Escuela Politécnica Superior de Jaén

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN

Trabajo Fin de Grado

ELABORACIÓN DE MATERIAL DOCENTE PARA EL CONTROL DE PROCESOS DISCRETOS

Alumno: Juan Francisco Martínez León

Tutores: Elisabet Estévez Estévez y

Silvia María Satorres Martínez

Dpto: Ingeniería Electrónica y Automática

Universidad de Jaén

Escuela Politécnica Superior de Jaén

Departamento de Ingeniería Electrónica y Automática

Doña Elisabet Estévez Estévez y Silvia María Satorres Martínez, tutoras del Proyecto Fin de Carrera titulado: Elaboración de material docente para el control de procesos discretos, que presenta Juan Francisco Martínez León, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2017

El alumno:

Juan Francisco Martínez León

Los tutores:

Silvia María Satorres Martínez

Elisabeth Estévez Estévez

Índice

Indice de figuras

Indice de Tablas

1. INTRODUCCIÓN

En esta memoria se describe como se ha elaborado material docente para incorporar al alumno en el mundo de la programación de autómatas programables utilizando el lenguaje de programación de alto nivel SCL (Structured Control Language), con una serie de ejercicios prácticos que van aumentando en dificultad y contenido, utilizando la herramienta TIA PORTAL.

El lenguaje de programación SCL es un lenguaje de control estructurado basado en texto, esto supone una diferencia considerable se compara con los lenguajes que se estaban usando en hasta ahora en la universidad de Jaén.
Se corresponde con la norma IEC 61131-3 (ST). IEC 61131-3 es la tercera parte, de 8 en total del estándar internacional IEC 61131 para controladores lógicos programables o PLC.

Finalmente puntuar que gracias al lenguaje estructurado de SCL, no solo se pueden guardar u almacenar datos en diferentes zonas de memoria, sino que con sus funcionalidades de lectura cíclica se puede realizar una seria de operaciones con estos datos tantas veces como se desee con un simple código de programación.

1.1. Motivación

La motivación principal para el desarrollo de este proyecto es que hasta ahora se estaban utilizando en mayor medida lenguajes de programación para PLC siemens como AWL, KOP o FUP.

Se puede describir estos lenguajes de programación brevemente de la siguiente manera:

El lenguaje de programación FUP es un lenguaje de Siemens gráfico que utiliza los cuadros del álgebra booleana para representar la lógica. Además, éste permite representar funciones más complejas, como funciones matemáticas mediante cuadros lógicos.

La ventaja que puede proponer FUP es que se pueden ver agrupados por bloques las diferentes lógicas y además cuando hay mucha lógica de tipo “bool” puede facilitar la programación ya que puedes visualizar el segmento completo más fácilmente.

El lenguaje de programación KOP (Funktionsbausteinsprache FUP Funktionsplan) es posiblemente el más extendido de todos los lenguajes de programación Siemens, ya que posiblemente es el más fácil de entender por el usuario, ya que es un lenguaje de programación muy visual que se proviene de la industria eléctrica, en definitiva, es la representación del cableado si se deseara hacer el mismo programa que se desea realizar con el PLC.

El lenguaje de programación AWL es un lenguaje de programación textual orientado a la máquina.

En un programa creado en AWL (Anweisungsliste englisch STL), las instrucciones son similares a los pasos con los que la CPU ejecuta el programa.

El lenguaje AWL es de los tres lenguajes descritos el más complejo de utilizar por el usuario.

El lenguaje de programación SCL (Structured Control Language) es un lenguaje de programación de alto nivel basado en texto, parecido a otros lenguajes de programación muy extendidos en otras áreas de la ingeniería, como Pascal o C.

Por lo tanto, SCL presenta un gran potencial en comparación de otros lenguajes como KOP o FUP que pueden ser más visuales para aplicaciones puramente booleanas, pero que quedan muy atrás respecto a las funcionalidades del lenguaje SCL cuando se debe operar con funciones lógicas o aplicar instrucciones de más alto nivel con estructuras como IF, WHILE, ELSE, THEN etc.

1.2. Objetivos

El objetivo primordial de este proyecto es el desarrollo de una serie de ejercicios prácticos que se usarán como medio educativo para que el alumno vaya adquiriendo paso a paso una serie de conocimientos en la programación de autómatas utilizando el lenguaje de programación SCL.

Para ello se ha desarrollado una combinación de ejercicios prácticos con los que el alumno deberá trabajar en casa previamente e implementar después en un laboratorio con las herramientas necesarias para el buen desarrollo de las mismas.

Para el desarrollo de estas prácticas se en la Universidad de Jaén se cuenta con una serie de PLC Siemens 1214 DC/DC/DC, por lo tanto se ha optado por utilizar un lenguaje de programación Siemens y dentro de la variedad de lenguajes, se ha optado por el lenguaje de programación SCL por os motivos explicados en el apartado anterior.

Se han desarrollado un total de 6 ejercicios prácticos en los que se comienza con la utilización de un mínimo de comandos y herramientas que van aumentando en cantidad y complejidad, planteando problemas que se pueden encontrar en el ámbito laboral y con los que el alumno podrá tratar sin problemas una vez terminada su formación aplicando los conocimientos aprendidos durante las prácticas.

1.3. Metodología

La metodología utilizada en este proyecto ha sido la aplicación de conocimiento en el entorno de programación de autómatas programables para el control de procesos discretos.

Para el desarrollo del código utilizado para la programación ha sido necesario el aprendizaje de un lenguaje de programación de alto nivel como SCL utilizando conocimientos adquiridos previamente de otros lenguajes de programación en este entorno como son KOP y AWL.

Finalmente, ha sido necesario el uso de diferentes entornos de programación como TIA Portal, para la programación de los diferentes ejercicios prácticos desarrollados para este proyecto, así como el programa PLCSIM, que consiste en una máquina virtual para la comprobación del funcionamiento y de las respuestas que obtendríamos de un autómata real, así como otros programas como Microsoft Visio, que he utilizado para la creación de diferentes diagramas de flujo para la elaboración de las solución de cada uno de los ejercicios propuestos.

1.4. Organización de la memoria

Este proyecto está dividido en 4 capítulos, de la siguiente manera:

- Capítulo 1. Introducción. En este capítulo se realiza una introducción a la temática de este proyecto, donde se habla de la motivación que ha llevado a su realización, el objetivo final por que se ha llevado a cabo, así como la metodología que se ha aplicado a la hora del desarrollo del proyecto.
- Capítulo 2. TIA portal. Aquí realiza una breve introducción al programa utilizado para realizar la programación de los ejercicios prácticos en SCL, explicando brevemente la estructura del programa y como iniciarlo y crear proyectos en él.
- Capítulo 3. Introducción al lenguaje de programación SCL.
- Capítulo 4. Uso de temporizadores y contadores.
- Capítulo 5. Tipos de variables y divergencias selectoras
- Capítulo 6. Uso de simultaneidades.
- Capítulo 7. Uso de bloques de función (FC).
- Capítulo 8. Uso de bloques de función con memoria (FB).
- Capítulo 9. Conclusión. Se describe la conclusión del proyecto.
- Capítulo 10. Bibliografía. Contiene las referencias y bibliografías utilizadas durante el desarrollo del proyecto.

2. TIA PORTAL

TIA portal es un software desarrollado por Siemens utilizado para optimizar los procedimientos de procesamiento, operación de máquinas y planificación.

Si se desea, se puede consultar cualquier manual e información en la página oficial de Siemens para TIA portal:

https://www.siemens.com/global/en/home/products/automation/industry-software/automation-software/tia-portal.html

Entre sus propiedades cabe destacar los siguientes:

- Consta de una interfaz intuitiva para el usuario
- Diferentes editores de programación como son SCL, KOP, FUP, AWL y GRAPH.
- Simulación con herramienta PLCSIM.
- Tecnología flexible: funcionalidad de movimiento escalable y efectiva para controladores S7-1500 y S7-1200.

La descarga del programa TIA portal en sus diferentes versiones, descarga de documentación, petición de una licencia temporal gratuita, así como la descarga de diferentes librerías, firmwares y actualizaciones pueden consultarse en el siguiente sitio web: *http://w5.siemens.com/spain/web/es/industry/automatizacion/simatic/tia-portal/tia_portal/pages/tia-portal.aspx*.

2.1. Versión de programa y CPU

Importante tener en cuenta que para el desarrollo de este programa se ha utilizado la versión del programa TIA portal V13.

Para las simulaciones se ha utilizado el programa PLCsim, con la versión V13.

La CPU que vamos a utilizar tanto virtualmente como físicamente para el desarrollo de los ejercicios prácticos que vamos a explicar en este proyecto es una CPU 1214 DC/DC/DC (6ES7214-1AG40-0XB0) que contiene 14 entradas digitales de 24V DC, 10 salidas digitales 24V DC 0.5 y 2 entradas analógicas 0-10V.

2.2. Creación de un nuevo proyecto

Una vez Descargado e instalado el programa TIA portal, si se procede a abrir dicho programa, aparecerá una interfaz inicial como en la siguiente figura:

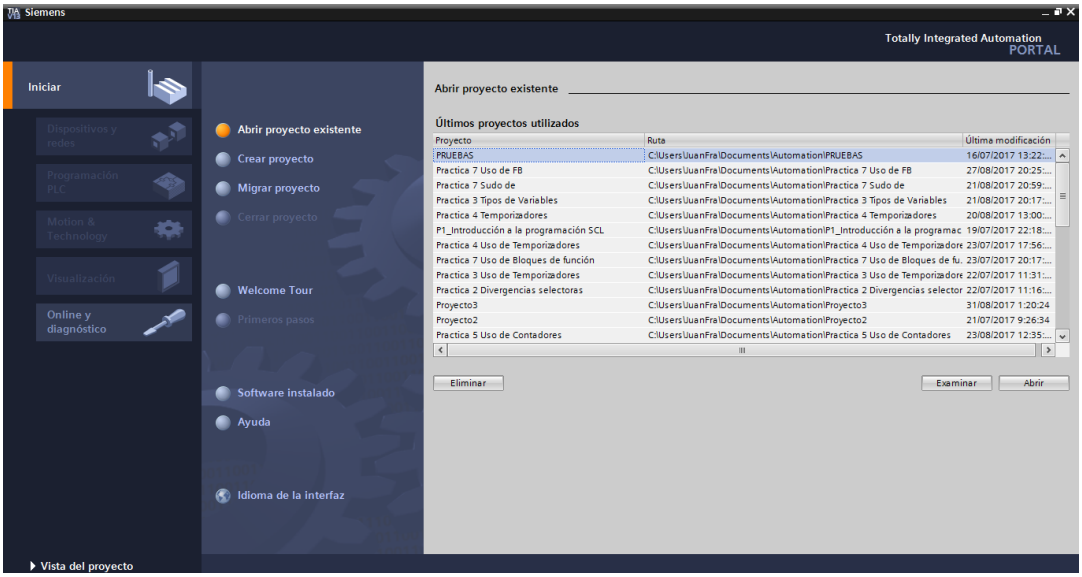

Figura 1: interfaz inicial TIA portal

Como puede verse en la imagen anterior, el usuario puede optar por varias opciones, abrir un proyecto existente, crear un nuevo proyecto o migrar un proyecto.

En este caso, para crear un proyecto desde cero, se clicará en la opción de crear un proyecto nuevo.

Una vez hecho esto, se abrirá una pantalla como la adjuntada abajo, en la que se dan varias opciones, como “configurar un dispositivo”, para configurar con que CPU y periféricos vamos a trabajar, “escribir programa PLC”, para comenzar la programación del proyecto, configurar objetos tecnológicos, “configurar una imagen HDMI” y abrir la vista del proyecto.”

Si se continúa usando la opción de “abrir vista de proyecto” se pueden configurar desde la interfaz principal todos los aspectos del proyecto.

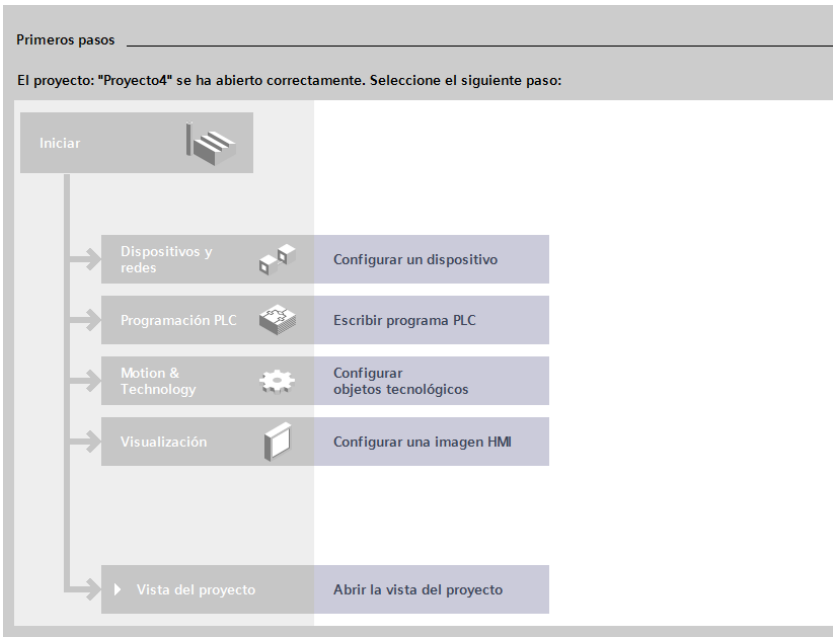

Figura 2: Primeros pasos TIA portal

Una vez abierta la vista del proyecto en la parte izquierda de la pantalla se encuentra la vista del árbol de proyecto donde se pueden seleccionar todos los aspectos del proyecto, en este caso al ser un proyecto creado desde cero, lo primero que deberá hacerse será configurar la CPU con la que se va a trabajar.

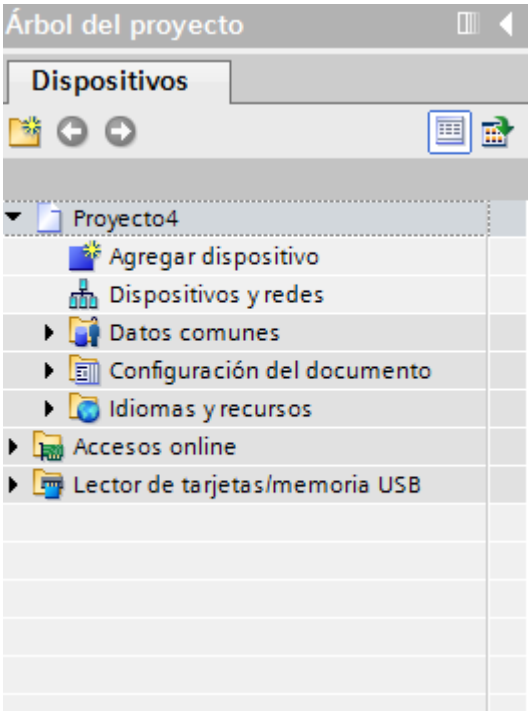

Figura 3: Árbol de proyecto inicial TIA portal

Para ello se utilizará la opción de agregar dispositivo y se seleccionara la CPU 1214C – 6ES7214-1AG40-0XB0 tal y como se muestra en la siguiente imagen.

