Cuadernillo del Taller de Automatización Industrial - Tomo II

Tec. Juan Pedro Lardet

Año: 2026

Licencia Creative Commons Atribución – No Comercial – Compartir Igual 4.0 Internacional.

Autor: Juan Pedro Lardet

Año: 2026

Se permite copiar, distribuir y modificar el material citando autoría y sin fines comerciales.

Índice general

Prólogo

Este segundo tomo nace de la necesidad de profundizar en los conceptos que quedaron esbozados en el primer cuadernillo. Si en el Tomo I aprendimos a caminar, aquí daremos pasos más largos y complejos: nos adentraremos en el mundo de las señales analógicas, los tipos de datos avanzados, los algoritmos estructurados y la simulación realista de procesos industriales mediante Factory I/O y PLCSIM.

El objetivo es que el estudiante sea capaz de abordar problemas de automatización de mediana complejidad, integrando sensores continuos, actuadores proporcionales y lógica de control más sofisticada. Todo ello con un enfoque práctico y apoyado en herramientas profesionales.

Agradezco a todos los estudiantes y colegas que, con sus inquietudes y desafíos, han impulsado la creación de este material. Espero que sea de utilidad en su formación y en su futura labor profesional.

Juan Pedro Lardet
2026

Introducción General

En el Tomo I nos familiarizamos con los fundamentos: lógica binaria, temporizadores, contadores, y la programación básica de PLCs. Ahora es momento de dar el salto a los sistemas que manejan variables continuas, como presión, temperatura, nivel o caudal. Estos sistemas requieren un tratamiento diferente: ya no basta con encender o apagar; hay que medir, comparar, calcular y actuar de forma proporcional.

Además, exploraremos lenguajes de programación más potentes como SCL (Structured Control Language) y técnicas de estructuración de código que nos permitirán abordar proyectos más grandes sin perder claridad.

Finalmente, la simulación 3D con Factory I/O nos brindará un entorno realista para probar nuestros algoritmos antes de llevarlos a un PLC real.

Parte I

Fundamentos de la Programación Avanzada

1 Repaso de Conceptos Básicos y Entorno TIA Portal

1.1. El ciclo de escaneo y su impacto en la programación

Recordemos que un PLC ejecuta su programa de forma cíclica: lectura de entradas, ejecución del programa y escritura de salidas. Este ciclo, conocido como *scan cycle*, tiene una duración típica de entre 1 ms y 150 ms. En aplicaciones sencillas este tiempo no es crítico, pero cuando trabajamos con señales analógicas o lazos de control rápidos, debemos considerar el retardo introducido por el ciclo.

Por ejemplo, si necesitamos leer una señal analógica y actuar sobre una salida en el mismo ciclo, el tiempo de respuesta será al menos un ciclo completo. En aplicaciones con realimentación, esto puede afectar la estabilidad del sistema.

1.2. Tipos de bloques en TIA Portal: OB, FB, FC, DB (repaso)

En TIA Portal, el código se organiza en bloques:

- **OB (Organization Block)**: Controlan la ejecución del programa. El más importante es el OB1 (ciclo principal). También existen OBs de arranque, de alarma, de error, etc.
- **FB (Function Block)**: Bloques con memoria. Tienen una instancia asociada en un bloque de datos (DB) que guarda los valores de sus variables internas entre llamadas. Ideales para máquinas de estado, temporizadores, etc.
- **FC (Function)**: Bloques sin memoria. Todas las variables locales se pierden al finalizar la ejecución. Útiles para operaciones matemáticas o lógicas reutilizables.
- **DB (Data Block)**: Almacenan datos globales o de instancia. Pueden ser globales (accesibles desde cualquier bloque) o de instancia (asociados a un FB).

1.3. Lenguajes de programación: LADDER, FBD, SCL (introducción)

TIA Portal soporta varios lenguajes de programación:

- **LADDER (KOP)**: Basado en diagramas de contactos. Es el más intuitivo para técnicos electricistas.

- **FBD (Function Block Diagram)**: Utiliza bloques funcionales interconectados. Muy útil para procesamiento de señales analógicas.
- **SCL (Structured Control Language)**: Lenguaje de alto nivel similar a Pascal. Permite implementar algoritmos complejos de forma compacta y estructurada.

En este tomo utilizaremos principalmente LADDER y SCL, y ocasionalmente FBD para operaciones analógicas.

1.4. Herramientas de depuración: tablas de vigilancia, referencias cruzadas

Para depurar programas, TIA Portal ofrece:

- **Tablas de vigilancia (watch tables)**: Permiten monitorizar y forzar variables en tiempo real.
- **Referencias cruzadas**: Muestran dónde se utiliza cada variable, ayudando a localizar errores.
- **Lista de errores**: Muestra los errores de compilación y advertencias.
- **Modo de simulación (PLCSIM)**: Permite probar el programa sin hardware real.

1.4.1. Ejercicios del capítulo 1

1. Explique por qué el tiempo de ciclo de escaneo puede afectar el control de un lazo analógico.
2. Diferencie entre un FB y un FC. ¿Cuándo usaría cada uno? Enumere los lenguajes de programación disponibles en TIA Portal y mencione una aplicación típica para cada uno.
3. Investigue cómo crear una tabla de vigilancia en TIA Portal y describa su utilidad.

