# Advanced Industrial Automation

PLC programming in simplest way with 110 solved examples

Himanshu Kumar

Dedicated to my father '**Shri Raj Kishore Singh**'

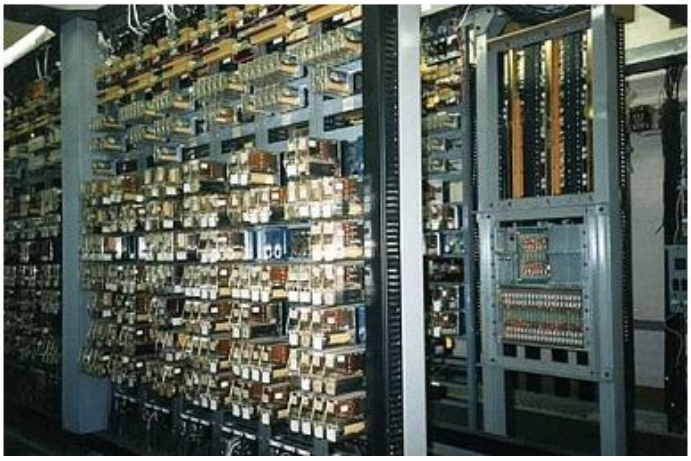
**You have always inspired me to keep moving and not to accept defeat.**

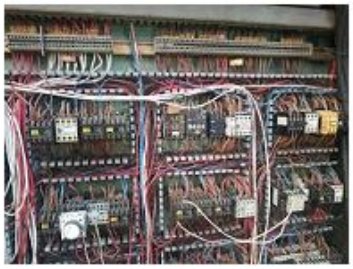
Being an engineer or technician our major role in industry is to maintain the system running without fail. If any machine gets shutdown, we need to restart it in short time. In today's industries where production rate and product demand are very high, machine should not get shutdown. If it is shutdown then it should be restarted very fast. Now our fist problem is that from where to start finding the problem in the machine which is very large in size and spread in large area. Actually, we have not to be worried about the largeness of the machine as largeness is just a mechanical structure. Often, we don't get any problem in structure once machine running trail is checked OK. So, we have not to be worried about the largeness of machine but we have to concentrate on three parts i.e.

| **INPUT** | **CONTROLLER** | **OUTPUT** |
|---|---|---|

These three parts of machine are the important parts in terms of finding problem. Most of time, machines are shutdown because of its input/ output issues. We find issues in controllers rarely. Now we go to find problem among inputs and we face another issue i.e., large number of inputs. It can be 100 or 1000. You can imagine how difficult is it to find problem among such large number of inputs and it will take time also to do shut troubleshooting. The same problem we face among outputs. If we have advanced controller with advance featured software, we can find problem in 5 to 10 minutes.

We shall have a discussion about controller. A controller monitors and controls any machine system. Initially when industry started it was all manual operation. Manual means a **man was the controller**.it was a man/ operator who used to monitor the system and he/ she was taking decision of operation too. This manual system was not accurate and was so time taking system. Very first automatic controller came in industry was **relay logic controller**.





It was a large complicated electrical electronics circuit full of many small and big devices. The logic was designed through wires with the help of electrical and electronic devices such as relay, contactor, hardware timer/ counter, diode resistor, capacitor etc. it was complicated but automatic system. This relay logic can be seen in some old industries. Railways also use relay logic control. Now it is being replaced with advanced controller. It was a good controller but obviously it has a lot some limitations. Once it was shutdown, it was so difficult to find problem in that large complicated circuit as it was no any monitoring system that time. Another issue was, once the circuit was ready for one application, it was very difficult to modify that circuit for another application. In fact, small modification was also difficult. You can see one example of relay logic control in college machine lab. The motor is on and off by two push buttons. Green and red coloured generally. Push button has spring system inside it. When pushbutton is pressed, it conducts and supply reaches to motor. When push button is released, due to spring, push button gets off and signal does not pass through it but motor still runs. How? There is no any microcontroller there you can see. It is continuously on because of relay logic. Motor is latched by a relay.

Another very famous controller came was **micro controller**.


This is a very famous controller especially in academics. It is a very good controller as it is very compact in size, very cheap in cost and the best part programmable. If you are able to program your controller, you can make modification in your existing system as per requirement. But still it is not used in industries to control machines and this is the reality. Micro controller is a very good controller but it has some limitations because of which it is not the main controller in manufacturing industries.

The main controller to control machines in industry is **PLC (Programmable Logic Controller).** It was originated in 1968 in automotive industry in USA on the proposal request of GM Hydramatic (General Motors) for replacement of relay logic control system. This proposal was written by Er. Edward R Clark. The selected proposal came from Bedford from Bedford Associates. First PLC '084' was built in 1969. It was their 84$^{th}$ project so named it 084. Bedford started a company dedicated to developing, manufacturing, selling and servicing the new product MoDiCon (Modular Digital Controller). Dick Morley, one of the people who worked on the project is considered father of PLC. This brand MODICON was sold to GOULD Electronics in 1977 and later to Schneider Electric, the current owner. The first model 084 is still with Schneider which was presented by GM when the unit was retired after nearly 20 years of uninterrupted service. Modicon used '84' moniker at the end of its product range until the 984 made its appearance. In a parallel development ODO Josef Struger is sometimes known as father of PLC. He involved in the invention of Allen Bradley PLC during 1958 to 1960. He invented the PLC acronym. Allen Bradley is now own by Rockwell Automation. Struger played a leadership role in developing IEC61131-3 PLC programming language standards.

In general PLC is an electronic digital device which takes input and generates output as per logic developed. It is a digital device. Only digital input/ outputs are directly connected to its I/O terminals. Analog I/Os are not directly connected to PLC directly. ADC (analog to digital converter) and DAC (digital to analog converter) are used to connect analog I/Os with PLC. Some PLCs provide few analog I/O terminals on its main unit. In that case ADC/ DAC are placed inside main module. In industry generally ADC/ DAC is used separately because the number of analog I/Os will be more. PLC is used to control machines in any kind of manufacturing industry such as Automobile industries, Cement plants, Power plants, Oil & Gas plants, Food processing industries, Chemical plants, Pharmaceutical industries, printing and packaging industries etc.

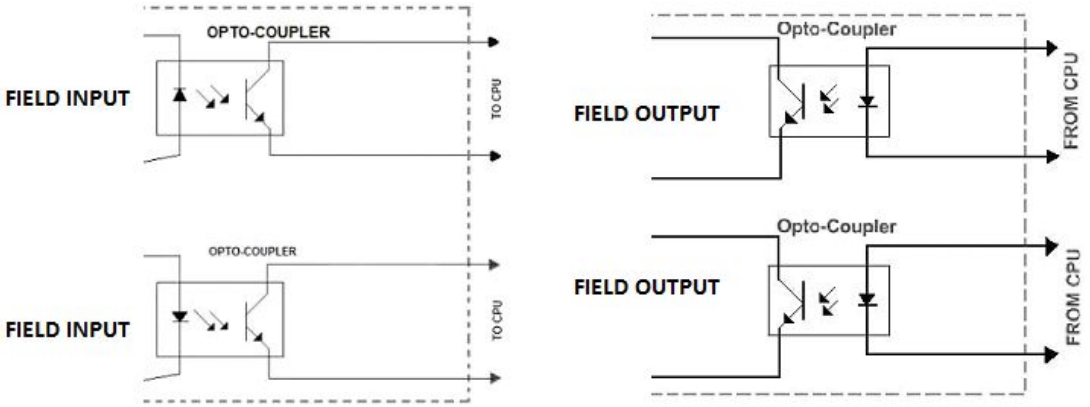
Before we discuss the limitation of microcontroller, let me clarify that advanced controllers also work on the working principle of microcontroller. Advanced features of advanced controller were not added to microcontroller but a new controller was introduced to sell at higher cost. Let us understand the limitation of microcontroller because of which it is not used in today's industries.

| Microcontroller | PLC |
|---|---|
| **Programming languages:** Its programming language is based on C language. It is not that easy language to be learned by all. | It has 6 programming languages. User can select language as per comfort. Ladder diagram is the easiest language. It can be understood very easily by maximum learner. Ladder diagram is preferred by industries because troubleshooting is very easy in it. |
| **Download/ Upload timing:** It takes more time to send program to controller. | It takes fraction of second or a few seconds to send program to controller. |
| **Program receive option:** It has no program receive option. If you don't have program backup file then you cannot recover program from controller. You will have to rewrite that logic again. Without receiving existing program of controller, troubleshooting is also very difficult as we won't be able to understand the machine operation logic. | Existing program of PLC can be received very easily even in running condition of machine. If your program backup file is lost from your computer then you can receive program from PLC. |
| **Monitor/ online mode:** Its software has no good monitor mode to check running status of machine. Its monitor mode is not that much helpful in troubleshooting machine. | It has good monitor mode in its software which helps a lot in troubleshooting machines. If any problem occurs in machine, we follow few steps to reach to the exact issue. Take laptop/ computer which have that PLC software installed in. connect PLC communication |

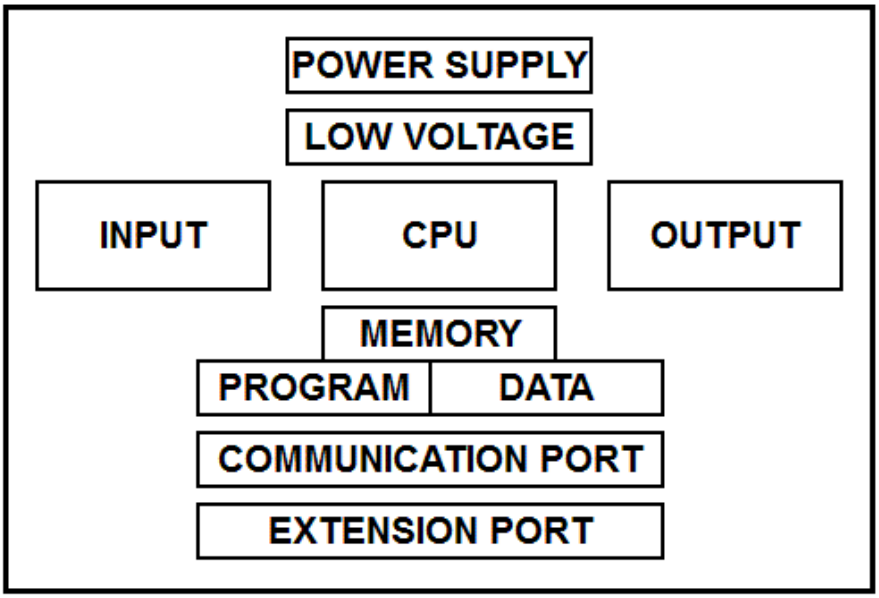
| | |
|---|---|
| | cable's one end to PLC and another to laptop/ computer. Open the software. Set communication and click receive/ upload/ read/ PLC to PC option given in that software. The existing program will be received in that PLC software. Now go to monitor/ online mode. You will be able to monitor the running/ present status of addresses used in that program. You can search your output which is not working as per sequence. You can easily monitor that why the main supply does not reach to that output. You will easily reach to problem within minutes. Without program receive and monitor mode it is very difficult to troubleshoot any complicated system. |
| **Online edit:**<br>Online option is latest and the best option in automation software Many times it is not possible to stop machines to avoid production lose but some modifications are required for efficiency. If you modify existing program then you need to rewrite it to PLC and it will stop machine. Microcontroller has no online edit option. | Now PLC software has option to rewrite modified ladder without stopping machine using online edit feature of software. |
| **Handling:**<br>It is not that much robust. It has less I/Os and registers available. It has not that much life compared to PLC. | It is manufactured to work in extreme conditions in industry. It has large number of I/Os and registers handling capacity. The first PLC '084' was retired after 20 years of uninterrupted service |

| | |
|---|---|
| | that too because that unit was retired. PLC can work long life. |
| **Usages:**<br>Generally, it is used for small applications or mostly in-home appliances where it is replaced with new one when out of order. | It is used to control small to big machines in industries with a large number of logic sequence. |

PLC input output terminals are not directly connected to PLC CPU. Signal is sent through light from terminal to CPU and from CPU to terminals through Opt couplers that is why PLC CPU last long. If nay wrong connection is made from field device then light is not going to harm CPU.




Let it be any brand or any type of PLC it has some essential components/ parts in it. Let us understand it using block diagram.




PLC BLOCK DIAGRAM

**Power supply:** To work on any electronic device we need to power it on. PLC provides terminals to connect power cables to it. It is manufacturer decided that you have to provide AC or DC power to PLC. If it is AC power supply then directly L, N and earth terminals of PLC will be connected to the socket. If it is DC power supply needed then it will be 24 V DC. An SMPS (Switch Mode Power Supply) will be the power source and the PLC terminals + - will be connected to it.

**Low voltage:** This low voltage is for PLC's internal use and we do not provide it from outside but it is converted by PLC itself from its power supply. This low voltage is 5 or 3.3 V DC in old and new PLC respectively. As PLC is a digital device, its internal work takes place on 0/ 1. Old PLC had 5 V DC as 1 and new PLC has 3.3 V DC as 1. PLC also converts power supply into 24 V DC as low voltage for its I/O wiring purpose.

**CPU:** Central Processing Unit is the brain of PLC. It is a microprocessor which monitors and controls the logic and communication execution. Processor name is different in different brand of PLC. **Z80, TSX, Dallas 8051, Powerpc, 6803** etc. are few processors used in different brand.

**I/ O:** input/ output in block diagram refer to input output terminals not the physical I/ Os like push button, sensors, motor, relay etc. initially I/ Os we had of three types named relay, transistor and triac but now we have only two types Relay and Transistor. If you have a relay type PLC then you will have all its I/ O terminal relay (initial two terminal can be transistor type in some brands for high-speed signal use). If you have a transistor type PLC

then all its I/ Os will be transistor. As per our project requirement we select Relay or Transistor type PLC. When we have high speed/ high frequency I/ O signal then we must select transistor type PLC because relay cannot read or generate high frequency signal. Let us see which type of application can have high speed/ frequency signal. In many applications we need to take number of motor rotation feedback. For this proximity or photo sensor can be arranged. It will give on signal per rotation but it will be very high speed/ frequency on of signal which cannot be read by a relay. The main device to take rotation feedback of motor is encoder. For each rotation it generates number of pulses. For example, it can give 36000 pulses per rotation. It depends on manufacturer. By dividing it by 360 we calculate number of pulses per degree of rotation if needed. 36000 pulses in one rotation is very high speed signal and it can be read by transistor terminal. Now let us see application where we need to generate high speed output signal. Whenever we use stepper or servo motor in in our project, we need transistor type PLC as number of pulses is required to run a stepper or a servo motor.  A stepper motor can require more than hundred pulses and a servo motor can require more than lakh pulses for one rotation. Large number of pulses is required for high precision work. More number of pulses, more angular division will be possible.

**Memory location:** PLC has two memory locations, program and data memory location. The memory location where program is saved is program memory. Memory location where all addresses (addresses that we use in our program) are pre located is data memory. It is EEPROM. PLC RAM and EEPROM is battery backed.

**Communication port:** It is used to communicate PLC with other devices such as laptop/ computer, HMI/ SCADA PC/ Drives etc. different communication ports can be found on different PLC. On branded PLC you can find RS 232, RS 422, RS 485 etc. you can have more than 1 communication port on one PLC but its cost will increase as number of communication port increases.

Ethernet

RS 232


RS 485

| MiniDIN 8 | Pin | – | Pin | MiniDIN 8 |
|---|---|---|---|---|
| HSo | 1 | – | 2 | HSi |
| HSi | 2 | – | 1 | HSo |
| TxD- | 3 | – | 5 | RxD- |
| GND | 4 | – | 4 | GND |
| RxD- | 5 | – | 3 | TxD- |
| TxD+ | 6 | – | 8 | RxD+ |
| | 7 | – | 7 | |
| RxD+ | 8 | – | 6 | TxD+ |
| shield | | – | | shield |

RS 422

RS 485

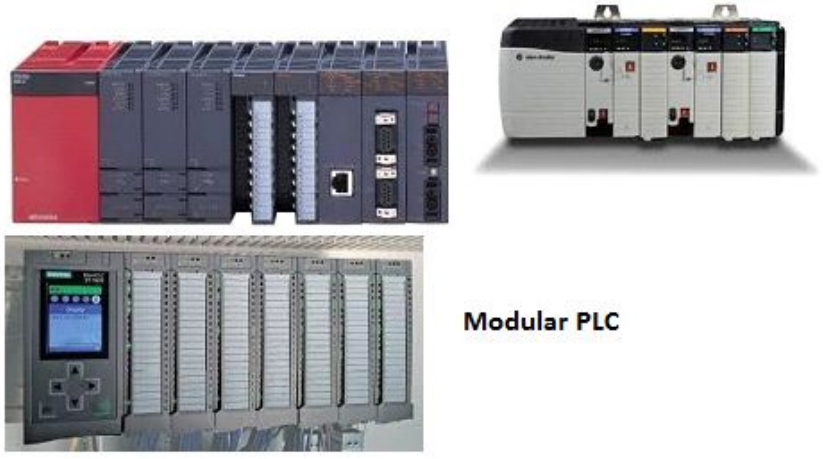
**Extension port:** on maximum of PLC, you will find extension port to extend its physical capacity. One small PLC CPU can control 100s of I/Os but on main unit of PLC you will find limited I/O terminals. If you want more digital I/O terminals then you can purchase I/O extension unit and you can connect it to PLC main unit through that extension port. Special purpose units such as ADC, DAC, high speed I/O module, communication module etc. are also connected through extension port of PLC.

## Type of PLC

As per its I/O terminal it is of 2 types i.e., Relay type and Transistor type. As per its size it was initially of three types' micro, mini and modular PLC. Micro PLC had I/O capacity 64, Mini had 256 and Modular had 4096. Now we have only two types micro and modular. Micro has 256 maximum I/Os capacity and modular has more than 5000 I/Os capacity. Exact number of I/Os capacity varies brand to brand.  You can recognize a micro or a modular PLC by seeing it. When all essential parts of block diagram are assembled in single unit is **micro-PLC.**

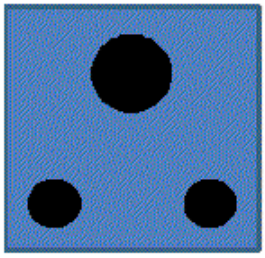
**Micro PLC**

We cannot remove any part. **Modular PLC** is assembled and can be dissembled easily. Almost each part of block diagram is as an individual module. Modular PLC is also called rack type and base type PLC. Initially it was manufactured very large in size and was arranged on rack so it is called rack type PLC. It has a base board with some connectors and slots. Modules/ cards are inserted to the slots. First slot is always reserved for power module and second slot is reserved for CPU. After that any other module such as digital I/O, analog I/O etc. can be inserted. Power module has power terminal and one indicator on it. CPU has few indicators, hard key for start/ stop, communication and memory on it. Digital and analog modules have terminals on it.



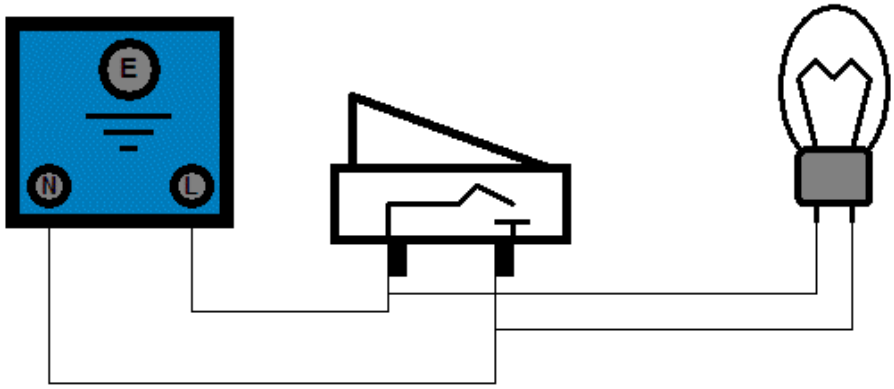
**Modular PLC**

Let us move towards programming of PLC. Programming is the most important and interesting part. Without learning programming, we won't be able to develop a new project, we won't be able to modify the existing program and we won't be able to troubleshoot machine easily. People get a bit afraid of hearing programming. Programming is always considered very difficult but PLC programming is not that difficult if you follow some steps/ tricks. If you learn programming in proper way then programming is so interesting. We are going to learn Ladder Diagram (LD) language. It is very similar to basic electrical diagram, Very basic diagram. LD is symbolic representation language you can say. Our target in this language is to send supply to output as per required sequence/ logic. If you understand basic electrical diagram, you will be able to understand ladder diagram easily.  Our all-electrical work starts from power socket.

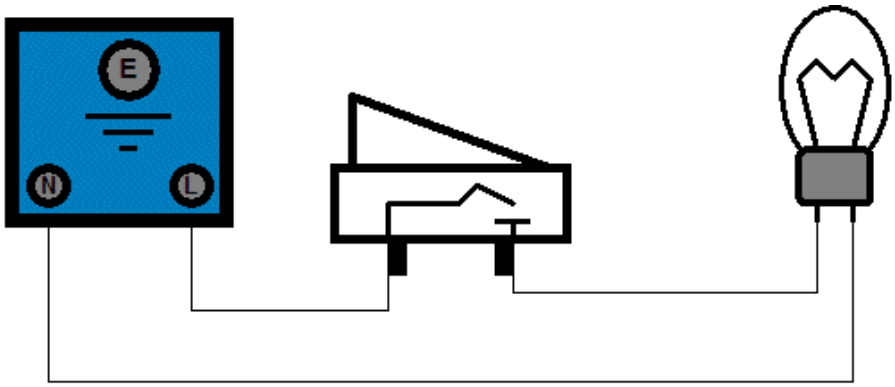


Do you know what this upper terminal is for? Yes, it is for earth in AC and ground in DC. Now tell me what this left terminal is for? It is Line, neutral or either of these? It can be any but universally right side is LINE and lest side terminal is NEUTRAL.

In this diagram you need to connect L N, switch and lamp to get the lamp operated by switch.

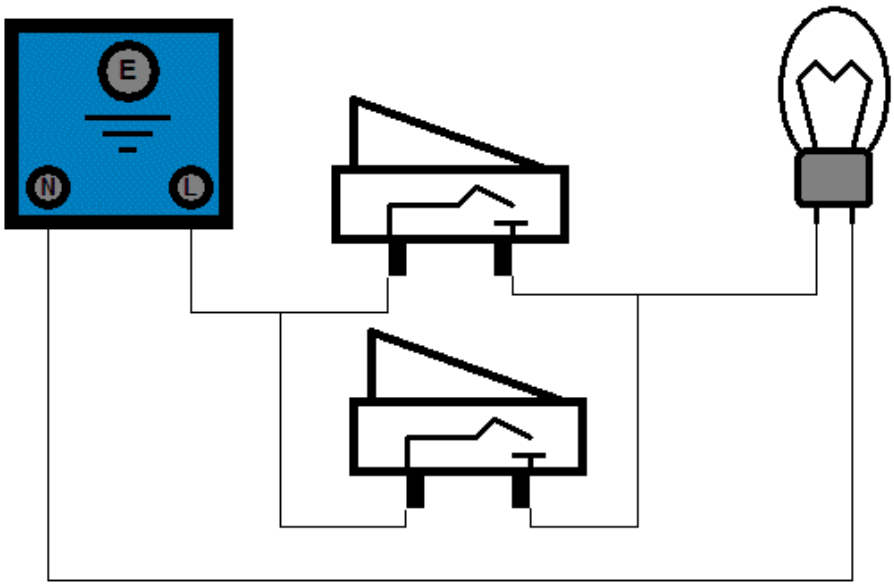



This diagram connection is wrong. Here we have connected line and neutral both to a switch; switch has just a contact inside it. As switch is on L and N will get short.



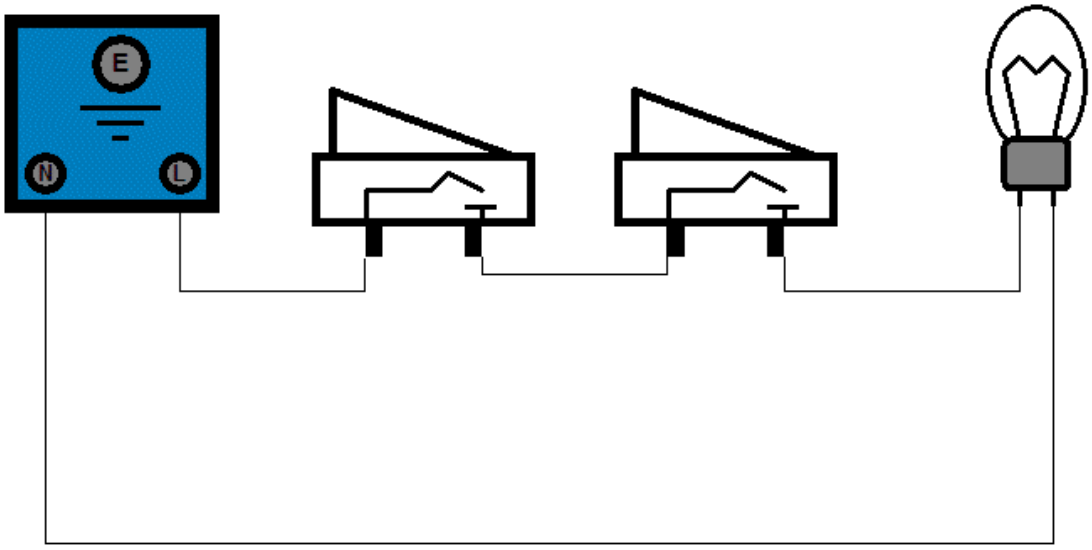

This is correct. Here when switch is on, Line path will get connected and lamp will get on. Always remember that switches are connected to single path not to both paths L and N. in single path too it should be connected to Line path. We don't connect switch to neutral path. It is directly connected to output/ appliance. Connecting switch to neutral not to Line path is not safe. Lamp will be operated properly but, in that case, only neutral path is broken/ disconnected and line is always available to the output appliance. When someone will go to the output for maintenance or repair purpose, he/ she can get touch in Line, as it is directly connected to output, and can meet saviour accident.

1> Now we have L, N, two switches and one lamp. Connect in such way that when you press either/ any switch, lamp should be operated.




This is parallel connection. Press any switch line will be connected and will reach to lamp resulting it on.

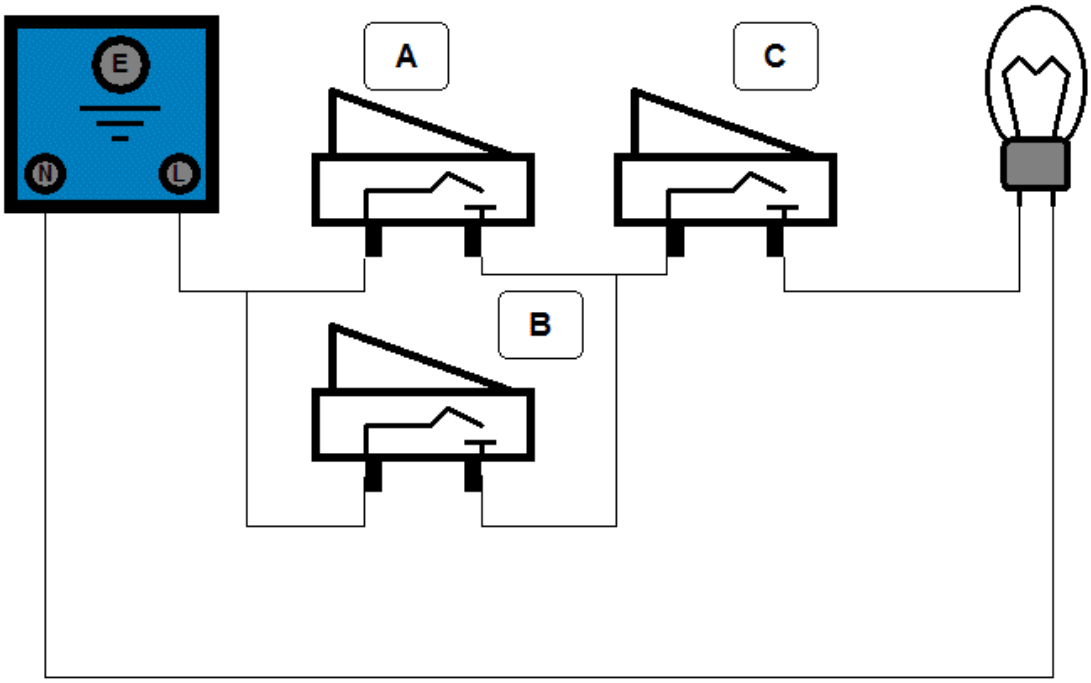
2> Now we have L, N, two switches and one lamp. Connect in such way that when you press both switches then only lamp should be operated not by single switch.




This is series connection.

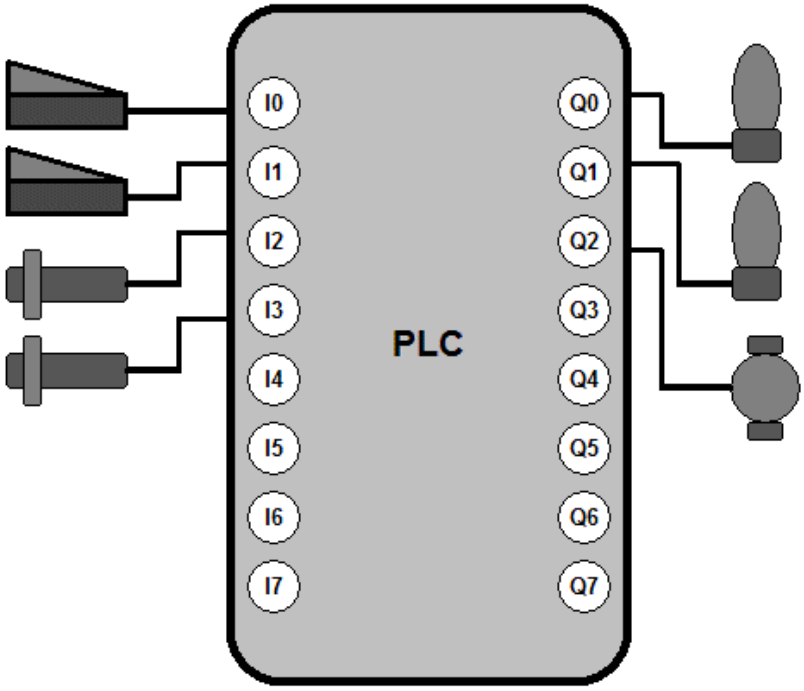
3> Now we have L, N, three switches (A, B & C) and one lamp. Connect in such way that when you press the given combinations B & C and C & A

then only lamp should be operated not by single switch.



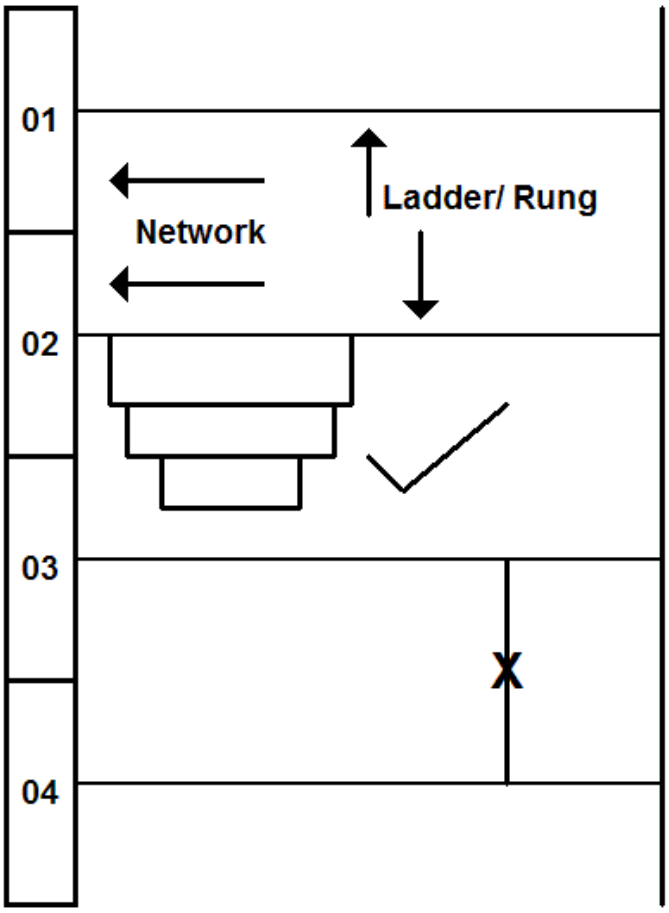

As C is the common switch in both combinations it would be connected in series whereas A and B will be in parallel.

We have seen yet that in wire logic there is a lot parallel and series connection of inputs.it is very difficult some times to satisfy the requirements with wire connection and it requires many other electrical and electronics components. For example, take last connection in consideration where logic is to get lamp operated when either B & C or C & A combination is pressed. Suppose we add interlock condition that it should work for the given combinations only not for other combination like when all 3 switches are pressed or when single switch is pressed. When this condition comes where lamp should not get on when all three switches are pressed then it is very difficult to design logic through wires. If we have PLC then such type of complications will be eliminated. When we have PLC then all switches and outputs will be directly connected to PLC I/ O terminals. There will not be parallel or series connection of inputs.



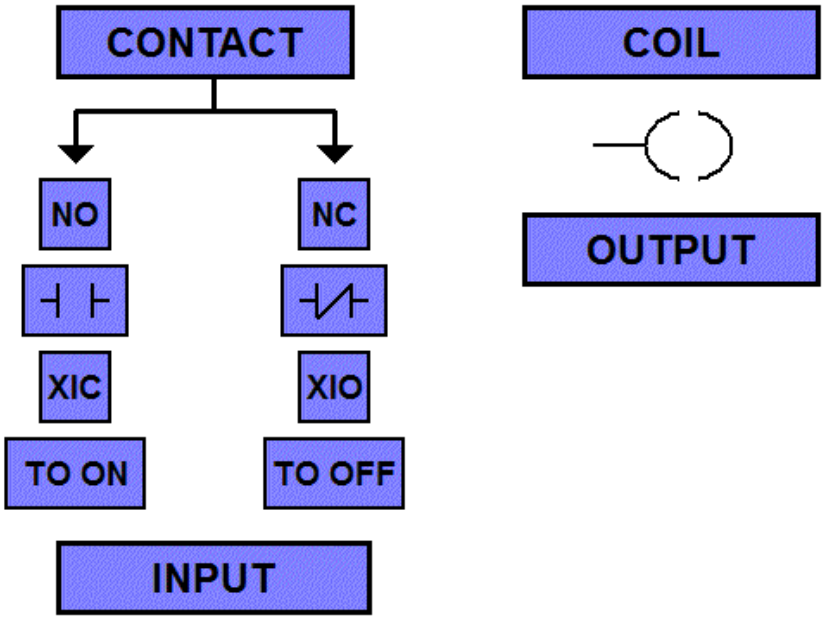

You can see this picture. Here all switches and outputs are directly connected to PLC I/ O terminals. This is not the correct connection. This picture is just to show you that there is no complication in wiring. Correct wiring connection we shall learn further. Any PLC will have fixed input output terminal address. Here we have assumed input as I and output Q. terminal address you can see we have I0, I1, I2…… Q0, Q1, Q2……

Let me give you overview of ladder diagram. When you open PLC software in laptop/ computer, you take a new file. It will ask You PLC hardware/ model name and the programming language where you have to select LD (ladder diagram). As you click Ok, ladder will be appeared as shown below.




You can take many ladders for your logic. Left side it is network and each network has one ladder. Ladder is also called rung. The rule of ladder is that in one network you can take many parallel connections but two networks cannot be connected.

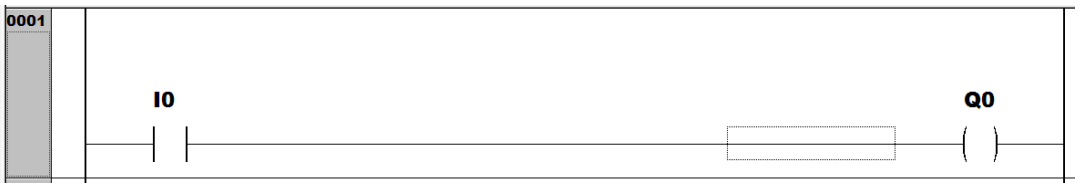
Finally, we are going to start learning PLC programming in ladder diagram programming language. For any language we need to learn instructions. Let us start with basic instruction i.e. contact and coil. Contact and coil will be used 90% in all examples/ programs.



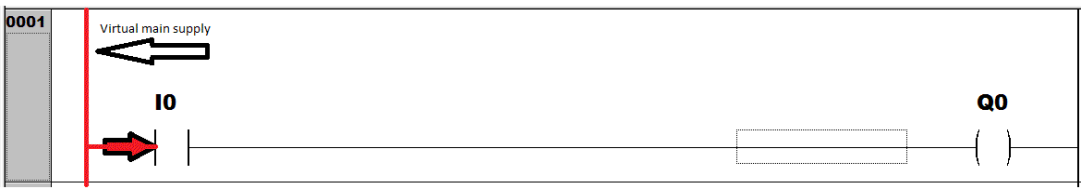

Contact we have of two types; NO (normally opened) and NC (normally closed). It is also called XIC (examine if close) and XIO (examine if open). We shall call it NO and NC. Its symbolic representation has been shown in picture. For now, let us consider that contacts we shall use for input and coil for output. Simplify it more. NO of input we shall use when we have to switch on any output and NC of input we shall use when we have to switch off any output. When NO will be on, it will conduct and when NC is on it will break the path.

Let us understand it by some examples.

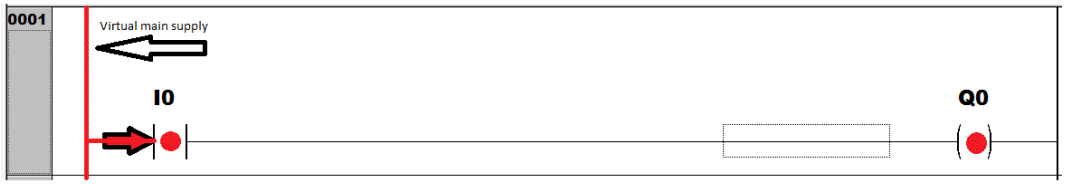
**EX 1: Press the switch connected to I0 terminal of PLC to get motor connected to Q0 terminal of PLC.** We can say this simply I0 on Q0 on.




This is the final program for given example. Let us understand that how does it work.



In this ladder, left line you consider is the main supply (It is virtual/ software part). Now initially (when machine is powered on but no one switch has been pressed) the main supply is up to NO contact. Supply cannot pass through the NO contact as this contact is open.

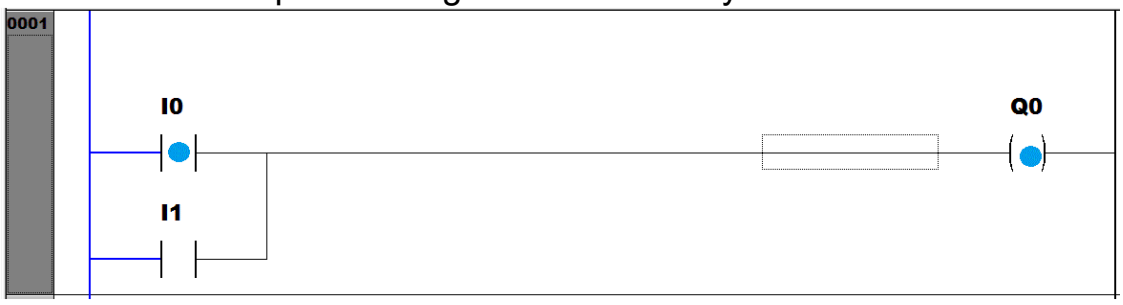



Now when switch I0 (on machine) is pressed, I0 in program gets on. As I0 NO contact gets on it starts conducting and main supply reaches to Coil Q0 and physical output connected to Q0 terminal of PLC (on machine) gets on.

Let us make another example. We shall go basic to advanced program gradually.

**EX 2: We have two switches I0 and I1 and one output/ motor Q0 connected to PLC on machine. When we press either I0 switch or I1 switch (any), motor Q0 should get on.**

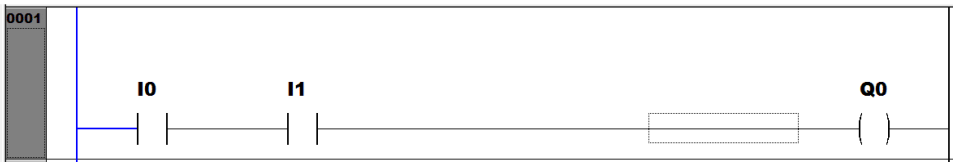

You can see the virtual main supply (in blue) is up to both NO contacts and cannot pass through it as NO initially does not conduct.



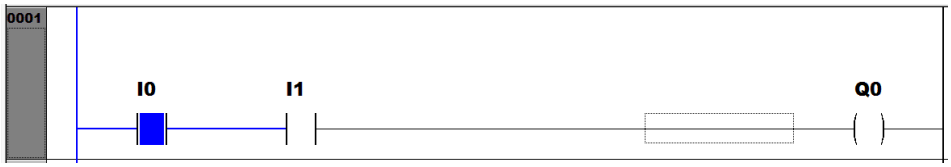

When we press I0 switch on machine, I0 NO contact gets high and supply reaches to coil Q0 resulting it on.


When we press I1 switch on machine, I1 NO contact gets high and supply reaches to coil Q0 resulting it on. You can see whenever either I0 or I1 switch is pressed, output Q0 is on.

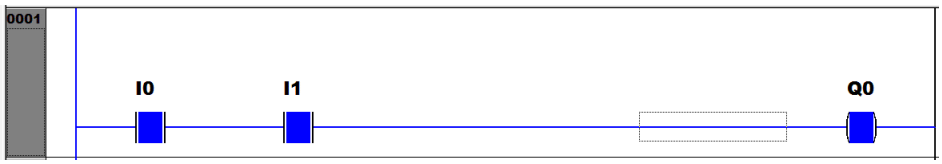
**EX 3: We have two switches I0 and I1 and one output/ motor Q0 connected to PLC on machine. When we press both switches I0 and I1 switch then only motor Q0 should get on. Q0 should not get on by single switch.**




**You can see that virtual main supply is up to I0 contact.**




When I0 is high, it conducts and supply reaches to I1 contact but I1 is still not pressed so it is not conducting.

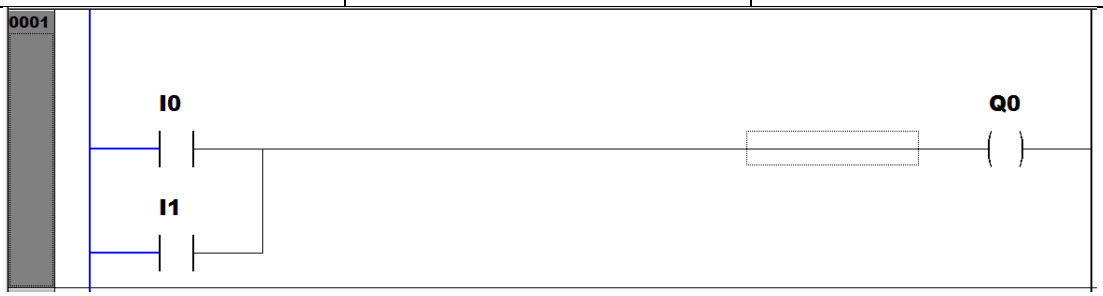



Now I1 is also pressed and I1 NO starts conducting so main supply reaches to Q0 and Q0 is on.

I hope you understand the concept of NO. It does not conduct initially. When it is on then it starts conducting.

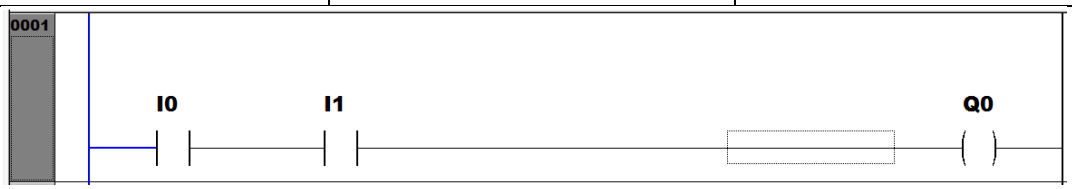
In college exam or in interview you might be asked to make program for digital gates. Already we have made program for two gates i. e. Or gate and AND gate. EX2 is OR gate as you can see the condition I0 **or** I1 is pressed Q0 is on.

| I0 | I1 | Q0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



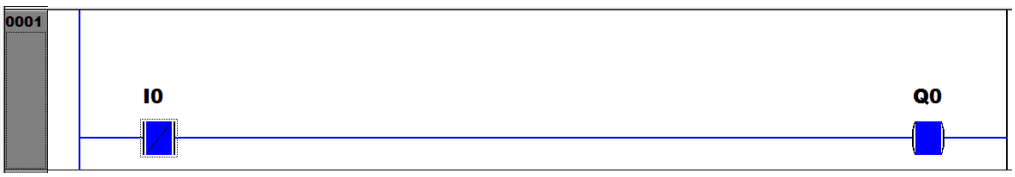

EX3 is AND gate as you can see the condition I0 **and** I1 is pressed then only Q0 is on.

| I0 | I1 | Q0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

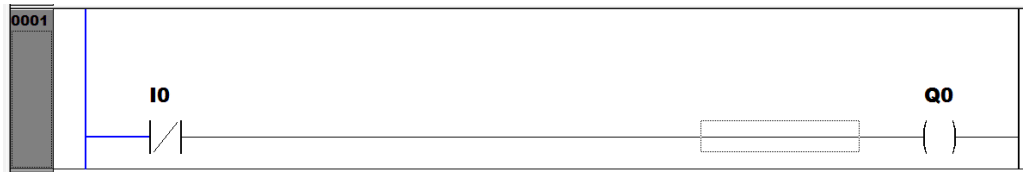



**EX4: NOT gate. We have one input I0 and one output motor Q0. When input is not pressed output should be on and when input is pressed output should be off. So here it is totally opposite. Motor should run without pressing switch. It means virtual supply should reach to coil directly and when switch is pressed motor should be off i. e. when switch is pressed main supply path should be broken.**

| I0 | Q0 |
|---|---|
| 0 | 1 |
| 1 | 0 |

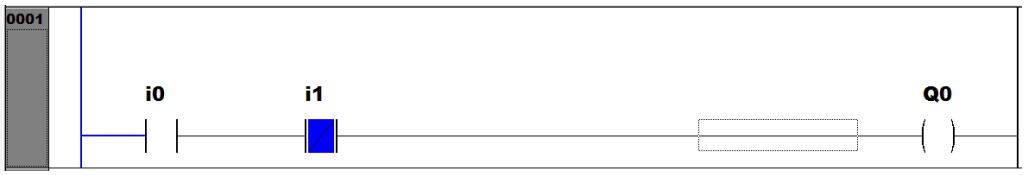


You can see I0 is NC contact and NC contact already conducts initially when it is not pressed. Because of NC contact, the main virtual supply reaches to Q0 coil and it is on initially. Be very clear that switch on machine is off now and motor is on. This is I0 off Q0 on condition.

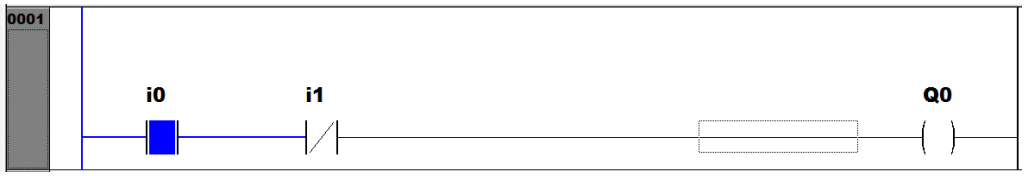



Now you see I0 is pressed and NC has broken the path resulting virtual main supply topped and Q0 off. This is I0 on and Q0 off condition.

So finally, we understand that NO conducts when it is on and NC breaks when it is on.




This is initial condition when I0 and I1 both are off. In off condition NC is white i. e. not conduction but NC is blue i. e. conducting.

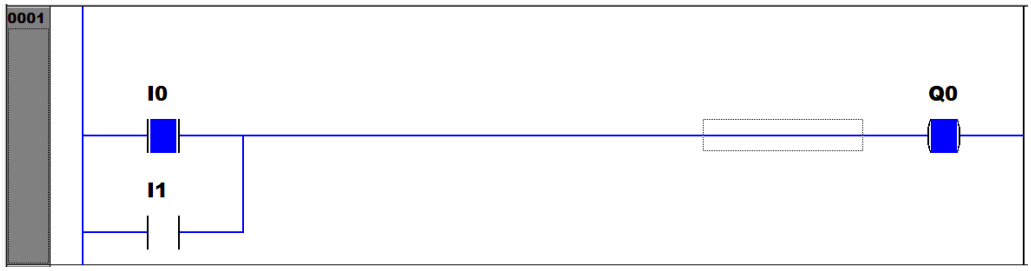



This is on condition when I0 and I1 both are pressed (on). In on condition NO is blue i. e. conducting but NC is white i. e. not conducting.

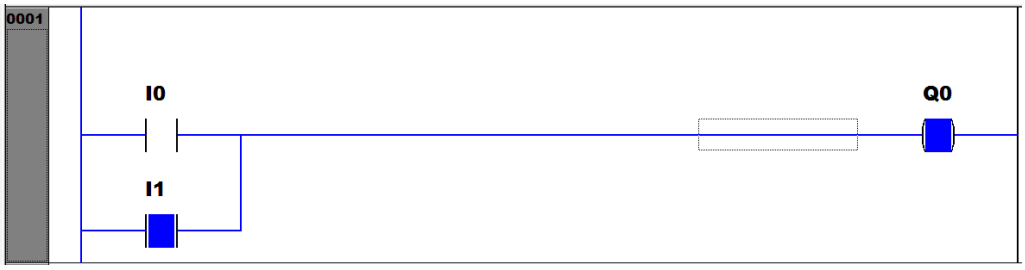
You keep this in mind always that ‘NO conducts when it is on and NC breaks when it is on’.

So, yet we have understood that to switch on we use NO and to switch off we use NC contact in program.  Now we proceed and make some more examples. Further I shall share the easiest way to make programs. In industry you will hear a term interlock. When client gives you some conditions to operate output, your output should be operated for the given conditions/ combinations only not for any other condition/ combination. For example, let us check parallel input program (EX2). Condition was to get Q0 on by pressing any single input I0 or I1.

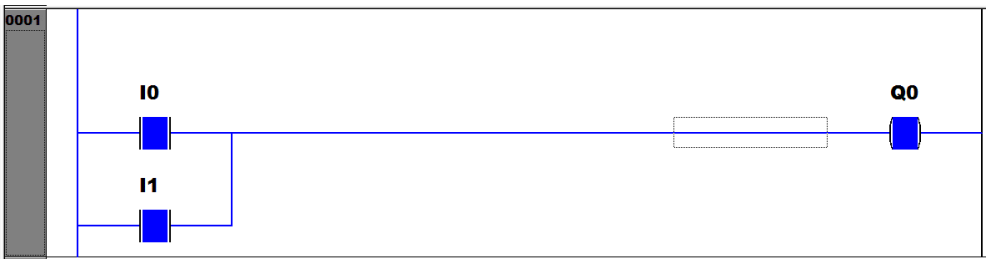

Let us check for given condition i. e. single switch press to get Q0 on.




Here we press single switch I0 and Q0 is on. Good! Let us check by another single switch.



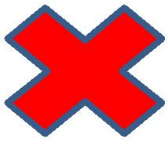

Here we press single switch I1 and Q0 is on. Very good! You must be very happy because your program is working as per client's conditions. Now client will check interlock. Q0 should get on by pressing single switch. It means Q0 should not get on by pressing both switches together. Let us check.



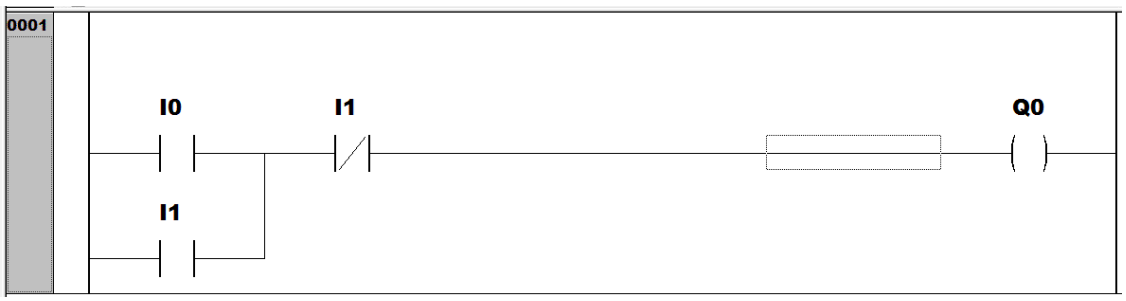

See this. When I0 and I1 both switches are pressed together then also Q0 is on. It means your program is wrong for industry as you have not followed interlock. Let us make program with interlock. There is a gate which has interlock condition. XOR (Exclusive OR)

**EX5: XOR (Exclusive OR)**

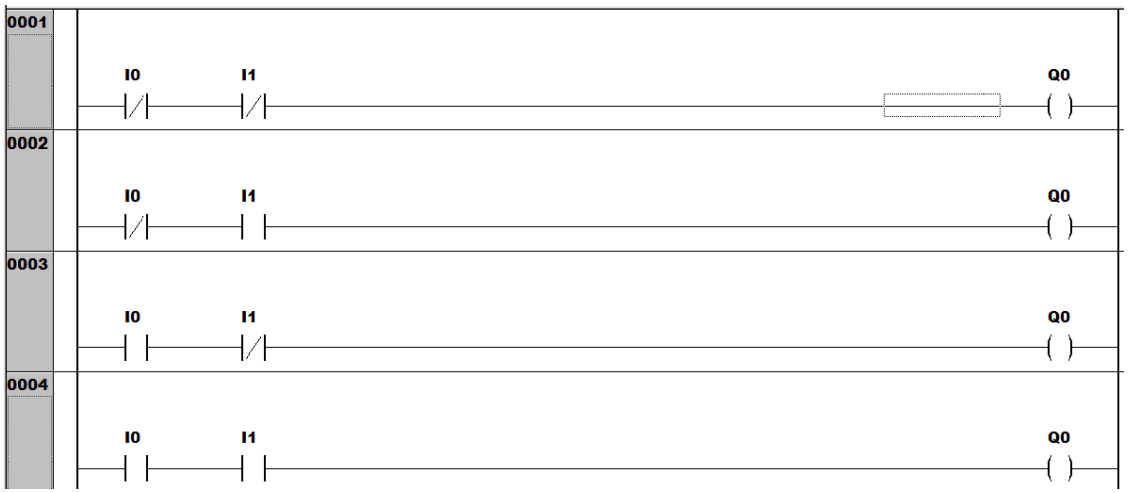
| I0 | I1 | Q0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




Wrong!




Wrong!




Wrong!

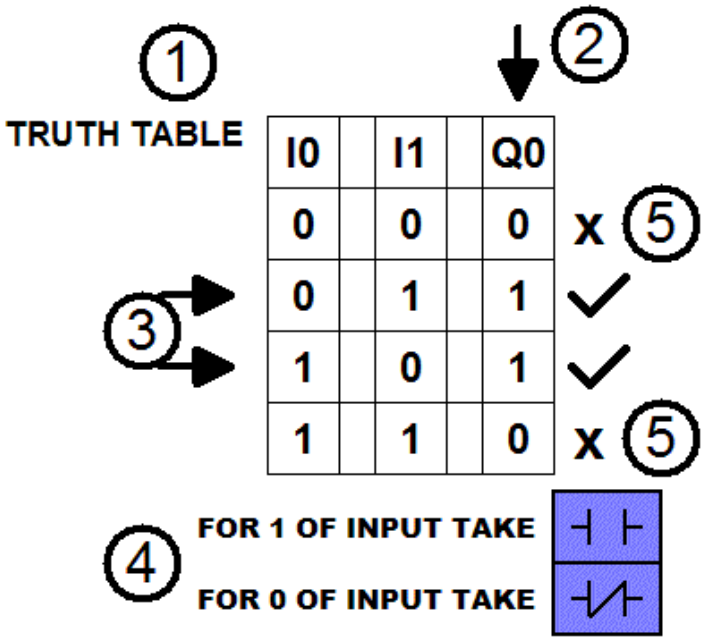
These are wrong solutions. Let me remind you the ladder rules again. If we have one output in our program then we shall have only one output coil. Do not repeat coil address. Input we can repeat many times. It means many NO NC contacts can be used and I0 & I1 will be repeated as per need.

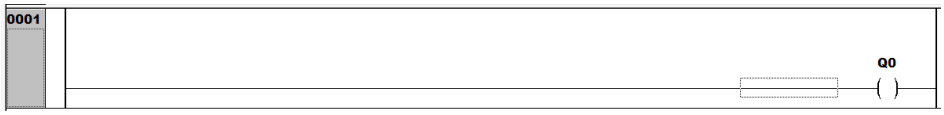
Now let us make the things easy. Follow some steps and you will be able to make any basic program. When you have to make program for simple conditions like just on and off (no complication of pulse, timer, counter etc.), you can follow some steps.

1> Make a truth table for given conditions.
2> Go to the first output and take a coil for it. Don't think about all outputs in your truth table. For some examples, we shall start with coil.
3> Check the true conditions of output and add the status of inputs used in example.
4> For input 1 take NO and for input 0 take NC.
5> Do not make program for 0 (zero) of output.

Repeat the same for other outputs.

Let us make XOR program by following above steps. 1) Make a truth table.

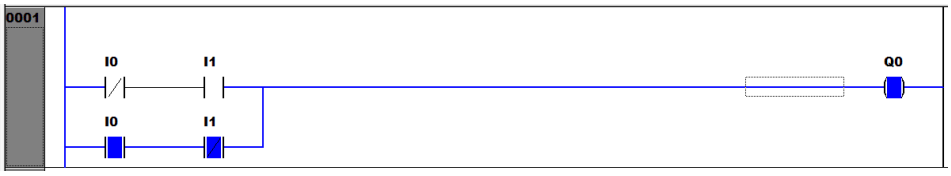


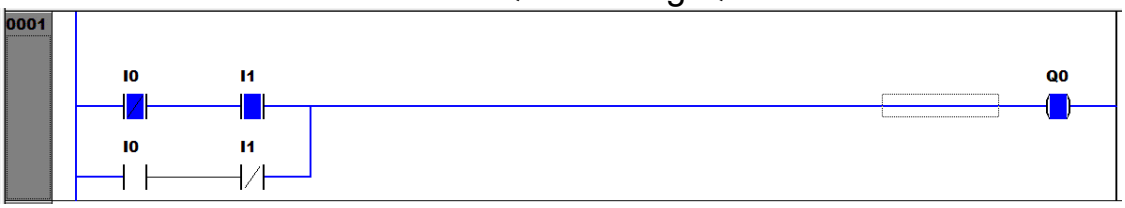

2) Now go to the first output Q0 (here we have only one output) and take a coil for it.


3) Check the true conditions of output and add the status of all inputs i. e. I0 and I1. 4) For 1 of input take NO and for 0 of input take NC. 5) Do not make program for 0 of output.

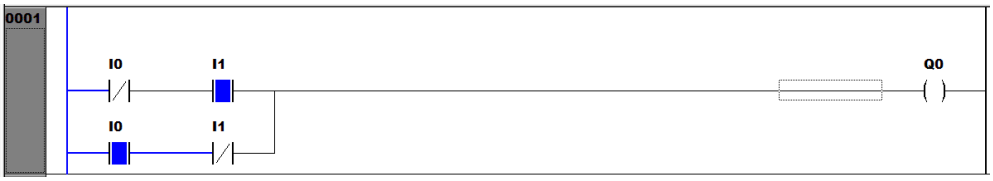
When output Q0 is 1 first time, status of inputs I0 & I1 is 0 & 1 respectively so ladder is NC of I0 and NO of I1. When output Q0 is on again, status of inputs I0 & I1 is 1 & 0 so ladder in parallel is NO of I0 and NC of I1.



When only I0 is pressed lower parallel path gets active and virtual supply reaches to Q0 resulting Q0 on.




When only I1 is pressed upper path is active and virtual supply reaches to Q0 resulting Q0 on.



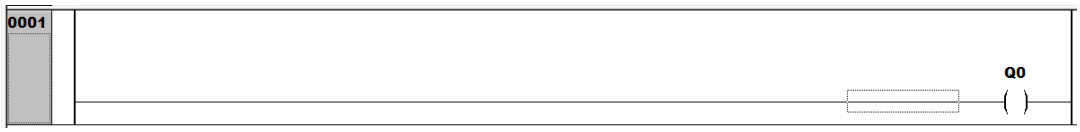

When I0 and I1 both are pressed together, I0 NC breaks upper path and I1 NC breaks lower path resulting Q0 off. You can see both NC contacts are white. When NC is white it means it is pressed and breaking the path. Let us practice some more examples based on truth table.

**EX6: we have 3 inputs I0, I1, I2 and one output Q0. Press any two switches to get Q0 on.**

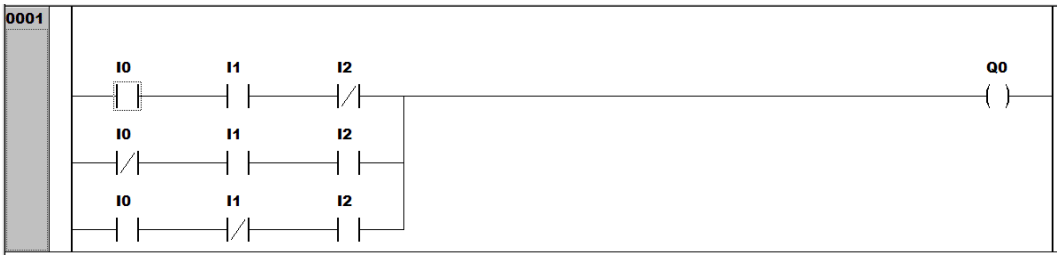
Obviously Q0 won't be on by pressing single switch and by pressing all 3 switches together. We can have 3 combinations to get Q0 on; I0 & I1, I1 & I2 and I0 & I2. Let us follow the steps.

| I0 | I1 | I2 | Q0 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

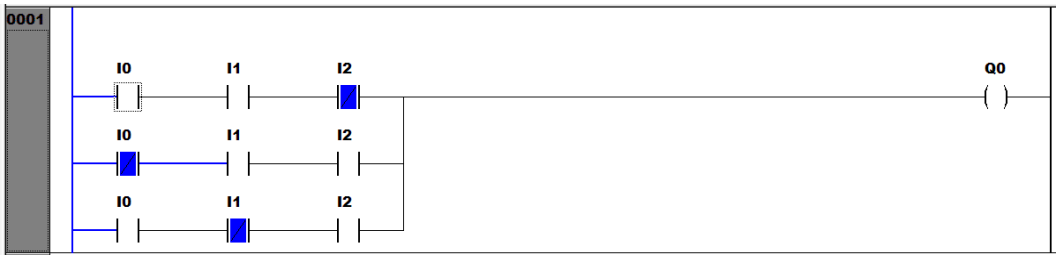
Make a truth table.




Go to the first output and take a coil for it.

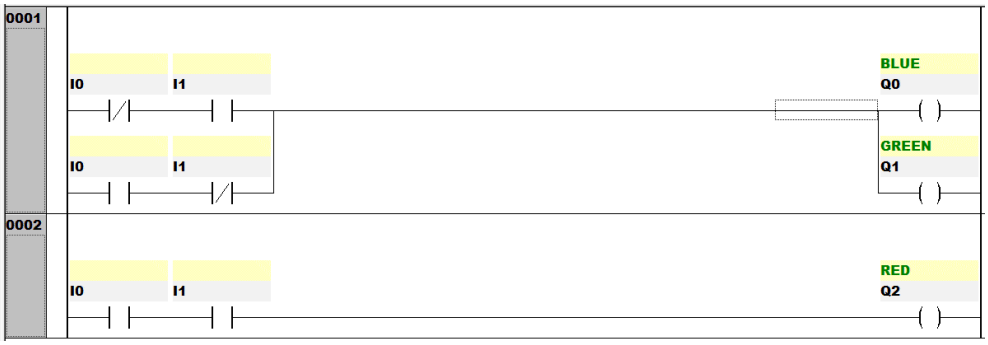



Check the true conditions of output and add the status of all inputs used in example.

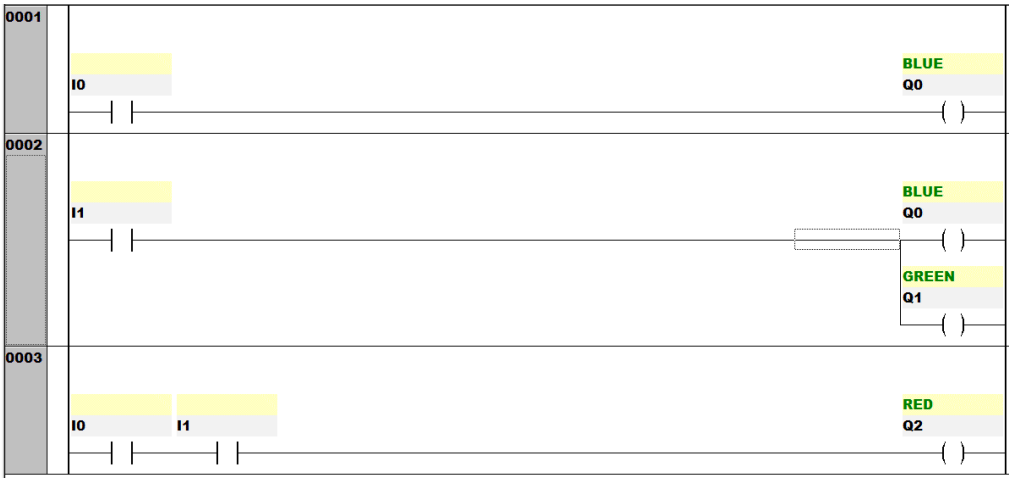



This is the initial status of supply.

**EX7: there are two switches I0 and I1. There are 3 outputs Q0, Q1, and Q2 (blue, green and red lamps). when I0 is pressed green lamp (Q1) should get on. When I1 is pressed blue and green both lamps (Q0 and Q1) should get on. When I0 and I1 both switches are pressed only Red lamp (Q2) should get on.**




Wrong!

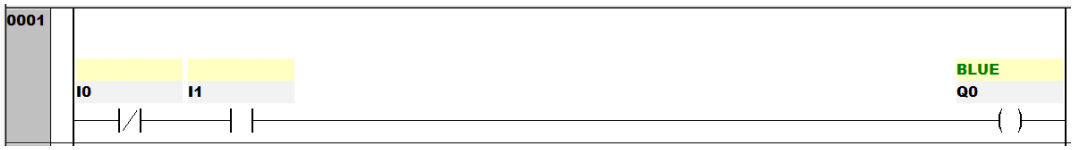



Wrong!

Don’t be over confident but please follow the steps. Don’t think about all conditions. Just go stepwise. Better make separate program for each output.

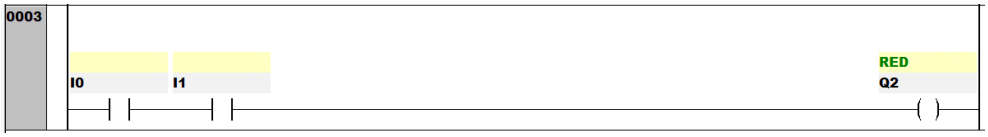
| I0 | I1 | BLUE Q0 | GREEN Q1 | RED Q2 |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

Make a truth table.

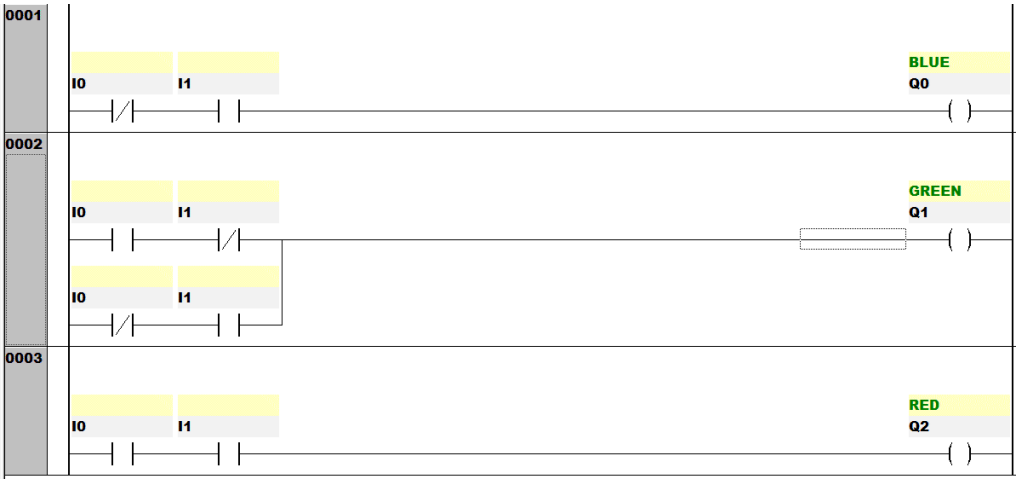



Go to the first output Blue and take a coil for it. When you are making program for Blue don’t think about other output. Don’t relate one output with other. Check the true condition of Blue and add the status of all inputs. Blue is on once only. Blue is finished.


Go to the second output Green and take a coil for it. When you are making program for Green don’t think about other output. Don’t relate one output with other. Make ladder separately for Green. Check the true conditions of Green and add the status of all inputs. Green is on twice so parallel ladder will be there to Green. Green ladder is finished.




Repeat the same for Red. Take a coil for it. Check the true condition and add the status of all inputs used. Red is also finished.



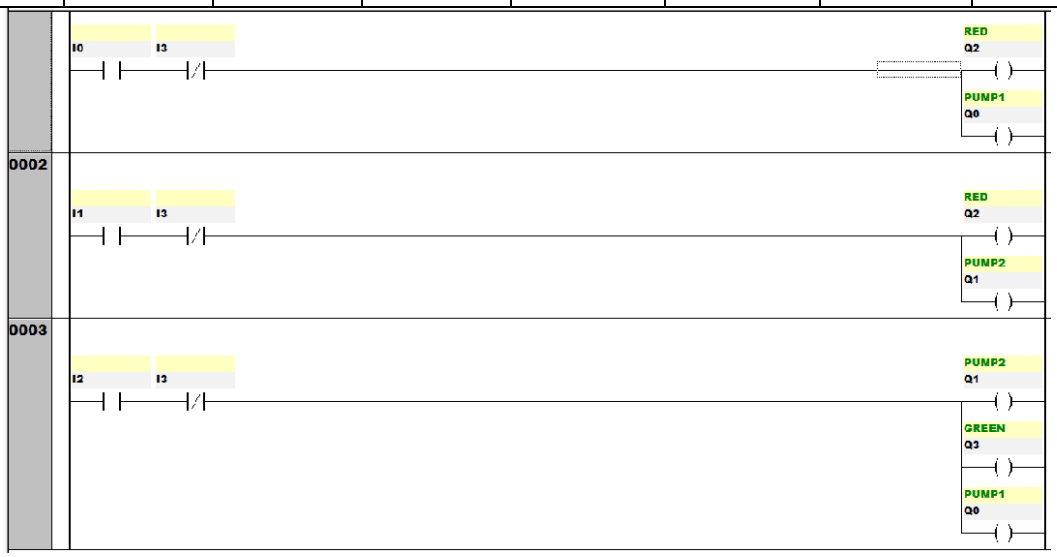

So, it is very much clear that when we follow steps without thinking too much, programming is easy. This is the final program.


This is the initial status of virtual supply.

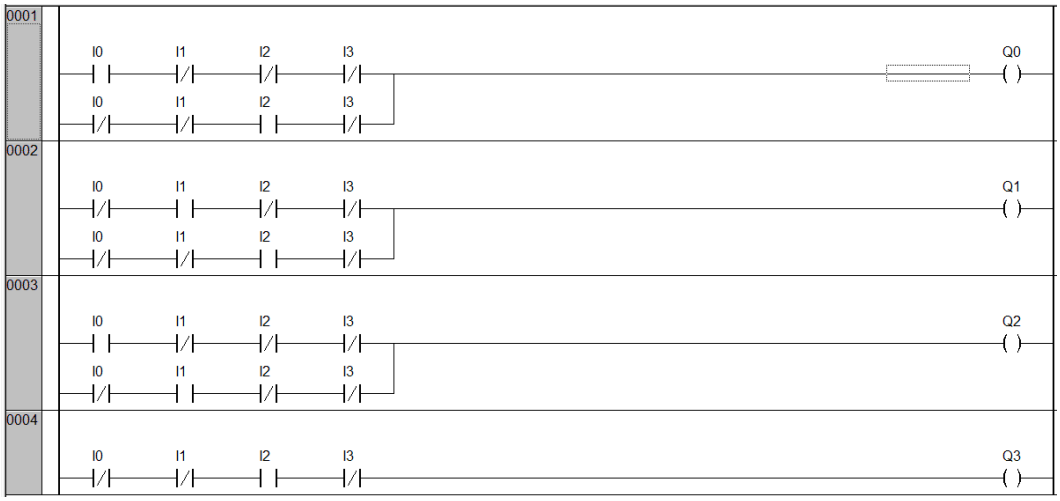
One more example to practice steps. I hope this one you will make correct.

**EX8: four inputs I0, I1, I2, and I3. Four outputs Q0, Q1, Q2 and Q3 (Pump1, Pump2, Red and Green lamps). When I0 is on Pump 1 and Red will be on. When I1 is on Pump 2 and Red will be on. When I2 is on Pump 1, Pump 2 and Green will be on. When I3 is on all outputs will be off.**

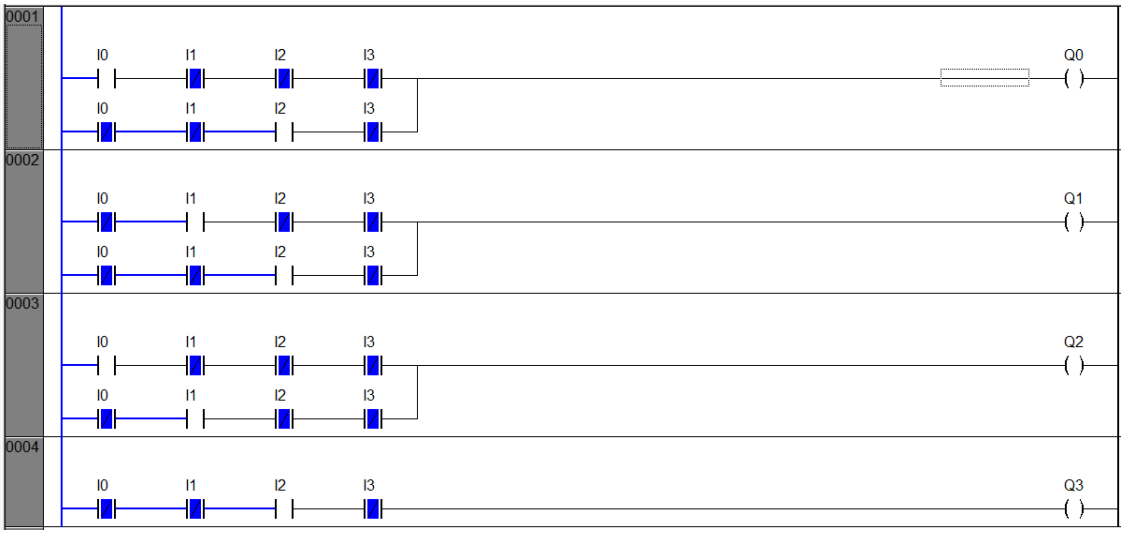
| I0 | I1 | I2 | I3 | Q0 Pump 1 | Q1 Pump 2 | Q2 Red | Q3 Green |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |




Wrong!

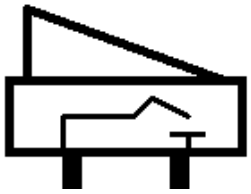



**Correct program**

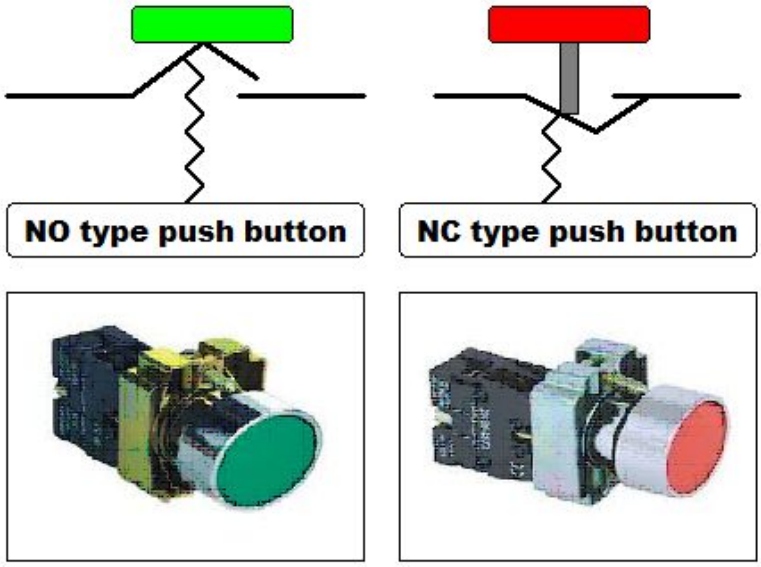



**Correct program with initial status of Virtual supply.**

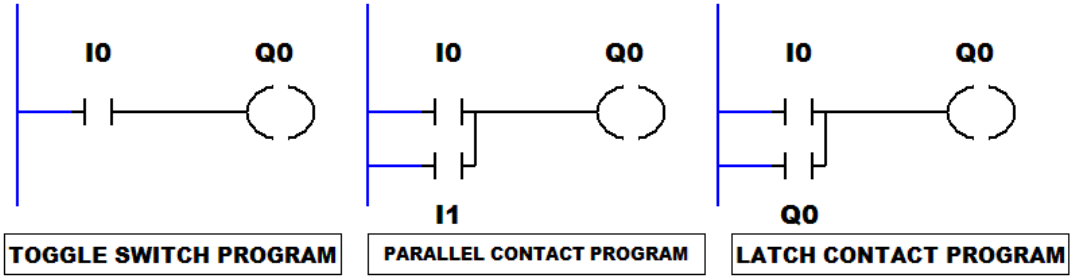
In all our examples we have used toggle switch. When we press the switch it is on, when we leave the switch it holds its position and is continue on. This is toggle switch property. Normal switches that we use at our home are toggle switches.



In industries most of time push button will be used.



Push button has spring inside it. When we press the button, it conducts. When we leave the button due to spring its contact is open and it stops conducting. We don't get supply through it any more as we leave the push button. For push button none of our examples will work. We need to modify our programs if we want those to function for push buttons. The modification is called Latch.



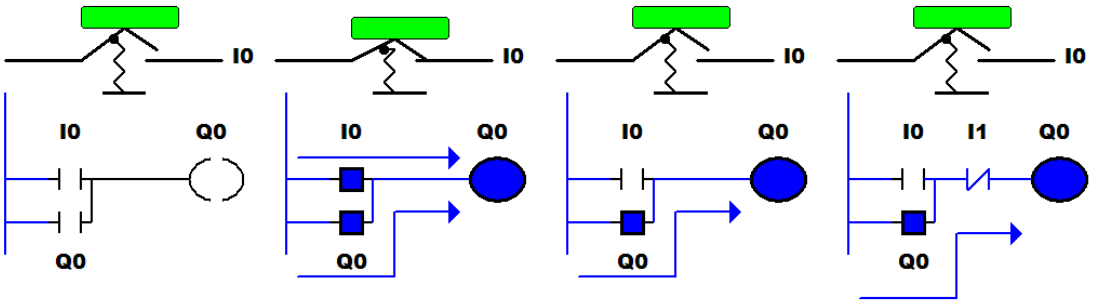

In these 3 ladders shown above, first one (left one) is the program for toggle switch. It will work for toggle switch. As the switch I0 is on Q0 will get on. When you leave the switch then also switch is continuing on so Q0 will be continue on until I0 switch gets off.

Second ladder (middle one) shows parallel contact connection. It is OR gate program.

Third ladder (right one) shows a latch contact program. When coil address will come in parallel to input as NO, it will be called latch contact not parallel contact. Parallel contact is shown in middle program.

Any bit address (Input, Output or Flag) can be given to contacts (NO/ NC) but to coil only Output or Flag address can be given not input address.

Let us understand that how will a latch contact make a coil continue on?



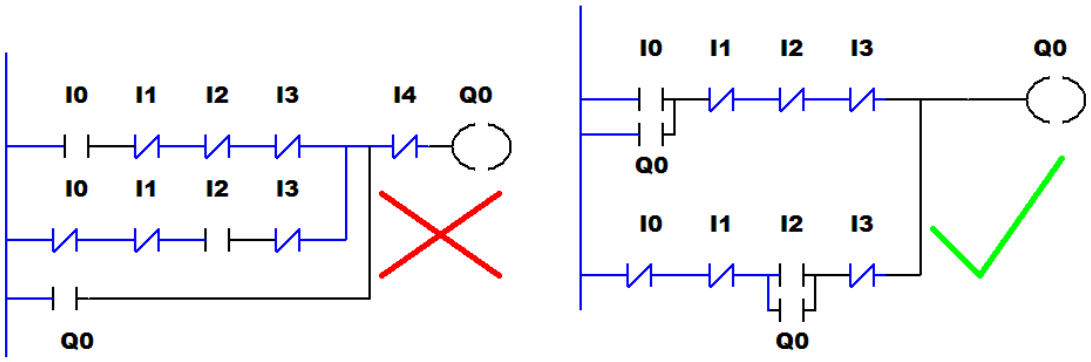

You can see first diagram where push button is connected to I0 terminal of PLC. We have a ladder where I0 NO and a Q0 coil with a latch. First diagram shows the initial status where push button is not pressed and in program nothing is on. Virtual main supply is up to I0 NO and Q0 NO.

In second diagram push button is pressed. Now I0 contact is high and main supply reaches to Q0 coil. As Q0 coil gets on, at the same time Q0 latch contact is also high because of same address. Now you can see that virtual supply in program reaches to coil Q0 through I0 contact and through Q0 latch contact too. So, both upper and lower paths are active and push button is still pressed as shown in middle / second diagram.

In third diagram push button is released and due to spring push button contact is open i. e. push button is off. It results I0 contact off so upper path is not active now but still virtual main supply reaches to coil Q0 through latch contact/ path Q0 and it results Q0 coil continue on.

This is the concept of latch. Now there is a problem. Once Q0 coil is on and is continue on because of latch, it will not get off (third diagram). To get coil Q0 off we need to break latch path by some NC contact. You can put one NC to break latch and can give address to that NC as per condition. I have given I1 as NC break in last diagram. When I1 is pressed, NC will break the latch path and Q0 output coil will get off.

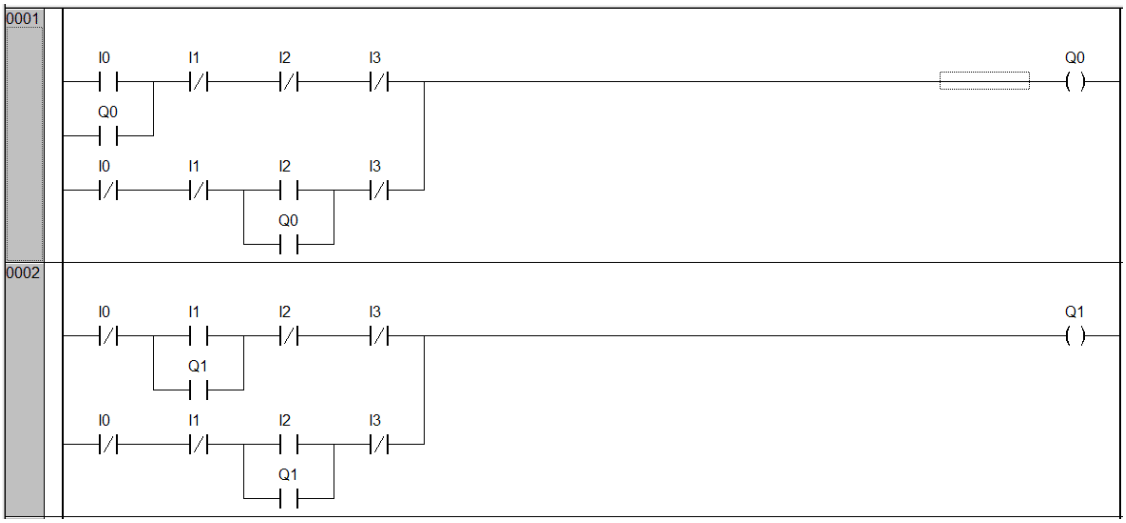
Let us modify some previous examples for push buttons and latch.

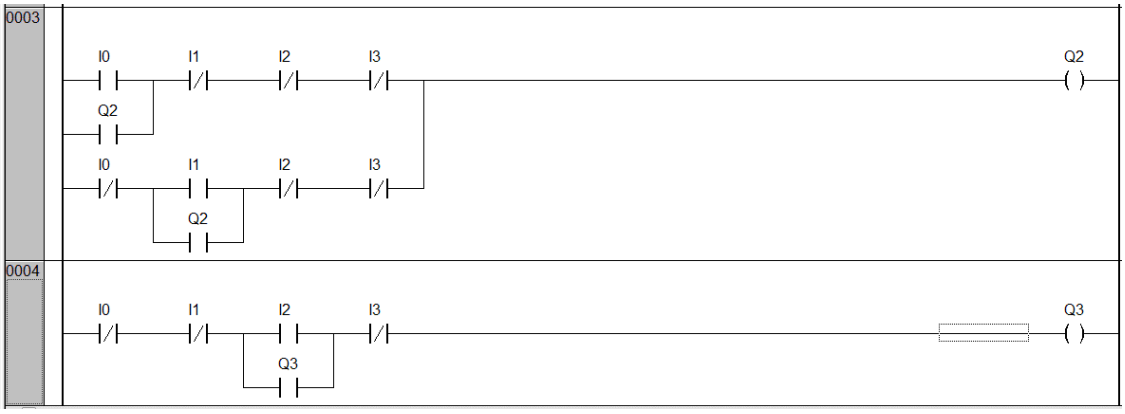



Here we shall latch the first network ladder of last example EX8. See the left side diagram. When we think about latch, we generally think to make latch like this. If we make a latch like this (left side diagram), we shall have some problem. If we make a latch it must be some NC contact to break that latch. Then we put one NC contact in latch path. Another issue raises that if we have one NC contact, it will require one address. Now what should be the address of that latch break NC contact? Will it be I4 because up to I3 we have used in our given program EX8? No, we cannot use a new input address as client has given us only 4 switches I0, I1, I2 and I3. We cannot use a new switch to break latch. We need to give latch break NC address from our program only. So, you can see left side latch program is wrong.

Better you follow one easiest way to convert toggle program in to push button/ latch program. **To convert toggle program in to push button program, take latch of output (coil address) below each NO contact in that network ladder.** You can see in right side diagram I have taken latch of Q0 below each NO contact and already there are many NC contacts available to break latch. We do not need to put new NC contact to break latch. NC contact can be placed before or after latch contact, there is no problem. I hope latch concept is clear now. You should latch all 4 outputs of EX8 for practice.

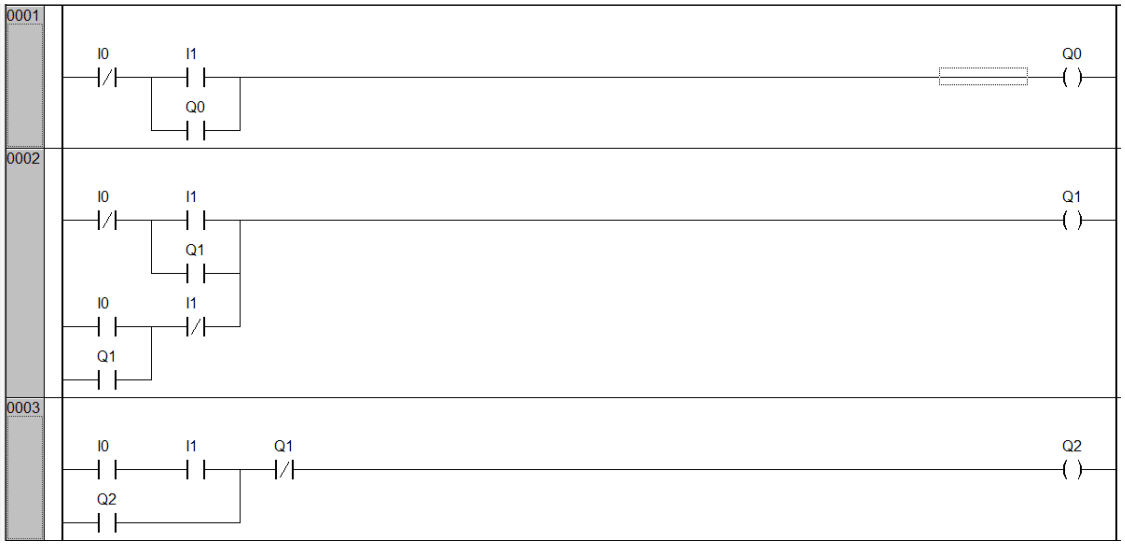
**EX9:**




So, take latch of output below each NO contact in that network ladder. NC is already there in program. If NC is not available in some network then you need to put one NC and address from your program. Let us latch EX7 (B, G & R).

**EX10:**

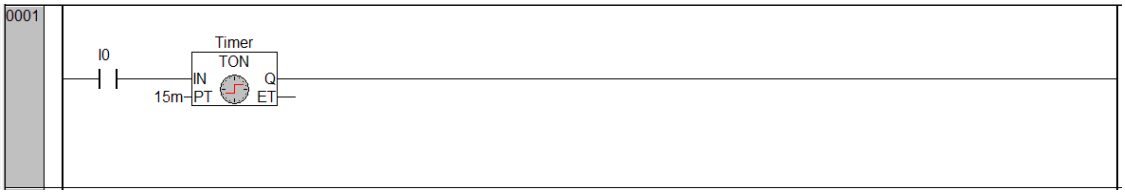



In network 1 we already have NC to break latch. In network also we already have NC to break latch but in network 3 there is no NC from before so we need to put one NC contact to break latch. OK we put one NC, now what should be the address of that NC contact? This address must be from our program only. Let us think. Q2 gets on when I0 & I1 both are pressed together. It means when single switch either I0 or I1 is pressed Q2 should get off. Let us put I0 or I1 NC in third network to break latch and check. One problem occurred that now Q2 will not get as we press I0 and I1 together to get Q0 on, I0/ I1 NC breaks the path so it is clear that we cannot put I0 or I1 NC to break Q0 latch. Now think again. Is there any signal getting on when we press single switch I0 or I1? Yes, it is Q1. When we press either I0 or I1, Q1 gets on and that time Q2 is off so we can put Q1 NC to break Q2 latch.

Yet we have made program for very basic examples. For a good automatic machine program, we need to learn some more instructions.

**Flag** is one instruction without which it would be impossible to make big industrial system program. **Flag is an internal bit memory which has no any physical existence outside of PLC. Flag is a bit memory like input and flag. It also gets on and off. Outside PLC it has no any existence; it is in program only and supports our logic.** If you compare flag and output, you will find some similarities like; output is a bit memory, flag is also a bit memory. In coil we use output and in coil we use flag too. Then what is the difference? In your program if you have a coil with output address suppose Q0, when this coil is on, something outside PLC will get on. Q0 has

physical existence. First existence of Q0 is Q0 terminal on PLC which gets on and further motor, fan, lamp, relay etc. are the physical existence of output Q0. In your program if you have a coil with flag addresses suppose M0 (it is denoted by M in most of PLC), when this coil is on, **there is nothing on outside of PLC called flag. It has no physical existence.** It is used to support our logic. Then where and why is a flag used in PLC program? **Flag** is **used to store temporary bit signal.** Let us understand it by one example. Suppose there is an on-delay timer (we shall learn timer further in detail). There is a push button I0. When we press I0 push button, on-delay timer should run for 15 minutes. How shall we write its ladder?

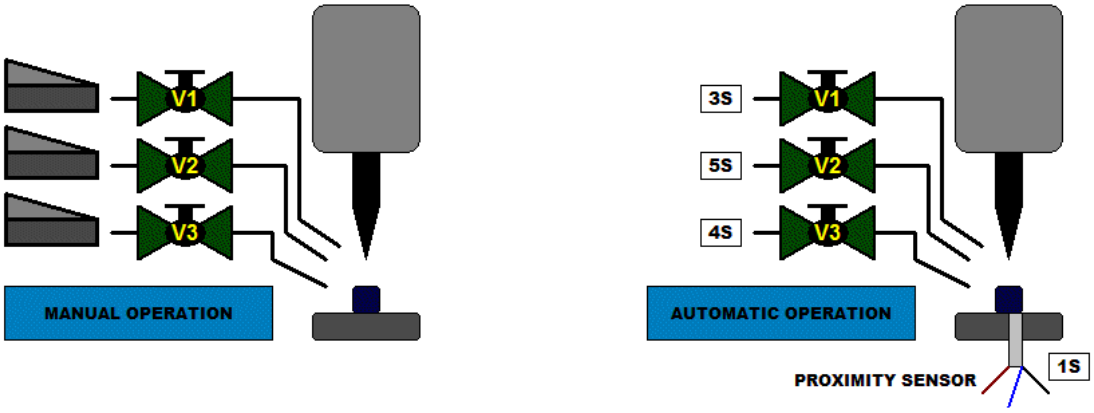



Is this correct? No, it is not correct because an on-delay timer needs continue supply to run but I0 is a push button which we are not going to press for 15 minutes. I0 is a temporary signal here.


This is correct. I0 is a temporary signal here and we cannot latch a timer and also, we cannot use any Q address here as in this example there is no output. if we had any output then we could have latched output and we could run TON easily but here it is only a push button and one TON. In this case flag will be used. Without flag it is not possible to make this program. I told you that flag we use to store temporary bit signal so let us store I0 in M0 first (see Network 1). Now I0 is stored in M0 so we can do all what we had to do

with I0. We have taken TON from M0 in network 2. As M0 is latched it is continuing on and TON needs continue signal. What will happen after 15 minutes that we shall see further. Here we understand that flag we use to store temporary bit signal.

**TIMER:** timer the main instruction to make any machine automatic. Whenever we have delay required in any sequence of input/ output operation, timer is used. In automation timer plays a major role. Let us understand it by one example. Consider we a manual drill machine and we need to make it automatic. First let us see the manual operation.

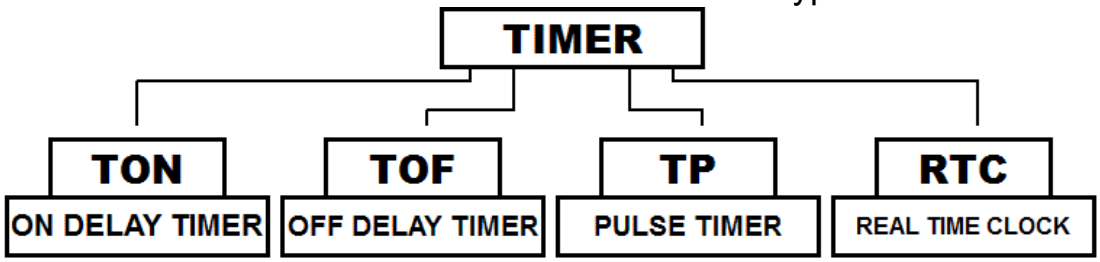



In manual operation one operator/ man will put the metallic job on work station. There are 3 valves; V1 for air which will remove dust from the job, V2 for some chemical and V3 for coolant. Operator will put the job on work station and will switch on V1. Suppose it should be on for 3 seconds. Operator will see the watch and after 3 second he will switch off V1 and will switch on V2 for 5 seconds. After 5 seconds he will switch off V2 and will switch on V3 for 4 seconds. After 4 seconds he will switch off V3 and will switch on drill. In manual operation you can understand that it will not be an accurate operation because manually it is not possible to maintain exact delay and a man cannot do the same task repeatedly for long duration.

Let us convert this system in to automatic operation. For that we have put one proximity/ metal detector sensor to sense the job presence. The sequence we can have is; when job is put on work station by any mean, the

sensor will be on. As sensor is on, in 1 second V1 should get on automatically without pressing any switch. As V1 is on, it should be on for 3 seconds. After 3 seconds V1 is off and V2 is on for 5 seconds automatically. After 5 seconds V2 is off and V3 is on for 4 seconds. After 4 seconds V3 is off and drill is on. Here you can see that there is no any operator to operate outputs. If there is no any operator then who does operate outputs? It is timer. You can notice that with each condition we have a timer to switch on and to switch off outputs. So, we understand that timer plays a major role in automation. When outputs of a system/ machine get on and off itself without involvement of human, the system/ machine will be called automatic.

Timer we have retentive and non-retentive. We shall understand the difference later. Timer we have of 4 types.




But all these four types of timer are not available in all PLC software. TON is available in all brands of PLC software. We shall use TON in all our examples so that the same logic can be used in all brands.

**On Delay Timer:** TON needs continue supply to run. It has 4 parameters: EN, PV, ACC and DN.

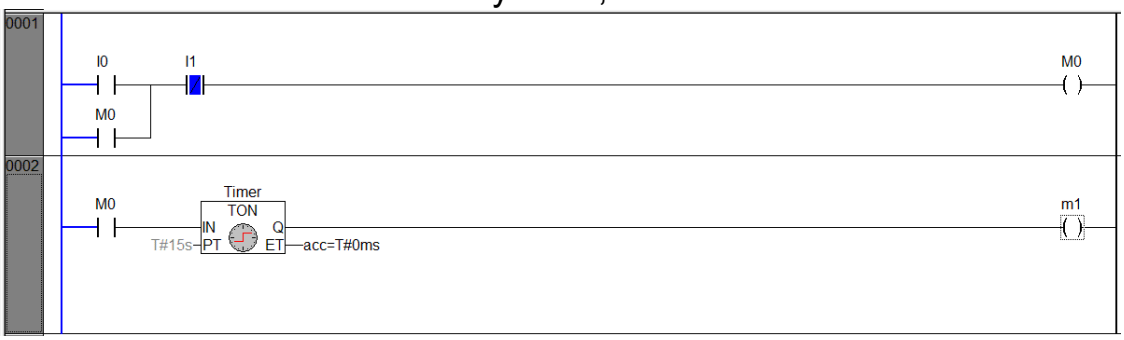
**EN/ Enable bit:** Enable bit is the signal from which timer starts running and for on delay timer it should be continue on.

**PV/ Pre-set valve:** pre-set value is the set point. How long you want to run the timer you need to mention there. It can be millisecond/ MS, second/ S or hour/H.

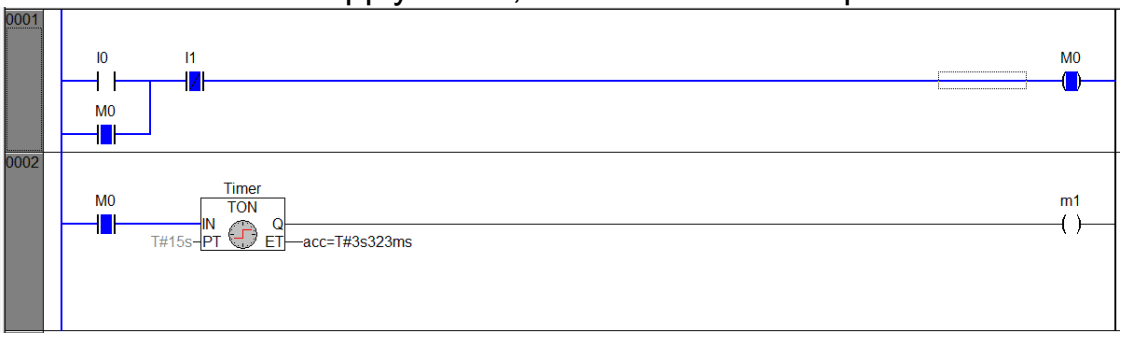
**ACC/ Accumulator:** accumulator is the running/ present/ current value of timer. You can say it is a display where we monitor the running value of timer.

**DN/ Done bit:** Done bit is the confirmation signal of timer. When ACC is equal to PV, done bit is on.

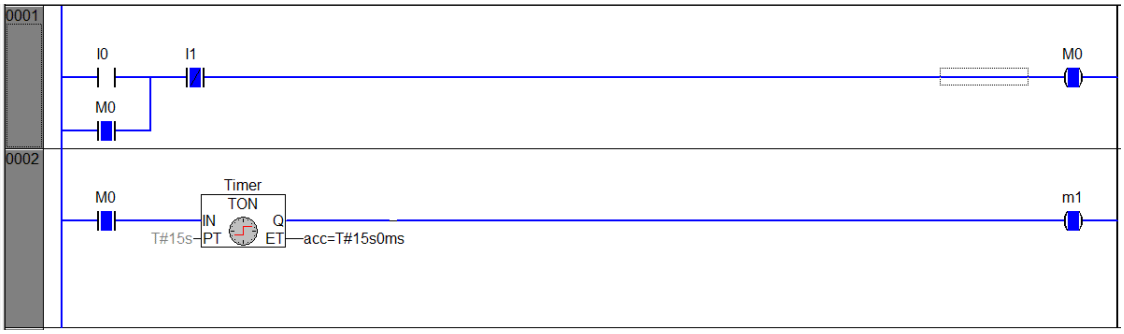
To reset on delay timer, break its enable.




Here I0 is a push button, when it is pressed TON should run for 15 seconds. As TON needs continue supply to run, we need to latch I0 pushbutton with a flag.



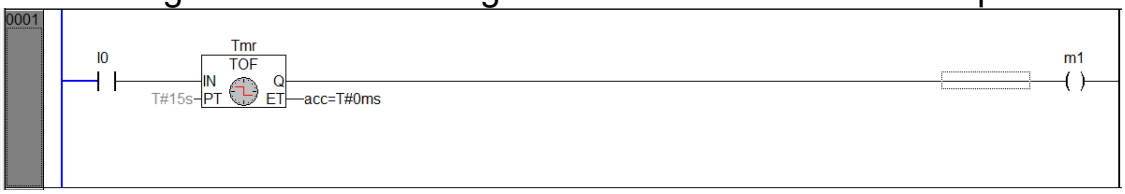

As I0 is pressed M0 is continue on because of latch and it has been given to the enable of TON so TON will start running. You can see the running value of timer in acc.



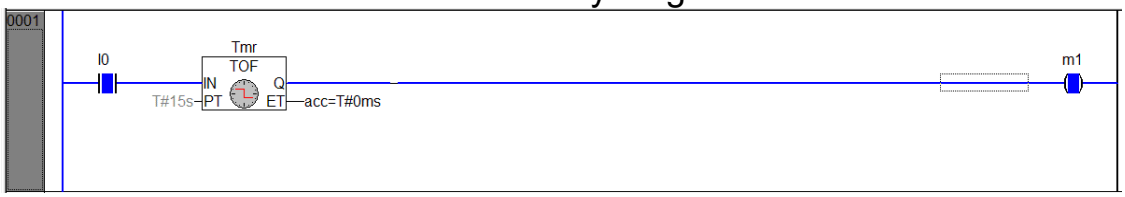

As accumulator is equal to PV, done bit M1 is on. As done bit in this software is a coil, you can take either output or flag address. It is better to use flag as done bit. Don't involve output everywhere. In this program M1 gets on after 15 seconds. Can't we say that M1 is 'after 15 second signal'? If after 15 seconds you have to switch on something, use M1 NO contact and if after 15 seconds you have to get something off then use M1 NC contact. As done bit M1 is on it is continue on. We can say the timer has got set. To reset this timer TON, break its enable path. You can put some NC between M0 and timer (2nd network) or you can break the latch. As enable is off TON will get

reset. Reset means done bit M0 will get off and Accumulator will become 0. This reset NC address will be a bit address and depends on the logic conditions.

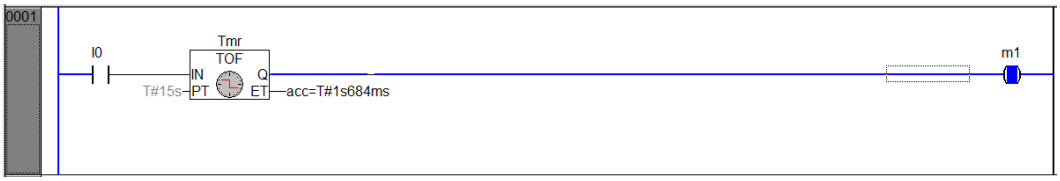
**TOF/ off delay timer:** this timer has also all four parameters. This timer's accumulator starts running when its enable is off. Its done bit gets high as enable is high and the done bit gets off when accumulator is equal to PV.



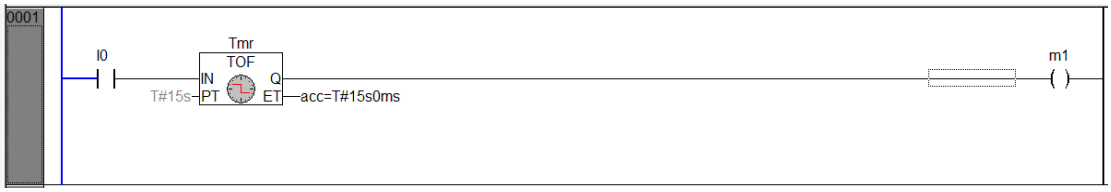

This is the initial condition. Everything is off. Accumulator is also zero.




This is the status when enable is high. You can see immediately done bit M1 gets high but accumulator is not running.



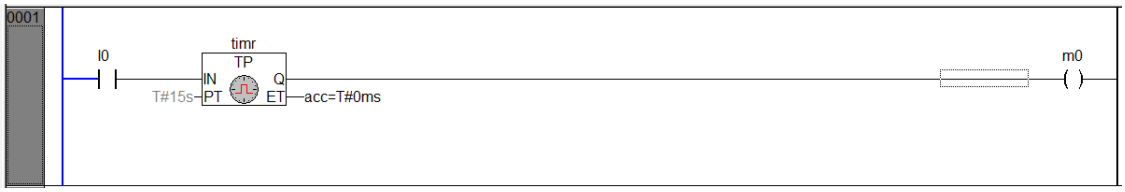

This is the status when enable is off and accumulator starts running. Done bit M1 is still on.



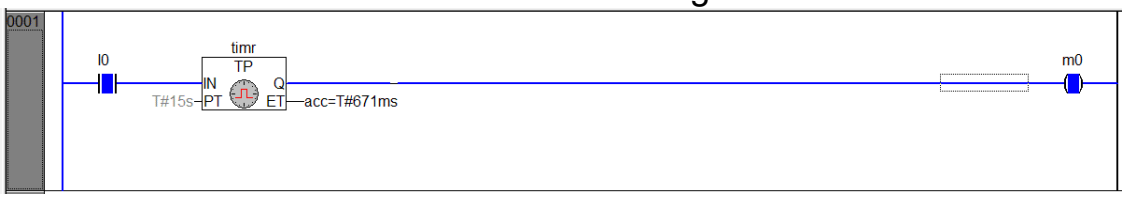

This is the condition when accumulator is equal to PV you can see and done bit M1 gets off. For TOF this is the set condition. To reset this timer, you need to make enable high.

**TP/ pulse timer:** Pulse time is very much similar to TOF. TOF accumulator runs when enable is off and TP accumulator runs when enable is high. It does not matter whether enable is a continue signal or a high pulse. Done bit

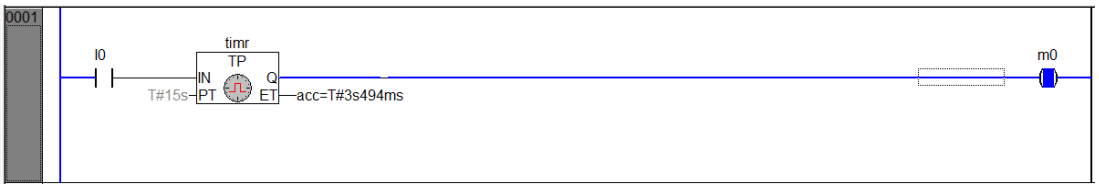
status is similar to TOF.




This is initial status where nothing is on. All zero.

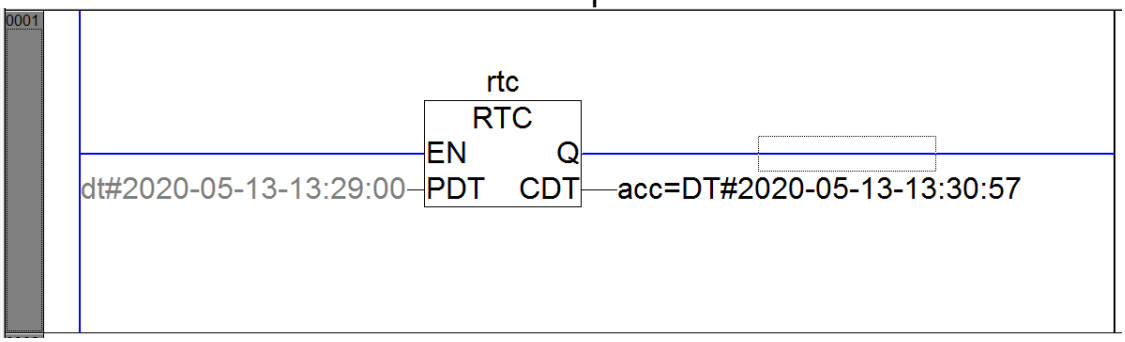



This is the status when enable is high. You can see immediately done bit M0 is on and accumulator starts running.




Once it starts running, it does not matter whether enable is on or off; accumulator will keep running when enable is off also. As accumulator is equal to PV, TP will get reset itself.

**RTC/ real time clock:** This timer contains year, month, day, hour, minute and second in its pre-set value.




You can generate signal by comparing your required year, month, day, hour, minute and second with the accumulator of this timer. It is used when you have to control outputs hourly/ daily/ weekly/ monthly yearly wise. For example, if you have to stop your machines daily during lunch time or tea break time automatically or you have to switch off lights and fans daily during lunch time and during off time of your office.

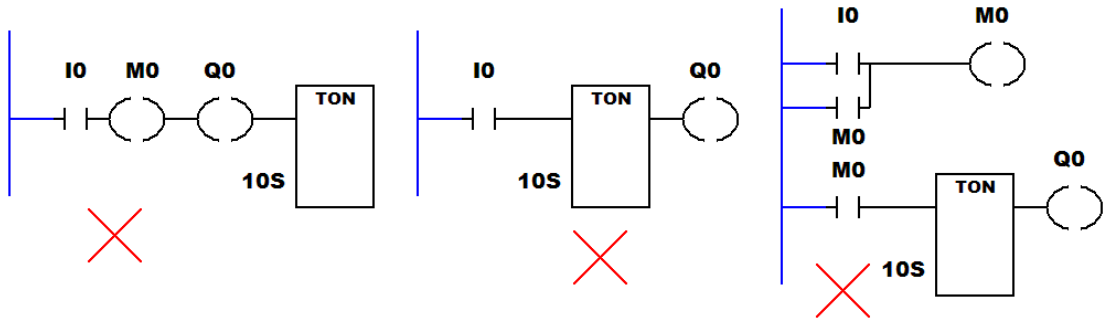
Now we shall make some programs based on timer. Let us go through the important points that should be remembered while making programs before we start making programs. These important points will help you to make program easily. There is no any rule for logic but if we follow important points then programming will be easy and you will be able to write logic any time.

**Important points:**

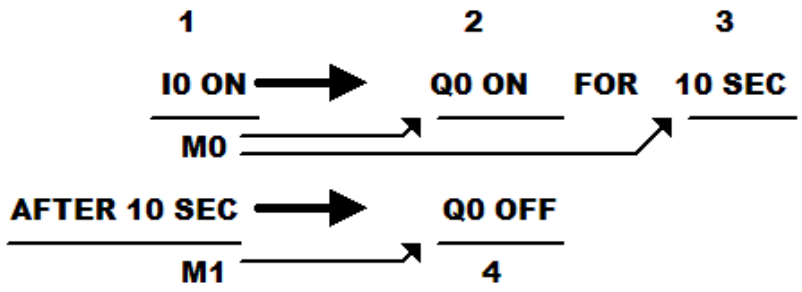
1> **To switch on use NO contact of the condition signal.**
2> **To switch off use NC contact of the condition signal.**
3> **When we have start signal, we should latch it with some flag.**
4> **When we have any stop signal, we should break the latch directly from it.**

We shall add some more points later.

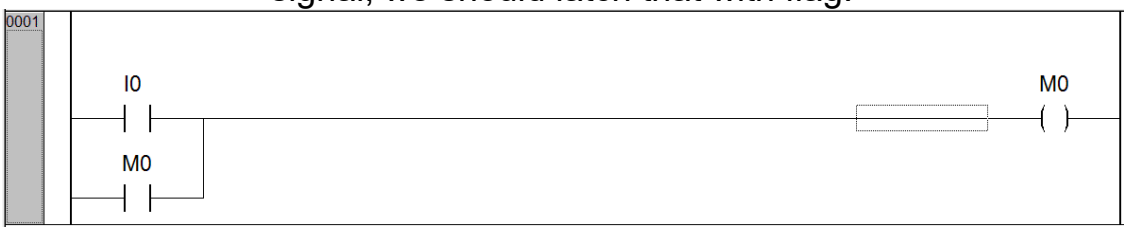
**EX11:** We have a push button I0 and a motor Q0. When we press I0, Q0 should get on for 10 seconds.



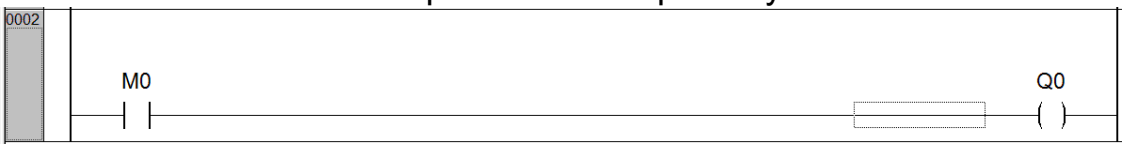

If you have made program like this then it is wrong. I think you understand what to do but you are not able to arrange the things in ladder as per your logic. Don't worry; we shall learn the easiest way again. Do not think about all conditions but start from first step and go further step wise. It is better first you write the conditions in steps. Initially I used to have flow chart of conditions but now I use only steps. Let it be small or big program, just write the steps first. For EX11 let us write the steps.



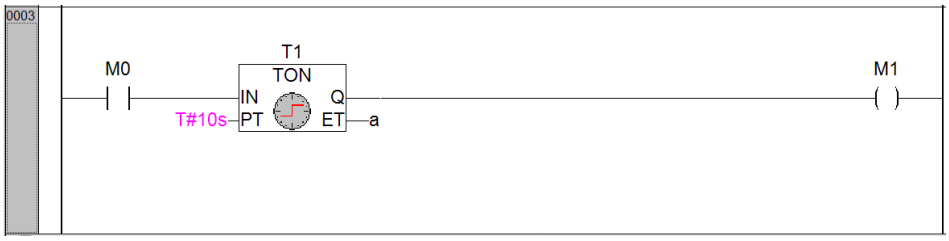
You can see we have four steps in this example. To write steps is an easy task. Now we shall make ladder step wise. Don't start thinking randomly. Go step wise and start from first step. When you think about 1st step, don't think about any other step. If you think about all steps together, your program will be wrong. Ok now first step is I0 start push button and when we have start signal, we should latch that with flag.



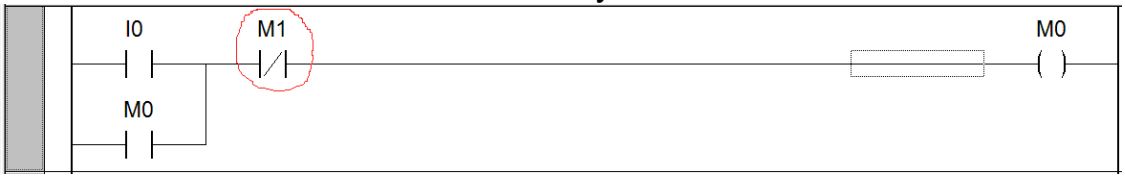

This is step 1. I0 is latched with M0. Now from I0 we get M0 so I have written M0 below I0 in first step. Now it is very much clear that what have we to do next. You can see in steps it is clearly shown M0 □ Q0 on. Please don't see 3rd step. Just see step 2 only.



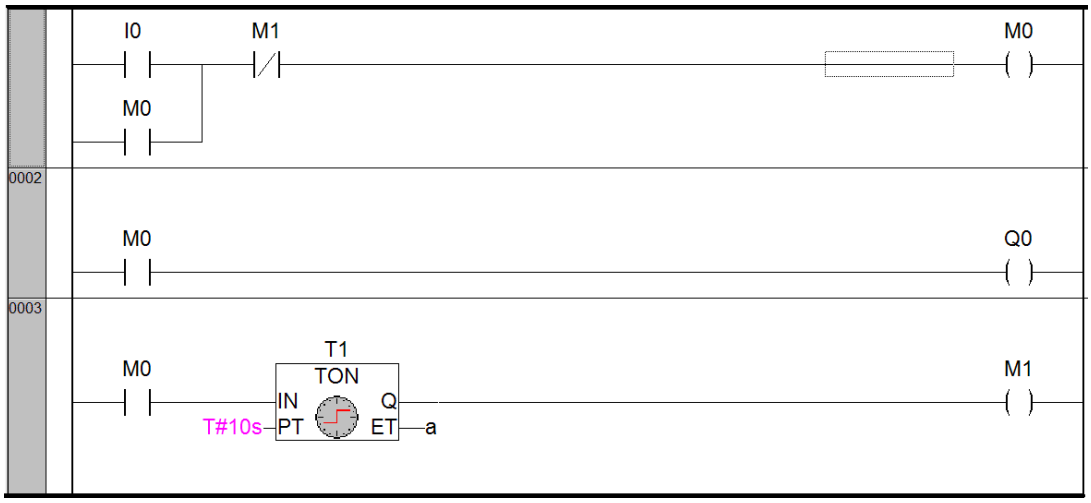

This is the ladder for step 2. Now again we go to step one and we see that from M0 we have to take a 10 second timer (M0 □ TON) for step 3.




This is the ladder for step 3. If we follow step, we do not need to worry that what should be the enable of timer. It is clearly shown in our steps. Now we get done bit M1 after 10 seconds. So, we write M1 in step. It is clear from step 4 that from M1 Q0 should get off (M1 □ Q0 off). So M1 is the stop signal and when we have stop signal, from that we should break latch directly.



This is the step 4. We go to the first network where we have a latch and we break it from M1 NC as per step 4.



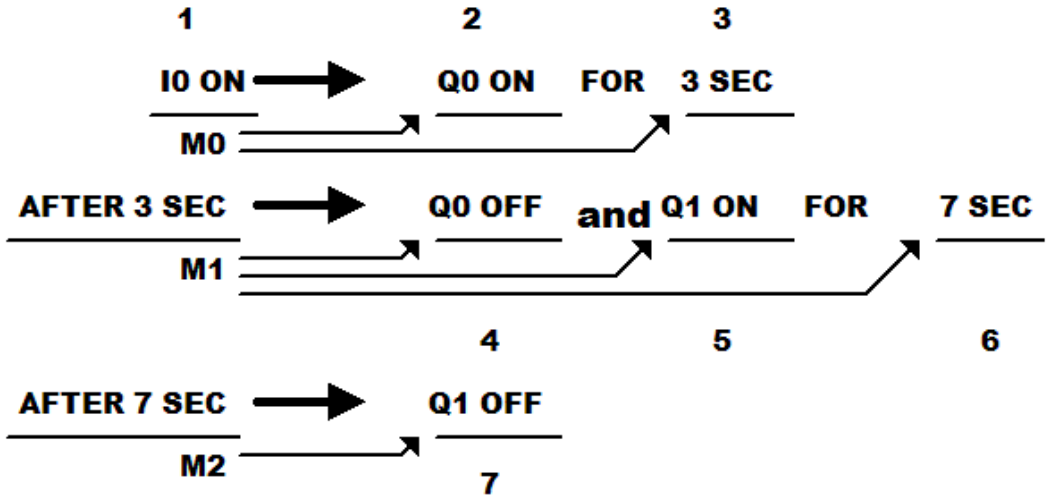

This is the final program for EX11.

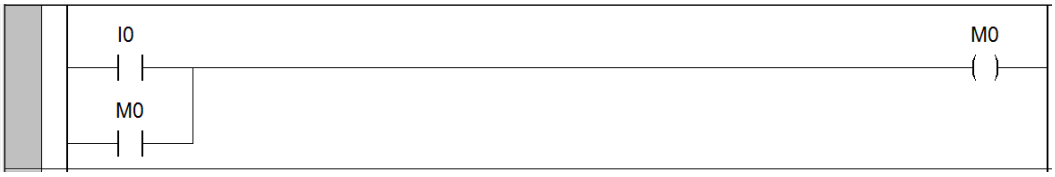
**Important points:**

1> **To switch on use NO contact of the condition signal.**
2> **To switch off use NC contact of the condition signal.**
3> **When we have start signal, we should latch it with some flag.**
4> **When we have any stop signal, we should break the latch directly from it.**
5> **When we get last signal in our program, we should break all latches from it. (It means that lastly after one cycle all coils must get off. If any coil will remain on after one cycle then machine will not run in next cycle.)**

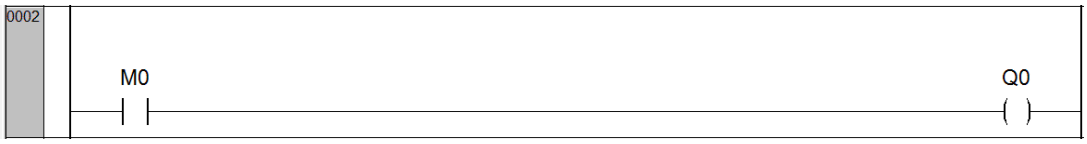
In last example11 M1 the last signal because from M1 we have not to get anything on further but we have to switch off so M1 has broken the latch. Breaking latch has made all coils off. Suppose after 10 seconds we have to get Q0 off and another output on. In that case M1 will not be on because we have to get another output on after 10 seconds. Now M1 will not break latch to get Q0 off but it will break Q0 ladder (network 2). Latch will always be broken by last signal or by emergency off/ stop signal.

**EX12: Now I shall write the steps and you write the ladders.**

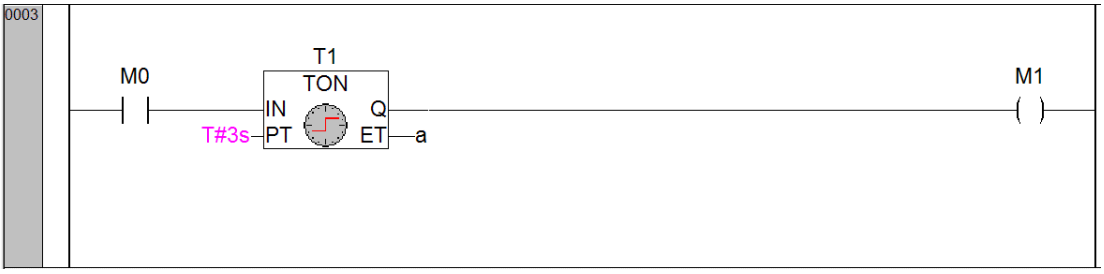


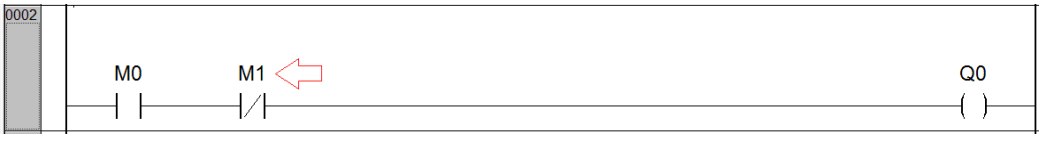

This is ladder for step 1. When we have start signal, we latch it with some flag so I0 is latched with M0.



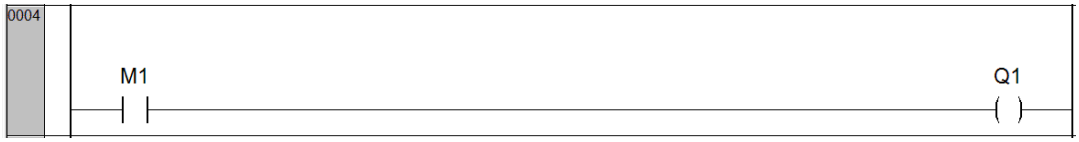

This is the ladder for step 2 i. e. M0 □ Q0 on.



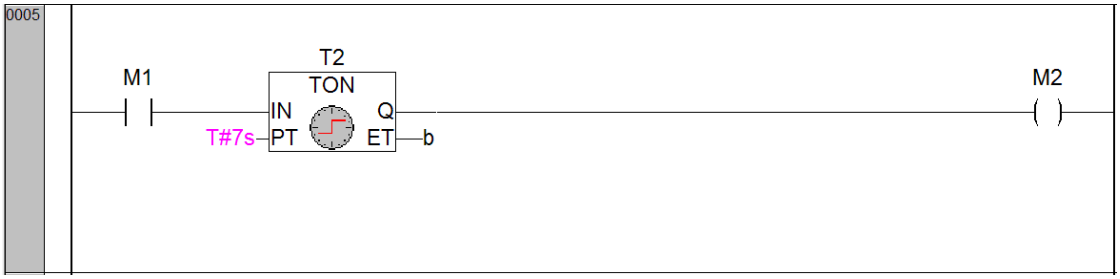

This is the ladder for step 3 i. e. M0 □ 3 Sec TON.




This is for step 4 i. e. M1 □ Q0 off. M1 is not last signal in this program because Q1 is to get on further from M1 so M1 will not break the latch but will break Q0 ladder.

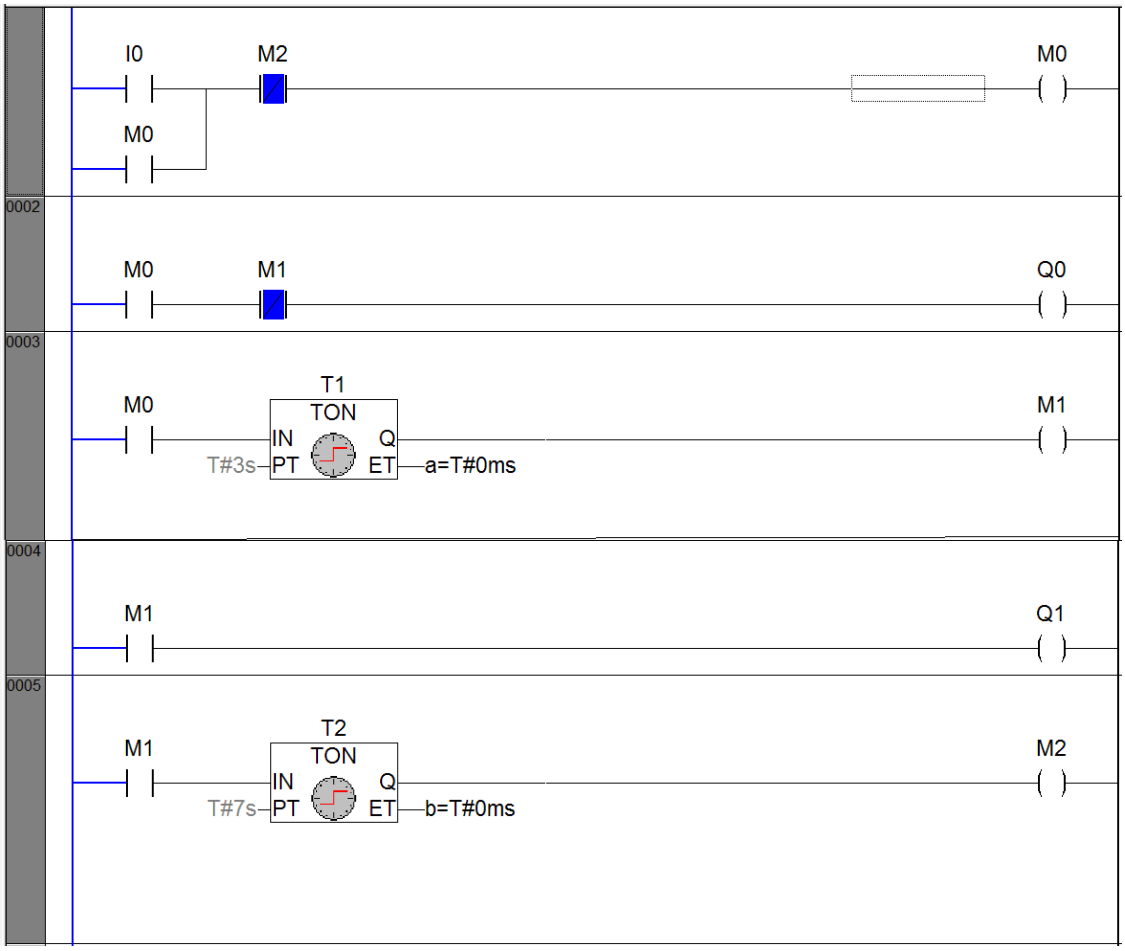



This is the ladder for step 5 i. e. M1 □ Q1 on.



