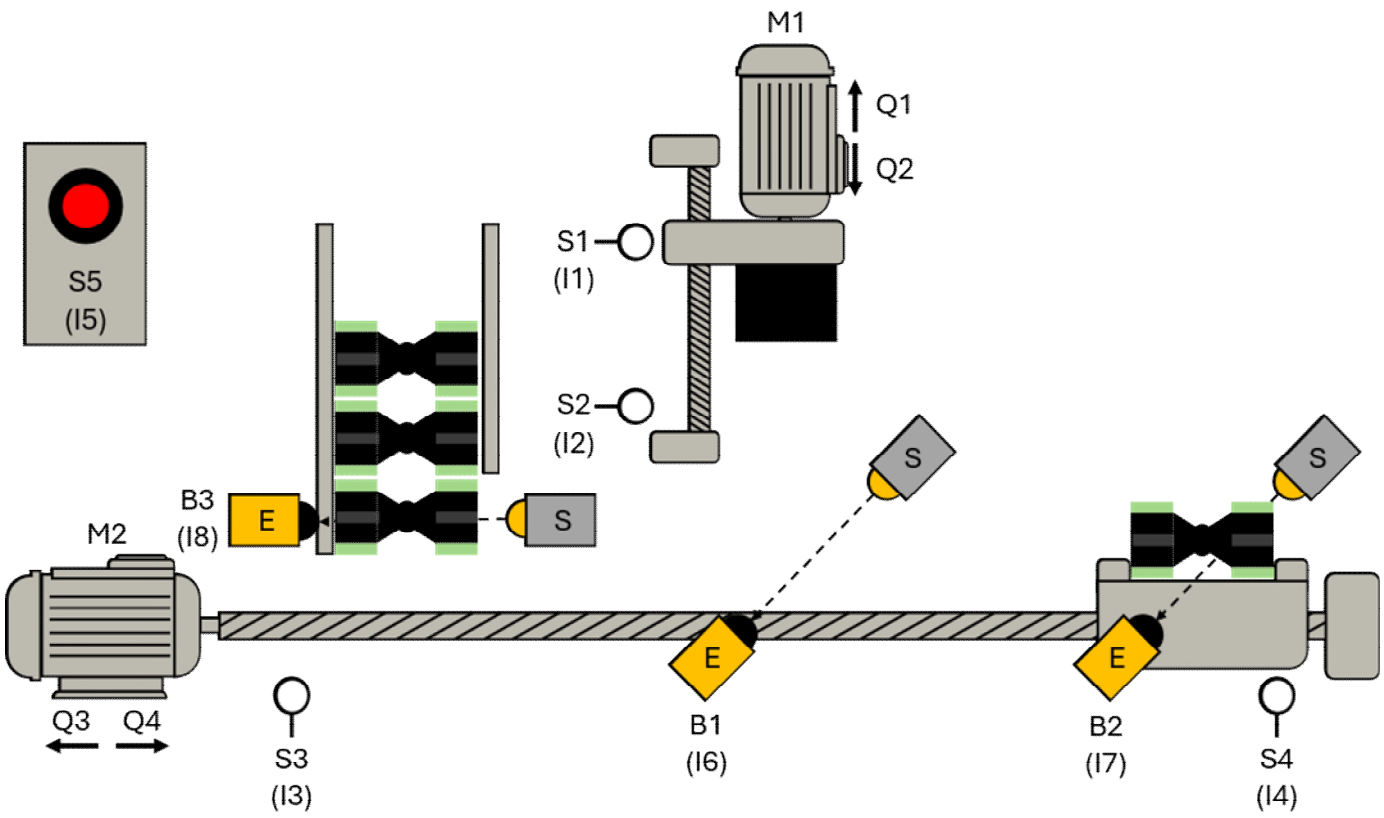
fischertechnik

GROLLMUS
KNOWLEDGE | SOFTWARE | AUTOMATION

Prensa Dobladora 24V

Conversión GRAFCET a código de programa



Índice

9 Conversión GRAFCET a código de programa

9.1 Descripción funcional de la cadena de procesos

Ahora debe crearse un programa PLC a partir del GRAFCET dado para un sistema de ejemplo. El procedimiento básico para convertir un Grafcet en código de programa se ilustra a continuación mediante una sencilla cadena de pasos:

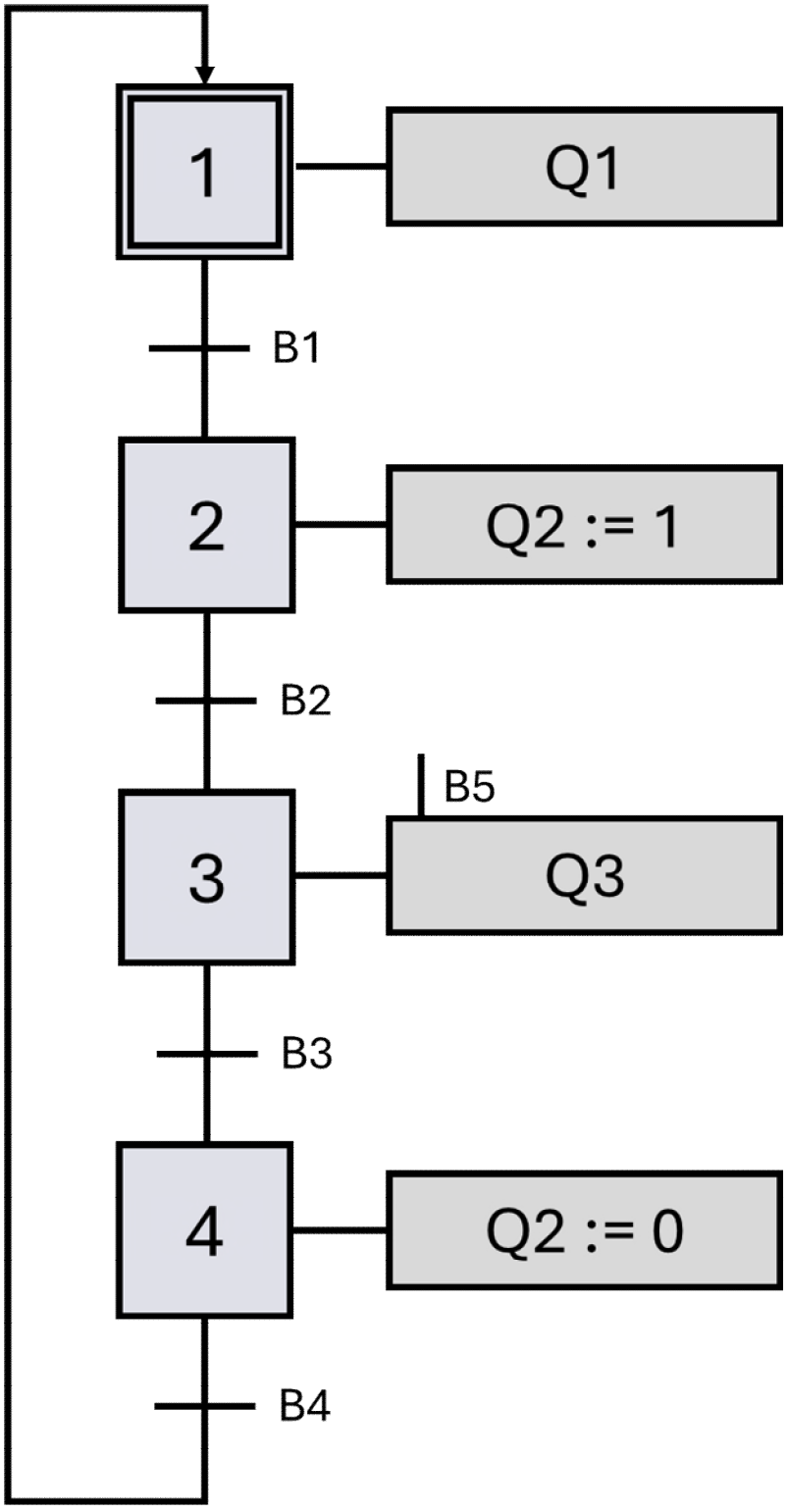


Imagen 1 Ejemplo de cadena GRAFCET

Descripción funcional

Paso 1 (paso inicial)

