CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA

Video 07 — Parte 4A: Sistemas de Entrada/Salida, Interfases, Direccionamiento y Adaptadores para Periféricos — Incompatibilidades Físicas, Registros Data/Status/Control, MMIO vs. PMIO, Handshaking, PIA 8255 y MCP23017

Parte del Curso: Parte 4A — Sistemas de Entrada/Salida, Interfases, Direccionamiento y Adaptadores
Arquitectura / Controlador Estudiado: Registros Data/Status/Control, Direccionamiento MMIO vs. PMIO, Handshaking Strobe/ACK, PIA 8255 y Expansor MCP23017

```
Decodificación MMIO: CS# = 0 cuando ADDR[31:12] == BASE_PERIPH | MCP23017: IODIR (0=Out, 1=In), GPPU, GPIO, OLAT
```

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Cinco funciones esenciales de toda interfase de E/S: adaptación de niveles de tensión/corriente, sincronización de velocidades (CPU en ns vs. planta en ms), conversión de formato de datos, retención/aislamiento de bus y aislamiento galvánico.
- Arquitectura interna de una interfase (Decodificador de Direcciones, Registro de Datos, Registro de Estado y Registro de Control) y comparación entre E/S Mapeada en Memoria (MMIO) y E/S Aislada por Puertos (PMIO IN/OUT).
- Sincronización por Sondeo (Polling), Interrupciones (IRQ) y Acceso Directo a Memoria (DMA); necesidad de Latches tipo D en salidas, buffers Tri-State (High-Z) en entradas y protocolo de Handshaking paralelo (STROBE y ACK/BUSY).
- Del Adaptador Periférico Programable clásico (PIA Intel 8255 con Puertos A, B, C en Modos 0, 1 y 2 y VIA 6522) a los expansores modernos por I2C/SPI (MCP23017 de 16 bits con interrupción) y conversores externos ADC Sigma-Delta de 16/24 bits.

2. Parámetros Nominales de Configuración del Controlador (Video 07)

Parámetro del Controlador / Interfase	Símbolo	Valor Nominal	Función en el Lazo de Control / Automatización
Ganancia Proporcional	`Kp (PB = 100/Kp)`	2.5	Esfuerzo inmediato proporcional al error actual e(t)
Tiempo Integral	`Ti [s]`	0.4 s	Elimina el error estacionario (Offset) acumulando la integral del error
Tiempo Derivativo y Filtro Divisor	`Td [s] / N`	Td = 0.05 s, N = 10	Anticipa la tendencia sobre -PV y atenúa el ruido de alta frecuencia
Periodo de Muestreo en Tiempo Real	`Ts [ms]`	2 ms	Intervalo determinista de conversión ADC, cálculo discreto y actualización PWM
Límites de Saturación Anti-Windup	`[u_min, u_max]`	[0, 4095]	Cota física del actuador para integración condicional (Clamping)
Rango Escalado de Ingeniería (SCALE_X)	`[PV_min, PV_max]`	[0, 24]	Conversión lineal desde cuentas crudas del ADC (0..1023 o 0..27648)

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: Necesidad Física y Funcional de una Interfase de Entrada/Salida

Resolución de Incompatibilidades de Tensión, Corriente, Velocidad Temporal, Formato de Datos y Aislamiento

- **Turno 1 (Masculina):** Al comenzar la Parte cuatro del curso abordamos un principio fundamental de la arquitectura de controladores: nunca es posible conectar los buses internos de una unidad central de procesamiento directamente a los sensores y actuadores de una máquina.
- **Turno 2 (Femenina):** Entre el mundo interno del silicio y la planta física siempre debe existir un bloque electrónico mediador llamado Interfase de Entrada y Salida, encargado de resolver cinco incompatibilidades físicas radicales.
- **Turno 3 (Masculina):** La primera es la incompatibilidad eléctrica de tensión y corriente: el núcleo del procesador opera con uno punto ocho o tres punto tres voltios y corrientes de microamperios, mientras que la planta industrial trabaja con señales de veinticuatro voltios en corriente continua o doscientos veinte voltios en corriente alterna y corrientes de varios amperios.
- **Turno 4 (Femenina):** La segunda es la incompatibilidad de velocidad y tiempo: la CPU ejecuta instrucciones en nanosegundos, mientras que un sensor mecánico o una electroválvula responde en milisegundos; a esto se suman la conversión de formato entre datos analógicos, serie y paralelo, y el aislamiento galvánico.

Diapositiva 02: Estructura Interna de una Interfase Periférica

Decodificador de Direcciones, Registros de Datos (Buffers/Latches), Registros de Estado y Registros de Control

- **Turno 1 (Masculina):** Si analizamos por dentro cualquier circuito de interfase conectado a los buses de direcciones, datos y control de un procesador, encontraremos siempre una estructura universal compuesta por cuatro bloques internos.
- **Turno 2 (Femenina):** El primer bloque es el Decodificador de Direcciones: un circuito lógico que vigila el bus de direcciones de la CPU y genera la señal de selección del chip únicamente cuando el procesador invoca la dirección asignada a ese periférico.
- **Turno 3 (Masculina):** El segundo bloque son los Registros de Datos: circuitos biestables que retienen temporalmente el byte o palabra que la CPU desea enviar hacia el exterior, o que capturan el dato proveniente del sensor hasta que la CPU pueda leerlo.
- **Turno 4 (Femenina):** El tercer y cuarto bloque son el Registro de Control, donde el programa escribe bits de configuración para elegir la dirección de los pines o la velocidad de transmisión, y el Registro de Estado, donde la CPU lee banderas que indican si el periférico está listo, ocupado o en condición de error.

Diapositiva 03: Direccionamiento de Entrada/Salida: Mapeado en Memoria (MMIO) vs. Aislado (PMIO)

Espacio Único de Direcciones con Instrucciones Estándar (ARM/RISC-V) frente a Espacio de Puertos Dedicado (IN/OUT)

- **Turno 1 (Masculina):** ¿Cómo distingue el procesador si una dirección binaria colocada en el bus se refiere a una celda de memoria RAM o a un registro de una interfase de entrada y salida? Existen dos filosofías históricas de direccionamiento: E S Aislada y E S Mapeada en Memoria.
- **Turno 2 (Femenina):** En la Entrada y Salida Aislada o Mapeada por Puertos, abreviada P M I O y utilizada en la arquitectura Intel equis ochenta y seis, los periféricos viven en un espacio de direcciones separado de la memoria, activado por una línea física de control adicional y usando instrucciones exclusivas llamadas IN y OUT.
- **Turno 3 (Masculina):** Por el contrario, en la Entrada y Salida Mapeada en Memoria, abreviada M M I O y adoptada por ARM Cortex, RISC-V y los microprocesadores Motorola clásicós, existe un único espacio global de direcciones donde conviven la Flash, la SRAM y todos los registros de las interfases.
- **Turno 4 (Femenina):** La gran ventaja de la arquitectura M M I O es su flexibilidad: el procesador no necesita instrucciones especiales de entrada y salida, sino que puede emplear todo su repertorio de punteros, cargas, almacenamientos y operaciones lógicas directamente sobre las direcciones de los periféricos.

