EJ501

PROGRAMMABLE LOGIC CONTROLLER (PLC) AND AUTOMATION
Objective

1. Language of PLC and PLC programming method.
2. Basic Logic Instruction Set
3. Special Sequential Function
4. Timer & Counter
PLC Programming Languages

IEC 1131-3 is the international standard for programmable controller programming languages. The following is a list of programming languages specified by this standard:

- Ladder diagram (LD)
- Sequential Function Charts (SFC)
- Function Block Diagram (FBD)
- Structured Text (ST)
- Instruction List (IL)

One of the primary benefits of the standard is that it allows multiple languages to be used within the same programmable controller. This allows the program developer to select the language best suited to each particular task.
Ladder Diagram

Ladder diagram is the main programming method used for PLC's. As mentioned before, ladder logic has been developed to mimic relay logic. The decision to use the relay logic diagrams was a strategic one. By selecting ladder logic as the main programming method, the amount of retraining needed for engineers and trades people was greatly reduced.

The first PLC was programmed with a technique that was based on relay logic wiring schematics. This eliminated the need to teach the electricians, technicians and engineers how to program - so this programming method has stuck and it is the most common technique for programming in today's PLC.

Mnemonic Instruction

There are other methods to program PLCs. One of the earliest techniques involved mnemonic instructions. These instructions can be derived directly from the ladder logic diagrams and entered into the PLC through a simple programming terminal.
Sequential Function Charts (SFC)

SFC have been developed to accommodate the programming of more advanced systems. These are similar to flowcharts, but much more powerful. This method is much different from flowcharts because it does not have to follow a single path through the flowchart.

Structured Text (ST)

Programming has been developed as a more modern programming language. It is quite similar to languages such as BASIC and Pascal. Structured Text (ST) is a high level textual language that is a Pascal like language. It is very flexible and intuitive for writing control algorithms.

Function Block Diagram (FBD)

FBD is another graphical programming language. The main concept is the data flow that start from inputs and passes in block(s) and generate the output.
Ladder Diagram

The programmable controller was developed for ease of programming using existing relay ladder symbols and expressions to represent the program logic needed to control the machine or process. The resulting programming language, which used these original basic relay ladder symbols, was given the name ladder language. Figure 9-1 illustrates a relay ladder logic circuit and the PLC ladder language representation of the same circuit.

*Note: The PLC will know the elements PB, LS, FS, and PL by their addresses once the address assignment has been performed.

Figure 9-1. Hardwired logic circuit and its PLC ladder language implementation.
Ladder Diagram Format

The ladder diagram language is a symbolic instruction set that is used to create PLC programs. The ladder instruction symbols can be formatted to obtain the desired control logic, which is then entered into memory. Since this type of instruction set consists of contact symbols, it is also referred to as contact symbology.

A continuous path is required for logic continuity

Figure 9-6. Ladder rung structure.

Figure 9-7. Illustration of several different continuity paths in a ladder rung.
Ladder Diagram Format

Figure 1-8. Monitoring device showing (a) power continuity through the rung—inputs 11 and 12 are ON, turning output 40 ON—and (b) power continuity through relay input 12, thus output 40 is not ON.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examine-ON/Normally Open</td>
<td></td>
<td>Tests for an ON condition in a reference address</td>
</tr>
<tr>
<td>Examine-OFF/Normally Closed</td>
<td></td>
<td>Tests for an OFF condition in a reference address</td>
</tr>
<tr>
<td>Output Coil</td>
<td></td>
<td>Turns real or internal outputs ON when logic is 1</td>
</tr>
<tr>
<td>NOT Output Coil</td>
<td></td>
<td>Turns real or internal outputs OFF when logic is 1</td>
</tr>
<tr>
<td>Latch Output Coil</td>
<td></td>
<td>Keeps an output ON once it is energized</td>
</tr>
<tr>
<td>Unlatch Output Coil</td>
<td></td>
<td>Resets a latched output</td>
</tr>
<tr>
<td>One-Shot Output</td>
<td></td>
<td>Energizes an output for one scan or less</td>
</tr>
<tr>
<td>Transitional Contact</td>
<td></td>
<td>Closes for one scan when its trigger contact makes a positive transition</td>
</tr>
</tbody>
</table>

Table 9-2. Ladder relay instructions.
Boolean

Some PLC manufacturers use **Boolean language**, also called **Boolean mnemonics**, to program a controller. The Boolean language uses Boolean algebra syntax (see Chapter 3) to enter and explain the control logic. That is, it uses the AND, OR, and NOT logic functions to implement the control circuits in the control program. Figure 9-3 shows a basic Boolean program.

