

Department of Engineering and Design



LONDON SOUTH BANK
UNIVERSITY

Control Lab Room T405

Microprocessor and PLC Applications

Unit DCL-2-246

Workshop Manual

Supervisor: Mr.S.Mondal

Technical Support & Authors:

Peter Adams & Mehdi Zahir



2011

<u>Contents</u>	<u>Page</u>
Introduction to programmable logic controllers PLCs	4
Ladder logic symbols	5
Ladder logic programming	5
Rung example 1 Series switches	5
Rung example 2 Parallel switches	6
Rung example 3. Contactor	6
Rung example 4. Bi-directional motor control	7
Rung example 5. Crane control	9
Internal input and output switches	9
Ladder logic and machine code	9
Introduction to the F1616BA PLC	10
PLC Memory Structure	11
Transferring Programs to the PLC	12
Precautions when Handling the PLC	12
List of Simulation Exercises	13
List of Application Exercises	13
i-Trilogi Software	14
Ladder logic editing tips	14-15
Complex series and parallel connections	16-17
Simulation exercise 1. Simple input output	18-22
Simulation exercise 2. AND function inputs	22
Simulation exercise 3. OR function inputs	23
Simulation exercise 4. TOGGLE output	23-24
Simulation exercise 5. Set-Reset output	25
Simulation exercise 6. Clocked output	26
Simulation exercise 7. Contactor	27-28
Simulation exercise 8. Remote Control Contactor	28

<u>Contents</u>	<u>Page</u>
Simulation Exercise 9. Contactor with a timeout	29
Simulation exercise 10. Three Phase Induction Motor Soft Starter	30
Simulation exercise 11. Three Phase Induction Motor Star- Delta- Starter	32
Simulation exercise 12. Sequence control	33-34
Ladder Logic Symbol Table	35

Note:

This exercise book is an introductory document. The F1616BA PLC is a versatile controller with numerous functions and capabilities. The exercises in this document are intended to be used as a quick learning guide. This document does not cover teaching in the TBasic language used by the controller. It also does not cover technical and physical details of the PLC. Further details of the latter are provided in the PLC's Technical User Manual.

Warnings:

It is forbidden for any student to attempt to connect any physical device/s to the PLC without prior consultation with the Technicians in charge of the Control Laboratory.

© Copyright London South Bank University 2011

Introduction to Programmable Logic Controllers

In today's industry modern processes need to be automatically controlled and may require a flexible programming system to allow easy re-tasking of their operations. The Programmable Logic Controller or PLC is a standard system based on a simple processor and includes an interface that can be programmed directly.

This makes the PLC a popular system that can directly switch ON/OFF external devices or read signals from sensors. The PLC reduces the amount of complicated interlocking mechanisms and additional wiring, relay switches, use of dedicated timers and counters that may be used in a process. The Programming language for a PLC is called **Ladder Logic** and is based on a limited and simple international set of symbols. This makes it possible for all who are familiar with Ladder Logic to understand the operation of a PLC connected to a system.

Using a PLC, a system's sequence of operation may be changed by simply writing a new program without the need to rewire the system for a new task. A PLC is designed to be very robust and be either used as a stand-alone unit or work with a network of other PLCs exchanging signals. A PLC also has an on board memory which makes storage of data possible. PLCs also contain Registers which hold the status of its inputs and outputs. The Registers can be accessed by the user to create Logical Switches within a ladder logic program which further reduces the amount of external physical switches.

PLCs are used in the following processes:

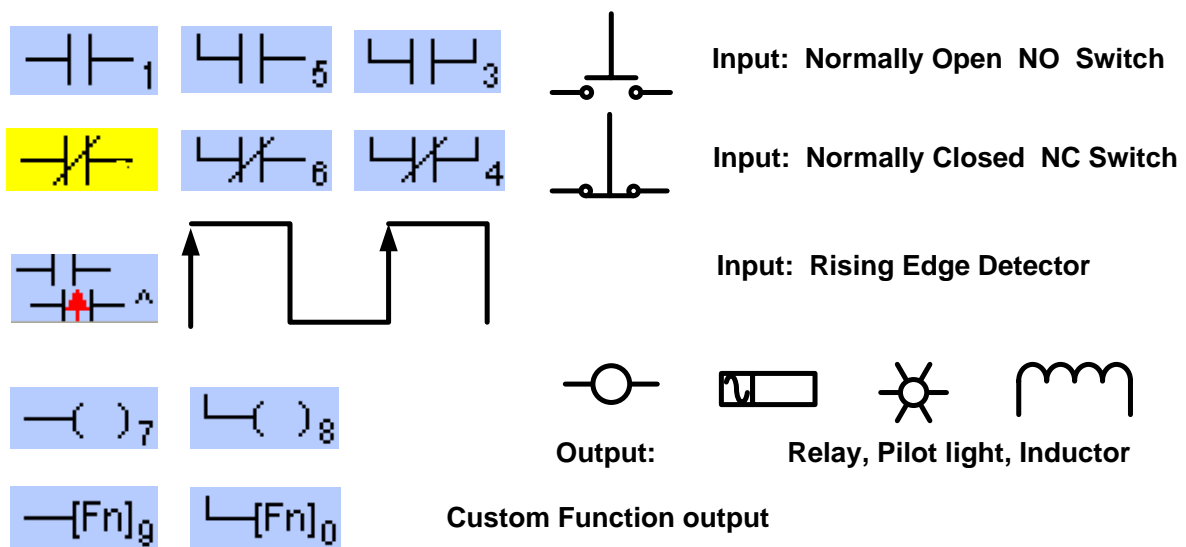
- Drive systems (Motors, Pumps, Fans, Lights, Heaters, Hydraulics etc) or a combination of the latter.
- Safety systems (Emergency stop due to a fault condition/s, Mechanical limit detectors, over heating limits, overflow limits, Gas detectors, Pressure limit detectors etc) or a combination of the latter.
- Human and Machine interface (Operate buttons to Start, Stop, Limit or Time the execution of a process. Sequential process execution etc)
- Robotics systems(synchronisation of a conveyor belt system with a Robot Arm operating in a task such as assembly, pick & place or intelligent sensing systems based on a set of logical occurrences etc.
- Lift systems

The ladder logic symbols resemble the basic electrical circuit components such as:

Inputs: Normally Open switches (NO), Normally Closed switches(NC),
Programmable Custom Inputs(Digital Inputs, A/D Converters)

Outputs: Relays or other type of Programmable Custom Outputs(D/A converters, Pulse Width Modulation PWM).

Ladder Logic Symbols

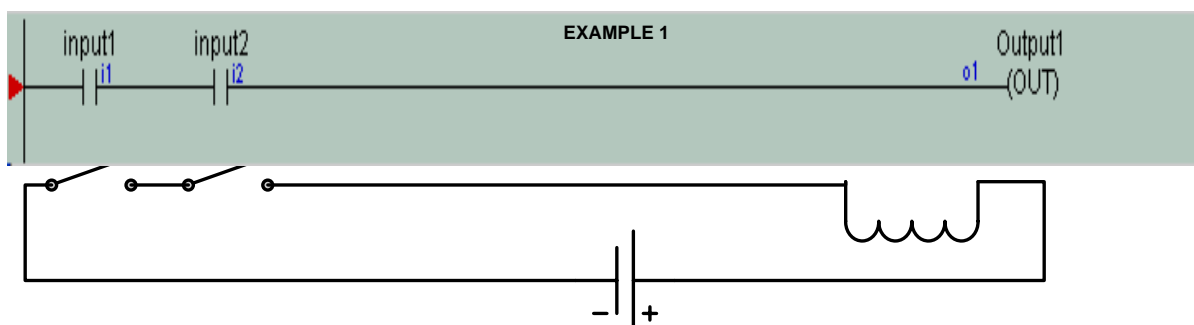


Ladder Logic Programming

In ladder logic programming, a **Rung** is defined as one line of ladder logic symbols connected in series and/or parallel to define the Logical control operation of the external system. Ladder logic programming by tradition, places all inputs on the left hand side and all outputs on the right hand side of a Rung respectively. When a ladder logic program is executed, the processor starts execution from the first Rung and on to the next in a downward scan manner. The first Rung can be used to set initial conditions or initialization commands. The **1st scan** puts the external system in a ready to operate condition. The scanning process loops back to the beginning of the ladder logic avoiding the **1st scan** as this is only executed once.

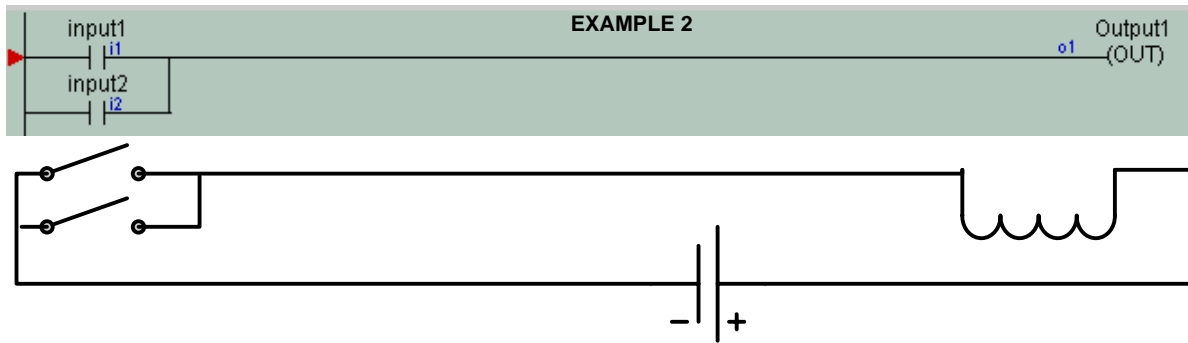
Rung Example 1

A Rung and its equivalent electrical circuit is shown below where a circuit containing two Normally Open(NO) switches are connected in series with a Coil:



The circuit works on the **condition** that both switches must be closed to allow current flow in the circuit.

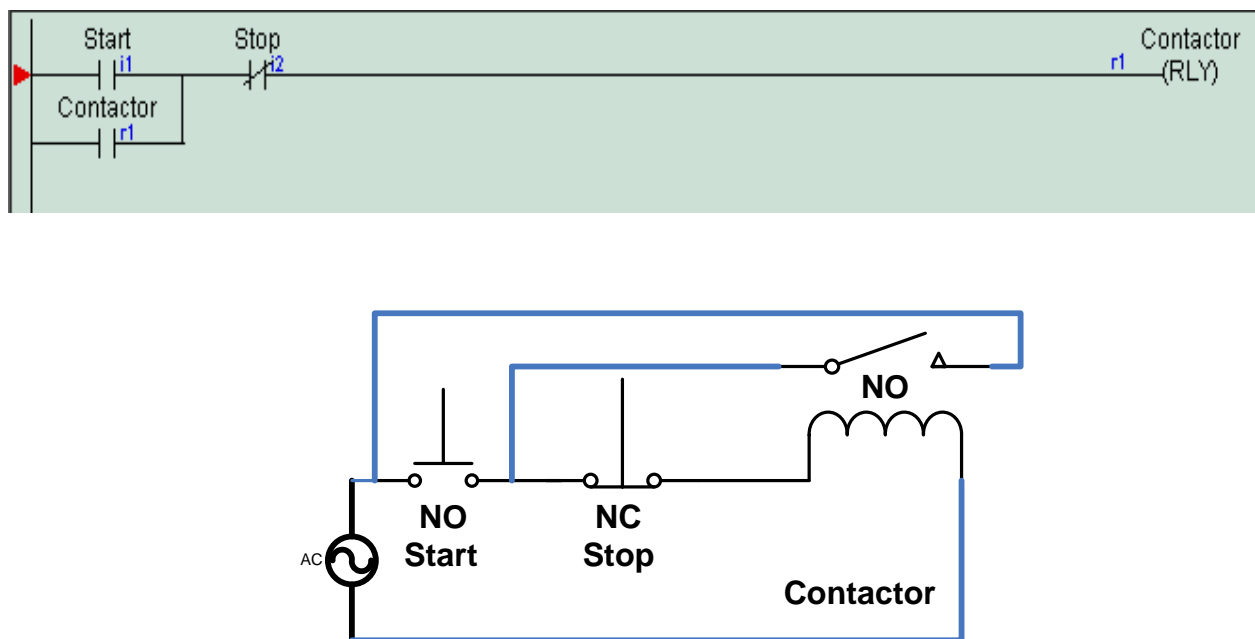
Rung Example 2



In this example a coil is energised by two switches connected in parallel. If either switch is closed, the coil is switched ON.

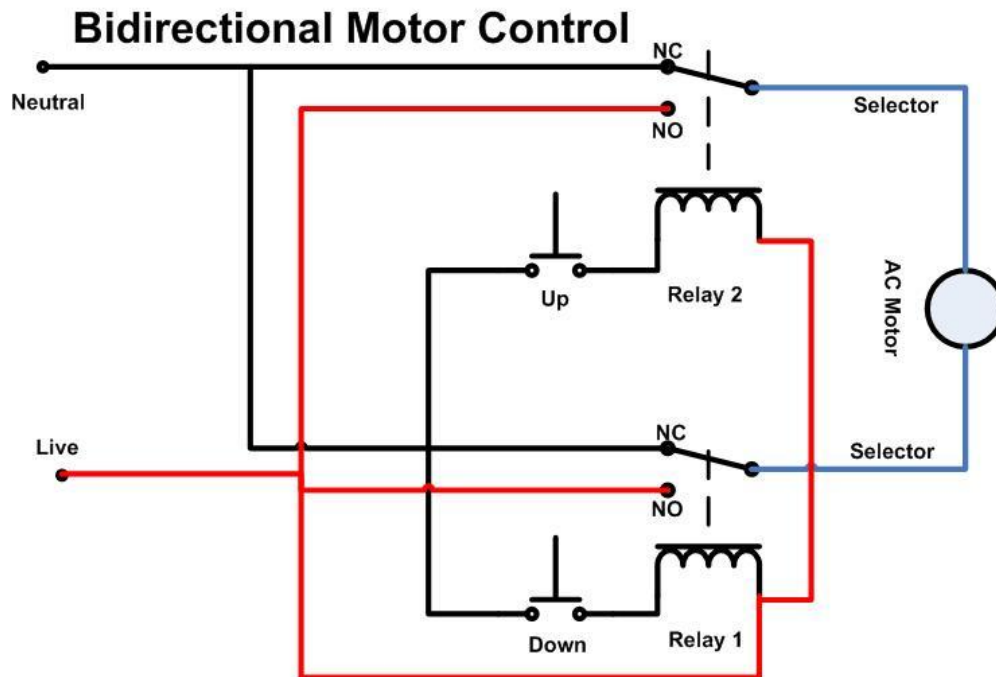
Similarly, a combination of switches connected in series or parallel can define a complex operation, with inputs from various sensors or one PLC Signalling another PLC or device.