Diapositiva 04: Esquemas de Sincronización y Transferencia: Polling, Interrupción y DMA

Sondeo Programado de Banderas de Estado, E/S Dirigida por Interrupciones y Acceso Directo a Memoria

- **Turno 1 (Masculina):** Dada la enorme diferencia de velocidad entre una CPU de cientos de megahercios y un periférico externo lento, necesitamos un mecanismo de sincronización que gobierne cuándo se transfiere cada dato a través de la interfase.
- **Turno 2 (Femenina):** El primer esquema es la Entrada y Salida Programada por Sondeo o Polling: la CPU lee repetidamente en un bucle el bit de Listo dentro del registro de estado de la interfase; es sencillo de codificar, pero desperdicia el cien por ciento de la capacidad del procesador mientras espera.
- **Turno 3 (Masculina):** El segundo esquema es la Entrada y Salida Dirigida por Interrupciones: la CPU continúa ejecutando el lazo de control y el resto del programa, y solo cuando la interfase tiene un dato nuevo activa la línea física de interrupción para que la CPU lo recoja en pocos microsegundos.
- **Turno 4 (Femenina):** Y el tercer esquema, reservado para flujos masivos de datos como arreglos de conversores analógico-digitales o pantallas gráficas, es el Acceso Directo a Memoria o D M A, donde la interfase transfiere bloques enteros directamente hacia la RAM sin involucrar a la CPU.

Diapositiva 05: Requerimientos Eléctricos: Latching, Salidas Tri-Estado y Adaptación de Nivel

Retención Temporal con Flip-Flops D, Aislamiento de Bus en Alta Impedancia (High-Z), Fan-Out y Traslación 3.3V-5V

- **Turno 1 (Masculina):** Desde el punto de vista electrónico digital, conectar múltiples interfases a un mismo bus de datos compartido exige cumplir tres requerimientos eléctricos estrictos: retención por Latch, aislamiento Tri-Estado y adaptación de nivel.
- **Turno 2 (Femenina):** Cuando la CPU escribe un byte hacia un actuador externo, ese dato solo permanece presente en el bus de datos durante unos pocos nanosegundos; por ello toda interfase de salida debe incluir un registro Latch de flip-flops tipo de que capture y mantenga firme el voltaje en los pines de salida.
- **Turno 3 (Masculina):** A la inversa, en las interfases de entrada no podemos conectar varios sensores directamente a las mismas líneas del bus porque entrarían en cortocircuito lógico: cada entrada debe pasar por un buffer de Tercer Estado o Tri-State que permanezca desconectado en Alta Impedancia mientras no sea seleccionado.
- **Turno 4 (Femenina):** Además, toda interfase debe respetar la capacidad de corriente o Fan-Out de las compuertas y realizar la traslación de niveles de tensión, empleando convertidores bidireccionales o buffers de la familia setenta y cuatro L V C cuando un microcontrolador de tres punto tres voltios dialoga con periféricos de cinco voltios.

Diapositiva 06: Protocolos de Handshaking (Apretón de Manos) por Hardware en Paralelo

Sincronización Asíncrona Segura mediante Líneas de Estrobo (STB), Buffer Lleno (IBF/OBF) y Reconocimiento (ACK)

- **Turno 1 (Masculina):** ¿Cómo garantizamos en una transferencia paralela rápida que el receptor lea el byte exactamente cuando todas las líneas de datos ya se han estabilizado, sin leer datos viejos ni perder bytes por sobrescritura? Mediante un protocolo de apretón de manos o Handshaking por hardware.
- **Turno 2 (Femenina):** El protocolo de Handshaking utiliza dos líneas de control dedicadas que acompañan al bus de datos: una línea de validación o Estrobo controlada por el emisor, y una línea de Ocupado o Reconocimiento controlada por el receptor.
- **Turno 3 (Masculina):** Veamos la secuencia exacta de cuatro pasos: primero, el emisor coloca el byte en las líneas de datos y espera unos nanosegundos; segundo, activa la línea de Datos Válidos o Strobe para avisar que el dato es estable.
- **Turno 4 (Femenina):** Tercero, el receptor detecta el pulso, captura el byte en su registro interno y levanta la línea de Ocupado o Acknowledge confirmando la recepción; y cuarto, el emisor retira el estrobo y espera a que el receptor baje su bandera antes de enviar el siguiente byte.

Diapositiva 07: Adaptador de Interfase Periférica Programable Clásico (PIA Intel 8255)

Arquitectura de Puertos A, B y C, Palabra de Control y Modos de Operación (Modo 0 Básico, Modo 1 Strobe y Modo 2)

- **Turno 1 (Masculina):** El circuito integrado que sentó las bases de todos los puertos de entrada y salida programables de la historia es el Adaptador de Interfase Periférica Programable, representado por el célebre chip Intel ochenta y dos cincuenta y cinco y su equivalente Motorola sesenta y ocho veintiuno.
- **Turno 2 (Femenina):** El integrado ochenta y dos cincuenta y cinco conecta al bus de datos de ocho bits del procesador y ofrece hacia el exterior veinticuatro pines de entrada y salida organizados en tres puertos de ocho bits: el Puerto A, el Puerto B, y el Puerto C divisible en dos mitades de cuatro bits.
- **Turno 3 (Masculina):** Escribiendo un único byte en su Registro de Palabra de Control, el programador configura el chip en uno de tres modos: en el Modo cero, los tres puertos operan como entradas o salidas digitales simples con retención Latch en las salidas.
- **Turno 4 (Femenina):** En el Modo uno, los puertos A y B trabajan como puertos de datos de ocho bits sincronizados por Handshaking automático, utilizando los pines del Puerto C como líneas físicas de Strobe, Buffer Lleno, Acknowledge y solicitud de interrupción hacia la CPU; mientras que el Modo dos convierte al Puerto A en un bus bidireccional completo.

