Grado en Ingeniería Electrónica Industrial y Automática

AUTOMATIZACIÓN
PRÁCTICA III

Pablo San Segundo (Universidad Politécnica de Madrid -UPM)

3. TRAMOS DE ESCALERAS MECÁNICAS

3.1. REQUISITOS PREVIOS

Para la realización de esta tercera práctica resultará imprescindible una lectura de los siguientes capítulos del libro "*Programación de autómatas con STEP 7*":

- *Capítulo X*: Funciones y bloques función; paso de parámetros
- *Capítulo XI*: Fundamentos de la programación estructurada. En concreto, el segundo caso práctico (sección 11.5) es el objeto de estudio de esta práctica

3.2. LIBRERÍAS DE BLOQUES

El entorno STEP 7 dispone de un conjunto de librerías de bloques ya programados que acompañan a la distribución. Algunos de ellos se emplearon en la práctica anterior, como la función SCALE de la librería *Standard Library->TI-S7 Converting Blocks*.

Es importante destacar que el usuario del entorno también puede crear (y usar en sus programas STEP 7) sus propias librerías. Para ello basta con crear una *librería nueva*, en lugar de un proyecto como se ha hecho hasta ahora: VentanaPrincipal: Archivo→Nuevo→Librerías (pestaña). La nueva librería de usuario se instala en el directorio por defecto ~*Step7**S7libs* y, desde ahí, son accesibles al programador desde la ventana de edición de bloques. Como ejemplo, la figura 1 muestra una librería de usuario *libmaniobras* pensada para contener bloques que permitan controlar maniobras simples.

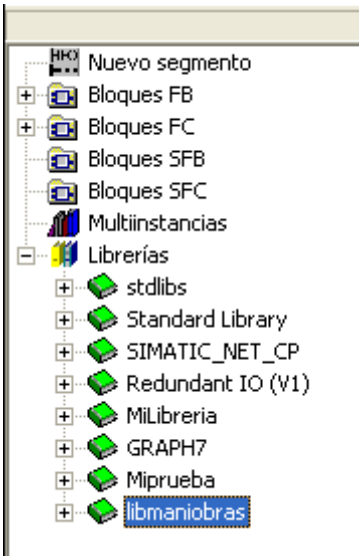

Figura 1. Sombreada, una librería de usuario (no acompaña a la distribución)

3.3. BLOQUES PARAMETRIZADOS PARA MANIOBRAS SIMPLES

Muchas acciones de control implican el desplazamiento de un elemento móvil (un cilindro, un AGV etc.) desde un punto a otro determinados por los respectivos sensores inicio y final de carrera. Esta acción de control puede generalizarse en una función parametrizada más o menos compleja. Un posible interfaz completo para el control de una maniobra con un tiempo límite de duración predefinido se muestra en la tabla 1.

Tabla 1. Interfaz de una maniobra simple con control de error

VAR_INPUT sfin : BOOL ; temporizador : TIMER ; tiempo : S5TIME ; **END_VAR** **VAR_OUTPUT** actuador_ON : BOOL ; bitDeFin : BOOL ; bitDeError : BOOL ; **END_VAR** **VAR_STATIC** bitDeTrabajo : BOOL ; **END_VAR**	**VAR_INPUT** sfin : BOOL ; temporizador : TIMER ; tiempo : S5TIME ; **END_VAR** **VAR_OUTPUT** actuador_ON : BOOL ; bitDeFin : BOOL ; **END_VAR** **VAR_STATIC** bitDeTrabajo : BOOL ; **END_VAR**
FB: "Maniobra con bit de error explícito"	**FB: "Maniobra que usa el biestable RB para la gestión del error"**

En ambos bloques se asume que el sensor de inicio de carrera está activo y se ejecuta el movimiento hasta su final de carrera (parametrizado como *sfin*). Si no se sobrepasa un tiempo límite (*tiempo*) se activa *bitDeFin* en la salida para indicar que la acción de

control terminó correctamente; en caso contrario se activa *bitDeError*. Observe que para que el bloque sea lo más general posible se incluye también el propio temporizador como parámetro de entrada (*temporizador*) que es de tipo TIMER. La variable STATIC *BitDeTrabajo* se corresponde con la plantilla de la sección 11.2 del libro de texto y permite ejecutar el segmento de arranque. Se ha definido de tipo STATIC para que no aparezca en el interfaz de la llamada a la función, simplificando así la tarea al usuario del bloque.