2 Tipos de Datos en Automatización

2.1. Datos elementales: BOOL, BYTE, WORD, DWORD, INT, DINT, REAL

En la programación de PLCs, las variables tienen un tipo de datos que determina su tamaño y el rango de valores que pueden almacenar. Los más comunes son:

Tipo	Descripción	Tamaño
BOOL	Booleano (verdadero/falso)	1 bit
BYTE	Número sin signo de 8 bits	1 byte
WORD	Número sin signo de 16 bits	2 bytes
DWORD	Número sin signo de 32 bits	4 bytes
INT	Entero con signo de 16 bits	2 bytes
DINT	Entero con signo de 32 bits	4 bytes
REAL	Número en coma flotante (32 bits)	4 bytes

Cuadro 2.1: Tipos de datos elementales en TIA Portal.

2.2. Rango y representación: enteros con signo, coma flotante

Es fundamental conocer el rango de cada tipo para evitar desbordamientos:

- **INT**: de -32 768 a 32 767.
- **DINT**: de -2 147 483 648 a 2 147 483 647.
- **REAL**: aproximadamente ±1.18e-38 a ±3.40e38, con 6-7 dígitos de precisión decimal.
- **BYTE, WORD, DWORD**: sin signo, respectivamente 0 a 255, 0 a 65 535, 0 a 4 294 967 295.

2.3. Constantes y literales

En los programas podemos usar constantes directamente:

- Booleanas: `true`, `false`
- Enteros: `5`, `-10`, `16FF` (hexadecimal)
- Reales: `3.1416`, `1.2e3`

2.4. Variables globales y locales: tablas de tags y bloques de datos

En TIA Portal, las variables se pueden definir en:

- **Tablas de tags (PLC tags)**: Variables globales accesibles desde cualquier bloque. Se asignan a direcciones físicas (%I, %Q, %M) o simbólicas.
- **Bloques de datos (DB)**: Pueden contener variables globales (DB global) o variables de instancia (asociadas a un FB).
- **Variables temporales (Temp)**: Se definen dentro de bloques y solo existen durante la ejecución del bloque.

2.5. Conversión entre tipos (CONV)

A menudo es necesario convertir un tipo a otro. TIA Portal proporciona la instrucción `CONV`. Por ejemplo, convertir un entero a real:

$$\#realVar := CONV(INT\_TO\_REAL(\#intVar))$$

También se pueden usar operadores de conversión implícita en algunos casos, pero es recomendable hacerlas explícitas para evitar errores.

2.5.1. Ejercicios del capítulo 2

1. Declare una variable de tipo DINT y asígnele el valor máximo posible. ¿Qué ocurre si intenta sumarle 1?
2. ¿Cuál es la diferencia entre una variable global definida en una tabla de tags y una variable en un DB global?
3. Escriba en TIA Portal un segmento que convierta un valor entero (INT) en un real (REAL) y lo multiplique por 2.5.
4. Investigue qué es el *endianness* y cómo afecta a la comunicación entre PLCs.

Parte II

Procesamiento de Señales Analógicas

3 Adquisición de Señales Analógicas

3.1. Sensores analógicos: tipos y conexión al PLC

En la industria, muchas variables son de naturaleza continua: presión, temperatura, nivel, caudal, velocidad, etc. Para capturar estas magnitudes se utilizan **sensores analógicos**, que entregan una señal proporcional a la variable medida. Las señales más comunes son:

- **4-20 mA** (corriente): Muy utilizada por su inmunidad al ruido y capacidad de detectar rotura de cable (si la corriente es 0 mA hay fallo). Rango típico: 4 mA = valor mínimo, 20 mA = valor máximo.
- **0-10 V** (tensión): Simple pero más susceptible a caídas de tensión y ruido. Se usa en distancias cortas.

La conexión al PLC se realiza mediante **módulos de entrada analógica** (AI). Estos módulos convierten la señal física a un valor digital que la CPU puede procesar.

3.2. Módulos de entrada analógica: resolución, rangos, configuración en TIA Portal

Los módulos analógicos se caracterizan por:

- **Número de canales**: 2, 4, 8, etc.
- **Resolución**: típicamente 12, 14 o 16 bits. Determina la precisión de la conversión. Por ejemplo, un módulo de 16 bits divide el rango en 65536 niveles.
- **Rango de entrada**: Configurable (0-10V, 4-20mA, ±10V, etc.).
- **Tiempo de conversión**: Tiempo que tarda en digitalizar una señal.

En TIA Portal, para configurar un módulo analógico:

1. Agregar el módulo en la configuración del hardware (ej. SM 1231 AI 4x13 bits).
2. En las propiedades del módulo, seleccionar el rango de cada canal (por ejemplo, “4-20 mA”).
3. Asignar las direcciones de E/S (por defecto se asignan automáticamente).

3.3. Lectura de valores analógicos: direccionamiento (IW, PIW)

Una vez configurado, el valor digitalizado se almacena en una dirección de entrada. Por ejemplo, si el módulo ocupa las direcciones `IW64` (palabra de entrada), podemos leerlo con:

- `IW64` (WORD) o `PIW64` (periferia, lectura directa sin imagen de proceso). Es más común usar `IW` para trabajar con la imagen de proceso.

El valor leído es un número entero sin signo (WORD) que varía entre 0 y 27648 (valor típico para señales de 0-10V o 4-20 mA en la mayoría de módulos SIEMENS). Algunos módulos usan 0-27648 como rango normalizado.

3.4. Ejemplo práctico: lectura de un potenciómetro o sensor de temperatura

Supongamos que tenemos un sensor de temperatura 4-20 mA conectado al canal 0 de un módulo analógico. La dirección asignada es `IW64`. Queremos leer ese valor y almacenarlo en una variable `temp_raw` de tipo INT.

