# PLC 完整學習教材

可程式邏輯控制器：從初級、中級到高級

**適用對象：** 自動化初學者、電控技師、系統整合工程師

**涵蓋範圍：** 硬體接線、梯形圖、IEC 61131-3、狀態機、PID、工業通訊、運動控制、功能安全、工業 4.0

**編排方式：** 三篇十七章，每章含學習目標、實務筆記與練習題

**版本日期：** 2026 年 7 月 5 日

**第一篇：初級**
入門與基本功

**第二篇：中級**
語言、設計與控制

**第三篇：高級**
通訊、安全與整合

# Contents

# 導讀：如何使用本教材

本教材依能力層級分為三篇：

- **第一篇（初級，第 1–5 章）**：不需任何 PLC 背景。目標是看懂硬體型錄與接線圖、理解掃描週期，並能獨立寫出含自保持、計時器、計數器的梯形圖程式。
- **第二篇（中級，第 6–11 章）**：以 IEC 61131-3 為主軸，學會結構化文字（ST）、資料型別、功能塊模組化與狀態機設計，並掌握類比訊號、PID 調節與 HMI ／警報設計。這是從「會寫」到「寫得好」的分水嶺。
- **第三篇（高級，第 12–17 章）**：面向系統整合：工業通訊協定、伺服運動控制、機械安全與功能安全、軟體工程實務（版本控制、模擬調機、測試），以及工業 4.0 ／ IIoT 資料整合。

建議的學習節奏：初級篇搭配一台入門 PLC（或廠商免費模擬器）逐章實作，約 2–4 週；中級篇以一個完整小專案（例如自動門或小型輸送分揀）貫穿，約 1–2 個月；高級篇依工作領域挑選章節深入。每章結尾的練習題請務必動手完成——PLC 是實作技能，只讀不寫是學不會的。

本書範例以 IEC 61131-3 標準語法為主（各家平台語法略有出入，觀念完全相通），梯形圖符號採國際慣用的 NO ／ NC 接點與線圈表示法。

# Part I

# 初級篇：入門與基本功

第 1 章

# 認識 PLC

**本章學習目標**

理解 PLC 的定義、發展背景與適用場景；能區分 PLC、繼電器控制、DCS 與微控制器；認識主流品牌產品線，看懂基本規格。

## 1.1 什麼是 PLC

可程式邏輯控制器（Programmable Logic Controller, PLC）是一種專為工業環境設計的數位運算電子裝置：它週期性地讀取輸入（按鈕、感測器、極限開關…），依使用者撰寫的程式進行邏輯運算，再驅動輸出（電磁閥、接觸器、指示燈、變頻器…）。與一般電腦相比，PLC 的核心特質是：

- **確定性（Deterministic）**：掃描週期固定且可預測，毫秒級反應。
- **強固性**：耐震動、耐電氣雜訊、寬溫域運作，設計壽命常達 10 年以上。
- **I/O 導向**：硬體與指令集都圍繞著大量離散 / 類比 I/O 點設計。
- **可維護性**：現場電機人員可透過梯形圖直接監看、除錯與修改。

## 1.2 發展簡史

1968 年，美國通用汽車（GM）Hydra-Matic 變速箱廠為了擺脫每次改車型就要重新配線數千顆繼電器的困境，公開徵求「可用程式取代配線」的控制器。1969 年 Modicon 084（Dick Morley 團隊）成為第一台商用 PLC。往後五十餘年的演進大致為：

| 年代 | 里程碑 |
|---|---|
| 1970s | 取代繼電器邏輯；梯形圖成為主流語言 |
| 1980s | 類比控制、PID 進入 PLC；各廠專有通訊網路出現 |
| 1990s | IEC 61131-3 標準發布（1993），統一五種程式語言 |
| 2000s | 工業乙太網興起（EtherNet/IP、PROFINET）；PLC 與運動控制整合為 PAC |
| 2010s | 安全 PLC 普及、OPC UA 標準化、軟體工程方法導入 |
| 2020s | IIoT / 邊緣運算、軟體定義 PLC（Soft PLC）、雲端整合與虛擬調機 |

## 1.3 PLC 與其他控制方案的比較

| | 繼電器控制 | PLC | DCS | 微控制器/工控機 |
|---|---|---|---|---|
| 邏輯變更 | 重新配線 | 改程式即可 | 改組態 | 改韌體、需開發能力 |
| 規模 | 小型、固定 | 數十至數萬點 | 大型連續製程 | 依設計而定 |
| 擅長領域 | 極簡單互鎖 | 離散/批次控制 | 石化、電廠等連續程序 | 量產產品、特殊演算法 |
| 即時性 | 硬體級 | ms 級、確定性 | 100 ms 級 | 依設計 |
| 維護人員 | 電匠 | 電控技師 | 儀控工程師 | 軟韌體工程師 |
| 單點成本 | 低（點少時） | 中 | 高 | 開發成本高、量產便宜 |

**實務筆記**

實務選型口訣：**點數少且永不改**——繼電器；**離散設備與產線**——PLC；**大型連續製程、需高階程序管理**——DCS；**量產設備內嵌控制**——微控制器。現代界線逐漸模糊：高階 PLC（PAC）已能承擔小型 DCS 的角色。

## 1.4 主流品牌與產品線概覽

| 品牌 | 代表系列 | 常見定位 |
|---|---|---|
| Siemens | LOGO! / S7-1200 / S7-1500 | 歐系主流；TIA Portal 整合開發環境 |
| Rockwell (AB) | Micro800 / CompactLogix / ControlLogix | 美系主流；Studio 5000、CCW |
| Mitsubishi | FX5U / Q / iQ-R | 日系主流；GX Works，亞洲市占高 |
| Omron | CP2E / NX / NJ | 日系；Sysmac Studio，機械控制強 |
| Keyence | KV-8000 系列 | 上手快、感測器整合佳 |
| Delta / FATEK / LS | DVP、AS / FBs / XGB | 高性價比，中小型機台常見 |
| Beckhoff | CX / TwinCAT | PC-based、EtherCAT 主導者 |

閱讀規格表時，先看五件事：**I/O 點數與擴充能力**、**程式容量與掃描速度**、**通訊介面**（乙太網路口數、串列埠、支援協定）、**特殊功能**（高速計數、脈衝輸出、PID 迴路數）、**開發軟體與授權費用**。

## 1.5 典型應用場景

輸送帶與分揀、包裝機、充填機、立體倉儲、電鍍與表面處理線、給排水與泵浦站、HVAC 空調箱、隧道與交通號誌、停車場設備、汙水處理廠、食品 CIP 清洗流程等。凡是「感測器進、致動器出、邏輯與順序控制為主」的場合，就是 PLC 的主場。

**本章練習**

1. 說明 PLC 與繼電器控制在「變更邏輯」上的本質差異，並舉一個實例。
2. 你的工廠要新建一條 800 點 I/O 的包裝線，以及一座 24 小時連續運轉的化工反應程序，各應選 PLC 還是 DCS？理由為何？
3. 找一份任意品牌入門 PLC 的型錄，列出它的 I/O 點數、程式容量、通訊介面與支援語言。

第 2 章

# 硬體架構與電氣接線

**本章學習目標**

認識 PLC 系統的組成模組；理解數位輸入的 Sink / Source（NPN / PNP）差異並能正確接線；比較三種數位輸出型式；掌握基本的雜訊對策。

## 2.1 系統組成


小型機（Brick / 一體式）把上述功能整合在單一機身並提供少量擴充；中大型機（模組式 / 機架式）則依需求插槽組合，可熱插拔、可遠端分散 I/O。

## 2.2 數位輸入：Sink 與 Source

24 VDC 數位輸入迴路必有「電流方向」問題，這是初學者最常接錯的地方：

| | **Sink 輸入（漏型）** | **Source 輸入（源型）** |
|---|---|---|
| PLC 輸入端子 | 電流**流入** PLC | 電流**流出** PLC |
| COM 端接法 | COM 接 +24 V | COM 接 0 V |
| 搭配感測器 | **NPN**（輸出拉到 0 V） | **PNP**（輸出拉到 +24 V） |
| 市場慣例 | 日系設備常見 | 歐美設備常見 |

**常見陷阱**

NPN 感測器接到 Source 型輸入（或反之）不會燒毀，但**訊號永遠不動作**。選購三線式感測器（棕 = +24 V、藍 = 0 V、黑 = 訊號）前，務必先確認 PLC 輸入型式。許多 PLC 的 COM 端子可自由接 +24 V 或 0 V，兩種皆可支援，但**同一組 COM 之下不能混用**。