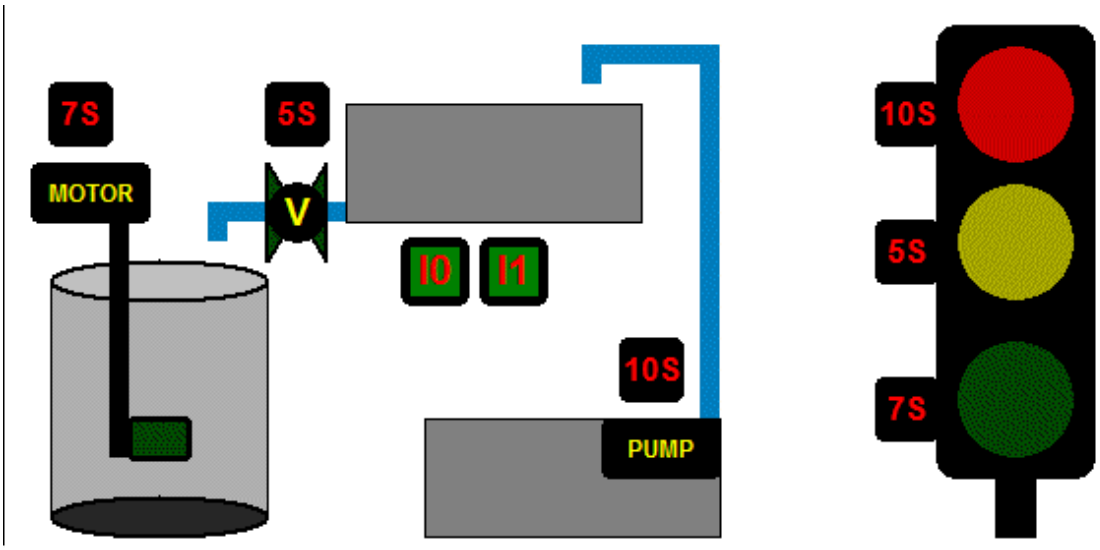

This is the ladder for step 6 i. e. M0 □ 7 Sec TON.


This is for step 7 i. e. M2 □ Q1 off. M2 is the last signal in this program so it will break the latch. As latch is broken, everything in program will get off so no need to break Q1 ladder.



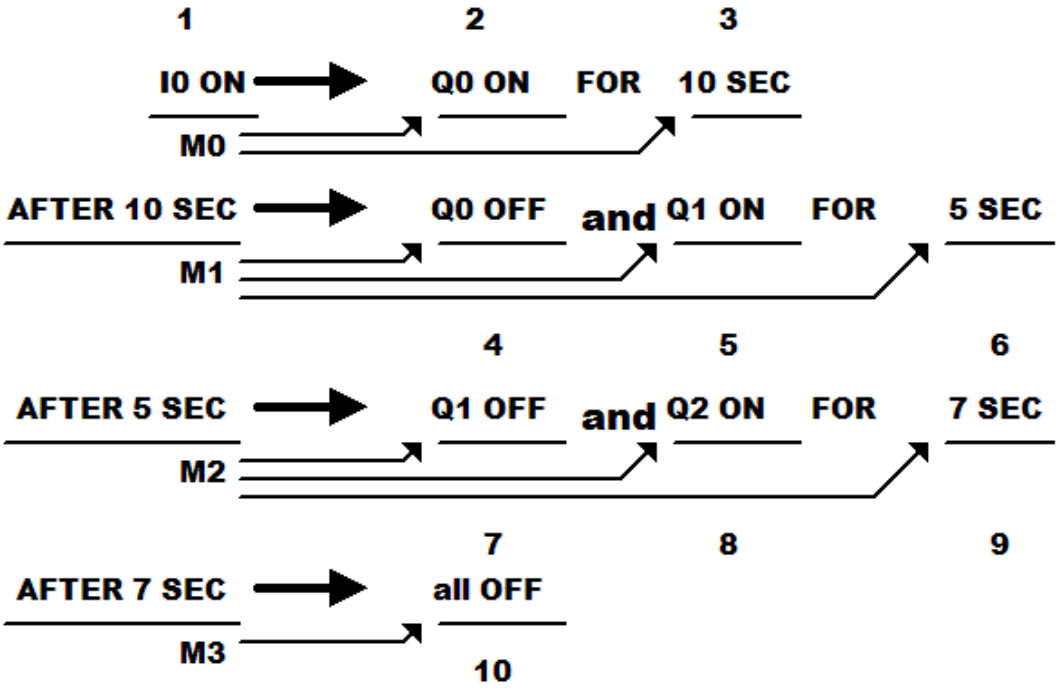

This is the final program for EX12.

**EX13:**

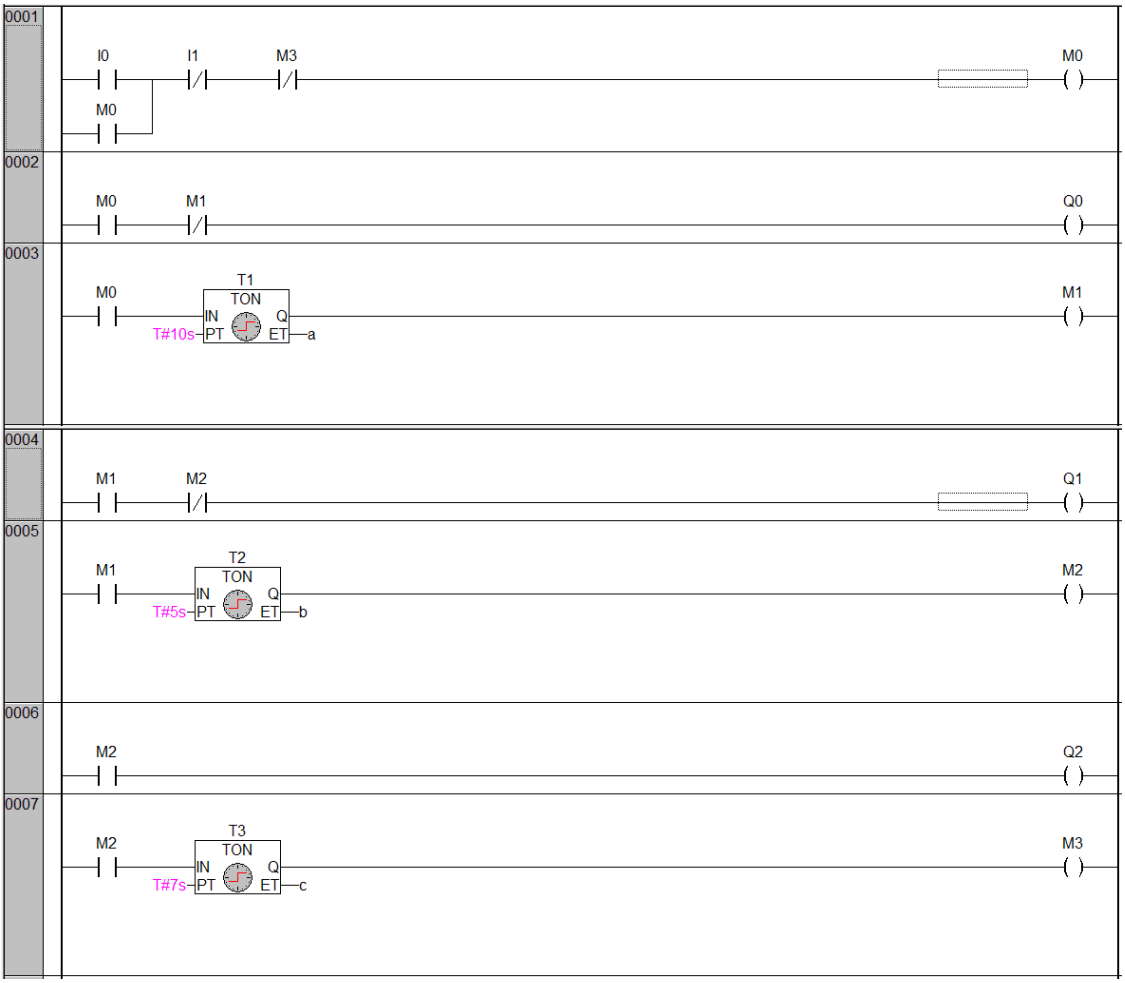



There are two systems here. Both will have same logic. There are 3 outputs Pump, valve and motor. There are two inputs I0 and I1 for start and stop. When I0 is pressed, pump Q0 should get on for 10 seconds. After 10 seconds Q0 off and valve Q1 should get on for 5 seconds. After 5 seconds Q1 off and motor Q2 should get on for 7 seconds. The same sequence is applicable on another system too. You can see another system also has 2 inputs and 3 outputs and the same sequence. So it does not matter what is the existence of physical output, program will be the same.

Let us write the steps:

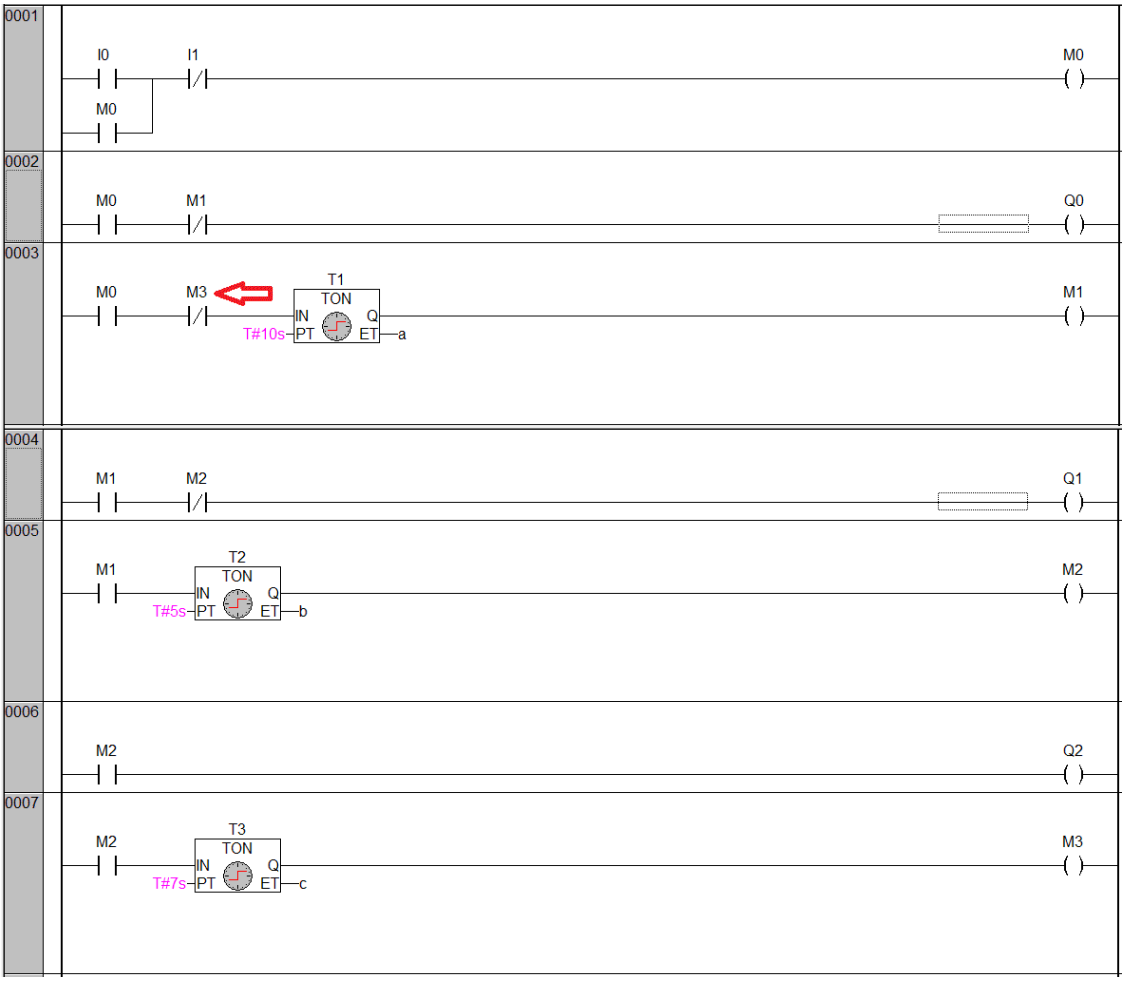



Follow the steps as you had followed for previous program. In previous example we had 2 outputs and, in this example, we have 3 outputs.




This is the program for EX13. M3 is the last signal so it breaks latch. I1 is the manual stop switch signal so it also breaks latch. M3 is automatic stop signal.

**EX14: Modify EX13 for cycle repeat process. In previous example we have process to get Q0 on, later Q1 on and then Q2 on. After one cycle whole process is stopped by breaking latch by M3. In this example the cycle should be repeated automatically. We have to press I0 push button once only to start the process.**




You can see in network 3 there is a small modification. **To repeat the process, take last signal and break first timer.** How will it repeat process? To repeat the process Q0 should get on again and all timers should restart as per sequence. When one cycle is finished, Q0 is off because of M1 (1st timer done bit) is on. As M3 breaks first timer, M1 is off. As M1 is off, two things will take place simultaneously; first thing is Q0 will get on and second thing is all timers reset. As M1 is off 2nd timer will get reset resulting M2 off and as M2 is off 3rd timer will get reset resulting M3 off. As M3 is off, M0 will reach to 1st timer enable and timer will start running and carrying the sequence forward. In repeat process M3 will not break latch.

**Important points:**

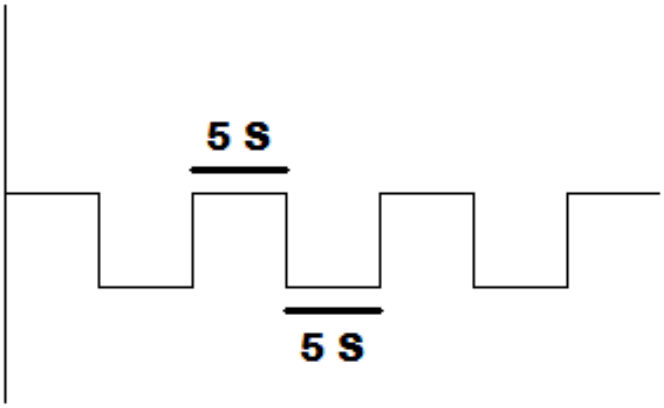
1> **To switch on use NO contact of the condition signal.**
2> **To switch off use NC contact of the condition signal.**
3> **When we have start signal, we should latch it with some flag.**
4> **When we have any stop signal, we should break the latch directly from it.**

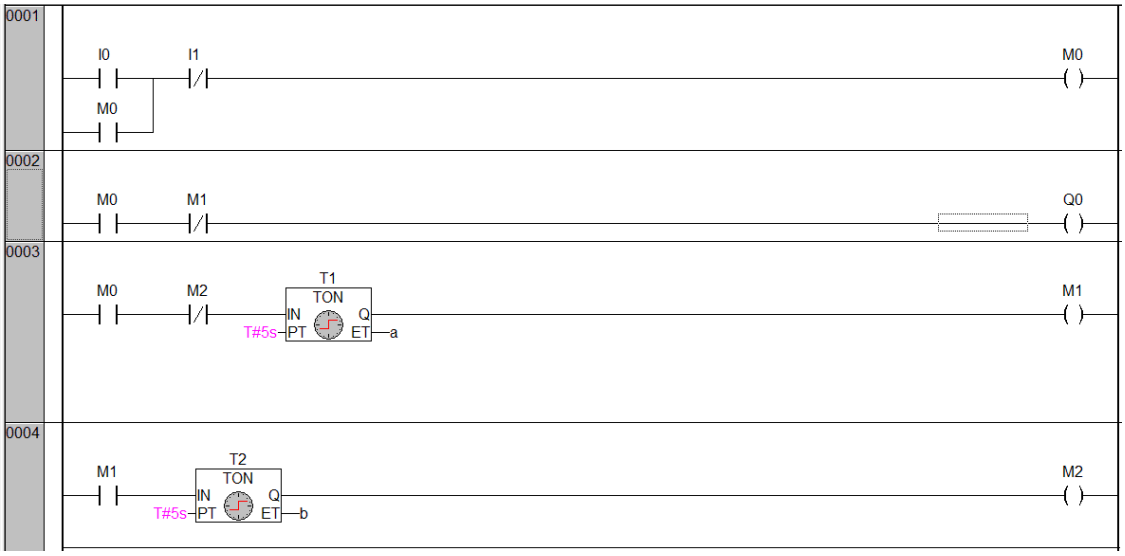
5> **When we get last signal in our program, we should break all latches from it. (It means that lastly after one cycle all coils must get off. If any coil will remain on after one cycle then machine will not run in next cycle.)**

6> **To repeat process in programs based on many timers, take last signal and break first timer.**

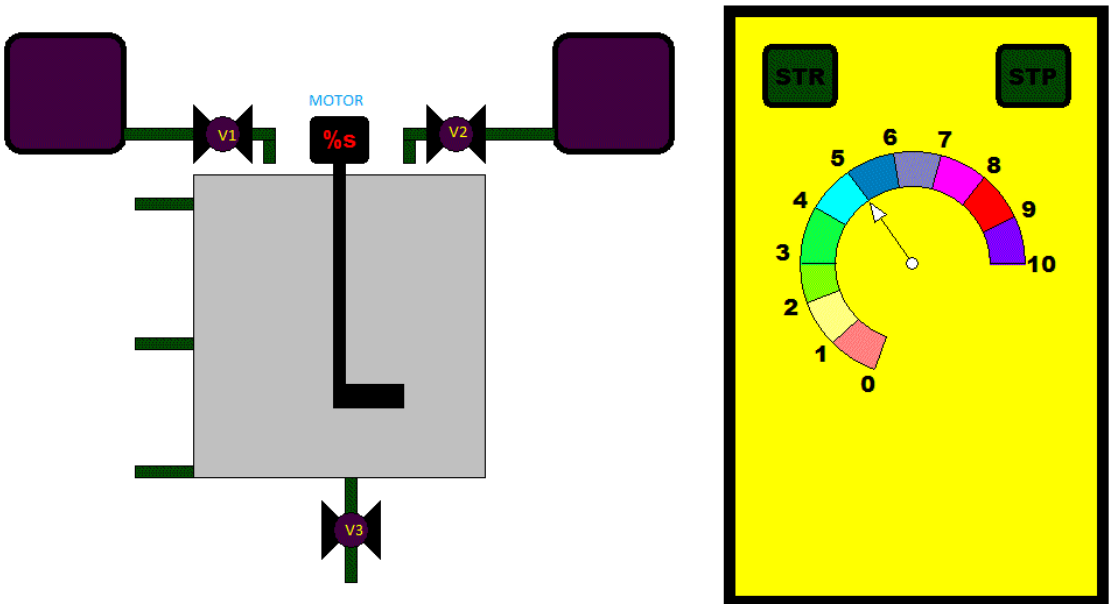
7> **To repeat process in programs based on many latches, take last signal and break all latches except start/ first latch.**

**EX15: square wave program. There are two push buttons I0 and I1 to start and stop the system. There is a motor Q0. When I0 is pressed, Q0 should get on for 5 seconds. After 5 seconds Q0 should get off for next 5 second and this process should repeat until stop button I1 is press.**

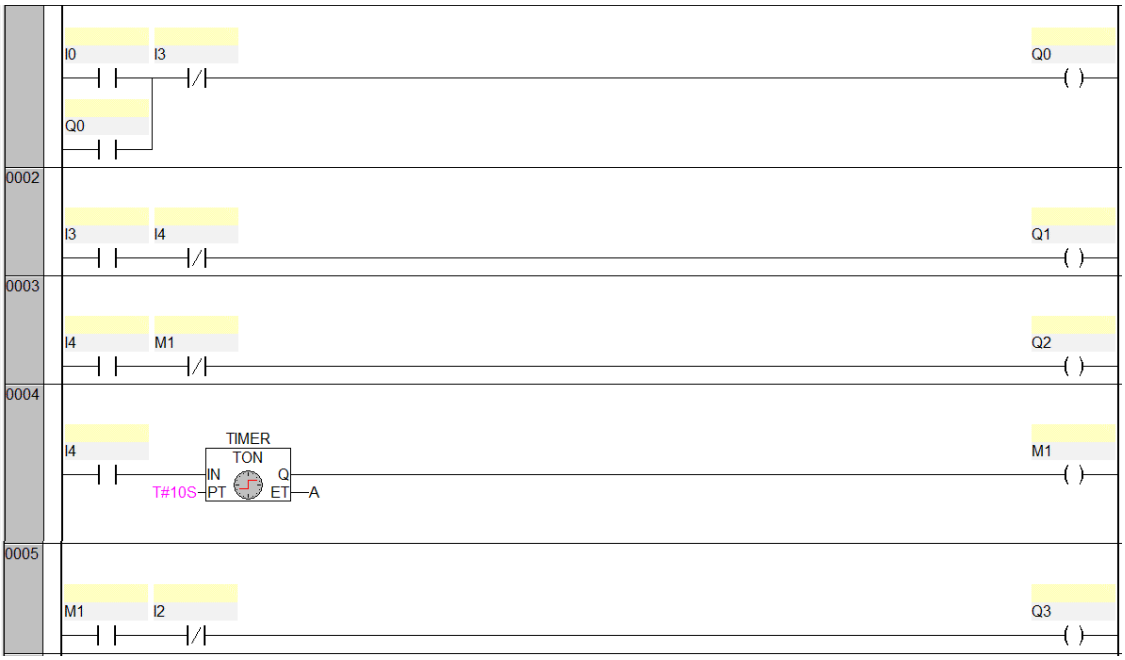


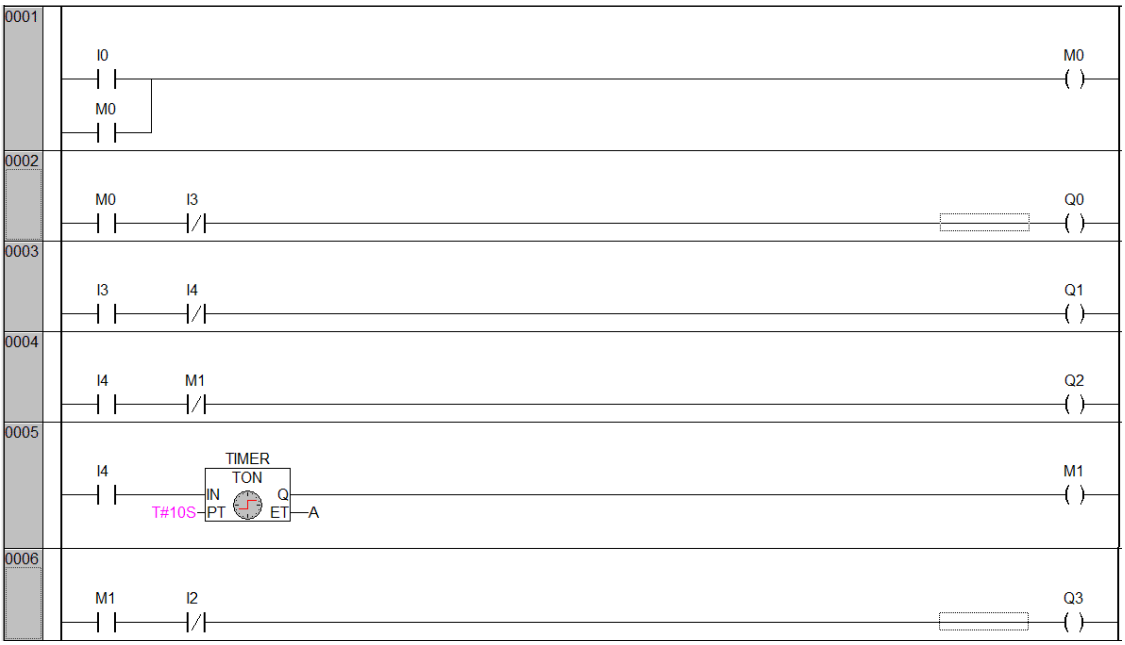
## EX16: Mixing application.



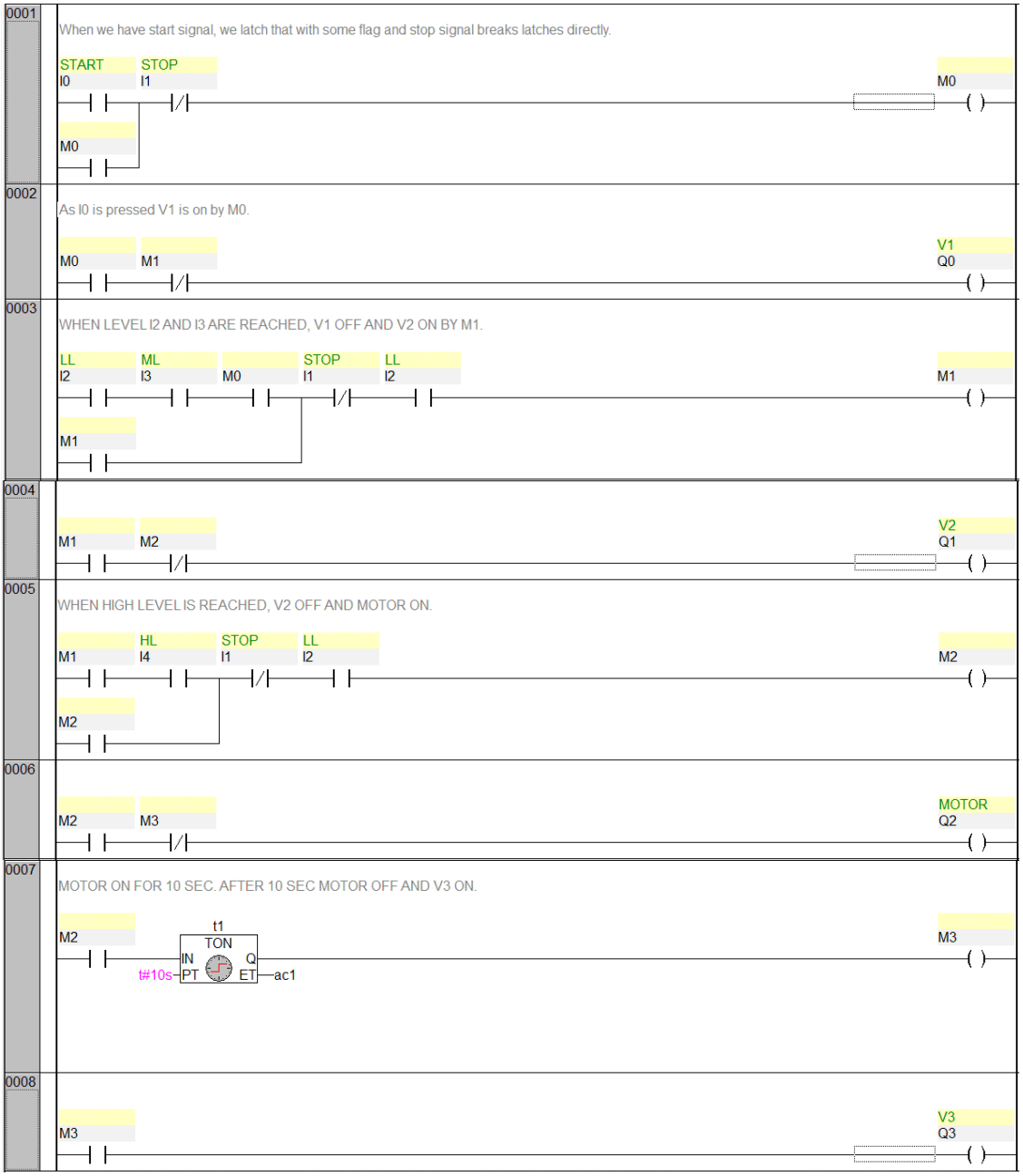

There are two push buttons I0 and I1 for start and stop (both push buttons are NO type/ green in physical existence). There are three level switches I2, I3 and I4 for low level, middle level and high level respectively. There are 3 valves; V1 for material 1, V2 for material 2 and V3 for drain. There is a motor for mixing two materials in the mixing tank. When I0 is pressed, Q0/ V1 will get on resulting material 1 inlet to mixing tank. As material 1 comes in mixing tank, low level sensor I2 will get on and further middle level sensor I3 will be on. As I3 is on, Q0/ V1 should get off and Q1/ V2 should get on. As V2 is on, material 2 will come in mixing tank resulting increase in level and reaches to I4 sensor. As I4 is on, Q1/ V2 will get off and motor Q2 will get on for 10 seconds. After 10 seconds, motor gets off and drain V3 gets on. As drain is on, mixing tank will start getting empty. Slowly level will go down resulting I4 off, I3 and I2 sensors off one by one. As I2 is off repeat the process. I1 is manual stop.




**This is wrong.**



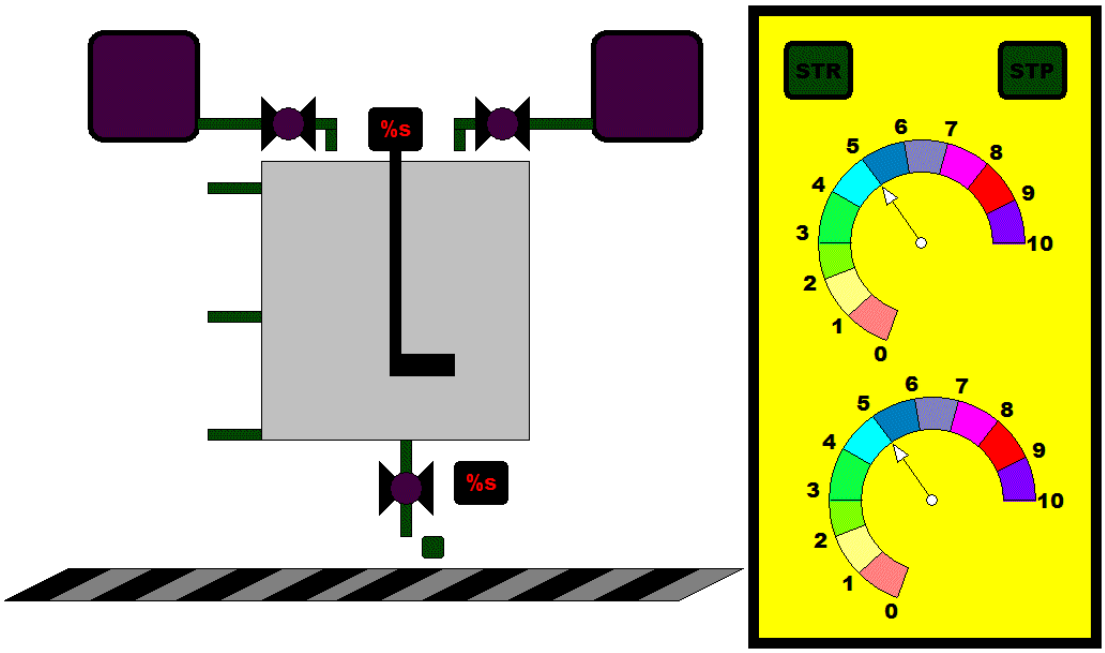

**This is also wrong.**

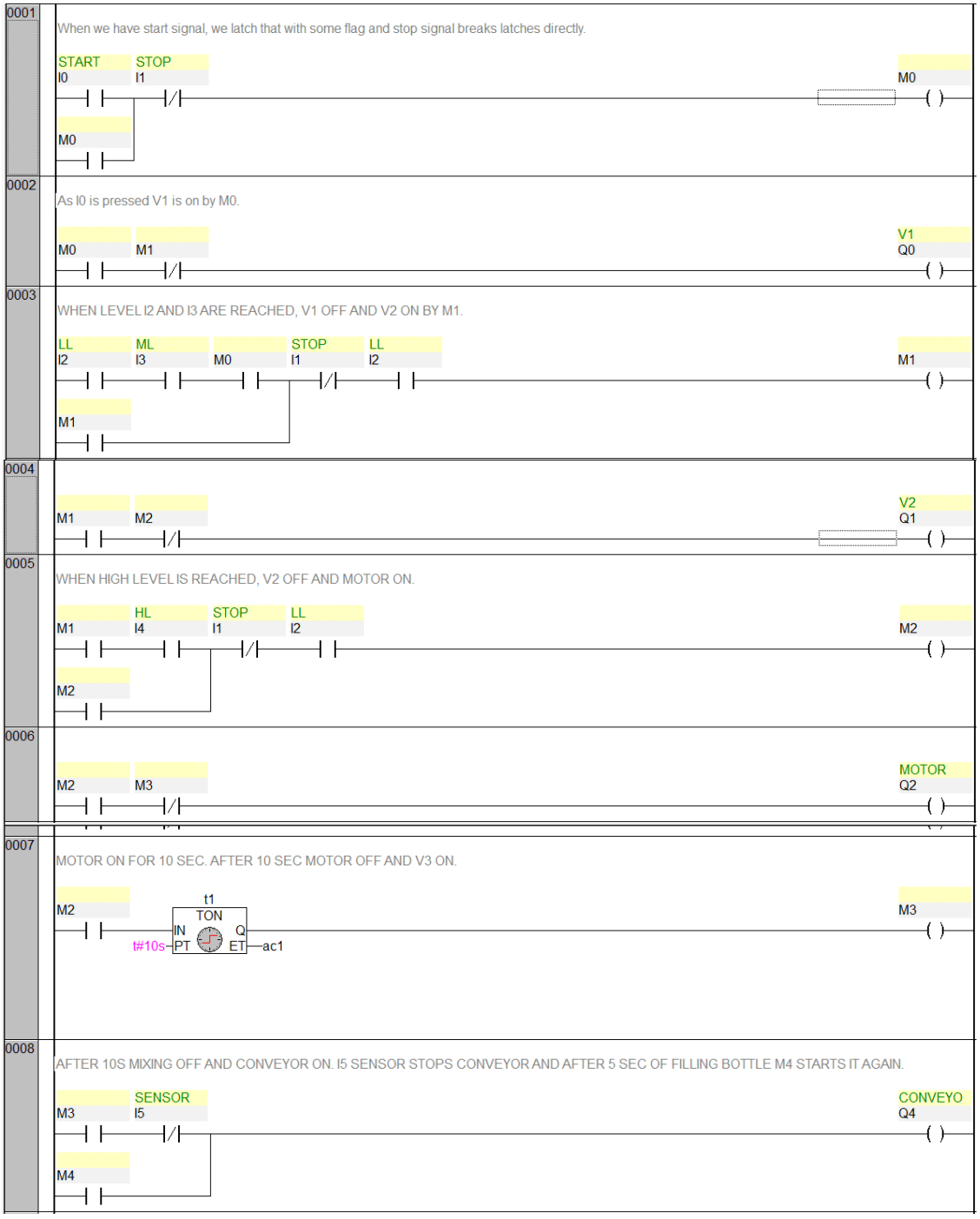


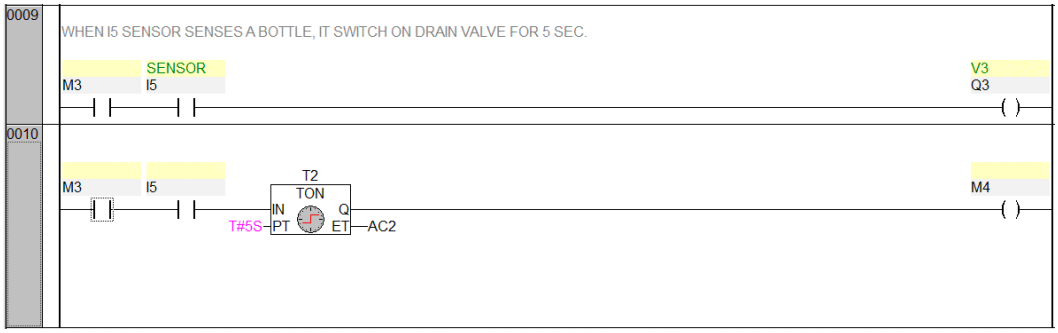

**This is correct.** When I2 is off, all latches should be broken except start latch by it so I2 NO breaks latches.

**EX17: Mixing and filling application.** It is the extension of previous example. There are two push buttons I0 and I1 for start and stop (both push buttons are NO type/ green in physical existence). There are three level switches I2, I3, I4 and I5 for low level, middle level high level and bottle detector sensor respectively. There are 3 valves; V1/ Q0 for material 1, V2/ Q1 for material 2 and V3/ Q3 for drain. There is a motor for mixing two materials in the mixing tank and one conveyor Q4. When I0 is pressed, Q0/

V1 will get on resulting material 1 inlet to mixing tank. As material 1 comes in mixing tank, low level sensor I2 will get on and further middle level sensor I3 will be on. As I3 is on, Q0/ V1 should get off and Q1/ V2 should get on. As V2 is on, material 2 will come in mixing tank resulting increase in level and reaches to I4 sensor. As I4 is on, Q1/ V2 will get off and motor Q2 will get on for 10 seconds. After 10 seconds, motor gets off and conveyor Q4 gets on. Bottle to be filled will move on it. As bottle will be detected by I5 sensor, conveyor Q4 will stop and drain V3/ Q3 will get on for 5 seconds. After 5 seconds again conveyor will be on and drain will get off. Whenever I5 detects bottle, the same cycle of conveyor and drain will repeat. As drain will get on many times, mixing tank will start getting empty. Slowly level will go down resulting I4 off, I3 and I2 sensors off one by one. As I2 is off repeat the process. I1 is manual stop.




0001
When we have start signal, we latch that with some flag and stop signal breaks latches directly.
START
I0
STOP
I1
M0
M0
M0
0002
As I0 is pressed V1 is on by M0.
M0
M1
V1
Q0
0003
WHEN LEVEL I2 AND I3 ARE REACHED, V1 OFF AND V2 ON BY M1.
LL
I2
ML
I3
M0
STOP
I1
LL
I2
M1
M1
0004
M1
M2
V2
Q1
0005
WHEN HIGH LEVEL IS REACHED, V2 OFF AND MOTOR ON.
M1
HL
I4
STOP
I1
LL
I2
M2
M2
0006
M2
M3
MOTOR
Q2
0007
MOTOR ON FOR 10 SEC. AFTER 10 SEC MOTOR OFF AND V3 ON.
M2
t1
TON
IN
PT
t#10s
Q
ET
ac1
M3
0008
AFTER 10S MIXING OFF AND CONVEYOR ON. I5 SENSOR STOPS CONVEYOR AND AFTER 5 SEC OF FILLING BOTTLE M4 STARTS IT AGAIN.
M3
SENSOR
I5
CONVEYO
Q4
M4




**EX18:**

**I0 on □ Q0 on.**

**I1 on □ Q0 off after 5 seconds of I1 press.**


**EX19:**

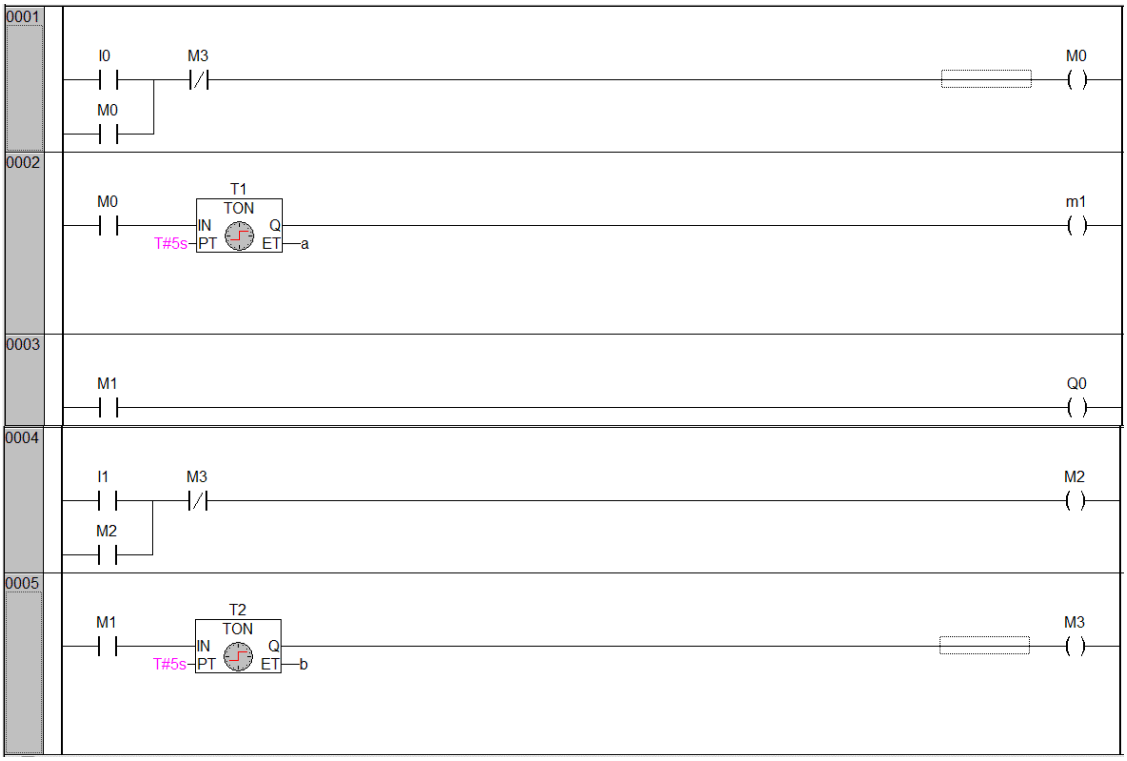
**I0 on □ Q0 on after 5 seconds.**

**I1 on □ Q0 off.**


**EX20:**

**I0 on □ Q0 on after 5 seconds.**

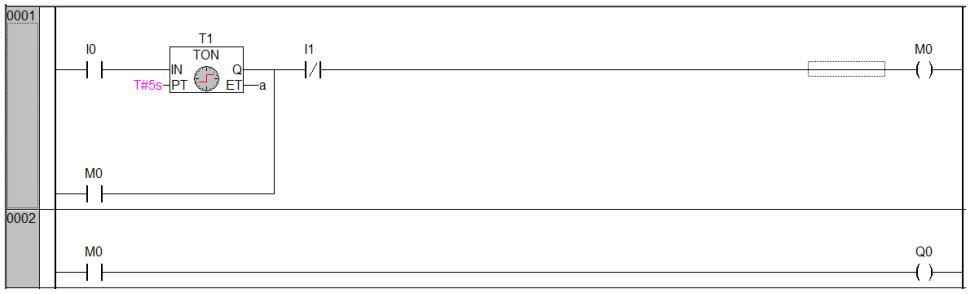
**I1 on □ Q0 off after 5 Sec of I1 press.**




**EX21:**

**I0 pressed for 5 seconds □ Q0 on**

**I1 on □ Q0 off**




**EX22:**

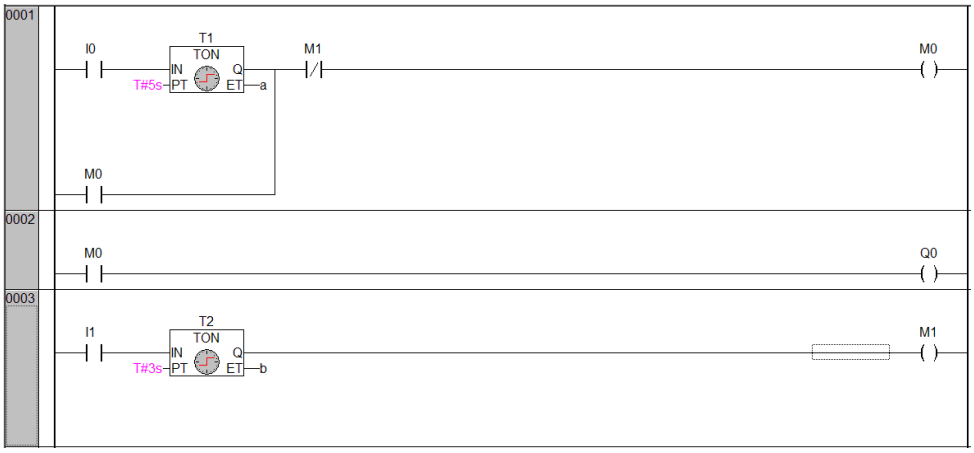
**I0 on □ Q0 on**

**I1 pressed for 5 seconds □ Q0 off**


## EX23:

## I0 pressed for 5 seconds  Q0 on

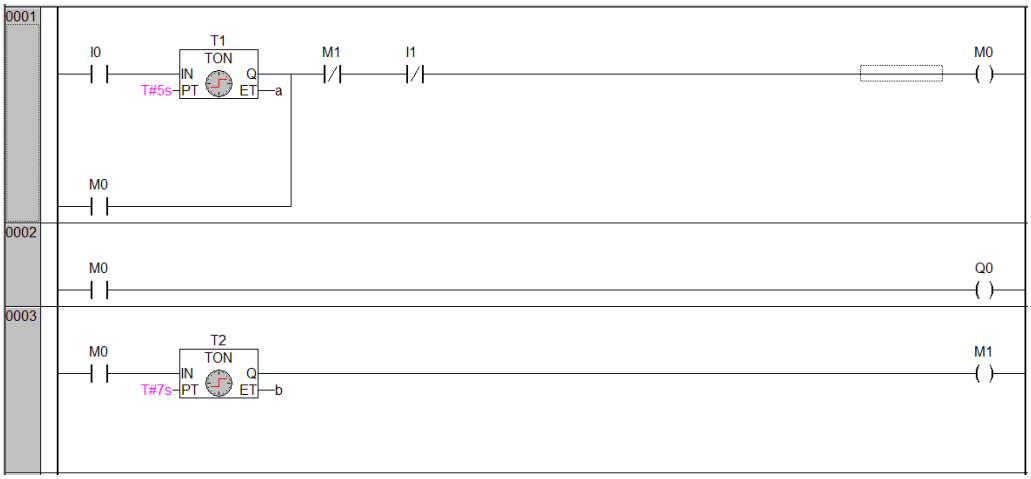
## I1 pressed for 3 seconds  Q0 off



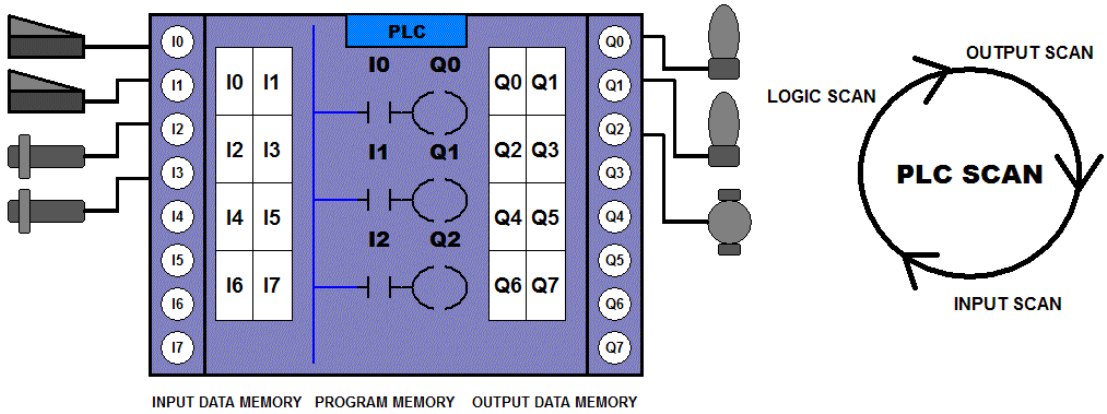

## EX24:

## I0 pressed for 5 seconds  Q0 on for 7 sec

## I1 on  Q0 off any time



## PLC SCAN/ PLC EXECUTION:




How does any physical output of any PLC get on/ off? What are the steps followed by PLC inside it? Status of any physical output of PLC gets changed because of PLC SCAN process. PLC Scan or PLC Execution takes place in three steps; Input scan, Logic scan and Output scan. If any step among these 3 steps will not execute then physical output status will not change. There are 3 main conditions when PLC scan takes place;

1> whenever PLC gets powered on,
2> Whenever there is a change in physical input status (on or off) and
3> Whenever there is change in logic (any new signal such as done bit gets on/ off).

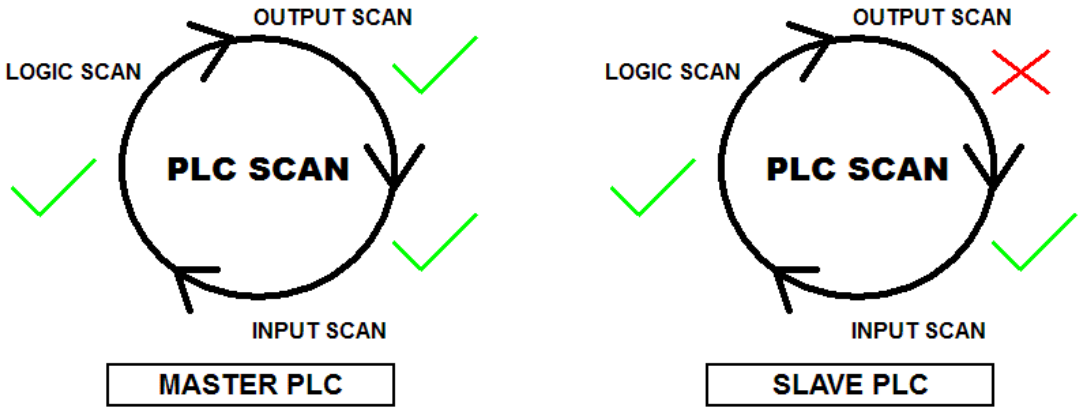
Let us understand SCAN in detail. You can see a PLC in above picture. Outside PLC some input outputs are connected to its I/ O terminals. Inside PLC we have program memory where program is stored sent by us and there is data memory too where all addresses are pre located. I have shown you some input addresses, program and some output addressed on PLC memory location. Whenever any condition for scan is satisfied, scan starts. As scan starts, it is input scan first in which all maximum number of input addresses in that PLC are checked/ verified. According to all input status update PLC starts logic scan and checks the status of all coils. Logic scan takes place left to write and top to bottom. It means it starts from first contact, checks each ladder one by one and reaches to last ladder/ last coil. As logic scan reaches to last coil, logic scan gets over and does not go back

to check again. There is no reverse scan. According to the logic scan update, output scan starts. It updates the status of all output addressed in output data memory. After this only any physical output status gets changed.

Time taken to complete one scan cycle is **one scan time** and this PLC scan time is not fixed. It depends on the length of program. Generally, for small program it is few milliseconds. Now a day in new PLCs scan time is in microseconds. If any step of scan cycle is interrupted, status of physical output will not change.

Scan of ladders can be controlled by using **Jump** and **Master control** instructions. If Jump instruction is applied to few ladders then those ladders' scan will be skipped if Jump instruction is high. If Master control instruction is applied to few ladders then those ladders' scan will be start if Master control instruction is high. We shall learn these instructions further.

Scan plays an important role in **redundant/ hot standby** systems too. In process industries where machine operation should not be interrupted in any condition, there redundant system is used to avoid interruption in case of controller failure by any mean. In redundant or hot standby system two PLCs of same brand/ same model are connected with each other. There are some redundant PLC models also available. We shall have 2 CPUs, 2 Power modules, 2 I/O modules, 2 communication channels etc. It means in redundancy we have every part of PLC double. It is hardware and software redundancy available. In hardware redundancy if any hardware module will face problem then automatically it will switch to another module. In software redundancy you need to generate conditions and you need to write logic to switch to another module.



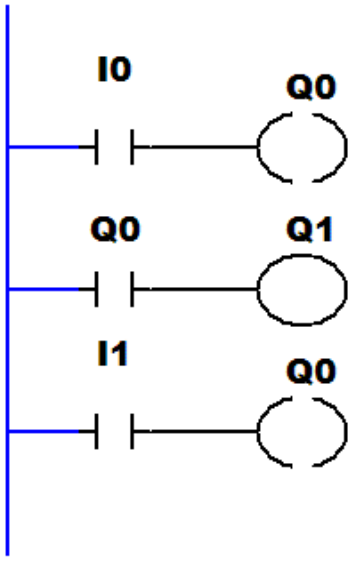

In redundant system as you know we have two PLCs so one is called master and another is slave. Both master and slave PLCs are connected to all input/ outputs of machine. When system is powered on or is being

executed then both PLCs/ CPUs will start scan at the same time. So input scan for both, logic scan for both but output scan for master PLC only, not for slave PLC. As output scan is in master PLC, only master's output terminals will be active and will operate physical outputs of the machine. Slave output terminals will not be active as output scan was note done by its CPU. During scan/ execution if any problem occurs in master PLC then slave PLC will start executing output scan and physical outputs of the machine will be operated by slave PLC. The new controller DCS (Distributed Control System) is totally based on redundancy.

Maximum scan time set for any controller is **Watch-Dog-Timer**. If your program scan time exceeds watch dog timer then PLC will get reset/ will not execute. It also reacts on any kind of software/ hardware malfunctioning and generate indication on CPU. Watch dog timer completely monitors PLC execution. Watch dog timer set for any PLC is 150ms to 200ms.

Now let us see scan importance by one example. I had mentioned initially that do not repeat coil address. It is not a rule but it can give you different result than expected.

**EX25:**



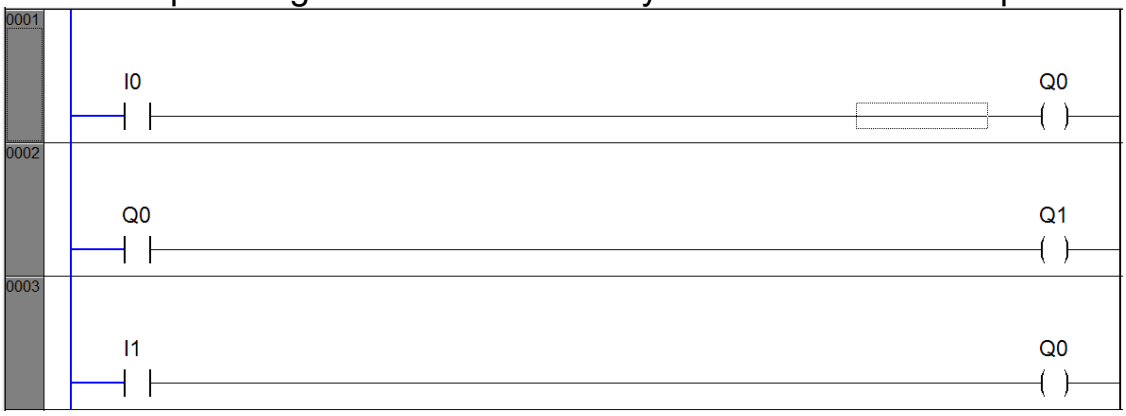

| I0 | I1 | Q0 | Q1 |
|---|---|---|---|
| 1 | 0 | | |
| 0 | 1 | | |
| 1 | 1 | | |

In this example I have repeated Q0 address in two coils. This is also logic but what would be the result of Q0 and Q1 as per conditions given for I0 and I1, you need to answer.

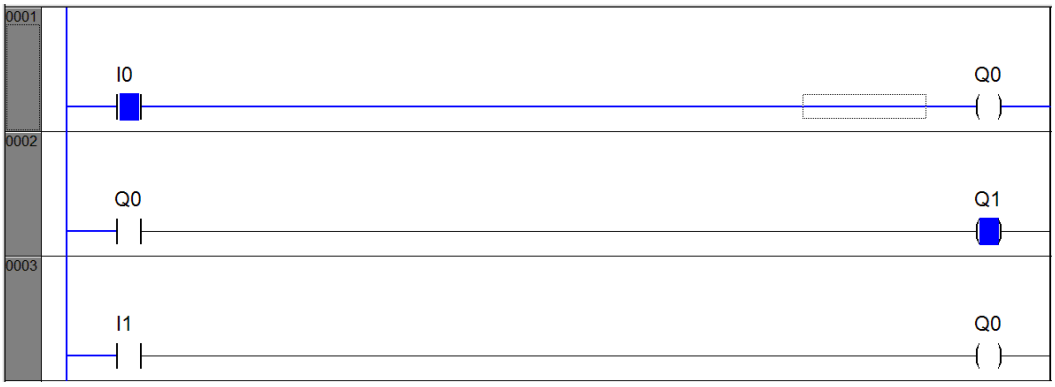
| I0 | I1 | Q0 | Q1 |
|---|---|---|---|
| 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

| I0 | I1 | Q0 | Q1 |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

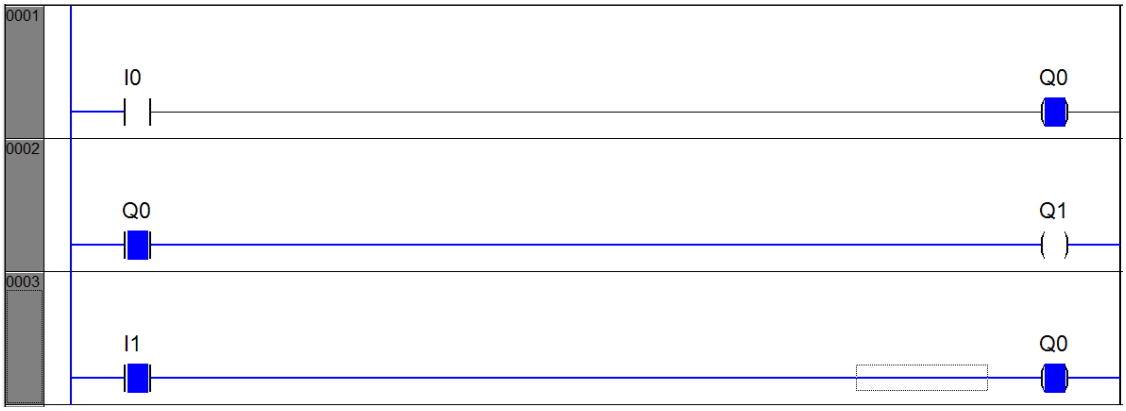
How? How did we get this result? It is because of scan. In logic scan I had mentioned that logic scan takes place from left to right and top to bottom. As last scan will reach to last coil, scan will be over and it will not go back to check or update again. Last status of any coil address will be updated.



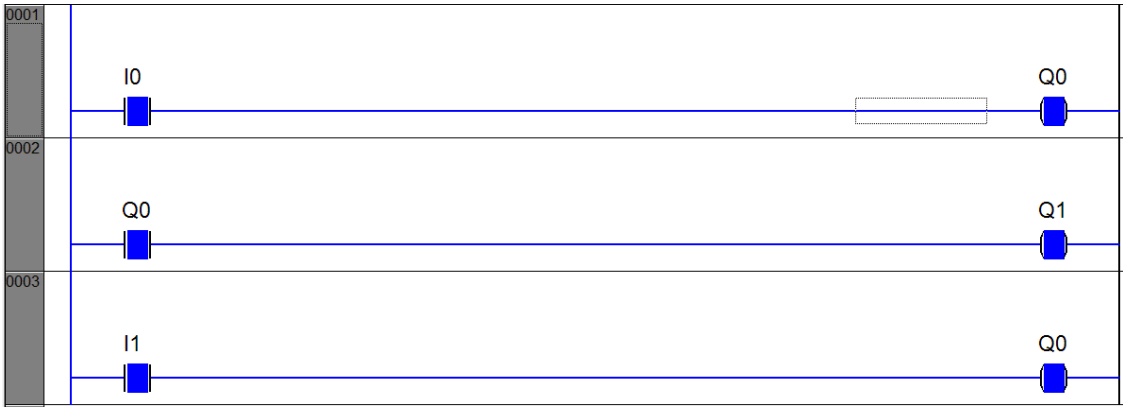

This is the status when I0 and I1 both are off (initial stage).



This is the status when I0 is on and I1 is off. As logic scan starts, it checks first ladder and finds Q0 on because of I0 on. In second ladder Q1 is on because of Q0 on. In third ladder scan finds Q0 off because of I1 off and as this is the last ladder, scan is over and will not go back to check again. So final status it finds Q0 off and Q1 on.



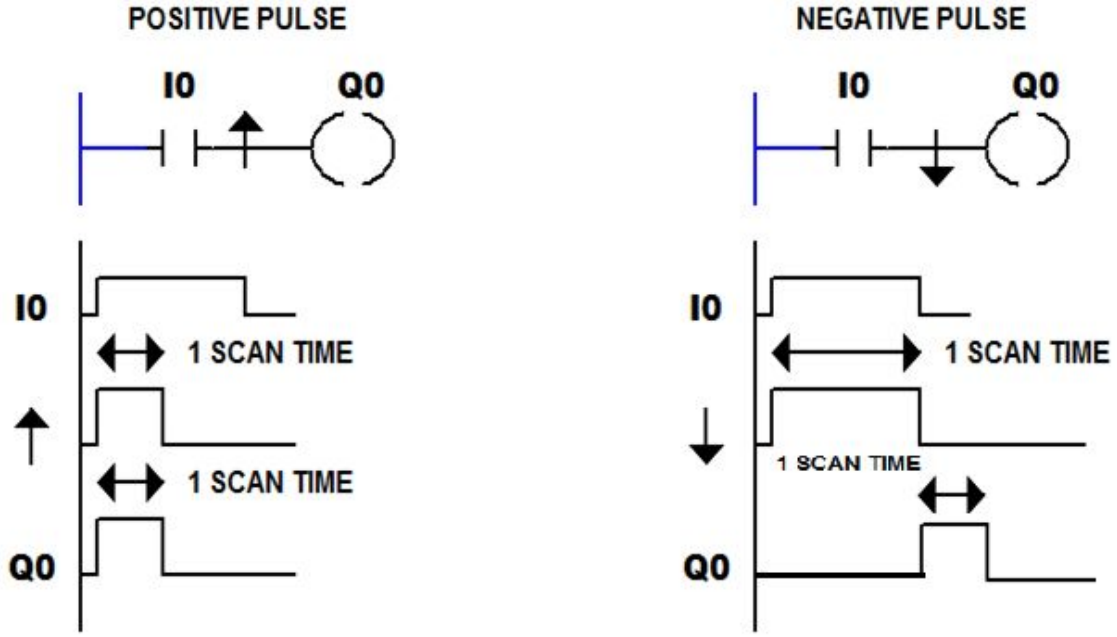

This is the status when I0 is off and I1 is on. As logic scan starts, it checks first ladder and finds Q0 off because of I0 off. In second ladder Q1 is off because of Q0 off. In third ladder scan finds Q0 on because of I1 on and as this is the last ladder, scan is over and will not go back to check again. So final status it finds Q0 on and Q1 off.




This is the status when I0 and I1 both are on. As logic scan starts, it checks first ladder and finds Q0 on because of I0 on. In second ladder Q1 is off because of Q0 on. In third ladder scan finds Q0 on because of I1 on and as this is the last ladder, scan is over and will not go back to check again. So final status it finds Q0 on and Q1 on. For now, please do not be worried about scan while making program, just follow the sequence steps of examples.

**PULSE: In very simple, Pulse is a short duration supply.** Duration means time, a short time supply. Suppose there is a switch and there is a

lamp. When switch is on, lamp is on and when switch is off the lamp is also off. Now let us a pulse with the switch and consider the duration of lamp is 10 seconds. Now switch is on, lamp will get on but for 10 seconds only. After 10 seconds lamp will get off automatically. It does not matter whether switch is on or off. We can say that pulse allows supply to pass through it for its duration. After its duration it stops the supply to pass through it. In PLC, pulse duration is not fixed but is equal to program scan time. Pulse is on during program scan only. Pulse is of two types; Positive pulse and Negative pulse. In different software we have different name like rising trigger/ falling trigger, one sort rise. One sort false etc. we shall call it positive pulse and negative pulse. Let us understand the difference by graph.

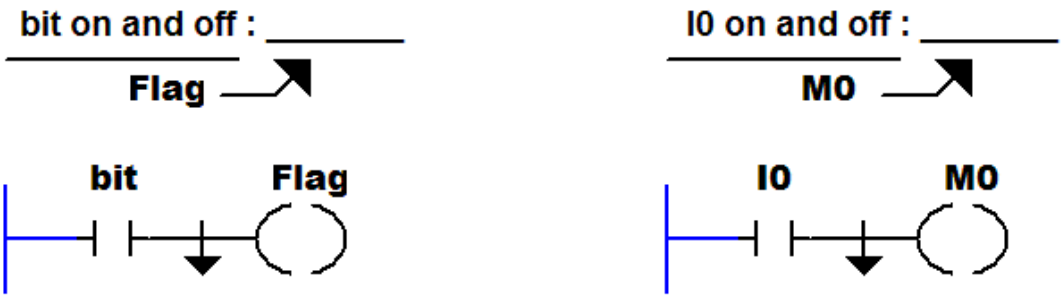



**POSITIVE PULSE:** You can see the graph of positive pulse. As I0 switch is pressed/ on, immediately positive pulse is active. Pulse is on and off as well. Output follows the pulse and it also gets on and off as I0 is pressed. You can see the switch is continuously pressed but pulse is not on again. When switch is off and on again then only pulse & output will get on again. Positive pulse will be active when switch is on.

**NEGATIVE PULSE:** You can see the graph of negative pulse. As I0 switch is pressed/ on, nothing happens to pulse or output. When I0 is off then only negative pulse is active and gets on and off. Output follows the pulse and it also gets on and off as I0 is off. When switch is off again then only pulse & output will get on again. Negative pulse will be active when switch is off.

Let it be positive pulse or negative pulse, both the pulses are on and off and both are of 1 scan time duration. The only difference is; positive pulse is

active when input is on and negative pulse is active when input is off. Negative pulse has more applications. We shall start with negative pulse. There is a push button and a motor. Press the push button/ switch is on but motor should not start. Keep the push button pressed; now also motor will not start. Leave the switch/ switch is off, now the motor starts. Here you see that motor starts when switch is off. In such case negative pulse will be used. We can say when a bit signal is on and off/ switch is on and off, in such case we use negative pulse. To use negative pulse in your logic/ ladder, better you remember this syntax.



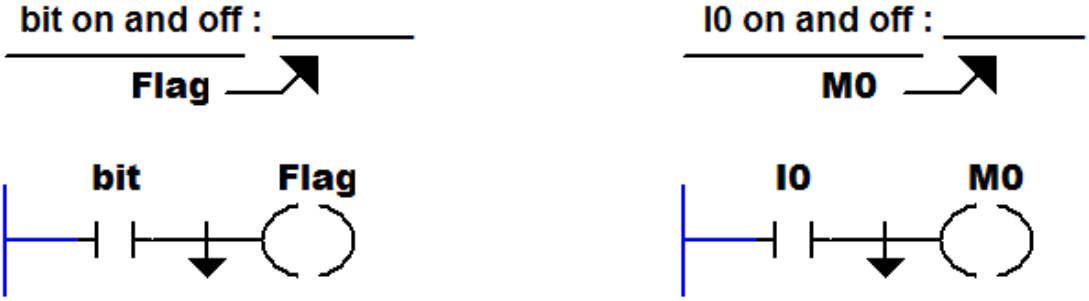

When any bit (I/ Q/ M) gets on & off and after that we have to perform some task then we should make our ladder like this; Bit NO contact, negative pulse and a flag. Now this flag will get a pulse after nit is off. You can use this flag as pulse to perform next tasks. You can see the example right side in picture. When I0 is on & off, after that we have to do some task. We need to write ladder for I0 first; I0 NO, negative pulse and a flag M0. Now this M0 will be used to perform next task. Let us revise important points before we start pulse programming.

**Important points:**

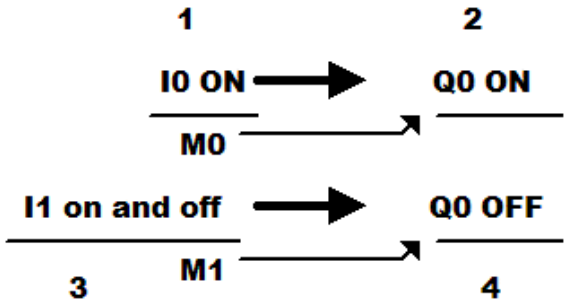
1> **To switch on use NO contact of the condition signal.**
2> **To switch off use NC contact of the condition signal.**
3> **When we have start signal, we should latch it with some flag.**
4> **When we have any stop signal, we should break the latch directly from it.**
5> **When we get last signal in our program, we should break all latches from it. (It means that lastly after one cycle all coils must get off. If any coil will remain on after one cycle then machine will not run in next cycle.)**
6> **To repeat process in programs based on many timers, take last signal and break first timer.**

7> **To repeat process in programs based on many latches, take last signal and break all latches except start/ first latch.**

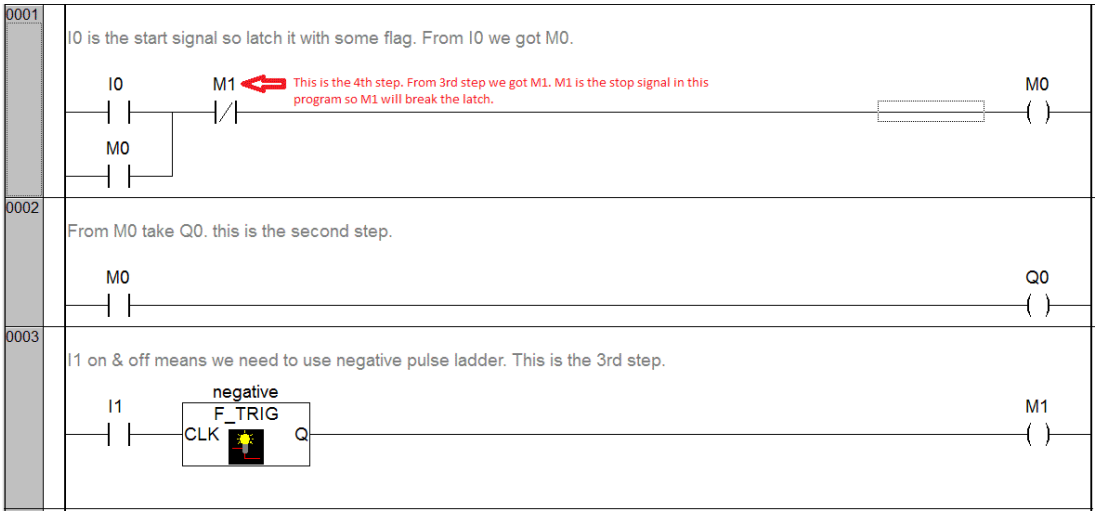
8> **When we have condition bit on and off then we have to use negative pulse syntax ladder.**




**EX26:**



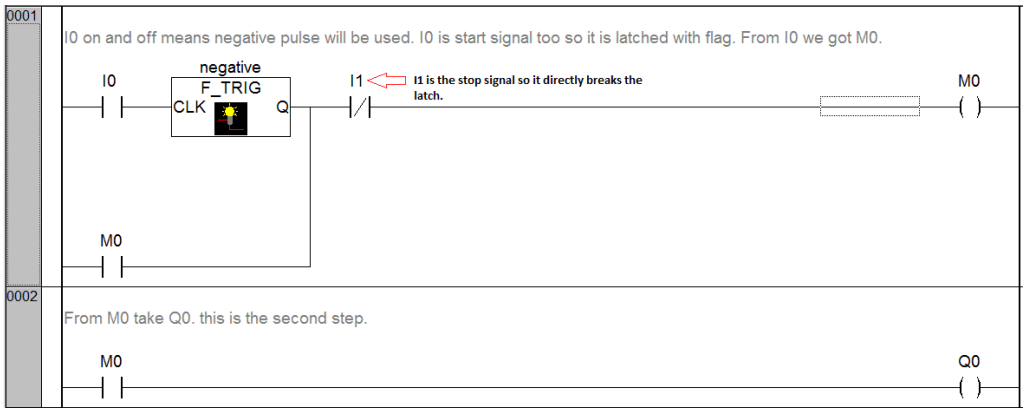

Just follow the steps. Don't think about all steps. Start from first step.




**EX27:**

I0 on and off  Q0 on

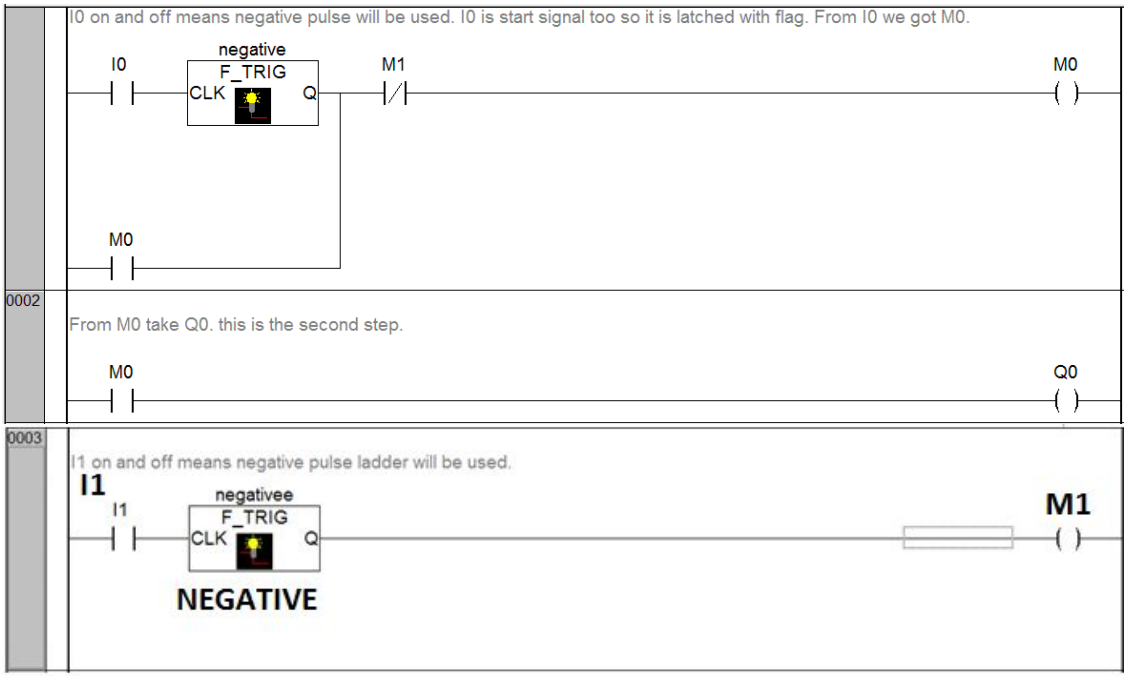
I1 on  Q0 off




**EX28:**

I0 on and off  Q0 on

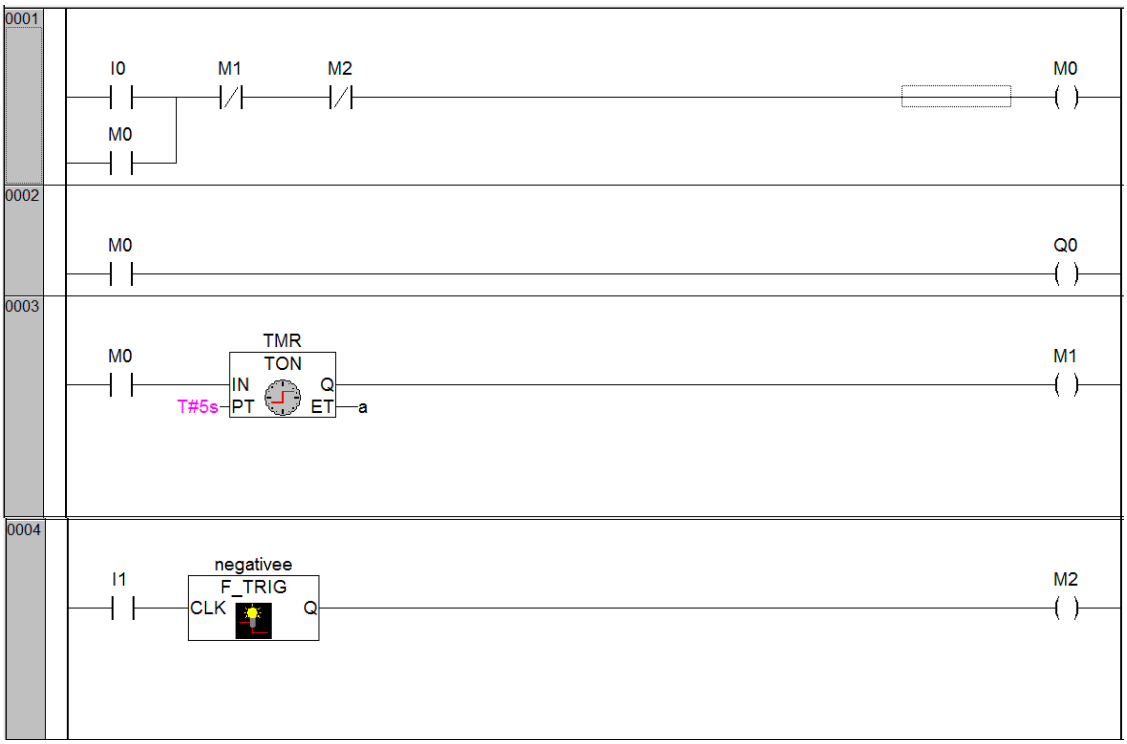
I1 on and off  Q0 off




**EX29:**

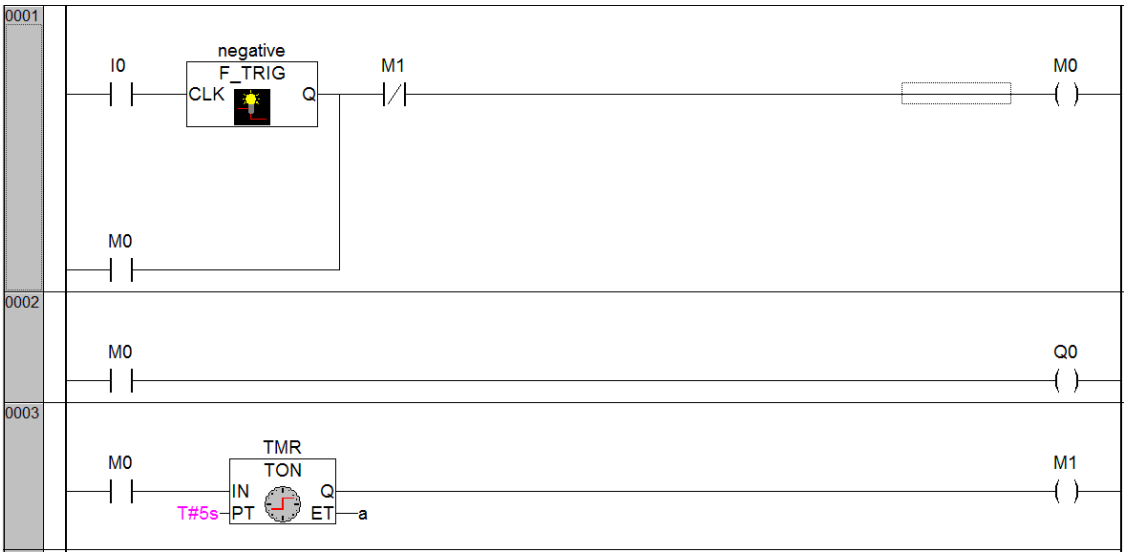
I0 on  Q0 on for 5 seconds

I1 on and off  Q0 off




**EX30:**

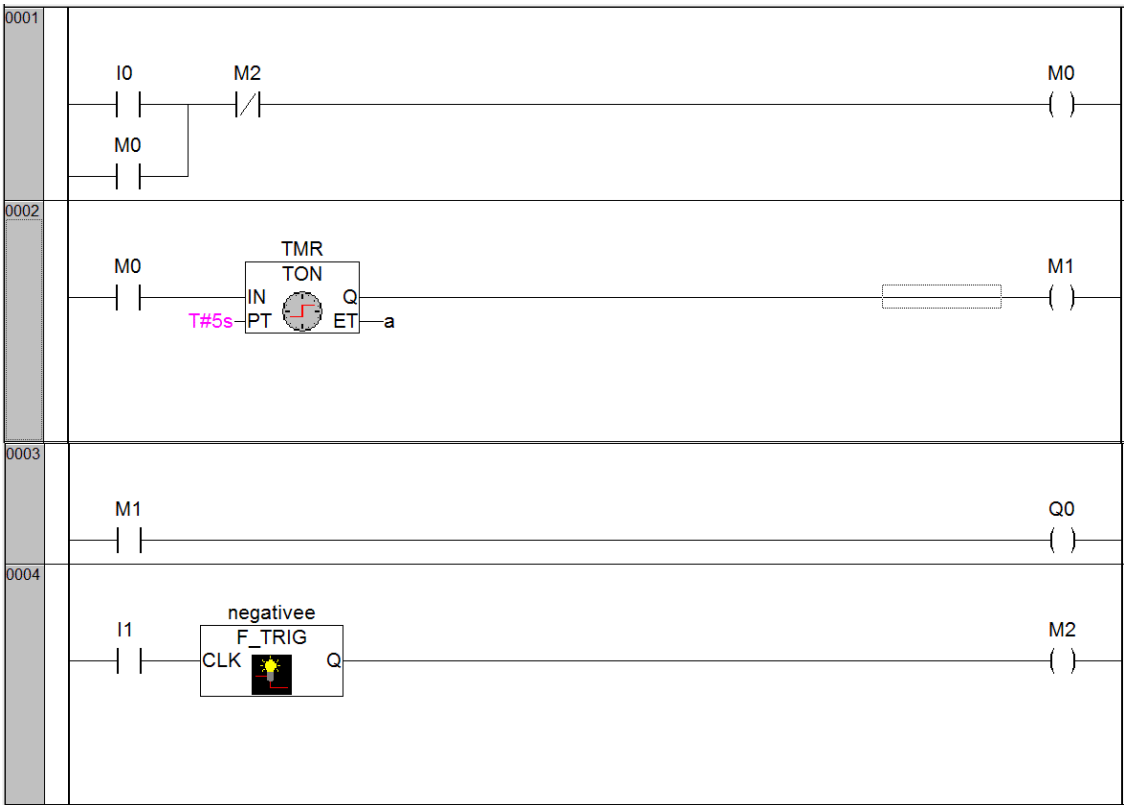
I0 on and off □ Q0 on for 5 seconds




**EX31:**

I0 on □ Q0 on after 5 seconds

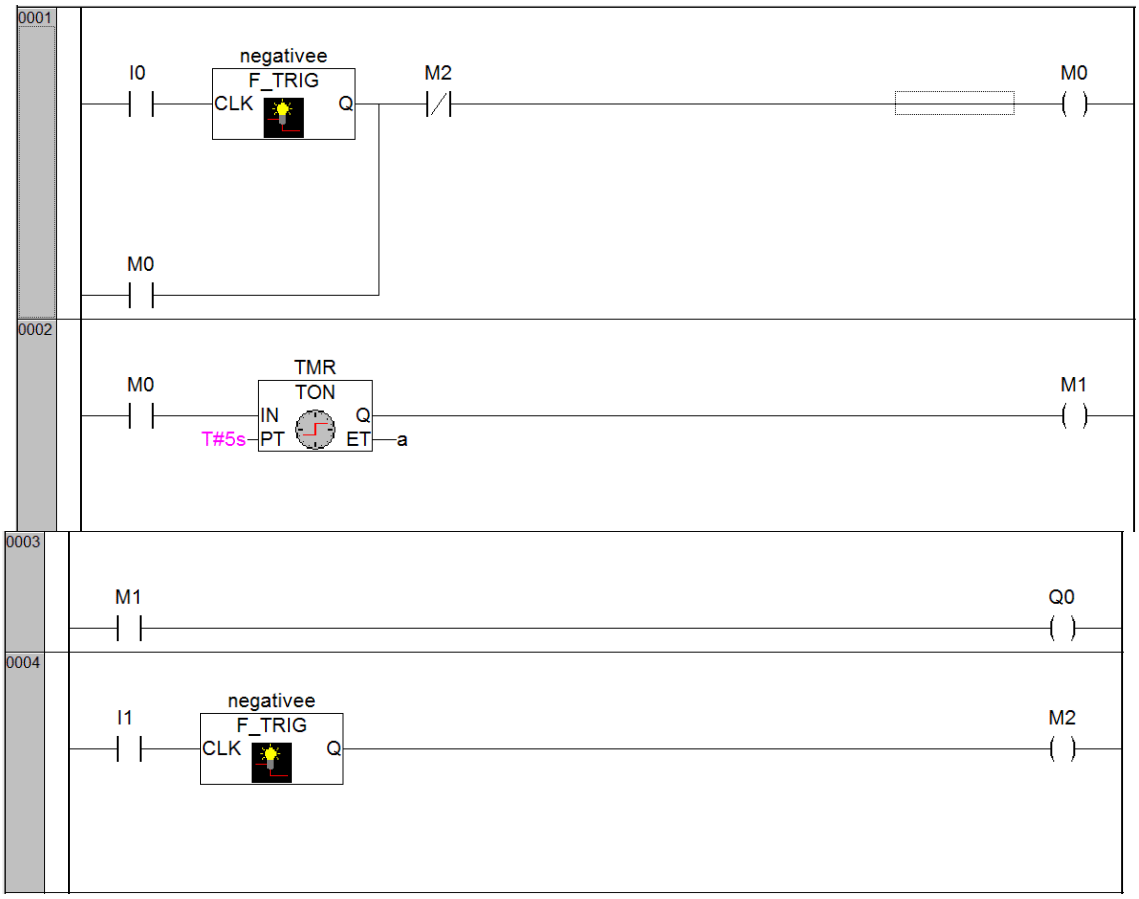
I1 on & off □ Q0 off




**EX32:**

I0 on and off  Q0 on after 5 seconds

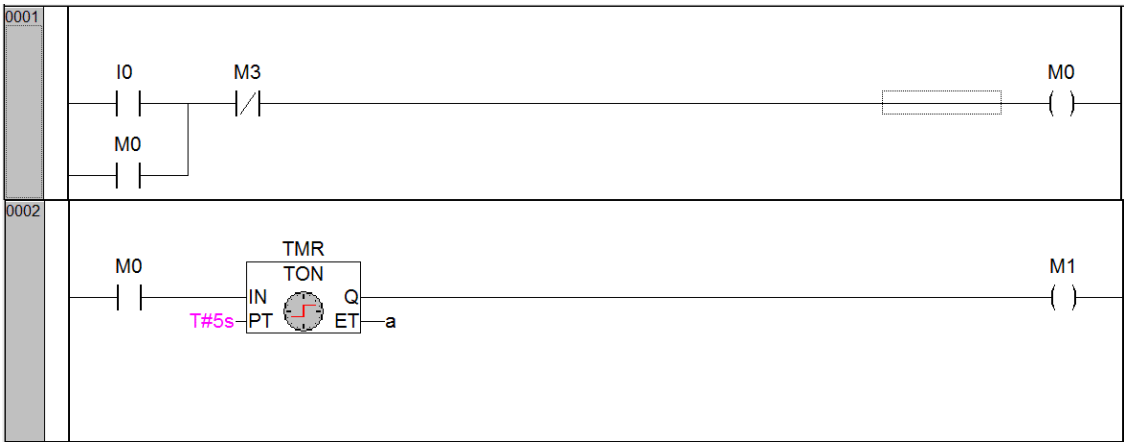
I1 on & off  Q0 off

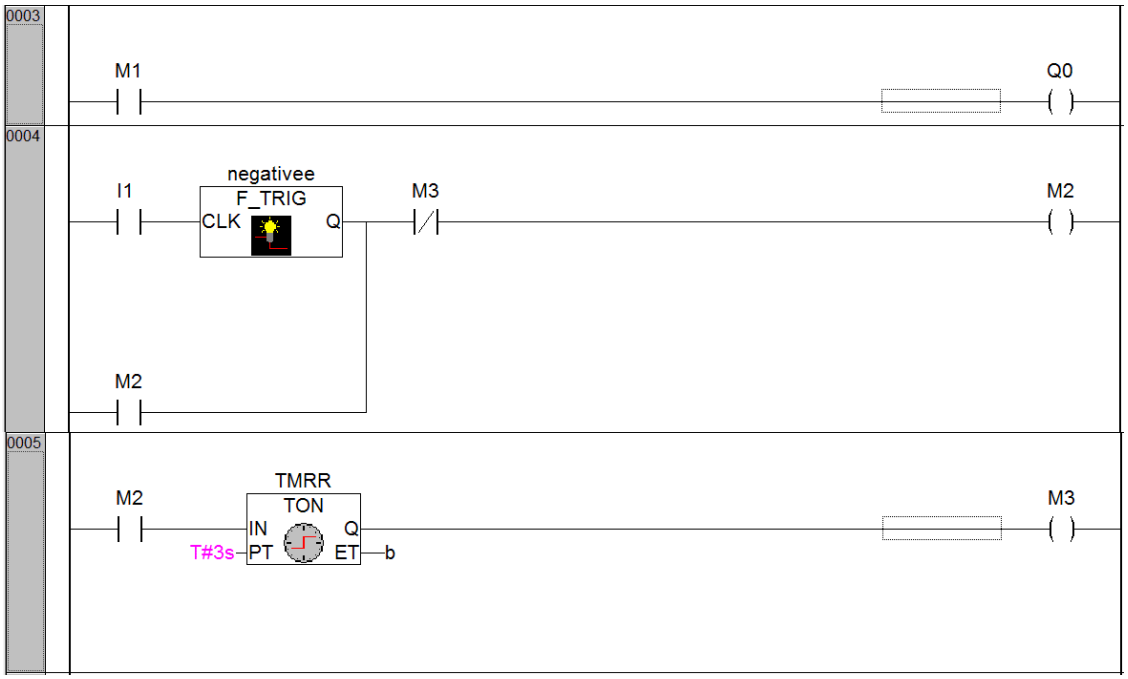



**EX33:**

I0 on  Q0 on after 5 seconds

I1 on and off  Q0 off after 3 seconds

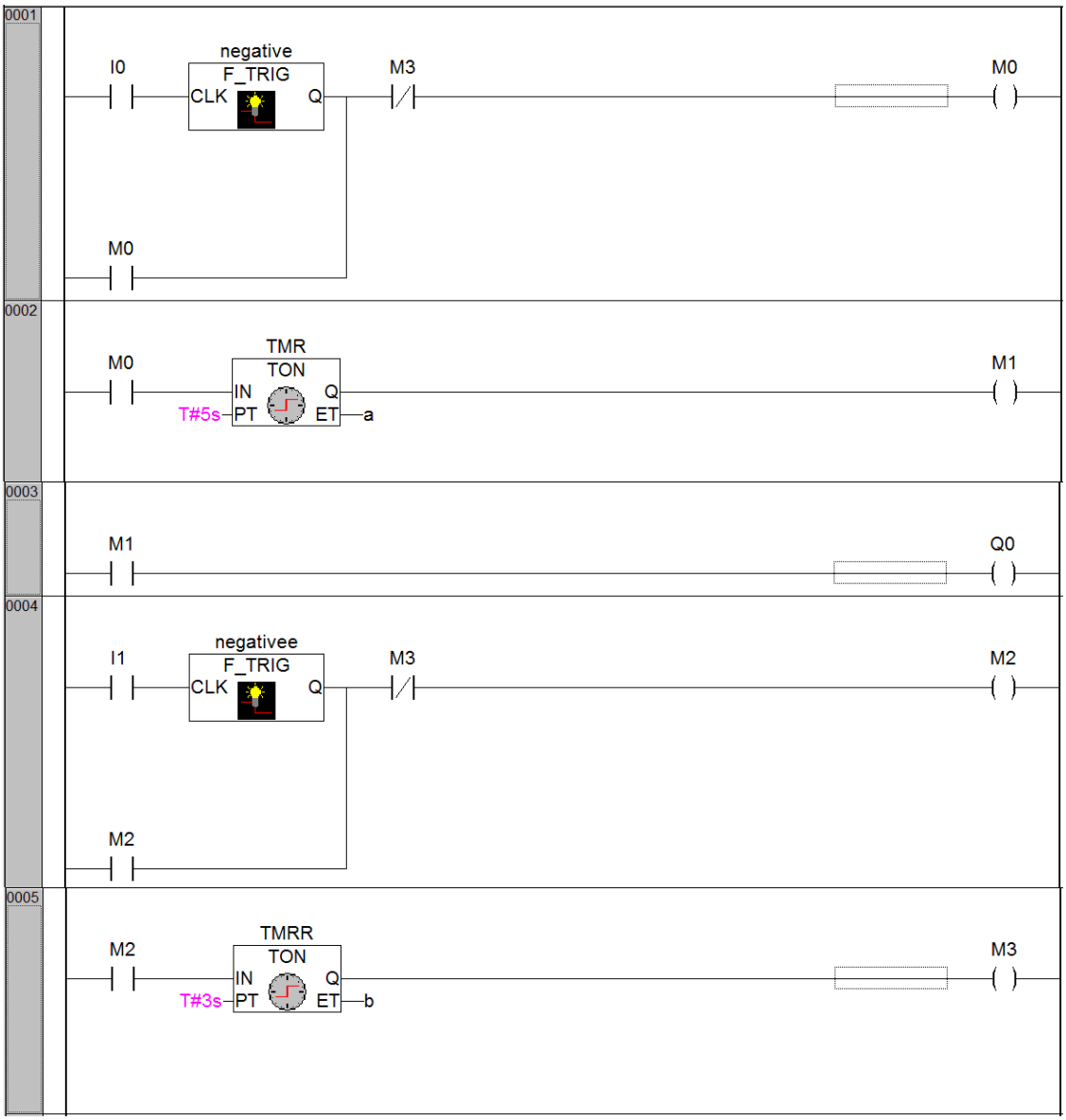





**EX34:**

I0 on and off  Q0 on after 5 seconds

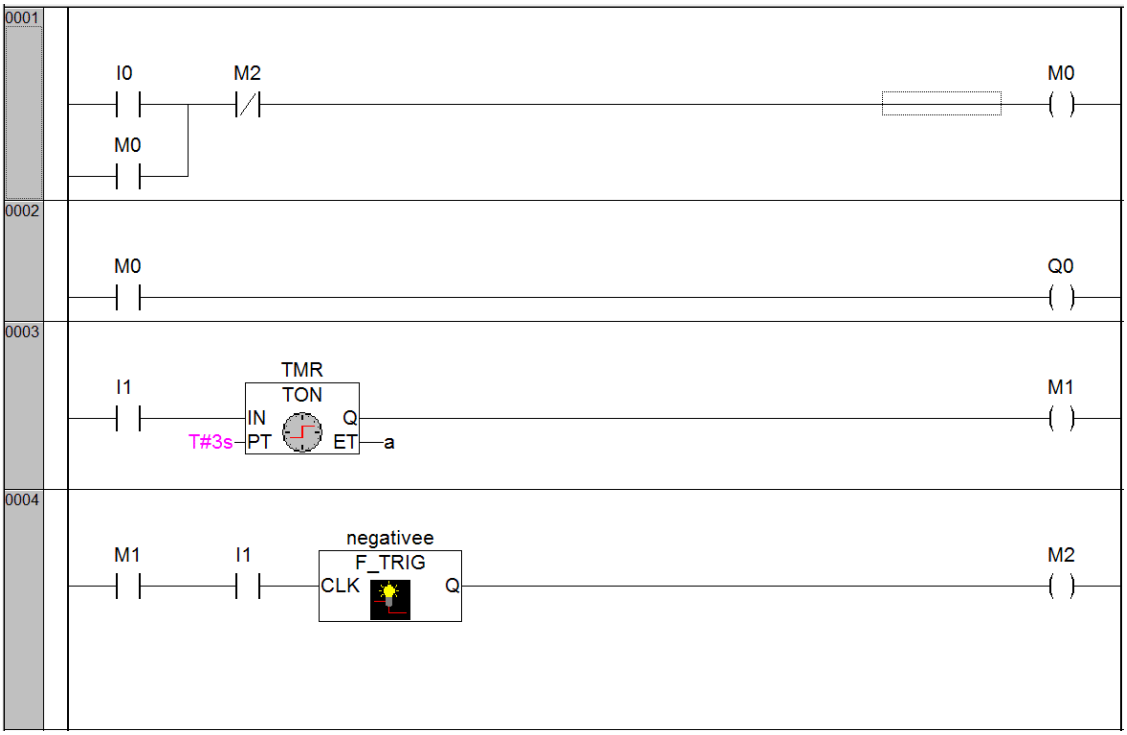
I1 on and off  Q0 off after 3 seconds




**EX35:**

I0 on □ Q0 on

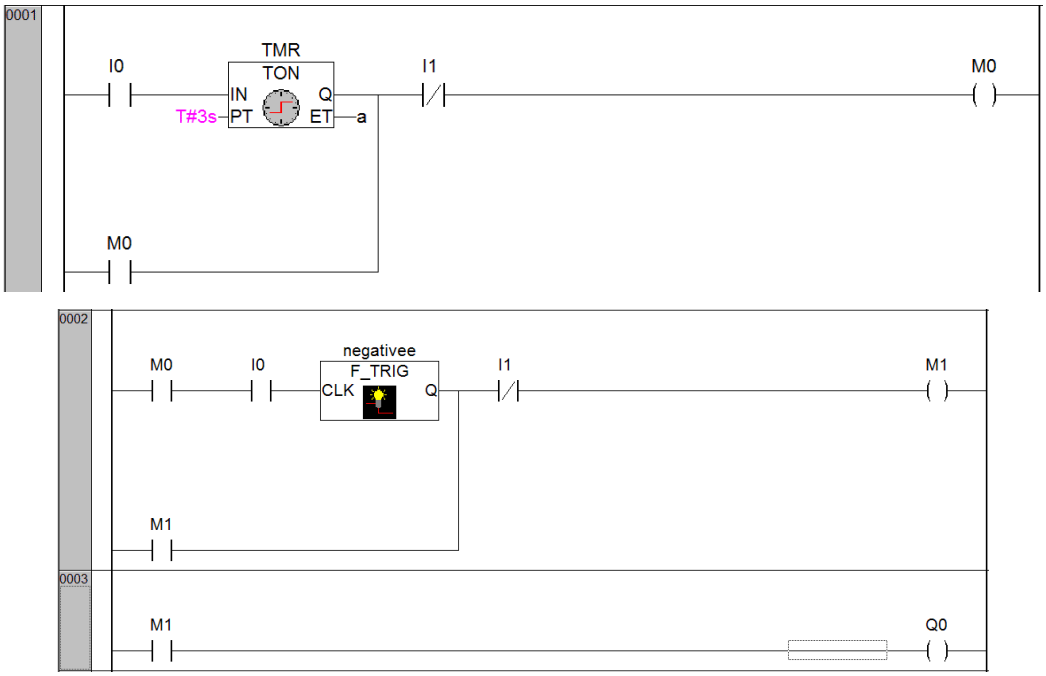
I1 pressed for 3 seconds □ Q0 off after I1 is released/ off.




**EX36:**

I0 pressed for 3 seconds  Q0 on after I0 is released/ off

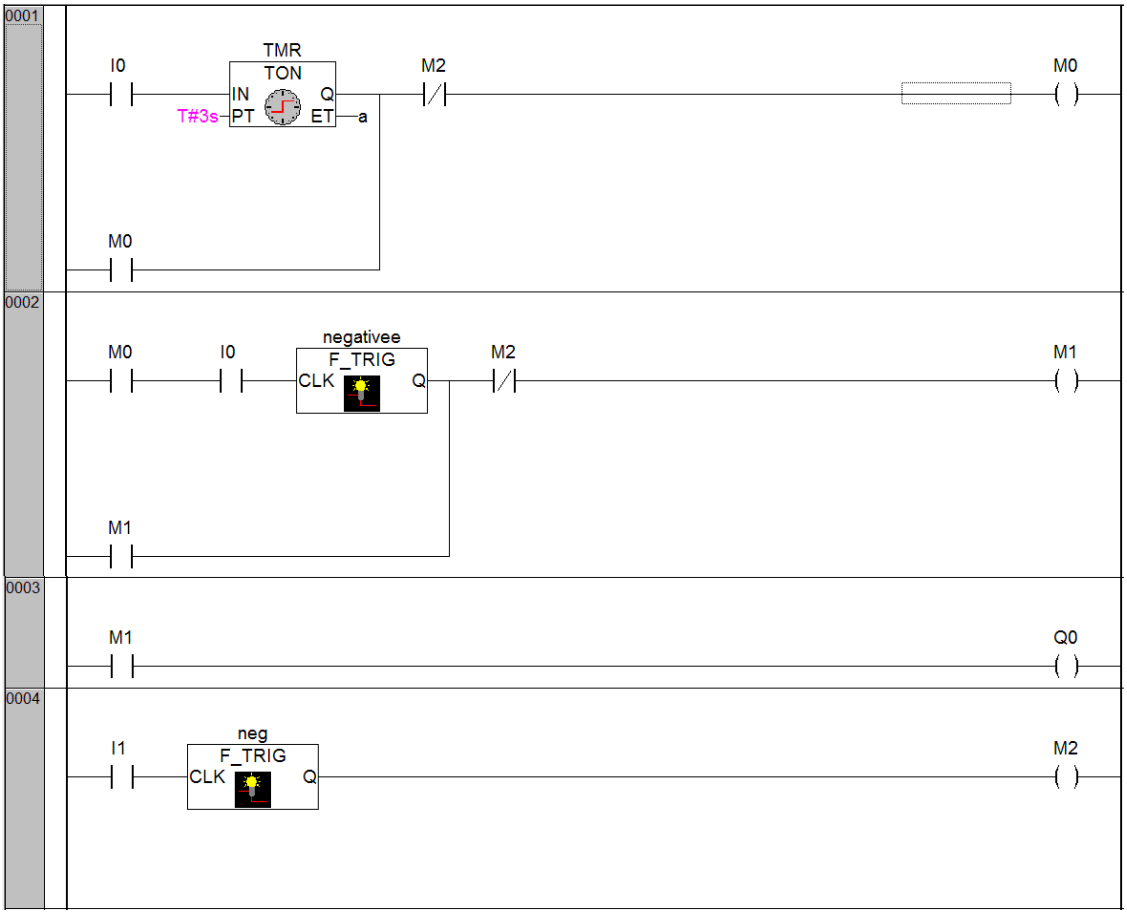
I1 on  Q0 off




**EX37:**

I0 pressed for 3 seconds  Q0 on after I0 is released/ off

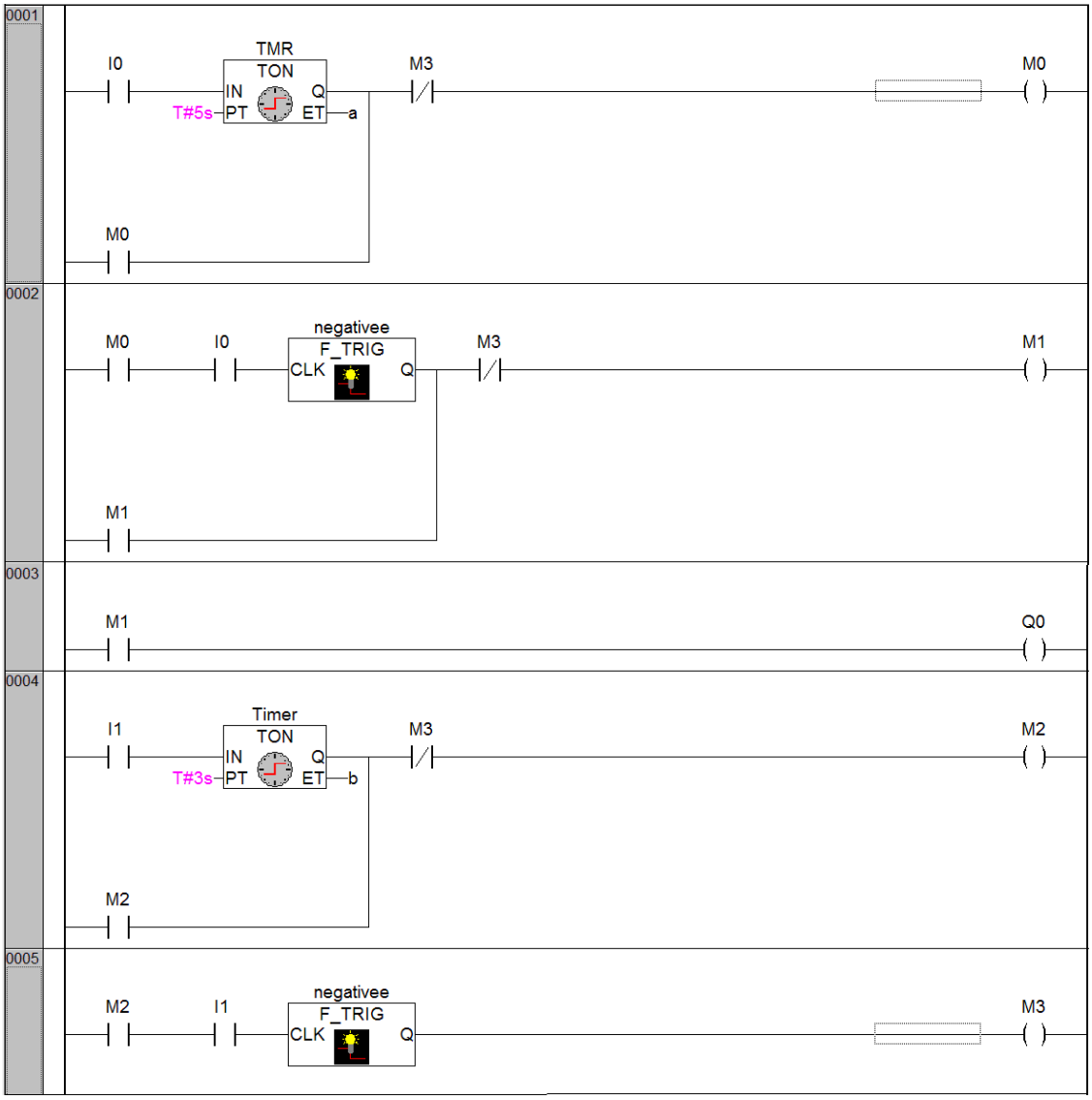
I1 on and off  Q0 off




**EX38:**

I0 pressed for 5 seconds  Q0 on after I0 is released/ off

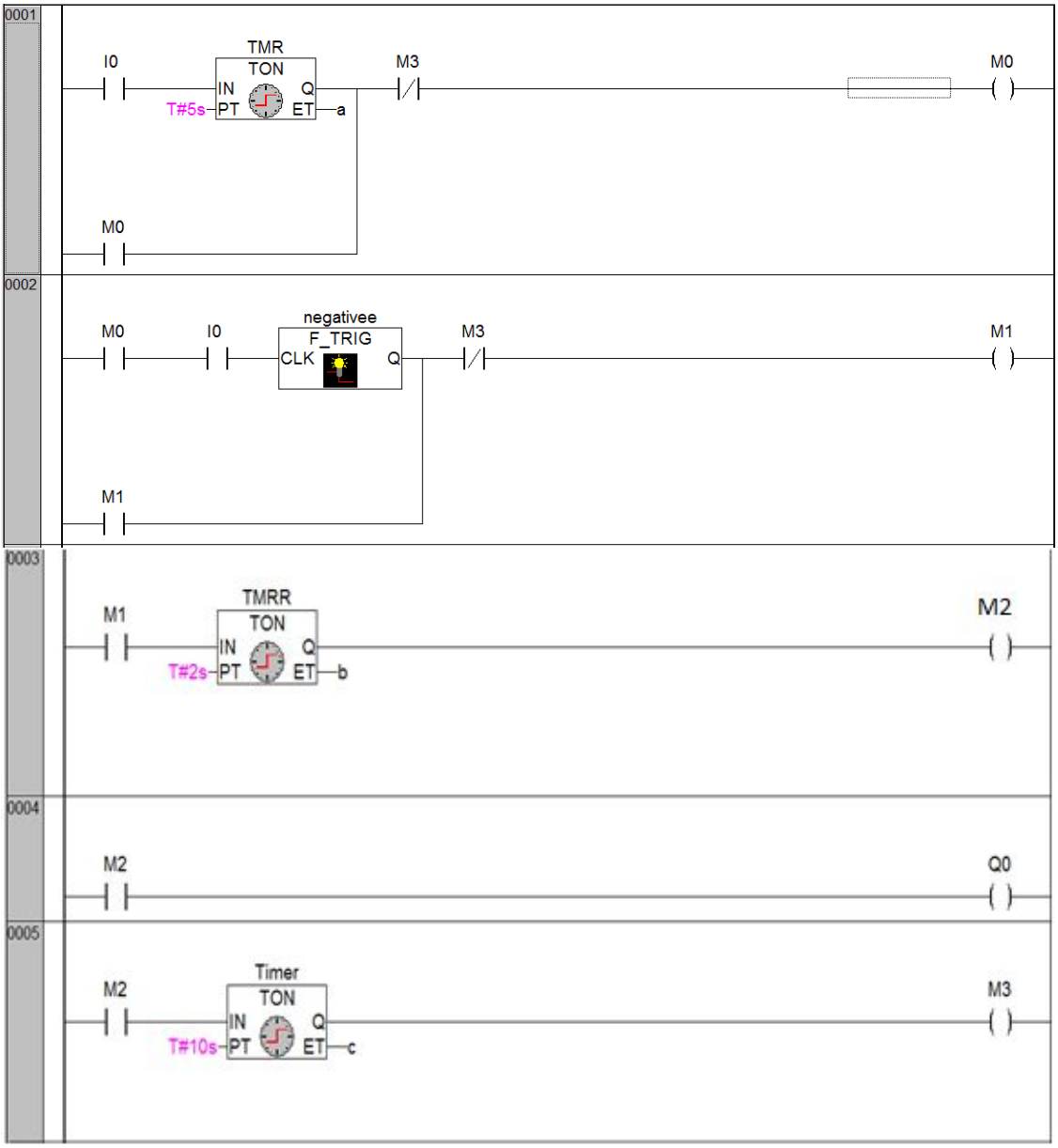
I1 pressed for 3 seconds  Q0 off after I1 is released/ off




**EX39:**

I0 pressed for 5 seconds □ Wait for 2 sec after I0 is released/ off

After 2 seconds □ Q0 on for 10 seconds



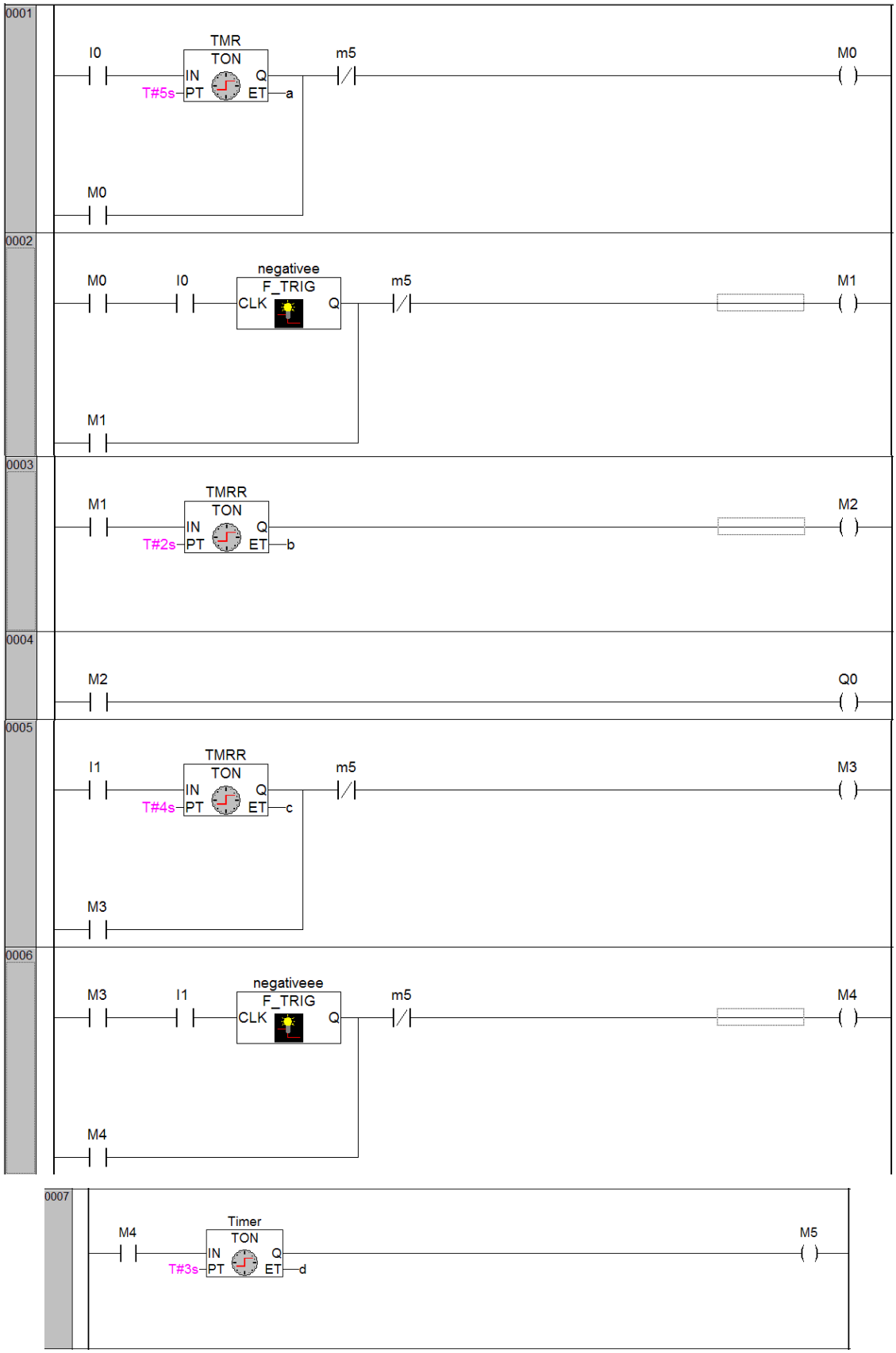

**EX40:**

I0 pressed for 5 seconds □ Wait for 2 sec after I0 is released/ off

After 2 seconds □ Q0 on

I1 pressed for 4 seconds □ Wait for 3 sec after I1 is released/ off

After 3 seconds □ Q0 off


0001
I0
TMR
TON
IN
PT
T#5s
Q
ET
a
m5
M0
M0
0002
M0
I0
negativee
F_TRIG
CLK
Q
m5
M1
M1
0003
M1
TMRR
TON
IN
PT
T#2s
Q
ET
b
M2
0004
M2
Q0
0005
I1
TMRR
TON
IN
PT
T#4s
Q
ET
c
m5
M3
M3
0006
M3
I1
negativeee
F_TRIG
CLK
Q
m5
M4
M4
0007
M4
Timer
TON
IN
PT
T#3s
Q
ET
d
M5

**EX41:**

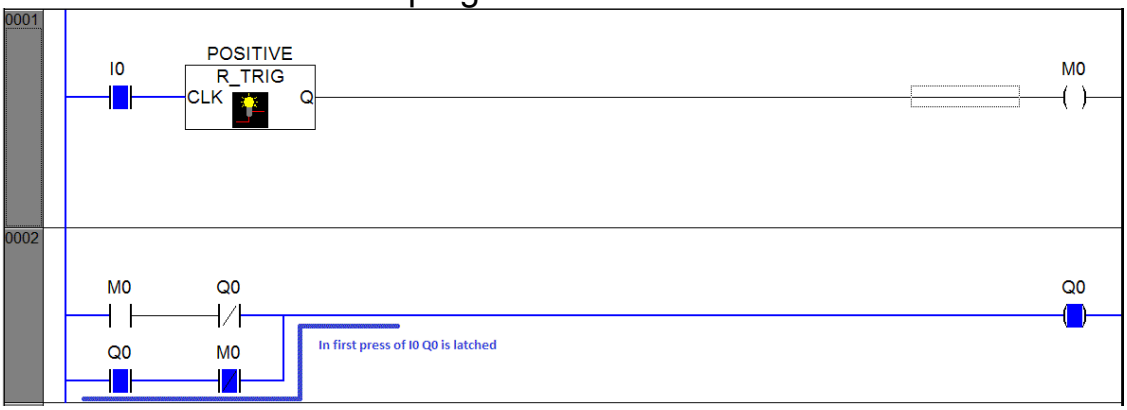
I0 pressed first time  Q0 on

I0 pressed again second time  Q0 off

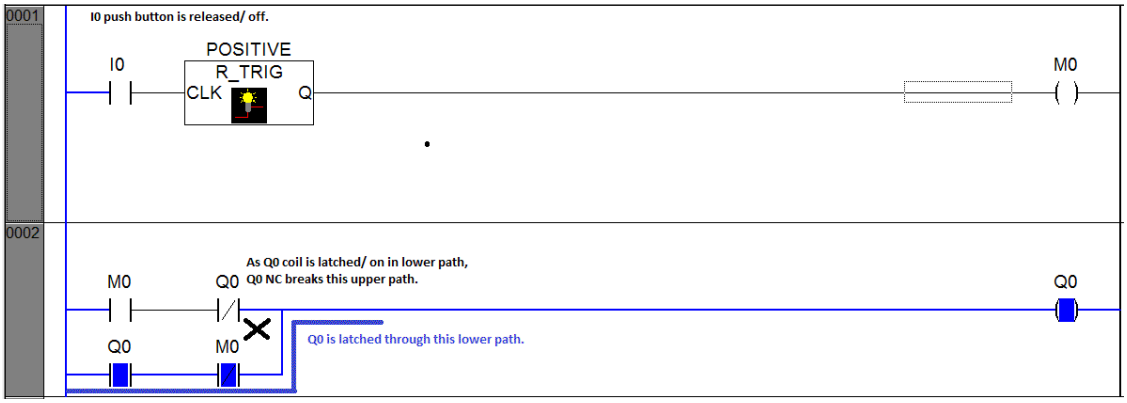
(By pressing I0 push button change Q0 status any time.)

This is an important example.

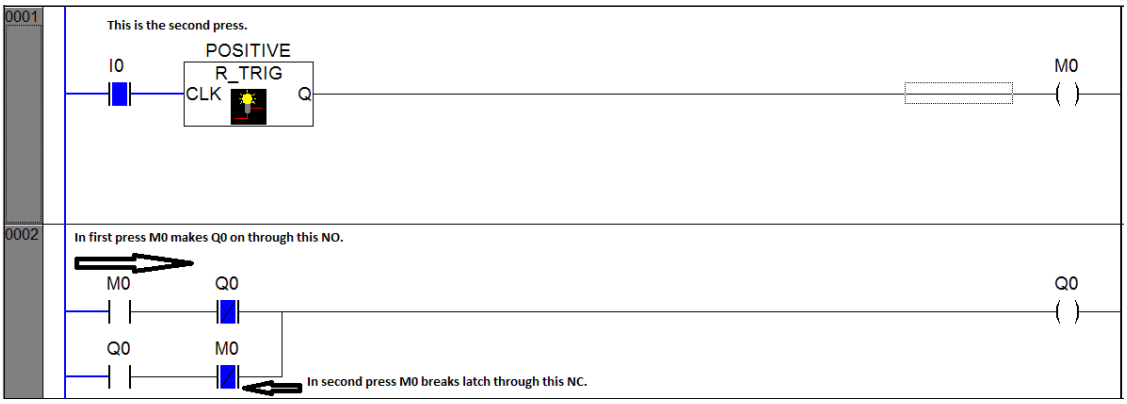

This is initial status of signal. Nothing is on. This program is totally based on scan. Pulse is compulsory here. Let me remind you that pulse is on for 1 program scan time.




When I0 is pressed first time, scan starts from first contact in ladder 1. As I0 is on, M0 is on. Scan reaches to second ladder. As M0 is on Q0 gets on and makes a latch. AS scan reaches to Q0 coil, scan finds it the last coil of the program so scan gets over. As scan is over pulse M0 also gets off automatically.




When I0 is off, Q0 is latched and Q0 NC breaks upper path. It is necessary to break upper path so that in second time press M0 does not reaches to Q0 because in second press Q0 has not to get on but off.




In second press of I0 push button again M0 pulse is high in first ladder. In second ladder M0 NO is high but does not reach to Q0 coil because of Q0 NC has been breaking upper path but at the same time M0 NC breaks latch in lower path and makes Q0 off. As Q0 is off, M0 pulse is also off automatically as scan is over.

**EX42:**

I0 on and off first time  Q0 on

I0 on and off again second time  Q0 off

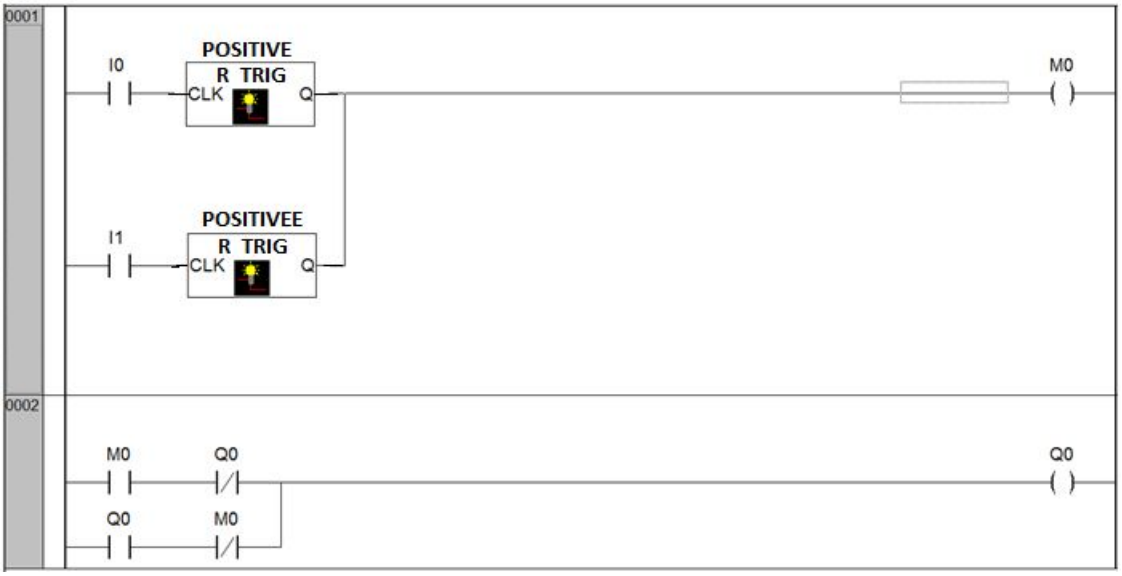

This will work on release/ off of push button I0.

**EX43: Staircase light/ motor operation by 2 push buttons. By pressing either push button changes the status of output.**

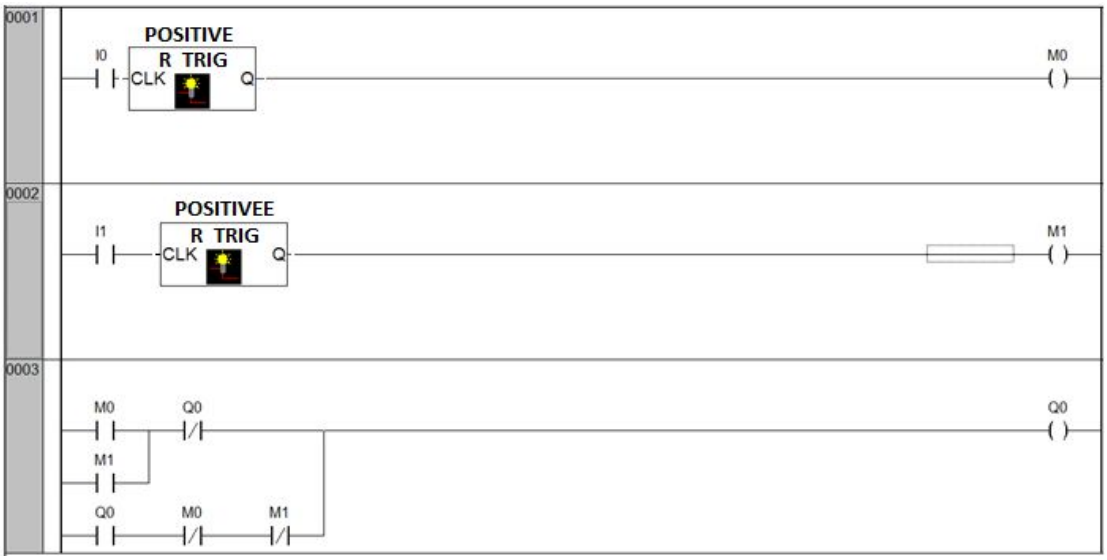
There is a motor on first floor and your office cabin is on ground floor. Place a push button I0 on first floor near motor and another push button I1 in your office cabin on ground floor. When you are on first floor, you can get motor on and off by the push button I0. When you are in your cabin, from cabin also you can get the motor on and off by I1. Suppose you start motor by I0, you can stop it by I1 also. If you start motor by I1 then you can stop it by I0 also. So, we can say that we have to change the status of motor Q0 by pressing either bush button. The same can be applied in staircase lamp also.


In previous example we made program for 1 push button to operate Q0. In this program we have to do the same by two push buttons. So, what you do by I0 you have to do the same with I1 also so take both push buttons in

parallel. You press I0 or I1, M0 will be high and it will change the status of Q0.




This is also correct.

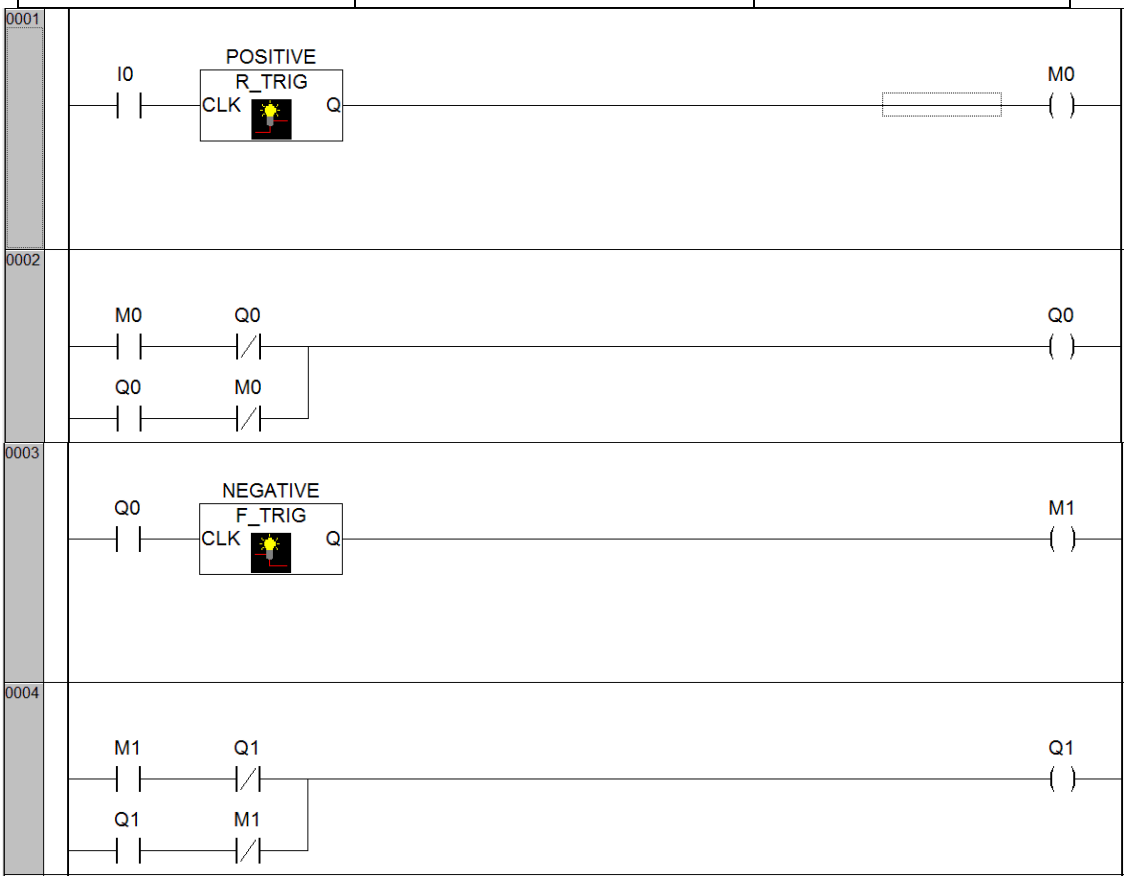



This is also a solution for the same example.