The Boolean language is primarily just a way of entering the control program into a PLC, rather than an actual instruction-oriented language. When displayed on the programming monitor, the Boolean language is usually viewed as a ladder circuit instead of as the Boolean commands that define the instruction. We will discuss Boolean programming, along with its instruction set, at the end of this chapter.

![Hardwired Circuit](image)

**Figure 9-3.** Hardwired logic circuit and its Boolean representation.
Ladder Programming

- The form of programming commonly used with PLCs is ladder programming.
- This involves each program task being specified as though a rung of a ladder.
- Thus such a rung could specify that the state of switches A and B, the inputs be examined and if A and B are both closed then a solenoid, the output is energized.

(a) & (b) Alternative ways of drawing an electric circuit, (c) comparable rung in a ladder program.
UNIT 4
PLC PROGRAMMING

Ladder symbols

Input as contacts not closed until input

Input as contacts which are closed until input

Output

Special instruction

Input 1

Output A occurs when input 1 occurs

Input 1
Input 3

Output B occurs when input 1 and input 3 occur

Input 4

Output C occurs when input 4 or input 5 occurs

Input 5

End of program

END
Logic Instruction

Logic instruction can be obtained by combinations of switches

1. **LOAD (LD) and LOAD NOT (LD NOT)**

   A prefix contacts at the bus bar. If at the prefix of the bus bar is the NO(normaly open) contacts, then the instruction is LOAD or LD. For the NC(normaly closed) contacts, then the instruction is LOAD NOT or LD NOT.
2. AND and AND NOT

The instruction for series connection in the ladder. For NO (normaly open) contacts, then the instruction is AND. For NC (normaly closed) contacts, then the instruction is AND NOT.
3. **OR and OR NOT**

The instruction for parallel connection in the ladder. For NO(normaly open) contacts, then the instruction is OR. For NC(normaly closed) contacts, then the instruction is OR NOT.
4. OR LD and AND LD

The union between a series connection which is connected into one, so if there are two series circuits are combined into one, then the instruction is OR LOAD.

The union between a parallel connection which is connected into one, so if there are two parallel circuits are combined into one, then the instruction is AND LOAD.
5. **OUT and OUT NOT**
The instruction for output or load, such as Relay, Contactor, Light, Buzzer, and others. If the output is NO(normaly open), then the instruction is OUT, but if the output is NC(normaly closed), then the instruction is OUT NOT.

![Diagram of OUT and OUT NOT instructions](image)

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>LD</td>
<td>00000</td>
</tr>
<tr>
<td>00001</td>
<td>OUT</td>
<td>00200</td>
</tr>
</tbody>
</table>

6. **END**
The instruction for the end of the program, if a program does not end with the END instruction code, the program will not be able to do (error).

![Diagram of END instruction](image)

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>LD</td>
<td>00000</td>
</tr>
<tr>
<td>00001</td>
<td>AND NOT</td>
<td>00001</td>
</tr>
<tr>
<td>00002</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>00003</td>
<td>END(01)</td>
<td></td>
</tr>
<tr>
<td>IEC 1131-3</td>
<td>Mitsubishi</td>
<td>OMRON</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>LD</td>
<td>LD</td>
<td>LD</td>
</tr>
<tr>
<td>LDN</td>
<td>LDI</td>
<td>LD NOT</td>
</tr>
<tr>
<td>AND</td>
<td>AND</td>
<td>AND</td>
</tr>
<tr>
<td>ANDN</td>
<td>ANI</td>
<td>AND NOT</td>
</tr>
<tr>
<td>OR</td>
<td>OR</td>
<td>OR</td>
</tr>
<tr>
<td>ORN</td>
<td>ORI</td>
<td>OR NOT</td>
</tr>
<tr>
<td>ST</td>
<td>OUT</td>
<td>OUT</td>
</tr>
</tbody>
</table>

Table: Instruction code mnemonics
Logic Solution and Ladder Diagram Representation
Special Sequential Function

