LD MICRO

Master manual

INTRODUCTION:

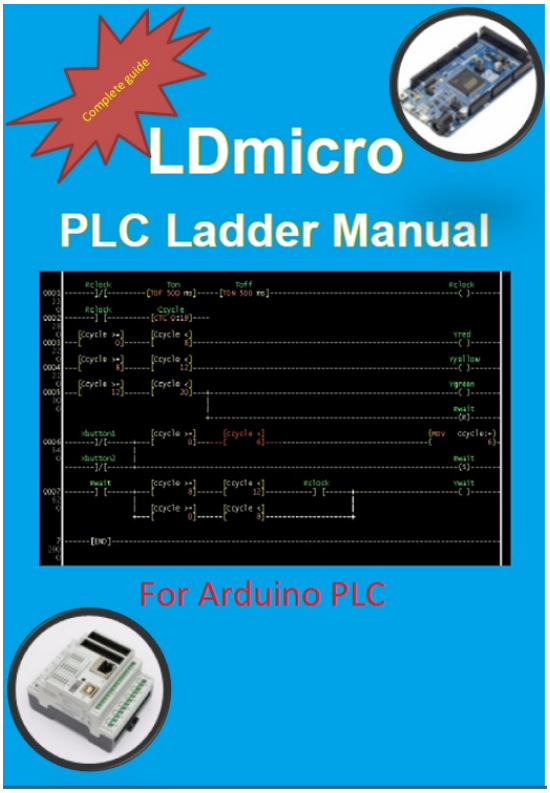
LDmicro compiles ladder logic to PIC16 or AVR code. The following processors are supported:

- * PIC16F877
- * PIC16F628
- * PIC16F876
- * PIC16F88
- * PIC16F819
- * PIC16F887
- * PIC16F886
- * PIC16F72
- * PIC12F675
- * PIC12F683
- * PIC16F1512
- * PIC16F1516
- * PIC16F1527
- * PIC10F202
- * PIC10F200
- * ATmega128
- * ATmega64
- * ATmega162

* ATmega32

- * ATmega32U4
- * ATmega16
- * ATmega8
- * ESP8266
- * STM32

* "PASCAL Code" for PC LPT and COM



Introduction to structured programming

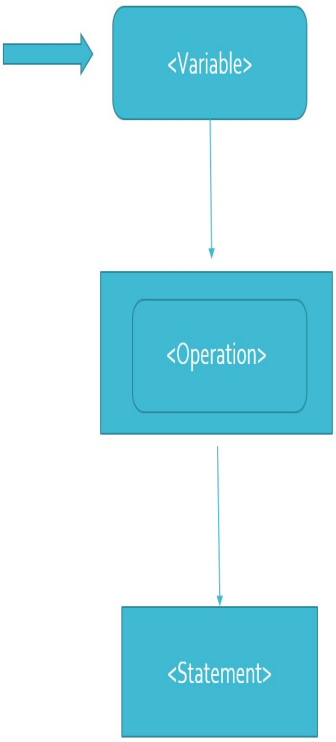
Basic - Variables

Overview of Pascal Data Types:

Name	Type of Data	Examples
String	Holds Text	'New York', 'Evan'
Integer	Holds whole numbers	3, 6, 1024
Real	Holds Decimal Numbers	3.14, 503.2
Boolean	Holds True or False	TRUE, FALSE
Character	Holds a single character	'A', 'E'

Variable Ranges

Data Type	Minimum Value	Maximum Value
Integer	-32,768	32,767
LongInt	-2,147,483,648	2,147,487,647
ShortInt	-128	128
Real	2.9 x 10 E-39	1.7 x 10 E+38




```
Var <Name>:bool;

Begin
<Name>:=<Operation>;

<statement>:= <Name>;

End;
```

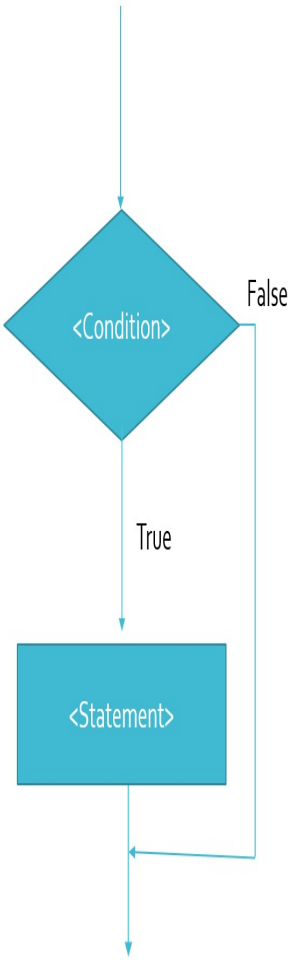

Introduction to structured programming

Basic-
(IF-Then)

Have the code
or the operation.

If <Condition>Then
<Statement>
End_If;

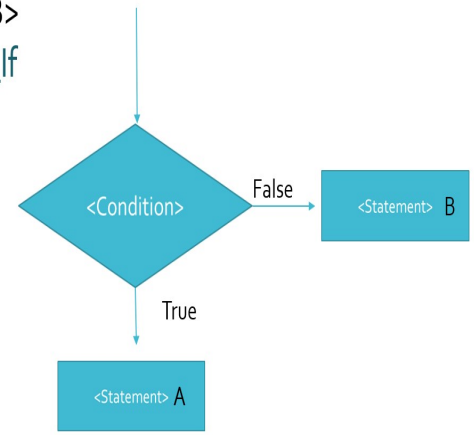
Reaction of the code



Introduction to structured programming

Basic- Else

```
If <Condition>Then
<Statement_A>
Else
<Statement_B>
        End_If
```



Continue A>>

Introduction to structured programming

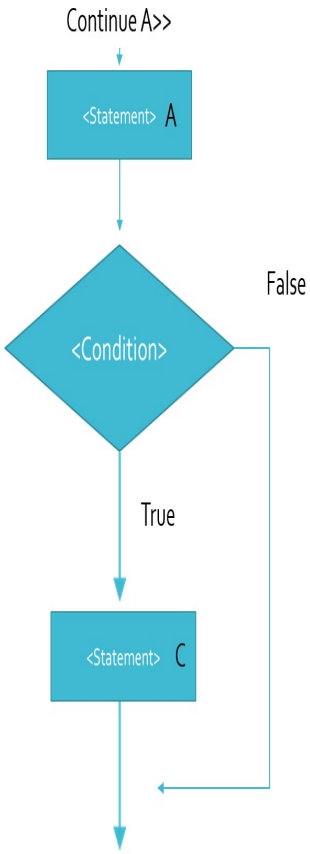
Basic-Mode operator

<Stament_A>

If <Condition> Then

<statement_C>

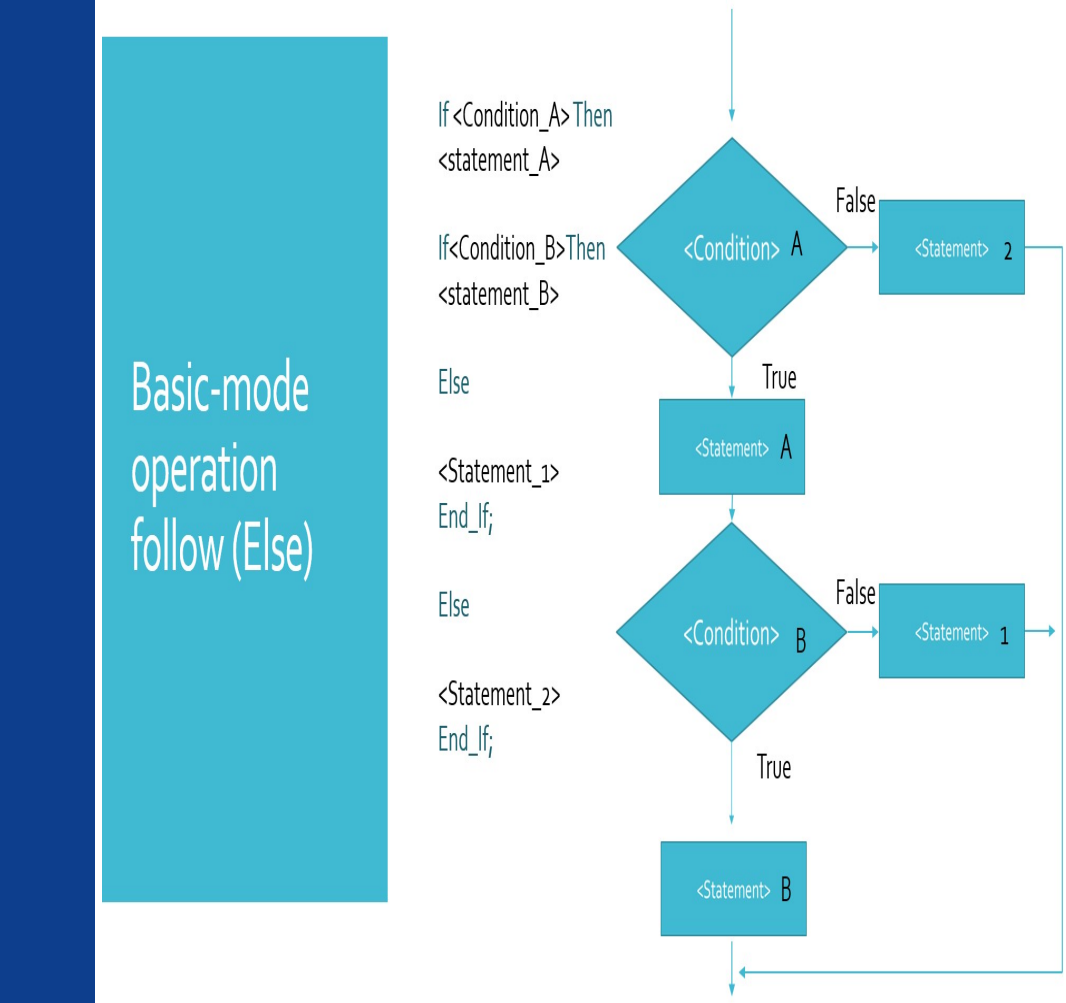
End_If;



