Capa de Aplicación en PLC: Secuencias

Implementación

⚠️ Post técnico

Empecemos

Clasificación

TIPOS DE SECUENCIA

TIPOS DE EJECUCIÓN

TIPOS DE IMPLEMENTACIÓN

TIPOS DE VARIABLE DE ESTADO

1

2

3

4

5

Tipos de Secuencia

Tipos de Secuencia

GRAFCET

Estados

Transiciones

(Gráfico Funcional de Etapas y Transiciones) es un lenguaje gráfico formal utilizado para describir el comportamiento secuencial de sistemas automatizados. La norma **IEC 60848** especifica este lenguaje como **Sequential Function Chart (SFC)**.

GRAFO

Vértices

Aristas

Un grafo es una estructura matemática que representa un conjunto de objetos y las relaciones entre ellos. Es una herramienta fundamental en matemáticas discretas, informática, y muchas otras disciplinas.

MÁQUINA DE ESTADOS

Estados

Transiciones

Una Máquina de Estados (**Finite State Machine - FSM**) es un modelo matemático abstracto que describe el comportamiento de un sistema que puede estar en un solo estado a la vez y transiciona entre estados en respuesta a eventos o entradas.

Tipos de Ejecución

CASE o IF-ELSEIF

Cuando se produce una transición, el cambio de estado se efectúa en el **siguiente ciclo** de scan.

IF-IF

Cuando se produce una transición, el cambio de estado se efectúa en el **mismo ciclo** de scan.

Tipos de Implementación →

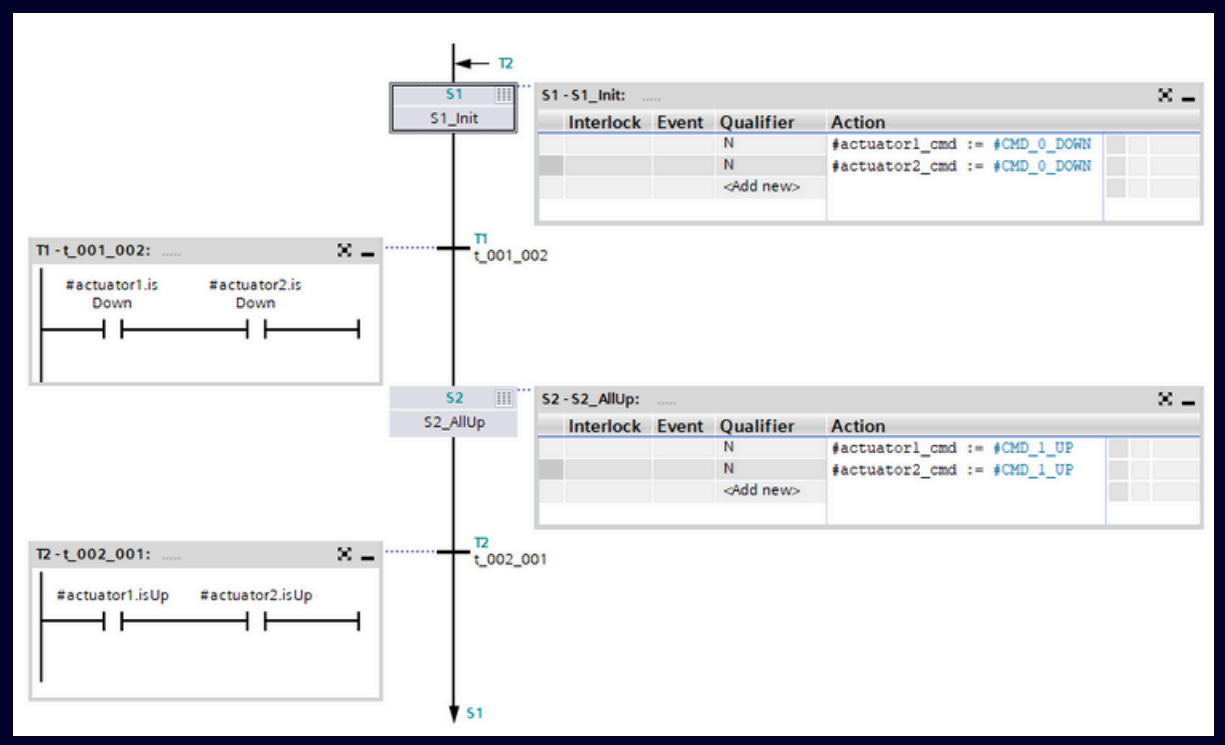
Graph (SFC)

🏃 Tipo de ejecución

- CASE

💾 Tipo de variable de estado

- BOOL por estado, permite paralelismo.
- INT para el estado actual y BOOL cuando hay más de un estado activo (paralelismo).