**EX44:**

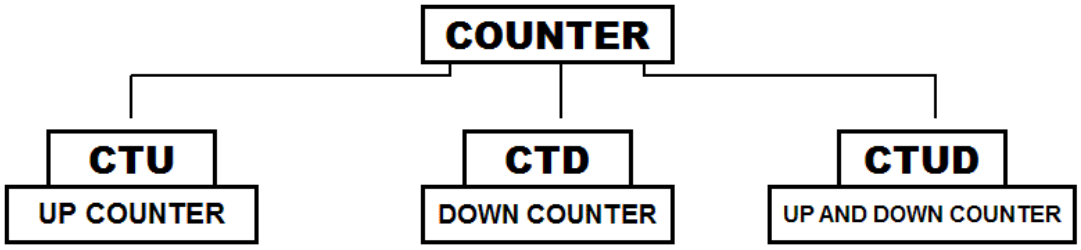
| I0 | Q0 | Q1 |
|---|---|---|
| **FIRST PRESS** | **1** | **0** |
| **SECOND PRESS** | **0** | **1** |

| | | |
|---|---|---|
| **THIRD PRESS** | **1** | **1** |
| **FORTH PRESS** | **0** | **0** |




Q0 is operated by I0 positive as you can see in truth table. Whenever I0 is pressed, Q0 status is changed. Q1 is operated by Q0 negative as you can notice in truth table that whenever Q0 gets off, Q1 status changes.

Let us learn another instruction **Counter.** Counter we use to count any bit signal. In any example if you have to count anything, you use counter instruction.




Counter we have of three types; Up counter, down Counter and Up & down counter.

Counter has 5 parameters. Four parameters are similar to timer parameters.

**EN/ Enable bit:** When this enable is high, counter counts.

**PV/ Pre-set value:** PV is the set point. How many counts you want; you can mention here.

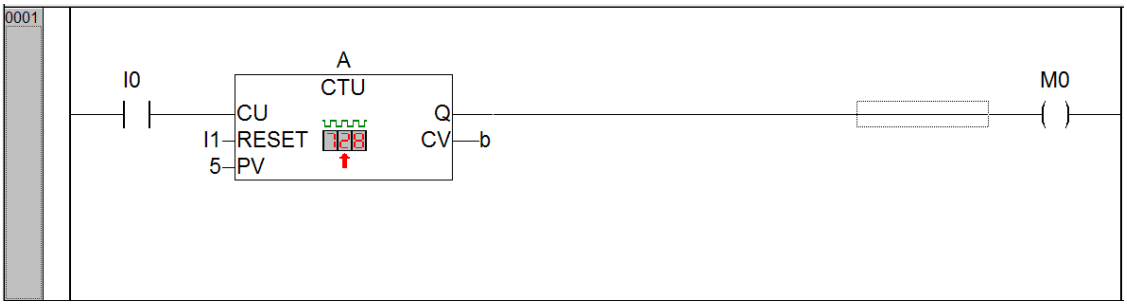
**ACC/ PV (Accumulator/ Count Value):** it is a display where we monitor the present/ current count.

**DN/ Done bit:** Done is the confirmation signal of counter.

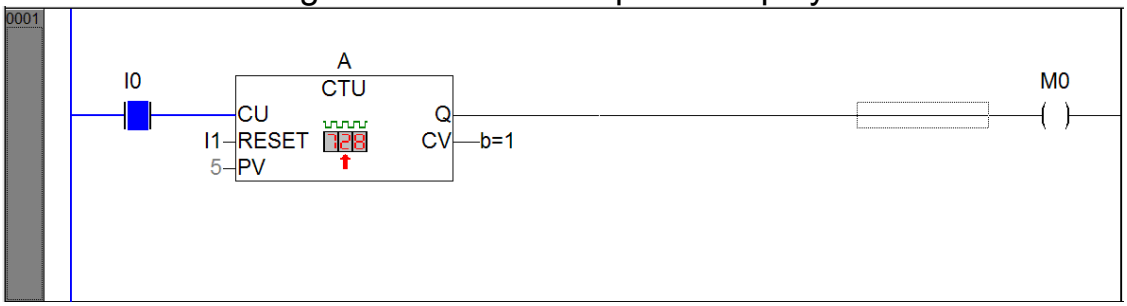
**R/ L (Reset/ Load):** reset or load is used to reset or load counter accumulator.

Counter does not need continue supply to its enable to count. It just counts the high/ on status of any bit signal. If enable is continue high then it does not start count continuously but count one only. Whenever enable bit is high counter counts (there are some counters that count on low state or both states of a bit signal).

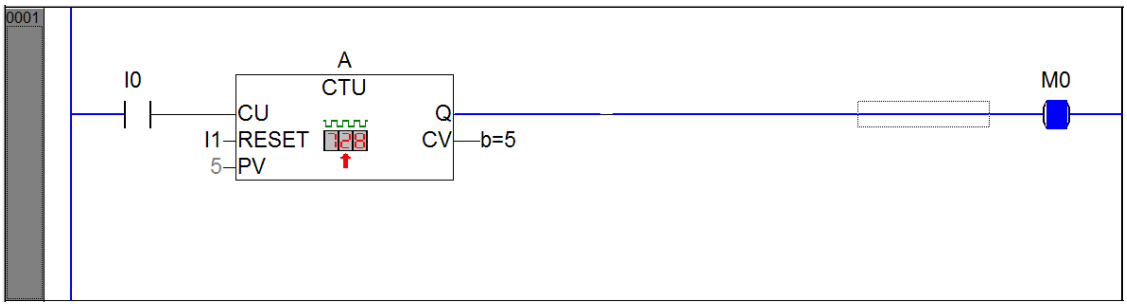
## CTU/ UP COUNTER




**CTU** has five parameters (EN/ CU, PV, CV, DN/ Q and Reset). When EN/ CU I0 is high counter counts up and displays it on CV.

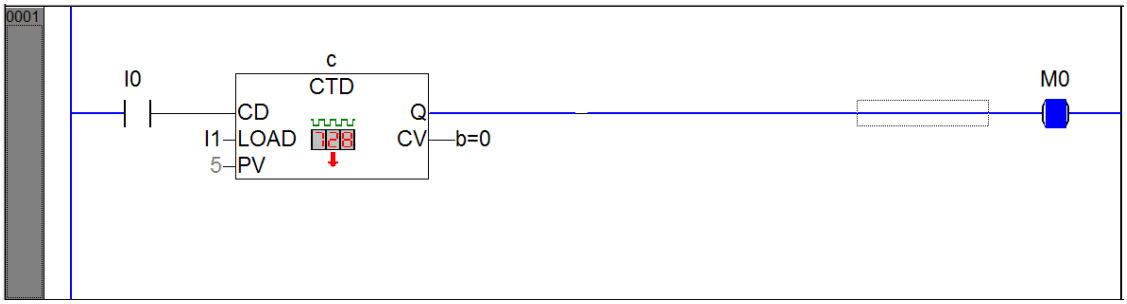



When I0 is high, you can see that CV displays 1. Whenever EN I0 is high, it would be increment by 1 in CV.

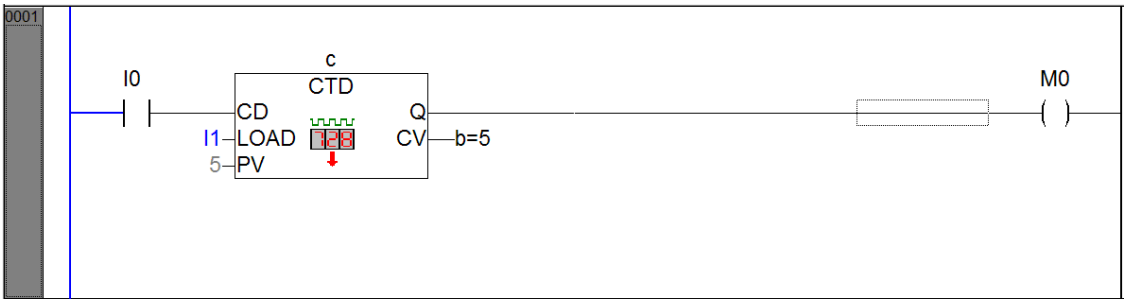



When I0 is on five times, CV is equal (CV>= PV □ DN on) to PV and done bit M0 is on. If I0 is on 6th time then CV will show 6 but once DN is high, it will be continuing high. When DN is high and CV>= PV, this is the set condition of CTU. To reset (DN off and CV=0) CTU we need to high Reset bit i. e. I1 in our above example.

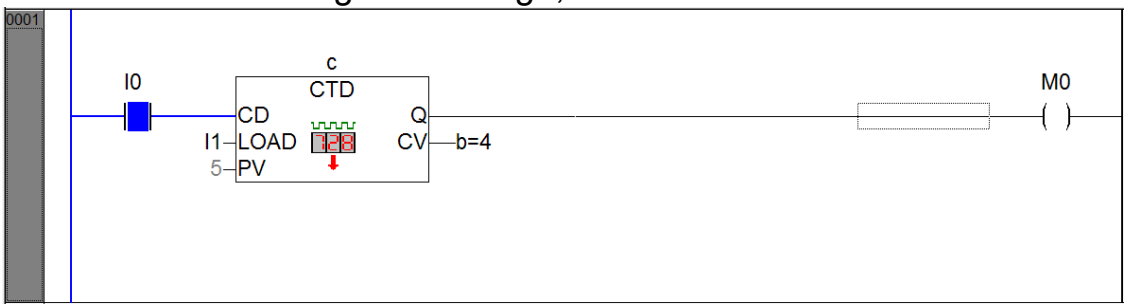
**CTD/ down counter:**




CTD has also five parameters (EN/ CU, PV, CV, DN/ Q and Load). This is the initial status of CTD in above picture. CTD is very similar to CTU. CTD also counts as CTU counts. Only the difference between CTU and CTD is its CV. CTU's CV displays up count 1, 2, 3, 4…… but CTD's CV displays down count 4, 3, 2, 1, 0. In CTU DN gets high when CV>= PV but in CTD, DN gets high when CV<= 0. As initially CV is 0 so you can see in above picture that DN is on initially. First of all, we need to apply load signal to load PV to CV so that it can count down.



You can see when load signal I1 is high, PV 5 is loaded to CV and DN is off now.



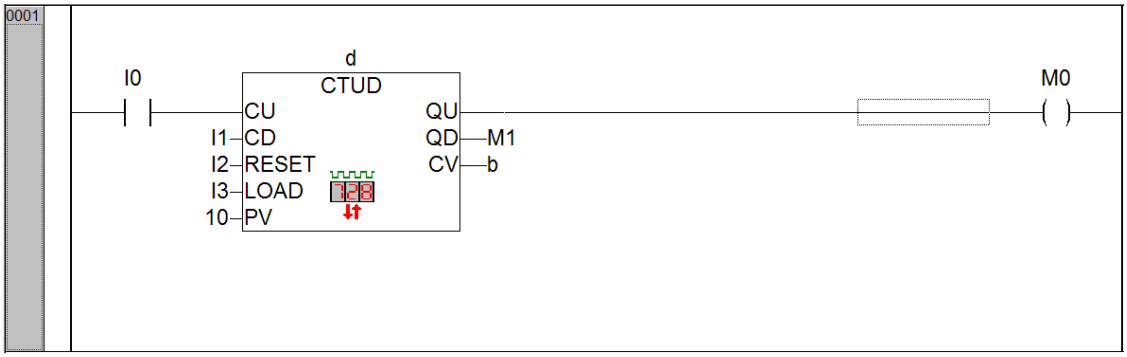

You can see that CV is 4 now when EN I0 is on. Whenever EN is high, it is decrement by 1 in CV.

So now we understand the difference between CTU and CTD.

CV>= PV □ CTU DN on. Reset on □ CTU DN off and CV= 0.

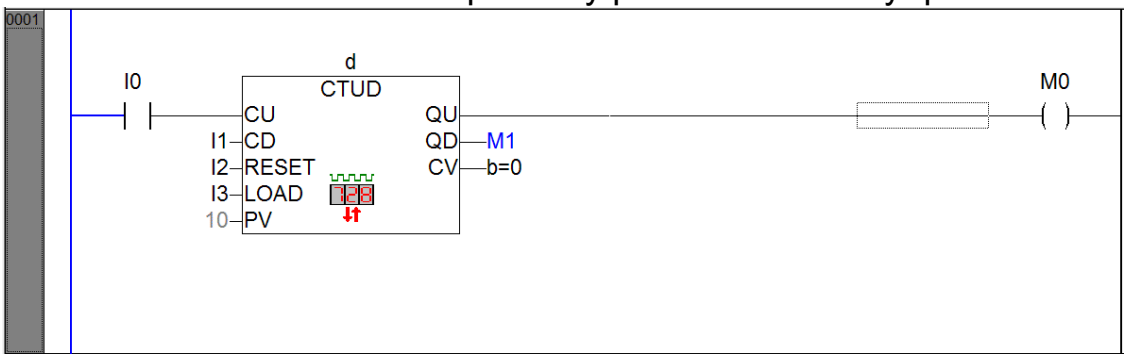
CV<=0 □ CTD DN on. Load on □ CTD DN off and CV= PV.

**CTUD/ UP & DOWN Counter:** CTUD is the combination of both counter CTU and CTD. If you understand CTU and CTD individually, you can understand CTUD.

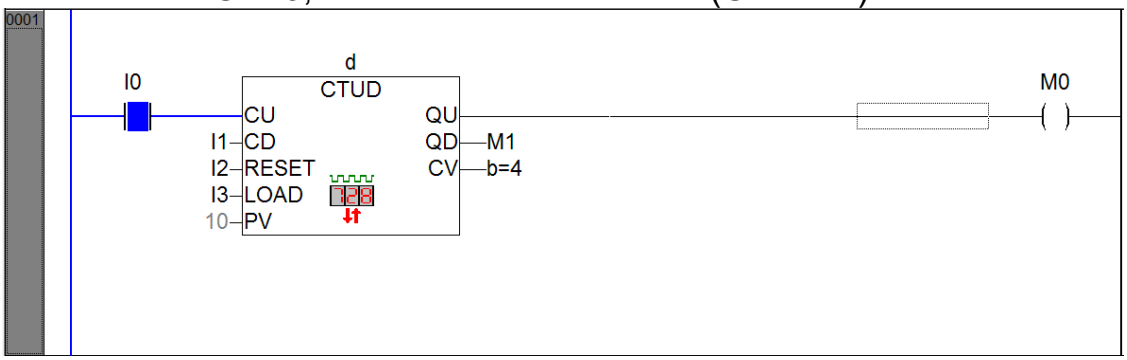



Here CU is the enable of UP counter. CD is the enable of DOWN counter. RESET is for resetting UP counter (CTU DN off and CV = 0). LOAD is for reseing DOWN counter (CTD DN off and CV= PV). PV is the set-point for both CTU and CTD. QU is the done bit of CTU and QD is the done bit of CTD. CV is the display for both the counters CTU and CTD.

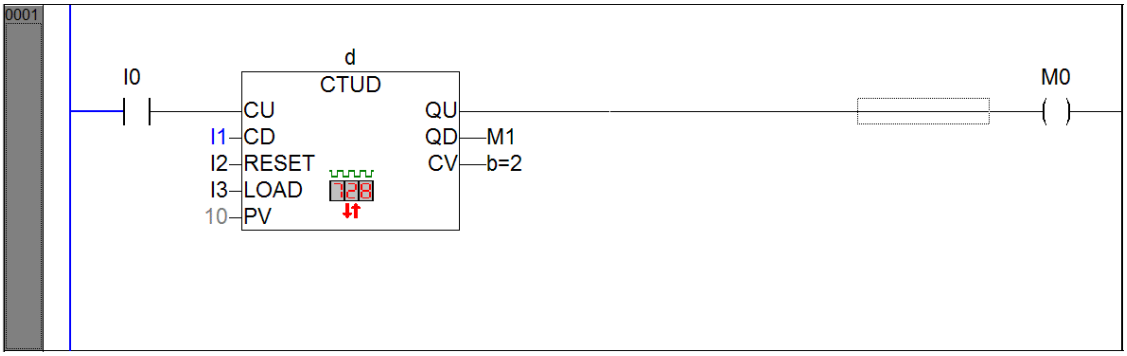
To understand its function perfectly please answer my questions.



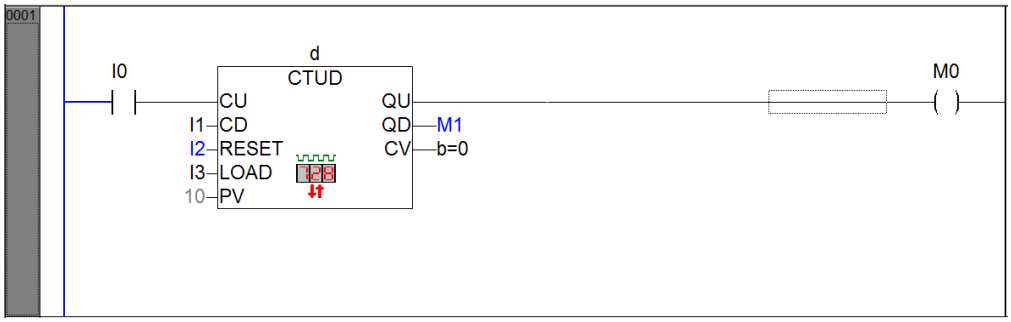

When system is powered on (nothing is pressed), what would be the value of CV? It is zero/ 0. When CV = 0, what is the status of M0 (CTU DN)? It is off. When CV=0, what is the status of M1 (CTD DN)? It is on.



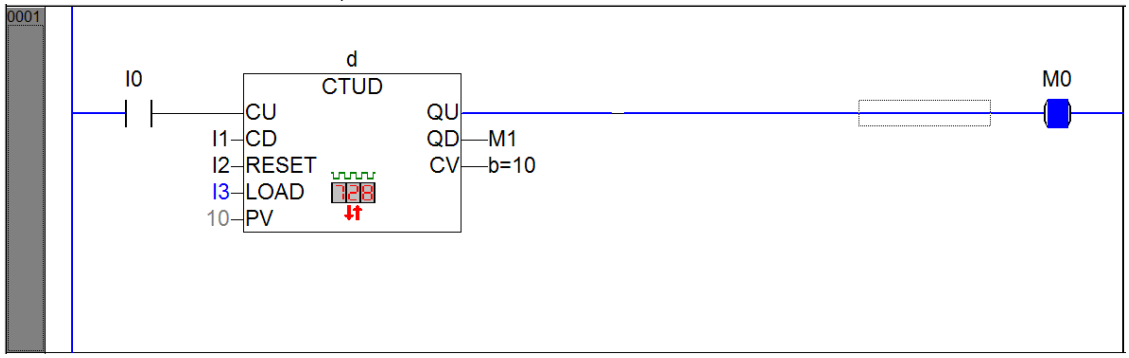

When we press I0 four times, what is the value of CV? It is 4 as I0 is for count up. When CV=4, what is the status of M0? It is off. When CV=4, what is the status of M1? It is also off.




Now press I1 twice. What is the value of CV? It is 2 because it was 4 and I1/ CTD EN is high twice so CV will show count down by 2. When CV=2, what is the status of M0? It is zero. When CV=2, what is the status of M1? It is also zero.



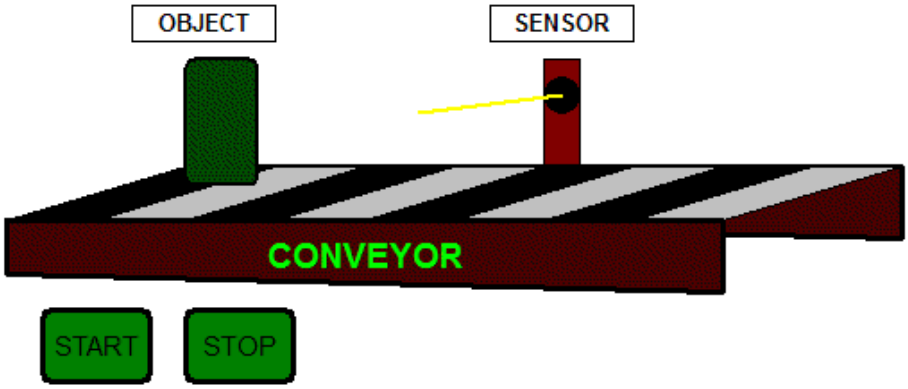

Now press I2. What is the valu0 of CV? It is zero as I2 is for reset and reset here means CV= 0. When CV=0, what is the status of M0? It is off. When CV=0, what is the status of M1? It is on.



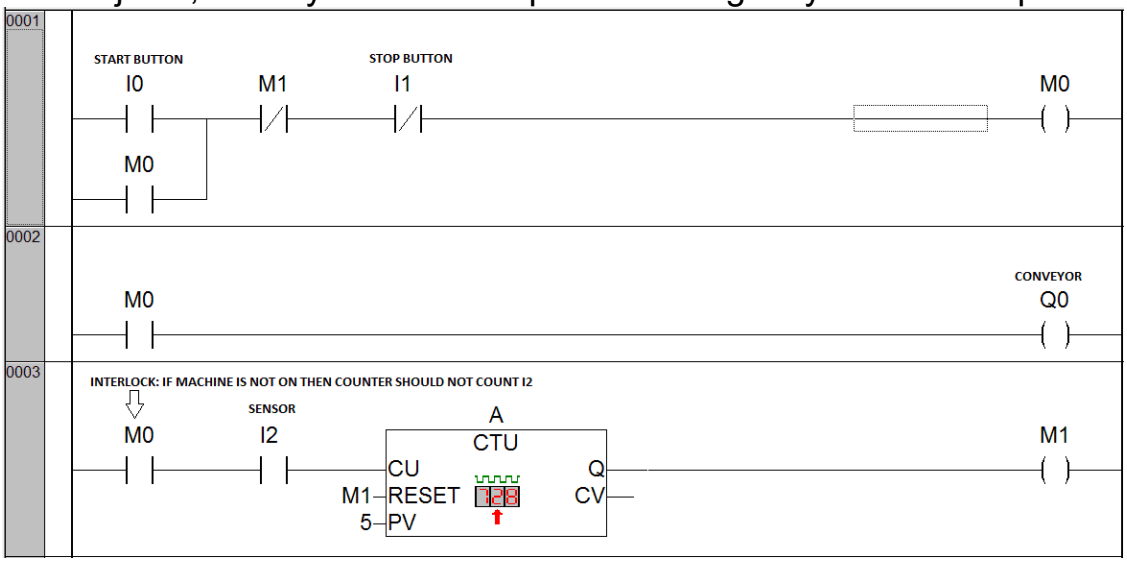

Now press I3. What is the value of CV? I3 is for load so PV 10 will be loaded to CV so CV=10. When CV=10, what is the status of M0? It is on. When CV=10, what is the status of M1? It is off.

You can notice here that you can see up and down both count on single display if you use CTUD. If you use separate CTU and Separate CTD then you will have to go through arithmetic operation to get the calculated value. CTUD has its own advantage in some applications like displaying number of vehicles in parking area where you have one-way entry and exit. In seminar hall or in Cinema hall also if you want to display number of people inside the hall then CTUD is useful.

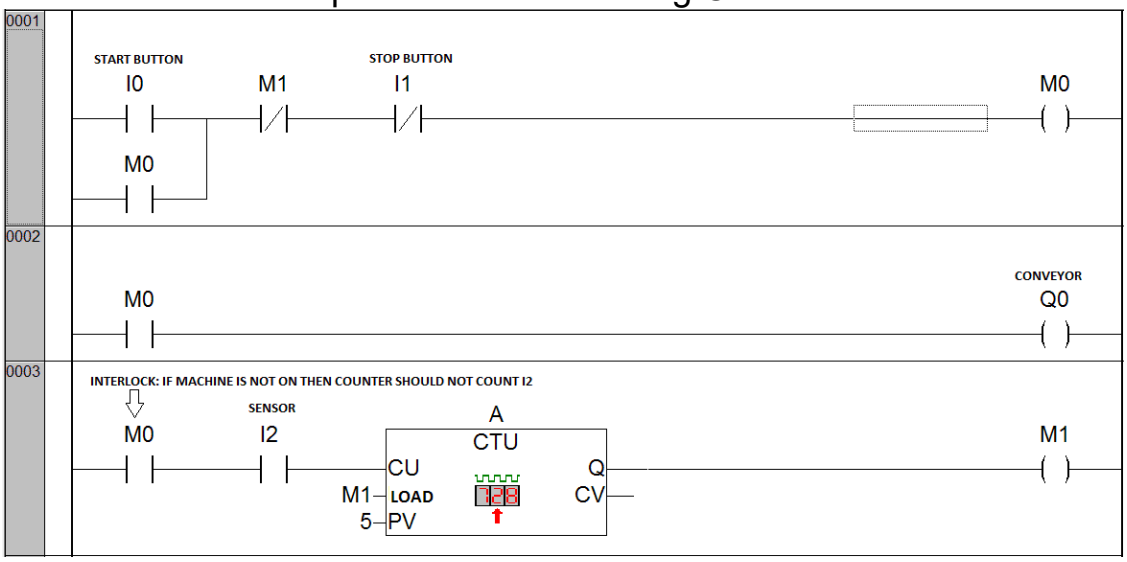
**EX45:**



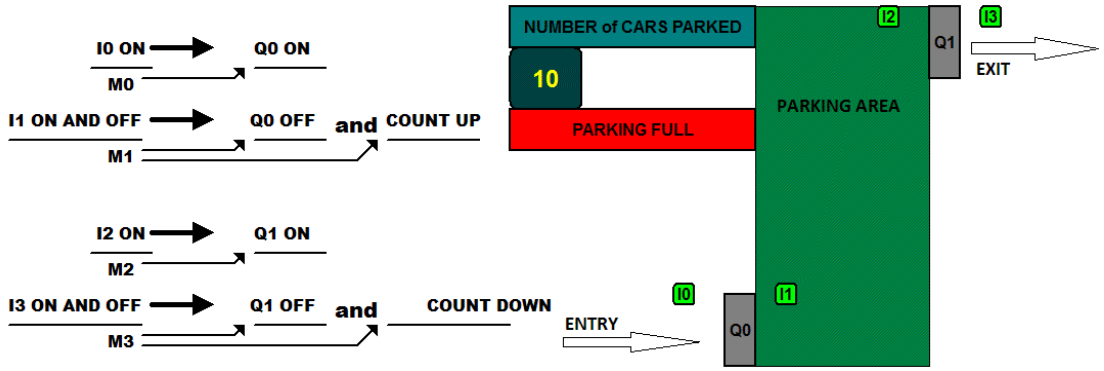

There are two push buttons I0 and I1 for start and stop. There is one photo sensor I2 and there is a conveyor Q0. When I0 is pressed, conveyor Q0 is on. When conveyor is on, objects will move on it. When sensor I2 detects 5 objects, conveyor should stop. I1 is emergency/ manual stop.



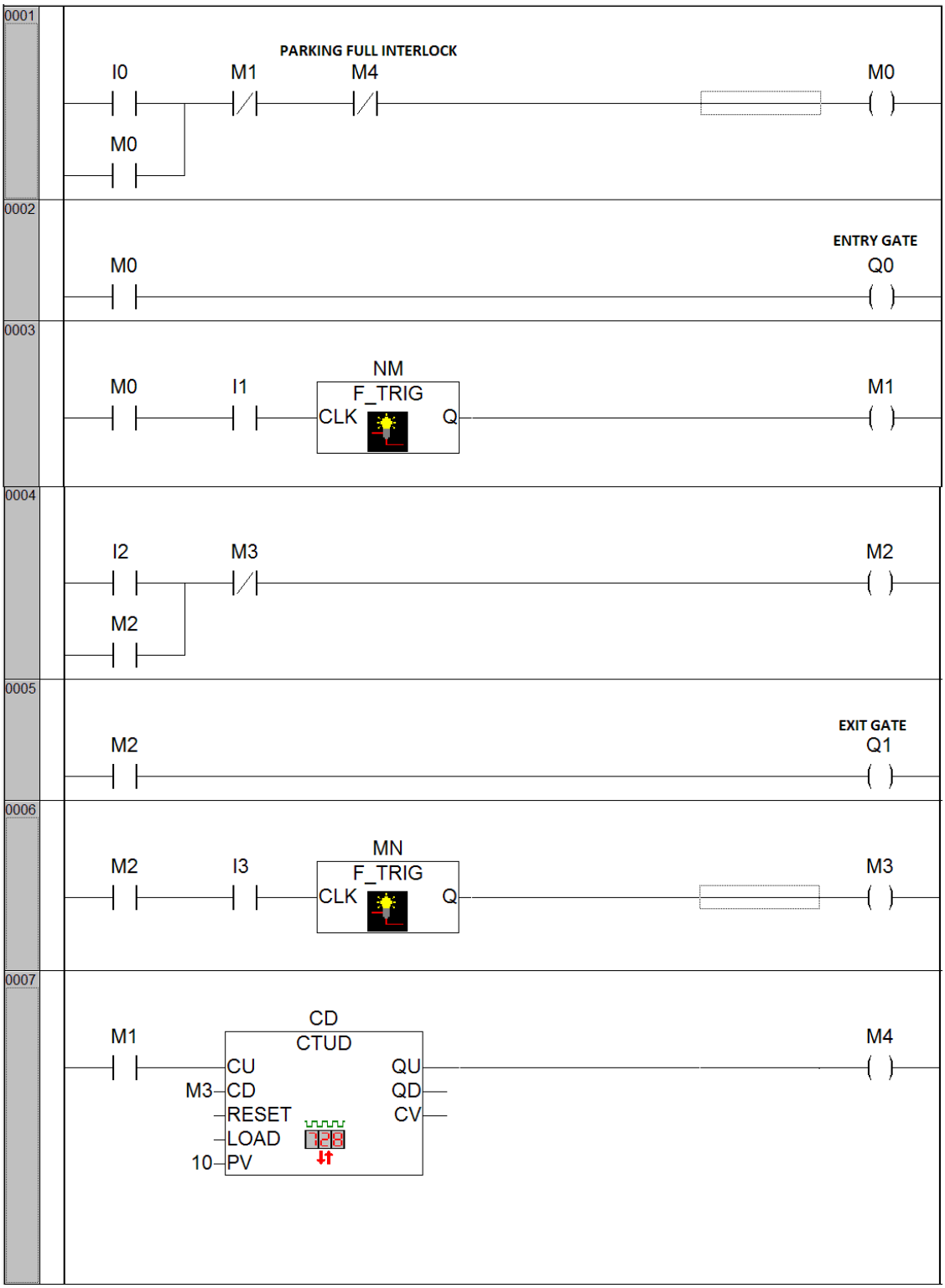
The same example can be made using CTD as shown below.




## EX46: CAR PARKING AREA

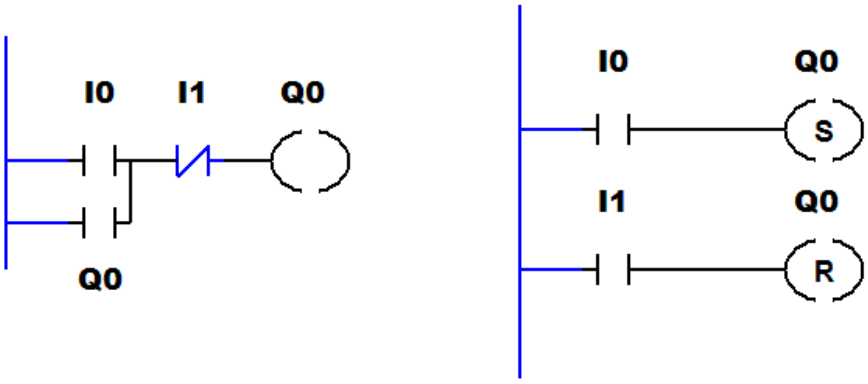



There is a car parking area where we have entry gate Q0 and exit gate Q1. On entry we have two sensors, I0 outside and I1 inside. On exit also we have two sensors I2 and I3. When car comes in front of I0 sensor, gate Q0 is open. When car reaches to I1 sensor, nothing will happen. When car crosses sensor I1 then gate Q0 will be closed and it would be count up. Inside parking area when car reaches to sensor I2, exit gate Q1 is open. When car crosses I3 sensor then Q1 is closed and it would be count down on display to show the present number of cars inside parking area. Maximum 10 cars will be parked. When parking is full and 11$^{th}$ car comes in front of entry sensor I0, entry gate should not get open. Follow the steps and make program for this.


0001
PARKING FULL INTERLOCK
I0
M1
M4
M0
M0
0002
ENTRY GATE
M0
Q0
0003
NM
M0
I1
F_TRIG
CLK
Q
M1
0004
I2
M3
M2
M2
0005
EXIT GATE
M2
Q1
0006
MN
M2
I3
F_TRIG
CLK
Q
M3
0007
CD
CTUD
M1
CU
QU
M4
M3
CD
QD
RESET
CV
LOAD
10
PV

Now we shall go through some tricky examples. Before that one more instruction we shall learn.

**SET/ RESET: it is the replacement of Latch and latch break.**




Here In this picture we have latched coil and NC break left side and SET/ RESET right side. We shall get the same result when we execute these two programs. When we press I0 Q0 is latched and is continuing on. When we have a latch, it is very important to put one NC to break the latch so we have put I1 NC. When I 1 is on/ pressed, Q0 gets off.

In SET/ RESET program when we press I0, Q0 gets set and is continuing on. Now this Q0 will not get off by NC. It is compulsory to use reset if set has been used in your program. If we have Q0 set then we must have a Q0 reset. When we press I1 in this program, Q0 gets reset/ off.

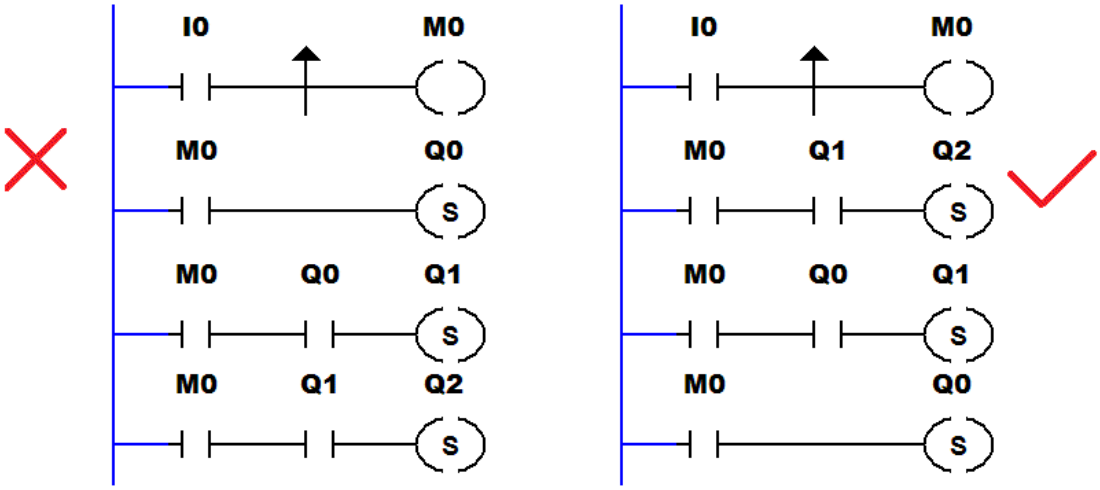
**EX47: output increment and decrement operation by two push buttons.**


| I0 PRESS | I1 PRESS | Q0 | Q1 | Q2 |
|---|---|---|---|---|
| FIRST | 0 | 1 | 0 | 0 |
| SECOND | 0 | 1 | 1 | 0 |
| THIRD | 0 | 1 | 1 | 1 |
| 0 | FIRST | 1 | 1 | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | SECOND | 1 | 0 | 0 |
| 0 | THIRD | 0 | 0 | 0 |

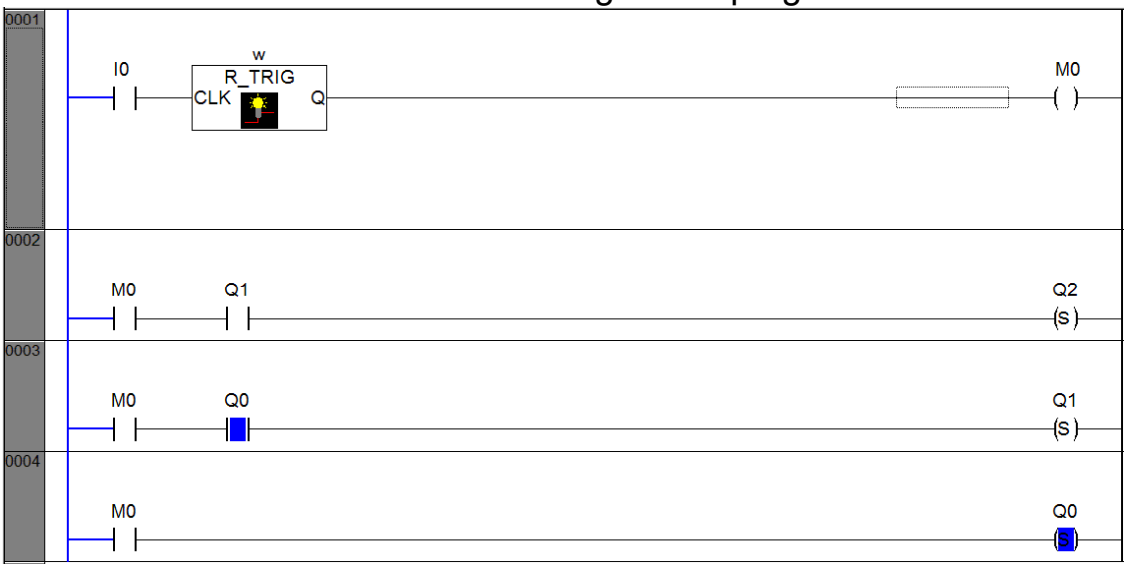
When I0 is pressed first time, Q0 is on. When I0 is pressed second time, Q0 and Q1 are on. When I2 is pressed third time, Q0, Q1 and Q2 all three outputs are on.

When I1 is pressed first time, Q2 is off. When I1 is pressed second time, Q2 and Q1 are off. When I1 is pressed third time, Q2, Q1 and Q0 all three outputs are off. If you make this program using latch it would be so lengthy and complicated. Let us make it in easiest way.

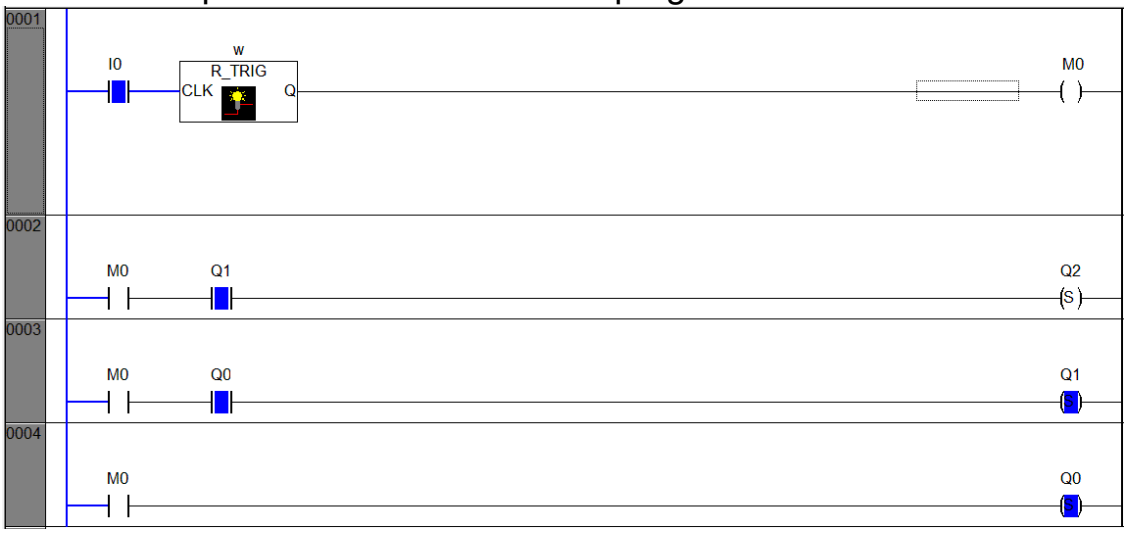



Let us check left side program first. In this program asI0 is pressed first time, M0 gets on. As M0 gets On Q0 gets on in second ladder. As M0 and Q0 get on, Q1 gets on in third ladder. As M0 and Q1 get on, Q2 gets on in forth ladder and scan is over so M0 is off because M0 is a pulse and pulse is on for 1 program scan time. So finally, we see that all three outputs get on at the same time in first press only whereas each output should get on one by one in each press of I0. Left side program is wrong.

Now we shall check right side program.

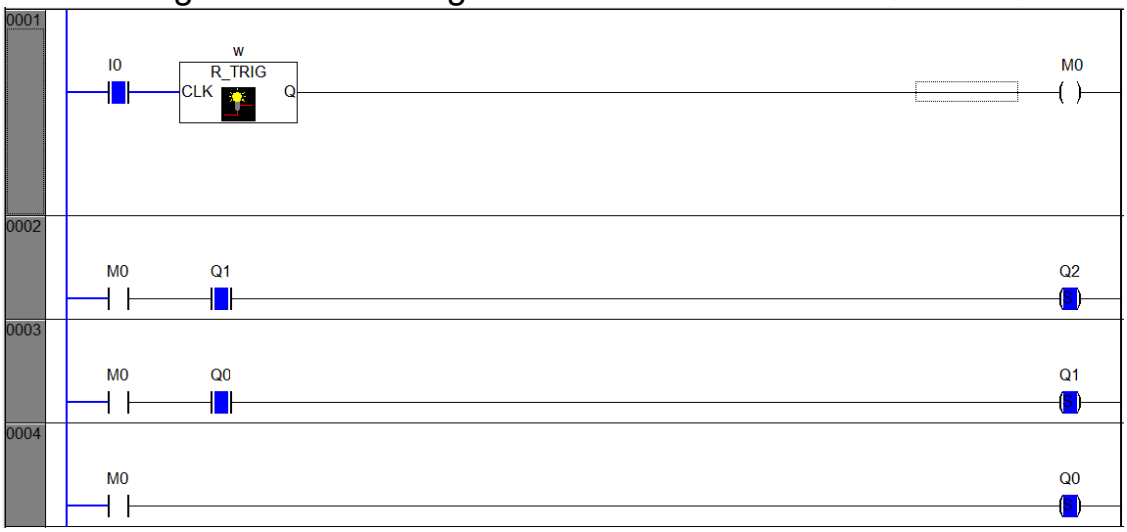



This is first press of I0. In right side program when we press I0, m0 gets high in first ladder. As M0 is on but Q1 is off in second ladder so signal does not reach to Q2 coil resulting Q2 remains off. M0 is on but Q0 is off in third ladder so Q1 also remains off. In third ladder M0 is on and signal reaches to Q0 coil resulting Q0 coil set/ on and scan is over so M0 gets off as it is a pulse. Pulse is on for one program scan time.

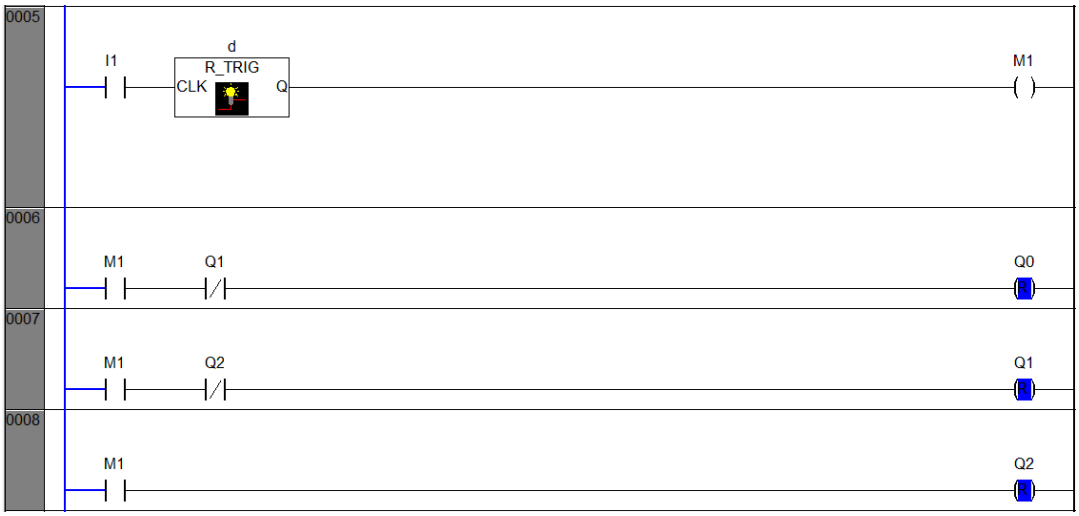



This is second press of I0. Now this is second press of I0. M0 gets on in first ladder. In second ladder M0 is on but Q1 is still off so Q2 remains off. In third ladder M0 is on and Q0 is also on (in last scan/ first press of I0) so signal reaches to Q1 resulting it set/ on. In fourth ladder things remains the same

and scan gets over resulting M0 off. So now we have Q0 and Q1 on.

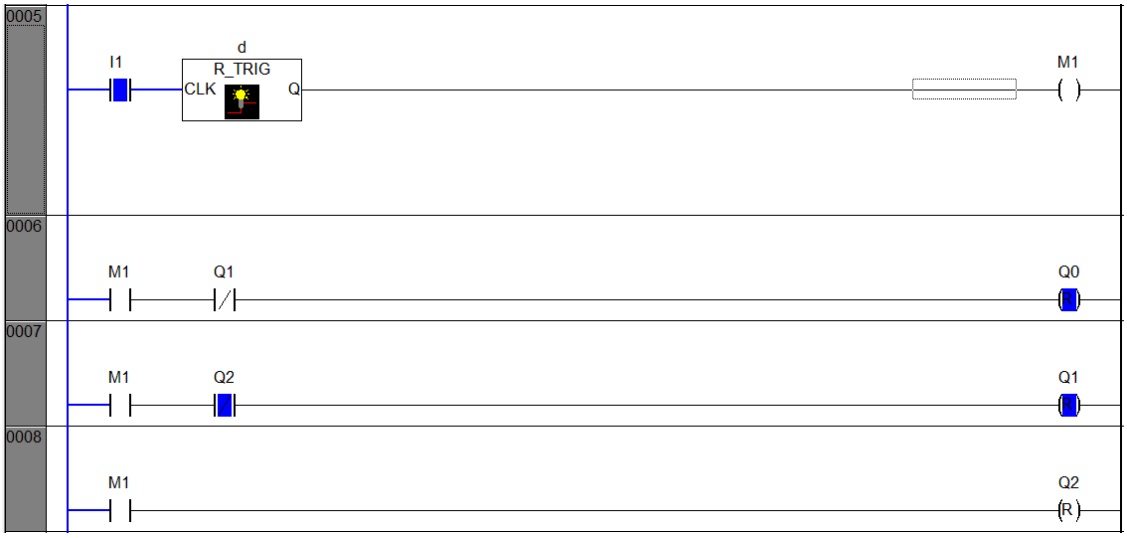



This is third press of I0 now. M0 gets on in first ladder. In second ladder M0 is on and now Q1 is also on so signal reaches to Q2 resulting it set/ on. In third and fourth ladders thing remains the same and scan gets over resulting M0 off automatically as M0 is a pulse.so finally in each press of I0, Output gets on one by one. It happened because of pulse/ scan. If you don't take pulse then all outputs will get on in first press of I0 only.

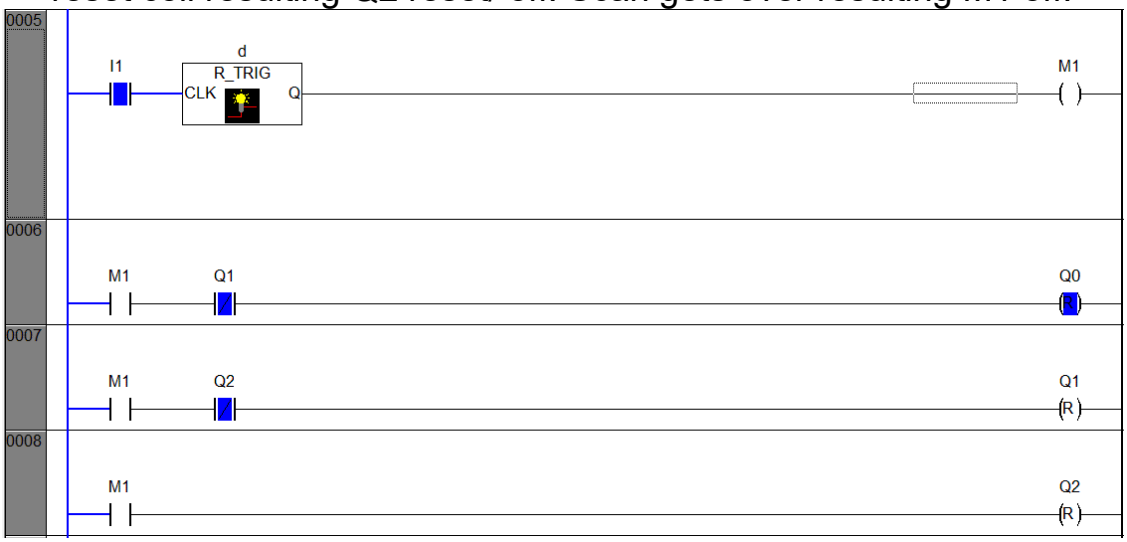



This is the program to off the outputs by pressing I1. This is the status after three press of I0. As all three outputs are on Q1 breaks sixth ladder and Q2

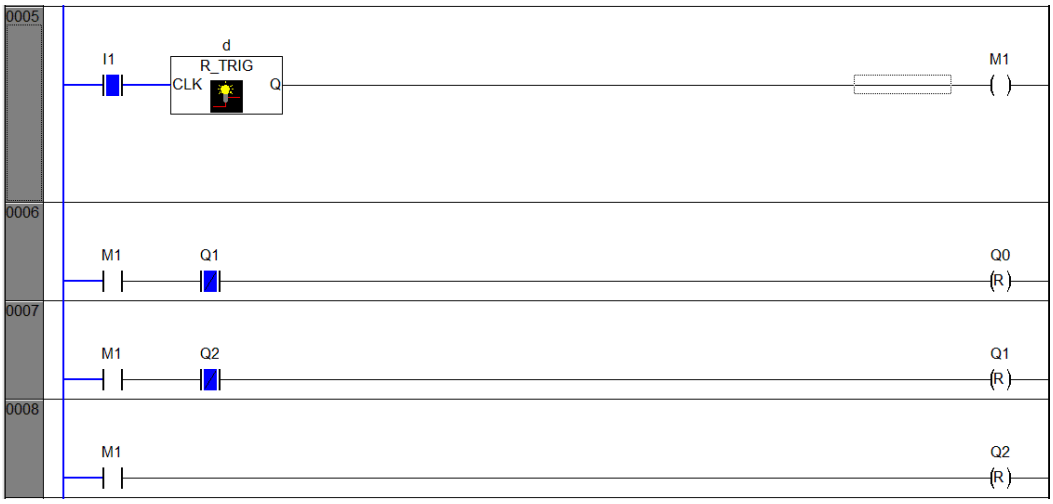
breaks seventh ladder.



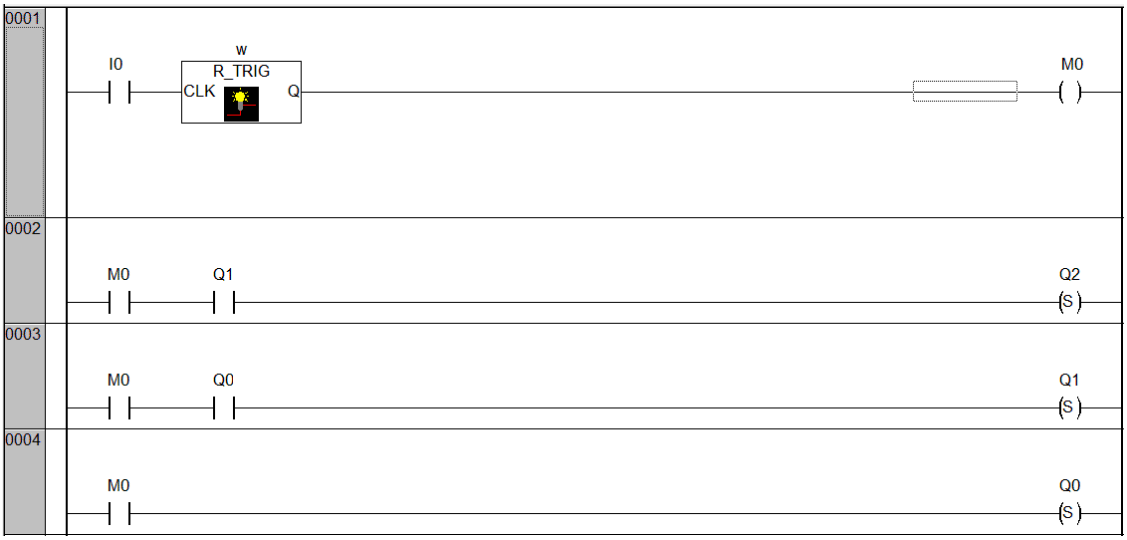

This is first press of I1. M1 gets on in fifth ladder. In sixth ladder M1 is high but Q1 NC has been breaking path as Q1 is on so the signal does not reach to Q0 reset coil resulting Q0 remains on. In seventh ladder M1 is on but Q2 NC breaks the path as Q2 is on so signal does not reach to Q1 reset coil resulting Q1 remains on. In eighth ladder M1 is on. Signal reaches to Q2 reset coil resulting Q2 reset/ off. Scan gets over resulting M1 off.

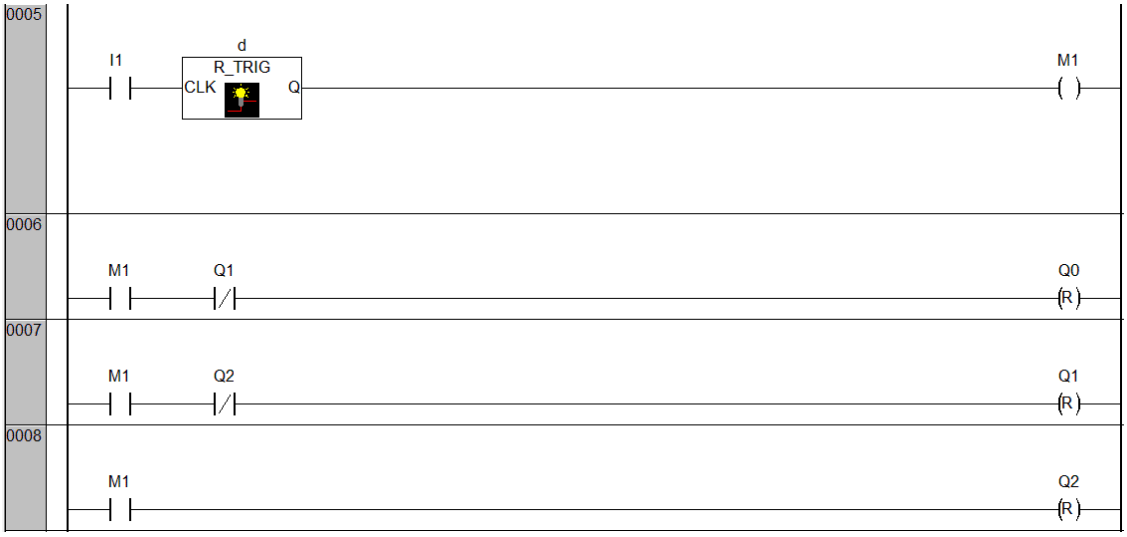



This is second press of I1. M1 gets on in fifth ladder. In sixth ladder M1 is high but Q1 NC has been breaking path as Q1 is on so the signal does not reach to Q0 reset coil resulting Q0 remains on. In seventh ladder M1 is on and now Q2 NC breaks is gone as Q2 got off in last scan so signal reaches to Q1 reset coil resulting Q1 reset/ off. In eighth ladder things remain the same. Scan gets over resulting M1 off. So now Q2 and Q1 are off after two press of I1.



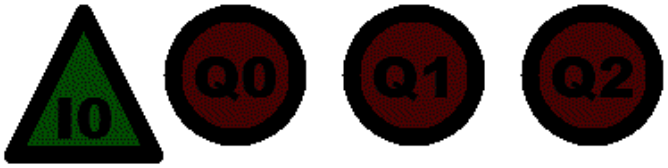

This is third press of I1. M1 gets on in fifth ladder. In sixth ladder M1 is high and now Q1 break is gone because Q1 got off in last scan so signal reaches to Q0 reset coil resulting Q0 reset/ off. In seventh and eighth ladder things remain the same. Scan gets over resulting M1 off. So now Q2, Q1 and Q0 all three outputs are off after three press of I1.





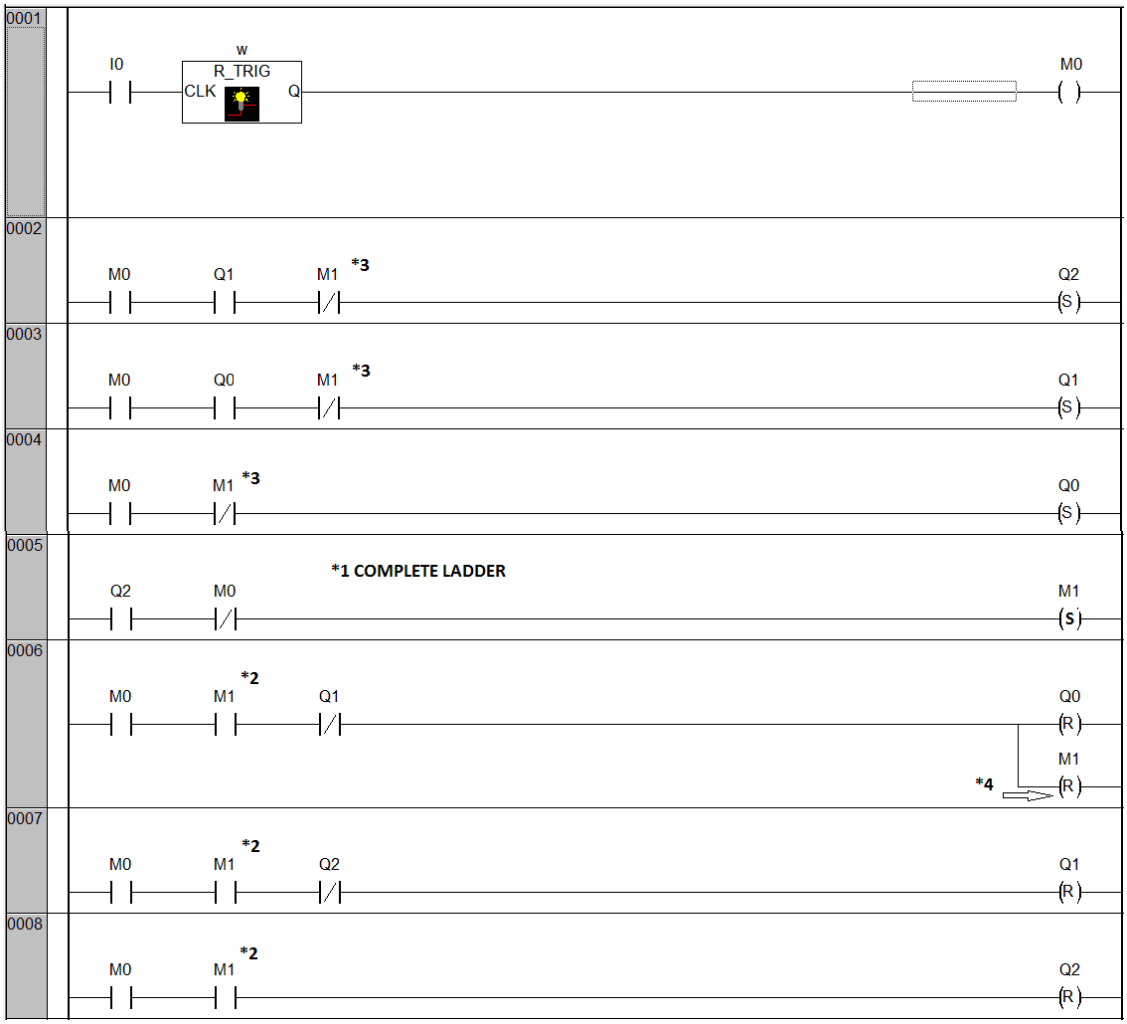

Complete program.

**EX48: output increment and decrement operation by single push button.**




In this example, one by one on and one by one off of output will be operated by each press of single push button I0.

| I0 PRESS | Q0 | Q1 | Q2 |
|---|---|---|---|
| FIRST | 1 | 0 | 0 |
| SECOND | 1 | 1 | 0 |
| THIRD | 1 | 1 | 1 |
| FOURTH | 1 | 1 | 0 |
| FIFTH | 1 | 0 | 0 |
| SIXTH | 0 | 0 | 0 |




This program is very similar to previous program where we had two push buttons. In this example we have one push button. Remember four points to convert previous push buttons program into single push button program.

*1: This middle ladder is required to generate a confirmation signal/ M1 that all outputs set work is over.

*2: NO contact of middle ladder confirmation signal M1 will be put to all reset ladders to start reset work.

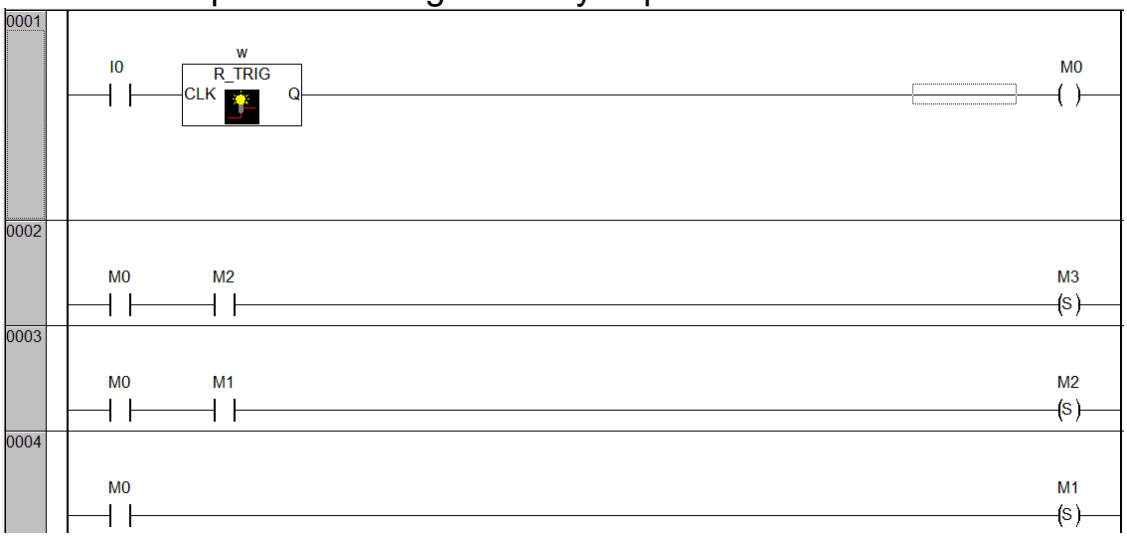
*3: NC contact of middle ladder confirmation signal M1 will be put to all set ladders to start set work.

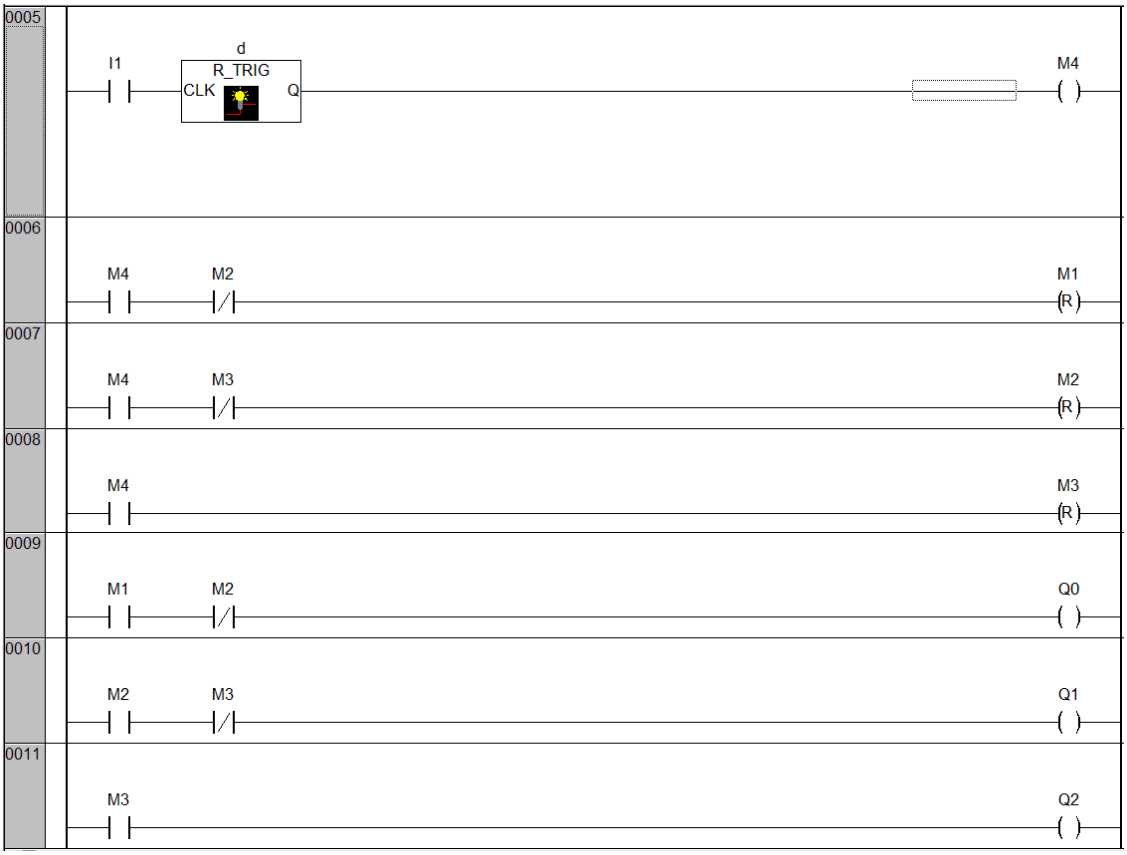
*4: Middle ladder confirmation signal M1 should also get reset lastly.

**EX49: Shift right and shift left operation of output by two push buttons.**


| I0 PRESS | I1 PRESS | Q0 | Q1 | Q2 |
|---|---|---|---|---|
| FIRST | 0 | 1 | 0 | 0 |
| SECOND | 0 | 0 | 1 | 0 |
| THIRD | 0 | 0 | 0 | 1 |
| 0 | FIRST | 0 | 1 | 0 |
| 0 | SECOND | 1 | 0 | 0 |
| 0 | THIRD | 0 | 0 | 0 |

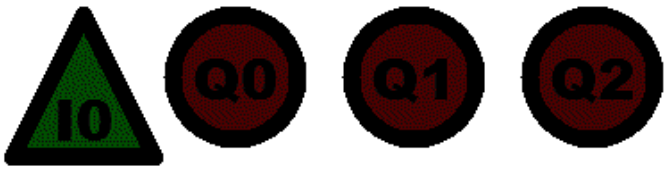
In this example we have two push buttons and three lamps. Lamps will get on one by one on (one lamp on at once) on each press of I0 push button. Lamps will get off one by one on each press of I1 push button. From I0 press output will shift right and by I1 press it will shift left.






In this program we have generated three flags M1, M2 and M3 on & off by push buttons I0 and I1 as per truth table conditions. As we got signal for each press of input, from that signal NO & NC we operated outputs. For output NO and for output off NC has been used (see ladder 9, 10 and 11).

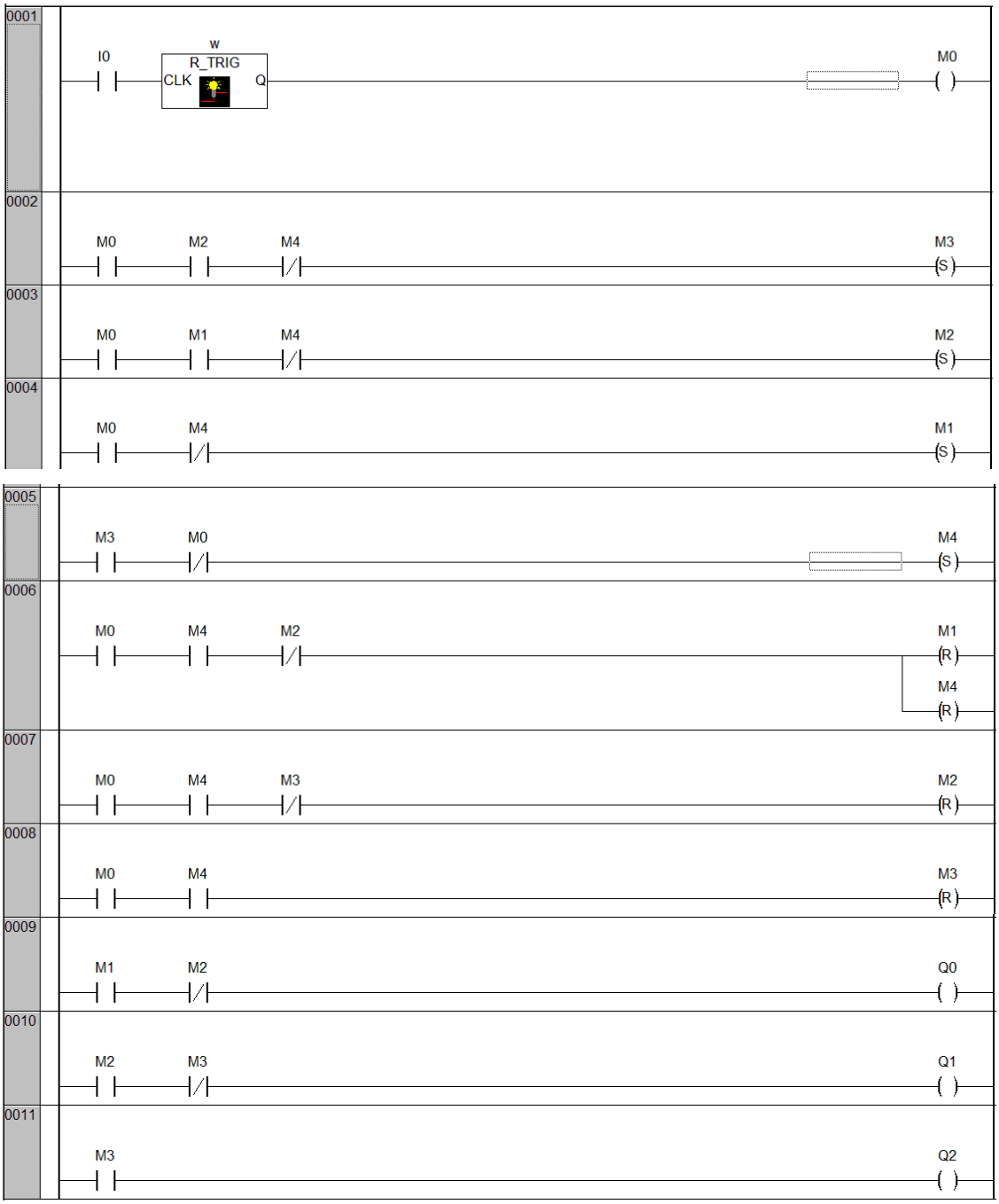
**EX50: Shift right and shift left operation of output by single push button.**




Previous example we have to make using single push button I0.

| I0 PRESS | Q0 | Q1 | Q2 |
|---|---|---|---|
| FIRST | 1 | 0 | 0 |
| SECOND | 0 | 1 | 0 |
| THIRD | 0 | 0 | 1 |
| FOURTH | 0 | 1 | 0 |

| | | | |
|---|---|---|---|
| **FIFTH** | **1** | **0** | **0** |
| **SIXTH** | **0** | **0** | **0** |

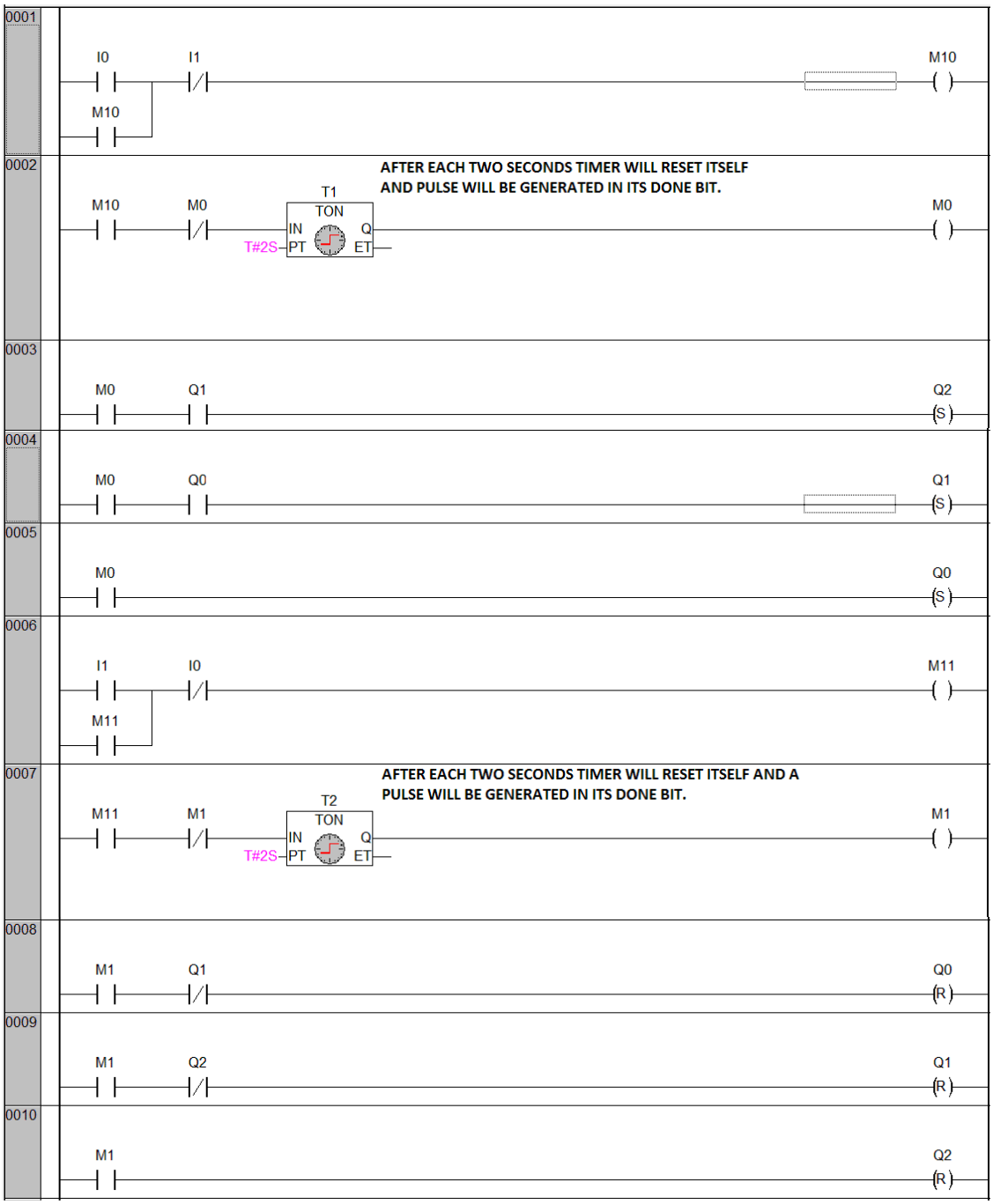



**EX51: Increment and decrement operation of output using two push buttons and two timers.**


When I0 push button is pressed, outputs will be incremented one by one on each 2 seconds interval. When I1 push button is pressed, outputs will be decremented one by one on each 2 seconds interval.

| I0 | TIMER 1 | I1 | TIMER 2 | Q0 | Q1 | Q2 |
|---|---|---|---|---|---|---|
| 1 | TIMER STARTS | 0 | | 1 | 0 | 0 |
| 0 | AFTER FIRST 2 SEC INTERVAL | 0 | | 1 | 1 | 0 |
| 0 | AFTER SECOND 2 SEC INTERVAL | 0 | | 1 | 1 | 1 |
| 0 | | 1 | TIMER STARTS | 1 | 1 | 0 |
| | | 0 | AFTER FIRST 2 SEC INTERVAL | 1 | 0 | 0 |
| | | 0 | AFTER SECOND 2 SEC INTERVAL | 0 | 0 | 0 |

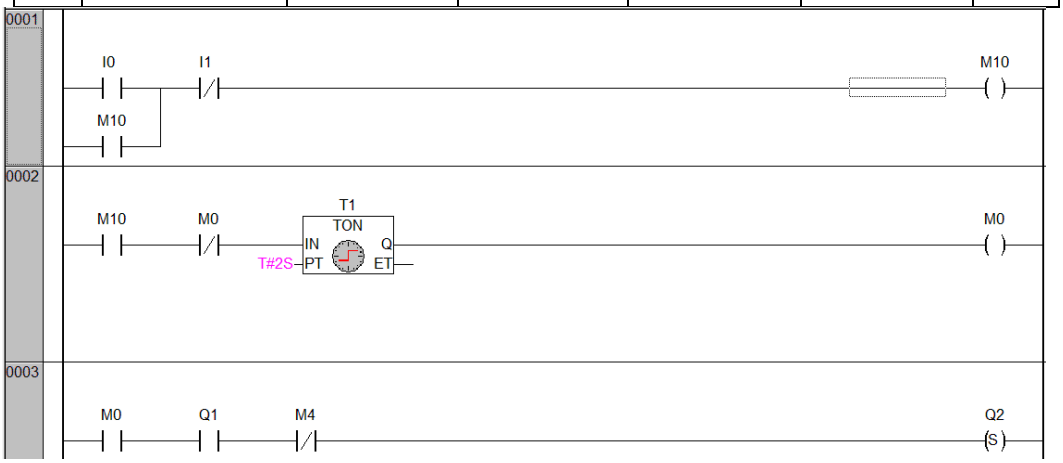
What we did using push button every time to operate outputs, the same we have to do using timer done bit.

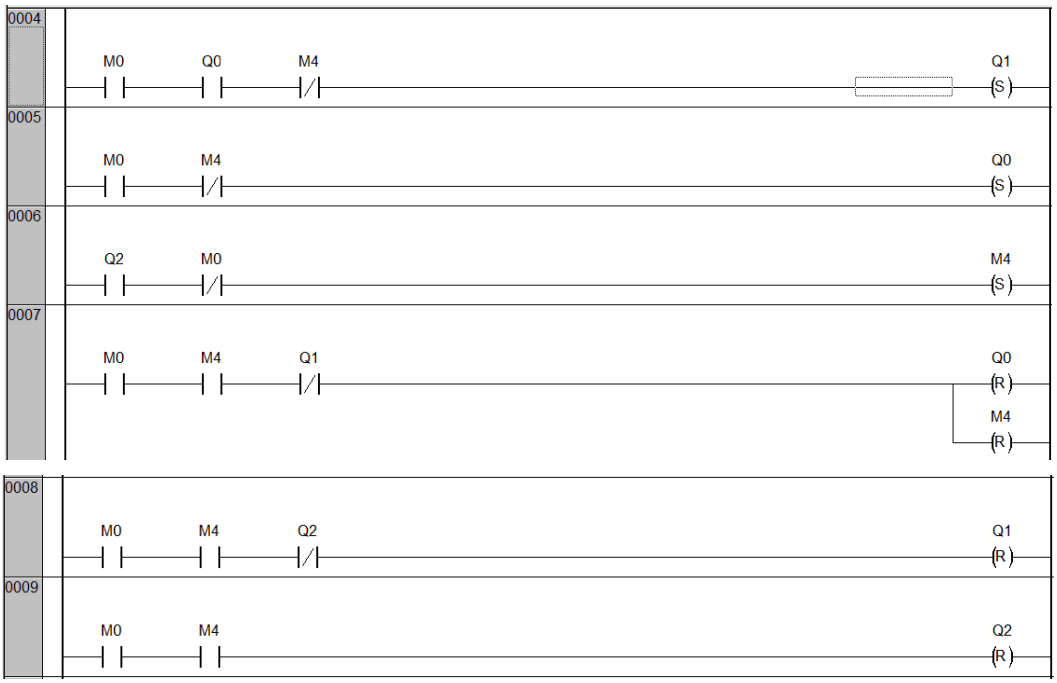



## EX52: Increment and decrement operation of output using single push button and single timer.

When I0 push button is pressed, outputs will be incremented and decremented one by one on each 2 seconds interval.

| I0 | TIMER 1 | | | Q0 | Q1 | Q2 |
|---|---|---|---|---|---|---|
| 1 | TIMER STARTS | | | 1 | 0 | 0 |
| 0 | AFTER FIRST 2 SEC INTERVAL | | | 1 | 1 | 0 |
| 0 | AFTER SECOND 2 SEC INTERVAL | | | 1 | 1 | 1 |
| 0 | AFTER THIRD 2 SEC INTERVAL | | | 1 | 1 | 0 |
| | AFTER FOURTH 2 SEC INTERVAL | | | 1 | 0 | 0 |
| | AFTER FIFTH 2 SEC INTERVAL | | | 0 | 0 | 0 |



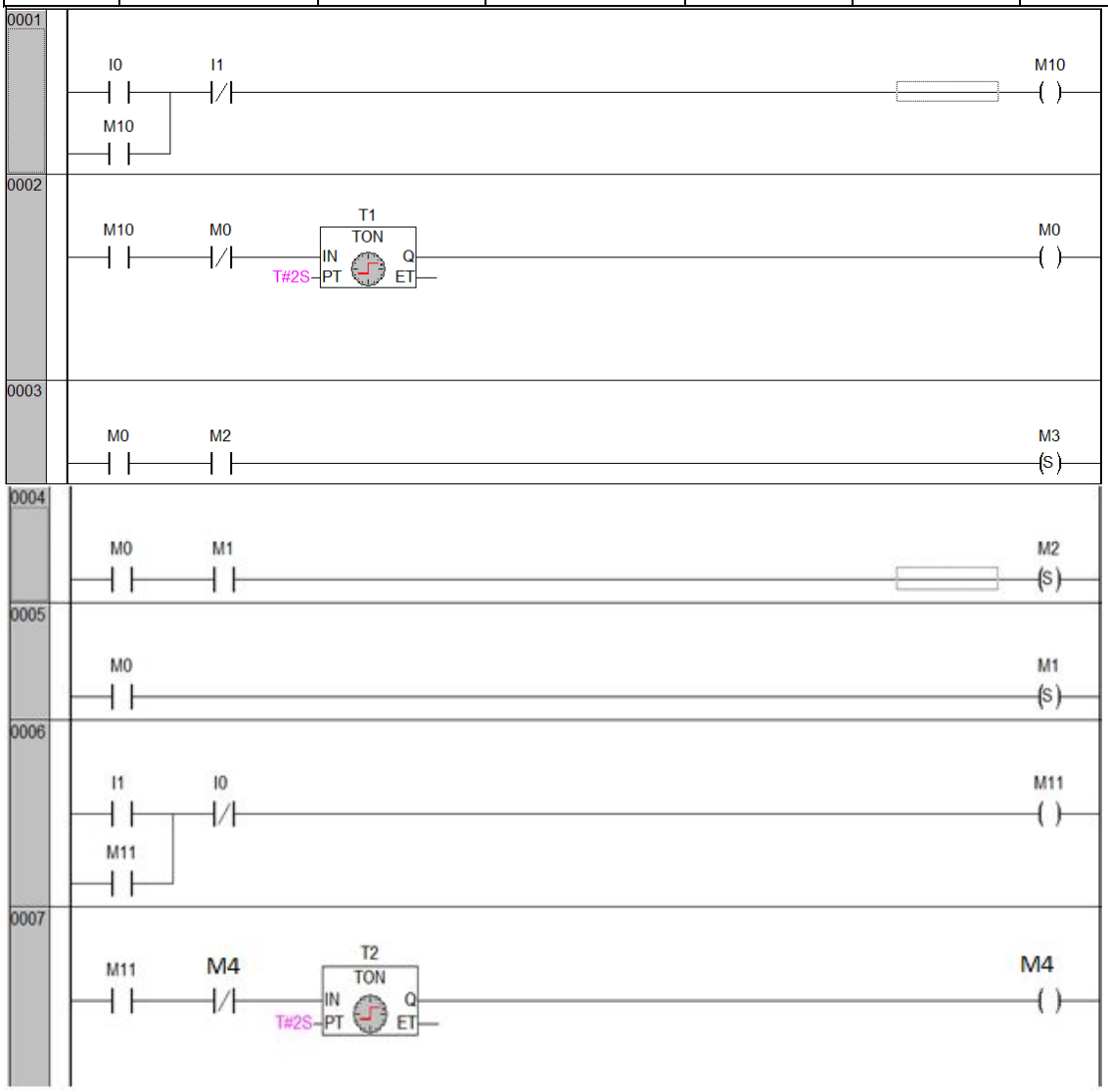


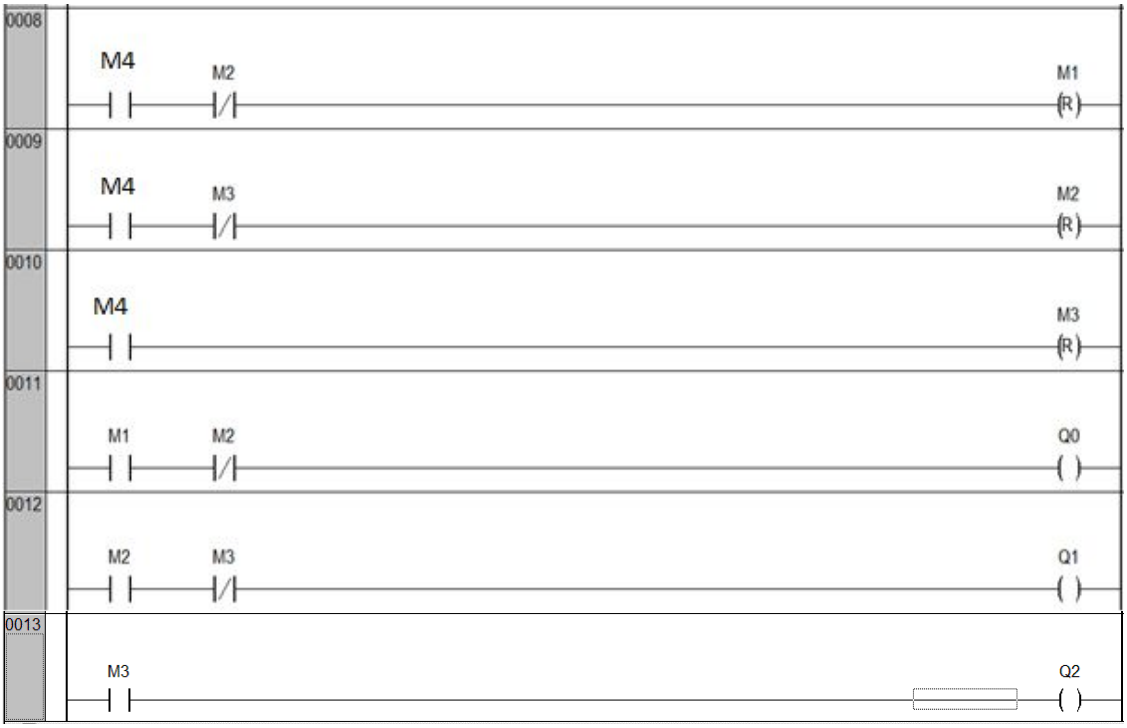

**EX53: Shift right and shift left operation of output using two push buttons and two timers.**


When I0 push button is pressed, outputs will be shifted right one by one on each 2 seconds interval. When I1 push button is pressed, outputs will be shifted left one by one on each 2 seconds interval.

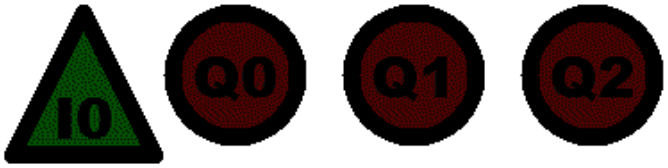
| I0 | TIMER 1 | I1 | TIMER 2 | Q0 | Q1 | Q2 |
|---|---|---|---|---|---|---|
| 1 | TIMER STARTS | 0 | | 1 | 0 | 0 |
| 0 | AFTER FIRST 2 SEC INTERVAL | 0 | | 0 | 1 | 0 |
| 0 | AFTER SECOND 2 SEC INTERVAL | 0 | | 0 | 0 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **0** | | **1** | **TIMER STARTS** | **0** | **1** | **0** |
| | | **0** | **AFTER FIRST 2 SEC INTERVAL** | **1** | **0** | **0** |
| | | **0** | **AFTER SECOND 2 SEC INTERVAL** | **0** | **0** | **0** |





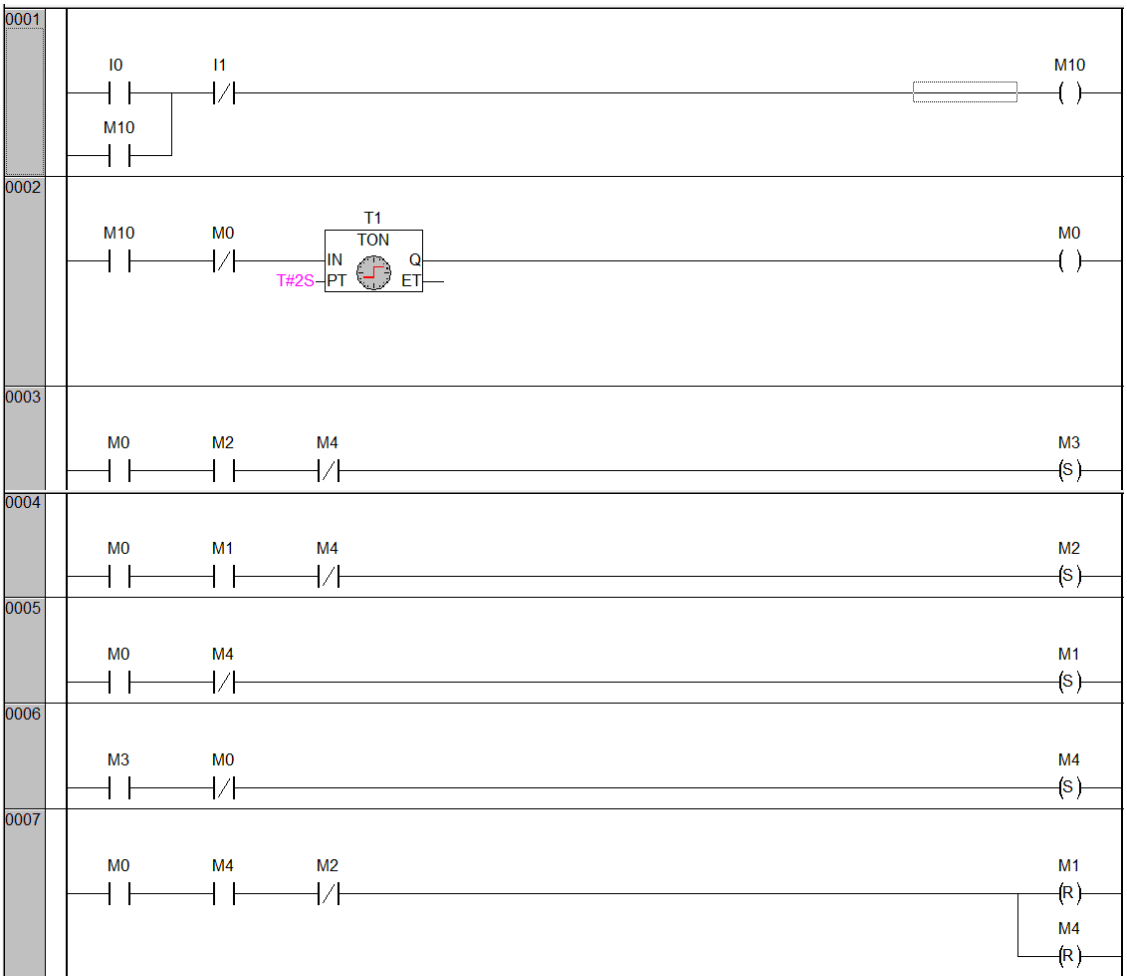

**EX54: Shift right and shift left operation of output using one push button and one timer.**

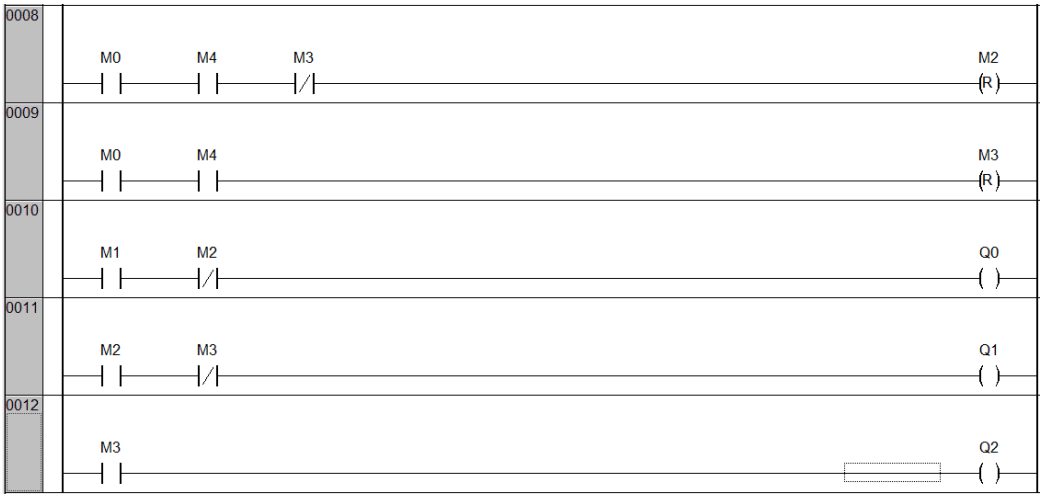



When I0 push button is pressed, outputs will be shifted right and left one by one on each 2 seconds interval.

| I0 | TIMER 1 | | | Q0 | Q1 | Q2 |
|---|---|---|---|---|---|---|
| 1 | TIMER STARTS | | | 1 | 0 | 0 |
| 0 | AFTER FIRST 2 SEC INTERVAL | | | 0 | 1 | 0 |
| 0 | AFTER SECOND 2 SEC INTERVAL | | | 0 | 0 | 1 |
| | | | | | | |

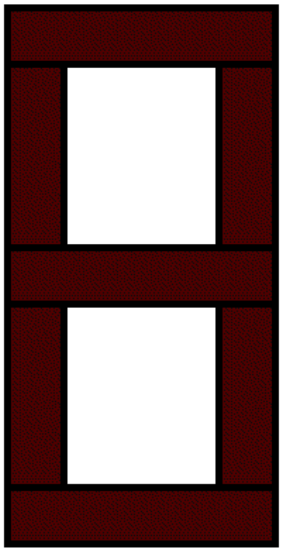
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | AFTER THIRD 2 SEC INTERVAL | | | 0 | 1 | 0 |
| | AFTER FOURTH 2 SEC INTERVAL | | | 1 | 0 | 0 |
| | AFTER FIFTH 2 SEC INTERVAL | | | 0 | 0 | 0 |



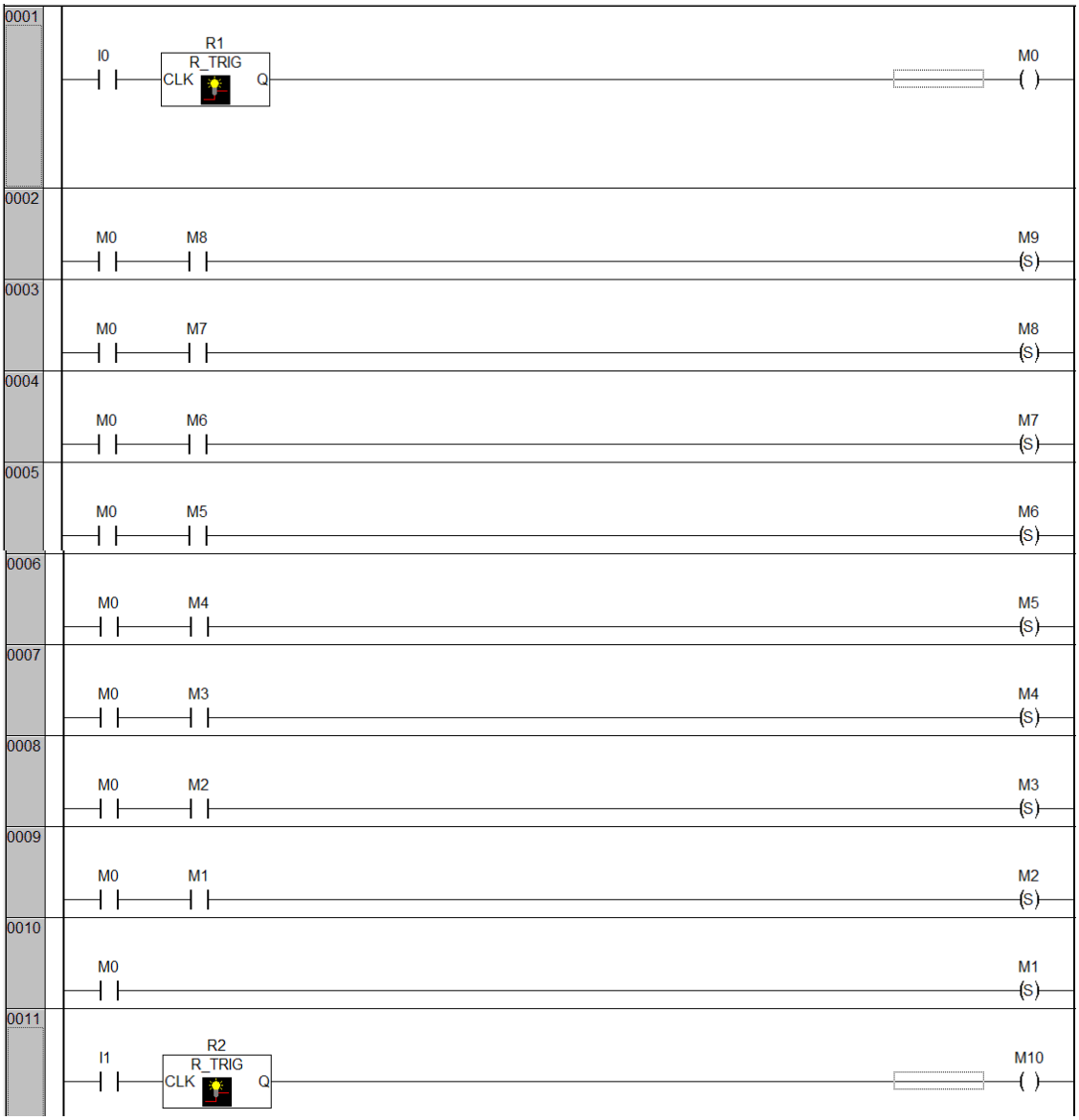


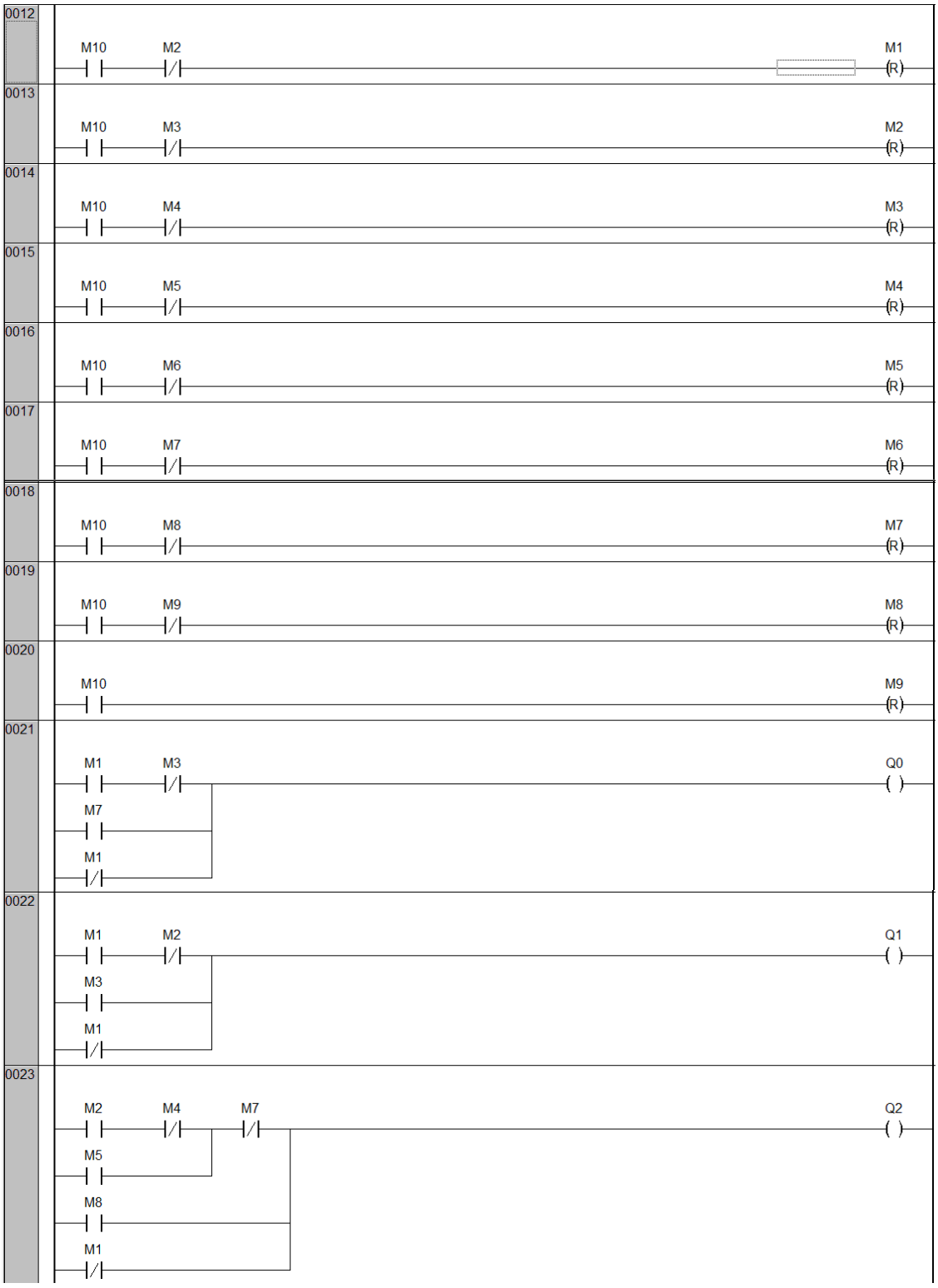

## EX55: SEVEN SEGMENT DISPLAY USING TWO PUSH BUTTONS.

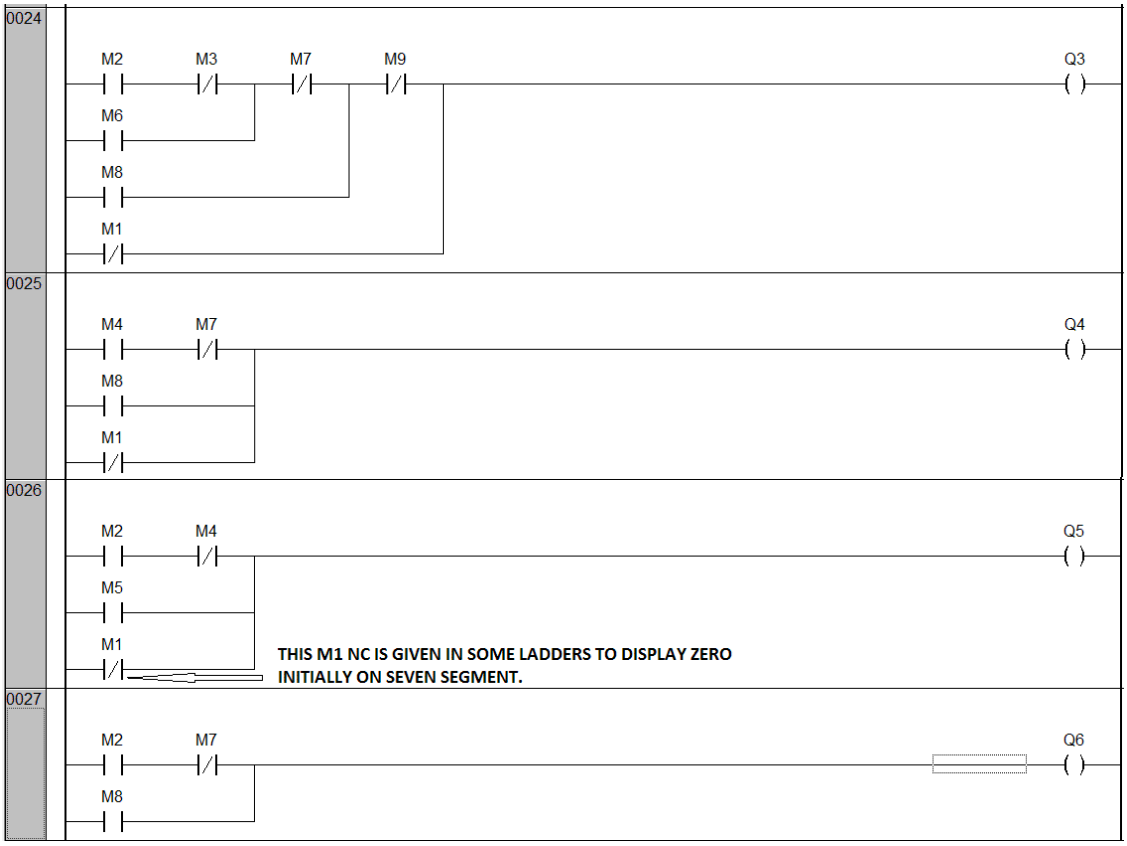
| SIGNAL ↓ | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 |
|---|---|---|---|---|---|---|---|
| M1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| M2 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| M3 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| M4 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| M5 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| M6 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| M7 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| M8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| M9 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |



To display number from 1 to 9, I0 has to be pressed 9 times. To display number 9 to 1, I1 has to be pressed nine times. For nine press of I0 let us generate nine flags and from each flag respective outputs will be operated as shown in truth table. As system is powered on, zero should be displays i. e. Q0, Q1, Q2, Q3, Q4 and Q5 on.

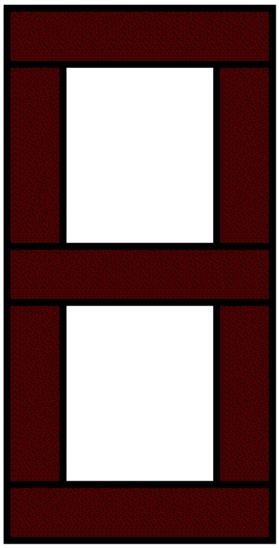

0001
I0
R1
R_TRIG
CLK
Q
M0
0002
M0
M8
M9
(S)
0003
M0
M7
M8
(S)
0004
M0
M6
M7
(S)
0005
M0
M5
M6
(S)
0006
M0
M4
M5
(S)
0007
M0
M3
M4
(S)
0008
M0
M2
M3
(S)
0009
M0
M1
M2
(S)
0010
M0
M1
(S)
0011
I1
R2
R_TRIG
CLK
Q
M10

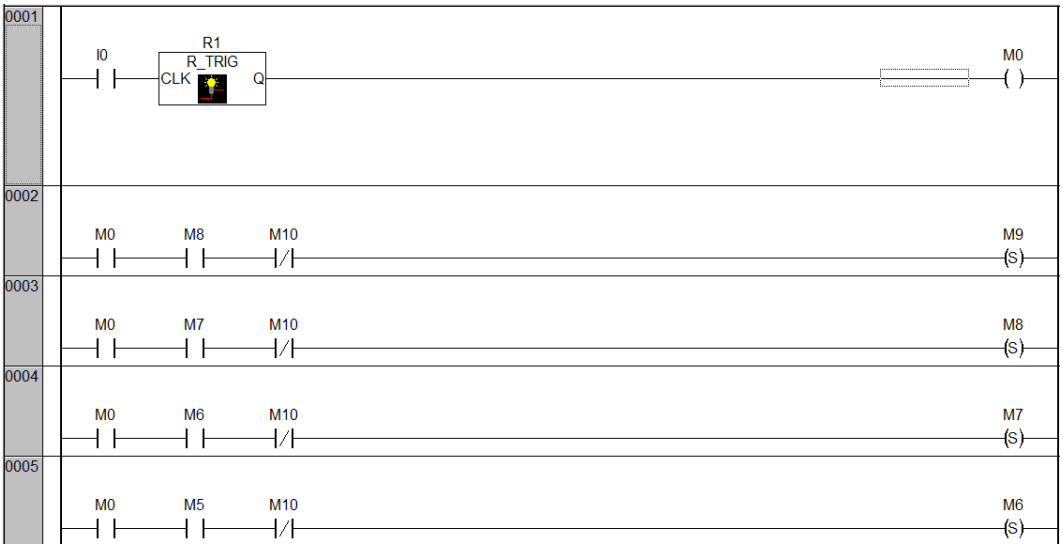

0012
M10
M2
M1
(R)
0013
M10
M3
M2
(R)
0014
M10
M4
M3
(R)
0015
M10
M5
M4
(R)
0016
M10
M6
M5
(R)
0017
M10
M7
M6
(R)
0018
M10
M8
M7
(R)
0019
M10
M9
M8
(R)
0020
M10
M9
(R)
0021
M1
M3
Q0
( )
M7
M1
0022
M1
M2
Q1
( )
M3
M1
0023
M2
M4
M7
Q2
( )
M5
M8
M1

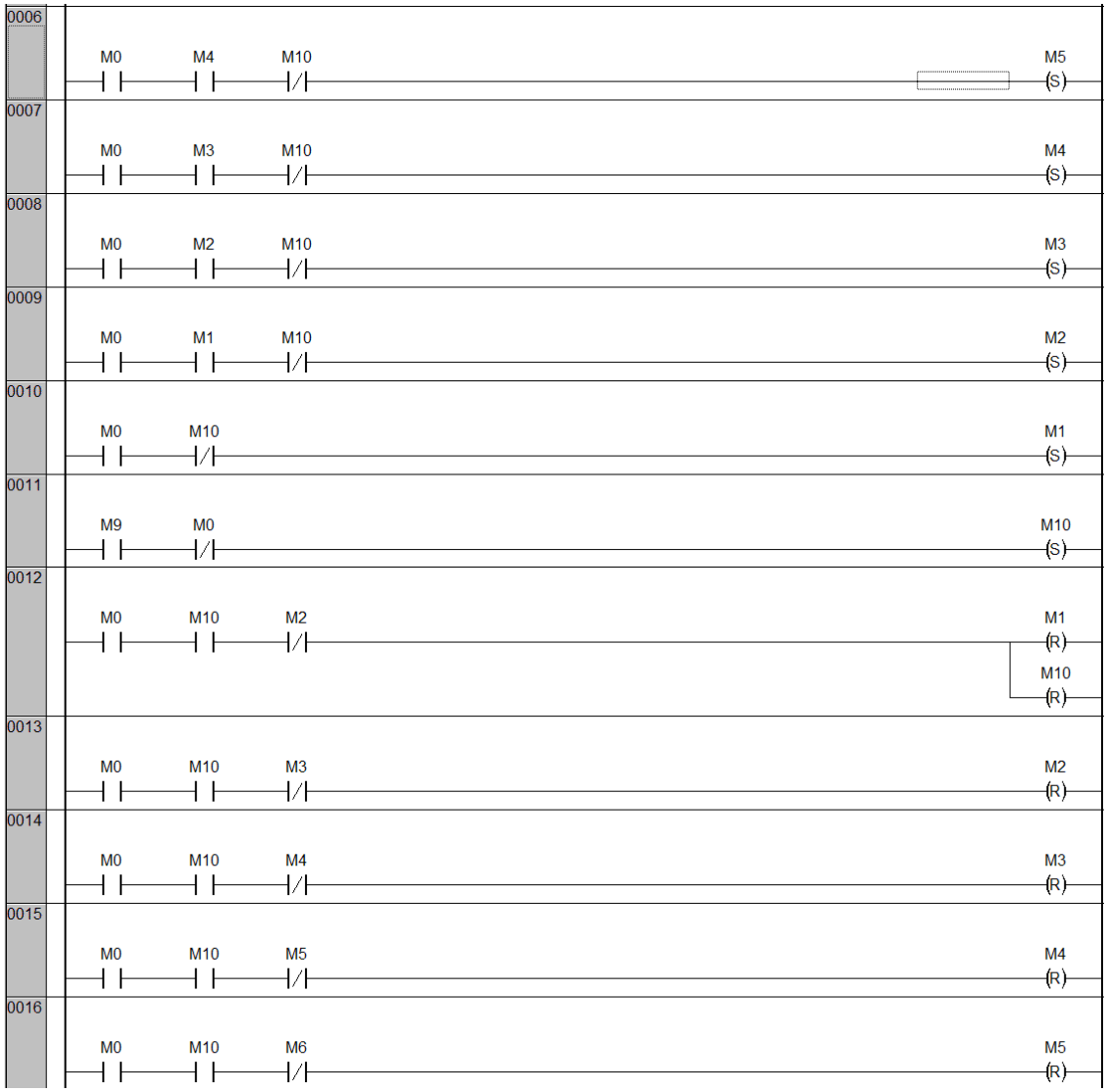


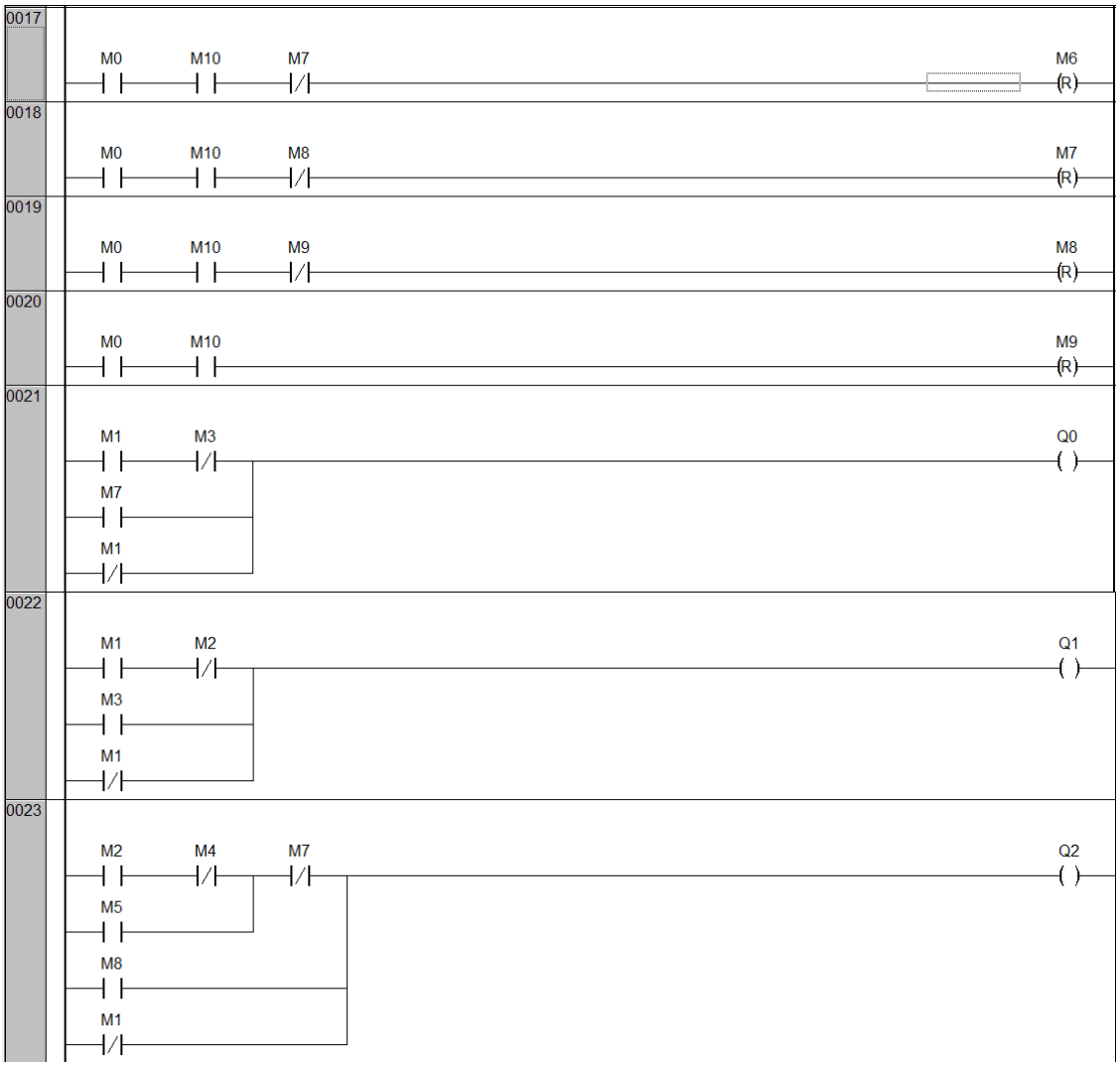

## EX56: SEVEN SEGMENT DISPLAY USING ONE PUSH BUTTON.

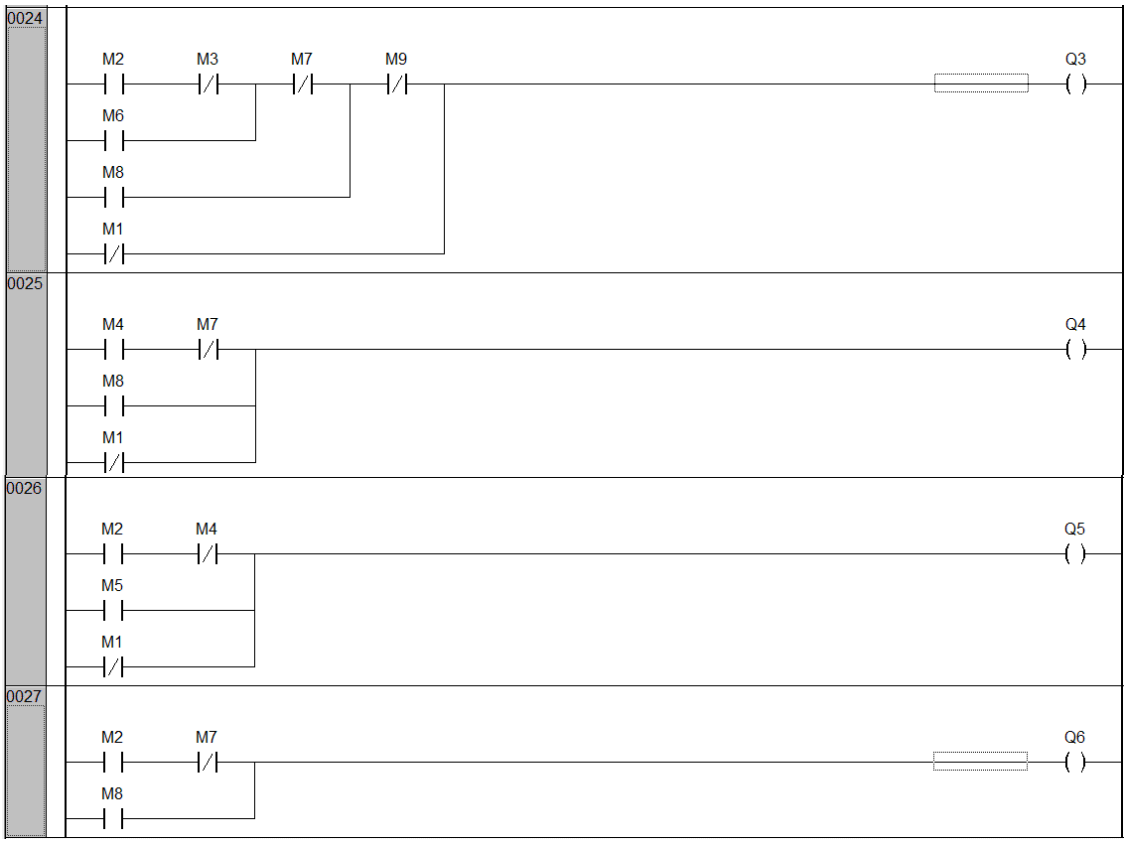
| SIGNAL ↓ | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 |
|---|---|---|---|---|---|---|---|
| M1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| M2 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| M3 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| M4 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| M5 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| M6 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| M7 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| M8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| M9 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |





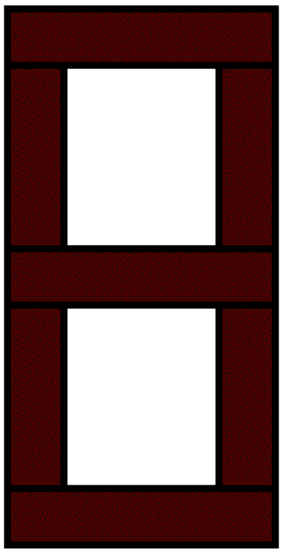

0006
M0
M4
M10
M5
(S)
0007
M0
M3
M10
M4
(S)
0008
M0
M2
M10
M3
(S)
0009
M0
M1
M10
M2
(S)
0010
M0
M10
M1
(S)
0011
M9
M0
M10
(S)
0012
M0
M10
M2
M1
(R)
M10
(R)
0013
M0
M10
M3
M2
(R)
0014
M0
M10
M4
M3
(R)
0015
M0
M10
M5
M4
(R)
0016
M0
M10
M6
M5
(R)


0017
M0
M10
M7
M6
(R)
0018
M0
M10
M8
M7
(R)
0019
M0
M10
M9
M8
(R)
0020
M0
M10
M9
(R)
0021
M1
M3
Q0
( )
M7
M1
0022
M1
M2
Q1
( )
M3
M1
0023
M2
M4
M7
Q2
( )
M5
M8
M1

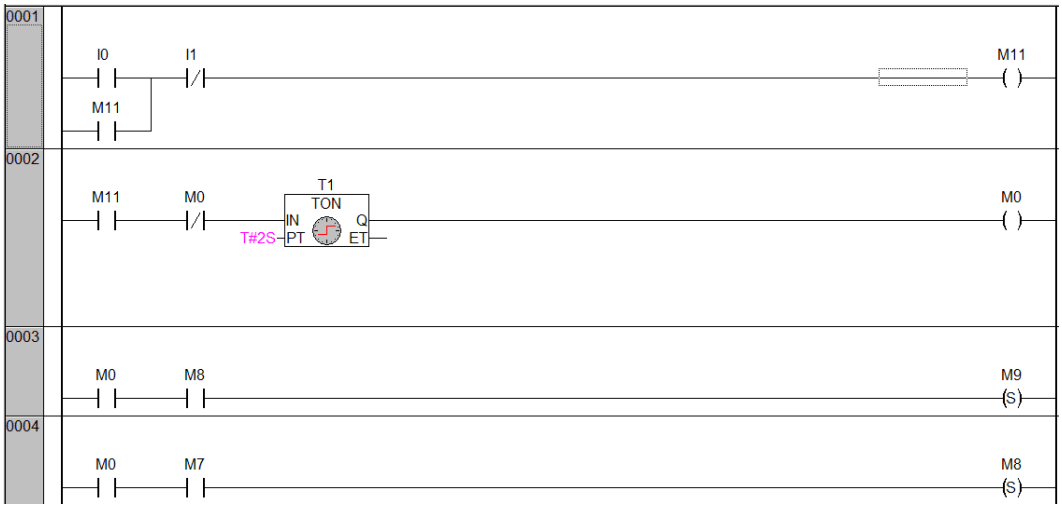


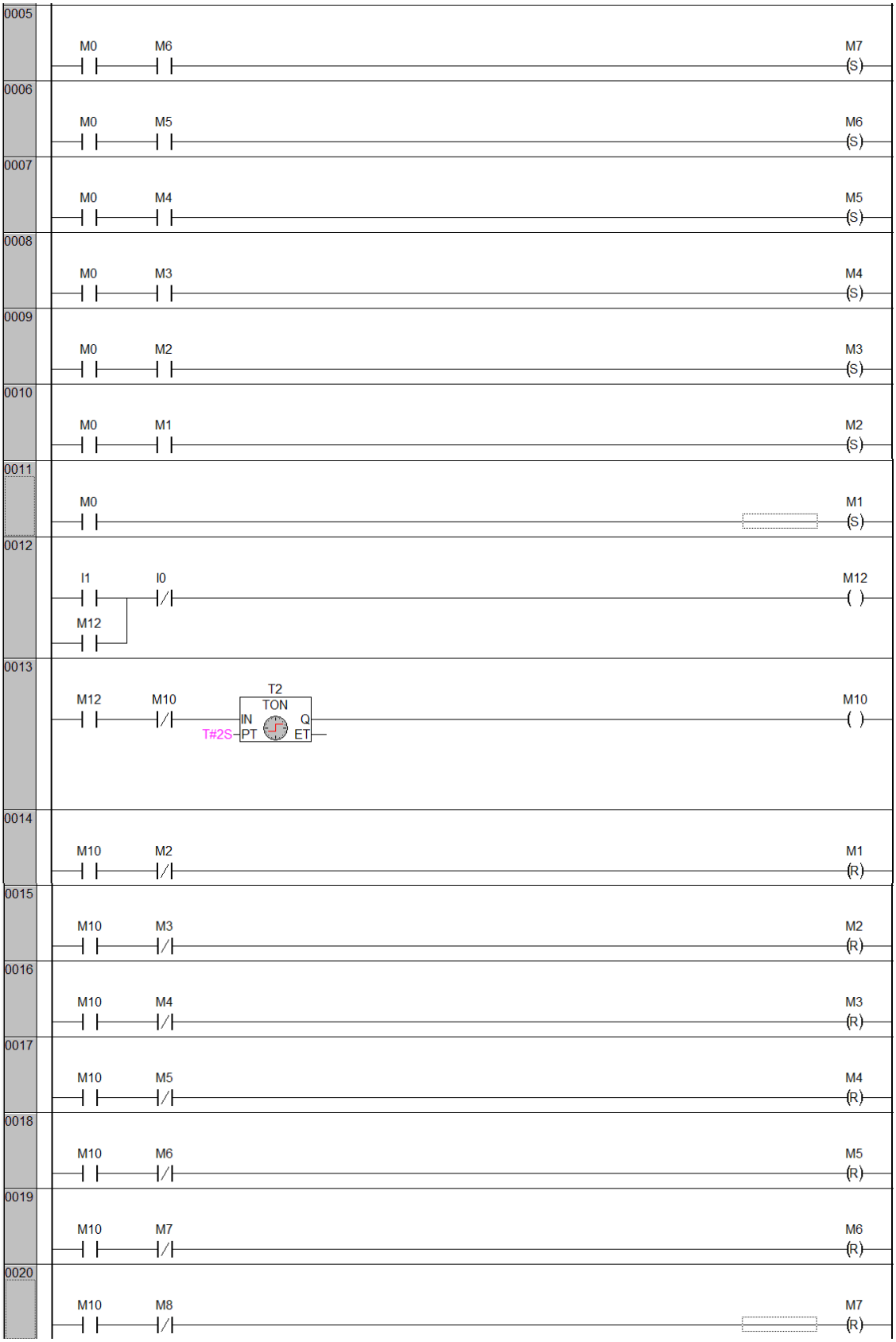

## EX57: SEVEN SEGMENT DISPLAY USING TWO PUSH BUTTONS AND TWO TIMERS.

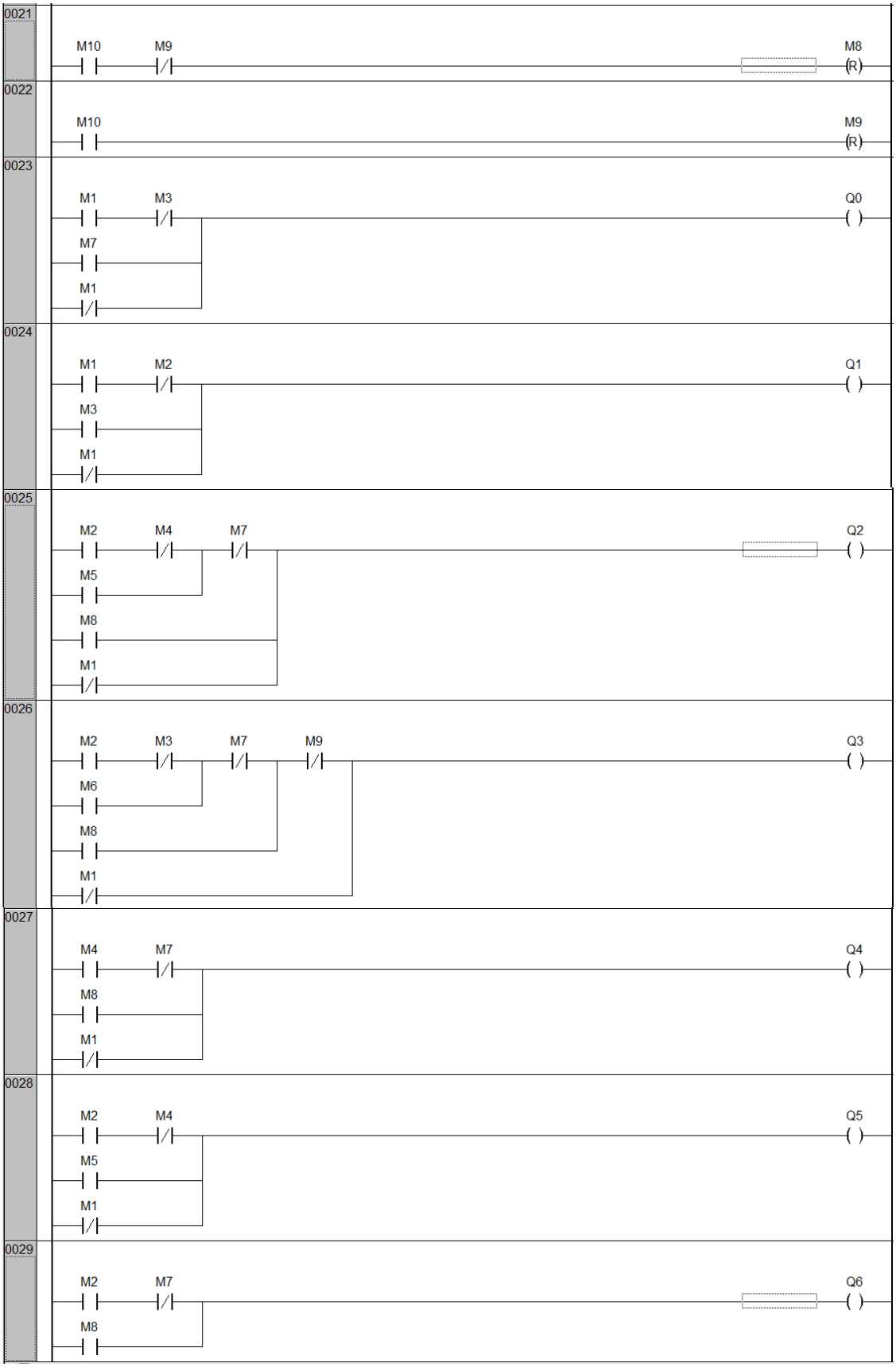
| SIGNAL ↓ | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 |
|---|---|---|---|---|---|---|---|
| M1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| M2 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| M3 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| M4 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| M5 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| M6 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| M7 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| M8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| M9 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |



As I0 push button is pressed, number display will change from 0 to 9 on each 2 seconds interval. When I1 push button is pressed, Number display from 9 to 0 will be displayed on each 2 seconds interval.