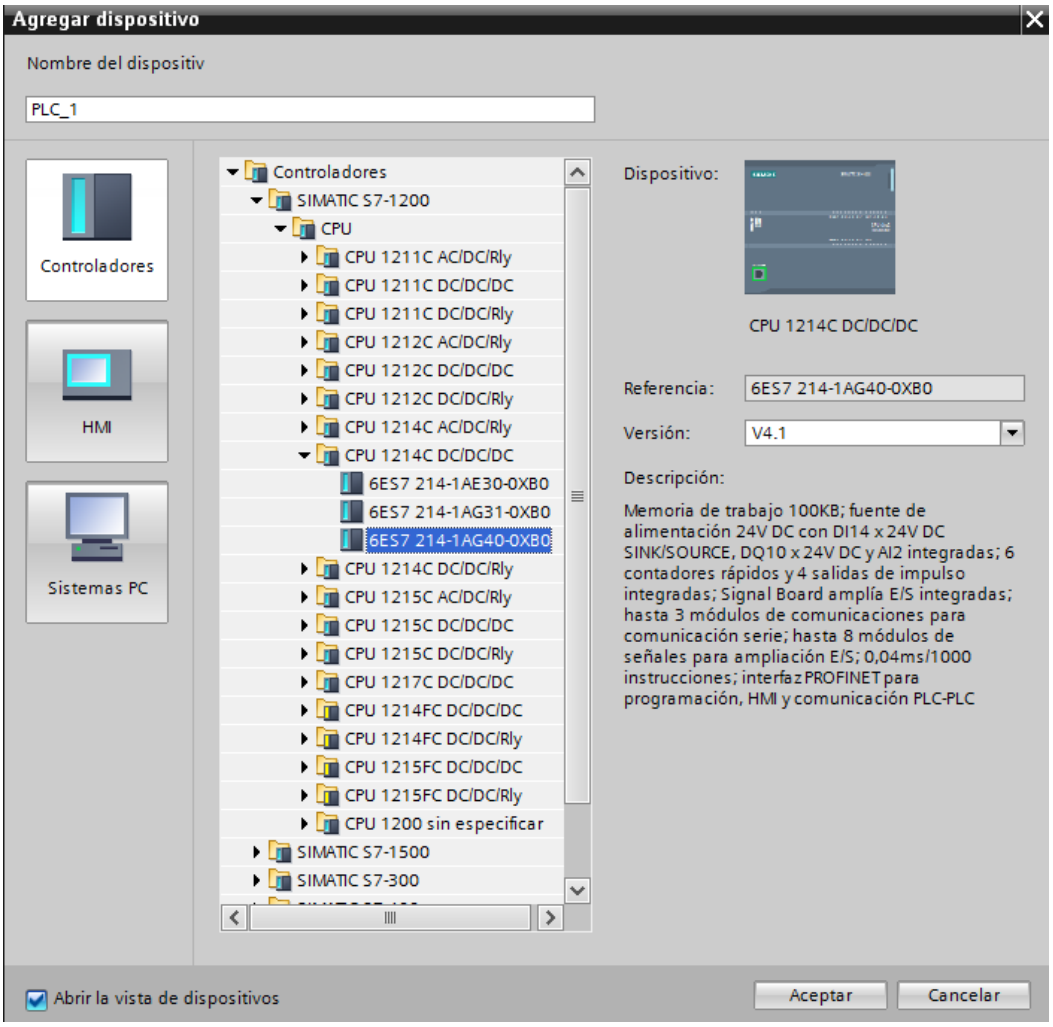

Figura 4: Agregar CPU TIA portal

Se selecciona esta opción para la CPU ya que es la que se va a utilizar durante las prácticas en la Universidad de Jaén.

Se debe utilizar la versión 4.1 para poder realizar las simulaciones pertinentes con el programa PLCsim.

2.3. Interfaz de programa

Una vez configurada la CPU con la que se va a trabajar, aparecerán en el árbol de proyecto todas las opciones para poder comenzar con la programación de nuestro proyecto.

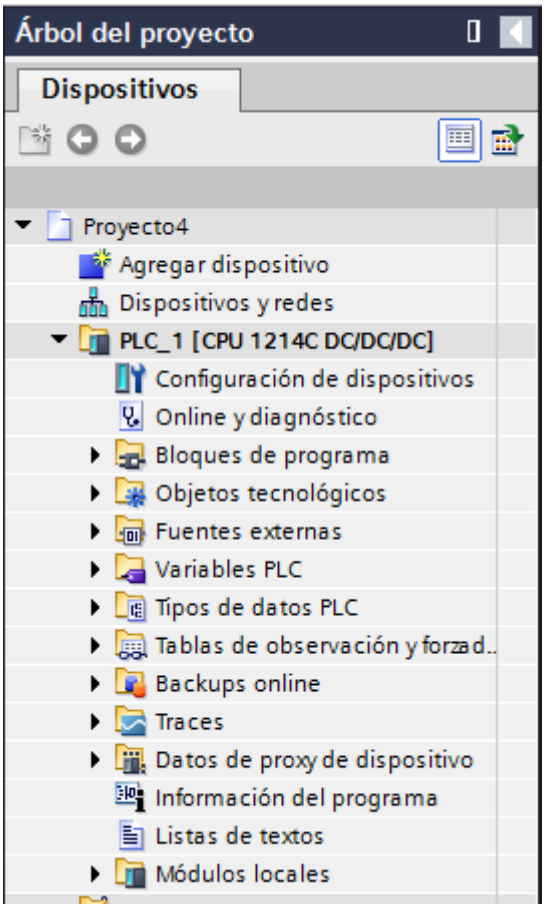

Figura 5: Árbol del proyecto TIA portal

Entre todas las opciones, cabe destacar para la elaboración de las prácticas incluidas en este proyecto la opción “Bloques de programa” y “Variables PLC”.

Si se despliega la opción de “Variables PLC”, se pueden encontrar varias opciones, como mostrar las variables de nuestro proyecto o generar una nueva tabla de variables.

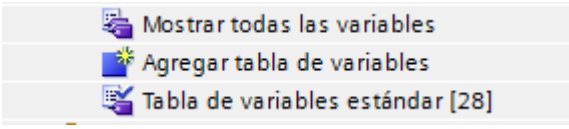

Figura 6: Variables PLC TIA portal

Dentro de la opción de “Bloques de sistema” se tiene la opción de modificar o visualizar todos los bloques que se estén desarrollando en el proyecto.

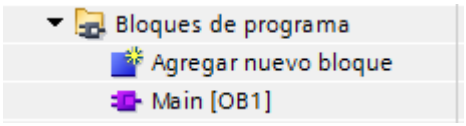

Figura 7: Bloques de programa TIA portal

Si se desea generar un nuevo bloque, se utilizará la opción de “Agregar nuevo bloque” y se desplegara una ventana como la mostrada en la siguiente figura.

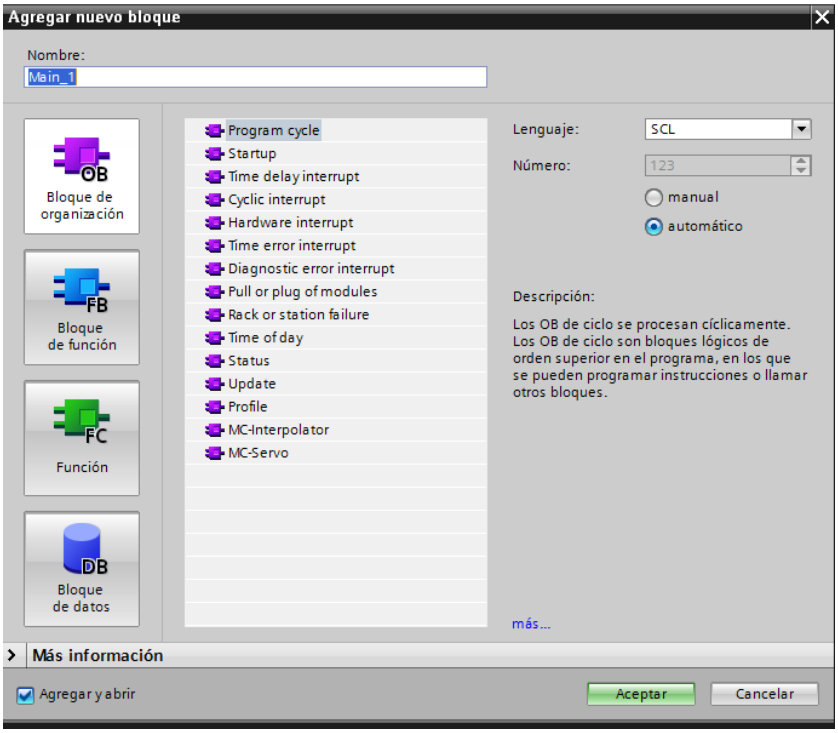

Figura 8: Agregar nuevo bloque TIA portal

Donde se podrá elegir crear cualquier tipo de bloque o función para el desarrollo del proyecto.

3. PRÁCTICA 1: INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN SCL

3.1. Introducción

La finalidad de esta primera práctica es que el alumno comience a sumergirse en el entorno de programación de TIA PORTAL con un lenguaje de programación de alto nivel como es SCL, que como ya se comenta anteriormente, es un lenguaje de alto nivel que tiene muchas ventajas a la hora de enfrentarse a problemas de mayor complejidad de los que podría abarcarse con otros lenguajes de programación más gráficos como KOP.

Para esta primera práctica se deberán utilizar comandos básicos en programación SCL como ejecuciones condicionales y asignación de variables.

Es importante que el alumno conozca una serie de conceptos básicos en programación SCL antes de realizar la práctica:

El primer paso antes de empezar a programar en el bloque principal o en algún otro es que se deben de inicializar las variables con las que se va a trabajar. Para ello se debe ir a *"Mostrar todas las variables*" en la interfaz de TIA portal. Consúltese el apartado "*2.3 Interfaz del programa"* de este proyecto para más información.

En la siguiente figura se puede ver la ventana de variables del proyecto con el que se está trabajando, en las que para cada variable se debe definir una serie de conceptos:

Variables PLC

	Nombre	Tabla de variables	Tipo de datos	Dirección	Rema...	Visibl...	Acces...	Comentario
1	PD	Tabla de variables e..	Bool	%I0.0	☐	☑	☑	Sensor de posición Derecho
2	PC	Tabla de variables e..	Bool	%I0.1	☐	☑	☑	Sensor de posición Central
3	PI	Tabla de variables e..	Bool	%I0.2	☐	☑	☑	Sensor de posición Izquierdo
4	Up	Tabla de variables e..	Bool	%I0.3	☐	☑	☑	Sensor posición Superior
5	Down	Tabla de variables e..	Bool	%I0.4	☐	☑	☑	Sensor posición inferior
6	Spieza	Tabla de variables e..	Bool	%I0.5	☐	☑	☑	Sensor para detectar si hay pieza para reco
7	MGD	Tabla de variables e..	Bool	%Q0.0	☐	☑	☑	Motor para mover la grúa a la derecha
8	MGI	Tabla de variables e..	Bool	%Q0.1	☐	☑	☑	Motor para mover la grúa a la izquierda
9	MGS	Tabla de variables e..	Bool	%Q0.2	☐	☑	☑	Motor para mover la grúa hacia arriba
10	MGB	Tabla de variables e..	Bool	%Q0.3	☐	☑	☑	Motor para mover la grúa hacia abajo
11	CERRP	Tabla de variables e..	Bool	%Q0.4	☐	☑	☑	Cerrar la pinza de la grúa para agarrar obje
12	X0	Tabla de variables e..	Bool	%M0.0	☐	☑	☑	

Figura 9: Variables en TIA portal

En la columna "*Nombre*" se debe definir el nombre con el que se va a llamar a nuestra variable.

En la columna "*Tipo de datos*" se debe seleccionar el tipo de variable con el que se va a trabajar. Para este ejercicio en particular solo se va a trabajar con variables digitales, por lo tanto, todas las variables serán de tipo "*Bool*", aunque más adelante se utilizará otro tipo de variables que se explicarán en más detalle.

En el apartado "*Dirección*", se está definida la dirección de la memoria que está ocupando cada variable en la CPU.

Por último, se puede utilizar la columna "*Comentario*" para realizar una breve explicación de en que consiste cada variable, aunque esto no tendrá ningún tipo de importancia en la programación.

Una vez definidas las variables se procederá a la programación del proyecto, para ello se debe ir a la opción de "*bloques de programa*". Consúltese el apartado "*2.3 Interfaz del programa"* de este proyecto para más información.

Se deben diferenciar entre dos tipos de bloques de programa para este primer ejercicio, OB de ciclo y OB de arranque.

Los OB de arranque se procesan una sola vez cuando el modo de operación de la CPU cambia de STOP a RUN. Tras el procesamiento del OB de arranque se inicia el procesamiento del OB de ciclo.

Los OB de ciclo se procesan cíclicamente. Los OB de ciclo son bloques lógicos de orden superior en el programa, en los que se pueden programar instrucciones o llamar otros bloques.

En lo que a implementación de código se refiere es importante que para comenzar con esta introducción al lenguaje SCL se conozca una serie de instrucciones:

- Asignación de una variable:

Para asignar un valor a una variable se debe escribir el nombre de la variable entre comillas, seguido de ":=" y el valor al que se desea que esté la variable, por ejemplo:

$$"Variable" := 0; \quad [1]$$

- Establecer una ejecución condicional:

Para realizar una condición se utiliza la instrucción "IF".

Esta instrucción se realiza de la siguiente manera, se propone una condición y cuando esta se cumpla se producirá una acción. Utilizando el comando "*ELSE*", se puede elegir además que pasará cuando no se produzca esta condición.

Como condiciones se pueden asignar sumas o multiplicaciones lógicas de diferentes variables con los comandos "OR" y "AND" respectivamente.