El paso 1 es el paso inicial.
Mientras el sistema se encuentra en este paso, el actuador Q1 se activa mediante una acción de acción continua.
Si el sensor emite una señal B1 1, el sistema salta al paso siguiente.

Paso 2

El actuador Q2 se activa al guardar ("TRUE").
Si el sensor B2 emite una señal 1, el sistema salta a la etapa siguiente.

Paso 3

Mientras el sistema se encuentre en este paso, el actuador Q3 se activa mediante una acción de acción continua con condición. Para que el actuador se controle con el valor "VERDADERO", el sensor B5 también debe suministrar una señal 1 además del paso activo.
Si el sensor B3 emite una señal 1, el sistema salta al paso siguiente.

Paso 4

El actuador Q2 se restablece ("FALSE").
Si el sensor B4 emite una señal 1, el sistema salta de nuevo al paso inicial.

9.2 Implantar la cadena de procesos GRAFCET en FUP

En esta sección, implementaremos la cadena de secuencias planificada en GRAFCET en el lenguaje de programación FBD.

9.2.1 Implementar pasos y transiciones en FUP

El flip-flop de reinicio dominante es el elemento central en la implementación de la cadena de secuencia GRAFCET en FBD. Un paso activo se representa mediante un flip-flop activado. Un paso se activa a través de la entrada set y, por lo tanto, puede considerarse como parte de la transición anterior. Un paso activo se abandona a través de la entrada de reset y, por lo tanto, forma parte de la transición posterior. Un marcador de paso independiente que representa el paso activado se conecta como memoria para el flip-flop.
Como variables para los indicadores de paso se pueden utilizar indicadores globales, variables de bloques de datos globales o variables locales del área estática de la interfaz del bloque.

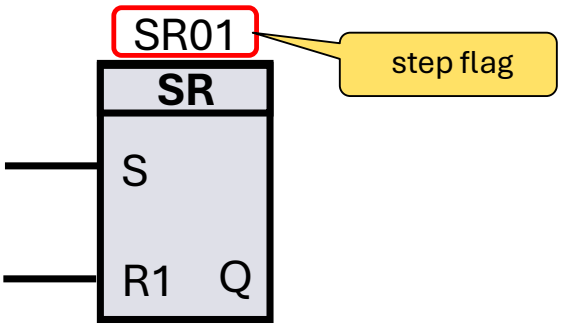


Imagen 2 Flip-flop con marcador de pasos

Paso inicial
El paso 1 es el paso inicial, que debe activarse cuando se inicializa la cadena (Init). El paso también se activa si la cadena se encuentra en el último paso (SR04) y se cumple la transición posterior (B4).

