86LDmicro 使用手冊

86Duino IDE 502

適用軟體：86LDmicro 2.0.0

平台：QEC-M-02（Vortex86EX2）

IDE：86Duino Coding 502 Preview（20261105）

對象：工程師 / 技術人員

版本：v2.0

發行日期：2026 年 11 月

修訂紀錄

版本	日期	主要變更
v2.0	2026-11	86LDmicro 2.0.0 首版：IEC 61131-3（LD、ST、SFC）、EtherCAT、Motion86、86HMI、線上監看

目錄

修訂紀錄 ii
目錄 iii
第 1 章　簡介 1
1.1　關於本手冊 1
1.2　關於 86LDmicro 3
1.3　86Duino 與 QEC-M-02 6
1.4　與 LDmicro 1.1 的主要差異 7
第 2 章　環境安裝 9
2.1　系統需求 9
2.2　安裝 86Duino Coding 502 10
2.3　安裝 86LDmicro 工具 12
2.4　硬體連接與 COM 埠 13
2.5　啟動 86LDmicro 16
2.6　選項 22
第 3 章　介面導覽 24
3.1　主視窗與狀態列 24
3.2　主選單 26
3.3　工具列 35
3.4　專案樹 36
3.5　POU 編輯器與右側欄 39
3.6　工具視窗 42
3.7　分頁、左右並排、縮放與尋找 45
第 4 章　專案與設定 49
4.1　專案的檔案結構 49
4.2　專案設定 51
4.3　工作（Task） 53
4.4　裝置 56
4.5　全域變數與資料型別 59
4.6　I/O 對應 63
4.7　新增 POU 與使用者程式 68
第 5 章　快速入門：從空白到上傳 71
5.1　步驟 1：開啟 86Duino IDE，建立 sketch，選擇開發板與序列埠 72
5.2　步驟 2：從 IDE 開啟 86LDmicro，建立新專案 73
5.3　步驟 3：認識新專案：專案樹與階梯圖 Main 74
5.4　步驟 4：在 GVL 宣告變數 75
5.5　步驟 5：插入常開接點 Xstart 76
5.6　步驟 6：串聯常閉接點 Xstop 與線圈 Rrun 77
5.7　步驟 7：並聯 Rrun 接點，形成自保迴路 78

5.8 步驟 8：新增網路 2：TON 計時器延遲 2 秒點亮指示燈 79
5.9 步驟 9：I/O 對應：把變數接到 QEC-M-02 的實體 I/O 81
5.10 步驟 10：檢查（Shift+F7）與模擬（F5） 82
5.11 步驟 11：建置、編譯並上傳（F8） 85
5.12 步驟 12：線上監看（Ctrl+F5） 87
第 6 章 實作範例 89
6.0 通用原則（請先閱讀） 89
6.1 啟動／停止自保 94
6.2 延時熄燈（TOF） 97
6.3 包裝計數（CTU＋比較） 99
6.4 跑馬燈（交替閃爍） 102
6.5 ST：注水控制 104
6.6 SFC：順序控制 107
6.7 C/C++：Serial1 傳送訊息 111
第 7 章 階梯圖指令參考 113
7.1 編輯操作 114
7.2 接點與線圈 120
7.3 邊緣與脈波 122
7.4 計時器 124
7.5 計數器 126
7.6 比較 127
7.7 運算與搬移 128
7.8 IEC 功能塊（CALL）與 ST 指令 131
7.9 流程控制 133
7.10 I/O、序列埠與其他 134
7.11 運動控制指令 136
第 8 章 ST 與 SFC 138
8.1 ST 編輯器 138
8.2 ST 語法速查 143
8.3 自訂 FUNCTION 與 FUNCTION_BLOCK 146
8.4 SFC 編輯器 150
8.5 步驟、轉移與分支 153
8.6 SFC 執行語意 160
第 9 章 模擬與除錯 162
9.1 模擬 162
9.2 模擬 I/O 166
9.3 監看、寫入與強制 168
9.4 趨勢圖 171
9.5 交互參照與搜尋 172
9.6 常見錯誤訊息 175
第 10 章 建置、上傳與線上監看 178

10.1 檢查、建置、編譯與上傳 178
10.2 產生的檔案 182
10.3 序列埠監看 184
10.4 線上監看 185
10.5 事件紀錄與看門狗 189
10.6 上傳 86HMI 資源 192
第 11 章 EtherCAT 193
11.1 概念 193
11.2 完整流程 194
11.3 EtherCAT Configurator 201
11.4 讀入 ENI 與對應變數 219
11.5 SubDevice 頁 223
11.6 PDO 線上值、SDO 與 CoE 226
11.7 疑難排解 229
第 12 章 Motion86 運動控制 230
12.1 概念 230
12.2 建立機台 231
12.3 機台設定 233
12.4 軸設定 235
12.5 機台試車 237
12.6 單軸試車 239
12.7 PLCopen MC 功能塊 240
12.8 M86 機台功能塊 245
12.9 安全機制 249
第 13 章 86HMI 人機介面 251
13.1 啟用 86HMI 251
13.2 綁定元件 257
13.3 元件與互動方式 262
13.4 文字、趨勢、警報與換頁 266
13.5 模擬與線上 269
13.6 資源上傳與語系 271
第 14 章 C/C++ 與 86Duino 整合 273
14.1 使用者程式檔 273
14.2 存取 PLC 變數 275
14.3 掛勾函式 278
14.4 範例 279
14.5 sketch 的 .ino 283
14.6 注意事項 285
第 15 章 匯入、AI、命令列與專案保護 286
15.1 匯入 LDmicro 1.1 專案 286
15.2 匯入 PLCopen XML 289

15.3 AI 助理 291
15.4 MCP 295
15.5 命令列工具 297
15.6 專案保護 299
第 16 章 常見問題 301
附錄 A LDmicro 1.1 對照 308
附錄 B 快捷鍵 310
B.1 主視窗（任何分頁） 310
B.2 尋找列與專案樹 311
B.3 階梯圖編輯器 311
B.4 SFC 編輯器 313
B.5 程式碼編輯器（ST、GVL、資料型別、C/C++、sketch） 313
B.6 AI 助理 313
附錄 C 變數命名與型別 314
C.1 命名習慣 314
C.2 命名規則 314
C.3 資料型別與範圍 315
C.4 常數與 TIME 寫法 316
聯絡我們 317

第 1 章　簡介

1.1　關於本手冊

本手冊是 **86LDmicro 2.0.0** 的完整使用手冊（版本 v2.0），說明如何在 86Duino Coding 502 Preview（20261105）中安裝、啟動 86LDmicro，以 IEC 61131-3 的階梯圖（LD）、結構化文字（ST）、順序功能圖（SFC）與 C/C++ 撰寫 PLC 程式，並在電腦上模擬、上傳到 QEC-M-02 控制器、線上監看。EtherCAT、Motion86 運動控制與 86HMI 人機介面也各有一章。

本手冊以繁體中文說明，主要操作截圖使用 86LDmicro 2.0.0 的英文介面；常用指令第一次出現時附上英文名稱，例如 監看（Watch）、模擬 I/O（Simulation I/O）、工作（Tasks）、資料型別（Data types）。章節編排如下：

章	內容	適合
第 1 ~ 2 章	簡介、環境安裝（IDE、86LDmicro、硬體連接、啟動、選項）	第一次使用
第 3 ~ 4 章	介面導覽（每個選單、工具列、專案樹、工具視窗）、專案與設定（工作、裝置、變數、I/O 對應）	第一次使用、查詢
第 5 章	快速入門：從空白 sketch 到上傳與線上監看，共 12 個步驟	第一次使用
第 6 章	實作範例：自保、延時熄燈、計數、跑馬燈、ST、SFC、C/C++	練習
第 7 ~ 8 章	階梯圖指令參考、ST 與 SFC	撰寫程式
第 9 ~ 10 章	模擬與除錯、建置上傳與線上監看	除錯、上線
第 11 ~ 13 章	EtherCAT、Motion86 運動控制、86HMI 人機介面	進階功能
第 14 ~ 15 章	C/C++ 與 86Duino 整合；匯入、AI、命令列與專案保護	進階功能
第 16 章、附錄	常見問題；LDmicro 1.1 對照、快捷鍵、變數命名與型別	查詢

***提示：還沒有控制器也可以先試用：完成 2.2、2.3 節的安裝後，直接做第 5 章步驟 1 ~ 10（步驟 1 的序列埠可先略過），在電腦上建立、檢查並模擬第一個程式。**

要接上 QEC 控制器時，再完成 2.4 節的硬體連接與第 5 章步驟 11、12（建置上傳、線上監看）。

1.1.1 閱讀方式與標示

- **粗體**：畫面上的按鈕、選單項目或欄位名稱，例如按 **確定**。
- 等寬字：要輸入的文字、變數名稱、檔名或資料夾，例如 `Xstart`、`data\86ld`。
- 「專案 > 新增 PROGRAM（ST）...」：依序點選選單。快捷鍵寫成 Ctrl+B、Shift+F7。
- `<IDE 資料夾>`：86Duino IDE 所在的資料夾（有 `86duino.exe` 的那一層）；`<sketch>`：86Duino sketch 的資料夾。
- **提示**、**注意**、**重要**、**安全**、**新功能** 方框：補充說明、容易出錯的地方與安全事項。

範例專案：第 5 章的快速入門專案 `MyPlc`、第 6 章的各個範例，以及綜合範例 `tools\example\ex30_dispense`）。

> ***提示：**最新版文件請見 86LDmicro「說明」選單開啟的線上文件網站
>
> https://www.qec.tw/86duino/86ldmicro/ ；離線文件放在 `<IDE 資料夾>\tools\LDmicro\tool\docs`。

1.2 關於 86LDmicro

86LDmicro 2.0 是 ICOP 為 86Duino QEC 系列控制器打造的 PLC 開發環境。

它延續 LDmicro 1.1 的階梯圖操作習慣，並以 IEC 61131-3 的專案架構（POU、工作 Task、全域變數、資料型別）重新設計，熟悉 PLC 的工程師可以直接上手，86Duino 的使用者也可以用 PLC 的方式控制機台。

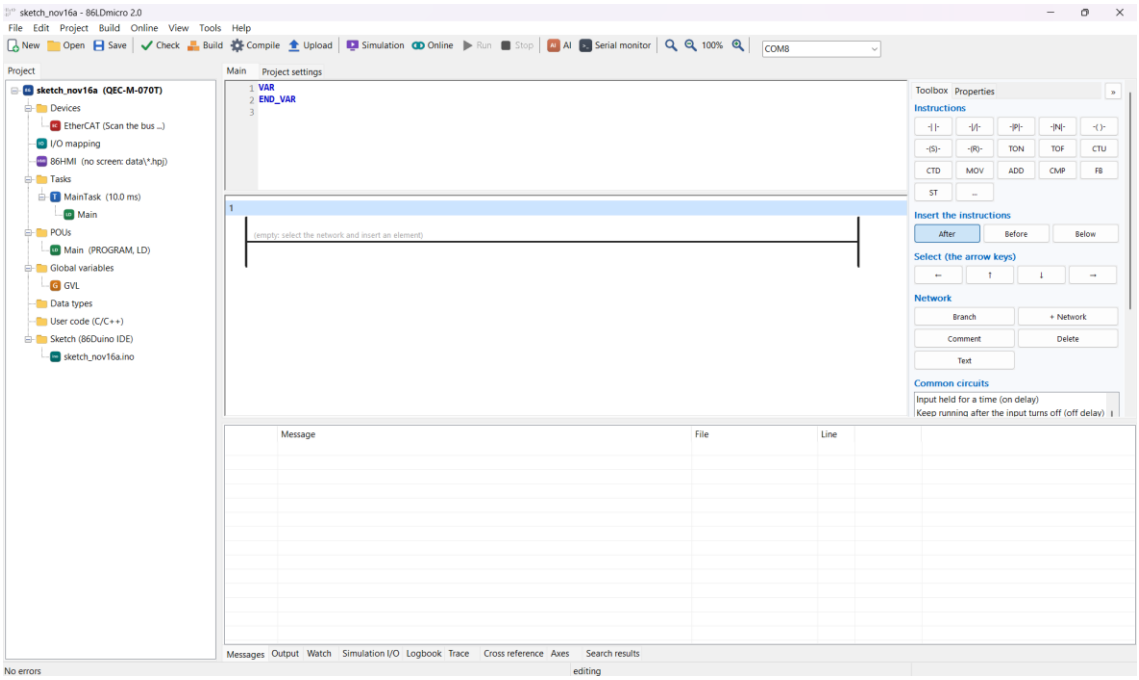


圖 1-0　86LDmicro 2.0.0 Interface

與一般「一行接一行執行」的 C 程式不同，PLC 程式以「掃描週期」的方式執行：

每個週期：讀入所有輸入（本機 DI、EtherCAT 輸入 PDO …）
→ 依工作（Task）中的順序執行每個 PROGRAM（LD / ST / SFC）
→ 把輸出變數寫到 DO、EtherCAT 輸出 PDO
→ 等待下一個週期（預設 MainTask 每 10 ms 一次）

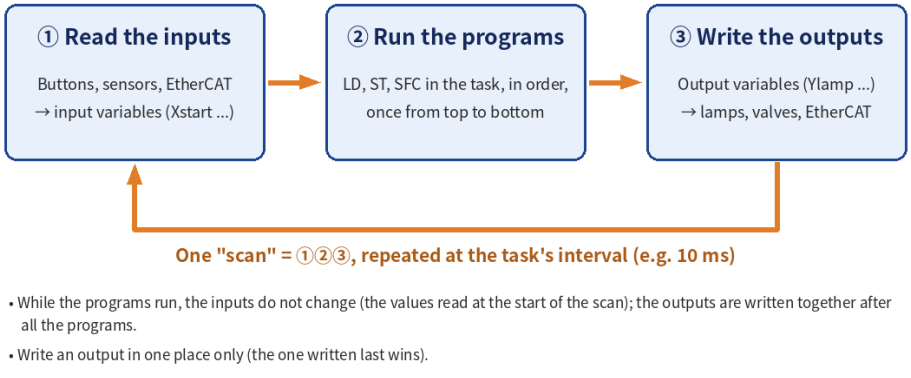


圖 1-1　一個掃描週期：讀入輸入 → 執行程式 → 寫出輸出（示意圖）

1.2.1 運作架構

86LDmicro 不在控制器上解譯程式，而是把整個 PLC 專案轉成 86Duino 的 C++ sketch，再用 86Duino IDE 的編譯器編譯、上傳。電腦與控制器以一條 USB 線連接，上傳與線上監看共用這條線：

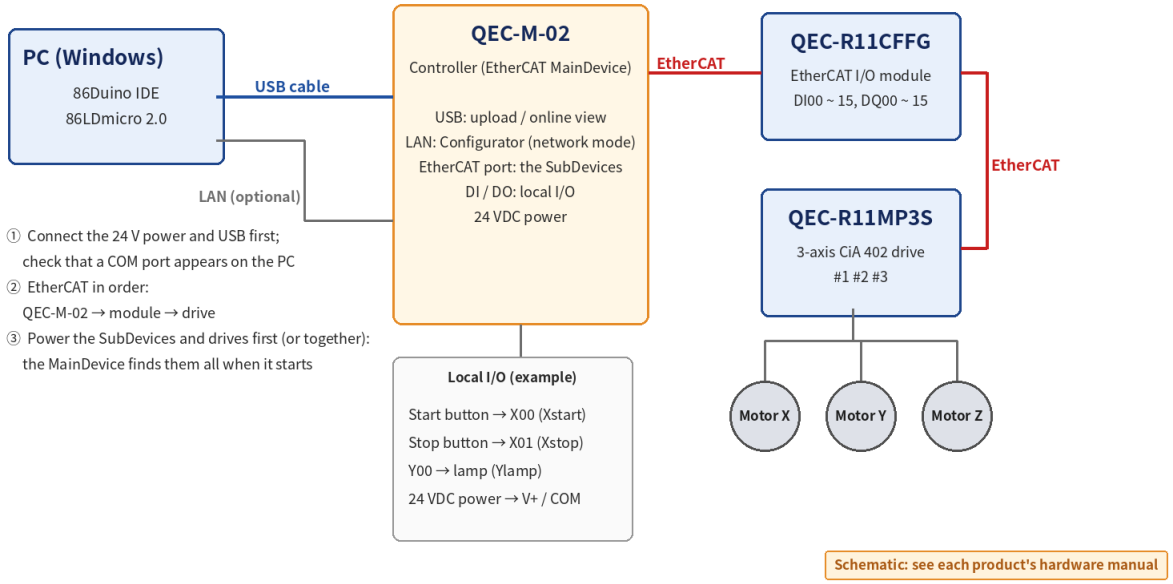


圖 1-2　典型系統：電腦（86Duino IDE + 86LDmicro）經 USB 連到 QEC-M-02，EtherCAT 串接 I/O 模組與驅動器（示意圖）

步驟	誰負責	產出
1. 撰寫	你在 86LDmicro 中編輯 LD / ST / SFC / C/C++、設定裝置與 I/O	`<sketch>\data\86ld\` 下的專案檔（文字檔）
2. 建置	86LDmicro（Ctrl+B）	sketch 中的 `plc86.cpp`、`plc86.h`、`plc86_rt.cpp`、`plc86_ecat.cpp` 等
3. 編譯	86Duino IDE 的 arduino-builder（F7）	控制器的執行檔
4. 上傳	v86dude 經 USB（F8）	程式寫入 QEC 並重新啟動
5. 執行	QEC 上的 86LD run-time	依工作週期掃描程式、讀寫 I/O、EtherCAT、Motion86
6. 監看	86LDmicro 經同一條 USB 線（Ctrl+F5）	即時數值、能流、強制、事件紀錄

因為最後是 C++ 程式，PLC 程式可以和 86Duino / Arduino 的函式庫（EtherCAT、Motion86、Modbus、Ethernet、Serial……）混合使用，C/C++ 程式也可以直接讀寫 PLC 變數（第 14 章）。

1.2.2 主要功能

類別	功能
程式語言	階梯圖（LD）、結構化文字（ST）、順序功能圖（SFC）、C/C++ 使用者程式；可自訂 FUNCTION、FUNCTION_BLOCK
專案架構	工作（週期 / 自由執行 / EtherCAT 週期）、全域變數表（GVL）、資料型別（結構、列舉、陣列）、RETAIN 斷電保持
I/O 與通訊	QEC 本機 DI / DO、EtherCAT（ENI）、86EVA、Modbus RTU / TCP、Ethernet、序列埠
運動控制	Motion86 機台（CiA 402 驅動器 CSP，最多 9 軸）、PLCopen MC_* 功能塊、M86_* 路徑與 G-code、試車頁面
除錯	電腦端完整模擬（含 I/O、計時器、軸）、線上監看、寫入 / 強制、能流、趨勢圖、事件紀錄、序列埠監看
EtherCAT 工具	內嵌 EtherCAT Configurator、SubDevice 頁（PDO 即時值、寫入 / 強制）、CoE 物件字典
人機介面	86HMI 畫面綁定、互動方式設定、模擬視窗、資源上傳
其他	I/O 表 Excel 匯入匯出、專案密碼保護、左右並排編輯、繁中 / 英文 / 日文介面、AI 助理與 MCP、命令列工具、LDmicro 1.1 與 PLCopen XML 匯入

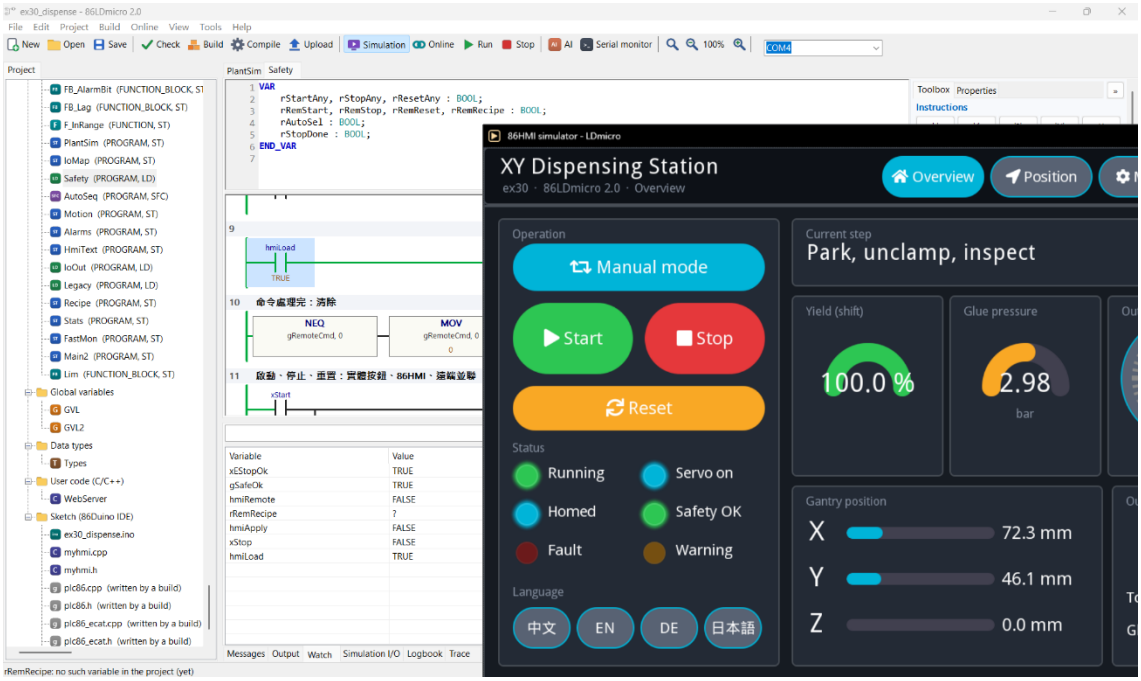


圖 1-3　86LDmicro 2.0：專案樹、階梯圖與工具視窗（模擬中）

1.3 86Duino 與 QEC-M-02

86Duino 是 DMP 以 x86 處理器（Vortex86 系列）為基礎、相容 Arduino 語法的開發平台；QEC 系列是以 86Duino 為核心的工業控制器，QEC-M-02 同時是 EtherCAT 主站。86LDmicro 產生的程式就是一個 86Duino sketch，以 86Duino Coding IDE 編譯後在 QEC 上執行。



圖 1-4　QEC-M-02 控制器

項目	規格 / 說明
處理器	DMP Vortex86EX2（32 位元 x86）
開發環境	86Duino Coding 502 Preview（20261105），Arduino 相容語法；86LDmicro 2.0.0 安裝在 IDE 的 `tools\LDmicro\tool`
本機 I/O	16 點 DI、16 點 DO；86LDmicro 中的通道為 `LocalIO.X00`～`X17`（輸入）、`LocalIO.Y00`～`Y17`（輸出），以八進位編號
EtherCAT	EtherCAT 主站埠：串接 QEC EtherCAT 模組（例如 QEC-R11CFFG）與 CiA 402 驅動器（例如 QEC-R11MP3S）
USB	上傳程式、線上監看、序列埠監看、EtherCAT Configurator（USB 模式）；在電腦上是一個 COM 埠（例如「QEC M02 於 COM4」）
LAN	EtherCAT Configurator 網路模式（預設 192.168.100.1）、Modbus TCP、Ethernet
序列埠	COM1～COM3（RS-232 / 485）：Modbus RTU、使用者程式的 Serial1～Serial3
電源	24 VDC
86LDmicro 整合	IDE「工具 > 86LDmicro」與 IDE 工具列的 86LDmicro 按鈕（2.5 節）
其他控制器	專案設定的「控制器」也可選 QEC-M-01、QEC-M-043T、QEC-M-070T、QEC-M-090T、QEC-M-150T，以及 IDE 中其他 QEC 板子（4.2 節）

***安全：**端子位置、Sink / Source 接法、電壓與電流額定值，請一律以 QEC-M-02 硬體手冊為準。

1.4 與 LDmicro 1.1 的主要差異

86Duino IDE 以前內附的是 LDmicro 1.1（使用手冊 LDmicro UM v1.6.3）。86LDmicro 2.0 保留階梯圖的操作與 1.x 指令，但專案架構、設定方式與除錯工具都不同：

項目	LDmicro 1.1	86LDmicro 2.0（本手冊）
啟動	86Duino IDE 的「工具」選單	IDE「工具 > 86LDmicro」或 IDE 工具列的 86LDmicro 按鈕；也可單獨執行（2.5 節）
程式結構	一個階梯圖（加 ST 區塊）	多個 POU（PROGRAM、FUNCTION_BLOCK、FUNCTION），依工作（Task）排程執行
語言	LD、部分 ST	LD、ST、SFC、C/C++ 使用者程式
變數	VAR LIST；以名稱字首（X、Y、R、T、C…）決定型別	全域變數表（GVL）+ POU 區域變數，IEC 型別（BOOL、INT、REAL、TIME、STRING、結構、陣列…）
掃描週期	Settings → General 的 PLC Cycle Time	工作（Task）的週期；可有多個工作（4.3 節）
設定	Settings 選單：General / Modbus / Virtual Arduino	專案設定、裝置、I/O 對應頁（第 4 章）
I/O 指定	IO List 的 Pin Name	I/O 對應頁：本機、EtherCAT、86EVA、Modbus、腳位；可匯出匯入 Excel（4.6 節）
產生程式	Code Generation 按鈕：`ladder.h`、`myplc.cpp`、`myplc.h`，回 IDE 編譯上傳	建置（Ctrl+B）：`plc86.cpp`、`plc86.h` 等；編譯（F7）、上傳（F8）在 86LDmicro 內完成（第 10 章）
.ino 的呼叫	`setupPlc()` / `loopPlc()`	`plc86_setup()` / `plc86_loop()`（匯入時自動替換）
EtherCAT	86EVA 虛擬裝置（86EVA Configurator）	86EVA，或專案自有主站（ENI）+ 內嵌 EtherCAT Configurator + SubDevice 頁 + CoE（第 11 章）
運動控制	VirtualServo / 脈波	Motion86 機台（CiA 402 CSP，最多 9 軸）、PLCopen MC 功能塊、試車頁面（第 12 章）
模擬	Simulate 選單：模擬模式（F7）、即時模擬（F8）、單一週期（Space）	模擬（F5）整個專案（含 I/O、計時器、軸、86HMI 模擬視窗），速度 x1 / x10（第 9 章）
除錯	線上監看	線上監看、寫入 / 強制、能流、趨勢圖、事件紀錄、交互參照、在專案中搜尋
專案檔	`data\myLD.ld` 單一檔	`data\86ld\` 中的文字檔（可用 git 版本控管，4.1 節）
其他	—	專案保護、AI 助理與 MCP、命令列工具、PLCopen XML 匯入（第 15 章）

LDmicro 1.1 的專案可以直接匯入：sketch 中有 `data\myLD.ld` 時，從 IDE 開啟 86LDmicro 會詢問是否匯入（2.5 節），也可用「檔案 > 匯入 LDmicro 1.x 程式或 PLCopen XML」（15.1 節）。原本的 `myLD.ld` 不會被修改。

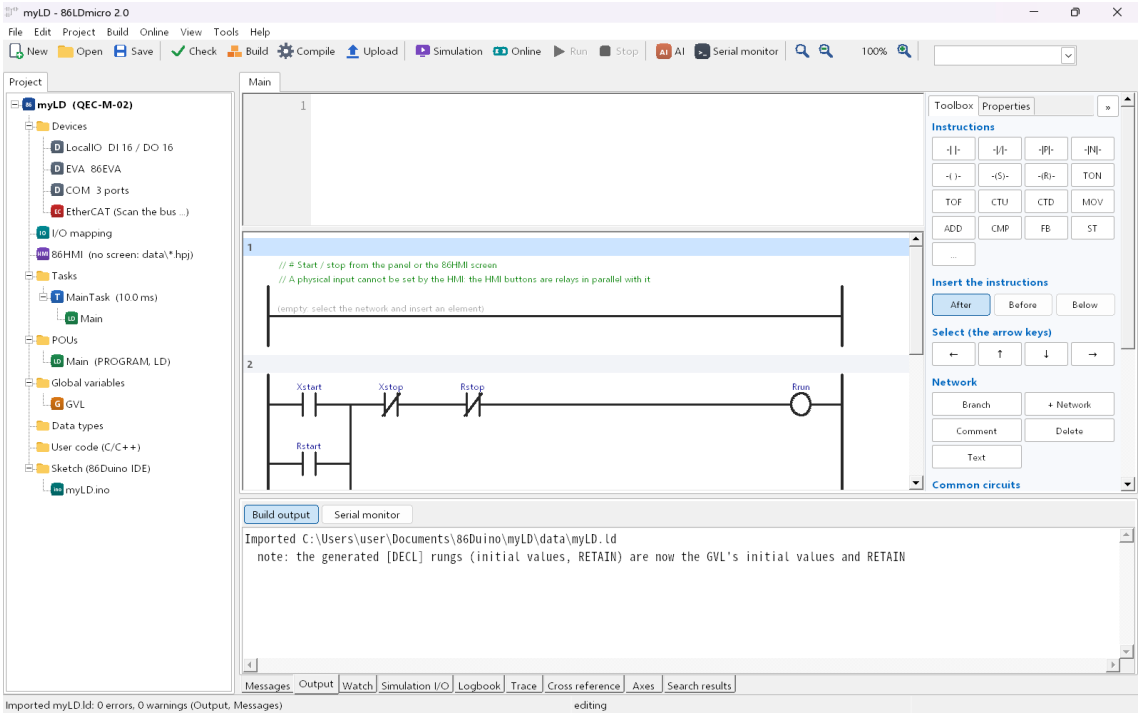


圖 1-5　匯入後的 1.1 專案：階梯圖變成 POU，變數放進 GVL

> ***提示**：1.x 的計時器、計數器等指令保留 1.x 的語意（以週期計數）；新程式建議改用 IEC 功能塊，例如 `CALL(t1, IN := @, PT := T#1s, Q => @)`（7.8 節）。
>
> 要改回 LDmicro 1.1，見 2.3 節最後的說明。

第 2 章　環境安裝

本章說明開始寫程式之前的準備：安裝 86Duino IDE 與 86LDmicro、接上控制器並找到 COM 埠、從 IDE 啟動 86LDmicro，以及「選項」中的編譯環境與語言設定。

2.1　系統需求

項目	需求
作業系統	Windows 10 / 11（64 位元）
86Duino IDE	86Duino Coding 502 Preview（20261105），內含 DMP-ECAT EtherCAT 函式庫與新版 Motion86（9 軸）
86LDmicro	86LDmicro 2.0.0 釋出版（zip），解壓縮到 IDE 的 `tools\LDmicro\tool`（2.3 節）
控制器	QEC-M-02（其他 QEC 板子在專案設定中選擇，4.2 節）
連線	可傳資料的 USB 線（上傳、線上監看、EtherCAT Configurator 共用）；Configurator 走網路時另需網路線
選用	Microsoft Edge（內嵌 EtherCAT Configurator 畫面）、Python 3（AI 的 MCP 功能，15.4 節）

*__注意__：請使用 502 Preview **20261105** 版的 86Duino IDE。更舊的 IDE 缺少 CoE 物件字典用到的 EtherCAT 函式，或只有舊版 Motion86，建置 EtherCAT 或運動控制專案時可能編譯失敗；只把新版 Motion86 複製到舊版 IDE 也會編譯失敗。

2.2 安裝 86Duino Coding 502

86LDmicro 使用 86Duino IDE 的編譯器（arduino-builder）、函式庫與上傳工具（v86dude），所以要先安裝 IDE。

1. 從 QEC 官方網站取得 86Duino Coding 502 Preview（20261105）。

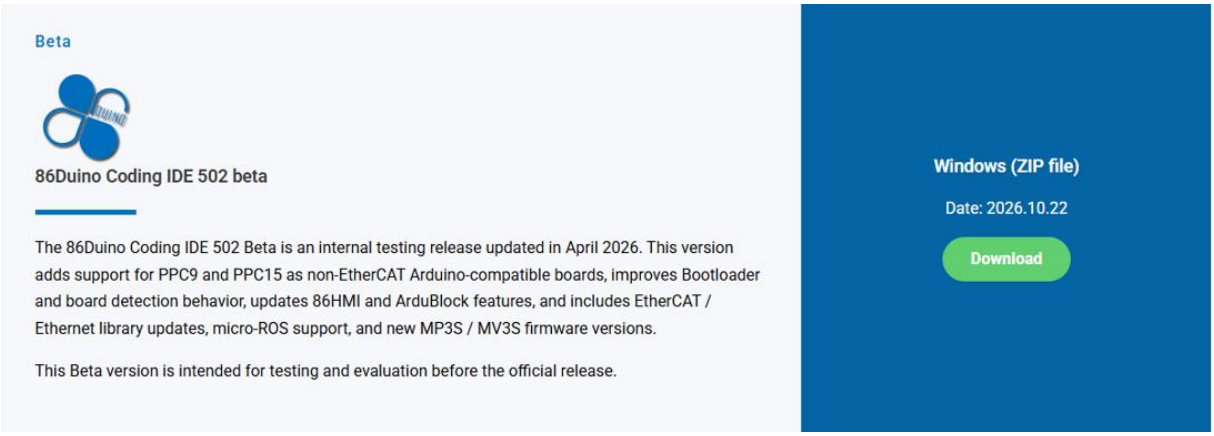


圖 2-1　86Duino IDE 502 於 QEC 網站下載連結

2. 將檔案解壓縮到本機的資料夾，例如 `C:\86Duino_Coding_502_Preview_20261105`。完成後資料夾中有 `86duino.exe`、`hardware`、`tools` 等。本手冊稱這個資料夾為 `<IDE 資料夾>`。
3. 雙擊 86duino.exe 開啟 IDE。圖 2-2　86Duino IDE 資料夾：86duino.exe、arduino-builder.exe、drivers、hardware、tools

名稱	修改日期	類型	大小
drivers	2026/11/16 下午 02:00	檔案資料夾	
examples	2026/11/16 下午 02:00	檔案資料夾	
hardware	2026/11/16 下午 02:01	檔案資料夾	
hermes	2026/11/16 下午 02:06	檔案資料夾	
java	2026/11/16 下午 02:06	檔案資料夾	
lib	2026/11/16 下午 02:07	檔案資料夾	
libraries	2026/11/16 下午 02:07	檔案資料夾	
tools	2026/11/16 下午 02:07	檔案資料夾	
tools-builder	2026/11/16 下午 02:07	檔案資料夾	
86duino.exe	2026/11/16 下午 02:00	應用程式	69 KB
86duino_debug.exe	2026/11/16 下午 02:00	應用程式	67 KB
86duino-cli.exe	2026/11/16 下午 02:00	應用程式	36,715 KB
arduino.l4j.ini	2026/11/16 下午 02:00	組態設定	1 KB
arduino_debug.l4j.ini	2026/11/16 下午 02:00	組態設定	1 KB
arduino-builder.exe	2026/11/16 下午 02:00	應用程式	23,401 KB
libusb0.dll	2026/11/16 下午 02:07	應用程式擴充	43 KB
msvcp100.dll	2026/11/16 下午 02:07	應用程式擴充	412 KB
msvcr100.dll	2026/11/16 下午 02:07	應用程式擴充	753 KB
README.md	2026/11/16 下午 02:07	MD 檔案	7 KB
revisions.txt	2026/11/16 下午 02:07	文字文件	97 KB

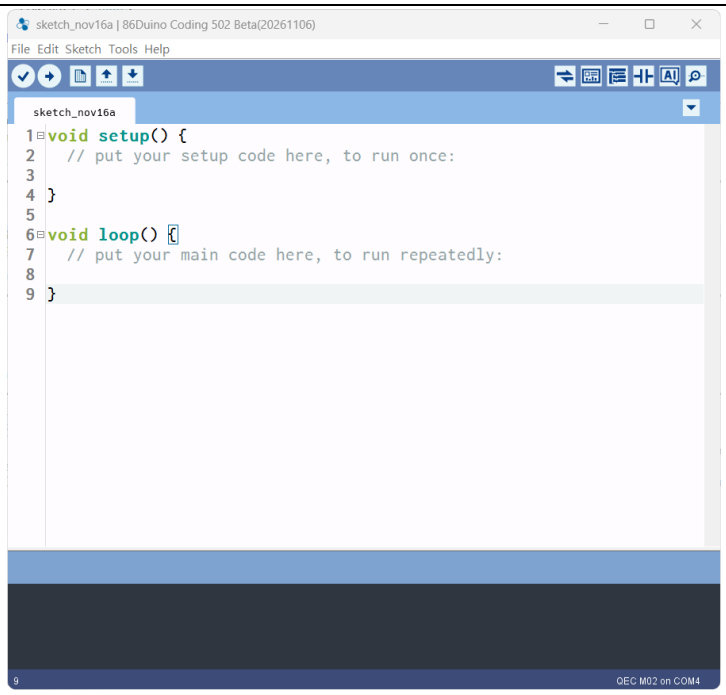


圖 2-3　86Duino Coding 502 開啟後的畫面（新的空白 sketch）

86LDmicro 的專案存在 sketch 資料夾中（`<sketch>\data\86ld`），所以 sketch 必須先存檔：用「檔案 > 開啟...」開啟已有的 sketch，或「檔案 > 新增」後先「另存新檔...」取一個名稱。IDE 內附的範例（examples）要先另存成自己的 sketch 才能開啟 86LDmicro。

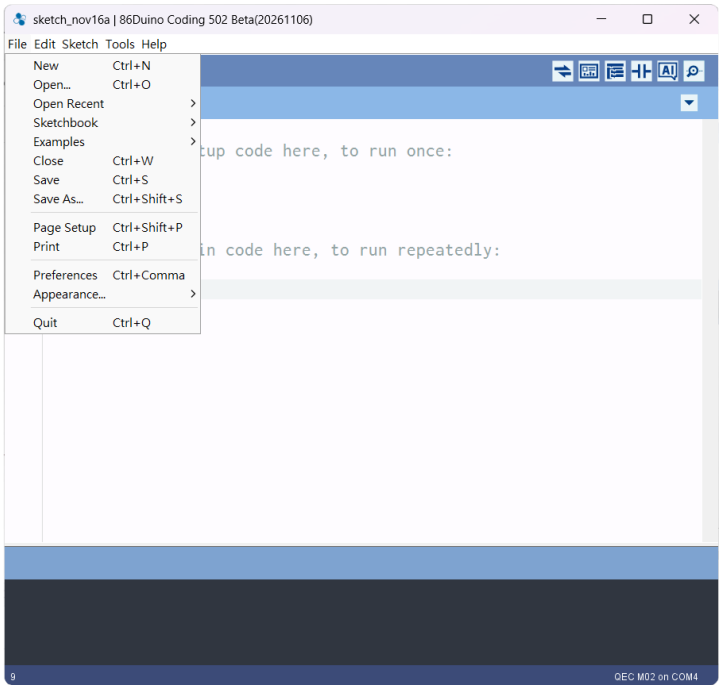


圖 2-4　86Duino IDE 的「檔案」選單：新增、開啟、另存新檔

2.3 安裝 86LDmicro 工具

86LDmicro 2.0.0 以一個 zip 檔釋出（例如 86LDmicro_2.0.0_v1r2.zip）。zip 的最上層是 tool 資料夾，它的內容要放在 <IDE 資料夾>\tools\LDmicro\tool。若你的 86Duino IDE 已內附 86LDmicro 2.0.0（<IDE 資料夾>\tools\LDmicro\tool\VERSION.txt 的內容為 2.0.0 2026-11-11），可以略過本節。

```
<IDE 資料夾>\tools\LDmicro\tool\
    LDmicro.jar          IDE 的「工具 > 86LDmicro」與 IDE 工具列按鈕
    VERSION.txt          版本與日期：2.0.0 2026-11-11（回報問題時請附上）
    bin\                 86LDmicro.exe（主程式）、86ldmicro-cli.exe（命令列，15.5 節）
    configurator\        EtherCAT Configurator（第 11 章）
    docs\                離線文件：USER_GUIDE.md、USER_CODE_API.md、ETHERCAT_MOTION.md、ai\、LICENSE.txt
    hmisim\              86HMI 模擬視窗（13.5 節）
    mcp\                 AI 助理用的 MCP server（15.4 節）
```

2.3.1 安裝步驟

1. 關閉 86Duino IDE。
2. （建議）把 <IDE 資料夾>\tools\LDmicro 整個資料夾複製一份備份，日後要改回 LDmicro 1.1 時使用。
3. 在下載的 zip 檔上按右鍵 > 內容，在「一般」分頁勾選「解除封鎖」（有這個選項時），按 確定，解壓縮出來的執行檔才不會被 Windows 封鎖。
4. 把 zip 解壓縮到 <IDE 資料夾>\tools\LDmicro：zip 裡的 tool 資料夾會成為 <IDE 資料夾>\tools\LDmicro\tool；出現同名檔案時選「取代目的地中的檔案」。不要多出一層資料夾（錯誤例：tools\LDmicro\86LDmicro_2.0.0_v1r2\tool）。
5. 確認 <IDE 資料夾>\tools\LDmicro\tool 中有上面列出的 LDmicro.jar、VERSION.txt、bin、configurator、docs、hmisim、mcp，且 VERSION.txt 的內容為 2.0.0 2026-11-11。
6. 重新開啟 86Duino IDE：「工具」選單出現 86LDmicro，即安裝完成（2.5 節）。

2.3.2 改回 LDmicro 1.1

關閉 86Duino IDE，把 <IDE 資料夾>\tools\LDmicro 換回安裝前備份的資料夾，再重新開啟 IDE。86LDmicro 2.0 的專案存在 sketch 的 data\86ld，匯入時也不會修改 1.1 的 data\myLD.ld，所以同一個 sketch 的兩種專案可以並存。

2.4 硬體連接與 COM 埠

電腦與 QEC-M-02 以 USB 線連接；上傳、線上監看與序列埠監看都使用這條線在電腦上的 COM 埠。

連線	從	到	用途
USB	電腦	QEC-M-02 的 USB 埠	上傳程式、線上監看、序列埠監看、Configurator（USB 模式）
EtherCAT	QEC-M-02 的 EtherCAT 埠	第一台從站 IN，之後 OUT → 下一台 IN	I/O 模組（例如 QEC-R11CFFG）、CiA 402 驅動器（例如 QEC-R11MP3S）
本機 I/O	QEC-M-02 端子台	按鈕、感測器、指示燈、繼電器	`LocalIO` 的 DI / DO（例如 X00、Y00）
LAN（選用）	電腦	QEC-M-02 的 LAN 埠	Configurator 網路模式（預設 192.168.100.1）、Modbus TCP
RS-232 / 485（選用）	外部設備	COM1 ~ COM3	Modbus RTU、使用者程式的 Serial1 ~ 3

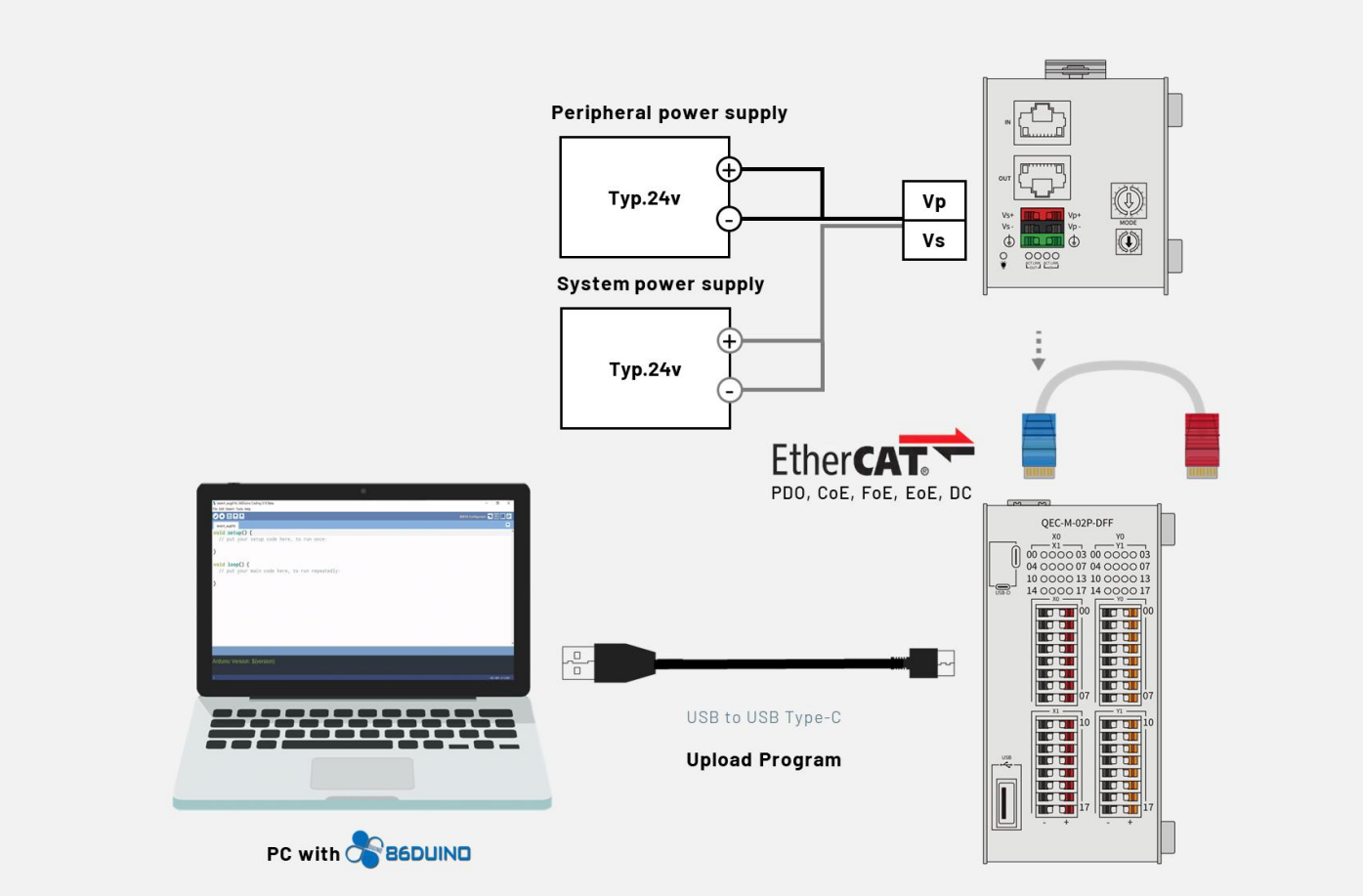


圖 2-5　QEC 與 PC 的硬體連接（示意圖）

*安全：**先斷電**再配線。電感性負載（繼電器、電磁閥）請加突波吸收元件。

緊急停止必須以硬體切斷驅動器電源或 STO，不可只靠 PLC 程式。

2.4.1 確認 COM 埠

1. QEC-M-02 送電，以 USB 線接到電腦。
2. 在 Windows「開始」按右鍵 > **裝置管理員**。
3. 展開「**連接埠 (COM 和 LPT)**」，找到新出現的「USB 序列裝置 (COMx)」，記下 COM 號碼（下圖為 COM4）。拔插 USB 線時會消失又出現的那一個就是控制器。

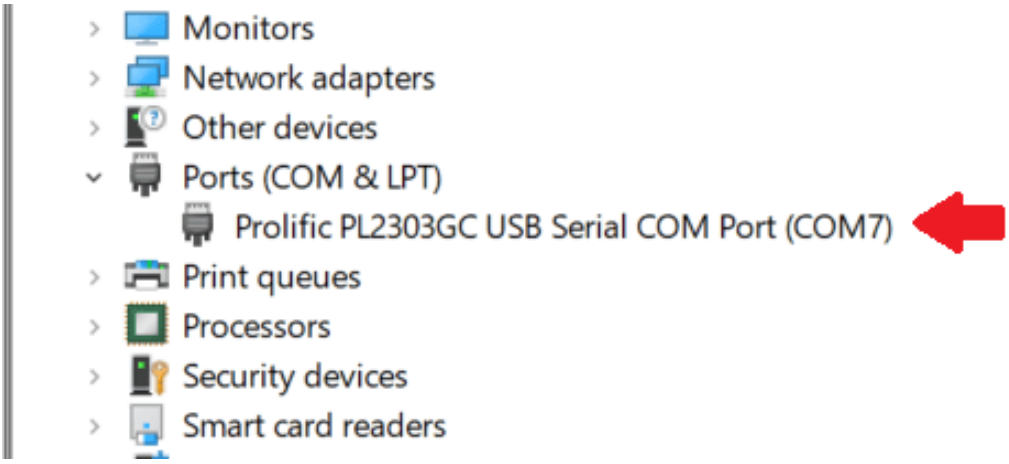


圖 2-6　裝置管理員：「連接埠 (COM 和 LPT)」下的 USB 序列裝置 (COM4)（示意圖）

*提示：看不到新的 COM 埠：換一條 USB 線（確定可傳資料）、換 USB 孔，或確認控制器已送電。

出現黃色驚嘆號：安裝 86Duino IDE 內附的驅動程式（`<IDE 資料夾>\drivers`），再拔插一次。

2.4.2 在 IDE 選擇開發板與序列埠

1. 在 86Duino IDE 按「**工具 > 開發板**」，選 **QEC M02**。

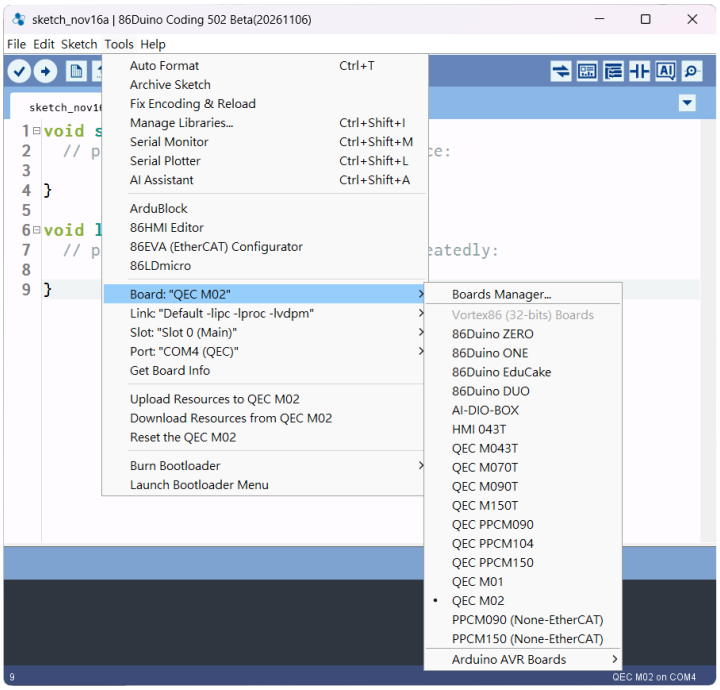


圖 2-7　「工具 > 開發板」選 QEC M02；同一個選單中有「86LDmicro」

2. 按「**工具 > 序列埠**」，選上面查到的 COM 埠（例如 COM4）。IDE 右下角顯示「QEC M02 於 COM4」。

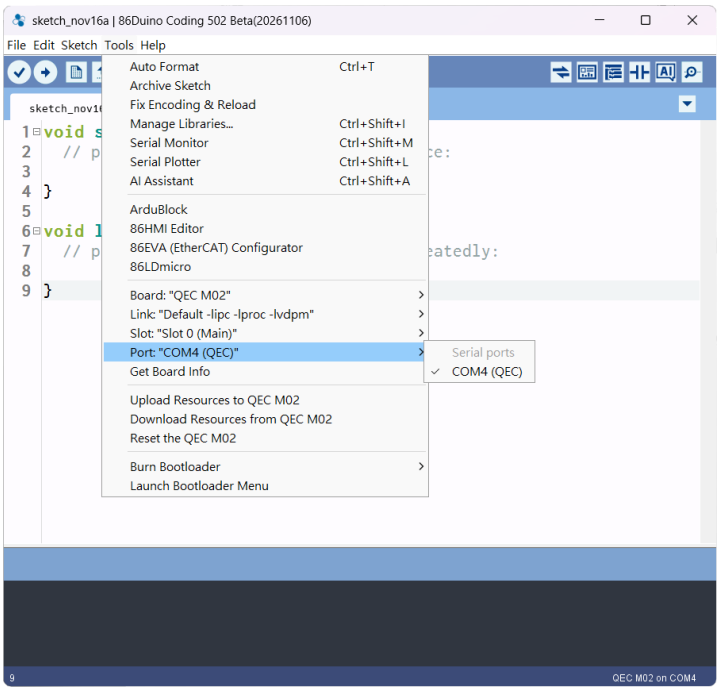


圖 2-8　「工具 > 序列埠」選 COM4（示意圖）

從 IDE 啟動 86LDmicro 時，會沿用 IDE 選的開發板與序列埠（2.5 節）；在 86LDmicro 中也可以從工具列右側的序列埠方塊改選（3.3 節）。

#	第一次連線檢查	如何確認
1	控制器已送電	電源燈亮
2	電腦認得 COM 埠	裝置管理員出現 USB 序列裝置 (COMx)
3	IDE 板子與埠正確	IDE 視窗右下角顯示「QEC M02 於 COMx」
4	沒有其他程式占用 COM 埠	關閉 IDE 的序列埠監控視窗、其他終端機程式
5	86LDmicro 可上傳、可連線	第 5 章快速入門的程式上傳成功，Ctrl+F5 後狀態列顯示「線上 COMx：RUN」

2.5 啟動 86LDmicro

建議從 86Duino IDE 啟動：86LDmicro 會開啟（或建立）該 sketch 的專案，並自動使用這個 IDE 的編譯器與 IDE 選的序列埠。

2.5.1 從 86Duino IDE 啟動（建議）

1. 在 IDE 開啟（或新增並存檔）一個 sketch。
2. 確認「工具 > 開發板」為 QEC 板子、「工具 > 序列埠」為正確的 COM 埠（2.4 節）。
3. 按「**工具 > 86LDmicro**」，或 IDE 工具列右側的 86LDmicro 按鈕（滑鼠停留時顯示「86LDmicro」）。

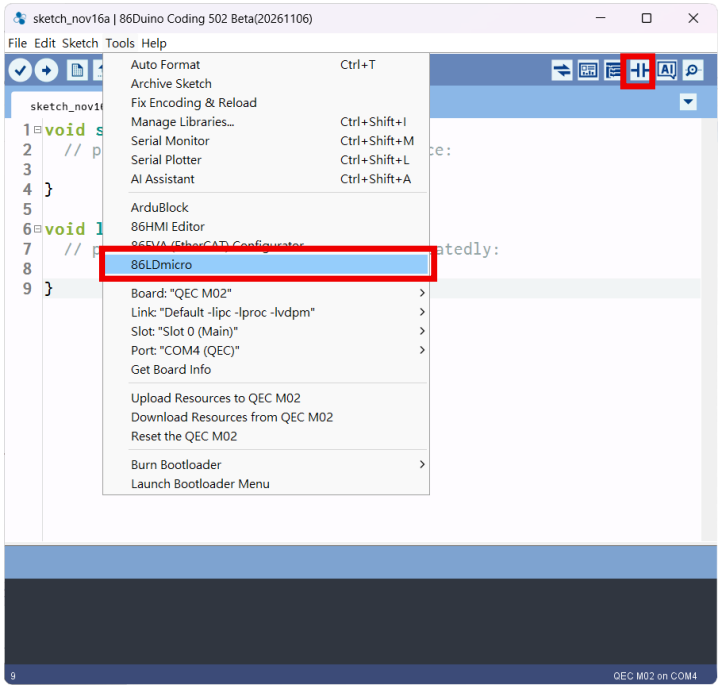
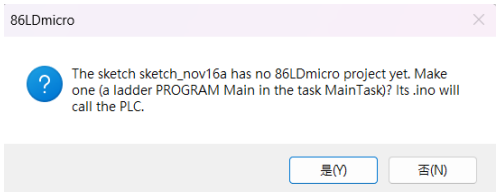
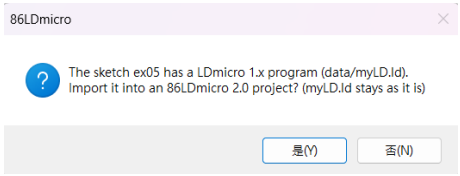
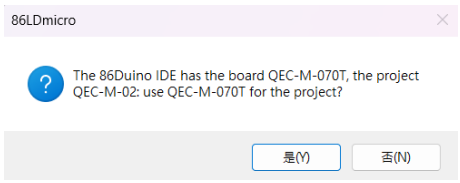


圖 2-9　IDE 工具列和右側的 86LDmicro 按鈕

86LDmicro 依 sketch 的狀態自動處理：

sketch 的狀態	86LDmicro 的動作	86LDmicro 視窗
沒有專案	詢問「sketch … 還沒有 86LDmicro 專案。要建立嗎？」，按 **是** 建立新專案（階梯圖 PROGRAM `Main`，在工作 `MainTask` 中執行）	 圖 2-10　sketch 還沒有專案時的詢問

sketch 的狀態	86LDmicro 的動作	86LDmicro 視窗
有 LDmicro 1.1 的 `data\myLD.ld`	詢問是否匯入成 86LDmicro 2.0 專案，按 **是** 匯入（`myLD.ld` 保持不變，15.1 節）	 圖 2-11　sketch 有 LDmicro 1.1 程式（data/myLD.ld）時的詢問
已有 2.0 專案（`data\86ld\project.86ld`）	直接開啟；IDE 選的板子與專案不同時，詢問專案是否改用 IDE 的板子	 圖 2-12　IDE 選的板子與專案不同時的詢問
同一個 sketch 已開著 86LDmicro	把那個視窗帶到最前面	

建立新專案後的 86LDmicro 主視窗如下：專案樹中已有工作 `MainTask`（10 ms）、階梯圖 `Main` 與全域變數表 `GVL`，並自動開啟 `Main`。各區域的說明見第 3 章。

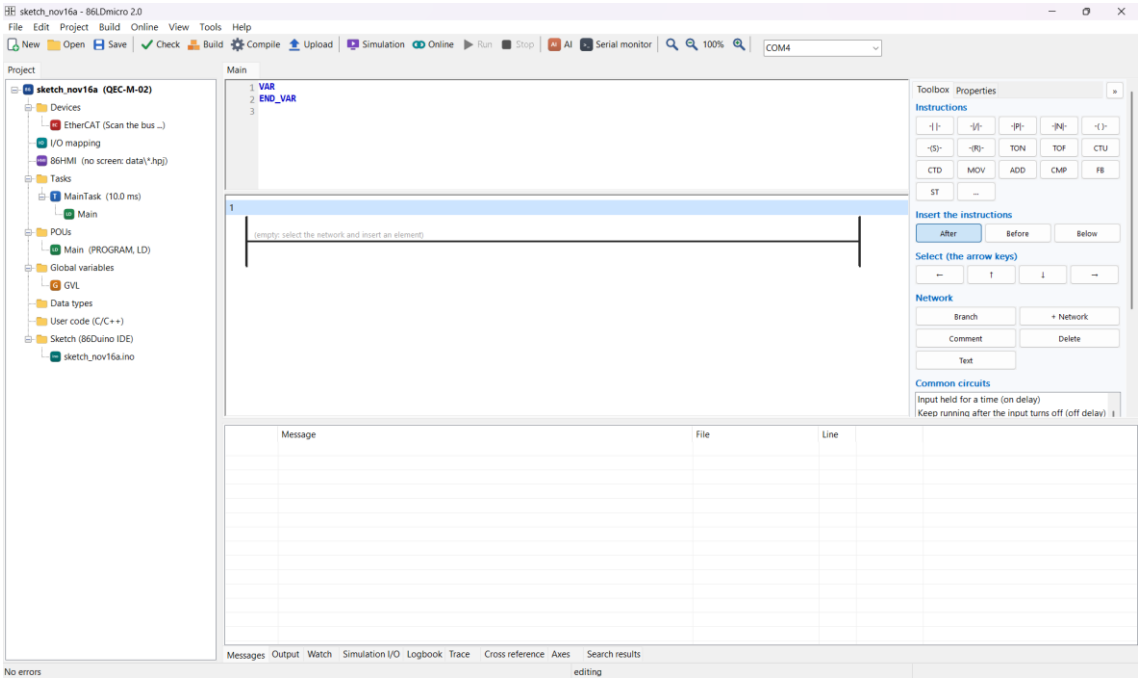


圖 2-13　從 IDE 建立新專案後的 86LDmicro：MainTask、Main、GVL 都已建立

- `.ino` 會被加上 `#include "plc86.h"`、`plc86_setup()`、`plc86_loop()`（1.1 的 `setupPlc()` / `loopPlc()` 會被換掉），其他程式碼不動。

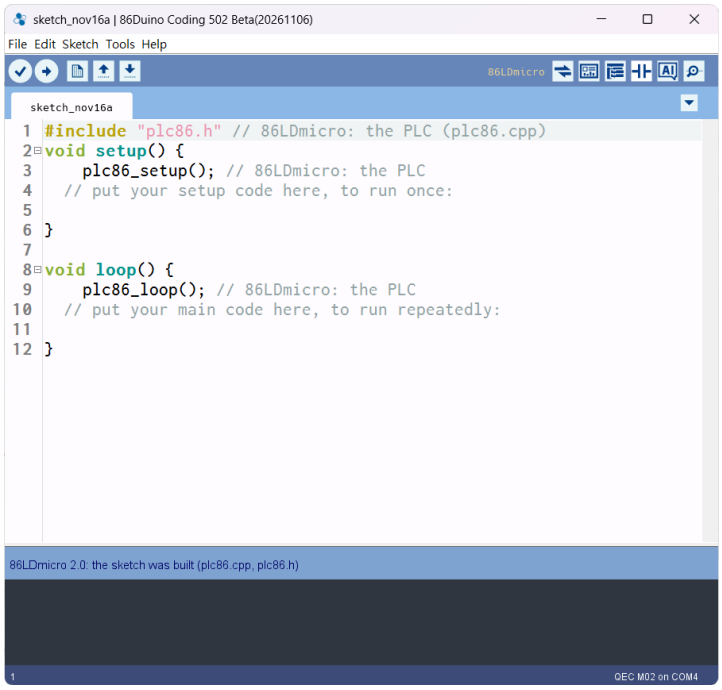


圖 2-14　.ino 加上 86LDmicro 的程式碼。

- 建置後 86LDmicro 通知 IDE 重新載入 sketch，IDE 會出現 `plc86.cpp` 等分頁；之後也可以直接在 IDE 編譯、上傳。不要手動修改這些自動產生的檔案（每次建置都會重寫）。

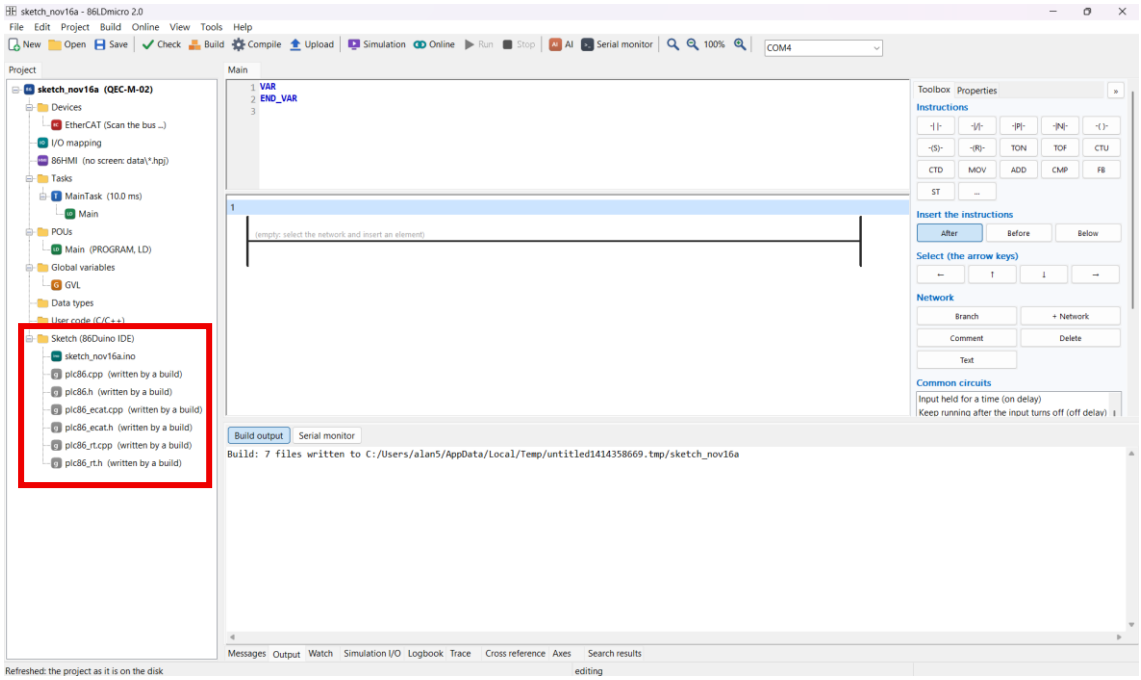


圖 2-15　86LDmicro 建置後產生 plc86 等檔案

建置後，若 86LDmicro 中 Sketch (86Duino IDE) 並未及時出現 `plc86.cpp` 等分頁，可以在 View 工具列中，按 Refresh (Ctrl + R) 選項。

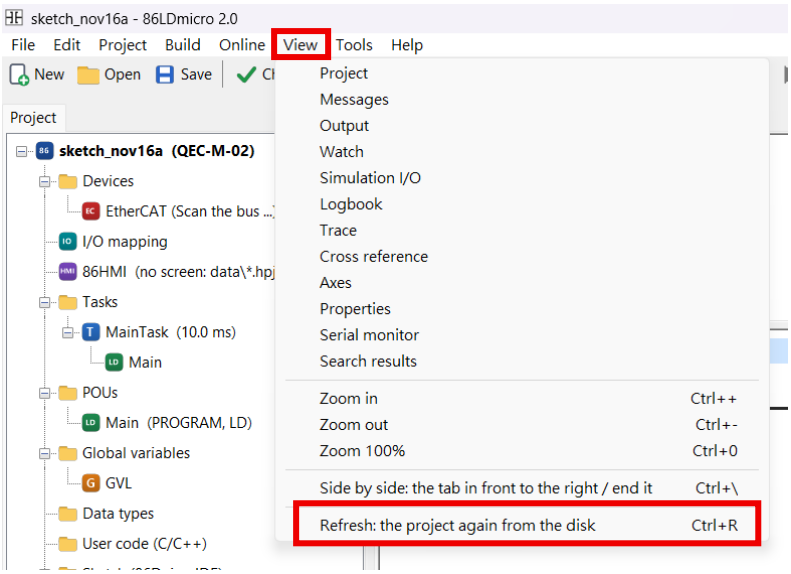


圖 2-16　按 Refresh (Ctrl+R) 後，Sketch 中出現 plc86.cpp 等檔案

- 開發板不是 QEC 板子時，IDE 下方狀態列顯示「86LDmicro 2.0: choose a QEC board (Tools > Board)」，86LDmicro 不會啟動；sketch 位於 IDE 的範例資料夾時，顯示「save the example as a sketch of your own first (File > Save As)」。

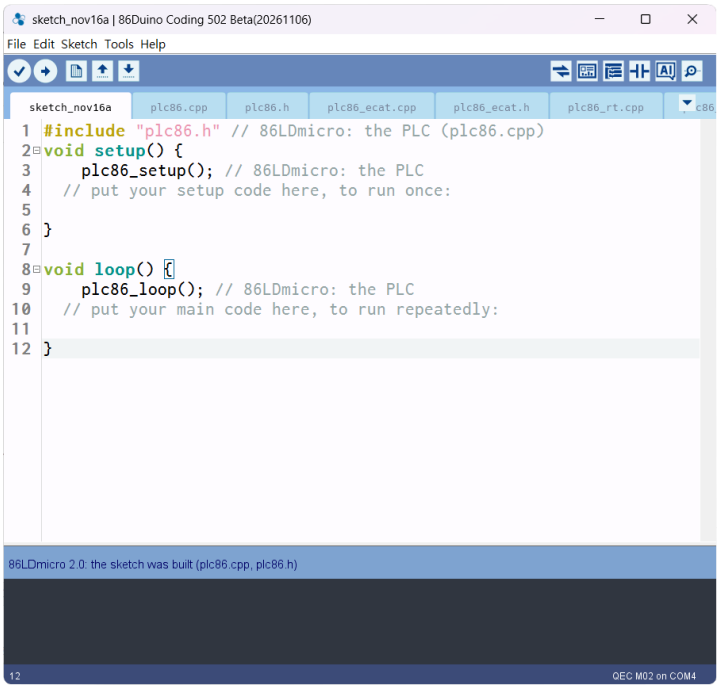


圖 2-17　建置後的 86Duino IDE：plc86.cpp、plc86.h、plc86_ecat.cpp 等分頁

***注意：**IDE 中尚未存檔的 `.ino` 修改，會在 86LDmicro 建置後被重新載入覆蓋。請先在 IDE 存檔，或改在 86LDmicro 的專案樹「Sketch (86Duino IDE) 」中編輯 `.ino` (3.4 節) 。

2.5.2 單獨啟動

也可以直接執行 `<IDE 資料夾> \tools\LDmicro\tool\bin\86LDmicro.exe`。

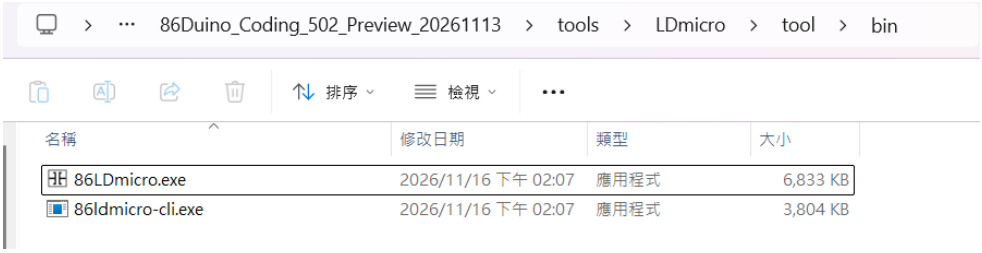


圖 2-18　單獨啟動、尚未開啟專案的 86LDmicro 路徑

單獨啟動時會自動開啟上次開啟的專案；沒有時專案樹顯示「沒有專案：檔案 > 新增、開啟或匯入」。

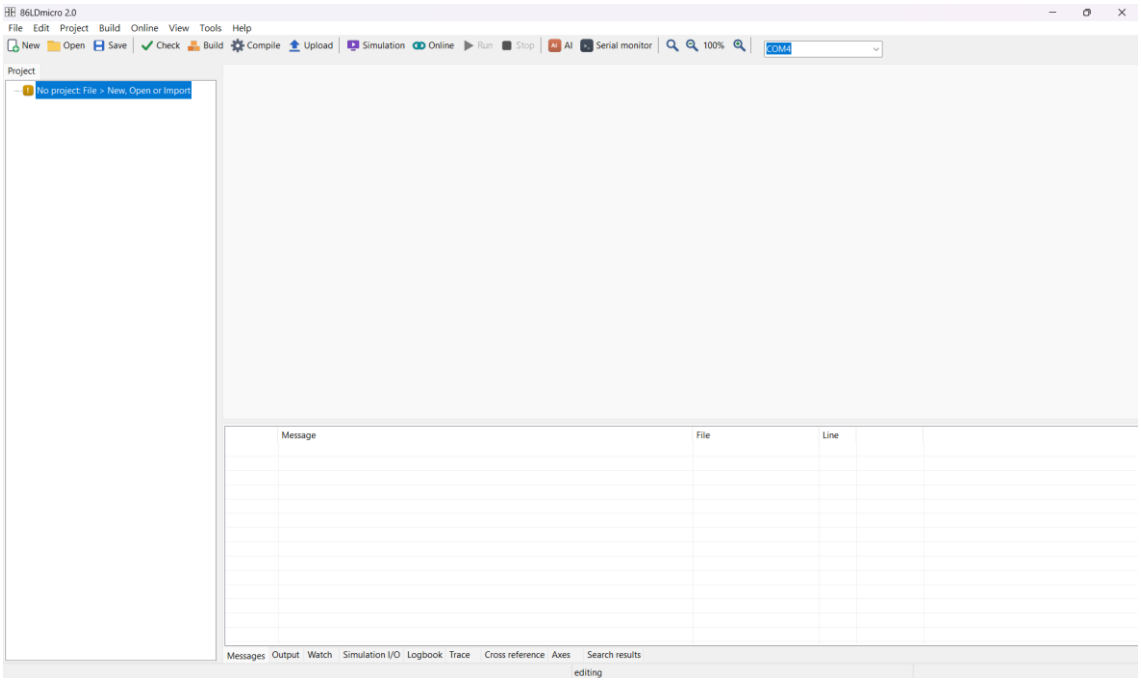
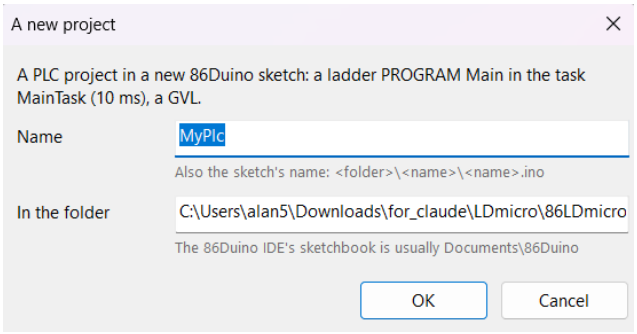


圖 2-19　單獨啟動、尚未開啟專案的 86LDmicro

要做的事	操作
建立新專案	「檔案 > 新增專案...」（Ctrl+N）：輸入名稱（也是 sketch 的名稱）與位置（預設 文件 `\86Duino`，即 IDE 的 sketchbook），按 **確定**。建立 `<位置>\<名稱>\<名稱>.ino` 與專案，內含階梯圖 `Main`、工作 `MainTask`（10 ms）與 `GVL`  圖 2-20　「檔案 > 新增專案...」：名稱也是 sketch 的名稱
開啟專案	「檔案 > 開啟專案...」（Ctrl+O）：選 `data\86ld\project.86ld`，或 sketch 的 `.ino`
匯入	「檔案 > 匯入 LDmicro 1.x 程式或 PLCopen XML (.ld, .xml)...」，或把 `.ld` / `.xml` 檔拖放到 86LDmicro 視窗（第 15 章）
最近的專案	「檔案 > 最近的專案」列出最近開啟的 8 個專案

*__重要__：單獨啟動時，第一次使用請先到「工具 > 選項...」指定 86Duino IDE 資料夾（2.6 節），否則只能檢查、建置與模擬，不能編譯與上傳。

2.6 選項

「**工具 > 選項...**」設定這台電腦上 86LDmicro 的編譯環境、介面語言與上傳後的連線方式（專案本身的設定在「專案設定」，4.2 節）。專案設定頁的 **選項...** 按鈕也開啟同一個視窗。

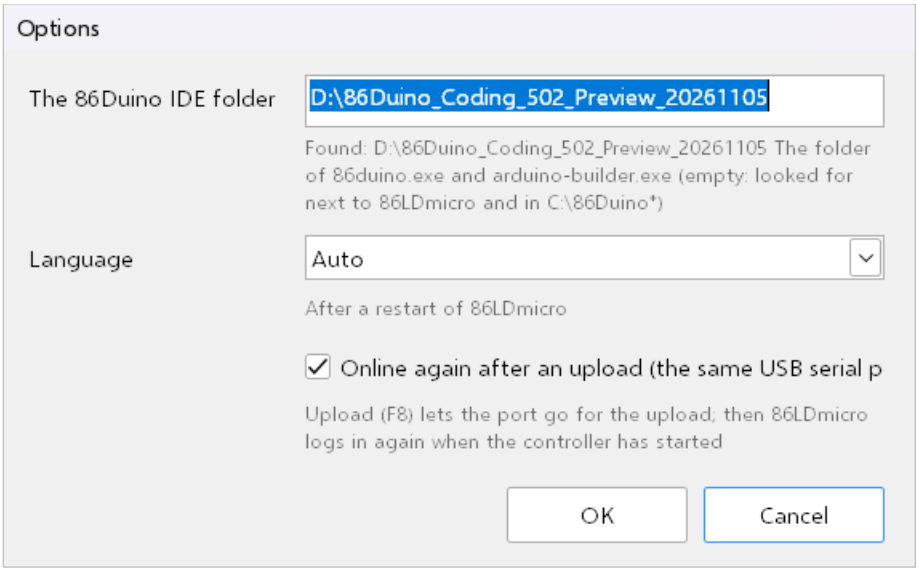


圖 2-21　選項：86Duino IDE 資料夾、語言、上傳後自動重新連線

欄位	說明
86Duino IDE 資料夾	編譯環境：`86duino.exe` 與 `arduino-builder.exe` 所在的資料夾。下方顯示「已找到：…」或「尚未找到。」；空白時在 86LDmicro 旁與 `C:\86Duino*` 中尋找。從 IDE 啟動時自動改成該 IDE 的資料夾。填入的資料夾不是 86Duino IDE 時，按確定會提示「此資料夾中沒有 86Duino IDE」
語言	Auto（依 Windows）、English、繁體中文、日本語；**重新啟動 86LDmicro 後生效**
上傳後自動重新連線（同一個 USB 序列埠，同 LDmicro 1.1）	預設勾選。上傳（F8）時先釋放序列埠，控制器啟動後 86LDmicro 自動重新登入線上監看

2.6.1 編譯環境的選擇

- **電腦上有多個 86Duino IDE 時**：從哪一個 IDE 按「工具 > 86LDmicro」，就使用那一個 IDE 的編譯器與函式庫；單獨啟動時使用「選項」指定的資料夾。
- **控制器（板子）**：在專案設定選擇（4.2 節），86LDmicro 依此找出 IDE 中的 FQBN（例如 `86duino:x86:86DuinoQECM02`）；特殊需求可在專案設定的「FQBN（選填）」直接指定。
- **函式庫**：使用該 IDE `hardware\86duino\x86\libraries` 中的函式庫（Ethercat、Motion86、Modbus……）。EtherCAT 與 Motion86 專案請使用 502 Preview 20261105 內附的函式庫。
- 找不到 IDE 時，編譯或上傳會顯示「找不到 86Duino IDE：請在 工具 > 選項 設定 86Duino IDE 資料夾」。

2.6.2 介面語言

在「語言」選擇後重新啟動 86LDmicro 即可切換。

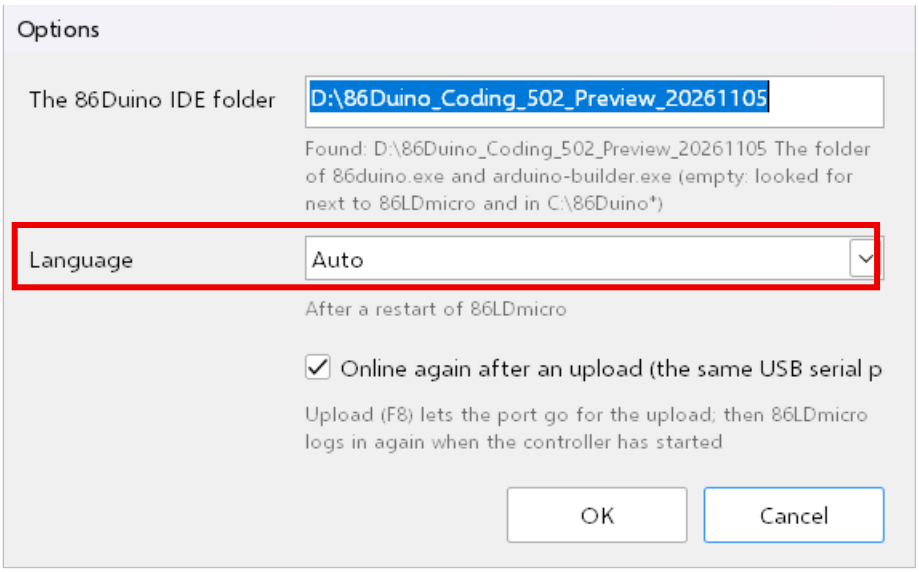


圖 2-22　選項：86Duino IDE 資料夾、語言、上傳後自動重新連線

三種語言的選單位置完全相同；本手冊的截圖使用英文介面。

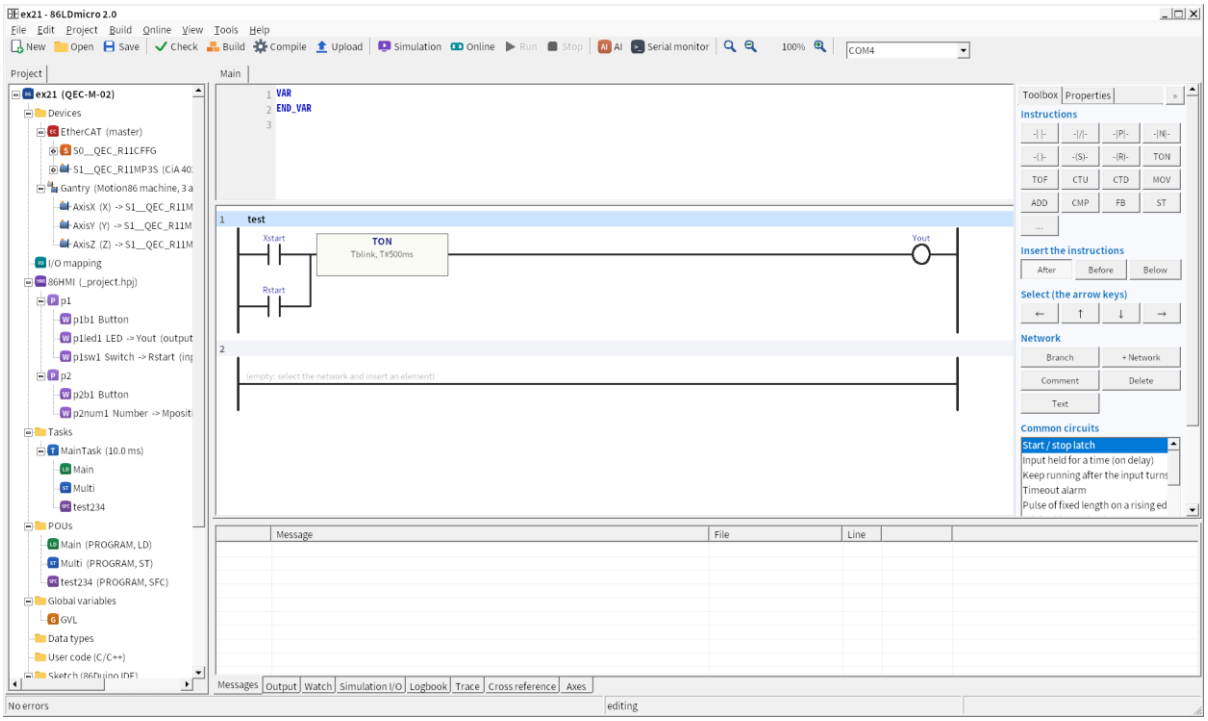


圖 2-23　語言選 English 時的介面（選單位置與繁體中文相同）

第 3 章　介面導覽

本章逐一介紹 86LDmicro 的主視窗：每個選單的每個項目、工具列、專案樹、POU 編輯器與右側欄、下方的工具視窗，以及分頁、並排、縮放與尋找。各功能的詳細操作在後面的章節，表中以節號標示。

3.1　主視窗與狀態列

主視窗由上而下分成選單列、工具列（右側為序列埠方塊），中間左邊是專案樹、中間是文件區（每個 POU 或設定頁一個分頁）、右邊是 POU 的右側欄，下方是工具視窗，最下面是狀態列。

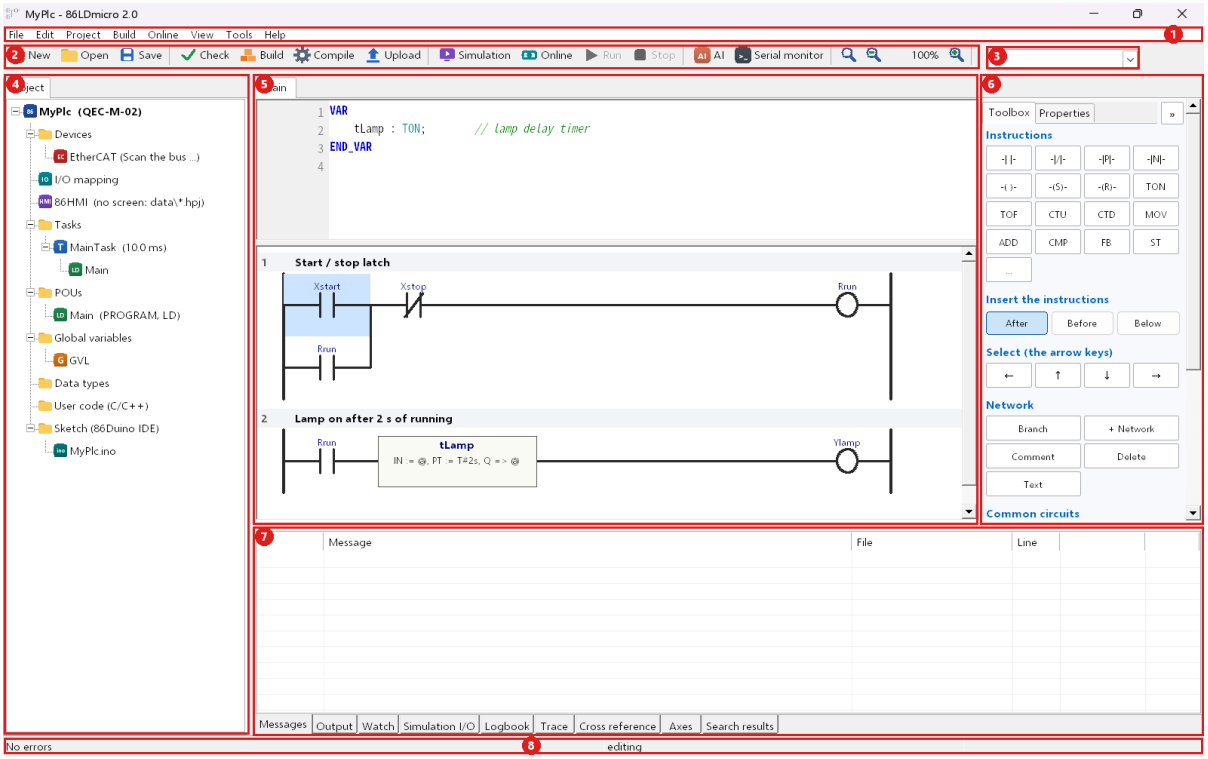


圖 3-1　86LDmicro 主視窗的各區域

編號	區域	說明
1	選單列	檔案、編輯、專案、建置、線上、檢視、工具、說明（3.2 節）
2	工具列	常用指令（3.3 節）；視窗變窄時自動改為只有圖示或分成兩列
3	序列埠	上傳、線上監看、序列埠監看與 USB 版 Configurator 使用的 COM 埠
4	專案樹	專案的所有內容；雙擊開啟、右鍵出現該節點的專屬選單（3.4 節）
5	文件區	每個 POU、設定頁一個分頁；可左右並排（3.7 節）
6	右側欄	LD、ST、SFC 分頁才出現：「工具箱」與「屬性」（3.5 節）
7	工具視窗	訊息、輸出、監看、模擬 I/O、事件紀錄、趨勢圖、交互參照、軸（試車）、搜尋結果（3.6 節）
8	狀態列	訊息、模式、掃描資訊（見下表）

標題列顯示「專案名稱 - 86LDmicro 2.0」；專案有未儲存的修改時，名稱後面多一個 *。

狀態列由左到右分成三格：

位置	內容	例子
左	最近一次操作的結果、檢查的錯誤數	沒有錯誤 / 沒有錯誤，2 個警告 / 3 個錯誤，0 個警告 / 已儲存
中	目前的模式	編輯中、編譯中...、模擬：RUN、模擬：STOP、線上 COM4：RUN、線上 COM4：STOP
右	模擬或線上時的掃描資訊	模擬：「1520 次掃描，15 秒」；線上：「掃描 230 us（最大 410）」（第一個工作）

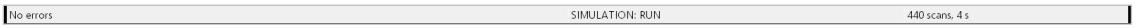


圖 3-2　模擬中的狀態列：沒有錯誤、模擬：RUN、掃描次數與時間

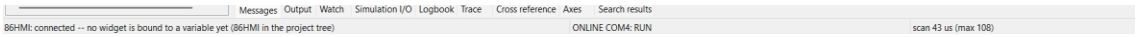


圖 3-3　線上時的狀態列：線上 COM4：RUN、掃描時間

3.2 主選單

主選單有八個選單：檔案、編輯、專案、建置、線上、檢視、工具、說明。選單名稱後括號中的字母可配合 Alt 使用，例如 Alt+F 開啟「檔案」。快捷鍵在任何分頁都有效；模擬或線上時不能使用的項目會變灰。以下各表依選單由上而下的順序列出所有項目（項目名稱省略括號中的字母）。

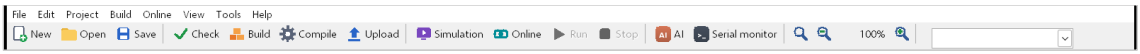


圖 3-4　選單列與工具列

3.2.1 檔案

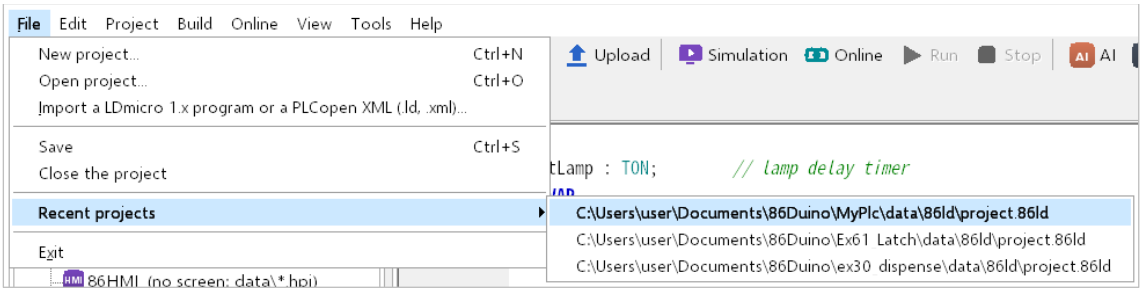


圖 3-5　「檔案」選單

項目	快捷鍵	用途
新增專案...	Ctrl+N	在新的 86Duino sketch 中建立專案：名稱、位置（2.5 節）
開啟專案...	Ctrl+O	開啟專案：選擇 `project.86ld` 或 sketch 的 `.ino`
匯入 LDmicro 1.x 程式或 PLCopen XML (.ld, .xml)...	—	把 LDmicro 1.x 的 `.ld` 或 PLCopen XML 匯入成新專案（15.1、15.2 節）；也可把檔案拖放到視窗
儲存	Ctrl+S	儲存專案的所有修改（含在 86LDmicro 中編輯的 sketch 檔）；狀態列顯示「已儲存」
關閉專案	—	關閉專案；有未儲存的修改時詢問「要儲存專案的變更嗎？」
最近的專案 ▸	—	最近開啟的 8 個專案，點一下即開啟；沒有時顯示「（無）」
結束	—	結束 86LDmicro（有未儲存的修改時先詢問）

