EMBEDDED SYSTEMS – IV

(Industrial Automation)

All boards are similar only the microcontroller varies…

→PLC's,SCADA,etc.

Advantage of Automation:

Automation has several advantages, including:

1.Increased Efficiency: Automation can complete tasks faster and more accurately than humans. This leads to an increase in productivity and a reduction in errors, resulting in greater efficiency and cost savings.

2.Improved Quality: Automation ensures that tasks are completed in a consistent and standardized manner, reducing the likelihood of errors and defects. This leads to improved quality of products and services.

3.Reduced Labor Costs: By automating tasks, businesses can reduce their labor costs by minimizing the need for human workers. This can lead to significant cost savings over time.

4.Increased Capacity: Automation can allow businesses to increase their production capacity without the need to hire additional staff. This can be particularly useful in industries where demand for products or services fluctuates.

5.Improved Safety: Automation can reduce the risk of workplace accidents and injuries by removing workers from dangerous or hazardous tasks. This can lead to a safer work environment and a reduction in workers' compensation claims.

6.Better Data Management: Automation can help businesses manage their data more efficiently, ensuring that it is accurate, up-to-date, and easily accessible. This can help businesses make better-informed decisions and improve their overall performance.

Overall, automation can provide numerous benefits to businesses, including increased efficiency, improved quality, reduced labor costs, increased capacity, improved safety, and better data management.

Disadvantages of Automation:

Despite its advantages, automation also has some disadvantages, including:

1.Initial Cost: The cost of implementing automation technology can be high, including purchasing and installing equipment, training employees, and integrating it into existing systems. This can be a significant barrier for smaller businesses.

2.Job Losses: Automation can lead to job losses, as machines replace human workers. This can be particularly challenging for employees who may not have the necessary skills to transition to other roles.

3.Reduced Flexibility: Automated processes are typically rigid and inflexible, which can make it challenging to adapt to changes in production or market demands. This can limit a business's ability to respond to unexpected events or opportunities.

4.Dependence on Technology: Automated systems can be vulnerable to technical failures or errors, which can cause delays or downtime. Businesses may also become overly reliant on technology, making them more vulnerable to cybersecurity threats.

5.Maintenance and Repair Costs: Automated equipment requires regular maintenance and repair, which can be expensive and time-consuming. If equipment breaks down, it can also lead to production delays and lost revenue.

6.Ethical Concerns: The use of automation can raise ethical concerns around the role of technology in the workplace and society, including issues around job displacement, privacy, and control over data.

Overall, while automation can provide numerous benefits, it is important to consider its potential drawbacks and to implement it in a way that minimizes negative impacts on employees, customers, and society as a whole.

INDUSTRIAL REVOLUTIONS :

The industrial revolution refers to a series of transformative periods in human history that led to significant changes in manufacturing, transportation, and communication. The four main industrial revolutions are as follows:

1.Industrial Revolution 1.0: The first industrial revolution occurred in the late 18th century and was characterized by the transition from manual labor to machine-based manufacturing. The use of steam power and the development of the factory system were key drivers of this revolution.

2.Industrial Revolution 2.0: The second industrial revolution took place in the late 19th and early 20th centuries and was characterized by the introduction of mass production, electrification, and the assembly line. This revolution saw the rise of new industries such as the automobile and oil industries.

3.Industrial Revolution 3.0: The third industrial revolution, also known as the Digital Revolution, occurred in the latter half of the 20th century and was characterized by the widespread use of computers, automation, and the internet. This revolution transformed the way we communicate, work, and access information.

4.Industrial Revolution 4.0: The fourth industrial revolution, also known as Industry 4.0, is the current era of technological change and is characterized by the integration of physical, digital, and biological systems. This revolution is driven by technologies such as artificial intelligence, robotics, and the Internet of Things (IoT) and is transforming industries such as manufacturing, healthcare, and transportation.

Each industrial revolution has had a profound impact on society, leading to significant changes in the way we work, live, and interact with each other. The fourth industrial revolution, in particular, is expected to bring about significant changes in the way we produce goods and services, and will likely continue to shape our world for decades to come.

AUTOMATION:

Automation refers to the use of technology and machines to perform tasks that were previously done by humans. It involves the use of software, sensors, and other technologies to control and operate machinery and equipment, often without human intervention.

Automation can be found in various industries, such as manufacturing, transportation, healthcare, and finance. In manufacturing, for example, automation is used to control assembly lines, machines, and robots to perform tasks such as welding, painting, and quality control. In transportation, automation is used to control vehicles, such as self-driving cars and trucks. In healthcare, automation is used to perform tasks such as dispensing medication and monitoring patient vital signs.

The benefits of automation include increased productivity, improved accuracy, and reduced labor costs. However, there are also concerns that automation can lead to job losses and the need for re-skilling workers. As such, the implementation of automation needs to be carefully managed to balance the benefits and potential drawbacks.

Overall, automation is an important technology that is transforming the way we work and live. It has the potential to make our lives easier and more efficient, but it also requires thoughtful planning to ensure that it is used in a responsible and sustainable way.

PLC

PLC stands for ***Programmable Logic Controller***. It is a specialized computer used to control and automate industrial processes and machinery. PLCs are commonly used in manufacturing plants, assembly lines, and other industrial applications to automate repetitive tasks, improve efficiency, and reduce costs.

PLCs are designed to be **highly reliable and rugged**, able to withstand harsh environments and operate 24/7 without interruption. They can be programmed to monitor and control a wide range of devices, including sensors, switches, motors, and other equipment. This allows them to perform complex automation tasks such as controlling the flow of materials, monitoring and adjusting temperatures, and sequencing operations in a production line.

PLCs are typically programmed using a specialized programming language such as **ladder logic**, which is designed to be easily understood by electricians and other professionals in the industrial automation field. The program is stored in the PLC's memory and executed by the PLC's processor, which is responsible for monitoring inputs and generating outputs based on the program's logic.

PLCs have become a critical component of modern industrial automation and are widely used in industries such as automotive manufacturing, food and beverage processing, and pharmaceutical production. They offer a high degree of flexibility and customization, allowing users to create highly tailored automation solutions to meet their specific needs.

PLCs work on a cycle of inputs, program execution, and outputs. Here is a general overview of how a PLC works:

1.Inputs: PLCs receive signals from sensors and other input devices, such as limit switches or temperature sensors. These inputs are used to monitor the status of equipment, such as whether a machine is on or off.

2.Program Execution: The PLC's processor executes a program stored in its memory. The program is created using a specialized programming language, such as ladder logic, which describes the logic and sequence of operations that the PLC should perform based on the inputs it receives.

3.Outputs: The PLC generates signals that are sent to output devices, such as motors or valves, to control the operation of equipment. The outputs are based on the logic defined in the program and the status of the inputs.

This cycle of inputs, program execution, and outputs continues in a loop as long as the PLC is powered on. PLCs can also communicate with other devices, such as human-machine interfaces (HMIs) or other PLCs, to exchange data and coordinate operations.

PLCs are highly configurable and can be customized to meet the specific needs of an industrial application. They are used to automate a wide range of processes, including assembly lines, material handling systems, and packaging operations, among others.

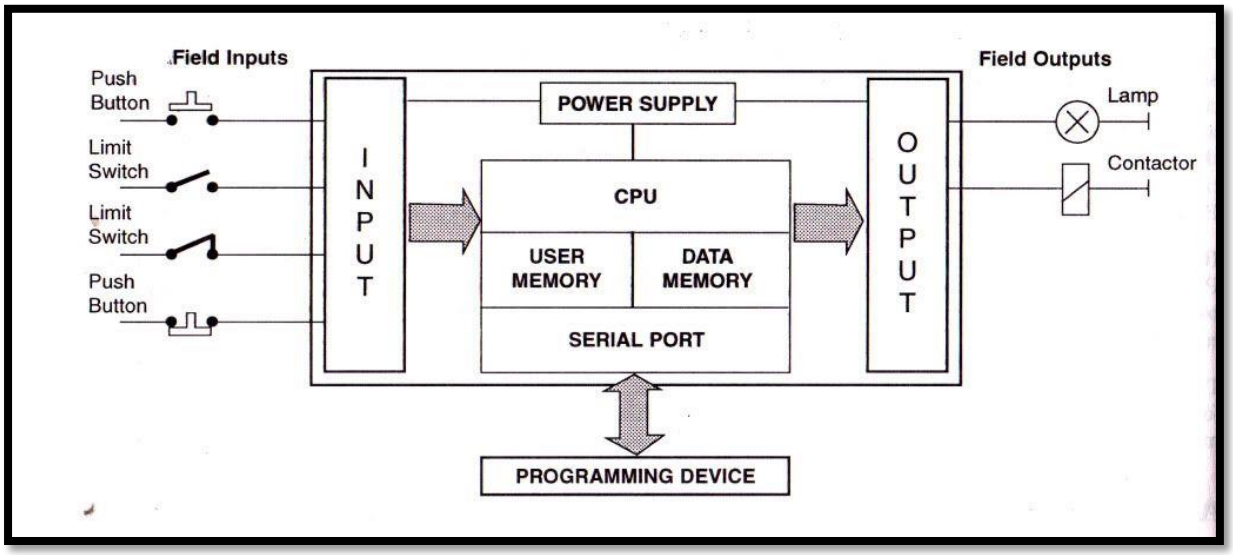
The voltage used by a PLC depends on its design and specifications, but most PLCs operate on a low voltage of **24V DC or 120V AC**.

The input signals received by a PLC are typically low voltage signals, such as 5V DC or 24V DC, generated by sensors or switches. These signals are then processed by the PLC's internal circuitry, which operates at a higher voltage level, typically 24V DC.

The output signals generated by the PLC to control devices such as motors, valves, or relays can operate at various voltage levels, depending on the specific application requirements. For example, a PLC may generate a 24V DC signal to control a motor or a 120V AC signal to control a solenoid valve.

It's important to note that PLCs are designed to be used with low voltage and current levels to ensure the safety of operators and equipment. They are not intended to be used with high voltage or current levels, which can pose a significant hazard to personnel and damage the PLC itself.

Working Module of Architecture of PLC



POWER SUPPLY

Digital i/p or O/p

CPU

Analog i/p or O/p

Communication Port

In industry these are used for communication port..

➔ RS232, RS432 , Modbus, Ethernet,etc…

5 Languages to write a code in PLC

1. **Ladder Diagram**
2. **Function Block Diagram**
3. **Structure Text**
4. **Instruction List**
5. **Sequential Function Chart**

How many types of Switches:

- ➔ **Mannual Switches**
- ➔ **Automatic Switches (Relay,….)**

- **Relay : ➔ Digital**

Relays are devices that allow a low-power signal to control a high-power circuit. They are commonly used in industrial control systems to control motors, lights, and other electrical equipment. Relays can be classified into different types based on their design, functionality, and application.

1.Electromechanical Relays: These are the most common type of relays and consist of a coil and one or more contacts. When the coil is energized, it creates a magnetic field that pulls the contacts together, completing a circuit. Electromechanical relays can be further classified into different subtypes, such as general-purpose relays, power relays, and latching relays.

These relays use a coil and one or more contacts to switch electrical circuits. The coil is energized by a low-voltage signal, typically ranging from 5 to 24 volts DC or AC. The contacts can switch circuits that operate at higher voltages, ranging from a few volts to hundreds of volts AC or DC.

2.Solid State Relays: These relays use semiconductor devices, such as transistors or thyristors, to control the flow of current. Solid state relays offer faster switching speeds and longer lifetimes than electromechanical relays, making them ideal for applications that require frequent switching or high reliability.

These relays use semiconductor devices, such as transistors or thyristors, to control the flow of current. Solid-state relays are typically rated for operation at low voltage levels, ranging from a few volts to around 32 volts DC. However, they can switch circuits that operate at higher voltages, such as 120VAC or 240VAC.

3.Reed Relays: These are specialized relays that use a reed switch as the contact mechanism. Reed relays are compact and have a high switching speed, making them useful in applications that require precise timing or limited space.

These relays use a reed switch as the contact mechanism. Reed relays are typically used in low-voltage circuits, such as those used in telecommunications or data transmission. They can operate at voltages ranging from a few volts to several hundred volts AC or DC.

4.Digital Relays: These are relays that operate on digital signals, such as those generated by a programmable logic controller (PLC) or a computer. Digital relays are commonly used in automation and control applications, where they can be programmed to perform complex logic functions.

These relays operate on digital signals, such as those generated by a programmable logic controller (PLC) or a computer. Digital relays are typically used in automation and control applications, where they can be programmed to perform complex logic functions. The voltage levels used by digital relays depend on the specific application requirements.

5.Analog Relays: These relays are designed to control analog signals, such as those generated by sensors or other measurement devices. Analog relays can be used to control the level, frequency, or

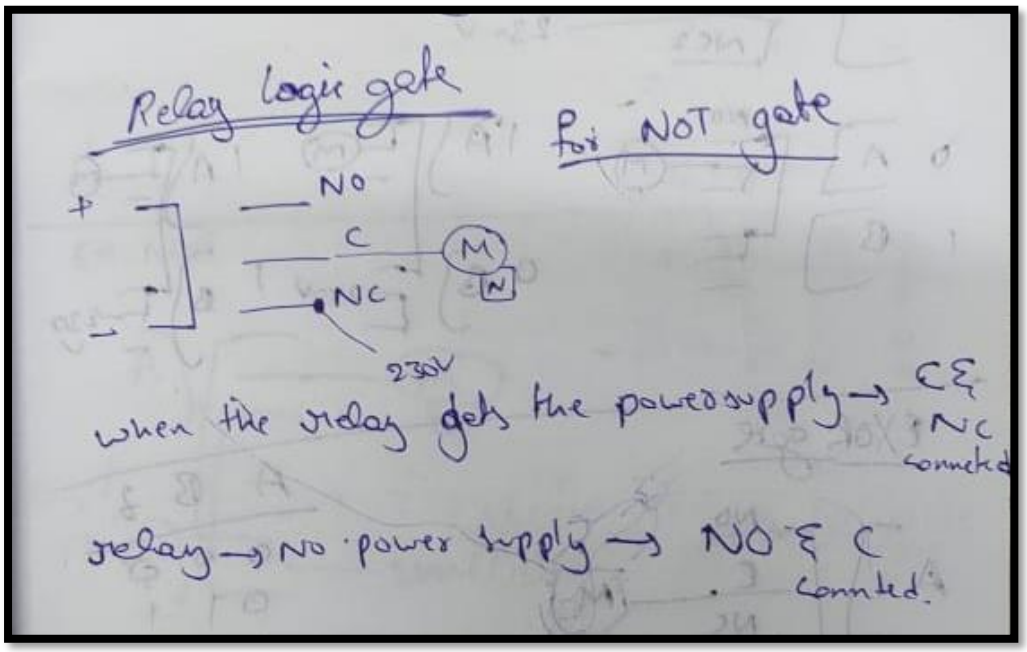
amplitude of an analog signal, making them useful in applications such as audio processing or power control.

These relays are designed to control analog signals, such as those generated by sensors or other measurement devices. Analog relays can switch circuits that operate at low voltage levels, ranging from a few volts to several hundred volts AC or DC, depending on the specific application.

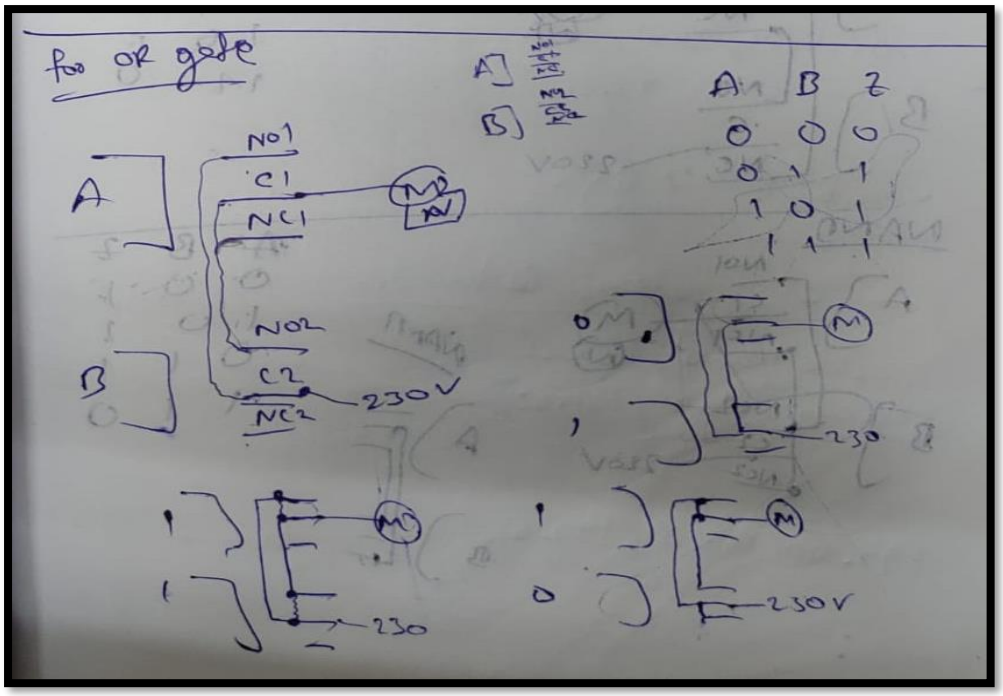
The specific type of relay used depends on the requirements of the application, such as the voltage and current levels, the switching speed, and the environmental conditions.

