GOKARAJU RANGARAJU

Institute of Engineering and Technology
(Autonomous)

CERTIFICATE

This is to certify that it is a bonafide record of practical work done by Mr./Ms. _____,
Reg No. _____ *in the* ***"PROGRAMMABLE LOGIC CONTROLLERS (PLC) LABORATORY"*** *in* ***I-Semester*** *of* ***IV-year*** *during* ***20__ to 20__***.

Internal Examiner
Signature

External Examiner
Signature

Head of the Department
Signature

PROGRAMMABLE LOGIC CONTROLLERS (PLC) LABORATORY

Index

Exp No.	*Experiment Name*	*Date*	*Marks*	*Signature*

GOKARAJURANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

PROGRAMMABLE LOGIC CONTROLLERS LAB

Course Code:GR20A4023 **L/T/P/C:0/0/4/2**

IV year I semester

COURSE OBJECTIVES

1. Know the different Programming Languages of PLC.
2. Execute Logic Gates in Ladder Logic of PLC.
3. Examine various experiments of PLC in FBD.
4. Apply Timer and Counter for different industrial applications.
5. Design various application of PLC like Traffic Light Control, Water Level Control, etc.

COURSE OUTCOMES

1. Ability to learn different programming languages of PLC.
2. Implement all the Logic Gates in Ladder Logic.
3. Perform different FBD programming experiments of PLC.
4. Ability to use PLC timers and counters for the various applications.
5. Design and implementation of different applications of PLC like Traffic Light Control, Water Level Control, etc

List of Experiments

Task-1: Experiments on Ladder Programming

- Logic Gates.
- Latching and Unlatching
- Interlocking
- Forward and Reverse direction control of Motors.

Task-2: Experiments on FBD Programming

- Different applications of Push buttons.
- Working of different types of Timers.
- Working of different types of Counters.
- Sequential operation of ON/OFF a set of lights.
- Latching and Unlatching of a Motor.

Task-3: Applications of PLC

- Water Level Controller.
- Traffic Light Control
- Lift Control System

CONTENTS

Experiments on Millenium Ladder Logic:

1. Logic Gates
2. Latching and Unlatching
3. Interlocking
4. Sequential operation of ON/OFF of a set of lights
5. Counters

Experiments on Millenium FBD Logic:

1. Different applications of Push buttons.
2. Working of different types of Timers.
3. Working of different types of Counters.
4. Sequential operation of ON/OFF of a set of lights.
5. Latching and Unlatching of a Motor.
6. Automatic indication of water tank level.
7. Traffic lights indication.

Experiments on Siemens PLC:

1. Logic Gates
2. Latching and Unlatching of Motors
3. Forward And Reverse Direction Control of Motors

Introduction to Millenium PLC.

Experiments on Millennium Ladder Logic

EXPERIMENT -1
Logic Gates

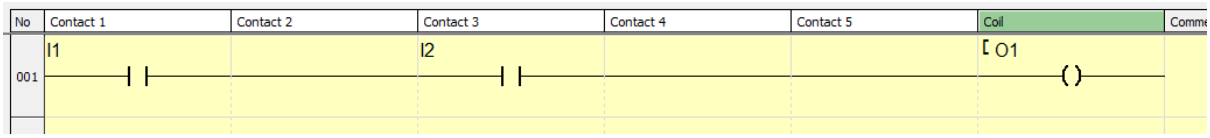
Basic Logic gates , AND , OR , NOT are demonstrated in ladder logic.

AND: Series operation: the output is high only when both inputs are high.

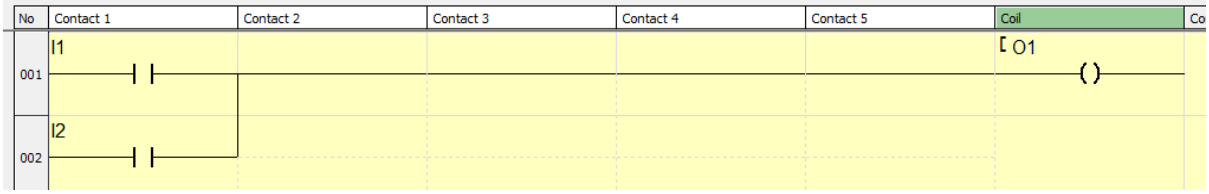
OR: Parallel operation: the output is high when any one of the inputs is high or both are high.

NOT: Inverted operation: the output is high when input is low and vice versa.

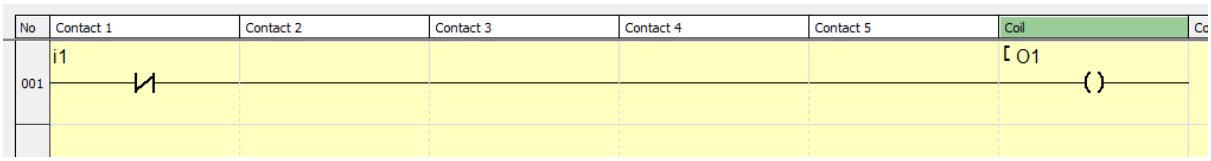
AND Logic:



OR Logic:



NOT Gate:



EXERCISE:

1. Implement NAND and NOR logics.

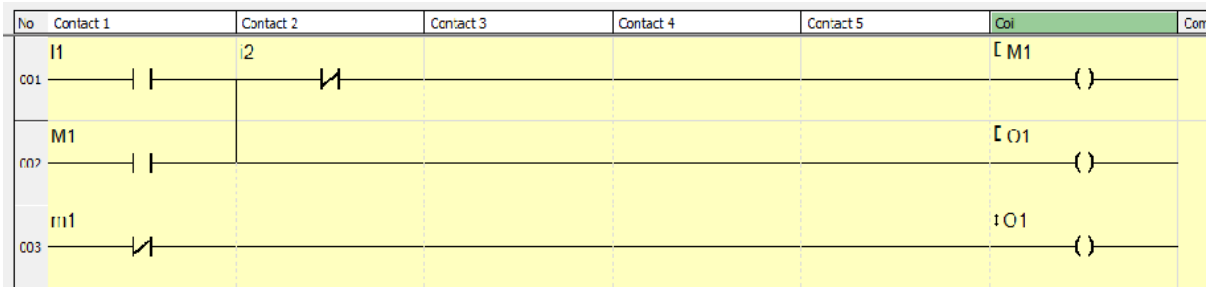
EXPERIMENT -2
Latching and Unlatching

When A Button Is Pressed Once The Output Is ON And Another Button Is Pressed To Stop The Output.

There are two input push button controls, one start button and one stop button. If start button is pressed, output is ON and even if start button is OFF position the output is still ON. To make the output OFF, stop button is used.

The above technique is called latching and unlatching.

Latching And Unlatching An Output:



EXERCISE:

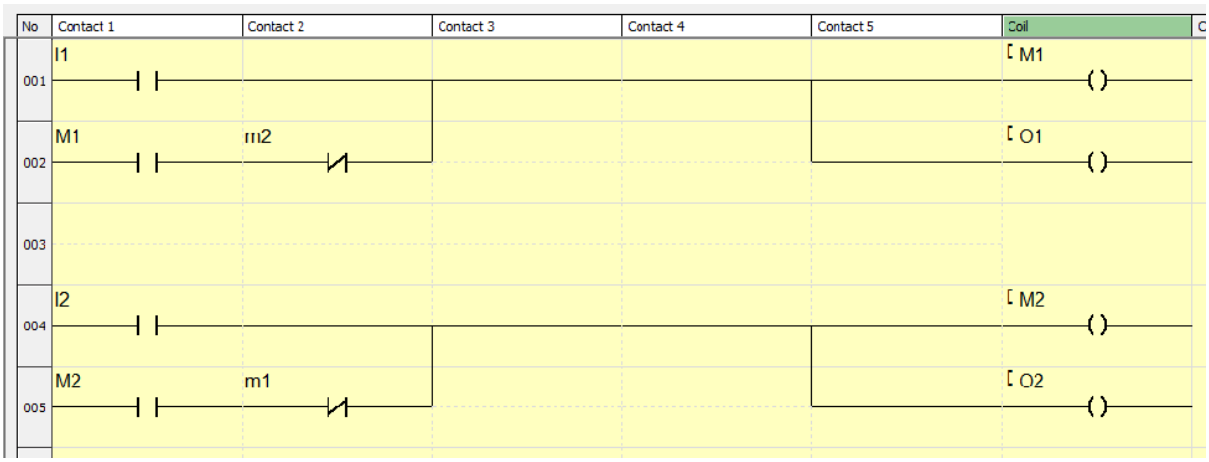
1. Ladder logic for the following motor control specifications:
   - A motor must be started and stopped from any one of the start/stop pushbutton stations.
   - Each start/stop station contains one NO start button and one NC stop button.
   - Motor OL contacts are to be hardwired.

EXPERIMENT -3
Interlocking

Interlocking is a kind of application when there are two devices and they are not required to operate at the same time. If one is ON the other should be OFF and vice versa.

This method of control is called interlocking.

Interlocking Of Two Devices:



EXERCISE:

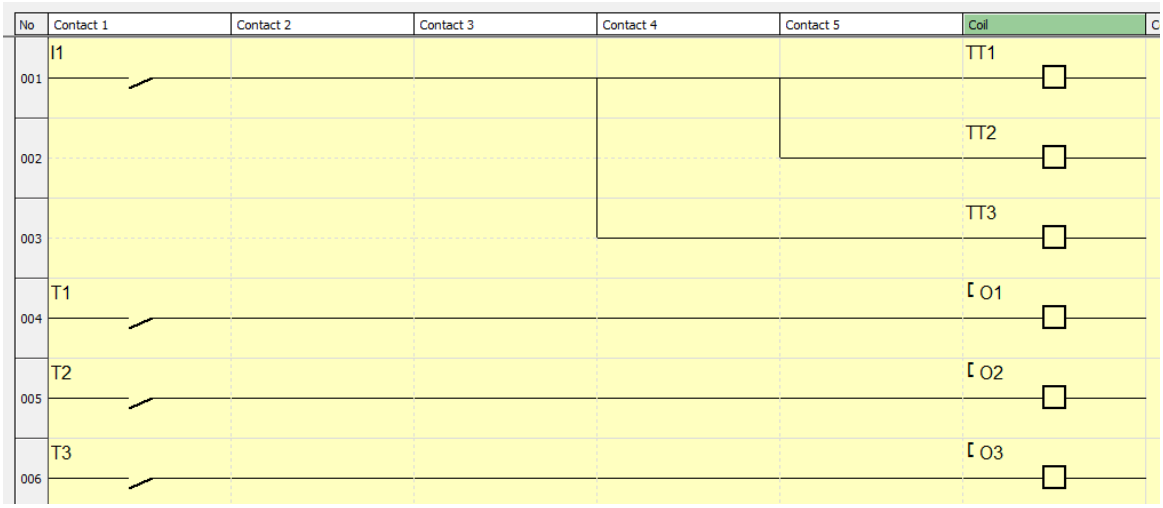
1. A pump is to be used to fill two storage tanks. The pump is manually started by the operator from a START/STOP station. When the first tank is full, the control logic must be able to automatically stop flow to the first tank and direct flow to the second tank through the use of sensors and electric solenoid valves. When the second tank is full, the pump must shut down automatically. Indicator lamps are to be indicated to signal when each tank is full.

Experiment -4
Sequential Operation Of On/Off Of A Set Of Lights

Sequential Operation Of ON/OFF Of Lights Using Timers

Three lights are switched on and the lights are automatically off in a given sequential order with a preset time delay using a timer.

Sequential Operation Of ON/OFF Of Lights Using Timers



EXERCISE:

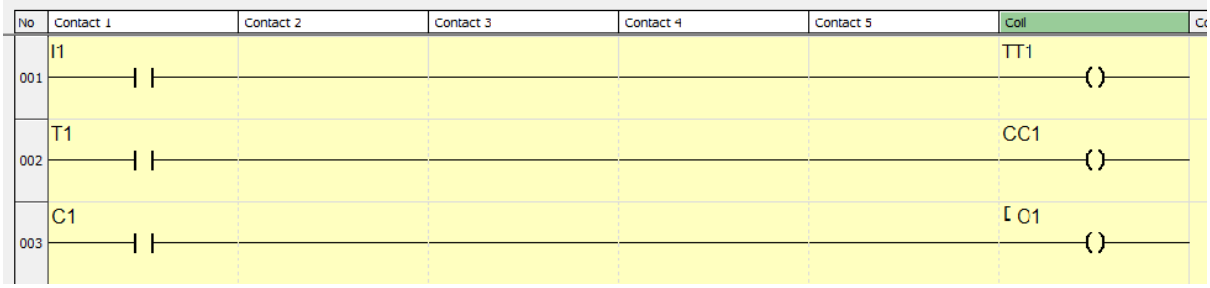
1. Implement alternate ON/OFF of 8 lights with a certain time delay.
2. Automatic switch ON of 8 lights one by one with a certain time delay and switch off one by one.
3. Decorative flickering of ON/OFF of lights continuously till a stop button is pressed.

EXPERIMENT -5
Counters

Counters are demonstrated using ladder logic. Different types of counters like upcounting, downcounting, preset counting etc also can be explored.

Upcounting is shown in the following logic.

Upcounting Of Pulses To A Preset Value:



EXERCISE:

1. Make a program to increase the counter by one with each pulse from the pulse generator (on rising edge), and decrease another counter by the same pulse.
2. Write a program that will increment a counter's accumulated value 1 count for every 60s. A second counter's accumulated value will increment 1 count every time the first counter's accumulated value reaches 60. The first counter will reset when its accumulated value reaches 60, and the second counter will reset when its accumulated value reaches 12.
3. Implement in hardware, speed measurement in rpm using a bicycle gear wheel by counting the number of tooth using a proximity sensor.

Experiments on Millennium FBD Logic:

EXPERIMENT -1

Different Applications Of Push Buttons

Different types of push button controls in automation technologies are discussed and simulated as shown below:

Electrical circuits need to be complete to work. The electricity must be able to flow uninterrupted through the various wires and components. But circuits that are complete all the time aren't as useful as ones that work only when we want them to. This is what a switch does. Some switches are hidden inside machinery; others are where we can see and use them. The push button switch has thousands of familiar uses, from elevators to car stereos. It comes in two basic kinds: momentary and non-momentary.

