



# 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

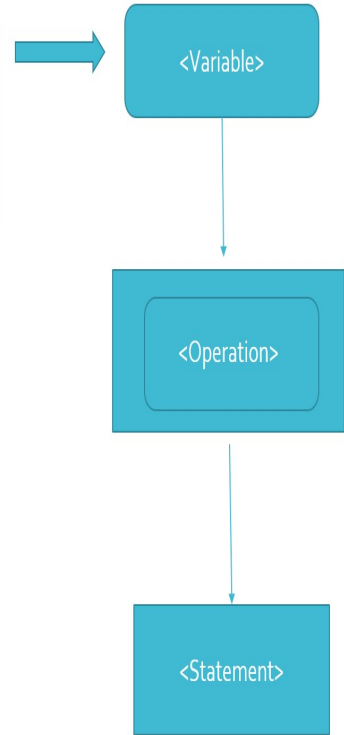
Data Type	Minimum Value	Maximum Value
Integer	-32,768	32,767
LongInt	-2,147,483,648	2,147,487,647
ShortInt	-128	128
Real	2.9 x 10 E-39	1.7 x 10 E+38

Var <Name>:bool;

Begin  
<Name>:=<Operation>;

<statement>:= <Name>;

End;



*Demo Version*

*full book yui987pr@yahoo.es*



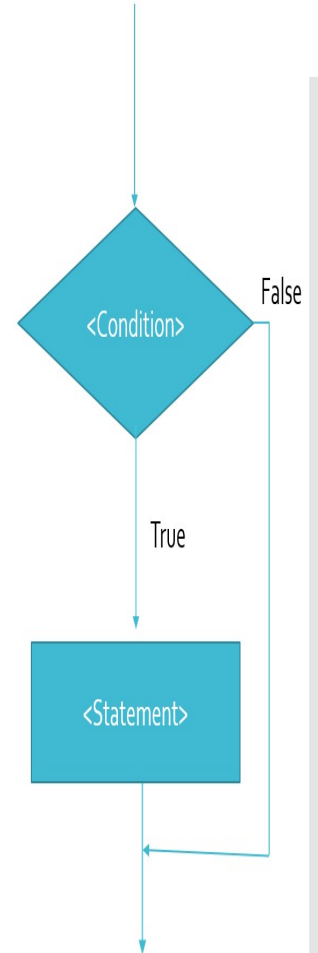
## Introduction to structured programming

Basic-  
(IF-Then)

If <Condition> Then  
<Statement>  
End\_If;

Have the code  
or the operation.

Reaction of the code

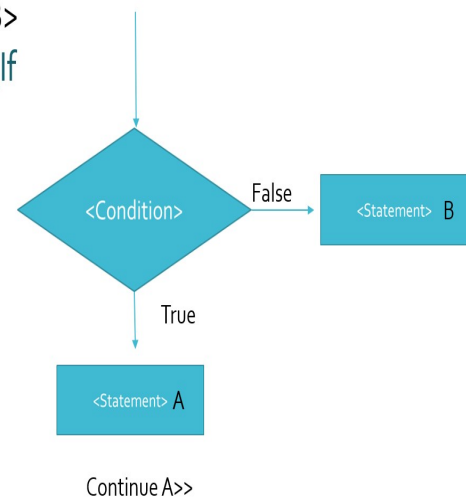


*Demo Version*  
*full book yui987pr@yahoo.es*

## Introduction to structured programming

### Basic- Else

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



*Demo Version*

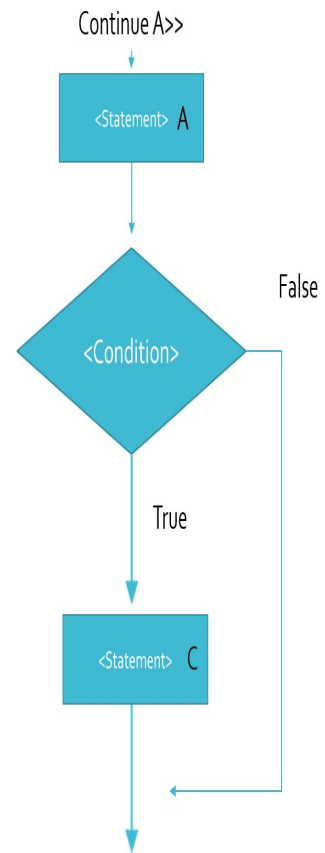
*full book yui987pr@yahoo.es*



## Introduction to structured programming

Basic-Mode  
operator

```
<Stament_A>  
If <Condition> Then  
<statement_C>  
End_If;
```