En LADDER, podemos usar una simple asignación (MOVE):

Listing 3.1: Lectura de valor analógico en LADDER

```
1    MOVE
2    EN ENO
3    IW64 IN OUT temp_raw
```

En SCL sería:

Listing 3.2: Lectura de valor analógico en SCL

```
1 #temp_raw := IW64;
```

3.5. Ejercicios del capítulo 3

1. Investigue las diferencias entre una señal de corriente 4-20 mA y una de tensión 0-10 V. ¿En qué casos es preferible usar corriente?

2. Configure un módulo analógico SM 1231 AI 4x13 bits en TIA Portal para un sensor 4-20 mA en el canal 0. Anote la dirección asignada.

3. Escriba un programa en LADDER que lea el valor analógico del canal 0 y lo copie a un DB global.

4. ¿Qué significa que el valor máximo normalizado sea 27648 y no 65535? Investigue la razón.

4 Escalado y Normalización de Señales

4.1. Necesidad de escalar: de valores crudos a magnitudes físicas

El valor leído (por ejemplo, 0 a 27648) no tiene significado físico directo. Para obtener la temperatura real (en °C), presión (en bar), etc., necesitamos **escalar** ese valor. La relación suele ser lineal:

$$\text{Valor físico} = \text{Valor mínimo del rango físico} + \left(\frac{\text{Valor crudo} - \text{Crudo mínimo}}{\text{Crudo máximo} - \text{Crudo mínimo}} \right) \times (\text{Rango físico})$$

Para un sensor 4-20 mA que mide temperatura de 0 a 100 °C, el valor crudo 0 corresponde a 0 °C y 27648 a 100 °C. Pero si el sensor tiene offset, por ejemplo 4 mA = 0 °C, entonces el valor crudo mínimo no es 0 sino el correspondiente a 4 mA. En módulos SIEMENS, 4 mA suele corresponder a 5530 (aproximadamente, porque 0-20 mA = 0-27648, entonces 4 mA = 27648 * 4/20 = 5530). Esto es importante: el crudo mínimo es 5530 y el máximo 27648.

4.2. Instrucciones de escalado: NORM_X y SCALE_X en TIA Portal

TIA Portal ofrece instrucciones específicas para normalizar y escalar:

- **NORM_X**: Normaliza un valor entero a un valor real entre 0.0 y 1.0. Se le proporciona el valor, el mínimo y el máximo.
- **SCALE_X**: Escala un valor normalizado (0.0 a 1.0) a un rango físico (mínimo y máximo) y lo convierte al tipo deseado (INT, REAL, etc.).

Ambas suelen usarse juntas:

Listing 4.1: Escalado con NORM_X y SCALE_X en LADDER

```
1   // Normalizar el valor crudo a un real entre 0 y 1
2   NORM_X INT TO REAL
3   EN
4   VALUE := IW64
5   MIN := 5530   // valor para 4 mA
6   MAX := 27648  // valor para 20 mA
7   OUT := #norm
8
9   // Escalar a temperatura real (0 a 100  C )
10  SCALE_X REAL TO REAL
11  EN
12  VALUE := #norm
```

```
13      MIN := 0.0
14      MAX := 100.0
15      OUT := #temp_real
```

En SCL es más directo:

Listing 4.2: Escalado manual en SCL

```
1 #temp_real := INT_TO_REAL(IW64 - 5530) / INT_TO_REAL(27648 - 5530)
     * 100.0;
```

4.3. Cálculo manual de escalado lineal: fórmula y ejemplos

Si no se desea usar las instrucciones, se puede calcular directamente:

$$\text{Valor físico} = \frac{(\text{Valor crudo} - \text{Crudo mínimo}) \times (\text{Rango físico})}{\text{Crudo máximo} - \text{Crudo mínimo}} + \text{Físico mínimo}$$

4.4. Ejercicios del capítulo 4

1. Un sensor de presión 4-20 mA mide de 0 a 10 bar. El valor crudo leído es 12000. Calcule la presión real (suponiendo crudo mínimo 5530, máximo 27648).

2. Implemente en LADDER el escalado para el sensor de presión anterior, usando NORM_X y SCALE_X.

3. ¿Qué ocurre si el valor crudo está fuera del rango (por ejemplo, menor que 5530)? ¿Cómo se puede manejar esta situación?

4. Repita el ejercicio 1 usando SCL con una fórmula directa.

5 Comparadores y Operaciones Aritméticas

5.1. Comparadores: CMP ==, <>, >, <, >=, <= en LADDER

Para tomar decisiones basadas en valores analógicos, se utilizan comparadores. En LADDER, los comparadores se encuentran en la paleta como "CMP". Por ejemplo, para comparar si la temperatura es mayor a 50 °C:

Listing 5.1: Comparador en LADDER

```
    CMP > REAL
    #temp_real IN1
    50.0 IN2
    OUT -> #alta_temperatura
```

También se pueden usar contactos especiales como "==" en algunos PLC, pero lo más común es usar la instrucción CMP.