Para explicar cómo se utiliza esta instrucción se va a proponer un ejemplo, en que a la una variable que vamos a llamar "*var*" le vamos a asignar el valor "1" cuando la etapa llamada "*X1*" y la entrada "*P1*" estén activas y un valor de "0", cuando no se produzca esta condición.

```
IF "X1" AND "P1" THEN
    "var" := 1;
ELSE
    "var" := 0;
END_IF;
```

Figura 10: Instrucción IF

3.2. Pinza recolectora

Esta práctica tiene como finalidad que el alumno comience a introducirse en el lenguaje de programación SCL comenzando por el manejo de unos comandos básicos, por lo tanto, se ha planteado un problema en el que se deben utilizar varias señales de sensores para realizar el movimiento de una pinza recolectora.

Una determinada empresa, desea mejorar la calidad de algunos de los procesos en su fábrica, para ello ha decidido empezar a automatizar alguna de las tareas que realizaban antes manualmente algunos de sus trabajadores.

Al llegar cierto punto de su cadena de montaje se desea utilizar una grúa para transportar una pieza de un punto a otro, utilizando para ello una pinza capaz de agarrar las piezas a transportar y varios sensores para conocer la posición de la grúa y si hay piezas disponibles para recoger.

En la siguiente figura se puede ver la disposición de la grúa:

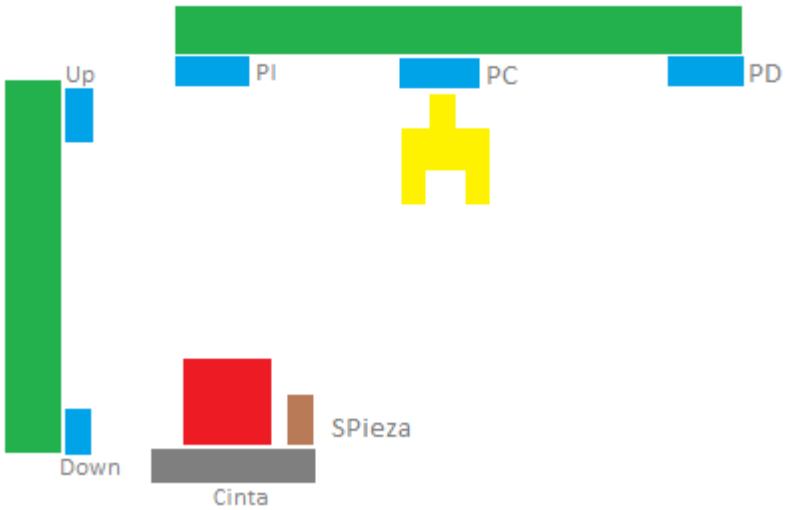

Figura 11: Pinza práctica 1

Para determinar la posición de la grúa se usarán 4 sensores de posición:

- **PD**: Sensor de posición en la parte derecha.
- **PI**: Sensor de posición en la parte izquierda.
- **PC**: Sensor de posición en la parte central.
- **Up**: Sensor de posición en la parte superior.
- **Down**: Sensor de posición en la parte inferior.

El movimiento de la Grúa se realiza mediante dos motores diferentes conectados a una serie de poleas. Uno de ellos maneja las funciones de subida y bajada de la grúa y el otro se encarga del desplazamiento hacia el lado derecho o izquierdo.

Para realizar los diferentes movimientos utilizamos las siguientes variables de salidas:

- **MGD**: Realiza el movimiento de la grúa hacia el lado derecho.
- **MGI**: Realiza el movimiento de la grúa hacia el lado izquierdo.
- **MGS**: Realiza el movimiento de la grúa a la parte superior.
- **MGB**: Realiza el movimiento de la grúa a la parte inferior.

El proceso a realizar es el siguiente:

La grúa permanecerá estática hasta que el operario utilice el pulsador de puesta en marcha (**PM**) y para evitar cualquier contratiempo y velar por la seguridad de los trabajadores estará disponible un pulsador de parada (**PP**). Si en cualquier momento este pulsador es utilizado, el proceso se parará, aunque no haya terminado de realizar el transporte de la pieza.

Una vez realizada la puesta en marcha, se deberá posicionar la grúa en la parte superior derecha en el caso de que no lo estuviera, teniendo en cuenta que nunca se deberá realizar el movimiento de izquierda a derecha o viceversa, mientras la grúa esté posicionada en la parte inferior del mecanismo.

Antes de empezar el proceso de recogida de la pieza se deberá asegurar que efectivamente se tiene una pieza que recoger, por lo tanto, se deberá esperar a que se detecte una pieza con el sensor **Spieza**. La pieza llegará a través de una cinta transportadora que se activa con la señal **Cinta**.

Una vez realizada la inicialización de la posición de la grúa y asegurar que hay una pieza para recoger, esta deberá recoger una pieza en la parte inferior izquierda del mecanismo y depositarla en la parte inferior central, recordando que en ningún caso se deberá realizar el movimiento de derecha a izquierda cuando la grúa esté situada en la parte inferior del proceso por motivos de seguridad.

El agarre de la pieza se realizará mediante la inyección de aire a presión, que cerrará las pinzas de la grúa a través de la orden **CERRP**.

Para evitar la caída de las piezas hasta la llegada al punto de recogida se ha instalado un mecanismo de roscado en la base de la pieza, por lo tanto, se deberá proceder al desenroscado de dicha pieza, haciendo girar la grúa sobre su eje. El giro de la grúa hacia izquierda o derecha se realiza con las funciones **RotateLeft** y **RotateRight**.

El eje de la grúa solo puede someterse a un número determinado de vueltas hacia un mismo lado, por lo cual se han utilizado dos finales de carrera (**FinRotateRight** y **FinRotateLeft**) que se activan para dejar de girar la grúa y causar daños al mecanismo.

Se podrá determinar cuando el final del desenroscado de la pieza con la activación de la entrada **DesRosca**.

El proceso de trasporte de piezas con la grúa se realizará indefinidamente hasta que el operario encargado, utilice el pulsador de parada.

3.2.1. Solución de la práctica 1

3.2.2. Diseño del GRAFCET

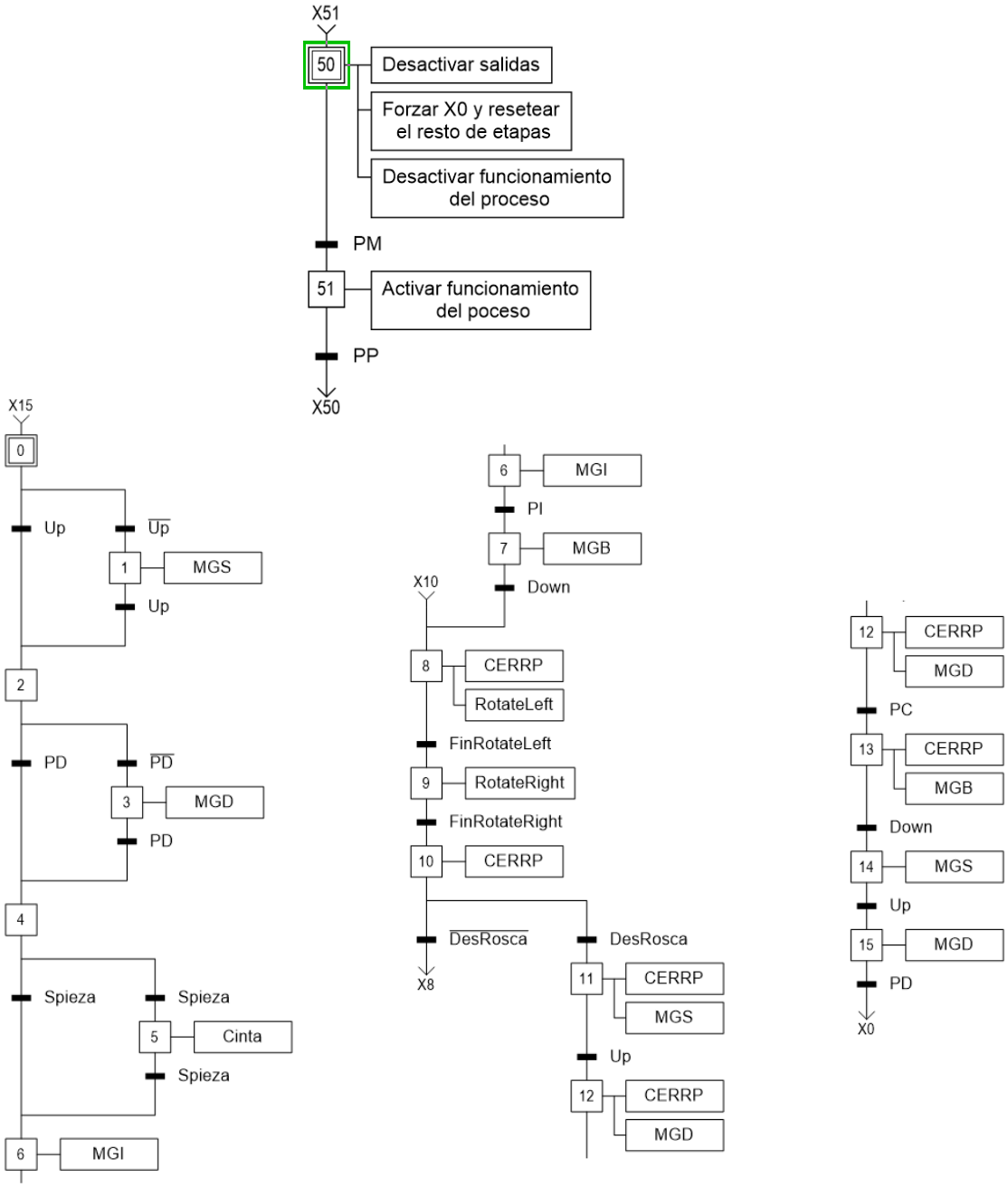

Figura 12: GRAFCET práctica 1

3.2.3. Tabla de evolución secuencial

ETAPA	SET	RESET
X0	I + X15*PP + X51*PM	X1 + X2
X1	X0*	X2 + X51
X2	X0*Up	X3 + X4 + X51
X3	X2*	X4 + X51
X4	X2*Pd	X5 + X6 + X51
X5	X4*	X6 + X51
X6	X4*Spieza	X7 + X51
X7	X6*PI	X8 + X51
X8	X7*Down + X10*	X9 + X51
X9	X8*FinRotateLeft	X10 + X51
X10	X9*FinRotateRight	X11 + X8 + X51
X11	X10*DesRosca	X12 + X51
X12	X11*Up	X13 + X51
X13	X12*PC	X14 + X51
X14	X13*Down	X15 + X51
X15	X14*Up	X0 + X51
X50	X51*PM	X51
X51	I + X50*PP	X50

Tabla 1: Evolución secuencial práctica 1

3.2.4. Tabla combinacional

SALIDA	Activación
MGD	X3 + X12 + X15
MGI	X6
MGS	X1 + X11 + X14
MGB	X7 + X13
CERRP	X8 + X10 + X11 + X12 + X13
RotateRight	X9
RotateLeft	X8
Cinta	X5

Tabla 2: Combinacional práctica 1

3.2.5. Tabla de variables

Esta tabla de variables ha sido obtenida de la tabla de variables realizada en TIA portal para la programación de esta práctica.

Nombre	Tipo	Dirección	Comentario
PD	Bool	%I0.0	Sensor de posición Derecho
PC	Bool	%I0.1	Sensor de posición Central
PI	Bool	%I0.2	Sensor de posición Izquierdo
Up	Bool	%I0.3	Sensor posición Superior
Down	Bool	%I0.4	Sensor posición inferior
Spieza	Bool	%I0.5	Sensor para detectar si hay pieza para recogida
MGD	Bool	%Q0.0	Motor para mover la grúa a la derecha
MGI	Bool	%Q0.1	Motor para mover la grúa a la izquierda
MGS	Bool	%Q0.2	Motor para mover la grúa hacia arriba
MGB	Bool	%Q0.3	Motor para mover la grúa hacia abajo
CERRP	Bool	%Q0.4	Cerrar la pinza de la grúa para agarrar objeto
X0	Bool	%M0.0	
X1	Bool	%M0.1	
X2	Bool	%M0.2	
X3	Bool	%M0.3	
X4	Bool	%M0.4	
X5	Bool	%M0.5	
X6	Bool	%M0.6	
X7	Bool	%M0.7	
X8	Bool	%M1.0	
X9	Bool	%M1.1	
X10	Bool	%M1.2	
X11	Bool	%M1.3	
X12	Bool	%M1.4	
X13	Bool	%M1.5	
X14	Bool	%M1.6	
X15	Bool	%M1.7	
X50	Bool	%M2.0	
X51	Bool	%M2.1	
PP	Bool	%I0.6	Pulsador de Paro
PM	Bool	%I0.7	Pulsador de Marcha
FinRotateRight	Bool	%I1.0	Fin de carrera del motor de rotación de la

			grúa hacia la derecha
FinrotateLeft	Bool	%I1.1	Fin de carrera del motor de rotación de la grúa hacia la izquierda
RotateRight	Bool	%Q0.5	Motor de rotación a la derecha de la grúa
RotateLeft	Bool	%Q0.6	Motor de rotación a la izquierda de la grúa
DesRosca	Bool	%I1.2	Sensor para comprobar el desenroscado de la pieza
Cinta	Bool	%Q0.7	Motor para traer piezas a través de una cinta mecánica

Tabla 3 Variables práctica 1

3.2.6. Codificación

Si se desea consultar la codificación de este ejercicio práctico, consulte el Anexo número 1.

4. PRÁCTICA 2: USO DE TEMPORIZADORES Y CONTADORES

4.1. Introducción

Para esta segunda práctica el alumno ya conoce algunos comandos básicos en programación SCL que se explicaron en la práctica 1, como asignaciones de variables o instrucciones condicionales.

En esta segunda práctica se busca afianzar los conocimientos adquiridos en la primera parte de esta serie de ejercicios añadiendo además el uso de algunas de las herramientas más utilizadas en programación de autómatas, los temporizadores y contadores.

Los temporizadores son tremendamente útiles cuando se desea realizar una acción durante un tiempo determinado.

Cuando se utiliza un temporizador, se está utilizando un DB de instancia en el que se guardan las variables estáticas del mismo.