Construction

A push button switch is a small, sealed mechanism that completes an electric circuit when you press on it. When it's on, a small metal spring inside makes contact with two wires, allowing electricity to flow. When it's off, the spring retracts, contact is interrupted, and current won't flow. The body of the switch is made of non-conducting plastic.

Momentary Contact

Momentary switches work only as long as you press on them, like the buttons on a phone, calculator or door buzzer. They can be subdivided into normally-on and normally-off types.

Normally-Off

With the normally-off switch, there's no connection till you push the button. Most push button switches are used this way. Examples include doorbell buttons, cell phone keys and garage door openers.

Normally-On

Here the switch conducts normally, but interrupts the circuit when you press on it. This is more specialized, and may be used in conjunction with a wiring trick. For example, connecting a

normally-on switch in parallel with a light bulb will light the bulb when the button's pushed; otherwise, current will flow through the switch, leaving the bulb off.

Non-Momentary Contact

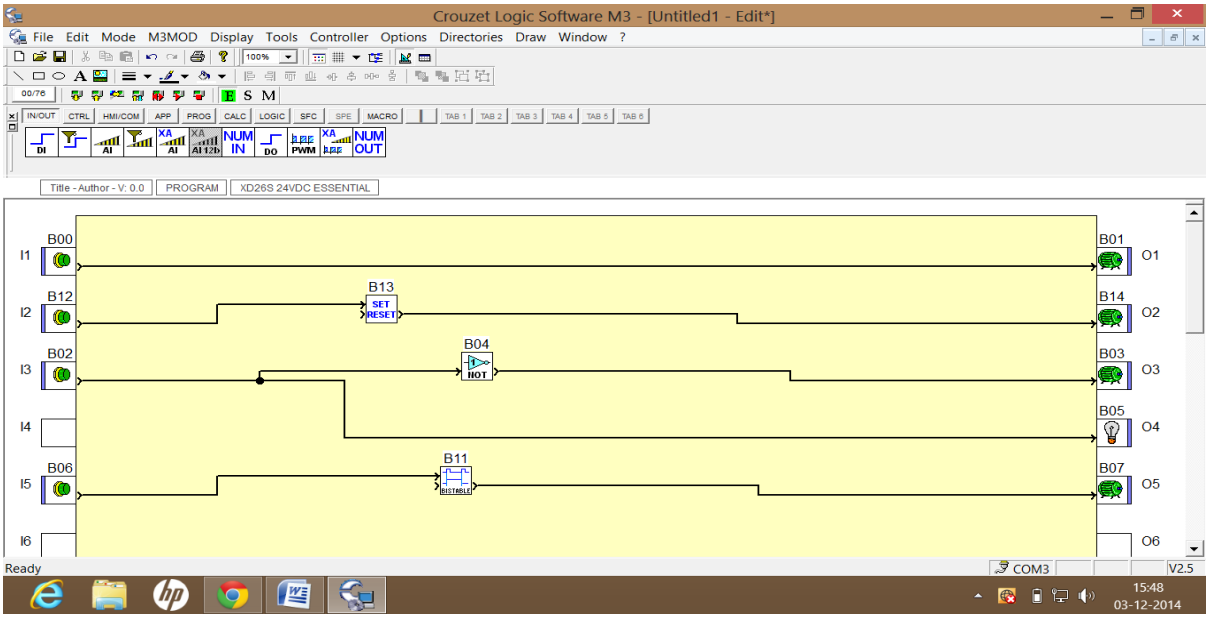
Non-momentary switches take one push to turn on, another to turn off. TVs and stereos use non-momentary switches for their power buttons.

Ratings

For reliability and safety, switches are rated for current and voltage. This is necessary since higher voltage or current requirements call for larger, more expensive parts, and switches, like most parts, are only as large as necessary. Cell phones and portable radios have small requirements; industrial machines have large requirements.

Each one of the above applications in push button technologies can be experimented on the PLC. A set of four experiments are carried out as follows:

Push Button Functions



EXPERIMENT -2
Working Of Different Types Of Timers

Use Of ON Delay Timer and OFF Delay Timer For Process Control

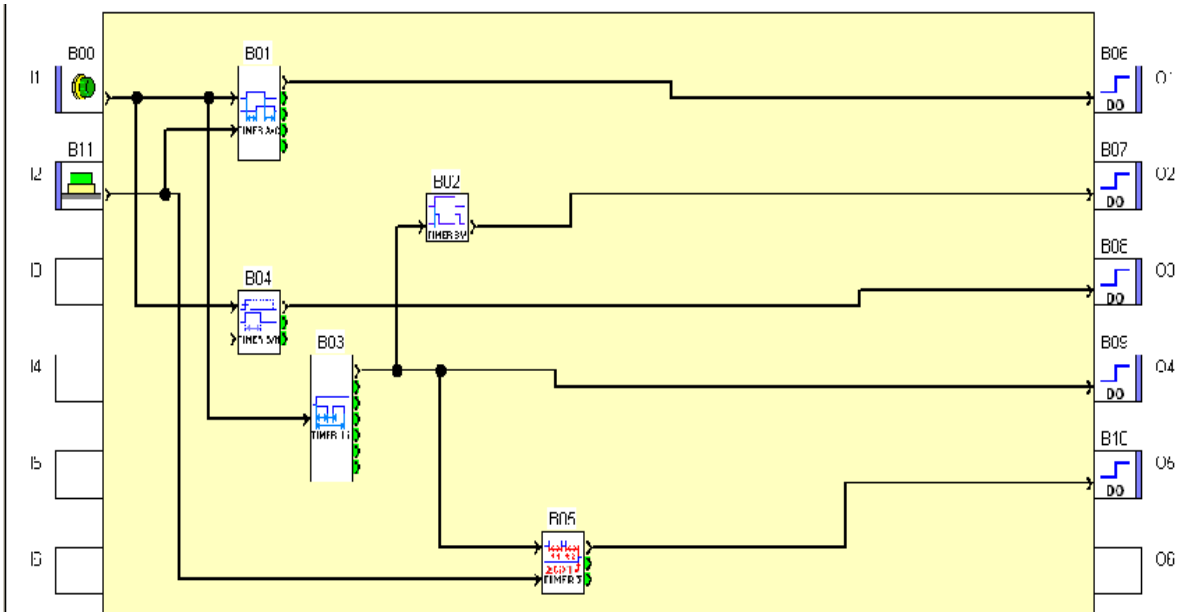
Different types of timers are explored.

1. AC Timer
2. BW Timer
3. Li/L Timer
4. B/H Timer
5. Totalizer function

HARDWARE REQUIREMENTS:

1. 5 lights of 230V , 10W
2. Set of single stranded wires
3. Relay card

Use Of ON Delay Timer And OFF Delay Timer For Process Control



EXERCISE:

1. Show application of all the different types of timers in hardware.

EXPERIMENT -3
Working Of Different Types Of Counters

Application Of Counters In Process Control.

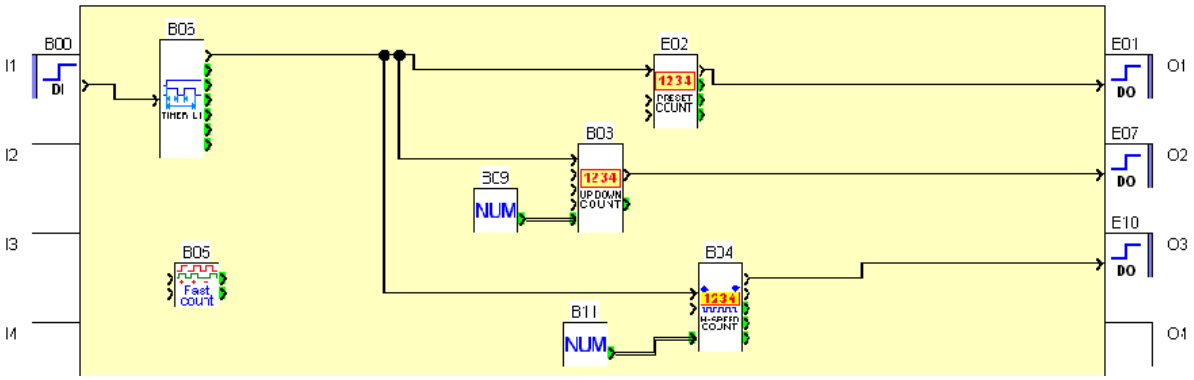
Different types of counters are explored

1. Preset Counter
2. UpDown Counter
3. HighSpeed Counter

HARDWARE REQUIREMENTS:

1. 8 lights of 240V, 10W (4 green and 4 red)
2. An arrangement for lights as shown in figure
3. Set of single stranded wires
4. Relay card

Application Of Counters In Process Control



EXERCISE:

1. Implement on hardware all the different types of counters.

EXPERIMENT -4
Sequential Operation Of ON/OFF Of A Set Of Lights

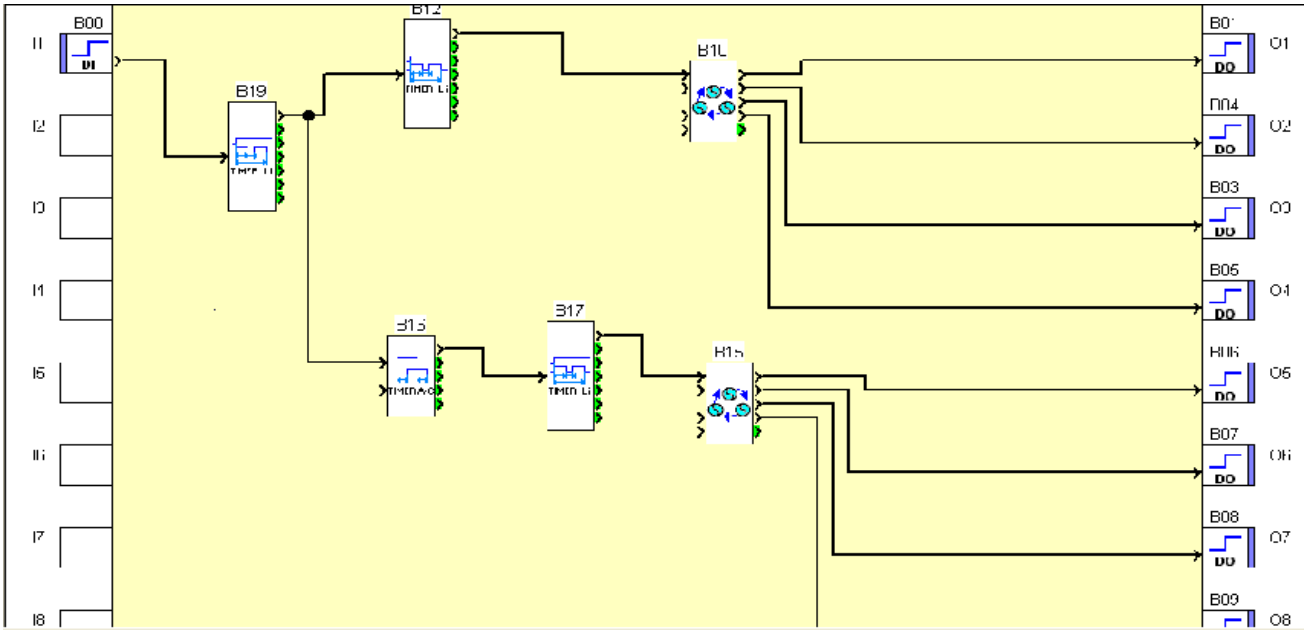
Sequential Operation Of Lights (One After Another)

A set of 8 lights are made to operate sequentially one after another with a preset timer, controlled with a single input.

HARDWARE REQUIREMENTS:

1. 8 lights of 230V, 10W
2. Set of single stranded wires
3. Relay card

Sequential Operation Of Devices (One After Another).



EXERCISE:

1. All the 8 lights should be ON for 5 sec and OFF for 5 sec , the process continues till a stop button is pressed.
2. All the 8 lights should be ON and they should get automatically OFF from 8th to first sequentially with a certain time delay.

EXPERIMENT -5
Latching and Unlatching of a Motor

When A Button Is Pressed Once The Motor Is Started And Another Button Is Pressed To Stop The Motor.

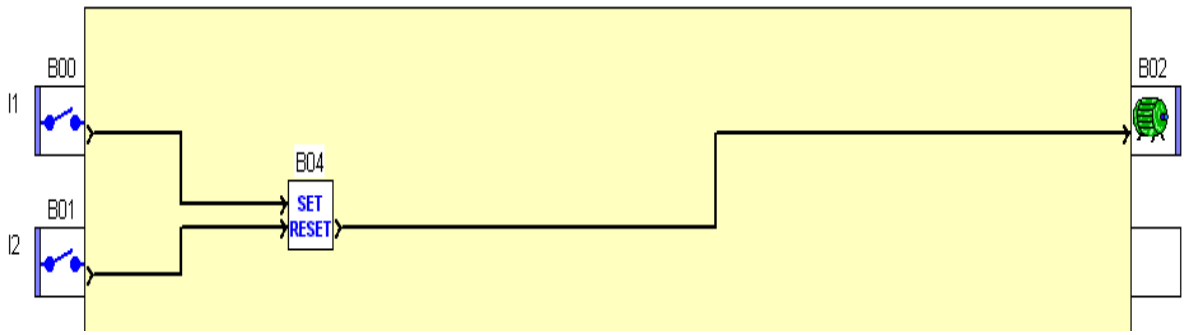
There are two input push button controls, one start button and one stop button. If start button is pressed, motor is ON and even if start button is OFF position the motor continues to run. To stop the machine stop button is used.

The above technique is called latching and unlatching.

HARDWARE REQUIREMEMNTS:

1. 1 single phase induction motor
2. NO/NC contactor / relay
3. 2 push button switches for 'start' and 'stop' operations.

Motor Control using Push Button



EXERCISE:

1. Three motors: main motor, lubrication pump and a cooling fan are to be controlled in following way - the main motor is ON for 10 sec, lubrication pump is ON/OFF alternately for 3 sec three times and after the process is done the cooling fan runs for 10 sec and the whole process repeats till a stop button is pressed.