Diapositiva 08: Adaptador de Interfase Versátil (VIA 6522) y Expansores Modernos (MCP23017)

Evolución de las PIA/VIA hacia Expansores de Puertos por I2C/SPI con Registros IODIR, GPIO, OLAT e Interrupciones

- **Turno 1 (Masculina):** A la PIA clásica le siguió el Adaptador de Interfase Versátil o V I A sesenta y cinco veintidós, que agregó dos temporizadores-contadores programables de dieciséis bits y un registro de desplazamiento para conversión entre paralelo y serie.
- **Turno 2 (Femenina):** En la ingeniería mecatrónica actual, la arquitectura interna de aquellas PIA y V I A vive dentro de los propios puertos G P I O de cada microcontrolador, y cuando necesitamos añadir decenas de pines adicionales empleamos expansores modernos como el M C P veintitrés cero diecisiete.
- **Turno 3 (Masculina):** El integrado M C P veintitrés cero diecisiete se conecta utilizando únicamente dos hilos del bus I dos C —o cuatro hilos SPI en su versión veintitrés ese diecisiete— y proporciona dieciséis pines de entrada y salida divididos en Puerto A y Puerto B, permitiendo conectar hasta ocho chips en el mismo bus para sumar ciento veintiocho pines.
- **Turno 4 (Femenina):** En su interior encontramos exactamente la misma estructura de registros: el registro I O D I R configura qué pines son entradas o salidas, el registro G P P U activa resistencias Pull-Up internas, el registro O L A T retiene las salidas, y los pines físicos de interrupción avisan al microcontrolador en cuanto cambia cualquier sensor.

Diapositiva 09: Interfases de Conversión Analógica-Digital (ADC) y Digital-Analógica (DAC) Externas

Conversores SAR de 16 Bits y Sigma-Delta de 24 Bits (ADS1115/ADS1256), Amplificador PGA y Referencias de Precisión

- **Turno 1 (Masculina):** Además de expandir entradas y salidas digitales, en instrumentación mecatrónica de precisión a menudo debemos acoplar interfases externas de conversión analógica-digital y digital-analógica superiores a las integradas en el microcontrolador.
- **Turno 2 (Femenina):** Los conversores ADC integrados en un MCU comparten el mismo sustrato de silicio ruidoso de los relojes digitales y suelen usar la propia alimentación de tres punto tres voltios como referencia; por ello para medir microvoltios de celdas de carga, galgas extensiométricas o termopares usamos conversores externos.
- **Turno 3 (Masculina):** Para lazos dinámicos rápidos elegimos conversores externos de Aproximaciones Sucesivas o S A R de dieciséis bits por bus SPI; mientras que para pesaje e instrumentación de alta resolución elegimos conversores Sigma-Delta de dieciséis a veinticuatro bits como el A D S once quince o el H X setecientos once.
- **Turno 4 (Femenina):** Estas interfases integran en su chip un multiplexor diferencial, un Amplificador de Ganancia Programable para elevar señales débiles, una referencia de tensión de muy baja deriva térmica y filtros digitales que rechazan el ruido de cincuenta y sesenta hercios de la red eléctrica.

Diapositiva 10: Síntesis de Sistemas de Entrada/Salida y Adaptadores Periféricos

Diseño Integral del Mapa de Periféricos y Transición hacia Comunicaciones en Serie y Acoplamiento Industrial

- **Turno 1 (Masculina):** En esta Parte cuatro A hemos estudiado la teoría y el hardware de los sistemas de entrada y salida: desde la necesidad física de la interfase y sus registros internos de datos, estado y control, hasta el direccionamiento mapeado en memoria y el protocolo de Handshaking.
- **Turno 2 (Femenina):** Comprendimos la evolución desde los adaptadores históricos como la PIA ochenta y dos cincuenta y cinco y la V I A sesenta y cinco veintidós hasta los expansores modernos M C P veintitrés cero diecisiete y los conversores analógicos externos de alta precisión.
- **Turno 3 (Masculina):** Con esta base arquitectónica, necesitamos ahora dominar los buses en serie que transportan estos datos y los circuitos electrónicos de potencia y aislamiento que acoplan el controlador con cargas reales de veinticuatro voltios y corriente alterna.
- **Turno 4 (Femenina):** En el siguiente módulo, correspondiente a la Parte cuatro B, estudiaremos a fondo los protocolos UART, RS cuatrocientos ochenta y cinco, S P I, I dos C y bus CAN, junto con ejemplos prácticos de acoplamiento con optoacopladores, transistores MOSFET y relés de estado sólido.