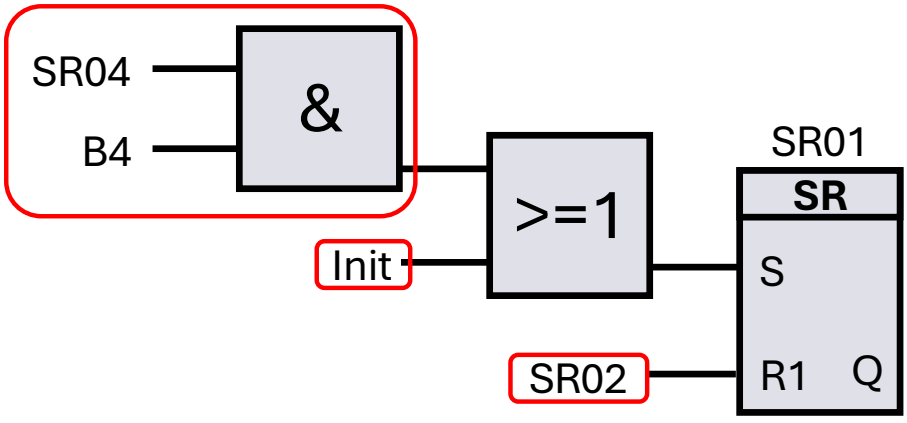


Imagen 3 Paso inicial de programación

El paso debe restablecerse si el paso siguiente (SR02) está activo.

Escalón estándar

Todos los pasos posteriores se activan si se cumplen el paso anterior y la transición correspondiente. Se restablecen cuando se activa el paso posterior o se inicializa la cadena.

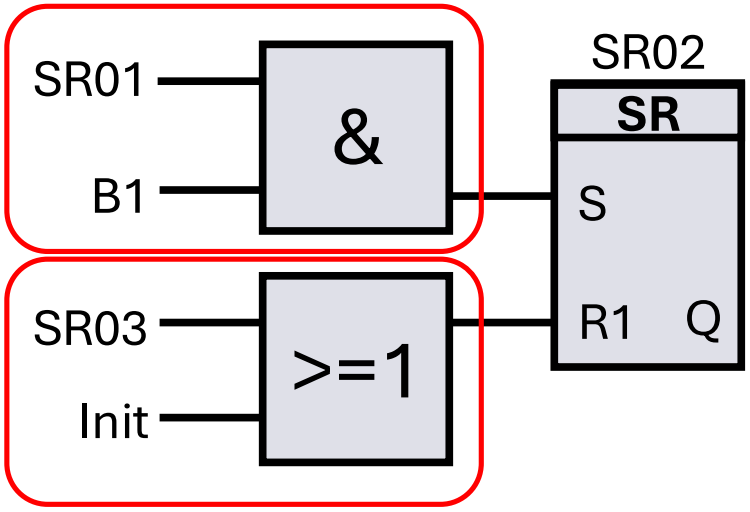


Imagen 4 Paso de programación

El último paso va seguido de un salto al primer paso de la cadena, por lo que el último paso se reinicia con el primero, ya que éste es el paso posterior.

El resultado es la siguiente cadena de pasos para el GRAFCET que se muestra al principio:

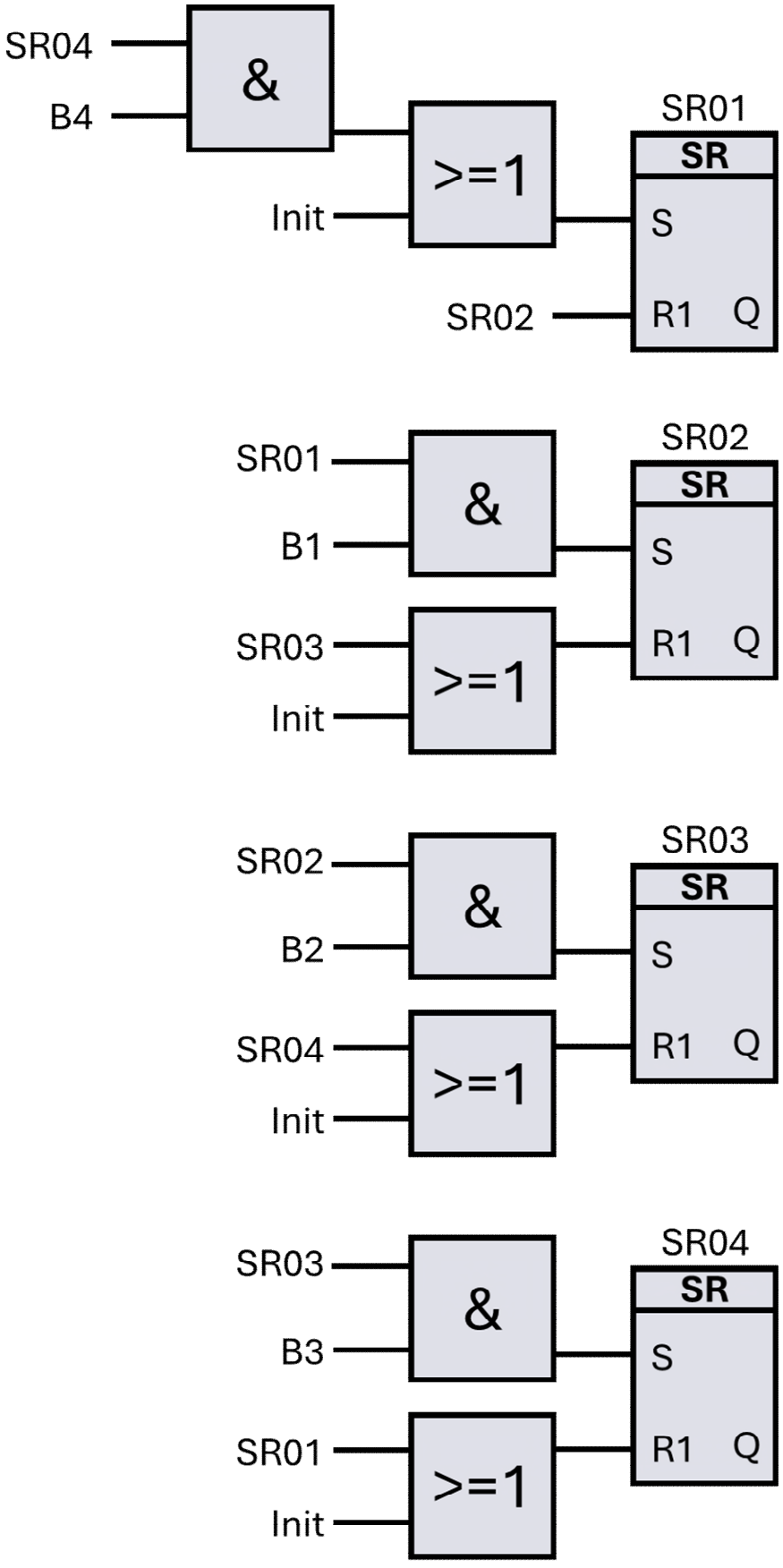


Imagen 5 Cadena de pasos de programación

9.2.2 Ejecutar las acciones de la FUP

Debido a aspectos de estructuración funcional, las acciones en los respectivos pasos de la cadena de secuencias pueden programarse en un módulo de control separado o directamente en las redes inmediatamente posteriores a la cadena en el mismo módulo.

Programación en un módulo de control independiente

Una opción es programar las acciones de los pasos en un módulo de control independiente. Este método tiene la ventaja de que la lógica de control de la cadena de pasos y la ejecución de las acciones están claramente separadas entre sí. Esto aumenta la claridad del programa, lo que resulta especialmente ventajoso en sistemas de control complejos. Esta separación también facilita el mantenimiento y la ampliación del programa, ya que las modificaciones de la lógica de control o de las acciones pueden realizarse independientemente unas de otras.

En este enfoque, la cadena de pasos se implementa en el módulo principal. Cuando un paso se activa, se envía una señal al módulo de control, que ejecuta las acciones correspondientes. Las banderas de paso suelen utilizarse como señales para el intercambio. Esta estructura permite una programación modular en la que cada módulo tiene una tarea claramente definida.

Programación en las redes directamente después de la cadena de pasos

Un método alternativo consiste en programar las acciones directamente en las redes inmediatamente después de la cadena de pasos en el mismo bloque. Este enfoque tiene la ventaja de que toda la lógica se concentra en un solo lugar, lo que facilita el seguimiento de la ejecución del programa. Puede ser una solución más eficaz y rápida, sobre todo para proyectos pequeños y menos complejos.