EXPERIMENT -6
Automatic Indication Of Water Tank Level

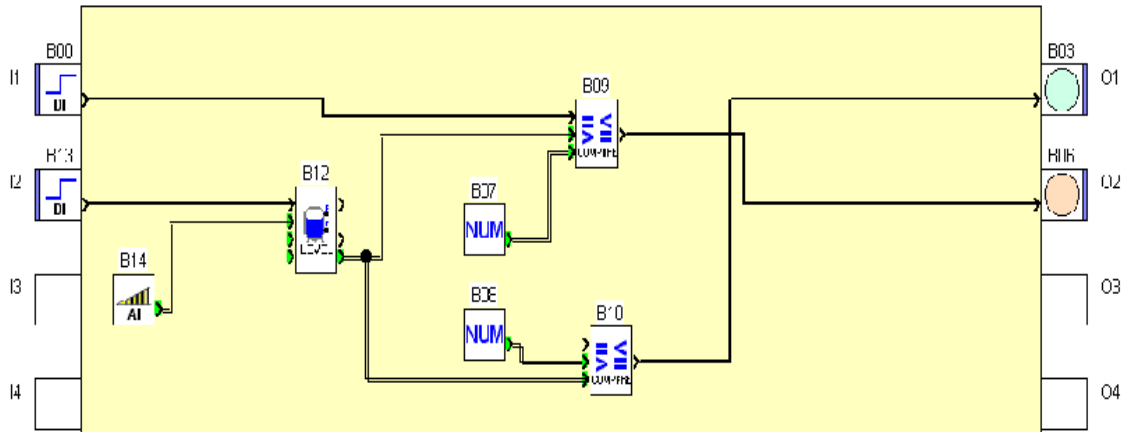
Tank Level Control With Automatic Level Indication By Analog Input

The full and empty levels of water in the tank are indicated using different lights. The levels are sensed using a sensor.

HARDWARE REQUIREMENTS:

1. 2 lights of 240V, 10W (1 green and 1 red)
2. An arrangement for water tank
3. 2 sensors
4. Set of single stranded wires
5. Relay card

Tank Level Control With Automatic Level Indication By Analog Input



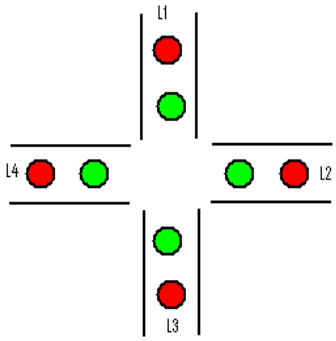
EXERCISE:

Implement the above experiment in hardware and also extend the same for an automatic control of filling the tank when the “empty” indication is shown.

EXPERIMENT -7
Traffic Lights Indication

Automatic Traffic Lights Control

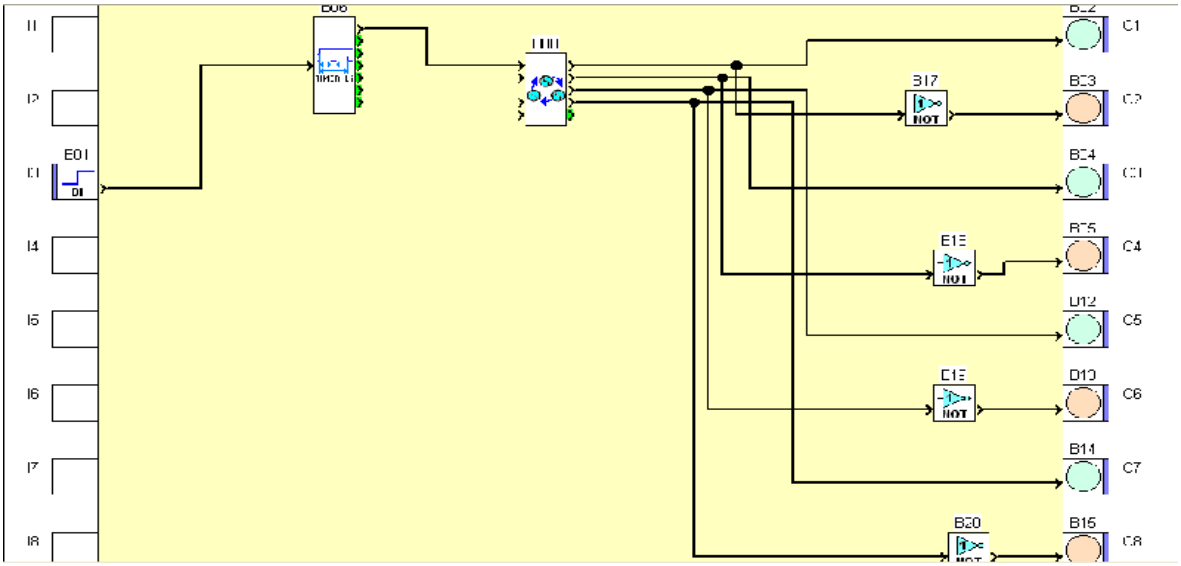
The green light in L1 and red lights in L2, L3, L4 are periodically shifted with preset time delay.



HARDWARE REQUIREMENTS:

1. 8 lights of 240V, 10W (4 green and 4 red)
2. An arrangement for lights as shown in figure
3. Set of single stranded wires
4. Relay card

Traffic Lights Control



EXERCISE:

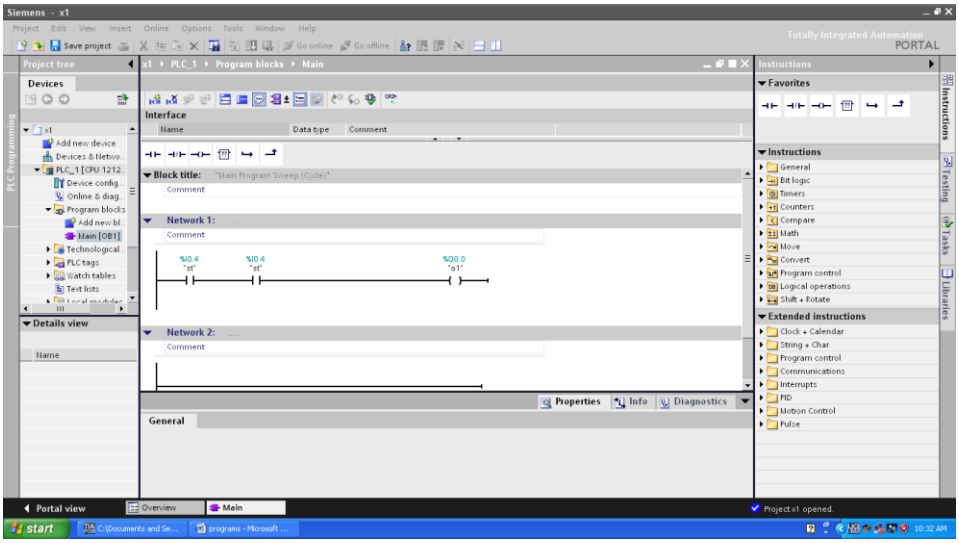
1. Implement the above experiment in hardware and extend the same logic with three lights in each lane.
2. Implement the same logic using sensors instead of timers.

Experiments On Siemen's Software:

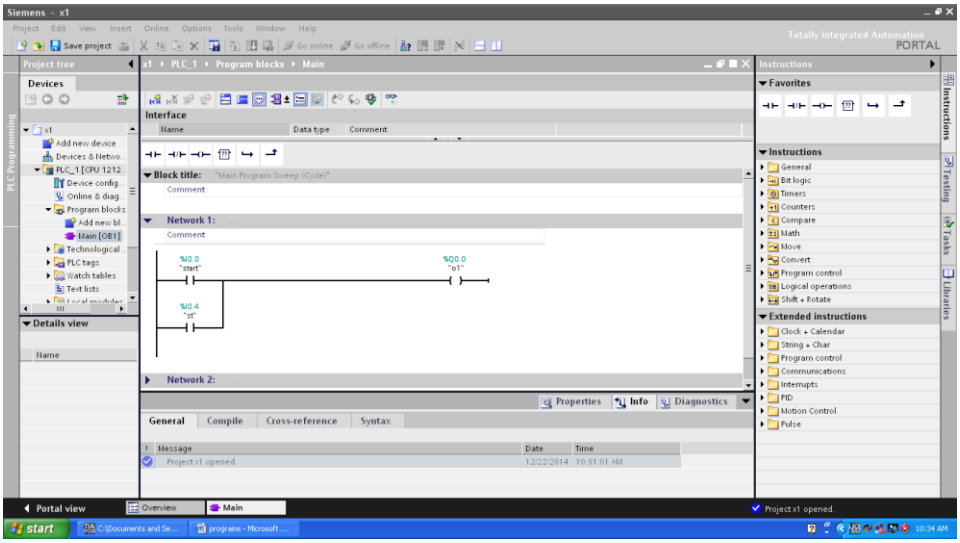
EXPERIMENT -1
Logic Gates

Basic Logic gates , AND , OR , NOT are demonstarated in ladder logic.

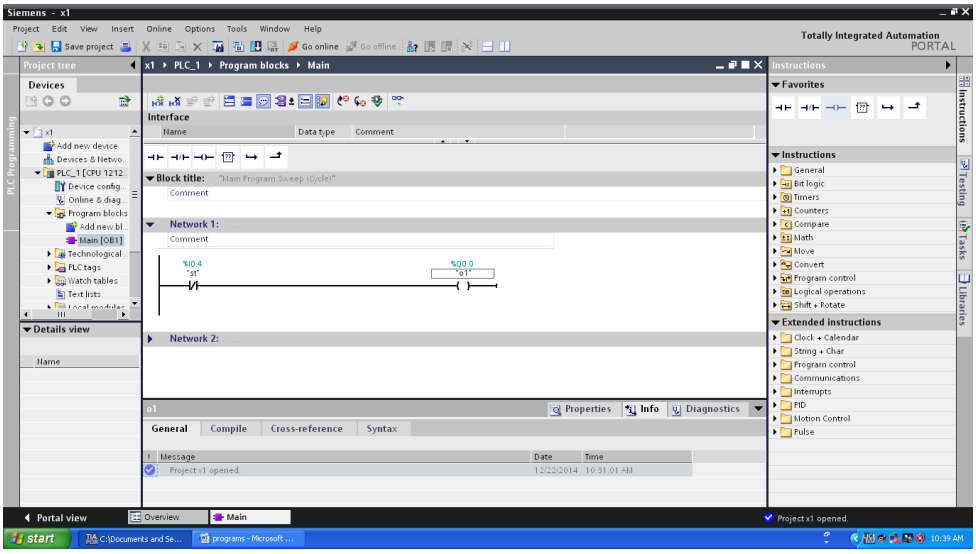
AND: Series operation: the output is high only when both inputs are high.



OR: Parallel operation: the output is high when any one of the inputs is high or both are high.



NOT: Inverted operation: the output is high when input is low and vice versa.



EXERCISE:

1. Implement NAND and NOR logics.
2. A PLC motor controller has two START buttons and two STOP buttons. The motor is to run if two RUN buttons depressed simultaneously. The motor should run when the buttons are released. Motor stops by depressing any STOP button stops. Construct a LAD for this motor control task. Use the following symbols for the inputs and output:

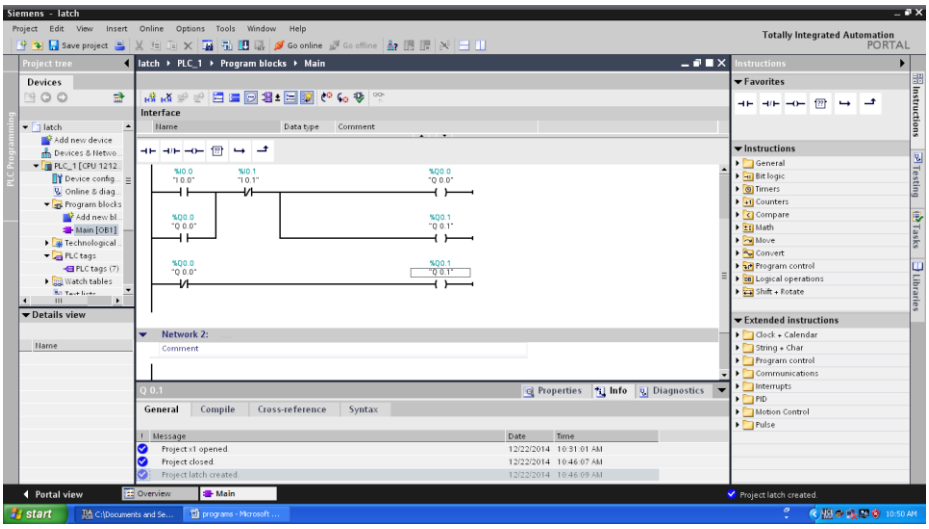
START Button-1	START1
START Button-2	START2
STOP Button-1	STP1
STOP Button-2	STP2
MOTOR Starter	MOTOR

EXPERIMENT -2
Latching And Unlatching Of A Motor

When A Button Is Pressed Once The Motor Runs And Another Button Is Pressed To Stop The Motor Stops.

There are two input push button controls, one start button and one stop button. If start button is pressed, motor is ON and even if start button is OFF position the motor continues to run. To stop the motor, stop button is used.

The above technique is called latching and unlatching.