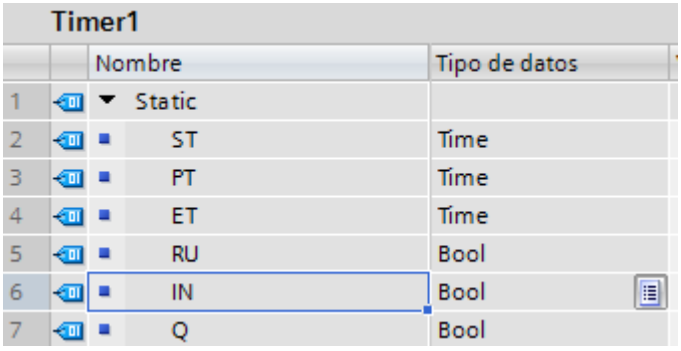

Figura 13: Variables estáticas del temporizador

Donde la variable “”*IN*” indica cuando debe de iniciarse donde debe iniciar el contaje el temporizador.

La variable “*PT*”, indica el tiempo que va a estar activo el contaje del temporizador.

Cuando se cumpla el tiempo de contaje de dicho temporizador se señalará mediante la activación de la variable “*Q*”.

"*ET*" es la variable temporal que indica en cada momento el valor del temporizador.

A continuación, se procede a explicar los diferentes tipos de temporizadores con lo que contamos en programación de autómatas Siemens y como realizar su llamada.

Para realizar la llamada de un temporizador se puede hacer rápidamente arrastrando el temporizador que se desee desde la parte de instrucciones básicas de TIA portal.

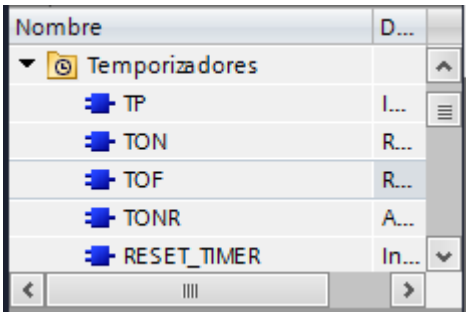

Figura 14: Temporizadores en instrucciones básicas

Se puede optar por elegir entre dos tipos diferentes de temporizadores, los temporizadores con retardo a la desconexión y temporizadores con retardo a la conexión.

Un contador con retado a la conexión, empieza a realizar el contaje cuando se activa la señal que tiene definida como entrada, mientras que el temporizador con retardo a la desconexión funciona de marta contraria, se inicia cuando deja de recibir una señal de su señal de activación.

El código para la llamada de un temporizador con retardo a la conexión se realiza como se muestra en la siguiente figura:

```
"Timer1".TON(IN:=_bool_in_,
             PT:=_time_in_,
             Q=>_bool_out_,
             ET=>_time_out_);
```

Figura 15: Programación de temporizador con retardo a la conexión

El código para la llamada de un temporizador con retardo a la desconexión se realiza de la siguiente manera:

```
"Timer2".TOF(IN:=_bool_in_,
             PT:=_time_in_,
             Q=>_bool_out_,
             ET=>_time_out_);
```

Figura 16: Programación de temporizador con retardo a la desconexión

La asignación de las variables asociadas al temporizador se realiza de la misma manera para ambos temporizadores, teniendo en cuenta que, el temporizador con retardo a la desconexión no se iniciará cuando se active la señal o las señales definidas en "*IN*", sino al desactivarse.

Para introducir el valor en una variable tipo "*time*" como es el caso de la variable "*PT"* se debe introducir de la siguiente manera:

$$"PT" := T\#As; \qquad [3]$$

Donde A debe de ser un valor numérico que indicará el tiempo que se desea introducir en a variable.

Una vez definido que es y cómo se utiliza un temporizador, llega el turno de hablar de los contadores.

Los contadores son herramientas tremendamente útiles en programación de procesos que deben repetirse durante un número definido de veces.

Los contadores son DBs de instancia propios de TIA portal al igual que los temporizadores, por eso se debe definir un bloque al que en este caso se le ha llamado Cunter1.

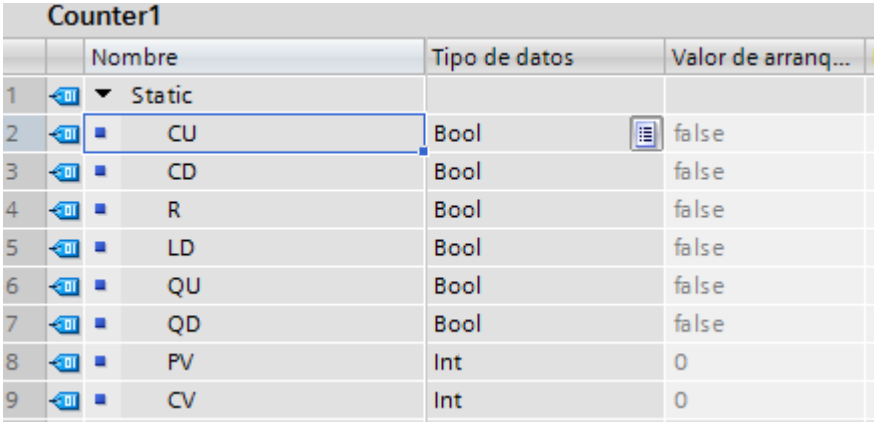

Figura 17: Variables estáticas de un contador

Para está practica en concreto se utilizará un contador para hacer parpadear una señal lumínica de un semáforo que tiene unos tiempos de encendido y apagados definidos con dos temporizadores y debe de repetirse un número determinado de veces.

Para realizar la llamada de un contador se puede hacer rápidamente arrastrando el contador que se desee desde la parte de instrucciones básicas de TIA portal.

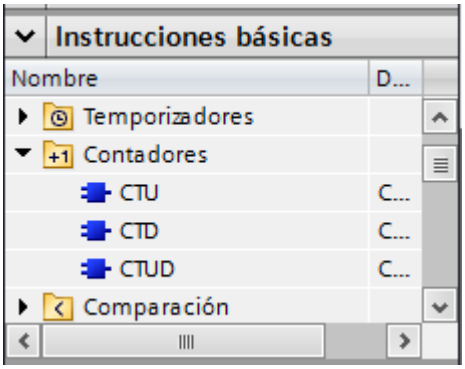

Figura 18: Contadores en instrucciones básicas

Se pueden diferenciar 3 diferentes tipos de contadores:

Contadores ascendentes que van amentando el valor de la variable "*CV*", hasta llegar al valor definido en "*PV*", cada vez que se produzca la activación de la variable "*CU*".

La variable "*R*" se utiliza para resetear el contador y "*Q*" se activa cuando se alcance el valor definido en la variable "*PV"*.

En la siguiente figura se puede ver cómo sería la programación de un contador:

```
"Cont".CTU(CU:=_bool_in_,
           R:=_bool_in_,
           PV:=_in_,
           Q=>_bool_out_,
           CV=>_out_);
```

Figura 19: Programación de contador ascendente.

Los contadores descendentes se diferencian de los ascendentes por que en lugar de partir de un valor de cero y ascender hasta un valor definido, empiezan en un valor definido y el mismo desciende hasta cero.

Se puede ver en la imagen adjunta debajo como sería la llamado a un contador descendente.

```
"Cont".CTD(CD:=_bool_in_,
           LD:=_bool_in_,
           PV:=_in_,
           Q=>_bool_out_,
           CV=>_out_);
```

Figura 20: Programación de contador descendente

Finalmente mencionar también los contadores ascendentes-descendentes, que pueden implementar en una sola llamada un contador que puede ascender o descender el valor de una variable, definiendo un valor límite por arriba y por abajo.

Se adjunta como se haría una llamada a un contador ascendente-descendente en la imagen de abajo.

```
"Cont".CTUD(CU:=_bool_in_,
            CD:=_bool_in_,
            R:=_bool_in_,
            LD:=_bool_in_,
            PV:=_in_,
            QU=>_bool_out_,
            QD=>_bool_out_,
            CV=>_out_);
```

Figura 21: Programador de contador ascendente-descendente

4.2. Funcionamiento de un semáforo

Para el correcto aprendizaje de uso y llamadas a temporizadores se ha realizado un ejercicio práctico en el que se programar el funcionamiento de un semáforo en una determinada calle.

El ayuntamiento de una determinada cuidad desea instalar una serie de semáforos y se desea que la iluminación de estos sea manejada a través de PLCs.

Por lo tanto, se debe de tener en cuenta que cada una de las diferentes señales lumínicas de un semáforo serán manejadas como salidas del PLC.

Recordar que cada semáforo dispone de 3 luces de señalización para vehículos (roja, verde y ámbar) y dos para peatones (roja y verde).

El funcionamiento debe se debe realizar respecto a las siguientes indicaciones:

Hay dos modos de funcionamiento dependiendo de si se está en horario diurno o nocturno.

Al PLC se le hará llegar una señal digital (**Dia**). a través de un ordenador, que usando su reloj podrá detectar en cada momento si se haya en horario diurno o nocturno.

Si se encuentra en horario diurno, la señalización de los semáforos debe funcionar con normalidad, esto significa que la señalización lumínica de los semáforos debe funcionar de la siguiente manera:

En un primer momento la circulación de vehículos deberá estar activa, por lo tanto, debe estar encendida la luz verde del semáforo para vehículos y roja para prohibir el paso de los peatones.

El cambio de color del semáforo para vehículos debe producirse cuando pase un minuto desde inicio del ciclo o en su defecto, cuando se utilice el pulsador para pedir el paso de los peatones (**PPeat**) una vez hayan pasado 30 segundos desde el inicio del ciclo.

Cuando ocurriera alguna de las situaciones anteriores y se permitiera el paso para los peatones y se prohibiera el mismo para los vehículos, se mantendría esta situación durante 25 segundos.

Para las transiciones entre señalizaciones de colores, tanto para la señalización de vehículos como para la peatonal, si se está en color rojo, cambiará directamente a color verde.

Para la transición entre verde y rojo el semáforo el semáforo peatonal parpadeará apagándose y encendiéndose cuatro veces en intervalos de dos segundos encendido y dos segundos apagado. Una vez el peatonal prohíba el paso, se procederá a permitir el paso a los vehículos, cambiando a color verde.

Mientras se realiza el cambio de verde a rojo, el semáforo para la señalización de vehículos estará en color ámbar durante 5 segundos y una vez el semáforo para vehículos cambie a color rojo, el peatonal lo hará al verde.

Si por el contrario se encuentra en horario nocturno, los semáforos correspondientes a los peatones permanecerán apagados y los pertenecientes a la circulación de vehículos permanecerán en color ámbar y serán alimentados con una señal PWM para que estén dos segundos encendidos y otros tdos segundos apagados, repitiendo el ciclo hasta que acabe el horario nocturno.

Las salidas utilizadas para manejar las señales lumínicas de un semáforo son las siguientes:

Variable	Descripción
VehiVerde	Enciende la señal lumínica verde del semáforo para permitir el paso de vehículos.
VehiAmbar	Enciende la señal lumínica ámbar del semáforo para señalizar el cambio de verde a rojo.
VehiRojo	Enciende la señal lumínica roja del semáforo para prohibir el paso de vehículos.
PeatVerde	Enciende la señal lumínica verde del semáforo para permitir el paso de peatones.
PeatRojo	Enciende la señal lumínica roja del semáforo para prohibir el paso de peatones.

Tabla 4: Descripción de variables práctica 2

4.3. Solución de la práctica 2

4.3.1. Diseño del GRAFCET

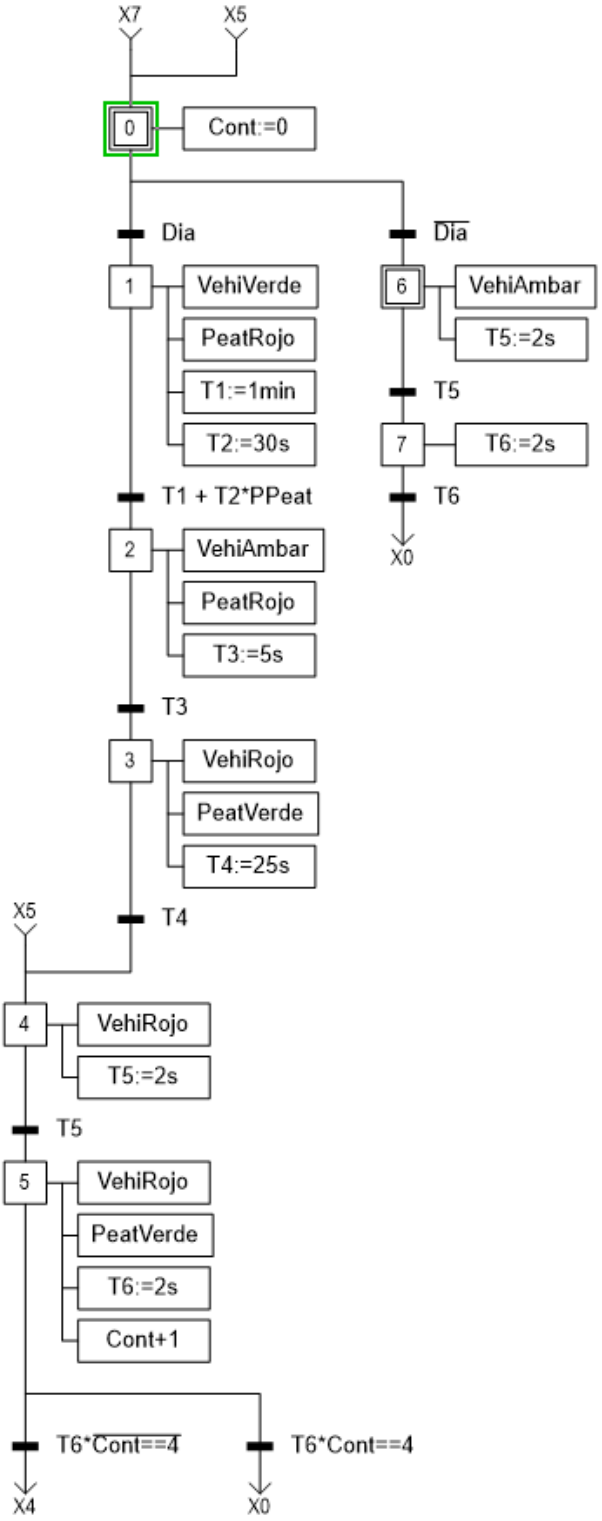

Figura 22: GRAFCET práctica 2

4.3.2. Tabla de evolución secuencial

ETAPA	SET	Activación
X0	I + X5*T6*(Cont==4) + X7*T6	X1 + X26
X1	X80*Dia	I + X2 + X8 + X14 + X20
X2	X1*T1 + X1*T2*PPeat	I + X3
X3	X2*T3	I + X4
X4	X3*T4 + X5* $\overline{(\mathrm{Cont} == 4)}$	I + X5
X5	X4*T5	I + X6
X6	X0*$\overline{\mathrm{Dia}}$	I + X7
X7	X6*T5	I + X28