En SCL, se usan operadores relacionales normales:

Listing 5.2: Comparador en SCL

```
IF #temp_real > 50.0 THEN
    #alta_temperatura := TRUE;
ELSE
    #alta_temperatura := FALSE;
END_IF;
```

5.2. Operaciones aritméticas: suma, resta, multiplicación, división (ADD, SUB, MUL, DIV)

En LADDER, las operaciones aritméticas se realizan con bloques como ADD, SUB, MUL, DIV. Por ejemplo, para sumar dos valores reales:

Listing 5.3: Suma en LADDER

```
    ADD REAL
    EN
    IN1 := #valor1
    IN2 := #valor2
    OUT => #resultado
```

En SCL es más natural:

Listing 5.4: Suma en SCL

```
#resultado := #valor1 + #valor2;
```

5.3. Uso combinado para control por umbrales y lazos sencillos

Podemos combinar comparadores y operaciones aritméticas para implementar lógica de control como histéresis. Por ejemplo, control de temperatura con histéresis: encender un ventilador cuando la temperatura supere 50 °C y apagarlo cuando baje de 45 °C.

Listing 5.5: Control con histéresis en SCL

```
IF NOT #ventilador AND #temp_real > 50.0 THEN
    #ventilador := TRUE;
ELSIF #ventilador AND #temp_real < 45.0 THEN
    #ventilador := FALSE;
END_IF;
```

5.4. Ejemplo: control de temperatura con histéresis

Implementaremos un sistema completo: un sensor de temperatura analógico, un ventilador digital y una alarma si la temperatura supera 60 °C.

Listing 5.6: Programa completo en SCL

```
// Lectura y escalado
#temp_raw := IW64;
#temp_real := INT_TO_REAL(#temp_raw - 5530) / INT_TO_REAL(27648 -
   5530) * 100.0;

// Hist resis
IF NOT #ventilador AND #temp_real > 50.0 THEN
    #ventilador := TRUE;
ELSIF #ventilador AND #temp_real < 45.0 THEN
    #ventilador := FALSE;
END_IF;

// Alarma
IF #temp_real > 60.0 THEN
    #alarma := TRUE;
ELSE
    #alarma := FALSE;
END_IF;

// Salidas
Q0.0 := #ventilador;
Q0.1 := #alarma;
```

5.5. Ejercicios del capítulo 5

1. Diseñe un control de nivel con histéresis para un tanque: la bomba de llenado se activa cuando el nivel baja de 2 metros y se desactiva cuando supera 8 metros (rango total 0-10 m). Use un sensor analógico.

2. Implemente en LADDER la comparación de dos valores analógicos y active una salida si el primero es mayor que el segundo.

3. Escriba un programa que calcule el promedio de 4 lecturas analógicas y lo almacene.

4. ¿Qué precauciones deben tomarse al dividir números enteros? (pista: división entera vs. real)

6 Control PID Básico (Introducción)

6.1. Concepto de lazo cerrado: setpoint, variable de proceso, salida de control

El control PID (Proporcional-Integral-Derivativo) es la técnica más extendida para controlar variables continuas en lazo cerrado. Sus componentes:

- **Setpoint (SP)**: Valor deseado de la variable (consigna).
- **Variable de proceso (PV)**: Valor medido actual.
- **Error**: Diferencia SP - PV.
- **Salida de control (Output)**: Señal que actúa sobre el actuador (ej. velocidad de bomba, apertura de válvula).

El PID calcula la salida como:

$$\text{Output} = K_p \cdot e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

6.2. El bloque PID Compact en TIA Portal: configuración básica

TIA Portal dispone del bloque `PID_Compact` (para S7-1200/1500). Para usarlo:

1. Agregar un FB "PID_Compact" desde la librería.
2. Configurar los parámetros básicos en el bloque de datos de instancia:
   - Setpoint: variable donde se escribe la consigna.
   - Input: variable de proceso (real).
   - Output: variable de salida (real) que luego se puede escalar a 0-27648 para una salida analógica.
3. Ajustar los parámetros PID (ganancia, tiempo integral, tiempo derivativo). Inicialmente se puede usar autosintonía (pretuning).

6.3. Ajuste de parámetros (P, I, D) de forma empírica

Para sintonizar un PID manualmente:

- **P** (proporcional): Aumentar hasta que el sistema oscile. Luego reducir a la mitad aproximadamente.

- **I** (integral): Disminuir el tiempo integral (aumentar acción integral) para eliminar el error en estado estacionario, pero con cuidado de no causar oscilaciones.
- **D** (derivativo): Ayuda a anticipar cambios, pero es sensible al ruido. A menudo se deja en cero inicialmente.

El bloque PID_Compact incluye una función de autosintonía que puede facilitar el ajuste.

6.4. Ejemplo: control de nivel o temperatura con simulación

Supongamos un tanque con bomba de velocidad variable (salida analógica) y sensor de nivel analógico. Queremos mantener el nivel en 5 m (setpoint).

Listing 6.1: Configuración básica de PID en SCL

```
// Llamada al PID_Compact (suponiendo que ya se ha creado la
   instancia)
#PID_Instance(Setpoint := #nivel_sp,
              Input := #nivel_pv,
              Output => #bomba_speed_raw);

// Escalar la salida del PID (0-100%) a valor anal gico (0-27648)
#bomba_analog := REAL_TO_INT(#bomba_speed_raw * 276.48);
QW64 := #bomba_analog;   // Salida anal gica a bomba
```

6.5. Ejercicios del capítulo 6

1. Explique la diferencia entre un control todo-nada (con histéresis) y un control PID. ¿Cuándo es preferible cada uno?
2. Configure un PID_Compact en TIA Portal para controlar la temperatura de un horno. Indique los parámetros necesarios.
3. Simule un lazo de control de nivel con Factory I/O y un PID. Ajuste los parámetros hasta obtener una respuesta estable.
4. ¿Qué es el “windup” integral y cómo se evita?