## 2.3 數位輸出型式比較

| | 繼電器輸出 | 電晶體輸出 | TRIAC 輸出 |
|---|---|---|---|
| 負載電源 | AC / DC 皆可 | DC（分 Sink/Source） | AC 專用 |
| 切換速度 | 慢（~10 ms） | 快（μs 級） | 中 |
| 壽命 | 機械壽命有限 | 半導體、近乎無限 | 半導體 |
| 負載能力 | 約 2 A/點 | 約 0.5 A/點 | 約 0.5–1 A/點 |
| 適用 | 通用、小型接觸器 | 高速輸出、電磁閥、脈衝 | AC 燈載、小型 AC 閥 |

**實務筆記**

需要輸出脈衝（步進 / 伺服定位）務必選**電晶體輸出**機種；驅動大型接觸器或馬達，一律經過中繼繼電器或接觸器，不要讓 PLC 輸出直接承受大電流與突波。感性負載（電磁閥、繼電器線圈）並聯突波吸收元件：DC 用飛輪二極體、AC 用 RC 突波吸收器。

## 2.4 類比 I/O 概述

工業類比訊號以 **4–20 mA** 電流迴路為首選（抗雜訊、可偵測斷線：低於 4 mA 即異常），其次為 0–10 V。解析度以位元數表示：12 bit = 4096 階、16 bit = 65536 階。詳細的工程單位換算與濾波在第 10 章說明。

## 2.5 配線與雜訊對策

- **動力與訊號分離**：動力線（變頻器輸出尤甚）與訊號線分槽、垂直交叉。
- **類比與通訊線使用隔離雙絞遮蔽線**，遮蔽層單端接地（通常控制盤端）。
- **接地**：控制盤設專用接地排，類比地與動力地分開匯流後單點接地。
- **電源品質**：PLC 電源與動力電源分開，必要時加隔離變壓器或 UPS。
- **緊急停止迴路必須是硬體迴路**（安全繼電器直接切斷動力），不可只靠 PLC 程式（詳見第 14 章）。

**本章練習**

1. 手邊有一顆 PNP 三線式光電感測器，PLC 輸入 COM 應接 +24 V 還是 0 V？畫出接線圖。
2. 一支電磁閥（DC 24 V, 0.3 A）與一台 5.5 kW 馬達的接觸器，各應如何由 PLC 輸出驅動？
3. 為什麼 4–20 mA 比 0–10 V 更適合長距離傳送？「活零點」（live zero）有什麼好處？

第 3 章

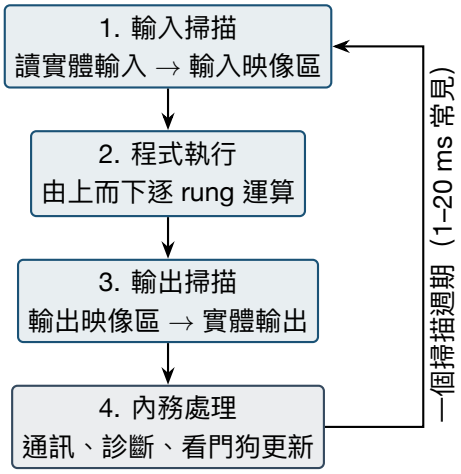
# PLC 運作原理：掃描週期

**本章學習目標**

理解輸入映像區、程式執行、輸出映像區的循環掃描模型；能解釋掃描時間對反應速度的影響；認識看門狗、輸入濾波與保持型記憶體。

## 3.1 循環掃描模型

PLC 不是「事件驅動」，而是**循環掃描（cyclic scan）**：




三個關鍵推論：

1. **同一週期內輸入是「凍結」的**：程式中無論讀幾次 `INPUT1`，值都相同（取自週期開頭的快照）。
2. **輸出在週期結尾才一次更新**：程式中對同一輸出多次寫入，只有**最後一次**生效——這就是「雙線圈問題」的根源（見 4.7 節）。
3. **rung 順序影響結果**：前面 rung 改寫的內部變數，後面 rung 立刻看得到；反之則要等下一個週期。

## 3.2 掃描時間與反應時間

最壞情況的輸入到輸出反應時間 ≈ 輸入濾波時間 + 2 個掃描週期 + 輸出元件動作時間。若掃描週期 5 ms、輸入濾波 10 ms，則保證反應約 20 ms。高速場合（例如編碼器計數、瞬間通過的物件偵測）應使用**高速計數器（HSC）**與 **中斷程式**，它們不受掃描週期限制。

## 3.3 看門狗計時器

CPU 以看門狗（watchdog）監控每次掃描是否在限定時間內完成（如 200 ms）。程式陷入無窮迴圈（如 `WHILE` 條件永真）會觸發看門狗，CPU 進入故障停機、輸出全部復歸——這是保護機制，也是 `WHILE` / `FOR` 在 PLC 中要小心使用的原因。

## 3.4 記憶體區與資料保持

| 區域 | 典型代號 | 說明 |
|---|---|---|
| 輸入映像 | I / X / %I | 每週期自實體輸入更新，程式中唯讀 |
| 輸出映像 | Q / Y / %Q | 程式寫入，週期結尾送實體輸出 |
| 內部繼電器 | M / B | 中間邏輯旗標 |
| 資料暫存器 | D / N / %M | 數值資料 |
| 保持型區域 | Retentive / Latched | **斷電後保留**，用於配方、累計值、狀態 |
| 系統旗標 | S / SM | 首次掃描、時鐘脈衝、故障旗標等 |

**常見陷阱**

「上電後狀態機不動」的常見原因之一：狀態變數放在非保持區，上電歸零後卻沒有任何初始化邏輯。務必利用**首次掃描旗標**（first scan）在上電第一個週期做初始化——決定哪些變數要保持、哪些要歸零，是設計階段就該想清楚的事。

**本章練習**

1. 掃描週期 8 ms、輸入濾波 3 ms，一顆按鈕被按下 5 ms 後放開，PLC 保證讀得到嗎？
2. 解釋為什麼「程式中間讀到的實體輸入」不會隨現場即時變化。
3. 什麼情況該用中斷或高速計數器，而不是一般掃描邏輯？各舉一例。

第 4 章

# 梯形圖基礎

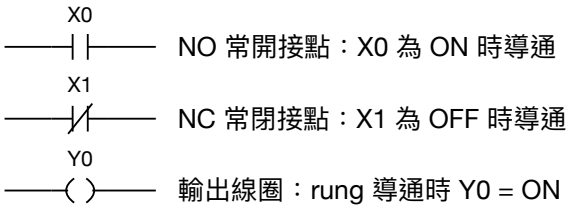
**本章學習目標**

看懂並繪製 NO / NC 接點與線圈；用串並聯實現 AND / OR 邏輯；熟練自保持（seal-in）與互鎖電路；理解執行順序與雙線圈問題。

## 4.1 從繼電器電路到梯形圖

梯形圖（Ladder Diagram, LD）承襲自繼電器控制圖：左右兩條「電源軌」，中間橫向的「梯級（rung）」由接點（條件）與線圈（結果）組成。想像「電力」從左軌經過接點流向右軌：路徑導通，線圈就激磁。

## 4.2 基本元件




**常見陷阱**

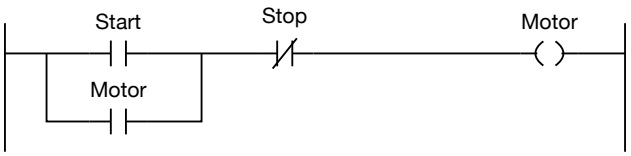
NC 接點的意義是「**變數為 0 時導通**」，與實體開關的常開 / 常閉是兩回事。例如停止按鈕實體上採**常閉接線**（斷線即停機，failsafe），此時 PLC 讀到的訊號平時為 1，程式中反而要用 **NO 接點**來表示「未按下」。把「實體接點型式」與「程式接點型式」搞混，是新手第一大坑。

## 4.3 串聯與並聯：AND 與 OR

接點**串聯**即 AND（全部成立才導通）；**並聯**即 OR（任一成立即導通）。布林代數的 $Y = (A \wedge \neg B) \vee C$ 寫成梯形圖：A 與 B 的 NC 串聯成一條路，C 並聯在旁邊。

## 4.4 自保持電路（Seal-in）

最重要的基本電路：按一下啟動鈕馬達持續運轉，按停止鈕才停。




按下 Start：rung 導通、Motor = ON；下一週期起，Motor 自身的 NO 接點取代 Start 維持導通（「自保持」）；按下 Stop（訊號變 0，NC 斷開）迴路切斷，Motor 復歸。注意 **Stop 串接在自保持迴路之後**，確保任何時候都能停機——「停止優先」。