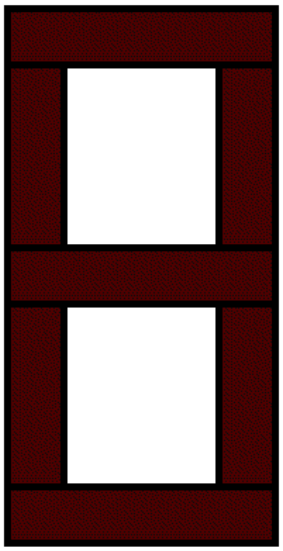



0005
M0
M6
M7
(S)
0006
M0
M5
M6
(S)
0007
M0
M4
M5
(S)
0008
M0
M3
M4
(S)
0009
M0
M2
M3
(S)
0010
M0
M1
M2
(S)
0011
M0
M1
(S)
0012
I1
I0
M12
( )
M12
0013
M12
M10
T2
TON
IN
Q
T#2S
PT
ET
M10
( )
0014
M10
M2
M1
(R)
0015
M10
M3
M2
(R)
0016
M10
M4
M3
(R)
0017
M10
M5
M4
(R)
0018
M10
M6
M5
(R)
0019
M10
M7
M6
(R)
0020
M10
M8
M7
(R)

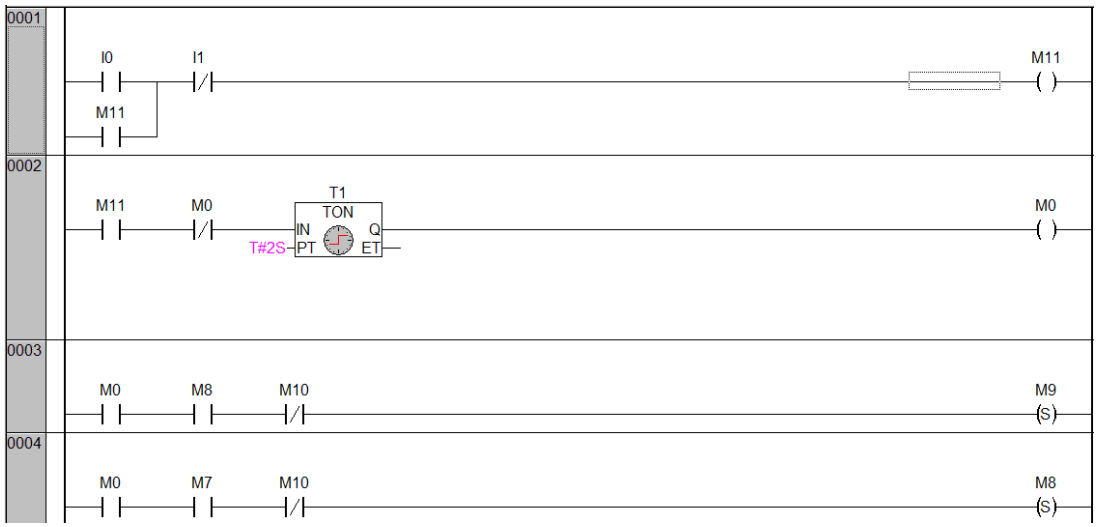

0021
M10
M9
M8
(R)
0022
M10
M9
(R)
0023
M1
M3
Q0
( )
M7
M1
0024
M1
M2
Q1
( )
M3
M1
0025
M2
M4
M7
Q2
( )
M5
M8
M1
0026
M2
M3
M7
M9
Q3
( )
M6
M8
M1
0027
M4
M7
Q4
( )
M8
M1
0028
M2
M4
Q5
( )
M5
M1
0029
M2
M7
Q6
( )
M8

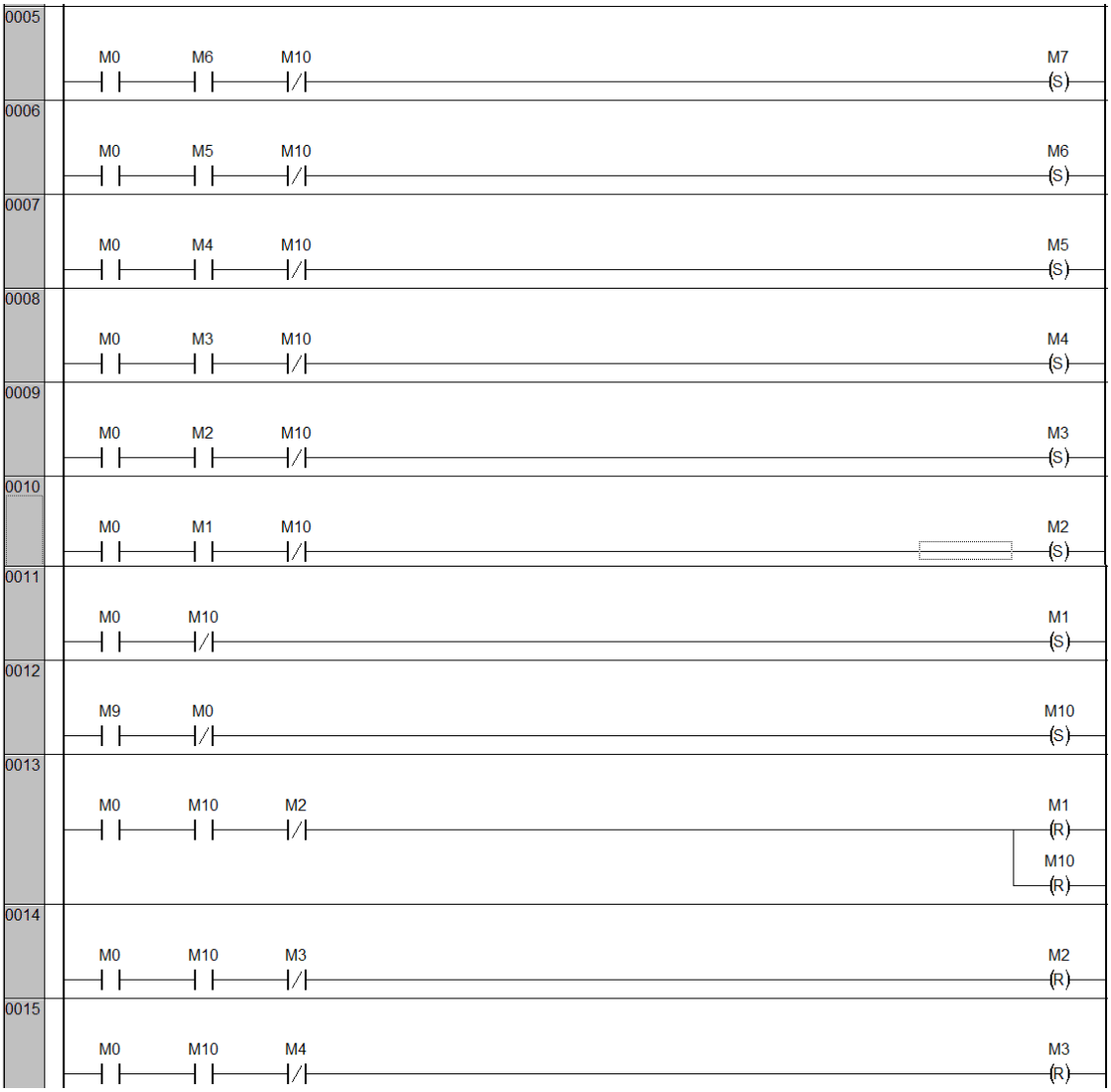
**EX58: SEVEN SEGMENT DISPLAY USING ONE PUSH BUTTON AND ONE TIMER.**

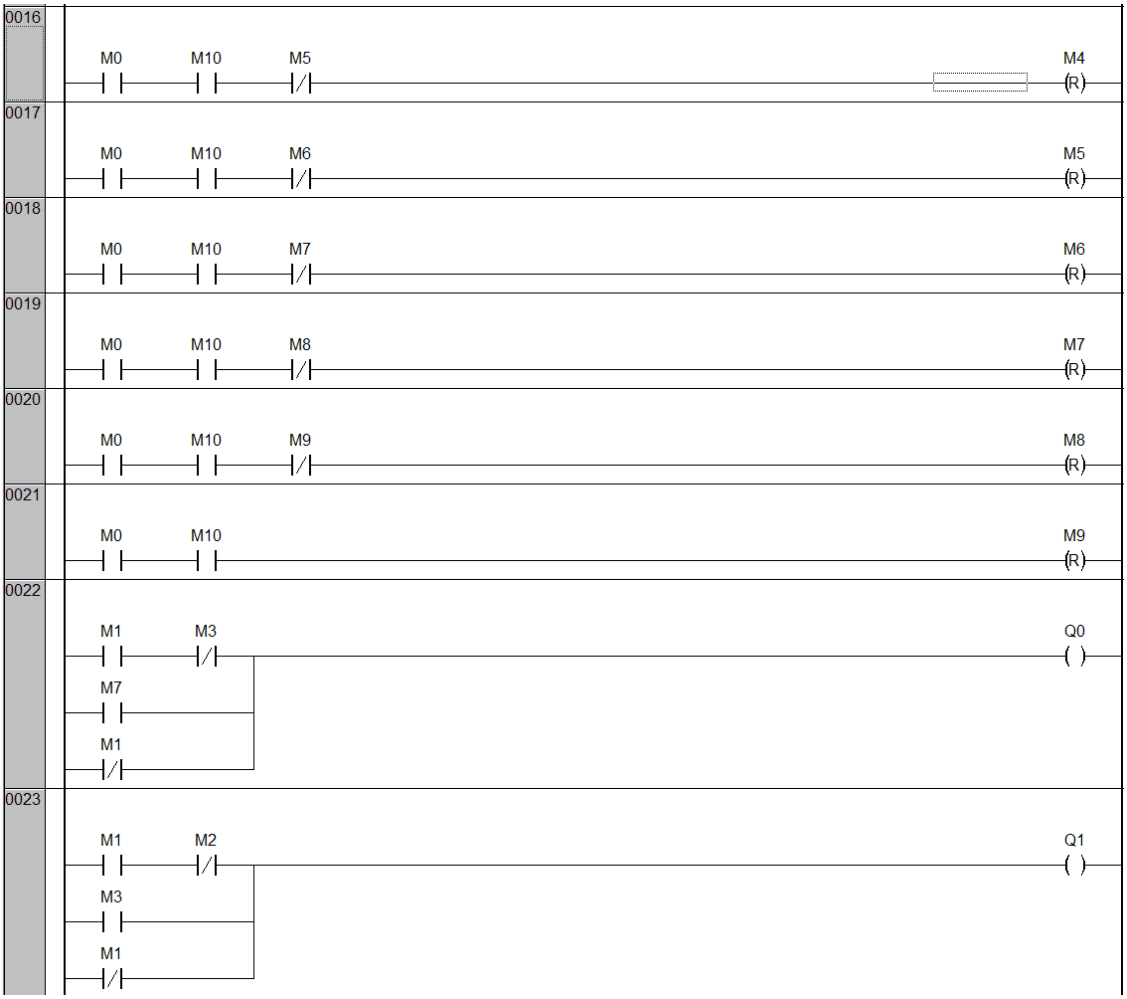
| SIGNAL ↓ | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 |
|---|---|---|---|---|---|---|---|
| M1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| M2 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| M3 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| M4 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| M5 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| M6 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| M7 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| M8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| M9 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

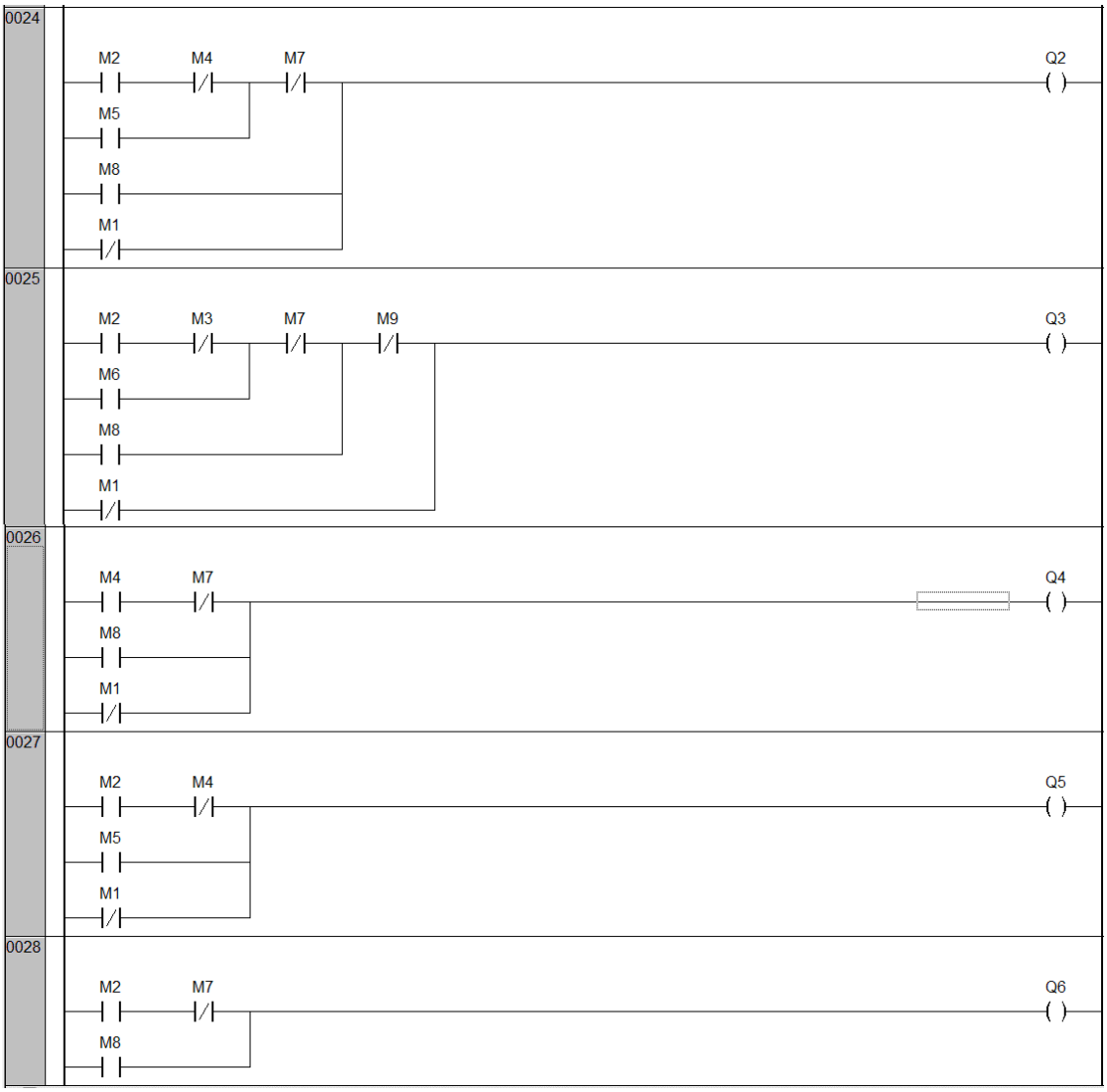


As I0 push button is pressed, number display will change from 0 to 9 AND 9 TO 0 on each 2 seconds interval.


0001
I0
I1
M11
M11
0002
M11
M0
T1
TON
IN
PT
Q
ET
T#2S
M0
0003
M0
M8
M10
M9
S
0004
M0
M7
M10
M8
S

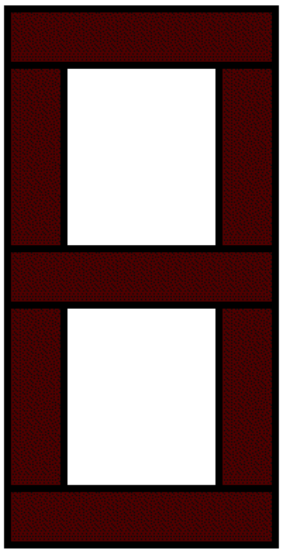

0005
M0
M6
M10
M7
S
0006
M0
M5
M10
M6
S
0007
M0
M4
M10
M5
S
0008
M0
M3
M10
M4
S
0009
M0
M2
M10
M3
S
0010
M0
M1
M10
M2
S
0011
M0
M10
M1
S
0012
M9
M0
M10
S
0013
M0
M10
M2
M1
R
M10
R
0014
M0
M10
M3
M2
R
0015
M0
M10
M4
M3
R


0016
M0
M10
M5
M4
(R)
0017
M0
M10
M6
M5
(R)
0018
M0
M10
M7
M6
(R)
0019
M0
M10
M8
M7
(R)
0020
M0
M10
M9
M8
(R)
0021
M0
M10
M9
(R)
0022
M1
M3
Q0
( )
M7
M1
0023
M1
M2
Q1
( )
M3
M1

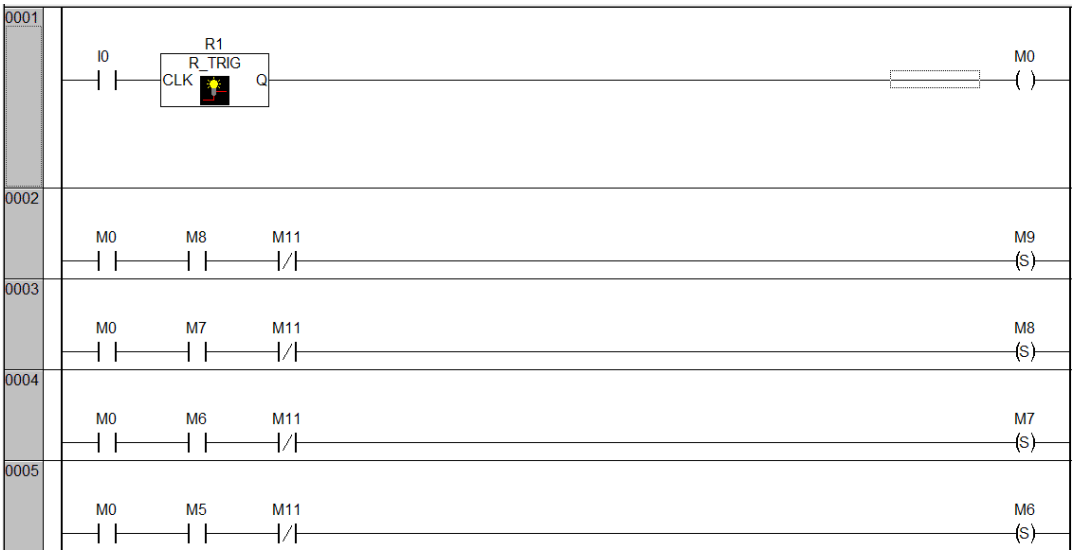


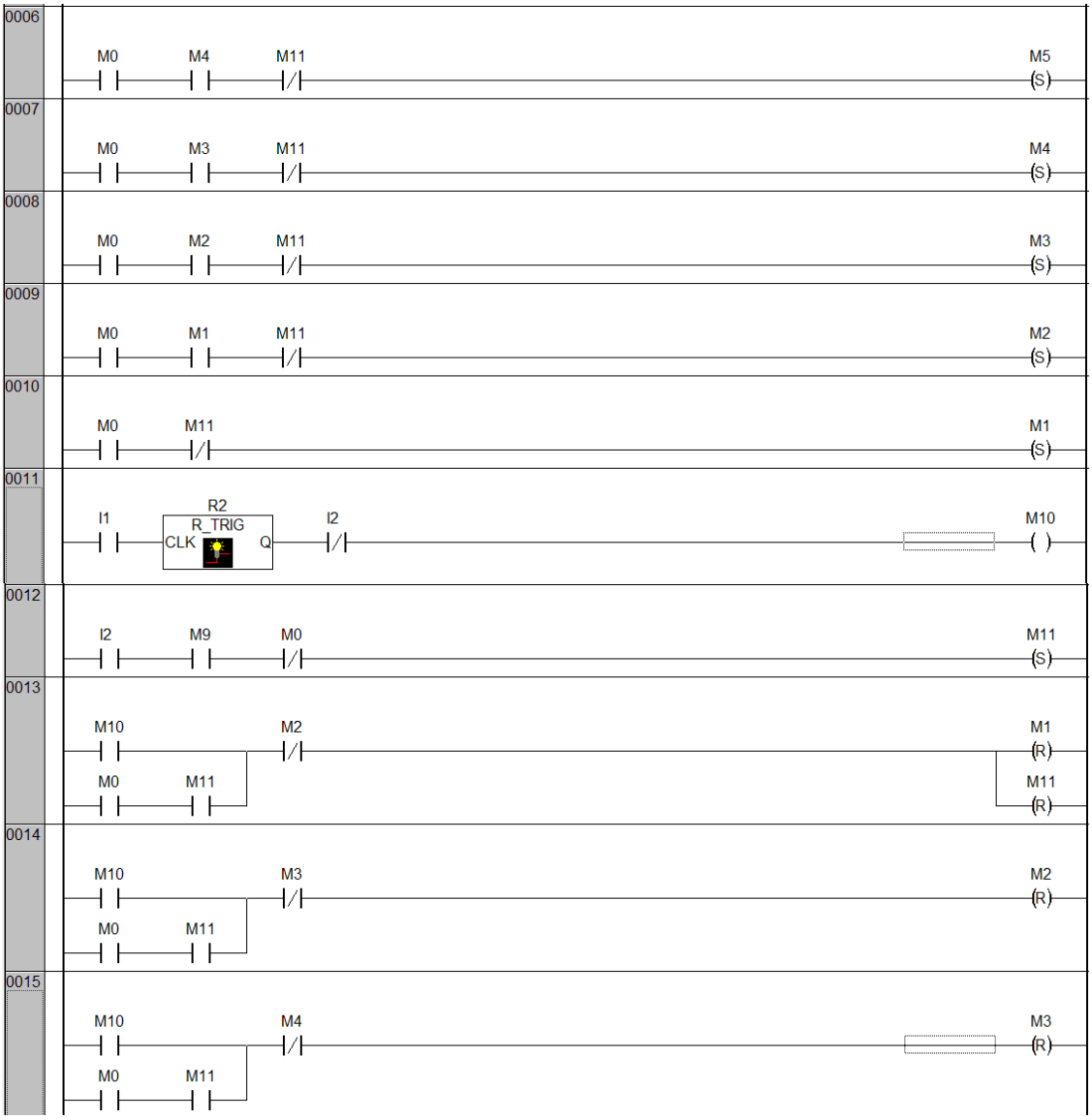

## EX59: SEVEN SEGMENT DISPLAY USING SINGLE AND DOUBLE PUSH BUTTONS AS PER I2 SELECTION.

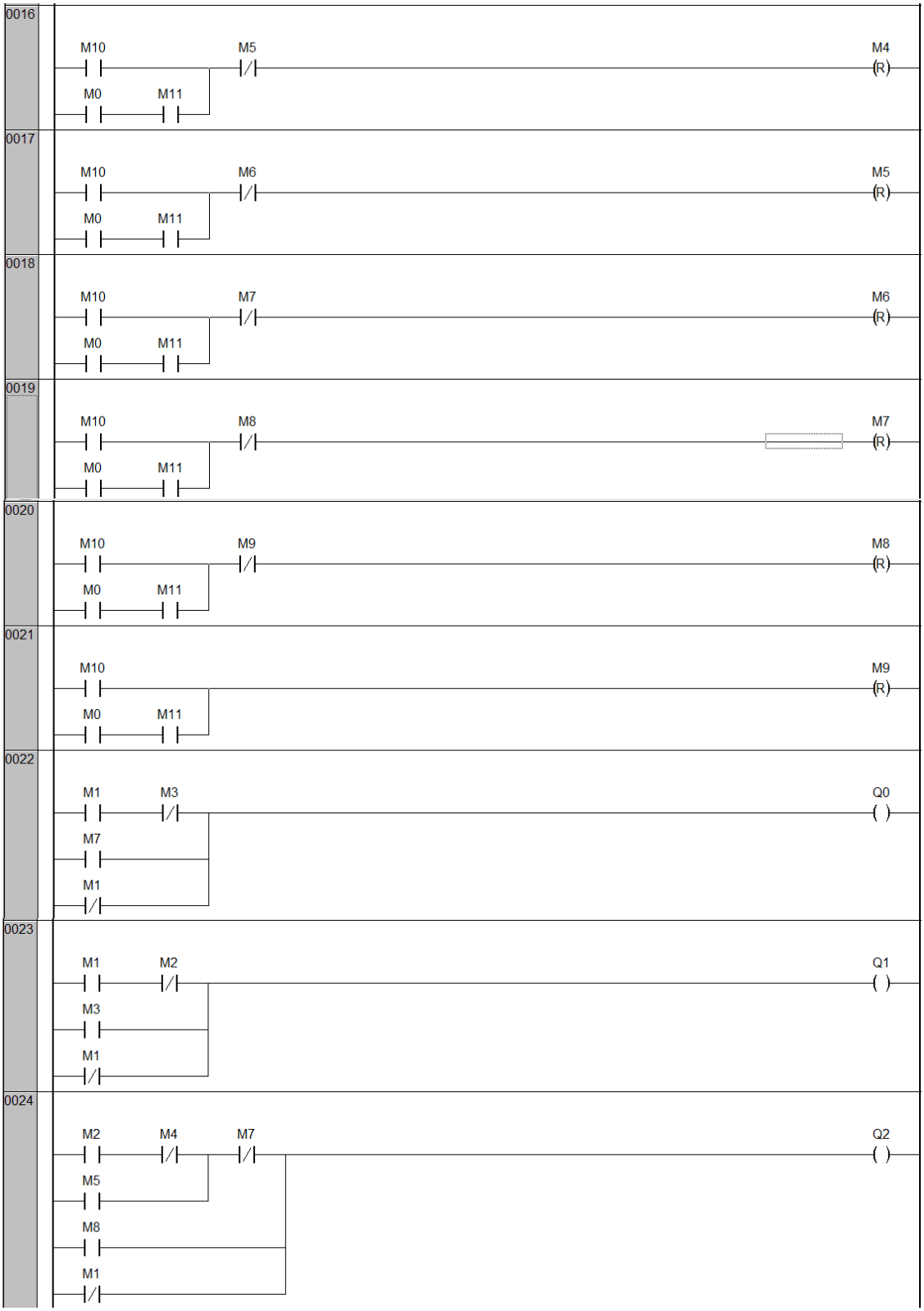
| SIGNAL ↓ | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 |
|---|---|---|---|---|---|---|---|
| M1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| M2 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| M3 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| M4 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| M5 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| M6 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| M7 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| M8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| M9 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

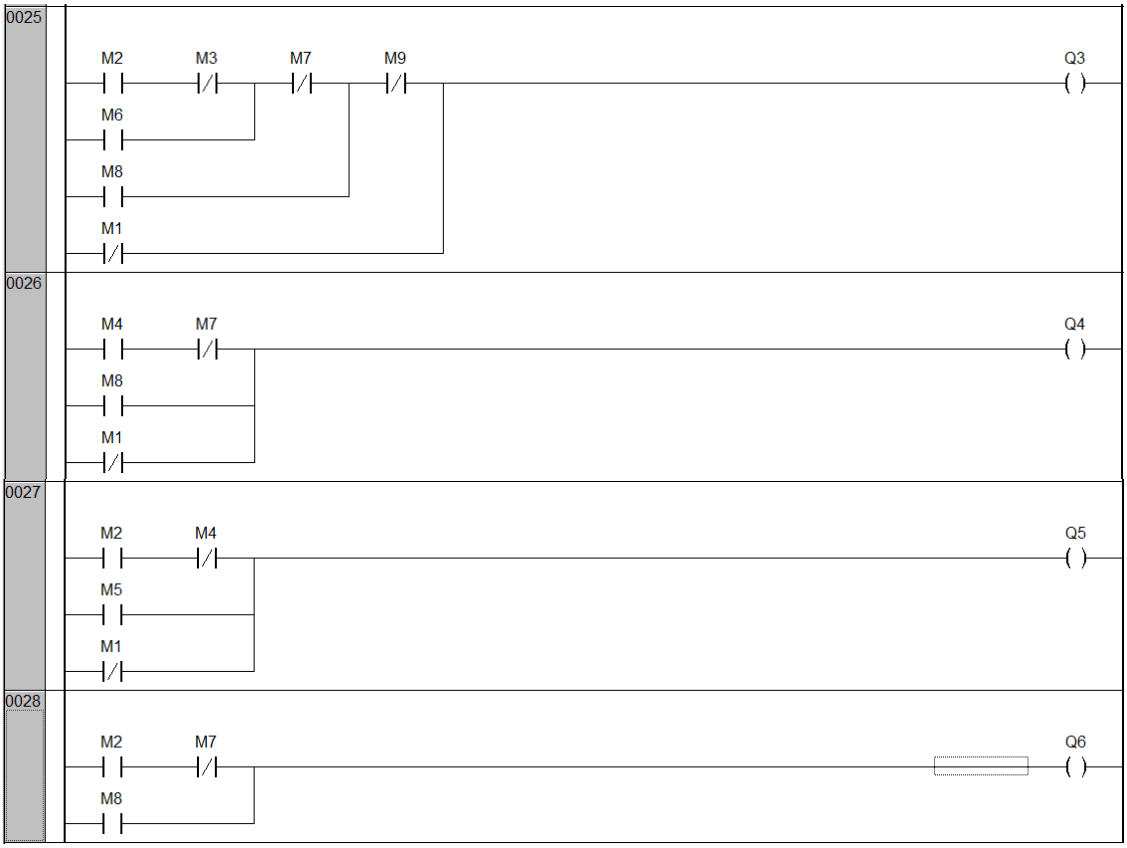


Normally it will run by two push buttons. When I2 is selected/ on, it will run by single push button I0.




0006
M0
M4
M11
M5
(S)
0007
M0
M3
M11
M4
(S)
0008
M0
M2
M11
M3
(S)
0009
M0
M1
M11
M2
(S)
0010
M0
M11
M1
(S)
0011
I1
R2
R_TRIG
CLK
Q
I2
M10
( )
0012
I2
M9
M0
M11
(S)
0013
M10
M2
M1
(R)
M0
M11
M11
(R)
0014
M10
M3
M2
(R)
M0
M11
0015
M10
M4
M3
(R)
M0
M11

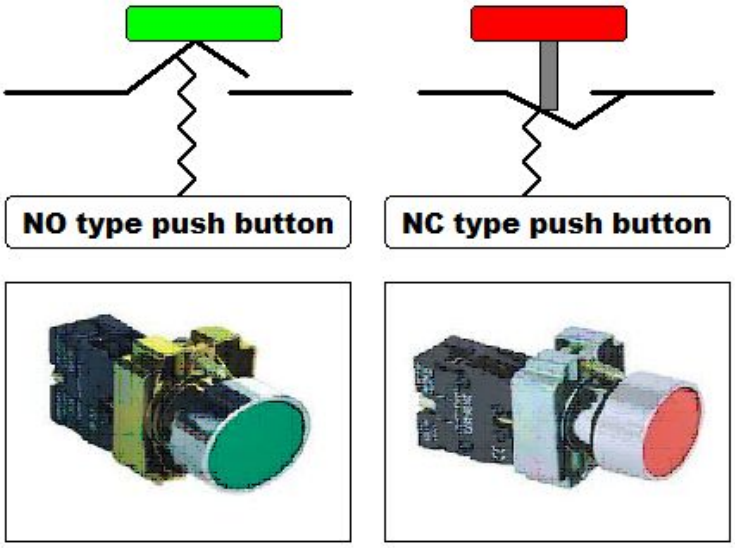

0016
M10
M5
M4
(R)
M0
M11
0017
M10
M6
M5
(R)
M0
M11
0018
M10
M7
M6
(R)
M0
M11
0019
M10
M8
M7
(R)
M0
M11
0020
M10
M9
M8
(R)
M0
M11
0021
M10
M9
(R)
M0
M11
0022
M1
M3
Q0
( )
M7
M1
0023
M1
M2
Q1
( )
M3
M1
0024
M2
M4
M7
Q2
( )
M5
M8
M1




We shall make more programs but before that let us know about physical digital inputs and outputs generally connected with PLC in industries.

## Digital inputs:

1> **Push buttons:** Push button has spring system inside it. It is an assembly of button and contact. Inside contact we have spring. It is available in NO and NC both types. It is generally used for start and stop purpose.

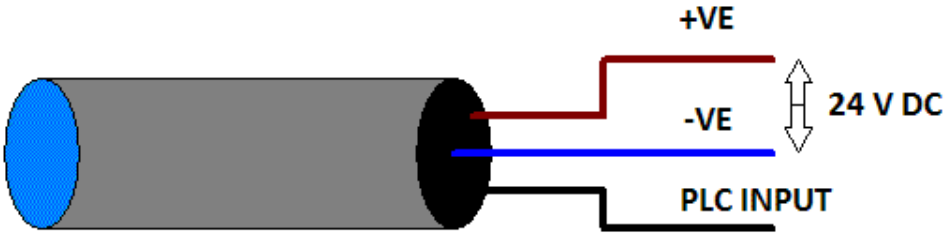



**2> Selector switch:** Selector switches are used on panel for generally auto/ manual mode selection and some other works. It has many positions to be selected. Many positions many inputs. It has mechanical interlock that at once only one input can be selected. It holds its position when operated.

**3> Rotary/ BCD switches:** rotary or binary coded decimal switches are like selector switch as it has also many positions to be selected but it has many position and limited number of inputs. You can select many positions but only few input terminals of PLC will be used and it will

result reduction in PLC cost.


**4> Electronic sensor:** Sensor are available to sense almost every type of objects like metal, wood, glass, magnetic field, colour etc. let it be to sense any object, all digital electronic sensors have universal concept. Basically, it has 3 wires generally.




Two wires are for its power supply and sensor used with PLC will have power supply 24 V DC. Its third wire goes to controller/ PLC input terminal. Its wires have also universal colour. Positive wire is brown/ red coloured, negative wire is blue coloured and output wire of sensor is black coloured. If you have four wires sensor then third wire will be NO type and fourth will be of NC type. You can find any type of sensor in automatic machine as per requirement but universal concept will be the same for all.

Inductive proximity sensor is one of the frequently used sensors. It is used to sense metallic objects/ jobs. Its sensing range is 3 to 8 mm generally.

INDUCTIVE PROXIMITY SENSOR

Capacitive/ photo electric proximity sensor is used to sense any object but object should not be transparent.

CAPACITIVE PROXIMITY SENSOR

There is one more frequently used sensor; Magnetic reed switch. It is used to sense magnetic field. It is used on pneumatic cylinders to sense the position of piston which has magnet on its end. It has two wires.

MAGNETIC REED SWITCH

**5> Limit Switch:** Limit switch is also called mechanical sensor. It senses the position of heavy moving object. It has spring system inside it. It is used on conveyor system, door lock, trolley movement on rail, elevators etc.

LIMIT SWITCH

**6> Level switch:** it is used to sense the position of material in some tank. It does not give the quantity of material but it gives position/ level.

LEVEL SWITCH

**Push buttons, selector switches and BCD switches are manual inputs as a man operates these switches.**
**Electronic sensors, limit switches and level switches are automatic inputs as these are operated by objects.**

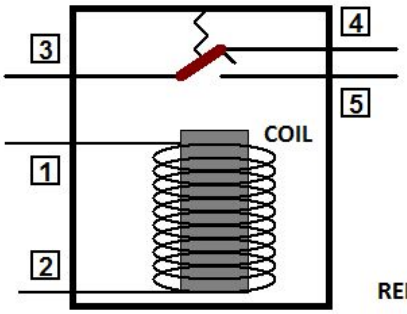
**Digital Outputs:**

1> Any lamp/ indicator, fan, motor, hooter etc. operated on 24 VDC (ampere rate as per mention in PLC catalogue) is the output of PLC and can be directly connected to PLC output terminals.

LAMP/ INDICATOR

2> **Relay:** When we have single phase AC outputs or high ampere rated DC outputs then a 24 V DC operated relay will be used to connect those outputs to PLC output terminals. 5 V DC or 12 V DC relays will not be used as PLC output terminals are 24 V DC operated. It is an electromagnetic switch which has a contact and a coil inside it.

When coil is energised, iron core inside coil gets magnetised and attracts the contact.

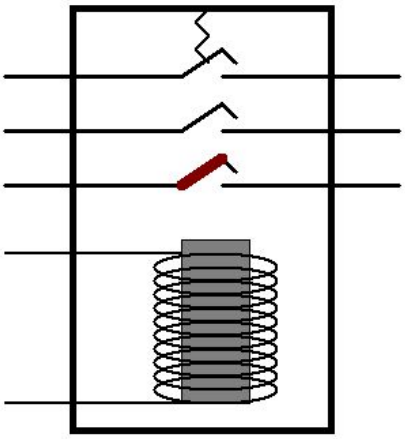



RELAY



1 and 2 terminals of relay are for coil and will be connected to PLC output. 3 and 4 terminals of relay are in NC connection. 3 and 4 terminals of relay are in NO connection.

3> **CONTACTOR:** It is also an electromagnetic switch but it is used for three phase output connection. It is high ampere rating devices. Its coil is also 24 V DC operated. It has three parallel contacts inside it.



CONTACTOR

4> **SOLENOID VALVE:** in many industries we have pipelines and we need to control its open and close by controller. To operate 100 % open and 100 % close of pipe line, solenoid/ electromagnetic valves are used. It is 24 V DC operated. It is also used in hydraulic and pneumatic system pipe line.


**WIRING:**

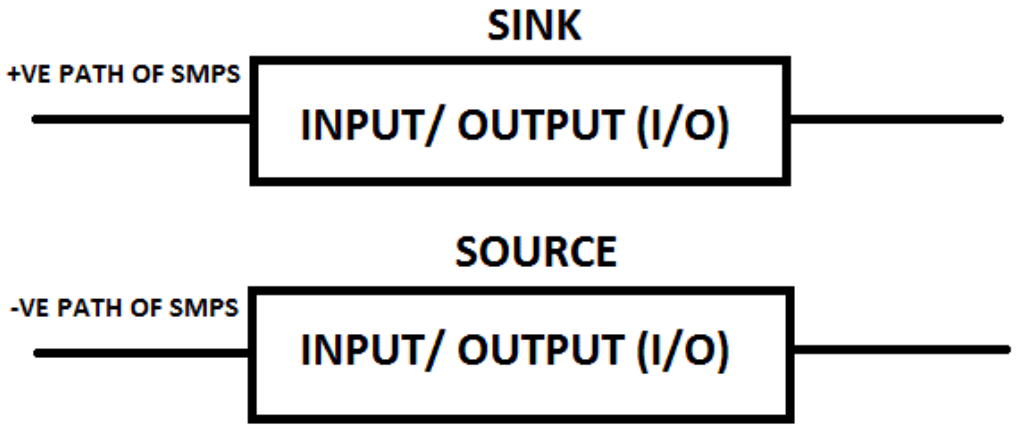
PLC input output wiring is not difficult if you remember few points.

1> PLC wiring is of two types; SINK and SOURCE.

SINK  Connect I/Os to PLC in such way that conventional direction of current should be towards PLC i. e. positive to negative.

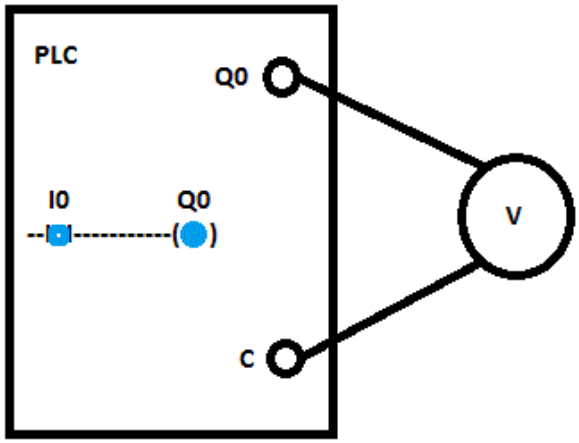
SOURCE  Connect I/Os to PLC in such way that conventional direction of current should be outside PLC i. e. negative to positive.

It might be bit difficult to remember always but don't worry I shall tell you the easiest way to remember SINK SOURCE connection. This sink source is applicable for input and output both. It is manufacturer decided so we need to check PLC catalogue whether input/ output is sink or source. Sink and source is applicable on transistor type PLC not on relay type PLC. To make it easy to remember sink/ source concept always remember that for SINK connection connect I/ O in between positive path of SMPS. For SOURCE connection connect I/O in between negative path of SMPS.



2> PLC's input output terminals are potential less.

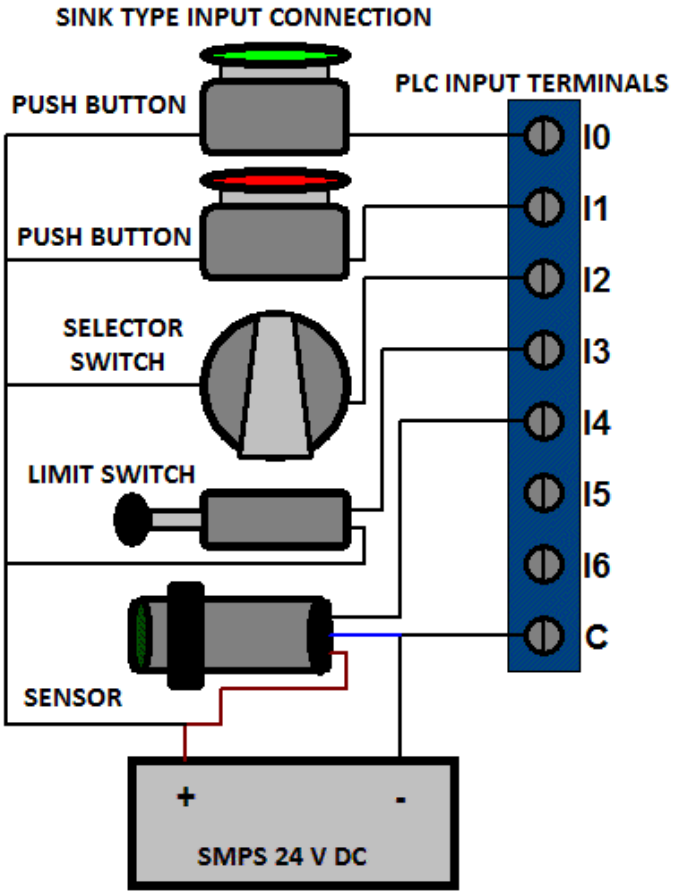
Suppose we have a PLC and we have sent program in it. We have not connected any physical output to its output terminals. Now if we make Q0 coil on in program by any mean then obviously Q0 terminal of PLC will get high. We check voltage across Q0 and output common terminals. What should be the voltage across these two terminals? Will it be 24 V DC because PLC I/O terminals are 24 V DC operated? No, it would be 0 V.



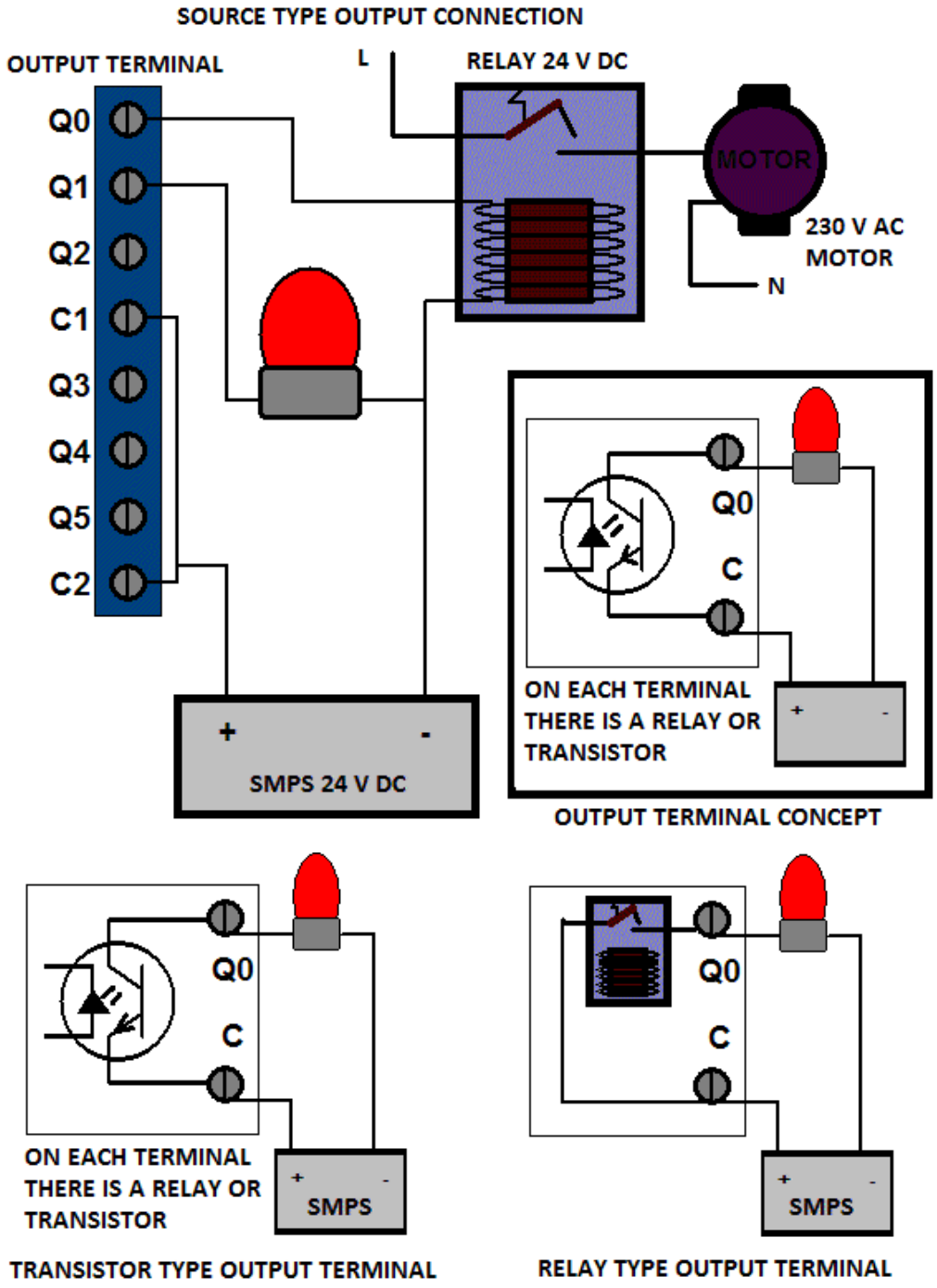

There is no any voltage across output and common terminals when output is on. It is continuity across Q0 and common terminals. It means PLC output terminals act as a switch.

3> We must have one SMPS 24 V DC to wire I/O to PLC. As we do not have voltage across terminals so we need to have a voltage source.

INPUT WIRING: While wiring inputs to PLC input terminals do not be worried about the type and shape of inputs but you just check number of wires in that input. You will find either two wires input or 3 wires input.




## Output Wiring:

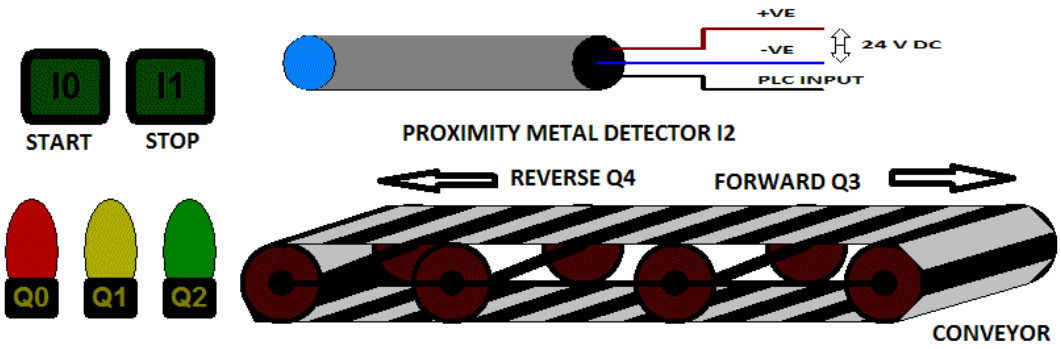



Let us make some more programs. We shall again start making programs by following steps and important points.

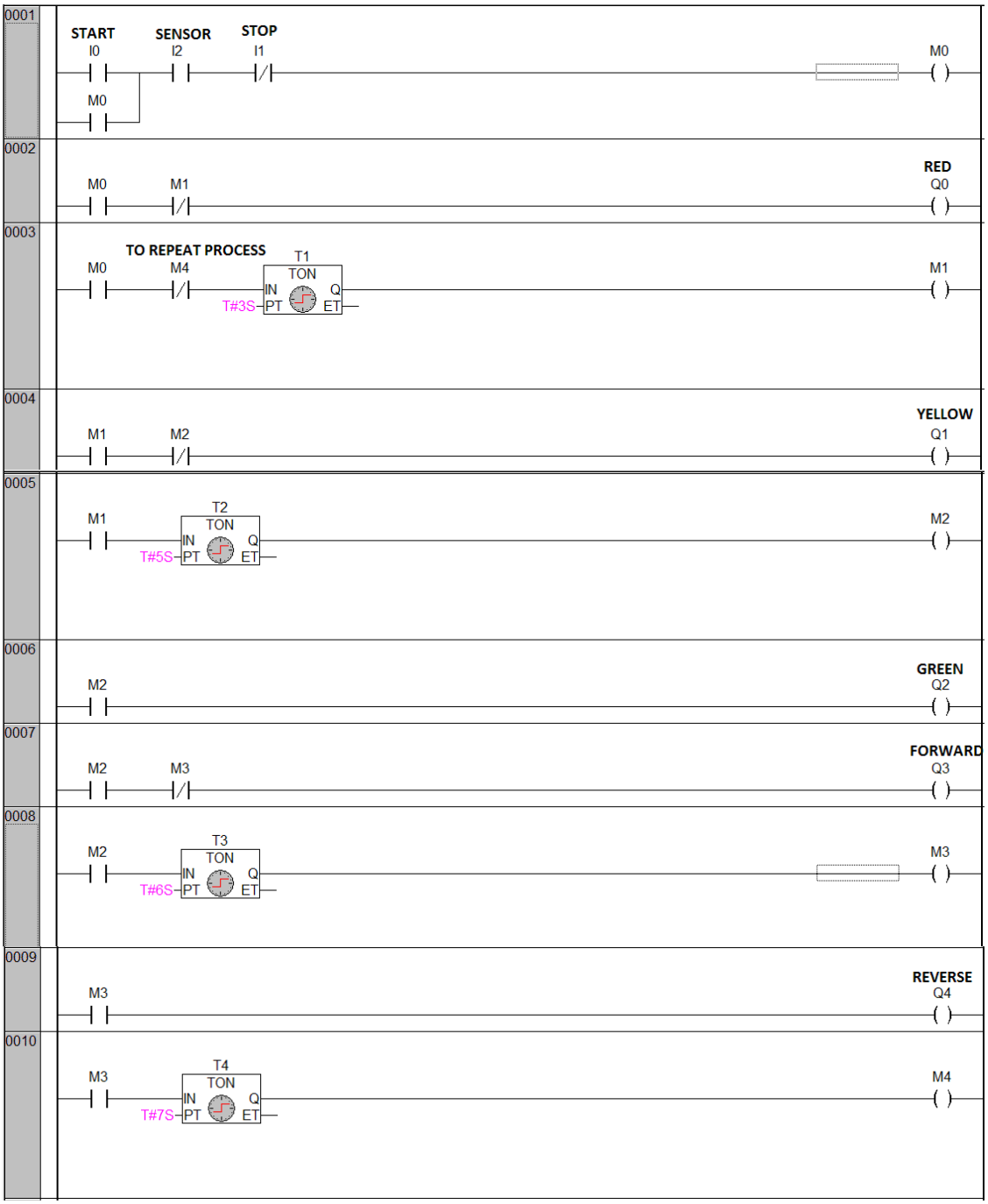
**Important points:**

1> **To switch on use NO contact of the condition signal.**
2> **To switch off use NC contact of the condition signal.**
3> **When we have start signal, we should latch it with some flag.**
4> **When we have any stop signal, we should break the latch directly from it.**
5> **When we get last signal in our program, we should break all latches from it. (It means that lastly after one cycle all coils must get off. If any coil will remain on after one cycle then machine will not run in next cycle.)**
6> **To repeat process in programs based on many timers, take last signal and break first timer.**
7> **To repeat process in programs based on many latches, take last signal and break all latches except start/ first latch.**

**EX60:**

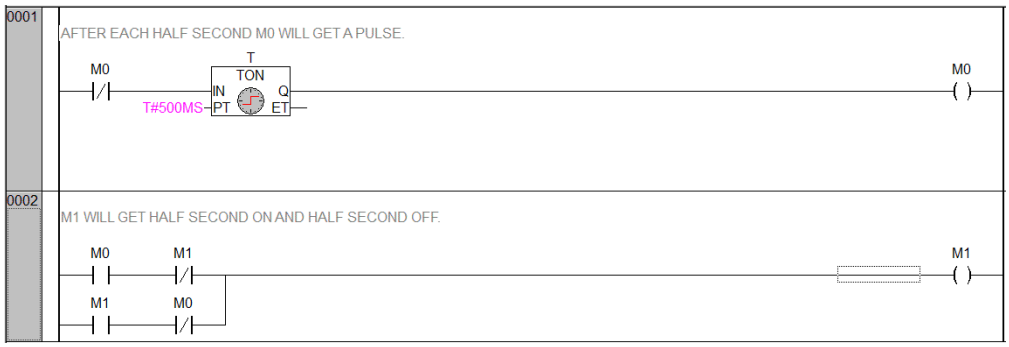



There are two push buttons I0 and I1 (both NO type) for start and stop. There is one proximity sensor I2. There are three lamps/ indicators Q0/ RED, Q1/ Yellow and Q2/ Green. There is a conveyor which can run in forward and reverse direction. For forward direction Q3 should get on and for reverse Q4 should get on. When I0 is pressed, red should get on for 3 seconds. After 3 seconds red should get off and yellow should get on for 5 seconds. After 5 seconds yellow should get off and forward should get on for 6 seconds. After 6 seconds forward will be off and reverse should get on for seven seconds. After seven seconds, repeat the process. I2 sensor must be on to run the machine. If I2 is off and I0 is pressed then machine won't start. When machine runs and in between the process i2 sensor gets off then machine will stop. Whenever conveyor runs either in forward or in reverse direction, green should be on. I1 is for emergency stop.

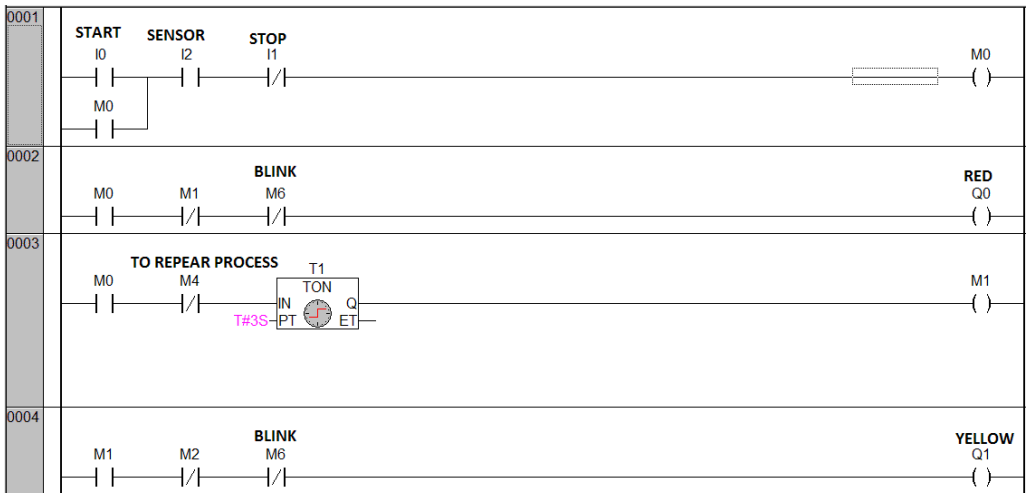


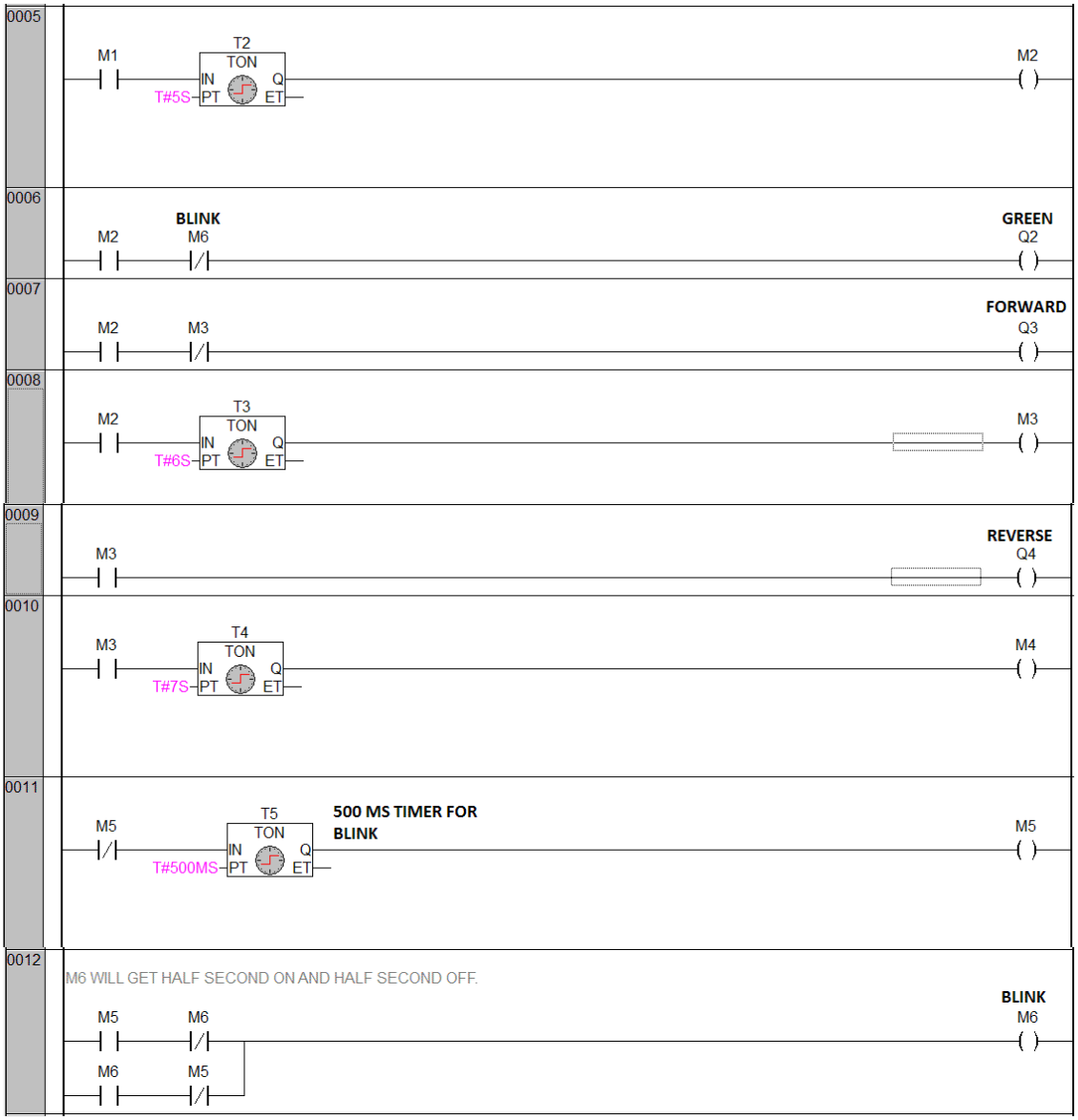

## Ex61: BLINK APPLICATION.

Generate blink of half second interval.

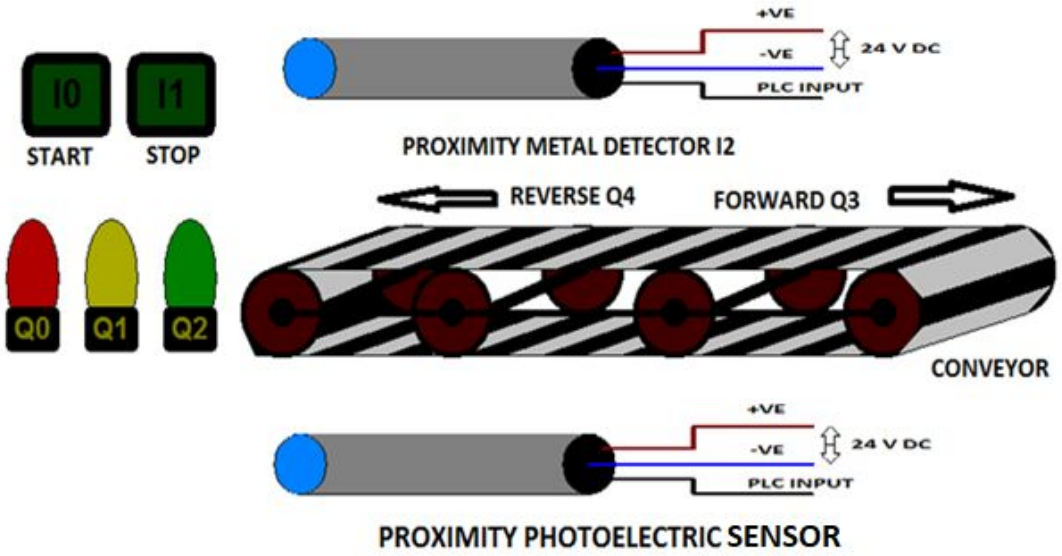



**EX62:** Modify EX60 to add half second interval blink to all lamps Q0, Q1 and Q2.

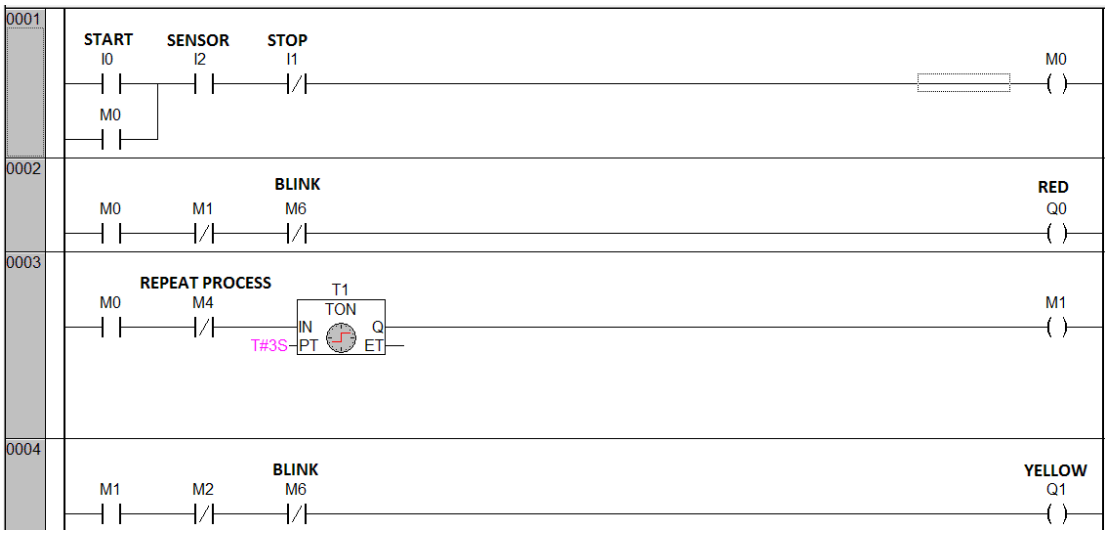


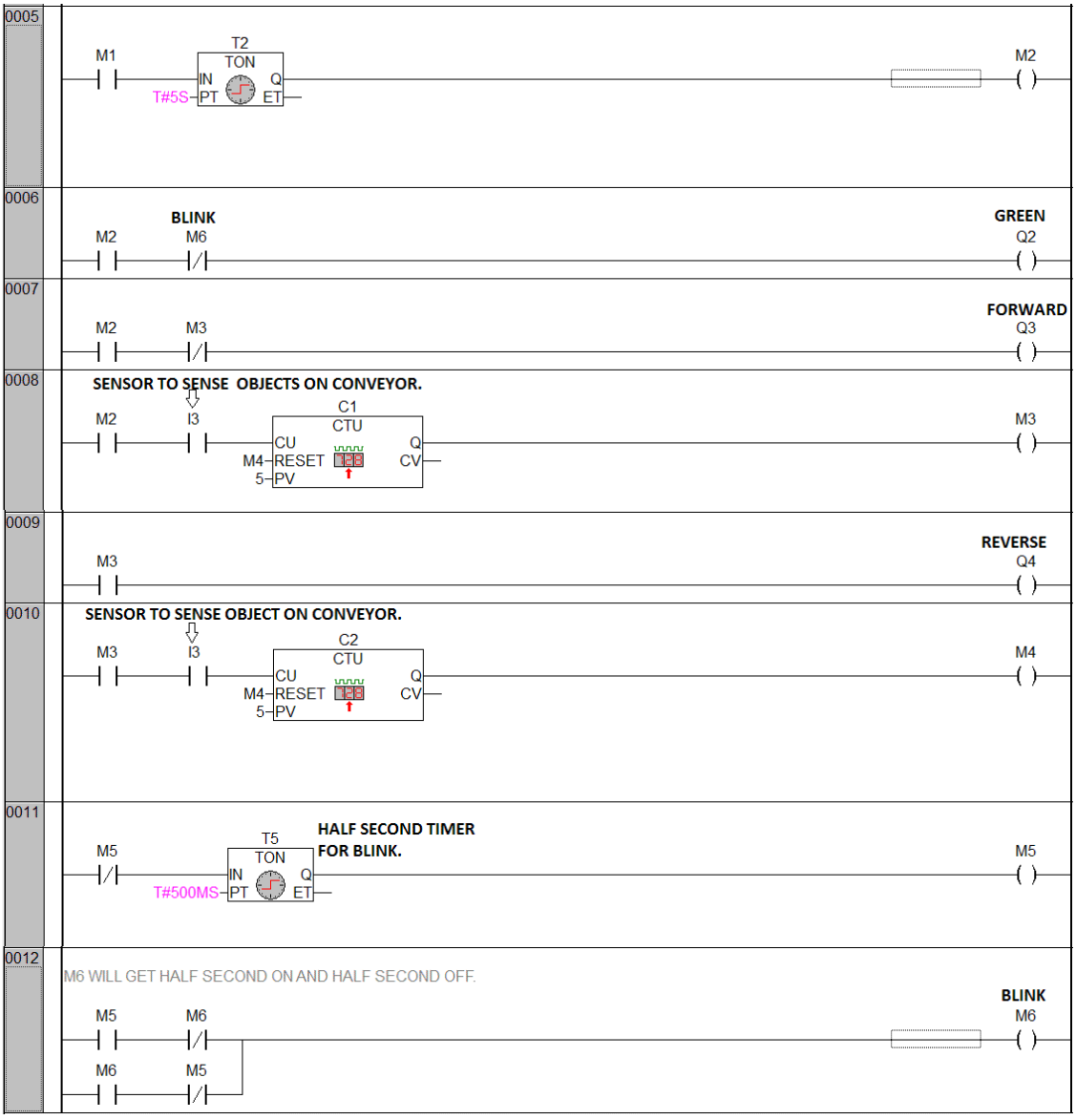

0005
M1
T2
TON
IN
PT
Q
ET
T#5S
M2
0006
BLINK
M2
M6
GREEN
Q2
0007
FORWARD
M2
M3
Q3
0008
M2
T3
TON
T#6S
M3
0009
REVERSE
M3
Q4
0010
M3
T4
TON
T#7S
M4
0011
M5
T5
TON
T#500MS
500 MS TIMER FOR BLINK
M5
0012
M6 WILL GET HALF SECOND ON AND HALF SECOND OFF.
BLINK
M5
M6
M6
M5
M6

EX63:

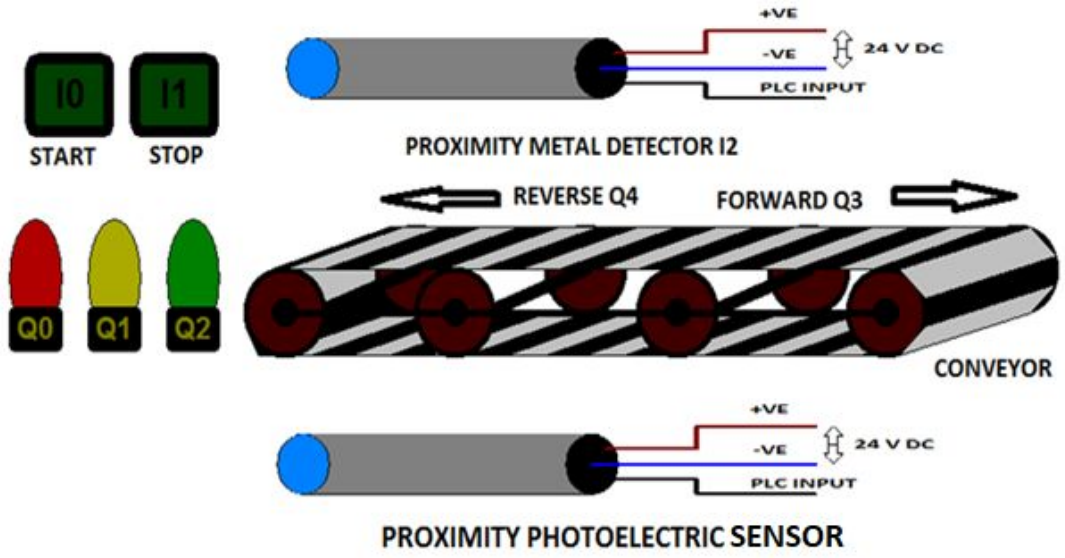



There are two push buttons I0 and I1 (both NO type) for start and stop. There is one INDUCTIVE proximity sensor I2. There is one photoelectric proximity sensor I3. There are three lamps/ indicators Q0/ RED, Q1/ Yellow and Q2/ Green. There is a conveyor which can run in forward and reverse direction. For forward direction Q3 should get on and for reverse Q4 should get on. When I0 is pressed, red should get on for 3 seconds. After 3 seconds red should get off and yellow should get on for 5 seconds. After 5 seconds yellow should get off and forward should get on. Now I3 sensor will sense five objects. After five objects sensed, forward will be off and reverse will be on. Again, I3 will sense 5 objects. After five objects sensed, repeat the whole process. I2 sensor must be on to run the machine. If I2 is off and I0 is pressed then machine won't start. When machine runs and in between the process I2 sensor gets off then machine will stop. Whenever conveyor runs either in forward or in reverse direction, green should be on. All lamps should blink with half second of interval. I1 is for emergency stop.


0001
START
I0
SENSOR
I2
STOP
I1
M0
M0
0002
BLINK
M0
M1
M6
RED
Q0
0003
REPEAT PROCESS
M0
M4
T1
TON
IN
PT
T#3S
Q
ET
M1
0004
BLINK
M1
M2
M6
YELLOW
Q1

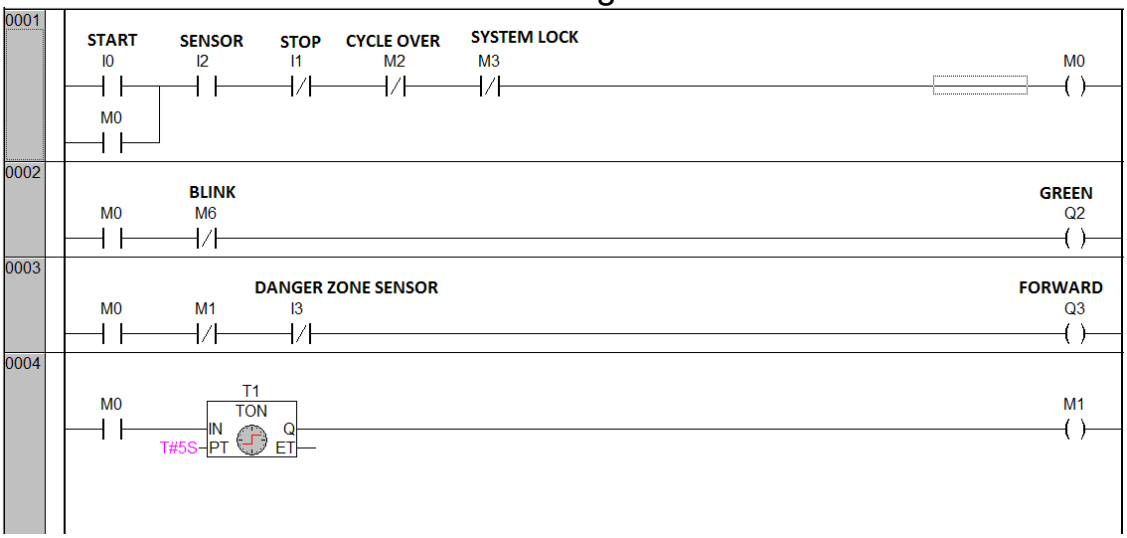

0005
M1
T2
TON
IN
PT
T#5S
Q
ET
M2
0006
BLINK
M2
M6
GREEN
Q2
0007
M2
M3
FORWARD
Q3
0008
SENSOR TO SENSE OBJECTS ON CONVEYOR.
M2
I3
C1
CTU
CU
M4
RESET
5
PV
Q
CV
M3
0009
M3
REVERSE
Q4
0010
SENSOR TO SENSE OBJECT ON CONVEYOR.
M3
I3
C2
CTU
CU
M4
RESET
5
PV
Q
CV
M4
0011
M5
T5
TON
IN
PT
T#500MS
Q
ET
HALF SECOND TIMER
FOR BLINK.
M5
0012
M6 WILL GET HALF SECOND ON AND HALF SECOND OFF.
M5
M6
M6
M5
BLINK
M6

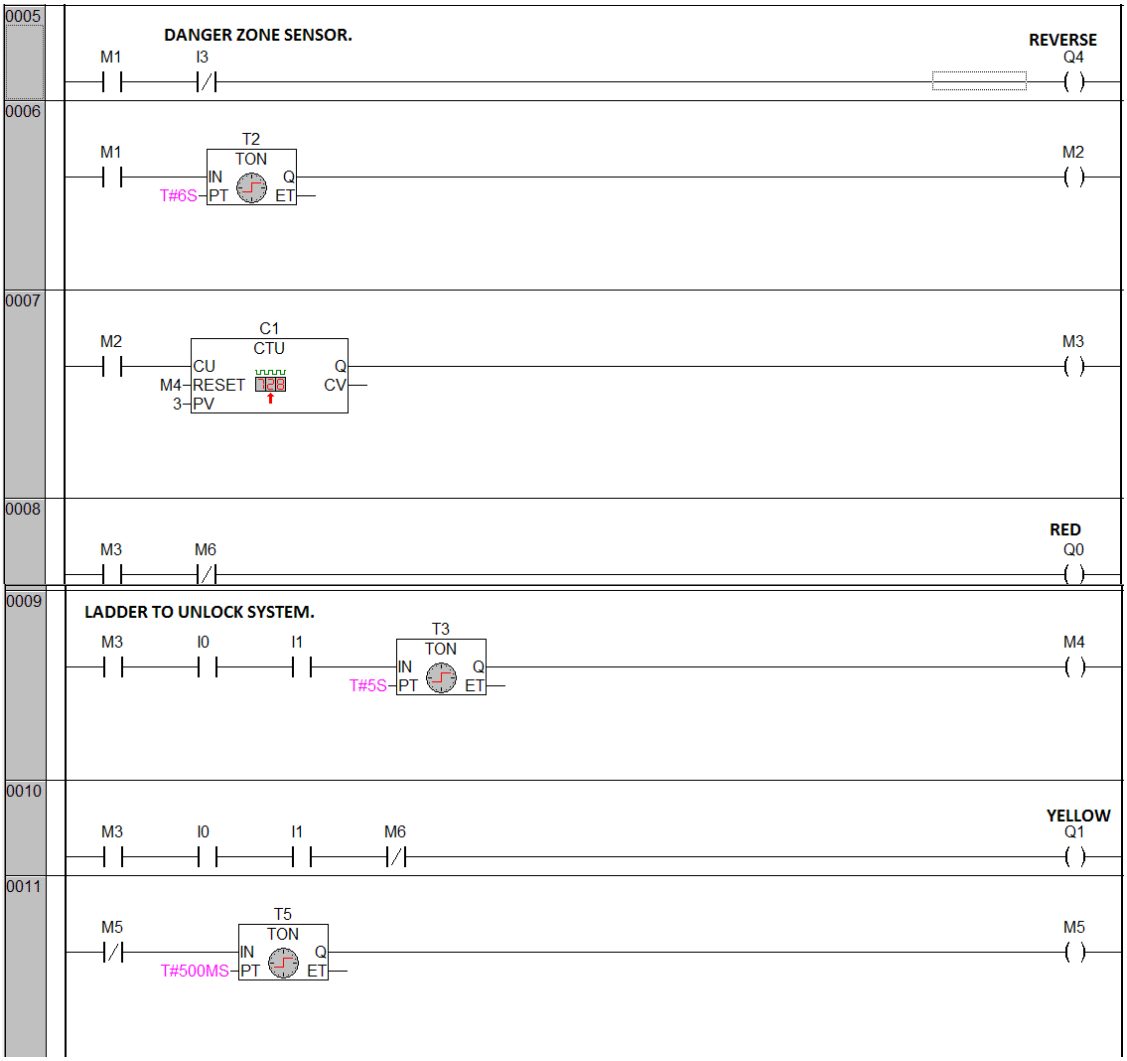
EX64:

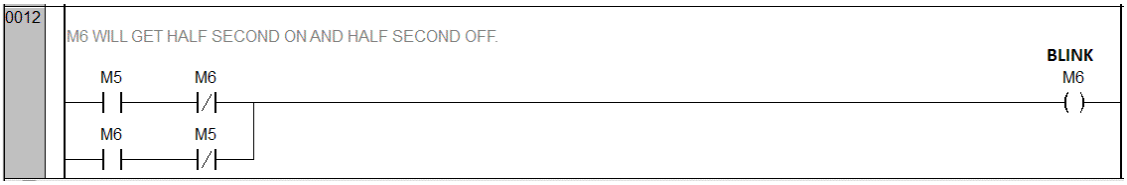



There are two push buttons I0 and I1 (both NO type) for start and stop. There is one INDUCTIVE proximity sensor I2. There is one photoelectric proximity sensor I3. There are three lamps/ indicators Q0/ RED, Q1/ Yellow and Q2/ Green. There is a conveyor which can run in forward and reverse direction. For forward direction Q3 should get on and for reverse Q4 should get on. When I0 is pressed, forward is on for 5 second. After 5 seconds forward off and reverse is on for 6 seconds. After 6 seconds reverse off/ break latch. This is one cycle. Press start button to start next cycle. Like this run machine 3 times/ 3 cycles. After 3 cycles machine will be locked. It means if you press start button fourth time, machine will not start. To unlock the system, press I0 and I1 simultaneously for 5 seconds. After 5 seconds machine will get unlocked and can be run. I2 sensor must be on to run the machine. If I2 is off and I0 is pressed then machine won't start. When machine runs and in between the process I2 sensor gets off then machine will stop. Whenever conveyor runs either in forward or in reverse direction, green should be on. When system is locked, red should be on. While unlocking the machine, yellow will be on. All lamps should blink with half second of interval. Whenever any person goes near conveyor/ I3 on, conveyor should be paused in either direction. I1 is for emergency stop. When I3 is off motor will run. While motor pause, other elements will keep

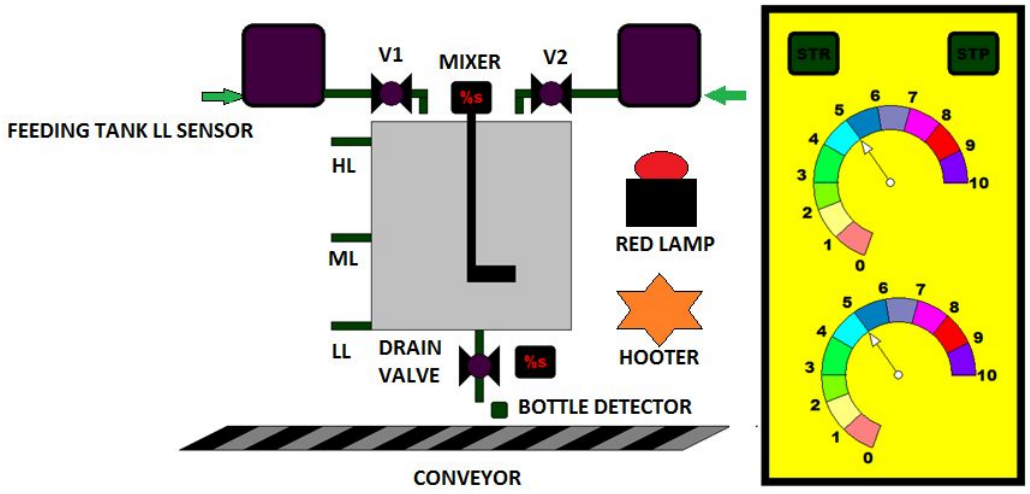
running.

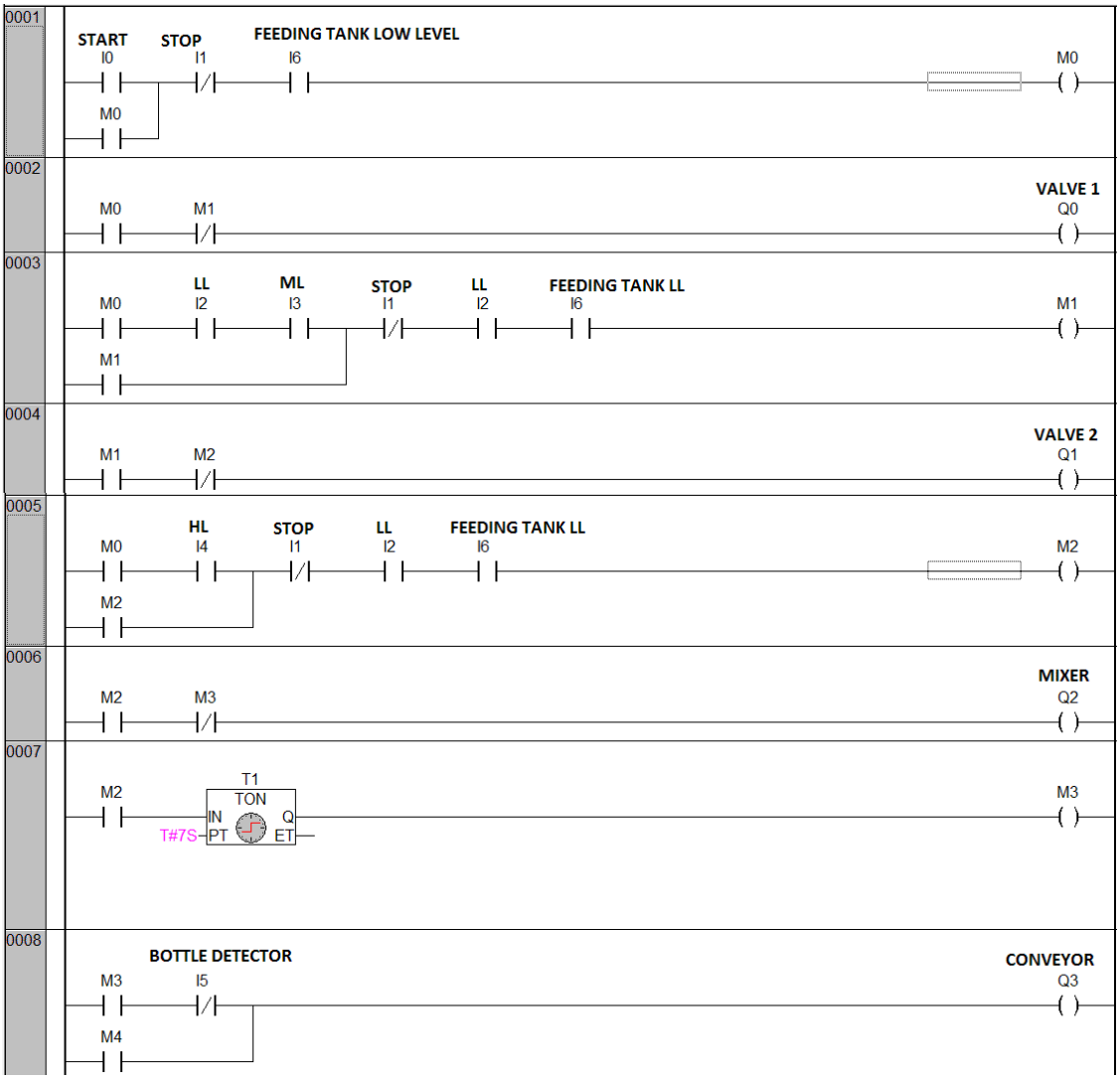

0001
START
I0
SENSOR
I2
STOP
I1
CYCLE OVER
M2
SYSTEM LOCK
M3
M0
M0
M0
0002
BLINK
M0
M6
GREEN
Q2
0003
DANGER ZONE SENSOR
M0
M1
I3
FORWARD
Q3
0004
M0
T1
TON
IN
PT
T#5S
Q
ET
M1

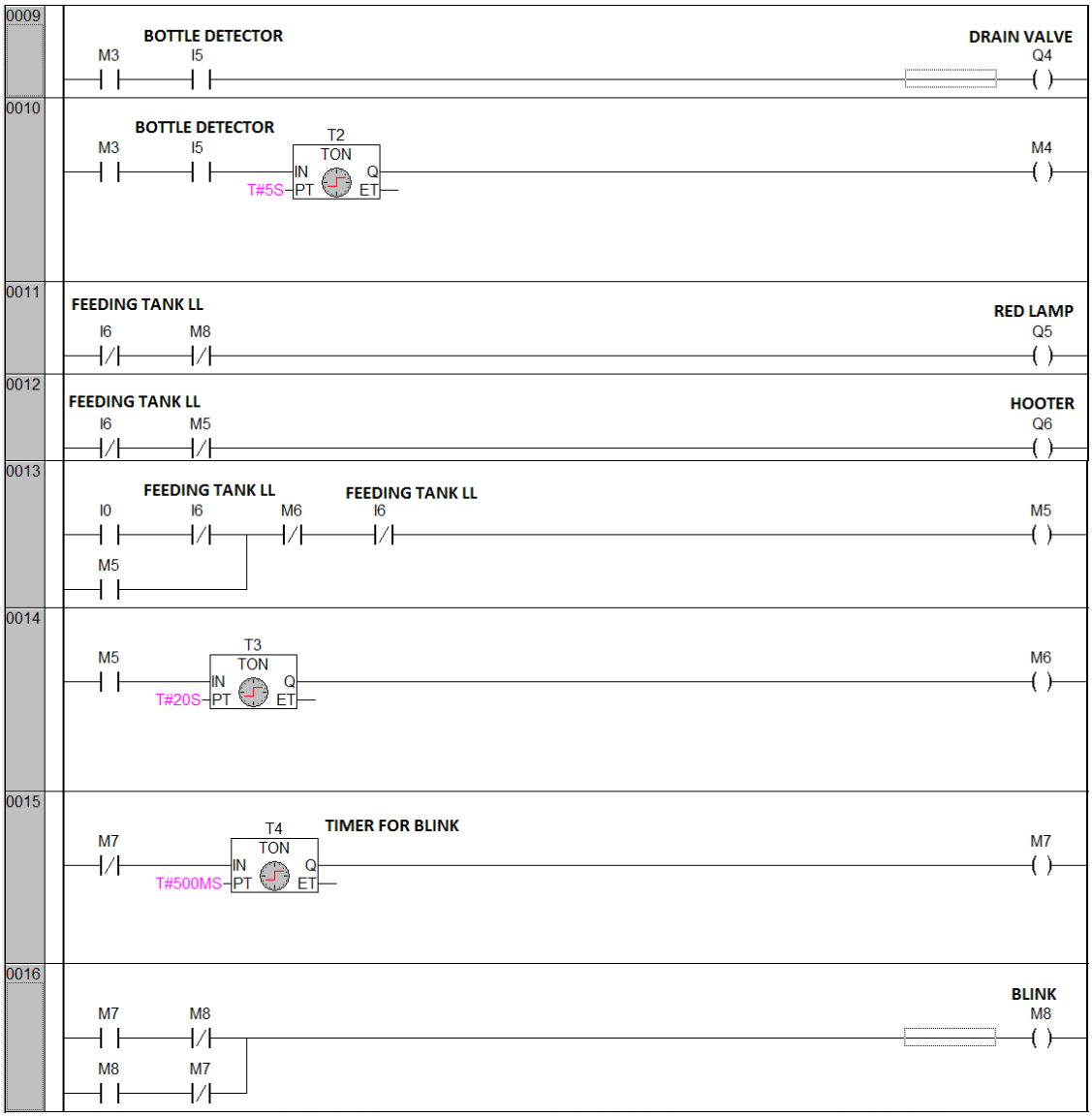

0005
DANGER ZONE SENSOR.
M1
I3
REVERSE
Q4
0006
M1
T2
TON
IN
PT
T#6S
Q
ET
M2
0007
M2
C1
CTU
CU
RESET
M4
PV
3
Q
CV
M3
0008
M3
M6
RED
Q0
0009
LADDER TO UNLOCK SYSTEM.
M3
I0
I1
T3
TON
IN
PT
T#5S
Q
ET
M4
0010
M3
I0
I1
M6
YELLOW
Q1
0011
M5
T5
TON
IN
PT
T#500MS
Q
ET
M5



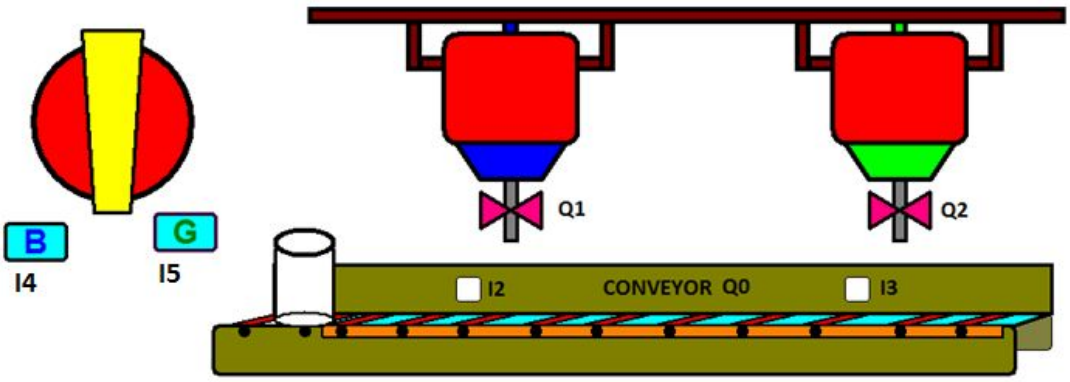

**EX65: Mixing and filling application with alarm, acknowledgement and reset.** There are two push buttons I0 and I1 for start and stop (both push buttons are NO type/ green in physical existence). There are three level switches I2, I3, I4 and I5 for low level, middle level high level and bottle detector sensor respectively. There are 3 valves; V1/ Q0 for material 1, V2/ Q1 for material 2 and V3/ Q3 for drain. There is a motor for mixing two materials in the mixing tank and one conveyor Q4. When I0 is pressed, Q0/ V1 will get on resulting material 1 inlet to mixing tank. As material 1 comes in mixing tank, low level sensor I2 will get on and further middle level sensor I3 will be on. As I3 is on, Q0/ V1 should get off and Q1/ V2 should get on. As V2 is on, material 2 will come in mixing tank resulting increase in level and reaches to I4 sensor. As I4 is on, Q1/ V2 will get off and motor Q2 will get on for 10 seconds. After 10 seconds, motor gets off and conveyor Q4 gets on. Bottle to be filled will move on it. As bottle will be detected by I5 sensor, conveyor Q4 will stop and drain V3/ Q3 will get on for 5 seconds. After 5 seconds again conveyor will be on and drain will get off. Whenever I5 detects bottle, the same cycle of conveyor and drain will repeat. As drain will get on many times, mixing tank will start getting empty. Slowly level will go down resulting I4 off, I3 and I2 sensors off one by one. As I2 is off repeat the process. I1 is manual stop. There is a common low level detector sensor in both upper feeding tanks. If feeding tank low level sensor I6 goes off, whole system should stop. A red lamp Q5 and a hooter Q6 should get on as I6 is off. This is alarm condition. As alarm occurs, someone will hear hooter and will come to monitor the system. I0 push button should be pressed to acknowledge it but by pressing I0 other things should not start. As it is acknowledged, hooter will stop for 20 seconds but red lamp will keep blinking with half second of interval. If problem is not solved with in twenty seconds then hooter will start again. It can be acknowledged again. When problem is solved i. e. feeding tanks are full then hooter and red lamp both will get off and machine can be started by pressing I0 push button.


V1
MIXER
V2
FEEDING TANK LL SENSOR
HL
ML
LL
DRAIN VALVE
RED LAMP
HOOTER
BOTTLE DETECTOR
CONVEYOR
STR
STP


0001
START
I0
STOP
I1
FEEDING TANK LOW LEVEL
I6
M0
M0
0002
M0
M1
VALVE 1
Q0
0003
M0
LL
I2
ML
I3
STOP
I1
LL
I2
FEEDING TANK LL
I6
M1
M1
0004
M1
M2
VALVE 2
Q1
0005
M0
HL
I4
STOP
I1
LL
I2
FEEDING TANK LL
I6
M2
M2
0006
M2
M3
MIXER
Q2
0007
M2
T1
TON
IN
PT
T#7S
Q
ET
M3
0008
BOTTLE DETECTOR
M3
I5
CONVEYOR
Q3
M4



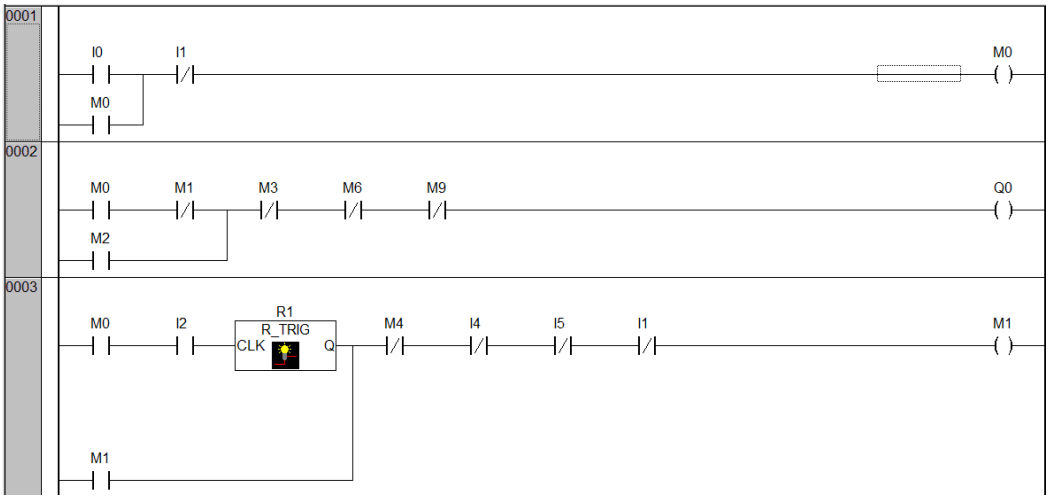

## EX66: filling application as per selection of product.

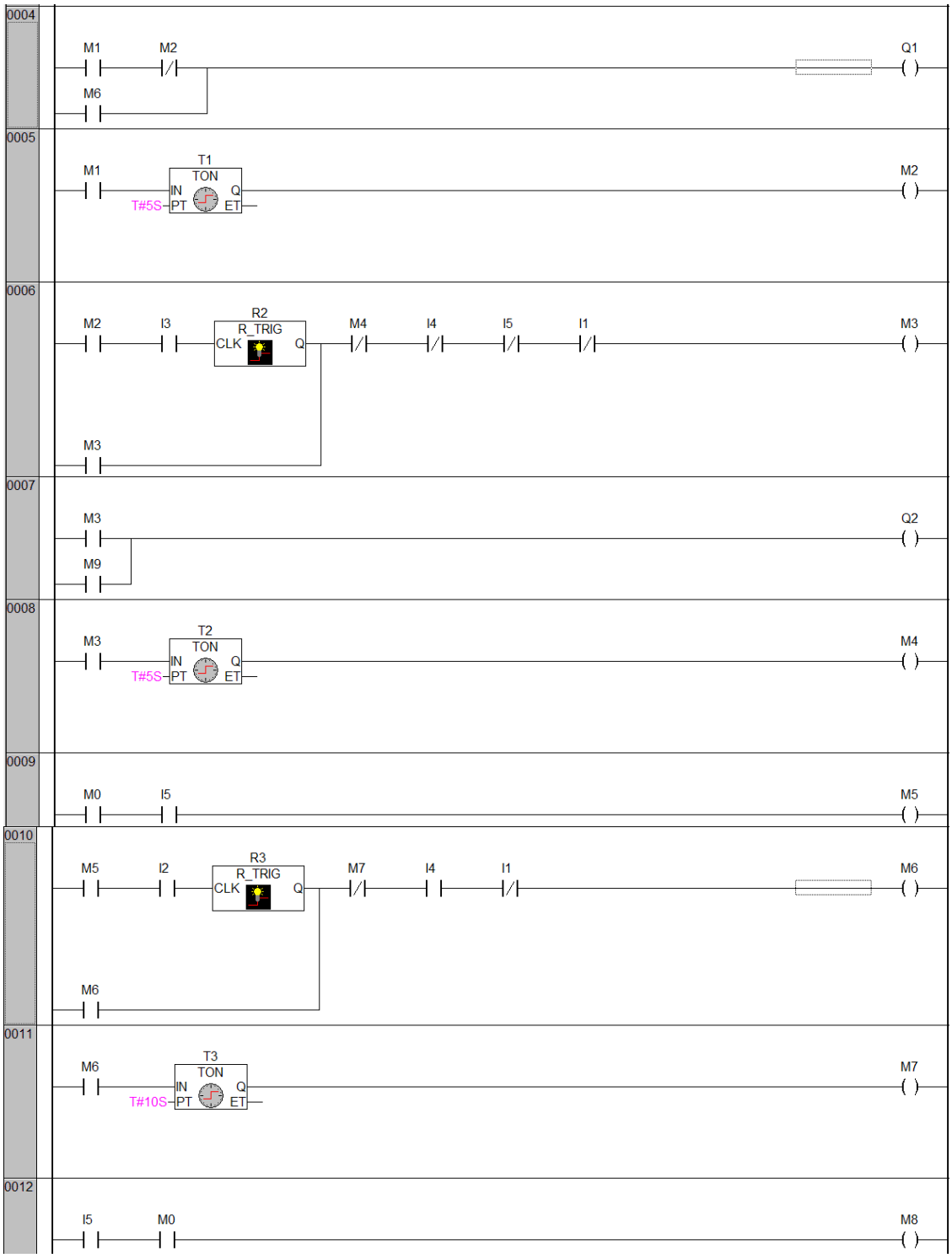


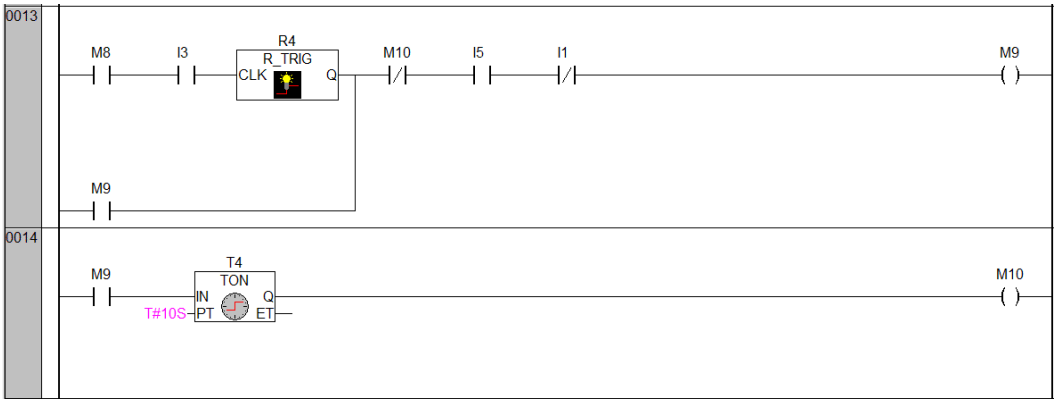

There is a conveyor Q0. There are two feeders with valve Q1 for blue colour and valve Q2 for green colour product. I0 is start push button, I1 is stop push button, I2 is a sensor below blue feeder and I3 is a sensor below green feeder. There is a selector switch with two inputs I4 and I5. When I0 is pressed conveyor will start. If I4/ blue product is selected and I2 gets on then conveyor will stop and Q1 will be on for ten seconds. After ten seconds Q1 valve will be off and conveyor Q0 will be on. Nothing will happen when sensor I3 is on as I4 is selected.

If I5/ green product is selected and I3 gets on then conveyor will stop and Q2 will be on for ten seconds. After ten seconds Q2 valve will be off and conveyor Q0 will be on. Nothing will happen when sensor I2 is on as I5 is selected.

If neither I4 nor I5 is selected and I2 is on then conveyor will stop and Q1 will be on for 5 seconds. After 5 seconds Q1 will be off and conveyor will start. When bottle reaches to I3 sensor then again conveyor will stop and valve Q2 will be on for 5 seconds. After five seconds Q2 will be off and conveyor will be on. I1 is for emergency stop.

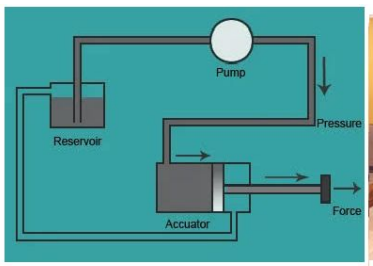



0004
M1
M2
M6
Q1
0005
M1
T1
TON
IN
PT
T#5S
Q
ET
M2
0006
M2
I3
R2
R_TRIG
CLK
Q
M4
I4
I5
I1
M3
M3
0007
M3
M9
Q2
0008
M3
T2
TON
IN
PT
T#5S
Q
ET
M4
0009
M0
I5
M5
0010
M5
I2
R3
R_TRIG
CLK
Q
M7
I4
I1
M6
M6
0011
M6
T3
TON
IN
PT
T#10S
Q
ET
M7
0012
I5
M0
M8

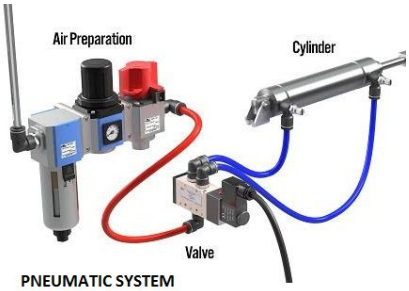



**Hydraulic and pneumatic system:** hydraulic and pneumatic system is frequently used in industrial automation. Hydraulic system has oil tank, pressure pump, inlet/ outlet valves and a cylinder with piston inside it. Compressed oil is used to move piston forward in hydraulic system. Heavy pressure can be generated in this system. It is used for heavy task such as in moulding machines, press machines, earth movers etc. when inlet valve is on compressed oil goes inside cylinder and moves the piston forward. When inlet valve is off and outlet valve is on then oil returns back to oil tank and piston moves back.

Pneumatic system has air compressor, inlet/ outlet valves and a cylinder with piston inside it. Piston is moves by compressed air. It has two types of cylinder i. e. double acting and single acting. To operate double acting cylinder two valves are used. When inlet valve gets on, air goes inside cylinder and piston moves in forward direction. When inlet valve is off and outlet valve is off then air is exhausted and piston moves in reverse direction. In single acting cylinder only one valve is used. When valve is on, air goes inside the cylinder and piston moves in forward direction. When valve is off then piston moves in reverse direction due to spring attached inside and air gets exhausted pneumatic system is used for low pressure work such as in clamping, stopper on conveyor, light object lifting etc. for home position and final position feedback of piston, reed/ magnetic switch is used on pneumatic cylinders. In hydraulic system proximity sensors or limit switches are used for position feedback. Program for hydraulic and pneumatic cylinder will be the same.



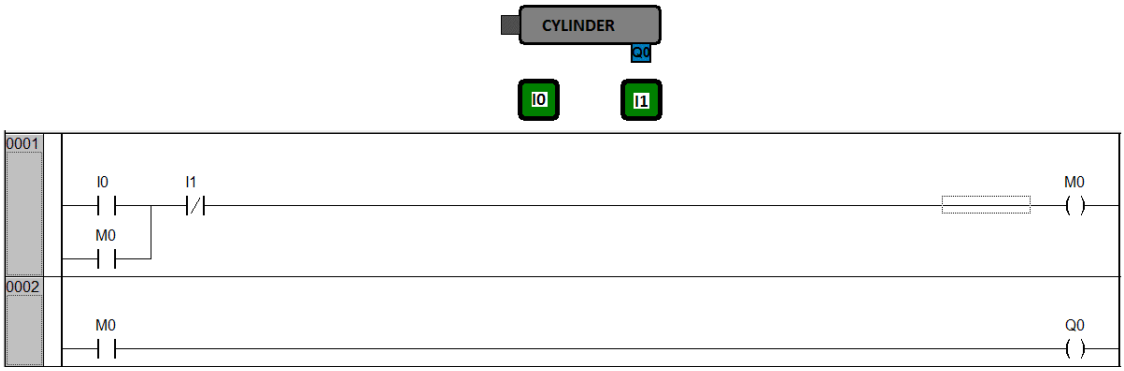

HYDRAULIC SYSTEM



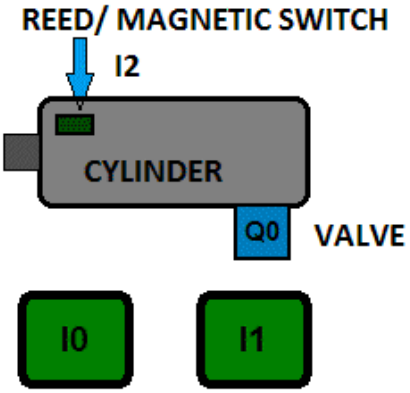

PNEUMATIC SYSTEM

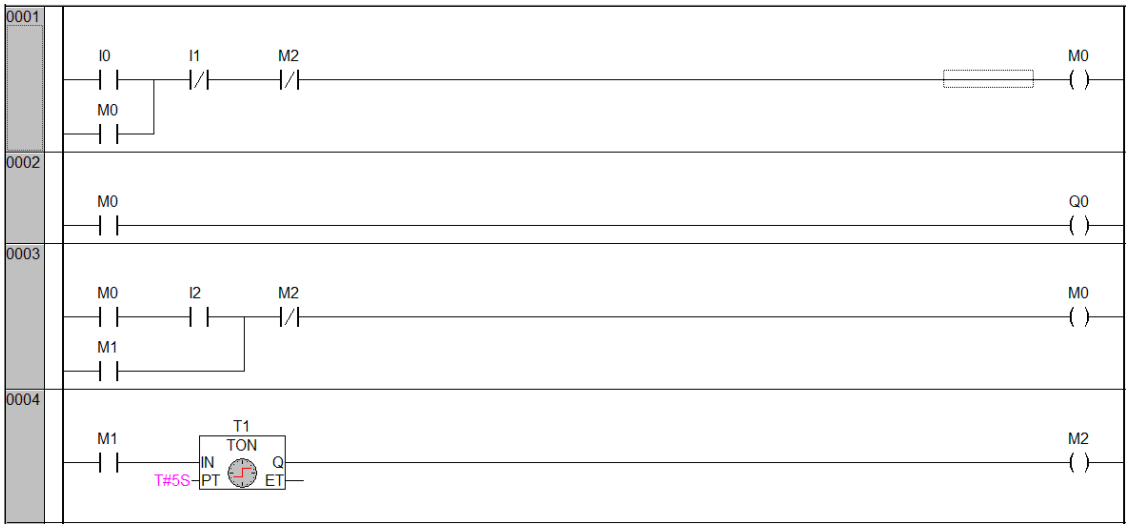
**Program for single acting cylinders:**

**EX67:** There are two push buttons I0 and I1 for start and stop. There is a single acting cylinder with valve Q0. When I0 is pressed piston will move forward and when I1 is pressed, piston should move in reverse direction.

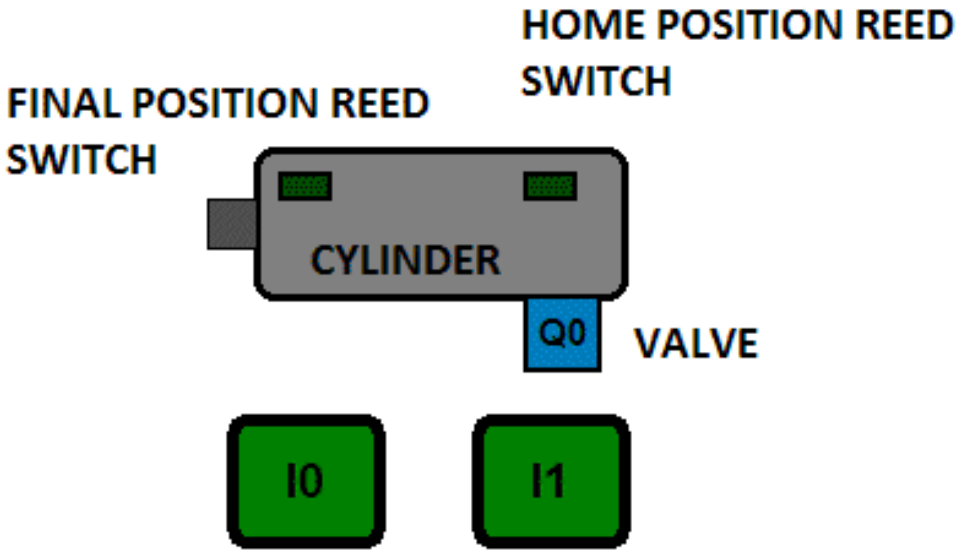


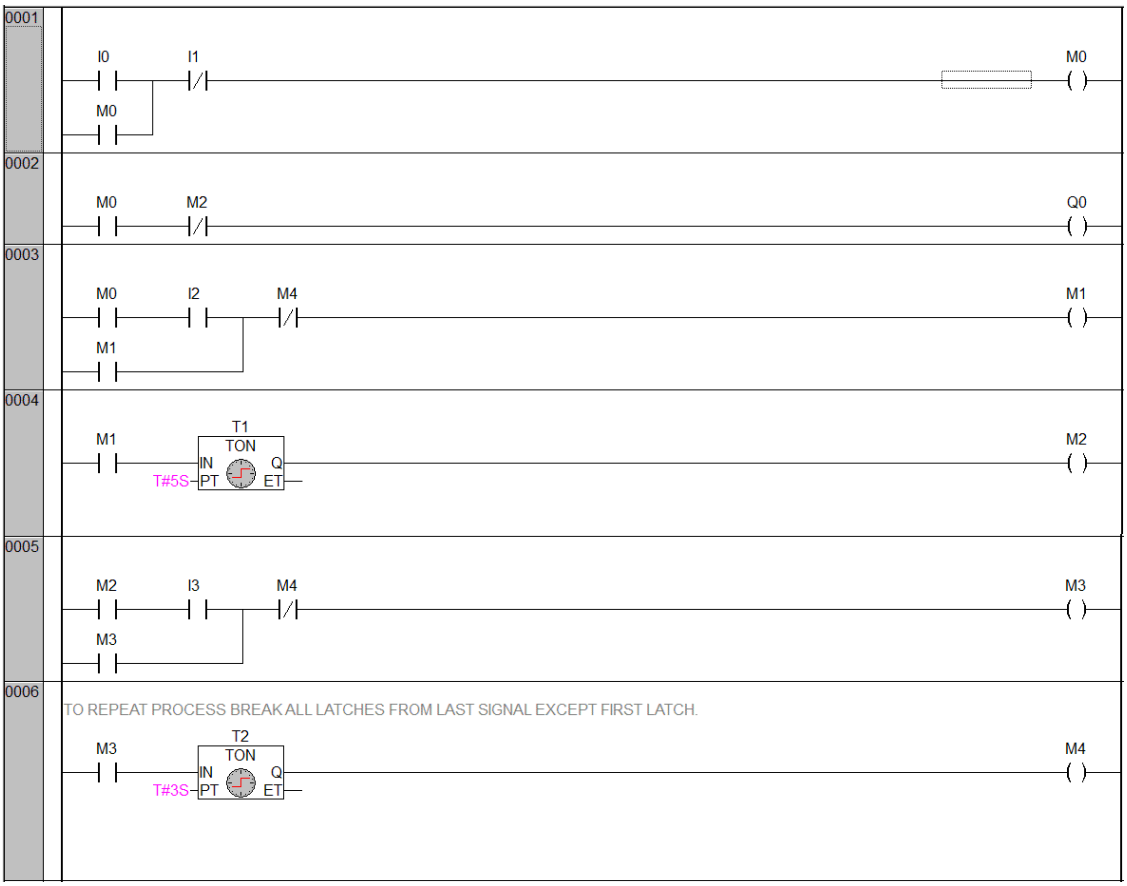

**EX68:** There are two push buttons I0 and I1 for start and stop. There is a single acting cylinder with valve Q0. There is a reed switch I2 on cylinder to sense the final position of piston. When I0 is pressed piston will move forward. As piston reaches to final position I2 will be on. When I2 is on, piston will wait for five seconds in final position. After five seconds piston will move back. I1 is for emergency stop.





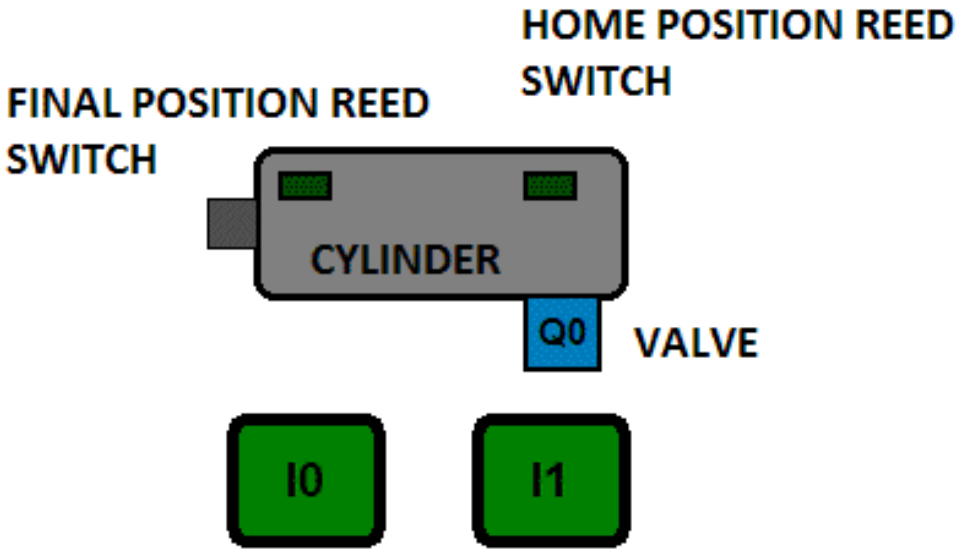

**EX69:** There are two push buttons I0 and I1 for start and stop. There is a single acting cylinder with valve Q0. There is a reed switch I2 on cylinder to sense the final position of piston. There is a reed switch I3 on cylinder to sense the home position of piston. When I0 is pressed piston will move forward. As piston reaches to final position I2 will be on. When I2 is on, piston will wait for five seconds in final position. After five seconds piston will move back and will reach to home position. Now as home position sensor I3 gets on, piston will wait for 3 seconds and after 3 seconds it will repeat process. I1 is for emergency stop.

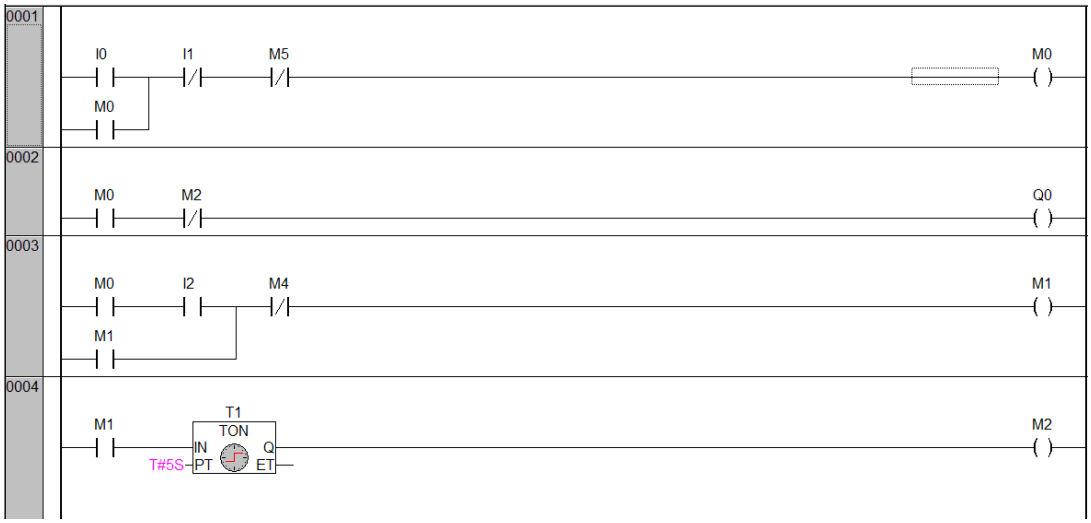


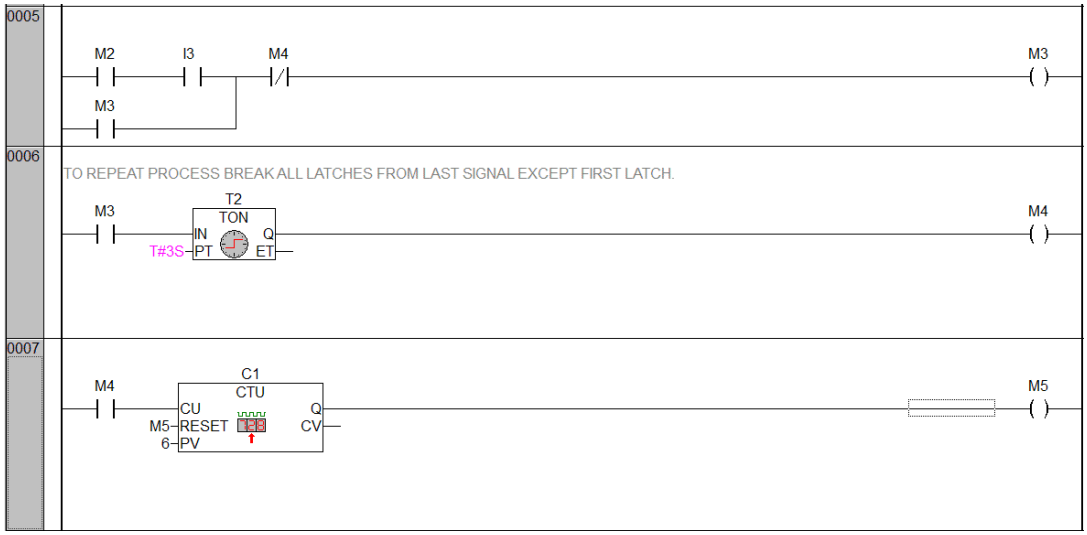



**EX70:** There are two push buttons I0 and I1 for start and stop. There is a single acting cylinder with valve Q0. There is a reed switch I2 on cylinder to sense the final position of piston. There is a reed switch I3 on cylinder to sense the home position of piston. When I0 is pressed piston will move forward. As piston reaches to final position I2 will be on. When I2 is on, piston will wait for five seconds in final position. After five seconds piston will move back and will reach to home position. Now as home position sensor I3 gets on, piston will wait for 3 seconds and after 3 seconds it will repeat

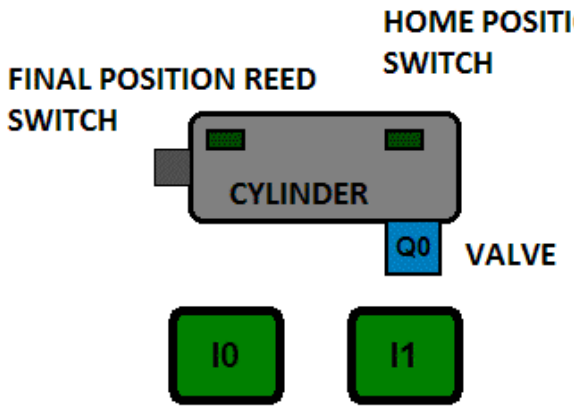
process for six cycles. I1 is for emergency stop.



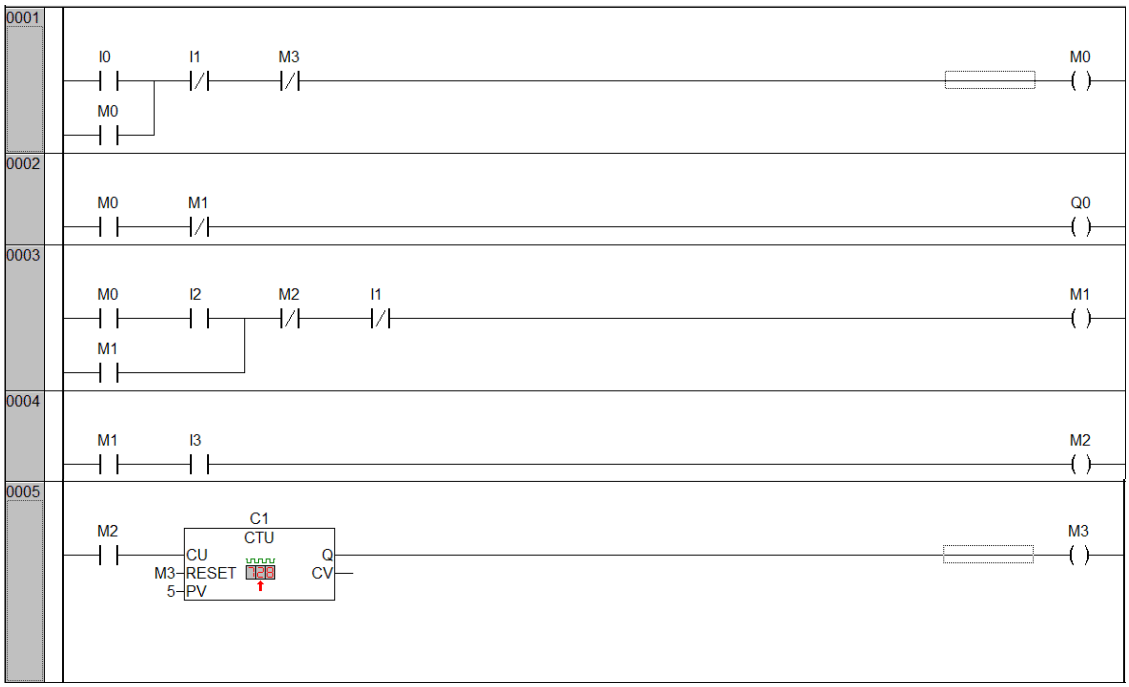
**EX71:** There are two push buttons I0 and I1 for start and stop. There is a single acting cylinder with valve Q0. There is a reed switch I2 on cylinder to sense the final position of piston. There is a reed switch I3 on cylinder to sense the home position of piston.




I0 on □ Q0 on.

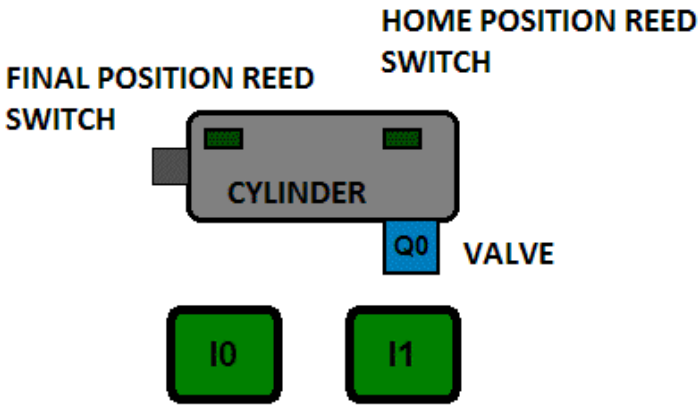
I2 on □ Q0 off.

I3 on □ repeat the process for 5 cycles.




**EX72:** There are two push buttons I0 and I1 for start and stop. There is a single acting cylinder with valve Q0. There is a reed switch I2 on cylinder to

sense the final position of piston. There is a reed switch I3 on cylinder to sense the home position of piston.