3.2.2 編輯

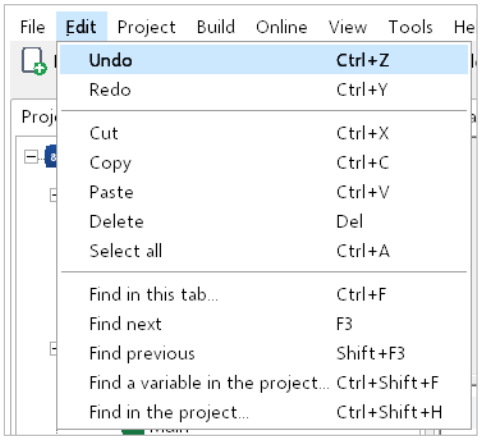


圖 3-6 「編輯」選單

項目	快捷鍵	用途
復原	Ctrl+Z	復原目前分頁的上一個修改
重做	Ctrl+Y	重做被復原的修改
剪下	Ctrl+X	程式碼：選取的文字；階梯圖：選取的元件，或沒有選元件時整個網路
複製	Ctrl+C	同上，複製
貼上	Ctrl+V	貼上；階梯圖中貼在選取的位置
刪除	Del	刪除選取的文字、元件或網路
全選	Ctrl+A	選取目前分頁的全部文字
在此分頁尋找...	Ctrl+F	在目前分頁上方開啟尋找列（3.7 節）
尋找下一個	F3	跳到下一個符合處
尋找上一個	Shift+F3	跳到上一個符合處
在專案中尋找變數...	Ctrl+Shift+F	輸入變數名稱，在「交互參照」列出它被讀、寫、宣告的位置（9.5 節）
在專案中搜尋...	Ctrl+Shift+H	在整個專案搜尋任意文字，結果列在「搜尋結果」（3.7、9.5 節）

3.2.3 專案

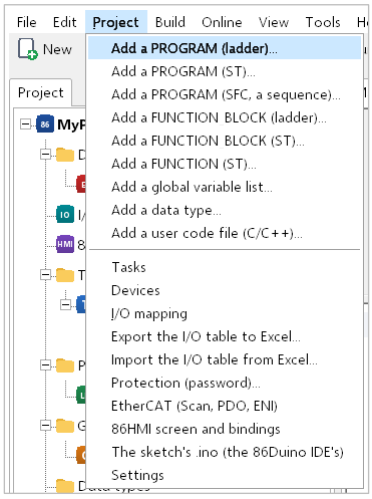


圖 3-7 「專案」選單

項目	快捷鍵	用途
新增 PROGRAM（階梯圖）...	—	新增階梯圖 PROGRAM；自動加入第一個工作（4.7 節）
新增 PROGRAM（ST）...	—	新增 ST PROGRAM（第 8 章）
新增 PROGRAM（SFC，順序流程圖）...	—	新增 SFC PROGRAM（8.4 節）
新增 FUNCTION_BLOCK（階梯圖）...	—	新增階梯圖功能塊
新增 FUNCTION_BLOCK（ST）...	—	新增 ST 功能塊（8.3 節）
新增 FUNCTION（ST）...	—	新增 ST 函式（8.3 節）
新增全域變數表...	—	新增一個 GVL（4.5 節）
新增資料型別...	—	新增一個資料型別檔（結構、列舉、陣列，4.5 節）
新增使用者程式檔（C/C++）...	—	新增 `.cpp86` 使用者程式（第 14 章）
工作 Task	—	開啟工作頁（4.3 節）
裝置	—	開啟裝置頁（4.4 節）
I/O 對應	—	開啟 I/O 對應頁（4.6 節）
匯出 I/O 表到 Excel...	—	把變數與 I/O 對應存成 .xlsx 或 .csv（4.6 節）
從 Excel 匯入 I/O 表...	—	從 .xlsx 或 .csv 宣告變數並設定 I/O 對應（4.6 節）
專案保護（密碼）...	—	以密碼把檔案設為唯讀或隱藏（15.6 節）
EtherCAT（掃描、PDO、ENI）	—	開啟 EtherCAT 頁（第 11 章）
86HMI 畫面與綁定	—	開啟 86HMI 頁（第 13 章）
Sketch 的 .ino（86Duino IDE 的主程式）	—	在 86LDmicro 中開啟並編輯 sketch 的 `.ino`（14.5 節）
設定	—	開啟專案設定頁（4.2 節）

3.2.4 建置

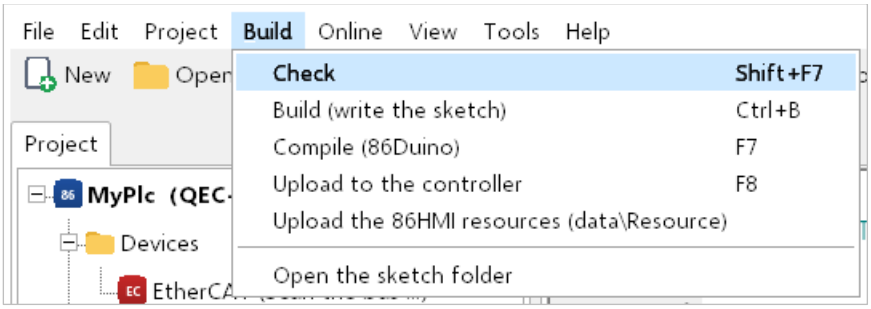


圖 3-8 「建置」選單

項目	快捷鍵	用途
檢查	Shift+F7	只編譯 PLC 程式並列出錯誤與警告（「訊息」視窗），不寫出檔案（10.1 節）
建置（寫入 sketch）	Ctrl+B	先儲存，再把 PLC 程式寫成 sketch 中的 `plc86.cpp`、`plc86.h` 等（10.1、10.2 節）
編譯（86Duino）	F7	建置後以 86Duino IDE 的編譯器編譯，過程顯示在「輸出」（不上傳）
上傳到控制器	F8	建置、編譯並上傳到工具列選的 COM 埠（10.1 節）
上傳 86HMI 資源（data\Resource）	—	只上傳 86HMI 的圖片、字型等資源，不重新編譯程式（10.6 節）
開啟 sketch 資料夾	—	在檔案總管開啟 sketch 資料夾

*提示：編譯或上傳進行中，「編譯」「上傳」「上傳 86HMI 資源」與「連線到控制器」會變灰，直到完成。

3.2.5 線上

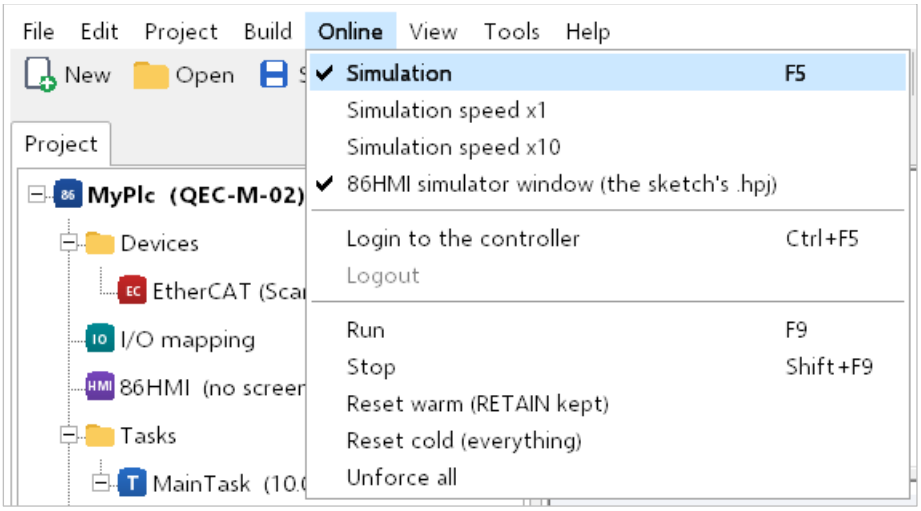


圖 3-9 「線上」選單（模擬中）

項目	快捷鍵	用途
模擬	F5	在電腦上執行整個專案；再按一次結束模擬。模擬中前面打勾（9.1 節）
模擬速度 x1	—	模擬以實際時間執行
模擬速度 x10	—	模擬時間以 10 倍速前進（計時器等依模擬時間動作）
86HMI 模擬視窗（sketch 的 .hpj）	—	勾選時，模擬會一起開啟 86HMI 模擬視窗（13.5 節）
連線到控制器	Ctrl+F5	經工具列的 COM 埠登入控制器，線上監看；線上時再按一次即中斷連線（10.4 節）
中斷連線	—	結束線上監看（只在線上時可用）
執行 RUN	F9	讓 PLC（模擬或控制器）開始執行
停止 STOP	Shift+F9	程式停止執行；輸入、輸出照常更新，輸出保持最後的值；所有 Motion86 機台停止（10.4 節）
暖重啟（保留 RETAIN）	—	詢問後重新開始：變數回到初值，RETAIN 變數保留
冷重啟（全部）	—	詢問後重新開始：所有變數（包含 RETAIN）回到初值
解除全部強制	—	解除所有強制中的變數（9.3 節）

「執行 RUN」到「解除全部強制」只在模擬或線上時可用，編輯中為灰色。

3.2.6 檢視

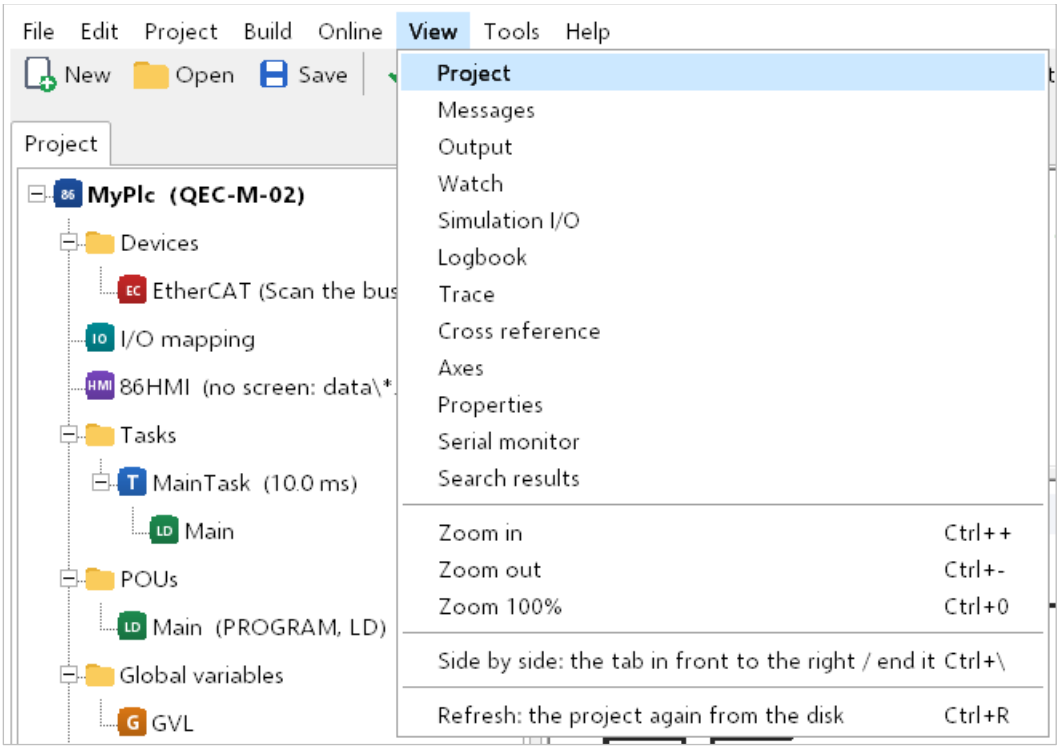


圖 3-10 「檢視」選單

項目	快捷鍵	用途
專案	—	顯示專案樹（3.4 節）
訊息	—	顯示「訊息」視窗：檢查與建置的錯誤、警告
輸出	—	顯示「輸出」視窗的建置輸出
監看	—	顯示「監看」視窗（9.3 節）
模擬 I/O	—	顯示「模擬 I/O」視窗（9.2 節）
事件紀錄	—	顯示「事件紀錄」（10.5 節）
趨勢圖	—	顯示「趨勢圖」（9.4 節）
交互參照	—	顯示「交互參照」（9.5 節）
軸（試車）	—	顯示「軸（試車）」視窗（12.5、12.6 節）
屬性	—	切到右側欄的「屬性」分頁（3.5 節）
序列埠監看	—	「輸出」視窗切到序列埠監看（10.3 節）
搜尋結果	—	顯示「搜尋結果」（3.7 節）
放大	Ctrl++	放大整個介面（3.7 節）
縮小	Ctrl+-	縮小整個介面
縮放 100%	Ctrl+0	回到 100%

項目	快捷鍵	用途
左右並排：目前分頁移到右側 / 結束並排	Ctrl+\	把前面的分頁移到右側並排；已並排時結束並排（3.7 節）
重新整理：從磁碟重新讀取專案	Ctrl+R	從磁碟重新讀取整個專案，並重新整理序列埠清單（見下方說明）

重新整理（Ctrl+R） 用在外部編輯器、命令列或 AI / MCP 改過專案檔之後：POU、GVL、資料型別、使用者程式、`project.86ld`、sketch 與 86HMI 全部重新讀取，並還原開啟的分頁，狀態列顯示「已重新整理：專案為磁碟上的最新內容」。有未儲存的修改時先詢問是否儲存（是：儲存後讀取；否：放棄修改；取消：不重新整理）。模擬或線上時會提示「請先停止模擬 / 離線」。

3.2.7 工具

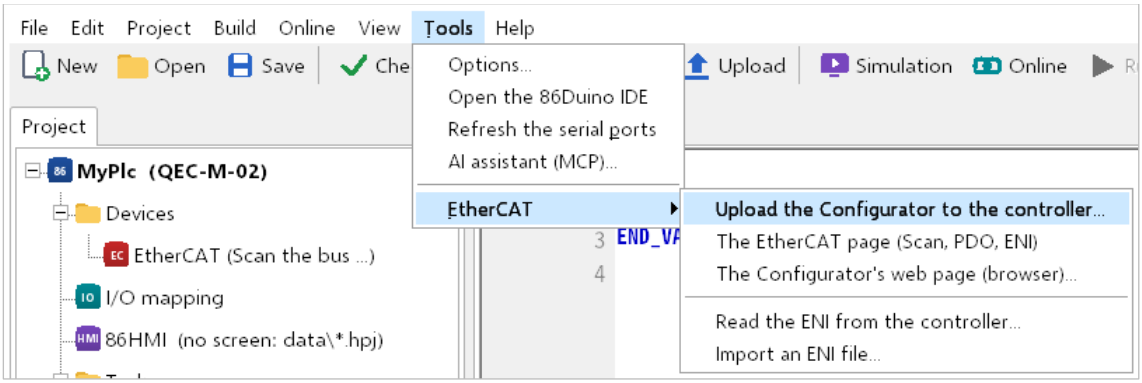


圖 3-11 「工具」選單與 EtherCAT 子選單

項目	快捷鍵	用途
選項...	—	86Duino IDE 資料夾、語言、上傳後自動重新連線（2.6 節）
開啟 86Duino IDE	—	開啟 86Duino IDE 並載入專案的 `.ino`
重新整理序列埠	—	重新列出工具列序列埠方塊中的 COM 埠（插上控制器後使用）
AI 助手（MCP）...	—	開啟「AI 助理」分頁：提問、插入程式碼、AI 服務設定與 Claude Desktop（MCP）設定（15.3、15.4 節）
EtherCAT ▸ 上傳 Configurator 到控制器...	—	把 EtherCAT Configurator 上傳到控制器（11.3 節）
EtherCAT ▸ EtherCAT 頁面（掃描、PDO、ENI）	—	開啟 EtherCAT 頁（與「專案 > EtherCAT」相同）
EtherCAT ▸ Configurator 網頁版（瀏覽器）...	—	在瀏覽器開啟 Configurator 網頁（網路模式）
EtherCAT ▸ 從控制器讀取 ENI...	—	從控制器讀回 ENI，建立從站與 PDO（11.4 節）
EtherCAT ▸ 匯入 ENI 檔...	—	從電腦上的 ENI 檔建立從站與 PDO（11.4 節）

3.2.8 說明

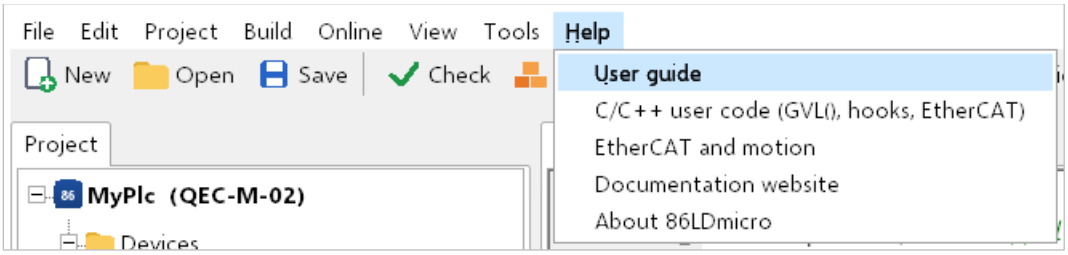


圖 3-12 「說明」選單

項目	快捷鍵	用途
操作說明	—	開啟線上文件網站
C/C++ 使用者程式（GVL()、掛勾、EtherCAT）	—	開啟線上文件網站
EtherCAT 與運動控制	—	開啟線上文件網站
說明文件網站	—	開啟線上文件網站 https://www.qec.tw/86duino/86ldmicro/
關於 86LDmicro	—	版本與版權資訊

前四項都在瀏覽器開啟 https://www.qec.tw/86duino/86ldmicro/ 。離線時可在 `<IDE 資料夾>\tools\LDmicro\tool\docs` 找到文件與使用手冊 PDF（日文版在 `docs\ja`）。

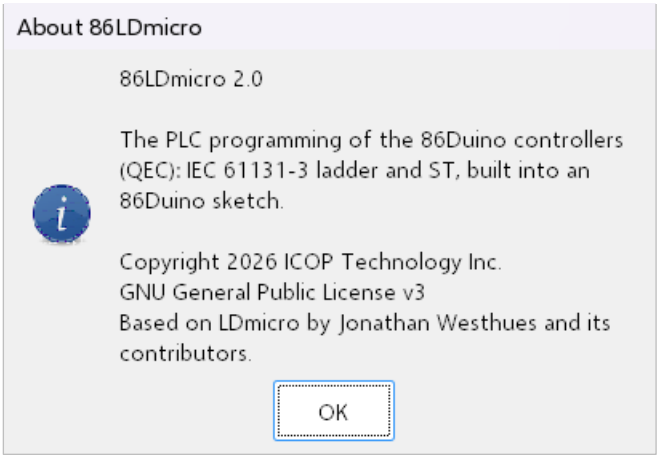


圖 3-13 「關於 86LDmicro」

3.3 工具列

工具列集中最常用的指令，右側是序列埠方塊。滑鼠停在按鈕上會顯示說明。



圖 3-14　工具列與序列埠方塊

按鈕	快捷鍵	功能
新增 / 開啟 / 儲存	Ctrl+N / Ctrl+O / Ctrl+S	新增專案、開啟專案、儲存
檢查	Shift+F7	只編譯 PLC 程式、列出錯誤
建置	Ctrl+B	寫出 sketch 檔案
編譯	F7	以 86Duino IDE 編譯
上傳	F8	編譯並上傳到控制器
模擬	F5	開始 / 結束模擬；模擬中按鈕保持按下
線上	Ctrl+F5	連線到控制器 / 中斷連線；線上時按鈕保持按下
RUN / STOP	F9 / Shift+F9	模擬或線上時執行 / 停止
AI	—	開啟「AI 助理」分頁（15.3 節）
序列埠監看	—	「輸出」視窗切到序列埠監看，看 Serial.print 的輸出（10.3 節）
放大鏡	Ctrl+F	在此分頁尋找
縮小 / 100% / 放大	Ctrl+- / Ctrl+0 / Ctrl++	縮放整個介面；中間顯示目前比例，按一下回到 100%
序列埠方塊	—	控制器的 COM 埠；可從清單選或直接輸入。空白時上傳或連線會先詢問「控制器的序列埠」

視窗變窄時，工具列依序改為：只有編譯、上傳、模擬、線上顯示文字，其他只顯示圖示；再窄則全部只顯示圖示（文字改成滑鼠停留說明），必要時分成兩列。

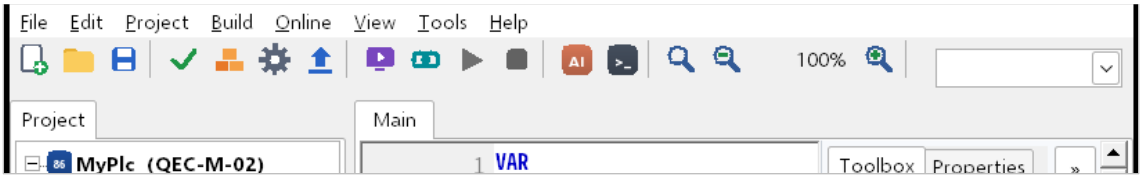


圖 3-15　視窗變窄時的工具列：只顯示圖示，分成兩列

3.4 專案樹

專案樹（左側「專案」窗格）列出專案的所有內容。**雙擊**節點開啟它的頁面或編輯器，**右鍵**出現該節點專屬的選單。第一次開啟專案時全部展開，之後記住展開狀態。

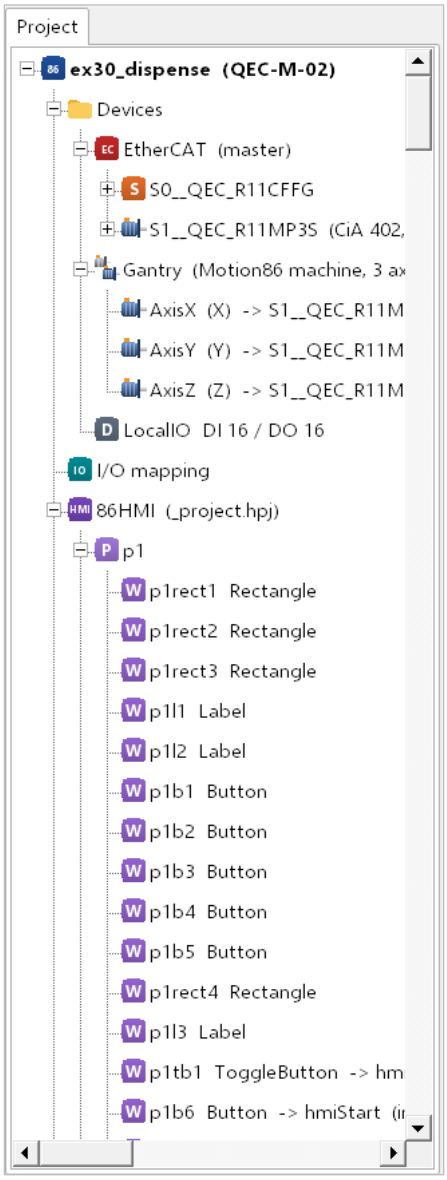


圖 3-16　專案樹（ex30_dispense）：裝置、I/O 對應、86HMI、工作、POU、全域變數、資料型別、使用者程式、Sketch

節點	內容	雙擊
專案名稱（控制器）	例如「MyPlc (QEC-M-02)」	開啟專案設定頁（4.2 節）
裝置	各裝置；EtherCAT 主站下為從站、驅動器與 PDO 項目；Motion86 機台下為軸；沒有 EtherCAT 時有「EtherCAT（掃描匯流排...）」	裝置頁（4.4 節）；從站開啟 SubDevice 頁、PDO 項目對應到變數、機台與軸開啟其頁面（第 11、12 章）
I/O 對應	變數 ↔ 通道	開啟 I/O 對應頁（4.6 節）
86HMI	`.hpj` 的頁面與元件；元件後面顯示綁定的變數；沒有畫面時顯示「無畫面：data*.hpj」	雙擊元件開啟綁定表單（第 13 章）

節點	內容	雙擊
工作 (Task)	每個工作（週期或類型）與其中依序執行的 PROGRAM	雙擊工作開啟它的設定（4.3 節）；雙擊程式開啟編輯器
POU	所有 PROGRAM / FUNCTION_BLOCK / FUNCTION，例如「Main (PROGRAM, LD)」；受保護時加註唯讀 / 隱藏	開啟編輯器
全域變數、資料型別	GVL、資料型別檔	開啟編輯器（4.5 節）
使用者程式（C/C++）	`.cpp86` 檔	開啟編輯器（第 14 章）
Sketch（86Duino IDE）	sketch 的 `.ino`、`.cpp`、`.h`；`plc86*` 標示「建置時產生」	在 86LDmicro 內開啟編輯

檔案有未儲存的修改時，節點名稱後面多一個 *。主要節點的右鍵選單：

節點	右鍵選單
專案名稱	專案設定... / 建置（寫出 sketch） / 編譯（86Duino） / 上傳到控制器 / 模擬 / 連線到控制器 / 開啟 sketch 資料夾 / 開啟 86Duino IDE / 關閉專案
裝置	裝置頁面 / 新增裝置 ▸（localio、eva、com、modbus、ethernet、modbus-server、ethercat、motion86）
一個裝置	編輯... / 移除 / 裝置頁面 / I/O 對應（Motion86 機台另有「軸面板（試車）」）
EtherCAT 主站	EtherCAT 頁面 / 上傳 Configurator 到控制器... / 開啟 Configurator（RD 網頁介面） / 從控制器讀取 ENI... / 匯入 ENI 檔... / 編輯... / 移除
PDO 項目	對應到變數... / 取消對應 / 加入監看 / 交互參照 / EtherCAT 頁面
I/O 對應	開啟 / 新增對應...
工作 (Task)	新增工作... / 工作頁面
一個工作	編輯工作... / 新增 PROGRAM（階梯圖）... / 新增 PROGRAM（ST）... / 移除工作 / 在這個工作執行 PROGRAM ▸
工作中的程式	開啟 / 在工作中提前 / 在工作中延後 / 移出這個工作
POU（資料夾）	新增 PROGRAM（階梯圖 / ST / SFC）、FUNCTION_BLOCK（階梯圖 / ST）、FUNCTION（ST）
一個 POU	開啟 / 重新命名... / 交互參照 / 從專案移除 / 在工作中執行 ▸（PROGRAM 才有；目前所在的工作打勾）
全域變數、資料型別、使用者程式	資料夾：新增全域變數表... / 新增資料型別... / 新增使用者程式檔（C/C++）...；檔案：開啟 / 從專案移除（資料型別另有「重新命名...」）
Sketch（86Duino IDE）	Sketch 的 .ino（86Duino IDE 的主程式） / 在 sketch 新增 .h / .cpp... / 開啟 sketch 資料夾 / 開啟 86Duino IDE

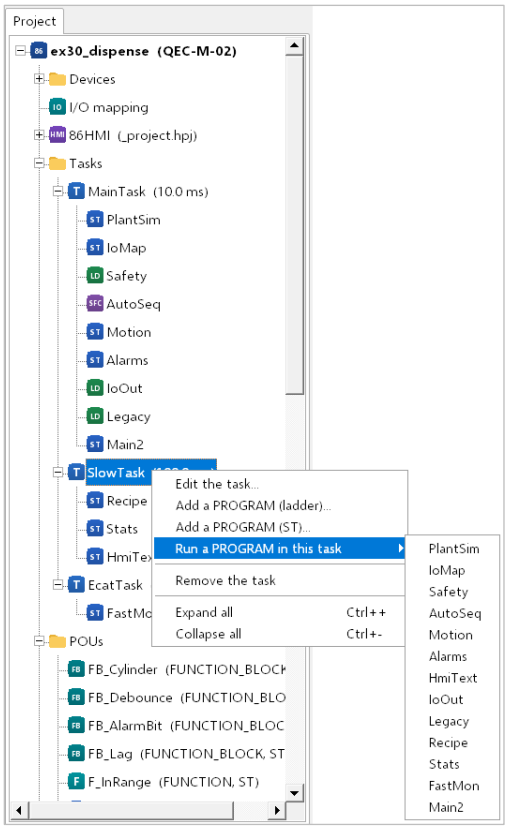


圖 3-17　工作節點的右鍵選單：「在這個工作執行 PROGRAM」子選單列出所有 PROGRAM

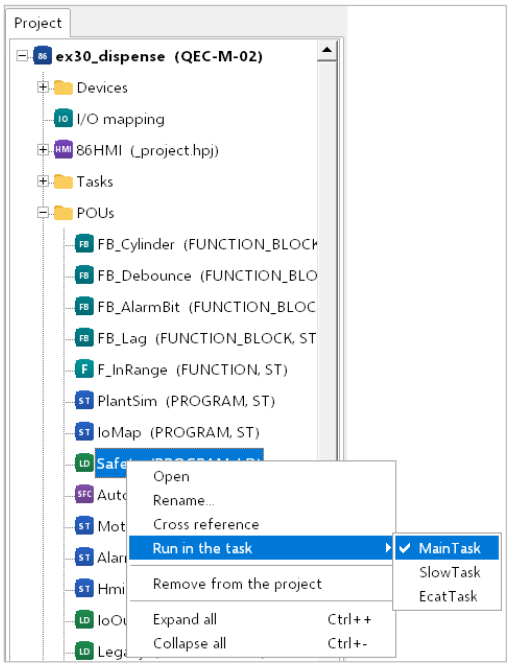


圖 3-18　POU 的右鍵選單：開啟、重新命名、交互參照、從專案移除、在工作中執行

- 每個右鍵選單（包括在空白處按右鍵）最下方都有 **全部展開**、**全部收合**；專案樹有焦點時也可按 **Ctrl+ +** / **Ctrl+ -** 。
- 「從專案移除」只把檔案移出專案，檔案仍保留在 `data\86ld` 中（會先詢問），需要時請自行刪除。
- 選取 PDO 項目或 86HMI 元件時，右側欄的「屬性」顯示它的變數與數值。

3.5 POU 編輯器與右側欄

在專案樹雙擊 POU，文件區開啟它的編輯器。LD 與 SFC 編輯器上方是宣告區（`VAR ... END_VAR`，與 ST 相同的程式碼編輯器，有語法顏色與補全），下方是圖形；ST 編輯器是整個程式的文字。LD、ST、SFC 分頁的右邊都有右側欄，上方兩個分頁「**工具箱**」「**屬性**」，右上角的 **»** 按鈕可收合 / 展開側欄。

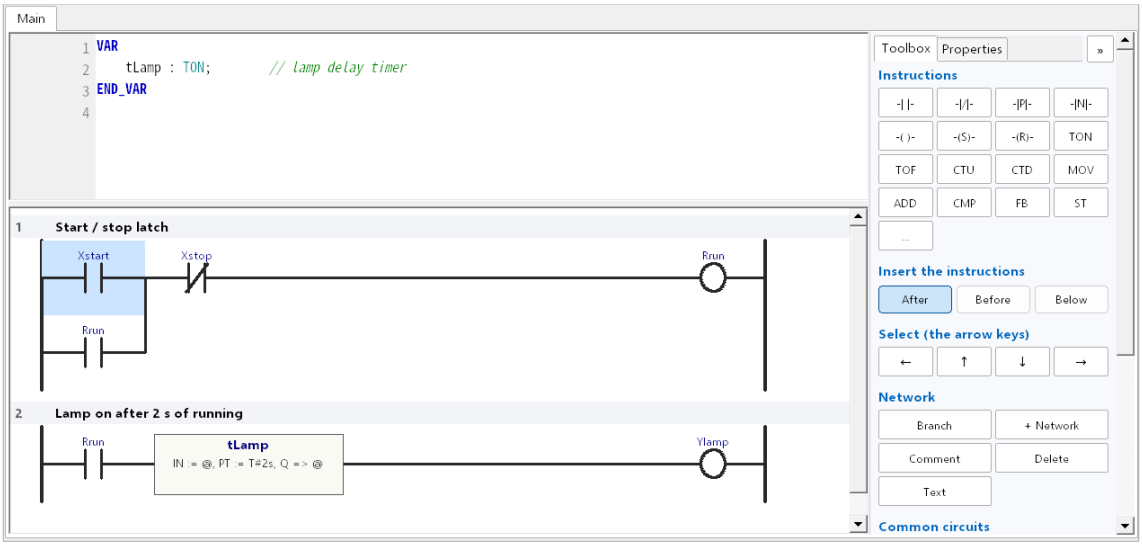


圖 3-19　階梯圖編輯器：上方宣告區、中間網路、右側欄「工具箱」

編輯器	「工具箱」的內容	詳細說明
階梯圖（LD）	指令（接點、線圈、TON、TOF、CTU、CTD、MOV、ADD、CMP、FB、ST，「...」列出全部指令）；指令插入位置（插在後、插在前、並聯下方）；選取（方向鍵）（←↑↓→）；網路（並聯、+ 網路、註解、刪除、文字）；常用接線（啟動 / 停止自保等範本）	第 7 章
結構化文字（ST）	ST 常用片段（IF、CASE、FOR、計時器……），雙擊插入	8.1 節
順序功能圖（SFC）	圖表：+ 步驟、選擇分支、並行分支、+ 分支、跳躍、編輯、刪除、文字	8.4 節

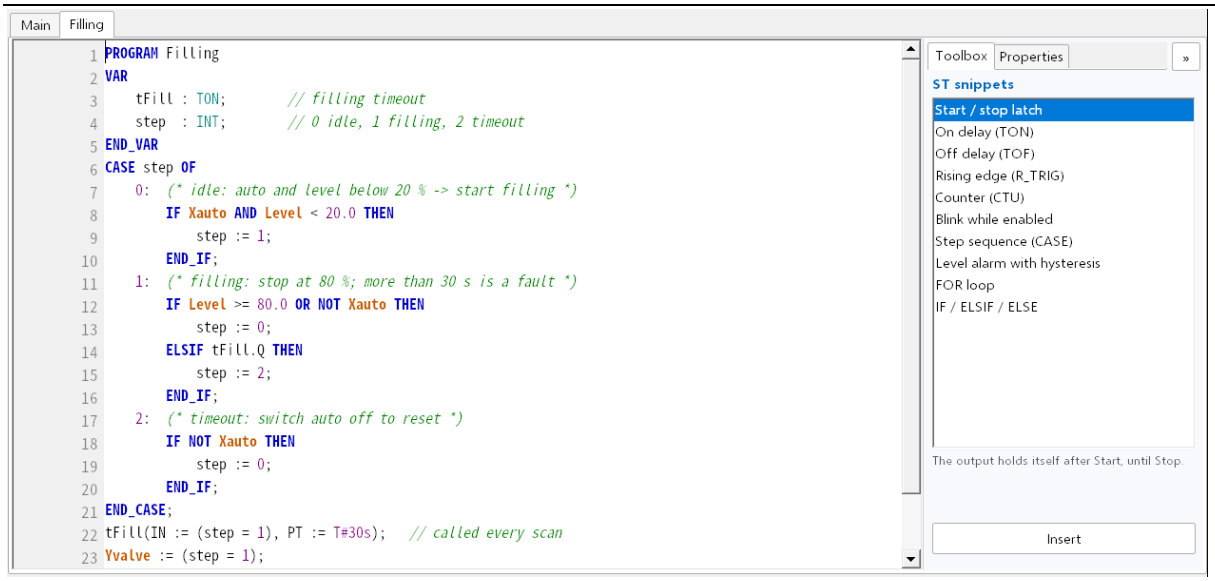


圖 3-20　ST 編輯器與右側的「ST 常用片段」

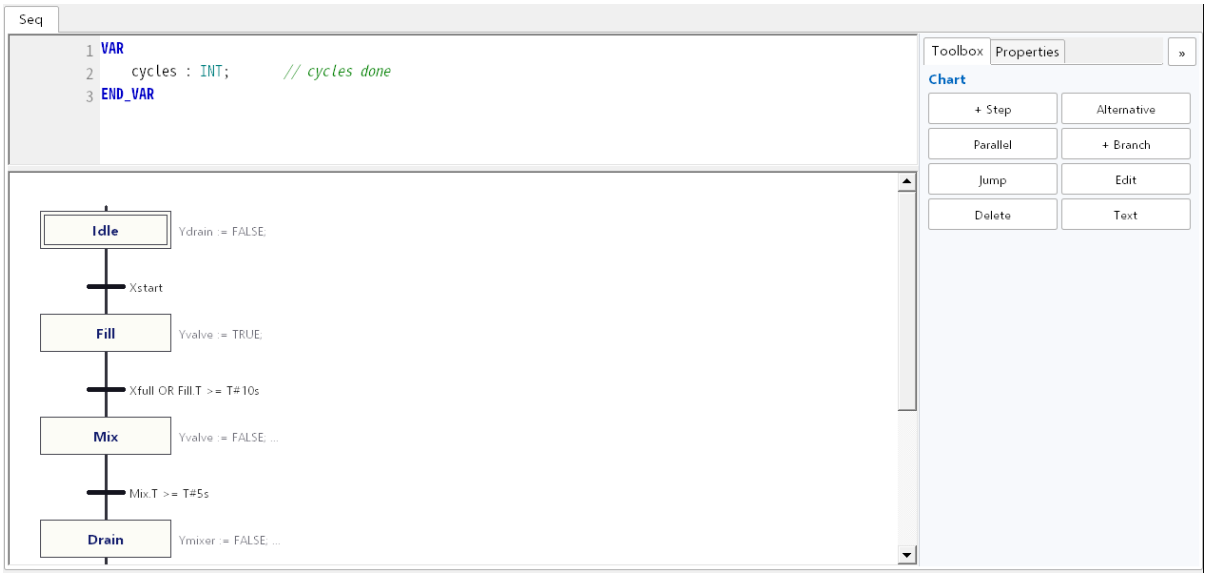


圖 3-21　SFC 編輯器：上方宣告區、中間圖表、右側「圖表」工具

「**屬性**」分頁顯示選取的階梯圖元件、監看列、EtherCAT 項目或 86HMI 元件的變數：型別、宣告於、初始值、註解、通道（I/O 對應）、86HMI 綁定，以及模擬或線上時的數值。沒有選取時顯示「選取階梯圖元件、監看列、EtherCAT 項目或 86HMI 元件：這裡顯示其變數、數值、I/O 對應與 86HMI 綁定。」

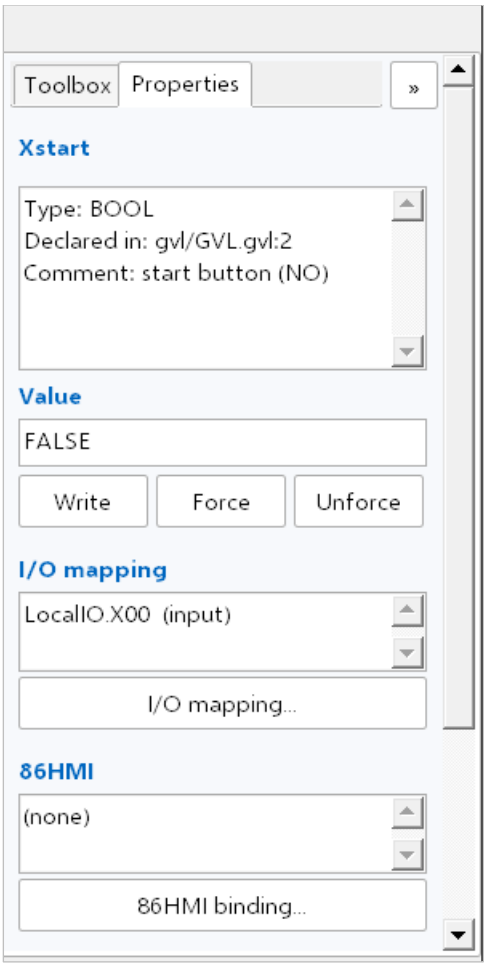


圖 3-22　右側欄「屬性」：選取元件的變數、型別、I/O 對應與數值

在階梯圖元件、程式碼中的變數或監看列上按右鍵，都有同一組變數選單（以變數 `Rrun` 為例）：寫入 Rrun...、強制 Rrun...、解除強制（以上三項在模擬或線上時才有）、Rrun 的 I/O 對應...、Rrun 的 86HMI 綁定...、將 Rrun 加入監看視窗、加入趨勢圖、Rrun 用在哪裡？、跳到宣告、屬性。

*__提示：__階梯圖與 SFC 按 Ctrl+T 切到文字模式（整個 POU 的 .lad / .sfc 文字），按「回到圖形」回到圖形編輯（7.1、8.4 節）。

3.6 工具視窗

下方的工具視窗以分頁切換；「檢視」選單可再叫出已關閉的視窗。分頁按右鍵可選「**浮動（獨立視窗）**」成為獨立視窗（例如拖到第二個螢幕），關閉浮動視窗即回到原位；區塊之間的分隔線可拖曳調整大小。

視窗	用途	詳細說明
訊息	檢查與建置的錯誤、警告：訊息、檔案、行。雙擊跳到出錯的元件或行；右鍵：前往、用 AI 修正（複製提問內容）、複製訊息	9.6 節
輸出	上方切換「建置輸出」與「序列埠監看」：建置、編譯、上傳的過程與 Serial.print 的內容	10.1、10.3 節
監看	上方輸入變數名稱按 **加入**；顯示變數、數值、型別；雙擊 BOOL 切換，右鍵寫入、強制、加入趨勢圖、從監看移除；**解除全部強制**	9.3 節
模擬 I/O	模擬時列出每個 I/O 對應的實體點（通道、變數、輸入 / 輸出）；雙擊切換數位輸入、輸入類比值	9.2 節
事件紀錄	控制器（或模擬器）的事件：啟動、EtherCAT 狀態、錯誤、看門狗、強制釋放	10.5 節
趨勢圖	監看中的變數右鍵「加入趨勢圖」（最多 8 個）；每 100 ms 取樣，顯示最近一分鐘；右鍵：暫停、清除取樣、移除全部變數、以 CSV 複製	9.4 節
交互參照	一個變數在各 POU 中被讀、寫、宣告的位置（Ctrl+Shift+F）	9.5 節
軸（試車）	所有軸的激磁、位置、狀態；可直接下命令	12.5、12.6 節
搜尋結果	在專案中搜尋（Ctrl+Shift+H）的結果；雙擊跳到該行	3.7、9.5 節

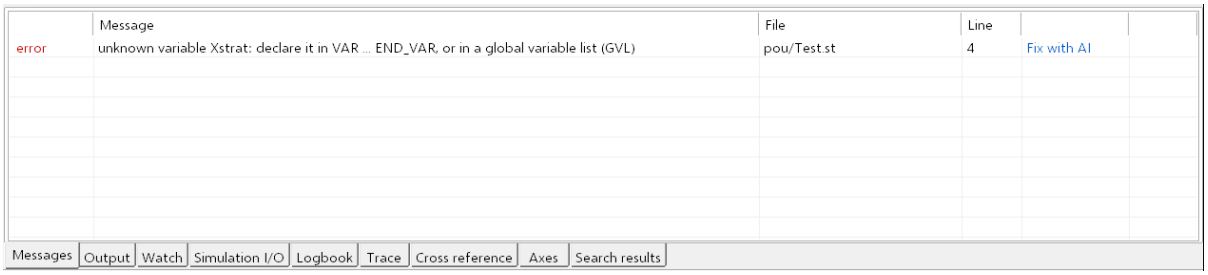


圖 3-23 「訊息」視窗：每列一個錯誤，雙擊跳到該元件

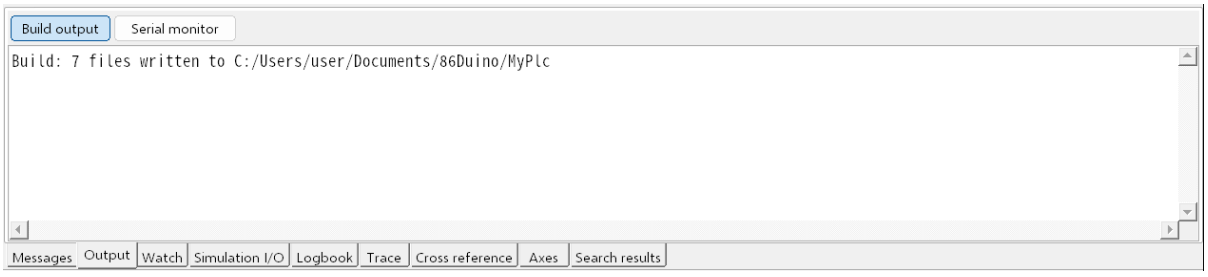


圖 3-24 「輸出」視窗的「建置輸出」：建置、編譯與上傳的過程

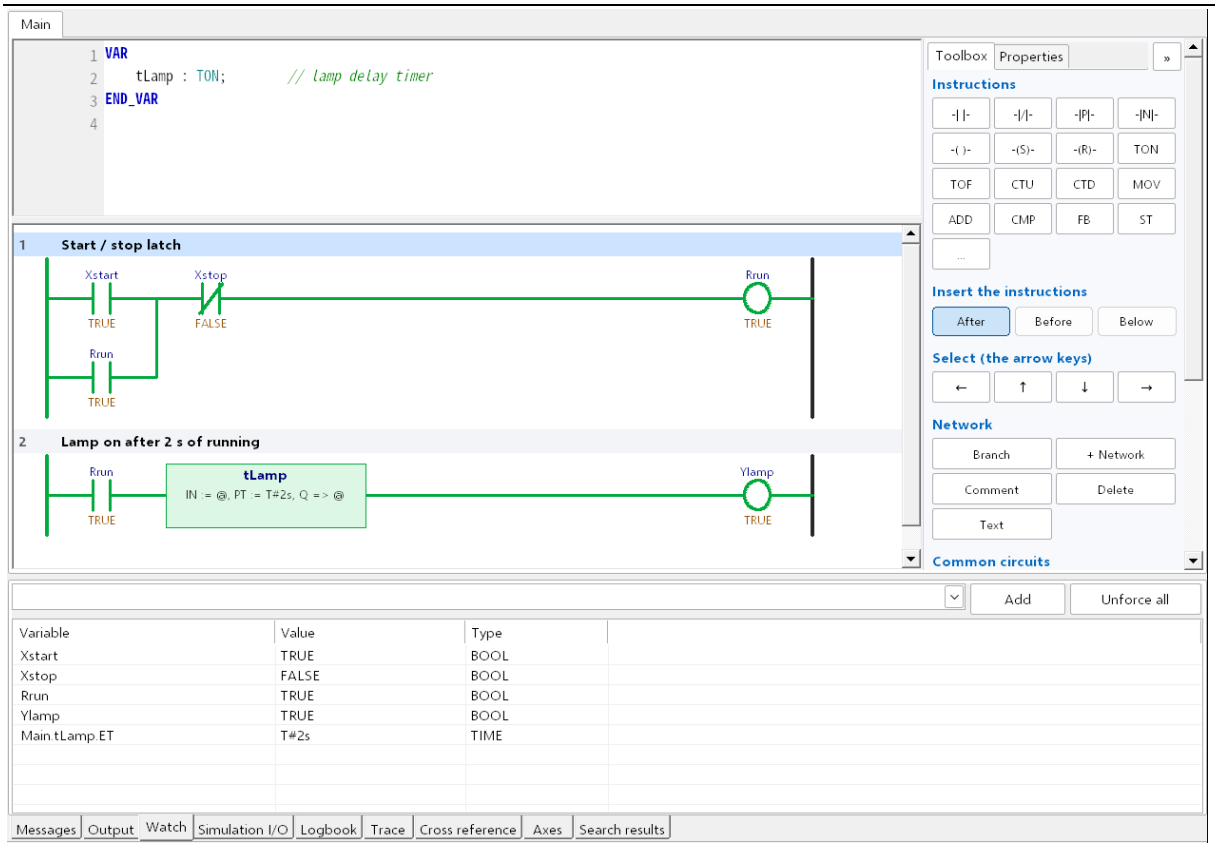


圖 3-25 「監看」視窗：加入變數後即時顯示數值（也可寫入、強制）

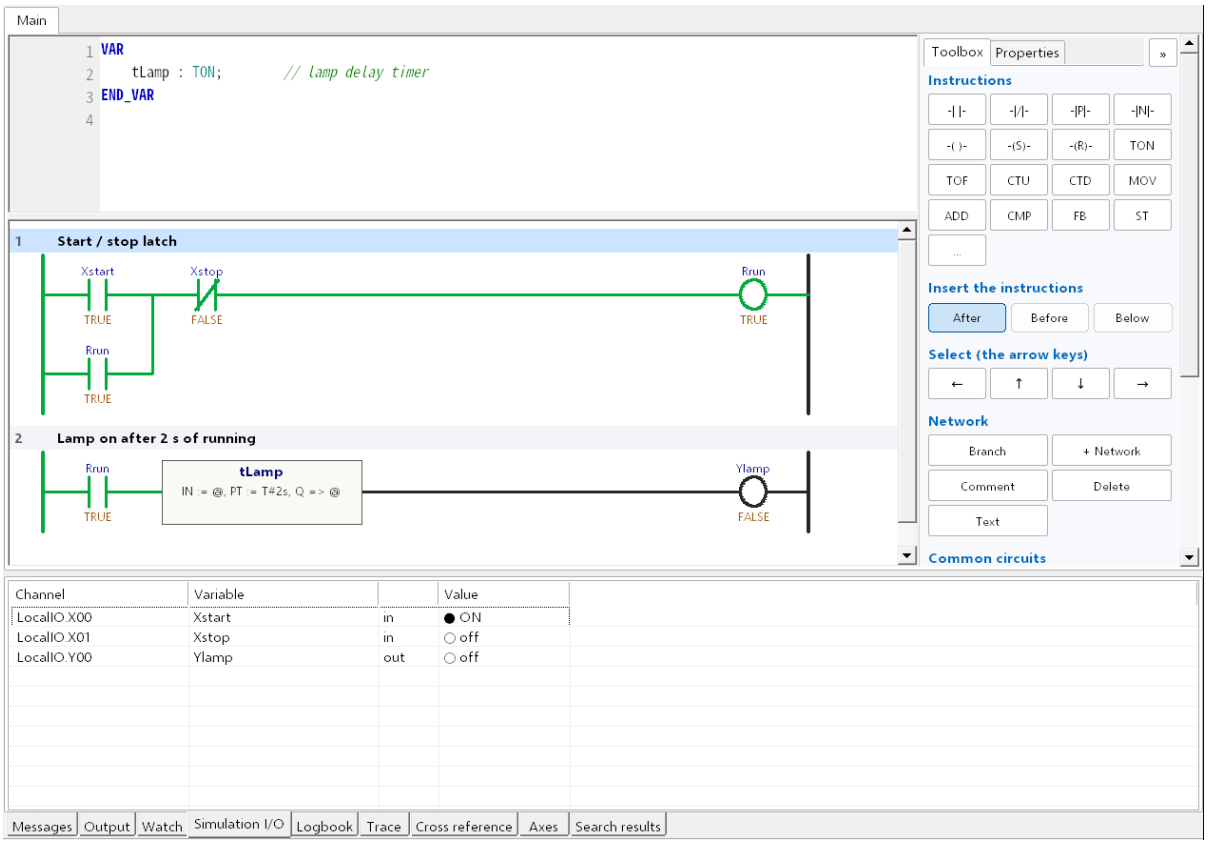


圖 3-26 「模擬 I/O」：LocalIO.X00 等實體點

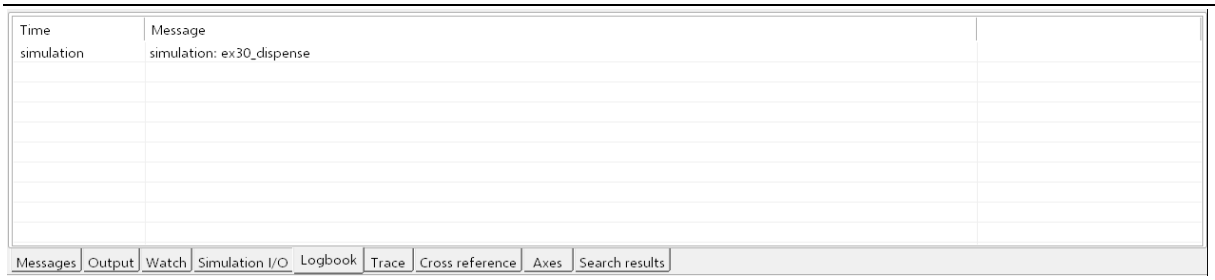


圖 3-27 「事件紀錄」：EtherCAT 主站啟動訊息與程式開始執行

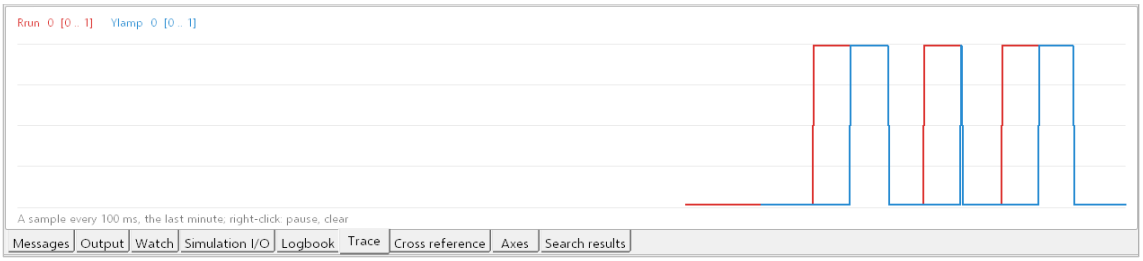


圖 3-28 「趨勢圖」：監看中的變數加入後的波形

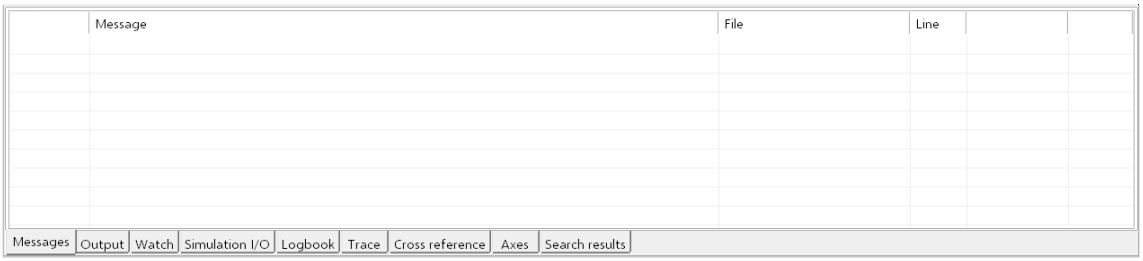


圖 3-29 「交互參照」：變數 Rrun 被讀、寫、宣告的位置

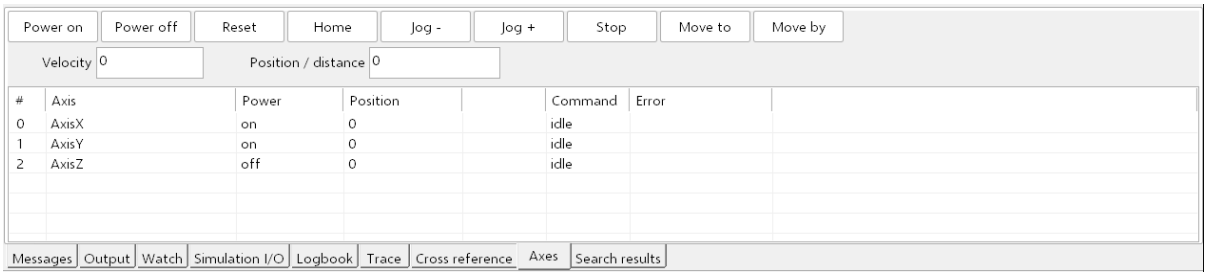


圖 3-30 「軸（試車）」視窗



圖 3-31 工具視窗分頁的右鍵選單：浮動（獨立視窗）

3.7 分頁、左右並排、縮放與尋找

3.7.1 分頁

- 每個 POU、GVL、設定頁一個分頁。**Ctrl+Tab / Ctrl+Shift+Tab** 切換分頁；**Ctrl+W** 或在分頁上按 **滑鼠中鍵** 關閉。
- 分頁按右鍵：關閉（Ctrl+W）、關閉其他、全部關閉、分割：移到右側（已並排時為「移到右側」/「移到左側」）、結束分割（全部回到左側）。
- 每個專案會記住開啟的分頁與分割狀態，下次開啟專案時還原。

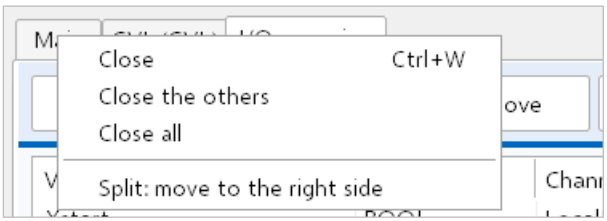


圖 3-32 分頁的右鍵選單

3.7.2 左右並排

1. 分頁按右鍵 >「**分割：移到右側**」，或把分頁拖到文件區右側 1/3 處，或「檢視 > 左右並排」（**Ctrl+**）。至少要開兩個分頁。
2. 中間分隔線可拖曳；分頁可用右鍵「移到左側 / 移到右側」或拖曳在兩側之間移動。
3. 點一下哪一側，那一側就是目前操作的一側（上方藍線標示）；儲存、尋找、Ctrl+Tab 作用在這一側。模擬與線上時兩側的值同時更新。
4. 一側的分頁全部關閉，或右鍵「結束分割（全部回到左側）」，即回到單欄。

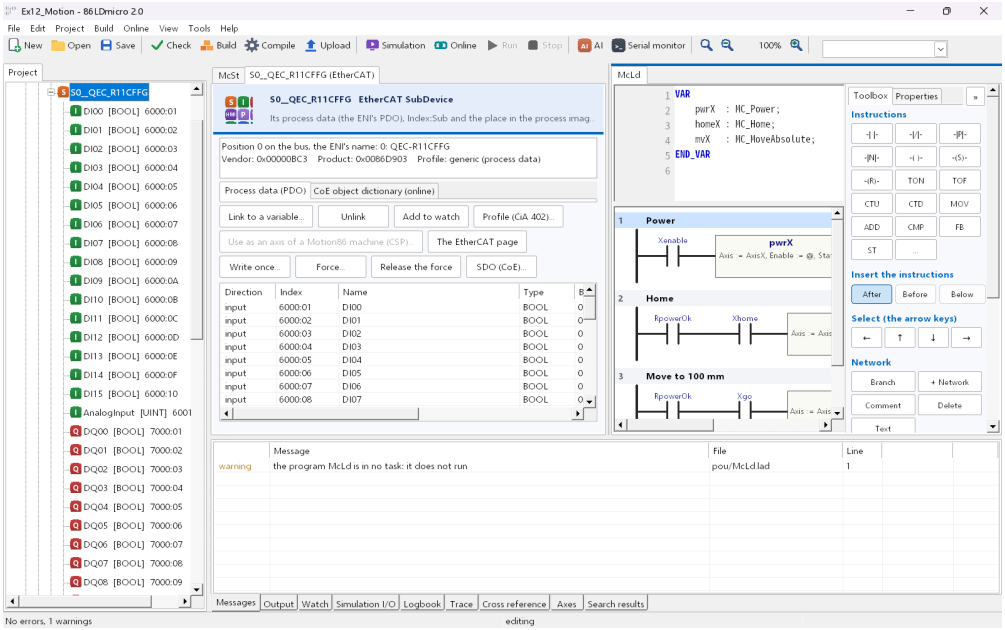


圖 3-33 左右並排：左側 SubDevice 頁、右側階梯圖

3.7.3 縮放

Ctrl+滾輪、**Ctrl++ / Ctrl+- / Ctrl+0** 或工具列的縮小、比例、放大按鈕。程式碼、階梯圖、設定頁、專案樹與工具視窗一起縮放，比例會被記住，下次啟動沿用。

3.7.4 在此分頁尋找（Ctrl+F）

1. 按 **Ctrl+F**（或工具列放大鏡、「編輯 > 在此分頁尋找...」），分頁上方出現尋找列。先選取一段文字再按，會直接尋找那段文字。
2. 輸入要找的文字；需要時勾選「大小寫相符」。右邊顯示「第幾個 / 共幾個」，沒有時顯示「找不到」。
3. **Enter** 或 **F3**、▼ 找下一個；**Shift+Enter** 或 **Shift+F3**、▲ 找上一個；**Esc** 或 ✕ 關閉。階梯圖中會依序選到符合的接點、線圈與網路標題。

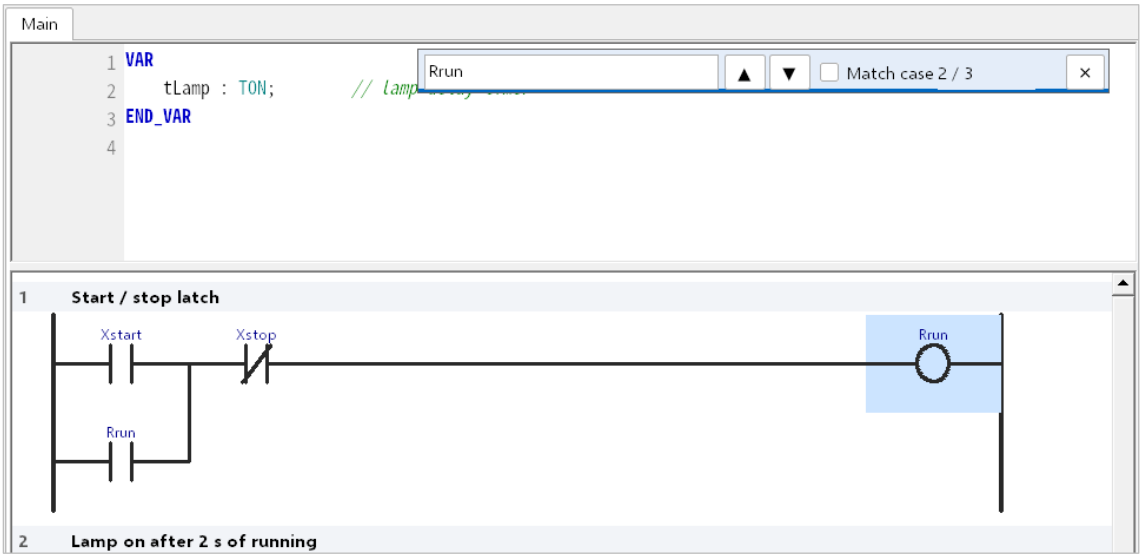


圖 3-34　尋找列：在階梯圖中尋找 Rrun（第 1 個，共 3 個）

3.7.5 在專案中尋找變數（Ctrl+Shift+F）與在專案中搜尋（Ctrl+Shift+H）

功能	搜尋什麼	結果	適合
在此分頁尋找（Ctrl+F）	目前分頁的文字或階梯圖元件	尋找列	在一個程式裡逐一找
在專案中尋找變數（Ctrl+Shift+F）	一個變數在各處的讀、寫、宣告	交互參照	確認變數被誰寫入
在專案中搜尋（Ctrl+Shift+H）	整個專案的任意文字：所有 POU（含網路標題、註解）、GVL、資料型別、使用者 C/C++、sketch 的 .ino	搜尋結果	找註解、字串、函式或不確定名稱的東西



圖 3-35 「尋找變數」：輸入變數名稱

1. 按 **Ctrl+Shift+H**，下方切到「搜尋結果」。（先在程式中選取一段文字再按，會直接搜尋那段文字。）
2. 輸入文字，按 **Enter** 或 **尋找**；需要時勾選「大小寫相符」或「全字相符」（例如找 `Pump` 時不列出 `PumpSpeed`）。右上角顯示共幾處。
3. **雙擊**一列（或選取後按 Enter），開啟該檔案並跳到那一行：ST 與 C/C++ 選取該行，階梯圖捲到並選取那個網路。

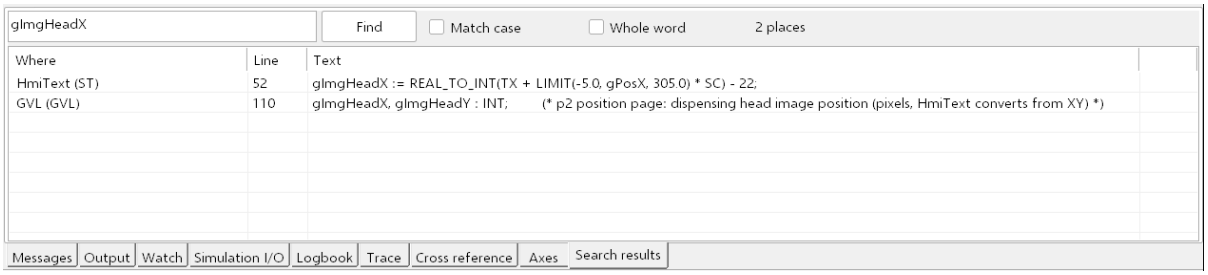


圖 3-36 「搜尋結果」：找到 2 處（ST 程式與全域變數表）

PlantSim | HmiText

```
gTxtPattern := gLang * 100 + LIMIT(0, hmiEdPattern, 9);

(* ---- Pages: tab buttons change pages in 86HMI itself (button's Page Change script, not via the PLC).
   On a new stop alarm the PLC requests p5: gPage is 5 for that one scan only, then back to 0 --
   the page changes only when the page variable changes, so after the operator moves to another page, the
newStop(CLK := gAlarmStop);
IF newStop.Q THEN
    gPage := 5;
ELSE
    gPage := 0;
END_IF;

(* ---- Display copies *)
gYieldPct := INT_TO_REAL(gYieldX10) / 10.0;
gPressBar := xGluePress;
gPressBar2 := xGluePress;
gCycleS := DINT_TO_REAL(gCycleAvgMs) / 1000.0;
gLastCycleS := DINT_TO_REAL(gCycleMs) / 1000.0;
gUphShow := gUph;
(* ---- p2 position page: image pixel positions (86HMI Image objects move via Image (x) / Image (y) bind
   Image top-left = center - half size; clamped to the table when out of travel *)
gImgHeadX := REAL_TO_INT(TX + LIMIT(-5.0, gPosX, 305.0) * SC) - 22;
```

圖 3-37　雙擊一列：開啟該 ST 程式並選取那一行

*提示：搜尋只會尋找、不會取代。要改變數名稱，請從結果逐一開啟修改，改完按「檢查」（Shift+F7）確認沒有遺漏。

在外部改過檔案後，按 **Ctrl+R**（檢視 > 重新整理）從磁碟重新讀取專案（3.2.6 節）。

第 4 章　專案與設定

本章說明一個 86LDmicro 專案由哪些檔案組成，以及專案設定、工作（Task）、裝置、全域變數與資料型別、I/O 對應的設定方式，最後是新增 POU 與使用者程式的方法。

4.1　專案的檔案結構

86LDmicro 的專案放在它所屬的 86Duino sketch 資料夾中，全部是 UTF-8 文字檔，可以直接用 git 等工具做版本控管。以第 5 章的 `MyPlc` 為例（位置預設為 `文件\86Duino`）：

```
文件\86Duino\MyPlc\
    MyPlc.ino                       呼叫 plc86_setup() / plc86_loop()（只在沒有時寫一次，之後由你維護）
    plc86.h、plc86.cpp              PLC 程式（建置產生，每次重寫，請勿手改）
    plc86_rt.h、plc86_rt.cpp        86LD run-time（建置產生）
    plc86_ecat.h、plc86_ecat.cpp    EtherCAT（建置產生）
    myhmi.h、myhmi.cpp              86HMI 編輯器「產生程式碼」寫出（有 86HMI 畫面時）
    data\
        *.hpj                       86HMI 畫面（有時）
        Resource\                   86HMI 的圖片、字型等資源（有時）
        86ld\
            project.86ld            專案設定（JSON）：控制器、工作、裝置、I/O 對應、綁定
            pou\                    POU：*.lad（階梯圖）、*.st（ST）、*.sfc（SFC）
            gvl\                    全域變數表 *.gvl
            dut\                    資料型別 *.dut
            usercode\               使用者 C/C++ *.cpp86
            ethercat\               匯入的 ENI 副本、CoE 物件字典快取
            build\                  plc86.symbols.json（線上監看用的符號表）、編譯輸出
```

新建立的專案只有 `MyPlc.ino`、`project.86ld`、`pou\Main.lad` 與 `gvl\GVL.gvl`；其他資料夾在新增對應的檔案、讀入 ENI 或建置時才出現。新專案的 `project.86ld` 如下：

```
{
  "format": "86LDmicro project",
  "version": 1,
  "name": "MyPlc",
  "target": {
    "board": "QEC-M-02",
    "mcu": "Vortex86EX2"
  },
  "tasks": [
    {
      "name": "MainTask",
      "kind": "cyclic",
      "intervalUs": 10000,
      "priority": 1,
      "programs": ["Main"]
```

```
    }
  ],
  "pous": ["pou/Main.lad"],
  "gvls": ["gvl/GVL.gvl"],
  "duts": []
}
```



圖 4-1　sketch 資料夾：.ino、建置產生的 plc86* 檔與 data\86ld

- `plc86*` 檔案與 `data\86ld\build` 每次建置都會重新產生，可以不納入版本控管。
- `.ino` 是你的檔案：86LDmicro 只在沒有時寫一次，之後只確保它呼叫 `plc86_setup()`、`plc86_loop()`（14.5 節）。
- 從專案移除的檔案不會被刪除，仍留在 `data\86ld` 中，需要時請自行刪除。
- 這些文字檔也可以在外部編輯器或由命令列、AI / MCP 修改，改完在 86LDmicro 按 **Ctrl+R**（檢視 > 重新整理）重新讀取（3.2.6 節）。階梯圖 `.lad` 的文字格式見 7.1 節。

4.2 專案設定

在專案樹雙擊專案名稱（最上面的節點），或「專案 > 設定」，開啟「專案設定」頁。頁面列出專案名稱、資料夾、控制器、線上監看與 86Duino IDE 的位置；上方三個按鈕：

按鈕	功能
編輯...	開啟「專案設定」對話框（見下）
選項...	開啟「選項」（86Duino IDE 資料夾、語言，2.6 節）
開啟資料夾	在檔案總管開啟 sketch 資料夾

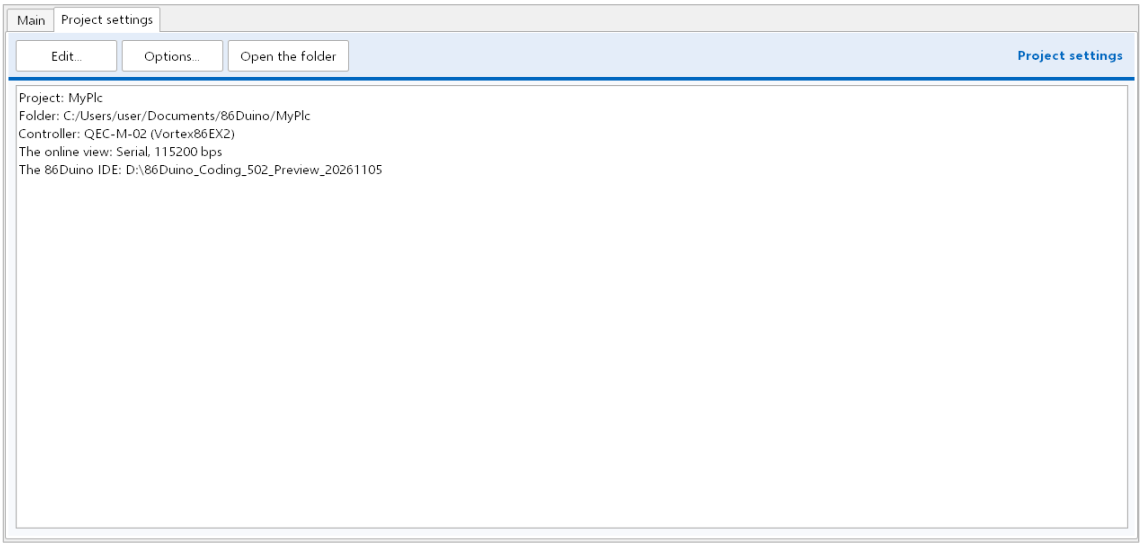


圖 4-2 「專案設定」頁：專案、資料夾、控制器、線上監看、86Duino IDE

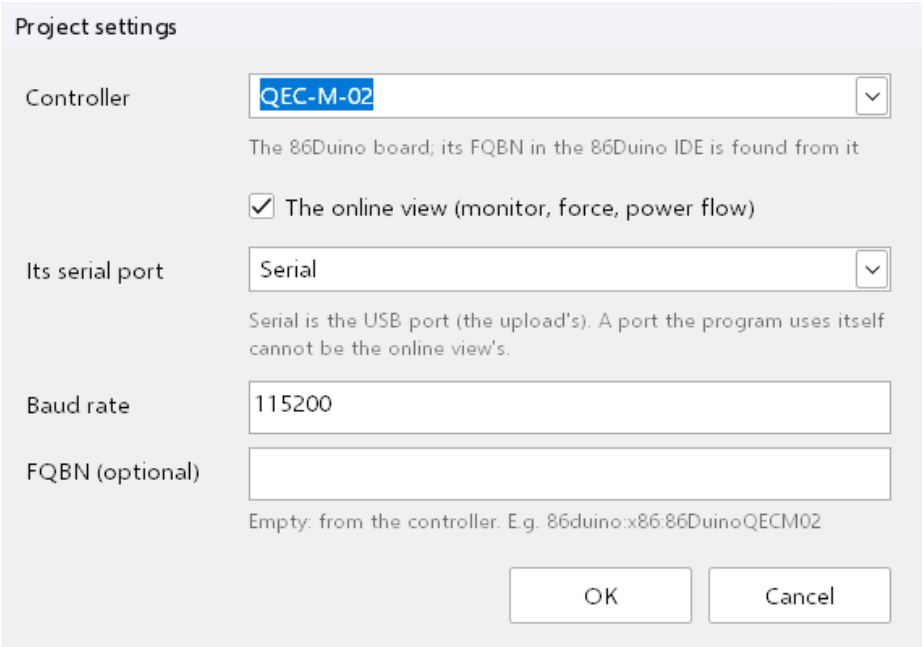


圖 4-3 「專案設定」對話框：控制器、線上監看、序列埠、鮑率、FQBN

欄位	預設	說明
控制器	QEC-M-02	決定編譯目標（86Duino IDE 中的 FQBN）與本機 I/O。清單有 QEC-M-02、QEC-M-01、QEC-M-043T、QEC-M-070T、QEC-M-090T、QEC-M-150T，以及 86Duino IDE 的 boards.txt 中其他 QEC 板子（例如 PPC 系列；IDE 版本較新時會自動出現）
線上監看（監看、強制、能流）	勾選	勾選後，上傳的程式內含線上監看功能（10.4 節）
使用的序列埠	Serial	線上監看用的埠：`Serial` 為 USB（即上傳用的埠，建議）；也可選 Serial1、Serial2、Serial3、SerialCOM。程式自己使用的序列埠不能作為線上監看埠
鮑率	115200	86LDmicro 線上連線時自動使用這個鮑率；修改後要重新建置、上傳
FQBN（選填）	空白	空白時依控制器自動決定；特殊板子可直接指定，例如 `86duino:x86:86DuinoQECM02`

*__注意__：若 sketch 的 `.ino` 在 `plc86_setup()` 之後又呼叫 `Serial.begin()`，會蓋掉線上監看的鮑率設定，造成連線失敗（10.4 節）。

從 86Duino IDE 啟動時，IDE 選的開發板與專案的控制器不同，86LDmicro 會詢問是否改用 IDE 的板子（2.5 節）。

4.3 工作（Task）

工作決定 PROGRAM 何時執行、以什麼順序執行。專案樹雙擊「工作 (Task)」（或「專案 > 工作 Task」）開啟工作頁：每個工作一列，顯示工作、類型、週期、看門狗與程式。上方按鈕為 **新增...**、**編輯...**、**移除**、**上移**、**下移**；雙擊一列等於 **編輯...**。

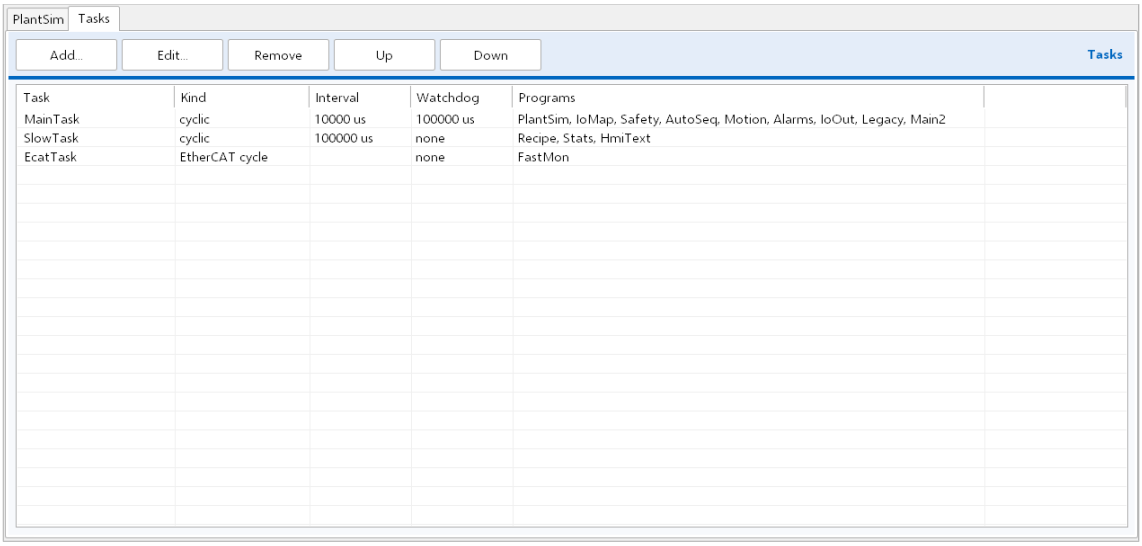


圖 4-4　工作頁：工作、類型、週期、看門狗、程式

欄位	說明
名稱	字母、數字、_（例如 `MainTask`）
類型	週期、自由執行、EtherCAT 週期（見下表）
週期（us）	週期工作的執行間隔，100 us ~ 10 s（輸入 100 ~ 10000000）。新專案的 MainTask 為 10000（10 ms）
看門狗（us）	單次掃描超過此時間即停止 PLC；0 表示不使用（見下方說明）
程式（依執行順序）	以逗號分隔的 PROGRAM 名稱，依序執行；欄位下方列出專案中所有的 PROGRAM

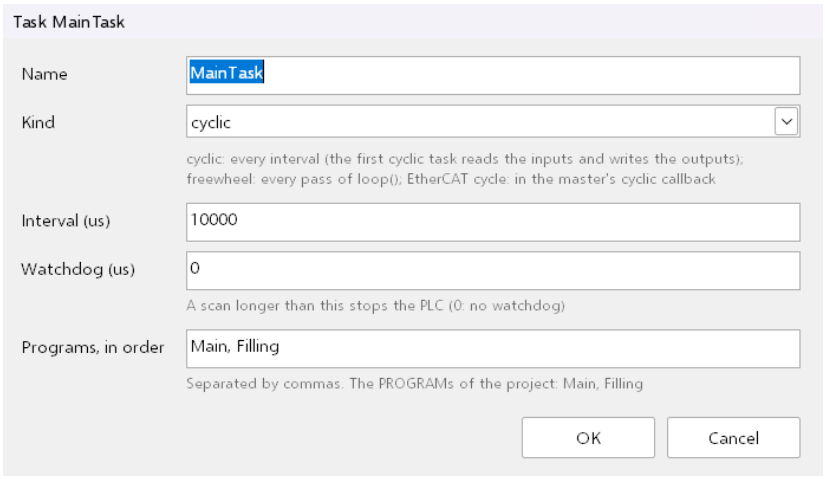


圖 4-5　工作的設定：名稱、類型、週期、看門狗、程式（依執行順序）

類型	執行時機	用途
週期	每隔「週期」執行一次	一般 PLC 邏輯。**第一個週期工作**在程式之前讀入所有輸入、之後寫出所有輸出
自由執行	loop() 每一輪	背景、低優先的處理
EtherCAT 週期	EtherCAT 主站的週期回呼（86EVA 或 EtherCAT 裝置）	需要與 EtherCAT 同步的控制；會中斷其他工作，請保持簡短

- 專案至少要有一個**週期**工作；**EtherCAT 週期**工作最多一個。
- 一個 PROGRAM 只能放在一個工作中（也不能在同一個工作中放兩次），否則檢查時會出錯。FUNCTION_BLOCK 與 FUNCTION 不放進工作，由 PROGRAM 呼叫。
- 「專案 > 新增 PROGRAM」建立的程式自動加入第一個工作的最後面。
- 也可以在專案樹操作：工作節點右鍵「在這個工作執行 PROGRAM ▸」；工作中的程式右鍵「在工作中提前」「在工作中延後」「移出這個工作」；POU 右鍵「在工作中執行 ▸」（3.4 節）。
- 週期工作的一次掃描超過週期時，控制器的事件紀錄會記下「task 工作名稱: a scan took … us (the interval is … us)」（第一次與之後每 1000 次記一次）。

4.3.1 看門狗

看門狗用來發現太慢的程式（掃描逾時檢查）。它只在控制器上、只對週期工作判定（模擬時不判定）：

- 判定的時間是單次掃描中程式本身的執行時間：EtherCAT 週期中斷、EEPROM 寫入（RETAIN）、運動控制郵箱與 86HMI 畫面更新的時間不計入。
- 每個工作的前 3 個掃描不判定（啟動時載入 RETAIN、第一次設定初值等）。
- 超過設定時 PLC 進入 STOP（與按 STOP 相同：程式不再執行，輸出保持最後的值，所有 Motion86 機台停止、寸動結束；10.4 節）。事件紀錄寫出工作名稱、這次掃描的時間與最慢的程式，例如「MainTask watchdog: STOP, 120 ms (3 other), AutoSeq 110 ms」。
- 線上時 86LDmicro 的「輸出」顯示「PLC 已停止（STOP）：有一次掃描超過工作的看門狗時間（專案 > 工作）。按 RUN（F9）重新執行。」，狀態列顯示「控制器的看門狗已停止 PLC：請看「輸出」」。
- 排除原因後按 **RUN**（F9）重新執行。處理方式：減少每個掃描的運算（迴圈、字串處理改分次做），或加大看門狗時間。

*__提示：__看門狗時間要比正常的掃描時間大得多（例如週期 10 ms 的工作設 100000 us），只用來發現異常。正常的掃描時間在線上監看時顯示於狀態列（3.1 節）。

4.4 裝置

裝置是專案使用的硬體與通訊：本機 I/O、EtherCAT、86EVA、序列埠、Modbus、網路與 Motion86 機台。專案樹雙擊「裝置」（或「專案 > 裝置」）開啟裝置頁，每個裝置一列（裝置、型別、設定摘要），上方按鈕為 **新增...**、**編輯...**、**移除**。

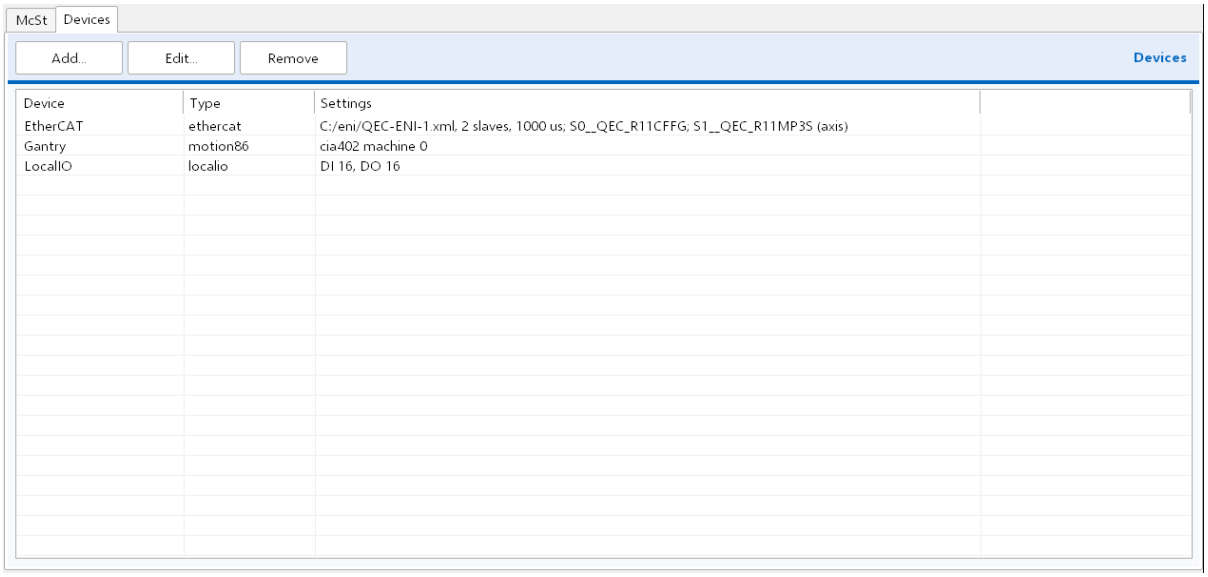


圖 4-6　裝置頁：localio（DI 16, DO 16）

1. 按 **新增...**（或在專案樹「裝置」按右鍵 >「新增裝置 ▸」直接選類型）。
2. 在「新增裝置」的「型別」選擇類型，按 **確定**。
3. 接著出現該裝置的設定表單：每個欄位一列（勾選框為開 / 關，數字、文字直接輸入），清單類的欄位（例如序列埠清單、Modbus 從站）以 JSON 文字編輯。按 **確定** 加入。

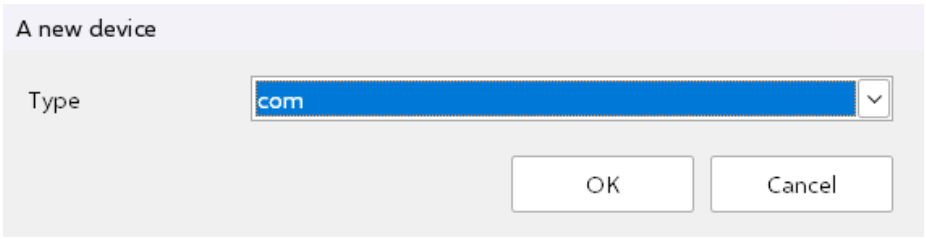


圖 4-7　「新增裝置」：選擇型別

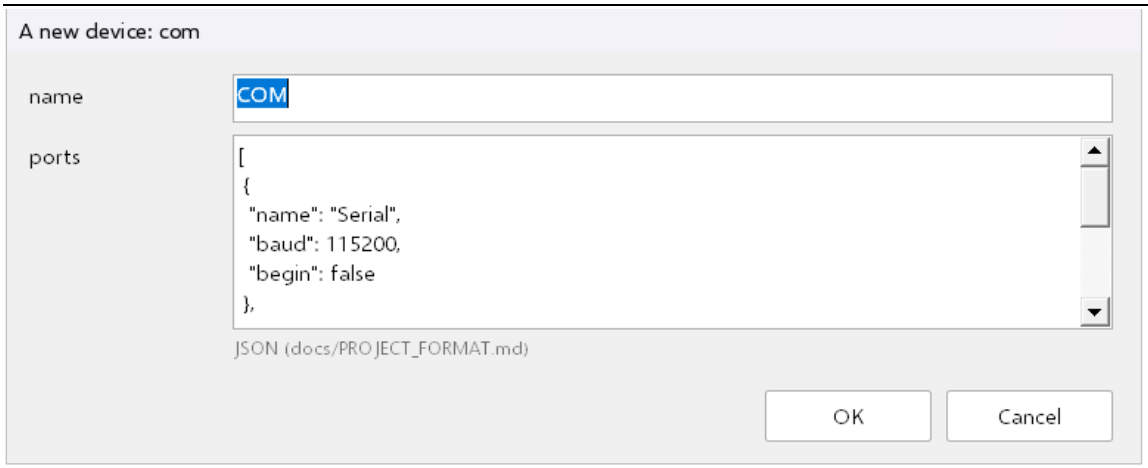


圖 4-8　com 裝置的設定表單：名稱與 ports（JSON）

型別	說明	新增時的預設內容
localio	QEC 本機 DI / DO 點數（QEC-M-02：16 / 16）	名稱 LocalIO，di 16、do 16
eva	86EVA 虛擬裝置（LDmicro 1.x 的 EtherCAT 方式）	enable、io 16、analog 4、decimal 0、servo 0、serial 0
com	序列埠：依序為 COM0、COM1…（UART 指令用）	Serial 115200（不 begin）、Serial1 9600
modbus	Modbus RTU / TCP 主站：匯流排與從站清單	匯流排 0：RTU、COM1、啟用；從站清單空白
ethernet	網路設定（static 固定 IP / dhcp）	static，192.168.1.100 / 255.255.255.0，閘道與 DNS 192.168.1.1
modbus-server	Modbus TCP 從站：把變數開放給 SCADA / HMI 讀寫	on、port 502、unit 1、vars 空白
ethercat	EtherCAT 主站：通常由讀入 ENI 建立，從站與 PDO 來自 ENI（第 11 章）	eni QEC-ENI.xml、port 0、cycleUs 1000；另有 mode（auto / sync / freerun）與 master（plc / sketch）
motion86	Motion86 機台與軸：開啟「新增 Motion86 機台」（第 12 章）	—

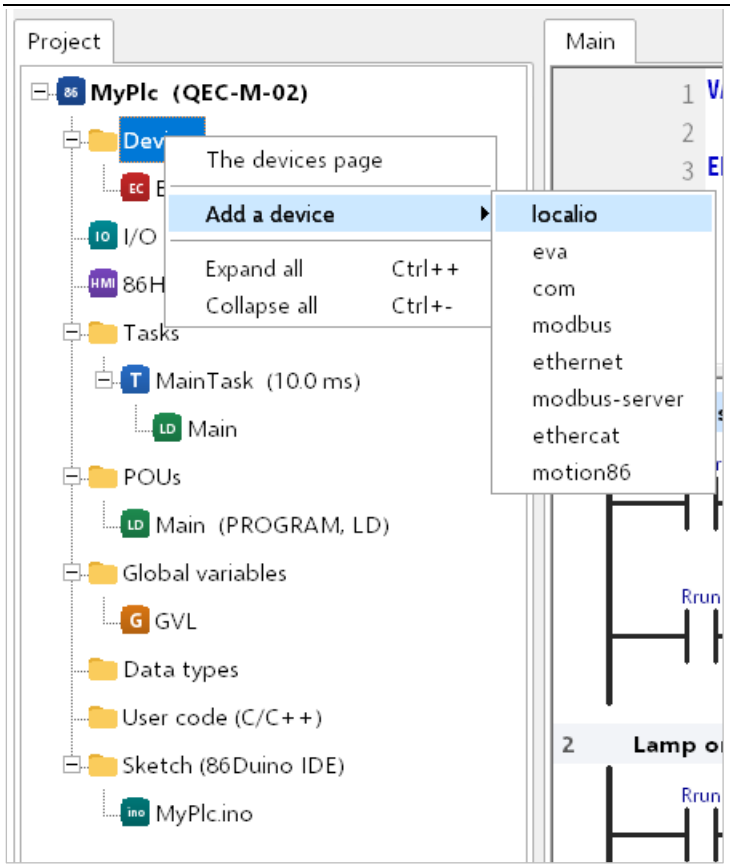


圖 4-9　專案樹「裝置」的右鍵選單：新增裝置 ▸ 列出 8 種型別

提示：新專案沒有 localio 裝置時，仍可使用 QEC-M-02 本機 I/O 的通道 `LocalIO.X00`～`X17`、`LocalIO.Y00`～`Y17`（預設 16 / 16 點）。