Tabla 5: Evolución secuencial práctica 2

4.3.3. Tabla combinacional

SALIDA	Activación
VehiVerde	X1
VehiAmbar	X2 + X6
VehiRojo	X3 + X4 + X5
PeatVerde	X3 + X5
PeatRojo	X1 + X2

Tabla 6: Combinacional práctica 2

4.3.4. Tabla de variables

Esta tabla de variables ha sido obtenida de la tabla de variables realizada en TIA portal para la programación de esta práctica

Nombre	Tipo	Dirección	Comentario
X0	Bool	%M0.0	
X1	Bool	%M0.1	
X2	Bool	%M0.2	
X3	Bool	%M0.3	
X4	Bool	%M0.4	
X5	Bool	%M0.5	
X6	Bool	%M0.6	
X7	Bool	%M0.7	
Dia	Bool	%I0.0	Variable que activa cuando estamos en horario diurno y se desactiva cuando estamos en horario nocturno
T1	Bool	%M1.0	Variable que se ativa al cumplir el temporizador 1
T2	Bool	%M1.1	Variable que se ativa al cumplir el temporizador 2
T3	Bool	%M1.2	Variable que se ativa al cumplir el temporizador 3
T4	Bool	%M1.3	Variable que se ativa al cumplir el temporizador 4
T5	Bool	%M1.4	Variable que se ativa al cumplir el temporizador 5
VehiVerde	Bool	%Q0.0	Activa la luz verde en el semáforo para vehículos
VehiAmbar	Bool	%Q0.1	Activa la luz ámbar en el semáforo para vehículos
VehiRojo	Bool	%Q0.2	Activa la luz roja en el semáforo para vehículos
PeatVerde	Bool	%Q0.3	Activa la luz verde en el semáforo para peatones
PeatRojo	Bool	%Q0.4	Activa la luz roja en el semáforo para peatones
T6	Bool	%M1.5	Variable que se ativa al cumplir el temporizador 6
Cont	Bool	%M1.6	Variable que se activa cuando se cumple el contador
PPeat	Bool	%I0.1	Variable que se activa al pulsar el pulsador para pedir el paso para los peatones

Tabla 7: Variables práctica 2

4.3.5. Codificación

Si se desea consultar la codificación de este ejercicio práctico, consulte el Anexo número 2.

5. PRÁCTICA 3: DIVERGENCIAS SELECTORAS Y TIPOS DE VARIABLES

5.1. Introducción

Después de las dos primeras prácticas el alumno ya conoce instrucciones básicas para poder realizar asignación de variables, establecer condiciones y utilizar herramientas de programación como contadores y temporizadores.

Ahora llega el turno de aprender a utilizar otro tipo de variables, ya que hasta ahora solo ha utilizado variables digitales o tipo “bool”.

Ya en la práctica anterior se hizo un guiño al tipo de variables “time” al hacer la llamada a un temporizador y se explicó cómo se tenían que definir este tipo de variables para darle un valor temporal.

Para definir una variable analógica, se debe definir en la tabla de variables de TIA portal como una variable tipo “int”.

Para esta práctica se van a utilizar variables analógicas, pero de un modo no tan convencional.

En este caso no es un ejercicio de programación de autómatas al uso en el que se define un proceso industrial y se va esperando cambios en diferentes sensores o actuadores para ir avanzando en el proceso, sino que se va a definir como entrada una clave numérica y según el valor de la misma se va a ir realizando una serie de acciones u otras.

Para procesar la información que se introduce como entrada el alumno deberá de realizar una serie de operaciones matemáticas para ir desglosando este valor numérico a la entrada en unidades de millar, centenas, decenas y unidades, ya que se dará como entrada un número de 4 dígitos y se deberá procesar hasta averiguar cuál es cada uno de esos dígitos.

Esta práctica ayudará al alumno a desenvolverse con soltura no solo con variables digitales, sino también con variables numéricas diferentes, a realizar

operaciones matemáticas y a realizar condiciones para una variable con más de 2 posibles valores para esta, lo que se llama divergencias selectoras.

Para realizar una divergencia selectora se debe de utilizar la instrucción “Case”, que tiene la forma que puede verse en el siguiente ejemplo en lenguaje SCL:

```
CASE variable_name OF
    1:  // Statement section case 1
        ;
    2..4:  // Statement section case 2 to 4
        ;
    ELSE  // Statement section ELSE
        ;
END_CASE;
```

Figura 23: Instrucción CASE

Donde se debe ir introduciendo que acciones se deben realizar para los diferentes valores que pueda tener la variable con la que estamos trabajando.

Esta variable tiene que ser una de tipo ordinal, dentro de las posibles en TIA portal se encuentran las siguientes:

Variables tipo “*WORD”*, que son enteros sin signo con un tamaño de 16 bits.

Variables tipo “*INT”* que corresponde un número entero con signo con un tamaño de 16 bits.

Variables tipo “*DWORD”* que es un número entero doble sin signo con un tamao de 32 bits.

Variables tipo “*DINT”* corresponde a un número entero con signo con un tamaño de 32 bits.

Variables tipo “*REAL”* que corresponde a un valor de 32 bits en coma flotante.

5.2. Software para el encendido de los faros de un vehículo

Una determinada empresa dedicada a la implementación de iluminación en vehículos quiere que se realice la programación de un Driver destinado a la automoción que usará después para realizar el proceso de iluminación de un vehículo mediante el lenguaje de programación SCL.

Antes de nada, hay que entender cómo funciona realmente el encendido de las diferentes funciones de iluminación del coche.

A través de la centralita del coche se manda un mensaje numérico (**TramaLB**), para pedir a vehículo que realice diferentes funciones.

Esta trama tiene la siguiente forma:

$$TramaLB = ABCD \quad [4]$$

Donde “A”, “B”, “C” y “D” son números que dependiendo de su valor estarán pidiendo al driver que realice una determinada función.

En esta práctica se propone la realización de la programación en SCL teniendo en cuenta los siguientes requisitos:

Se comenzarán a mandar mensajes o tramas en cuanto se produzca el encendido del coche (**EnviarTrama**).

Una vez enviada una trama, si dicha trama activa alguna señal de las luces de corto alcance en alguno de los faros, no se volverá a leer otra trama hasta que se active la señal “**TramaOK**”.

Se puede controlar como salidas del sistema el encendido del faro izquierdo (**LBL)** y del faro derecho del coche (**LBR**).

Se deberá indicar al inicio del proceso el valor de la corriente inicial con la que se deben alimentar los LEDs de las luces de los faros (**ICLB**), ya que, dependiendo del modelo del coche, estos serán de un valor u otro. Este valor debe de ser un valor real entre 0 y 1500 mA, que es lo máximo que aguantan los LEDs utilizados.

El cliente indicará que los valores que se tendrán que tener en cuenta de la variable “Trama” son los siguientes:

El término “A” indica el tipo de función para el que va asociado el mensaje.

Los valores del término “A” para la función encendido de las luces son 0 y 1, los valores entre 2 y 9 están dedicados para otras funciones, por lo tanto, mientras la trama no vaya referida a la función de encendido o apagado de las luces del faro, se mantendrá a la espera de la llegada de otra trama.

Término A		
Valor	0-1	2-9
Función	Mensaje de iluminación de faros	Otras funciones

Tabla 8: Funciones del término A

Los valores del término “B” sirven para indican si el mensaje va dirigido para el faro derecho, izquierdo, para ambos o se trata de un mensaje de diagnóstico.

Si el valor del término “B” es igual a 0 se refiere al faro izquierdo, si su valor es 1, se refiere al faro derecho, si su valor es 2 el mensaje va dirigido a ambos faros y si su valor está comprendido entre 3 y 9 significa que es un mensaje de diagnóstico por lo cual se pide que se vuelva a leer la trama hasta que esta envíe un mensaje que se dirija a la función encendido o apagado de las luces de los faros del vehículo.

Término B				
Valor	0	1	2	3-9
Función	Faro izquierdo	Faro derecho	Ambos faros	

Tabla 9 Funciones del término A

El valor de la variable “C” es un CRC o código de seguridad que introduce el cliente, mientras su valor esté entre 0 y 3, todo estará correcto, pero si su valor está comprendido entre 4 y 9 entonces significará que hay un error en la ejecución de la función y se deberá entrar en modo de fallo.

Término A		
Valor	0-3	3-9
Función	OK	Modo Fallo

Tabla 10: Funciones del término C

Si el CRC introducido da una señal de error, se apagarán las luces de corto alcance, y se indicará al conductor mediante una señal de aviso (**Error**).

Si se entra en modo de fallo y se conoce si la señal iba dirigida al faro izquierdo, al derecho o a ambos gracias al valor en “C”, se deberá apagar al faro al que iba dirigida la señal además de mostrar el mensaje de error, que sería común para todos los casos.

Una vez reparado el problema, para salir del modo de fallo se enviará una trama con el valor 7777, hasta entonces el sistema se mantendrá en modo de fallo.

Por último, el término “D” de la variable indicará que hacer con las luces de los faros del vehículo.

Si su valor es un 0, se deben apagar las luces, si su valor es 1 indicará que se deben alimentar los LEDs con una corriente igual a la corriente inicial, si su valor es igual a 2 se deberá alimentar los LEDs con un 20% más de corriente que la indicada inicialmente, si su valor es un 3 se deberán alimentar dichos LEDs con un 40% más de intensidad y finalmente si el valor del término “D” está comprendido entre 4 y 9 se entrará en modo de fallo.

Término D					
Valor	0	1	2	3	4-9
Función	Apagar	Encender con corriente inicial	Encender con la corriente un 20% aumentada	Encender con la corriente un 40% aumentada	Modo de fallo

Tabla 11: Funciones del término C

La corriente que introducimos a los LEDs viene definida tanto para el faro izquierdo (**CurrentLBL**) y como para el faro derecho (**CurrentLBR**).

5.2.1. Solución de la práctica 3

5.2.2. Diseño del GRAFCET

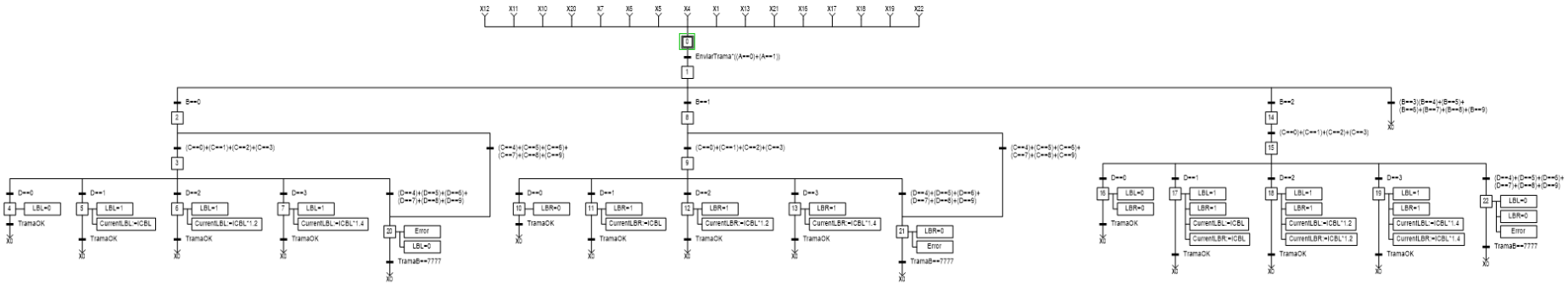

Figura 24: GRAFCET práctica 3

5.2.3. Tabla de evolución secuencial

ETAPA	SET	RESET
X0	I + X1*(B=3..9) + TramaOK*(X4 + X5 + X6 + X7 + X10 + X11 + X12 + X13 + X16 + X17 + X18 + X19) + (TramaLB=7777)*(X20 + X21 + X22)	X1
X1	X0* EnviarTrama*(A=0..1)	I + X2
X2	X1*(B=0)	I + X3 + X20
X3	X2*(C=0..3)	I + X4 + X5 + X6 + X7
X4	X3*(D=0)	I + X0
X5	X3*(D=1)	I + X0
X6	X3*(D=2)	I + X0
X7	X3*(D=3)	I + X0
X8	X1*(B=1)	I + X9 + X21
X9	X8*(C=0..3)	I + X10 + X11 + X12 + X13
X10	X9*(D=0)	I + X0
X11	X9*(D=1)	I + X0
X12	X9*(D=2)	I + X0
X13	X9*(D=3)	I + X0
X14	X1*(B=2)	I + X15 + X21
X15	X14*(C=0..3)	I + X16 + X17 + X18 + X19
X16	X15*(D=0)	I + X0
X17	X15*(D=1)	I + X0
X18	X15*(D=2)	I + X0
X19	X15*(D=3)	I + X0
X20	X2*(C=4..9)	I + X0
X21	X8*(C=4..9)	I + X0
X22	X14*(C=4..9)	I + X0

Tabla 12: Evolución secuencial práctica 3

5.2.4. Tabla combinacional

SALIDA	SET			Reset
LBL	X5 + X6 + X7 + X17 + X18 + X19			X4 + X16
LBR	X11 + X12 + X13 + X17 + X18 + X19			X10 + X16
CurrentLBL	X5 + X17	X6 + X18	X7 + X19	-
	CurrentLBL=ICLB	CurrentLBL=ICLB*1.2	CurrentLBL=ICLB*1.4	-
CurrentLBR	X11 + X17	X12 + X18	X13 + X19	-
	CurrentLBR=ICLB	CurrentLBR=ICLB*1.2	CurrentLBR=ICLB*1.4	-
Error	X20 + X21 + X22			-

Tabla 13: Combinacional práctica 3

5.2.5. Tabla de variables

Nombre	Tipo	Dirección	Comentario
X0	Bool	%M0.0	
X1	Bool	%M0.1	
X2	Bool	%M0.2	
X3	Bool	%M0.3	
X4	Bool	%M0.4	
X5	Bool	%M0.5	
X6	Bool	%M0.6	
X7	Bool	%M0.7	
X8	Bool	%M1.0	
X9	Bool	%M1.1	
X10	Bool	%M1.2	
X11	Bool	%M1.3	
X12	Bool	%M1.4	
X13	Bool	%M1.5	
X14	Bool	%M1.6	
X15	Bool	%M1.7	
X16	Bool	%M2.0	
X17	Bool	%M2.1	
X18	Bool	%M2.2	
X19	Bool	%M2.3	
X20	Bool	%M2.4	
X21	Bool	%M2.5	
X22	Bool	%M2.6	
LBL	Bool	%Q0.0	Activa las luces de corto alcance del faro izquierdo

LBR	Bool	%Q0.1	Activa las luces de corto alcance del faro derecho
TramaLB	Int	%IW6	Trama a enviar
ICLB	Int	%IW8	Corriente inicial a la que deben de ir los LEDs de los faros
CurrentLBL	Int	%MW10	Corriente final a la que van los LEDs del faro izquierdo
CurrentLBR	Int	%MW12	Corriente final a la que van los LEDs del faro derecho
EnviarTrama	Bool	%I0.0	Se activa cuando se quiera enviar una trama al Driver
TramaOK	Bool	%I0.1	Se activa cuando se ha recibido la trama y ha a realizado bien la operación
Error	Bool	%Q0.2	Envía una señal de Error al conductor

Tabla 14: Variables práctica 3

5.2.6. Codificación

Si se desea consultar la codificación de este ejercicio práctico, consulte el Anexo número 3.