4. Firmware C++/MCU: PID Discreto Tustin, FSM y CRC-16 (controlador_tiempo_real_video_07.ino)

```
/*
 * ============================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 07: Video 07 — Parte 4A: Sistemas de Entrada/Salida, Interfases, Direccionamiento y Adaptadores para
Periféricos
 * Archivo: controlador_tiempo_real_video_07.ino
 * ----------------------------------------------------------------------------
 * Descripción Técnica:
 *   Firmware determinista en C/C++ Embebido para Microcontrolador (compatible
 *   con arquitectura AVR / ARM Cortex-M STM32 / ESP32 / RP2040) que implementa:
 *     1. Acceso a nivel de registros con punteros 'volatile' y máscaras de bits.
 *     2. Controlador PID Discreto por Transformación Bilineal de Tustin con
 *        derivada sobre la variable medida (-PV), filtro derivativo N y
 *        esquema Anti-Windup por retro-cálculo y Clamping.
 *     3. Máquina de Estados Finitos (FSM de Moore) no bloqueante para
 *        secuenciación segura e interbloqueo de emergencia.
 *     4. Trama de telemetría con verificación CRC-16-CCITT hacia SBC/PLC.
 *
 * Enfoque del Módulo:
 *   Registros Data/Status/Control, Direccionamiento MMIO vs. PMIO, Handshaking Strobe/ACK, PIA 8255 y Expansor
MCP23017
 * ============================================================================
 */

#include <Arduino.h>
#include <stdint.h>

// --- Definición de Pines de Interfase (Entradas Optoaisladas y Salidas PWM/MOSFET) ---
const uint8_t PIN_ADC_PV_SENSOR   = A0; // Entrada analógica PV (ej. Shunt 165 ohm de lazo 4-20 mA)
const uint8_t PIN_ADC_SP_CONSIGNA = A1; // Consigna local o referencia analógica (0..5V / 0..3.3V)
const uint8_t PIN_DI_MARCHA_NO    = 2;  // Entrada optoacoplada PC817: Pulsador de Marcha (NO)
const uint8_t PIN_DI_PARO_NC      = 3;  // Entrada optoacoplada PC817: Paro / Emergencia (Cableado NC)
const uint8_t PIN_PWM_ACTUADOR    = 9;  // Salida PWM hacia Driver MOSFET / Puente H
const uint8_t PIN_DO_SSR_ALARMA   = 8;  // Salida hacia Relé de Estado Sólido (SSR) / Piloto

// --- Parámetros Nominales del Controlador PID Discreto (Video 07) ---
const float KP_GAIN      = 2.500f;
const float TI_SEC       = 0.400f;
const float TD_SEC       = 0.050f;
const float N_FILT       = 10.0f;
const float TS_SEC       = 0.0020f; // Periodo de muestreo Ts = 2 ms
const float U_MIN_LIM    = 0.0f;
const float U_MAX_LIM    = 4095.0f;
const float PV_MIN_ENG   = 0.0f;
const float PV_MAX_ENG   = 24.0f;

// --- Variables Volátiles Compartidas con Interrupción (ISR) ---
volatile bool     g_flagTickControl = false;
volatile uint32_t g_contadorPulsosEncoder = 0;

// --- Estructura de Estado del Controlador PID Discreto (Tustin + Anti-Windup) ---
struct ControladorPIDDiscreto {
  float kp;
  float ki;
  float kd;
  float n_divisor;
  float ts;
  float u_min;
  float u_max;
  float integral_acum;
  float termino_d_prev;
  float pv_prev;
  float error_prev;
  float u_salida;
};

ControladorPIDDiscreto g_pid;
```

```
// --- Máquina de Estados Finitos (FSM No Bloqueante de Moore) ---
enum EstadoControlador : uint8_t {
  ESTADO_REPOSO_SEGURO   = 0,
  ESTADO_PRECARGA_SUAVE  = 1,
  ESTADO_LAZO_PID_ACTIVO = 2,
  ESTADO_FALLA_PARO_NC   = 3
};

EstadoControlador g_estadoActual = ESTADO_REPOSO_SEGURO;
uint32_t g_prevMicrosLazo  = 0;
uint32_t g_timestampEstado = 0;
uint32_t g_pasoMuestreo    = 0;

// --- Rutina de Servicio de Interrupción Externa (Encoder / Sensor Rápido) ---
void isrPulsoEncoder() {
  g_contadorPulsosEncoder++;
}

// --- Inicialización de Coeficientes del PID Discreto ---
void inicializarPID(ControladorPIDDiscreto &c) {
  c.kp             = KP_GAIN;
  c.ki             = (TI_SEC > 1e-5f) ? (KP_GAIN / TI_SEC) : 0.0f;
  c.kd             = KP_GAIN * TD_SEC;
  c.n_divisor      = N_FILT;
  c.ts             = TS_SEC;
  c.u_min          = U_MIN_LIM;
  c.u_max          = U_MAX_LIM;
  c.integral_acum  = 0.0f;
  c.termino_d_prev = 0.0f;
  c.pv_prev        = 0.0f;
  c.error_prev     = 0.0f;
  c.u_salida       = 0.0f;
}

// --- Evaluación en Tiempo Real del PID por Regla Trapezoidal de Tustin ---
float evaluarPasoPID_Tustin(ControladorPIDDiscreto &c, float sp, float pv) {
  float error = sp - pv;

  // 1. Término Proporcional
  float p_term = c.kp * error;

  // 2. Término Derivativo sobre -PV filtrado por Tustin (Evita Derivative Kick)
  float tau_d  = (c.kp > 1e-6f) ? ((c.kd / c.kp) / c.n_divisor) : 0.0f;
  float denom  = 2.0f * tau_d + c.ts;
  float alpha1 = (2.0f * tau_d - c.ts) / denom;
  float beta0  = (2.0f * c.kd) / denom;
  float d_term = alpha1 * c.termino_d_prev - beta0 * (pv - c.pv_prev);

  // 3. Candidato de Integración Trapezoidal (Tustin)
  float delta_i = 0.5f * c.ki * c.ts * (error + c.error_prev);
  float i_cand  = c.integral_acum + delta_i;

  // 4. Suma Pre-Saturación y Esquema Anti-Windup Condicional (Clamping)
  float u_unsat = p_term + i_cand + d_term;
  float u_sat   = u_unsat;
  if (u_unsat > c.u_max) {
    u_sat = c.u_max;
    if (error < 0.0f) c.integral_acum = i_cand; // Solo integra si ayuda a desaturar
  } else if (u_unsat < c.u_min) {
    u_sat = c.u_min;