Parte III

Algoritmos Complejos y Estructuras de Programación

7 Máquinas de Estados (Step-by-Step)

7.1. Concepto de máquina de estados: diagrama de flujo y transiciones

Una máquina de estados es un modelo de comportamiento donde el sistema se encuentra en un estado a la vez, y puede cambiar a otro estado cuando se cumplen ciertas condiciones (transiciones). Es ideal para procesos secuenciales (ej. una línea de envasado, un brazo robótico).

Los elementos clave:

- **Estados**: Situaciones discretas (ej. reposo, avanzando, girando, etc.).
- **Transiciones**: Condiciones que provocan el cambio de estado.
- **Acciones**: Actividades que se realizan en cada estado.

7.2. Implementación con registros de paso (INT) y comparadores

Una forma sencilla de implementar una máquina de estados en PLC es usar un registro de paso (variable INT) y una estructura CASE (en SCL) o comparadores en LADDER.

En SCL:

Listing 7.1: Máquina de estados simple en SCL

```
CASE #estado OF
    0: // Reposo
        Q0.0 := FALSE;
        Q0.1 := FALSE;
        IF #start THEN
            #estado := 10;
        END_IF;
    10: // Avanzar cilindro
        Q0.0 := TRUE;   // V lvula de avance
        IF #fin_carrera_avance THEN
            #estado := 20;
        END_IF;
    20: // Retroceder
        Q0.0 := FALSE;
        Q0.1 := TRUE;   // V lvula de retroceso
        IF #fin_carrera_retroceso THEN
            #estado := 0;
        END_IF;
END_CASE;
```

7.3. Uso de bloques FB para encapsular la lógica

Para máquinas más complejas, conviene encapsular la máquina de estados en un FB. Así se pueden reutilizar y mantener más fácilmente. El FB tendrá una variable interna para el estado y métodos para las transiciones.

7.4. Ejemplo: secuencia de un brazo robótico con múltiples pasos

Diseñaremos un brazo con 3 movimientos (avanzar, girar, sujetar) en secuencia, con parada de emergencia.

Listing 7.2: FB de máquina de estados para brazo

```
FUNCTION_BLOCK FB_Brazo
VAR
    estado : INT := 0;
END_VAR

CASE estado OF
    0: // Reposo
        Q_avance := FALSE;
        Q_giro := FALSE;
        Q_sujetar := FALSE;
        luz_activo := FALSE;
        IF start THEN
            estado := 10;
            luz_activo := TRUE;
        END_IF;
    10: // Avanzar
        Q_avance := TRUE;
        IF sensor_avance THEN
            estado := 20;
        END_IF;
    20: // Girar
        Q_avance := FALSE;
        Q_giro := TRUE;
        IF sensor_giro THEN
            estado := 30;
        END_IF;
    30: // Sujetar
        Q_giro := FALSE;
        Q_sujetar := TRUE;
        temporizador(IN:=TRUE, PT:=T#2s);
        IF temporizador.Q THEN
            estado := 40;
        END_IF;
    40: // Retorno (simplificado)
        Q_sujetar := FALSE;
        // ... movimientos de retorno ...
```

```
        IF fin_ciclo THEN
            estado := 0;
            luz_activo := FALSE;
        END_IF;
END_CASE;

// Parada de emergencia
IF emergencia THEN
    estado := 0;
    luz_activo := FALSE;
    Q_avance := FALSE;
    Q_giro := FALSE;
    Q_sujetar := FALSE;
END_IF;
```

7.5. Ejercicios del capítulo 7

1. Dibuje el diagrama de estados para un semáforo peatonal (rojo/verde con pulsador).
2. Implemente en SCL la máquina de estados del semáforo.
3. ¿Por qué es importante incluir un estado de reposo o inicial?
4. Modifique el ejemplo del brazo para incluir un reset que vuelva al estado inicial en cualquier momento.

8 Manejo de Arrays y Estructuras de Datos

8.1. Declaración de arrays en DB y su uso

Un array es una colección de elementos del mismo tipo, accesibles por índice. En TIA Portal, se declaran en DB o en variables temporales. Ejemplo en un DB global:

Listing 8.1: Declaración de array en DB

```
    miArray : ARRAY[1..10] OF INT;
```

Para acceder: `miArray[3]`.

8.2. Acceso a elementos mediante índices

Podemos usar índices variables. Por ejemplo, para sumar todos los elementos:

Listing 8.2: Suma de array en SCL

```
#suma := 0;
FOR #i := 1 TO 10 DO
    #suma := #suma + #miArray[#i];
END_FOR;
```

8.3. Estructuras (STRUCT) y UDT (User Defined Types)

Una estructura agrupa variables de diferentes tipos. Se puede definir un tipo de datos propio (UDT) para reutilizarlo. Ejemplo:

Listing 8.3: Definición de UDT en TIA Portal

```
TYPE MotorData :
STRUCT
    nombre : STRING[20];
    velocidad_actual : INT;
    temperatura : REAL;
    en_marcha : BOOL;
END_STRUCT;
END_TYPE
```

Luego se puede usar en DB: `motor1 : MotorData;`

8.4. Ejemplo: almacenar recetas con parámetros

Podemos crear un array de estructuras para guardar varias recetas:

Listing 8.4: Array de recetas

```
TYPE Receta :
STRUCT
    temperatura_sp : REAL;
    tiempo_coccion : TIME;
    agitacion : BOOL;
END_STRUCT;
END_TYPE

VAR_GLOBAL
    recetas : ARRAY[1..5] OF Receta;
END_VAR
```

8.5. Ejercicios del capítulo 8

1. Declare un array de 10 reales en un DB y escriba una función que calcule el promedio.
2. Defina una UDT para un sensor que incluya: tipo (STRING), valor$_a ctual(REAL), unidad(STRIN$

2. Cree un DB con 3 instancias de esa UDT y asígneles valores.
3. ¿Qué ventajas tiene usar UDT frente a variables sueltas?