EtherCAT 裝置的 mode：auto = 有 CiA 402 伺服時用 DC 同步，否則 free run；sync = DC 同步（失敗時改 free run）。master：plc = 由 86LDmicro 啟動主站（plc86_ecat.cpp）；sketch = 由你的 .ino 自行 begin、attach、start，完成後設定 `plc86_ecOk = true`。

4.5 全域變數與資料型別

86LDmicro 中所有變數都要先宣告才能使用：未宣告的名稱在階梯圖中以紅底顯示，檢查時列出錯誤。宣告在**全域變數表（GVL）**的變數，所有 POU、86HMI 與 C/C++ 都看得到；宣告在 POU 的 VAR 中的只有該 POU 可用。

4.5.1 全域變數表（GVL）

專案樹雙擊「全域變數 > GVL」開啟編輯器，一行宣告一個變數，按 Ctrl+S 存檔。編輯器有語法顏色與補全，`//` 或 `(* *)` 後面是註解（會顯示在補全清單與屬性中）。

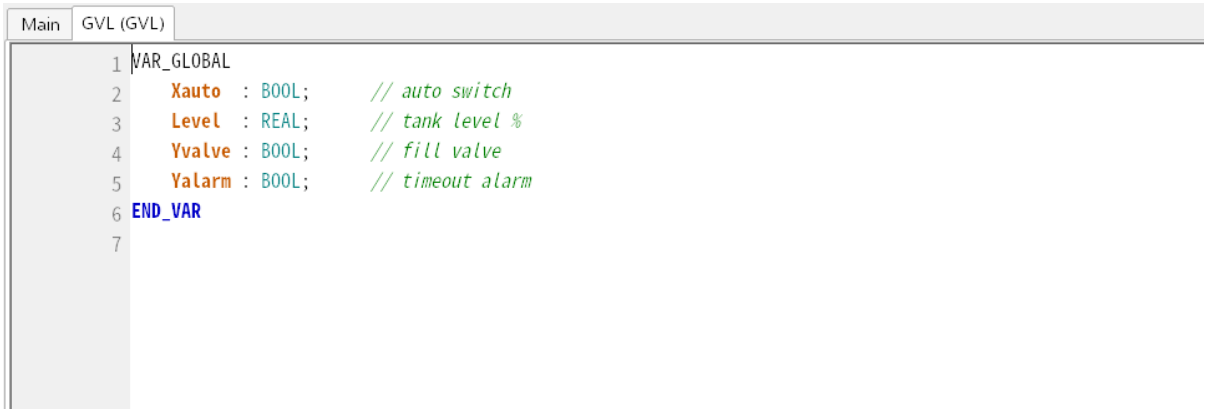


圖 4-10　GVL 編輯器：一行一個變數，// 後面是註解

```
VAR_GLOBAL
    Xstart   : BOOL;                    // 註解會顯示在補全清單與屬性中
    Speed    : REAL := 12.5;            // 初值
    Recipe   : ARRAY[1..5] OF INT;      // 陣列（索引 1～5）
    Axis1    : Axis_t;                  // 自訂結構（資料型別，見下方）
    Mode     : Mode_t;                  // 自訂列舉
END_VAR
VAR_GLOBAL RETAIN                       // 斷電保持：存在 EEPROM，重新上電後還原
    TotalRuns : DINT;
END_VAR
VAR_GLOBAL CONSTANT                     // 常數：程式中不能修改
    MAX_SPEED : REAL := 100.0;
END_VAR
```

區段	說明
VAR_GLOBAL	一般全域變數；`:=` 值 設定初值（沒寫時為 0 / FALSE / 空字串）
VAR_GLOBAL RETAIN	斷電保持：值存在控制器的 EEPROM，重新上電後還原；暖重啟保留、冷重啟清除（3.2.5 節）
VAR_GLOBAL CONSTANT	常數：以 `:=` 值 給定數值；程式中寫入它時，檢查會出錯

*注意：RETAIN 不是每次賦值就立即寫入：控制器約每秒檢查一次，有變動時才把新的快照分批寫入 EEPROM（在掃描之間進行），整筆寫完才生效。新快照還沒寫完就斷電時，重新上電會還原到上一筆已寫完的值。EEPROM 有寫入次數上限。不要把一直在變的變數（計數器、計時值、類比值）設成 RETAIN；需要保存時，只在必要時（停機、換配方）複製到 RETAIN 變數。

可依用途分成多個 GVL（「專案 > 新增全域變數表...」，例如 `IO`、`Recipe`），所有 GVL 的變數名稱不可重複。

4.5.2 POU 的區域變數

POU 的宣告區用 VAR（區域變數，功能塊實例通常放這裡，例如 tLamp : TON;）、VAR_TEMP（暫存：每次呼叫都從初值開始，不保留）；FUNCTION 與 FUNCTION_BLOCK 另有參數 VAR_INPUT、VAR_OUTPUT、VAR_IN_OUT（PROGRAM 沒有參數）。完整的支援範圍見 8.2.1 節。其他 POU 與監看視窗以 程式名.變數 讀取區域變數，例如 Main.tLamp.ET。

*提示：FUNCTION_BLOCK 與 FUNCTION 可以讀基本型別的全域變數與常數（例如 MAX_SPEED）；陣列與結構型別的全域變數不能在其中直接存取：請以 VAR_INPUT 傳入（傳入的是複本）；要讓它改寫陣列或結構時，用 VAR_IN_OUT 傳入呼叫端 ST 程式的區域變數，呼叫後再複製回全域變數（8.2.1 節）。

4.5.3 基本資料型別

型別	範圍 / 說明	型別	範圍 / 說明
BOOL	TRUE / FALSE	REAL	32 位元浮點
SINT / USINT	8 位元有號 / 無號	LREAL	64 位元浮點
INT / UINT	16 位元有號 / 無號	TIME	時間，例如 `T#1s500ms`、`T#100ms`
DINT / UDINT	32 位元有號 / 無號	STRING	字串，例如 `'hello'`
LINT / ULINT	64 位元有號 / 無號	BYTE / WORD / DWORD / LWORD	位元字串（8 / 16 / 32 / 64 位元）

4.5.4 資料型別（結構、列舉、陣列）

「專案 > 新增資料型別...」建立一個 `.dut` 檔，在 `TYPE ... END_TYPE` 中定義結構、列舉與具名陣列，之後就可以在 GVL 與 POU 中當型別使用：

```
TYPE
    Axis_t : STRUCT
        pos   : REAL;
        speed : REAL;
        ready : BOOL;
    END_STRUCT;
    Mode_t : (Idle, Auto, Manual);       // 列舉
    Recipe_t : ARRAY[1..5] OF INT;       // 具名陣列
END_TYPE
```

程式中以 `Axis1.speed`、`Recipe[2]`、`Mode := Auto;` 存取。新增資料型別時的預設內容是一個只有 `x : INT;` 的結構，可直接改寫。

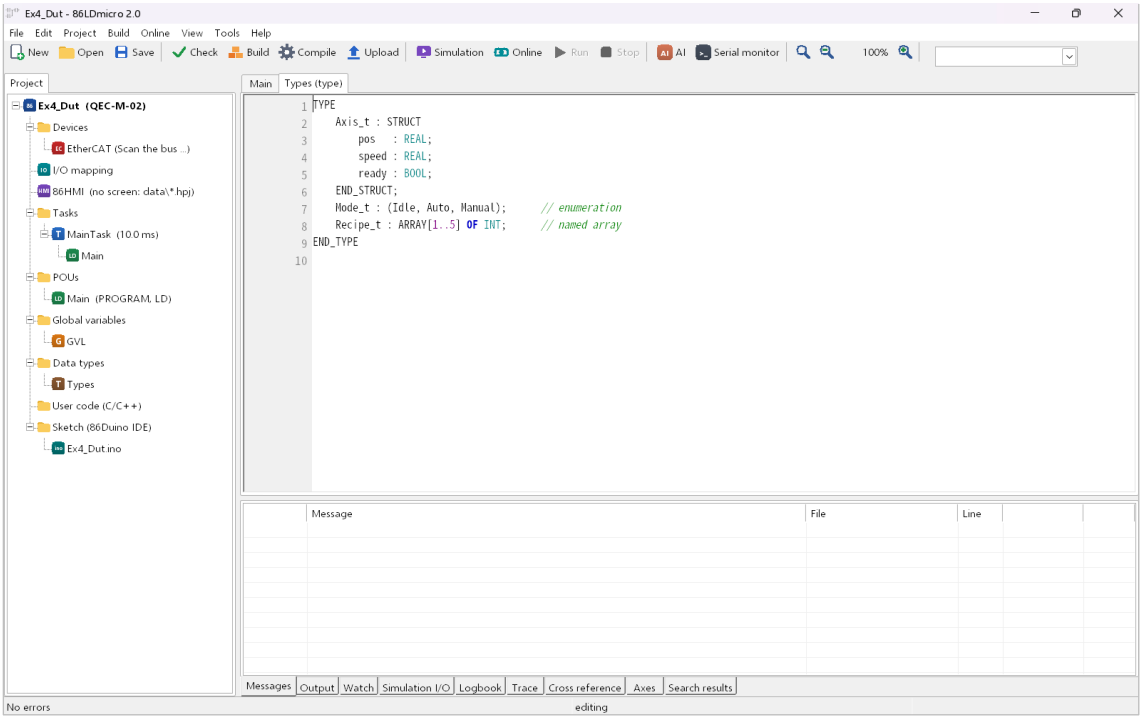


圖 4-11　資料型別編輯器：結構、列舉與具名陣列

4.5.5 新增變數的方法

方式	操作	變數放在
1. 直接編輯 GVL	專案樹雙擊「全域變數 > GVL」，輸入 `名稱 : 型別;`	全域
2. POU 的宣告區	LD / ST / SFC 分頁上方的 `VAR ... END_VAR` 中輸入	區域（只有該 POU 可用）
3. I/O 對應	新增 I/O 對應時在「變數」欄輸入新名稱（4.6 節）	自動加入第一個 GVL
4. 常用接線	插入常用接線時表單中的新名稱（7.1 節）	輸入 / 輸出 / 內部位元放 GVL，計時器等放該 POU 的 VAR
5. 86HMI 綁定	綁定元件時在「變數」欄輸入新名稱（13.2 節）	自動加入 GVL
6. Excel 匯入	I/O 對應頁「從 Excel 匯入...」（4.6 節）	指定的 GVL（RETAIN 放 RETAIN 區）
7. AI 助理插入程式碼	AI 回答的程式碼按「插入程式碼」（15.3 節）	VAR 區塊中的新變數自動宣告

***提示：**命名習慣：`X` 開頭為輸入、`Y` 為輸出、`R` 為內部位元、`T` 計時器、`C` 計數器、`M` 數值（與 LDmicro 1.1 相同），對應 I/O 時一眼就能分辨。名稱只能用字母、數字、`_`，以字母開頭，不分大小寫，不可使用 IEC 保留字（附錄 C）。

4.6 I/O 對應

PLC 程式只讀寫變數；變數與實體點（本機 I/O、EtherCAT PDO、86EVA、Arduino 腳位、Modbus）的對應在「I/O 對應」頁設定，程式不必知道接線。專案樹雙擊「I/O 對應」（或「專案 > I/O 對應」）開啟：

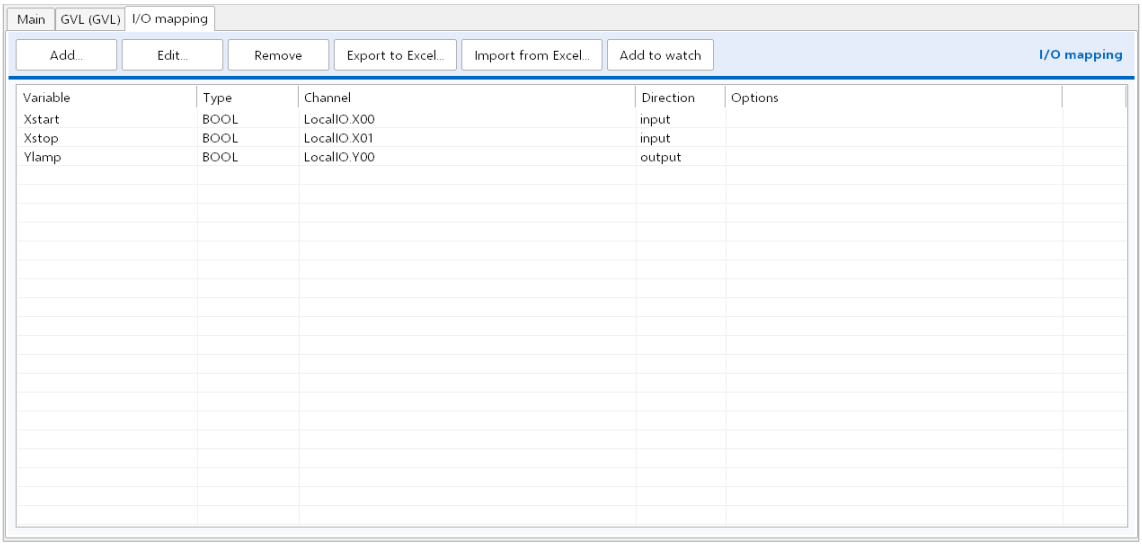


圖 4-12　I/O 對應頁：變數、型別、通道、方向、選項；上方六個按鈕

按鈕	功能
新增...	新增一個對應（「新增 I/O 對應」視窗）
編輯...	修改選取的對應；雙擊一列也可以
移除	移除選取的對應（變數本身不刪除）
匯出 Excel...	把變數與對應存成 .xlsx 或 .csv
從 Excel 匯入...	從 .xlsx 或 .csv 宣告變數並設定對應
加入監看	把選取的列（Ctrl / Shift 可多選）的變數加入「監看」視窗

4.6.1 新增 I/O 對應

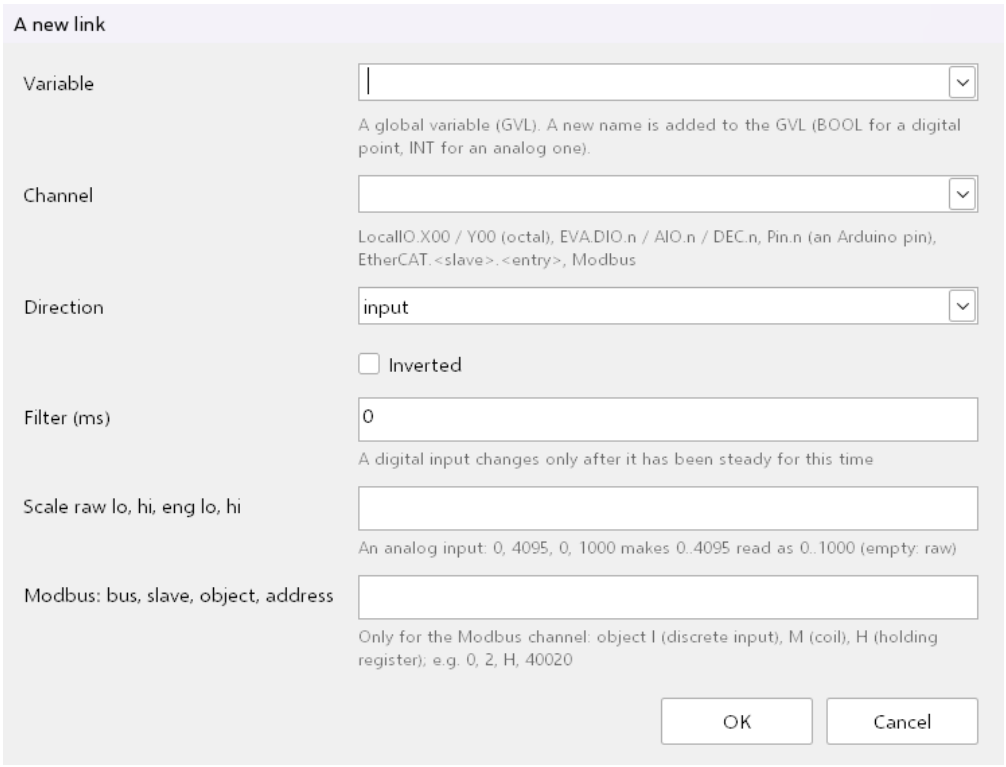


圖 4-13 「新增 I/O 對應」：變數、通道、方向、反相、濾波、換算、Modbus

欄位	說明
變數	全域變數（下拉列出 GVL 的變數）。輸入新名稱會自動加入第一個 GVL：數位點為 BOOL，類比點為 INT，EtherCAT 項目為該 PDO 的型別，並以通道名稱作為註解
通道	`LocalIO.X00`～`X17` / `Y00`～`Y17`（本機，八進位）、`EVA.DIO.n` / `AIO.n` / `DEC.n`、`Pin.n`（Arduino 腳位）、`EtherCAT.<從站>.<項目>`、`Modbus`；下拉列出可用的通道
方向	輸入 / 輸出。LocalIO 與 EtherCAT 的方向由通道決定，其他依此欄
反相	數位點反相（例如常閉接線的按鈕）
濾波（ms）	數位輸入需穩定維持此時間後才改變（去彈跳）
換算：原始下限, 上限, 工程下限, 上限	類比輸入，例如 `0, 4095, 0, 1000` 表示 0..4095 換算為 0..1000；空白為原始值
Modbus：匯流排, 從站, 物件, 位址	僅 Modbus 通道：物件 I（離散輸入）、M（線圈）、H（保持暫存器），例如 `0, 2, H, 40020`（需先新增 modbus 裝置，4.4 節）

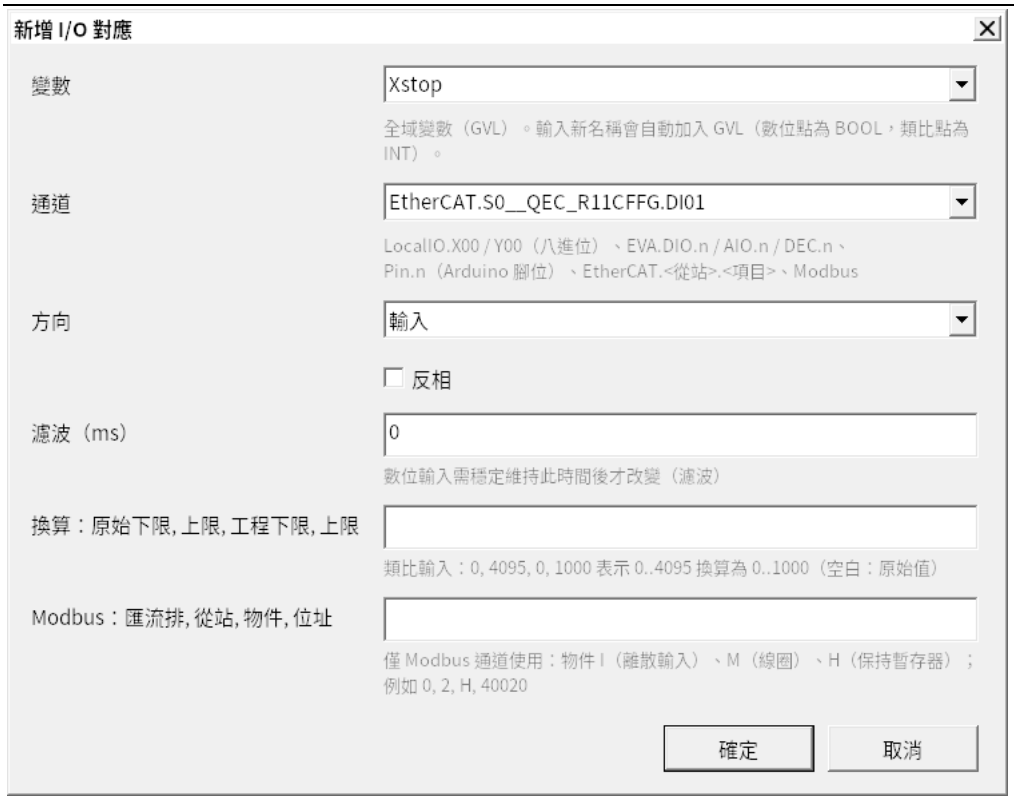


圖 4-14　EtherCAT 通道的對應：Xstop → EtherCAT.S0__QEC_R11CFFG.DI01

EtherCAT 的通道在讀入 ENI 後才出現（第 11 章）；也可以在專案樹的 PDO 項目上按右鍵「對應到變數...」。

4.6.2　加入監看

在 I/O 對應頁選取一列或多列（Ctrl / Shift 多選），按上方的 **加入監看** 或右鍵 **加入監看**，這些變數就加入「監看」視窗，狀態列顯示「已將 N 個變數加入監看視窗」；模擬或線上時直接看數值、強制。右鍵選單另有 **新增...**、**編輯...**、**移除**。

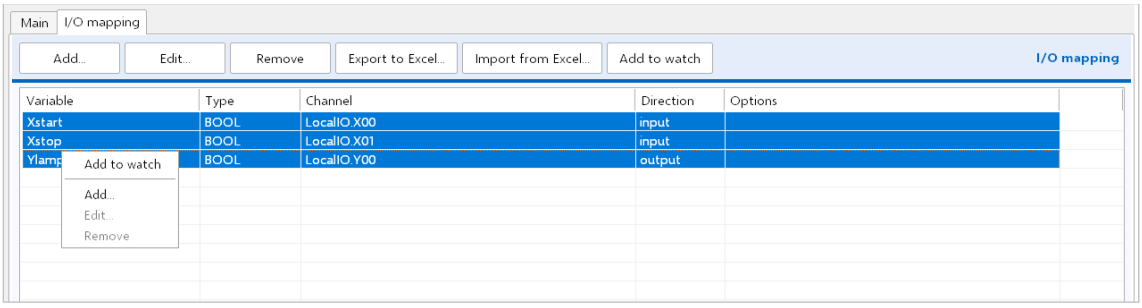


圖 4-15　I/O 對應頁：選取三列後按右鍵「加入監看」

4.6.3 I/O 表 Excel 匯入 / 匯出

I/O 很多時，可以在 Excel 中規劃後一次匯入。

1. I/O 對應頁按 **匯出 Excel...**（或「專案 > 匯出 I/O 表到 Excel...」），存成 .xlsx（也可存 .csv UTF-8）。存檔後自動以 Excel 開啟。
2. 「IO」工作表每個變數一列；「可用通道」工作表列出所有可用通道與目前對應的變數。
3. 在 Excel 中規劃或修改（欄位見下表），存檔。
4. 按 **從 Excel 匯入...**（或「專案 > 從 Excel 匯入 I/O 表...」）選檔。標題列中英文皆可、欄位順序不限。確認「共 N 列：將新變數宣告到 GVL 並設定 I/O 對應（這些變數原有的對應會被取代）？」後按 **是**。
5. 完成後顯示結果摘要（新增、修改的變數與略過的列）。要還原時，不存檔並重新開啟專案即可。

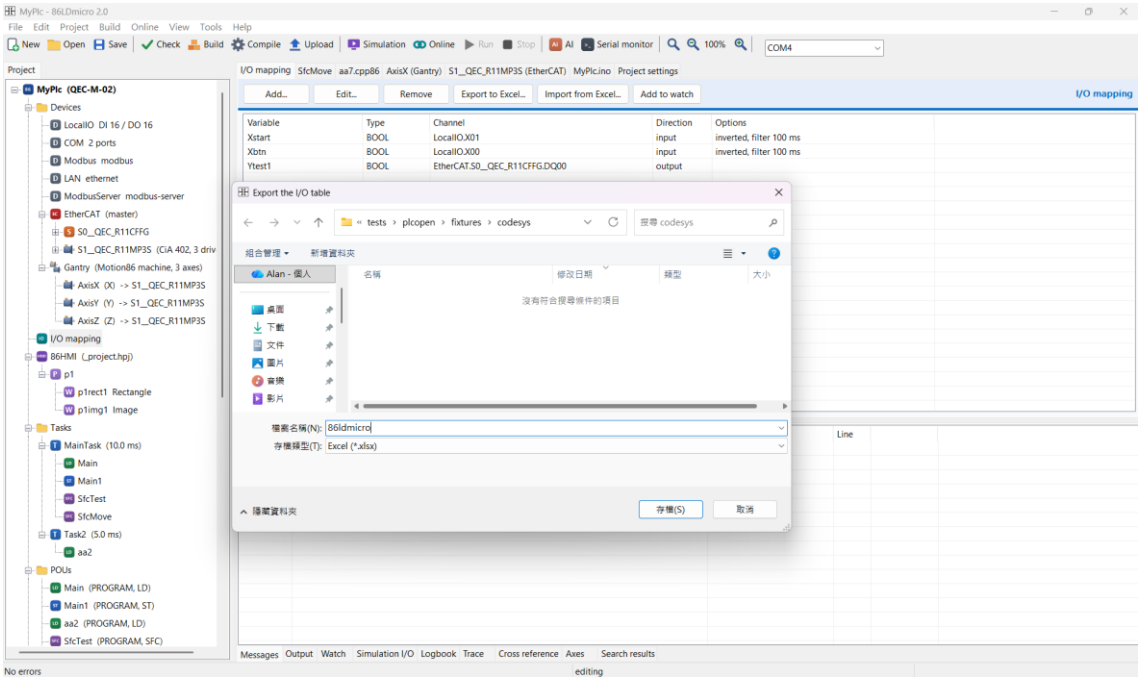


圖 4-16　匯出的 I/O 表（Excel）：「IO」工作表

欄位	說明
變數	變數名稱（必填）
型別	BOOL、INT、REAL…；空白時新變數為 BOOL、既有變數不改
初值、註解	寫入宣告
通道	與「新增 I/O 對應」相同；空白表示移除這個變數的對應
方向、反相、濾波 ms、換算（原始低, 原始高, 工程低, 工程高）	與「新增 I/O 對應」相同
斷電保持	新變數放進 GVL 的 RETAIN 區
GVL	新變數放進哪一個 GVL（空白：第一個）
其他（JSON）	其他對應屬性，例如 Modbus 的 bus、slave、object、address

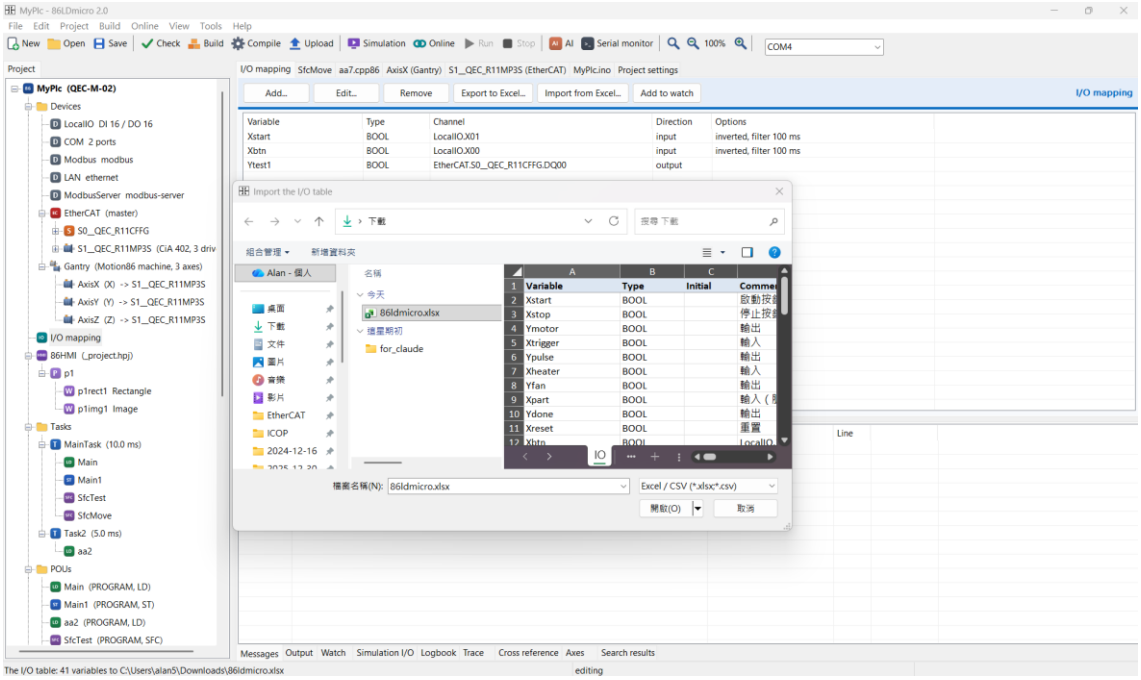


圖 4-17　從 Excel 匯入前的確認

***提示：**匯入只在編輯中可用；模擬或線上時會提示「請先停止模擬 / 離線後再編輯」。

同一個變數在表中出現兩次時只用第一列；名稱不合法的列會被略過，都列在結果摘要中。命令列：`86ldmicro-cli iotable export|import <專案> <檔案>`（15.5 節）。

4.7 新增 POU 與使用者程式

在「**專案**」選單（或專案樹對應節點的右鍵）新增，輸入名稱後按 **確定**，新檔案立即開啟。名稱只能用字母、數字、_，以字母開頭，且不能與專案中其他 POU、GVL、資料型別、使用者程式同名。

指令	建立的檔案	預設名稱	說明
新增 PROGRAM（階梯圖 / ST / SFC）...	`pou\<名稱>.lad` / `.st` / `.sfc`	Program	新的 PROGRAM 自動加入第一個工作
新增 FUNCTION_BLOCK（階梯圖 / ST）...	`pou\<名稱>.lad` / `.st`	MyFb	有內部狀態、可重複使用的功能塊（8.3 節）
新增 FUNCTION（ST）...	`pou\<名稱>.st`	MyFunction	沒有記憶的函式（8.3 節）
新增全域變數表...	`gvl\<名稱>.gvl`	GVL2	內容為空的 `VAR_GLOBAL ... END_VAR`
新增資料型別...	`dut\<名稱>.dut`	MyType	結構範本（4.5 節）
新增使用者程式檔（C/C++）...	`usercode\<名稱>.cpp86`	MyCode	附說明與掛勾函式範本（第 14 章）

*重要：新增 PROGRAM 的預設名稱 `Program` 與 IEC 保留字 PROGRAM 相同（名稱不分大小寫），直接按確定會在檢查時出現錯誤。請改成有意義的名稱，例如 `Filling`、`Conveyor`。

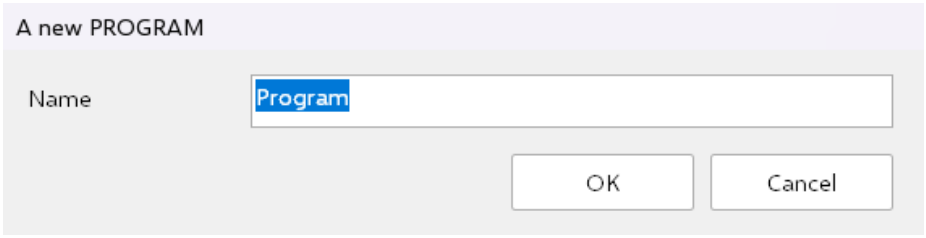


圖 4-18 「新增 PROGRAM」：輸入名稱

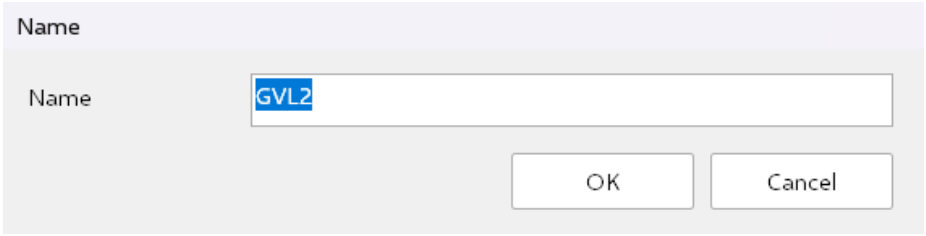


圖 4-19 「新增全域變數表」：輸入名稱（預設 GVL2）

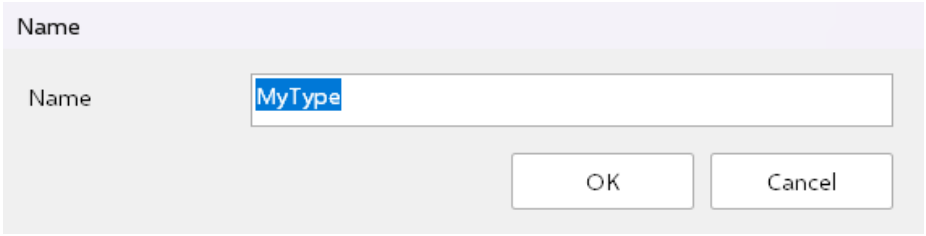


圖 4-20　「新增資料型別」：輸入名稱

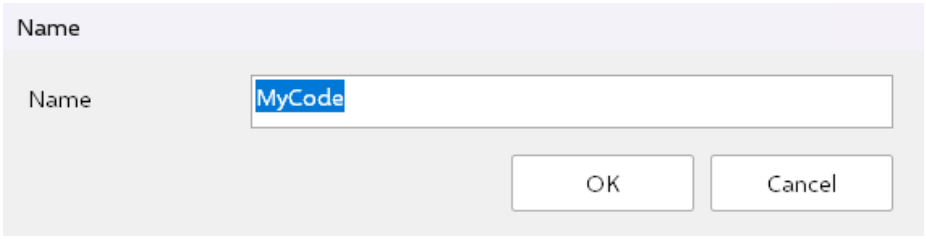


圖 4-21　「新增使用者程式檔（C/C++）」：輸入名稱

新檔案的內容是可直接編輯的範本，例如 ST 的 PROGRAM 與 FUNCTION_BLOCK：

```
PROGRAM Filling
VAR
END_VAR

END_PROGRAM

FUNCTION_BLOCK MyFb
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
END_VAR

END_FUNCTION_BLOCK
```

FUNCTION 的範本為 `FUNCTION MyFunction : INT`（回傳 INT，最後一行 `MyFunction := 0;`）；SFC 的範本是一個初始步驟 `Init` 與一個 1 秒後跳回的步驟 `Step1`（8.4 節）。

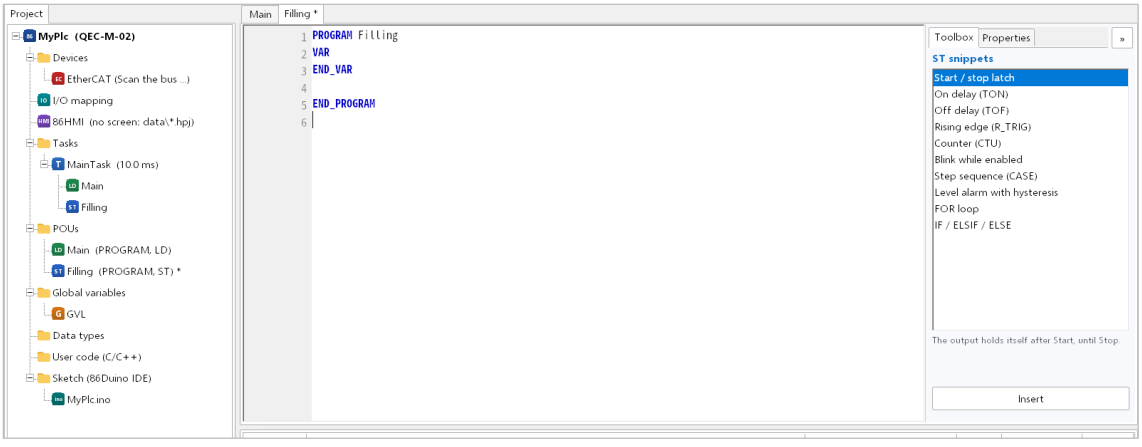


圖 4-22　新增的 ST PROGRAM Filling：已加入 MainTask 並開啟

4.7.1 使用者程式（C/C++）

使用者程式檔（`.cpp86`）與 PLC 一起編進 `plc86.cpp`。新檔案的開頭有說明：PLC 變數要寫成 `GVL(名稱)`（全域變數）、`ST(程式.變數)`（程式的變數）、`FUN(名稱)(參數)`（專案的 ST FUNCTION），並附上掛勾函式 `plc86_user_setup`、`plc86_user_before`、`plc86_user_after`、`plc86_user_loop` 等範本。詳細說明見第 14 章。

Main | SendMsg.cpp86

```
1  // SendMsg: when Nmsg changes, send one line on Serial1
2
3  void plc86_user_setup(void) {
4      Serial1.begin(9600);              // COM1, 9600 bps
5  }
6
7  // every pass of loop(): non-blocking work only (no delay())
8  void plc86_user_loop(void) {
9      static int32_t sent = 0;          // number already sent
10     int32_t n = GVL(Nmsg);
11     if(n != sent) {
12         sent = n;
13         Serial1.print("Button pressed, count = ");
14         Serial1.println(n);
15     }
16 }
17
```

圖 4-23　使用者程式編輯器：掛勾函式與 GVL() 存取 PLC 變數

- POU 改名：專案樹的 POU 右鍵「重新命名...」；工作中的程式名稱一起更新。
- 移除：右鍵「從專案移除」，檔案仍保留在 `data\86ld` 中。
- sketch 自己的 `.h` / `.cpp`：專案樹「Sketch（86Duino IDE）」右鍵「在 sketch 新增 .h / .cpp...」（14.5 節）。

第 5 章　快速入門：從空白到上傳

以下 12 個步驟示範一個最小但完整的流程：在 86Duino IDE 建立 sketch，開啟 86LDmicro 建立專案，宣告變數，畫出「啟動 / 停止自保 + 延遲 2 秒點亮指示燈」的階梯圖，對應 QEC-M-02 的本機 I/O，先在電腦上模擬，再建置、上傳到控制器並線上監看。

完成後的動作：按一下啟動按鈕（X00），內部位元 `Rrun` 變 TRUE 並自我保持；`Rrun` 持續 2 秒後指示燈 `Ylamp`（Y00）亮；按停止按鈕（X01），`Rrun` 與指示燈都熄滅。

變數	型別	I/O 對應	用途
`Xstart`	BOOL	`LocalIO.X00`（輸入）	啟動按鈕（常開）
`Xstop`	BOOL	`LocalIO.X01`（輸入）	停止按鈕（常開接線，程式中用常閉接點）
`Rrun`	BOOL	—（內部位元）	運轉中（自保）
`Ylamp`	BOOL	`LocalIO.Y00`（輸出）	運轉指示燈
`tLamp`	TON（IEC 功能塊）	—	指示燈延遲 2 秒（宣告在 Main 的 VAR）

***重要**：畫好階梯圖之後，一定要在「I/O 對應」把 `Xstart`、`Xstop`、`Ylamp` 對應到實體的 `LocalIO.X00`、`X01`、`Y00`（步驟 9）。沒有對應的變數只是記憶體中的位元：檢查會通過（輸入沒有對應時，建置只會出現警告），但上傳後實體 I/O 不會動作。

***提示**：開始前請先完成第 2 章：86Duino Coding 502 與 86LDmicro 已安裝、QEC-M-02 已用 USB 線接到電腦，並知道它的 COM 埠（2.4 節）。

5.1 步驟 1：開啟 86Duino IDE，建立 sketch，選擇開發板與序列埠

開啟 86Duino Coding 502。按「**檔案 > 新增**」（Ctrl+N）建立新的 sketch，再按「**檔案 > 另存新檔...**」（Ctrl+Shift+S）存成 `MyPlc`。sketch 的名稱也就是之後 86LDmicro 專案的名稱。



圖 5-1　86Duino IDE 的「檔案」選單：新增、另存新檔

按「**工具 > 開發板**」選 **QEC M02**，再按「**工具 > 序列埠**」選控制器的 COM 埠。IDE 視窗右下角會顯示「QEC M02 於 COMx」。

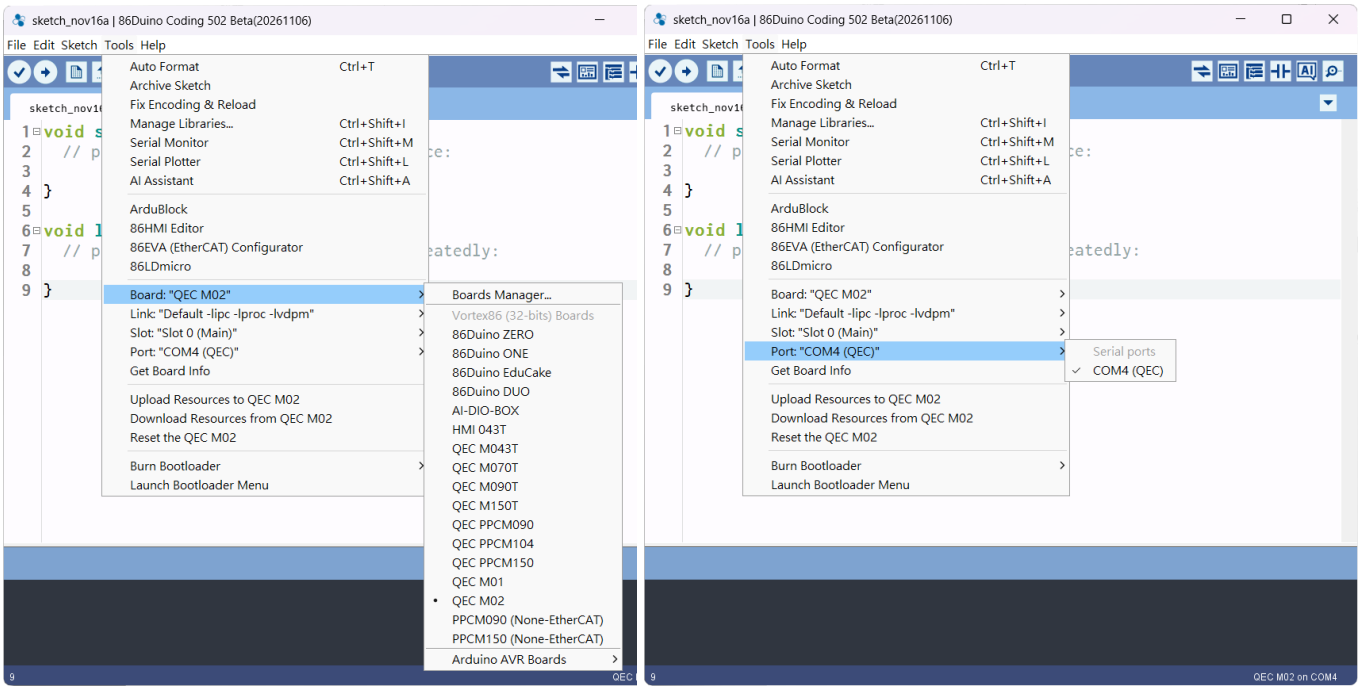


圖 5-2 & 圖 5-3　「工具 > 開發板」選 QEC M02、「工具 > 序列埠」選擇控制器的 COM 埠

5.2 步驟 2：從 IDE 開啟 86LDmicro，建立新專案

按「**工具 > 86LDmicro**」，或按 IDE 工具列右側的 **86LDmicro** 按鈕。這個 sketch 還沒有 86LDmicro 專案，因此會出現詢問視窗，按 **是(Y)** 建立新專案。

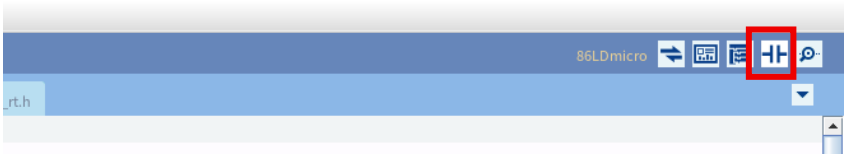


圖 5-4　IDE 工具列右側的 86LDmicro 按鈕

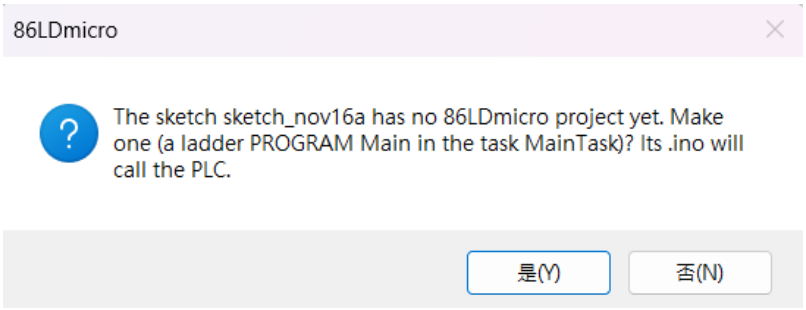


圖 5-5　sketch 還沒有 86LDmicro 專案時的詢問

- 從 IDE 開啟時，86LDmicro 自動使用該 IDE 的編譯器與 IDE 選的序列埠，不必另外設定。
- `.ino` 會自動加上 `#include "plc86.h"`、`plc86_setup()`、`plc86_loop()`，其他程式碼不動。
- 也可以單獨啟動 86LDmicro，按「**檔案 > 新增專案...**」（Ctrl+N），輸入名稱 `MyPlc` 與位置（預設在 文件\86Duino），按 **確定**（第 2.5 節）。

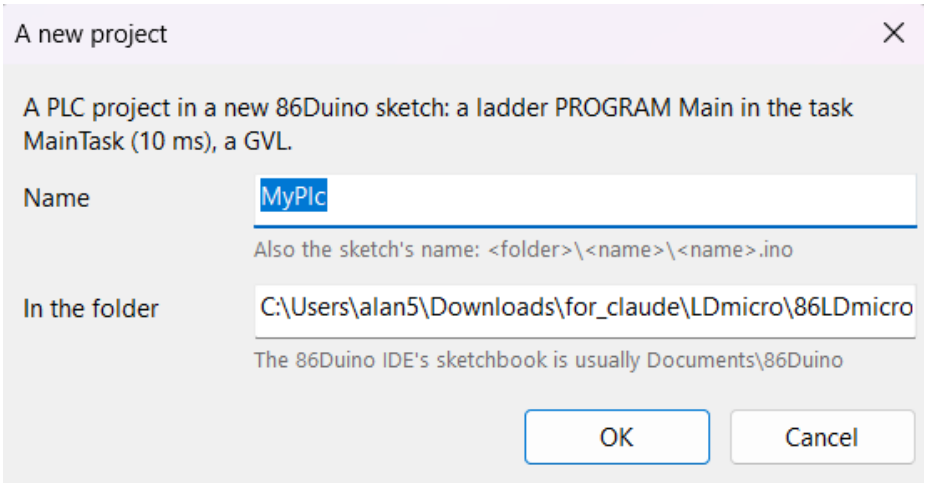


圖 5-6　單獨啟動時的「新增專案」：名稱也是 sketch 的名稱

5.3 步驟 3：認識新專案：專案樹與階梯圖 Main

新專案已經建立好下列項目，並自動開啟階梯圖 `Main`：

專案樹節點	內容
裝置	控制器的裝置（本機 I/O、EtherCAT 等，第 4.4 節）
I/O 對應	變數與實體通道的對應表（步驟 9）
工作（Task）> MainTask（10.0 ms）	每 10 ms 執行一次，執行 `Main`
POU > Main（PROGRAM, LD）	階梯圖程式，目前只有一個空白的網路 1
全域變數 > GVL	全域變數表（步驟 4）
Sketch（86Duino IDE）	sketch 的 `.ino`

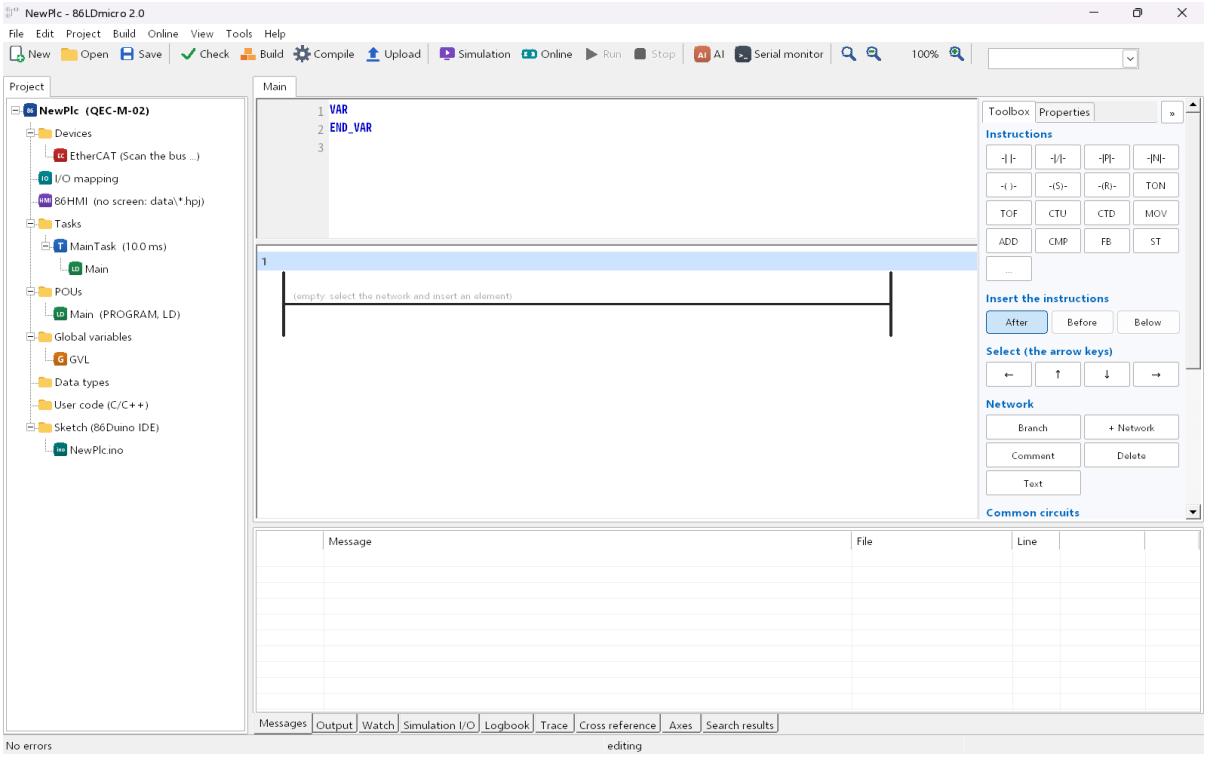


圖 5-7　新專案：裝置、I/O 對應、工作、POU（Main）、GVL 都已建立

階梯圖分頁分成三部分：上方是 **宣告區**（`VAR ... END_VAR`，放這個 POU 自己的變數與功能塊實例），中間是 **網路**，右側欄是 **工具箱**（「指令」、「指令插入位置」、「選取（方向鍵）」、「網路」、「常用接線」）。各區的詳細說明見第 3.5 節。

5.4 步驟 4：在 GVL 宣告變數

86LDmicro 2.0 的每個變數都要先宣告並指定型別。在專案樹雙擊「全域變數 > GVL」，以下列內容取代 GVL 的全部內容（含 VAR_GLOBAL 與 END_VAR），按 Ctrl+S 存檔：

```
VAR_GLOBAL
    Xstart : BOOL;      // 啟動按鈕（常開）
    Xstop  : BOOL;      // 停止按鈕（常開接線）
    Rrun   : BOOL;      // 運轉中
    Ylamp  : BOOL;      // 運轉指示燈
END_VAR
```

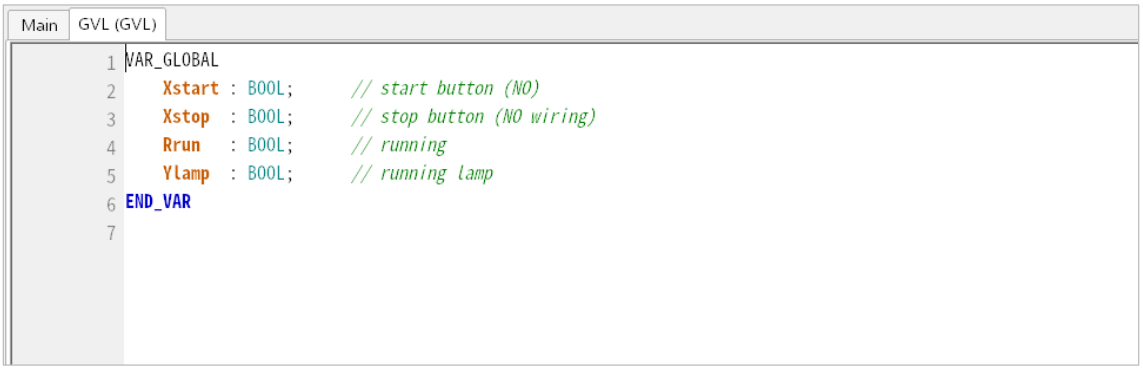


圖 5-8　GVL：宣告 Xstart、Xstop、Rrun、Ylamp

*提示：命名習慣：X 開頭為輸入、Y 為輸出、R 為內部位元（同 LDmicro 1.1），對應 I/O 時一眼就能分辨。名稱只是習慣，型別由宣告決定。

階梯圖中用到尚未宣告的名稱時，元件會以紅底顯示錯誤（第 6.0 節）。宣告語法與資料型別見第 4.5 節。

5.5 步驟 5：插入常開接點 Xstart

在專案樹雙擊「**POU > Main**」回到階梯圖。確認右側欄「**指令插入位置**」選的是 **插在後**。

1. 點一下網路 1（空白網路）選取它。
2. 按右側欄「**指令**」的 **-| |-**（常開接點），或直接按鍵盤 **C**。
3. 在跳出的視窗中，「變數」欄的下拉列出所有已宣告的變數，選 `Xstart`，按 **確定**。

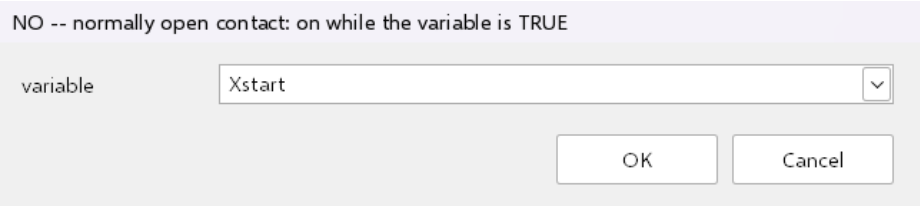


圖 5-9　常開接點（NO）的參數視窗：選擇變數

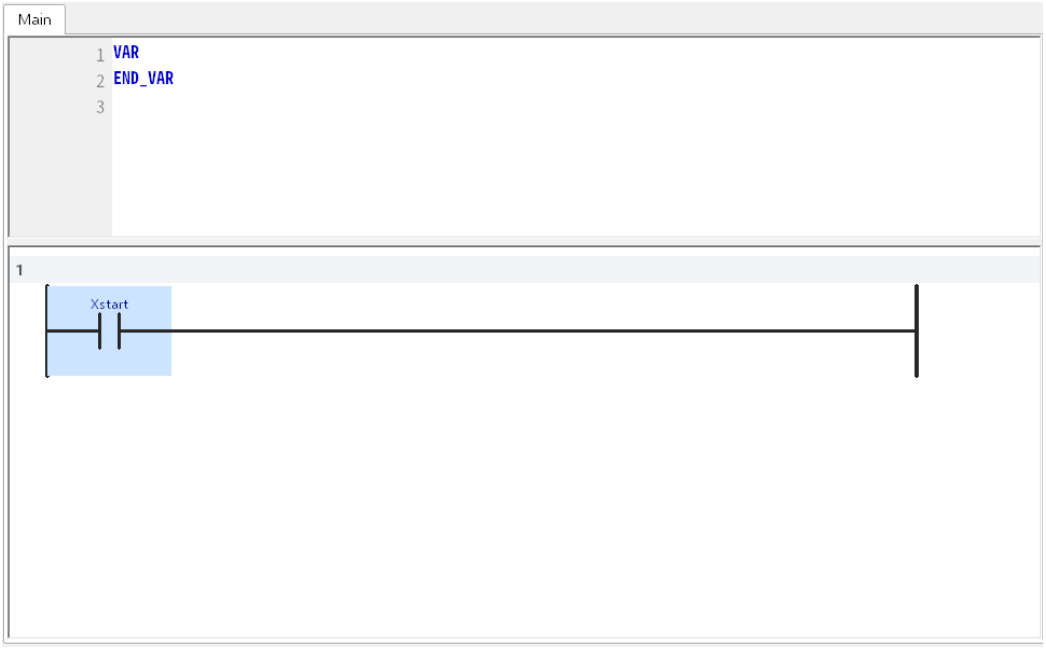


圖 5-10　網路 1 出現 Xstart 常開接點（藍底表示目前選取）

5.6 步驟 6：串聯常閉接點 Xstop 與線圈 Rrun

1. 在 `Xstart` 仍被選取時，按 **-|/|-**（常閉接點，鍵盤 **N**），變數選 `Xstop`，按 **確定**。新元件插在選取元件之後，也就是**串聯**。
2. 按 **-( )-**（線圈，鍵盤 **L**），變數選 `Rrun`，按 **確定**。線圈會自動放在網路的右側（右母線旁）。

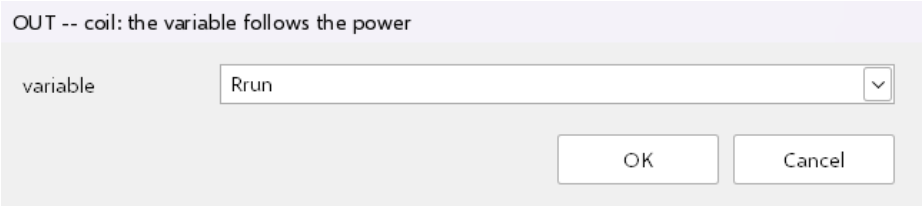


圖 5-11　線圈（OUT）的參數視窗

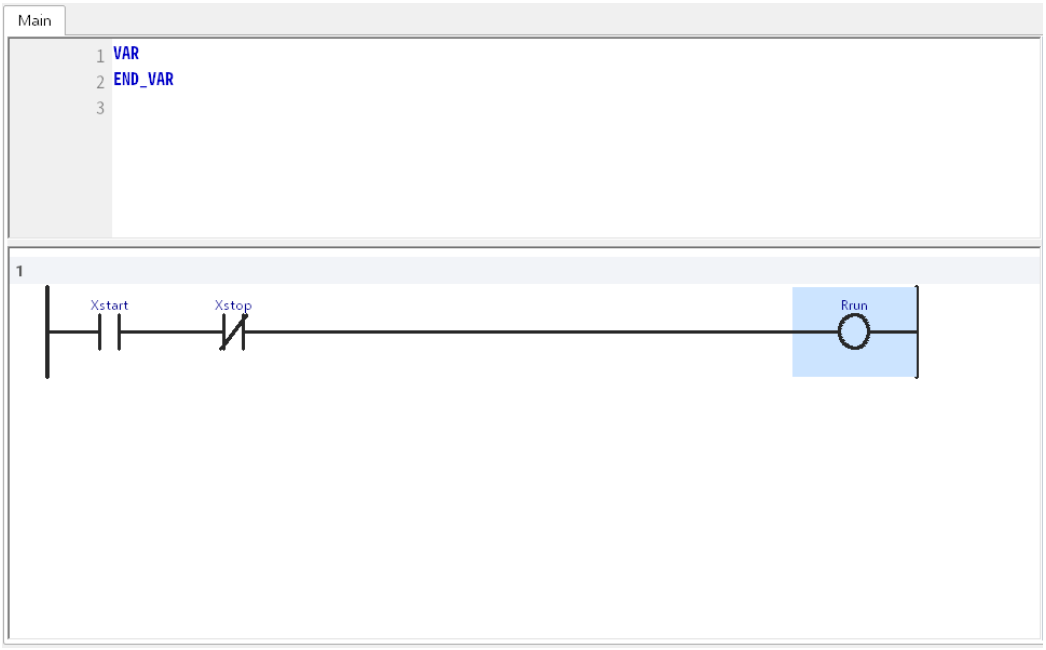


圖 5-12　Xstart、Xstop 串聯，驅動線圈 Rrun

*提示：停止按鈕若採常閉接線（斷線時等同按下停止，較安全），程式中改用常開接點 `NO(Xstop)`，或在步驟 9 的 I/O 對應勾選「反相」。

5.7 步驟 7：並聯 Rrun 接點，形成自保迴路

啟動按鈕放開後，要靠 Rrun 自己的接點維持導通，因此在 Xstart 下方並聯一個 Rrun 常開接點：

1. 按右側欄「選取（方向鍵）」的 ← （或鍵盤 ←）兩次，把選取移到 Xstart。
2. 「指令插入位置」改按 **並聯下方**。
3. 按 **-| |-**（或 **C**），變數選 Rrun，按 **確定**：Rrun 接點並聯在 Xstart 下方。
4. 把「指令插入位置」改回 **插在後**。

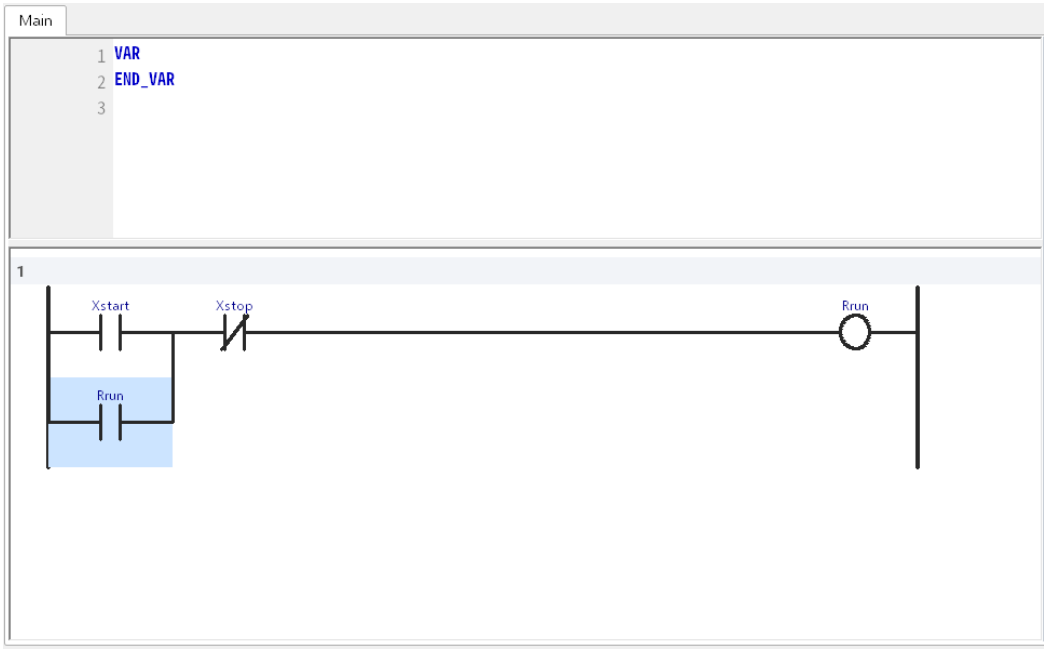


圖 5-13　Rrun 並聯在 Xstart 下方：按 Xstart 後 Rrun 自我保持，按 Xstop 解除

> ***提示**：快捷方式：選取 Xstart 後直接按鍵盤 **B**，會在它下方並聯一個常開接點（再輸入變數 Rrun），不必切換「指令插入位置」。

5.8 步驟 8：新增網路 2：TON 計時器延遲 2 秒點亮指示燈

先在 `Main` 上方的宣告區宣告一個 IEC 的 TON 計時器實例：在 `VAR` 與 `END_VAR` 之間輸入 `tLamp : TON;`。

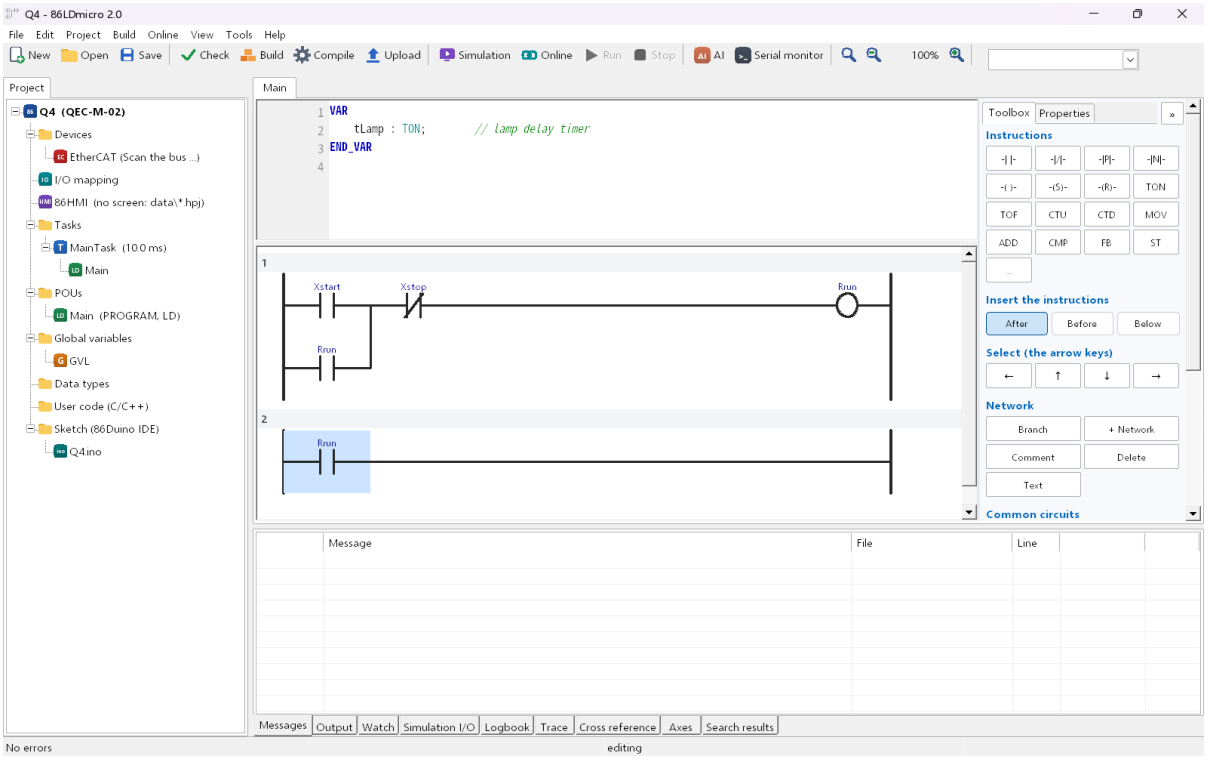


圖 5-14　宣告區加入 tLamp : TON（指示燈延遲計時器）

1. 點網路 1 的標題列（編號 1 那一列）選取網路，按右側欄「網路」的 **+ 網路**（或 **Ctrl+Enter**），在下方新增網路 2 並選取它。
2. 按 **-| |-**（**C**），變數選 `Rrun`。
3. 按 **FB**（功能塊，鍵盤 **K**），在「元件」欄輸入 `CALL(tLamp, IN := @, PT := T#2s, Q => @)`，按 **確定**。`@` 代表梯級的能流：左邊進來的能流接到 IN，Q 的輸出成為往右的能流。
4. 按 **-( )-**（**L**），變數選 `Ylamp`。

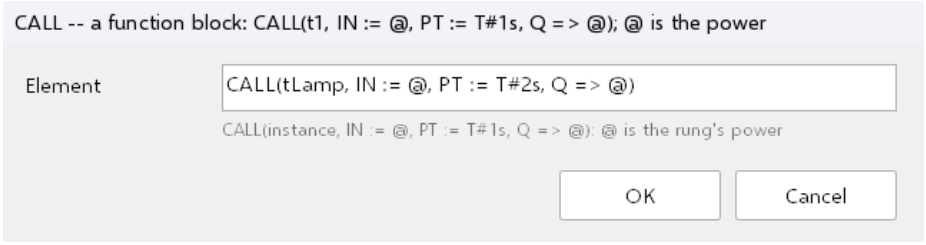


圖 5-15　功能塊（CALL）的參數：IN 接能流、PT 為 2 秒、Q 輸出接回能流

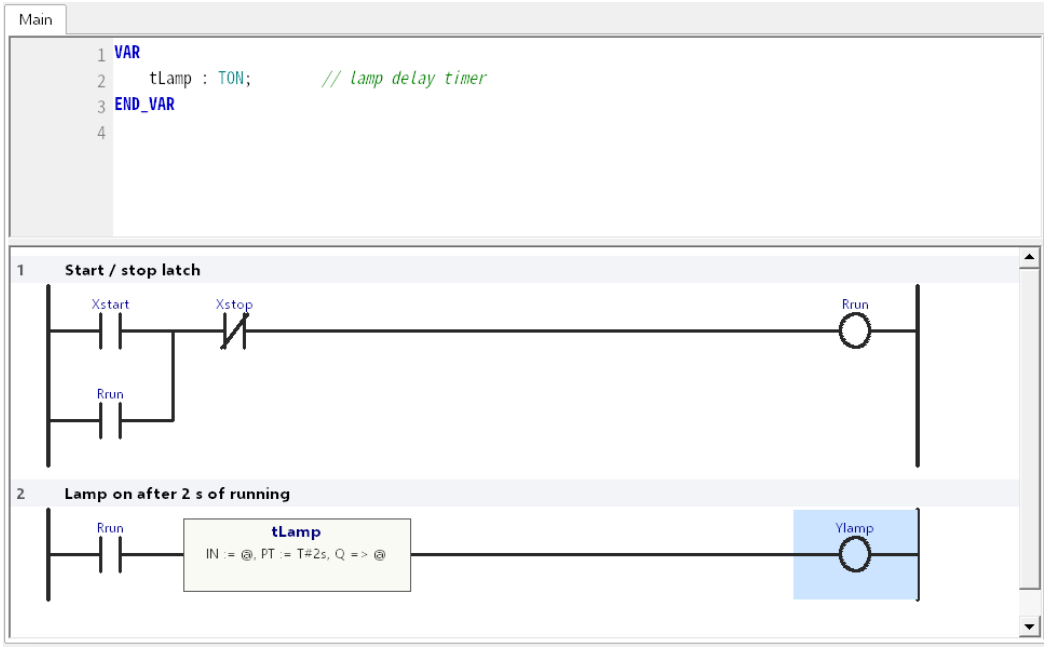


圖 5-16　完成的兩個網路：Rrun 持續 2 秒後 Ylamp 亮

最後替網路加上標題：雙擊網路 1 的標題列，「標題」輸入 啟動 / 停止自保，按 **確定**；網路 2 輸入 運轉 2 秒後指示燈亮。視窗下方的「內容（文字）」就是這個網路的 .lad 文字。

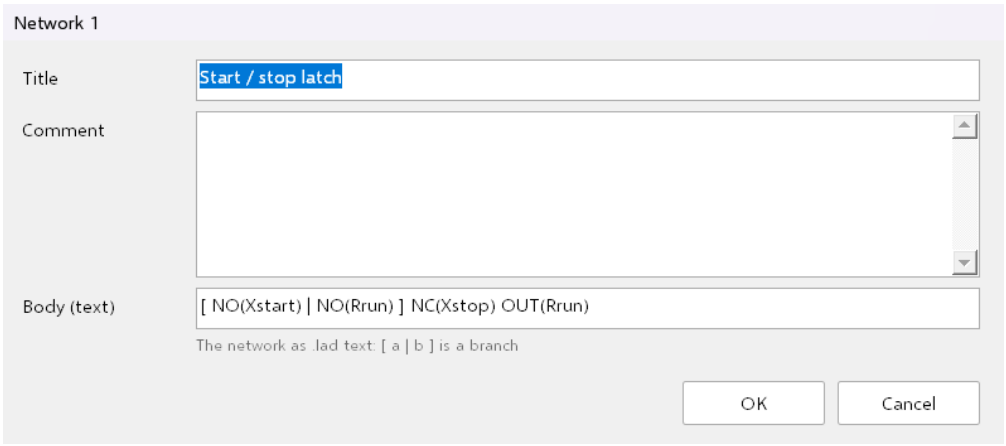


圖 5-17　網路的標題、註解與內容（.lad 文字）

按右側欄「網路」的 **文字**（Ctrl+T）可以看到整個 Main 的 .lad 文字，應與下列相同（按 **回到圖形** 返回）：

```
PROGRAM Main
VAR
    tLamp : TON;        // 指示燈延遲計時器
END_VAR
NETWORK '啟動 / 停止自保'
    [ NO(Xstart) | NO(Rrun) ] NC(Xstop) OUT(Rrun)
NETWORK '運轉 2 秒後指示燈亮'
    NO(Rrun) CALL(tLamp, IN := @, PT := T#2s, Q => @) OUT(Ylamp)
END_PROGRAM
```

5.9　步驟 9：I/O 對應：把變數接到 QEC-M-02 的實體 I/O

目前 `Xstart`、`Xstop`、`Ylamp` 只是記憶體中的變數，必須對應到控制器的實體點，上傳後按鈕與指示燈才會動作。

1. 在專案樹雙擊「**I/O 對應**」，按 **新增...**。
2. 「變數」選 `Xstart`，「通道」選 `LocalIO.X00`，按 **確定**（本機 I/O 的方向由通道決定，X 為輸入）。
3. 再按 **新增...**：`Xstop` 接 `LocalIO.X01`。
4. 再按 **新增...**：`Ylamp` 接 `LocalIO.Y00`（輸出）。
5. 按 **Ctrl+S** 存檔。

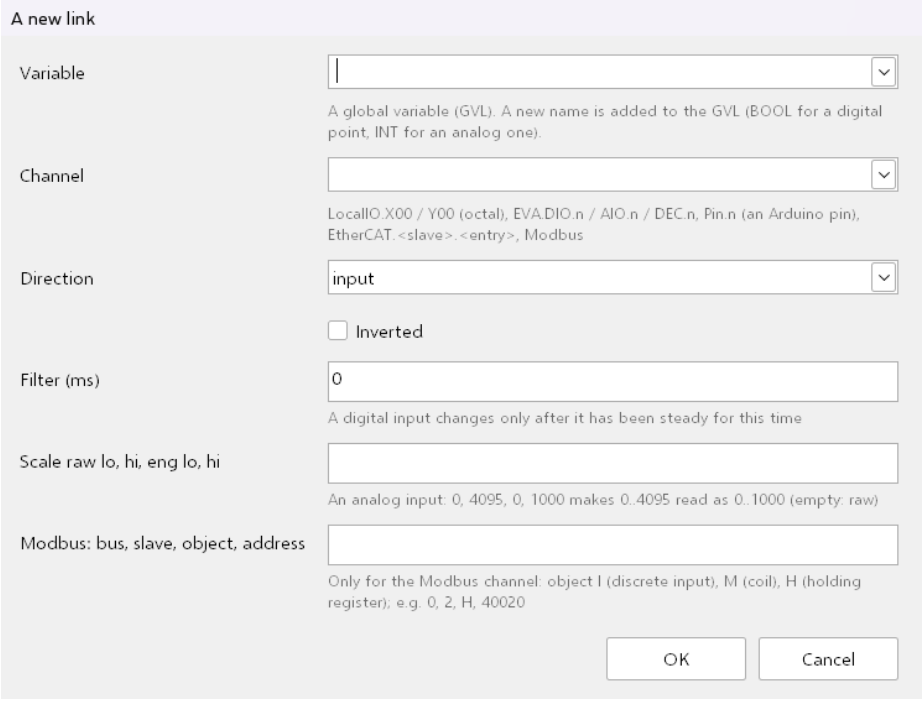


圖 5-18　新增 I/O 對應：變數、通道、方向、反相、濾波、換算

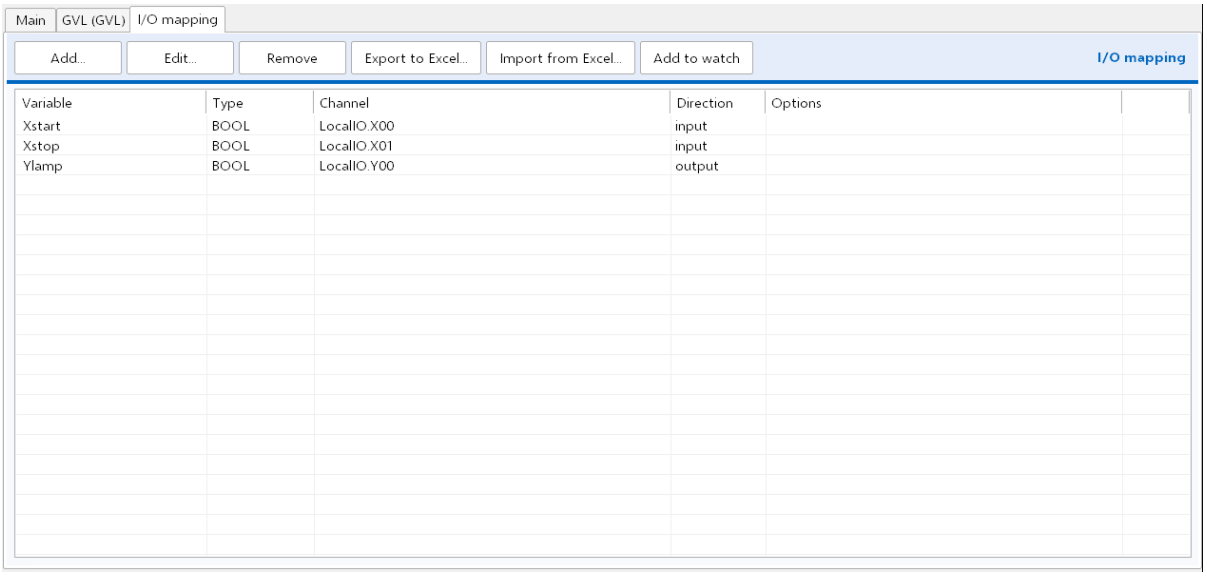


圖 5-19　I/O 對應頁：每一列是一個變數與通道的對應

- QEC-M-02 本機 I/O 的通道為 `LocalIO.X00`～`X17`（輸入）與 `LocalIO.Y00`～`Y17`（輸出），以**八進位**編號（X07 之後是 X10）。
- `Rrun` 與 `tLamp` 是內部變數，不需要對應。
- 也可以在階梯圖的變數上按右鍵「**Xstart 的 I/O 對應...**」直接開啟對應視窗。EtherCAT 模組的通道見第 11 章，I/O 對應的完整說明見第 4.6 節。

5.10 步驟 10：檢查（Shift+F7）與模擬（F5）

上傳之前先在電腦上確認程式邏輯，可以省下大量「燒錄、測試、再修改」的時間。

1. 按工具列 **檢查**（Shift+F7）。錯誤列在下方「**訊息**」視窗，雙擊會跳到出錯的元件；狀態列顯示「沒有錯誤」即可繼續。
2. 按工具列 **模擬**（F5）。狀態列顯示「模擬：RUN」與掃描次數，程式在電腦上開始執行。
3. 切到下方的「**模擬 I/O**」視窗：列出每個 I/O 對應的實體點。雙擊 `LocalIO.X00` 一次（ON）再雙擊一次（OFF），模擬按一下啟動按鈕。
4. 觀察階梯圖：有能流的線段變成綠色，元件下方顯示 TRUE / FALSE。`Rrun` 變 TRUE 並保持；`tLamp` 方塊計時，2 秒後 `Ylamp` 變 TRUE，「模擬 I/O」中 `LocalIO.Y00` 顯示 ON。
5. 雙擊 `LocalIO.X01` 兩次（ON、OFF）模擬按停止：`Rrun`、`Ylamp` 回到 FALSE。
6. 再按一次 **模擬**（F5）結束模擬，回到編輯狀態。

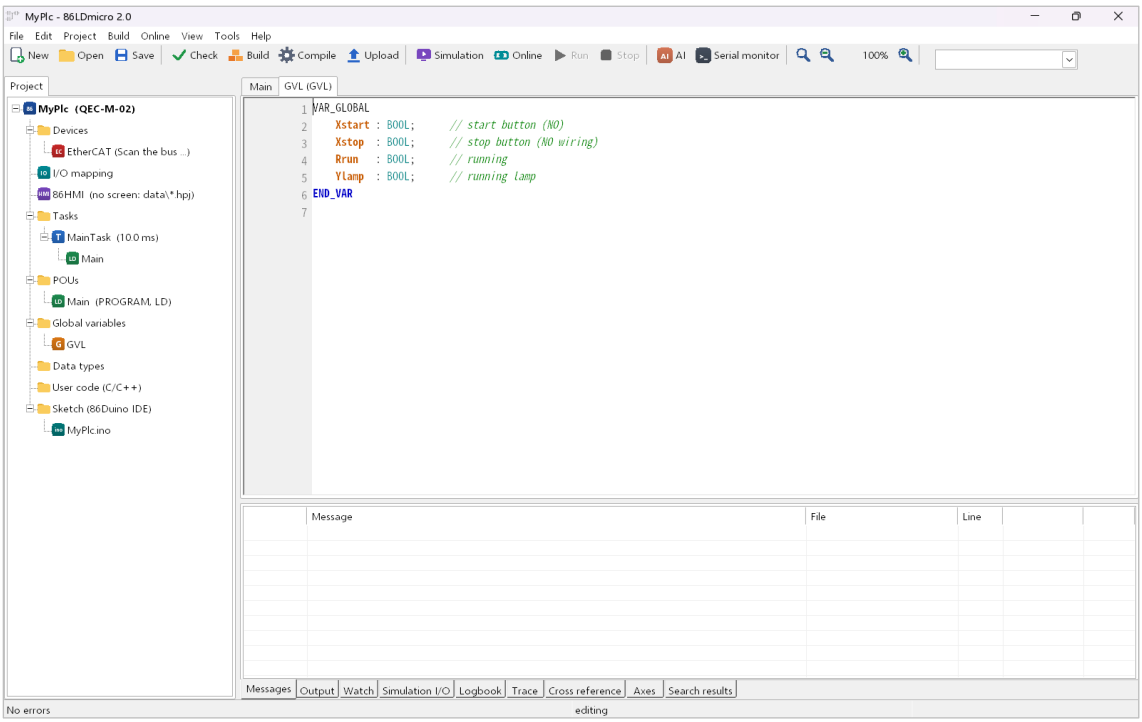


圖 5-20　檢查完成：訊息視窗沒有錯誤，狀態列顯示「沒有錯誤」

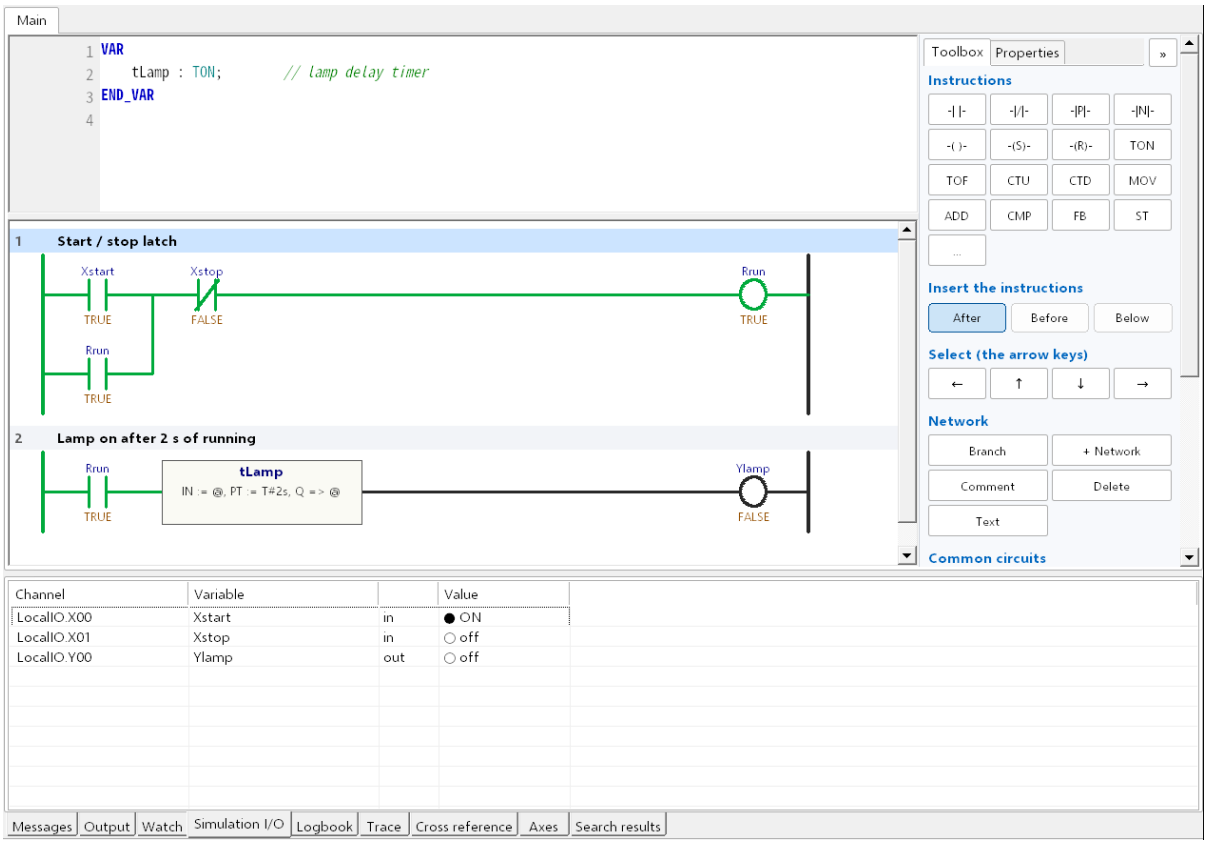


圖 5-21　模擬開始：「模擬 I/O」列出 LocalIO.X00（Xstart）等實體點

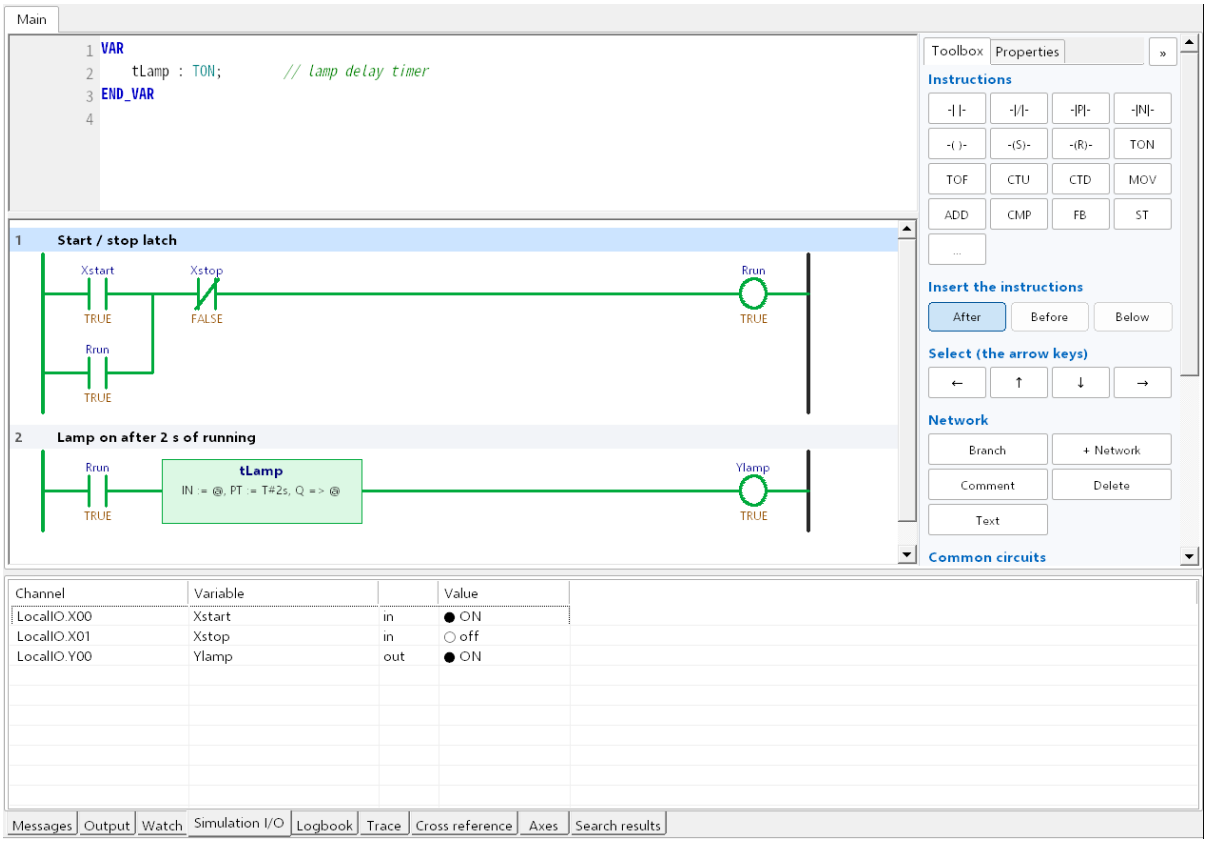


圖 5-22　模擬中：綠色為能流，計時器方塊顯示經過時間，LocalIO.Y00 為 ON

需要同時看多個變數時，切到「**監看**」視窗，在上方輸入變數名稱（例如 `Rrun`、`Main.tLamp.ET`）按 **加入**，即可看到即時數值（第 9.3 節）。

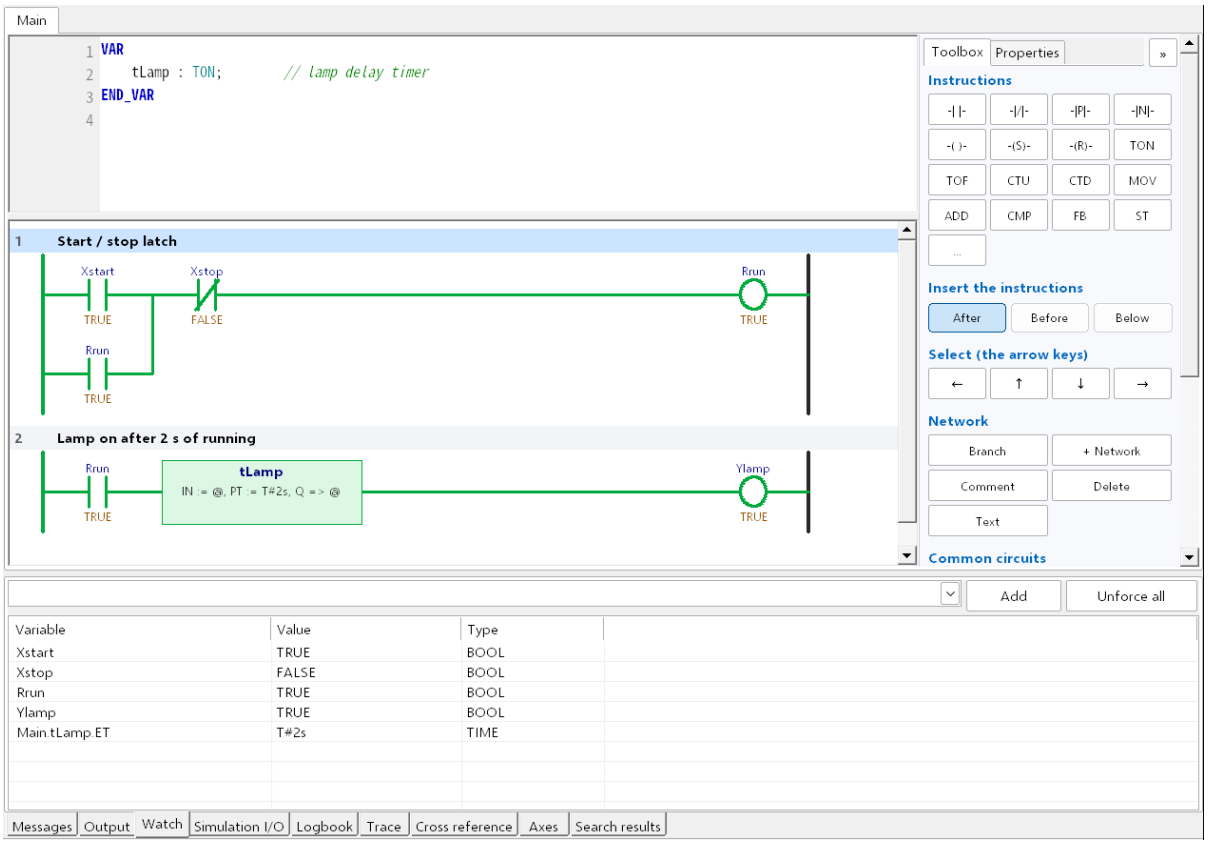


圖 5-23　監看視窗：加入變數後即時顯示數值

> ***注意**：已對應實體輸入的變數（例如接到 `LocalIO.X00` 的 `Xstart`）在模擬時由「模擬 I/O」決定；在監看中直接寫入，下一個掃描就會被輸入值覆蓋。請在「模擬 I/O」切換輸入（第 9.2 節）。

