

# **LadderDIP PLC Studio**

© 1995-2018 MicroSHADOW Research

# **LadderDIP PLC Studio**

---

# LadderDIP PLC Studio

© 1995-2018 MicroSHADOW Research

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: gennaio 2018 in La Spezia (Italy)

## **Publisher**

*MicroSHADOW Research*

# Table of Contents

Foreword	I
<b>Part I Main index</b>	<b>2</b>
<b>Part II Before You Start</b>	<b>5</b>
<b>Part III What's new</b>	<b>7</b>
<b>Part IV Copyright notice</b>	<b>10</b>
<b>Part V About this document</b>	<b>14</b>
<b>Part VI LadderDIP IV General Informations</b>	<b>17</b>
1 What is LadderDIP IV? .....	17
2 LadderDIP IV Features and benefits .....	18
3 Module Pinout .....	19
4 DC/AC Parameters and absolute max ratings .....	19
5 I/O Memory Mapping .....	20
6 System Memory Mapping .....	21
7 LadderDIP IV FAQ(s) .....	22
8 LadderDIP IV Firmware Upgrading .....	22
<b>Part VII Getting Started with LadderDIP IV</b>	<b>30</b>
1 LadderDIP IV IMPORTANT NOTES .....	30
2 What i need to work with LadderDIP IV .....	31
3 My basic LadderDIP IV connection .....	32
4 Connecting and powering my module .....	33
5 Checking connection .....	34
6 Flashing my first LED in 1 minute .....	35
7 Pin Configuration .....	38
8 Using the AD (Analog to Digital) .....	39
9 Using the DAC .....	40
10 Using the PWM (Pulse Width Modulation) .....	40
<b>Part VIII Getting Started with LadderDIP IV Development Board</b>	<b>42</b>
1 What is LadderDIP IV Development Board .....	42
2 What i need to work with LadderDIP IV Development Board .....	43
3 Power up LadderDIP IV Development Board .....	44
4 Using the "RELAY" outputs .....	45



5 Using the "INSULATED" inputs .....	47
6 Using the Analog Input (0-10V) .....	50
7 Using the On-Board Trimmer .....	52
8 Using the Analog Output (DAC) (0-10V) .....	54
9 LadderDIP IV Development Board parameters .....	56
<b>Part IX Introduction to PLC .....</b>	<b>58</b>
1 PLC Overview .....	58
2 Run-time environment .....	58
3 Input read and output write scan .....	58
4 Housekeeping .....	59
5 Program scan .....	59
6 CEI / IEC 1131-3 programming languages .....	59
7 Ladder LOGIC .....	60
8 Ladder logic's elements .....	61
<b>Part X LadderDIP PLC Studio Software .....</b>	<b>63</b>
1 Basic topics .....	63
Understanding the IEC 1131-3 / CEI 61131-3 addressing .....	63
Defining variables .....	64
Logical links .....	66
Identifiers .....	67
Literals .....	68
The REFERENCE Code .....	70
Mathematical expressions .....	71
2 LadderDIP PLC Studio Tutorial .....	72
What LadderDIP PLC Studio is .....	72
How does LadderDIP PLC Studio work? .....	72
Basic diagram elements .....	73
A first project, the basic I/O Transfer .....	74
And now, blink the first led in 1 minute .....	77
A more complex project, a basic counter .....	78
The "Walking One" (Led Chaser) project .....	80
Analog threshold with hysteresis .....	81
3 Integrated Development Enviroment (IDE) .....	83
System requirements .....	84
Installing the software .....	84
Dongle Protection Key .....	84
Launch the program .....	84
Integrated development enviroment .....	85
The menu .....	86
File menu .....	87
Edit menu .....	88
Build menu .....	89
View menu .....	90
Zoom menu .....	91
Options menu .....	91
Placing and modifying components .....	92

Connetting components .....	93
Setting components properties .....	94
Building the code .....	95
Uploading the code .....	95
Running and stopping the PLC .....	96
Compiling and debugging your project .....	96
Changing On-The-Fly Values .....	98
Adding watches .....	101
Using help .....	102
<b>4 Library .....</b>	<b>102</b>
Library conventions .....	103
AD_CONV .....	104
ADD .....	105
AND .....	106
ASSIGN .....	107
ASSIGN32 .....	108
BIT .....	109
BIT32 .....	110
BITS .....	111
BITSR .....	112
BITR .....	113
CLOCK .....	114
CMP_W .....	115
CONST .....	117
COUNTER .....	118
CTD .....	119
CTU .....	120
CTUD .....	121
DEBOUNCE .....	123
DEC1-8 .....	124
DELAY .....	125
DIV .....	127
EINPUT .....	128
ENCINPUT .....	129
EOUTPUT .....	130
F_TRIG .....	131
FFD .....	132
HBYTE .....	133
IDENT .....	134
IDENT32 .....	135
INPUT .....	136
INPUT_N .....	137
INPUT_P .....	138
LBYTE .....	139
LIMIT .....	140
MAX .....	141
MEMWR16 .....	142
MEMWR32 .....	143
MIN .....	144
MKWORD .....	145
MOD .....	146
MUL .....	147
MUX .....	148
NCINPUT .....	149
NOT .....	150

OR .....	151
OUTPUT .....	152
OUTPUT_I .....	153
OUTPUT_N .....	154
OUTPUT_P .....	155
OUTPUT_R .....	156
OUTPUT_S .....	157
PWMOUT .....	158
R_TRIG .....	159
READVAR .....	160
RDVAR32 .....	161
RELAY .....	162
ROL .....	163
ROR .....	164
RS .....	165
SCALE_W .....	166
SEL .....	169
SEMA .....	170
SHL .....	171
SHR .....	172
SR .....	173
SUB .....	174
THRESHLD .....	175
TMI .....	177
TOF .....	178
TON .....	179
TP .....	180
TSQ .....	181
WRE16 .....	182
WRE32 .....	183
XOR .....	184
<b>5 Error messages .....</b>	<b>185</b>
<b>Ladder/FBD NET errors .....</b>	<b>185</b>
NET0706.....	185
NET0707.....	186
NET0708.....	186
NET0709.....	187
NET0720.....	187
NET0710.....	188
NET0754.....	188
NET0757.....	189
NET0758.....	189
NET0760.....	190
NET0794.....	190
NET0795.....	190
NET0796.....	191
NET0797.....	191
NET0798.....	192
NET0802.....	192
NET0803.....	193
NET0804.....	193

## Part XI MODBUS/UDP 195

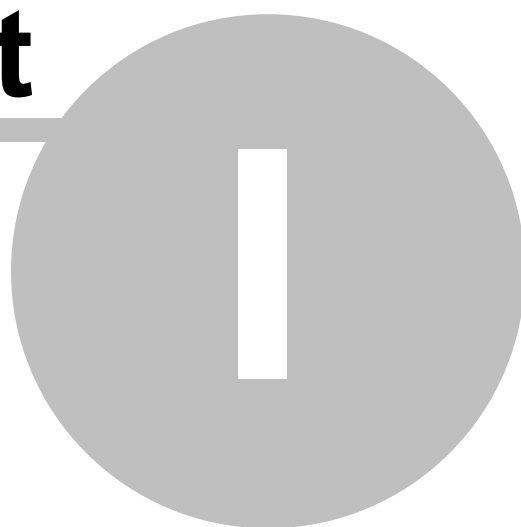
<b>1 What is MODBUS/UDP .....</b>	<b>195</b>
-----------------------------------	------------

2	MODBUS/UDP Packet Format .....	195
3	MBAP Block .....	196
4	MOSBUS Function Codes .....	198
	Read Holding Registers (Code = 0x03) .....	198
	Write Mutiple Registers (Code = 0x10) .....	200
<b>Part XII MODBUS RTU (RS232/RS485)</b>		<b>203</b>
1	WHAT IS MODBUS RTU .....	203
	MODBUS RTU Function Codes .....	203
	Read Holding Registers (Code = 0x03).....	203
	Write Mutiple Registers (Code = 0x10).....	205
	MODBUS RTU With LadderDIP IV .....	206
	<b>Index</b>	<b>208</b>

# Foreword

# Part

---



# 1 Main index

## LadderDIP PLC Studio



IMPORTANT: Before you start **ANY** operation with LadderDIP IV carefully follow the notes here [LadderDIP IV IMPORTANT NOTES](#)

- [Before You Start](#)
- [Copyright notice](#)
- [About this document](#)
- [Getting Started with LadderDIP IV](#)
- [LadderDIP IV Informations](#)
- [Introduction to PLC](#)
- [Integrated Development Environment](#)
- [Tutorial](#)
- [Basic topics](#)
- [Library](#)
- [MODBUS/UDP](#)
- [Error messages](#)

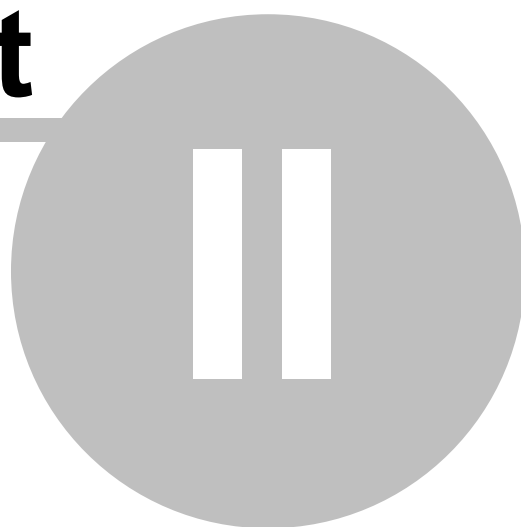
*Document update : Dec 28, 2017*





# Part

---



## 2 Before You Start

Before you start with LadderDIP IV and LadderDIP PLC Studio you have to keep in mind the following important things

### 1) WHAT ARE WE TALKING ABOUT?

LadderDIP IV and LadderDIP PLC Studio, together, form a LADDER programmable PLC (Programmable Logic Controller) system. The LadderDIP IV is supplied in the form of an electronic module that requires just a power supply and gives a general purpose set of TTL (3.0V) I/O for your specific application. The module also add some important features like A/D channels, D/A, PWM, RTC and non volatile memory.

### 2) LADDER LANGUAGE IS NOT "C"

Many people tend to think to LADDER language like an alternative to the "C" language. **This is completely wrong**. The tool you are going to use (and probably purchase) is absolutely different from any other traditional script languages you know (C, Basic, Java, Phyton, Forth .. etc ... there are tons of those)

There are huge differences between the two environments. The "C" language, for example, allow you to start a project "from the scratch", having almost 360 degrees of flexibility in supporting and handling your embedded system. By the other hand "C" requires a good processor knowledge and a deep programming attitude. LADDER language is mostly used in automation field and is hugely preferred by electricians and installators. Essentially with the LADDER language you don't have to know anything about processor architecture, interrupts, procedures and so on. With the LADDER language you simply place functional objects in your worksheet, you connect the components with wires and configure the components properties. All the rest is done by the software.

So, definitely, consider the LADDER language the best way to obtain, quickly, automation control applications. If you need to play with a different level of microprocessor resources, convince yourself, LadderDIP PLC Studio is not your tool! :)

### 3) LadderDIP IV POINTS OF FORCE

LadderDIP IV is the perfect product if you want to integrate a complete PLC core into your embedded products. LadderDIP IV consumes few PCB space, it has features comparable to high-end branded PLC but it cost less. LadderDIP IV concept raised up with the necessity to have an easy to program LADDER environment suitable to be embedded in different electronics boards and applications. With the using of LadderDIP IV you save tons of money related to the developing of the visual interface and you immediately get a running PLC in your product

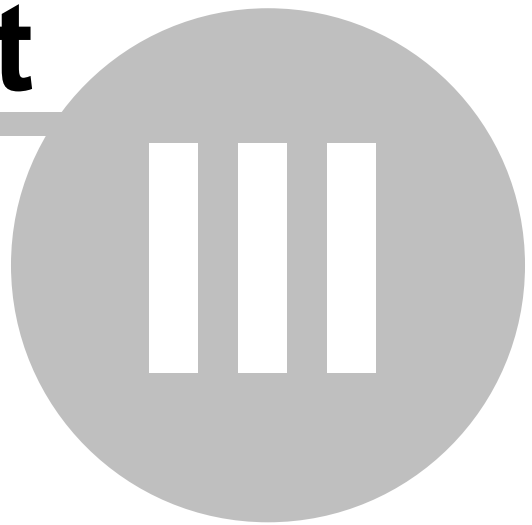
### 4) WHICH ARE THE MAIN APPLICATION AREAS WHERE LadderDIP IV WAS INTEGRATED?

LadderDIP IV was happily used in the following application fields

- General PLC applications
- Building/Home Automation
- A/C systems
- Integration in RTU (Remote Terminal Units) giving the possibility to have a complete remote PLC device
- Perfect for hobby markets, students, universities

# Part

---



## 3 What's new

### ABOUT LadderDIP IV

LadderDIP is an innovative controller module based on an ARM(R) 100MHz processor with 100 MIPS (Million Instructions Per Second) peak performance.

This core module will allow you to build an **IoT-Ready** (Internet-Of-Things) complete **PLC** (Programmable Logic Control) system to fit your companies needs using the remarkable power and simplicity of our LADDER language as the PLC programming language.

LadderDIP IV comes with our most versatile LADDER development environment, LadderDIP PLC Studio.

What this means to you is that you don't need to know anything at the microprocessor level such as assembler, interrupts or hardware architectures. This is all done for you saving you time and money. All you have to do is program the PLC in the familiar LADDER electrical scheme incorporating switches as inputs, relays (memory functions) and lamps as outputs.

Finally, a workable solution to control automation realized in a few thoughtful minutes with LadderDIP.

LadderDIP IV is packaged in a 100 mils (2.54mm) 60-pins socket giving up the following resources

- 27 bits of general purpose I/O
- 6 A/D channels (10 bits)
- 1 DAC channel (10 bits)
- 2 PWM(s) channels
- Precision RTC
- 4Kbit of ferro-electric NVM
- Native 100Mbit Ethernet Interface
- MODBUS/UDP
- 32K of user Ladder program
- 4 KWords of user memory



**Part**

---

**IV**

## 4 Copyright notice

# LICENSE AGREEMENT

Carefully read all the terms and conditions of this agreement prior to using the LadderDIP PLC Studio software. **By breaking the media envelope or installing this software you indicate your acceptance of these terms and conditions.**

### COPYRIGHT

Copyright 1997-2017 MicroSHADOW Research, all rights reserved.

MicroSHADOW Research(R) LOGO, LadderWORK(R), LadderDIP(R) LOGO, and LadderWORK ARM LOGO are registered marks (R) of MicroSHADOW Research

The program and its related documentation are copyrighted. The user may not copy the program or its documentation except for your own back-up purposes and load the program into the computer as part of executing the program. All other copies of the program and its documentation are in violation of this agreement. No part of the manual may be photocopied or reproduced in any form or electronic media without prior written authorization from MicroSHADOW Research.

### SINGLE USER LICENSE AGREEMENT

License : The user has the non-exclusive right to use the enclosed program on a single CPU or personal computer. The user may physically transfer the program from a PC to another provided that the program is used on only CPU at time. The user may not electronically transfer the program from one PC to another over a network. The user may not distribute copies of the program or related documentation to others. The user may not modify or translate the program or related documentation without the prior written consent of MicroSHADOW Research. The user may not use, copy, modify or transfer the program or documentation, or any copy thereof, or permit anyone else to do so, except as expressly provided in this agreement.

Back-up and transfer : The user may make one ( 1 ) copy of the program solely for own back-up purposes. The user must reproduce and include the copyright notice on the back-up copy. Transfer of program and license to another party may only be made after written approval from MicroSHADOW Research, provided the other party agrees to the terms and conditions of this agreement and completes and returns a product registration form to MicroSHADOW Research. If the user transfer the program, at the same time he must transfer the documentation and back-up copies or transfer the documentation and destroy the back-up copies.

Upgrades : LadderWORK ARM software is sold without warranty. Furthermore MicroSHADOW Research reserve the right to make changes to any products herein to improve reliability, functionality and performance. Customer can download software upgrades directly from the web according to its hardware key programmed version. Normally the hardware key, sold with the software, can accept major software number next to the purchased issue. Free demo version available from the web has to be intended for evaluation only and gives no warranty of any kind.

Media defects Warranty : MicroSHADOW Research warrants to the original licensee that the media ( diskettes, cd-rom or others ) which the program is recorded be free from defects in materials and workmanship under normal use and a free substituting service is available for a

period of 90 days from the date of delivery, accompanied by a copy of the purchase invoice.

### **DISCLOSURE**

The informations contained in the manuals ( traditional or electronic medias ) are subject to change without prior notice. MicroSHADOW Research assumes no responsibility or liability for any errors or inaccuracies ( technical or editorial ).

The program is supplied "as is" without warranty of any kind. The entire risk as to quality and performance of the program is charged to the user.

MicroSHADOW Research assumes no liability for any damages resulting from defects, software bugs or design's solutions of LadderWORK ARM software. It makes no warranty of any kind ( express, implied or statutory ) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes and noninfringement of third part rights.

In no event shall MicroSHADOW Research, its directors, officers, engineers, employees or agents be liable for any indirect, special, incidental or consequential damages ( including damages for loss of profits, loss of business, loss of use or data, interruption of business an the like ), even if MicroSHADOW Research has been advised of the possibility of such damages arising from any defects or error in the documentation or software.

LadderWORK ARM software can't be used for medical instrumentation or life-support equipment.

Any referement to Corporations, names and data used in the screen's reproduction are purely casuals and it has to be intended as tutorial purpose only. Product and corporate names appearing in the documentation may or may not be registered trademarks or copyrights of their respective companies, and are used only for identification or explanation and to the owner's benefit, without intent to infringe.

### **MISCELLANEOUS**

This license agreement shall be governed by the laws of ITALY and shall insure to the benefit of MicroSHADOW Research.





# Part

---




V

## 5 About this document

### Conventions used in this document

<b>LD</b>	These initials stays for <b>Ladder Diagram</b>
<b>PLC</b>	These initials stays for <b>P</b> rogrammable <b>L</b> ogic <b>C</b> ontroller

The following conventions are used in this section :

<b>bold</b>	Bold text denotes menus, menu items, or dialog box buttons or options. In addition bold text denotes <b>LD</b> input and output parameters.
<i>Italic</i>	Italic text denotes emphasis, a cross reference, or an introduction to a key concept.
<b><i>Bold italic</i></b>	Bold italic text denotes a note, caution or warning.
Courier	Courier font denotes text or characters that you enter using keyboard. Selections of code, programming examples, syntax examples, and messages and responses that the computer automatically prints to the screen also appear in this font.
Courier Bold	Courier Bold denotes file names
→	This symbol leads you through nested menu items and dialog box options to a final action. The sequence <b>Options → Colors</b> directs you to pull down the Options menu and select the Color option entering the appropriate dialog box.
	Very important notes or informations are signaled using this icon



# Part

---



VI

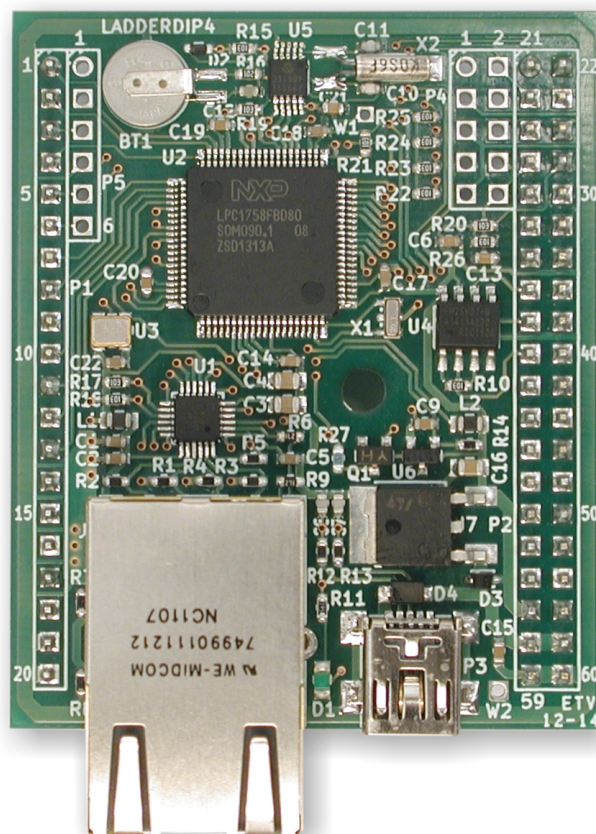
## 6 LadderDIP IV General Informations

This section introduces to you the LadderDIP IV product

### 6.1 What is LadderDIP IV?

LadderDIP IV is an OEM Embedded PLC (Programmable Logic Controller) module based on a 32 bit processor core. Compared with a regular PLC product, LadderDIP PLC Studio allow you to build your own PLC and IoT (Internet of Things) products fitting it on a simple PCB board. LadderDIP IV works using 3.3V logic levels so you can design your electrical interfaces (Relays, Opto, NPN .. and so on) according to your specific requirements.

This is a picture of the module



## 6.2 LadderDIP IV Features and benefits

### FEATURES

LadderDIP PLC Studio has the following features

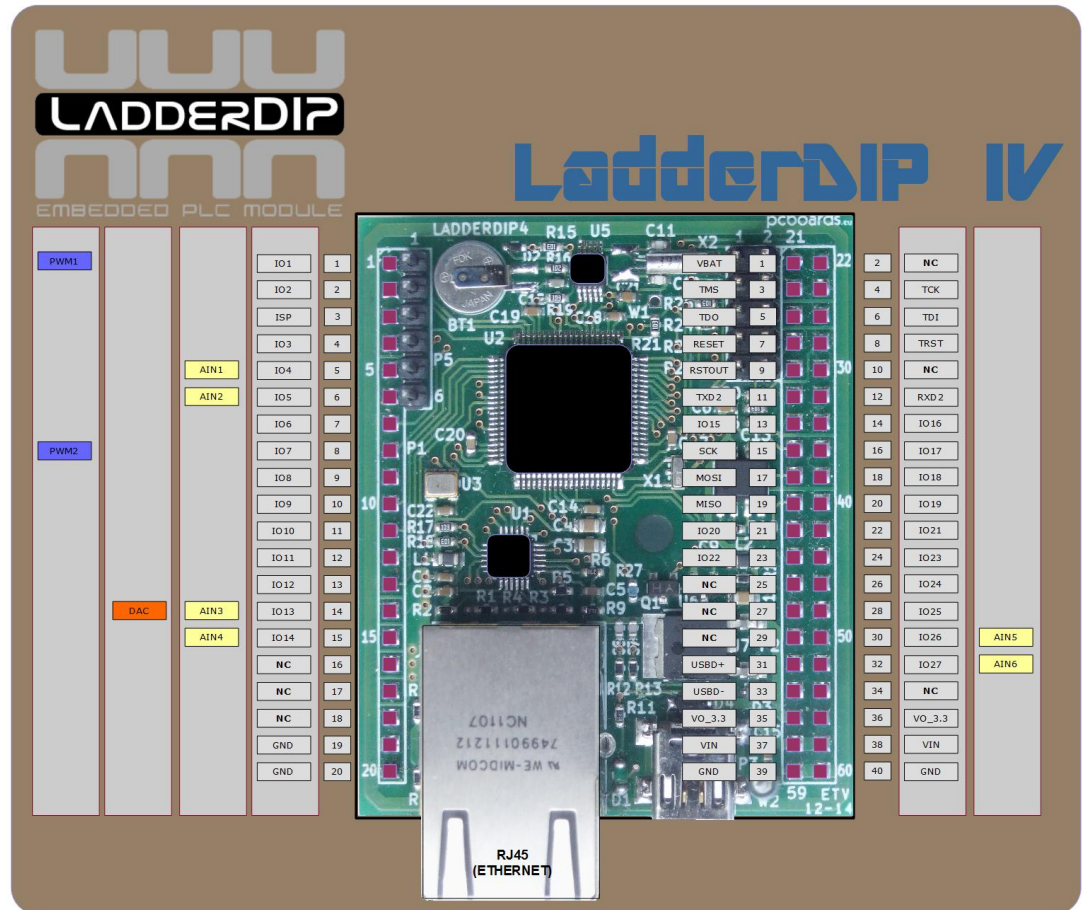
- Ladder / FBD / Relay Logic programmable Embedded module
- 100 Mhz 32 bit ARM (Cortex M3) core
- Integrated 10/100 Mbit Ethernet interface (RJ45)
- 27 general purpose I/O
- 6 analog inputs (12 bits resolution)
- General purpose communication UART
- High precision RTC (Real Time Clock) with backup battery
- 4 KBytes of NVM (Non Volatile Memory / Retentive memory)
- Wide input power supply voltage range (5-20 VDC)
- JTAG Interface
- ISP Interface

### BENEFITS

- Allow to build your own PLC basing your project on this powerful core module
- Allow you to get an IoT (Internet of Things) product ready at once
- Allow to design your own field interfaces (Relay/Opto...etc) without any restriction. The module works with TTL 3.0V levels
- Ready-to-use Ladder interface running on a Windows(R) PC
- Ladder interface allow your product to be programmed by your customers (No script languages like C/C++/Basic ... etc)
- Native 100 Mbit Ethernet interface enhance your connectivity
- Native MODBUS/UDP enable you to control the module remotely
- Modern 32 bits 100 Mhz core architecture give you all the necessary computation power

## 6.3 Module Pinout

Here is shown the module Pinout



## 6.4 DC/AC Parameters and absolute max ratings



LadderDIP IV is an OEM Embedded Module that should be integrated in your hardware. **Carefully**, when interfacing the module, take in consideration the following voltage and current max ratings.



Symbol	Parameter	Min Value	Max Value	Unit	Notes
V <sub>IN</sub>	Power supply voltage (VIN Pin)	5	9	V	Exceeding this value will turn in a permanent device damage
	Power supply current (VIN Pin)				
V <sub>IH</sub>	HIGH level Input voltage	2.31	3.3	V	
V <sub>IL</sub>	LOW level Input voltage	0	0.95	V	
I <sub>IH</sub>	HIGH level input current	10		nA	
I <sub>IL</sub>	LOW level input current	10	-	nA	
V <sub>OH</sub>	HIGH level output voltage	2.9	-	V	
V <sub>OL</sub>	LOW level output voltage	-	0.4	V	
I <sub>OUT</sub>	Output current (High or Low)	4		mA	
I <sub>SHORT</sub>	Output short breakdown current		40	mA	Exceeding this value will turn in a permanent device damage
V <sub>AMAX</sub>	Max input voltage when a pin is used as analog input		3	V	Exceeding this value will turn in a permanent device damage
V <sub>AMIN</sub>	Min input voltage when a pin is used as analog input	0		V	
tw	Temperature Range (Working Range)	0	45	°C	

## 6.5 I/O Memory Mapping

LadderDIP IV has the following memory mapping

**Note that, when used as I/O pins, the address changes according to the configured direction using two different memory areas.**

Pin Name	Location (Pinout)	Primary Function	Secondary Function	Address when I/O is configured as INPUT	Address when I/O is configured as OUTPUT	
IO1	P1-1	I/O		%IX0.0	%QX80.0	
IO2	P1-2	I/O		%IX0.1	%QX80.1	
IO3	P1-4	I/O		%IX0.2	%QX80.2	

IO4	P1-5	I/O	AIN1	%IX0.3	%QX80.3
IO5	P1-6	I/O	AIN2	%IX0.4	%QX80.4
IO6	P1-7	I/O		%IX0.5	%QX80.5
IO7	P1-8	I/O		%IX0.6	%QX80.6
IO8	P1-9	I/O		%IX0.7	%QX80.7
IO9	P1-10	I/O		%IX0.8	%QX80.8
IO10	P1-11	I/O		%IX0.9	%QX80.9
IO11	P1-12	I/O		%IX0.10	%QX80.10
IO12	P1-13	I/O		%IX0.11	%QX80.11
IO13	P1-14	I/O		%IX0.12	%QX80.12
IO14	P1-15	I/O	AIN3	%IX0.13	%QX80.13
IO15	P2-13	I/O	AIN4	%IX0.14	%QX80.14
IO16	P2-14	I/O		%IX0.15	%QX80.15
IO17	P2-16	I/O		%IX1.0	%QX81.0
IO18	P2-18	I/O		%IX1.1	%QX81.1
IO19	P2-20	I/O		%IX1.2	%QX81.2
IO20	P2-21	I/O		%IX1.3	%QX81.3
IO21	P2-22	I/O		%IX1.4	%QX81.4
IO22	P2-23	I/O		%IX1.5	%QX81.5
IO23	P2-24	I/O		%IX1.6	%QX81.6
IO24	P2-26	I/O		%IX1.7	%QX81.7
IO25	P2-28	I/O	AIN5	%IX1.8	%QX81.8
IO26	P2-30	I/O	AIN6	%IX1.9	%QX81.9
IO27	P2-32	I/O		%IX1.10	%QX81.10

## 6.6 System Memory Mapping

LadderDIP IV has the following system memory mapping  
User diagrams should use the areas classified as "USER" only

Area	Start Address	End Address	Number of words	Class
I/O	%MW0	%MW199	200	RESERVED
System	%MW200	%MW599	400	RESERVED
Retentive (User)	%MW600	%MW619	20	USER
System	%MW800	%MW999	200	RESERVED
User Data	%MW1000	%MW1999	1000	USER
Retentive (System)	%MW2000	%MW2079	80	RESERVED
System	%MW2080	%MW2199	120	RESERVED
User Data	%MW2200	%MW3999	1800	USER

## 6.7 LadderDIP IV FAQ(s)

Here you can find the FAQ(s) (Frequently Asked Questions) for LadderDIP IV product

**Q1) How can i read and write the WORD TABLE from an host system?**

**A1) LadderDIP IV integrates the MODBUS/UDP protocol. It is available on the native ethernet port of the module. See the [MODBUS/UDP](#) section on this same document**

## 6.8 LadderDIP IV Firmware Upgrading

### LadderDIP IV Firmware Upgrading Procedure

PLEASE CAREFULLY FOLLOW ALL THE INSTRUCTION TO UPGRADE YOUR LadderDIP IV FIRMWARE

#### WHAT I NEED TO EXECUTE A FIRMWARE UPGRADE

To upgrade the firmware i need

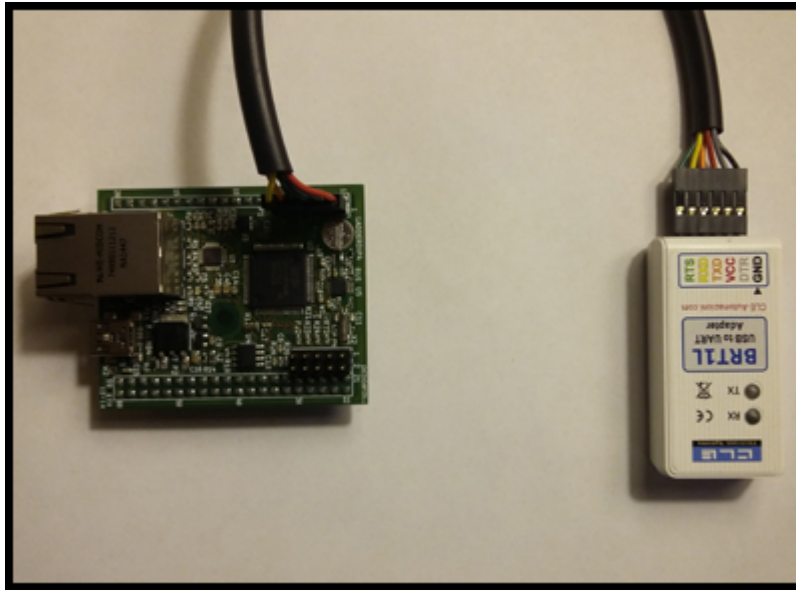
- FlashMagic software (<http://www.flashmagictool.com/>)
- BRT1L USB/ISP Adapter (FTDI driver needed)
- The updated BIN/HEX firmware file (Supplied by LadderDIP IV Manufacturer)



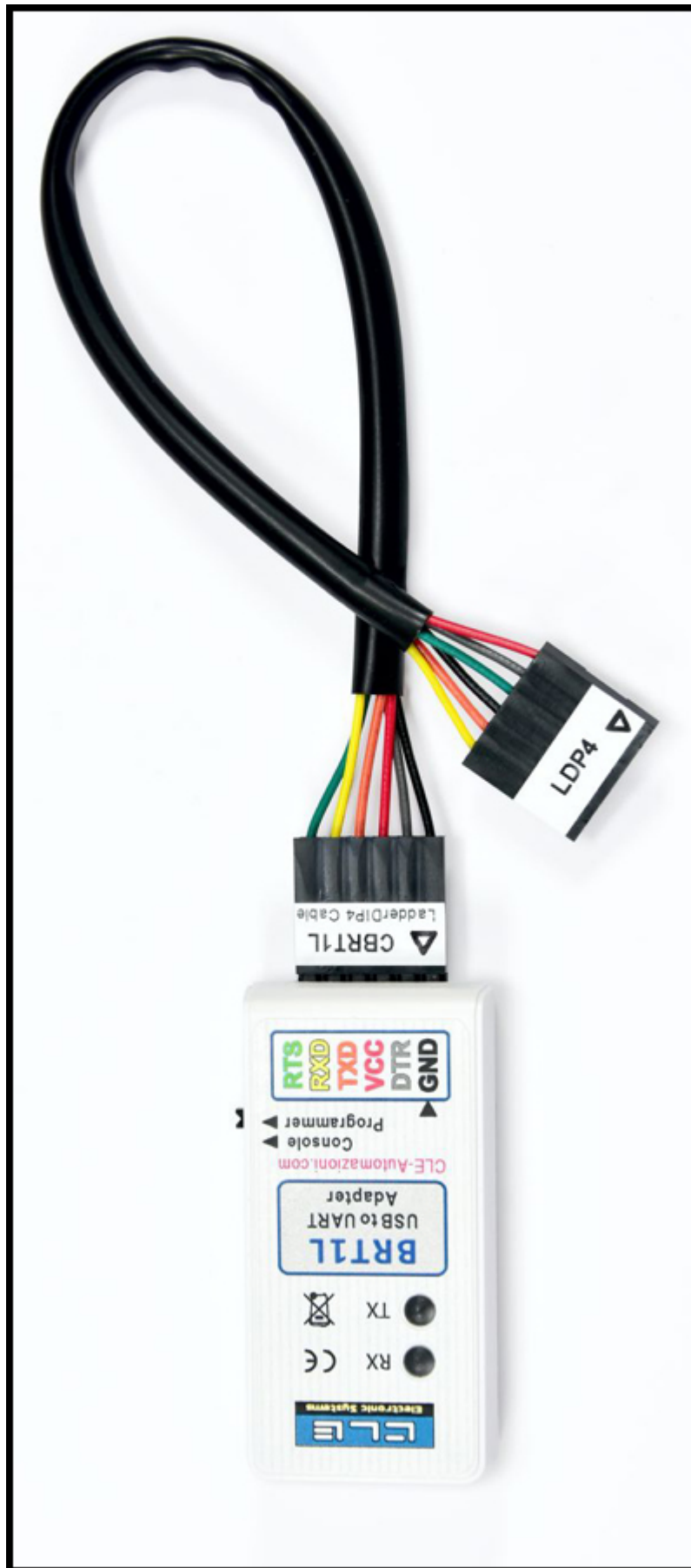
#### VERY IMPORTANT

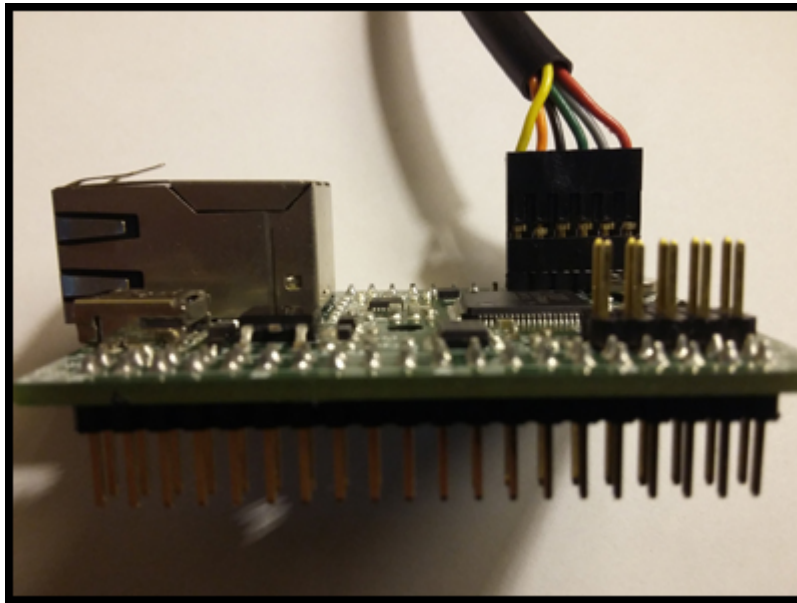
**The firmware update MUST be executed with the LadderDIP IV module completely disconnected by any other cable or board. ONLY THE BRT1L ADAPTER MUST be attached to the LadderDIP IV module**