Rung Example 3 (Self Locking Relay or Contactor)



This is a very popular circuit used in most Buildings and Industrial processes. The contactor is a Self-Locking relay that is activated with a Normally Open(NO) switch by the user i.e the Start switch. The Locking contact is the NO contact on the relay itself that closes when the relay becomes active. When the user presses and then releases the Start switch the relay remains energized, until such time the Normally Closed(NC) contact(Stop) is pressed to disconnect the power from the relay. This is called a re-settable contactor. The power source can be AC or DC provided the Relay is also AC or DC. There is also another use for the contactor. If there is a power failure in the process, the process will not resume when the power is restored, until the user presses the NO switch again to Start the process. This prevents accidental power surges in processes that need soft starting.

Rung Example 4 (Bi-directional Motor Control)



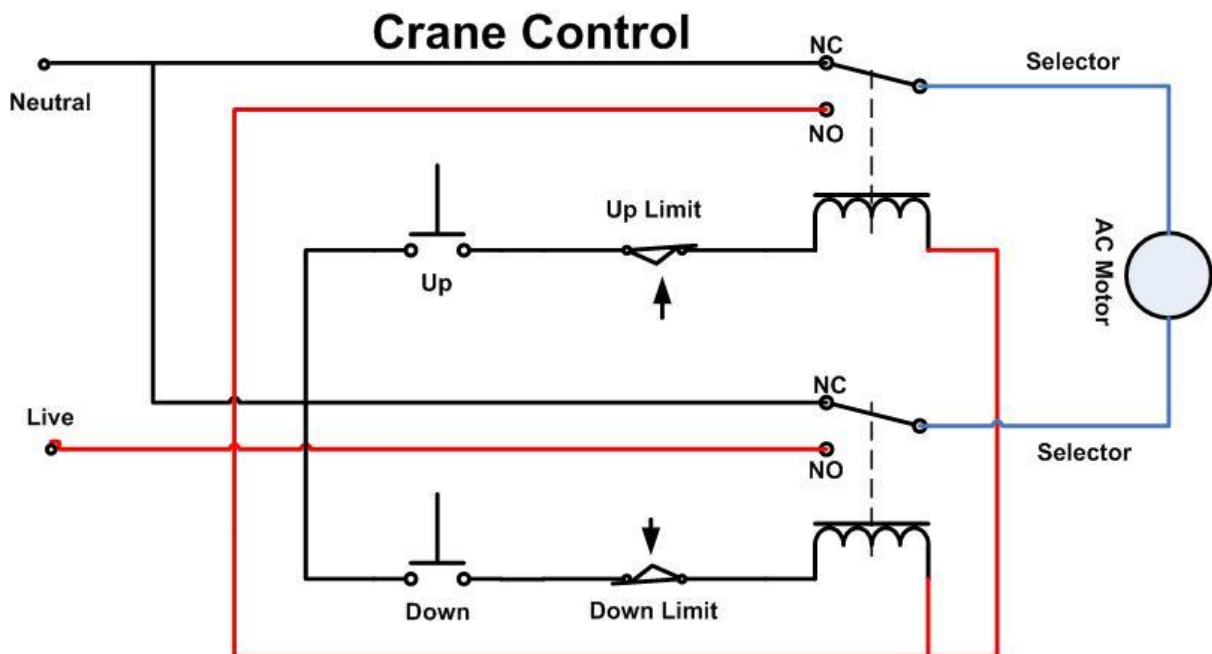
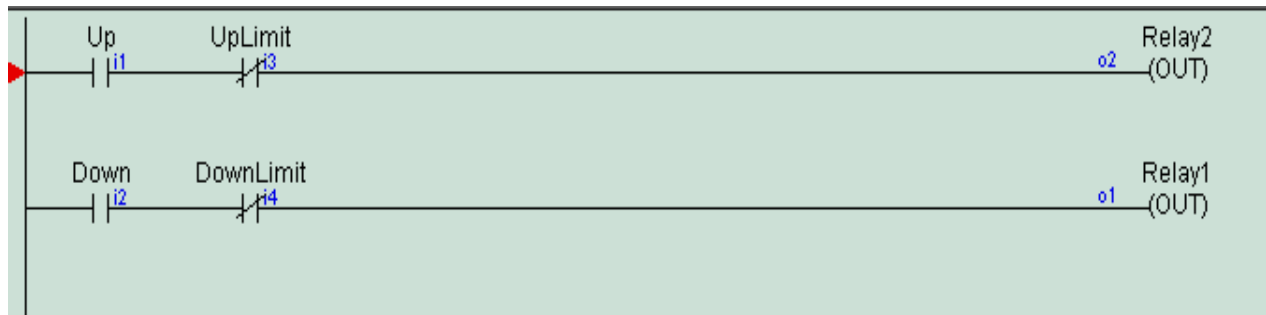
This circuit is comprised of two Relays with their selector contacts connected to a Motor. When the circuit is not active the both Relay selectors are connected to Neutral(N) via the Normally Connected(NC) contacts. If the Up switch is pressed, then Relay 2 is energized and the selector moves down to the NO contact connecting the Live(L) to the Motor. Therefore, the Motor turns, say clockwise.

The Down switch works the same way but reverses the Live and Neutral across the Motor terminals. Therefore, it is possible to control the Motor direction. The circuit above can also work with DC Motors.

Note: if both Up and Down switches are pressed, then both sides of the Motor terminals are connected to Live and so the Motor does not operate.

The ladder logic switches Up and Down control the Relays which are physically wired to operate the Motor.

Rung Example 5 (Crane Control)



This circuit operates the same way as the Bi-directional motor controller in the previous example, but has extra switches in series with the Up and Down switches.

In a Crane system, the operator has 2 control buttons for moving the Crane Up or Down manually. The Crane system is fitted with Up and Down mechanical Limit switches to stop the Crane when it reaches its maximum and minimum height limits. The limit switches are placed in series with the up and down switches, so that the operator can keep either of the control switches pressed, until the Crane reaches it limit safely.

Since the limit now disconnects the Up travel Relay, the selector contact of Relay 2 connects to Neutral again stopping the Motor. When the down switch is pressed the selector of Relay 1 connects to the Live(L), via the NO contact and drives the Motor to travel in the down direction. The up direction limit switch is then released. The down limit- switch also works in the same way, limiting the travel. So the Crane can be operated safely within the Limits.

Internal Input and Output Switches

A PLC is able to hold data about the status of all its inputs and outputs, in its processor registers. This enables the PLC to monitor which Inputs or Outputs are ON or OFF. It is possible to include Logical internal switches(NO or NC) to set conditions within a ladder logic program, without the need for Physical switches. These can be used to indicate the state of an Output internally to some other part the Ladder logic Rung. They can also be used for Timers and Counters to indicate a Timeout Condition or A Countdown condition.

Ladder Logic and Machine Code

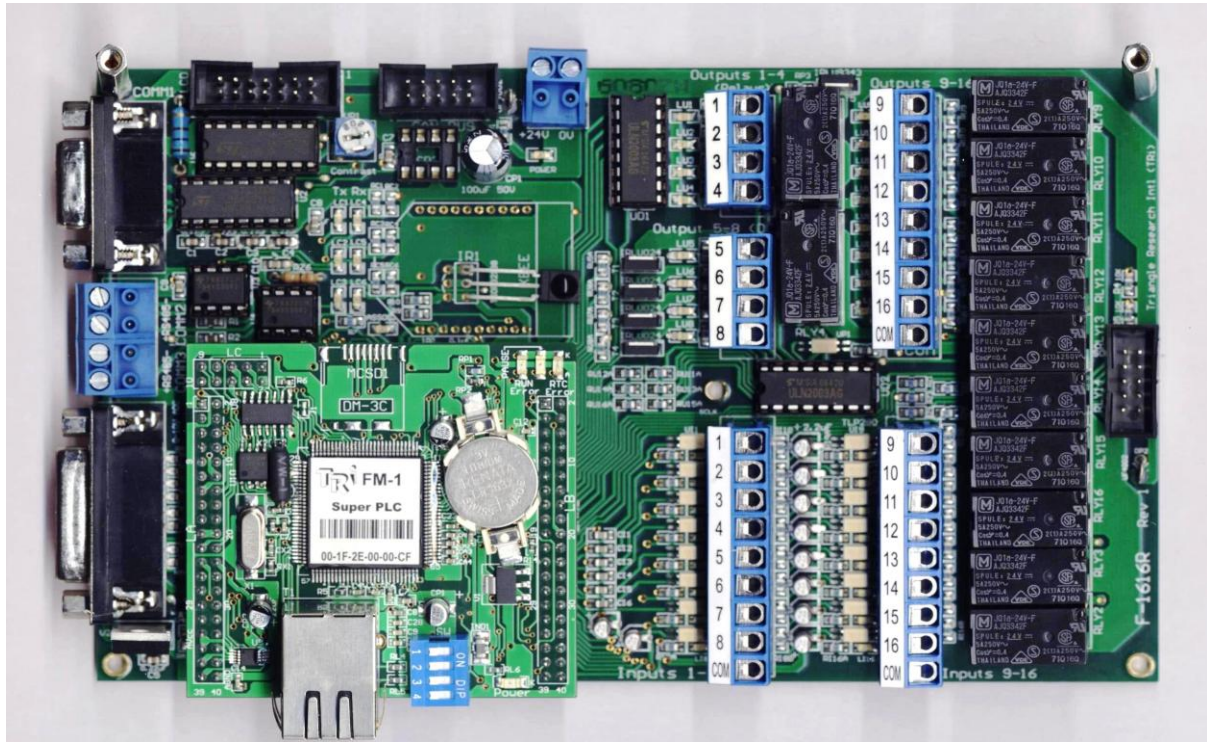
Ladder logic program symbols are created for the benefit of the user who, needs a quick user friendly interface between the Device and the PLC. The ladder logic is internally converted to machine code specific to the processor used. The machine code is then executed to perform the control operation.

Modern PLCs can embed programs in various languages in their ladder logic as Custom Functions. A user can specify a section of code to be executed in a Custom Output and carryout more than just switching the Output ON or OFF. The Output may be a D/A converter outputting an Analogue voltage to control a device (vary the speed of a Motor etc)

Custom Functions can also be used to acquire or input data from Analogue sources such an A/D converter reading a sensor that outputs a voltage indicating temperature or pressure etc. Therefore, the Inputs may be of Analogue or Digital types.

Introduction to F1616BA Super PLC

The F1616BA super PLC is a general purpose industrial grade PLC (Triangle Research) originally designed for Building Automation (BA) and is designed to be used in “Smart Building” applications. These applications involve monitoring and control of energy consumption in buildings.



The PLC has the following features: 32 Digital Input/Outputs (expandable to 256) that interfaces directly to 24 Volt DC or AC power. Four of the D/IO are DC PWM outputs, which can also be used to drive stepper motors. There are 3 high speed encoder inputs, 6 pulse frequency inputs as well as one Infra Red Remote control input. There are also 8 Analogue Inputs (0-5V) and 4 Analogue Outputs (0-10V). Custom Functions may be defined by the user in Tbasic Programming language.

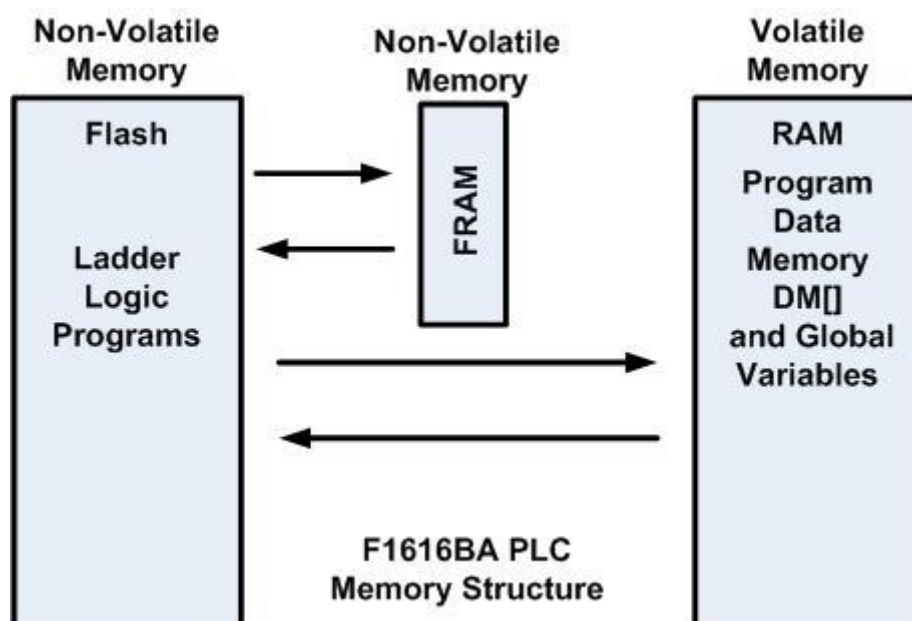
The unit also has 2 RS485, and one RS232 ports with an additional Ethernet port. The RS232 or the Ethernet can be used to program the PLC and the Ethernet port can also be used in remote control internet applications, programming, networked systems and SCADA systems.

The aim of the exercises carried out in the control lab is to familiarize you with some of the features of the PLC and also introduce you to its windows based programming and simulation software i-Trilogi.

PLC Memory Structure

Most modern PLC memory structures contain two types of memory, **Volatile** and **Non-Volatile**. A Volatile memory is known as a Random Access Memory or RAM which is temporarily used to store data. When a PLC's Power is turned OFF the contents of the memory are lost. A Non-Volatile memory can be an Electrically Erasable Programmable Read Only Memory or EEPROM or a Flash memory. When power is Turned OFF the EEPROM and the Flash retain their programs/data.

The F1616BA PLC has a Ferromagnetic RAM (FRAM) of 6K words which may hold data or sections of program code such as sub-routines, that are commonly used by one or more programs. The Main Flash memory is used to store the actual Ladder-logic program. Once a PLC is programmed it will automatically execute the program code upon power up. The following diagram shows how the F1616BA PLC's memory structure is organised:



It is important to understand how the memories can be accessed, as this information will be used in the Application programming examples that follow later in this document.