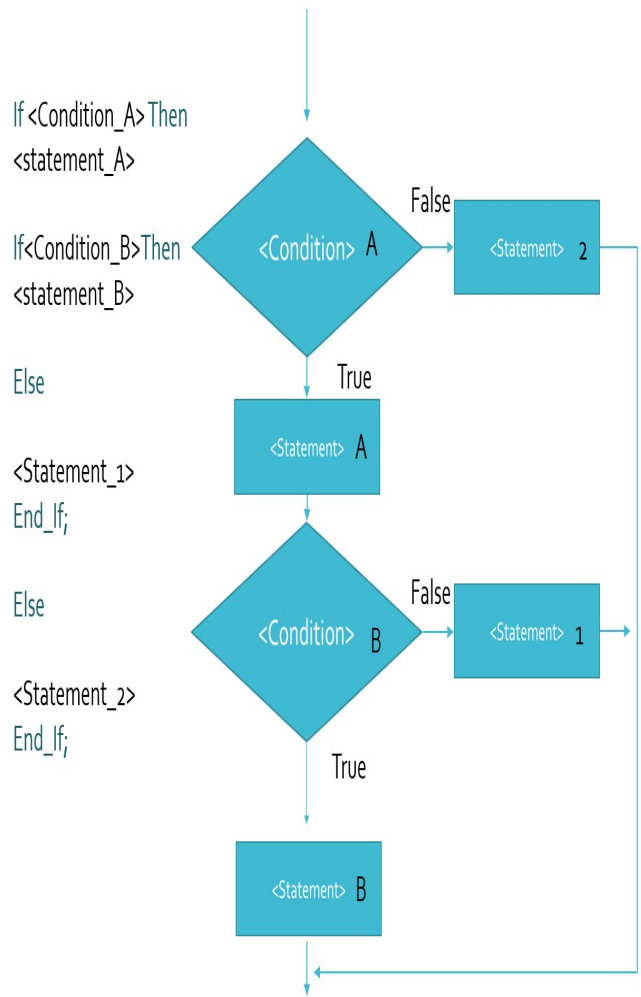
*Demo Vercion*

*full book yui987pr@yahoo.es*



# Introduction to structured programming

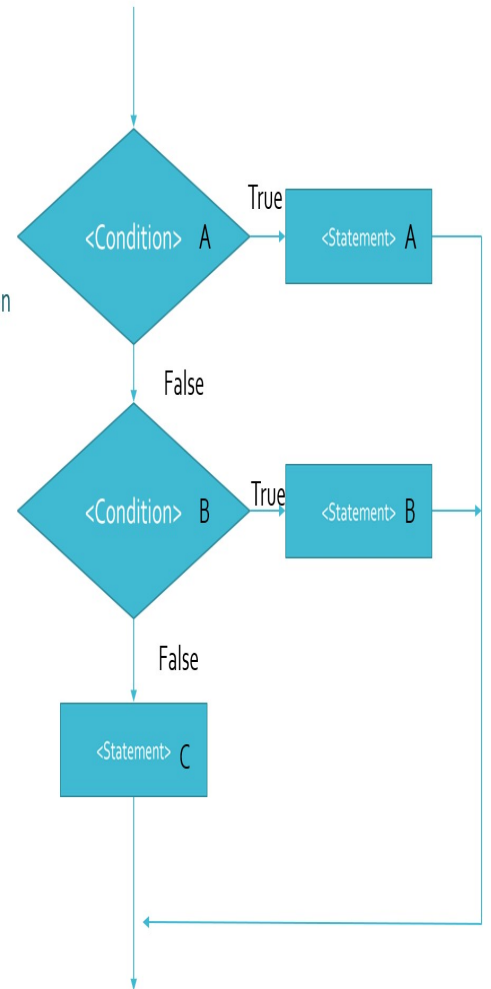
Basic-mode  
operation  
follow (Else)



# Introduction to structured programming

## Basic- Elsif

```
If<Condition_A>Then  
<Statement_A>  
Elsif <Condition_B>Then  
<Statement_B>  
Else  
<Statement_C>  
End_If;
```



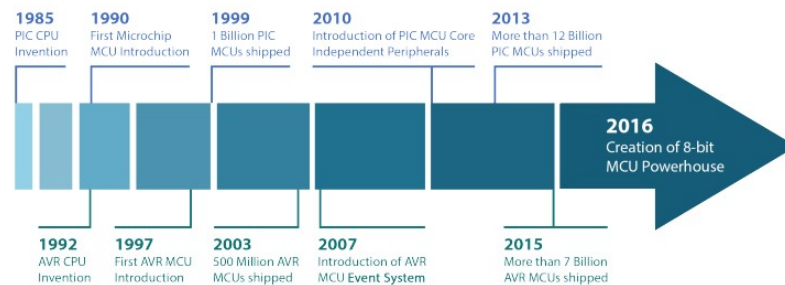


## INTRODUCTION:

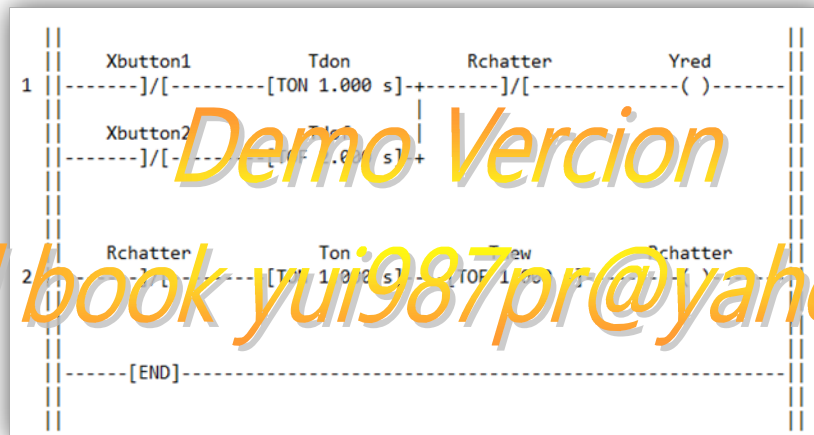
LDmicro generates native code for certain Microchip PIC16 and Atmel AVR microcontrollers. Usually software for these microcontrollers is written in a programming language like assembler, C, or BASIC. A program in one of these languages comprises a list of statements. These languages are