## 4.5 互鎖電路（正反轉）

馬達正轉與反轉接觸器絕不可同時吸合（相間短路）。程式互鎖：正轉 rung 串入「反轉線圈」的 NC 接點，反轉 rung 串入「正轉線圈」的 NC 接點；兩者互相封鎖。實務上還必須加**硬體互鎖**（兩顆接觸器的機械互鎖與輔助接點互鎖），程式互鎖只是第一道防線。

## 4.6 執行順序

梯形圖由上而下、每個 rung 由左而右求值。設計時把「事件偵測」放前面、「狀態決策」在中間、「輸出驅動」放最後，形成 **輸入處理** $\rightarrow$ **邏輯** $\rightarrow$ **輸出**的三段式結構，可讀性與可預測性最好。

## 4.7 雙線圈問題

同一輸出線圈在兩個 rung 出現時，只有**最後一個 rung** 的結果會送到實體輸出（前面的結果被覆寫）。正確做法：把多個條件以並聯合併到單一 rung，或改用 SET / RESET 指令並嚴格管理優先權。

**本章練習**

1. 畫出 $Y = (A \lor B) \land \neg C$ 的梯形圖。
2. 停止按鈕為什麼實體要用常閉接線？若採常開接線，斷線時會發生什麼事？
3. 設計「兩地啟停」：現場與控制室各有一組啟動 / 停止按鈕，任一處皆可啟停同一台泵。
4. 解釋雙線圈問題為何與掃描模型（第 3 章）直接相關。

第 5 章

# 基本指令：計時、計數與資料處理

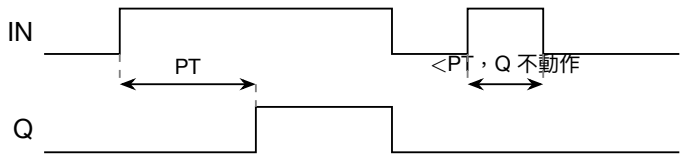
**本章學習目標**

熟練 TON / TOF / TP 三種計時器與 CTU / CTD 計數器；掌握 SET/RESET 與邊緣偵測；會用比較、運算與傳送指令；完成一個綜合小專案。

## 5.1 計時器

| 型式 | 名稱 | 行為 |
|---|---|---|
| TON | 通電延遲 | IN 為 ON 持續達 PT 後 Q 才 ON；IN 斷開立即復歸 |
| TOF | 斷電延遲 | IN 為 ON 時 Q 立即 ON；IN 斷開後 Q 再維持 PT 才 OFF |
| TP | 脈衝 | IN 上升緣觸發，Q 固定輸出 PT 寬度的脈衝 |
| RTO | 保持型 | 同 TON 但 IN 斷開時累計值保留，需另行復歸（部分平台） |

TON 時序圖：




典型用途：TON——馬達啟動後延遲確認壓力建立、防抖動；TOF——停機後風扇續轉散熱；TP——固定寬度的沖洗 / 噴膠脈衝。

## 5.2 計數器

CTU（上數）在 CU 上升緣 $+1$，計數值 $\geq$ PV 時 Q 為 ON；R 復歸。CTD（下數）自 PV 往下數到 0。CTUD 雙向。**計數器吃的是邊緣**：訊號保持 ON 不會連續累加。產線常用「入口感測器 CU、出口感測器 CD」統計區間內工件數——這正是第 9 章狀態機中「車輛計數」的原型。

## 5.3 SET / RESET 與邊緣偵測

SET（S）使線圈 ON 後保持，RESET（R）復歸；等效於 IEC 的 SR / RS 功能塊。優先權要想清楚：**安全相關一律「復歸優先」（RS）**。

邊緣偵測（R_TRIG / F_TRIG、OSR/ONS）把電平轉為單一掃描週期的脈衝，用於「按一下」語意、計數觸發與狀態轉移。**優先使用平台內建指令**，不要手寫「本次值與上次值比較」的三 rung 寫法——上一份審查報告（Rung 1–6）示範了手寫版本的風險：中間變數命名不一致會讓邊緣偵測整組失效。

## 5.4 比較、運算與傳送

比較（EQU/NEQ/GEQ/LEQ…）把數值條件變成 rung 導通條件；運算（ADD/SUB/MUL/DIV）與傳送（MOV）處理資料。兩個實務要點：**整數除法會截斷**（需要小數就先轉 REAL）；**運算指令若每個週期都執行**，累加類邏輯必須搭配邊緣觸發，否則一秒內就加了數百次。

## 5.5 綜合範例：停車場車位管理

需求：入口與出口各一支車輛感測器；顯示剩餘車位；滿位時亮「滿」燈並禁止入口柵欄開啟。

```
VAR
  EntrySensor, ExitSensor : BOOL; // 車輛通過感測器
  trigIn, trigOut : R_TRIG;
  Capacity : INT := 50;
  Occupied : INT;           // 建議放保持區
  FullLamp, GateEnable : BOOL;
END_VAR

trigIn(CLK := EntrySensor);
trigOut(CLK := ExitSensor);

IF trigIn.Q AND (Occupied < Capacity) THEN
   Occupied := Occupied + 1;
END_IF;
IF trigOut.Q AND (Occupied > 0) THEN
   Occupied := Occupied - 1; // 下限保護，避免負數
END_IF;

FullLamp := (Occupied >= Capacity);
GateEnable := NOT FullLamp;
```

麻雀雖小五臟俱全：邊緣觸發、上下限保護、輸出為狀態的純函數。這三個習慣請從第一天就養成。

**本章練習**

1. 用 TON 設計「按鈕長按 3 秒才啟動」的防誤觸邏輯。
2. 紅綠燈：綠 30 s → 黃 3 s → 紅 30 s 循環。先用計時器串接實作，並思考：如果之後要加「行人按鈕縮短紅燈」，這種寫法好改嗎？(第 9 章給出更好的答案)
3. 修改停車場範例：加入「入口柵欄開啟後 10 秒自動關閉，若感測器仍偵測到車則延長」。

4. 為什麼累加運算要用邊緣觸發？寫出不加邊緣觸發時，掃描週期 10 ms 下一秒鐘的誤差。

# Part II

# 中級篇：語言、設計與控制

第 6 章

# IEC 61131-3 與五種程式語言

**本章學習目標**

認識 IEC 61131-3 標準的架構；理解五種語言的特性與適用場合；能讀寫基本的結構化文字（ST）。

## 6.1 標準概觀

IEC 61131-3 是 PLC 程式語言的國際標準，定義了共通的**資料型別**、**程式組織單元（POU）**與**五種語言**。學會標準語法後，跨品牌只需適應開發環境差異，觀念完全可攜。

## 6.2 五種語言

| 語言 | 型態 | 擅長 | 不擅長 |
|---|---|---|---|
| LD 梯形圖 | 圖形 | 離散邏輯、互鎖、現場除錯 | 數學運算、迴圈、字串 |
| FBD 功能塊圖 | 圖形 | 訊號流、連續控制、PID 串接 | 複雜分支邏輯 |
| ST 結構化文字 | 文字 | 演算法、狀態機、資料處理 | 現場人員即時監看較不直觀 |
| SFC 順序功能圖 | 圖形 | 明確的順序 / 批次流程 | 非順序性的平行邏輯 |
| IL 指令表 | 文字 | （已於標準第 3 版棄用） | —— |

## 6.3 ST 快速上手

ST 語法近似 Pascal。以下範例涵蓋八成日常語法：

```
VAR
  temp  : REAL;
  level : INT;
  pumps : ARRAY[1..4] OF BOOL;
  i     : INT;
  tDelay : TON;
END_VAR

(* 條件分支 *)
IF temp > 80.0 THEN
   level := 3;
ELSIF temp > 60.0 THEN
   level := 2;
ELSE
   level := 1;
```

```
END_IF;

(* CASE 多路分支：狀態機的核心語法 *)
CASE level OF
  1: pumps[1] := TRUE;
  2: pumps[1] := TRUE; pumps[2] := TRUE;
  3: FOR i := 1 TO 4 DO
       pumps[i] := TRUE;
     END_FOR;
END_CASE;

(* 功能塊呼叫：計時器 *)
tDelay(IN := pumps[1], PT := T#5s);
IF tDelay.Q THEN
   (* 泵 1 已連續運轉 5 秒 *)
END_IF;
```

**常見陷阱**

FOR / WHILE 迴圈在**單一掃描週期內**跑完全部迭代。迴圈次數過大或條件永真會拖垮掃描時間、觸發看門狗。「等待某事發生」永遠不要用 WHILE 空轉，而是讓狀態機跨週期等待。

## 6.4 語言選擇準則

實務上常見的健康組合：**主流程與設備邏輯用 ST**（好版控、好重構）、**與現場電氣直接對應的互鎖與輸出層用 LD**（電機人員可即時監看）、**清楚的批次順序用 SFC**。同一專案混用語言是正常且被鼓勵的。