5.11 步驟 11：建置、編譯並上傳（F8）

1. 確認 QEC-M-02 以 USB 線接到電腦，在工具列右側的序列埠下拉選單選擇控制器的 COM 埠（從 IDE 開啟時已自動帶入 IDE 選的埠）。
2. 按工具列 **上傳**（F8）。86LDmicro 依序 **建置**（把專案寫成 sketch 的 `plc86.cpp` 等檔案）、**編譯**（使用 86Duino IDE 的編譯器）、**上傳**，過程顯示在下方「**輸出**」視窗的「**建置輸出**」。
3. 輸出依序出現「建置：已寫入...」、「== 編譯完成...」、「== 上傳到 COMx ...」與進度，最後「== 上傳完成」。控制器自動重新啟動並開始執行 PLC 程式。

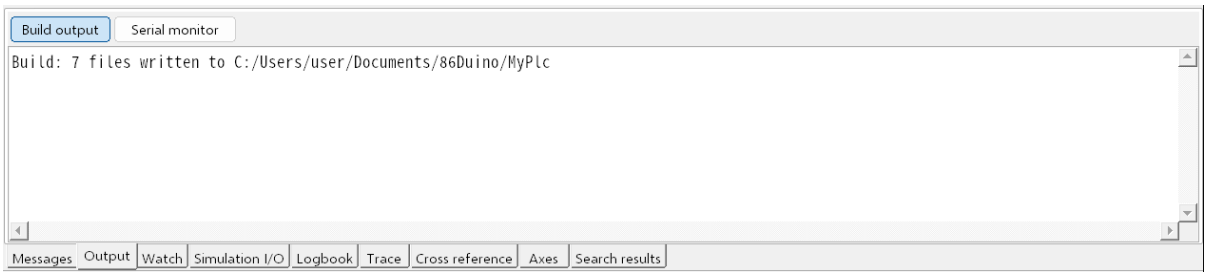


圖 5-24　輸出視窗的「建置輸出」：建置、編譯與上傳的過程都顯示在這裡

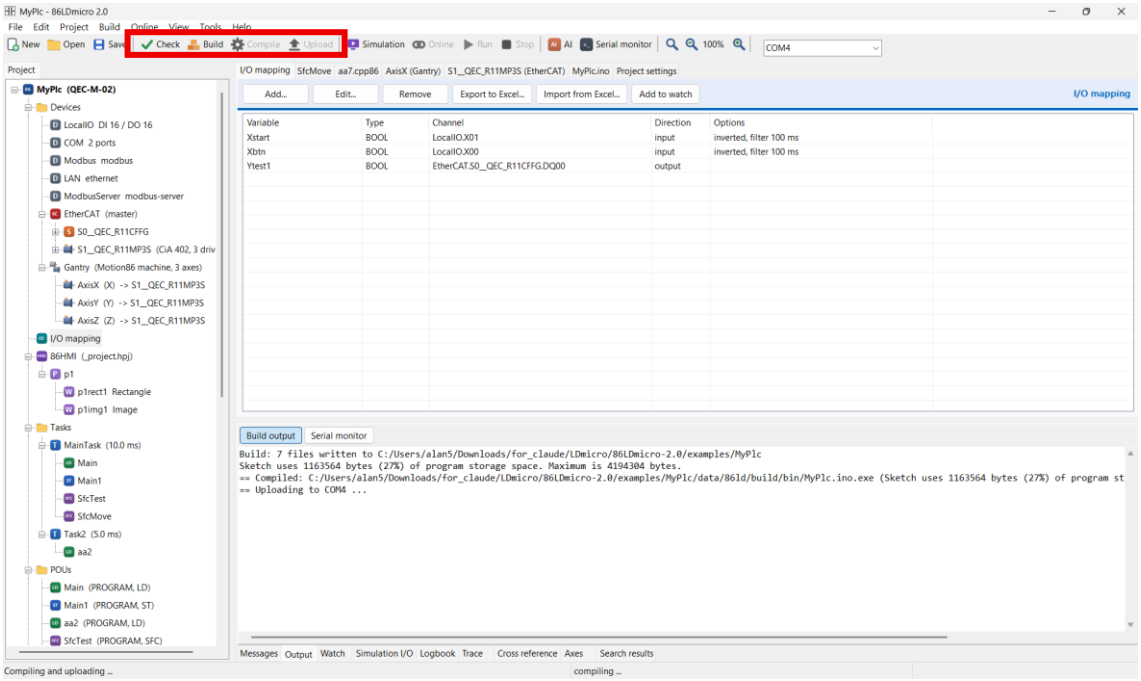


圖 5-25　上傳中

建置後，從 IDE 開啟的 86LDmicro 會通知 IDE 重新載入 sketch：IDE 出現 `plc86.cpp`、`plc86.h` 等分頁。之後也可以直接在 IDE 編譯、上傳。

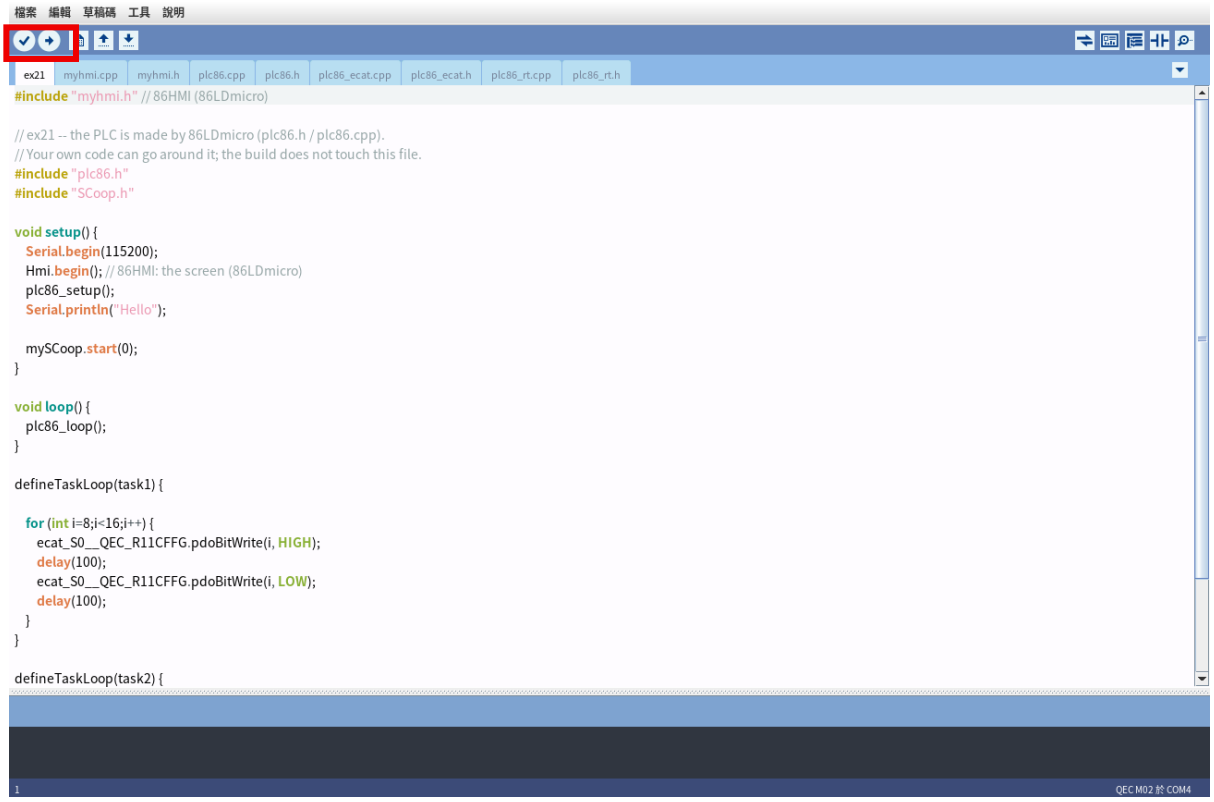


圖 5-26　建置後的 86Duino IDE：sketch 多了 plc86.cpp、plc86.h 等分頁

> ***注意**：IDE 中尚未存檔的 `.ino` 修改會在建置後被重新載入覆蓋，請先在 IDE 存檔。
>
> 上傳失敗「無法開啟 COMx」：關閉其他使用同一個 COM 埠的程式（例如 IDE 的序列埠監控視窗）後再按 F8。其他訊息與處理見第 10.1 節。

5.12　步驟 12：線上監看（Ctrl+F5）

1. 上傳後 86LDmicro 會自動連線（「工具 > 選項」的「上傳後自動重新連線」，預設開啟）；或按工具列 **線上**（Ctrl+F5，「線上 > 連線到控制器」）。
2. 狀態列顯示「線上 COMx：RUN」與掃描時間，階梯圖顯示控制器上的即時能流與數值。
3. 按下實體的啟動按鈕（X00）：`Rrun` 變 TRUE，2 秒後 Y00 的指示燈亮；按停止按鈕（X01）熄滅。
4. 要結束時按「**線上 > 中斷連線**」。

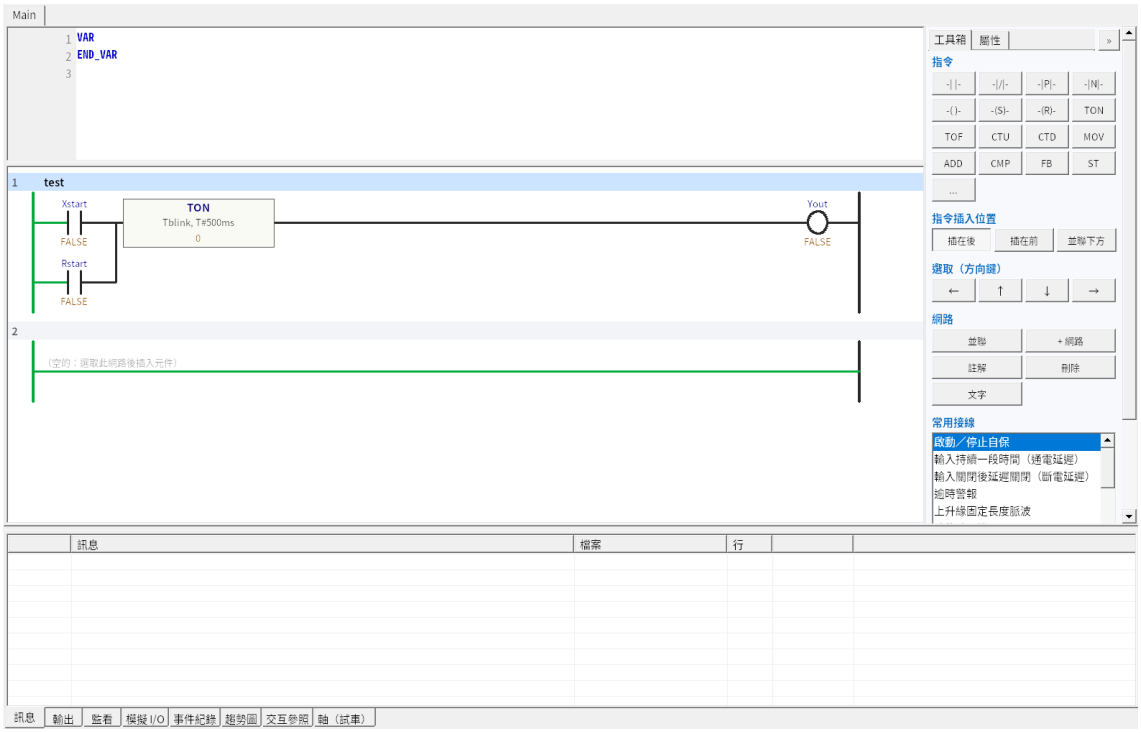


圖 5-27　線上監看：階梯圖顯示控制器的即時狀態

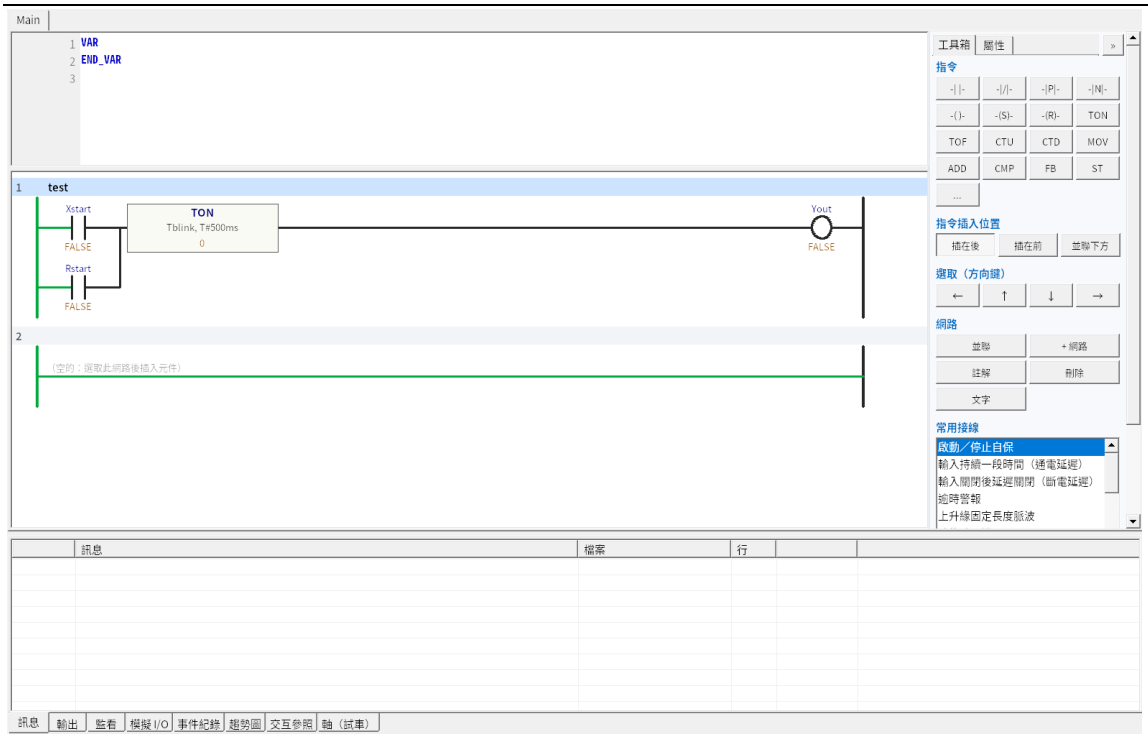


圖 5-28　線上時的狀態列：線上 COMx：RUN 與掃描時間

在「監看」加入變數，雙擊 BOOL 可切換，右鍵可強制（第 9.3 節）。線上監看、RUN / STOP、事件紀錄等的完整說明見第 10.4 節。

> ***提示：**完成！第 6 章以同樣的流程提供更多常用範例（TOF、計數、閃爍、ST、SFC、C/C++）；第 7 章列出全部階梯圖指令。

第 6 章　實作範例

上一章以「啟動 / 停止自保 + 延遲指示燈」完整走過一次流程。本章再提供 7 個常用範例，每個範例都依相同的格式：**情境 → 變數規劃表 → 程式（.lad / ST / SFC / C++ 文字）→ 詳細步驟表 → 預期結果 → 畫面**。

每個範例都是一個獨立的 sketch（名稱見各節），流程與第 5 章相同：建立專案 → 宣告變數 → 編寫程式 → I/O 對應 → 檢查與模擬 → 上傳。程式文字可以在階梯圖的文字模式（Ctrl+T）直接貼上，也可以依步驟表用工具箱逐一插入。

> ***重要**：變數規劃表中的「I/O 對應」為建議值，請依實際配線調整。每個 X… / Y… 變數都要在「I/O 對應」對應到實體通道（第 5 章步驟 9），否則上傳後 I/O 不會動作。

6.0　通用原則（請先閱讀）

下列四個原則貫穿本章所有範例，也是從 LDmicro 1.1 移轉時最需要調整的觀念。理解之後，編譯與上機時幾乎不會卡住。

6.0.1　原則 ①：變數一定要先宣告，型別由宣告決定

LDmicro 1.1 依名稱字首（X、Y、R、T、C）與元件視窗的 Source 自動產生變數；86LDmicro 2.0 依 IEC 61131-3，**每個變數都要先宣告並指定型別**。名稱的字首只是命名習慣，不決定變數的種類。

宣告在哪裡	寫法	誰可以用	適合
全域變數表 GVL（專案樹「全域變數」）	`VAR_GLOBAL ... END_VAR`	所有 POU、I/O 對應、86HMI、C/C++（`GVL.名稱`）	輸入、輸出、跨程式共用的旗標與數值
POU 上方的宣告區	`VAR ... END_VAR`	只有這個 POU（其他地方以 `程式.變數` 讀取）	功能塊實例（`tLamp : TON;`）、1.x 計時器、步驟編號等區域變數
GVL 的保持區	`VAR_GLOBAL RETAIN ... END_VAR`	同 GVL	斷電後要保留的數值（例如累計次數）

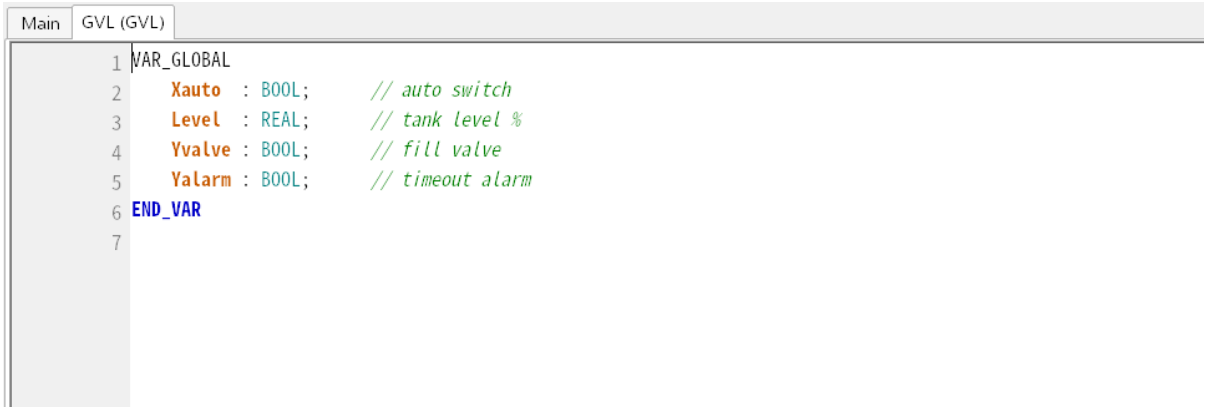


圖 6-1　GVL 編輯器：一行一個變數，型別寫在冒號之後，// 後面是註解

階梯圖用到未宣告的名稱（例如打錯字 `Xstrat`）時，元件以**紅底**顯示；按 **檢查**（Shift+F7）後「訊息」視窗列出「未知的變數 Xstrat：請在 VAR ... END_VAR 中宣告，或宣告在全域變數表（GVL）」，雙擊該列即跳到元件。

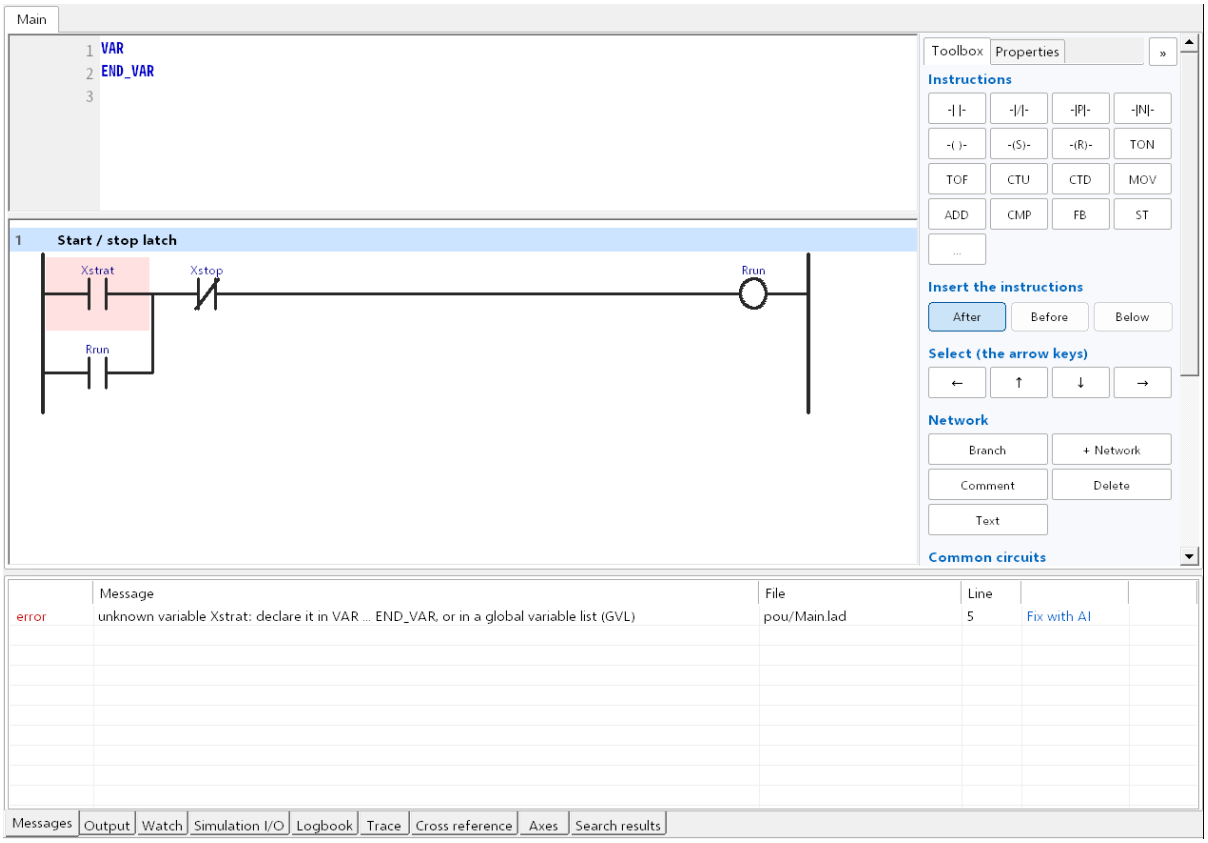


圖 6-2　未宣告的變數：元件紅底，建置輸出顯示「專案有錯誤」

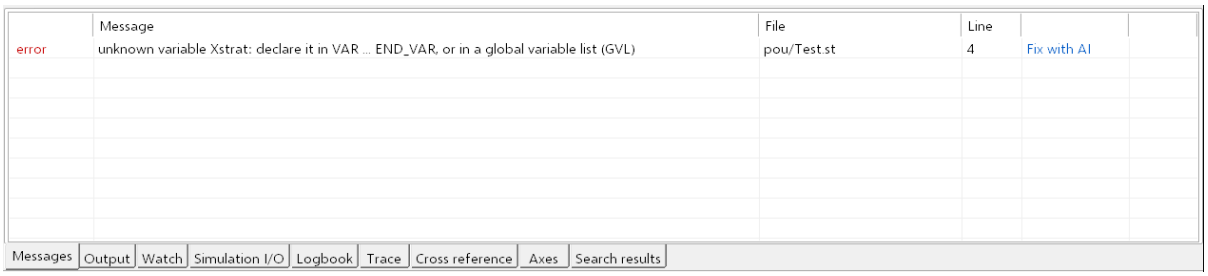


圖 6-3　「訊息」視窗：每列一個錯誤，雙擊跳到該元件

6.0.2 原則 ②：I/O 只透過「I/O 對應」連到實體點

變數不會因為名稱是 `X00` 或 `Xstart` 就變成實體輸入。只有在「I/O 對應」中指定通道（例如 `LocalIO.X00`）的變數，才會在每個掃描開始時讀入實體輸入、在掃描結束時寫出實體輸出。

變數種類	需要 I/O 對應？	說明
實體輸入（按鈕、感測器）	要	對應 `LocalIO.X00`～`X17`（八進位）或 EtherCAT 模組的輸入
實體輸出（指示燈、電磁閥、馬達）	要	對應 `LocalIO.Y00`～`Y17` 或 EtherCAT 模組的輸出
類比量（液位、溫度）	要（類比通道）	在對應視窗的「換算」把原始值換成工程單位（第 4.6 節）
內部旗標、數值（`R…`、計數值）	不需要	只在記憶體中
功能塊實例、1.x 計時器 / 計數器變數	不需要	宣告在 POU 的 VAR 即可

階梯圖的接點讀取了一個 BOOL 全域變數，但它既沒有 I/O 對應、也沒有任何程式或 86HMI 寫入它時，建置會出現警告「… 在這裡只被讀取：沒有 I/O 對應，也沒有任何地方寫入它 —— 它永遠是 FALSE」，提醒你漏了對應。

*__提示：__驗證捷徑：完成程式後先按 **檢查**（Shift+F7）→ **模擬**（F5），在「模擬 I/O」雙擊輸入通道、觀察輸出通道與階梯圖的能流是否符合預期。模擬正常後再上傳，可省下大量「燒錄、測試、再修改」的時間。

6.0.3 原則 ③：一個變數只由一個地方寫入；注意掃描與網路的順序

PLC 以「讀入輸入 → 依序執行程式 → 寫出輸出」為一個掃描，依工作的週期重複（新專案為每 10 ms）。同一個 POU 的網路**由上而下**執行，同一個工作中的程式依工作設定的順序執行（第 4.3 節）。

- 同一個變數不要用兩個 `OUT` 線圈驅動：後面的網路會蓋掉前面的結果。例如網路 1 `NO(Xstart) OUT(Ymotor)`、網路 2 `NO(Xstop) OUT(Ymotor)`，即使 `Xstart` 為 TRUE，`Ymotor` 仍由網路 2 決定（FALSE）。需要多個條件時，把它們並聯或串聯在**同一個**線圈之前。
- `SET` / `RST` 成對使用是例外：寫在後面的網路優先（例如停止優先時，把 `RST` 放在 `SET` 之後）。
- 要找出某個變數被哪些程式寫入，在變數上按右鍵「**<變數> 用在哪裡？**」，或用「交互參照」（Ctrl+Shift+F，第 9.5 節）。
- 階梯圖、ST、C/C++ 可以共用 GVL 變數；同樣要遵守「一個寫入者」，其他地方只讀取（範例 6.7）。

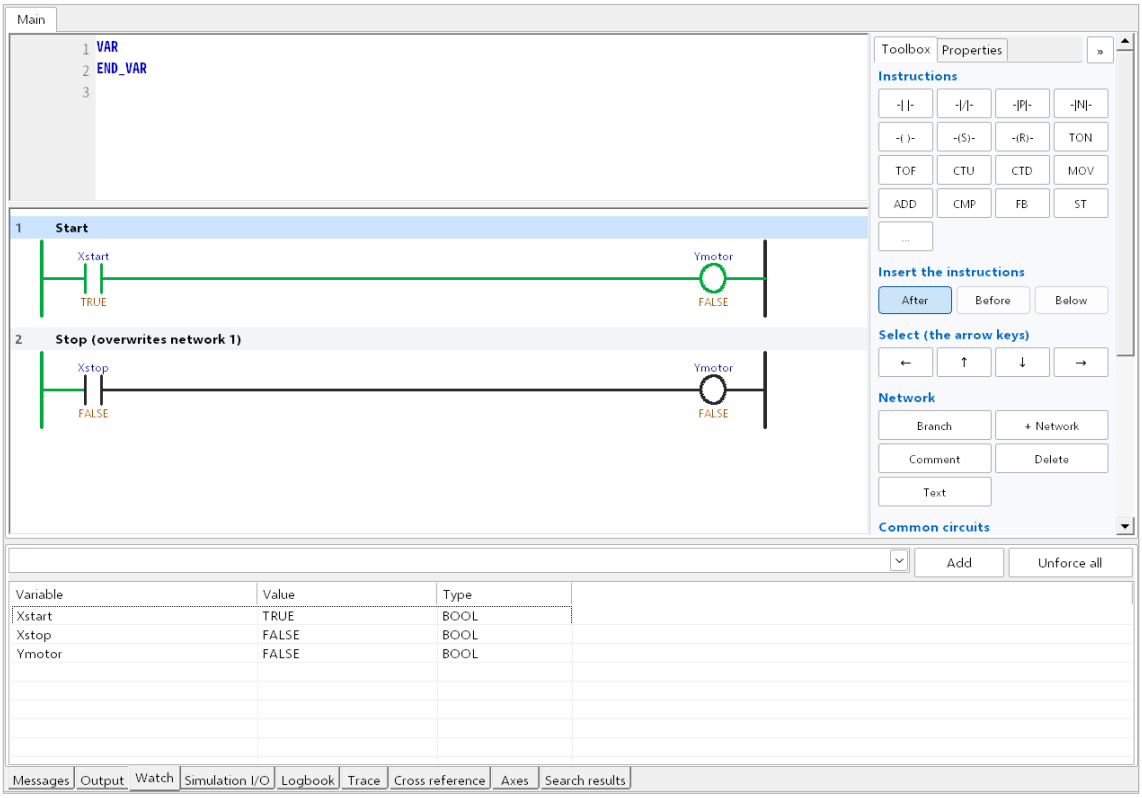


圖 6-4　兩個 OUT 驅動同一個變數：後面的網路蓋掉前面的結果

6.0.4 原則 ④：計時器與計數器：IEC 功能塊或 1.x 方塊

86LDmicro 2.0 有兩種計時器與計數器，可以混用：

	IEC 功能塊（建議）	**LDmicro 1.x 相容方塊**
種類	TON、TOF、TP、CTU、CTD、CTUD、R_TRIG、F_TRIG、SR、RS	TON、TOF、RTO、RTL、TCY、THI、TLO、CTU、CTD、CTC、CTR…
插入方式	**FB**（鍵 K）：`CALL(實例, IN := @, PT := T#2s, Q => @)`	工具箱的 **TON**（O）、**TOF**（F）、**CTU**（U）、**CTD**（D），其餘在「…」
宣告	實例宣告在 VAR：`tLamp : TON;`	計時器 / 計數器變數宣告在 VAR：`t1 : DINT;`、`c1 : INT;`
時間	實際時間，與工作週期無關	以 PLC 週期計數（延遲 ÷ 週期，取整），與 1.1 相同
讀取結果	任何地方都可讀成員：`NO(tLamp.Q)`、`CMP(cBox.CV >= 10)`、ST 的 `tLamp.ET`	讀計數變數的值（例如 `GEQ(c1, 10)`）

LDmicro 1.1 不能把計時器或計數器直接當成接點，必須先驅動一個 `R_…` 內部繼電器再「橋接」；在 2.0 中，IEC 功能塊的輸出（`.Q`、`.CV`、`.ET`）就是變數，可以直接用在任何網路的接點或比較中。

6.1 啟動 / 停止自保

最典型的 PLC 迴路，相當於繼電器的自保持：兩個按鈕控制一個輸出，按下啟動後即使放開按鈕，輸出仍保持 ON，直到按下停止。

情境：以啟動、停止兩個按鈕控制一台馬達（或一盞燈）；按一下啟動就持續運轉。sketch 名稱：`Ex61_Latch`。

變數	型別	I/O 對應	用途
`Xstart`	BOOL	`LocalIO.X00`	啟動按鈕（常開）
`Xstop`	BOOL	`LocalIO.X01`	停止按鈕（常開接線）
`Ymotor`	BOOL	`LocalIO.Y00`	馬達 / 運轉燈

程式（Main，.lad 文字）

```
PROGRAM Main
VAR
END_VAR
NETWORK '啟動 / 停止自保'
    // 按 Xstart 啟動並自保，按 Xstop 停止
    [ NO(Xstart) | NO(Ymotor) ] NC(Xstop) OUT(Ymotor)
END_PROGRAM
```

詳細步驟

#	操作
1	依第 5 章步驟 1～3 建立 sketch `Ex61_Latch` 與新專案。
2	雙擊「全域變數 > GVL」，宣告 `Xstart`、`Xstop`、`Ymotor`（BOOL），Ctrl+S 存檔。
3	開啟 `Main`，點網路 1 選取；按 **C** → 變數 `Xstart` → 確定；按 **N** → `Xstop`；按 **L** → `Ymotor`。
4	按 ← 兩次選取 `Xstart`，按 **B**（在下方並聯常開接點）→ 變數 `Ymotor` → 確定。
5	雙擊網路 1 的標題列，標題輸入 `啟動 / 停止自保`、註解輸入 `按 Xstart 啟動並自保，按 Xstop 停止。`
6	「I/O 對應」新增：`Xstart` → `LocalIO.X00`、`Xstop` → `LocalIO.X01`、`Ymotor` → `LocalIO.Y00`。
7	Shift+F7 檢查；F5 模擬，在「模擬 I/O」雙擊 `LocalIO.X00` 兩次（ON、OFF），再雙擊 `LocalIO.X01` 兩次，確認自保與停止。
8	F8 上傳。

預期結果：按一下 X00 後 Y00 = ON 並保持；按一下 X01 後 Y00 = OFF。兩個按鈕同時按住時 Y00 = OFF（停止優先）。

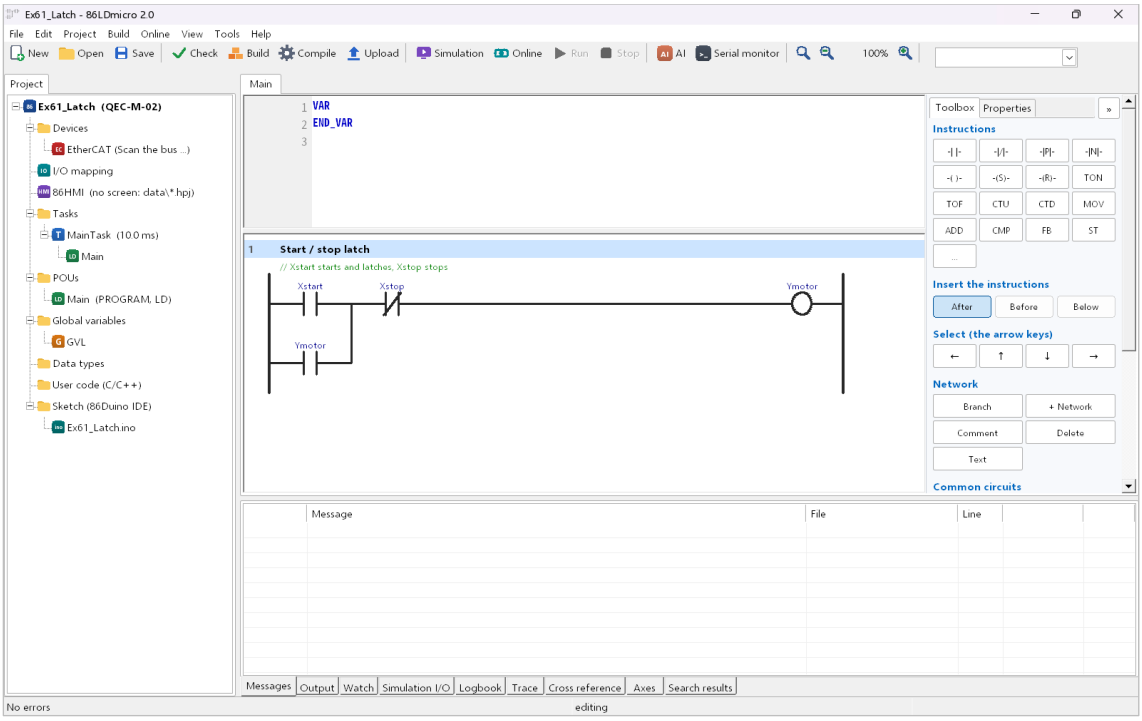


圖 6-5　6.1 的階梯圖：Xstart 與自保接點 Ymotor 並聯，串聯 Xstop，驅動 Ymotor

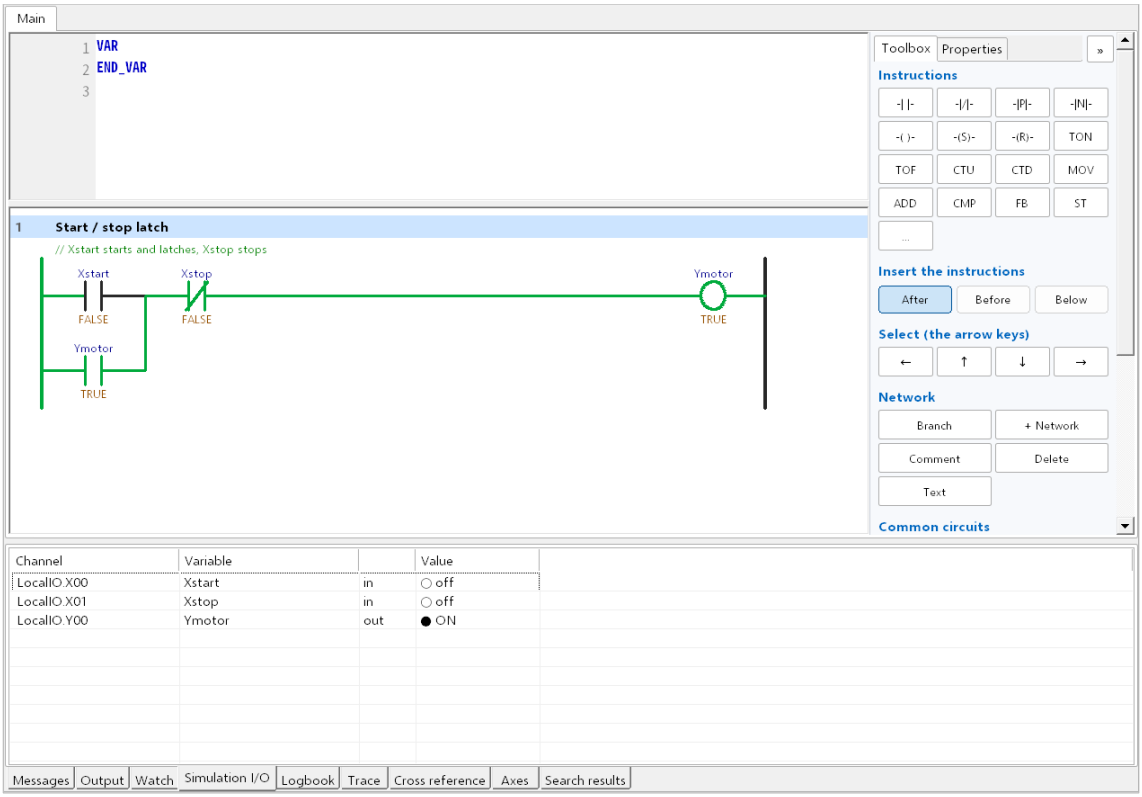


圖 6-6　6.1 模擬：放開啟動按鈕後 Ymotor 仍為 TRUE（自保）

另一種做法：常用接線。 右側欄「常用接線」選 **啟動 / 停止自保**，按 **插入**，填入啟動按鈕、停止按鈕與輸出的名稱，就會插入相同的網路，並自動在 GVL 宣告新變數。

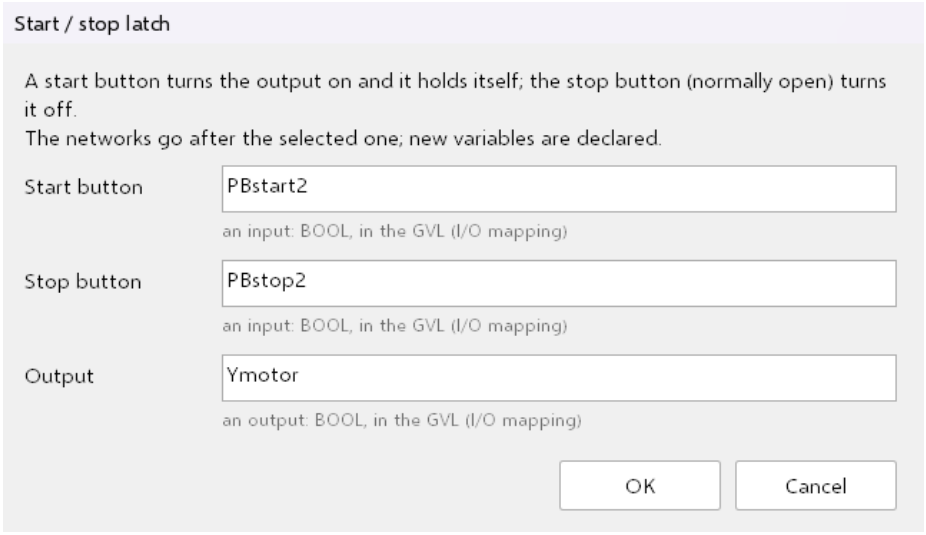


圖 6-7　常用接線「啟動 / 停止自保」：填入按鈕與輸出的名稱

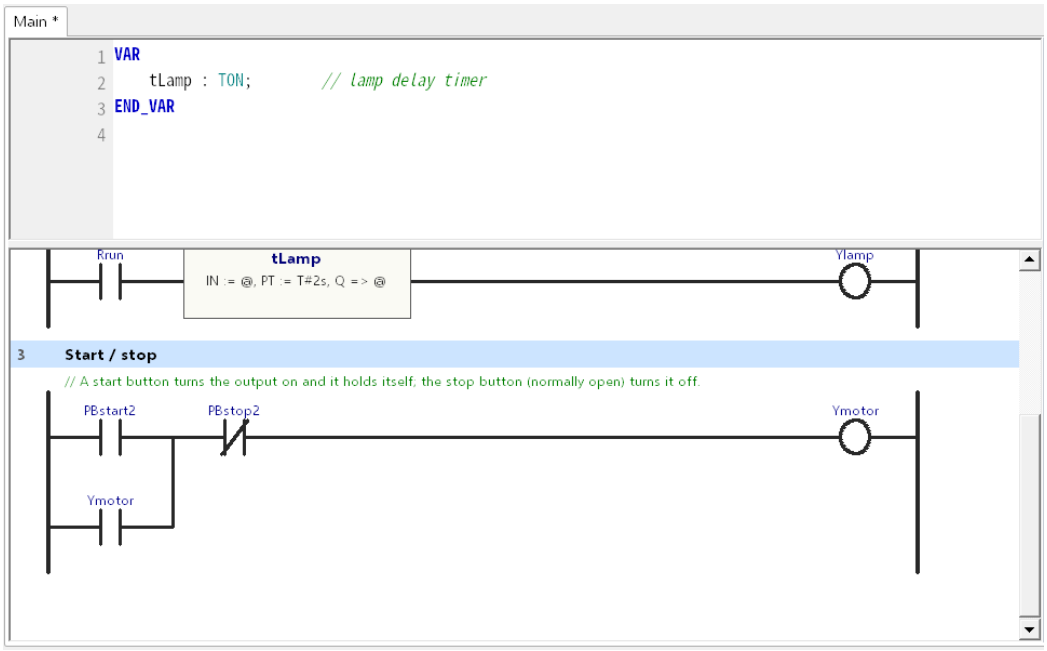


圖 6-8　插入後：範本產生的自保網路，新變數已自動加入 GVL

變化：SET / RST。 同樣的功能也可以用置位、復位線圈寫成兩個網路；RST 寫在後面，所以兩個按鈕同時按下時停止優先：

```
NETWORK '啟動：置位'
    NO(Xstart) SET(Ymotor)
NETWORK '停止：復位（寫在後面，停止優先）'
    NO(Xstop) RST(Ymotor)
```

6.2 延時熄燈（TOF）

開關打開時燈立即亮；開關關閉後，燈再亮 5 秒才熄滅。常用於走廊燈、廁所燈、會議室照明。

情境：開關 ON → 燈立即 ON；開關 OFF → 延遲 5 秒後燈 OFF。sketch 名稱：`Ex62_Tof`。

變數	型別	I/O 對應	用途
`Xsw`	BOOL	`LocalIO.X00`	燈開關
`Ylight`	BOOL	`LocalIO.Y00`	照明燈
`tOff`	TOF（IEC 功能塊，Main 的 VAR）	—	斷電延遲 5 秒

程式（Main，.lad 文字）

```
PROGRAM Main
VAR
    tOff : TOF;          // 斷電延遲 5 秒
END_VAR
NETWORK '延時熄燈'
    // 開關 ON 燈立即亮；開關 OFF 後再亮 5 秒
    NO(Xsw) CALL(tOff, IN := @, PT := T#5s, Q => @) OUT(Ylight)
END_PROGRAM
```

詳細步驟

#	操作
1	建立 sketch `Ex62_Tof` 與新專案；GVL 宣告 `Xsw`、`Ylight`（BOOL）。
2	開啟 `Main`，在上方宣告區的 `VAR` 與 `END_VAR` 之間輸入 `tOff : TOF;`。
3	點網路 1 選取，按 **C** → 變數 `Xsw`。
4	按 **K**（FB）→「元件」輸入 `CALL(tOff, IN := @, PT := T#5s, Q => @)` → 確定。
5	按 **L** → 變數 `Ylight`。
6	「I/O 對應」：`Xsw` → `LocalIO.X00`、`Ylight` → `LocalIO.Y00`（`tOff` 不需要對應）。
7	F5 模擬：雙擊 `LocalIO.X00` 設為 ON，`Ylight` 立即 TRUE；再雙擊設為 OFF，在「監看」加入 `Main.tOff.ET` 觀察經過時間，5 秒後 `Ylight` 變 FALSE。
8	F8 上傳。

預期結果：`Xsw` = 1 時 `Ylight` 立即 = 1；`Xsw` 由 1 變 0 後，`Ylight` 保持 1 共 5 秒再變 0（`tOff.ET` 由 0 計到 `T#5s`）。在 5 秒內再打開開關，計時重來，燈持續亮。

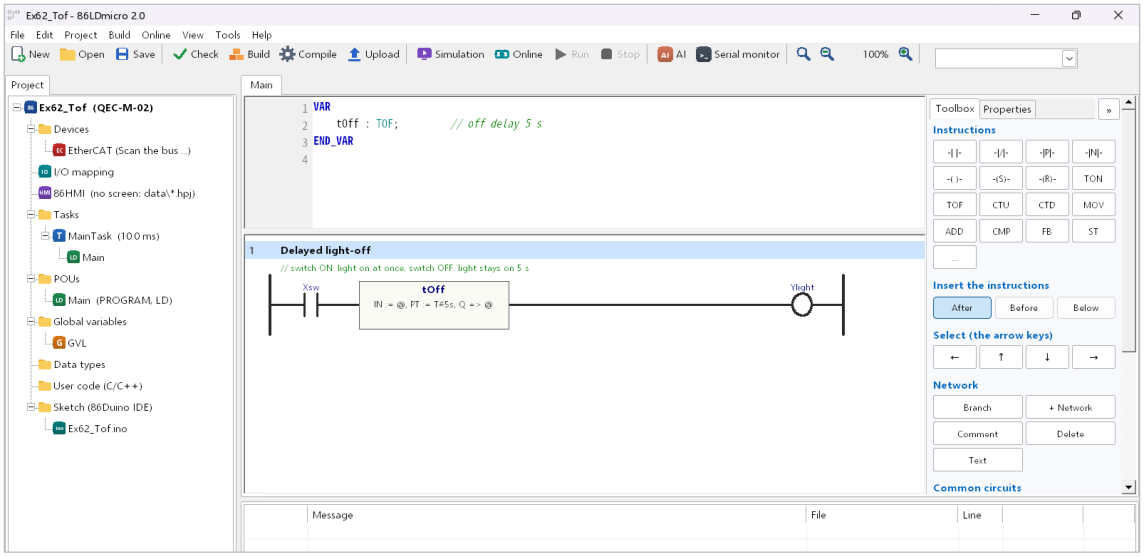


圖 6-9　6.2 的階梯圖：Xsw → TOF 功能塊 tOff (PT := T#5s) → Ylight

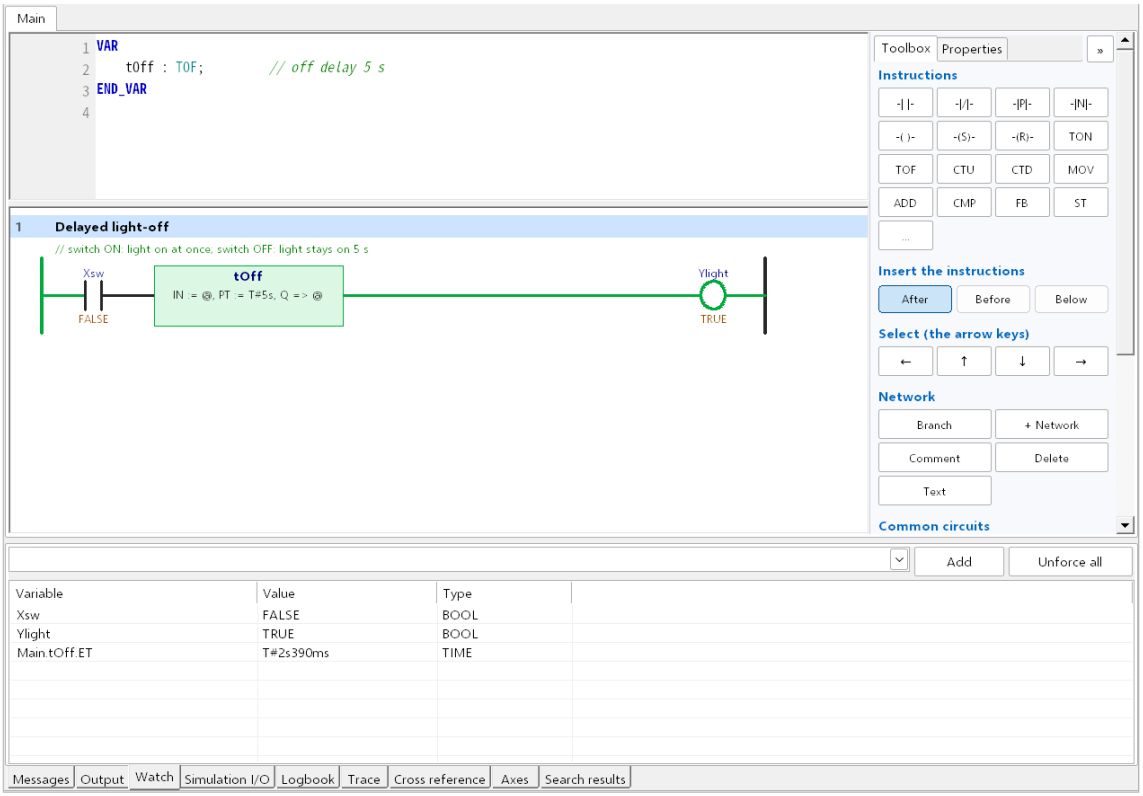


圖 6-10　6.2 模擬：開關已 OFF，TOF 計時中，Ylight 仍為 TRUE

*提示：**變化 1：按一下亮 N 秒**（自動熄燈按鈕）：把 `tOff : TOF` 換成 `tOn : TP`，寫成 `NO(Xsw) CALL(tOn, IN := @, PT := T#5s, Q => @) OUT(Ylight)`。TP 在輸入的上升緣輸出固定 5 秒的脈波，不論按鈕按多久。

*__變化 2：1.x 方塊__：按工具箱的 **TOF**（F），寫成 `NO(Xsw) TOF(t1, T#5s) OUT(Ylight)`，並在 VAR 宣告 `t1 : DINT;`（以 PLC 週期計時，第 7.4 節）。

6.3 包裝計數（CTU + 比較）

感測器每偵測到一個產品就加 1；累計 10 個時，驅動裝箱機構 1 秒，並自動歸零繼續計數。本例示範以比較指令讀取計數值，再以內部旗標 Rdone 帶動後續網路。

情境：生產線計數，每箱 10 個，自動裝箱、自動歸零循環。sketch 名稱：`Ex63_Counter`。

變數	型別	I/O 對應	用途
`Xsensor`	BOOL	`LocalIO.X00`	產品感測器
`Ybox`	BOOL	`LocalIO.Y00`	裝箱機構
`Count`	INT	—	目前計數（可在監看、86HMI 顯示）
`Rdone`	BOOL	—	滿 10 個的旗標
`cBox`	CTU（Main 的 VAR）	—	產品計數器
`tBox`	TP（Main 的 VAR）	—	裝箱 1 秒脈波

程式（Main，.lad 文字）

```
PROGRAM Main
VAR
    cBox : CTU;          // 產品計數
    tBox : TP;           // 裝箱 1 秒脈波
END_VAR
NETWORK '計數產品'
    // 感測器每個上升緣加 1；Rdone 為 TRUE 時歸零
    NO(Xsensor) CALL(cBox, CU := @, R := Rdone, PV := 10, CV => Count)
NETWORK '比較：滿 10 個'
    CMP(Count >= 10) OUT(Rdone)
NETWORK '裝箱 1 秒'
    NO(Rdone) CALL(tBox, IN := @, PT := T#1s, Q => @) OUT(Ybox)
END_PROGRAM
```

詳細步驟

#	操作
1	建立 sketch `Ex63_Counter` 與新專案；GVL 宣告 `Xsensor`、`Ybox`、`Rdone`（BOOL）與 `Count`（INT）。
2	`Main` 宣告區輸入 `cBox : CTU;` 與 `tBox : TP;`。
3	網路 1：按 **C** → `Xsensor`；按 **K** →「元件」輸入 `CALL(cBox, CU := @, R := Rdone, PV := 10, CV => Count)`。
4	Ctrl+Enter 新增網路 2：按 **E**（條件接點 CMP）→「條件」輸入 `Count >= 10`；按 **L** → `Rdone`。
5	Ctrl+Enter 新增網路 3：按 **C** → `Rdone`；按 **K** → `CALL(tBox, IN := @, PT := T#1s, Q => @)`；按 **L** → `Ybox`。

#	操作
6	「I/O 對應」：`Xsensor` → `LocalIO.X00`、`Ybox` → `LocalIO.Y00`（`Count`、`Rdone`、`cBox`、`tBox` 不需要對應）。
7	F5 模擬：在「模擬 I/O」反覆雙擊 `LocalIO.X00`（ON、OFF 為一個產品），共 10 次；在「監看」加入 `Count` 觀察計數。
8	F8 上傳。

動作說明

1. `Xsensor` 每個上升緣，`cBox` 加 1，`Count` 跟著更新。
2. `Count` 到 10 時，網路 2 的條件 `Count >= 10` 成立，`Rdone` = TRUE。
3. 網路 3：`Rdone` 的上升緣觸發 `tBox`（TP），`Ybox` = TRUE 固定 1 秒。
4. 下一個掃描，網路 1 的 `R := Rdone` 把計數歸零（`Count` = 0），網路 2 的 `Rdone` 回到 FALSE；`Ybox` 仍由 TP 維持到 1 秒結束。
5. 循環計數下一箱。

預期結果：每收到 10 個脈波，Ybox 通電 1 秒，計數歸零重新開始。前提：裝箱的 1 秒內不會再湊滿下一個 10 個（TP 在脈波期間不接受新的觸發，下一箱的 Rdone 會被忽略）。

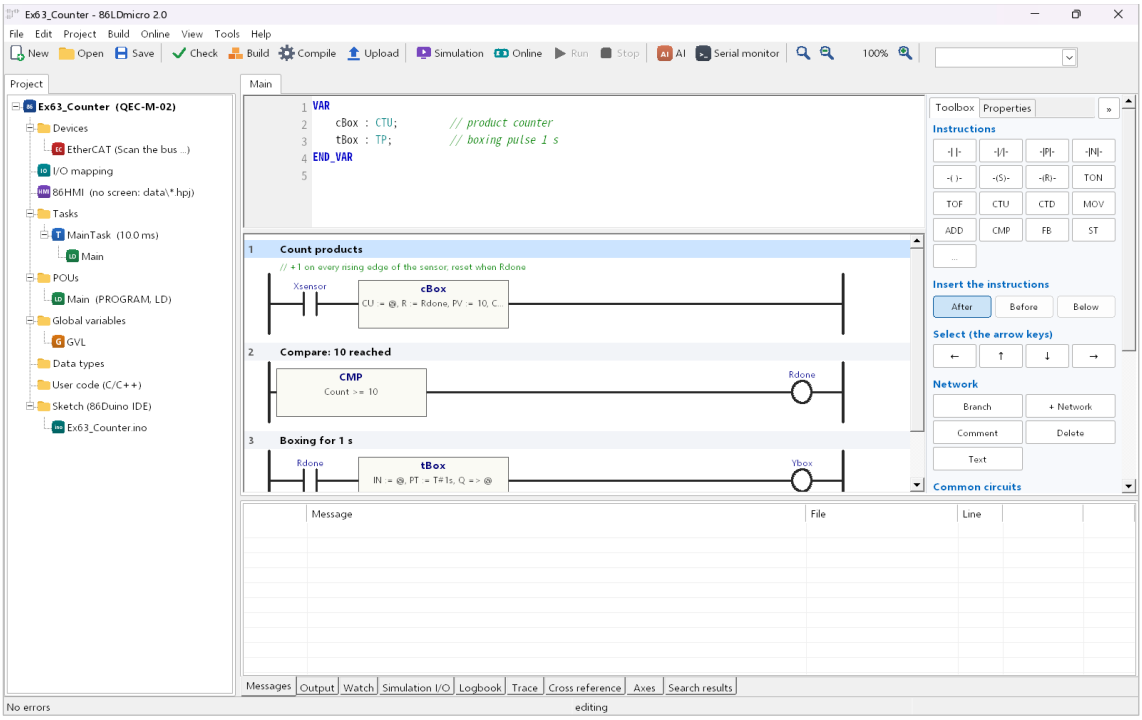


圖 6-11　6.3 的階梯圖：CTU 計數、CMP 比較、TP 裝箱脈波三個網路

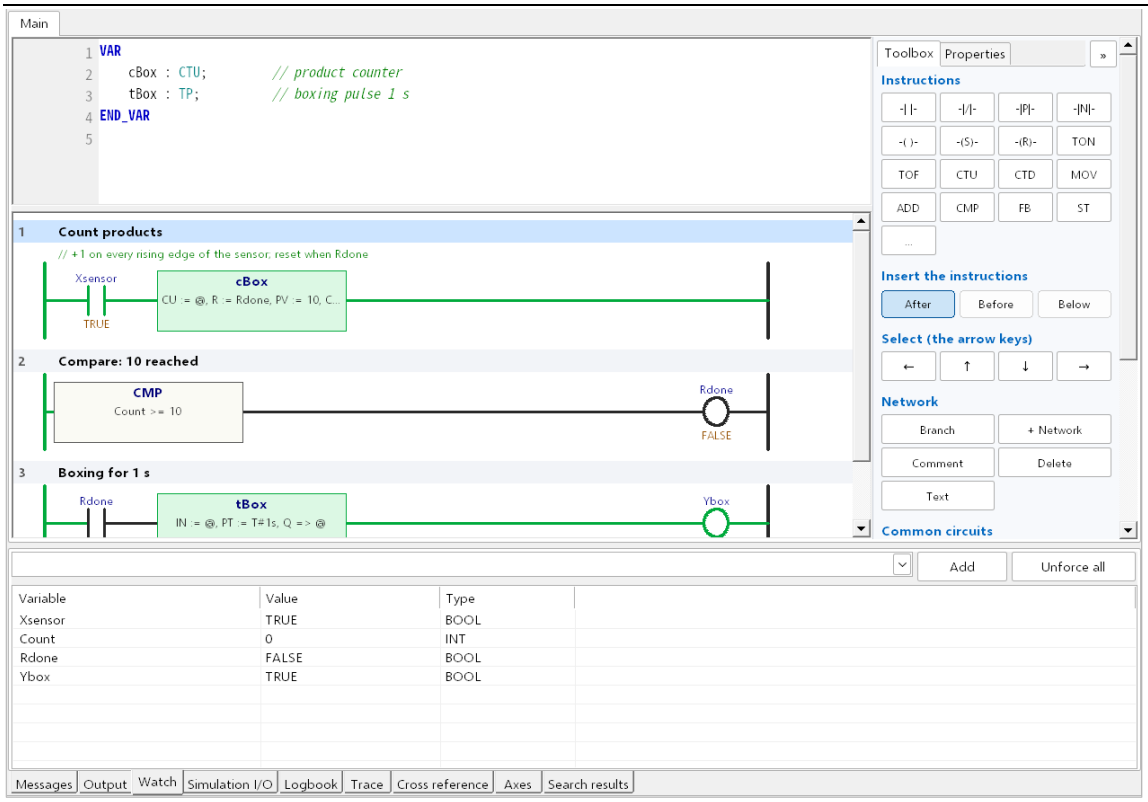


圖 6-12　6.3 模擬：第 10 個產品後 Ybox 通電，Count 已歸零

*提示：比較也可以直接讀功能塊的輸出：`CMP(cBox.CV >= 10)`，或接點 `NO(cBox.Q)`（CV 達到 PV 時為 TRUE）。

感測器訊號有彈跳時，可在 I/O 對應設定「濾波（ms）」，或使用常用接線「輸入去彈跳」。

***注意：產品來得很快時，請加上忙碌互鎖（例如裝箱的 Ybox 期間停止輸送帶），或改成累積待裝箱數量的寫法，避免漏掉一箱。**

計數歸零的那一個掃描若剛好有感測器的上升緣，該產品不會被計入（CTU 的 R 優先）；要求每個產品都計入時，請用上述累積寫法。

6.4 跑馬燈（交替閃爍）

兩盞燈以 500 ms 交替閃爍：一盞亮時另一盞滅，每 500 ms 同步切換一次。本例使用 LDmicro 1.x 相容的循環計時器 **TCY** 產生方波，存在內部位元 `Rstate`，再由兩個網路分別以常開、常閉接點驅動兩盞燈。

情境：開關 ON 時，兩盞燈以反相交替閃爍；開關 OFF 時兩盞都熄滅。sketch 名稱：`Ex64_Blink`。

變數	型別	I/O 對應	用途
`Xrun`	BOOL	`LocalIO.X00`	閃爍開關
`Rstate`	BOOL	—	方波狀態（跨網路共用）
`Yled1`	BOOL	`LocalIO.Y00`	與 `Rstate` 同相
`Yled2`	BOOL	`LocalIO.Y01`	與 `Rstate` 反相
`tClock`	DINT（Main 的 VAR）	—	TCY 的計時器變數 [1.x]

程式（Main，.lad 文字）

```
PROGRAM Main
VAR
    tClock : DINT;      // TCY 計時器 [1.x]
END_VAR
NETWORK '500 ms 方波'
    NO(Xrun) TCY(tClock, T#500ms) OUT(Rstate)
NETWORK '燈 1：與方波同相'
    NO(Xrun) NO(Rstate) OUT(Yled1)
NETWORK '燈 2：與方波反相'
    NO(Xrun) NC(Rstate) OUT(Yled2)
END_PROGRAM
```

詳細步驟

#	操作
1	建立 sketch `Ex64_Blink` 與新專案；GVL 宣告 `Xrun`、`Rstate`、`Yled1`、`Yled2`（BOOL）。
2	`Main` 宣告區輸入 `tClock : DINT;`。
3	網路 1：按 **C** → `Xrun`；按右側欄「指令」的 **...** →「功能方塊」→ **TCY**，「計時器」填 `tClock`、「延遲」填 `T#500ms`（「調整」留空）→ 確定；按 **L** → `Rstate`。
4	Ctrl+Enter 新增網路 2：**C** → `Xrun`；**C** → `Rstate`；**L** → `Yled1`。
5	Ctrl+Enter 新增網路 3：**C** → `Xrun`；**N** → `Rstate`；**L** → `Yled2`。
6	「I/O 對應」：`Xrun` → `LocalIO.X00`、`Yled1` → `LocalIO.Y00`、`Yled2` → `LocalIO.Y01`。
7	F5 模擬：雙擊 `LocalIO.X00` 設為 ON，觀察 `LocalIO.Y00`、`Y01` 每 500 ms 交替 ON / OFF。
8	F8 上傳。

預期結果：`Xrun` = 1 時，`Yled1`、`Yled2` 以相反的狀態約每 500 ms 切換一次（週期約 1 秒）；`Xrun` = 0 時兩者都是 0。

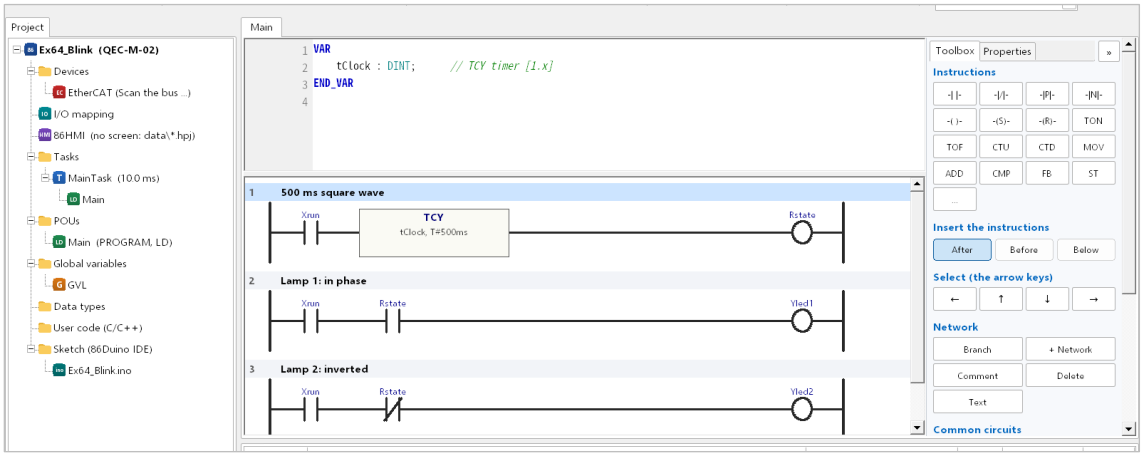


圖 6-13　6.4 的階梯圖：TCY 產生方波 Rstate，兩個網路以 NO / NC 驅動兩盞燈

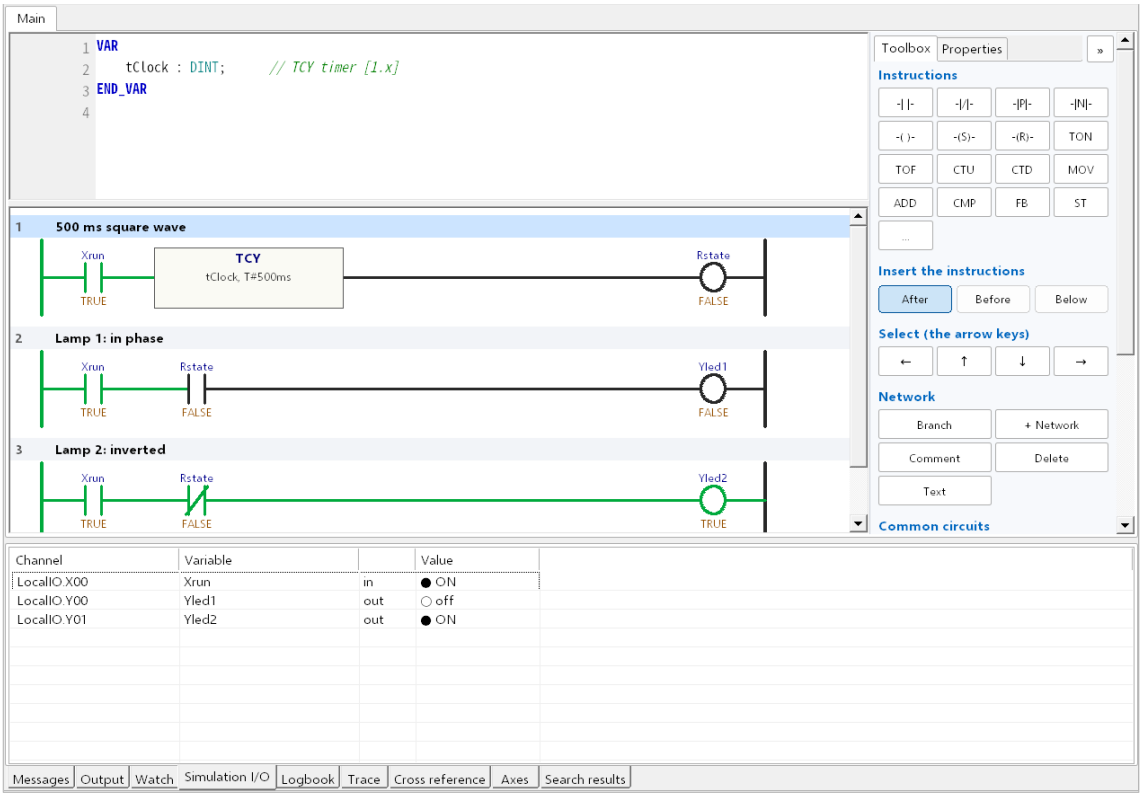


圖 6-14　6.4 模擬：Yled1 與 Yled2 交替亮滅

*提示：TCY 等 1.x 計時器以 PLC 週期計數：工作週期 10 ms 時，`T#500ms` 為 50 個週期。延遲短於工作週期時無法計時（檢查會出現警告）。

也可以不用 1.x 指令：右側欄「常用接線」的 **致能時閃爍** 可分別設定亮、滅時間；或在 ST 用 TON 自行組成閃爍功能塊（第 8.3 節）。

6.5 ST：注水控制

以結構化文字（ST）寫一個有步驟的注水控制：自動開關打開且液位低於 20 % 時開閥注水，到 80 % 關閥；注水超過 30 秒仍未到達，視為異常並輸出警報。ST 適合條件多、有計算或步驟的邏輯。

情境：水槽自動補水，附逾時警報；關閉自動開關即停止並復歸警報。sketch 名稱：`Ex65_Fill`。

變數	型別	I/O 對應	用途
`Xauto`	BOOL	`LocalIO.X00`	自動注水開關
`Level`	REAL	類比輸入（依硬體，第 4.6 節）	液位（%）
`Yvalve`	BOOL	`LocalIO.Y00`	注水閥
`Yalarm`	BOOL	`LocalIO.Y01`	注水逾時警報
`tFill`	TON（Filling 的 VAR）	—	注水逾時 30 秒
`step`	INT（Filling 的 VAR）	—	0 待機、1 注水、2 逾時

程式（Filling，ST）

```
PROGRAM Filling
VAR
    tFill : TON;        // 注水逾時
    step  : INT;        // 0 待機、1 注水、2 逾時
END_VAR
CASE step OF
    0:  (* 待機：自動且液位低於 20 % 就開始注水 *)
        IF Xauto AND Level < 20.0 THEN
            step := 1;
        END_IF;
    1:  (* 注水：到 80 % 停止；超過 30 秒視為異常 *)
        IF Level >= 80.0 OR NOT Xauto THEN
            step := 0;
        ELSIF tFill.Q THEN
            step := 2;
        END_IF;
    2:  (* 逾時：關閉自動開關即復歸 *)
        IF NOT Xauto THEN
            step := 0;
        END_IF;
END_CASE;
tFill(IN := (step = 1), PT := T#30s);   // 每個掃描都呼叫
Yvalve := (step = 1);
Yalarm := (step = 2);
END_PROGRAM
```

詳細步驟

#	操作
1	建立 sketch `Ex65_Fill` 與新專案；GVL 宣告 `Xauto`、`Yvalve`、`Yalarm`（BOOL）與 `Level`（REAL）。
2	「專案 > 新增 PROGRAM（ST）...」，名稱輸入 `Filling` → 確定。新的 PROGRAM 自動加入第一個工作 MainTask，並開啟 ST 編輯器。
3	輸入上面的程式（輸入變數名稱的前幾個字會列出補全）；按 **Ctrl+T** 自動排版，Ctrl+S 存檔。空白的 `Main` 可以保留。
4	「I/O 對應」：`Xauto` → `LocalIO.X00`、`Yvalve` → `LocalIO.Y00`、`Yalarm` → `LocalIO.Y01`。`Level` 先不對應，以便在模擬中直接寫入；上機時再對應到實際的類比輸入通道並設定「換算」（例如原始值 0～4095 換成 0～100）。
5	Shift+F7 檢查；F5 模擬：「模擬 I/O」雙擊 `LocalIO.X00` 設為 ON；「監看」加入 `Level`、`Filling.step`、`Yvalve`，雙擊 `Level` 輸入 `10` → `Yvalve` = TRUE；輸入 `85` → `Yvalve` = FALSE。
6	再把 `Level` 改為 `10` 並等待 30 秒：`Filling.step` = 2、`Yalarm` = TRUE；把 `LocalIO.X00` 設為 OFF，警報復歸。
7	F8 上傳。

預期結果：自動開關 ON 且液位 < 20 % 時注水閥打開；液位到 80 % 關閉。注水 30 秒仍未到 80 % 時關閥並輸出警報，直到自動開關 OFF。

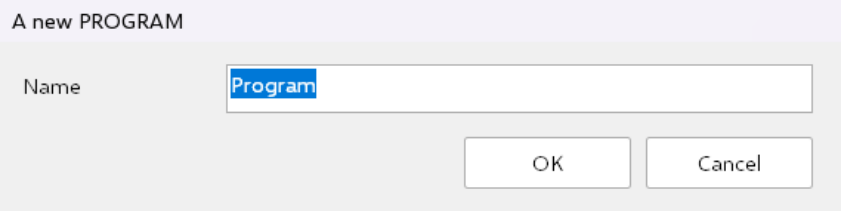


圖 6-15　「專案 > 新增 PROGRAM（ST）...」：輸入程式名稱

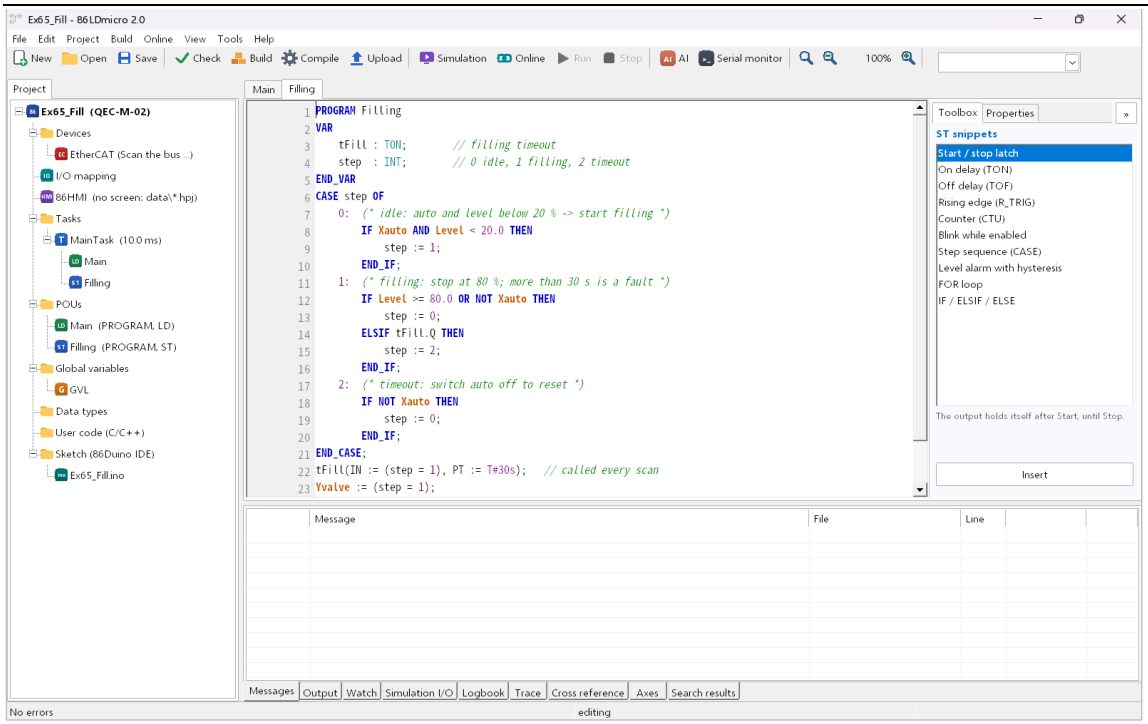


圖 6-16　6.5 的 ST 程式 Filling：CASE 步驟與 TON 逾時

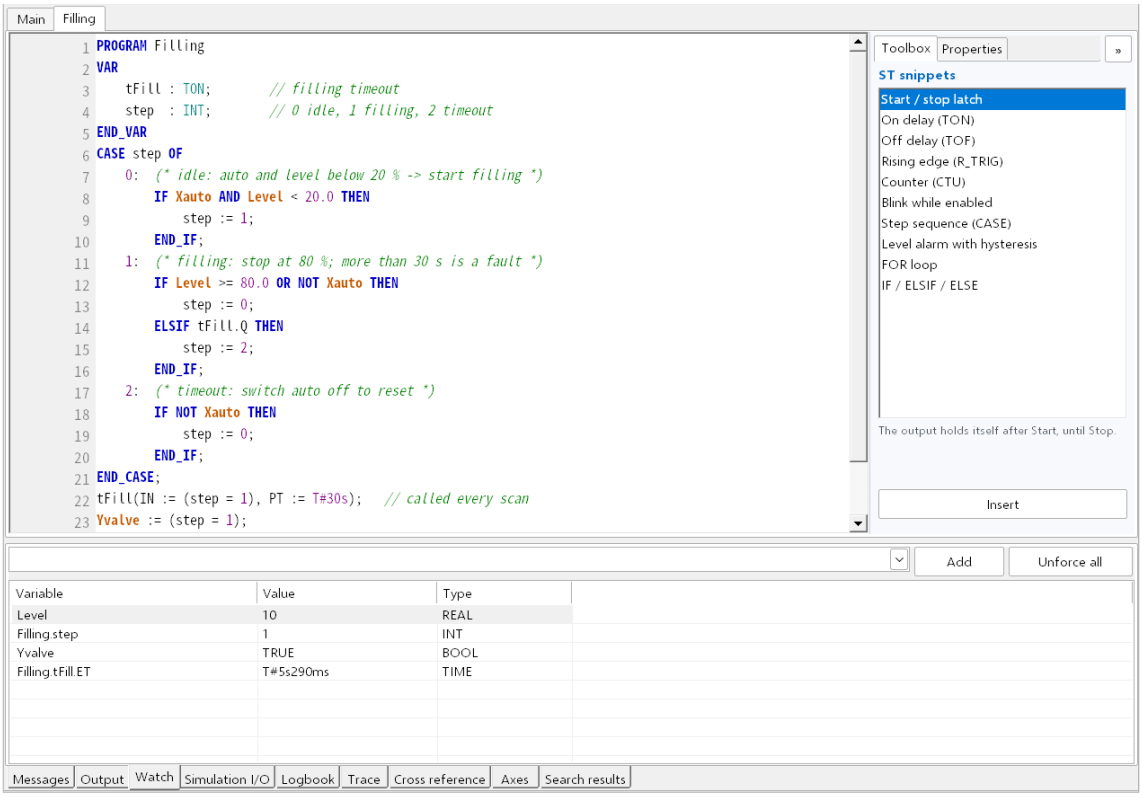


圖 6-17　6.5 模擬：Level = 10，step = 1，注水閥開啟

***提示：**功能塊實例要在每個掃描都呼叫，計時器才會累計時間。本例把 `tFill(...)` 放在 CASE 之外、以 `IN := (step = 1)` 呼叫；離開步驟 1 時 IN 變 FALSE，計時自動歸零。ST 語法見第 8.2 節。

6.6 SFC：順序控制

以順序功能圖（SFC）描述一個「進水 → 攪拌 → 排水」的固定流程。SFC 以**步驟**與**轉移條件**表示流程，一次只有啟動中的步驟執行動作，適合機台的自動流程。

情境：按啟動後：進水閥打開，直到滿水位或 10 秒；攪拌 5 秒；排水 3 秒；完成後回到待機，等待下一次啟動。sketch 名稱：`Ex66_Seq`。

變數	型別	I/O 對應	用途
`Xstart`	BOOL	`LocalIO.X00`	啟動按鈕
`Xfull`	BOOL	`LocalIO.X01`	滿水位開關
`Yvalve`	BOOL	`LocalIO.Y00`	進水閥
`Ymixer`	BOOL	`LocalIO.Y01`	攪拌馬達
`Ydrain`	BOOL	`LocalIO.Y02`	排水閥
`cycles`	INT（Seq 的 VAR）	—	完成次數

程式（Seq，.sfc 文字）

```
PROGRAM Seq
VAR
    cycles : INT;       // 完成次數
END_VAR
SFC
STEP Idle INITIAL
    Ydrain := FALSE;
END_STEP
TRANSITION Xstart
STEP Fill
    Yvalve := TRUE;
END_STEP
TRANSITION Xfull OR Fill.T >= T#10s
STEP Mix
    Yvalve := FALSE;
    Ymixer := TRUE;
END_STEP
TRANSITION Mix.T >= T#5s
STEP Drain
    Ymixer := FALSE;
    Ydrain := TRUE;
END_STEP
TRANSITION Drain.T >= T#3s
STEP Done
    cycles := cycles + 1;
END_STEP
TRANSITION TRUE
JUMP Idle
END_SFC
END_PROGRAM
```

詳細步驟

#	操作
1	建立 sketch `Ex66_Seq` 與新專案；GVL 宣告 `Xstart`、`Xfull`、`Yvalve`、`Ymixer`、`Ydrain`（BOOL）。
2	「專案 > 新增 PROGRAM（SFC，順序流程圖）…」，名稱 `Seq` → 確定。新圖表已有 `Init` → `TRUE` → `Step1` → `Step1.T >= T#1s` → 跳到 `Init`。在上方宣告區輸入 `cycles : INT;`。
3	雙擊 `Init`：名稱改 `Idle`，「動作」輸入 `Ydrain := FALSE;` → 確定（跳到它的 JUMP 會一起改名）。
4	雙擊 `Idle` 之後的轉移條件：「條件（ST）」輸入 `Xstart`。
5	雙擊 `Step1`：名稱 `Fill`，動作 `Yvalve := TRUE;`。雙擊其後的轉移條件，改為 `Xfull OR Fill.T >= T#10s`。
6	選取這個轉移條件，按右側欄「圖表」的 **+ 步驟**（Ins）：在它之後加入一個步驟與 `TRUE` 轉移。雙擊新步驟：名稱 `Mix`，動作 `Yvalve := FALSE; Ymixer := TRUE;`；轉移條件改為 `Mix.T >= T#5s`。
7	同樣選取最後的轉移條件按 **+ 步驟**：步驟 `Drain`（動作 `Ymixer := FALSE; Ydrain := TRUE;`），轉移 `Drain.T >= T#3s`。
8	再按一次 **+ 步驟**：步驟 `Done`（動作 `cycles := cycles + 1;`），轉移保持 `TRUE`，其後就是原有的「跳到 Idle」。
9	「I/O 對應」：依變數規劃表對應 X00、X01、Y00 ~ Y02。
10	F5 模擬：雙擊 `LocalIO.X00` 兩次（ON、OFF），觀察啟動中的步驟（綠色）依序移動與右側顯示的已啟動時間；也可在 Fill 時雙擊 `LocalIO.X01` 模擬滿水位。
11	F8 上傳。

預期結果：按一下 X00：`Yvalve` ON（最多 10 秒，X01 ON 時提早結束）→ `Ymixer` ON 5 秒 → `Ydrain` ON 3 秒 → 回到 `Idle`、`Ydrain` OFF、`cycles` 加 1。

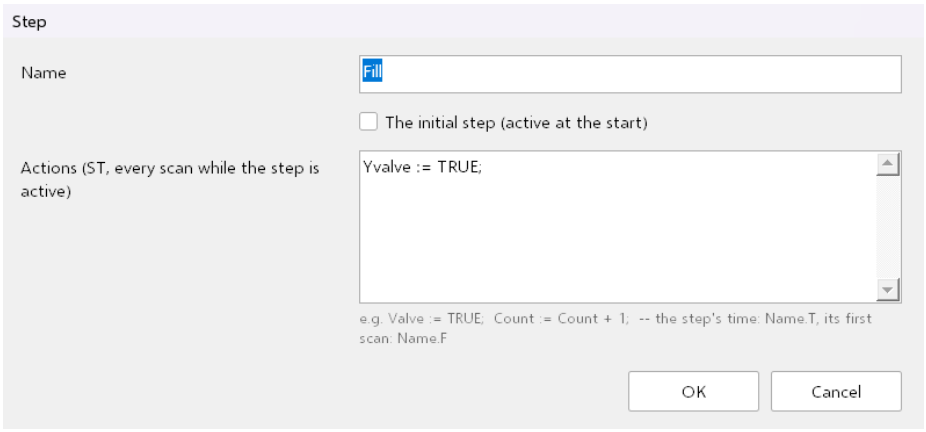


圖 6-18　步驟的設定視窗：名稱、初始步驟、動作（ST）



圖 6-19　轉移條件的設定視窗：條件（ST）與註解

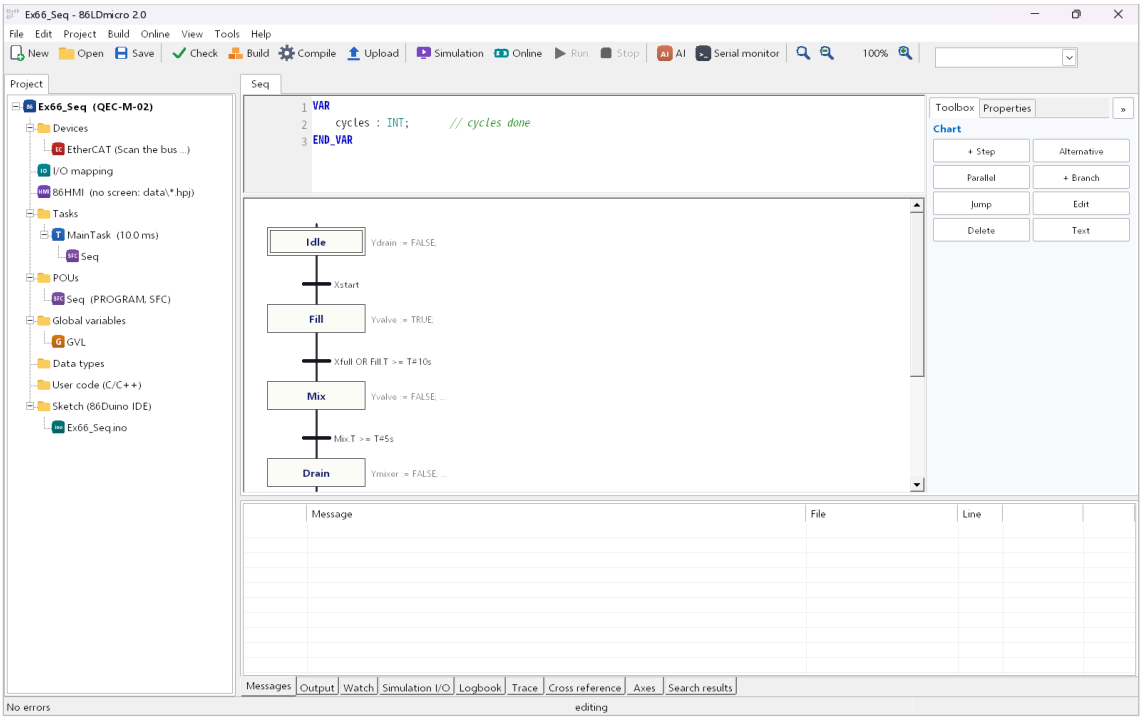


圖 6-20　6.6 的 SFC 圖表：Idle → Fill → Mix → Drain → Done → 跳到 Idle

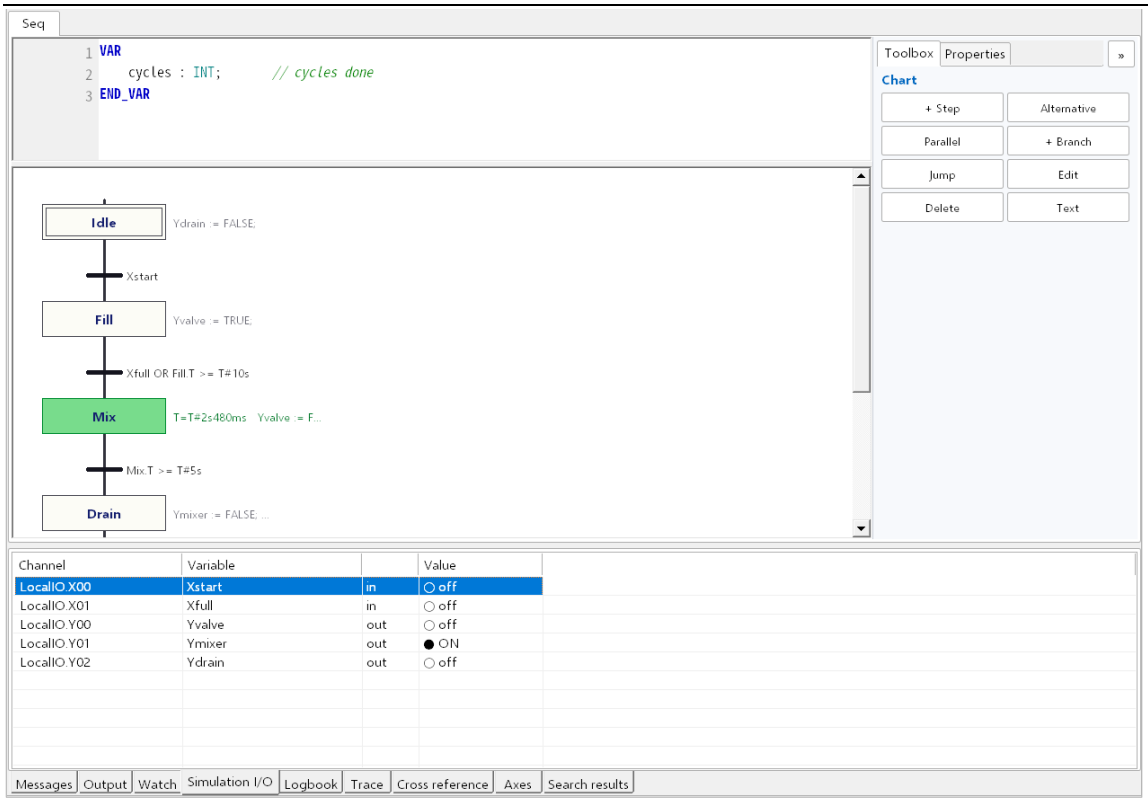


圖 6-21　6.6 模擬：Mix 步驟啟動中（綠色），右側顯示已啟動時間

*提示：SFC 步驟中設定的輸出會**保持**，需要關閉時在後續步驟寫 `:= FALSE`（本例在 Mix 關閉進水閥、在 Drain 關閉攪拌、在 Idle 關閉排水）。