EXERCISE:

Implement on hardware for latching and unlatching using pushbuttons for

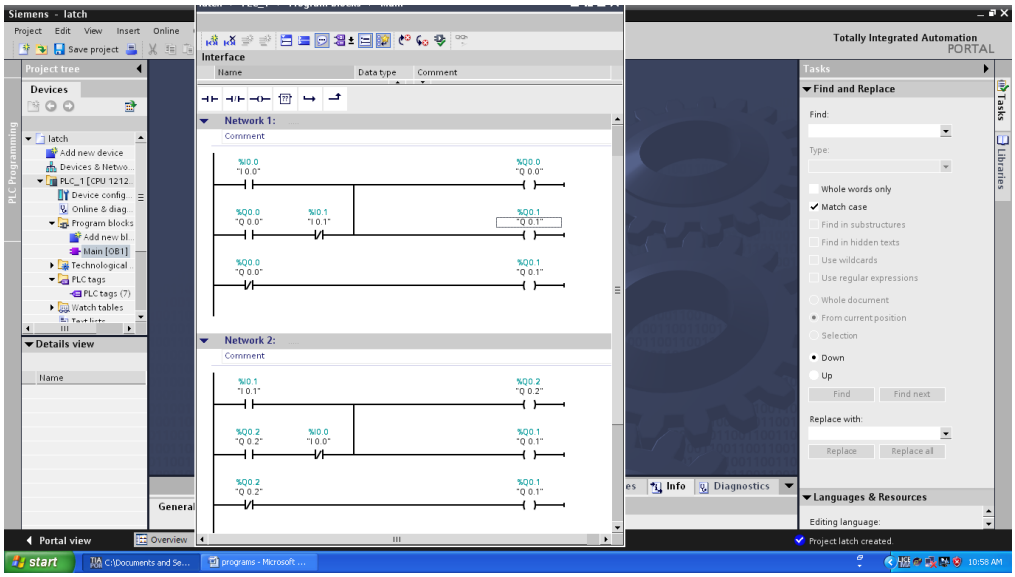
(i) Single phase induction motor

(ii) Three phase induction motor

Experiment -3
Forward And Reverse Direction Control Of Motors

Two push buttons are used to control the direction of rotation of a motor. If two phases of a three phase induction motor are interchanged, the direction of rotation of a motor is reversed. The same principle is programmed using ladder logic.

The two switches named “forward” and “reverse” are used to switch the direction of rotation of the motor.



EXERCISE:

1. Implement the same experiment on hardware.

Introduction on Millenium PLC

INTRODUCTION:

Programming a Millenium 3 logic controller is simple and straightforward. This can be accomplished using either Ladder Logic (LD) or the more intuitive Function Block Diagrams (FBD).

LD Language: Ladder Language

Ladder Diagram (LD) language is a graphic language. It can be used to transcribe relay diagrams, and is suited for combinational programs.

You can use basic graphic symbols: contacts, coils, and blocks. Specific calculations can be executed within the operation blocks.

It has certain limitations: you can't use analog inputs and there is no option for arithmetic operations.

FBD Language: Function Block Diagram

FBD mode allows graphic programming based on the use of predefined function blocks. It offers a large range of basic functions: timer, counters, logic, etc.

These languages use:

Predefined function blocks: Timers, Counters

Specific Functions: Time Management, Data Conversion, Communication, etc.

For simple programs both programming options are available. For more complex applications FBD is the recommended option. This document will focus only on the FBD language.

Operating Modes:

There are several operating modes for the programming workshop:

1. **Edit Mode:**

   The Edit mode is used to construct programs in FBD mode, which corresponds to the development of the application. In this mode you can: create macros, password protect your program, display dependencies between blocks, display a parameter summary table, preview function blocks by theme, or obtain online help for each function block.

2. **Simulation Mode:**

   When in Simulation mode the program is executed offline directly in the programming workshop (simulated on the PC). Each action on the chart (changing the state of an input,

output forcing) updates the simulation windows. You can simulate a power failure or program timing, modify analog variables via the Millenium's virtual screen, or create a time-based jump event without changing the time on the PC.

3. Monitoring Mode:

When in Monitoring mode the program is executed on the controller and the programming workshop is connected to the controller (PC □□ controller connection). In this mode you can view machine operation in near real time on your PC, modify parameters via the front panel, or conduct progressive debugging and validate each part of the application.

In simulation and monitoring modes, it is possible to:

1. View the output states and function block parameters of the program corresponding to the wiring sheet in the supervision window.
2. Force the inputs/outputs to test program behavior under specific conditions.

GETTING STARTED WITH THE MILLENIUM-3

The MILLENIUM 3 is programmed using the **CLS M3** software workshop. It should therefore be connected to your PC. You cannot create or modify a program from the Millenium front panel.

PC Resources:

PC Pentium II 300 MHz (600 MHz recommended), 128 Mb of RAM memory (256 MB recommended). Compatible with windows 2000, NT 4.0 SP5, XP, Vista and Windows 7.

Installing The Software Workshop:

If you have the M3 SOFT CD ROM (Part Number **88 970 111**) insert it into your PC and follow the instructions.

The programming software is also available for download from www.crouzet.com. Once downloaded extract the files and run the **Setup.exe** file. This will start the installation process.

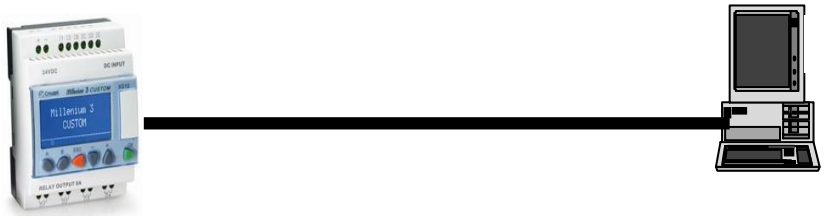
Multiple installations are possible with the different available languages: English, French, German, Italian, and Spanish.

Connection To The pc

The connection with your PC can be made with any of the following options:

1. Via the USB cable, P/N 88 **970 109**.

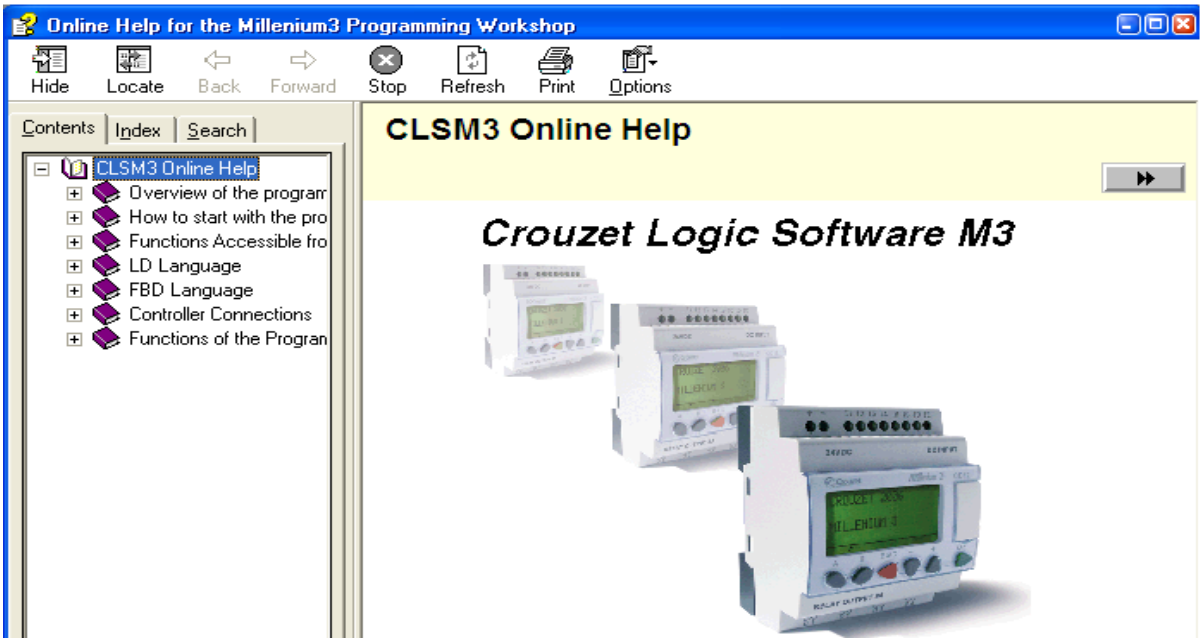
2. Via the Serial cable, P/N
   88 970 102
3. Via the Bluetooth adapter, P/N **88 970 104**



M3 Software Environment:

Accessing Help:

The CLS 3 software workshop Help is accessible from the menu bar by clicking on **?** then Help, or by clicking on the **?** icon on the Standard Toolbar.



Help is also available for each function block. Just double click on the function block and then click on the **?** button.

Toolbars

The toolbars contain shortcuts to elements in the menu.

The Controller Toolbar

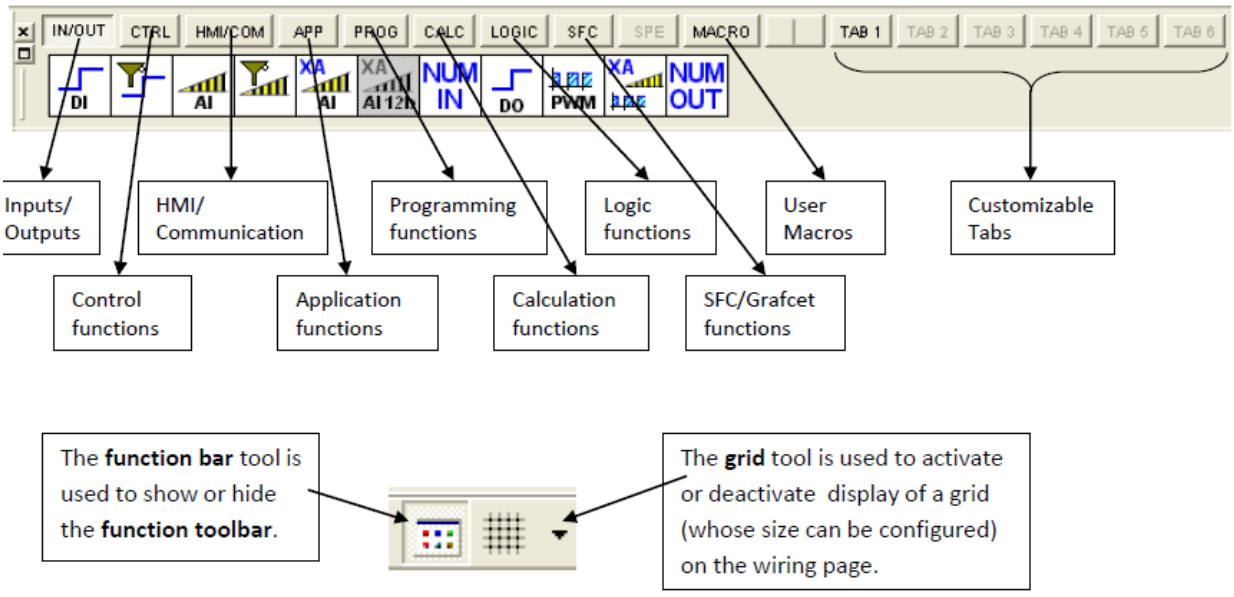
This toolbar is used to manage actions on the Millenium and also to select the application mode (Editing, Supervision, Monitoring). Passing the cursor over the button icon displays the action associated with the button.

The Controller Toolbar

This toolbar is used to manage actions on the Millenium and also to select the application mode (Editing, Supervision, Monitoring). Passing the cursor over the button icon displays the action associated with the button.

The Function Toolbar

The function bar contains all the Millenium functions.



FUNCTIONS

Note: The following descriptions are illustrated in some cases with working examples. Double-click on the file icon to open the application, then select Simulation mode.

1. Inputs/Outputs:

DI (Discrete Input): On/Off input

The type of Discrete input can be selected from the Parameters window. This is displayed in the edit and supervision windows. These include: Discrete input, Contact, Limit switch, Proximity sensor, Presence sensor, Illuminated pushbutton, Selector switch, Pushbutton, and Normally open relay.

DI.pm3

See Help: double-click on the block and click on **?**.

AI (Analog Input): Analog inputs are only available on DC powered controllers. This type of input can accept an input voltage of 0 to 10 VDC, or 0 to Controller Supply Voltage (Potentiometer), corresponding to an internal value of 0 to 1023.

AI.pm3

See Help: double-click on the block and click on **?**

The type of Analog input can be selected from the Parameters window. This is then displayed in the Edit and Supervision windows. These include: Analog Input (by default), Analog Input 0-10V, Potentiometer, and Temperature.

Filtered inputs: These types of inputs can be used to suppress interference.

DI_1.pm3

Management of a light signal which is activated when 10 products are at the end of the line. Since the product is subject to bounce on arrival at the sensor, the input should be filtered.

See Help: double-click on the block and click on **?**

DO (Discrete output): on/off output.

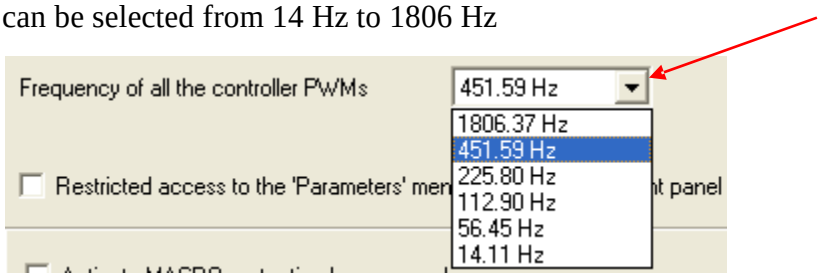
The type of discrete output can be selected from the Parameters window: Discrete Output, Normally open relay, Lamp, Solid state relay, Valve, Actuator, Motor, Resistance, Audible signal, Green indicator light, Red indicator light, Orange indicator light, Indicator light, Heating, Fan.

DO.pm3

See Help: double-click on the block and click on **?**

PWM (Pulse Width Modulation) output: PWM outputs are available on controllers with solid state (transistor) outputs. The value on this type of outputs can vary between 0 and 255.

The frequency for the PWM outputs on the base controller is set during programming by clicking on the PROGRAM button at the top of the wiring sheet and then going to the Configuration tab. This basic frequency can be selected from 14 Hz to 1806 Hz



PWM.pm3

See Help: double-click on the block and click on **?**

NUM IN, NUM OUT: Used for exchanging 16-bit integers between the controller and a communication extension.