**STEP 1) CONNECT THE MODULE TO THE BRT1L ADAPTER LIKE SHOWN IN THE PICTURES**



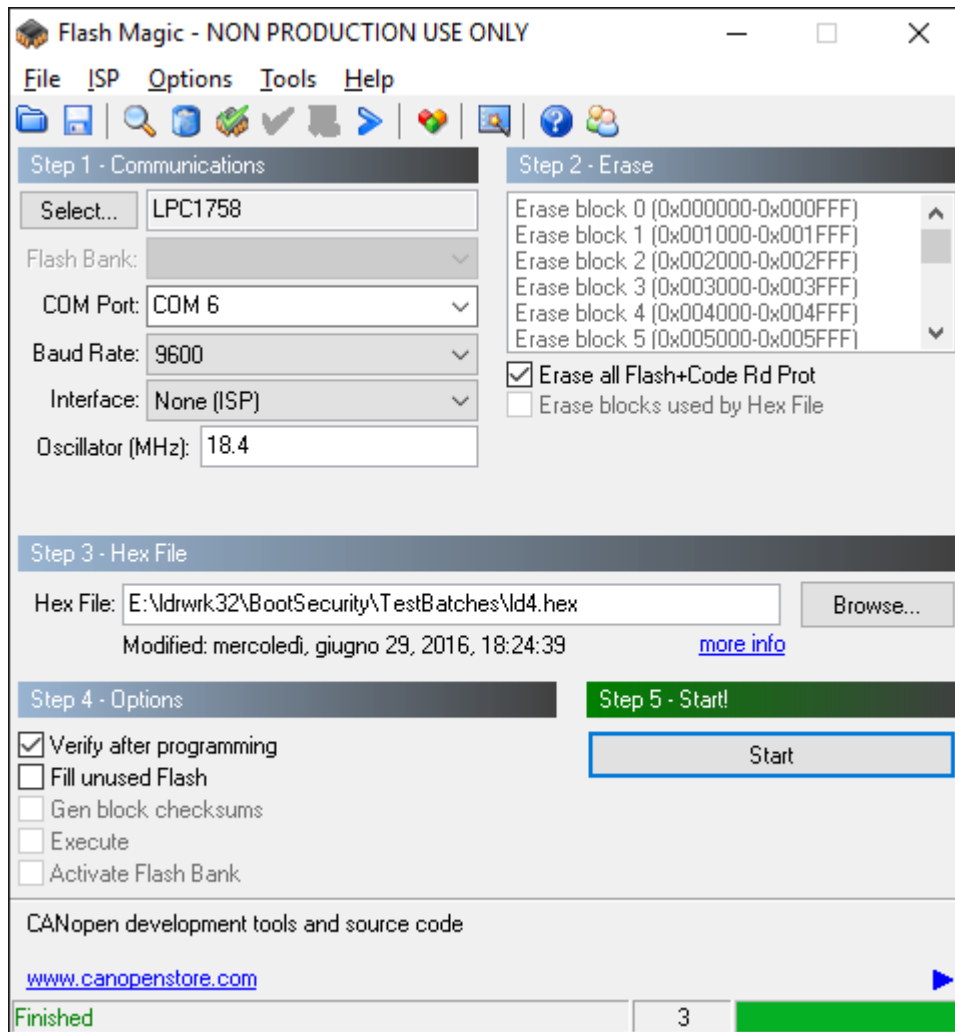
**PLEASE RESPECT PINS COLORS AND POLARITIES**





**STEP 2) USE FLASHMAGIC TO BURN THE FIRMWARE INTO THE MODULE**

Launch the FlashMagic software and configure as indicated here



Configure FlashMagic as indicated here

- Select the "LPC1758" part
- Select Interface = ISP
- Select Oscillator = 18.4Mhz
- Valid baud rates are 9600 up to 115200
- Select the port related to the BRT1L adapter
- Check the "Erase all Flash+Code Rd Prot"
- Check "Verify after programming"
- Enter the path of the firmware hex file

The LadderDIP IV module is powered by the adapter, YOU DO NOT and YOU DO NOT HAVE TO connect any other cable. The module MUST be ALONE

Once you configured you can press the <Start> button and wait for a successful programming

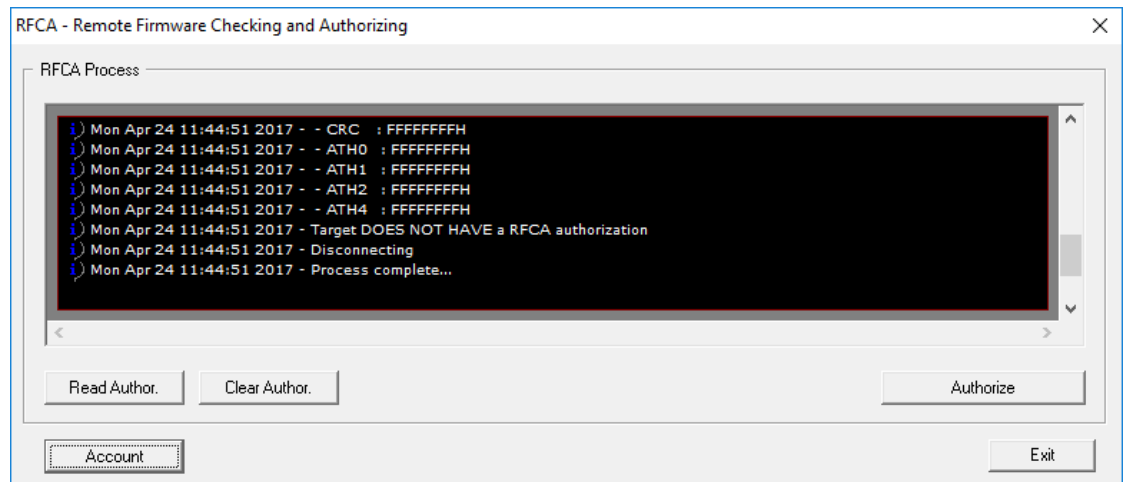
## RFCA PROCEDURE

Once you have upgraded the new firmware into the module a further operation is needed, this

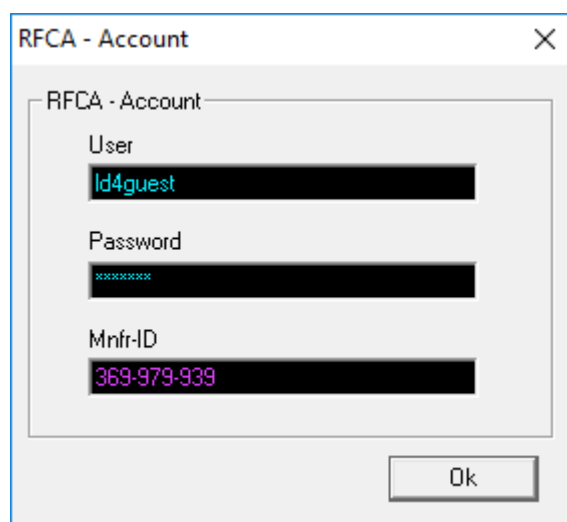
is called **RFCA** (Remote Firmware Checking and Authorizing)

```
//configuration//ide//rfca_access
<rfca_access_enable value="purpleCode"/>

    <configuration>
      <ide>
        <rfca_access>
          <rfca_access_enable value="purpleCode"/>
        </rfca_access>
      </ide>
    </configuration>
```



<b>USER</b>	<b>ld4guest</b>
<b>PSW</b>	<b>jupiter</b>
<b>MNFRID</b>	<b>369-979-939</b>







**Part**

---



## 7 Getting Started with LadderDIP IV

This topic will help you to run your first projects using LadderDIP IV embedded controllers

### 7.1 LadderDIP IV IMPORTANT NOTES



### LadderDIP IV IMPORTANT NOTES

Before starting to use LadderDIP IV please carefully follow the listed notes and guidelines

1) LadderDIP IV is an OEM module. It is sold as OPEN BOARD electronic circuit. The module must be handled carefully. Pay attention to mechanical shocks and any kind of mechanical damage



2) LadderDIP IV is an ESD (Electro Static Sensitive) device. Before handling with your hands be sure to observe all the necessary rules to avoid permanent damages to the module

3) OUTPUT PIN CURRENT: LadderDIP IV pins are directly driven by a NXP microcontroller. Please carefully do not exceed current and limits like specified in this table [DC/AC Parameters and absolute max ratings](#)

4) OUTPUT PINS SAFE RESISTORS: Since LadderDIP IV pins can be configured as input or output, some erroneous configurations could generate undesired over-currents when a pin is wrongly used as OUTPUT but the external circuit DRIVES it assuming it is an INPUT. **WE STRONGLY SUGGEST TO APPLY ALWAYS A 1K RESISTOR IN SERIES TO ANY INPUT DRIVING CIRCUIT.** In this case, even if the pins is erroneously configured as OUTPUT, the limiting resistor will preserve the on-board circuitry.

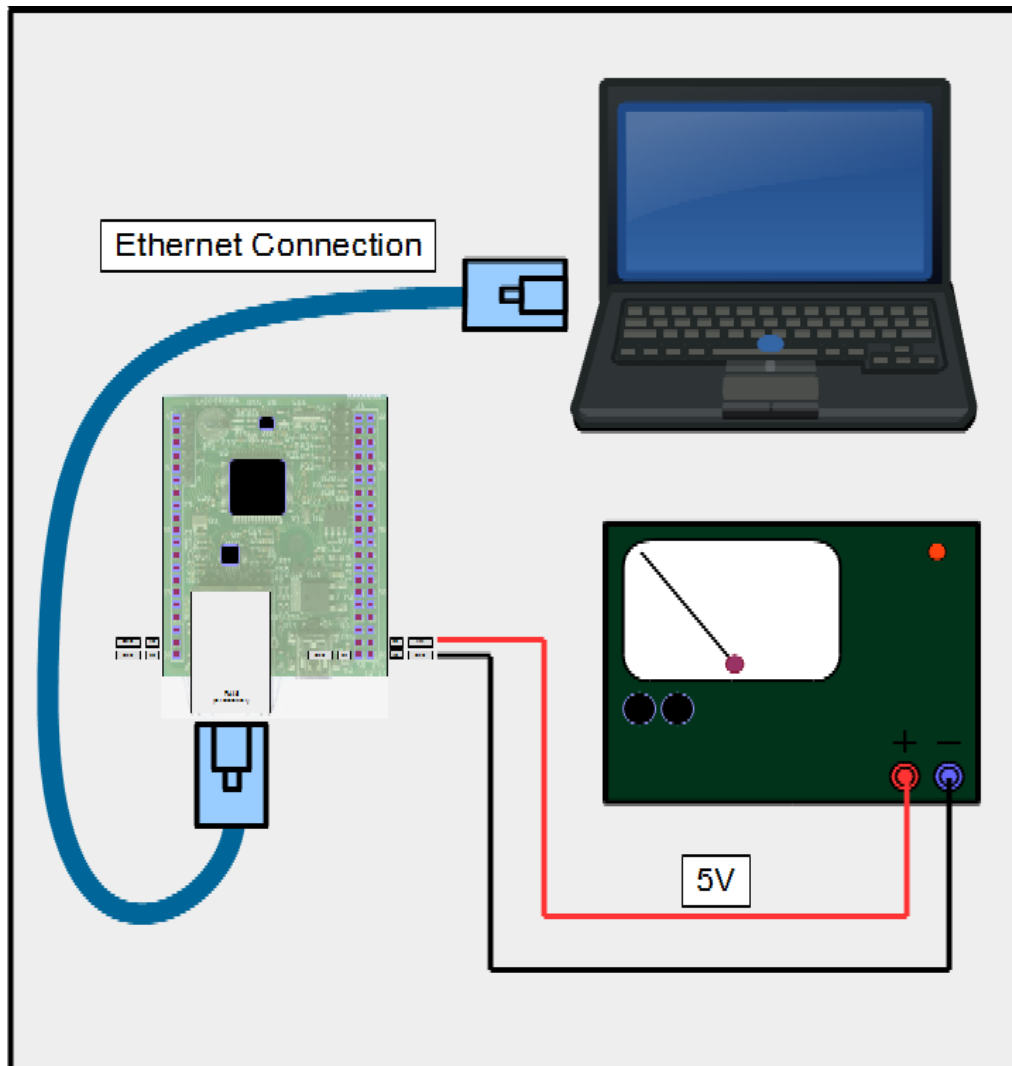
## 7.2 What i need to work with LadderDIP IV

In order, to work with LadderDIP IV you need the following things

- A LadderDIP IV Module
- The LadderDIP PLC Studio software setup program
- 5V Power Supply
- Ethernet Cable

## 7.3 My basic LadderDIP IV connection

The picture below represent the basic <%PRODUCTNAME%> connection used for development. To work with the module you just need to provide the power supply and the Ethernet connection



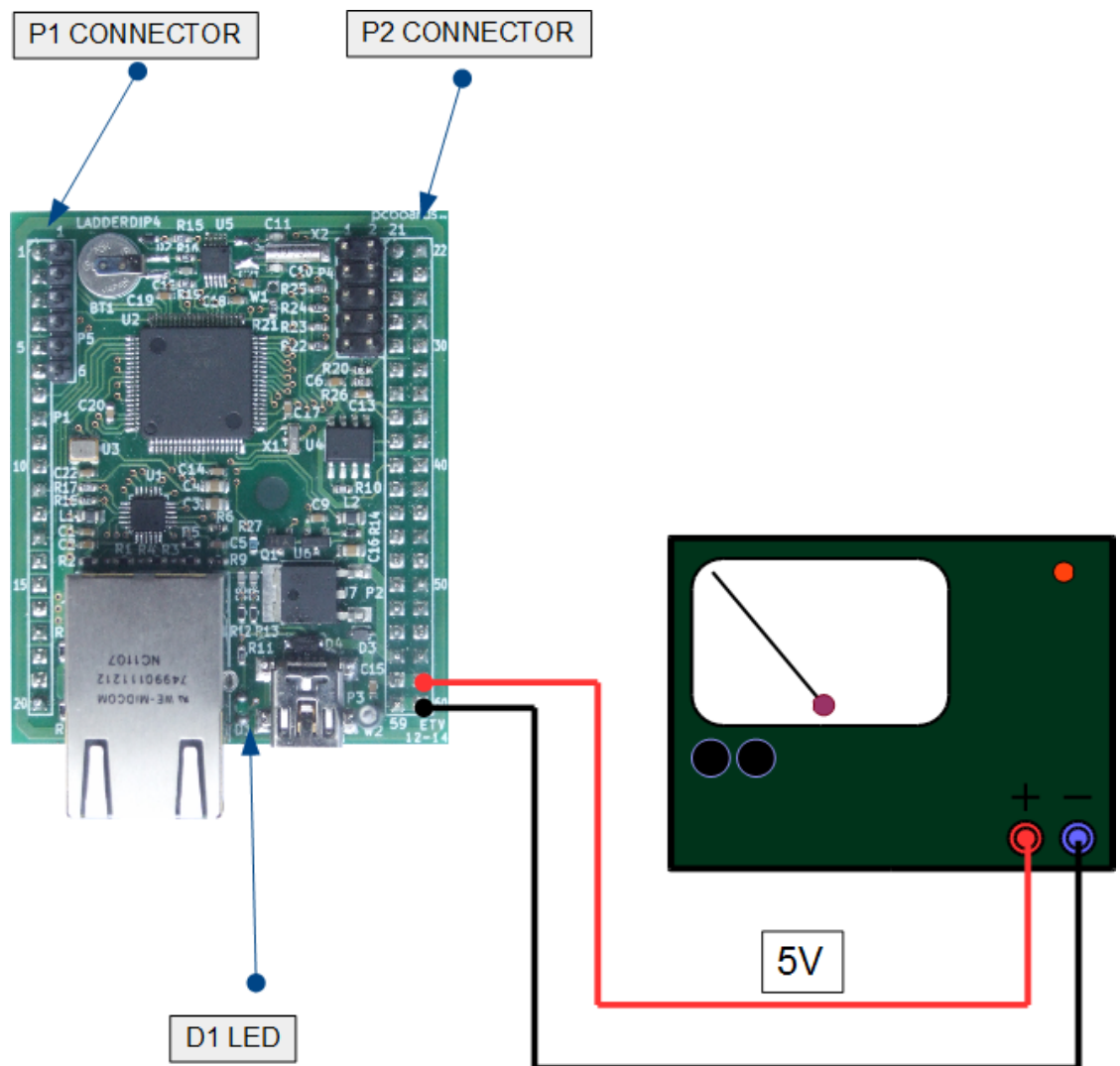
## 7.4 Connecting and powering my module

In order, to work as standalone module, LadderDIP IV needs two essential connections

- A 5V DC power supply
- An Ethernet connection (10/100 Mbits)

Power supply must be connected to:

- Pin P2-38 (+5V) - (Marked as "58" in the right side of the double line connector P2)
- Pin P2-40 (GND) - (Marked as "60" in the right side of the double line connector P2)



Once you have powered-up the module the D1 led will perform some initial flashing

## 7.5 Checking connection

Once i connected and powered-up the [LadderDIP IV](#) i can check if the module is working properly following the listed operation

Connect the Ethernet cable supplied in the starter-kit from the [LadderDIP IV](#) Ethernet Port to your PC

Remember to configure your PC Ethernet port with the necessary IP address and SUBNET mask.

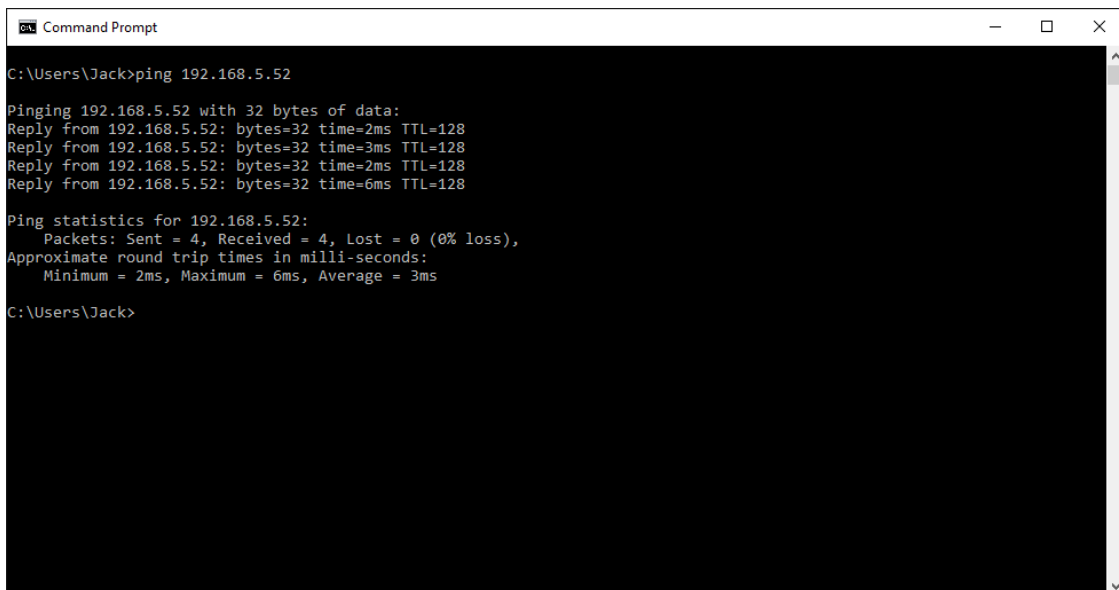
For example, if you [LadderDIP IV](#) device respond to the address "192.168.5.52" (Factory setting) you have to assign an IP address to your PC in the same subnet (i.e. 192.168.5.91) and set the subnet mask to 255.255.255.0

**NOTE: After power up the module requires about 6 seconds to be active, during this period no answers will be given on Ethernet Port**

- The module communicates with the PC using the Ethernet connection. The module communicates at 100mbps
- From your PC execute, using a command prompt, the following command

```
ping 192.168.5.52
```

If everything is ok the module will answer to the ping request



```
Command Prompt
C:\Users\Jack>ping 192.168.5.52

Pinging 192.168.5.52 with 32 bytes of data:
Reply from 192.168.5.52: bytes=32 time=2ms TTL=128
Reply from 192.168.5.52: bytes=32 time=3ms TTL=128
Reply from 192.168.5.52: bytes=32 time=2ms TTL=128
Reply from 192.168.5.52: bytes=32 time=6ms TTL=128

Ping statistics for 192.168.5.52:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 6ms, Average = 3ms

C:\Users\Jack>
```

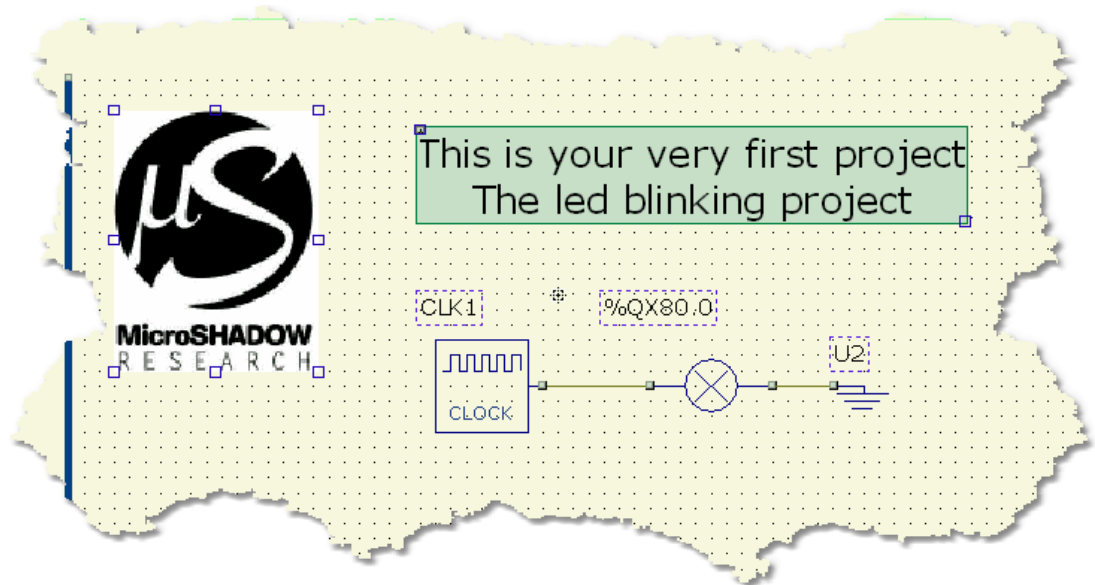
After these checkings the module is properly powered and connected

## 7.6 Flashing my first LED in 1 minute

This section will lead you to run your very first project. The objective of this project is to blink a led using a very simple project called **LedBlink**.

Follow the indicated steps

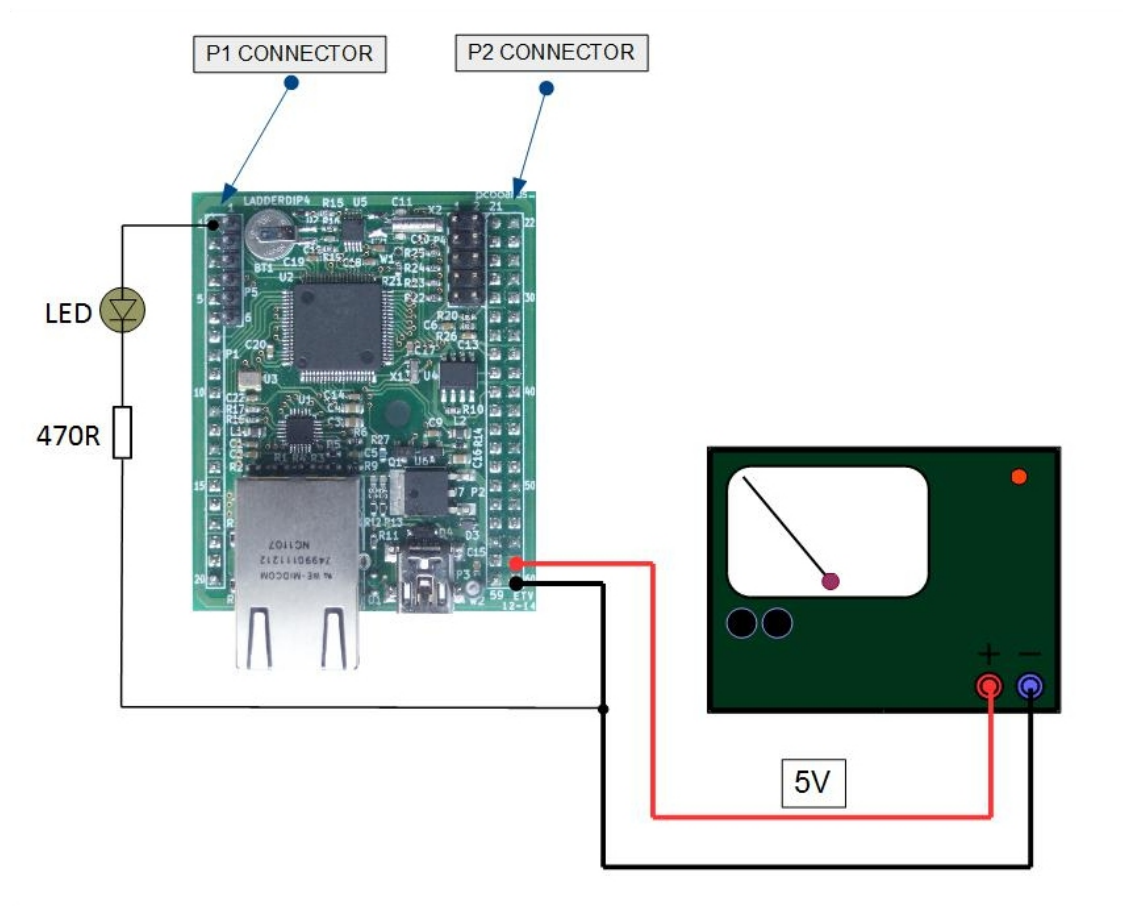
- Launch the LadderDIP PLC Studio software
- Open the **LedBlink** project "..\projects\samples\ladderdip4\ledblink\ledblink.lww", the following diagram should appear



The project uses three only components. The first component, the **CLOCK**, generates a square wave pulse stream. The second component, attached to the **CLOCK** output, is an electrical generic output block (**EOUTPUT**). The output block is associated with the address **%QX80.0** that means the first output **OUT1** of LadderDIP

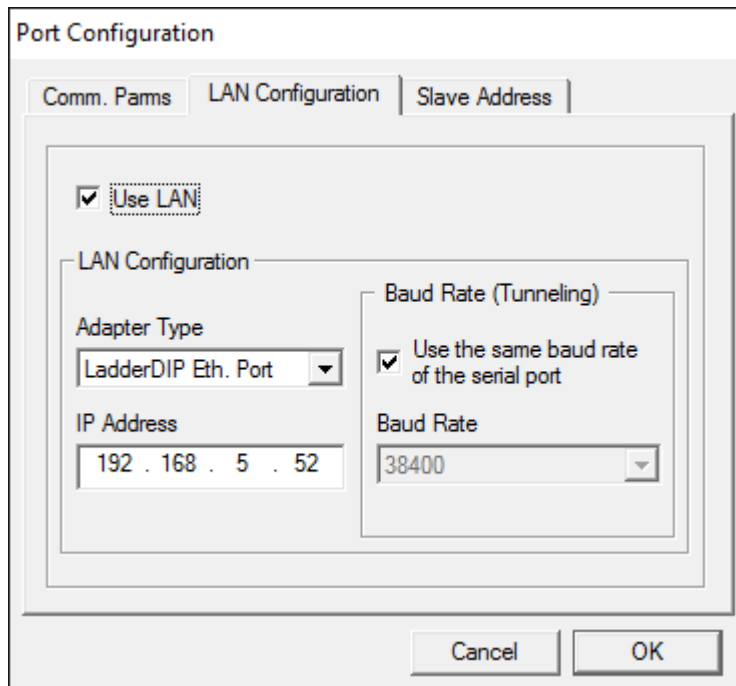
To make this project running you need to connect a simply LED on the pin **IO1** of your LadderDIP. The LED also need a resistor for current limiting  
Please refer to the image below





The project is already configured to use the IO1 pin of LadderDIP module as an OUTPUT.

- Check your Ethernet port configuration accessing, from the menu, the **Options → Port Dialog (Lan Configuration Tab)** . The following screen will appear



Please carefully check the following parameters:

- The "Use Lan" Check must be set
- The "Adapter Type" must be set to "LadderDIP Eth. Port"
- The "IP Address" must be the one set for the module



#### IMPORTANT NOTE

**LadderDIP PLC Studio uses the Ethernet port of your PC. You could have Firewalls or Anti-Viruse softwares that block any access to this resource**

**Please properly configure, if you have installed in your PC, any Firewall or protection that locks the access to the Ethernet Port**

#### TESTING THE CONNECTION OF LadderDIP PLC Studio with LadderDIP IV

Read the following paragraph to check the connection: [Checking connection](#)

At this point, using the LadderDIP PLC Studio software, you can check if the connection is OK simply pressing the connect button

- Once everything is ready you can press the Build All button (Also indicated as **Build & Upload & Run** in the build menu command), this will execute all the necessary operations to download and run a program into the **LadderDIP IV**

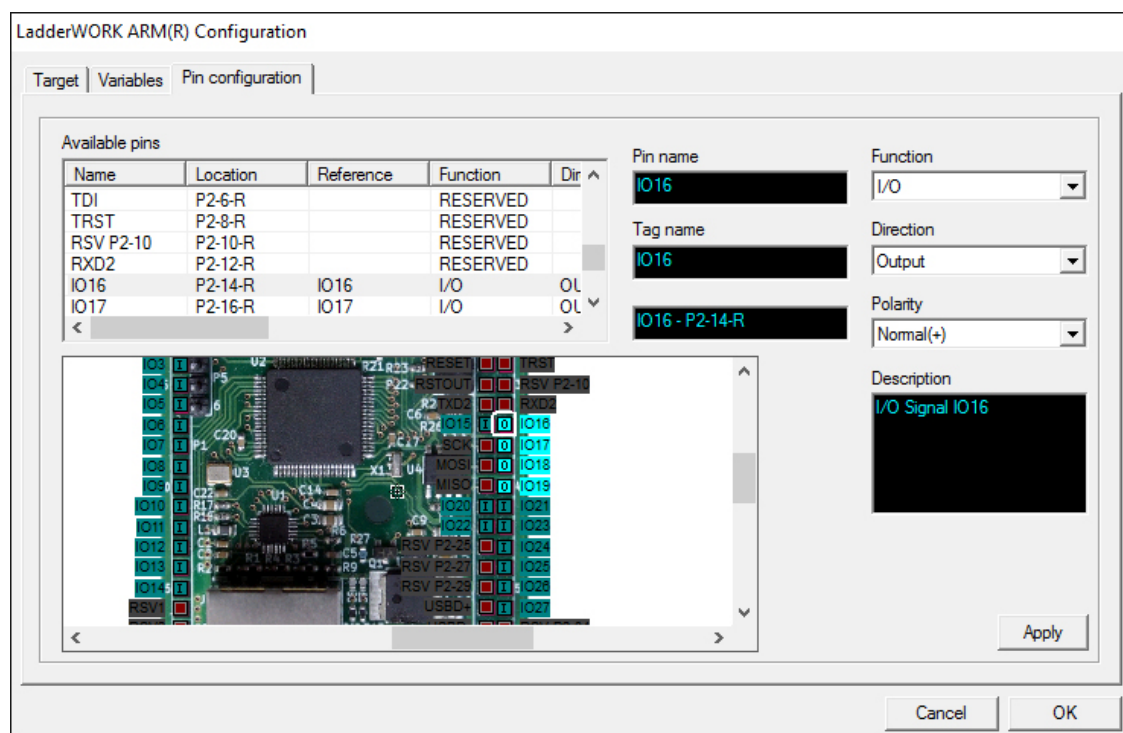
module. The executed processes will be: Compile, connect, erase, download and run. These processes will be accomplished by dialogs and progression bar to check the current status of operations

If everything is ok you will see the project running. You can see your LED blinking ... enjoy it

## 7.7 Pin Configuration

To configure the I/O properties (Function, direction and polarity) you have to access the *Options* → *Configuration* menu.

The following Dialog will appear



To configure a pin property you can select the resource by clicking in the "Available Pin" list or directly on the picture.

To change the configuration of a pin use the three combo boxes named "Function", "Direction" and "Polarity" and press "Apply"

## 7.8 Using the AD (Analog to Digital)



### VERY IMPORTANT

Analog inputs of LadderDIP are directly connected to the on-board NXP microcontroller. Input voltage for these signals **MUST NOT** exceed the range 0 - 3.0V

Refer to this table: [DC/AC Parameters and absolute max ratings](#)

There are six on-board **A/D** (Analog to Digital Converter) available on LadderDIP. These resources are accessible, in the Ladder diagram, using the **AD\_CONV** function block.

This block has a unique output pin that gives the value of the converted analog input. The output value can span from 0 to 1023 (10 bits)

In order, to use the **AD\_CONV** block, follow the listed instructions

- Use the **AD\_CONV** block configuring the CHANNEL parameter according to the following table

ANALOG INPUT	POSITION	CHANNEL
AIN1	P1-5	0
AIN2	P1-6	1
AIN3	P1-14	2
AIN4	P1-15	3
AIN5	P2-30	4
AIN6	P2-32	5

- Also configure the **AD\_CONV** block with the following parameters

PARAMETER	VALUE	NOTE
SPAN	1	
OFFSET	0	

- Using the Pin Configuration Dialog (**Options → Configuration → "Pin Configuration Tab"**) set the pin you want to use as Analog Input to "**AD**"

## 7.9 Using the DAC

The on-board **DAC** (Digital to Analog Converter) of LadderDIP is accessible, in the Ladder diagram, using the **DAC** function block.

This block has an unique input pin that receive the value to be converted in an analog signal (0...3V). The supplied value can span from 0 to 1023 (10 bits)

In order, to use the **DAC** block, follow the listed instructions

- Use the DAC block configuring the CHANNEL parameter to zero
- Using the Pin Configuration Dialog (**Options** → **Configuration** → "Pin Configuration Tab") set the IO13 (P1-14) pin to the function "DA"

## 7.10 Using the PWM (Pulse Width Modulation)

The two on-board PWM (Pulse Width Modulation) of LadderDIP is accessible, in the Ladder diagram, using the **PWMOUT** function block.