🤔 ¿Cuándo usarlo?

- LA CPU tiene recursos suficientes.
- El Grafcet es complejo.
- Mueves varios actuadores.
- La ejecución en varios ciclos no penaliza la aplicación.

⚠️ ¡Peligro!

- Si usas muchos Graphs puede afectar al ciclo de scan ya que necesita muchos recursos para su ejecución.

LAD/FBD →

LAD/FBD

🏃 Tipo de ejecución

- IF-IF

💾 Tipo de variable de estado

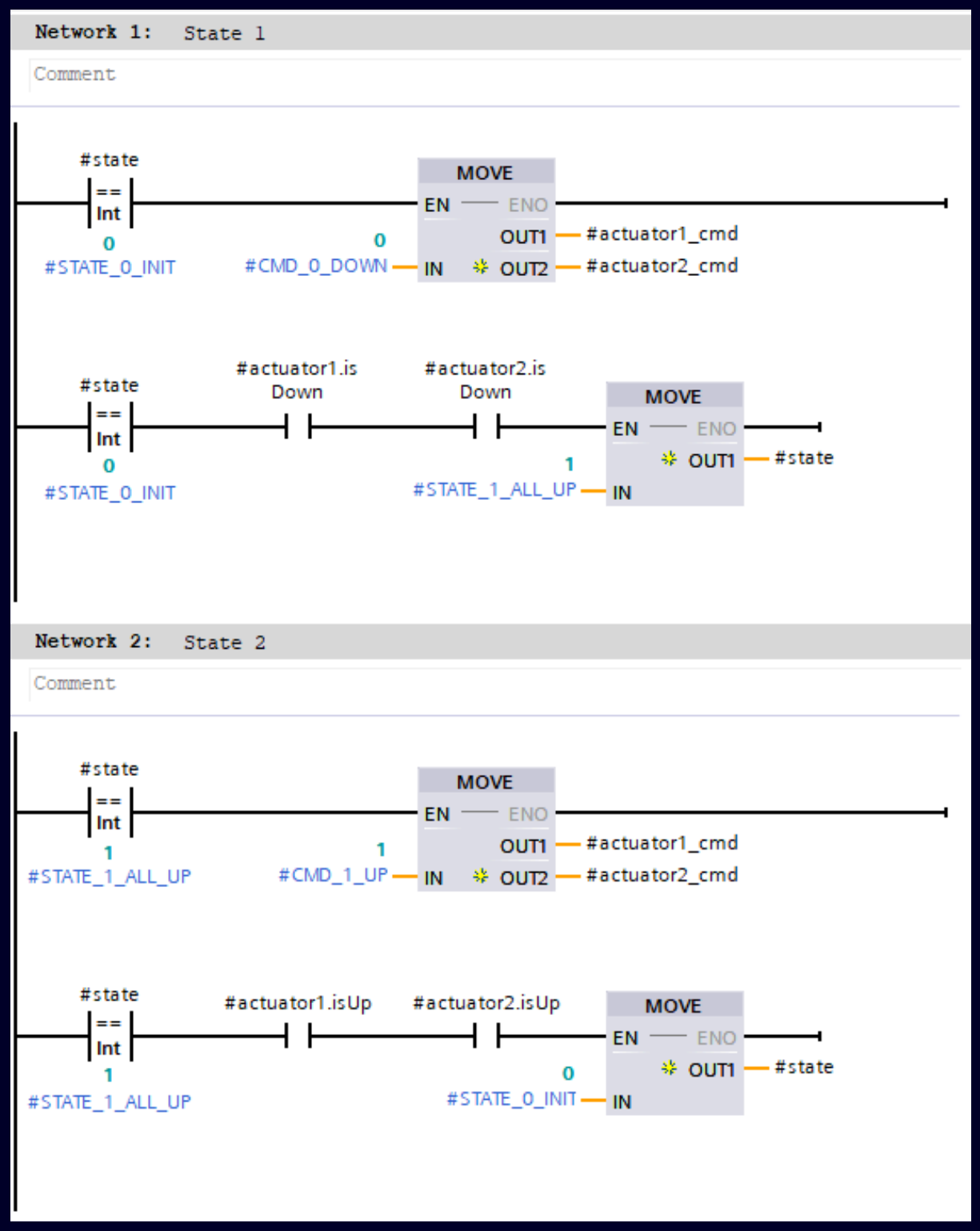
- INT para el estado actual y BOOL cuando hay más de un estado activo (paralelismo).