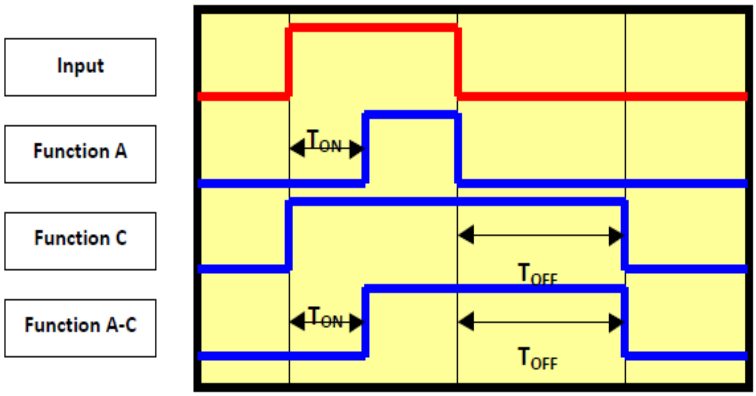
See Help: double-click on the block and click on **?**.

2. Control Functions:

Timers: You can select between 5 different types of timer functions.

After you place a timer block on the wiring sheet you can check the External setpoint box, if required: in this case the time delay setpoints will be integer-type inputs rather than being internal configurable parameters.

Timer A-C: Applies an ON delay, an OFF delay, or both delays to the output signal in relation to the input signal.



TIMER_A_C.pm3

In this example you can see how to create an A-C timer.

See Help: double-click on the block and click on ?.

BW Timer: Generates a cycle duration pulse on a rising or falling edge or on both edges of an input, according to the setting chosen in the parameters.

TIMER_BW.pm3

This block can be used to convert pushbutton actions into pulses so they can be counted. If several pushbuttons are connected to a counter input and a user holds down the pushbutton, pressing the other pushbuttons would have no effect.

See Help: double-click on the block and click on **?**

Timer Li/L: Generates pulses when the input is active. It can start in the ON part of the cycle (Li) or on the OFF part of the cycle (L).

This example shows how to make an alarm and the display flash.

See Help: double-click on the block and click on ?.

Timer B/H: Generates a pulse (the time can be configured) on a rising edge of the input.

This example shows how this timer operates.

See Help: double-click on the block and click on ?.

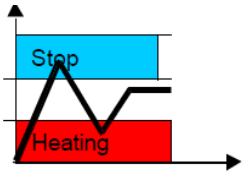
Totalizer function: It allows to count for how long the input has been held active (or inactive, depending on the mode selected). If the input is in the rest condition then the timing progress is held.

Several types of totalizer modes can be selected from the Parameters window: At, Ht, T, and Tt.

See Help: double-click on the block and click on ?.

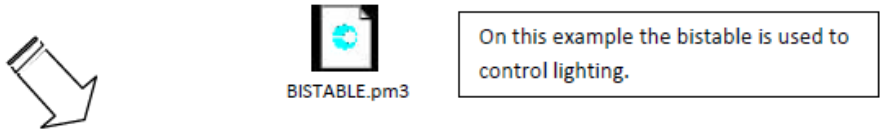
Trigger: The output changes state if the input is lower than **Schmitt** the minimum value, and the output changes state again if the input is higher than the maximum value. If the input is between the two, the output remains unchanged

This is an example of temperature regulation: the heating comes on when the input is lower than a certain temperature and goes off when this input reaches a given temperature.

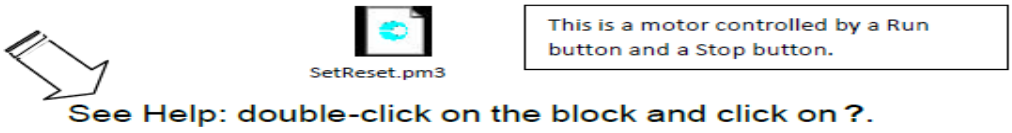


See Help: double-click on the block and click on ?.

Bistable: Provides the functionality of an impulse relay. An initial impulse sets the output to 1 then a second impulse is required to change the output back to 0.

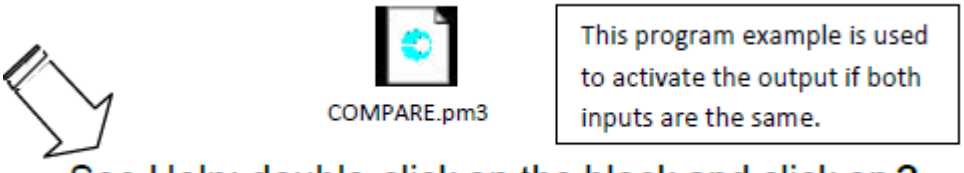


Set-Reset:Element consisting of two inputs: R for Reset and S for Set. To activate the output, simply generate a pulse on S; to deactivate it, generate a pulse on R. The priority defines the output state when both inputs are at 1.

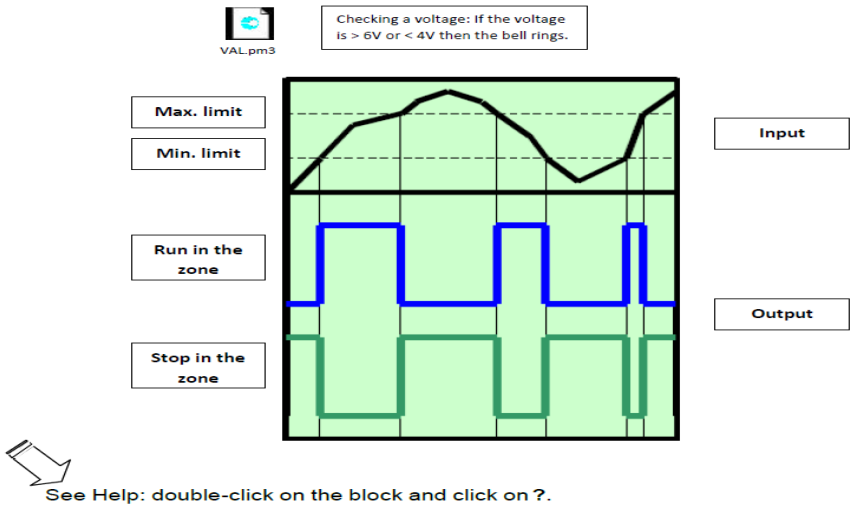


1 sec: Internal clock with a period of one second.

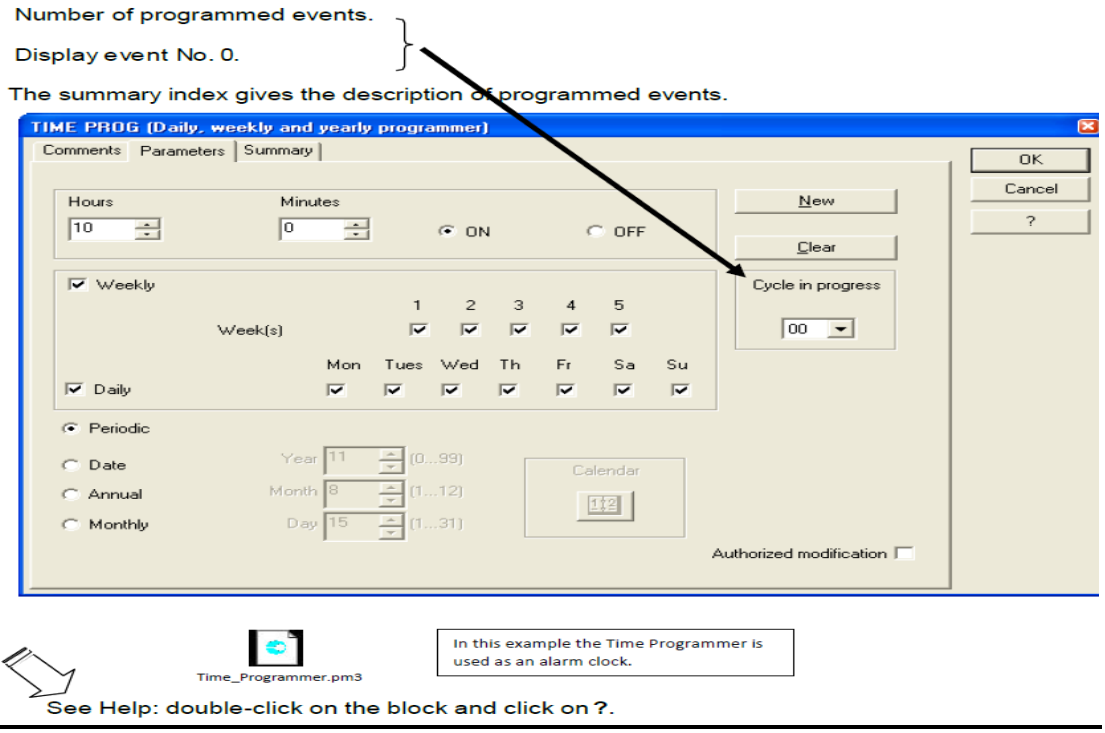
Comparison of two values: Compares two analog values using the >, ≥, =, ≠, ≤, and < operators. The output is discrete and it's activated if the comparison is true.



Zone comparison: Compares a value between two setpoints (the min and max values delimit the zone).



Time Prog (Daily, Weekly and Yearly Programmer): Activates or deactivates the output at a precise moment in the day, week or year. This block works on the event principle. To create an event, go into the Parameter tab, enter an active event number. Choose the time when this event occurs, then define the state of the output at this instant. You can select the frequency of this event. You can use the calendar at the right of the screen.



Counter: Function used for counting up to a value defined in the parameter-setting window. Once this value has been reached, the output changes to 1 until reset if the fixed output is selected or for a certain period if the pulse output is selected. The count value and the maximum value can be displayed.

The user has the option of counting from zero to the defined value or from the defined value to zero.

Here is a conveyor carrying parts to be packed. After every 5 parts the conveyor stops and the operator packs the parts. Then he presses the button again to reset the counter and thus restart the conveyor.

See Help: double-click on the block and click on ?.

Preset Hour Counter: Measures the duration of the input state at 1. After a preset duration, the output changes state. This block can, for example, be used as an alert on a machine for maintenance purposes.

This is the principle used to warn of the need for maintenance. Every 30 hours of operation, to change a filter on the machine, for example.

See Help: double-click on the block and click on ?.

3. HMI/Communication Functions:

Display on the LCD Screen: Displays text or an integer on the LCD screen on the controller front panel. For example, you can display a decimal derived from an integer. For more details, please refer to the example.

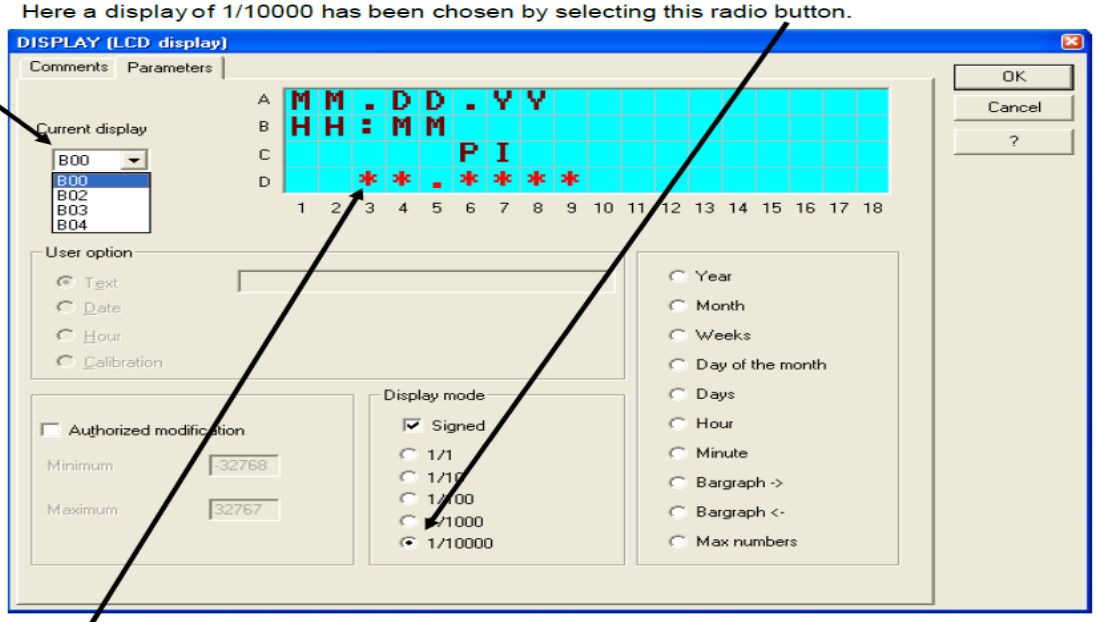
The Display function is used to display text, variables, and the time and date on the Millenium display.

You can have up to 8 Display blocks active. They are placed on the screen based on the numbering. If you have more than 8 Display blocks active at the same time only the first 8 blocks will be displayed.

The function window is used for displaying the variable with decimal places and for editing the text.

In this example 4 display blocks are used:

- Here B00 is selected, which displays the content of the variable B01;
- Here a display of 1/10000 has been chosen by selecting this radio button.



- You can place the text or data on the exact position that you require just by clicking on that position on the grid.

Note: Calibration compensates for drifting of the Millenium clock. If the calibration button is activated the display will allow modification of this value. The unit is in seconds per week.

LCD.pm3

This is an example of using the controller's LCD display. The date, time, text and a decimal value(from an integer) are displayed on it.

See Help: double-click on the block and click on ?.

TEXT function: Displays text and/or numerical values on the LCD screen on the controller's front panel. You can connect up to 4 numerical values to the same block and locate them, along with text, on the screen. You can also display the date and time and the calibration value for the clock drift.

This function doesn't allow for any special formatting on the numerical values. If that is required then the Display block has to be used.

You can only have **one** Text block active at the same time, since it covers the entire screen. If you have more than one active only the block with the highest number will be displayed.

TEXT.pm3

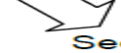
Example of use of the Text function. The date, time, text, and a couple of numerical values are displayed on two different screens.

See Help: double-click on the block and click on ?.

LCD backlighting: This block is processed like an output. When it is active it activates the backlighting of the display

See Help: double-click on the block and click on **?**.

Buttons: You can use the buttons on the front panel of the Millenium in your application: A, B, ESC, OK, + and -.



ABEscOK.pm3

This example shows how to use the front panel keys on a program.

See Help: double-click on the block and click on ?.

SL In, SL Out: Used for transmitting and receiving data via a serial link through the programming port of the controller. You have 24 input addresses and 24 output addresses. The SL In Protected block is used when you want to protect the data in case of a power failure.