**本章練習**

1. 把第 4 章的自保持電路改寫成 ST（提示：`Motor := (Start OR Motor) AND NOT StopPressed;`）。
2. 用 CASE 改寫第 5 章的紅綠燈練習，比較與計時器串接版本的可讀性。
3. 哪些邏輯你會堅持留在 LD？為什麼？

第 7 章

# 資料型別、變數與命名

**本章學習目標**

掌握基本與衍生資料型別；理解變數範圍與實體位址映射；建立一致的命名規範。

## 7.1 基本資料型別

| 型別 | 大小 | 範圍 / 說明 | 典型用途 |
|---|---|---|---|
| BOOL | 1 bit | TRUE / FALSE | 接點、旗標 |
| INT | 16 bit | $-32768 \sim 32767$ | 一般整數、原始類比值 |
| DINT | 32 bit | 約 $\pm 2.1 \times 10^9$ | 計數、位置 |
| UINT / UDINT | 16/32 bit | 無號整數 | 編碼器原始值 |
| REAL | 32 bit | IEEE 754 單精度 | 工程單位、PID |
| LREAL | 64 bit | IEEE 754 雙精度 | 高精度運算 |
| TIME | 32 bit | `T#1h30m15s500ms` | 計時器設定 |
| STRING | 可變 | 字元串 | 配方名、條碼 |
| WORD / DWORD | 16/32 bit | 位元集合 | 通訊封包、狀態字 |

**常見陷阱**

三個高頻錯誤：(1) **INT 溢位**——累計計數請用 DINT；(2) **REAL 不可用等號比較**（`IF r = 0.3` 幾乎必失敗），改用區間比較；(3) **有號/無號混算**——編碼器 UDINT 直接塞進 INT 會變負數。

## 7.2 衍生型別：陣列、結構與列舉

```
TYPE E_MotorState : (IDLE, STARTING, RUNNING, STOPPING, FAULT); END_TYPE

TYPE ST_Motor : STRUCT
   Cmd_Run : BOOL;
   Fb_Running : BOOL; // 接觸器輔助接點回授
   Current : REAL;   // 電流 (A)
   RunHours : DINT;  // 累計運轉時數（保持型）
   State   : E_MotorState;
END_STRUCT; END_TYPE

VAR_GLOBAL
   Motors : ARRAY[1..8] OF ST_Motor; // 八台馬達共用同一套邏輯
END_VAR
```

把「一台設備的所有資料」收進一個結構（UDT），是模組化（第 8 章）與 HMI 標籤規劃（第 11 章）的基礎。**列舉型別取代魔術數字**：寫 `STATE := FAULT` 而不是 `STATE := 5`，上一份審查報告的狀態機若用列舉，可讀性與安全性都會更好。

## 7.3 變數範圍與實體位址

區域變數（POU 內部）優先於全域變數；全域變數只留給真正跨程式共享的資料（實體 I/O 映射、模式旗標、跨設備互鎖）。實體位址以 `AT` 綁定：

```
VAR_GLOBAL
   di_EStopOk AT %IX0.0 : BOOL; // 緊急停止迴路正常
   do_ConveyorOn AT %QX0.0 : BOOL;
   ai_TankLevel AT %IW2 : INT; // 液位原始值
END_VAR
```

**實務筆記**

**I/O 映射層**：實體位址只在一個「映射程式」中讀寫，其餘程式一律使用具名變數。好處：感測器換接點位置只改一行；模擬測試時把映射層換成模擬值即可。

## 7.4 命名規範

一套可行的規範（重點是**全案一致**）：

| 前綴 | 意義 | 範例 |
|---|---|---|
| di_ / do_ | 數位輸入 / 輸出 | `di_PhotoEyeA`、`do_GateOpen` |
| ai_ / ao_ | 類比輸入 / 輸出 | `ai_TankLevel` |
| cmd_ / fb_ | 命令 / 回授 | `cmd_Start`、`fb_Running` |
| t / c | 計時器 / 計數器 | `tHoldDwell`、`cPartsOut` |
| E_ / ST_ / FB_ | 列舉 / 結構 / 功能塊型別 | `E_MotorState` |

**本章練習**

1. `INT` 計數器每秒 +100，多久溢位？換成 `DINT` 呢？
2. 為第 5 章停車場範例設計一個 `ST_ParkingLot` 結構與合理命名。
3. 寫出判斷 REAL 變數「約等於 25.0（允差 0.01）」的正確條件式。

第 8 章

# 程式組織與功能塊模組化

**本章學習目標**

區分 Program、Function Block、Function 三種 POU；理解 FB 實例的記憶特性；能把重複邏輯封裝成可重用的功能塊。

## 8.1 三種 POU

| POU | 有無內部記憶 | 用途 |
|---|---|---|
| Program | 有 | 頂層流程，由任務（Task）排程執行 |
| Function Block (FB) | **有（每個實例獨立）** | 設備邏輯：馬達、閥、軸、PID——可多實例 |
| Function (FC) | 無（純函數） | 換算、限幅、單位轉換等無狀態計算 |

FB 的關鍵在「**實例**」：宣告 `M1, M2 : FB_Motor;` 會得到兩份獨立的內部記憶，同一套邏輯服務多台設備——這是 PLC 世界的物件導向。

## 8.2 範例：馬達控制功能塊

需求：啟動命令後輸出接觸器；2 秒內未收到運轉回授即報故障；故障需復歸才能再啟動。

```
FUNCTION_BLOCK FB_Motor
VAR_INPUT
   Cmd_Run : BOOL; // 運轉命令
   Fb_Aux  : BOOL; // 接觸器輔助接點回授
   Reset   : BOOL; // 故障復歸（上升緣）
END_VAR
VAR_OUTPUT
   Out_Contactor : BOOL;
   Fault      : BOOL;
END_VAR
VAR
   tFbCheck : TON;
   trigReset : R_TRIG;
END_VAR

trigReset(CLK := Reset);
IF trigReset.Q AND NOT Fb_Aux THEN
   Fault := FALSE;          // 確認接觸器已釋放才允許復歸
END_IF;
```

```
Out_Contactor := Cmd_Run AND NOT Fault;

(* 命令已下但回授未到，計時 2 秒判故障 *)
tFbCheck(IN := Out_Contactor AND NOT Fb_Aux, PT := T#2s);
IF tFbCheck.Q THEN
   Fault := TRUE;
END_IF;
END_FUNCTION_BLOCK
```

呼叫端：

```
VAR
   fbPump1, fbPump2 : FB_Motor;
END_VAR

fbPump1(Cmd_Run := cmd_Pump1, Fb_Aux := di_Pump1Aux, Reset := cmd_Reset);
do_Pump1 := fbPump1.Out_Contactor;

fbPump2(Cmd_Run := cmd_Pump2, Fb_Aux := di_Pump2Aux, Reset := cmd_Reset);
do_Pump2 := fbPump2.Out_Contactor;
```

## 8.3 分層架構

成熟專案的常見分層，依賴方向一律由上往下：


**實務筆記**

判斷模組化品質的試金石：**「加一台同型設備要改幾行？」**答案應該是「宣告一個新實例＋接上 I/O」而已。若要複製貼上大段邏輯再逐行改名，表示抽象層次不對。

**本章練習**

1. 把 `FB_Motor` 擴充：加入運轉時數累計（`RunHours`）與「維修模式禁止啟動」輸入。
2. 設計 `FB_Valve`（開 / 關命令、雙極限回授、行程超時故障），列出介面即可。
3. FC 與 FB 各舉三個適用例子，說明為什麼「有無內部記憶」是分界。

# 第 9 章

# 狀態機設計

**本章學習目標**

理解為何順序邏輯應以狀態機組織；掌握「單一寫入點、轉移互斥、逾時處理」三大原則；能以 ST 的 CASE 或 SFC 實作可靠的狀態機。

## 9.1 為什麼需要狀態機

自保持與互鎖能處理「組合邏輯」，但設備的本質是**順序**：待機 → 啟動 → 運轉 → 停止，每一步的合法事件都不同。用一堆旗標與自保持硬拼順序，很快就會出現「旗標爆炸」：沒人說得清楚 12 個 BOOL 的哪 4 個同時為真代表什麼狀態。狀態機把系統行為收斂成**一個狀態變數 + 一張轉移表**，任何時刻只有一個狀態為真。

## 9.2 三大設計原則

1. **單一寫入點**：狀態變數只在一個 CASE 區塊（或一組集中的 rung）被寫入。分散寫入會讓「誰最後寫誰贏」，正確性綁死在掃描順序上。
2. **轉移條件互斥**：同一狀態的多個出口不可同時成立；若可能同時成立，必須明訂優先權（例如故障轉移永遠最優先）。
3. **每個「等待中」的狀態都要有逾時出口**：等不到的回授、卡住的工件，逾時一律轉往 FAULT / ALARM 狀態，**絕不默默跳回待機**（上一份審查報告的問題 6 正是反例：60 秒逾時直接回 IDLE，車還在車道裡）。