9 Programación con SCL (Structured Control Language)

9.1. Introducción a SCL: ventajas frente a LADDER

SCL es un lenguaje de alto nivel (similar a Pascal) que permite:

- Algoritmos complejos de forma compacta.
- Estructuras de control como IF, CASE, FOR, WHILE.
- Mejor legibilidad para tareas matemáticas o de manejo de datos.
- Reutilización de código mediante funciones y bloques.

9.2. Sintaxis básica: variables, asignaciones, condicionales (IF, CASE), bucles (FOR, WHILE)

- **Asignación**: `variable := expresion;`
- **IF**:

```
IF condicion THEN
    ...
ELSIF otra THEN
    ...
ELSE
    ...
END_IF;
```

- **CASE**:

```
CASE variable OF
    1: ...
    2: ...
    ELSE ...
END_CASE;
```

- **FOR**:

```
FOR i := 1 TO 10 DO
    ...
END_FOR;
```

- **WHILE**:

```
WHILE condicion DO
    ...
END_WHILE;
```

9.3. Implementación de funciones matemáticas y lógicas

En SCL podemos usar funciones matemáticas estándar: `SQRT, SIN, COS, ABS, etc.`. Ejemplo:

```
#raiz := SQRT(#valor);
```

9.4. Ejemplo: convertir un algoritmo LADDER a SCL

Supongamos un circuito de enclavamiento en LADDER. En SCL sería:

```
IF (marcha AND NOT paro) OR (motor AND NOT paro) THEN
    motor := TRUE;
ELSE
    motor := FALSE;
END_IF;
```

9.5. Ejercicios del capítulo 9

1. Re-escriba el programa de histéresis del capítulo 5 en SCL.
2. Implemente un bucle FOR que calcule el factorial de un número (usando DINT).
3. ¿Cómo se declara una función (FC) en SCL? Dé un ejemplo.
4. Convierta a SCL el ejemplo de máquina de estados del capítulo 7.

Parte IV

Simulación Avanzada con Factory I/O y PLCSIM

10 Integración Profunda entre TIA Portal, PLCSIM y Factory I/O

10.1. Configuración de la comunicación: driver Siemens PLCSIM

Factory I/O puede comunicarse con PLCSIM mediante el driver "Siemens PLCSIM". Pasos:

1. En TIA Portal, compilar y cargar el programa en PLCSIM (iniciar simulación).
2. En Factory I/O, abrir el escenario deseado.
3. Ir a `File >Drivers` y seleccionar `Siemens PLCSIM`.
4. Asegurarse de que PLCSIM esté en ejecución y el PLC en RUN.

10.2. Asignación de direcciones de E/S en Factory I/O

Cada elemento del escenario (botones, sensores, luces, motores) tiene una dirección asignable. En Factory I/O, se puede configurar qué entrada/salida del PLC controla cada elemento. Por ejemplo, asignar el botón START a I0.0, la luz piloto a Q0.0, etc.

10.3. Simulación de escenas complejas: sensores analógicos y actuadores

Factory I/O incluye sensores analógicos (ultrasonidos, barreras, etc.) y actuadores analógicos (variadores de velocidad, posicionadores). Para usarlos, hay que configurar la dirección analógica correspondiente (por ejemplo, IW64 para un sensor de distancia) y luego escalar en el PLC.

10.4. Depuración: uso de tablas de vigilancia y forzado

Mientras se simula, se pueden usar tablas de vigilancia en TIA Portal para monitorear y forzar variables, y ver cómo reacciona el escenario.

10.5. Ejercicios del capítulo 10

1. Configure la comunicación entre TIA Portal, PLCSIM y Factory I/O para el escenario "Basic - Start/Stop Conveyor". Verifique que el motor arranca y para.

2. Agregue un sensor inductivo en la escena y asígnelo a una entrada. Programe que cuando el sensor detecte una pieza, se detenga la cinta.

3. ¿Qué ventajas ofrece la simulación 3D frente a la simulación con PLCSIM solo?

11 Escenarios Analógicos en Factory I/O

11.1. Sensores analógicos en Factory I/O: distancia, nivel, presión

Factory I/O dispone de sensores con salida analógica:

- Sensor de distancia (ultrasonidos): entrega un valor proporcional a la distancia de un objeto.
- Sensor de nivel: en tanques, mide el nivel de líquido.
- Sensor de presión: mide presión en circuitos neumáticos o hidráulicos.

Estos sensores se configuran en una dirección de entrada analógica (ej. IW64) y el rango de valores (0-27648) corresponde al rango físico configurable en el escenario.

11.2. Lectura de valores analógicos en el PLC y escalado

Se procede igual que con señales reales: leer la palabra de entrada y escalar a unidades físicas según el sensor. Por ejemplo, un sensor de nivel configurado de 0 a 10 m.

11.3. Actuadores analógicos: variadores de velocidad, posicionadores

También hay actuadores analógicos:

- Variador de velocidad: se controla con una salida analógica (QW) que determina la velocidad (0-27648 = 0-100 %).
- Posicionador: para cilindros con control proporcional.

11.4. Ejemplo: control de nivel de un tanque con bomba de velocidad variable

Escenario: tanque con sensor de nivel (IW64) y bomba con variador (QW64). Queremos mantener nivel en 5 m.

