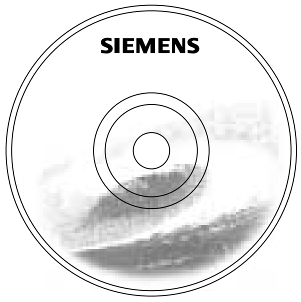
Programming A PLC

STEP 7-Micro/WIN32 is the program software used with the S7-200 PLC to create the PLC operating program. STEP 7 consists of a number of instructions that must be arranged in a logical order to obtain the desired PLC operation. These instructions are divided into three groups: standard instructions, special instructions, and high-speed instructions.



Standard Instructions

Standard instructions consists of instructions that are found in most programs. Standard instructions include; timer, counter, math, logical, increment/decrement/invert, move, and block instructions.

Special Instructions

Special instructions are used to manipulate data. Special instructions include shift, table, find, conversion, for/next, and real-time instructions.

High-Speed Instructions

High-speed instructions allow for events and interrupts to occur independent of the PLC scan time. These include high-speed counters, interrupts, output, and transmit instructions.

It is not the purpose of this text to explain all of the instructions and capabilities. A few of the more common instructions necessary for a basic understanding of PLC operation will be discussed. PLC operation is limited only by the hardware capabilities and the ingenuity of the person programming it. Refer to the **SIMATIC S7-200 Programmable Controller System Manual** for detailed information concerning these instructions.

Micro/WIN32

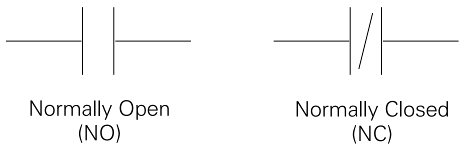
The programming software can be run Off-line or On-line. Off-line programming allows the user to edit the ladder diagram and perform a number of maintenance tasks. The PLC does not need to be connected to the programming device in this mode. On-line programming requires the PLC to be connected to the programming device. In this mode program changes are downloaded to the PLC. In addition, status of the input/output elements can be monitored. The CPU can be started, stopped, or reset.

Symbols

In order to understand the instructions a PLC is to carry out, an understanding of the language is necessary. The language of PLC ladder logic consists of a commonly used set of symbols that represent control components and instructions.

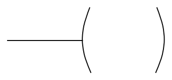
Contacts

One of the most confusing aspects of PLC programming for first-time users is the relationship between the device that controls a status bit and the programming function that uses a status bit. Two of the most common programming functions are the normally open (NO) contact and the normally closed (NC) contact. Symbolically, power flows through these contacts when they are closed. The normally open contact (NO) is true (closed) when the input or output status bit controlling the contact is 1. The normally closed contact (NC) is true (closed) when the input or output status bit controlling the contact is 0.



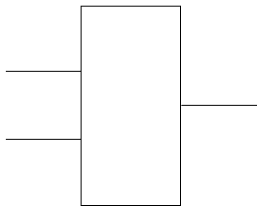
Coils

Coils represent relays that are energized when power flows to them. When a coil is energized, it causes a corresponding output to turn on by changing the state of the status bit controlling that output to 1. That same output status bit may be used to control normally open and normally closed contacts elsewhere in the program.



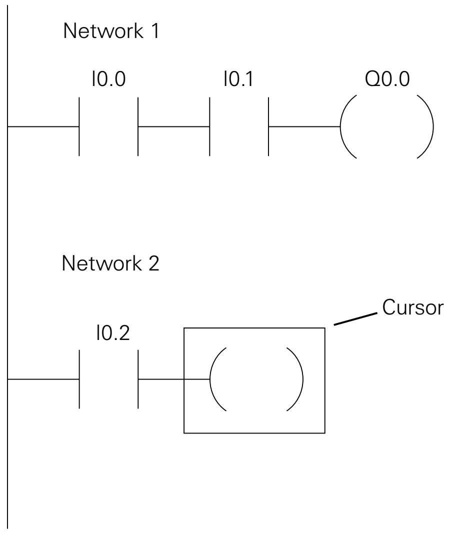
Boxes

Boxes represent various instructions or functions that are executed when power flows to the box. Typical box functions are timers, counters, and math operations.



Entering Elements

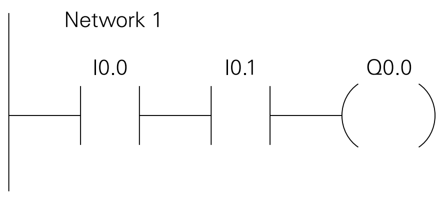
Control elements are entered in the ladder diagram by positioning the cursor and selecting the element from a lists. In the following example the cursor has been placed in the position to the right of I0.2. A coil was selected from a pull-down list and inserted in this position.



An AND Operation

Each rung or network on a ladder represents a logic operation. The following programming example demonstrates an AND operation. Two contact closures and one output coil are placed on network 1. They were assigned addresses I0.0, I0.1, and Q0.0. Note that in the statement list a new logic operation always begins with a load instruction (LD). In this example I0.0 (input 1) and (A in the statement list) I0.1 (input 2) must be true in order for output Q0.0 (output 1) to be true. It can also be seen That I0.0 and I0.1 must be true for Q0.0 to be true by looking at the function block diagram representation.

Ladder Diagram Representation

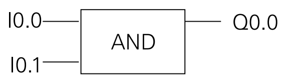


Statement List Representation

```
Network 1
LD        I0.0
A         I0.1
=         Q0.0
```

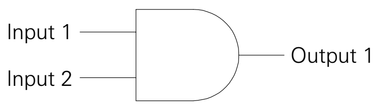
Function Block Diagram Representation

Network 1



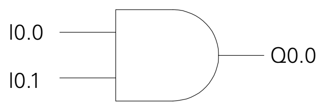
Another way to see how an AND function works is with a Boolean logic diagram. In Boolean logic an AND gate is represented by a number of inputs on the left side. In this case there are two inputs. The output is represented on the right side. It can be seen from the table that both inputs must be a logic 1 in order for the output to be a logic 1.