🤔 ¿Cuándo usarlo?

- Vives en los años 80/90.
- No tienes otro lenguaje para hacerlo.
- Mueves varios actuadores.

⚠️ ¡Peligro!

- La depuración se complica cuando la secuencia es grande y compleja ya que todas las expresiones LAD son evaluadas en cada ciclo de scan.



LAD/FBD con saltos →

LAD/FBD con saltos

🏃 Tipo de ejecución

- CASE: Si usas lista de saltos
- IF-IF: Si usas saltos por estado

💾 Tipo de variable de estado

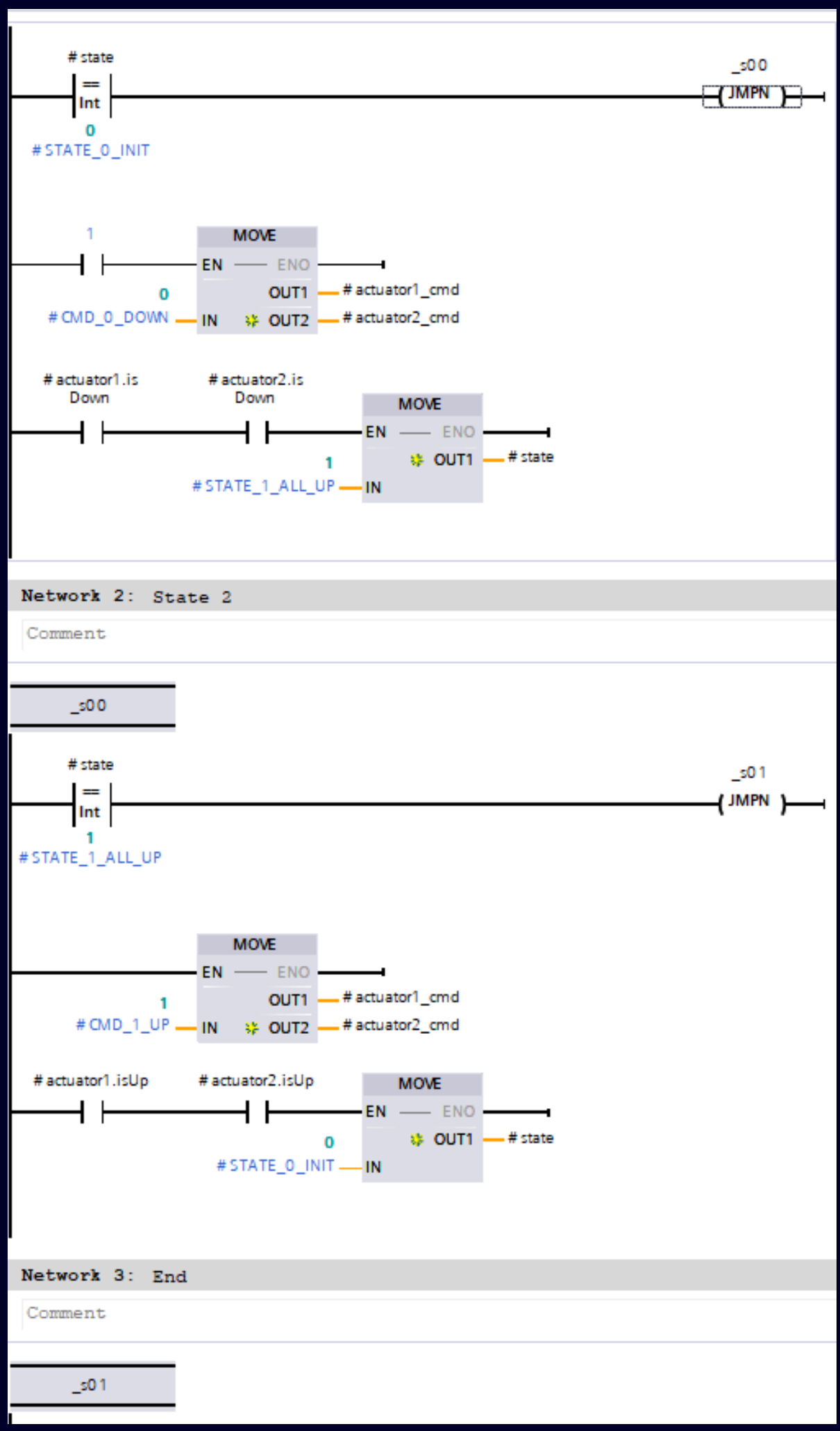
- INT para el estado actual y BOOL cuando hay más de un estado activo (paralelismo).