    if (error > 0.0f) c.integral_acum = i_cand;
  } else {
    c.integral_acum = i_cand;
  }

  // Actualizar memoria de estados discretos z^-1
  c.termino_d_prev = d_term;
```

```
  c.pv_prev        = pv;
  c.error_prev     = error;
  c.u_salida       = u_sat;
  return u_sat;
}

// --- Escalado Lineal Equivalente a NORM_X + SCALE_X ---
inline float escalarCuentasADC(int cuentasCrudas, int rawMin, int rawMax, float engMin, float engMax) {
  float u_norm = (float)(cuentasCrudas - rawMin) / (float)(rawMax - rawMin);
  if (u_norm < 0.0f) u_norm = 0.0f;
  if (u_norm > 1.0f) u_norm = 1.0f;
  return engMin + u_norm * (engMax - engMin);
}

// --- Cálculo de Verificación de Redundancia Cíclica CRC-16-CCITT (Polinomio 0x1021) ---
uint16_t calcularCRC16(const uint8_t *datos, uint8_t longitud) {
  uint16_t crc = 0xFFFFU;
  for (uint8_t i = 0; i < longitud; i++) {
    crc ^= ((uint16_t)datos[i] << 8);
    for (uint8_t b = 0; b < 8; b++) {
      if (crc & 0x8000U) {
        crc = (crc << 1) ^ 0x1021U;
      } else {
        crc <<= 1;
      }
    }
  }
  return crc;
}

void setup() {
  Serial.begin(115200);
  pinMode(PIN_DI_MARCHA_NO, INPUT_PULLUP);
  pinMode(PIN_DI_PARO_NC, INPUT_PULLUP);
  pinMode(PIN_PWM_ACTUADOR, OUTPUT);
  pinMode(PIN_DO_SSR_ALARMA, OUTPUT);

  attachInterrupt(digitalPinToInterrupt(PIN_DI_MARCHA_NO), isrPulsoEncoder, RISING);
  inicializarPID(g_pid);
  g_prevMicrosLazo  = micros();
  g_timestampEstado = millis();

  Serial.println(F("# ================================================================"));
  Serial.println(F("# CURSO CONTROLADORES EN MECATRONICA - VIDEO 07"));
  Serial.println(F("# Columnas: t_s,estado_fsm,SP_eng,PV_eng,u_PID,pulsos_enc,crc16"));
  Serial.println(F("# ================================================================"));
}

void loop() {
  uint32_t ahoraMicros = micros();
  uint32_t periodoMicros = (uint32_t)(TS_SEC * 1000000.0f);

  if (ahoraMicros - g_prevMicrosLazo >= periodoMicros) {
    g_prevMicrosLazo += periodoMicros;
    float t_sec = g_pasoMuestreo * TS_SEC;

    // 1. Lectura de Entradas Digitales (Seguridad de Paro NC y Marcha NO)
    bool paroCableadoOK = (digitalRead(PIN_DI_PARO_NC) == HIGH);
    bool ordenMarcha    = (digitalRead(PIN_DI_MARCHA_NO) == LOW);

    // 2. Lectura y Escalado de Entradas Analógicas (NORM_X -> SCALE_X)
    int rawPV = analogRead(PIN_ADC_PV_SENSOR);
    int rawSP = analogRead(PIN_ADC_SP_CONSIGNA);
    float pv_eng = escalarCuentasADC(rawPV, 0, 1023, PV_MIN_ENG, PV_MAX_ENG);
    float sp_eng = escalarCuentasADC(rawSP, 0, 1023, PV_MIN_ENG, PV_MAX_ENG);

    // 3. Evaluación de la Máquina de Estados Finitos (FSM de Moore No Bloqueante)
    if (!paroCableadoOK) {
```

```
    g_estadoActual = ESTADO_FALLA_PARO_NC;
  }

  float mandoActuador = 0.0f;
  switch (g_estadoActual) {
    case ESTADO_REPOSO_SEGURO:
      mandoActuador = 0.0f;
      digitalWrite(PIN_DO_SSR_ALARMA, LOW);
      if (ordenMarcha && paroCableadoOK) {
        g_timestampEstado = millis();
        g_estadoActual    = ESTADO_PRECARGA_SUAVE;
      }
      break;

    case ESTADO_PRECARGA_SUAVE:
      mandoActuador = 0.25f * U_MAX_LIM;
      if (millis() - g_timestampEstado >= 200UL) {
        // Transferencia sin saltos (Bumpless) al activar el lazo PID
        g_pid.integral_acum = mandoActuador;
        g_pid.pv_prev       = pv_eng;
        g_estadoActual      = ESTADO_LAZO_PID_ACTIVO;
      }
      break;

    case ESTADO_LAZO_PID_ACTIVO:
      mandoActuador = evaluarPasoPID_Tustin(g_pid, sp_eng, pv_eng);
      digitalWrite(PIN_DO_SSR_ALARMA, LOW);
      break;

    case ESTADO_FALLA_PARO_NC:
    default:
      mandoActuador = 0.0f;
      g_pid.integral_acum = 0.0f;
      digitalWrite(PIN_DO_SSR_ALARMA, HIGH);
      if (paroCableadoOK && ordenMarcha) {
        g_estadoActual = ESTADO_REPOSO_SEGURO;
      }
      break;
  }

  // 4. Escritura de Salida PWM hacia Etapa de Potencia (MOSFET / Puente H)
  float u_norm_pwm = (mandoActuador - U_MIN_LIM) / (U_MAX_LIM - U_MIN_LIM);
  int duty8bit     = constrain((int)(u_norm_pwm * 255.0f), 0, 255);
  analogWrite(PIN_PWM_ACTUADOR, duty8bit);

  // 5. Lectura Atómica de Contador de Interrupción y Telemetría con CRC-16
  if (g_pasoMuestreo % 10 == 0) {
    noInterrupts();
    uint32_t pulsosCopia = g_contadorPulsosEncoder;
    interrupts();

    uint8_t tramaBytes[6];
    tramaBytes[0] = (uint8_t)g_estadoActual;
    tramaBytes[1] = (uint8_t)duty8bit;
    tramaBytes[2] = (uint8_t)((rawPV >> 8) & 0xFF);
    tramaBytes[3] = (uint8_t)(rawPV & 0xFF);
    tramaBytes[4] = (uint8_t)((pulsosCopia >> 8) & 0xFF);
    tramaBytes[5] = (uint8_t)(pulsosCopia & 0xFF);
    uint16_t crc  = calcularCRC16(tramaBytes, 6);

    Serial.print(t_sec, 3);              Serial.print(',');
    Serial.print((int)g_estadoActual);   Serial.print(',');
    Serial.print(sp_eng, 2);             Serial.print(',');
    Serial.print(pv_eng, 2);             Serial.print(',');
    Serial.print(mandoActuador, 2);      Serial.print(',');
    Serial.print(pulsosCopia);           Serial.print(",0x");
    Serial.println(crc, HEX);
  }
```

```
    g_pasoMuestreo++;
  }
}
```