### History of Innovation

## PIC<sup>®</sup> MCU



## AVR<sup>®</sup> MCU

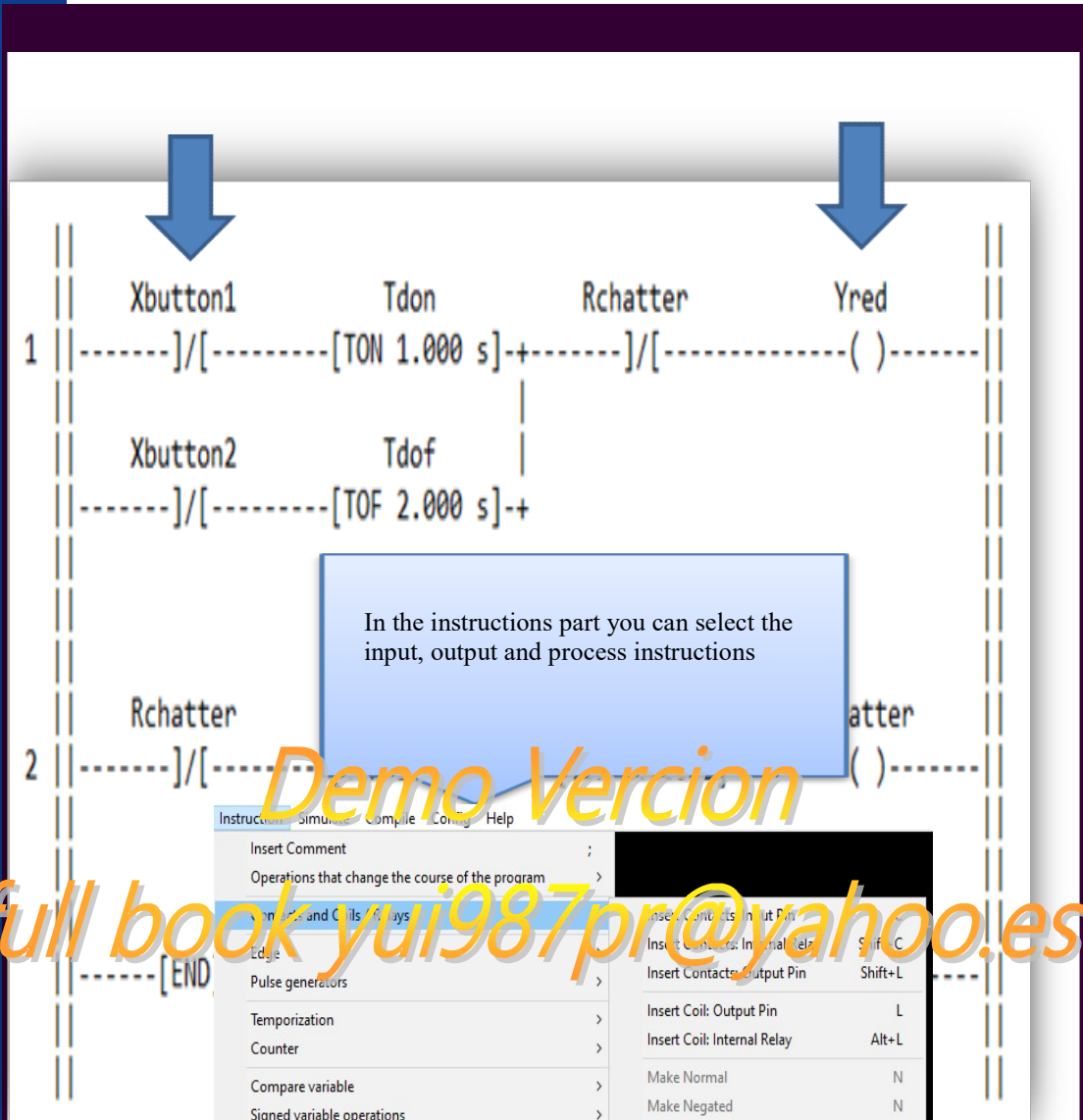


*Demo Version*  
*full book yui987pr@yahoo.es*

(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:



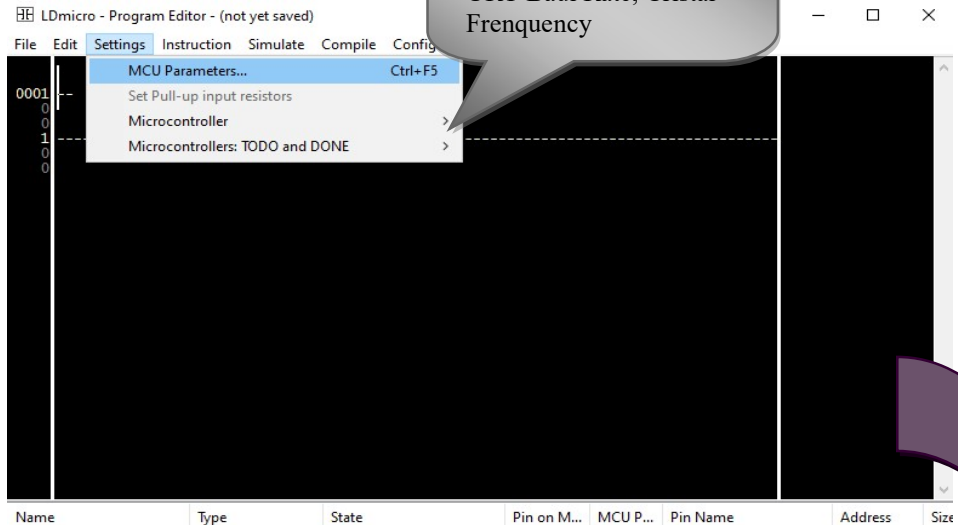
*full book yui987pr@yahoo.es*

- Instructions
- Simulate
- Compile
- Configure
- Help
- Insert Comment
- Operations that change the course of the program
- Insert Contacts: Input Pin
- Insert Contacts: Internal Relay
- Insert Contacts: Output Pin
- Insert Coil: Output Pin
- Insert Coil: Internal Relay
- Make Normal
- Make Negated
- Make Set-Only
- Make Reset-Only
- Make T-trigger
- Formatted string
- UART functions
- SPI functions
- I2C functions
- Insert Set PWM Output
- Insert A/D Converter Read
- Insert Make Persistent
- TODO: Insert Software PWM (AVR136 AppNote)
- Insert QUADRATURE ENCODER
- Displays
- Special Function for AVR (Obsolete)



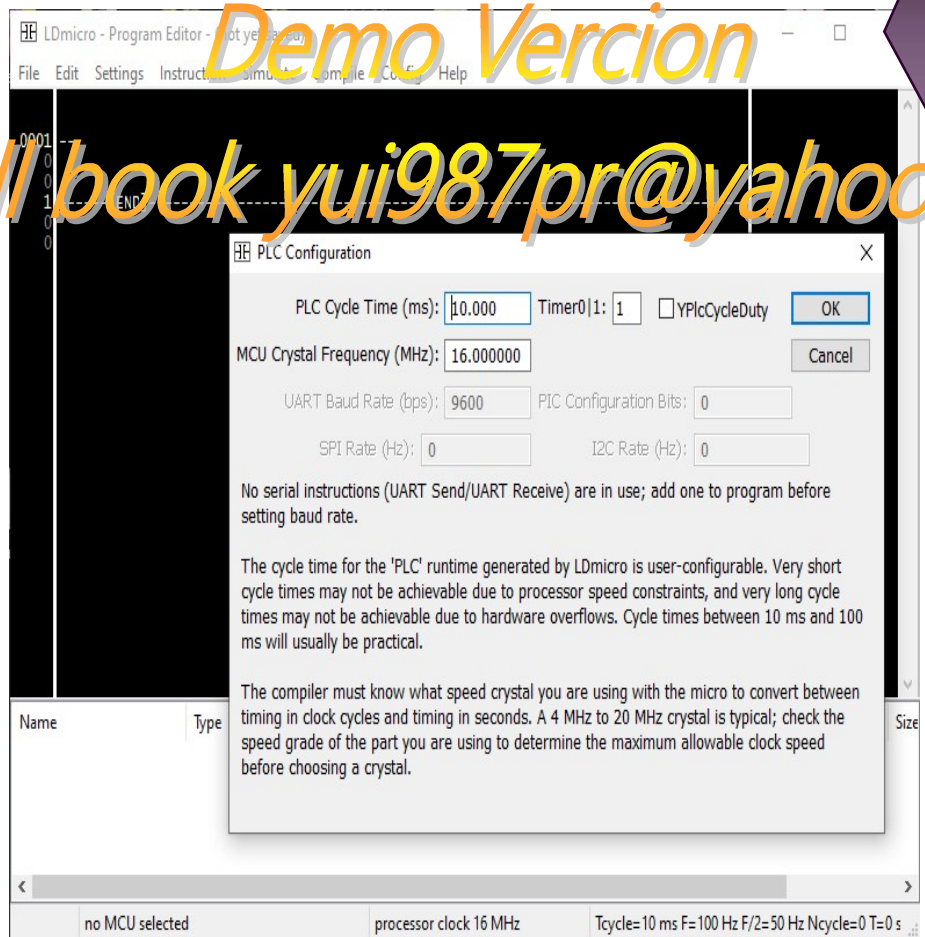
## INTRODUCTION:

In setting then in MCU you can configure Cycle Time, URT Baul Rate, Cristal Frequency



modified no MCU selected processor clock 16 MHz Tcycle=10 ms F=100 Hz F/2=50 Hz Ncycle=0 T=0

*Demo Vercion*  
*full book yui987pr@yahoo.es*



## 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

Unzip the 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

Execute the xLoader.exe file

Choose Firmware. There is a .hex file included in the install folder. You can also download 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 reporting an issue.