I0 on  forward/ Q0 on

I2on  reverse/ Q0 off

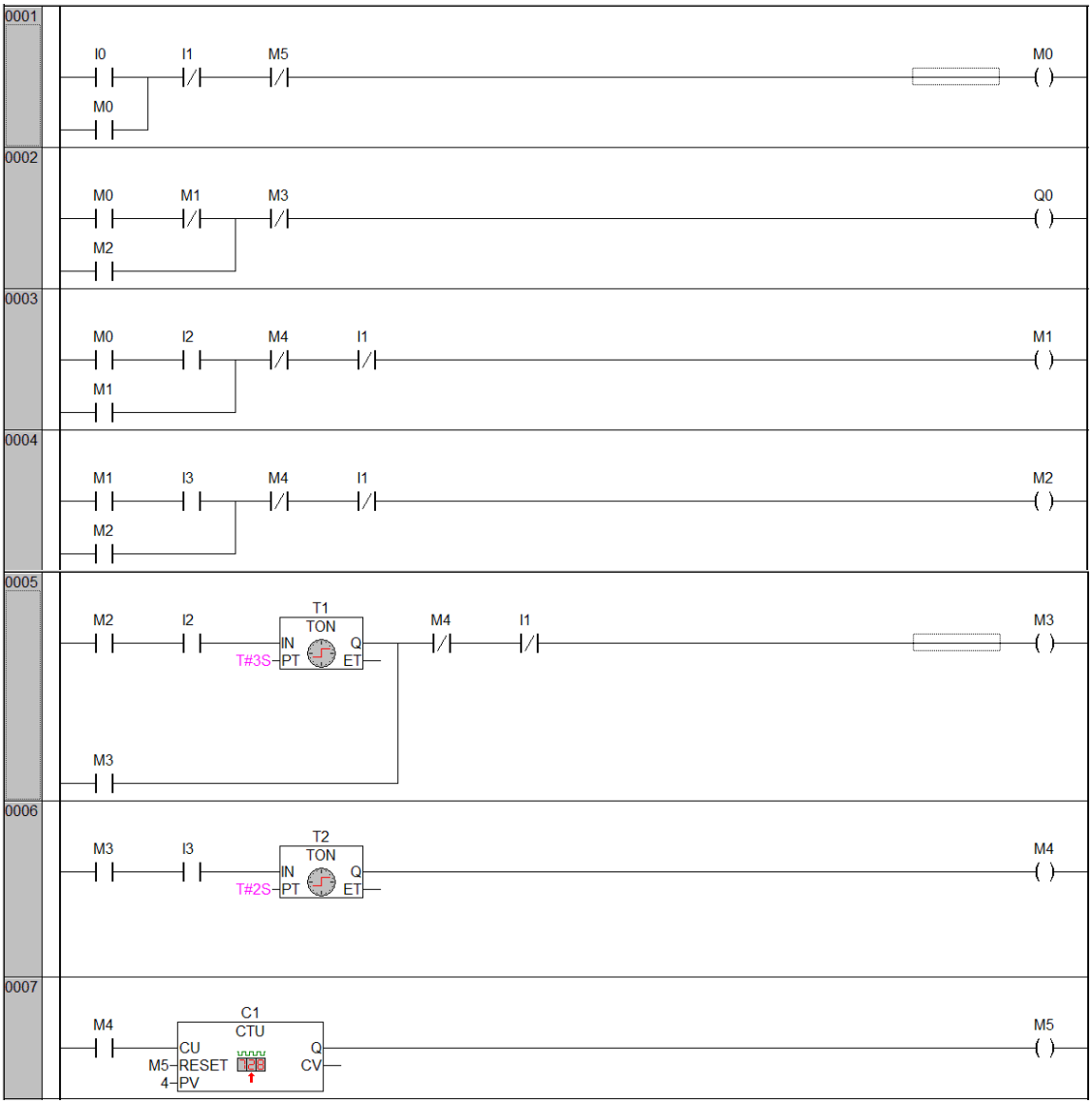
I3 on  forward/ Q0 on

I2 on  wait for 3 seconds

After 3 seconds  reverse/ Q0 off

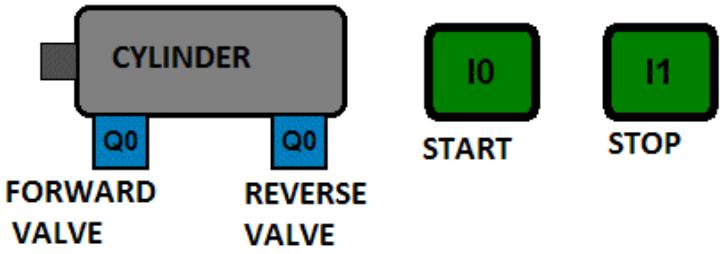
I3 on  wait for 2 seconds

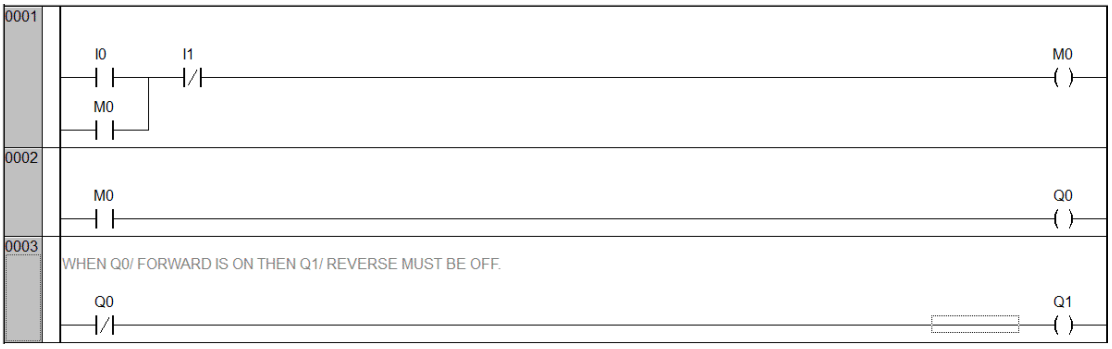
After 2 seconds  repeat the process for 4 cycles.



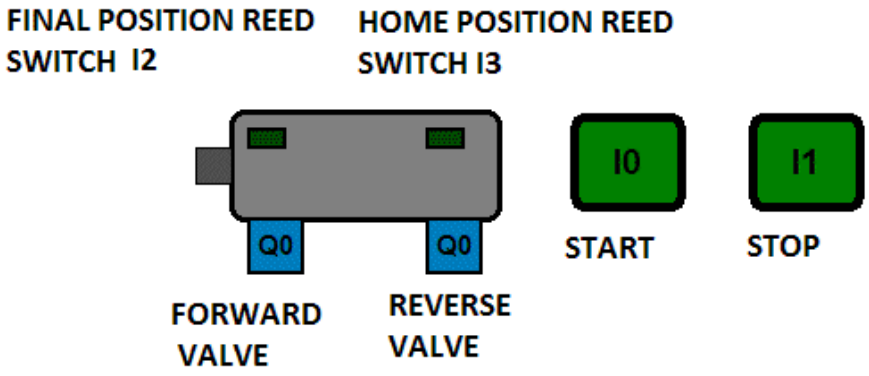

## DOUBLE ACTING CYLINDERS:

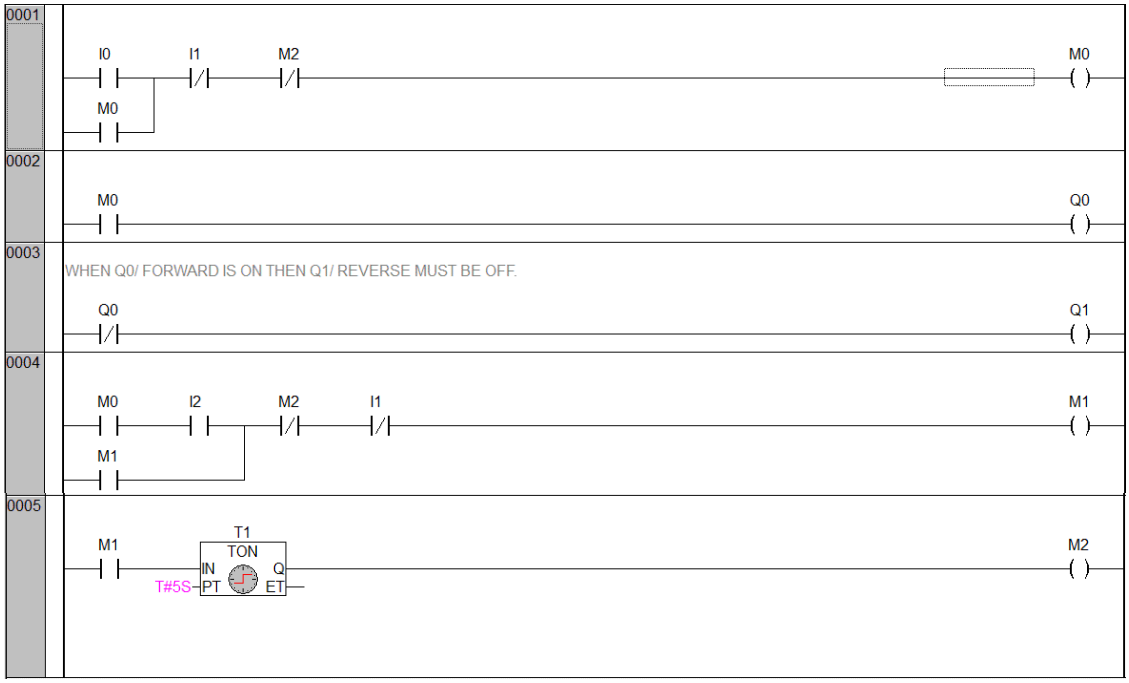
**EX73:** There are two push buttons I0 and I1 for start and stop. Q0 valve for forward and Q1 valve for reverse. When I0 is pressed piston should move in forward direction and when I1 is pressed piston should move in reverse direction.



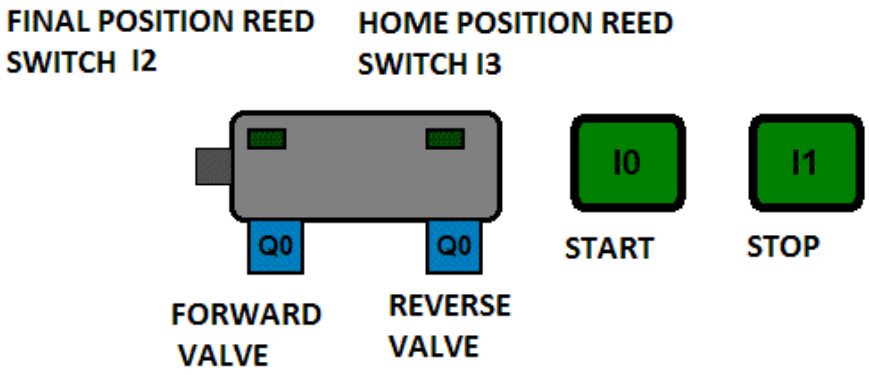



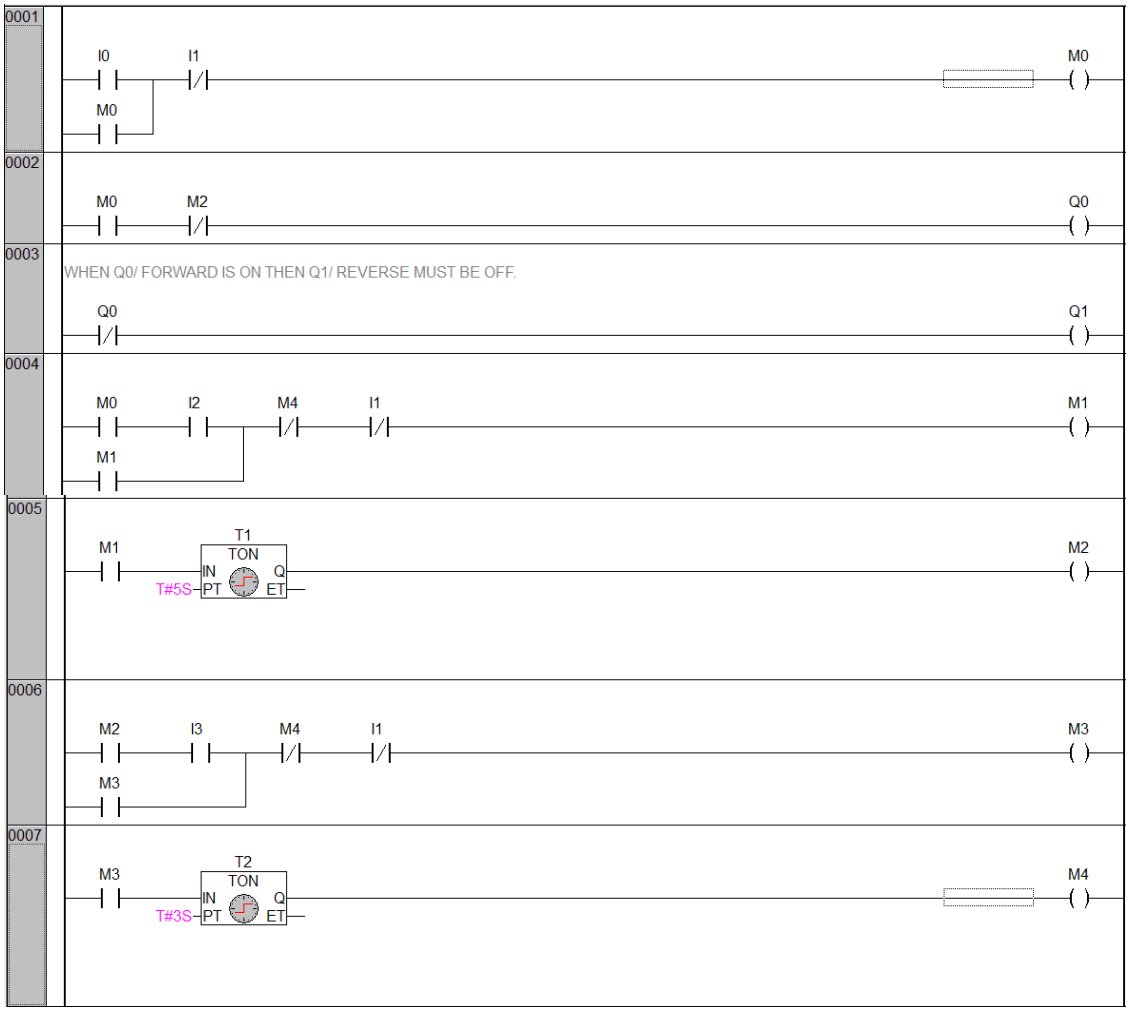

**EX74:** There are two push buttons I0 and I1 for start and stop. I2 is final position reed switch. Q0 valve for forward and Q1 valve for reverse. When I0 is pressed piston should move in forward direction. When piston reaches to final position I2 is on. As I2 is on, piston will wait for 5 seconds and after 5 seconds piston should move in reverse direction. I1 is an emergency off button.



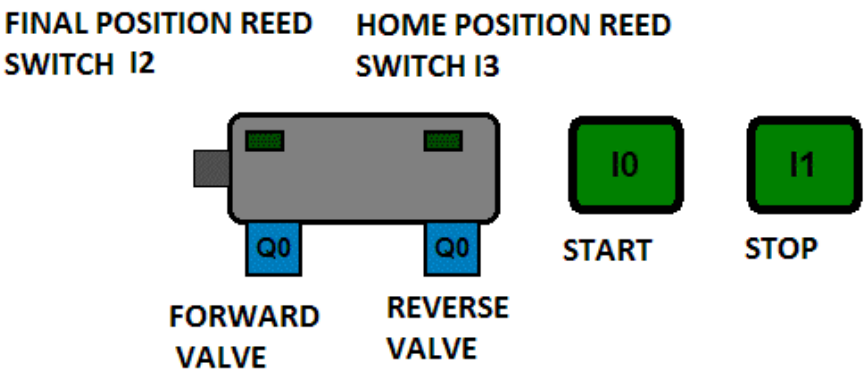


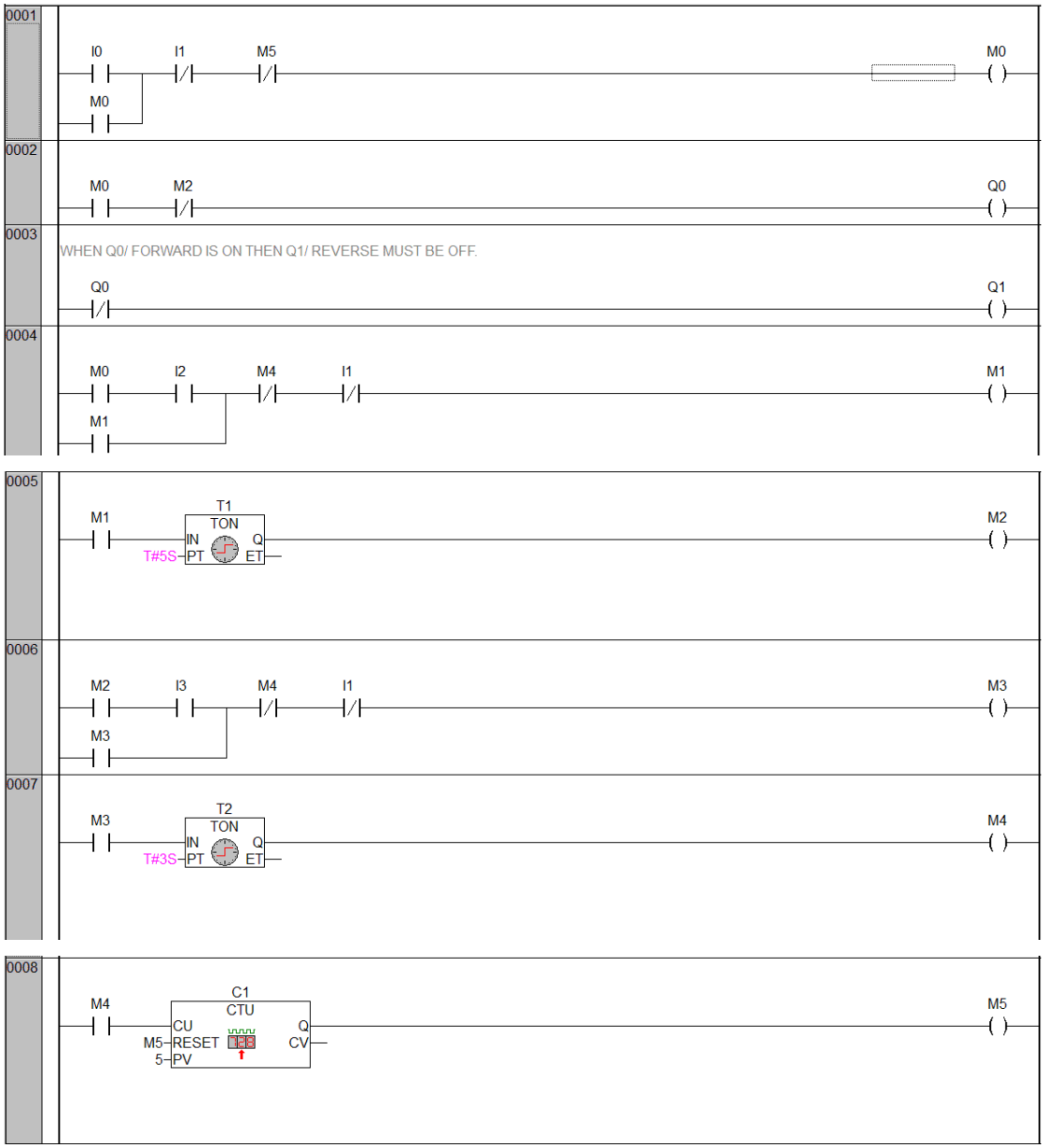

**EX75:** There are two push buttons I0 and I1 for start and stop. I2 is final position reed switch. I3 is home position reed switch. Q0 valve for forward and Q1 valve for reverse. When I0 is pressed piston should move in forward direction. When piston reaches to final position I2 is on. As I2 is on, piston will wait for 5 seconds and after 5 seconds piston should move in reverse direction. When piston reaches to home position I3 is on. As I3 is on, piston will wait for 3 seconds and after 3 seconds repeat the process. I1 is an emergency off button.





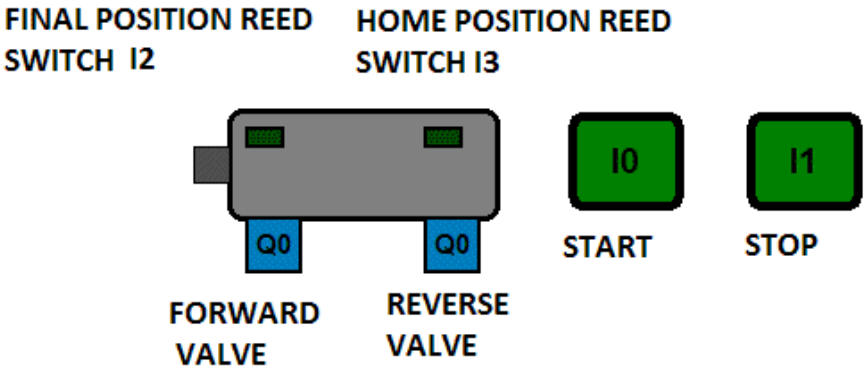

**EX76:** There are two push buttons I0 and I1 for start and stop. I2 is final position reed switch. I3 is home position reed switch. Q0 valve for forward and Q1 valve for reverse. When I0 is pressed piston should move in forward direction. When piston reaches to final position I2 is on. As I2 is on, piston will wait for 5 seconds and after 5 seconds piston should move in reverse direction. When piston reaches to home position I3 is on. As I3 is on, piston will wait for 3 seconds and after 3 seconds repeat the process FOR 5 CYCLES. I1 is an emergency off button.

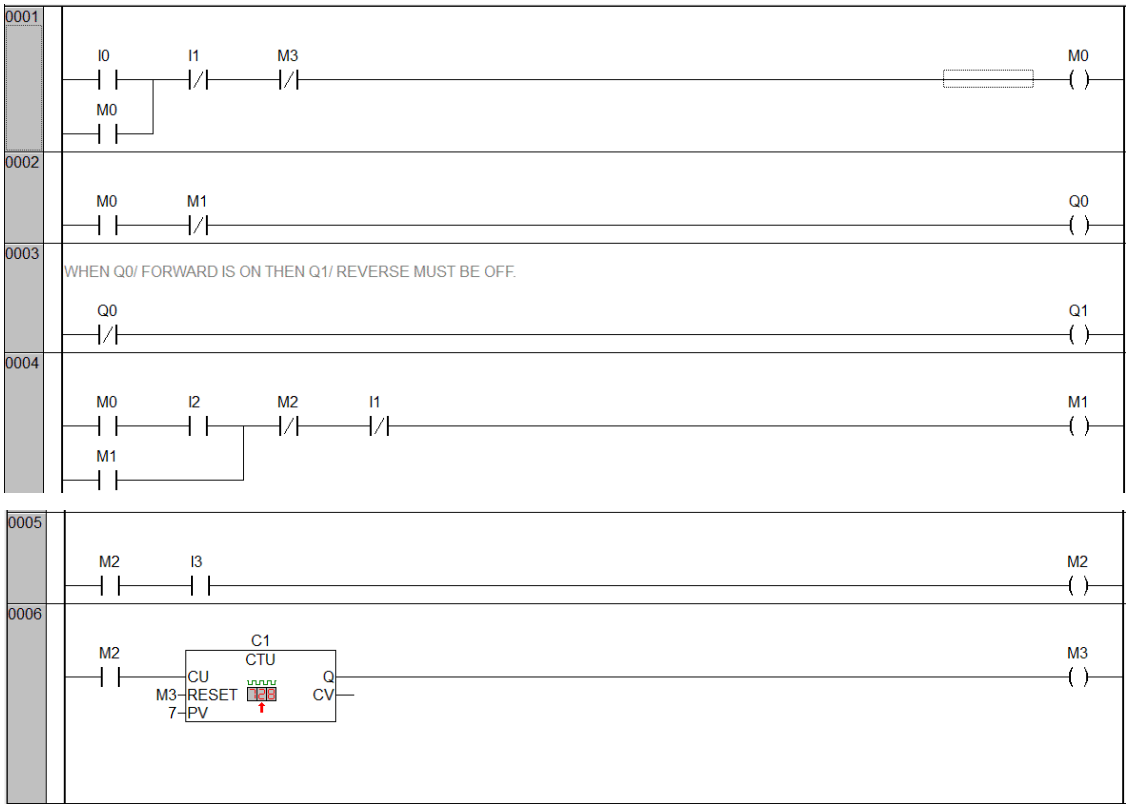



**EX77:** There are two push buttons I0 and I1 for start and stop. I2 is final position reed switch. I3 is home position reed switch. Q0 valve for forward

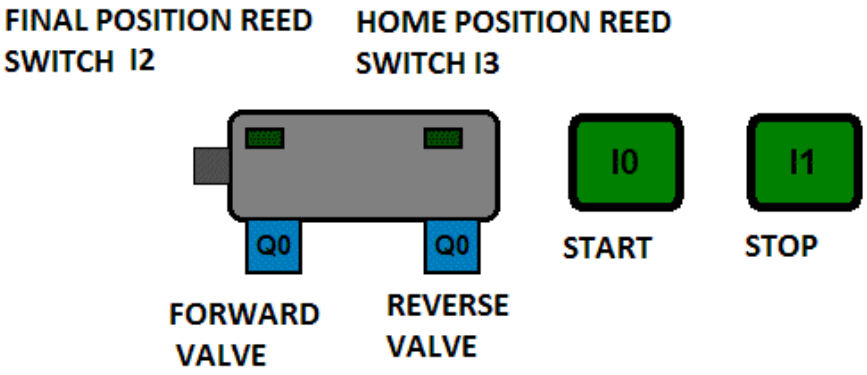
and Q1 valve for reverse. When I0 is pressed piston should move in forward direction. When piston reaches to final position I2 is on. As I2 is on, piston will move in reverse direction. When piston reaches to home position I3 is on. As I3 is on, repeat the process FOR 7 CYCLES. I1 is an emergency off button.

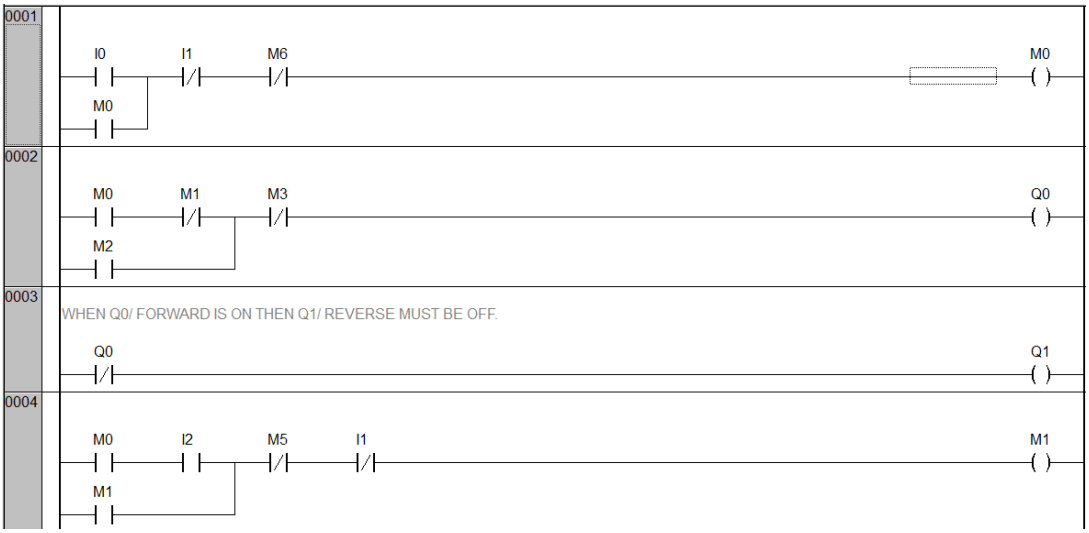


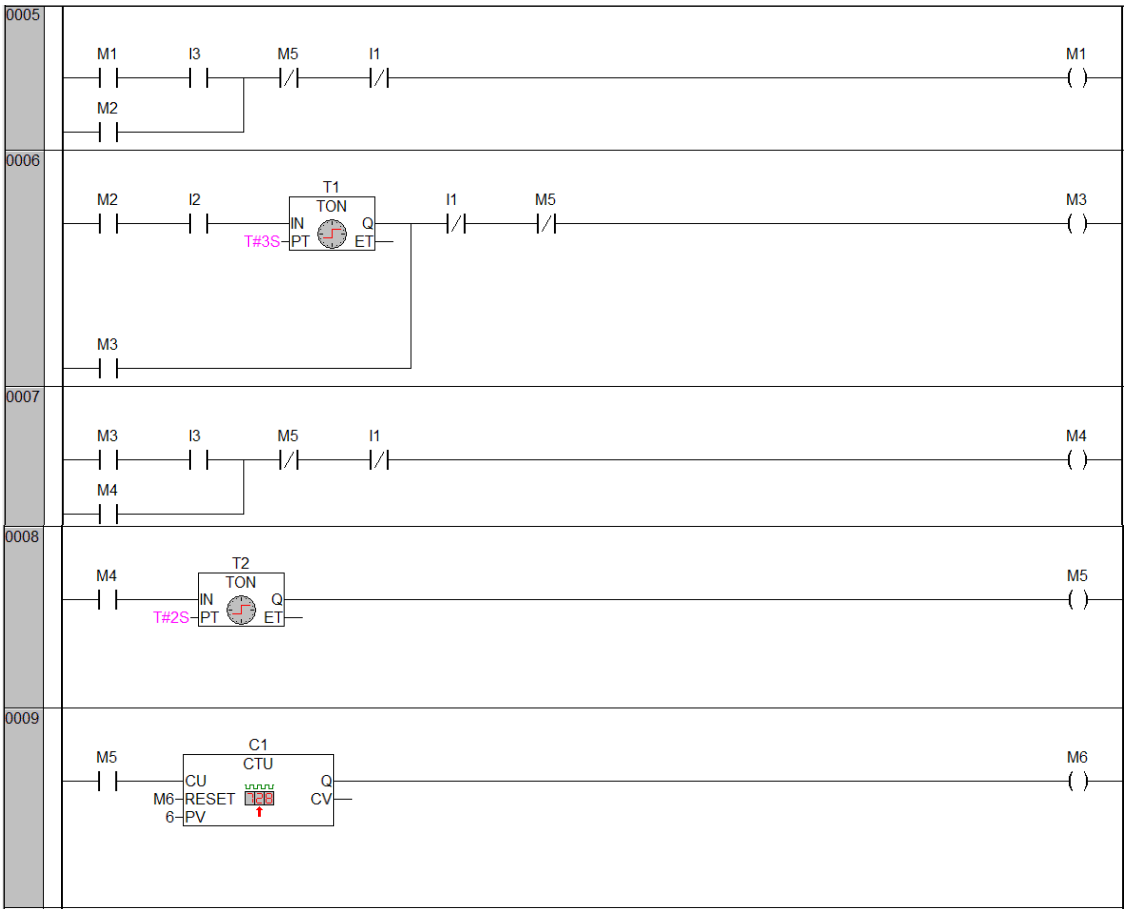




**EX78:** There are two push buttons I0 and I1 for start and stop. I2 is final position reed switch. I3 is home position reed switch. Q0 valve for forward and Q1 valve for reverse. When I0 is pressed piston should move in forward direction. When piston reaches to final position I2 is on. As I2 is on, piston will move in reverse direction. When piston reaches to home position I3 is on. As I3 is on, piston will move in forward direction. As it reaches to final

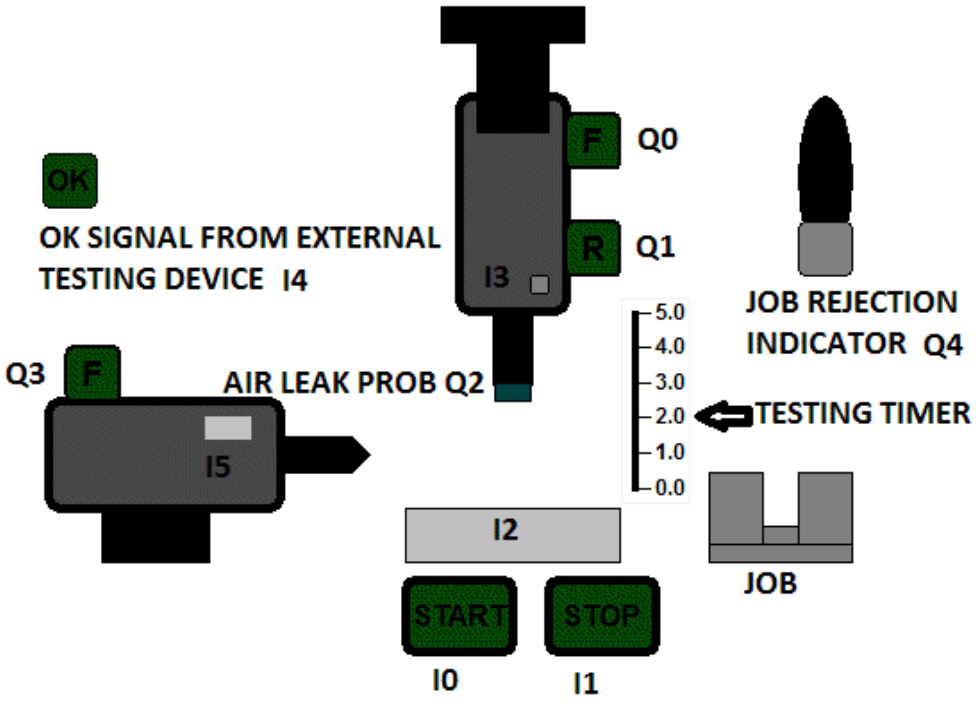
position, it will wait for 3 seconds. After 3 seconds it will move in reverse direction. As it reaches to home position it will wait for 2 seconds and after 2 seconds it will repeat process for 6 cycles. I1 is an emergency off button.



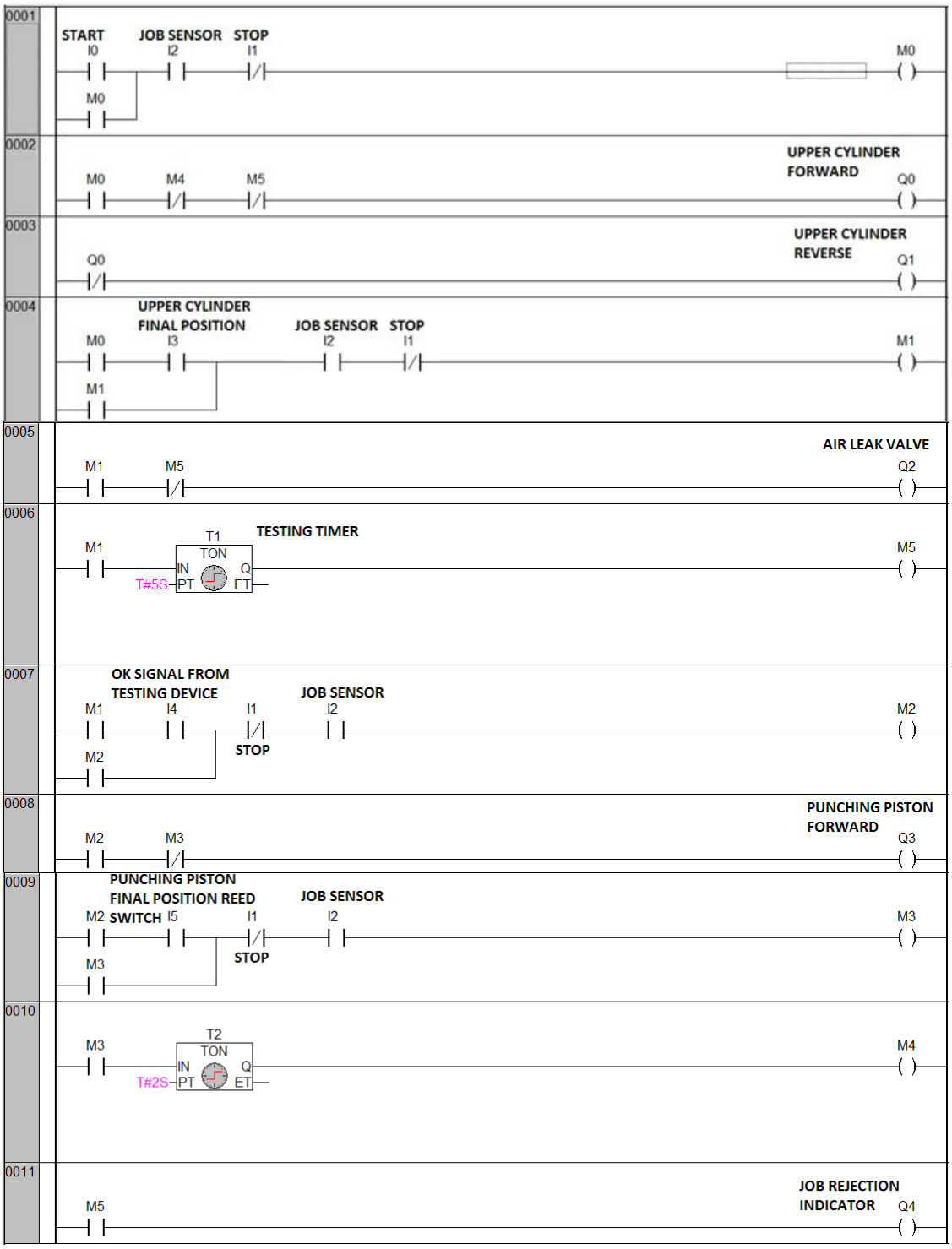



## EX79: JOB TESTING/ SPECIAL PURPOSE MACHINE (SPM)

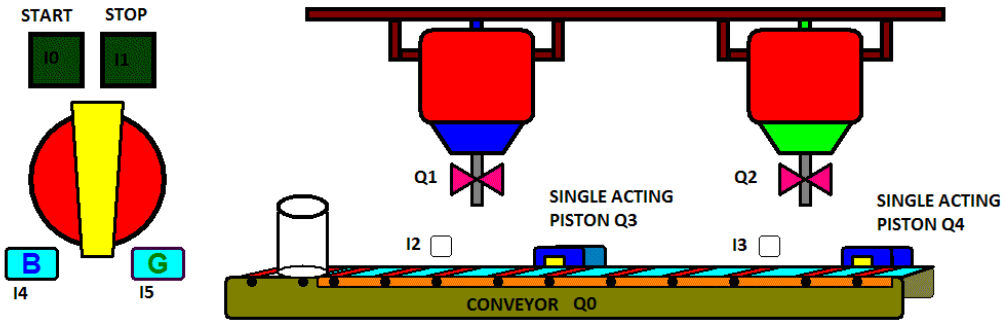



When metallic job is put on work station, proximity sensor I2 gets on. It confirms the presence of job on work station and it must be on to run the machine. I0 start push button is pressed and upper cylinder piston moves forward i. e. Q0 is on. An air leak probe is attached with this piston which will leak air inside job to check the surface accuracy of the job. As piston reaches final position/ I3 is on, Q2 will get on and air will be leaked inside the job for five seconds. If OK signal I4 comes within five seconds then Q2 will be off and Q3 will be on resulting punching piston move forward to punch a mark on job. As piston reaches to final position/ I5 is on, Q3 will get off and piston will return back to home position. Now system will wait for 2 seconds and after two seconds upper cylinder piston will move back i. e. Q0 off and Q1 on. System will not start until tested job is removed from the work station. If one cycle is finished and I0 is pressed, job is not removed and you press I0 then system should not start.

Another condition for job test is; if OK signal I4 does not come within 5 seconds and timer done bit gets high then air leak Q2 will be off and job rejection indicator Q4 will be on. At the same time upper cylinder piston will move in reverse direction i. e. Q0 is off and Q1 is on. Q4 indicator and system will get reset by removing tested job from work station.




## EX80: BOTTLE FILLING AS PER SELECTION OF PRODUCT INCLUDING PNEUMATIC STOPPERS.




When neither I4 nor I5 is selected:

I0 on □ Q0 on

I2 on □ Q3 on for 6 seconds and Q1 on for 5 seconds.

I3 on □ Q4 on for 6 seconds and Q2 on for 5 seconds.

When I4 is selected:

I2 on □ Q3 on for 11 seconds and Q1 on for 10 seconds.

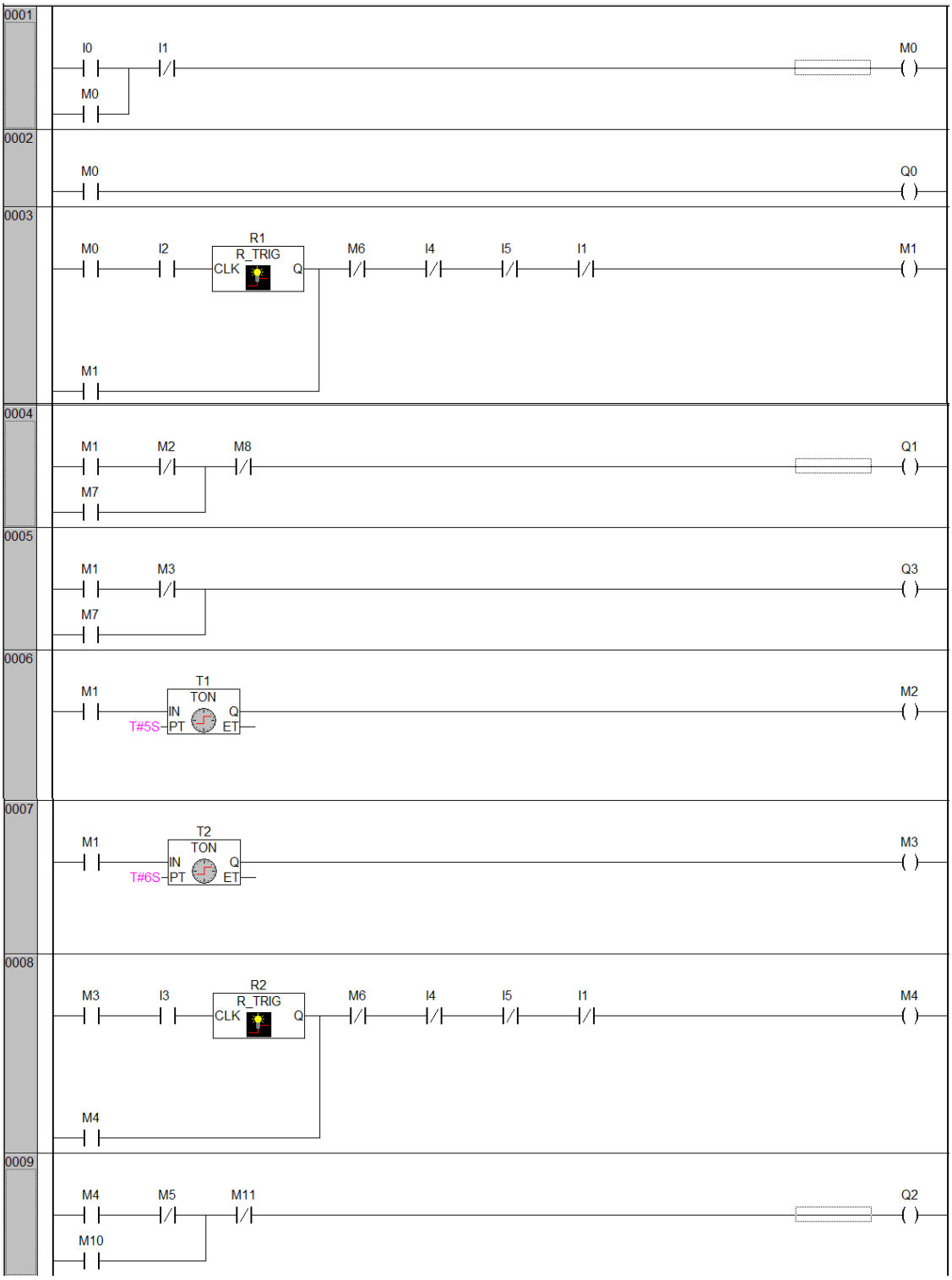
I3 on □ nothing happens.

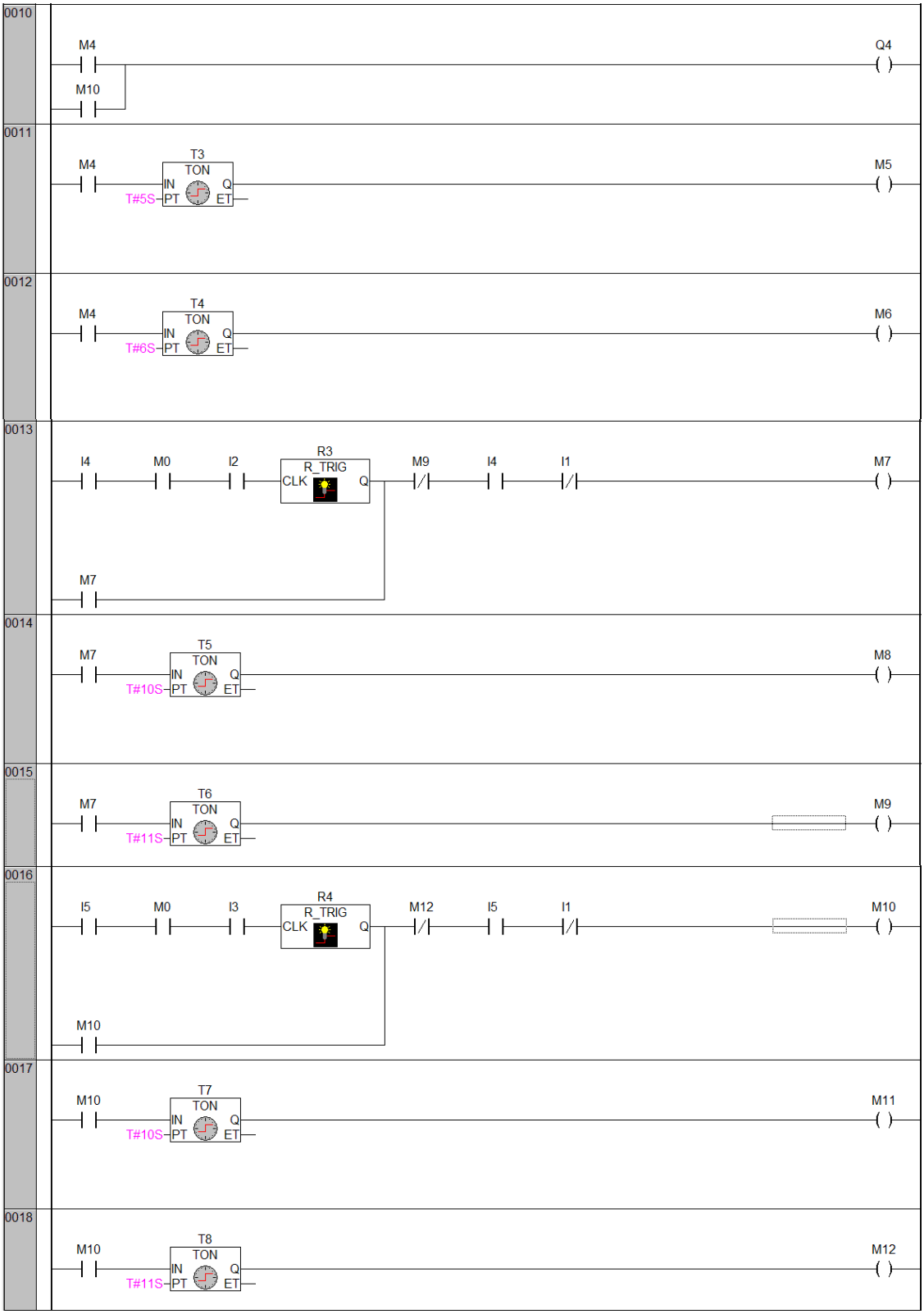
When I5 is selected:

I2 on □ nothing happens.

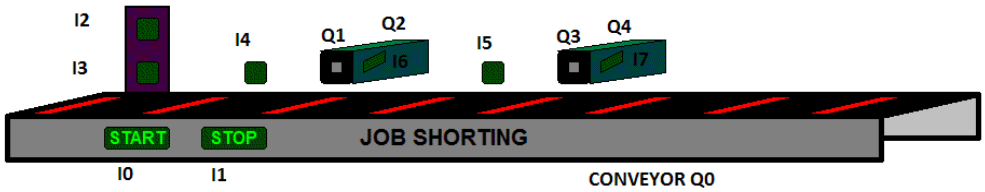
I3 on □ Q4 on for 11 seconds and Q2 on for 10 seconds.

I1 is emergency stop button.


0001
I0
I1
M0
M0
0002
M0
Q0
0003
M0
I2
R1
R_TRIG
CLK
Q
M6
I4
I5
I1
M1
M1
0004
M1
M2
M8
Q1
M7
0005
M1
M3
Q3
M7
0006
M1
T1
TON
IN
PT
T#5S
Q
ET
M2
0007
M1
T2
TON
IN
PT
T#6S
Q
ET
M3
0008
M3
I3
R2
R_TRIG
CLK
Q
M6
I4
I5
I1
M4
M4
0009
M4
M5
M11
Q2
M10

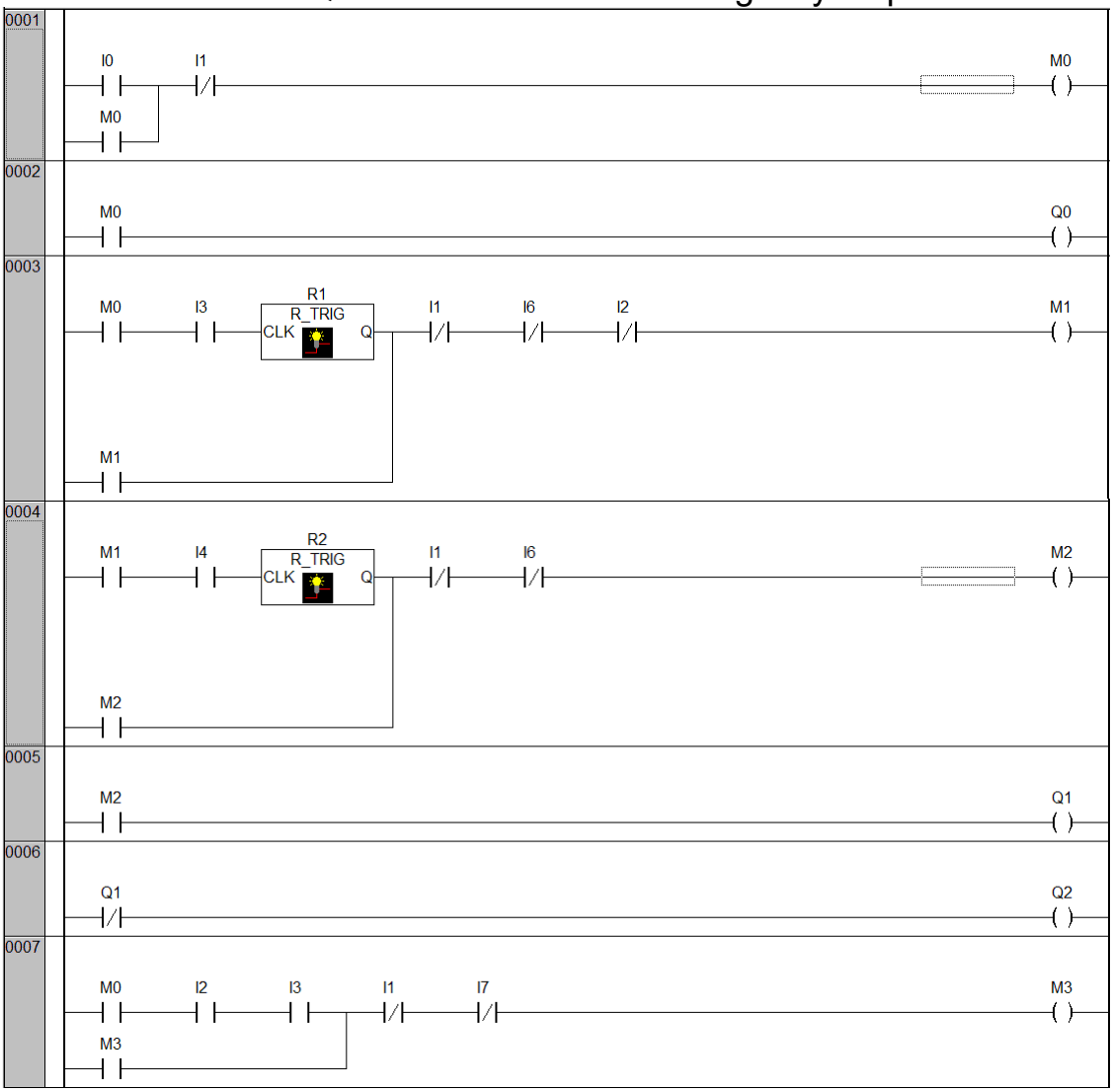


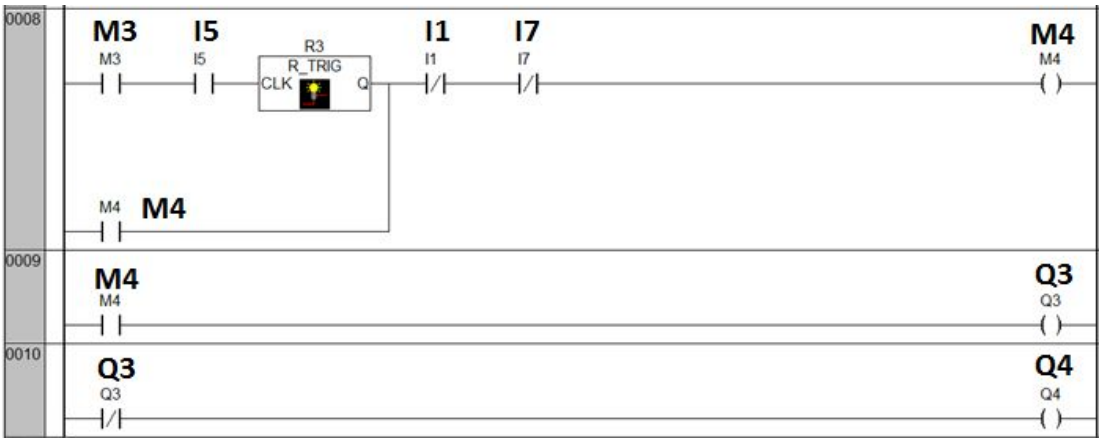

## EX81: Job shorting machine



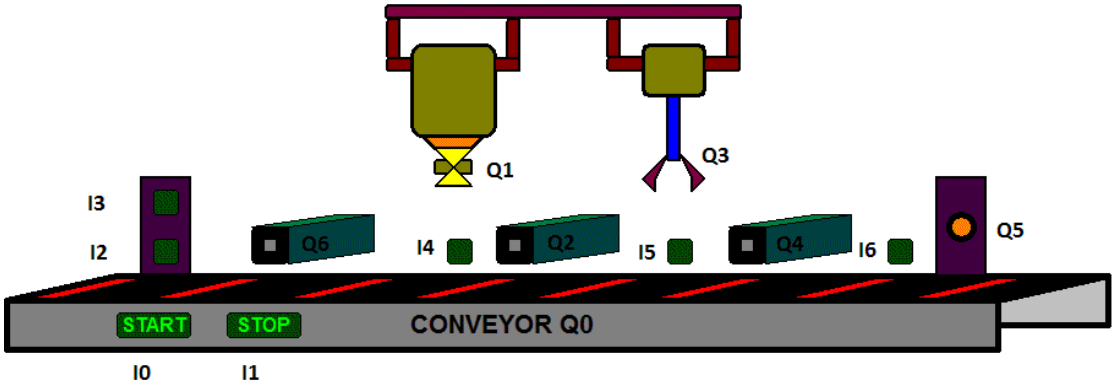

When I0 is on conveyor Q0 will be on. If only I3 lower sensor is on and after that I4 sensor gets on, stopper 1 forward Q1 will be on. When stopper 1 final position reed switch I6 gets on then Q1 will be off and reverse valve Q2 will be on. When I5 sensor gets on, nothing will happen.

If I2 and I3 both sensors get on together and after that I5 sensor gets on, nothing will happen. When I5 sensor gets on stopper 2 forward Q3 will be on. When stopper 2 final position reed switch I7 gets on then Q3 will be off and reverse valve Q4 will be on. I1 is an emergency stop button.



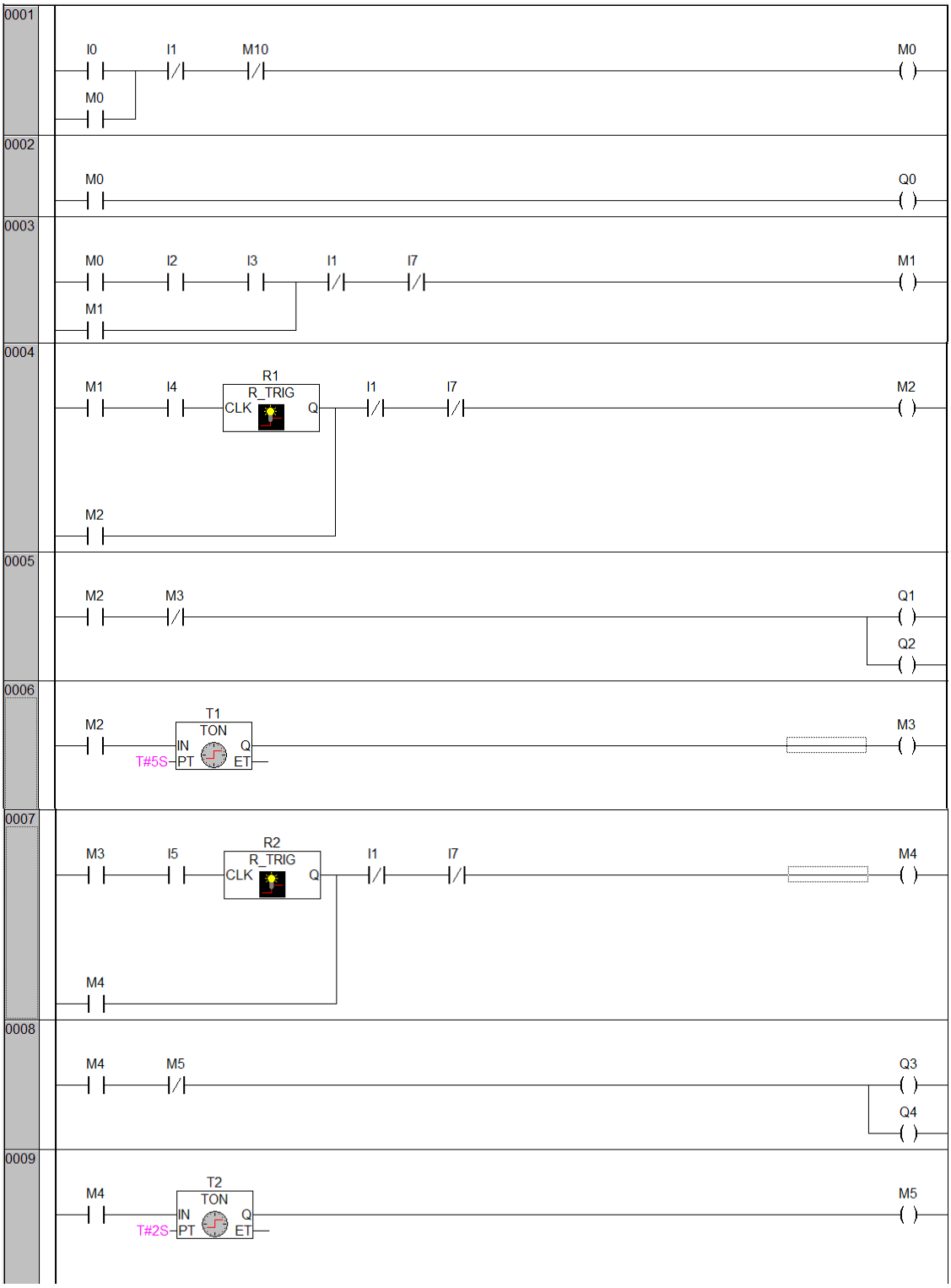


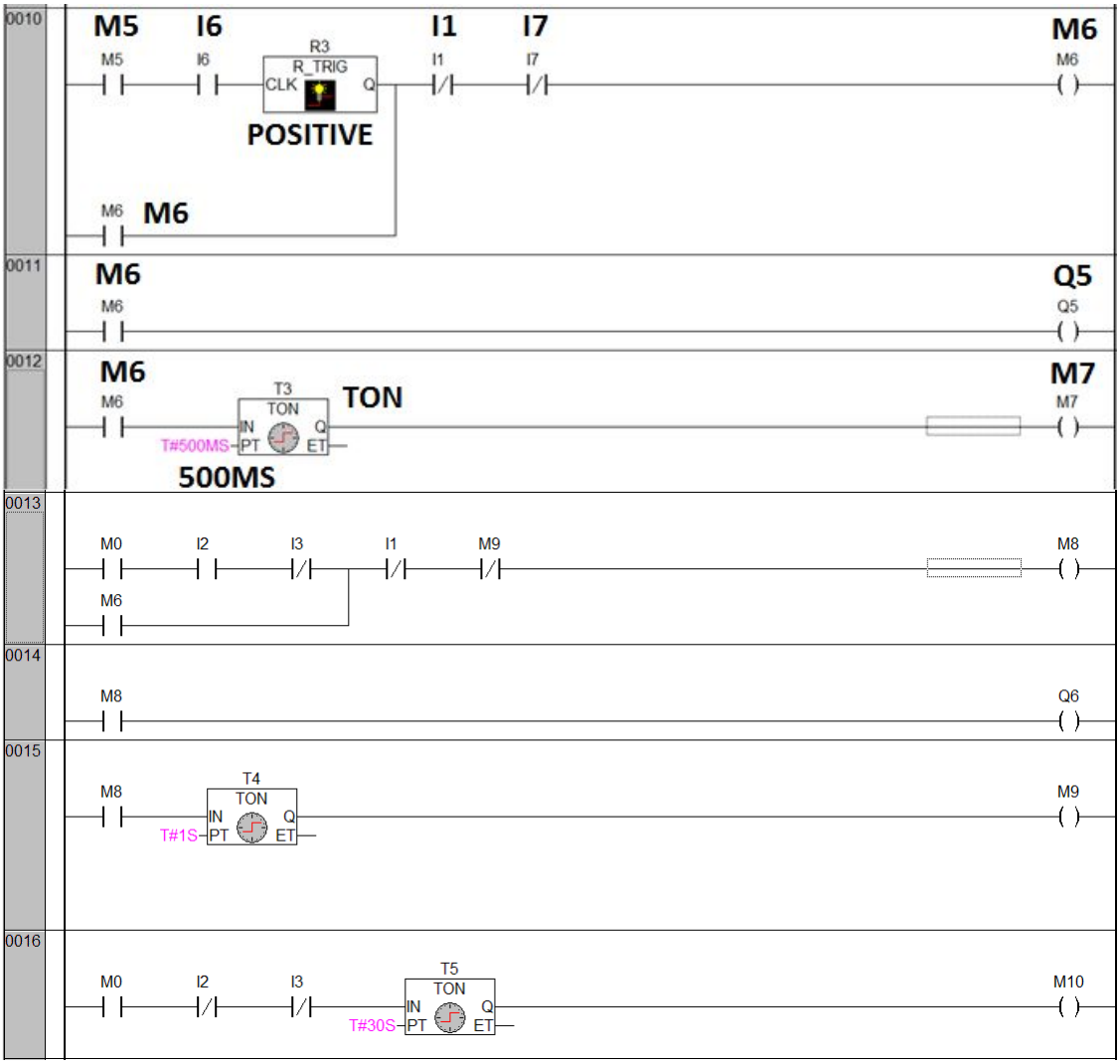

**EX82: multifunction machine (filling, capping and labelling)**



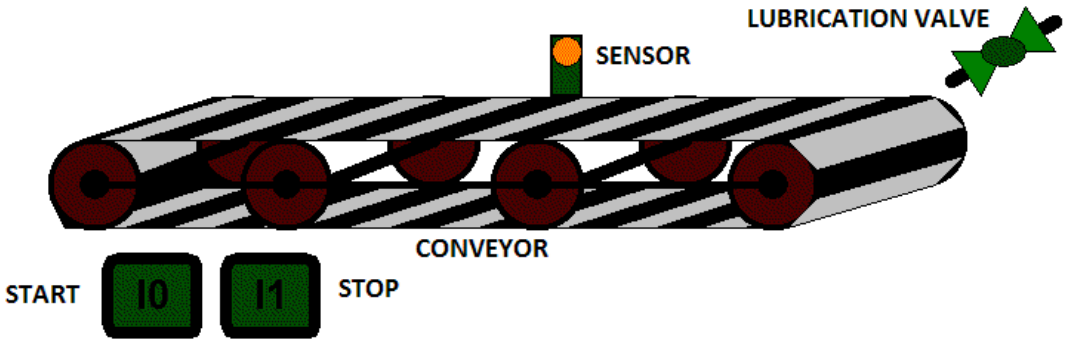

I0 is start push button, I1 is stop push button, I2 lower sensor, I3 upper sensor, I4 filling position bottle detector, I5 capping position bottle detector, I6 labelling position bottle detector, Q0 is conveyor, Q1 is filling valve, Q2 is single acting stopper 1, Q3 is capping output, Q4 is single acting stopper 2, Q5 is laser printer output and Q6 is single acting falling bottle ejector.

When I0 is pressed conveyor will be on. When I2 and I3 are on together and I4 is on, Q1 and Q2 will get on for 5 second. After this when I5 is on then Q3 and Q4 will be on for 2 seconds. After this when I6 is on then Q5 will be on for half second. If only I2 will get on then Q6 will be on for 1 second. If neither I2 nor I3 gets on for 30 seconds then system will get off automatically.  I1 is for emergency stop.


0001
I0
I1
M10
M0
M0
0002
M0
Q0
0003
M0
I2
I3
I1
I7
M1
M1
0004
M1
I4
R1
R_TRIG
CLK
Q
I1
I7
M2
M2
0005
M2
M3
Q1
Q2
0006
M2
T1
TON
IN
PT
T#5S
Q
ET
M3
0007
M3
I5
R2
R_TRIG
CLK
Q
I1
I7
M4
M4
0008
M4
M5
Q3
Q4
0009
M4
T2
TON
IN
PT
T#2S
Q
ET
M5



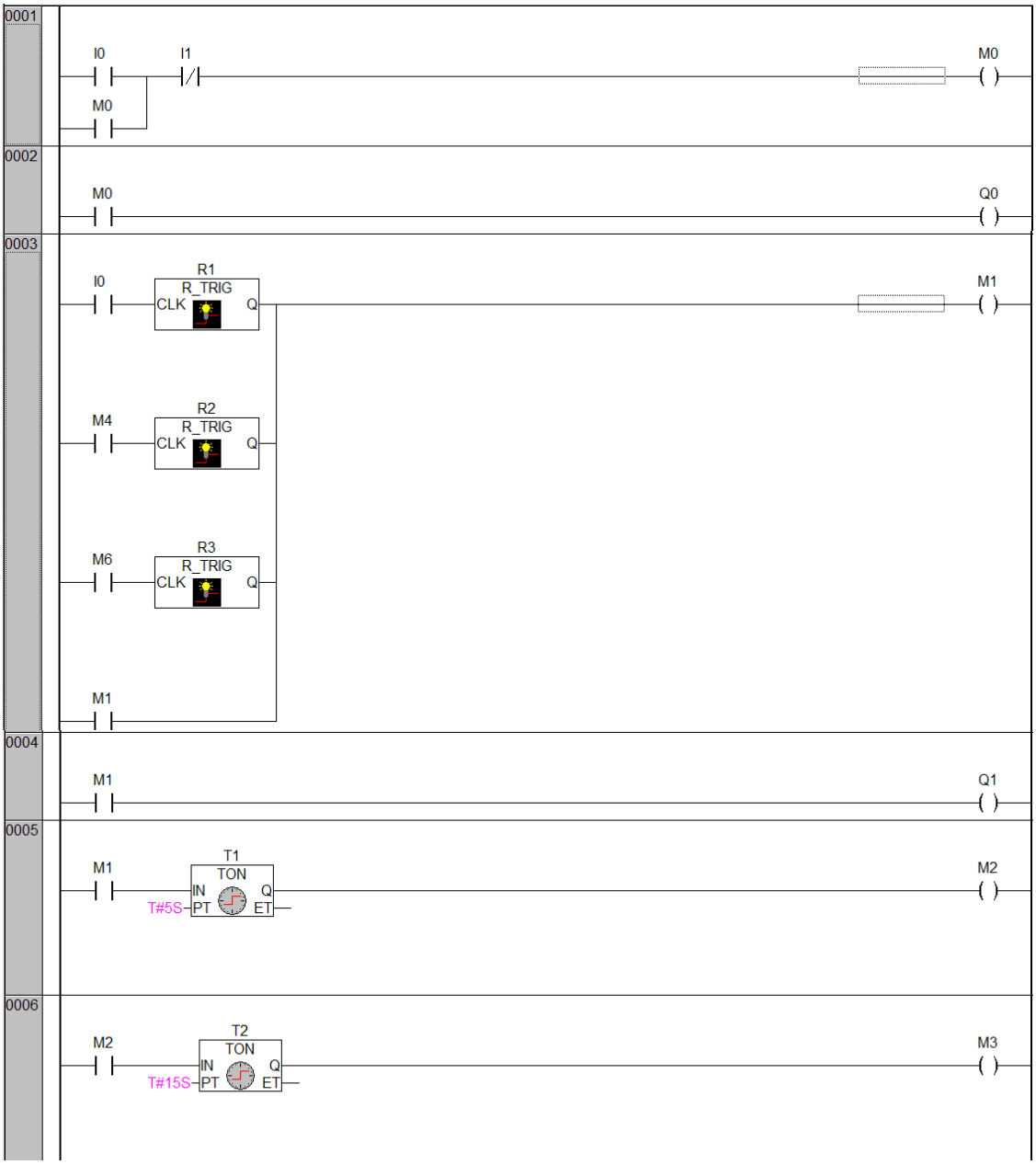

## EX83: Automatic lubrication

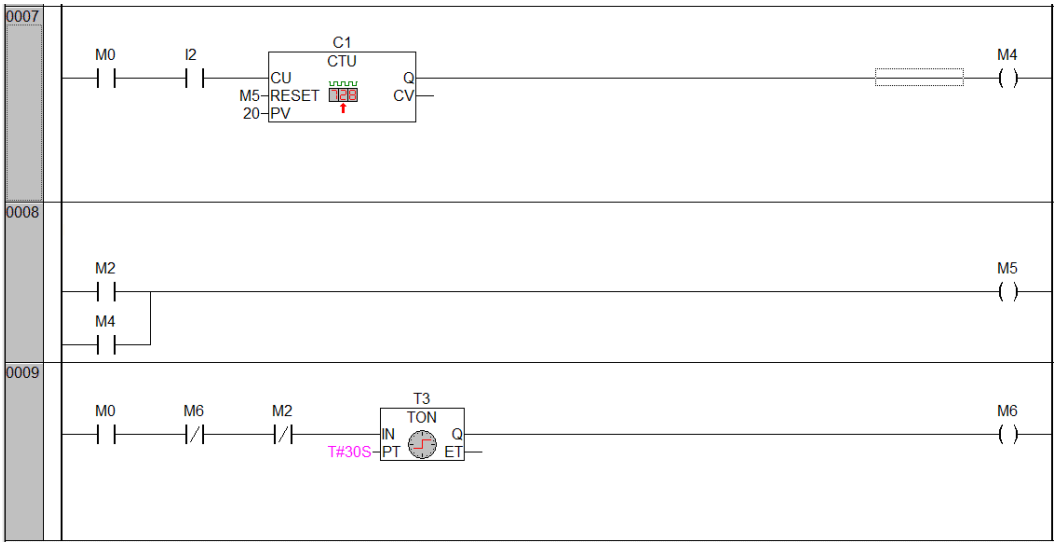



Conditions for lubrication:

1> Whenever I0 is pressed.
2> Whenever I2 finishes 20 jobs count.
3> After each 30 seconds of one lubrication.

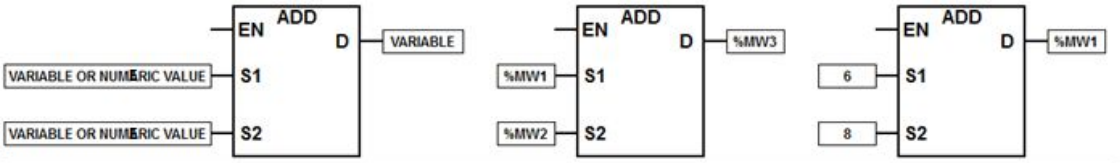
It must be gap of 15 seconds between two lubrications.





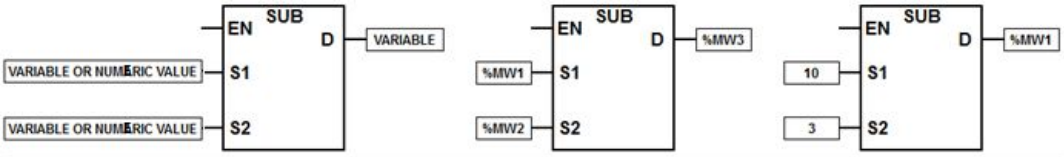

## Arithmetic operation (ADD, SUB, MUL, DIV, INC and DEC)

### ADD: In arithmetic operation destination is always a variable.



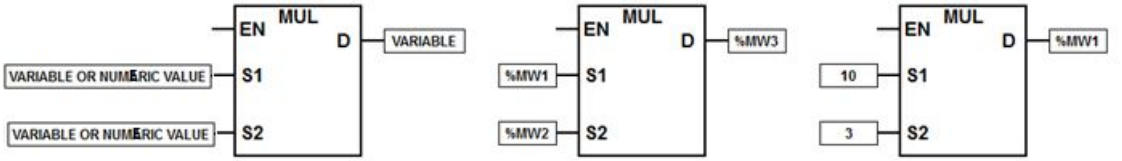

| |
|---|
| S1 is source1, S2 is source2 and D is destination, S1 and S2 can be a variable address or it can be a numaric value. |
| %MB, %MW, %MD, %VB, VW, %VD, ACC, N7, D are some example of variable address in different brand of PLC software. |
| In addition S1+ S2= D |

### SUB:



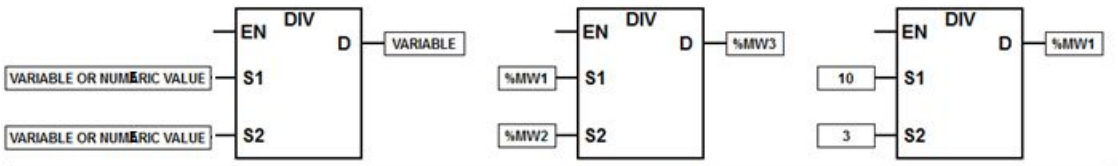

| |
|---|
| S1 is source1, S2 is source2 and D is destination, S1 and S2 can be a variable address or it can be a numaric value. |
| %MB, %MW, %MD, %VB, VW, %VD, ACC, N7, D are some example of variable address in different brand of PLC software. |
| In subtraction S1- S2= D |

### MUL:



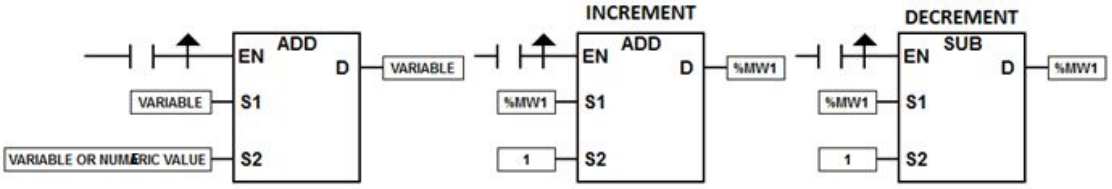

| |
|---|
| S1 is source1, S2 is source2 and D is destination, S1 and S2 can be a variable address or it can be a numaric value. |
| %MB, %MW, %MD, %VB, VW, %VD, ACC, N7, D are some example of variable address in different brand of PLC software. |
| In multiplication S1* S2= D |

**DIV:**



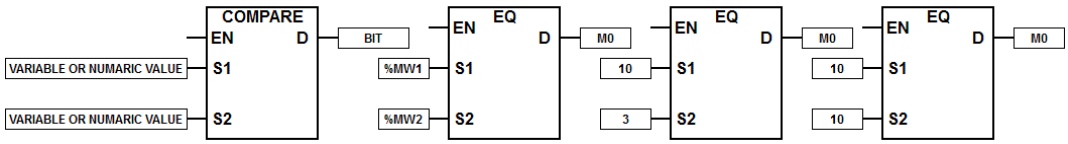

| |
|---|
| S1 is source1, S2 is source2 and D is destination, S1 and S2 can be a variable address or it can be a numaric value. |
| %MB, %MW, %MD, %VB, VW, %VD, ACC, N7, D are some example of variable address in different brand of PLC software. |
| In division S1/S2= D. In MUL and in DIV destination is double word. |
| IF you have taken MW1 as destination address then MW1 and MW2 both will be used automatically. |
| So donot use MW2 in that case,will not be used for another use in your program. |

You may get **increment and decrement** instruction in some PLC software. If you don't find it available then you can use ADD as increment and SUB as decrement.

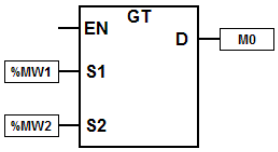


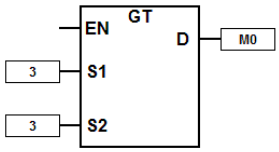

| |
|---|
| S1=D=Variable and S2 will be the value by which you want increment or decrement. Here we have taken increment and decrement by 1 in %mw1. |
| Enable must be a pulse signal or else it will start showing continue increment in destination. |
| Pulse is needed for one by one increment or decrement in destination on each press of enable. |
| If you want 2, 4, 6, ... increment or by decrement then put 2 in S2. How does it work? |
| Initially S1 and D has value 0. S2 have1. when enable is high S1 and S2 are added i. e. 0+1= 1 in destination. |
| As S1 and D have same address so now S1=D=1 and S2=1. In next enable signal again S1 and S2 will be added. |

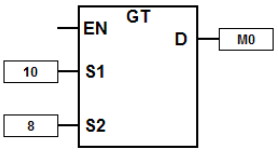
## COMPARE Instruction (EQ, GT, LT, GE, LE)



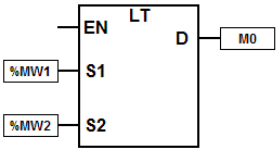

| |
|---|
| S1 is source1, S2 is source2 and D is destination, S1 and S2 can be a variable address or it can be a numaric value. D is always a bit. in compare. |
| In EQ (Equal) instruction if S1=S2 then D will be 1/ high. in D you can take a flag or an output address but it is better to take a flag. |
| You can see the example where 10 is being compared with 3 in EQ instruction. its destination will be zero/ low/ off. |
| You can see the example where 10 is being compared with 10 in EQ instruction. its destination will be one/ high/ on. |
| compare gives result as 0 or 1/ low or high. |

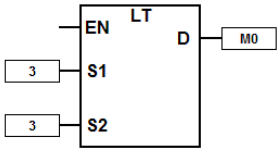


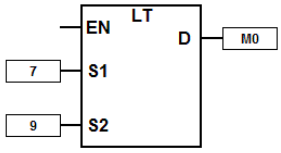



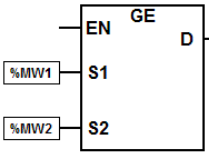

| |
|---|
| S1 is source1, S2 is source2 and D is destination, S1 and S2 can be a variable address or it can be a numaric value. D is always a bit. in compare. |
| In GT (GREATER THAN) instruction if S1>S2 then D will be 1/ high. in D you can take a flag or an output address but it is better to take a flag. |
| You can see the example where 3 is being compared with 3 in GT instruction. Its destination will be zero/ low/ off. |
| You can see the example where 10 is being compared with 8 in GTinstruction. its destination will be one/ high/ on. |
| compare gives result as 0 or 1/ low or high. |

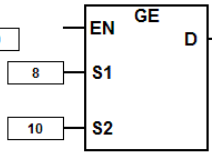


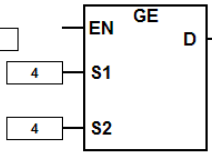



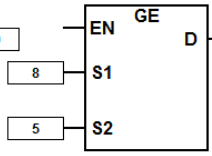




| |
|---|
| S1 is source1, S2 is source2 and D is destination, S1 and S2 can be a variable address or it can be a numaric value. D is always a bit. in compare. |
| In LT (LESS THAN) instruction if S1<S2 then D will be 1/ high. in D you can take a flag or an output address but it is better to take a flag. |
| You can see the example where 3 is being compared with 3 in LT instruction. Its destination will be zero/ low/ off. |
| You can see the example where 7 is being compared with 9 in LT instruction. its destination will be one/ high/ on. |
| compare gives result as 0 or 1/ low or high. |

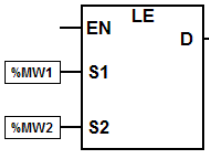


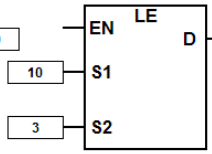



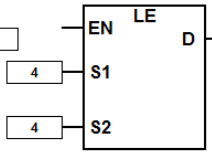



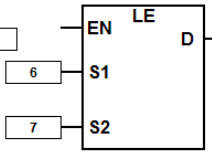




| |
|---|
| S1 is source1, S2 is source2 and D is destination, S1 and S2 can be a variable address or it can be a numaric value. D is always a bit. in compare. |
| In GE (GREATER THAN EQUAL) instruction if S1>=S2 then D will be 1/ high. in D you can take a flag or an output address but it is better to take a flag. |
| You can see the example where 8 is being compared with 10 in GE instruction. Its destination will be zero/ low/ off. 8 is neither greater nor equal to 10. |
| You can see the example where 4 is being compared with 4 in GE instruction. Its destination will be one/ high/ on. |
| You can see the example where 8 is being compared with 5 in GE instruction. its destination will be one/ high/ on. |
| compare gives result as 0 or 1/ low or high. |



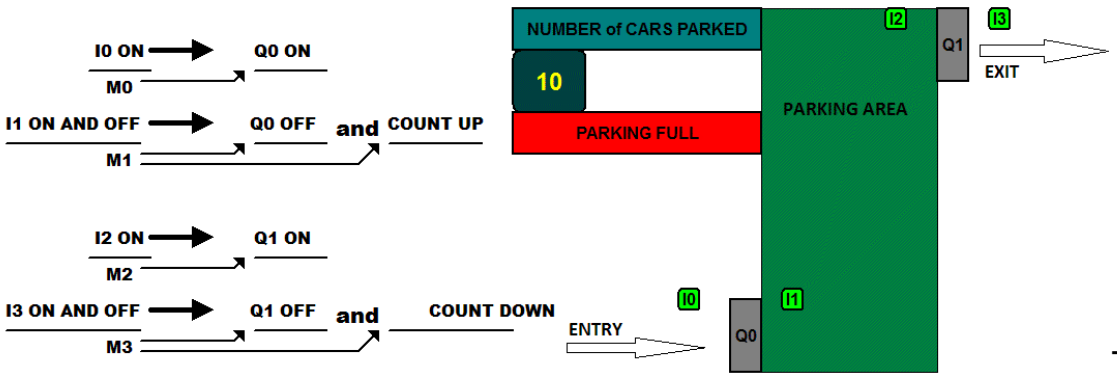



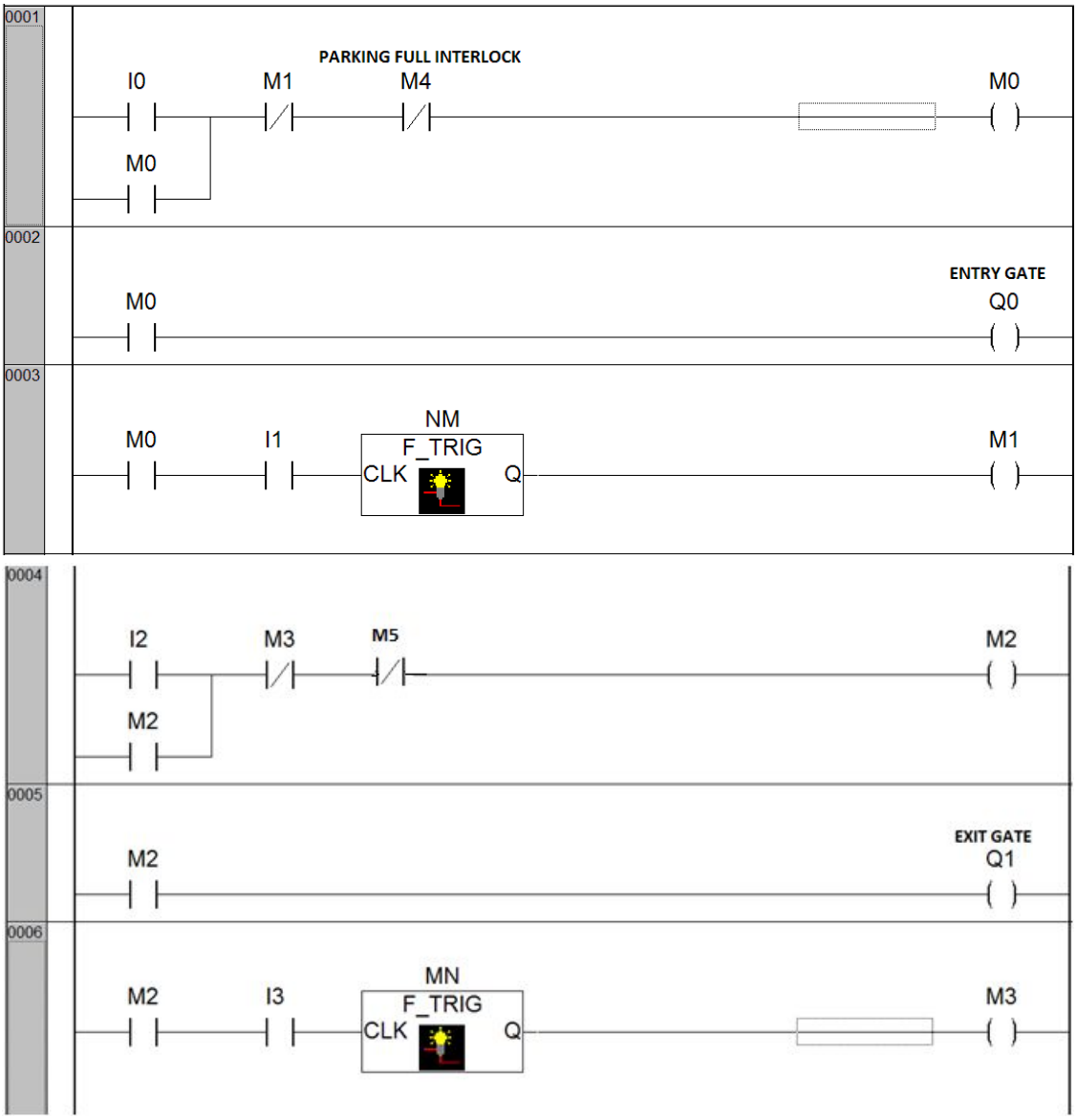



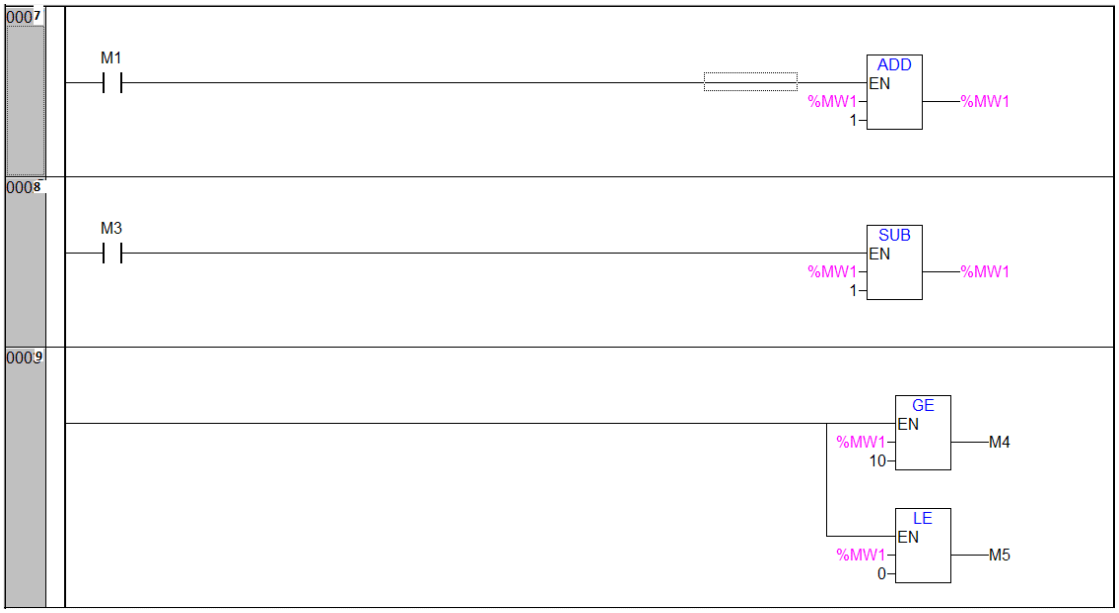

| |
|---|
| S1 is source1, S2 is source2 and D is destination, S1 and S2 can be a variable address or it can be a numaric value. D is always a bit. in compare. |
| In LE (LESS THAN EQUAL) instruction if S1<=S2 then D will be 1/ high. in D you can take a flag or an output address but it is better to take a flag. |
| You can see the example where 10 is being compared with 3 in LE instruction. Its destination will be zero/ low/ off. 10 is neither less nor equal to 3. |
| You can see the example where 4 is being compared with 4 in LE instruction. Its destination will be one/ high/ on. |
| You can see the example where 6 is being compared with 7 in LE instruction. its destination will be one/ high/ on. |
| compare gives result as 0 or 1/ low or high. |

# EX84: CAR PARKING AREA USING ARITHMETIC AND COMPARE INSTRUCTION.

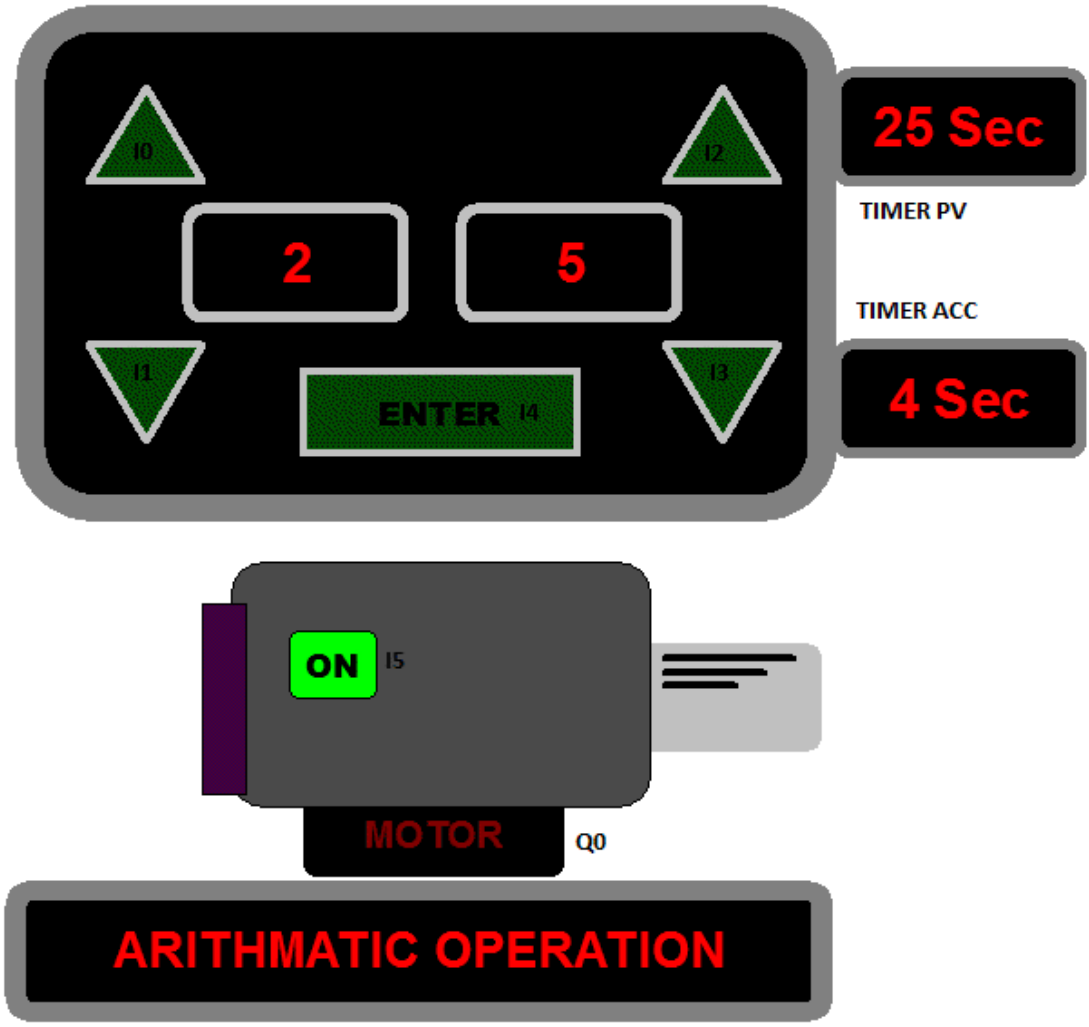



There is a car parking area where we have entry gate Q0 and exit gate Q1. On entry we have two sensors, I0 outside and I1 inside. On exit also we have two sensors I2 and I3. When car comes in front of I0 sensor, gate Q0 is open. When car reaches to I1 sensor, nothing will happen. When car crosses sensor I1 then gate Q0 will be closed and it would be count up. Inside parking area when car reaches to sensor I2, exit gate Q1 is open. When car crosses I3 sensor then Q1 is closed and it would be count down on display to show the present number of cars inside parking area. Maximum 10 cars will be parked. When parking is full and $11^{th}$ car comes in front of entry sensor I0, entry gate should not get open. Follow the steps and make program for this.


0001
PARKING FULL INTERLOCK
I0
M1
M4
M0
M0
0002
ENTRY GATE
M0
Q0
0003
M0
I1
NM
F_TRIG
CLK
Q
M1
0004
I2
M3
M5
M2
M2
0005
EXIT GATE
M2
Q1
0006
M2
I3
MN
F_TRIG
CLK
Q
M3

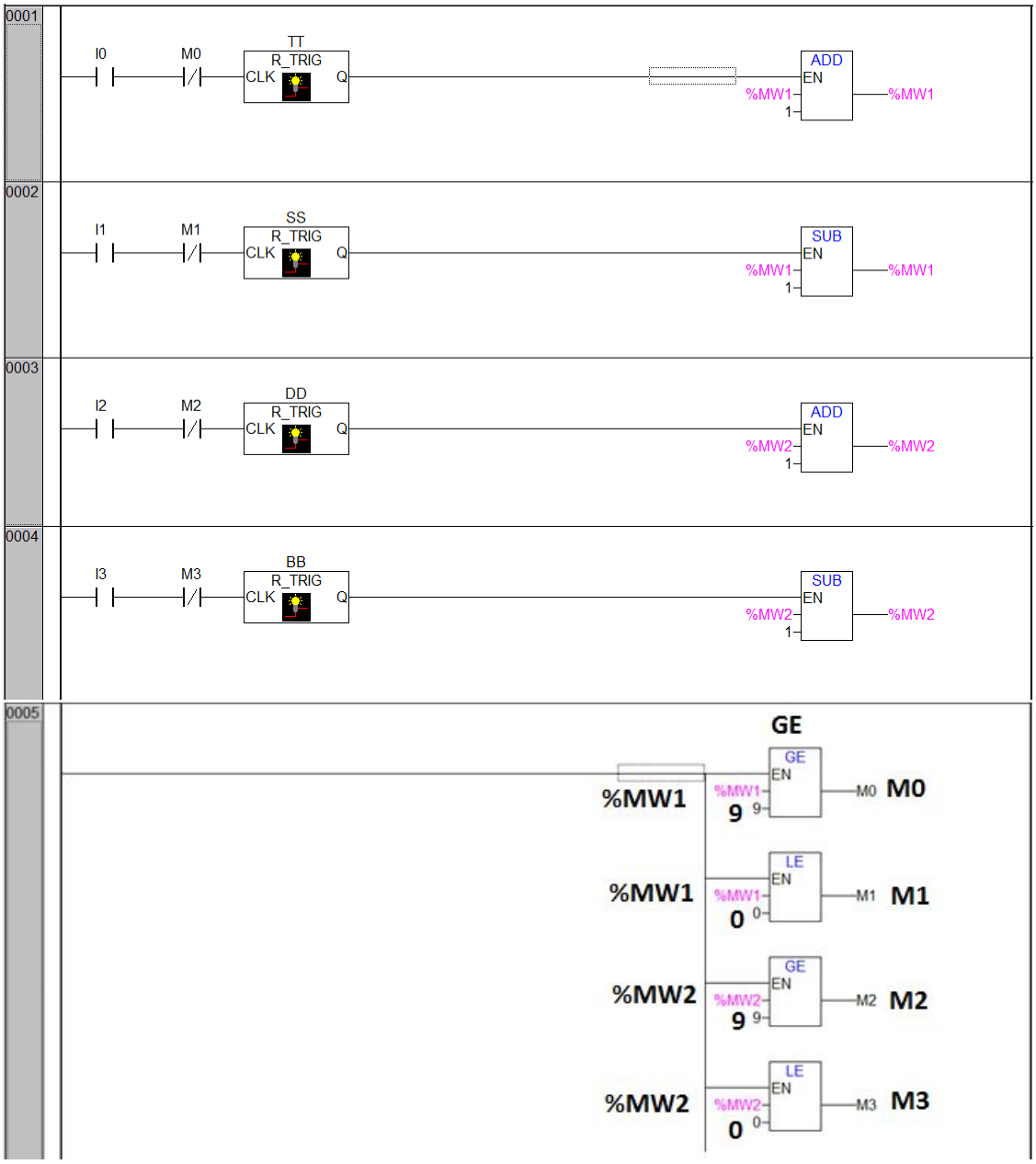


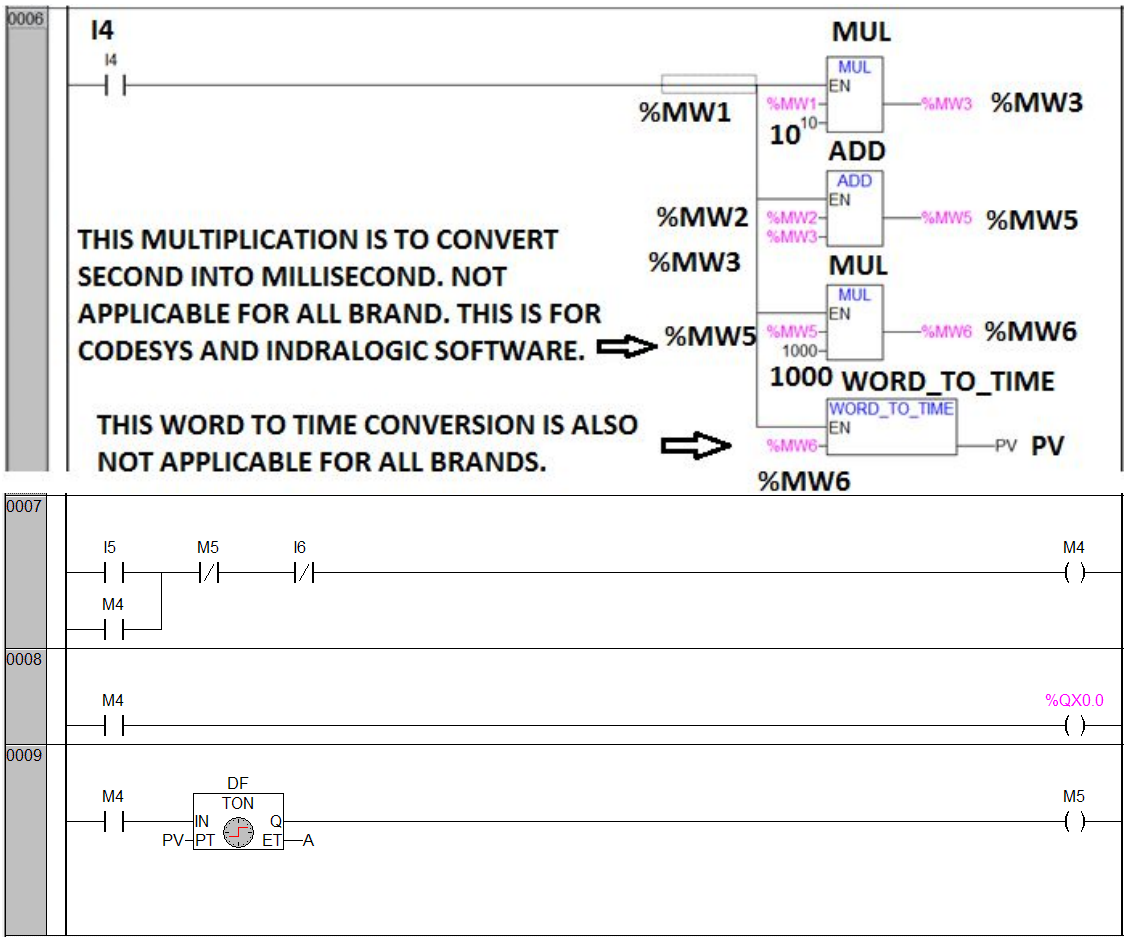

**EX85: Setting timer pre-set value using push buttons and arithmetic/ compare instructions.**




There are two variables %mw1 and %mw2. Value of %mw1 can be set by using push buttons I0 and I1 from 0 to 9.

Value of %mw1 can be set by using push buttons I2 and I3 from 0 to 9. Value should not increase more than 9 and it should not go less than 0. When I4 is pressed, value of %mw1 and the value of %mw2 should come on a new variable as a new combined value. (eg. %mw1=2 and %mw2=5. After I4 is pressed it should become 25 and should be kept in new variable). Now this new value should be sent to timer pre-set. When I5 push button is pressed then motor Q0 should get on for that duration. I6 is stop button.

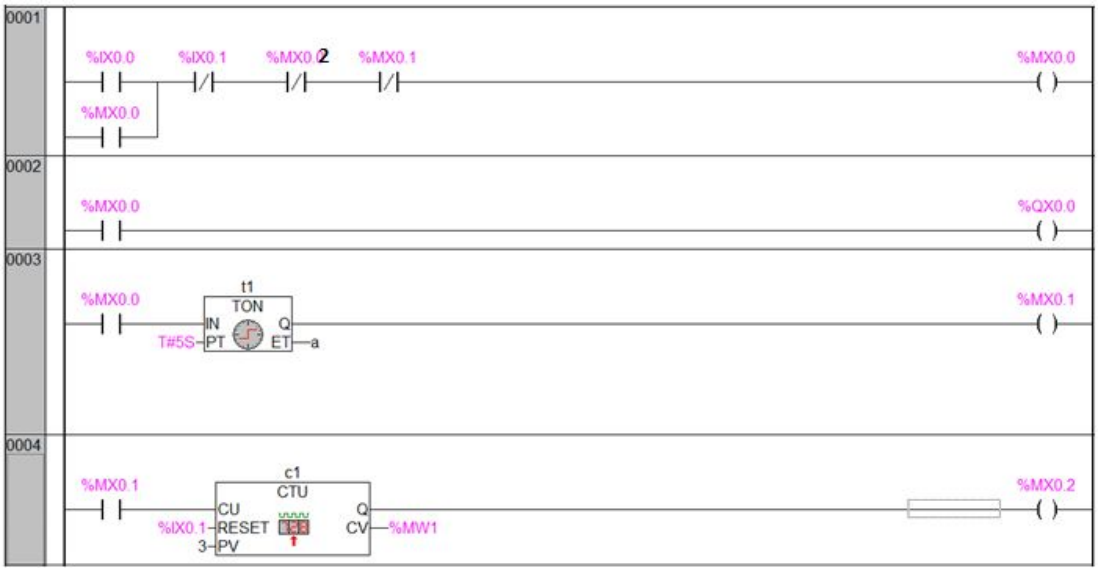

0001
I0
M0
TT
R_TRIG
CLK
Q
ADD
EN
%MW1
1
%MW1
0002
I1
M1
SS
R_TRIG
CLK
Q
SUB
EN
%MW1
1
%MW1
0003
I2
M2
DD
R_TRIG
CLK
Q
ADD
EN
%MW2
1
%MW2
0004
I3
M3
BB
R_TRIG
CLK
Q
SUB
EN
%MW2
1
%MW2
0005
GE
%MW1
GE
EN
%MW1
9
M0
M0
%MW1
LE
EN
%MW1
0
M1
M1
%MW2
GE
EN
%MW2
9
M2
M2
%MW2
LE
EN
%MW2
0
M3
M3




Now let us understand the concept of **bit and word/ variable** address. We shall take one example in CODESYS software addressing format. In CODESYS input address is %IX0.0, output %QX0.0, flag %MX0.0 and variable %MW1.

**EX86:**

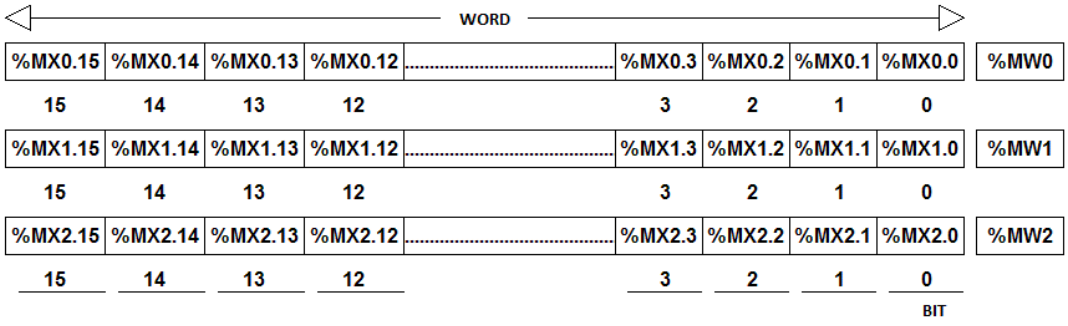
When I0 is on & off, Q0 should be on for 5 seconds. After 5 seconds break latch and this is one cycle. Press I0 again and run the same cycle. Like this run 3 cycles. After 3 cycles system will get locked. It means if you press I0 fourth time then system will not start. To unlock, press I1.





## Relation between bit and variable address:

4 bit= 1 nibble, 8 bit= 1 byte, 16 bit= 1 word, 32 bit= 1 double word. Variable address is a combination of bit addresses.

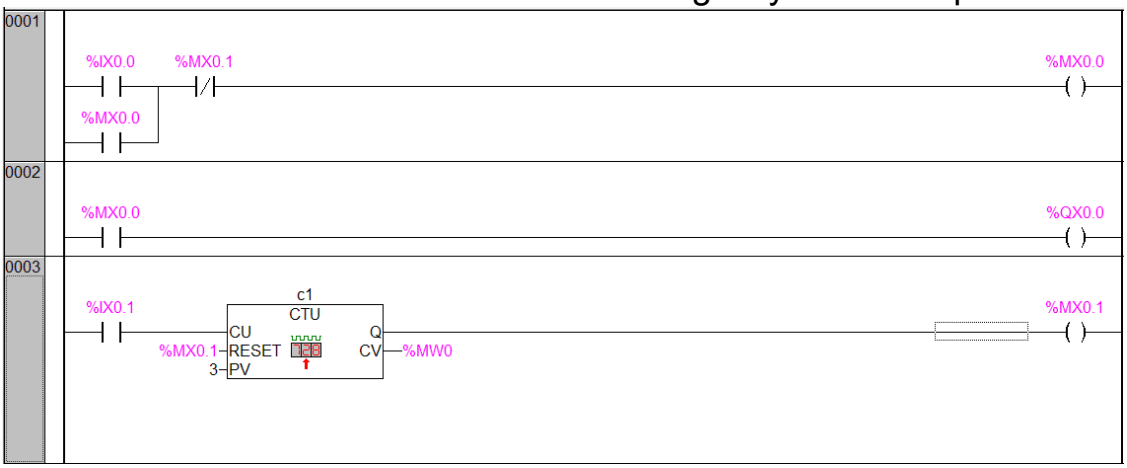



You can see that each word location has 16 bit addresses. 0 word location variable %mw0 has bit from %mx0.0 t0 %mx0.15

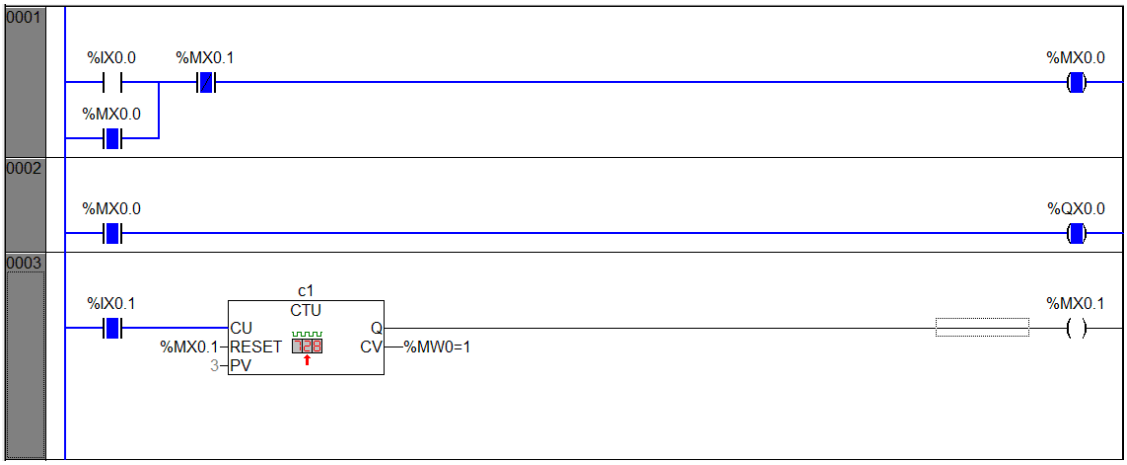
Like that in 2 word location variable %mw2 has bit from %mx2.0 t0 %mx2.15

So %mx0.2 is the third bit on word location %mw0 and %mx2.0 is the first bit on word location %mw2.

Let us understand its effect on our logic by one example.




In this program when I0 is pressed Q0 should get on. When I1 is pressed 3 times then counter done bit %mx0.1 will get on and will break latch and will reset counter. You can notice that I have taken %MW0 in CV of counter. Because of %MW0 we shall face some problem in the result of this program when it is executed. Result will not be as we expect. Let us see what will happen.

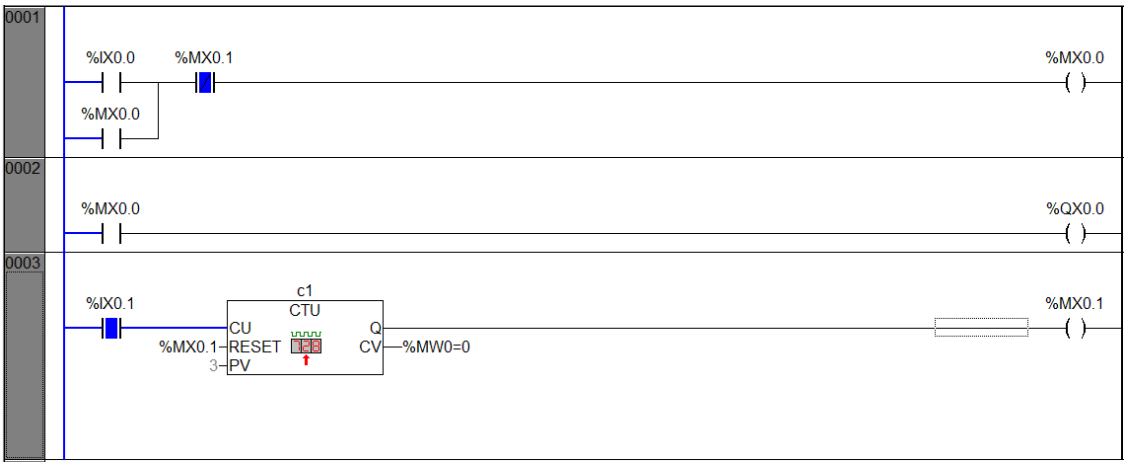



I did not press %ix0.0 but I have directly pressed %ix0.1 and you can see counter has counted 1 in CV %mw0. Without pressing start signal %ix0.0, output %qx0.0 is on. How? It is so interesting. Actually, we have taken %mw0 and we have used flag bit %mx0.0/ %mx0.1 in this program. When counter counts 1 then %mw0 gets decimal value 1. Binary of 1 is '0 1'.

| %MX0.15 | %MX0.14 | %MX0.13 | %MX0.12 | ..... | %MX0.3 | %MX0.2 | %MX0.1 | %MX0.0 | %MW0 |
|---|---|---|---|---|---|---|---|---|---|

Binary of 1 is '0 1'. It means zero location bit i. e. %mx0.0 becomes 1 and one number bit location i. e. %mx0.1 is 0. So you can see that as %MW0

gets decimal value 1, %mx0.0 gets on and it makes %qx0.0 on without pressing %ix0.0.




Now what will happen when we press %ix0.1 second time? %MW0 will get decimal value 2 and its binary is '1 0'. It means %mx0.1 will get on and %mx0.0 will get off this time. You can see %qx0.0 gets off as %mx0.0 is off and counter gets reset on second press only as %mx0.1 is on. Counter had to get reset on five counts. This is the effect of word and bit address on your logic. Finally, we understand that if you use a bit of some word location in your program then don't use that word. If you use a word in your program then don't use its bit addresses. As we have used bit %mx0.0 in our program so we had not to use %MW0 but we could have used %MW1 or other location word.

## EX: 87 Program without output coil.

| I0 PRESS | Q0 | Q1 |
|---|---|---|
| FIRST | 1 | 0 |
| SECOND | 0 | 1 |
| THIRD | 1 | 1 |
| FOURTH | 0 | 0 |

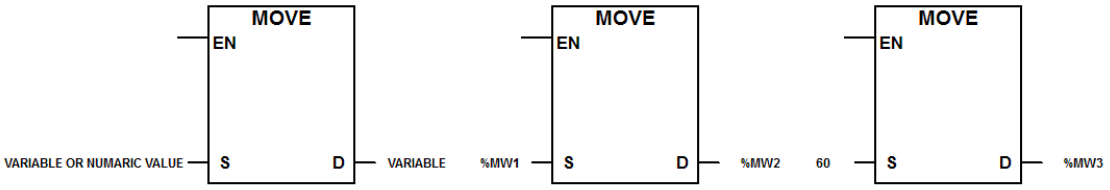
This example we have solved in example 44 using bit addresses. This time we shall make it using word address.

This is the solution. It is so interesting that without any output coil in program, physical output connected to %qx0.0 and %qx0.1 output terminals of PLC will be operated as per truth table conditions. How will it work? You can see that I have used %QW0 in CV of counter. %qw0 is the combination of 16 outputs from %qx0.0 to %qx0.15. Through counter we send decimal value to %QW0 in each press of %ix0.0 and as per binary the output bit is operated.

| %QX0.15 | %QX0.14 | %QX0.13 | %QX0.12 | ..... | %QX0.3 | %QX0.2 | %QX0.1 | %QX0.0 | %QW0 |
|---|---|---|---|---|---|---|---|---|---|
| In first press of %IX0.0 | %QW0=1 and binary of 1 is 0 1 | | | | | | 0 | 1 | |
| %QX0.15 | %QX0.14 | %QX0.13 | %QX0.12 | ..... | %QX0.3 | %QX0.2 | %QX0.1 | %QX0.0 | %QW0 |
| In 2nd press of %IX0.0 | %QW0=2 and binary of 2 is 1 0 | | | | | | 1 | 0 | |
| %QX0.15 | %QX0.14 | %QX0.13 | %QX0.12 | ..... | %QX0.3 | %QX0.2 | %QX0.1 | %QX0.0 | %QW0 |
| In third press of %IX0.0 | %QW0=3 and binary of 3 is 1 1 | | | | | | 1 | 1 | |
| %QX0.15 | %QX0.14 | %QX0.13 | %QX0.12 | ..... | %QX0.3 | %QX0.2 | %QX0.1 | %QX0.0 | %QW0 |
| In fourth press of %IX0.0 | %QW0=4 and binary of 4 is 1 0 0 | | | | | 1 | 0 | 0 | |

In fourth press done bit of counter %mx0.0 gets on and it reset counter making %QW0=0 resulting all outputs off.

In such types of program, counter cannot be used everywhere. You can use move instruction to move decimal value to variable addresses to operate its bit addresses.




MOVE instruction is used to move source value to some destination location. Source can be a variable address or numaric value but destination is always a variable address. When enable is high source copy is moves to destination.

**EX88:**

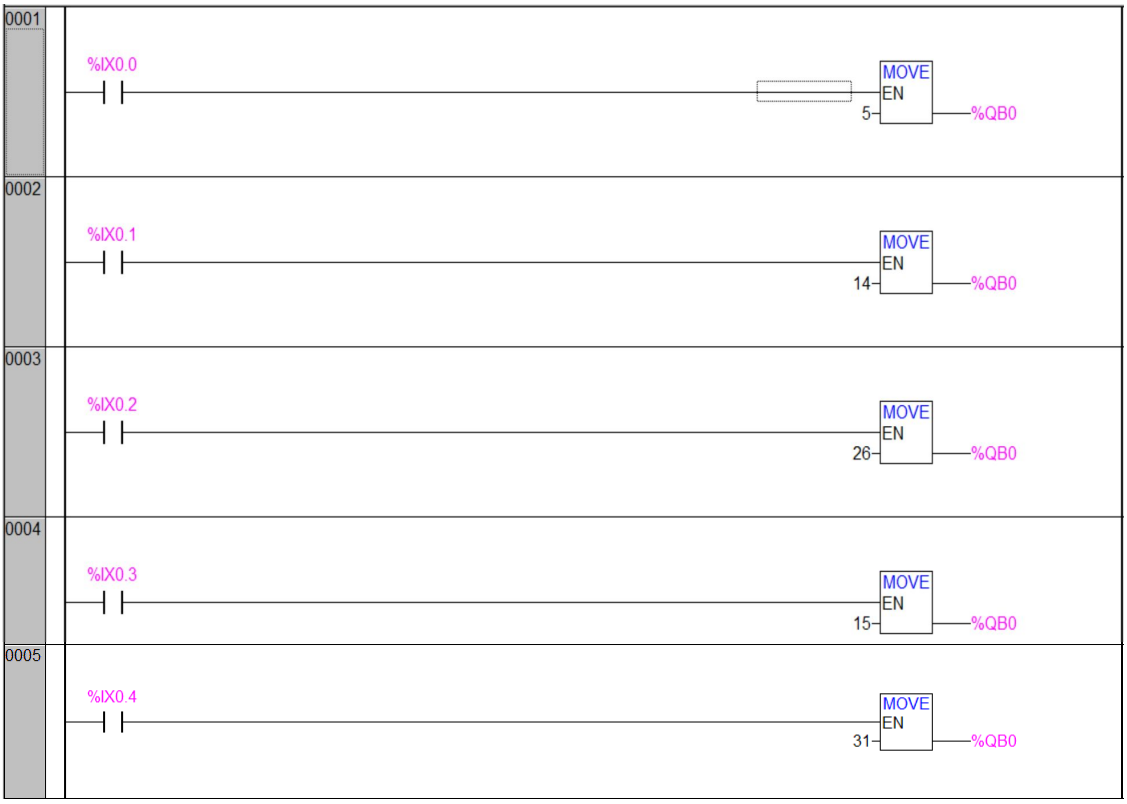
| | 16 | 8 | 4 | 2 | 1 | |
|---|---|---|---|---|---|---|
| **I0 ON** | **Q4** | **Q3** | **Q2** | **Q1** | **Q0** | **DECIMAL** |
| **I1 ON** | **0** | **0** | **1** | **0** | **1** | **5** |
| **I2 ON** | **0** | **1** | **1** | **1** | **0** | **14** |
| **I3 ON** | **1** | **1** | **0** | **1** | **0** | **26** |
| **I4 ON** | **0** | **1** | **1** | **1** | **1** | **15** |
| **I5 ON** | **1** | **1** | **1** | **1** | **1** | **31** |

Outputs are in binary form. On press of each push button, we have output binary and we have calculated its decimal value right most side. When I0 is

pressed, outputs are 00101 and its decimal value is 5. If 5 is moved to output variable then respective outputs will get on.

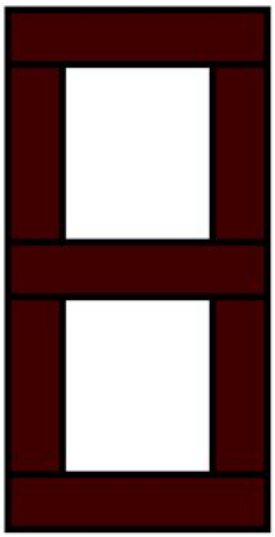
| Flag variable | | Output variable | |
|---|---|---|---|
| **MB0** | **8 FLAGS ARE COMBINED** | **QB0** | **8 OUTPUTS ARE COMBINED** |
| **MW0** | **16 FLAGS ARE COMBINED** | **QW0** | **16 OUTPUTS ARE COMBINED** |
| **MD0** | **32 FLAGS ARE COMBINED** | **QD0** | **32 OUTPUTS ARE COMBINED** |
| | | | |

In our example we have only five outputs so no need to use %QW0 but we can use %QB0.

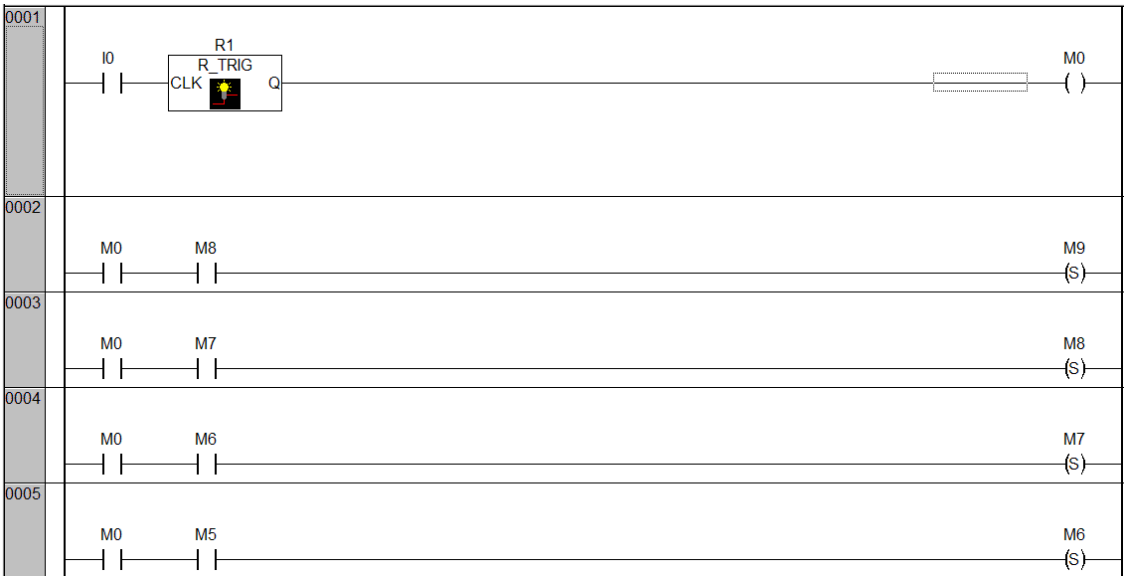


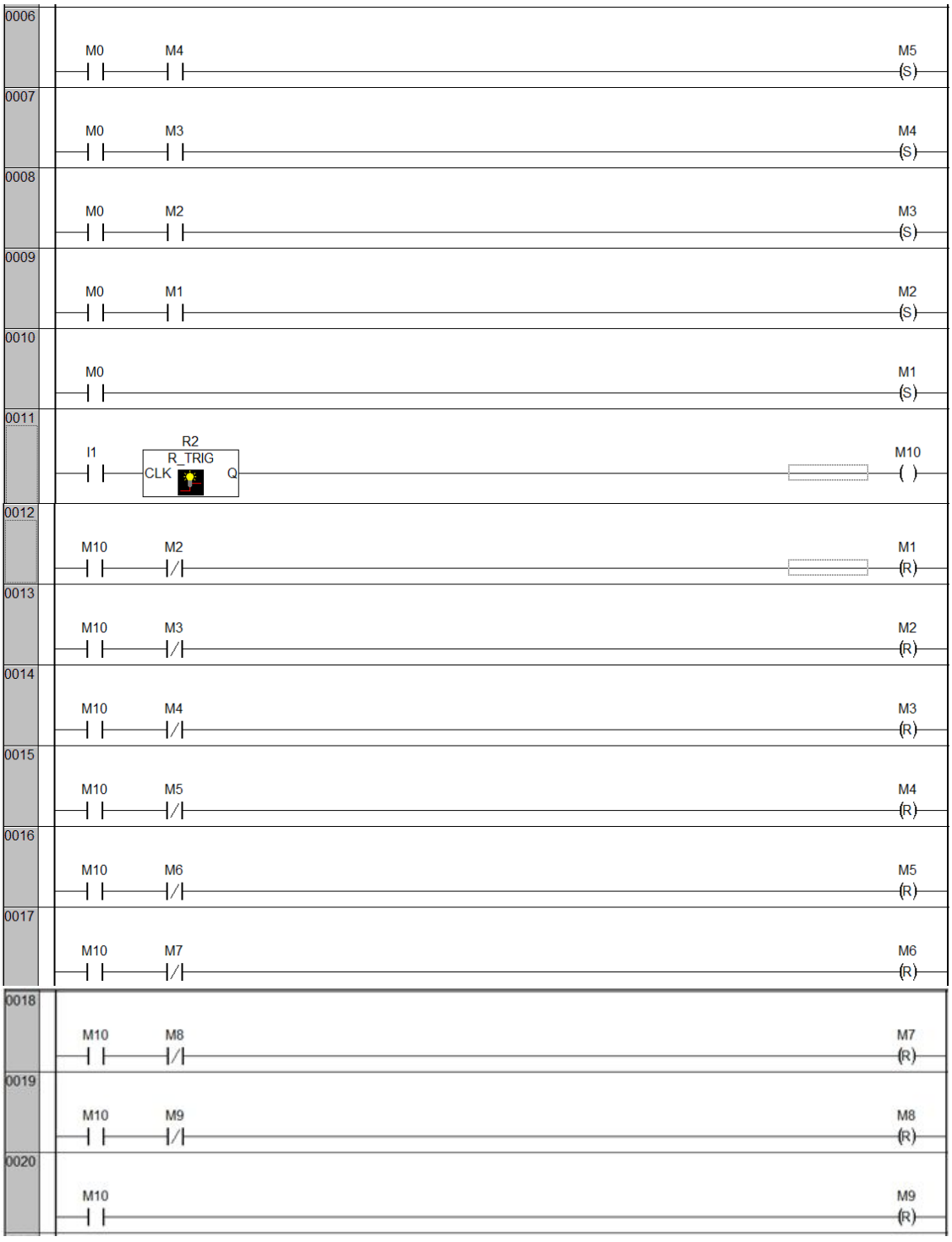

**EX89: Seven segment program using output variable.**

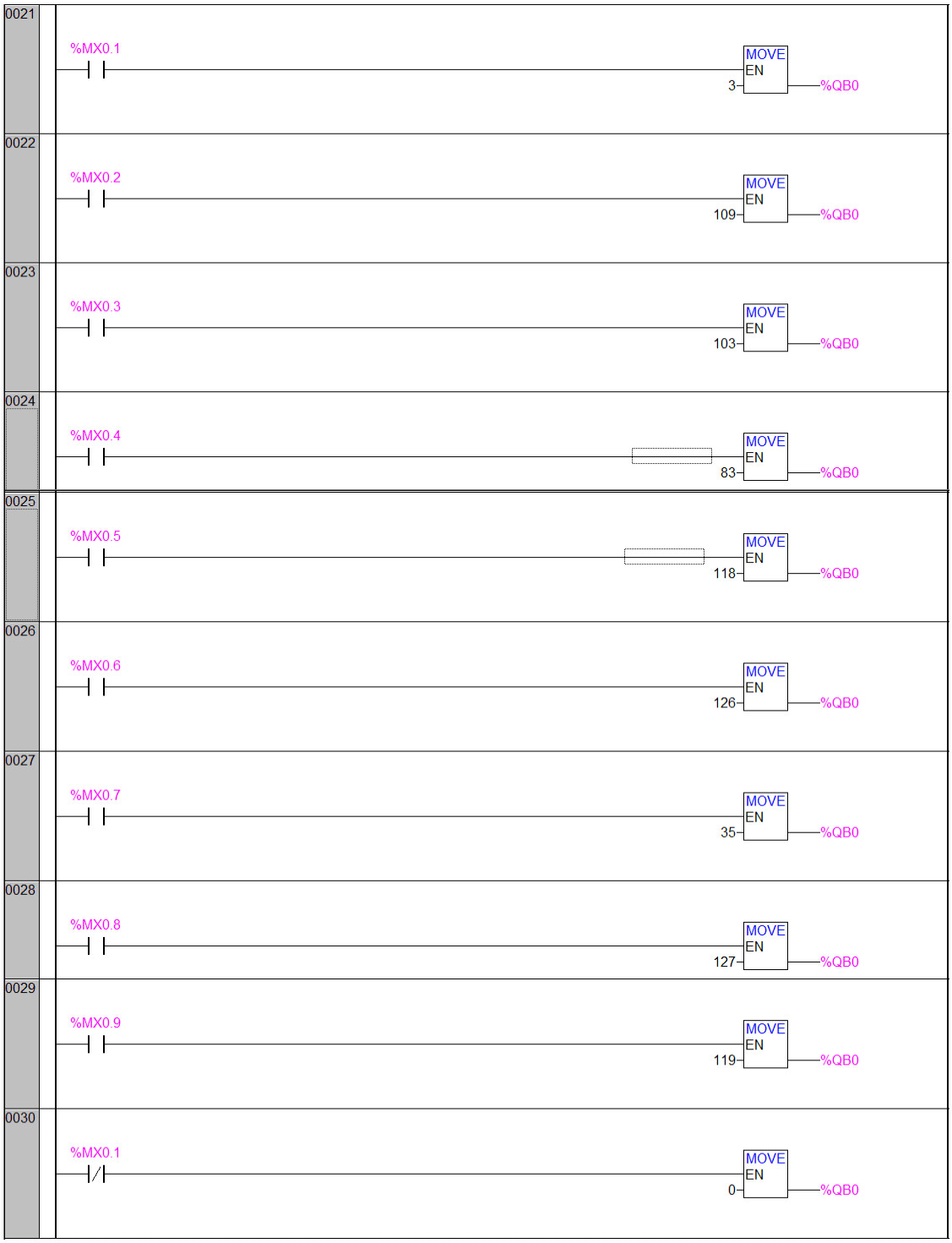
| | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |
|---|---|---|---|---|---|---|---|---|
| SIGNAL ↓ | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 | |
| M1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 3 |
| M2 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 109 |
| M3 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 103 |
| M4 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 83 |
| M5 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 118 |
| M6 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 126 |
| M7 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 35 |
| M8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 127 |
| M9 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 119 |



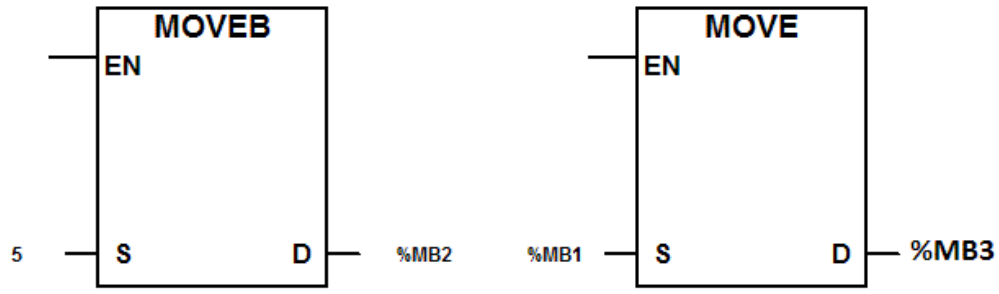
To display number from 1 to 9, I0 has to be pressed 9 times. To display number 9 to 1, I1 has to be pressed nine times. For nine press of I0 let us generate nine flags and from each flag respective outputs will be operated as shown in truth table. As system is powered on, zero should be displays i. e. Q0, Q1, Q2, Q3, Q4 and Q5 on.



