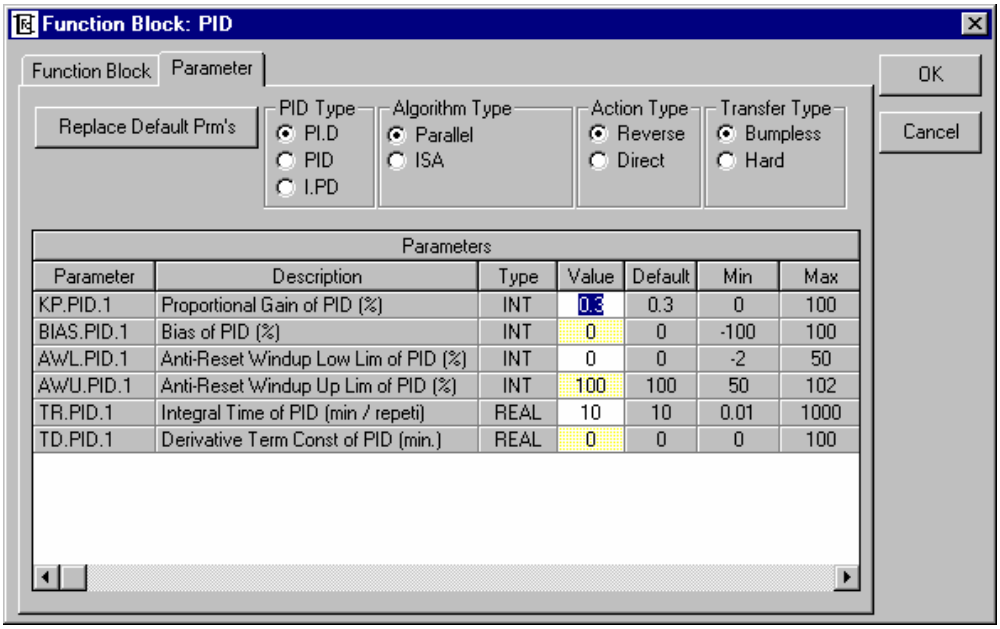


Figure 3.41- The PID Function Block- Setting Parameters

After the block has been inserted, parameters can be edited by pointing to the block and right clicking the mouse.

Note that at this time the data type for the inputs and outputs have not been determined, and is therefore indicated as "ANY_NUM". CONF700 automatically generates default tags for the inputs and outputs.

For function blocks the enabled input (EN) must be true for the block to execute. This may be achieved by connecting it to the left power rail. See the connection of the enabled input to the left power rail (at R1C1).

To avoid warning messages when checking the configuration, all unused ENO outputs should be connected. Therefore, a coil has been inserted (at R1C3). To associate the PID block output with the common unused point, click on the ENO output and add a coil element in this output.

In order to give TAGs to the PID block inputs and outputs, select the desired function block input or output. Check the TAG tab. If, for example, we click at a PID output the TAG tabs automatically shows all available links for this input:: other function blocks inputs or outputs, I/O modules inputs or outputs.

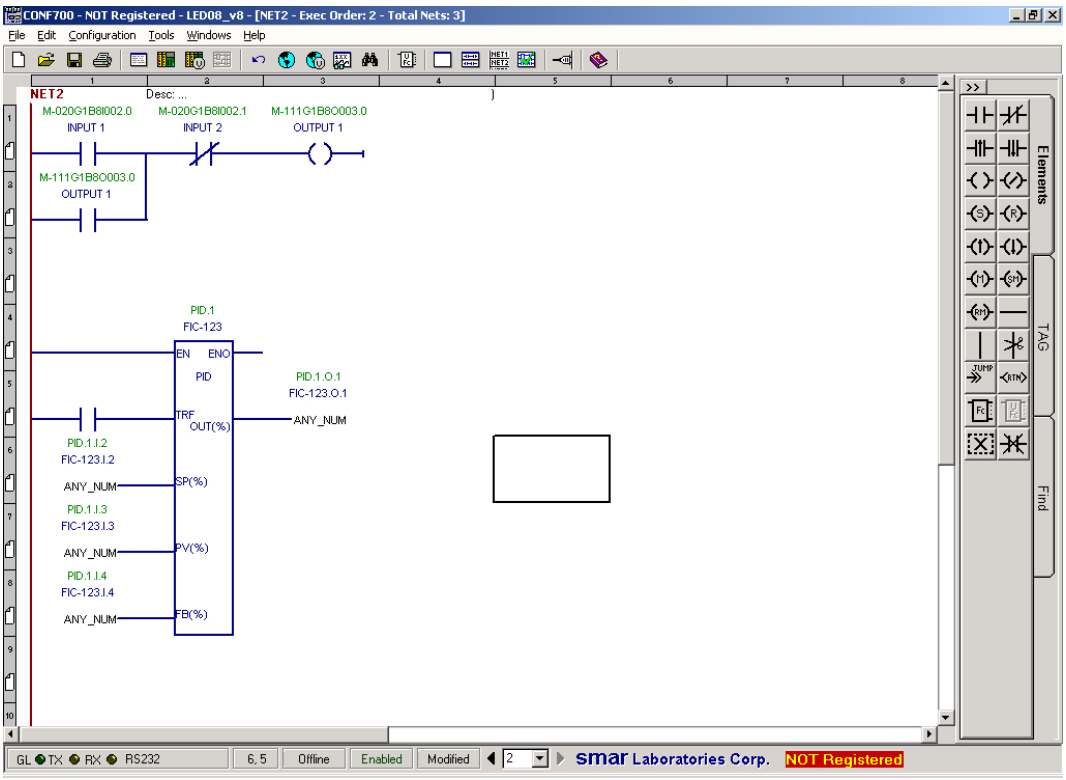


Figure 3.42- Setting Function Block Inputs And Outputs

Next, select a Function Block input or output link, an I/O Module Input or Input in the TAG tabs. To give this TAG to the desired FB input or output, simply click the tag and place it to the FB I/O. Or highlight the FB I/O, double click the tag that is going to be linked. CONF700 will automatically assign the tag to that FB I/O.

Double click on the PV input to configure which I/O module and channel, or other function block, it shall be associated to.

To configure PV to one of the physical current inputs, click on the PV input, make sure that “I/O Group” button is selected and pick the desired module and terminal. If the user selects to display user tag the tag configured for the point in the global I/O table will be indicated. If PV comes from another function block select instead the “FB output”. Once one of the inputs or outputs are selected the data type for all inputs and outputs of the block is set automatically. All inputs and outputs must be of the same data type.

Deleting elements with the button delete

Click on these buttons to activate the delete function. Next place the mouse in this element and click at the element to delete it.

To delete a region with ladder elements and click at the button below. Select a region and click at it to delete.

Function Blocks Links

There is a grid to display all the links connected to the output. When the user wants to cut the link, the user only needs to highlight the link (row) in the displaying grid, then click on the “Cut” command button to cut this link.

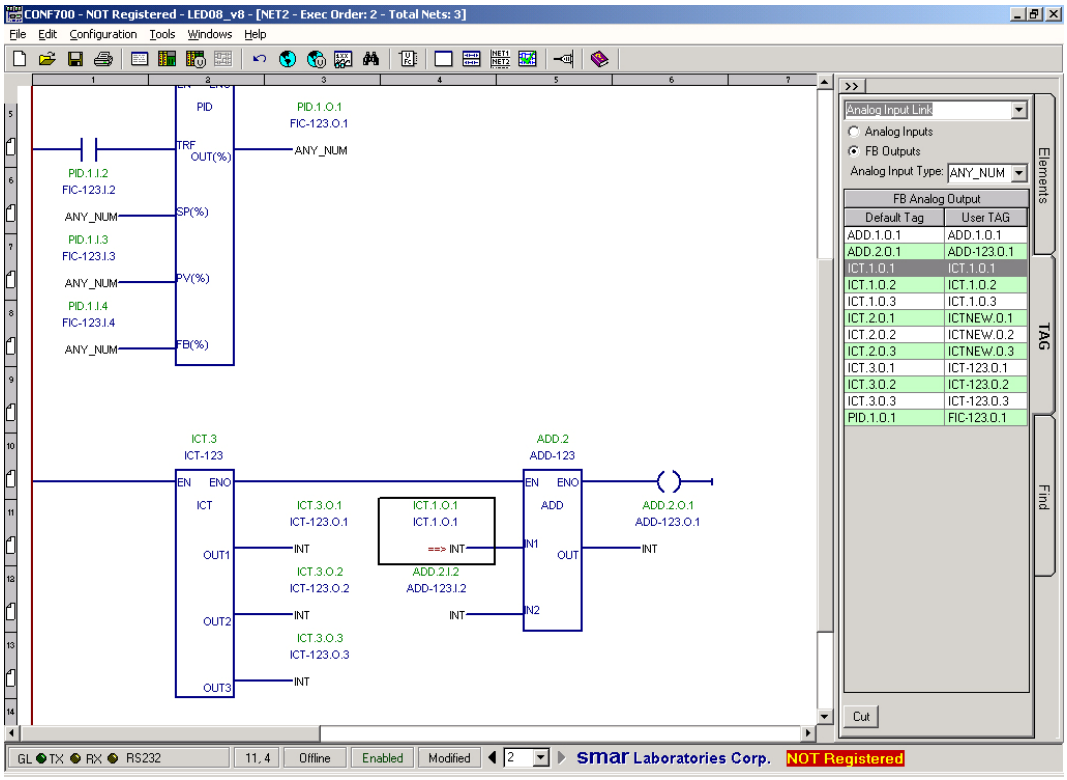


Figure 3.43- Function Block Links

PID loop automatic/manual operation

For the automatic/manual mode switch a contact has been configured (at R2C1 in the above example). For an automatic/manual mode switch to work, an associated virtual memory point is required.

Memory location can be created and given a suitable user tag as described above. Then the auto/manual contact has to be associated to this memory location.

Select the track feedback (TRF) input of the PID block in order to configure it. Then, in the TAG tabs, select the virtual memory point that will be associated with this input.

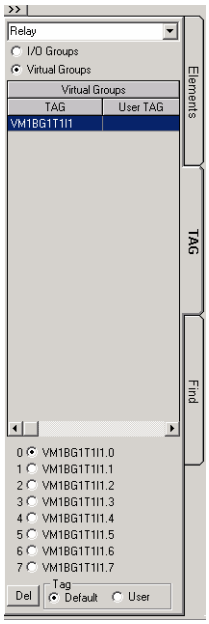


Figure 3.44- Automatic/Manual Operation/

Select “Virtual group” and choose the associated point. If the user tag for the point has been configured in the virtual global table, the respective tag will be indicated in the list.

The Mode of the PID block can now be set from the operator workstation by writing a value to the Modbus register that corresponds to the virtual point.

In the manual mode the output value cannot be written straight to the PID module output because the physical output is not updated. Typically a sample and hold (SMPL) block is connected to the PID output used as an auto/manual station and in manual mode the output is written to the SMPL block output instead.

Click to the point in the diagram where the block is to be inserted. Then enter the block by clicking on "Fc" in the toolbox and select the block type.

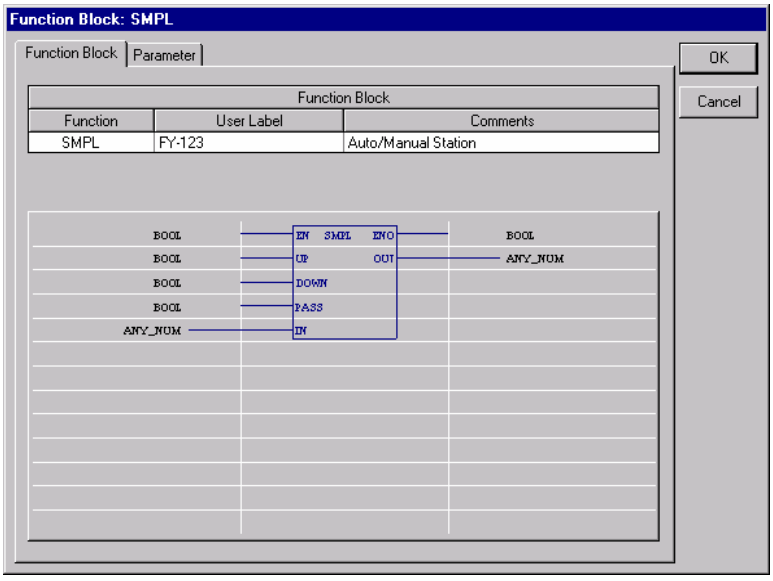


Figure 3.45- Adding A SMPL Function Block

Configure the ladder elements for EN input and ENO output. The Up and Down inputs should have Normally Open (NO) contacts, and the Pass input a Normally Closed (NC) inverting contact.

The Pass input shall be configured to the auto/manual point, i.e. the same point as the TRF input of the PID block. The SMPL block is inverted by the NC contact.

Follow the same procedure for the TRF input. The up and down inputs are likewise configured to the previously defined FALSE virtual memory point. Configure the SMPL input from the PID output by clicking on the SMPL input, select the "FB Output", and choosing the PID block output from the list.

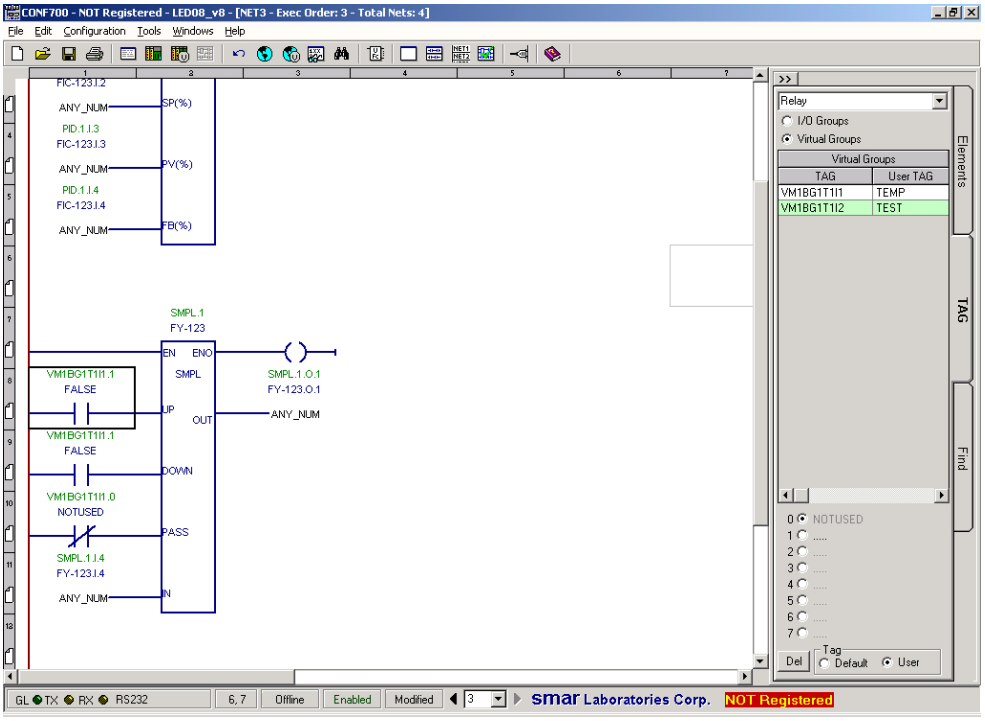


Figure 3.46- Linking the SMPL Block to the PID Block

Once the PID output has been linked to the SMPL block the user can see the tag of the PID output at the SMPL input confirming the link. Also note how the TRF input of the PID is the same as the Pass input of the SMPL, but that it is inverted.

For the PID block the feedback input (FB) also has to be configured in order to ensure bumpless transfer. The value must come from the output of the block that goes to the current output, i.e. in most cases the SMPL output shall be connected to the FB input of the PID.

Click on the FB input and select “FB input” in the TAGs tab and choose the corresponding SMPL output from the list.

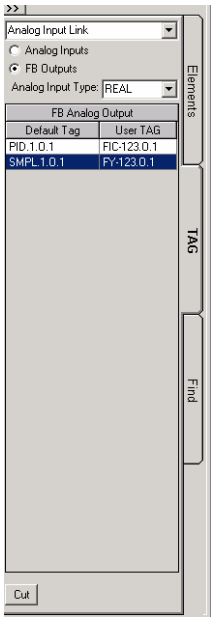


Figure 3.47- Choosing Inputs

In the special case that any arithmetic is done on the PID output in other blocks after the PID (e.g. feed-forward), the final output value must have a corresponding inverse function before it is connected to the PID feedback input.

To make the configuration work easier the user can change the user tag of the SMPL block to indicate the loop tag and that it is the output of the loop, e.g. loop_tag.OUT. The CONF700 ensures that there is no duplication of tags.

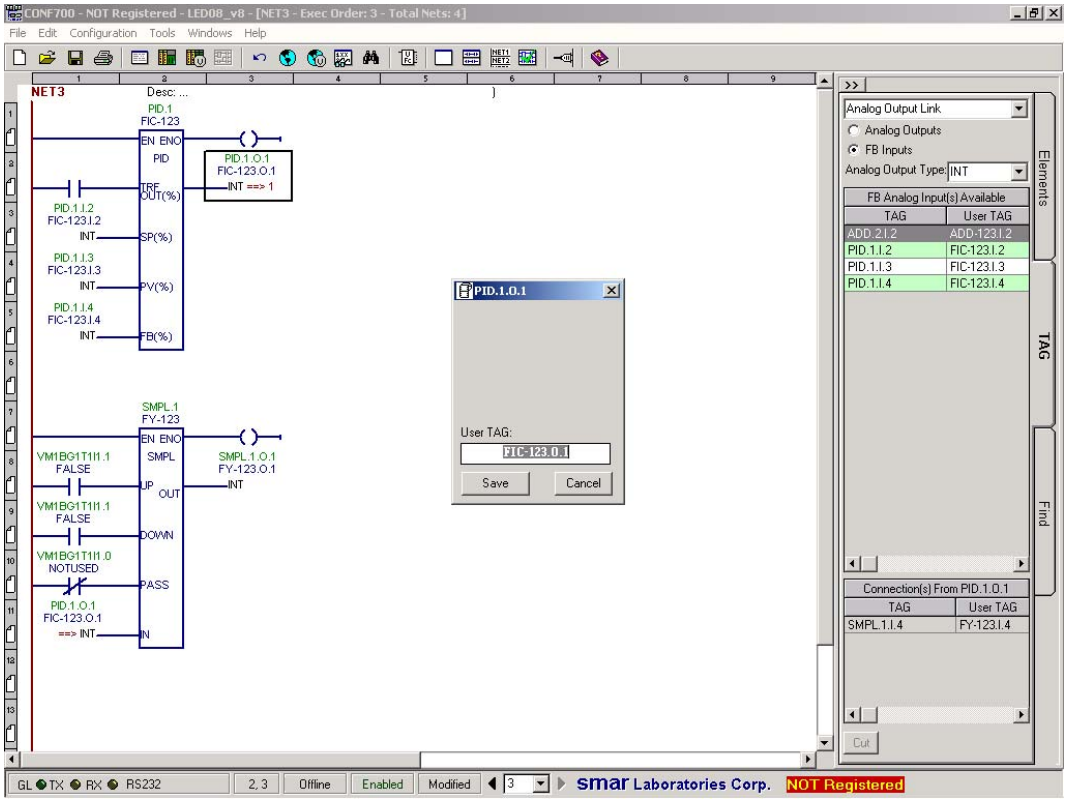


Figure 3.48- PID And SMPL Blocks Ladder Diagram

See the diagram above how the SMPL output goes to the PID feedback input. In the manual mode the output value can now be written to the Modbus register corresponding to the output of the SMPL block. Finally, the loop output must be associated with the physical I/O point. Click on the SMPL output (OUT) to configure it to an output module (“I/O Group”).

PID loop setpoint operation

The PID block needs to have a setpoint (SP) value input to operate.

Therefore an integer constant block (ICT) has to be configured. Again it is a good idea to have a consistent methodology for your ICT blocks. E.g. keep only one loop per ladder diagram (network) and use one ICT block for the setpoint and alarm limits of that loop. Or, use a separate network to contain all of the ICT blocks for the entire system, divided in a manner that one block handles setpoints, another alarm limits etc.

Select the desired cell and click on the “Fc” icon in order to insert this block. Select the ICT block and give it a tag. Make sure the EN input is true by connecting it to the left power rail using a horizontal link.

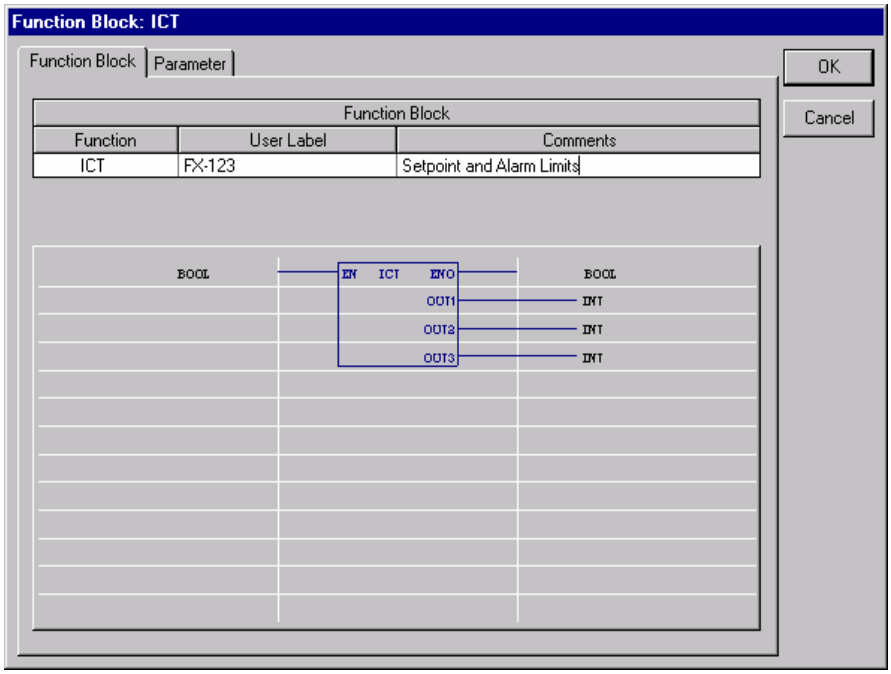


Figure 3.49- Inserting an ICT Function Block

Each ICT block output can be given a different user tag to make them easier to refer to, this is done by right clicking over the output and then typing in the desired name. Again, consistency makes it easier to configure and understand the control strategy. E.g., all of the setpoint tags may be constructed from the loop tag followed by loop_tag.SP.

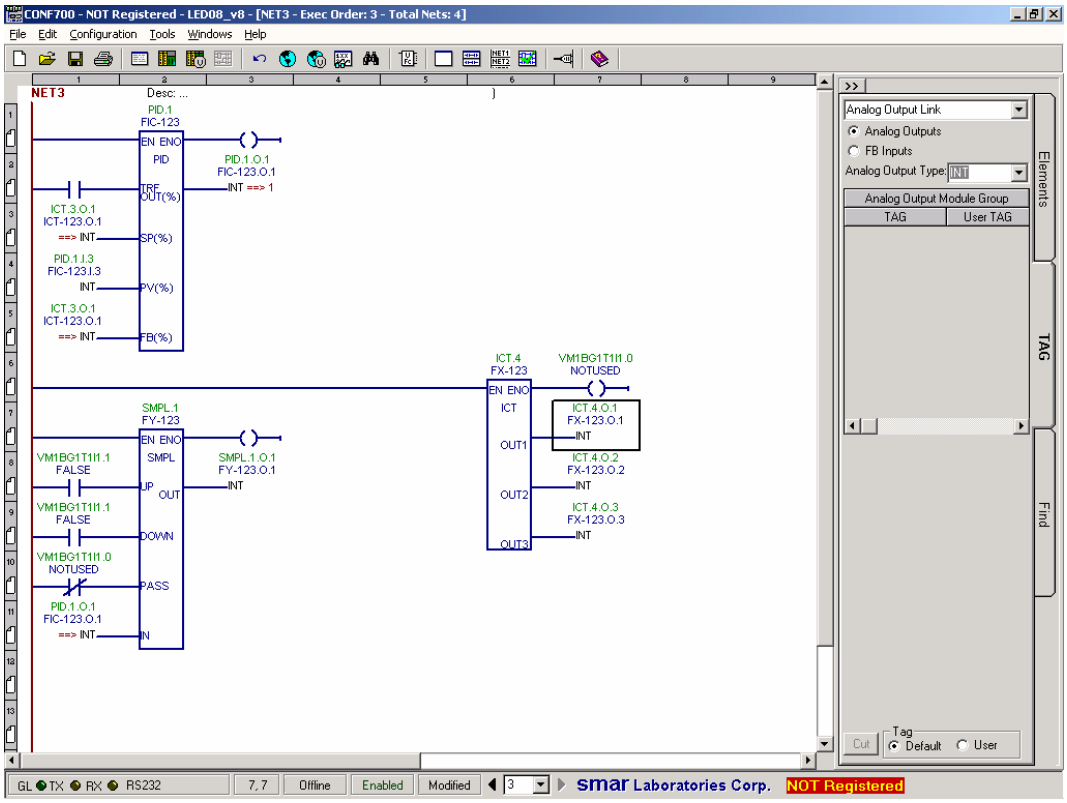


Figure 3.50- Setting the Setpoint using an ICT Function Block

Click the setpoint input of the PID block in order to configure it. Then, click at the TAGs tab and choose the ICT input.

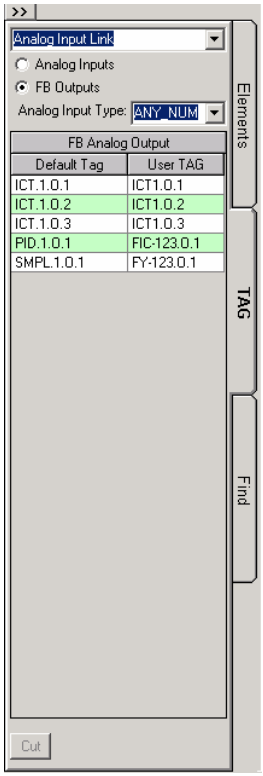


Figure 3.51- Connecting the ICT Outputs to the PID Block to generate a Setpoint

Select the “FB input” and choose the corresponding output from the ICT block. The setpoint of the PID block can now be set from the operator workstation by writing a value to the Modbus register that corresponds to the ICT block internal parameter. Note that the user cannot write directly to the ICT block output.

Note in the above configuration that the PID block SP input (at R3C1) has the same tag as the ICT block output (at R12C3), because they have been “linked”.

General Hints on the Network

Jumps can be done between subroutines based on conditions. All parameters in the LC700 are global and can be used in any network.

Row and column for the cursor are displayed at the bottom of the window as r, c.

Row and column are also used by the automatic CONF700 strategy checking them to indicate any cell with an error or potential discrepancy. By clicking Tools and then Preferences the user can select which information is to display in the ladder diagram while the user is building the control strategy. The information for the element can also be edited on the ladder diagram screen by clicking the right mouse button.

If the user needs more networks go to the menu: Edit/Add Network or click on  .

Printing Documentation

CONF700 prints out all of the relevant documentation for the hardware and software configuration, eliminating the need for separate documentation on third party tools. The documentation has been generated automatically as it is configured. All of the user descriptions, remarks and annotations that have been entered into the configuration, e.g. the global tables, are saved in the file as part of the configuration, and is printed.

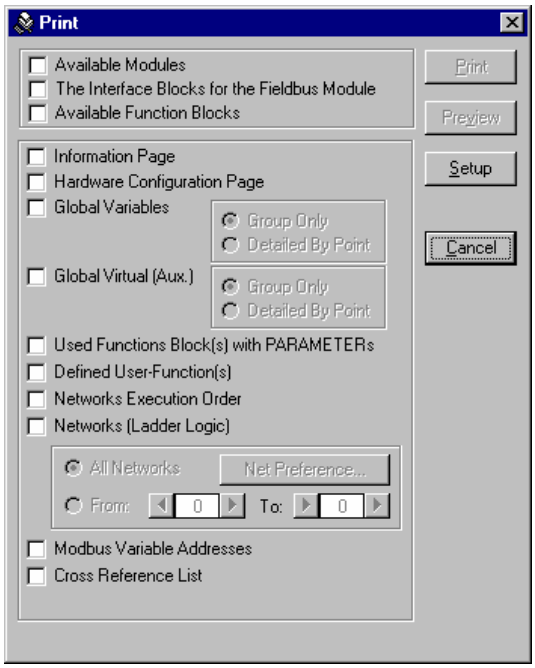


Figure 3.52- Printing Settings Window

Finding Things in the Networks

CONF700 allows the user to find, search and/or replace tags, FBs, UFs and their links in the ladder network. Basically a project has several pages of ladder network. So this tool is very helpful to the user.

Click on the Find tab and the following options are displayed in the toolbar at the right side of the CONF700 interface.

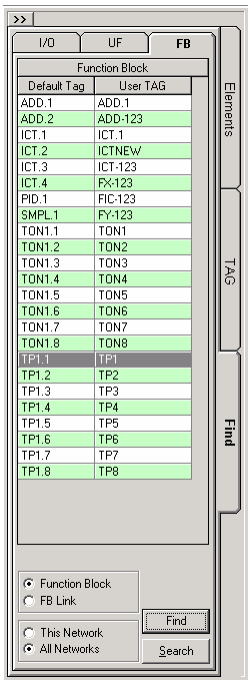


Figure 3.53 - CONF700 Toolbar: Elements Main Tab

The tab I/O in the option Find presents all the I/O modules inputs and outputs available. The user can select any tag from the I/O tab to find, search or replace it in the network. You can also select (click on) any relay or coil associated with a tag, CONF700 will automatically get the tag in the I/O tab.

The user always has the option whether to do this in “This Network” or in “All Networks”.

The tab UF will display all user function names, etc. while the tab FB displays all available function block names or their output links.

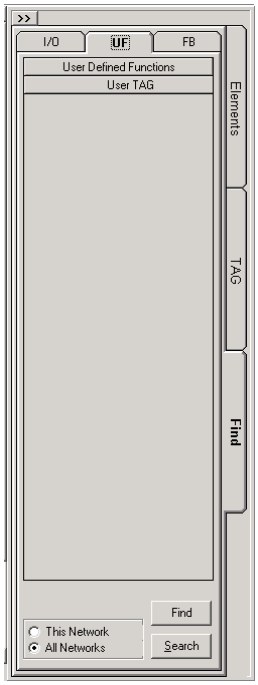


Figure 3.54- CONF700 Toolbar: UF Tab

Select (Click on) any element and click on the Find button. CONF700 will automatically highlight the element in the ladder logic.

Alternatively the user can also use the Search function. CONF700 will pop out the following window:

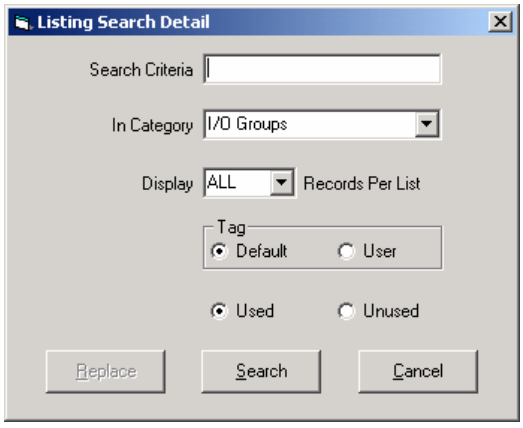


Figure 3.55- The CONF700 Search Feature

The field “Search Criteria” allows the user to type a string to get the matched result. The search criteria can also include a wildcard “*”, such as “aaa*”, “*aaa” or “*aaa*”, which represents to find something beginning with “aaa”, ending with “aaa” or having “aaa”. For example, if the element to be found is all M-020 inputs, the user can enter “M-020*” as the search criteria. Alternatively the user can select a tag, etc. in the Tab Find and click on the search button.

The user must specify which category of the desired element is.

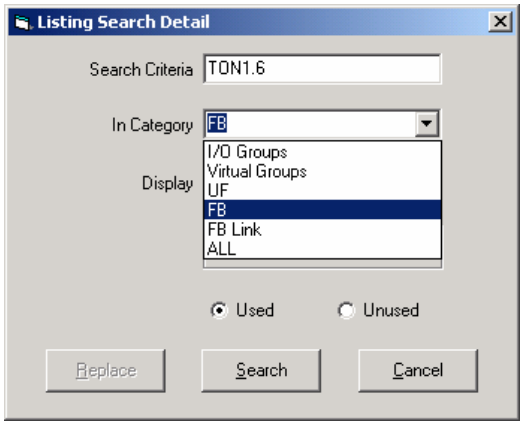


Figure 3.56- Searching A Specific Element

The categories are:

- FB- CONF700 will search for FB.
- I/O Groups- CONF700 will search for I/O group inputs and outputs
- UF- CONF700 will search for User Functions
- FB link- CONF700 will search for Function Block links.
- All- CONF700 will search all the categories.

Using the I/O Find Option

Select the desired I/O input or output and click on the Search button.

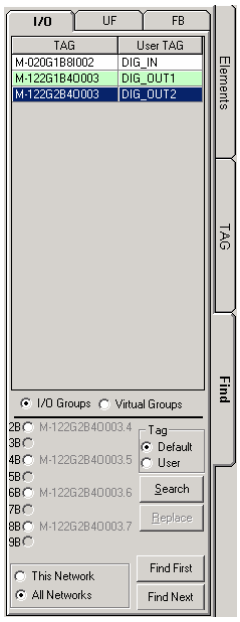


Figure 3.57- The find I/O Feature Tab

This will open the following window dialog:

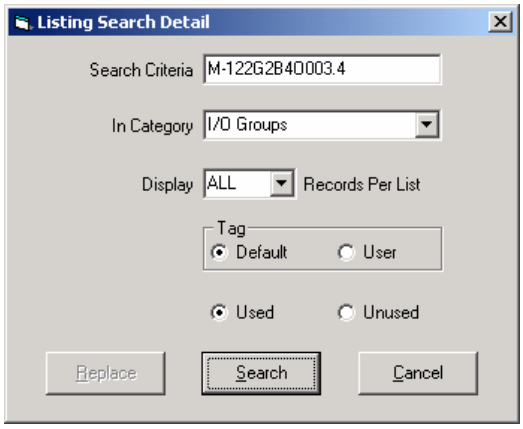


Figure 3.58-Searching An I/O Element

Next click on the Search button and CONF700 will automatically displays all occurrences (with their position information) of this input in a list at the bottom of the CONF700 interface:

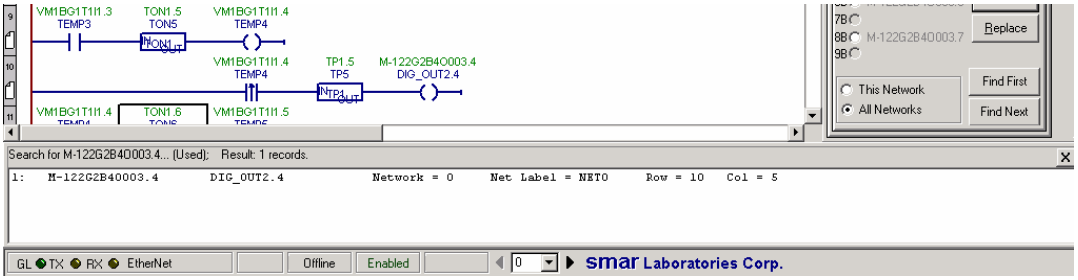


Figure 3.59- Listing The Found I/O Elements

Click any record in the list. CONF700 will automatically highlight the desired element in the network.

If the user wants to replace the search result with a specific tag, “Replace” command button is able to this. Note: Replace function is only for I/O tags replacement.

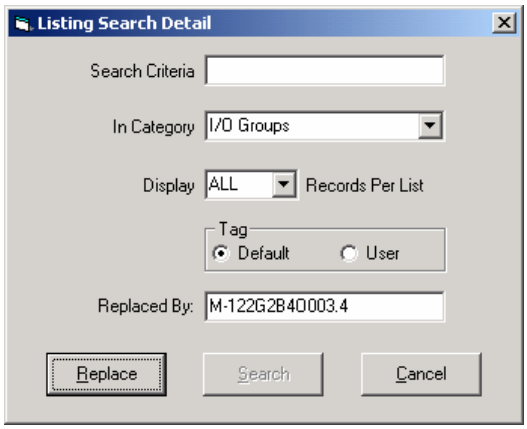


Figure 3.60- Replacing An Element In The Ladder Logic

The user needs to select a tag in the I/O tab of the “Find” tab as the replacement. CONF700 will automatically fill it in the “Replace By:” field in above window. The user can not type in this field to avoid an invalid entry.

Type the desired criteria in the field Search Criteria like in the “Search” function. Click on the Replace button. A replacement list will appear at the bottom of the CONF700 interface as following:

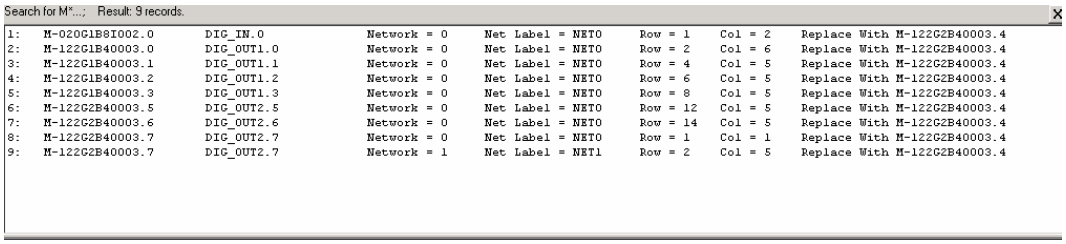


Figure 3.61- Replacement List

Like the Search function, single click on any record in the list, CONF700 will highlight the cell of that element. Double click on any record in the list, CON700 will confirm whether to replace it or not:

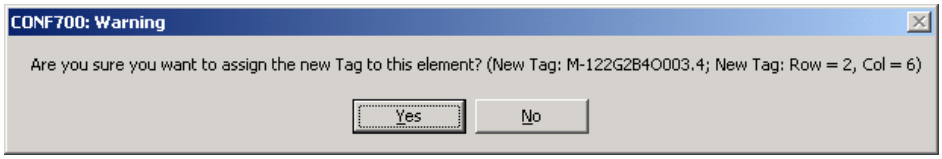


Figure 3.62- Confirming The Replacement

Click on Yes to replace the tag. CONF700 will also validate whether the replacement tag can be assigned to the element after the user clicks on Yes.

The User Function Find Tab

The user can also find user function blocks. To do that, simply select the desired User Function in the grid.

Click on the Find or Search button to get the desired UF in the network or a list of it at the bottom of CONF700 interface. Click at any record in the list. CONF700 will automatically highlight the desired UF in the network.

This option does not support the replace option.

The Function Blocks Find Tab

The user can find function blocks and function blocks links.

Simply select the desired Function Block in the grid. Next, click on the Find First button to get the first occurrence of the desired function block or function block link in the network. The Find Next button will get the next occurrence of the desired function block or function block link. The user can also click on the Search button to get a list of it at the bottom of CONF700 interface. Click at any record in the list. CONF700 will automatically highlight the desired FB or FB link in the network.

This option does not support the replace option.