AS  
use  
re-

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

*Demo Vercion*  
*full book yui987pr@yahoo.es*

**Digital I/O (D22 - D53)**

- D23: PA 1 Ext Memory addr bit 1
- D25: PA 3 Ext Memory addr bit 3
- D27: PA 5 Ext Memory addr bit 5
- D29: PA 7 Ext Memory addr bit 7
- D31: PA 9 Ext Memory addr bit 9
- D33: PA 11 Ext Memory addr bit 11
- D35: PA 13 Ext Memory addr bit 13
- D37: PA 15 Ext Memory addr bit 15
- D39: PA 17 Ext Memory addr bit 17
- D41: PA 19 Ext Memory addr bit 19
- D43: PA 21 Ext Memory addr bit 21
- D45: PA 23 Ext Memory addr bit 23
- D47: PA 25 Ext Memory addr bit 25
- D49: PA 27 Ext Memory addr bit 27
- D51: PA 29 Ext Memory addr bit 29
- D53: PA 31 Ext Memory addr bit 31

**Analog Inputs (A0 - A15)**

- (Pin Int 16) (D42) PC 0: AIN8
- (Pin Int 17) (D43) PC 1: AIN9
- (Pin Int 18) (D44) PC 2: AIN10
- (Pin Int 19) (D45) PC 3: AIN11
- (Pin Int 20) (D46) PC 4: AIN12
- (Pin Int 21) (D47) PC 5: AIN13
- (Pin Int 22) (D48) PC 6: AIN14
- (Pin Int 23) (D49) PC 7: AIN15

**Digital I/O (D0 - D21)**

- D13: PB 7 PWM T0A, Pin Int 7
- D12: PB 6 PWM T1B, Pin Int 6
- D11: PB 5 PWM T1A, Pin Int 5
- D10: PB 4 PWM T2A, Pin Int 4
- D9: PB 3 PWM T2B
- D8: PB 2 PWM T4C
- D7: PB 1 PWM T4B
- D6: PB 0 PWM T5A
- D5: PB 3 PWM T3A
- D4: PB 2 PWM T0B
- D3: PB 0 PWM T3C, INT5
- D2: PE 4 PWM T3B, INT4
- D1: PE 1 USART3 TX
- D0: PE 8 USART0 RX, pin Int 8
- D14: PJ 1 USART3 TX, Pin Int 10
- D15: PJ 1 USART3 RX, Pin Int 9
- D16: PH 1 USART2 TX
- D17: PH 0 USART2 RX
- D18: PD 0 USART1 TX, Ext Int 3
- D19: PD 2 USART1 RX, Ext Int 2
- D20: PD 1 I2C SDA, Ext Int 1
- D21: PD 0 I2C SCL, Ext Int 0

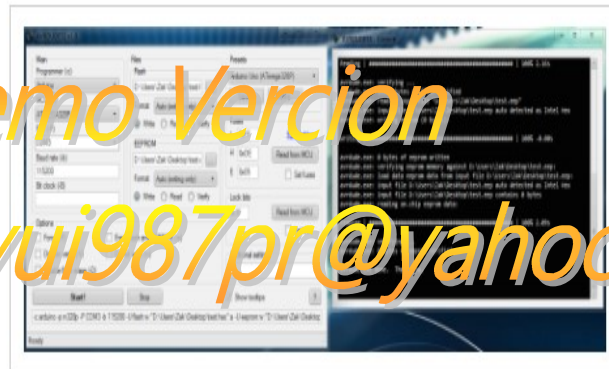
Arduino Mega 2560 Pin Description



## 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.



*Demo Version*  
*full book yui987pr@yahoo.es*

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

AVRDUDESS 2.13 (avrdude version 6.3)

Programmer (c): Wiring  
Port (P): COM4  
Baud rate (b): 115200  
Bit clock (B):  
MCU (p): ATmega2560  
Flash: 256 KB 1E9801  
EEPROM: 4 KB Detect  
Flash: C:\Users\User\Documents\LDMicro\Youtube Tutorial\Tutorial 22\Blink\Blink.hex  
Preset: Arduino Mega (ATmega2560)  
Write Read Verify Go Format Auto (writing only) Manager  
EEPROM: [empty]  
Write Read Verify Go Format Auto (writing only) H 0x08 [empty] Set lock  
Options:  
 Force (F)  Erase flash and EEPROM (-e)  
 Disable verify (-V)  Do not write (-n)  
 Disable flash erase (-D) Verboosity 0  
LB 0x3F Read Write  
 Set lock  
Bit selector  
Program! Stop Options ? Additional command line args  
-c wiring -p m2560 -P COM4 -b 115200 -D -U flash:w:"c:\Users\User\Docum

```
avrdude.exe: input file C:\Users\User\Documents\LDMicro\Youtube Tutorial\Tutorial 22\Blink\Blink.hex  
avrdude.exe: reading on-chip flash data:  
  
Reading | ##### | 100% 0.09s  
  
avrdude.exe: verifying ...  
avrdude.exe: 538 bytes of flash verified  
  
avrdude.exe done. Thank you.
```