And (A) Function



Input 1	Input 2	Output 1
0	0	0
0	1	0
1	0	0
1	1	1

And (A) Function

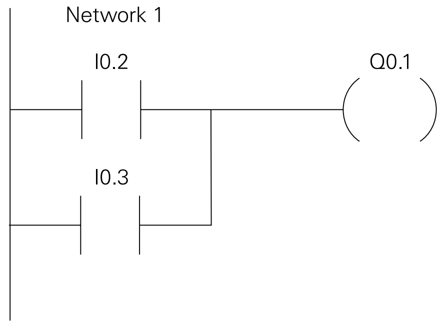


I0.0	I0.1	Q0.0
0	0	0
0	1	0
1	0	0
1	1	1

An OR Operation

In this example an OR operation is used in network 1. It can be seen that if either input I0.2 (input 3) or (O in the statement list) input I0.3 (input 4), or both are true, then output Q0.1 (output 2) will be true.

Ladder Diagram Representation

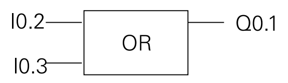


Statement List Representation

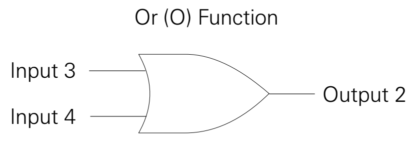
```
Network 1
LD      I0.2
O       I0.3
=       Q0.1
```

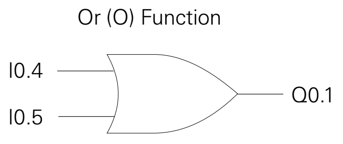
Function Block Diagram Representation

Network 1



Another way to see how an OR function works is with a Boolean logic diagram. The symbol differs slightly from an AND function. The OR function is represented by a number of inputs on the left side. In this case there are two inputs. The output is represented on the right side. It can be seen from the table that any input can be a logic 1 in order for the output to be a logic 1.



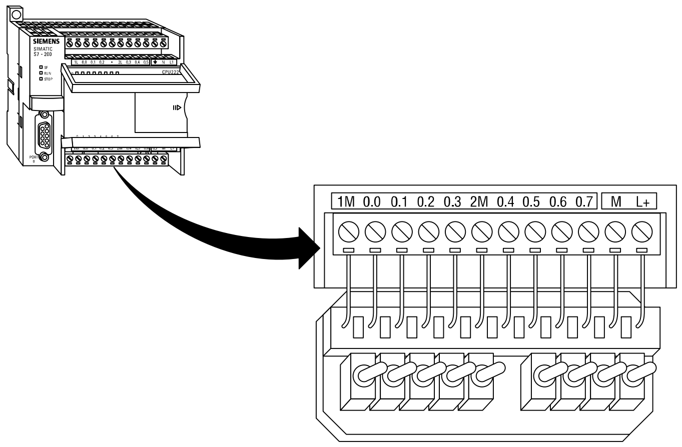


Input 3	Input 4	Output 2
0	0	0
0	1	1
1	0	1
1	1	1

I0.4	I0.5	Q0.1
0	0	0
0	1	1
1	0	1
1	1	1

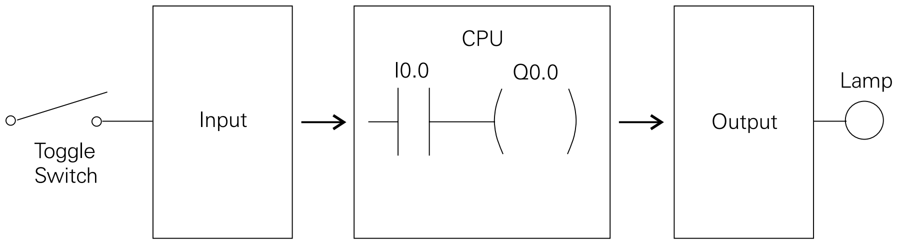
Testing a Program

Once a program has been written it needs to be tested and debugged. One way this can be done is to simulate the field inputs with an input simulator, such as the one made for the S7-200. The program is first downloaded from the programming device to the CPU. The selector switch is placed in the RUN position. The simulator switches are operated and the resulting indication is observed on the output status indicator lamps.

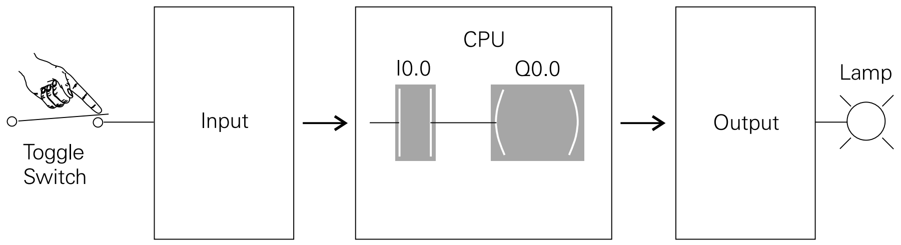


Status Functions

After a program has been loaded and is running in the PLC, the actual status of ladder elements can be monitored using STEP 7 Micro/WIN32 software. The standard method of showing a ladder element is by indicating the circuit condition it produces when the device is in the deenergized or nonoperated state. In the following illustration input 1 (I0.0) is programmed as a normally open (NO) contact. In this condition, power will not flow through the contacts to the output (Q0.0).