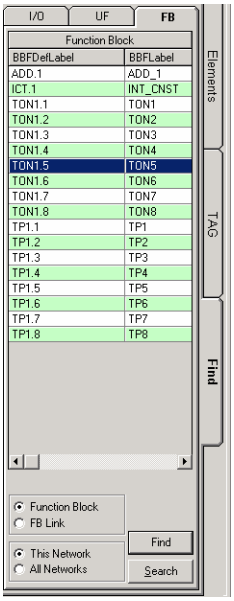


Figure 3.63- Function Blocks Find Tab

Adding Notes to the Ladder Logic Programming Lines

If the user checks the “Ruler/Note” option in the “Preferences” settings, CONF700 will allow the user to add a note to each Ladder programming line. This makes the ladder to be easier to read and debug.

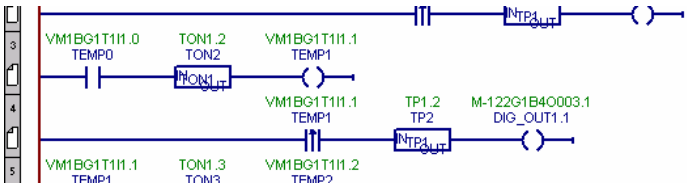


Figure 3.64- Adding Notes to a Ladder Diagram

Click on the small note icon in the ruler that is located at the left of the CONF700 interface as the following picture.

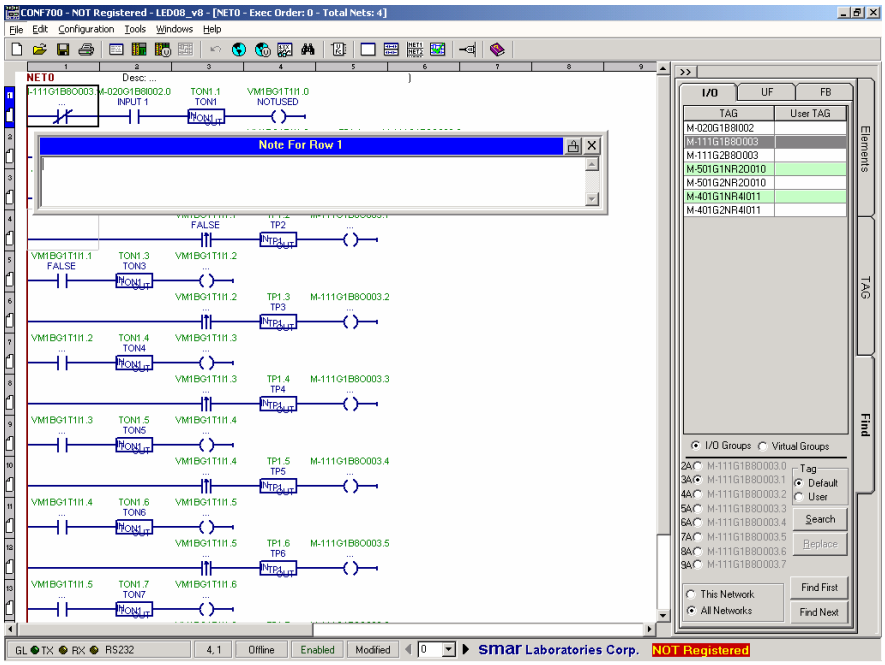


Figure 3.65- CONF700 Notes: Each Ladder Programming Line has its Notes Window

CONF700 will pop out a box where the user can add/modify note to each ladder programming line. The box will be under the line. If it doesn't fit under the line, it will be placed above the line. The user can drag and drop the Note box to any place and also resize it and lock the note box position by clicking on the Lock command button. It is a Lock/Unlock toggle command button.

So, the user can alternate through all notes of the current network just clicking at each note tool in the left side of CONF700 interface.

Configuration Memory Usage and Execution Time Estimation

In the configurator software CONF700 Menu item: "Check Configuration" from "Configuration" menu, will make an automatic analysis of the configuration and give as result the memory usage and execution time estimation for this configuration.

After downloading this configuration to the CPU, the Online connection Dialog gives the exactly execution time.

For manually estimate the memory usage, use the Hints on the next chapter.

CPU Memory

Memory	Size
CPU-700-D3	28 KB
CPU-700-D3R	23 KB
CPU-700-E3	52 KB
CPU-700-E3R	44 KB

Memory Usage: To calculate the user configuration memory usage (in bytes), use the following formula:

Memory = Modules + Networks

Where:

Modules
All I/O modules used in the configuration require a memory space, to save the I/O values and scan tables. For each module used, determine the memory size required from the I/O MODULE table, and add all results from this calculation.

Networks
The User Configuration, represented by the Ladder Logic Diagram, requires a memory space. To calculate this value, use the NETWORK Table, and add the values for each elements, function blocks, user function, …, used in the Ladder Logic (Network).

User Function
Require a space for itself, that contain the code to be executed and tables. For each User Function used in the Network, determine the size from the USER FUNCTION table, and add all results from this calculation.

Function Blocks
Also requires a memory space for it's parameters and outputs values.
For each Function Blocks used in the Network, determine the size from the FUNCTION BLOCKS table, and add all results from this calculation.

I/O MODULES

Name	Description	Bytes
M-001	2 Groups of 8 24VDC Inputs (Optically Isolated)	9
M-002	2 Groups of 8 48VDC Inputs (Optically Isolated)	9
M-003	2 Groups of 8 60VDC Inputs (Optically Isolated)	9
M-004	2 Groups of 8 125VDC Inputs (Optically Isolated)	9
M-005	2 Groups of 8 24VDC Inputs-Sink (Optically Isolated)	9
M-010	2 Groups of 4 120VAC Inputs (Optically Isolated)	9
M-011	2 Groups of 4 240VAC Inputs (Optically Isolated)	9
M-012	2 Groups of 8 120 VAC Inputs (Isolated)	9
M-013	2 Groups of 8 240 VAC Inputs (Isolated)	9
M-020	1 Group of 8 On/Off Switches Inputs	9
M-101	2 Groups of 8 Open Collector Driver Outputs (Optically Isolated)	13
M-102	2 Groups of 8 Transistor Outputs (source	13
M-110	2 Groups of 4 120/240VAC Outputs (Optically Isolated)	11
M-111	2 Groups of 8 120/240VAC Outputs (Optically Isolated)	13
M-120	2 Groups of 4 NO Relay Outputs (Double Level of Isolation)	11
M-121	2 Group of 4 NC Relay Outputs (Double Level of Isolation)	11
M-122	1 Group of 4 NO and 4 NC Relay Outputs (Double Level of Isolation)	11
M-123	2 Groups of 8 NO Relays Outputs (Double Level of Isolation)	13
M-124	2 Groups of 8 N.O. Relays Outputs (optically Isolation)	13
M-125	2 Groups of 4 N.C. Relay Outputs (optically isolated)	11
M-126	1 Group of 4 N.O. and 1 Group of 4 N.C. Relay Outputs	11
M-201	1 Group of 8 24VDC Inputs and 1 Group of 4 NO Relays Outputs (Optically Isolated)	18
M-202	1 Group of 8 48VDC Inputs and 1 Group of 4 NO Relays Outputs (Optically Isolated)	18
M-203	1 Group of 8 60VDC Inputs and 1 Group of 4 NO Relays Outputs (Optically Isolated)	18
M-204	1 Group of 8 24VDC Inputs and 1 Group of 4 NC Relays Outputs (Optically Isolated)	18
M-205	1 Group of 8 48VDC Inputs and 1 Group of 4 NC Relays Outputs (Optically Isolated)	18
M-206	1 Group of 8 60VDC Inputs and 1 Group of 4 NC Relays Outputs (Optically Isolated)	18
M-207	1 Group of 8 24VDC Inputs and 1 Group of 2 NO and 2 NC Relays Outputs (Optically Isolated)	18
M-208	1 Group of 8 48VDC Inputs and 1 Group of 2 NO and 2 NC Relays Outputs (Optically Isolated)	18
M-209	1 Group of 8 60VDC Inputs and 1 Group of 2 NO and 2 NC Relays Outputs (Optically Isolated)	18
M-302	2 Groups of 8 Pulse Input , 16 channels, 2 KHz	

Name	Description	Bytes
M-401	8 Analog Inputs for Current and Voltage	46
M-402	8 Low Signal Inputs for RTD, TC, mV, Ohm	32
M-501	4 Analog Outputs with Individual Terminals for Current and Voltage	63
FB-700	Fieldbus Module with Fieldbus Foundation protocol	*1
PS-AC-R	Power Supply Module	4

Notes:
For the FB-700 - Fieldbus Module: there is one table to use (the following tables) to calculate the number of bytes used in one configuration, each Fieldbus Module has one different memory allocation that depend on the number of function block (CIDD, CIAD, CODD, COAD) used in the module. The first row of the Table indicates the number of bytes use by the Module itself, even if no blocks are used. For each function block, just count the number of byte if at least one block is used. If no function blocks of that type are used, count as ZERO bytes, for that block.

FB-700	Fieldbus Module with Fieldbus Foundation protocol
	BYTES
Module FB-700	89
CIDD (if any)	3
CIAD (if any) , (n = number of CIAD)	7 + n*32
CODD (if any), (n = number of CODD)	5 + n*16
COAD (if any), (n = number of COAD)	9 + n*128

The Network (Ladder Diagram)

ELEMENTS	BYTES	EXECUTION TIME (µs)
RELAY	4	27
COIL	4	34
JUMP	4	25
RETURN	4	24
LINE	0	0
VERTICAL LINE *(1)	2	18
COLUMN *(2)	8	20
User Function *(3)	*	*
Function Block *(3)	*	*
Network Page (Ladder Diagram)	12	-

Note:
Lines are solved in column processing.
For each group of consecutive vertical line in the column.

Example:

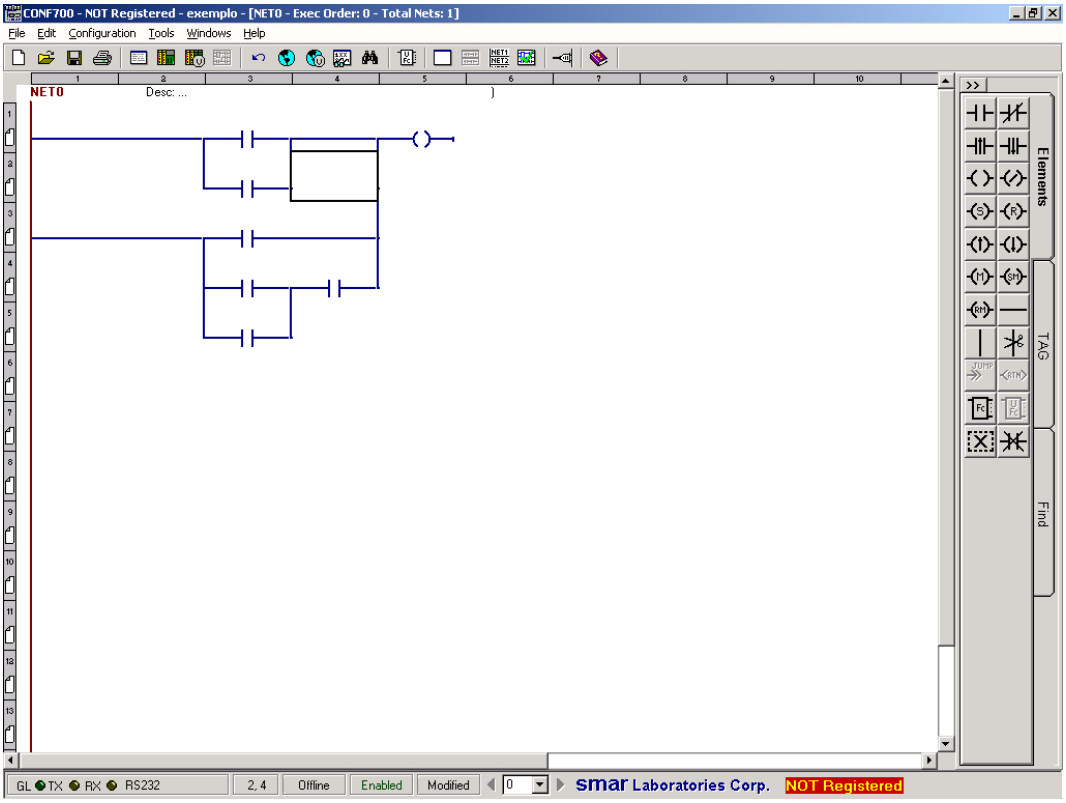


Figure 3.66- Estimating the memory usage of a Ladder Configuration

In this example we have 2 groups of vertical lines on column 1, 2 groups of vertical lines on column 2 and 1 group of vertical lines on column 3.

Then memory usage = 2x2 + 2x2 + 1x2

For each used column in the network page (Ladder Diagram).

Example: In the example shown above, we have 4 column used. Then memory usage = 4x8.
For User Function and Function Block memory space and execution time, look at the specific table. Each Function Block and each User Function allocates some memory space when inserted in the Ladder Diagram.

USER FUNCTION (UF)	BYTES
Each User Function	20
Each temporary used (TEMPn)	4
Each output used (OUTn)	7
Each AND or OR operator	4
Each “0” or “1” constant	5
Each Variable	11
Variable preceded by a rising or falling edge symbol (| or !).	17

1) The Function itself takes 20 bytes;
2) 4 bytes for each times a temporary variable in used (TEMPn);
3) 7 bytes for each function output (OUTn);
4) 11 bytes for any other variable not preceded by a TRANSITION-SENSING symbol (^ or !).
5) 17 bytes for any other variable preceded by a TRANSITION-SENSING symbol (^ or !).
6) 4 bytes for each "AND" and "OR" operation;
7) 5 bytes for each "0" or "1" constant;

Example: Consider the User-Function called "SELECT". This function simulates a MULTIPLEXER switch with four inputs (INA, INB, INC, IND) selected by SEL1 and SEL2. OUT1 represents the output of the switch, OUT2 is to show if any of the inputs is "0". The function also prepares the COIL24 to indicate how the selection line could be tolerated any change.

```
SELECT:
TEMP1:=/SEL1*/SEL2*INA;
TEMP2:=/SEL1* SEL2*INB;
TEMP3:= SEL1*/SEL2*INC;
TEMP4:= SEL1* SEL2*IND;
COIL24:= ^SEL1 + !SEL1 + ^SEL2 + !SEL2;
OUT1:=TEMP1 + TEMP2 + TEMP3 + TEMP4;
OUT2:= INA * INB * INC * IND;
END_SELECT
```

It has:

Rule	Description	Total of Bytes
1	We need 20 bytes for the User-Function	20
2	Temporary variables were used 8 times.	8x4
3	Two function outputs (OUT1, OUT2).	2x7
4	We have 12 variables not using the transition-sensitive symbol (^ or !)	12x11
5	We have 4 variables using the transition-sensitive symbol.	4x17
6	We have 14 "AND" and "OR" operations.	14x4
7	No constants	0
	Total of bytes required	322

Function Blocks

ID	FUNCTION NAME	DESCRIPTION	VER	Bytes Ladder	Bytes Parameters	TIME (µs) integer (*)	TIME (µs) real (*)
0	**TON1**	Timer On-Delay	1.00	6	18	34	-
1	**TOF1**	Timer Off-Delay	1.00	6	18	33	-
2	**TP1**	Timer Pulse	1.00	6	18	37	-
3	**CTU1**	Counter Up	1.00	6	18	38	-
4	**TON**	Timer On-Delay	2.00	6	14	41	-
5	**TOF**	Timer Off-Delay	2.00	6	14	38	-
6	**TP**	Timer Pulse	2.00	6	14	58	-
7	**CTU**	Counter Up	2.00	6	12	52	-
8	**CTD**	Counter Down	2.00	6	12	62	-
9	**BWL**	Bitwise LOGIC	2.30	6	8 + 2*n + T	60	-
10	**NOT**	Bitwise NOT	2.30	6	6 + T	43	-
11	**ICT**	Integer Constants	2.00	6	20	43	-
12	**RCT**	Real Constants	2.00	6	38	-	183
13	**ITR**	Conversion Int To Real	2.00	6	8	-	463
14	**RTI**	Conversion Real To Int	2.00	6	6	557	-
15	**TRC**	Truncation	2.00	6	6 + T	61	-
16	**ABS**	Absolute Value	2.00	6	6 + T	46	192
17	**SQR**	Square Root	2.30	6	18 + T	2746	1933
18	**ADD**	Addition	2.00	6	4 + 2*n + T	55	666
19	**MUL**	Multiplication	2.00	6	4 + 2*n + T	74	373
20	**SUB**	Subtraction	2.00	6	8 + T	50	409
21	**DIV**	Division	2.00	6	8 + T	65	797

ID	FUNCTION NAME	DESCRIPTION	VER	Bytes Ladder	Bytes Parameters	TIME (µs) integer (*)	TIME (µs) real (*)
22	**MOD**	Modulo	2.00	6	8 + T	60	-
23	**SEL**	Binary Selection	2.30	6	8 + T	54	62
24	**MAX**	Maximum	2.00	6	4 + 2*n + T	58	373
25	**MIN**	Minimum	2.00	6	4 + 2*n + T	58	567
26	**LMT**	Limiter	2.00	6	10 + T	353	375
27	**MUX**	Multiplexer	2.30	6	4 + 2*n + T	73	371
28	**GT**	Decreasing Sequence	2.00	6	2 + 2*n	51	541
29	**GE**	Decreasing Monotonic Sequ.	2.00	6	2 + 2*n	51	250
30	**EQ**	Equality	2.00	6	12 + 2*n	609	740
31	**LE**	Increasing Monotonic Sequ.	2.00	6	2 + 2*n	54	646
32	**LT**	Increasing Sequence	2.00	6	2 + 2*n	54	349
33	**NE**	Inequality	2.00	6	16	301	442
34	**XLIM**	Cross Limit / Rate-of-Change	2.30	6	40 + T	1097	2529
35	**PID**	PID Controller	2.00	6	62 + T	5678	4825
36	**STP**	STEP Control	2.00	6	26	273	1003
37	**RTA**	Real Time Alarm Clock	2.00	6	18	43	-
38	**BTI**	BCD_TO_INT Conversion	2.30	6	6 + T	47	-
39	**ITB**	INT_TO_BCD Conversion	2.30	6	6 + T	46	-
40	**BTB**	BYTE_TO_BIT Conversion	2.30	6	2	39	-
41	**TOT**	Totalization	2.30	6	30	2008	1995
42	**SMPL**	Sample Hold with Up/Down	2.30	6	40 + T	4142	3126
43	**ARAMP**	Automatic Up/Down Ramp	2.30	6	36 + T	4481	4454
44	**LIN**	Linearization	2.30	6	38 + T	89	1683
45	**FIFO**	FIFO (Data Logger)	4.36	6	34 + 2*F_Size	2040	1700
46	**ACC**	PULSE ACCUMULATOR	4.37	6	34	140	*
47	**ACC_N**	PULSE ACCUMULATOR	8.41	6	136	200	*
48	**OSEL**	Binary Output Selection	8.45	6	24	89	102
49	**MATH**	Multivariable Equations	8.45	6	112	29860	30068

Where:
F_Size = number of Modbus register selected for the FIFO (FIFO Size)
n = number of Function Block INPUT **IN**, selected in the Extensible dialog box (Extensible Value Parameter) when inserted THIS Function Block in the Network.
T = number of bytes for the selected TYPE for the Function Block Output (See Table Below)

TYPE	(T) BYTES
BOOL	1
INT	2
WORD	2
REAL	4
ANY	8
ANY_NUM	8
ANY_INT	8
ANY_REAL	8
ANY_BIT	8

Execution Time: is in microseconds, and depends on the type of Inputs/Outputs selected for the function block. Function Block that has a differ**ent type than Integer or Real, we use the table column integer to show its value.**

Connecting to the LC700

The easiest way to connect a PC to the LC700 controller is by using the serial port. The first port of the LC700 CPU is a RS-232C that can communicate with any PC that has a standard serial port available. Most PCs come with two serial ports described as COM1 and COM2.

Cable

The user can use a Smar made "C232-700" 72-inch cable that connects a DB9 Male serial port from a PC to the DB9 Female in the CPU-700 (LC700 CPU) or assemble a cable. (See the drawing below). The figures illustrate how to assemble the cable in case the PC has either a DB9 or a DB25 connector.

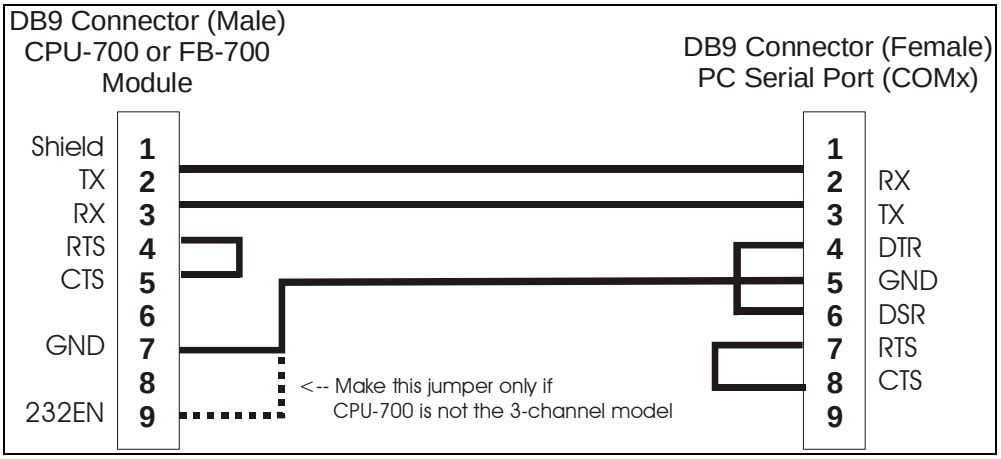


Figure 3.67- Serial Cable To Connect PC and CPU-700: DB9 Connector

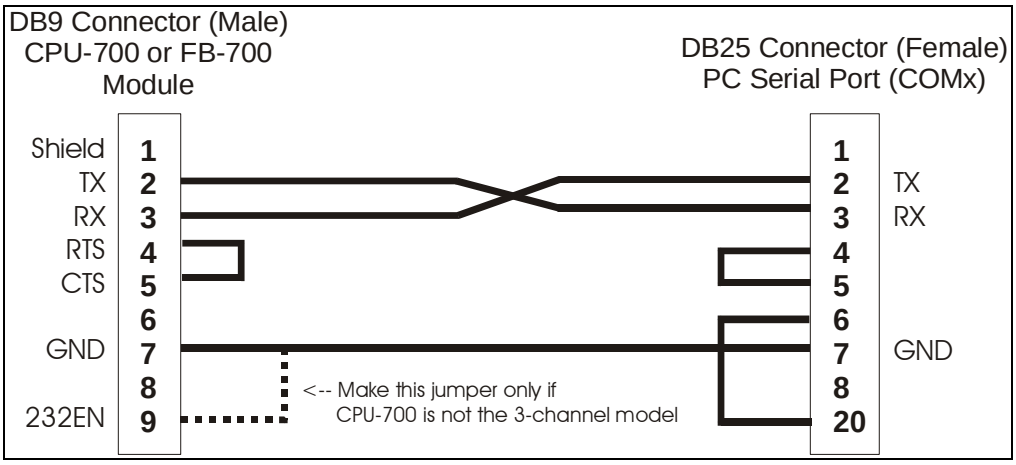


Figure 3.68- Serial Cable To Connect PC and CPU-700: DB25 Connector

There are other cables that can be used in a LC700 System. Please refer to the LC700 User's Guide for further information.

Now the user will need to know the communication settings for the LC700. The user will need to locate and place the communication setting switch to the default position. This is in case the user has forgotten how the CPU was set or if this is the first attempt at communication.

Communication Switch

In the CPU module, between the communication ports, there is a group of 4 switches. Using a small screwdriver make sure that the lowest key is pointing to the left (user looking to the front of the module). In this position the CPU has default communication parameters: Device ID= 1, baudrate= 9600 bps and even parity.

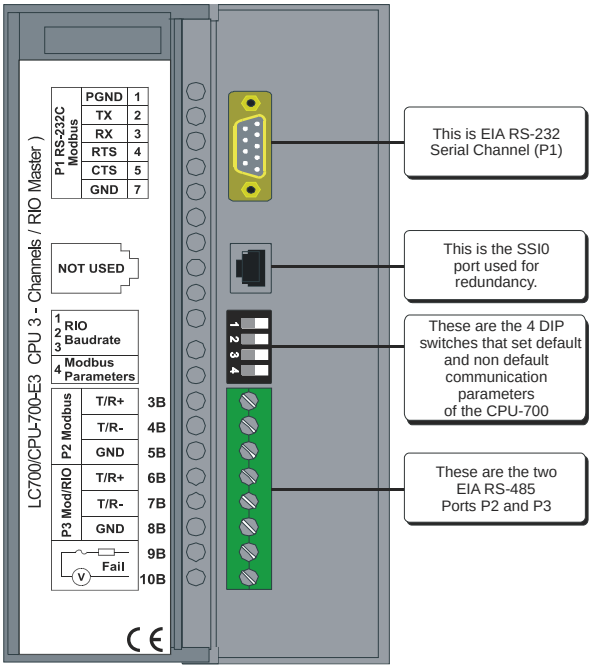


Figure 3.69- CPU-700 Channels

Later these parameters may be changed using the CONF700 but they will only have effect when the communication key is on the non-default position.

For further details refer to the LC700 User's Guide.

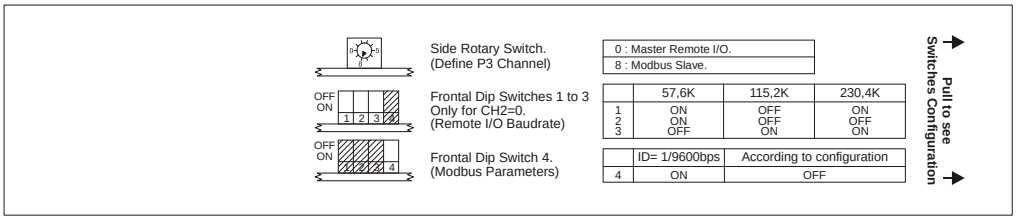


Figure 3.70- The Dip Switches And Rotary Key Specifications

Physical Layer and Time out

Now verify the communication settings to allow the CONF700 to communicate with the LC700 CPU. Go to the menu Tools/Comm. Settings… and the following dialog box will open.
Set interface for RS-232 physical layer as show in the figure.

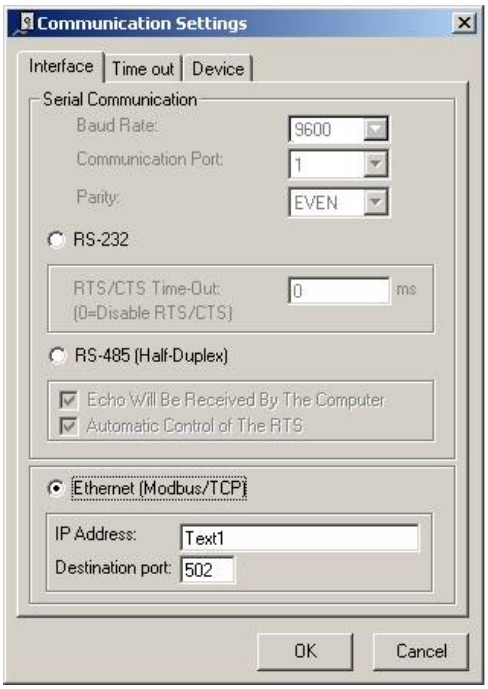


Figure 3.71- Setting the EIA-232 Configuration Parameters

Next click on the "Time out" tab, set the time out and the number of times the computer should try in case of a bad communication.

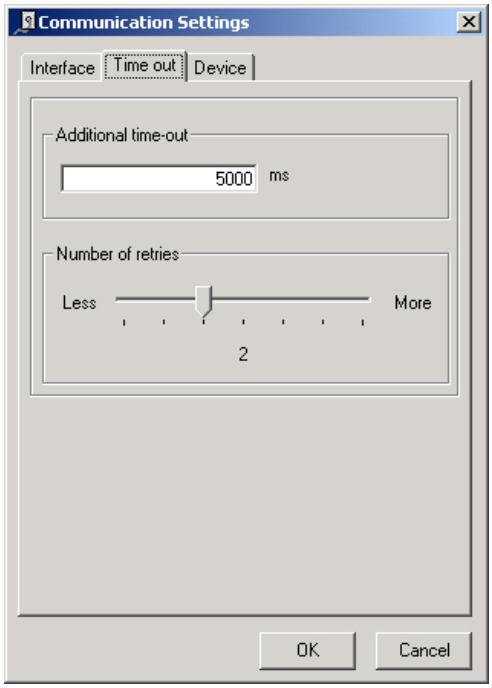


Figure 3.72- Communication Settings: Time Out

Changing the CPU-700 Communication Settings

Consider that we have a serial RS-232 connection between the CPU-700 and the PC running the CONF700. Make sure the serial cable is installed, the CONF700 is set for RS-232 and the Communication Switch is in the default position. Now open the LC700 ONLINE dialog box using the menu: Tools/Online or click on  .

CONF700 will try to connect with a LC700 CPU as soon as the online mode is called. If CONF700 cannot detect the LC700 presence it will time out and wait with the LC700 ONLINE dialog box opened. This gives the user a chance to modify some parameters to correctly set the communication.

In case CONF700 finds a CPU700 that matches the communication parameters, this means it was already configured, it will add information on the Device, Version, Release, Configuration Name and present Status. See dialog box below.

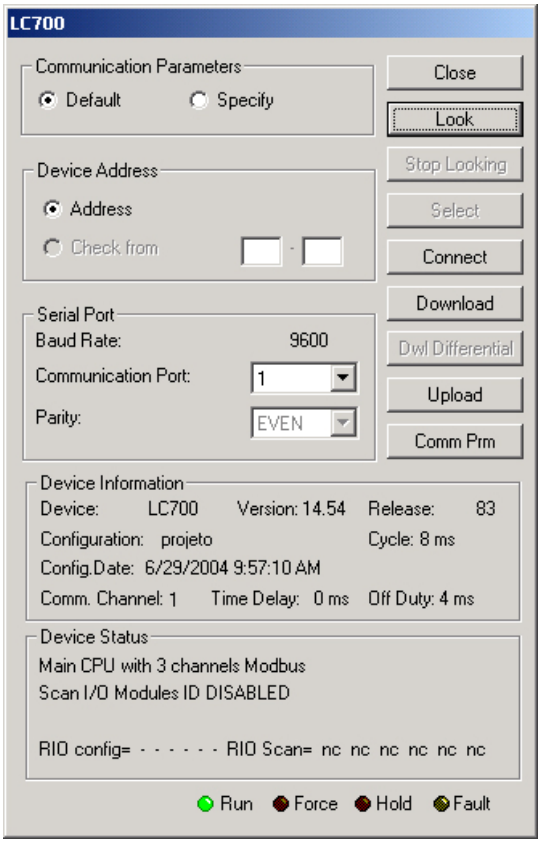


Fig 3.73- CPU-700 Communication Settings

It is important to stress that the CPU module has 4 Dip Switches used to set the communication default parameters. In this case the device address is 1 and baudrate is 9600 bps. The easiest way to meet these conditions is to select the “Default” option under “Communication Parameter” on the figure above. In this condition we cannot make changes in the Serial Port frame.

Changing the LC700 Communication Parameters

To change the LC700 communication parameter, click on the “Comm Prm” button and work in the following dialog box.

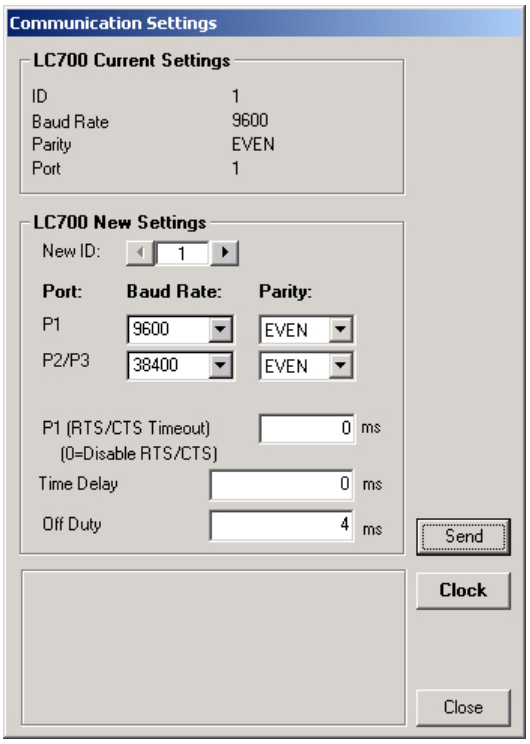


Figure 3.74- CPU-700 Communication Parameters

After these parameters have been changed the Send button will be enabled. The LC700 CPU will receive information and will inform of these modifications which will only take place when the user changes the communication DIP switch (see Chapter 4 Troubleshooting for more details).

There are 3 serial ports available in the CPU-700. A port P1 (EIA RS232) and two EIA RS485 ports (P2 and P3). The user will be able to set each one of these ports: baudrate, parity and other specific parameters.

Port P1
- **Baudrate** (9600 bps)
- **Parity** (Odd, Even or without parity)

- **RTS/CTS Timeout**

CTS: Signal that indicates the device is ready for transmission.
RTS: Signal to request data transmission

The PC requests a connection with the CPU. The LC700 receives this request and processes it. Next, the LC700 sends the RTS signal and it waits for the CTS signal during the time interval set on the RTS/CTS Timeout parameter.

- **Off Duty:** Available time for communication when the CPU is not running a ladder diagram. The bigger the value set for Off Duty, the bigger the available time for communication.

- **Time Delay:** The PC sends a frame to the CPU-700, i.e., makes a communication request. The CPU-700 waits the amount of time set on Time Delay to process the frame request and send the answer.

NOTE:
For better performance of your system we recommend:

- OFF Duty must be set as 20 % of the Ladder cycle of execution, and also is recommended do the fine tuning according to the communication performance.
- Time Delay depends on the Workstation's processor. If the processor is higher than a Pentium MMX 233 MHz we recommend that Time Delay is set for 4 ms. Otherwise, we recommend that the Time Delay is set with a value higher than the default value.

IMPORTANT
If the Time Delay or Offdutty is configured with the value “0”, CPU will work with the default values for the selected parameter.

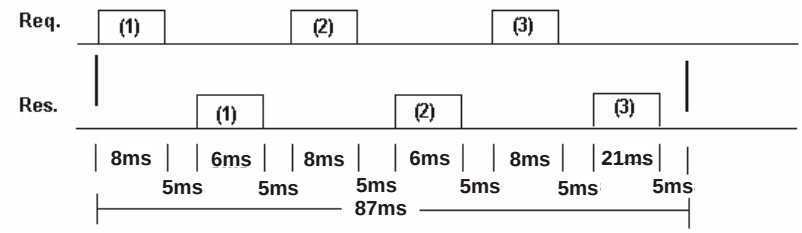
Ports P2 and P3
Port P2 is the serial EIA RS-485 channel and has two configurable parameters: baudrate (9600 to 115200 bps) and parity (odd, even or without parity). Port P3 is the other EIA-485 serial port. When the port 3 is used in Modbus communication (refer to the item Operation mode of the CPU-700 in the LC700 – User´s Guide), the configured parameters baud rate and parity are also valid for this port.

Communication Optimized

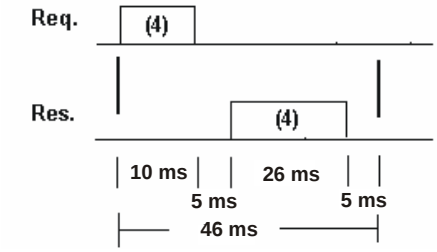
A more optimized method for the communication uses BlockView. The View is a list of the Configuration points (AI, DI, Ao and DO). It is configured inside the LC700 and can optimize the communication speed, thus, it compresses the configured points in the same Modbus frame message.

There is a comparison, below, between reading of the points using the View and not using the View feature.

Reading Without View
1) Module M020 – 8 pts (req: 8 bytes – res: 6 bytes)
2) Module M110 – 8 pts (req: 8 bytes – res: 6 bytes)
3) Module M402 – 8 pts (req: 8 bytes – res: 21 bytes)



Reading With View
4) The same reading above (req: 10 bytes – res: 26 bytes)



Observation: The programmed Time Delay is 5 ms.

Modbus Message Framing

The Modbus RTU protocol has a frame basically starting with an interval of silence followed by an address, function, data, and CRC checking (Cyclic Redundant Checking). Another interval of silence to indicate the end of the message

Interval of start of message	Address	Function	Data	CRC check	Interval of end of message

Frame size: 255 bytes

In the Modbus RTU standard data is transmitted in bytes with hexadecimal information from 0-9, A-F. Each byte is sent in the following format:

1 Start Bit
8 Data bits, least significant sent first.
1 Bit for parity (odd or even, if there is parity)
1 Stop bit

Start Bit	Data (8 bits)	Parity	Stop bit

List of Implemented Modbus Commands

Function Number	Function Name	Description	Implementation
1/01H	Read Coil Status	Reads the ON/OFF status of discrete outputs (0xxxx - references, coils) in the slave	OK
2/02H	Read Input Status	Reads the ON/OFF status of discrete inputs (1xxxx - references) in the slave	OK
3/03H	Read Holding Registers	Reads the binary contents of holding registers (4xxxx references) in the slave	OK
4/04H	Read Input Registers	Reads the binary contents of input registers (3xxxx references) in the slave	OK
5/05H	Force Single Coil	Forces a single coil (0xxxx reference) to either ON or OFF	OK
6/6H	Preset Single Coil	Presets a Value into a single holding register (4xxxx reference)	OK
15/0FH	Force Multiple Coils	Forces each coil (0X ref.) in a sequence of coils to either ON or OFF	**OK
16/10H	Preset Multiple Regs	Presets values into a sequence of holding registers (4xxxx references)	*OK
17/11H	Report Slave ID (the LC700 ID do LC700 is 2)	Return a description of the type of controller present at the slave address, the current status of the slave Run indicator, and other information specific to the slave device	OK
22/16H	Mask Write 4xxxx Register	Used to set or clear individual bits in the a 4xxxx register	*OK
23/17H	Read/Write 4xxxx Register	Performs a combination of one read and one write operation in a single Modbus transaction. The function can write new contents to a group of 4xxxx register, and then return the contents of another group of 4xxxx registers	OK

* Only Implemented in firmware versions 1.30xx or higher
** Only Implemented in firmware versions 2.31.03 or higher

Ethernet Communication Settings

Time out for the LAN

For the Ethernet connection the time out will be very dependent on the execution cycle for the CPU-700 and the traffic in the network. It is a good hint to begin with 5000 ms and 3 communication retries before error notification.

Go to menu: Tools/Comm. Settings then click on Time out tab.