🤔 ¿Cuándo usarlo?

- Vives en los años 80/90.
- Quieres que te odien. 😆
- No tienes otro lenguaje para hacerlo.
- La verdad, no entiendo el motivo, he visto estas implementaciones en muchas empresas serias y todas han sido fuente de problemas graves.

⚠️ ¡Peligro!

- La depuración es **dramática**, aprendes a tener paciencia.



STL →

STL

🏃 Tipo de ejecución

- CASE: Si usas lista de saltos
- IF-IF: Si usas saltos por estado

💾 Tipo de variable de estado

- INT para el estado actual y BOOL cuando hay más de un estado activo (paralelismo).

🤔 ¿Cuándo usarlo?

- Necesitas velocidad de ejecución.
- Necesitas optimizar código.

⚠️ ¡Peligro!

- La depuración se **complica** cuando la seucencia es grande y compleja, es recomendable partirla en secuencias más pequeñas.

```
// State 0
      L     #state
      L     #STATE_0_INIT
      ==I
      JCN   _s00

// Associated actions
      L     #CMD_0_DOWN
      T     #actuator1_cmd
      T     #actuator2_cmd

// Transition
      A     #actuator1.isDown
      A     #actuator2.isDown
      JCN   _s00

      L     #STATE_1_ALL_UP
      T     #state

_s00: NOP 0
```

Network 2: State 1

Comment

```
// State 1
      L     #state
      L     #STATE_1_ALL_UP
      ==I
      JCN   _s01

// Associated actions
      L     #CMD_1_UP
      T     #actuator1_cmd
      T     #actuator2_cmd

// Transition
      A     #actuator1.isUp
      A     #actuator2.isUp
      JCN   _s01

      L     #STATE_0_INIT
      T     #state

_s01: NOP 0
```

SCL →

SCL

🚶 Tipo de ejecución

- CASE: Si usas Case o Elseif.
- IF-IF: Si usas IF.

💾 Tipo de variable de estado

- INT para el estado actual y BOOL cuando hay más de un estado activo (paralelismo).

🤔 ¿Cuándo usarlo?

- La mayoría de aplicaciones.
- Comunicaciones IT.
- Secuencias complejas.

⚠️ ¡Peligro!

- La complejidad de la depuración es proporcional a la complejidad de la secuencia.

```
CASE #state OF
  #STATE_0_INIT:  // State 0
    // Associated actions
    #actuator1_cmd :=
    #actuator2_cmd := #CMD_0_DOWN;

    // Transition
    IF #actuator1.isDown AND #actuator2.isDown THEN
      #state := #STATE_1_ALL_UP;
    END_IF;

  #STATE_1_ALL_UP:  // State 1
    // Associated actions
    #actuator1_cmd :=
    #actuator2_cmd := #CMD_1_UP;

    // Transition
    IF #actuator1.isUp AND #actuator2.isUp THEN
      #state := #STATE_0_INIT;
    END_IF;

END_CASE;
```

```
// State 0
IF #state = #STATE_0_INIT THEN
  // Associated actions
  #actuator1_cmd :=
  #actuator2_cmd := #CMD_0_DOWN;

  // Transition
  IF #actuator1.isDown AND #actuator2.isDown THEN
    #state := #STATE_1_ALL_UP;
  END_IF;
END_IF;

// State 1
IF #state = #STATE_1_ALL_UP THEN
  // Associated actions
  #actuator1_cmd :=
  #actuator2_cmd := #CMD_1_UP;

  // Transition
  IF #actuator1.isUp AND #actuator2.isUp THEN
    #state := #STATE_0_INIT;
  END_IF;
END_IF;
```

```
// State 0
IF #state = #STATE_0_INIT THEN
  // Associated actions
  #actuator1_cmd :=
  #actuator2_cmd := #CMD_0_DOWN;

  // Transition
  IF #actuator1.isDown AND #actuator2.isDown THEN
    #state := #STATE_1_ALL_UP;
  END_IF;

// State 1
ELSIF #state = #STATE_1_ALL_UP THEN
  // Associated actions
  #actuator1_cmd :=
  #actuator2_cmd := #CMD_1_UP;

  // Transition
  IF #actuator1.isUp AND #actuator2.isUp THEN
    #state := #STATE_0_INIT;
  END_IF;
END_IF;
```

Para finalizar →

¿Y tú?

¿Cómo implementas tus secuencias?

¿Tienes encuenta el ciclo de scan, paralelismo u optimización?

👇 ¡Nos vemos en los comentarios!

#PLC #Graph #LAD #STL #SCL

⚠️ Continuará...