Lenguaje ST
(Structured Text)

Controladores Lógicos Programables

Introducción

ST (Structured Text): 5to lenguaje definido en la norma IEC 61131-3

Lenguaje de texto de alto nivel, con sintaxis similar a la de PASCAL

- Lenguaje pequeño
  - 40 palabras reservadas
  - 10 tipos de sentencias

Particularmente útil para cálculos aritméticos complejos

Existen menos implementaciones que LD, casi todas basadas en IEC 61131-3

Estructura de programa ST

Programa ST es un archivo de texto

- Consiste en un conjunto de sentencias
- Cada sentencia contiene expresiones ST válidas

Una expresión ST resulta en un único resultado. Se compone de:

- Operandos
- Operadores:

Paréntesis	( )
Invocación a función	Identificador(parámetros)
Exponente	EXPT
Aritméticos	+ - / * MOD
Comparación	< > <= >= = <>
Booleanos	NOT AND OR XOR

Expresión ST: ejemplo

Expresión con operadores booleanos

```
VAR1 := ( VAR1  AND  VAR2 )   OR    VAR3
```

1	Resultado 1 = (VAR1 AND VAR 2)
2	Resultado 2 = (Resultado 1) OR VAR3
3	Asigna Resultado 2 a VAR1

Sentencias

Comentario

(* *)

(* Comentario *)

Asignación

:= Transfiere valor de una expresión en una variable

```
A := B;
C := C + 1;
VAR := 7;
VAR := VAR * 10;
Y := COS(X) +  12;
```

Invocación a bloque funcional

1. Definición de instancia de bloque funcional en la declaración de variables

```
TIMER1: TON;
```

2. Invocación de nombre de la instancia seguido de asignaciones de entrada

```
TIMER1 (IN := DISPARO_TIMER,  PT := TIEMPO);
```

3. Acceso a las salidas del bloque funcional

```
TIMEOUT_TIMER1 := TIMER1.Q;
TIMER1 (IN := DISPARO_TIMER,  PT := TIEMPO, Q=>TIMEOUT_TIMER1);
```

Retorno

RETURN Termina una función o un programa ST

```
IF ( A > B ) THEN
    RETURN;
END_IF;
```

Selección

IF.. THEN .. ELSE .. END_IF

CASE .. OF.. END_CASE

Iteración

FOR .. DO

WHILE .. DO

REPEAT .. UNTIL

EXIT

Sentencias condicionales

IF: ejecución condicionada a resultado de expresión booleana

```
IF ( Expresión boolena 1 ) THEN
    (* Instrucciones si Expresión booleana 1 = TRUE  *)
ELSIF (Expresión booleana 2) THEN
    (* Instrucciones si Expresión booleana 2 = TRUE  *)
.....
ELSE
    (* Instrucciones si no se cumple ninguno de los anteriores  *)
END_IF;
```

Ejemplo:

```
IF A = 3   THEN
   B := 7;
ELSE
   B := 3;
END_IF;
```

Sentencias condicionales

CASE: ejecución condicionada al valor de una variable

```
CASE <VAR> OF
    <VALOR1>: (* Instrucción 1 *)
    <VALOR3, VALOR4>: (* Instrucción 2 *)
    <VALOR5 .. VALOR6>: (* Instrucción 3 *)
.....
ELSE
    (* Instrucciones si no se cumple ninguno de los anteriores  *)
END_CASE;
```

Ejemplo:

```
CASE INT1 OF
1:          BOOL1 := TRUE;
            BOOL3 := FALSE;
2:          BOOL3 := TRUE;
ELSE
            BOOL6 := NOT BOOL1;
            BOOL7 := BOOL1 OR BOOL2;
END_CASE;
```

Sentencias de iteración

FOR: ejecuta instrucciones N veces

```
IVAR : INT ;
FOR IVAR := <VALOR_INICIAL> TO <VALOR_FINAL>  { BY <PASO> } DO
```

Condición de finalización se verifica antes de la ejecución de las Instrucciones

```
(* Instrucciones que se ejecutan si IVAR < VALOR_FINAL *)
```

Cada vez que se ejecutan las Instrucciones se incrementa IVAR por < PASO >

```
END_FOR;
```

Ejemplo 1:

```
FOR CONTADOR :=1 TO 5 BY 1 DO
        VAR1 := VAR1*2;
END_FOR;
```

Ejemplo 2: uso con EXIT

```
FOR CONTADOR := 1 TO 5 BY 1 DO
        VAR1 := VAR1*2;
        IF VAR1 > 5 THEN
                EXIT;
        END_IF;
END_FOR;
```

Evitando bucles infinitos: VALOR_FINAL debe estar en el rango de IVAR

Sentencias de iteración

WHILE: ejecuta instrucciones si el resultado de una expresión booleana es TRUE

WHILE < Expresión booleana >

La expresión booleana se verifica antes de la ejecución de las Instrucciones

(* Instrucciones que se ejecutan si la expresión booleana es TRUE *)