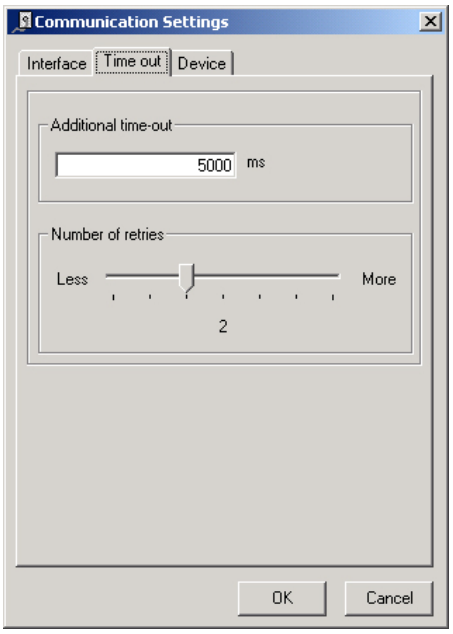


Figure 3.75-Time Out For Lan

ENET-700/ENET-710 IP address

To connect the PC to the LC700 system throughout the Ethernet the user will need an ENET-700/ENET-710 module and an Ethernet adapter card in the PC. The next figure shows where the ENET-700/ENET-710 module IP address is entered. Note that the communication through the Ethernet is based on the fast MODBUS/TCP. The port 502 is configured for this operation, but, if it is necessary, this port can be changed.

Go to menu: Tools/Comm. Settings.

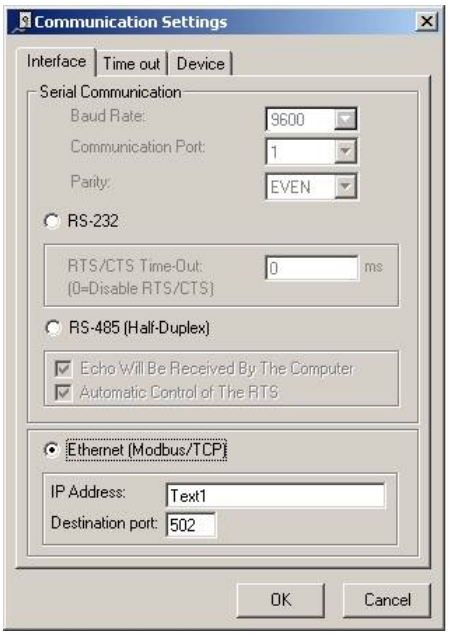


Figure 3.76- Setting the ENET-700/ENET-710 IP Address

Using ENET-700

The ENET-700 needs to be prepared according to the network settings established by the network administrator. The best thing to do is to consult your network administrator for details before proceeding.

The ENET-700 comes with a factory IP address but the user can change it. Windows application, Hyper Terminal, allows the user to make all initial setups on the ENET-700 using its factory IP address and referring to the port 9999 (configuration). See the factory IP address at the label fixed on the side of the ENET-700 module.

The next figure shows what the user basically will see in the Hyper Terminal when trying to make a connection.

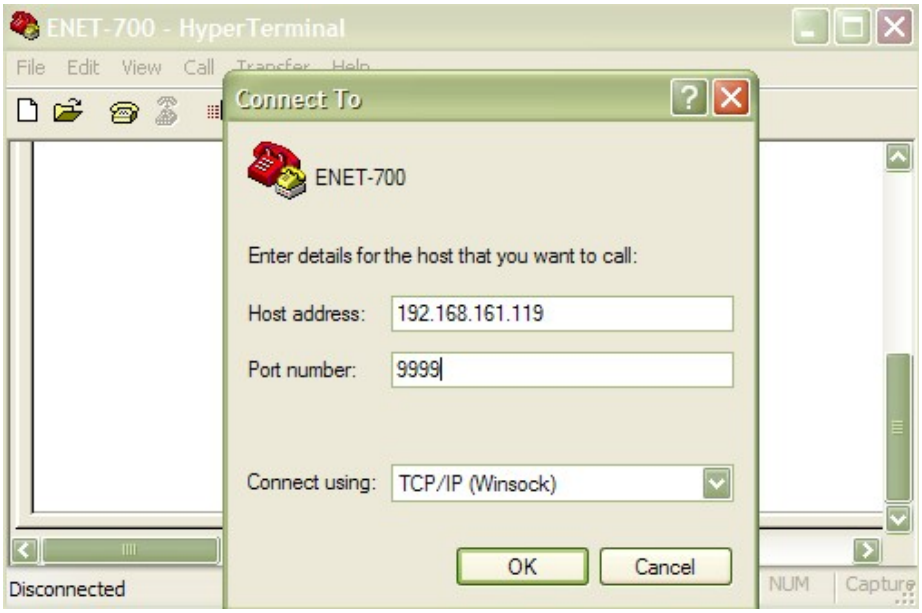


Figure 3.77- Connecting to the ENET-700/ENET-710 Module Through Telnet

As soon as the user clicks on the OK button and the computer settings are compatible to communicate with the ENET-700 the user will receive a text back. Press ENTER to begin configuration and the screen will show as follows.

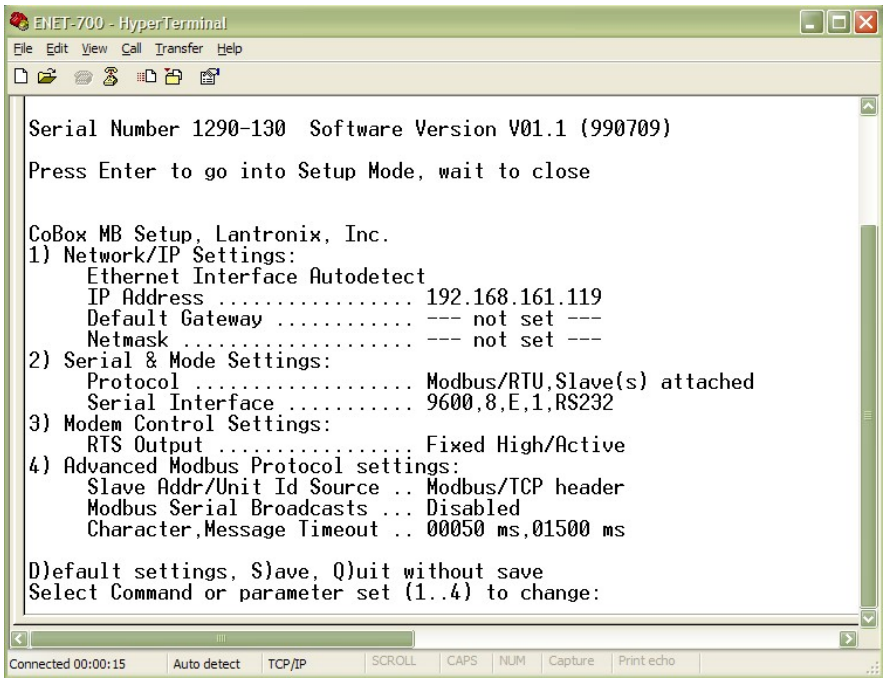


Figure 3.78- Setting The ENET-700 Module

Type the number of the item to be changed and follow the sequence of parameters At the end,

press “S” to save or “Q” to quit without saving.

Item 1 is the ENET-700 setting for the relationship with the network.

Note: For the Class C, the available IP Address Range is from 192.0.1.1 to 223.255.254.254, because the addresses that end with 0 and 255 are reserved.

Item 2 are the settings that need to mach the CPU-700 serial port.

Item 3 does not need to be changed.

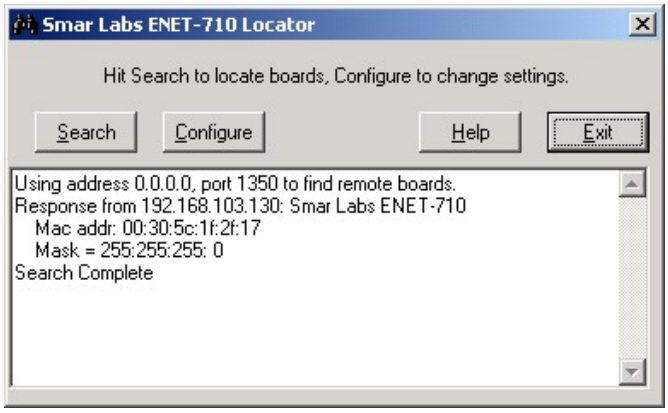
Item 4:

Slave Addr/Unit Id Source . .	Modbus/TCP header (as default)
Modbus Serial Broadcasts . . .	Disable (as default)
Character, Message Timeout . .	00050 ms, 01500ms (recommended)

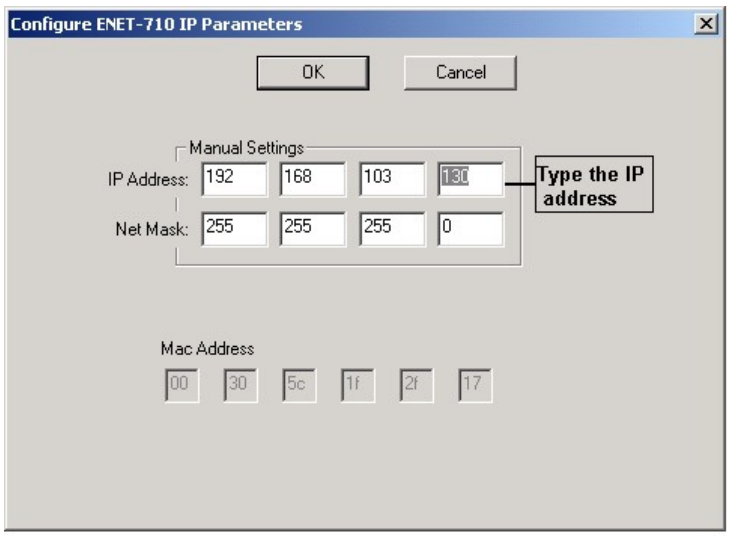
Using ENET-710

Double click on the 

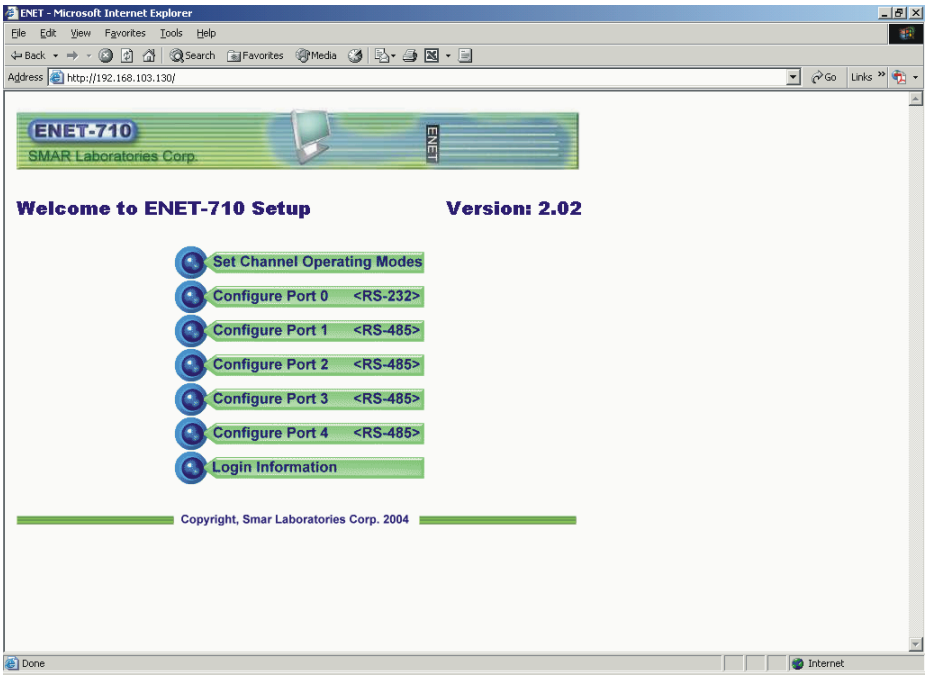
icon to open it. The following window will be showed.



Click on *Configure* button to set the IP address of ENET-710 module. After that, click on OK button.



Open a web browser and enter the IP address of the module into the web browser’s address bar. A view of the top-level configuration menu is shown below.

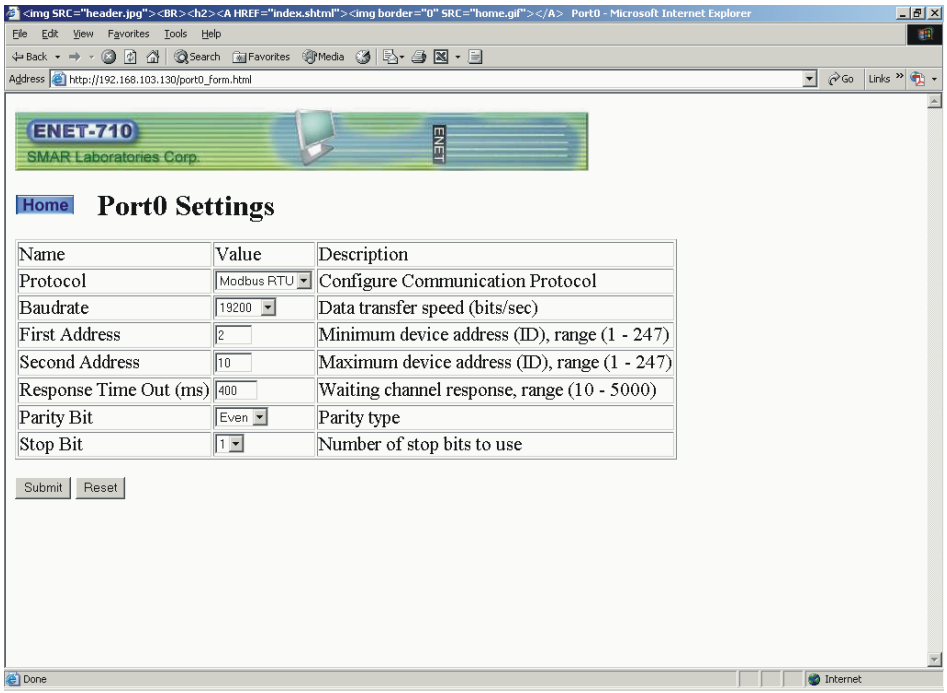


Configuration menu

The configuration buttons are username/password protected. Upon the initial selection of any one button, a dialog box will appear asking for a user name and password. Once the correct username and password are entered, the ENET-710 will remember them for the rest of the session. If you forget your current username or password or IP address, see the Defaults Switch section of this manual. Note that the username and password are case sensitive.

Serial Port Configuration

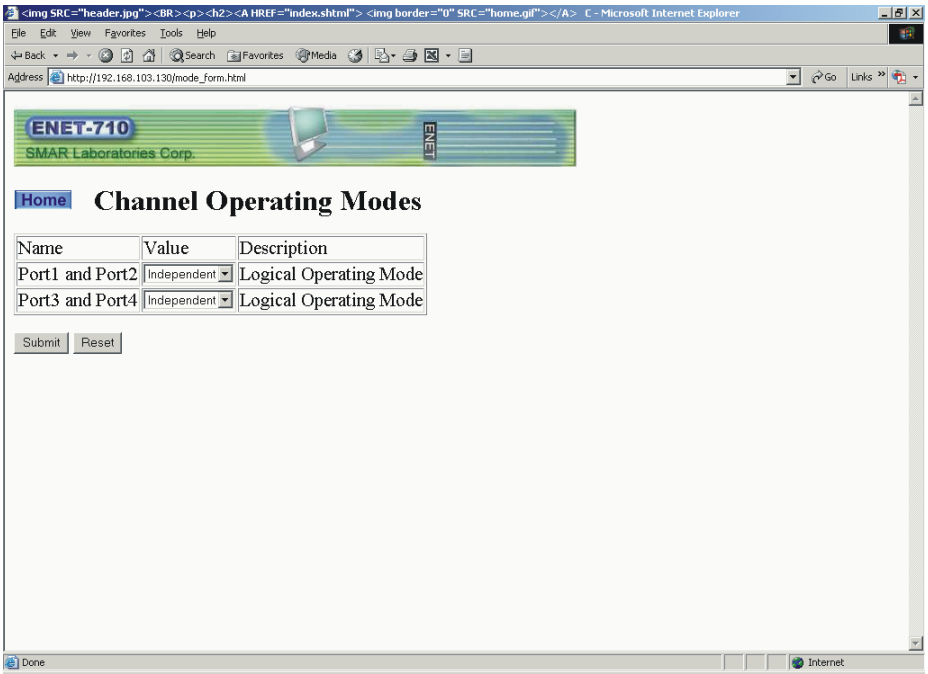
To set up any of the five serial ports, click on one of the Configure Port X (where X is 0,1,2,3, or 4) buttons on the main menu. The port configuration menu for port X is seen below (all 5 ports similar). Choose the desired serial protocol Modbus/RTU. Select the serial baud rate and character format (parity, stop bits). Parity can be even, odd, or none. You may choose 1 or 2 stop bits. Set up the serial device address ranges by entering the range low address into the First Address box and the range high address into the Second Address box. Enter desired device response timeout into the timeout box. Finally, click on the Submit button to save the new information.



Configuring the ENET-710's ports

RS-485 Port Operating Modes

To set up the RS-485 port operating modes, select the Set Channel Operating Modes button on the main configuration menu. Here a redundant mode configuration for the RS-485 ports can be enabled. The serial port pairs 1-2 and 3-4 can provide redundant operation in case the connection of one member of the pair fails. The modes include Independent (no redundancy), Parallel, and Ring. The menu is shown.

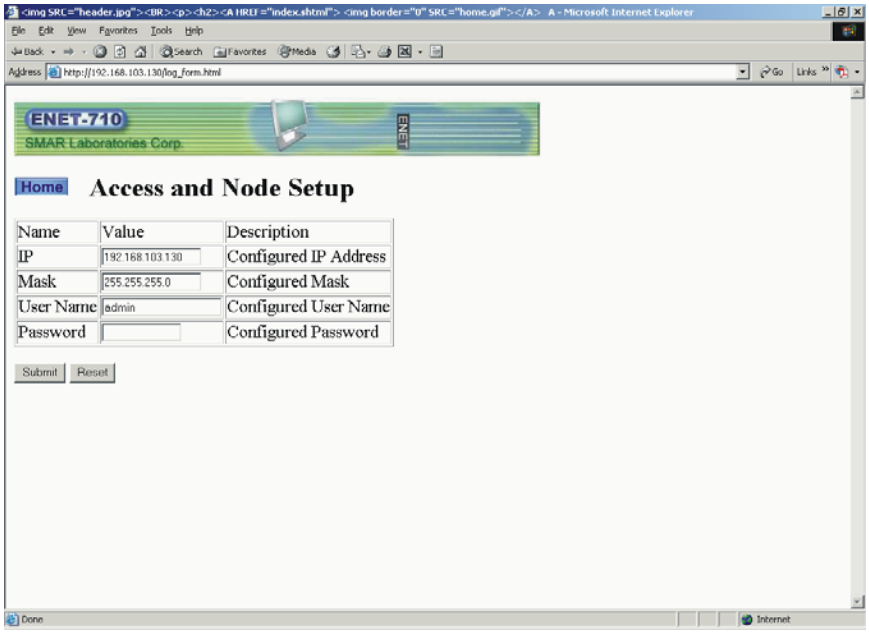


Configuring the RS-485 ports

For both parallel and ring mode, the operation is as follows: if communication is directed to an address that is configured to be on one port of the pair but there is no response, the ENET-710 will then try to communicate to the device using the other port in the pair.
Choose the desired mode of operation and click the Submit button to enable the setting.

Changing the IP Address and Username/Password

To change the login username/password pair or module IP address, select the Login Information button on the main menu. This brings up the page to setup the desired IP address and subnet mask, plus the username and password. Note that the username and password are case sensitive. To change any of these, enter the new information and click the Submit button to save. See the figure below.



For further information about configuration/problem correction, refer to the ENET-710 User manual.

Setting the Time out for ENET-700/ENET-710

For the timeout issue when using the CONF700 with the Ethernet one may consider how busy the network traffic is. Most of the time, the best way to evaluate is trying different values and seeing how successful a download or an upload can be. These tests should be done with “Times of retry” equal to 0 (zero). This way it can be evaluated as to how frequently a communication error occurs.
For a good performance on downloads and uploads we may put our efforts to achieve a relatively short time out. Typically a value between 500ms and 2000ms is expected.

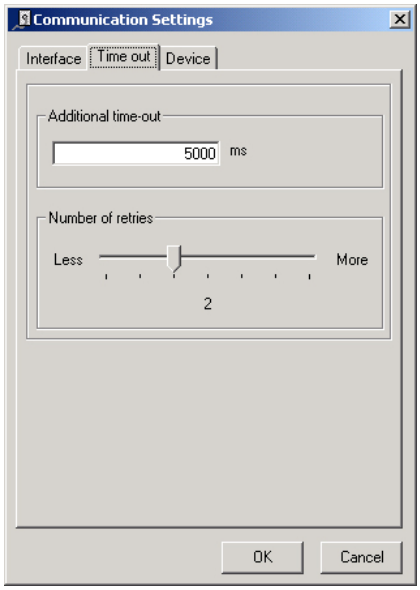


Figure 3.79- Timeout

Working ON-LINE

The on-line mode allows the user to download the configuration, operate the LC700, monitor, troubleshoot and optimize the configuration etc. Click on the On-line icon. Make sure the communication parameters selected match the ones in the CPU700 and that the cable is connected.

Go to menu: Tools/Online or click on  .

The LC700 will be located and identified. The CPU version, configuration name and operational status will be indicated.

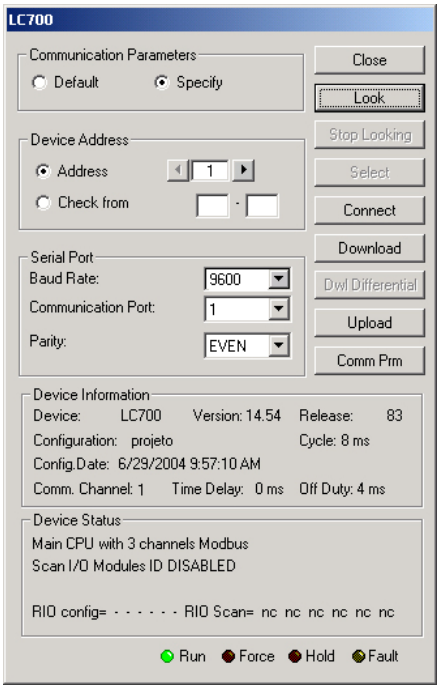


Figure 3.80- The LC700 ON-LINE Window

Downloading the configuration

It is a safe practice to save the configuration when it is ready. The user can also check for errors by running the Check Configuration procedure. To activate, go to the menu and select Tool/Check Configuration. If the verification comes clear then the user can send the configuration to the CPU-700. The user may also know that a configuration can be saved to a disk even if it is not completely cleared from errors for later editing.

The Download Processes

Click the “Download” button on the LC700 ONLINE dialog box to send the configuration to the CPU-700. At this time the CONF700 prepares all data that needs to be sent for buffers through a special compilation and begins to send. A horizontal bar graph will indicate the progress.

When the CPU-700 begins to receive the download information it will freeze the outputs to their last values and clear the Non-volatile memory to replace its contents with the NEW configuration. When the NEW configuration is completely downloaded, the CPU-700 will replace the frozen outputs with just the downloaded Safe-Values and ask the user if he wants to begin in the RUN or HOLD mode.

Whatever is the option to begin in HOLD or RUN mode, the CPU-700 starts with the outputs on the Safe-Value as configured. The user has the responsibility to remove the Safe-Value mode by pressing the “SAFE” button in the next dialog box.

Figure 3.81- CONF700 connected to the LC700

 **RUN/HOLD**

Click the Run button to toggle the CPU program execution between run and hold. Run a green LED indicates mode; hold a yellow LED indicates (pause) mode. Run is the normal mode where the user's application program is executed. In hold mode the user's application program is not being executed.

 **FREEZE INPUTS**

Click the Freeze button to toggle the input between scanning and frozen. A red LED indicates freeze mode. In freeze mode the inputs are not being scanned, and the user program executes based on the status of the inputs at the time of freezing the inputs.

 **FREEZE OUTPUTS**

Click the Freeze button to toggle the output between updating and frozen. A red LED indicates freeze mode. In freeze mode the outputs are not being updated, but the user program executes as per normal. Once the mode is returned to normal from the output freeze mode, the outputs will be updated accordingly.

 **FORCE FAIL-SAFE OUTPUT**

Click the Safe button to toggle the output between updating and safe. A red LED indicates force fail-safe mode. In fail-safe mode the outputs are forced to the predefined value configured in the global table.

On-line Monitoring

This is really an important tool for verifying how the control strategy is working in a complete relationship with the CPU700.

Monitoring is only possible if the CONF700 is already Online. See the Go Online is enabled click on this button. Otherwise it is necessary to upload the configuration present in the CPU.

The CPU700 must be in the RUN mode for monitoring purposes. See that the RUN LED is on, otherwise go menu: Tools/Run or click on  .

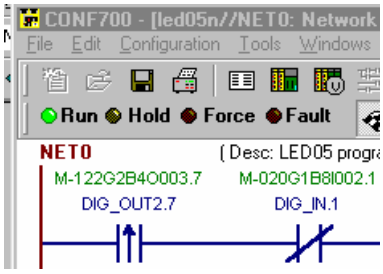


Figure 3.82- The Run Led

CPU700 in the RUN mode

Monitoring allows the user to see how each individual network is performing and make sure it is working the way it was desired during programming.

Inputs and outputs of function blocks can also be monitored. While in the Monitoring mode right click on the function block to be monitored and choose the option Yes on the preceding dialog box.

After the upload is complete go to menu: Tools/Monitor or click on  .

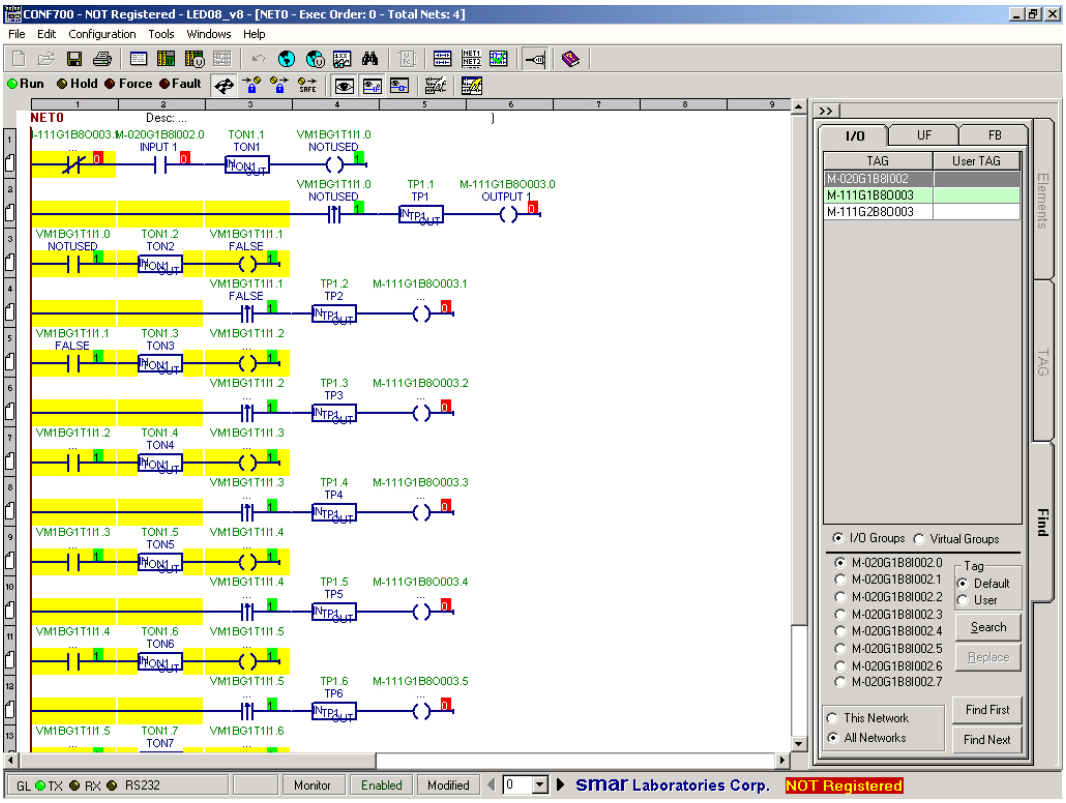


Figure 3.83- CPU-700 In Run Mode

Monitoring Function Blocks and Ladder Elements State

The user will be able to automatically monitor values of the inputs and outputs of all function blocks and all Ladder elements (relays and coils) in the current net, by clicking on the buttons below. These buttons will be enabled as soon as the LC700 is ON LINE with the workstation and after the user has selected to monitor the logic configuration.

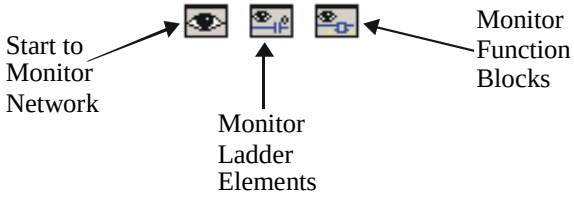


Figure 3.84- Monitor Buttons

Monitor Networks

This allows the use to monitor network state. The yellow lines indicate the power flow on the ladder logic diagram.

Monitoring Speed

Go to menu: Tools/Preferences and click on Misc tab. Inside the Online frame set the "Network Monitoring Period". Note that, according to the computer performance and the way the connection is set with the CPU700, it may not be possible to reach this kind of desired speed.

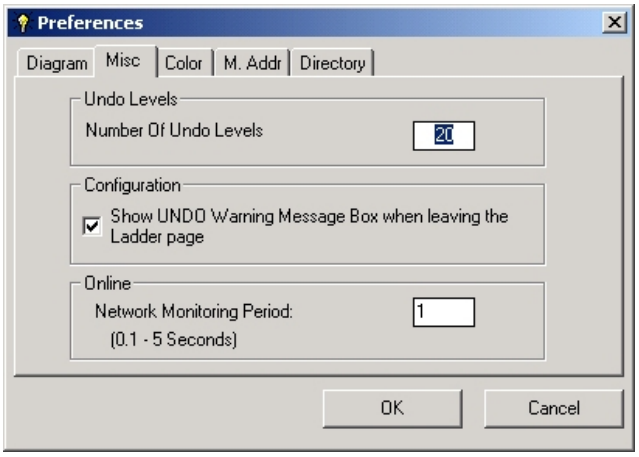


Figure 3.85- Setting the Monitoring Speed

Block I/O monitoring

It is also possible to see the analog value in the input and output links of a function block. Click on the function block and on the appearing dialog box click on the YES button. This action activates monitoring of the analog I/O links of that particular function block. Other function blocks can be similarly selected for monitoring.

In a similar way the monitoring of a function block can be disabled.

The more function blocks you select to monitor, the highest will be the monitoring time period.

The user may also use the button  to enable function block monitoring.

Forcing elements

To force a discrete element as a relay or coil, just click on it while in the monitoring mode. A dialog box will give the option to force to 0 (false) or 1 (true) during one cycle of the logic network of the LC700. After that the signal will assume a value according to the logic or nature of the forced point. For instance, if input was forced, after the forcing cycle, its value will be according to the input scan. If it is a virtual variable that is not being used in the logic and not changed via communication, it will keep the forced value.

Using the Monitoring feature in the MODBUS Addresses page

Using this feature the user will be able to monitor the values of the I/O points, Function Block variables, parameters and the values of Special Registers in the 'Modbus Address' page. The user should be online to view the values of the points. To monitor the desired points, click on the 'Modbus Address' button from the tool bar or from the Configuration menu item. The following window will appear.

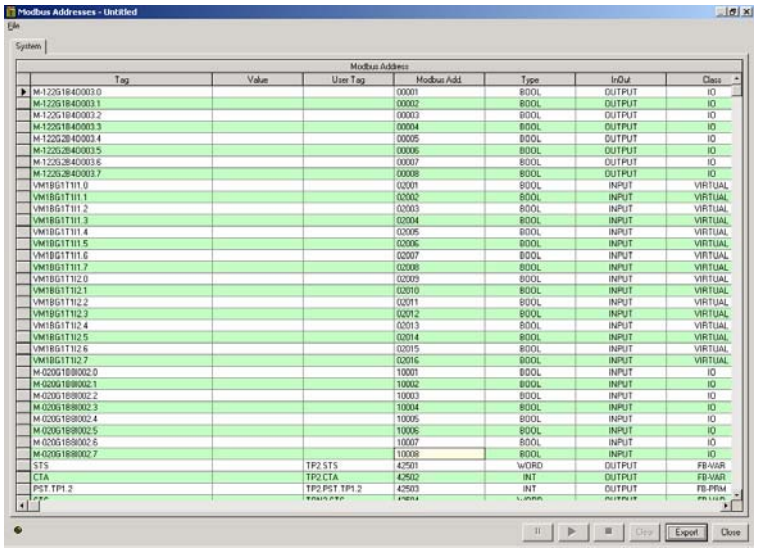


Figure 3.86- Using the Monitoring Button to Monitor MODBUS Variables In The MODBUS Address Window

The monitoring buttons are disabled because the user has to select at least one point for them to be enabled.

Select the desired points to view the values. This can be done by clicking the mouse on the button on the left side of each row. The cursor changes to an arrow when the user points the mouse on this button. To deselect the points simply click on the left of the row again. To clear the points selected click on the 'Clear' button.

To monitor the values click on the 'Monitor Values' button  .

The window now shows only the points being monitored along with their values.

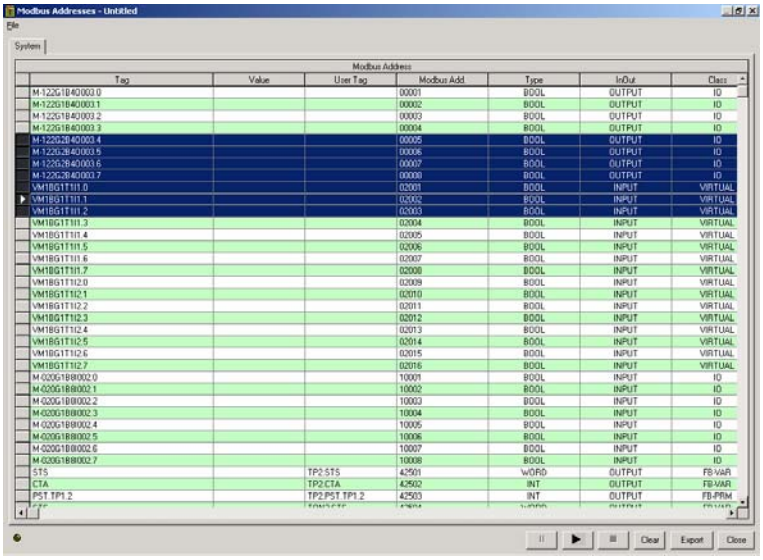


Figure 3.87- Monitoring Variables In The MODBUS Address Page

To temporarily stop monitoring simply click on the 'Pause' button  .

To start monitoring again click on the 'Continue' button (which is same as the pause button)  ,

To stop monitoring click on the stop button  . This will take the user back to the old window, which shows all of the points.

The points selected before will remain selected so the user doesn't have to select them again if it is necessary to monitor. Even if the user closes the window and comes back at a later time the points are still selected for the user convenience, simply click the 'Clear' button to clear the selection.

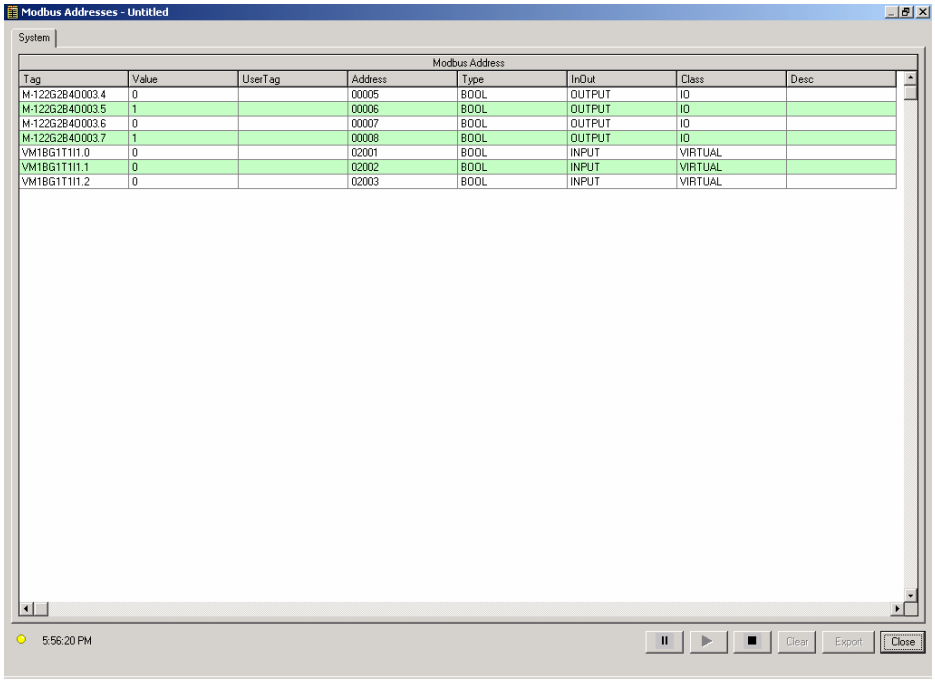


Figure 3.88- Selected MODBUS Variable Points

The user can also select multiple points by using the <shift> key. Select a point, press the <shift> key and select another point, all of the points in-between will also be selected.

ONLINE Mode

The LC700 controller supports two online edition modes: “Online Edit” and “Full Online Edition”. These two modes are different and competitive, the option to use one or the other depends on the situation.

- Online Edit  .

- Full Online Edition  .

In order to make the Ladder configuration easy to read in both online modes, the user can configure two different background colors for the Ladder in each online mode. In the toolbar, click on the Tool option and choose Preferences. Click on the tab Color. The following picture will be showed.

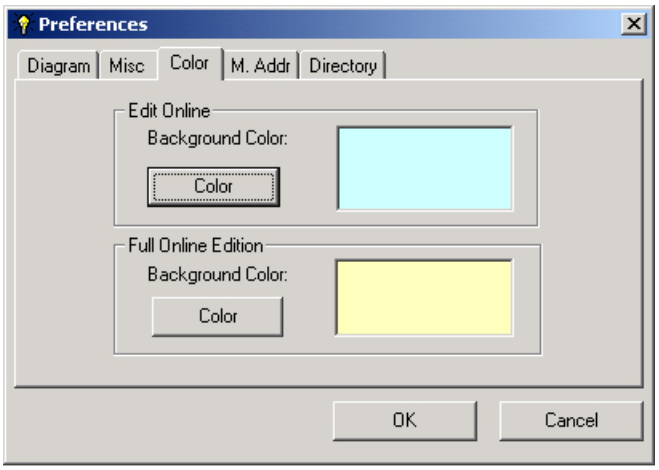


Figure 3.89- Changing the background color

Online Editing

In this mode, the user can:

- Change logic elements
- Change the logic element tags.
- Change blocks parameters.
- Change function blocks.
- Visualize the process after the changes (through monitoring the logic net).
- Undo changes, if they were not saved in the controller.
- Edit only one logic net by time.