Desigining the Logic Gates using Relay Logic Diagrams

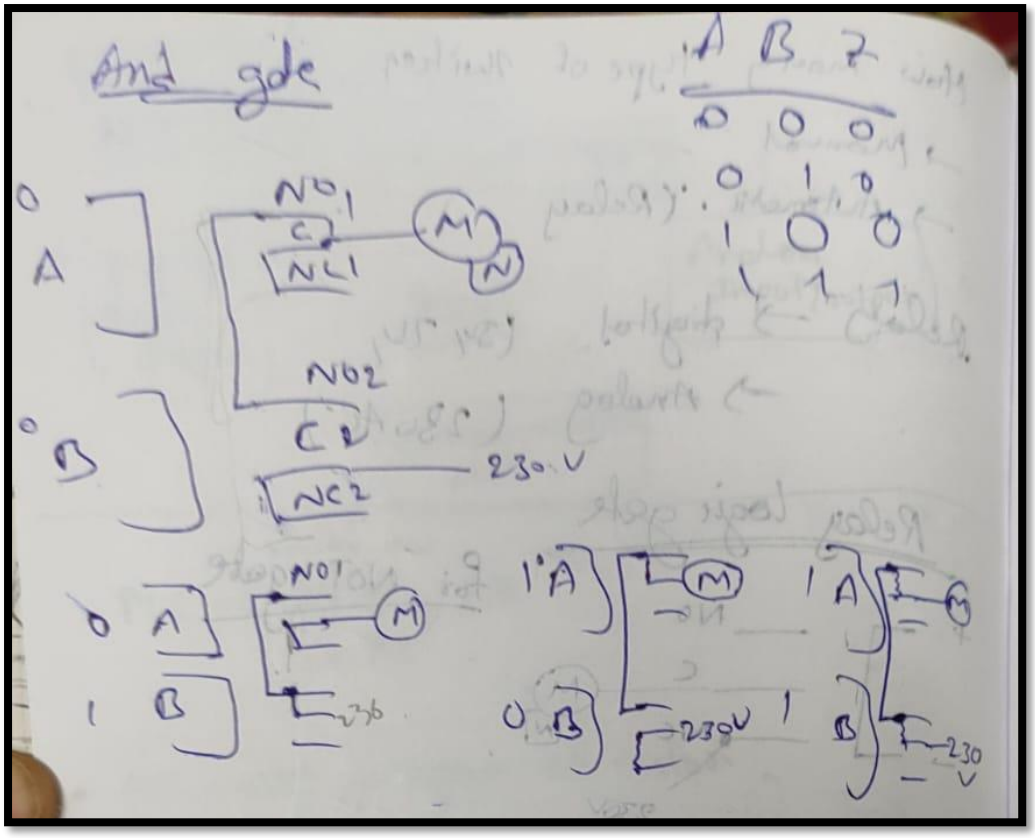
NOT Gate :



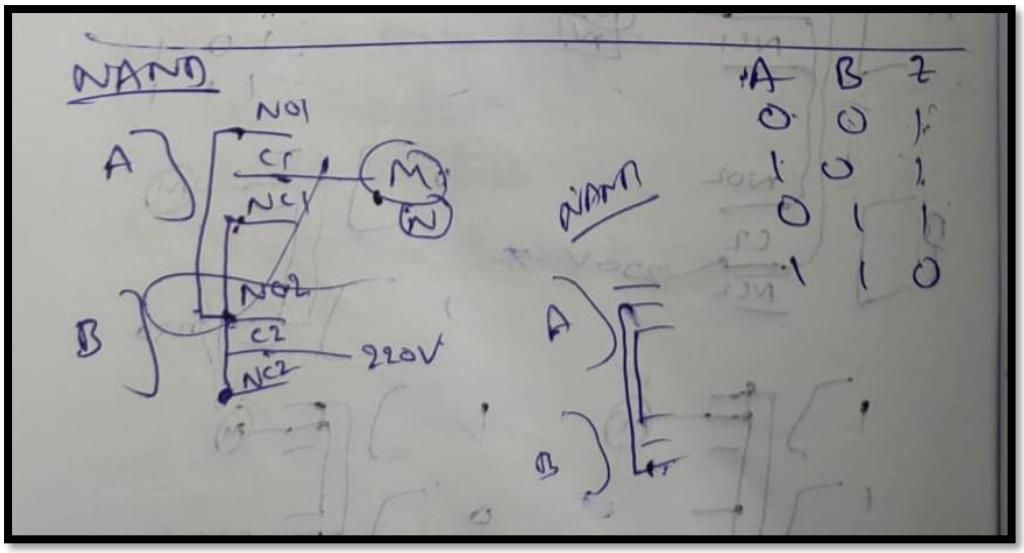
OR Gate :



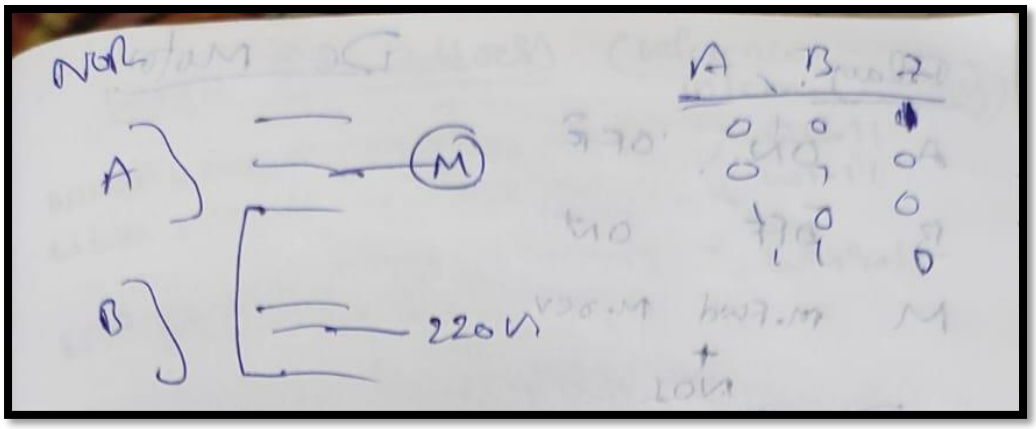
AND Gate :



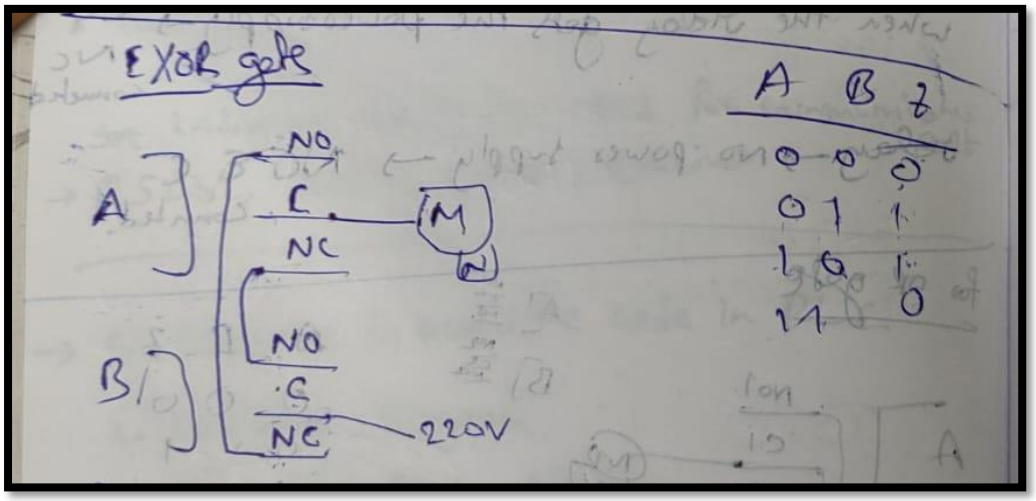
NAND Gate :



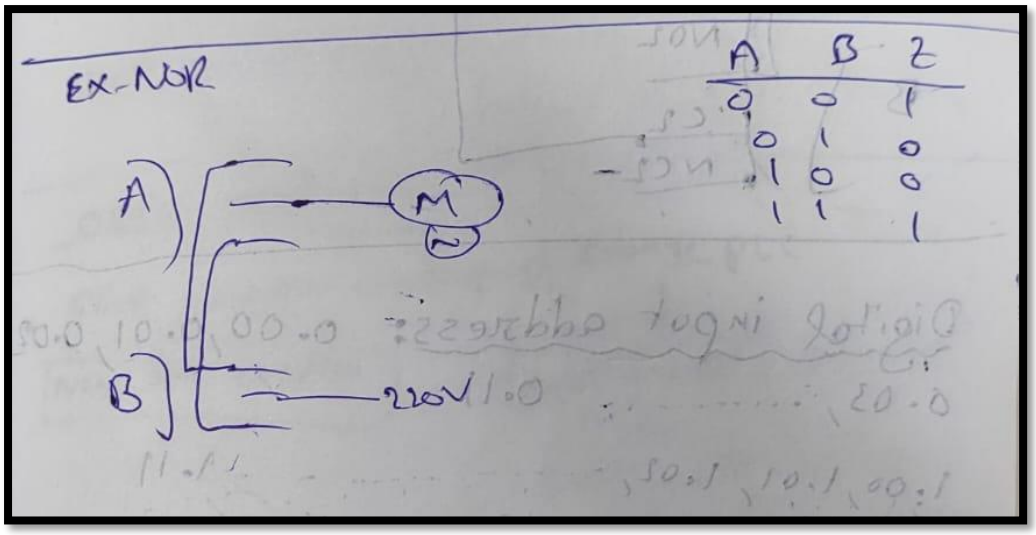
NOR Gate :



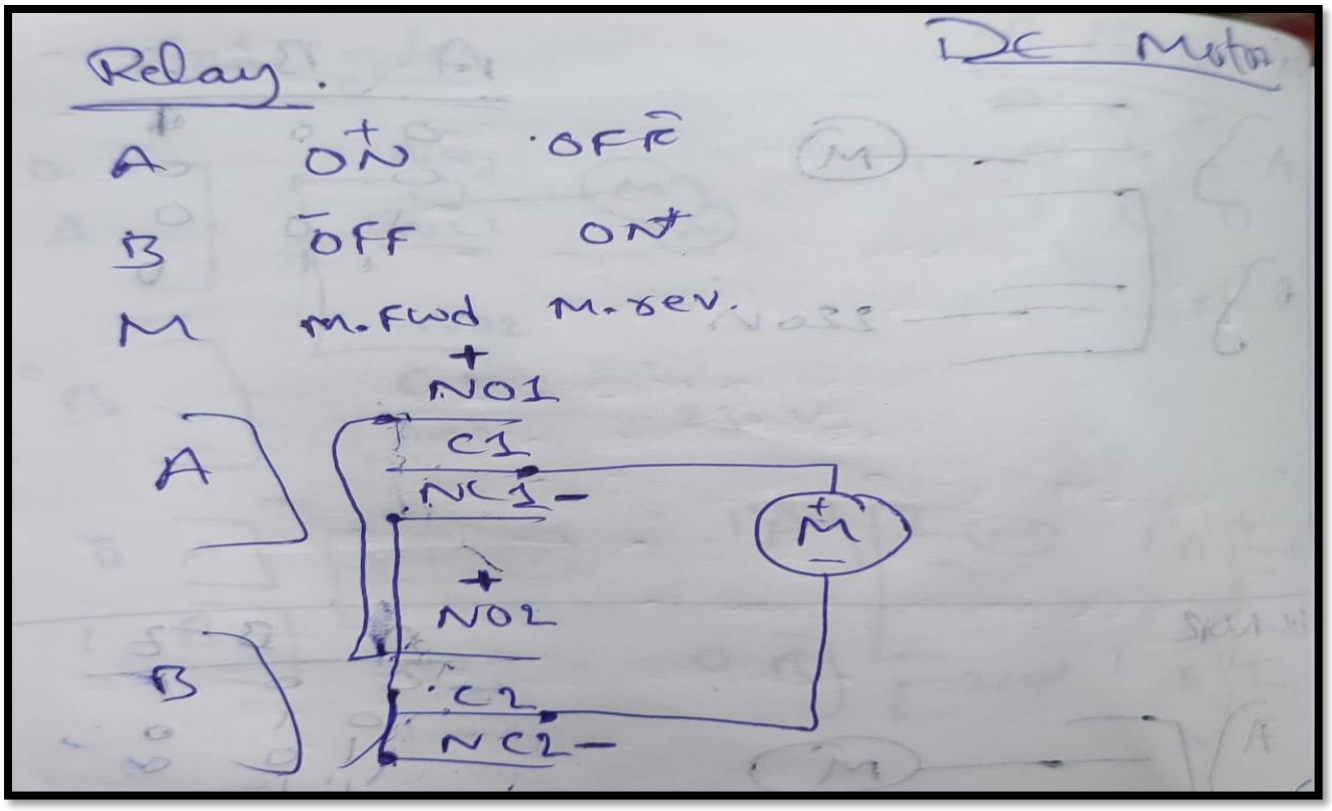
EX-OR Gate :



EX-NOR Gate :



DC Motor :



Digital input Address :

0.00 , 0.01 , 0.02 , 0.03 , ... , 0.11

1.00 , 1.01 , 1.02 , 1.03 , ... , 1.11

. . . . . .

. . . . . .

99.00 , 99.01 , 99.02 ,99.03 , ... , 99.11

Digital Output Address :

100.00 , 100.01 , 100.02 , ... 100.07

101.00 , 101.01 ,101.02 , ... 101.07

102.00 , 102.01 ,102.02 , ... , 102.07

. . . . . .

. . . . . .

199.00 , 199.01 , 199.02 , ... , 199.07

Work Bit Address :

W0.00 , W0.01 , W0.02 , ... , W0.11

W1.00 , W1.01 , W1.02 , ... , W1.11

. . . . . .

. . . . . .

W49.00, W49.01 , W49.02 , ... ,W49.11

Holding Bit Address :

H0.00 , H0.01 , H0.02 , ... , H0.11

H1.00 , H1.01 , H1.02 , ... , H1.11

. . . . . .

. . . . . .

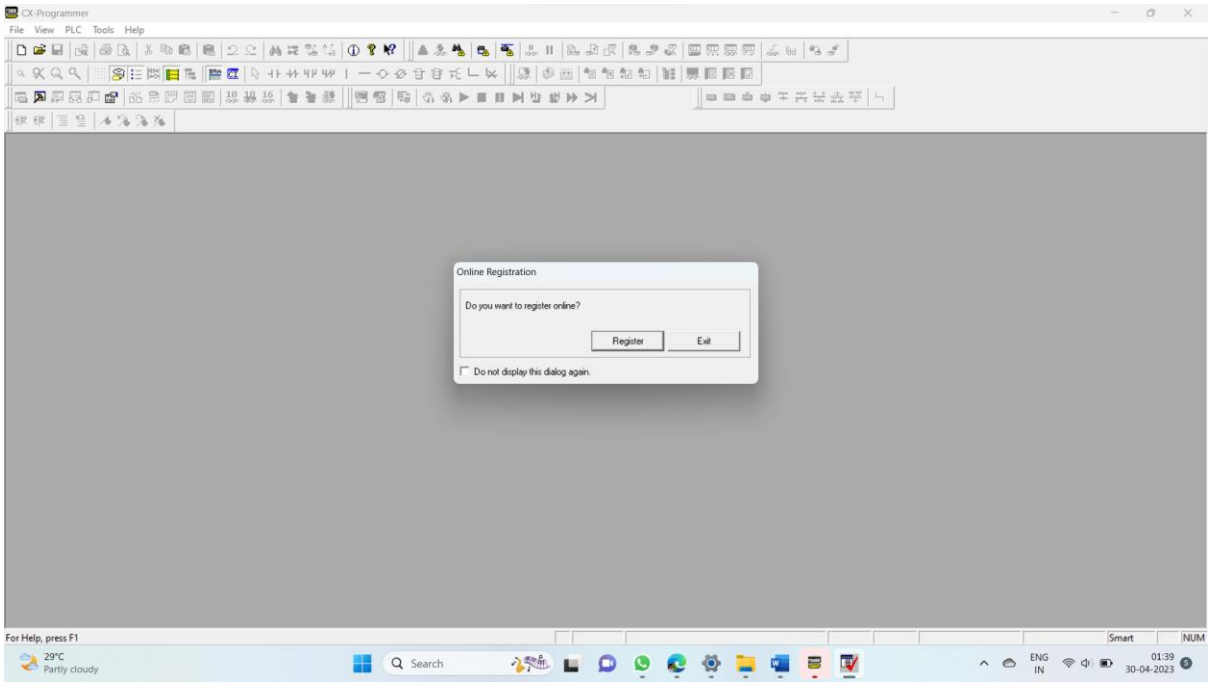
H49.00 , H49.01 , H49.02 , ... , H49.11

Open Application

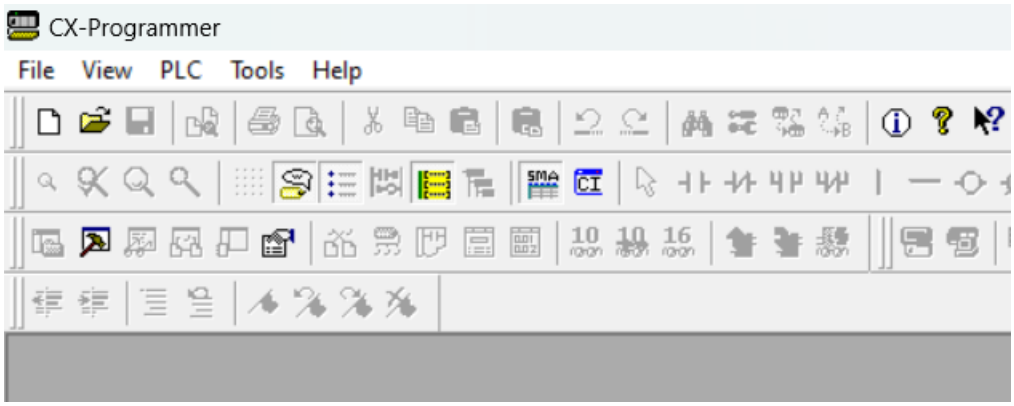
File → New → *(pop-up)* ***Change PLC → Device Name***

- Open CX-P software

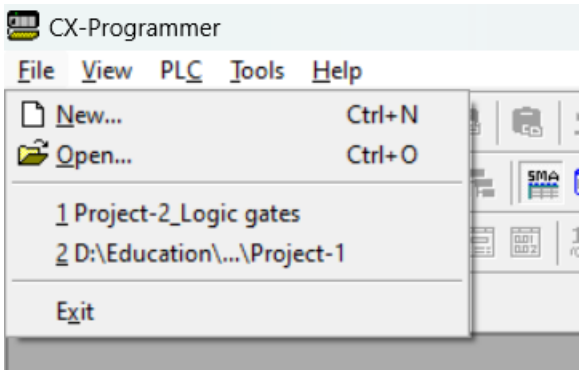
- One pop-up appears . Press “EXIT”



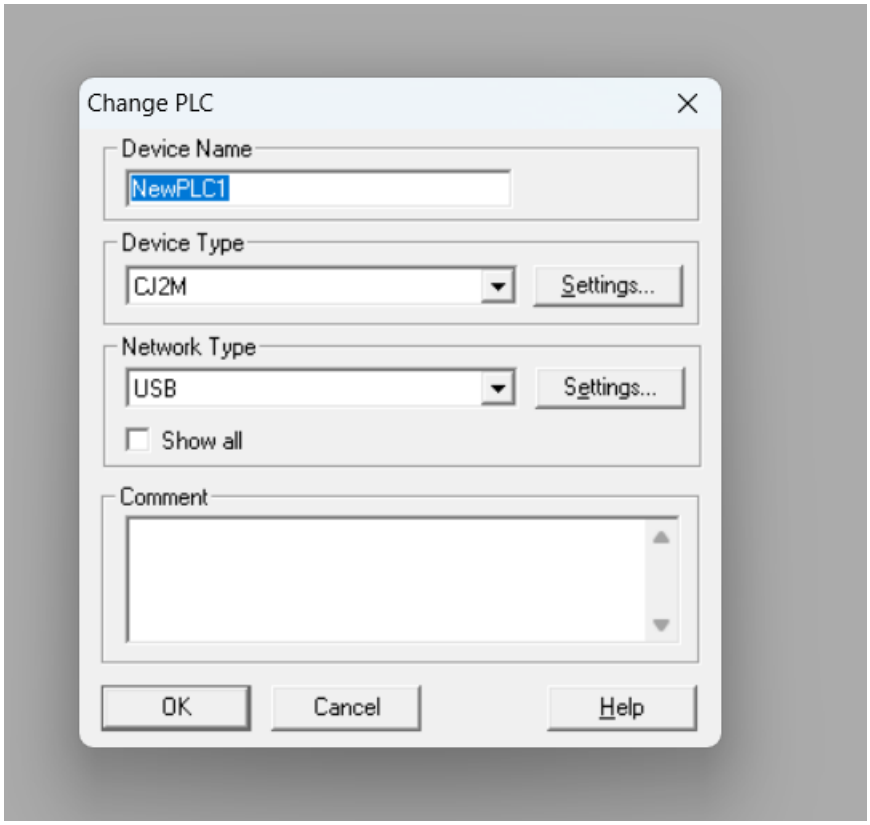
- Let’s create a “NEW FILE” . Click on “FILE”.