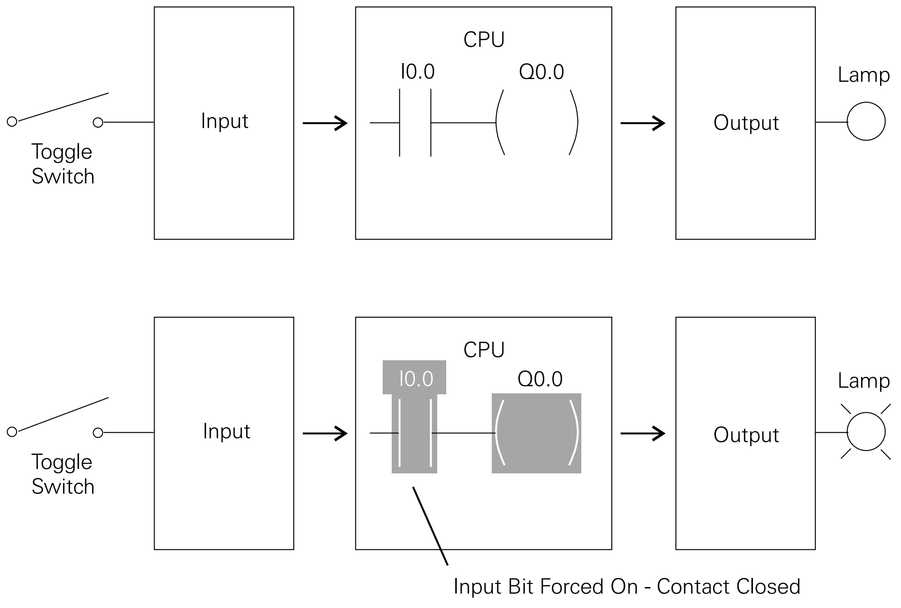


When viewing the ladder diagram in the status mode, control elements that are active, or true (logic 1), are highlighted. In the example shown the toggle switch connected to input 1 has been closed. Power can now flow through the control element associated with input 1 (I0.0) and activate the output (Q0.0). The lamp will illuminate.

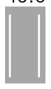
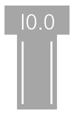
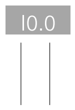
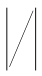
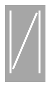
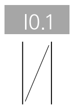
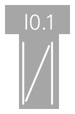
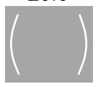
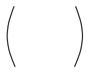
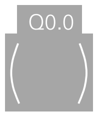
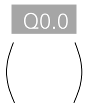


Forcing

Forcing is another useful tool in the commissioning of an application. It can be used to temporarily override the input or output status of the application in order to test and debug the program. The force function can also be used to override discrete output points. The force function can be used to skip portions of a program by enabling a jump instruction with a forced memory bit. Under normal circumstances the toggle switch, shown in the illustration below, would have to be closed to enable input 1 (I0.0) and turn on the output light. Forcing enables input 1 even though the input toggle switch is open. With input 1 forced high the output light will illuminate. When a function is forced the control bit identifier is highlighted. The element is also highlighted because it is on.

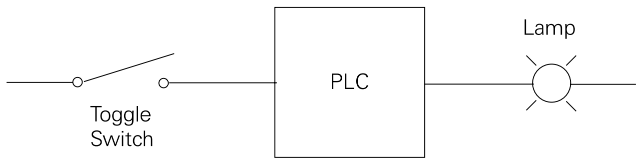


The following table shows the appearance of ladder elements in the Off, forced, and On condition.

Normally Open Contacts	Status Bit On Contacts Closed I0.0 	Status Bit Off Contacts Open I0.0	Status Bit Forced On Contacts Closed I0.0 	Status Bit Forced Off Contacts Open I0.0 
Normally Closed Contacts	Status Bit On Contacts Open I0.1 	Status Bit Off Contacts Closed I0.1 	Status Bit Forced On Contacts Open I0.1 	Status Bit Forced Off Contacts Closed I0.1 
Output Coils	Status Bit On Coil Energized Q0.0 	Status Bit Off Coil Deenergized Q0.0 	Status Bit Forced On Coil Energized Q0.0 	Status Bit Forced Off Coil Deenergized Q0.0 

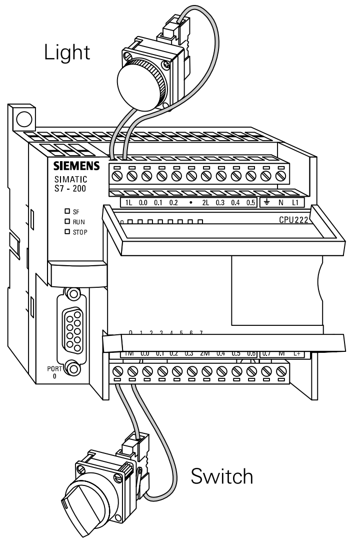
Discrete Inputs/Outputs

To understand discrete control of a programmable controller the same simple lamp circuit illustrated with forcing will be used. This is only for instructional purposes as a circuit this simple would not require a programmable controller. In this example the lamp is off when the switch is open and on when the switch is closed.

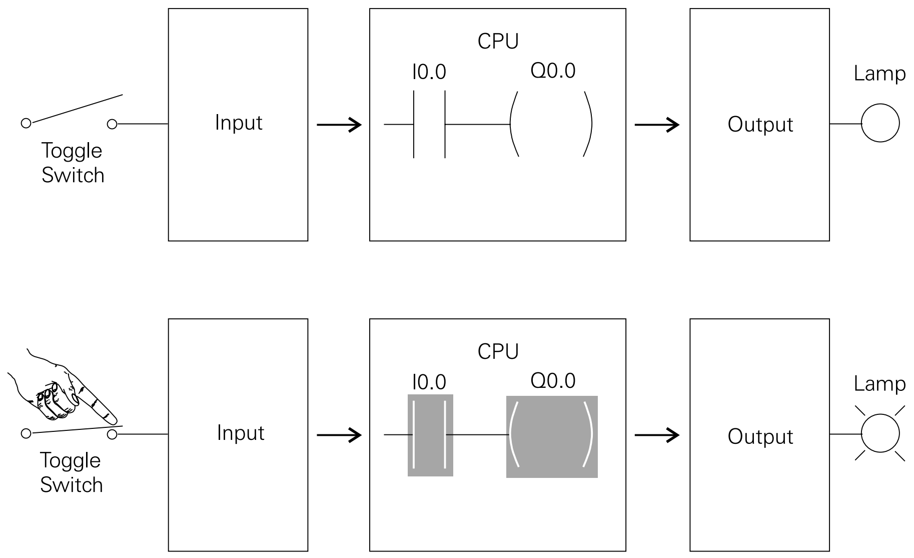


Wiring

To accomplish this task, a switch is wired to the input of the PLC and an indicator light is wired to output terminal.



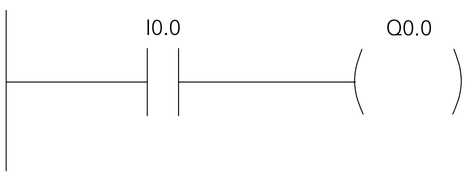
The following drawing illustrates the sequence of events. A switch is wired to the input module of the PLC. A lamp is wired to the output module. The program is in the CPU. The CPU scans the inputs. When it finds the switch open I0.0 receives a binary 0. This instructs Q0.0 to send a binary 0 to the output module. The lamp is off. When it finds the switch closed I0.0 receives a binary 1. This instructs Q0.0 to send a binary 1 to the output module, turning on the lamp.