## 9.3 ST 標準樣板

```
TYPE E_DoorState : (INIT, CLOSED, OPENING, OPEN, CLOSING, FAULT); END_TYPE

VAR
   State    : E_DoorState;  // 唯一狀態變數
   tMove    : TON;          // 動作逾時
   tHoldOpen : TON;         // 開門停留
   trigOpenCmd, trigReset : R_TRIG;
END_VAR

trigOpenCmd(CLK := di_RadarDetect OR cmd_OpenButton);
trigReset(CLK := cmd_Reset);
```

```
CASE State OF
  INIT:                          (* 上電初始化：確認位置 *)
    IF di_ClosedLimit THEN State := CLOSED;
    ELSIF di_OpenLimit THEN State := OPEN;
    ELSE                   State := CLOSING; // 位置不明，先關門
    END_IF;

  CLOSED:
    IF trigOpenCmd.Q THEN State := OPENING; END_IF;

  OPENING:
    IF di_OpenLimit THEN State := OPEN;
    ELSIF tMove.Q THEN   State := FAULT; // 逾時必入故障
    END_IF;

  OPEN:
    IF tHoldOpen.Q AND NOT di_SafetyBeam THEN
                         State := CLOSING;
    END_IF;

  CLOSING:
    IF di_SafetyBeam THEN State := OPENING; // 防夾：立即反向
    ELSIF di_ClosedLimit THEN State := CLOSED;
    ELSIF tMove.Q THEN   State := FAULT;
    END_IF;

  FAULT:
    IF trigReset.Q THEN  State := INIT; END_IF;
END_CASE;

(* 計時器由狀態驅動 *)
tMove(IN := (State = OPENING) OR (State = CLOSING), PT := T#15s);
tHoldOpen(IN := (State = OPEN), PT := T#5s);

(* 輸出是狀態的純函數 *)
do_MotorOpen := (State = OPENING);
do_MotorClose := (State = CLOSING);
do_FaultLamp := (State = FAULT);
```

注意樣板的固定結構：**邊緣偵測** → **CASE 轉移** → **計時器** → **輸出**。輸出區沒有任何 IF——輸出永遠可由目前狀態唯一決定，除錯時看一眼狀態值就知道全機該有的行為。

## 9.4 SFC 觀點

SFC 用「步（Step）＋轉移條件（Transition）＋動作（Action）」畫出同一件事，天生保證單一活動步。批次流程（清洗 → 充填 → 封蓋 → 貼標）用 SFC 最直觀；含大量平行分支或非順序性互鎖時，ST 狀態機較靈活。

**常見陷阱**

狀態機四大反模式（皆為實案常見錯誤）：(1) 狀態變數多處寫入；(2) 上電狀態（INIT）沒有出口；(3) 逾時直接跳回待機而非故障；(4) 復歸按鈕無條件強制改狀態。寫完程式後拿這四條逐一檢查，可擋掉大多數現場事故。

**本章練習**

1. 為自動門範例畫出完整狀態轉移圖（含 FAULT 的進出）。
2. 把第 5 章紅綠燈改為狀態機版本，並加入「行人按鈕：綠燈至少已亮 10 秒即提前轉黃」。
3. 設計單車道雙向會車控制的狀態機（兩端感測器 A、B），列出狀態、事件、轉移表，並說明你如何處理「兩端同時來車」與「車輛中途拋錨」。

第 10 章

# 類比訊號處理與 PID 控制

**本章學習目標**

會做原始值與工程單位的換算與濾波；理解 PID 三項的物理意義；掌握基本調參步驟與 anti-windup 等實務要點。

## 10.1 工程單位換算（Scaling）

類比模組把 4–20 mA 轉成原始整數（例如 0–27648 或 6400–32000，依平台而異）。線性換算公式：

$$EU = EU_{\text{min}} + \left( \frac{Raw - Raw_{\text{min}}}{Raw_{\text{max}} - Raw_{\text{min}}} \right) \times (EU_{\text{max}} - EU_{\text{min}})$$

```
FUNCTION F_Scale : REAL
VAR_INPUT
   Raw : INT;
   RawMin, RawMax : INT;
   EuMin, EuMax : REAL;
END_VAR
F_Scale := EuMin + (INT_TO_REAL(Raw - RawMin) /
        INT_TO_REAL(RawMax - RawMin)) * (EuMax - EuMin);
END_FUNCTION
```

搭配斷線偵測：4–20 mA 訊號低於對應 3.6 mA 的原始值即判定斷線，鎖住最後有效值並發警報，不要讓 PID 拿斷線值去全開閥門。

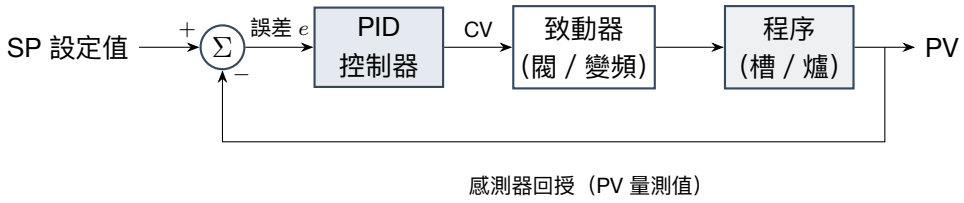
## 10.2 濾波

一階低通濾波（指數平滑）就能解決大部分雜訊：

$$y_k = y_{k-1} + \alpha\,(x_k - y_{k-1}), \qquad 0 < \alpha \le 1$$

$\alpha$ 越小越平滑但越遲鈍。注意濾波會增加相位延遲，控制迴路內的濾波要節制。

## 10.3 PID 原理




$$CV(t) = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

- **P（比例）**：誤差多大就出多少力。只有 P 會留下穩態誤差。
- **I（積分）**：把「一直存在的小誤差」累積成力量，消除穩態誤差；太強會震盪。
- **D（微分）**：對誤差變化率反應，抑制過衝；對雜訊敏感，溫控常用、流量控制常關。

## 10.4 調參

Ziegler–Nichols 閉環法：先只開 P，加大 $K_p$ 直到系統等幅震盪，記下臨界增益 $K_u$ 與震盪週期 $T_u$：

| 控制器 | $K_p$ | $T_i$ | $T_d$ |
|---|---|---|---|
| P | $0.5\,K_u$ | — | — |
| PI | $0.45\,K_u$ | $T_u/1.2$ | — |
| PID | $0.6\,K_u$ | $T_u/2$ | $T_u/8$ |

Z–N 給的是「能動的起點」，通常偏激進；實務上常再把 $K_p$ 降 30–50%。溫控類慢程序更常用開環階躍測試（反應曲線法）或廠商自動調諧（autotune）。

## 10.5 實務要點

- **積分抗飽和（anti-windup）**：CV 已到上下限時停止積分累積，否則設定值大幅變動後會嚴重過衝。多數平台內建，須確認有啟用。
- **無擾切換（bumpless transfer）**：手動/自動模式切換時，讓 CV 從目前值延續。
- **取樣週期**：PID 應以固定週期任務執行（如 100 ms），不要放在掃描時間會漂移的主程式。
- **方向（正/反作用）**：加熱是反作用（PV 高 → CV 小）、冷卻是正作用，接反的迴路會全速衝向極限。

**本章練習**

1. 原始值 0–27648 對應 4–20 mA、量程 0–16 bar：原始值 17280 是幾 bar？
2. 只有 P 控制為什麼會有穩態誤差？I 項如何消除它？

3. 一個溫控迴路超調嚴重、收斂很慢，你會先動 $K_p$、$T_i$ 還是 $T_d$？說明理由。

第 11 章

# HMI 介接與警報設計

**本章學習目標**

規劃清晰的 PLC–HMI 標籤介面；設計符合操作習慣的畫面層級；建立有紀律的警報系統。

## 11.1 PLC 與 HMI 的分工

原則：**所有控制邏輯在 PLC，HMI 只做顯示與命令發送**。HMI 腳本裡藏邏輯（例如按鈕按下時由 HMI 計算後直接寫多個標籤）是維護災難：HMI 當機或通訊中斷時邏輯就消失，而且沒人會去 HMI 裡找 bug。