5. Script Python SBC: Sintonización, Índices IAE/ITAE y Tramas (supervision_sbc_y_sintonizacion_video_07.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
VIDEO 07: Video 07 — Parte 4A: Sistemas de Entrada/Salida, Interfases, Direccionamiento y Adaptadores para
Periféricos
Archivo: supervision_sbc_y_sintonizacion_video_07.py
--------------------------------------------------------------------------------
Descripción Técnica:
  Script integral en Python para Computadora de Placa Única (Raspberry Pi /
  SBC) y estación de ingeniería que implementa:
    1. Simulación en lazo cerrado de un controlador PID discreto (Tustin +
       Filtro Derivativo N + Anti-Windup por Clamping y Back-Calculation).
    2. Cálculo cuantitativo de métricas transitorias (tr, tp, Mp, ts) y de
       los cuatro índices integrales de desempeño (IAE, ISE, ITAE, ITSE).
    3. Reglas automáticas de sintonización de Ziegler-Nichols (1.º y 2.º
       método), Cohen-Coon y sintonización robusta IMC / Lambda.
    4. Construcción y validación de tramas binarias Maestro-Coprocesador
       (SBC <-> MCU/PLC) protegidas con suma de verificación CRC-16-CCITT.
================================================================================
"""

import math
import struct
from typing import Dict, List, Tuple

# Parámetros nominales del Video 07
KP_NOMINAL = 2.5
TI_NOMINAL = 0.4
TD_NOMINAL = 0.05
N_FILTRO   = 10
TS_SEG     = 0.0020
U_MIN      = 0
U_MAX      = 4095


def sintonizar_pid_ziegler_nichols_y_lambda(
    K_planta: float, L_retardo: float, T_constante: float, lambda_deseado: float
) -> Dict[str, Dict[str, float]]:
    """
    Calcula las ganancias PID (Kp, Ti, Td) mediante:
      - Ziegler-Nichols 1.er Método (Curva de Reacción FOPDT)
      - Método de Cohen-Coon (para retardo significativo L/T)
      - Método IMC / Regla Lambda (respuesta sin sobreimpulso)
    """
    r = L_retardo / max(T_constante, 1e-6)

    # 1. Ziegler-Nichols Lazo Abierto (PID Estándar ISA)
    kp_zn = (1.2 * T_constante) / (K_planta * max(L_retardo, 1e-6))
    ti_zn = 2.0 * L_retardo
    td_zn = 0.5 * L_retardo

    # 2. Cohen-Coon (PID)
    kp_cc = (1.0 / (K_planta * r)) * (4.0 / 3.0 + r / 4.0)
    ti_cc = L_retardo * (32.0 + 6.0 * r) / (13.0 + 8.0 * r)
    td_cc = L_retardo * 4.0 / (11.0 + 2.0 * r)

    # 3. IMC / Lambda (PI robusto sin sobreimpulso + término D para retardo Padé)
    kp_imc = T_constante / (K_planta * (lambda_deseado + L_retardo))
    ti_imc = T_constante
    td_imc = (T_constante * L_retardo) / (2.0 * T_constante + L_retardo)

    return {
        "Ziegler_Nichols_FOPDT": {
            "Kp": round(kp_zn, 4), "Ti_s": round(ti_zn, 4), "Td_s": round(td_zn, 4)
        },
```

```
        "Cohen_Coon": {
            "Kp": round(kp_cc, 4), "Ti_s": round(ti_cc, 4), "Td_s": round(td_cc, 4)
        },
        "IMC_Lambda_Robusto": {
            "Kp": round(kp_imc, 4), "Ti_s": round(ti_imc, 4), "Td_s": round(td_imc, 4)
        },
    }


def simular_lazo_pid_discreto_tustin(
    kp: float, ti: float, td: float, n_filt: float, ts: float, pasos: int = 800
) -> Dict[str, float]:
    """
    Simula una planta mecatrónica de 2.º orden controlada por un PID discreto
    discretizado por Tustin con Anti-Windup y calcula tr, Mp, ts, IAE, ISE, ITAE.
    """
    ki = kp / ti if ti > 1e-6 else 0.0
    kd = kp * td
    tau_d = (td / n_filt) if n_filt > 0 else 0.0
    denom_d = 2.0 * tau_d + ts
    alpha_d = (2.0 * tau_d - ts) / denom_d
    beta_d  = (2.0 * kd) / denom_d

    # Estado de la planta (wn = 8 rad/s, zeta = 0.45, K = 1.0)
    wn = 8.0
    zeta = 0.45
    y_pos = 0.0
    y_vel = 0.0
    sp = 1.0

    integral = 0.0
    d_prev = 0.0
    y_prev = 0.0
    e_prev = 0.0

    iae = 0.0
    ise = 0.0
    itae = 0.0
    itse = 0.0
    y_max = 0.0
    t_pico = 0.0
    t_10 = None
    t_90 = None
    t_estable = 0.0

    for k in range(pasos):
        t = k * ts
        error = sp - y_pos

        # Acumulación de índices integrales de desempeño
        abs_e = abs(error)
        iae  += abs_e * ts
        ise  += (error ** 2) * ts
        itae += (t * abs_e) * ts
        itse += (t * (error ** 2)) * ts

        # Algoritmo PID Discreto con filtro derivativo sobre -PV
        p_term = kp * error
        d_term = alpha_d * d_prev - beta_d * (y_pos - y_prev)
        i_cand = integral + 0.5 * ki * ts * (error + e_prev)

        u_unsat = p_term + i_cand + d_term
        u_cmd = max(U_MIN, min(U_MAX, u_unsat))

        # Anti-Windup condicional (Clamping)
        if u_unsat == u_cmd or (u_unsat > U_MAX and error < 0) or (u_unsat < U_MIN and error > 0):
            integral = i_cand
```

```
        d_prev = d_term
        y_prev = y_pos
        e_prev = error

        # Integración de la planta física (Euler semi-implícito)
        acel = (wn ** 2) * u_cmd - 2.0 * zeta * wn * y_vel - (wn ** 2) * y_pos
        y_vel += acel * ts
        y_pos += y_vel * ts

        if y_pos >= 0.10 * sp and t_10 is None:
            t_10 = t
        if y_pos >= 0.90 * sp and t_90 is None:
            t_90 = t
        if y_pos > y_max:
            y_max = y_pos
            t_pico = t
        if abs(sp - y_pos) > 0.02 * sp:
            t_estable = t

    tr = (t_90 - t_10) if (t_10 is not None and t_90 is not None) else 0.0
    mp_pct = max(0.0, (y_max - sp) / sp * 100.0)

    return {
        "tiempo_subida_tr_s": round(tr, 4),
        "tiempo_pico_tp_s": round(t_pico, 4),
        "sobreimpulso_Mp_pct": round(mp_pct, 2),
        "tiempo_establecimiento_ts_2pct_s": round(t_estable, 4),
        "IAE": round(iae, 4),
        "ISE": round(ise, 4),
        "ITAE": round(itae, 4),
        "ITSE": round(itse, 4),
    }


def crc16_ccitt(data: bytes) -> int:
    """Calcula CRC-16-CCITT (0x1021, init 0xFFFF) para enlace serie SBC <-> MCU."""
    crc = 0xFFFF
    for byte in data:
        crc ^= (byte << 8)
        for _ in range(8):
            if crc & 0x8000:
                crc = ((crc << 1) ^ 0x1021) & 0xFFFF
            else:
                crc = (crc << 1) & 0xFFFF
    return crc