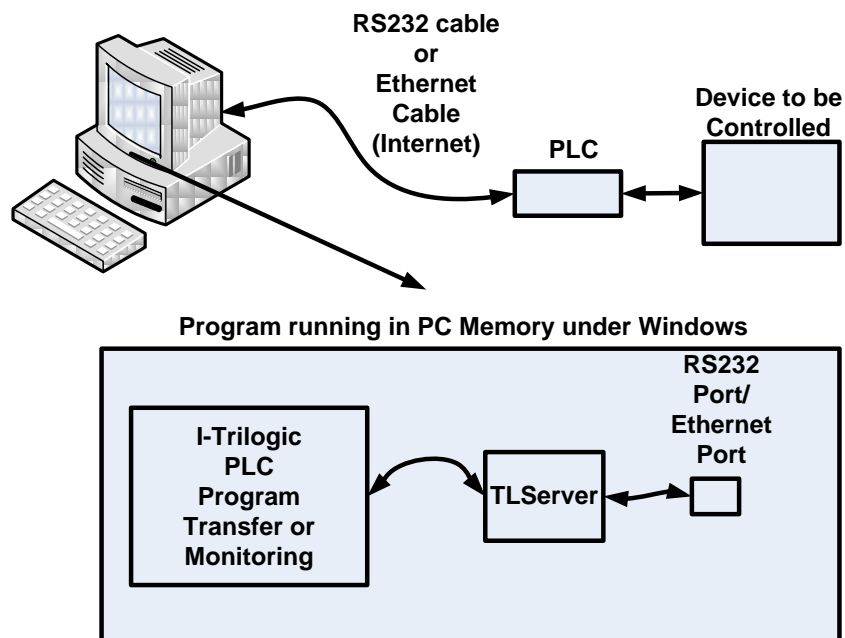
Access to FRAM and RAM is only possible from a ladder-logic program running in the Flash memory of the PLC. It is not possible to exchange data from the FRAM to the RAM directly. The RAM memory is extensively used by programs to store data, such as a data acquisition program sampling an A/D converter. The RAM can also hold **Global Variables** used in various Ladder logic custom functions. The Data Memory is accessed via a Global Array pointer called **DM[n]** where **DM** stands for Data Memory and **[n]** gives its storage location **n**. If data is to be stored in a number of memory locations, then an index variable can be used to create a mass storage area, say from location $n=1$ to $n=50$ etc for 50 values of data.

Transferring Programs to the PLC

The i-Trilogi Software is a windows based Integrated Development Environment package, where Ladder logic and TBasic programs can be edited, compiled and loaded to the PLC. The Software can also run Pure Simulations including monitoring the state of all I/O devices and has a Virtual LCD Display. The Transfer of programs can be done either via the RS232 or the Ethernet port for Internet remote Programming, Control and Monitoring applications.

However, the transfer and monitoring part of the i-Trilogi software requires a Bridging Program that provides communication between either port from PC to the PLC. This is known as the TLServer(TriLogi Server or PLC web Server). The following diagram shows the use of the TLserver software.

The F1616BA PLC also stores the name and folder location of the program loaded into its memory so that the user can be aware which program is running and where it is stored on the PC.



Precautions when handling the PLC

The PLC is a very delicate and sophisticated board and must be handled carefully to prevent it being damaged. As with all electronic microcontrollers and processors, the board components are static sensitive. You must touch and earth any static charge on your clothing by touching the Table frames in the lab as these are earthed and will discharge residual static electricity. The boards are protected by two layers of acrylic sheet to reduce the chance of coming into contact with the electronics on the boards.

Note: if you are not sure about connecting the PLCs to external devices, then you must ask the Technicians in charge to do this for you.

List of Simulation exercises

The exercises in this document include introduction to ladder logic programming to provide practical support for the theory taught in class room lectures. In the following exercises you will learn how to, run a pure simulation initially and monitor its state and later, load the ladder logic programs to the PLC and run it.

- 1) Simple Input Output: activate a digital output using a digital input
- 2) OR Function digital input output control
- 3) AND Function digital input output.
- 4) Toggle digital I/O control
- 5) Set – reset Digital output control
- 6) Clocked output
- 7) Contactor
- 8) Remote control Contactor
- 9) Contactor with Timeout
- 10) 3 phase Induction Motor Delta connection Soft starter
- 11) 3 phase Induction Motor Star-Delta connection starter
- 12) Sequence Control

List of Application exercises(available on separate sheets)

These exercises include the use of the knowledge gained in simulations, for a real Control application. The aim here is to develop ladder logic programs to operate a number of rigs in the laboratory.

These rigs are:

- 1) Water level control in 2 tanks
- 2) Temperature control
- 3) Flow control using a digital flow sensor and PWM pump drive
- 4) Industrial Control Trainer Assembly Rig
- 5) Traffic Lights Simulator
- 6) Washing Machine Simulator
- 7) Elevator Control

Due to the limited number of rigs available, the exercises should first be simulated to ensure that the ladder logic operates as expected. You may then store the program on a Memory Stick and run it on the PLC connected to the Rig via a PC.

i-Trilogi Software

This is a user friendly windows based program that includes help files and example ladder logic programs. Locate the folder named **F1616BA** on the **Desktop** and open it. You will find a file named **Blank.pc6** in the folder. Double click the file and open it.

NOTE: **Blank.pc6** will be the default file you will use to develop any other program and all programs must be saved in this folder. Programs found in any other folder will be deleted.

The following window appears:

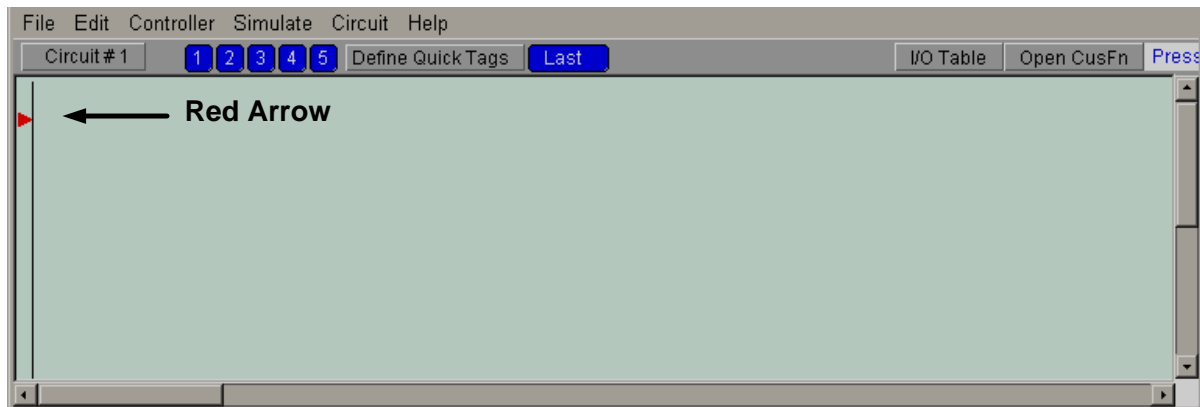


Instructions will be given for choosing a suitable filename for your exercises in the subsequent pages.

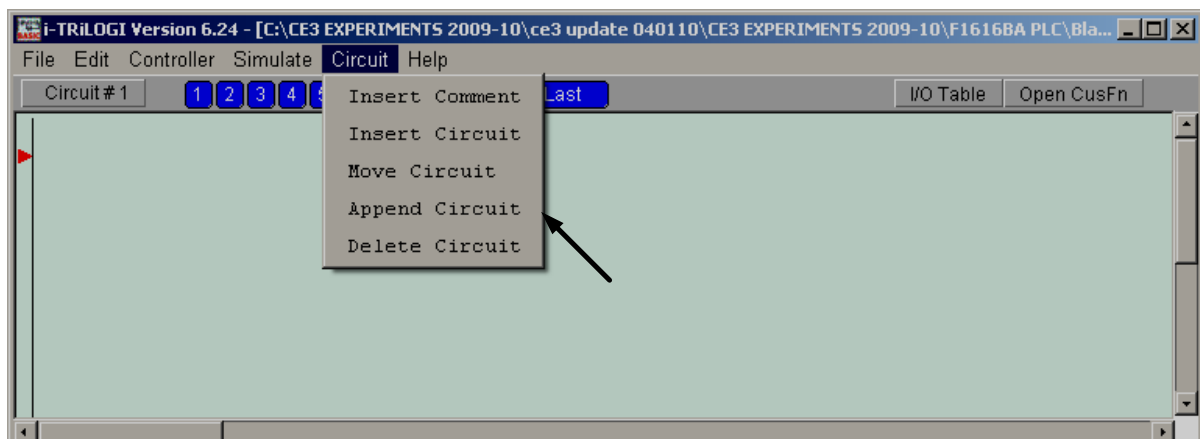
Do Not attempt to create ladder logic programs now, just complete reading the subsequent pages to get an idea of what to expect.

Ladder Logic editing tips

This section gives a general overview on how circuits are edited and what to expect. **You are not required to attempt any editing now.** The exercise section of this document will cover pragmatic editing details.



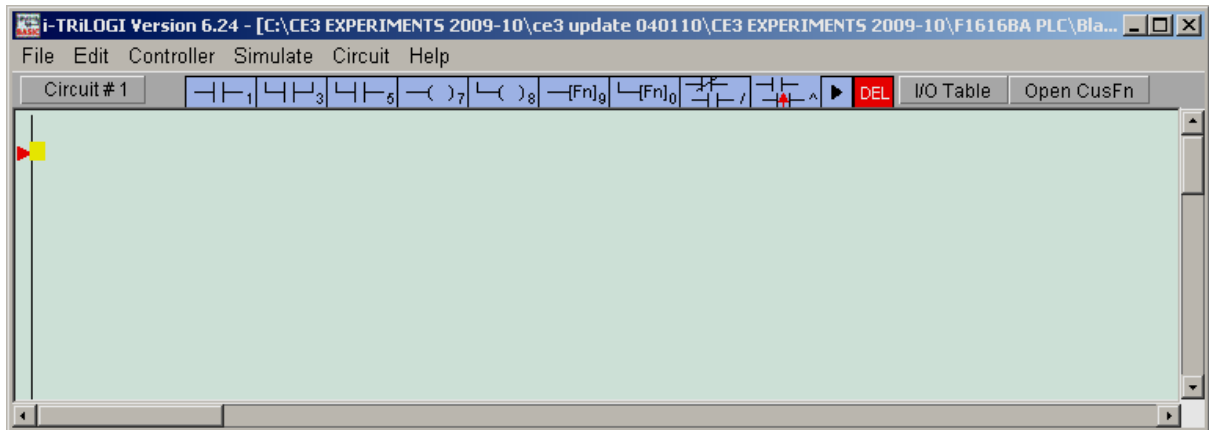
The above picture shows the i-Trilogi editing window. The Red arrow in the left hand corner, is an insertion point indicator. A circuit can be inserted at any point by clicking the left hand side of each Rung.



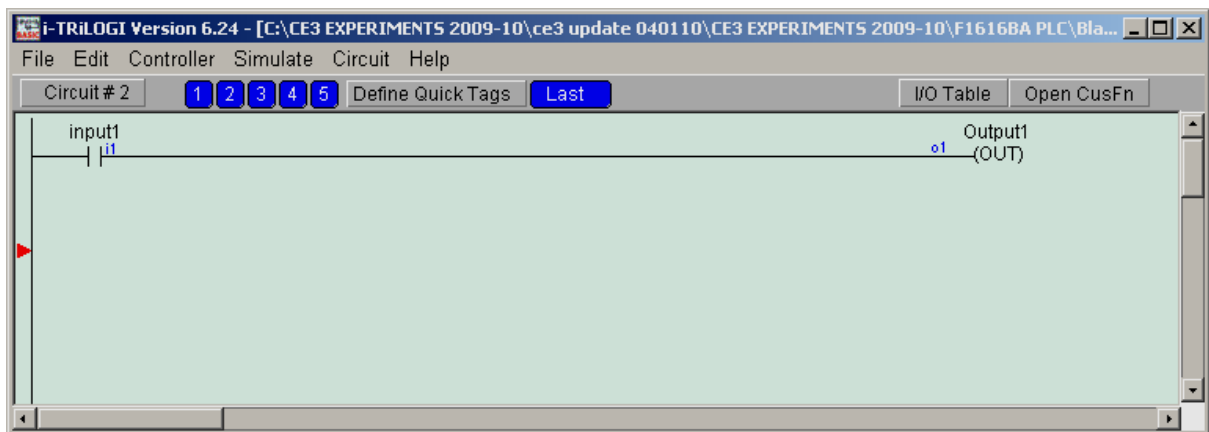
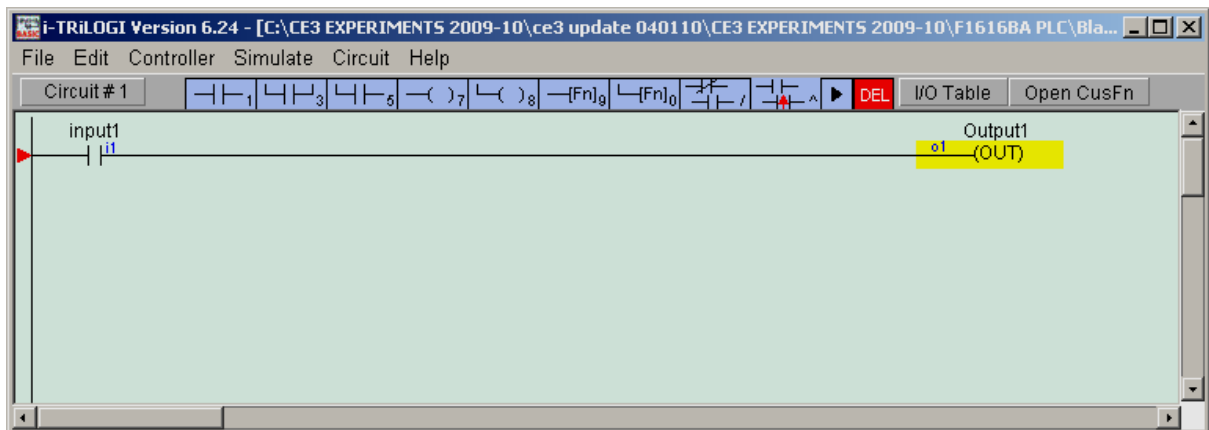
The Circuit menu allows the users to insert a Comment, a Circuit or edit an existing circuit by moving, appending or deleting elements from a Rung.

Note: It is recommended that you always design the ladder logic circuit on paper and prepare the I/O Table before attempting to create a ladder logic program. Inserting comments is highly recommended to clarify the operation of each Rung.

When you click on **Insert Circuit** a small menu of Ladder Logic symbols appear as shown overleaf. All Inputs and Outputs must be defined and named in the **I/O Table** before attempting to create a ladder logic program.



Elements are inserted from the IO Table.

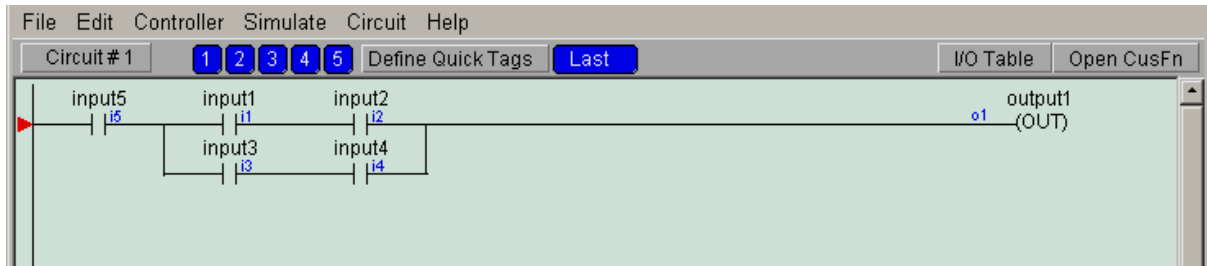


The next Rung is selected by clicking on the left hand Rung. The Red Arrow advances to the next Rung indicating the next insertion point.