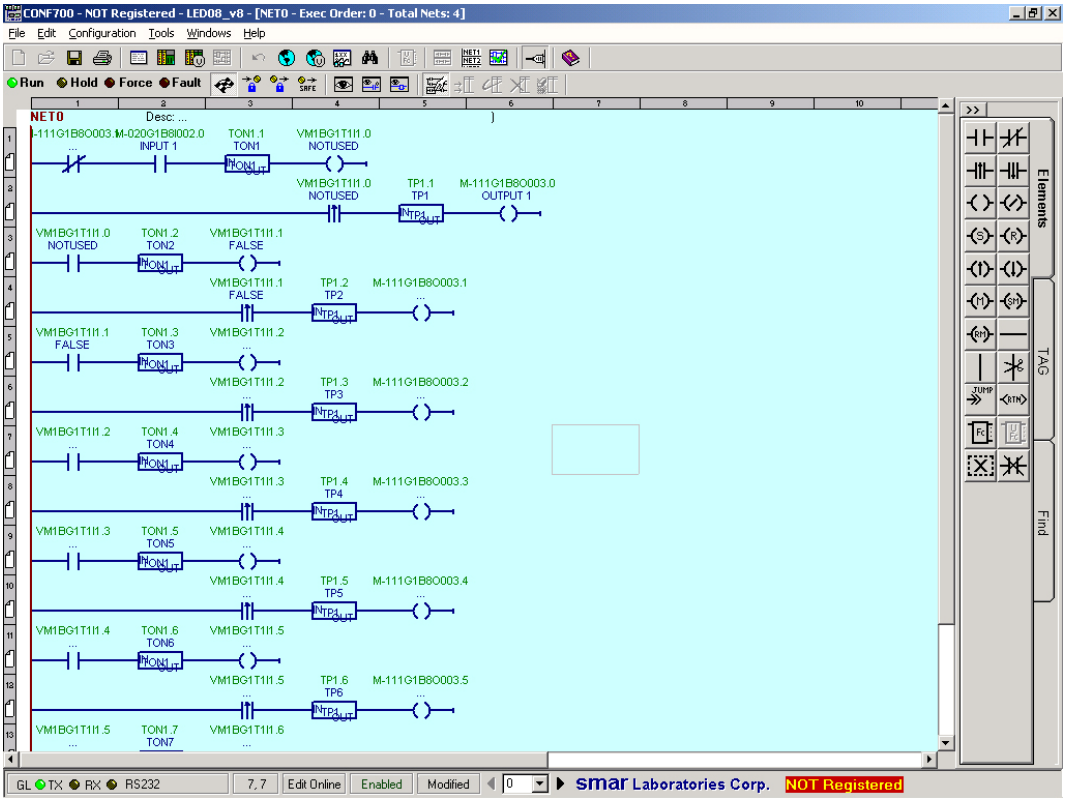


Figure 3.90- Editing Online Mode

How it works?

During the Online editing, the CPU700 allocates a separated memory area only for the new changes. That particular logic network that is being edited will be running during the whole editing process. In this way the user can progressively observe the system response for each of the changes.

Many modifications can be done before the decision to finally save them in the CPU700.

It will still be possible to cancel all modifications sent to the CPU700 if they were not saved to the EEPROM at this time!

To modify a logic element, its corresponding label or a function block parameter, “right click” on the cell where the object is. Many other modifications can be done in the same logic network before partially sending to the CPU700 to see the systems reaction. After several partial modifications, the user can finally decide to confirm the new configuration by saving it to the CPU700 EEPROM. Remember to save the modifications to the disk.

The online editing buttons

 Send all temporary changes to the CPU700. Temporary changes are indicated by a (**tc**) mark close to the object.

 Save all previous changes on the CPU700 NVRAM. Previous changes are represented by a (**pc**) mark close to the object. The logic object has this mark only when the modification has been sent to the CPU700 but not yet saved.

 Go back to the original or last saved logic network. The logic network will be replaced by the last one saved in the CPU700 NVRAM.

 Erase the temporary change selected in the network. In order to select the temporary change, the user should click in the changed element with the left button of the mouse.

IMPORTANT
The FIFO block cannot have its parameters changed in the Online Editing mode.

Full Online Edition

This option allows the user to change the configuration while the logic controller still executes the control cycle, without interruptions in the process, to guarantee the continuity without changes in the project.

In this mode, the user can also:
- Add/Remove Networks.
- Add/Remove Modules.
- Add/Remove Virtual Modules.
- Add/Remove RIO Interfaces.
- Add/Remove User Functions.
- Change Module Configurations.
- Move Modules.
- Undo changes that were sent to the controller.
- Visualize the control process after the update (through monitoring the logic net).

Important information to be considered before using the Full Online Edition mode

After executing the procedure for changes in the Full One Edition mode, the user should save the configuration file and generate the tag list and also the MCT table, through the TagList Generator software. This procedure is necessary because some tags can be included, deleted or changed.

During the full online editing of the configuration, the block parameters cannot be changed through supervisory programs. Due to, after sending the configuration to the CPU, the parameters that are obtained via configuration upload will be valid.

Using the Full Online Edition

The Full Online Edition mode can be accessed through three options, when the user is online.

- Click on the button on the toolbar  .
- Click on Tool → Full Online Edition → Edit.

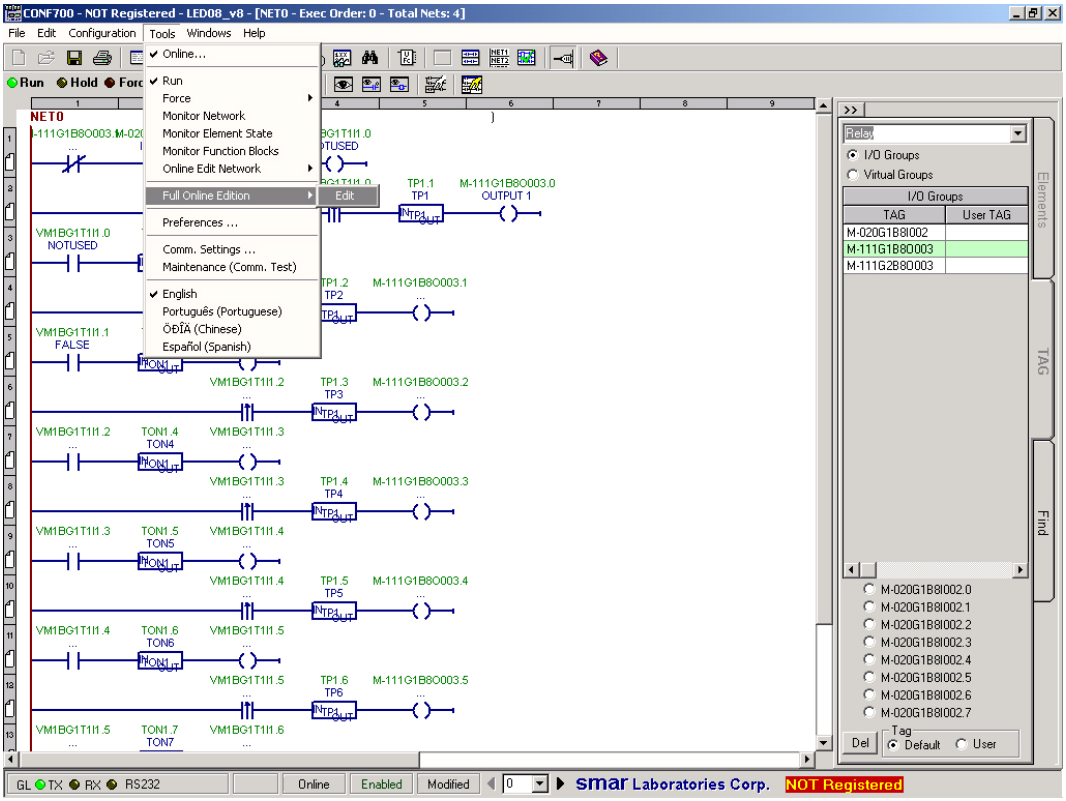


Figure 3.91- Option to start the Full Online Edition

- In the Network Page, click with the right button and the Full Online Edition window will appear.

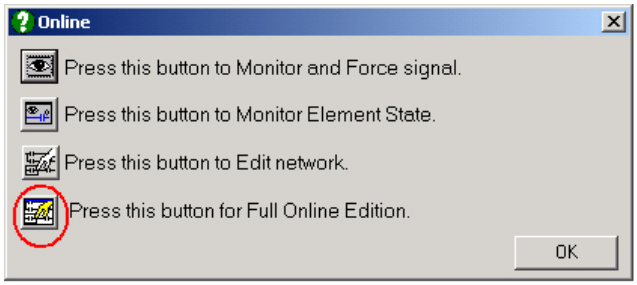


Figure 3.92- Option to start the Full Online Edition

The following dialog box will be showed to indicate that the user chose Full Online Edition mode.

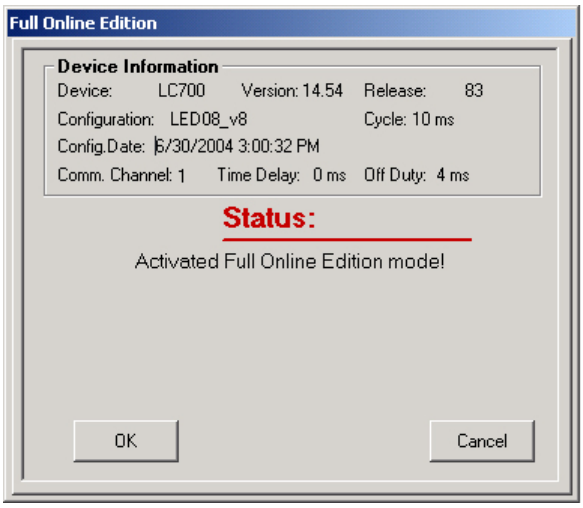


Figure 3.93- Active Full Online Edition

The Network Page will display the configured background by the user previously and at the lower - right corner will indicate “Full Online Edition”. The configuration will be on edit state.

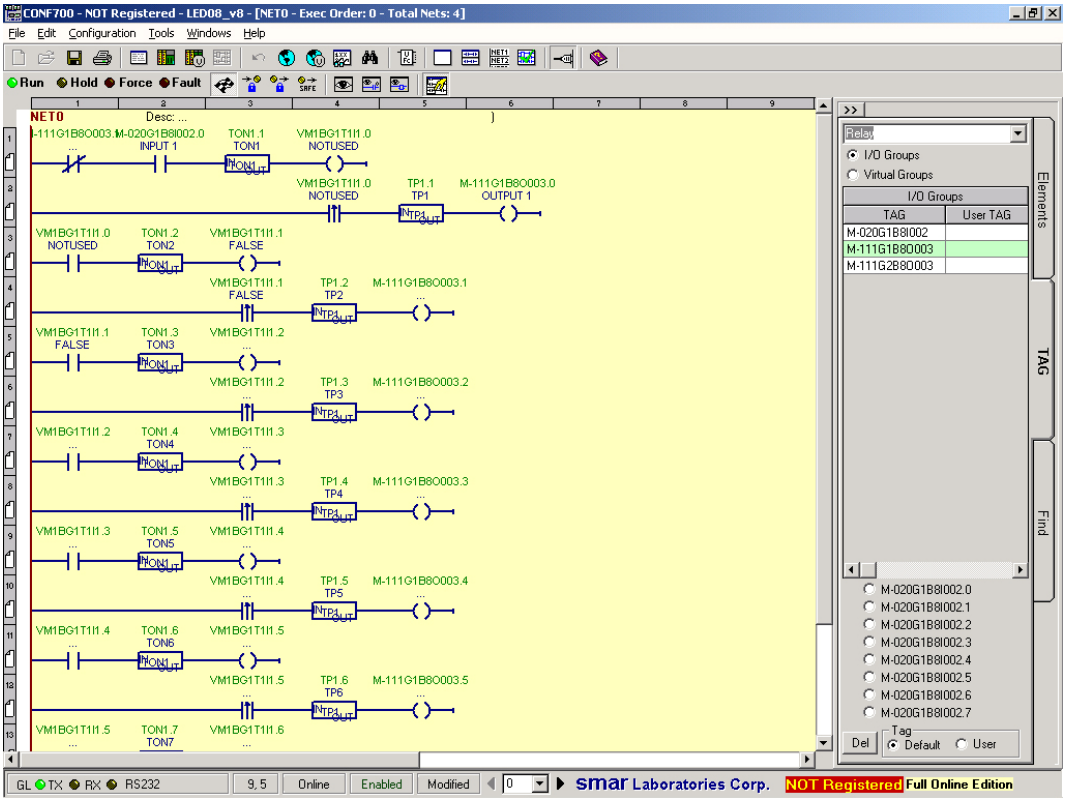


Figure 3.94- Ladder Window

After the first change in this configuration, it is impossible monitor any operations. This inhibition is due to the configuration in the LC700 controller being different from that displayed in the CONF700. If there is some monitoring, it will be disabled and the following message will be showed.

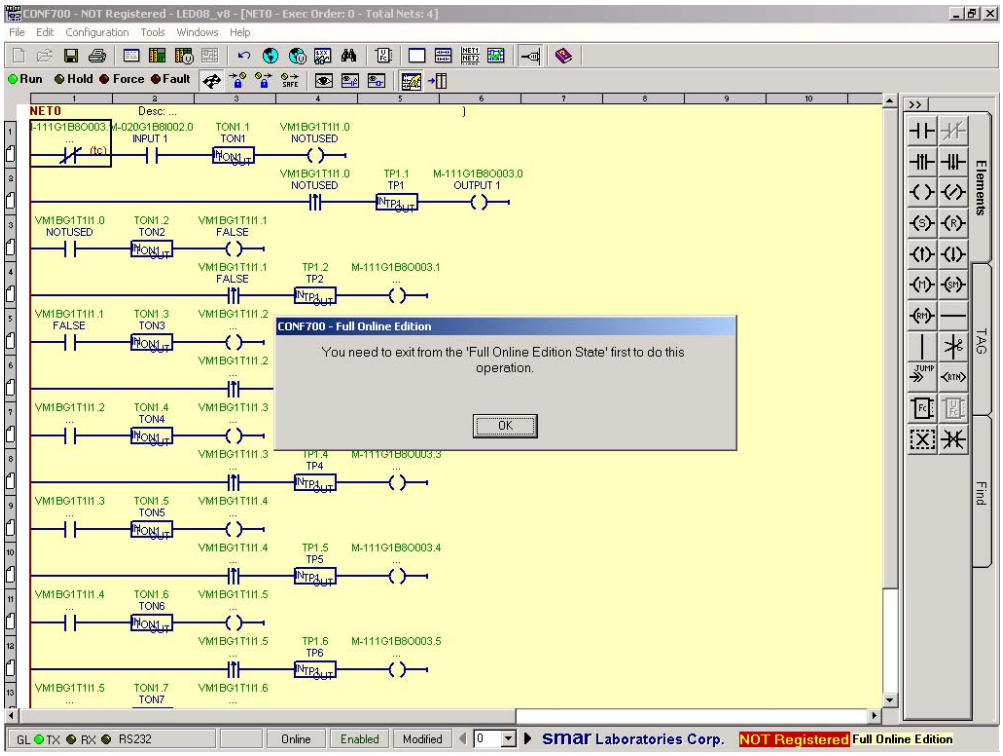


Figure 3.95- Monitoring Stopped

Add/Change Ladder Elements

Ladder elements, such as relays and coils, can be added, changed or removed. When an element is added, one (N) New indication appears at the side of this element. See the following picture:

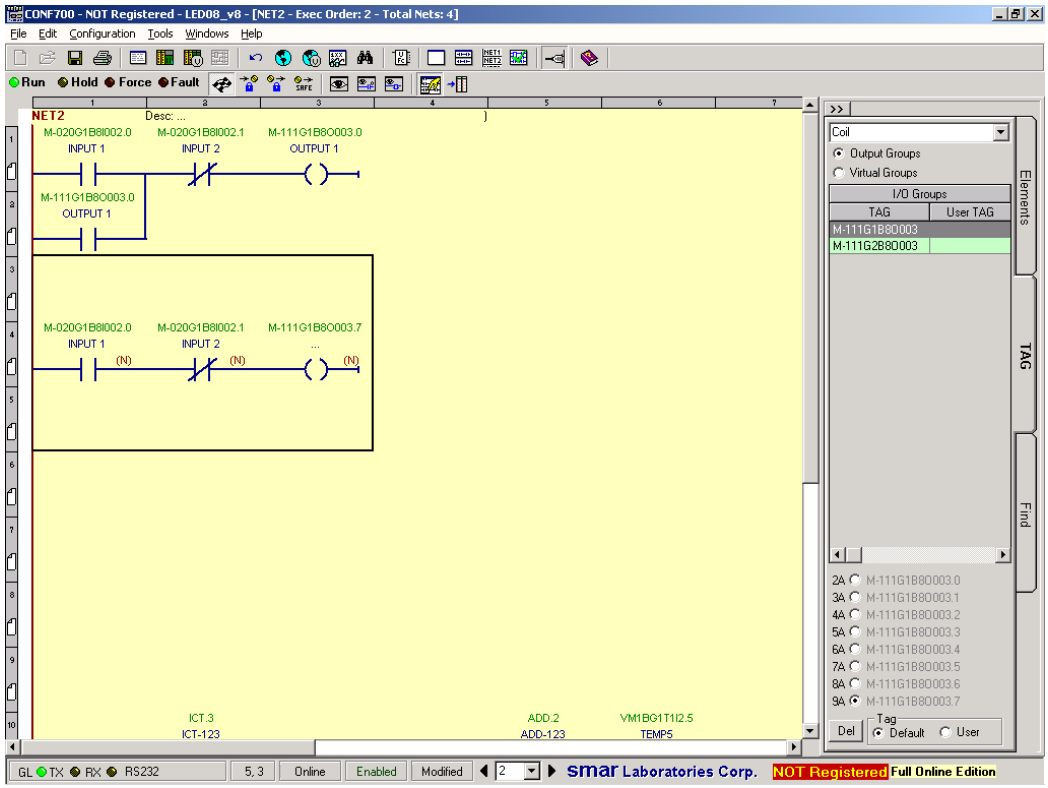


Figure 3.96- Add New Elements

When an element is replaced or modified, the indication tc (temporary change) appears.

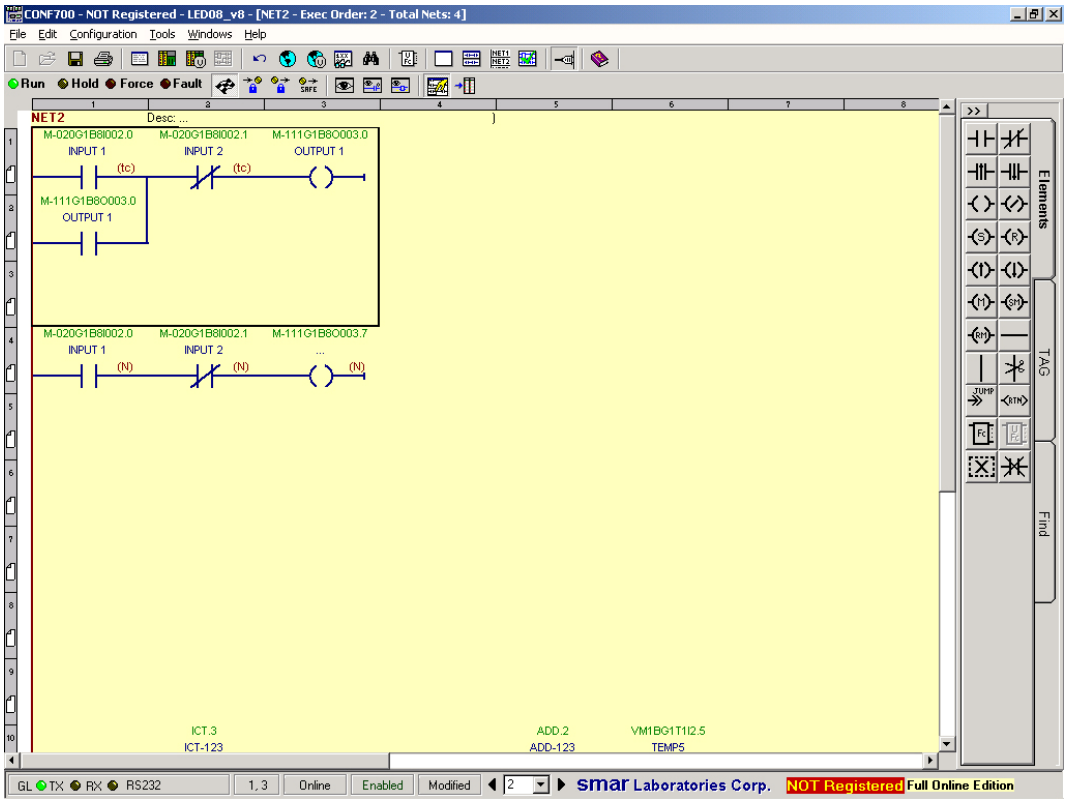


Figure 3.97- Replace Elements

Add/Remove Networks

With this option, the user can add or remove Networks on the online mode. In order to make it, click on Add Network  button. A New Network will be created. To visualize the previous Network and the New, click on the Tile Horizontally  button. The following picture shows two Networks in the same window.

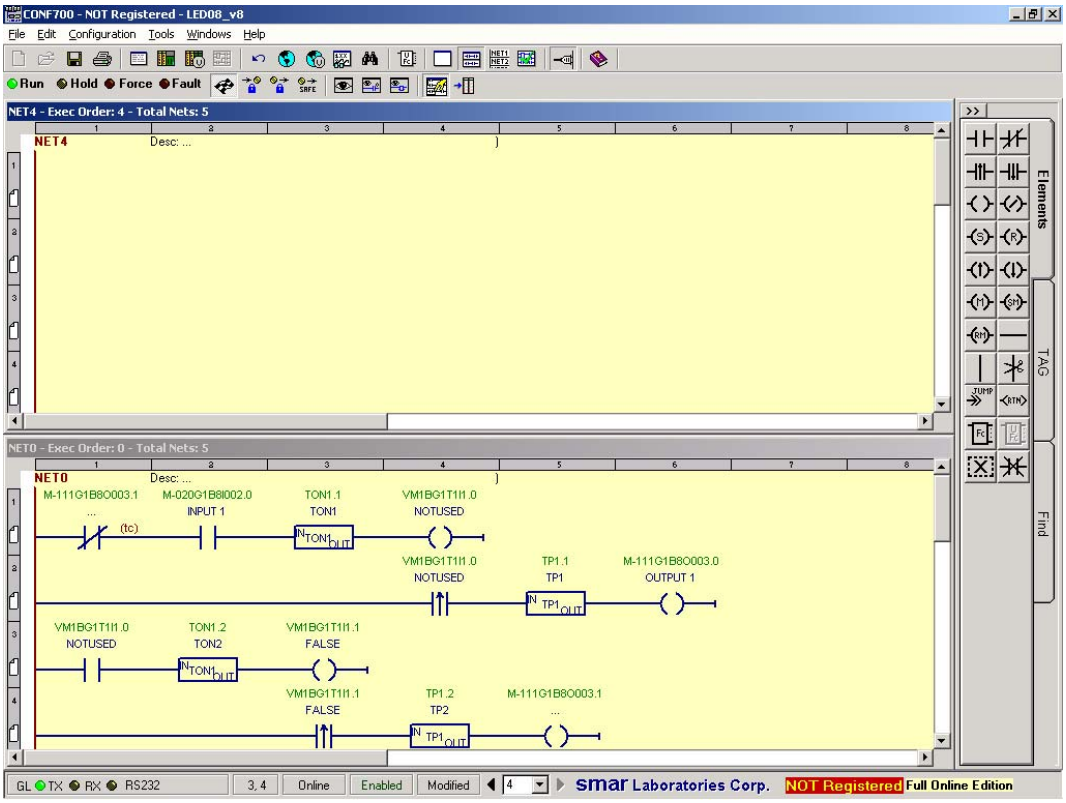


Figure 3.98- Add Networks

If it is necessary to remove one Ladder Network, click on Network Management  button, the picture below will open. The user can delete the desired network clicking desired Network , and then ,clicking on the Delete button.

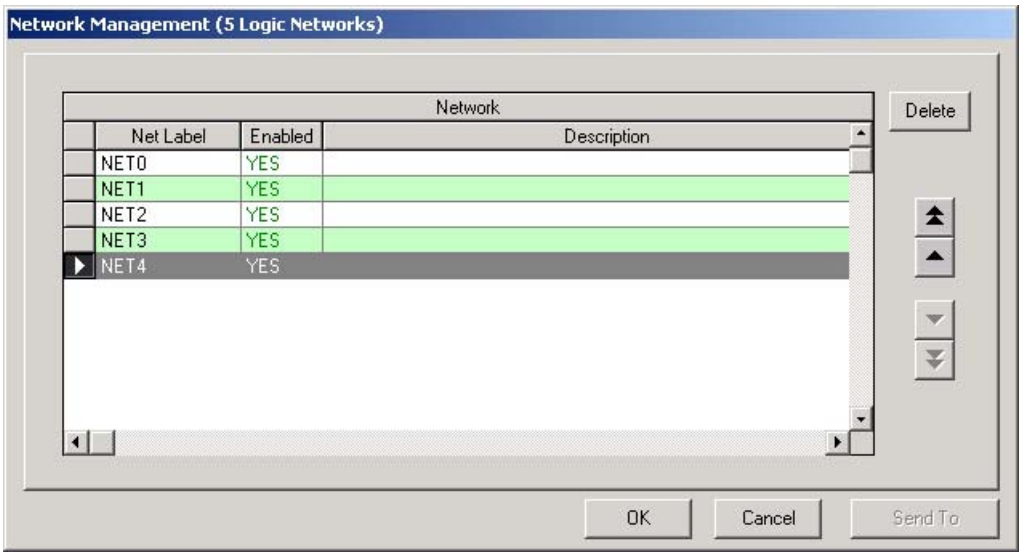


Figure 3.99- Delete Networks on the Full Online Edition

Add/Remove Modules

In the Full Online Edition mode, the user can add, delete or replace Hardware modules in the Hardware Page.

When a new module is added, the indication New appears, as in the picture below.

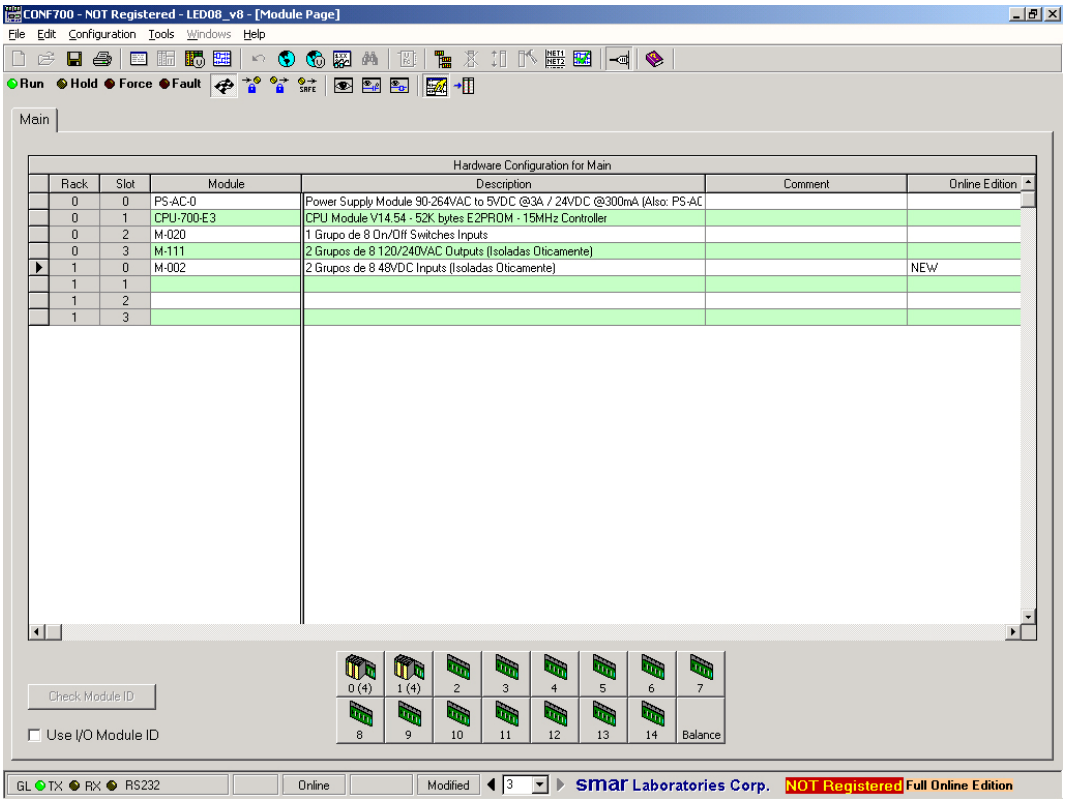


Figure 3.100- Add Modules in the Full Online Edition

When an existent module is deleted, the indication Deleted appears.

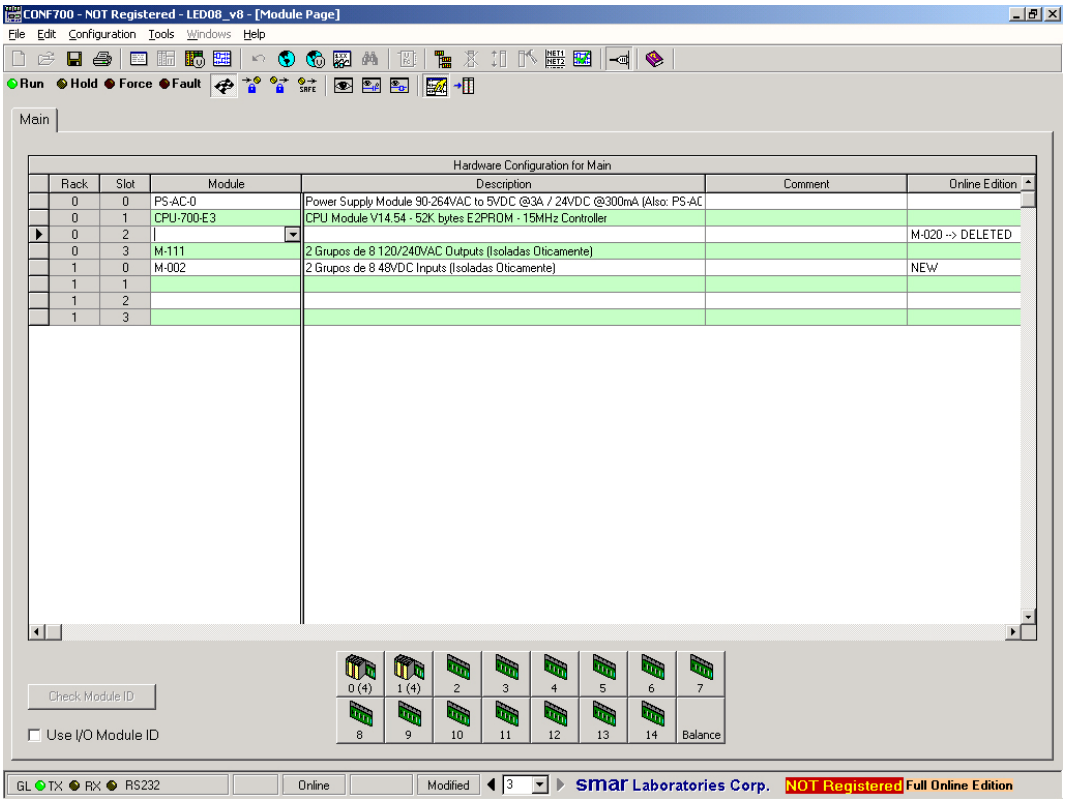


Figure 3.101- Delete Modules on the Full Online Edition

When a module is replaced, the indication Replaced appears.

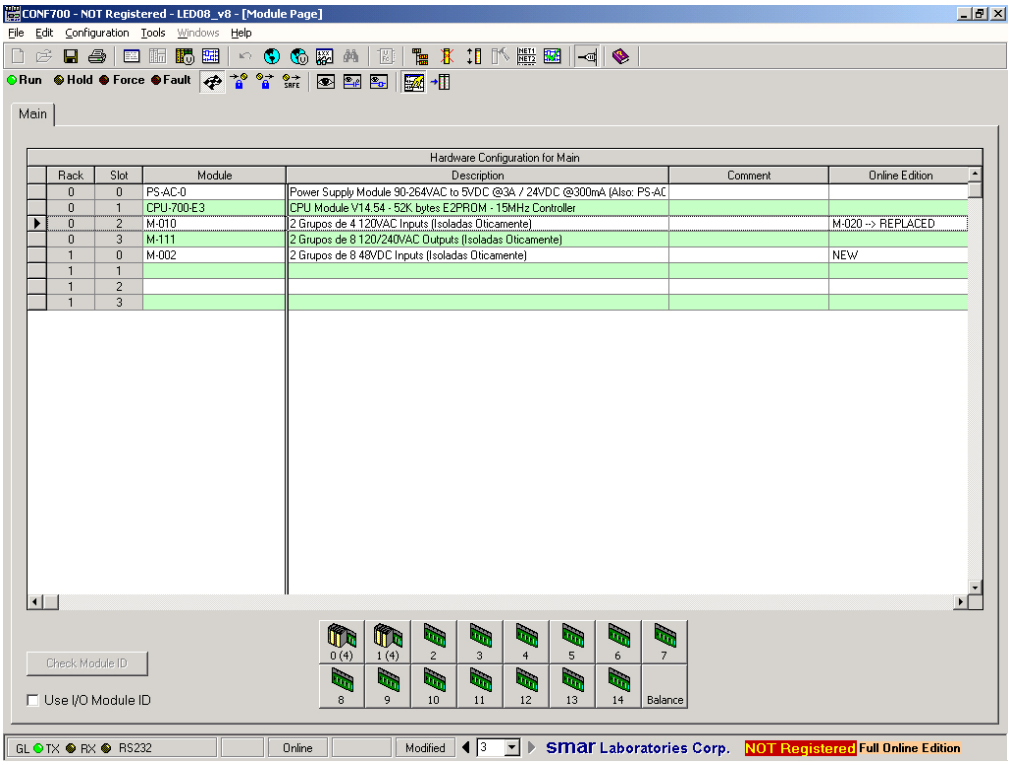


Figure 3.102- Replace Networks on the Full Online Edition

Add/Remove Virtual Modules

Virtual Modules can be added or removed in this mode. In order to make it, proceed the same as in the Offline mode.

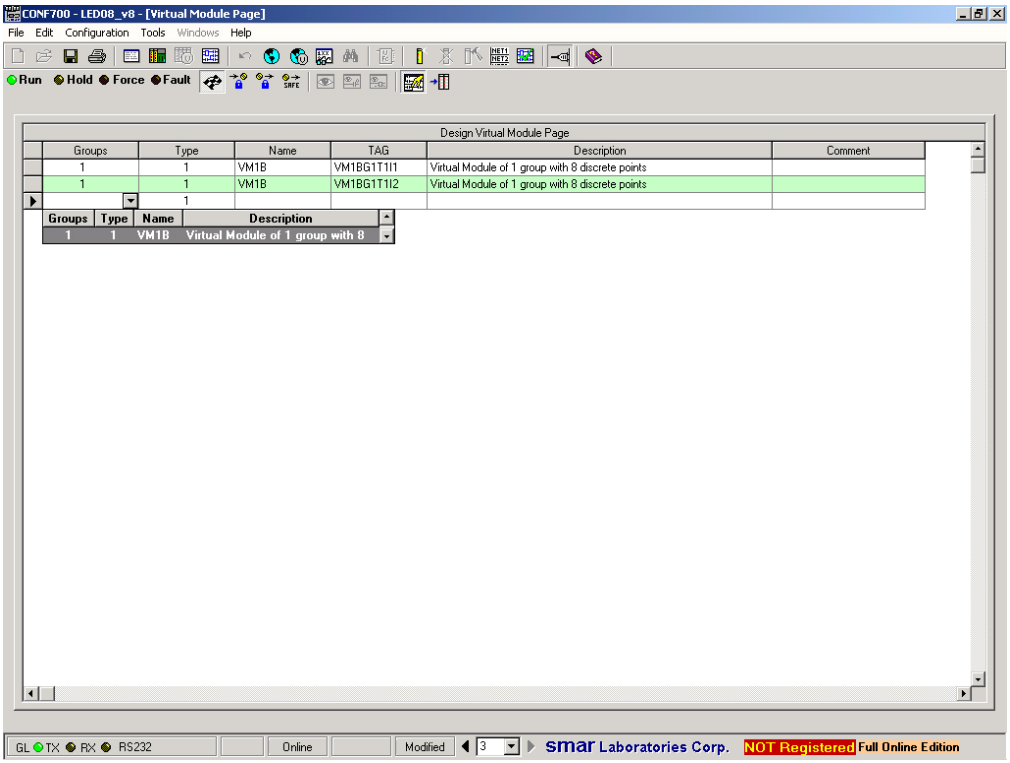


Figure 3.103- Add Virtual Module on the Full Online Edition

Add/Remove RIO Interface

The user can add or remove RIO Interfaces in this mode.

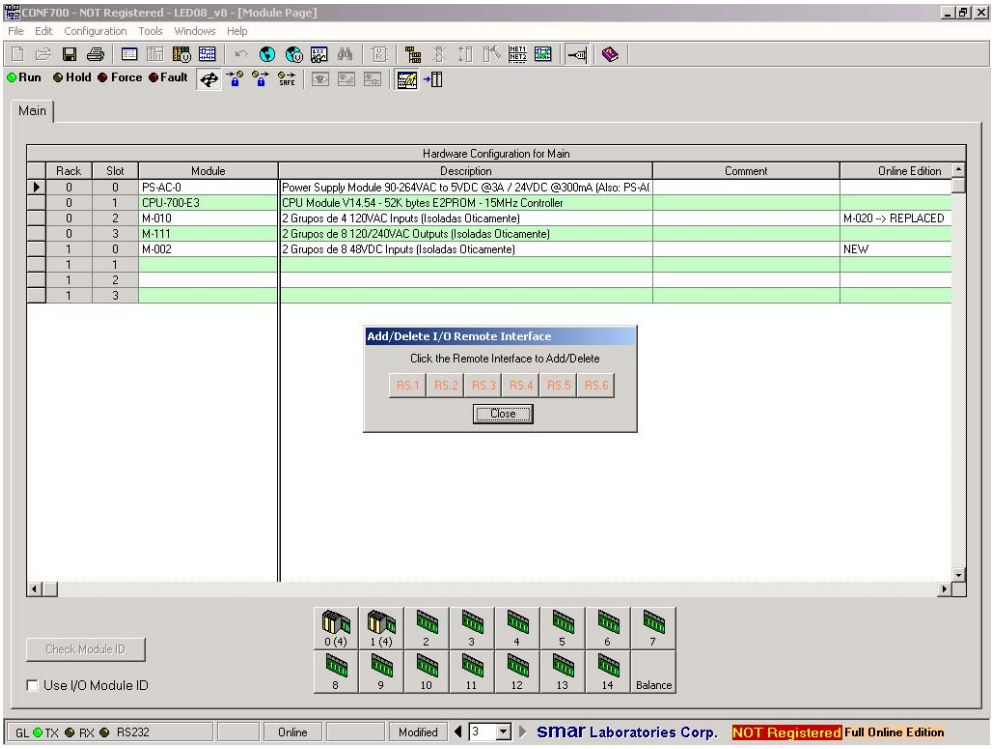


Figure 3.104- Add RIO Interface

Add/Remove User Functions

User Functions can be added or removed from Ladder.