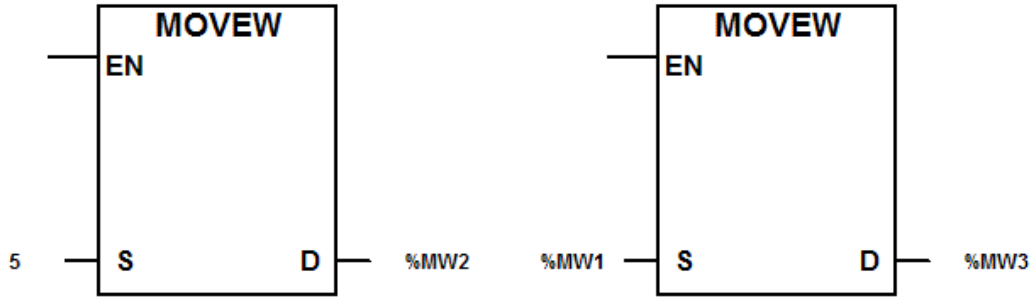

0006
M0
M4
M5
(S)
0007
M0
M3
M4
(S)
0008
M0
M2
M3
(S)
0009
M0
M1
M2
(S)
0010
M0
M1
(S)
0011
R2
I1
R_TRIG
CLK
Q
M10
( )
0012
M10
M2
M1
(R)
0013
M10
M3
M2
(R)
0014
M10
M4
M3
(R)
0015
M10
M5
M4
(R)
0016
M10
M6
M5
(R)
0017
M10
M7
M6
(R)
0018
M10
M8
M7
(R)
0019
M10
M9
M8
(R)
0020
M10
M9
(R)



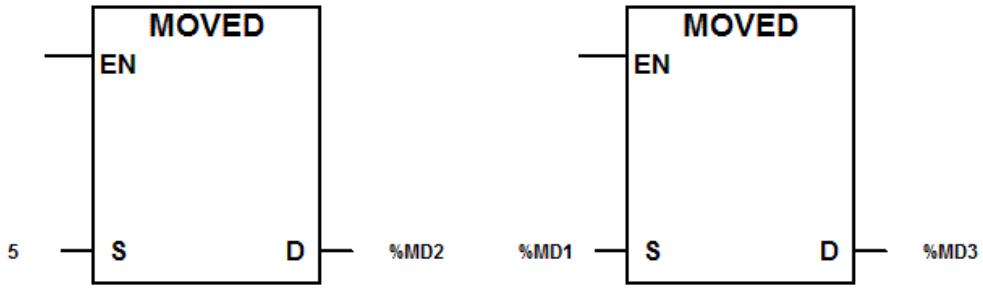

In many software you will find MOVEB, MOVEW, MOVED, BMOVE and FMOVE.



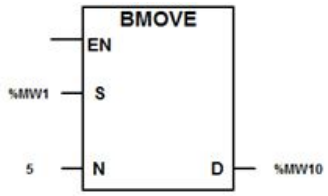

**In MOVEB (Move byte) byte source will be moved to byte destination**




**In MOVEW (Move WORD) word source will be moved to word destination**



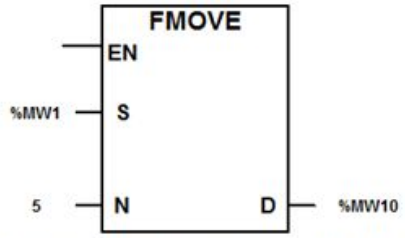

**In MOVED (Move DWORD) double word source will be moved to double word destination**




**In BMOVE (Block Move) a set of consecutive variable source content is moved to a set of consecutive variable destination.**

**S is source, N is number of consecutive variable and D is destination.**

**In this block it is shown that content of %MW1 to %MW5 will be moved to %MW10 to %MW15 (5 consecutive source and destination.)**




In FMOVE (FILE Move) content of one variable source is moved to a set of consecutive variable destination.

S is source, N is number of consecutive variable and D is destination.

In this block it is shown that content of %MW1 will be moved to %MW10 to %MW14(5 consecutive destination.)

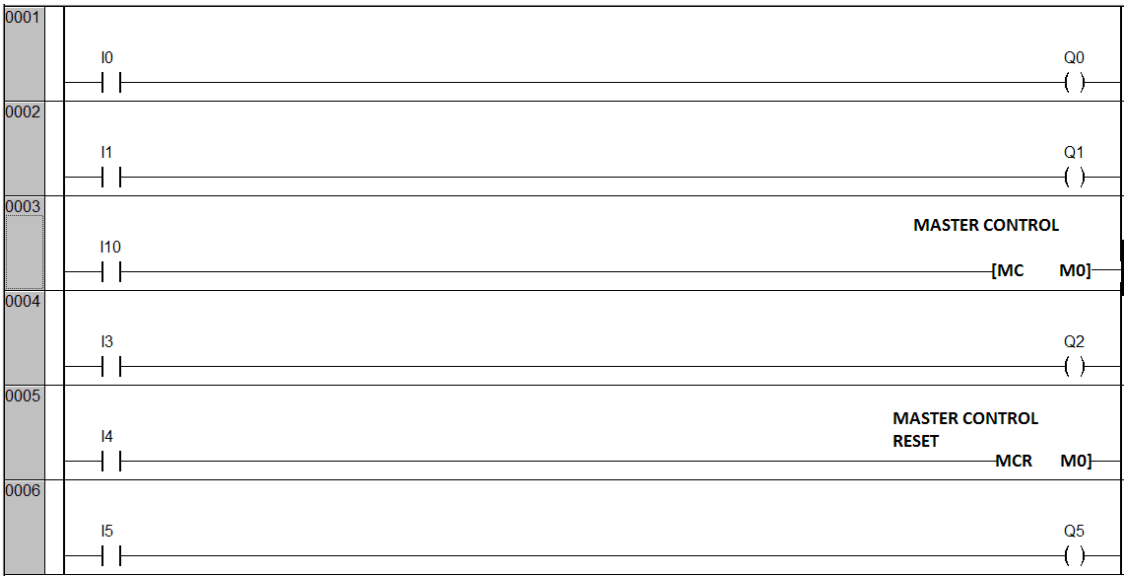
**In FMOVE source can be a variable or it can be numeric value.**

**Jump and label:** Jump instruction is used to skip selected ladders' scan. Jump is put on ladder from where scan of ladder should be skipped. Label is put on the ladder up to which scan should be skipped. Ladders between jump and label are not scanned and it results no change in coil status.


Jump has been put in third ladder and it will be active when I10 is on. Label is in sixth ladder. When jump is not active then all ladders are being scanned and coils are on.


When jump is active, you can see that coil status of ladders between Jump and Label has not got changed. Other coils changed their status. It happened because scan of ladders between jump and label got skipped.

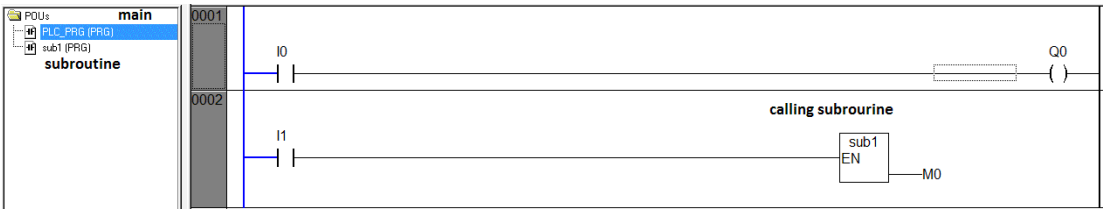
**MASTER CONTROL and RESET:** When master control is active then only scan of ladders between master control and master control reset will take place. It is opposite of Jump/ Label.



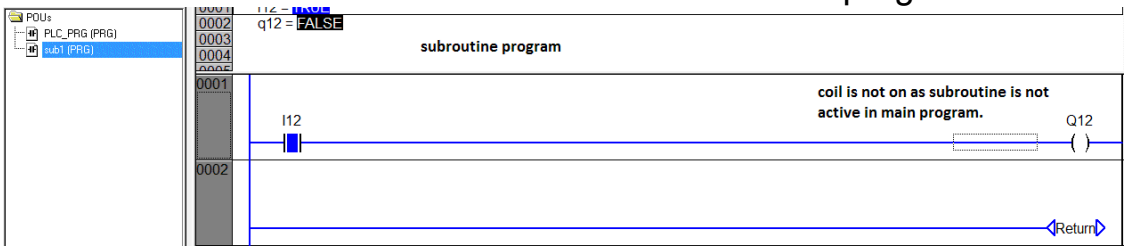

Ladder number four will be scanned when master control is active or else it will be skipped.

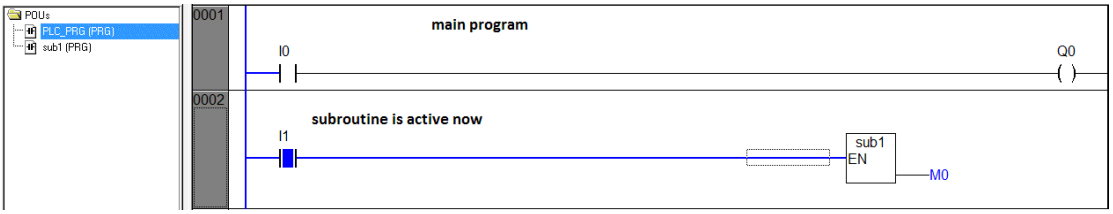
**Subroutine:** Subroutine is used to reduce scan time of program by splitting it in subroutine when it is a very big logic. If there are some set of logic which is being repeated many times in different conditions then it is better to make

a subroutine file and it can be called from main program whenever required. Subroutine program is scanned when it is called from main program.

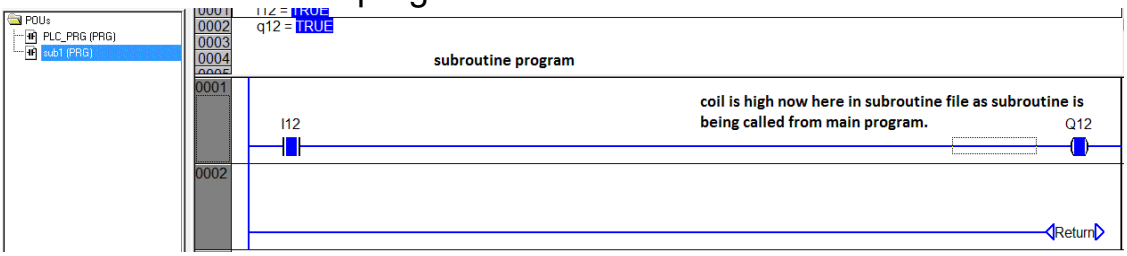



In second ladder subroutine 1 is not active so subroutine program will not work.




In main program now subroutine is active.




Now let us learn to make logic for different brand of PLC software. We shall start with **MITSUBISHI.**

**MITSUBISHI MELSEC**

**HARDWARE**

- A CPU
- QnA CPU
- Q CPU [A MODE]
- Q CPU [Q MODE]
- FX CPU

MODULAR PLC

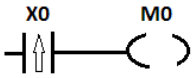
MICRO PLC

**SOFTWARE**

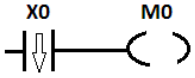
- GX DEVELOPER
- GX WORKS

| INPUT | OUTPUT | FLAG |
|---|---|---|
| X00 | Y00 | M0 |
| X01 | Y01 | . |
| . | . | . |
| . | . | . |
| X7 | Y7 | . |
| X10 | Y10 | |
| . | . | |
| . | . | |
| X17 | Y17 | |

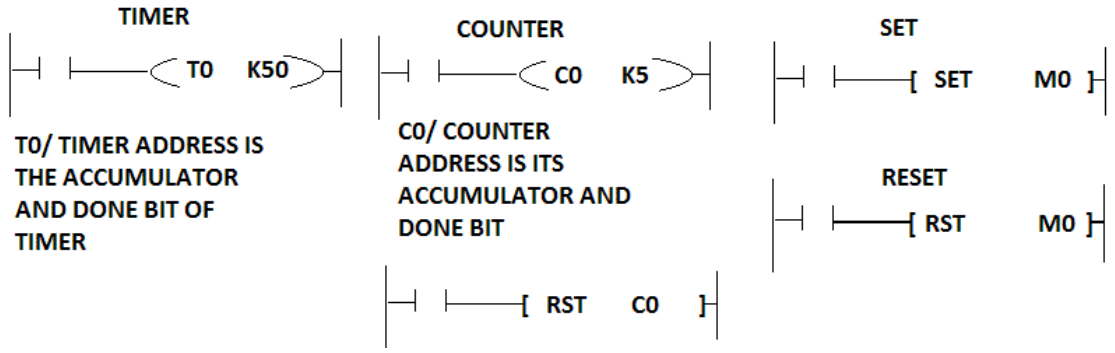
**POSITIVE PULSE**




**NEGATIVE PULSE**



Mitsubishi is Japanese brand. It has many products. Among all product PLC is called MELSEC. It is one of the leading brands in industrial automation. It has micro and modular both types of PLC as mentioned in above picture. In each CPU there are many models. Software is GX Developer/ GX works for programming of any CPU. Input address is X and address for output is Y both in octal number for micro/ FX CPU & Hexadecimal (0 to F) for modular CPU. Flag is M and in decimal number system. Readymade pulse contacts are available.




Timer is taken in coil. T is the timer so first timer is T0. Next timer will be T1, T2, T3 up to T255. K50 is pre-set value for 5 seconds. Timer is 100 MS time base so pre-set value is multiple of 10.

$$PV = \frac{\text{DESIRED TIME}}{\text{TIME BASE}} = \frac{5S}{100MS} = \frac{5S}{100MS}$$

$$= \frac{5 * 1000}{100} = 50$$

In Mitsubishi timer address is its accumulator and done bit as well. No need to take flag as done bit. Timer address can be directly used in NO and NC contacts.

Counter is also taken in coil. C is the counter and C0 is the first counter. 256 counters are available. Counter address is its accumulator and done bit as well. To reset a counter, RST instruction is used as shown in in picture. SET and RESET instruction is used in big bracket.

## ARITHMETIC INSTRUCTION

| | | |
|---|---|---|
| --[ ADD K5 K10 D0 ] | | --[ + K5 K10 D0] |
| --[ SUB K7 K5 D0 ] | | --[ - K7 K5 D0] |
| | OR | |
| --[ MUL K8 K2 D0] | | --[ * K8 K2 D0] |
| --[ DIV K6 K3 D0] | | --[ / K6 K3 D0] |

D is the variable address in Mitsubishi.

## COMPARE INSTRUCTION

--[ > K5 K3 ]-----( M0 )
--[ < K5 K8 ]-----( M0 )
--[ = K5 K5 ]-----( M0 )
--[ >= K5 K4]-----( M0 )
--[ <= K5 K10]---( M0 )

IN SOME MICRO CPU CMP INSTRUCTION IS USED TO COMPARE

--[ CMP K5 K10 M0 ]
Automatically three consecutive flags are used i. e. M0, M1 and M2 as we have taken M0 as destination here.

S1> S2 => M0 on, S1= S2 => M1 on, S1< S2=> M2 on.

## MOVE

--[ MOV S D ]
--[ MOVB S D ]
--[ MOVW S D ]
--[ MOVD S D ]
--[ BMOV D0 D10 K5]
--[ FMOV D0 D10 K5]

S is source

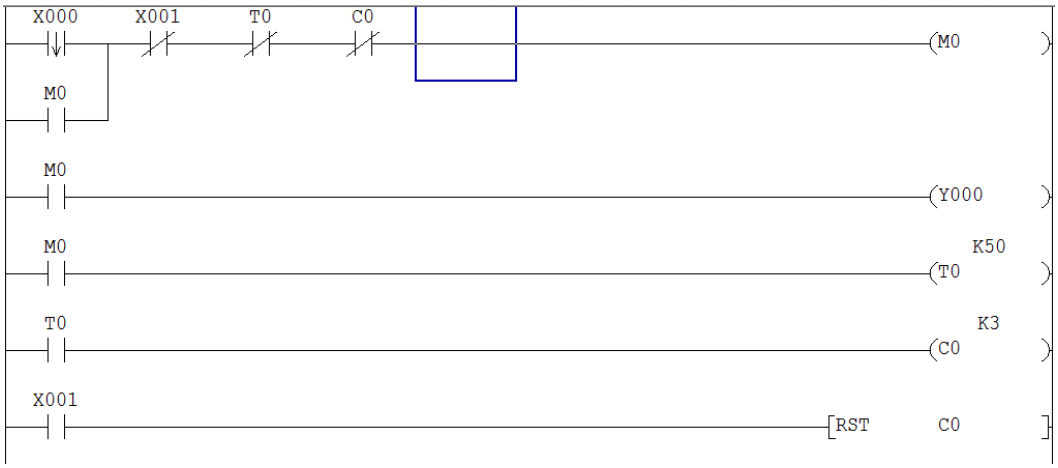
D is destination

D0 and D10 are variable addresses

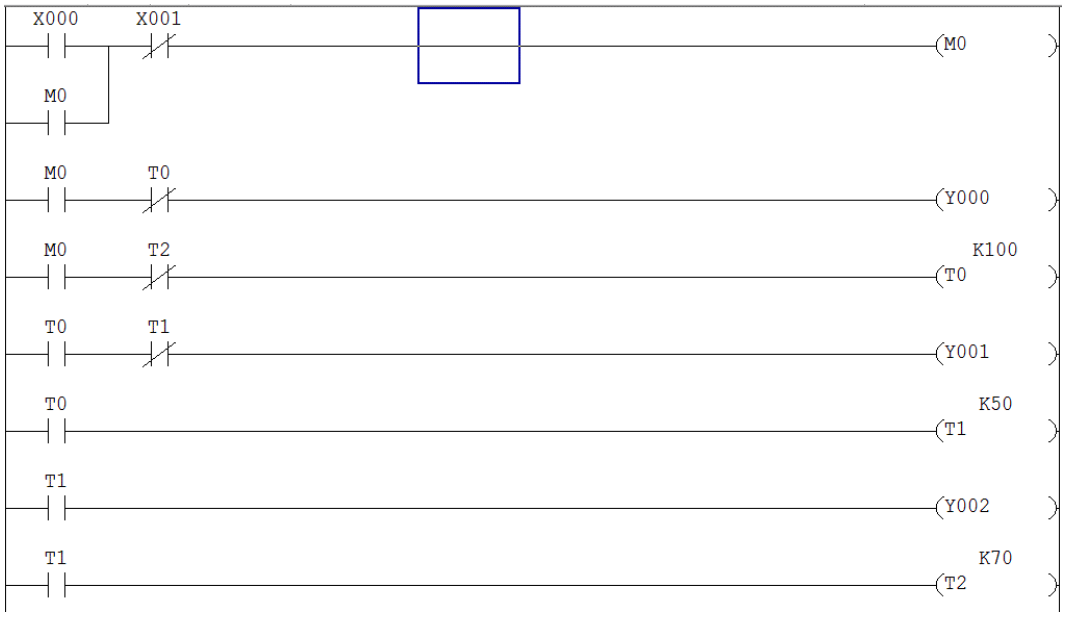
K denotes decimal

Variable address: D0/ K2M0/ K4M0, K2Y0/ K4Y0.

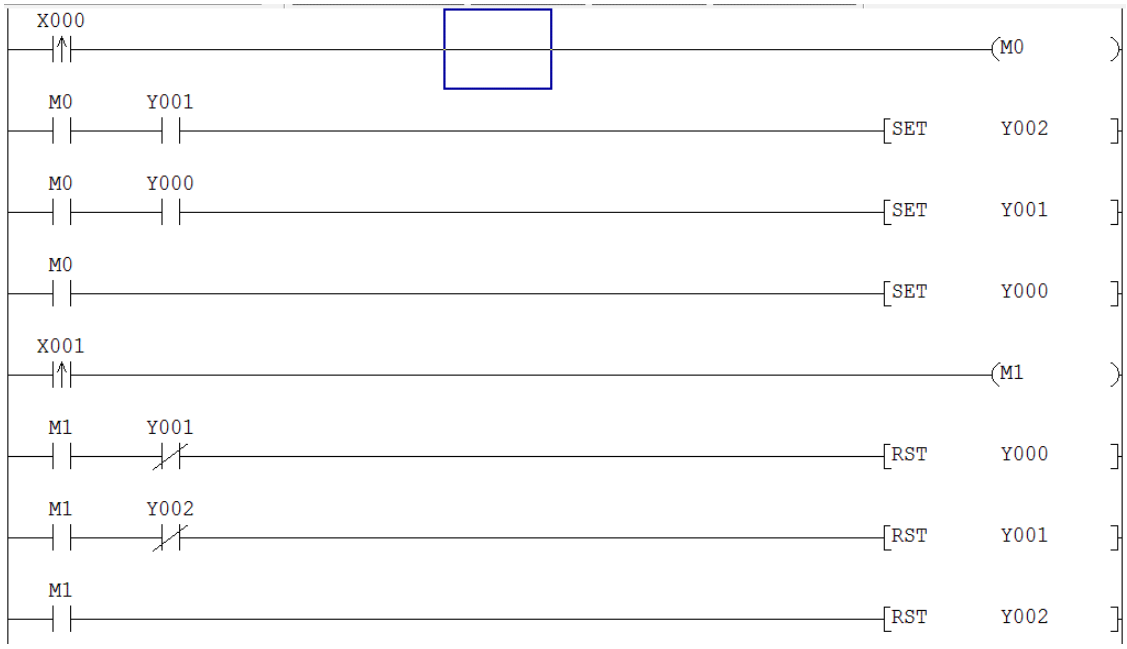
Let us practice few examples for MELSEC. Make EX86 for GX Developer.



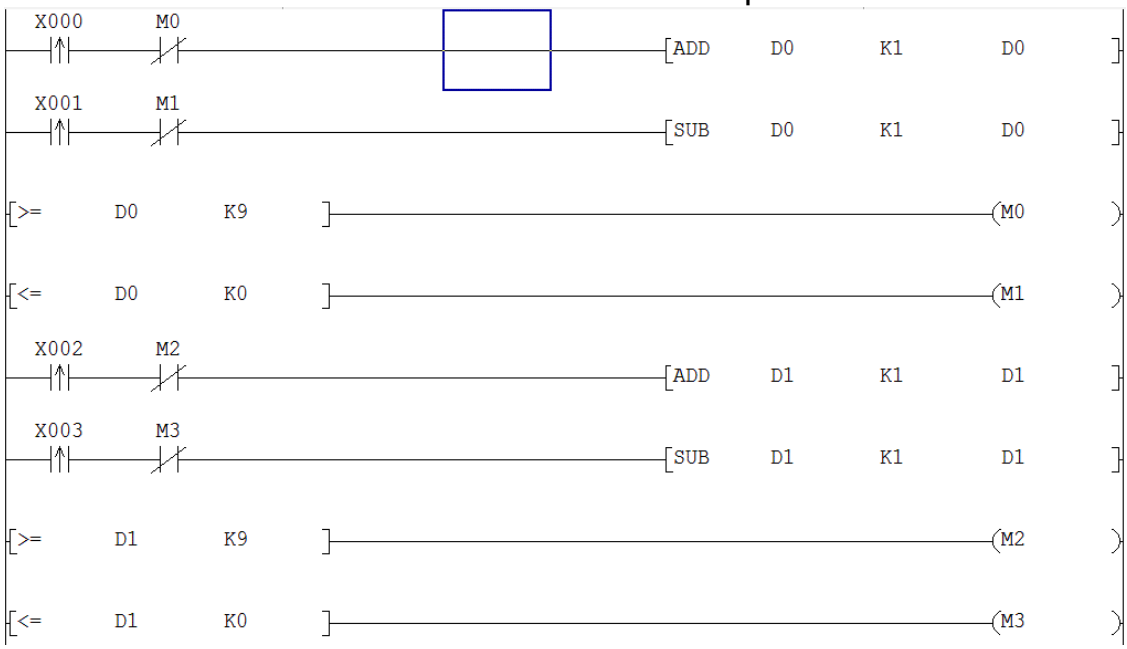

Make EX14 for GX Developer.

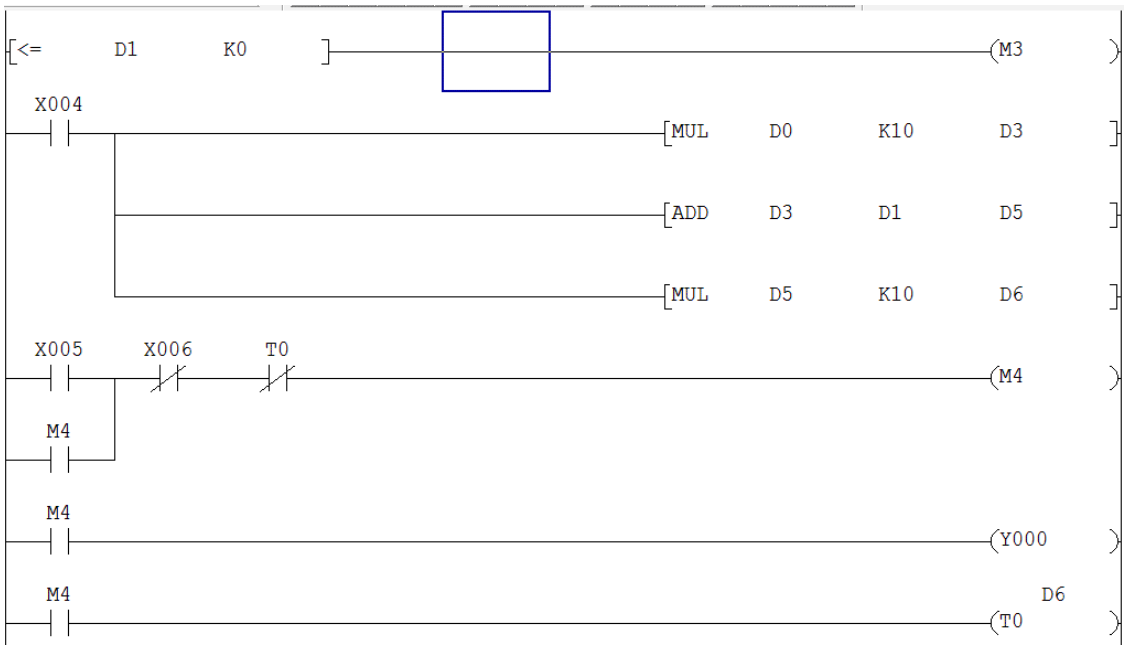


Make EX47 for GX Developer.




Make EX85 for GX Developer.






Here D5 gets the calculated value but it is multiplied by 10 because timer pre-set is multiple of 10.

Make EX88 for GX Developer.

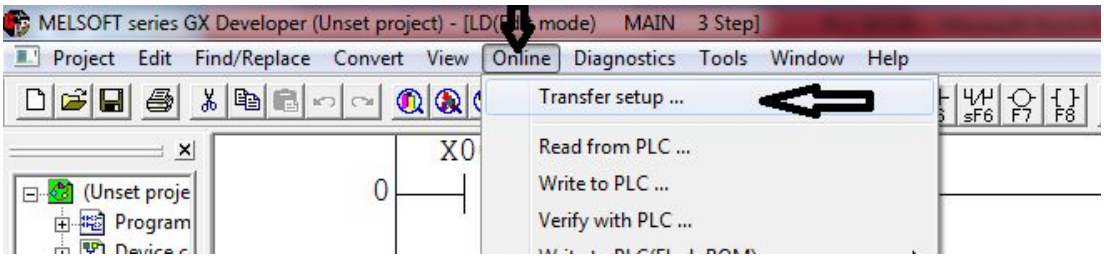
In Mitsubishi

%MB0 □ K2M0, %MW0 □ K4M0, %QB0 □ K2Y0 and %QW0 □ K4Y0.

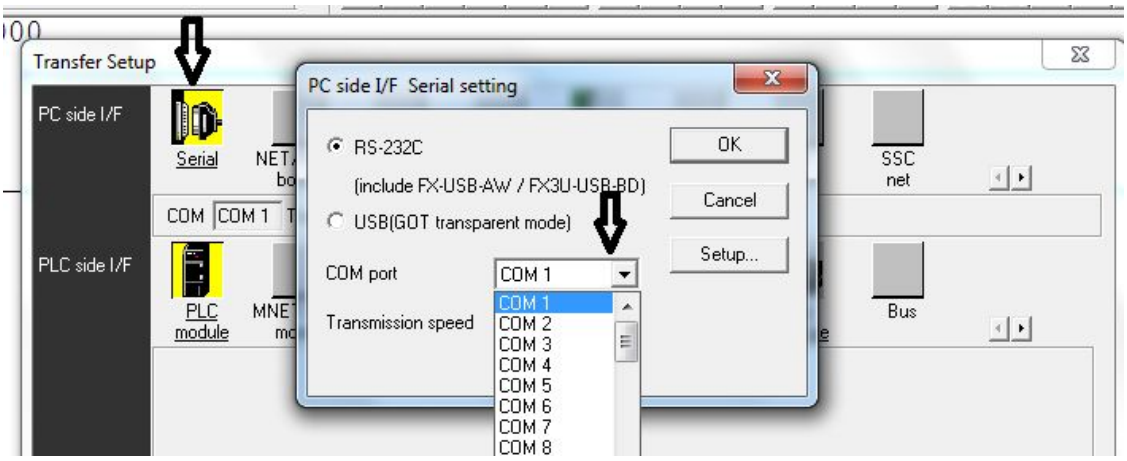

## DELTA PLC PROGRAMMING IS VERY SIMILAR TO MITSUBISHI PROGRAMMING

After making program in PLC software, you need to send it to PLC hardware. You should have PLC communication cable of that brand. You should have USB-to-Serial convertor cable as laptop has no 9 pins serial port and to

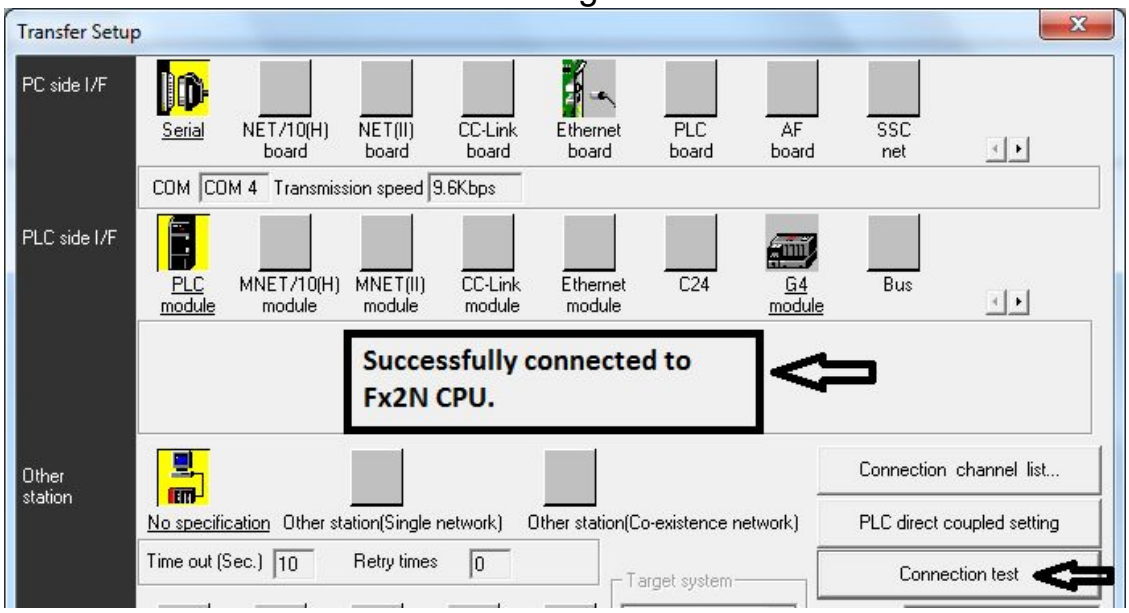
maximum PLC communication cable have 9 pin serial ports. You must install USB-to-serial convertor driver in your laptop then only cable connection will be detected in laptop. Connect your PLC communication cable to PLC and to laptop.  As you connect it to laptop, check communication port number in device manager.



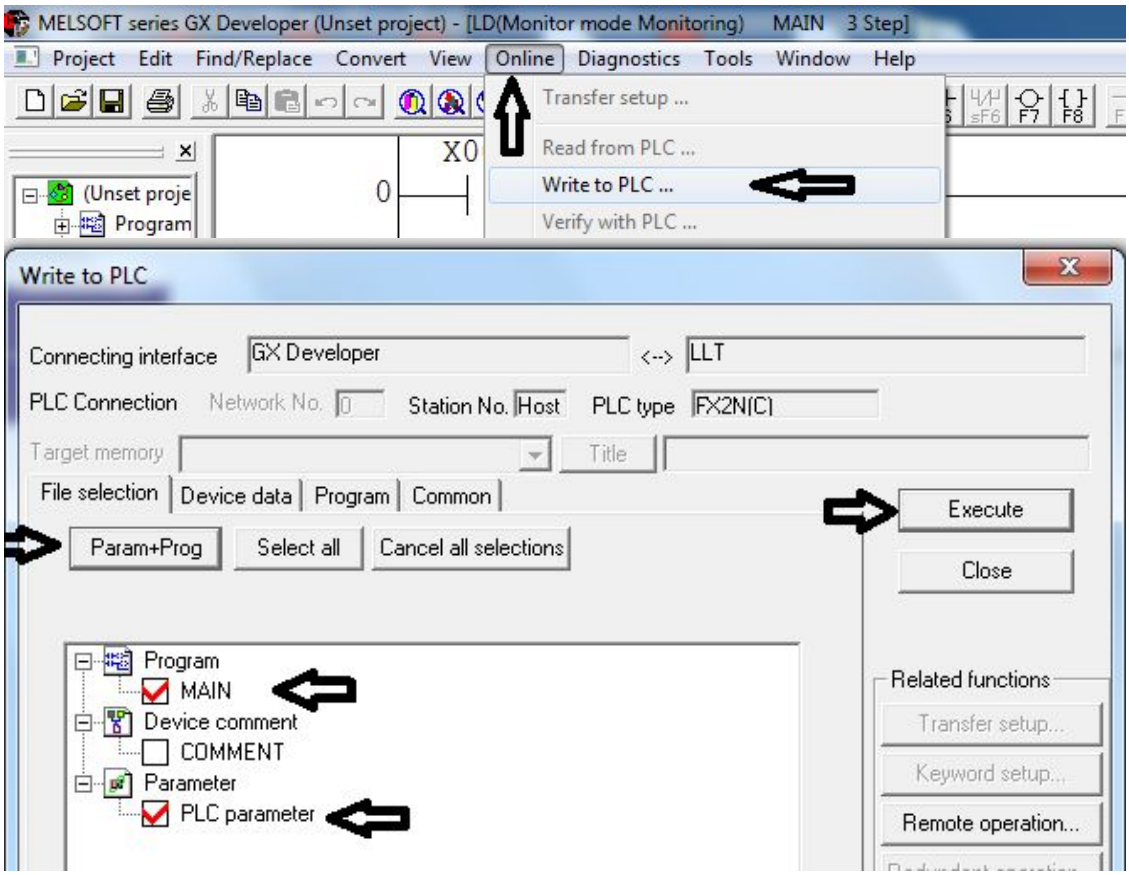

In GX developer we go to Online □ transfer setup.




We double click Serial and set the communication port number as per device manager.



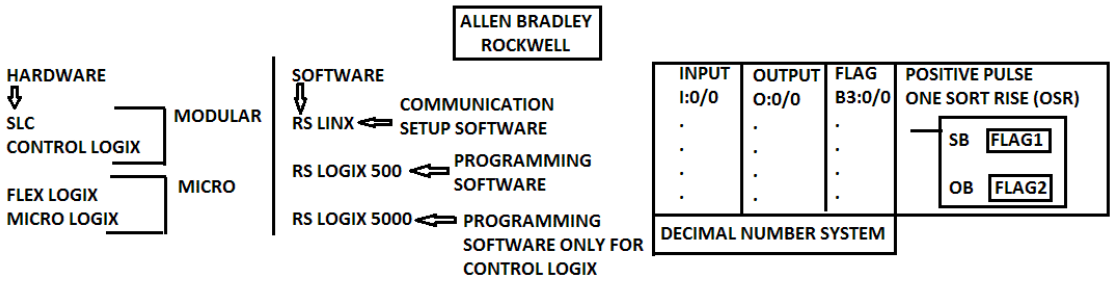
Then we click connection test. If PLC gets connected with laptop then it will show you successfully connected message. Communication setup steps we can have different in different brand. Now you can send program to PLC. To send program we can have different option in different brand like Download/ Send to PLC/ PC  PLC/ Write to PLC etc.



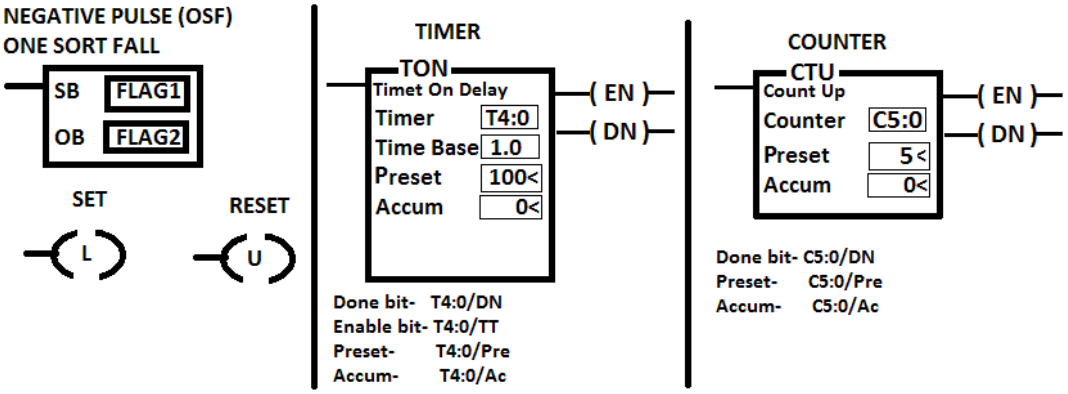

Online  Write to PLC  Select Main and Parameter  Execute.

To receive program from PLC we can have different option in different brand like Upload/ receive from PLC/ PLC  PC/ Read from PLC etc.

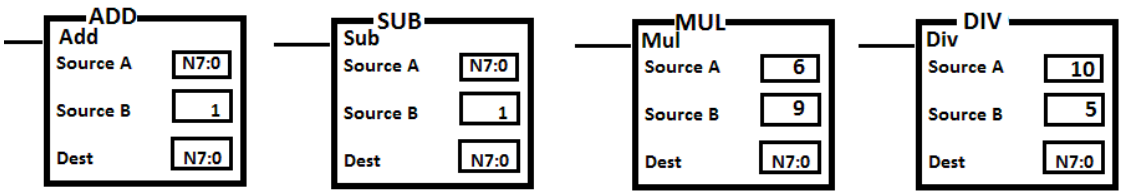
**Allen Bradley/ Rockwell PLC:** it is USA brand PLC and one of the leading brands.



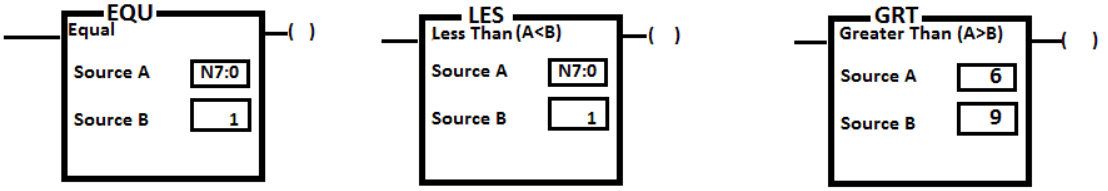
It has also micro and modular both types of PLC. it requires two software. One is RS Linx used for communication setup. Another software RS Logix is for programming. Input address is I:0/0, I:0/1 and continue in decimal number system. Output address is O:0/0, O:0/1 and continue in decimal number system. Flag address is B3:0/0, B3:0/1 and continue in decimal number system. Positive pulse is OSR in which two flags are used. One flag is as storage bit and another output bit. Output bit/ flag 2 is used further as pulse.



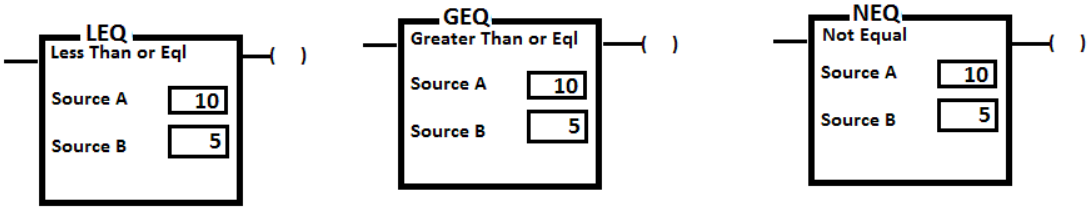

Negative pulse is OSF in which two flags are used. One flag is as storage bit and another output bit. Output bit/ flag 2 is used further as pulse. Timer is T4 and first timer address is T4:0. 256 timers can be used. Timer done bit is T4:0/DN. There is no need of flag as done bit. T4:0/Pre and T4:0/Ac is generally used in HMI and SCADA for timer pre-set and accumulator display. Counter is C5 and first counter address is C5:0. 256 counters can be used. Counter done bit is C5:0/DN. There is no need of flag as done bit. C5:0/Pre and C5:0/Ac is generally used in HMI and SCADA for timer pre-set and accumulator display.

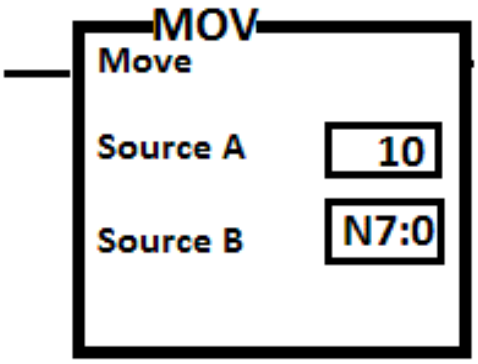



N7 is variable in RS Logix




N7 is variable in RS Logix




**Variable: N7:0, B3:0.0, O:0.0**

**SIEMENS: German brand.**

SIEMENS
SIMATIC

| HARDWARE | SOFTWARE |
|---|---|
| S7 200 | S7 microwin |
| S7 1200 | TIA portal |
| S7 300<br>S7 400<br>S7 1500 | SIMATIC MANAGER |

S7 200 and S7 1200 are micro PLCs.

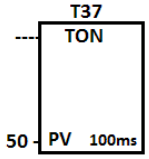
S7 300, S7 400 AND S7 1500 are modular PLCs.

| INPUT | OUTPUT | FLAG |
|---|---|---|
| I0.0 | Q0.0 | M0.0 |
| I0.1 | Q0.1 | M0.1 |
| . | . | . |
| . | . | . |
| I0.7 | Q0.7 | M0.7 |
| I1.0 | Q1.0 | M1.0 |
| . | . | . |
| . | . | . |
| . | . | . |
| I1.7 | Q1.7 | M1.7 |

OCTAL NUMBER SYSTEM

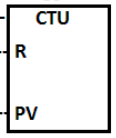
| POSITIVE PULSE | NEGATIVE PULSE | SET | RESET |
|---|---|---|---|
| --I P I-- | --I N I-- | M0.0<br>--( S )<br>1 | M0.0<br>--( R )<br>1 |

TIMER
T37



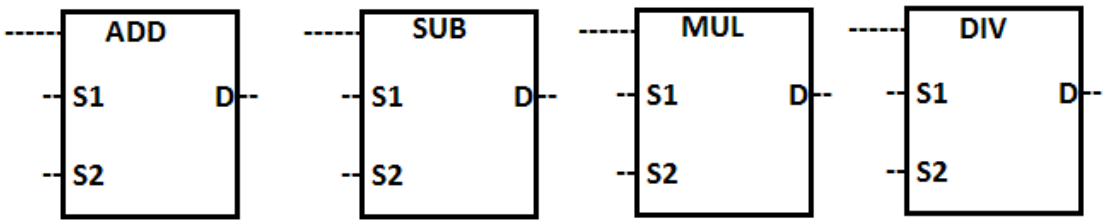

DN & ACC-- T37

COUNTER
C0




DN and ACC-- C0

## ARITHMETIC FUNCTION

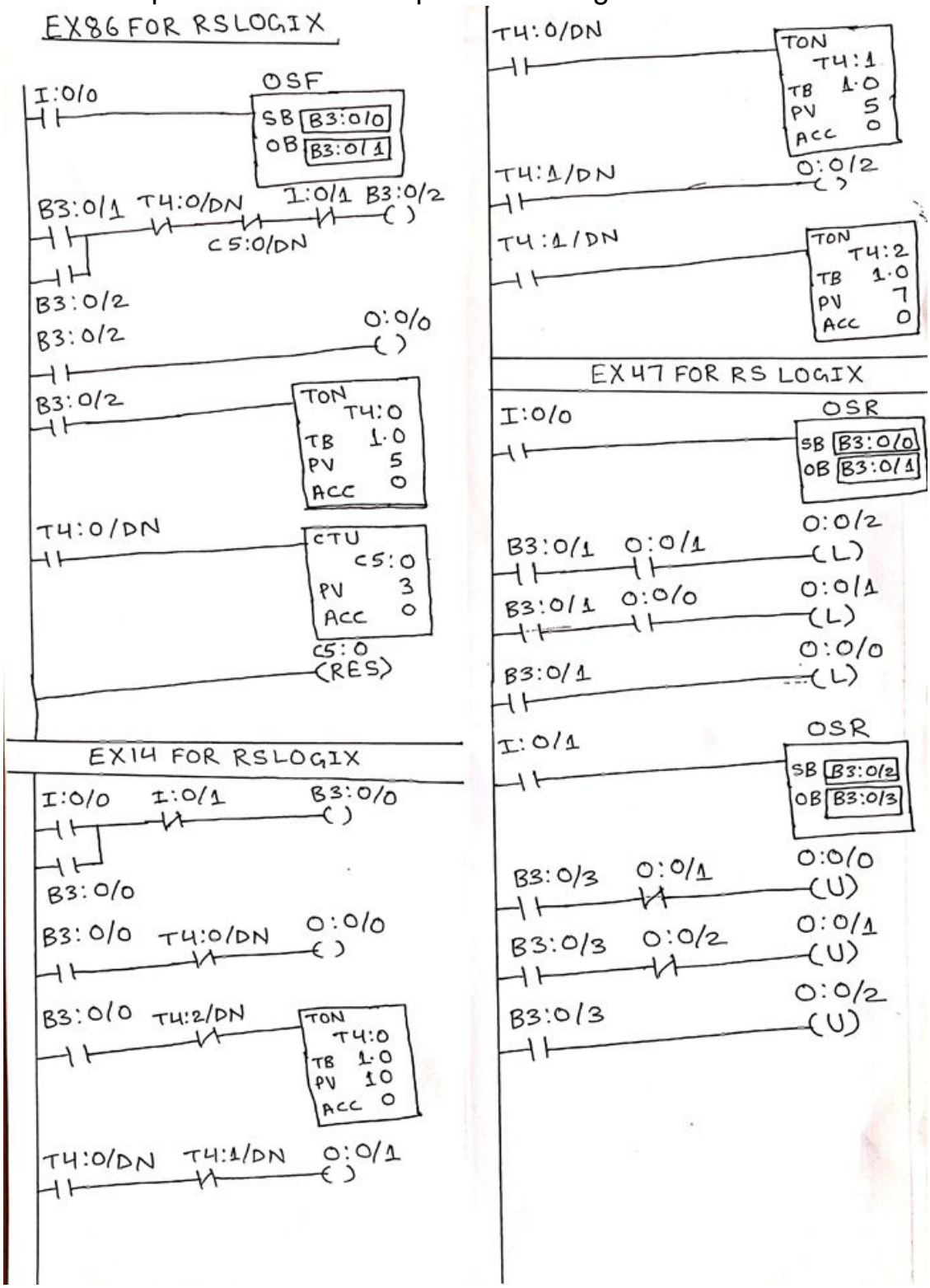


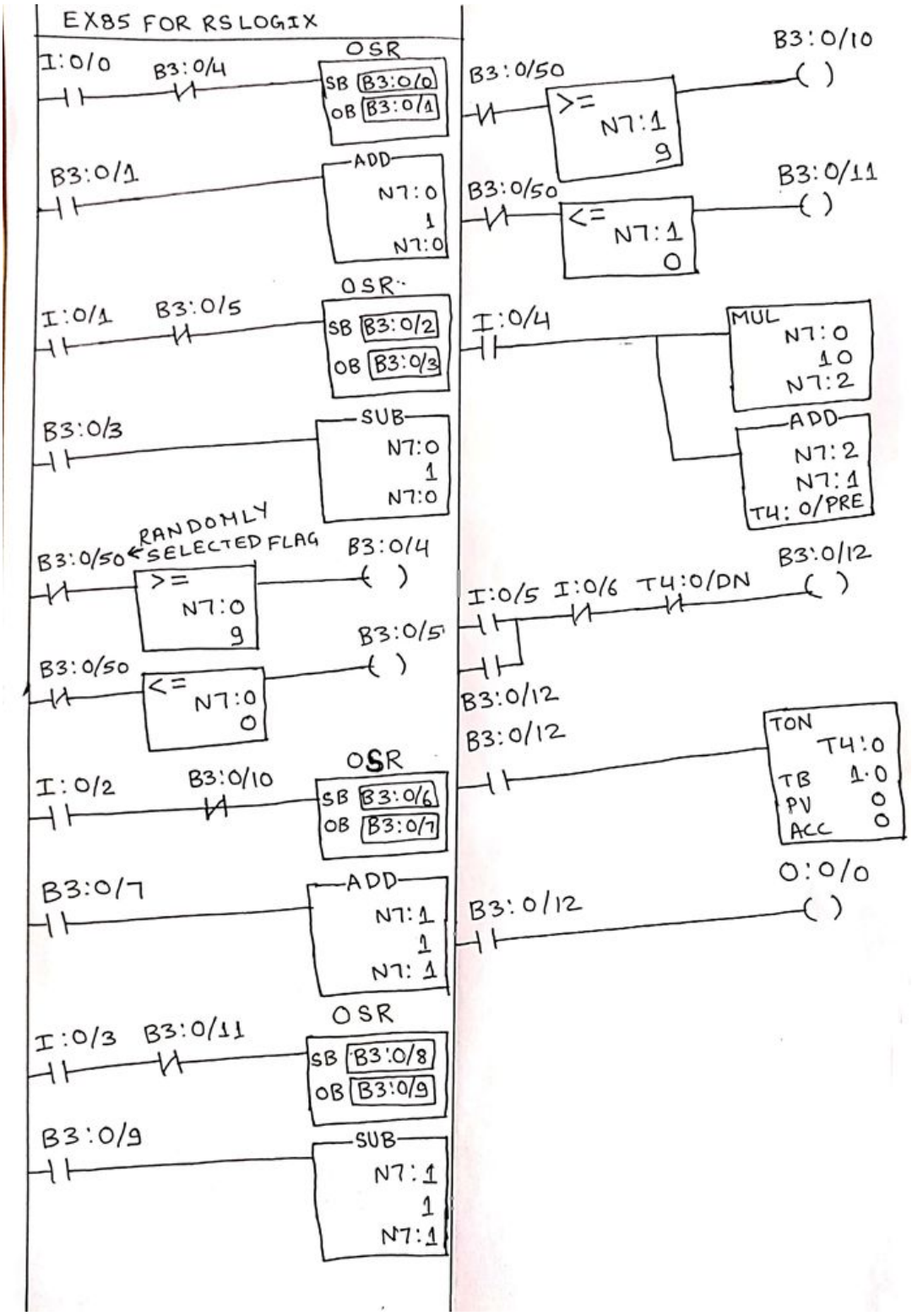

## COMPARATOR FUNCTION

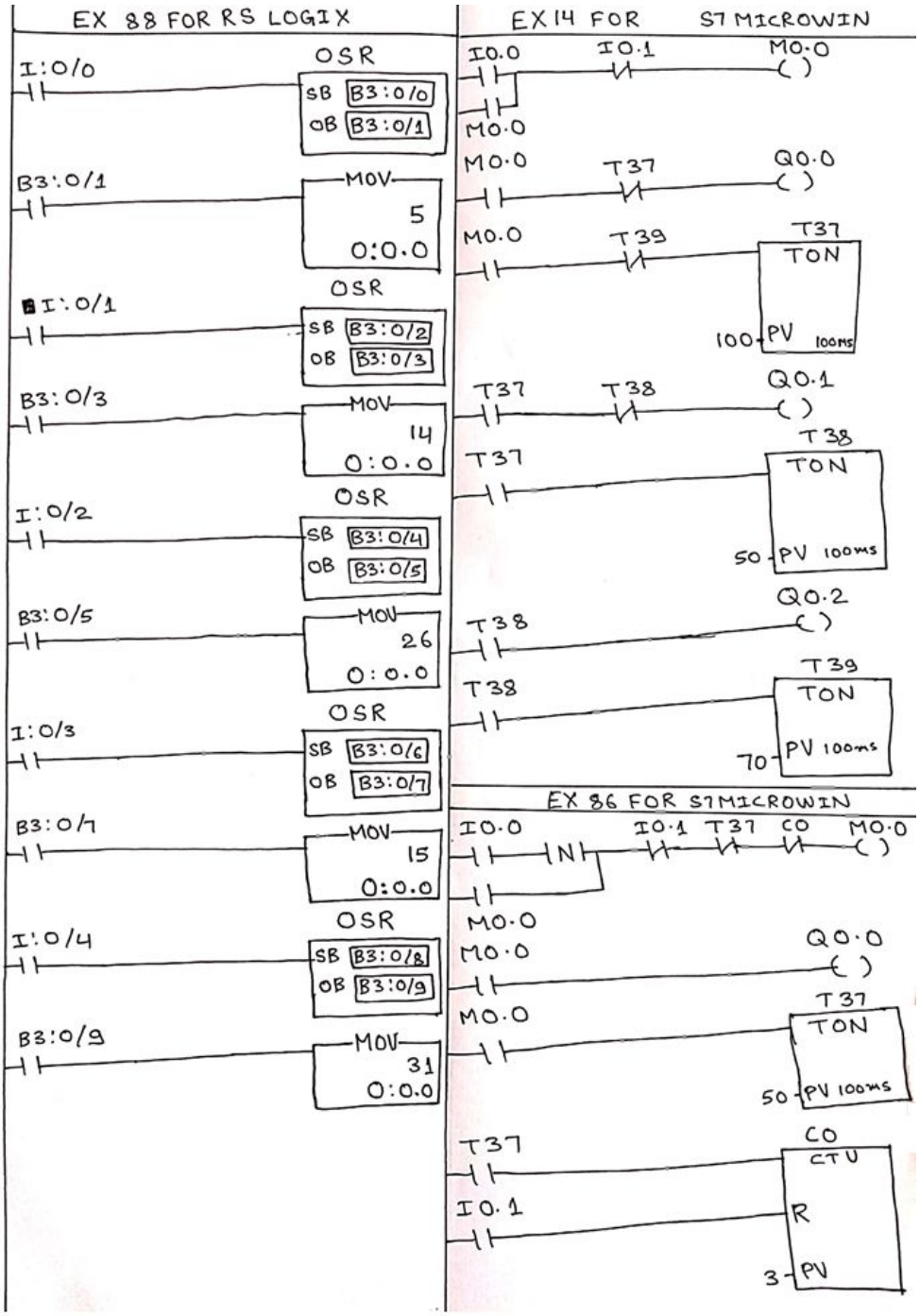
| Equal | Greater Than | Less Than | Greater Than Equal |
|---|---|---|---|
| --I I---I == I-----( ) | --I I---I > I----( ) | --I I---I < I----( ) | --I I---I >= I----( ) |

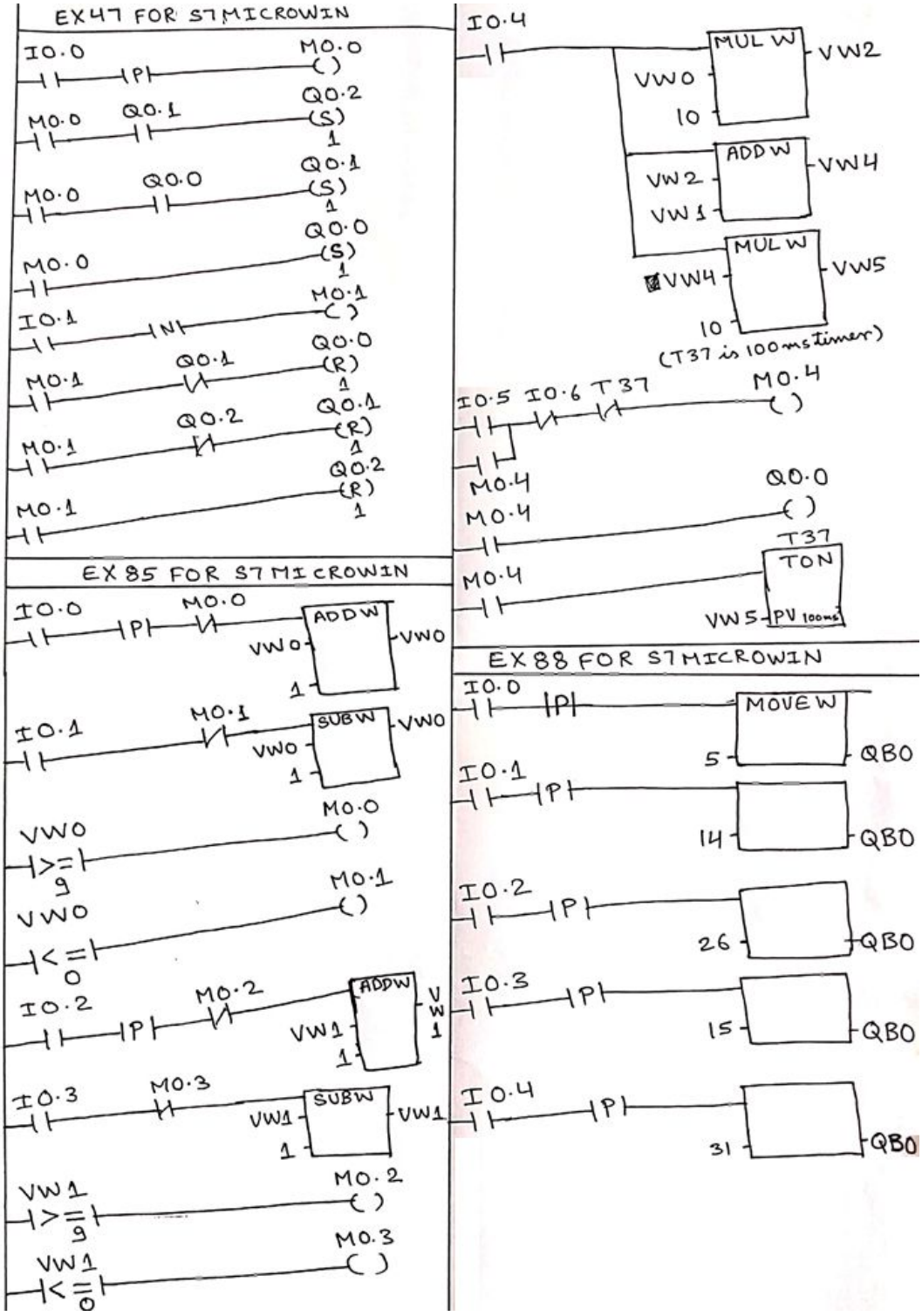
| Less Than Equal | Variable: VB0, VW0, VD0, MB0, MW0, MD0, QB0, QW0, QD0, ACC0. |
|---|---|
| --I I---I <= I----( ) | |

Let us practice some examples for RS logix and S7 microwin.




EX85 FOR RSLOGIX
I:0/0
B3:0/4
OSR
SB B3:0/0
OB B3:0/1
B3:0/1
ADD
N7:0
1
N7:0
I:0/1
B3:0/5
OSR
SB B3:0/2
OB B3:0/3
B3:0/3
SUB
N7:0
1
N7:0
B3:0/50
RANDOMLY SELECTED FLAG
>=
N7:0
9
B3:0/4
B3:0/50
<=
N7:0
0
B3:0/5
I:0/2
B3:0/10
OSR
SB B3:0/6
OB B3:0/7
B3:0/7
ADD
N7:1
1
N7:1
I:0/3
B3:0/11
OSR
SB B3:0/8
OB B3:0/9
B3:0/9
SUB
N7:1
1
N7:1
B3:0/50
>=
N7:1
9
B3:0/10
B3:0/50
<=
N7:1
0
B3:0/11
I:0/4
MUL
N7:0
10
N7:2
ADD
N7:2
N7:1
T4:0/PRE
I:0/5
I:0/6
T4:0/DN
B3:0/12
B3:0/12
B3:0/12
TON
T4:0
TB 1.0
PV 0
ACC 0
B3:0/12
O:0/0


EX 88 FOR RS LOGIX
I:0/0
OSR
SB B3:0/0
OB B3:0/1
B3:0/1
MOV
5
0:0.0
I:0/1
OSR
SB B3:0/2
OB B3:0/3
B3:0/3
MOV
14
0:0.0
I:0/2
OSR
SB B3:0/4
OB B3:0/5
B3:0/5
MOV
26
0:0.0
I:0/3
OSR
SB B3:0/6
OB B3:0/7
B3:0/7
MOV
15
0:0.0
I:0/4
OSR
SB B3:0/8
OB B3:0/9
B3:0/9
MOV
31
0:0.0
EX 14 FOR S7 MICROWIN
I0.0
I0.1
M0.0
M0.0
M0.0
T37
Q0.0
M0.0
T39
T37
TON
100
PV 100ms
T37
T38
Q0.1
T37
T38
TON
50
PV 100ms
Q0.2
T38
T38
T39
TON
70
PV 100ms
EX 86 FOR S7MICROWIN
I0.0
I0.1
T37
C0
M0.0
M0.0
M0.0
Q0.0
M0.0
T37
TON
50
PV 100ms
T37
C0
CTU
I0.1
R
3
PV



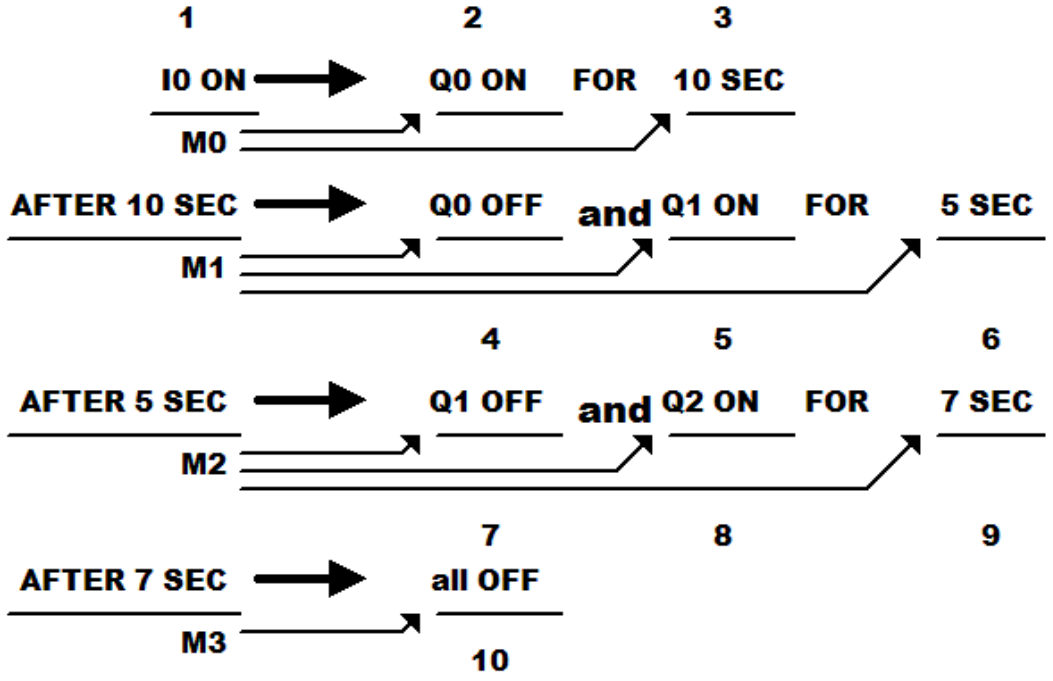

**Operation Mode:** Machines have two operation modes; Auto mode and Manual Mode. Already we have gone through auto mode programming. Let us learn manual mode programming. It is also called maintenance mode. In

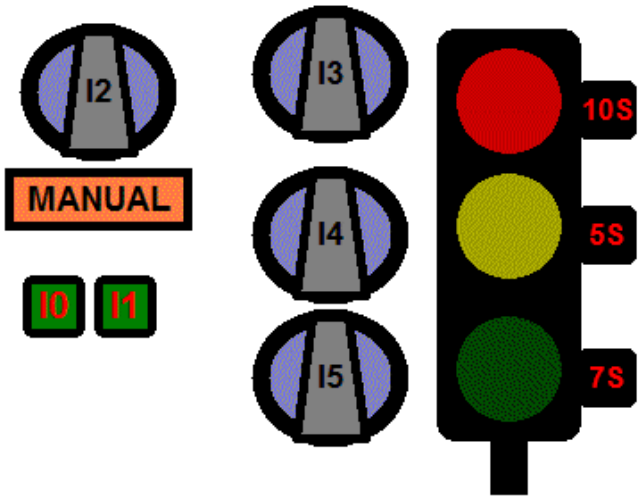
auto mode, each output of machine gets monitored and controlled by controller PLC. In manual mode, we give optional individual control of output to operator. In manual mode operator can operate individual output any time as per need. During maintenance also we need to control output as per our requirement. Suppose there is a chain conveyor and we need to lubricate its chain and gears. After lubrication, conveyor should run for some duration so that lubricant can spread properly. Sometimes we need to operate individual output to check issues occurred in it. It is not very difficult to add manual logic in existing auto program. Some hardware and some logic modifications are required. Let us understand the modification using an example.

**EX93:**

There are 3 outputs Red lamp, Yellow lamp and Green lamp. There are two inputs I0 and I1 for start and stop. When I0 is pressed, pump Q0 should get on for 10 seconds. After 10 seconds Q0 off and valve Q1 should get on for 5 seconds. After 5 seconds Q1 off and motor Q2 should get on for 7 seconds. The same sequence is applicable on another system too. You can see another system also has 2 inputs and 3 outputs and the same sequence. So it does not matter what is the existence of physical output, program will be the same.

Let us write the steps:





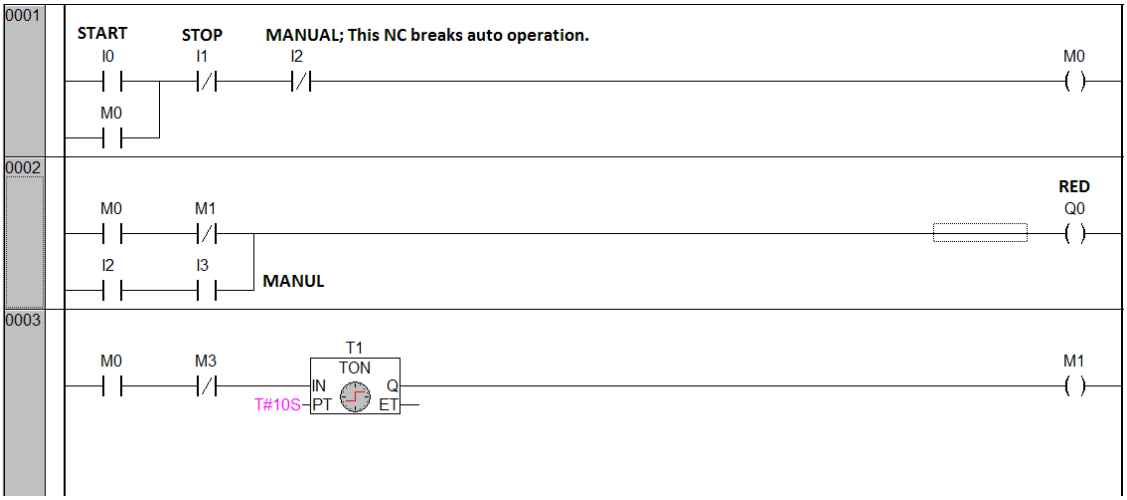

For manual operation we need to take one selector switch for manual selection. When manual selector is on then auto program should not work. Selector switch has toggle property. With this manual selector we need selector switch for each output of the machine so we have taken three more selector switches for three outputs.

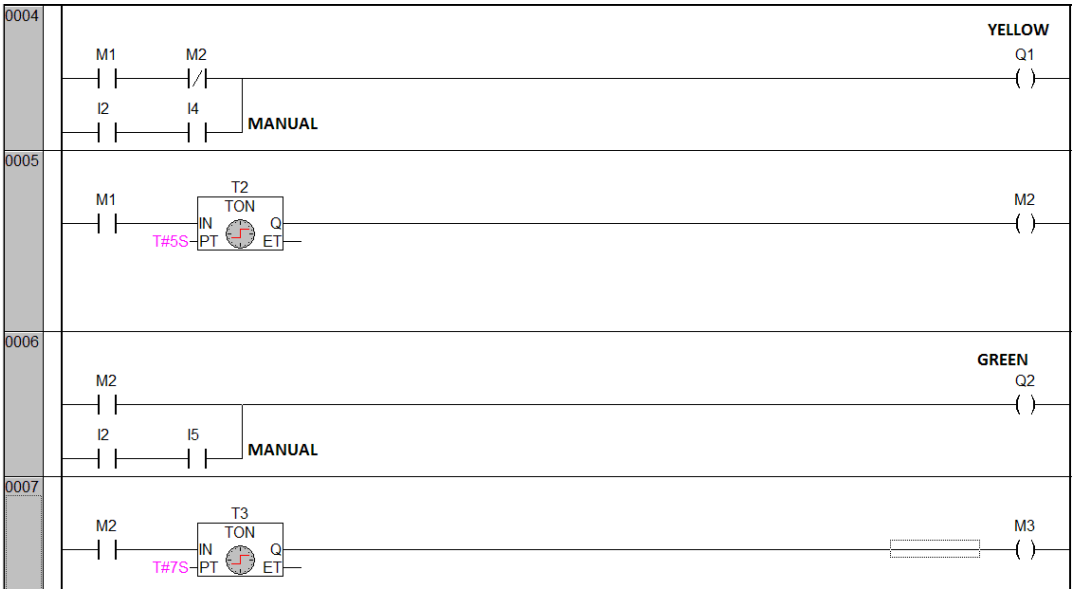
For manual operation of each output is as given below:

I2 + I3 □ Q0 on,

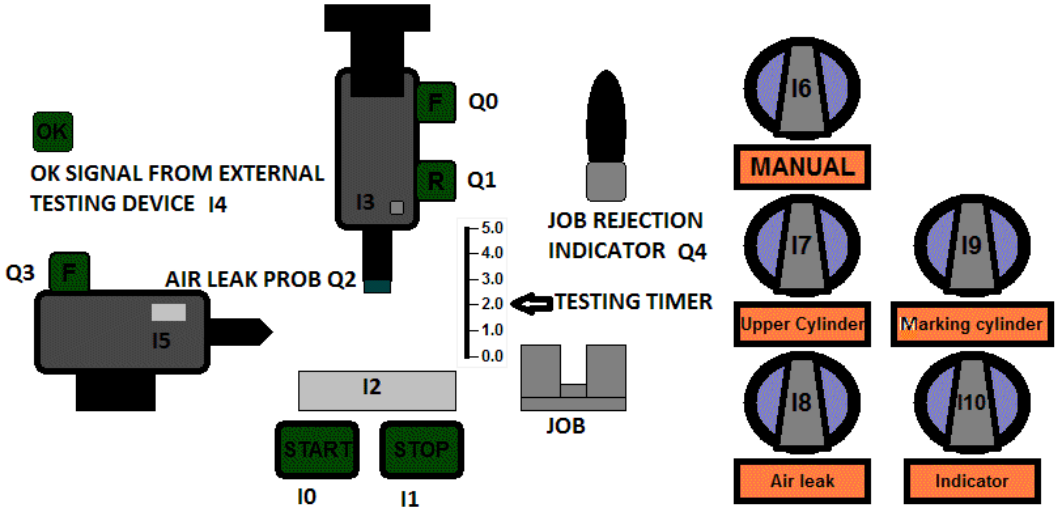
I2 + I4 □ Q1 on and

I2 + I5 □ Q2 on.





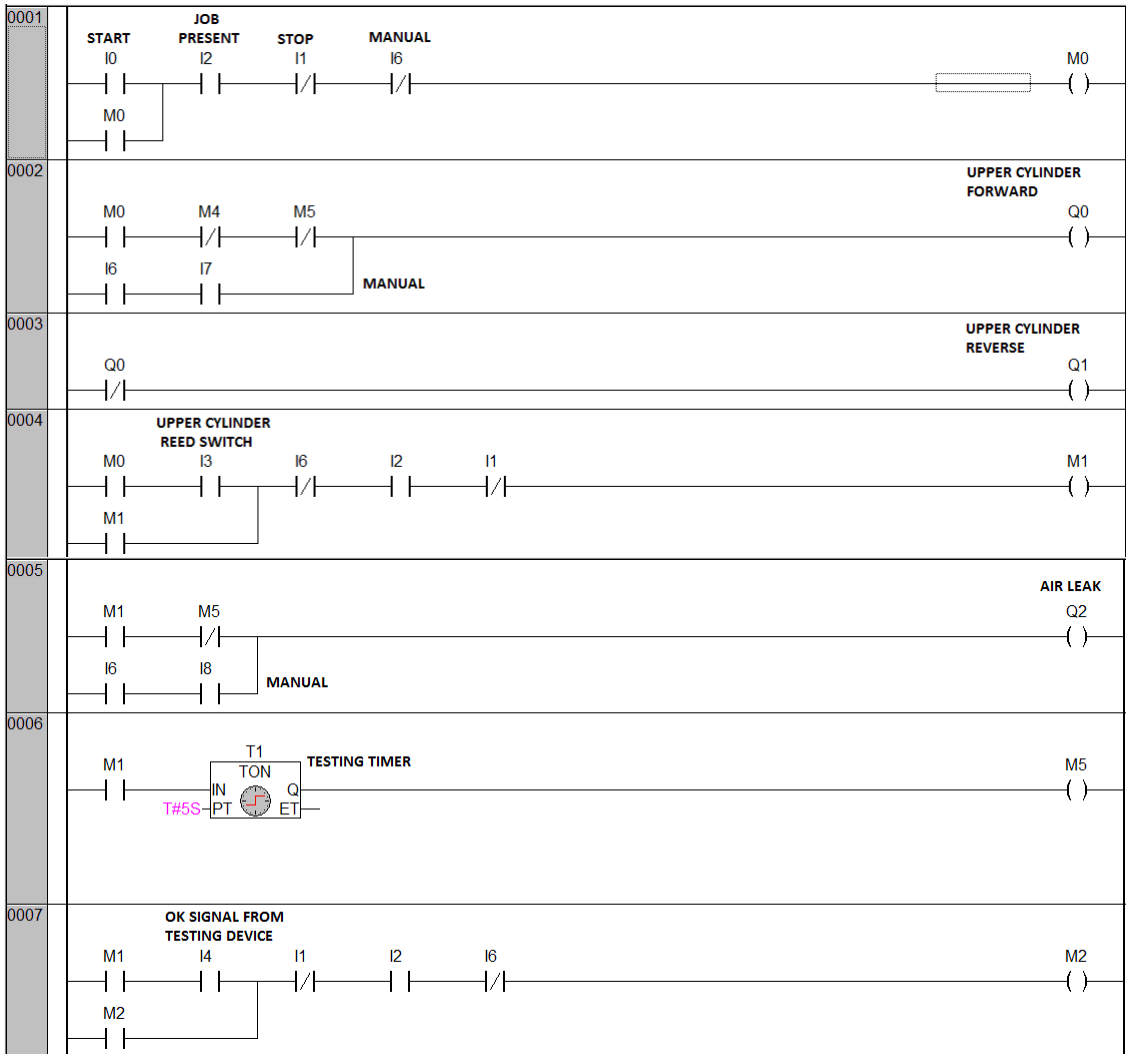

## EX94: JOB TESTING/ SPECIAL PURPOSE MACHINE (SPM)

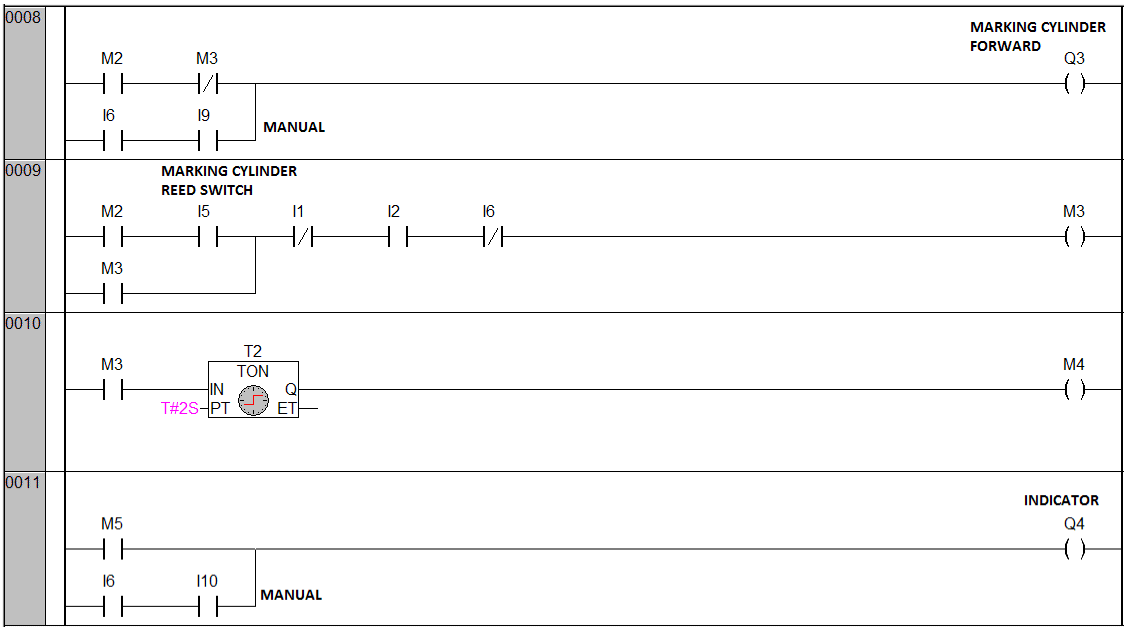



When metallic job is put on work station, proximity sensor I2 gets on. It confirms the presence of job on work station and it must be on to run the machine. I0 start push button is pressed and upper cylinder piston moves forward i. e. Q0 is on. An air leak probe is attached with this piston which will leak air inside job to check the surface accuracy of the job. As piston reaches final position/ I3 is on, Q2 will get on and air will be leaked inside the job for five seconds. If OK signal I4 comes within five seconds then Q2 will be off and Q3 will be on resulting punching piston move forward to punch a mark on job. As piston reaches to final position/ I5 is on, Q3 will get off and piston will return back to home position. Now system will wait for 2 seconds and after two seconds upper cylinder piston will move back i. e. Q0 off and

Q1 on. System will not start until tested job is removed from the work station. If one cycle is finished and I0 is pressed, job is not removed and you press I0 then system should not start.

Another condition for job test is; if OK signal I4 does not come within 5 seconds and timer done bit gets high then air leak Q2 will be off and job rejection indicator Q4 will be on. At the same time upper cylinder piston will move in reverse direction i. e. Q0 is off and Q1 is on. Q4 indicator and system will get reset by removing tested job from work station.






So, it is not difficult to add manual logic in auto program. Manual is added in parallel to each output ladder.

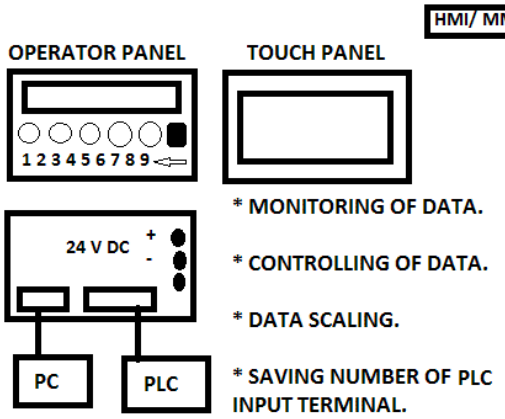
**HMI/ MMI and SCADA:**

With PLC one machine can work automatic but for a good machine there are two major requirements to be fulfilled; **Monitoring of DATA and controlling/ handling of data**. Monitoring of data means we need to monitor the present status of addresses used in our program. We need to monitor the status of input outputs. We need to monitor the status of timer and counter. We need to monitor the status of temperature, pressure, flow, level, displacement etc. on single platform. There is one possibility. PLC software has monitor/ online mode. We can receive existing program of PLC in PLC software. There we have option monitor/ online mode. As we click it we can monitor the present status of each and every address used in our program. Practically it is not possible to use PLC software for monitoring as one operator has to monitor the status and operator does not know to handle PLC software usually. Another problem is that we will have to scroll program up and down as it would be very big program for a good machine. It will kill time also. Once PLC is ready, it is locked in control panel and it is not allowed to connect our laptop/ computer any time to it. Without monitoring the present status of data we will not be able to understand what is going on in machine. Many input outputs will be inside machine where we cannot reach and we need to know the present status of those field devices too. Monitoring of data is very important for a good and flexible system.

Controlling/ handling of data are also important. Generally, in a good flexible system we require change of numerical value on some data locations such as pre-set value of timer, counter, temperature, pressure, flow, level, displacement etc. without changing program. Suppose there is a machine which runs for 10 minutes for product 1. The same timer should run for 30 seconds for product 2 and it should run for 25 minutes for product 3. Such requirement we can have in packaging machines. For product 1, 20 jobs should be packaged in each box. For product 2, 10 jobs should be packaged in each box and for product 3, 12 jobs should be packaged in each box. There is a boiler. For product 1 it should be 100*C, for product 2 it should be 1200*C and for product 3 it should be 600*C. So here you see that we need to change timer/ counter/ temperature pre-set value as per our requirement any time. Again, there is possibility i. e. PLC software. We can receive existing program of PLC, can modify the pre-set value and can rewrite the program to PLC. Again, operator will not have access to modify program and it would be a big program so whole process of modification will kill time.

In industry with a good automatic system, we use HMI/ MMI and SCADA.

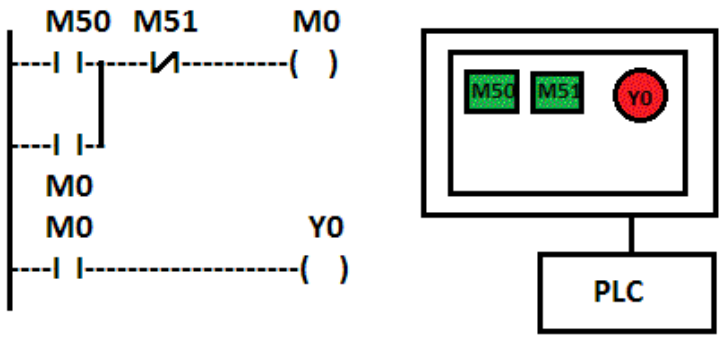
**Human Machine Interface/ Man Machine Interface (HMI/ MMI):** HMI is an interfacing device which monitor and controls the data of controller.




| MAKE | HARDWARE | SOFTWARE |
|---|---|---|
| MITSUBISHI | GOT | GT DESIGNER |
| ALLEN BRADLEY | PANEL VIEW | PANEL BUILDER |
| SIEMENS | OP/ TP | WIN CC FLEX/ TIA |
| BEIJER | E-TERMINAL | E DESIGNER |
| DELTA | DOP | DOP SOFT |

Through HMI we can scale data. For example, if you remember Mitsubishi timer, we needed to multiply pre-set value with 10 (50 for 5 seconds). Through HMI if we enter pre-set value then we don't need to enter 50 for 5 seconds but we can scale it and we just need to enter 5 for 5 seconds in HMI and to PLC it will go 50.

When we use HMI then we can replace manual switches/ inputs with the virtual keys of HMI. Virtual keys will operate flags in our program to execute logic. In this way a smaller number of PLC input terminal will be required and PLC cost will be reduced.

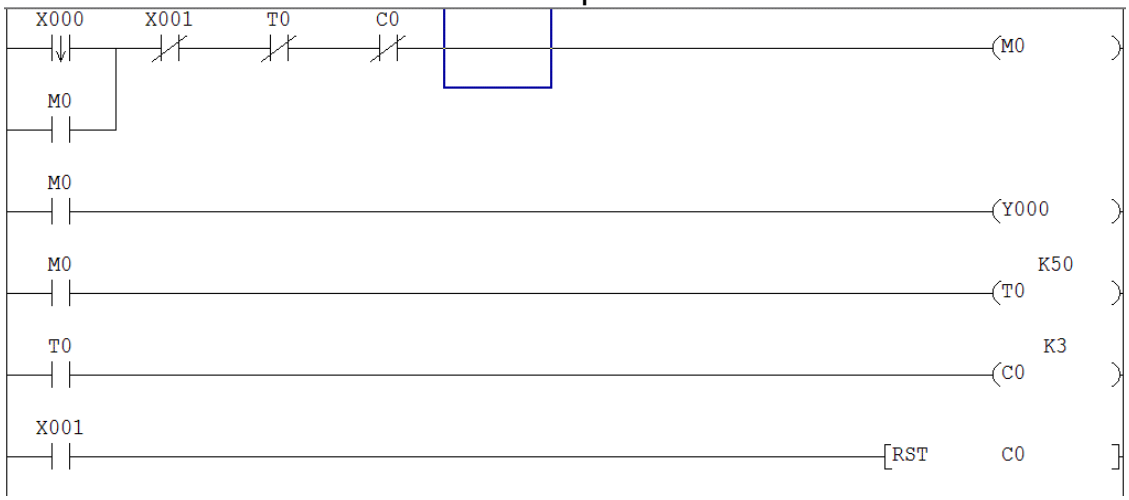



You can see that instead of physical inputs X0 and X1 we have used random flag M50 and M51 as start& stop. These flags are being operated by HMI virtual key.

According to hardware we have two types of HMI hardware; operator panel and touch panel. Operator panel has one small screen, readymade function keys, numeric keys, backspace and enter key. Touch panel has just one screen and keys or other things we design in HMI software using its design tools. In one good touch screen complete machine can be designed. In HMI software we have design library. There are many readymade pictures available there and we just need to pick and place. Behind any HMI we find power terminal and 24 V DC is required generally in maximum of HMIs. We find minimum two communication ports also behind HMI. One port is used to interface PLC with HMI and through this HMI is always connected to PLC. Another port is programming port which is used to connect PC/ Laptop to send or receive HMI design. There are many HMI manufacturers. I have shown a few here. The question is, can we interface any brand of HMI with any brand of PLC? Yes, we can interface any brand of HMI with any brand of PLC, the requirement is that HMI software must have that PLC driver file/ PLC name in its list of controllers.

Screen development concept is almost the same for any brand of HMI. Let us learn the important concept. Very first if you want to change numeric value of PLC data through HMI/ SCADA then a small modification is required in PLC program. The data location where you want to change numeric value, that location must have variable address not a fixed number.

Example:

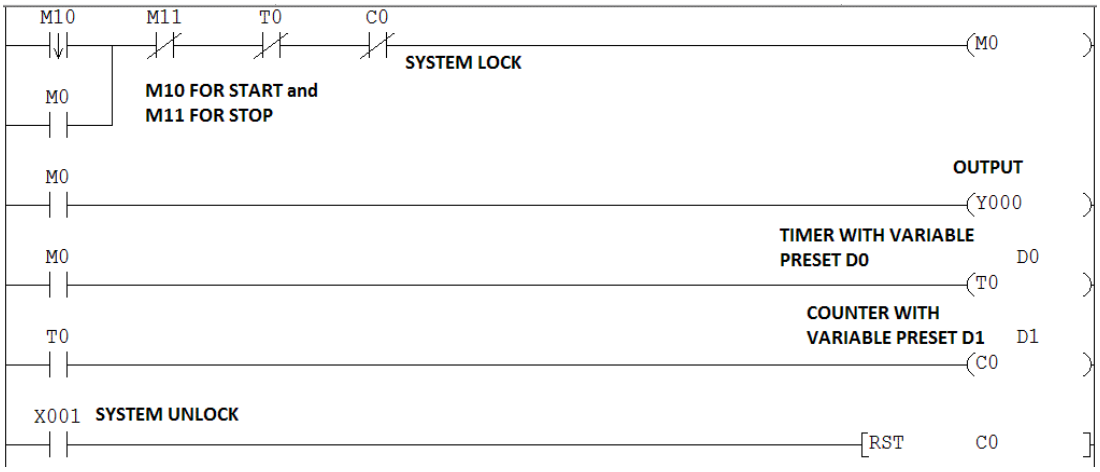



In this program when we press push button X0 then output Y0 gets on for 5 seconds. After 5 seconds output is off and counter counts one cycle. Like this every time when X0 is pressed, cycle runs and is counted. After 3 cycles the system gets locked by counter done bit c0. To unlock the system X1 is pressed. X1 is used for emergency stop too.

Let us decide now that what we have to do from HMI.

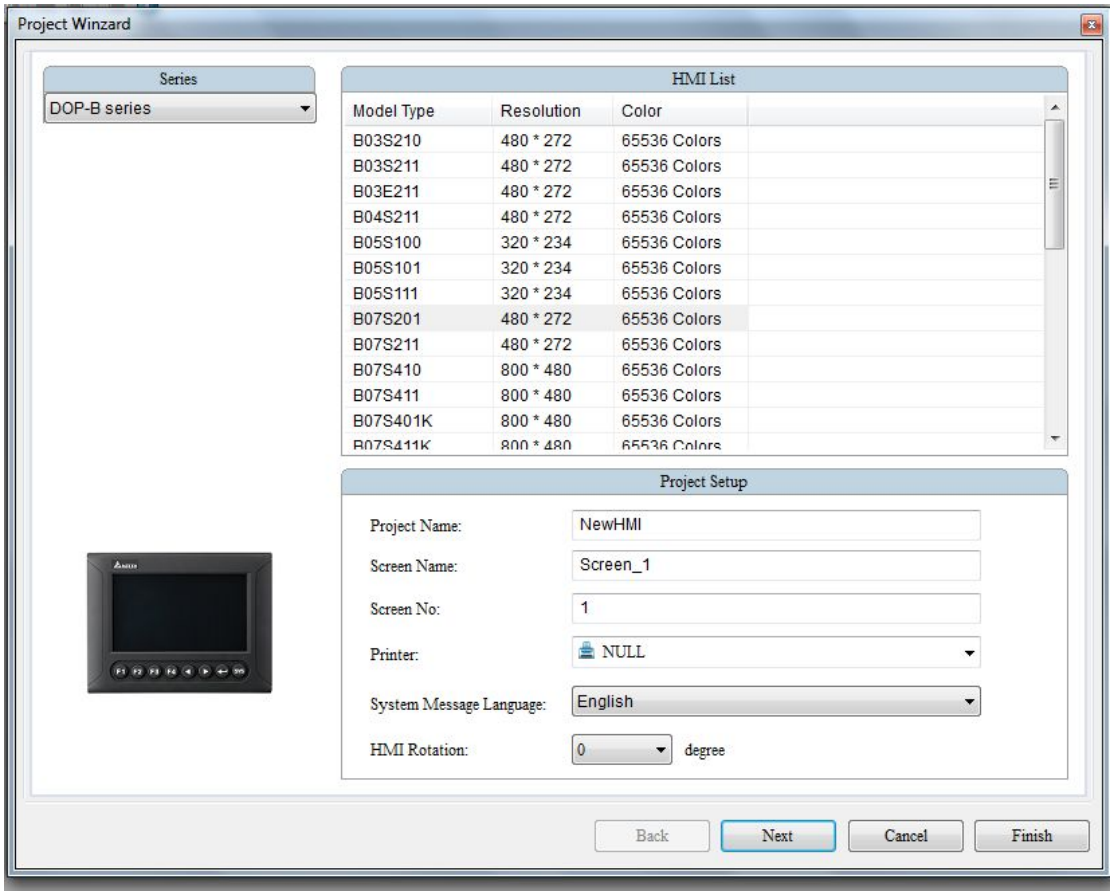
1> As we have HMI so we shall not use physical button X0 and X1. Operate start and stop by HMI keys.
2> Display the status of output on HMI.
3> Display the running/ present/ accumulator value of timer on HMI.
4> Display the running/ present/ accumulator value of counter on HMI.
5> Change pre-set value of timer from HMI.
6> Change pre-set value of counter from HMI.
7> Display SYSTEM LOCK on HMI when system is locked.

In our program we shall replace x0 and X1 with any two flags. Let us take M10 and M11. To change pre-set value of timer we will have to replace K50 with variable address and in Mitsubishi it is D. let us take D0. For counter pre-set change from HMI we shall replace K3 with D1. Let us modify program.

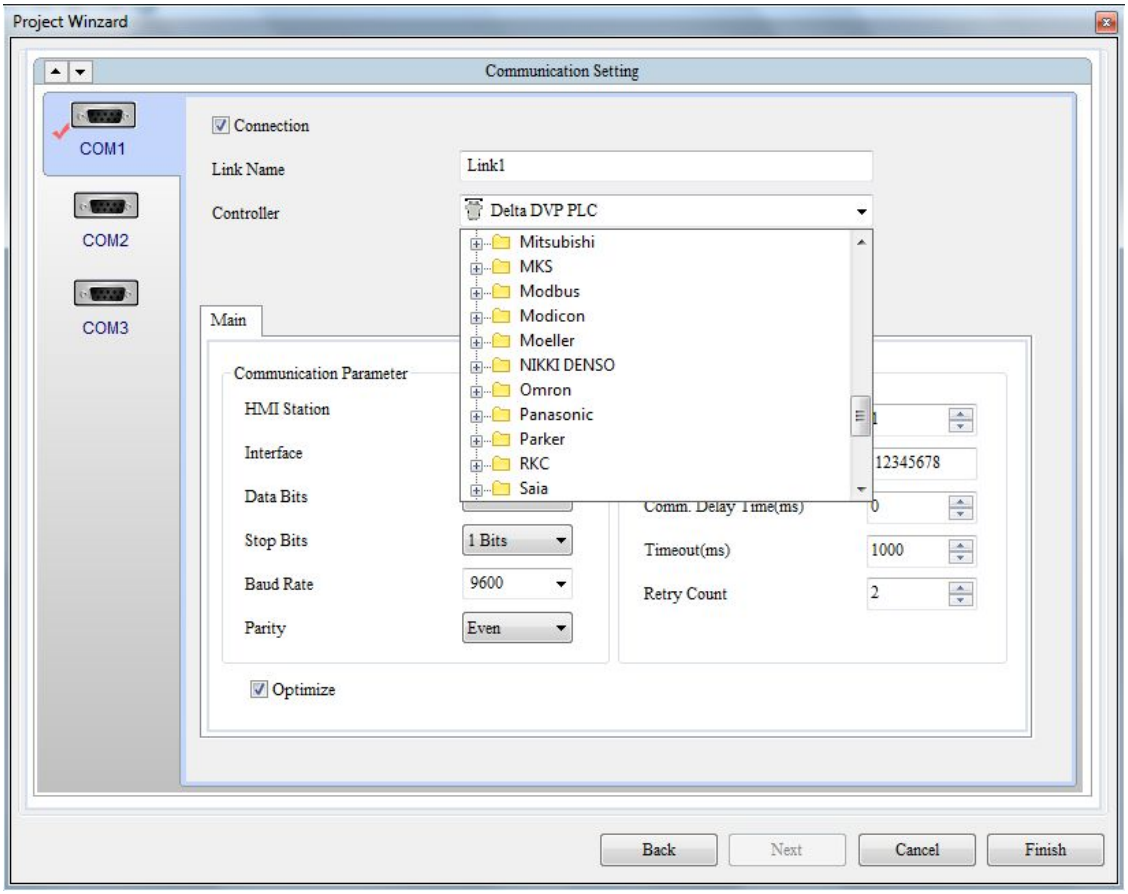



So, we have modified program as per requirements. Now let us make a list of data to be configured in HMI.

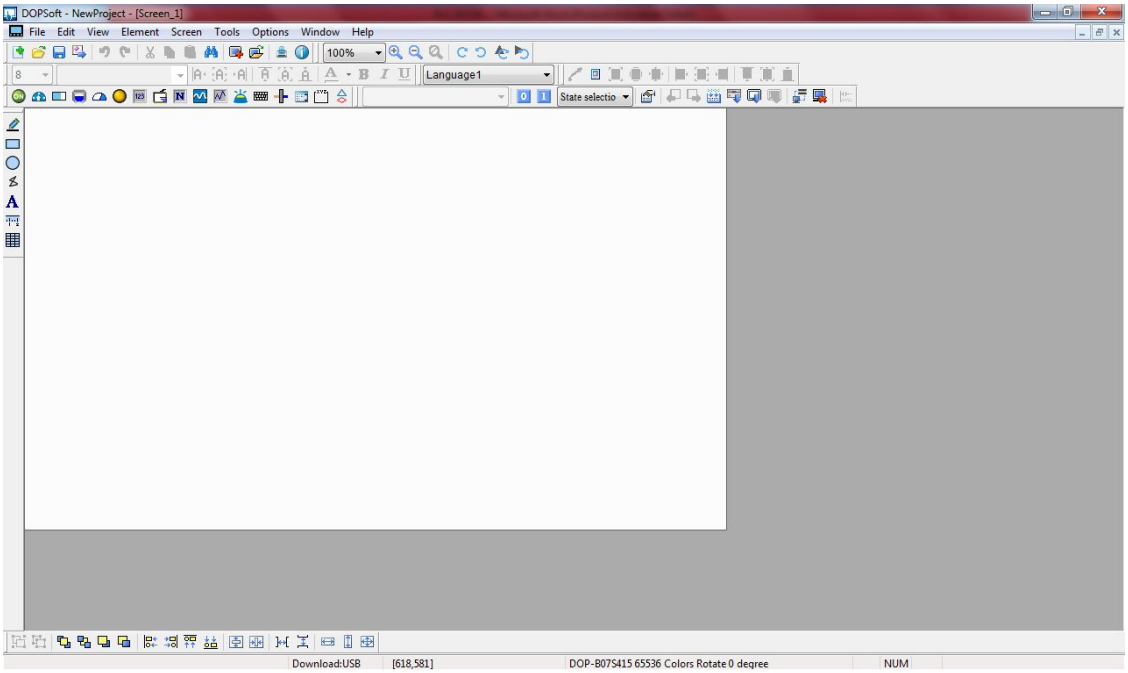
1> M10 □ START
2> M11 □ STOP
3> Y0 □ LAMP
4> T0 □ TIMER ACCUMULATOR/ RUNNING VALUE/ PRESENT VALUE
5> C0 □ COUNTER ACCUMULATOR/ RUNNING VALUE/ PRESENT VALUE
6> D0 □ TOMER PRESET VALUE
7> D1 □ COUNTER PRESET VALUE
8> C0 □ SYSTEM LOCK




As we take new file in any HMI software, it asks to select HMI hardware model name. This model name you will find on HMI hardware.



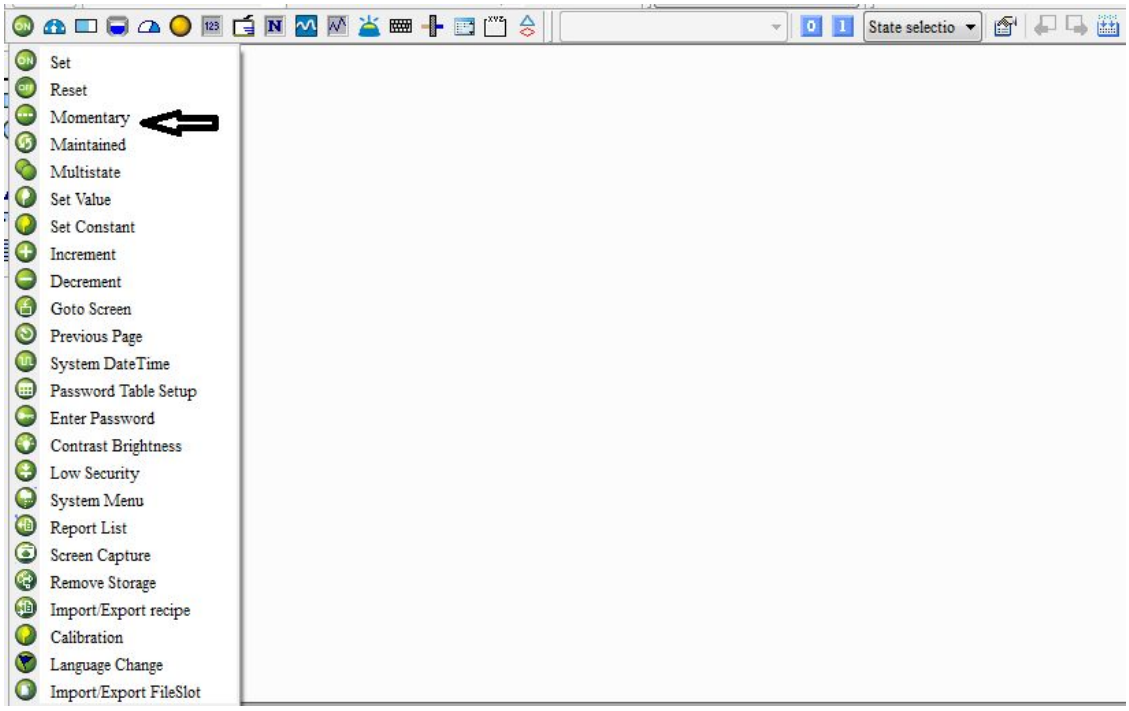

After selecting HMI model name, it will ask you to select HMI communication port number through which you have connected your PLC to it. It will also ask you to select your PLC brand and PLC model name. It will show you a list of branded PLCs where you can select your PLC brand and model. If your PLC is not in that list then you need to download or purchase driver file for your PLC brand.



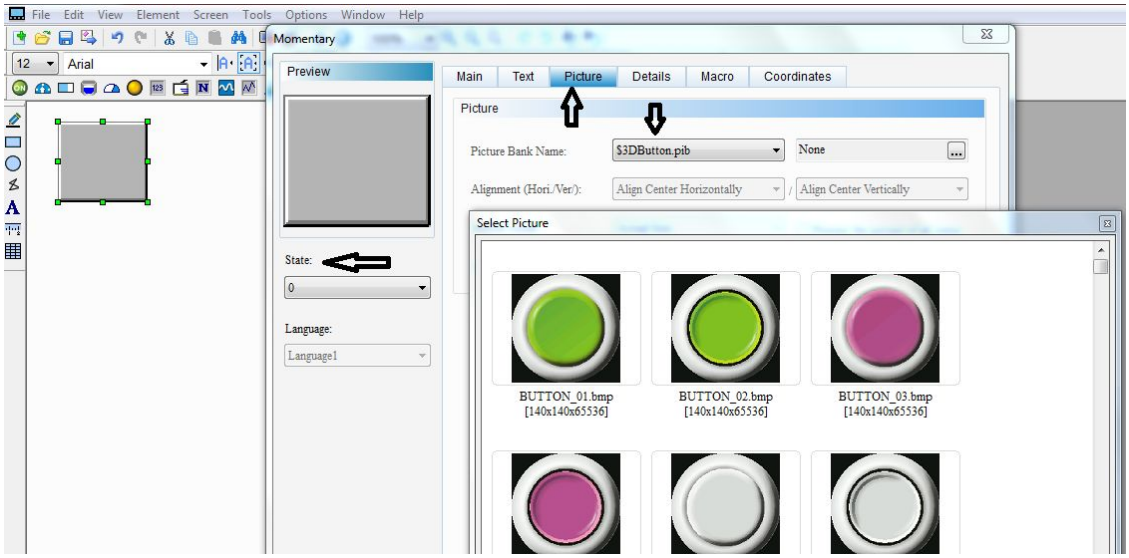

After that, main screen will be appeared with all kind of tools. Here you can design your machine as per your requirements. Design depends on you but there are some elements which we have to design compulsorily. So, when you work on new brand of HMI then you search five important tools for screen development for your machine; Input/ Switch, Output, Pre-set value/ Numeric entry, Accumulator/ numeric display/ meter display and text display. If you are able to search these elements then you will be able to monitor and control the data of your controller/ PLC. You can take help of user guide of that HMI software.

**Input/ Switch:**

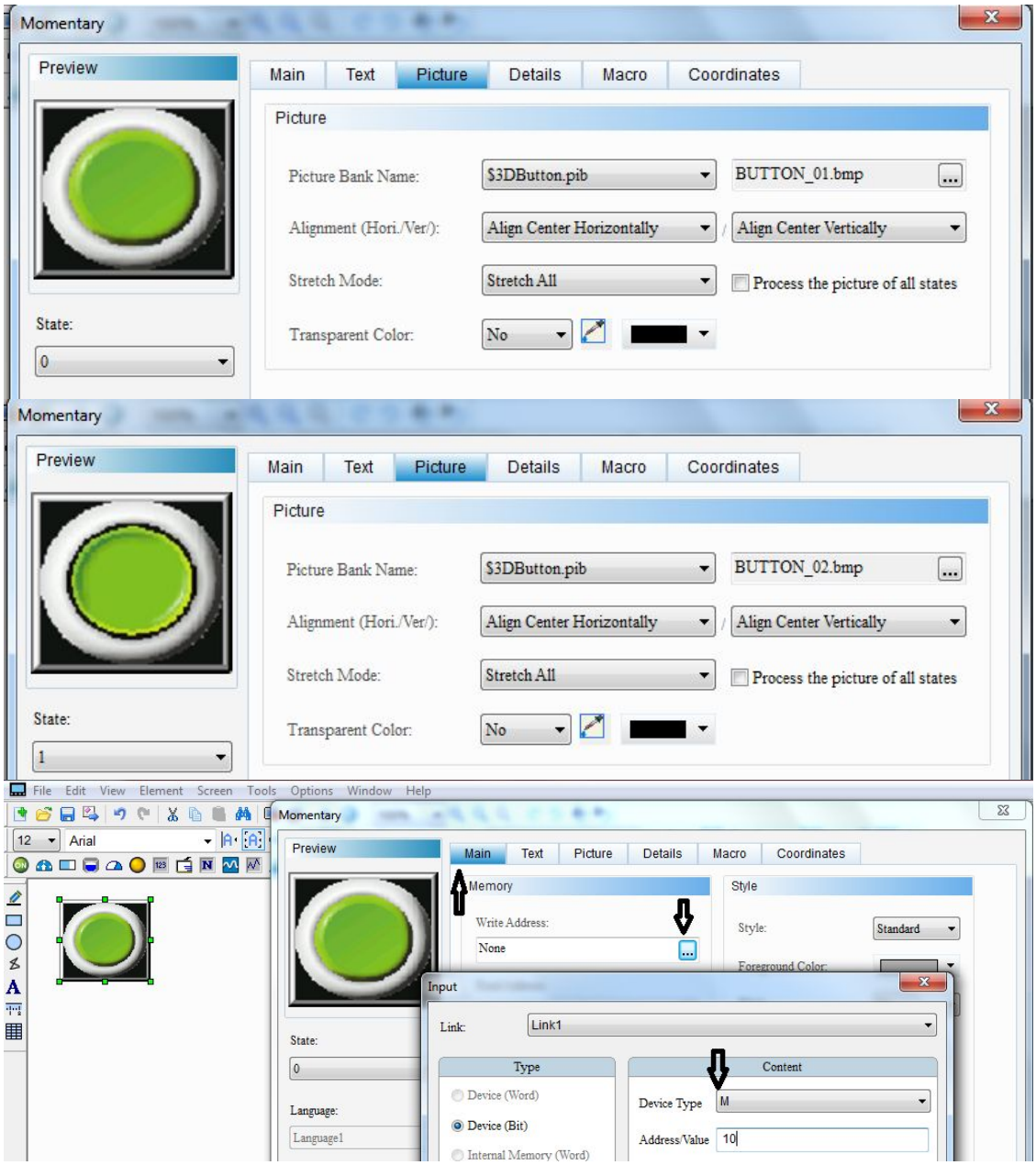

In different software we shall find different name of element like bit switch or input etc. Here it is button.




In button we have many types of switch and we have to select momentary for push button. Somewhere it can be tap or push given for push button.

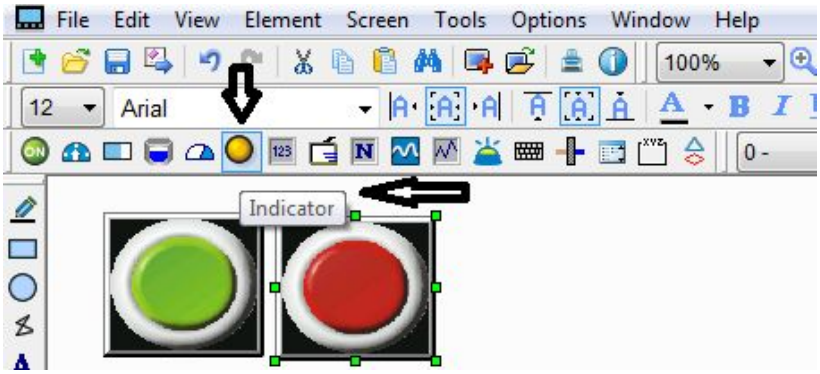



We click momentary and design a switch. We double click that switch and we shall get a lot option to design or modify it. We have two options here; either we take two colours for 0/ off state and 1/ on state of the switch or we can go to picture and there we have 3Dbutton and we can select desired switch design. In some software you will find library for designs.

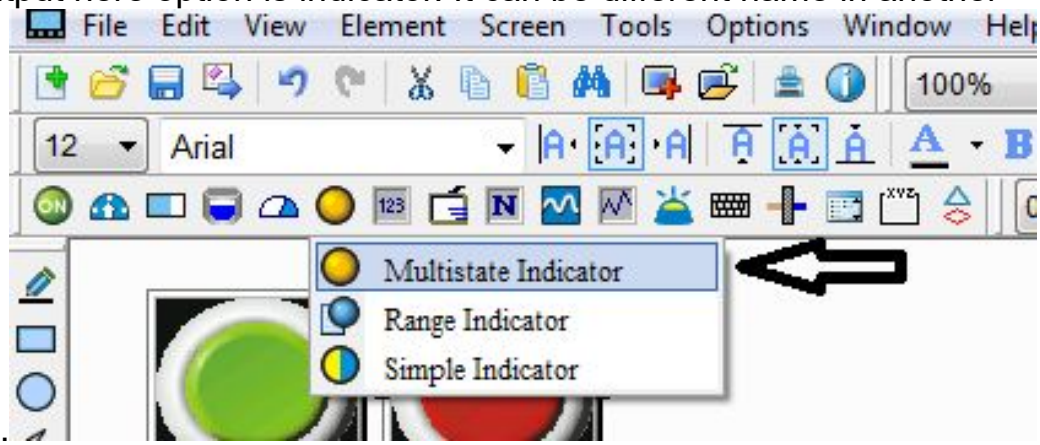



After selecting two pictures for Off and on state we go to main and there we give the address of that switch.

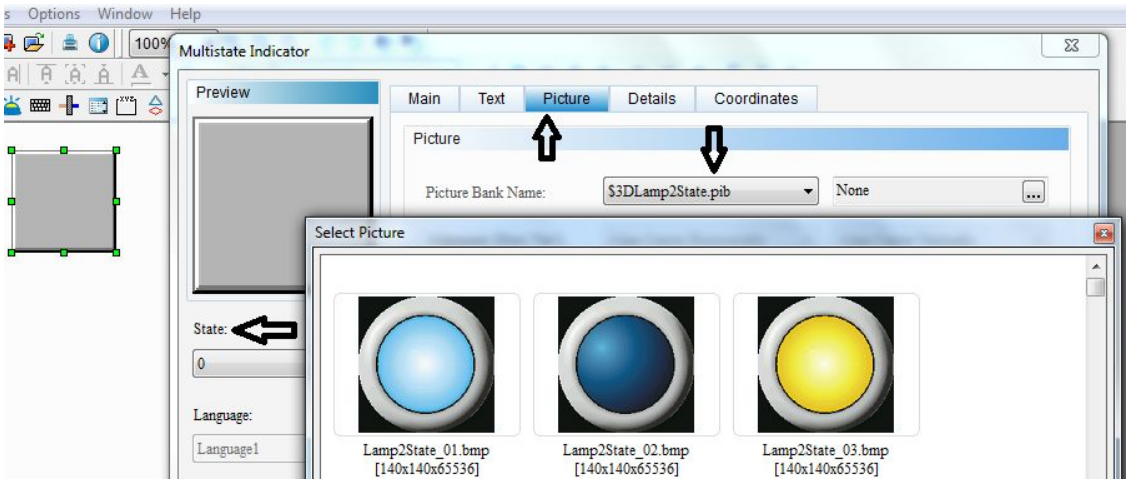
**Output:**

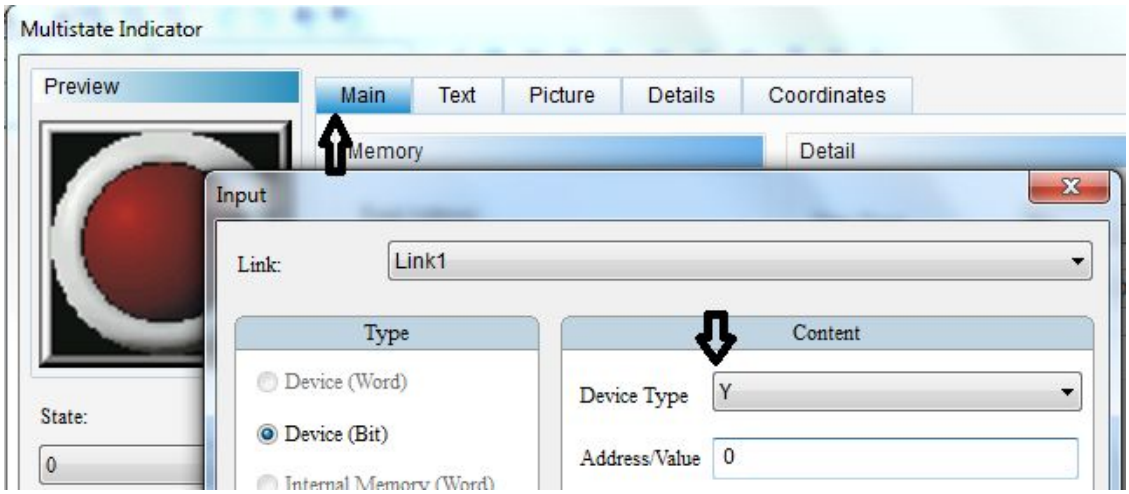



For output here option is indicator. It can be different name in another



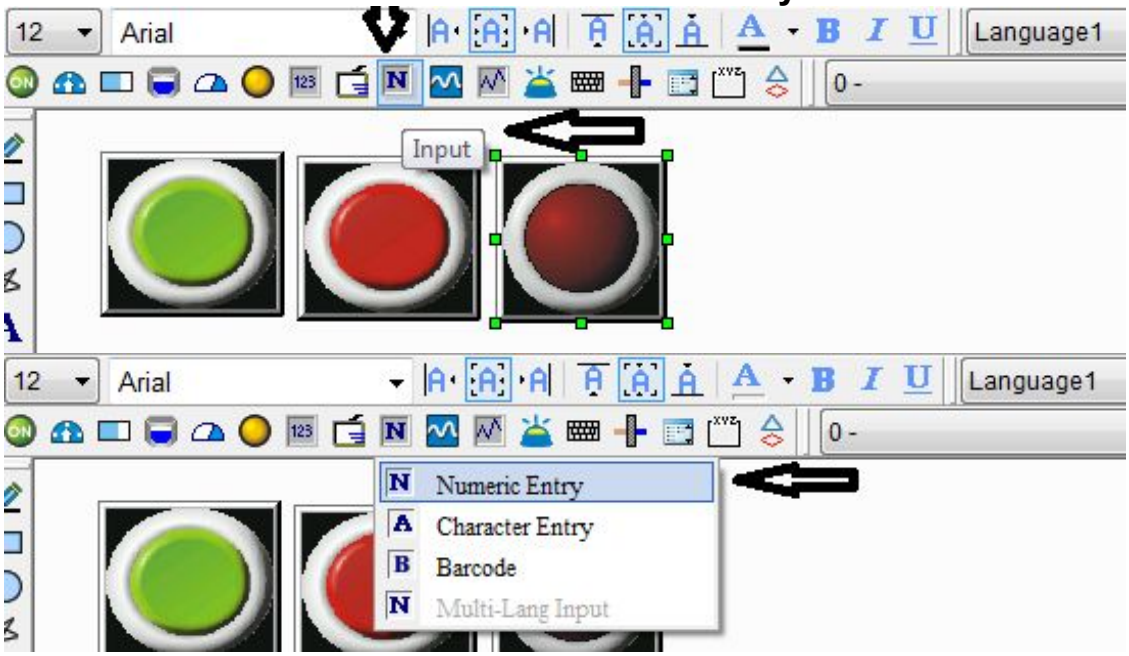

software.

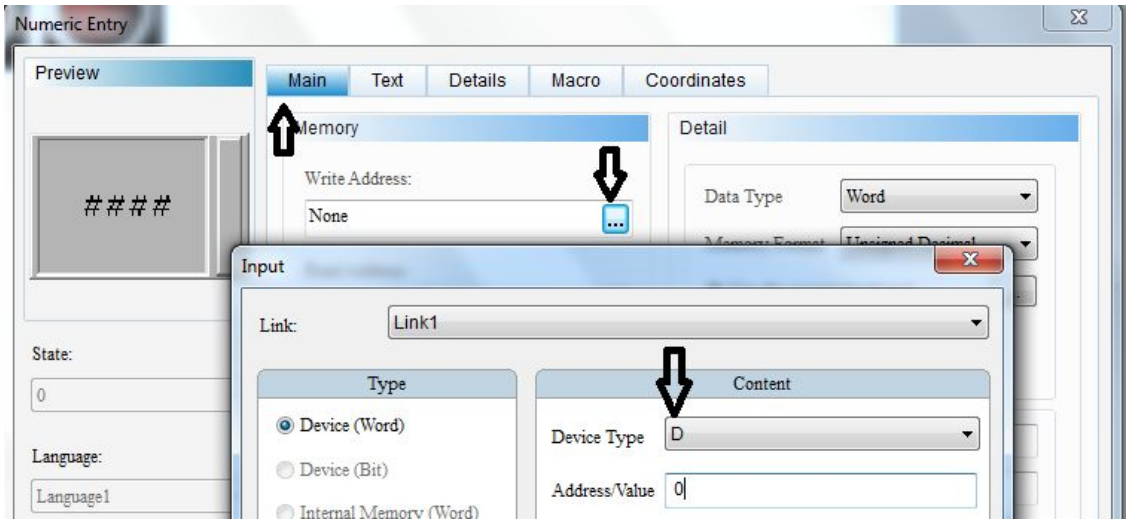




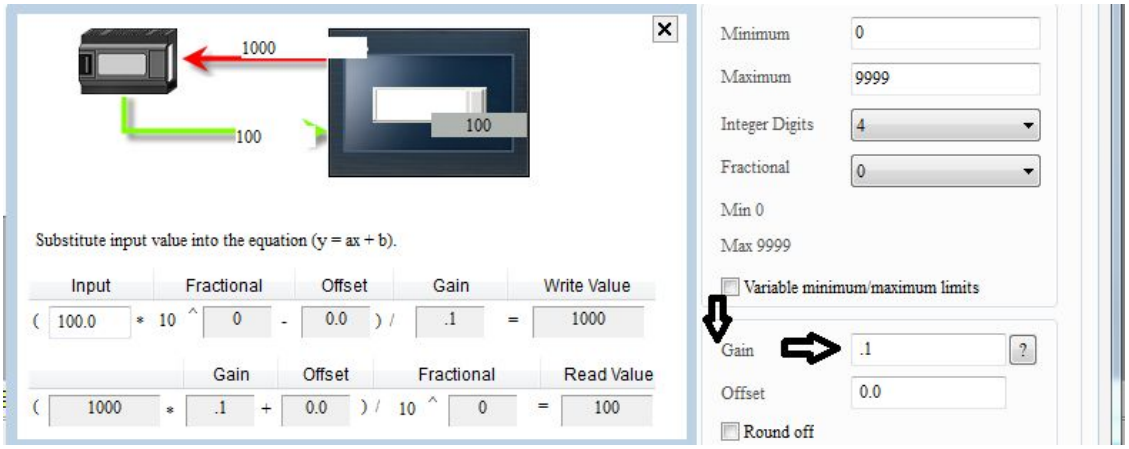

For output also we follow the same steps what we followed for inputs.

**Pre-set value/ numeric entry:**



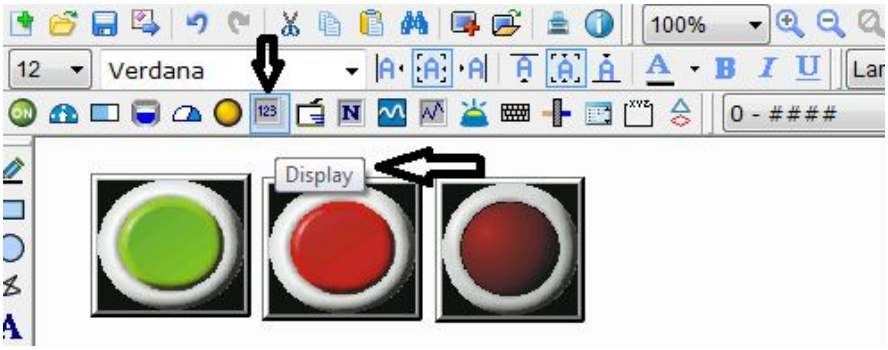


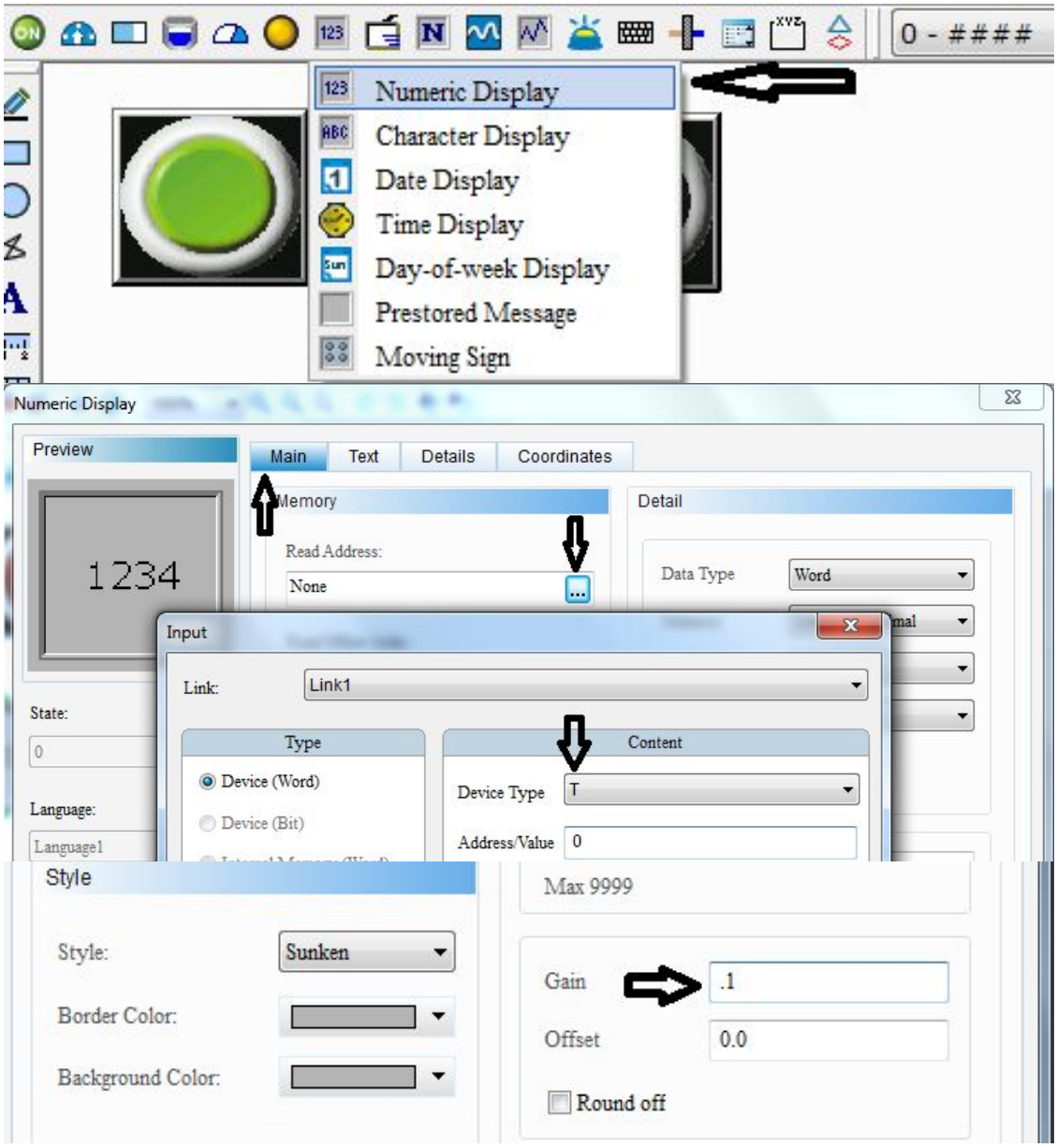

Option for Pre-set is INPUT  numeric entry. We need to just go to main and have to give address.




You can scale it also if needed. For scaling data, you will be given help window there.