SLIN-SLOUT.pm3

Use of SLIN and SLOUT functions to interface with an HMI.

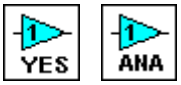
To set the address range on the SL In and SL Out functions, simply double-click on the block or right-click and select the parameter-setting window. See Help: double-click on the block and click on ?.

4. Programming Functions:

Constants: You can use constants to set values on certain function blocks. There are both numerical and discrete constants

NUM.pm3

See Help: double-click on the block and click on ?.



YES, YES NUM: These function blocks are very useful on macros. They allow for an input to be connected to several blocks inside a macro. The YES block is for discrete signals and the YES NUM block is for numerical signals.

Memory function: Used for saving a numerical value. When the Archive input changes from 0 to 1 the value at the output is replaced by the value on the input. You can select whether you want to keep this value or not in case of a power failure.

See Help: double-click on the block and click on **?**

Calculation Functions:

Gain: Allows the use of a scale factor and is applicable to all analog data.

Example: This is a program which uses a counter, a comparator, a gain and the counter read-out display. An alarm is activated after the sensor has been passed 20 times.

In this example, an alarm is activated after the sensor has been passed 20 times. The number of impulses is divided by 5.

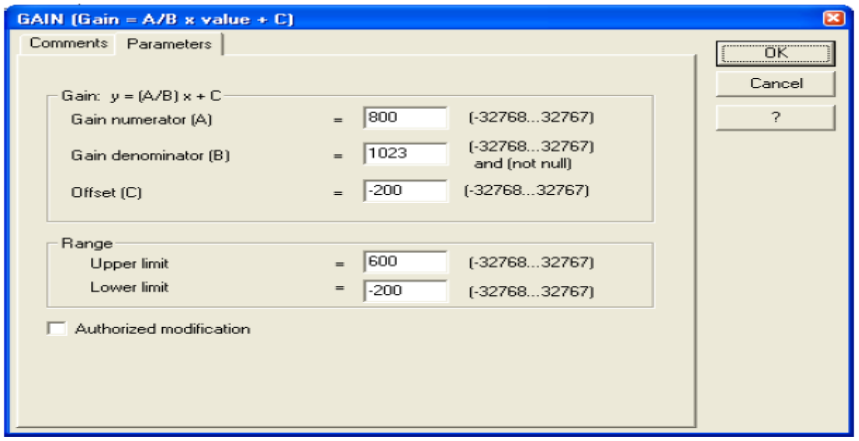
See Help: double-click on the block and click on ?.

Example of a gain function used for displaying the temperature measured by a PT 100 temperature probe between -20 and + 60°:

- The measurement scale A = 80 (-20 to +60); these 80°C are divided into 1024 points.
- The offset corresponds to -20°C; the limit display values would be 60 and -20.

Range: 800
Resolution: 0 to 1023
Min. Value: -200

Max. Temp: 60.0 °C
Min. Temp: -20.0 °C



This example combines some blocks that have already been introduced in order to control temperature and display it on the screen. The Gain function is used to convert the data provided by the sensor into a more useful format.

In this example, the chosen display is 1/100°C and therefore all the parameters of the Schmitt Trigger function and the Gain function should be multiplied by 100 with the exception of the 1023 denominator constant.

GAIN_1.pm3

Arithmetic functions: Perform arithmetic operations on numerical integers. You can do additions, subtractions, multiplications and divisions.

If an input is not connected it is set to 0 (ADD-SUB) or to 1 (MUL-DIV).

See Help: double-click on the block and click on **?**

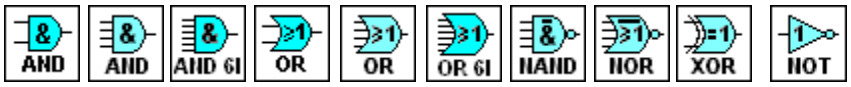
Word-Bit conversions: Allow to break down an integer into individual bits or to take the individual bits and form an integer with them. Each integer consists of 16 bits.

These function blocks are especially useful when there is communication (Modbus, Serial Link) involved in the application since the data exchanged are words and sometimes you need to address individual bits.

See Help: double-click on the block and click on **?**

5. Logic Functions:

Logic Gates:



Logic gates can be used to construct logic (combinational) circuits.

The available functions are NOT, AND, OR, NAND, NOR, and XOR. They are available on 2, 4 and 6 input versions, depending on the function. Only the inputs connected are taken into account.

ee Help: double-click on the block and click on **?**

Boolean functions: On these functions the output(s) react(s) according to the truth table described in the parameters. There is one version with

4 inputs and 1 output and another version with 6 inputs and 2 outputs. Only the inputs connected are taken into account.

BOOLEAN.pm3

See Help: double-click on the block and click on **?**

6. SFC/Grafcet Functions:

SFC functions are similar to Grafcet language. The principle is simple, since it involves sequential programming, with steps succeeding one another surrounded by transitions. When a step is active, wait for the next transition to become active in order to go to the next step.

SFC.pm3

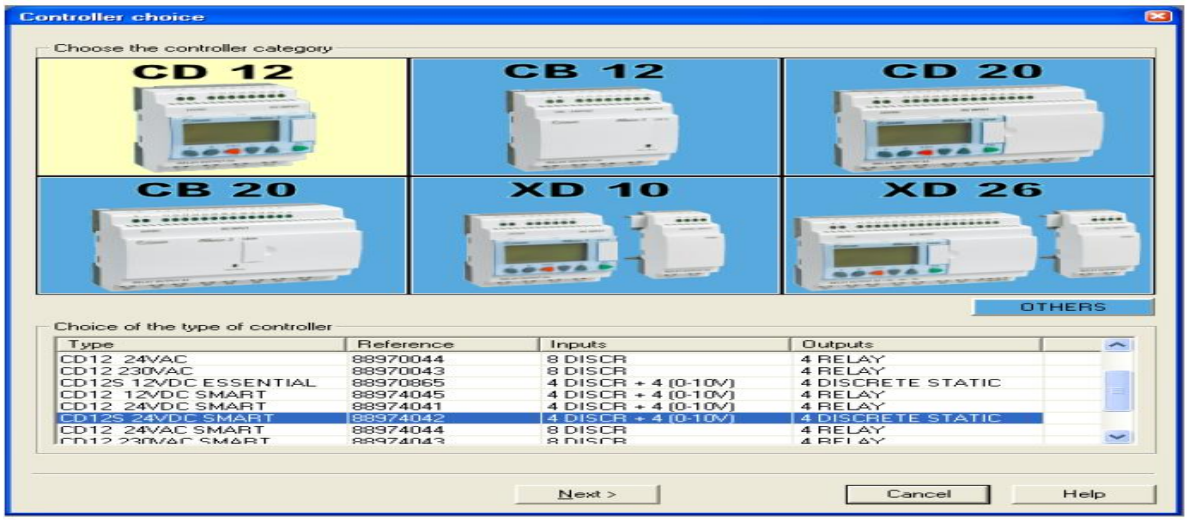
This example shows the sequence of a program using SFC functions.

See Help: double-click on the block and click on ?.

STARTING AN APPLICATION

1. The Edit window

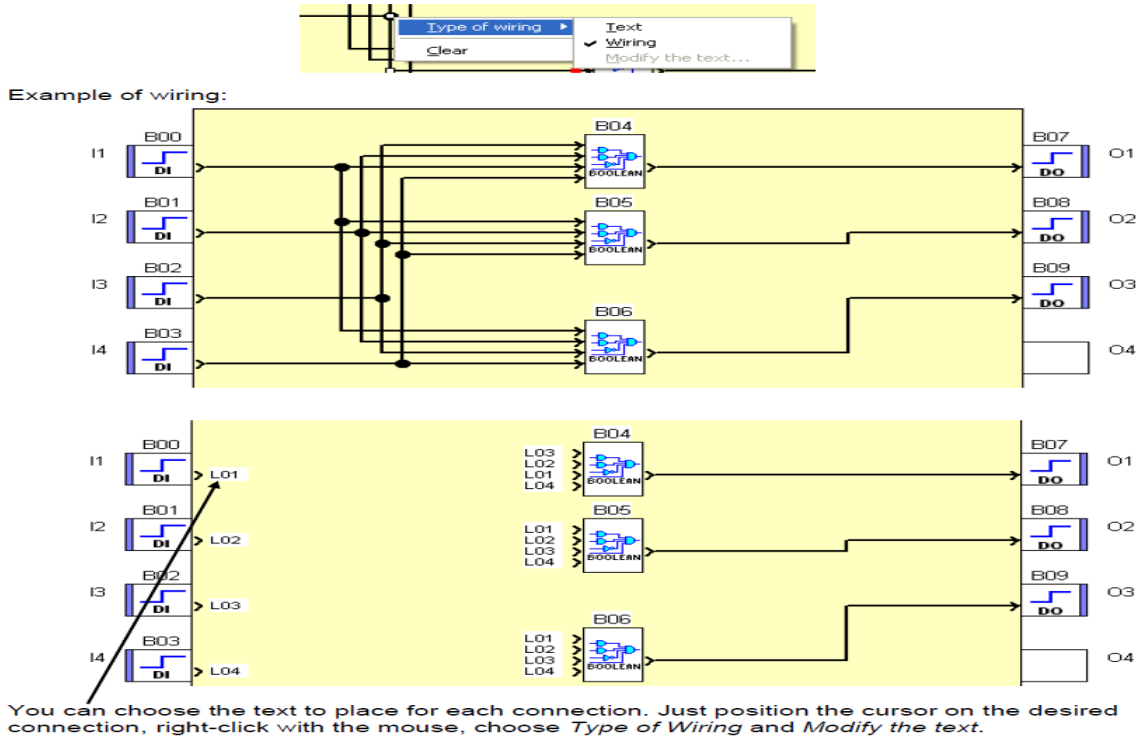
Select New File and click the of Millenium that you have chosen. Select the part number corresponding to the controller.



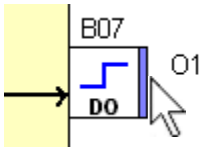
The edit window opens with a blank wiring sheet and you are ready to build your application. The part number of the selected Millenium controller then appears at the top of the wiring sheet.

Blocks are positioned by clicking on the block, holding down and dragging it onto the programming page. Links between blocks are created directly by selecting block inputs and outputs. In the wiring mode tool, you can choose wire as the wiring type, and you will see the links between the various elements. If you choose text mode, the links will be marked but they will no longer be visible.

To change this parameter, right-click on a link and select the wiring type: wire or text.



When you want to move an input or output which is already assigned to an element you can move it by using the handle on the side.



It is possible to change an input or output type. This option does not affect operation.

If you want to change an input or output type, simply double-click on the icon and choose an alias. On the wiring page, you can add a comment and drawings. To do this, you can use the draw toolbar and also the draw menu bar.

To change the line thickness, the line color or the background color, you need to select the

element and click on the icon associated with the desired action in the toolbar.

2. **Editing your program: Edit mode**

At the top of the wiring page you can see these three buttons.

Displays the selected millennium 3

1. By clicking title Author, you can write in the project name, date and author.
2. By clicking Program, you can select the application **cycle duration** (10ms by default).
3. Then you can choose the date format.
4. If you are using PWM outputs, you can select the frequency of all the PWM outputs (by default 1806.37 Hz).

To build your application:

Select the input blocks and place them on the input terminals, select the output blocks and place them on the output terminals.

Select the function blocks, create the wiring between the various points. Double-click on the functions in order to set the parameters.

Each function block is numbered in the order of placing the blocks on the programming page. Deleting blocks results in a break in the numbering. To renumber, select the blocks then **Tools**, **Renumber functions.**

In text mode wiring, each link is numbered in the order of placing the wiring on the wiring page. Deleting links results in a break in the numbering. To renumber, select the links then **Tools**, **Renumber links.**

By selecting a number of blocks, you can align them according to the icon on the Draw bar. Align left, right, center, etc.

Supervision:

Select **window** then **Supervision**. Simply drag the inputs/outputs and function blocks of your choice from the wiring page to the supervision window. You can illustrate your application using the draw tools. You can also choose a .BMP background image by right-clicking in the supervision window; Modify background, Bitmap.

This window explicitly displays the elements you have dragged from the wiring page in their own environment. When you change to simulation or monitoring mode, the inputs and the outputs are updated; it is also possible to force an input in the same way as with the edit window. Here is an example of using supervision mode:

SUPERV.pm3

Import

You have the option of recovering all or part of the programming page of an existing file. To import a wiring scheme, you should already have opened a file. First select File, then Import. Next choose the file to be imported. When importing a wiring scheme, you will see that the previously opened file stays open. You can therefore drag a selection from the edit window of the imported wiring scheme to the edit window of the previous wiring scheme.

3. Testing your program: Simulation mode

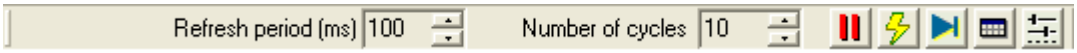
Our software also lets you fully test your program before implementing it. Simply select **S** or simulation mode. Simulation on discrete or analog inputs can be temporary or permanent. Force the input or output by clicking on the link or on the input or output pin. It is not necessary for the controller to be connected to the PC to perform simulation.

Front Panel Display

In simulation mode, click on **Window** then on **Front Panel**. The keys illustrated on the front panel are activated by clicking and holding down.

Simulation Mode Parameters

The monitoring/simulation bar is used to change the number of cycles executed at each simulation stage, and is similar to a time multiplier. Moreover, the refresh period is the frequency at which the output and parameter values are updated in the application window.