也可以按「圖表」的 **文字**（Ctrl+T）直接輸入上面的 .sfc 文字，再按 **回到圖形**。分支、跳躍與執行語意見第 8.4 ~ 8.6 節。

6.7 C/C++：Serial1 傳送訊息

按下按鈕時，由 C/C++ 使用者程式從 Serial1（控制器的 COM1）送出一行文字，例如給條碼印表機、上位電腦或另一台設備。階梯圖負責判斷按鈕，C/C++ 負責通訊，兩者共用 GVL 變數。

情境：每按一次按鈕，Serial1 以 9600 bps 送出 `Button pressed, count = N`。sketch 名稱：`Ex67_Serial`。

變數	型別	I/O 對應	用途
Xbtn	BOOL	LocalIO.X00	傳送按鈕
Nmsg	DINT	—	按鈕次數（訊息編號），只由階梯圖寫入

程式 1（Main，.lad 文字）：按鈕的上升緣把 Nmsg 加 1。

```
PROGRAM Main
VAR
END_VAR
NETWORK '按一下：訊息編號加 1'
    PT(Xbtn) ADD(Nmsg, Nmsg, 1)
END_PROGRAM
```

程式 2（使用者程式 SendMsg.cpp86）：Nmsg 改變時送出一行訊息。

```
// SendMsg：Nmsg 改變時，從 Serial1 送出一行訊息

void plc86_user_setup(void) {
    Serial1.begin(9600);               // COM1，9600 bps
}

// loop() 每一圈：只做不會卡住的工作（不要 delay()）
void plc86_user_loop(void) {
    static int32_t sent = 0;           // 已送出的編號
    int32_t n = GVL(Nmsg);
    if(n != sent) {
        sent = n;
        Serial1.print("Button pressed, count = ");
        Serial1.println(n);
    }
}
```

詳細步驟

#	操作
1	建立 sketch Ex67_Serial 與新專案；GVL 宣告 `Xbtn : BOOL;` 與 `Nmsg : DINT;`。
2	開啟 Main，網路 1：按 **P**（正緣接點 PT）→ Xbtn；按 **A**（ADD）→「目的」Nmsg、「a」Nmsg、「b」1 → 確定。

#	操作
3	「專案 > 新增使用者程式檔（C/C++）…」，名稱 `SendMsg` → 確定。範本已列出各個掛勾函式：把 `plc86_user_setup()` 與 `plc86_user_loop()` 改成上面的內容，刪除不用的 `plc86_user_before()`、`plc86_user_after()`，Ctrl+S 存檔。
4	「I/O 對應」：`Xbtn` → `LocalIO.X00`。
5	Shift+F7 檢查；F5 模擬：在「模擬 I/O」反覆雙擊 `LocalIO.X00`，「監看」中的 `Nmsg` 每按一下加 1。
6	把 COM1 接到電腦（例如 USB 轉 RS-232 / 485 轉接器），F8 上傳。
7	下方「輸出」切到 **序列埠監看**，選轉接器的序列埠、鮑率 `9600`，按 **連線**；每按一下 X00 就收到一行訊息。

預期結果：每按一下 X00，`Nmsg` 加 1，`Serial1` 送出一行 `Button pressed, count = 1`、`= 2`…。

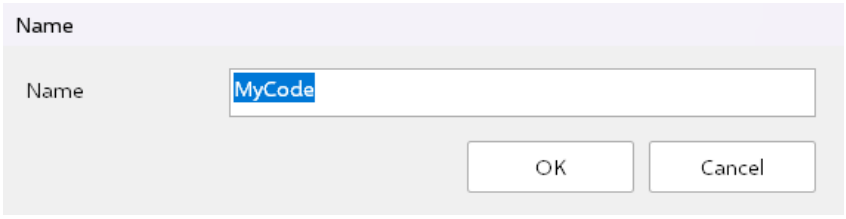


圖 6-22　「專案 > 新增使用者程式檔（C/C++）…」：輸入檔名

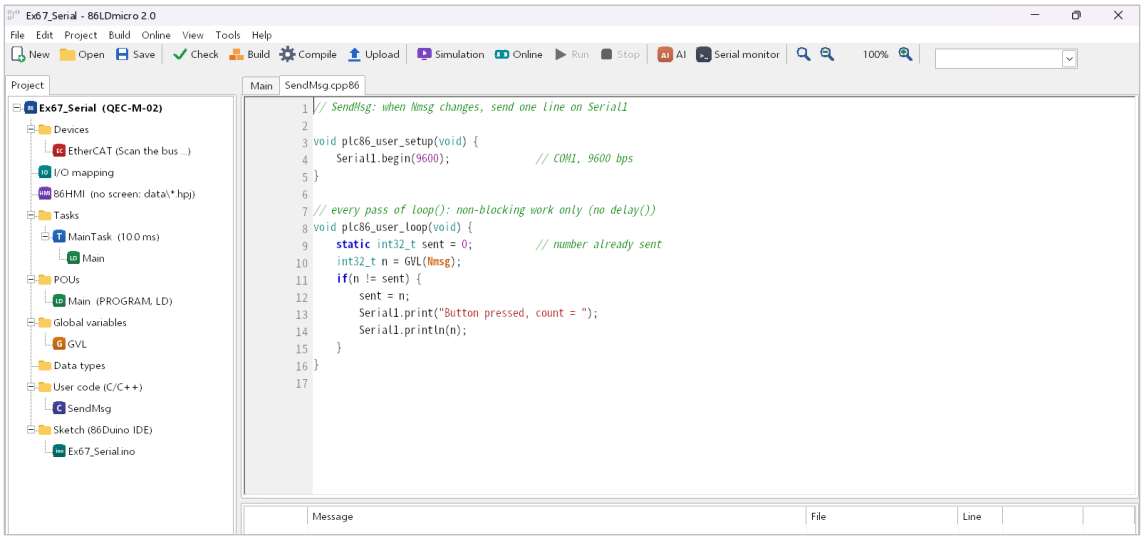


圖 6-23　6.7 的使用者程式 SendMsg.cpp86：plc86_user_setup() 與 plc86_user_loop()

*__注意：__USB 的 `Serial` 用於上傳與線上監看；與外部設備通訊請用 `Serial1`～`Serial3`。

C/C++ 中 PLC 變數要寫成 `GVL(Nmsg)`，不能直接寫名稱。`plc86_user_loop()` 在每次 `loop()` 執行，不要使用 `delay()`。

本例由階梯圖寫入 `Nmsg`、C/C++ 只讀取，符合 6.0 原則 ③。掛勾函式與其他寫法見第 14 章。

第 7 章　階梯圖指令參考

86LDmicro 2.0 的階梯圖共有 90 個指令（與命令列 `86ldmicro-cli instructions` 列出的相同），本章依用途分成 10 組逐一說明。每一節包含簡介、選單或參數視窗的畫面，以及指令表：**指令**（.lad 中的名稱）、**.lad 寫法**（範例，可直接貼到文字模式）、**按鍵**（選取元件或網路後直接按的鍵）、**說明**。

- 標示 **[1.x]** 的是 LDmicro 1.x 相容指令：行為與 1.1 相同，計時器以 PLC 週期計數。新程式建議使用 IEC 功能塊（7.8 節）。
- 參數可以是變數、常數或 ST 運算式；時間請寫 `T#500ms`、`T#2s` 等 IEC 時間常數。
- 工具箱沒有按鈕的指令，請按右側欄「指令」的 **...**，在「接點」、「線圈」、「功能方塊」、「程式流程」、「功能塊」子選單中選取（選單會列出每個指令的說明）。
- 插入任何指令後都會跳出參數視窗，每個參數一欄；「變數」類的欄位可以從下拉選擇已宣告的變數。

7.1 編輯操作

階梯圖分頁上方為宣告區（`VAR ... END_VAR`），中間為網路，右側欄為工具箱。本節說明插入、選取、移動、刪除元件與網路的方法，以及文字模式與常用接線。

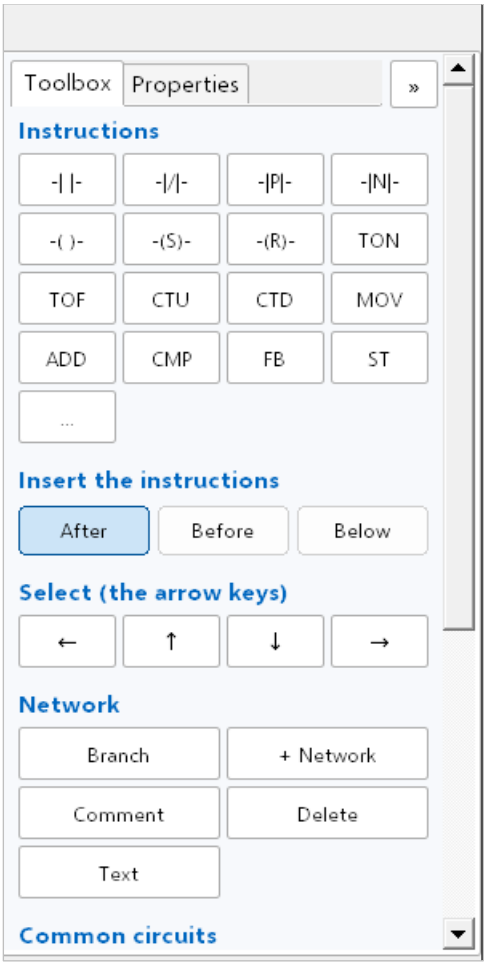


圖 7-1　階梯圖右側欄的工具箱：指令、指令插入位置、選取（方向鍵）、網路、常用接線

工具箱區塊	內容
指令	常用元件按鈕；滑鼠停留顯示指令說明與按鍵；**...** 依類別列出全部指令
指令插入位置	**插在後**：插在選取元件之後（串聯；選取整個網路時放在線圈之前）；**插在前**：插在選取元件之前；**並聯下方**：在選取元件下方加一個並聯分支
選取（方向鍵）	← →：左右的元件；↑ ↓：上下的元件，或前一個 / 下一個網路
網路	**並聯**（在選取元件下方加並聯分支，同鍵 B）、**+ 網路**、**註解**（網路標題與註解）、**刪除**、**文字**（以 .lad 文字編輯整個 POU，Ctrl+T）
常用接線	現成的網路範本：選一項按 **插入**（或雙擊），填入名稱後插入

7.1.1 插入元件

先選取一個元件（插入位置以它為準）或一個網路（點網路的標題列），再用下列任一方式插入：

- **按鈕**：按工具箱「指令」的按鈕。按住 **Shift** 再按按鈕，插在選取元件之前。
- **按鍵**：直接按下表的鍵；**Shift + 鍵** 插在選取元件之前；**B** 在選取元件下方並聯一個常開接點。
- **「...」選單**：按 **...**，依類別選擇任何指令。

鍵	指令	鍵	指令	鍵	指令
C	常開接點 NO	S	置位線圈 SET	U	上數計數器 CTU [1.x]
N	常閉接點 NC	R	復位線圈 RST	D	下數計數器 CTD [1.x]
P	正緣接點 PT	O	通電延遲 TON [1.x]	M	搬移 MOV
L	線圈 OUT	F	斷電延遲 TOF [1.x]	A	加法 ADD
E	條件接點 CMP	K	功能塊 CALL（FB）	T	ST 方塊
B	下方並聯常開接點	Shift + 鍵	插在選取元件之前		

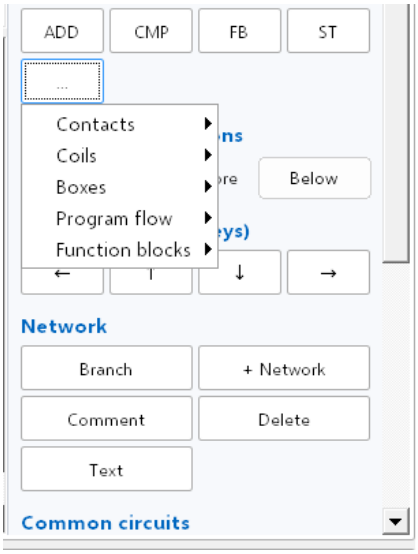


圖 7-2 「...」依類別列出全部指令：接點、線圈、功能方塊、程式流程、功能塊

插入的位置由「指令插入位置」決定。例如在 `Xstart` 選取時插入常閉接點：

指令插入位置	結果（.lad）
插在後	`NO(Xstart) NC(Xstop)`
插在前（或 Shift + N）	`NC(Xstop) NO(Xstart)`
並聯下方	`[ NO(Xstart) | NC(Xstop) ]`

線圈類（OUT、SET、RST...）一律放在網路的右側；網路尾端連續的多個線圈都接在右母線旁，由同一個能流驅動。

7.1.2 選取、修改、移動與刪除

操作	方法
選取元件	點一下元件；或用 ← → ↑ ↓
選取網路	點網路的標題列（編號那一列）；Home / End 選第一個 / 最後一個網路
修改元件的參數	雙擊元件，或選取後按 Enter
移動元件	Ctrl+← / Ctrl+→：與同一串聯中左右的元件交換位置
移動網路	選取網路後 Ctrl+↑ / Ctrl+↓
刪除	Del（或 Backspace），或「網路」的 **刪除**
剪下、複製、貼上	Ctrl+X / Ctrl+C / Ctrl+V（元件或整個網路）
復原、重做	Ctrl+Z / Ctrl+Y

在元件上按右鍵，除了編輯、剪下、複製、貼上、刪除之外，還有該變數的指令：**<變數> 的 I/O 對應...**、**<變數> 的 86HMI 綁定...**、**將 <變數> 加入監看視窗**、**加入趨勢圖**、**<變數> 用在哪裡？**、**跳到宣告**、**屬性**；模擬或線上時另有 **寫入**、**強制**、**解除強制**。

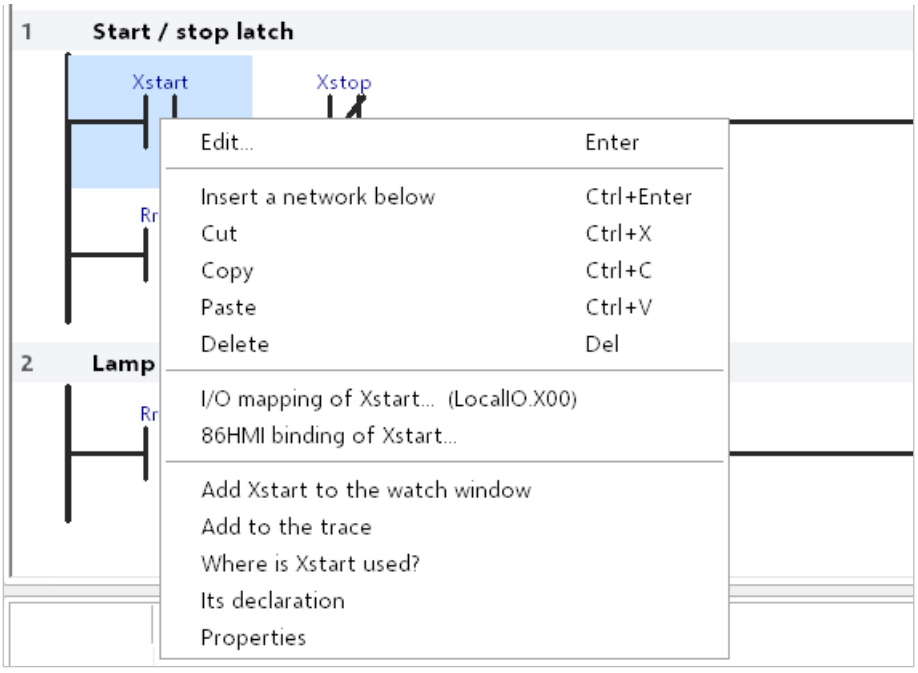


圖 7-3　元件的右鍵選單：編輯、網路、剪貼，以及變數的 I/O 對應、監看、交互參照

***注意**：模擬或線上時不能編輯階梯圖（狀態列顯示「請先停止模擬 / 離線後再編輯」）；此時雙擊接點或線圈會切換該變數的值。

7.1.3 網路

操作	方法
新增網路	「網路」的 **+ 網路** 或 Ctrl+Enter：插在選取的網路之後；Ctrl+Shift+Enter：插在之前
標題與註解	雙擊網路標題列；或右鍵「網路標題與註解...」；或「網路」的 **註解**
刪除網路	選取網路後 Del 或「網路」的 **刪除**
空白網路	顯示「（空的：選取此網路後插入元件）」，選取後即可插入

網路的視窗有三欄：**標題**（顯示在網路上方）、**註解**（多行，顯示在標題下方）、**內容（文字）**（這個網路的 .lad 文字，可以直接修改或貼上一整段邏輯，按確定時檢查語法）。

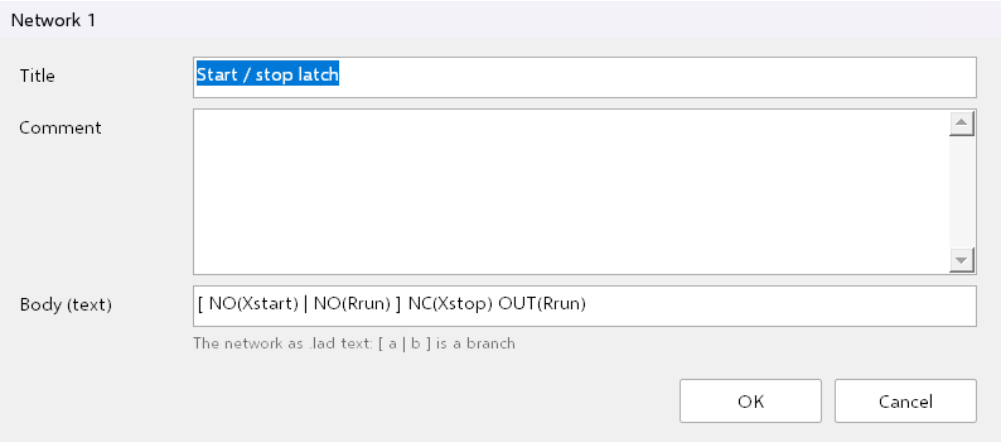


圖 7-4　網路的標題、註解與內容（.lad 文字）

7.1.4 文字模式

按「網路」的 **文字**（或 Ctrl+T）把整個 POU 切換成 .lad 文字編輯，適合大量修改、複製貼上或用版本控管比較差異；按上方的 **回到圖形** 時檢查格式。檔案有語法錯誤時，開啟就會停在文字模式並指出錯誤位置。



圖 7-5　文字模式：整個 POU 的 .lad 文字，上方為「回到圖形」

```
PROGRAM Main
VAR
    tLamp : TON;
END_VAR
NETWORK '啟動 / 停止自保'
    // 註解
    [ NO(Xstart) | NO(Rrun) ] NC(Xstop) OUT(Rrun)
NETWORK
    NO(Rrun) CALL(tLamp, IN := @, PT := T#2s, Q => @) OUT(Ylamp)
END_PROGRAM
```

- `PROGRAM` / `FUNCTION_BLOCK` 開頭，對應的 `END_…` 結尾；宣告區的寫法與 ST 相同。
- `NETWORK '標題'` 開始一個網路，其後的 `//` 行是註解。
- 一個網路是空白分隔的**串聯**元件；`[ a | b | c ]` 是**並聯**分支，每個分支本身可以是串聯，也可以再巢狀並聯。
- 元件寫成 `指令(參數, ...)`；沒有參數的指令只寫名稱（例如 `OSR`、`MCR`）。功能塊參數寫 `名稱 := 值`、輸出寫 `名稱 => 目標`，`@` 代表能流。

7.1.5 常用接線

右側欄「常用接線」提供現成的網路範本：選一項（下方顯示說明），按 **插入**（或雙擊），在表單中填入變數名稱與時間，網路就插在選取的網路之後，新變數自動宣告（輸入、輸出、內部位元宣告在 GVL，計時器與計數器宣告在這個 POU）。

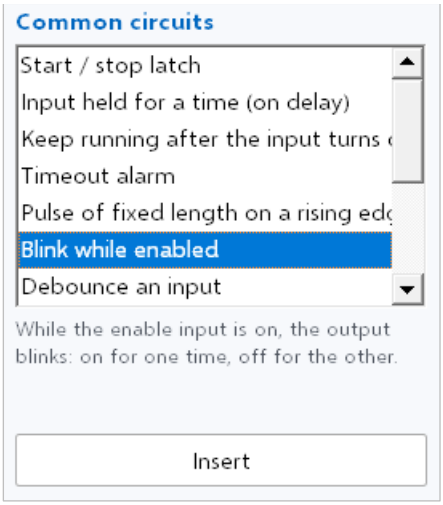


圖 7-6　常用接線清單：選一項後下方顯示說明，按「插入」

範本	產生的網路（.lad，名稱為預設值）
啟動 / 停止自保	`[ NO(Xstart) | NO(Ymotor) ] NC(Xstop) OUT(Ymotor)`
輸入持續一段時間（通電延遲）	`NO(Xsensor) TON(Tconfirm, T#2000ms) OUT(Yconfirmed)`
輸入關閉後延遲關閉（斷電延遲）	`NO(Xheater) TOF(Tafter, T#5000ms) OUT(Yfan)`
逾時警報	`NO(Rstep) NC(Xdone) TON(Ttimeout, T#10000ms) SET(Yalarm)`；`NO(Xreset) RST(Yalarm)`
上升緣固定長度脈波	`PT(Xtrigger) SET(Ypulse)`；`NO(Ypulse) TON(Tpulse, T#500ms) RST(Ypulse)`
致能時閃爍	兩個 TON 輪流 SET / RST 輸出，致能 OFF 時 RST
輸入去彈跳	`NO(Xswitch) TON(Tdb_on, T#50ms) SET(Rswitch)`；`NC(Xswitch) TON(Tdb_off, T#50ms) RST(Rswitch)`
計數到 N	`NO(Xpart) CTU(Cparts, 10) OUT(Ydone)`；`NO(Xreset) RES(Cparts)`
正反轉互鎖	正轉、反轉各一個自保網路，互相以常閉接點互鎖
液位警報（遲滯）	`GRT(Alevel, 3000) SET(Yhigh)`；`LES(Alevel, 2800) RST(Yhigh)`
單鍵 ON / OFF（切換）	`NO(Xbutton) TOGGLE(Ylight)`
方波（循環計時器）	`NO(Xrun) TCY(Tcycle, T#1000ms) OUT(Yclock)`
IEC 計時器功能塊（TON）	`NO(Xstart) CALL(tonStart, IN := @, PT := T#1000ms, Q => @) OUT(Ydelayed)`

7.2 接點與線圈

接點讀取 BOOL 值並決定能流是否通過；線圈依能流寫入 BOOL 變數。接點的變數可以是 GVL 或 VAR 中的 BOOL 變數，也可以是功能塊的 BOOL 輸出（例如 `NO(tLamp.Q)`）。

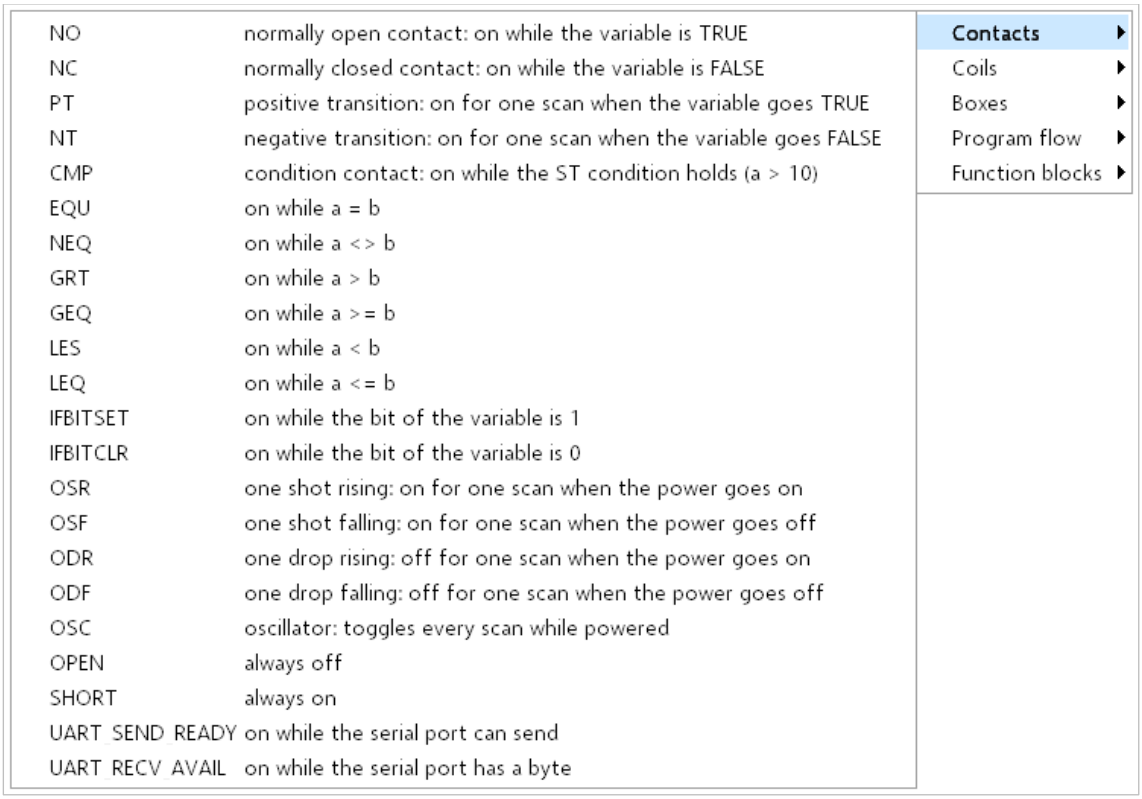


圖 7-7 「...」>「接點」：全部接點類指令

指令	.lad 寫法	按鍵	說明
NO	`NO(Xstart)`	C	常開接點：變數為 TRUE 時導通
NC	`NC(Xstop)`	N	常閉接點：變數為 FALSE 時導通
OPEN	`OPEN`	—	永遠斷開（例如暫時切斷一段迴路）
SHORT	`SHORT`	—	永遠導通（例如暫時跳過一個條件）
OUT	`OUT(Ylamp)`	L	線圈：變數跟隨能流（有能流 TRUE，無能流 FALSE）
OUTN	`OUTN(Yready)`	—	反相線圈：變數為能流的反相
SET	`SET(Yrun)`	S	置位線圈：有能流時設為 TRUE 並保持（沒有能流時不改變）
RST	`RST(Yrun)`	R	復位線圈：有能流時設為 FALSE 並保持
TOGGLE	`TOGGLE(Ylight)`	—	切換線圈：能流每次上升緣時反轉變數 [1.x]

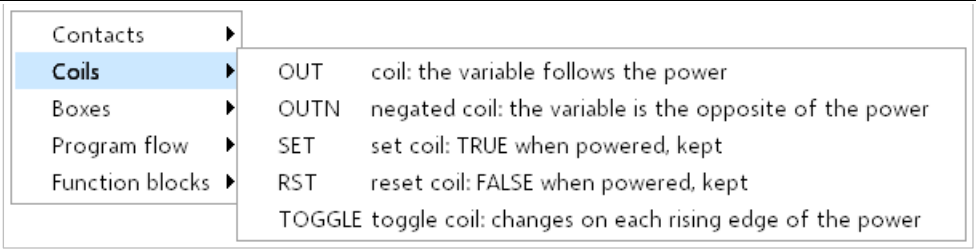


圖 7-8 「...」>「線圈」：OUT、OUTN、SET、RST、TOGGLE

*提示：同一個變數只用一個 OUT 線圈驅動；需要多個條件時把它們並聯或串聯在同一個線圈之前（第 6.0 節原則 ③）。SET / RST 成對使用時，寫在後面的網路優先。

7.3 邊緣與脈波

邊緣指令只在狀態改變的那一個掃描導通，用於「按一下做一次」的動作（計數、加 1、送出一筆資料）。PT、NT 偵測變數；OSR、OSF、ODR、ODF 偵測左邊進來的能流。

指令	.lad 寫法	按鍵	說明
PT	PT(Xstart)	P	正緣接點：變數由 FALSE 變 TRUE 時導通一次掃描
NT	NT(Xstart)	—	負緣接點：變數由 TRUE 變 FALSE 時導通一次掃描
OSR	NO(Xa) OSR	—	上升緣單發：能流導通時導通一次掃描 [1.x]
OSF	NO(Xa) OSF	—	下降緣單發：能流斷開時導通一次掃描 [1.x]
ODR	NO(Xa) ODR	—	上升緣斷開：能流導通時斷開一次掃描 [1.x]
ODF	NO(Xa) ODF	—	下降緣斷開：能流斷開時斷開一次掃描 [1.x]
OSC	NO(Xrun) OSC OUT(Yclk)	—	振盪器：通電時每個掃描切換一次（週期為 2 個掃描）[1.x]

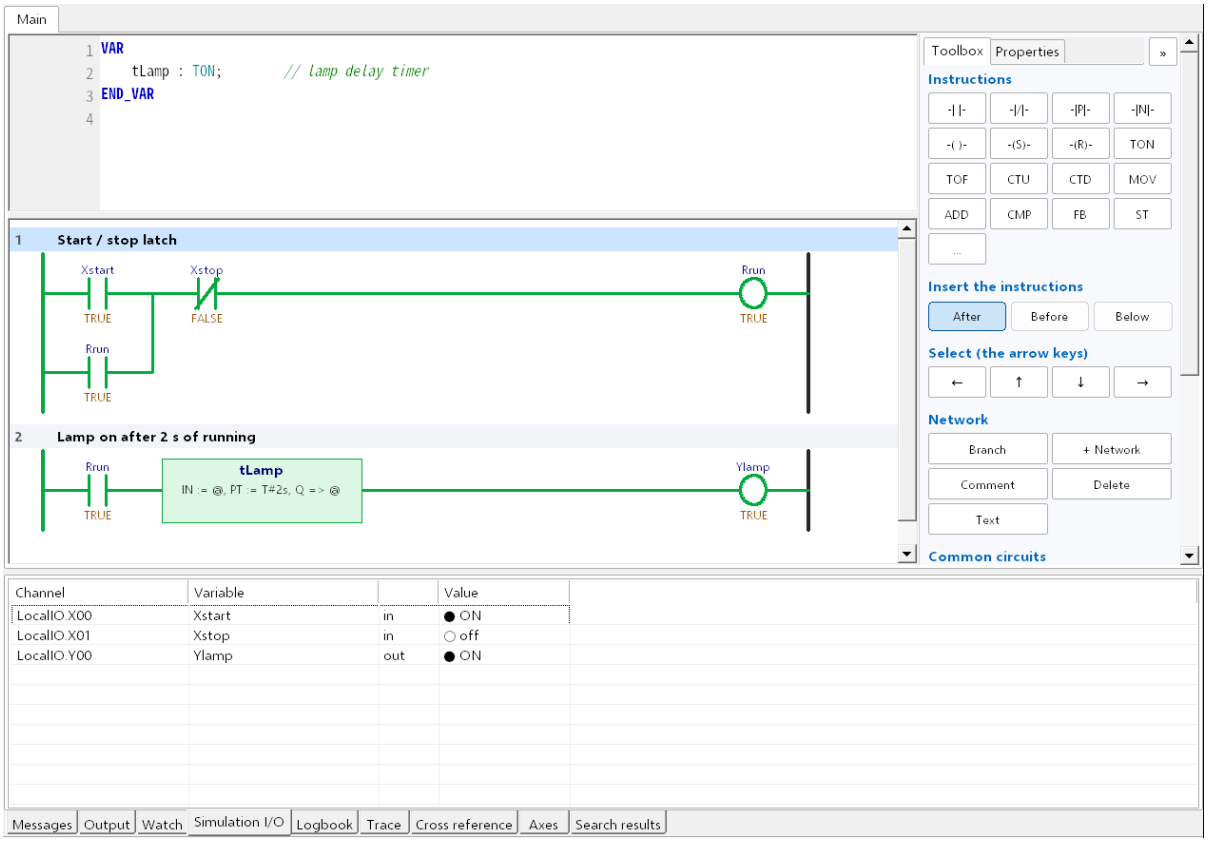


圖 7-9　邊緣接點的例子（網路 3）：PT(Xstart) 每按一次，ADD 把 Count 加 1

脈波輸出：下列 1.x 指令在數位輸出產生脈波列。輸出參數是一個在 I/O 對應中對應到實體輸出的 BOOL 變數。需要加減速的定位控制請改用 Motion86 軸（第 12 章）。

指令	.lad 寫法	按鍵	說明
NPULSE	`PT(Xgo) NPULSE(100, 1000, Ypls)`	—	能流上升緣時，在輸出送出 N 個脈波；參數：數量、頻率（Hz）、輸出變數 [1.x]
NPULSE_OFF	`NO(Xstop) NPULSE_OFF`	—	有能流時停止 N 脈波 [1.x]
PULSER	`NO(Xa) PULSER(10, 10, 0, pc, Ypls)`	—	在線圈變數輸出脈波列；參數：P1、P0（ON、OFF 的長度）、加速、計數器、線圈 [1.x]
STEPPER	`NO(Xb) STEPPER(stp, Nsteps, 1, 1, 0, Ystep)`	—	步進馬達脈波；參數：名稱、最大值（步數）、P、n、曲線、線圈。不支援加速表（n > 1 且曲線 > 0 時錯誤，請改用 NPULSE 或 Motion86）[1.x]

*提示：IEC 的邊緣偵測功能塊 `R_TRIG`、`F_TRIG` 以 CALL 使用，例如 `NO(Xa) CALL(rStart, CLK := @, Q => @) ADD(Count, Count, 1)`（7.8 節）。

7.4 計時器

計時器有兩種：**IEC 功能塊**（TON、TOF、TP，以實際時間計時，以 CALL 插入，7.8 節）與 **1.x 計時器方塊**（本節表格）。1.x 計時器的第一個參數是計時器變數（宣告為 `DINT` 等整數，記錄已經過的 PLC 週期數），計時以 PLC 週期計數：延遲 ÷ 工作週期，取整數。

- **延遲**：寫 `T#500ms`、`T#2s`。沒有單位的數字依 1.x 視為微秒，檢查時會出現警告。延遲短於工作週期時無法計時（警告）。
- **調整**（選填）：加到延遲上的 PLC 週期數（可為負），與 1.x 相同。
- TON、TOF、TCY、THI、TLO 在輸入改變時自動重設；RTO、RTL 會累計時間，要用 `RES` 清除（7.5 節）。

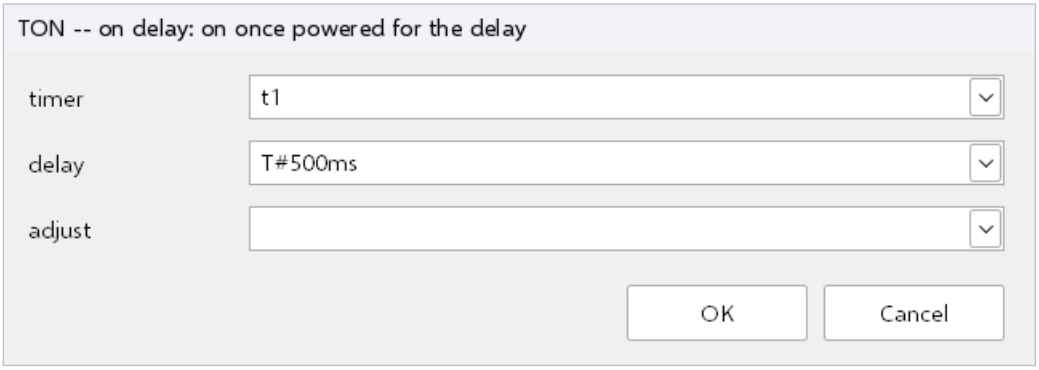


圖 7-10　1.x TON 的參數視窗：計時器、延遲、調整

指令	.lad 寫法	按鍵	說明
TON	`NO(Xa) TON(t1, T#500ms) OUT(Ya)`	O	通電延遲：通電滿延遲時間後導通 [1.x]
TOF	`NO(Xa) TOF(t2, T#1s) OUT(Ya)`	F	斷電延遲：斷電後仍導通延遲時間 [1.x]
RTO	`NO(Xa) RTO(t3, T#2s) OUT(Yb)`	—	累計通電延遲：累加通電時間，達到延遲後導通（RES 清除）[1.x]
RTL	`NO(Xa) RTL(t4, T#2s) OUT(Yb)`	—	斷電累計延遲：累加斷電時間 [1.x]
TCY	`NO(Xrun) TCY(t5, T#500ms) OUT(Yclk)`	—	循環計時器：通電時每經過延遲時間切換一次 ON / OFF [1.x]
THI	`NO(Xa) THI(t6, T#1s) OUT(Yp)`	—	上升緣時輸出延遲長度的脈波 [1.x]
TLO	`NO(Xa) TLO(t7, T#1s) OUT(Yp)`	—	下降緣時輸出延遲長度的低脈波 [1.x]
T2CNT	`NO(Xa) T2CNT(Dcycles, T#1s)`	—	把延遲換算成 PLC 週期數存入變數 [1.x]

IEC 計時器功能塊（建議）：實例宣告在 VAR（例如 `tLamp : TON;`），以 **FB**（K）插入。

功能塊	.lad 寫法	動作
TON	`NO(Xa) CALL(tLamp, IN := @, PT := T#2s, Q => @) OUT(Ya)`	IN 持續 TRUE 達 PT 後 Q = TRUE；IN 變 FALSE 時 Q 與 ET 立即歸零
TOF	`NO(Xa) CALL(tOff, IN := @, PT := T#5s, Q => @) OUT(Yb)`	IN 為 TRUE 時 Q = TRUE；IN 變 FALSE 後 Q 再保持 PT
TP	`NO(Xa) CALL(tPulse, IN := @, PT := T#1s, Q => @) OUT(Yc)`	IN 上升緣時 Q = TRUE 固定 PT，期間 IN 的變化不影響

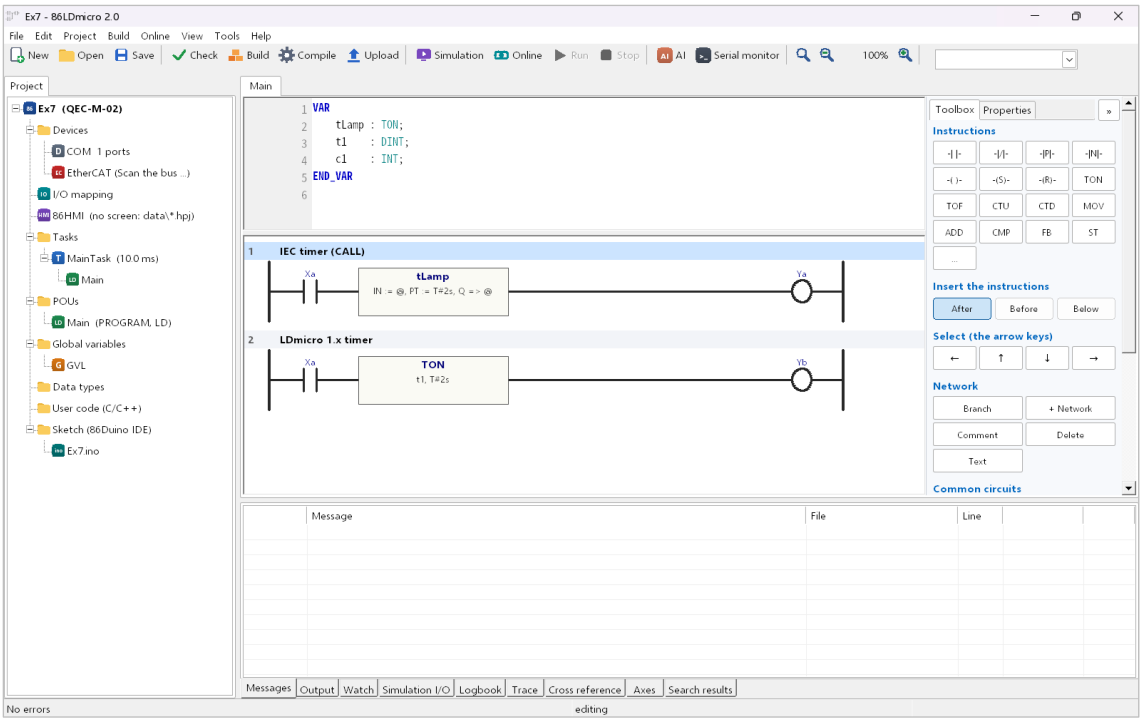


圖 7-11　IEC TON（上）與 1.x TON（下）在模擬中：功能塊顯示參數，1.x 方塊顯示已計數的週期

7.5 計數器

1.x 計數器的第一個參數是計數變數（宣告為 `INT` 等整數），其值就是目前的計數，可直接在比較指令中使用。IEC 計數器功能塊（CTU、CTD、CTUD）見 7.8 節。

- 參數：**計數器**（變數）、**最大值 / 最小值**、**初值**（選填，預設 0）、**輸入**（選填，計數的時機：`RISING`（預設，上升緣）、`FALLING`（下降緣）、`HIGH`（通電的每個掃描）、`LOW`（斷電的每個掃描））。
- `RES` 把計時器、計數器或變數清為 0；`RES` 是方塊，放在網路的最右側。

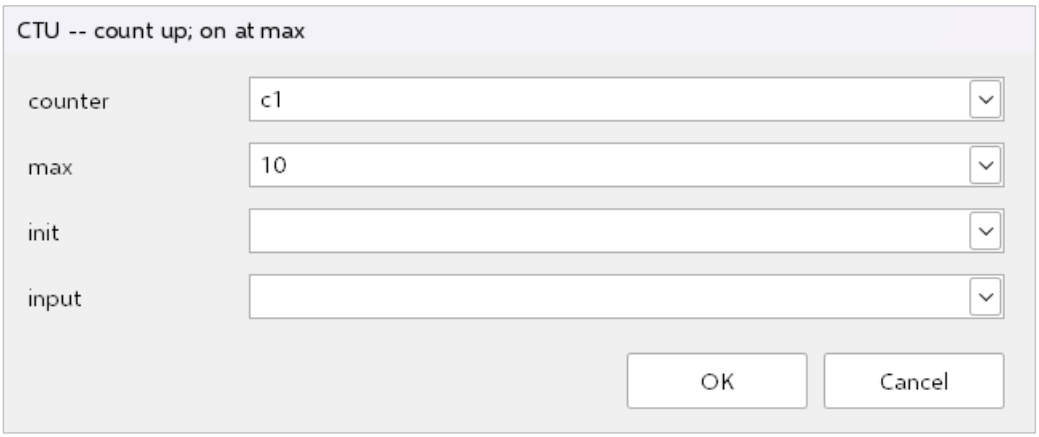


圖 7-12　1.x CTU 的參數視窗：計數器、最大值、初值、輸入

指令	.lad 寫法	按鍵	說明
CTU	`NO(Xa) CTU(c1, 10) OUT(Ya)`	U	遞增計數；計數 ≥ 最大值時導通 [1.x]
CTD	`NO(Xb) CTD(c2, 0, 10) OUT(Yb)`	D	遞減計數（此例初值 10）；大於最小值時導通 [1.x]
CTC	`NO(Xb) CTC(c3, 9, 0, FALLING)`	—	循環計數器：計數超過最大值時回到初值，回到初值的那一次導通（此例在下降緣計數）[1.x]
CTR	`NO(Xb) CTR(c4, 0, 9)`	—	循環遞減計數器：計數低於最小值時回到初值，回到初值的那一次導通 [1.x]
RES	`NO(Xreset) RES(c1)`	—	重置計時器 / 計數器 / 變數為 0 [1.x]

7.6 比較

比較類指令都是**接點**：條件成立時導通。建議使用 **CMP**（條件接點，按 E），直接寫 ST 的比較運算式，可以比較任何型別（INT、REAL、TIME、功能塊成員…）並組合 AND / OR；1.x 的 EQU ~ LEQ 仍可使用。

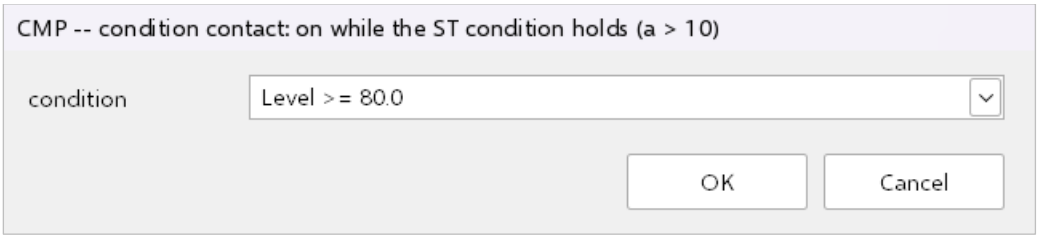


圖 7-13　CMP 條件接點的參數視窗：輸入 ST 條件

指令	.lad 寫法	按鍵	說明
CMP	`CMP(Level >= 80.0) OUT(Yhigh)`	E	條件接點：ST 條件成立時導通，例如 `Count = 5`、`tLamp.ET > T#1s`、`(a > 10) AND NOT b`
EQU	`EQU(Count, 10)`	—	a = b 時導通 [1.x]
NEQ	`NEQ(Count, 0)`	—	a <> b 時導通 [1.x]
GRT	`GRT(Level, 5)`	—	a > b 時導通 [1.x]
GEQ	`GEQ(Level, 5)`	—	a >= b 時導通 [1.x]
LES	`LES(Level, 90)`	—	a < b 時導通 [1.x]
LEQ	`LEQ(Level, 90)`	—	a <= b 時導通 [1.x]
IFBITSET	`IFBITSET(Nbits, 3)`	—	變數的第 n 個位元為 1 時導通（位元 0 為最低位元）[1.x]
IFBITCLR	`IFBITCLR(Nbits, 0)`	—	變數的第 n 個位元為 0 時導通 [1.x]

7.7 運算與搬移

運算與搬移指令都是**方塊**：有能流時執行一次運算（每個掃描都執行；只要做一次時，前面接 PT 等邊緣接點）。第一個參數是結果的目的變數。四則運算的任一運算元為 REAL 時以浮點計算，目的為整數時取最接近的整數。複雜的計算建議用 ST 方塊（7.8 節）或 ST 程式。

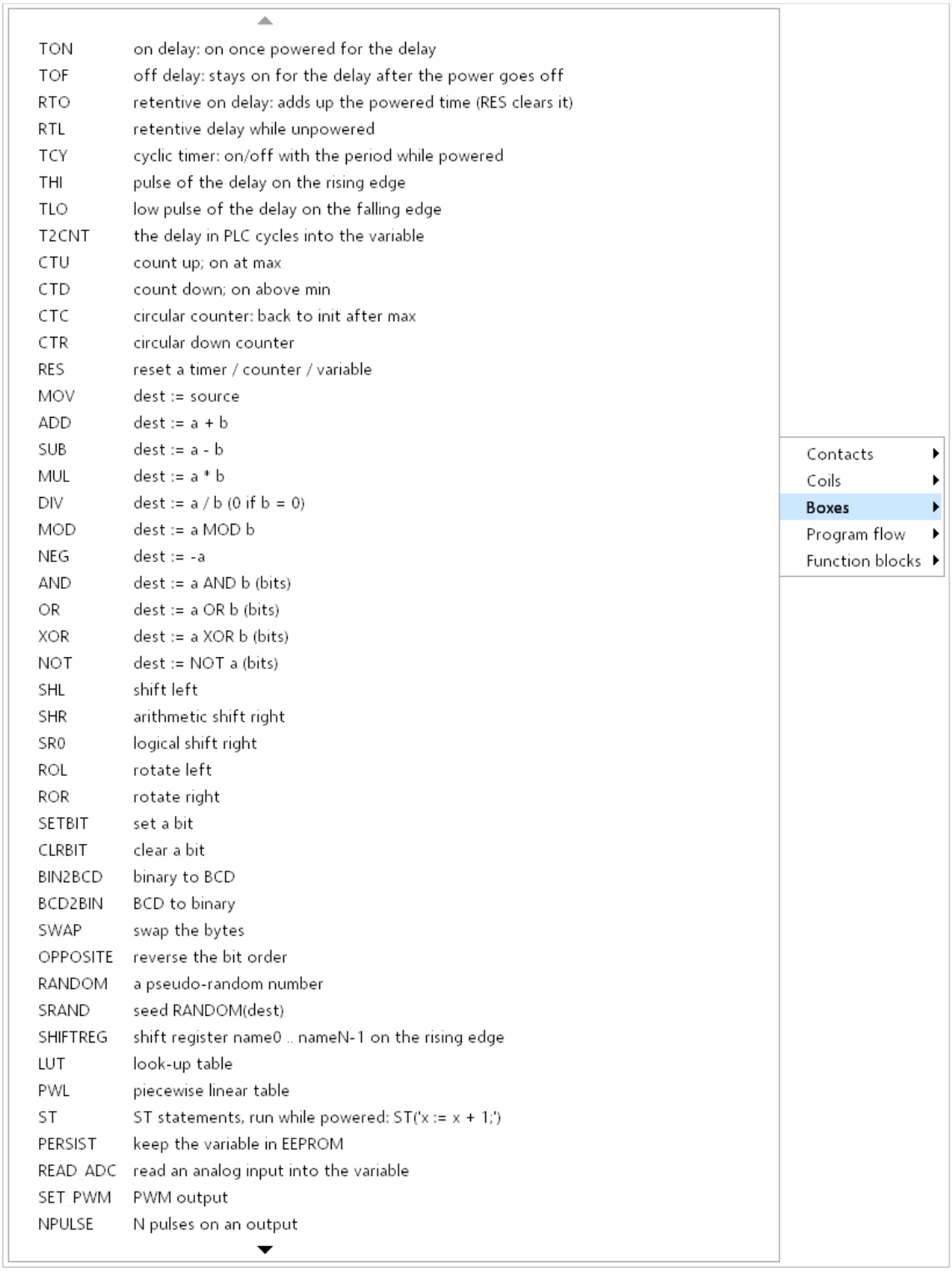


圖 7-14 「...」>「功能方塊」：計時器、計數器、運算、搬移與 I/O 方塊

7.7.1 搬移與四則運算

指令	.lad 寫法	按鍵	說明
MOV	`MOV(Val, 100)`	M	目的 := 來源
ADD	`ADD(Count, Count, 1)`	A	目的 := a + b
SUB	`SUB(Val, Val, 1)`	—	目的 := a - b
MUL	`MUL(Val, Val, 2)`	—	目的 := a * b
DIV	`DIV(Val, Val, 3)`	—	目的 := a / b（整數除以 0 時結果為 0）
MOD	`MOD(Val, Val, 7)`	—	目的 := a MOD b（餘數）
NEG	`NEG(Val, Val)`	—	目的 := -a

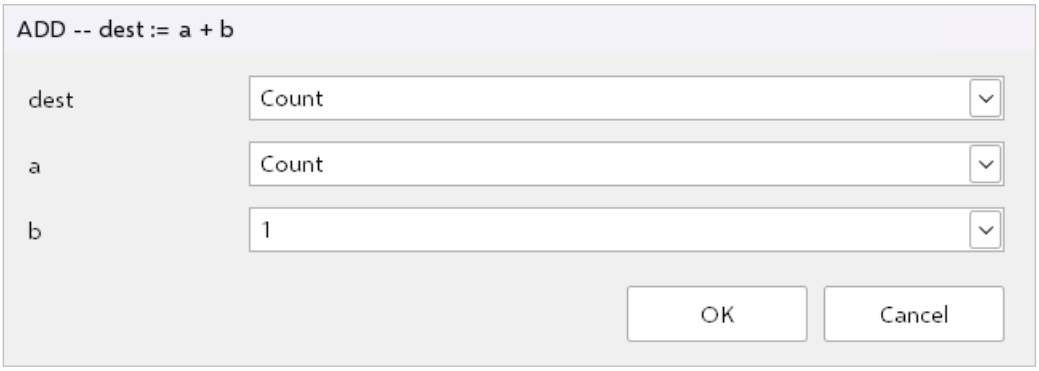


圖 7-15　ADD 的參數視窗：目的、a、b

7.7.2 位元運算

指令	.lad 寫法	按鍵	說明
AND	`AND(Nbits, Nbits, 16#0F)`	—	目的 := a AND b（逐位元）[1.x]
OR	`OR(Nbits, Nbits, 1)`	—	目的 := a OR b（逐位元）[1.x]
XOR	`XOR(Nbits, Nbits, 2)`	—	目的 := a XOR b（逐位元）[1.x]
NOT	`NOT(Nbits, Nbits)`	—	目的 := NOT a（逐位元反相）[1.x]
SHL	`SHL(Nbits, Nbits, 1)`	—	左移 n 位元 [1.x]
SHR	`SHR(Nbits, Nbits, 1)`	—	算術右移（保留正負號）[1.x]
SR0	`SR0(Nbits, Nbits, 1)`	—	邏輯右移（左側補 0）[1.x]
ROL	`ROL(Nbits, Nbits, 1)`	—	循環左移 [1.x]
ROR	`ROR(Nbits, Nbits, 1)`	—	循環右移 [1.x]
SETBIT	`SETBIT(Nbits, 2)`	—	把變數的第 n 個位元設為 1 [1.x]
CLRBIT	`CLRBIT(Nbits, 5)`	—	把變數的第 n 個位元清為 0 [1.x]

7.7.3 轉換、亂數與表格

指令	.lad 寫法	按鍵	說明
BIN2BCD	`BIN2BCD(Nbcd, Val)`	—	二進位轉 BCD [1.x]
BCD2BIN	`BCD2BIN(Val, Nbcd)`	—	BCD 轉二進位 [1.x]
SWAP	`SWAP(Nraw, Val)`	—	交換位元組 [1.x]
OPPOSITE	`OPPOSITE(Nraw, Val)`	—	反轉位元順序 [1.x]
RANDOM	`RANDOM(Val)`	—	虛擬亂數存入目的 [1.x]
SRAND	`SRAND(Val, 1234)`	—	設定 RANDOM 的種子 [1.x]
SHIFTREG	`NO(Xclk) SHIFTREG(sreg, 4)`	—	移位暫存器：能流上升緣時 `sreg0`→`sreg1`→…→`sreg3` 依序移位；`sreg0`～`sreg3` 要自行宣告（同型別）[1.x]
LUT	`LUT(Nout, Nidx, 10, 20, 40, 80)`	—	查表：目的 := 第「索引」個數值（索引從 0 開始）[1.x]
PWL	`PWL(Nout, Nraw, 0, 0, 100, 50, 200, 200)`	—	分段線性表：依 (x0, y0)、(x1, y1)… 的折線把索引換算成目的；x 須遞增，數值為整數，數值過大時檢查會提示縮小 [1.x]

7.8 IEC 功能塊（CALL）與 ST 指令

CALL 把任何功能塊放進網路：IEC 標準功能塊、PLCopen / Motion86 運動控制功能塊（7.11 節），以及專案中自己寫的 FUNCTION_BLOCK（第 8.3 節）。功能塊實例要先宣告在 VAR（或 GVL）。

1. 在宣告區宣告實例，例如 `tLamp : TON;`、`cParts : CTU;`、`rStart : R_TRIG;`。
2. 選取元件或網路，按 **FB**（K）。
3. 在「元件」欄輸入 `CALL(實例, 參數 := 值, 輸出 => 目標)`，按確定。

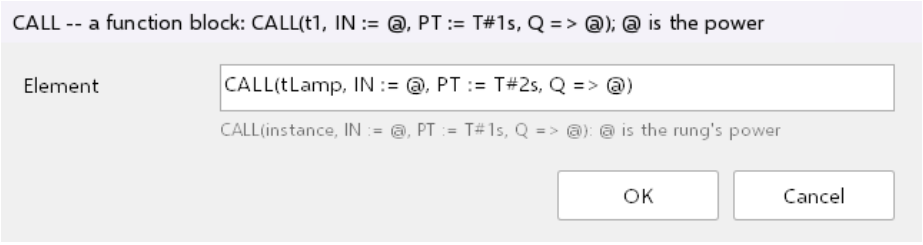


圖 7-16　CALL 的參數視窗：在「元件」欄輸入整個 CALL

- `IN := @`：把左邊進來的能流接到這個輸入；`Q => @`：這個輸出成為往右的能流（接到後面的元件）。
- 其他輸入可以接變數或常數（`R := Xreset`、`PV := 10`），輸出可以存到變數（`CV => Count`）。不用的參數省略即可。
- 功能塊的輸出也可以在其他網路直接讀取：`NO(tLamp.Q)`、`CMP(cParts.CV >= 5)`。

功能塊	輸入	輸出	.lad 寫法範例
TON / TOF / TP	IN, PT	Q, ET	`CALL(tLamp, IN := @, PT := T#2s, Q => @)`
CTU（CTU_DINT）	CU, R, PV	Q, CV	`CALL(cParts, CU := @, R := Xreset, PV := 10, Q => @, CV => Count)`
CTD（CTD_DINT）	CD, LD, PV	Q, CV	`CALL(cDown, CD := @, LD := Xload, PV := 5, Q => @)`
CTUD（CTUD_DINT）	CU, CD, R, LD, PV	QU, QD, CV	`CALL(cUD, CU := @, CD := Xdown, R := Xreset, PV := 10, QU => @)`
R_TRIG / F_TRIG	CLK	Q	`CALL(rStart, CLK := @, Q => @)`
SR（置位優先）	S1, R	Q1	`CALL(sr1, S1 := @, R := Xstop, Q1 => @)`
RS（復位優先）	S, R1	Q1	`CALL(rs1, S := @, R1 := Xstop, Q1 => @)`

ST 方塊：有能流時執行一段 ST 敘述（按 **ST** 或 T）。字串中的換行寫 `$N`。適合在階梯圖中做一小段計算，較長的邏輯請寫成 ST 程式（第 8 章）。

指令	.lad 寫法	按鍵	說明
CALL	`NO(Xa) CALL(tLamp, IN := @, PT := T#2s, Q => @) OUT(Ya)`	K	功能塊：@ 為能流
ST	`NO(Xa) ST('Count := Count + 1; Level := Count * 2;')`	T	ST 敘述，有能流時執行

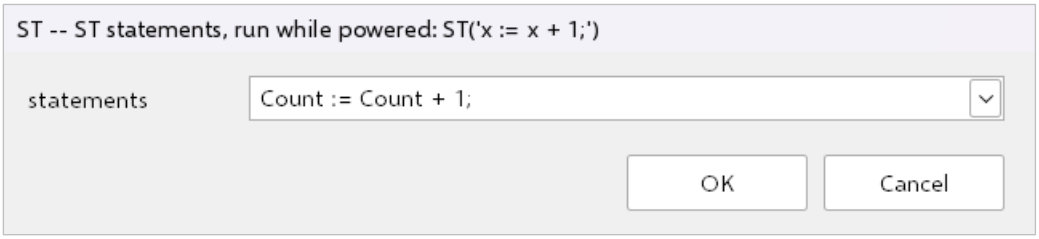


圖 7-17　ST 方塊的參數視窗：輸入 ST 敘述

*提示：「...」>「功能塊」子選單中只有 CALL；要插入哪一個功能塊，由 CALL 的第一個參數（實例）的型別決定。

7.9 流程控制

流程控制指令改變網路的執行順序，與 LDmicro 1.x 相同。一般程式很少需要；需要分步驟的流程建議改用 SFC（第 8.4 節），需要重複使用的邏輯建議寫成 FUNCTION_BLOCK。

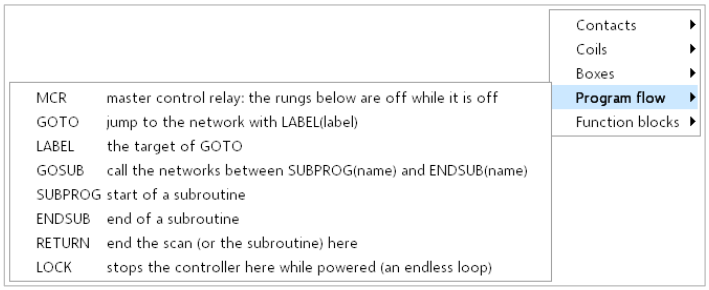


圖 7-18 「...」>「程式流程」：MCR、GOTO、LABEL、GOSUB、SUBPROG、ENDSUB、RETURN、LOCK

指令	.lad 寫法	按鍵	說明
MCR	`NO(Xauto) MCR ... MCR`	—	主控繼電器：條件為 OFF 時，其下方網路的能流全部斷開，直到下一個 MCR [1.x]
GOTO	`NO(Xskip) GOTO(skip)`	—	跳到含 `LABEL(skip)` 的網路（也可寫網路編號）[1.x]
LABEL	`LABEL(skip)`	—	GOTO 的目標，放在網路的開頭 [1.x]
GOSUB	`NO(Xa) GOSUB(Blink)`	—	呼叫 `SUBPROG(Blink)` 與 `ENDSUB(Blink)` 之間的網路 [1.x]
SUBPROG	`SUBPROG(Blink)`	—	副程式開始（單獨一個網路）[1.x]
ENDSUB	`ENDSUB(Blink)`	—	副程式結束（單獨一個網路）[1.x]
RETURN	`NO(Xb) RETURN`	—	在此結束本次掃描（在副程式中：返回呼叫處）[1.x]
LOCK	`NO(Xa) NO(Xb) NO(Xc) LOCK`	—	通電時讓控制器停在此處（無窮迴圈）[1.x]

```
NETWORK '自動模式才執行下面兩個網路'
    NO(Xauto) MCR
NETWORK
    NO(Xb) OUT(Ya)
NETWORK
    NO(Xc) OUT(Yb)
NETWORK '主控結束'
    MCR
NETWORK
    NO(Xa) GOSUB(Blink)
NETWORK
    SUBPROG(Blink)
NETWORK
    NO(Xa) OUTN(Yc)
NETWORK
    ENDSUB(Blink)
```

*安全：LOCK 通電時讓控制器停在無窮迴圈、不再掃描，只為 1.x 程式移轉而保留，新程式請勿使用。

7.10 I/O、序列埠與其他

本節為 LDmicro 1.x 的硬體相關指令。新專案的 I/O 一般直接用「I/O 對應」（第 4.6 節），序列通訊建議用 C/C++ 使用者程式（第 14 章）；下列指令主要供 1.1 程式移轉使用。

- **COM 埠參數**：寫 `COM0`、`COM1`...，依序對應專案「裝置 > COM」中的序列埠（`Serial`、`Serial1`...）。`Serial`（USB）保留給上傳與線上監看，與外部設備通訊請用 `Serial1`～`Serial3`。
- **類比與 PWM**：`READ_ADC` 的變數要在 I/O 對應中對應到類比輸入；`SET_PWM` 的腳位參數是在 I/O 對應中對應到 `Pin.n`（Arduino 腳位）的變數。

指令	.lad 寫法	按鍵	說明
READ_ADC	`NO(Xa) READ_ADC(Nadc)`	—	讀取類比輸入到變數（變數要在 I/O 對應中對應到類比輸入，例如 `Pin.n`）[1.x]
SET_PWM	`NO(Xa) SET_PWM(Nduty, 1000, Npwm)`	—	PWM 輸出；參數：責任比、頻率、腳位（對應 `Pin.n` 的變數）、解析度（選填）[1.x]
PERSIST	`NO(Xa) PERSIST(Count)`	—	將變數保存在 EEPROM：第一次通電的掃描讀回，之後值改變時寫入 [1.x]
UART_SEND	`PT(Xb) UART_SEND(Val, COM1)`	—	從序列埠傳送變數；參數：變數、COM、位元組數（選填）、等待（選填）[1.x]
UART_RECV	`UART_RECV_AVAIL(COM1)` `UART_RECV(Val, COM1)`	—	從序列埠接收一個位元組到變數（通常接在 UART_RECV_AVAIL 之後）；參數：變數、COM、位元組數（選填）、等待（選填）[1.x]
UART_SEND_READY	`UART_SEND_READY(COM1)`	—	接點：序列埠可傳送時導通 [1.x]
UART_RECV_AVAIL	`UART_RECV_AVAIL(COM1)`	—	接點：序列埠有資料時導通 [1.x]
FMTSTR	`PT(Xc) FMTSTR(Count, 'Count = \5\r\n', COM1)`	—	能流上升緣時把格式化字串送到序列埠（每個掃描一個字元）；`\1`～`\5` 為變數的位數，`\-5` 含正負號 [1.x]
STRING	`NO(Xa) STRING(sMsg, Count, 'N=%d')`	—	格式化字串存入 STRING 變數（C 的 printf 格式）[1.x]
DELAY	`NO(Xa) DELAY(10)`	—	忙碌等待（微秒），會延長掃描時間 [1.x]

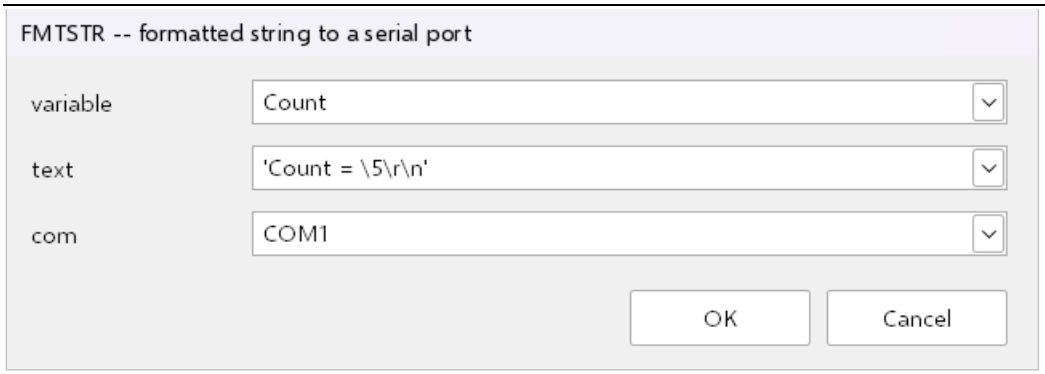


圖 7-19　FMTSTR 的參數視窗：變數、文字、COM

*注意：PERSIST 寫入 EEPROM（快閃記憶體）有壽命限制，請只用在不常改變的設定值；斷電保持的一般變數請宣告在 VAR_GLOBAL RETAIN。

DELAY 在掃描中忙碌等待，會延長掃描時間並可能觸發看門狗（第 10.5 節）。

7.11 運動控制指令

階梯圖中的運動控制有兩種寫法：

- **PLCopen MC 與 Motion86 功能塊**（建議）：`MC_Power`、`MC_MoveAbsolute`、`MC_Home`...，以及機台的 `M86_*`，以 **CALL** 插入，能流接到 Enable / Execute。`Axis :=` 填軸名，`Machine :=` 填機台名（第 12 章）。
- **1.x 的 MC 元件**：`MC(功能, 軸, a1, a2, a3, a4, 完成, 忙碌, 錯誤)`，供 LDmicro 1.1 移轉的程式使用；軸為已綁定 86EVA 伺服的變數，未用的參數填 `-`。

```
VAR
    pwrX  : MC_Power;
    homeX : MC_Home;
    mvX   : MC_MoveAbsolute;
END_VAR
NETWORK '激磁'
    NO(Xenable) CALL(pwrX, Axis := AxisX, Enable := @, Status => @) OUT(RpowerOk)
NETWORK '原點復歸'
    NO(RpowerOk) NO(Xhome) CALL(homeX, Axis := AxisX, Execute := @, Done => @) OUT(RhomeDone)
NETWORK '移動到 100 mm'
    NO(RpowerOk) NO(Xgo) CALL(mvX, Axis := AxisX, Execute := @, Position := 100.0, Velocity := 50.0,
Done => @) OUT(RmoveDone)
```

指令	.lad 寫法	按鍵	說明
CALL（MC_* / M86_*）	`NO(Xgo) CALL(mvX, Axis := AxisX, Execute := @, Position := 100.0, Velocity := 50.0, Done => @)`	K	PLCopen / Motion86 功能塊；功能塊一覽與參數見第 12.7、12.8 節
MC	`PT(Xgo) MC(MC_MoveAbsolute, Naxis, 100, 50, -, -, Rdone, Rbusy, Rerr)`	—	CiA 402 伺服的 PLCopen 運動方塊（LDmicro 1.x）；功能為 MC_Power、MC_MoveAbsolute、MC_MoveRelative、MC_MoveVelocity、MC_Home、MC_Stop、MC_Halt、MC_Reset、MC_ReadActualPosition、MC_ReadStatus、MC_ReadAxisError；未用參數填 - [1.x]

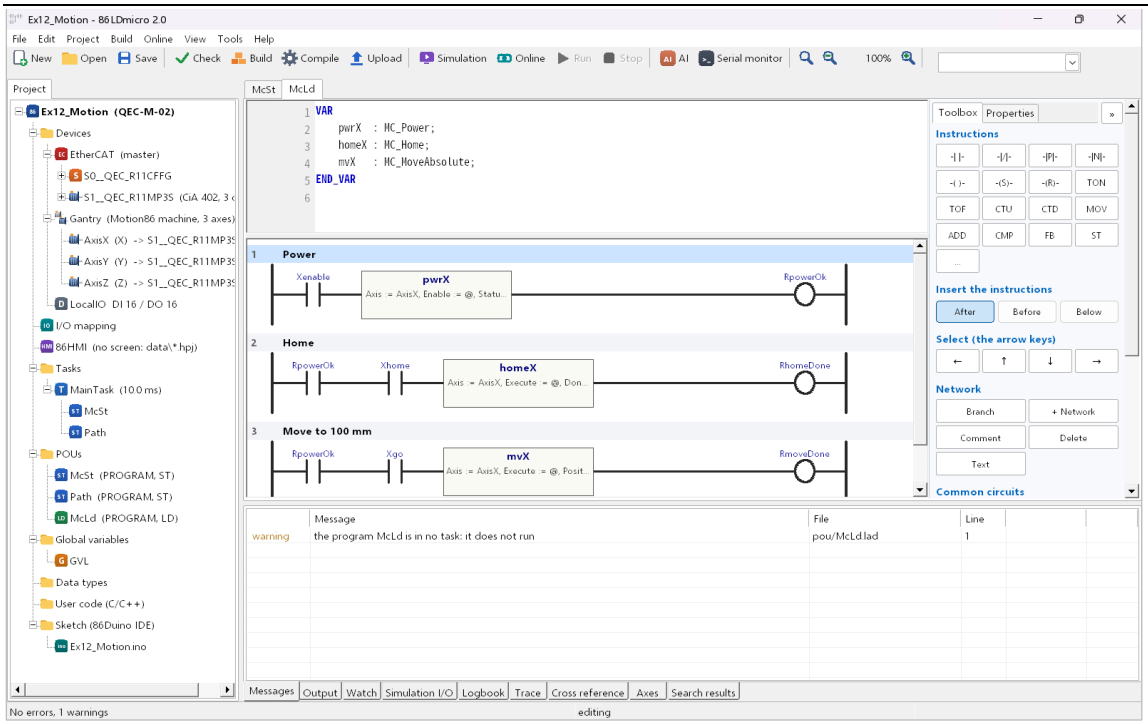


圖 7-20　階梯圖中的 PLCopen 功能塊：MC_Power、MC_Home、MC_MoveAbsolute

*安全：Execute 類功能塊（MoveAbsolute、Home…）在 Execute 的**上升緣**送出命令；要再送一次，先讓能流斷開。

功能塊實例要在每個掃描都被呼叫（網路不要被 MCR、GOTO 跳過），否則 Done / Busy 不會更新。第一次試車請先在機台頁、軸頁「線上」以低速確認方向與極限（第 12.5、12.6 節）。

第 8 章　ST 與 SFC

本章說明結構化文字（ST）與順序功能圖（SFC）的編輯器與語法。ST 適合計算、比較、迴圈與資料處理；SFC 適合有先後步驟的流程。兩者都是 IEC 61131-3 語言，與階梯圖（LD）可以放在同一個專案、同一個工作中依序執行。完整的範例請見 6.5 節（ST：注水控制）與 6.6 節（SFC：順序控制）。

8.1　ST 編輯器

ST 程式以「專案 > 新增 PROGRAM（ST）...」建立，也可在專案樹的 POU 上按右鍵新增。輸入名稱按 **確定** 後，檔案 `pou/<名稱>.st` 立即開啟，新的 PROGRAM 自動加入第一個工作。

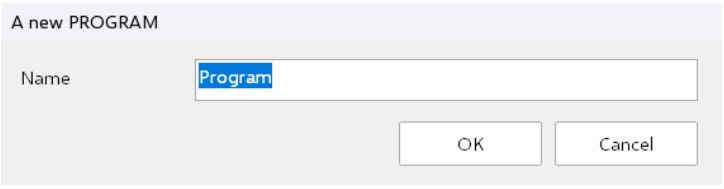


圖 8-1　新增 PROGRAM（ST）：輸入程式名稱

ST 編輯器分為兩區：左邊是程式文字（宣告區 VAR … END_VAR 與程式本體寫在同一個檔案），右側欄是「工具箱」中的 **ST 常用片段**。

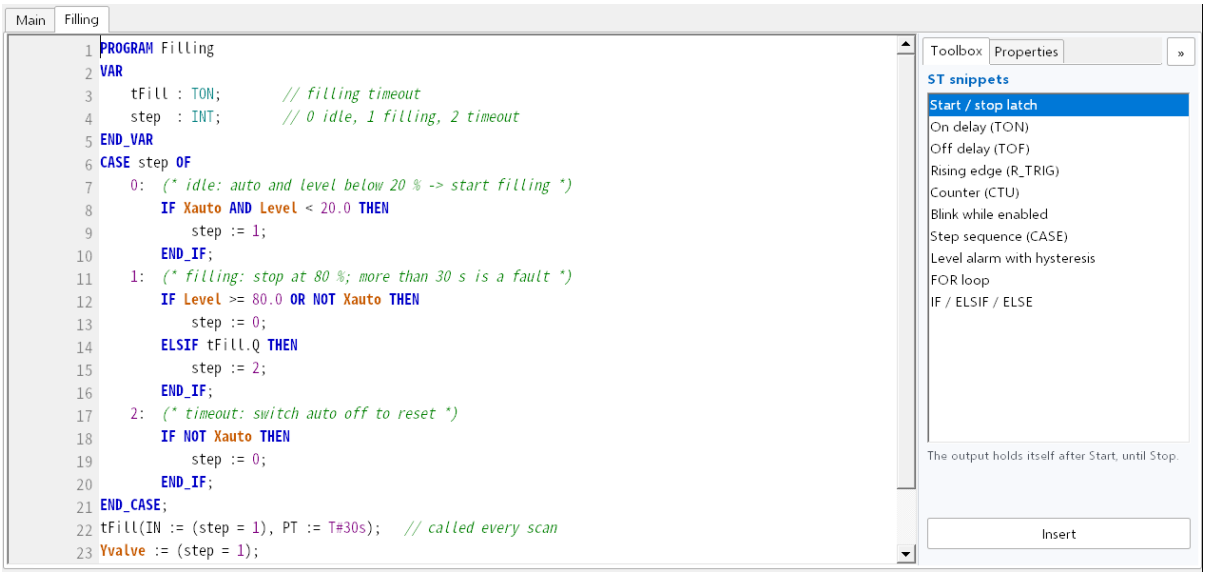


圖 8-2　ST 編輯器：左邊為程式文字，右側為「ST 常用片段」

8.1.1 語法顏色

編輯器依語法自動上色，打錯字時很容易看出來：

顏色	內容	例
藍色粗體	關鍵字	IF、THEN、END_IF、VAR、CASE
藍綠色	資料型別	BOOL、INT、REAL、TON
綠色斜體	註解	(* … *)、// …
深紅色	字串	'G1 X10 Y10'
紫色	數值、時間常數	100、16#FF、T#2s
黑色	變數名稱與其他	Level、tFill

有錯誤的地方以底線標示，滑鼠停在底線上顯示錯誤說明；按 **檢查**（Shift+F7）後，錯誤也列在「訊息」視窗（9.6 節）。



圖 8-3　錯誤底線：滑鼠停留顯示錯誤說明

8.1.2 補全與參數提示

輸入變數或關鍵字時，補全清單會自動開啟，列出符合的名稱與其型別、出處（GVL 或區域變數）與註解：

- 輸入名稱的**前兩個字母**時自動開啟；輸入 `.`（例如 `tFill.` 列出 Q、ET 等成員）、`#`、`(` 時也會開啟。
- 以 ↑↓（或 PageUp / PageDown）選擇，**Enter** 或 **Tab** 插入，**Esc** 關閉（同一個字不再自動開啟）。
- 輸入功能塊或函式的 `(` 或 `,` 時，畫面顯示其參數提示（例如 `TON(IN, PT)`），輸入 `)` 或 `;` 時消失。
- 註解與字串中不開啟補全。

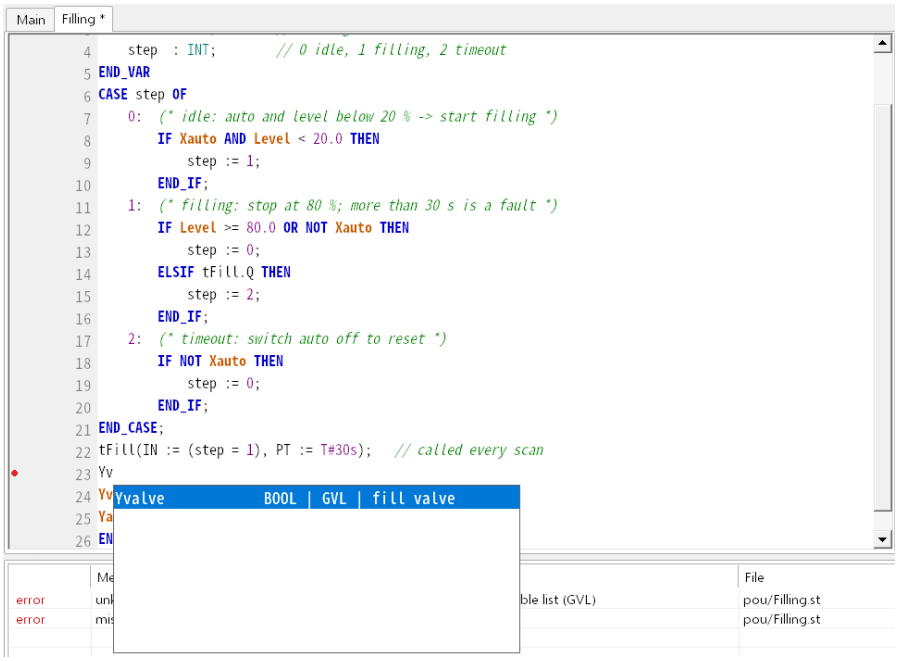


圖 8-4　補全：輸入 Yv 即列出 Yvalve（BOOL · GVL · 註解）

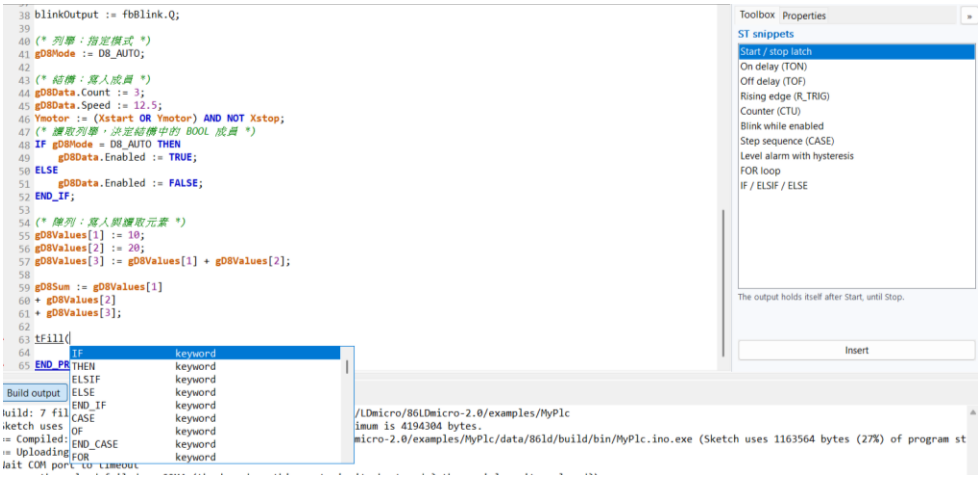


圖 8-5　參數提示：輸入 tFill(時顯示 TON 的輸入

> ***注意**：86LDmicro 2.0.0 中，**Ctrl+Space 不會開啟補全清單**；請直接輸入名稱的前兩個字母（或 `.`），清單會自動出現。

8.1.3 ST 常用片段

右側欄的 **ST 常用片段** 提供常用寫法的範本。選一個片段，下方顯示說明，按 **插入**（或雙擊）開啟表單，填入變數名稱後按 **確定**，程式碼插入游標處；片段需要的功能塊實例（例如 TON）自動加入程式的 VAR，新的全域變數自動加入 GVL。

片段	用途
啟動 / 停止自保	按啟動後輸出自我保持，按停止解除
通電延遲（TON）	輸入持續 ON 達設定時間後，輸出 ON
斷電延遲（TOF）	輸入 OFF 後，輸出再保持設定時間
上升緣（R_TRIG）	輸入每次由 OFF 變 ON 時執行一次
計數器（CTU）	計數上升緣，到設定值時輸出 ON，可重置
致能時閃爍	致能期間，輸出每半個週期反轉一次
步序控制（CASE）	狀態機：每一步完成後跳到下一步
液位警報（遲滯）	高於上限時 ON，低於下限才 OFF
FOR 迴圈	i 由 1 到 N 重複執行
IF / ELSIF / ELSE	條件判斷

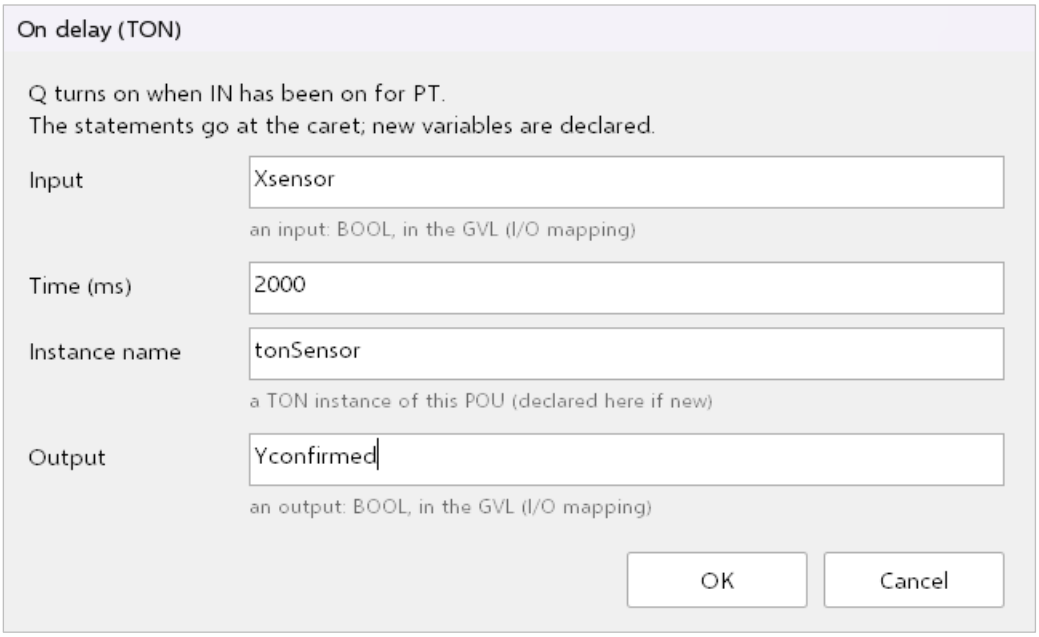


圖 8-6　插入「通電延遲（TON）」片段：填入輸入、時間、實例名稱與輸出

8.1.4 自動排版、數值顯示與其他操作

操作	按鍵	說明
自動排版	Ctrl+T	重新縮排 VAR、IF / ELSIF / ELSE、CASE、FOR / WHILE / REPEAT；狀態列顯示「已自動排版（Ctrl+T）；Ctrl+Z 可復原」
縮排 / 取消縮排	Tab / Shift+Tab	插入 4 個空白；Shift+Tab 移除行首最多 4 個空白
自動縮排	Enter	新的一行沿用上一行的縮排；在 THEN、DO、OF 等之後多縮一層
復原 / 重做	Ctrl+Z / Ctrl+Y	—
尋找	Ctrl+F、F3、Shift+F3	在此分頁尋找（3.7 節）
變數的選單	在變數上按右鍵	加入監看、加入趨勢圖、「<變數> 用在哪裡？」、跳到宣告、I/O 對應、86HMI 綁定；模擬或線上時另有寫入、強制

滑鼠停在變數上會顯示其型別；**模擬或線上時**另外顯示目前的數值，不必把變數加入監看。

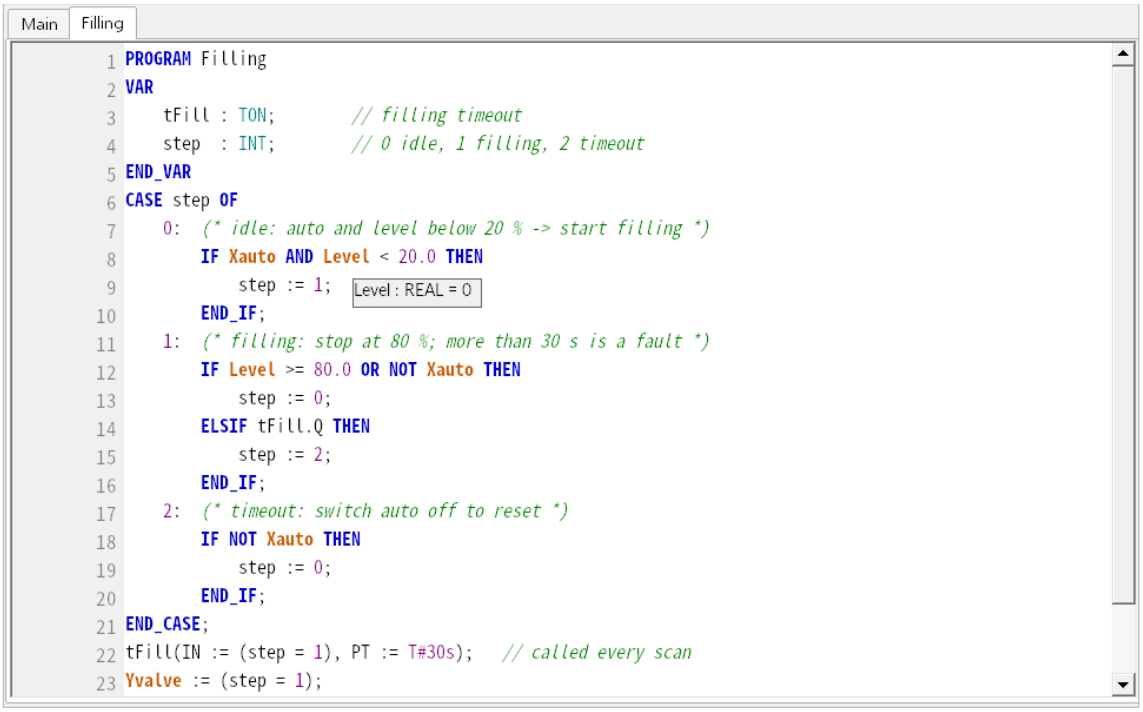


圖 8-7　模擬中：滑鼠停在變數上顯示型別與目前數值

8.2 ST 語法速查

下表整理 86LDmicro 的 ST 常用語法（IEC 61131-3）。表中的寫法都已在 86LDmicro 2.0.0 中確認；它們是片段，要貼上檢查時，請放在宣告了相關變數的程式中（例如 4.5 節的 Recipe、Axis1）。

8.2.1 程式結構與宣告

```
PROGRAM 名稱                    // 或 FUNCTION_BLOCK 名稱、FUNCTION 名稱 : 型別
VAR                             // 區域變數（PROGRAM、FUNCTION_BLOCK 的值會保留到下一個掃描）
    tFill : TON;                // 功能塊實例
    step  : INT := 0;           // 可給初值
    buf   : ARRAY[0..9] OF INT; // 陣列
    s     : STRING[40];         // 字串（最大長度）
END_VAR
VAR CONSTANT
    MAXN : INT := 10;           // 常數
END_VAR
(* 程式本體 *)
END_PROGRAM
```

區塊	用在	說明
VAR … END_VAR	全部	區域變數
VAR CONSTANT	全部	常數，不可寫入
VAR_TEMP	全部	暫存變數：每次呼叫都從初值開始，不保留到下一次
VAR_INPUT	FUNCTION、FUNCTION_BLOCK	輸入參數
VAR_OUTPUT	FUNCTION、FUNCTION_BLOCK	輸出（功能塊：呼叫後以 實例.名稱 讀取；函式：呼叫時以 名稱 => 變數 取回）
VAR_IN_OUT	FUNCTION、FUNCTION_BLOCK	傳入的變數本身，可以直接改寫它；傳入的必須是呼叫端 ST 程式的區域變數（不能是常數、運算式或全域變數）
VAR_GLOBAL	全域變數表（GVL）	全專案共用的變數（4.5 節）

PROGRAM 不能宣告 VAR_INPUT、VAR_OUTPUT、VAR_IN_OUT（檢查時會出錯）。上表為 86LDmicro 2.0.0 的實際支援範圍。

8.2.2 敘述

項目	寫法
指定	`x := y + 1;`
註解	`(* 多行 *)`、`// 單行`
條件	`IF a THEN ... ELSIF b THEN ... ELSE ... END_IF;`
多重選擇	`CASE step OF 0: ...; 1, 2: ...; 10..20: ...; ELSE ... END_CASE;`
FOR 迴圈	`FOR i := 1 TO 10 BY 2 DO ... END_FOR;`
WHILE 迴圈	`WHILE n > 0 DO ... END_WHILE;`
REPEAT 迴圈	`REPEAT ... UNTIL n >= 3 END_REPEAT;`
離開迴圈	`EXIT;`
結束本次執行	`RETURN;`（PROGRAM 在本次掃描不再往下執行）
呼叫功能塊	`t1(IN := start, PT := T#2s);  IF t1.Q THEN ... END_IF;`
呼叫函式	`y := LIMIT(0.0, x, 100.0);  n := REAL_TO_INT(r);`
陣列、結構	`buf[i] := 0;  Recipe[2] := 10;  Axis1.pos := 10.0;`