def empaquetar_trama_consigna_sbc(comando: int, setpoint_float: float) -> bytes:
    """Construye trama binaria: [0xAA, 0x55, CMD, LEN=4, float32_LE, CRC16_BE]."""
    cabecera_y_payload = struct.pack("<BBBf", 0xAA, 0x55, comando, 4, setpoint_float)
    crc = crc16_ccitt(cabecera_y_payload)
    return cabecera_y_payload + struct.pack(">H", crc)


if __name__ == "__main__":
    print("=" * 78)
    print(" CURSO: CONTROLADORES EN MECATRÓNICA — VIDEO 07")
    print(" Enfoque:", "Registros Data/Status/Control, Direccionamiento MMIO vs. PMIO, Handshaking Strobe/ACK, PIA
8255 y Expansor MCP23017")
    print("=" * 78)

    tablas = sintonizar_pid_ziegler_nichols_y_lambda(
        K_planta=1.25, L_retardo=0.15, T_constante=0.85, lambda_deseado=0.35
    )
    print("\n[1] Sintonización Comparativa (Ziegler-Nichols, Cohen-Coon e IMC Lambda):")
    for metodo, gan in tablas.items():
        print(f"    {metodo:24s} -> Kp = {gan['Kp']:6.3f} | Ti = {gan['Ti_s']:6.3f} s | Td = {gan['Td_s']:6.3f}
s")
```

```
metricas = simular_lazo_pid_discreto_tustin(KP_NOMINAL, TI_NOMINAL, TD_NOMINAL, N_FILTRO, 0.005)
print("\n[2] Desempeño Temporal e Índices Integrales (IAE, ISE, ITAE, ITSE):")
for k_m, v_m in metricas.items():
    print(f"    {k_m:34s} : {v_m}")

trama = empaquetar_trama_consigna_sbc(comando=0x01, setpoint_float=12)
print("\n[3] Trama Binaria Híbrida SBC -> MCU con CRC-16-CCITT:", trama.hex(" ").upper())
```

6. Programa PLC IEC 61131-3 (ST + IL): Enclavamiento, GRAFCET y Escalado 4-20 mA (automatizacion_plc_iec61131_video_07.st)

```
(* ================================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 07: Video 07 — Parte 4A: Sistemas de Entrada/Salida, Interfases, Direccionamiento y Adaptadores para
Periféricos
 * Archivo: automatizacion_plc_iec61131_video_07.st
 * --------------------------------------------------------------------------------
 * Estándar: IEC 61131-3 — Texto Estructurado (ST) + Equivalencia en Lista de
 *           Instrucciones (IL) y Peldaños de Escalera (Ladder Diagram)
 * Descripción Técnica:
 *   Bloque de Función Industrial (FB_ControladorCeldaMecatronica_V07) que
 *   integra el Ciclo de Barrido (Scan), circuito de enclavamiento prioritario
 *   al paro (cableado NC), secuenciador GRAFCET de 4 etapas, temporizador TON,
 *   contador de lotes CTU, registro de corrimiento de rechazo de calidad SHRB,
 *   escalado analógico 4-20 mA (NORM_X / SCALE_X con detección de rotura de
 *   hilo < 3.6 mA) y algoritmo PID discreto con Anti-Windup.
 * ================================================================================ *)

FUNCTION_BLOCK FB_ControladorCeldaMecatronica_V07
VAR_INPUT
    (* Entradas Digitales de la Tabla de Imagen de Proceso PII (%IX) *)
    xPulsadorMarcha_NO   : BOOL; (* %IX0.0 - Contacto Normalmente Abierto *)
    xPulsadorParo_NC     : BOOL; (* %IX0.1 - Cableado Físicamente NC en Campo *)
    xReleTermico_NC      : BOOL; (* %IX0.2 - Guardamotor NC (1 = Sin Falla) *)
    xSensorPiezaEntrada  : BOOL; (* %IX0.3 - Sensor Inductivo PNP 24V DC *)
    xCamaraPiezaDefecto  : BOOL; (* %IX0.4 - 1 = Envase Defectuoso Detectado *)
    xPulsoEncoderBanda   : BOOL; (* %IX0.5 - Reloj de Desplazamiento de Banda *)
    xResetContadores     : BOOL; (* %IX0.6 - Reinicio Manual de Lote *)

    (* Entradas Analógicas Crudas del Convertidor ADC (%IW) *)
    iEntradaAnalogicaRaw : INT;  (* %IW64  - Lazo 4-20 mA: 0 a 27648 cuentas *)
    rSetpointIngenieria  : REAL := 14.4;
END_VAR

VAR_OUTPUT
    (* Salidas Digitales hacia la Tabla de Imagen de Salidas PIQ (%QX) *)
    xMotorBandaContactor : BOOL; (* %QX0.0 - Contactor Principal Enclavado *)
    xValvulaDosificadora : BOOL; (* %QX0.1 - Solenoide 24V DC con Diodo Flyback *)
    xCilindroExpulsor    : BOOL; (* %QX0.2 - Rechazo en Posición Bit 5 del SHRB *)
    xAlarmaRoturaCable   : BOOL; (* %QX0.3 - Corriente < 3.6 mA o Paro Activo *)
    xLoteCompletado      : BOOL; (* %QX0.4 - Salida Q del Contador CTU *)

    (* Salida Analógica Escalada y Variables de Diagnóstico HMI *)
    iSalidaAnalogicaRaw  : INT;  (* %QW80  - Mando 0..27648 hacia Variador/SSR *)
    rVariableProcesoPV   : REAL; (* Variable física escalada en unidades de ing. *)
    rSalidaControlPID    : REAL; (* Esfuerzo de control 0.0 a 100.0 % *)
    iEstadoGrafcet       : INT;  (* Etapa activa del GRAFCET: 0, 10, 20, 30 *)
END_VAR

VAR
    (* Instancias de Temporizadores y Contadores Estándar IEC 61131-3 *)
    tonDosificado        : TON;
    ctuContadorLotes     : CTU;

    (* Variables Internas de Escalado NORM_X / SCALE_X *)
    rValorNormalizado    : REAL;
    rMinIngenieria       : REAL := 0.0;
    rMaxIngenieria       : REAL := 24.0;

    (* Registro de Corrimiento de 16 Bits (Shift Register para Rechazo) *)
    wRegistroCorrimiento : WORD := 16#0000;
    xPulsoEncoderPrev    : BOOL := FALSE;
    xFlancoEncoder       : BOOL := FALSE;