Introduction to structured programming

Basic-mode operation follow (Else)

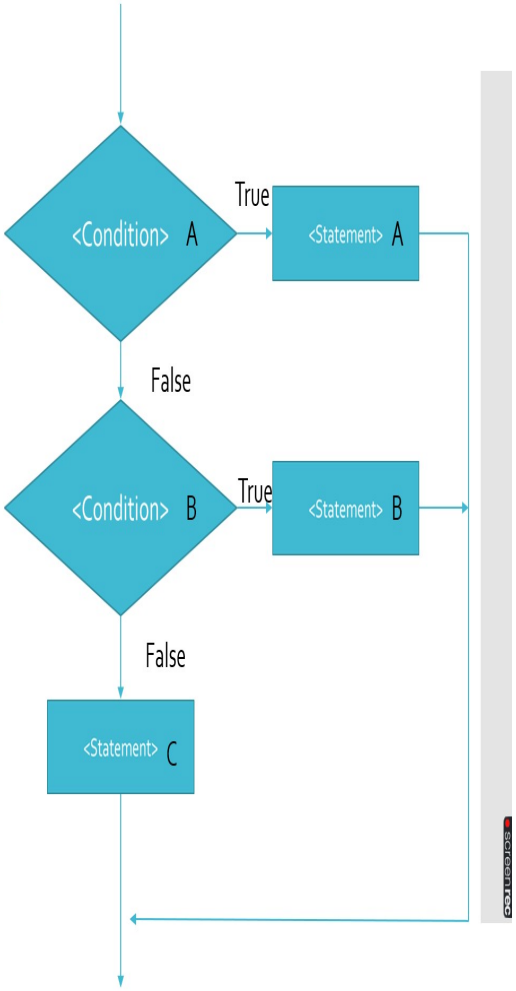
```
If <Condition_A> Then
<statement_A>

If<Condition_B>Then
<statement_B>

Else

<Statement_1>
End_If;

Else

<Statement_2>
End_If;
```



Introduction to structured programming

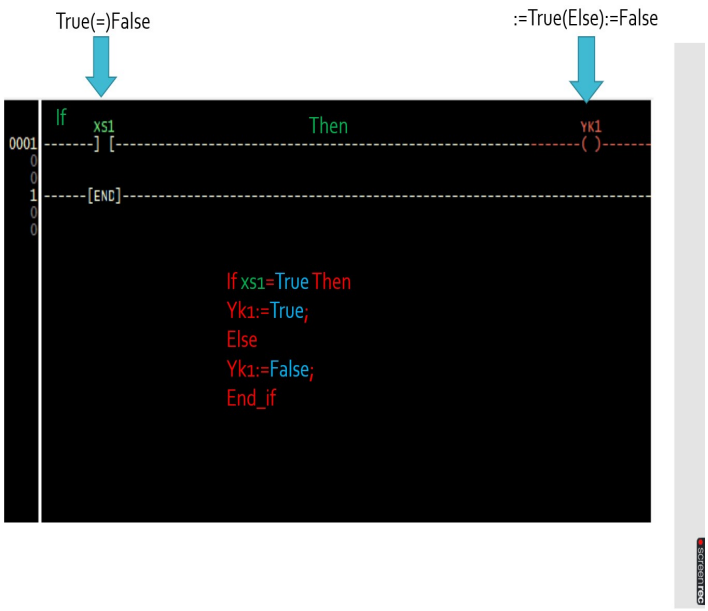
Basic- Elsif

```
If<Condition_A>Then
<Statement_A>

Elsif <Condition_B>Then
<Statement_B>

Else

<Statement_C>

End_If;
```

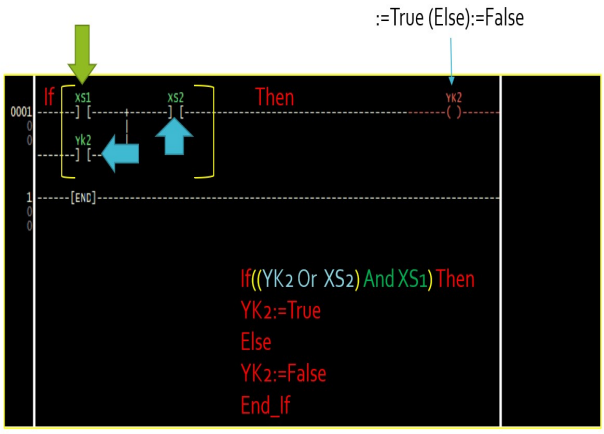


Introduction to structured programming

PLC(ladder to Code)-input to output

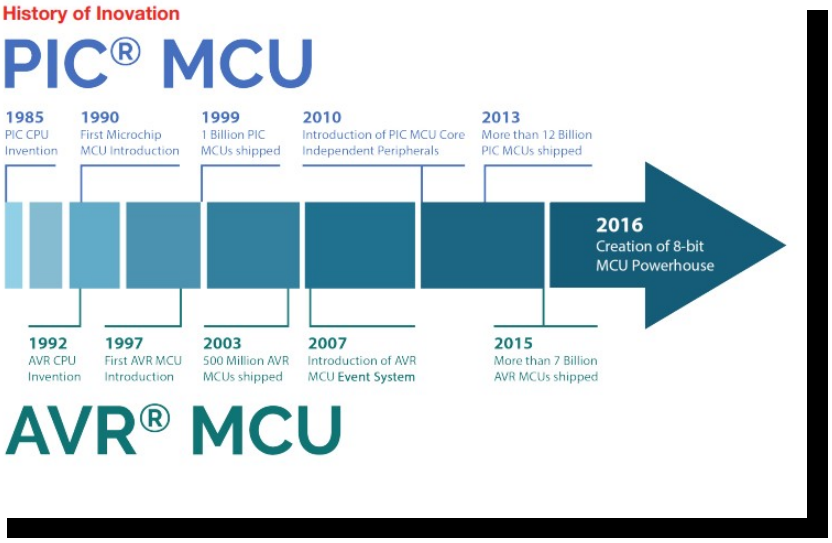


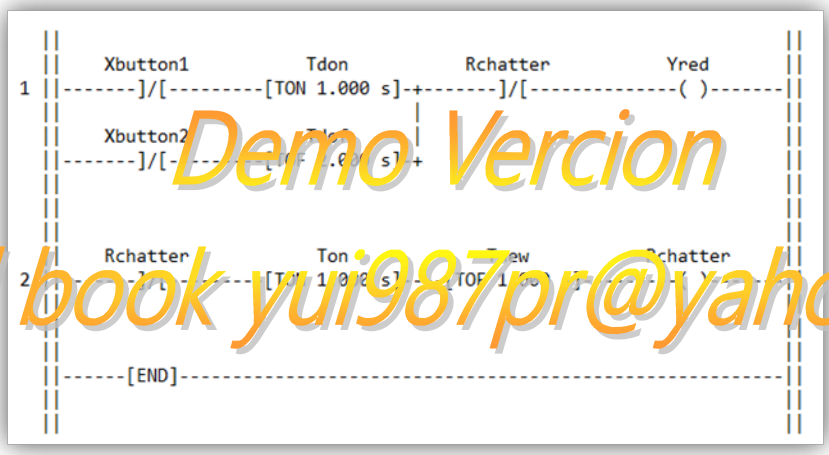
PLC-latching relay



INTRODUCTION:

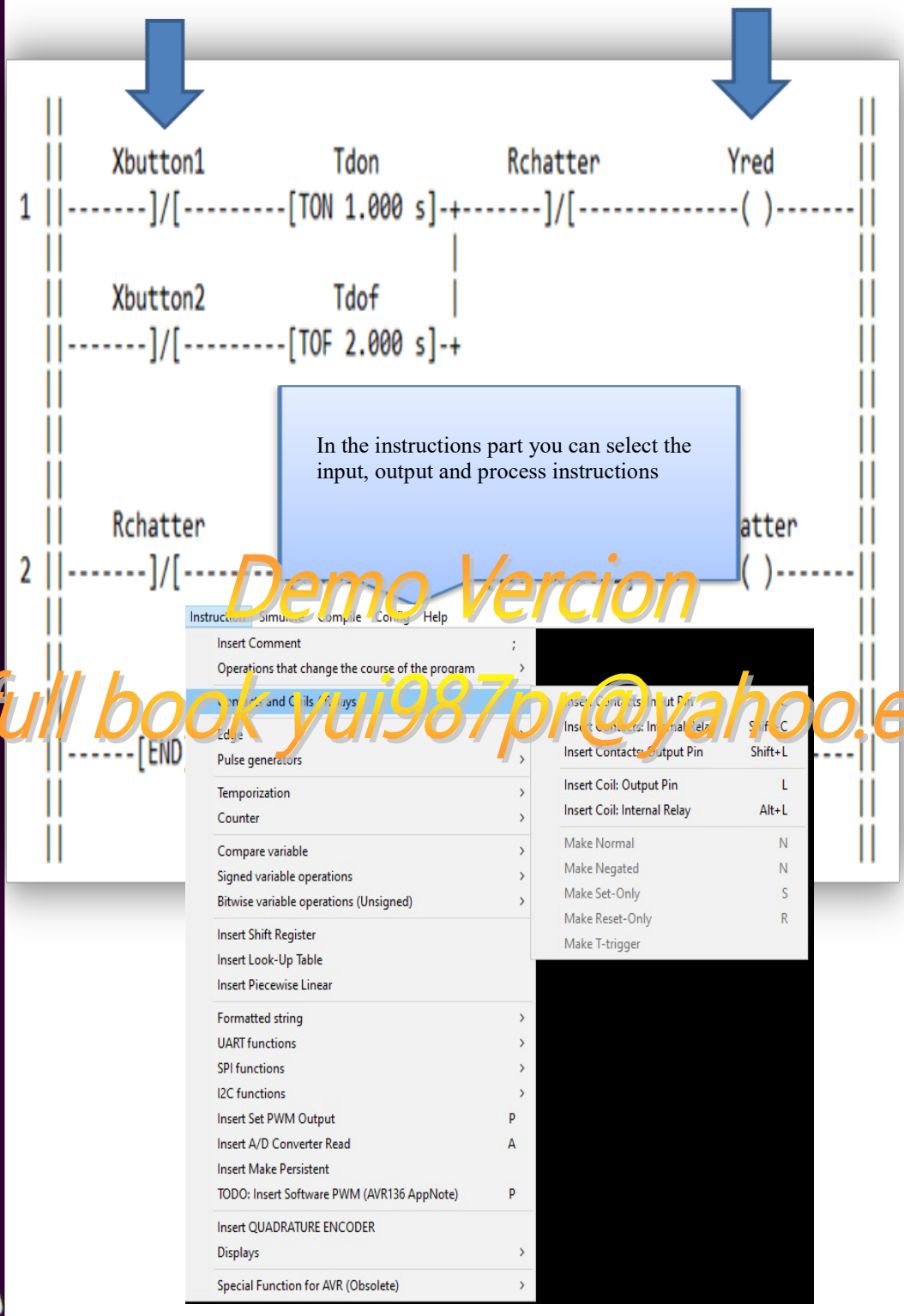
LDmicro generates native code for certain Microchip PIC16 and Atmel AVR
microcontrollers. Usually software for these microcontrollers is written
in a programming language like assembler, C, or BASIC. A program in one
of these languages comprises a list of statements. These languages are