CPU

Demo Version  
full book yui987pr@yahoo.es

MEGA 2560



## 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.

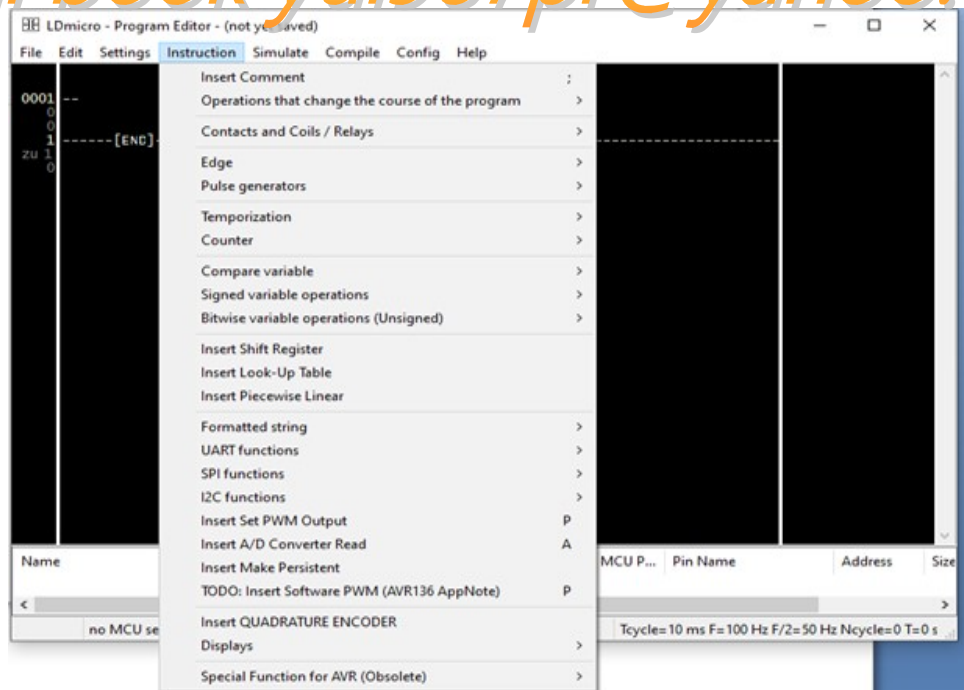
If you did not load an existing program then you will be given a program with one empty rung. You could add an instruction to it; for example

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

*Demo Version*  
*full book yui987pr@yahoo.es*

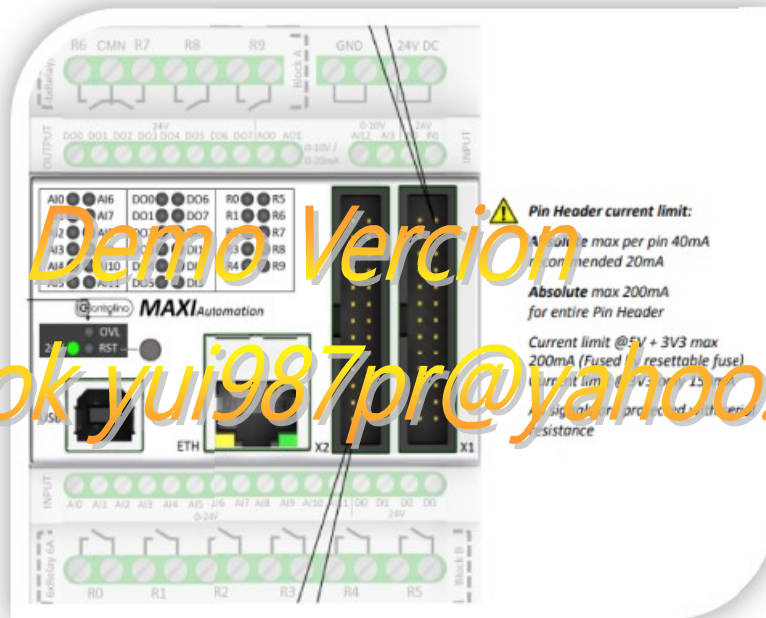


# PLC Controllino

ATmega 2560	18 Inputs	2 0-10V Analog Inputs	8 Outputs	2 0-10V (0-20mA) Analog Outputs	10 Relays	2 Slots	Ethernet Port	2 Interfaces	1 Interface	1 Interface	Real Time Clock



## MAXI Automation PINOUT V1.1



*Demo Version*  
*full book yui987pr@yahoo.es*



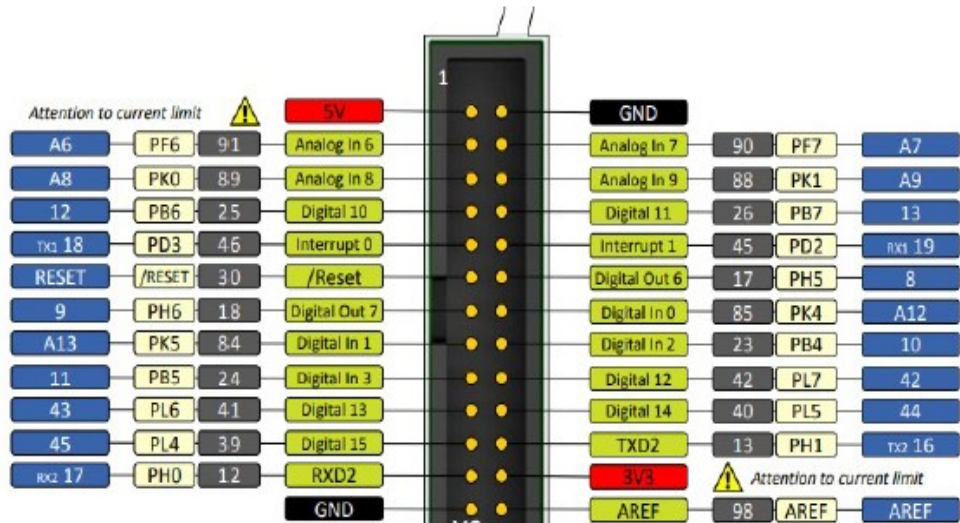
GND
POWER
CONTROLLINO Function
PHYSICAL PIN
PORT PIN
ARDUINO Mega Board