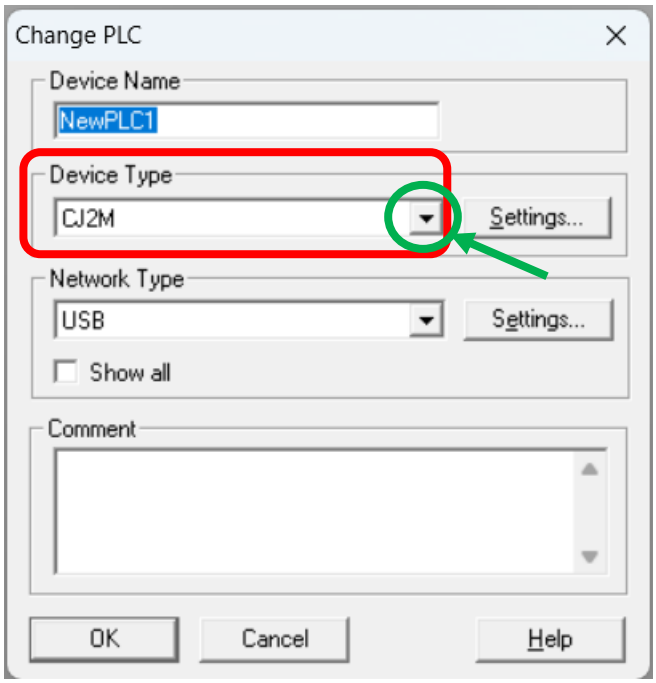
- Select “NEW”. Otherwise you can open Previous files.

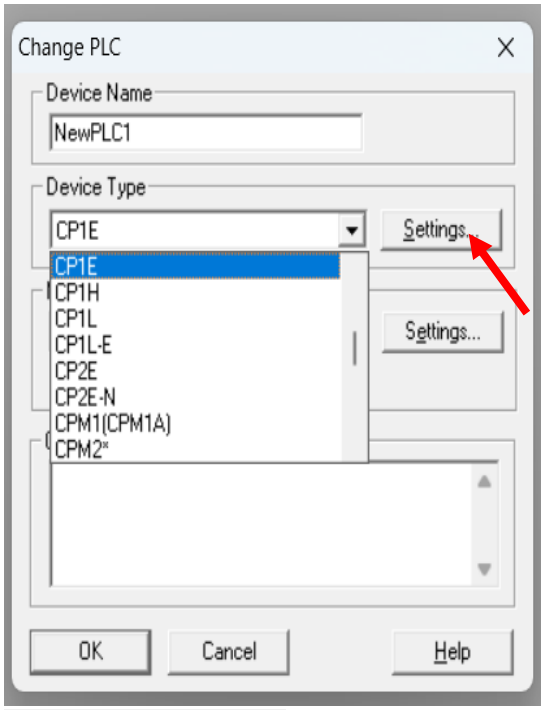


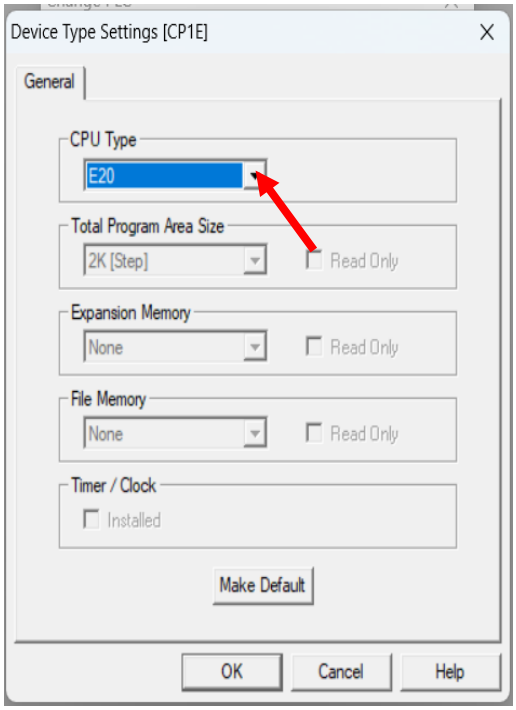
- After clicking “NEW” . A pop-up appears “Change PLC”.

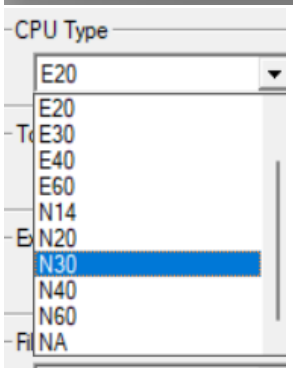


- Change “Device Name” as your wish.
- Device Type → “**CPIE**”. (Settings → CPUtype → N30 ,Click Ok)
- Network Type → USB (present we use USB. You can connect to Ethernet).

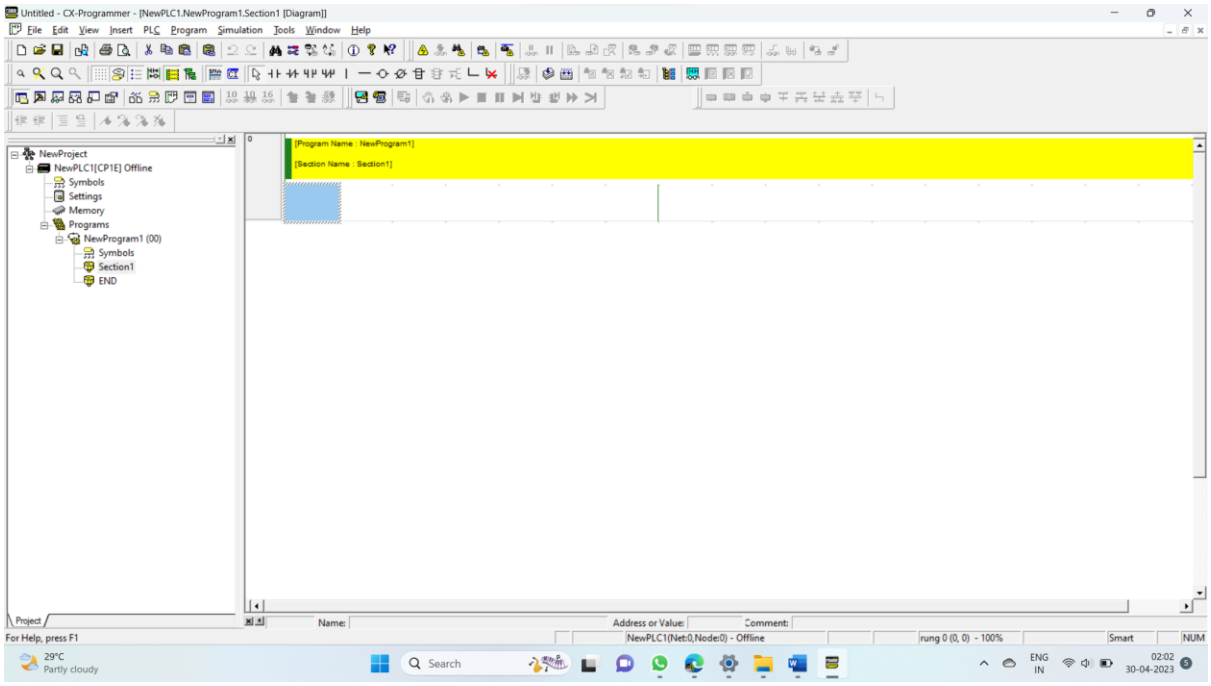




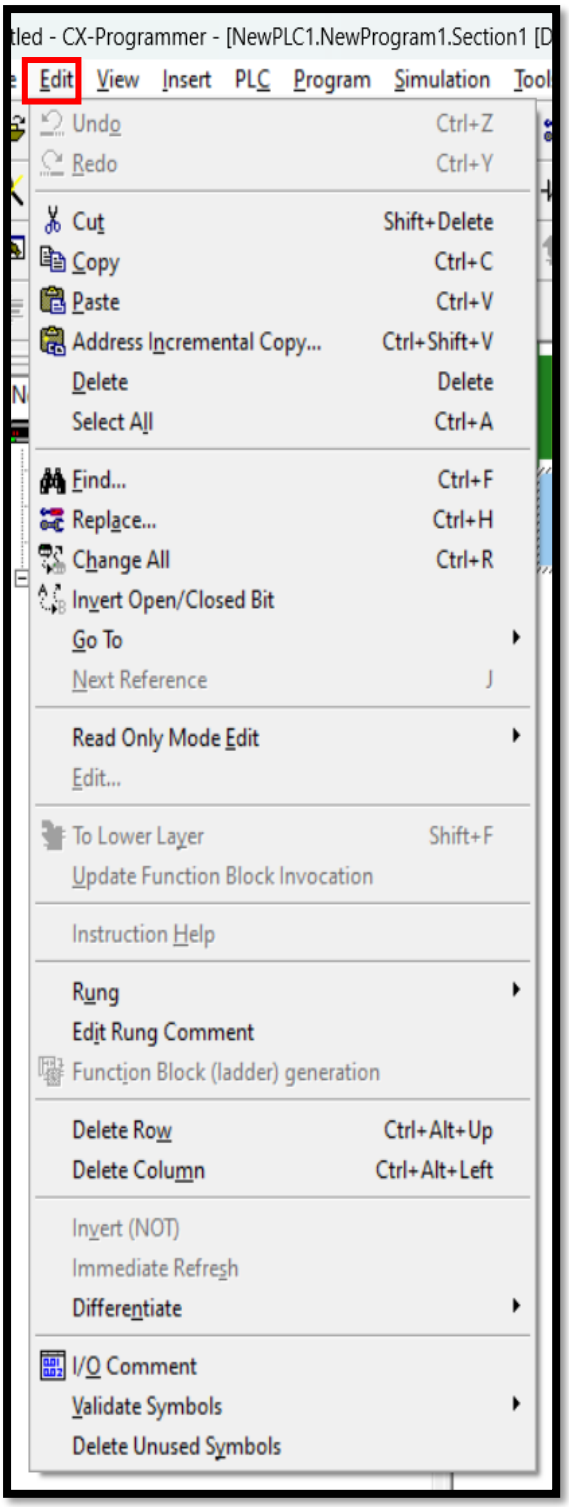


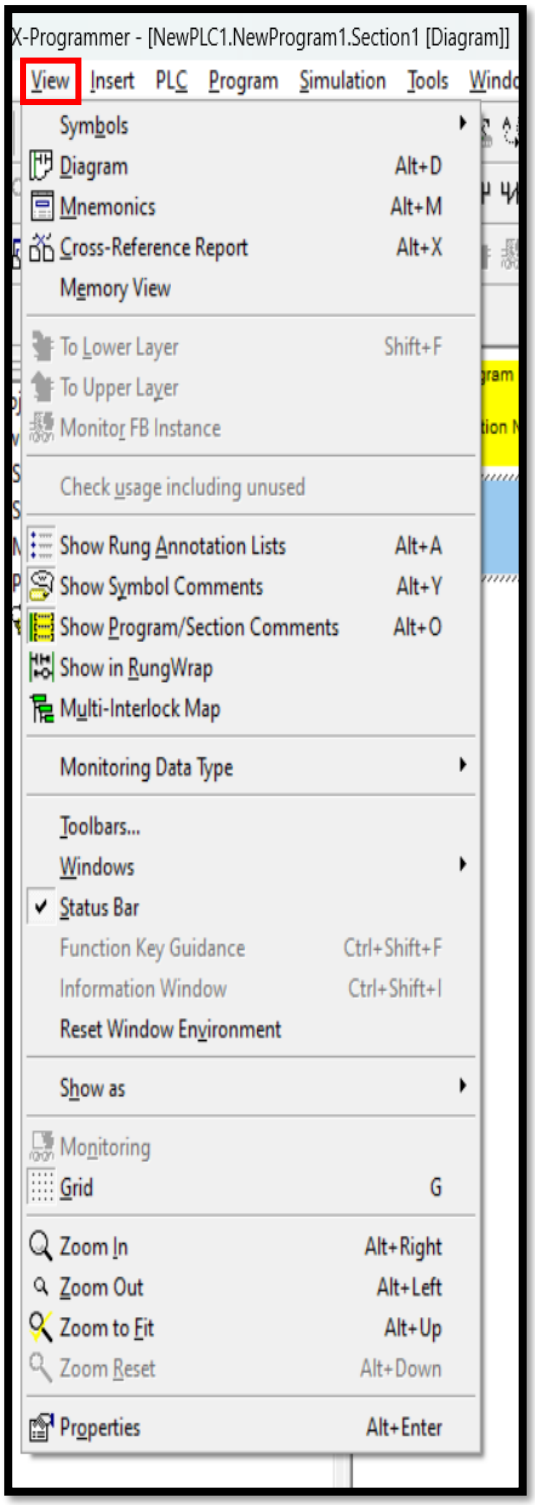


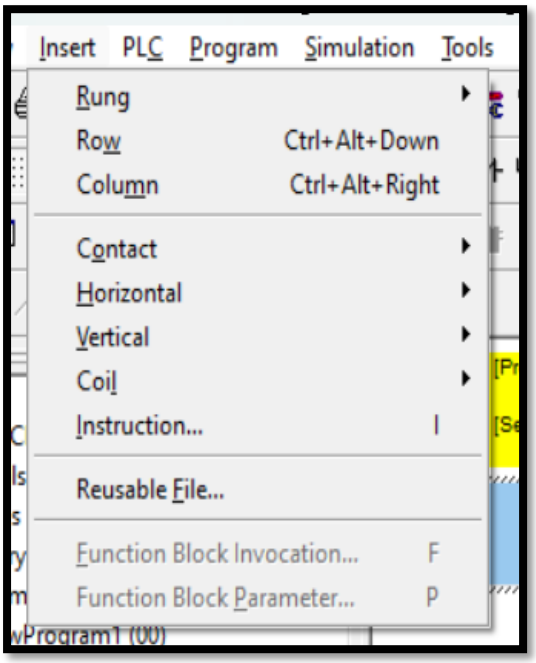
- Click OK
- A Pop-up Window arrives

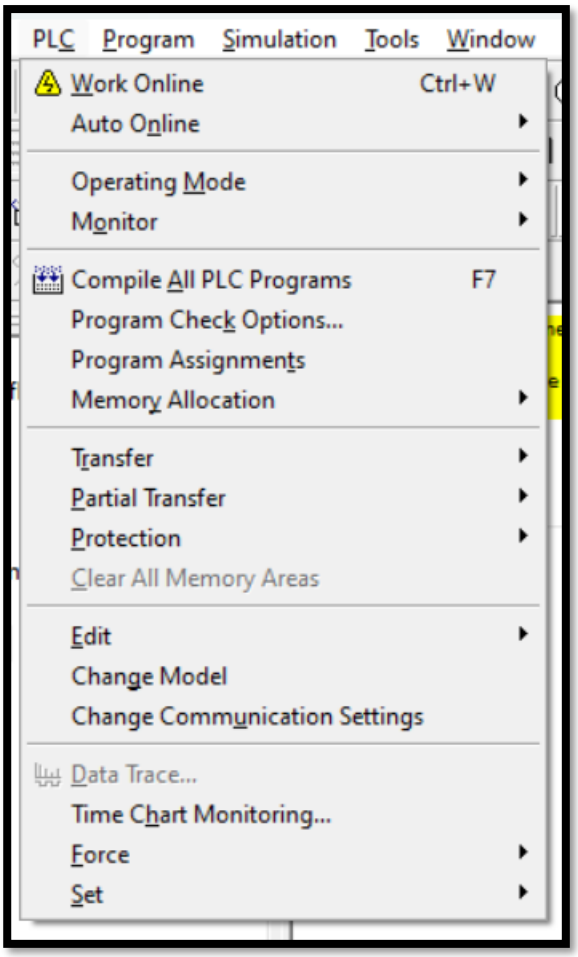


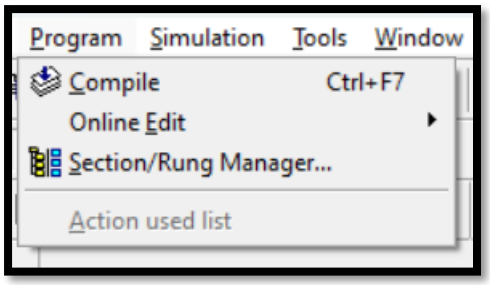
- Here all the operations are to be done. All logics are written here.