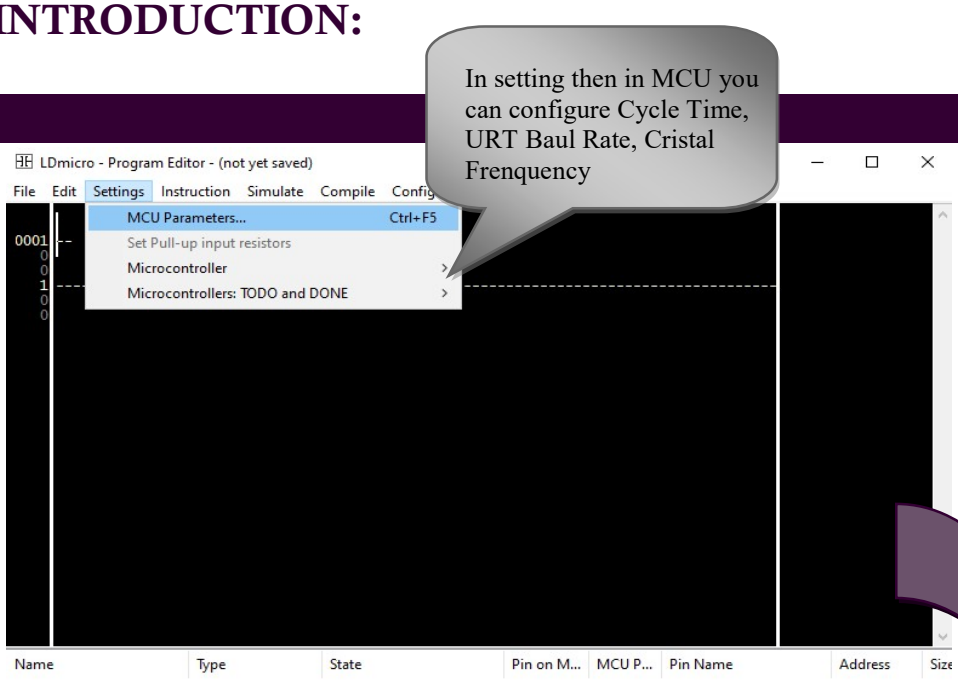


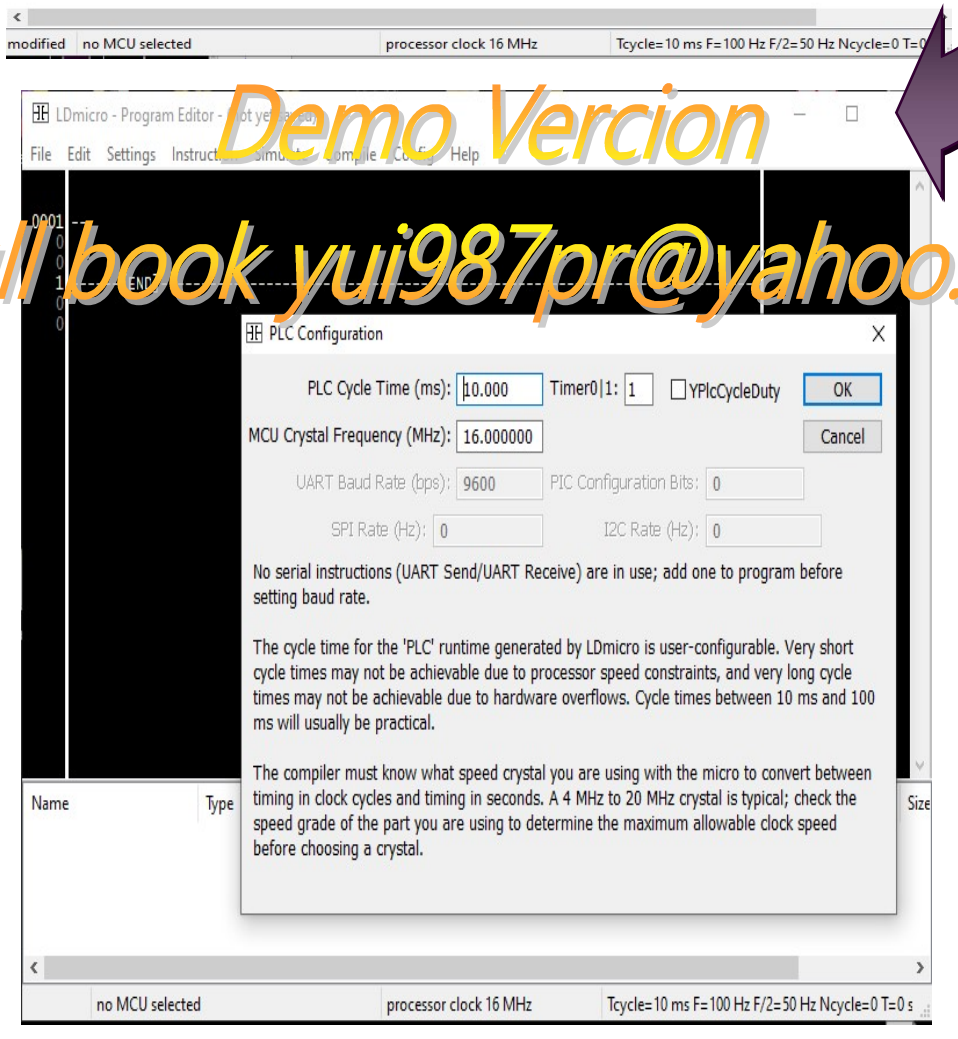
(TON is a turn-on delay; TOF is a turn-off delay. The --] [-- statements are inputs, which behave sort of like the contacts on a relay. The --( )-- statements are outputs, which behave sort of like the coil of a relay. Many good references for ladder logic are available on the Internet and elsewhere; details specific to this implementation are given below.)

INTRODUCTION:



INTRODUCTION:





INTRODUCTION:

This screen we will see how to select our microcontroller

(TON is a turn-on delay; TOF is a turn-off delay. The --] [-- statements are inputs, which behave sort of like the contacts on a relay. The --( )-- statements are outputs, which behave sort of like the coil of a relay. Many good references for ladder logic are available on the Internet and elsewhere; details specific to this implementation are given below.)

A number of differences are apparent:

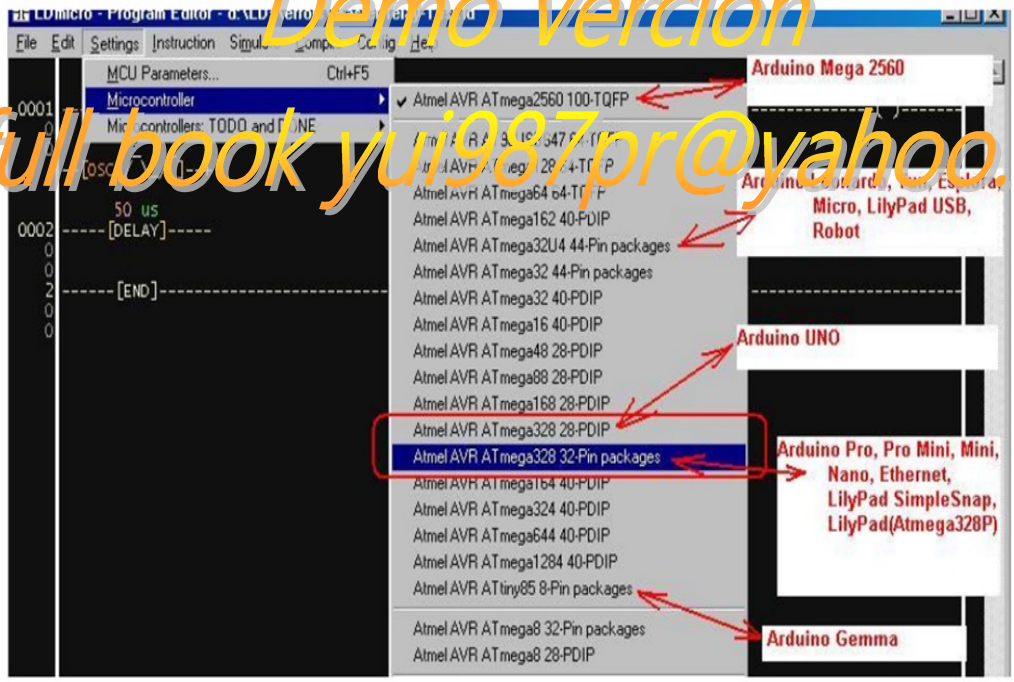
* The program is presented in graphical format, not as a textual list of statements. Many people will initially find this easier to understand.