    (* Memoria Retentiva del Bloque PID Discreto *)
    rKp                  : REAL := 2.500;
    rTi_s                : REAL := 0.400;
    rTd_s                : REAL := 0.050;
```

```
    rTs_s                : REAL := 0.0020;
    rIntegralAcum        : REAL := 0.0;
    rErrorPrev           : REAL := 0.0;
    rPvPrev              : REAL := 0.0;
END_VAR

(* ================================================================================
 * 1. EQUIVALENCIA EN LISTA DE INSTRUCCIONES (IL) DEL PELDAÑO DE ENCLAVAMIENTO:
 *      LD     xPulsadorMarcha_NO
 *      OR     xMotorBandaContactor
 *      AND    xPulsadorParo_NC
 *      AND    xReleTermico_NC
 *      ANDN   xAlarmaRoturaCable
 *      ST     xMotorBandaContactor
 * ================================================================================ *)
xMotorBandaContactor := (xPulsadorMarcha_NO OR xMotorBandaContactor)
                        AND xPulsadorParo_NC
                        AND xReleTermico_NC
                        AND NOT xAlarmaRoturaCable;

(* ================================================================================
 * 2. NORMALIZACIÓN (NORM_X) Y ESCALADO LINEAL (SCALE_X) DE LAZO 4-20 mA
 *    Detección de rotura de hilo si iEntradaAnalogicaRaw < -1728 (< 3.0 mA)
 * ================================================================================ *)
IF (iEntradaAnalogicaRaw < -1728) OR (NOT xPulsadorParo_NC) OR (NOT xReleTermico_NC) THEN
    xAlarmaRoturaCable := TRUE;
    rValorNormalizado  := 0.0;
ELSE
    xAlarmaRoturaCable := FALSE;
    (* NORM_X: Convierte 0..27648 cuentas a rango flotante 0.0..1.0 *)
    rValorNormalizado  := INT_TO_REAL(iEntradaAnalogicaRaw) / 27648.0;
    IF rValorNormalizado < 0.0 THEN rValorNormalizado := 0.0; END_IF;
    IF rValorNormalizado > 1.0 THEN rValorNormalizado := 1.0; END_IF;
END_IF;

(* SCALE_X: Convierte 0.0..1.0 a Unidades de Ingeniería [rMinIngenieria .. rMaxIngenieria] *)
rVariableProcesoPV := rMinIngenieria + rValorNormalizado * (rMaxIngenieria - rMinIngenieria);

(* ================================================================================
 * 3. REGISTRO DE CORRIMIENTO DE BITS (SHRB / BSL) SINCRONIZADO CON ENCODER
 * ================================================================================ *)
xFlancoEncoder    := xPulsoEncoderBanda AND NOT xPulsoEncoderPrev;
xPulsoEncoderPrev := xPulsoEncoderBanda;

IF xFlancoEncoder AND xMotorBandaContactor THEN
    wRegistroCorrimiento := SHL(IN := wRegistroCorrimiento, N := 1);
    IF xCamaraPiezaDefecto THEN
        wRegistroCorrimiento := wRegistroCorrimiento OR 16#0001;
    END_IF;
END_IF;

(* Si el bit desplazado llega a la estación 5 (bit 4, máscara 16#0010), activa expulsor *)
xCilindroExpulsor := (wRegistroCorrimiento AND 16#0010) <> 16#0000;

(* ================================================================================
 * 4. SECUENCIADOR GRAFCET / SFC DE 4 ETAPAS CON TEMPORIZADOR TON Y CONTADOR CTU
 * ================================================================================ *)
IF NOT xMotorBandaContactor THEN
    iEstadoGrafcet := 0;
END_IF;

CASE iEstadoGrafcet OF
    0: (* Etapa 0: Reposo esperando pieza en estación *)
        xValvulaDosificadora := FALSE;
        IF xMotorBandaContactor AND xSensorPiezaEntrada THEN
            iEstadoGrafcet := 10;
        END_IF;
```

```
    10: (* Etapa 10: Dosificación temporizada con bloque TON *)
        xValvulaDosificadora := TRUE;
        IF tonDosificado.Q THEN
            xValvulaDosificadora := FALSE;
            iEstadoGrafcet       := 20;
        END_IF;

    20: (* Etapa 20: Incremento de contador de lote y espera de liberación *)
        xValvulaDosificadora := FALSE;
        IF NOT xSensorPiezaEntrada THEN
            iEstadoGrafcet := 0;
        END_IF;
END_CASE;

tonDosificado(IN := (iEstadoGrafcet = 10), PT := T#1500MS);
ctuContadorLotes(CU := (iEstadoGrafcet = 20), R := xResetContadores, PV := 50);
xLoteCompletado := ctuContadorLotes.Q;

(* ================================================================================
 * 5. BLOQUE TECNOLÓGICO PID DISCRETO CON ANTI-WINDUP Y DES-ESCALADO ANALÓGICO
 * ================================================================================ *)
IF xMotorBandaContactor AND NOT xAlarmaRoturaCable THEN
    VAR
        rErrorActual : REAL;
        rTerminoP    : REAL;
        rTerminoD    : REAL;
        rCandI       : REAL;
        rSumaUnsat   : REAL;
    END_VAR
    rErrorActual := rSetpointIngenieria - rVariableProcesoPV;
    rTerminoP    := rKp * rErrorActual;
    rTerminoD    := -rKp * rTd_s * (rVariableProcesoPV - rPvPrev) / rTs_s;
    rCandI       := rIntegralAcum + (rKp / rTi_s) * rTs_s * 0.5 * (rErrorActual + rErrorPrev);

    rSumaUnsat   := rTerminoP + rCandI + rTerminoD;
    IF rSumaUnsat > 100.0 THEN
        rSalidaControlPID := 100.0;
    ELSIF rSumaUnsat < 0.0 THEN
        rSalidaControlPID := 0.0;
    ELSE
        rSalidaControlPID := rSumaUnsat;
        rIntegralAcum     := rCandI;
    END_IF;

    rErrorPrev := rErrorActual;
    rPvPrev    := rVariableProcesoPV;
ELSE
    rSalidaControlPID := 0.0;
    rIntegralAcum     := 0.0;
END_IF;

(* Conversión de 0.0..100.0 % a 0..27648 cuentas crudas de salida analógica %QW80 *)
iSalidaAnalogicaRaw := REAL_TO_INT((rSalidaControlPID / 100.0) * 27648.0);

END_FUNCTION_BLOCK
```