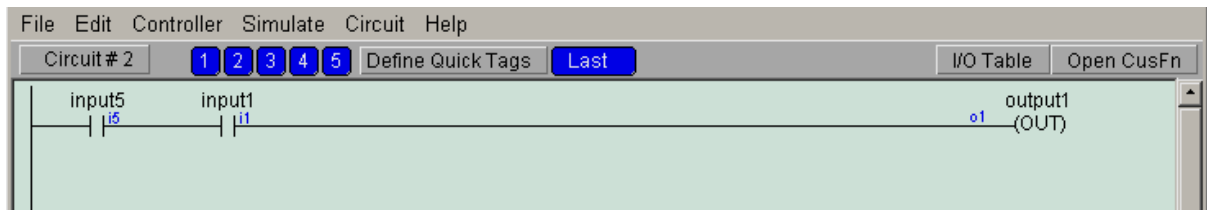
Complex Series and Parallel connections

There is a few points that users must bear in mind when connecting one or more inputs enclosed by others.

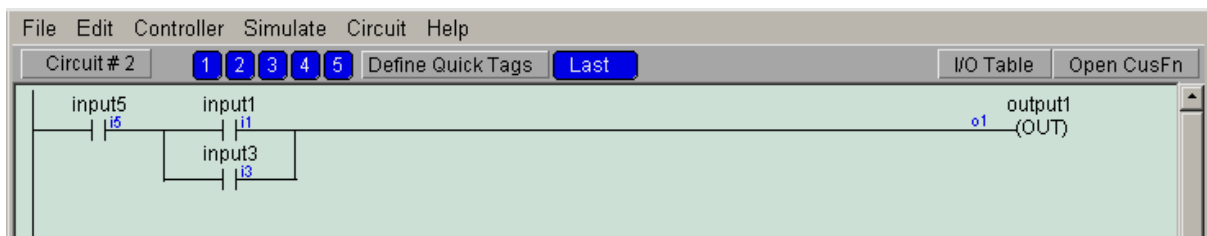
Consider the following circuit in which 4 inputs are connected in series and parallel.



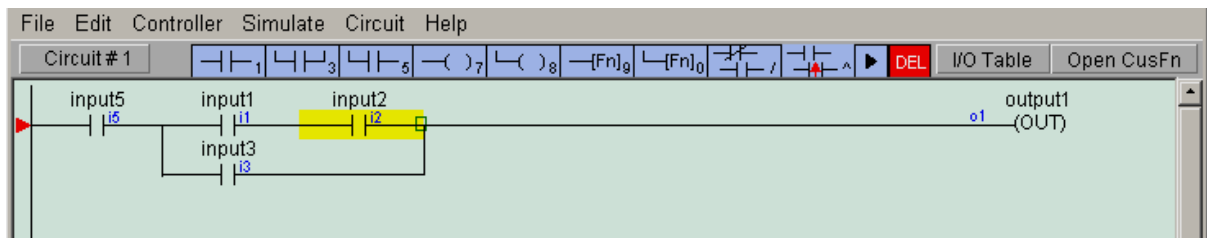
Here start with inserting input5, input1 and output1.



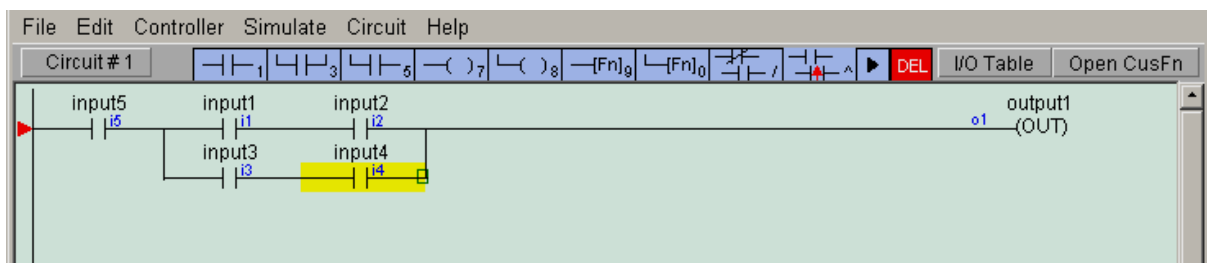
Click on input1 to highlight it. Now insert input3 as a parallel input.



Then click on input1 again to highlight it. Now insert input2 in series.



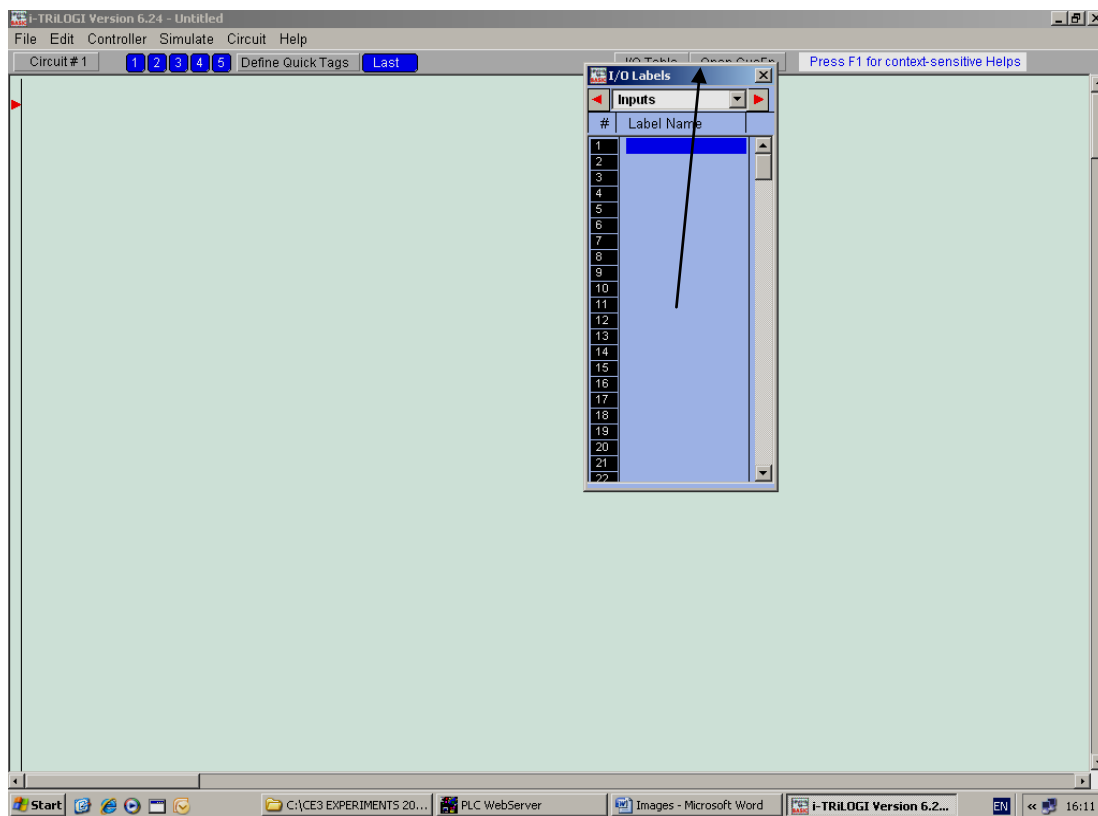
Click on input3 now to highlight it. Then insert input4 in series.



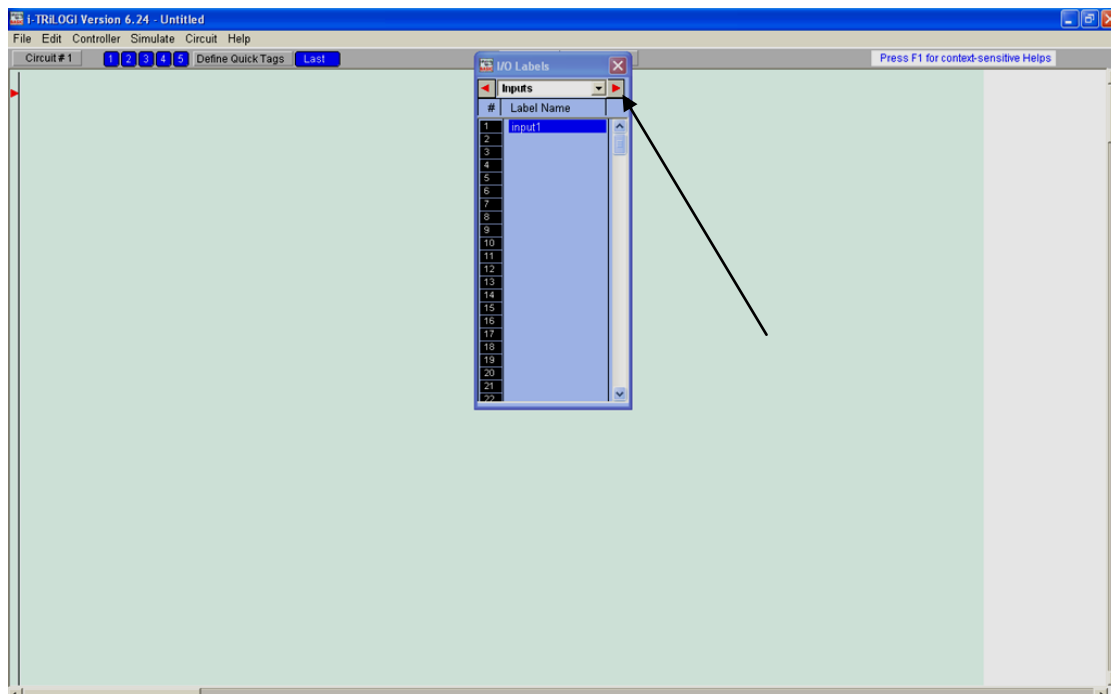
Let's create a simple program.....

Exercise 1. Simple Input / output: Activate a Digital Output using a Digital Input.

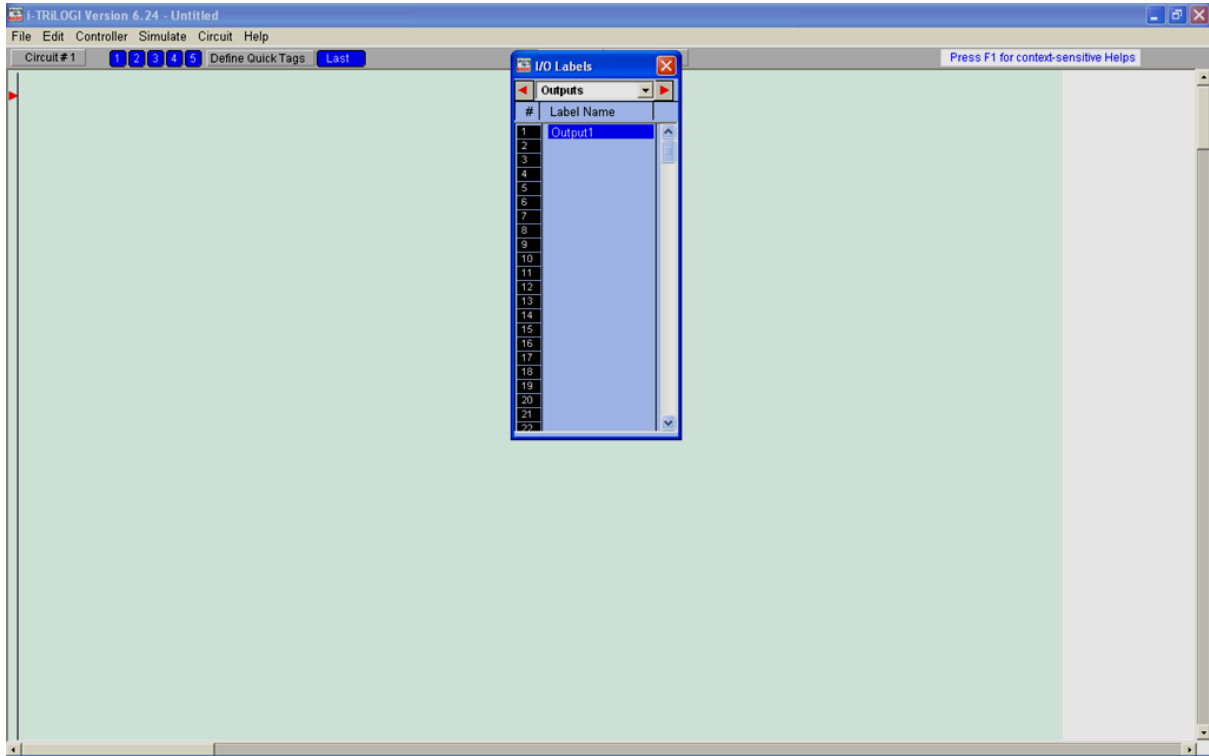
You can label an input or output by clicking on the I/O Table Tab shown:



Now click next to number 1 input box and type in: **Input1** and Press the **Enter** key.

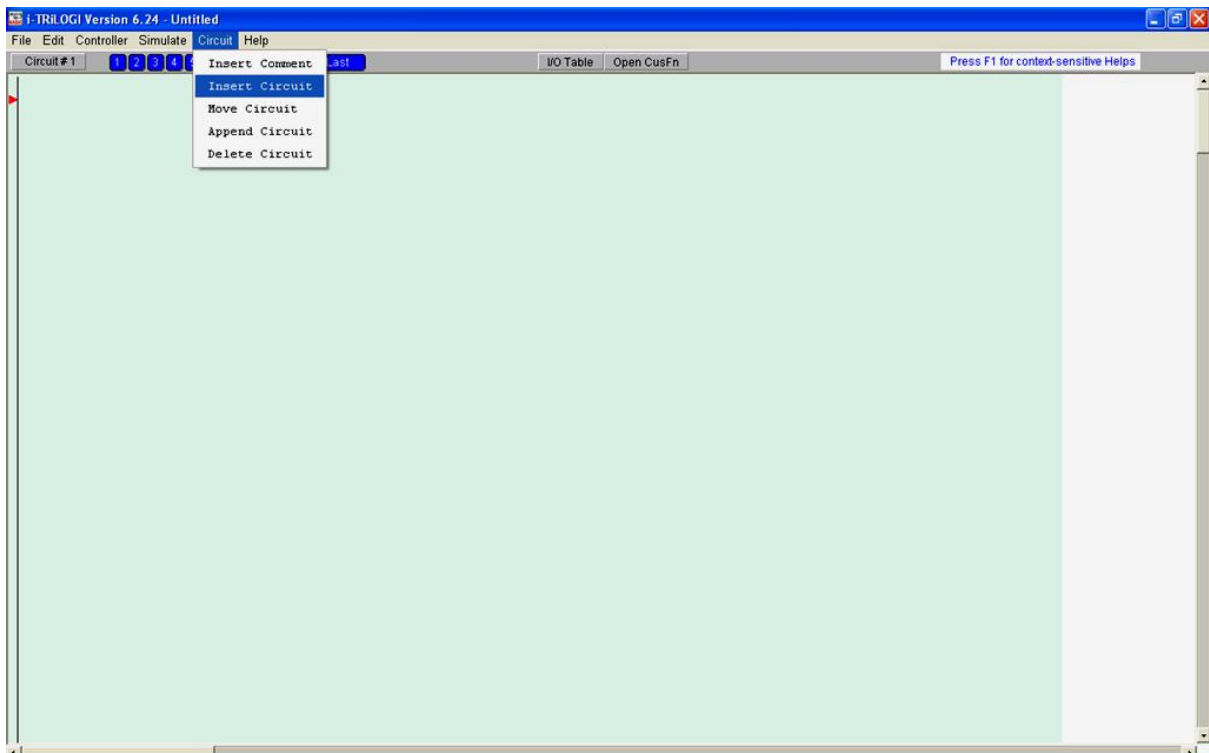


Now click on the Red Right Arrow to select the Output Table.