8.2.3 運算子與常數

類別	寫法
算術	`+  -  *  /  MOD`（整數除以 0 結果為 0）
比較	`=  <>  <  <=  >  >=`
邏輯（BOOL）與位元（WORD 等）	`AND  OR  XOR  NOT`
整數	`100`、`-5`、`16#FF`（16 進位）、`2#1010`（2 進位）
實數	`1.5`、`1.5E3`
時間	`T#2s`、`T#1s500ms`、`T#100ms`
布林、字串	`TRUE`、`FALSE`、`'文字'`

8.2.4 常用函式與功能塊

類別	名稱
數學	ABS、SQRT、SIN、COS、EXPT、TRUNC
選擇與限制	MIN、MAX、LIMIT、SEL、MUX
位元	SHL、SHR、ROL、ROR
字串	LEN、CONCAT、LEFT、RIGHT、MID、FIND
型別轉換	<型別>_TO_<型別>，例如 INT_TO_REAL、REAL_TO_DINT、INT_TO_STRING、STRING_TO_REAL
計時器	TON（通電延遲）、TOF（斷電延遲）、TP（脈衝）
計數器	CTU、CTD、CTUD
邊緣	R_TRIG（上升緣）、F_TRIG（下降緣）
正反器	SR（設定優先）、RS（重置優先）
運動控制	MC_*（PLCopen，12.7 節）、M86_*（Motion86 機台，12.8 節）

*提示：功能塊實例（例如 TON）要**每個掃描都呼叫**，計時器才會累計時間。把呼叫寫在 IF 或 CASE 裡面時，條件不成立的掃描不會更新；穩健的寫法是放在 IF / CASE 之外，以條件當輸入，例如 `tFill(IN := (step = 1), PT := T#30s);`。

8.3 自訂 FUNCTION 與 FUNCTION_BLOCK

重複使用的運算可以寫成自己的 FUNCTION 或 FUNCTION_BLOCK，在 ST、LD、SFC 中呼叫，就像內建的函式與功能塊：

種類	記憶	建立方式	使用方式
FUNCTION	沒有實例狀態：每次呼叫從頭計算，適合計算與換算（只依參數計算時，相同參數得到相同結果）	「專案 > 新增 FUNCTION（ST）...」	`y := Scale(raw, 0.0, 100.0);`
FUNCTION_BLOCK	有：每個實例保留自己的內部變數（例如計時器）	「專案 > 新增 FUNCTION_BLOCK（ST）...」或「新增 FUNCTION_BLOCK（階梯圖）...」	先宣告實例 `bl : Blinker;`，再呼叫 `bl(...)`

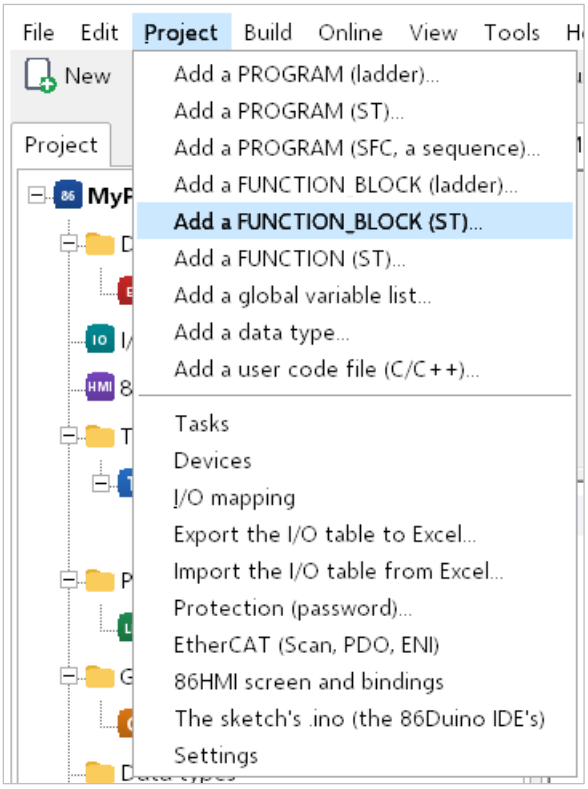


圖 8-8 「專案」選單：新增 FUNCTION_BLOCK（ST）與新增 FUNCTION（ST）

8.3.1 範例：類比換算與閃爍

情境：類比輸入的原始值 `AnalogIn`（0～4095）要換算成液位百分比 `Level`；運轉中（`Rrun`）時，指示燈 `Yblink` 每 0.5 秒亮滅一次。換算寫成 FUNCTION `Scale`，閃爍寫成 FUNCTION_BLOCK `Blinker`，由 ST 程式 `Panel` 呼叫。

變數	型別	I/O 對應	用途
Rrun	BOOL	—（由其他程式寫入）	運轉中
AnalogIn	INT	類比輸入（4.6 節）	類比原始值 0～4095
Level	REAL	—	液位（%）
Yblink	BOOL	數位輸出	閃爍燈

FUNCTION Scale（`pou/Scale.st`）：函式名稱就是傳回值，在函式內指定給它。

```
FUNCTION Scale : REAL                  // 類比換算：0～4095 → lo～hi
VAR_INPUT
    raw : INT;
    lo, hi : REAL;
END_VAR
Scale := lo + (hi - lo) * INT_TO_REAL(raw) / 4095.0;
END_FUNCTION
```

FUNCTION_BLOCK Blinker（`pou/Blinker.st`）：內部的 TON 與輸出 `Out` 在每個實例中各自保留。

```
FUNCTION_BLOCK Blinker                 // 閃爍：有記憶的功能塊
VAR_INPUT
    Enable : BOOL;
    Half   : TIME;
END_VAR
VAR_OUTPUT
    Out : BOOL;
END_VAR
VAR
    t : TON;
END_VAR
t(IN := Enable AND NOT t.Q, PT := Half);   // 每半個週期反轉一次
IF t.Q THEN
    Out := NOT Out;
END_IF;
IF NOT Enable THEN
    Out := FALSE;
END_IF;
END_FUNCTION_BLOCK
```

PROGRAM Panel（pou/Panel.st，放在工作中執行）：

```
PROGRAM Panel
VAR
    bl : Blinker;                       // 功能塊實例
END_VAR
Level := Scale(AnalogIn, 0.0, 100.0);  // 0～4095 → 0～100 %
bl(Enable := Rrun, Half := T#500ms);
Yblink := bl.Out;
END_PROGRAM
```

同一個功能塊也可以在階梯圖中以 **CALL** 使用（7.8 節）。下面的網路宣告另一個實例 bl2，能流接到 Enable，Out 驅動線圈 Ylamp2：

```
PROGRAM Main
VAR
    bl2 : Blinker;
END_VAR
NETWORK '自訂功能塊：閃爍'
    NO(Rrun) CALL(bl2, Enable := @, Half := T#250ms, Out => @) OUT(Ylamp2)
END_PROGRAM
```

#	操作
1	「專案 > 新增 FUNCTION（ST）...」，名稱 Scale，輸入上面的程式。
2	「專案 > 新增 FUNCTION_BLOCK（ST）...」，名稱 Blinker，輸入上面的程式。
3	在 GVL 宣告 Rrun、AnalogIn、Level、Yblink（Ylamp2）。
4	「專案 > 新增 PROGRAM（ST）...」，名稱 Panel，輸入上面的程式（自動加入第一個工作）。
5	按 **檢查**（Shift+F7），確認「訊息」沒有錯誤。
6	按 **模擬**（F5），在監看視窗加入 Rrun、AnalogIn、Level、Yblink；把 AnalogIn 寫成 2048、Rrun 寫成 TRUE。

預期結果：AnalogIn = 2048 時 Level 約為 50.01；Rrun 為 TRUE 時 Yblink 約每 0.5 秒反轉一次（階梯圖的 Ylamp2 約每 0.25 秒反轉一次）；Rrun 為 FALSE 時兩者都熄滅。

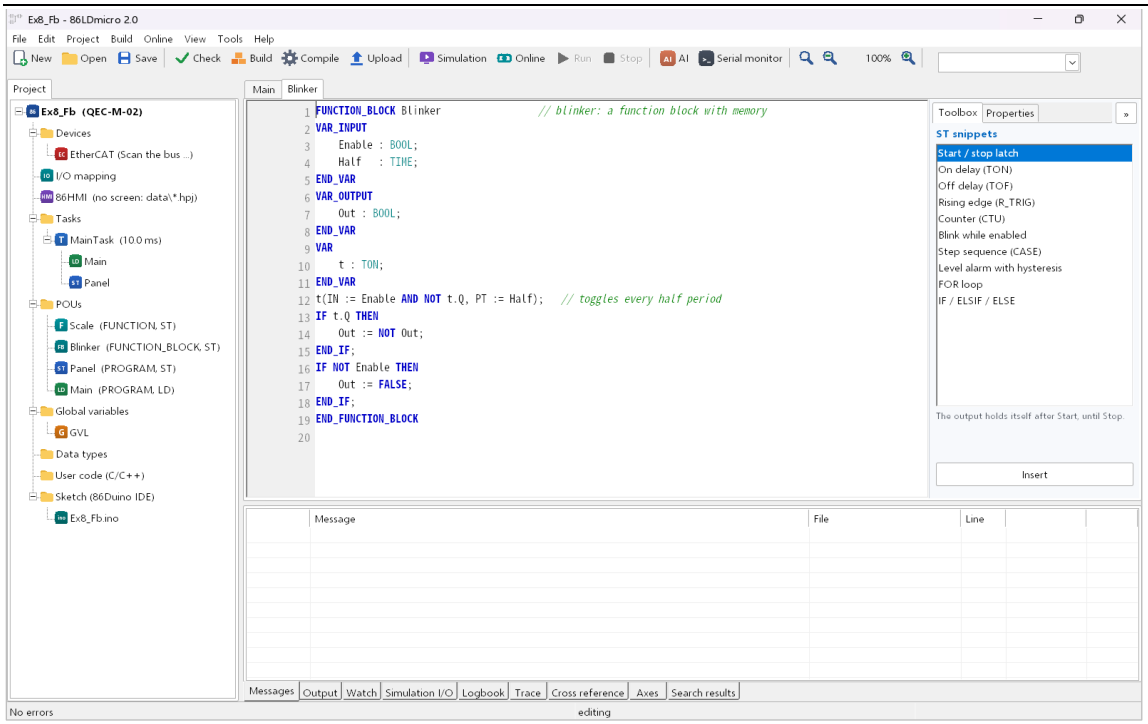


圖 8-9　FUNCTION_BLOCK Blinker 的編輯畫面

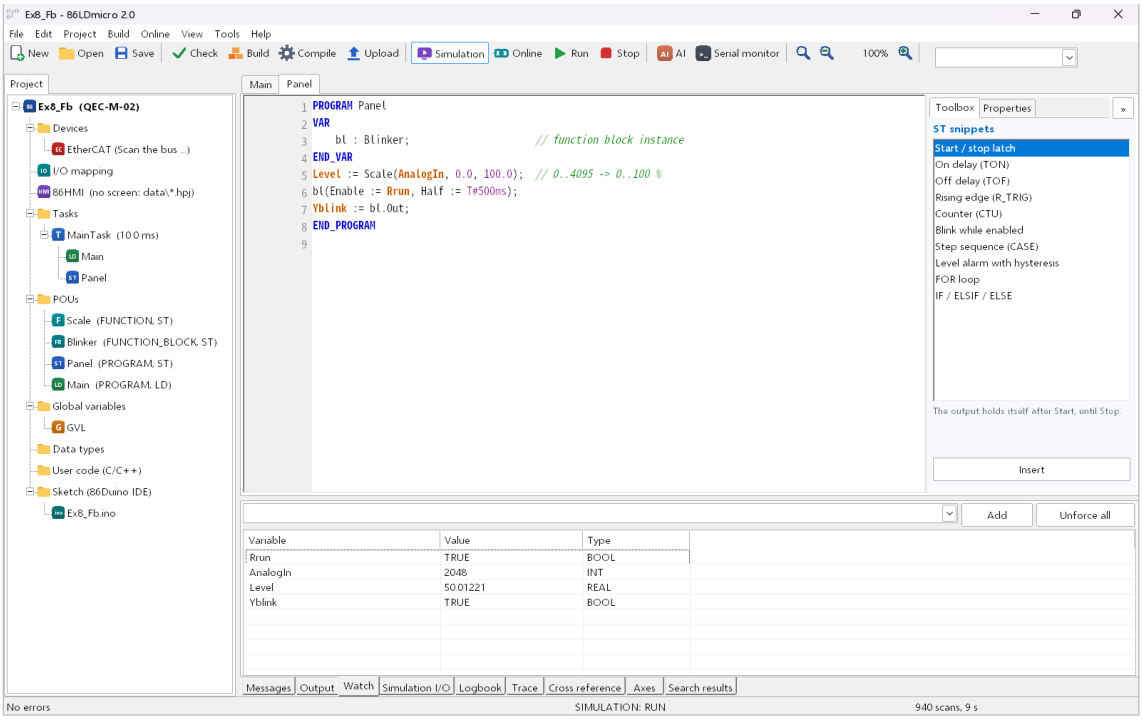


圖 8-10　模擬中：Level 為 50.01、Yblink 閃爍

***提示：**FUNCTION 的區域變數每次呼叫都從初值開始（不保留）；需要計時器、計數器或邊緣偵測等有記憶的功能時，請寫成 FUNCTION_BLOCK。FUNCTION 與 FUNCTION_BLOCK 不放進工作，由呼叫它的程式決定何時執行。

8.4 SFC 編輯器

SFC 以**步驟**（Step）與**轉移條件**（Transition）描述流程：只有啟動中的步驟執行其動作，轉移條件成立時移到下一步。適合機台的自動流程、配方程序等。

以「專案 > 新增 PROGRAM（SFC，順序流程圖）...」或專案樹 POU 右鍵建立，檔案為 `pou/<名稱>.sfc`，與 LD、ST 一樣放進工作中執行。新的 SFC 已有一個簡單的圖表（初始步驟 Init → 步驟 Step1 → 1 秒後跳回 Init），可直接修改。

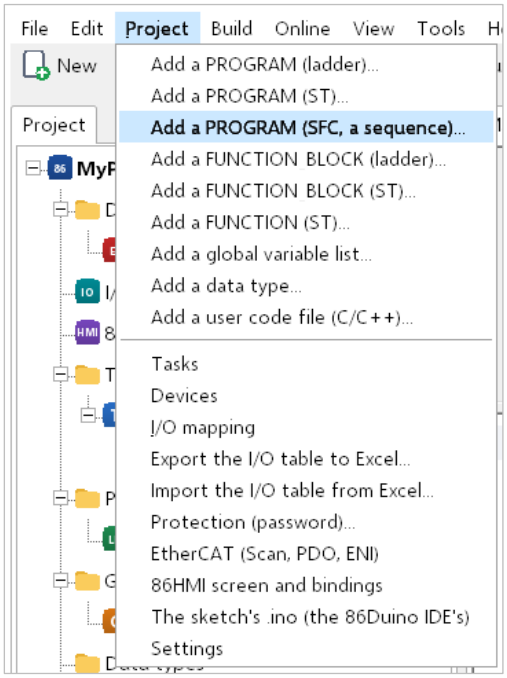


圖 8-11　「專案 > 新增 PROGRAM（SFC，順序流程圖）...」

SFC 編輯器分為三區：上方是**宣告區**（VAR ... END_VAR），中間是**圖表**，右側欄是「**圖表**」工具按鈕。

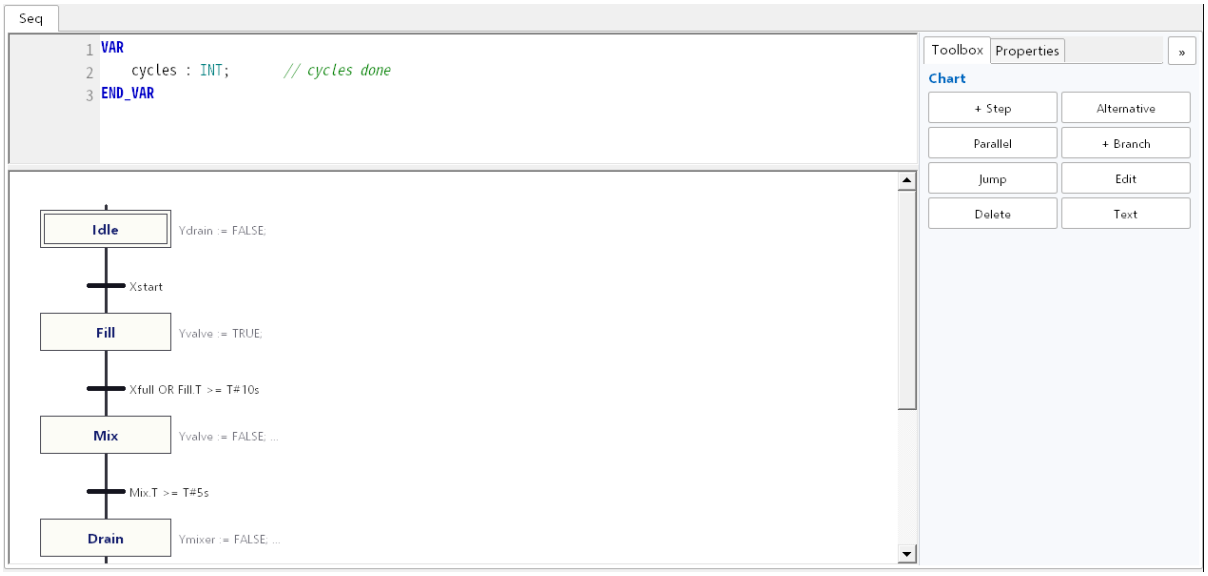


圖 8-12　SFC 編輯器：上方宣告區、中間圖表、右側「圖表」工具

8.4.1 「圖表」工具

按鈕	按鍵	功能
+ 步驟	Ins	在選取處之後加入一個轉移條件與步驟
選擇分支	—	選取步驟後按：在步驟之後分支，由左至右，第一個成立的轉移條件生效
並行分支	—	選取轉移條件後按：所有分支同時執行，由下一個轉移條件匯合
+ 分支	—	在選取的選擇 / 並行分支中再加一條分支
跳躍	—	放在分支或圖表最後一個轉移條件之後：跳回（或跳到）某個步驟
編輯	Enter、雙擊	編輯步驟名稱與動作、轉移條件
刪除	Del	刪除選取項目（步驟連同其轉移條件）
文字	Ctrl+T	以 .sfc 文字檢視、編輯圖表；按「回到圖形」時檢查語法

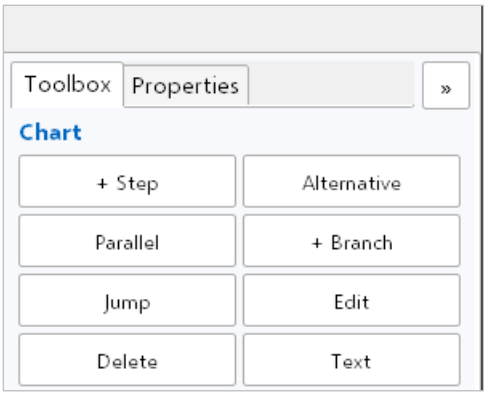


圖 8-13　右側欄的「圖表」工具按鈕

在圖表上按右鍵也有相同的指令。其他操作：

- 方向鍵：↑↓ 選同一序列的前後項目，← → 選相鄰的分支。
- Ctrl+Z / Ctrl+Y：復原 / 重做。
- 模擬或線上時圖表不能修改（先停止模擬或中斷連線）。
- 專案受密碼保護時，SFC 為唯讀（15.6 節）。

8.4.2 文字模式

按 **文字**（Ctrl+T）可直接編輯整個圖表的 .sfc 文字（格式見 8.5.4），適合大量修改或從其他 SFC 複製；按 **回到圖形** 時檢查語法，有錯誤會提示並停在文字模式。

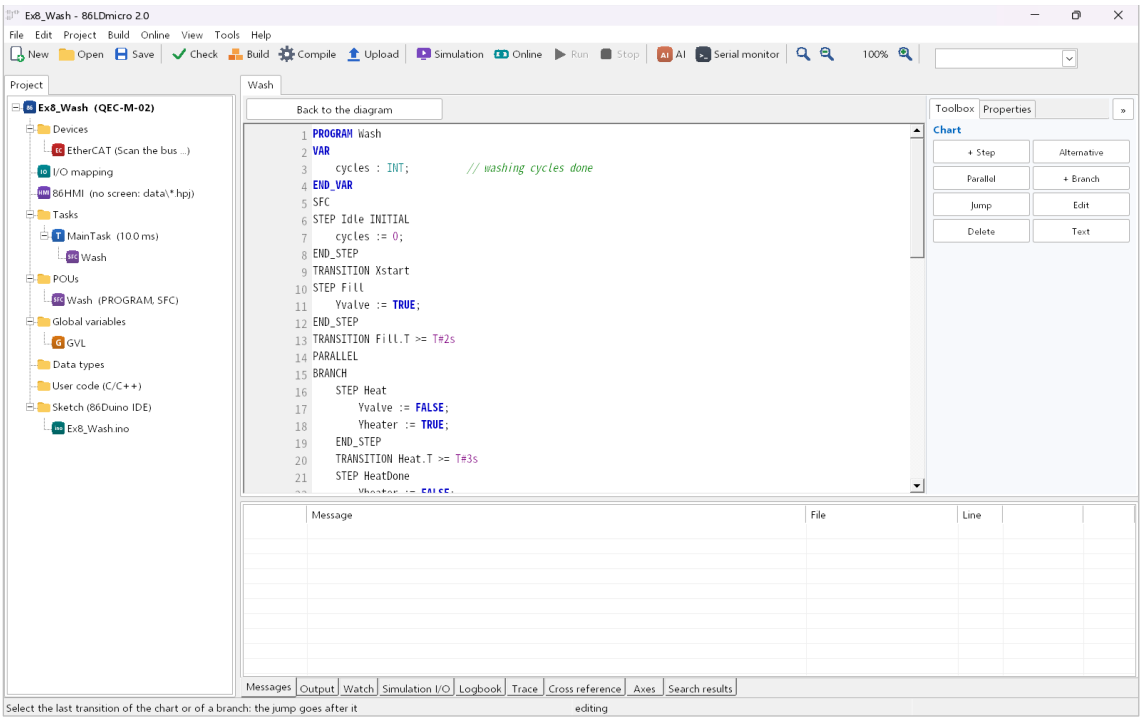


圖 8-14　SFC 的文字模式：整個圖表的 .sfc 文字

8.5 步驟、轉移與分支

8.5.1 步驟

選取步驟按 **編輯**（Enter 或雙擊）開啟「步驟」視窗：

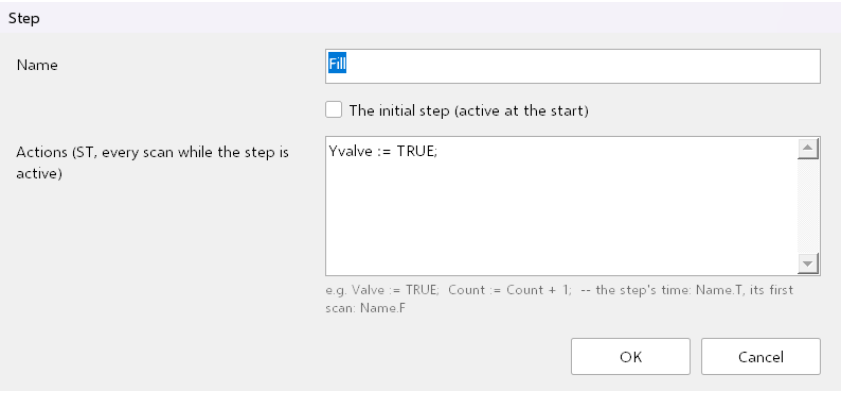


圖 8-15　步驟：名稱、初始步驟、動作

欄位	說明
名稱	英文字母、數字與 _；改名時，指向它的跳躍（JUMP）跟著改
初始步驟（開始時啟動）	PLC 開始執行時啟動的步驟，以雙框表示；一個圖表只能有一個
動作（ST，步驟啟動期間每個掃描都執行）	ST 敘述，例如 `Valve := TRUE;  Count := Count + 1;`；動作的第一行顯示在圖表的步驟右側

8.5.2 轉移條件

選取轉移條件按 **編輯** 開啟「轉移條件」視窗。條件是 ST 運算式：`TRUE`、一個 BOOL 變數，或任何結果為 BOOL 的運算式：

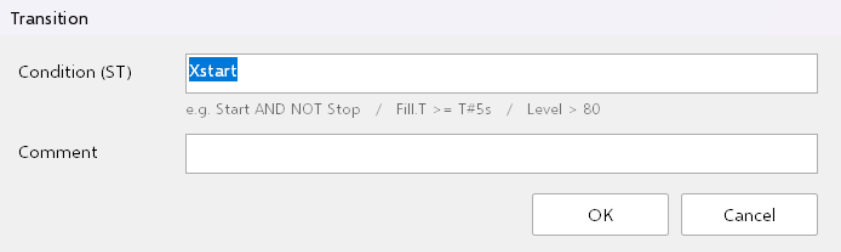


圖 8-16　轉移條件：條件（ST）與註解

條件	意義
`Start AND NOT Stop`	按下啟動且未按停止
`Fill.T >= T#5s`	步驟 Fill 已啟動 5 秒
`Level > 80`	液位超過 80
`mv.Done`	功能塊（例如 MC_MoveAbsolute）完成
`TRUE`	立即轉移（下一個掃描）

8.5.3 分支與跳躍

結構	怎麼建立	怎麼執行
選擇分支 （ALTERNATIVE）	選取步驟 → **選擇分支**；每條分支以轉移條件開頭；**+ 分支** 再加一條	由左至右檢查，**第一個成立**的轉移條件那條分支生效（左邊優先）
並行分支（PARALLEL）	選取轉移條件 → **並行分支**；每條分支以步驟開頭；**+ 分支** 再加一條	所有分支**同時**啟動；各分支最後的步驟都啟動、且下一個轉移條件成立時匯合
跳躍（JUMP）	選取分支或圖表最後一個轉移條件 → **跳躍...**，選要跳到的步驟	轉移條件成立時，跳到指定的步驟（可以往回形成迴圈）

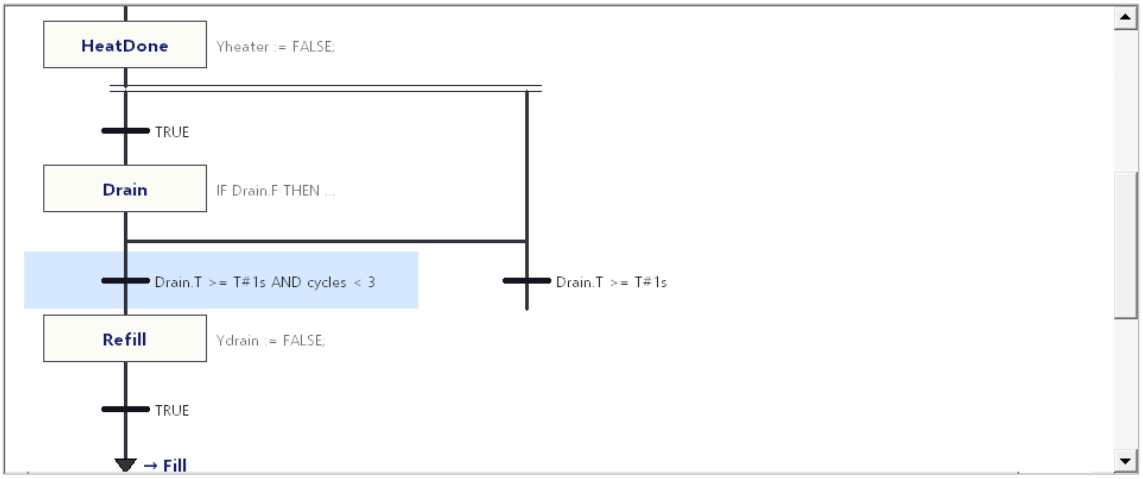


圖 8-17　選擇分支：Drain 之後兩條路徑，左邊回到 Fill、右邊往 Done

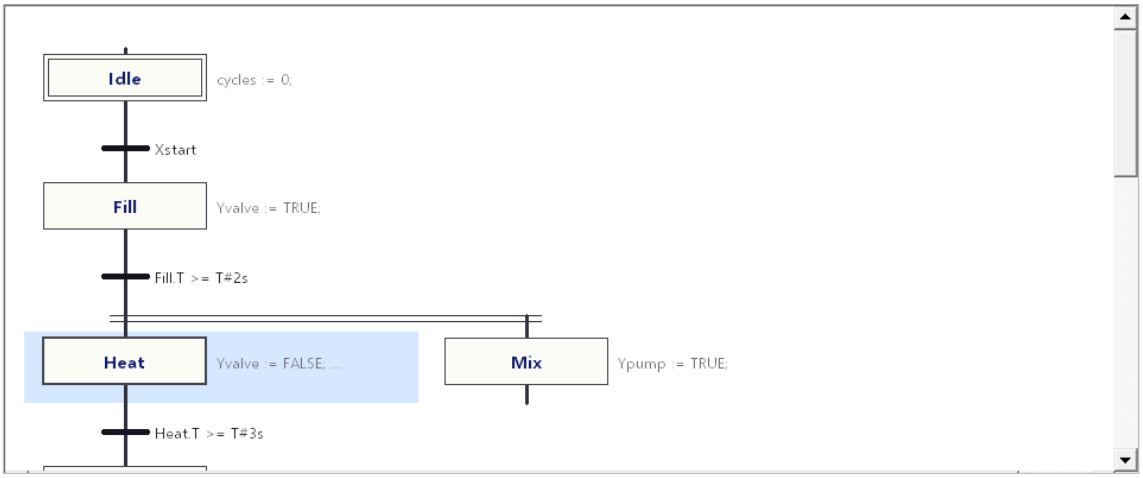


圖 8-18　並行分支：Heat 與 Mix 同時執行，由 TRUE 轉移條件匯合

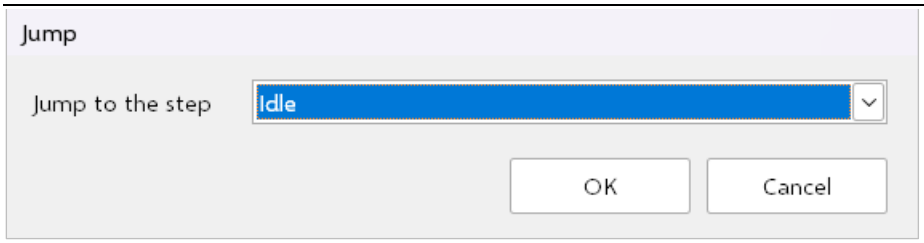


圖 8-19　跳躍：選擇要跳到的步驟

結構不正確時（例如兩個步驟直接相連、並行分支以轉移條件開頭），錯誤列在下方的「訊息」視窗（錯誤內容、檔案與行號），雙擊可跳到該行。

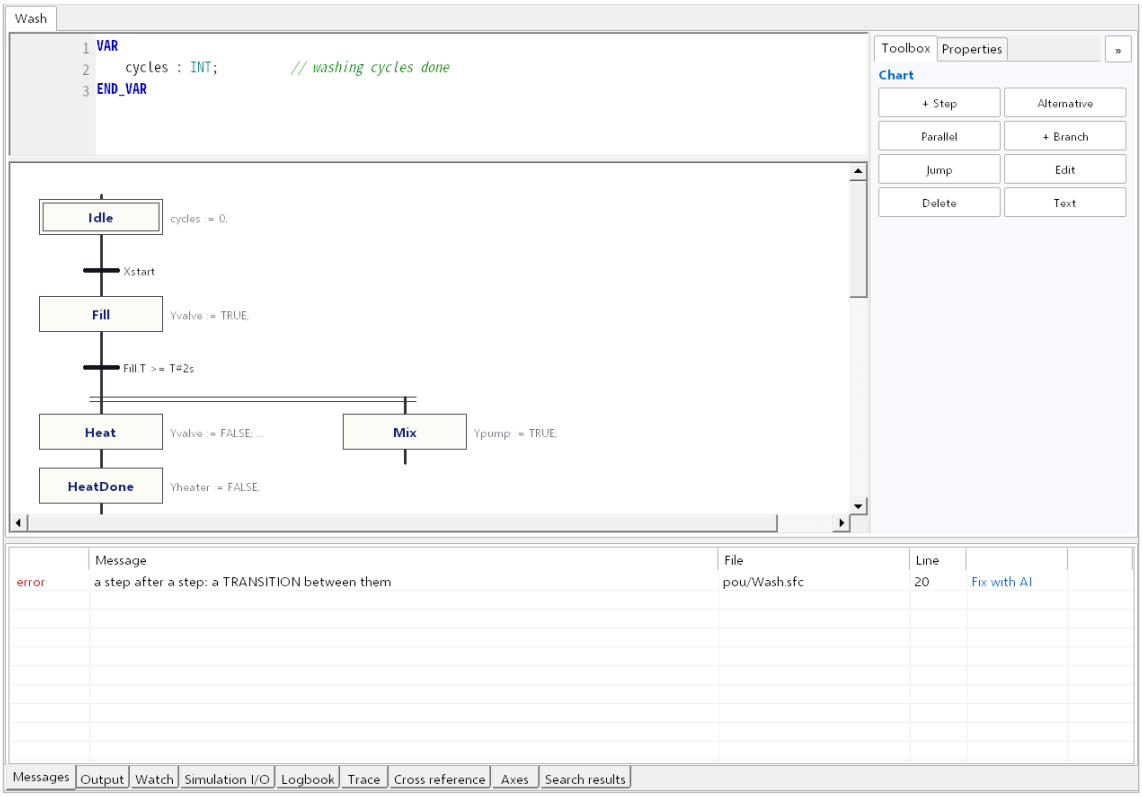


圖 8-20　結構錯誤：下方「訊息」列出錯誤、檔案與行號

8.5.4 範例：清洗流程（含選擇與並行分支）

情境：開關 `Xstart` 打開後：進水 2 秒 → 加熱 3 秒與攪拌同時進行 → 排水 1 秒；重複 3 次後亮完成燈，開關關閉後回到待機。

變數	型別	I/O 對應	用途
Xstart	BOOL	數位輸入	啟動開關
Yvalve	BOOL	數位輸出	進水閥
Yheater	BOOL	數位輸出	加熱器
Ypump	BOOL	數位輸出	攪拌泵
Ydrain	BOOL	數位輸出	排水閥
Ydone	BOOL	數位輸出	完成燈
cycles	INT	—（Wash 的區域變數）	已完成的清洗次數

在 SFC 編輯器按 **文字**（Ctrl+T），貼上以下內容後按 **回到圖形**：

```
PROGRAM Wash
VAR
    cycles : INT;          // 已完成的清洗次數
END_VAR
SFC
STEP Idle INITIAL
    cycles := 0;
END_STEP
TRANSITION Xstart
STEP Fill
    Yvalve := TRUE;
END_STEP
TRANSITION Fill.T >= T#2s
PARALLEL
BRANCH
    STEP Heat
        Yvalve := FALSE;
        Yheater := TRUE;
    END_STEP
    TRANSITION Heat.T >= T#3s
    STEP HeatDone
        Yheater := FALSE;
    END_STEP
BRANCH
    STEP Mix
        Ypump := TRUE;
    END_STEP
END_PARALLEL
TRANSITION TRUE
STEP Drain
    IF Drain.F THEN
        cycles := cycles + 1;
    END_IF;
    Yvalve := FALSE;
    Ypump := FALSE;
    Ydrain := TRUE;
END_STEP
ALTERNATIVE
BRANCH
```

```
    TRANSITION Drain.T >= T#1s AND cycles < 3
    STEP Refill
        Ydrain := FALSE;
    END_STEP
    TRANSITION TRUE
    JUMP Fill
BRANCH
    TRANSITION Drain.T >= T#1s
END_ALTERNATIVE
STEP Done
    Ydrain := FALSE;
    Ydone := TRUE;
END_STEP
TRANSITION NOT Xstart
STEP Reset
    Ydone := FALSE;
END_STEP
TRANSITION TRUE
JUMP Idle
END_SFC
END_PROGRAM
```

文字	對應的圖形
`STEP 名稱 [INITIAL] … END_STEP`	步驟與其動作；INITIAL 為初始步驟
`TRANSITION 條件`	轉移條件
`ALTERNATIVE BRANCH … BRANCH … END_ALTERNATIVE`	選擇分支（每條分支以 TRANSITION 開頭）
`PARALLEL BRANCH … BRANCH … END_PARALLEL`	並行分支（每條分支以 STEP 開頭）
`JUMP 步驟`	跳躍
`SFC … END_SFC`	圖表的開始與結束（在 END_PROGRAM 之前）

#	操作
1	「專案 > 新增 PROGRAM（SFC，順序流程圖）...」，名稱 `Wash`。
2	在 GVL 宣告 Xstart、Yvalve、Yheater、Ypump、Ydrain、Ydone（BOOL）。
3	按 **文字**（Ctrl+T），以上面的文字取代全部內容，按 **回到圖形**。
4	按 **檢查**（Shift+F7）確認沒有錯誤。
5	按 **模擬**（F5），在監看視窗把 Xstart 寫成 TRUE，觀察圖表與輸出。

預期結果：Fill 啟動，Yvalve ON；2 秒後 Heat 與 Mix 同時啟動（Heat 關閉 Yvalve；Yheater、Ypump ON）；3 秒後 Heat 結束進入 HeatDone（Yheater OFF），並行分支匯合到 Drain（Ypump OFF，Ydrain ON，cycles 加 1）；1 秒後 cycles 小於 3 走左邊分支回到 Fill，第 3 次走右邊分支到 Done（Ydone ON）。把 Xstart 寫成 FALSE 後回到 Idle。整個過程約 18 秒。

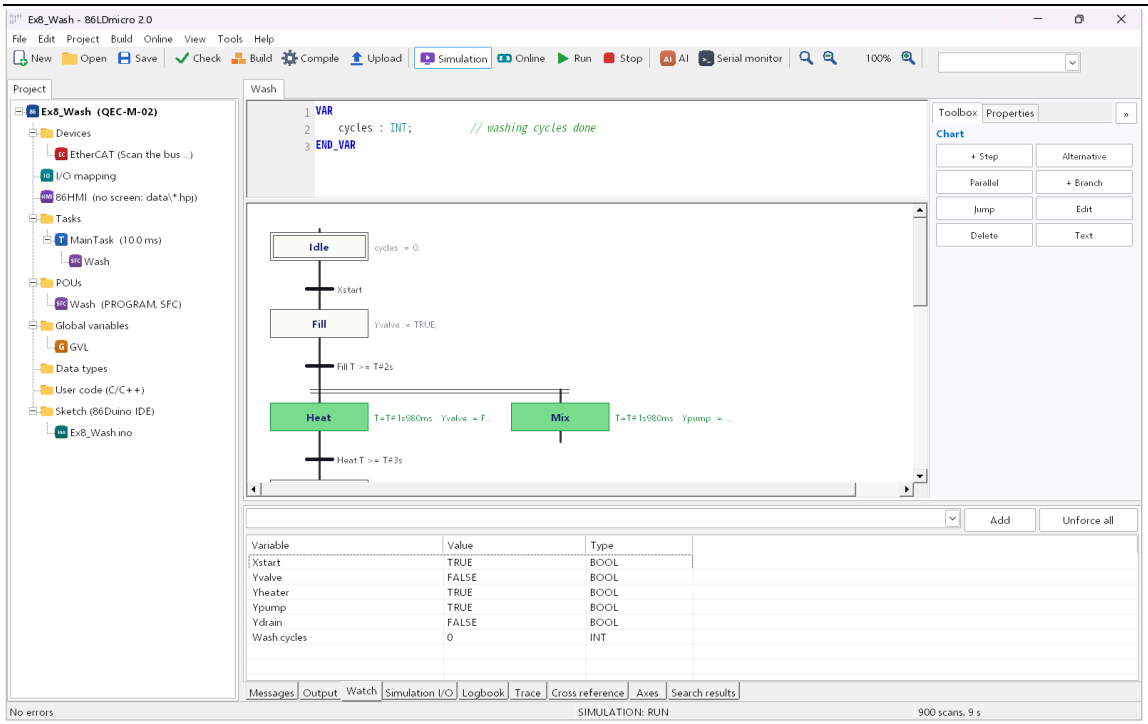


圖 8-21　模擬中的 Wash：啟動中的步驟為綠色，右側顯示已啟動時間

8.6 SFC 執行語意

了解 SFC 在每個掃描中做了什麼，才能寫出可預期的流程。

8.6.1 每個掃描的順序

1. 以「**掃描開始時**」的步驟狀態，計算所有啟動中步驟之後的轉移條件。
2. 觸發成立的轉移條件：離開前一個步驟、啟動下一個步驟（同一步驟後的選擇分支，**左邊優先**）。
3. 執行所有啟動中步驟的動作。

因此一個掃描最多只前進一步；`TRUE` 轉移條件會在下一個掃描才通過。

8.6.2 步驟旗標

寫法	型別	意義	用在
步驟.X	BOOL	步驟啟動中	同一個 SFC 的條件與動作，例如 `Fill.X`
步驟.T	TIME	步驟已啟動的時間（離開後歸零）	逾時、延時轉移，例如 `Fill.T >= T#5s`
步驟.F	BOOL	步驟啟動後的**第一個掃描**為 TRUE	只做一次的動作，例如計數 `IF Drain.F THEN … END_IF;`
程式.步驟_X	BOOL	從**其他 POU** 讀取步驟是否啟動	例如 `Wash.Done_X`；監看視窗也用這個寫法

8.6.3 輸出會保持

動作中設定的變數會保持到被其他程式碼改變（類似 CODESYS 一般動作的寫法）。步驟離開時，86LDmicro 不會自動把它設回 FALSE。需要關閉時，在後續步驟寫 := FALSE（例如上例 Heat 步驟一開始關閉 Yvalve、Drain 步驟關閉 Ypump；若沒有關閉，進水閥會在加熱、攪拌期間一直開著）。

> ***提示：**功能塊（例如 MC_MoveAbsolute）放在步驟的動作中時，步驟離開後就不再呼叫；請在下一個步驟把它的 Execute 設回 FALSE，下次回到該步驟時才有新的上升緣。需要一直呼叫的功能塊（例如 MC_Power）建議放在另一個 ST 或 LD 程式中常駐呼叫（12.7 節）。

8.6.4 模擬與線上的顯示

模擬（F5）或線上（Ctrl+F5）時，啟動中的步驟以**綠色**表示，右側顯示已啟動時間；監看視窗可加入 程式.步驟_X。



圖 8-22 模擬中：啟動中的步驟為綠色，右側顯示已啟動時間

第 9 章　模擬與除錯

86LDmicro 內建模擬器，不接控制器就能在電腦上執行整個專案、操作輸入、觀察輸出。本章說明模擬、模擬 I/O、監看與強制、趨勢圖、交互參照與搜尋，以及常見的錯誤訊息。監看、強制與趨勢圖在**線上**（連線到控制器，10.4 節）時的操作方式完全相同。

9.1　模擬

按工具列的 **模擬**（F5），或「線上 > 模擬」開始模擬；再按一次 F5 結束。模擬會執行整個專案：所有工作、I/O 處理映像、計時器、序列埠、EEPROM、軸與 Motion86 機台，與控制器上的執行方式相同。

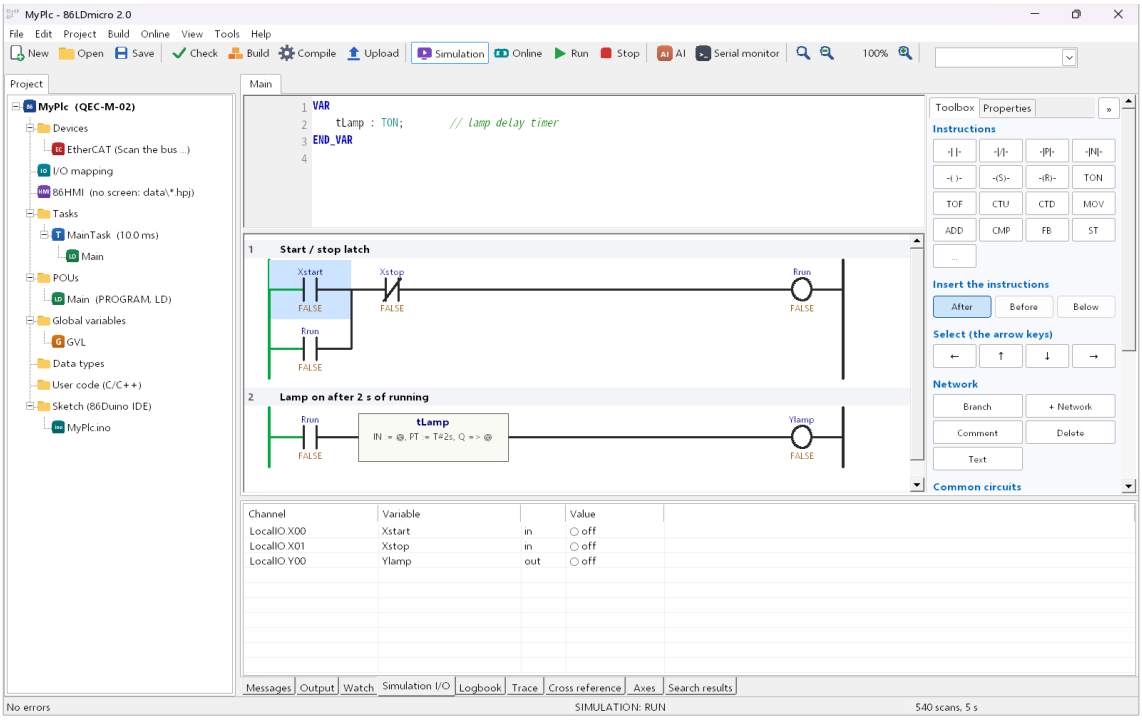


圖 9-1　工具列的「模擬」按鈕（按下狀態）與狀態列的「模擬：RUN」

9.1.1 模擬中的畫面

編輯器	模擬中顯示	操作
階梯圖（LD）	綠色線段為能流；元件下方顯示變數值，計時器方塊顯示經過時間	雙擊接點或線圈，切換該變數的 TRUE / FALSE
ST	滑鼠停在變數上，顯示型別與目前數值	在變數上按右鍵：寫入、強制、加入監看
SFC	啟動中的步驟為綠色，右側顯示已啟動時間	—
機台頁、軸頁、軸（試車）	軸的位置、速度與狀態	激磁、寸動、移動、原點復歸（12.5、12.6 節）
86HMI 模擬視窗	sketch 有 86HMI 畫面時自動開啟	在電腦上直接按畫面上的元件（13.5 節）

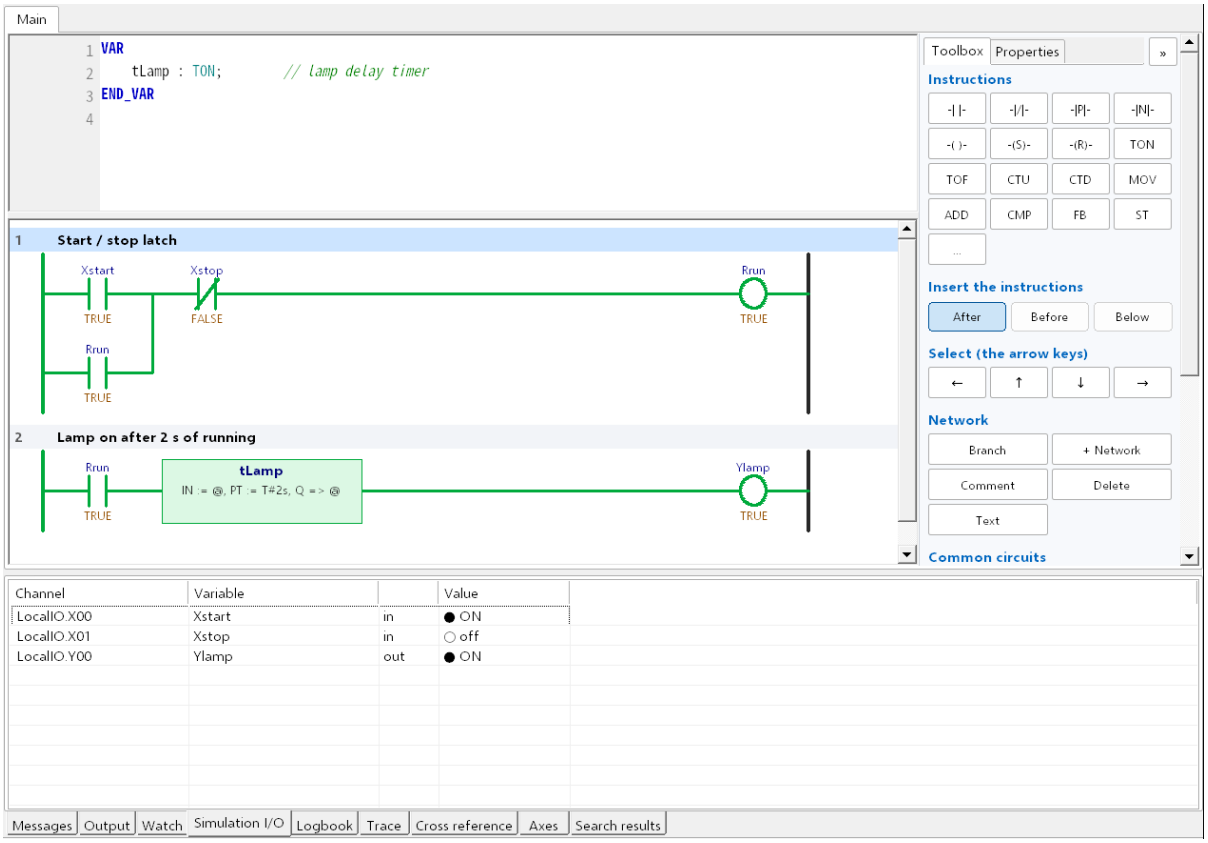


圖 9-2　模擬中的階梯圖：綠色為能流，計時器方塊顯示經過時間

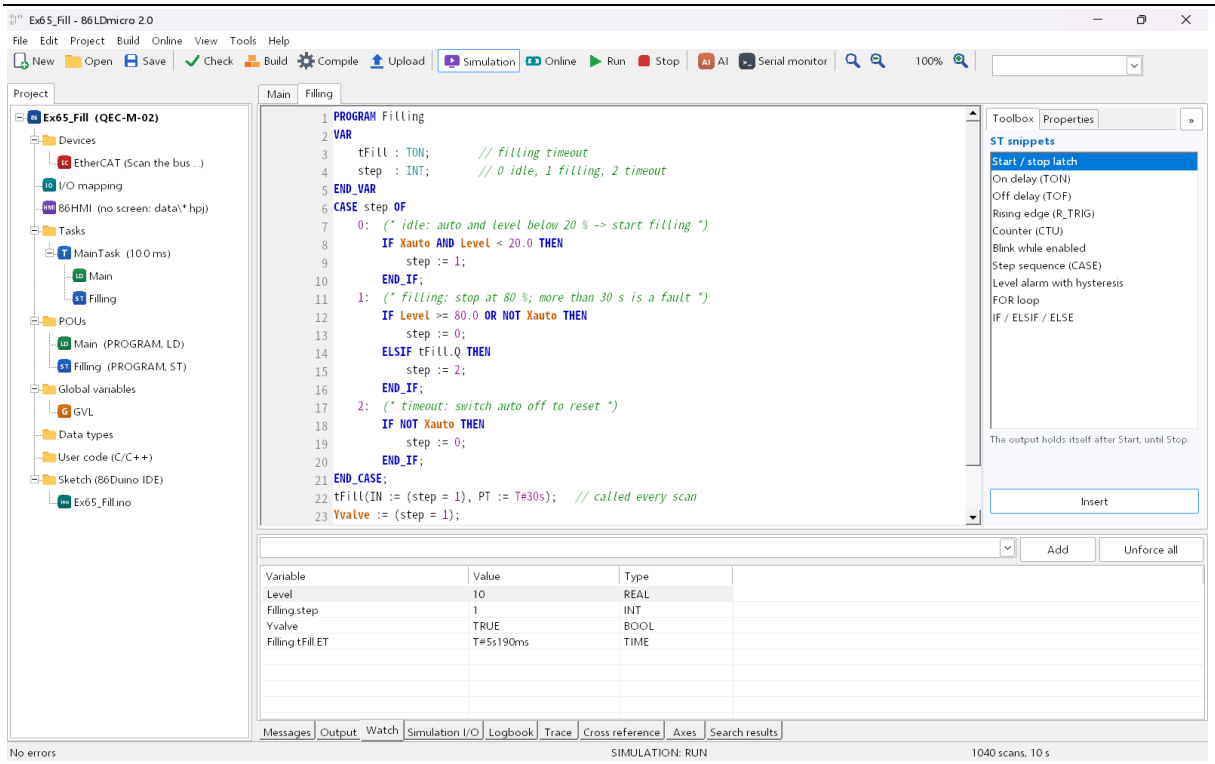


圖 9-3　模擬中的 ST 程式與監看視窗

***注意：**模擬中不能修改程式。要修改時先按 F5 結束模擬，改完再按 F5 重新開始（模擬會從初值開始）。

9.1.2 模擬速度與 RUN / STOP

「線上」選單中與模擬有關的指令：

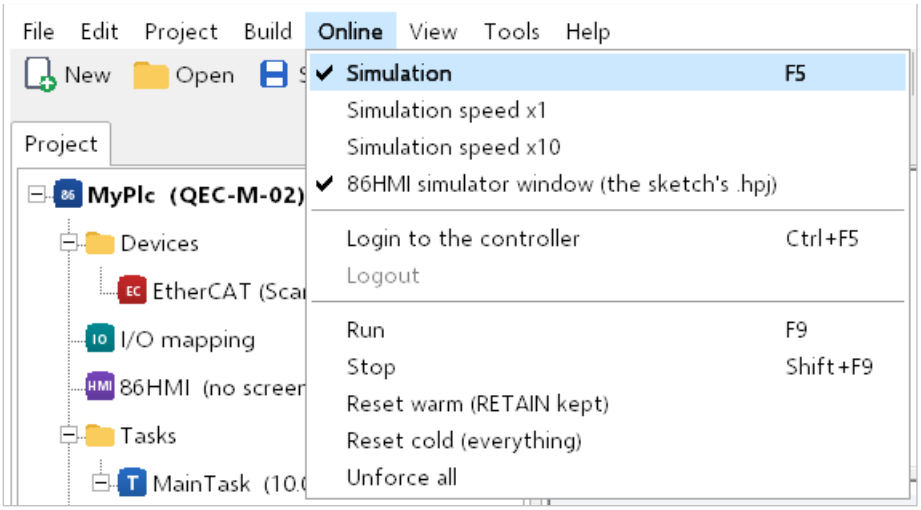


圖 9-4 「線上」選單：模擬、模擬速度、RUN / STOP、暖重啟 / 冷重啟

項目	快捷鍵	用途
模擬	F5	開始 / 結束模擬
模擬速度 x1	—	以實際時間執行（預設）
模擬速度 x10	—	時間加快 10 倍：計時器、軸移動都加快，適合測試長時間的流程
86HMI 模擬視窗（sketch 的 .hpj）	—	開啟 / 關閉 86HMI 模擬視窗（13.5 節）
執行 RUN	F9	程式繼續執行
停止 STOP	Shift+F9	程式停止執行（狀態列顯示「模擬：STOP」）
暖重啟（保留 RETAIN）	—	變數回到初值，RETAIN 變數保留
冷重啟（全部）	—	所有變數（含 RETAIN）回到初值
解除全部強制	—	解除所有強制（9.3 節）

*提示：模擬中的程式執行錯誤（例如陣列索引超出範圍）會讓模擬停止（STOP），原因記錄在「事件紀錄」，例如 `STOP: Main line 12: …`。修正程式後重新模擬。

9.2 模擬 I/O

下方工具視窗的「**模擬 I/O**」分頁列出每個 I/O 對應的實體點（4.6 節），包括本機 I/O 與 EtherCAT 的 PDO 項目：通道、變數、方向與目前的數值。

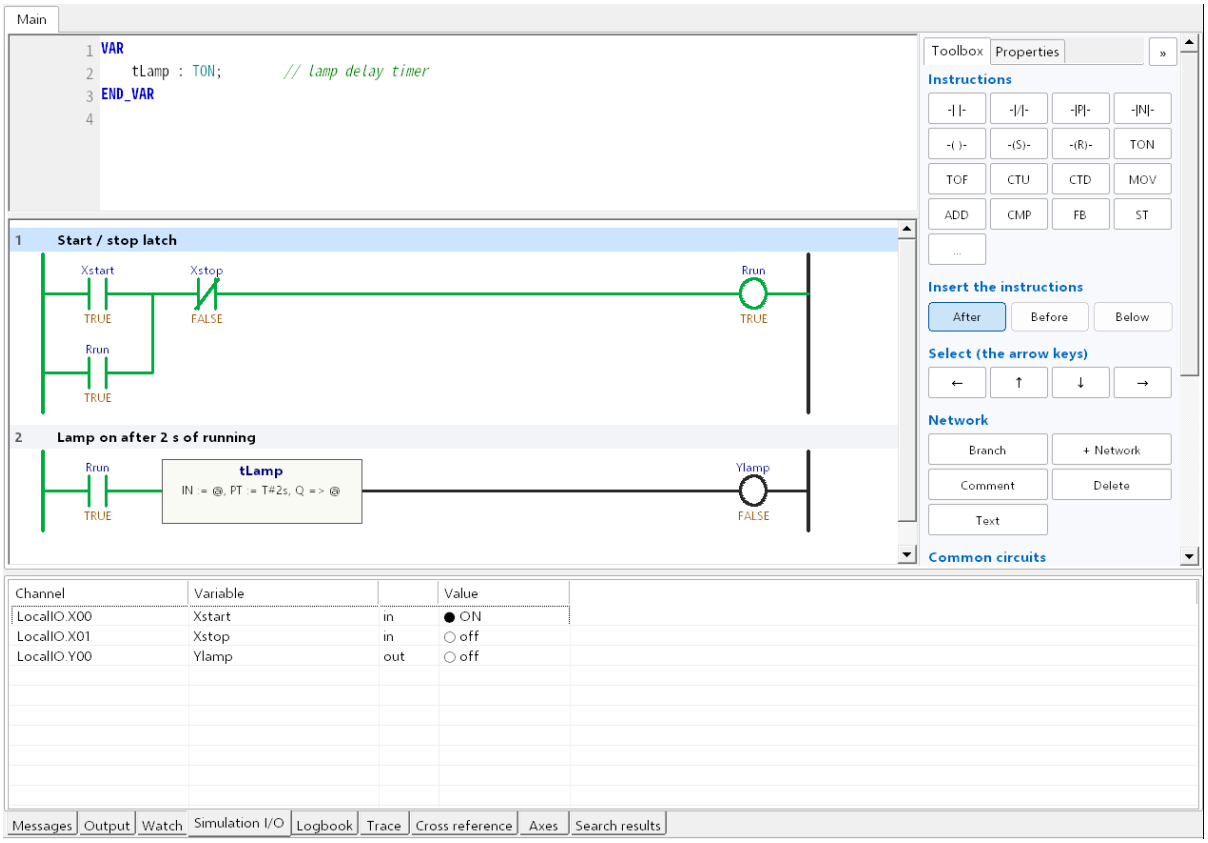


圖 9-5　模擬 I/O：LocalIO.X00（Xstart）等實體點

實體點	操作
數位輸入	雙擊切換 ON / OFF，就像按下或放開實體按鈕
類比輸入	雙擊後輸入數值
輸出（數位、類比）	只顯示，由程式決定

I/O 對應中的反相、濾波與換算，在模擬中的處理與控制器相同。

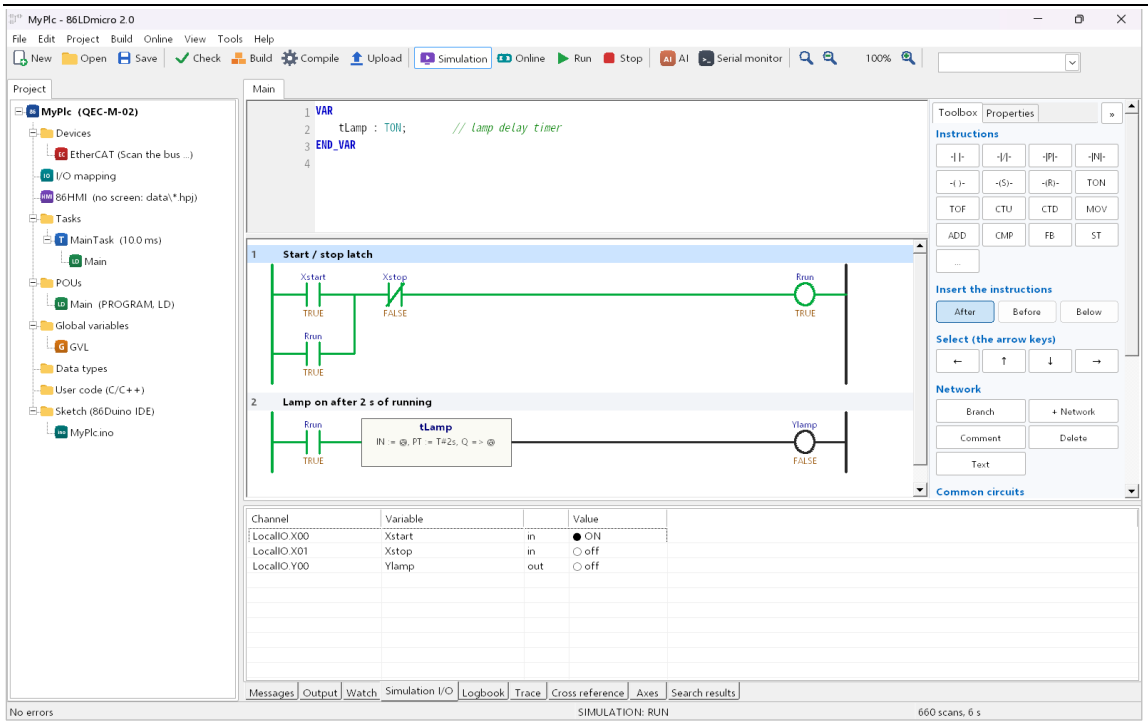


圖 9-6　雙擊 LocalIO.X00 後：Xstart 為 ON，Rrun 自保持

> ***注意**：已對應實體輸入的變數（例如接到 `LocalIO.X00` 的 `Xstart`）在模擬時由「模擬 I/O」決定；在監看視窗直接寫入它，下一個掃描就會被輸入值覆蓋。請在「模擬 I/O」切換，或使用強制（9.3 節）。

9.3 監看、寫入與強制

「**監看**」視窗列出你選的變數與其即時數值，並可寫入或強制。模擬或線上時才會顯示數值、才能寫入。

9.3.1 加入變數

- 在監看視窗上方的輸入框輸入變數名稱（下拉列出全部變數），按 **加入**。可輸入 程式.變數（例如 `Wash.cycles`）、實例.成員（例如 `Panel.bl.Out`）、陣列元素或 SFC 的 程式.步驟_X。
- 或在程式中的變數上按右鍵 >「將 <變數> 加入監看視窗」。
- 選取一列按 Del，或右鍵「從監看移除」，移除該變數。

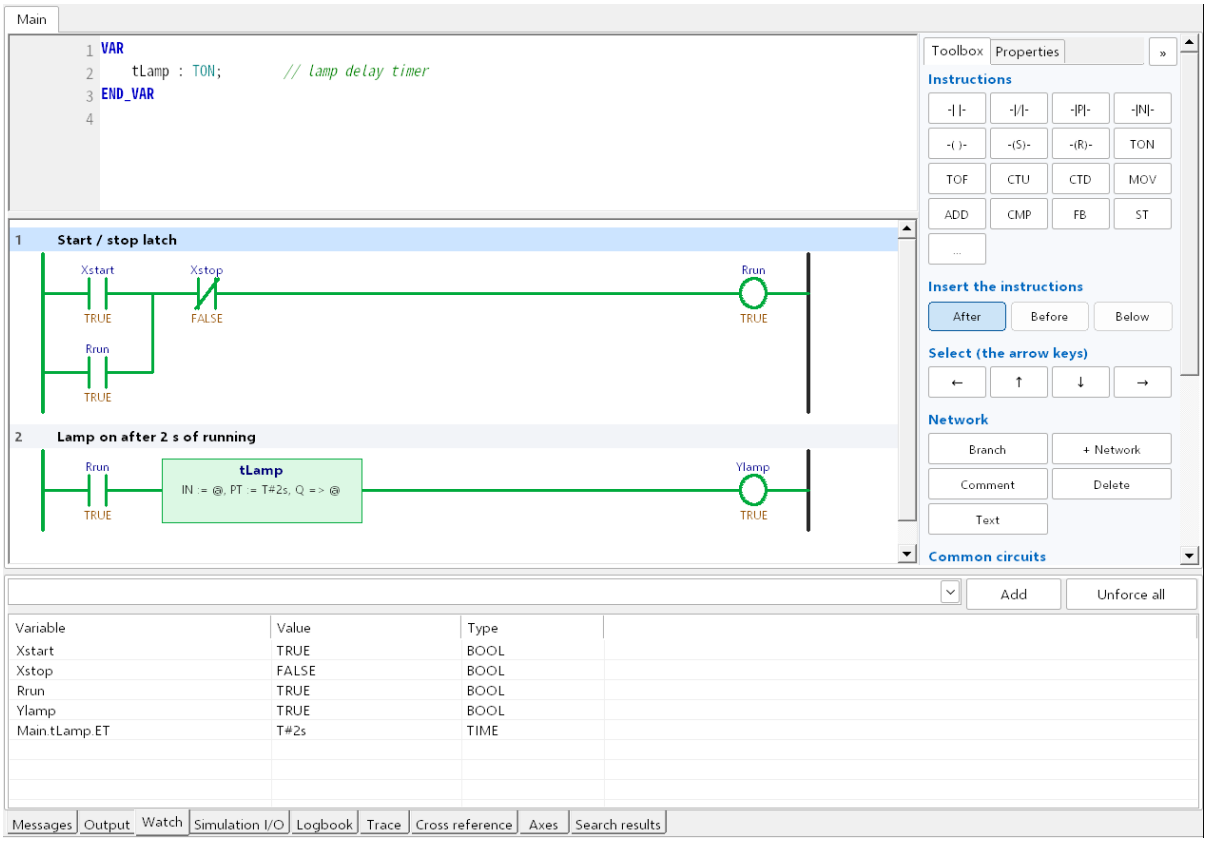


圖 9-7　監看視窗：加入變數後即時顯示數值

9.3.2 寫入與強制

功能	怎麼做	效果
寫入	雙擊監看的一列：BOOL 直接切換 TRUE / FALSE，其他型別輸入數值；或右鍵「寫入 <變數>...」	只寫入**一次**，程式之後仍可改變它
強制	右鍵「強制 <變數>...」，輸入數值	每個掃描都寫入同一個值，數值欄後面顯示 [F]
解除強制	右鍵「解除強制」	解除這個變數的強制
解除全部強制	監看視窗右上角 **解除全部強制** 按鈕，或「線上 > 解除全部強制」	解除所有變數的強制

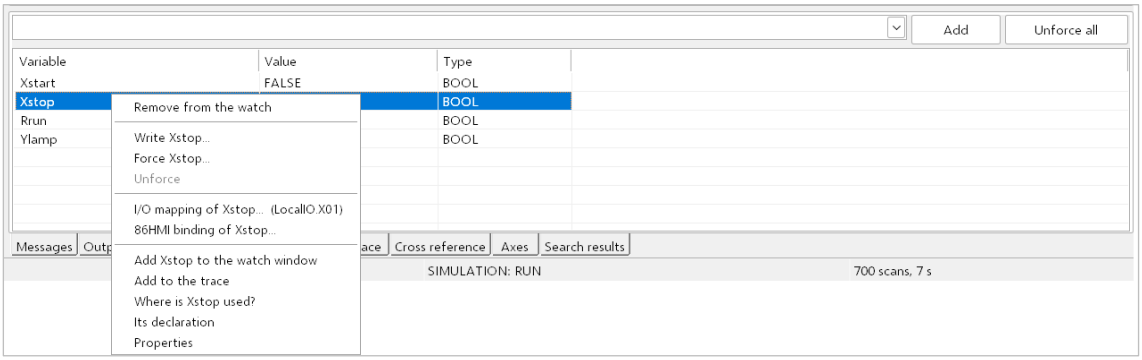


圖 9-8　監看視窗的右鍵選單：寫入、強制、解除強制、加入趨勢圖等



圖 9-9　強制 Xstop：輸入要保持的數值

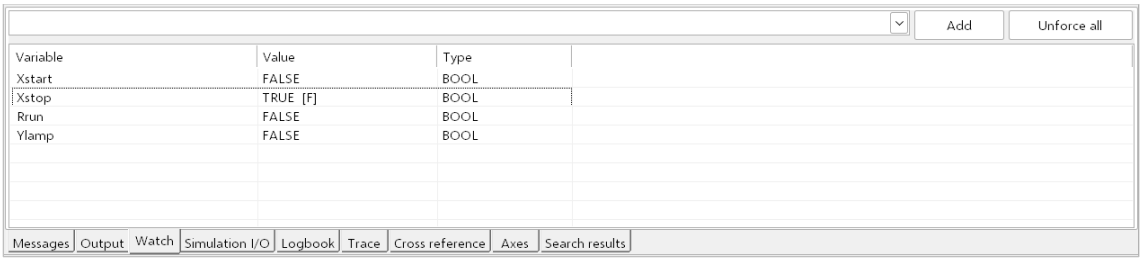


圖 9-10　強制中的變數在數值後面顯示 [F]

9.3.3 強制在什麼時候寫入

強制值在每個掃描寫入**兩次**，與 CODESYS 相同：

1. 讀入輸入之後、程式執行之前：程式讀到的是強制值。
2. 程式執行之後、寫出輸出之前：寫到實體輸出的是強制值。

如果某個程式在掃描中自己寫這個變數，同一個掃描中**排在它後面**的程式讀到的是程式寫的值，而不是強制值；到了掃描結束才又被強制值蓋回。所以：

- 強制**由 I/O 對應寫入的輸入變數**（實體輸入）與**輸出變數**最有效。
- 被程式每個掃描重寫的中間變數，強制後在監看中看到的是強制值，但其他程式不一定讀得到。
- 強制了卻「沒有作用」或「一下就回復」時，用交互參照（9.5 節）查這個變數被哪些程式寫入。

*提示：例：`tools\example\ex30_dispense` 的展示模式（`gDemo = TRUE`）由第一個程式 PlantSim 重寫 `sim*`，IoMap 再重寫 `x*`，強制這些變數會被覆寫（一個流程後就回復）。要用強制測試流程，請先關閉展示模式（`gDemo = FALSE`，也可以強制它），再強制實體輸入 `diEntry`、`diDoorClosed` 等。

*安全：強制會讓輸出不受程式控制，在實機上請確認現場安全再操作。與控制器的連線中斷 3 秒時，控制器會自動解除所有強制。

9.4 趨勢圖

「**趨勢圖**」視窗把變數的變化畫成曲線，適合觀察計時、類比值與軸位置。

1. 在監看視窗（或程式中）的變數上按右鍵 >「**加入趨勢圖**」。最多 8 個變數。
2. 「趨勢圖」分頁自動切到前面：每 100 ms 取樣一次，顯示最近一分鐘。
3. 在趨勢圖上按右鍵：**暫停**、**清除取樣**、**移除全部變數**、**以 CSV 複製**（貼到 Excel 分析）。

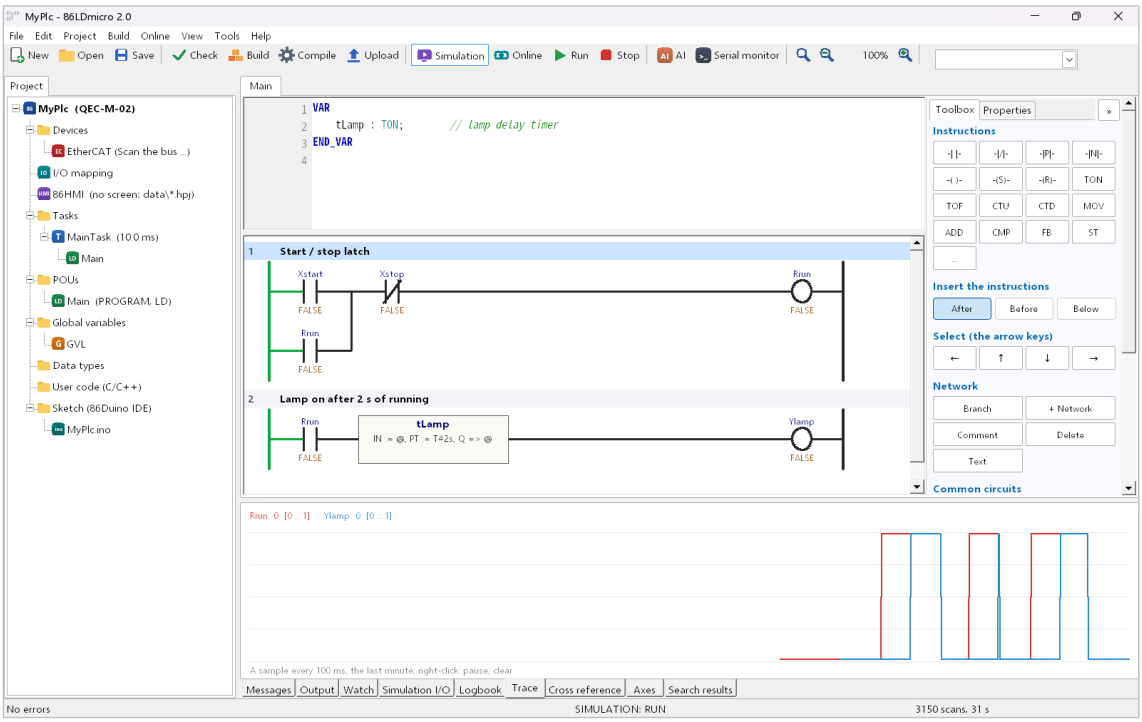


圖 9-11　趨勢圖：Rrun 與 Ylamp 的變化

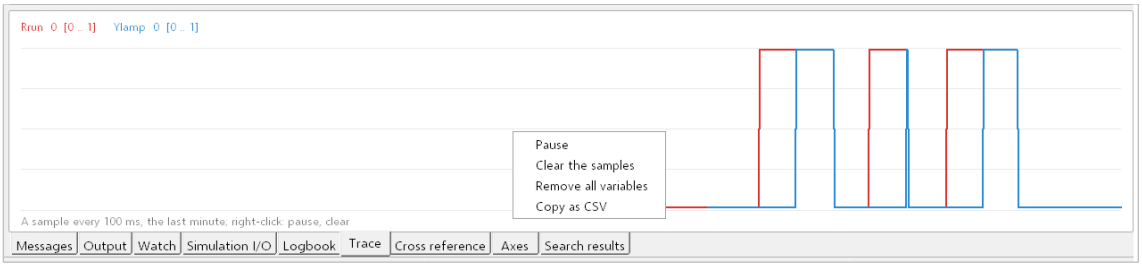


圖 9-12　趨勢圖的右鍵選單

9.5 交互參照與搜尋

專案變大後，常需要找「這個變數用在哪裡」或「這段文字在哪個程式」。86LDmicro 提供三種尋找方式：

功能	快捷鍵	搜尋什麼	適合
在此分頁尋找	Ctrl+F（F3 / Shift+F3 下一個 / 上一個）	目前分頁的文字或階梯圖元件	在一個程式裡逐一找
在專案中尋找變數（交互參照）	Ctrl+Shift+F	一個變數在各處的讀取、寫入、宣告	確認變數被誰寫入
在專案中搜尋	Ctrl+Shift+H	整個專案的任意文字（含註解、.ino）	找註解、字串、函式或不確定名稱的東西

9.5.1 交互參照（Ctrl+Shift+F）

1. 按 **Ctrl+Shift+F**（「編輯 > 在專案中尋找變數...」），在「尋找變數」視窗輸入名稱，按 **確定**。也可以在變數上按右鍵 >「<變數> 用在哪裡？」。
2. 下方「**交互參照**」分頁列出每一處：檔案、行、用途（**宣告**、**讀取**、**寫入**）與該行內容；狀態列顯示共幾處。
3. 雙擊一列，開啟該檔案並跳到那一行。

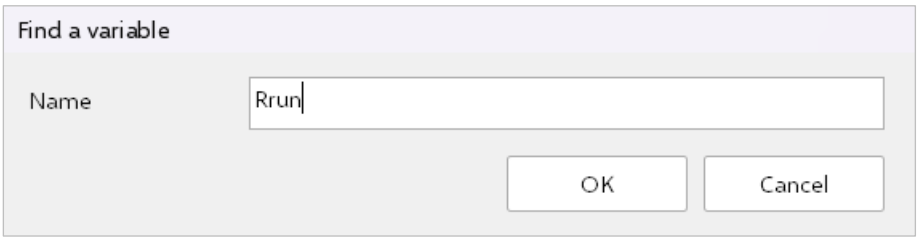


圖 9-13 「尋找變數」視窗

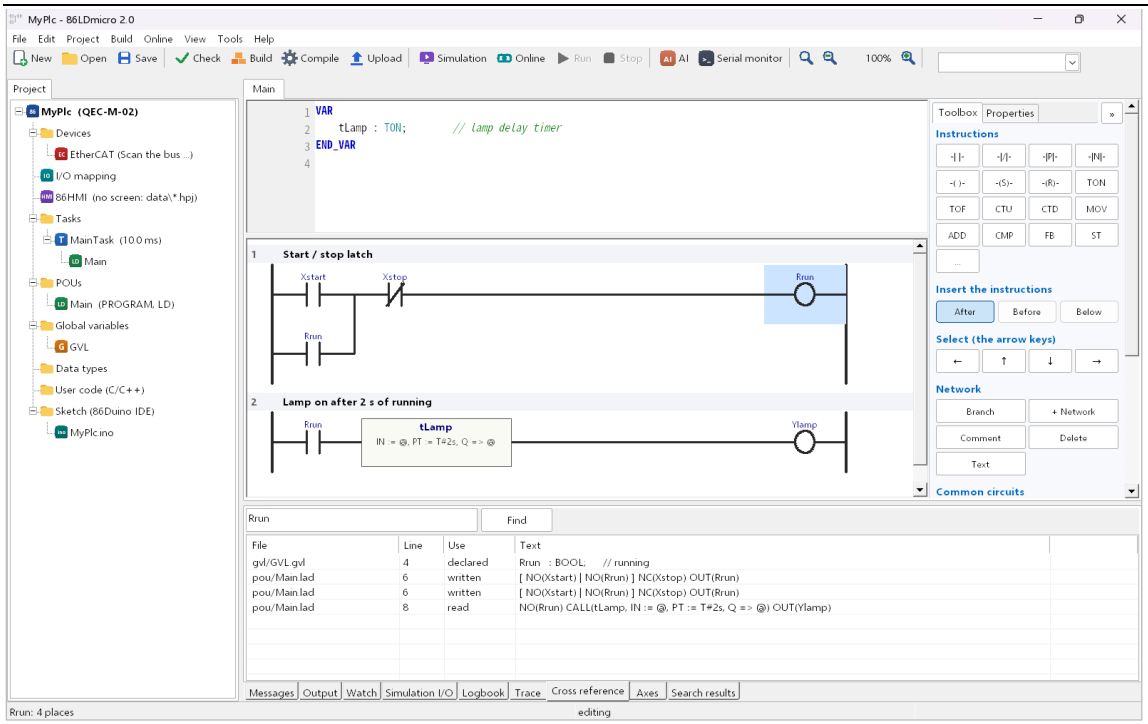


圖 9-14　交互參照：Rrun 的宣告、讀取與寫入位置

***提示：**「用途」為「寫入」的位置是指 := 的左邊，或階梯圖的 OUT、SET、RST、MOV 等。一個變數有兩個以上的程式寫入時，最後執行的程式決定掃描結束時的值。

9.5.2 在專案中搜尋（Ctrl+Shift+H）

搜尋所有 POU（階梯圖的網路標題、註解與元件，ST、SFC）、全域變數表、資料型別、使用者 C/C++，以及 sketch 的 .ino。

1. 按 **Ctrl+Shift+H**（「編輯 > 在專案中搜尋...」）。下方工具視窗切到「**搜尋結果**」。先在程式中選取一段文字再按，會直接搜尋那段文字。
2. 輸入要找的文字，按 **Enter** 或 **尋找**。需要時勾選「**大小寫相符**」或「**全字相符**」（例如找 `Pump` 時不列出 `PumpSpeed`）。
3. 結果每列一處：位置（POU 名稱與語言）、行號與該行內容；右上角顯示共幾處。
4. **雙擊**一列（或選取後按 Enter），開啟該檔案並跳到那一行：ST 與 C/C++ 選取該行，階梯圖捲到並選取那個網路。

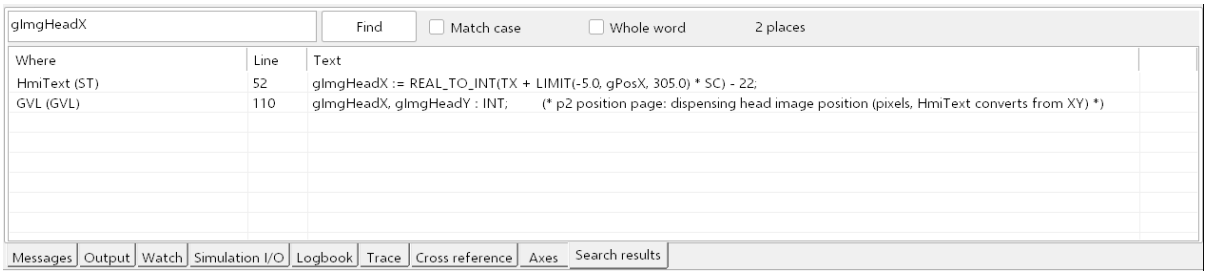


圖 9-15 「搜尋結果」：搜尋 gImgHeadX，找到 2 處（ST 程式與全域變數表）

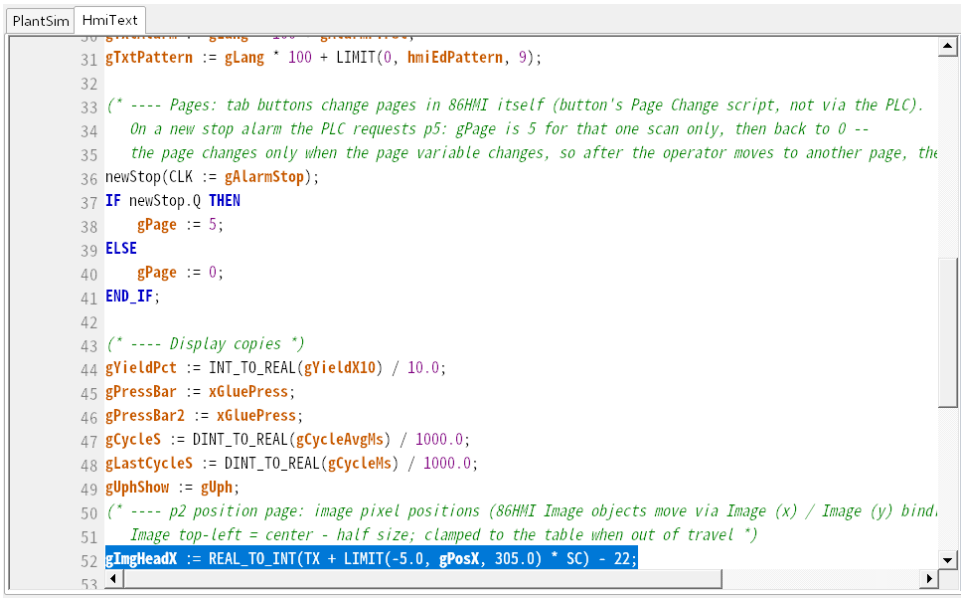


圖 9-16 雙擊第一列：開啟 HmiText 並選取第 51 行

> ***提示**：搜尋只會找、不會取代。要改變數名稱，請從結果逐一開啟修改，改完按 **檢查**（Shift+F7）確認沒有遺漏。命令列也可搜尋：`86ldmicro-cli search <專案> <文字> [--case] [--word]`（15.5 節）。

9.6 常見錯誤訊息

按 **檢查**（Shift+F7）或建置時，「**訊息**」視窗列出錯誤（必須修正）與警告（建議確認）。錯誤的元件在階梯圖中以**紅底**標示，ST 以底線標示。

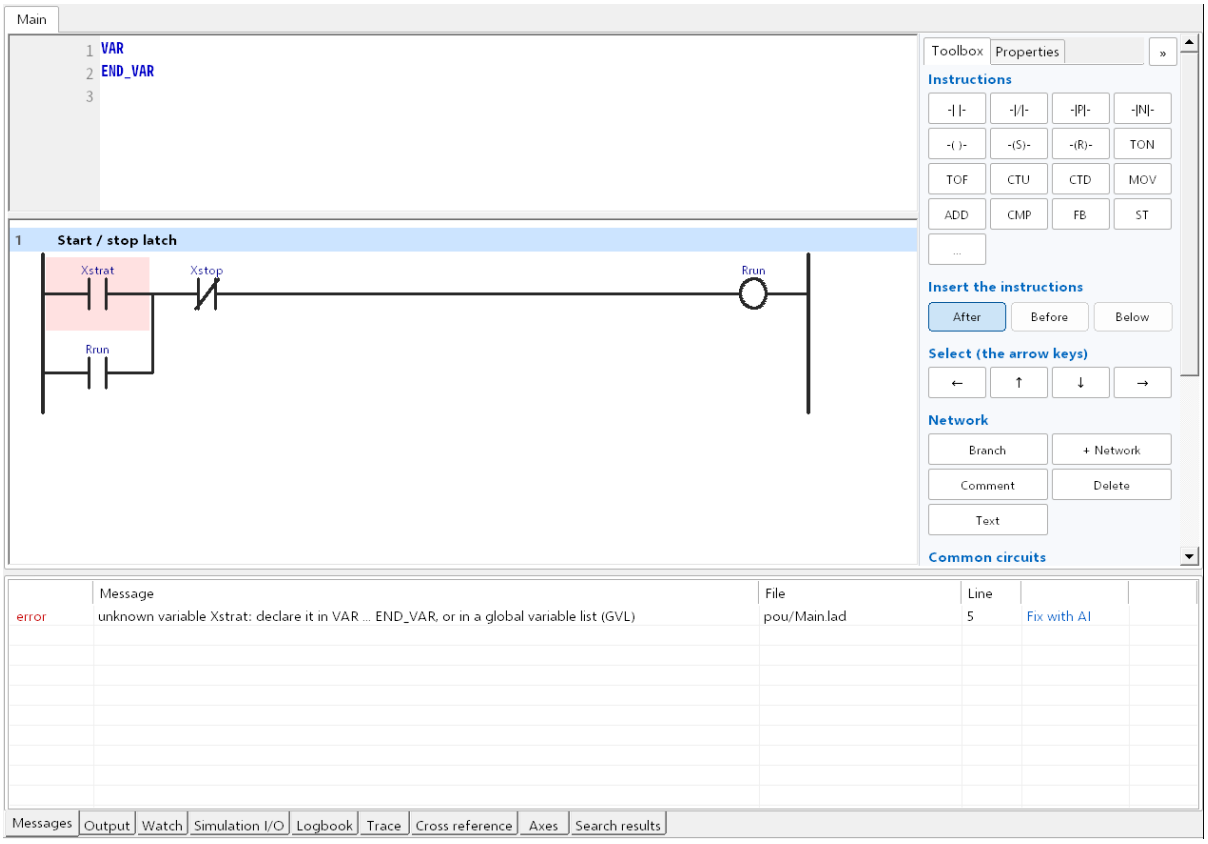


圖 9-17　打錯的變數名稱：元件紅底，建置輸出顯示「建置：專案有錯誤（見「訊息」）」

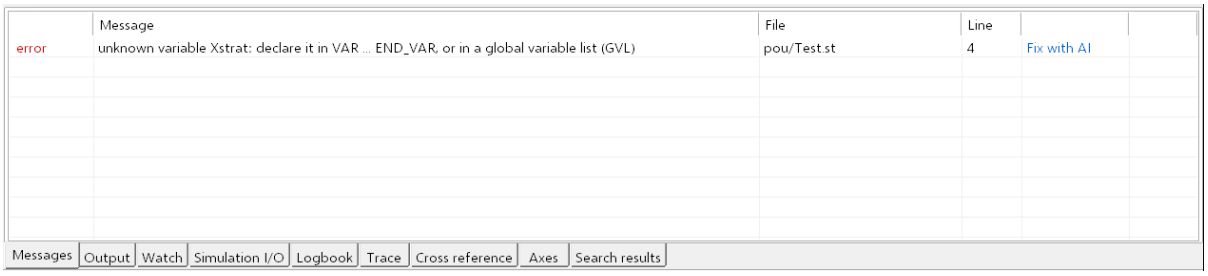


圖 9-18　「訊息」視窗：每列一個錯誤，雙擊跳到該元件

1. 切到下方「訊息」分頁。
2. 雙擊錯誤列，畫面跳到出錯的元件或程式行並選取它。
3. 修改後再按 **檢查**（Shift+F7），直到沒有錯誤。
4. 需要時按該列的 **用 AI 修正**，把錯誤與程式送給 AI 助理（15.3 節）。

9.6.1 編譯訊息對照

下表為常見訊息（引號中的名稱依實際程式而不同）：

訊息（節錄）	原因	處理
未知的變數 Xstrat：請在 VAR ... END_VAR 中宣告，或宣告在全域變數表（GVL）	名稱打錯或未宣告	修正拼字，或在 GVL / VAR 宣告
REAL 不是 BOOL	把數值當接點或條件	寫比較：`x <> 0`、`CMP(x > 10)`
預期一個敘述，但找到 '1'（缺 END_IF）	ST 的 IF / FOR / CASE 沒有結尾	補上 END_IF / END_FOR / END_CASE
缺少 ';'	ST 敘述結尾少分號	在提示位置後補 `;`
預期 ')'，但找到 ';'	括號不成對	檢查括號
未知的功能塊實例 tLamp	CALL 使用的實例未宣告	宣告 `tLamp : TON;`
TON 沒有輸入 PTT（輸入：IN, PT）	功能塊參數名稱錯誤	使用訊息列出的參數名稱
Count 在 GVL 中宣告兩次	變數重複宣告	刪除其中一個
數值 100000 超出 INT 範圍	常數超出型別範圍	改用 DINT，或修正數值
未知的變數 AxisQ	軸名稱錯誤（MC 功能塊）	使用機台頁中的軸名稱（例如 AxisX）
工作 MainTask 執行 Ghost，但它不是專案的 PROGRAM	工作中的 POU 被改名或刪除	到工作設定移除或改選（4.3 節）
步驟後面接步驟（SFC）	兩個步驟之間缺轉移條件	在中間插入轉移條件
警告：Xauto 只在這裡被讀取：沒有 I/O 對應，也沒有程式寫入它 —— 它一直是 FALSE	輸入變數忘了對應	在 I/O 對應頁加入對應（4.6 節），或確認這是預期的
警告：`the program Test is in no task: it does not run`（英文訊息）	新增的 PROGRAM 沒有放進工作	在工作設定加入（4.3 節）