Program Instruction

When the switch is open the CPU receives a logic 0 from input I0.0. The CPU sends a logic 0 to output Q0.0 and the light is off.

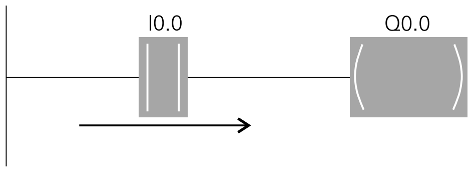
Network 1



```
LD      I0.0
=       Q0.0
```

When the switch is closed the CPU receives a logic 1 from input I0.0. The CPU sends a logic 1 to output Q0.0, thus activating Q0.0. The light turns on.

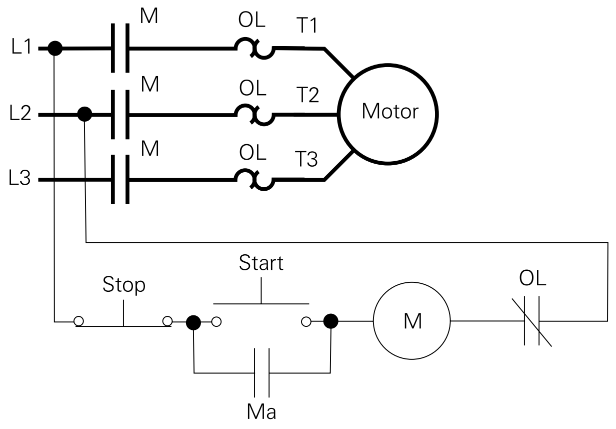
Network 1



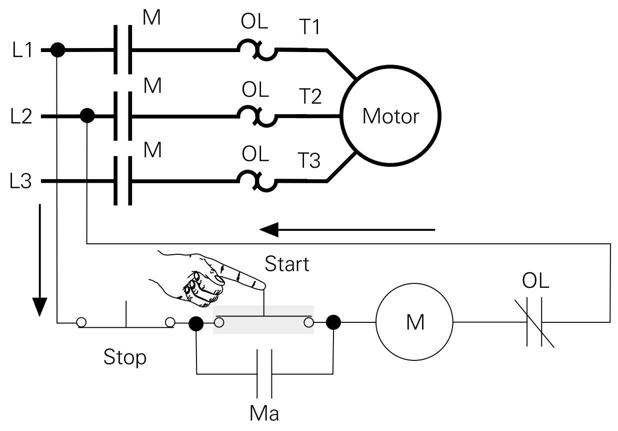
```
LD      I0.0
=       Q0.0
```

Motor Starter Example

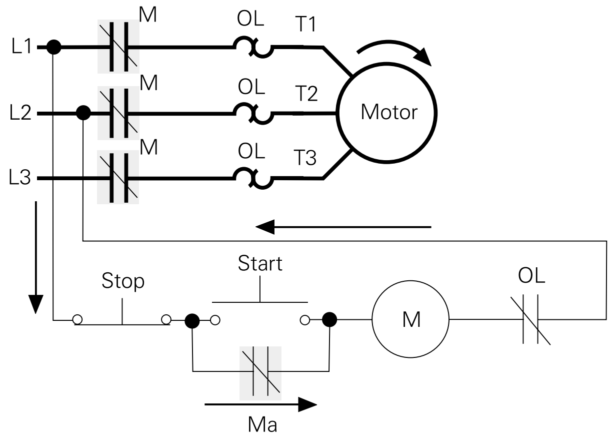
The following example involves a motor start and stop circuit. The line diagram illustrates how a normally open and a normally closed pushbutton might be used in a control circuit. In this example a motor started (M) is wired in series with a normally open momentary pushbutton (Start), a normally closed momentary pushbutton (Stop), and the normally closed contacts of an overload relay (OL).



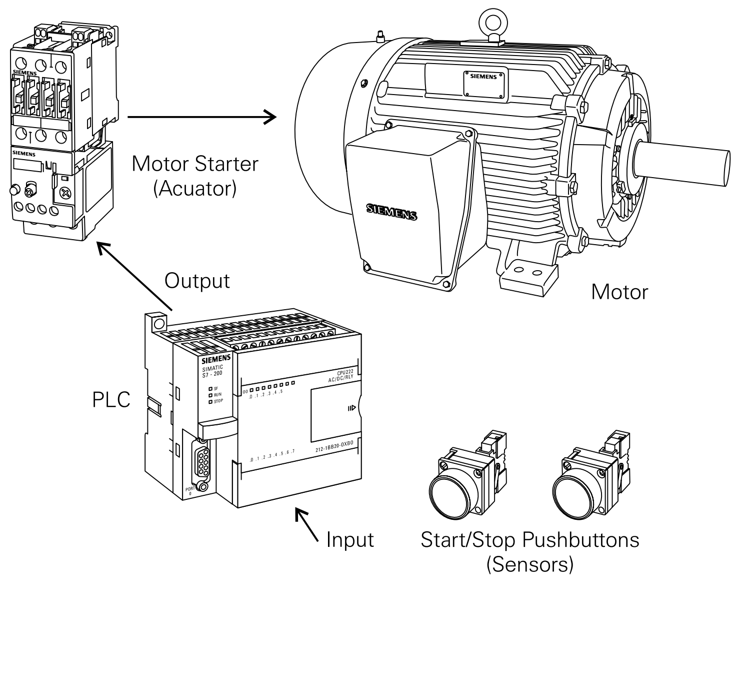
Momentarily depressing the Start pushbutton completes the path of current flow and energizes the motor starter (M).



This closes the associated M and Ma (auxiliary contact located in the motor starter) contacts. When the Start button is released a holding circuit exists to the M contactor through the auxiliary contacts Ma. The motor will run until the normally closed Stop button is depressed, or the overload relay opens the OL contacts, breaking the path of current flow to the motor starter and opening the associated M and Ma contacts.