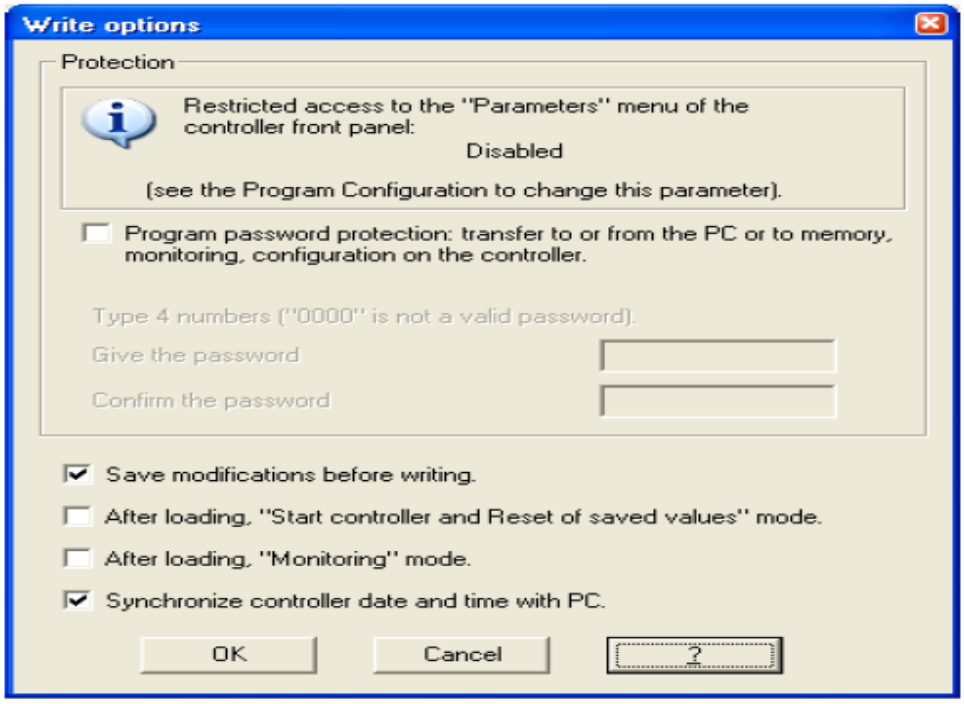


Writing to the Controller And Running

Once your application has been debugged and verified, you can transfer it to the Millenium controller. The Write to the controller function translates the program into data that can be loaded into the controller and transfers it from the PC to the controller.

To write data to the controller, it must be in STOP mode. After the program has been stopped go into the Controller menu, click on Write to the controller.

This option will open the Compilation Results window. If the compilation is successful then following windows appears:

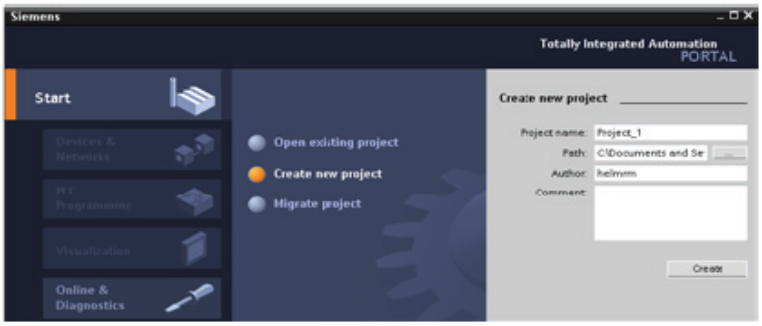


INTRODUCTION OF SIMENS PLC

GETTING STARTED WITH SIMENS PLC:

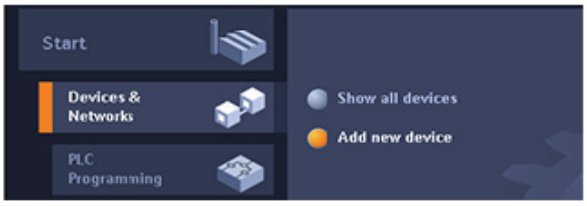
STEP 1: Create a project

Working with STEP 7 is easy! See how quickly you can get started with creating a project.



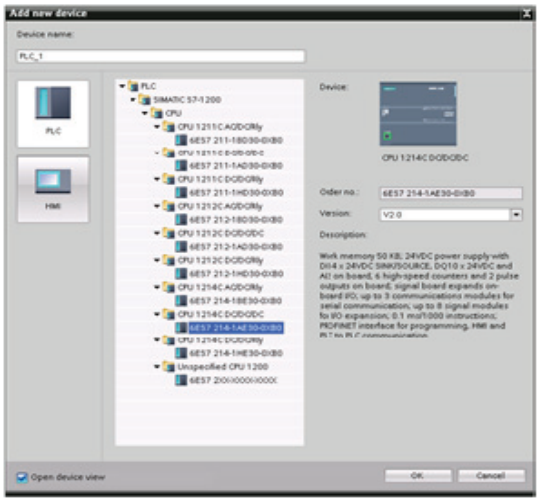
In the Start portal, click the "Create new project" task.

Enter a project name and click the "Create" button.



After creating the project, select the Devices & Networks portal.

Click the "Add new device" task.



Select the CPU to add to the project:

1. In the "Add new device" dialog, click the "SIMATIC PLC" button.
2. Select a CPU from the list.
3. To add the selected CPU to the project, click the "Add" button.

Note that the "Open device view" option is selected. Clicking "Add" with this option selected opens the "Device configuration" of the Project view.

The Device view displays the CPU that you added.

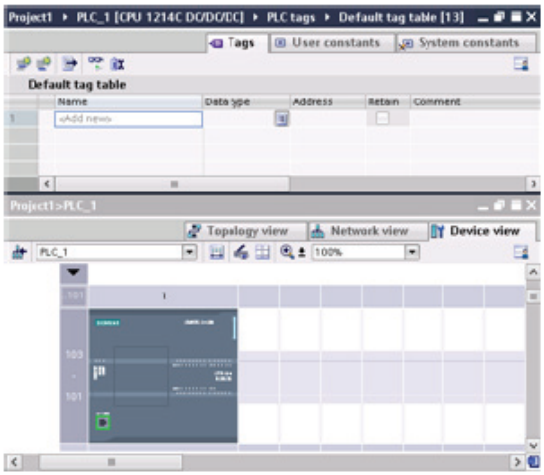
STEP 2: Create Tags for The I/O Of The CPU

"PLC tags" are the symbolic names for I/O and addresses. After you create a PLC tag, STEP 7 stores the tag in a tag table. All of the editors in your project (such as the program editor, the device editor, the visualization editor, and the watch table editor) can access the tag table.

With the device editor open, open a tag table.

You can see the open editors displayed in the editor bar.

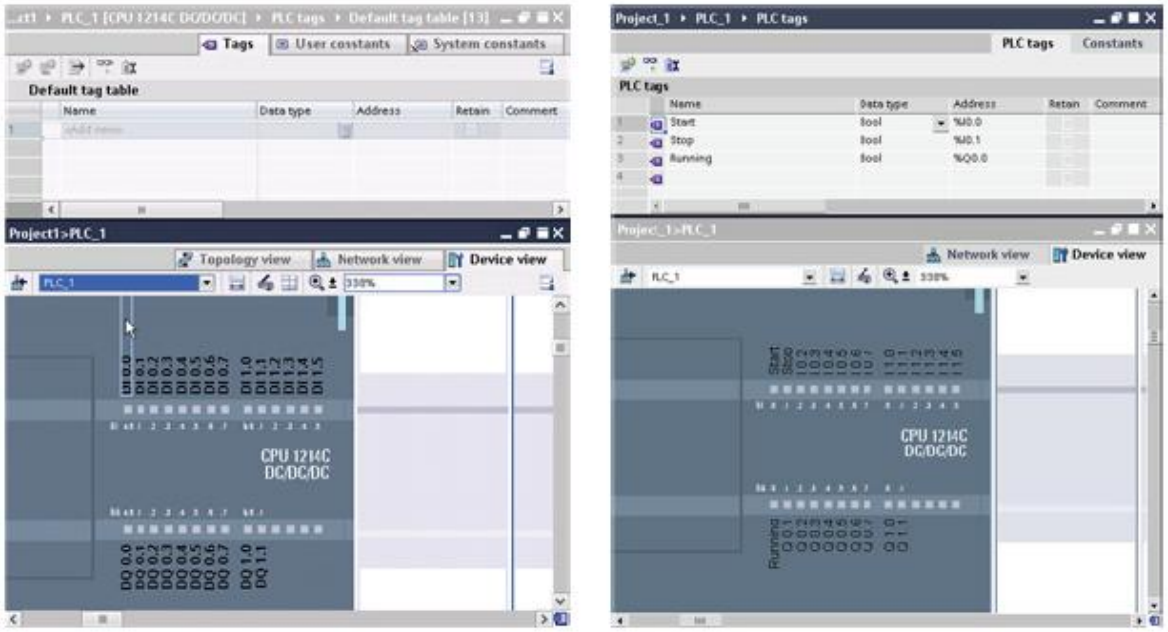
In the tool bar, click the "Split editor space horizontally" button.



STEP 7 displays both the tag table and the device editor together.

Zoom the device configuration to over 200% so that the I/O points of the CPU are legible and selectable. Drag the inputs and outputs from the CPU to the tag table:

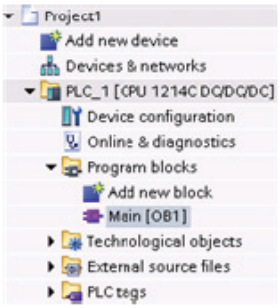
1. Select I0.0 and drag it to the first row of the tag table.
2. Change the tag name from "I0.0" to "Start".
3. Drag I0.1 to the tag table and change the name to "Stop".
4. Drag Q0.0 (on the bottom of the CPU) to the tag table and change the name to "Running".



With the tags entered into the PLC tag table, the tags are available to your user program.

STEP 3: Create A Simple Network In Your User Program

Your program code consists of instructions that the CPU executes in sequence. For this example, use ladder logic (LAD) to create the program code. The LAD program is a sequence of networks that resemble the rungs of a ladder.

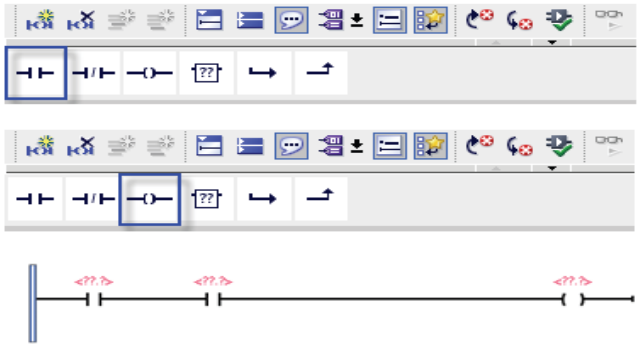


To open the program editor, follow these steps:

1. Expand the "Program blocks" folder in the Project tree to display the "Main [OB1]" block.
2. Double-click the "Main [OB1]" block.

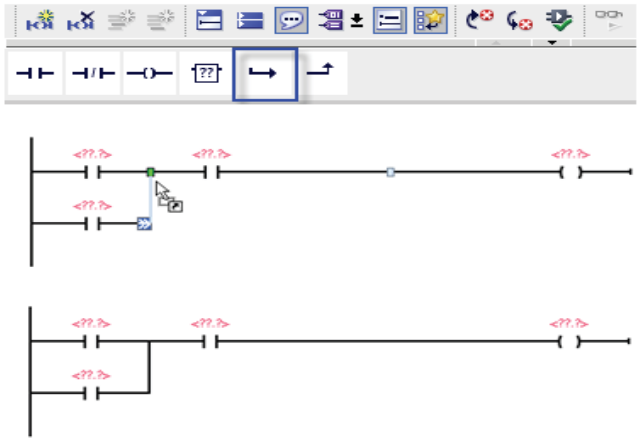
The program editor opens the program block (OB1).

Use the buttons on the "Favorites" to insert contacts and coils onto the network.



1. Click the "Normally open contact" button on the "Favorites" to add a contact to the network.
2. For this example, add second contact.
3. Click the "Output coil" button to insert a coil.

The "Favorites" also provides a button for creating a branch

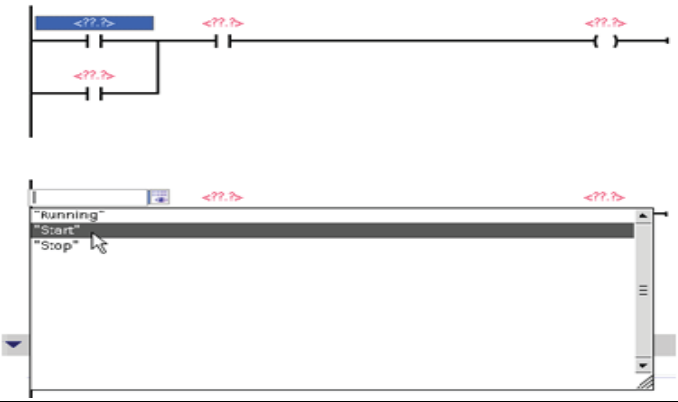


1. Select the left rail to select the rail for the branch.
2. Click the "Open branch" icon to add a branch to the rail of the network.
3. Insert another normally open contact to the open branch.
4. Drag the double-headed arrow to a connection point (the green square on the rung) between the two contacts on the first rung.

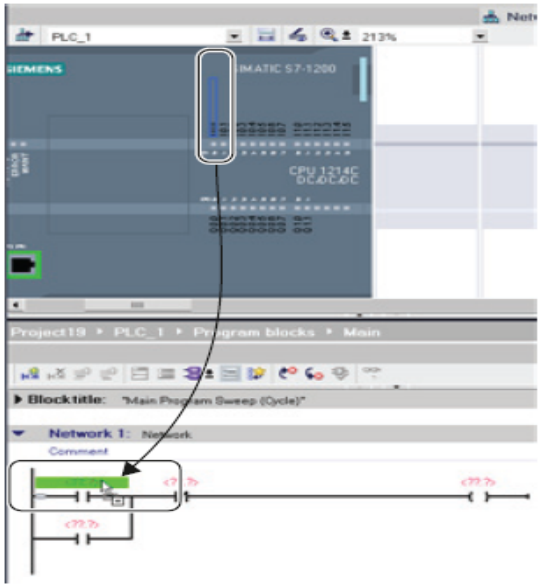
To save the project, click the "Save project" button in the toolbar. Notice that you do not have to finish editing the rung before saving. You can now associate the tag names with these instructions.

STEP 4: Use The PLC Tags In The Tag Table For Addressing The Instructions

Using the tag table, you can quickly enter the PLC tags for the addresses of the contacts and coils.