General Information  
 Pay attention

CHIP used ATmega 2560-16AU

# 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!



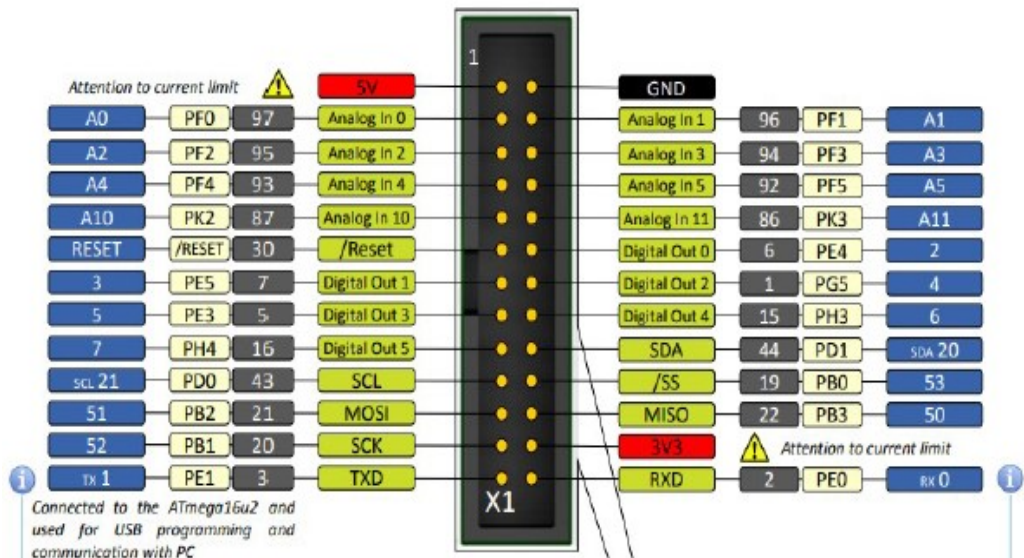
*Demo Version*

Not 100% pin compatible

*full book yui987pr@yahoo.es*

Not 100% pin compatible

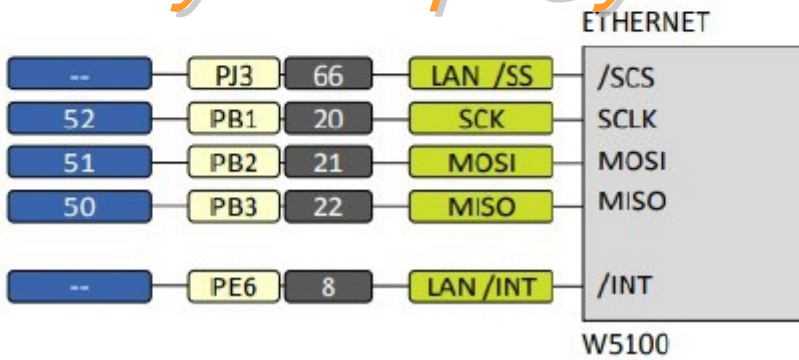
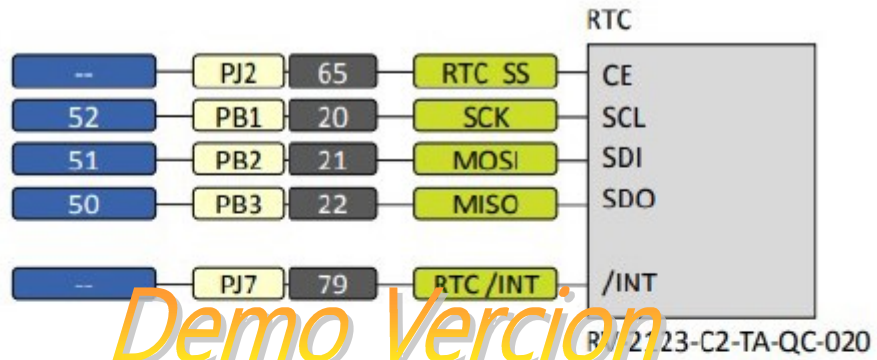
to X1 @MINI,MAXI,MEGA



Connected to the ATmega16u2 and used for USB programming and communication with PC