## 11.2 標籤介面設計

為每台設備定義固定的介面結構，HMI 只讀寫這一區：

```
TYPE ST_HMI_Motor : STRUCT
   (* HMI -> PLC：命令 *)
   Cmd_Start, Cmd_Stop, Cmd_Reset : BOOL; // PLC 收到後自行清除
   Cmd_SpeedSetpoint : REAL;
   (* PLC -> HMI：狀態 *)
   Sts_Running, Sts_Fault : BOOL;
   Sts_State : INT;   // 對應列舉，HMI 以文字清單顯示
   Sts_Current : REAL;
END_STRUCT; END_TYPE
```

**實務筆記**

按鈕命令用「PLC 收到後清除」的握手方式（momentary 由 PLC 清 flag），不要依賴 HMI 的「放開時寫 0」——通訊瞬斷時按鈕會「卡住」。數值設定則要在 PLC 端做**上下限檢查**，永遠不信任來自 HMI 的原始輸入。

## 11.3 畫面設計原則

- 層級：總覽 → 區域 → 設備詳情 → 參數/維護，任何畫面三次點擊內可達。
- 高效能 HMI 理念：灰階為主，**顏色只留給異常**（紅 = 故障、黃 = 警告），正常運轉的畫面應該是「安靜」的。
- 狀態顯示文字化（「運轉中/停止/故障」），不要只用顏色區分（色盲友善）。

- 關鍵操作（清除計數、切換模式）加確認對話與權限分級。

## 11.4 警報設計

參考 ISA-18.2 的核心觀念：**每個警報都必須「需要操作員採取行動」**，否則它是事件（event），只該進紀錄不該響鈴。

- 每個警報定義：條件、延遲（防抖動）、優先級、操作員應採取的行動。
- 類比警報加**死區（deadband）**與**延遲**，避免在門檻附近抖動洗版。
- 區分「未確認/已確認」與「動作中/已復歸」四種組合狀態。
- 警報洪水（alarm flood）對策：分組抑制——上游斷電時抑制所有下游衍生警報。

**本章練習**

1. 為第 8 章的 `FB_Motor` 設計對應的 HMI 介面結構。
2. 「油壓低」警報在門檻附近每秒進出十幾次，給出兩種平息它的方法。
3. 為什麼「HMI 放開按鈕寫 0」不可靠？畫出通訊中斷時的事件順序。

# Part III

# 高級篇：通訊、安全與系統整合

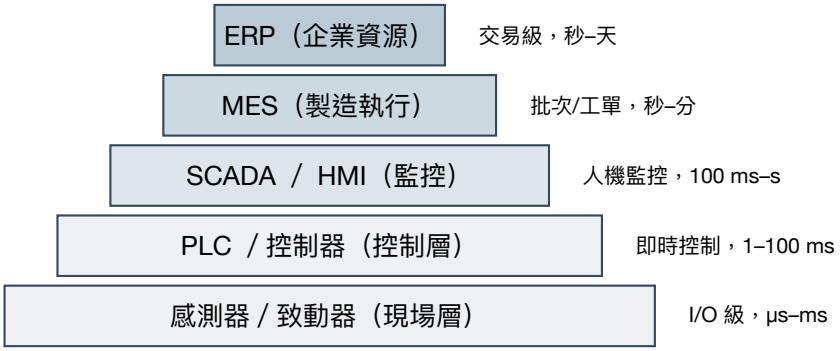
第 12 章

# 工業通訊

**本章學習目標**

理解工廠通訊層級；掌握 Modbus RTU/TCP 的定址與功能碼；比較主流工業乙太網協定；認識 OPC UA 與 MQTT 的角色。

## 12.1 通訊層級（自動化金字塔）




越往下即時性要求越高、資料量越小；越往上相反。協定選擇即是在對的層級用對的工具。

## 12.2 串列通訊與 Modbus RTU

RS-485 差動傳輸、多點匯流排（1 主多從，常見 32 節點）、距離可達 1200 m。Modbus RTU 是其上最普及的協定，記憶體模型四區：

| 區域 | 傳統位址 | 存取 | 常用功能碼 |
|---|---|---|---|
| Coils（線圈） | 0xxxx | 讀寫 bit | 01 讀、05/15 寫 |
| Discrete Inputs | 1xxxx | 唯讀 bit | 02 讀 |
| Input Registers | 3xxxx | 唯讀 16-bit | 04 讀 |
| Holding Registers | 4xxxx | 讀寫 16-bit | 03 讀、06/16 寫 |

**常見陷阱**

Modbus 整合三大地雷：(1) **位址偏移一位**——文件寫 40001，封包裡是 0；(2) **32-bit 值的字序（word swap）**——兩個暫存器組 REAL 時高低字順序各家不同；(3) **輪詢節奏**——RS-485 上從站多、輪詢快，逾時設定不當會拖垮整條匯流排。整合前先拿 Modbus 測試工具（如 Modbus Poll）驗證對方定址，再寫 PLC 程式。

## 12.3 工業乙太網

| | Modbus TCP | EtherNet/IP | PROFINET | EtherCAT |
|---|---|---|---|---|
| 主要陣營 | 開放/施耐德 | Rockwell/ODVA | Siemens/PI | Beckhoff/ETG |
| 即時性 | 一般（TCP） | 中（CIP, UDP I/O） | 中–高（RT/IRT） | 極高（μs 級） |
| 拓撲 | 標準乙太網 | 標準乙太網 | 標準/專用交換器 | 線形串接（從站硬體） |
| 典型應用 | 儀表、電表、簡單整合 | 美系產線 | 歐系產線 | 高速運動控制 |
| 組態難度 | 低 | 中 | 中 | 中 |

選擇通常跟著 PLC 品牌走（AB → EtherNet/IP、Siemens → PROFINET、運動控制密集 → EtherCAT），跨陣營整合時 Modbus TCP 常是最務實的公分母。

## 12.4 OPC UA 與 MQTT

**OPC UA**：跨平台的工業資料標準——自帶資訊模型（物件、階層、單位）、安全機制（憑證、加密）與服務（讀寫、訂閱、歷史）。是控制層與 IT 層之間的標準橋樑，新一代 PLC 多半內建 UA Server。

**MQTT**：輕量發布/訂閱協定，經 broker 轉發，適合大量分散節點上雲。搭配 Sparkplug B 規範可補足工業語意。粗略分工：**廠內設備互通用 OPC UA，上雲遙測用 MQTT**。

**本章練習**

1. 變頻器手冊寫「頻率命令在 40003」，PLC 的 Modbus 讀取位址該填多少？兩種可能都寫出來。
2. 一個 REAL 佔幾個 Modbus holding register？讀出來數值錯亂時你會檢查什麼？
3. 你的產線有 AB 與 Siemens 兩套 PLC 要交換 20 個 tag，提出兩種方案並比較。

第 13 章

# 運動控制

**本章學習目標**

認識伺服系統組成與兩種指令介面；理解原點復歸、電子齒輪與電子凸輪；會使用 PLCopen 標準功能塊的基本流程。

## 13.1 伺服系統組成

控制器（PLC/運動控制器）→ 驅動器（Drive/Amplifier）→ 伺服馬達（含編碼器）→ 機構（滾珠螺桿、皮帶、減速機）。三層迴路由內而外：電流環、速度環、位置環，前兩者在驅動器內，位置命令來自控制器。

## 13.2 兩種指令介面

| | 脈衝列（Pulse Train） | 通訊型（EtherCAT/CC-Link IE 等） |
|---|---|---|
| 接線 | 脈衝 + 方向，點對點 | 網路線串接多軸 |
| 軸數擴充 | 受限於 PLC 脈衝輸出點 | 數十軸以上 |
| 回授資訊 | 少（需另接） | 位置/轉矩/警報全都有 |
| 精度與同步 | 一般 | 高（分散時鐘同步） |
| 成本 | 低 | 中高 |
| 適用 | 1–4 軸簡單定位 | 多軸插補、凸輪、飛剪 |

## 13.3 基本概念

- **原點復歸（Homing）**：增量編碼器上電不知道位置，須以原點開關 + Z 相建立座標；絕對值編碼器免復歸，斷電記憶位置。
- **電子齒輪比**：把「指令單位（mm、度）」換算為編碼器脈衝的比例設定；設錯會出現「走 100 mm 實際走 99.7 mm」這類系統性誤差。
- **電子凸輪（E-CAM）**：從軸位置依凸輪表跟隨主軸位置，取代機械凸輪；飛剪、旋切、追膜的核心技術。
- **加減速與 Jerk**：梯形 / S 曲線速度規劃，S 曲線減少機構衝擊。