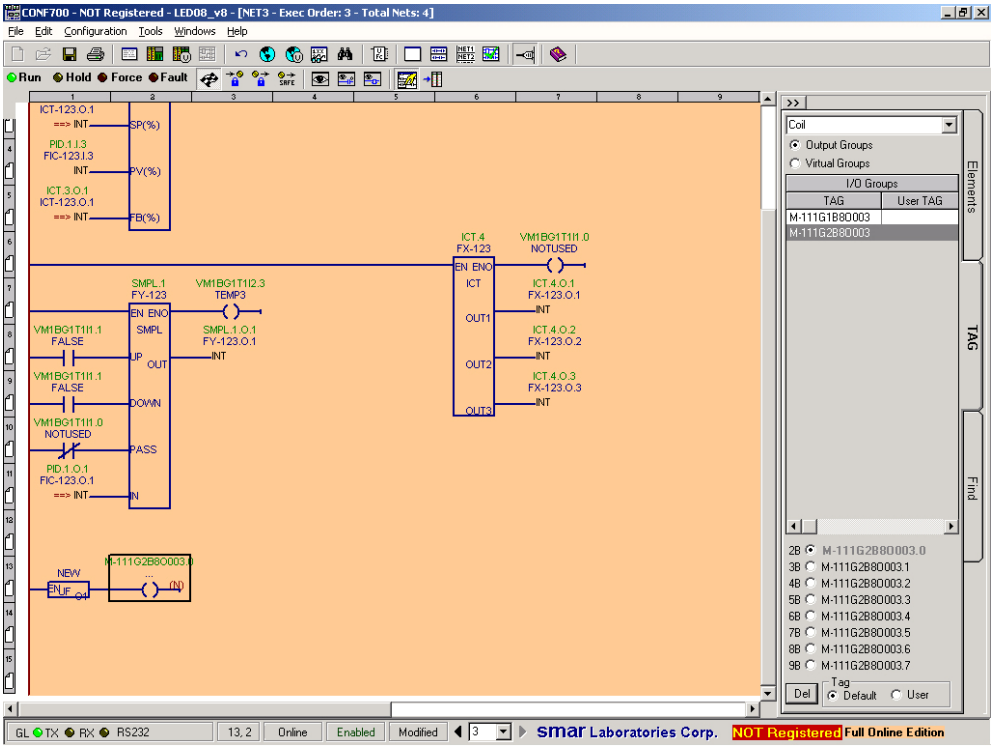


Figure 3.105- Add User Function

NOTE
It is important to remember; only one instance can be done for each User Function.

Change Module Configuration

In the Full Online Edition, it is possible to change the hardware module configuration in the Page Module. After the hardware module configuration changes, the Note “Modified” will be displayed in the field “Online Edition” for the changed module, according to the picture below.

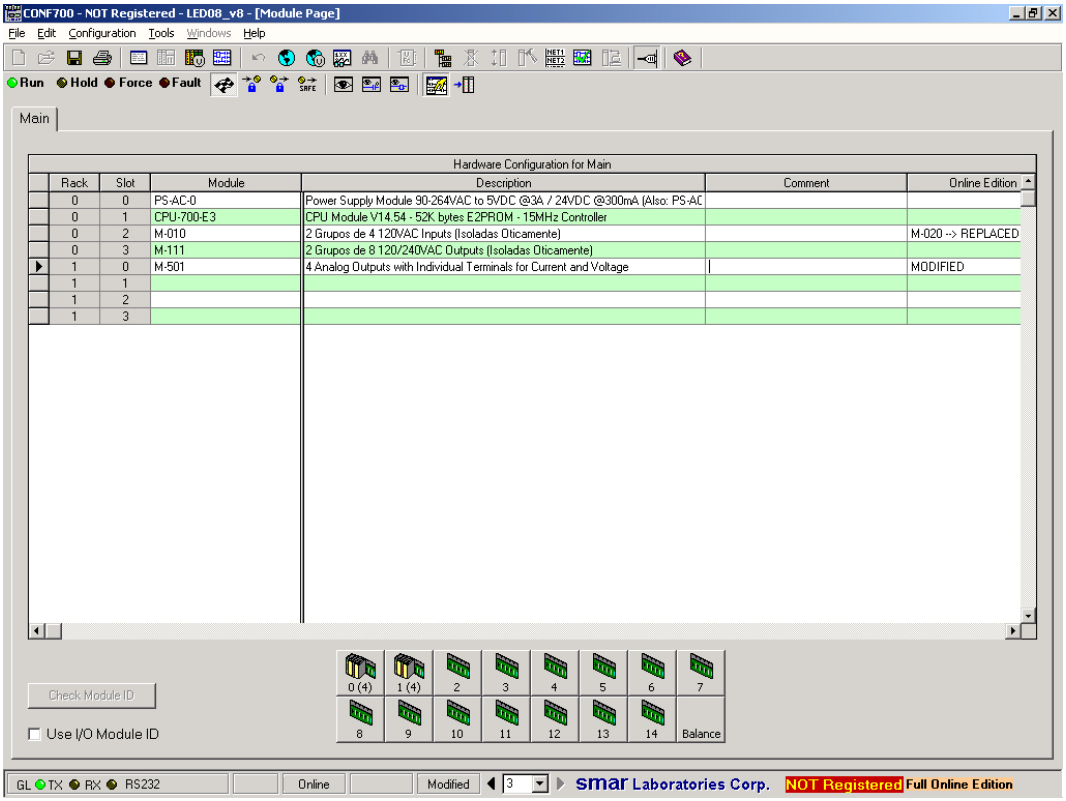


Figure 3.106- Change Hardware Modules in the Module Page

Move Modules in the Module Page

With Full Online Edition, the user can move modules at any Main or RIO systems.
After the module is moved, the Note “MOVED FROM <SRS Source> TO <SRS Destiny>” appears, as it can showed.
SRS – System Rack Slot
System: 0- Main
1 to 6 – Remote Interfaces

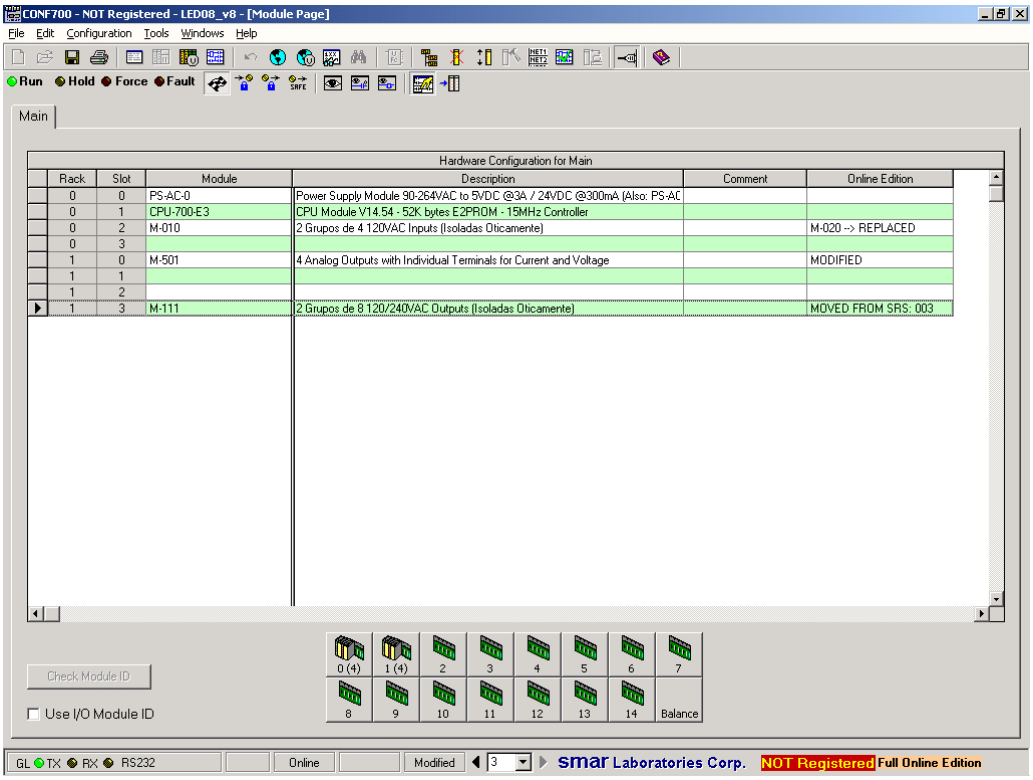


Figure 3.107- Module moved in the Module Page

Update in the Full Online Edition

After the desired updates, it is possible to update the LC700. In order to make it, on the toolbar, select Tool and Full Online Edition → Send, or click on the button  .

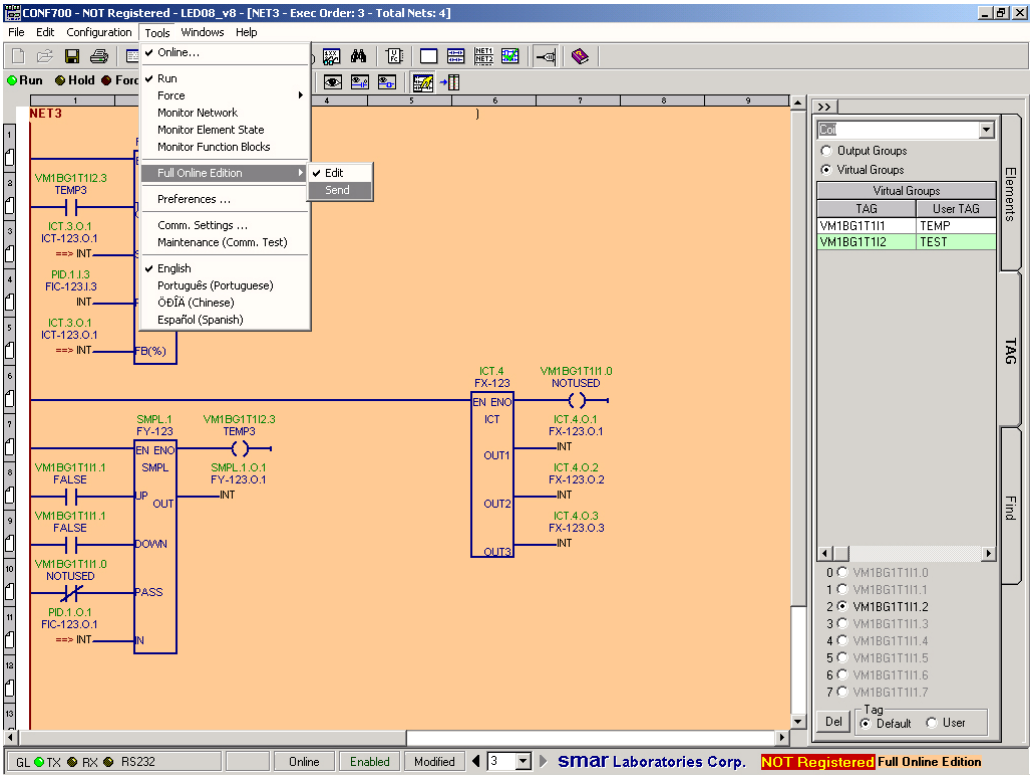


Figure 3.108- Change update on the Full Online Edition

The following window will be displayed. It shows the CPU actual data and the options Update and Close.

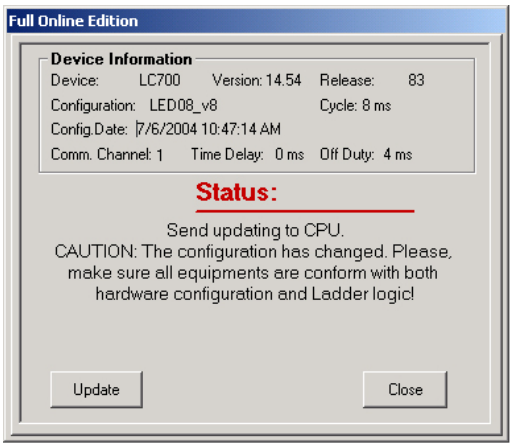


Figure 3.109- Configuration Update

If the option Update was selected, the changed configuration will be updated in the CPU. The progress indication is showed, according to the next screens.

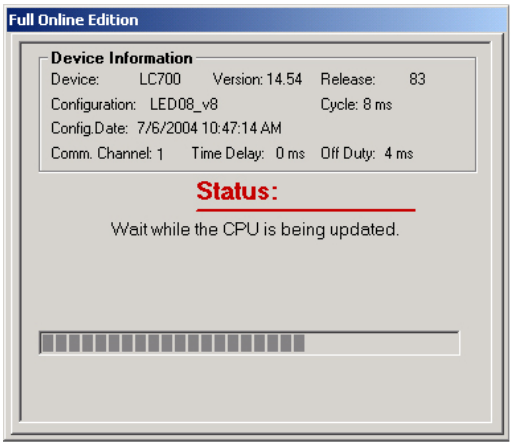


Figure 3.110- Update in progress

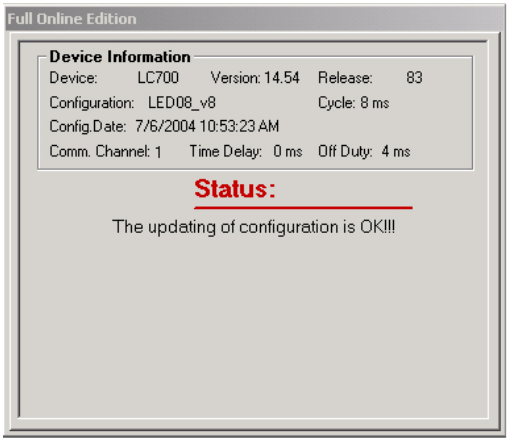


Figure 3.111- Configuration update concluded

NOTE
After the update is finished successfully, CONF700 shows the parameter values ConfigDate and the new Configuration cycle.

If there was a communication or control transfer failure to the new configuration, the LC700 will continue executing the control referring the original configuration and the CONF700 will display the following message.

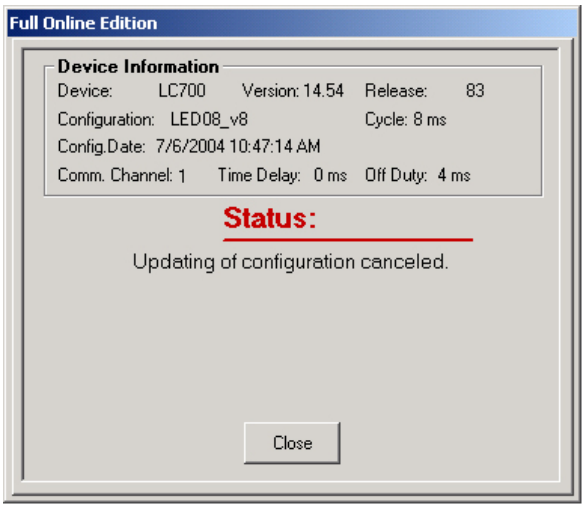


Figure 3.112- Canceled Update

System Test after the update

After a successful update CONF700 stays in “test” state. In this state, the monitoring is enabled again and the user can observe the system conditions with the updates, thus, the user can choose one of the following options:
Confirm the changes done. Thus, the system executes the control with the new configuration and removes the previous configuration.
Cancel the changes done, keeping the actual values of dynamic variables. In this case, the system returns to execute the previous configuration and delete the new configuration. The existent dynamic variables of the previous configuration are updated with the actual values.

To confirm the changes, it must select in the toolbar the Full Online Edition →Accept Changes, or click on the button  .

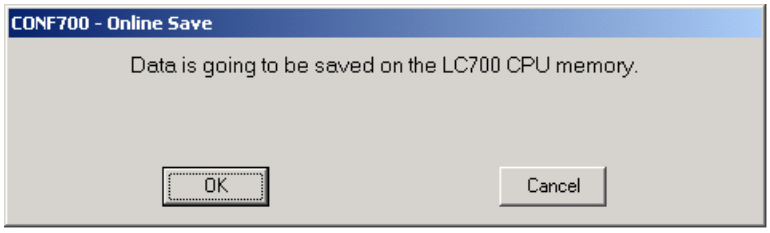


Figure 3.113- Online Save

To cancel the changes, it must selected in the toolbar Full Online Edition → Remove All, or click on the button  .

If the user does not do any of the options above and goes to the Offline mode, CPU-700 will accept changes after 5 minutes automatically.

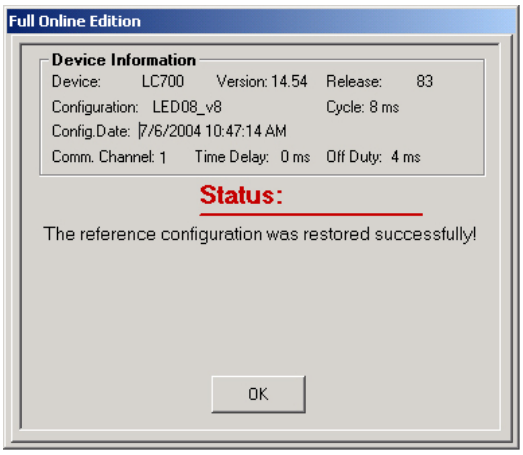


Figure 3.114- Remove the Changes in the Configuration

Differential Download

The Differential Download is a way to update the LC700 configuration, whose edition was done in the offline mode.

If the user needs to edit an offline configuration in a workstation far from the process, it will use the "Using Base Configuration" option. In order to make it, it is necessary to edit the same configuration sent to CPU-700, do the changes and update the LC700 that executes the process control.

An example will be displayed of an edit configuration that uses this option.

1º Step

The user must guarantee the edit configuration is the same that is executed in the CPU. Thus, the CPU must not have any change in its configuration.

NOTE
If the configuration executed in the CPU-700 is not the same edited by the user in the workstation far from the process, it is impossible to send the configuration to the CPU-700 unless interrupting the process.

Condition Table

The configuration file posses two data**:**
1 - Save Data (DataSave): data of the last save.
2 - Reference Data (DataRef): data that will be sent to CPU-700 and also will be referenced to allow the differential download.
DateOfLC700: configuration data that is in the CPU-700.

Rules:

Action Type	Option	Algorithm

Normal Download	Only Download	IF FILE_MODIFIED THEN
		DateRef ← Now
		ELSE
		DateRef←DateSave
		END

Normal Download	Save and Download	DateRef← Now
Upload		DateRef←DateOfLC700
Save or Save as		IF OFFLINE AND FILE_MODIFIED_AFTER_DOWNLOAD AND NOT_USING_BASE_CONFIGURATION THEN DateRef ← NULL END
Update or Differential Download		DateRef ← Now

NOTE
If the field DateRef = Null indicates the configuration changed since the last download for the CPU-700, thus it will not be allowed to enable the Using Base Configuration option.

2º Step

By the toolbar, select Tool → Using Base Configuration.

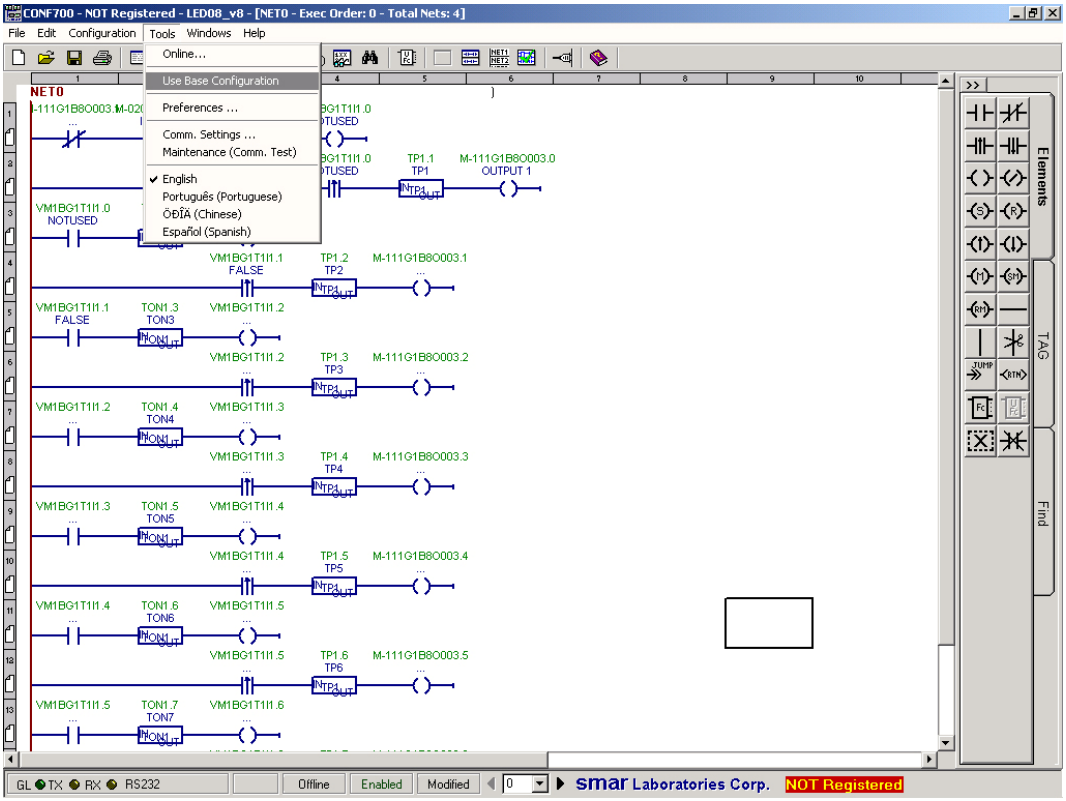


Figure 3.115- Set the Using Base Configuration option

At the lower right corner indicates the Base Configuration is being used.

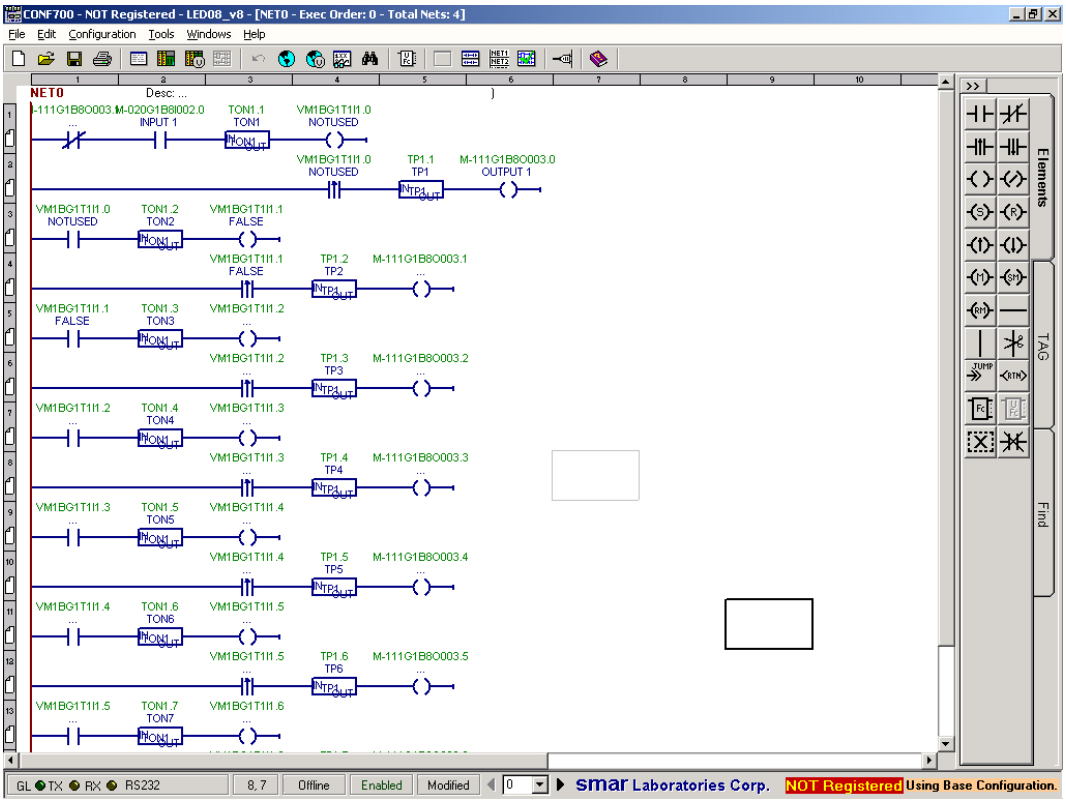


Figure 3.116- Using Base Configuration set

3º Step:

Do the needed changes in the configuration.

4º Step

Case 1: CPU-700 is connected to the PC

After doing the changes in the configuration, click on the Online button. The Differential Download option will be enabled.

Case 2: CPU-700 is not connected to the PC or is desired to send the change late.

In this case, configuration must be saved which will be open when it was doing the Differential Download in a PC connected to the LC700.

NOTE
The Differential Download option only will be enabled if the option DateRef = DateOfLC700, that is, the file reference date is the same to CPU-700.

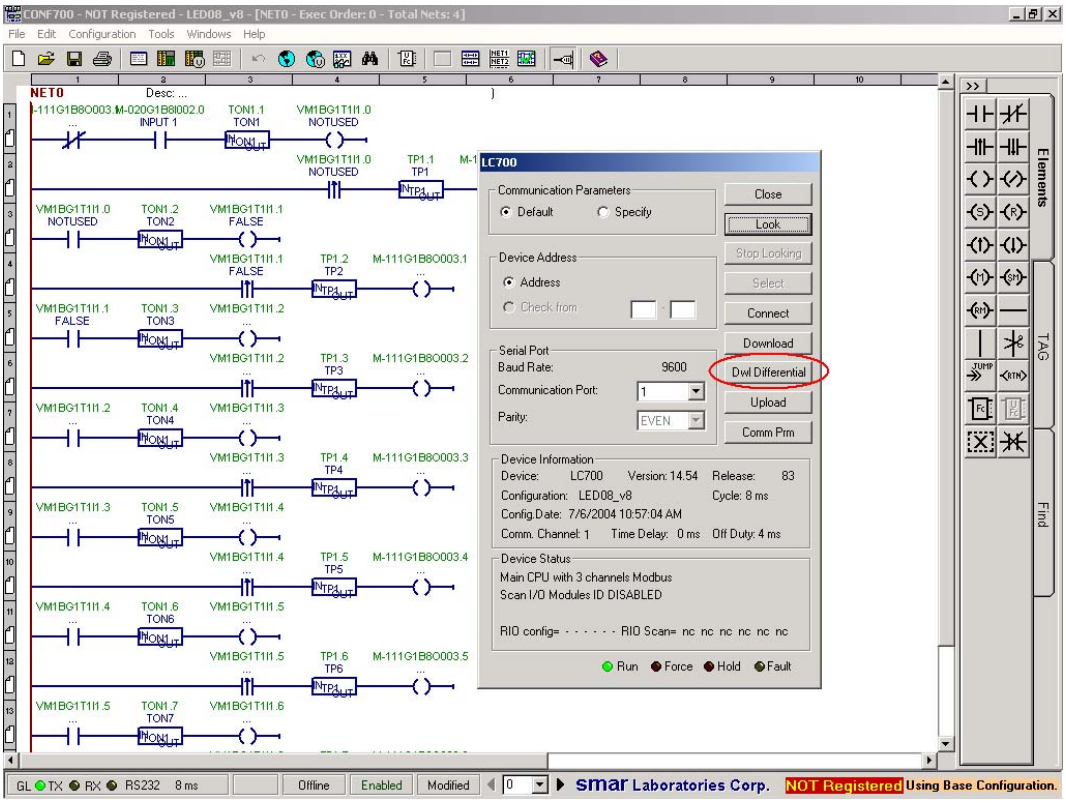


Figure 3.117- Enabled Differential Download

After doing the Differential Download, the changes done will be sent to the CPU.

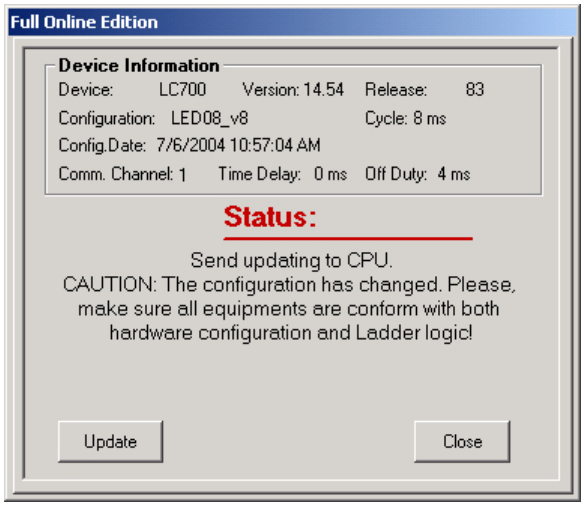


Figure 3.118- Change Update

A CPU update message will be displayed.

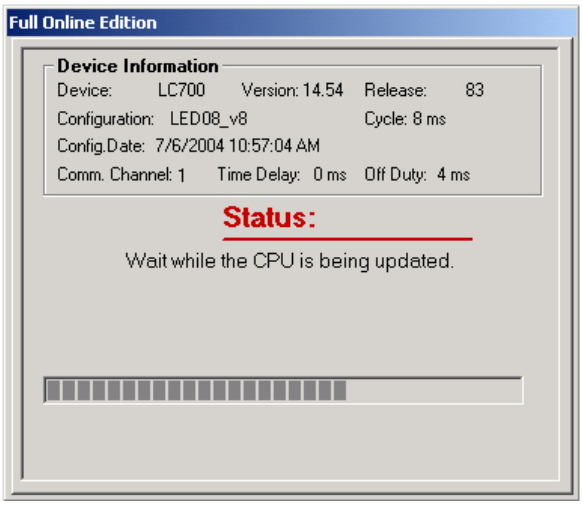


Figure 3.119- Update in progress

CONF700 goes to Full Online Edition with the configuration already updated on the CPU.

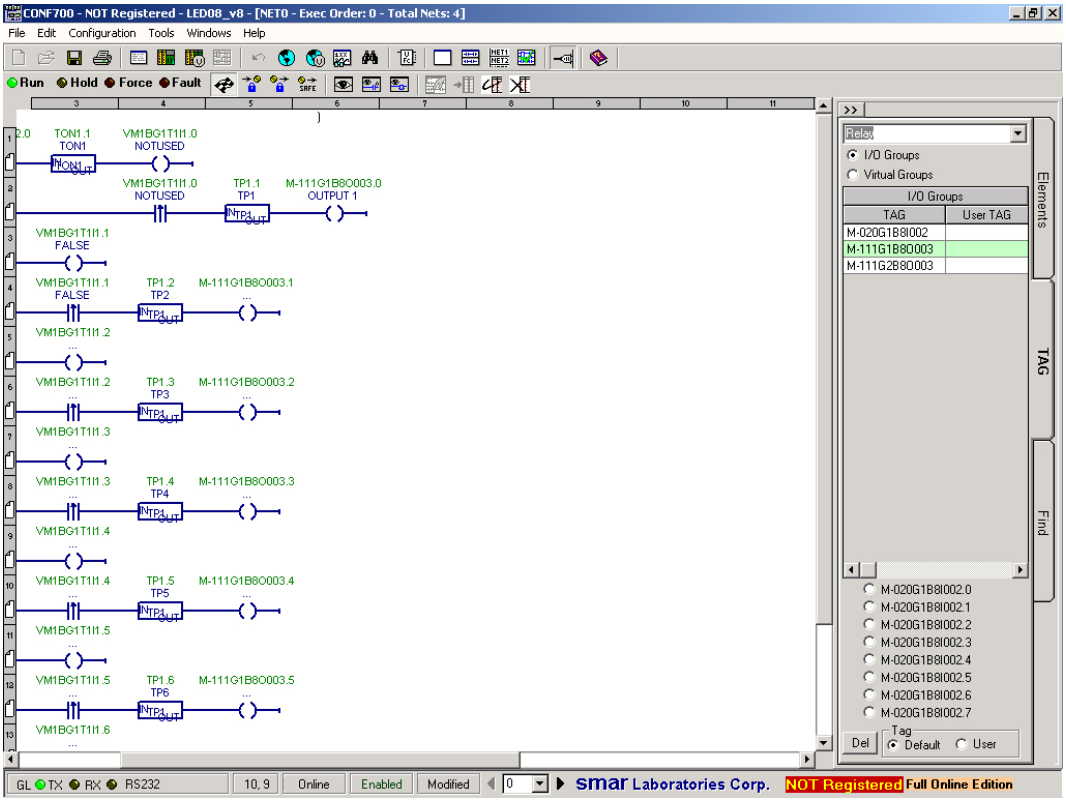


Figure 3.120- Full Online Edition

After the update, CONF700 will be online in test state.

Differences between Online Edition and Full Online Edition

When the user is in Online Edition can only change element tags, block parameters and relays/coils types. With Full Online Edition, all the changes, that it would be necessary to be on offline mode to do, can be done online, easily for the user observation.

In the Online Edition there is the “Undo Change” option, when the user do not want to save any change. In the Full Online Edition, the user has two options Remove All and Undo during the Logic Network edition.

Full Online Edition Advantages

The best advantage of Full Online Edition is that the user can do the changes without interrupting the process, thus the I/O values and function block variables are preserved during the transferring for the changed configuration.

It is possible to edit the configuration in the Offline mode with the option to do the update lately.

The changes can be done with CPU-700 in any state mode: Run, Hold, Freeze or Safe.

Notes

- After the control transference for the new configuration, the digital and analog output modules added (or moved) will be started with the safe values (defined in the Global Table).

- The activation of the Full Online Edition disables the Online Edition automatically, and vice versa.

- When the configuration updates are done by Full Online Edition in redundant systems (CPU-700E3R), the passive CPU follows the active CPU on the configuration interchange, thus there is not synchronism between CPUs. In order to make it is necessary that the Two CPUs do not have the inter CPU cable disconnected during the update process.

- The Input, Output and Function Blocks Modbus Addresses are kept (when the automatic mode to determine the Modbus Addresses is used), except the following situations:
  - Modules moved to other system (from Master to RIO or from RIO to Master or other RIO).
  - FB-700-1S and FB-700 Modules that have added groups and there is not the enough quantity of available and subsequent Modbus Addresses.
  - Function blocks which have the number type changed from integer (INT) to Real (REAL) or ANY_NUM.
  -

- FIFO block can have the Modbus Address changed.

Note for M-402 Module

The M-402 Module has an internal configuration. When an existent M-402 module in the original configuration has its configuration changed in the Full Online Edition, LC700 will interrupt the reading of this module from the receipt of the command which has the new configuration. After the control transference for the new configuration, the module can be read again.

If the changes were cancelled, the M-402 removed or changed, the configuration will be recovered, after returning to the previous configuration.

Note for FB-700 Module

The Fieldbus I/O Modules (FB-700 and FB-700-1S) also have internal configuration. If there are changes in the original configuration, the module reading/writing will not be done while the internal configurations do not match the CPU configuration. Because of this, the reading/writing stop will not occur before or after the control transference for the new configuration, depending on the internal configuration change of the modules was done before or after the LC700 update.

Note for Block View Communication

During the update, using Full Online Edition, the BlockViews which are configured in the LC700 are preserved. However, points that have the Modbus Addresses changed will not be supervised correctly.

Communication Failures

a) Before using the Send button

If the communication between the CONF700 and LC700 is lost, during the edition, LC700 will execute the Number of retries determined on the Communication Settings – Number of retries. If the communication is not restored during this time, the following message will appear.

Figure 3.121- Communication Failure Warning

With the communication restored, the user has two options: click on Retry or Go Offline button.

- If Retry option is chosen, CONF700 returns to online configuration, waiting the user clicks on the Send button.

- If Go Offline is chosen, CONF700 goes to Offline and at the lower right corner indicates “Using Base Configuration”, and it is allowed to do the Differential Download, explained previously.

b) After using the Send button

After clicking on the Send button and if there is communication problems between the CONF700 and LC700, two situations are possible:

b.1) The CPU does not still have the new configuration

The following message will be displayed.

Figure 3.122-Incomplete Acknowledgment Warning

This message indicates there is failure at the sent of the configuration. Click on the OK, the following message will appear.

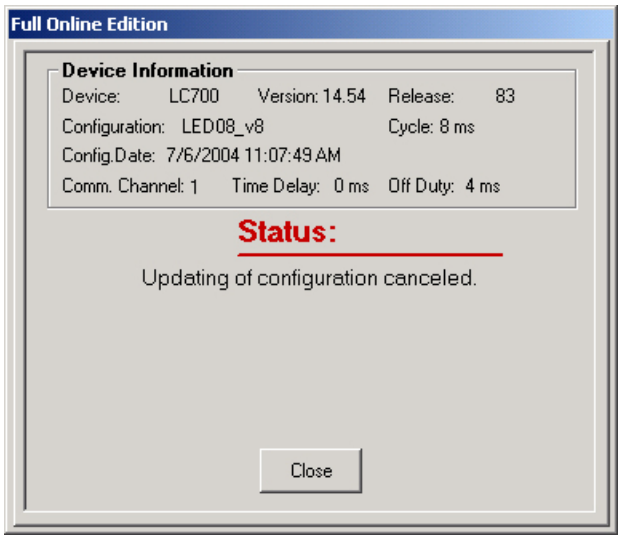


Figure 3.123-Configuration Update Canceling

CONF700 will be in the Full Online Edition, waiting the configuration to be updated in the CPU.

b.2) The CPU already has the new configuration

The following message will be showed.

Figure 3.124- Communication Failure Warning

With the communication restored, the user has two options: click on Retry or Go Offline button.

- If Retry option is chosen, CONF700 returns to online configuration, waiting the user clicks on the Accept Changes or Remove All.
- If Go Offline is chosen, the following message will be displayed to indicate the user will lose the possibility of Remove all changes done in the configuration and the Differential Download option will be disabled. If, in fact, the user chooses this option, the changes done before the communication lost will be saved in the CPU-700.

Figure 3.125- Options Remove All and Download Differential disabled

c) After using the Accept Changes button

When there are communication problems between the CONF700 and LC700, after clicking on Accept Changes, the following message will appear:

Figure 3.126- Communication Failure Warning

CONF700 will not be in the Full Online Edition, but online. Clicking on Retry, with the communication restored between the CONF700 and LC700, it can choose between Online Edition and Full Online Edition.