This block has an unique input pin that receive the value to be converted in an analog signal (0...3V). The supplied value can span from 0 to 1023 (10 bits)

In order, to use the **PWMOUT** block, follow the listed instructions

- Use the **PWMOUT** block configuring the CHANNEL parameter according to the following table

ANALOG INPUT	POSITION	PIN	CHANNEL
PWM1	P1-1	IO1	1
PWM2	P1-8	IO7	2

- Using the Pin Configuration Dialog (**Options** → **Configuration** → "Pin Configuration Tab") set the pin function to "PWM"

**Part**

---



## 8 Getting Started with LadderDIP IV Development Board

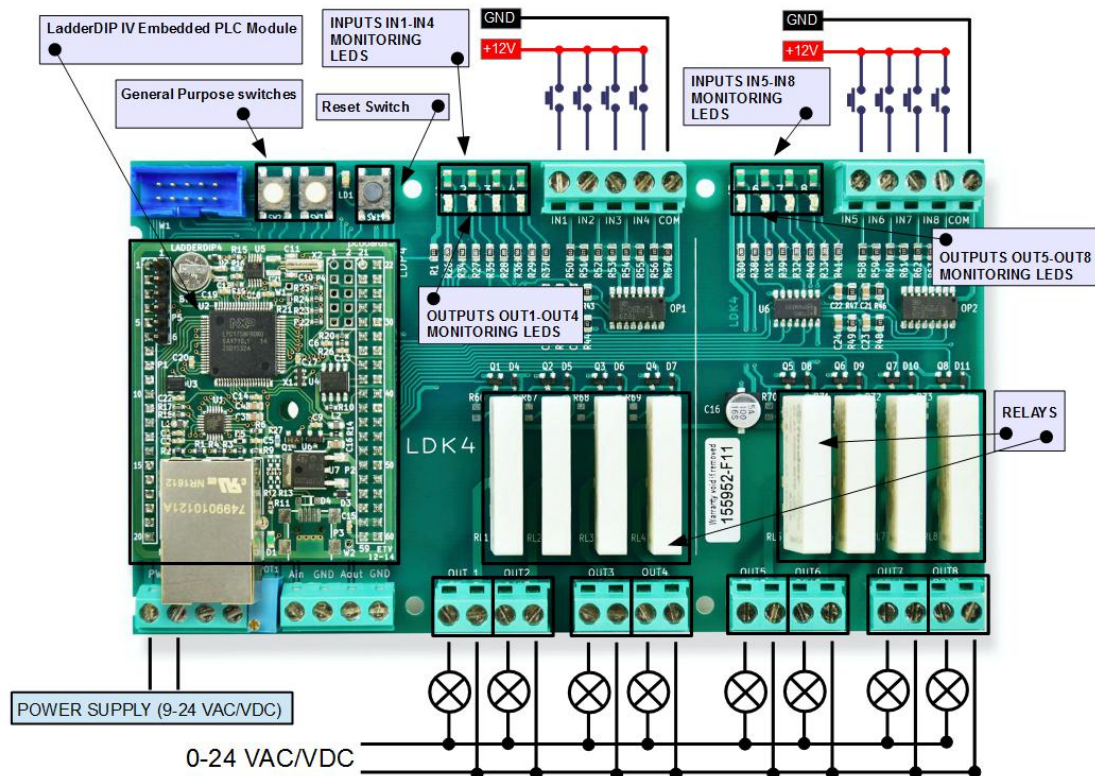
This topic will help you to set-up and get your LadderDIP IV Development Board running in few easy steps

### 8.1 What is LadderDIP IV Development Board

LadderDIP IV Development Board is a kit that allow you to transform LadderDIP IV into two different forms

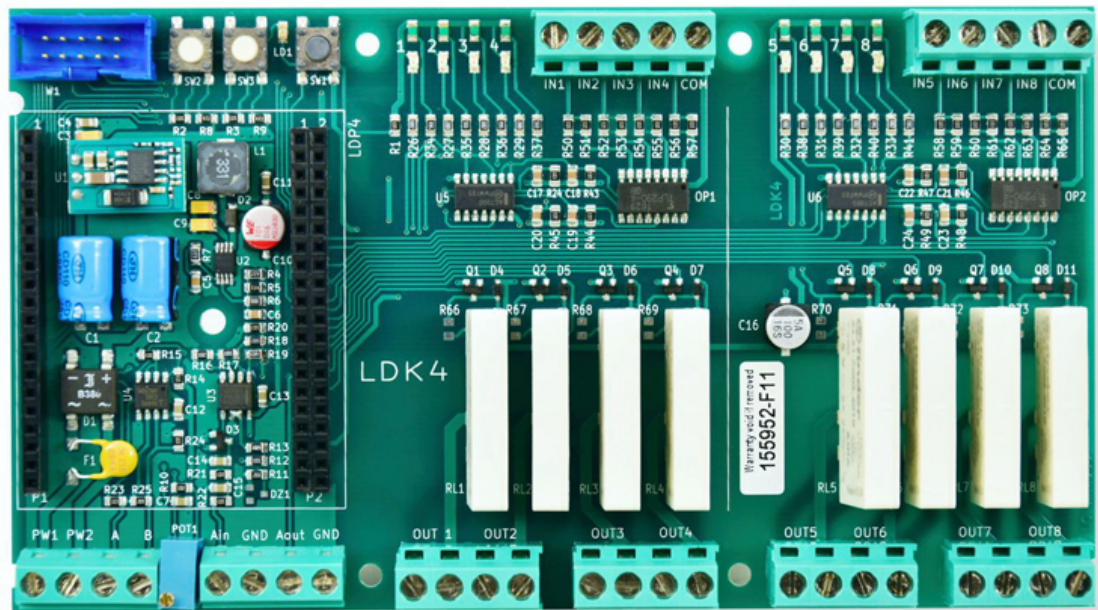
- A complete Development Board for the LadderDIP IV
- An industrial-proof enclosure that allow you to convert the LadderDIP IV module into a professional rugged-grade PLC

Let's take a look to the LadderDIP IV Development Board board



The LadderDIP IV Development Board acts like a "mother-board (or backplane)" for the LadderDIP IV Development Board module





The board gives the following features

- 9-24 Wide Range VAC/VDC Power Supply
- 8 x 250VAC 6A Relay with LED Monitoring
- 8 x Insulated Inputs 9-24 VAC/VDC with LED Monitoring
- 2 x General Purpose Switch
- Reset Button
- 1 x Analog Input (0..10V)
- 1 x On-Board Trimmer (Connected to LadderDIP Analog Input AIN6)
- 1 x Analog Output (0..10V)
- RS485 expansion

## 8.2 What i need to work with LadderDIP IV Development Board

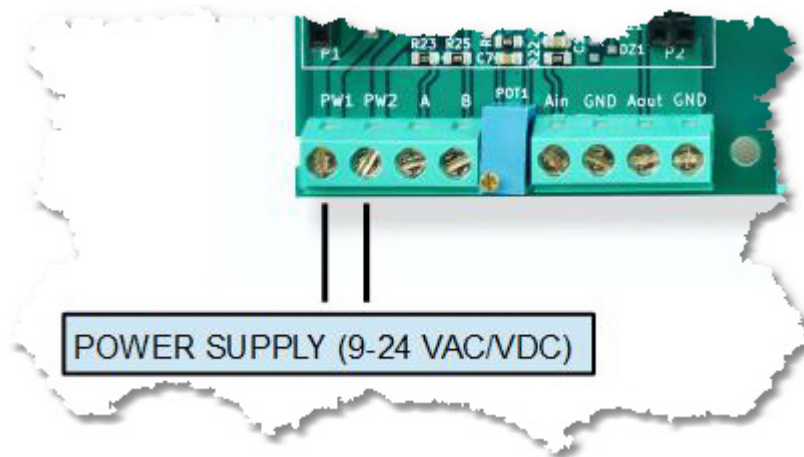
In order, to work with LadderDIP IV Development Board you need the following things

- A LadderDIP IV Development Board kit
- The LadderDIP PLC Studio software setup program
- Ethernet Cable



### 8.3 Power up LadderDIP IV Development Board

LadderDIP IV Development Board power supply requires 9-24 VAC/VDC on PWR1 & PWR2 clamps

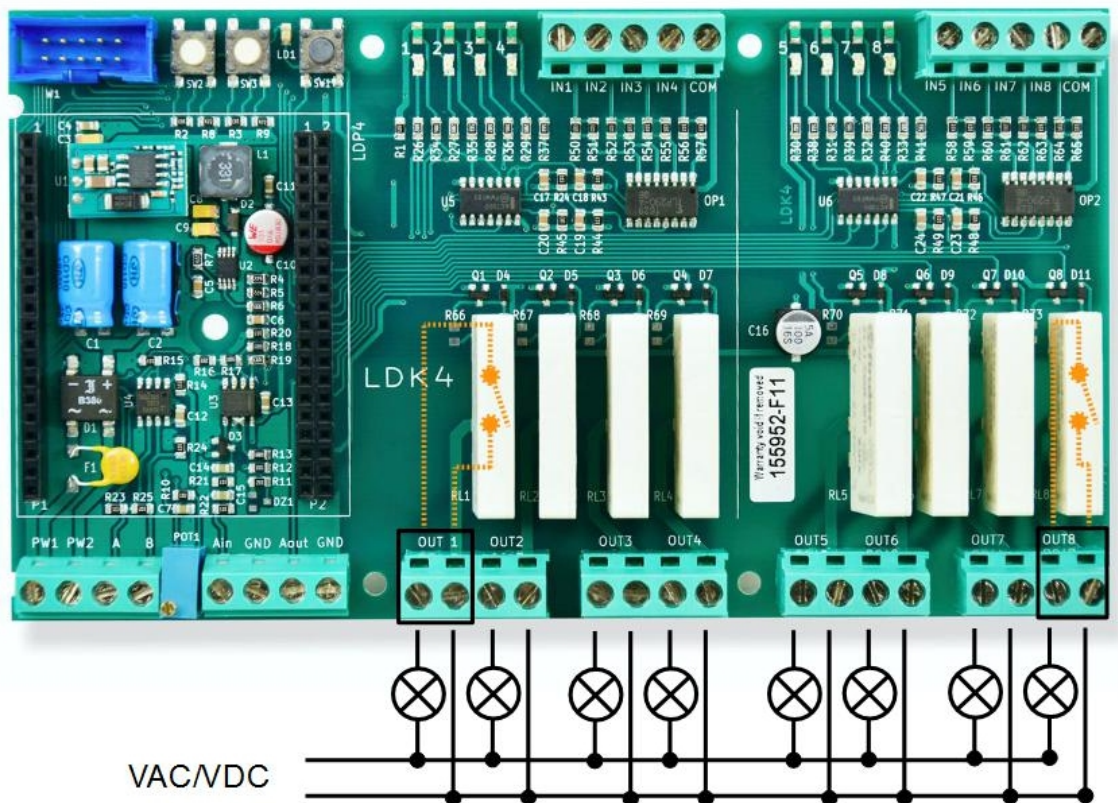


## 8.4 Using the "RELAY" outputs



LadderDIP IV Development Board mounts LOW-LOAD relays. Carefully observe current and voltage limits when attaching load to these outputs. Please, before using relays, read this table here: [LadderDIP IV Development Board parameters](#)

LadderDIP IV Development Board have eight on-board relays that could be used to drive loads. The relays are configured as NO (Normally Open) contacts. This means that, normally, when the relay is not driven, the contact is open.

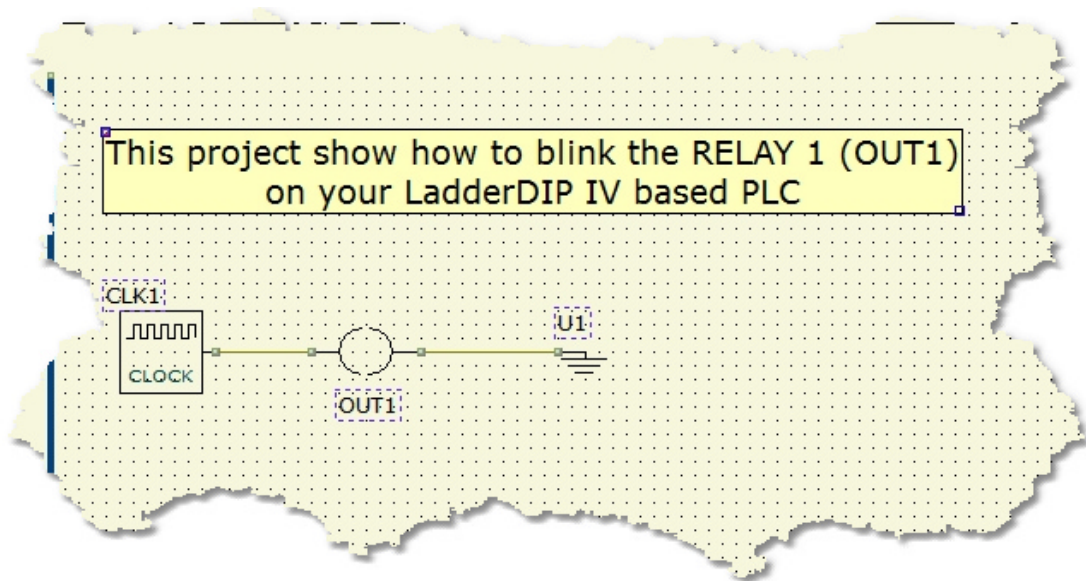


The picture above shows how to connect simple lamps to the eight available relay outputs.

Let's start with a practical example. The objective of this first project is to blink a relay using a very simple project called **RelayBlink**.

Follow the steps listed here below

- Launch the LadderDIP PLC Studio software
- Open the **RelayBlink** project "..\projects\samples\ladderbox\relayblink\relayblink.lww", the following diagram should appear

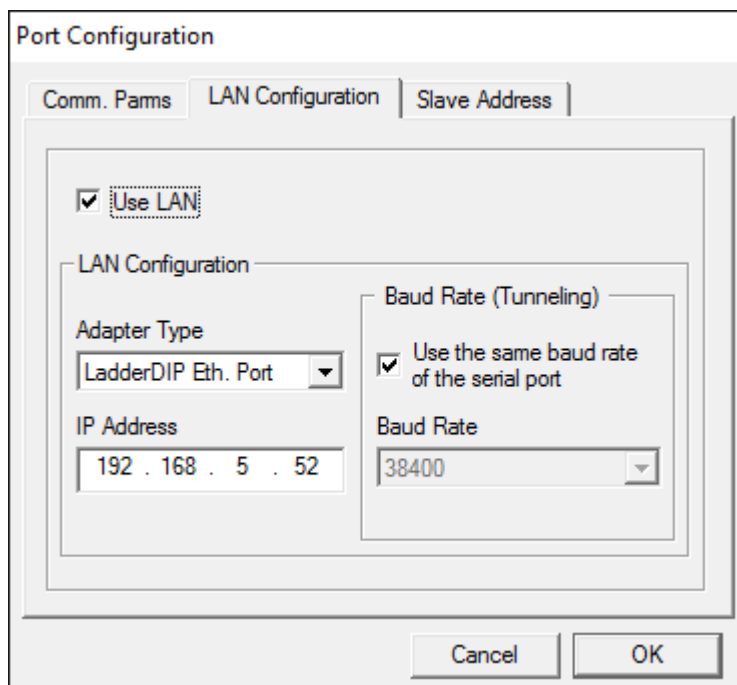


The project is very easy and it takes just two components. The CLOCK component generates a train of 1Hz pulses feeding the output named OUT1 (Logically related to RELAY 1)

Now it is time to run your project

Follow the next instructions

- Check your Ethernet port configuration accessing, from the menu, the **Options → Port** Dialog (Lan Configuration Tab) . The following screen will appear



Please carefully check the following parameters:

- The "Use Lan" Check must be set
- The "Adapter Type" must be set to "LadderDIP Eth. Port"
- The "IP Address" must be the one set for the module




### IMPORTANT NOTE


**LadderDIP PLC Studio uses the Ethernet port of your PC. You could have Firewalls or Anti-Viruse softwares that block any access to this resource**

**Please properly configure, if you have installed in your PC, any Firewall or protection that locks the access to the Ethernet Port**

### TESTING THE CONNECTION OF LadderDIP PLC Studio with LadderDIP IV Development Board

Read the following paragraph to check the connection: [Checking connection](#)

At this point, using the LadderDIP PLC Studio software, you can check if the connection is OK simply pressing the connect button 

- Once everything is ready you can press the Build All button  (Also indicated as **Build & Upload & Run** in the build menu command), this will execute all the necessary operations to download and run a program into the **LadderDIP IV Development Board** module. The executed processes will be: Compile, connect, erase, download and run. These processes will be accomplished by dialogs and progression bar to check the current status of operations

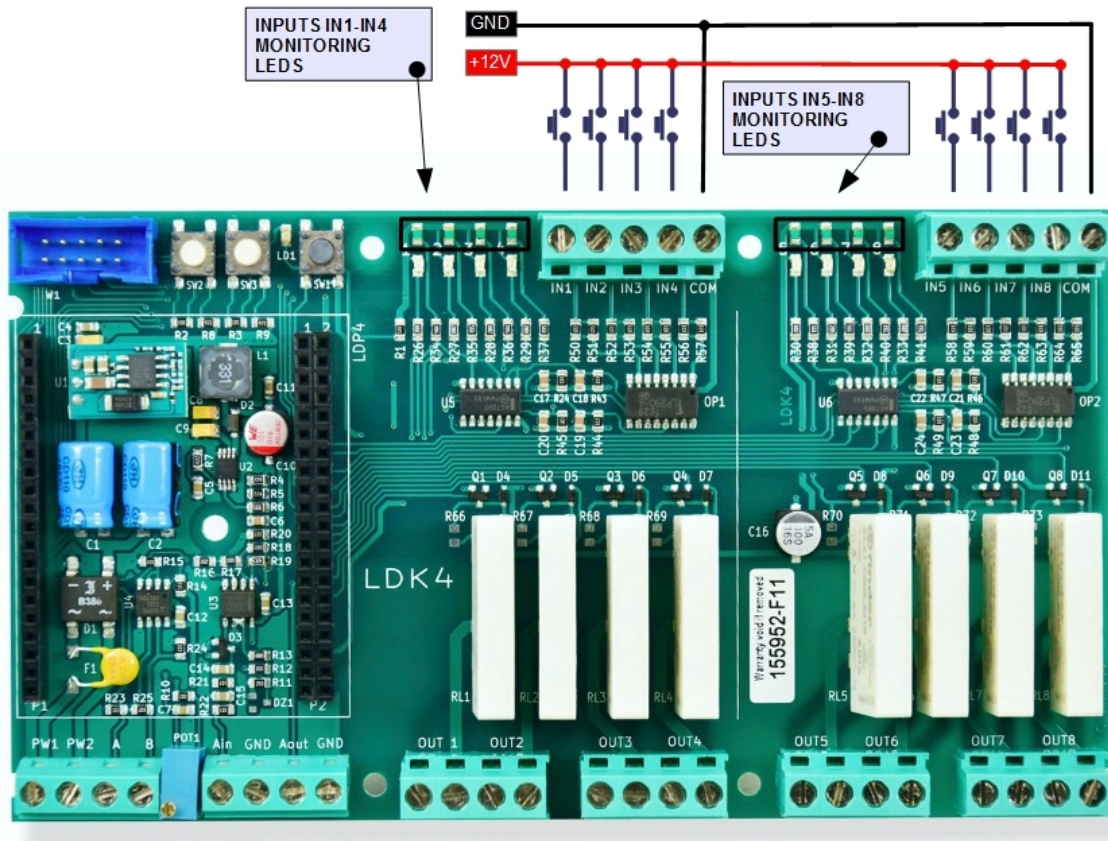
If everything is ok you will see the project running. You can see the OUT1 blinking and you can also hear the relay clicks ... enjoy it

## 8.5 Using the "INSULATED" inputs



Please, before using opto-insulated inputs, read this table here: [LadderDIP IV Development Board parameters](#). Exceeding max voltages will turn in a permanent damage

LadderDIP IV Development Board have eight on-board opto-insulated inputs that could be used to connect external switches or direct voltages. On board led(s) are available to monitor inputs state (led off = no signal, on = signal present)

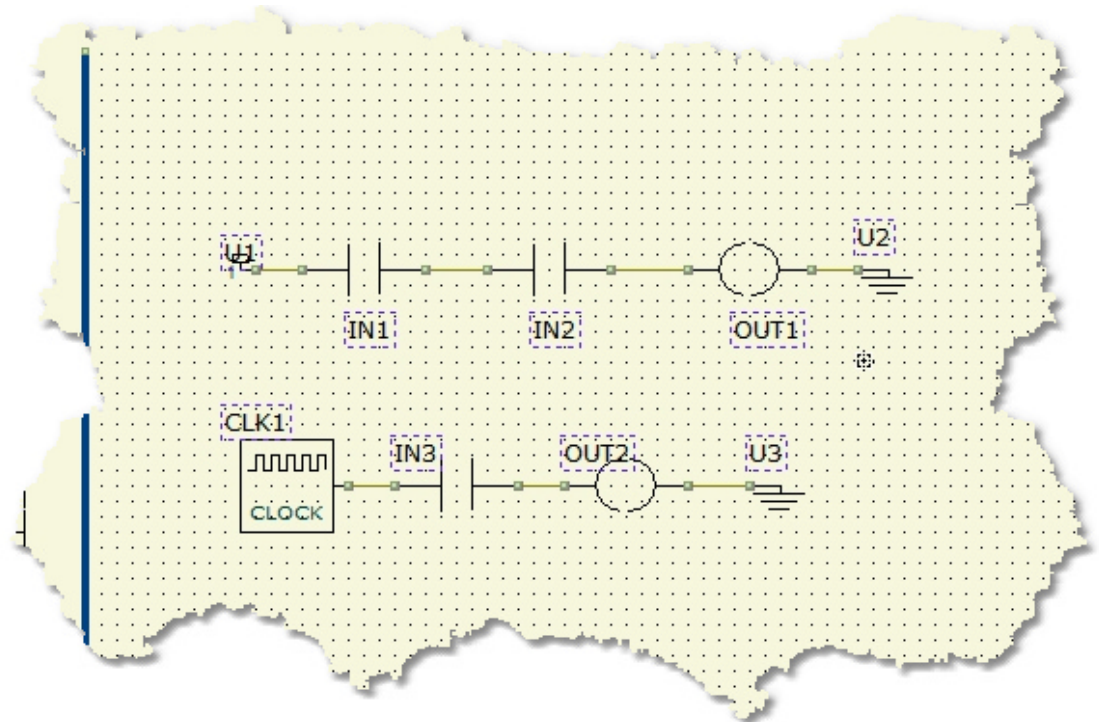


It is time for a practical example. We will run a first project that uses the opto-insulated inputs associated with the INPUT input block

Follow the steps listed here below

- Launch the LadderDIP PLC Studio software
- Open the project "..\projects\samples\ladderdip4\ledblink\inputs.lww", the following diagram should appear





Now, the project has two sections, in old LADDER programs they are called RUNGS. These two portions of diagrams works as parallel processes.


The first section, formed by IN1, IN2 and OUT1 forms an "AND" function. The OUTPUT OUT1 is activated only when the two INPUTS (IN1 and IN2) are active


There is a direct relationship between the names applied on the diagram (IN1, IN2 and OUT2), and the physical connection on LadderDIP IV Development Board board, so,

the INPUT IN1 correspond to the physical clamp named IN1, same concept for all the rest of signals.

Applying a voltage to the opto-insulated inputs will activate the corresponding signal (or function block) on the diagram. The LED monitor on the PCB will also turn on when the corresponding signal is activated.

Now, as we already made for the previous projects, it is time to run everything

At this point, using the LadderDIP PLC Studio software, you can check if the connection is OK simply pressing the connect button 

- Once everything is ready you can press the Build All button  (Also indicated as **Build & Upload & Run** in the build menu command), this will execute all the necessary operations to download and run a program into the **LadderDIP IV Development Board** module. The executed processes will be: Compile, connect, erase, download and run. These processes will be accomplished by dialogs and progression bar to check the current status of operations

If everything is ok you will see the project running. This is the project behaviour:

- OUTPUT OUT1 is activated when both IN1 and IN2 are active. You have to apply a 12VDC voltage to both the inputs, in order, to see the output on. For

- The OUTPUT OUT2 will blink at 1Hz frequency when the IN3 is active (Signal is transferred from CLK1 to OUT2 passing through IN2)

You can see how these two portions of diagrams work in distinct parallel mode

## 8.6 Using the Analog Input (0-10V)



### VERY IMPORTANT

Carefully respect the max voltage values as written in the product specifications

Refer to this table: [DC/AC Parameters and absolute max ratings](#)

There is one on-board **A/D** (Analog to Digital Converter) 0-10V input on LadderDIP IV Development Board. This resource is accessible, in the Ladder diagram, using the **AD\_CONV** function block.

This block has a unique output pin that gives the value of the converted analog input. The output value can span from 0 to 4095 (16 bits)

In order, to use the analog input with the **AD\_CONV** block, follow the listed instructions

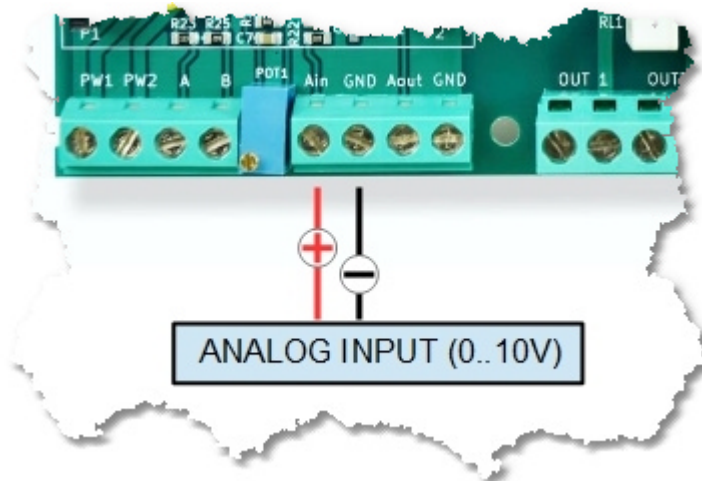
- Use the **AD\_CONV** block configuring the CHANNEL parameter according to the following table

ANALOG INPUT	CHANNEL
AIN(0-10V)	3

- Also configure the **AD\_CONV** block with the following parameters

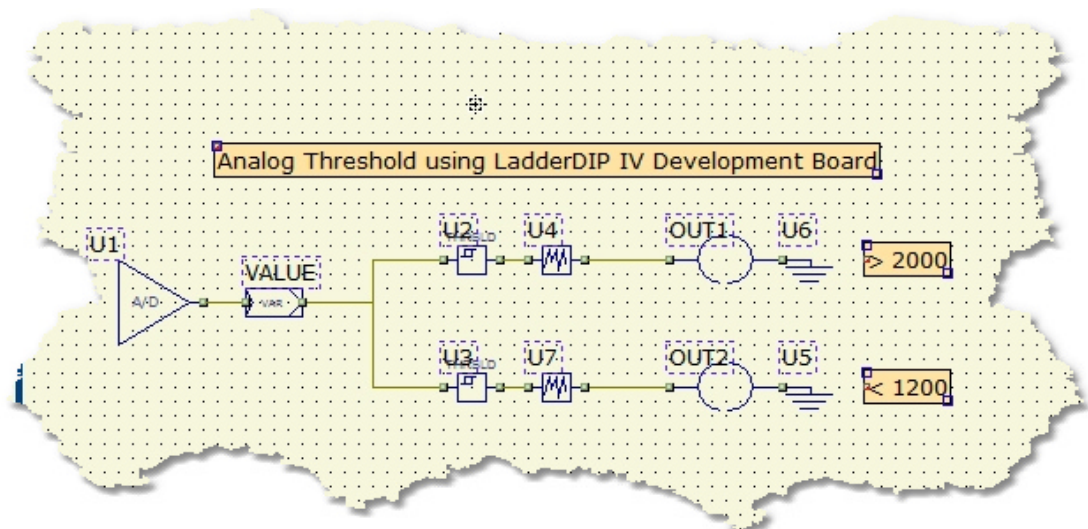
PARAMETER	VALUE	NOTE
SPAN	1	
OFFSET	0	

Internally, the 0-10V analog input is connected, through an operational amplifier, to the analog input AIN4 (P1-15) on the LadderDIP IV module




Now we can try how the 0-10V analog input works opening the project named **AnalogThresholds** which is located at  
 "..\projects\samples\ladderbox\analogthresholds\analogthresholds.lww"

Once you opened the project the following diagram will appear



**NOTE:** To test this project i need a voltage source applied between the AIN and the GND screw connector. You can use a simple battery (For example 9V) connected with a trimmer to the AIN input, or, any other voltage source that could span between a range of 0 to 10V is of course good

- Once everything is ready you can press the Build All button  (Also indicated as **Build & Upload & Run** in the build menu command), this will execute all the necessary operations to download and run a program into the **LadderDIP IV Development Board** module. The executed processes will be: Compile, connect, erase, download and run. These processes will be accomplished by dialogs and progression bar to check the current status of operations

If everything is ok you will see the project running. This is the project behaviour:

- The analog voltage value applied to the AIN input is processed by the AD\_CONV component



(A/D)

- The converted value, normalized in the range 0..4095, is then written to the variable named **VALUE**. During the debug the raw value is visible here
- A threshold component (U2) is configured to activate its output when the value is above 2000, when this occurs the corresponding output is activated
- A second threshold component (U3) is configured to activate its output when the analog value is under 1200, when this happen the output is activated
- The two DEBOUNCE blocks (U4/U7), configured with a convolution time of 100mS, are used to filter spikes and noises present on the analog input

## 8.7 Using the On-Board Trimmer

LadderDIP IV Development Board has an on-board trimmer that generates an internal analog voltage. This resources is accessible, in the Ladder diagram, using the **AD\_CONV** function block.

This block has an unique output pin that gives the value of the converted analog input. The output value can span from 0 to 4095 (16 bits)

In order, to use the trimmer with the **AD\_CONV** block, follow the listed instructions

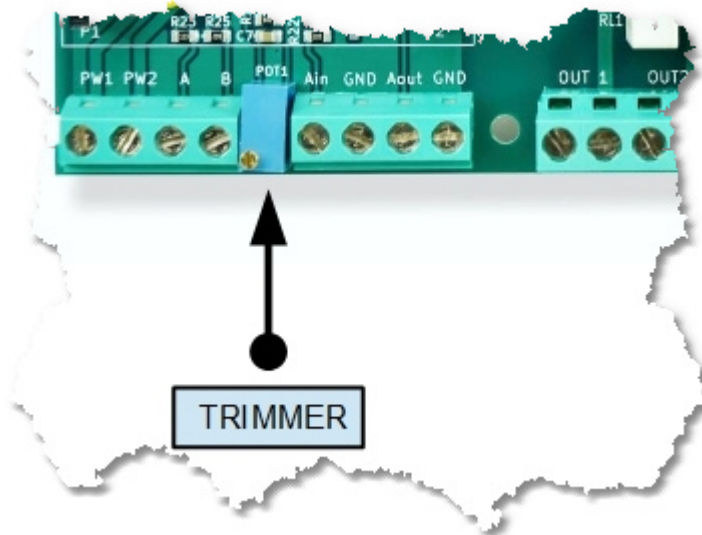
- Use the **AD\_CONV** block configuring the CHANNEL parameter according to the following table

ANALOG INPUT	CHANNEL
TRIMMER	5

- Also configure the **AD\_CONV** block with the following parameters

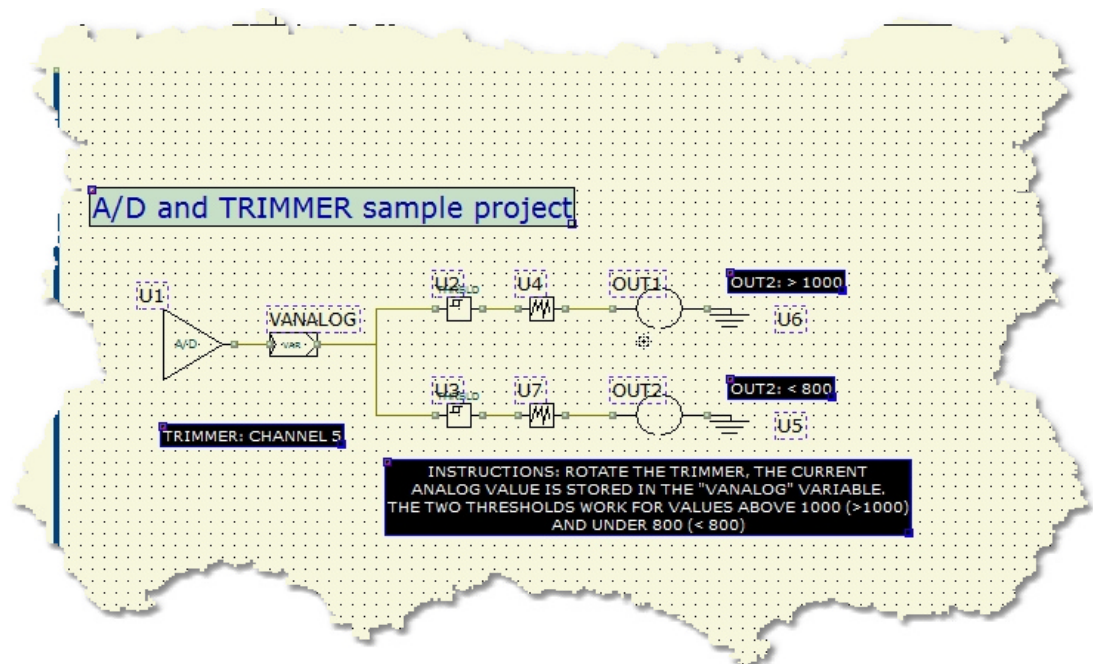
PARAMETER	VALUE	NOTE
SPAN	1	
OFFSET	0	


Internally, the trimmer analog voltage is connected to the analog input AIN6 (P2-32) on the LadderDIP IV module



Now we can try how the TRIMMER works opening the project named "**Trimmer**" which is located at "..\projects\samples\ladderbox\trimmer\trimmer.lww"

Once you opened the project the following diagram will appear



Once everything is ready you can press the Build All button  (Also indicated as **Build & Upload & Run** in the build menu command), this will execute all the necessary operations to download and run a program into the **LadderDIP IV Development Board** module. The executed processes will be: Compile, connect, erase, download and run. These processes will be accomplished by dialogs and progression bar to check the current status of operations

If everything is ok you will see the project running. This is the project behaviour (You need a

small screwdriver to rotate the TRIMMER)

- The analog voltage, generated by the TRIMMER network, is stored in the variable named VANALOG (Connected to the A/D channel 5)
- The A/D range is 0-4095 (12 Bits)
- A threshold component (U2) is configured to activate its output when the value is above 1000, when this occurs the corresponding output is activated
- A second threshold component (U3) is configured to activate its output when the analog value is under 800, when this happen the output is activated
- The two DEBOUNCE blocks (U4/U7), configured with a convolution time of 100mS, are used to filter spikes and noises present on the analog input

## 8.8 Using the Analog Output (DAC) (0-10V)



Please, before using the DAC output, read this table here: [LadderDIP IV Development Board parameters](#). Exceeding max load currents onto this signal will turn in a permanent damage

LadderDIP IV Development Board has an on-board DAC (Digital to Analog Converter) output that can generate a 0 to 10V voltage. This resources is accessible, in the Ladder diagram, using the **DAC** function block.