El interfaz de la izquierda se corresponde con una gestión explícita del evento error por sobrepasar un tiempo límite. El interfaz de la derecha emplea el biestable RB de la palabra de estado (plantilla EN-ENO sección 10.2.4 del libro de texto). Dicho bit se coloca a nivel bajo cuando se detecta el error y es el bloque invocante el que debe gestionarlo adecuadamente.

IMPORTANTE: El texto fuente de ambos bloques acompaña al guión de la práctica. Si no lo tiene pídalo al profesor en el aula.

TAREA 1. Se pide crear una librería *libmaniobra* con las dos funciones y crear dos proyectos S7, uno para cada función, que utilicen los bloques FB para implementar una expansión controlada de un cilindro neumático. El resultado se mostrará al profesor en PLCSIM al principio de la práctica.

3.4. TRAMOS DE ESCALERAS MECÁNICAS

Como sistema a controlar en esta práctica se proponen dos tramos de escaleras mecánicas consecutivos que permiten a un usuario superar un desnivel de altura. El sistema está descrito y modelado en la sección 11.5 del libro de texto y el esquema de componentes se muestra en la figura 2. Para esta práctica se considerarán las siguientes especificaciones:

BOTONERA

Se dispone de pulsadores ON, OFF y EMERGENCIA (Pon, Poff y PEMER respectivamente).

COMPONENTES

El sistema consta de dos tramos de escaleras TR1 y TR2 accionadas por motores. Asimismo consta de una barrera telescópica con accionamiento de subida y bajada. La sensórica incluye dos sensores mecánicos de inicio y fin de carrera para la barrera (SBS, SBB) y dos sensores ópticos (SE1 y SE2) que detectan el paso de personas, uno en cada plataforma.

CICLO DE FUNCIONAMIENTO

El primer tramo de la escalera sólo empieza a funcionar cuando se activa el fotosensor 1 (SE1) con la llegada de un usuario. De manera análoga, el segundo tramo sólo se activa tras la detección de presencia en el fotosensor 2 (SE2). Para ahorrar energía, cada tramo se parará *cuando hayan transcurrido al menos 50 segundos sin activarse su sensor correspondiente*. Obsérve que esto implica un tiempo de subida nominal por tramo sensiblemente menor a ese tiempo.

Con el flanco de subida de la señal ON se baja la barrera de entrada telescópica que da acceso a la escalera, quedando ésta en situación normal de funcionamiento cuando la barrera alcanza SBB.