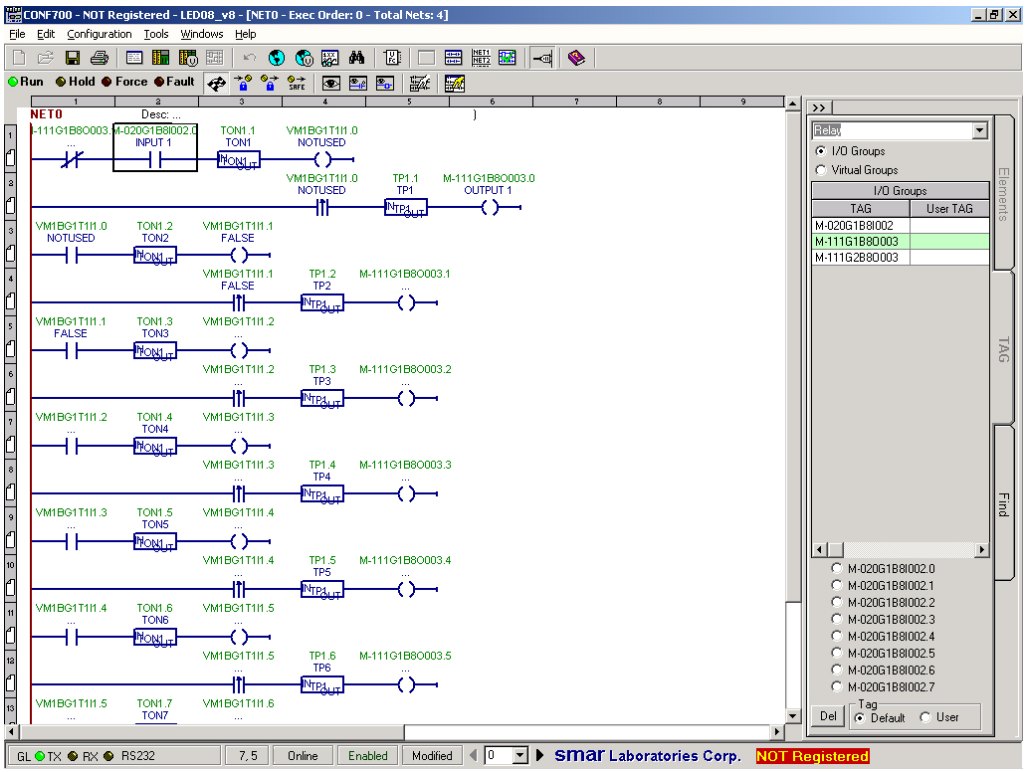


Figure 3.127- Accept Changes

It is important to remember even the communication between the CPU-700 and CONF700 is lost, but the Accept Changes button has already been selected, these changes will be save in the CPU, independently of the chosen option after the communication restore.

Update Desistance in the Full Online Edition

In order to desist of doing the updates in the configuration, the user must click on the toolbar, on the Full Online Edition button to go out from this mode. The following window will be displayed, offering three options to the user:

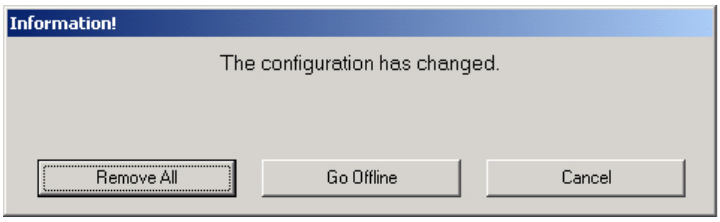


Figure 3.128- Update Desistance

If the Remove All option was chosen, all the changes done in the configuration will be removed. CONF700 will be online and the user can chose between Online Edition and Full Online Edition.

If the user chooses Go Offline, the following message appears.

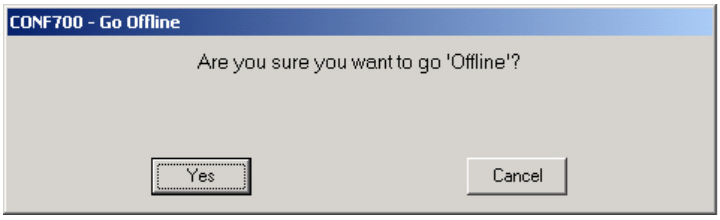


Figure 3.129- Go to Offline mode

If the user chooses Yes, will go out of the Full Online Edition, being Offline, but on “Using Base Configuration”, with the possibility to do the Differential Download without interrupting the process, returning to the Full Online Edition.

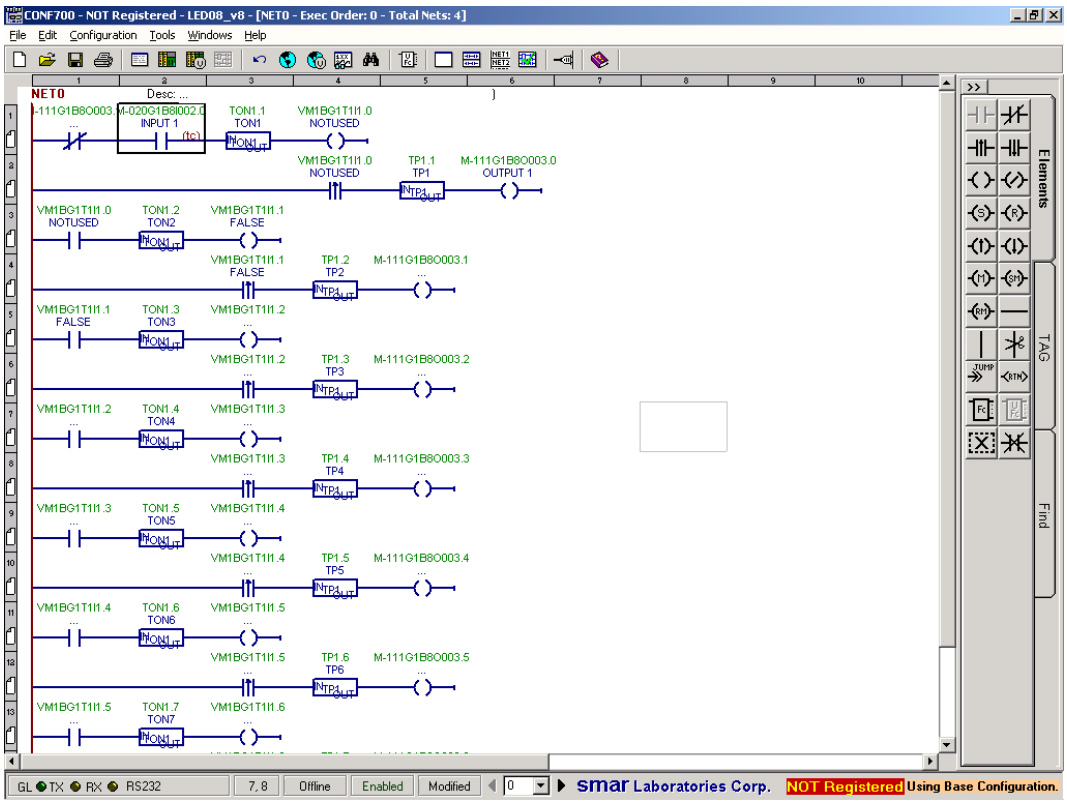


Figure 3.130- CONF700 in the Offline mode

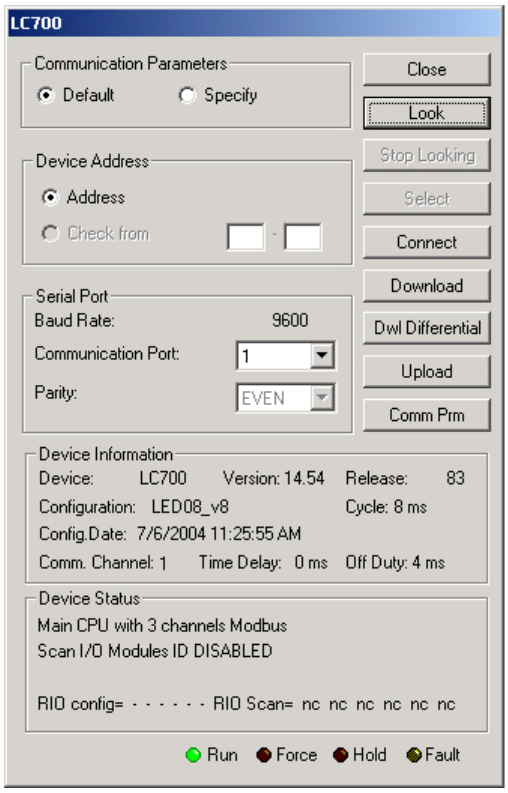


Figure 3.131- Differential Download enabled

NOTE
After the update successfully made, it is advised to save the configuration file, because the automatic save is not executed and the LC700 is with the new configuration.

Example for Full Online Edition

Example 1

From an existent configuration, the user can add modules, network pages and new configurations.
1- Make the configuration download.

2 - Choose the option Full Online Edition.

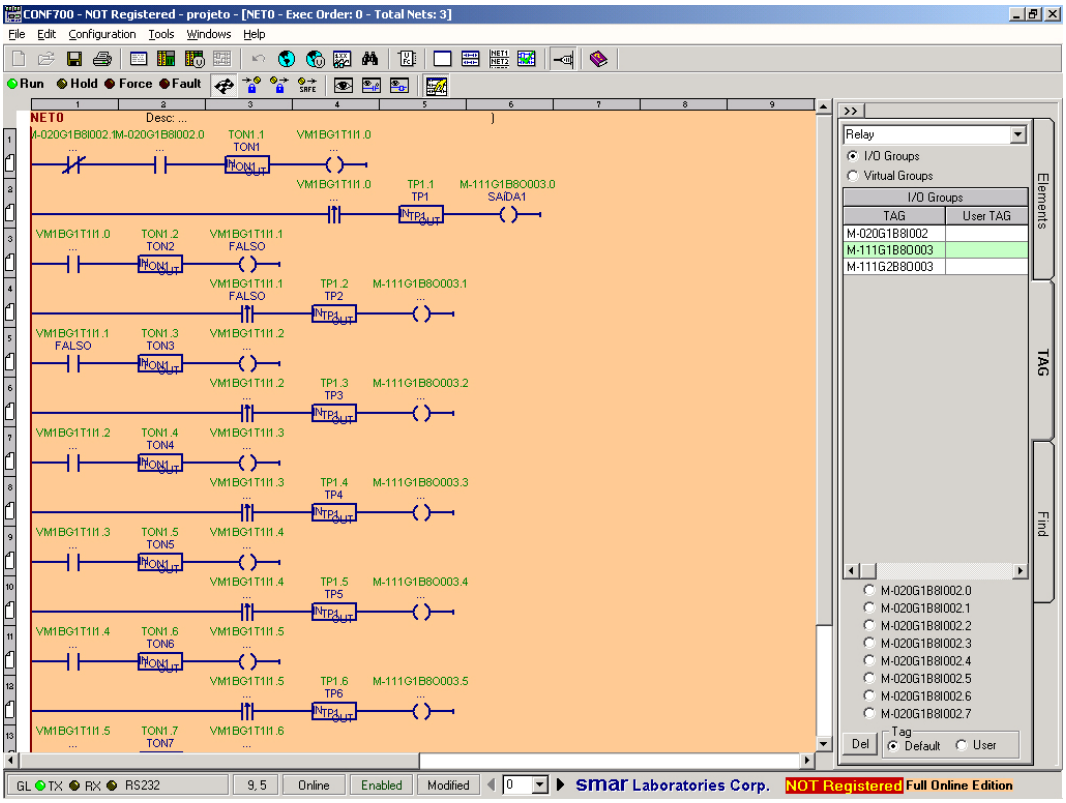


Figure 3.132- Configuration in the Full Online Edition

3 - Go to the Hardware Page. Add a new rack and module.

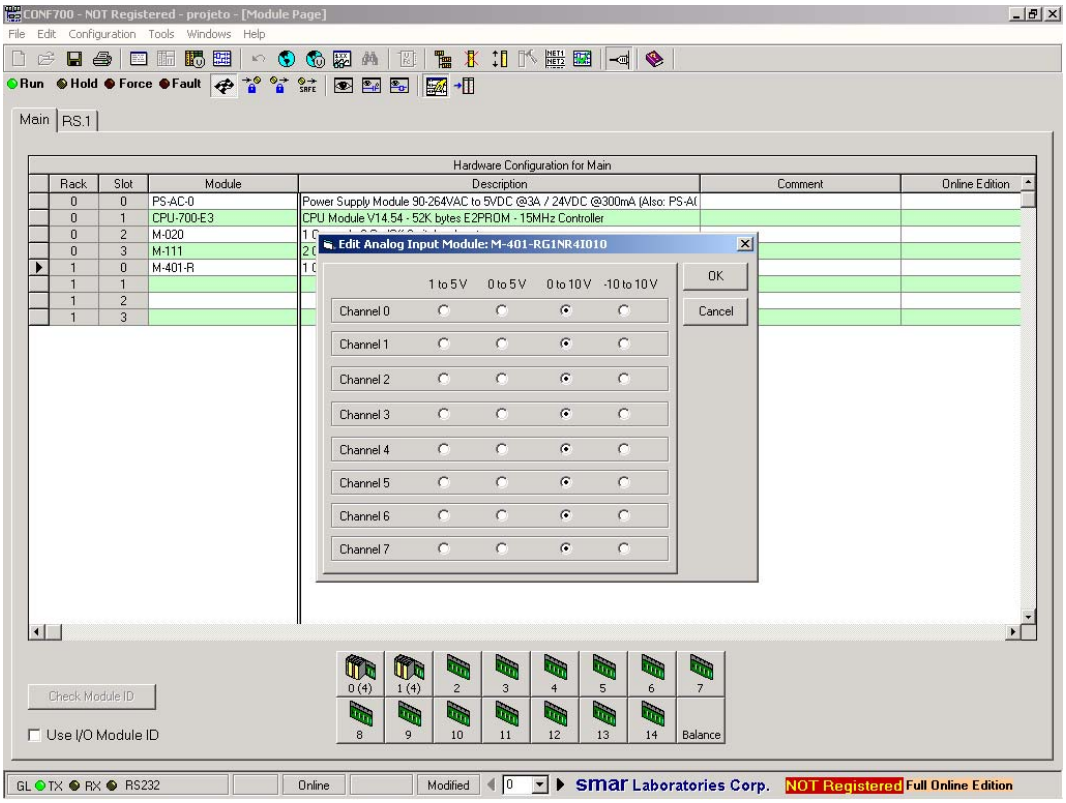


Figure 3.133- Adding new rack and module

4- Add another module and configure the safe values.

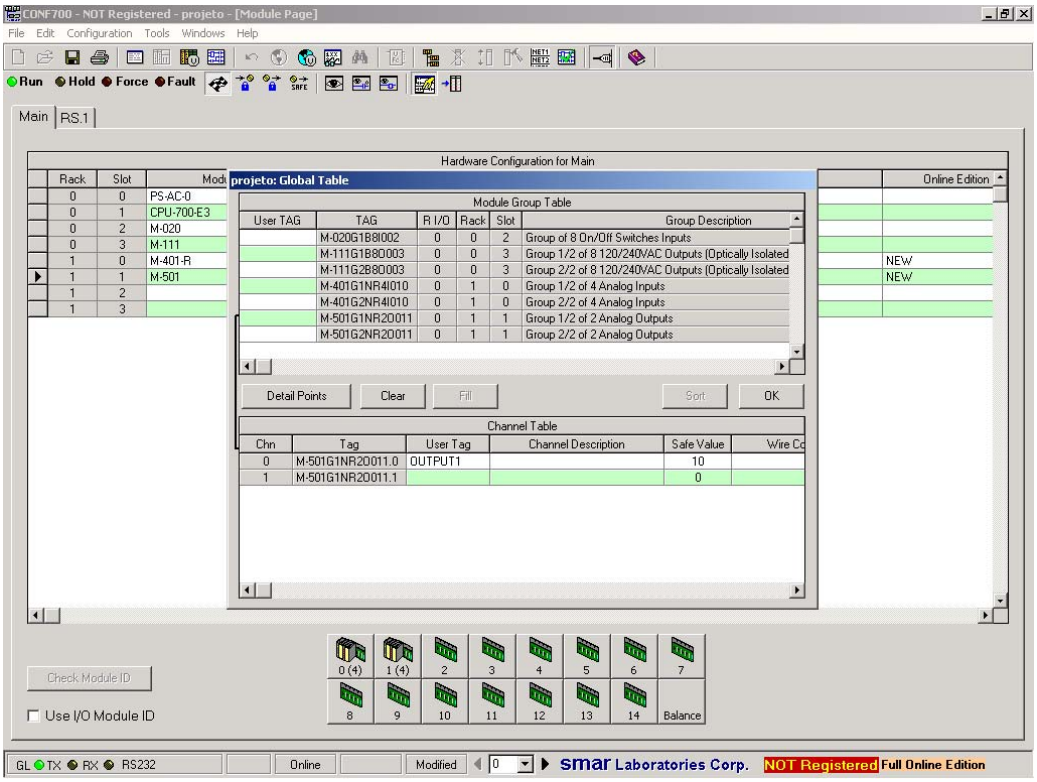


Figure 3.134- Adding the M-501 module

5- Click on the Virtual Module Page and add another module.

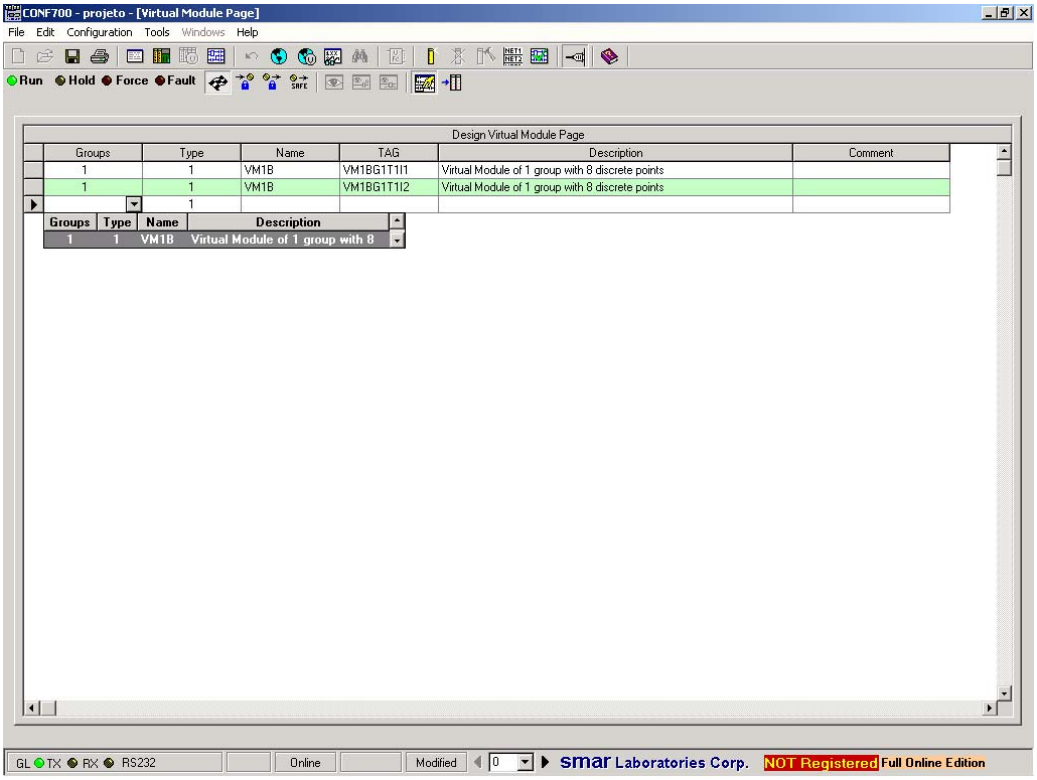


Figure 3.135- Adding a new Virtual module

6- Go to the Network Page and add another Network. Add the new configuration.

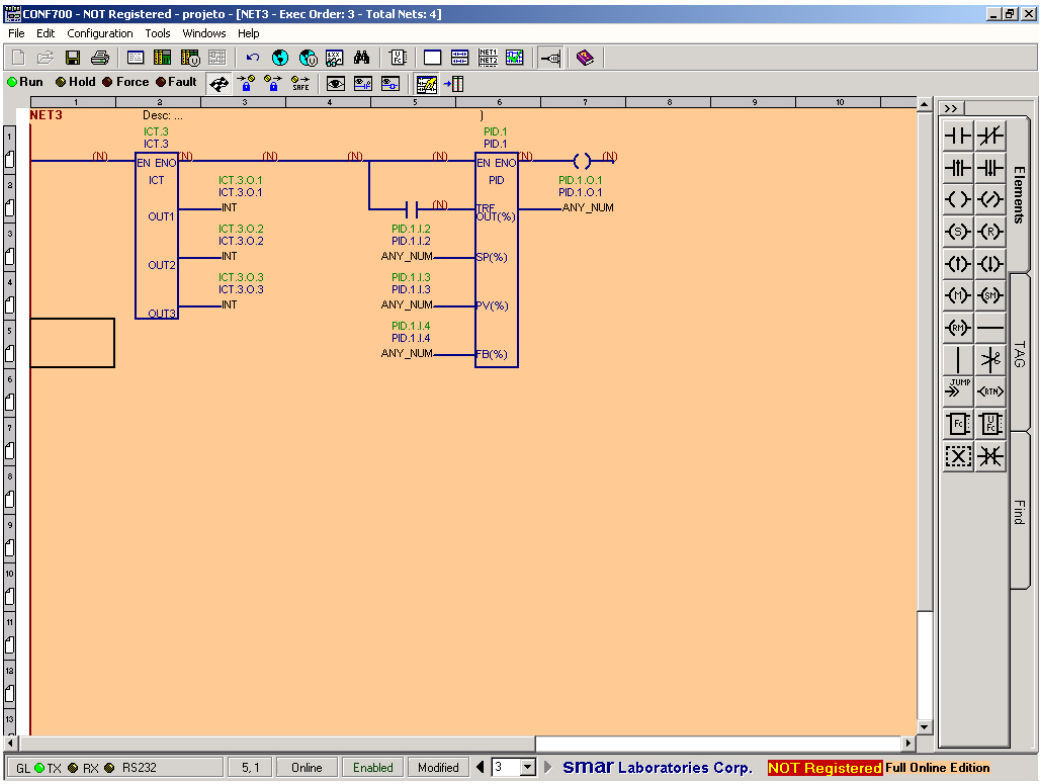


Figure 3.136- Configuration in the Full Online Edition

7- Go to the Hardware Page and add a Temperature Module (M-402).

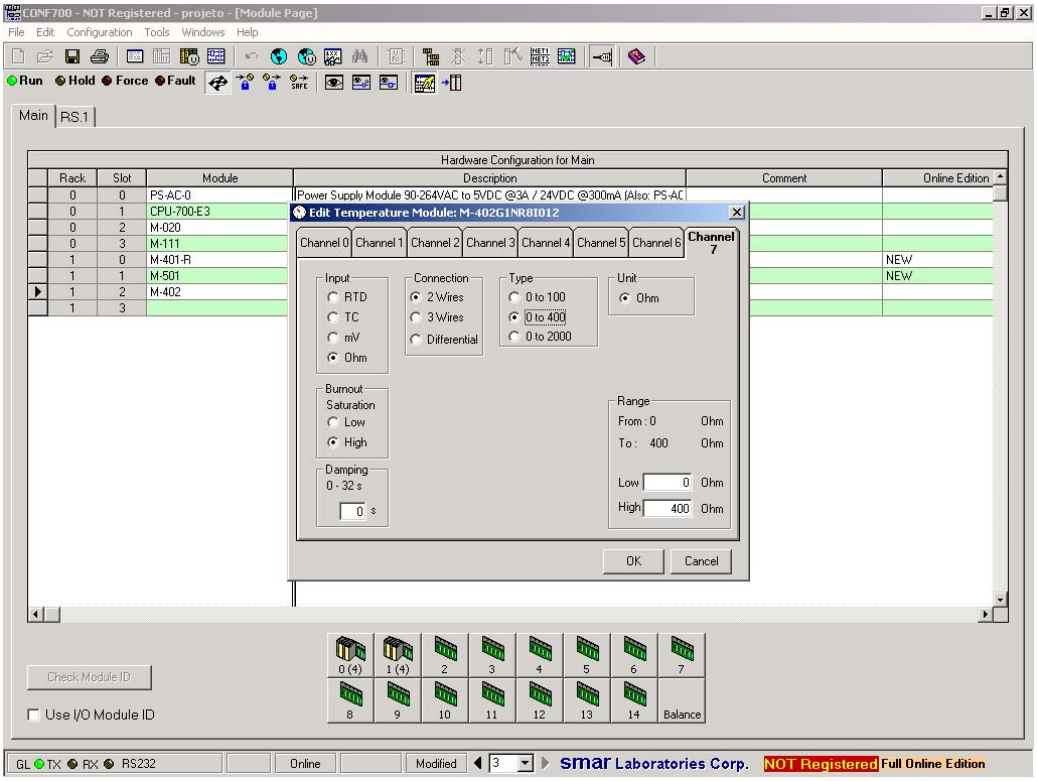


Figure 3.137- Adding the Temperature Module

8 In the Network Page, add another Network. Add the GE function block to make the comparison between the temperatures.

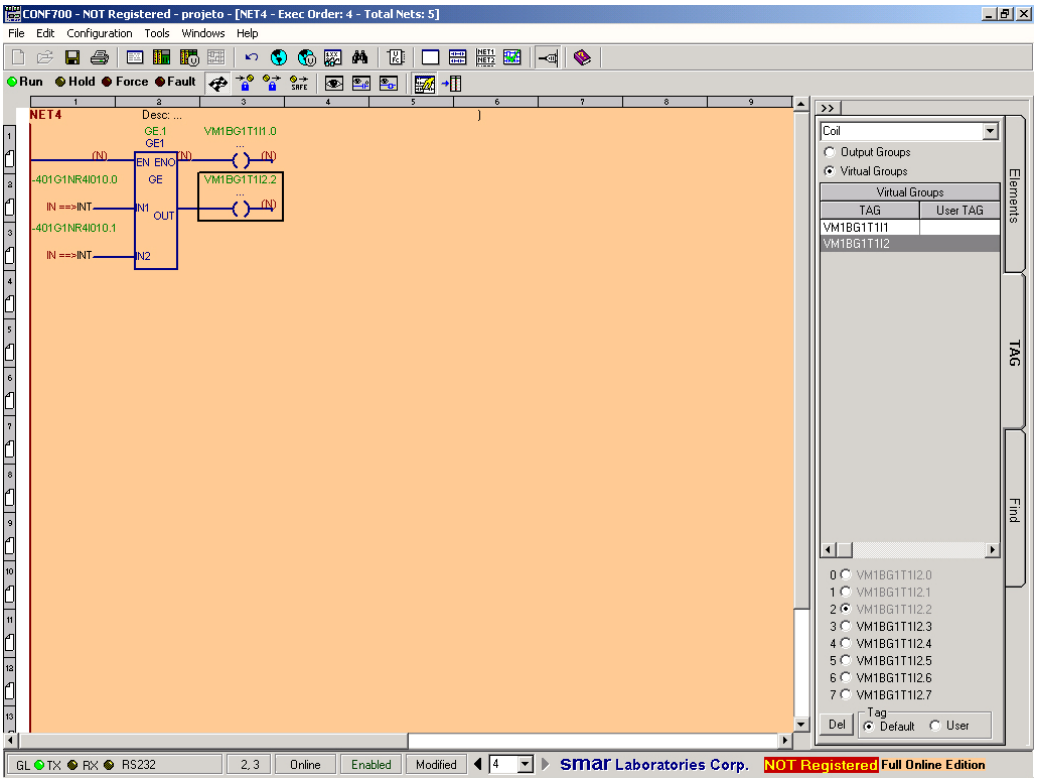


Figure 3.138- New Network with the GE function block

9 Click on the Send button to send the configuration to the LC700.

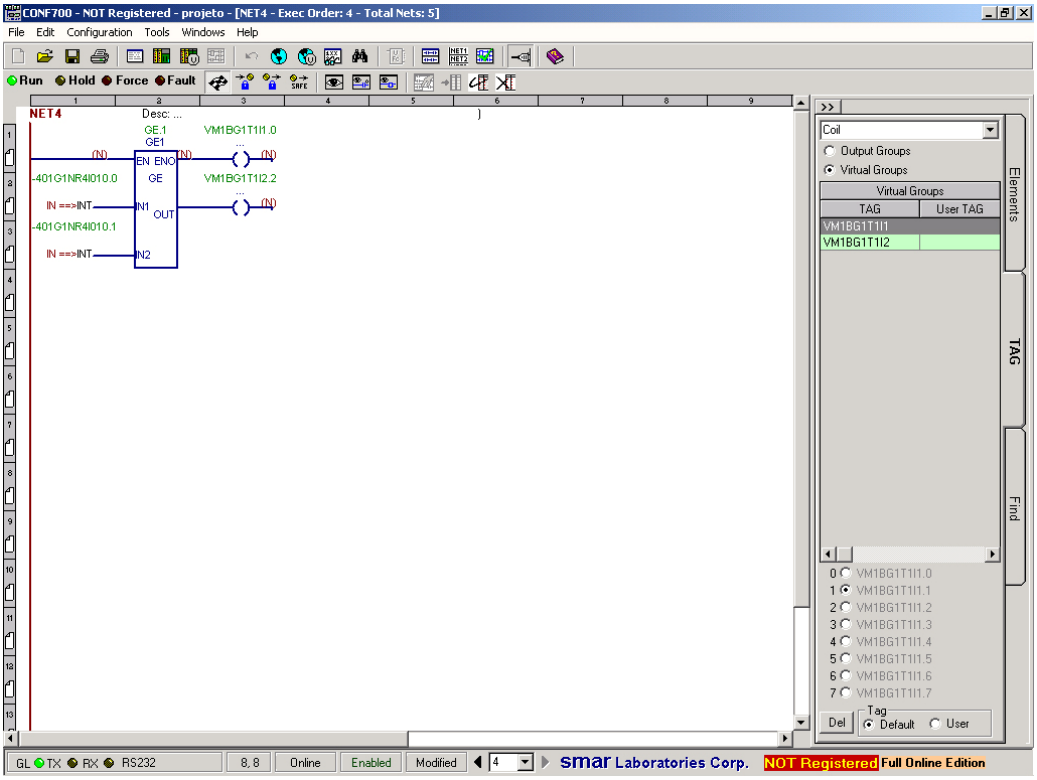


Figure 3.139- After the Send button

b) Example 2

From an existent configuration, it is possible to replace Remote Interfaces.
1 Make the configuration download.
2 Choose the option Full Online Edition

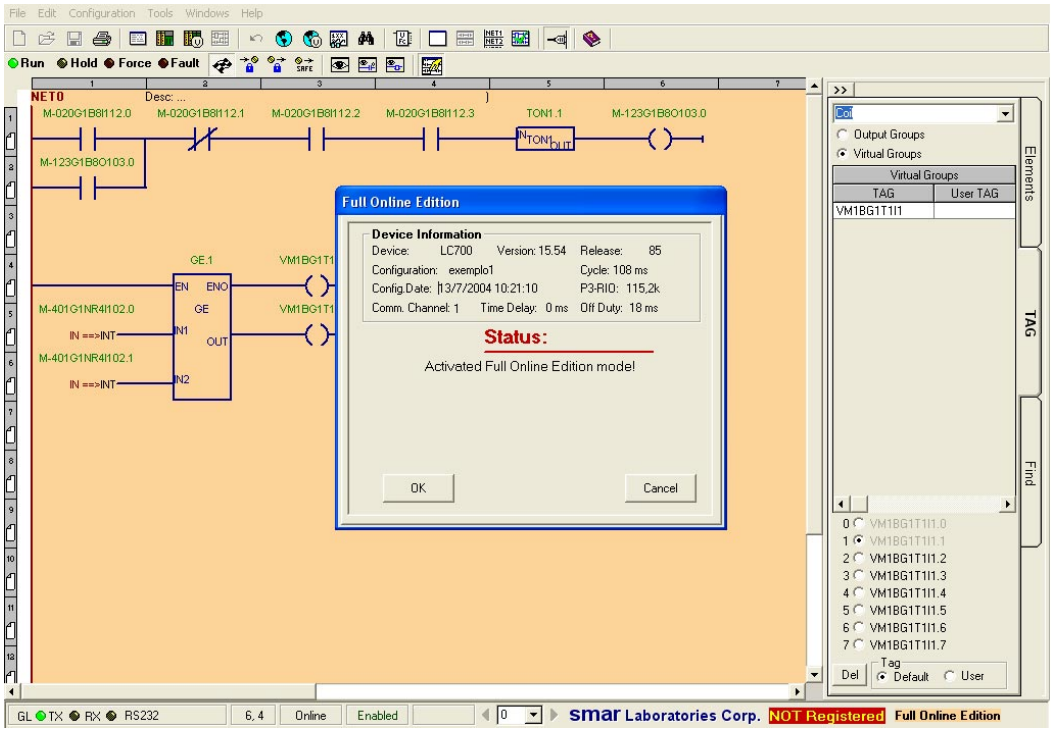


Figure 3.140- Full Online Edition active

3 Go to the Hardware Page. This configuration is composed by CPU Main and Remote Interface.

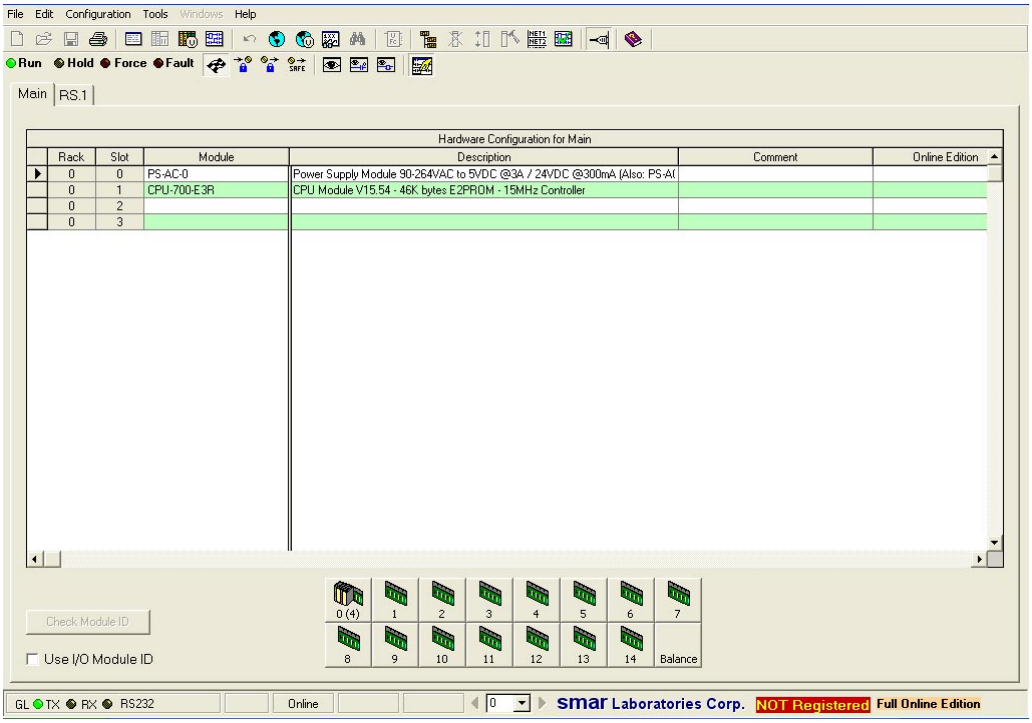


Figure 3.141- Hardware Page

4 Add another Remote Interface.

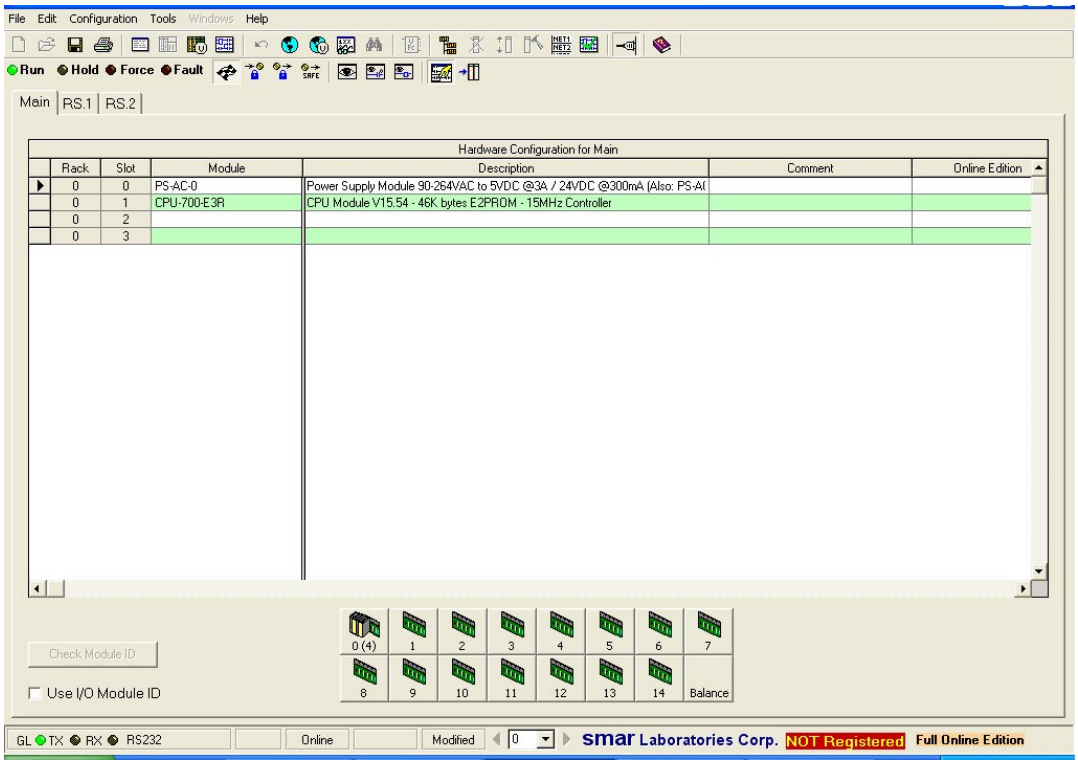


Figure 3.142- Adding another Remote Interface

5 Remove the existent modules from the Remote Interface 1 to Remote Interface 2.

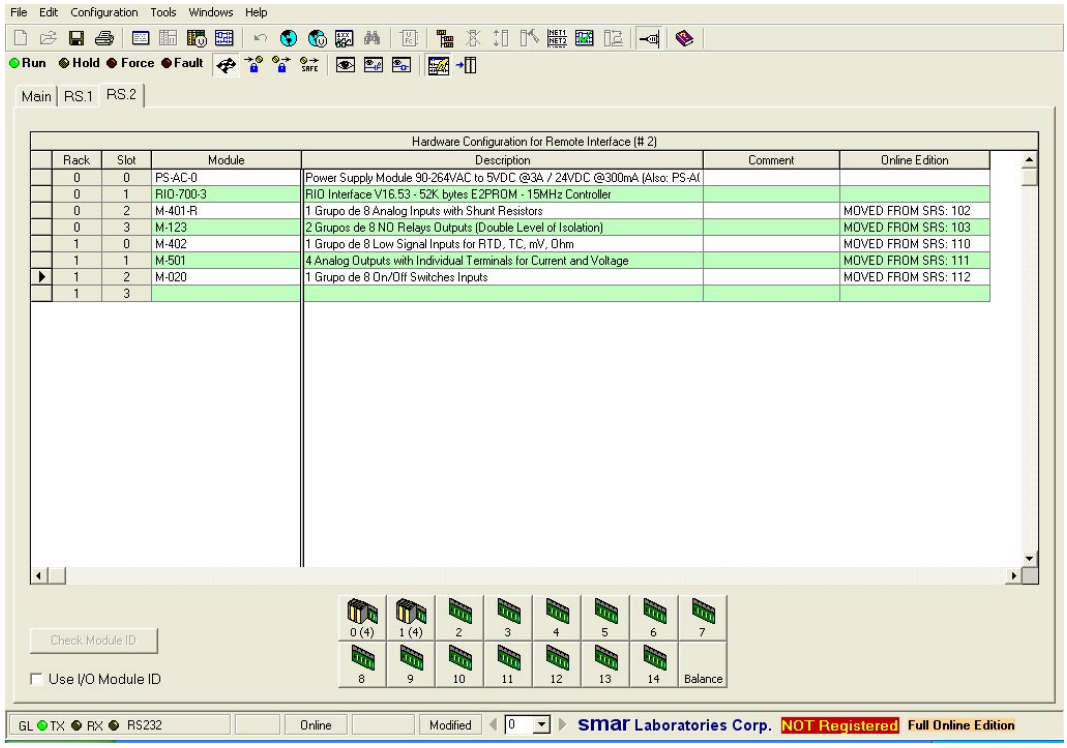


Figure 3.143- Moving modules from the Remote Interface 1 to Remote Interface 2

6- After moving the modules from Remote Interface 1 to Remote Interface 2, is can, in fact, remove RIO1. In the following picture, a confirmation message appears to confirm the action.