At PCB level, this output is named **AOUT**

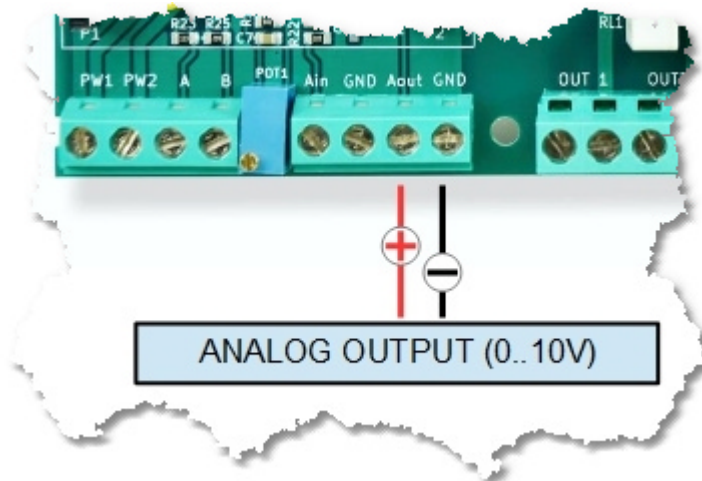
The **DAC** block has an unique input pin, where the applied value, is directly transferred to the DAC hardware

The DAC function uses the internal LadderDIP IV Digital to Analog converter which has a 10 bits resolution (0..1023 values)

In order, to use the AOUT output with the **DAC** block, follow the listed instructions

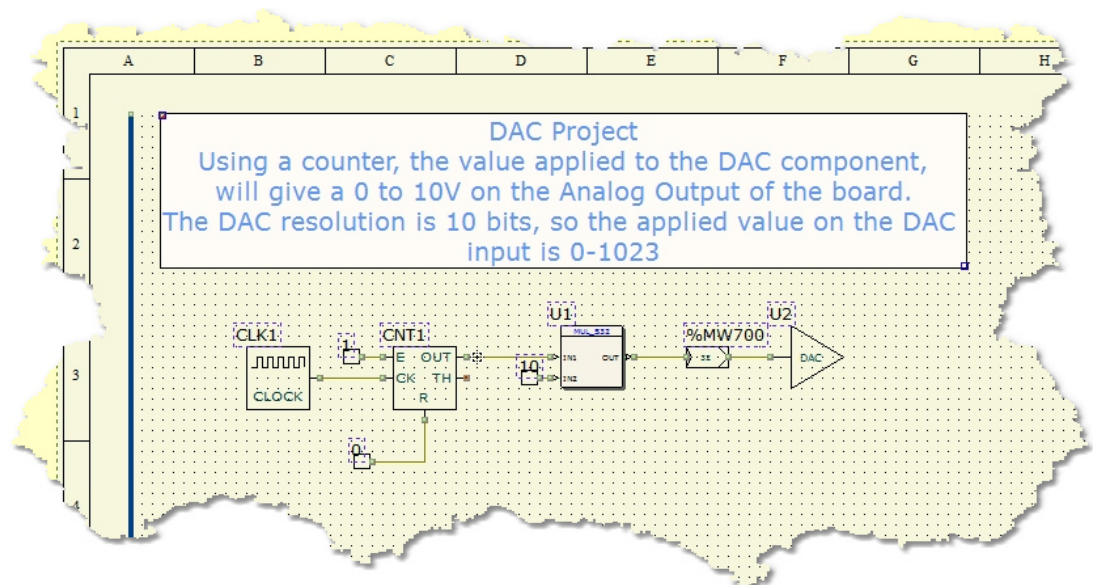
- Use the **DAC** block configuring the CHANNEL parameter according to the following table

ANALOG OUTPUT	CHANNEL
AOUT	0




Now we can try how the TRIMMER works opening the project named "**DAC**" which is located at "..\projects\samples\ladderbox\dac\dac.lww"

The following diagram will appear



To see this project working you need to connect a multimeter or an oscilloscope on the analog output plugs. What you will see will be a ramp of voltages spanning from zero to almost 10 volts

Once everything is ready you can press the Build All button  (Also indicated as **Build & Upload & Run** in the build menu command), this will execute all the necessary operations to download and run a program into the **LadderDIP IV Development Board** module. The executed processes will be: Compile, connect, erase, download and run. These processes will be accomplished by dialogs and progression bar to check the current status of operations

## 8.9 LadderDIP IV Development Board parameters



LadderDIP IV Development Board product is designed to be connected with field signals. **Carefully**, when interfacing the board, take in consideration the following voltage and current max ratings.

Symbol	Parameter	Min Value	Max Value	Unit	Notes
VCC	Power Supply	9	24	VDC	
RLYLoadAC	Max relays contact load when using VAC voltages		1000	W	4A 250VAC
RLYLoadDC	Max relays contact load when using VDC voltages		22	W	0.2A 110VDC
VOpto	Opto-Inputs voltage	9	12	V	
DACVout	Analog output voltage range (DAC)	0	10	V	10 bits D/A converter. Full scale is 1023
DACIout	Analog output max current (DAC)		20	mA	
ADVin	Analog input voltage (A/D)	0	10	V	A/D resolution is 12 bits. Full scale gives a value of 4095
tw	Temperature Range (Working Range)	0	45	°C	

# Part

---



IX

## 9 Introduction to PLC

This section is a brief introduction to PLC ( Programmable Logic Controller ). The goal is the understanding of the main PLC concepts.

### 9.1 PLC Overview

A Programmable Logic Controller (PLC) is an industrial computer specialized for real time applications. PLCs are integrated systems containing a processor, power supply, input modules, output modules and special purpose modules. Input modules interface with plant equipment and convert the field signals to logic levels for the processor to read. The processor uses these inputs to perform control functions based on application software. Output modules transmit the signals via an interface with the plant equipment. In addition there are special modules for communication with other computers, specialized dedicated functions, and conventional high-level language co- processors. Ladder logic will be used in the examples for the purpose of exposition.

### 9.2 Run-time enviroment

The PLC runtime environment is firmware which provides the operating systemservices and library functions associated with the PLC. In the RUN mode,the PLC firmware runs as real-time executive which processes the (LadderLogic) instructions that have been loaded into the program RAM area. The program runs in a continuous loop which consists of the following major phases:

- Input read and output write scan
- Housekeeping
- Program scan (logic solve).

### 9.3 Input read and output write scan

During the input/output (I/O) scan, the processor updates its internal input and output buffers with data being read from or written to I/O devices. Local I/O devices are the input and output cardsresiding in the same physical chassis as the PLC processor. Remote I/Odevices reside external to this chassis and are communicated with the processor's peer communications interface port.

I/O data for input and output cards used in the application are maintained in input and output image tables. Typically the PLC will organize the I/O image tables. This means that the inputs which are present will read into an area in memory. The program will write into another area of memory which is used to represent the outputs. It can be said that the input image table is representative of 'how the inputs are perceived', and the output image table is 'the desired state' of the outputs. These tables are accessible to the Ladder Logic program as data files. During the I/O scan, data read from input cards are placed in appropriate locations in the input image table. At the same time, output data written to the output image table by the Ladder Logic are transferred to the appropriate output cards.

## 9.4 Housekeeping

Following the I/O scan, the PLC performs what is referred to as "housekeeping." This portion of the program cycle is used by the real-time executive to maintain and update its own internal state.

## 9.5 Program scan

The program scan is the portion of the overall cycle where Ladder Logic instructions of the user's application software are executed. Here, the embedded firmware program operates on the portions of memory (E2PROM/FLASH/RAM) that have been loaded previously with the application software from the binary file.

Program files contain the actual instructions to be executed. Data files are used to maintain program variables and other data structures required by the logic. It is the responsibility of the firmware program to properly decode and execute instructions in the program files. The program must also properly update the contents of the data files based on these instructions.

## 9.6 CEI / IEC 1131-3 programming languages

CEI / IEC 1131, Part 3, specifies the semantics and syntax of a unified suite of five programming languages for PLCs. These languages can be grouped into two categories: textual and graphical. Graphical languages are based upon graphical representation, that is, lines, boxes and text to represent specific relations among inputs and outputs. Appropriate quantities flow along lines between elements according to well defined rules. There are three graphical programming languages: Ladder Logic, Sequential Function Charts, and Functional Block Diagrams. Ladder logic is the most common of PLC languages and is discussed in the following section of this appendix. Sequential function charts can be used as a simple language, but their most important function is to integrate modules written in other languages into a single higher level program. Function Block Diagrams uses block diagrams to interconnect the function.

Textual languages consist of a defined set of characters, rules for combining characters with one another to form words or other expressions, and the assignment of meaning to some of the words or expressions. There are two textual languages defined in the standard: Instruction List (IL) and Structured Text (ST). IL is a very low-level language, and may be considered as a standard Assembly Language for PLCs. Structured text is a textual programming language using assignment, sub-program control, and selection and iteration statements to represent the application program for a PLC. ST, as distinguished from IL, is the high-level text-based language for PLCs. Much of its syntax is derived from Pascal.

The models of execution, program organization, and variable handling of all CEI / IEC 1131-3 languages are based on a common hierarchical architecture consisting of Configurations, Resources, Tasks, and Programs.

Configurations are the highest level at which Global variables and Directly Represented Variables may be shared and accessed. A Configuration may often correspond to a single PLC unit, but certain types of PLC Network Architectures as well as multi-processor PLCs also meet this definition. A Configuration is composed of one or more Resources. Each Resource corresponds to a signal processing function, its associated man-machine interface functions, and sensor-actuator interface functions. A single-processor stand-alone PLC Configuration



would have but a single Resource. A Configuration composed of a dozen processors capable of sharing the defined global variables and directly represented variables, on the other hand, would have 12 Resources associated with it.

Each Resource may have Global Variables (which are limited in scope to that Resource), zero or more defined Tasks, and Programs associated with those Tasks. Tasks may be defined as periodic, in which case they are defined with a specified periodicity, or as non-periodic, in which case they are executed upon the detection of the rising edge of a boolean variable. Tasks may also be assigned an execution priority. Tasks may also be scheduled pre-emptively, or non-preemptively. A Program not assigned to a Task will execute repetitively at the lowest priority level.

Programs in the IEC 1131-3 architecture begin with a variable declaration section, followed by the program statements themselves. Programs may contain calls to Functions, which return a single value, or Function Blocks, which return one or more values. Each Program, Function, or Function Block is written in one of the five IEC 1131-3 defined languages. Multi-language programming is accomplished by calling a Function or Function Block written in one language from a Program, Function, or Function Block written in another.

Variables in IEC 1131-3 languages may be either Symbolic Variables, or Directly Represented Variables. Directly Represented Variables provide a standard nomenclature for direct access to specific addresses of the I/O and internal memory map of the PLC. All Directly Represented Variables begin with a '%' character, followed by a location prefix, a size prefix, and then a sequence of numbers to indicate the actual location. Some examples of these and their meanings:

- %QX75 Output (Q) Bit (X) number 75
- %IW215 Input (I) Word (W) number 215
- %MW48 Internal (M) Word (W) number 48

The precise meaning of the hierarchy of location numbers is not defined, so it is possible to have constructs like the following, taken from an actual PLC architecture:

%MW3.23.8.12.2.4,

which corresponds in this particular case to, from right to left, the 4th Internal 16 Bit Integer Word located in Module subsection 2 of module 12 of Rack 8, of the unit at drop 23 of MODBUS® Network 3. Directly Represented Variables do not have to be declared. Their use is legal only in Programs and Configurations.

Symbolic Variables do need to be declared. In the case where Symbolic Variables refer to actual input and output points, the declaration assigns them to these points by associating them with the appropriate Directly Represented Variables. Symbolic Variables that do not refer to I/O points need not be assigned a Directly Represented Variable - the IEC 1131-3 language system will assign these an address at compile time.

## 9.7 Ladder LOGIC

Ladder Logic is an instruction set to provide services of real time, I/O, user interface, and similar services. These services are associated with the special requirements of the PLC applications domain. Because Ladder Logic is targeted toward special applications, it provides features that are compatible with real-time control application requirements. These features, when used correctly and appropriately can contribute to the safe operation of the program. The origin of Ladder Logic is the Relay Ladder Logic notation which was first introduced to represent combinations of contacts and coils of relays using specific notation. These combinations implemented logical functions (e.g., AND or OR). The introduction of PLCs transformed Ladder Logic from a hardware design notation to a high level language, specialized for process and logic control. The Ladder Logic language, in the case of the PLC, is not the traditional limited Ladder Logic implemented with relays, but an advanced language supported by the numerical capabilities of the processor, while the Ladder Logic notation

serves only a graphical user interface. Ladder Logic supports all types of programming structures from advanced subroutines, parameter passing, loops, mathematical functions, proportional plus integral plus derivative (PID) controllers, I/O calls, timers, and any other features of a high-level language. Although much changed from their original purpose and implementation, current forms of Ladder Logic are still similar to relay logic, allowing electrical engineering personnel who have traditionally have been in charge of factory automation to review and understand the code. This is an important advantage throughout the development process.

Ladder Logic is not a formally defined programming language. Each manufacturer has its own variation of Ladder Logic. In addition, many of the features associated with programming the PLC are not features of Ladder Logic itself, but the programming environment, the "shell," and the firmware mentioned above. The variety of ladder logic implementations is due to the strong coupling between software and hardware dictated by the requirements of the industrial control applications domain.

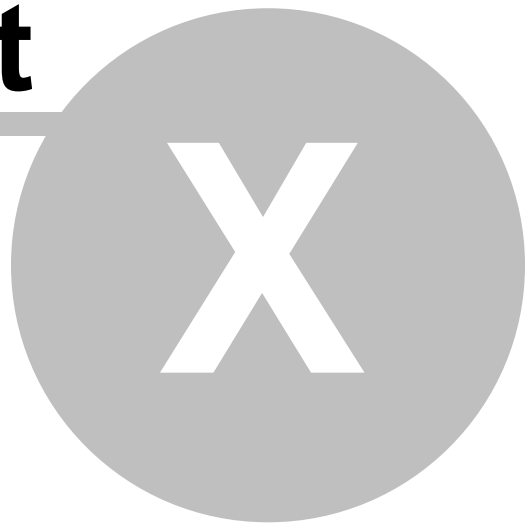
## 9.8 Ladder logic's elements

Ladder Logic programs consist of the following types of elements

- **Power rails:** Ladder Logic networks are delimited on the left and right by vertical lines known as left and right power rails, respectively. The right power rail may be explicit or implied.
- **Link elements and states:** Links indicate power flow in the rungs of the Ladder Logic diagram. A link element may be horizontal or vertical. A horizontal link transmits the state of the element to its immediate left to the element to its immediate right. The state of an element can be either ON or OFF. A vertical link intersects with one or more horizontal links on each side and its state is the inclusive OR of the states of the horizontal links on its left. This state is transmitted to all horizontal links attached to the vertical link on its right.
- **Contacts:** A contact is an element which imparts a state to the horizontal link on its right side equal to the AND of the state of the horizontal link on its left side with an appropriate function. A contact does not modify the value of the associated Boolean variable.
- **Coils:** A coil copies the state of the link on its left to the link on its right without modification, and stores an appropriate function of the state or transition of the left link into the associated Boolean variable.
- **Functions and function blocks:** A function is a program unit which, when executed, yields exactly one result. A function block may yield more than one result. Internal variables of a function or function block are not accessible to users of the function. In Ladder Logic, at least one Boolean input and one Boolean output is shown for each function block to allow power to flow through the block.

# Part

---



## 10 LadderDIP PLC Studio Software

This section introduces you in using LadderDIP PLC Studio

### 10.1 Basic topics

In this section we discuss some fundamental concepts related to LadderDIP PLC Studio software

- [Understanding the IEC 1131-3 / CEI 61131-3 addressing](#)
- [Defining variables](#)
- [Logical links](#)
- [Identifiers](#)
- [Literals](#)
- [The REFERENCE code](#)
- [Mathematical expressions](#)

#### 10.1.1 Understanding the IEC 1131-3 / CEI 61131-3 addressing

This section discuss the IEC 1131-3 addressing identifiers. This kind of identifiers are used in a PLC to address resources. A IEC address always starts with the '%' character.

We can classify two main classes of addressing

- Addresses that refer to a PHYSICAL resource in the PLC (Input, Outputs and so on)
- Addresses that refer to an INTERNAL MEMORY word of the PLC

**IEC variables that refers to a PHYSICAL resource of a PLC are expressed in the generic form**

**% [Q | I ] [X] [adr2] . [adr1] . [adr0]**

Where

'%' is the header of an IEC address

'Q' specify an OUTPUT resource

'I' specify an INPUT resource

The effective address is expressed in a hierarchical form that could consist of more numbers separated by dots. For example the address %QX1.2.3 could express the output bit 3 on the second word of the module 1

The IEC directive do not specify how deep a hierarchy can be, this is related to your particular PLC system

IEC variables that refers to an INTERNAL MEMORY resource of a PLC are expressed in the following form

**% M [ X | W | D ] [adr2] . [adr1] . [adr0]**

Where

'%' is the header of an IEC address

'M' specify a MEMORY resource

The 'X', 'W', 'D' define, respectively, a BOOLEAN, a 16-bits WORD and a 32-bits double WORD

The address hierarchy, like explained before, follow

For example

%MW200	Refers to the WORD (16-Bits) number 200
%MX350.5	Refers to bit number 5 of the word 350
%MD4000	Refers to double word (32-Bits) number 4000
%MX4.207.8	Refers to bit 8 of word 207 of PLC number 4

## 10.1.2 Defining variables

You can define and use variables in different ways



Variables can be referred using two kinds of representation: **Direct Memory addressing** and **TAG Names**. A direct memory addressing is done entering a string which begins with the '%' character and directly refers to a resource into the PLC. A **TAG Name** is a "mnemonic" name you use to "Abstract" a physical or logical resource of your PLC.

### A) Use of a Direct Memory Address %...

LadderDIP PLC Studio supports standard IEC/CEI 1131-3 variable notation which can be used simply typing this kind of identifier in the REFERENCE field of the I/O components ( I/O blocks and variable blocks )

Some examples

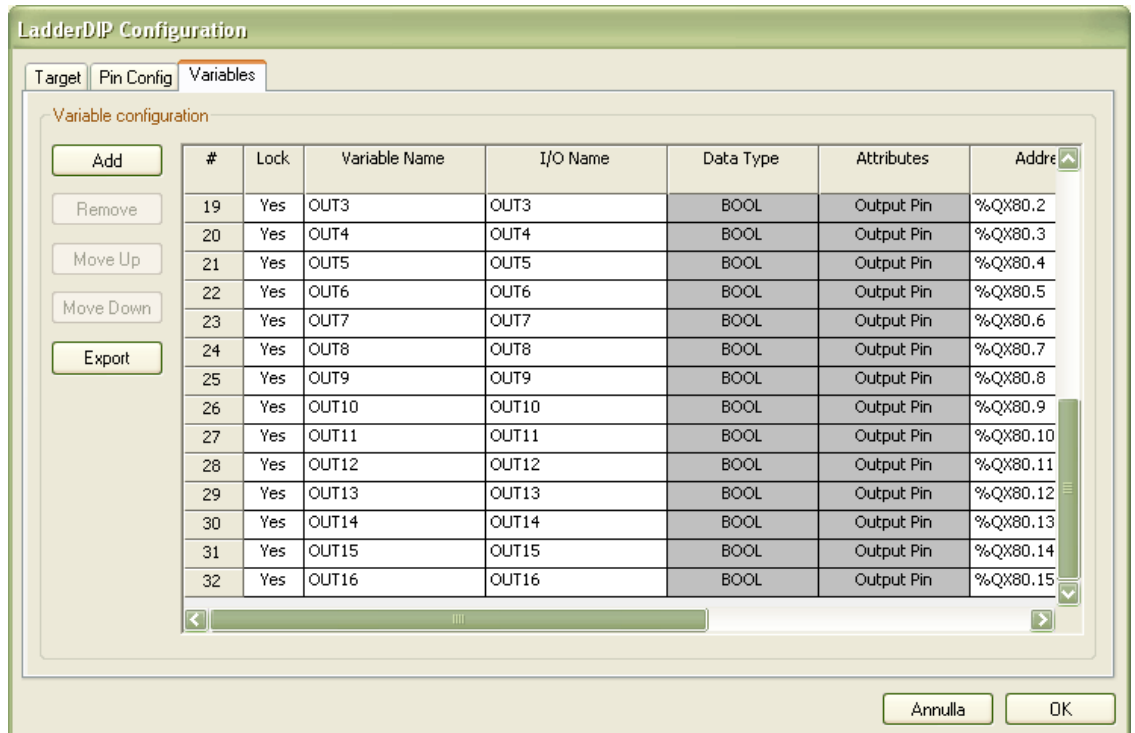
%MW3	Uses the 16-bits word number three <b>Note</b> : Variables are numbered from zero [0..n-1]
%QX2.3	Uses the OUTPUT port located in the bit#3 of the word#2
%IX0.5	Uses the INPUT port assigned to the bit#5 of the word#0

### B) Defining a TAG name (Identifier)

A TAG Name is a "mnemonic" or an [identifier](#) that you use instead of a direct memory addressing and it works like a sort of "Abstraction" between a well readable and understandable

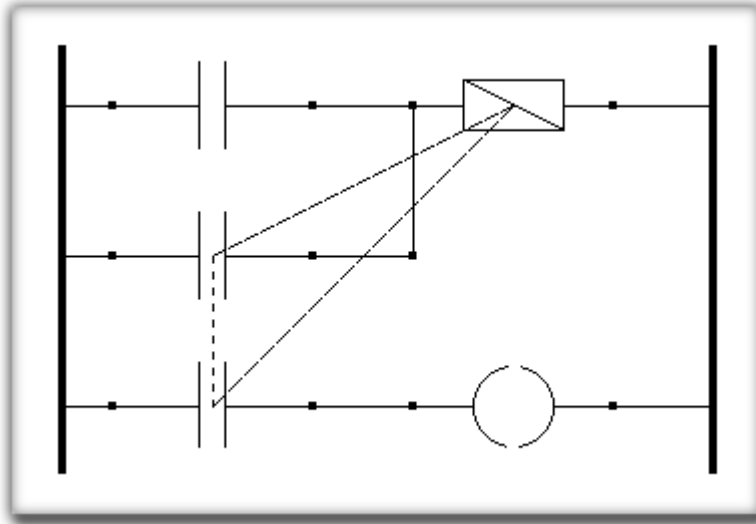
name and a real physical resource of the PLC. Often is useful to have in a diagram a name like "GEAR" or "STOP\_ENGINE" instead of a string like %MW56.6.

Accessing **Options** → **Configuration** from the menu and selecting the **Variables** tab you can access a table where you can add and edit your Tag Names. If you want, for example, to add a Tag named "ENGINE\_ON" which should refer to the physical address %QX2.5, press the Add button and enter the TAG name into the new "Variable Name" column field and the address in the Address field. Then press the OK button to confirm changes.



Once you defined a TAG name you can use it in all the I/O blocks and memory access blocks. This is simply done applying the desired resource in the REFERENCE field

### 10.1.3 Logical links



The LOGICAL LINKS are particular connections which can be done between [RELAY](#), [OUTPUT](#), [EOUTPUT](#) components and [INPUT](#), [EINPUT](#), [NCINPUT](#) and [ENCINPUT](#) devices. For convention we will call the components RELAY, OUTPUT and EOUTPUT the output devices. A LOGICAL LINKS is done assigning to an INPUT device the same [REFERENCE](#) code of an output device. In this way we tell the system to treat the output and the input as an unique object. The value assumed by the output device will be assumed also for all the input devices linked with. With the LOGICAL LINKS method it is possible to drive several sections in the net starting just from a master signal. In the example below we connected two inputs on a RELAY. The first input activates the RELAY and the second input, linked with the same RELAY, will sustain the RELAY activation. In the same way another input is logical linked to the RELAY so a generic output can be driven starting from the same signal. There's no limit on the number of the input that you can associate to an output device. In the LadderDIP PLC Studio software the LOGICAL LINKS are showed as a dashed line. This dashed line joint the components. The displaying of the LOGICAL LINKS is configurable by software.

### 10.1.4 Identifiers

**According to CEI/IEC 1131-3: 1993 - 2.1.2 :**

An identifier is a string of letters, digits, and underline characters which begin with a letter or underline character.

Underlines are significant in identifier, for example "LADD\_ER" and "LAD\_DER" are interpreted as different identifiers. Multiple leading or multiple embedded underlines are not allowed.

Identifiers must not contain the space character.

LadderWORK ARM supports 15 character of uniqueness)

Examples

No.	Feature description	Examples
1	Upper case and numbers	LW218 LW218T QX567 MYIDENT
2	Upper and lower case, numbers, embedded underlines	<b><i>The above example plus:</i></b> MY_IDENT_15 My_Ident_15 abcd ab_cd
3	Upper and lower case, numbers, leading or embedded underlines	<b><i>The above example plus:</i></b> _LEADING _12345



## 10.1.5 Literals

With LadderDIP PLC Studio user can define constants using the standard IEC / CEI 1131-3 notation. We can distinguish numerical constants and time constants. A literal constant must be used with the [IDENT](#) component.

### Base 2, base 8 and base 16 notation

Numbers in base 2, 8 and 16, start respectively with the prefix 2#, 8# and 16#. The number, expressed in the specified base follow. For clarity, the underscore character '\_' could be used to separate part of the number. This is useful, for example, for binary numbers. Numerical constants

### Examples

100	Decimal 100
+100	Decimal 100
2#10000001	Hex 81, Decimal 129
2#1000_0001	Hex 81, Decimal 129
8#10	Octal 10, Decimal 8
16#FFFF	Hex FFFF, Decimal 65535
-276	Integer decimal -276
65535	Integer decimal 65535
TRUE	1
FALSE	0

### Time constants

A time literal is used whenever you need to program a specified time. A time literal always begin with one of these prefix T#, t#, TIME# or time#. After this prefix a time specification string follow. A lot of standard suffix defines the time in the desired unit. Between single time specifier you can use the underscore character '\_' to increase literal readability.

Time unit suffix	Meaning
ms	milliseconds
s	seconds
m	minutes
h	hours
d	days

### Examples

TIME#14.73ms	Time duration specification 14.73 milliseconds
time#14.73ms	Time duration specification 14.73 milliseconds
t#14.73ms	Time duration specification 14.73 milliseconds
T#14.73ms	Time duration specification 14.73 milliseconds
t#6m_34s_15ms	Time duration is : 6 minutes, 34 seconds and 15 milliseconds
time#1h30m	Time is : 1 hour and half
TIME#4d_5h_34m_10s_25ms	Time is : 4 days, 5 hours, 34 minutes, 10 seconds and 25 milliseconds

### 10.1.6 The REFERENCE Code

The REFERENCE is a particular code that can identify an object inside the schematic. The REFERENCE must be supplied for all the I/O blocks and memory access blocks like INPUT , EINPUT , NCINPUT , ENCINPUT , OUTPUT and EOUTPUT. In fact, this device must know where to put/get the signal to be treated. The REFERENCE field must be configured through the property dialog associated with the component. For the mentioned items, the system gives the possibility to select one of the available PLC resources. For all the other components the REFERENCE is optional and the software automatically assigns a unique reference code for all the components in the schematic . The REFERENCE code gives the possibility to obtain particular configurations called LOGICAL LINKS .

**See also :** [INPUT](#) , [EINPUT](#) , [NCINPUT](#) , [ENCINPUT](#) , [OUTPUT](#) , [EOUTPUT](#) , [LOGICAL LINKS](#) , [RELAY](#) .

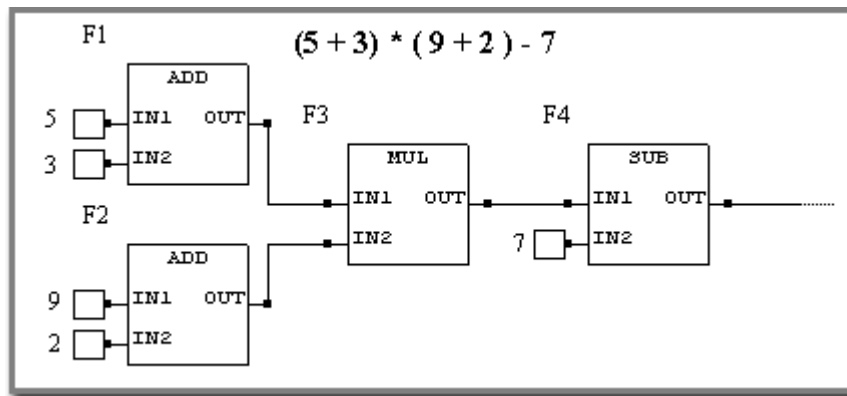
### 10.1.7 Mathematical expressions

#### Entering formulas using function block format

Formulas can be entered in the schematic using the FUNCTION BLOCK notation. The figure below, for example, represents the way to compute the formula  $(5 + 3) * (9 + 2) - 7$ .

LadderWORK ARM software supplies the classical arithmetic functions like ADD, SUB, DIV, MUL and MOD. Moreover the software gives you the possibility to operate with the logical operators like SHR, SHL, ROL, ROR and BIT.

Constant values can be entered using the [CONST](#) and [IDENT](#) components



## 10.2 LadderDIP PLC Studio Tutorial

In this section you will learn about

- [What LadderDIP PLC Studio is?](#)
- [How does LadderDIP PLC Studio work?](#)
- [Basic diagram elements](#)
- [A first project, the basic I/O transfer](#)
- [And now, blink the first led in 1 minute](#)
- [A more complex project, a basic counter](#)
- [The "Walking One"project](#)
- [Analog threshold with hysteresis](#)

### 10.2.1 What LadderDIP PLC Studio is

LadderDIP PLC Studio is a program development application, much like various commercial C or BASIC development system. However, LadderDIP PLC Studio is different from those applications in one respect. Other programming system use *text-based* languages to create the appropriate lines of code for each particular family of **PLC**, while LadderDIP PLC Studio uses *graphical* programming language, compliant to **IEC 1131-3** / **CEI 61131-3** symbolism, to create programs.

You can use LadderDIP PLC Studio with little programming experience. LadderDIP PLC Studio uses terminology, icons and ideas familiar to engineers and electronic technicians and relies on graphical symbols rather than textual language to describe programming actions.

LadderDIP PLC Studio has an extensive library of Ladder language idioms so it can match easily to different automation systems. LadderDIP PLC Studio includes conventional program development tools, so you can see how data passes through the program and make debugging and program development easier.

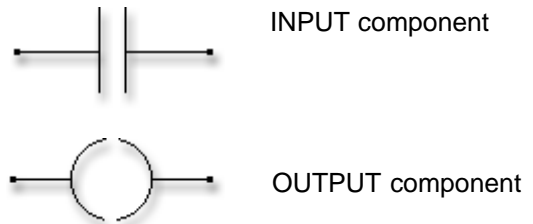
### 10.2.2 How does LadderDIP PLC Studio work?


LadderDIP PLC Studio programs are a mix of two languages called LD ( Ladder Diagram ) and FBD ( Function Block Diagram ). Their appearance and operation imitate electrical circuits. However, they are analogous to functions from conventional script language programs. Each of the possible elements of the component library represents a module of the program. When the user selects and places a new component on the actual project layout connecting it by wires to the circuit automatically he links the correlated program module to the program under construction. During the compiling phase the diagram is then translated in a microprocessor-compatible program which can be sent and stored to the PLC


### 10.2.3 Basic diagram elements

In this paragraph we'll introduce the basic elements of a Ladder Diagram. Essentially the Ladder language take concepts, symbolism and philosophy directly by a real electrical circuit. As you can see in the pictures there are basic elements that directly represents the same functions of a hardware switch and a real lamp. These objects are normally referred as Input/Output objects.

Let us introduce to you our first two very basic components. These components ALWAYS are present in any Ladder environment. This objects are called INPUT and OUTPUT

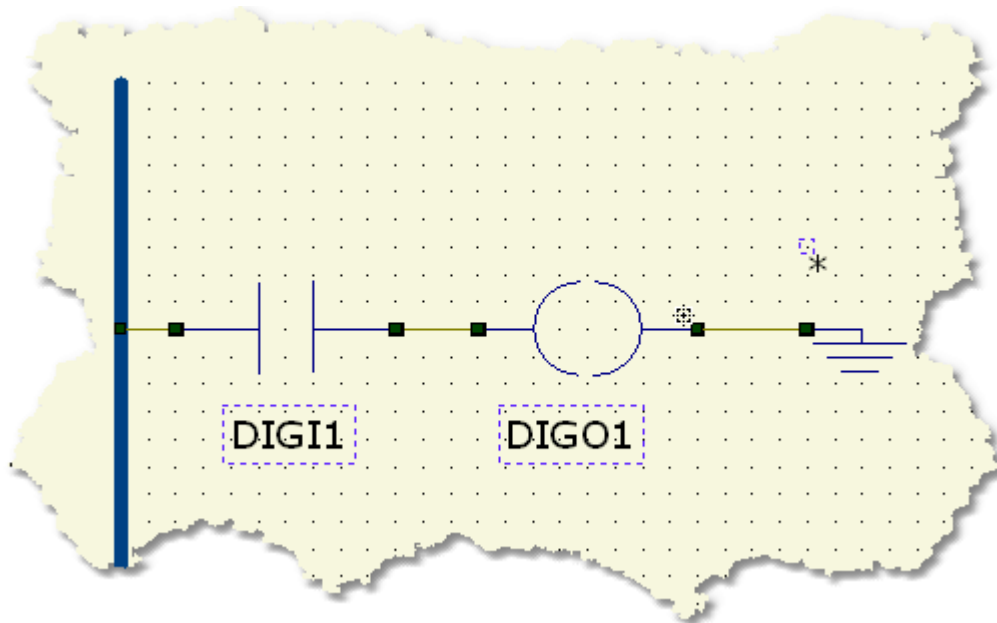


 The INPUT component is the object that allow you to ENTER an information to the PLC. This information, in this case, is a BOOLEAN value (0/1 or TRUE/FALSE) and it comes from a real electrical signal of the PLC hardware. The INPUT block reflects the value applied on the left-side pin on the right-side pin when the associated signal (We'll discuss about it later) is TRUE or ASSERTED.

 The OUTPUT component is the object that allow you to EXIT an information from the PLC. Like the INPUT block, this information is always a BOOLEAN information and it will cause the associated physical output to reflect the value of the left-side pin. The right-side pin of this component simply reflects the value of the input pin (left-side)

### 10.2.4 A first project, the basic I/O Transfer

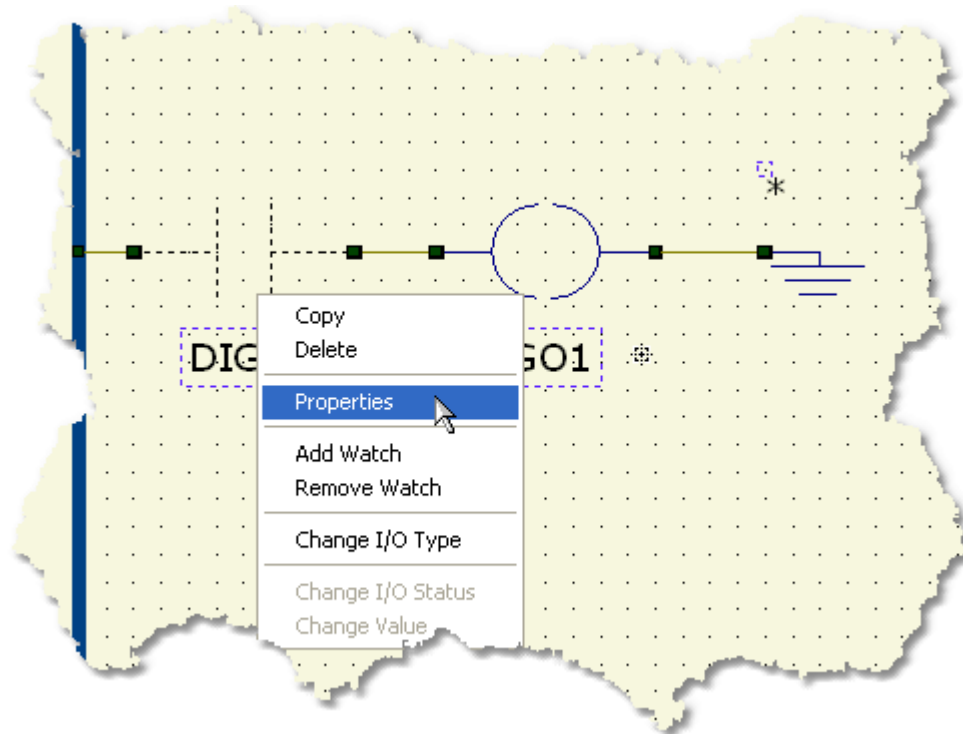
At this point, once we understood the function of the basic I/O blocks we can easily start with our first project. This project aims to transfer an input of the PLC to an output. The signal applied on the desired INPUT will be transferred to the configured OUTPUT. The project consist of three components only



The diagram in the picture is also called "RUNG". A Ladder diagram can be formed by an undefined number of rungs. If you take a look to the picture you can see other important elements that enter in play in this kind of language. The BAR present on the LEFT of the schematic is called the POWER BAR and it represents the "source of electricity" for your elements. Like any real electrical circuit this object represent the way to supply power to the components.