1. Double-click the default address <*??.?*> above the first normally open contact.
2. Click the selector icon to the right of the address to open the tags in the tag table.
3. From the drop-down list, select "Start" for the first contact.
4. For the second contact, repeat the preceding steps and select the tag "Stop".
5. For the coil and the latching contact, select the tag "Running".



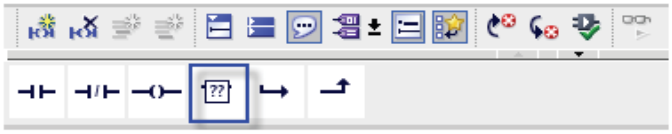
You can also drag the I/O addresses directly from the CPU. Simply split the work area of the Project view (Page 30).

You must zoom the CPU to over 200% in order to select the I/O points.

You can drag the I/O on the CPU in the "Device configuration" to the LAD instruction in the program editor to create not only the address for the instruction, but also to create an entry in the PLC tag table.

STEP 5: Add A "Box" Instruction

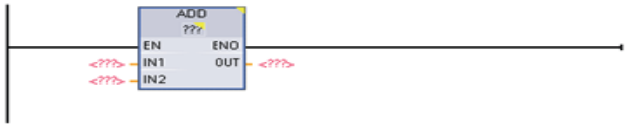
The program editor features a generic "box" instruction. After inserting this box instruction, you then select the type of instruction, such as an ADD instruction, from a drop-down list.



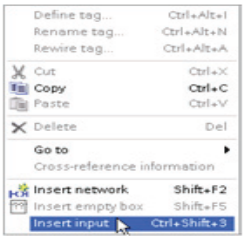
Click the generic "box" instruction in the "Favorites" tool bar.

The generic "box" instruction supports a variety of instructions. For this example, create an ADD instruction:

1. Click the yellow corner of the box instruction to display the drop-down list of instructions.
2. Scroll down the list and select the ADD instruction.
3. Click the yellow corner by the "?" to select the data type for the inputs and output.



You can now enter the tags (or memory addresses) for the values to use with the ADD instruction.



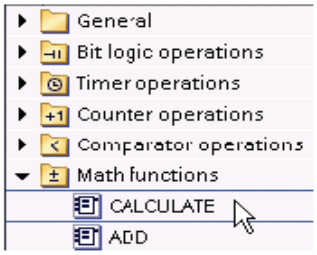
You can also create additional inputs for certain instructions:

1. Click one of the inputs inside the box.
2. Right-click to display the context menu and select the "Insert input" command.

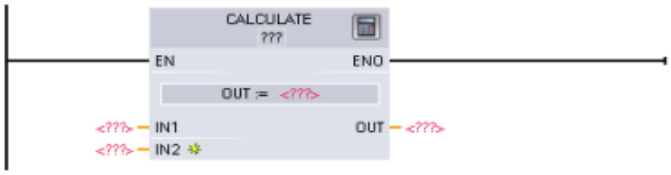
The ADD instruction now uses three inputs.

STEP 7: Use The CALCULATE Instruction For A Complex Mathematical Equation

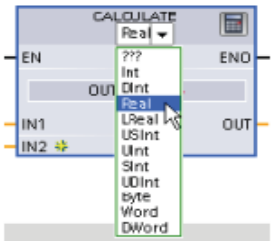
The Calculate instruction lets you create a math function that operates on multiple input parameters to produce the result, according to the equation that you define.



In the Basic instruction tree, expand the Math functions folder. Double-click the Calculate instruction to insert the instruction into your user program.



The unconfigured Calculate instruction provides two input parameters and an output parameter.



Click the "???" and select the data types for the input and output parameters. (The input and output parameters must all be the same data type.)

For this example, select the "Real" data type.

Click the "Edit equation" icon to enter the equation.

For this example, enter the following equation for scaling a raw analog value. (The "In" and "Out" designations correspond to the parameters of the Calculate instruction.)

$Out_{value} = (Out_{high} - Out_{low}) / (In_{high} - In_{low}) * (In_{value} - In_{low}) + Out_{low}$

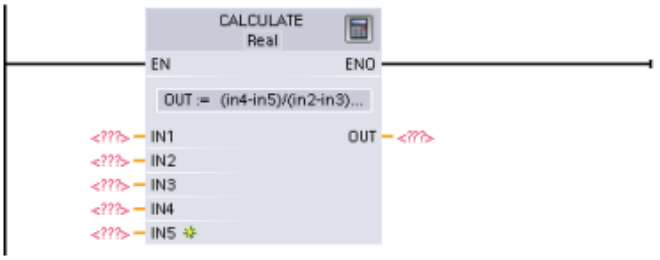
Out = (in4 - In5) / (In2 - In3) * (In1 - In3) + In5

Where:	Out_{value}	(Out)	Scaled output value
	In_{value}	(In1)	Analog input value
	In_{high}	(In2)	Upper limit for the scaled input value
	In_{low}	(In3)	Lower limit for the scaled input value
	Out_{high}	(In4)	Upper limit for the scaled output value
	Out_{low}	(In5)	Lower limit for the scaled ouput value

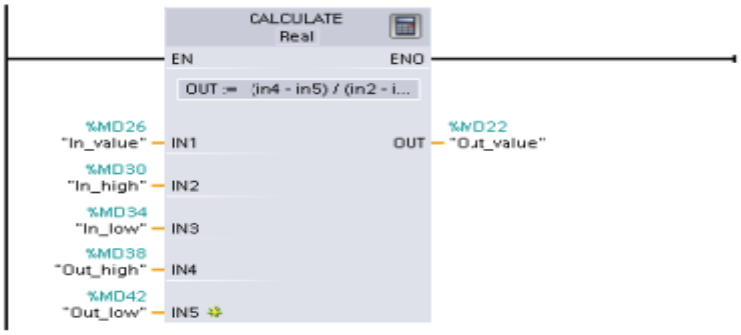
In the "Edit Calculate" box, enter the equation with the parameter names:

OUT = (in4 - In5) / (In2 - In3) * (In1 - In3) + In5

When you click "OK", the Calculate instruction creates the inputs required for the instruction.

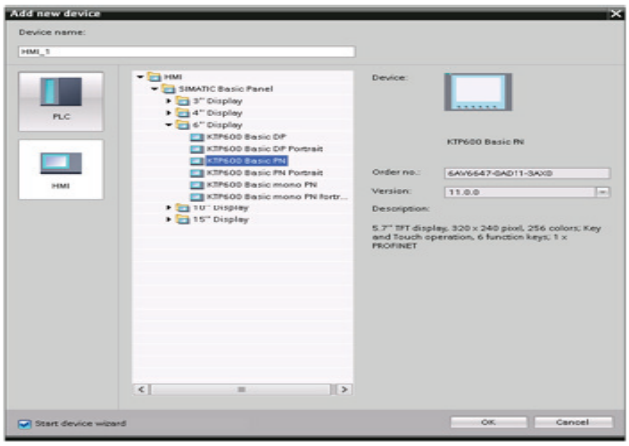


Enter the tag names for the values that correspond to the parameters.



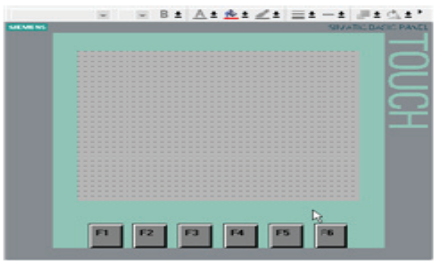
STEP 7: Add An HMI Device To The Project

Adding an HMI device to your project is easy!



1. Double-click the "Add new device" icon.
2. Click the "SIMATIC HMI" button in the Add new device" dialog.
3. Select the specific HMI device from the list.

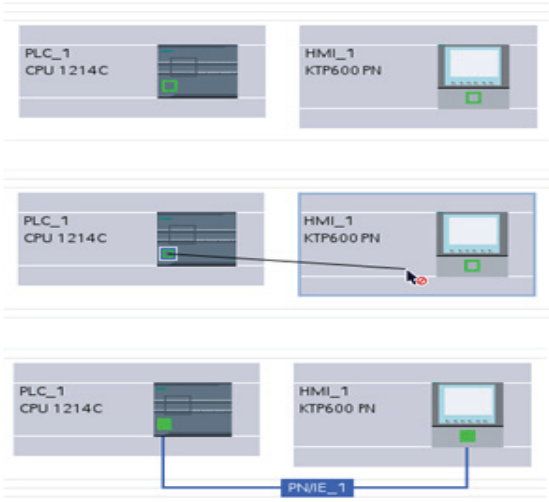
   You can choose to run the HMI wizard to help you configure the screens for the HMI device.
4. Click "OK" to add the HMI device to your project.



The HMI device is added to the project.
STEP 7 provides an HMI wizard that helps you configure all of the screens and structure for your HMI device.

If you do not run the HMI wizard, STEP 7 creates a simple default HMI screen.

STEP 8: Create A Network Connection Between The CPU And HMI Device

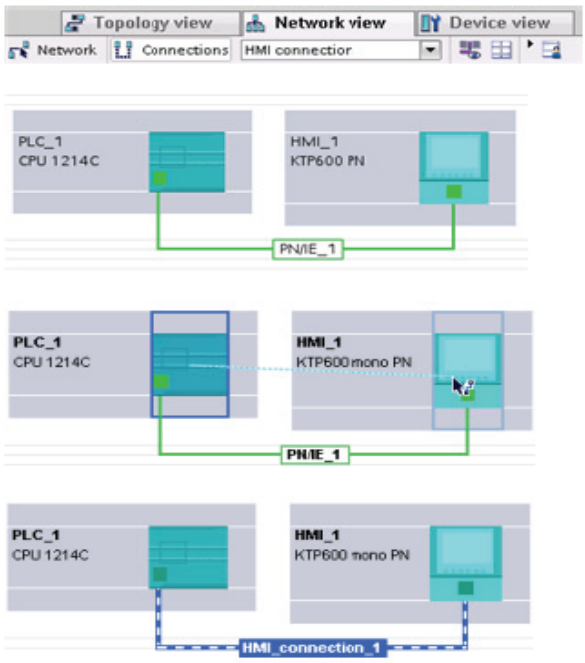


Creating a network is easy!

- Go to "Devices and Networks" and select the Network view to display the CPU and HMI device.
- To create a PROFINET network, drag a line from the green box (Ethernet port) on one device to the green box on the other device.

A network connection is created for the two devices.

STEP 9: Create An HMI Connection To Share Tags



By creating an HMI connection between the two devices, you can easily share the tags between the two devices.

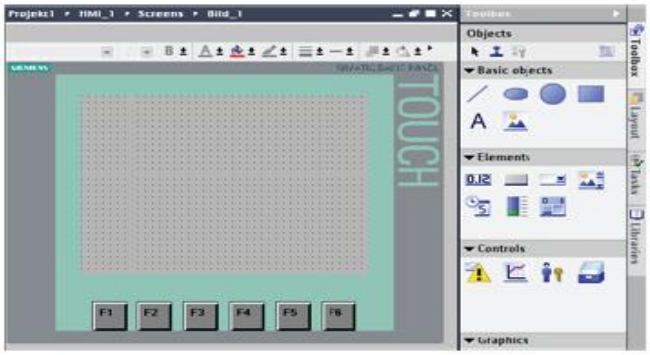
- With the network connection selected, click the "Connections" button and select "HMI connection" from the drop-down list.
- The HMI connection turns the two devices blue.
- Select the CPU device and drag the line to the HMI device.
- The HMI connection allows you to configure the HMI tags by selecting a list of PLC tags.

You can use other options for creating an HMI connection:

- Dragging a PLC tag from the PLC tag table, the program editor or the device configuration editor to the HMI screen editor automatically creates an HMI connection.
- Using the HMI wizard to browse for the PLC automatically creates the HMI connection.

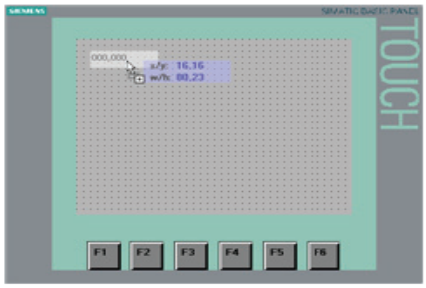
STEP 10: Create An HMI Screen

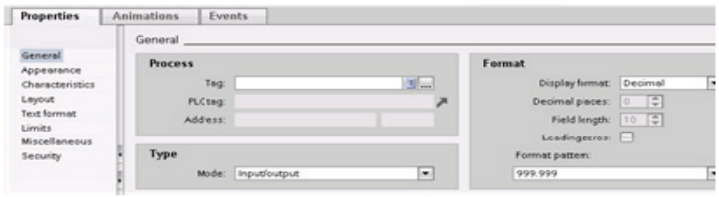
Even if you do not utilize the HMI wizard, configuring an HMI screen is easy.



STEP 7 provides a standard set of libraries for inserting basic shapes, interactive elements, and even standard graphics.

To add an element, simply drag and drop one of the elements onto the screen. Use the properties for the element (in the Inspector window) to configure the appearance and behavior of the element.

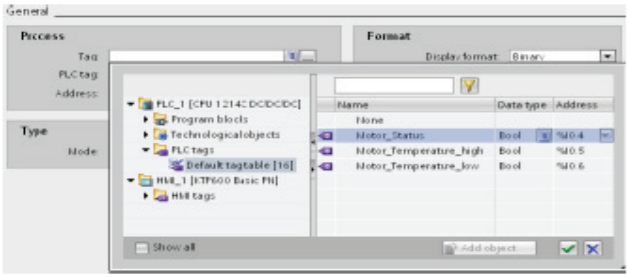




You can also create elements on your screen by dragging and dropping PLC tags either from the Project tree or the program editor to the HMI screen. The PLC tag becomes an element on the screen. You can then use the properties to change the parameters for this element.

STEP 11: Select A PLC Tag For The HMI Element

After you create the element on your screen, use the properties of the element to assign a PLC tag to the element. Click the selector button by the tag field to display the PLC tags of the CPU.



You can also drag and drop PLC tags from the Project tree to the HMI screen. Display the PLC tags in the "Details" view of the project tree and then drag the tag to the HMI screen.