Las instrucciones no se ejecutan si la expresión booleana es FALSE en la primer iteración

END_WHILE;

Ejemplo :

```
WHILE CONTADOR <> 0  DO
  VAR := VAR * 2;
  CONTADOR := CONTADOR – 1;
END_WHILE;
```

Evitando bucles infinitos: hay que asegurar que ocurra la condición de salida, por ejemplo con un contador auxiliar

Sentencias de iteración

REPEAT: continúa la ejecución de instrucciones si el resultado de una expresión booleana es TRUE

REPEAT

(* Instrucciones *)

Las instrucciones se ejecutan una vez si la expresión booleana es FALSE en la primer iteración

UNTIL < Expresión booleana >

La expresión booleana se verifica después de la ejecución de las Instrucciones

END_REPEAT;

Ejemplo :

```
REPEAT
  VAR := VAR*2;
  CONTADOR := CONTADOR – 1;
UNTIL CONTADOR < 0
END_REPEAT;
```

Evitando bucles infinitos: hay que asegurar que la expresión booleana sea TRUE, por ejemplo con un contador auxiliar

Ejemplo

Se desea escribir un programa que controle el encendido - apagado de una bomba.

La bomba será encendida si:

1) Se pulsa el botón de arranque.

2) La protección térmica está deshabilitada.

3) Está abierto el botón de emergencia.

4) Está abierto el botón de parada.

Desde un tiempo T después del encendido, no puede haber ni sobre corriente ni baja corriente. Expresado de otra forma, desde un tiempo T después del arranque, la corriente I debe cumplir $I_{MIN} < I < I_{MAX}$, siendo I_{MIN} e I_{MAX} límites prefijados.

Ejemplo

El motor de la bomba se apagará si:

1) Se pulsa el botón de parada.
2) Se activa la protección térmica.
3) Se pulsa el botón de emergencia.
4) La corriente está fuera de los límites.

Ejemplo ST - Versión 1

```
(* Declaración de Variables *)
VAR
        ESTADO_OFF :BOOL := TRUE;
        ESTADO_ARRANQUE: BOOL;
        ESTADO_ON: BOOL;
        ...
END_IF;

TIMER_BOMBA(IN:=BOMBA_ON , PT:= TIEMPO_ARRANQUE, Q=> SALIDA_TIMER);

(* Estado OFF *)
IF ESTADO_OFF THEN
        BOMBA_ON := FALSE;
        IF ARRANQUE AND NOT PARADA AND NOT TERMICO AND NOT ALARMA THEN
                ESTADO_OFF := FALSE;
                ESTADO_ARRANQUE := TRUE;
        END_IF;
END_IF;
```

Estado Inicial

Acción

Transición

Cambio de Estado

Ejemplo ST - Versión 1

```
(* Estado ARRANQUE *)
IF ESTADO_ARRANQUE THEN
        BOMBA_ON := TRUE;
        (* Estado ARRANQUE a Estado OFF*)
        IF PARADA OR TERMICO OR ALARMA THEN
                ESTADO_ARRANQUE := FALSE;
                ESTADO_OFF := TRUE;
        END_IF;
        (* Estado ARRANQUE a Estado ON*)
        IF SALIDA_TIMER THEN
                ESTADO_ARRANQUE := FALSE;
                ESTADO_ON := TRUE;
        END_IF;
END_IF;
```

Acción

Transición

Cambio de Estado

Transición

Cambio de Estado

Acciones definidas dentro de los estados

Ejemplo ST - Versión 1

```
(* Estado ON *)
IF ESTADO_ON THEN
        ERROR :=  (CORRIENTE > CORRIENTE_MAX) OR (CORRIENTE < CORRIENTE_MIN);
        IF PARADA OR TERMICO OR ALARMA  OR ERROR THEN
                ESTADO_ON := FALSE;
                ESTADO_OFF:= TRUE;
        END_IF;
END_IF;
```

Ejemplo en ST – Versión 2

Declaración de variables del programa:

```
VAR
   DELAY: TIME  := T#5s;
   TIMER1: TON;
   ESTADO_APAGADO: BOOL:=TRUE;
   ESTADO_TRANSITORIO: BOOL;
   ESTADO_ENCENDIDO: BOOL;
   TIMEOUT_TIMER1: BOOL;
   CORRIENTE_MAX: INT := 120;
   CORRIENTE_MIN: INT := 50;
END_VAR
```

Estado Inicial

Ejemplo en ST – Versión 2

Declaración de variables globales:

```
VAR_GLOBAL
   ARRANQUE AT %IX4000.0 : BOOL;
   PARADA AT %IX4000.1 : BOOL;
   ALARMA AT %IX4000.2 : BOOL;
   TERMICO AT %IX4000.3 : BOOL;
(* ... *)
   BOMBA_ON AT %QX4000.0 : BOOL;
   CORRIENTE AT %IW0 : INT;
END_VAR
```