## PLC Controllino



*Demo Version*  
*full book yui987pr@yahoo.es*

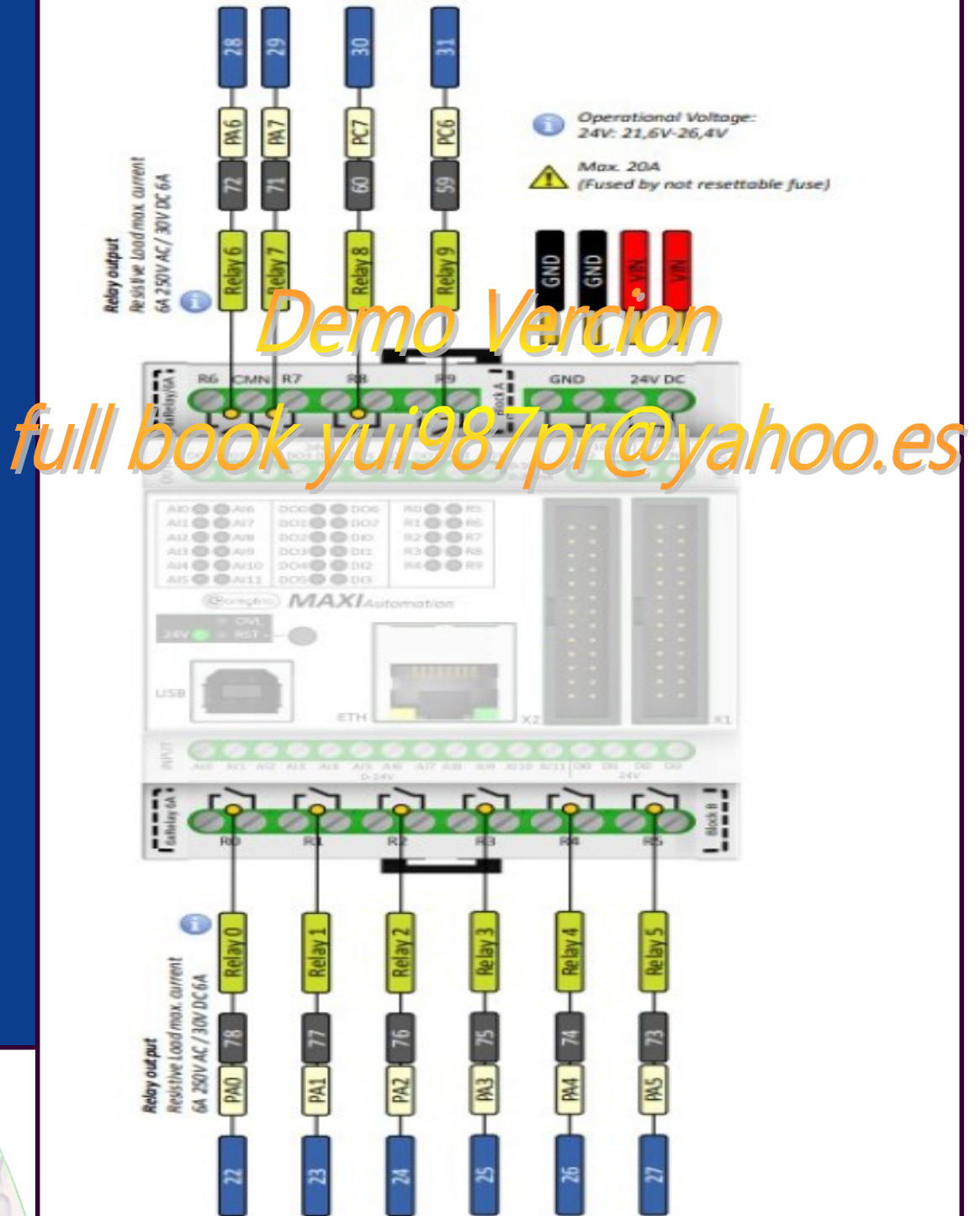
# PLC Controllino



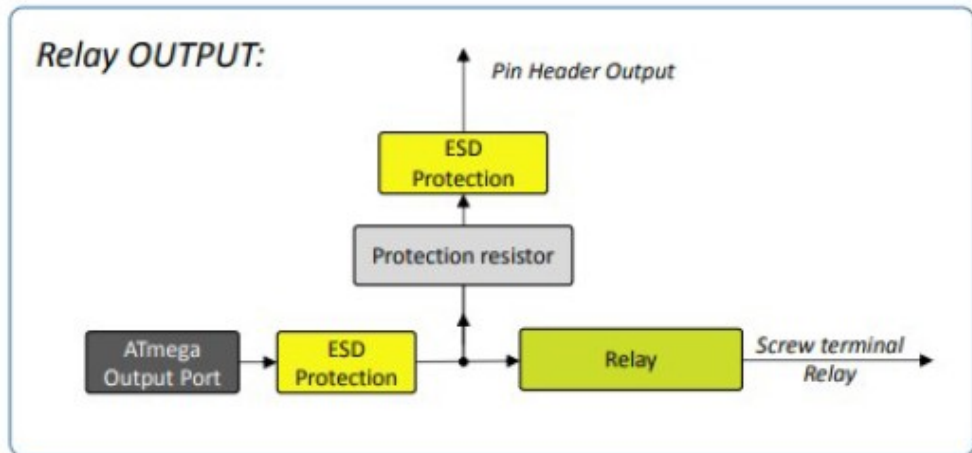
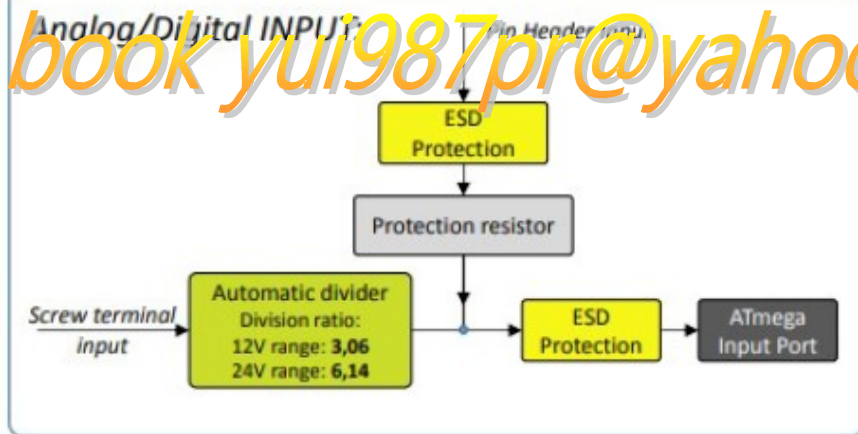
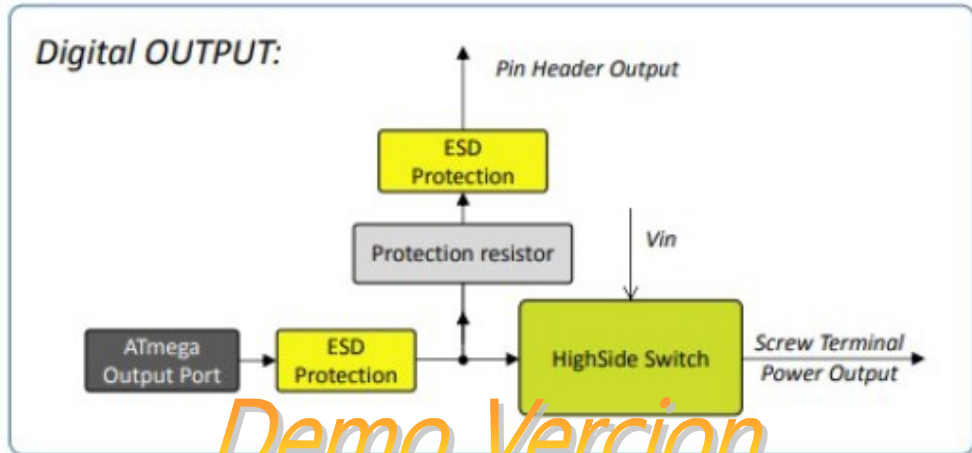
*Demo Version*  
*full book yui987pr@yahoo.es*



## Program download to PLC



## Program download to PLC



*Demo Version*  
*full book yui987pr@yahoo.es*

## Variable

The screenshot displays the LDmicro Program Editor interface. A code window shows the following assembly code:

```
0001 {MOV dest:=}
0002 { src}
[END]
```

A 'Move' dialog box is open, showing:

- Destination: `dest`
- Source: `src`

A callout box points to the variable names with the text: "Variable name that begins with symbol '#'" (Note: the image contains a typo, it should be '#').

Below the dialog box, a table lists the variables:

Name	Type
dest	general var
src	general var

The background features a large watermark: "Demo Version full book yui967pr@yahoo.es" and an illustration of a character in a white hooded robe.

## 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 ATmega2560.

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.