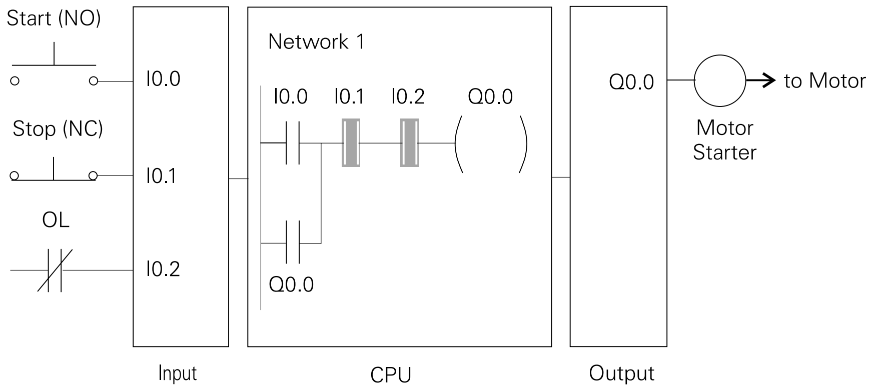


This control task can also be accomplished with a PLC.

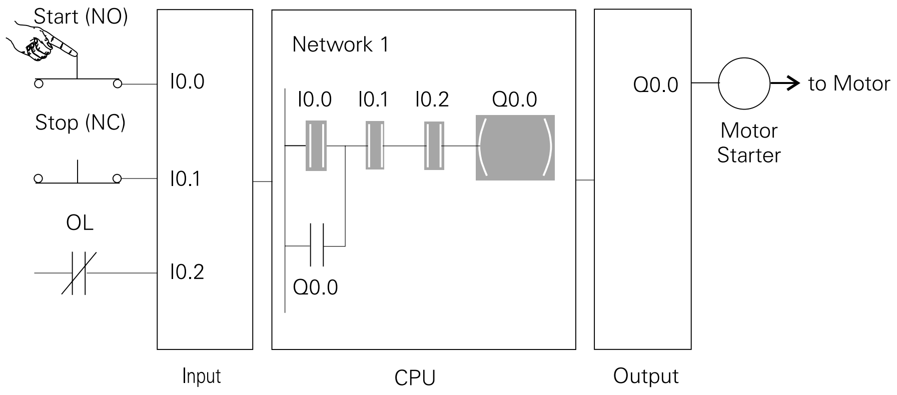


Program Instruction

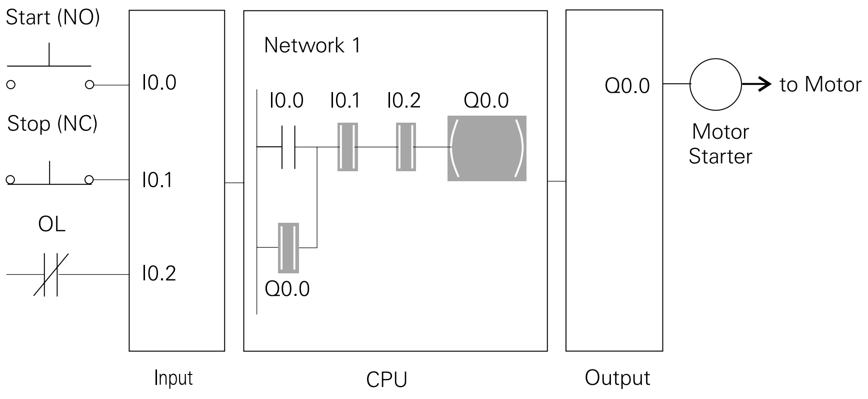
A normally open Start pushbutton is wired to the first input (I0.0), a normally closed Stop pushbutton is wired to the second input (I0.1), and normally closed overload relay contacts (part of the motor starter) are connected to the third input (I0.2). The first input (I0.0), second input (I0.1), and third input (I0.2) form an AND circuit and are used to control normally open programming function contacts on Network 1. I0.1 status bit is a logic 1 because the normally closed (NC) Stop Pushbutton is closed. I0.2 status bit is a logic 1 because the normally closed (NC) overload relay (OL) contacts are closed. Output Q0.0 is also programmed on Network 1. In addition, a normally open set of contacts associated with Q0.0 is programmed on Network 1 to form an OR circuit. A motor starter is connected to output Q0.0.



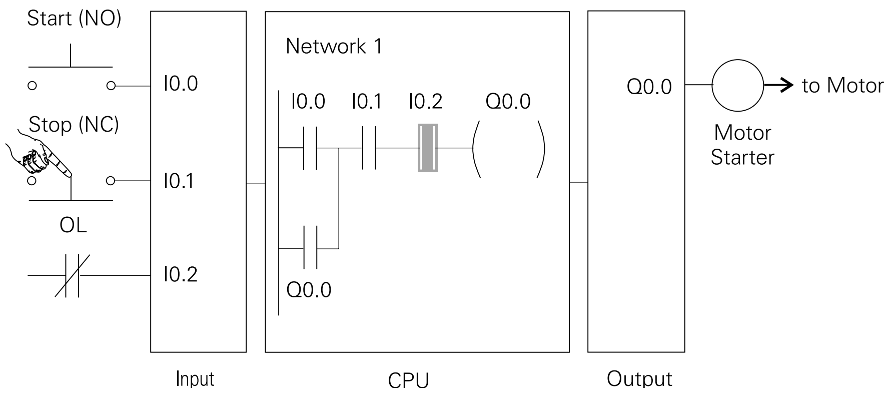
When the Start pushbutton is depressed the CPU receives a logic 1 from input I0.0. This causes the I0.0 contact to close. All three inputs are now a logic 1. The CPU sends a logic 1 to output Q0.0. The motor starter is energized and the motor starts.