Ejemplo en ST – Versión 2

```
TIMER1 (IN := ESTADO_TRANSITORIO, PT := DELAY );
TIMEOUT_TIMER1 := TIMER1.Q;

IF (ESTADO_APAGADO AND ARRANQUE) THEN
   ESTADO_TRANSITORIO := TRUE;
   BOMBA_ON := TRUE;
   ESTADO_APAGADO := FALSE;
ELSIF (ESTADO_APAGADO AND (NOT ARRANQUE)) THEN
            BOMBA_ON := FALSE;
```

Ejemplo en ST – Versión 2

```
ELSIF (ESTADO_TRANSITORIO AND (TERMICO OR ALARMA OR PARADA)) THEN
        ESTADO_TRANSITORIO := FALSE;
        BOMBA_ON := FALSE;
        ESTADO_APAGADO := TRUE;
ELSIF (ESTADO_TRANSITORIO AND TIMEOUT_TIMER1)
THEN
        ESTADO_TRANSITORIO := FALSE;
        ESTADO_ENCENDIDO := TRUE;
```

Acciones definidas en las transiciones

Ejemplo en ST – Versión 2

```
ELSIF (ESTADO_ENCENDIDO AND (TERMICO OR ALARMA OR PARADA OR (CORRIENTE
   > CORRIENTE_MAX) OR(CORRIENTE<CORRIENTE_MIN)))
THEN
        ESTADO_ENCENDIDO := FALSE;
        BOMBA_ON := FALSE;
        ESTADO_APAGADO := TRUE;
END_IF;
(* END_IF termina sentencia IF para el diagrama de estados *)
```

Reset del PLC

- Reset (“Warm”) – apagar/encender el PLC o Online -> Reset
- Reset “Cold” – situación posterior a la descarga inicial al PLC
- Reset “Original” – borra el programa de usuario, lleva al PLC al estado original

Variables Retentivas

- VAR RETAIN
- VAR PERSISTENT

x = valores mantenidos

- = valores reinicializados

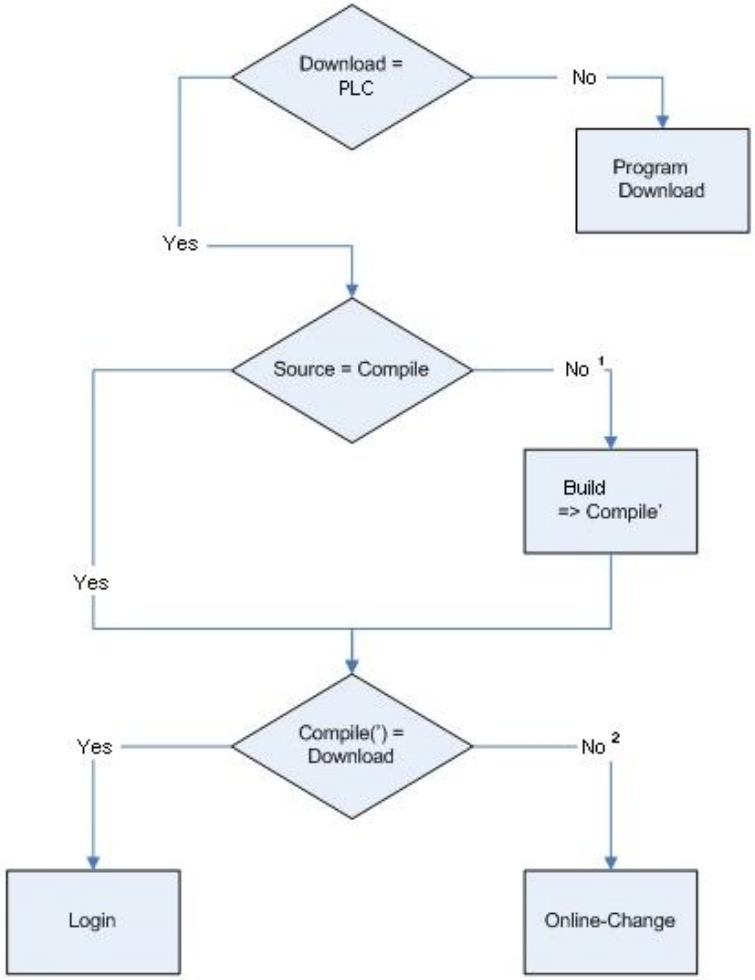
after Online command	VAR	VAR RETAIN	VAR PERSISTENT	VAR RETAIN PERSISTENT VAR PERSISTENT RETAIN
Reset	-	x	-	x
Reset cold	-	-	-	-
Reset origin	-	-	-	-
Download	-	-	x	x
Online Change	x	x	x	x

No funciona en la Simulación

Build-Download-Online

Archivos de código:

- Source: proyecto actual (*.pro)
- Compile: compilado del último “build” (*.ci)
- Download: información de la última carga al PLC (*.ri)
- PLC: proyecto actualmente en el PLC (*.prg)



[1]: e.g. changed libraries
[2]: e.g. changed compiler version or changed program code