* At the most basic level, programs look like circuit diagrams, with relay contacts (inputs) and coils (outputs). This is intuitive to programmers with knowledge of electric circuit theory.

* The ladder logic compiler takes care of what gets calculated where. You do not have to write code to determine when the outputs have to get recalculated based on a change in the inputs or a timer event, and you do not have to specify the order in which these calculations must take place; the PLC tools do that for you.

INTRODUCTION:

Flashing Steps:
Download xLoader.zip file.
Virus Check the xLoader.zip file
Unzipthe xLoader.zip file
Connect IP01 and CC01/CR0x together
Insert IP01 and CC01/CR0x into an available USB port
Wait for eventual drivers to be installed, if driver installation fail, goto USB Driver
Executethe xLoader.exe file
Choose Firmware. There is a .hex file included in the install folder. You can alsodownload a specific radio ID pairs.
Choose your COM port. If no COM port is available, goto USB Driver
Click Upload
Wait for the text Uploading... to be replaced by <nnnn> bytes uploaded
Unplug IP01 and CC01/CR0x
USB Driver
If you have issues with USB drivers for the IP01, then go here IP01, and install the driver from the drivers folder.



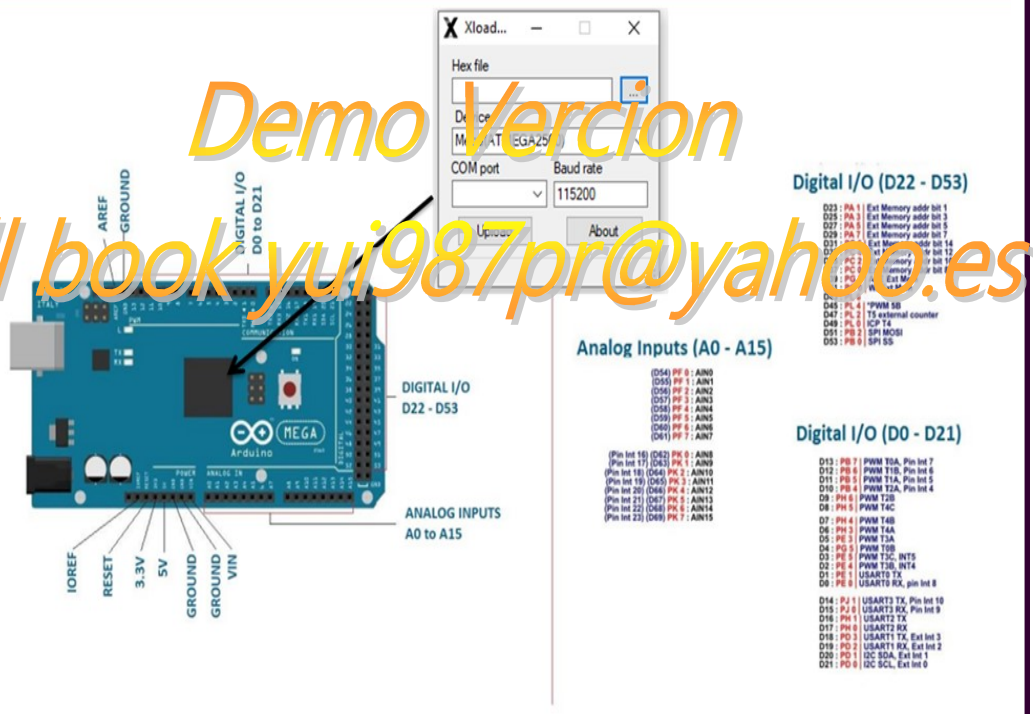
Program download to PLC

Flashing on Non-Windows Operating Systems

These instructions are provided
IS. If you encounter difficulty,
the instructions above before
porting an issue.

AS
use
re-

You will need to install the `avrdude` command. On several linux systems this is

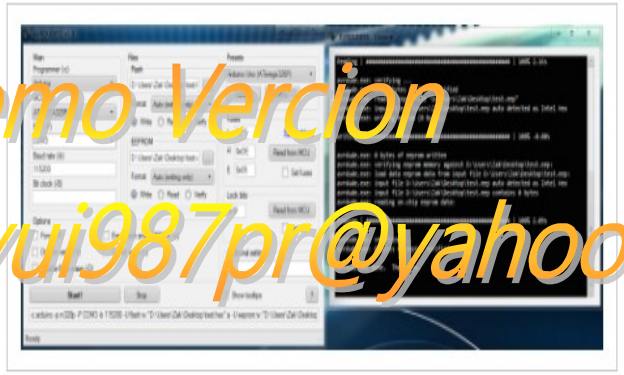


Arduino Mega 2560 Pin Description

Program download to PLC

GUI for old good AVRDUDE

AVRDude is an open source command line tool used to program AVR microcontrollers. It's been integrated into many handy tools like Arduino IDE, MPide and more. This is where we usually don't see much of it as programming is performed automatically. AVRDude is great software because it supports most of AVR programmer adapters including majority of DIY and most of commercial ones. The only downside of it that you may need use console and write long commands in order to program or read chips.

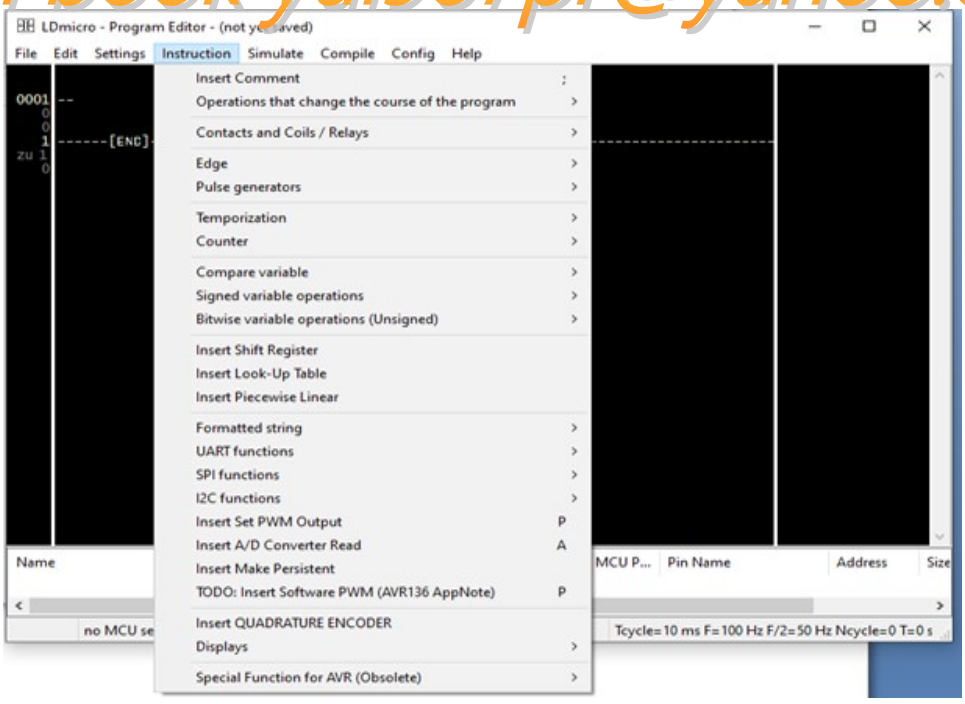


If you don't feel comfortable with using command line tool, then probably the best option is find a good IDE which would wrap it with nice and intuitive graphical interface. So you could with few button clicks load binary in to AVR microcontroller. You can find many AVRDude GUI's around the Internet. Some of them are obsolete, other are dedicated to one type of programmer adapter. If you are looking for universal GUI take a look at AVRDUDESS which is pretty intuitive and universal. In order to run it you will need at least .NET Framework 4.0 version. Among features we can mention that it supports most of programmer adapters. You can drag and drop files for faster uploading, easy choose parameters with drop down lists. Give it a try and tell what you think.

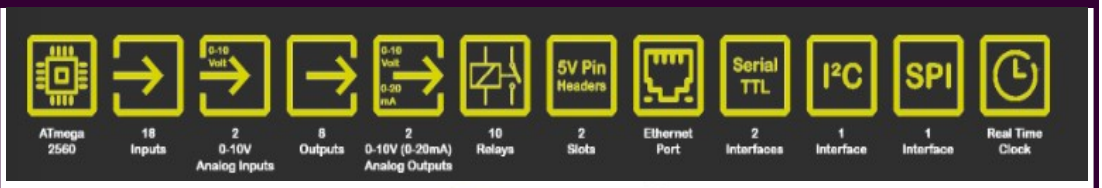
Program download to PLC

INTRODUCTION:

BASICS:
If you run LDmicro with no arguments then it starts with an empty program. If you run LDmicro with the name of a ladder program (xxx.ld)
on the command line then it will try to load that program at startup.
LDmicro uses its own internal format for the program; it cannot import
logic from any other tool.
If you did not load an existing program then you will be given a program
with one empty rung. You could add an instruction to it; for example
you could add a set of contacts (Instruction -> Insert Contacts) named
`Xnew'. `X' means that the contacts will be tied to an input pin on the microcontroller. You could assign a pin to it later, after choosing a