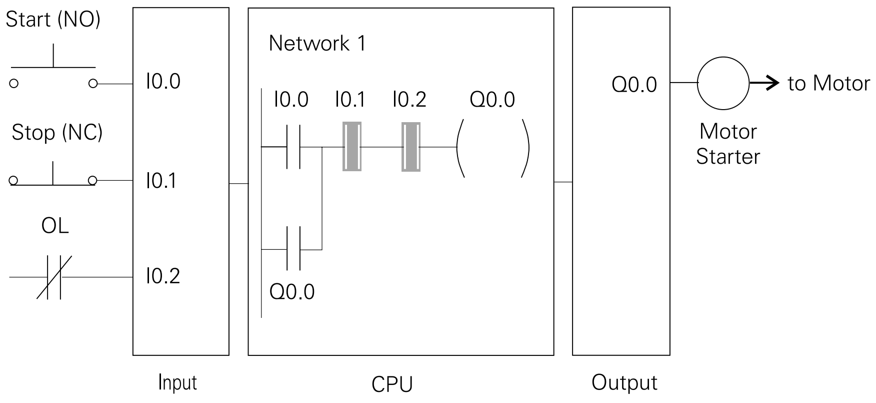
When the Start pushbutton is pressed, output Q0.0 is now true and on the next scan, when normally open contact Q0.0 is solved, the contact will close and output Q0.0 will stay on even if the Start pushbutton has been released.



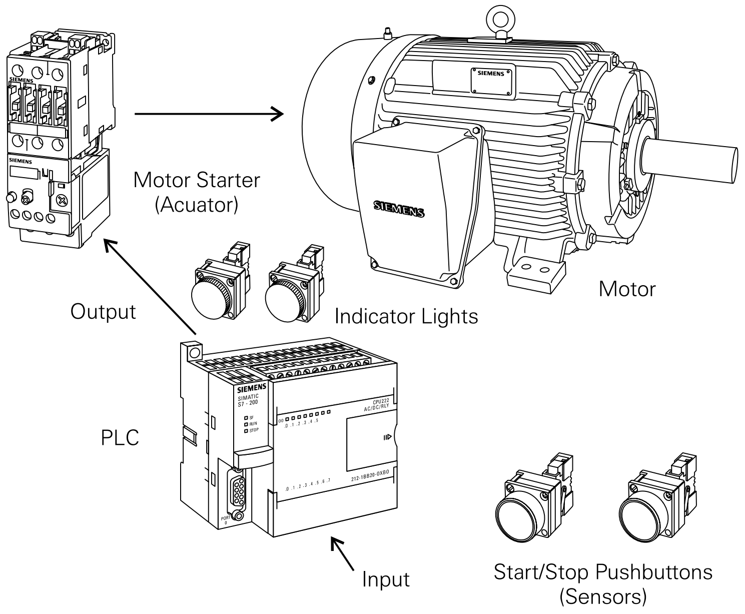
The motor will continue to run until the Stop pushbutton is depressed. Input I0.1 will now be a logic 0 (false). The CPU will send a binary 0 to output Q0.0. The motor will turn off.



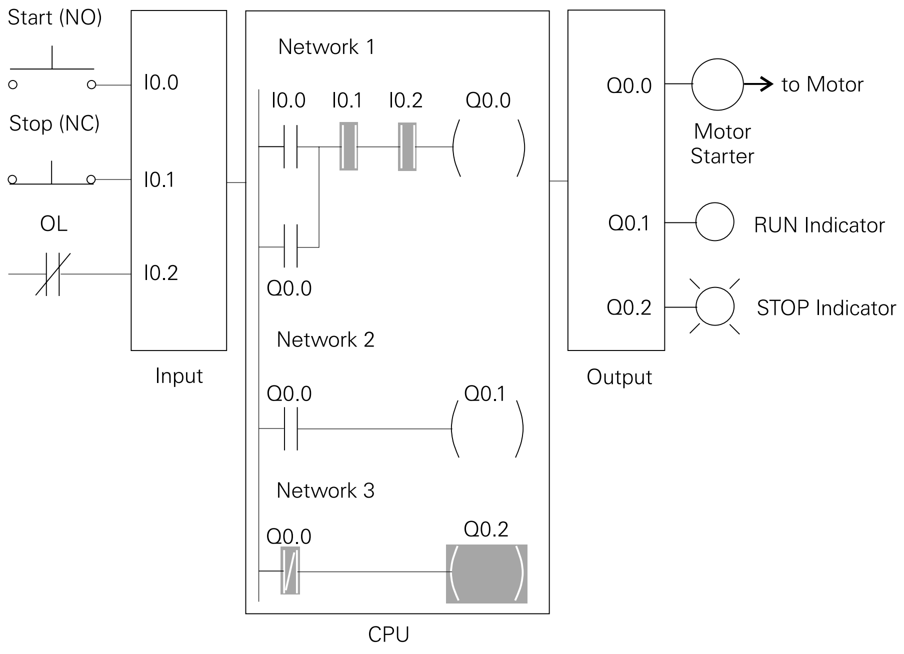
When the Stop pushbutton is released I0.1 logic function will again be true and the program ready for the next time the Start pushbutton is pressed.



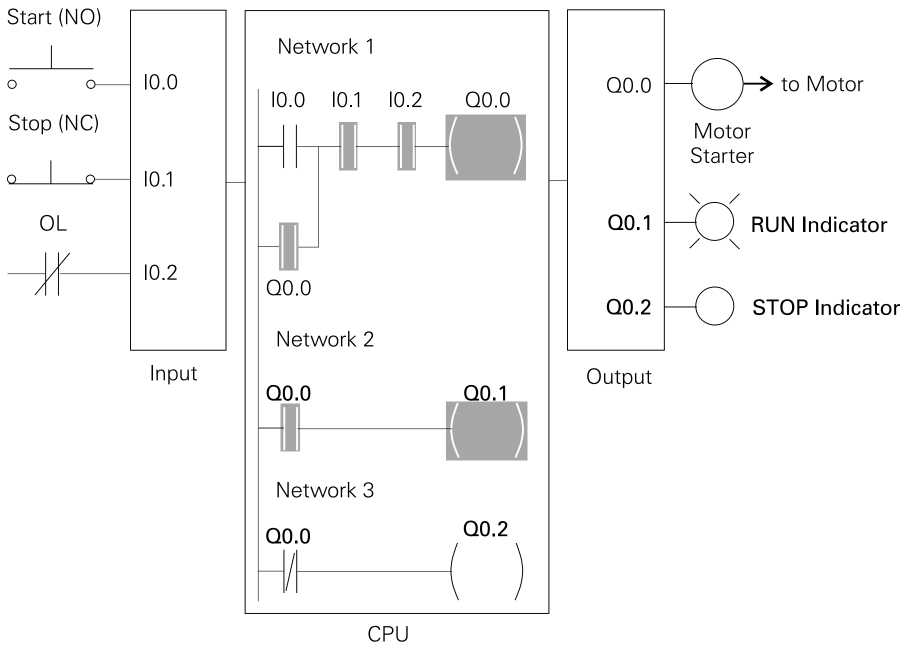
Expanding the Application The application can be easily expanded to include indicator lights for RUN and STOP conditions. In this example a RUN indicator light is connected to output Q0.1 and a STOP indicator light is connected to output Q0.2.