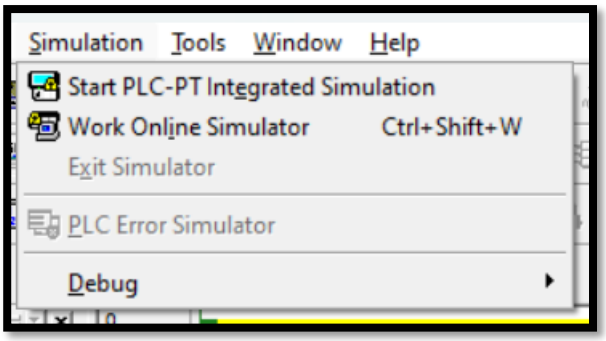

tled - CX-Programmer - [NewPLC1.NewProgram1.Section1 [D
Edit View Insert PLC Program Simulation Tool
Undo Ctrl+Z
Redo Ctrl+Y
Cut Shift+Delete
Copy Ctrl+C
Paste Ctrl+V
Address Incremental Copy... Ctrl+Shift+V
Delete Delete
Select All Ctrl+A
Find... Ctrl+F
Replace... Ctrl+H
Change All Ctrl+R
Invert Open/Closed Bit
Go To
Next Reference J
Read Only Mode Edit
Edit...
To Lower Layer Shift+F
Update Function Block Invocation
Instruction Help
Rung
Edit Rung Comment
Function Block (ladder) generation
Delete Row Ctrl+Alt+Up
Delete Column Ctrl+Alt+Left
Invert (NOT)
Immediate Refresh
Differentiate
I/O Comment
Validate Symbols
Delete Unused Symbols

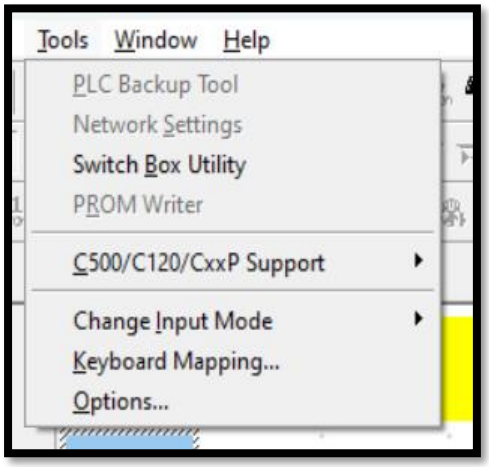

X-Programmer - [NewPLC1.NewProgram1.Section1 [Diagram]]
View Insert PLC Program Simulation Tools Windo
Symbols
Diagram Alt+D
Mnemonics Alt+M
Cross-Reference Report Alt+X
Memory View
To Lower Layer Shift+F
To Upper Layer
Monitor FB Instance
Check usage including unused
Show Rung Annotation Lists Alt+A
Show Symbol Comments Alt+Y
Show Program/Section Comments Alt+O
Show in RungWrap
Multi-Interlock Map
Monitoring Data Type
Toolbars...
Windows
Status Bar
Function Key Guidance Ctrl+Shift+F
Information Window Ctrl+Shift+I
Reset Window Environment
Show as
Monitoring
Grid G
Zoom In Alt+Right
Zoom Out Alt+Left
Zoom to Fit Alt+Up
Zoom Reset Alt+Down
Properties Alt+Enter

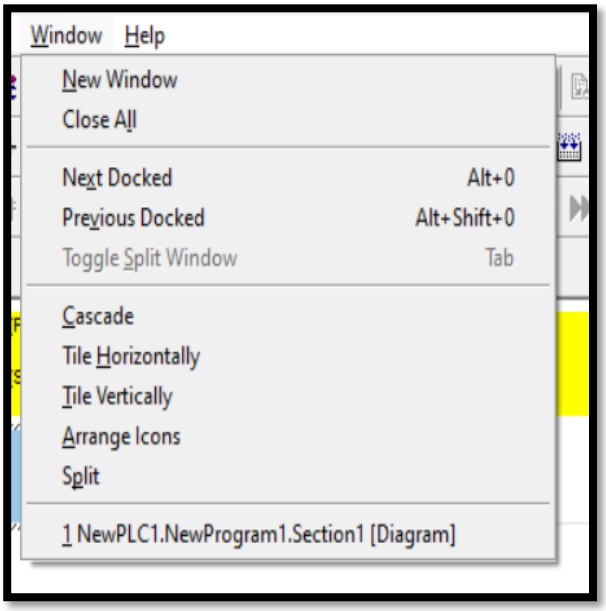

Insert PLC Program Simulation Tools
Rung
Row Ctrl+Alt+Down
Column Ctrl+Alt+Right
Contact
Horizontal
Vertical
Coil
Instruction... I
Reusable File...
Function Block Invocation... F
Function Block Parameter... P
Program1 (00)

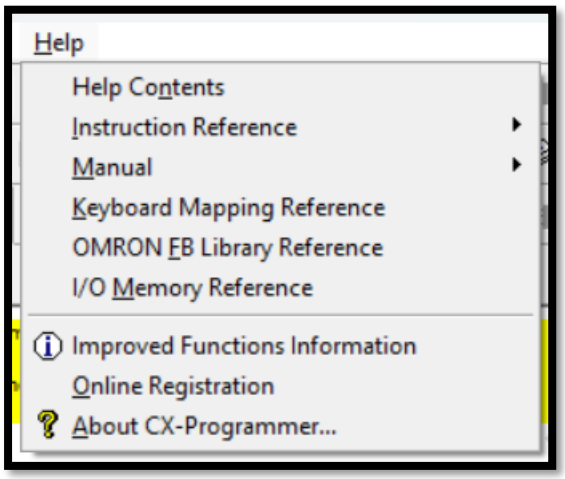

PLC Program Simulation Tools Window
Work Online Ctrl+W
Auto Online
Operating Mode
Monitor
Compile All PLC Programs F7
Program Check Options...
Program Assignments
Memory Allocation
Transfer
Partial Transfer
Protection
Clear All Memory Areas
Edit
Change Model
Change Communication Settings
Data Trace...
Time Chart Monitoring...
Force
Set





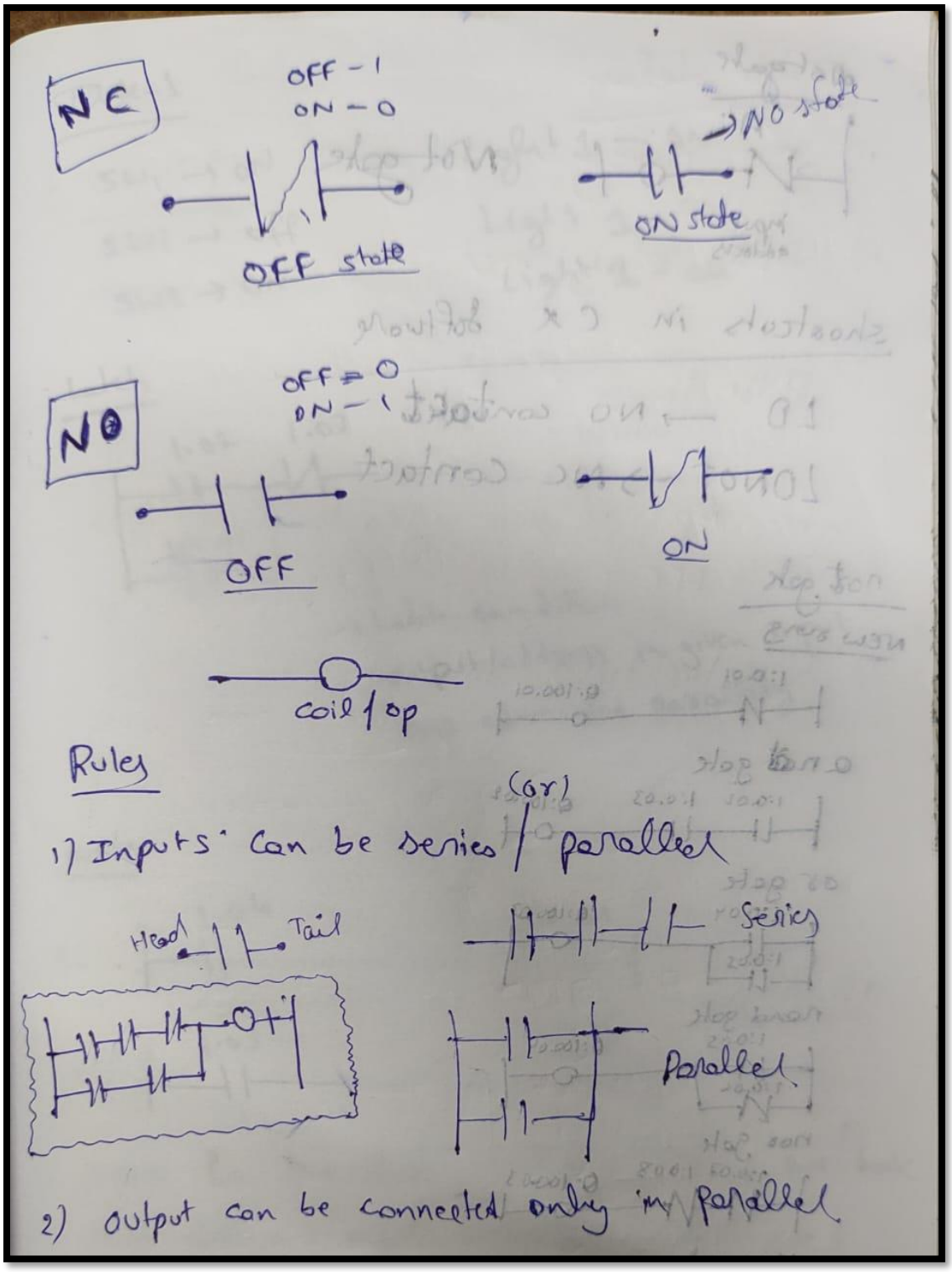






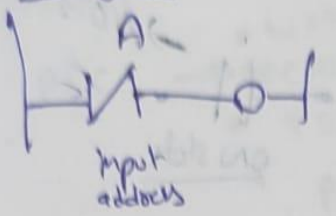
- Rung → new Line
- Compile
- Work Online Simulator [Ctrl + W].

Components used to design the logic [NO contact , NC contact]…



Rules for NO & NC Contacts are given above..
Not gate given below. And some shortcuts to NO & NC in CX-P software.

Not gate



shortcuts in CX software

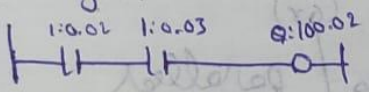
LD → NO contact

LDNOT → NC contact

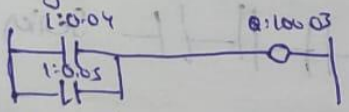
not gate

NEW RUNG

and gate

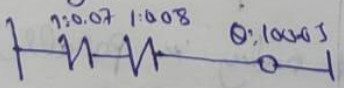


or gate

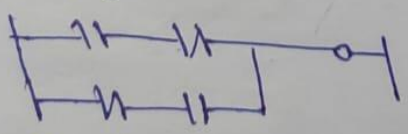


nand gate

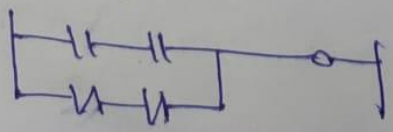
nor gate



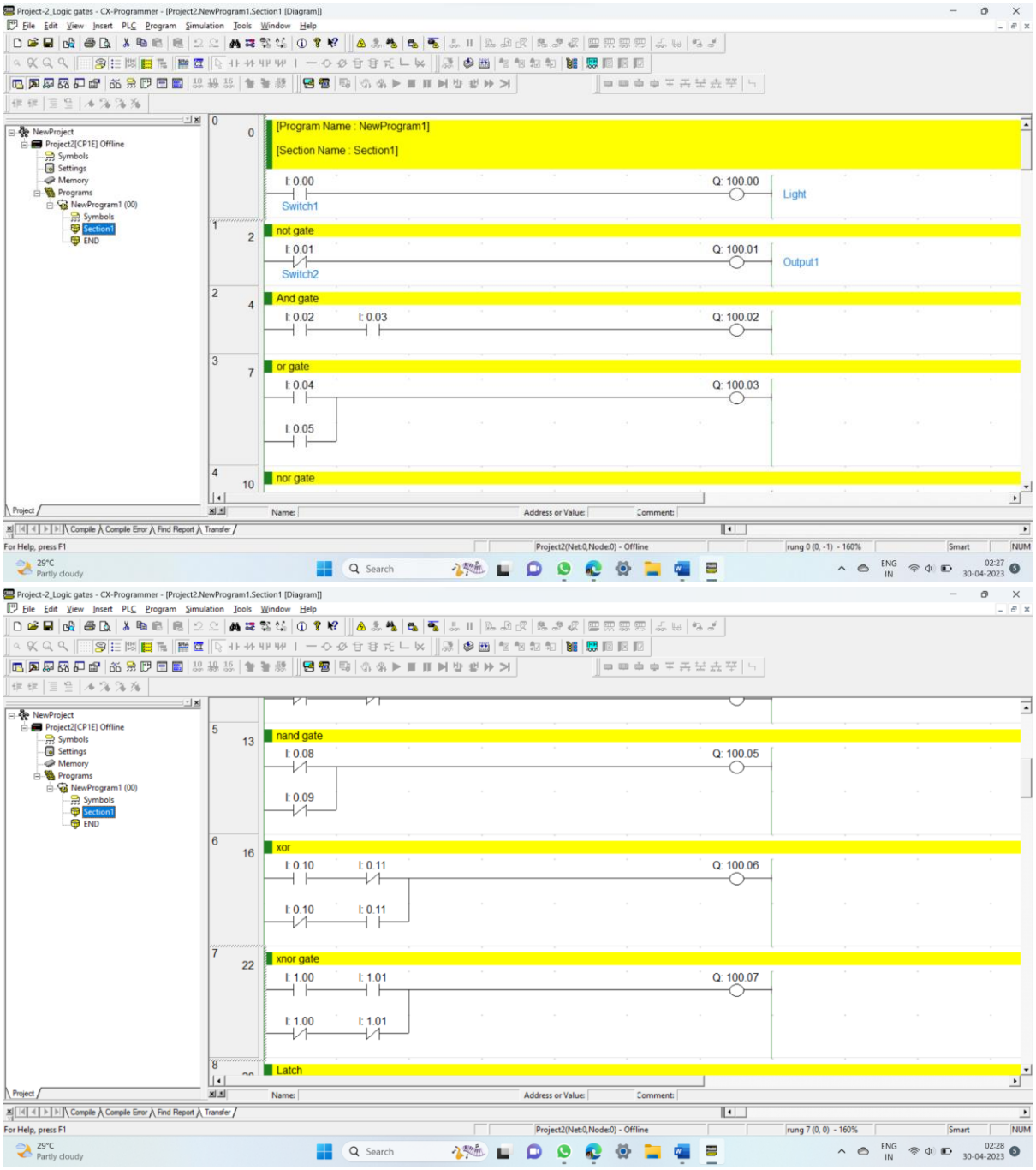
xor gate



xnor gate



Desigined the Logic Gates using the Ladder Logic diagram..



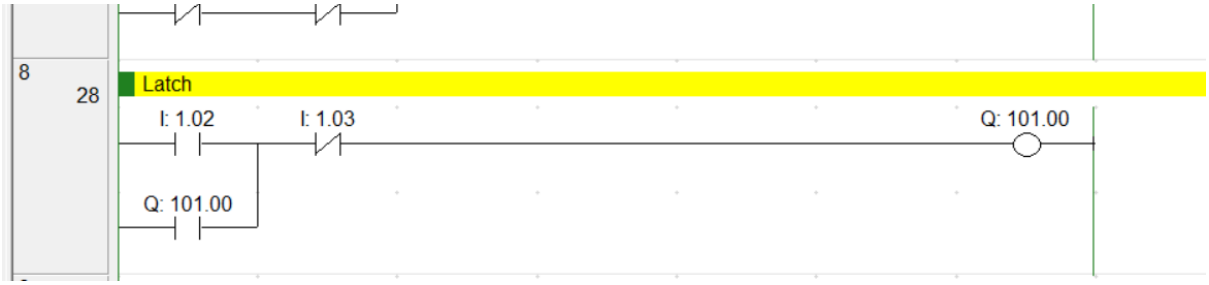
TASK-1 :

When SW1 → ON then Light1→ ON

SW1 → OFF then Light1→ ON

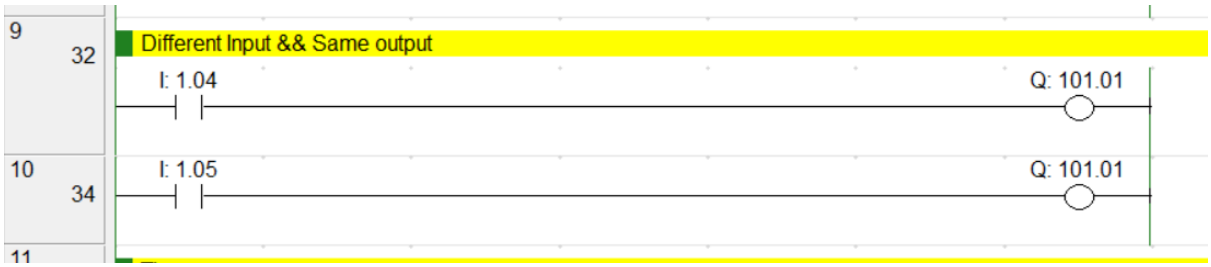
SW2 → ON then Light1→OFF

Ans : **"Latch"** concept



→Latch condition is “Output address is given to input” [NO is connected parallelly.]

Note:



Hence for two different “rungs”(new line), we took two different inputs but same output.

Operation:

When 1.04 → ON (1) then 1 output is ON.But 2 output is OFF although it has some address.

When 1.05 → ON (1) then 1 output is ON. But 2 output is also ON. Because the PLC program runs from top to bottom.

So, When simulation takes place from top to bottom executes.1st also got the output when 2nd output switch is ON.

TIME DELAY:

1.ON delay timer:
2.OFF delay timer:
3)Pulse timer:
4).Retentive timer/totalising timer:

Timer No.: 000-256 timers
Timer base:
1ms timer,**10ms** timer,**100ms** timer, **1sec**ond timer, **1min**ute timer.

Timer syntex:

- **On delay timer:**

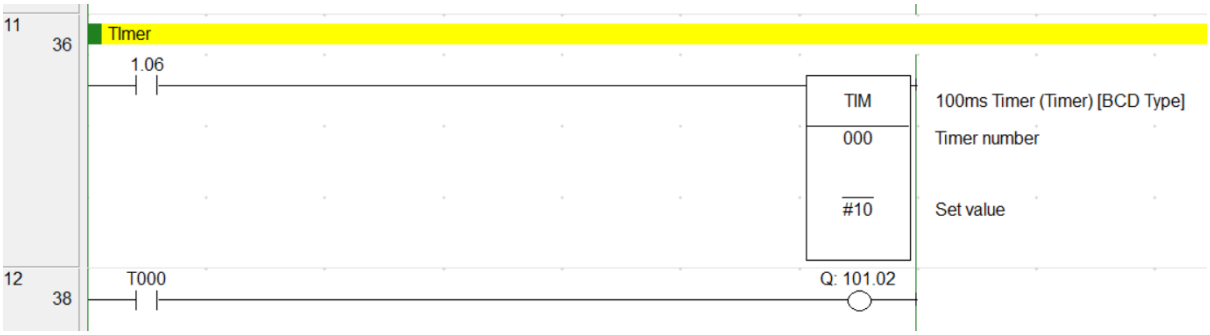
  TIM 000 #10 TIM → 100ms timer
  000→ Timer Number
  #10 → delay time

  Note : delay time should not exceed 65335.
  In the syntex #delay not written it shows an error.