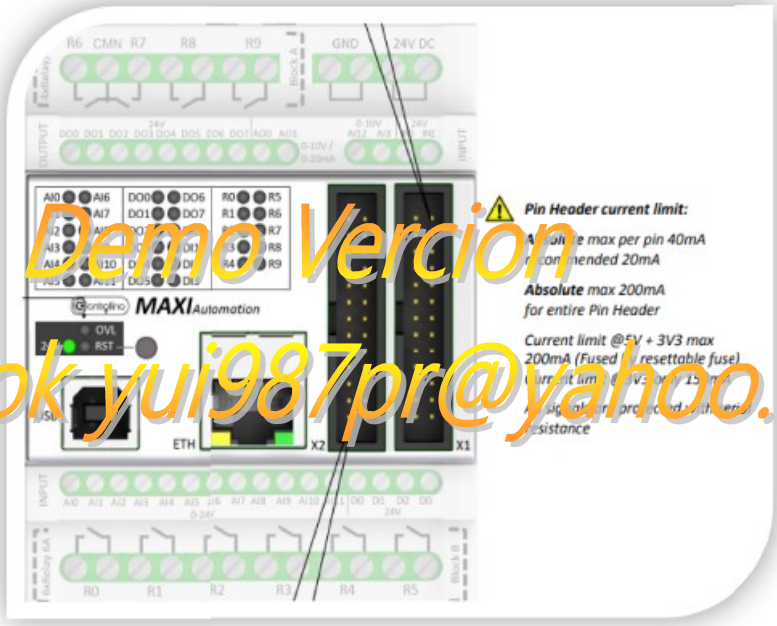
PLC Controllino



Controllino

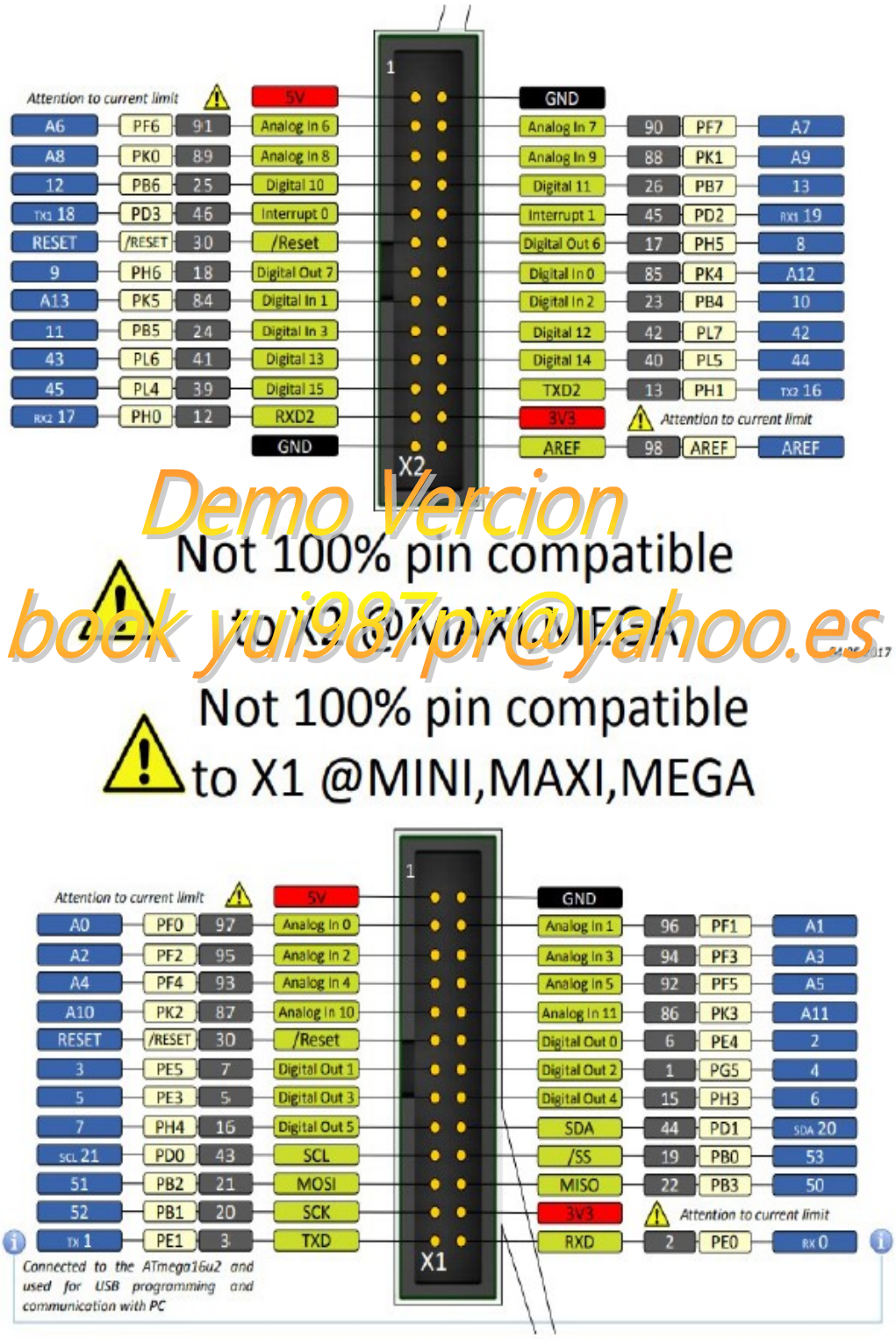
MAXI Automation

PINOUT V1.1

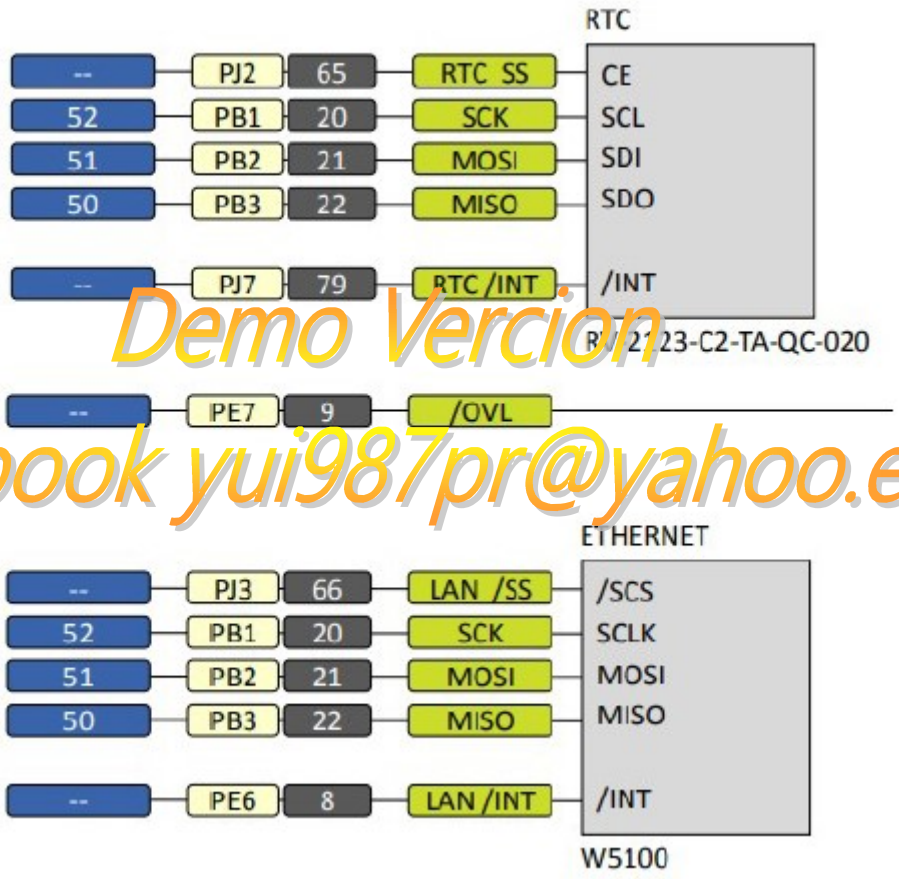


PLC Controllino

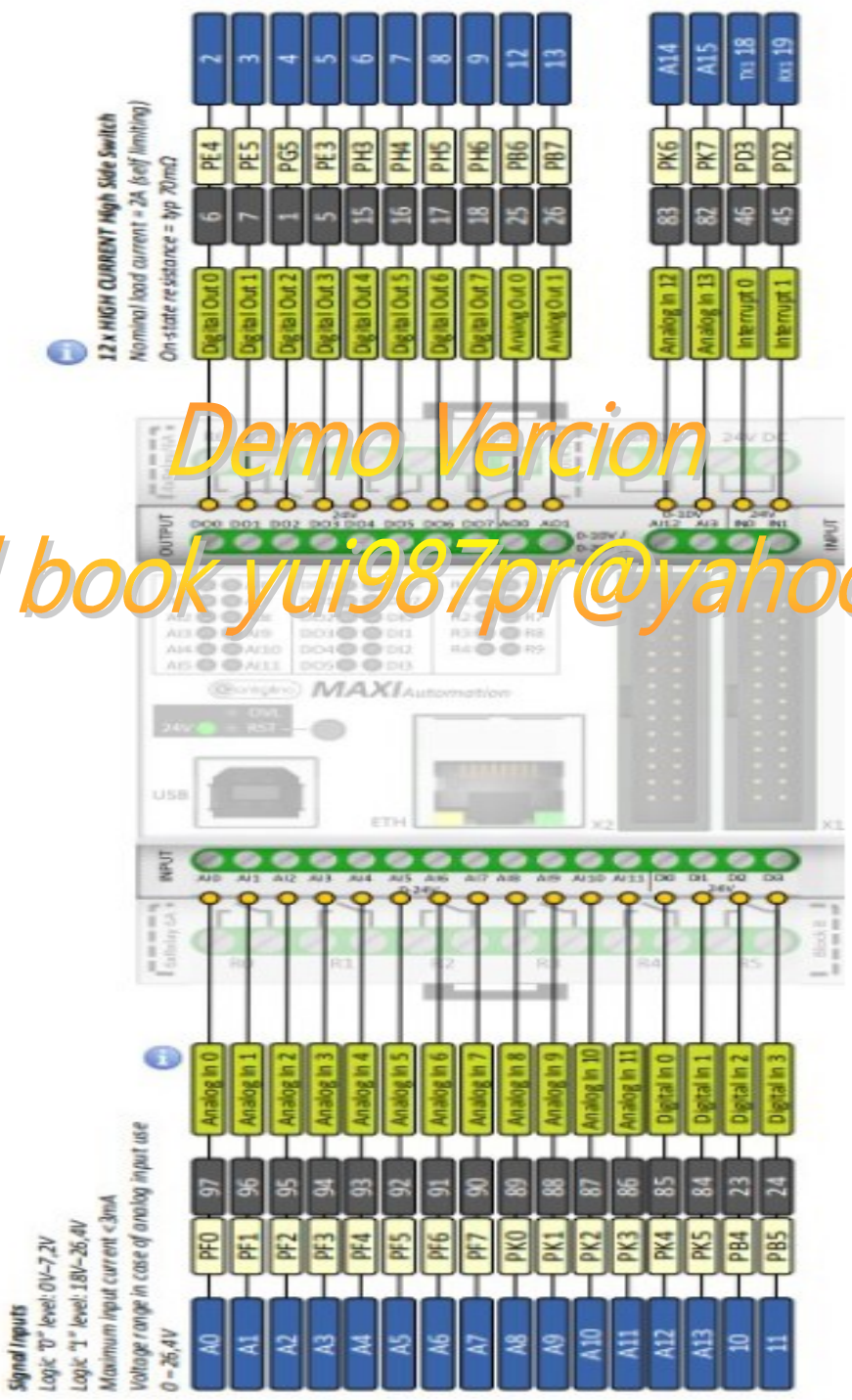
The MAXI Automation is the version of Controllino MAXI specifically tailored for automation specialists that need 0-10V I/Os! It is the perfect compromise between compact size and big input and output quantity. The core competence is its flexibility. Pinheader protection caps included!