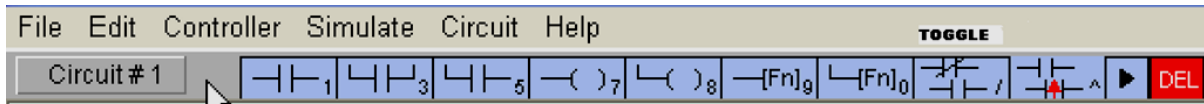


Type in: **Output1** in the first output box and press the **Enter** key.

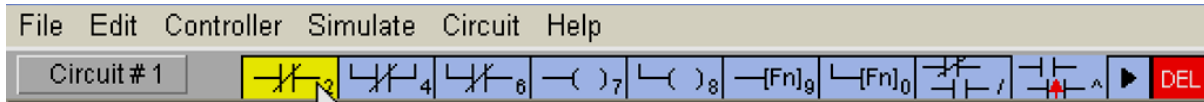
Close the I/O table window and click on the File Menu **Circuit** then select **Insert Circuit**.



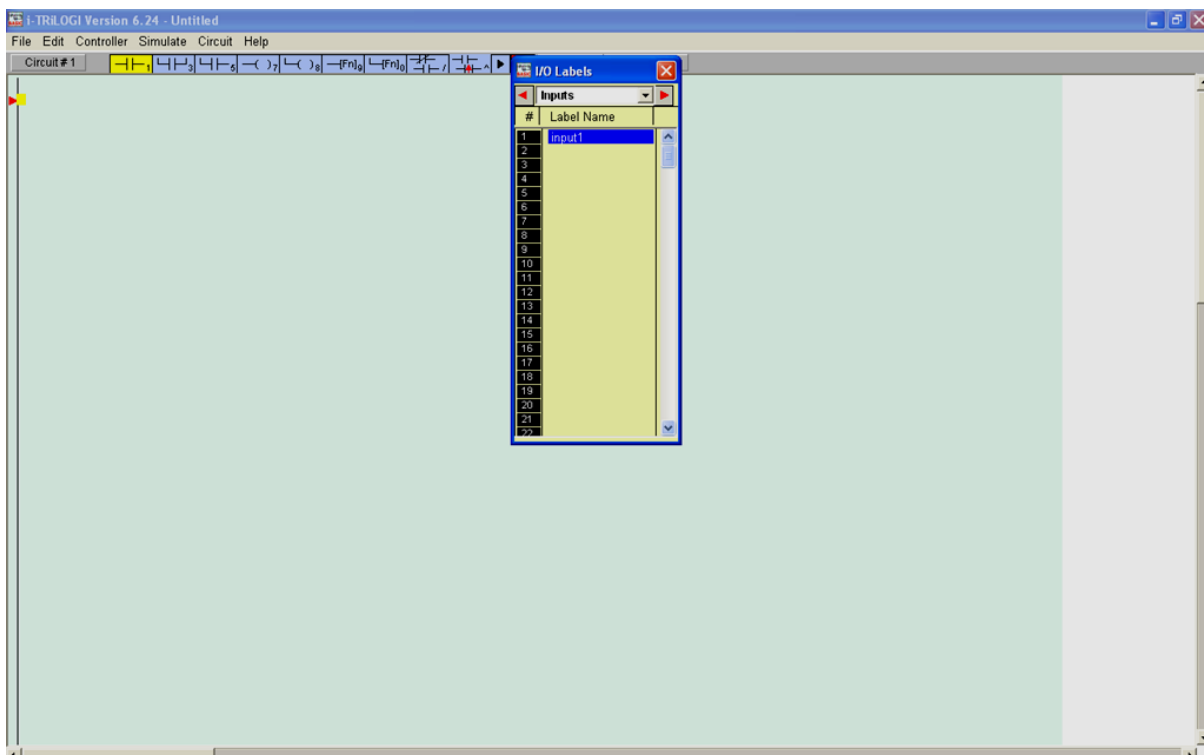
A table of Logic symbols will appear as shown on the next page.



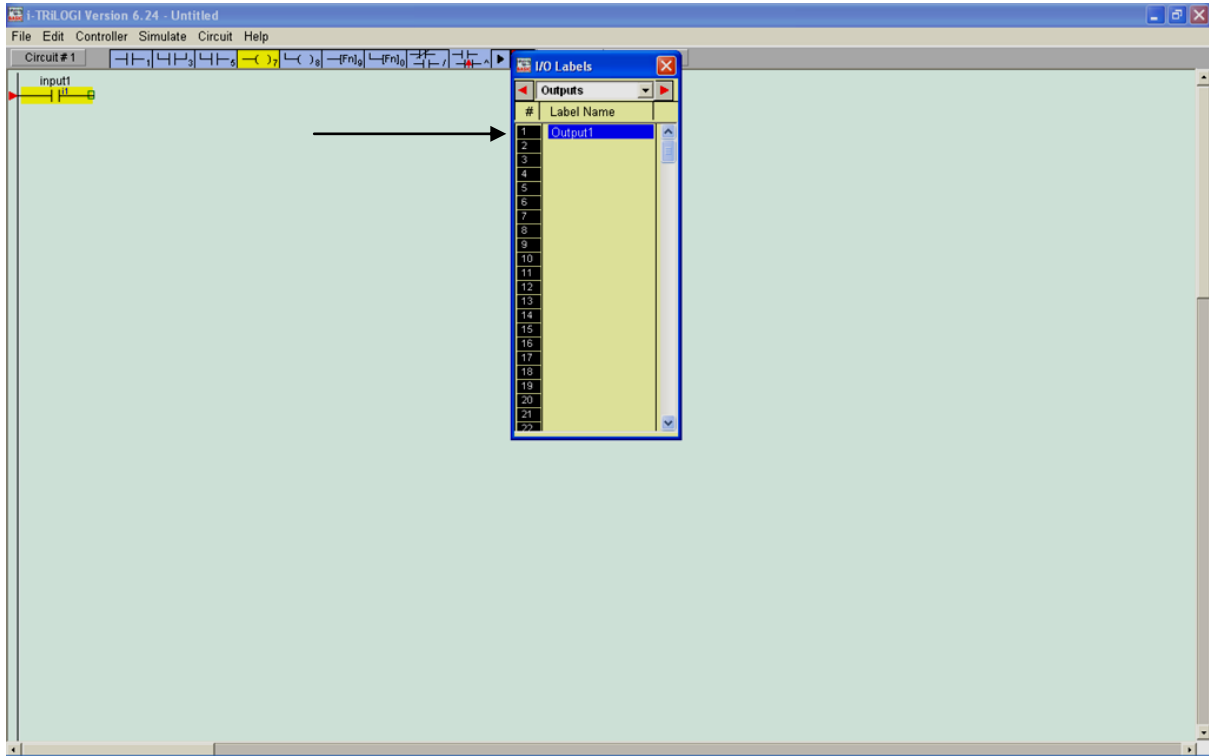
Note that the symbols above are numbered. The first three symbols define Normally Open(NO) Inputs: 1 in series, 3 in parallel, 5 in parallel branching to another in series. If a Normally Closed(NC) input is required, then right-click the mouse pointing to the symbol and a new set of NC inputs will be displayed.



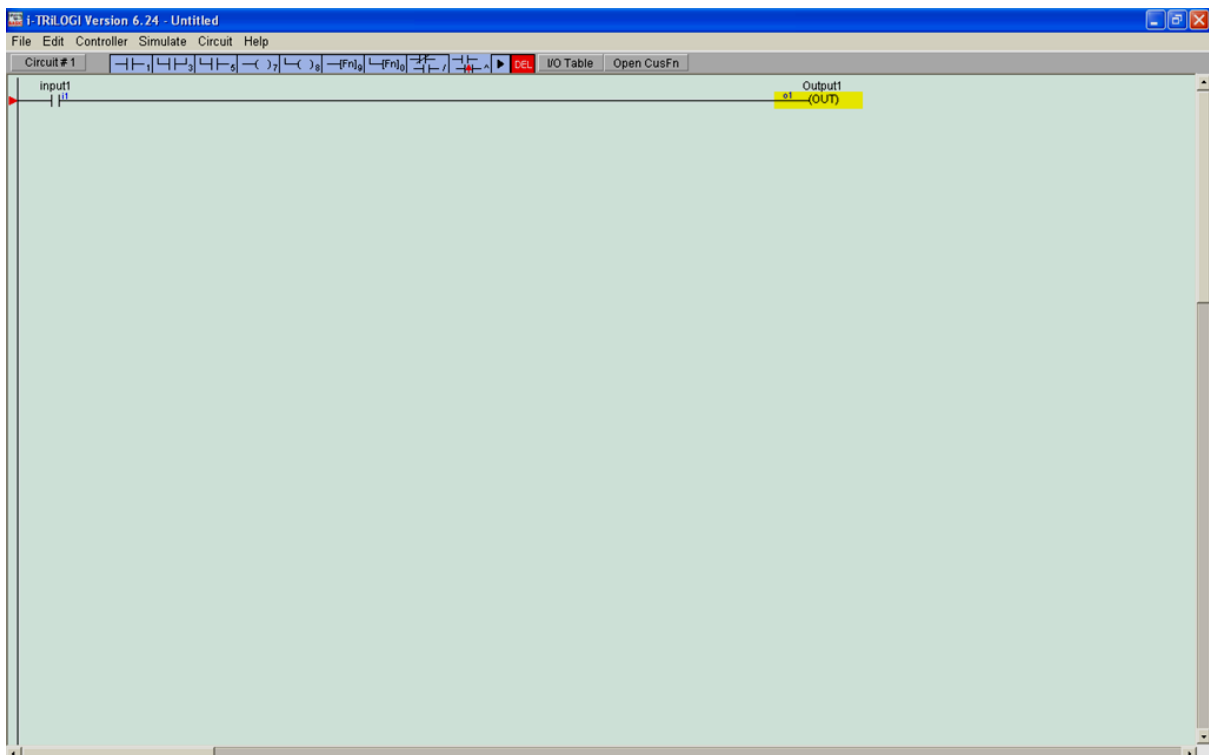
The NC inputs are numbered 2, 4, 6. The remaining symbols are Outputs 7 and 8 for Series and parallel connection. Alternatively, to change an Input symbol in the ladder diagram from NO to NC or vice versa, you can use the **Toggle** symbol. Symbols 9 and 10 are user defined functions. You may then go to the I/O table and choose the Input.



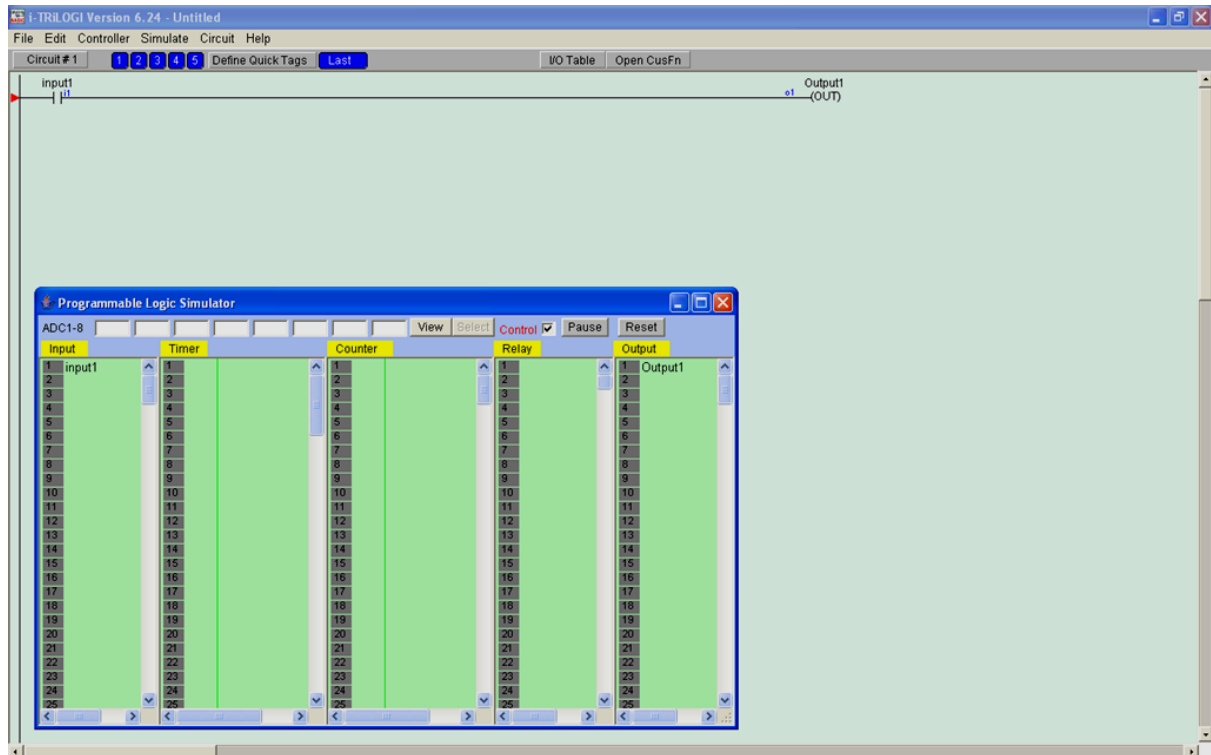
Click on the Normally Open Switch input symbol appearing in Yellow. Then go to the I/O table and select Inputs using the Left Red Arrow. Click on 1 appearing under the # symbol. The input will be inserted and will appear in Yellow. Now click on the Output symbol 8.