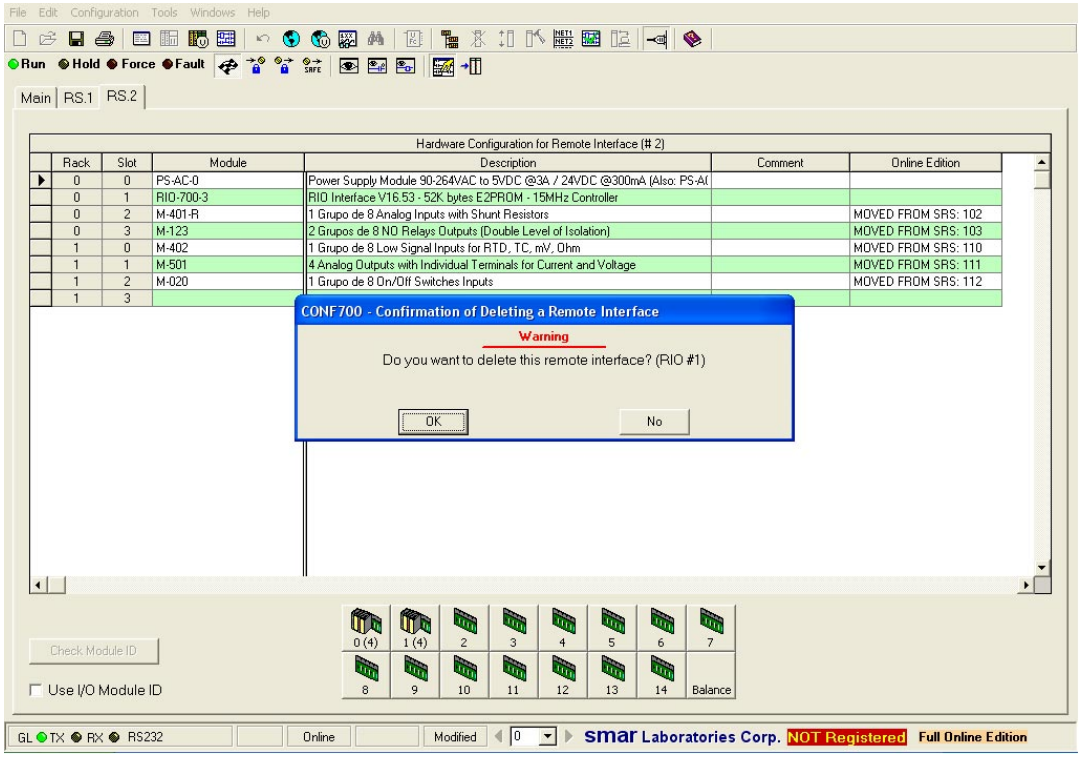


Figure 3.144- Confirming the Remote Interface removing

7 - The Hardware Page will be as in the picture below.

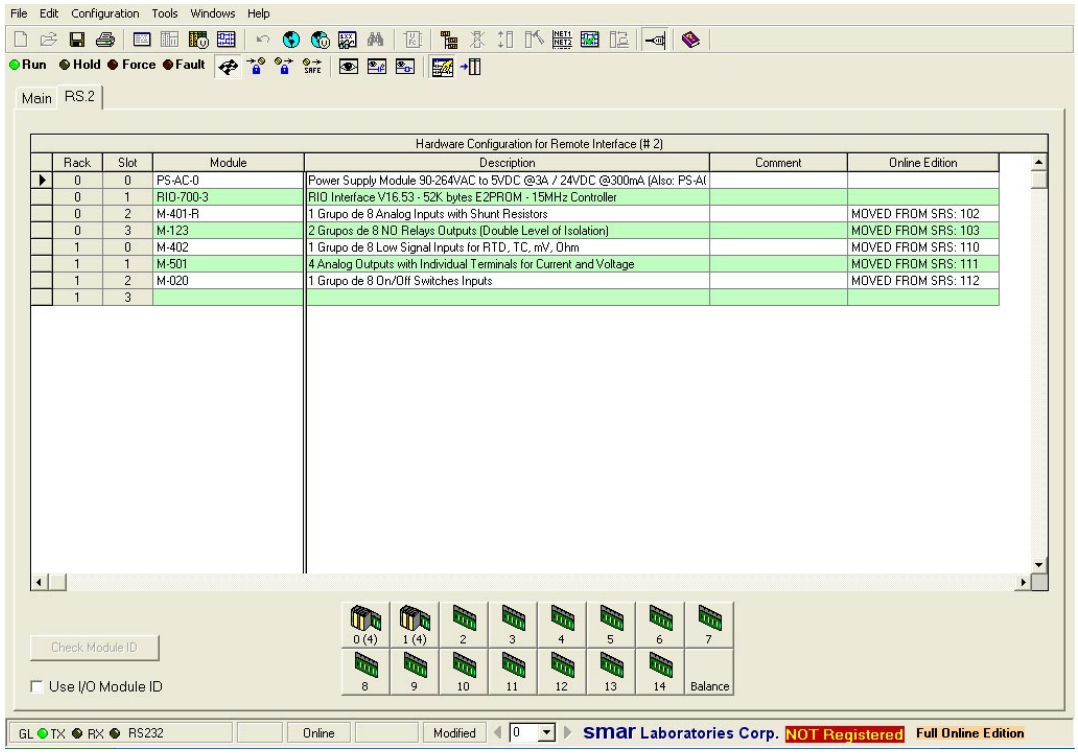


Figure 3.145- Hardware Page after removing the RIO1

8- Return to the Network Page and continue the configuration update.

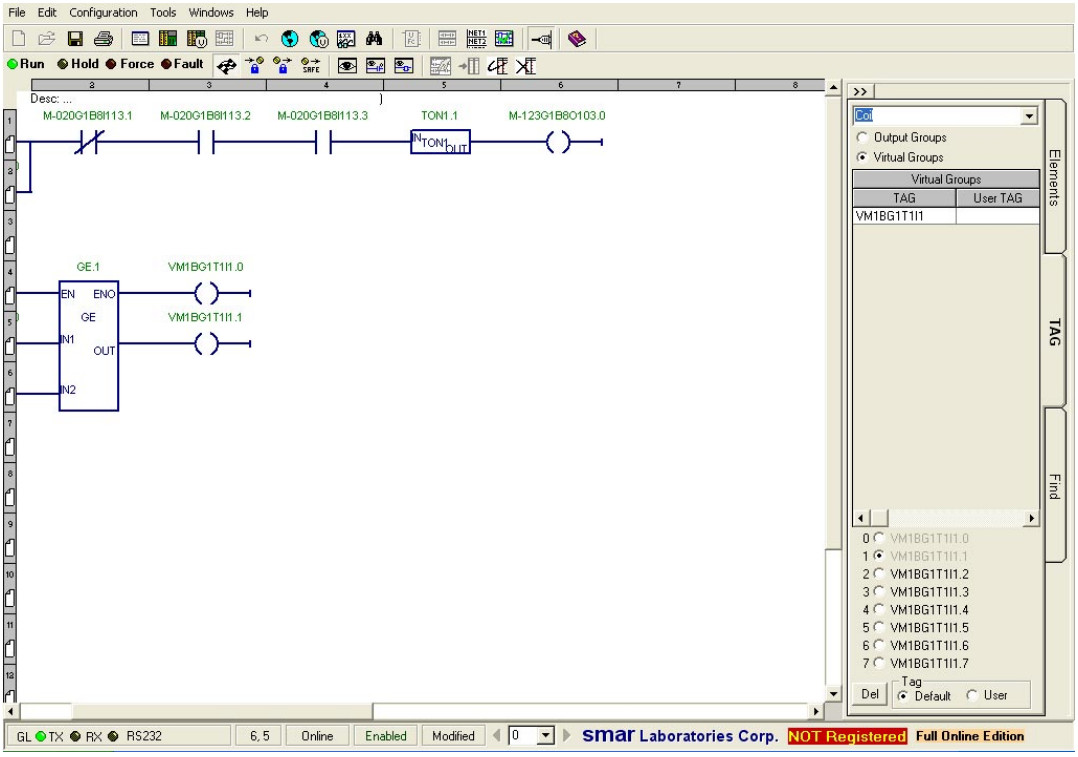


Figure 3.146- After the Send button

Connecting the LC700 to an HMI

Connecting a host computer to one or more LC700s can be done using a serial port or an Ethernet adapter. Variables/parameters acquisition can be done with standard Modbus/RTU or Modbus/TCP drivers that work directly with the software application to act as an HMI.

Another technique is to select HMI ready to work as an OPC client based only on variable Tags and then install the LC700 OPC server to directly communicate with the LC700 units. See more information on both processes below.

OPC (OLE for Process Control)

What is OPC?
OPC is a widely accepted industry standard. It is a client-server technology for interchanging parameter values between applications. It also makes device variables available in a standard way where multiple clients can simultaneously access them. Regardless of whether the server is located in the same workstation or remotely over a Microsoft network. OPC technology runs on Windows NT or Windows 2000.

Benefits for the LC700
The OPC technology enables a single distributed Fieldbus database to be shared among workstations, eliminating inconsistencies. After the configurations for all of the LC700s in the system have been created and debugged, all of the variable tags, their corresponding CPU addresses and Modbus reference addresses are exported as a “Tag List” to the LC700 OPC server.

The LC700 OPC server has all of the information needed to access any Tag on the corresponding CPU-700 and therefore provide all of the parameters/variables based solely on their tag for requisitions coming from the OPC client. Considering this fact, neither the user nor the HMI software needs to deal with addresses anymore but only with Tags!

This means the system becomes completely transparent, and the user should no longer worry about addresses at all.

Even if the Modbus register addresses change due to configuration modifications, it does not affect the OPC clients. Therefore no reconfiguration is required because the OPC client should always refer to a variable through its Tag and not its address.

In case the configuration changes, a new Tag List must be recompiled and registered to the LC700 OPC server again. This task is very fast and can be accomplished in a matter of minutes.

The software on the workstation must be an OPC client compatible application. It accesses LC700 data through the LC700 OPC server. Most typically all of the HMIs in the market are ready for the OPC. The figure below shows a typical relationship between different components of the OPC architecture.

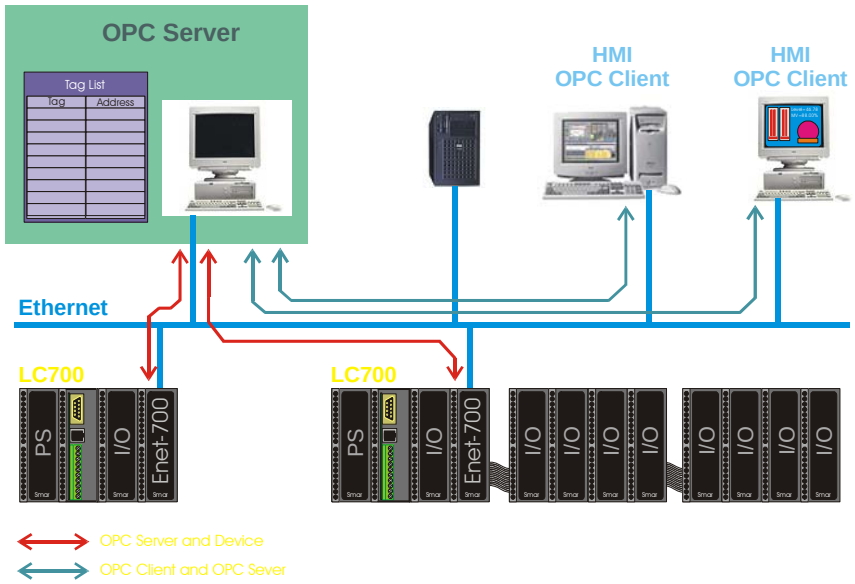


Figure 3.147- A Typical OPC Architecture

Smar LC700 OPC Server
The LC700 OPC server is a Windows application developed by Smar. Based on a given Tag List it can access data from one or more LC700 CPUs and make the data available for any OPC client in the network.

The LC700 OPC server can get data from the CPU-700 modules through serial ports or Ethernet connections.

Smar LC700 Tag List Generator
The LC700 Tag List generator is a Windows application that can combine one or more LC700 configurations. It can create a Tag List with device addresses (Modbus ID or IP address for Modbus/TCP) and Modbus addresses for every variable/parameter configuration selected.

The Tag List generator is also responsible for registering the Tag List on the computer where the LC700 OPC server will run.

Please refer to the Tag List Generator Manual for further information.

Configuring the OPC client
The OPC client in the workstation typically has an OPC browser that allows the user to navigate through to the tag to be displayed. First select the OPC server "Smar.LC700Server.0" then the configuration, in this example "mn_700", followed by the user tag, e.g. “FIC-123.PV”. I.e. the same tag that was configured in the CONF700 is used by all software applications throughout the system, with no need for retyping or renaming. Remember to write values in ICT blocks (e.g. setpoints) to the internal parameters, not to the output. Likewise, PID output should be written in manual to the output of the SMPL block.

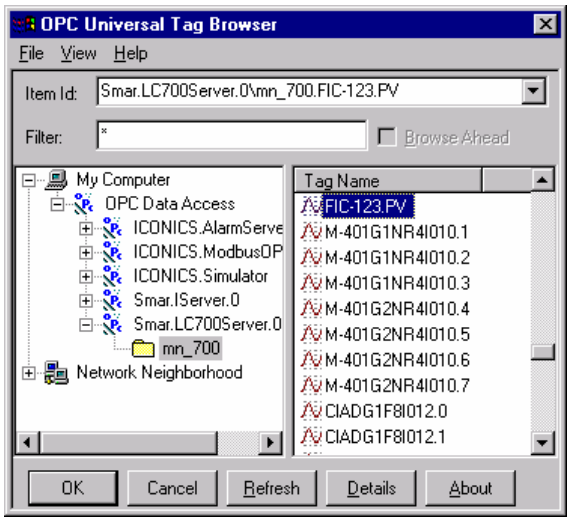


Figure 3.148- An OPC Client

The LC700 analog values are represented in percentage scale as an integer 0-10,000. To get an Engineering Unit readout in the OPC client software it has to be on a scale of the analog values from the OPC server.

Using Communication Drivers with MODBUS

Modbus Communication

HMI Softwares that do not support OPC cannot use the OPC server to get data to and from the LC700. Therefore they need to communicate directly with the LC700 using the Modbus register numbers to address the data in the memory of the CPU700.

The CONF700 software automatically assigns Modbus memory register addresses to all elements and block parameters, and generates a cross-reference listing for all of the Modbus register numbers. This makes mapping of the data in the application software easier.

Select on menu: Configuration/Modbus Address to call a Modbus list on the screen. This list can also be printed for convenience, go to menu File/Print and select the option for the Modbus Variable address

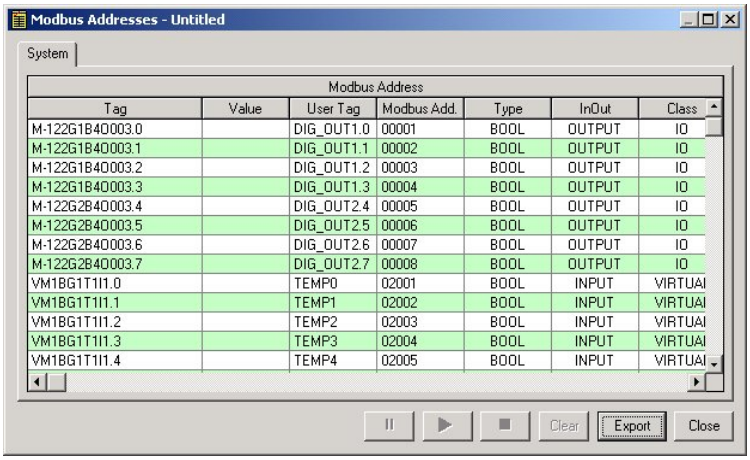


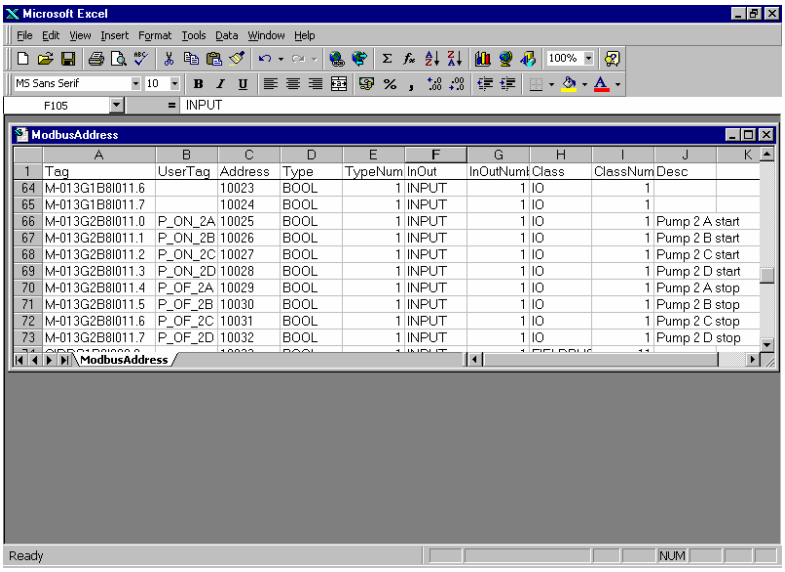
Figure 3.149- Modbus Addresses

Because CONF700 is designed to work on the standard Windows operating system, it benefits from the latest information technology integrating with the common MS-Office suite. The Modbus register's list can be exported to MS-Excel at the click of a button. This flexibility and openness makes the data available to other applications. In Excel the user can easily sort and filter the information to get just the the information necessary.

Click the “save” button and select the type of file necessary to export and rename the new file.

Figure 3.150- Exporting MODBUS Addresses

Example of a Modbus address list opened in Excel



	A	B	C	D	E	F	G	H	I	J	K
1	Tag	UserTag	Address	Type	TypeNum	InOut	InOutNum	Class	ClassNum	Desc	
64	M-013G1B8I011.6		10023	BOOL	1	INPUT	1	IO	1		
65	M-013G1B8I011.7		10024	BOOL	1	INPUT	1	IO	1		
66	M-013G2B8I011.0	P_ON_2A	10025	BOOL	1	INPUT	1	IO	1	Pump 2 A start	
67	M-013G2B8I011.1	P_ON_2B	10026	BOOL	1	INPUT	1	IO	1	Pump 2 B start	
68	M-013G2B8I011.2	P_ON_2C	10027	BOOL	1	INPUT	1	IO	1	Pump 2 C start	
69	M-013G2B8I011.3	P_ON_2D	10028	BOOL	1	INPUT	1	IO	1	Pump 2 D start	
70	M-013G2B8I011.4	P_OF_2A	10029	BOOL	1	INPUT	1	IO	1	Pump 2 A stop	
71	M-013G2B8I011.5	P_OF_2B	10030	BOOL	1	INPUT	1	IO	1	Pump 2 B stop	
72	M-013G2B8I011.6	P_OF_2C	10031	BOOL	1	INPUT	1	IO	1	Pump 2 C stop	
73	M-013G2B8I011.7	P_OF_2D	10032	BOOL	1	INPUT	1	IO	1	Pump 2 D stop	

Figure 3.151- Opening MODBUS Addresses list on MS-Excel

Modbus address coding

The LC700 uses the open industry standard Modbus/RTU and Modbus/TCP protocols for communication. In Modbus, information is mapped into registers. There are four “references” (groups):

0xxxx (Output Coils), Discrete output 1xxxx (Input Contacts), Discrete input, Fieldbus status 3xxxx (Input Registers), Analogue input 4xxxx (Holding Registers), Analogue output

CONF700 automatically assigns these Modbus addresses to I/O, ladder elements and function block parameters etc. eliminating tedious work and human errors. If the OPC server is used, the user no longer needs to be concerned about Modbus registers at all, since these will be completely handled by OPC server.

Implications in Modifying a LC700 Configuration

In older versions of the CPU-700 Modbus registered addresses were changed every time the user changed the configuration. In the current version, all registered addresses are kept. Modbus variables are physically stored in a sequential order of absolute address. Starting from the main I/O and next RIO1 to RIO6. Modbus areas have logic separation ranges so that all these ranges refer to the main CPU or to the remote I/O CPU.

When the user adds a new module, the old area of Modbus registers will be in the same position. CONF700 only inserts the new positions in the area reserved without any dislocation. The user no longer needs to worry about this allocation because the CONF700 does it automatically.

Digital Memory Map

Type Unit	Modbus Address	Number Of Points
Inputs	**Range 1x xxx**	
DI – MASTER	10 001 – 13 000	3000
DI – RIO #1	13 001 – 13 500	500
DI – RIO #2	13 501 – 14 000	500
DI – RIO #3	14 001 – 14 500	500
DI – RIO #4	14 501 – 15 000	500
DI – RIO #5	15 001 – 15 500	500
DI – RIO #6	15 501 – 16 000	500
Outputs	**Range 0x xxx**	
DO – MASTER	00 001 – 02 000	2000
Discrete Virtual Variables	**02 001 – 03 000**	**1000**
DO – RIO #1	03 001 – 03 500	500
DO – RIO #2	03 501 – 04 000	500
DO – RIO #3	04 001 – 04 500	500
DO – RIO #4	04 501 – 05 000	500
DO – RIO #5	05 001 – 05 500	500
DO – RIO #6	05 501 – 06 000	500
Discrete Virtual Variables *	**06 001 – 07 000**	**1000**

*** Additional Range**

Analog Memory Map

Type Unit	Modbus Address	Number Of Points
Inputs	**Range 3x xxx**	
AI – MASTER	30 001 – 31 000	1000
AI – RIO #1	31 001 – 31 250	250
AI – RIO #2	31 251 – 31 500	250
AI – RIO #3	31 501 – 31 750	250
AI – RIO #4	31 751 – 32 000	250
AI – RIO #5	32 001 – 32 250	250
AI – RIO #6	32 251 – 32 500	250
Reserved	32 501 – 34 000	1500
Outputs	**Range 4x xxx**	
AO – MASTER	40 001 – 41 000	1000
AO – RIO #1	41 001 – 41 125	125
AO – RIO #2	41 126 – 41 250	125
AO – RIO #3	41 251 – 41 375	125
AO – RIO #4	41 376 – 41 500	125
AO – RIO #5	41 501 – 41 625	125
AO – RIO #6	41 626 – 41 750	125
Reserved	41 751 – 41 875	750
Function Blocks	**42 501 – 49 950**	**7450**
Special Registers	**49 951 – 49 999**	**49**

Special Registers

The CONF700 has a few special registers that can be acessed by an HMI interface through the MODBUS addresses of these registers. To access these registers, on CONF700 click at MODBUS REGISTERS.

The table below shows the special registers:

Special Registers	MODBUS Address	Description
RTC_Sec	49951	Real time clock : Seconds (00-59)
RTC_Hour	49953	Hour (00-23)
RTC_Dweek	49994	Day Of The Week (01-07)
RTC_Day	49955	Day of The Month (01-31)
RTC_Mon	49956	Month (01-12)
RTC_Year	49957	Year (00-99)
ScanCicleTime	49958	Tempo real de execução do programa
TimeOutPort01	49959	P1 port Communication Timeout (multiple of 10 miliseconds)
TimeOutPort02	49960	P2 port Communication Timeout (multiple of 10 miliseconds)
TimeOutPort03	49961	P3 port Communication Timeout (multiple of 10 miliseconds s)
TransferState	49962	Reserved
ReadyScanRio (*)	49963	Communication Status between master CPU and RIO Module(valid only for Redundant Systems)
SSIO Status (*)	49964	Communication status between CPUs (valid only for Redundant Systems)
Bat Status	49965	Battery status 1 – Is not OK 0 - OK

(*) Refers only to the CPU-700-E3R

ReadyScanRio

If the bit is on state On (1), it is possible to communicate with the RIO module and the configuration is the same both in the main CPU and in the RIO Module.

Bit	Description	States	
		0	1
0	Not used	-	-
1	Ready for Scan RIO 1	No	Yes (RIO 1)
2	Ready for Scan RIO 2	No	Yes (RIO 2)
3	Ready for Scan RIO 3	No	Yes (RIO 3)
4	Ready for Scan RIO 4	No	Yes (RIO 4)
5	Ready for Scan RIO 5	No	Yes (RIO 5)
6	Ready for Scan RIO 6	No	Yes (RIO 6)
7	Ready for Scan RIO 7	-	-

SSIOStatus

Bit	Description	States	
		0	1
0	Good Connection on the SSIO Channel	No	Yes
1	Redundant Buffer is busy	No	Yes
2	Configuration Check	The CPU Main and Backup Names are the same	The CPU Main and Backup Names are different
3	CPU Identification	Backup	Main
4	-	There are either CPU Main or two CPU Backup	There are CPU Main and Backup

Bit	Description	States	
		0	1
5	Key States (rotary key and dip switch)	The CPU Main keys are different from CPU Backup	The CPU Main and Backup keys are the same
6	Firmware Check version	The CPU Main and Backup versions are different	The CPU Main and Backup versions are the same
7	Status received from another CPU by the SSIO channel	No	Yes

Note: These data types are Word (“MSB LSB”). The first eight least significant bits (LSB) represent this status.

Manual Modbus Addresses Attribution

Each I/O Module and Function Block included in the configuration allocates memory space in the CPU. This memory space has a Modbus Address. The user has the option to manually allocate this address or let the application do it automatically. Once allocated the Modbus Address is fixed unless changed by the user.

In the automatic mode, CONF700 attributes addresses to all of the points but the user can change the address by switching to the manual mode. The user can change from automatic to manual and vice versa by choosing 'M.Addr' tab in the preferences window which is under 'Tools' menu item.

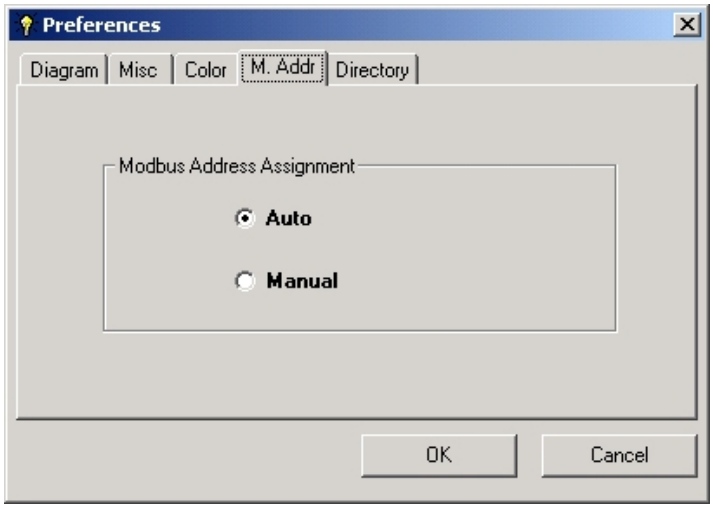


Figure 3.152- Setting Manual or Automatic MODBUS Addressing

Automatic Modbus Address Allocation

Select 'Preferences' from the 'Tools' menu item. Click on the 'M.Addr' tab. If the 'Auto' option is already selected just click 'Cancel', if not select the 'Auto' option and click 'OK'. The Modbus Address attributed to all of the points is now given automatically. It will be given in the sequential manner, by selecting the first available slot in which all of the points in a Module or a Function block fit. It follows a different address range for different data types as explained in the previous section.

Manual Modbus Address Allocation

Select 'Preferences' from the 'Tools' menu item. Click on the 'M.Addr' tab. If the 'Manual' option is already selected click 'Cancel', if not select the 'Manual' option and click 'OK'. In this mode the user will be given the option to select a Modbus Address each time a Module or a Function Block is included.

I/O Module Modbus Address Allocation

When the user inserts a I/O Module in the 'Hardware Page' the 'Modbus Address' window will open. This window has the following options as shown in the figure 3.93. It shows the Modbus Addresses for the module M-207, which has one group of eight Inputs and one group of four Outputs.

I/O Type: Indicates that the points are of type Digital Input(DI), Digital Output(DO), Analog Input(AI) or Analog Output(AO).

Point(s): Gives the number of points of the 'I/O Type' in the module.

Data Type: Gives the data type of the points in the module.

Modbus Address: The default Modbus Address that the system gives.

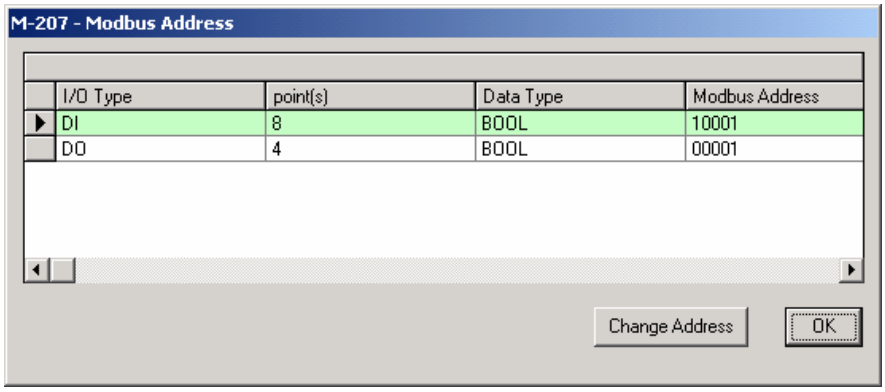


Figure 3.153- Setting MODBUS Addresses Manually

To change the Modbus Address to a desired location, click on the 'Change Address' command button. This gives a list of available Modbus Addresses as shown in the figure below.

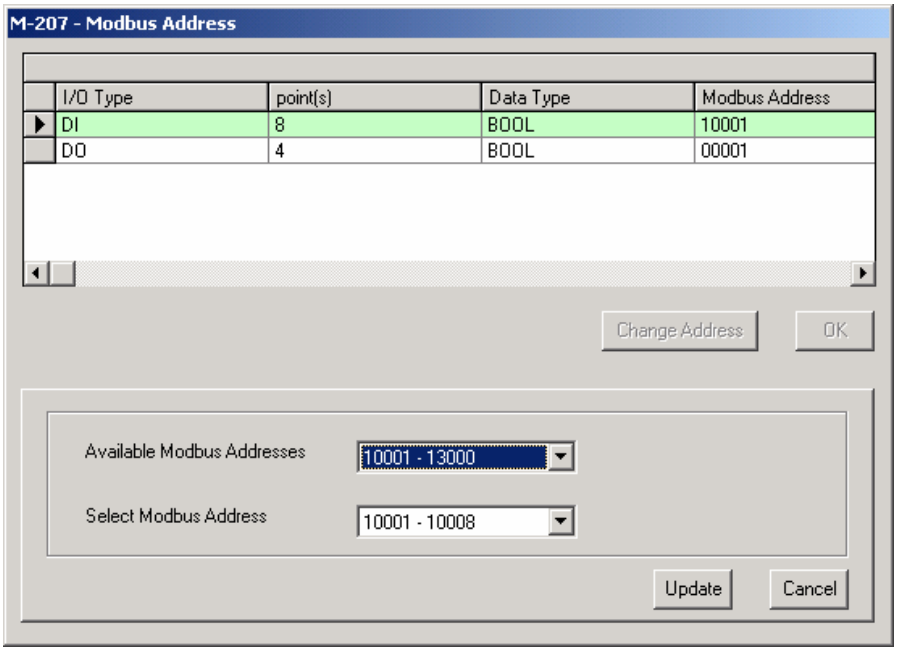


Figure 3.154- Selecting MODBUS Addresses Range

Select a slot from the 'Available Modbus Addresses' list and the 'Select Modbus Address' list box shows the list of the addresses available in that slot. Select the desired Address. To accept the new address, click on 'Update'. To keep the old address click on 'Cancel'. If the user wants to change the address again click on 'Change Address' otherwise click 'OK' to close the Modbus Address window.

Function Block Modbus Address Allocation:

When the user inserts a Function Block in the 'Network Page' the 'Modbus Address' window will open. This window has the following options as shown in the figure 3.95. It shows the Modbus Address for the Function Block 'ABS'.

Function Block: Gives the type of Function Block.

No. of Registers: Gives the number of Registers allocated in the memory for this Function Block.

Modbus Address: Gives the default Modbus Address.

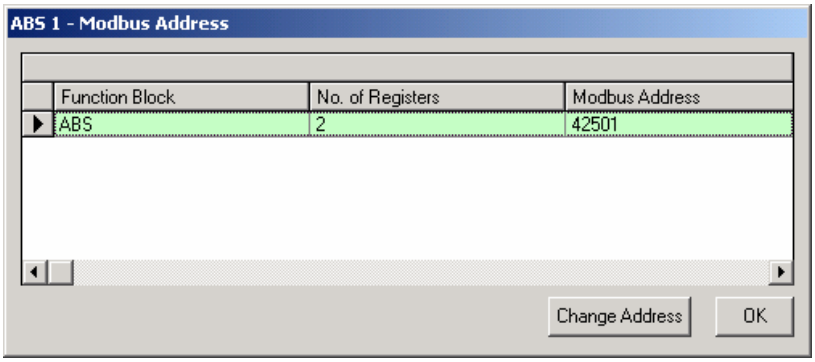


Figure 3.155- Setting Function Blocks MODBUS Addresses

To change the Modbus Address to a desired location, click on the 'Change Address' command button. This gives a list of available Modbus Addresses as shown in the figure below.

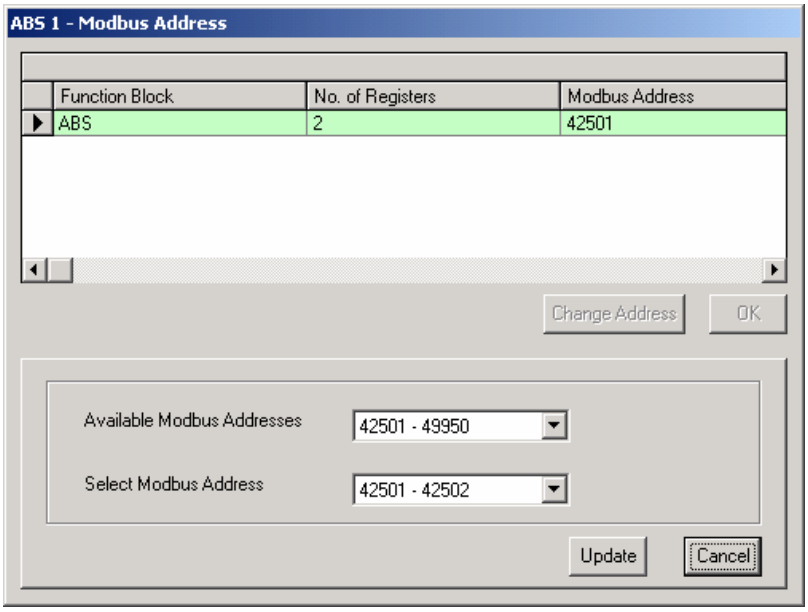


Figure 3.156- Setting Function Blocks MODBUS Addresses Range

Select a slot from the 'Available Modbus Addresses' list and the 'Select Modbus Address' list box shows the list of the addresses available in that slot. Select the desired Address. To accept the new address, click on 'Update'. To keep the old address click on 'Cancel'. If the user wants to change the address again click on 'Change Address' otherwise click 'OK' to close the Modbus Address window.

User Function Blocks

Introduction

User function blocks (UF) are boolean functions created by the user to simulate combinations of normally closed relays (NC), normally open relays (NO) and coils.

To create user functions it is required to have one or more module on the Hardware Page or one or more auxiliary variables in the Virtual Page Table. Once these functions are created, they will be available for use in the design of a logic Ladder diagram.

User defined functions may be used only once in the ladder logic configuration.

$$\left\{\begin{array}{c}\text{Actual Ouput}\\ \text{Function Output}\\ \text{Temporary Variable}\end{array}\right\} := \text{Function of } \left\{\begin{array}{c}\text{Actual Outputs/Inputs}\\ \text{Output of the defined functions}\\ \text{Previous Temporary Variables}\end{array}\right\}$$

The left-hand side (L.H.S) of the equation is for specifying the actual output, the function output, or the temporary variable. The symbol: = is the "Define" symbol and represents the equal sign. The right-hand side (R.H.S) of the equation is for actual I/O, or previous temporary variables or the combination among them via Boolean operators. The R.H.S of the Boolean equation could also be a function of the previous L.H.S temporary variables or the actual I/O.

Creating An User Function

1. In the main window of CONF700 click on the icon  or in the Edit menu and select Edit User Function. The following dialog box will open:

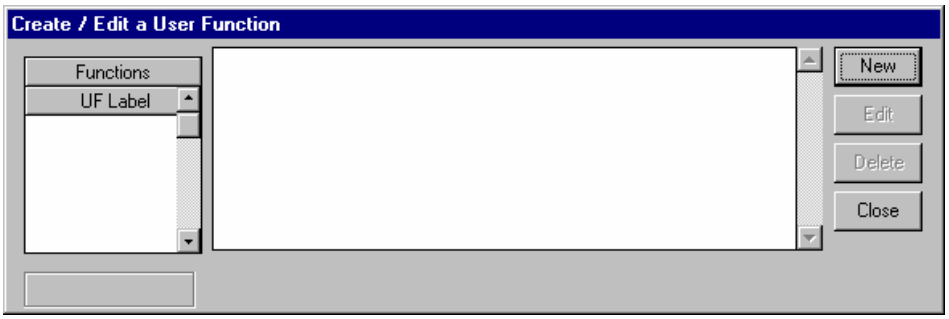


Figure 3.157- The User Function Creation/Edition Window

2. Click on the New button. This action will move the cursor to the UF column Label. Next give a name to this function.

3. Click on the button Edit in the above dialog box. CONF700 will generate a new UF name at the start of the logic program and the END command at the end of it. The program template will show as the example below:

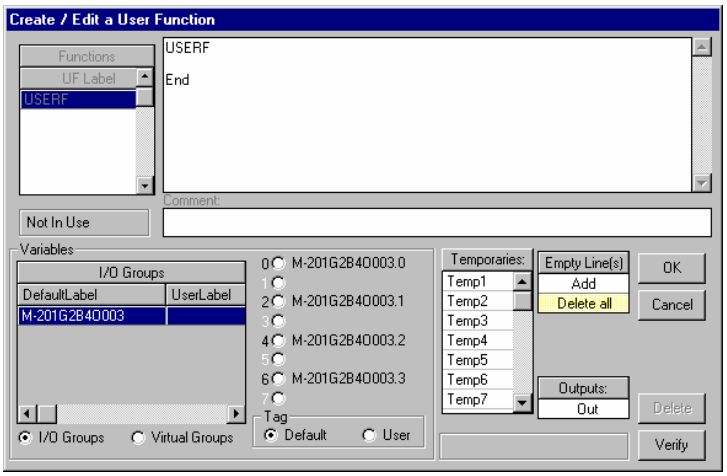


Figure 3.158- Programming a New User Function: Variables And I/O

In the Empty Line(s) box the user will be able to delete a whole line by clicking on Delete All or add a new line by clicking on Add. Verify will check programming logic syntax.