## 13.4 PLCopen 運動控制功能塊

IEC 61131 生態的標準運動介面，各廠名稱幾乎一致：

```
(* 典型單軸定位流程 *)
MC_Power(Axis := Axis1, Enable := TRUE); // 軸致能
MC_Home(Axis := Axis1, Execute := trigHome.Q); // 原點復歸
MC_MoveAbsolute(Axis := Axis1,
            Execute := trigGo.Q,
            Position := 250.0, // mm
            Velocity := 100.0, // mm/s
            Acceleration := 500.0);
MC_Stop(Axis := Axis1, Execute := cmd_Stop);
MC_Reset(Axis := Axis1, Execute := trigReset.Q); // 清除軸故障
```

軸本身就是一個狀態機（Disabled、StandStill、Homing、Discrete Motion、ErrorStop…），功能塊的 `Done` / `Busy` / `Error` 輸出必須逐一處理——把第 9 章的狀態機方法套到軸控流程，是多軸專案不亂掉的關鍵。

**常見陷阱**

`Execute` 類輸入吃**上升緣**：條件保持 TRUE 不會重複執行，而在 `Busy` 期間再次觸發，各平台行為不一（緩衝、覆蓋或報錯）。另外：**機械極限開關要接進驅動器的硬體極限輸入**，不能只靠軟體極限。

**本章練習**

1. 滾珠螺桿導程 10 mm、馬達編碼器 23 bit/轉、減速比 5:1，走 1 mm 需要多少脈衝當量？
2. 為「取放機構」（原點 → 取料位 → 放料位循環）設計軸控狀態機。
3. 什麼場合值得從脈衝控制升級到 EtherCAT？列出三個判斷指標。

第 14 章

# 機械安全與功能安全

**本章學習目標**

理解「一般控制」與「安全功能」必須分離；認識 ISO 13849 的 PL 與 IEC 62061 的 SIL；掌握急停、安全門、光柵等典型安全迴路的設計原則。

## 14.1 核心觀念：安全不能只靠程式

一般 PLC 的單一 CPU、單一輸出電晶體都可能以「危險側」失效（輸出短路導通、程式凍結）。因此**安全功能必須由安全等級的元件鏈**（安全感測器 → 安全繼電器或安全 PLC → 強制導向接觸器）實現，一般 PLC 只「知悉」安全狀態、做停機善後與顯示，**不參與安全功能本身**。

## 14.2 法規與標準地圖

| 標準 | 角色 |
|---|---|
| ISO 12100 | 風險評估與風險減低的總綱（本質安全 → 防護 → 使用資訊） |
| ISO 13849-1 | 控制系統安全相關部件；以 **PL a–e** 分級；類別 B, 1–4 結構 |
| IEC 62061 / IEC 61508 | 功能安全；以 **SIL 1–3** 分級 |
| IEC 60204-1 | 機械電氣設備安全（急停類別 0/1/2、配電、保護連結） |

PL（Performance Level）由風險圖（傷害嚴重度 S、暴露頻率 F、可避免性 P）決定需求等級，再以結構類別、MTTF$_d$、診斷覆蓋率 DC 與共因失效 CCF 驗證達成等級。實務上 PL d / Cat. 3（雙通道、單一故障不喪失安全功能）是機械業最常見的設計點。

## 14.3 典型安全迴路

- **急停（E-Stop）**：常閉雙通道接點 → 安全繼電器 → 切斷動力接觸器；復歸必須**手動且在確認危險排除後**，不可自動復歸。
- **安全門**：門開時停止危險運動；高慣性設備配**安全鎖**（停止完成才允許開門）。
- **安全光柵**：進入偵測；與停止性能距離計算（安全距離公式）綁定。
- **雙手按鈕**：沖壓類設備，兩鈕需在 0.5 s 內同時按下且全程保持。
- **強制導向接觸器回授（EDM）**：安全繼電器檢查接觸器確實釋放才允許再啟動。

## 14.4 安全 PLC

安全 PLC 以雙 CPU 互檢、測試脈衝輸出、SIL/PL 認證韌體實現可程式的安全邏輯，適合安全 I/O 多、邏輯複雜（分區停機、模式相依的安全功能）的設備。安全程式與一般程式**分開編譯**、**簽章與權限管理**；修改安全程式屬於變更管理事件，需重新驗證。

**常見陷阱**

常見違規：把光柵訊號接進一般 PLC，由梯形圖去停馬達——這不構成安全功能，稽核與事故責任上都不成立。急停按鈕串進一般 DI「順便看看」可以，但**切斷動力的路徑必須是安全鏈**。

**本章練習**

1. 說明急停類別 0 與類別 1 的差異，各舉一種適用設備。
2. 為一台有伺服軸的組裝機設計急停與安全門架構（畫出安全鏈方塊圖）。
3. 為什麼安全繼電器要檢查接觸器的強制導向輔助接點（EDM）？

第 15 章

# 軟體工程實務：版本、模擬與測試

**本章學習目標**

把一般軟體工程的紀律帶進 PLC 專案：版本控制、程式規範、模擬與虛擬調機、以及可測試的程式結構。

## 15.1 版本控制與變更管理

- 專案檔進 Git（多數現代 IDE 支援文字化匯出或原生 Git 整合），至少做到：每次出機 / 改機一個 tag、變更說明寫「為什麼」。
- **線上修改（online edit）後必須回收程式**：現場搶修完，把 CPU 內的最新版本上傳回版本庫，否則下次下載會把熱修覆蓋掉——這是業界最常見的「程式遺失」事故。
- 建立「黃金備份」：CPU 程式＋硬體組態＋ HMI 專案＋驅動器參數＋配方，缺一不可，並定期演練還原。

## 15.2 程式規範

團隊層級的規範文件應涵蓋：命名規則（第 7 章）、分層架構（第 8 章）、狀態機樣板（第 9 章）、警報準則（第 11 章）、註解要求（每個 POU 開頭：用途、作者、修改歷史；每段非顯而易見邏輯：意圖而非行為）。

## 15.3 模擬與虛擬調機

| 層級 | 工具 | 能驗證什麼 |
|---|---|---|
| 軟體模擬 | 廠商模擬器（PLCSIM、Sysmac 模擬等） | 邏輯正確性、狀態機轉移 |
| I/O 模擬 | 模擬程式取代 I/O 映射層 | 含時序的流程（感測器延遲、逾時） |
| 虛擬調機 | 3D 機構模型＋物理引擎 | 干涉、節拍、多軸協調 |

第 7–8 章的 I/O 映射層在此展現價值：把映射層換成「行為模型」（下達開閥命令 $\rightarrow$ 2 秒後回授到位），整個主程式可以在辦公室先跑過所有流程與故障情境，現場調機時間常可縮短一半以上。

## 15.4 可測試的程式與單元測試思維

FB 介面乾淨（輸入/輸出明確、不偷用全域變數）就可以被測試：寫一個測試程式，餵入序列化的輸入劇本，檢查輸出與狀態。至少為共用庫（換算函數、通用設備 FB）建立測試案例，升級韌體或重構後重跑一次，防止「改 A 壞 B」。

**實務筆記**

審查他人（或自己）PLC 程式的簡明清單：狀態變數是否單一寫入點？每個等待狀態有無逾時出口？上電初始化是否明確？輸出是否集中且為狀態的函數？計時器/計數器命名是否語義化？邊緣與電平使用是否正確？——這六問可以抓出大部分結構性問題。

**本章練習**

1. 為第 8 章 `FB_Motor` 寫一份測試劇本（表格：時間、輸入、預期輸出）。
2. 設計「輸送帶＋光電」的 I/O 行為模型，用於離線驗證第 9 章的狀態機。
3. 你的工廠發生過「下載舊版覆蓋現場熱修」嗎？寫出一套防止它的流程。

第 16 章

# 試車、除錯與維護

**本章學習目標**

建立系統化的試車流程；安全地使用線上監視與強制；掌握常見故障的排除路徑。

## 16.1 試車流程

1. **冷測（斷動力）**：配線點檢、絕緣/迴路量測、I/O 對點（逐點觸發輸入看 PLC、逐點強制輸出看現場——做記錄、簽名）。
2. **單體測試**：手動模式逐台設備動作，確認方向、極限、回授。
3. **空車測試（dry run）**：自動模式無料運轉，驗證流程、互鎖與節拍。
4. **帶料測試（wet run）**：漸進負載，驗證真實時序與例外處理（堵料、缺料、急停後復機）。
5. **驗收（SAT/FAT）**：依規格逐項驗證並留存紀錄與最終版本備份。

## 16.2 線上監視與強制（Force）

線上監視 rung 導通狀態與變數值是 PLC 最強的除錯武器；「交叉參照（cross reference）」可列出變數所有讀寫位置——查「這個輸出為什麼會 ON」永遠從交叉參照開始。