9.6.2 上傳、連線與執行時的訊息

訊息 / 現象	處理
無法開啟 COMx（是否被其他程式使用中，例如 86Duino IDE 的序列埠監控視窗？）	關閉使用該埠的程式後重新上傳（10.1 節）
控制器執行的是其他程式（或此程式的其他版本）	重新上傳（F8）後再連線（10.4 節）
線上 COMx：沒有回應	確認專案設定「線上監看」已啟用；程式不要占用 USB 的 Serial；重新上傳
序列埠監看：`EtherCAT: begin failed`	EtherCAT 線未接到主站的 EtherCAT 埠或從站未送電（11.7 節）
序列埠監看：`EtherCAT: the ENI … cannot be used`	ENI 未上傳到主站或與實際網路不符；重新 Scan、匯出並上傳 ENI（11.7 節）
序列埠監看：`EtherCAT: no slave`	EtherCAT 線未接、從站未送電、接在 OUT 而非 IN
事件紀錄：`MainTask watchdog: STOP, …`	掃描時間超過看門狗設定，PLC 進入 STOP（10.5 節）
事件紀錄：驅動器離開 Operation Enabled	看驅動器警報碼；排除後 MC_Reset 再 MC_Power（12.9 節）

更多問題與處理方式請見第 16 章。

第 10 章　建置、上傳與線上監看

程式在模擬中確認無誤後，就可以建置、編譯並上傳到 QEC 控制器，再以線上監看觀察控制器上的即時狀態。本章依序說明這些指令、建置產生的檔案、序列埠監看、線上監看、事件紀錄與看門狗，以及只上傳 86HMI 資源的方法。

10.1　檢查、建置、編譯與上傳

四個指令在「建置」選單與工具列，由左到右一個比一個多做一步：上傳會先建置、編譯，編譯會先建置。

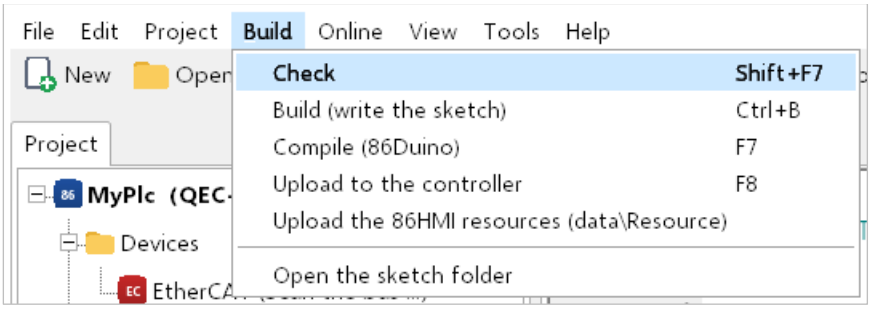


圖 10-1　「建置」選單

項目	快捷鍵	用途	需要
檢查	Shift+F7	只編譯 PLC 程式，在「訊息」列出錯誤與警告；不寫任何檔案	—
建置（寫入 sketch）	Ctrl+B	把專案寫成 sketch 的 `plc86.cpp` 等檔案（10.2 節）；從 IDE 啟動時通知 IDE 重新載入	—
編譯（86Duino）	F7	建置後以 86Duino IDE 的 arduino-builder 編譯；C/C++ 錯誤會指回 .cpp86 的行號	86Duino IDE（2.6 節）
上傳到控制器	F8	編譯後以 v86dude 經 USB 上傳；sketch 有 `data\Resource` 時一併上傳	IDE、USB 線、工具列選好序列埠
上傳 86HMI 資源（data\Resource）	—	只上傳 86HMI 的圖片、字型與畫面，不編譯、不改程式（10.6 節）	IDE、USB 線、序列埠
開啟 sketch 資料夾	—	以檔案總管開啟 sketch 資料夾	—



圖 10-2　工具列：檢查、建置、編譯、上傳與序列埠選擇

過程顯示在下方「輸出」視窗的「**建置輸出**」，錯誤進「**訊息**」（9.6 節）。

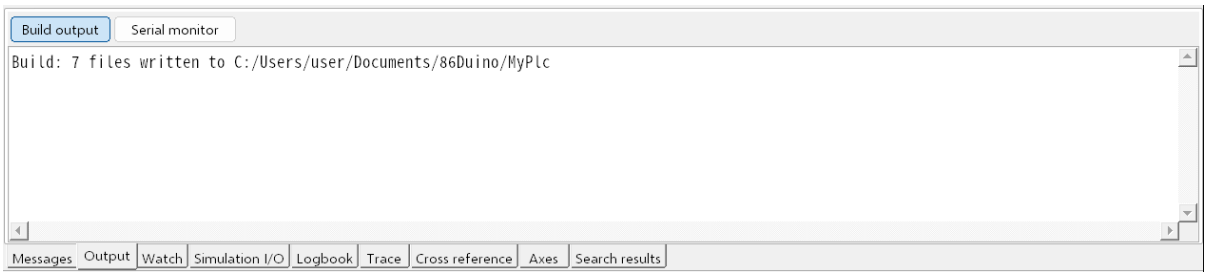


圖 10-3　「建置輸出」：建置、編譯與上傳的過程都顯示在這裡

10.1.1　上傳的步驟

1. 用 USB 線連接電腦與 QEC 控制器，控制器送電。
2. 在工具列右側的序列埠清單選控制器的 COM 埠（例如 COM4；找不到時按「工具 > 重新整理序列埠」，2.4 節）。
3. 按 **上傳**（F8）。「建置輸出」依序出現「建置：已寫入...」、「== 編譯完成...」、「== 上傳到 COM4 ...」與進度，最後「== 上傳完成」。
4. 控制器重新啟動並開始執行程式。若「工具 > 選項」勾選了「上傳後自動重新連線」且原本在線上，會自動重新連線（2.6 節）。

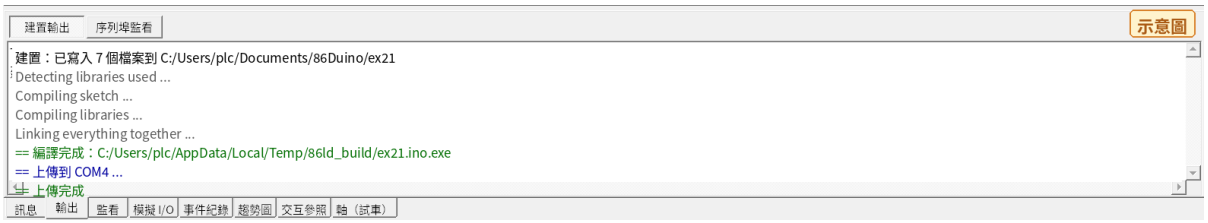


圖 10-4　上傳成功：建置 → 編譯完成 → 上傳到 COM4 → 上傳完成（示意圖）

10.1.2　上傳失敗時

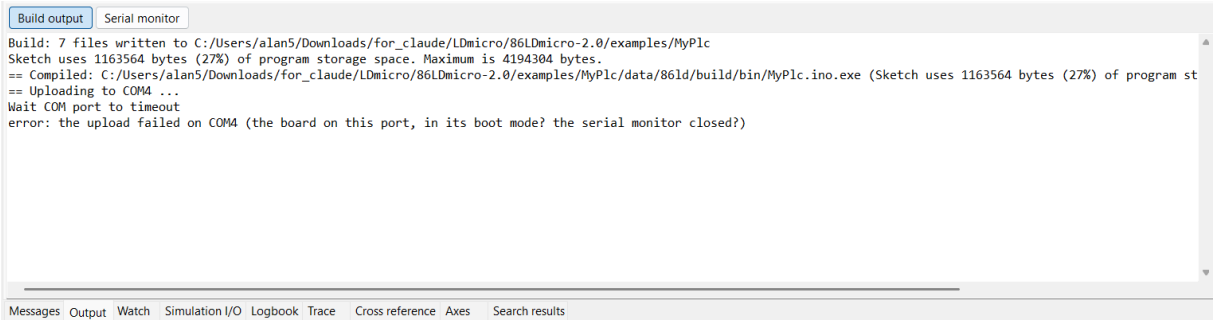


圖 10-5　上傳失敗：無法開啟 COM4（序列埠被其他程式占用）

輸出中的訊息	原因	處理
無法開啟 COMx（是否被其他程式使用中…）	86Duino IDE 的序列埠監控視窗、終端機程式開著同一埠	關閉該程式後再按 F8
找不到 COMx	USB 線拔掉、選錯埠、控制器未送電	確認 COM 埠（2.4 節），重新選擇
找不到 86Duino IDE	單獨啟動 86LDmicro 且未設定 IDE 資料夾	「工具 > 選項」指定 86Duino IDE 資料夾（2.6 節）
編譯錯誤（.cpp 檔名與行號）	C/C++ 使用者程式或函式庫錯誤	雙擊訊息跳到 .cpp86 的行號修正（第 14 章）
建置：專案有錯誤	PLC 程式有錯（紅底元件）	看「訊息」分頁（9.6 節）
上傳到一半逾時	USB 線或 HUB 不穩、控制器重新開機中	直接接電腦的 USB 孔，重新上傳

10.1.3 一條 USB 線的四種用途

QEC 的 USB 線在電腦上是一個序列埠（工具列選的 COM 埠），它同時被下列四件事使用，但**同一時間只能有一個在用**；86LDmicro 會自動讓出與接回：

用途	誰使用	說明
上傳	v86dude（F8）	把程式寫入控制器；上傳時 86LDmicro 自動讓出序列埠（若在線上先離線），完成後自動接回
線上監看	86LDmicro（Ctrl+F5）	專案設定「線上監看」啟用、序列埠為 `Serial` 時，監看、強制、能流都走這條線
Serial.print	你的程式	程式以 `Serial.print()` 印出的文字；線上時夾在監看封包之間，自動顯示在「序列埠監看」
Configurator（USB）	EtherCAT 頁	USB 模式的 EtherCAT Configurator 也走這條線（3 Mbps），上傳或線上時自動暫停（第 11 章）

*__注意__：使用者 C/C++ 程式請用 `Serial1`～`Serial3`（控制器的 COM1～3）與外部設備通訊，USB 的 `Serial` 保留給上傳與線上監看；不要在 `plc86_user_ethercat()` 中列印。

若程式自己用 `Serial.print` 印到 USB，而電腦沒有在讀這個埠（86LDmicro 未連線、序列埠監看未開），緩衝區滿時 `Serial.print` 會一直等待，整個 PLC 掃描因此停住。正式運轉的程式不要印到 USB，或先檢查 `Serial.availableForWrite()`。

10.2 產生的檔案

建置把整個專案（LD、ST、SFC、GVL、I/O 對應、裝置、86HMI 綁定）轉成 C++，寫進 sketch 資料夾，再由 86Duino IDE 的編譯器編譯。這些檔案**每次建置都重新產生，請勿手改**。

檔案	產生者	內容
`plc86.cpp`、`plc86.h`	建置（Ctrl+B）	PLC 程式本體：所有 POU、全域變數、I/O 處理、工作排程；`plc86_setup()` / `plc86_loop()`
`plc86_rt.cpp`、`plc86_rt.h`	建置	執行環境：線上監看協定、強制、RUN / STOP、重啟、RETAIN、事件紀錄、趨勢取樣、看門狗
`plc86_ecat.cpp`、`plc86_ecat.h`	建置	EtherCAT 主站（ENI、從站、PDO 對應、Motion86 機台）；沒有 EtherCAT 裝置時為空的檔案
`<sketch>.ino`	新增專案時建立	呼叫 `plc86_setup()`、`plc86_loop()`；你可以加入自己的程式碼，建置只補上 PLC 需要的部分（例如 `#include "plc86.h"`、`Hmi.begin();`）
`data\86ld\build\plc86.symbols.json`	建置	符號表：線上監看用來比對控制器上的程式與找到變數
`data\86ld\build\bin\`	編譯（F7）	編譯的中間檔與執行檔（`<sketch>.ino.exe`）
`myhmi.cpp`、`myhmi.h`、`data\Resource\`	86HMI 編輯器的「產生程式碼」	86HMI 畫面與資源（第 13 章），不是建置產生

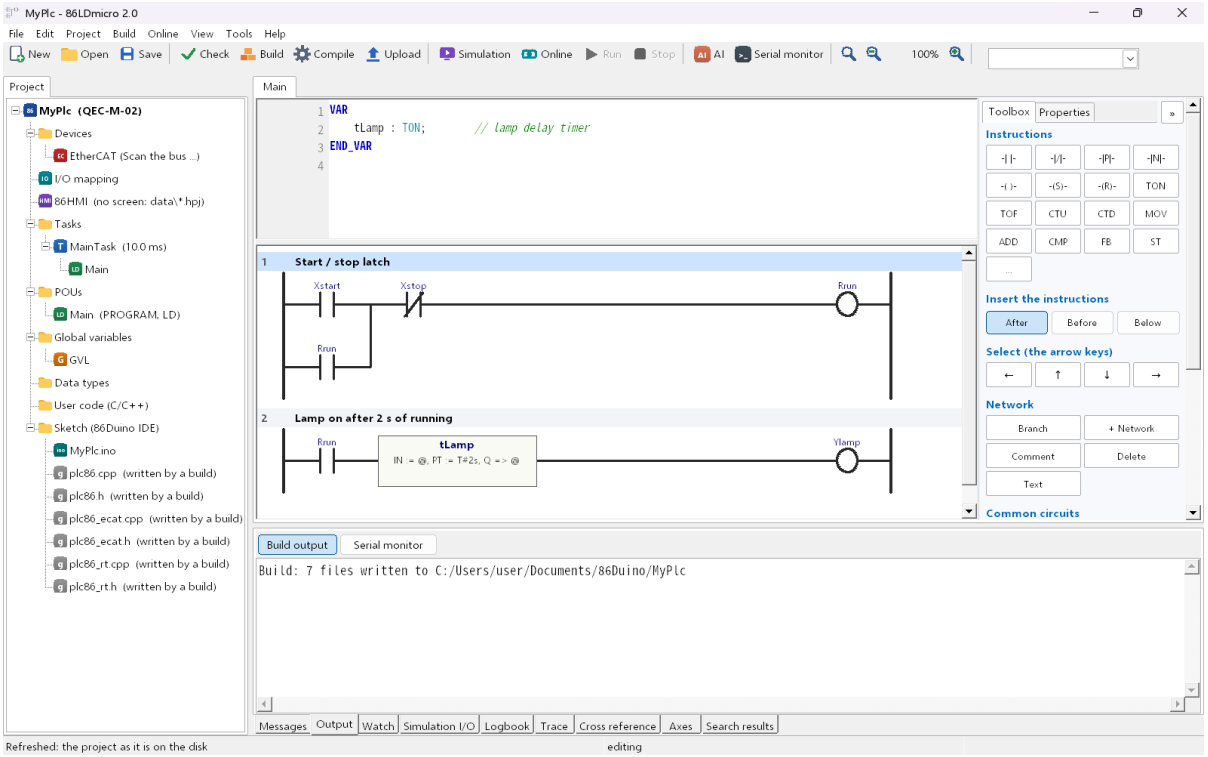


圖 10-6　專案樹的「Sketch（86Duino IDE）」：建置產生的檔案標示「（建置時產生）」，建置輸出顯示「已寫入 7 個檔案」

在專案樹「Sketch（86Duino IDE）」下可以直接開啟 .ino 與這些檔案查看；在 86Duino IDE 中也會看到它們的分頁。

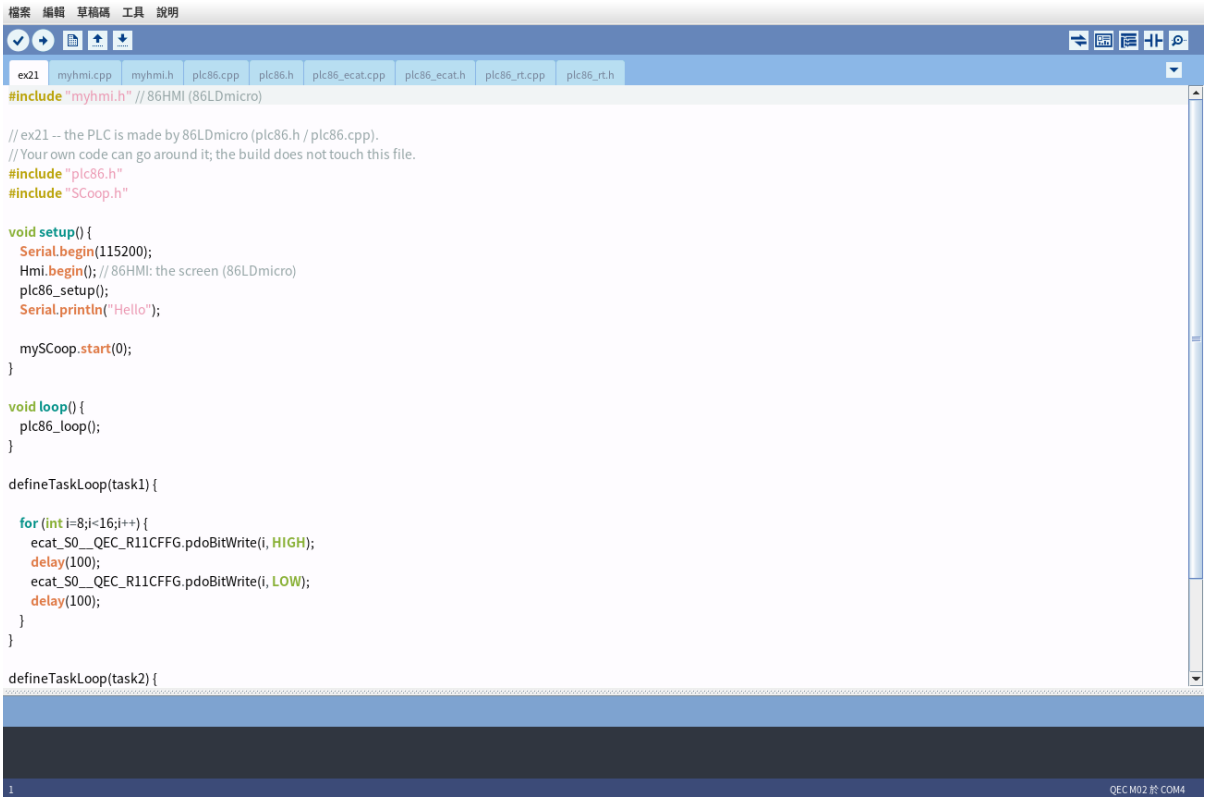


圖 10-7　建置後的 86Duino IDE：plc86.cpp、plc86.h、plc86_ecat.cpp 等分頁

> ***提示**：`plc86*` 檔案與 `data\86ld\build` 可以不納入版本控管（git 等），需要時重新建置即可。

10.3 序列埠監看

「序列埠監看」顯示控制器從序列埠印出的文字，例如程式的 `Serial.print()` 與 EtherCAT 主站的狀態。按工具列的 **序列埠監看**，或在下方「輸出」視窗上方切換到「序列埠監看」。

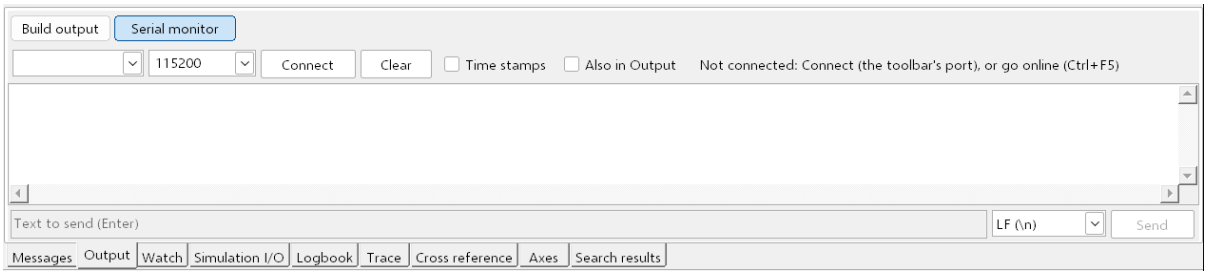


圖 10-8　序列埠監看：選埠、鮑率、連線、時間戳記、傳送

狀況	怎麼看
線上時（Ctrl+F5）	程式 `Serial.print()` 的內容經由線上連線自動顯示，不需另外連線；上線時不能傳送
未上線時	選序列埠（空白 = 工具列的序列埠；也可選 USB 轉 RS-232 / 485 看 Serial1～3）與鮑率，按 **連線**；按 **中斷連線** 結束
上傳時	自動讓出序列埠，上傳完成、控制器重新啟動後自動接回

- **時間戳記**：每行前面加上收到的時間。
- **同時顯示在輸出**：每行也出現在「建置輸出」。
- 下方輸入框輸入文字按 Enter 或 **傳送**（可選換行字元）；**清除** 清空畫面。

有 EtherCAT 裝置時，主站的狀態也印在這裡，不需上線就能確認，例如：

```
EtherCAT: running, 2 slaves, 1000 us, sync, ENI C:/eni/QEC-ENI-1.xml
```

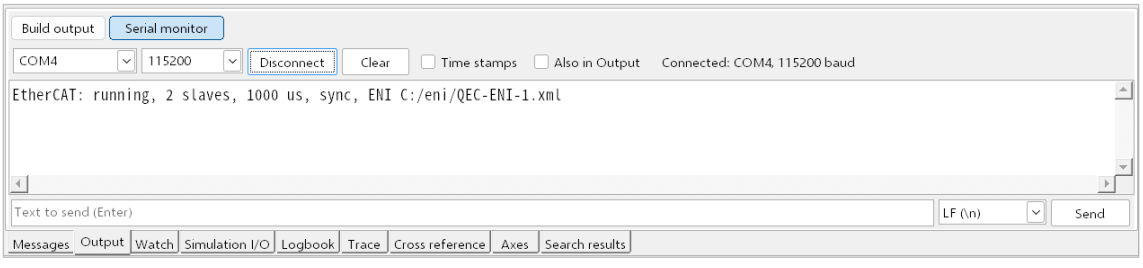


圖 10-9　序列埠監看中的 EtherCAT 主站狀態

10.4 線上監看

線上監看讓 86LDmicro 連到控制器，在電腦上看控制器正在執行的程式：能流、變數值、SFC 步驟，並可寫入、強制、RUN / STOP。

10.4.1 準備

「專案 > 設定」的「**線上監看（監看、強制、能流）**」預設為勾選，使用的序列埠為 `Serial`（USB，與上傳同一條線），鮑率預設 115200。改了這些設定要重新建置、上傳（4.2 節）。

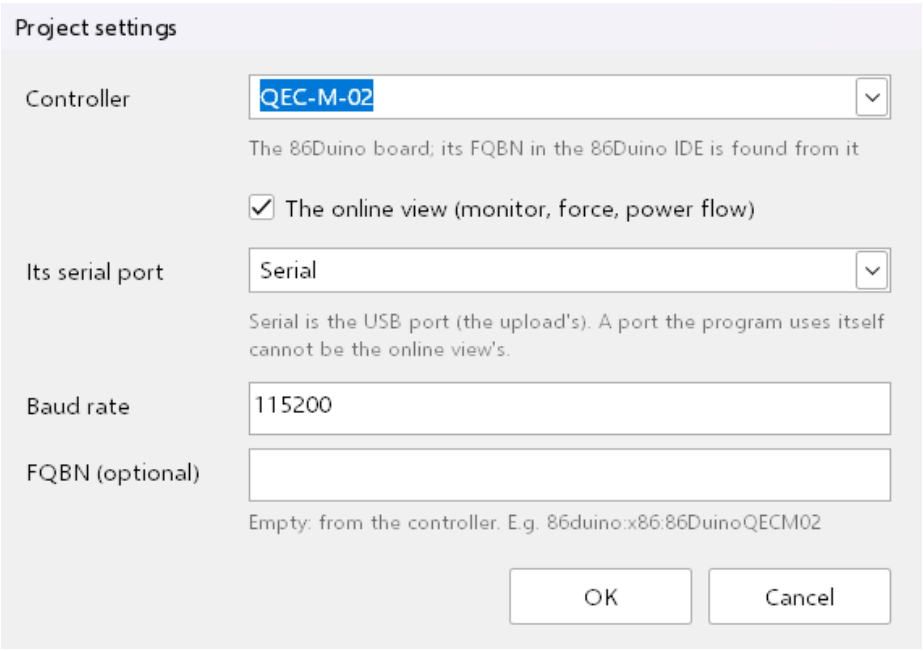


圖 10-10　專案設定：線上監看、使用的序列埠、鮑率

10.4.2 連線

1. 上傳程式（F8）。
2. 按工具列 **線上**，或「線上 > 連線到控制器」（Ctrl+F5）。
3. 連線後，狀態列顯示「線上 COM4：RUN」，以及掃描時間、最大值、掃描次數與逾時次數；階梯圖與 SFC 顯示即時能流與數值。
4. 結束時按「線上 > 中斷連線」，或再按一次工具列的 **線上**。

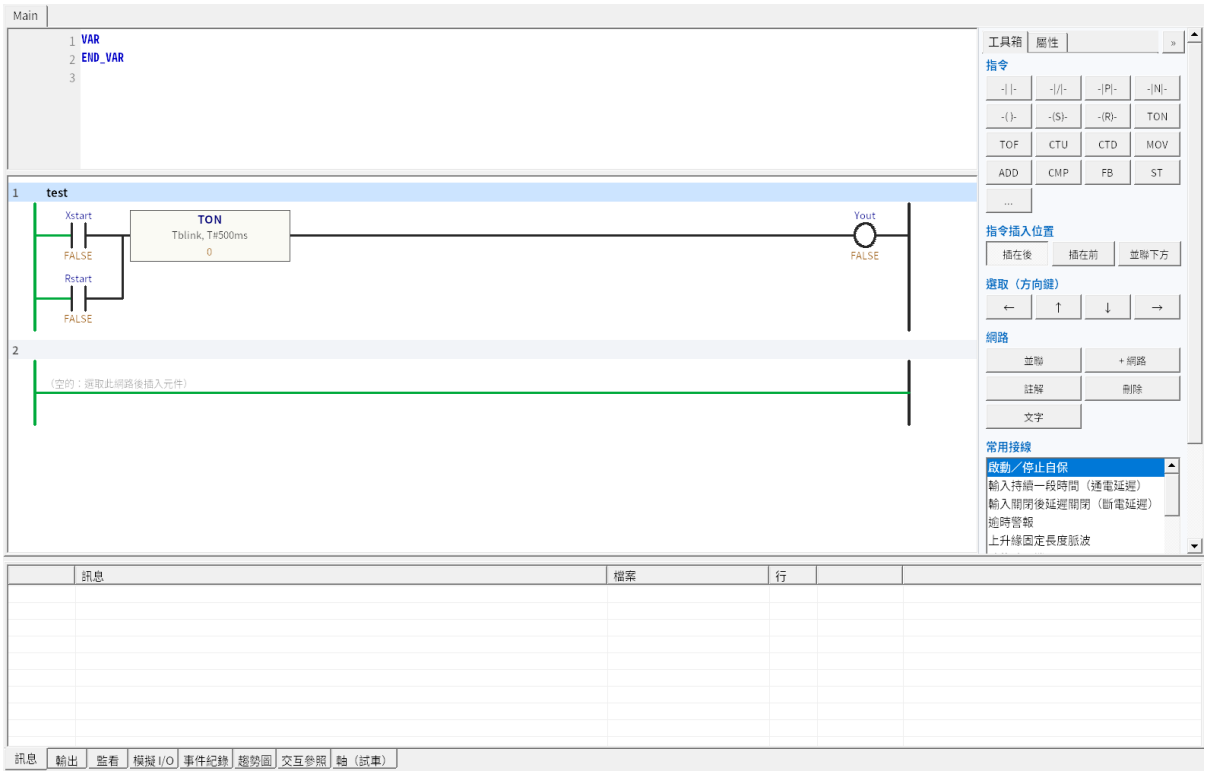


圖 10-11　線上監看：階梯圖顯示控制器上的即時能流



圖 10-12　線上時的狀態列：線上 COM4：RUN 與掃描時間

控制器執行的若不是這次建置的程式，86LDmicro 會拒絕連線並提示「控制器執行的是其他程式（或此程式的其他版本）…請產生程式碼並上傳後再上線」。請重新上傳（F8）。



圖 10-13　控制器上的程式與這次建置不同時的提示

10.4.3 線上的操作

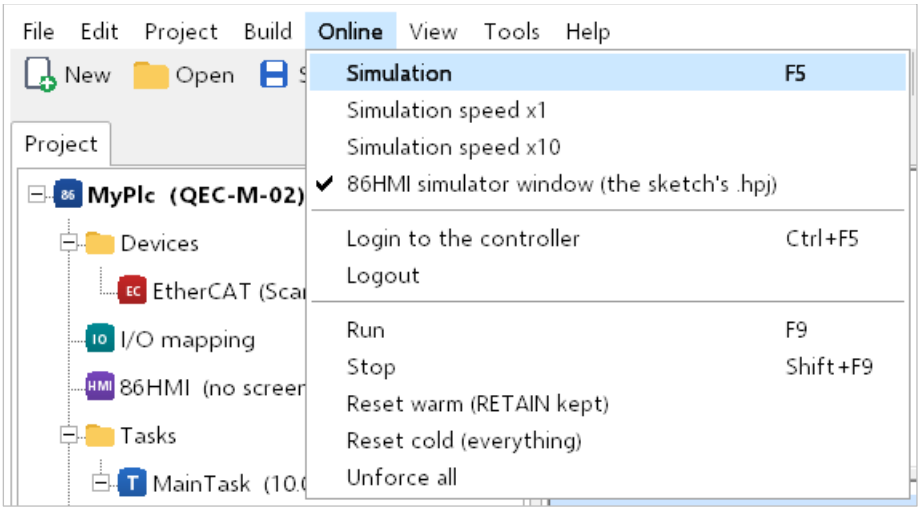


圖 10-14　線上時的「線上」選單

項目	快捷鍵	用途
連線到控制器	Ctrl+F5	開始線上監看
中斷連線	—	結束線上監看（控制器繼續執行）
執行 RUN	F9	程式繼續執行
停止 STOP	Shift+F9	程式停止執行；I/O 與 EtherCAT 照常更新，輸出保持最後的值；所有 Motion86 機台停止、寸動結束
暖重啟（保留 RETAIN）	—	變數回到初值，RETAIN 變數保留，並解除全部強制；RUN / STOP 狀態不變（RUN 時程式從初值重新開始）
冷重啟（全部）	—	所有變數（含 RETAIN）回到初值
解除全部強制	—	解除所有強制

*安全：STOP 只是不執行程式，**不是安全停止**：輸出保持最後的值。設備的緊急停止請以硬體安全迴路實現。

監看、寫入、強制、趨勢圖的操作與模擬相同（9.3、9.4 節）：

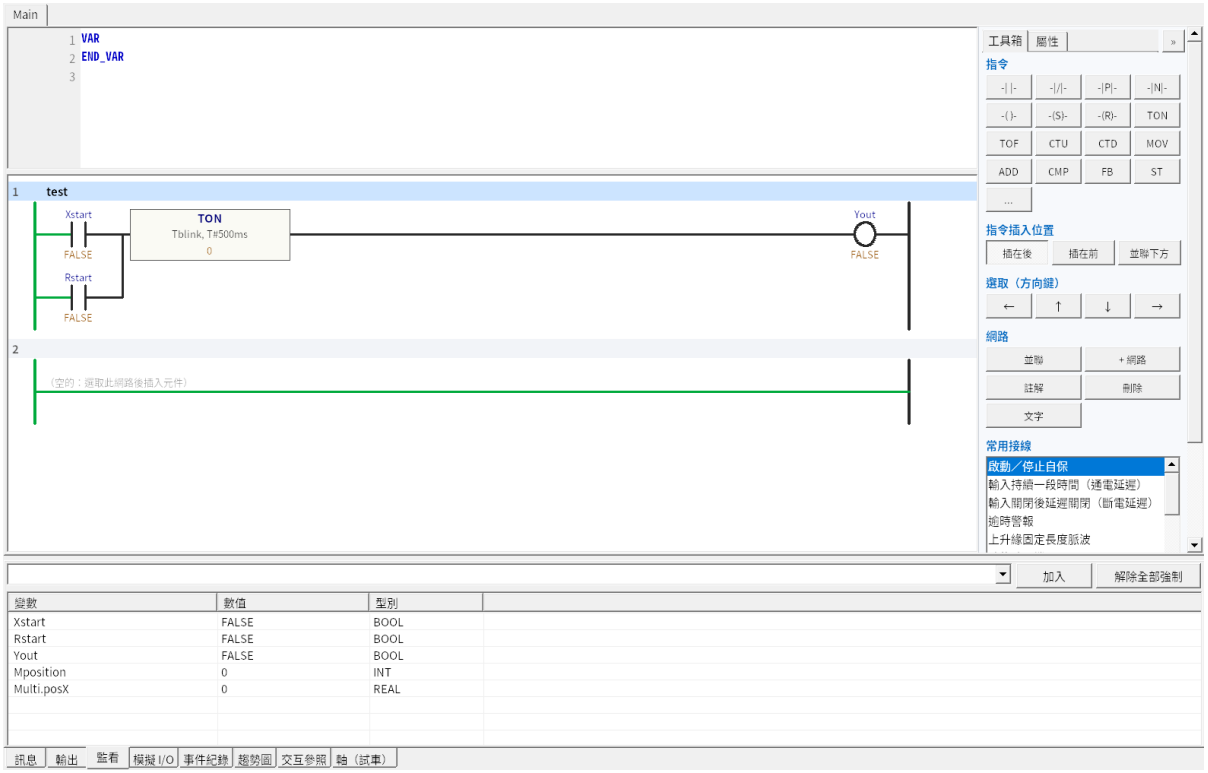


圖 10-15　線上：階梯圖與監看視窗顯示控制器上的數值

- **能流與數值**：階梯圖的綠色線段、元件下方的數值；SFC 的綠色步驟；ST 中滑鼠停在變數上的數值。
- **監看視窗**：雙擊寫入一次；右鍵強制（顯示 `[F]`）、解除強制、加入趨勢圖。
- **EtherCAT**：SubDevice 頁顯示每個 PDO 項目的即時值（11.6 節）。
- **軸**：機台頁、軸頁與「軸（試車）」視窗操作控制器上的軸（第 12 章）。
- **事件紀錄**：控制器的 logbook（10.5 節）。

> ***注意**：連線中斷 3 秒（拔掉 USB 線、電腦休眠等），控制器會自動解除所有強制。

10.5 事件紀錄與看門狗

10.5.1 事件紀錄

「**事件紀錄**」視窗（下方工具視窗的分頁）顯示控制器的 logbook：最近 32 筆事件，每筆有時間（控制器開機後的秒數）與訊息。模擬時顯示模擬器的紀錄，時間欄顯示「模擬」。

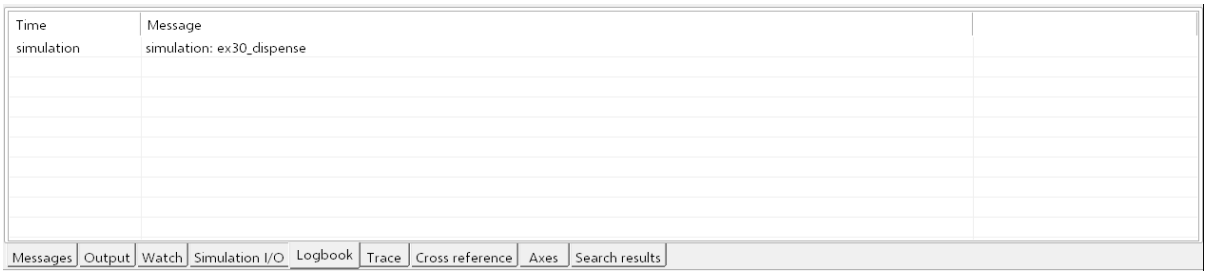


圖 10-16　事件紀錄：EtherCAT 主站啟動訊息與程式開始執行

事件	例
開機、程式開始	`start: MyPlc (build 1A2B3C4D)`（括號中為這次建置的識別碼）
RETAIN	`RETAIN: the values kept are loaded`、`RETAIN: nothing stored (first start, or RETAIN changed): initial values`
RUN / STOP（由電腦）	`RUN (from the PC)`、`STOP (from the PC)`
暖重啟 / 冷重啟	`warm restart: the variables to their initial values, RETAIN kept`、`cold restart: every variable to its initial value`
連線中斷	`the PC is gone: the forces are released`
Motion86	`Motion86: an axis' drive left operation enabled: the machine stopped (MC_Reset, MC_Power again)` 等（12.9 節）
掃描逾時	`task MainTask: a scan took 12500 us (the interval is 10000 us)`（第一次與每 1000 次記錄一筆）
看門狗	`MainTask watchdog: STOP, 152 ms (3 other), Motion 140 ms`
EtherCAT	`EtherCAT: running, 2 slaves, 1000 us, sync, …`、`EtherCAT: begin failed`、`no slave N`
模擬的執行錯誤	`STOP: Main line 12: …`（只在模擬）

10.5.2 看門狗

看門狗是掃描逾時檢查：每個週期工作都可以設定「看門狗（us）」，一次掃描中程式的時間超過這個值，PLC 就停止（STOP）；0 表示不使用看門狗。在專案樹「工作（Task）」下雙擊工作即可設定（4.3 節）。

> ***注意：看門狗在每次掃描結束、程式返回後才判斷時間。程式在掃描中進入無窮迴圈而不再返回時，看門狗無法中斷它；它不是獨立的硬體看門狗，也不能取代設備的安全迴路。**
>
> 請勿在運轉中的設備上測試無窮迴圈；有限長度的長迴圈可用來確認看門狗會停止 PLC。

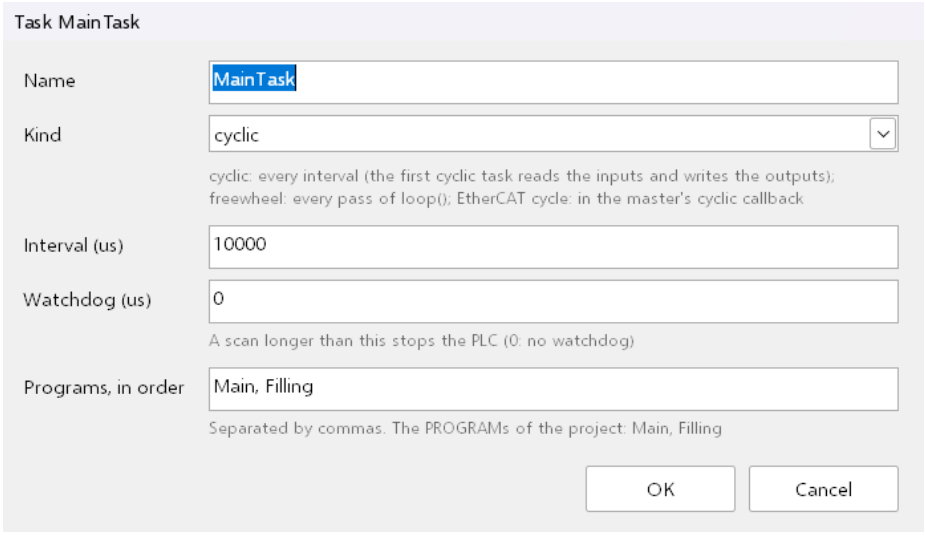


圖 10-17　工作的設定：週期與看門狗（us）

控制器判斷看門狗的方式：

- 量測的是該工作**一次掃描中程式本身的時間**：整個掃描時間，扣掉其中不屬於程式的時間——EEPROM 寫入（PERSIST；86Duino 的 EEPROM 偶爾要清除 flash sector，可能停頓數百 ms）與 EtherCAT 週期中斷（Motion86、EtherCAT 工作）。
- RETAIN 的寫入在掃描之間進行，86HMI 畫面的更新在第一個工作的掃描之後執行，都**不算**在掃描時間內。
- 開機後的**前 3 個掃描**不判斷（載入 RETAIN、變數第一次初始化）。
- 超過設定時：PLC 進入 STOP（程式不再執行，輸出保持最後的值），所有 Motion86 機台停止，並在事件紀錄寫入一筆看門狗紀錄。
- 模擬器不判斷看門狗；請在控制器上確認。

事件紀錄中的看門狗紀錄格式：

```
MainTask watchdog: STOP, 152 ms (3 other), Motion 140 ms
```

欄位	意義
`MainTask`	超時的工作
`152 ms`	這次掃描的總時間
`(3 other)`	其中 EEPROM 寫入、EtherCAT 中斷等非程式的時間（ms），不列入判斷
`Motion 140 ms`	這次掃描中**最慢的程式**與其時間：優先從這個程式找原因

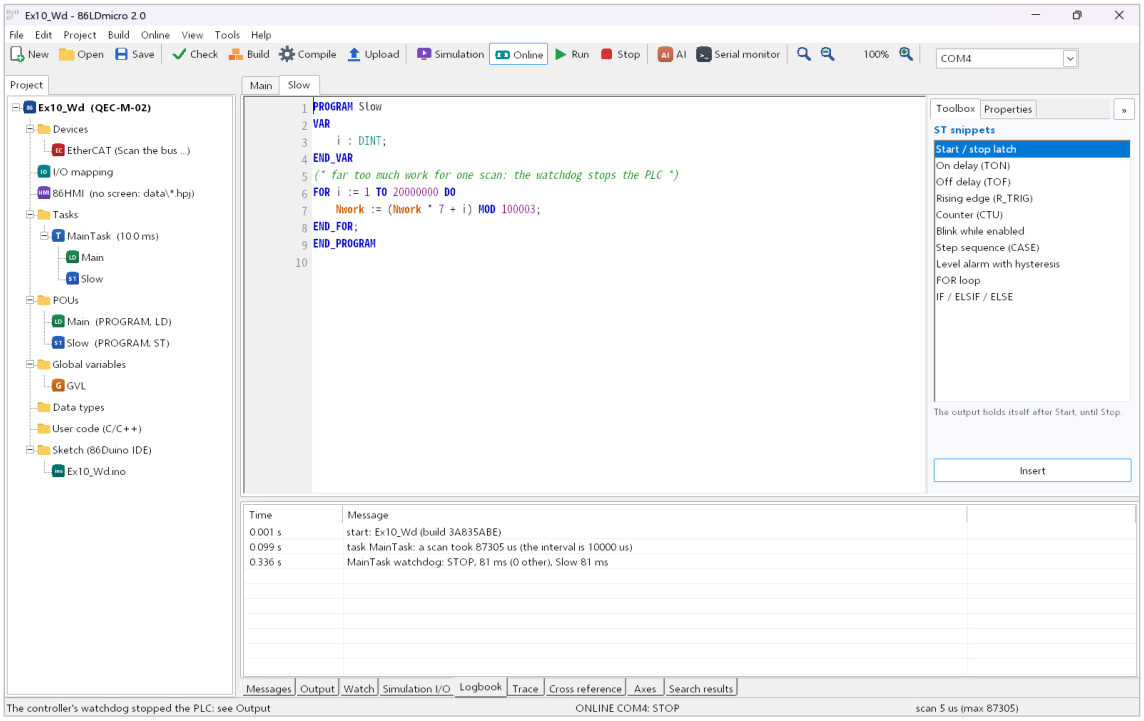


圖 10-18　事件紀錄中的看門狗紀錄

處理方式	說明
減少每個掃描的運算	長迴圈分散到多個掃描（例如每個掃描處理 100 筆）；只在需要時計算
移到較慢的工作	不需要每 10 ms 執行的程式（統計、配方、文字）放到 100 ms 的工作；這只降低執行頻率，不會縮短單次掃描的時間，單次太長仍要分散到多個掃描
加大看門狗時間	確認掃描時間合理後，把看門狗設為正常最大掃描時間的數倍
重新開始	排除原因後按 RUN（F9），或重新上傳

*__提示：__線上時的狀態列顯示每個掃描的時間與最大值（「掃描 … ms（最大 …）」）與逾時次數，可用來決定看門狗與工作週期。

10.6　上傳 86HMI 資源

86HMI 編輯器「產生程式碼」時，除了 `myhmi.cpp` / `myhmi.h`，也會把畫面用到的圖片、字型與畫面資料寫到 sketch 的 `data\Resource`。這些**資源**存在控制器上，和程式分開（第 13 章）。

情況	做法
第一次上傳，或程式與畫面都改了	按 **上傳**（F8）：程式與 `data\Resource` 一起上傳
只改了圖片、字型、畫面外觀，PLC 程式沒改	按 **上傳 86HMI 資源**：只送資源，不編譯，控制器上的程式不變（比 F8 快）

1. 工具列確認 COM 埠（例如 COM4）。
2. 「建置 > 上傳 86HMI 資源（data\Resource）」，或在 86HMI 頁按 **上傳資源**。
3. 「建置輸出」出現「== 上傳資源到 COM4: …\data\Resource ...」與 v86dude 的進度，最後「== 資源上傳完成」。
4. 原本在線上時，上傳前會自動離線，完成後自動重新連線。

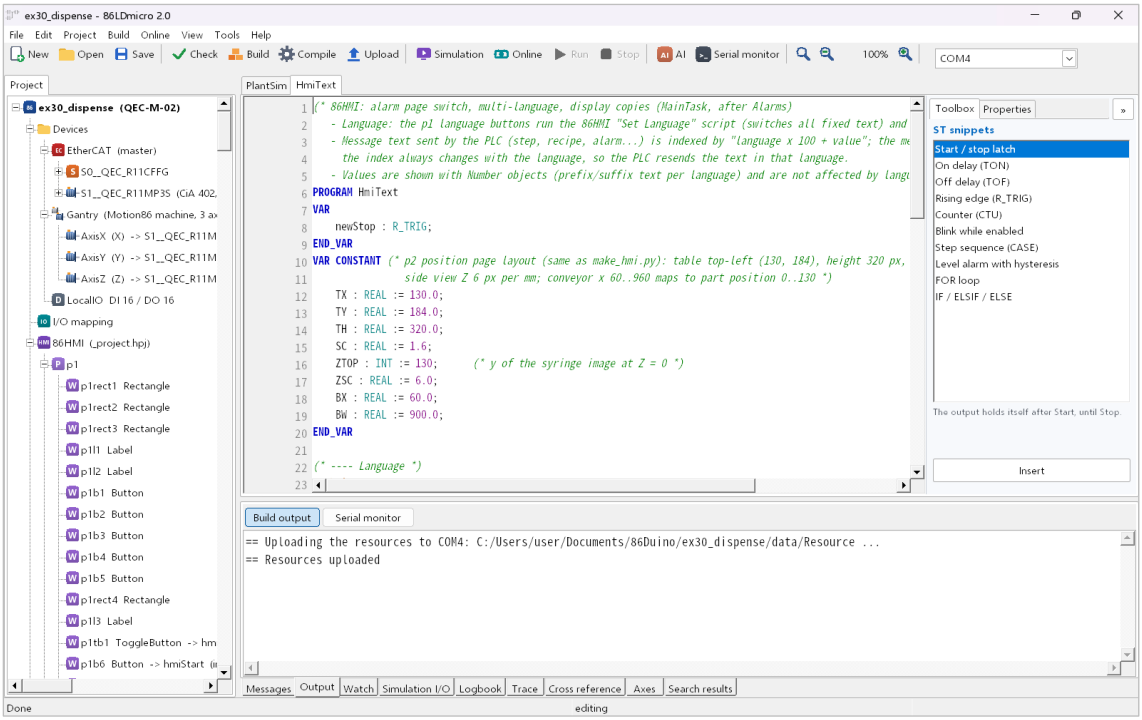


圖 10-19　上傳 86HMI 資源：建置輸出顯示上傳進度與「資源上傳完成」

> ***提示**：這個功能與 86Duino IDE 的「工具 > 上傳資源檔至 QEC M02」相同。若出現「沒有可上傳的資源」，表示 sketch 沒有 `data\Resource` 或是空的：先在 86HMI 編輯器開啟 .hpj 按「產生程式碼」。命令列：`86ldmicro-cli upload-res <專案> <COM>`（15.5 節）。

第 11 章　EtherCAT

QEC-M-02 是 EtherCAT 主站。86LDmicro 2.0 的專案可以擁有自己的 EtherCAT 主站：以內建的 EtherCAT Configurator 掃描網路、設定從站並匯出 ENI，再把 ENI 讀回專案，把每個 PDO 項目對應到變數，程式就能直接讀寫遠端 I/O 與伺服驅動器。本章以 `tools\example\ex30_dispense`（QEC-R11CFFG I/O 模組 + QEC-R11MP3S 三軸驅動器）為例。

11.1　概念

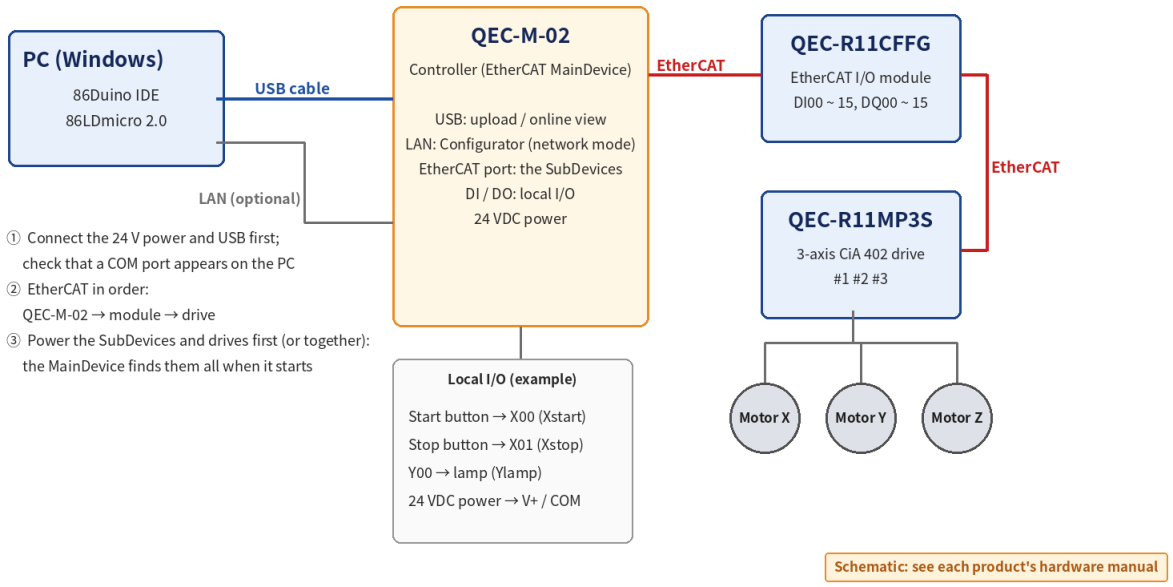


圖 11-1　典型系統：電腦經 USB 連到 QEC-M-02，EtherCAT 串接 I/O 模組與驅動器（示意圖）

名詞	說明
主站（MainDevice）	QEC-M-02 控制器，負責每個 EtherCAT 週期與所有從站交換資料
從站（SubDevice）	EtherCAT 網路上的 I/O 模組、伺服驅動器等，例如 QEC-R11CFFG、QEC-R11MP3S
PDO（過程資料）	每個週期交換的資料：RxPDO 為主站送到從站（輸出），TxPDO 為從站送回主站（輸入）
SDO / CoE	非週期的參數讀寫（物件字典，Index:Sub），例如驅動器的參數
ESI	從站廠商提供的描述檔（.xml）：物件字典、PDO、圖示、模組定義
ENI	網路組態檔（.xml）：有哪些從站、交換哪些 PDO、DC 同步、啟動參數；存放在控制器的 `C:/eni/`
Configurator	QEC 的 EtherCAT 設定工具（網頁介面）：Scan、Configure、PDO、DC、CoE、Export ENI
DC（分散式時鐘）	讓所有從站在同一時刻動作；伺服驅動器（CSP）建議使用

86LDmicro 支援兩種 EtherCAT 使用方式：

方式	說明	何時使用
EtherCAT 網路（ENI）	專案自己的主站，網路由 ENI 描述；支援任意 EtherCAT 從站、CiA 402 驅動器、Motion86 CSP	2.0 的建議方式（本章）
86EVA	1.x 的方式：86EVA 工具產生 `myeva.h` / `cpp` 管理主站與虛擬 I/O	沿用 1.x 專案

*__注意：__兩種方式不可同時使用（都要驅動同一個主站）。改用 ENI 時，請移除 .ino 中 1.x 的 `#include "myeva.h"` 與 `EVA.begin()`，並把 sketch 資料夾中的 `myeva.cpp` / `myeva.h` 移走（建置時會提出警告）。

11.2 完整流程

從一個空的網路到程式可以讀寫 PDO，依序完成下列步驟。前五步在 Configurator（11.3 節）中進行，之後回到 86LDmicro。

步驟	在哪裡	做什麼	章節
1	86LDmicro EtherCAT 頁	選連線方式，按 **上傳 Configurator**（只需一次）	11.3.1
2	Configurator	（建議）上傳並載入從站的 ESI	11.3.5
3	Configurator	**Scan** 掃描從站 → **Configure**	11.3.3
4	Configurator	視需要設定 PDO、DC、Startup、Slots；**Start** 測試	11.3.6
5	Configurator	**Export ENI** → **Upload** 存到控制器的 `C:/eni/`	11.3.5
6	86LDmicro	**讀入專案**：從站與 PDO 載入專案	11.4
7	86LDmicro	把 PDO 對應到變數；CiA 402 驅動器建立 Motion86 機台	11.4、第 12 章
8	86LDmicro	建置、上傳 PLC 程式（取代 Configurator）；確認主站狀態	11.4.4
9	86LDmicro SubDevice 頁	線上看 PDO 值、寫入 / 強制、SDO、CoE 物件字典	11.5、11.6

11.2.1　步驟 1：開啟 EtherCAT 頁，上傳 Configurator

「專案 > EtherCAT（掃描、PDO、ENI）」，或在專案樹雙擊「裝置 > EtherCAT」。選連線方式後按 **上傳 Configurator**。

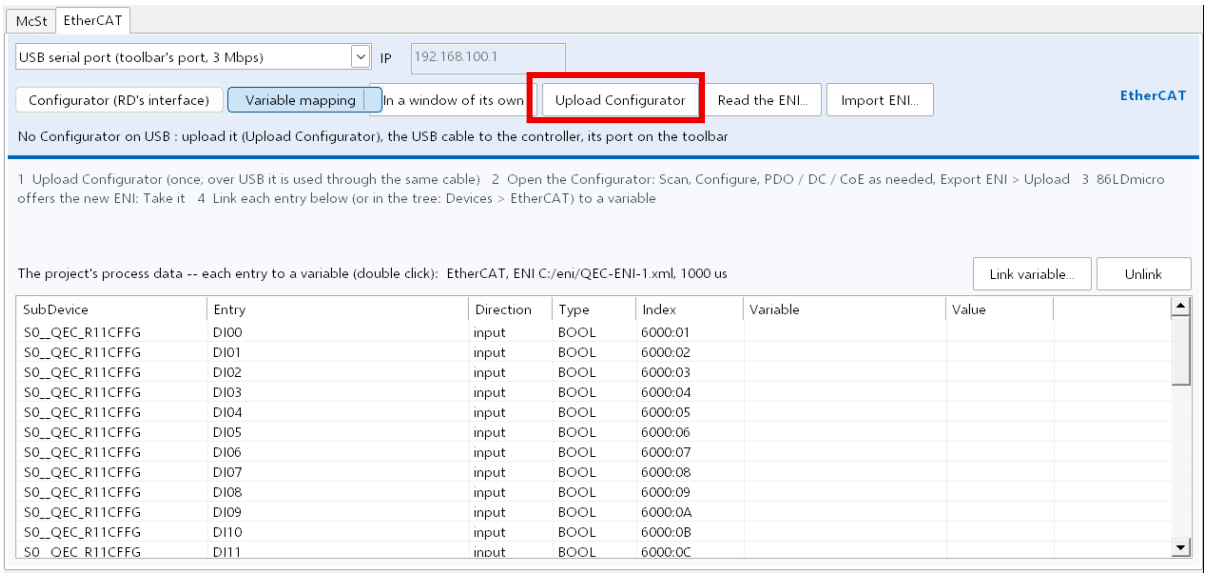


圖 11-2　EtherCAT 頁：連線方式、Configurator（RD 介面）、變數對應、上傳 Configurator、讀回 ENI、匯入 ENI

11.2.2　步驟 2：在分頁中開啟 Configurator

按 **Configurator（RD 介面）**，Configurator 網頁直接顯示在 EtherCAT 頁中。

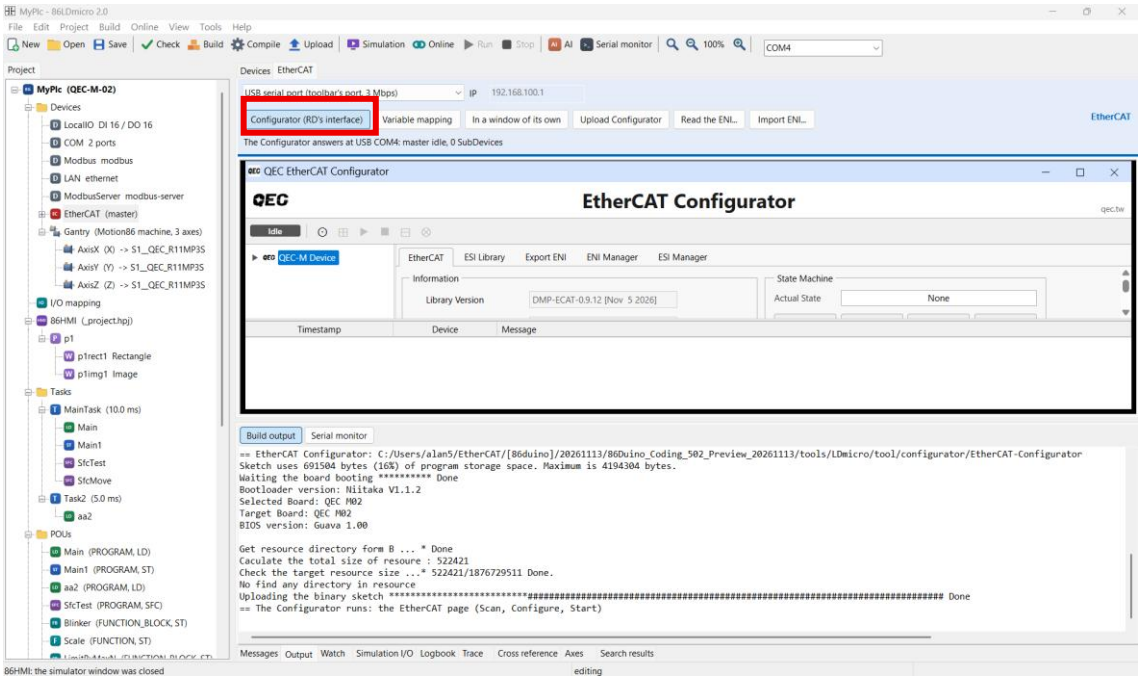


圖 11-3　Configurator 網頁嵌入在 EtherCAT 頁中（USB 模式）

可以點選 **在獨立視窗開啟**，Configurator 會獨立顯示頁面。

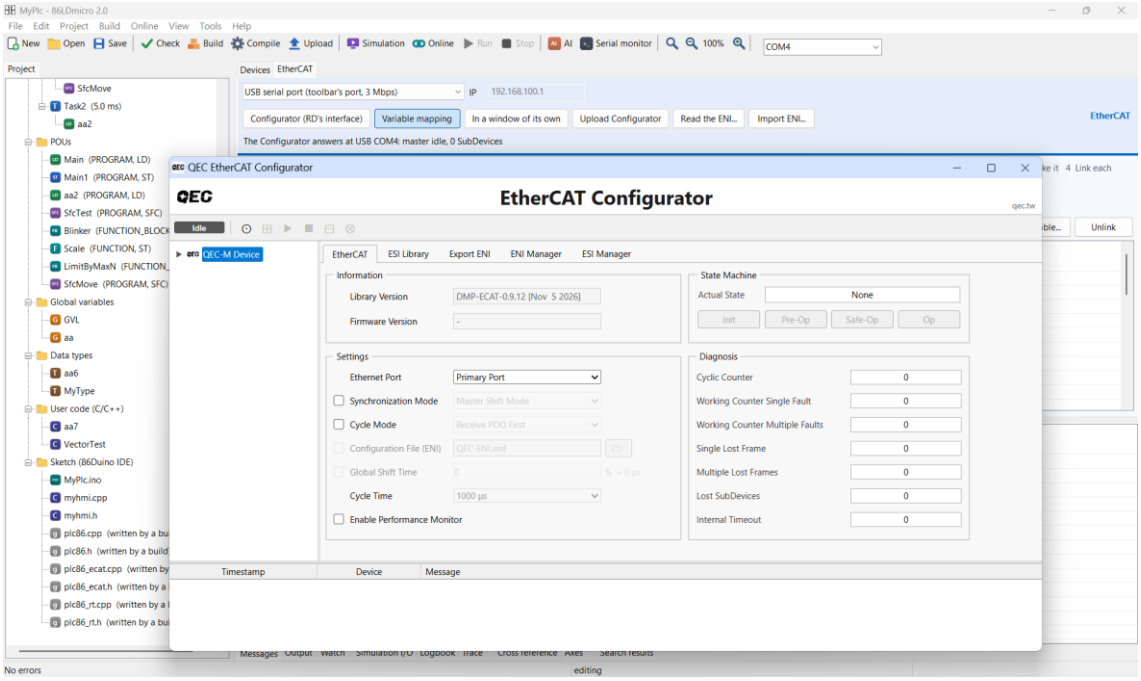


圖 11-3-1　Configurator 網頁獨立頁（USB 模式）

11.2.3　步驟 3、4：Scan、Configure、Start 測試

在 Configurator 按 **Scan**，左側裝置樹列出所有從站。



圖 11-4-1　狀態標籤 Idle（尚未掃描）

按 **Configure** 讀取 PDO。



圖 11-4-2　狀態標籤 Init

按 **Start** 後，所有從站進入 OP 狀態。



圖 11-4-3　狀態標籤 Config（Configure 完成）

所有從站進入 OP 狀態。



圖 11-4-4　狀態標籤 Run（週期交換中）

左側裝置樹列出所有從站。

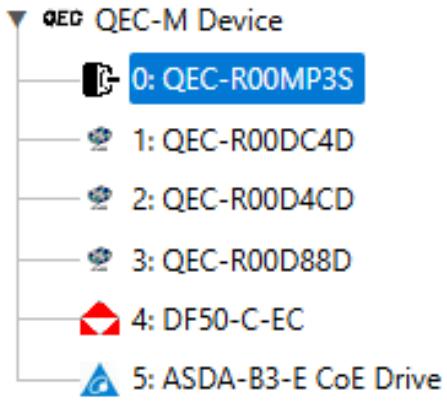


圖 11-4-5　Scan 之後：裝置樹列出每個從站

在從站的 **Online** 分頁看即時值。

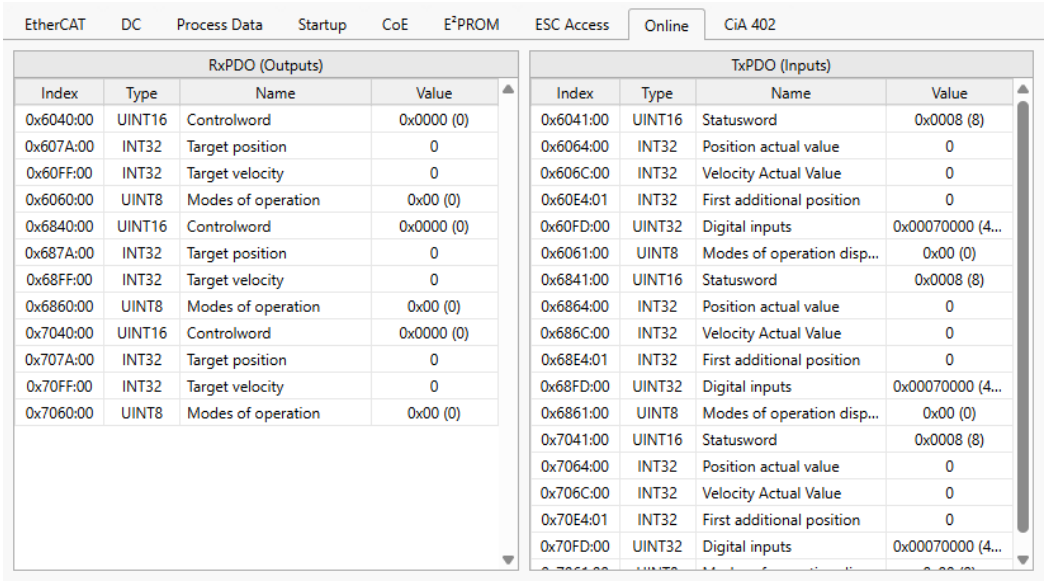


圖 11-4-6　Start 之後：從站的 Online 分頁顯示即時 PDO 值

11.2.4 步驟 5：匯出 ENI 並上傳到控制器

點選 **QEC-M Device > Export ENI** 分頁。

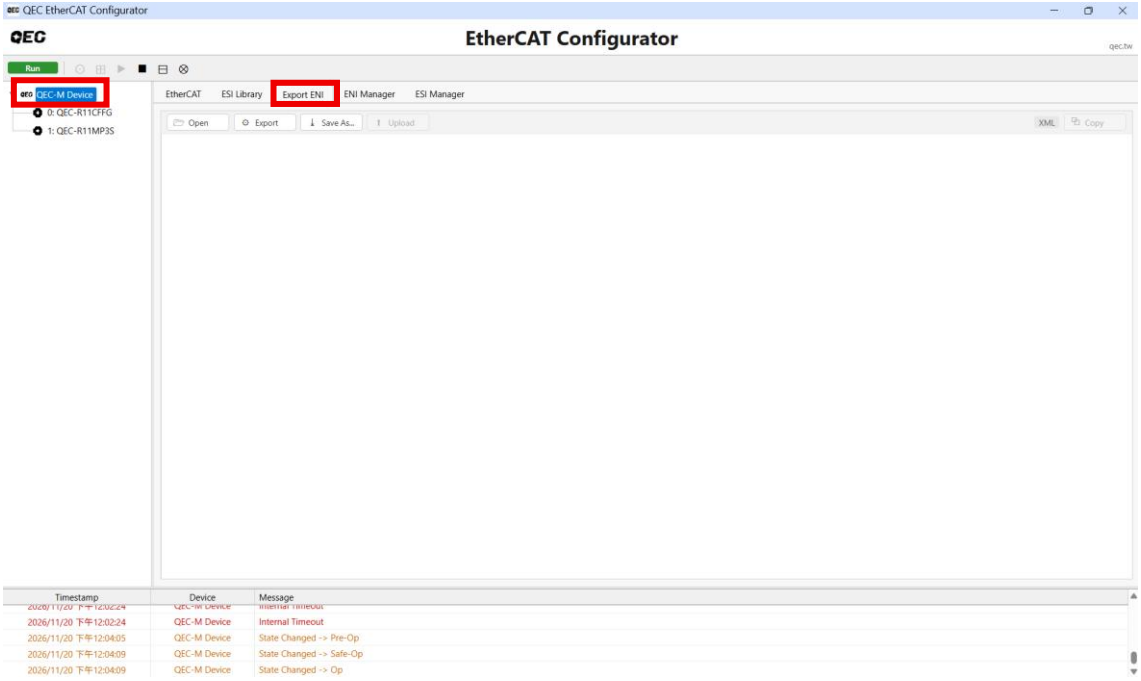


圖 11-5-1　Export ENI 分頁

按 **Export** 產生 ENI。

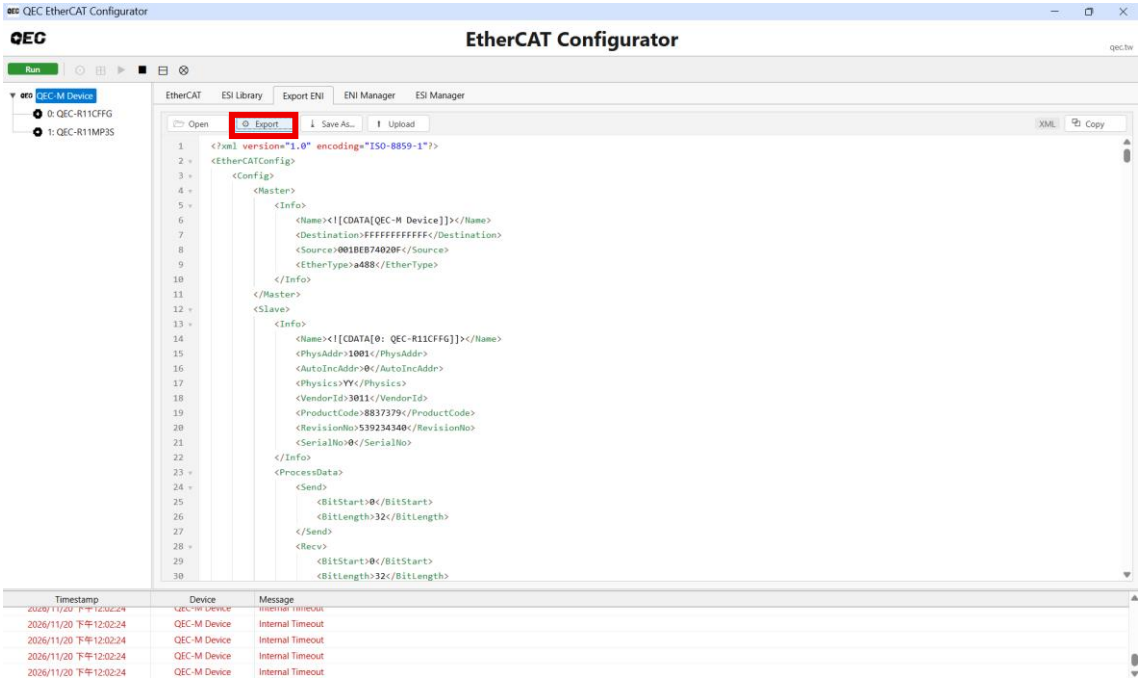


圖 11-5-2　按 Export 產生 ENI

再按 **Upload** 輸入檔名上傳。

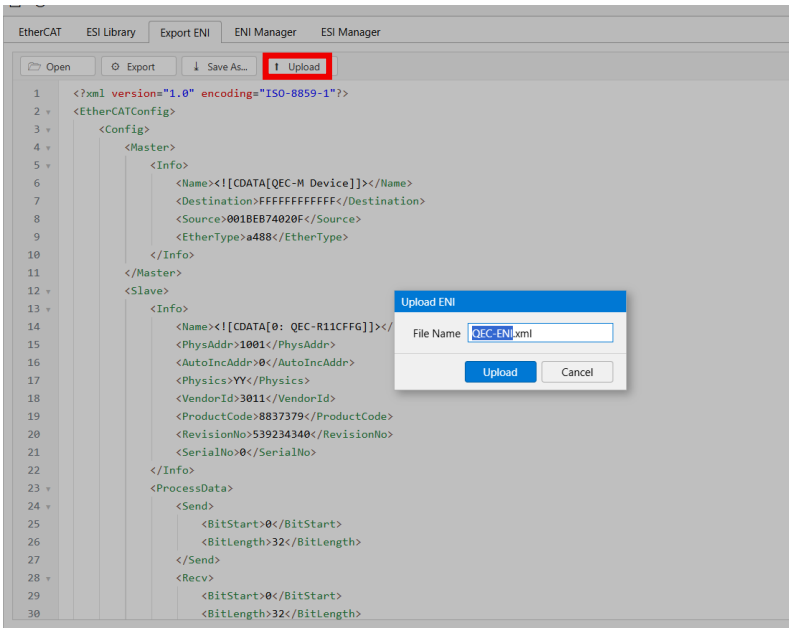


圖 11-6　Upload ENI：輸入檔名，存到控制器的 C:/eni/

11.2.5　步驟 6、7：讀入專案並對應變數

回到 86LDmicro，偵測到控制器上的新 ENI 時出現提示，按 **讀入專案**。左方專案樹的 EtherCAT 類別將會出現 ENI 所對應的從站裝置名稱。

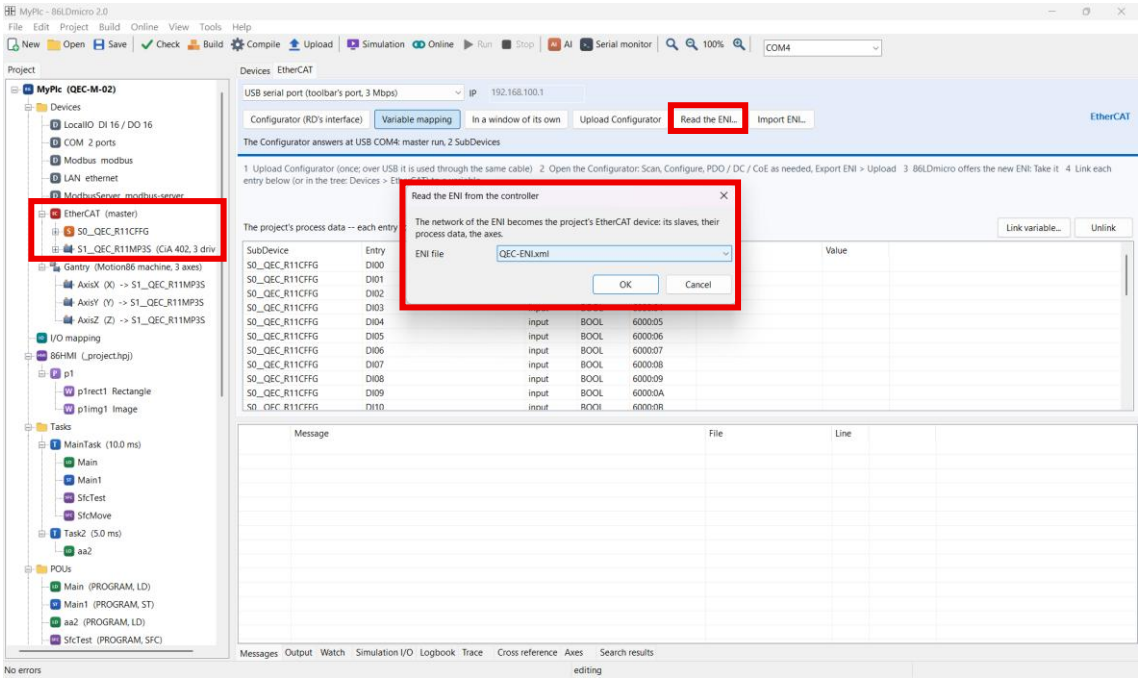


圖 11-7-1　偵測到控制器上的新 ENI：讀入專案

再到「變數對應」把每個 PDO 項目對應到變數。

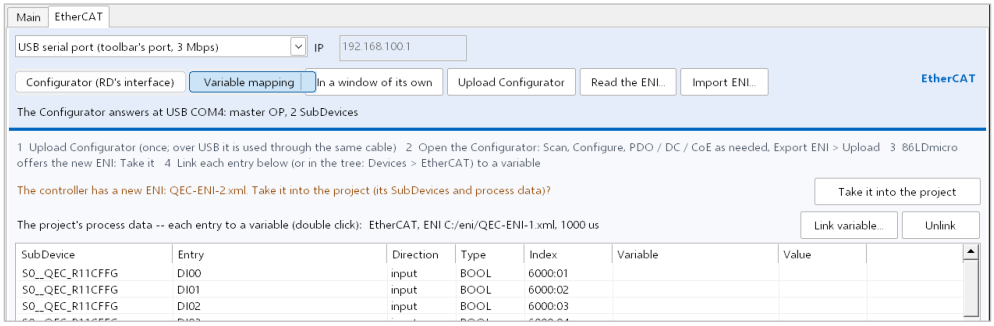


圖 11-7-2　偵測到控制器上的新 ENI：讀入專案

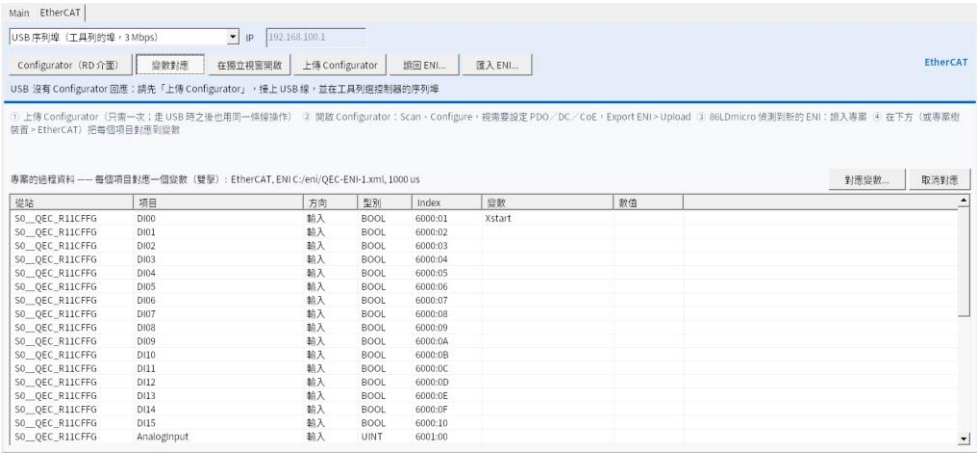


圖 11-8　「變數對應」：每個 PDO 項目一列，已對應的顯示變數

11.2.6　步驟 8、9：上傳 PLC 程式並確認

按 **上傳**（F8）。

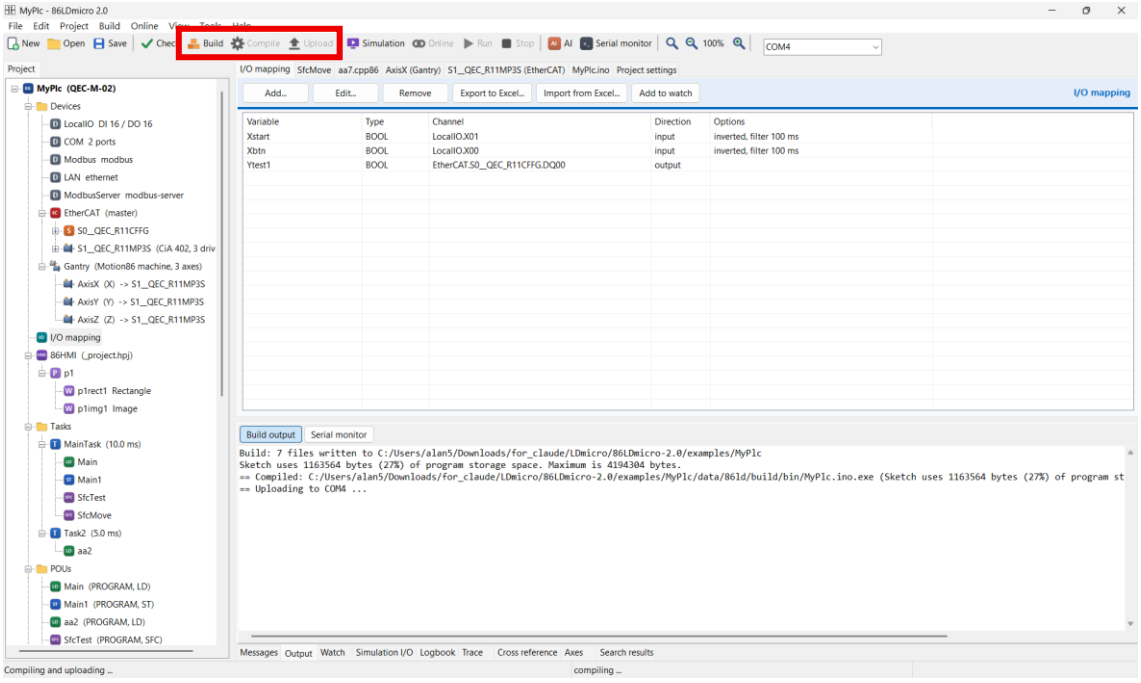


圖 11-9-1　上傳

PLC 程式取代 Configurator，開機時主站以 ENI 啟動。打開「序列埠監看」確認主站狀態，上線後在 SubDevice 頁看 PDO 即時值。

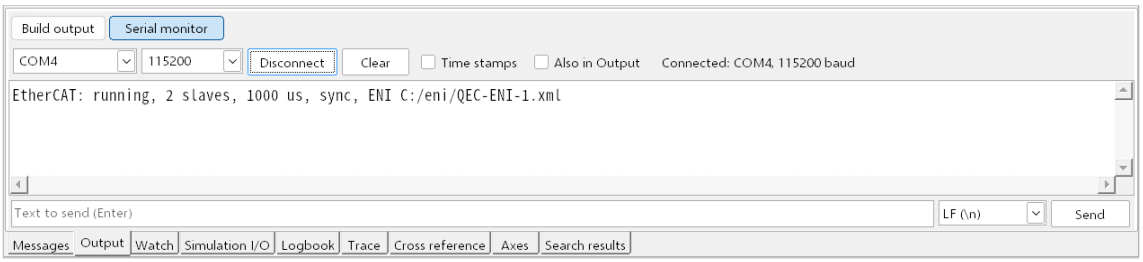


圖 11-9　序列埠監看：EtherCAT: running, 2 slaves, 1000 us, sync

11.3　EtherCAT Configurator

EtherCAT Configurator 是執行在 QEC 控制器上的設定程式，以網頁操作。86LDmicro 負責把它上傳到控制器，並把網頁直接嵌入 EtherCAT 頁中；網路設定完成後匯出 ENI，交回 86LDmicro。

11.3.1　上傳與開啟

連線方式	說明
USB 序列埠（工具列的埠，3 Mbps）	預設。與上傳、線上監看同一條 USB 線；86LDmicro 在電腦上提供網頁畫面，只有 API 經序列埠送到控制器
網路（控制器 IP）	電腦接控制器的 LAN 埠，填控制器的 IP（預設 `192.168.100.1`）

1. 在 EtherCAT 頁選連線方式（USB 時，在工具列選控制器的 COM 埠）。
2. 按 **上傳 Configurator**（只需一次）。86LDmicro 編譯並上傳 Configurator 程式，它會**暫時取代**控制器上的 PLC 程式。也可用「工具 > EtherCAT > 上傳 Configurator 到控制器...」。
3. 上傳完成後，頁面自動偵測 Configurator 是否回應與主站狀態，例如「COM4 的 Configurator 已回應：主站 Idle，0 個從站」。
4. 按 **Configurator（RD 介面）**：Configurator 網頁顯示在這個分頁中（以 Microsoft Edge 嵌入）。按 **在獨立視窗開啟** 可拆成獨立視窗（例如拖到第二螢幕）；按 **變數對應** 回到 PDO 清單。

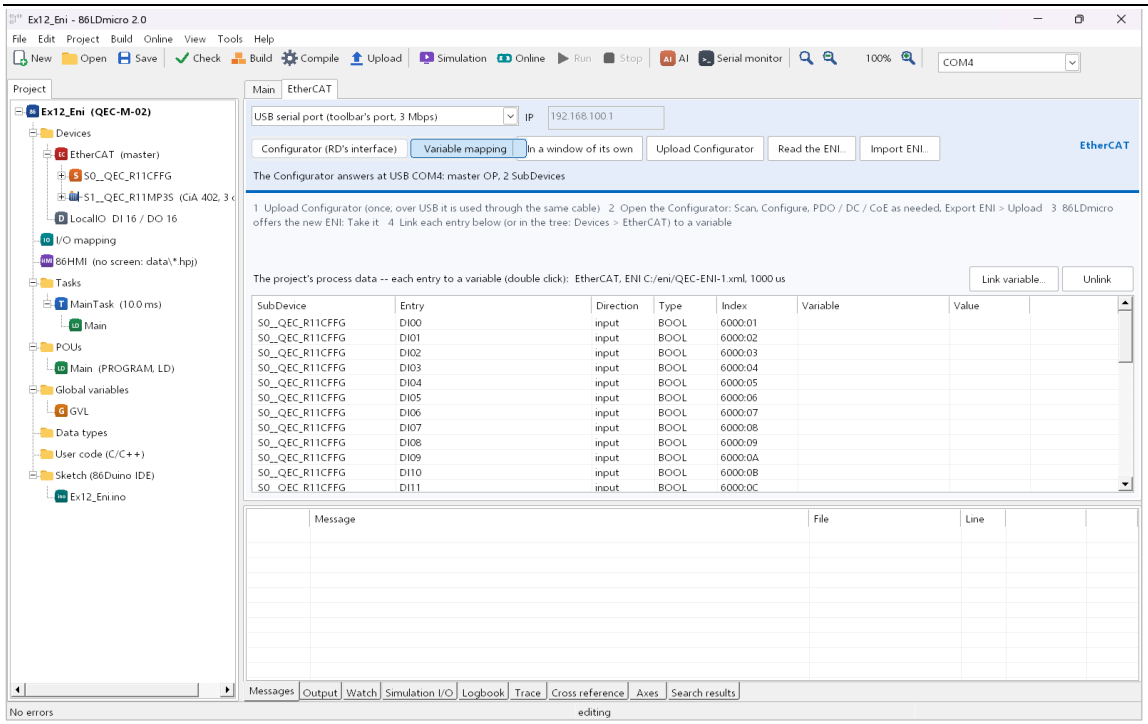


圖 11-10　上傳 Configurator：建置輸出的進度與完成後頁面上的回應訊息

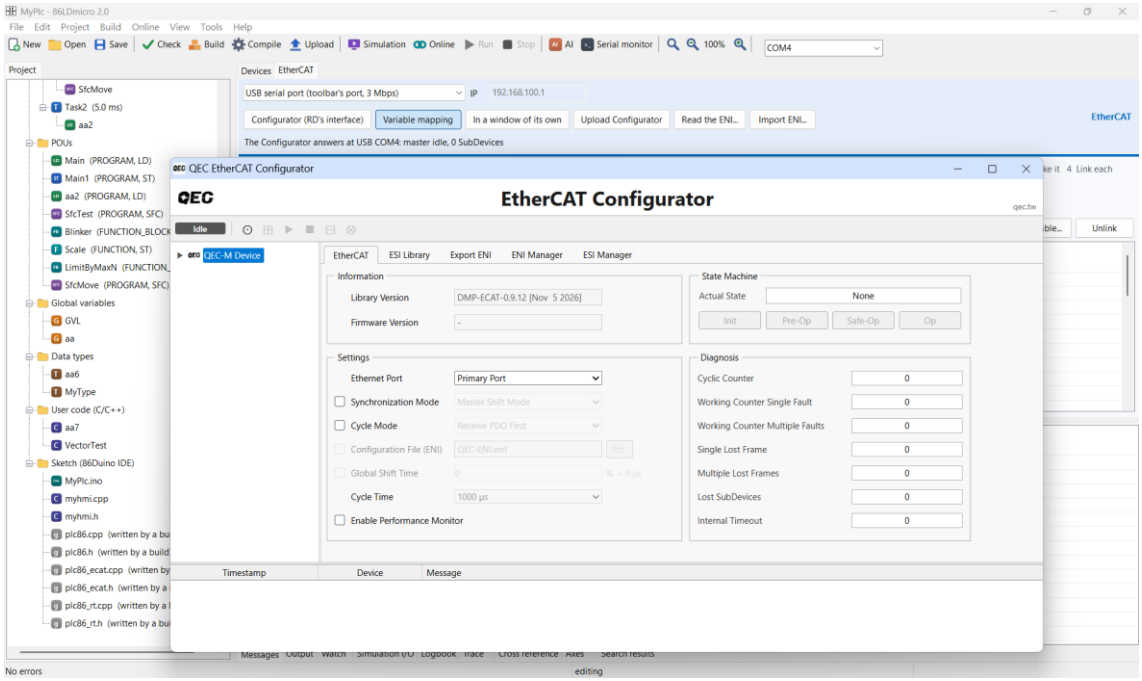


圖 11-11　「在獨立視窗開啟」：Configurator 以獨立視窗顯示

***提示：**USB 模式時，上傳或線上監看會自動暫停 Configurator 的連線；EtherCAT 頁不在前景時也不佔用序列埠（閒置 2 秒即釋放）。

網路模式也可以用瀏覽器開啟：「工具 > EtherCAT > Configurator 網頁版（瀏覽器）...」。

11.3.2 畫面

Configurator 畫面由上而下分為五區：

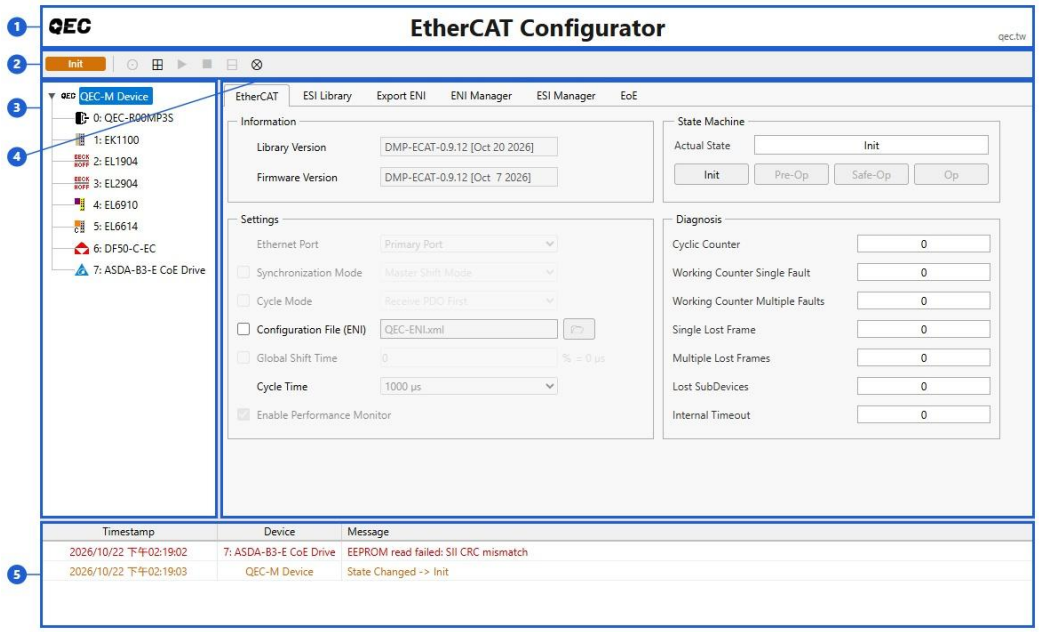


圖 11-12　Configurator 畫面：① 標題 ② 主工具列 ③ 裝置樹 ④ 分頁內容 ⑤ 記錄

區域	內容
① 標題	QEC 標誌與標題
② 主工具列	主站狀態與六個動作按鈕：Scan、Configure、Start、Stop、Deconfigure、Disconnect
③ 裝置樹	主站（QEC-M Device）與掃描到的從站
④ 分頁內容	裝置樹中選取節點的設定分頁（分頁依節點種類而不同）；分頁太多時以滑鼠滾輪左右捲動，或按右側 ▾ 列出全部分頁
⑤ 記錄	操作紀錄與系統事件（11.3.7）