6. PRÁCTICA 4: Procesos con simultaneidades

6.1. Introducción

Para esta cuarta práctica el alumno la mayoría de instrucciones básicas en programación de autómatas para el lenguaje de programación SCL.

A estas alturas se debe manejar con normalidad llamadas a herramientas como contadores y temporizadores, ser capaz de trabajar con variables digitales, analógicas, así como temporales.

Además, el alumno debe manejar con soltura instrucciones como asignación de variable e instrucciones condicionales con 2 o más valores diferentes para la misma variable.

Por lo tanto, en esta práctica se pondrá a prueba todo lo aprendido en este manual, incorporando, además, un proceso de simultaneidad.

Finalmente, en esta práctica el alumno deberá realizar la programación de un proceso industrial, en la que en una fábrica de refrescos deben desarrollarse varios procesos simultáneamente.

6.2. Fábrica de refrescos

La Empresa UJA-Cola desea automatizar su planta de producción de refrescos.

En la planta se pueden diferenciar tres procesos que se desarrollan simultáneamente una vez se pulsa el pulsador de puesta en marcha (**PM**):

El primero de ellos es donde se produce el mezclado de productos para obtener la bebida que se desee producir en ese momento. Para la mezcla de componentes se utiliza una tolva acumulativa en la cual primero se introduce agua mediante una válvula que se activa con una señal de salida del PLC (**Agua**), hasta llegar a cierto peso predefinido, se enviará una señal al PLC (**Peso1**), en ese momento se deja de introducir agua en la tolva para empezar a introducir Refresco en polvo (**RefP**).

Una vez alcanzado el peso de la mezcla deseado, se mandará otra señal al PLC (**Peso2**). Este peso no variará dependiendo del tipo de polvos de refresco que se utilicen,

Una vez recibida la señal, abrirá la compuerta inferior de la tolva (**Valv1**) para dejar caer la mezcla en un tanque.

Este proceso se repetirá durante 3 veces para llenar completamente el tanque.

Una vez el tanque contenga el total de la mezcla, se procederá en el mismo al agitado de la mezcla durante 20 segundos para obtener una mezcla uniforme dentro del mismo.

El segundo proceso corresponde con el traslado de las botellas vacías a través de una cinta transportadora (**Cinta**) hasta la zona donde se llenarán con la mezcla debidamente agitada. Se debe activar el movimiento de la cinta trasportadora hasta que un sensor de posición detecte que hay una caja de botellas (**SBotella**).

El tercer proceso implicado en esta simultaneidad tiene que ver con la inicialización de un elevador, el cual puede subir o bajar mediante las señales **UpElev** y **DElev** respectivamente.

Para conocer la posición del elevador se usan tres finales de carrera, uno para conocer cuando este está en la posición inferior (**EDown**), otro para conocer cuando se sitúa en la posición central (**EC**) y finalmente otro para conocer cuando el elevador está situado en la parte superior (**EUp**).

Para terminar con los tres procesos simultáneos se debe tener el producto en el tanque donde se ha producido el agitado listo para comenzar el llenado en las botellas, una caja de botellas en posición, lista para llenarse y el elevador en la planta intermedia.

Una vez finalizado este tramo se procederá al llenado de las botellas mediante la apertura de varias válvulas (**VTanque**) que llenarán todas las botellas de la caja al mismo tiempo.

El proceso de llenado de las botellas se realiza durante exactamente 8 segundos desde que se realiza la apertura de la válvula para el llenado de las mismas.

Finalizado el proceso anterior se procederá a transportar mediante otra cinta transportadora (**Cinta2**) la caja hasta el elevador, se sabrá que se ha situado la caja en el elevador gracias a un sensor de posición (**Selevador)**.

Una vez situada la caja con las botellas en el elevador hay que elegir a que planta debe llevarse. Puesto que cada planta corresponde al proceso de etiquetado de un refresco distinto.

En la planta inferior se produce el etiquetado del refresco sabor naranja, en la planta intermedia el del refresco sabor cola y en la planta superior el del refresco sabor limón.

Para conocer el tipo de refresco que se utiliza un sensor TCS3200 que es capaz de distinguir el color mediante un fotodiodo y proporciona dos salidas digitales que se alternan en valor según el color que se detecte, como se indica en la siguiente tabla:

Color	S1	S2
Blanco	L	L
Amarillo	L	H
Naranja	H	L
Negro	H	H

Tabla 15: Salidas del sensor de color práctica 4

Finalmente se desplaza el elevador hasta la planta correspondiente y se usa una cinta (**CintaElev**) enviar la caja de botellas a la siguiente planta, el proceso vuelve a comenzar una vez se deja de detectar la caja en el elevador.

6.3. Solución de la práctica 4

6.3.1. Diseño del GRAFCET

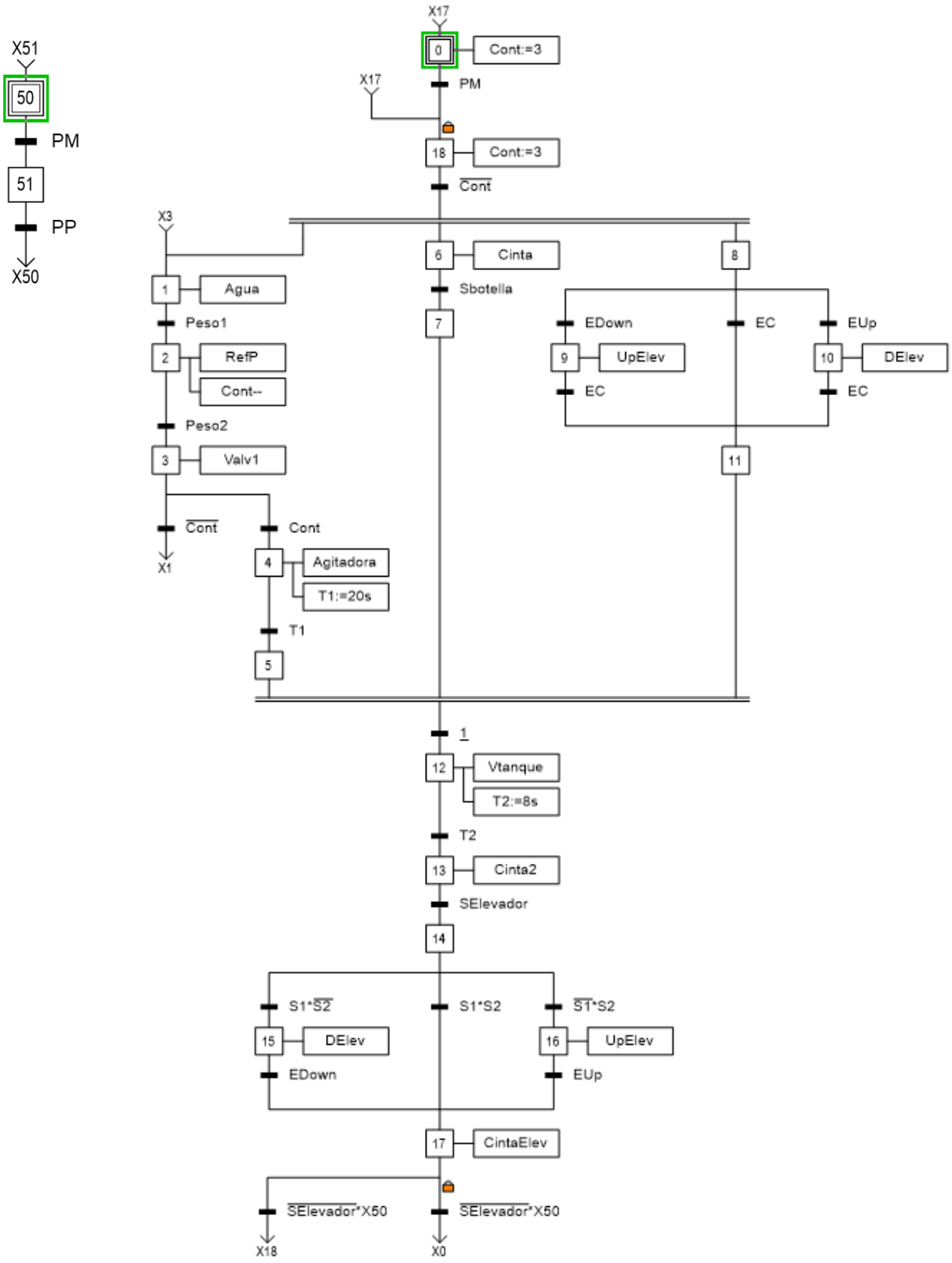

Figura 25: GRAFCET práctica 4

6.3.2. Tabla de evolución secuencial

ETAPA	SET	RESET
X0	I + X17*$\overline{\text{SElevador}}$*X50	X1 + X6 + X8
X1	X18*$\overline{\text{Cont}}$ + X3* $\overline{\text{Cont}}$	I + X2 + X6 + X8
X2	X1*Peso1	I + X3
X3	X2*Peso2	I + X4
X4	X3*Cont	I + X5
X5	X4*T1	I + X12
X6	X0*PM	I + X7
X7	X6*Sbotella	I + X12
X8	X0*PM	I + X9 + X10 + X11
X9	X8*EDown	I + X11
X10	X8*EUp	I + X11
X11	X8*Ec	I + X12
X12	X5 + X7 + X11	I + X13
X13	X12*T2	I + X14
X14	X13*SElevador	I + X15 + X16 + X17
X15	X14*S1*$\overline{\text{S2}}$	I + X17
X16	X14*S1*S2	I + X17
X17	X14*$\overline{\text{S1}}$*S2	I + X0
X18	X0*PM + X17*SElevador*X51	I + X1

Tabla 16: Evolución secuencial práctica 4

6.3.3. Tabla de Combinacional

SALIDA	Activación
Agua	X1
RefP	X2
Valv1	X3
Agitadora	X4
Cinta	X6
DElev	X10
UpElev	X9
VTanque	X12
Cinta2	X13
CintaElev	X17

Tabla 17: Combinacional práctica 4

6.3.4. Tabla de variables

Nombre	Tipo	Dirección	Comentario
X0	Bool	%M0.0	
X1	Bool	%M0.1	
X2	Bool	%M0.2	
X3	Bool	%M0.3	
X4	Bool	%M0.4	
X5	Bool	%M0.5	
X6	Bool	%M0.6	
X7	Bool	%M0.7	
X8	Bool	%M1.0	
X9	Bool	%M1.1	
X10	Bool	%M1.2	
X11	Bool	%M1.3	
X12	Bool	%M1.4	
X13	Bool	%M1.5	
X14	Bool	%M1.6	
X15	Bool	%M1.7	
X16	Bool	%M2.0	
X18	Bool	%M2.5	
X50	Bool	%M2.6	
X51	Bool	%M2.7	
PM	Bool	%I0.0	Pulsador de puesta en marcha
Peso1	Bool	%I0.1	Entrada que indica cuando en la tolva acumulativa se ha llegado al valor deseado de producto 1
Peso2	Bool	%I0.2	Entrada que indica cuando en la tolva acumulativa se ha llegado al valor deseado de producto 1 más producto 2
Sbotella	Bool	%I0.5	Sensor para indicar la llegada de una botella
SElevador	Bool	%I0.7	Sensor que indica si hay producto en el elevador
PP	Bool	%I1.3	Pulsador de paro
RefP	Bool	%Q0.0	Apertura de la válvula que inyecta refresco en polvo
Agua	Bool	%Q0.1	Apertura de la válvula para inyectar agua
Agitadora	Bool	%Q0.2	Inicio del proceso de agitado
Valv1	Bool	%Q0.3	Inyección de la mezcla en el tanque de agitado
Cinta	Bool	%Q0.4	Motor cinta 1
UpElev	Bool	%Q0.5	Subir Elevador

DElev	Bool	%Q0.6	Bajar Elevador
VTanque	Bool	%Q0.7	Inyección del producto en las botellas
Cinta2	Bool	%Q1.0	Motor cinta 2
CintaElev	Bool	%Q1.1	Motor cinta 3
X17	Bool	%M2.1	
EDown	Bool	%I0.3	Sensor de posición que indica que el elevador está en la parte inferior
EUp	Bool	%I0.6	Sensor de posición que indica que el elevador está en la parte superior
EC	Bool	%I1.0	Sensor de posición que indica que el elevador está en la parte intermedia
S1	Bool	%I1.1	Salida digital número 1 del sensor de color
S2	Bool	%I1.2	Salida digital número 2 del sensor de color
T1	Bool	%M2.2	Variable que se activa cuando se carga el temporizador número 1
T2	Bool	%M2.3	Variable que se activa cuando se carga el temporizador número 2
Cont	Bool	%M2.4	Variable que se carga cuando el valor del contador 1 es igual a 3

Tabla 18: Variables práctica 4

6.3.5. Codificación

Si se desea consultar la codificación de este ejercicio práctico, consulte el Anexo número 4.

7. PRÁCTICA 5: Uso de bloques de función (FC)

7.1. Introducción

El alumno se encuentra en la penúltima práctica y en este caso se va a enfocar en la creación de bloques de función sin memoria o FC.

Una FC es un bloque de función en el que se puede crear una subrutina que puede realizarse varias veces realizando una llamada a dicha FC.

Estos bloques de función no tienen memoria por lo tanto no pueden llamarse simultáneamente en dos situaciones y no trabaja con parámetros actuales.

Para crear una un bloque de función se debe acceder a la sección de bloques de programa, véase el punto 2.3. Interfaz de programa de este manual.

Una vez en esta sección se utilizará la opción de agregar nuevo bloque y se seleccionará la opción función FC al igual que en la siguiente figura.