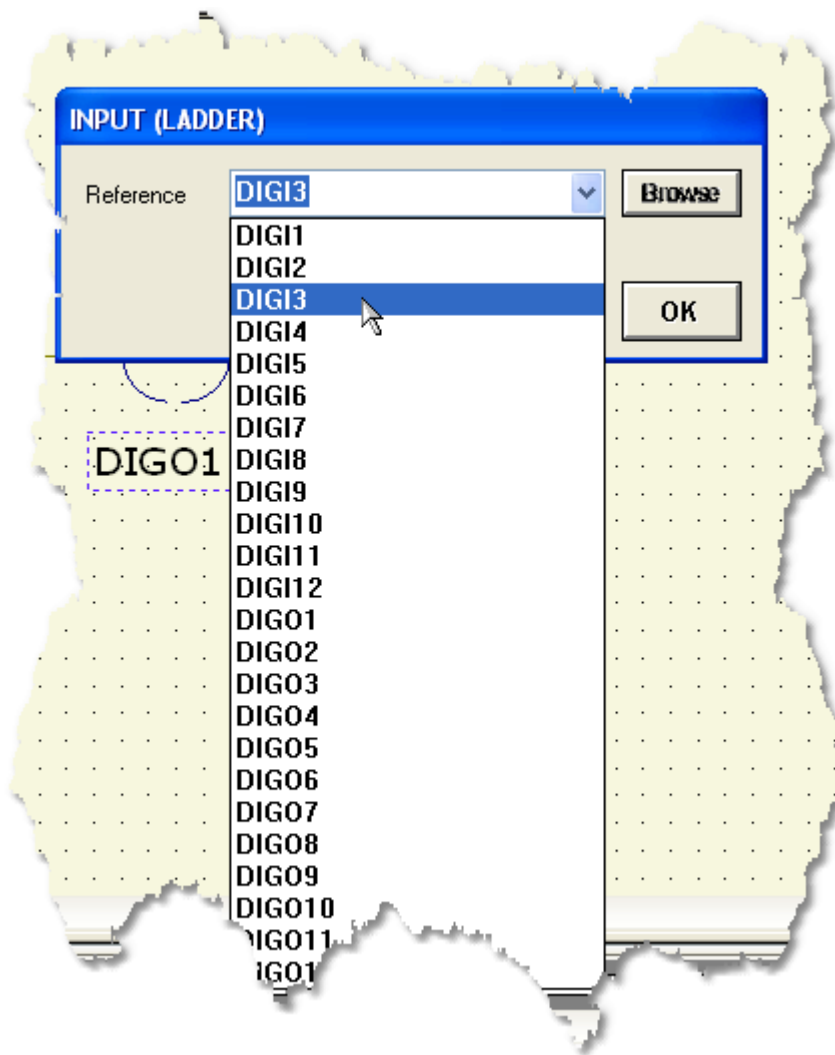
The other important element is the GROUND component (GND). This component, always like you do in a regular electrical plant, allow you to "close" the circuit and allow the current flow.

At this point the next step is to associate the INPUT and OUTPUT block to a physical signal on the PLC. This procedure is accessible selecting the **Properties** entry from the menu related to the component. This is activated selecting the component with the left mouse button and clicking on its symbol with the right mouse button.




This procedure allow you to enter the configuration Dialog of a component. For component like INPUT and OUTPUT the software allow you to select a PHYSICAL signal of the PLC. This signals are accessible through mnemonic names that you can change for your purposes.





You have to configure both the INPUT and OUTPUT block associating, respectively, an INPUT signal and an OUTPUT signal. Press the OK button to confirm any changes

This tutorial does not consider the final hardware you are working on so refer to your hardware manual for PLC setting-up and connections.

Once everything is ready you can press the Build All button  (Also indicated as **Build & Upload & Run** in the build menu command), this will execute all the necessary operations to download and run a program into your PLC. The executed processes will be: Compile, connect, erase, download and run. These processes will be accomplished by dialogs and progression bar to check the current status of operations

Once the program is correctly loaded and running you can apply the input signal to the configured physical input and see the signal reflected to the specified output.

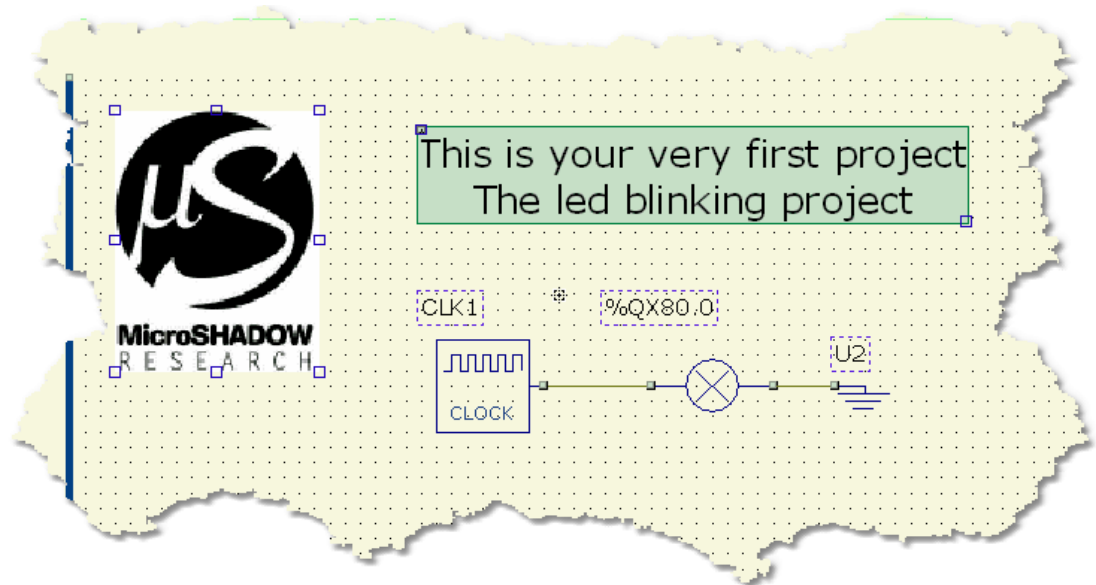
For further information about these components see [INPUT](#) and [OUTPUT](#) .

### 10.2.5 And now, blink the first led in 1 minute

This section will lead you to run your very first project. The objective of this project is to blink a led using a very simple project called **Blink**.

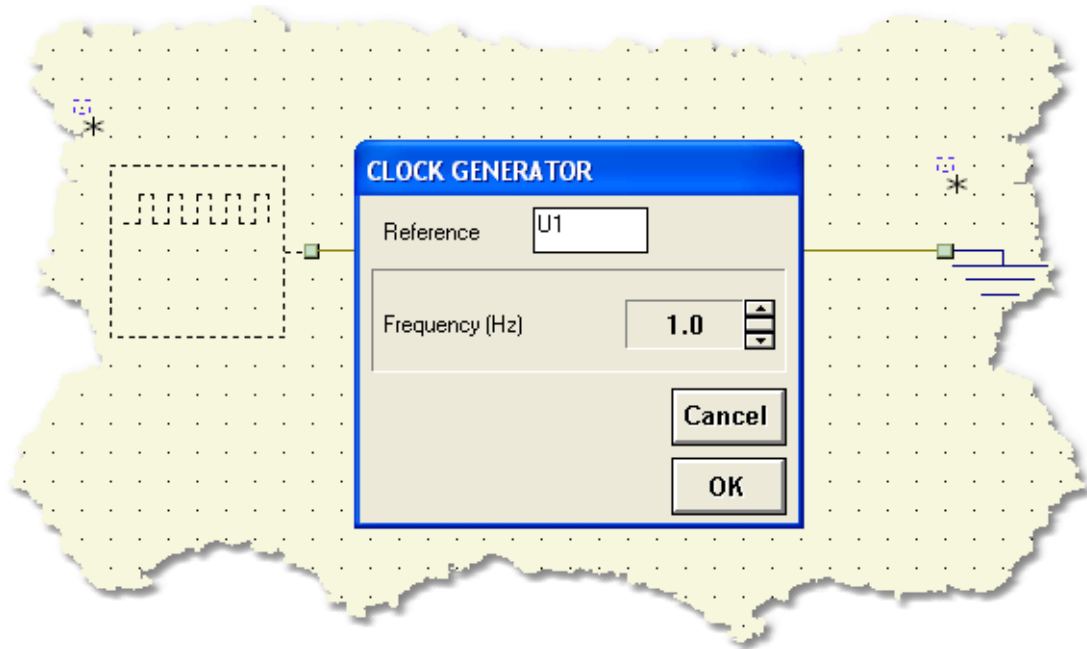
The Blink project is almost present in all LadderDIP PLC Studio distribution and it is normally present under the samples directory.


Open the Blink project "..\projects\samples\ladderdip4\ledblink\ledblink.lww", a diagram similar like this should appear



The project uses three only components. The first component, the CLOCK, generates a square wave pulse stream. The second component, attached to the CLOCK output, is a electrical generic output block (EOUTPUT). The output block is associated with an IEC address (%...) or a tag-name

The frequency of the CLOCK component can be up to 10Hz (The dot point is the fraction of Hertz). If you enter, for example, 1.0 you will get a one hertz square wave to the output. This signal is then injected to the output block which will drive the preferred PLC physical output.



Once everything is ready you can press the Build All button  (Also indicated as **Build & Upload & Run** in the build menu command), this will execute all the necessary operations to download and run a program into the PLC. The executed processes will be: Compile, connect, erase, download and run. These processes will be accomplished by dialogs and progression bar to check the current status of operations

If everything is ok you will see the project running. You can see the configured OUTPUT blinking, and, if the OUTPUT is physically attached to a RELAY will also hear the device tick ... enjoy it

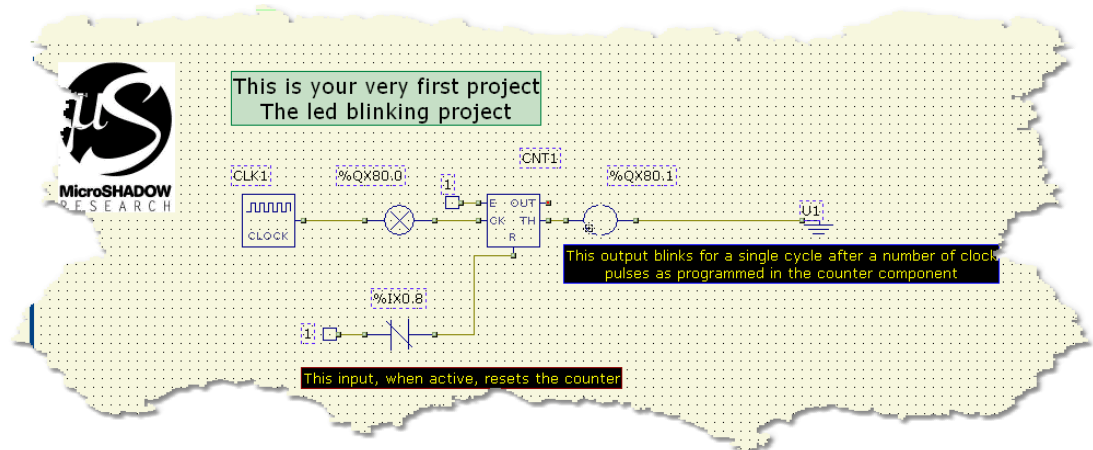
Click here for more information about the [CLOCK](#) block

### 10.2.6 A more complex project, a basic counter

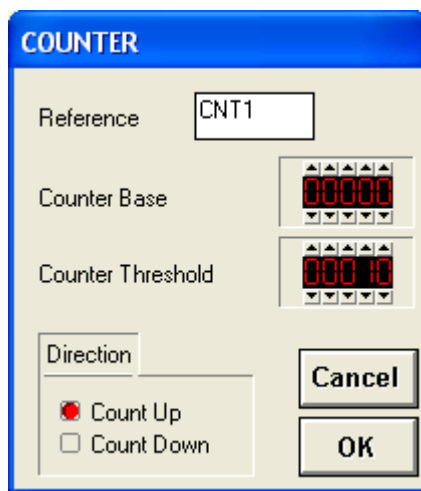
Now that we are familiar with many elements we can start with a more complex project. This diagram introduces a new component called COUNTER. This object allow to count pulses applied to its input giving the total count value on the output. The COUNTER object acts like a "ring counter". Accessing the properties of this component you can configure this block to count in a predefined range (Allowable values are from 0 to 65535).

The enable (E) input allow you to enable or disable the counting. The reset (R) input acts like an asynchronous reset of the count. The threshold (TH) pin become TRUE when the counting value reach the maximum programmed value. This object can be programmed to count up or down.

In the diagram there are other new elements we are going to describe. The first component is called [IDENT](#) and allow you to enter constant values in the diagram. In this case this object is used to enter the value 1 (TRUE) to the enable input of the counter, this makes the COUNTER working properly. The IDENT component, when configured with the value 1, is also equivalent to the left power bar of any Ladder diagram. Often programmers prefers to use this method allowing less wiring and a more clear schematic. As you can see on this project the IDENT component is also used to feed a [NCINPUT](#) (Normally closed INPUT) on its left pin. This little piece of diagram allow to apply a reset to the COUNTER



The COUNTER properties Dialog looks like here



Now, what we would aim from this project is to divide a certain number of pulses by 11. The diagram is very intuitive and since you are now very well-versed with this language you already understood what you will get.

A CLOCK component is feeding an OUTPUT block that will flash the programmed signal to the base programmed frequency (For example 1Hz). The right-side pin of the OUTPUT block transfer the same square-wave signal to the clock input of the COUNTER. The threshold output of the COUNTER is connected to a second OUTPUT and it will be ON when the counter will reach the value 10. Practically this works like a by-11 divisor. Once the value 11 is reached the counter will reset to zero again and a new cycle will be started.

What we expect by this project?

Once you load and run this project on your PLC you will see the first OUTPUT blinking at 1Hz frequency. After 10 pulses you will see the 2nd OUTPUT become ON and after a pulse it returns to OFF again. The cycle is repeated forever ... at least if you don't remove the power from your PLC ...

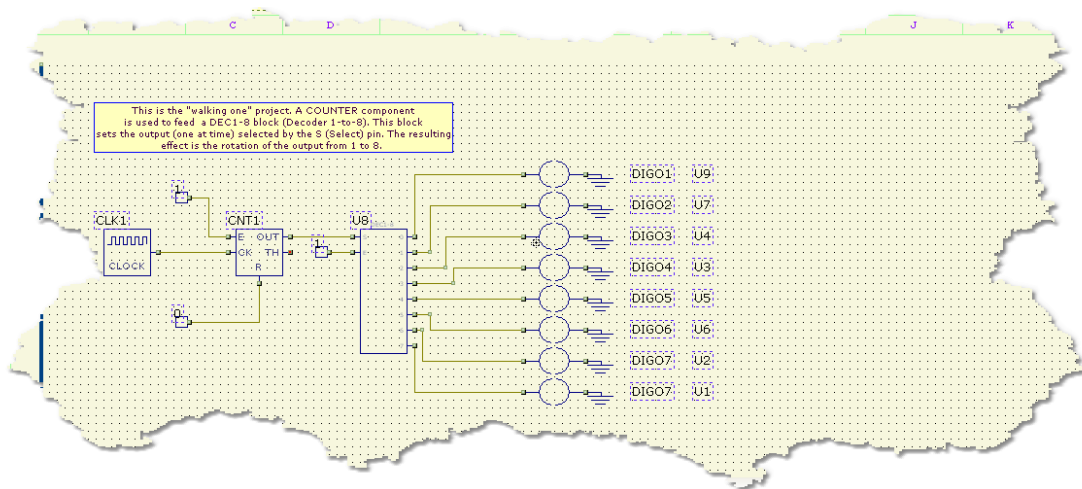
Click the link for a better description of the [COUNTER](#) component

### 10.2.7 The "Walking One" (Led Chaser) project

This more complex project show you how to rotate eight outputs (Led Chaser). In this project we introduce a new component called [DEC1-8](#). This component acts like any standard DECODER allowing you to transfer a boolean information from the E (Enable) pin to the desired output pin which is selected by the S (Select) pin.

The diagram is really simple. You are very familiar with the CLOCK and the COUNTER components so you easily understand the left side of the circuit.

The CLOCK is generating a square wave that feeds a COUNTER programmed to count in the range 0-7. This value is now available to the OUT pin of the COUNTER and injected to the S (Select) pin of the decoder. The decoder will transfer the value of the E (Enable) pin, which is granted to be one using the IDENT component. Eight OUTPUT block are connected to the respective outputs of the decoder.



Now, let's take a look to the object properties. The CLOCK generator is programmed to generate a 10Hz square wave and the COUNTER component to count UP from 0 to 7

**CLOCK GENERATOR**

Reference:

Frequency (Hz):

**COUNTER**

Reference:

Counter Base:

Counter Threshold:

Direction:

☒ Count Up

☐ Count Down

Ok, now we are ready to run our project. Press the build all button and see the rotating effect

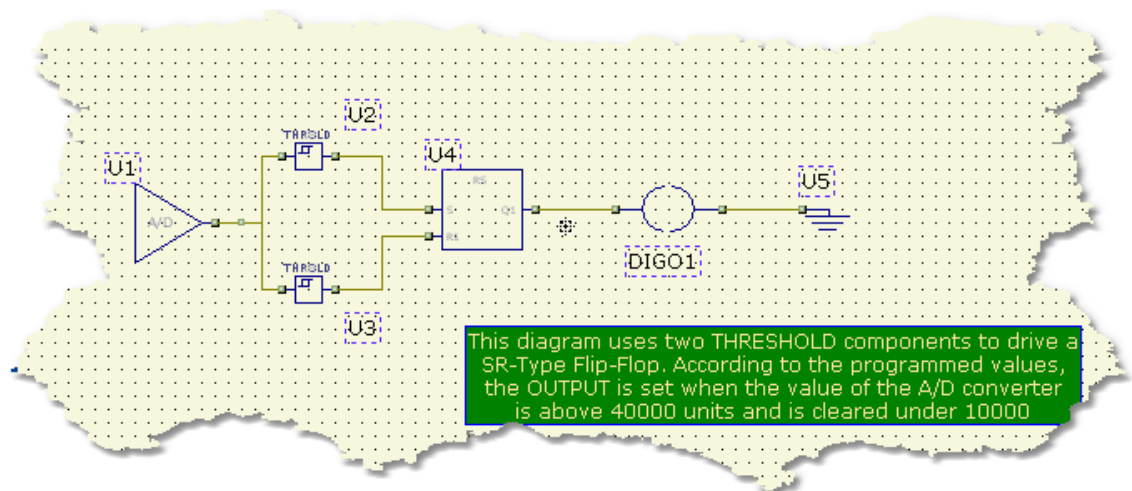
### 10.2.8 Analog threshold with hysteresis

In this application note we introduce other important components like the [AD\\_CONV](#) (Analog to Digital Converter) and the [THRESHLD](#) (Threshold). The diagram shows how to create a threshold control with hysteresis circuit. The circuit is really simple and it takes just five components. The AD\_CONV component gives, on its single output, the numeric value converted by a physical A/D converter on the PLC. The value is applied to two different comparators ( THRESHLD component ) to create the threshold gap.

For example, the high-side comparator (U2) was programmed to be true when the input signal is greater of 40.000 and the low-side comparator (U1) was programmed for values less than 10.000.

Remember that the values supplied by AD\_CONV components are always normalized in the range 0-65535 independently by the A/D circuit resolution. The signals coming from the comparators are connected to a [SR](#) block ( SET-RESET FLIP-FLOP WITH RESET DOMINANT FEATURE ) and the Q output of this function block is suitable for load driving.

With respect with the example programmed values , the Q output will become active for values greater than 40.000 and will be released for values less than 10.000



Here are the properties of any component

**THRESHOLD**

Reference: U2

Compare Value: 40000

Qualifier:

- ☐ =
- ☐ <>
- ☐ >=
- ☐ <=
- ☒ >
- ☐ <

Cancel OK

**THRESHOLD**

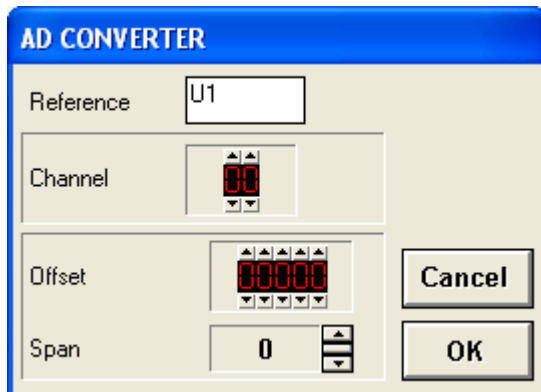
Reference: U3

Compare Value: 10000

Qualifier:

- ☐ =
- ☐ <>
- ☐ >=
- ☐ <=
- ☐ >
- ☒ <

Cancel OK



As you can see the THRESHOLD component can be programmed to set its output according to the specified condition. In this case the U2 component will activate its output for values above 4000 and U3 will be set on for values under 10000.

Note the the CHANNEL parameter in the AD\_CONV Dialog properties selects from which physical A/D converter take the analog value. The OFFSET and SPAN parameters are useful to change A/D dynamics and gain

At this point you can run the project. Changing voltage, in the A/D specified range, will cause the OUTPUT to be ON or OFF according to the analog value. In the RANGE 10000 to 30000 there is a dead-band where no changes are applied (Hysteresis gap)

Note that this circuit is the typical Water level control. The A/D converted value can be the level of the water and the OUTPUT can be connected (In negated state) to a pump which will control the flow inside the tank. When the water level is under "10000" the pump is run and the water begin to flow into the tank. The process is stopped when the water level reach a value of "40000".

## 10.3 Integrated Development Enviroment (IDE)

This section discuss how to operate with the Integrated Development Enviroment (IDE). The IDE is lauched executing the LadderDIP PLC Studio executable program which is named LWARM.EXE .



### 10.3.1 System requirements

What you need to install LadderDIP PLC Studio software is listed below.

- Personal computer with Windows OS (Minimum Version: Windows XP SP2 )
- Ethernet port for communication
- USB port for Dongle Key Protection (Only for Dongle Protected Licenses)

### 10.3.2 Installing the software

Software installation procedure is different depending on which version you have.

#### **Installing from CD-ROM**

Open the computer resources icon and select your CD-ROM drive. Run the program called **setup.exe** present on the root directory of the CD-ROM and follow the instructions of the installation program.

#### **Installing from Self-Extracting file.**

Run the program called **setup.exe** and follow the instructions of the installation program.

### 10.3.3 Dongle Protection Key

If you have a full functional version of LadderDIP PLC Studio in your kit you could have an USB Dongle Key protection. This device does not require any driver or software installation. Just plug in your dongle into a USB port. The Dongle Key is driver-less

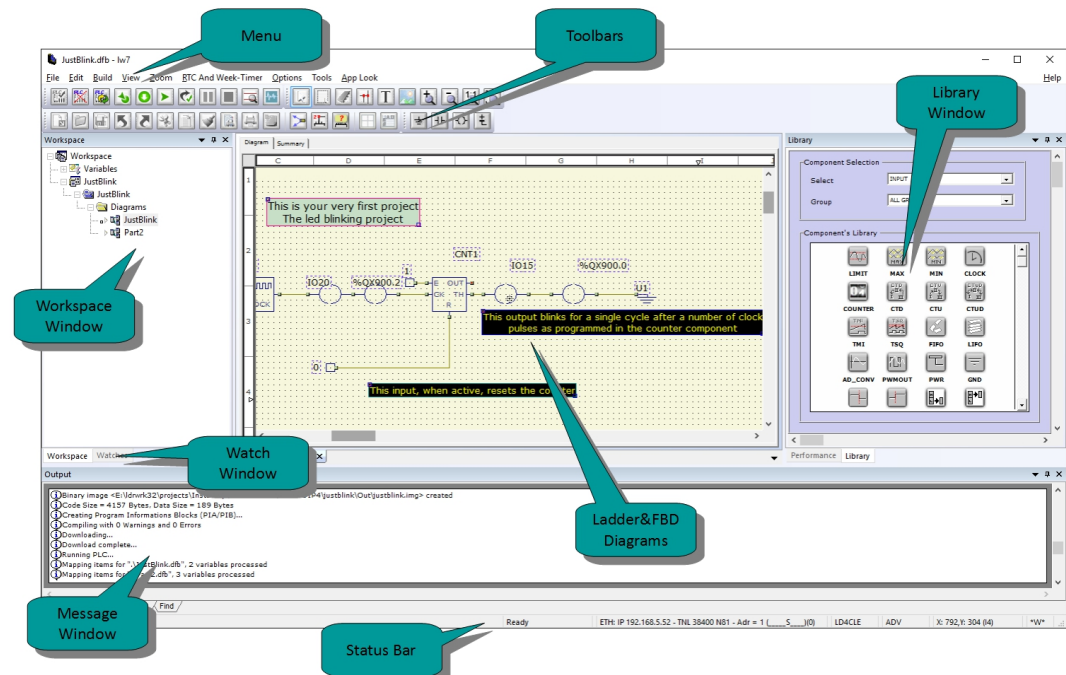
### 10.3.4 Launch the program

LadderDIP PLC Studio software can be launched using the **START** menu of Windows 95/98/me/NT4/2000/XP/Vista/7 operating system, selecting the program entry LadderDIP PLC Studio and choosing the executable program LWARM

With Windows operating system you have the possibility to create short-cut icons on your desktop screen to facilitate program launch operations.

### 10.3.5 Integrated development environment

The picture below, represents the appearance of the program LadderDIP PLC Studio on your computer. LadderDIP PLC Studio has an integrated environment feature, allowing you to draw schematics, compile programs and upload code to PLC always working on the same window. The integrated environment are composed by several parts described below.



#### *Integrated environment overview*

##### **Menu**

The menu is a Windows standard menu.

##### **Toolbars**

The toolbars are the window you can use to have a quick access to the main functions like Connect, compile and download

##### **Message Window**

The message window is mostly used during compiling phase to give you informations about errors

##### **Workspace Window**

The workspace window contains, in hierarchical form, the structure of your project

##### **Components bar**

This floating window contains the components that you can place in the schematic. The components are grouped for functionality

##### **Status bar**

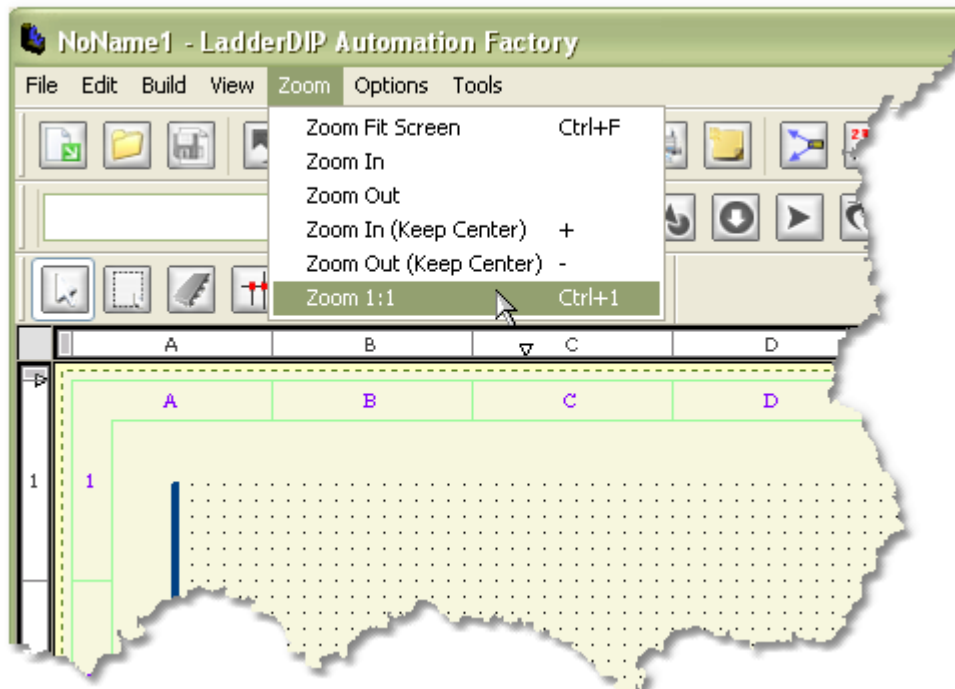
This window contains information about the software status and configuration.

##### **Watch window**

The watch window allow you to monitor variables during PLC running. The watching feature is related to the PLC where you are working on.

### 10.3.6 The menu

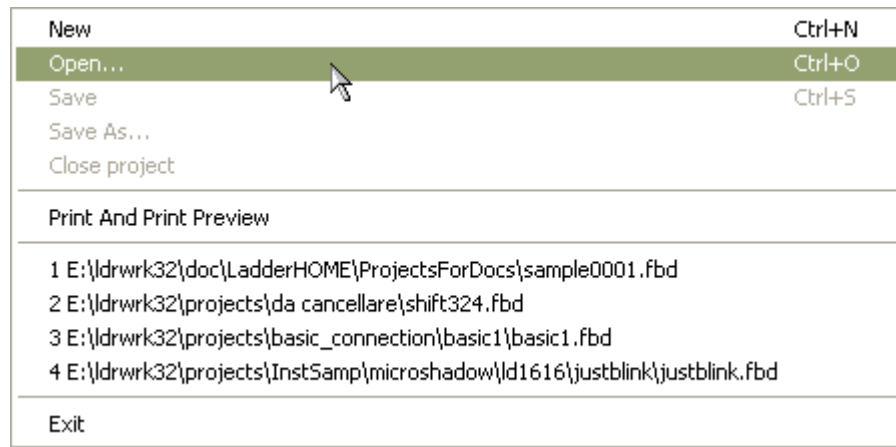
LadderDIP PLC Studio menu, is a Windows standard menu. Use the mouse to select a menu entry and click with the mouse left button to select an option. A shortcut key may be present near the menu command. Use shortcuts to entry commands using the keyboard.



*The menu*

### 10.3.7 File menu

The file menu allow to execute the main program functions as open and save projects.



*The file menu*

#### **New** (Shortcut: Ctrl+N)

This command allow you to start a new project file. If any project is opened, the software ask for confirmation before discharge changes. This command automatically assigns a default name for the new diagram

#### **Open** (Shortcut: Ctrl+O)

Select Open to load an existing project file

#### **Save** (Shortcut: Ctrl+S)

This command will save the project using the current assigned name

#### **Save As**

This command gives you the option of specifying a file name for the current project saving

#### **Close Project**

This command closes the current project

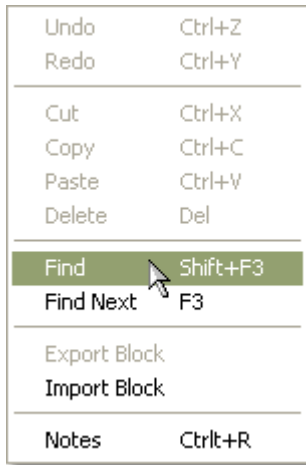
#### **Print And Print Preview**

This option allow you to enter in the Printing Environment of the software

#### **Exit**

Exits the program

### 10.3.8 Edit menu



*Edit Menu*

#### **Undo** (Shortcut: CTRL+Z)

The Undo command restores the schematic to the state before the last action you performed. The Undo queue is five operations deep.

#### **Redo** (Shortcut: CTRL+Y)

The Redo command restores the schematic to the state before the last Undo action you performed.

#### **Cut** ( Shortcut: Ctrl+X )

With the Cut command you can move in the global clipboard a single object or a group of objects from the schematic. To move a single object select the object pressing the left button of the mouse when you are above the object then perform a Cut command. To move a group of object first select the objects with the select tool then use the Cut command.

#### **Copy** ( Shortcut: Ctrl+C )

The Copy command allows you to copy a single object or a group of objects in the private clipboard. To copy a single object first select the object pressing the left button of the mouse when you are above the object then perform a Copy Command. To copy a group of object first select the objects with the select tool then use the Copy command.

#### **Paste** (Shortcut: Ctrl+V)

The Paste command pastes a single object or a group of objects in the worksheet previously copied with the **Copy** command

#### **Delete** (Shortcut: Canc/Del)

The Delete command has the same behaviour of the **Cut** command

#### **Find** (Shortcut: Shift+F3)

This command allow you to find occurrences of a particular strings in the items present in the diagram. The search results are shown in the Find tab of the message window. This command is often used to search a particular REFERENCE identifier to locate a component in the schematic.

#### **Find Next** (Shortcut: F3)

This command can be issued after a previous **Find** command allowing you to move on the next occurrence which honour the established conditions.

**Notes** (Shortcut: Ctrl+R)

This command allow you to enter project notes.

### 10.3.9 Build menu

**IMPORTANT :** Remember that the build commands are available only if you have assigned a name for your project. That means that you have to place at least one component on the sheet and save the project



Compile	F5
Upload	F6
Run	F7
Stop	F8
<hr/>	
Compile & Upload	F9
Compile & Upload & Run	F10

**Build menu****Compile** (Shortcut: F5)

With the Compile command you activate the Compile Process. See the section Building the code for further information.

**Download** (Shortcut: F6)

Many PLC models supported by LadderDIP PLC Studio software have the remote control feature, so you can upload the code directly from the integrated environment. If your PLC doesn't have this feature you can't use this command.

**Run** (Shortcut: F7)

If your PLC supports a remote control feature you can run the PLC simply by executing this command.

**Stop** (Shortcut: F8)

If your PLC supports a remote control feature you can stop the PLC simply by executing this command.

**Compile & Upload** (Shortcut: F9)

The Compile & Upload command execute in sequence the Compile and the Upload sessions.

**Compile & Upload & Run** (Shortcut: F10)

This command execute in sequence the Compile, Upload and Run sessions.

### 10.3.10 View menu

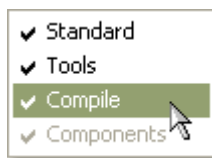
View menu allow to enable or disable some elements present in the IDE and in the sheet where you draw the design.



*View menu*

#### **Toolbar**

This menu entry opens the following sub-menu.



*Toolbar sub-menu*

#### **Standard**

This command enables/hides the Windows standard tool bar. The standard toolbar is the bar where you can find the traditional windows commands like new, load, save and so on.

#### **Tools**

This menu entry enables/hides the tool bar window. The tool bar is the window where you can find the tools for placing components, wires, text and bitmaps. In this bar you can also find the commands for change zoom factors in the current view.

#### **Compile**

Checking/Unchecking this entry will enable/disable the compile bar window. The compile bar contains the command to build the code and other commands for PLC controlling like upload, run and stop.

#### **Components**

Checking/Unchecking this entry will enable/disable the components/library bar window. This window contains the components that you can place in the schematic.

#### **Status Bar**

This menu entry allow you to show or hide the IDE's status bar

#### **Grid**

The grid check enables/hides the grid in the worksheet.

#### **Reference Grid**

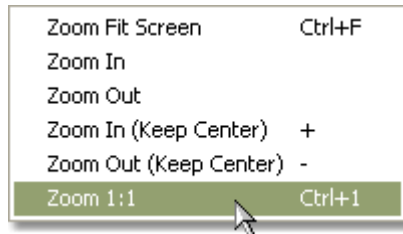
Checking/Unchecking this entry will enable/disable the reference grid in the worksheet .

#### **Watch Sections (Shortcut: F4)**

If your software version handles the variable watching this menu allow you to enter the Watch

Sections Dialog.

### 10.3.11 Zoom menu



*Zoom menu*

#### **Zoom Fit Screen** ( Shortcut : Ctrl+F )

This command will fit your schematic in the current view

#### **Zoom in**

This command performs a Zoom-in

#### **Zoom out**

This command performs a Zoom-out

#### **Zoom In (Keep Center)** (Shorcut: +)

This command performs a Zoom-in keeping the current center point

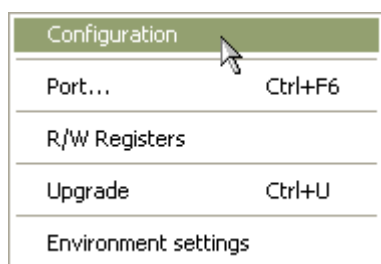
#### **Zoom Out (Keep Center)** (Shortcut: -)

This command performs a Zoom-out keeping the current center point

#### **Zoom 1:1** (Shortcut: Ctrl+1)

This command restores the current view to the default zoom factor

### 10.3.12 Options menu



*Options menu*

#### **Configuration**

Entering this section will allow you to configure many stuffs related with the PLC environment: Variables, data types and so on.

#### **Port** (CTRL+F6)

This command allow you to enter the Port configuration Dialog. These parameters are related



to the port used to communicate with the PLC which allow you to perform commands like Download/Play and Stop.

#### **R/W Registers**

Some PLC supported by this software allow you to perform on-the-fly variable reading/writing. This command opens the dialog related to this feature.