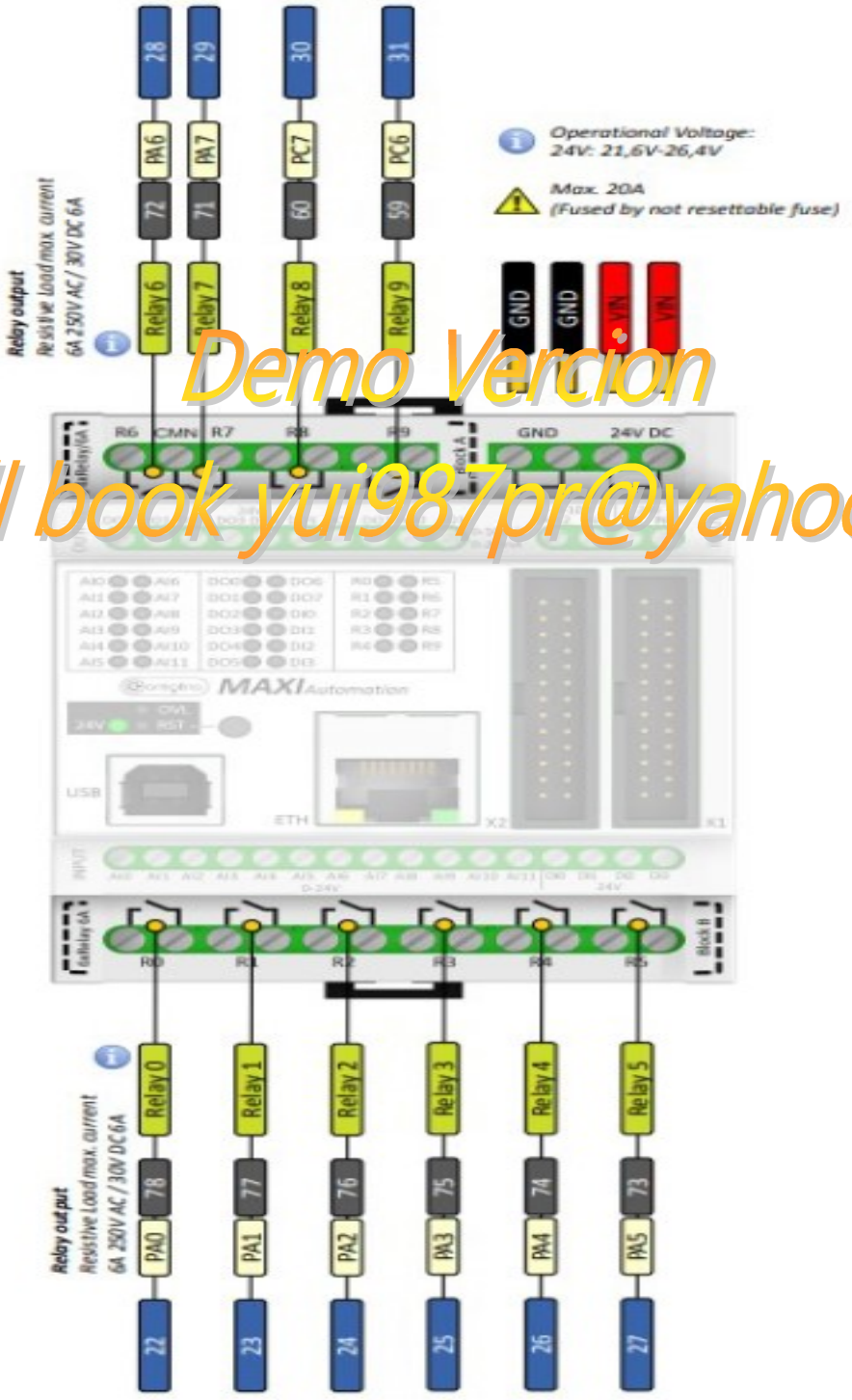
PLC Controllino



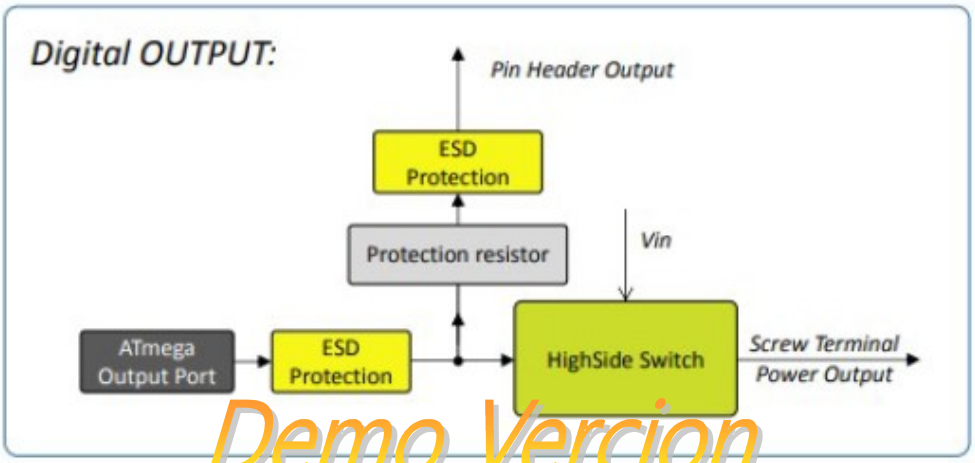
PLC Controllino

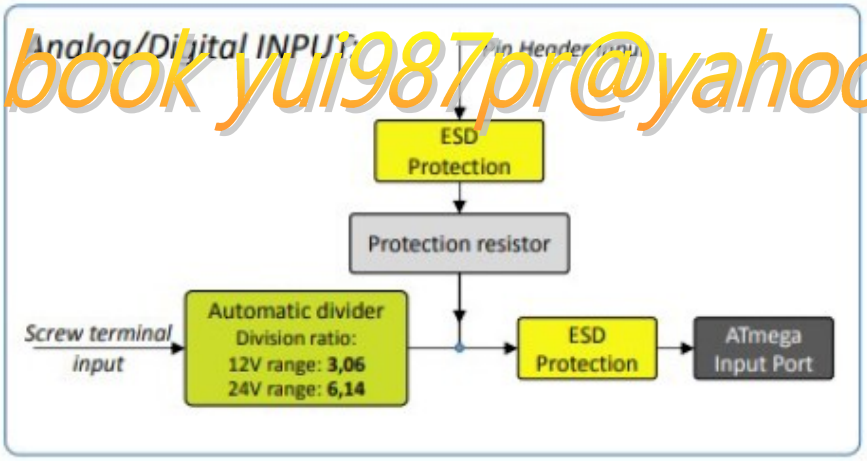


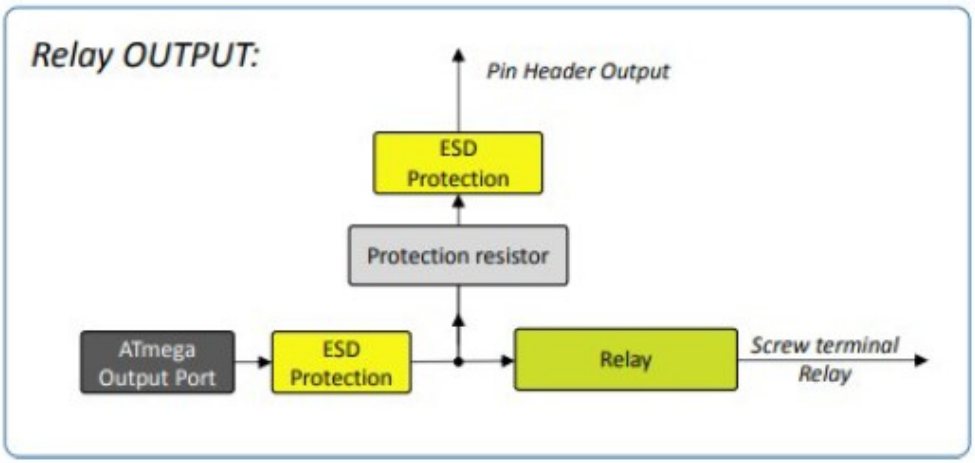
Program download to PLC



Program download to PLC







Variable

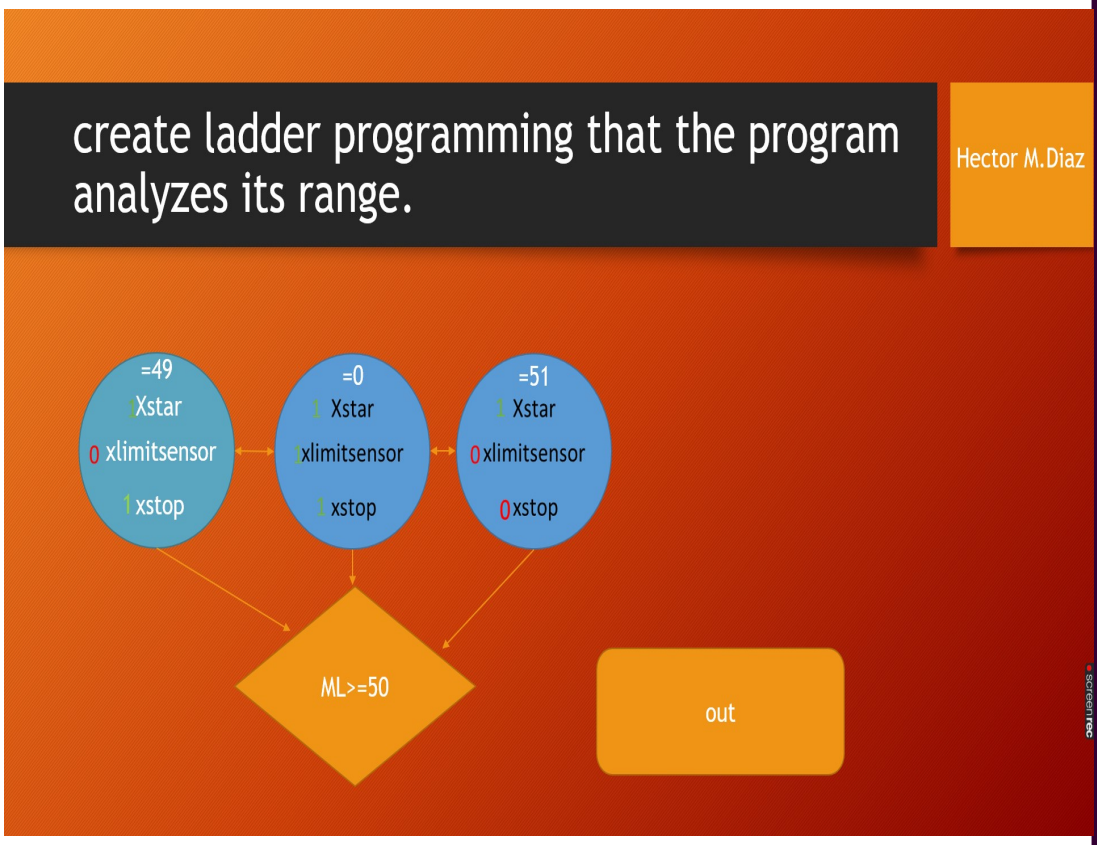
Variable

Variable name that begins with symbol '#' like
#PORTA, #PORTB, #PORTC, ... treated as output hardware port.
Variable name that begins with symbol '#' like
#PINA, #PINB, #PINC, ... treated as input hardware port.
Variable name that begins with symbol '#' like
#TRISA, #TRISB, #TRISC, ... treated as data direction register of corresponding ports
#PORTA, #PORTB, #PORTC, ...
Note: Address PORTn and PINn are equals for PIC's.
The variable name that begins with a '#' character and subsequent a number
(commonly a hexadecimal) treated as immediate address of hardware register.
Value like #0xXXXX is immediate, direct, explicit addressing.
For example:
#0x8E treated as the address of the hardware PCON register