**常見陷阱**

**強制（Force）會繞過所有程式邏輯直接鎖定 I/O**，等同拆掉一道防線：強制前確認現場無人、掛牌告知；一次只強制一點；用畢立即解除並檢查「強制表已空」。帶著殘留強制把設備交還產線，是重大事故的經典開場。

## 16.3 常見故障排除路徑

| 症狀 | 排查順序 |
|---|---|
| 輸出不動作 | 交叉參照查邏輯 → 輸出映像值 → 模組指示燈 → 端子電壓 → 負載 |
| 輸入讀不到 | 感測器指示燈 → 端子電壓 → COM 極性（Sink/Source）→ 濾波時間 |
| CPU 故障燈 | 讀診斷緩衝區（永遠先看診斷，不要瞎猜）：電池、模組、程式錯誤 |
| 通訊中斷 | 實體層（線材、終端電阻、指示燈）→ 位址/參數 → 逾時與輪詢負載 |
| 偶發誤動作 | 懷疑雜訊：檢查接地、遮蔽、動力線距離；檢查邊緣/電平誤用與掃描相依邏輯 |

## 16.4 預防維護

備品清單（CPU、電源、關鍵模組）、電池更換週期、備份排程、盤內清潔與端子緊固年檢、韌體版本清冊。把「診斷資訊上 HMI」做在專案裡：模組健康、通訊狀態、掃描時間、電池電壓——讓維護人員不用接電腦就能判斷八成的問題。

**本章練習**

1. 制定一份 I/O 對點表格式（欄位自訂），以 10 點 DI/DO 為例填寫。
2. 一顆輸出強制 ON 之後被遺忘，寫出可能的事故鏈與三道防止措施。
3. 為你熟悉的一台設備列出「診斷上 HMI」的十個項目。

第 17 章

# 工業 4.0 與 IIoT 整合

**本章學習目標**

理解 PLC 在智慧工廠資料架構中的位置；掌握資料採集的設計要點；認識邊緣運算與 PLC 技術的發展方向。

## 17.1 資料整合架構

傳統 ISA-95 金字塔逐層上報的架構，正被「**控制層維持金字塔、資料層直上平台**」的混合架構取代：PLC 依然負責即時控制，但透過 OPC UA / MQTT 把資料直接送往歷史資料庫（Historian）、MES 與雲端分析平台，邊緣閘道器（Edge Gateway）負責協定轉換、緩存與前處理。

## 17.2 資料採集設計要點

- **在 PLC 端就把資料整理好**：帶時戳、帶品質旗標、帶批次/工單關聯，而不是丟一堆裸暫存器讓上層猜。
- **事件比輪詢有價值**：狀態轉移、警報、批次開始/結束打包成事件上報；高頻類比才用固定週期採樣。
- **通訊負載隔離**：資料採集使用獨立網口或獨立任務，不允許上層輪詢拖慢控制掃描。
- **斷線緩存**：邊緣端須能在網路中斷時暫存資料，恢復後補傳（store and forward）。

## 17.3 OEE：最常見的第一個應用

設備總合效率 $OEE = \square\square\square \times \square\square\square \times \square\square$。PLC 要提供的原始資料：運轉/停機狀態與停機原因碼、理論節拍與實際產量、良品/不良品計數。**停機原因碼**最有價值也最難做好——在狀態機（第 9 章）的 FAULT 與等待狀態中順手記錄原因碼，遠勝事後人工填報。

## 17.4 資安基本盤

控制網路與 IT 網路分區（DMZ / 防火牆）、關閉不用的服務與埠、變更預設密碼、遠端連線走 VPN 且留存紀錄、韌體弱點追蹤（訂閱廠商資安公告）。IEC 62443 是工控資安的參考框架；最低限度：**絕不把 PLC 直接暴露在網際網路上**。

## 17.5 PLC 的未來

軟體定義 PLC（虛擬化、容器化執行環境）、開放自動化（IEC 61499、廠商中立執行時）、內建 AI 推論的控制器、以及「IT 工具鏈（Git、CI/CD、模擬）成為標配」——本書中級篇與高級篇的工程方法，正是迎接這些趨勢的基本功。

**本章練習**

1. 為一台包裝機設計 OEE 資料介面：列出 tag 清單（名稱、型別、更新時機）。
2. 說明為什麼「PLC 裸暴露於網際網路」不可接受，列出三種真實風險。
3. 規劃你工廠的第一個 IIoT 專案：目標、資料流、所需元件與分工。

第 A 章

# 學習路線圖與資源

## A.1 建議學習路線

| 階段 | 里程碑 | 建議實作專案 |
|---|---|---|
| 初級（1–2 月） | 獨立完成接線＋基本梯形圖 | 三段式輸送帶啟停、紅綠燈、停車場計數 |
| 中級（3–6 月） | ST 狀態機＋模組化＋ PID | 自動門、小型分揀線、恆溫水槽 |
| 高級（6 月起） | 通訊整合＋伺服＋安全設計 | 變頻器 Modbus 整合、單軸取放、含安全門的設備改造 |

## A.2 練習環境

各大廠皆提供免費或試用的開發環境與模擬器（如 Siemens TIA Portal 試用版＋ PLCSIM、Rockwell CCW 免費版、Omron/三菱試用版、CODESYS 免費開發環境）。沒有硬體時，**CODESYS ＋軟體模擬**是零成本練完本書中級篇的可行途徑；初級篇的接線與感測器實務，建議至少購置一台入門 PLC 與幾顆感測器實作。

## A.3 延伸閱讀方向

IEC 61131-3 標準文本與各平台程式手冊、ISA-18.2（警報管理）、ISO 13849-1（機械安全）、IEC 62443（工控資安）、PLCopen Motion 規範，以及各廠牌的最佳實務文件（如 Rockwell 的程式設計指南、Siemens 的程式風格指南）。

# 第 B 章

# 詞彙表

| 術語 | 說明 |
|---|---|
| PLC / PAC | 可程式邏輯控制器 / 可程式自動化控制器（整合運動、程序控制的高階 PLC） |
| Scan Cycle | 掃描週期：輸入 → 程式 → 輸出的循環 |
| I/O Image | 輸入/輸出映像區：實體 I/O 在記憶體中的快照 |
| Watchdog | 看門狗：監控掃描逾時的保護機制 |
| NO / NC | 常開 / 常閉（接點型式；注意實體與程式層面的區別） |
| Sink / Source | 數位 I/O 的電流方向型式，對應 NPN / PNP 感測器 |
| Seal-in | 自保持電路：輸出以自身接點維持導通 |
| TON / TOF / TP | 通電延遲 / 斷電延遲 / 脈衝計時器 |
| R_TRIG / F_TRIG | 上升緣 / 下降緣偵測功能塊 |
| POU | 程式組織單元：Program、Function Block、Function |
| FB Instance | 功能塊實例：同一 FB 型別的獨立記憶副本 |
| UDT | 使用者定義型別（結構） |
| Retentive | 保持型記憶體：斷電後資料保留 |
| First Scan | 首次掃描旗標：上電後第一個週期為真，用於初始化 |
| HSC | 高速計數器：不受掃描週期限制的硬體計數 |
| PID | 比例–積分–微分控制 |
| Anti-windup | 積分抗飽和：輸出飽和時停止積分累積 |
| Scaling | 原始值與工程單位間的線性換算 |
| HMI / SCADA | 人機介面 / 監控與資料擷取系統 |
| Alarm Flood | 警報洪水：短時間大量警報淹沒操作員 |
| Modbus RTU/TCP | 最普及的開放工業通訊協定（串列 / 乙太網） |
| OPC UA | 跨平台工業資料交換標準，含資訊模型與安全機制 |
| MQTT | 輕量發布/訂閱通訊協定，常用於 IIoT 上雲 |
| EtherCAT | 高速工業乙太網，運動控制常用 |
| Homing | 原點復歸：建立軸座標系的程序 |
| E-CAM | 電子凸輪：以凸輪表使從軸跟隨主軸 |
| PL / SIL | 安全性能等級（ISO 13849）/ 安全完整性等級（IEC 61508/62061） |
| E-Stop | 緊急停止；類別 0（立即斷電）/ 類別 1（受控停止後斷電） |
| EDM | 外部裝置監控：安全繼電器檢查接觸器強制導向接點 |
| Force | 強制：繞過程式直接鎖定 I/O 值（高風險除錯手段） |
| Cross Reference | 交叉參照：列出變數在程式中所有讀寫位置 |
| FAT / SAT | 出廠驗收測試 / 現場驗收測試 |

| 術語 | 說明 |
|---|---|
| OEE | 設備總合效率 = 稼動率 × 性能率 × 良率 |
| Digital Twin | 數位孿生：設備的虛擬模型，用於模擬與虛擬調機 |
| IEC 61131-3 | PLC 程式語言國際標準（LD、FBD、ST、SFC、IL） |
| IEC 62443 | 工業自動化與控制系統資安標準 |