**Accumulator/ numeric display:**

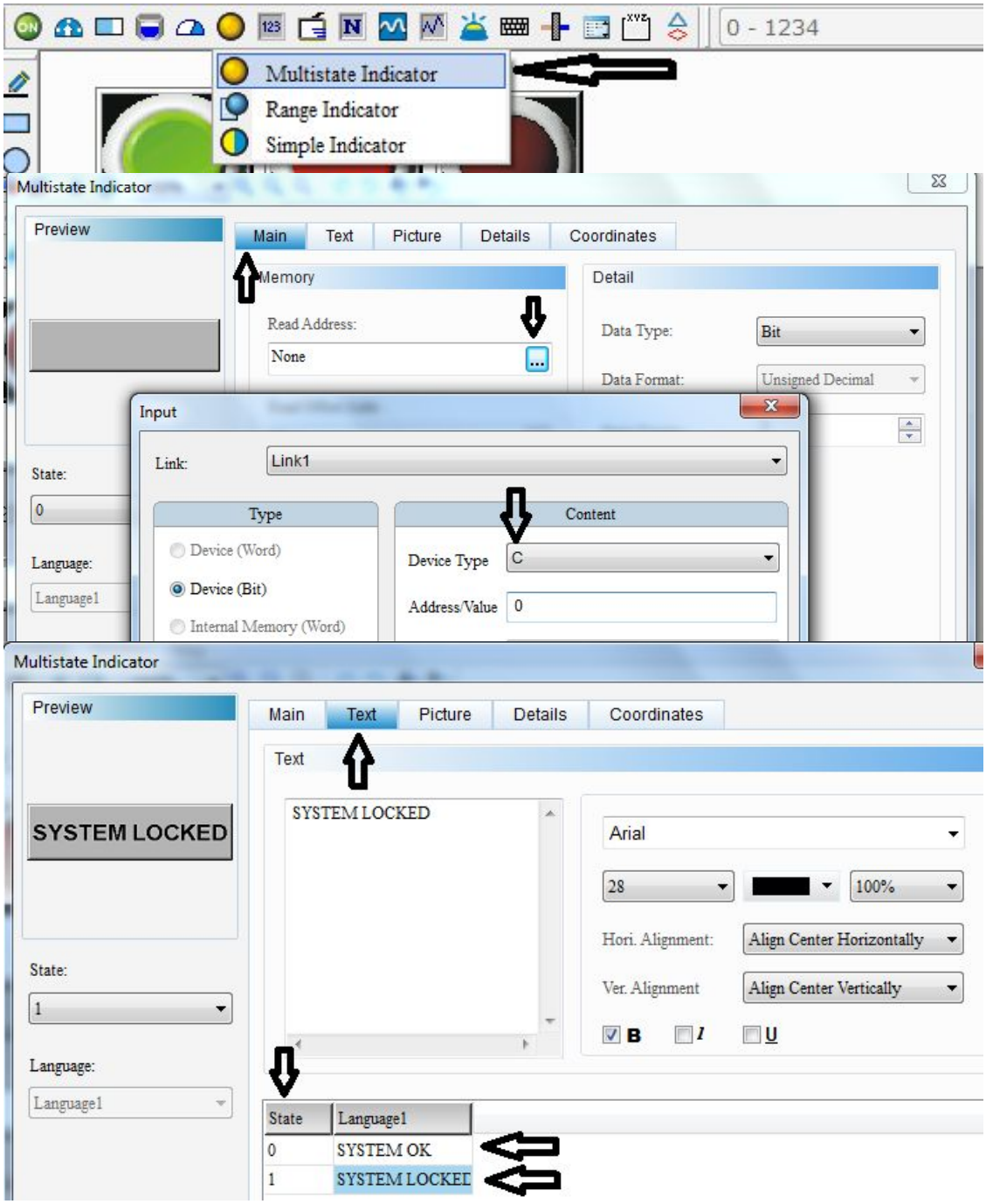


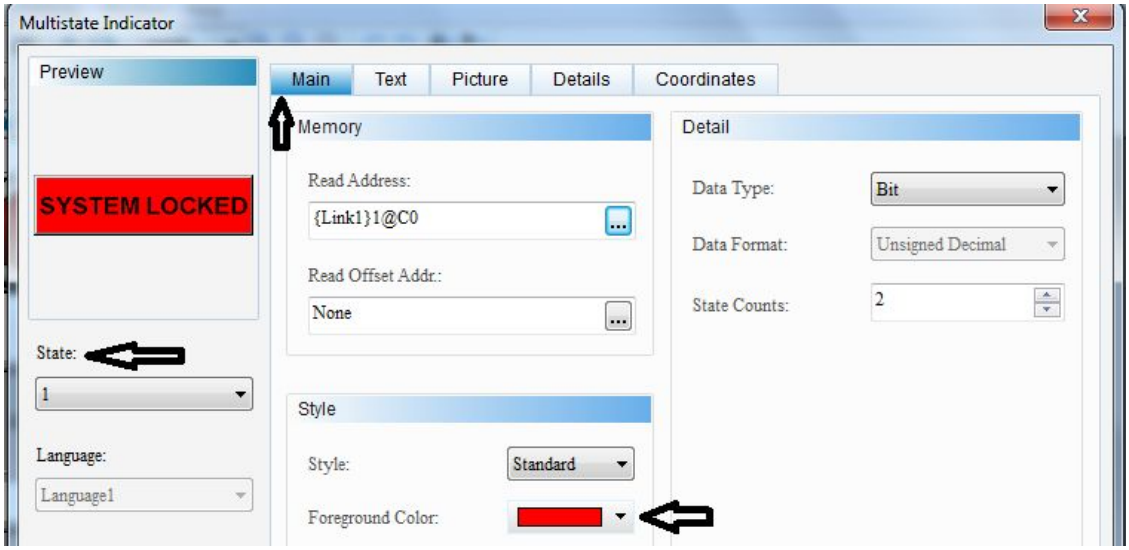



For accumulator we have Display  Numeric display. We have to follow the same steps that we followed for pre-set.

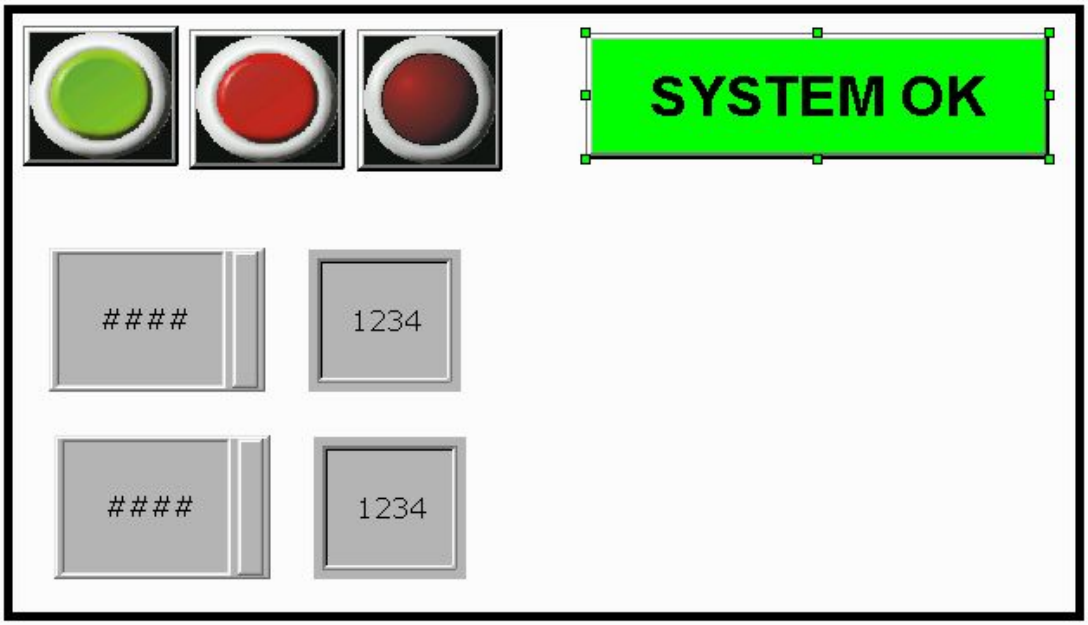
For counter also we follow the same steps as we followed for timer. For counter we do not need scale.

**Text display:** in some software you will be given option for text display. Here we shall use output option only for text display. Indicator  Multistate indicator.

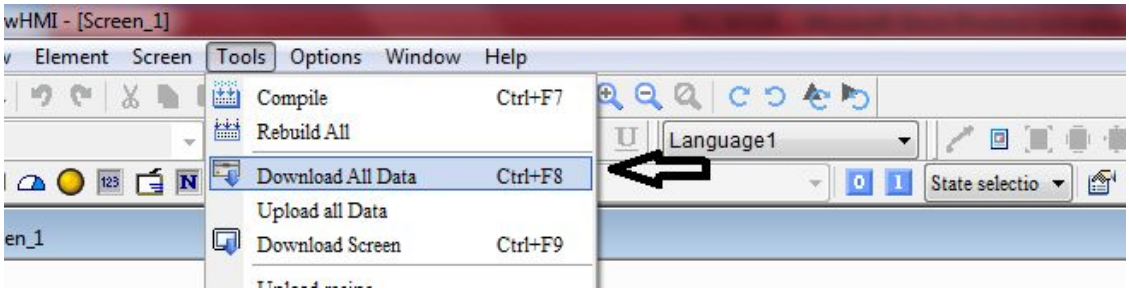

0 - 1234
Multistate Indicator
Range Indicator
Simple Indicator
Multistate Indicator
Preview
Main
Text
Picture
Details
Coordinates
Memory
Read Address:
None
Detail
Data Type:
Bit
Data Format:
Unsigned Decimal
Input
Link:
Link1
Type
Device (Word)
Device (Bit)
Internal Memory (Word)
Content
Device Type
C
Address/Value
0
State:
0
Language:
Language1
Multistate Indicator
Preview
SYSTEM LOCKED
Main
Text
Picture
Details
Coordinates
Text
SYSTEM LOCKED
Arial
28
100%
Hori. Alignment:
Align Center Horizontally
Ver. Alignment
Align Center Vertically
B
I
U
State:
1
Language:
Language1
State
Language1
0
SYSTEM OK
1
SYSTEM LOCKED




For sensor status display also, we follow the same.

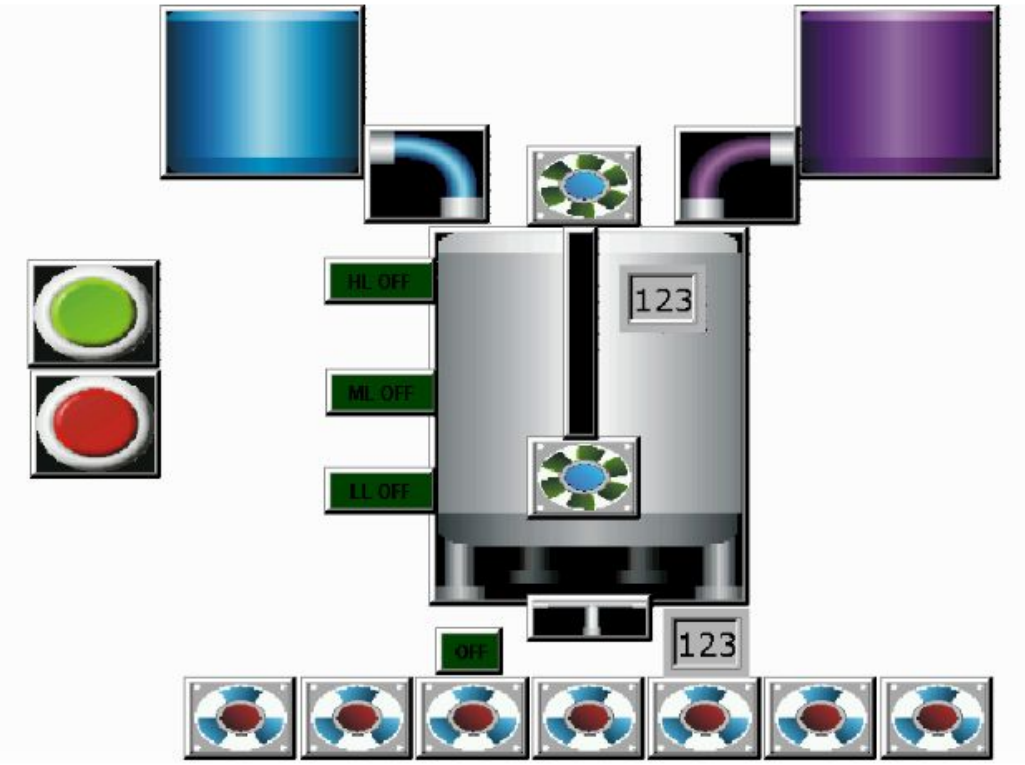


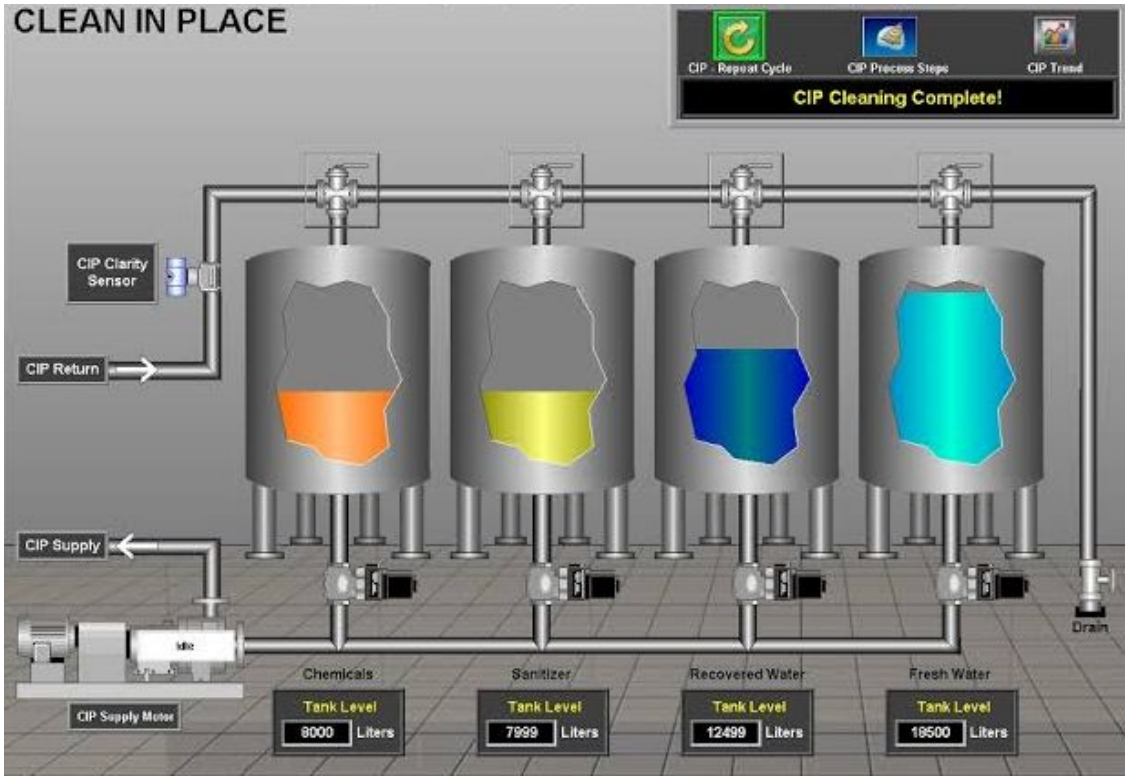

So finally, we get our screen ready.

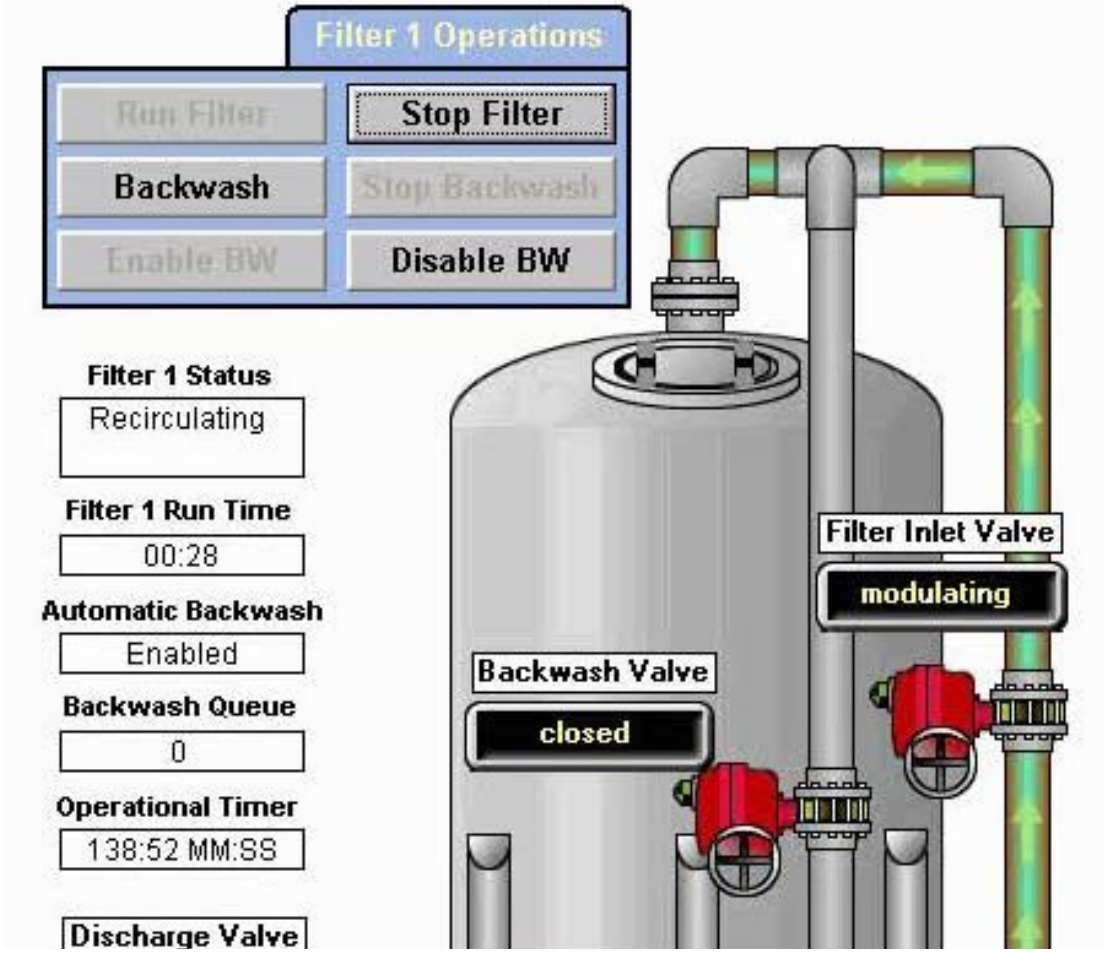


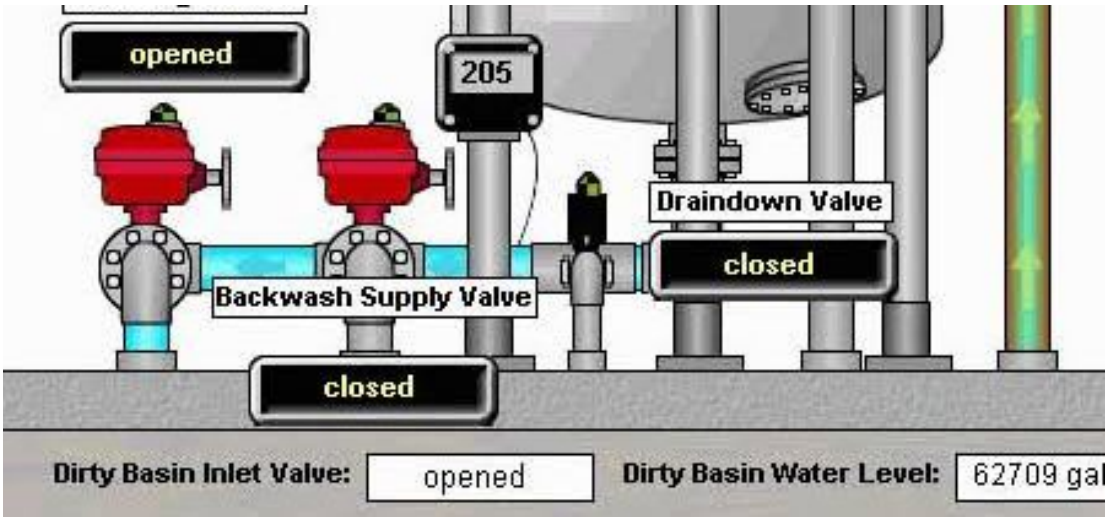

Now you can send this screen from software to hardware. In HMI we have a lot function. We can create many screens as we shall have large number of

data to be displayed for a good machine. We also have option of password protection to screen and data. When you create many screens then you can use ‘Go to Screen’ switch to go to one screen to another.  We have option for Macro and Recipe. There is a lot to do with HMI. It makes our machine/ system so flexible. I have given you the basic idea. For more detail you can go through user manual of HMI hardware and software. Generally, one PLC is connected with one HMI but number of PLC to be connected with one HMI depends on the number of communication ports on HMI. If you have 3 communication ports on HMI and you take new file in HMI software, if it shows you option of all three communication ports while selecting controller then you can connect three PLCs with that HMI. We can also receive existing design of HMI.




CLEAN IN PLACE
CIP - Repeat Cycle
CIP Process Steps
CIP Trend
CIP Cleaning Complete!
CIP Clarity Sensor
CIP Return
CIP Supply
CIP Supply Motor
Drain
Chemicals
Tank Level
9000
Liters
Sanitizer
Tank Level
7999
Liters
Recovered Water
Tank Level
12499
Liters
Fresh Water
Tank Level
19500
Liters


Filter 1 Operations
Run Filter
Stop Filter
Backwash
Stop Backwash
Enable BW
Disable BW
Filter 1 Status
Recirculating
Filter 1 Run Time
00:28
Automatic Backwash
Enabled
Backwash Queue
0
Operational Timer
138:52 MM:SS
Discharge Valve
Filter Inlet Valve
modulating
Backwash Valve
closed




**SCADA: Supervisory Control And DATA Acquisition.**

Basically, SCADA is also used to monitor and handle the data of controller/ PLC. In SCADA also we can design machines. We can monitor the status of all addresses used in our PLC program. From SCADA also we can change pre-set value. If we do the same thing with HMI and SCADA then what is the difference between these two?

1> HMI is hardware and SCADA is software that is installed in computer.
2> HMI has less data storage memory capacity and SCADA has a large number of data storage memory capacities.
3> HMI is used to monitor and control individual machine and SCADA is used to monitor and control whole industry/ all machines of industry.

SCADA is generally not used in small industries but it is used when industry is spread in large area. SCADA is used in industries like Cement plant, Power plant, Automobile, Oil & gas plant, Pharmaceutical plant, Chemical plant, Food processing plant etc. It is also used in smart cities now. In industry there is a control room with some computers and SCADA software installed in it. All machines and designed in this software using its tool and library. All machines/ PLCs are connected to the SCADA computer using higher communication protocol such as Ethernet, CC link, Modbus, Profinet, Melsecnet etc. depending on PLC brand. By sitting in one control room, whole industry process is monitored. If any problem occurs then error message is flashed on screen and it is rectified easily. Sometimes when it is not possible to place operator to operate machine because of extreme conditions like temperature, mud, water, radiation etc. then it is controlled from SCADA control room.

Now we shall learn how to develop SCADA screen for a machine. You need to remember few points to work on any SCADA software.

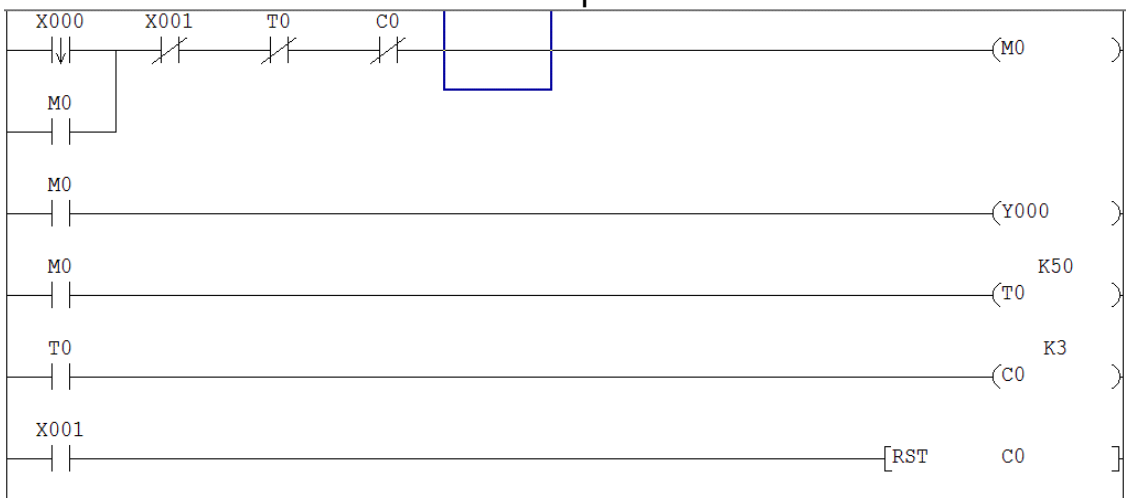
1> **Design/ Mimic:** Design depends on you. You can design it as per requirement using library and other tools but important part you need to search in design is Input, Output, Pre-set value, Accumulator/ Display and Text display. This is the same thing what we did in HMI design.

**2> Driver selection and configuration:** It is to select PLC hardware/ brand and its communication port setting. As we can connect many PLCs with SCADA PC so we need to mention that which brand and which model we are going to interface and through which communication port we are going to interface it with SCADA.

**3> Tag creation and configuration:** this is for addressing the designs. In HMI we have given direct address to each design but in SCADA we have no option to give direct address but for each address one tag needs to be created.

Mitsubishi has Ellipse SCADA software, Allen Bradley has Factory talk SCADA software and Siemens has Win CC SCADA software. There are many SCADA software developers. We can interface any brand of PLC with any brand of SCADA software. The requirement is; we must have that PLC driver in that SCADA software. If your PLC is not in that SCADA software then you can keep PLC driver file in your computer and can browse it.

Example:

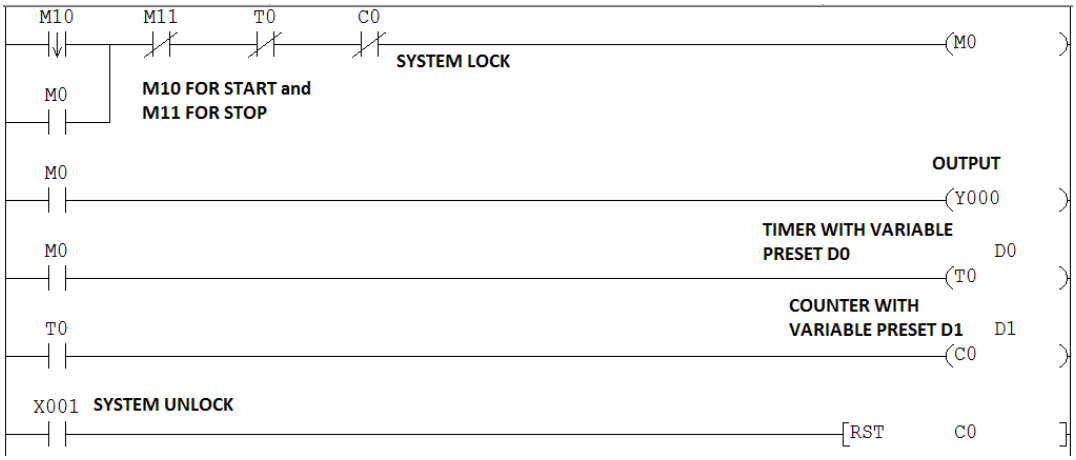



In this program when we press push button X0 then output Y0 gets on for 5 seconds. After 5 seconds output is off and counter counts one cycle. Like this every time when X0 is pressed, cycle runs and is counted. After 3 cycles the system gets locked by counter done bit c0. To unlock the system X1 is pressed. X1 is used for emergency stop too.

Let us decide now that what we have to do from SCADA.

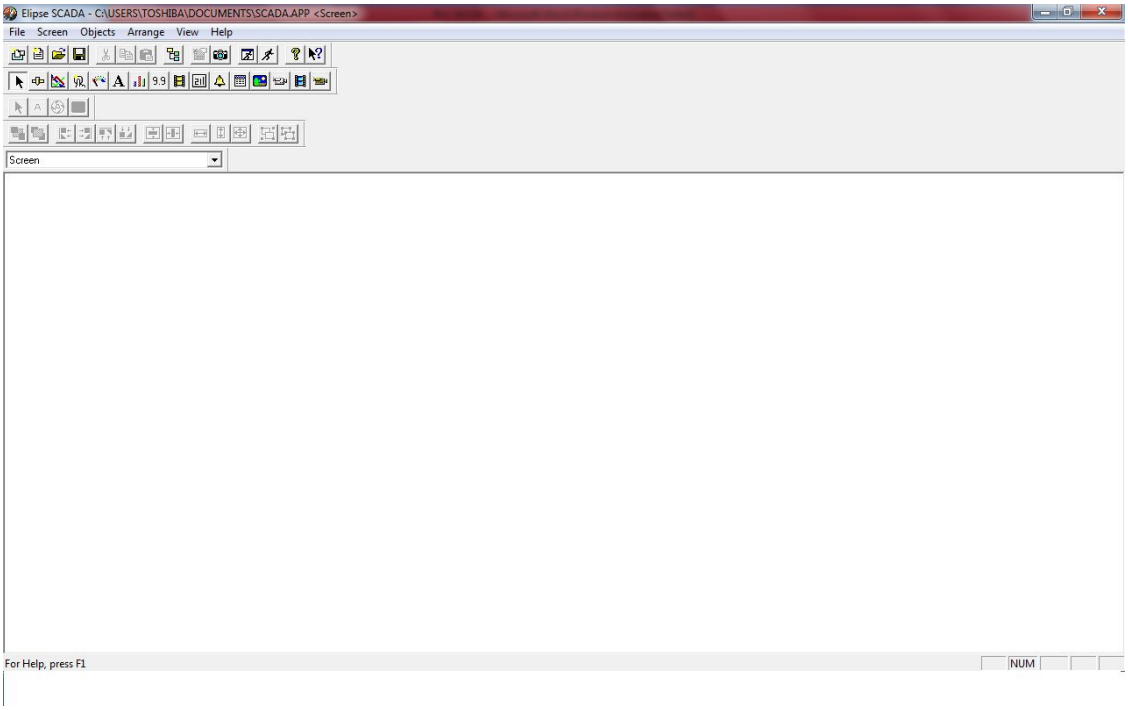
1. As we have SCADA so we shall not use physical button X0 and X1. Operate start and stop by HMI keys.
2. Display the status of output on SCADA.
3. Display the running/ present/ accumulator value of timer on SCADA.
4. Display the running/ present/ accumulator value of counter on SCADA.
5. Change pre-set value of timer from SCADA.
6. Change pre-set value of counter from SCADA.
7. Display SYSTEM LOCK on SCADA when system is locked.

In our program we shall replace x0 and X1 with any two flags. Let us take M10 and M11. To change pre-set value of timer we will have to replace K50 with variable address and in Mitsubishi it is D. let us take D0. For counter pre-set change from SCADA we shall replace K3 with D1. Let us modify program.

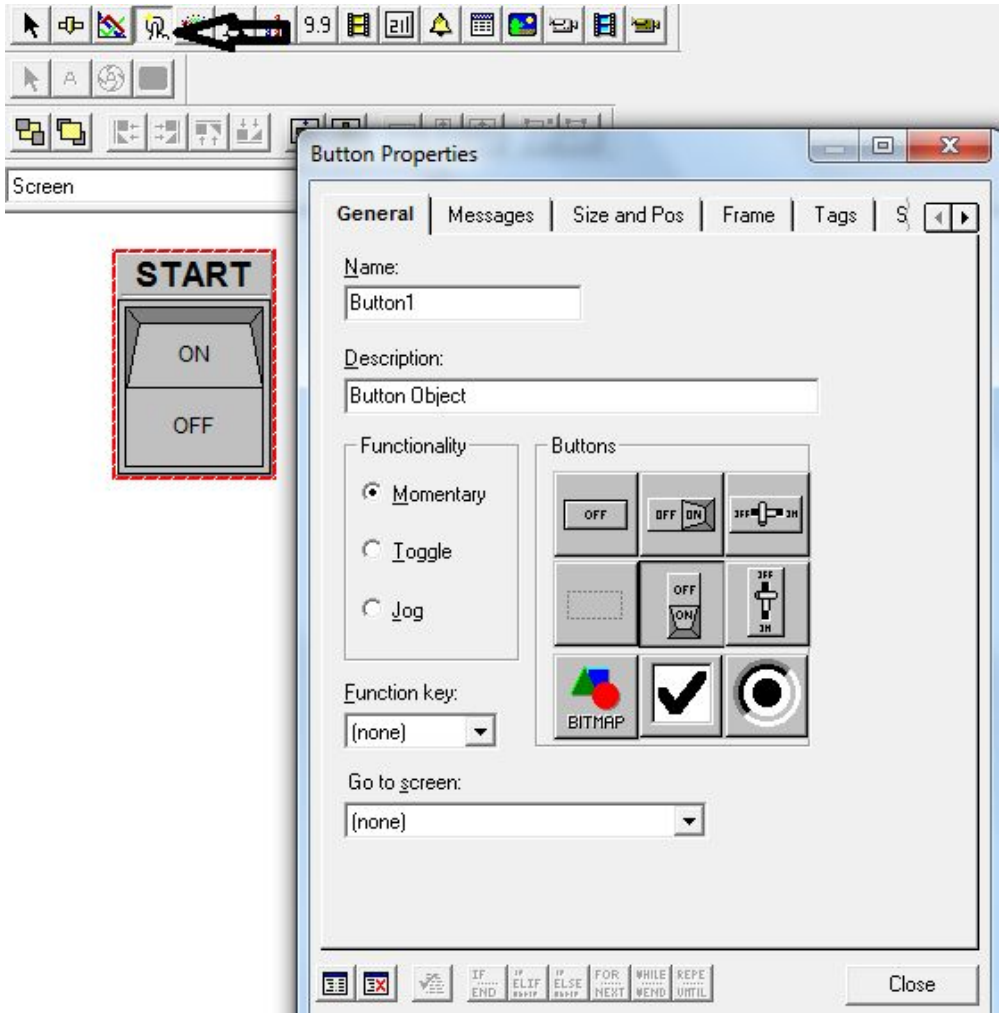



So we have modified program as per requirements. Now let us make a list of data to be configured in SCADA.

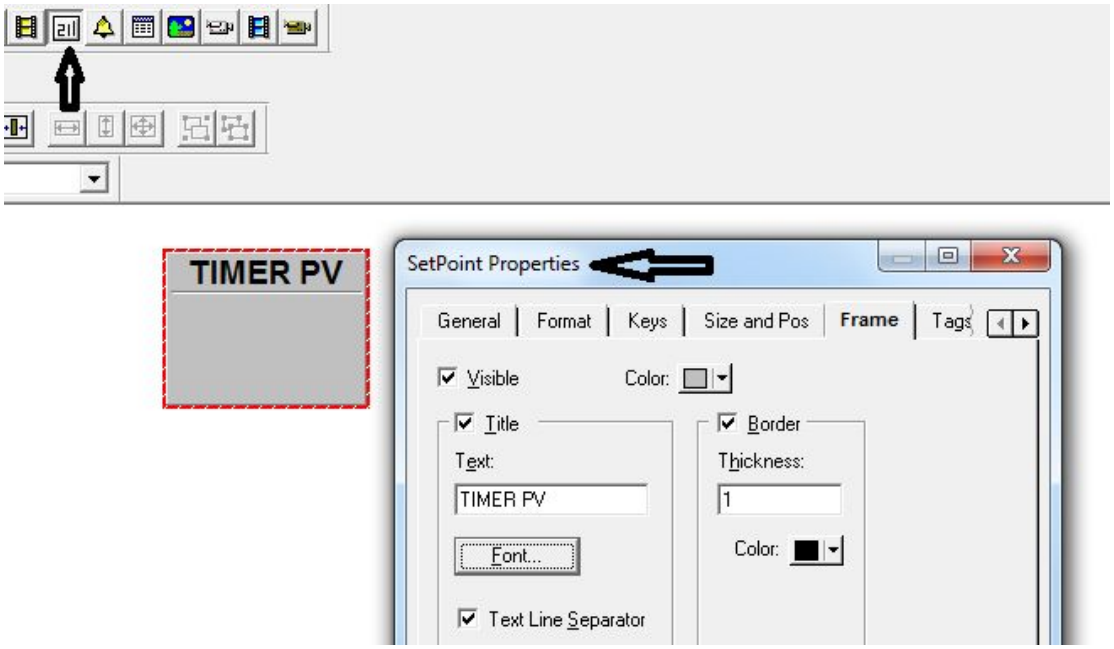
1. M10 □ START
2. M11 □ STOP
3. Y0 □ LAMP
4. T0 □ TIMER ACCUMULATOR/ RUNNING VALUE/ PRESENT VALUE
5. C0 □ COUNTER ACCUMULATOR/ RUNNING VALUE/ PRESENT VALUE
6. D0 □ TIMER PRESET VALUE
7. D1 □ COUNTER PRESET VALUE
8. C0 □ SYSTEM LOCK



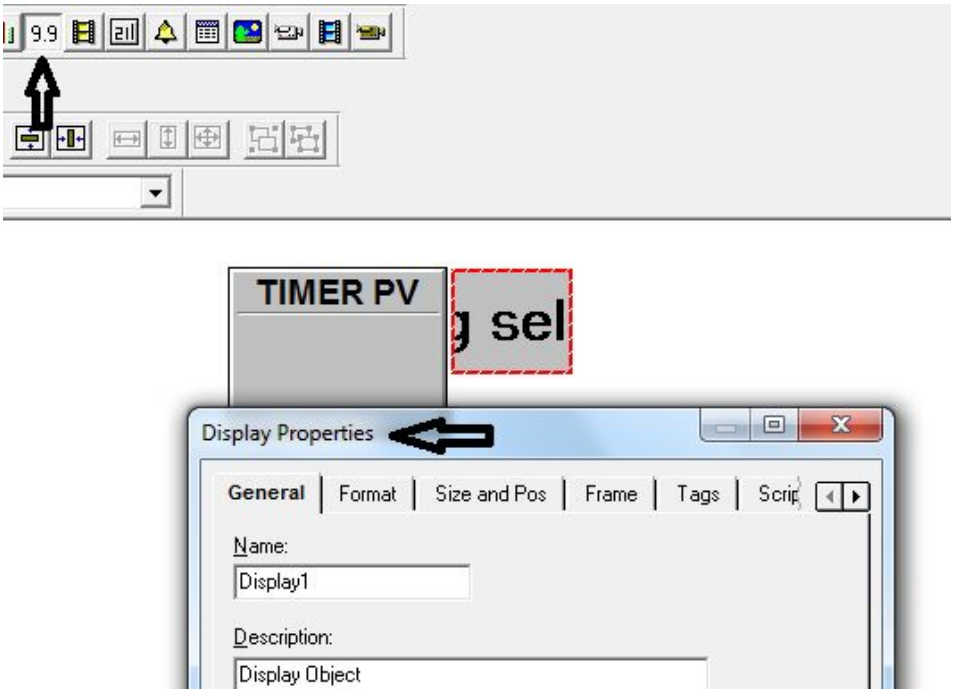

We take a new file in SCADA software and one screen is appeared.



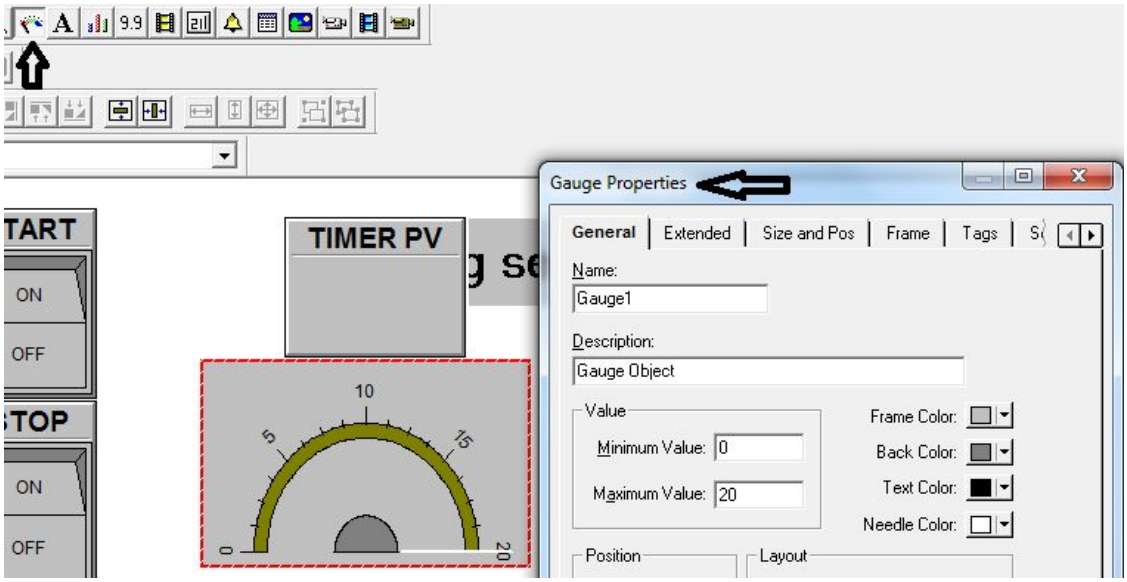

In this software it is Button for switch. We can select any design and switch type.



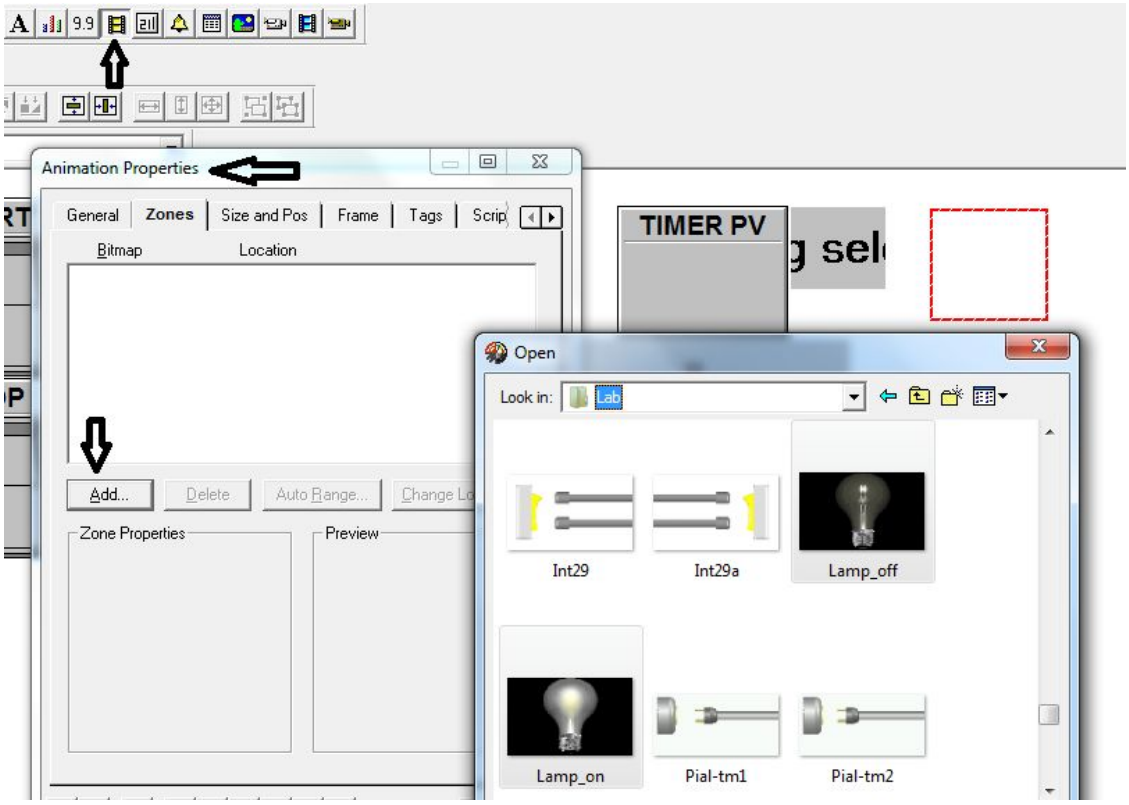

For pre-set value it is Set point here.



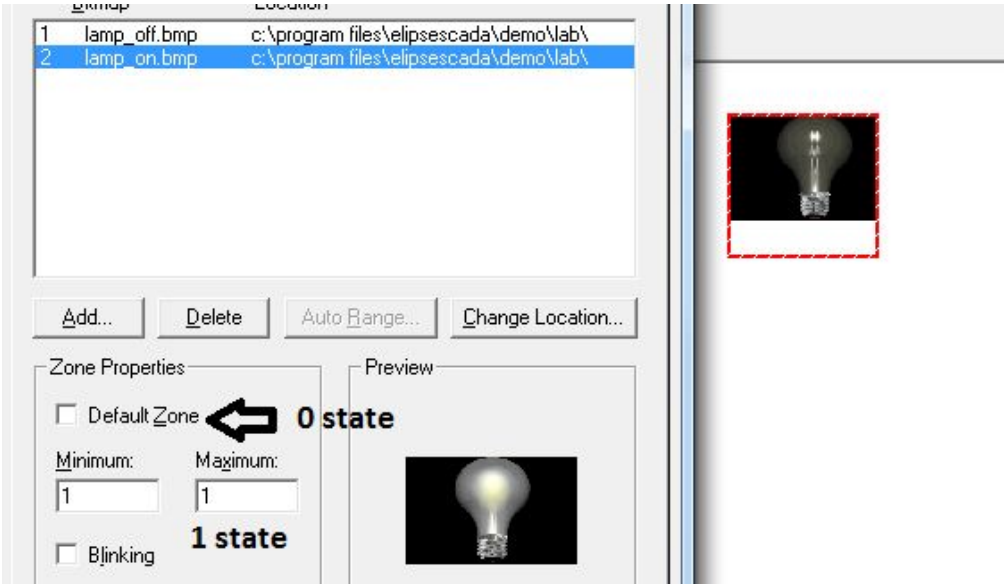

For accumulator it is Display.

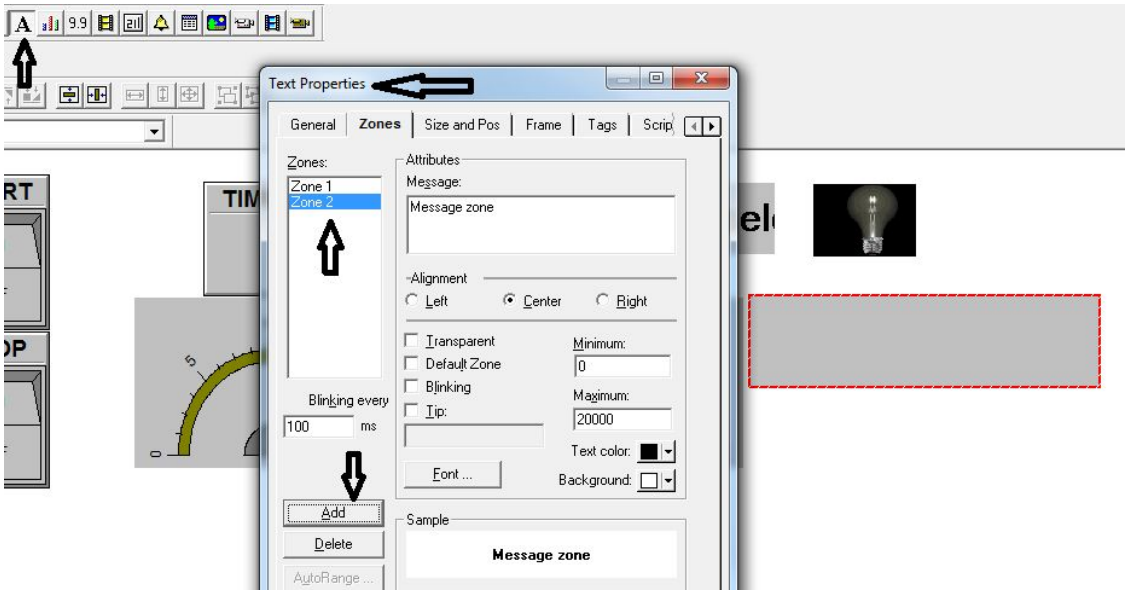



We can use meter also for accumulator.

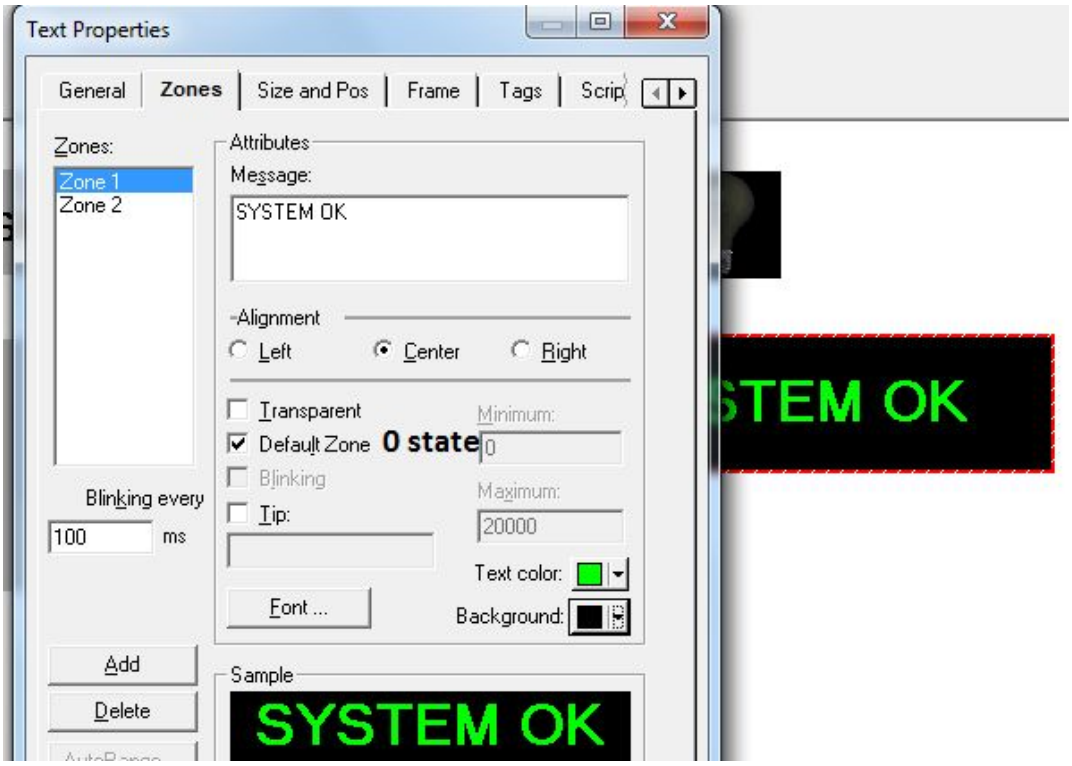



For output we have to take two pictures; one for off and another for on. Here we have to take option Animation. In animation we go to Zones and there we add two pictures from library. One picture will be 0 state and another will be 1 state.

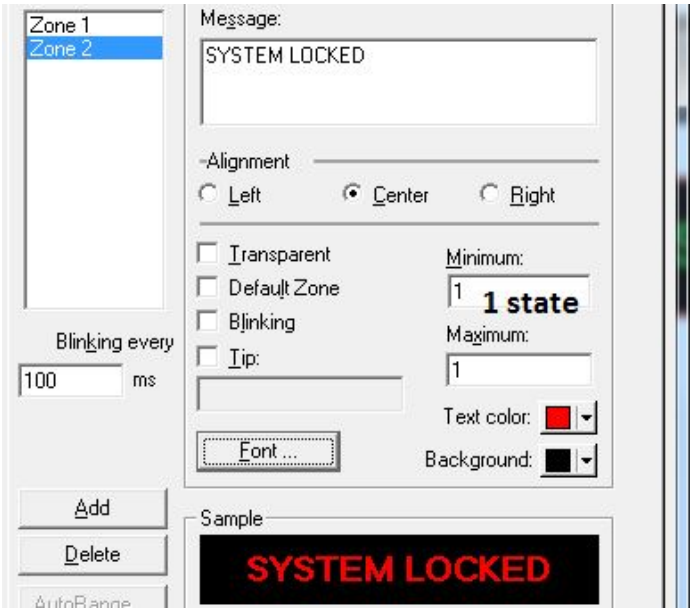


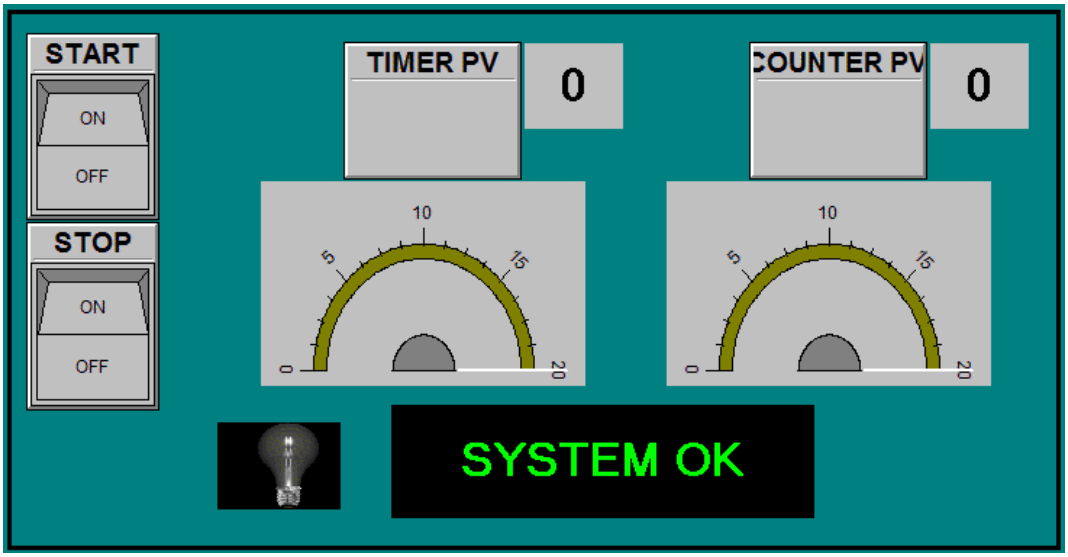

For message/ text display we have option Text. It is like output.



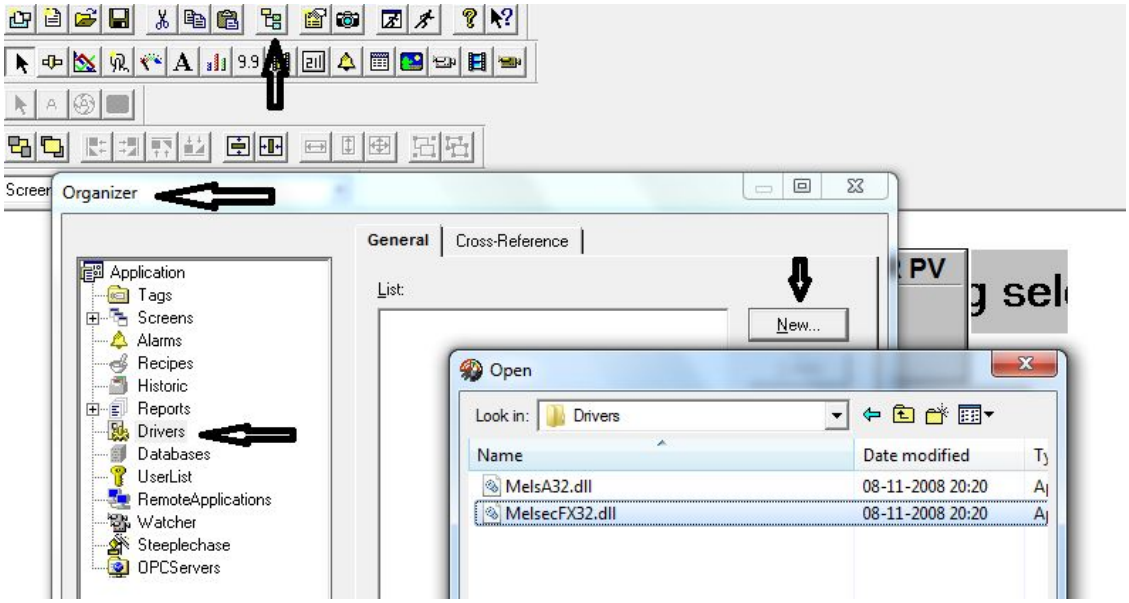

Here we add two zones; zone1 and zone2. We write message for both zones. Zone 1 we have taken default/ 0 state.



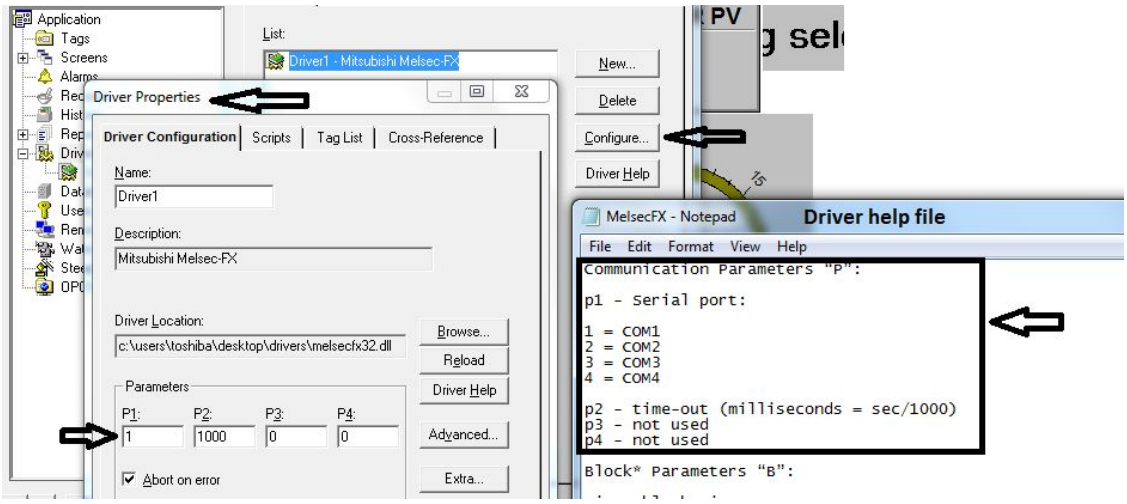

Zone 2 we have taken 1 state.



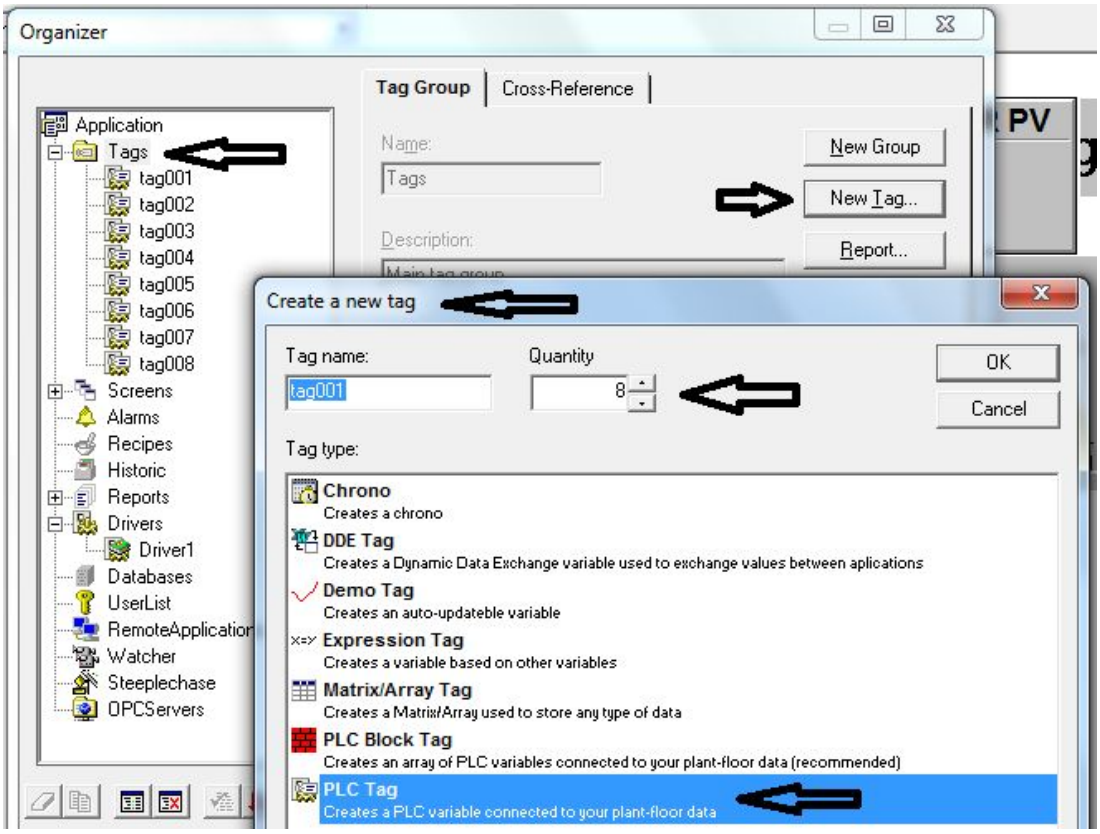

Yet we have gone through design only. Now we shall go for Driver/ communication and Tag/ addresses. Screen colour can be changed in screen property.




For Driver and Tag we need to go to File  Application Organizer. There we have to click Driver. In Driver we have to click New to select PLC driver.

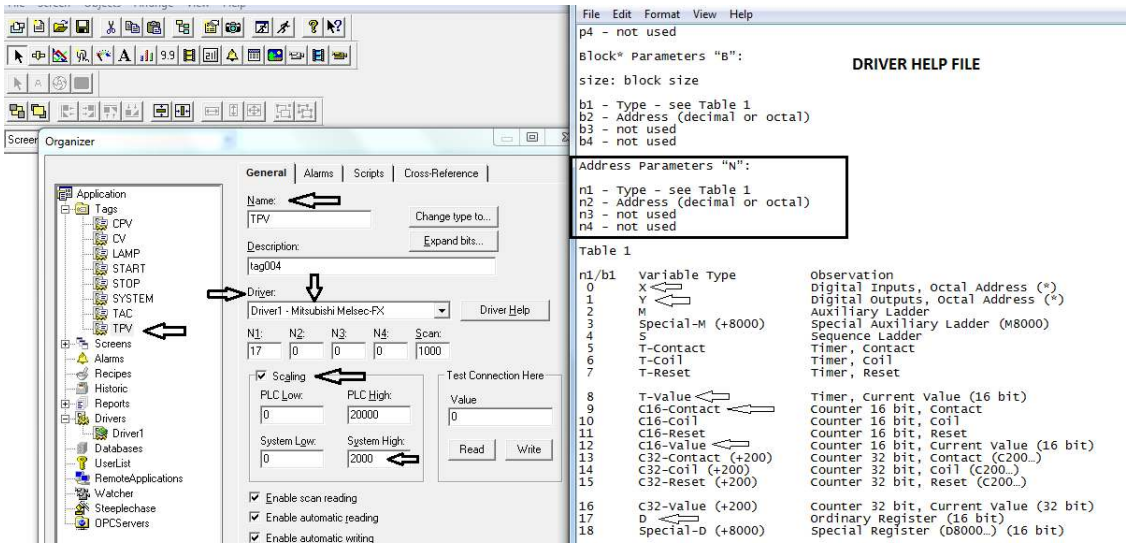



After selecting Driver, we need to configure it as per rules given in Driver help file. In configuration we have P1, P2, P3 and P4. We can see the help file. It is given that P1 is for communication port number. You have to check communication port number in device manager and have to fill in P1. P2 is for time-out that we have given 1000millisecond. Each driver will have its own rule so you can go through its help file.




Now we go for Tag □ New tag □ PLC tag □ Number of tag □ OK. Number of tag depends on the number of addresses in data list. We have 8

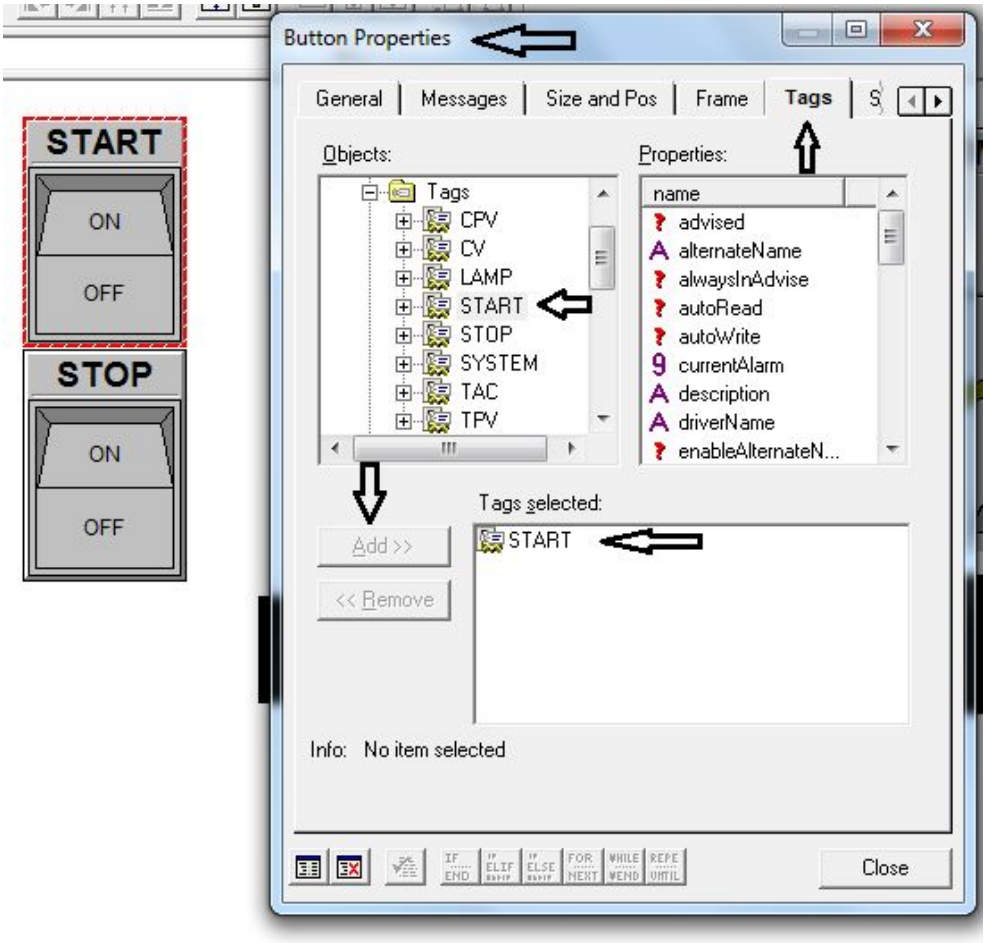
addresses. 8 tags have been selected by us from Tag1 to Tag8. Now we need to configure each.



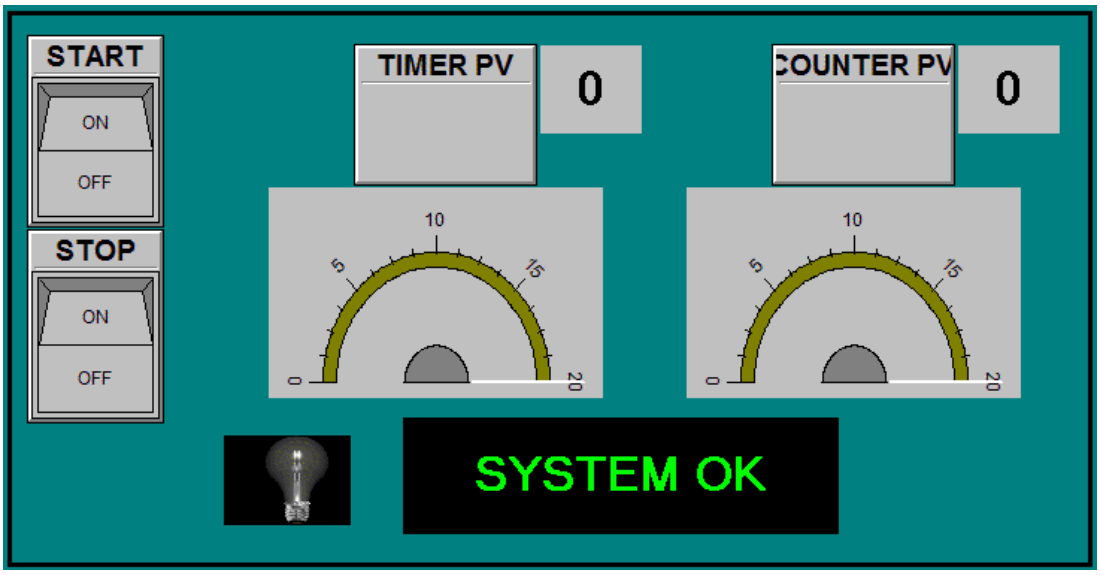

We go to each tag. We give a name to the tag. We select Driver and then it comes N1, N2, N3 and N4. Again, we go to Driver help file and we see that N1 is type and N2 is address. In X0 X is type and 0 is address. According to help file we configure each tag. If scaling is required then we scale it. We have scaled timer pre-set and accumulator.

| | N1 | N2 |
|---|---|---|
| 1. M10 → | 2 | 10 |
| 2. M11 → | 2 | 11 |
| 3. Y0 → | 1 | 0 |
| 4. T0 → | 8 | 0 |
| 5. C0 → | 12 | 0 |
| 6. D0 → | 17 | 0 |
| 7. D1 → | 17 | 1 |
| 8. C0 → | 9 | 0 |

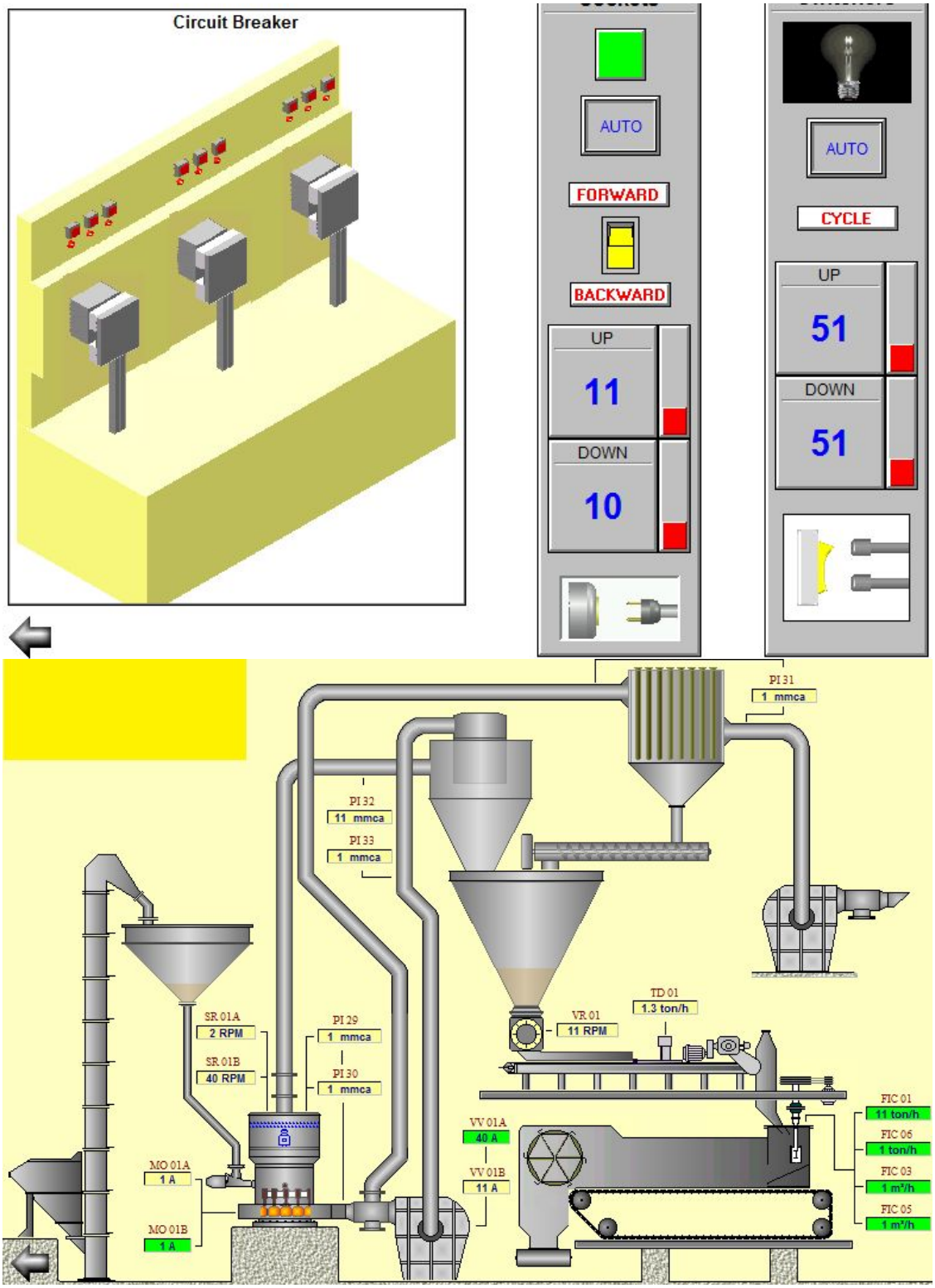
| n1/b1 | Variable Type |
|---|---|
| 0 | X |
| 1 | Y |
| 2 | M |
| 3 | Special-M (+8000) |
| 4 | S |
| 5 | T-Contact |
| 6 | T-Coil |
| 7 | T-Reset |
| 8 | T-Value |
| 9 | C16-Contact |
| 10 | C16-Coil |
| 11 | C16-Reset |
| 12 | C16-Value |
| 13 | C32-Contact (+200) |
| 14 | C32-Coil (+200) |
| 15 | C32-Reset (+200) |
| 16 | C32-Value (+200) |
| 17 | D |
| 18 | Special-D (+8000) |

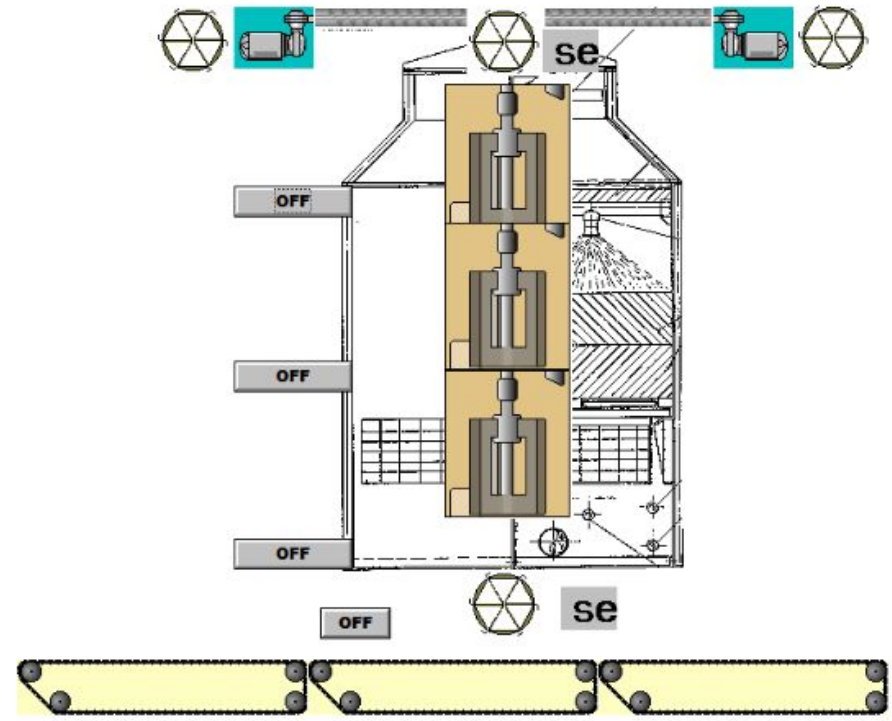


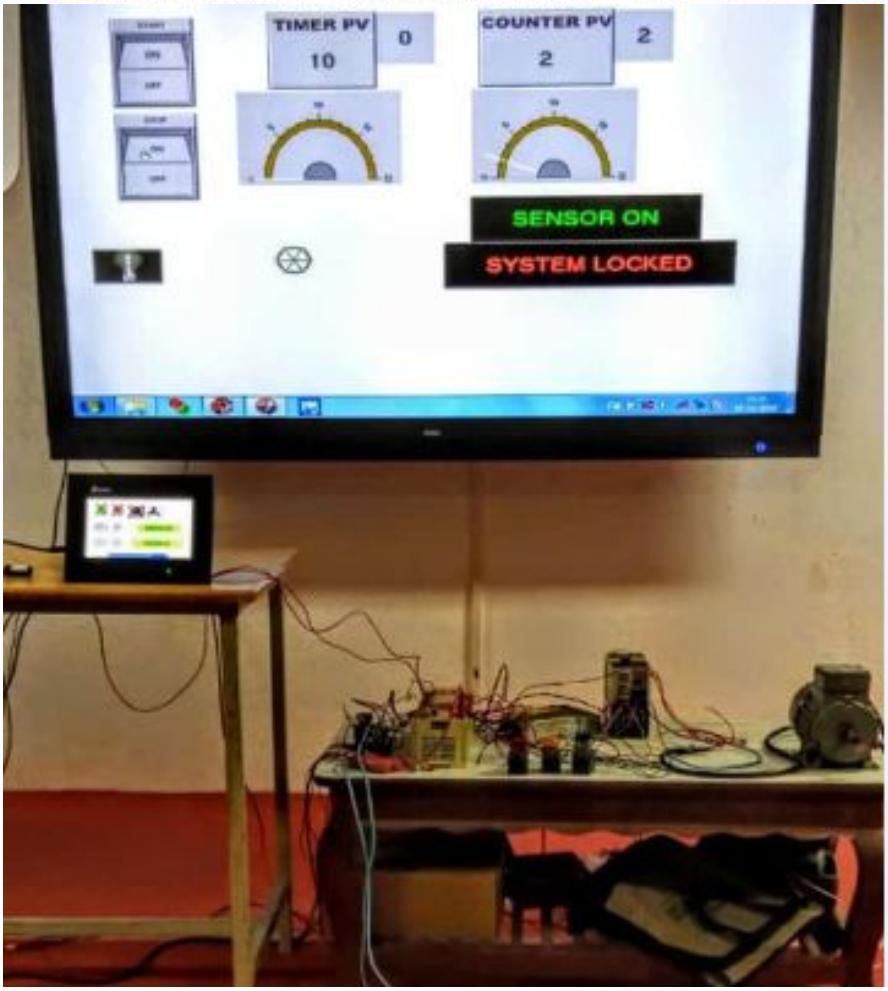

Now we double click each design and add tag.




Finally, screen is ready to function. We can create multiple screens. In SCADA there is a lot functions like Historic record, recipe, OPC server, data base, remote operations, Alarm, User password etc.

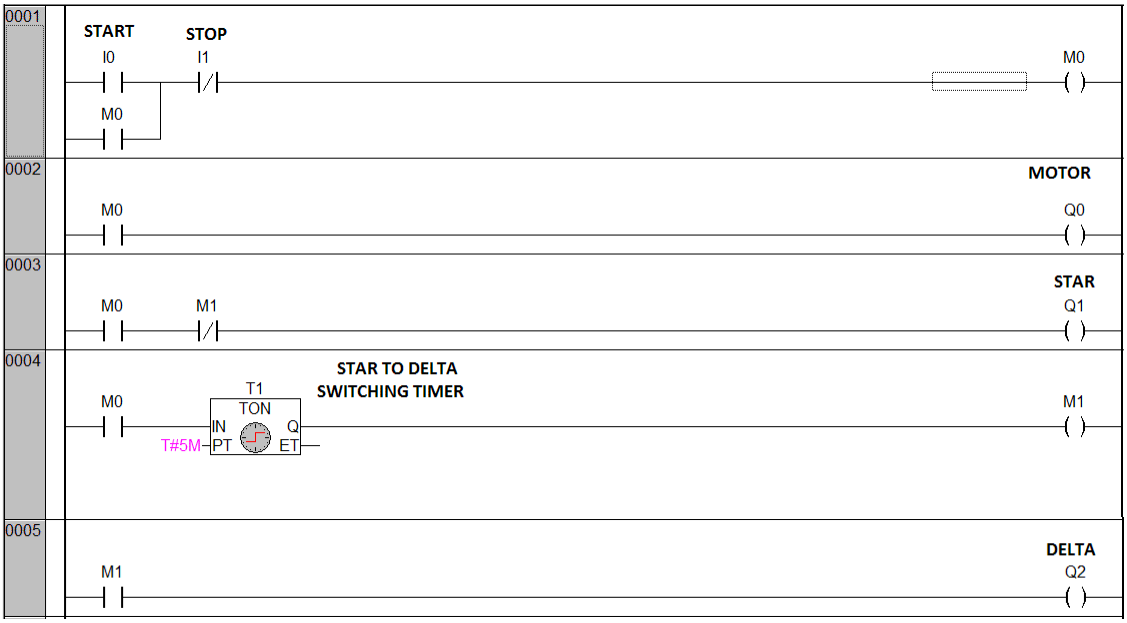

Circuit Breaker
AUTO
FORWARD
BACKWARD
UP
11
DOWN
10
AUTO
CYCLE
UP
51
DOWN
51
PI 31
1 mmca
PI 32
11 mmca
PI 33
1 mmca
SR 01A
2 RPM
SR 01B
40 RPM
PI 29
1 mmca
PI 30
1 mmca
TD 01
1.3 ton/h
VR 01
11 RPM
VV 01A
40 A
VV 01B
11 A
MO 01A
1 A
MO 01B
1 A
FIC 01
11 ton/h
FIC 06
1 ton/h
FIC 03
1 m³/h
FIC 05
1 m³/h


se
OFF
OFF
OFF
OFF
se


TIMER PV
10
0
COUNTER PV
2
2
SENSOR ON
SYSTEM LOCKED

**EX95: Star to Delta switching.**

I0 is start push button. I1 is stop push button. Q0 is main motor contactor. Q1 is Star connection contactor. Q2 is Delta connection contactor. When I0 is pressed, motor Q0 and Star Q1 will be on. After five minutes Star will be off and Delta Q2 will be on.




**AC DRIVE/ VFD (VARIABLE FREWUENCY DRIVE):** Motors are the main output in all kind of industries. More energy is consumed by this output only in industries. Most of time you will notice that motor runs on maximum speed/ frequency but load attached with it runs slowly as per requirement. Common example is a conveyor. Often conveyor belts or chain runs at

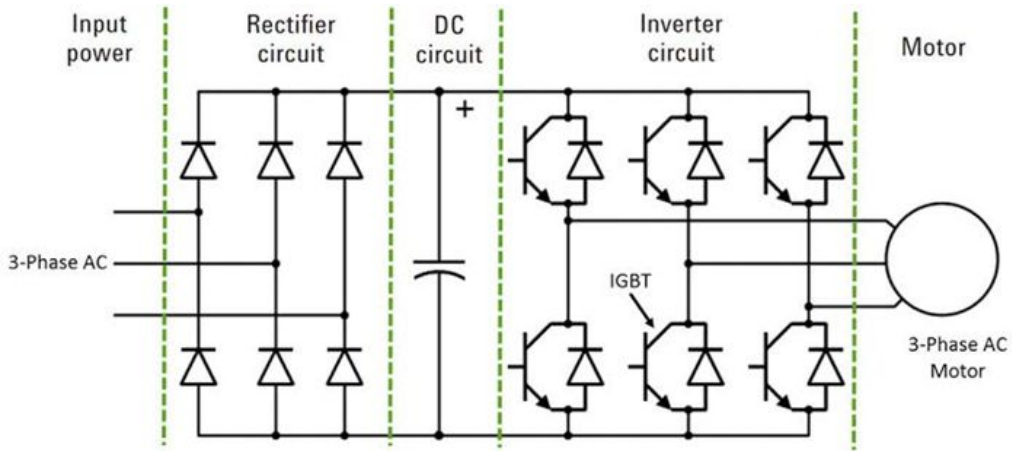
slower speed that motor speed. To reduce speed of load generally gears or some other mechanical arrangement is made. The requirement is to run load at low speed but motor runs at high speed i. e. it consumes maximum power which is not required. It is wastage of energy. Wastage of energy is wastage of money too. To reduce consumption of energy and to avoid wastage of money, motors are driven by some drives nowadays in industries. It saves a lot of energy and money.

Following methods are employed for speed control of induction motors:

1> Pole changing.
2> Stator voltage control.
3> Eddy current coupling.
4> Rotor resistance control.
5> Slip power recovery.
6> Supply frequency control.

We are going to study Supply frequency control method.  VFD is a motor controller that drives an electric motor by varying the frequency. By varying frequency, we can increase or decrease the speed as per requirement of load. VFD operates under the principle that the synchronous speed of an AC motor is determined by the frequency of the AC supply and the number of poles in the stator winding, according to the relation RPM= 120*f/p. Voltage induced in stator is proportional to the product of supply frequency and area of flux. If stator drop is neglected, terminal voltage can be considered proportional to the product of frequency and flux. Any reduction in supply frequency without change in terminal voltage causes an increase air gap flux. The increase in flux will saturate the motor. While an increase in flux beyond rated value is undesirable from the consideration of saturation effects. A decrease in flux is avoided to rotation torque capability of the motor. Therefore, the variable frequency control below the rated frequency is generally carried out at rated air gap by varying the terminal voltage with the frequency so as to maintain V/F ratio constant at rated value.

VFD has three sections; Converter (AC to DC), DC filter and Inverter (DC to AC).




Converter first converts AC to DC but not smooth DC. To smoothen it filter capacitor is used. Inverter section contains different types of electronic switches i. e. IGBT and MOSFETs. The switches are rapidly turned on and off resulting pulsating voltage i. e. very similar to AC. output frequency is proportional to the switching rate. High switching  high output frequency.

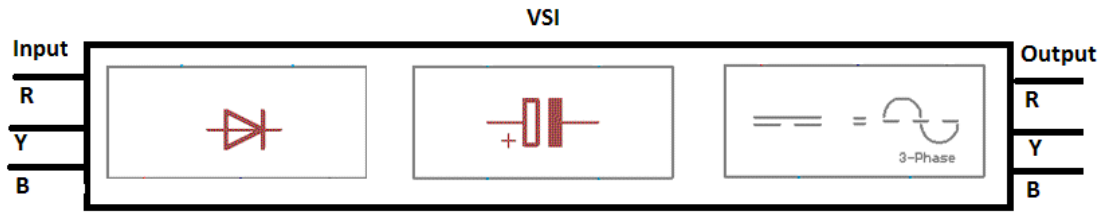
POWER RATING:

VFD (VARIABLE FREWUENCY DRIVE) designed to operate at 110 V to 690 V are often classified as low voltage unit and typically designed for use of motors rated to 0.2 KW/ ¼ HP to at least 750 KW/ 1000 HP. Medium voltage unit are designed to operate at 2400-4160V (60 Hz)/ 3300V (50 Hz) to 10 KV. In some applications step up transformer is placed between a low voltage drive and medium voltage load. Motor rate is 375 KW/ 1500 HP and above. Above 7 KV/ 1500- 10000 HP is one-of-a-kind design. There is a minimum of six rectifiers for a three phase AC VFD. It can be more. Manufactures offer 12, 18, 24 and 30 pulse drives. Standard six pulse drive has six rectifiers. A 12-pulse drive has two sets of six rectifiers. If power connected to each set of rectifiers is phase shifted then some of the harmonics is produced by one set of rectifiers will be opposite of the other set of rectifiers. The two wave forms effectively cancel each other out. In order to use phase shifting, a special transformer with multiple secondary windings must be used. For example, with a 12 pulse VFD a DELTA/ DELTA Wye transformer with each of the secondary phase shifted by 30* should be used.

TYPES of VFD: there are many types of VFD but main three types of VFD are VSI, CSI and PWM.

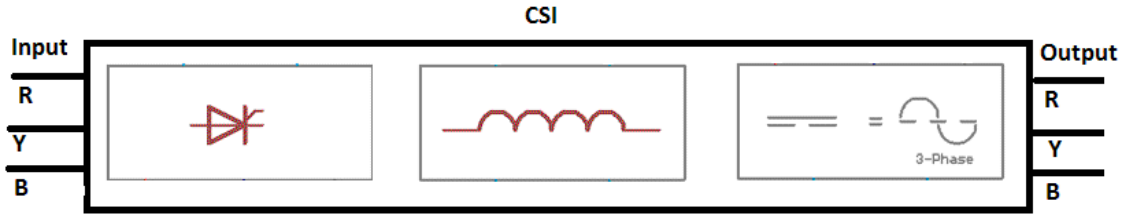
**VSI (Voltage Source Inverter) VFD:** it is common type of VFD. It has simple diode bridge to convert AC to DC and a capacitor is used to smoothen DC

and to store energy. Inverter circuit uses this stored energy and provides output.




It has a good speed range. Multiple motors can be connected with single VFD. It has simple design and is cost effective. Due to cogging effect the load motor faces jerking during start and stop. The output provides different types of harmonics and causes noise. Decrease in speed causes poor power factor.

**CSI (Current Source Inverter) VFD:** It depends on current. In CSI, SCR bridge rectifier is used instead of diode bridge rectifier. Its filter is inductor to smoothen current output. It is like current generator. Instead of square wave voltage, CSI provides square wave current.

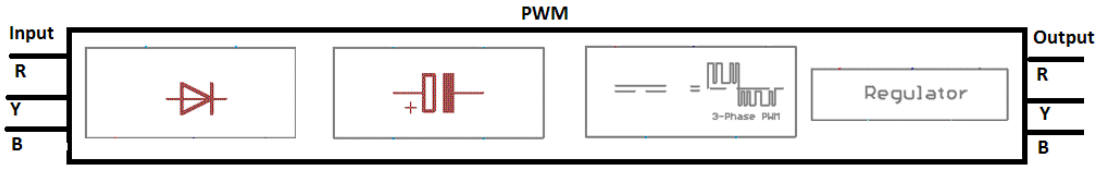



It is reliable than VSI. It supports higher horse power induction motors where VSI is not a suitable choice. Simple design and it has good generation capability. It has overall poor power factor at low RPM. Due to cogging effect shaft vibrates in running condition. It is not suitable for multiple motor connections.

**PWM (Pulse Width Modulation) VFD:** Using PWM technique VFDs are capable of providing stable voltage output maintained with a frequency ratio. It uses Diode Bridge to rectify AC to DC. The switching circuit controls the duty cycle in a variable frequency range. An additional regulator is used to regulate the PWM output to provide stable and proper voltage to the load.

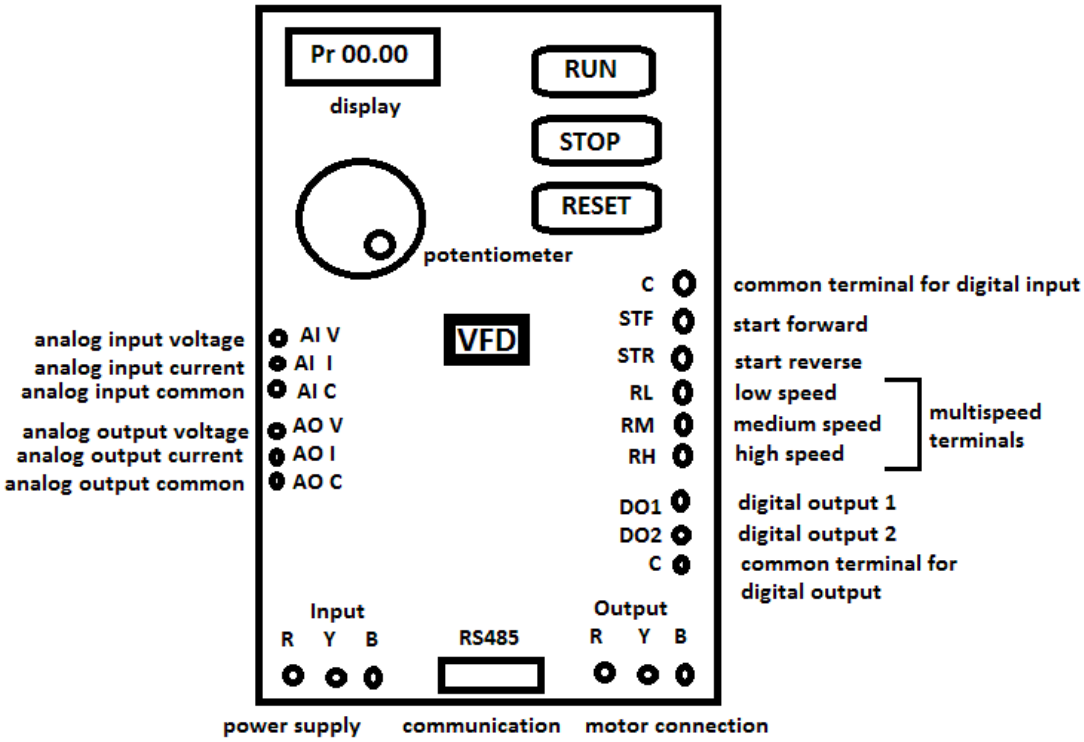
It has no clogging and jerking effect. It has wide speed and control range. It has different types of protection circuits. It has constant power factor. It

induces very high efficiency. It is energy efficient. It is complex in design and in implementation. It requires additional hardware.




When you run a motor at high speed (50 Hz/ 60 Hz), you do not have any saving but any reduction pays the reduction cubed. If you run motor at 10 % slower speed using VFD then you save 27%. How? If you cube 90% or .9by multiplying (.9 X .9 X.9) you will get .729. So now you are using only 72.9 % of the total energy.

Any VFD will have some common things on it. It has one screen to monitor running status and to monitor parameter modifications. It has a nob or key pad to set required values. It has run, stop and reset key on it. VFD will have some digital and analog input output terminal too. It will have input power terminal for its power on and output power terminals to connect motor. VFDs are available with single phase to three phase output and three phases to three phase output too. If its power supply is 230 V AC then then its output power will be 230 V AC only but three phases. VFD output is always three phases. Generally, motor is connected in Delta connection with VFD. VFD size increases with the increase in its HP capacity. VFD has some universal digital terminals on it. VFD can control all parameters itself but in industry it is connected in sequence controlled by PLC so PLC sends signal to VFD to operate motors. For external control signal VFD has universal terminals and communication ports.



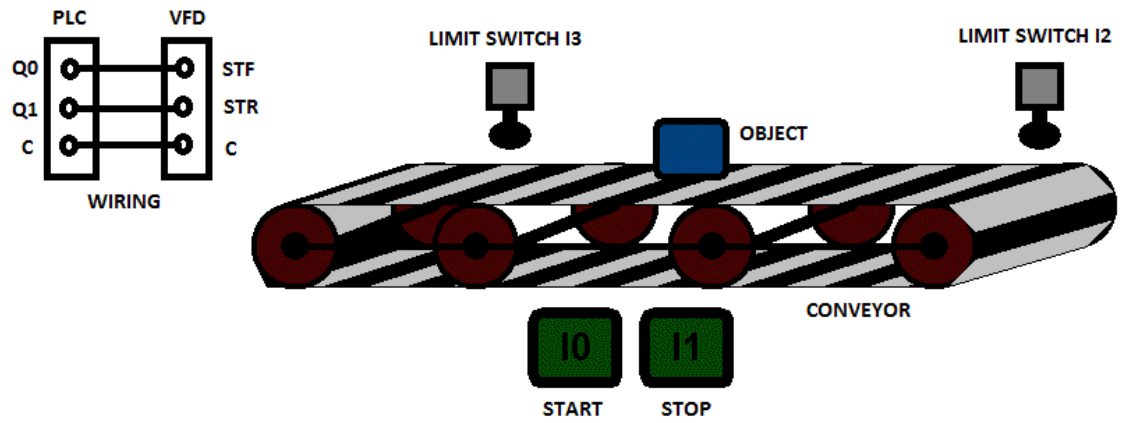

STF is to run motor in forward direction. STR is to run in reverse direction. RL, RM and RH are multispeed terminals. By varying these three terminals signal we can run motor in different 7 speeds as per setting in VFD. Different drive will give different number of multispeed. If we have 4 multispeed terminals then 15 different speeds can be selected and run in once only. VFD will have few digital output terminals too. These terminals are usually used to get feedback of drive like over load, maximum speed reach signal or some error etc. feature of digital input output terminals can be changed by changing setting in VFD. One cannot work without user manual as different model of VFD will have different rules and memory number. In same brand too different model will have different rule. VFD has analog terminal. It accepts analog input (0- 10 V/ 4- 20 ma) and it generates also analog output (0- 10 V/ 4- 20 ma). Suppose there is an application that when temperature increases then speed of motor should increase. As temperature will vary, temperature transmitter will send 0 to 10 V variations to VFD analog input terminal. Some settings will be done in VFD as per given in user manual. In the same way for analog output consider there is a control valve operated on 0 to 10 V. as speed of motor increases then opening og control valve should increase. In this case as motor speed will vary then o to 10 V variation will be sent to control valve from VFD analog output terminal. VFD can control any parameter of motor control. Some common controls are as given below:

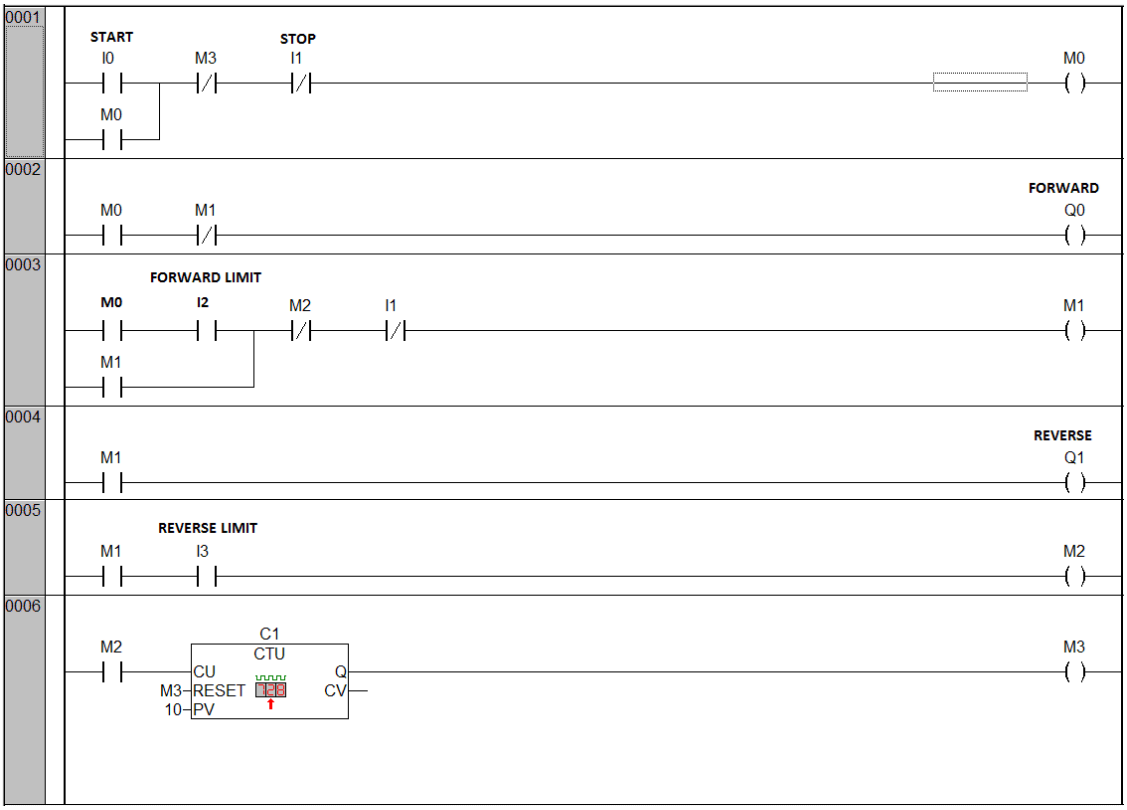
- Speed/ frequency control.
- Direction control.
- Acceleration time control.
- Deceleration time control.
- Multispeed control.

In acceleration time control we can set the time taken to reach maximum speed from zero speed. In deceleration we can set the time taken to reach zero speed from maximum speed. If we set 15 seconds deceleration time then it will take exact 15 seconds to stop the shaft. If it is set 1 second then in 1 second only shaft will be chocked. In multispeed control all 7/ 8/ 15 speed can be set in VFD parameters as per user manual. As start will be pressed, motor will run in different speed, direction and duration as per multispeed setting in VFD. VFD can run in internal mode and external mode. In internal mode no signal is given from external devices. In external mode signal is given from external devices. Mode is changed by changing setting in VFD as per user manual.

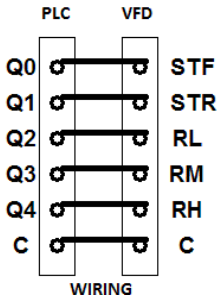

**EX96: Direction control of induction motor through VFD.**

When push button I0 is pressed, motor/ conveyor should run in forward direction Q0. When limit switch I2 is on then it should run in reverse direction Q1. When limit switch I3 is on then repeat process for 10 cycles. I1 is emergency stop switch.



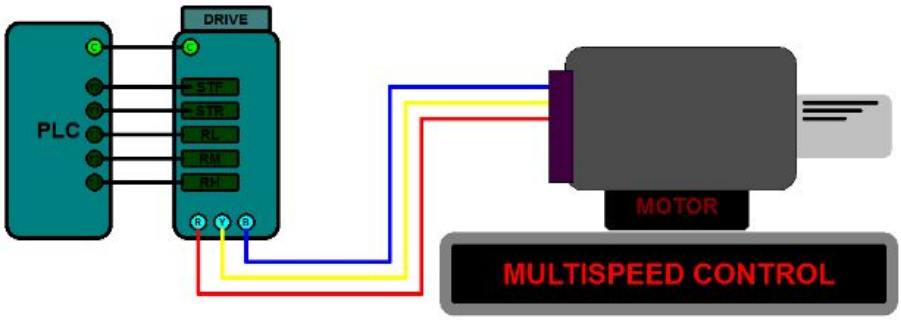



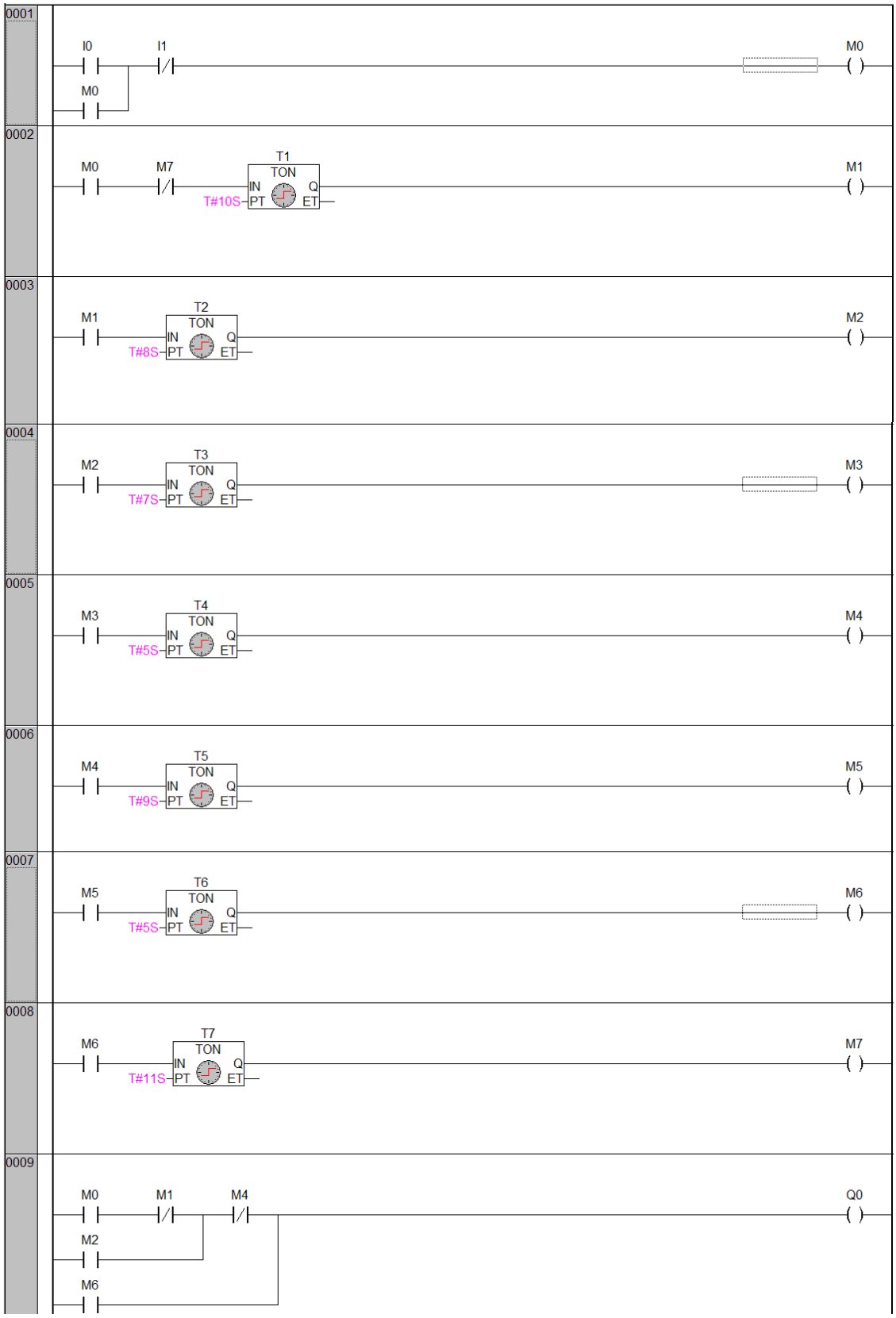

## EX97: Multispeed control of an induction motor using VFD controlled by PLC. (Using output coil)

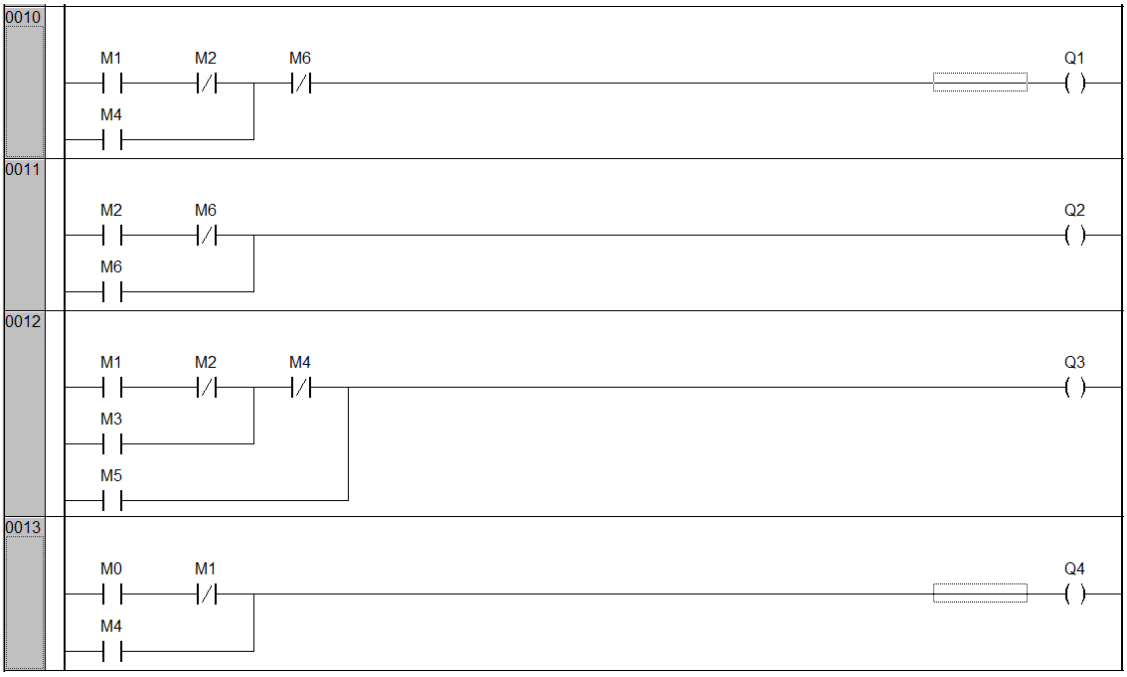



| | | | | Q4 | Q3 | Q2 | Q1 | Q0 | |
|---|---|---|---|---|---|---|---|---|---|
| Pr. No. | FLAG FOR EACH SPEED | SPEED Hz. | TIME IN SECOND | RH | RM | RL | R | F | QB0 ⇩ |
| 4 | M0 | 50 | 10 | 1 | 0 | 0 | 0 | 1 | 17 |
| 5 | M1 | 25 | 8 | 0 | 1 | 0 | 1 | 0 | 10 |
| 6 | M2 | 10 | 7 | 0 | 0 | 1 | 0 | 1 | 5 |
| 24 | M3 | 30 | 5 | 0 | 1 | 1 | 0 | 1 | 13 |
| 25 | M4 | 40 | 9 | 1 | 0 | 1 | 1 | 0 | 22 |
| 26 | M5 | 35 | 6 | 1 | 1 | 0 | 1 | 0 | 26 |
| 27 | M6 | 15 | 11 | 1 | 1 | 1 | 0 | 1 | 29 |

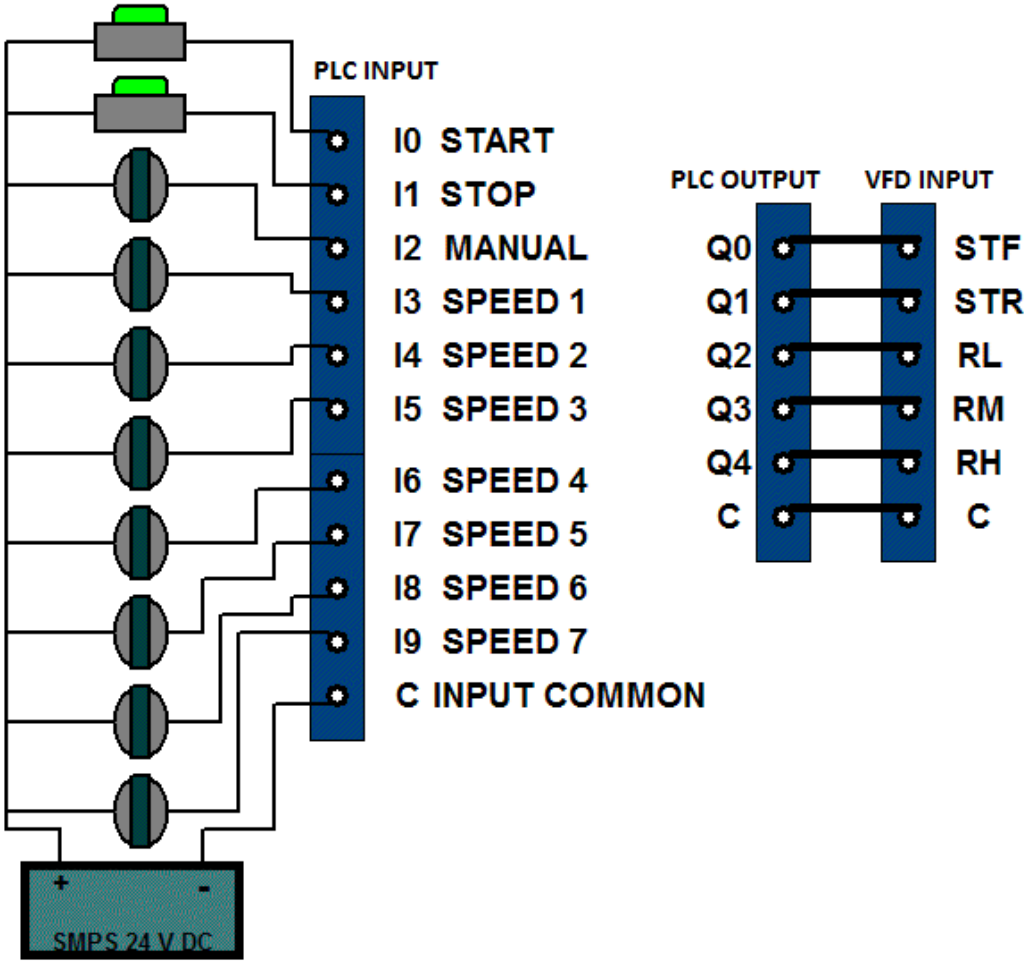
Pr. is parameter of VFD. For multispeed parameter number you need to refer VFD user manual. It is manufacturer decided. Flag for each speed we are going to generate in PLC program to operate outputs. Speed we shall decide and will feed in VFD parameters. Time we shall take in PLC program. RL, RM and RH are decided by manufacturer. Forward and reverse direction we can decide. For forward, reverse, RL, RM and RH we are going to generate PLC output. We have decided outputs Q0, Q1, Q2, Q3 and Q4. We can make this program either using output coils or by using output variable. Let us make it using output coils.

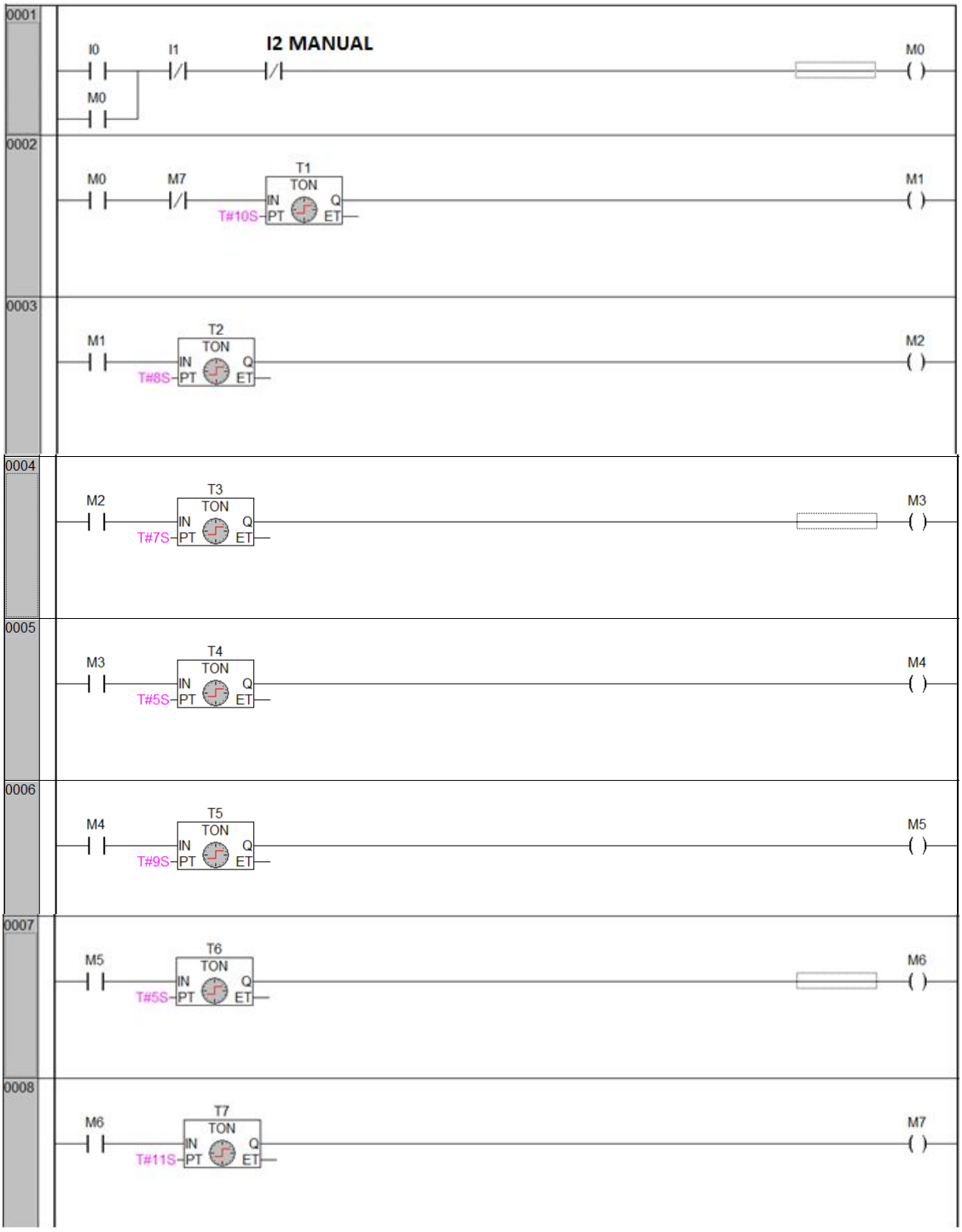

0001
I0
I1
M0
M0
0002
M0
M7
T1
TON
IN
PT
Q
ET
T#10S
M1
0003
M1
T2
TON
T#8S
M2
0004
M2
T3
TON
T#7S
M3
0005
M3
T4
TON
T#5S
M4
0006
M4
T5
TON
T#9S
M5
0007
M5
T6
TON
T#5S
M6
0008
M6
T7
TON
T#11S
M7
0009
M0
M1
M4
Q0
M2
M6

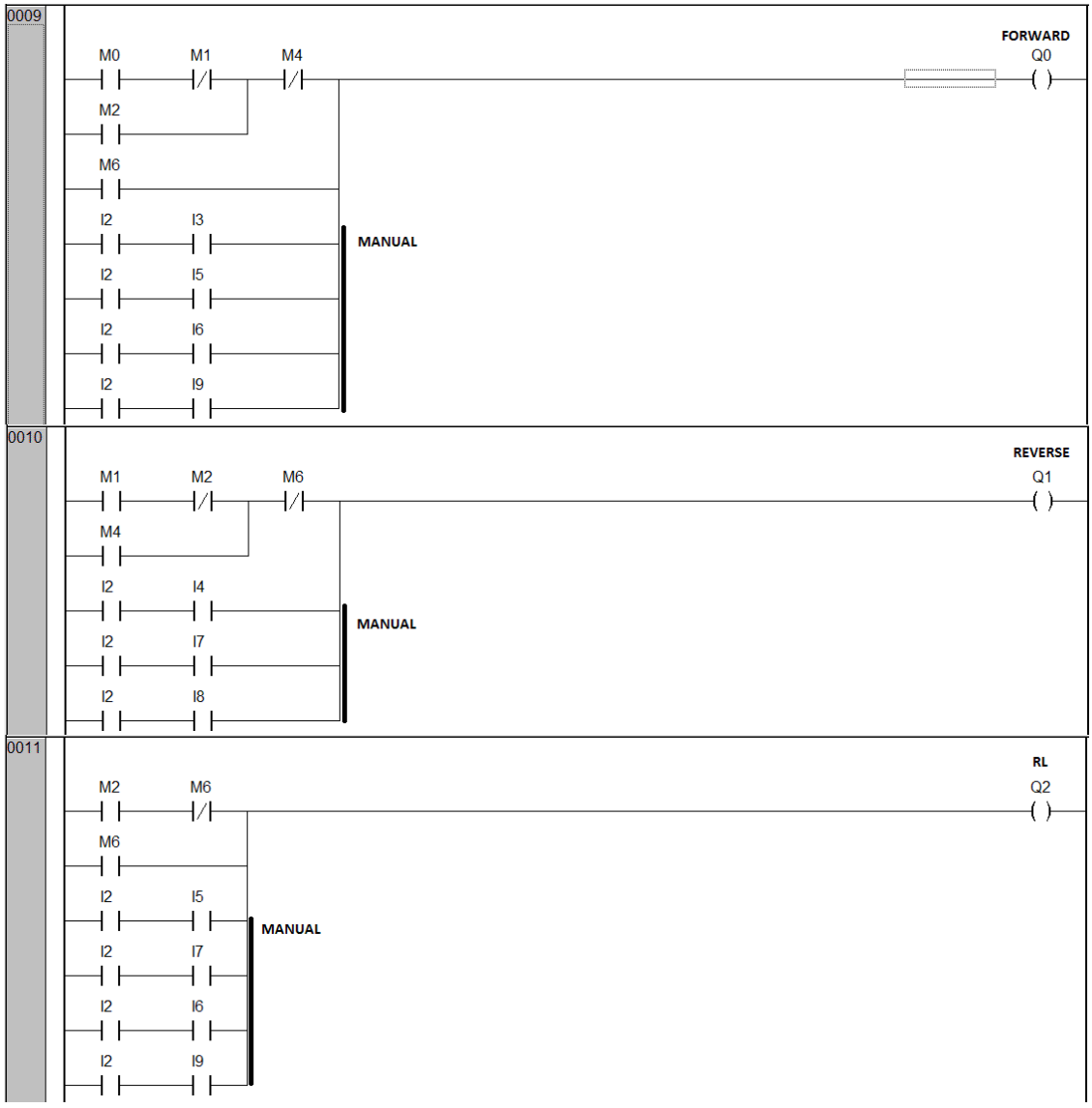


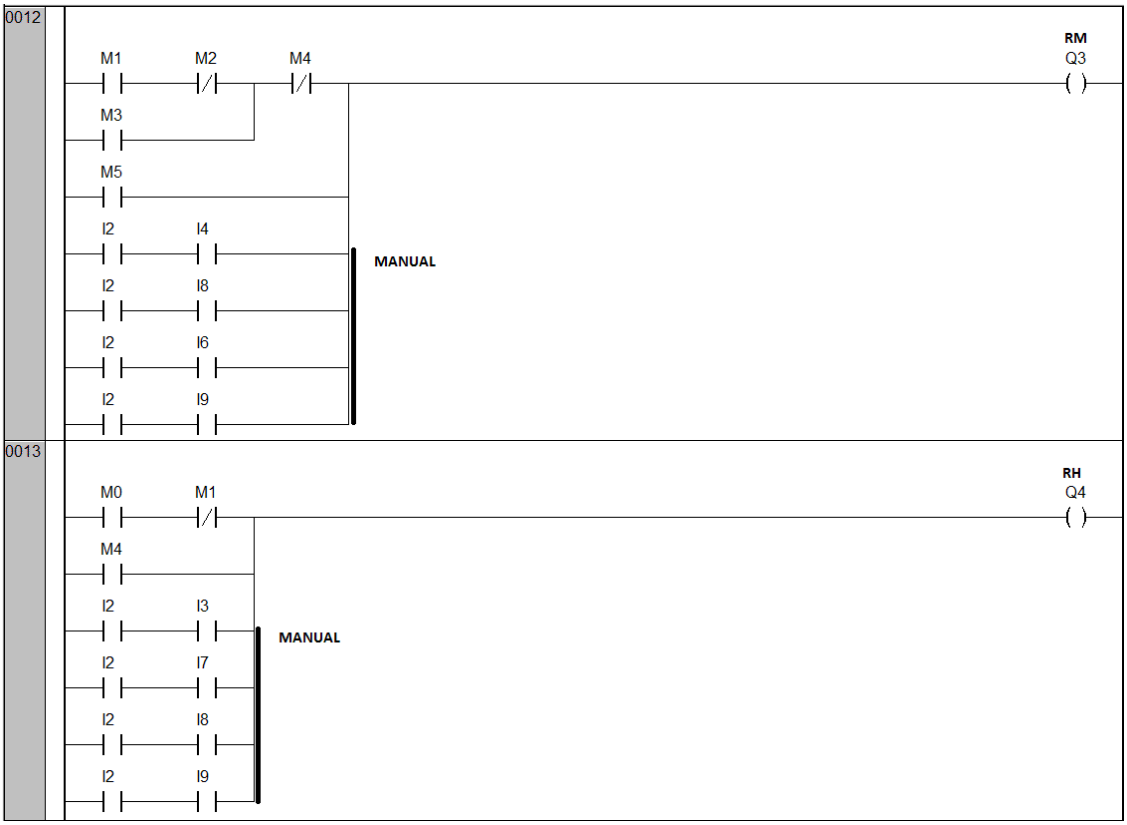

**EX98:** Add manual operation in previous example to operate individual speed.



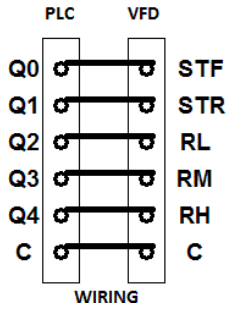

For manual operation we have taken here one selector switch for manual selection and seven selector switches for seven speeds.