- **Totalisining Timer:**

  TTIM 001 #20

➪ *ON delay timer examples:*



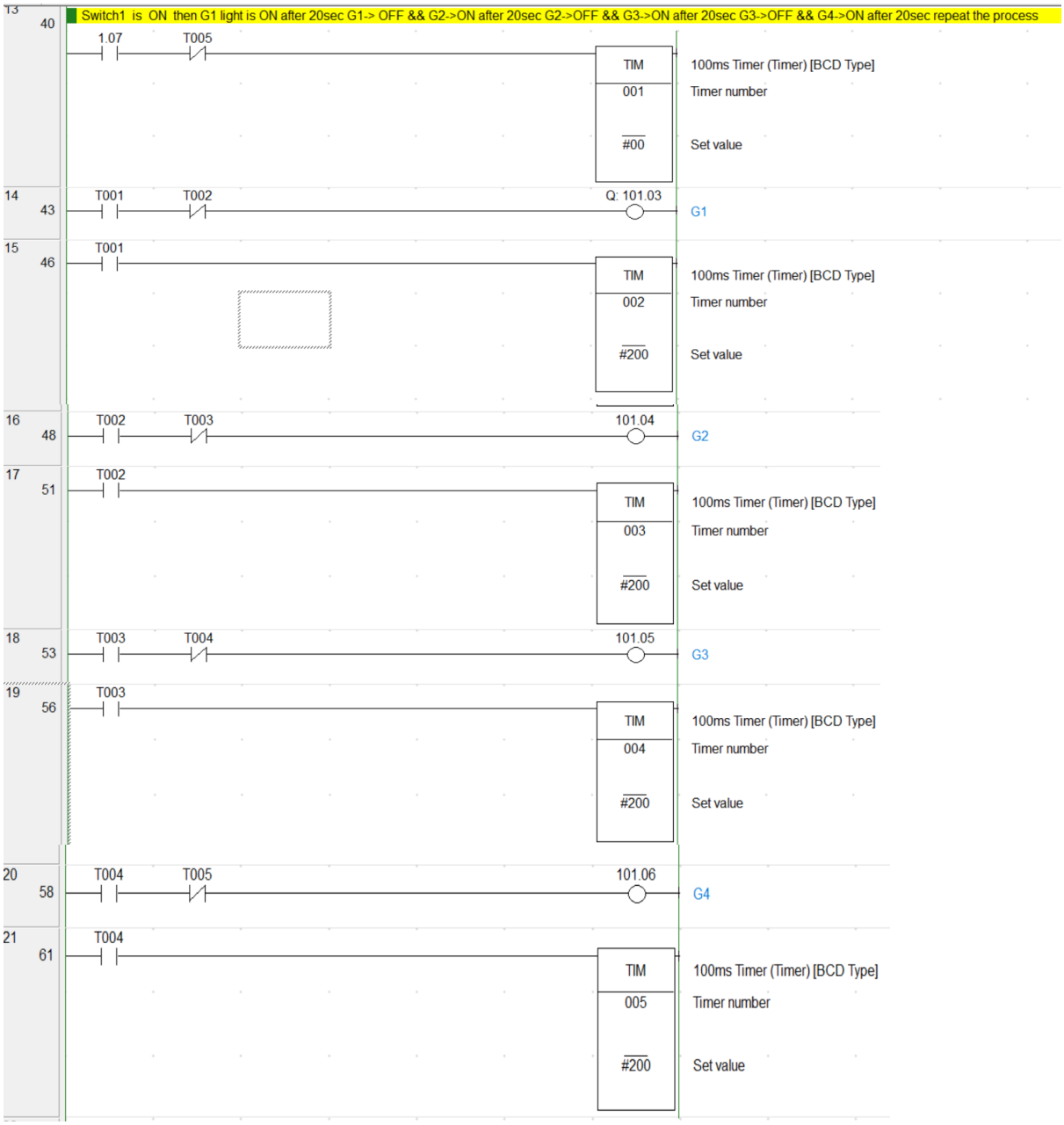
When timer has input then only output is ON.

TASK-2

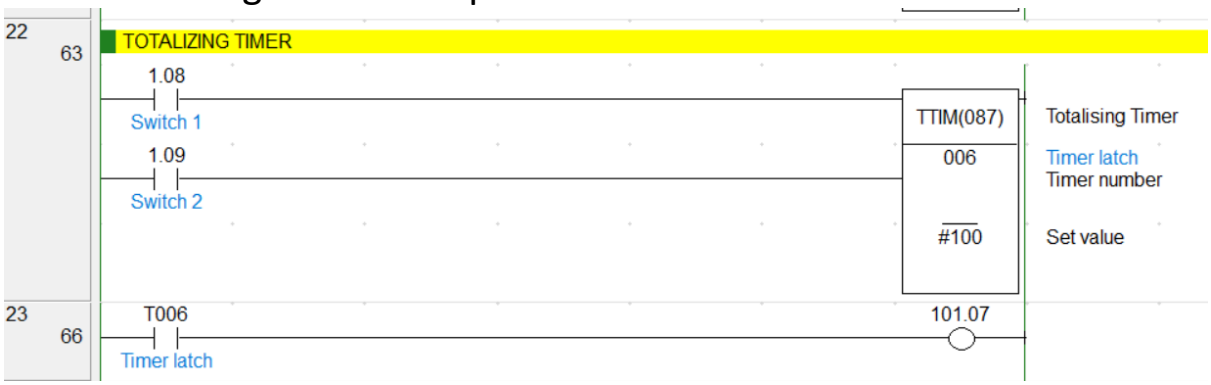
When SWITCH is ON →

- G1 → ON for 20sec
- G1 → OFF
- G2 → ON for 20sec
- G2 → OFF
- G3 → ON for 20sec
- G3 → OFF
- G4 → ON for 20sec
- G4 → OFF
- Loops repeates

Hint: use timer with latch concepts..



➪ Totalising timer examples:



Here when “Switch 1” is ON ,then the timer starts counting from “000 to highest value(100 given)” and then output is ON when timer reaches max. value..

For resetting the timer we should turn ON the “Switch 2”,then it will be reset to 000.

Application:

1.Stop watches,…..
2.The case is in an industry, The machine is running based on timer.

Suppose timer count is 60 out of 100.

Suddenly power goes OFF. then the machine is OFF.

When after the power came, the timer works from starting then it will be waste of time. In order to avoid that “Totalising timer” is used. Because it counts from where it stopped previously.

Note: But in ON delay timer,after the power gone OFF & when it Came then the timer counts from starting..

COUNTER:

1.UP counter
2.Reversible counter / UP- Down Counter

Counter No(000 – 255)

1.UP counter:

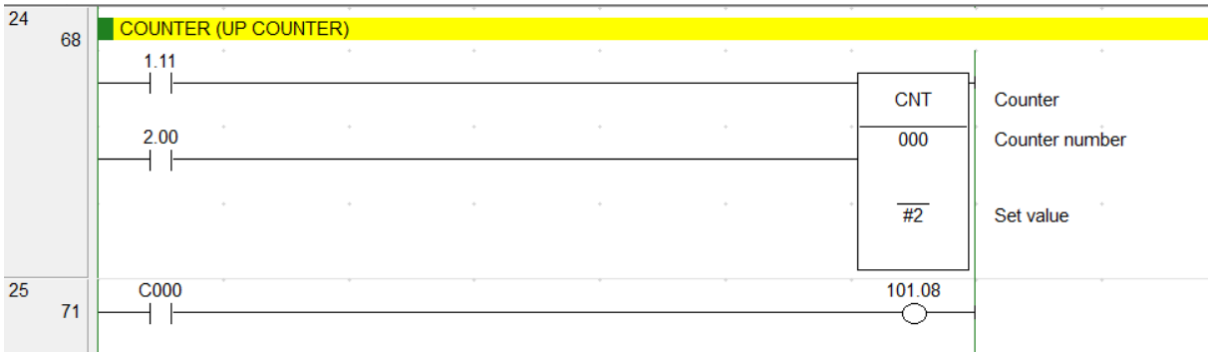
Syntex: CNT 000 #2

CNT → Counter
000 → Counter number
#2 → counter value (max count should be 2 after that count will not increased).

2.Reversible counter / Up-down counter :

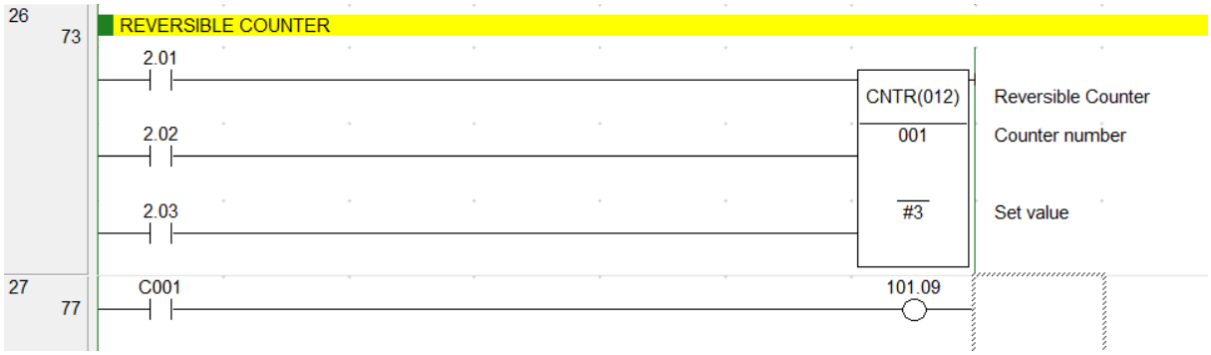
Syntex: CNTR 001 #3

CNTR →Reversible counter
001 → Counter number
#3 → Max count value



Here in Up counter, when **Switch1(**1.11**)** ON then the count decrease from max value by -1.like that switch ON & OFF takes place the count becomes “Zero” [0] then the output is ON.

Switch2 (2.00) is ON counter is to be reset. Then counter value set max.

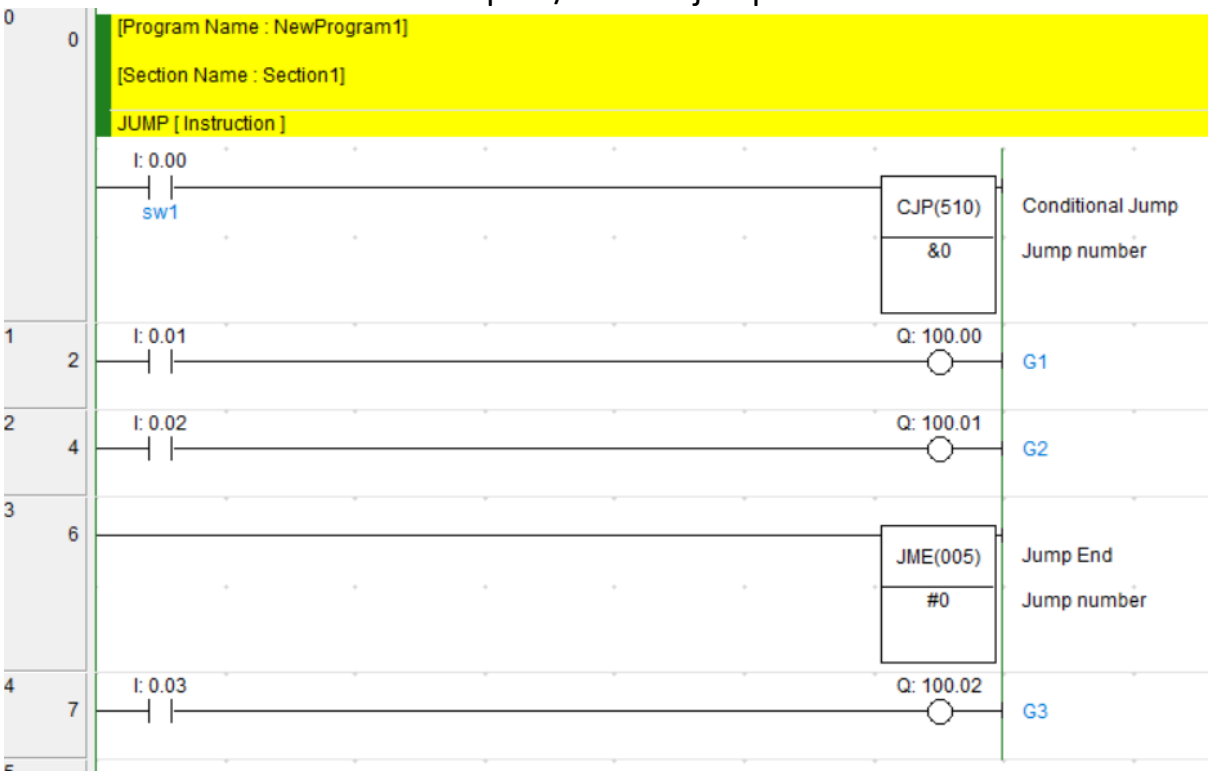


Here in reversible counter 2.01 → count value increase to max.
2.02 → count value decrease
2.03 → reset

The out is ON when count value is at Zero .

JUMP:

Syntex: CJ &234 CJ/CJP/CJPN → Jump instruction
&012 → Address
JME 234 →Jump to /end the jump



When ever JMP switch is ON
Here 2 & 3 rungs will not be executed by PLC
i.e it will be skipped upto JMP end line
When JMP Switch is OFF
Then the 2 & 3 rungs will be executed.

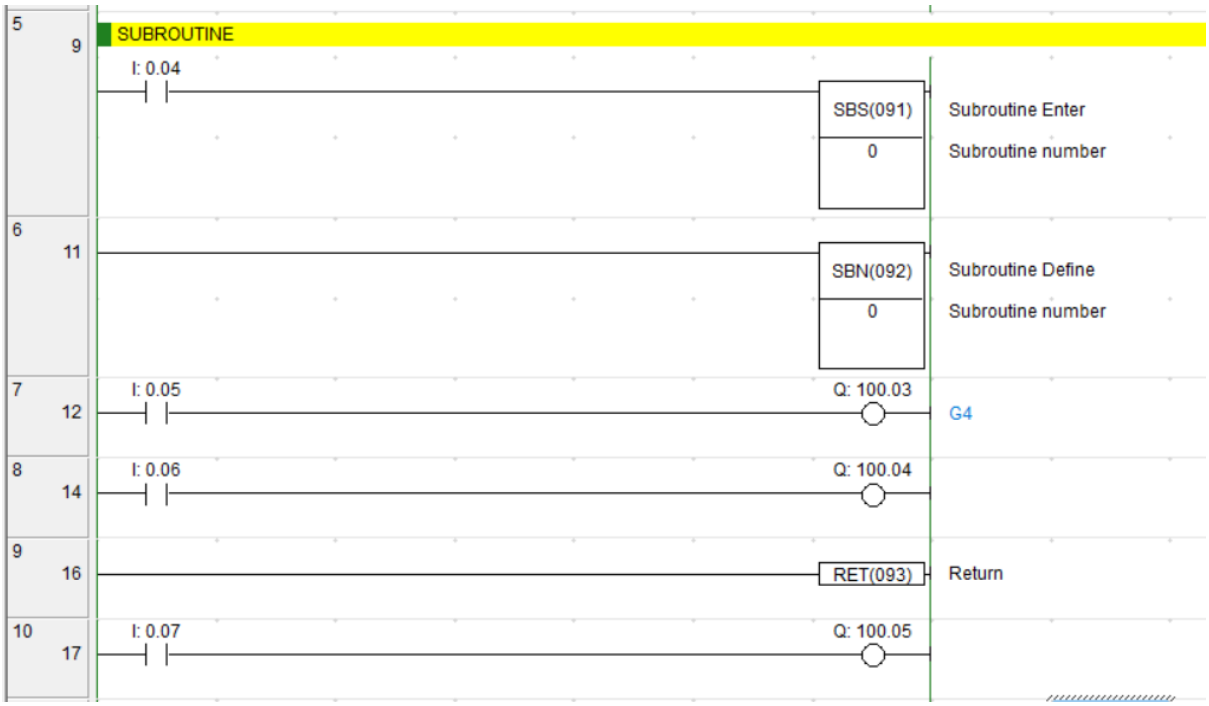
SUBROUTINE

Syntex: SBS 0123

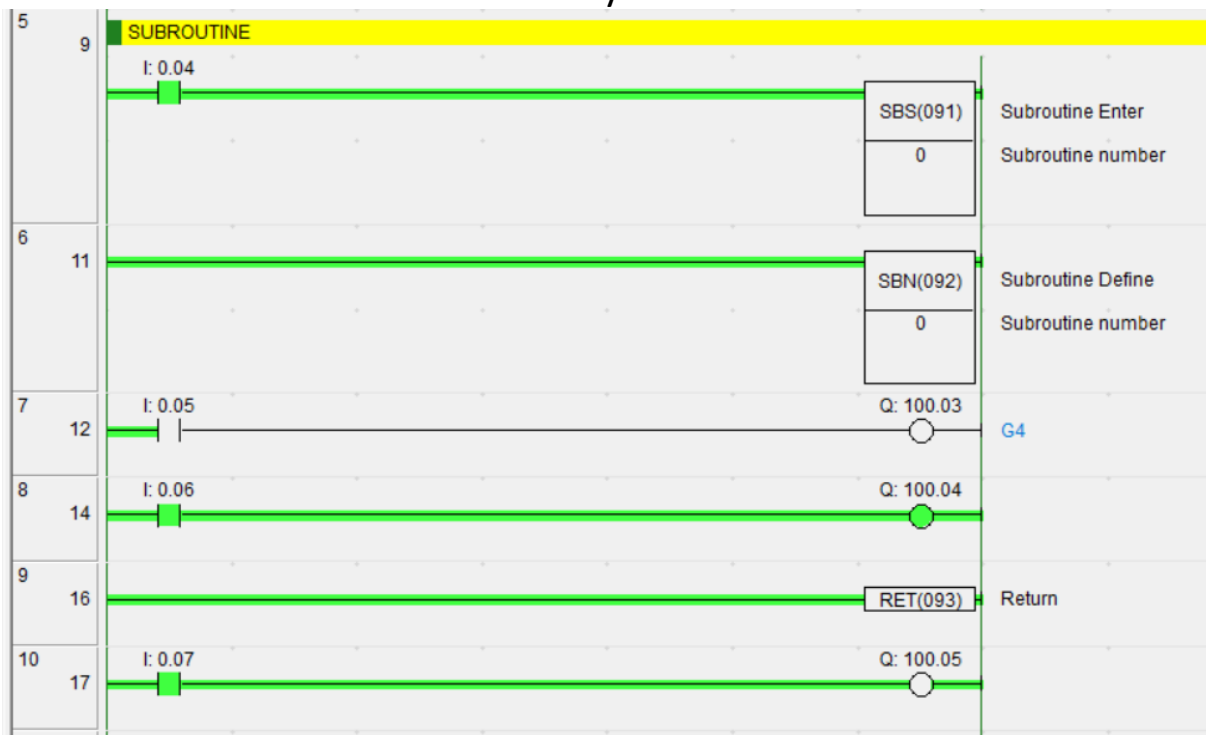
SBN 0123

SRET

➔ Whatever we wrote within the subroutine that only will be executed/runs.



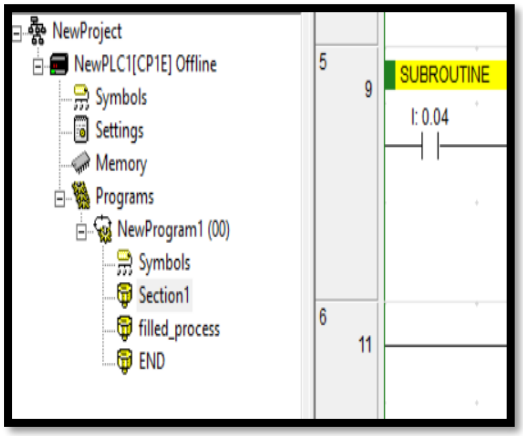
Executes within the subroutine only

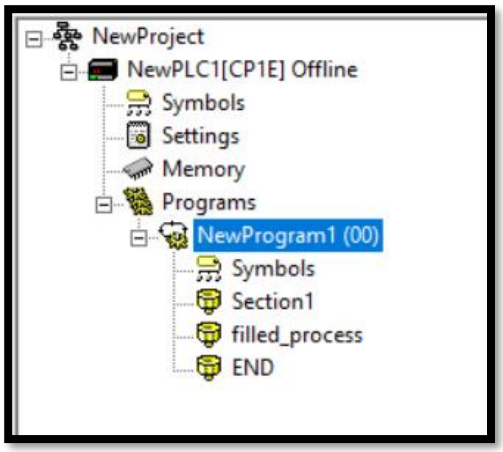


Here after the program will not run.