#### **Upgrade** (Shortcut: CTRL+U)

This command enters the License/Upgrade Dialog

#### **Environment settings**

With this command you can open the dialog where you can modify many stuffs like colors and software attributes

### **10.3.13 Placing and modifying components**

#### **Placing components**

To place a component, select a object in the component bar. Automatically the place tool will be selected. The shape of the selected object is shown if you move the mouse in the worksheet area. Placement is made by clicking with the left button on the mouse at the selected point. There are regions in the worksheet where the component can't be placed. For example you can't overlap the new component with an existing component. You can understand if the current position is right for placement observing the cursor. If an NO-ACCESS symbol appear near the cursor then the components can't be placed there.

#### **Moving Components**

In order, to move a component the following operations should be executed. First choose the pointer in the tools bar. Now select a particular object in the schematic clicking with the left button on the mouse onto the component. Always keeping the left button pressed now you can move the components anywhere in the sheet. The wrong positions are always indicated with the NO-ACCESS symbol. The component may be placed in the new position simply releasing the mouse button.

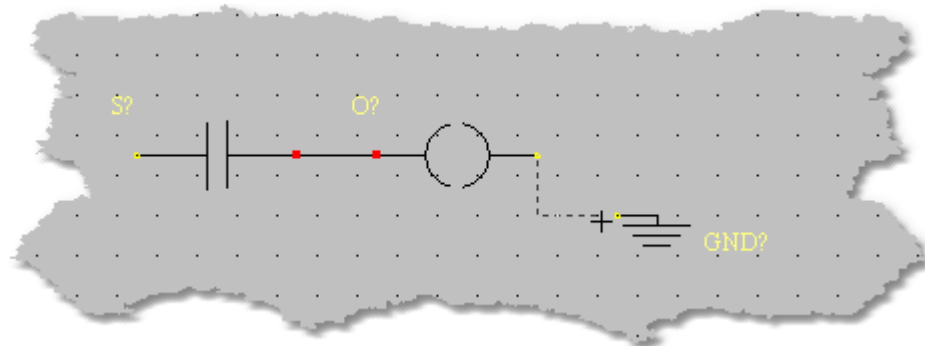
Groups of components can be moved executing the following operations. Select the pointer or the area tool in the tools bar. Now, keeping the left button pressed on the mouse select a region including the desired components. Releasing the button will select all the devices included in the selected area. Now you can move the block performing a operation analogue to the single component moving. First selecting with the mouse one of the components included in the area and keeping the left button on the mouse pressed move the block anywhere in the sheet. The NO-ACCESS symbol will automatically appear if you enter into wrong regions. Effective placements will be made after releasing the button on the mouse.

#### **Deleting Components**

Components can be deleted individually or in group. To delete a single component select with the mouse the component. The component must be selected by clicking with the left button on the mouse on the component. Once selected the components will appear. Now you can use the delete command ( pressing the CANCEL KEY ) or the cut command to remove the object from the schematic. Using the cut command gives you the undo feature.

Group of components can be deleted choosing the pointer in the tool bar and selecting a region in the schematic. All the components included in that region will appear. Now you can definitely remove the group of objects pressing the CANCEL KEY or cutting the block in the clipboard with the cut command. Using the cut allows you to restore the deleting with the undo feature.

### 10.3.14 Connetting components



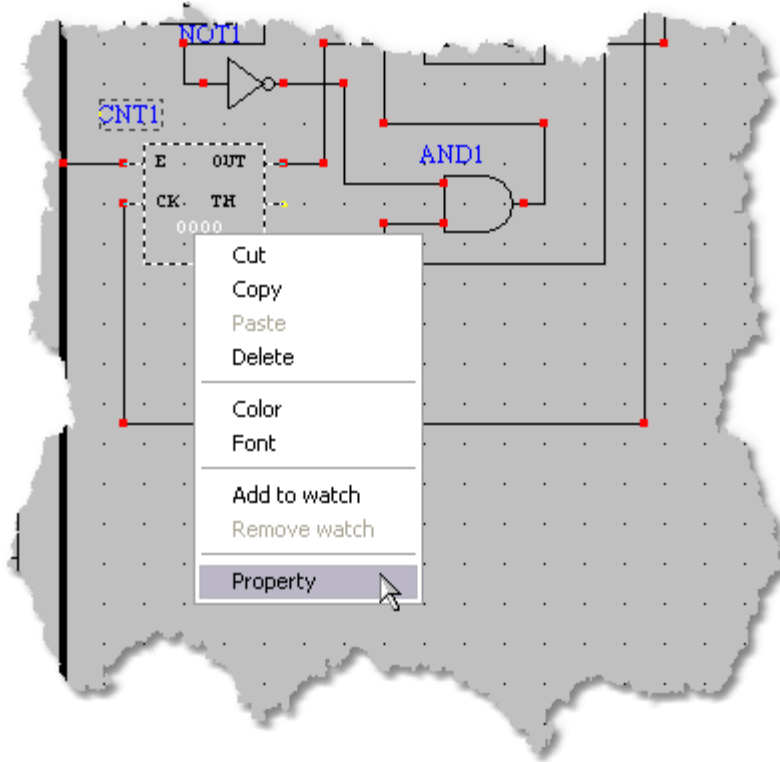
Once the components are placed in the schematic you have to connect the device's pins with wires. To draw a wire first select the wire tool from the tools bar . With the mouse click on a object pin and keep the left button on the mouse pressed, drag the wire to the destination position. When the destination pin is reached the wire will be effectively placed realing the mouse button. Wires extremes are snapped by a fixed grid and errors in placement are impossible. Remember that wires are always orthogonals so you have to split your wire in more parts if the destination pin is obstructed by other objects in the schematic. The effective good connection between components can be checked by looking at the wires extremity. If the wire terminal is marked with a coloured box then the connection is right. Wrong connections are indicated by a cross .

Wires can also be connected to the power bars. Connecting a wire to the left power bar will give a logical one to the net ( VCC ), connecting a wire to the right power bar gives a logical zero to the net (GND) . Sometimes it is more practical to use GND devices instead of the right power bar .

Wires can be moved using the same procedures used for components. Also the wires length can be modified performing the following procedure. Select with the mouse a wire extremity, a double arrow symbol will indicate the resizing, and keeping the left button on the mouse pressed move the extremity in the desired point. Wire length will be modified by releasing the mouse button. It can happen that a wire terminal is connected to other wires. To select a particular wire in a node click once the mouse button on the node until the desired wire is selected then procede with the resizing.

### 10.3.15 Setting components properties

After placing components, you have to set the components property. Generally, properties are variables local to the components that affects the behavior of the object. For example if you have placed a counter, the system should know the count range and the direction of step. All these parameters can be configured performing a double click on the component or selecting the Property entry on the list opened with the right button on the mouse.



#### *Component's property*

The property command will open the dialog associated with the component. The parameters included in the dialogs are relative to the placed object. See the [Library](#) section to find individual information about the parameters in the dialogs.

For example, if we request the property command for a COUNTER object we will be prompted for the following dialog. Once the parameters are configured you simply have to push the OK button to save the changes or press the CANCEL button to discharge the changing.

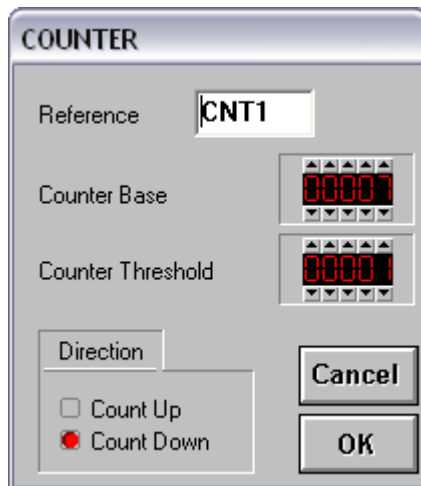


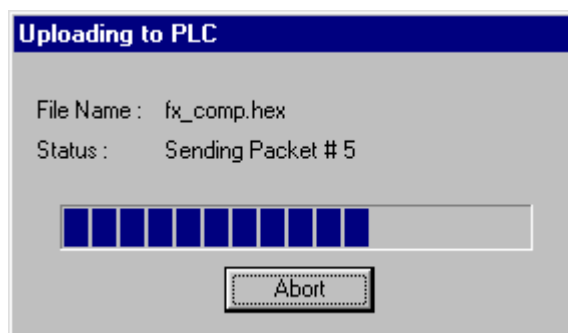
Figure 18 - Property dialog sample

### 10.3.16 Building the code

Code could be generated simply using the shortcut key F5 or accessing the **Build → Compile** menu command

### 10.3.17 Uploading the code

Some PLC models have a direct upload capability. Using this feature you can upload the generated code directly to PLC using a standard RS232C port of your PC. To upload the code onto the PLC simply press the F6 button on the keyboard or select Upload from the Build menu. Also uploading can be activated pressing the Upload button in the Compile bar. Sometimes, to automatize the Compile & Upload & Run processes it is advisable to use the F10 button. The three processes are executed sequentially simply pressing this button



Upload session

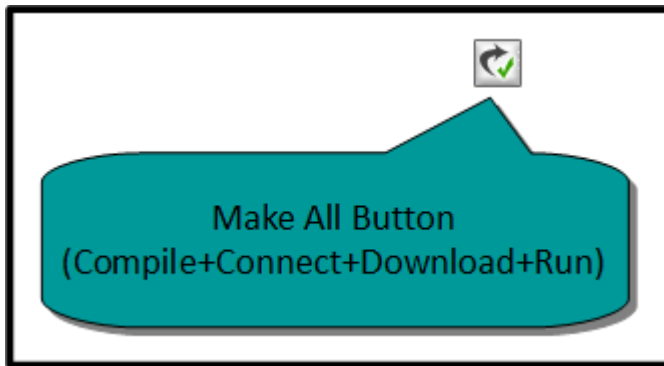
### 10.3.18 Running and stopping the PLC

PLC is normally started after the upload session automatically by the system if you use the F10 shortcut key. If your PLC supports the direct-upload feature than other operations like PLAY and STOP could be performed. Use the PLAY button to run the PLC and use the STOP button to stop the PLC.

### 10.3.19 Compiling and debugging your project

The project can be compiled, downloaded and run into the PLC simply using the **MAKE ALL** button.

The same function is accessible using the menu **Build → Compile & Download & Run**



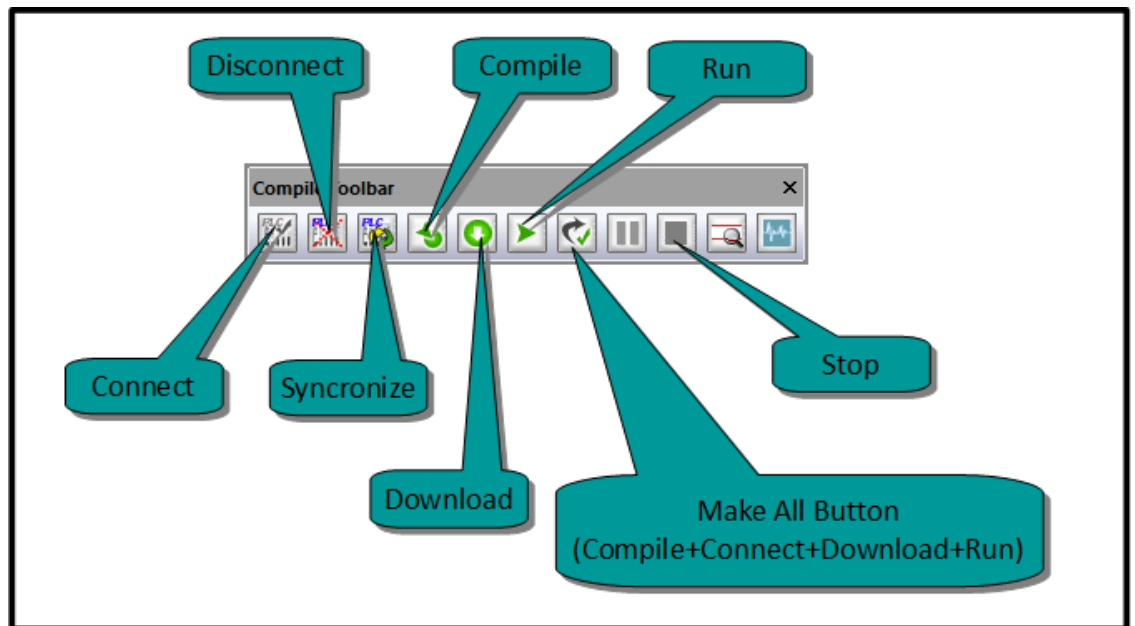
Once you pressed the button the software will build the code for your project, will connect the PLC using the configured port, download the code into the PLC and will run it

During the build phase a certain number of messages will be shown in the **Build** tab of the message window. In case of compiling errors the software will stop the automated **MAKE ALL** sequence

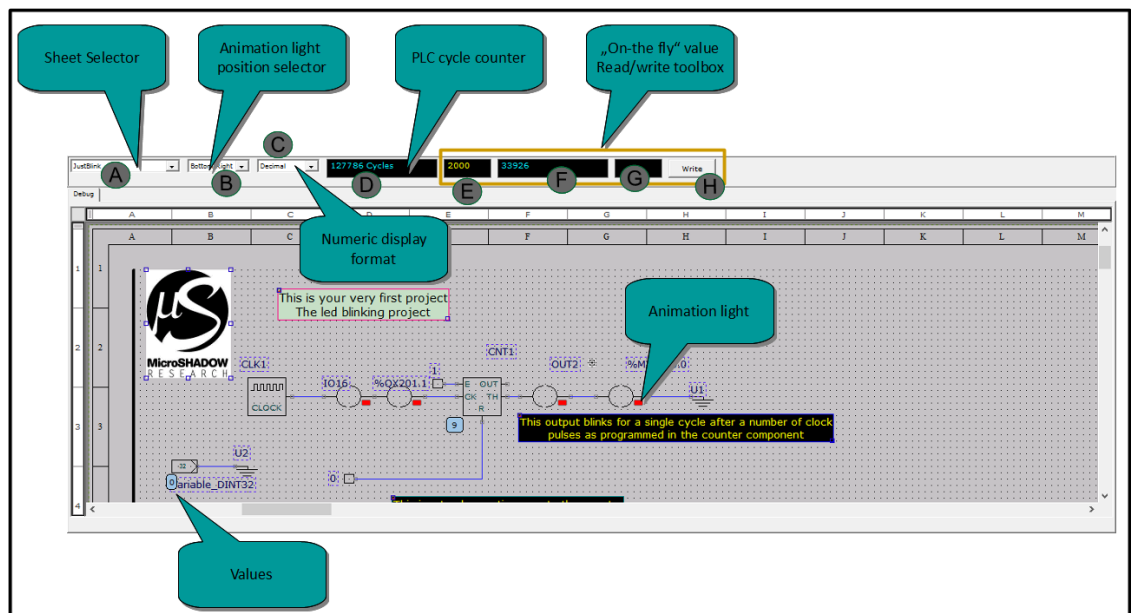
If everything in the diagrams are ok the software, after the build phase, will execute, in order the following steps

- Connect the PLC
- Downloading
- Run

The processes that are automatically run using the **MAKE ALL** function can be executed singularly using the Compile Toolbar



After that all the compiling and downloading processes are executed on the Debug Window will be opened. The window looks like in the picture below



Let's describe any section shown in the picture above

#### (A) Sheet Selector

Using this combo box will allow you to select which sheet (diagram) you want to debug

#### (B) Animation light position selector

This combo box allow to select the position used for I/O components monitoring. The state of I/O components is represented by a light that assumes two different colors: red for the zero and green for the one

**(C) Numeric display format**

This combo box selects the numeric format used to display values

**(D) PLC Cycle Counter**

This read-only field show the current PLC cycle. It is very useful to see if the system is working properly

**"On-the-fly" value "words" read/write toolbox**

In this section we have four items. This "Toolbox" is used to read and write on-the-fly values while the PLC is normally running

**(E) Word address**

The address of the word that you want to read/write is entered here (values from 0 to x representing the standard PLC words %MW0 .. %MWx)

**(F) Current word value**

This read-only field contain the value of the word address by the field (E) is shown here in real-time

**(G) Value to write**

The value to be writed into the word addressed by the field (E) is entered here

**(H) Write command**

The writing of a word, with address specified by field (E) and value entered in field (G), will take place pressing this button

### 10.3.19.1 Changing On-The-Fly Values

For some components like INPUT and READVAR there is the possibility to change the associated word (or bit) value **on-the-fly**

This function is enabled when the PLC is in running state

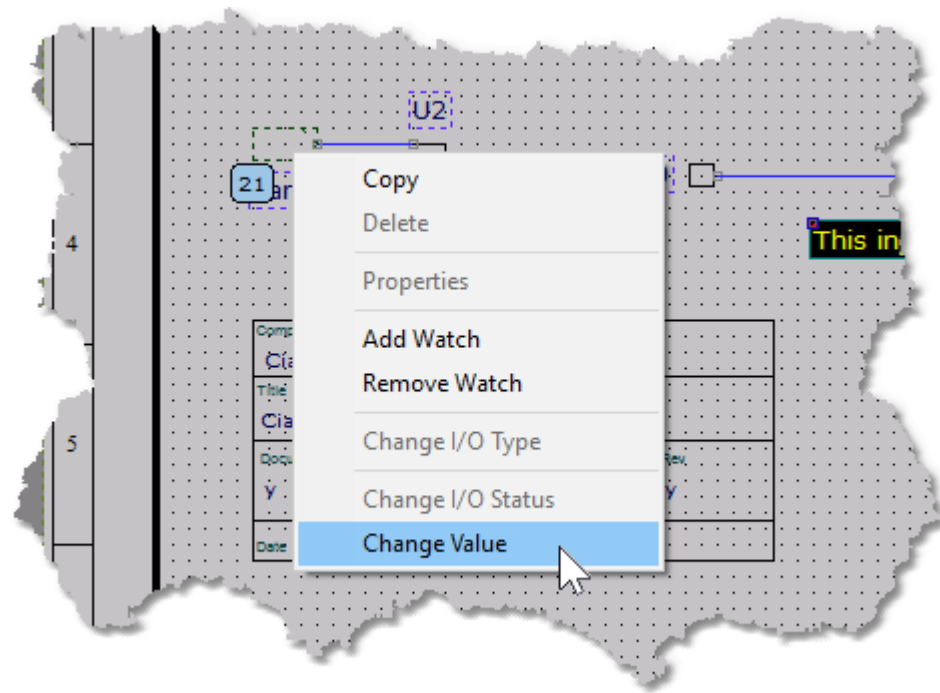
The value you are going to alter using this function is related to the word referenced by the REFERENCE field and configured using the component properties dialog



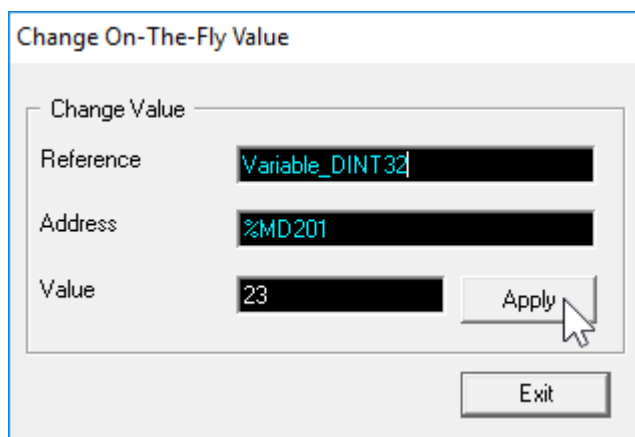
**IMPORTANT:** When you are going to alter a value you have to be sure that the variable is not written (or referred) by some other component in your project. If you are altering a variable that is driven (written) by some other block, your operation will not take place.

So, for example, if you are changing, on-the-fly, the value related to a READVAR component you alter the WORD %MWxyz entered in its REFERENCE field

The function is accessible selecting the component and opening the right mouse button menu



Activating this function will open the follow dialog

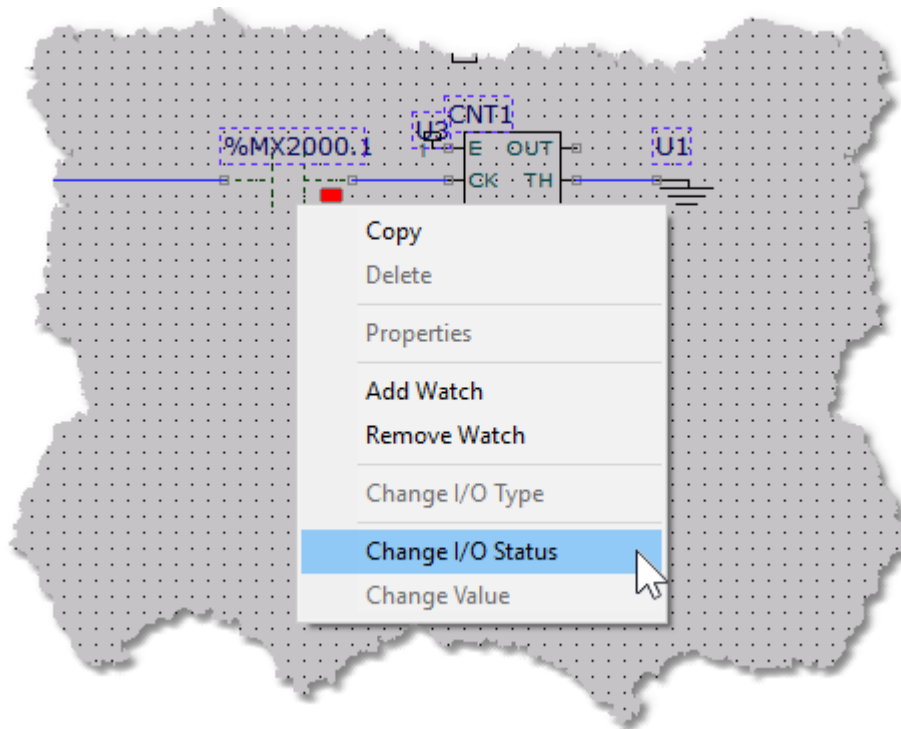


The software automatically recognize the Tagname and the associated address for the variable. At this point you can enter a value and press the **Apply** button.

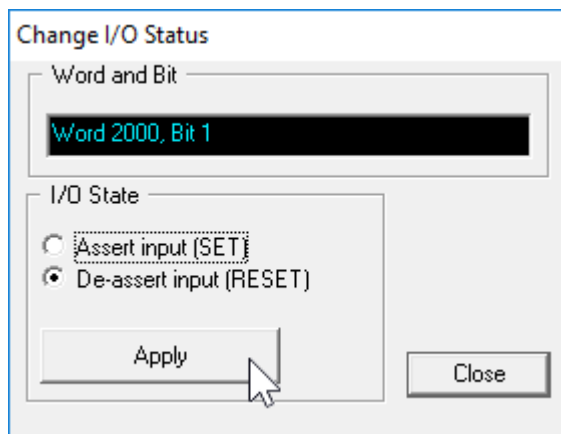
The variable writing will take place when pressing

The same functionality is applicable to INPUT blocks. When you are in running state you can alter the bit variable associated with the component simply using with the mouse right button, opening the context menu and selecting "Change I/O Status".





At this point, selecting this option, this dialog will appear



The bit state, related to that component, will be changed after selecting the SET/RESET option and pressing Apply

### 10.3.20 Adding watches

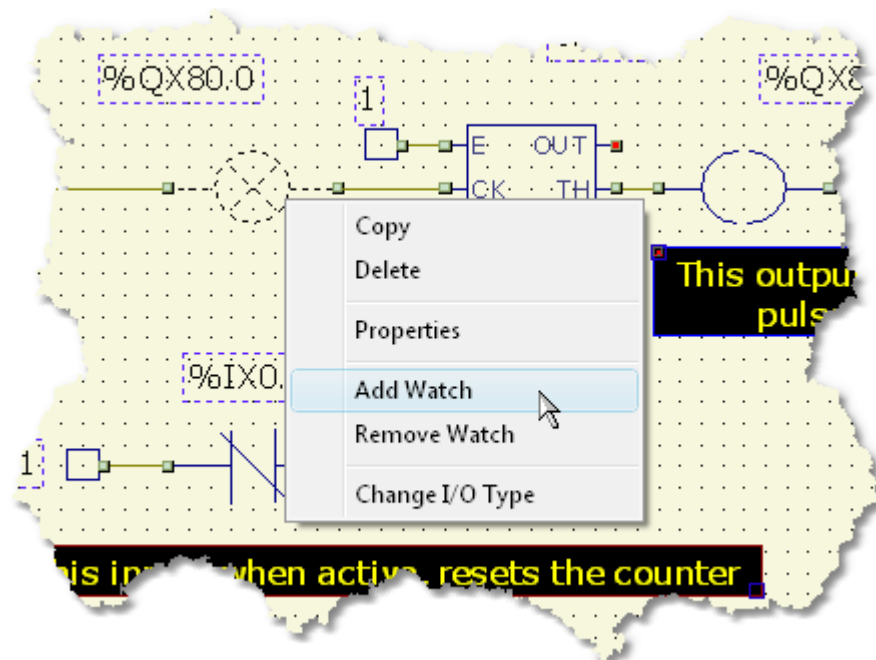
The Watches are variable that are shown into the dedicated IDE window. This is useful to analyze run-time values during PLC running. Users can debug projects checking variable and I/O values when PLC process programs. Watches can be added or removed on-the-fly

[illegible]

Watches can be added simply following the listed operations

- Select a component on the diagram with the mouse's left button
- Access the context menu clicking over it with the mouse's right button
- Click on the **Add Watch** command

Here is a sample image of the adding sequence



Once a variable is in the watch window its value will be automatically refreshed when the PLC is running.

### 10.3.21 Using help

The on-line help is activated using the standard shortcut F1 or selecting Help from the Help menu .

## 10.4 Library

### Library Index by function

#### Input/Output devices

[INPUT](#)  
[NCINPUT](#)  
[EINPUT](#)  
[ENCINPUT](#)  
[OUTPUT](#)  
[EOUTPUT](#)

#### Analog Input/Output

[AD\\_CONV](#)  
[PWMOUT](#)

#### Timers

[DELAY](#)  
[TP](#)  
[TON](#)  
[TOF](#)  
[TMI](#)  
[TSQ](#)  
[CLOCK](#)

#### Counters

[COUNTER](#)  
[CTU](#)  
[CTD](#)  
[CTUD](#)

#### Mathematical

[ADD](#)  
[DIV](#)  
[LIMIT](#)  
[MAX](#)  
[MIN](#)  
[MOD](#)  
[MUL](#)  
[SUB](#)

#### Shift/Rotate

[SHL](#)  
[SHR](#)  
[ROL](#)  
[ROR](#)

#### Bit-Wise

[BIT](#)  
[BITR](#)  
[BITS](#)  
[BITSR](#)

#### Byte/Word Manipulation

[HBYTE](#)  
[LBYTE](#)  
[MKWORD](#)

#### Logical ports

#### Selectors & Mux

[AND](#)  
[OR](#)  
[NOT](#)  
[XOR](#)

#### **Queues**

#### **Edge detectors**

[F\\_TRIG](#)  
[R\\_TRIG](#)

#### **Relay / Coils**

[RELAY](#)

#### **Constant and identifiers**

[CONST](#)  
[IDENT](#)  
[IDENT32](#)

#### **Filters**

[DEBOUNCE](#)

[SEL](#)  
[DEC1-8](#)  
[BIT](#)  
[BIT32](#)  
[MUX](#)

#### **Flip Flops**

[FFD](#)  
[RS](#)  
[SR](#)  
[SEMA](#)

#### **Assign / Read**

[ASSIGN](#)  
[READVAR](#)  
[ASSIGN32](#)  
[RDVAR32](#)  
[MEMWR16](#)  
[MEMWR32](#)  
[WRE16](#)  
[WRE32](#)

#### **Comparators**

[THRESHLD](#)  
[CMP\\_W](#)

#### **User functions**

### 10.4.1 Library conventions

We used the following conventions for the library section

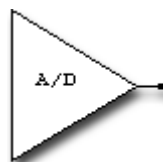
#### **Property settings**

The Property setting are the parameters that you can change accessing the properties (Properties Dialog) of a particular component in the library. Not all the components have properties

#### **Net plugs**

The net plugs are the pin of a particular component where you can connect other elements in the schematic

## 10.4.2 AD\_CONV



Some PLC models has one or more analog to digital converters. This device allows you to acquire analogs value and convert it in a numerical value. The value supplied by the converter is normalized in the range 0-65535 independently by the converter resolution. Two parameters, called OFFSET and SPAN, are available to change the converter dynamic. The OFFSET parameter adds a base value to the converted value and the SPAN parameter changes the converter gain.

With values OFFSET=0 and SPAN=1.0 no alteration will be applied to the converted value. SPAN values greater than 1.0 produce a gain, SPAN values less than 1.0 produce value attenuation.

### Property settings

Parameter	Description
OFFSET	This parameter adds an OFFSET value to the converted value
SPAN	This parameter changes the gain of the converter

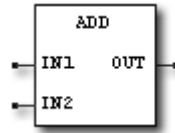
### Net plugs

Plug	Description
OUT	Data output (normalized 0-65535)

See also : [PWMOUT](#)

### 10.4.3 ADD

IEC / CEI 1131-3 Compliant



This device performs the sum of the two values present at inputs IN1 and IN2 giving the result on the OUT pin

#### Net plugs

Plug	Description
IN1	First operand
IN2	Second operand
OUT	Result of the sum

See also : [Mathematical Expressions](#), [SUB](#), [MUL](#), [DIV](#), [MOD](#)

#### 10.4.4 AND



The AND device performs a logical AND between two boolean signals. The output of this device is true when both the inputs are true.

**See also :** [NOT](#), [OR](#), [XOR](#)

### 10.4.5 ASSIGN



Assign creates a variable in memory and unconditionally assigns to it the value present at component input pin.

The associated variable name is specified in the REFERENCE parameter. LadderWORK ARM software handles integer unsigned 16/32 bits variables. As written in IEC / CEI 1131-3 specification, we suggest, for this kind of variable, the use of the standard %MW prefix which means a memory word variable. When a variable is created using ASSIGN it is public to the entire net.

It is possible to read the value assigned by an ASSIGN object using the READVAR device. ASSIGN transfer without changes the value present on its input pin to the output pin.

#### Property settings

Parameter	Description
REFERENCE	This parameter specify the name of the variable

#### Net plugs

Plug	Description
IN	Value that will be assigned to the variable
OUT	This pin gives the same value of the input pin

See also : [READVAR](#)



## 10.4.6 ASSIGN32



Assign creates a 32 bits variable in memory and unconditionally assigns to it the value present at component input pin.

The associated variable name is specified in the REFERENCE parameter. LadderWORK ARM software handles integer unsigned 16/32 bits variables. As written in IEC / CEI 1131-3 specification, we suggest, for this kind of variable, the use of the standard %MD prefix which means a memory double word variable. When a variable is created using ASSIGN32 it is public to the entire net.

It is possible to read the value assigned by an ASSIGN32 object using the RDVAR32 device. ASSIGN32 transfer without changes the value present on its input pin to the output pin.

### Property settings

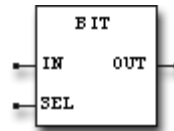
Parameter	Description
REFERENCE	This parameter specify the name of the variable

### Net plugs

Plug	Description
IN	Value that will be assigned to the variable
OUT	This pin gives the same value of the input pin

See also : [RDVAR32](#)

### 10.4.7 BIT



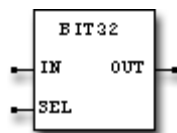
This function block allow to extract a single bit from a word. The bit number is specified by the value applied to the input SEL and the word must be placed at input IN. The resulting boolean value will be available on the output named OUT.

#### Net plugs

Plug	Description
IN	The word where the bit must be extracted
SEL	The bit selector
OUT	The resulting extracted bit

See also : [BIT32](#)

### 10.4.8 BIT32



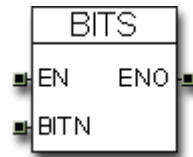
This function block allow to extract a single bit from a double word ( 32 bits ) . The bit number ( 0..31 ) is specified by the value applied to the input SEL and the double word, where i want to extract the single bit, must be placed at input IN. The resulting boolean value will be available on the output named OUT.

#### Net plugs

Plug	Description
IN	The word where the bit must be extracted
SEL	The bit selector
OUT	The resulting extracted bit

See also : [BIT](#)

### 10.4.9 BITS



This function block allow you to SET a bit into a 16-bits word. The bit specified by the input BITN (0..15), related to the word assigned to the REFERENCE field, is SE when the EN (Enable) pin is active. The REFERENCE field in the property dialog must contain a valid word address specifier. The ENO output reflects the value applied on the EN pin

#### Property settings

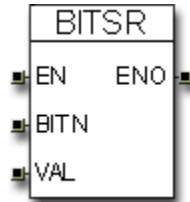
Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identificator %Mx or specify a tag-name

#### Net plugs

Plug	Description
EN	Enable input
ENO	Enable output. The ENO output reflects the value applied on the EN pin
BITN	Bit to set within the word (0..15)

See also : [BITR](#), [BITSR](#)

### 10.4.10 BITSR



This function block allow you to specify a value (Set/Reset) for a particular bit into a 16-bits word. The bit specified by the input BITN (0..15), related to the word assigned to the REFERENCE field, is set to the value applied at the VAL pin when the EN (Enable) pin is active. The REFERENCE field in the property dialog must contain a valid word address specifier. The ENO output reflects the value applied on the EN pin.

#### Property settings

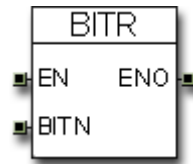
Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %Mx or specify a tag-name

#### Net plugs

Plug	Description
EN	Enable input
ENO	Enable output. The ENO output reflects the value applied on the EN pin
BITN	Bit to alter (0..15)
VAL	Value for the bit, 0 or 1

See also : [BITS](#), [BITR](#)

### 10.4.11 BITR



This function block allow you to CLEAR (Reset) a bit into a 16-bits word. The bit specified by the input BITN (0..15), related to the word assigned to the REFERENCE field, is CLEARED when the EN (Enable) pin is active. The REFERENCE field in the property dialog must contain a valid word address specifier. The ENO output reflects the value applied on the EN pin

#### Property settings

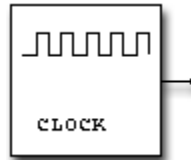
Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %Mx or specify a tag-name

#### Net plugs

Plug	Description
EN	Enable input
ENO	Enable output. The ENO output reflects the value applied on the EN pin
BITN	Bit to clear within the word (0..15)

See also : [BITS](#), [BITSR](#)

## 10.4.12 CLOCK



The CLOCK device generates fixed frequency pulses. The frequency of the pulses can be programmed through the FREQUENCY parameter.

### Property settings

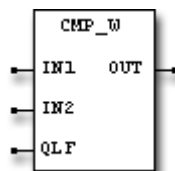
Parameter	Description
FREQUENCY	This parameter defines the frequency of the pulses. The minimum and maximum frequency depends by the used PLC. For further information about the timing resolution of the devices CLOCK and DELAY see TIMING RESOLUTION

### Net plugs

Plug	Description
OUT	Generated pulses are available on this output

See also : [PWMOUT](#)

### 10.4.13 CMP\_W



The CMP\_W component compares the magnitude of the values applied to the input pins IN1 and IN2.

The result of comparing is function of the value applied to the qualifier input QLF.

Using this function block you have the possibility to change, during run-time, the logic of comparing simply changing the value applied to the QLF pin.

The function block returns TRUE, on its OUT output pin, if the condition is verified else it returns FALSE

QLF Pin	Equation	Related function
0	$IN1 == IN2$	TRUE when IN1 is equal to IN2
1	$IN <> IN2$	TRUE when IN1 is different by IN2
2	$IN1 >= IN2$	TRUE when IN1 is greater or equal to IN2
3	$IN1 <= IN2$	TRUE when IN1 is less or equal to IN2
4	$IN1 > IN2$	TRUE when IN1 is greater than IN2
5	$IN1 < IN2$	TRUE when IN1 is less than IN2

If the value applied to the QLF pin is greater than 5 then the function block returns always FALSE.

For supply a constant value to the QLF pin you can use the [CONST](#) or the [IDENT](#) function blocks.