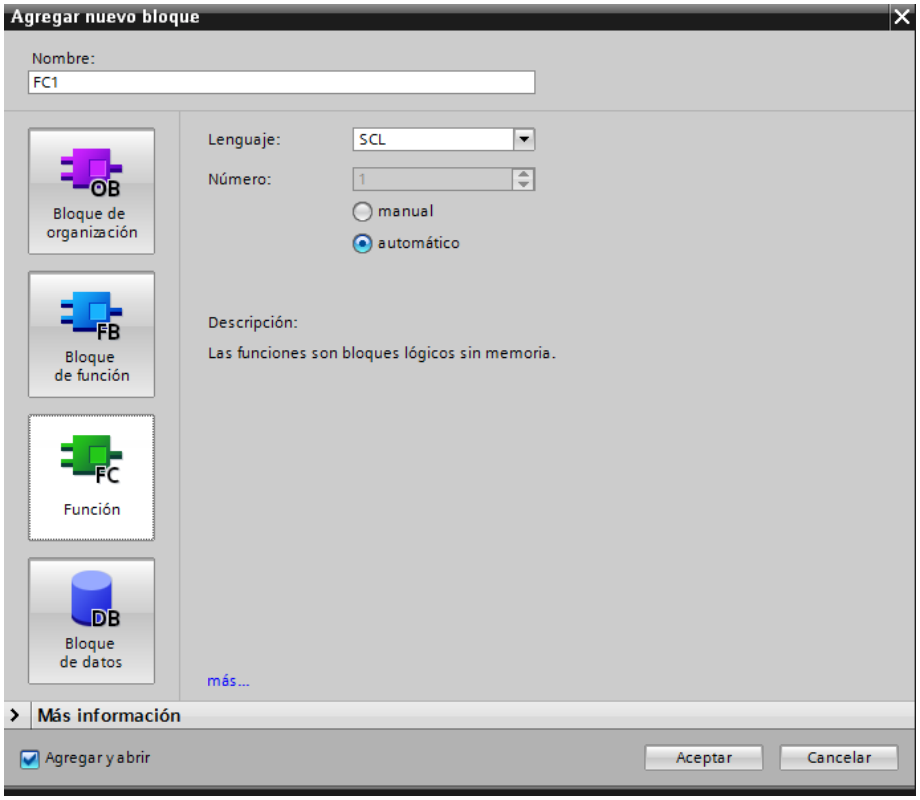

Figura 26: Agregar función TIA portal

Para realizar una llamada a dicha FC en nuestro OB1 o bloque principal se realiza con la siguiente sintaxis:

```
"FC1"();
```

Figura 27: Llamada a un bloque de función

7.2. Domótica

La empresa AutomatizoTodo ha sido contratada para automatizar las viviendas de una determinada urbanización, la descripción de la automatización deseada en ellas es la siguiente:

Para inicializar el sistema de automatizado de la vivienda se usará un pulsador de marcha (**PM**) y para desconectarlo se hará mediante un pulsador de paro (**PP**).

Se han determinado 3 diferentes procesos de cada vivienda a automatizar:

El primer proceso, es el de iluminación de la casa.

Por este motivo se han colocado un sistema sensores de luminosidad y una serie de toldos y persianas para evitar que la luz directa moleste a los habitantes de las viviendas.

Por lo tanto, utilizando un sensor de luminosidad y un convertidor A/D, se le proporcionará a nuestro autómata una entrada digital que indicará si hay una luminosidad exterior por encima de un valor definido (**Sol**).

En el caso de que dicha luminosidad esté por encima de dicho valor se procederá a activar las funciones de desplegado de toldos (**Toldo**) y persianas (**Persianas**).

No obstante, los toldos deberían replegarse, cuando la fuerza con la que sople el viento pueda ser peligrosa para el uso de los mismos. El valor de esta fuerza se ha considerado peligrosa cuando supere los 50km/h y un sensor de viento proporcionará una señal digital (**Viento**) al autómata cuando supere dicho valor.

Si la luminosidad exterior está por debajo del valor predefinido, se procederá a activar la iluminación de la vivienda (**Iluminación**).

El segundo proceso a automatizar es el de la Climatización de la vivienda:

Para lo cual se desea que se active el aire acondicionado en modo frío siempre que la temperatura se encuentre por encima de 28ºC y que se active el modo calor cuando se encuentre por debajo de 20ºC.

Una vez activados los modos Frío o Calor, no se desactivará la climatización hasta haber llegado a los 25ºC de temperatura ambiente.

La temperatura ambiente será registrada a través de un sensor de temperatura M35 (**Tamb**) que proporcionará los siguientes valores analógicos dependiendo de la temperatura en el interior de la casa:

Temperatura	Señal analógica
20ºC	200 mV
28ºC	280 mV

Tabla 19: Salidas del sensor de temperatura

La activación o desactivación del aire acondicionado se realizará mediante la variable **AireON** y se elegirá entre las funciones frío y calor con la activación de las variables **Cold** y **Hot** respectivamente.

El tercer proceso tiene que ver con la seguridad de la vivienda y para ello se debe realizar la programación de la alarma de la vivienda.

La alarma debe estar programada para proteger la vivienda contra intrusos que deseen entrar cuando no haya nadie en casa.

La alarma debe iniciarse cuando alguno de los integrantes de la casa pulse el botón de encendido (**AlarmaON**)

Una vez activada la alarma, el propietario tiene 20 segundos para para salir de la vivienda y cerrar la puerta.

El autómata detectará si la puerta está abierta o cerrada a través de un sensor de final de carrera (**ODoor**) situado en la cerradura de la puerta.

Si al pasar estos 20 segundos la puerta está cerrada, la alarma se mantendrá en estado de espera hasta que se vuelva a abrir la puerta, en ese momento la

persona que abra la puerta dispondrá de 15 segundos para introducir la contraseña de la alarma.

Si la persona introduce la contraseña correcta, la alarma volverá al estado inicial, esperando a ser activada de nuevo, pero si por el contrario pasan los 15 segundos o se introduce mal la contraseña, se activará una alarma sonora (**Alarma)**.

Si por el contrario al pasar los 20 segundos desde la activación de la alarma, se detecta que la puerta sigue abierta se activará la alarma sonora igualmente.

Una ver la alarma sonora está activada, la única manera de pararla es introduciendo la contraseña correcta, con lo cual volvería al estado inicial.

Para la introducción de la contraseña en la alarma se debe escribir en la Variable **Pass** la contraseña que se debe introducir y pulsar el botón de introducir la contraseña (**InputPass)**.

Para esta práctica se usará la contraseña “2017”, aunque el alumno puede cambiarla en el código si lo desea.

7.3. Solución de la práctica 5

7.3.1. Diseño del GRAFCET

7.3.1.1. OB1

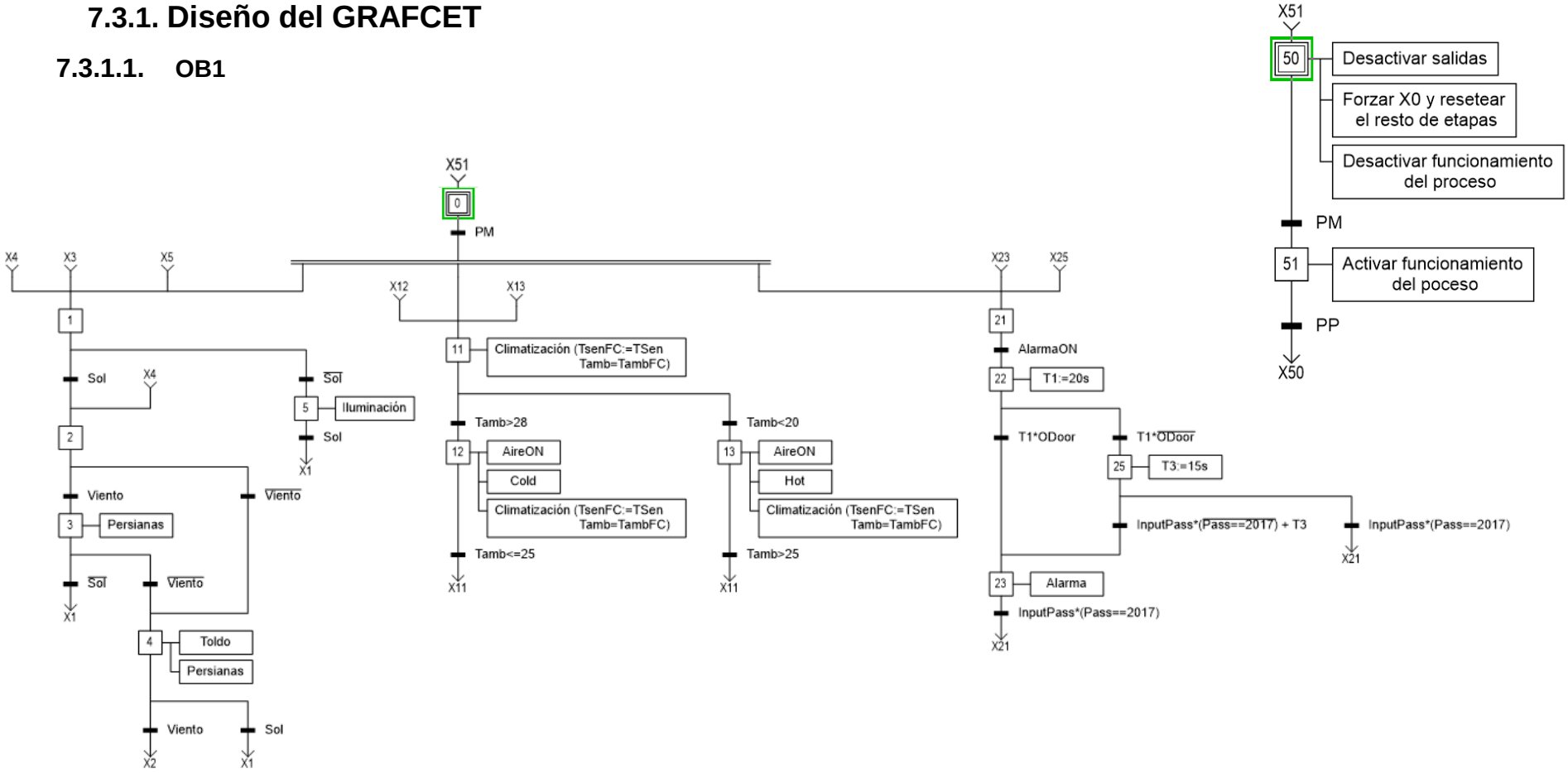

Figura 28: GRAFCET OB1 práctica 5

7.3.2. Tabla evolución secuencial

ETAPA	SET	RESET
Bloque principal		
X1	X0*PM + X3* +X4* + X5*Sol	I +X2 + X5
X2	X1*Sol + X4*Viento	I + X3 + X4
X3	X2*Viento	I + X1 + X4
X4	X1*X3*	I + X1 + X2
X5	X1*	I + X1
X11	X0*PM + X12*(Tamb<260) + X13*(Tamb>240)	I + X12 + X13
X12	X11*(Tamb>280)	I + X11
X13	X11*(Tamb<200)	I + X11
X21	X0*PM +(X26 + X23)*InputPass*(Pass=2017)	I + X22
X22	X21 + AlarmaON	I + X23 + X25
X23	X22*T1*ODoor + X26*T3 + X26*InputPass*(Pass2017)	I + X21
X25	X22*T1*$\overline{\text{ODoor}}$	I + X23 + X21

Tabla 20: Evolución secuencial práctica 5

7.3.3. Tabla combinacional

SALIDA	SET
Persianas	X3 + X4
Toldo	X4
Ilumin	X5
AireON	X12 + X13
Cold	X12
Hot	X13
Alarma	X23

Tabla 21: Combinacional práctica 5

7.3.4. Tabla de variables

Nombre	Tipo	Dirección	Comentario
Sol	Bool	%I0.0	Sensor de luminosidad
Viento	Bool	%I0.1	Sensor para medir la fuerza del viento
AlarmaON	Bool	%I0.2	Pulsador para conectar la alarma
InputPass	Bool	%I0.6	Pulsador para introducir contraseña de la alarma
ODoor	Bool	%I0.7	Sensor para determinar si la puerta está abierta o cerrada
PM	Bool	%I1.2	Pulsador de puesta en marcha
PP	Bool	%I1.3	
Tamb	Int	%IW4	Salida en voltaje del sensor de temperatura
Tsen	int	%MW	Conversión de voltaje en temperatura
Pass	Int	%IW8	Contraseña
Persianas	Bool	%Q0.0	Activación de las persianas para proteger de iluminación solar
Toldo	Bool	%Q0.1	Activación de los toldos para proteger de iluminación solar
Ilumin	Bool	%Q0.2	Activación de la iluminación de la vivienda
AireOn	Bool	%Q0.3	Activación del sistema de climatización
Cold	Bool	%Q0.4	Salida para bajar la temperatura ambiente
Hot	Bool	%Q0.5	Salida para subir la temperatura ambiente
Alarma	Bool	%Q0.7	Activación de la alarma sonora
X0	Bool	%M0.0	
X1	Bool	%M0.1	
X2	Bool	%M0.2	
X3	Bool	%M0.3	
X4	Bool	%M0.4	
X11	Bool	%M0.5	
X12	Bool	%M0.6	
X13	Bool	%M0.7	
X21	Bool	%M1.3	
X22	Bool	%M1.4	
X23	Bool	%M1.5	
X25	Bool	%M1.7	
X26	Bool	%M2.0	
X31	Bool	%M2.1	
X32	Bool	%M2.2	
X33	Bool	%M2.3	
X5	Bool	%M2.6	

X34	Bool	%M2.7	
T5	Bool	%M3.2	Final del temporizador 5
T4	Bool	%M3.3	Final del temporizador 4
T1	Bool	%M3.4	Final del temporizador 1
T3	Bool	%M3.6	Final del temporizador 3

Tabla 22: Variables práctica 5

7.3.5. Codificación

Si se desea consultar la codificación de este ejercicio práctico, consulte el Anexo número 4.

8. PRÁCTICA 6: Uso de bloques de función con memoria

8.1. Introducción

Con esta práctica se daría por concluida la formación del alumno en programación de autómatas en lenguaje SCL.

Hasta ahora se ha desarrollado una serie de prácticas graduales en las que el alumno ha aprendido a trabajar con diferentes tipos de variables, analógicas, digitales y temporales, conoce como realizar diferentes tipos de funciones matemáticas y realizar condiciones usando el valor de una variable digital, usando varios valores diferentes para una misma variable analógica o multiplicaciones y sumas lógicas entre varias variables.