0001
I0
I1
I2 MANUAL
M0
M0
0002
M0
M7
T1
TON
IN
PT
Q
ET
T#10S
M1
0003
M1
T2
TON
T#8S
M2
0004
M2
T3
TON
T#7S
M3
0005
M3
T4
TON
T#5S
M4
0006
M4
T5
TON
T#9S
M5
0007
M5
T6
TON
T#5S
M6
0008
M6
T7
TON
T#11S
M7


0009
M0
M1
M4
M2
M6
I2
I3
I2
I5
I2
I6
I2
I9
MANUAL
FORWARD
Q0
0010
M1
M2
M6
M4
I2
I4
I2
I7
I2
I8
MANUAL
REVERSE
Q1
0011
M2
M6
M6
I2
I5
I2
I7
I2
I6
I2
I9
MANUAL
RL
Q2

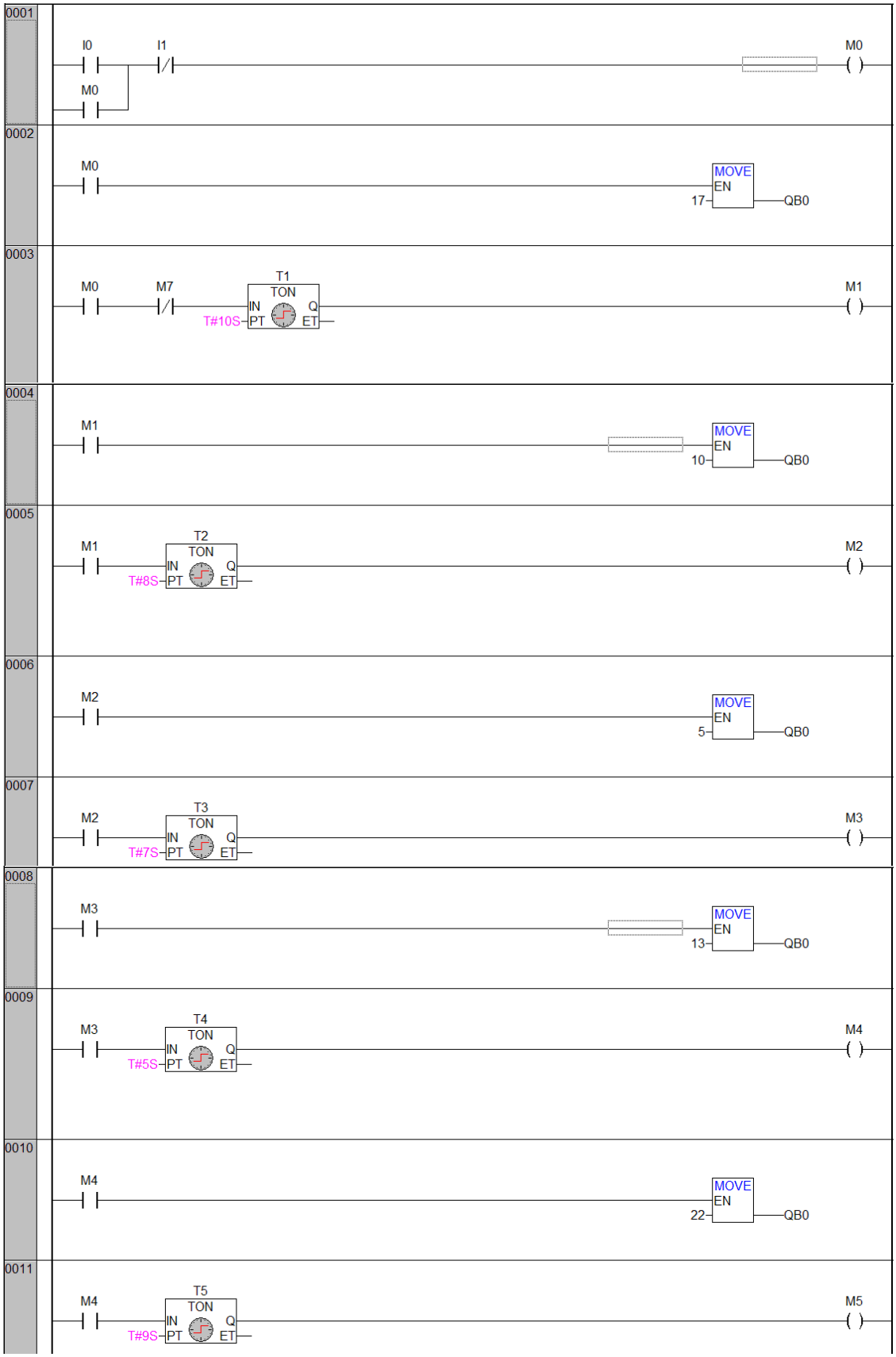


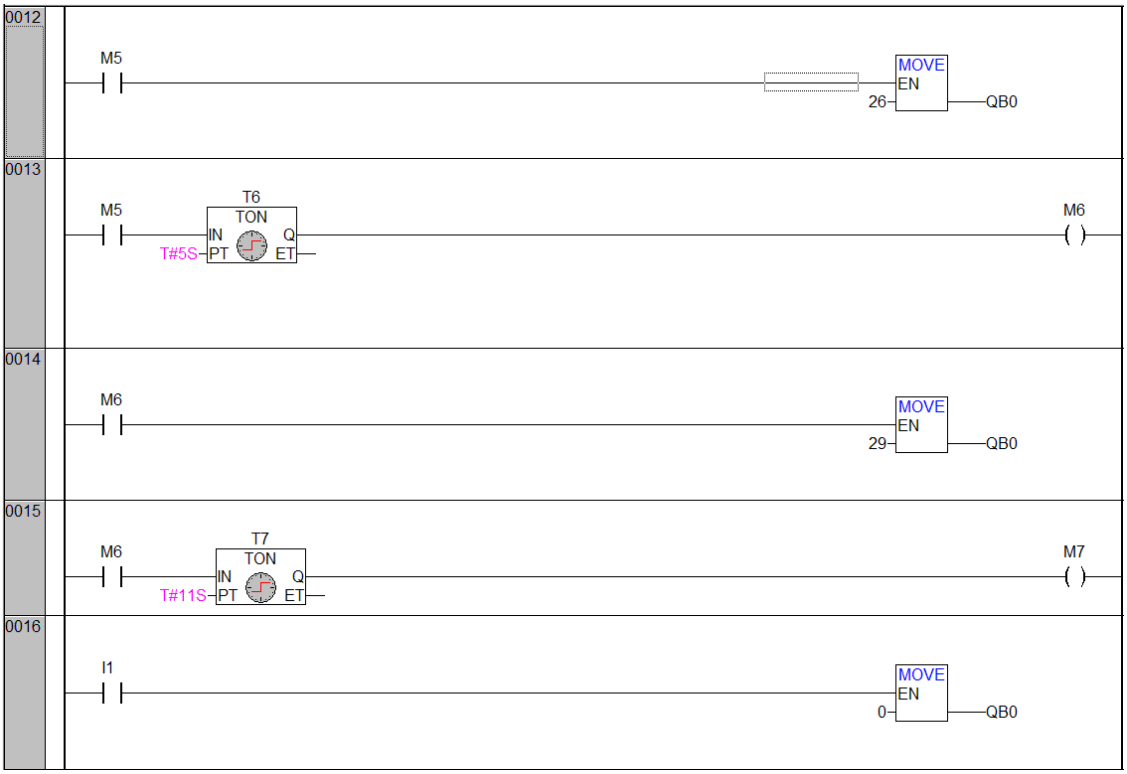

## EX99: Multispeed control of an induction motor using VFD controlled by PLC. (Using output variable)



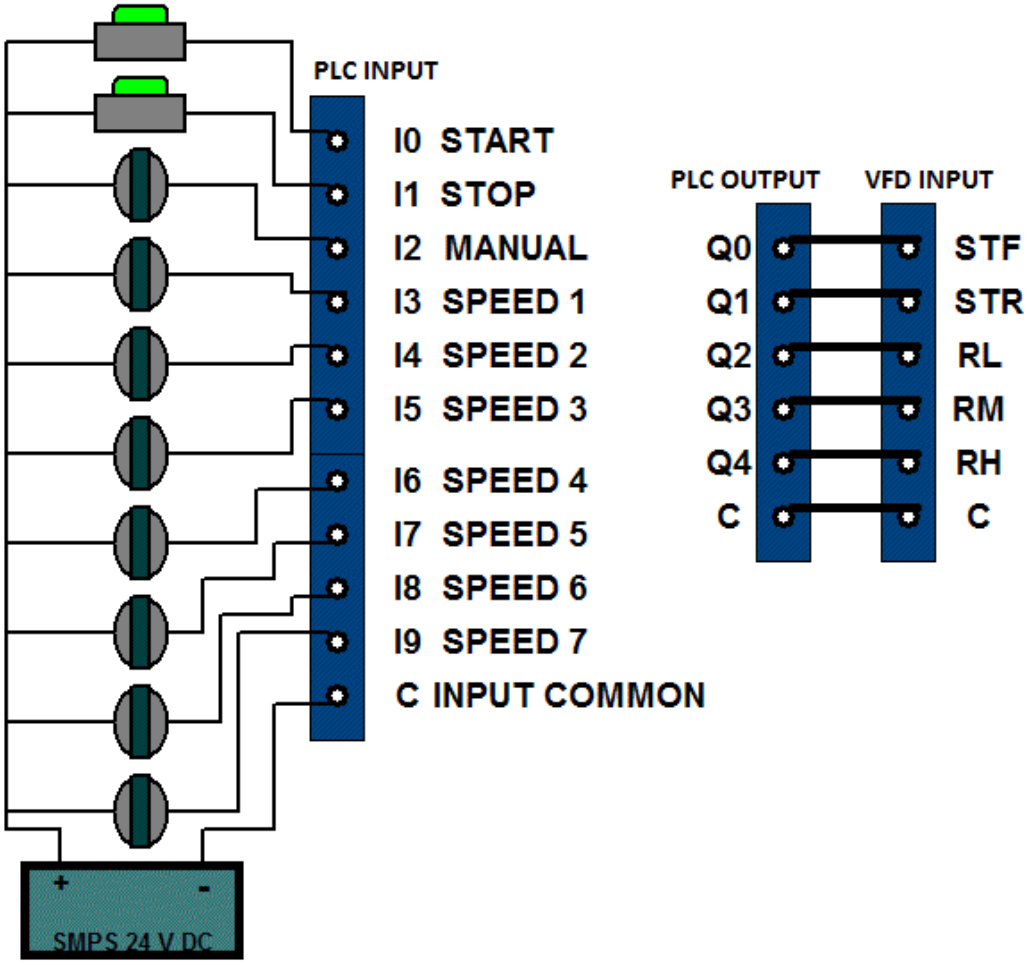

| | | | | Q4 | Q3 | Q2 | Q1 | Q0 | |
|---|---|---|---|---|---|---|---|---|---|
| Pr. No. | FLAG FOR EACH SPEED | SPEED Hz. | TIME IN SECOND | RH | RM | RL | R | F | QB0 ⇩ |
| 4 | M0 | 50 | 10 | 1 | 0 | 0 | 0 | 1 | 17 |
| 5 | M1 | 25 | 8 | 0 | 1 | 0 | 1 | 0 | 10 |
| 6 | M2 | 10 | 7 | 0 | 0 | 1 | 0 | 1 | 5 |
| 24 | M3 | 30 | 5 | 0 | 1 | 1 | 0 | 1 | 13 |
| 25 | M4 | 40 | 9 | 1 | 0 | 1 | 1 | 0 | 22 |
| 26 | M5 | 35 | 6 | 1 | 1 | 0 | 1 | 0 | 26 |
| 27 | M6 | 15 | 11 | 1 | 1 | 1 | 0 | 1 | 29 |

Pr. is parameter of VFD. For multispeed parameter number you need to refer VFD user manual. It is manufacturer decided. Flag for each speed we are going to generate in PLC program to operate outputs. Speed we shall decide and will feed in VFD parameters. Time we shall take in PLC program. RL, RM and RH are decided by manufacturer. Forward and reverse direction we can decide. For forward, reverse, RL, RM and RH we are going to generate PLC output. We have decided outputs Q0, Q1, Q2, Q3 and Q4. We can make this program either using output coils or by using output variable. Let us make it using output variable.

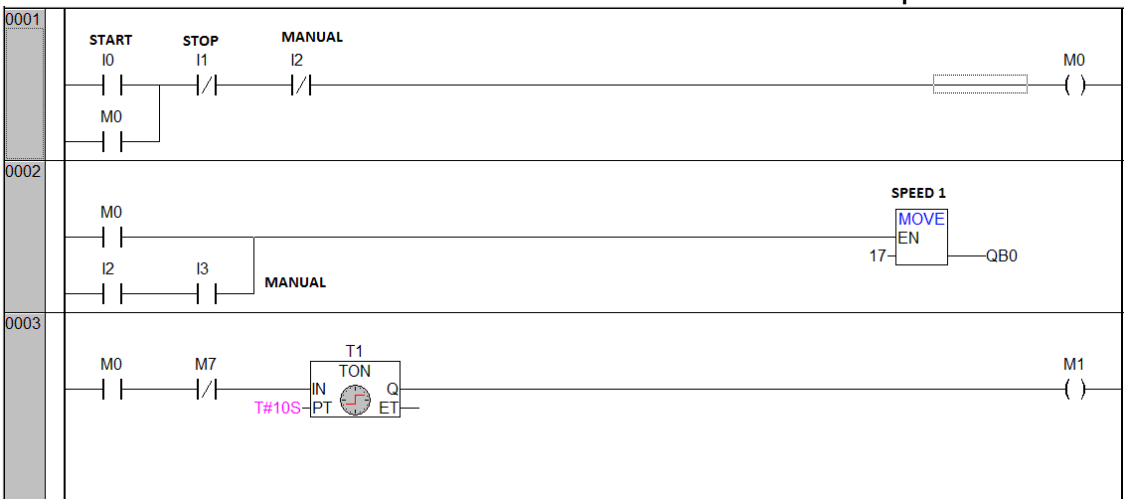

0001
I0
I1
M0
M0
0002
M0
MOVE
EN
17
QB0
0003
M0
M7
T1
TON
IN
PT
T#10S
Q
ET
M1
0004
M1
MOVE
EN
10
QB0
0005
M1
T2
TON
IN
PT
T#8S
Q
ET
M2
0006
M2
MOVE
EN
5
QB0
0007
M2
T3
TON
IN
PT
T#7S
Q
ET
M3
0008
M3
MOVE
EN
13
QB0
0009
M3
T4
TON
IN
PT
T#5S
Q
ET
M4
0010
M4
MOVE
EN
22
QB0
0011
M4
T5
TON
IN
PT
T#9S
Q
ET
M5

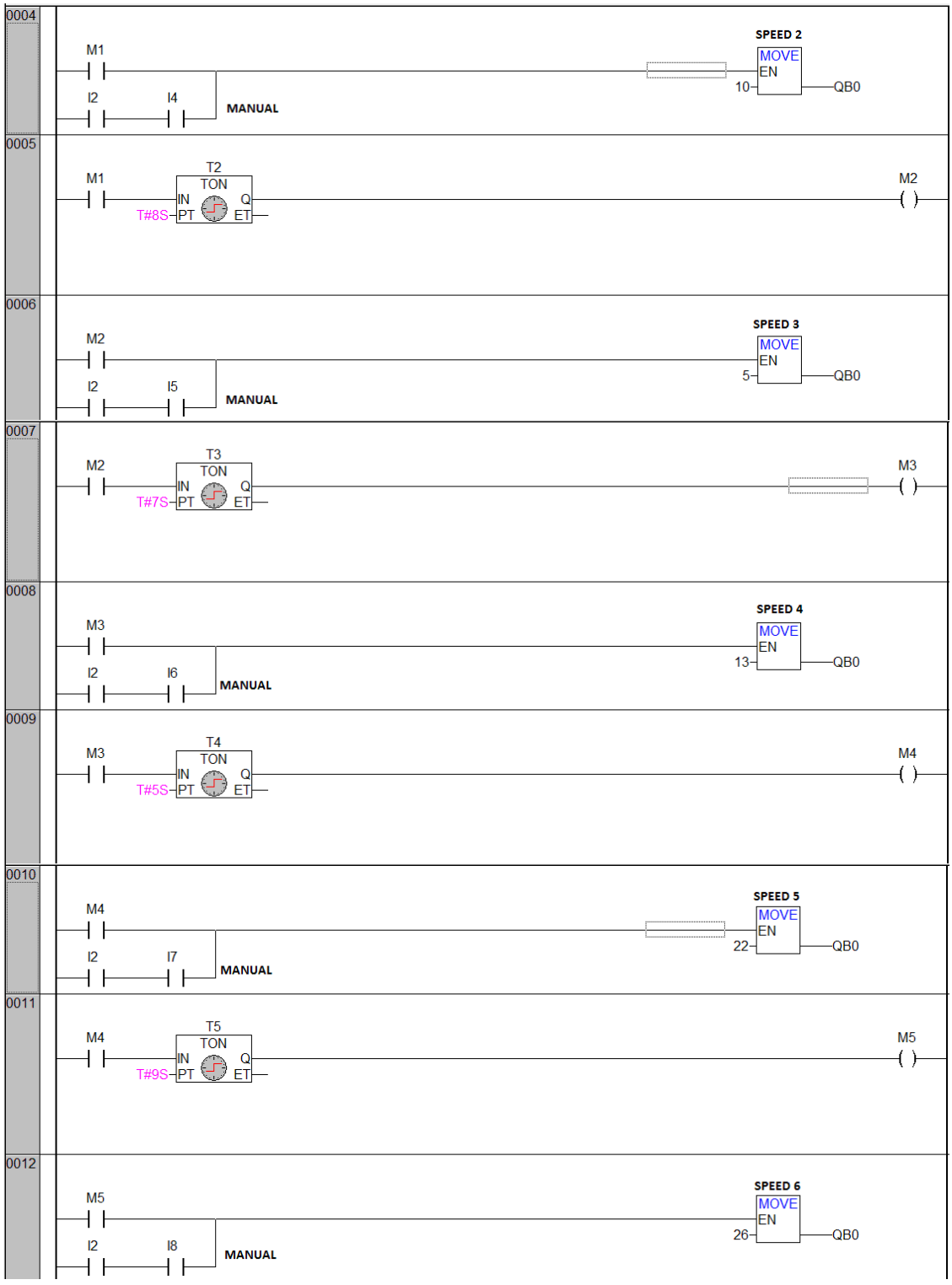


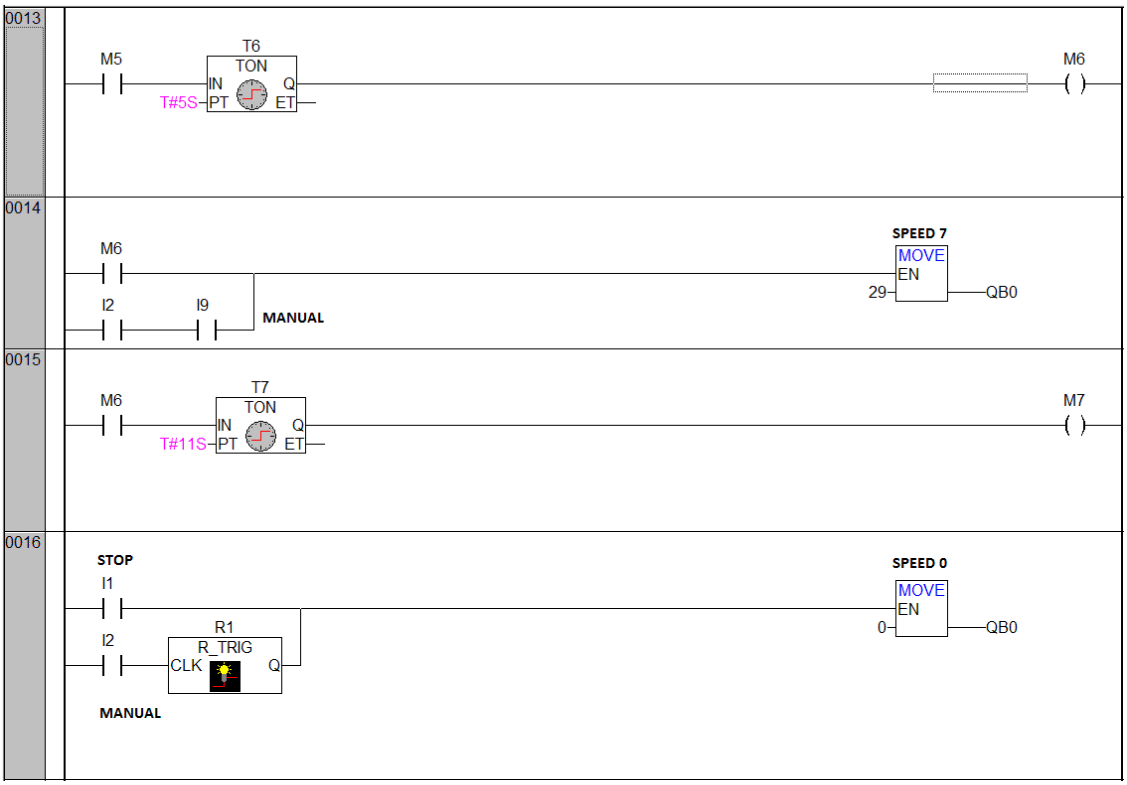

**EX100:** Add manual operation in previous example 99 to operate individual speed.




For manual operation we have taken here one selector switch for manual selection and seven selector switches for seven speeds.

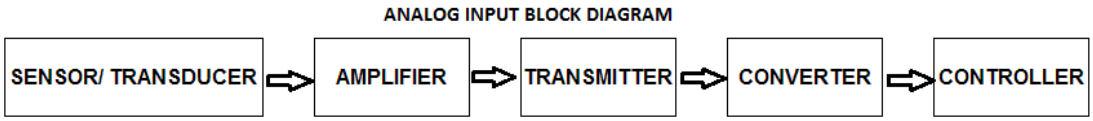



0004
M1
I2
I4
MANUAL
SPEED 2
MOVE
EN
10
QB0
0005
M1
T2
TON
IN
PT
T#8S
Q
ET
M2
0006
M2
I2
I5
MANUAL
SPEED 3
MOVE
EN
5
QB0
0007
M2
T3
TON
IN
PT
T#7S
Q
ET
M3
0008
M3
I2
I6
MANUAL
SPEED 4
MOVE
EN
13
QB0
0009
M3
T4
TON
IN
PT
T#5S
Q
ET
M4
0010
M4
I2
I7
MANUAL
SPEED 5
MOVE
EN
22
QB0
0011
M4
T5
TON
IN
PT
T#9S
Q
ET
M5
0012
M5
I2
I8
MANUAL
SPEED 6
MOVE
EN
26
QB0



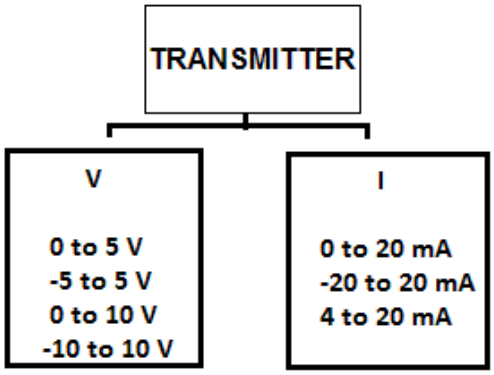

## ANALOG INPUT OUTPUT:

Yet we have gone through digital signal where we had two states ON and OFF. In industry you will have to work on analog input output signal too. Analog signal does not have just two states but it has signal variation like 0 to 5V, 0 to 10, 0 to 20mA, 4 to 20mA. When we have application where we have analog input like temperature, pressure, flow, level, displacement, luminous intensity etc. there we will have to write analog program. When we have analog output like control valve, vibrator, motor speed controller/ drives etc. there we will have to write analog program. Analog input like temperature, pressure, flow, level, displacement, luminous intensity etc. are not electrical phenomenon. First it needs to be converted to electrical energy. If we are able to convert non electrical phenomenon to electricity by any means then we will be able to control it. The device used to convert one form of energy (nonelectrical here) to another form of energy is called **Transducer.** There are different types of transducer available depending on the method used to convert one form to another. So here we have two terms; analog sensor and transducer. Question can be asked, what is the difference between analog sensor and transducer? **A transducer can be a sensor but a sensor cannot be a transducer.** To work easily on analog system in industrial automation let us go through block diagram.



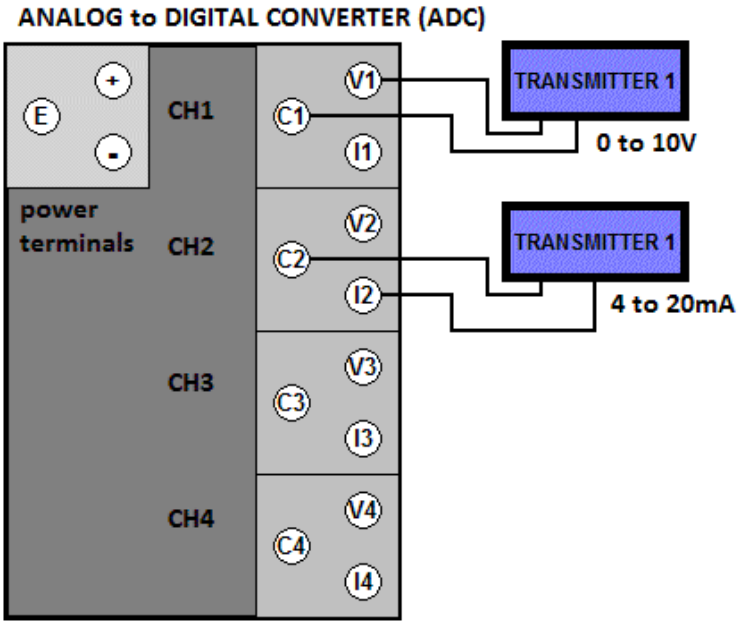

Sensor/ transducer sense the variation and convert it into a very small amount of electrical energy i. e. millivolt. This small amount of energy is amplified and is sent to transmitter.

Transmitter converts that signal into standard signal and transmits to converter. Transmitter converts signal into voltage and current standard.



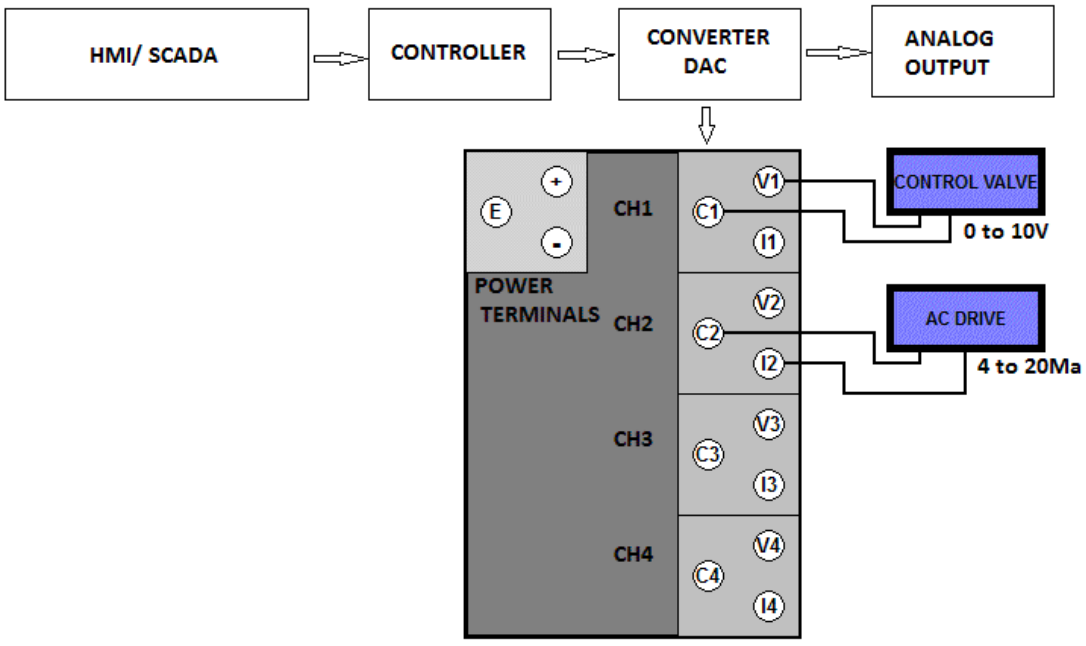

Which standard is better, voltage or current? Current signal is better than voltage because in terms of voltage we have voltage drop when we have long distance wiring but current remains the same at both the ends. In voltage 0 to 10 v is generally used and in current 4 to 20 mA. Again, a question can be asked that why 4 to 20 mA is preferred? It is preferred because of safety reason. Suppose temperature is 0*C, in 0 to 20 mA it will show 0 ampere and in wire break or device failure also it will show 0 ampere. If we have taken 4 to 20 mA and the temperature is 0*C then it will show minimum 4 mA but in case of wire break or device failure it will show 0. Another reason of 4 to 20 mA is that control valve works linear on 4 to 20 mA than 0 to 20 mA.

Converter in analog input is ADC (analog to digital converter) which converts analog signal into digital signal. In very simple if we say, varying voltage or ampere is converted into digit. This ADC has different name like analog input module, AD card/ module etc. it is available in 2AD, 4 AD, 8AD, 16AD and in 64AD module. If it is 4AD module then it will have 4 channels where transmitter can be connected. Each channel has 3 terminals V, I and C. it means each channel gives us option to connect either voltage standard or current standard signal. From transmitter two wires come to channel.




Converter ADC will convert analog to digit and this digit will be received by controller/ PLC by using analog program.

Analog output:




Analog output is opposite to analog input. In this function digit is converted to analog signal. I have taken HMI/ SCADA in block diagram because we need to change the digit to be converted to analog many times without changing PLC program. From HMI/ SCADA we send digit to PLC. PLC sends this digit to DAC (digital to analog converter). DAC convers this digit to analog signal

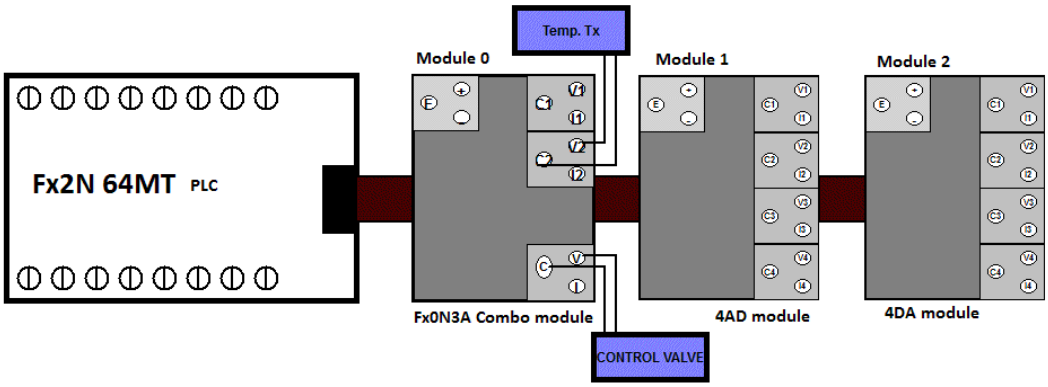
and sends it to analog output field device. DAC module has also many channels and each channel give option to connect either V or current operated device.

Now we come to the main important part, analog programming. It totally depends on manufacturer. Without user manual you cannot make program for ADC or DAC. Different module has its own rule. PLC brand and ADC/ DAC module brand should be the same to be interfaced. To make program for ADC/ DAC we need to search few things in its user manual.

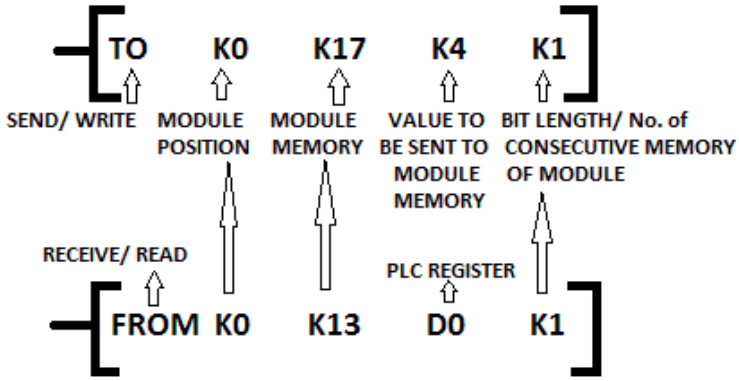
1> Channel selection: As you will get many channels on ADC/ DAC so you need to select the channel from which analog I/ O is connected.
2> AD/ DA conversion start: When transmitter signal is on channel, we need to give command to start analog to digit conversion. In the same way we need to give command for digit to analog command.
3> Signal standard (V or I) selection.
4> Analog input memory: Converted digit from analog signal is stored in some memory of ADC. We need to search that memory to take converted digit.
5> Analog output memory: To convert digit to analog we need to send digit to some specific memory of DAC. We need to search that memory.

Let me show you how to make program using this information by one example.

**EX101:** Here we have used Mitsubishi Fx2N64MT PLC and FxON3a combo module with 2 channels analog input and 1 channel analog output. We have one temperature transmitter connected to analog channel 2 and we have one control valve connected to analog output channel. Receive converted digit in D0 register of PLC. Send digit of register D2 to analog output memory to be converted into analog signal.



In different brand of PLC, you will get different instruction for analog programming. That instruction and one program example will be given in analog module user manual. For Mitsubishi it is ‘To’ and ‘From’ instructions.




In Mitsubishi K denotes decimal. When PLC has to send something to other module, TO instruction is used. Module number or module position mentions that to which module something is to be sent. Module just after PLC is module number 0. Module memory is the memory of ADC/ DAC here which is connected to PLC.  Value or that something you want to send to the memory of module is mentioned here. Bit length is the number of consecutive memories of module where you send the value. K1 denotes 16 bit or one memory. If it is K3 then value will be sent to 3 consecutive memories. There is no need to write the same logic thrice. In example given in above picture, TO instruction says that **decimal value 4 is being sent to one memory location 17 of module number 0.**

In FROM instruction also we have almost similar content. When PLC has to receive something from other module then FROM instruction is used. In above picture FROM instruction says that PLC receives something from memory location 13 of module 0 and stores in its register D0.

Now let us go through the rules of Fx0N3A module.

### 3.4 Allocation of buffer memories (BFM)

| BFM No. | b15-b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|
| # 0 | Reserved | Current value of input data (stored in 8 bits) of the A/D channel selected by b0 of BFM#17 | | | | | | | |
| # 16 | | Current value of output data on D/A channel (stored in 8 bits) | | | | | | | |
| # 17 | Reserved | | | | | | D/A start | A/D start | A/D channel |
| # 1- # 15 #18-# 31 | Reserved | | | | | | | | |

- BFM #17:
  b0 = 0 analog input channel 1 is selected
  b0 = 1 analog input channel 2 is selected
  b1 = 0 ⇨ 1, the A/D conversion process is started
  b2 = 1 ⇨ 0, the D/A conversion process is started
- Note: These buffer memory devices are stored/located within the FX0N-3A.

BFM is the memory of module. BFM 17 is for analog input channel selection, AD start and DA start.

BFM 17 is 16-bit memory in which only 3 bit b0, b1 and b2 we can use for read and write as per its rule.

b0 CHANNEL SELECTION

0 □ Channel 1

1 □ Channel2

b1 analog to digit conversion start

0 □ 1 (0 to 1 we can make by writing 0 in one ladder and 1 in another ladder to b1 of BFM 17)

b2 digit to analog conversion start

1 □ 0 (1 to 0 we can make by writing 1 in one ladder and 0 in another ladder to b1 of BFM 17)

As per user manual BFM0 stores converted digit of the analog channel selected.

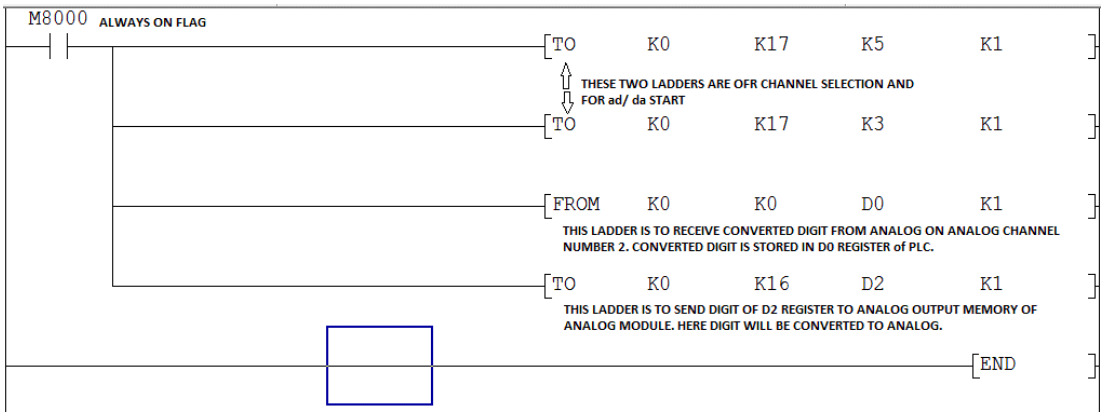
BFM 16 is output BFM. Any decimal digit sent to BFM16 will be converted to analog signal.

Now we come to our example requirements. We have to select analog input channel 2. We have to start SD and DA conversion. We have to receive converted digit from BFM to D0 of PLC and we have to send value of D2 to BFM 16 to convert digit to analog.

| **BFM17** | **b2** | **b1** | **b0** |
|---|---|---|---|
| | **1** | **0** | **1** |
| | **0** | **1** | **1** |

In BFM 17 we have made b0 1 for analog input channel 2 selection, b1 0 □ 1 for AD start and b2 1 □ 0 for DA start.

So, we have to write binary 101 in one ladder its decimal is 5 and we have to write binary 011 to next ladder which decimal is 3.




Each module will have its own minimum and maximum value range such as 0 to 255, 0 to 2000, 0 to 4000 etc. we need to check the module catalogue for this. For Fx0N3A module it is 0 to 255. If temperature range is 0 to 100*C then at 0*C it will show converted digit 0 and for 100*C it will show the digit 255. But obviously you would like to display digit 100 for 100*C.

| TEMPERATURE in *C | ANALOG MODULE VALUE | REQUIRED VALUE |
|---|---|---|
| 0 | 0 | 0 |
| 50 | 127.5 | 50 |
| 100 | 255 | 100 |

For required value display you need to scale it by using a formula.

$$\text{REQUIRED VALUE} = \frac{\text{PRESENT RAW VALUE - MODULE MINIMUM VALUE}}{\text{MODULE MAXIMUM VALUE - MODULE MINIMUM VALUE}} \times \text{REQUIRED MAXIMUM VALUE}$$

$$\text{REQUIRED VALUE} = \frac{\text{D0 - 0}}{\text{255 - 0}} \times 100$$

$$= \frac{127.5 - 0}{255 - 0} \times 100$$

( SUPPOSE PRESENT RAW VALUE IS 127.5)

$$= \frac{12750}{255}$$

$$= 50$$

**ANALOG INPUT SCALING**

$$\frac{\text{PRESENT RAW VALUE - MODULE MINIMUM VALUE}}{\text{MODULE MAXIMUM VALUE - MODULE MINIMUM VALUE}} \times \text{REQUIRED MAXIMUM VALUE}$$

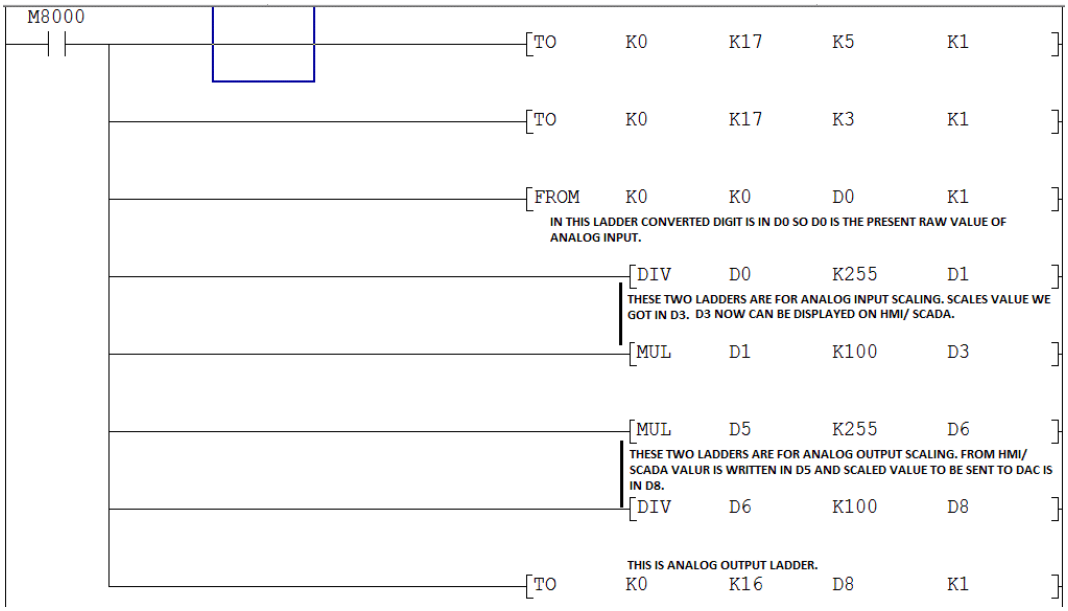
$$\frac{\text{PRESENT RAW VALUE}}{\text{MODULE MAXIMUM VALUE}} \times \text{REQUIRED MAXIMUM VALUE} \quad \text{(WHEN MODULE MINIMUM VALUE IS 0)}$$

**ANALOG OUTPUT SCALING**

$$\frac{\text{PLC PRESENT RAW ANALOG OUTPUT VALUE} \times \text{MODULE MAXIMUM VALUE}}{\text{REQUIRED MAXIMUM VALUE}}$$

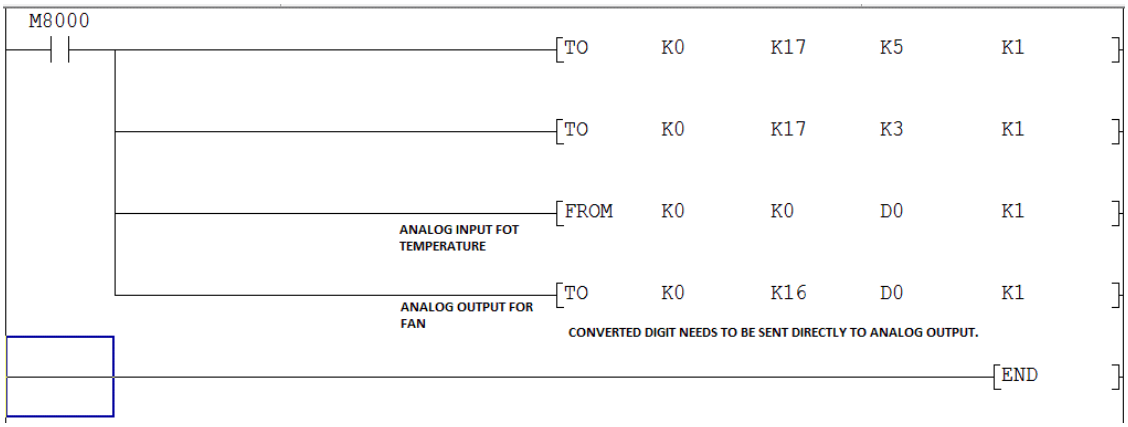
If we have HMI/ SCADA in our project then scaling can be done in HMI/ SCADA directly.

**EX102:** ADC/ DAC programming with scaling for Fx0N3A module.

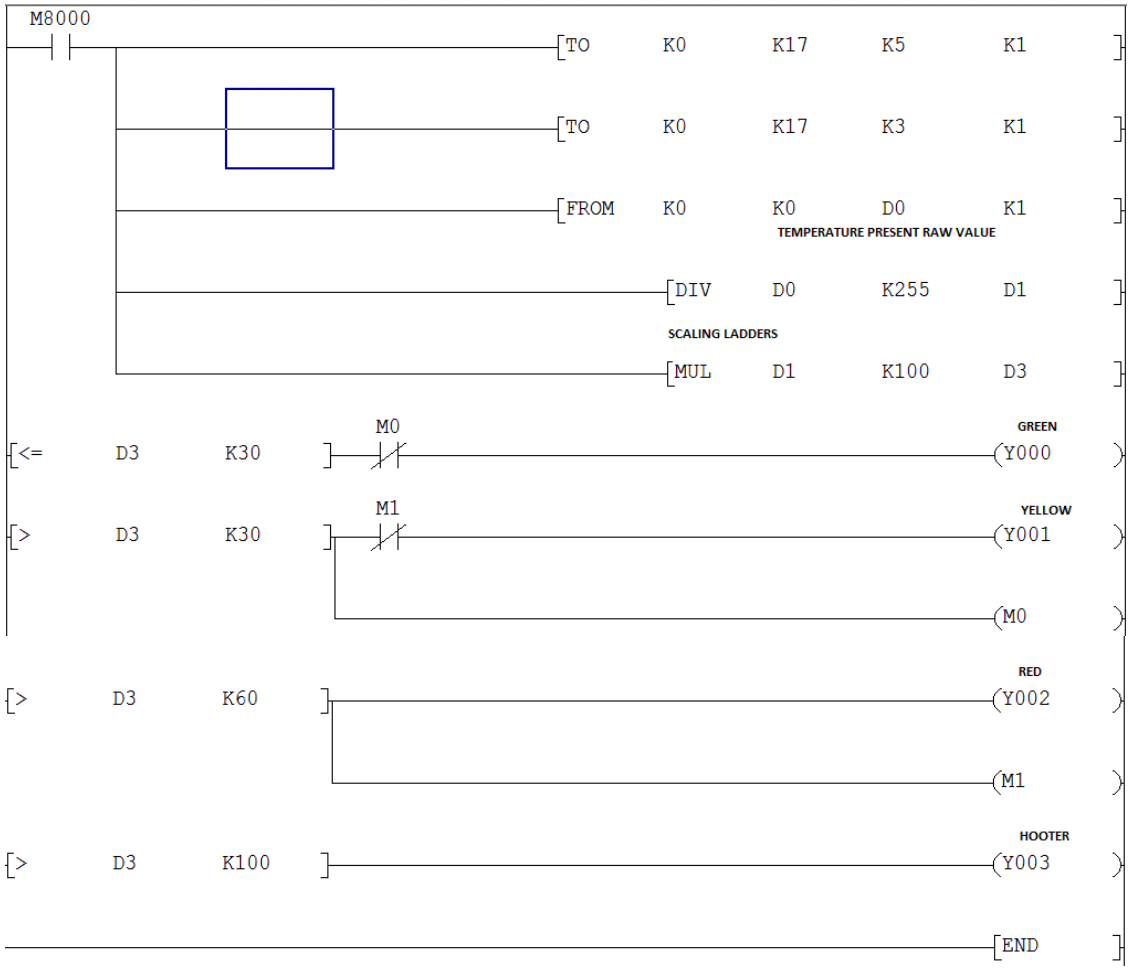



**EX103:** There is a temperature transmitter and there is a fan driven by VFD. As temperature increases, fan speed should increase and as temperature

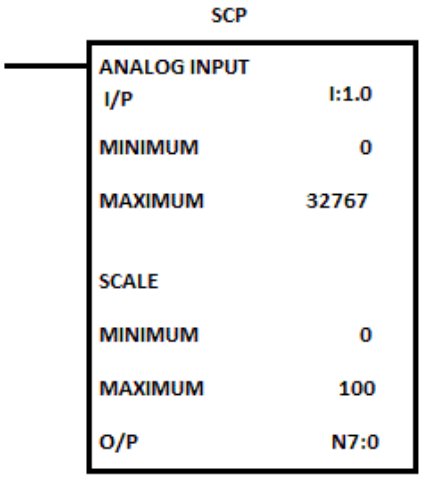
decreases then fan speed should also decrease. Fx0N3A module is used.

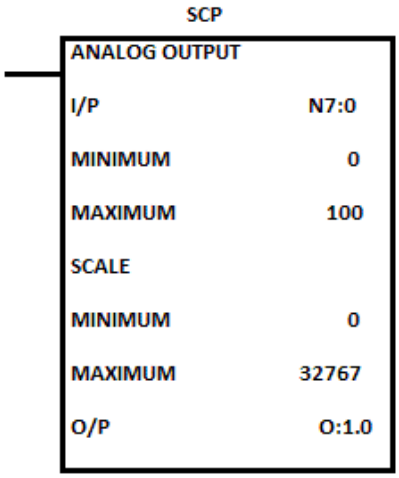



**EX104: T**here is a temperature transmitter connected to analog input channel number 2 of module Fx0N3A. When temperature is 10*C to 30*C, get the green lamp on. When temperature is 31*C to 60*C, get the yellow lamp on and when temperature is 61*C to 100*C then get the red lamp on. Above 100*C hooter will be on.



In many brands of PLC analog input and output address for each channel is prefixed. You just need to search it in user manual and can use directly. You will get converted value directly in those analog input channel addresses. You can scale it using formula or by scale instruction given in that PLC. In Allen Bradley PLC it is SCP (scale with parameter) for scaling analog values.

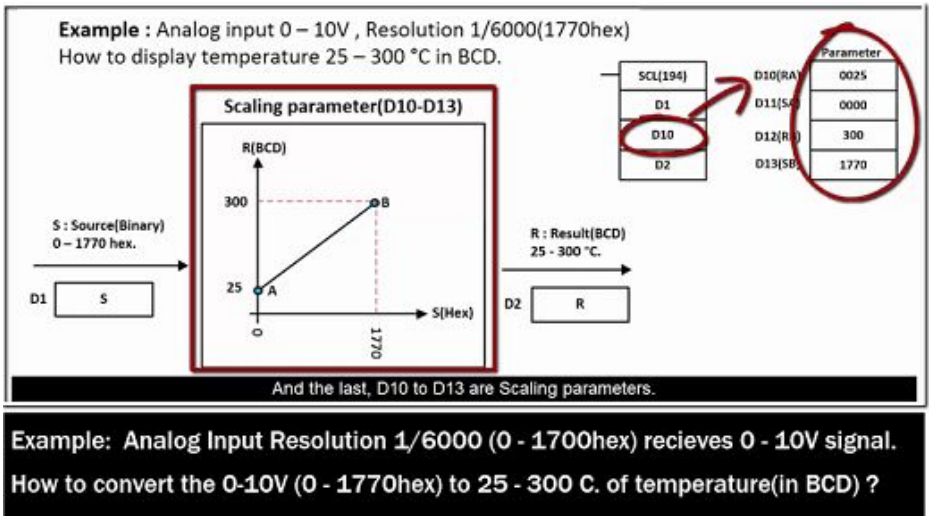


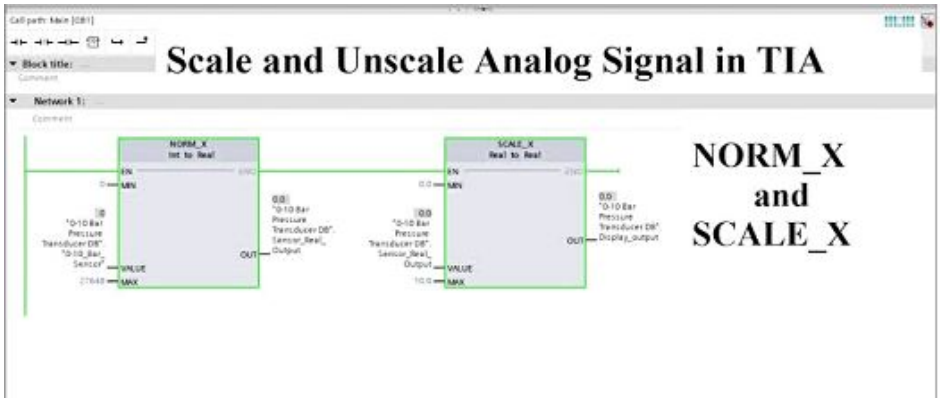

If I:1.0 is the analog input address of channel 1 then converted value will directly come to I:1.0. You can see its minimum and maximum value in analog module user manual. In scale you can give your minimum and maximum desired values.

If O:1.0 is the channel number analog output then you can send your digit directly to this address. You can give your minimum and maximum value then you can give that analog module minimum and maximum value as given in user manual.

In new Mitsubishi PLCs you have been given direct address of analog input output channels and you can use SCL instruction and can directly scale it.




For Siemens NORM X and SCALE X instruction is used for scaling analog value.




**PID (Proportional Integral Derivative) controller:**

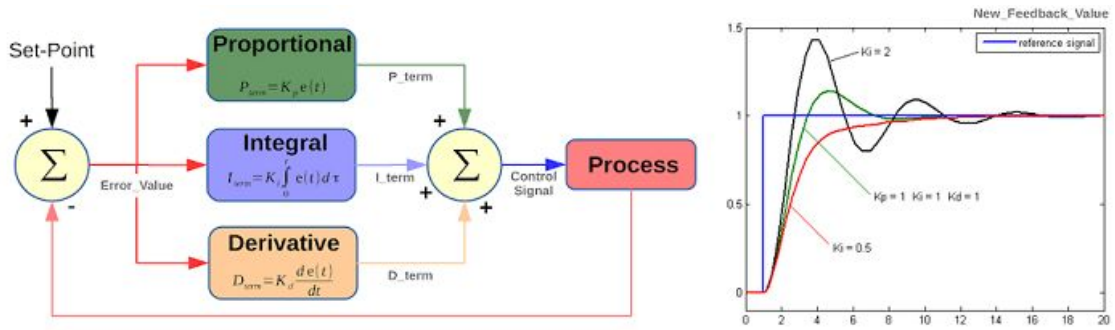
PID is a controller with three controls. It is a hardware used to maintain or achieve the required set-point in analog system. It is a loop control where controller maintains the system, checks error, calculates and takes corrective action. It is an accurate and reliable controller. In industries it is used to maintain required temperature, pressure, flow, level etc. in a close loop system. It takes analog input signal as feedback, compares with set-point, takes calculative action and operates analog output to maintain set-point.


**PID CONTROLLER HARDWARE**

PID has three controllers inside it; P, I and D. P is the main controller. I and D are not used without P. P controller must be there. We use P control, PI control, PD control and PID control. Proportional controller P reduces the error and reaches to the set-point. This proportion is set as per the nature of material in field. If we take an example of a tank where we have to maintain the set-point of liquid inside it, P will totally depend on the nature of liquid. If liquid is thin like water, petrol, spirit etc. then proportion will be set less. If you set high proportion for such liquid then control valve will be on with high percentage and more liquid will go out very fast causing negative error in tank level. If liquid is thick like honey, grease, Mobil etc. then proportion will be set so that set-point can be reached fast as thick liquid will move out slowly from tank. Proportion is set by error and trial method generally. P monitors present error only and do calculations. With P controller only error is not eliminated 100%. Some error remains.

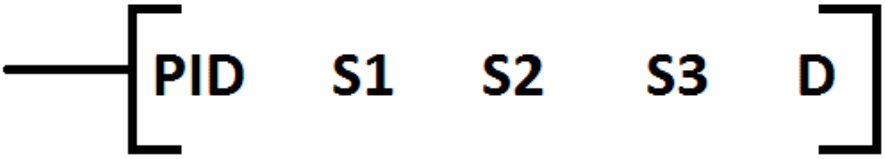
To reach to accuracy integral controller I is used. It takes past error on given interval of time, integrates and starts P again and again to reach accurate set-point. I monitor past error. PI is accurate controller.

Derivative controller D works on future error. It monitors the rate of change of error and brings it to set point in short time. It excites the process fast initially and leaves it on PI further. D is not used in all systems. It is used where process is slower like temperature and we need to achieve set-point faster. We generally don’t use D in case of liquid process like level and flow control.




On PID hardware you will get few keys to feed required data/ values. Analog input signal from transmitter will be directly connected to PID hardware. Analog output device will also be directly connected to PID hardware.

If you have PLC in your project then you can use PID instruction in PLC program. PID instruction can be called soft PID. You should have ADC and DAC connected to your PLC. When you use PID instruction in PLC program then you do not need PID hardware. In all brands of PLC they have provided some specific data/ registers/ addresses to feed required values of PID. Let me show you PID implementation with an example in Mitsubishi PLC. User manual will be required to see the rules.




There are three sources and one destination.

Source 1 (S1) is the set point. It uses single data register.

Source 2 (S2) is for current/ present value/ process value/ feedback from analog input sensor. It uses single data register.

Source 3 (S3) is for PID setting and it uses 25 consecutive data registers S3+0 to S3+24.

Destination (D) is for manipulated/ calculated/ output value of PID which will go to field analog output such as control valve. It uses single data register.

**PID setup parameters; S3**

The PID setup parameters are contained in a 25 register data stack. Some of these devices require data input from the user, some are reserved for the internal operation and some return output data from the PID operation.

**Parameters $S_3$+0 through $S_3$+6 must be set by the user.**

| Parameter S3 + P | Parameter name/function | Description | | Setting range |
|---|---|---|---|---|
| S3+0 | Sampling time **TS** | The time interval set between the reading the current Process Value of the system ($PV_{nf}$) | | 1 to 32767 msec |
| S3+1 | Action - reaction direction and alarm control | b0 | Forward operation(0), Reverse operation (1) | Not applicable |
| | | b1 | Process Value ($PV_{nf}$) alarm enable, OFF(0)/ ON(1) | |
| | | b2 | Output Value (MV) alarm enable, OFF(0)/ON(1) | |
| | | b3 - 15 | Reserved | |
| S3+2 | Input filter α | Alters the effect of the input filter. | | 0 to 99% |
| S3+3 | Proportional gain **KP** | This is a factor used to align the proportional output in a known magnitude to the change in the Process Value ($PV_{nf}$). This is the **P** part of the PID loop. | | 1 to 32767% |
| S3+4 | Integral time constant $T_I$ | This is the **I** part of the PID loop. This is the time taken for the corrective integral value to reach a magnitude equal to that applied by the proportional or **P** part of the loop. Selecting 0 (zero) for this parameter disables the **I** effect. | | (0 to 32767) x 100 msec |
| S3+5 | Derivative gain **KD** | This is a factor used to align the derivative output in a known proportion to the change in the Process Value ($PV_{nf}$) | | 1 to 100% |
| S3+6 | Derivative time constant **TD** | This is the **D** part of the PID loop. This is the time taken for the corrective derivative value to reach a magnitude equal to that applied by the proportional or **P** part of the loop. Selecting 0 (zero) for this parameter disables the **D** effect. | | (0 to 32767) x 10 msec |
| S3+7 to S3+19 | Reserved for use for the internal processing | | | |
| S3+20 | Process Value, maximum positive change | Active when S3+1, b1 is set ON. | This is a user defined maximum limit for the Process Value (PVnf). If the Process Value (PVnf) exceeds the limit, S3+24, bit b0 is set On. | 0 to 32767 |
| S3+21 | Process Value, minimum value | | This is a user defined lower limit for the Process Value. If the Process Value (PVnf) falls below the limit, S3+24, bit b1 is set On. | |
| S3+22 | Output Value, maximum positive change | Active when S3+1, b2 is set ON. | This is a user defined maximum limit for the quantity of positive change which can occur in one PID scan. If the Output Value (MV) exceeds this, S3+24, bit b2 is set On. | |
| S3+23 | Output Value, maximum negative change | | This is a user defined maximum limit for the quantity of negative change which can occur in one PID scan. If the Output Value (MV) falls below the lower limit, S3+24, bit b3 is set On. | |
| S3+24 | Alarm flags **(Read Only)** | b0 | High limit exceeded in Process Value (PVnf) | Not applicable |
| | | b1 | Below low limit for the Process Value (PVnf) | |
| | | b2 | Excessive positive change in Output Value (MV) | |
| | | b3 | Excessive negative change in Output Value (MV) | |
| | | b4 - 15 | Reserved | |

**P/ Proportional change: Proportional to error.** When a proportional factor is applied, it calculates the difference between the current error value and the previous error value. Proportional change is based upon how fast the process/ present value is moving closer or further away from the set-point

value not upon the actual difference between process value and set-point value. It is set in S3+3.

**I/ Integral change: Proportional to integration of error over time.** Once a proportional change has been applied to an error situation, ‘fine tuning’ the correction can be performed with the I or integral element. Initially only a small change is applied but as time increases and the error is not corrected then integral effect is increased. It is important to note how Ti actually effects how fast the total integral effect correction is applied. The smaller Ti is, the bigger effect the integral will have. It is set in S3+4. Putting value 0 to S3+4 will disable I control.

**D/ Derivative change: Proportional to rate of change of error over time.** The derivative function supplements the effects caused by the proportional response. The derivative effect is the result of a calculation involving elements Td, Ts and the calculated error. It causes the derivative to initially output a large corrective action which dissipates rapidly over time. The speed of this dissipation can be controlled by the value Td, the derivative gain. The action of Kd could be considered as a filter allowing the derivative response to be scaled between 0 to 100%. The phenomena of chasing or overcorrecting both too high and too low is most often associated with the derivative portion of the equation because of the large initial correction factor. It is set in S3+6. Setting 0 in S3+6 will disable D control.

**PID Equations**

ward $PV_{nf} > SV$

$$\Delta MV = K_P\left\{(EV_n - EV_{(n-1)}) + \frac{T_S}{T_I}EV_n + D_n\right\}$$

$$EV_n = PV_{nf} - SV$$

$$D_n = \frac{T_D}{T_S + K_D \cdot T_D}(-2PV_{nf-1} + PV_{nf} + PV_{nf-2}) + \frac{K_D \cdot T_D}{T_S + K_D \cdot T_D} \cdot D_{n-1}$$

$$MV_n = \sum \Delta MV$$

rse $SV > PV_{nf}$

$$\Delta MV = K_P\left\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I}EV_n + D_n\right\}$$

$$EV_n = SV - PV_{nf}$$

$$D_n = \frac{T_D}{T_S + K_D \cdot T_D}(2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{K_D \cdot T_D}{T_S + K_D \cdot T_D} \cdot D_{n-1}$$

$$MV_n = \sum \Delta MV\Delta$$

$PV_{nf} = PV_n + \alpha(PV_{nf-1} - PV_n)$

$EV_n$ = the current Error Value
$EV_{n-1}$ = the previous Error Value
SV = the Set Point Value ($S_1$)
$PV_n$ = the current Process Value ($S_2$)
$PV_{nf}$ = the calculated Process Value
$PV_{nf-1}$ = the previous Process Value
$PV_{nf-2}$ = the second previous Process Value
$\Delta MV$ = the change in the Output Manipulation Values
$MV_n$ = the current Output Manipulation Value (D)

$D_n$ = the Derivative Value
$D_{n-1}$ = the previous Derivative Value
$K_P$ = the Proportion Constant
$\alpha$ = the Input Filter
$T_S$ = the Sampling Time
$T_I$ = the Integral Time Constant
$T_D$ = the Time Derivative Constant
$K_D$ = the Derivative Filter Constant

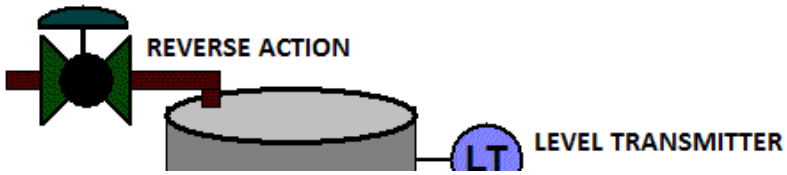
Please see the Parameter setup section for a more detailed description of the variable parameters and in which memory register they must be set.

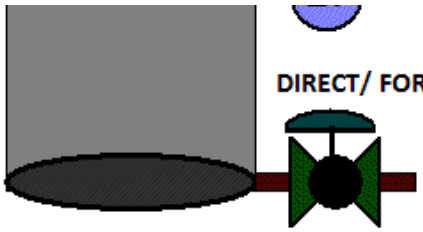
**Forward and Reverse operation ($S_3$+1, b0)**

The Forward operation is the condition where the Process Value, $PV_{nf}$, is greater than the Set Point, SV. An example is a building that requires air conditioning. Without air conditioning, the temperature of the room will be higher than the Set Point so work is required to lower $PV_{nf}$.

The Reverse operation is the condition where the Set Point is higher than the Process Value. An example of this is an oven. The temperature of the oven will be too low unless some work is done to raise it, i.e. - the heating element is turned On.

The assumption is made with PID control that some work will need to be performed to bring the system into balance. Therefore, $\Delta MV$ will always have a value. Ideally, a system that is stable will require a constant amount of work to keep the Set Point and Process Value equal.



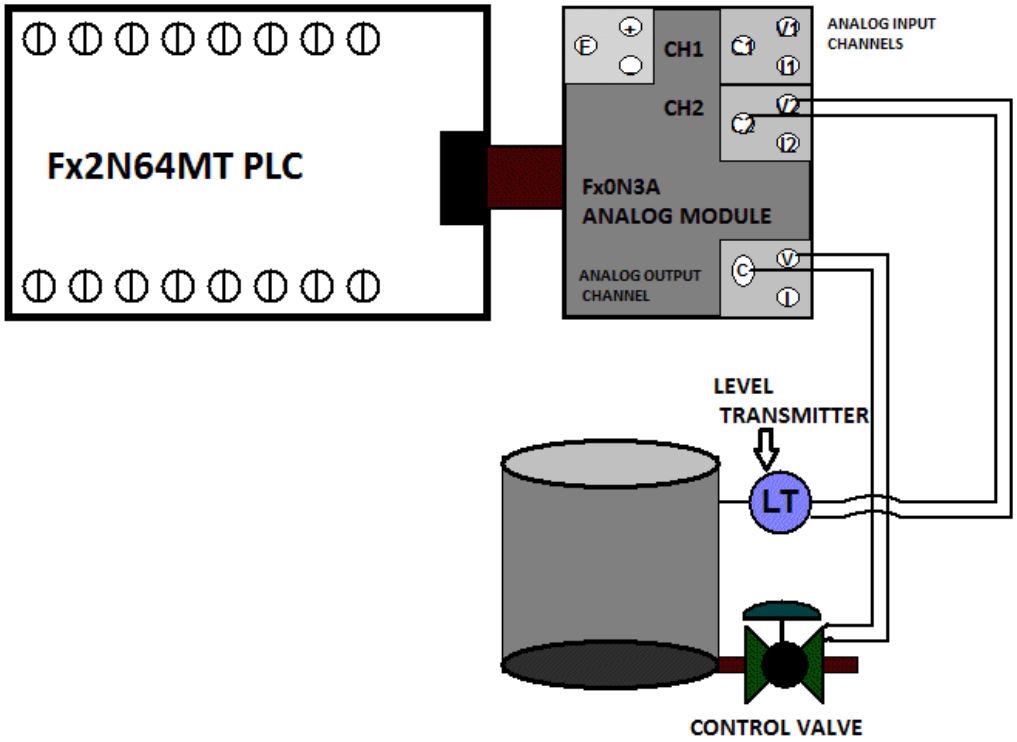



**FORWARD / DIRECT ACTION: PROCESS/ FEEDBACK/ PRESENT VALUE > SET POINT/ VALUE.**

**REVERSE ACTION: PROCESS/ FEEDBACK/ PRESENT VALUE < SET POINT/ VALUE.**

When you need to open outlet valve to maintain set point, it is forward action and when you need to close outlet valve, it is reverse action.

**EX105: PID control program**




**S1 □ SV/ SP □ D200**

**S2 □ PV □ D201**

**S3 □ PID VALUE □ D500 To D524**

**D □ D525**

**S3+0 □ D500 □ Ts □ 500 MS**

**S3+1 □ D501 □ FORWARD ACTION and ALARM DISABLED □ 0**

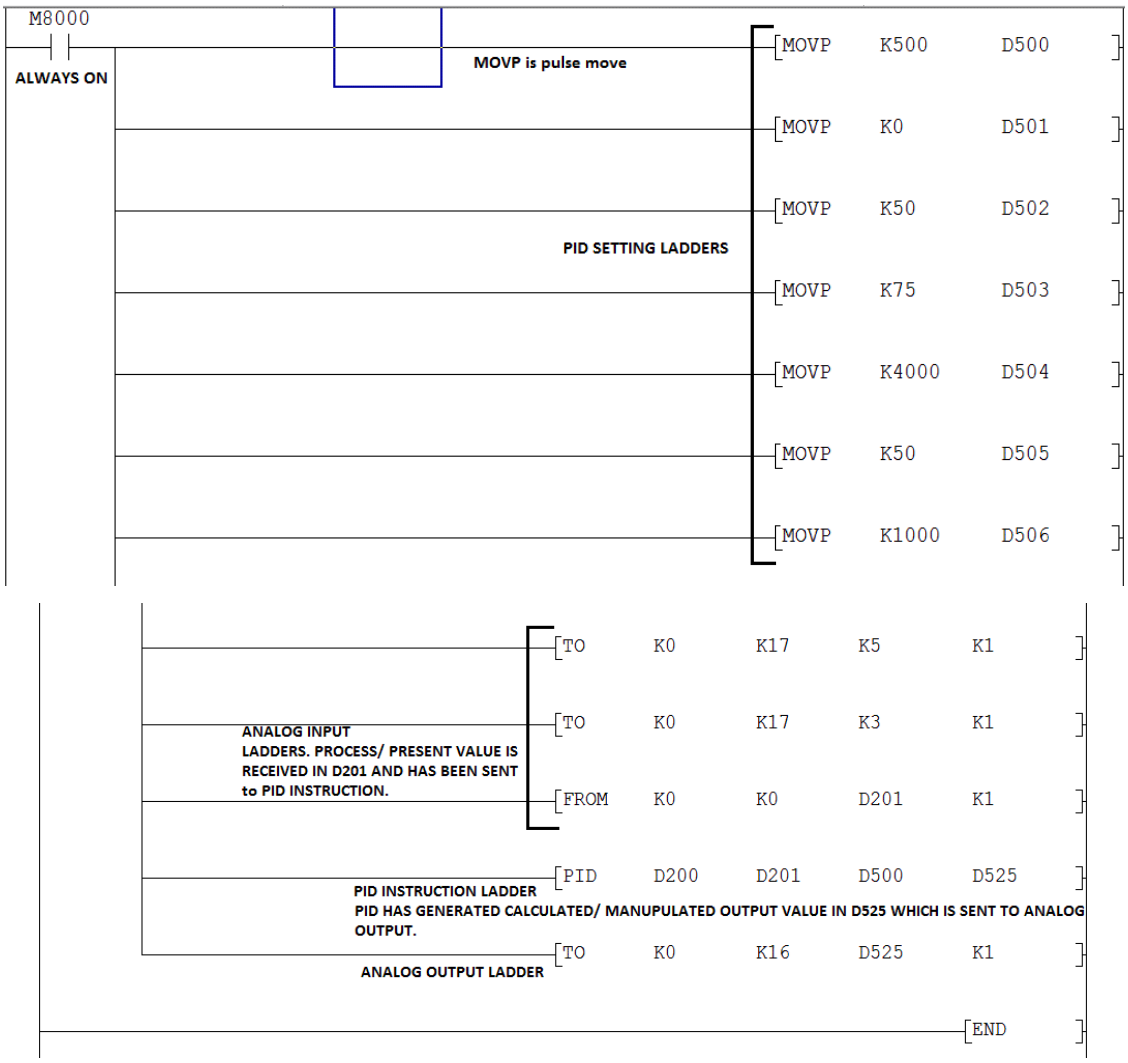
**S3+2 □ D502 □ INPUT FILTER □ 50%**

**S3+3 □ D503 □ Kp □ 75%**

**S3+4 □ D504 □ Ti □ 4000 MS (4 to 10 TIMES GREATER THAN Td)**

**S3+5 □ D505 □ Kd □ 50%**

**S3+6 □ D506 □ Td □ 1000 MS**




## PID TUNING STEPS:

1> Set all gains to 0.
2> Increase the P gain until the response to a disturbance is steady oscillation.
3> Increase the D gain until the oscillation go away i. e. it is critically damped.
4> Repeat steps 2 and 3 until increasing the D gain does not stop the oscillations.

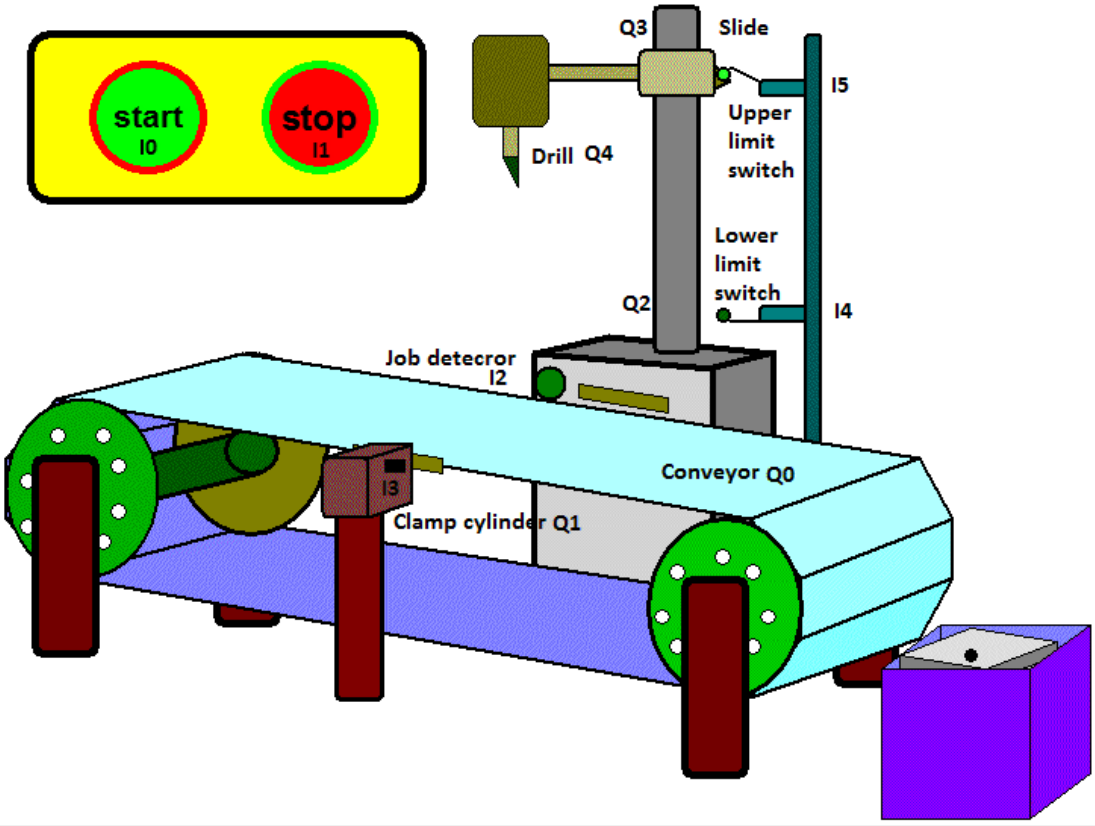
5> Set P and D to the last stable values.
6> Increase the I gain until it brings you to the set-point with the number of oscillations desired.

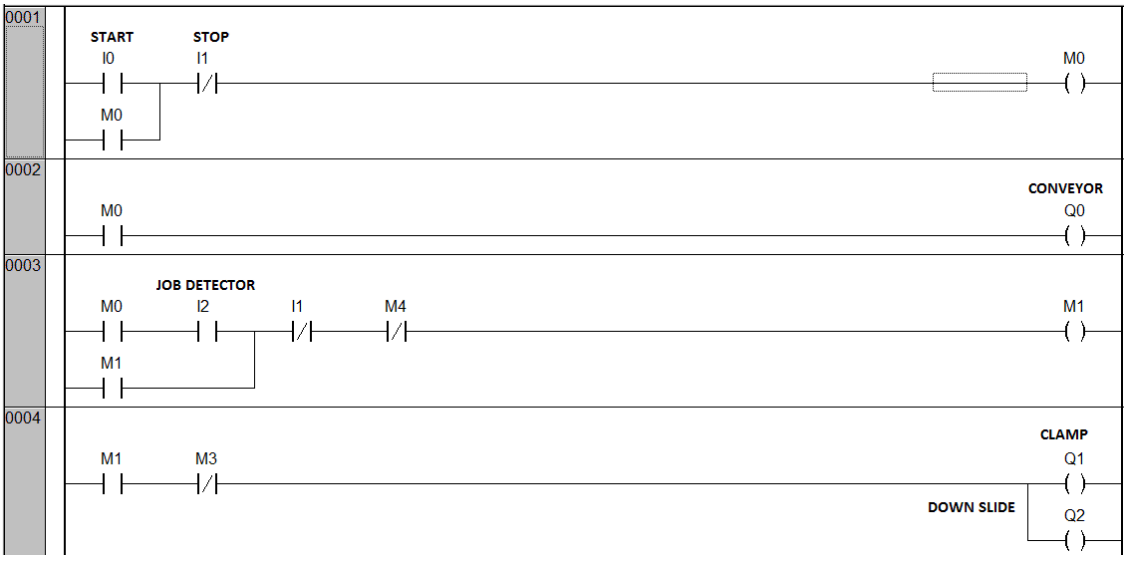
What disturbance you see depends on the mechanism the controller is attached to. Normally moving the mechanism by hand away from the set-point and letting go is enough. If the oscillations grow bigger and bigger then you need to reduce the P gain. If you set the D gain too high the system will begin to chatter.

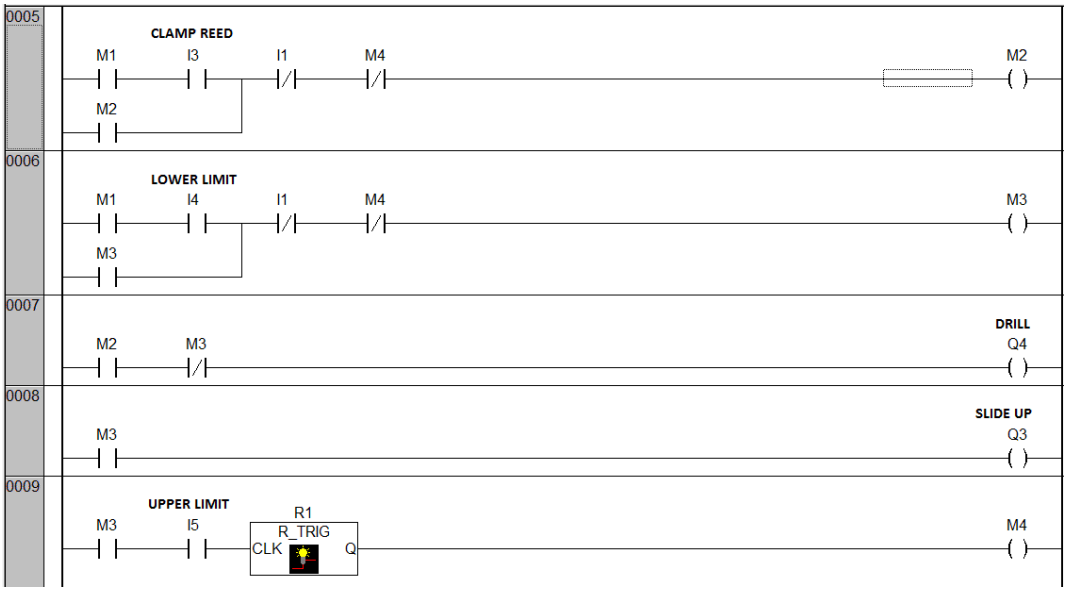
**EX106: Automatic Drill Machine.**

There is a conveyor Q0. There is a job clamp single acting pneumatic cylinder Q1. Q2 is drill system down slide. Q3 is drill system up slide. Q4 is drill. I0 is start push button. I1 is stop push button. I2 is job detector. !3 is clamp final position reed switch. I4 is lower limit switch. I5 is upper limit switch. I1 is stop push button.

As I0 is pressed, conveyor starts. Job moves on conveyor and reaches to sensor I2. As I2 gets on clamp cylinder piston Q1 moves forward and clamps the job. At the same time drill system moves down by Q2 on. As clamp reed switch is on, drill Q4 starts. AS drill system hits lower limit switch I4, drill stops, clamp moves back and drill system goes back by Q2 off and Q3 on. As drill system hits upper limit switch I5, process repeats.


start
I0
stop
I1
Q3
Slide
I5
Upper
limit
switch
Drill Q4
Lower
limit
switch
Q2
I4
Job detecror
I2
Conveyor Q0
I3
Clamp cylinder Q1


0001
START
I0
STOP
I1
M0
M0
0002
CONVEYOR
Q0
M0
0003
JOB DETECTOR
M0
I2
I1
M4
M1
M1
0004
CLAMP
Q1
M1
M3
DOWN SLIDE
Q2




**EX107: Work station selection.**

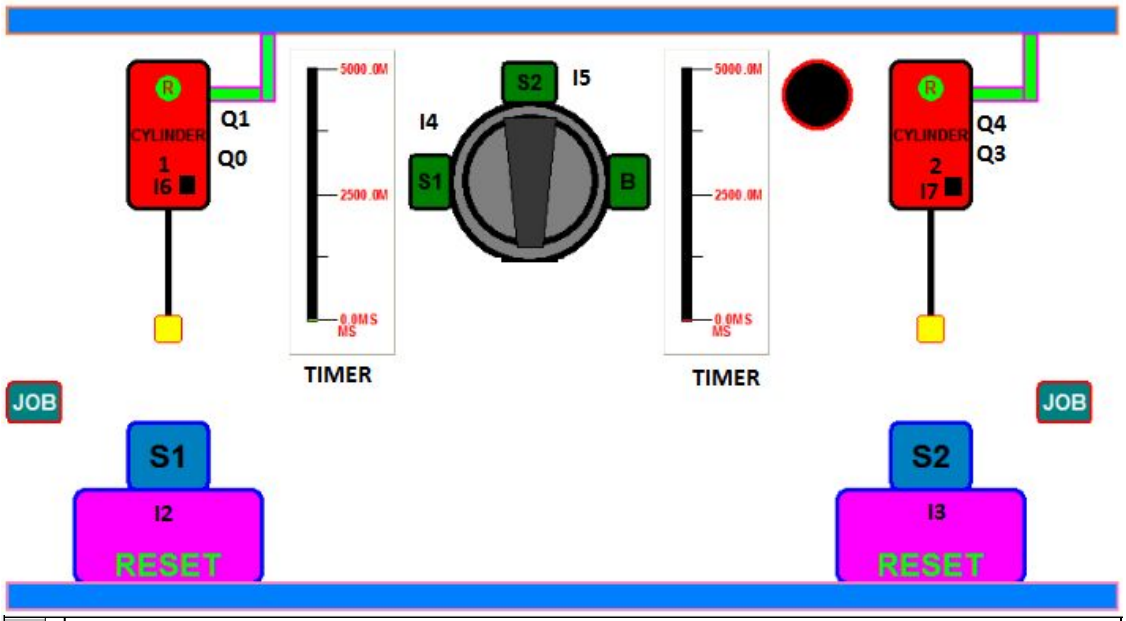
There are two work stations. Operator can select work station; station1, station2 or both. Each station has a double acting cylinder, one job finishing device and job detector.

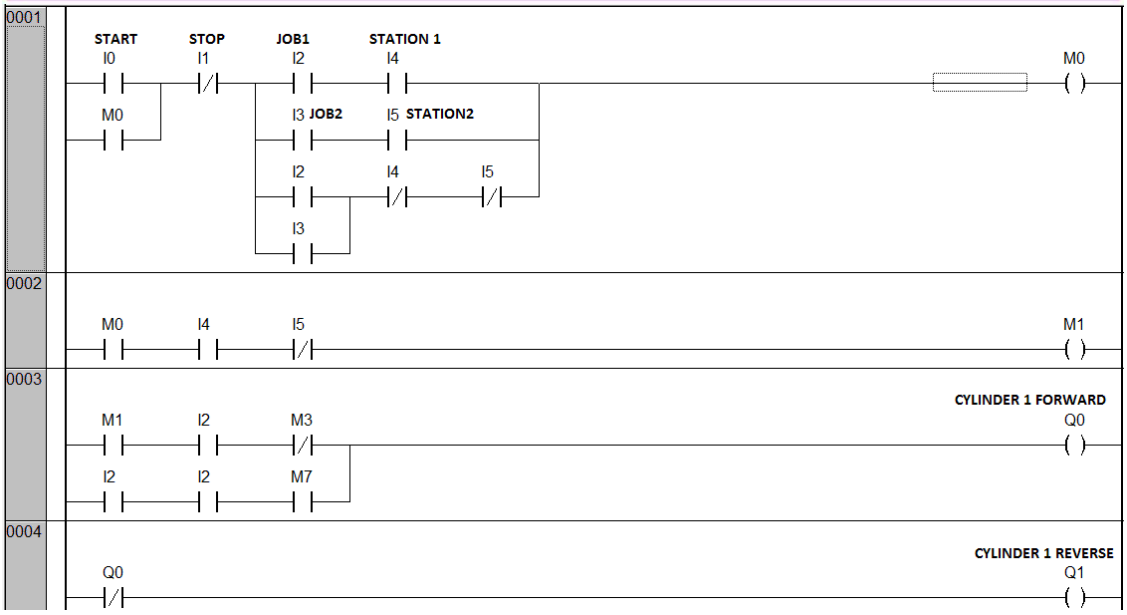
When station 1 is selected: (when selector is on toward I5) Job is put on work station 1. It is detected by sensor I2. I0 start push button is pressed. Piston moves down by Q0 on. As it reaches to final position, I6 gets on. As I6 gets on, job finisher Q2 gets on for 5 seconds. After 5 seconds Q0 and Q1 get off and Q2 gets on resulting piston move back. System does not start again until job is removed from work station. Station 2 will not function in any condition.

When station 2 is selected: (when selector is on toward I5) Job is put on work station 2. It is detected by sensor II3. I0 start push button is pressed. Piston moves down by Q3 on. As it reaches to final position, I7 gets on. As I7 gets on, job finisher Q5 gets on for 5 seconds. After 5 seconds Q3 and Q5 get off and Q4 gets on resulting piston move back. System does not start again until job is removed from work station. Station 1 will not function in any condition.

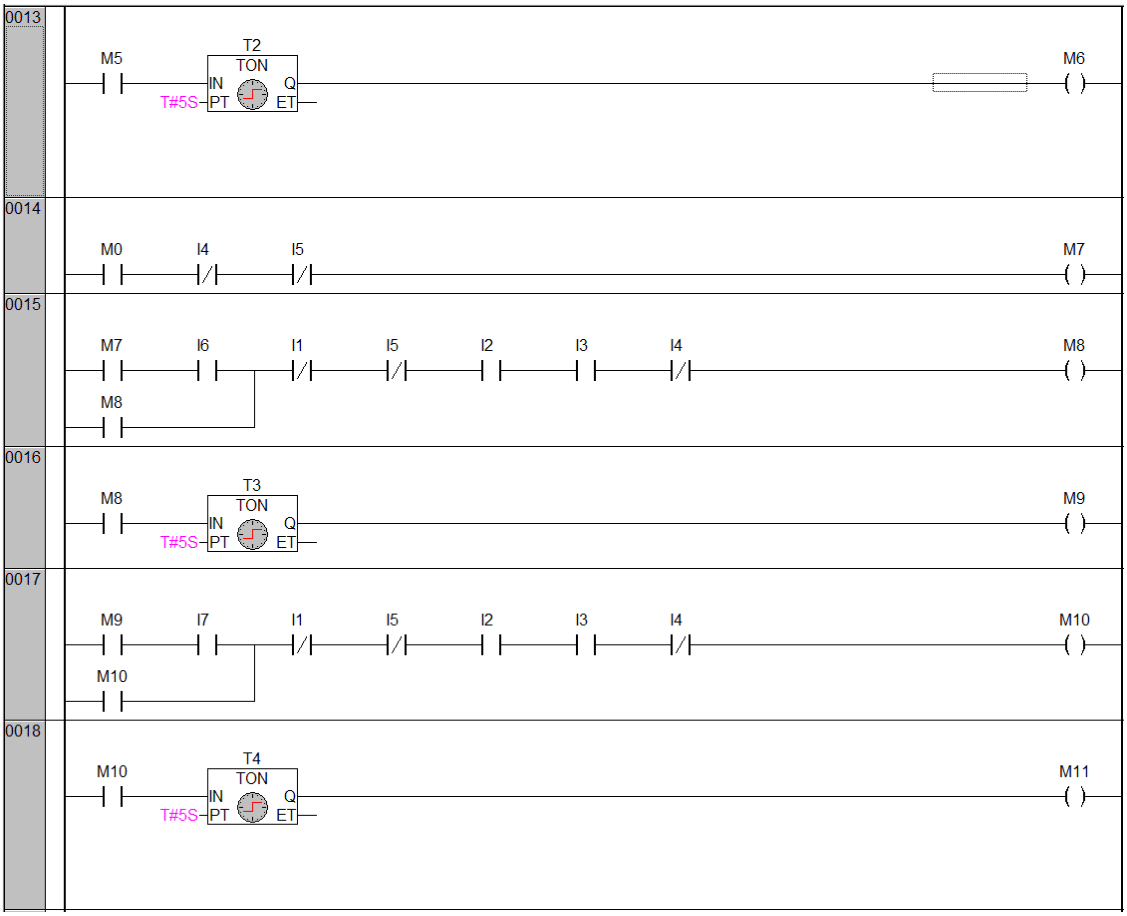
When both stations are selected: (When selector switch is off) Job is put on both work stations. It is detected by sensor I2 and sensor I3 respectively. I0 start push button is pressed. Piston 1 moves down by Q0 on. As it reaches to final position, I6 gets on. As I6 gets on, job finisher Q2 gets on for 5 seconds. After 5 seconds Q0 and Q1 get off and Q2 gets on resulting piston

move back. Station 2 starts functioning after work station 1 is finished. Piston 2 moves down by Q3 on. As it reaches to final position, I7 gets on. As I7 gets on, job finisher Q5 gets on for 5 seconds. After 5 seconds Q3 and Q5 get off and Q4 gets on resulting piston move back. System does not start again until both jobs are removed from work stations. I1 is stop push button.



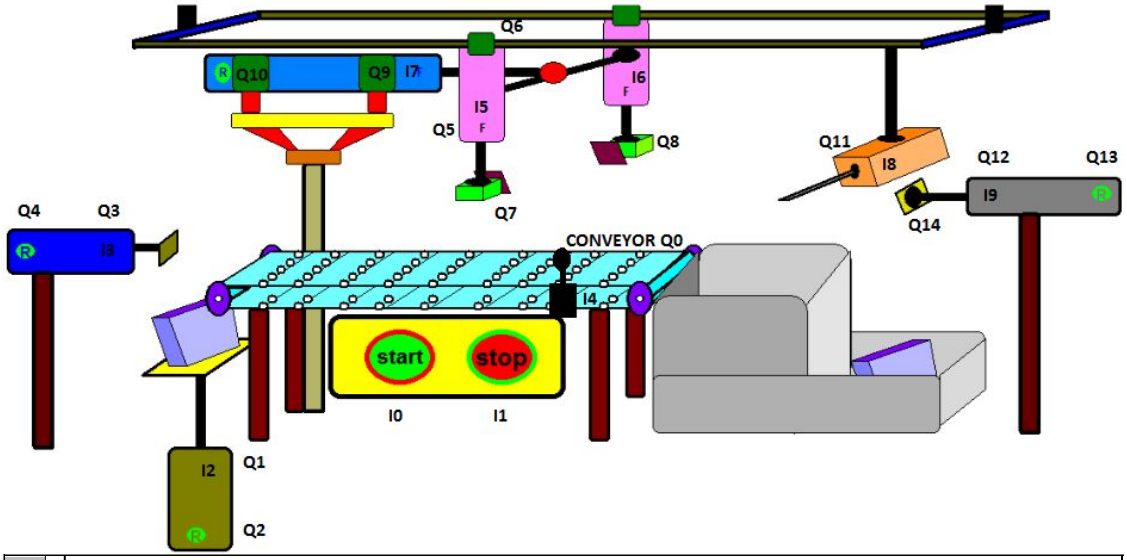



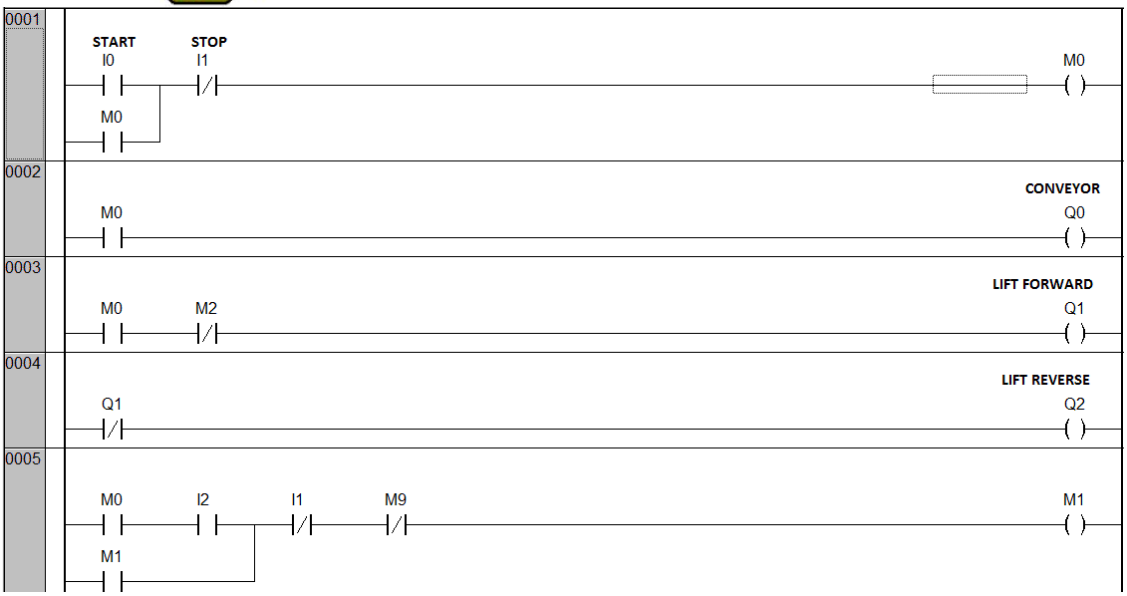

0005
CYLINDER 1 REED SWITCH
M1
I6
I1
I4
I2
M2
M2
0006
JOB FINISHER 1
M2
M3
M9
Q2
0007
M2
T1
TON
IN
PT
T#5S
Q
ET
M3
0008
M0
I5
I4
M4
0009
CYLINDER 2 FORWARD
M4
I3
M6
M11
Q3
M9
I3
I2
0010
CYLINDER 2 REVERSE
Q3
Q4
0011
CYLINDER 2 REED SWITCH
M4
I7
I1
I3
I5
M5
M5
0012
JOB FINISHER 2
M5
M6
M11
Q5
M10

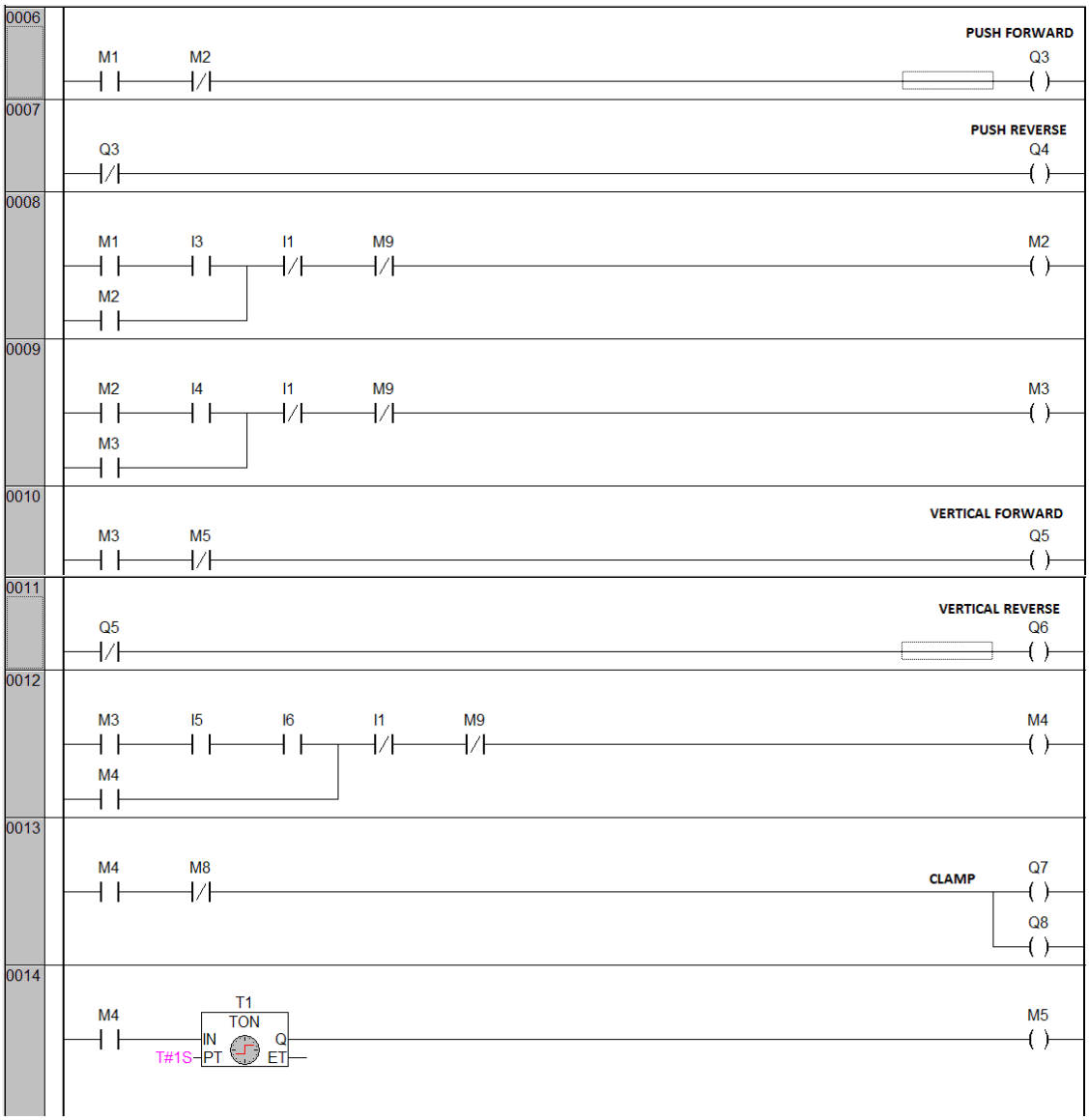


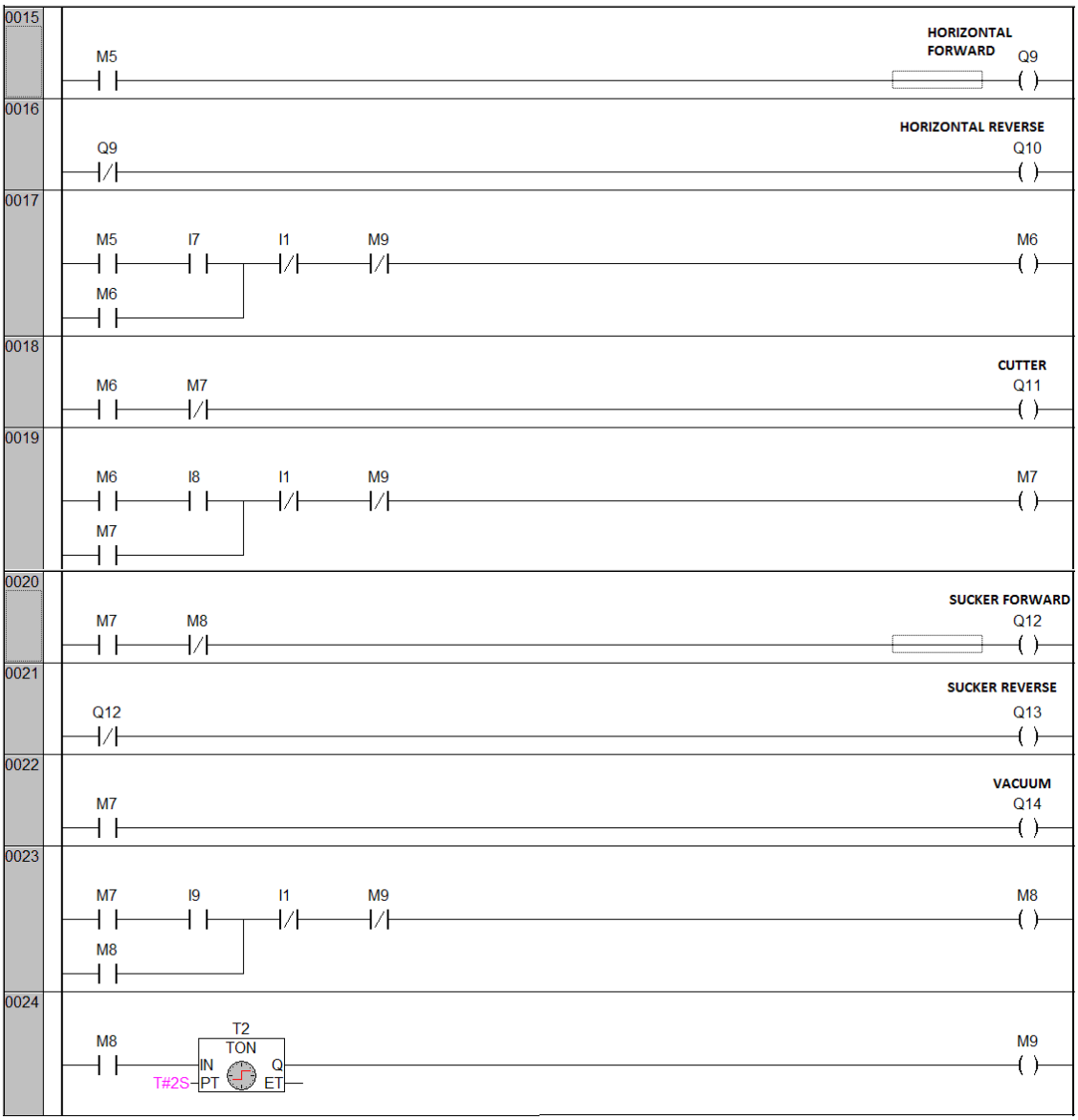

**EX108: Poison bag handling.**

When start push button I0 is pressed, conveyor Q0 will start and lifting cylinder will move up by Q1 on. As reed switch I2 is on, pusher cylinder will move forward by Q3 on and will place the poisonous bag on conveyor. As reed switch I3 gets on, lifting and pusher cylinder will get back by Q2 and Q4 on (Q1 and Q3 will get off). As bag will hit limit switch I4, vertical cylinder will move down by Q5 on. As reed switches I5 and I6 get on, clamp cylinder will hold the bag by Q7 and Q8 on. After one second of clamp on, vertical cylinder will move back by Q5 off and Q6 on. At the same time horizontal cylinder will move forward by Q9 on. As reed witch I7 gets on, cutting blade cylinder will move forward by Q11 on. As reed switch I8 gets on, cutter will move back by Q11 off. At the same time vacuum sucker will move forward by Q12 on. As reed switch I9 gets on, vacuum Q14 gets on and sucker cylinder moves back by Q12 off and Q13 on. After 2 seconds process repeats. I1 is stop button.


Q6
Q10
Q9
I7
I5
F
Q5
I6
F
Q8
Q7
Q11
I8
Q12
Q13
I9
Q14
Q4
Q3
I3
CONVEYOR Q0
I4
start
stop
I0
I1
I2
Q1
Q2

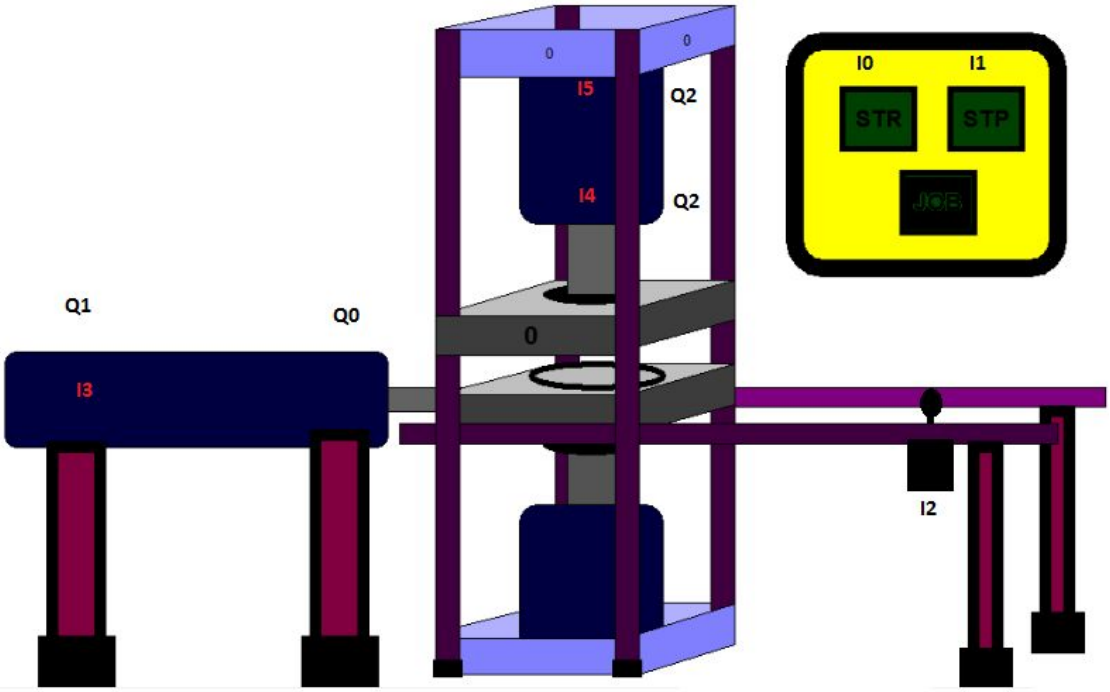

0001
START
I0
STOP
I1
M0
M0
0002
CONVEYOR
M0
Q0
0003
LIFT FORWARD
M0
M2
Q1
0004
LIFT REVERSE
Q1
Q2
0005
M0
I2
I1
M9
M1
M1

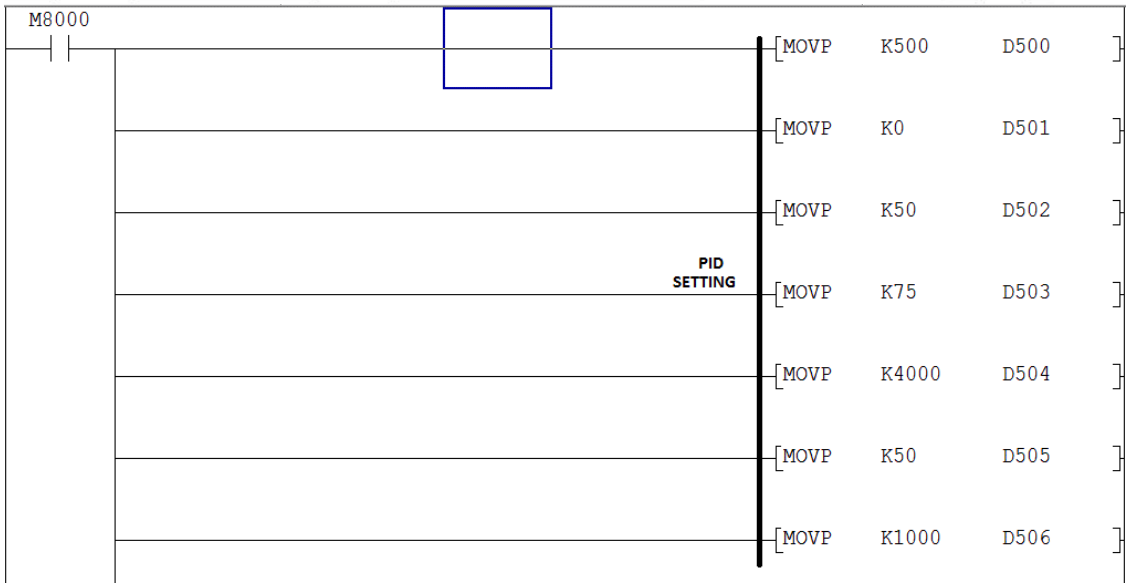

0006
PUSH FORWARD
M1
M2
Q3
0007
PUSH REVERSE
Q3
Q4
0008
M1
I3
I1
M9
M2
M2
0009
M2
I4
I1
M9
M3
M3
0010
VERTICAL FORWARD
M3
M5
Q5
0011
VERTICAL REVERSE
Q5
Q6
0012
M3
I5
I6
I1
M9
M4
M4
0013
M4
M8
CLAMP
Q7
Q8
0014
M4
T1
TON
IN
PT
Q
ET
T#1S
M5

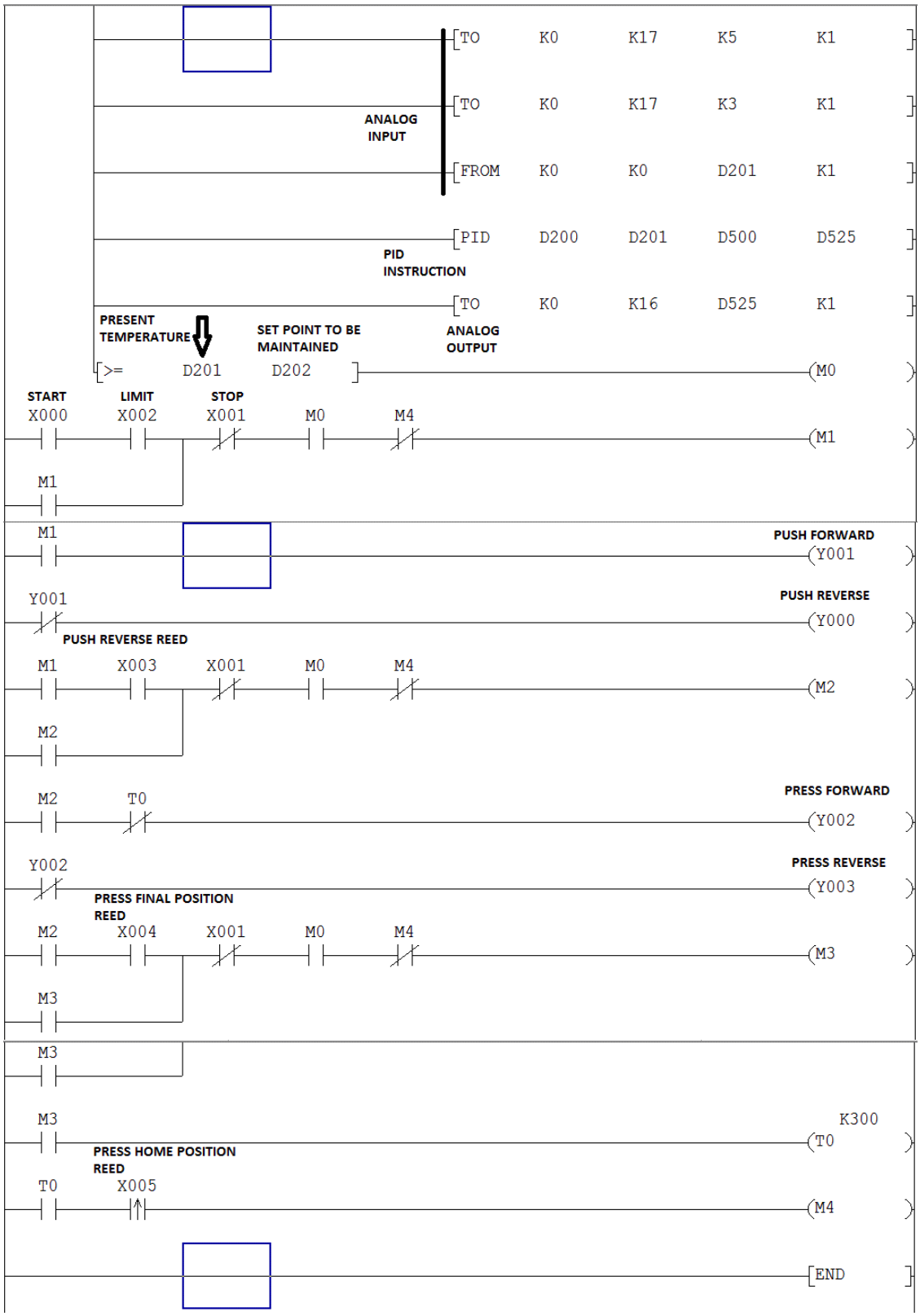



## EX109: Moulding Machine.

Initially mould pusher will be in final position. It means initially Q0 will be on and limit switch I2 will be on. Now operator will put the material on mould and will press start button I0 resulting mould moving reverse by Q0 off and Q1 on. As reverse reed switch I3 gets on, upper mould will move down by Q2 on. As reed switch I4 gets on, mould will wait for 30 seconds. After 30 seconds upper mould will move back by Q2 off and Q3 on. As reed switch I5 gets on, one cycle will be finished. I1 is stop button. Mould temperature will be maintained to some set point set from HMI/ SCADA. PID will be used to maintain temperature. Without required temperature machine will not start.


I0
I1
STR
STP
JOB
I5
Q2
I4
Q2
Q1
Q0
0
I3
I2


M8000
MOVP K500 D500
MOVP K0 D501
MOVP K50 D502
PID SETTING
MOVP K75 D503
MOVP K4000 D504
MOVP K50 D505
MOVP K1000 D506


TO K0 K17 K5 K1
TO K0 K17 K3 K1
ANALOG INPUT
FROM K0 K0 D201 K1
PID D200 D201 D500 D525
PID INSTRUCTION
TO K0 K16 D525 K1
PRESENT TEMPERATURE
SET POINT TO BE MAINTAINED
ANALOG OUTPUT
>= D201 D202 M0
START X000
LIMIT X002
STOP X001
M0 M4 M1
M1
M1
PUSH FORWARD Y001
Y001
PUSH REVERSE Y000
PUSH REVERSE REED
M1 X003 X001 M0 M4 M2
M2
M2 T0
PRESS FORWARD Y002
Y002
PRESS REVERSE Y003
PRESS FINAL POSITION REED
M2 X004 X001 M0 M4 M3
M3
M3
M3
K300 T0
PRESS HOME POSITION REED
T0 X005 M4
END

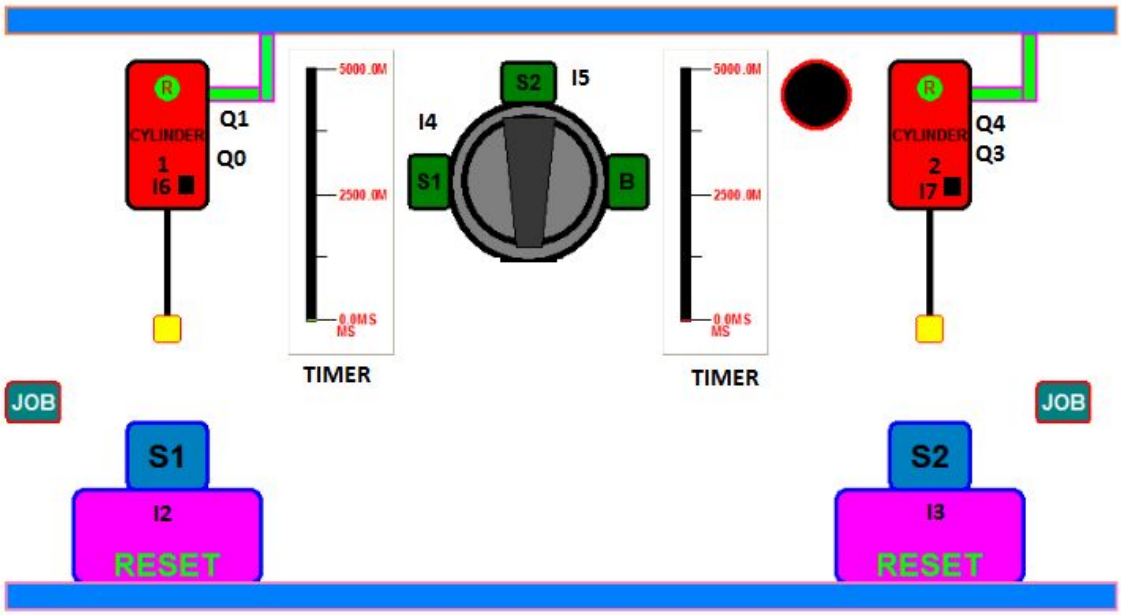
**EX110: Work station selection using subroutine.**

There are two work stations. Operator can select work station; station1, station2 or both. Each station has a double acting cylinder, one job finishing device and job detector.

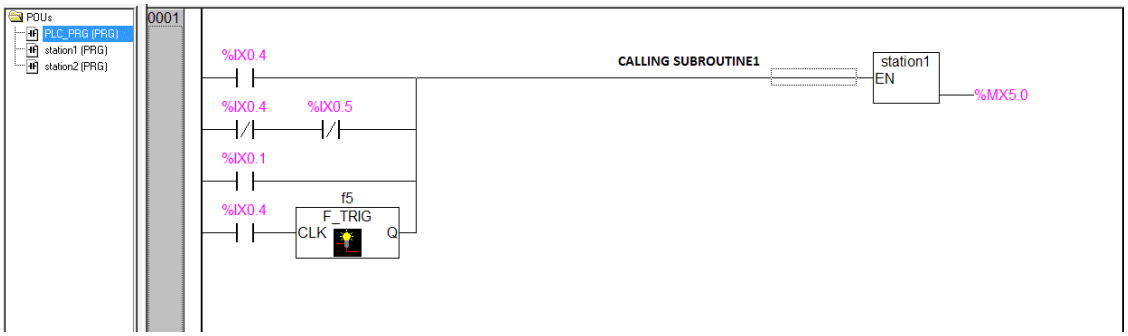
When station 1 is selected: (when selector is on toward I5) Job is put on work station 1. It is detected by sensor I2. I0 start push button is pressed. Piston moves down by Q0 on. As it reaches to final position, I6 gets on. As I6 gets on, job finisher Q2 gets on for 5 seconds. After 5 seconds Q0 and Q1 get off and Q2 gets on resulting piston move back. System does not start again until job is removed from work station. Station 2 will not function in any condition.

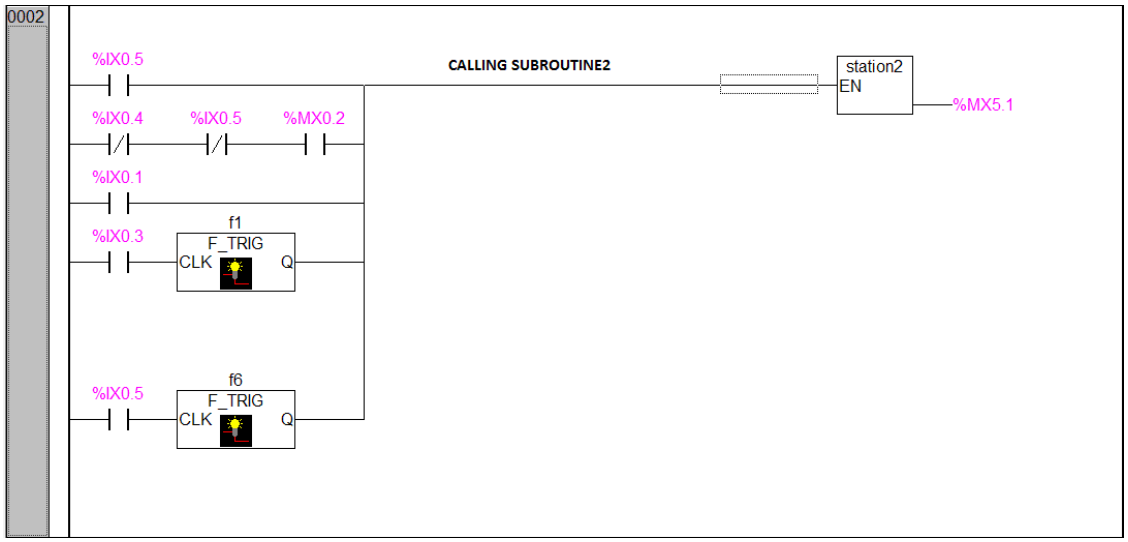
When station 2 is selected: (when selector is on toward I5) Job is put on work station 2. It is detected by sensor II3. I0 start push button is pressed. Piston moves down by Q3 on. As it reaches to final position, I7 gets on. As I7 gets on, job finisher Q5 gets on for 5 seconds. After 5 seconds Q3 and Q5 get off and Q4 gets on resulting piston move back. System does not start again until job is removed from work station. Station 1 will not function in any condition.

When both stations are selected: (When selector switch is off) Job is put on both work stations. It is detected by sensor I2 and sensor I3 respectively. I0 start push button is pressed. Piston 1 moves down by Q0 on. As it reaches to final position, I6 gets on. As I6 gets on, job finisher Q2 gets on for 5 seconds. After 5 seconds Q0 and Q1 get off and Q2 gets on resulting piston move back. Station 2 starts functioning after work station 1 is finished. Piston 2 moves down by Q3 on. As it reaches to final position, I7 gets on. As I7 gets on, job finisher Q5 gets on for 5 seconds. After 5 seconds Q3 and Q5 get off and Q4 gets on resulting piston move back. System does not start again until both jobs are removed from work stations. I1 is stop push button.

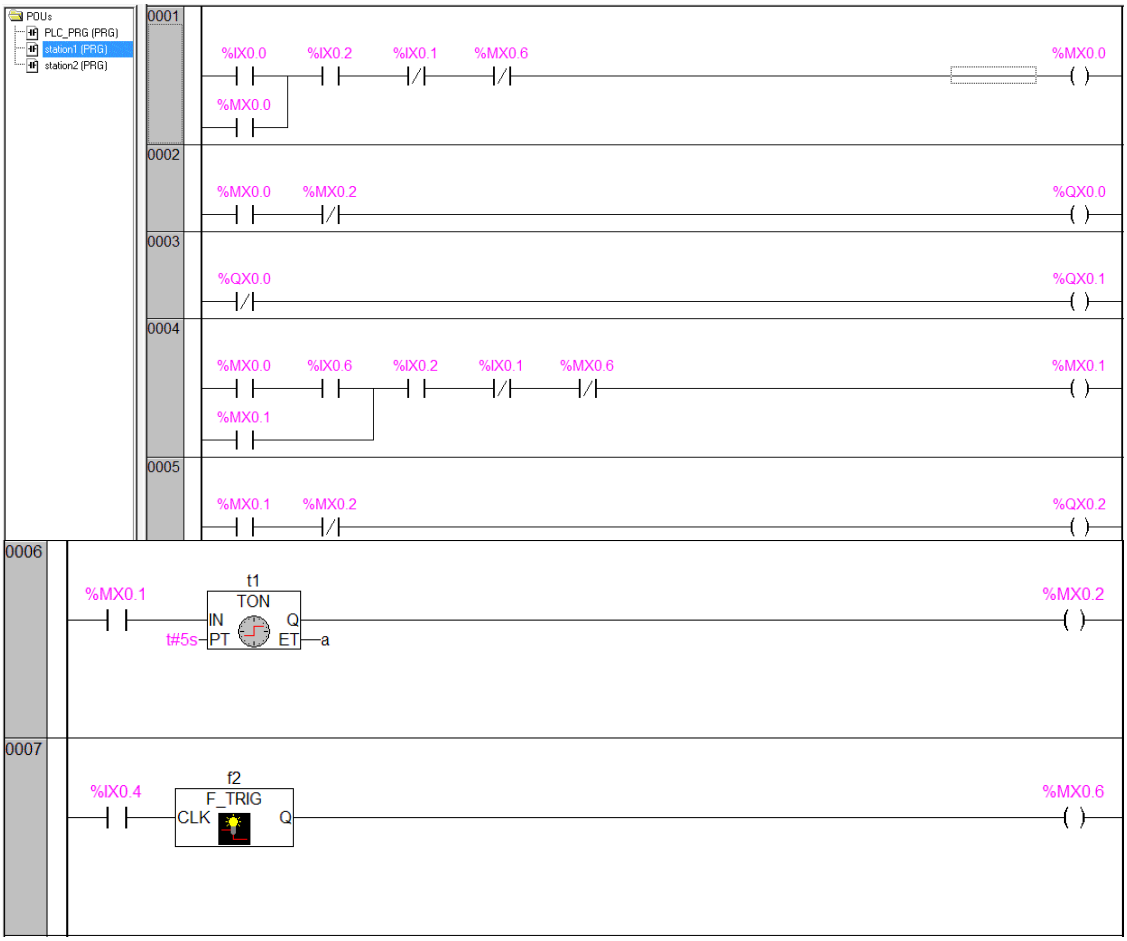



## MAIN PROGRAM

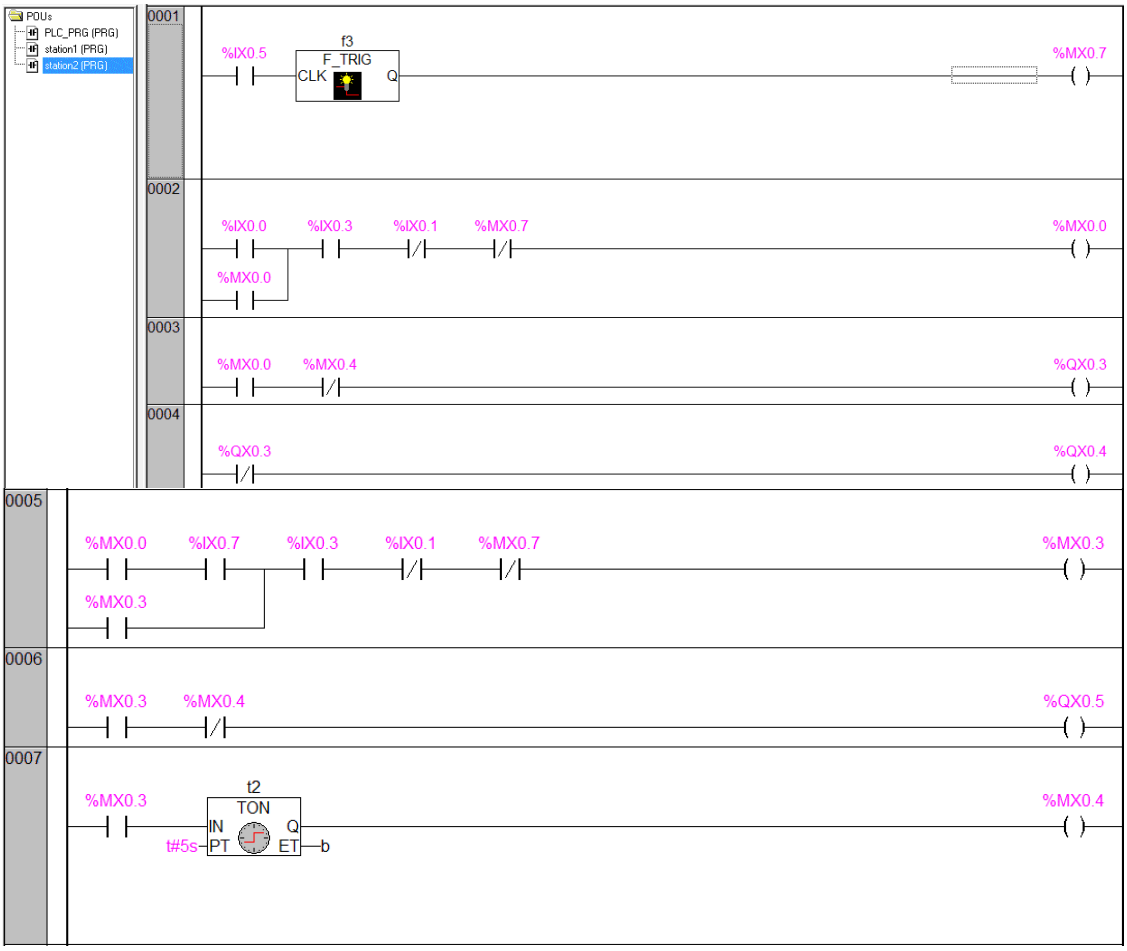


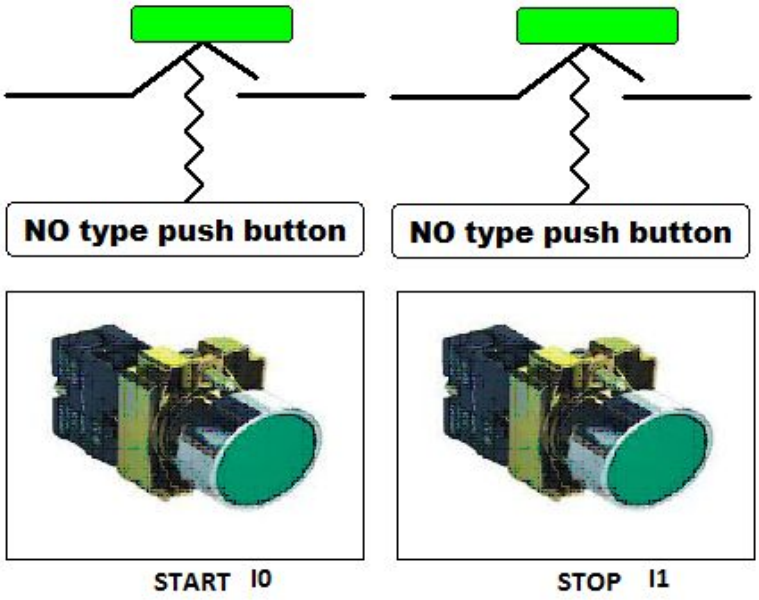

## SUBROUTINE1

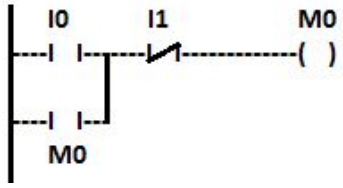



## SUBROUTINE2

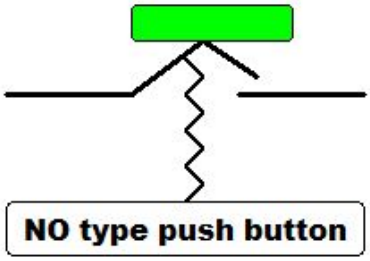



**NO & NC push button programming concept. In all our programs we have used NO type push buttons for start and stop.**

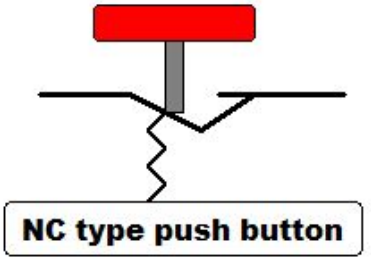



If push button type changes then you should change the contact in program too.

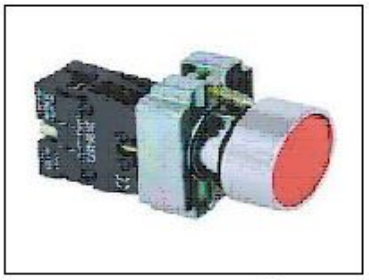


NO type push button

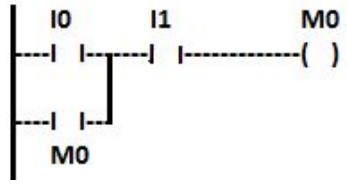


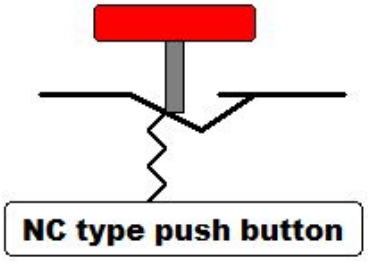
NC type push button

START I0

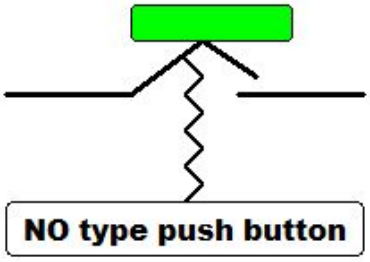


STOP I1

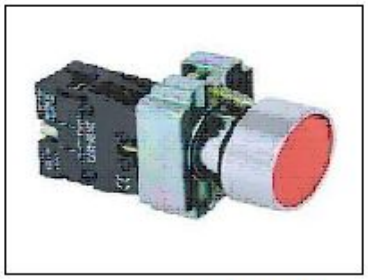


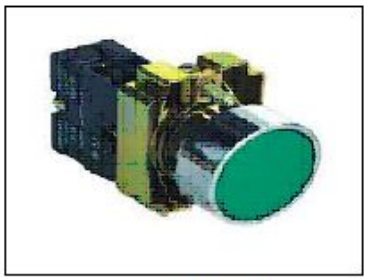
NC type push button



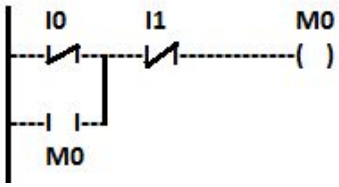
NO type push button

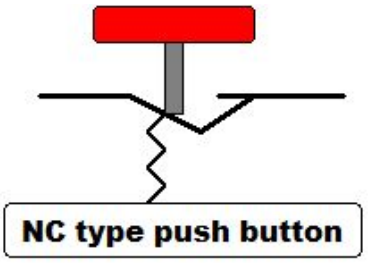


START I0

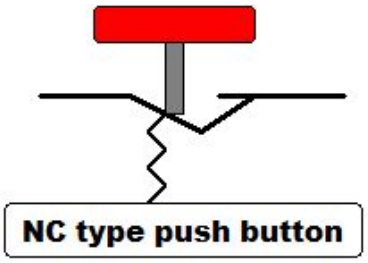


STOP I1

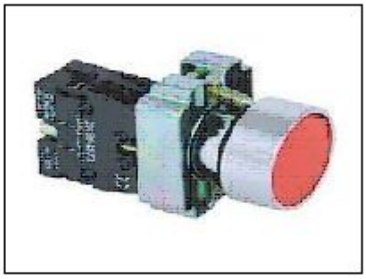




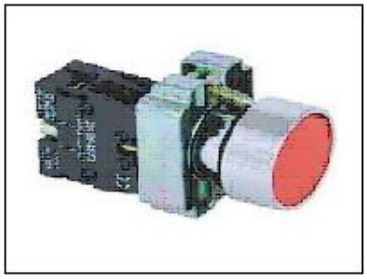
NC type push button



NC type push button



START I0



STOP I1