Then click on the output number **1**. Digital output will be inserted as shown:



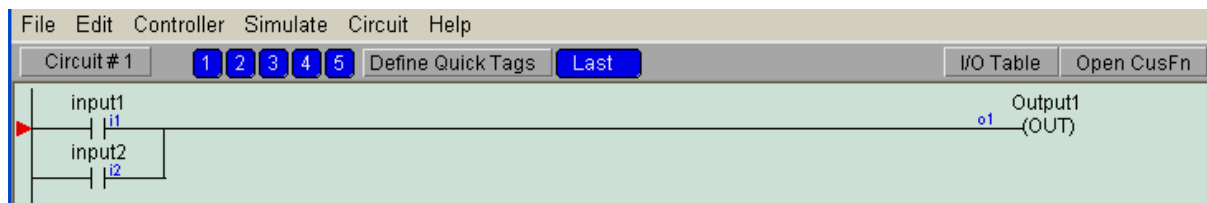
Your first circuit is complete. You may now save your program by clicking on **File / Save As(Local Drive)** and save the file with the name **SimpleIO**. Now run the simulation by clicking on the File menu **Simulate, Run(All I/O reset)**



An additional window appears as shown above showing the state of the I/O and other devices. To test the simulation move the mouse pointer to the number 1 input box, Click and Hold. You will see the Input1 and the Output1 symbols in the ladder logic appear in Blue and also in the Simulation window in Red indicating that they are active. ***To activate the Input and and latch it, you can Right Click on the Input where it will remain latched until you click on it again.***

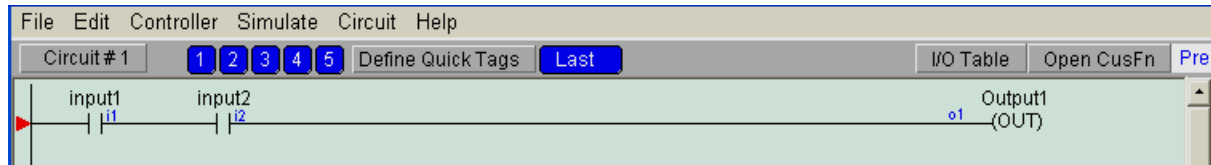
Exercise 2. OR function inputs

As described before open the Default file **Blank.pc6** and Save your file to disk and name it **ORfunction**. Edit a new circuit as shown below defining 2 inputs in parallel activating one output. Test the circuit by running the simulation. Activate Input1 and latch it to remain active, then activate input2. Comment on how the circuit works.



Exercise 3. AND function inputs

Open the Default **Blank.pc6** file. Save your file to disk and name it **ANDfunction**. Edit a new circuit defining two inputs in series activating one output. Test the circuit by running the simulation. Activate Input1 and latch it to remain active, then activate input2. Comment on how the circuit works.

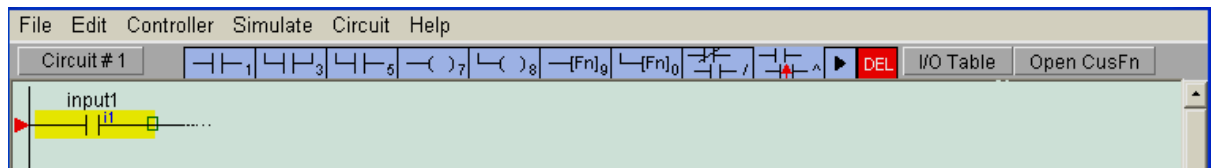


Exercise 4. Toggle Output

In this exercise the use of a Custom/User defined function will be shown. This requires careful input of data into the IO Table. The circuit that will be made is shown below, but you must follow the instruction given exactly. Open the Default **Blank.pc6** file and save the file with the name **ToggleOutput**.

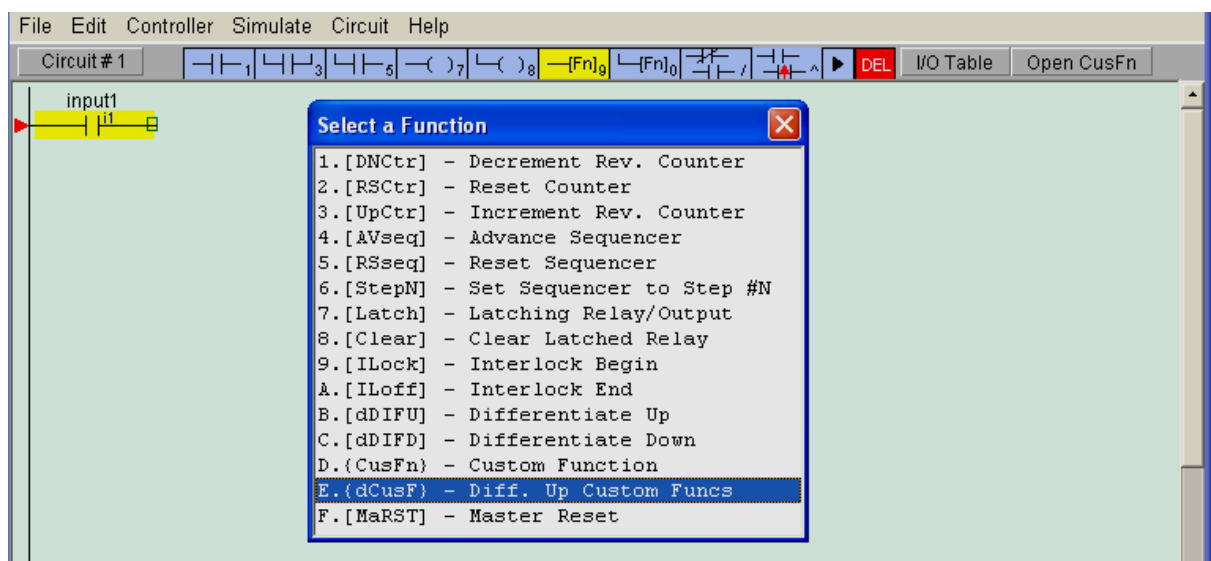
In this circuit **input1** can be defined as in previous exercises using the IO Table. Now define an **Output** and label it **Output1** in the output table. In the IO table

Cust Func define a function called **ToggleOutput**. Insert input1 into the edit window and follow the steps given below for the custom function:

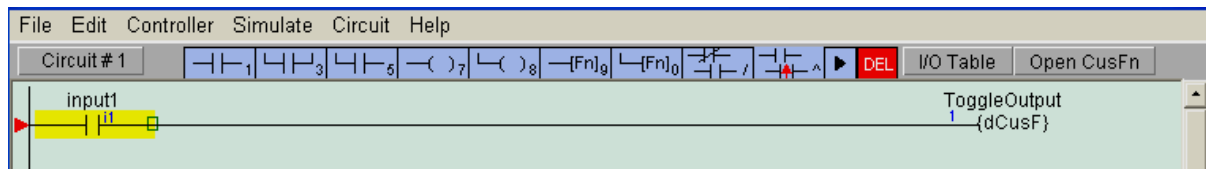


Now click on symbol 9, a custom function output. The following table appears. Select option **E.(dCusF)** and click. The custom output should appear as shown and should be highlighted in Yellow. Now click on OpenCusFn next to the IO Table tab.

The following window opens:

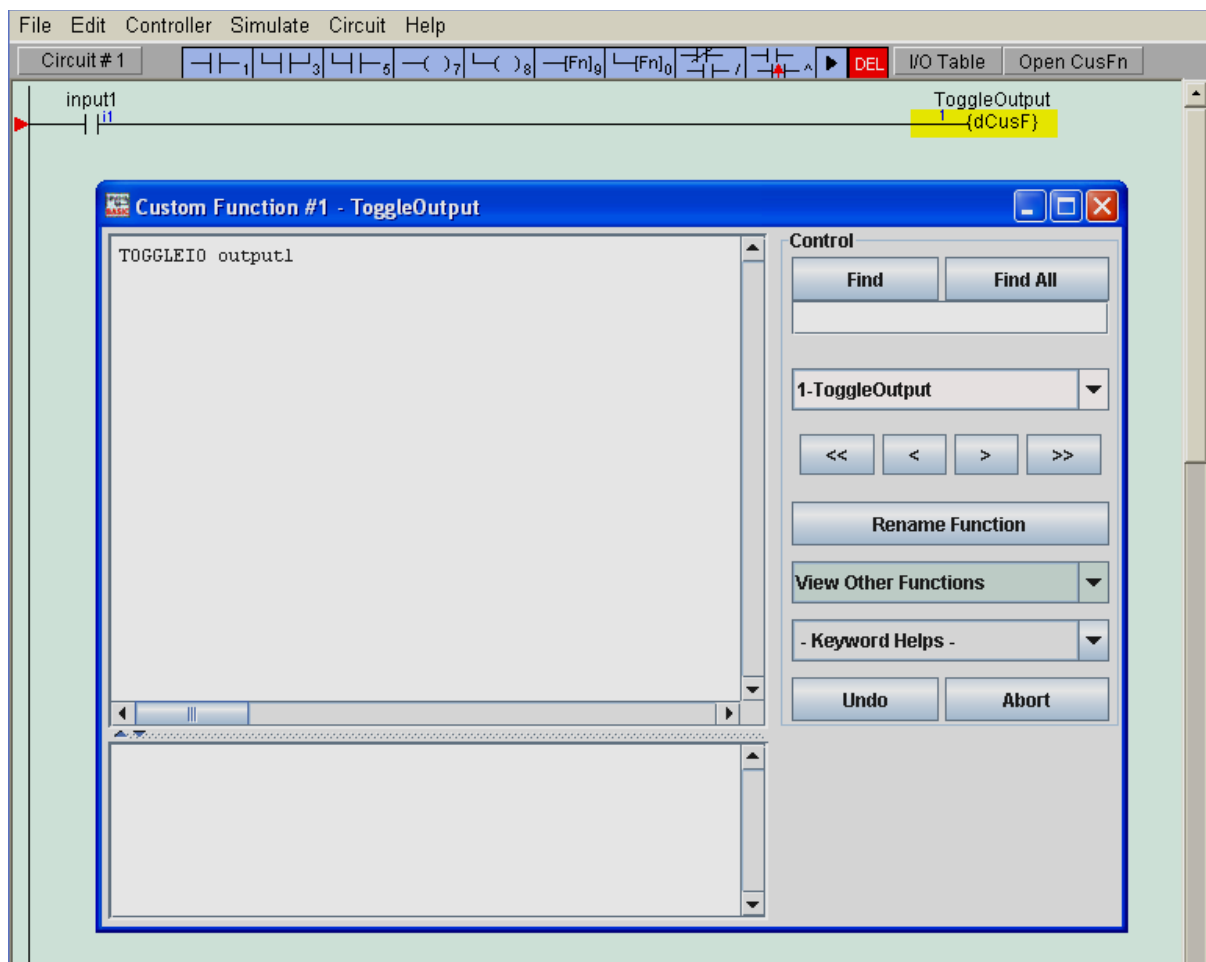


You should now see the following window with the Custom function highlighted in Yellow:



Double click ToggleOutput symbol to edit the function.

The following window appears:



You can now type in the Tbasic command **TOGGLEIO output1** in to the window as shown. Close the window. This completes the custom function command.

Run the simulation as before and click once on input1. Comment on the running of the simulation. Click again on the input and observe the output1 status.

Exercise 5. Set-Reset Digital output control

This exercise includes the use of TBASIC commands SETIO and CLRIO with a named digital output which can be turned ON or OFF respectively. A custom function must be declared as in the previous exercise to achieve this.

SETIO name Turn ON digital output labelled name

CLRIO name Turn OFF digital output labelled name

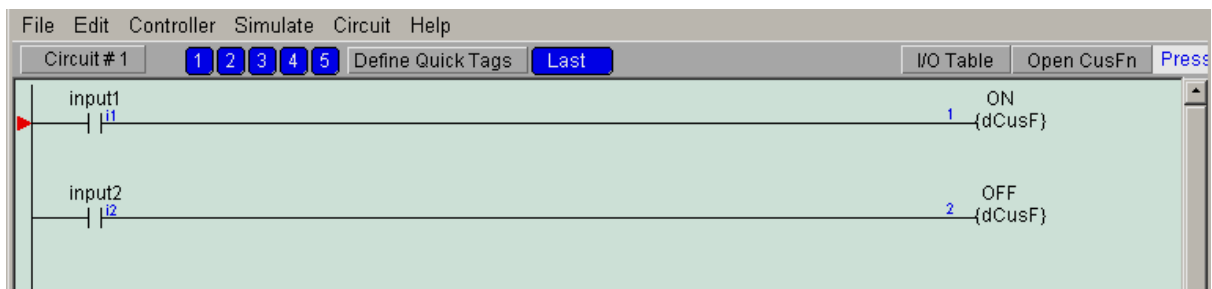
Eg: if digital output 1 is named Motor then SETIO Motor will turn digital output 1 ON and CLRIO Motor will turn it OFF.

Now load the previous File **ToggleOutput** and use **Save as(local file)** to rename this to **SetReset**. You can then declare the following items in the IO Table.

Inputs: input1, input2

Outputs: OUT1

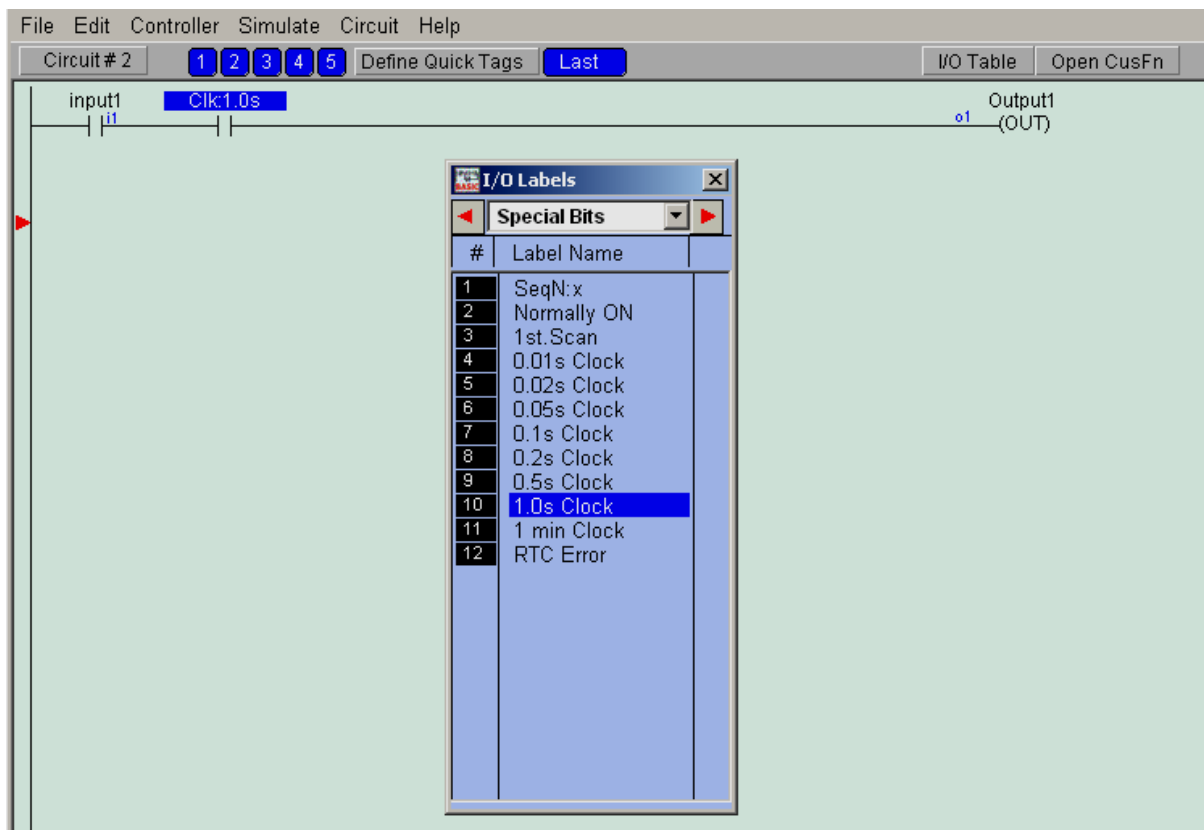
Now insert custom functions labelled ON and OFF as shown below and edit the functions to contain the commands SETIO OUT1 and CLRIO OUT1 respectively. Save the file and run the simulation. When you click input1, output1 switches ON and when input2 is clicked, output1 turns OFF.