#### Net plugs

Plug	Description
IN1	The first value to be compared should be applied to this pin
IN2	The second value to be compared should be applied to this pin



QLF            A value, from 0 to 5, applied to this pin selects the appropriate logic equations for the comparing

OUT           The result of the comparing will be available on this pin

See also : [THRESHLD](#)

#### 10.4.14 CONST



This device gives you the possibility to enter constant word values as input for many library devices available in the software. For example you can use CONST to define the count range for a CTU device applying this component to the PV ( programmed value ) input. The constant value is entered double-clicking on the object and configuring the VALUE parameter on the property dialog.

##### Property settings

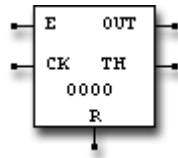
Parameter	Description
VALUE	The constant must be entered using this parameter and must respect the rules explained in the section <a href="#">Literals</a>

##### Net plugs

Plug	Description
OUT	The constant programmed word value will be available on this output

See also : [Literals](#), [IDENT](#)

## 10.4.15 COUNTER



The COUNTER device counts pulses applied to its CLOCK input. The device can be programmed to be UP counter or DOWN counter. The counter's counting is incremented or decremented on the raising edge of the CLOCK pulse when the enable pin E is asserted. At startup the counter is initialized to the value programmed on the BASE parameter. The counter proceeds with its counting until the THRESHOLD value is reached. When this is reached the threshold signal TH will become true. The next cycle will load the counter with the BASE value again.

The counter can be initialized by asserting a true signal into the R pin. When the R signal is asserted, the counter is loaded with the BASE value.

The maximum counting value for this device is 65535.

### Property settings

Parameter	Description
BASE	The base value for the counting
THRESHOLD	The threshold value for the counting
UPDOWN	This parameter changes the direction of the counting

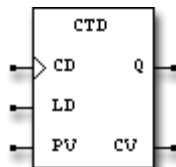
### Net plugs

Plug	Description
E	Counting enable pin
CK	CLOCK input
R	RESET input
TH	THRESHOLD output
OUT	The counter value output

See also : [CTU](#), [CTD](#), [CTUD](#)

## 10.4.16 CTD

### IEC / CEI 1131-3 Compliant



The CTD object represents a DOWN COUNTER. A rising-edge on CD input will decrement the counting by one. The Q output become TRUE when the current counting value is equal or less than zero. Applying a TRUE signal on LD (LOAD) input will load the counter with the value present at input PV ( Asynchronous load ). The CV output pin reports the current counting value.

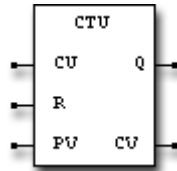
#### Net plugs

Plug	Description
CD	A rising-edge on this input will decrement the counter by one
LD	Applying a TRUE signal on this input will load the counter with the value present at input PV
PV	When the LD pin is asserted, the value applied to this pin will be loaded as current count value. User should use a <a href="#">CONST</a> or <a href="#">IDENT</a> object to enter numerical constant
Q	This output become TRUE when the counting is less or equal than zero
CV	This output reports the current counting value

**See also :** [COUNTER](#) , [CTU](#) , [CTUD](#) , [IDENT](#) , [CONST](#)

## 10.4.17 CTU

### IEC / CEI 1131-3 Compliant



The CTU object represents an UP COUNTER. A rising-edge on CU input will increment the counting by one. When the programmed value, applied to the input PV, is reached, the Q output become TRUE.

Applying a TRUE signal on R input will reset the counter to zero ( Asynchronous reset ). The CV output pin reports the current counting value.

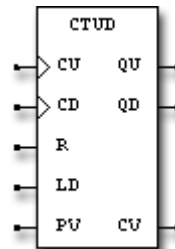
#### Net plugs

Plug	Description
CU	A rising-edge on this input will increment the counter by one
R	Applying a TRUE signal on this input will reset the counter
PV	The Q output become TRUE when the current counting value reaches the value applied to this input. User should use a <a href="#">CONST</a> or <a href="#">IDENT</a> object to enter numerical constant
Q	This output become TRUE when the counting is equal or greater than the programmed value
CV	This output reports the current counting value

**See also :** [COUNTER](#), [CTD](#), [CTUD](#), [IDENT](#), [CONST](#)

## 10.4.18 CTUD

### IEC / CEI 1131-3 Compliant



This device represents an UP/DOWN programmable counter. A rising-edge on the CU ( COUNT-UP) input will increment the counter by one while a rising-edge on the CD ( COUNT-DOWN ) decreases the current value.

Applying a TRUE signal on R input will reset the counter to zero. A TRUE condition on the LD signal will load the counter with the value applied to the input PV ( PROGRAMMED VALUE ) . QU output becomes active when the current counting value is greater or equal to the programmed value. The QD output becomes active when the current value is less or equal to zero.

The CV output reports the current counter value.

As specified in the IEC / CEI 1131-3 standard this kind of counter has a counting range expressed by an integer 16 bit variable. This means that this counter can span from -32768 to +32767 .

#### Net plugs

Plug	Description
CD	A rising-edge on this input will decrement the counter by one
CU	A rising-edge on this input will increment the counter by one
R	Applying a TRUE signal on this input will reset the counter
LD	Applying a TRUE signal on this input will load the counter with the value present at input PV
PV	When the LD pin is asserted, the value applied to this pin will be loaded as current count value. User should use a <a href="#">CONST</a> or <a href="#">IDENT</a> object to enter numerical constant

---

QU	This output become TRUE when the counting is equal or greater than the programmed value
QD	This output become TRUE when the counting is less or equal than zero
CV	This output reports the current counting value

**See also :** [COUNTER](#), [CTD](#), [CTU](#), [IDENT](#), [CONST](#)

### 10.4.19 DEBOUNCE



The DEBOUNCE device can be used in conjunction with the standard input device like INPUT, EINPUT, NCINPUT, ENCINPUT. The main function of this device is to eliminate the typical spikes/noises generated by a hardware switch or button.

The DEBOUNCE device computes a unitary integration in the programmed time. When the integration time is elapsed the output will become true if the computed value is greater than a pre/programmed threshold.

Typically, the main function of this device is to eliminate spikes and noises on PLC digital physical inputs.

#### Property settings

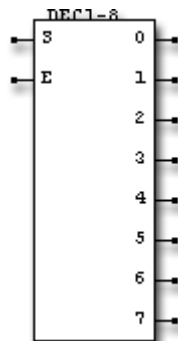
Parameter	Description
INTEGRATION TIME	This parameter ( expressed in milliseconds ) changes the filter integration time. For signals generated by switches a filtering time of 100 ms is sufficient

#### Net plugs

Plug	Description
INPUT	Filter input
OUT	Filter output



### 10.4.20 DEC1-8

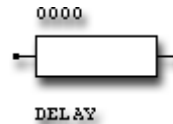


This function block represents a one-to-eight decoder. The boolean value applied to the input E is transferred to the output selected by the value applied to the input S.

**Net plugs**

Plug	Description
S	The output selector
E	The boolean value that will be transferred to the selected output
0..7	One of these outputs ( the selected ) will reply the boolean value applied to the E pin

### 10.4.21 DELAY



The DELAY device generates delayed signals respect to the input signal. Two kinds of functionality are available : DELAY MODE or HOLD MODE. In DELAY MODE a pulse applied on its input generates a single pulse after the programmed time. In HOLD MODE a pulse on its input activates the output for all the programmed time.

There is the possibility to condition the behavior of the device for the pulses following the first trigger pulse. Using the NO RETTRIGGERABLE option the pulse following the first trigger pulse will be ignored. With the RETTRIGGERABLE option set, the elapsed time will be cleared every time a rising-edge is recognized on the input so the time-past event will take place starting from the last pulse.

DELAY device, like CLOCK and DEBOUNCE , are SYSTEM TIMER dependant.  
For further information see TIMING RESOLUTION .

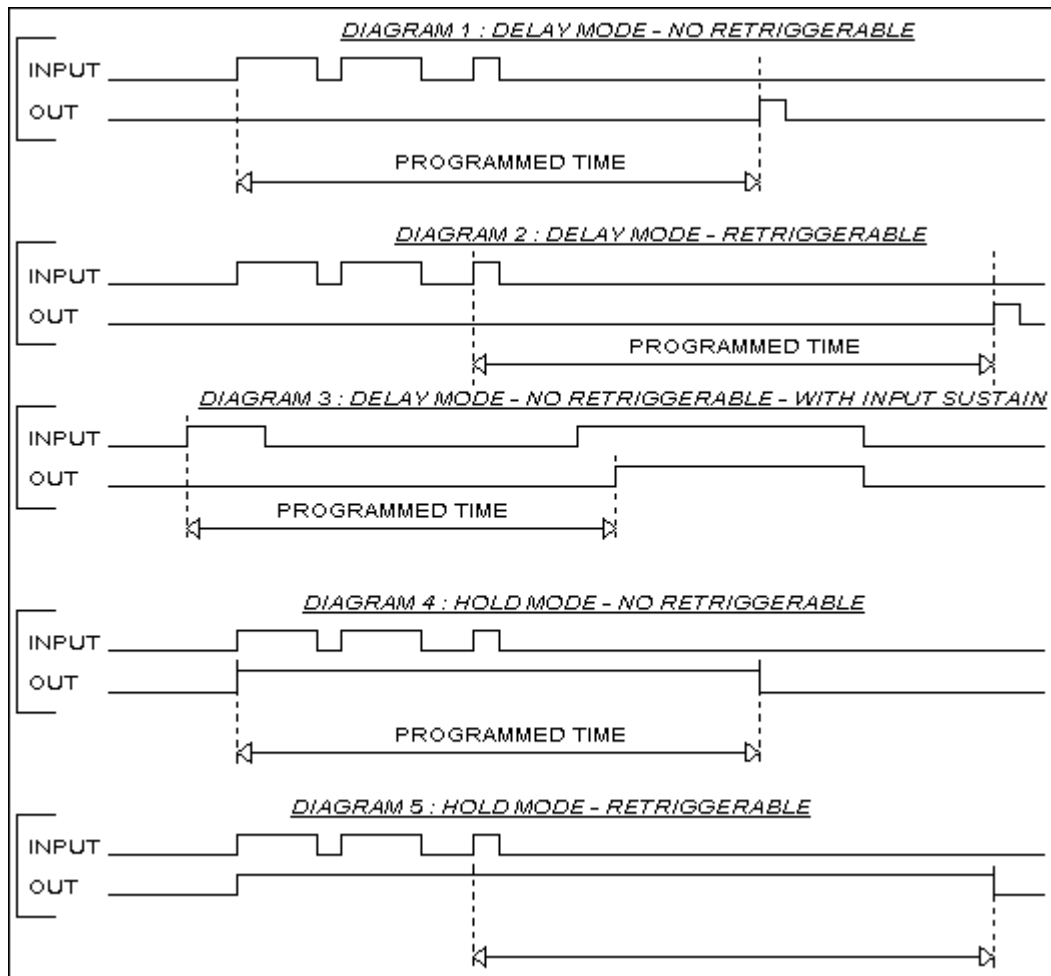
#### Property settings

Parameter	Description
DELAY TIME	Delay time expressed in seconds
MODE	Selects HOLD MODE or DELAY MODE
TRIGGER MODE	The behavior of the DELAY device to pulses that follows the first pulse can be changed through this parameter ( See description )

#### Net plugs

Plug	Description
INPUT	Delay input. This pin is raising-edge sensitive
OUT	The delay output

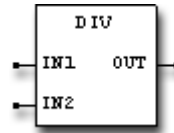
#### Timing Diagrams



See also : [TP](#), [TON](#), [TOF](#), [TMI](#), [TSQ](#)

## 10.4.22 DIV

### IEC / CEI 1131-3 Compliant



This function block divide (as unsigned 16 bits) the value applied to the input IN1 (DIVIDEND) by the value applied to the input IN2 (DIVISOR) and the result of the division is available on the OUT pin

#### Net plugs

Plug	Description
IN1	Dividend
IN2	Divisor
OUT	Result of division

**See also :** [Mathematical Expressions](#), [ADD](#), [MUL](#), [MOD](#), [SUB](#)

### 10.4.23 EINPUT



EINPUT represents the normally-open input device in electrical symbology. This component may represent a physical input of PLC or a logical input to associate with a [RELAY](#) device. To establish if an input is physical or logical opportune values must be programmed on the REFERENCE parameter of the property dialog. If the [REFERENCE](#) value refers to a physical resource

of the PLC then a physical input will be created. If the REFERENCE field does not refer to any physical resources then the EINPUT will be configured as logical input to associate with a RELAY device. This configuration is also called [LOGICAL LINKS](#).

There is no limit on the EINPUT devices that can be related with a RELAY device.

In general, an input devices perform a logical AND between the signal supplied as input and the switch signal. For switch signal we mean the signal that closes the switch ( logical or physical ). The output of this device will be TRUE if both the input and the switch signal are TRUE. The output will show FALSE everytime the switch is OPEN or the input is FALSE.

**See also :** [INPUT](#), [NCINPUT](#), [ENCINPUT](#), [OUTPUT](#), [EOUTPUT](#), [RELAY](#)

#### 10.4.24 ENCINPUT



ENCINPUT represents the normally-closed input device in electrical symbology. This component may represent a physical input of PLC or a logical input to associate with a [RELAY](#) device. To establish if an input is physical or logical opportune values must be programmed on the [REFERENCE](#) parameter of the property dialog. If the REFERENCE value refers to a physical resource of the PLC then a physical input will be created. If the REFERENCE field does not refer to any physical resources then the EINPUT will be configured as logical input to associate with a RELAY device. This configuration is also called [LOGICAL LINKS](#).

There's no limit on the EINPUT devices that can be related with a RELAY device.

**See also :** [INPUT](#), [NCINPUT](#), [EINPUT](#), [OUTPUT](#), [EOUTPUT](#), [RELAY](#)

## 10.4.25 EOUTPUT



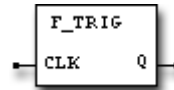
This object represents a generic output device in electrical symbology. This device can be associated to a physical output of the PLC or linked to other input in the schematic. The physical or logical property is opportunely configurable selecting a physical resource or not on the associated dialog. Through the [REFERENCE](#) field on the configuration dialog allows the user to select one of the available physical resources. If one of these resources is selected then the device will be configured as physical output. Any other selection will produce a logical device to associate with components like [INPUT](#) , [NCINPUT](#) , [EINPUT](#) , [ENCINPUT](#) .

For further information about input-output device association see the section [LOGICAL LINKS](#) . In detail, when an output device is configured as logical object, a global variable is created and is public to all the net. The EOUTPUT device reply the signal present on its input on the output, so more EOUTPUT devices can be chained on the same rung.

**See also :** [INPUT](#) , [NCINPUT](#) , [EINPUT](#) , [ENCINPUT](#) , [OUTPUT](#) , [RELAY](#)

### 10.4.26 F\_TRIG

#### IEC / CEI 1131-3 Compliant



This device is a falling-edge detector. The Q output become TRUE when a 1 to 0 ( or TRUE to FALSE or ON to OFF ) condition is detected on the CLK input and it sustain this state for a complete scan cycle.

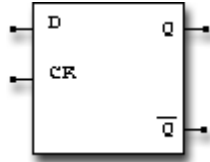
#### Net plugs

Plug	Description
CLK	The falling-edge detector input
Q	When a falling-edge is detected this output become true for a single scan cycle

See also : [R\\_TRIG](#)



### 10.4.27 FFD



This component represents a D-TYPE FLIP-FLOP. The D-TYPE FLIP-FLOP represent the elementary memory cell on the logic circuits.

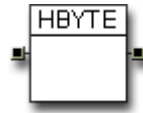
The behavior of the device is the following :

The data present on the D input is frozen on the raising edge of the CK signal. The Q output reports the value of the last freeze cycle and the /Q signal reports the complement of the Q signal.

#### Net plugs

Plug	Description
D	Boolean data input
CK	CLOCK signal
Q	Direct ( not negated ) output
/Q	Negated ouput ( complement of Q output )

### 10.4.28 HBYTE



This function block allow you to extract the HIGH BYTE (8-bits) from the 16 bits word applied to the left input pin

#### Net plugs

Plug	Type	Description
IN	WORD	16 bits input word
OUT	BYTE	8 bits output giving the high byte of the input word

See also : [LBYTE](#), [MKWORD](#)

## 10.4.29 IDENT

### IEC / CEI 1131-3 Compliant



This device gives you the possibility to enter constant values as specified in the IEC / CEI 1131-3 standard. The constant value is entered double-clicking on the object, entering a literal string on the REFERENCE parameter. For detailed informations about literal string see [LITERALS](#).

#### Property settings

Parameter	Description
REFERENCE	The literal or the identifier is entered using this field

#### Net plugs

Plug	Description
OUT	The value of the literal/identifier is available from this pin

See also : [CONST](#) , [LITERALS](#)

### 10.4.30 IDENT32

#### IEC / CEI 1131-3 Compliant



This device gives you the possibility to enter constant values as specified in the IEC / CEI 1131-3 standard giving a 32 bits output. The constant value is entered double-clicking on the object, entering a literal string on the REFERENCE parameter. For detailed informations about literal string see [LITERALS](#) .

#### Property settings

Parameter	Description
REFERENCE	The literal or the identifier is entered using this field

#### Net plugs

Plug	Description
OUT	The value of the 32 bits literal/identifier is available from this pin

See also : [CONST](#) , [LITERALS](#)

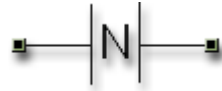
### 10.4.31 INPUT



INPUT represents the standard normally-open input device in the LADDER symbology. This component may represent a physical input of PLC or a logical input to associate with a [RELAY](#) device. To establish if an INPUT is physical or logical opportune value must be programmed on the [REFERENCE](#) parameter of the property dialog. If the REFERENCE value refers to a physical resource of the PLC then a physical INPUT will be created. If the REFERENCE field does not refer to any physical resources then the INPUT will be configured as logical input to associate with a RELAY device. This configuration is also called [LOGICAL LINKS](#) . There's no limit on the INPUT devices that can be related with a RELAY device.

**See also :** [EINPUT](#), [NCINPUT](#), [ENCINPUT](#), [OUTPUT](#), [EOUTPUT](#), [RELAY](#)

### 10.4.32 INPUT\_N



INPUT\_N represents the standard NEGATIVE-EDGE input device in the LADDER symbology. If the previous associated bit state was one and the input pin is zero then sets the bit state to one

#### Property settings

Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %MXa.b, %QXa.b, %IXa.b or specify a tag-name

**See also :** [EINPUT](#), [NCINPUT](#), [ENCINPUT](#), [OUTPUT](#), [EOUTPUT](#), [RELAY](#)

### 10.4.33 INPUT\_P



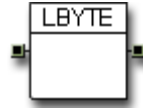
INPUT\_P represents the standard POSITIVE-EDGE input device in the LADDER symbology. If the previous associated bit state was zero and the input pin is one then sets the bit state to one

#### Property settings

Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %MXa.b, %QXa.b, %IXa.b or specify a tag-name

**See also :** [EINPUT](#), [NCINPUT](#), [ENCINPUT](#), [OUTPUT](#), [EOUTPUT](#), [RELAY](#)

### 10.4.34 LBYTE



This function block allow you to extract the LOW BYTE (8-bits) from the 16 bits word applied to the left input pin

#### Net plugs

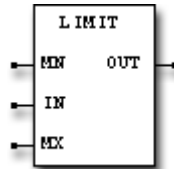
Plug	Type	Description
IN	WORD	16 bits input word
OUT	BYTE	8 bits output giving the low byte of the input word

See also : [HBYTE](#), [MKWORD](#)



### 10.4.35 LIMIT

#### IEC / CEI 1131-3 Compliant



This function block compares the value applied to the input IN with the values applied to the inputs MN ( MINIMUM VALUE ) and MX ( MAXIMUM VALUE ). If the input value is less than the value applied to the MN input then the output reports the value of MN pin. If the input value is greater than the value applied to the MX input then the output gives the value applied to the MX pin. When the value applied to the input is inside the two limits the value is transferred to the output without limitations.

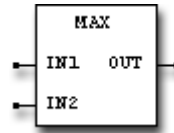
#### Net plugs

Plug	Description
MN	Minimum allowable value
MX	Maximum allowable value
IN	Value to be limited
OUT	Limited value

See also : [MAX](#), [MIN](#)

### 10.4.36 MAX

IEC / CEI 1131-3 Compliant



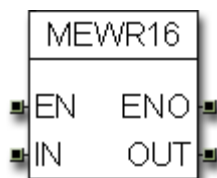
This function block compares the magnitude of the values present at input IN1 and IN2 reporting on its output the largest value ( maximum value ).

#### Net plugs

Plug	Description
IN1	First operand
IN2	Second operand
OUT	Maximum value

See also : [LIMIT](#), [MIN](#)

### 10.4.37 MEMWR16



This function block allow you to write the associated variable, defined by the REFERENCE field, only when the EN (Enable) pin is active. According to this, this block allow you to have a "gate-driven-writing". The OUT pin reflects the value of the associated variable.

#### Property settings

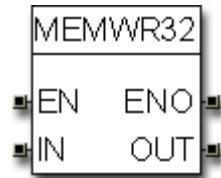
Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identificator %Mx or specify a tag-name

#### Net plugs

Plug	Description
EN	Enable input. The input value (16 bits), applied to the IN pin, will be writed in the associated variable when this signal is active
ENO	Enable output. The ENO output reflects the value applied on the EN pin
IN	The input value that will be written to the associated variable
OUT	The OUT pin reflects the value of the associated variable

See also : [MEMWR32](#)

### 10.4.38 MEMWR32



This function block allow you to write the associated variable, defined by the REFERENCE field, only when the EN (Enable) pin is active. According to this, this block allow you to have a "gate-driven-writing". The OUT pin reflects the value of the associated variable.

#### Property settings

Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identificator %Mx or specify a tag-name

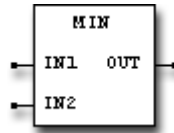
#### Net plugs

Plug	Description
EN	Enable input. The input value (32 bits), applied to the IN pin, will be writed in the associated variable when this signal is active
ENO	Enable output. The ENO output reflects the value applied on the EN pin
IN	The input value that will be written to the associated variable
OUT	The OUT pin reflects the value of the associated variable

See also : [MEMWR16](#)

### 10.4.39 MIN

#### IEC / CEI 1131-3 Compliant



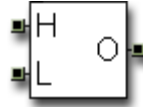
This function block compares the magnitude of the values present at input IN1 and IN2 reporting on its output the smallest value ( minimum value ).

#### Net plugs

Plug	Description
IN1	First operand
IN2	Second operand
OUT	Minimum value

See also : [LIMIT](#), [MAX](#)

#### 10.4.40 MKWORD



This block allow you to create a 16 bits word starting from two bytes applied to its pin (L and H). The L pin represent the LOW order byte and the H pin the HIGH order.

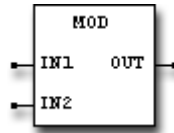
##### Net plugs

Plug	Description
H	The high order byte of the 16 bits words
L	The LOW order byte of the 16 bits words
O	This output pin gives the 16 bits word composed by the values applied to H and L

See also : [LBYTE](#), [HBYTE](#)

## 10.4.41 MOD

### IEC / CEI 1131-3 Compliant



This function block gives the remainder, result of division of the value applied to the input IN1 by the value applied to the input IN2. The computed value is available on the OUT pin.

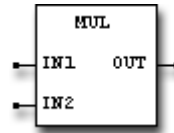
#### Net plugs

Plug	Description
IN1	The dividend
IN2	The divisor
OUT	Remainder ( MODULE ) of the division

**See also :** [Mathematical Expressions](#), [ADD](#), [SUB](#), [MUL](#), [DIV](#)

### 10.4.42 MUL

IEC / CEI 1131-3 Compliant



This device multiplies the values applied to the pins IN1 and IN2 giving the result on the OUT pin.

#### Net plugs

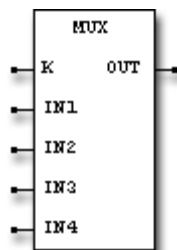
Plug	Description
IN1	First operand
IN2	Second operand
OUT	Result of multiplication

See also : [Mathematical Expressions](#), [ADD](#), [SUB](#), [DIV](#), [MOD](#)



### 10.4.43 MUX

#### IEC / CEI 1131-3 Compliant



This function block selects one of the four possible values applied to the inputs IN1..IN4 transferring the value to the OUT pin. The selection of the input is performed through the K input. A zero on K pin selects the IN1, a value equal to three selects the input IN4.

#### Net plugs

Plug	Description
K	The selector
IN1..IN4	Multiplexer inputs
OUT	This pin gives the value of the selected word

See also : [SEL](#)

#### 10.4.44 NCINPUT

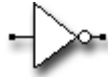


NCINPUT represents the normally-closed input device in LADDER symbology. This component may represent a physical input of PLC or a logical input to associate with a [RELAY](#) device. To establish if an input is physical or logical opportune values must be programmed on the [REFERENCE](#) parameter of the property dialog. If the REFERENCE value refers to a physical resource of the PLC then a physical input will be created. If the REFERENCE field does not refer to any physical resources then the EINPUT will be configured as logical input to associate with a RELAY device. This configuration is also called LOGICAL LINKS .

There is no limit on the NCINPUT devices that can be related with a RELAY device.

**See also :** [EINPUT](#), [INPUT](#), [ENCINPUT](#), [OUTPUT](#), [EOUTPUT](#), [RELAY](#)

#### 10.4.45 NOT



The NOT device performs a boolean data inversion of the signals input. If the input is true the output is false and viceversa.

**See also :** [AND](#), [OR](#), [XOR](#)

#### 10.4.46 OR



The OR device performs a logical OR between two boolean signals. The output become true when at least one input is true.

**See also :** [AND](#), [NOT](#), [XOR](#)

## 10.4.47 OUTPUT



This object represents a generic output device. This device can be associated to a physical output of the PLC or linked to other inputs in the schematic. The physical or logical property is opportunely configurable selecting a physical resource or not on the associated dialog.

Through the [REFERENCE](#) field on the configuration dialog allows the user to select one of the available physical resources. If one of these resources is selected then the device will be configured as physical output. All the other selections will produce a logical device to associate with components like `INPUT` , `NCINPUT` , `EINPUT` , `ENCINPUT` .

For further information about input-output device association see the section [LOGICAL LINKS](#) . In detail, when a `OUTPUT` device is configured as a logical object, a global variable is created and is public to all the net. The `OUTPUT` device will reply the signal present on its input on the output, so more `OUTPUT` devices can be chained on the same rung.

**See also :** [INPUT](#), [EINPUT](#), [NCINPUT](#), [ENCINPUT](#), [EOUTPUT](#), [RELAY](#)

#### 10.4.48 OUTPUT\_I



OUTPUT\_I represents the standard NEGATED-OUTPUT device in the LADDER symbology. The associated bit (boolean value), expressed by the REFERENCE, is reflects the negated value applied on the on the input pin. .

##### Property settings

Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %MXa.b, %QXa.b, %IXa.b or specify a tag-name

**See also :** [EINPUT](#), [NCINPUT](#), [ENCINPUT](#), [OUTPUT](#), [EOUTPUT](#), [RELAY](#)

#### 10.4.49 OUTPUT\_N



OUTPUT\_N represents the standard NEGATIVE-EDGE output device in the LADDER symbology. If the previous associated bit state was ONE and the input pin is ZERO then sets the bit state to ZERO

##### Property settings

Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %MXa.b, %QXa.b, %IXa.b or specify a tag-name

### 10.4.50 OUTPUT\_P



OUTPUT\_P represents the standard POSITIVE-EDGE output device in the LADDER symbology. If the previous associated bit state was zero and the input pin is one then sets the bit state to one.

#### Property settings

Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %MXa.b, %QXa.b, %IXa.b or specify a tag-name



### 10.4.51 OUTPUT\_R



OUTPUT\_R represents the standard RESET-OUTPUT device in the LADDER symbology. The associated bit (boolean value), expressed by the REFERENCE, is cleared when the input pin is active.

#### Property settings

Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %MXa.b, %QXa.b, %IXa.b or specify a tag-name

**See also :** [EINPUT](#), [NCINPUT](#), [ENCINPUT](#), [OUTPUT](#), [EOUTPUT](#), [RELAY](#)

### 10.4.52 OUTPUT\_S



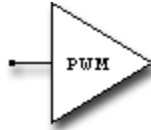
OUTPUT\_S represents the standard SET-OUTPUT device in the LADDER symbology. The associated bit (boolean value), expressed by the REFERENCE, is set when the input pin is active.

#### Property settings

Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %MXa.b, %QXa.b, %IXa.b or specify a tag-name

**See also :** [EINPUT](#), [NCINPUT](#), [ENCINPUT](#), [OUTPUT](#), [EOUTPUT](#), [RELAY](#)

### 10.4.53 PWMOUT



This device, available only for some PLC models, sets an hardware PWM converter, or a generic D/A interface, with the value applied to the input PIN.

The input value must be normalized in the range 0 +65535 which means that zero will give the minimum analog value while 65535 gives the maximum analog value. There are two parameters that control the D/A dynamic : OFFSET and SPAN.

With OFFSET you add a base constant to the value applied to the input. The SPAN parameter allow you to change the converter gain. With values equal to 1.000 no modify will be applid to the input. Values greater than one will increase the gain while valued less than one will produce attenuation.

#### Property settings

Parameter	Description
OFFSET	This parameter adds a OFFSET value to the input value
SPAN	This parameter changes the gain of the D/A converter

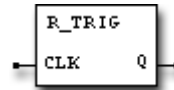
#### Net plugs

Plug	Description
IN	Data input (normalized 0-65535)

See also : [AD\\_CONV](#)

### 10.4.54 R\_TRIG

#### IEC / CEI 1131-3 Compliant



This device is a rising-edge detector. The Q output become TRUE when a 0 to 1 ( or FALSE to TRUE or OFF to ON ) condition is detected on the CLK input and it sustain this state for a complete scan cycle.

#### Net plugs

Plug	Description
CLK	The raising-edge detector input
Q	When a raising-edge is detected this output become true for a single scan cycle

See also : [F\\_TRIG](#)

### 10.4.55 READVAR



This device reads the numerical value relative to the associated variable that was created using the [ASSIGN](#) object.

The name of the variable, that must be supplied through the [REFERENCE](#) parameter must be a variable name that exists in the net.

With the using of the pair ASSIGN / READVAR it is possible to transfer numerical values from one point to another in the net.

#### Property settings

Parameter	Description
REFERENCE	This parameter specify the name of the variable

#### Net plugs

Plug	Description
OUT	This pin gives the value of the associated variable

See also : [ASSIGN](#)

### 10.4.56 RDVAR32



This device reads the numerical value ( 32 bits ) relative to the associated variable that was created using the [ASSIGN32](#) object.

The name of the variable, that must be supplied through the [REFERENCE](#) parameter must be a variable name that exists in the net.

With the using of the pair ASSIGN32 / RDVAR32 it is possible to transfer numerical 32 bits values from one point to another in the net.

#### Property settings

Parameter	Description
REFERENCE	This parameter specify the name of the variable

#### Net plugs

Plug	Description
OUT	This pin gives the value of the associated variable

See also : [ASSIGN32](#)

### 10.4.57 RELAY

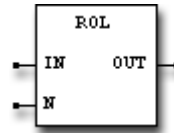


This object always represents a logical device. The RELAY device has a behavior analogue to electromechanical RELAYS of electrical plants. The RELAY device must be used in conjunction with `INPUT`, `NCINPUT`, `EINPUT`, `ENCINPUT` devices. For further information about this argument, see the section [LOGICAL LINKS](#). In details, the RELAY device creates a global boolean variable that can be used by the entire net. The RELAY device transports the value of the input to its output so more than a RELAY can be cascaded together.

**See also :** [INPUT](#), [EINPUT](#), [NCINPUT](#), [ENCINPUT](#), [EOUTPUT](#), [OUTPUT](#)

## 10.4.58 ROL

IEC / CEI 1131-3 Compliant



This function rotates left the word applied to the input IN giving the result on the OUT pin. The number of rotations is specified by the value applied to the pin N. During every single rotation the most significant bit of the word is transferred to the less significant bit.

### Net plugs

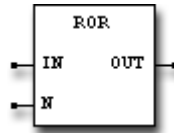
Plug	Description
IN	The word to be rotated
N	The number of rotation to be performed
OUT	The resulting word

See also : [ROR](#), [SHL](#), [SHR](#)



## 10.4.59 ROR

### IEC / CEI 1131-3 Compliant



This function rotates right the word applied to the input IN giving the result on the OUT pin. The number of rotations is specified by the value applied to the pin N. During every single rotation the less significant bit of the word is transferred to the most significant bit.

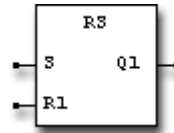
#### Net plugs

Plug	Description
IN	The word to be rotated
N	The number of rotation to be performed
OUT	The resulting word

See also : [ROL](#), [SHL](#), [SHR](#)

## 10.4.60 RS

### IEC / CEI 1131-3 Compliant



This function represents a standard reset-dominant set/reset flip flop. The Q1 output become TRUE when the input S is TRUE and the R1 input is FALSE. In the same way, the Q1 output become FALSE when the input S is FALSE and the R1 input is TRUE. After one of these transitions, when both the S and R1 signals return to FALSE, the Q1 output keeps the previous state until a new condition occurs. If you apply a TRUE condition for both the signals, the Q1 output is forced to FALSE ( reset-dominant ).

#### Net plugs

Plug	Description
S	The SET input
R1	The RESET-DOMINANT input
Q1	The FLIP-FLOP output

See also : [SR](#), [SEMA](#)

### 10.4.61 SCALE\_W



This component allow you to scale a 16 bits unsigned value into another 16 bits unsigned value. The object allow you to define a fixed-point factor that could be 1,2 or three decimals deep. Using fixed point arithmetic allow you to increase overall performances and decrease computation time. Essentially this component scales the input value into the defined output range honouring the given number of fixed decimal digits

The following transformation is applied

#### *Input Values*

*MultFactor = 10,100,1000*

*FinalDiv = 1,10,100,1000*

*inL = Input Low Limit*

*inH = Input High Limit*

*outL = Output Low Limit*

*outH = Output High Limit*

#### *Intermediate computed values*

*SpanIn = ( inH - inL )*

*SpanOut=( outH - outL )*

*Span = ( ( SpanOut \* MultFactor ) / SpanIn )*

#### *Computed output value*

*Out = ( Span \* ( Input - inLow ) + inL ) / FinalDiv*

#### **Property settings**

Parameter	Meaning	Description
REFERENCE	The REFERENCE field	Don't care
FACTOR	Fixed point factor	This parameter allow you to define the fixed point factor that can be one,two or three decimals. This parameter determine the "precision" used by the component to obtain the resulting output. This technique is also known as "Fixed Point Math".

FINALDIV	Final division	This is the divide factor applied as last operation before returning the output value.
INPUT_MIN	Input value (minimum)	This is the low-side input value
INPUT_MAX	Input value (maximum)	This is the high-side input value
OUTPUT_MIN	Output value (minimum)	The low side output value
OUTPUT_MAX X	Output value (maximum)	The high side output value

### Net plugs

Plug	Description
IN	The unsigned 16 bits input value
OUT	The unsigned 16 bits output value

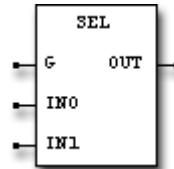
### Some Examples

Input Values	Result
<i>MultFactor = 1000</i> <i>FinalDiv = 1</i>  <i>inL = 0</i> <i>inH = 10</i>  <i>outL = 0.000</i> <i>outH = 100.000</i>  <i>Input value = 5</i>	50
<i>MultFactor = 1000</i> <i>FinalDiv = 1</i>  <i>inL = 0.000</i> <i>inH = 10.000</i>  <i>outL = 0.000</i> <i>outH = 5.000</i>  <i>Input value = 5</i>	2 (2.5 rounded to 2)

<i>MultiFactor</i> = 1000 <i>FinalDiv</i> = 1  <i>inL</i> = 0.000 <i>inH</i> = 12.345  <i>outL</i> = 0.000 <i>outH</i> = 67.898  <i>Input value</i> = 5	<div>27 (27.5 rounded to 27)</div>
--	--

### 10.4.62 SEL

IEC / CEI 1131-3 Compliant



This function block selects one of the two possible values applied on the inputs IN0 and IN1 transferring the value to the OUT pin. The selection of the input is performed through the G input. Placing a zero on the G input will select the value applied to IN0 else the selected value will be IN1.