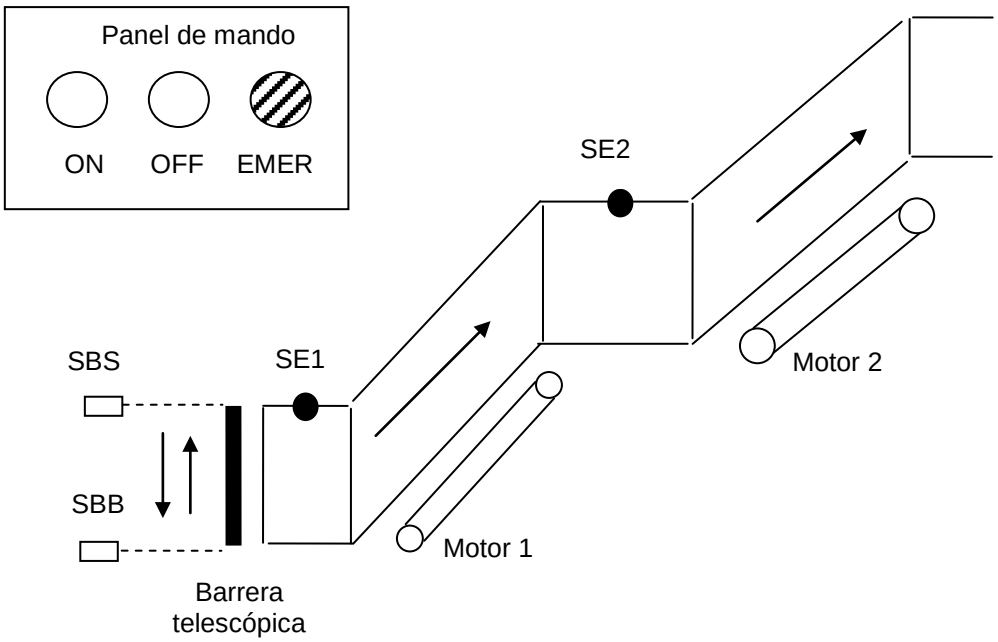

Figura 2. Sistema constituido por dos tramos de escaleras mecánicas

PARADA DE CICLO

La *parada de ciclo* (nivel alto del pulsador Poff) tiene las siguientes características:

- La barrera sube nada mas detectar la parada hasta su sensor fin de carrera SBS
- Los dos tramos de escaleras funcionan normalmente, *hasta que todos los usuarios presentes en la escalera en ese momento hayan alcanzado la plataforma superior*. Posteriormente, el sistema pasa al reposo.

EMERGENCIA

La activación de la seta de emergencia (PEMER) detiene toda acción. Existen además emergencias locales para las maniobras de subida y bajada de la barrera si se sobrepasa un tiempo límite de 10s. Esta situación de defecto local no altera el funcionamiento de los motores (p. ej. las personas dentro del recinto son evacuados con normalidad si la subida de la barrera falla).

Tras pulsar PEMER el sistema pasa a defecto. El rearme al reposo se produce tras desenclavar la seta de emergencia y comprobar que el resto de alarmas locales están resueltas.

Un posible modelado del ciclo básico (sin incluir los estados de defecto globales y locales) podría ser el de la figura 3 (no es la única opción). Observe la notación moderna SFC para el modelado de los actuadores Subir y Bajar barrera (figura 5.10 del libro de texto).

