Structured text programming

The structured text (ST) language is high level language currently used in PLC programming. Like any other high level language it contains may functions and key words. Many of which also do equivalent functions as in the other programming languages. It enables PLC programmer to develop and program more complex control algorithms. Structurally it highly looks like the well known and efficient PASCAL programming language.

Comments

Comments are used to increase the readability of programs. They are used to mark the beginnings and ends of program sections and also to give brief explanation to the program instructions or parts they headed. Here we have the one line (//) comments and the multiline or block comments {(* comments *)}.

Logical Statements

Logical statements consist of one or more logical operators (AND, OR, XOR, NOT) and any same data type operands (Bit, Byte, Word, Dword, Lword).

Example 1: Write ST program to develop 2-to-4 decoding circuit. Assume S1 and S0 are the decoder inputs and Y3,Y2,Y1, and Y0 are the decoder outputs.

Solution: Y3:= S1 AND S0; Y2:= S1 AND NOT (S0); Y1:= NOT(S1) AND S0;
Y0:=NOT(S1) AND NOT(S0);

Example 2: Write ST program segment to show the complement of a product of variables equals the sum of the complements of the variables.

Solution: Y0:= NOT (%MW1.1 AND %MW1.0); // Y0 is the complements of product.
Y1:= NOT %MW1.1 OR NOT %MW1.1; // Y1 is the sum of complements.

Example 3: Show the complement of sum of variables equals the product of complements of the variables. Assume two variables.

Solution: Y0:= NOT (%MW1.1 OR %MW1.0); // Y0 is the complements of sum.
Y1:= NOT %MW1.1 AND NOT %MW1.1; // Y1 is the product of complements.

Example 4: Write program segment to invert memory bit %MW1.2 each scan if %MW1.1 is true.

Solution: CLOCK:= (%MW1.1 XOR CLOCK); // CLOCK=%MW1.2

Example 5: X is 64 bit length bit-type variable. Suggest solution to invert the middle four bits each new scan.

Solution:

X:=X XOR 2#0000000000000000000000000000111100000000000000000000000000000000; (*X is LWORD bit type variable.*)

Example 6: Write the ST equivalent of the circuit shown in figure 10.1.

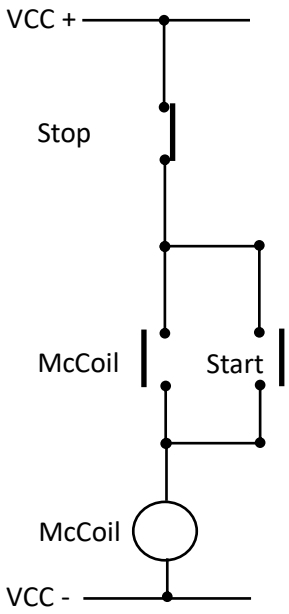


Figure 1: Latch circuit

Solution:

McCoil:=Stop AND (Start OR McCoil); (* McCoil is the output contactor coil bit, Stop is the Stop pushbutton input bit, and Start is the Start pushbutton input bit*)

Example 7, Draw the traditional control diagram of a changeover switch which supplies electric power from either the national grid or local generator with the highest priority to the national grid. Also write the ST equivalent of that controller.

Solution:

(* Let the availability of national grid represented by the input %IX0.0.0 and has the label NGOK. Also assign the label LGOK for the %IX0.0.1 to represents the availability state of the local generator. For the changeover contactors, assign NG_C for %QX0.0.0 output bit responsible to control the national grid interfacing power contactor and LG_C for %QX0.0.1 for the contactor connecting the generator to the load.*)