Exercise 6. Clocked output

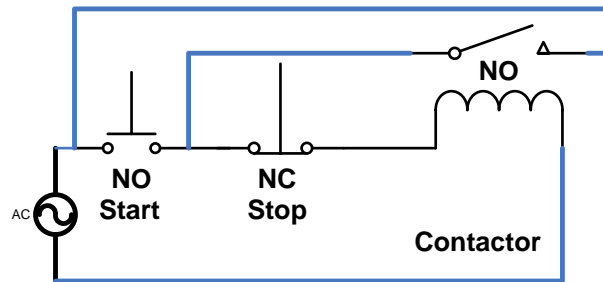
This exercise involves the use of the Special Bits command(clk) which is a clock with a set real time period. Open the Default **Blank.pc6** file. Save your file as before with the name **Clockoutput**.

Define two Inputs: Input1 as a NO switch and a NO switch defined as a 1 second clock from the Special Bits menu in the IO Table. The output can be defined as output1. Run the simulation and observe output1 switching ON and OFF every second when input1 becomes active.

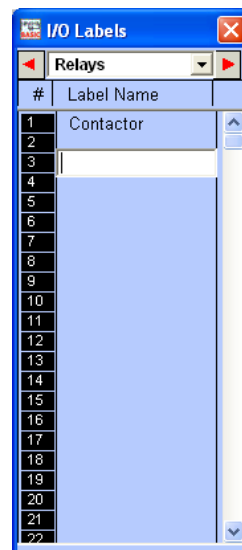
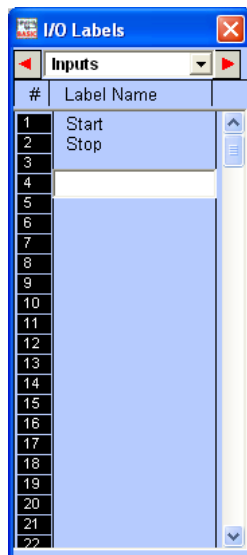


Exercise 7. Contactor

This is a direct application of OR and AND function exercises you have carried out.



Create a new file from **Blank.pc6** and immediately use **File/Save as(local drive)** from the file menu to name the file **Contactor**. Click I/O Table and define the following Inputs: **Start** and **Stop** then go to **Relays** and define relay 1 as Contactor.

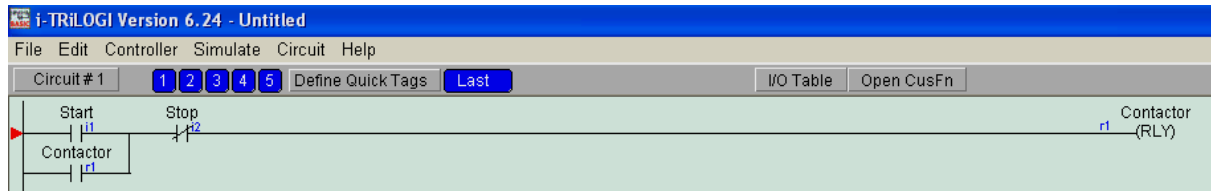


Then click **Circuit, Insert Circuit**. Then select a NO input from the menu. The I/O Table window opens automatically. Now choose **Inputs** and click on Input 1 i.e Start.

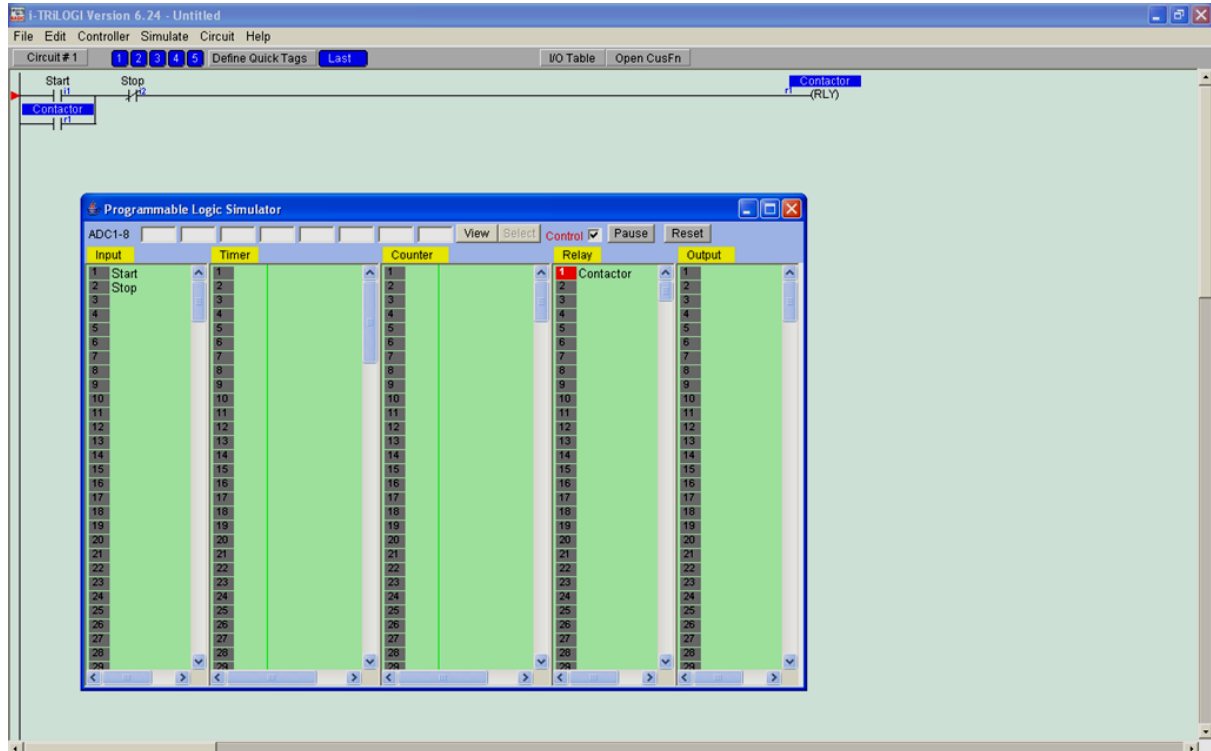
The input will be inserted and you should see it highlighted in Yellow on screen. Then click on NO input again from the menu and choose input 2 i.e Stop.

The input will be inserted and highlighted in Yellow. Now let's change this to a NC input using the Toggle Input option. You will see a message indicating the input has been changed. Click OK to continue. As you insert the components you should see the following circuit shown in the picture below.

Click on the Output symbol number 7 and select Relays from the I/O Table then click on Contactor.. We will now insert a NO contact for the Contactor Relay. Now click on the Start input to select it, then click on symbol 3 i.e NO input in Parallel. Then select Relays Contactor from the I/O table.



Now run the simulation as before and click on the Start input as shwon below:



The contactor is now activated and remains active until you click the Stop input.

Exercise 8. Remote control contactor.

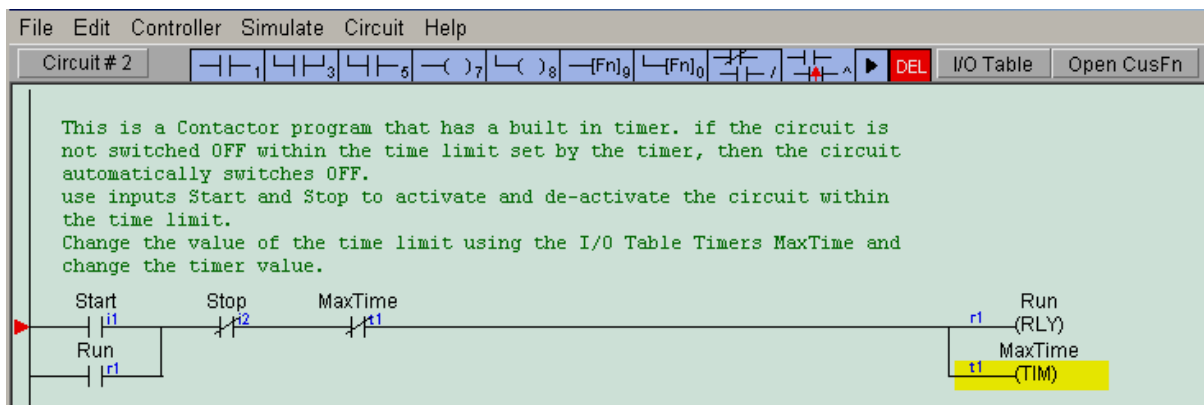
An Industrial operation requires the use of a Pump which can be operated by three different operators located in different rooms.

Save the Contactor file using **Save as(local drive)** with a new name **RemoteContactor** and modify the file to create a new ladder logic program to turn the contactor ON/OFF using separate Start/Stop switches.

Exercise 9. Contactor with a Timeout

The following exercise involves the use of a Timer to act as a Timeout safety feature in a process that needs to be switched OFF either manually or automatically after a set period. The nature of the process is such that an operator may Start the process at any time of the day and may switch it OFF by pressing the Stop switch. Alternatley if the process is not switched OFF manually, the timer will automatically Stop it after a set period.

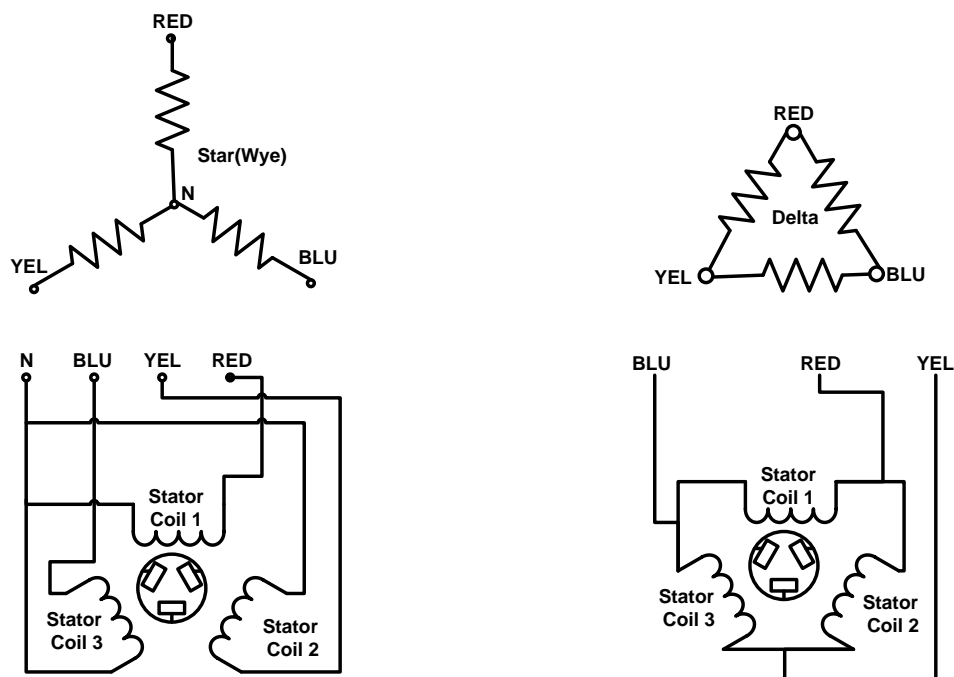
You can now edit the following Timer output and its NC switch in series with the previous contactor circuit. You can load the previous file **Contactor** and use File/save as(local drive) with a new name **TimeoutContactor**. Edit the file and insert the new input and output. Define **Maxtime** as an output **Timer** in the I/O Table with a timeout value of 200 and a NC input as **Maxtime** which will open its contact once the time has elapsed and switch OFF the Process.



Run the simulation and test the circuit by initially activating the Start input to see the Run Relay and the Timer activated. You will see the timer value decreasing in seconds. Once the time has elapsed the circuit will switch OFF. Alternatley the circuit can be stopped by activating the Stop input within the set time.

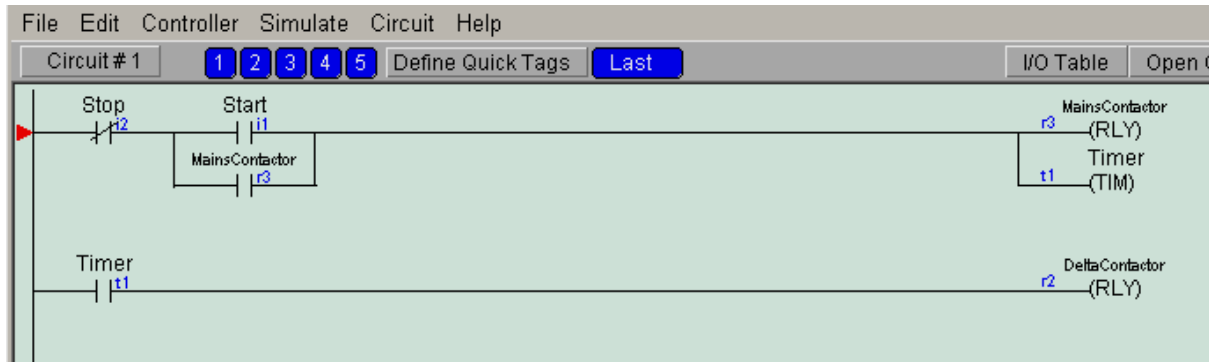
Exercise 10. Three phase Induction Motor Soft starter

An induction motor is a special motor in which the rotor is not directly energised by a power supply. The stator of the motor may have one or more coil windings which **induces** a current in the rotor, therefore the word **Induction**, which in turn creates an opposing magnetic field and therefore the rotor starts to rotate. Induction motors can be started in several ways and can be single phase or three phase. The exercises below focus on three phase motors. A high power induction motor needs a large starting current initially to overcome the inertia of the rotor. This surge in starting current can be several times the normal operating current and can sometimes damage the motor windings if not started correctly. It can also affect the power loading on the mains and result in a voltage drop that can affect other equipment connected to the same three phase supply. Therefore, it is important to limit the surge in the starting current and allow the rotor to gain speed by a lower power applied to the stator windings. Once the rotor has gained speed and overcome the inertia, then full power can be applied to the motor. The following circuit diagrams show two ways (Star and Delta) in which the motor can be connected. The phase power lines are typically coloured in industry with RED (Brown), YELLOW (Black) and BLUE (Grey).