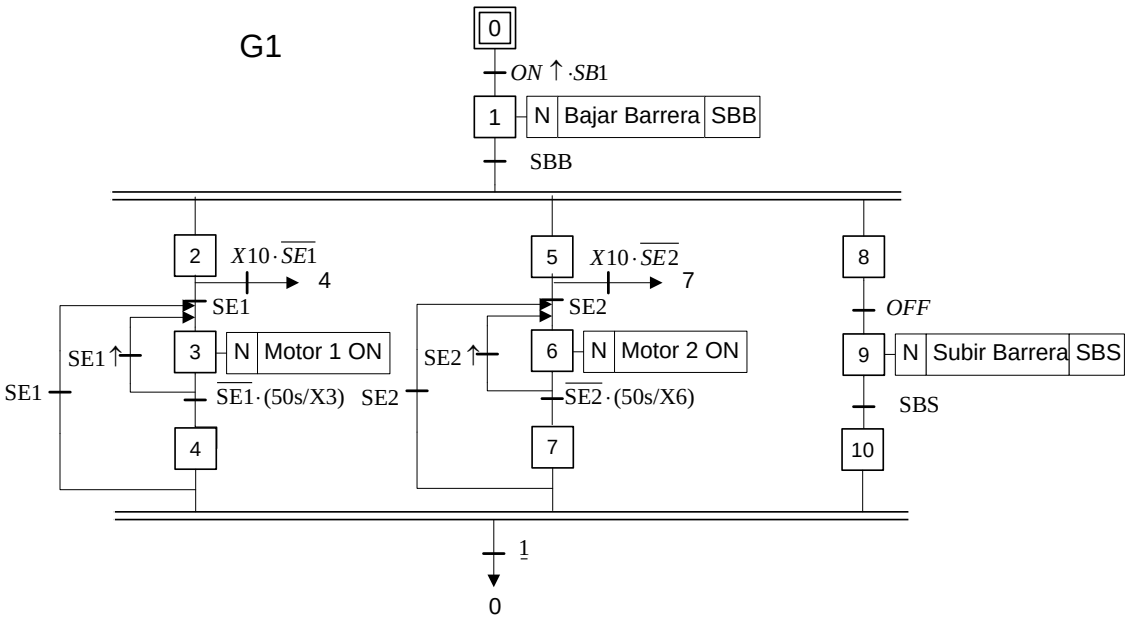

Figura 3. Posible modelado grafcet (sin emergencia) del sistema

3.5. BLOQUE TRAMO DE ESCALERA: PLANTILLA "START-STOP"

Se observa a simple vista en el grafcet propuesto que los motores 1 y 2 son idénticos desde la perspectiva del control y surge de manera natural la idea de hacer una función FB que encapsule el comportamiento de ambos para después crear dos motores diferentes empleando diferentes bloques de datos de instancia (DB) en la llamada a FB.

Para ello se propone usar una plantilla START-STOP dedicada con el siguiente comportamiento:

- **Entrada *start*** (tipo IN, BOOL): En nivel alto (acción START) el motor se sitúa en posición de espera activa (*standby*), y se pone en marcha al detectarse una persona en el correspondiente sensor óptico. En nivel bajo (acción STOP) el

motor se para y queda inhabilitado para ponerse en funcionamiento, lleguen personas o no a la embocadura del tramo.

- **Salidas** (tipo OUT): El actuador del motor y todos aquellos eventos que intervengan en su paso a parada; en el ejemplo resulta especialmente relevante la situación de espera activa por lo que habría que definir un parámetro de salida *standby* (tipo OUT, BOOL).

Si este bloque se implementa correctamente, bastará con gestionar adecuadamente la variable conectada a *start* en la llamada para conseguir un programa limpio y estructurado. Además añadir nuevos tramos de escalera al control será extremadamente sencillo.

Una posible interfaz completa para el bloque FB se muestra en la tabla 2.

Tabla 2. Interfaz completo de un bloque motor

```
VAR_INPUT
  se1 : BOOL ;
  tiempo : S5TIME ;          //tiempo de espera (50s) para pasar a standby
  temporizador : TIMER ;     //temporizador a emplear
  start : BOOL ;             //FALSE:motor en reposo. TRUE: motor en standby
END_VAR
VAR_OUTPUT
  motor_ON : BOOL ;
  standby : BOOL ;           //TRUE si en standby)
END_VAR
VAR_STATIC
  x0 : BOOL ;                //estado de standby (esperando, pero OFF)
  x1 : BOOL ;                //estado de marcha (motor ON)
  x2 : BOOL ;                //estado de standby tras marcha (tras 50s)
  xparada : BOOL ;           //modo START
  xmarcha : BOOL ;           //modo STOP
  bitDeTrabajo : BOOL ;
END_VAR
```

FB: "Maniobra con bit de Error explícito"

Intuitivamente la implementación tiene un grafcet maestro con estados START y STOP que fuerza un grafcet de producción; este último alterna el estado con el motor en funcionamiento al estado de *standby*. El bit de trabajo desactivado permite el arranque del estado X0 cuando START está activo, por lo que debe desactivarse al llegar a STOP. Observe que todas las memorias internas se han declarado como STATIC para que no tengan que pasarse en la llamada al bloque en la invocación.

TAREA 2. Implemente el bloque FB de la tabla 2 y compruebe que su comportamiento es el correcto en una sesión PLCSIM. Al inicio de la práctica deberá mostrar la sesión correctamente configurada al profesor

3.6. NORMATIVA GENERAL DE PROGRAMACIÓN

Tanto en esta como en el resto de prácticas de programación será imprescindible cumplir con el conjunto de requisitos generales siguientes, con independencia de las instrucciones concretas que se detallen en cada caso. Éstos son:

- Proyecto de programación STEP 7 debidamente configurado
- Tabla de símbolos adecuada
- Sesión de simulación PLCSIM configurada adecuadamente para mostrar las variables fundamentales del control y los símbolos. Dicha sesión deberá incluir un archivo de organización (extensión *.lay) dentro del proyecto STEP 7 que guarde la configuración predefinida por el alumno. De esta manera se asegura la portabilidad del proyecto para su simulación en otro equipo.
- Caso de requerirse la entrega del proyecto STEP 7 completo se generará una versión comprimida (*.zip) desde la ventana principal (**VP**) del Administrador SIMATIC mediante los comandos **VP**:*Archivo*→*Archivar* y la selección del proyecto del alumno. Este archivo comprimido, y solo éste, es el que se entregará al profesor.
- Abundantes comentarios: línea, segmento y bloque
- Estructuración adecuada de los segmentos en cada bloque

3.7. TAREAS A REALIZAR PARA EL INICIO DE LA PRÁCTICA

Al inicio de la práctica el alumno (además de cumplir con los requisitos previos recogidos en la sección 2.1) debe traer una solución software SIMATIC S7 con dos proyectos, uno para cada tarea descrita anteriormente (TAREA 1, sección 3.3; TAREA 2, sección 3.5).

TAREAS 1 y 2

Al comienzo de la práctica el alumno deberá simular correctamente las funciones FB descritas en las secciones 3.3 y 3.5 desde PLCSIM.

Nota: *El layout de la sesión de simulación (archivo *.lay) debe estar guardado previamente en los proyectos para evitar la pérdida de tiempo que supone realizar la configuración manualmente en el aula.*

3.8. ARQUITECTURA DE BLOQUES DEL CONTROL

Para superar correctamente la práctica se pide la siguiente arquitectura de bloques:

- **Bloque FB (sección 3.3) para las maniobras de subida y bajada de barrera**: Habrá dos invocaciones desde el bloque principal, una para cada maniobra, cada una con un bloque de datos de instancia distinto (p.ej. DB1 y DB4).
- **Bloque FB (sección 3.5) para el control de los motores**: Habrá dos invocaciones, una para cada motor (bloques de instancia DB2 y DB3)
- **Bloque FC para el grafcet de emergencia global**
- **Bloques OB1 y OB100**

A modo de ejemplo, la ventana principal del proyecto solución SIMATIC podría ser el de la figura 4.

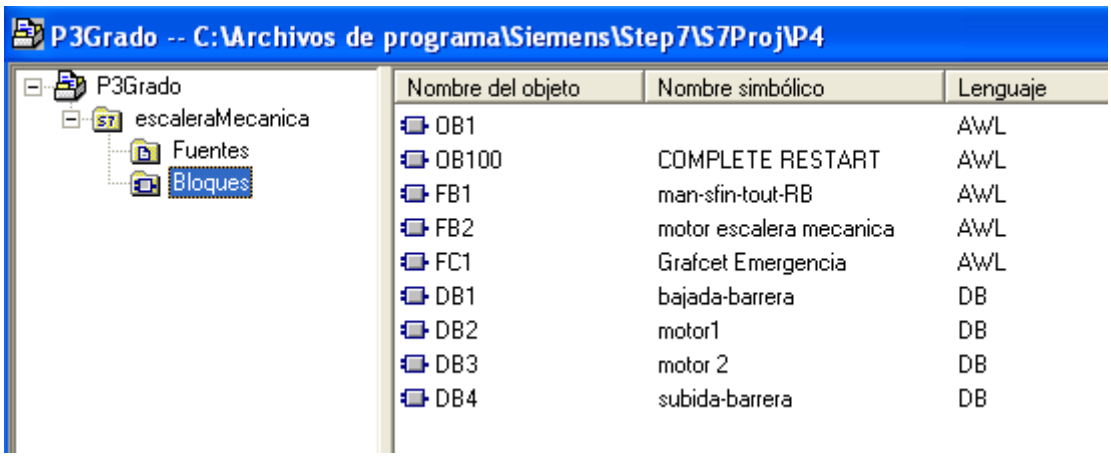

Figura 4. Ventana principal posible del proyecto solución

3.9. ENTREGA AL PROFESOR

Se pide:

Programa de control STEP 7 sujeto a la normativa general de la sección 3.6 para el control de los tramos de escalera mecánica descrito.

El código del programa de control se entregará al profesor en papel al comienzo de la práctica 4.