of the microcontroller Microchip PIC16F628.
#0x136 treated as the address of the hardware UDR3 register
(USART3 I/O Data Register) of the microcontroller Atmel AT-mega2560.
The variable name that begins with a '#' character and subsequent as a general-purpose variable name treated as the indirect address of hardware register.

For example:

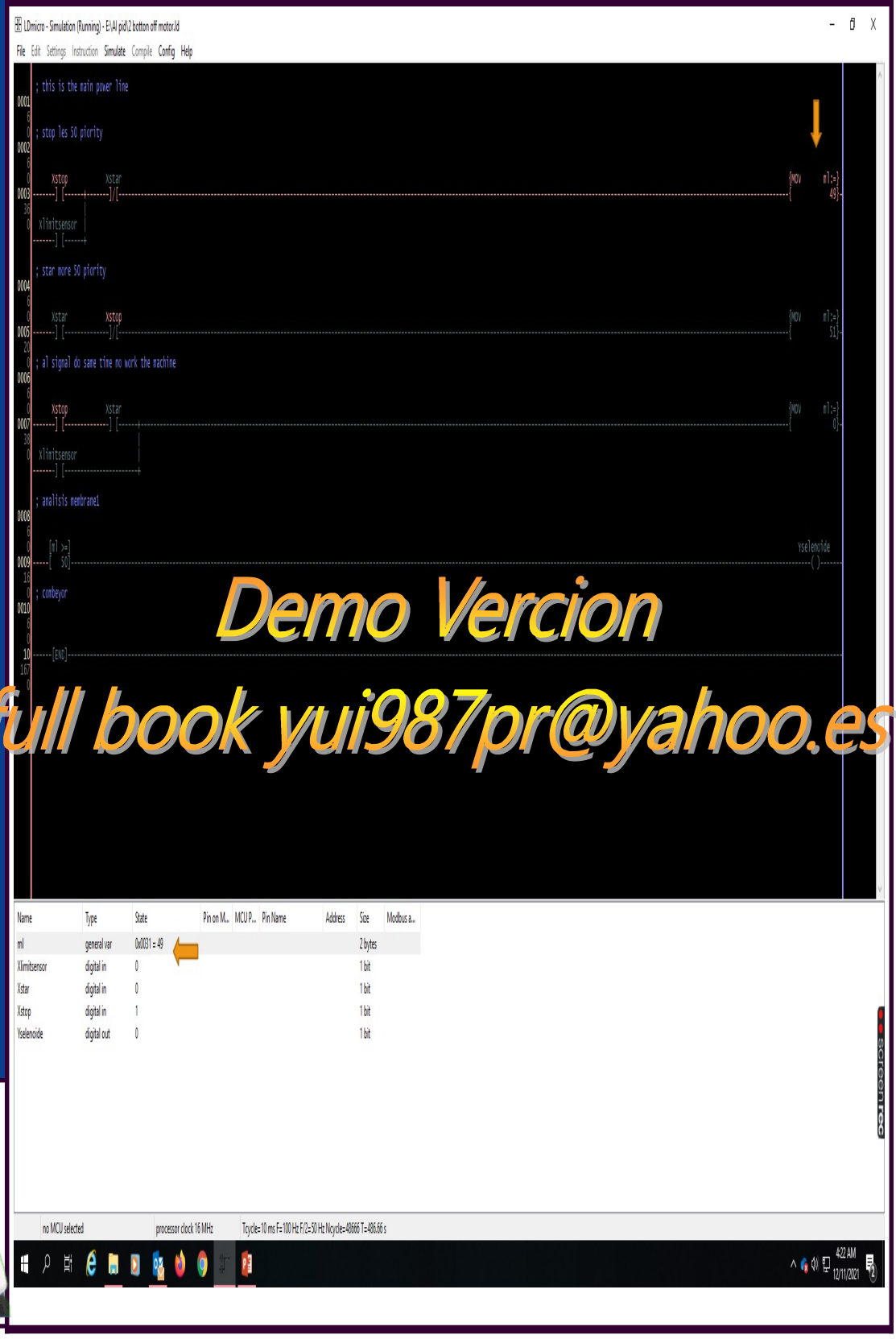
```
  {MOV  portAddr:=}   portAddr is a general-purpose variable containing the value 0x05.
--{          0x05}--

  {MOV #portAddr:=}    #portAddr treated as the indirect address of the register #0x05 PORTA (Microchip PIC16F887).
--{          0xF0}--  Output pins PORTA are set to a binary value of 11110000.

  {MOD portAddr:=}    The general-purpose portAddr variable now a value of 0x06
--{portAddr  +   1}--

  {MOV #portAddr:=}    #portAddr treated as the indirect address of the register #0x06 PORTB (Microchip PIC16F887).
--{          0x0F}--  Output pins PORTB are set to a binary value of 00001111.
```