It can be seen from the ladder logic that a normally open output Q0.0 is connected on Network 2 to output Q0.1 and a normally closed Q0.0 contact is connected to output Q0.2 on network 3. In a stopped condition output Q0.0 is off. The normally open Q0.0 contacts on Network 2 are open and the RUN indicator, connected to output Q0.1 light is off. The normally closed Q0.1 on Network 3 lights are closed and the STOP indicator light, connected to output Q0.2 is on.

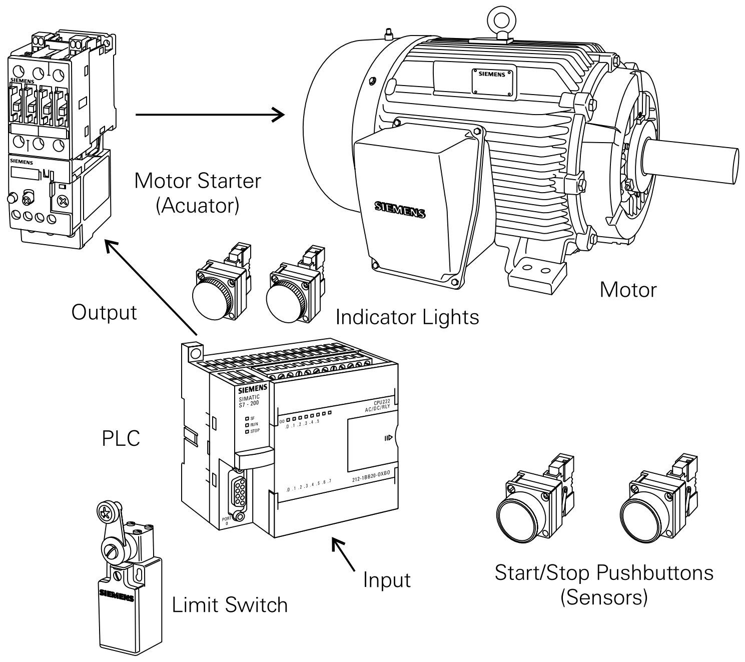


When the PLC starts the motor output Q0.0 is now a logic high (On). The normally open Q0.0 contacts on Network 2 now switch to a logic 1 (closed) and output Q0.1 turns the RUN indicator on. The normally closed Q0.0 contacts on Network 3 switch to a logic 0 (open) and the STOP indicator light connected to output Q0.2 is now off.