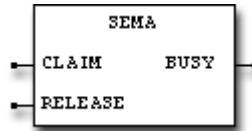
#### Net plugs

Plug	Description
IN0	Input # 0
IN1	Input # 1
G	The selector pin
OUT	This pin reports the value of the selected pin

See also : [MUX](#)

### 10.4.63 SEMA

#### IEC / CEI 1131-3 Compliant



This function block implement a semaphore function. Normally this function is used to synchronize events. The BUSY output is activated by a TRUE condition on the CLAIM input and it is de-asserted by a TRUE condition on the RELEASE input.

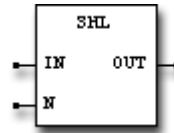
#### Net plugs

Plug	Description
CLAIM	The CLAIM input
RELEASE	The RELEASE input
BUSY	The BUSY output

See also : [RS](#), [SR](#)

### 10.4.64 SHL

IEC / CEI 1131-3 Compliant



This function perform a logical left shift of the value applied to the input IN and the resulting word is available on the OUT pin. The number of shifts is specified by the value applied to the pin N. The most significant bit of the word is filled with zero.

#### Net plugs

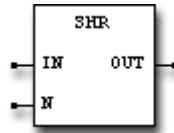
Plug	Description
IN	The word to be shifted
N	The number of shifting to be performed
OUT	The resulting word

See also : [ROL](#), [ROR](#), [SHR](#)



## 10.4.65 SHR

### IEC / CEI 1131-3 Compliant



This function perform a logical right shift of the value applied to the input IN and the resulting word is available on the OUT pin. The number of shifts is specified by the value applied to the pin N. The less significant bit of the word is filled with zero.

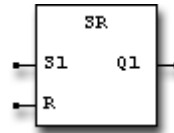
#### Net plugs

Plug	Description
IN	The word to be shifted
N	The number of shifts to be performed
OUT	The resulting word

See also : [ROL](#), [ROR](#), [SHL](#)

## 10.4.66 SR

### IEC / CEI 1131-3 Compliant



This function represents a standard set-dominant set/reset flip flop. The Q1 output become TRUE when the input S1 is TRUE and the R input is FALSE. In the same way, the Q1 output become FALSE when the input S1 is FALSE and the R input is TRUE. After one of these transitions, when both the S1 and R signals return to FALSE, the Q1 output keeps the previous state until a new condition occurs. If you apply a TRUE condition for both the signals, the Q1 output is forced to TRUE ( set-dominant ).

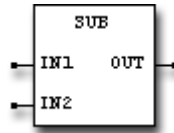
#### Net plugs

Plug	Description
S1	The SET-DOMINAT input
R	The RESET input
Q1	The FLIP-FLOP output

See also : [RS](#), [SEMA](#)

## 10.4.67 SUB

### IEC / CEI 1131-3 Compliant



This device subtract the value present at IN2 from the value present at input IN1 giving the result on the OUT pin.

#### Net plugs

Plug	Description
IN1	First operand
IN2	Second operand
OUT	Result of subtraction

**See also :** [Mathematical Expressions](#), [ADD](#), [MUL](#), [DIV](#), [MOD](#)

### 10.4.68 THRESHLD



The threshold device compares the magnitude of the value present in its input with a pre-programmed value. The value of the output (TRUE/FALSE) is conditioned by the result of this compare according to the QUALIFIER parameter.

#### Meaning of QUALIFIER parameter

Equation	Related function
=	The output becomes true when the input value is equal to the programmed value
<>	The output becomes true when the input value is not equal to the programmed value
>=	The output becomes true when the input value is greater or equal to the programmed value
<=	The output becomes true when the input value is less or equal to the programmed value
>	The output becomes true when the input value is greater than the programmed value
<	The output becomes true when the input value is less than the programmed value

#### Property settings

Parameter	Description
COMPARE VALUE	The value to be compared with the input value
QUALIFIER	This parameter will establish the comparing equation

#### Net plugs

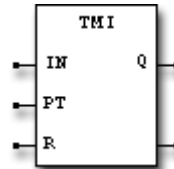
Plug	Description
INPUT	The comparator input

OUT                      The comparator output

**See also :** [CMP\\_W](#)

### 10.4.69 TMI

#### IEC / CEI 1131-3 Compliant



This device, also called INTEGRAL TIMER, accumulates the sum of the time for the periods where the IN input is in assert state. This means that the computed time is the sum of the ON times of the IN input.

If the computed sum reaches the programmed time, applied to the PT input, the Q output become TRUE.

At this point, the only way to reset the integral timer can be performed asserting the R ( RESET ) input.

The ET output pin reports the current elapsed time.

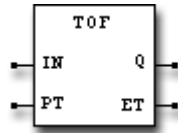
#### Net plugs

Plug	Description
IN	The timer accumulates the time when this input is TRUE
R	The integral timer may be resetted using this input
PT	To this input user must apply the programmed time. User should use an <a href="#">IDENT</a> object to enter a time constant
Q	After that the pre-programmed time threshold is reached, this output become TRUE
ET	This output reports the current elapsed time

See also : [TP](#), [TON](#), [TOF](#), [TSQ](#), [IDENT](#)

## 10.4.70 TOF

### IEC / CEI 1131-3 Compliant



Asserting the input signal IN of this device immediately activates the Q output. At this point, releasing the input IN will start the time elapsing. When the programmed time, applied to the input PT, is elapsed and the input IN is still de-asserted, the Q output become FALSE. This condition will be kept until the input IN is remains de-asserted. If the IN input is asserted again before time elapsing, the time counting will be cleared and the Q output remains ON. The ET output pin reports the current elapsed time.

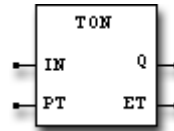
#### Net plugs

Plug	Description
IN	A TRUE condition on this input starts the TOF timer
PT	To this input user must apply the programmed time. User should use an <a href="#">IDENT</a> object to enter a time constant
Q	This output become TRUE after an high to low transition on pin IN and if the programmed time is elapsed
ET	This output reports the current elapsed time

See also : [TP](#), [TMI](#), [TON](#), [TSQ](#) , [IDENT](#)

### 10.4.71 TON

#### IEC / CEI 1131-3 Compliant



Asserting the input signal IN of this device starts the time elapsing of timer. When the programmed time, applied to the input PT, is elapsed and the input IN is still asserted, the Q output become TRUE. This condition will continue until the input IN is deasserted. If the IN input is released before time elapsing, the timer will be cleared. The ET output pin reports the current elapsed time.

#### Net plugs

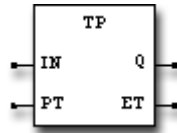
Plug	Description
IN	A TRUE condition on this input starts the TON timer
PT	To this input user must apply the programmed time. User should use an <a href="#">IDENT</a> object to enter a time constant
Q	If the input IN is asserted and the programmed time is elapsed this output become TRUE
ET	This output reports the current elapsed time

**See also :** [TP](#), [TMI](#), [TOF](#), [TSQ](#), [IDENT](#)



## 10.4.72 TP

### IEC / CEI 1131-3 Compliant



This kind of timer has the same behavior of a single-shot timer or a monostable timer. When a rising-edge ( OFF to ON or FALSE to TRUE ) transition is detected on the IN input, the Q output become immediately TRUE .

This condition continue until the programmed time PT, applied to the relative pin, is elapsed. After that the programmed time is elapsed, the Q output keeps the ON state if the input IN is still asserted else the Q output returns to the OFF state.

This timer is not re-triggerable. This means that after the timer started it can't be stopped until the complete session ends.

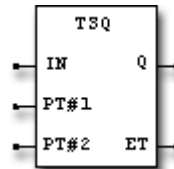
The ET output pin reports the current elapsed time.

#### Net plugs

Plug	Description
IN	A rising-edge on this pin will trigger the pulse timer
PT	To this input user must apply the programmed time. User should use an <a href="#">IDENT</a> object to enter a time constant
Q	When the pulse timer starts, and before that the programmed time is elapsed, this output become TRUE
ET	This output reports the current elapsed time

**See also :** [TMI](#), [TON](#), [TOF](#), [TSQ](#) , [IDENT](#)

### 10.4.73 TSQ



This device, also called SQuare wave timer, allow to generate square waves with variable duty-cycle. When the input is asserted, the time elapsing proceeds. When the elapsed time reaches the pre-programmed threshold, applied to the input PT#1, the Q output become TRUE. Time elapsing continue until the second pre-programmed threshold, applied to the input PT#2 is reached. When this condition become true, the Q output is putted to OFF state and the elapsed time is cleared so a new cycle could begin.

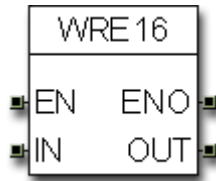
De-asserting the IN input will freeze the timer in the last condition. Applying a ON state again on the IN input will start the timer at the same point where it was left. The ET output pin reports the current elapsed time.

#### Net plugs

Plug	Description
IN	The square wave generation proceeds when this input is asserted
PT#1	The Q output become TRUE when the elapsed time reaches the value applied to this input. User should use an <a href="#">IDENT</a> object to enter a time constant
PT#2	The Q output become FALSE when the elapsed time reaches the value applied to this input. User should use an <a href="#">IDENT</a> object to enter a time constant
Q	This output is function of the pre-programmed time PT#1 and PT#2
ET	This output reports the current elapsed time

See also : [TP](#), [TMI](#), [TON](#), [TOF](#), [IDENT](#)

#### 10.4.74 WRE16



This function block allow you to write the associated variable, defined by the REFERENCE field, only when the EN (Enable) pin is active. According to this, this block allow you to have a "gate-driven-writing". The OUT pin reflects the value of the associated variable.

##### Property settings

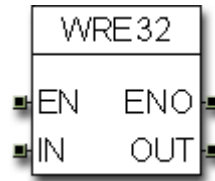
Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %Mx or specify a tag-name

##### Net plugs

Plug	Description
EN	Enable input. The input value (16 bits), applied to the IN pin, will be writed in the associated variable when this signal is active
ENO	Enable output. The ENO output reflects the value applied on the EN pin
IN	The input value that will be written to the associated variable
OUT	The OUT pin reflects the value of the associated variable

See also : [WRE32](#)

### 10.4.75 WRE32



This function block allow you to write the associated variable, defined by the REFERENCE field, only when the EN (Enable) pin is active. According to this, this block allow you to have a "gate-driven-writing". The OUT pin reflects the value of the associated variable.

#### Property settings

Parameter	Description
REFERENCE	The REFERENCE field in the properties must refer to a word in the table. User can use a direct address identifier %Mx or specify a tag-name

#### Net plugs

Plug	Description
EN	Enable input. The input value (32 bits), applied to the IN pin, will be writed in the associated variable when this signal is active
ENO	Enable output. The ENO output reflects the value applied on the EN pin
IN	The input value that will be written to the associated variable
OUT	The OUT pin reflects the value of the associated variable

See also : [WRE16](#)

#### 10.4.76 XOR



The XOR device performs a logical XOR between two boolean signals. The output become true when the input signals are different.

**See also :** [AND](#), [NOT](#), [OR](#)

## 10.5 Error messages

The software error messages are catalogued in this section

### 10.5.1 Ladder/FBD NET errors

This section reports the common error messages for a ladder diagram. This category of error is expressed in the form NETxxxx

#### 10.5.1.1 NET0706

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_NETWORK_TOO_COMPLEX
<b>Category</b>	FATAL ERROR
<b>Description</b>	The diagram contains too many objects or connections
<b>Possible cause</b>	Many
<b>Possible solution</b>	Try to split the diagram in smaller portions and avoid complex connections between elements

### 10.5.1.2 NET0707

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_INTERNAL_LIMIT_REACHED
<b>Category</b>	FATAL ERROR
<b>Description</b>	The diagram contains too many elements (components, connections and so on)
<b>Possible cause</b>	Many
<b>Possible solution</b>	Normally there is no solution to this error and the diagram complexity must be reduced

### 10.5.1.3 NET0708

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_NET_TOO_MANY_PLUGS_FOR_NODE
<b>Category</b>	FATAL ERROR
<b>Description</b>	The diagram contains a node with a number of connections greater than 16
<b>Possible cause</b>	Multiple connections on a node with more than 16 attached pins
<b>Possible solution</b>	Divide the node using input and outputs blocks or use ASSIGN/READVAR pairs to split the diagram

**10.5.1.4 NET0709**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_NET_TOO_MANY_PLUGS
<b>Category</b>	FATAL ERROR
<b>Description</b>	The diagram contains a too large number of PLUGS (Component's pins)
<b>Possible cause</b>	Many
<b>Possible solution</b>	Reduce diagram complexity

**10.5.1.5 NET0720**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_NET_TOO_MANY_POLARISATION
<b>Category</b>	FATAL ERROR
<b>Description</b>	A bidirectional components was evaluated for more than two times. Normally this condition could happen in case of close-circle connections
<b>Possible cause</b>	Many
<b>Possible solution</b>	Check for close-circle connections



**10.5.1.6 NET0710**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_NET_TOO_MANY_CONSTANTS
<b>Category</b>	FATAL ERROR
<b>Description</b>	The diagram contains a too large number of CONSTANT (Component's properties elements)
<b>Possible cause</b>	Many
<b>Possible solution</b>	Reduce diagram complexity

**10.5.1.7 NET0754**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_EMPTY_IDENTIFIER
<b>Category</b>	ERROR
<b>Description</b>	A component requires an IDENTIFICATOR in the REFERENCE field of the component properties (Variable or direct IEC 1131-3 address) that was not entered
<b>Possible cause</b>	The component REFERENCE is required but was not entered
<b>Possible solution</b>	Enter the REFERENCE into the component's properties

**10.5.1.8 NET0757**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_REFERENCE_TOOLONG
<b>Category</b>	ERROR
<b>Description</b>	The REFERENCE string entered in the variable table or directly into the component properties is too long
<b>Possible cause</b>	REFERENCE string too long
<b>Possible solution</b>	Reduce the REFERENCE string length

**10.5.1.9 NET0758**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_VAR_IDENT_EXPECTED
<b>Category</b>	ERROR
<b>Description</b>	A component requires an IDENTIFICATOR in the REFERENCE field of the component properties (Variable or direct IEC 1131-3 address) that was not entered
<b>Possible cause</b>	The component REFERENCE is required but was not entered
<b>Possible solution</b>	Enter the REFERENCE string as required

**10.5.1.10 NET0760**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_MISPLACED_GROUND
<b>Category</b>	ERROR
<b>Description</b>	A GND component was placed in in a strange mix of connection
<b>Possible cause</b>	Misplaced GND component
<b>Possible solution</b>	GND component must be used to terminate a rung only

**10.5.1.11 NET0794**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_NET_JAM
<b>Category</b>	ERROR
<b>Description</b>	The NETLIST file contains inchoerent informations
<b>Possible cause</b>	The NETLIST file contains inchoerent informations
<b>Possible solution</b>	Please contact MicroSHADOW Research

**10.5.1.12 NET0795**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_NET_NODE_NUMBER
<b>Category</b>	ERROR
<b>Description</b>	Node number out of range parsing NETLIST file

<b>Possible cause</b>	Probable software bug
<b>Possible solution</b>	Please contact MicroSHADOW Research

#### 10.5.1.13 NET0796

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_NET_UNEXPECTED_EOF
<b>Category</b>	ERROR
<b>Description</b>	Unexpected EOF processing NETLIST file
<b>Possible cause</b>	Many
<b>Possible solution</b>	Please contact MicroSHADOW Research

#### 10.5.1.14 NET0797

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_NET_FILE_IO_ERROR
<b>Category</b>	ERROR
<b>Description</b>	A severe I/O error occurred during the processing of the NETLIST file
<b>Possible cause</b>	File system errors, sharing and access errors.
<b>Possible solution</b>	Please contact MicroSHADOW Research

**10.5.1.15 NET0798**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETERR_SYNTAX_IN_NET_FILE
<b>Category</b>	ERROR
<b>Description</b>	Generic error in .NET file
<b>Possible cause</b>	The .NET file (The intermediate NETLIST of the diagram) contains errors
<b>Possible solution</b>	Probable software bug, please contact MicroSHADOW Research

**10.5.1.16 NET0802**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETWAR_TIME_DURATION_NODE_JAM
<b>Category</b>	WARNING
<b>Description</b>	A time/duration constant (t#...) is connected to a node that contains other elements (BOOLEAN and WORD paths)
<b>Possible cause</b>	Time/duration constant mixed with other kind of paths
<b>Possible solution</b>	Never mix duration constants with other elements

**10.5.1.17 NET0803**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETWAR_TIME_DURATION_WRONG_FEEDING
<b>Category</b>	WARNING
<b>Description</b>	
<b>Possible cause</b>	
<b>Possible solution</b>	

**10.5.1.18 NET0804**

<b>Module</b>	SOLVER MODULE
<b>Token</b>	NETWAR_DEPRECATED_GND_FEEDING_INPUT
<b>Category</b>	ERROR
<b>Description</b>	The GND component ( GROUND ) can't be used to feed an INPUT pin
<b>Possible cause</b>	A GND component is connected to an INPUT pin
<b>Possible solution</b>	Don't connect GND components to INPUT pins but use only to end a rung or a diagram tree

# Part

---



XI

## 11 MODBUS/UDP

In this section we discuss technical informations about the integrated MODBUS/UDP protocol. The protocol uses the Ethernet media as communication transport layer and allow to read/write information from/to the PLC

### IMPORTANT INFORMATIONS

- MODBUS/UDP uses port 502
- MODBUS/UDP IP address is factory set to 192.168.5.52

### 11.1 What is MODBUS/UDP

MODBUS/UDP is a protocol directly derived by the industry standard MODBUS. MODBUS/UDP uses Ethernet instead of the canonical RS232C/RS485 interface

### 11.2 MODBUS/UDP Packet Format

The MODBUS/UDP packets are "encapsulated" inside a standard UDP (User Datagram Protocol) packet

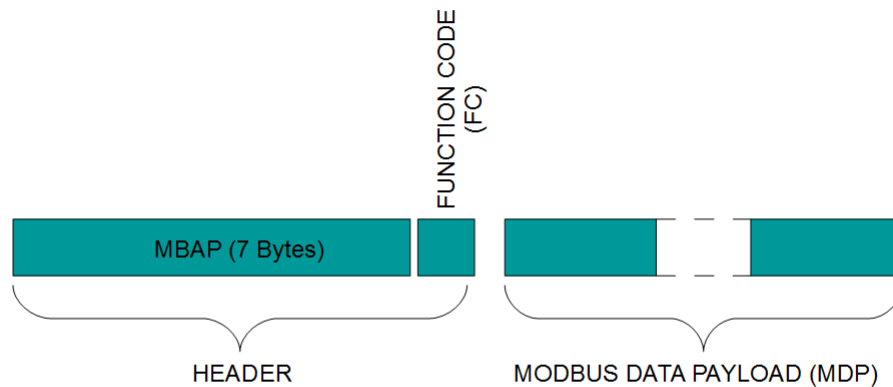
The UDP payload is divided in three major fields which are

- MBAP (MODBUS Application Protocol)
- FC (Function Code)
- MDP (MODBUS Data Payload)

The MBAP and the FC code, together, represent the HEADER while the MDP field contains the data related to the request/response being issued



## MODBUS/UDP PACKET FORMAT



### 11.3 MBAP Block

A dedicated header is used on UDP packet to identify the MODBUS Application Data Unit. This header is called MBAP (MODBUS Application Protocol)

This header provides some differences compared to the MODBUS RTU application data unit used on a regular serial line (RS232C or RS485)

- The MODBUS 'slave address' field usually used on MODBUS Serial Line is replaced by a single byte named 'Unit Identifier' within the MBAP Header. The 'Unit Identifier' is used to communicate via devices such as bridges, routers and gateways that use a single IP address to support multiple independent MODBUS end units
- All MODBUS requests and responses are designed in such a way that the recipient can verify that a message is finished. For function codes where the MODBUS PDU has a fixed length, the function code alone is sufficient. For function codes carrying a variable amount of data in the request or response, the data field includes a byte count. *f*
- When MODBUS is carried over UDP, additional length information is carried in the MBAP header to allow the recipient to recognize message boundaries even if the message has been split into multiple packets for transmission. The existence of explicit and implicit length rules, and use of a CRC-32 error check code (on Ethernet) results in an infinitesimal chance of undetected corruption to a request or response message.

The MBAP Header contains the following fields

Field	Lenght	Description	Client	Server
-------	--------	-------------	--------	--------

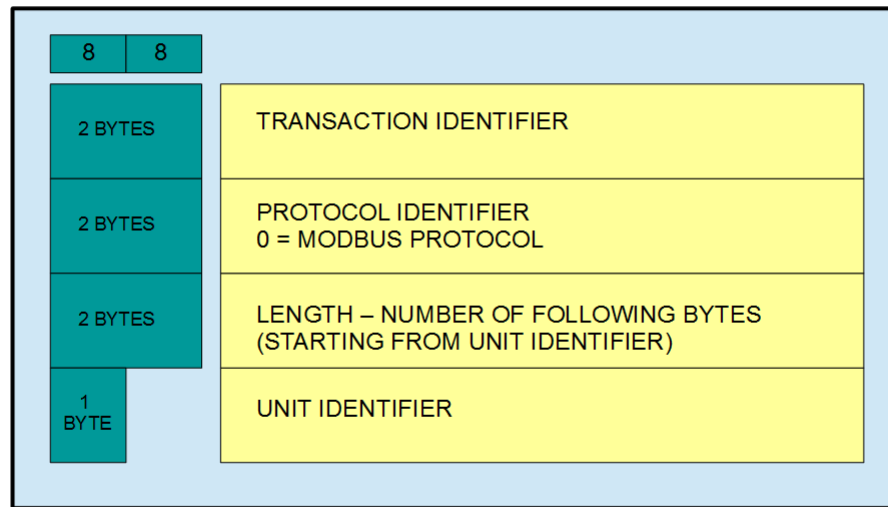
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction	Initialised by the client	Recopied by the server form the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialised by the client	Recopied by the server form the received request
Lenght	2 Bytes	Number of following bytes	Initialised by the client (request)	Initialised by the server (response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses	Initialised by the client	Recopied by the server form the received request

The header is 7 bytes long and is formed by

- Transaction Identifier - It is used for transaction pairing, the MODBUS server copies in the response the transaction identifier of the request
- Protocol Identifier – It is used for intra-system multiplexing. The MODBUS protocol is identified by the value 0
- Length - The length field is a byte count of the following fields, including the Unit Identifier and data fields
- Unit Identifier – This field is used for intra-system routing purpose. It is typically used to communicate to a MODBUS+ or a MODBUS serial line slave through a gateway between an Ethernet UDP network and a MODBUS serial line. This field is set by the MODBUS Client in the request and must be returned with the same value in the response by the server

All MODBUS/TCP ADU are sent via UDP to registered port 502

## MBAP HEADER



### 11.4 MOSBUS Function Codes

In this section is described, in detail, the implemented MODBUS function codes, The function code is sent, in the UDP packet, in the FC field

#### 11.4.1 Read Holding Registers (Code = 0x03)

The function code 0x03 (Read Holding Registers) allow to read informations, randomly, from the main PLC table

The PLC module acts like SERVER and the request should be prepared by a host system working as CLIENT. The PLC table is 16-bits word aligned array of general purpose data registers. The register address is in the range from zero to the maximum number of registers of your PLC (See specific technical data). For example, if my PLC has 1000 words, my address range is from 0 to 999

The fields related with the request are listed here

Field	Lenght	Description
Start Address	2 Bytes	This field keeps the "first location address" that will be readed from the PLC table. The field represent a 16bits-aligned entity and can span from 0 to the maximum table size
Quantity of Registers	2 Bytes	This is number of registers that will be readed starting from the first specified address

## Sample UDP Packet Session (Request/Response)

REQUEST (SENT BY CLIENT SYSTEM)					
Fields	Mnemonic	Description	Lenght	Sample Data	Sent bytes and order
MBAP		Transaction Identifier	2 Bytes	0x2356	23H 56H
		Protocol Identifier	2 Bytes	0x0000	00H 00H
		Lenght	2 Bytes	0x0006	00H 06H
		Unit Identifier	1 Byte	0x05	05H
MODBUS Request	FC	Function Code	1 Byte	0x03	03H
		Start Address	2 Bytes	0x0240	02H 40H
		Quantity of Registers	2 Bytes	0x0004	00H 04H

RESPONSE (SENT BY SERVER PLC)					
Fields	Mnemonic	Description	Lenght	Sample Data	Sent bytes and order
MBAP		Transaction Identifier	2 Bytes	0x2356	23H 56H
		Protocol Identifier	2 Bytes	0x0000	00H 00H
		Lenght	2 Bytes	0x0006	00H 06H
		Unit Identifier	1 Byte	0x05	05H
MODBUS Response	FC	Function Code	1 Byte	0x03	03H
		Content of register 0x0240	2 Bytes	0x1234	12H 34H
		Content of register 0x0241	2 Bytes	0x5678	56H 78H
		Content of register 0x0242	2 Bytes	0x9ABC	9AH BCH
		Content of register 0x0243	2 Bytes	0xDEFO	DEH FOH

**Note:** Readed data is sent using Big Endian notation (MSB byte first)

### 11.4.2 Write Mutiple Registers (Code = 0x10)

The function code 0x10 (Write Multiple Registers) allow to write informations, randomly, into the main PLC table

The PLC module acts like SERVER and the request should be prepared by a host system working as CLIENT. The PLC table is 16-bits word aligned array of general purpose data registers. The register address is in the range from zero to the maximum number of registers of your PLC (See specific technical data). For example, if my PLC has 1000 words, my address range is from 0 to 999

The fields related with the request are listed here

Field	Lenght	Description
Start Address	2 Bytes	This field keeps the "first location address" that will be written to the PLC table. The field represent a 16bits-aligned entity and can span from 0 to the maximum table size
Quantity of Registers	2 Bytes	This is number of registers that will be written starting from the first specified address
Data	(Quantity of Registers) * 2 Bytes	Data to be written. 16bits aligned entity

Sample UDP Packet Session (Request/Response)  
Writing 4 registers starting from address 0x240

REQUEST (SENT BY CLIENT SYSTEM)					
Fields	Mnemonic	Description	Lenght	Sample Data	Sent bytes and order
<b>MBAP</b>		Transaction Identifier	2 Bytes	0x2356	23H 56H
		Protocol Identifier	2 Bytes	0x0000	00H 00H
		Lenght	2 Bytes	0x0006	00H 06H
		Unit Identifier	1 Byte	0x05	05H
<b>MODBUS Request</b>	<b>FC</b>	Function Code	1 Byte	0x10	10H
		Start Address	2 Bytes	0x0240	02H 40H
		Quantity of Registers	2 Bytes	0x0004	00H 04H
		Data#0	2 Bytes	0x1234	12H 34H
		Data#1	2 Bytes	0x5678	56H 78H
		Data#2	2 Bytes	0x9ABC	9AH BCH
		Data#3	2 Bytes	0xDEFO	DEH FOH

### RESPONSE (SENT BY SERVER PLC)

Fields	Mnemonic	Description	Lenght	Sample Data	Sent bytes and order
MBAP		Transaction Identifier	2 Bytes	0x2356	23H 56H
		Protocol Identifier	2 Bytes	0x0000	00H 00H
		Lenght	2 Bytes	0x0006	00H 06H
		Unit Identifier	1 Byte	0x05	05H
MODBUS Response	FC	Function Code	1 Byte	0x10	10H
		Start Address(	2 Bytes	0x0240	02H 40H
		Quantity of Registers	2 Bytes	0x0004	00H 04H

**Note:** Data to write is sent using Big Endian notation (MSB byte first)

# Part

---



XII

## 12 MODBUS RTU (RS232/RS485)

In this section we discuss technical informations about the integrated MODBUS RTU protocol (RS232C/RS485)

### 12.1 WHAT IS MODBUS RTU

MODBUS RTU is a very popular industry standard protocol. MODBUS RTU uses a standard RS232C/RS485 bus

The MODBUS protocol specifies one master and up to 247 slaves on a common communication line, each slave is assigned a fixed unique device address in the range 1 to 247. The master always initiates a transaction. Transactions are either a query/response type ( only a slave is accessed at a time ) or a broadcast/no response type ( all slaves are accessed at the same time). A transaction comprises a single query and a single response frame or a single broadcast frame.

Frame synchronization is maintained in RTU transmission mode only by simulating a synchronous message. The MODBUS protocol uses a timeout to determine the completion of the frame. This parameter is often called EOF (End Of Frame).

#### 12.1.1 MODBUS RTU Function Codes

The implemented MODBUS RTU function codes are described in this section

##### 12.1.1.1 Read Holding Registers (Code = 0x03)

The function code 0x03 (Read Holding Registers) allow to read informations, randomly, from the main PLC table

The PLC module acts like SERVER and the request should be prepared by a host system working as CLIENT. The PLC table is 16-bits word aligned array of general purpose data registers. The register address is in the range from zero to the maximum number of registers of your PLC (See specific technical data). For example, if my PLC has 1000 words, my address range is from 0 to 999

The fields related with the request are listed here

Field	Lenght	Description
-------	--------	-------------



Start Address	2 Bytes	This field keeps the "first location address" that will be readed from the PLC table. The field represent a 16bits-aligned entity and can span from 0 to the maximum table size
Quantity of Registers	2 Bytes	This is number of registers that will be readed starting from the first specified address

Sample RS232/RS485 Packet Session (Request/Response)

REQUEST (SENT BY CLIENT SYSTEM)					
Fields	Mnemonic	Description	Lenght	Sample Data	Sent bytes and order
MODBUS Request	FC	Function Code	1 Byte	0x03	03H
		Start Address	2 Bytes	0x0240	02H 40H
		Quantity of Registers	2 Bytes	0x0004	00H 04H
	CRC	CRC	2 Bytes	0xABCD	ABH CDH

RESPONSE (SENT BY SERVER PLC)					
Fields	Mnemonic	Description	Lenght	Sample Data	Sent bytes and order
MODBUS Response	FC	Function Code	1 Byte	0x03	03H
		Content of register 0x0240	2 Bytes	0x1234	12H 34H
		Content of register 0x0241	2 Bytes	0x5678	56H 78H
		Content of register 0x0242	2 Bytes	0x9ABC	9AH BCH
		Content of register 0x0243	2 Bytes	0xDEF0	DEH F0H
	CRC	CRC	2 Bytes	0xABCD	ABH CDH

**Note:** Readed data is sent using Big Endian notation (MSB byte first)

### 12.1.1.2 Write Multiple Registers (Code = 0x10)

The function code 0x10 (Write Multiple Registers) allow to write informations, randomly, into the main PLC table

The PLC module acts like SERVER and the request should be prepared by a host system working as CLIENT. The PLC table is 16-bits word aligned array of general purpose data registers. The register address is in the range from zero to the maximum number of registers of your PLC (See specific technical data). For example, if my PLC has 1000 words, my address range is from 0 to 999

The fields related with the request are listed here

Field	Lenght	Description
Start Address	2 Bytes	This field keeps the "first location address" that will be written to the PLC table. The field represent a 16bits-aligned entity and can span from 0 to the maximum table size
Quantity of Registers	2 Bytes	This is number of registers that will be written starting from the first specified address
Data	(Quantity of Registers) * 2 Bytes	Data to be written. 16bits aligned entity

Sample RS232/RS485 Packet Session (Request/Response)  
Writing 4 registers starting from address 0x240

REQUEST (SENT BY CLIENT SYSTEM)					
Fields	Mnemonic	Description	Lenght	Sample Data	Sent bytes and order
MODBUS Request	FC	Function Code	1 Byte	0x10	10H
		Start Address	2 Bytes	0x0240	02H 40H
		Quantity of Registers	2 Bytes	0x0004	00H 04H
		Data#0	2 Bytes	0x1234	12H 34H
		Data#1	2 Bytes	0x5678	56H 78H
		Data#2	2 Bytes	0x9ABC	9AH BCH
		Data#3	2 Bytes	0xDEFO	DEH FOH
	CRC	CRC	2 Bytes	0xABCD	ABH CDH

RESPONSE (SENT BY SERVER PLC)					
Fields	Mnemonic	Description	Lenght	Sample Data	Sent bytes and order

<b>MODBUS Response</b>	<b>FC</b>	Function Code	1 Byte	0x10	10H
		Start Address	2 Bytes	0x0240	02H 40H
		Quantity of Registers	2 Bytes	0x0004	00H 04H
	<b>CRC</b>	CRC	2 Bytes	0xABCD	ABH CDH

**Note:** Data to write is sent using Big Endian notation (MSB byte first)

### 12.1.2 MODBUS RTU With LadderDIP IV

When using LadderDIP IV with MODBUS RTU the following memory mapping is applied

MODBUS Address Range Low	MODBUS Address Range High	Domain	Access Type	Range Size	Variable accessed in the PLC table
10000	13999	<b>ALL TABLE</b>	WORD	4000 Words	0-3999 (%MW0-%MW3999)
40000	41799	<b>USER</b>	WORD	1800 Words	2200-3999 (%MW2200-%MW3999)
5000	6999	<b>USER</b>	DWORD	1800 Words	2200-3999 (%MD2200-%MD3999)

The MODBUS Configuration is accessible through **Options → Configuration → MODBUS RTU Configuration**

LadderWORK ARM(R) Configuration

Target | Variables | Pin configuration | Global Parameters | **MODBUS RTU Configuration**

MODBUS RTU Configuration

☒ Enable MODBUS RTU Kernel

Parms

☒ ENABLE MODBUS RTU On UART2 (LadderDIP IV P2.11/P2.12)

Station Address  
1

Baud rate  
115200

RS485 TX Control Pin  
Use IO8 to control the RS485 TX Pin (P1-9 - On Micro P2.3)

Byte-To-Byte Timing (BTB) 50 ms      End-Of-Frame Timing (EOF) 100 ms

Cancel OK

MODBUS Configuration parameters are explained in the table here below

Parameter	Mnemonic	Description
Enable MODBUS RTU Kernel		Enables the task (Kernel) for the MODBUS communication
Enable MODBUS RTU on UART2		Enables the communication UART on the specified pins (TX/RX)
Station Address	<b>SA</b>	The MODBUS station address (1..247)
Baud Rate	<b>BR</b>	UART Baud Rate
RS485 TX Control Pin		Selects, if any, the pin that will be used as TX control pin for a half-duplex RS485 connection
Byte-To-Byte Timing	<b>BTB</b>	Byte to Byte timeout
End-Of-Frame Timing	<b>EOF</b>	End of frame timeout

# Index

## - A -

AD\_CONV 104  
ADD 105  
AND 106  
ASSIGN 107

## - B -

BIT 109  
BIT32 110  
Building menu 89  
Building the code 89, 95

## - C -

CEI / IEC 1131-3 programming languages 59  
Close Project 87  
Compile 89  
Compile & Upload 89  
Compile & Upload & Run 89  
Compile bar 85  
Components bar 85  
Connecting components 92

## - D -

Deleting components 92

## - E -

Edit menu 88  
Exit 87

## - F -

File menu 87

## - G -

Grid 90

## - H -

Housekeeping 59

## - I -

IDE 83, 85  
Input read and output write scan 58  
Installing the software 84  
Integrated developemnt enviroment 83, 85

## - L -

Ladder logic 60  
Ladder logic's elements 61  
Launching the program 84

## - M -

Menu 85, 86  
Message window 85  
Moving components 92

## - N -

New 87

## - O -

Open 87  
Options menu 91

## - P -

Placing components 92  
PLC Overview 58  
Print 87  
Print Preview 87  
Print Setup 87  
Program scan 59  
Protection key 84

## - R -

Reference Grid 90  
Run 89

Run-time environment 58

## - S -

Save 87

Save As 87

Set components property 94

Standard toolbar 85

Status bar 85

Stop 89

System Edit 91

System requirements 84

## - T -

Tool bar 85

## - V -

View menu 90

## - W -

Watch window 85

## - Z -

Zoom 1:1 91

Zoom Fit Screen 91

Zoom In 91

Zoom menu 91

Zoom Out 91



Back Cover