Aquí, las acciones se programan directamente en las redes que siguen a los pasos correspondientes en la cadena de pasos. Este método puede optimizar el tiempo de ejecución del programa, ya que no se requieren llamadas adicionales a bloques y la estructura del programa sigue siendo compacta.

Acción continua en FUP:

Por defecto:

La acción se ejecuta mientras el paso esté activo.

Realización:

El marcador de paso correspondiente se escribe directamente en la salida mediante una asignación.

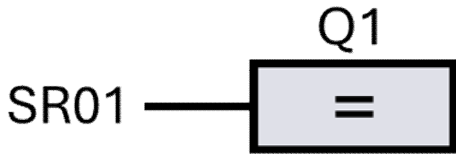


Imagen 6 Asignación continua

Asignación continua con condición de asignación adicional:

Por defecto:

La acción se ejecuta mientras el paso esté activo y se cumpla la condición de asignación.

Realización:

El marcador de paso correspondiente se vincula lógicamente a la condición de asignación y el resultado de la vinculación se asigna a la salida.

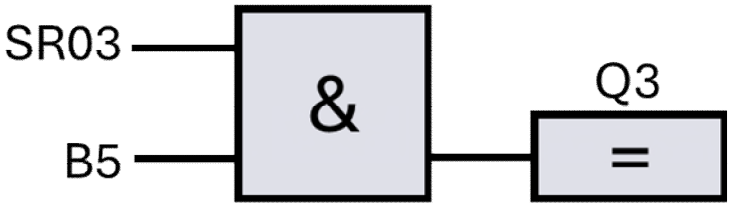


Imagen 7 Asignación continua con condición

Acción salvadora:

Por defecto:

La acción guardada permanece activa durante varios pasos.

Realización:

Las banderas de paso correspondientes para establecer y restablecer la acción están conectadas a un flip-flop. El flip-flop almacena la salida.

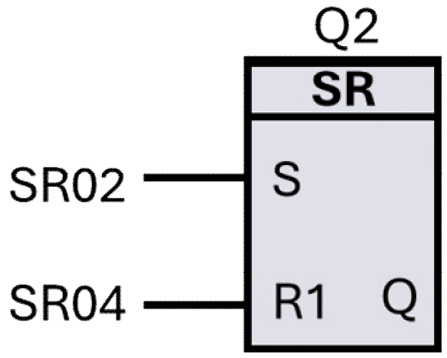


Imagen 8 Acción de almacenamiento

Cuando se inicializa la cadena, la acción puede permanecer establecida ya que el paso 4 (SR04) no se ha activado. Para evitarlo, la variable de la solicitud de

inicialización (Init) también puede conectarse a la entrada de reset mediante un enlace OR.

9.3 Implementar la cadena de procesos GRAFCET en ST / SCL

En esta sección, implementaremos la cadena de secuencias planificada en GRAFCET en el lenguaje de programación ST / SCL.

Se recomienda utilizar una estructura CASE para asignar los pasos individuales del GRAFCET. Para ello, primero debe definirse una variable de índice con un tipo de datos entero (por ejemplo, INT). Esta es la variable de recuento que representa el paso actual.