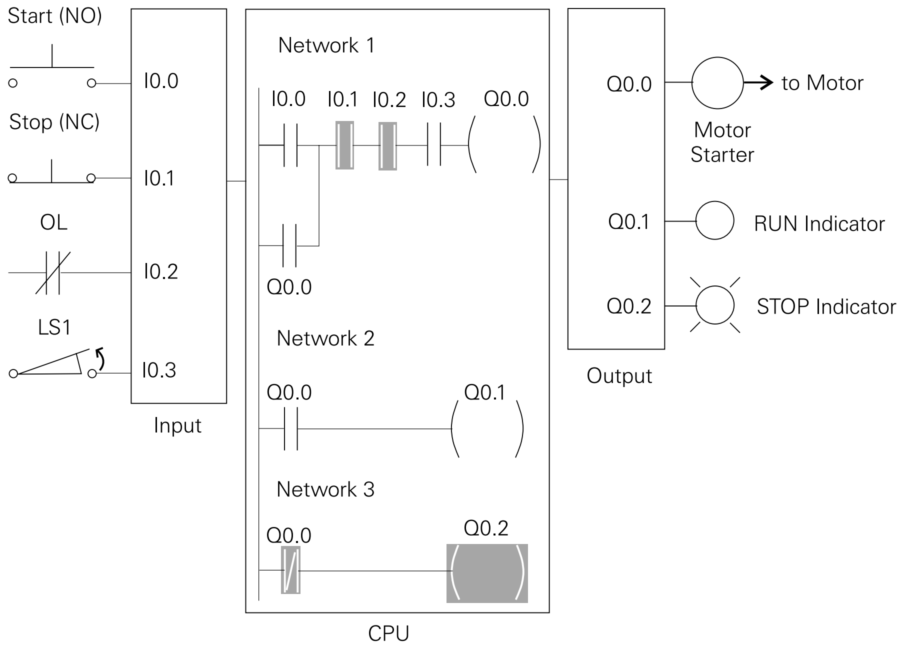


Adding a Limit Switch

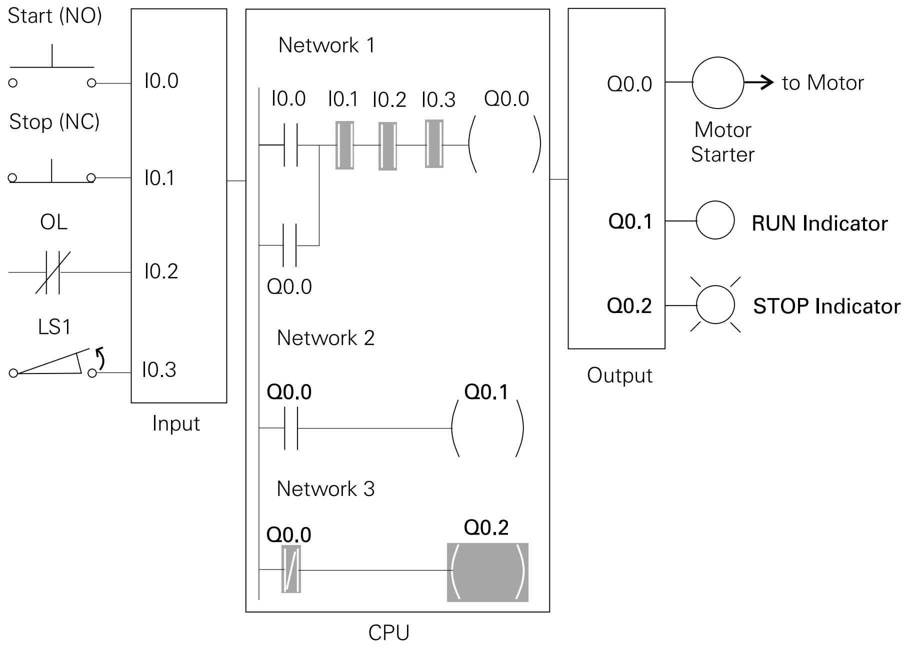
The application can be further expanded by adding a limit switch with normally open contacts to input I0.3.



A limit switch could be used to stop the motor or prevent the motor from being started. An access door to the motor, or its associated equipment, is one example of a limit switch's use. If the access door is open, the normally open contacts of LS1 connected to input I0.3 are open and the motor will not start.

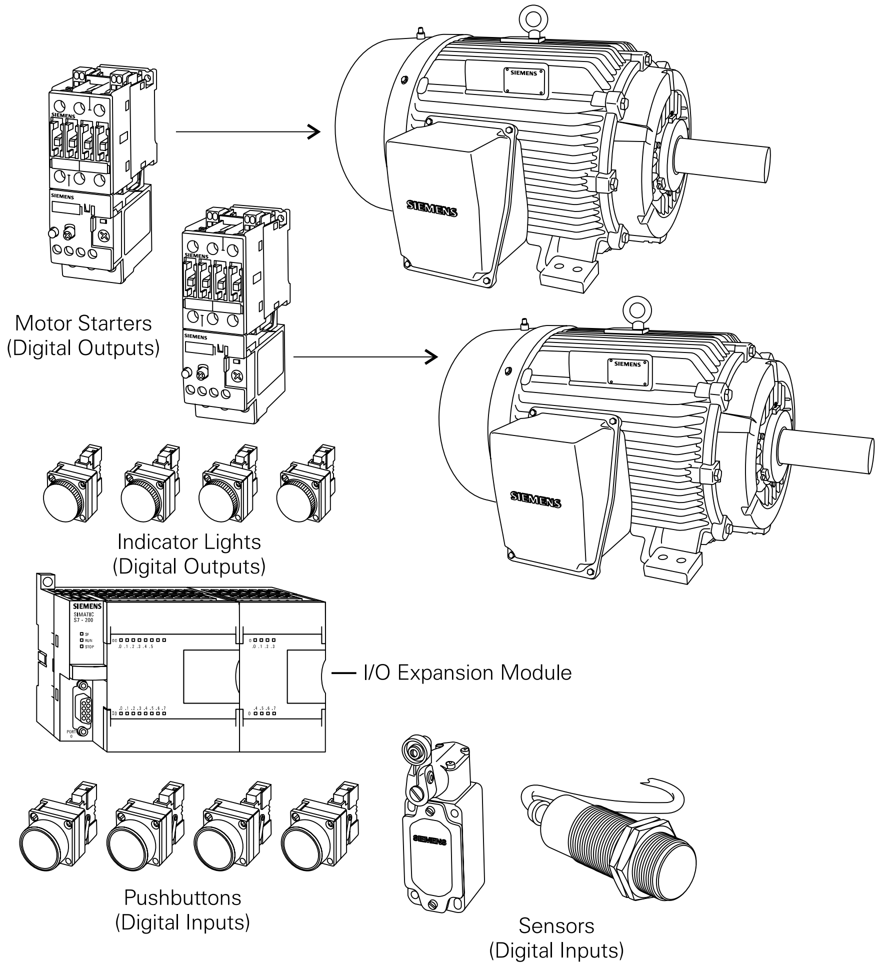


When the access door is closed, the normally open contacts on the limit switch (LS1) are closed. Input I0.3 is now on (logic 1), and the motor will start when the Start pushbutton is pressed.



Expansion

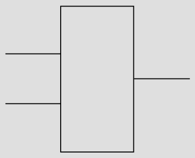
The PLC program can be expanded to accommodate many commercial and industrial applications. Additional Start/Stop pushbuttons and indicator lights can be added for remote operation, or control of a second motor starter and motor. Overtravel limit switches can be added along with proximity switches for sensing object position. In addition, expansion modules can be added to further increase the I/O capability. The applications are only limited by the number of I/Os and amount of memory available on the PLC.



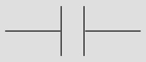
Review 4

1. Identify the following symbols:

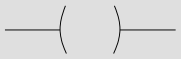
a. ___________



b. ___________

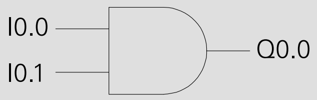


c. ___________



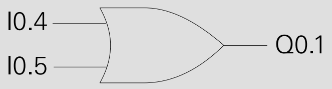
2. Complete the following tables:

And (A) Function



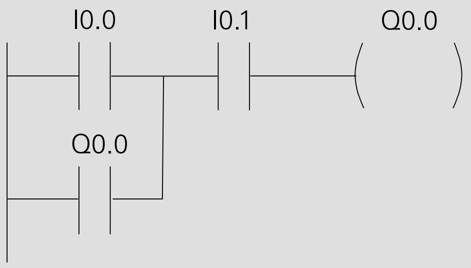
I0.0	I0.1	Q0.0
0	0	a. _____
0	1	b. _____
1	0	c. _____
1	1	d. _____

Or (O) Function



I0.4	I0.5	Q0.1
0	0	e. _____
0	1	f. _____
1	0	g. _____
1	1	h. _____

3. In the following instruction Q0.0 will be true (logic 1) when ___________ or ___________ is true, and when ___________ is true.


I0.0
I0.1
Q0.0
Q0.0