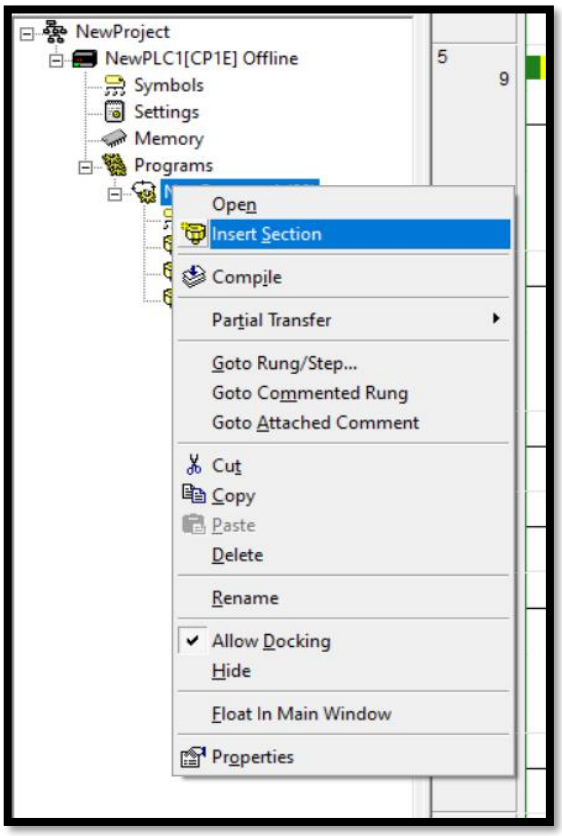
⇨ We need to take “New Section”

Goto **My Program** -> **add section**





right click on New Program



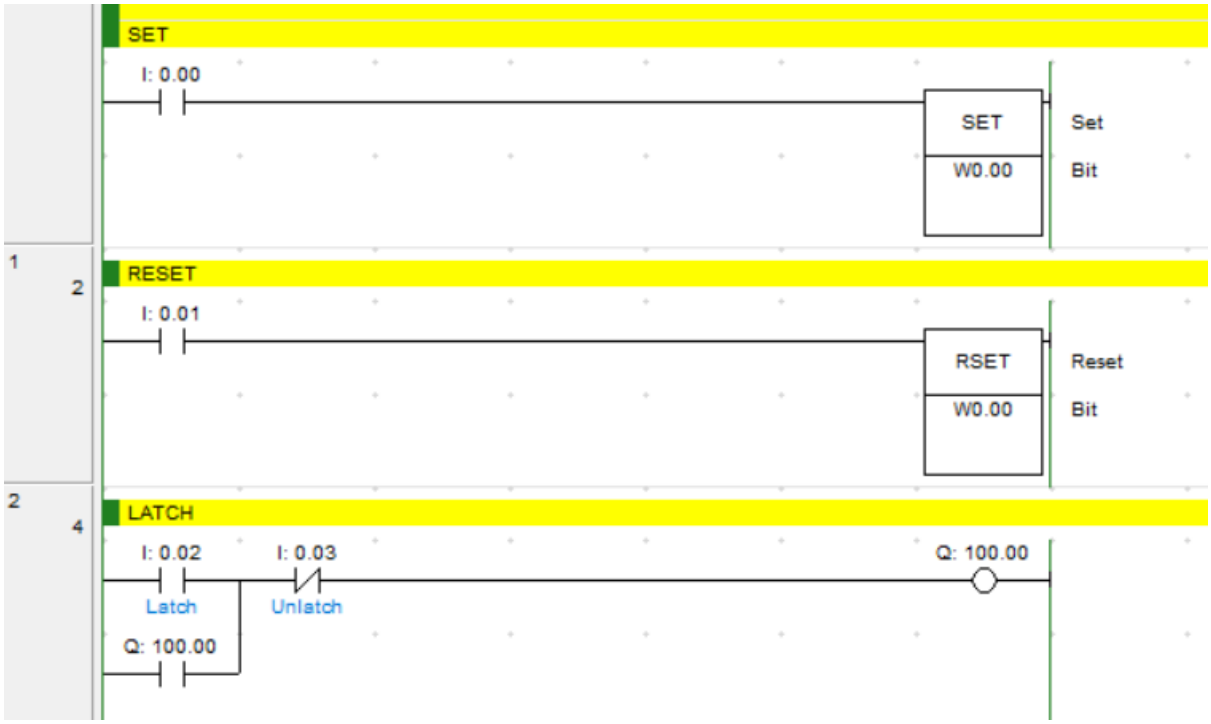
Click on Insert Section to continue further operation/projects/works.

SET & RESET

Set(LATCH) and Reset (UNLATCH)

Set Syntex: SET address(W0.00)

RESET syntax: RSET address(W0.00)



Initially all the NC & NO are 0's

I:0.02 → 0, I:0.03→ 0 , then Q:100.00 → 0

1	0	1
0	0	1(feedback)
0	1	0

Function Block Diagram

→ Graphical language for programmable logic controller.

→ This defines the function b/w i/p & o/p variables.

→ This involves C language

- Conditions
- Loops

IF:

Syntex:

if(condition) then

:= Statement;

end_if;

IF ELSE:

Syntex:

If(condition) then

:=Statement1;

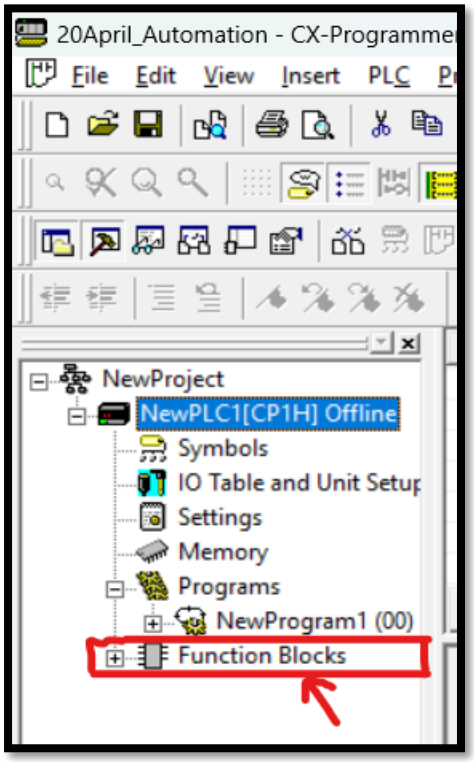
Else

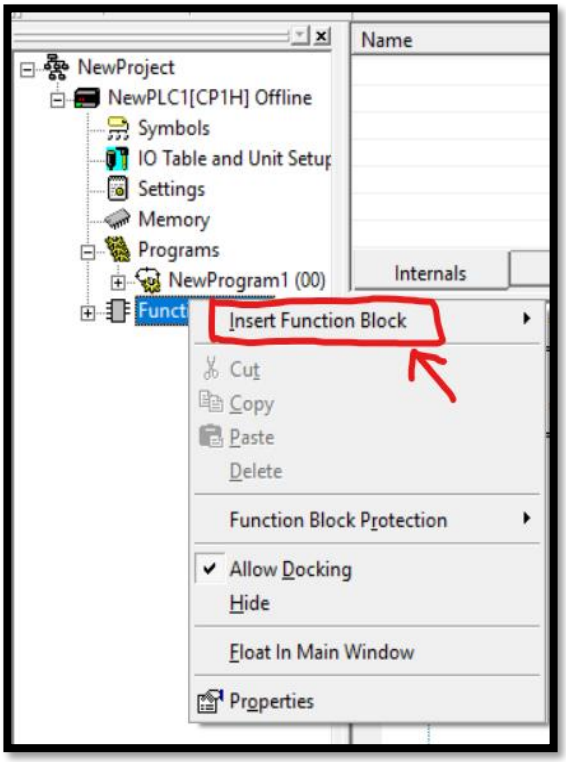
:=Statement2;

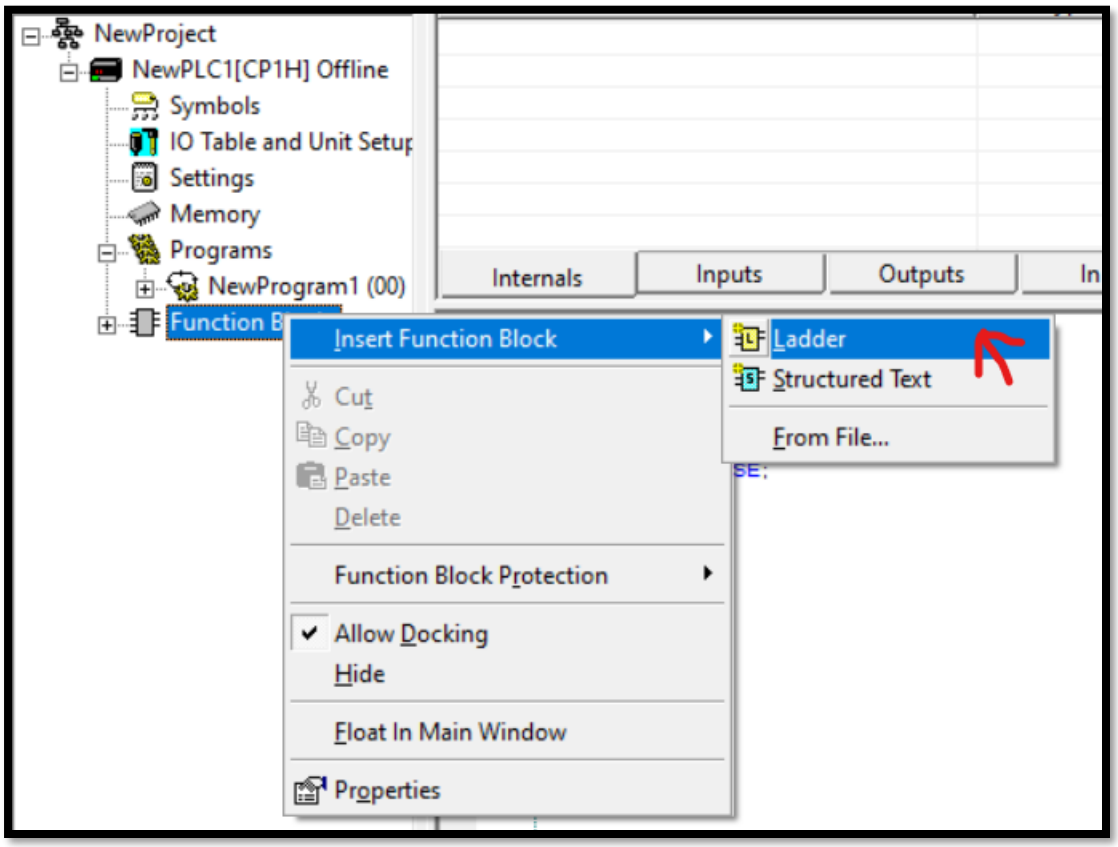
end_if;

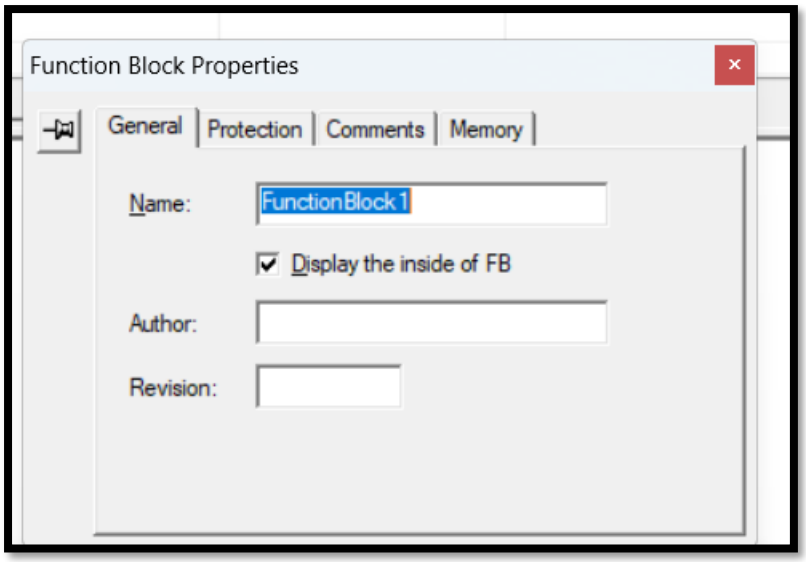
Creating a Function Block Diagram follws the below steps:

➔ When you open the “**CX-Programer**” → “**Create a new file**” → there you observe left side contents.There you observe the “**Function Blocks”**

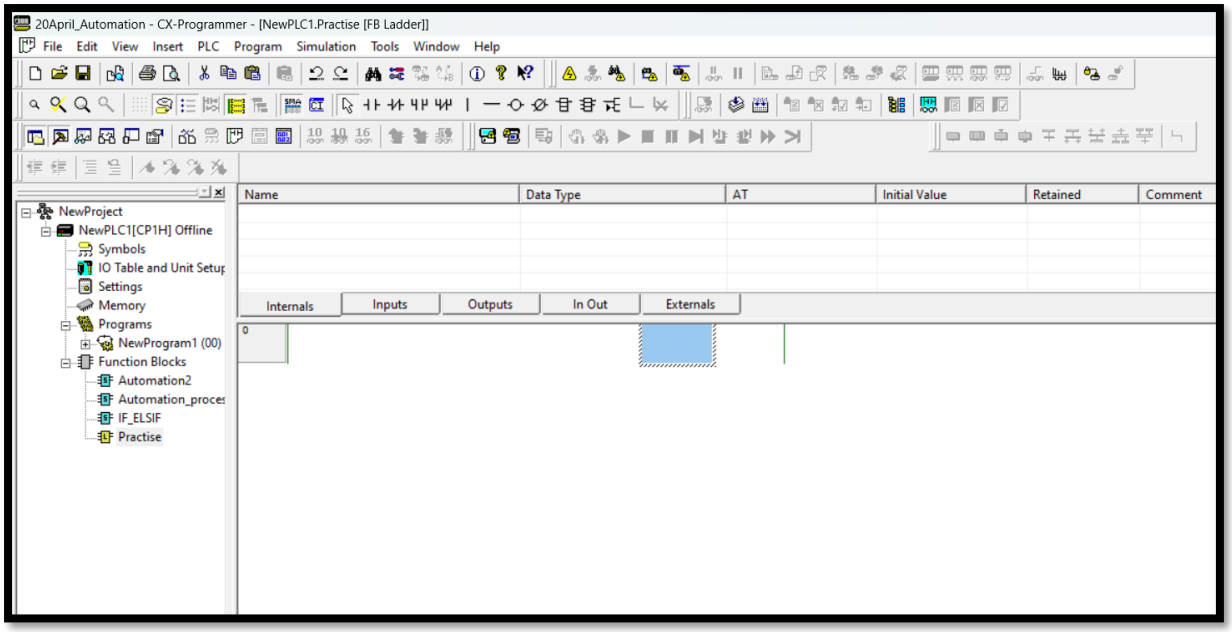


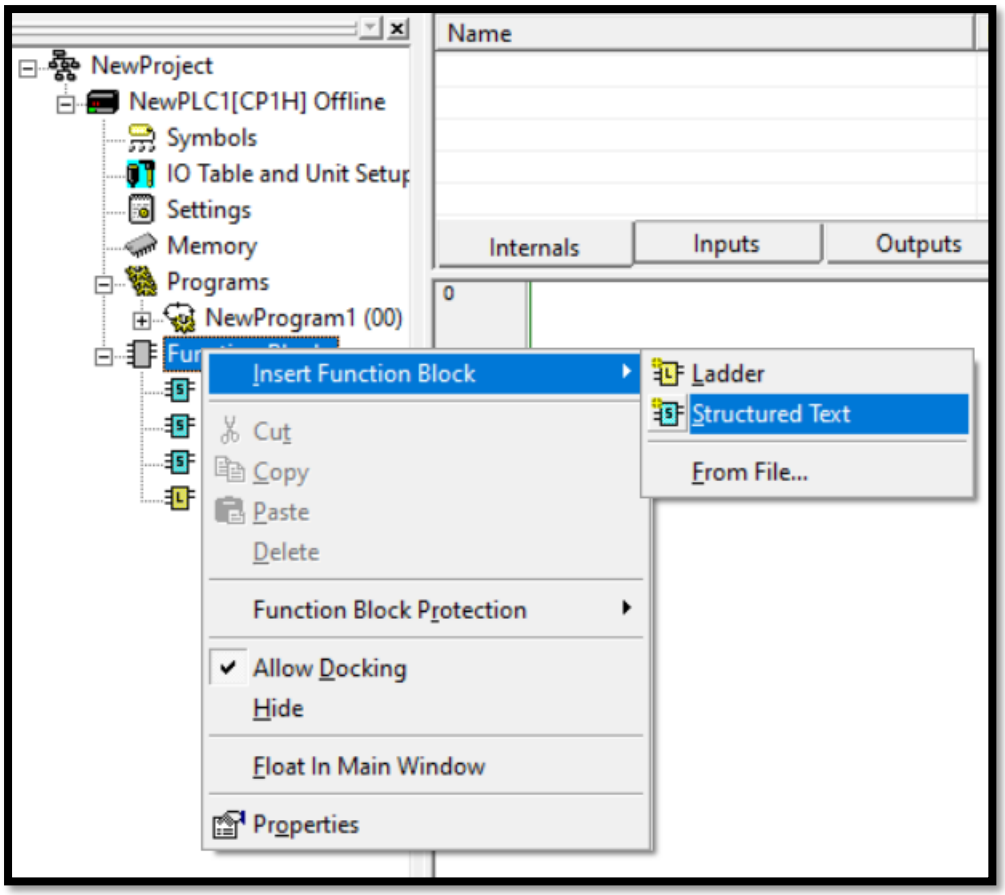


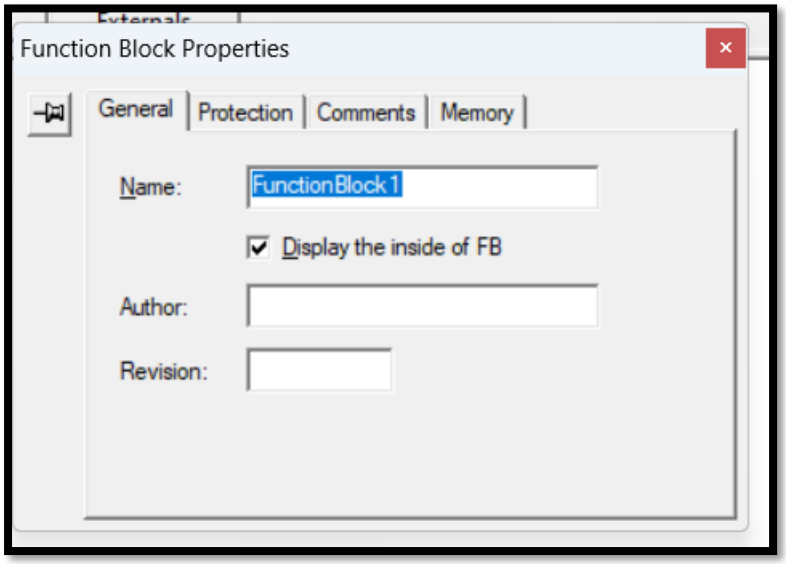


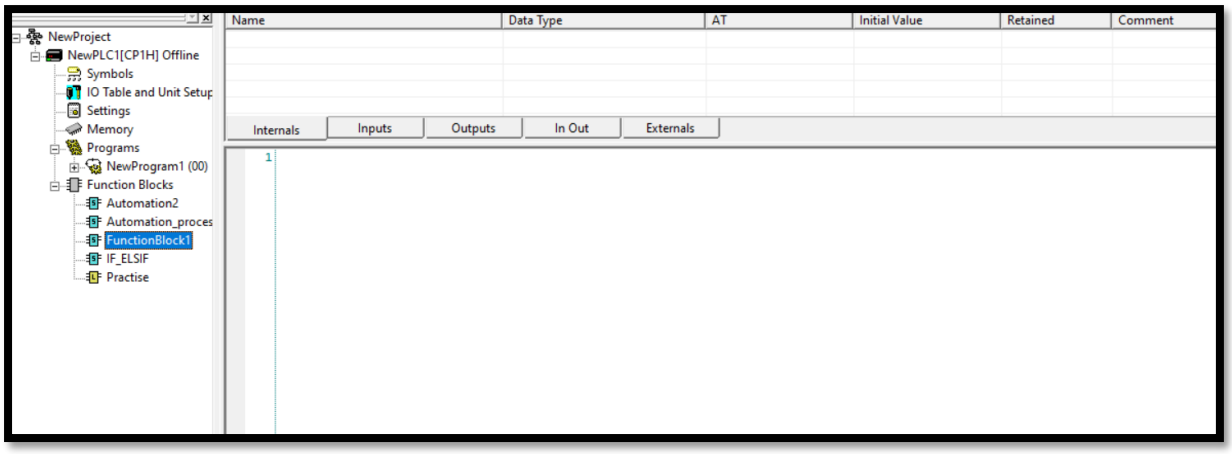

Function Block Properties
General
Protection
Comments
Memory
Name:
FunctionBlock1
Display the inside of FB
Author:
Revision:

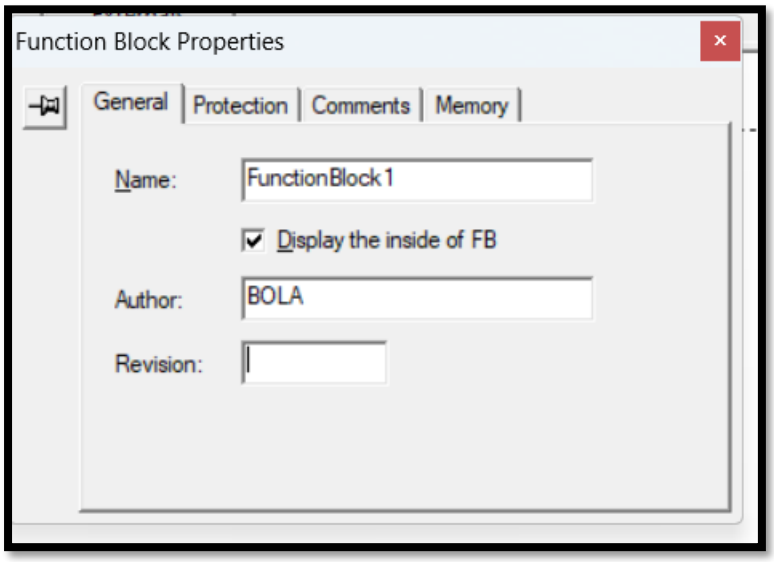
"**Practise**"


20April_Automation - CX-Programmer - [NewPLC1.Practise [FB Ladder]]
File Edit View Insert PLC Program Simulation Tools Window Help
NewProject
NewPLC1[CP1H] Offline
Symbols
IO Table and Unit Setup
Settings
Memory
Programs
NewProgram1 (00)
Function Blocks
Automation2
Automation_proces
IF_ELSIF
Practise
Name
Data Type
AT
Initial Value
Retained
Comment
Internals
Inputs
Outputs
In Out
Externals


NewProject
NewPLC1[CP1H] Offline
Symbols
IO Table and Unit Setup
Settings
Memory
Programs
NewProgram1 (00)
Name
Internals
Inputs
Outputs
0
Insert Function Block
Cut
Copy
Paste
Delete
Function Block Protection
Allow Docking
Hide
Float In Main Window
Properties
Ladder
Structured Text
From File...


Function Block Properties
General
Protection
Comments
Memory
Name:
FunctionBlock1
Display the inside of FB
Author:
Revision:





After this press “Enter”
Then it will open a “Structed text”
Here we can write the Coding part and then execution follow on.
→After that one function block is created “FunctionBlock1”.
→Double click on “FunctionBlock1”.
→One pop-up arrives.
→In CP1H Block diagram user defined timer and counter cannot be used.
i.e We cannot use pre-defined timer & counter in FunctionBLock.

DataTypes:

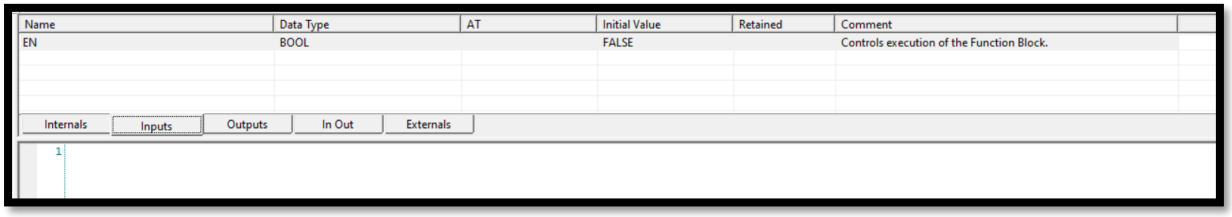
Boolean-> 0/1 True/flase On/Off
Integer-> 16bit memory /1 word
int -> -32768 to +32767
uint-> 0 to 65535
dint-> double integer(32bit memory)

dint-> -2147483648 to +2147483647
udint-> o to 42949667295

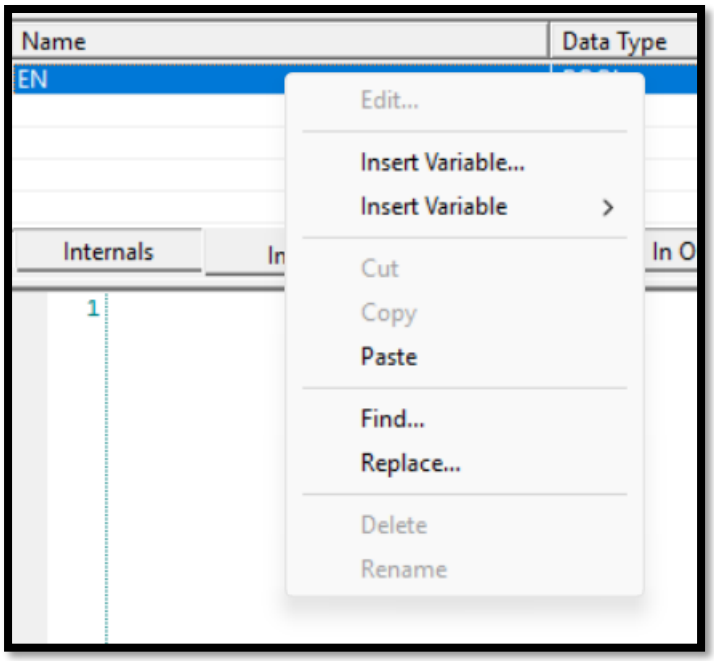
Now we will proceed to execution part of problems

Open "**CX-Programmer**" → click on Function block you created "**FunctionBlock1**" →

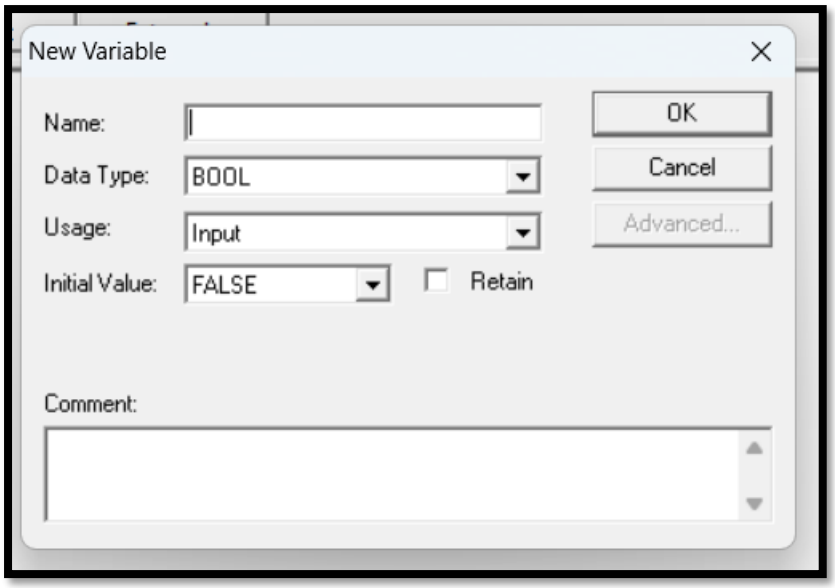
- ❖ goto "**Inputs**" → you'll see "**EN**" →



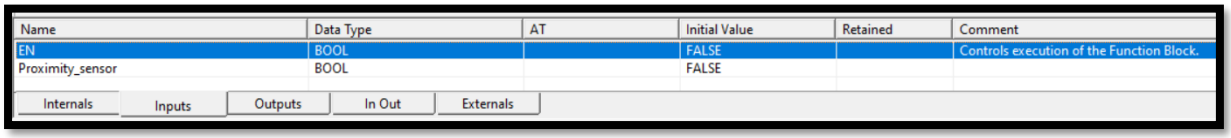
right click on it →



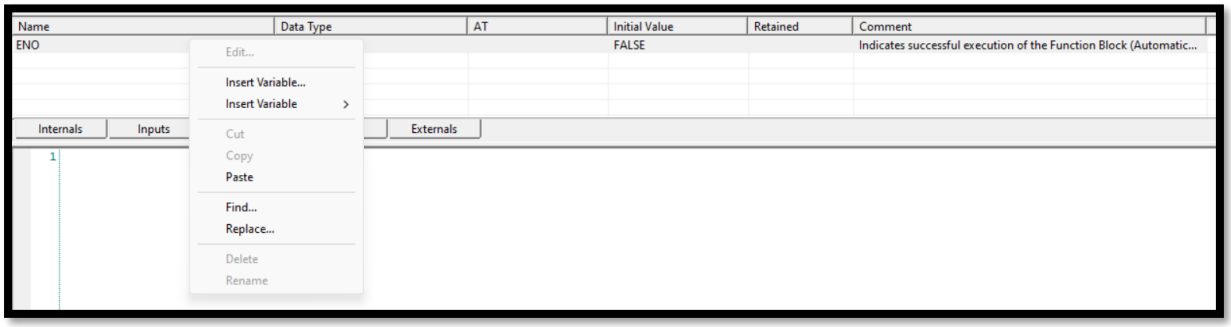
Select Insert Variable →then a pop-up arrives

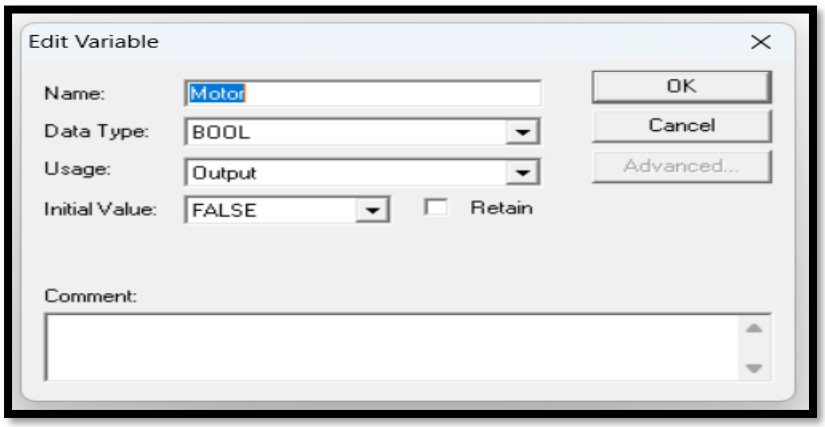


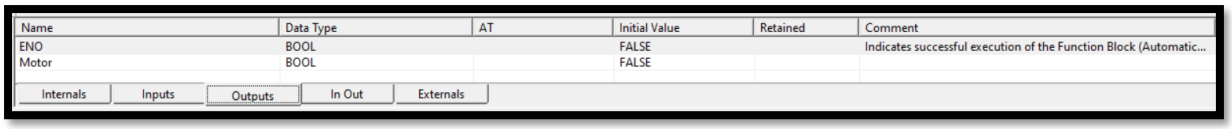
Give any name, datatype, usage, & initial values..
Retain [X] i.e do not tick that ..
Write any comment if required..
Then click ok.



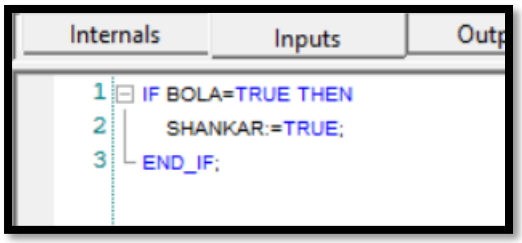
Similarly like this way define "**Output Variables**"
Goto "**Outputs**" → Right click on"**ENO**"→ insert variables → give name,datatypes,usage, & initial value → click ok



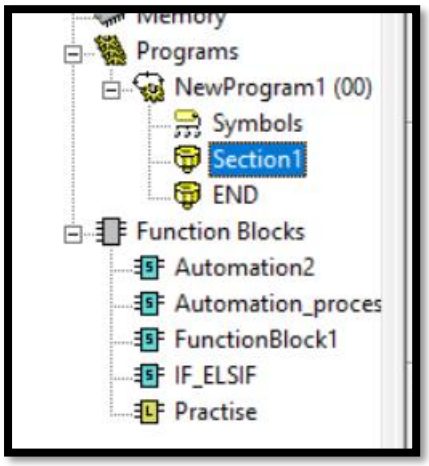




WRITE THE CODE



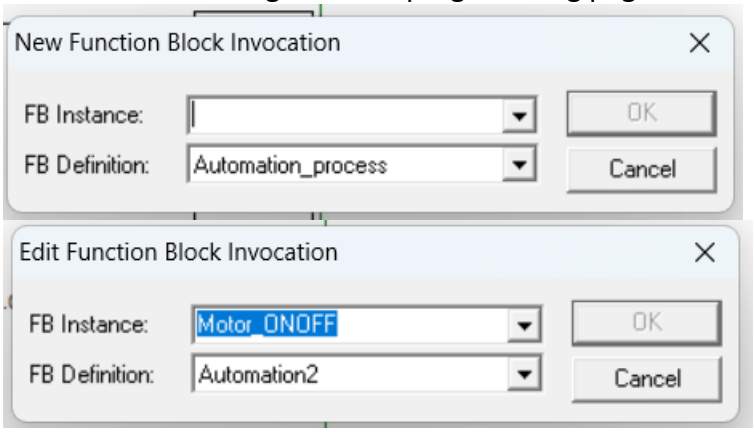
→Click on Section1 [i.e back to programming page]



At the top “ ” New function block invocation

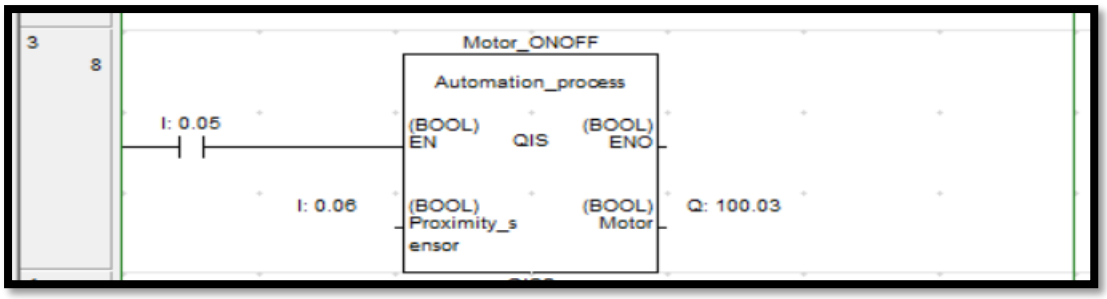
Note: If it appears when only you select “**CP1H**” before.

Click on and drag that into programming page. After that it will appear a pop-up



To get exit from the functionBlock click on “**Esc**”.