KEEP

KEEP instruction is used to store the status of a bit operand based on two execution conditions. For this purpose, KEEP instruction is connected to two lines of instructions. The first instruction line is used to enable a operand bit, while the second instruction line is used to disable a operand bit, it will happen if conditions on the execution of instruction line associated is ON. To activate KEEP press FUN 11.
SET

Turns ON B for an ON execution condition; does not affect B for an OFF execution condition

RESET

Turns OFF B for an ON execution condition; does not affect B for an OFF execution condition.
JUMP (JMP) and JUMP END (JME)

JUMP instruction (JMP), same as the interlock instructions, coupled with the JUMP END (JME). If the execution condition and a JUMP instruction is ON, the program worked normally as if nothing JUMP instruction. However, if the condition executed and a JUMP instruction is OFF, then execution of the program directly to the JUMP END instruction without making any changes in status between JUMP and JUMP END instruction. All JUMP and JUMP END instruction was given the number 01 to 99. This number is only used once, meaning that, in the ladder diagram can not exist JUMP numbers double or more and one, except for number 00. JUMP instruction is done with a certain number, then the execution will go directly to JUMP END with the same number. To activate JUMP press FUN 04 and JUMP END press FUN 05.
INTERLOCK (IL) and INTERLOCK CLEAR (ILC)

These instructions can also be used to overcome the branching points encountered on the ladder diagrams. Instruction IL - ILC is always used together. Instruction interlock if the execution condition is OFF (all instructions between interlock and interlock clear will not be done). To activate IL press FUN 02 and ILC press FUN 03.
DIFFERENTIATE UP (DIFU) and DIFFERENTIATE DOWN (DIFD)

DIFU instruction used to enable bit operand instantaneous (only one cycle) during transition of the execution condition from OFF to ON. While DIFD instruction used for the same purpose with DIFU, only when there is a transition state of the execution from ON to OFF (reverse transition DIFU). To activate press DIFU FUN 13 and DIFD press FUN 14.
SHIFT REGISTER (SFT)

Function of Shift Registers (SFT) is to shift the data by using the clock pulse. Data which can be shifted are IR, AR, HR, LR. Shift Registers has three inputs; data input (I), clock input (P) and reset (R). Data Input is used to enter data into location of data channel. Clock input is used to enter the clock to shift the data that has been incorporated into the input data through location of data channel. Reset input have function to create the initial conditions (0) all bits in the locations of data channel. The other is the initial of data lines (St) occupied bits start bit is shifted and the final of the data channel (E) which serves as the final borders of the bits are bits that are shifted. St must be greater or equal than E and must be on the same data area. To operate SFT press FUN 10.
MOVE (MOV)

Move instruction is a function to move or transfer or to copy data from source (S) to destination (D). To operate MOV press FUN 21.
ADD (30)

Have function to add three pieces of parameters: data 1 (Au), data 2 (Ad) and carry (CY), then the results are placed on data 3 (R). Carry will be set to 1 if the sum is greater than 9999.
SUB (31)

Have function to subtract three pieces of parameters: data 1 (Mi), data 2 (Su) and carry (CY), then the results are placed on data 3 (R). If the result is negative then carry will be set to 1.
**TIMER (TIM)**

A timer is activated when its execution condition goes ON and is reset (to SV) when the execution condition goes OFF. Once activated, TIM measures in units of 0.1 second from the SV.

If the execution condition remains ON long enough for TIM to time down to zero, the Completion Flag for the TC number used will turn ON and will remain ON until TIM is reset (that is, until its execution condition goes OFF).

N : Timer Number --> 000 - 511
SV : Set value (#0050) – 5 sec (50 x 0.1sec)
COUNTER (CNT)

Counter is used for count the setting value (SV) towards 0 when the pulse counter (CP) changes from OFF to ON. After the counter value changes to 0 then the counter output status will change from OFF to ON and will persist as long as the reset button (R) has not been pressed. When the reset button is pressed, status of the output counter will be OFF and the counter value back to the original setting value. The number of counter that can be used up to 511. Do not give the same number on counter Timer.

N : Counter Number --> 000 - 511
SV : Set value (#0010) – 10 counter

http://program-plc.blogspot.com/
Thank You

Mahalo

Kiitos

Toda

Obrigado

Merci

Grazie

Gracias

Takk