Programa en SCL:

```
#nivel_raw := IW64;
#nivel_real := INT_TO_REAL(#nivel_raw) / 27648.0 * 10.0;  // 0-10 m
#error := 5.0 - #nivel_real;

```

```
// P simple
#velocidad := #Kp * #error;
IF #velocidad < 0.0 THEN #velocidad := 0.0; END_IF;
IF #velocidad > 100.0 THEN #velocidad := 100.0; END_IF;

QW64 := REAL_TO_INT(#velocidad * 276.48);
```

11.5. Ejercicios del capítulo 11

1. En Factory I/O, seleccione el escenario “Tank”. Configure el sensor de nivel y la bomba con variador. Programe un control proporcional para mantener el nivel en 50 %.

2. Agregue histéresis al control para evitar oscilaciones.

3. Implemente un control PID completo y compare la respuesta.

12 Simulación de Procesos Complejos

12.1. Escenario "Sorting by Height.º similar: detección y clasificación

El escenario “Sorting by Height” (o “Sorting by Size”) de Factory I/O presenta una cinta con piezas de diferentes alturas, un sensor de altura (analógico) y expulsores neumáticos. Es ideal para practicar clasificación.

12.2. Programación con múltiples sensores y actuadores

Se deben leer sensores (altura, presencia) y controlar actuadores (cinta, expulsores). Usaremos máquina de estados y arrays para llevar el conteo.

12.3. Uso de temporizadores, contadores y comparadores

Para clasificar, se puede usar un contador para cada categoría, y temporizadores para controlar el tiempo de expulsión.

12.4. Ejemplo: clasificación de piezas por altura con expulsor

Supongamos que queremos clasificar piezas en dos categorías: bajas (<5 cm) y altas (>=5 cm). Al llegar al sensor de altura, se determina la categoría y cuando la pieza llegue al expulsor correspondiente, se activa.

Programa:

```
// Detecci n de pieza en sensor de entrada
IF #sensor_entrada AND NOT #prev_entrada THEN
    #altura := IW64;  // leer altura
    IF #altura < 5.0 * 2764.8 THEN  // 5 cm escalado a crudo
        #categoria := 0;  // baja
    ELSE
        #categoria := 1;  // alta
    END_IF;
    // Guardar en cola (array FIFO)
    // ...
END_IF;
#prev_entrada := #sensor_entrada;

```

```
// Cuando la pieza llega al expulsor, activar seg n categor a
IF #sensor_expulsor_baja AND #categoria_actual = 0 THEN
    Q_expulsor_baja := TRUE;
    TON_expulsor(IN:=TRUE, PT:=T#500ms);
    IF TON_expulsor.Q THEN
        Q_expulsor_baja := FALSE;
    END_IF;
END_IF;
```

12.5. Ejercicios del capítulo 12

1. Implemente en Factory I/O y TIA Portal un clasificador de 3 categorías.
2. Añada un contador de piezas totales y por categoría, mostrados en una HMI virtual.
3. Simule fallos (atascos) y programe una alarma.

13 Depuración y Puesta a Punto en Simulación

13.1. Herramientas de diagnóstico en TIA Portal

Repaso de herramientas: referencias cruzadas, lista de errores, tablas de vigilancia, gráficas de tendencias (para señales analógicas).

13.2. Simulación de fallos en Factory I/O

Factory I/O permite inyectar fallos: sensores atascados, objetos que no se detectan, etc. Útil para probar la robustez del programa.

13.3. Estrategias para probar el programa paso a paso

- Probar cada sensor por separado forzando entradas.
- Usar breakpoints en SCL (si el entorno lo permite).
- Probar la máquina de estados con todas las transiciones.
- Verificar los límites de escalado.

13.4. Ejercicios del capítulo 13

1. Tome un programa previo y use la tabla de vigilancia para forzar entradas y verificar salidas.
2. Simule un fallo de sensor en Factory I/O (por ejemplo, sensor de altura siempre en cero) y observe cómo reacciona el programa. Mejore el programa para detectar esta condición.
3. Realice un test completo de un sistema de clasificación, documentando los casos de prueba.

Parte V

Proyectos Integradores

14 Proyecto Integrador Nivel 1 - Control de una Prensa con Señal Analógica

14.1. Consigna

Diseñar un sistema para una prensa neumática con control de presión. El sistema cuenta con: - Un cilindro de doble efecto con válvula 5/2. - Un sensor de presión analógico en el circuito neumático (0-10 bar, 4-20 mA). - Pulsadores de marcha (I0.0) y parada (I0.1). - Parada de emergencia (I0.2, NC). - Sensor de posición retraído (I0.3) y extendido (I0.4). - Salidas: Q0.0 (avance), Q0.1 (retroceso), Q0.2 (luz verde activo).

14.2. Requisitos funcionales

1. Al presionar marcha, si el cilindro está retraído y no hay emergencia, se inicia el ciclo. 2. El cilindro avanza hasta llegar al final de carrera extendido. 3. Una vez extendido, debe alcanzar una presión de 8 bar (medida por el sensor analógico) y mantenerla durante 3 segundos. 4. Luego retrocede hasta el final de carrera retraído. 5. El ciclo se detiene y queda listo para una nueva marcha. 6. Si la presión supera 9 bar en cualquier momento, debe abortar el ciclo y retroceder inmediatamente, activando una alarma (luz roja Q0.3). 7. La parada de emergencia detiene todo inmediatamente y requiere un reset (pulsador de reset o nueva marcha) para reanudar.