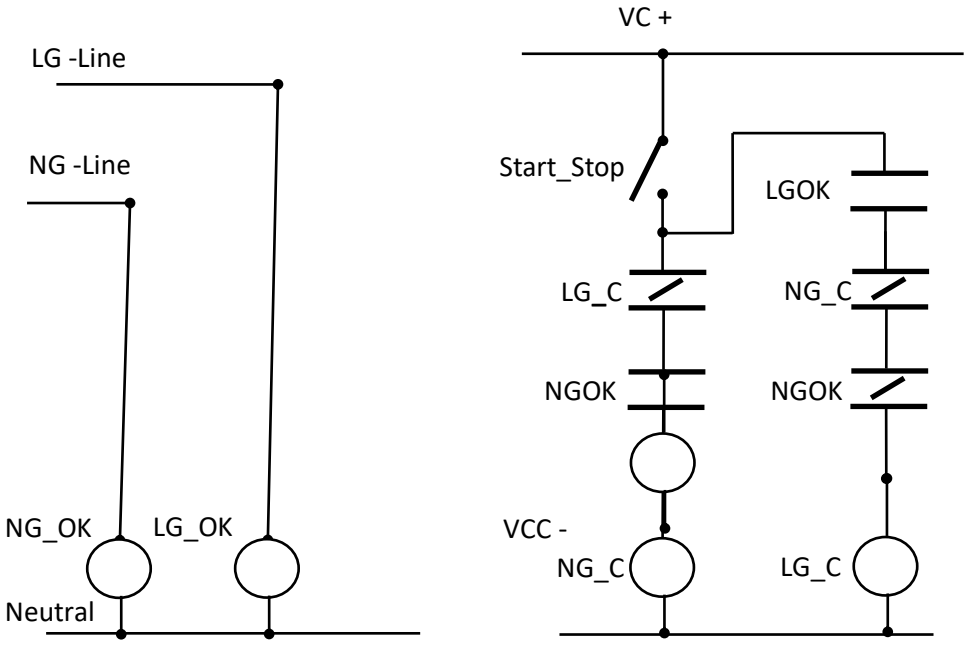


Figure 2: Changeover switch control

NG_C:=Start_Stop AND NOT LG_C AND NGOK;
LG_C:=Start_Stop AND LGOK AND NOT NG_C AND NOT NGOK;

Example 9: Write the ST text that functions as a 4-input multiplexing circuit.

Solution:

Y3:=S1 AND S0 AND I3;
Y2:=S1 AND NOT S0 AND I2;
Y1:=NOT S1 AND S0 AND I1;
Y0:=NOT S1 AND NOT S0 AND I0;

Y:=Y1 OR Y2 OR Y3 OR Y4;

Example 10: Write the ST equivalent of JK flip-flop.

dummy:= (J AND NOT Q) OR (K AND Q);

R_Trig(Clk:=dummy,q=>Flip);

Q:=Flip XOR Q;

Arithmetic Statements

Arithmetic statements also consist of one or more arithmetic operators and any same num-type operands. The operators involved here are the addition (+), the subtraction (-), the multiplication (*), the division (/),the remainder (MOD), and the exponential (**). All these types allow any numerical data type except the exponential one is restricted to real data ones.

Example 11: Write the ST text that generates the Triangular membership function drawn in figure10.3.

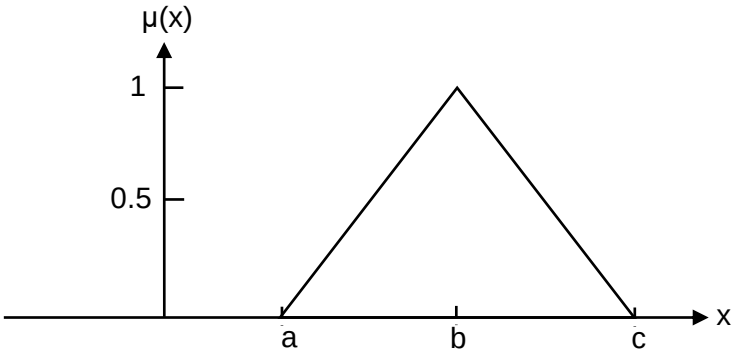


Figure 3: Triangular membership function

Solution:

U_x= MAX(0,MIN((x-a)/(b-a),(c-x)/(c-b))); // a,b,c are real type variables.

Example 12, Write the ST text that generates the bell-shaped membership function drawn in figure10.4.

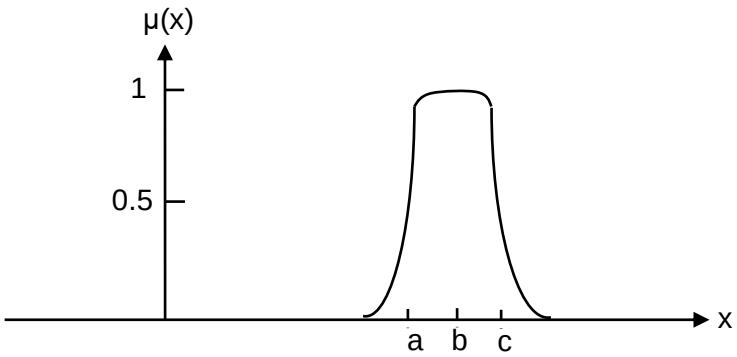


Figure 4: Bell-shaped membership function

U_x:=1/(1+(ABS((x-b)/a))**(2*c)// a,b,c are real type variables.

Example 13, Write the ST text that generates the sigmoid-right membership function drawn in figure10.5.