{OD portAddr:=} The general purpose portAddr variable now a value of 0x6.
--{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.
```



*Demo Version*  
*full book yui987pr@yahoo.es*

## Variable

create ladder programming that the program analyzes its range.

Hector M. Diaz



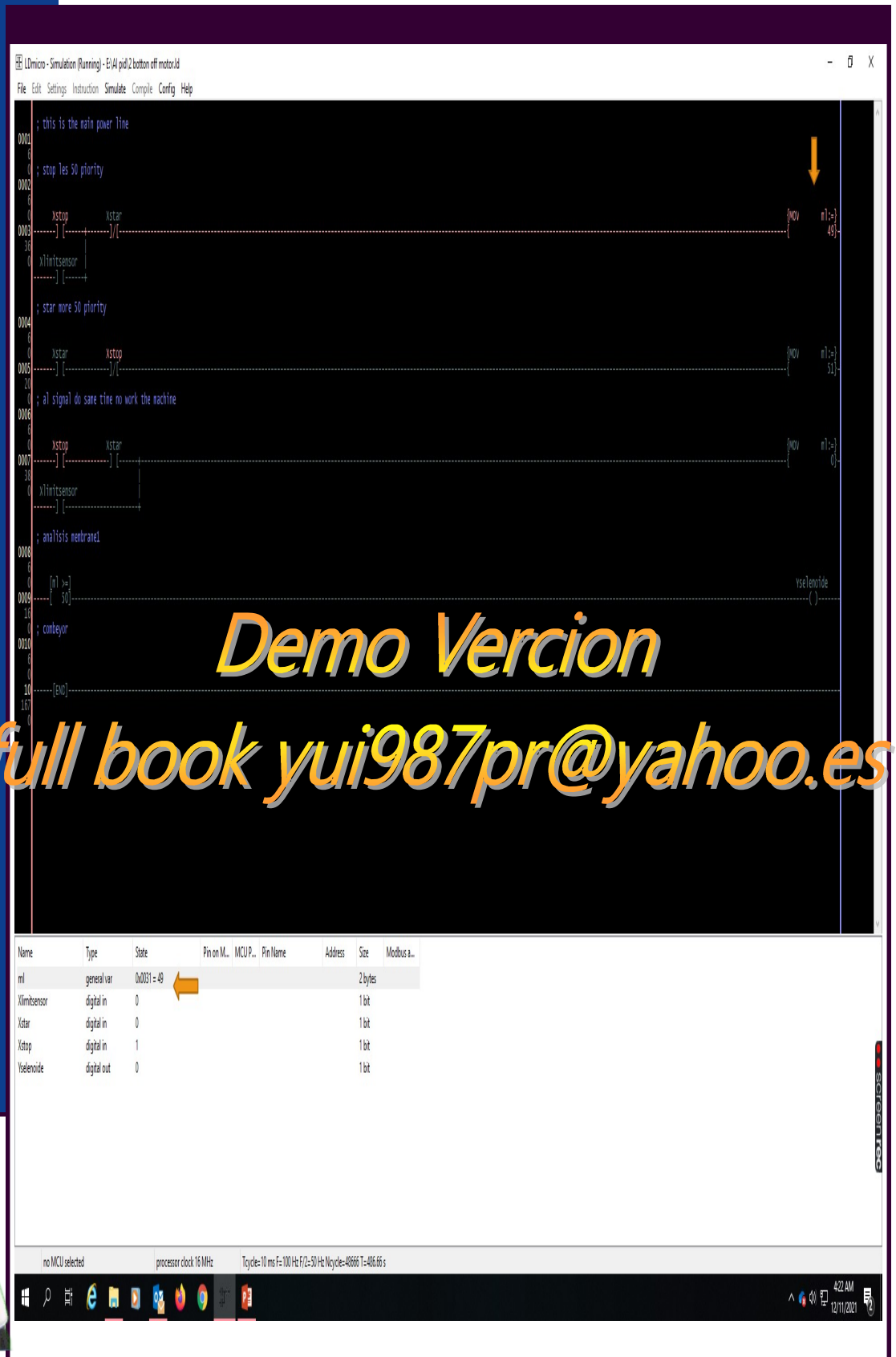
If the digital input that has a value enters the logic of greater than 50, it will have an output response. Next, you will see in the ladder

*Demo Version*

*full book yui987pr@yahoo.es*



## Variable



The screenshot shows the LDmicro simulation interface. The assembly code is as follows:

```
0001 ; this is the main power line
0002
0003 ; stop 1es 50 priority
0004
0005 xstop xstar {MOV r1:=49}
0006 xlimbsensor
0007 ; star more 50 priority
0008 xstar xstop {MOV r1:=51}
0009 ; al signal do sare time no work the machine
0010 xstop xstar {MOV r1:=0}
0011 xlimbsensor
0012 ; anal1s1s membranel
0013 [r1 >= 50] yselenoid
0014 combeyor
0015 [END]
```

The variable table at the bottom of the window is:

Name	Type	State	Pin on M...	MCU P...	Pin Name	Address	Size	Modbus a...
mi	general var	D0031 = 49					2 bytes	
xlimbsensor	digital in	0					1 bit	
xstar	digital in	0					1 bit	
xstop	digital in	1					1 bit	
yseleoid	digital out	0					1 bit	

At the bottom of the window, the status bar shows: no MCU selected, processor clock 16 MHz, Tcycle=10 ms F=100 Hz F12=50 Hz Ncycle=48666 T=486.66 s. The system tray shows the date 12/11/2021 and time 4:22 AM.

*Demo Version*  
*full book yui987pr@yahoo.es*



# Variable

The screenshot displays the LDmicro simulation environment. The main window shows assembly code with comments and timing diagrams for digital signals. A yellow arrow points to the 'mi' variable in the table below. The table lists variables with their types, states, and addresses.

Name	Type	State	Pin on ML	MCU P..	Pin Name	Address	Size	Modbus a..
mi	general var	0x0000=0	←				2 bytes	
Xlimbsensor	digital in	1					1 bit	
Xstar	digital in	1					1 bit	
Xstop	digital in	1					1 bit	
Ysolenoid	digital out	0					1 bit	

no MCU selected    processor clock 16 MHz    Tcycle=10 ms F=100 Hz F2=50 Hz Ncycle=51146 T=511.46 s

4:23 AM  
12/11/2021

*Demo Version*  
*full book yui987pr@yahoo.es*



## Variable



The screenshot displays the LDmicro simulation environment. The main window shows a ladder logic program with several rungs. The first rung is labeled "this is the main power line". The second rung is labeled "stop 1es 50 priority" and contains a normally open contact labeled "xstop" and a normally closed contact labeled "xstar". The third rung is labeled "star more 50 priority" and contains a normally open contact labeled "xstar" and a normally closed contact labeled "xstop". The fourth rung is labeled "al signal do same tire no work the machine" and contains a normally open contact labeled "xstop" and a normally closed contact labeled "xstar". The fifth rung is labeled "analisis membranel" and contains a normally open contact labeled "[n] >=" and a normally closed contact labeled "50". The sixth rung is labeled "combeyor" and contains a normally open contact labeled "[ENC]".

Below the ladder logic program, there is a table showing the state of various variables:

Name	Type	State	Pin on M...	MCU P...	Pin Name	Address	Size	Modbus a...
ml	general var	0x0035 = 51					2 bytes	
Xlimitensor	digital in	0					1 bit	
Xstar	digital in	1					1 bit	
Xstop	digital in	0					1 bit	
Yselencide	digital out	1					1 bit	

At the bottom of the screenshot, there is a Windows taskbar showing the system clock as 4:24 AM on 12/17/2021. The status bar at the bottom of the LDmicro window displays "no MCU selected", "processor clock 16 MHz", and "Tcycle=10 ms F=100 Hz F12=50 Hz Ncycle=55533 T=535.33 s".

*Demo Version*  
*full book yui987pr@yahoo.es*