An I/O box allows user to select a channel of an I/O module to be part of the boolean equation.

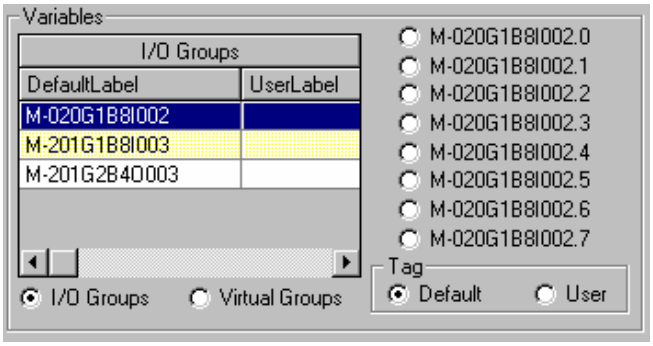


Figure 3.159- I/O Module as Variables In The User Function

Virtual variables may be selected by choosing Virtual Groups.

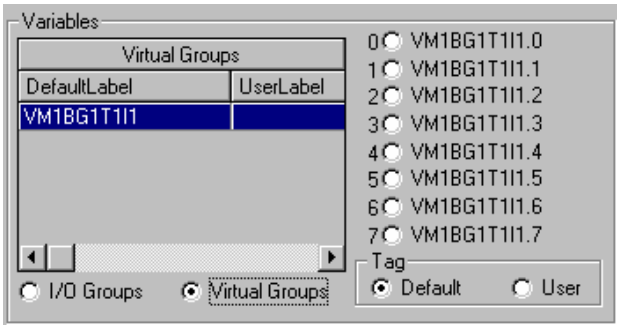


Figure 3.160- Virtual Variables as Variables in the User Function

Logic operations can be implemented by using the operators in the Symbols box.

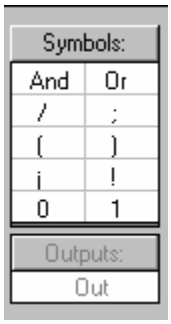


Figure 3.161- Symbols Used In the Creation Of User Function Logical Equations

These symbols are described in the table below:

Symbol	Description
And	Logic Operator AND (Binary AND)
Or	Logic Operator OR (Binary OR)
/	Logic Operator NOT (Binary NOT)
;	End of a logic statement (statement Syntax) , end of line, end of operation.
(	Group Logic Operations
)	Group Logic Operations
!	Logic Function of negative transition. Ex: X=!A; where X is 1 and if A changes from 1 to 0. (Descending Edge Transition)
I	Logic Function of positive transition. Ex: X=iA; onde X is 1 if A changes from 0 to 1. (Ascending Edge Transition)
0	Negative Logic Constant (FALSE)
1	Positive Logic Constant (TRUE)

CONF700 enables 16 temporary variables that may be used in the boolean equation.

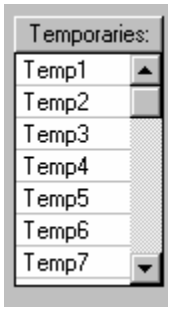


Figure 3.162- Temporary Variables

The user can set up to 8 outputs. To create an output the user has to click at the Outputs box. CONF700 will add a line with the value for the output as shown below:

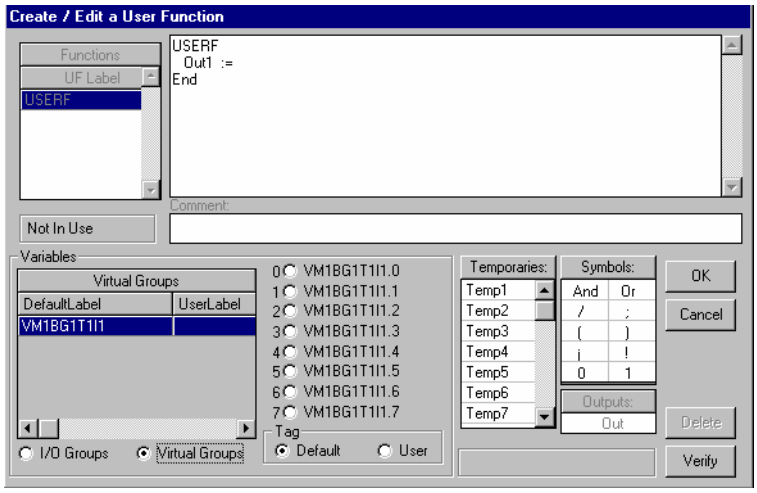


Figure 3.163- Creating An User Function

In case it is necessary to create a new expression for a temporary variable the user must click on the Temporaries box and select one of the 16 available variables.

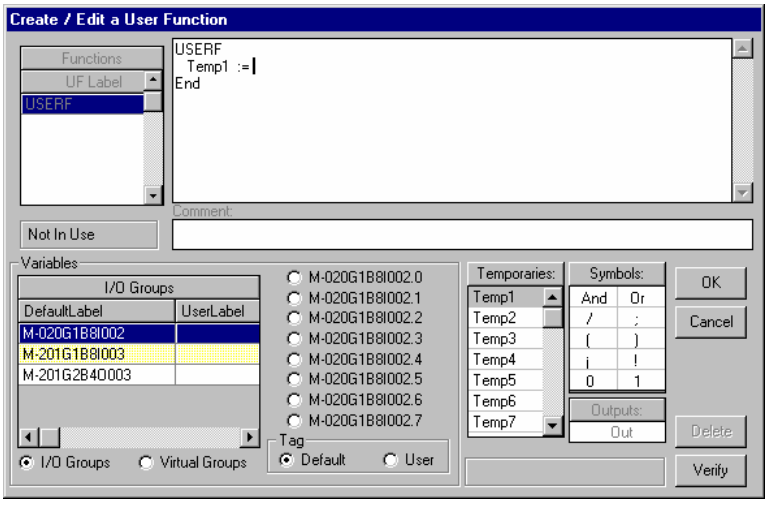


Figure 3.164- Creating an User Function

4. Clicking on the Outputs box we insert a Out1 line choosing the first variable clicking on one of inputs of the M-020 module through the I/O box. In the Symbols box we select the logic function AND. Next select the second variable as another input of the M-020 module. To end this statement we add a ";" symbol to indicate end of statement.

Next click on the Temporaries box, select one of the 16 variables available. CONF700 inserts this variable in the next line. We make this variable equal to the value of an input of the M-201 module, being cautious to insert a ";" symbol at the end of this statement/line. Next click on the Output box and generate a new output making this output equal to the result of the temporary variable 1 and logic with a third input of the M-020 module.

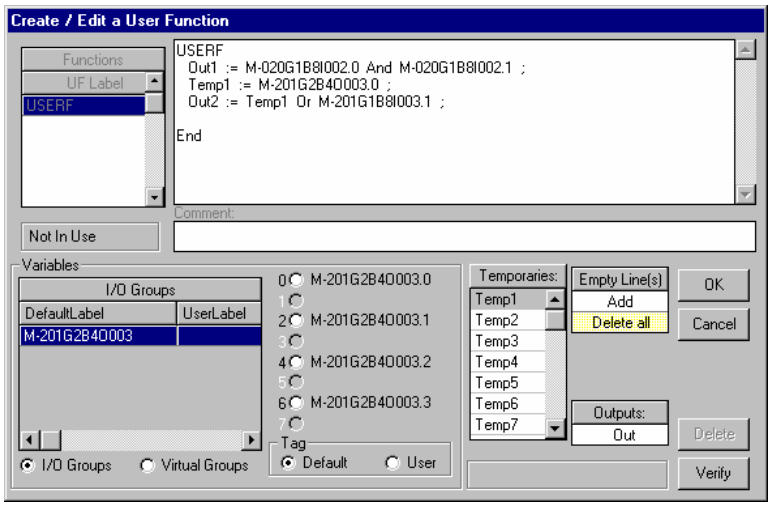


Figure 3.165- A New User Function: Check The Logical Equation

We get back to the first window. Click on Close, close it and the UF is available for use within the ladder logic. The user only needs to select UF in the Toolbar and insert the user function created.

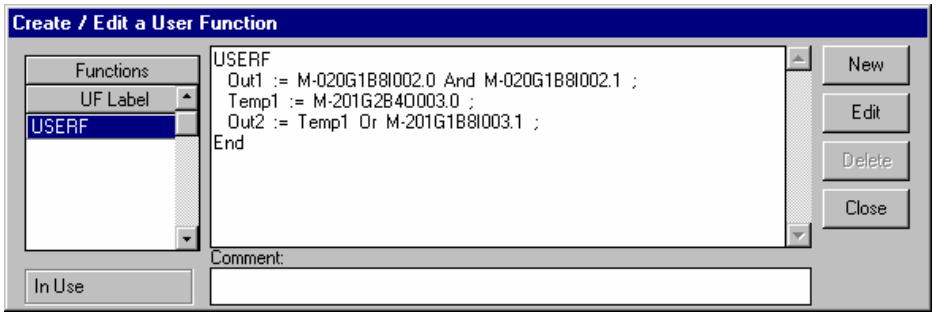


Figure 3.166- A New User Function Created

Warning messages

If the user tries to delete a logic programming element using the backspace key, the following window is shown.

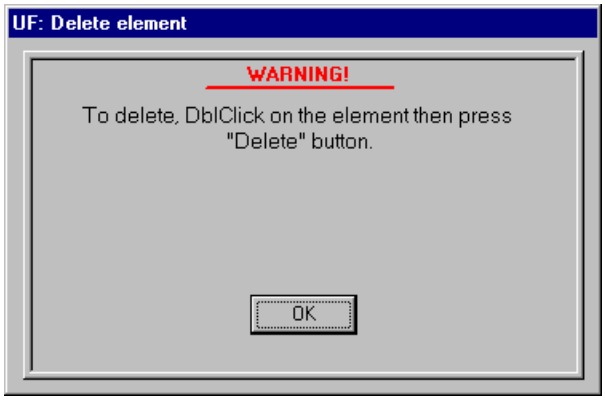


Figure 3.167- Warning Messages

To delete logic elements click twice over the element and then click the delete button. If user forgets to finish a statement line not including a ";", the following message will be shown.

Figure 3.168- Warning Messages

If user tries to delete an output the following warning message is displayed:

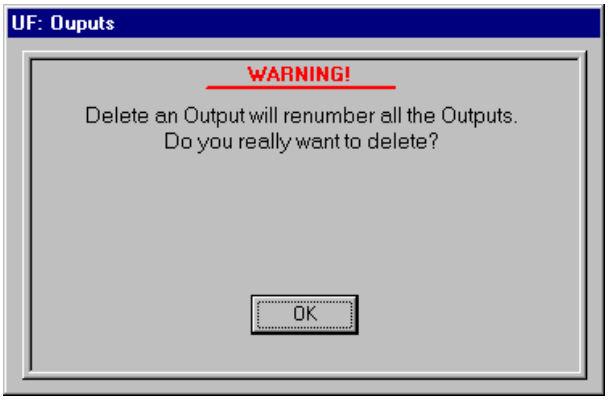


Figure 3.169- Warning Messages

Unlike deleting an input, erasing an output of a user function block will result in re numeration of all outputs.

How to estimate memory space used by user functions

Each user function may be used just once in any logic network. UF allows that logic is inserted directly through a boolean equation through the click-and-write method.

To estimate the memory size used by a user function we apply the following rules:

Any function itself uses 20 bytes
- 4 bytes for every time any temporary variable is used (Tempn)
- 7 bytes to each function output (Outn)
- 11 bytes to any variable not preceded by a symbol of transition sensor (^ or !)
- 17 bytes to any variable preceded by a symbol of transition sensor (^ or !)
- 4 bytes to every “AND”or “OR” operation
- 5 bytes for every constant “1” or “0”>

Example: Function SELECT. It simulates a multiplexer switch with 4 inputs (INA, INB,INC and IND) selected by SEL1 and SEL2. OUT1 represents the output of this switch. OUT2 shows if any of the inputs is zero. This function also prepares the coil 24 to indicate how a line selection could tolerate any change.

```
SELECT
TEMP1:=/SEL1*/SEL2*/INA;
TEMP2:=/SEL1*SEL2*INB;
TEMP3:=SEL1*/SEL2*INC;
TEMP4:= SEL1*SEL2*IND;
COIL24:=^SEL1+!SEL1+^SEL2+!SEL2;
OUT1:=TEMP1+TEMP2+TEMP3+TEMP4;
OUT2:=INA+INB+INC+IND;
END_SELECT
```

Thus:

Rule	Description	Number of bytes
1	It is necessary 20 bytes for one User Function	20
2	Temporary variables were user 8 times	8x4
3	This UF has two outputs	2x7
4	12 variables do not use the transition sensor ^ or !	12x11
5	4 variables using symbols of transition sensors	4x17
6	14 ANDs and ORs	14x4
7	No constants used	0
	Total	**322 bytes**

Editing an user function

Click on the icon  or in the menu Edit => User Function. A dialog box will be displayed. Choose the UF by clicking on it. Click on Edit. Following the steps described previously user might edit the UF.

Optimizing Hardware for an Application

As a general rule, the more details of the current application are known, more precision this estimation will have.

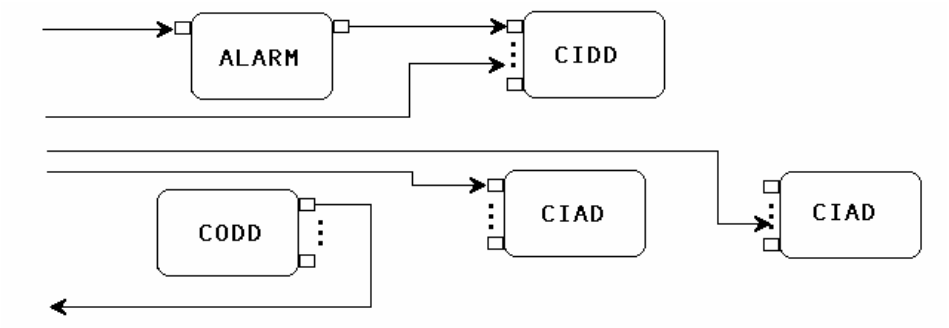
It is important to have in mind that hardware for a LC System does not depend on the number of I/Os necessary to interact with a field Also another important factor Is that it is required to evaluate the complexity of the Ladder Logic Network (requirement for memory setting) and maximum acceptable runtime.

Below we present an example where we try to estimate hardware based on the number of I/Os. For this example, the CPU-700-E3 has been used and it supports up to 2000 digital points. Suppose the LC700 will not consider either the amount of space necessary to the Ladder Logic or running time.

Ex:

User wants to have 1100 digital inputs, 60 digital outputs and one fieldbus channel.

Assuming the fieldbus block will have the following block use:



To estimate the amount of memory used for the FB channel it is necessary to focus on the blocks used, not on the number of links. Alarm blocks do not reserve any space of memory while the other blocks reserve an amount of digital points or analog points according to the right column of the table in the Fieldbus section.

In our example there's a CIDD and a CODD, each one requires 8+8 digital points. Besides, two CIAD blocks are used, which requires a memory space of 16+16 analog signals (float) and 8+8 digital points

Conclusion taken is, the FB module will use 48 digital points and 32 analog signals.

Next we estimate modules used.

	Digital Inputs	Digital Outputs	Reminding Points
	1100 Points	600 Points	(2000-DI-DO)
System PLC 01	550	300	1150
System PLC 02	550	300	1150

System PLC 01:

Digital Points	Digital Inputs	Digital Outputs	Total	Limits
16 Points I/O Modules	430	100	530	
8 Points I/O Modules	120	150	270	
4 Points I/O Modules		50	50	
2 Points I/O Modules			0	
FB modules			48	
		Total	898	< 2000 Digital Points

Analog Points	Analog Inputs	Analog Outputs	Total	Limits
16 Points I/O Modules	0	0	0	
8 Points I/O Modules			0	
4 Points I/O Modules			0	
2 Points I/O Modules			0	
FB modules			32	
		Total	32	< 1024 Analog Points

Modules	Inputs	Outputs	Total	Limits
CPU Module(250 mA)			1	
16 Points I/O Modules	28	7	35	
8 Points I/O Modules	16	10	26	
4 Points I/O Modules				
2 Points I/O Modules				
FB Modules (300 mA)			1	
Power Source Modules 5VDC@3A/24VDC@300mA			2	Aprox. 80 mA (5V)/ module
	Total		66	<120 modules

Accessories	Totals
Racks	17
Flat- Cables	16
Bus Terminator	1

Chapter 4

HELP FOR PLANT STARTUP WITH THE LC700

The objective of this Chapter is to guide how to obtain the values of the communication parameters for a good performance in communication.

- ✓ Have at hand the manuals: User Guide LC700 and Tag List Generator LC700.

1) Communication Parameters

1.1) Time Delay: See page 3.53
Adjust Time Delay with (0) default value, if the station used is equivalent to or above a Pentium III or another that does not cause a communication failure.

- If failures happen in communication, increase the **Time Delay** value to a maximum up to 50ms. See page 3.53.

1.2) Offduty: See page 3.53

- **Tipe of CPU:**
  - ***D3/D3R:*** Use the (0) default value, which makes the CPU dynamically calculate the **Offduty** with a value greater than 20% of the logic ladder execution time. Should the supervisory be slow or failing, define an Offduty value larger than 20% of the cycle to improve the performance.
  - ***E3/E3R:*** The Offduty parameter **is only effective** during the **on-line edition** procedure. Make tests executing the on-line edition and check if the supervisory works without failure. If necessary, adjust the Offduty parameter, increase it gradually every 10ms until the test is carried out successfully.

1.3) Baudrate for P1: (See in "LC700 User Guide " Manual - page 3.13)
Always use 9600bps.

- Use the RS-232-C gate for maintenance purposes.

1.4) Baudrate for P2 and P3 Modbus:
Use 38400bps for redundant systems and up to 115200bps for non redundant systems.

- The RS-485 gates are recommended for supervision work.

1.5) Baudrate for P3 RIO:
Use 115200bps:

- If the cycle time is very high and does not respond to control, use 230400bps;
- If the environment is too noisy and causes communication failures in the RIO-Master , use 57600bps;

2) Time outs

2.1) **CONF700:** adjust the value making **download/upload** tests, departing from the default value (5000ms). Keep the same default value if it works.

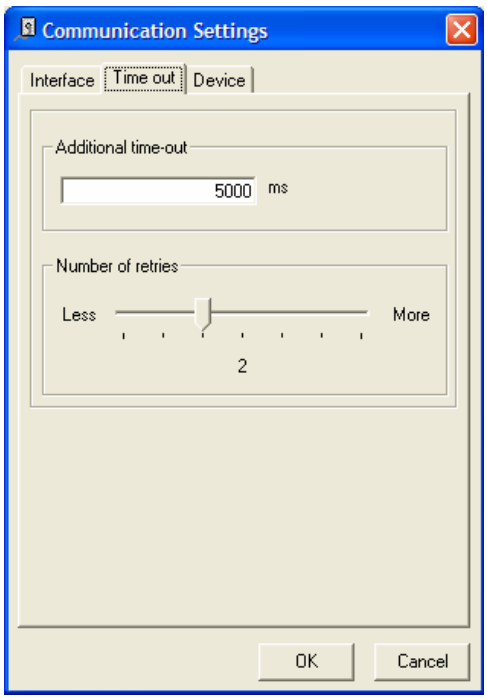


Figure 4.1 –CPU-700 Communication Parameters

2.2) **Ethernet Interface (ENET-710, MB-700, etc.):** (See "LC700 User Guide" Manual - page 3.101).
For Supervision: adjust the value to twice the largest cycle time between the PLCs connected to the interface.
For On-line Edition with CONF700: It is necessary a **Time out** much larger than those for the supervision and an alteration of the interface **Time out** should be made for at least 5000ms.

2.3) **TagList Generator:** See page 3.69.
Minimum value = 2x(Ethernet interface time out) x (number of stations that execute the access to the Ethernet interface).

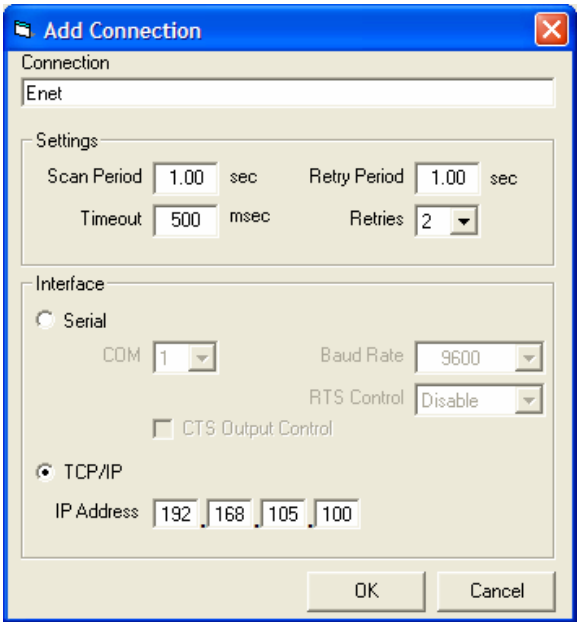


Figure 4.2 - Adjusting Timeout in the Tag List

Considerations

1) On-line Edition in CONF700: See page 3.67
Some On-line Edition commands require a **Time out** much larger than those set for the supervision. Therefore, make the On-line Edition via the P1 (RS-232-C) gate directly from a Lap-top.

- To make the On-line Edition via an Ethernet interface, make a **temporary alteration** of the Time out interface for a minimum 5000ms value.

2) Decision among the use or not of the options "Use Block View" and "Use Digital Block" in TagList Generator: See "Tag List Generator LC700" Manual - page 8
The simplest way to verify which option is best is through tests using all tag combinations. For each one of them check the number of commands used by the OPC Server through the OPC (**NbrCommCmmd**) status Tag.

- Use the combination resulting into a smaller number of commands.

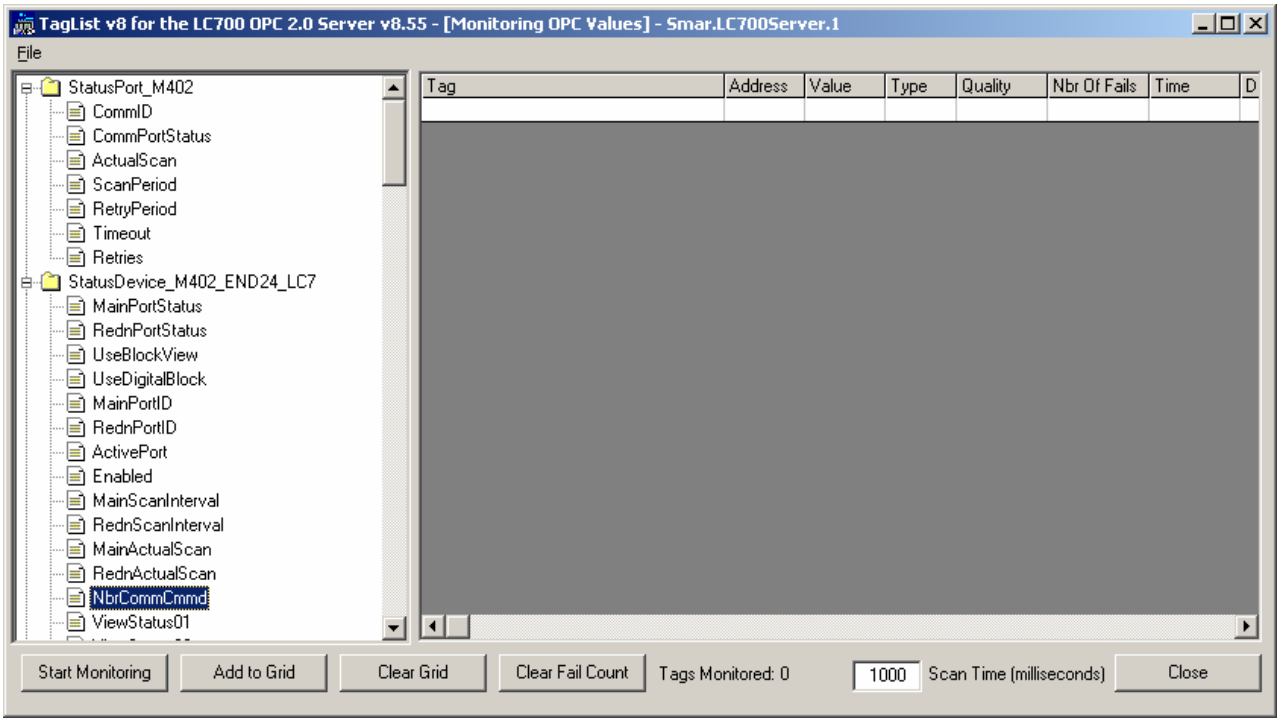


Figure 4.3 –OPC NbrCommCmmd Status Tag

The "**Use Digital Block**" tag in most cases optimizes communication, although it depends on the number of monitored digital points.

- If the "Use Block View " can monitor everything with a single command, use only this tag.

- For a large number of digital points (more than 90) use the "Use Digital Block " tag, as it always optimizes communication.

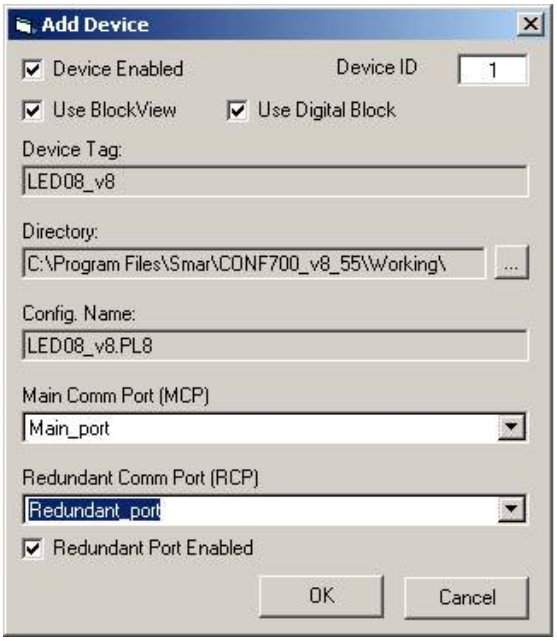


Figure 4.4 - LC700 Configuration in the Tag List

TROUBLESHOOTING

1. When I try to establish communication between my workstation and LC700 CPU I get the following message: "Unknown Device".

Solution: Download firmware again through software LC700Tools.

2. When I try to download my configuration to the LC700 CPU I receive the following message: "Acknowledgement for sending the function block is incomplete".

Solution: Go to the Tools menu, then Comm. Settings and change Timeout for 5000 ms.

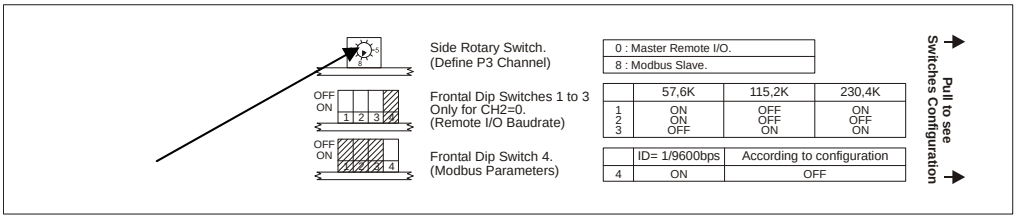
3 When I am monitoring a plant, the monitoring seems to be too slow.

Solution: To increase monitoring speed go to menu **Tools→Preferences** and click on the Misc tab then change the field "Network Monitoring Period".

4. Failure in a peer-to-peer communication between the CPU master and the PC. After starting communication and clicking on "Look", I can't establish communication.

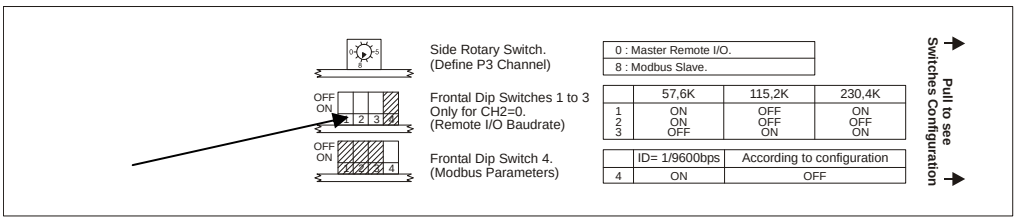
STEP1: Check the communication cable for a proper connection in the serial ports of your workstation and LC700.

STEP2: Check the Rotary Key located in the CPU.



This key must be placed in the position 8 if the configuration does not have an RIO. Otherwise, i.e., if there is an RIO connected to the master, this key must be placed in the position 0.

STEP3: Move the CPU DIP Switch to the default position.



The default position will be set if the key is placed in the position above. On the CONF700, in the window LC ONLINE make sure the default option was selected.

STEP4: If you proceeded with all previous steps and communication still was not established, the user should check in which serial port of his workstation he connected the LC700 CPU. 3 serial ports are generally available: COM1, COM2 and COM3. User should verify that the correct port was set and check the field "Communication Port" on the LC ONLINE window.

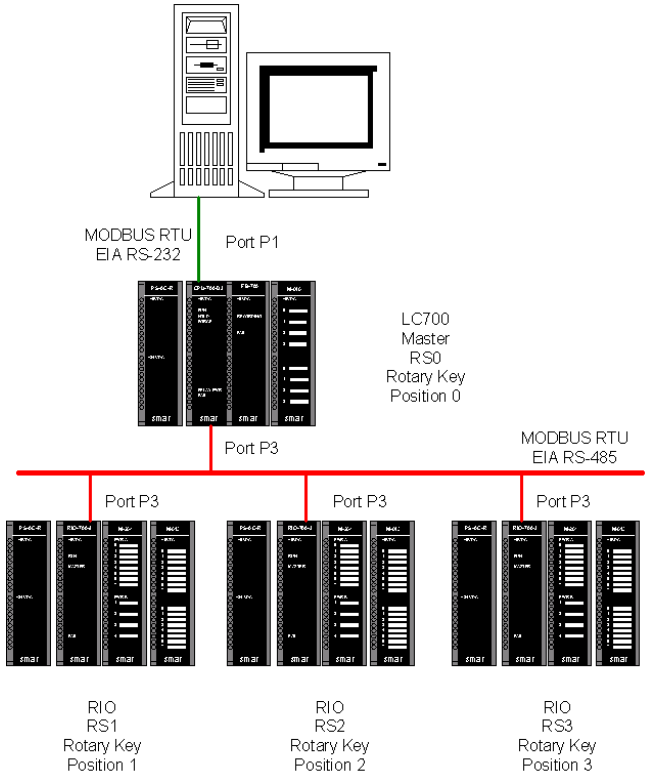
STEP5: Go to menu Tools> Comm. Settings and select Option RS-232. Assure that CTS/RTS Timeout parameter is set at 0.

5. There is a failure during communication between the CPU master and the remote CPU.

Solution:

STEP1: Check the Rotary Keys of the remote CPUs. They must be in the position equivalent to the their position within the configuration. If user has set 3 RIOs and named them RS1, RS2 and RS3, the rotary keys must be set as 1,2 and 3 respectively to each remote unit. The Master CPU requires the rotary key placed in the zero position.

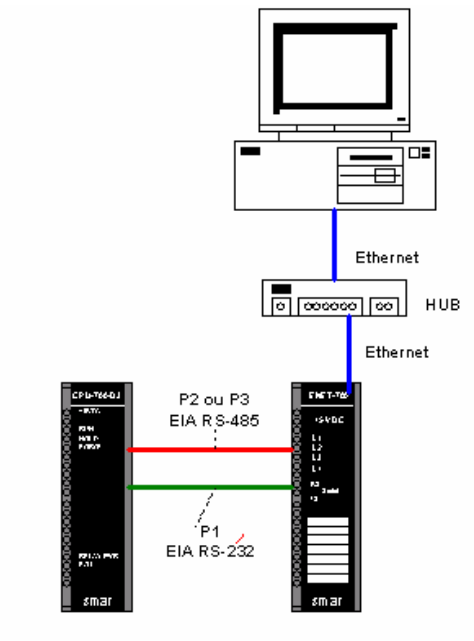
STEP2: DIP SWITCHES of each CPU (master and RIO) should be placed in the same position. Both RIOs and the main CPU must have equal communication parameters of the P3 port. Communication is through the P3 port of each CPU (serial interface EIA-RS-232).



6. Communication through Ethernet between CON700 and CPU fails. After starting communications and clicking on "look" I can't establish communication.

Solution:
STEP1: Connect the CPU and the PC through the LC700 P1 port setting the baudrate of the P2 port (or P3) and communication parameters (Comm. Parameters). In a network make sure to attribute different IDs to the CPU and other elements of this network.

STEP2: Set the ENET-700 and ENET-710 communication parameters. We advise that the ENET-700's baudrate be set as 19200 bps and the ENET-710's baudrate be set as 115200 bps. Connect the CPU and the ENET-700/ENET-710 through ports P1 or P2 or P3. (As shown below)



Certify that the CPU Dip Switch is not in the default position.

STEP3: On CONF700, on menu Tools>Comm. Param select option Ethernet (Modbus TCP/IP) and type the ENET700 IP address in the Ethernet network.

7. Communication between the CONF700 and the CPU through the modem fails. After starting communication and clicking on "Look", I can't establish communication.

Solution:
STEP1: Connect workstation and CPU through P1 port. Set CPU700 communication parameters. RTS/CTA Timeout and Baudrate.

STEP2: Be sure the DIP Switch is in the default position.

STEP3: On CONF700's menu, Tools→ Comm. Param. Select RS232 option and set values of RTS/CTS Timeout (they don't need to be equal to the ones in the CPU). CONF700 communication parameters must be the same as the parameters set in STEP1.

8. CPU does not communicate with M-402 or modules in the racks.

Solution:
STEP1: Check the key in the racks to be sure it is placed in the same position as displayed in the CONF700. The key in the rack is located in the back of the CPU module. Be sure the number there is the same as in the established configuration on the CONF700. For ex: if we have two racks, the first receives the number zero (0) on the CONF700. When another rack is added, the CONF700 attributes a number to this rack according to the user's selection. This number must be the same as the keys in the racks as indicated.

STEP2: Check connection of flat cables between racks.

Appendix A

smar	FSR – Service Request Form	
	LC700 – Programmable Controller	Proposal Nº:

COMPANY INFORMATION

Company: _____
Unit: _____
Invoice: _____

COMMERCIAL CONTACT

Full Name: _____
Phone: _____ _____ _____ _____ **Fax:** _____
E-mail: _____

TECHNICAL CONTACT

Full Name: _____
Phone: _____ _____ _____ _____ **Extension:** _____
E-mail: _____

EQUIPMENT DATA

Model: _____
Serial Number: _____

PROCESS DATA

Process Type (Ex. boiler control)**:** _____
Operation Time: _____
Failure Date: _____

FAILURE DESCRIPTON

(Please, describe the failure, if it is repetitive, how it reproduces, etc.)

_____
_____
_____
_____

OBSERVATIONS

_____
_____
_____
_____

USER INFORMATION

Company: _____
Contact: _____
Section: _____
Title: _____ **Signature:** _____
Phone: _____ _____ _____ _____ **Extension:** _____
E-mail: _____ **Date:** _____/ _____/ _____

SMAR WARRANTY CERTIFICATE

1. SMAR guarantees its products for a period of 24 (twenty four) months, starting on the day of issuance of the invoice. The guarantee is valid regardless of the day that the product was installed.

2. SMAR products are guaranteed against any defect originating from manufacturing, mounting, whether of a material or manpower nature, provided that the technical analysis reveals the existence of a quality failure liable to be classified under the meaning of the word, duly verified by the technical team within the warranty terms.

3. Exceptions are proven cases of inappropriate use, wrong handling or lack of basic maintenance compliant to the equipment manual provisions. SMAR does not guarantee any defect or damage caused by an uncontrolled situation, including but not limited to negligence, user imprudence or negligence, natural forces, wars or civil unrest, accidents, inadequate transportation or packaging due to the user's responsibility, defects caused by fire, theft or stray shipment, improper electric voltage or power source connection, electric surges, violations, modifications not described on the instructions manual, and/or if the serial number was altered or removed, substitution of parts, adjustments or repairs carried out by non-authorized personnel; inappropriate product use and/or application that cause corrosion, risks or deformation on the product, damages on parts or components, inadequate cleaning with incompatible chemical products, solvent and abrasive products incompatible with construction materials, chemical or electrolytic influences, parts and components susceptible to decay from regular use, use of equipment beyond operational limits (temperature, humidity, etc.) according to the instructions manual. In addition, this Warranty Certificate excludes expenses with transportation, freight, insurance, all of which are the customer's responsibility.

4. For warranty or non-warranty repair, please contact your representative.

   Further information about address and contacts can be found on www.smar.com/contactus.asp

5. In cases needing technical assistance at the customer's facilities during the warranty period, the hours effectively worked will not be billed, although SMAR shall be reimbursed from the service technician's transportation, meals and lodging expenses, as well dismounting/mounting costs, if any.

6. The repair and/or substitution of defective parts do not extend, under any circumstance, the original warranty term, unless this extension is granted and communicated in writing by SMAR.

7. No Collaborator, Representative or any third party has the right, on SMAR's behalf, to grant warranty or assume some responsibility for SMAR products. If any warranty would be granted or assumed without SMAR's written consent, it will be declared void beforehand.

8. Cases of Extended Warranty acquisition must be negotiated with and documented by SMAR.

9. If necessary to return the equipment or product for repair or analysis, contact us.
   See item 4.

10. In cases of repair or analysis, the customer must fill out the Revision Requisition Form (FSR) included in the instructions manual, which contains details on the failure observed on the field, the circumstances it occurred, in addition to information on the installation site and process conditions. Equipments and products excluded from the warranty clauses must be approved by the client prior to the service execution.

11. In cases of repairs, the client shall be responsible for the proper product packaging and SMAR will not cover any damage occurred in shipment.

12. In cases of repairs under warranty, recall or outside warranty, the client is responsible for the correct packaging and packing and SMAR shall not cover any damage caused during transportation. Service expenses or any costs related to installing and uninstalling the product are the client´s sole responsibility and SMAR does not assume any accountability before the buyer.

13. It is the customer's responsibility to clean and decontaminate products and accessories prior to shipping them for repair, and SMAR and its dealer reserve themselves the right to refuse the service in cases not compliant to those conditions. It is the customer's responsibility to tell SMAR and its dealer when the product was utilized in applications that contaminate the equipment with harmful products during its handling and repair. Any other damages, consequences, indemnity claims, expenses and other costs caused by the lack of decontamination will be attributed to the client. Kindly, fill out the Declaration of Decontamination prior to shipping products to SMAR or its dealers, which can be accessed at www.smar.com/doc/declarationofcontamination.pdf and include in the packaging.

14. This warranty certificate is valid only when accompanying the purchase invoice.