14.3. Guía de desarrollo

- Escalar la señal analógica de presión. - Implementar máquina de estados. - Incluir enclavamientos y condiciones de emergencia.

15 Proyecto Integrador Nivel 2 - Línea de Empaquetado con Clasificación

15.1. Consigna

Diseñar una línea de empaquetado que clasifica piezas por altura y las cuenta en cajas. Elementos: - Cinta transportadora principal con motor (Q0.0). - Sensor de entrada (I0.0) para detectar piezas. - Sensor de altura analógico (IW64) que mide altura de 0 a 10 cm. - Dos expulsores neumáticos (Q0.1 y Q0.2) para desviar piezas a dos bandas laterales. - Sensores de final de carrera de los expulsores (I0.1, I0.2) para confirmar movimiento. - Contadores: por cada categoría, mostrar en una HMI virtual (o en variables).

15.2. Requisitos funcionales

1. Clasificar piezas en tres categorías: pequeñas (<3 cm), medianas (3-7 cm), grandes (>7 cm). 2. Las pequeñas se expulsan con el primer expulsor, las medianas con el segundo, las grandes siguen por la cinta principal (fin de línea). 3. Llevar un conteo de piezas por categoría. 4. Si una categoría llega a 10 piezas, detener la cinta principal y encender una luz indicadora de "lote completo". Reanudar con un botón de reanudar. 5. Incluir parada de emergencia.

16 Proyecto Final - Estación de Mezclado con Control PID

16.1. Consigna

Diseñar una estación de mezclado con los siguientes componentes: - Tanque con agitador (motor Q0.0). - Válvula de llenado (Q0.1) y válvula de vaciado (Q0.2). - Sensor de nivel analógico (IW64, 0-10 m). - Sensor de temperatura analógico (IW66, 0-100 °C). - Bomba de recirculación con velocidad variable (salida analógica QW64) para control de temperatura mediante intercambiador. - HMI virtual para ingresar setpoint de nivel y temperatura, y visualizar variables.

16.2. Requisitos funcionales

1. Ciclo de llenado: abrir válvula de llenado hasta alcanzar el setpoint de nivel (por ejemplo, 5 m). Cerrar válvula. 2. Activar agitador y control de temperatura: mediante PID, regular la velocidad de la bomba para mantener la temperatura en el setpoint (ej. 50 °C). Usar el sensor de temperatura como PV. 3. Una vez transcurrido un tiempo de mezclado (parametrizable), abrir válvula de vaciado hasta que el nivel sea cero. 4. El ciclo puede repetirse automáticamente o esperar una nueva orden. 5. Alarmas: temperatura >90 °C, nivel >9 m, fallo de sensor (señal fuera de rango).

16.3. Guía de desarrollo

- Crear UDT para parámetros de receta (setpoint nivel, setpoint temp, tiempo mezclado). - Implementar máquina de estados. - Configurar PID para temperatura. - Usar arrays o estructuras para manejo de alarmas.

A Anexo A: Tabla de Tipos de Datos en TIA Portal

Tipo	**Descripción**	**Rango**
BOOL	Booleano	TRUE/FALSE
BYTE	Entero sin signo 8 bits	0..255
WORD	Entero sin signo 16 bits	0..65535
DWORD	Entero sin signo 32 bits	0..4294967295
INT	Entero con signo 16 bits	-32768..32767
DINT	Entero con signo 32 bits	-2147483648..2147483647
REAL	Coma flotante 32 bits	±1.18e-38 a ±3.40e38
S5TIME	Duración en formato S5	0..9990 ms, etc.
TIME	Duración en ms	-24d20h31m23s647ms a +24d20h31m23s647ms
STRING	Cadena de caracteres	hasta 254 caracteres

B Anexo B: Resumen de Instrucciones de Escalado y Comparación

- **NORM_X**: Normaliza un valor entero a real entre 0.0 y 1.0.
- **SCALE_X**: Convierte un valor normalizado a un rango físico.
- **CMP**: Compara dos valores (EQ, NE, GT, LT, GE, LE).
- **LIMIT**: Limita un valor entre un mínimo y un máximo.
- **SEL**: Selecciona entre dos valores según una condición booleana.
- **MAX/MIN**: Devuelve el máximo/mínimo de dos valores.

C Anexo C: Guía Rápida de SCL

```
// Declaraci n de variables en FB/FC
VAR_INPUT
    in1 : INT;
END_VAR
VAR_OUTPUT
    out1 : REAL;
END_VAR
VAR_TEMP
    temp : BOOL;
END_VAR

// Asignaci n
out1 := INT_TO_REAL(in1) * 2.5;

// Condicional
IF temp THEN
    out1 := 0.0;
END_IF;

// Bucle
FOR i := 1 TO 10 DO
    array[i] := i * 2;
END_FOR;
```

D Anexo D: Configuración de Escenarios en Factory I/O para Simulación Analógica

Pasos para usar un sensor analógico: 1. En Factory I/O, seleccionar un sensor con salida analógica (ej. Ultrasonic Sensor). 2. En sus propiedades, asignar la dirección de entrada (ej. IW64). 3. Configurar el rango físico del sensor (ej. 0-100 cm). 4. En el PLC, leer IW64 y escalar según el rango configurado.

E Anexo E: Glosario de Términos Avanzados

SCL Structured Control Language.

PID Proporcional-Integral-Derivativo.

SP Setpoint.

PV Process Variable.

UDT User Defined Type.

FIFO First In First Out (cola).

Windup Acumulación excesiva del término integral.

Histeresis Diferencia entre umbral de activación y desactivación.