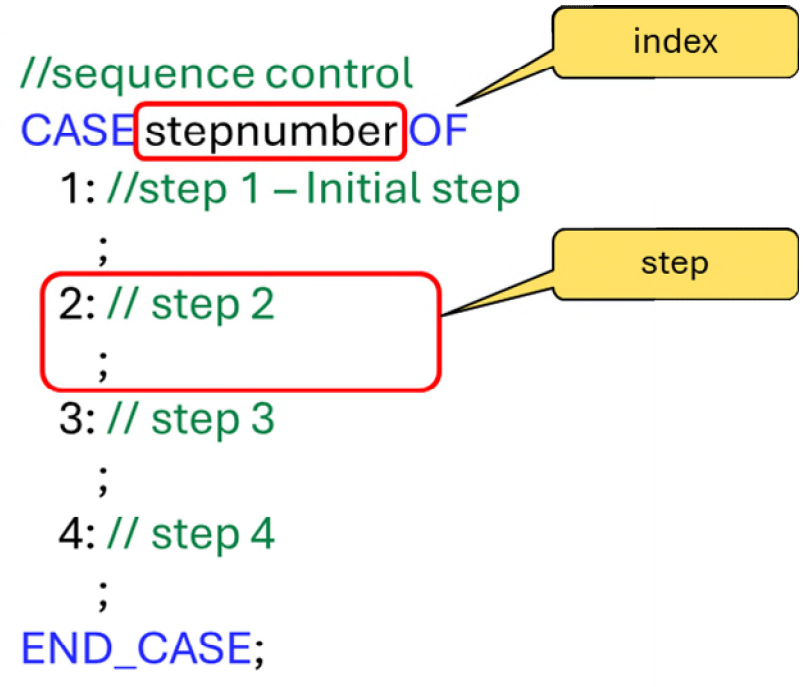


Imagen 9 Estructura CASE

9.3.1 Inicialización de la cadena de procesos

El paso 1 es el paso inicial, que debe activarse al inicializar la cadena (Init). Para ello, debe colocarse una sentencia IF antes de la estructura CASE, que resetea la variable índice (número de paso) a 1 cuando se solicita la inicialización.

```
//initialize sequence control
IF Init THEN
   stepnumber := 1;
END_IF;
```

Imagen 10 Inicialización

9.3.2 Estructura de un paso (CASE)

Cada paso (CASE) se estructura según el siguiente esquema:
En primer lugar, se implementan las acciones que deben realizarse en el paso. A esto le sigue una sentencia IF, que representa la transición y establece la variable de índice de la cadena de pasos en el paso siguiente si se cumplen las condiciones de transición.

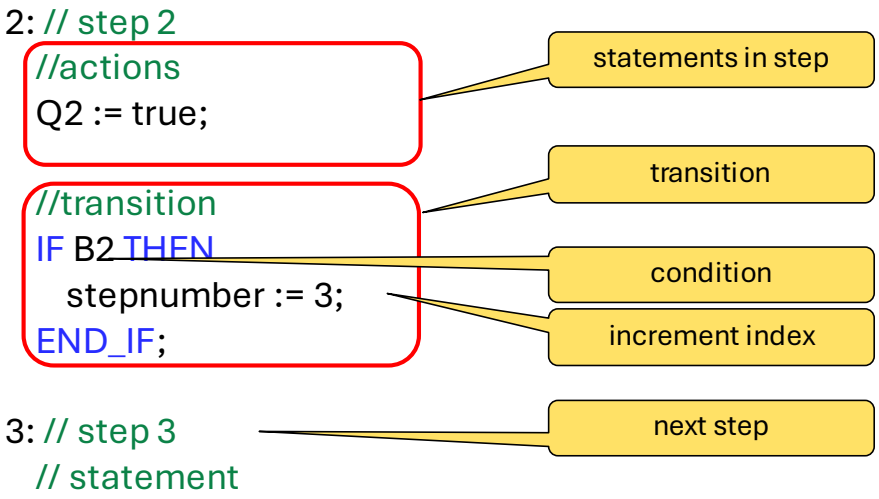


Imagen 11 Paso de estructura

En función de la acción prevista, deben aplicarse de forma diferente:

Asignación continua:
Por defecto
La acción se ejecuta mientras el paso esté activo.
Realización
Asignando el estado de señal "TRUE" a la variable, se consigue un comportamiento de ahorro. Sin embargo, esto no se desea en este momento. Por lo tanto, la acción debe reiniciarse al salir del paso, lo que sigue al cumplimiento de la condición de transición.