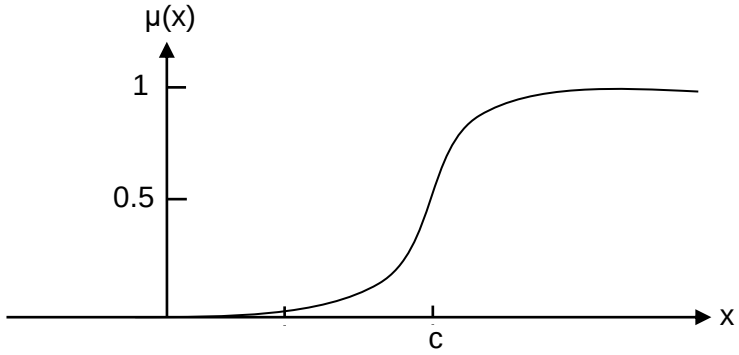
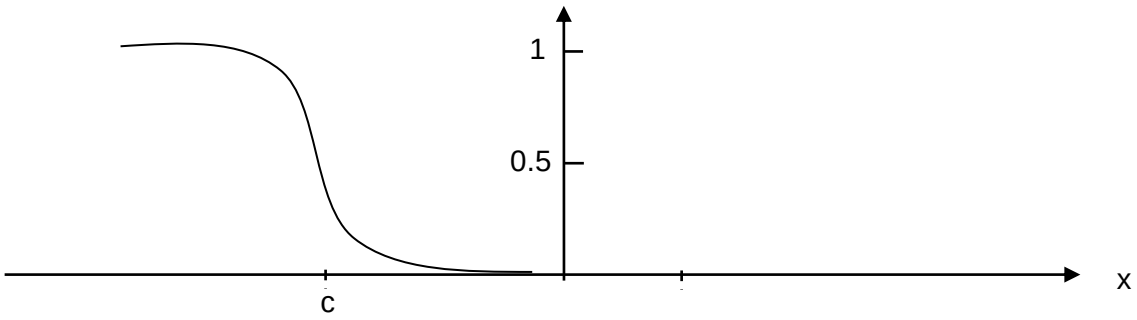


Figure 5: Sigmoid-right membership function

Solution:

U_x= 1/(1+EXP(IN:=a*(c-x))); // a is positive number

Example 14, Write the ST text that generates the sigmoid-left membership function drawn in figure6.



Solution:

U_x= 1/(1+EXP(IN:=a*(c-x))); // a is negative number

Example 10.15, Write the ST text that solves the equation of the definite minimum inverse time over current relay.

Solution:

T:= c/((I/IS)**a -1); //

IF Conditional Statements

IF conditional statements control (allow or deny) the execution of one or more statements bounded by their start and termination identifiers. The execution or deny depends upon the true or false results of the comparison expression following the statements identifiers (IF, ELSE, ELSIF). These statements have a number of structures as illustrated in figure 7.

```
IF Condition THEN
    statement 1; . . . .
    statement n;
END_IF;
```

```
IF Condition THEN
    statement A1; . . . .
    statement An;
ELSE
   statement B1;. . . .
   statement Bm;
END_IF;
```

```
IF Condition1 THEN
      statement A1; . . . .
      statement An;
ELSIF Condition k
     statement K1;. . . .
     statement Km;
ELSE
      statement V1;
       statement Vj;
END_UF;
```

Figure 7: Conditional statement structures

Example 16, Write the ST text to get set / reset function.

Solution :

```
IF S=1 Then Q:=1;

END_IF;

IF R=1` Then Q:=0;

END_IF;
```

Example 17, Write the ST text to test two integer type variables X and Y and assign the value of the smallest one to Z;

Solution:

```
IF X<=Y THEN

Z:=X;

ELSE

Z:=Y;

END_IF:
```

Example 18, Write the ST text to obtain the membership function sketched in figure10.8

Solution:

```
IF X<=a THEN
Y:=0;
ELSIF X>=b THEN
Y:=1;
 ELSE
 Y:=(X-a)/(b-a);
END_IF;
```

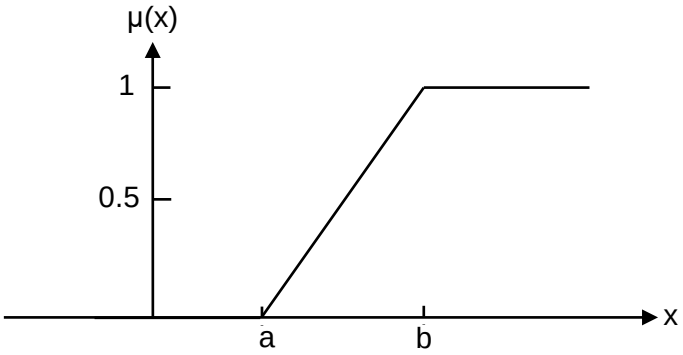


Figure 8: S-like membership function

Example 19, Write the ST text to obtain the membership function sketched in figure9.