Además, debe de manejar con soltura herramientas como temporizadores y contadores que serán usados también en esta práctica.

En el ejercicio anterior se explicó el funcionamiento de una FC o bloque de función y para esta práctica final se va a aprender a usar los bloques de funciones con memoria.

La diferencia entre una FC y una FB es la siguiente:

Una FB es un bloque de función donde se puede crear una subrutina que se podrá usar tantas veces como se desee utilizando parámetros actuales.

Estos FB tienen asociados DB de instancias, que son la memoria de dichos FB, donde estarán definidos los parámetros normales, es decir, las variables utilizadas dentro de estos FB estarán guardadas en los DB.

En cambio, una FC es un bloque de función que no tiene memoria, por lo que sus variables no se almacenan.

Para el uso de estas FB se plantea un ejercicio en el que se deben proceder a un proceso de similares características en 2 sitios diferentes, realizados por el mismo FB, dependiendo de si se va a trabajar con una pieza metálica o de plástico.

8.2. Proceso de separación de piezas

Se desea automatizar parte del proceso en la fabricación de juguetes, en concreto una zona la planta de trabajo en la que se dedican a seleccionar y empaquetar diferentes tipos de piezas.

Por lo tanto, se desea que una vez se utilice un pulsador de marcha (**PM**), comience a funcionar el motor de una cinta transportadora (**Cinta1**) que traslada diferentes piezas. Estas piezas pueden ser metálicas o de plástico.

Para conocer el material del que está hecha la pieza, se dispone de dos sensores, en primer lugar, se encuentra un sensor ferromagnético (**SFerro**) que detecta si hay una pieza de metal, a poca distancia, se encuentra otro sensor de posición que detecta si hay una pieza sin importar el material con la que esté fabricada (**SPieza**).

Cuando se detecte una pieza gracias a este último sensor, se detendrá el motor de la cinta trasportadora y en caso de que sea una pieza metálica se accionará un cilindro de simple efecto (**CDer**) para llevar a esa pieza a una plataforma situada a la derecha de la cinta transportadora.

Si por el contrario se detecta una pieza de plástico, se detendrá el motor de la cinta trasportadora y se accionará un cilindro de simple efecto (**CIzq**) para llevar a esa pieza a una plataforma situada a la izquierda de la cinta transportadora.

Aunque se produzca en sitios diferentes el proceso a realizar cuando se tienen el total de piezas de alguno de los dos tipos esperando en su plataforma correspondiente es el mismo.

Primero se accionará un cilindro de simple efecto (**Cilindro**) para elevar una cintra transportadora por uno de los dos lados, de forma que quede inclinada, se conoce que 3 segundos es un tiempo suficiente para que termine la elevación de dicha cinta.

Acto y seguido se activará dicha cinta transportadora (**Cinta**) hasta que se detecte al final de ella las piezas con un sensor de posición (**SCint**).

Después de esto se volverá a poner la cinta transportadora en posición horizontal. Al igual que para elevarla, se considera que un tiempo de 3 segundos es suficiente para realizar dicha acción.

Finalmente se volverá a llevar a dichas piezas después del proceso de separación a una cinta transportadora situada en la parte central (**Cinta2**), con la activación de un cilindro de simple efecto para las piezas metálicas (**CFe**) y con otro cilindro de simple efecto diferente si se trata de una pieza de plástico (**CPl**).

Cualquiera de los dos cilindros se mantendrá activado durante dos segundos.

Para finalizar el proceso se activará esta última cinta transportadora (**Cinta2**) durante 10 segundos para llevar las piezas a la zona de empaquetado.

Cuando transcurra este tiempo se volverá a reiniciar activando la primera cintra transportadora donde hay piezas de los dos tipos.

Para finalizar el proceso se debe de usar un pulsador de paro (**PP**), al inicio del proceso antes de que una pieza llegue a los sensores para no producir errores.

8.3. Solución de la práctica 6

8.3.1. Diseño del GRAFCET

8.3.1.1. OB1

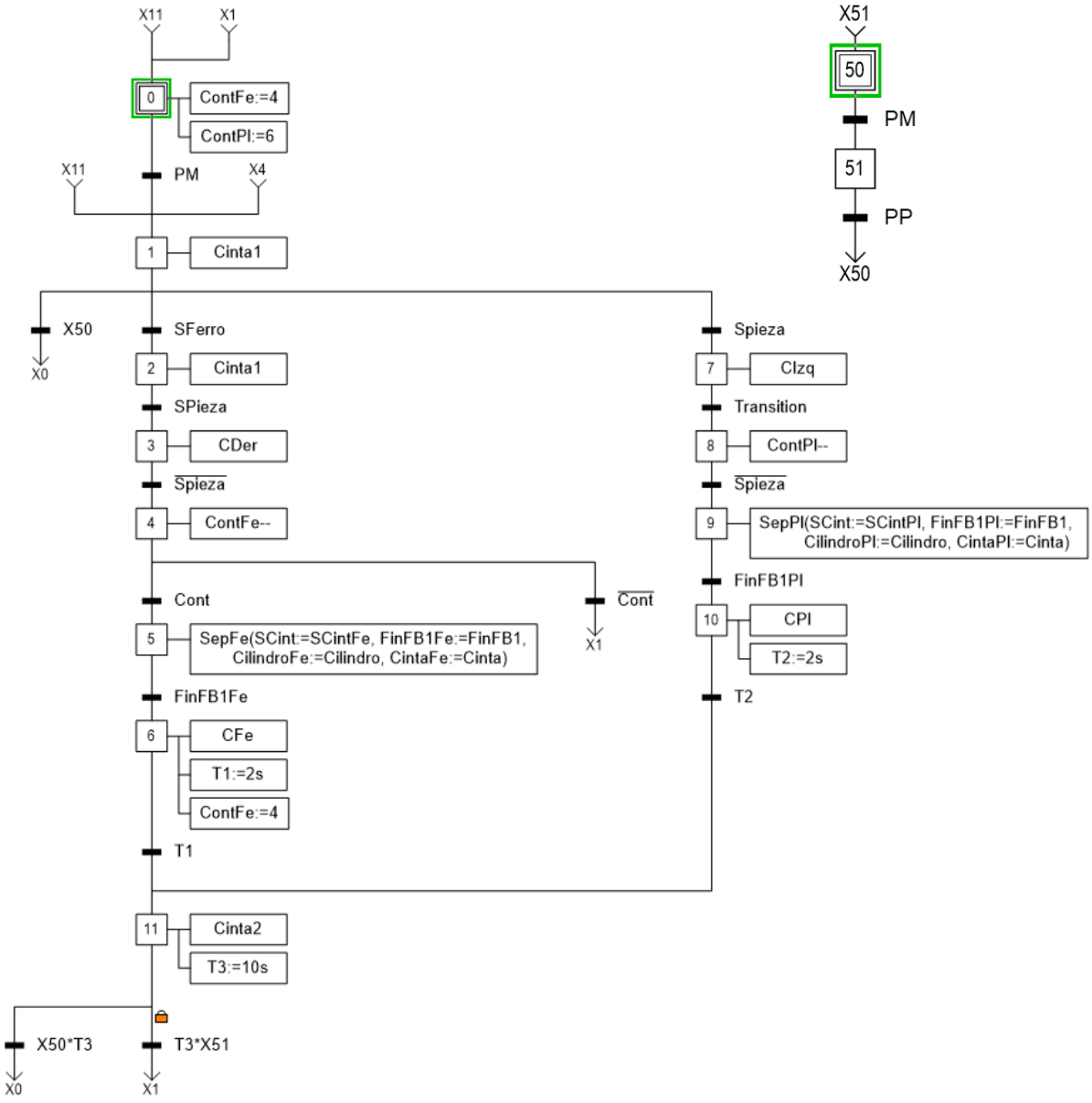

Figura 29: GRAFCET OB1 práctica 6

8.3.1.2. FB1

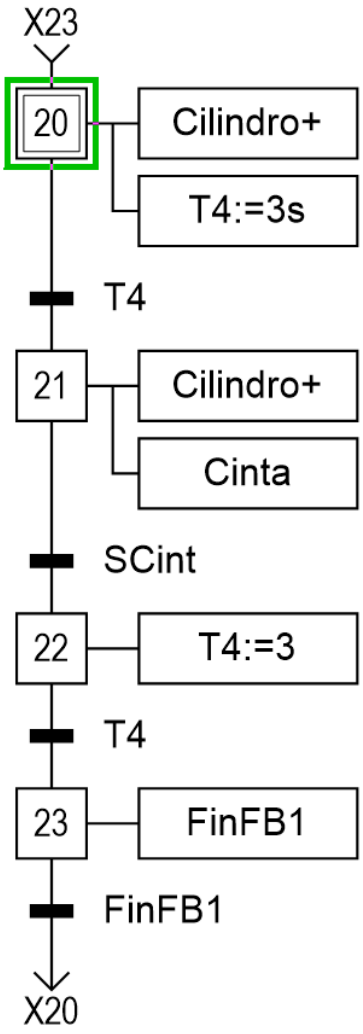

Figura 30: GRAFCET FB1 práctica 6

8.3.2. Tabla evolución secuencial

ETAPA	SET	RESET
OB1		
X0	I + X1*PP	X1
X1	X0*PM + X4*$\overline{(\text{ContFe} == 4)}$ + X8*$\overline{(\text{ContPl} == 6)}$ + X11*T3	I + X2 + X7
X2	X1*SFerro	I + X3
X3	X2*SPieza	I + X4
X4	X3* $\overline{\text{SPieza}}$	I + X5 + X1
X5	X4*(ContFe==4)	I + X6
X6	X5*FinFB1	I + X11
X7	X1* SPieza	I + X8
X8	X7* $\overline{\text{SPieza}}$	I + X9
X9	X8*(ContPl==6)	I + X10
X10	X9*FinFB1Pl	I + X11
X11	T3*(X10 + X11)	I + X1
FC1		
X20	X5 + X9	I + X21
X21	X20*T4	I + X22
X22	X21*SCint	I + X23
X23	X22*T4	I + X1 + X6 + X10

Tabla 23: Activación/desactivación de etapas práctica 6

8.3.3. Tabla combinacional

SALIDA	Activación
Cinta1	X1 + X2
CDer	X3
Cizq	X7
CIiz2	X6
CDer2	X10
Cinta2	X11
Cilindro	X20 + X21
Cinta	X21

Tabla 24: Activación de salidas práctica 6

8.3.4. Tabla de variables

Nombre	Tipo	Dirección	Comentario
X0	Bool	%M0.0	
X1	Bool	%M0.1	
X2	Bool	%M0.2	
X3	Bool	%M0.3	
X4	Bool	%M0.4	
X5	Bool	%M0.5	
X6	Bool	%M0.6	
X7	Bool	%M0.7	
X8	Bool	%M1.0	
X9	Bool	%M1.1	
X10	Bool	%M1.2	
X11	Bool	%M1.3	
T1	Bool	%M2.0	
T2	Bool	%M2.1	
T3	Bool	%M2.2	
T4	Bool	%M2.3	
ContFe	Bool	%M2.4	Variable que se actica cuando hay esperando 4 piezas metálicas
ContPl	Bool	%M2.5	Variable que se activa cuando hay esperando 6 piezas de plástico
PM	Bool	%I0.0	Pulsador de marcha
SFerro	Bool	%I0.1	Sensor ferromagnético que detecta piezas metálicas
SPieza	Bool	%I0.2	Sensor de posición que detecta cuando hay una pieza delante
PP	Bool	%I0.3	Pulsador de paro
SCintFe	Bool	%I0.4	Sensor que detecta cuando de posición las piezas metálicas llegan a su fin
SCintPl	Bool	%I0.5	Sensor que detecta cuando de posición las piezas de plástico llegan a su fin
Cinta1	Bool	%Q0.0	Señal que activa la cinta transportadora 1
CDer	Bool	%Q0.1	Cilindro de simple efecto que mueve las piezas metálicas a la plataforma de espera
CIzq	Bool	%Q0.2	Cilindro de simple efecto que mueve las piezas de plástico a la plataforma de espera
CFe	Bool	%Q0.3	Cilindro que mueve un conjunto de 4 piezas metálicas a una cinta transportadora central
CPl	Bool	%Q0.4	Cilindro que mueve un conjunto de 6 piezas de plástico a una cinta transportadora central

CilindroFe	Bool	%Q0.5	Cilindro de simple efecto que inclina la cinta transportadora para piezas metálicas
CilindroPl	Bool	%Q0.6	Cilindro de simple efecto que inclina la cinta transportadora para piezas de plástico
CintaFe	Bool	%Q0.7	Señal que activa la cinta transportadora para piezas metálicas
CintaPl	Bool	%Q1.0	Señal que activa la cinta transportadora para piezas de plástico
Cinta2	Bool	%Q1.1	Señal que activa el motor de la cinta transportadora 2

Tabla 25: Variables práctica 6

8.3.5. Codificación

Si se desea consultar la codificación de este ejercicio práctico, consulte el Anexo número 3.

9. CONCLUSIÓN

Se ha planteado en este proyecto de una manera ordenada y eficaz como introducir al alumno en el mundo de la programación de autómatas utilizando un lenguaje de programación de alto nivel.

En un primer lugar se ha realizado una guía para el inicio del uso del programa TIA portal con el que se van a desarrollar todas las prácticas que se especifican en este proyecto.

Desde un primer momento se ha planteado el aprendizaje del alumno de manera gradual diseñando una serie de ejercicios prácticos en los que se comienza con una introducción al lenguaje de programación SCL como primera toma de contacto con el mismo.

Esto es necesario ya que este lenguaje de programación es definido como un lenguaje de programación de alto nivel estructurado de control basado en texto.

Durante el desarrollo de la asignatura los ejercicios prácticos introducidos en este proyecto harán que el alumno complete una formación que le será de ayuda en siguientes asignaturas relacionadas con la programación de autómatas, así como en el ámbito laboral.

El hecho de realizar este proyecto con lenguaje SCL es muy importante ya que este lenguaje de programación, aunque tiene una mayor complejidad a la hora de trabajar con él, permite desarrollar proyectos de mayor complejidad que con otros lenguajes de programación, como KOP, AWL o FUP.

10. BIBLIOGRAFÍA

SIEMENS. *Manual SCL para SIMATIC S7-300/400 Programación de bloques*. Ref 6ES7811-1CA02-8DA0-01.

TAOS. *CS3200, TCS3210 PROGRAMMABLE COLOR LIGHT-TO-FREQUENCY CONVERTER Datasheet.*

TEXAS INSTRUMENTS, *LM35 Precision Centigrade Temperature Sensors Datasheet.*