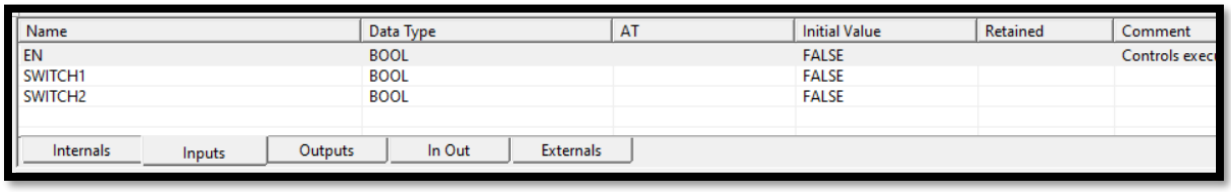
Click ok

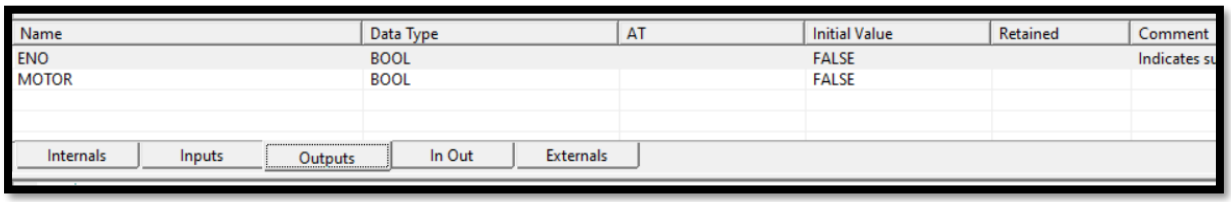


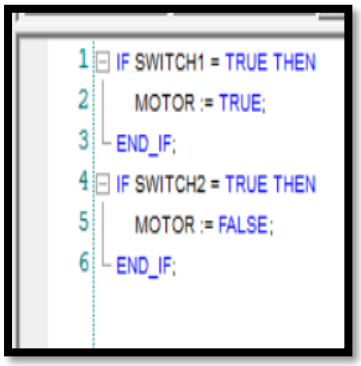
Complete the desigining and turn on "**Simulation**" (**Ctrl + Shift + W**)

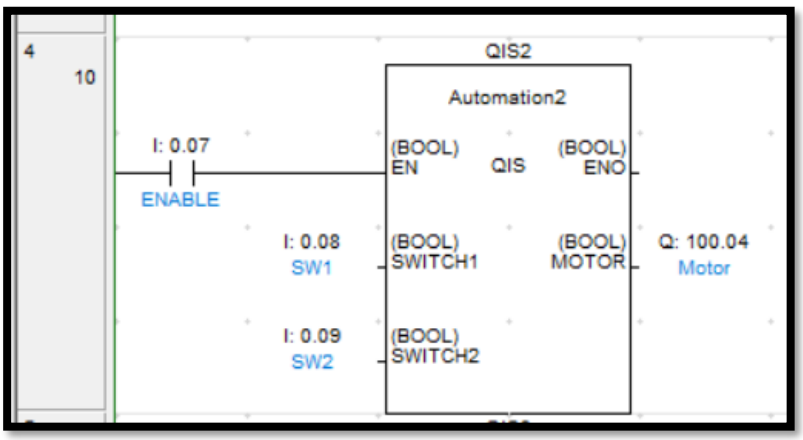
Process:
I: 0.05 →OFF & **Proximity_sensor** → OFF Then **Motor** → OFF
I: 0.05 →ON & **Proximity_sensor** → ON Then **Motor** → ON
I: 0.05 →ON & **Proximity_sensor** → ON Then **Motor** → ON

Similary we do with other conditional statements like IF ELSE ,IF_ELIF,etc









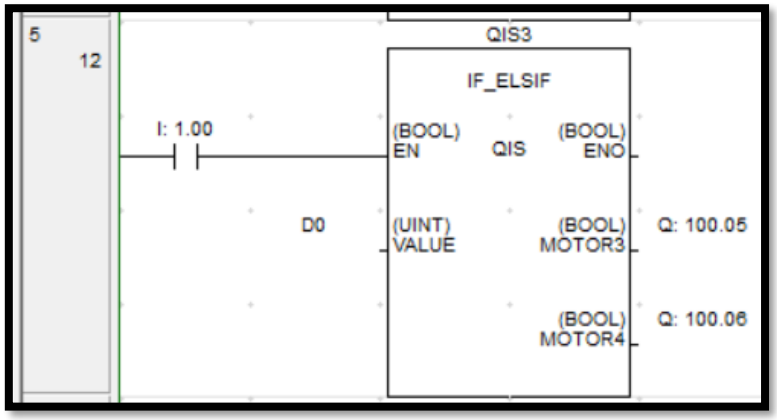
Name	Data Type	AT	Initial Value	Retained	Comment
EN	BOOL		FALSE		Controls execution of the Func
VALUE	UINT		0		

Internals | Inputs | Outputs | In Out | Externals

Name	Data Type	AT	Initial Value	Retained	Comment
ENO	BOOL		FALSE		Indicates successful execution of th
MOTOR3	BOOL		FALSE		
MOTOR4	BOOL		FALSE		

Internals | Inputs | Outputs | In Out | Externals

```
1  IF VALUE = 1 THEN
2      MOTOR3 := TRUE;
3      MOTOR4 := FALSE;
4  ELSIF VALUE = 2 THEN
5      MOTOR3 := FALSE;
6      MOTOR4 := TRUE;
7  ELSE
8      MOTOR3 := FALSE;
9      MOTOR4 := FALSE;
10 END_IF;
```



DATA REGISTER

Data register address: D0,D1,D2……..D3999

→Single data register carry 16bit of data memory.

MOVE OPERATION

SYNTEX: MOV S D where S-> Source, D->Destination

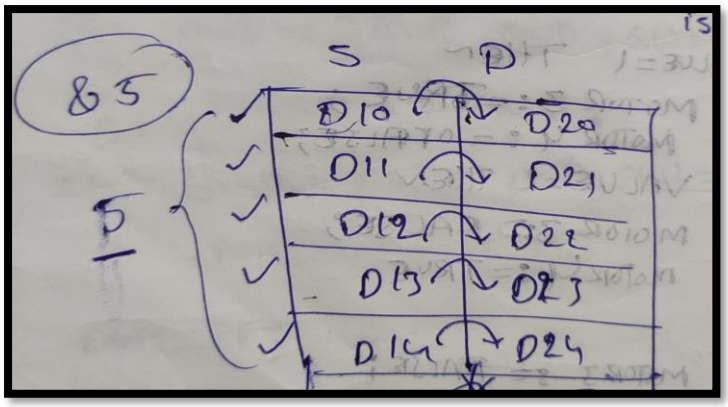
XFER &5 S D &5 defines no.of data is to be moved

MOVE OPERATION

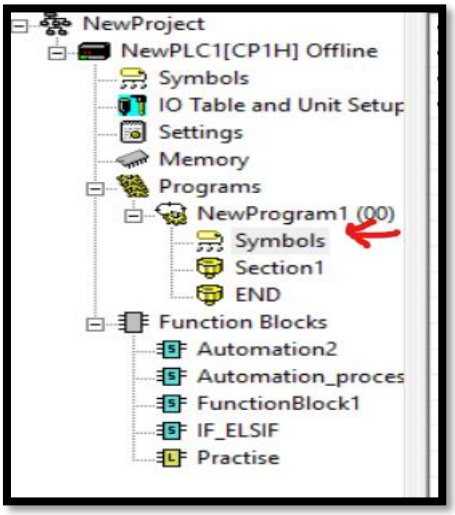
Syntex: Mov S D where S→ Source

D→ Destination

XFER &5 S D &5 represents/defines no.of data is to be moved.

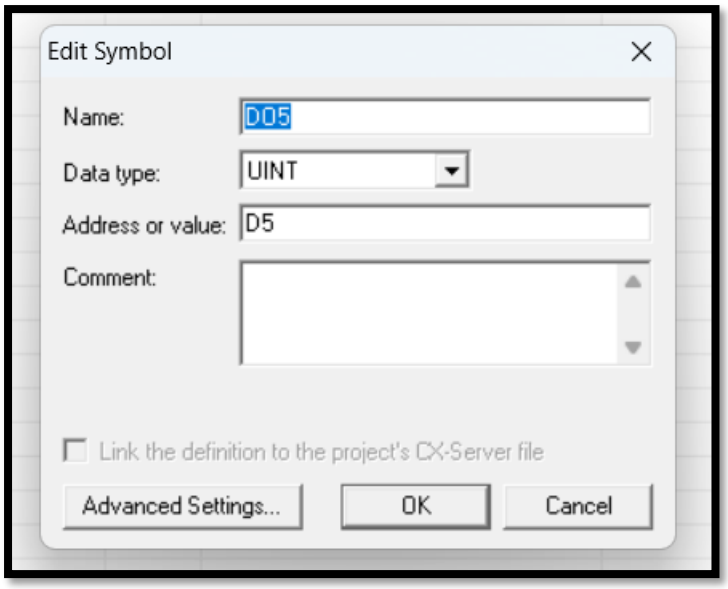


Note: Intially in Mov operation we get Hexadecimal Value.
For getting decimal value we go to symbol



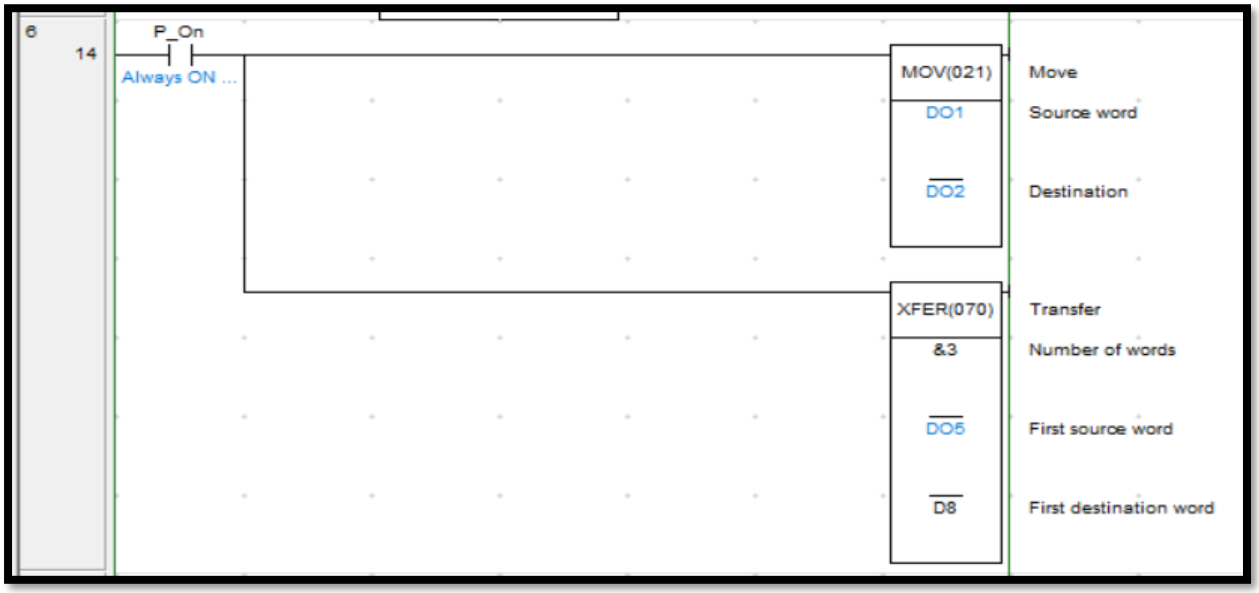
Name	Data Type	Address / Value	Rack Locati...	Usage	Comment
DO1	UINT	D1		Work	
DO2	UINT	D2		Work	
DO5	UINT	D5		Work	
DO6	UINT	D6		Work	

You can change datatype of the addresses DO1,DO1,DO5,DO6 to other as required.



Click ok

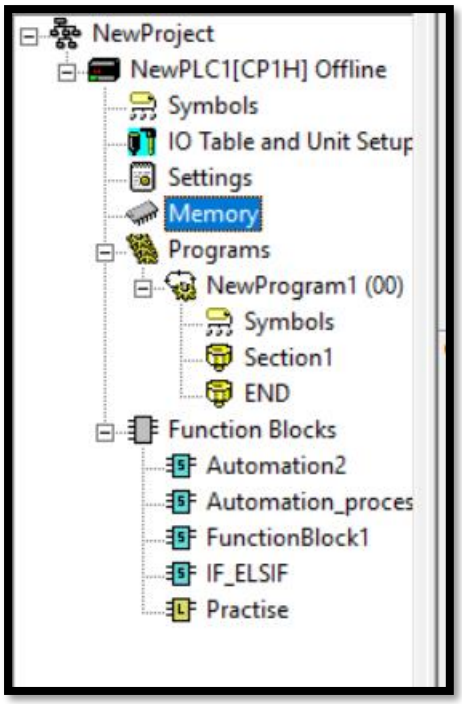
- P_On [LD P_On]

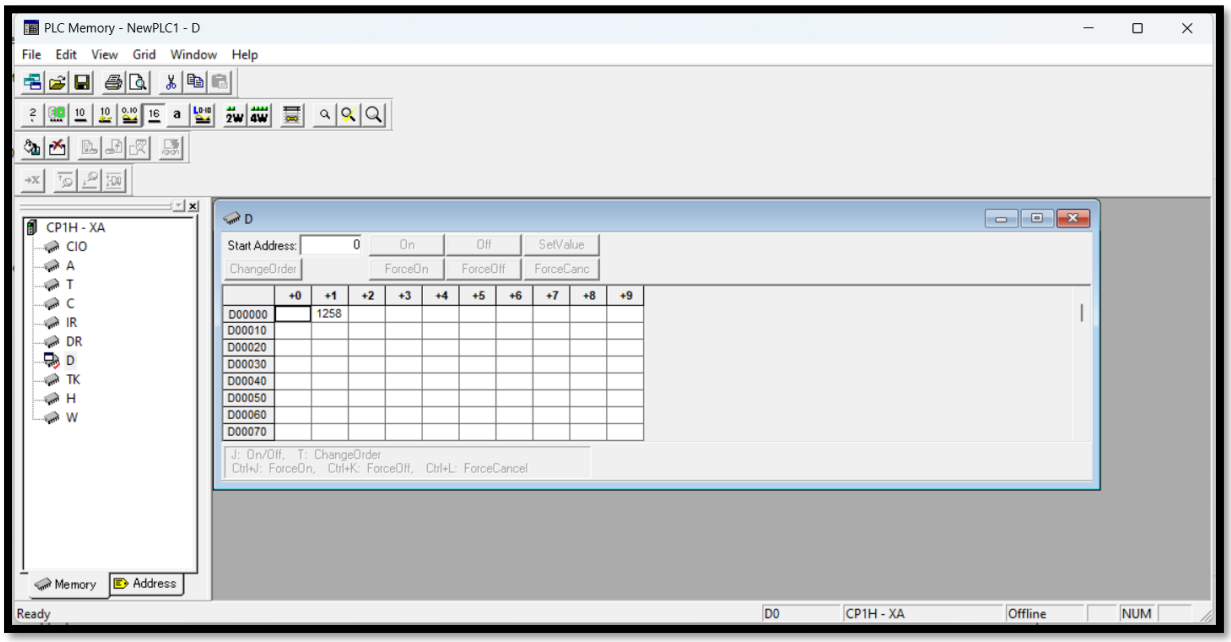


For performing mov data operation
→It should be in simulation online mode [Ctrl + Shift + W]

Note: For entering data values we go to "**memory**".

Memory(double click on it) → D(double click on it) → Pop-up arrives → Online (top_Middle) → Monitor → D [Monitor] → Click on Monitor





Enter your values here

Change D1→ D2
D5→D8
D6→D9
D7→D10

Saved the data

ARITHMETIC OPERATION:(INTEGER)

ADD: + D20 D21 D22 //D20+D21=D22

SUB: - D24 D25 D26 //D24-D25=D26

→It can store 16 bit memory

Note: The result value should not exceed the integer Range values.

MUL: * D27 D28 D29 //D27*D28=D29

DIV: / D30 D31 D32 // D30/D31=D32

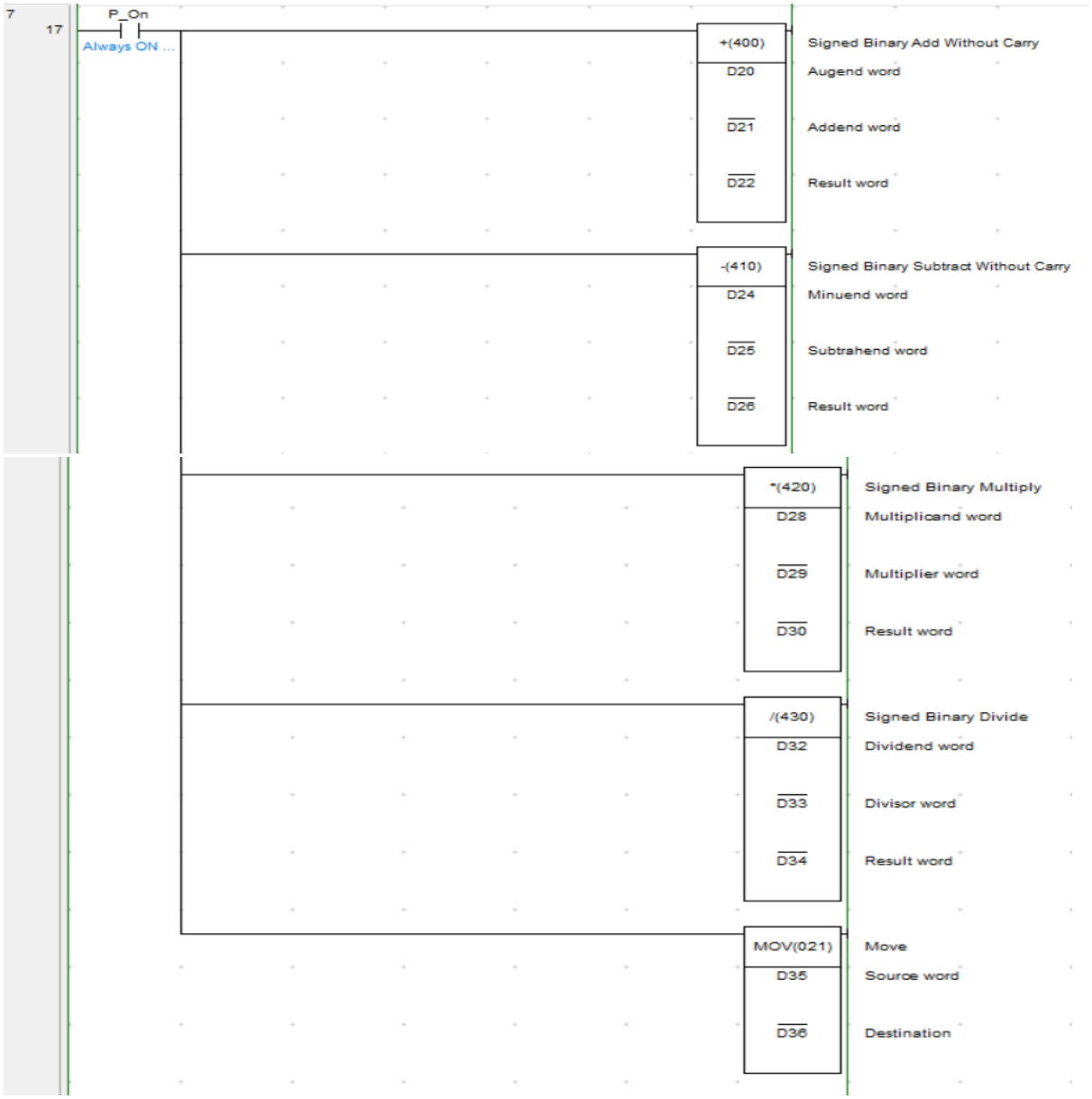
Where D32 stores quotient

(next successive address of OUTPUT) →D33 stores Remainder

@MODULO DIVISION is not possible in PLC

→The Next Successive address stores the remainder value

i.e here D33 stores remainder value



INTEGER TO FLOAT CONVERSION

FLT 16bit integer to 32 bit float conversion

Float arithmetic operation

Addition : +F

Subtraction : -F

Multiplication : *F

Division : /F

FLOAT to INTEGER Conversion (Analog type)

FIX 32bit Float to 16bit Integer conversion.

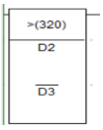
In some other PLC's : INT

COMPARISION:

In industry,we use "analog level meter" to measure the water level of the tank.

<, >, <=, >=, =

- **"> D2 D3"** →



COMMUNICATION PROTOCALS

RS232 → Recomended Standard

Communication → Serial & Parallel

- ❖ Full duplex & Half duplex modes.

Half duplex → we can either send or receive data .but not

(HD) At a time will not happen

Ex: Walky-Talky

Full duplex → At a time both operation like sending

(FD) Receiving will take place.

Ex: Cell-Phone,etc.

RS485(HD,FD),RS422(HD,FD),

Modbus,profibus,profinet,hostlink

→uploading and downloading take place with these communication Protocols.

Thank you Manjeet Sir

Manjeet Kumar sir || Hari sir

Done by :

BOLA SHANKAR VELIDI

Roll No. **: 20491A0402**

QIS College of Engineering & Technology