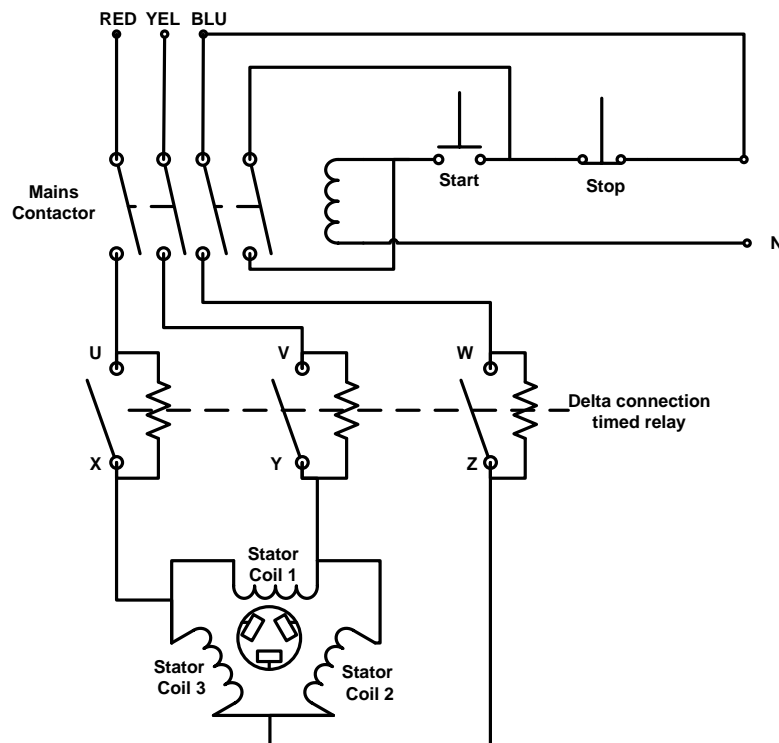


The Star connection requires the use of the Neutral (N) line and has much less surge current than the Delta connection. The Delta connection does not have a Neutral and its power surge current is much higher than the Star. The diagrams above also show how the Stator coils are connected for both. The next stage is to devise a way in which an Induction motor can be safely started. We will discuss two methods in which a three phase motor can be started. The first method is called a **Soft Starter**. This method involves a Delta connection which has resistances in series with each winding to reduce the surge current.

Initially, a circuit is used to keep the resistances in series with the Stator Coil windings and power is applied to the motor. Once the rotor has gained enough speed the resistors are shorted by a Timer Relay to apply full power to the motor for normal operation. The following is a circuit diagram that shows the connection of a relay across the Power Resistors. You may use the ladder logic diagram below as a reference.



Soft Start Delta Connection



The second method involves a Star connection and then switching to Delta for full power applied and maintained for normal operation. This requires understanding the typical electrical connections to the motor using several relays and timers.

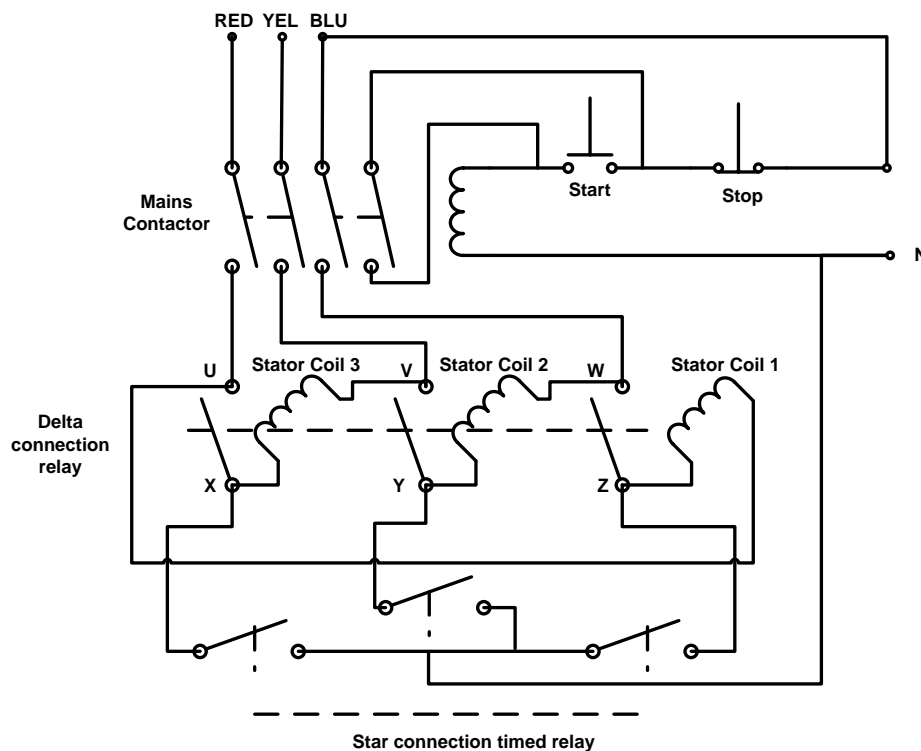
Note1: It is possible to purchase dedicated motor starter Modules which do not require the use of PLCs. However, if the motor is to be controlled automatically as part of an industrial process involving several other devices and particular sequence of operations, then a PLC can be used to achieve this.

Note2: Delta Relay Coil is not shown in the above picture.

Using the knowledge gained in previous exercises, create a new file with the name **SoftDelta** and design the Ladder logic to simulate the above circuit. You can use a Mains contactor and an additional Timer Relay. The Mains contactor can be activated as before using the Start button. This applies power to the Delta connection putting the Power Resistors in Series with the Windings. The Timer Relay should be OFF initially and become active after a set delay shorting points U,X and V,Y and W,Z.

Exercise 11. Three Phase Induction Motor Star-Delta starter

The circuit diagram below shows the connection of a Motor and its electrical starter circuit.



The circuit operates with a typical Mains contactor which can apply the three phase power lines to two switching relays. The Delta Relay is initially OFF and therefore the switches between X,U and Y,V and Z,W are normally open. The Star Relay has two contacts that can short circuit the X,Y,Z points to Neutral. The **Star** relay contacts are closed by a timer which allows sufficient time for the rotor to turn and gain speed. Once the set time has elapsed, then the Star relay is switched OFF and the Delta relay switched ON. This cause the Star connection points at X,Y,Z to break and connects the points X,U and Y,V and Z,W to create the **Delta** connection.

Note: Delta and Star Relay Coils are not shown in the above picture.

Using the knowledge gained in previous exercises, create a new file with the name **StarDelta** and design the Ladder logic to simulate the above circuit.

Comments on Exercise 10 & 11

There are fundamental safety problems with the circuits in exercises 10 and 11.

What are these?

How can you make the circuits fully safe?

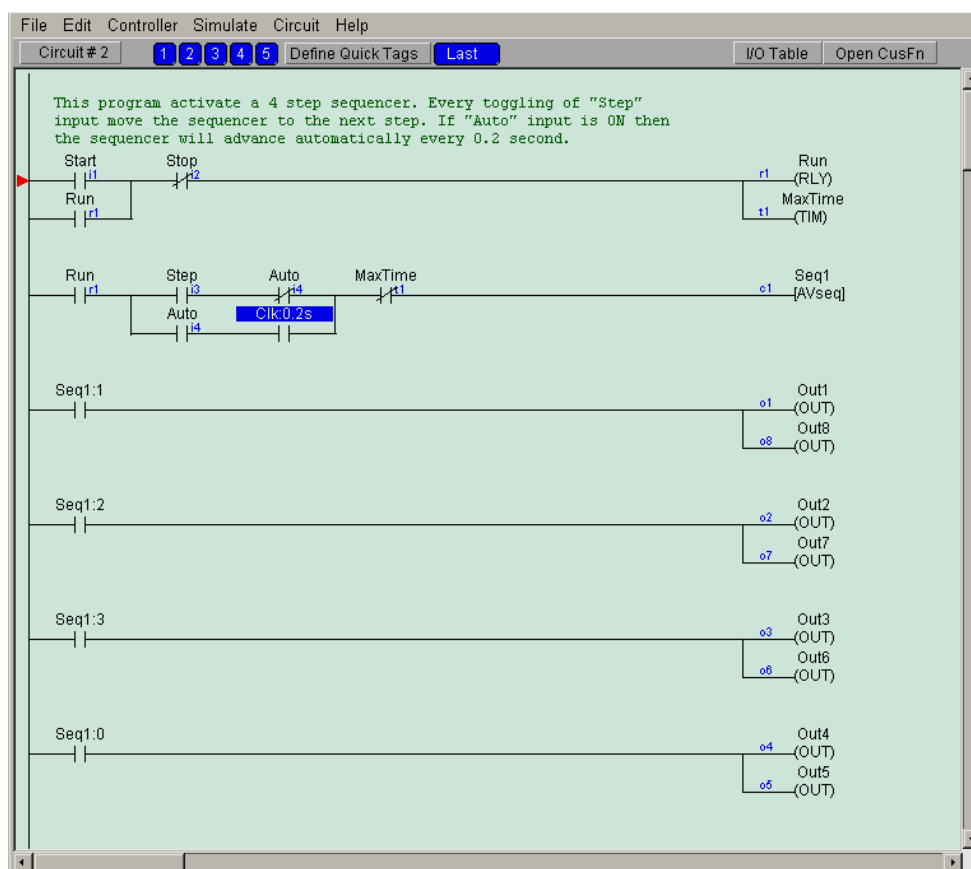
Exercise 12. Sequence control

In some industrial applications, a series of devices may need to be simultaneously controlled and may be activated in a sequence. The following ladder logic diagram is an example of such a system.

Sequential control specification

A system is comprised of 4 pairs of outputs which should be controlled in the following sequence:

Output 1 & 8, followed by 2 & 7, 3 & 6, 4 & 5. The sequence must be started by a Master contactor and may be stopped manually at any time using the Stop control. The system must be capable of automatically rotating the sequence above by pressing an Auto control or perform the sequence by a Step control. The system must also have a Timer which will stop the operation of the sequencer after a specified time but must not deactivate the Master contactor. The period between each sequence should be 0.2 seconds.



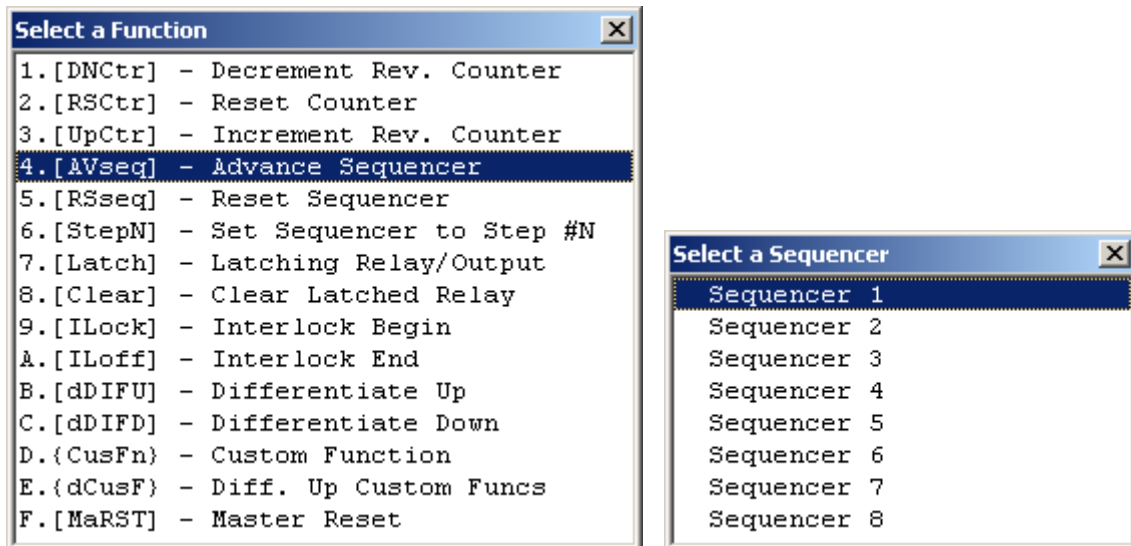
Using the above ladder logic circuit, define the following components needed to create the controller:

Inputs: Start, Stop, Run, Step, Auto, Clk(Special Bits Menu)

Outputs: out1,out2,out3,out4,out5,out6,out7,out8 **Relays:** Run

Timer: Maxtime with value 1000, **Counter:** Seq1 with value of 3

Function: (Fn, symbol 9) as Advance Sequence [AVseq] and select Sequencer 1 from the menu and label it Seq1. As shown below:














Counter Inputs: Seq1:0, Seq1:2, Seq1:3, and Seq1:4

Once you have completed the circuit , run the simulation and carefully observe the behaviour of the sequencer. Click the **Step** input once at a time and observe the output status. Click and latch the **Auto** input and observe the output status.

Describe in detail the full function of the sequencer and record all your observations in your Logbook.

Insert Ladder Element - You create the ladder circuit element simply by moving the mouse pointer to the icon and pressing either the left or the right mouse button to insert a ladder logic element to the currently highlighted element. The following is a description of the functions of each icon. A yellow colour highlight bar will appear which you can move to select an element in the ladder circuit.

	<1> - Left click to insert a normally-open series contact. <2> - Right click to insert a normally-closed series contact.
	<3> - Left click to insert a N.O. parallel contact to highlighted element <4> - Right click to insert a N.C. parallel contact to highlighted element
	<5> - Left click to insert a N.O. parallel contact to enclose one or more elements. <6> - Right click to insert a N.C. parallel contact to enclose one or more elements.
	<7> - Insert a normal coil which may be an output, relay, timer or counter.
	<8> - Insert a parallel output coil (not an entire branch) to the current coil.
	<9> - Insert a special function coil which includes execution of CusFn
	<0> - Insert a parallel special function coil to the current coil.
	</> - Invert the element from N.O. to N.C. or from N.C. to N.O.
	<^> - Convert the element to a rising-edge triggered contact (one shot)
	Click to move the highlight bar to the right (same effect as pressing the right arrow key). This can be used to move the cursor to a junction which cannot be selected by mouse click.
	Double-click to delete a highlighted element. This acts as a safety against mistake.