Solution:

```
IF X>=a THEN
Y:=0;
ELSIF X<=b THEN
Y:=1;
ELSE
Y:=(X-a)/(b-a);
END_IF;
```

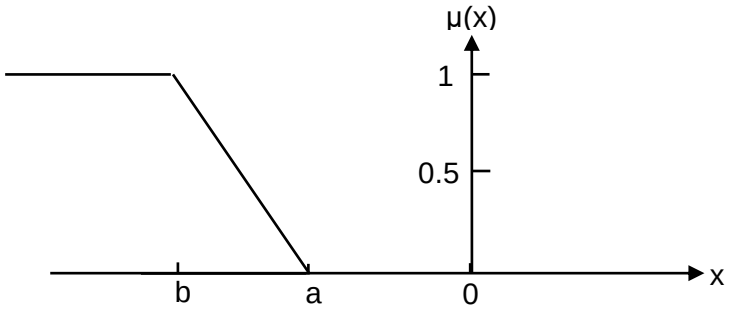


Figure 9: Z-like membership function

CASE Conditional Statements

CASE is used to execute one out of many statements. The selected statement is that one whose line starts with the current value of the integer type expression following the reserved word "CASE". The two possible basic structures of this selection facility are pictured in figure 10.

```
CASE  Expression OF
 1: statement ; . . .
    statement;
        :
 n: statement ; . . .
    statement;
 END_CASE
```

```
CASE  Expression OF
 1: statement ; . . .
    statement;
        :
 n: statement ; . . .
    statement;
 ELSE
 statement;
statement;
END_CASE
```

Figure 10: CASE statement structures

Example 20, Propose ST program to implement the block diagram shown in figure 11.

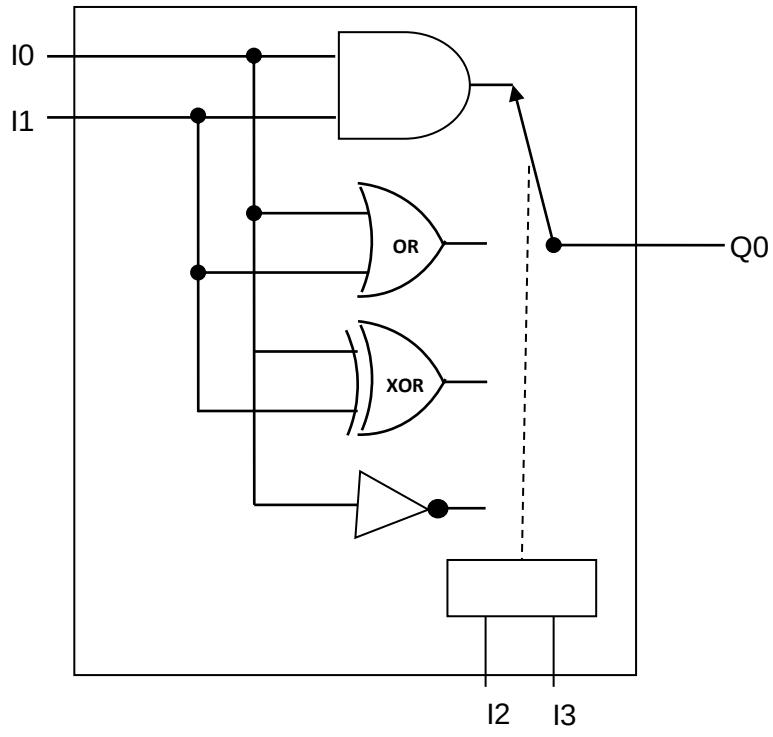


Figure 11: One out of four logic functions

Solution:

```
V:=WORD_TO_INT(IN:=(SHR_WORD(IN:=IW0.0.0,N:=2) AND 16#0003));
CASE V OF
0: %QX0.0.0:= %IX0.0.0 AND %IX0.0.1;
1: %QX0.0.0:= %IX0.0.0 OR %IX0.0.1 ;
2: %QX0.0.0:= %IX0.0.0 XOR %IX0.0.1;
3: %QX0.0.0:= NOT %IX0.0.0 ;
END_CASE;
```