Variable

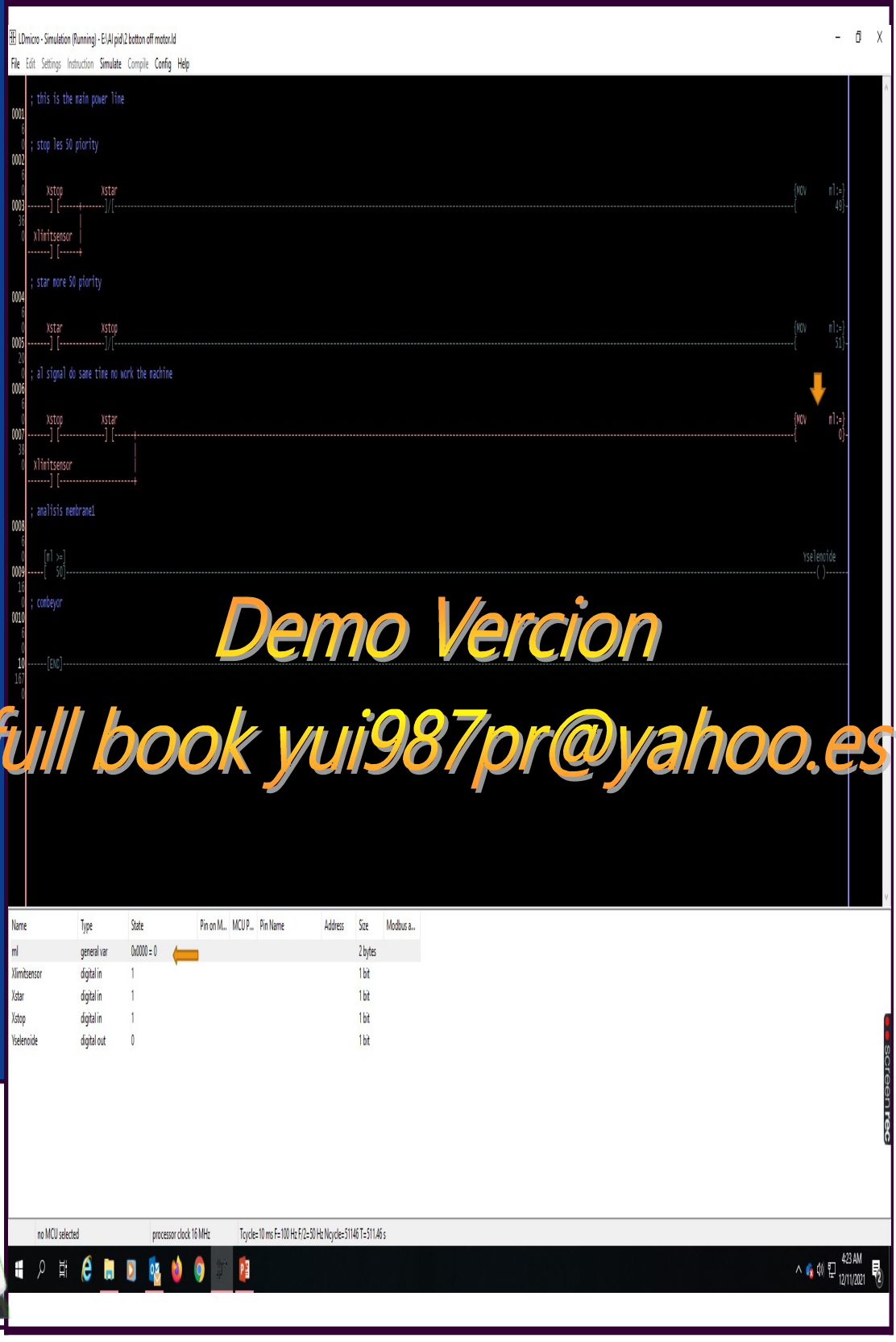


If the digital input that has a value enters the logic of greater than 50, it will have an output response. Next, you will see in the ladder

Variable


LDmicro - Simulation (Running) - E:\AI pid\2 bottom off motor.ld
File Edit Settings Instruction Simulate Compile Config Help
; this is the main power line
; stop les 50 piority
Xstop
Xstar
Xlimitsensor
; star more 50 piority
; al signal do same time no work the machine
; analisis membrane1
; conbeyor
[END]
Demo Vercion
full book yui987pr@yahoo.es
Name Type State Pin on M... MCU P... Pin Name Address Size Modbus a...
ml general var 0x0031 = 49 2 bytes
Xlimitsensor digital in 0 1 bit
Xstar digital in 0 1 bit
Xstop digital in 1 1 bit
Yselenoide digital out 0 1 bit
no MCU selected
processor clock 16 MHz
Tcycle=10 ms F=100 Hz F/2=50 Hz Ncycle=48666 T=486.66 s
4:22 AM
12/11/2021

Variable

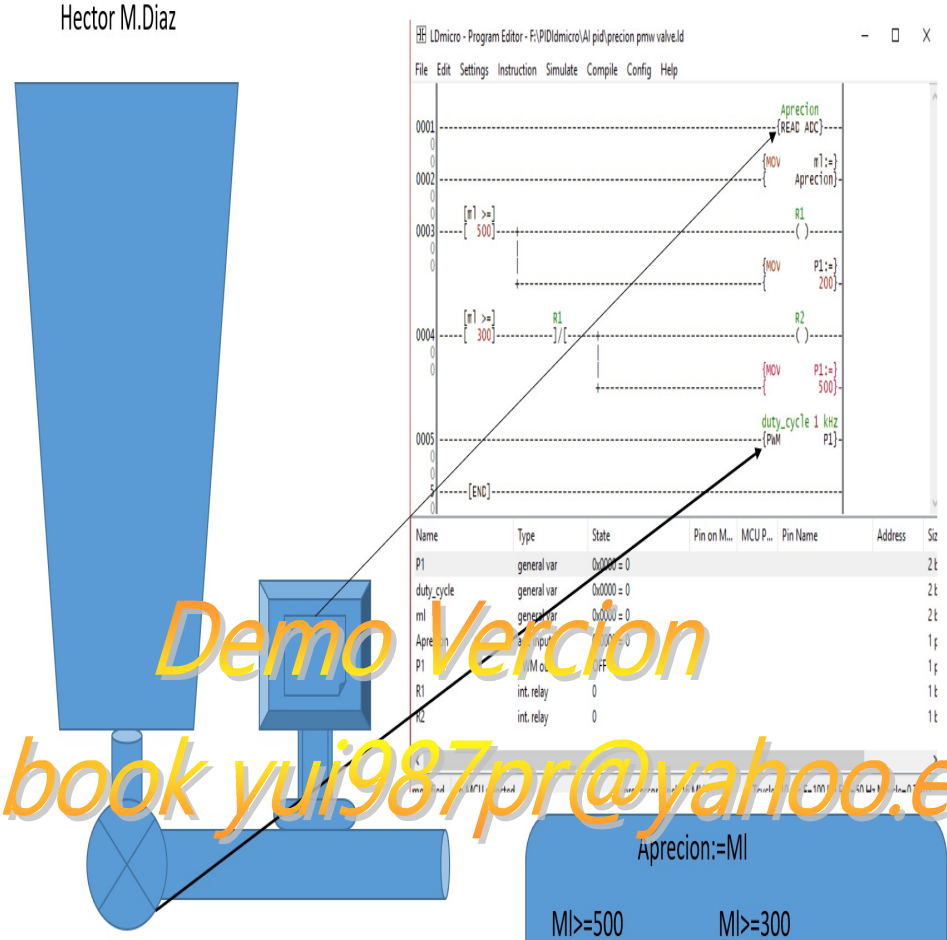

LDmicro - Simulation (Running) - EVAI pid12 botton off motor.ld
File Edit Settings Instruction Simulate Compile Config Help
; this is the main power line
; stop les 50 piority
Xstop
Xstar
Xlimitsensor
MOV
ml := 49
; star more 50 piority
Xstar
Xstop
ml := 51
; al signal do same time no work the machine
Xstop
Xstar
Xlimitsensor
ml := 0
; analisis membranel
ml >= 50
Yselenoide
; conbeyor
END
Demo Vercion
full book yui987pr@yahoo.es
Name Type State Pin on M... MCU P... Pin Name Address Size Modbus a...
ml general var 0x0000 = 0 2 bytes
Xlimitsensor digital in 1 1 bit
Xstar digital in 1 1 bit
Xstop digital in 1 1 bit
Yselenoide digital out 0 1 bit
no MCU selected
processor clock 16 MHz
Tcycle=10 ms F=100 Hz F/2=50 Hz Ncycle=51146 T=511.46 s
4:23 AM
12/11/2021

Variable


LDmicro - Simulation (Running) - El AI pid 2 botton off motor.ld
File Edit Settings Instruction Simulate Compile Config Help
; this is the main power line
; stop les 50 piority
Xstop
Xstar
Xlimitsensor
mov
ml:=
49
; star more 50 piority
Xstar
Xstop
51
; al signal do same tine no work the machine
0
; analisis membrane1
ml >=
50
Yselenoide
; conbeyor
END
Demo Vercion
full book yui987pr@yahoo.es
Name
Type
State
Pin on M...
MCU P...
Pin Name
Address
Size
Modbus a...
ml
general var
0x0033 = 51
2 bytes
Xlimitsensor
digital in
0
1 bit
Xstar
digital in
1
1 bit
Xstop
digital in
0
1 bit
Yselenoide
digital out
1
1 bit
no MCU selected
processor clock 16 MHz
Tcycle= 10 ms F= 100 Hz F/2= 50 Hz Ncycle= 55533 T= 555.33 s
4:24 AM
12/11/2021

Variable

Hector M.Diaz



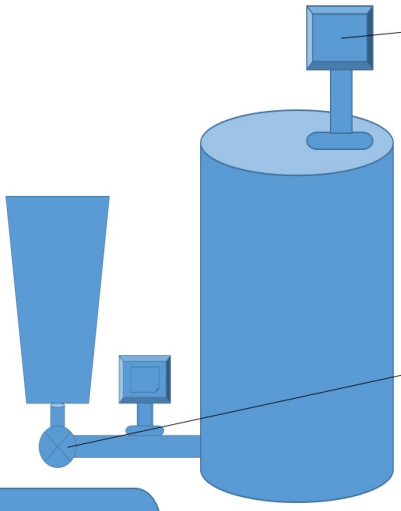
Aprecion:=Ml

Ml>=500
R2: Out
MOV 200:=P1

Ml>=300
R1:NC and R2Out
MOV 300:=P1

Variable

Hector M.Diaz



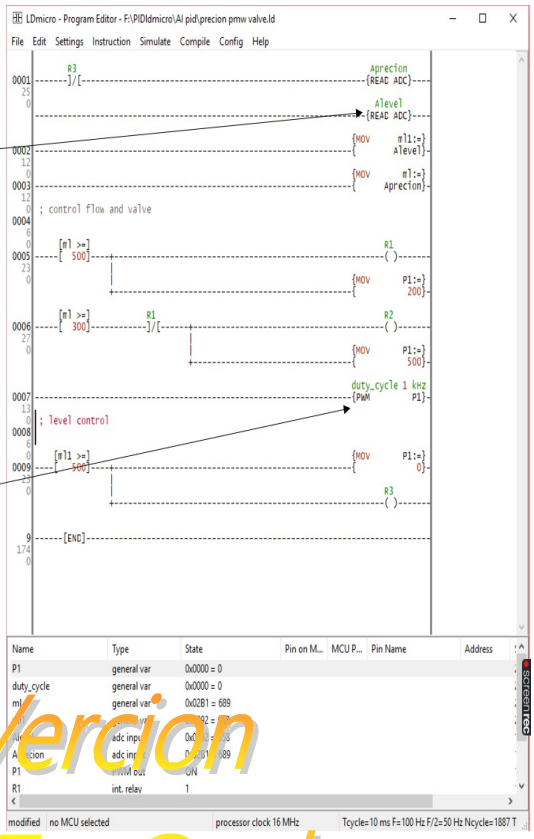
Alevel:MOV ML1

ML1>=500

MOV 0 :=P1

R3: Out

R3:NC In



complete book includes 237 pages and more than 70 examples of programming in LD micro PLC