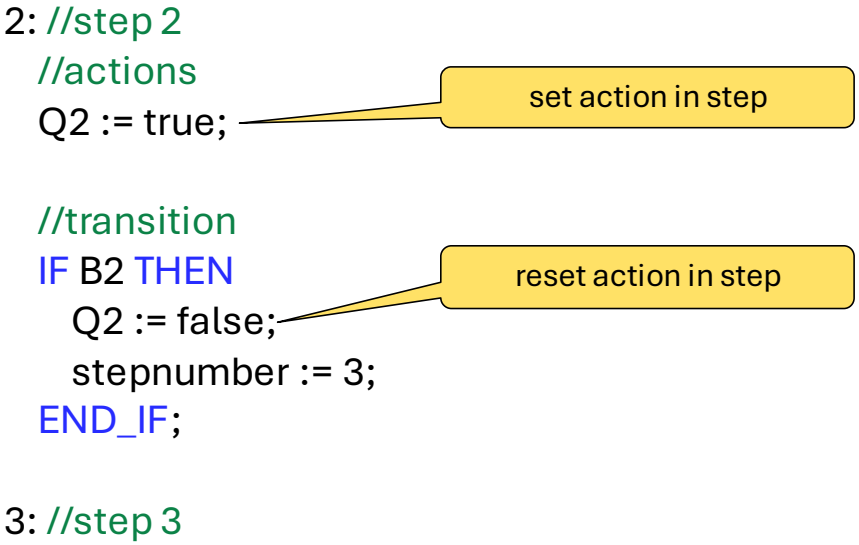


Imagen 12 Asignación continua con condición

Asignación continua con condición de asignación adicional:

Por defecto

La acción se ejecuta mientras el paso esté activo y se cumpla la condición de asignación.

Realización

La condición de asignación adicional puede asignarse directamente a la acción.

Al igual que en la asignación continua, la acción correspondiente también debe anularse al salir del paso.

```
3: //step 3
  //action
  Q3 := B5;

  //transition
  IF B3 THEN
    Q3 := false;
    stepnumber := 4;
  END_IF;

4: //step 4
```

set action in step, depending on the signal of the assignment condition

reset action in step

Fotografía 13 Asignación continua con condición

Acción salvadora:

Por defecto

Con las acciones almacenadas, la acción se ejecuta en varios pasos.

Realización

En principio, todas las acciones son acciones con efecto de ahorro debido a la asignación directa de los estados de las señales. Si se desea este comportamiento, la acción no debe restablecerse en la sentencia IF de la transición, sino en el paso correspondiente.

```
2: //step 2
    //action
    Q2 := true;

    //transition
    IF B2 THEN
      stepnumber := 3;       //next step
    END_IF;

  4: //step 4
    //action
    Q2 := false;             //reset action

    //transition
    IF B4 THEN
```

set action in step

reset action in step

Imagen 14 Acción de almacenamiento

El resultado es la siguiente cadena de pasos para el GRAFCET que se muestra al principio:

```
//initialize sequence control
IF Init THEN
  stepnumber := 1;
END_IF;

//sequence control
CASE stepnumber OF
  1: //step 1 – Initial step
    //action
    Q1 := true;              //set action

    //transition
    IF B1 THEN
      Q1 := false;           //reset action
      stepnumber := 2;       //next step
    END_IF;

  2: //step 2
    //action
    Q2 := true;

    //transition
    IF B2 THEN
      stepnumber := 3;       //next step
    END_IF;

  3: //step 3
    //action
    Q3 := B5;                //set action as long as B5 is active

    //transition
    IF B3 THEN
      Q3 := false;           //reset action
      stepnumber := 4;       //next step
    END_IF;

  4: //step 4
    //action
    Q2 := false;             //reset action

    //transition
    IF B4 THEN
      stepnumber := 1;       //jump to step 1
    END_IF;
END_CASE;
```

Fotografía 15 Cadena de pasos de programación