Example 21, Propose ST program to output the content of the selected location in table 1 into %QW0.0.0.

```
CASE Location OF
1: %QW0.0.0:= MOVE_WORD (IN:=16#8001);
2: %QW0.0.0:= MOVE_WORD (IN:= 16#4002);
3: %QW0.0.0:= MOVE_WORD (IN:= 16#2004);
4: %QW0.0.0:= MOVE_WORD (IN:= 16#1008;
5: %QW0.0.0:= MOVE_WORD (IN:= 16#0810);
6: %QW0.0.0:= MOVE_WORD (IN:= 16#0420);
7: %QW0.0.0:= MOVE_WORD (IN:= 16#0240);
8: %QW0.0.0:= MOVE_WORD (IN:= 16#0180);
9: %QW0.0.0:= MOVE_WORD (IN:= 16#0240);
10: %QW0.0.0:= MOVE_WORD (IN:= 16#0420);
11: %QW0.0.0:= MOVE_WORD (IN:= 16#0810);
12: %QW0.0.0:= MOVE_WORD (IN:= 16#1008);
13: %QW0.0.0:= MOVE_WORD (IN:= 16#2004);
14: %QW0.0.0:= MOVE_WORD (IN:= 16#4002);
15: %QW0.0.0:= MOVE_WORD (IN:=16#8001);
16: %QW0.0.0:= MOVE_WORD (IN:=16#0000);
END_CASE;
```

Location	Contents
1	16#8001
2	16#4002
3	16#2004
4	16#1008
5	16#0810
6	16#0420
7	16#0240
8	16#0180
9	16#0240
10	16#0420
11	16#0810
12	16#1008
13	16#2004
14	16#4002
15	16#8001
16	16#0000

FOR Statements

FOR is used to do predetermined repetition times . It takes the general form shown in figure 12. The instruction integer data type counter has three control parameters, the start, the end, and the increment .

```
FOR Counter:= Start To END BY increment  DO
Statement 1;
Statement n;
END_FOR;
```

Figure 12: FOR instruction general form.

Example 22, Write PLC program to sort the contents of the array Y whose elements are stored in memory location %MW10 to %MW15 in an ascend manner;

Solution:

```
FOR I:= 0  TO 5 DO
    FOR J:= I +1 TO 5 DO
        IF Y[J]< Y[I] THEN
        Dummy:=Y[I];
        Y[I]:=Y[J];
         Y[J]:=Dummy;
        END_IF;
    END_FOR;
END_FOR;
```

Example 23, Write PLC program to sort the contents of the array Y whose elements are stored in memory location %MW10 to %MW15 in a descendent manner;

Solution:

```
FOR I:= 0  TO 5 DO
```

```
    FOR J:= I +1 TO 5 DO
        IF Y[J]> Y[I] THEN
        Dummy:=Y[I];
        Y[I]:=Y[J];
         Y[J]:=Dummy;
        END_IF;
    END_FOR;
END_FOR;
```

WHILE Statements

WHILE is used to do repetition with unknown number of times . Here the continuity of repetition depends upon the logical result of the expression comes immediately after the WHILE word. That means it first examines the continuity flag and then decides to inter the loop or goes out of it to allow the next statement to be executed. The syntax of this statement is :

WHILE Expression is true **DO**

Statement 1;

Statement n;

END_WHILE;

Example 24, Write PLC program to search the contents of the array Y seeking a given number. Assume Y to be one dimensional array of length equals 6;

Solution:

```
J:=0;  //  an integer data type
Location:=6;  // the assigned value should be any number greater than Y length.
WHILE J<=5 DO
IF item=Y[J] THEN Location:=J;  EXIT; END_IF;
J:=J + 1;
END_WHILE;
```

REPEAT Statements

REPEAT is used to do repetition with unknown number of times . Here the continuity of repetition depends upon the logical result of the expression comes immediately after the UNTIL word. That means it first executes the loop and then decides to repeat the loop or goes out of it to allow the next statement to be executed. With this statement. the loop is executed at least once because the continuity flag is examined at the end of the loop. The syntax of this statement is :

```
REPEAT
Statement 1;
Statement n;
UNTIL Condition is true  // no semicolon here
END_REPEAT;
```

Example 25, Using Newton's method, find the root of the equation $x^3 - x - 1 = 0$

Solution:

```
Xn_1:=0.5;  //  assume an initial value to Xn_1;
REPEAT
Xn:=(2*Xn_1**3 +1)/(3*Xn_1**2-1);  //  Xn= Xn + f(x)/f '(x);
FX:=Xn**3- Xn -1;
Xn_1:=Xn;
UNTIL (ABS(FX)<=0.0000001)
END_REPEAT;
```