11.3.3 主工具列與狀態

工具列左側的狀態標籤顯示主站目前的狀態：**Idle**、**Init**、**Config**、**Run**、**Bootstrap**。每個按鈕只有在對應的狀態下才能按，其餘時候為灰色。



圖 11-13　狀態標籤 Idle（尚未掃描）

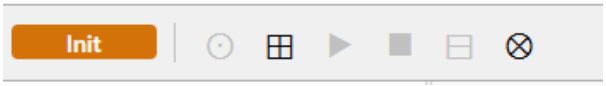


圖 11-13-1　狀態標籤 Init



圖 11-14　狀態標籤 Config（Configure 完成）



圖 11-15　狀態標籤 Run（週期交換中）

按鈕	圖示	說明
Scan	⊙	掃描匯流排上的所有從站，在裝置樹建立節點
Configure	⊞	依 ENI 檔（或線上組態）設定各從站，讀取 PDO 資料
Start	▶	啟動週期性的過程資料交換，從站進入 Op
Stop	■	停止週期性交換，回到 Safe-Op
Deconfigure	⊟	還原組態，釋放 CiA 402 軸物件，回到 Init
Disconnect	⊗	中斷與匯流排的連線，清空裝置樹

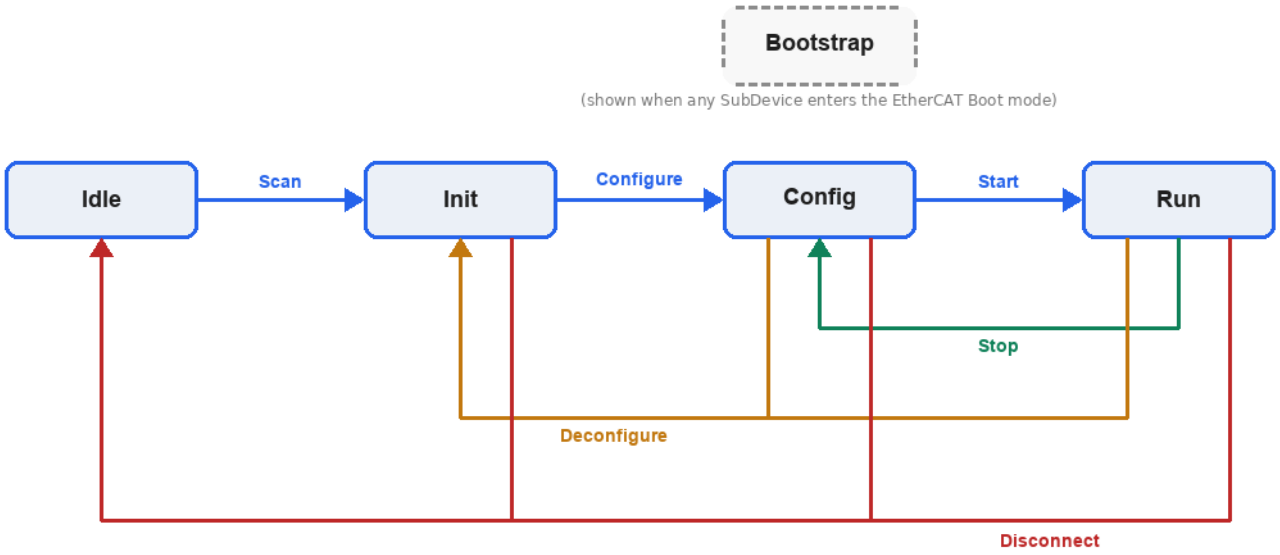


圖 11-16　主站狀態轉換：方框為狀態，箭頭為按下的按鈕

在裝置樹的 **QEC-M Device** 上按右鍵，也有同樣六個動作。按下動作後出現進度視窗（例如 Scan Network），顯示 `Step:` 目前步驟 `/` 總步驟、進度列與正在處理的裝置，完成（或失敗）後自動關閉。

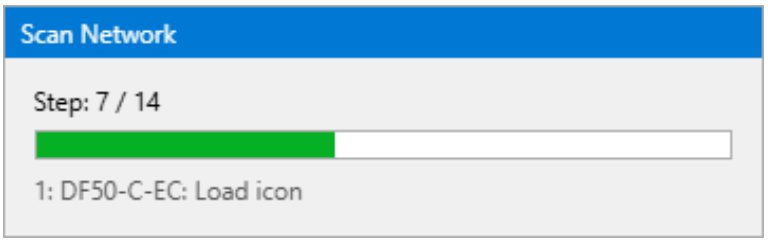


圖 11-17　進度視窗

11.3.4　裝置樹

- Scan 之前只有一個節點 **QEC-M Device**（主站）。
- Scan 成功後，每個從站一個節點，命名為 `編號: 從站名稱`，例如 `0: QEC-R11CFFG`、`1: QEC-R11MP3S`。
- 點選節點，中間顯示它的設定分頁；在從站上按右鍵 > Rename 可改名。

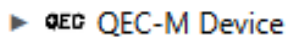


圖 11-18　Scan 之前的裝置樹：只有 QEC-M Device

11.3.5　QEC-M Device（主站）的分頁

選取 **QEC-M Device** 時，分頁有 EtherCAT、ESI Library、Export ENI、ENI Manager、ESI Manager；有從站支援 EoE 時多一個 EoE，啟用效能監看且在 Run 時多一個 Performance。

EtherCAT 分頁：左半為 Information（函式庫與韌體版本）與 Settings，右半為 State Machine 與 Diagnosis（診斷計數）。

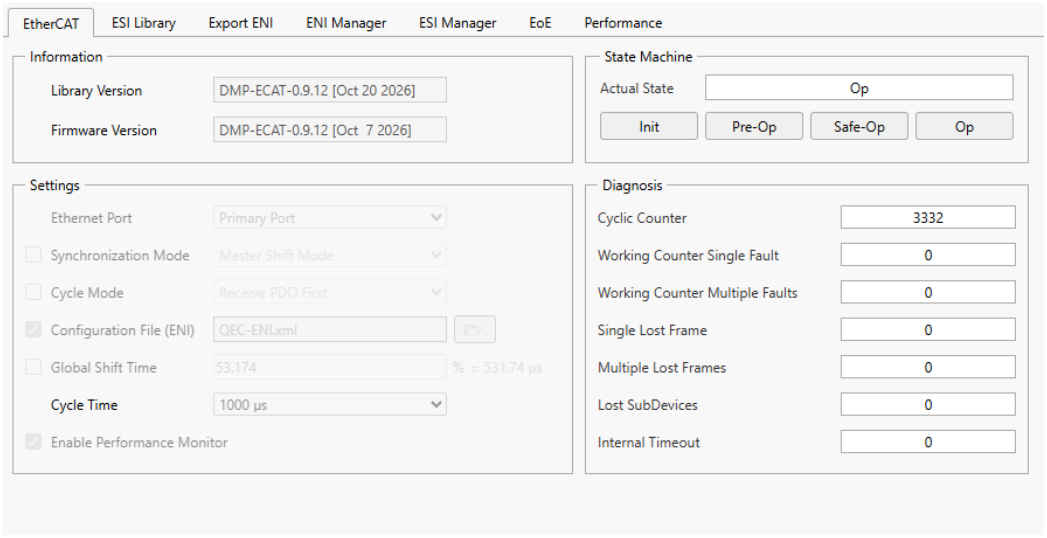


圖 11-19　QEC-M Device 的 EtherCAT 分頁：設定、狀態機與診斷計數

設定	說明	何時可改
Ethernet Port	Primary Port / Secondary Port / Cable Redundancy	Idle
Synchronization Mode	Master Shift Mode / Bus Shift Mode（Scan 時使用）	Idle
Cycle Mode	Receive PDO First / Send PDO First	Idle
Configuration File (ENI)	勾選後輸入路徑或按 🗁 選控制器上的 ENI；Configure 時使用	Init
Global Shift Time	以百分比輸入，旁邊顯示換算的 µs（Start 時使用）	Config
Cycle Time	125 ~ 10000 µs	Config
Enable Performance Monitor	勾選後，Run 時出現 Performance 分頁	Idle

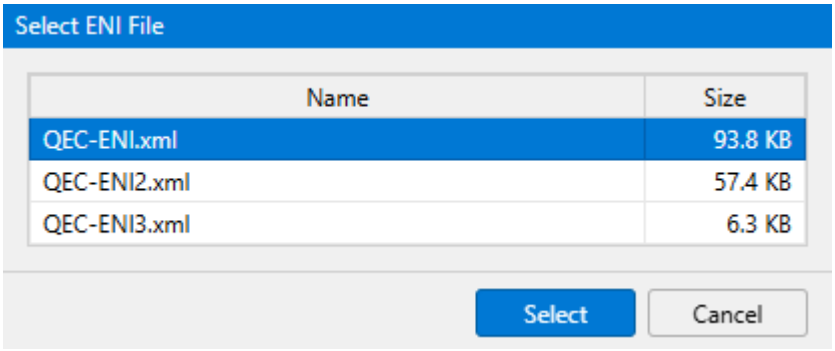


圖 11-20　Select ENI File：選擇控制器上的 ENI 檔

State Machine 的 `Init` / `Pre-Op` / `Safe-Op` / `Op` 按鈕可強制所有從站進入該狀態；Diagnosis 顯示 Cyclic Counter、Working Counter 錯誤、遺失的訊框、遺失的從站等計數。

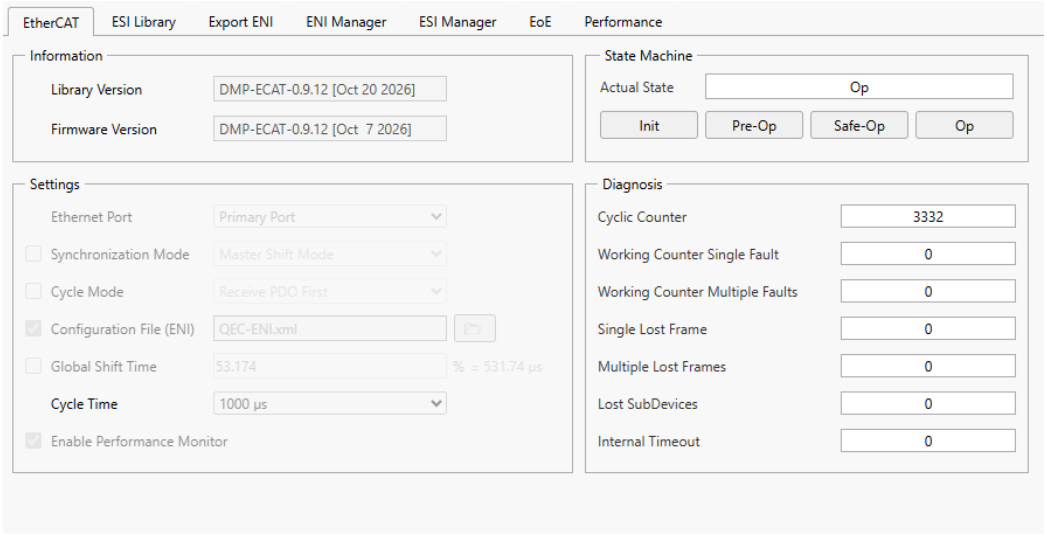


圖 11-20-1　QEC-M Device 的 EtherCAT 分頁：設定、狀態機與診斷計數

ESI Manager 與 ESI Library 分頁：ESI 載入後，Scan 時可以取得物件字典、從站圖示、Startup 指令與 Slots 模組定義（建議載入）。控制器開機時**不會**自動載入 ESI。

1. **ESI Manager** 分頁：在 Upload to Remote 按 **Browse** 選本機的 ESI 檔（.xml），按 **Upload** 上傳到控制器。上傳時顯示進度視窗。
2. **ESI Library** 分頁：在 ESI Tree 空白處按右鍵 > **Load ESI**（Idle 狀態）。載入時顯示 Loading ESI 視窗。
3. 樹狀結構為 Vendor → Group → Device（模組為 Vendor → Modules → Module）；點選 Device 可看 Product Code、PDO 數、Slots、Mailbox、DS402 Channels 等。

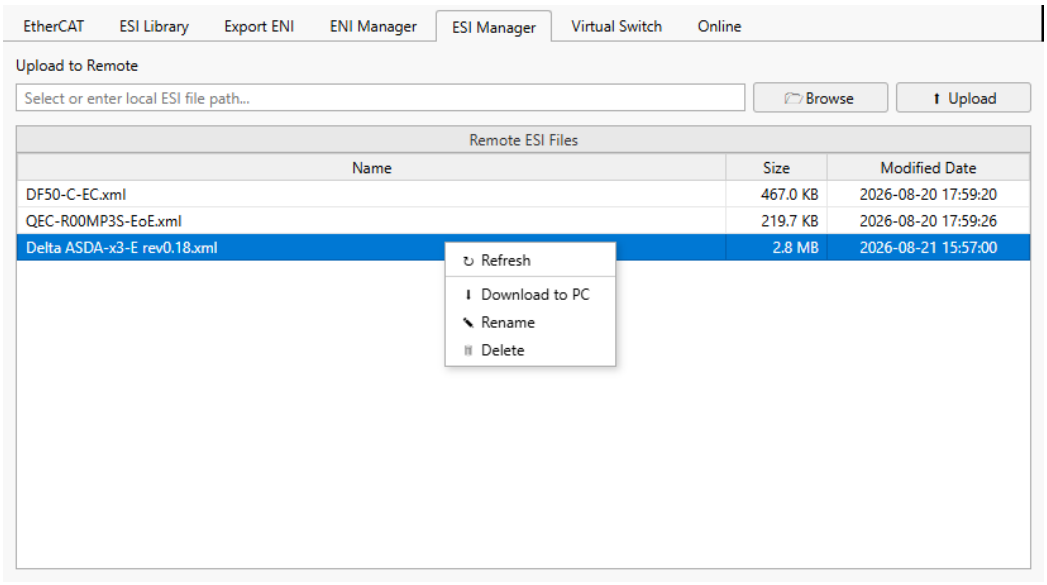


圖 11-21　ESI Manager：控制器上的 ESI 檔

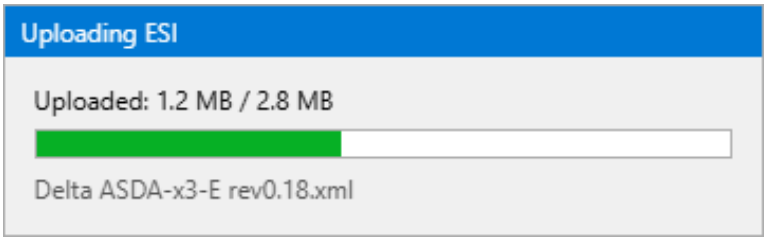


圖 11-22　上傳進度視窗

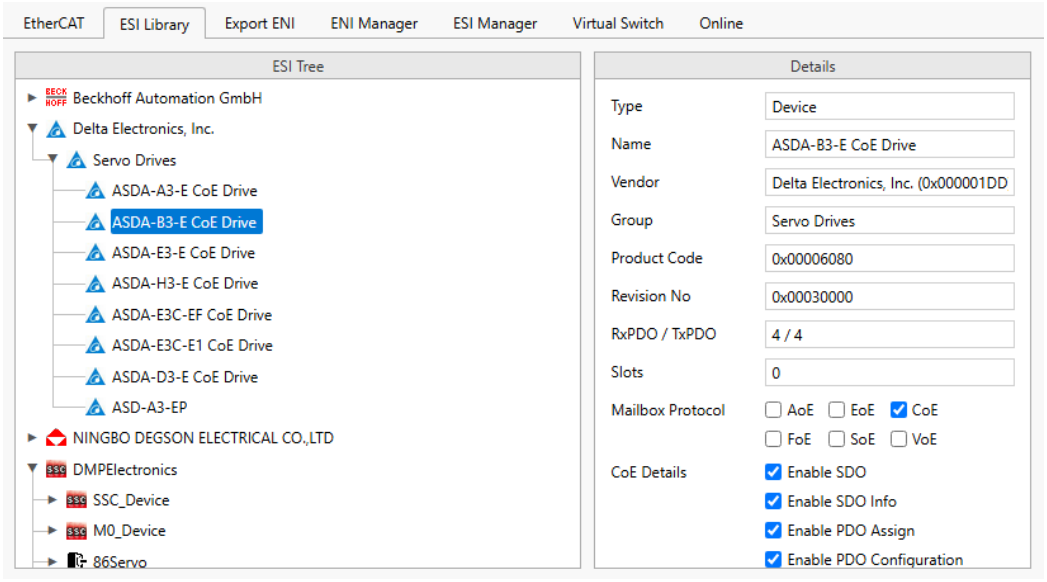


圖 11-23　ESI Library：已載入的 ESI

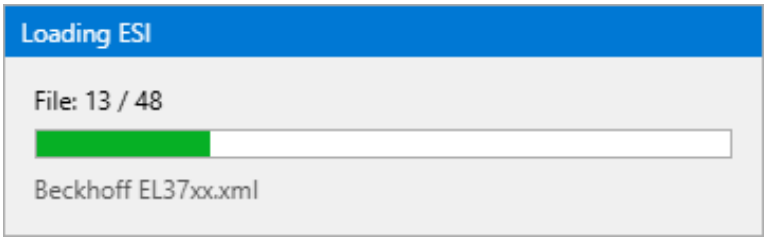


圖 11-24　Loading ESI：載入進度

*提示：在 ESI Manager 上傳、改名或刪除 ESI 後，要到 ESI Library 先 **Unload ESI** 再 **Load ESI** 才會生效。

Export ENI 分頁：依目前的組態產生 ENI，上傳到控制器（11.2.4）。

按鈕	說明
Open（🗁）	選控制器上的 ENI 檔並顯示
Export（⚙）	依目前的裝置樹資料（Slots、Process Data、Startup、DC 等）產生完整的 ENI XML
Save As...（⭳）	把顯示的 XML 存成電腦上的 `QEC-ENI.xml`
Upload（⭱）	輸入檔名上傳到控制器（`C:/eni/`）
Copy（⧉）	複製顯示的 XML

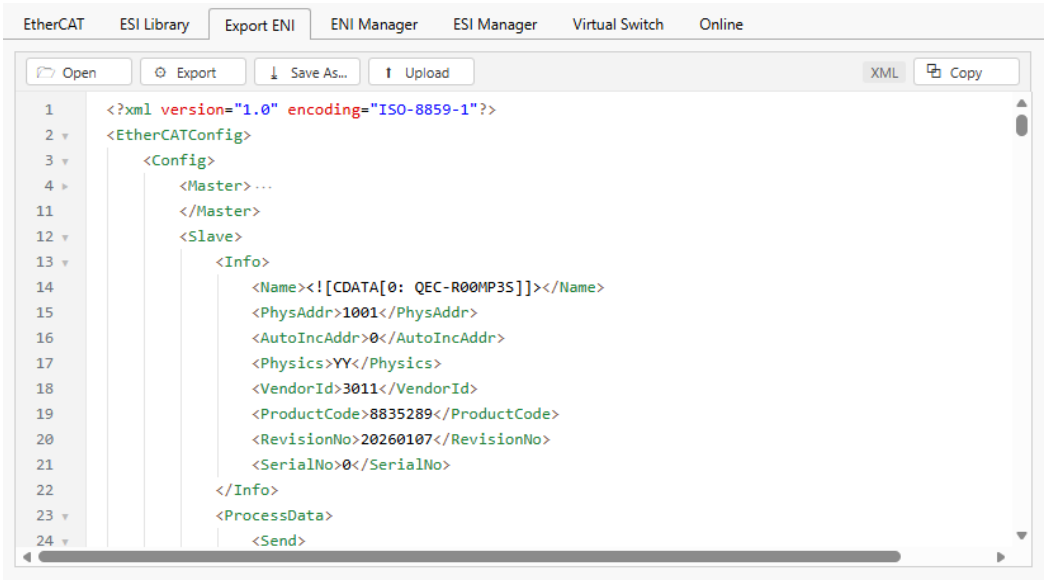


圖 11-25　Export ENI 分頁：產生的 ENI XML

ENI Manager 分頁：管理控制器上的 ENI 檔：表格可依名稱、大小、日期排序；右鍵 Refresh、Download to PC、Rename、Delete。

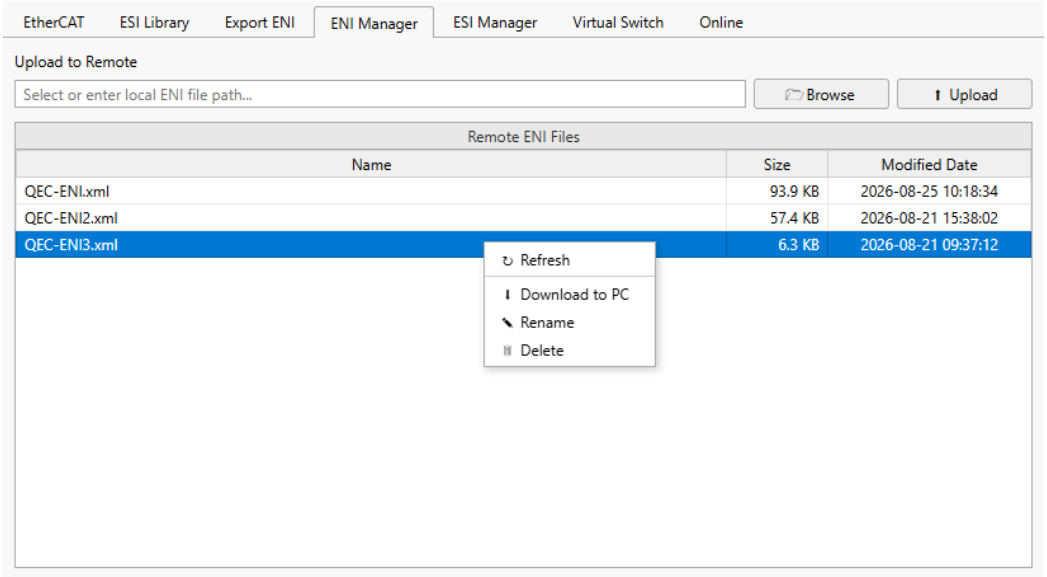


圖 11-26　ENI Manager：控制器上的 ENI 檔

EoE 分頁：有從站支援 EoE（Ethernet over EtherCAT）時出現：勾選加入虛擬交換器，設定 MAC、IP、子網路遮罩、閘道與 DNS，按 Set 寫入（Config / Run 狀態）。

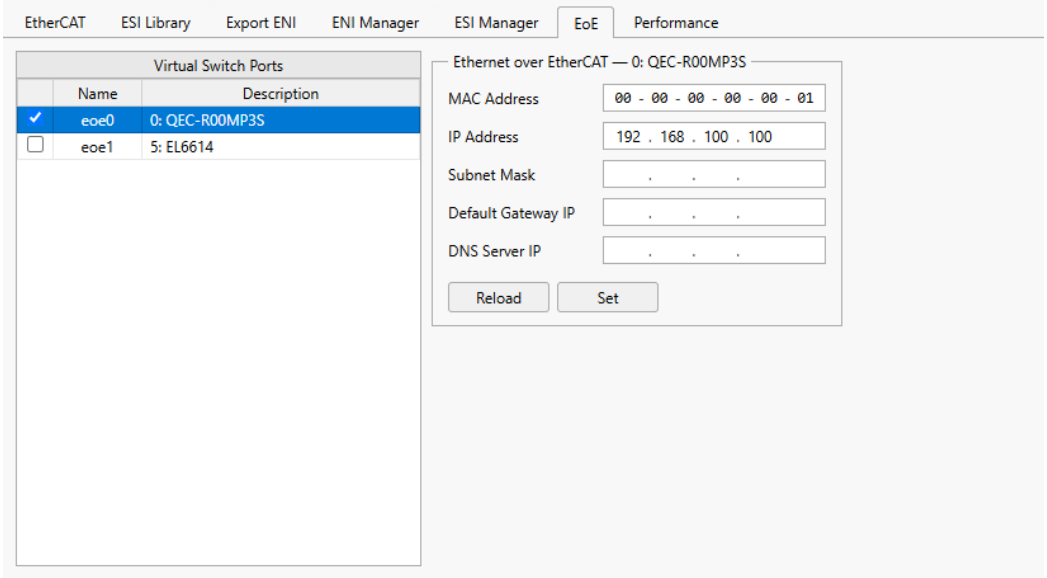


圖 11-27　EoE 分頁：虛擬交換器與從站的網路設定

Performance 分頁：在 EtherCAT 分頁勾選 Enable Performance Monitor 且主站在 Run 時出現，每 500 ms 更新：每個週期各階段（Receive PDO、User Application、Send PDO、Process Acyclic）的最小 / 平均 / 最大時間、實際週期與 Callback Jitter，以及週期時間軸與最近約 60 秒的抖動圖。

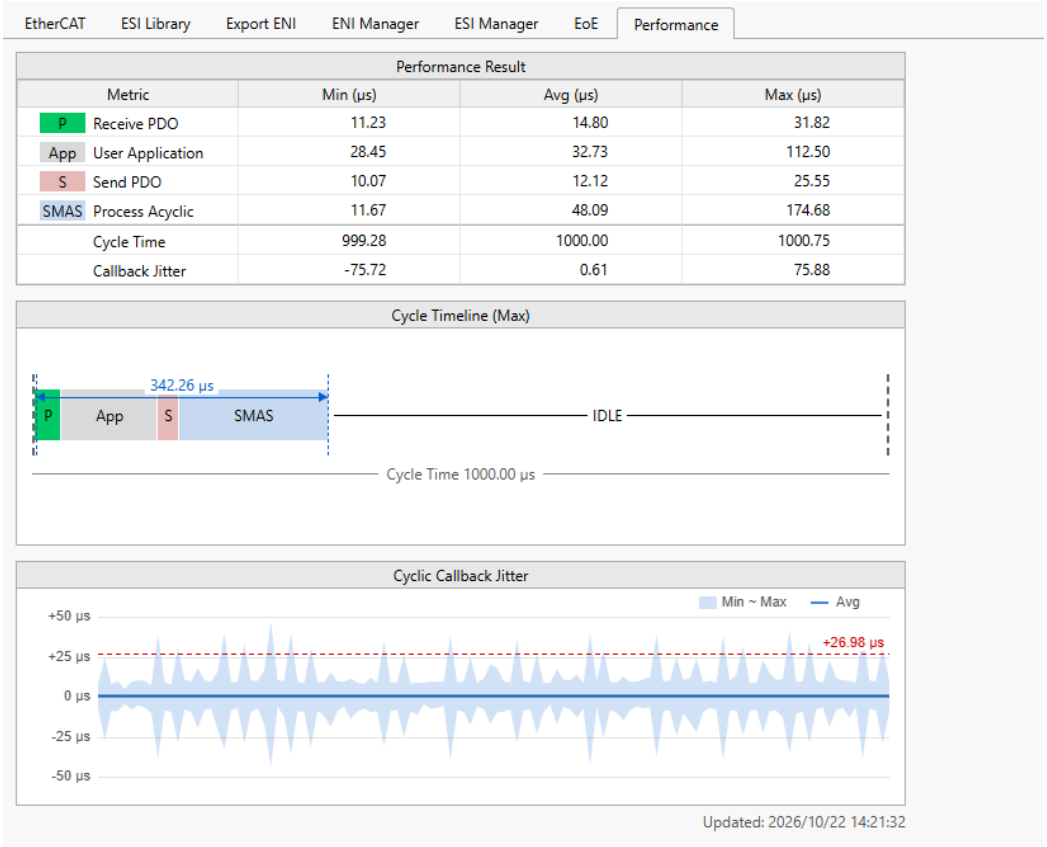


圖 11-28　Performance 分頁：週期各階段的時間與抖動

11.3.6 從站（SubDevice）的分頁

選取從站時，分頁有 EtherCAT、DC、E²PROM、ESC Access、Online；依從站另有 Slots（ESI 有定義 Slots）、Process Data / Startup / CoE（支援 CoE）、CiA 402（支援 CiA 402 且主站在 Config / Run）。

分頁	用途
EtherCAT	從站資訊（名稱、Vendor ID、Product Code、Revision、Serial、DS402 Channels、對應的 ESI 檔）、Mailbox 協定、狀態機（Init / Pre-Op / Safe-Op / Op、Bootstrap）、FoE 檔案讀寫
DC	分散式時鐘：Enable SYNC0、SYNC0 週期倍率與 Shift Time、SYNC1；Start 時寫入（使用 ENI 時以 ENI 的設定為準）
Slots	模組化從站：在 Slot 按右鍵 Insert 選模組，系統自動產生 Startup 指令
Process Data	PDO 映射：勾選要交換的 RxPDO（0x1C12）/ TxPDO（0x1C13），右鍵 Insert / Delete / Move / Edit；右邊為選取 PDO 的內容
Startup	Pre-Op → Safe-Op 時寫入從站的 SDO 指令（Index、CA、Data、Comment）；未使用 ENI 時 Start 寫入，也會輸出到匯出的 ENI
CoE	物件字典：Refresh、Upload、Download、Download (CA)；來源可切換 Load from ESI / Load from SDO Info
E²PROM	Station Alias；從 ESI 產生、讀取、寫入 EEPROM；Hex 編輯器
ESC Access	ESC 暫存器（進階）：讀取、寫入、位元欄位解碼
Online	Run 時每 200 ms 更新的 PDO 即時值；右鍵可修改 RxPDO（輸出）
CiA 402	驅動器測試：選軸、狀態機 Enable / Disable、伺服設定、Mode of Operation、實際值、各模式的操作面板

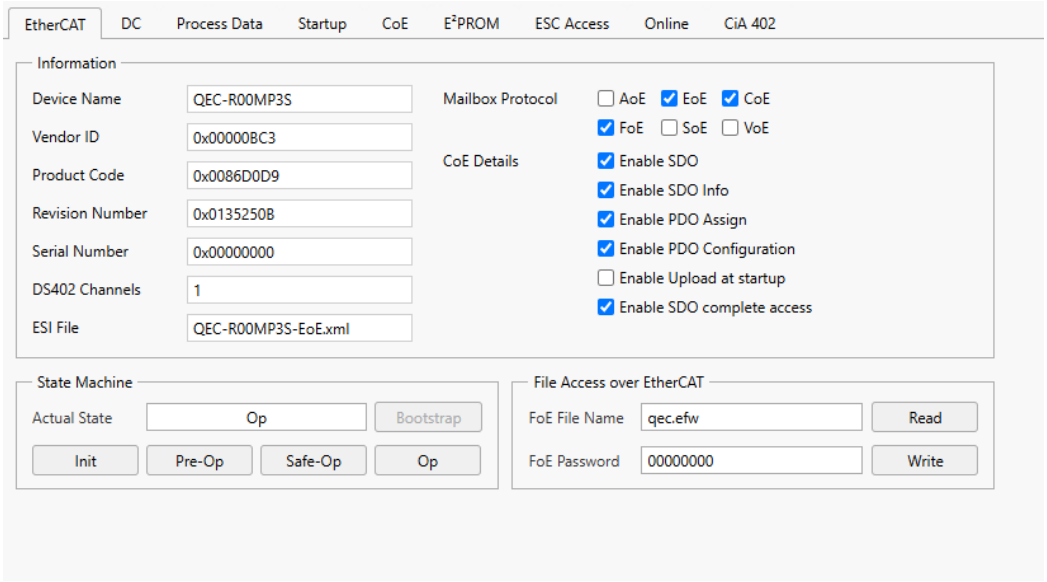


圖 11-29　從站的 EtherCAT 分頁：資訊、狀態機與 FoE

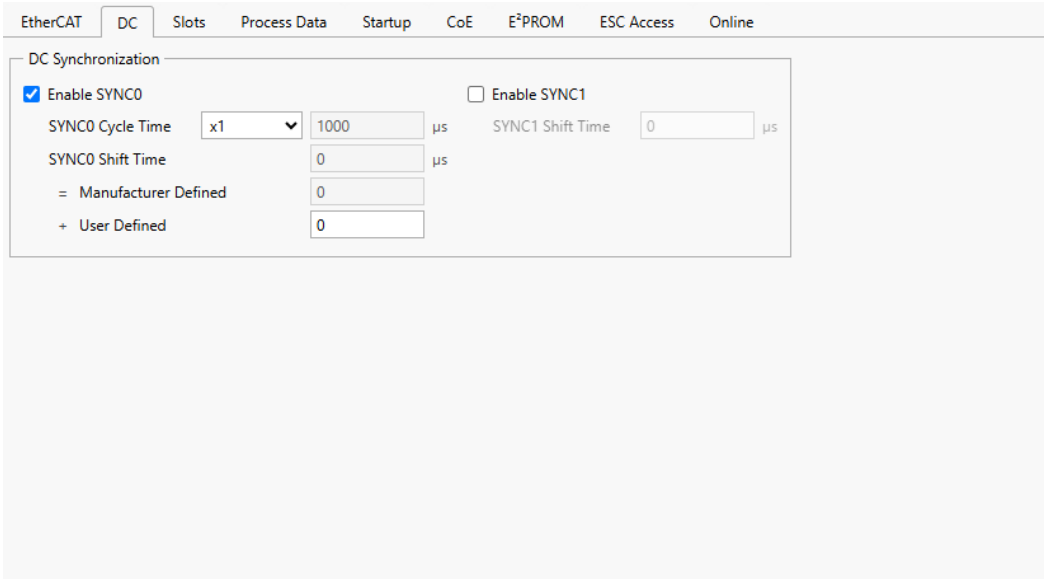


圖 11-30　DC 分頁：SYNC0 / SYNC1 設定

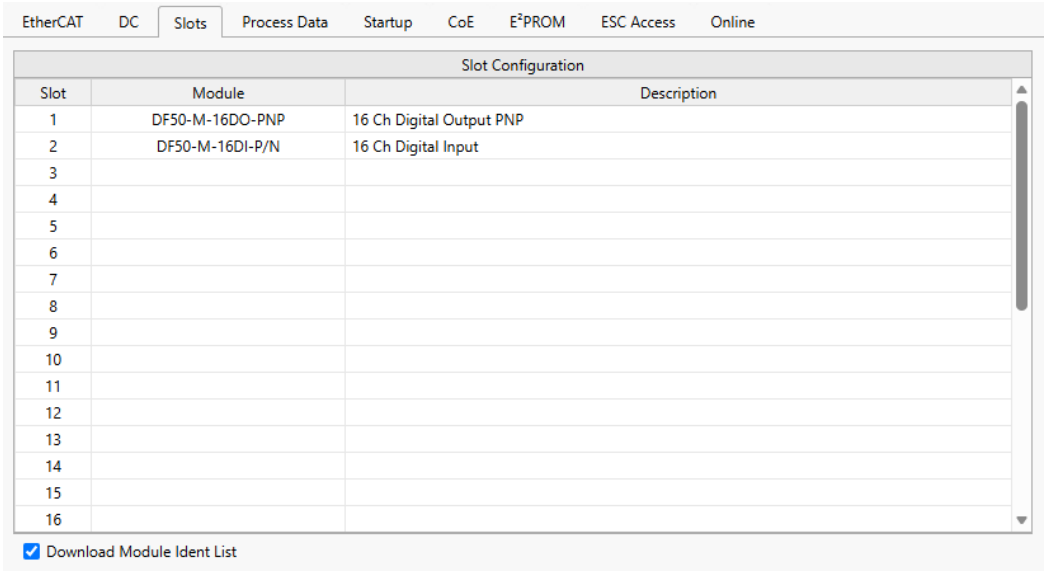


圖 11-31　Slots 分頁：模組化從站的 Slot 與模組

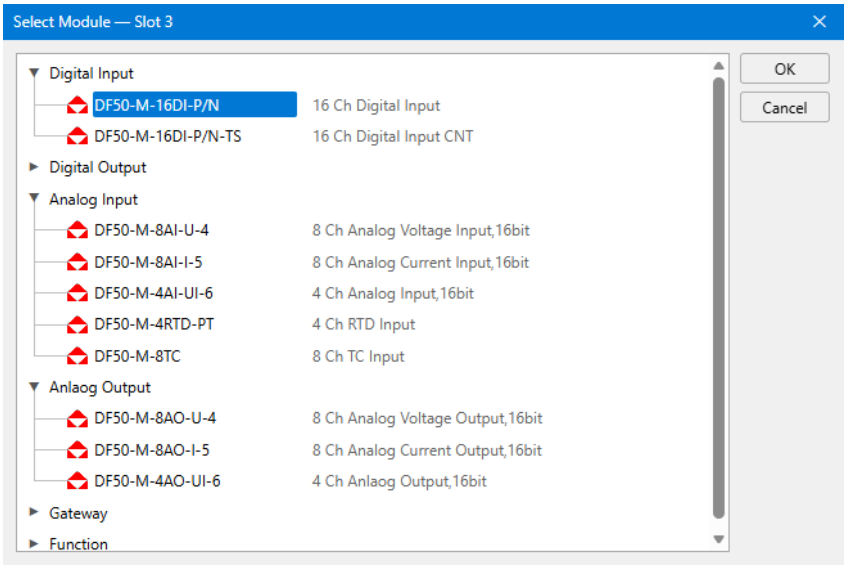


圖 11-32　Select Module：選擇要插入 Slot 的模組

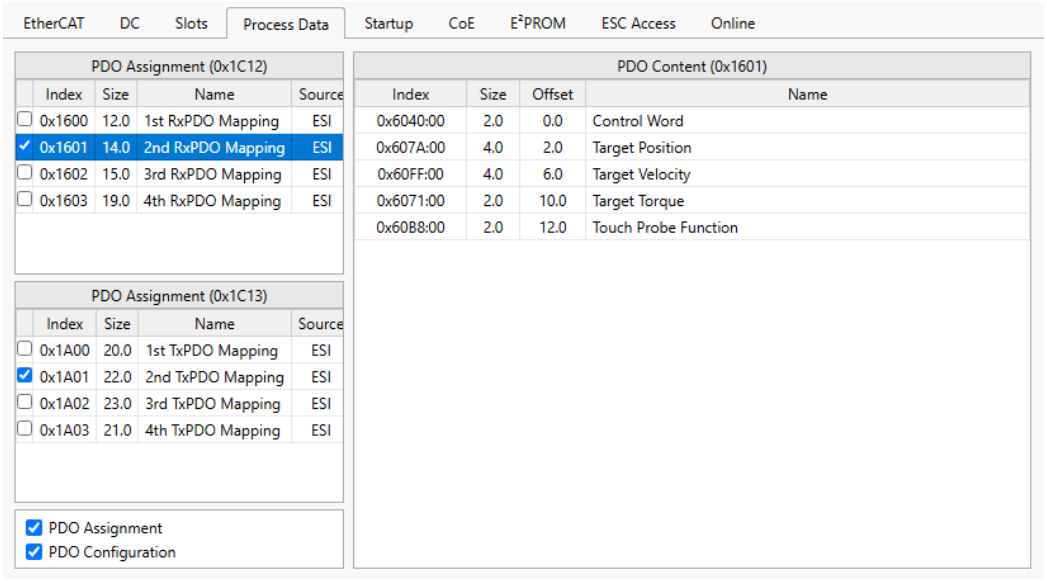


圖 11-33　Process Data 分頁：左邊為 PDO 指派，右邊為選取 PDO 的映射內容

Process Data 分頁的 `Source` 欄說明 PDO 的來源：`ESI`（載入的 ESI 或 Slots 插入的模組）、`SII`（從站 EEPROM）、`SDO`（Configure 時以 SDO 讀取）、`User`（自己新增）。固定或必要的項目受保護，不能刪除。

*__重要：__使用 Motion86 的 CSP 模式時（第 12 章），CiA 402 驅動器的 PDO 必須包含：輸出 6040h（Controlword）、607Ah（Target position），輸入 6041h（Statusword）、6064h（Position actual value）。多軸驅動器的第 n 軸位址加 800h×(n−1)。

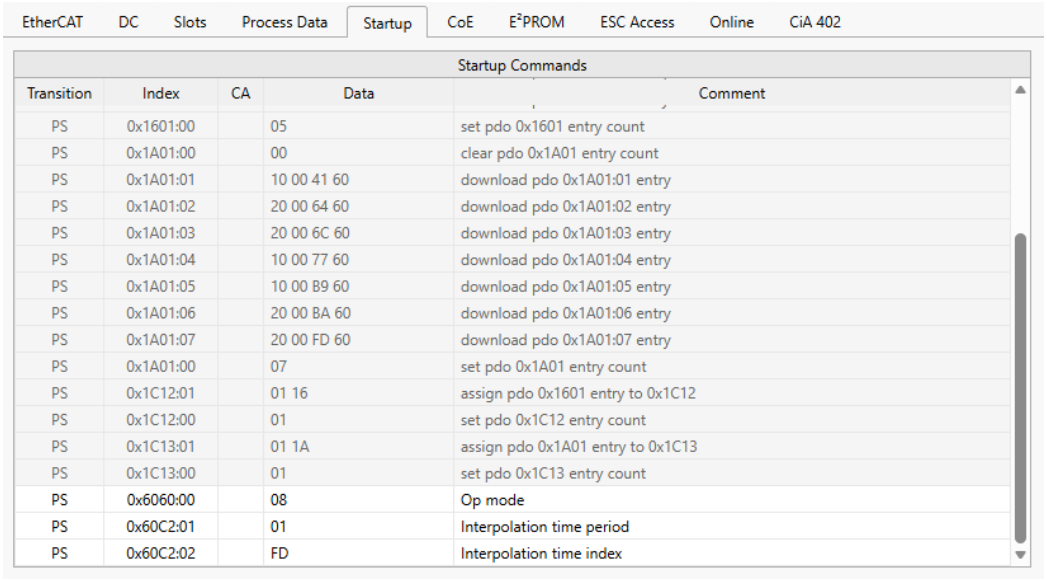


圖 11-34　Startup 分頁：啟動時寫入的 SDO 指令

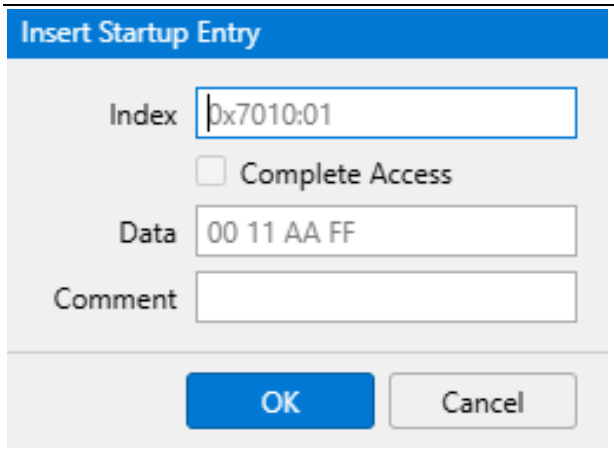


圖 11-35　Insert/Edit Startup Entry：新增一筆 Startup 指令

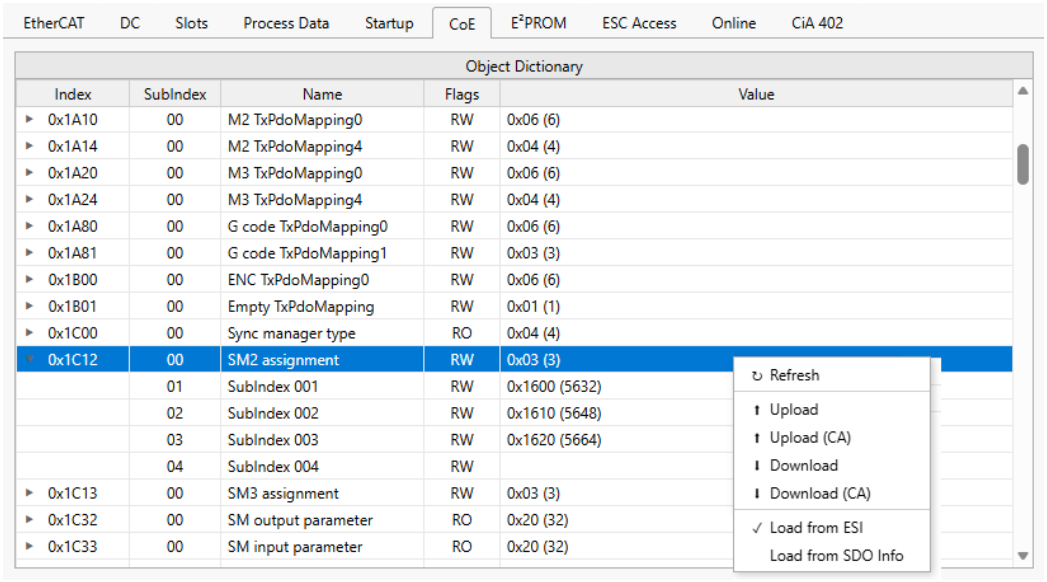


圖 11-36　CoE 分頁：物件字典

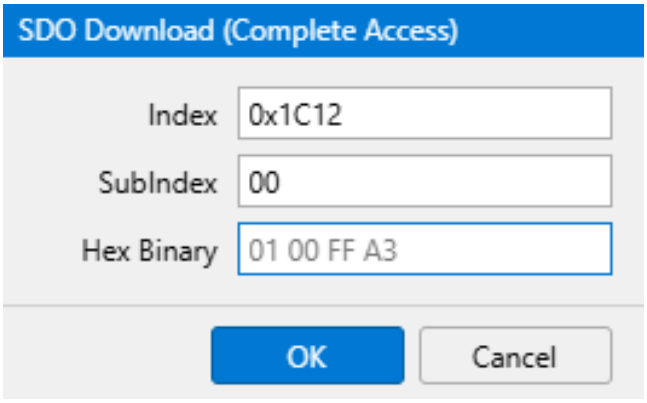


圖 11-37　SDO Download (Complete Access)：以 16 進位寫入整個物件

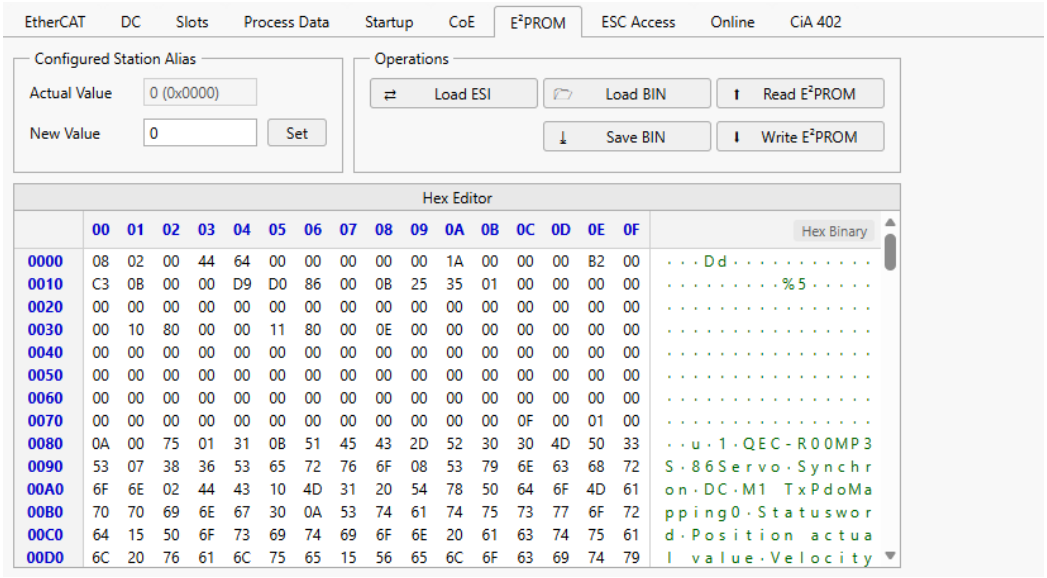


圖 11-38　E²PROM 分頁：Station Alias、EEPROM 操作與 Hex 編輯器

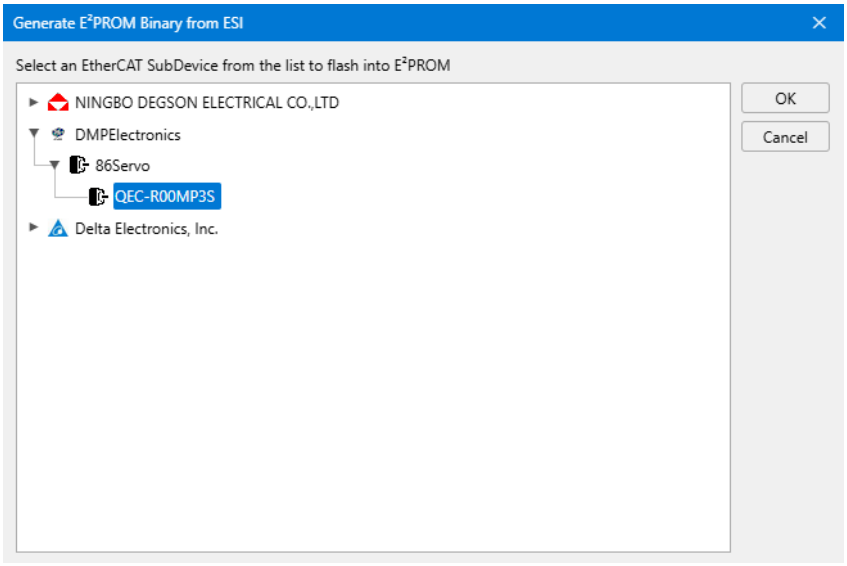


圖 11-39　Generate E²PROM Binary from ESI：由 ESI 產生 EEPROM 內容

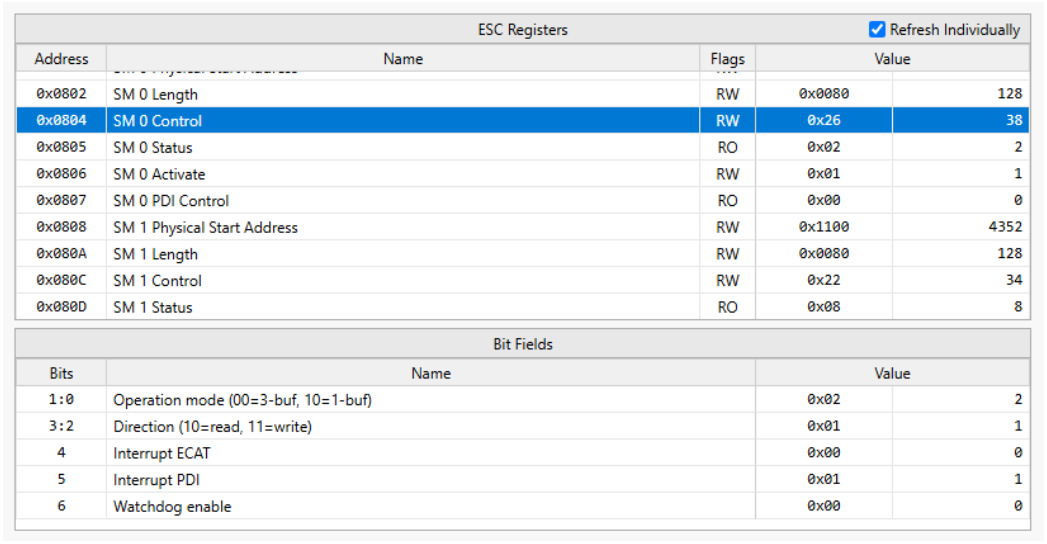


圖 11-40　ESC Access 分頁：ESC 暫存器

CiA 402 分頁：不寫程式就能測試驅動器。建議順序：選軸 → 設定 Mode of Operation → 按 **Enable** 激磁 → 在該模式的面板輸入參數按 **Run** → 按面板的 **Stop** 或直接 **Disable**。

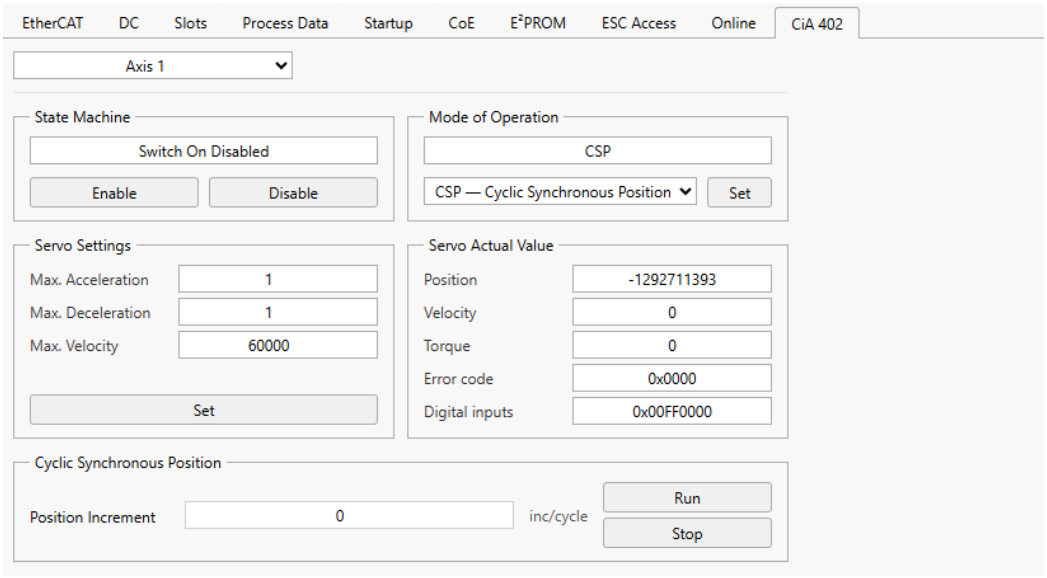


圖 11-41　CiA 402 分頁：狀態機、伺服設定、模式與實際值

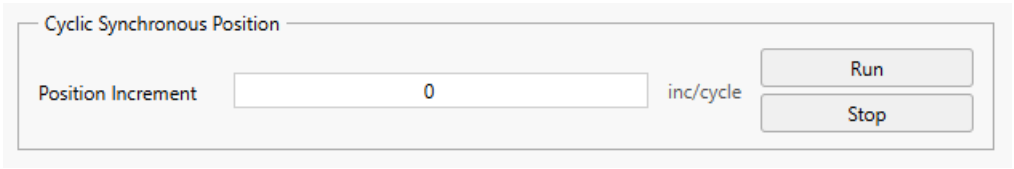


圖 11-42　CiA 402 分頁的 Cyclic Synchronous Position（CSP）面板

模式	名稱
PP	Profile Position
PV	Profile Velocity
TQ	Profile Torque
HM	Homing
CSP	Cyclic Synchronous Position（Motion86 使用此模式）
CSV	Cyclic Synchronous Velocity
CST	Cyclic Synchronous Torque

*安全：CiA 402 分頁會直接讓馬達轉動。測試前請確認機構可自由移動、緊急停止有效，先用低速。

11.3.7　記錄面板

畫面下方的記錄面板列出 Timestamp、Device、Message：每次操作的成功或失敗，以及主站不在 Idle 時每 500 ms 讀回的系統事件。點選列（Shift 複選、Ctrl+A 全選），右鍵 Copy 或 Clear。

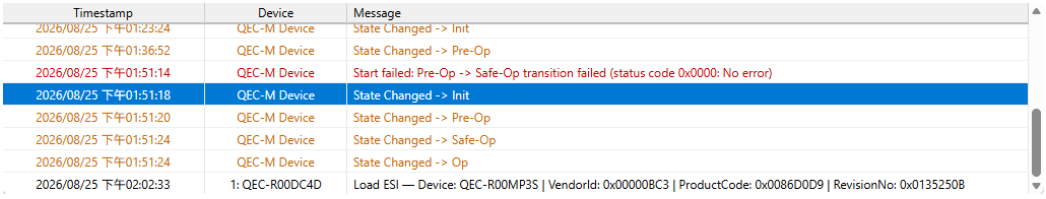


圖 11-43　記錄面板

11.3.8　設定步驟摘要

1. （建議）ESI Manager 上傳 ESI，ESI Library 按右鍵 Load ESI。
2. QEC-M Device 的 EtherCAT 分頁確認 Ethernet Port、Synchronization Mode、Cycle Mode。
3. 按 **Scan**，等待裝置樹列出所有從站。
4. （選用）要使用既有 ENI：EtherCAT 分頁勾選 Configuration File (ENI) 並選檔；否則直接在各從站的 Slots / Process Data / Startup 設定。
5. 按 **Configure**，確認各從站的 PDO、CoE 正確載入；視需要調整 Cycle Time、Global Shift Time 與 DC。
6. 按 **Start**，在 Online 分頁確認數值；CiA 402 驅動器可在 CiA 402 分頁測試。
7. Export ENI → **Upload**（例如 `QEC-ENI-1.xml`）。
8. 依序 **Stop** → **Deconfigure**。

11.4 讀入 ENI 與對應變數

11.4.1 讀入 ENI

- **自動**：EtherCAT 頁偵測到控制器上有新的或更新的 ENI 時，提示「控制器有新的 ENI：…。要讀入專案（從站與過程資料）嗎？」，按 **讀入專案**。
- **讀回 ENI...**：從控制器讀取（也在「工具 > EtherCAT > 從控制器讀取 ENI...」）。
- **匯入 ENI...**：選電腦上的 ENI 檔（例如 Configurator 的 Save As... 存的檔案）。
- 命令列：`86ldmicro-cli eni <專案> <eni.xml> [控制器上的名稱]`（15.5 節）。

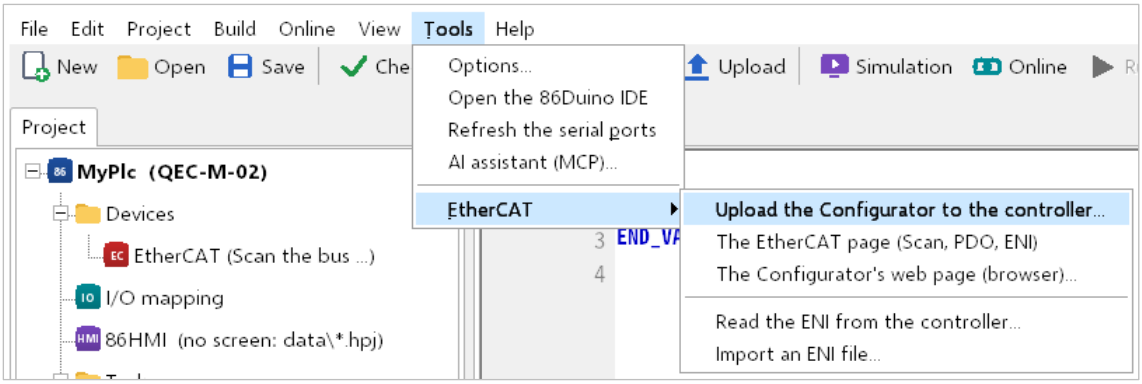


圖 11-44　「工具 > EtherCAT」子選單

讀入後，從站與每個 PDO 項目出現在專案樹「裝置 > EtherCAT」下，ENI 的副本存在 `data\86ld\ethercat\`。CiA 402 驅動器以馬達圖示表示，多軸驅動器（例如 QEC-R11MP3S）每軸一個節點。

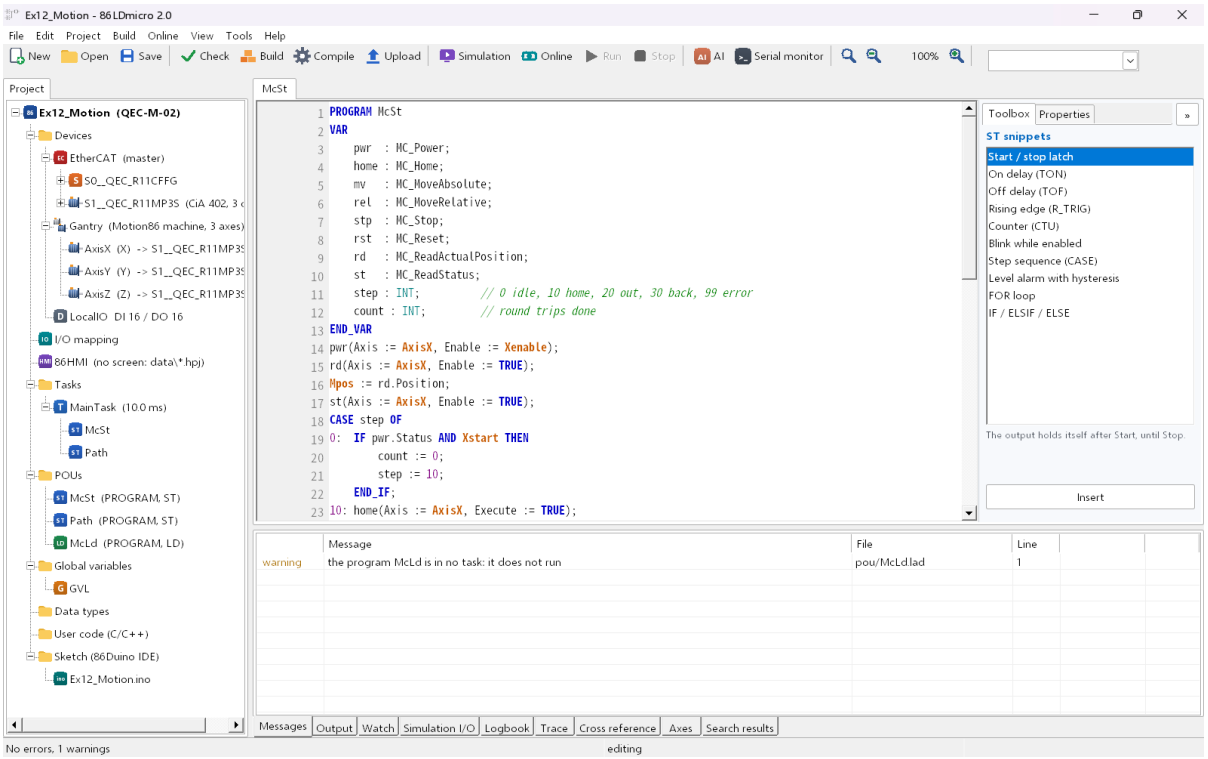


圖 11-45　讀入 ENI 後的專案樹：S0__QEC_R11CFFG（I/O 模組）、S1__QEC_R11MP3S（3 軸驅動器）

- 從站名稱格式為 `S<位置>__<型號>`，例如 `S0__QEC_R11CFFG`。
- 同一從站有同名的 PDO 項目時（例如多軸驅動器），第 2、3 個自動改名為「名稱 #2」「名稱 #3」，也可以用 Index（例如 `6864:00`）指定。
- PDO 有 6040h / 6041h 的從站自動視為 CiA 402 驅動器，不限廠商；也可在 SubDevice 頁的 **Profile（CiA 402）...** 手動指定（11.5 節）。
- 重新讀入 ENI 時，已建立的對應、機台與手動指定的 Profile 會保留。

11.4.2 把 PDO 對應到變數

PDO 項目要對應到變數，程式才能讀寫。三種做法：

- EtherCAT 頁按 **變數對應**：選一列按 **對應變數...**（或雙擊）；**取消對應** 解除。
- 專案樹「裝置 > EtherCAT > 從站」下的 PDO 項目雙擊。
- SubDevice 頁選一列按 **對應到變數...**（11.5 節）。

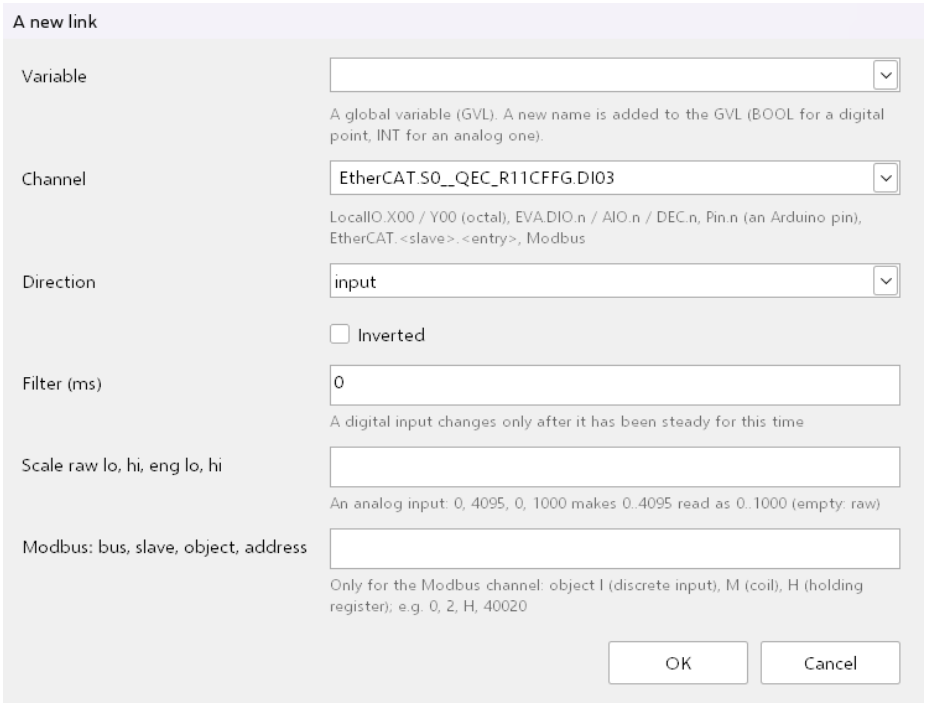
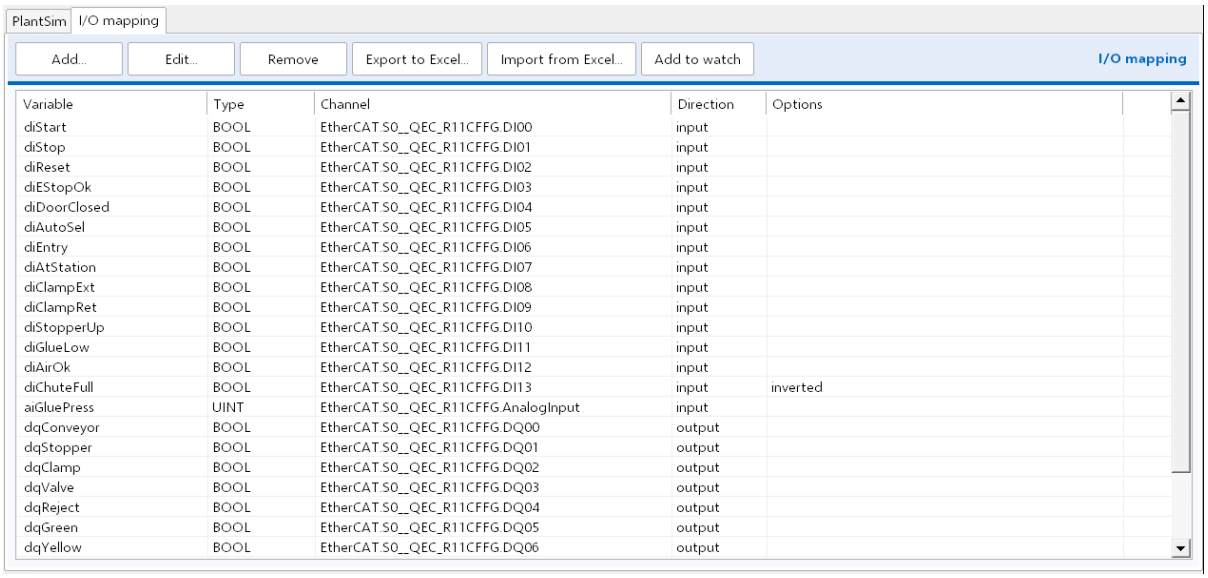


圖 11-46　對應變數：通道已帶入（例如 EtherCAT.S0__QEC_R11CFFG.DI02），輸入或選擇變數

通道名稱為 `EtherCAT.<從站>.<PDO 項目>`。輸入新的變數名稱時，變數自動加入 GVL（型別依 PDO）。對應完成後，「I/O 對應」頁也會列出這些對應（4.6 節），變數可在 LD、ST、SFC、C/C++（`GVL(名稱)`）中直接使用。



Variable	Type	Channel	Direction	Options
diStart	BOOL	EtherCAT.S0_QEC_R11CFFG.DI00	input	
diStop	BOOL	EtherCAT.S0_QEC_R11CFFG.DI01	input	
diReset	BOOL	EtherCAT.S0_QEC_R11CFFG.DI02	input	
diEStopOk	BOOL	EtherCAT.S0_QEC_R11CFFG.DI03	input	
diDoorClosed	BOOL	EtherCAT.S0_QEC_R11CFFG.DI04	input	
diAutoSel	BOOL	EtherCAT.S0_QEC_R11CFFG.DI05	input	
diEntry	BOOL	EtherCAT.S0_QEC_R11CFFG.DI06	input	
diAtStation	BOOL	EtherCAT.S0_QEC_R11CFFG.DI07	input	
diClampExt	BOOL	EtherCAT.S0_QEC_R11CFFG.DI08	input	
diClampRet	BOOL	EtherCAT.S0_QEC_R11CFFG.DI09	input	
diStopperUp	BOOL	EtherCAT.S0_QEC_R11CFFG.DI10	input	
diGlueLow	BOOL	EtherCAT.S0_QEC_R11CFFG.DI11	input	
diAirOk	BOOL	EtherCAT.S0_QEC_R11CFFG.DI12	input	
diChuteFull	BOOL	EtherCAT.S0_QEC_R11CFFG.DI13	input	inverted
aiGluePress	UINT	EtherCAT.S0_QEC_R11CFFG.AnalogInput	input	
dqConveyor	BOOL	EtherCAT.S0_QEC_R11CFFG.DQ00	output	
dqStopper	BOOL	EtherCAT.S0_QEC_R11CFFG.DQ01	output	
dqClamp	BOOL	EtherCAT.S0_QEC_R11CFFG.DQ02	output	
dqValve	BOOL	EtherCAT.S0_QEC_R11CFFG.DQ03	output	
dqReject	BOOL	EtherCAT.S0_QEC_R11CFFG.DQ04	output	
dqGreen	BOOL	EtherCAT.S0_QEC_R11CFFG.DQ05	output	
dqYellow	BOOL	EtherCAT.S0_QEC_R11CFFG.DQ06	output	

圖 11-47　I/O 對應頁：EtherCAT 通道與本機通道一起列出

> ***提示：**程式讀取的輸入變數若沒有對應，建置時「訊息」會警告「… 在這裡只被讀取：沒有 I/O 對應，也沒有任何地方寫入它 —— 它永遠是 FALSE」。

11.4.3 EtherCAT 裝置的設定

在專案樹「裝置 > EtherCAT」按右鍵 > 編輯（或在裝置頁選 EtherCAT 按編輯）可調整主站的執行方式：

設定	說明
mode	`auto`：有 CiA 402 伺服時用 DC 同步，否則 free run（預設）；`sync`：DC 同步（失敗時改 free run）；`freerun`
master	`plc`：由 86LDmicro 產生的程式啟動主站（預設，`plc86_ecat.cpp`）；`sketch`：由你的 .ino 自行 begin、attach、start，完成後設定 `plc86_ecOk = true`
cycleUs	EtherCAT 週期（µs），預設 1000

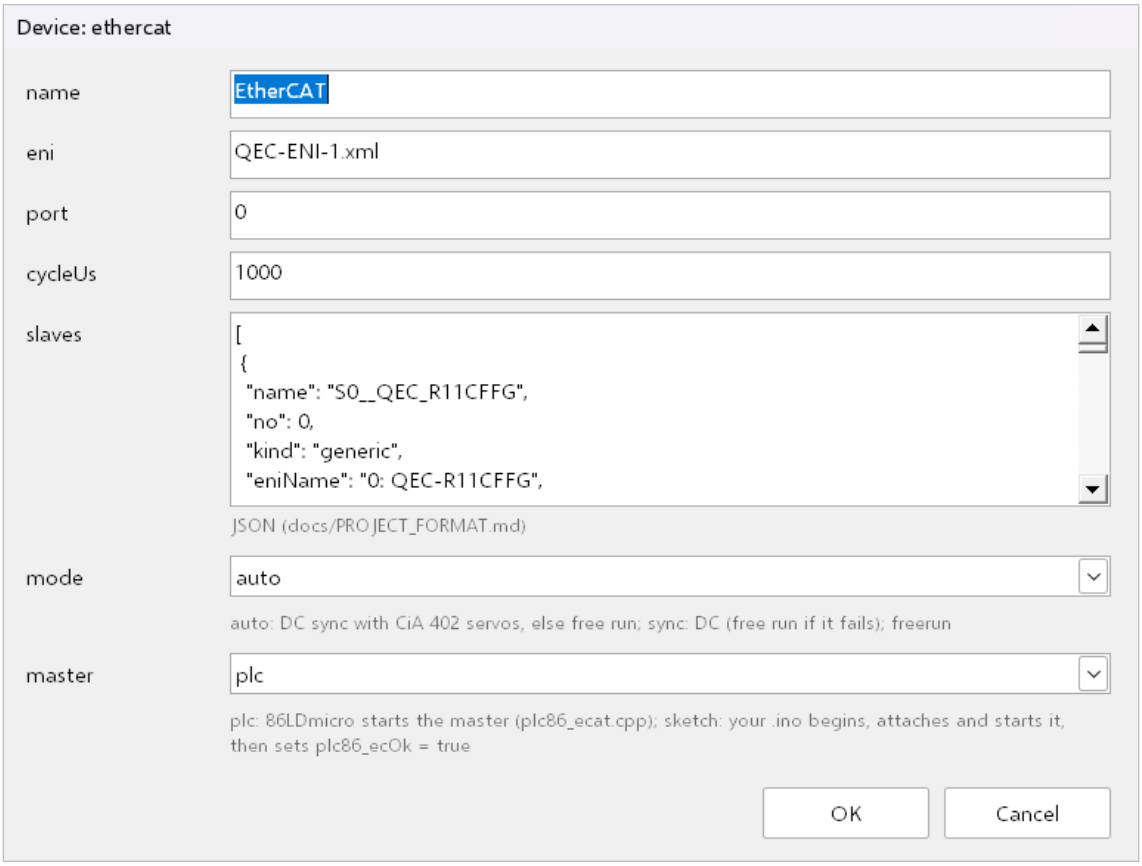


圖 11-48　EtherCAT 裝置的設定

11.4.4 建置上傳並確認主站

1. 按 **上傳**（F8）。PLC 程式取代 Configurator；開機時主站以 `C:/eni/<檔名>` 啟動，各從站 attach，驅動器設定模式，然後開始週期交換。
2. 開「序列埠監看」（不需上線，10.3 節）：主站狀態每 5 秒印一次（正常運作時只在啟動後印 3 次，失敗時持續印出），例如 `EtherCAT: running, 2 slaves, 1000 us, sync, ENI C:/eni/QEC-ENI-1.xml`，或失敗原因。
3. 上線後，「事件紀錄」也會記錄主站狀態（10.5 節）。

11.5 SubDevice 頁

在專案樹雙擊「裝置 > EtherCAT」下的從站，開啟它的 SubDevice 頁：上方顯示匯流排上的位置、ENI 中的名稱、廠商碼 / 產品碼、Profile 與程式碼中的物件名稱；下方分為「**過程資料（PDO）**」與「**CoE 物件字典（線上）**」兩個分頁。

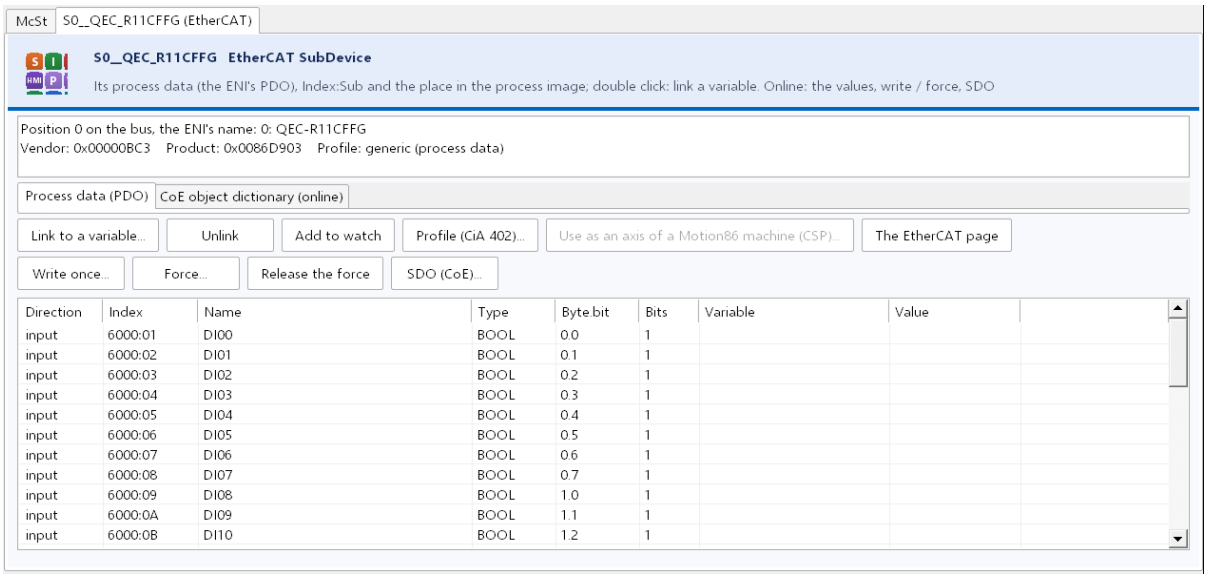


圖 11-49　QEC-R11CFFG 的 SubDevice 頁：每個 PDO 項目的方向、Index、名稱、型別、Byte.bit、位元數、變數

按鈕	說明
對應到變數... / 取消對應	把選取的 PDO 項目對應到變數或解除
加入監看	把對應的變數加入監看視窗
Profile（CiA 402）...	手動指定 Profile（自動 / CiA 402 / 一般）與驅動器數量（第三方驅動器）
作為 Motion86 機台的軸（CSP）...	把選取的驅動器加入 Motion86 機台（12.2 節）
EtherCAT 頁面	回到 EtherCAT 頁
寫入一次... / 強制... / 解除強制	線上：直接寫入或強制 PDO 輸出（11.6 節）
SDO（CoE）...	線上：讀寫單一個物件（11.6 節）

CiA 402 驅動器的 SubDevice 頁下方另外列出每個驅動器（#1、#2、#3…）對應的 Motion86 軸：

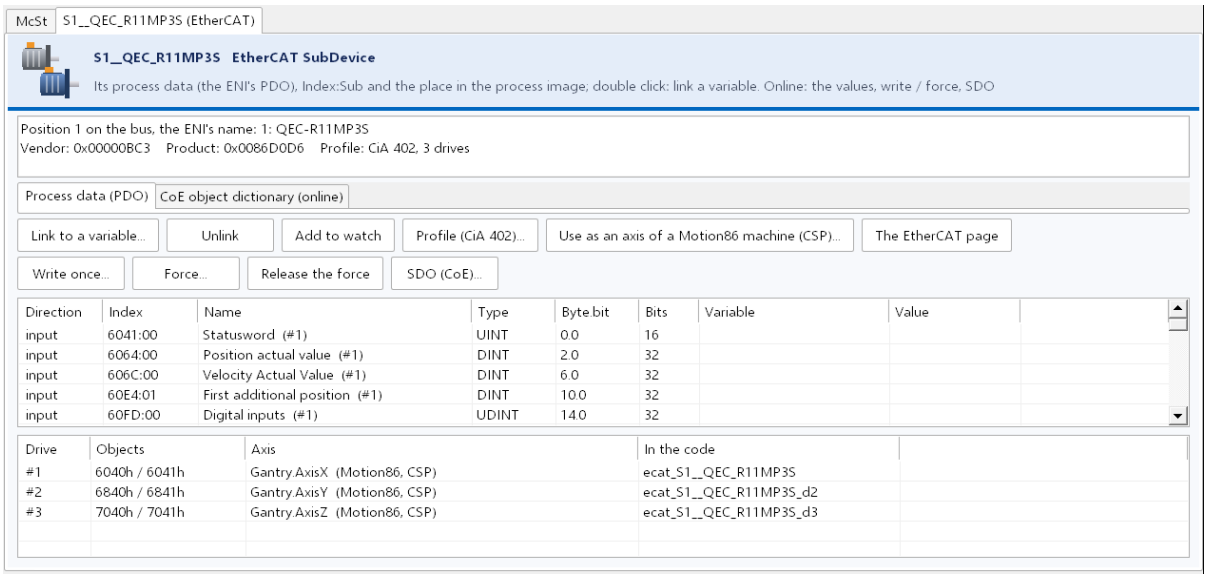


圖 11-50　QEC-R11MP3S（3 軸驅動器）：下方列出各驅動器對應的 Motion86 軸

第三方驅動器的 PDO 沒有標準的 6040h / 6041h 時，按 **Profile（CiA 402）...** 手動指定：

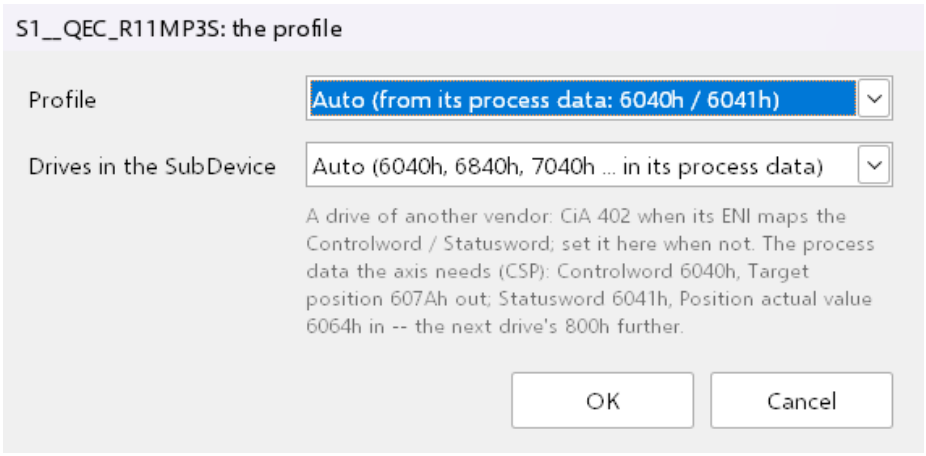


圖 11-51　Profile 設定：自動、CiA 402 或一般，以及 SubDevice 中的驅動器數

SubDevice 頁可以和程式左右並排（「檢視 > 左右並排：目前分頁移到右側 / 結束並排」，Ctrl+\，3.7 節），一邊看 PDO 值一邊看程式：

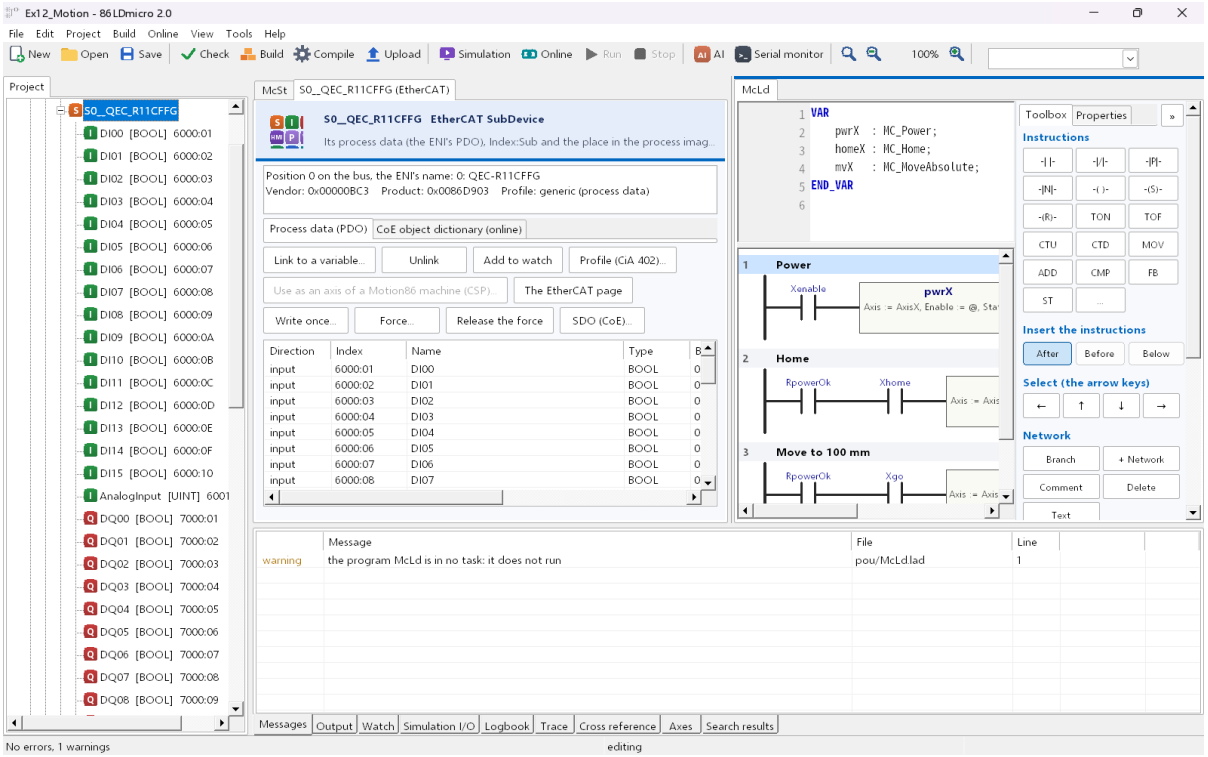


圖 11-52　左右並排：左側 SubDevice 頁、右側階梯圖

11.6 PDO 線上值、SDO 與 CoE

11.6.1 PDO 線上值

連線後（Ctrl+F5），SubDevice 頁的所有 PDO 項目（不論有無對應變數）都顯示目前值：控制器每 100 ms 更新、頁面每 0.5 秒刷新。REAL 顯示浮點值，WORD / DWORD 另附 16 進位。

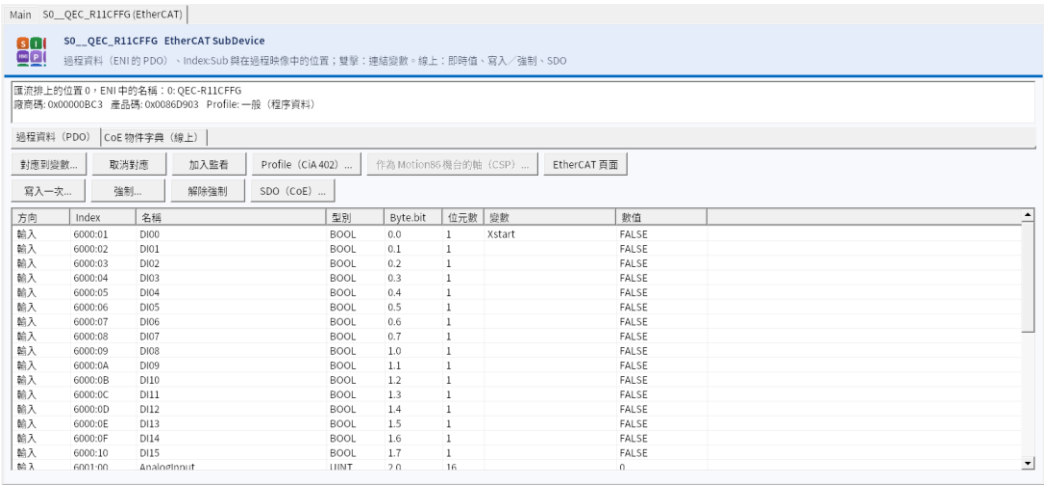


圖 11-53　線上：每個 PDO 項目顯示即時值

11.6.2 寫入一次與強制

功能	說明
寫入一次...	寫入一次後不再寫，程式仍可改變它
強制...	每個掃描都寫入，直到「解除強制」或連線中斷 3 秒；強制中顯示 `[F]`
解除強制	解除選取項目的強制

- 已對應變數的項目，寫入或強制的是**那個變數**（模擬中也可以）；未對應變數的**輸出**直接寫入過程資料（僅限線上）。
- **不可手動寫入**：輸入（由從站寫入；要測試程式請對應變數後強制該變數），以及 CiA 402 驅動器的運動物件（6040h、607Ah、60FFh、6060h 等，由軸控制）。

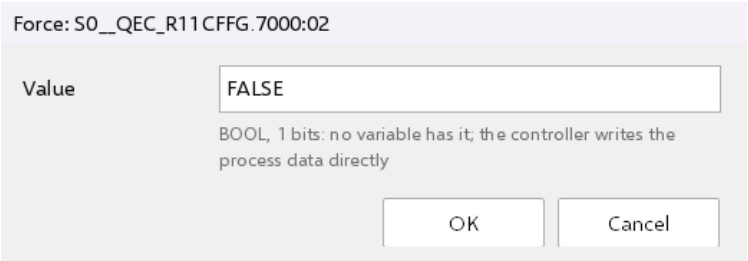


圖 11-54　強制一個未對應變數的輸出（DQ01）：由控制器直接寫入過程資料

> ***安全**：強制輸出會讓設備不受程式控制，請確認現場安全再操作。

11.6.3　SDO（CoE）：讀寫單一物件

選一個 PDO 項目後按 **SDO（CoE）...**（會帶入其 Index），選動作（讀取或寫入）、Index:Sub（16 進位，例如 `1018:01`）、大小（1 / 2 / 4 / 8 位元組）、值（寫入用：10 進位，或 0x 開頭 16 進位）。失敗時顯示 abort code 與說明。

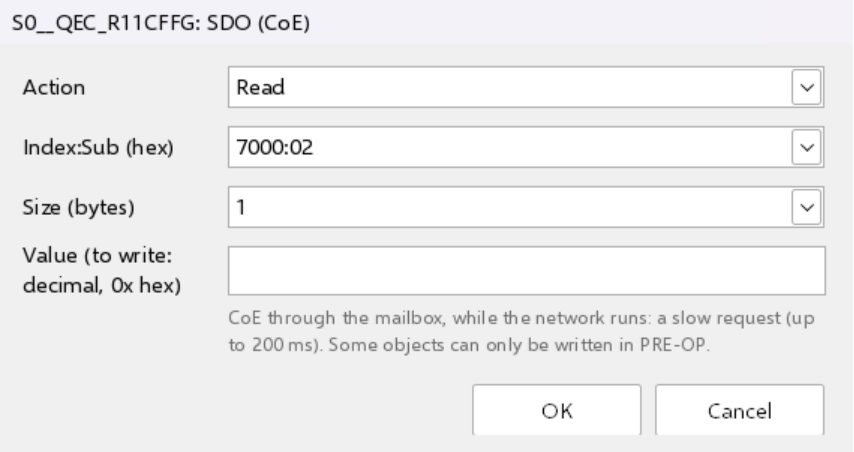


圖 11-55　SDO（CoE）：讀取或寫入單一物件

11.6.4　CoE 物件字典（線上）

類似 TwinCAT 的 CoE - Online，可以瀏覽與修改驅動器的全部參數：

1. 連線後切到 SubDevice 頁的 **CoE 物件字典（線上）** 分頁，按 **讀取物件字典**：以 SDO Info 向從站取得完整清單（名稱、型別、存取權限），存入專案，下次開啟直接顯示。
2. 按 **讀取全部數值**；或勾選 **自動更新** 並選間隔（0.5 / 1 / 2 / 5 秒），畫面上看得到的列會持續更新，數值變化時底色短暫變黃。
3. 上方 **過濾** 以 Index 或名稱篩選，例如 `6098`、`Homing`。
4. 雙擊 RW 項目（或按 **寫入...**）修改參數，寫入後立即讀回。

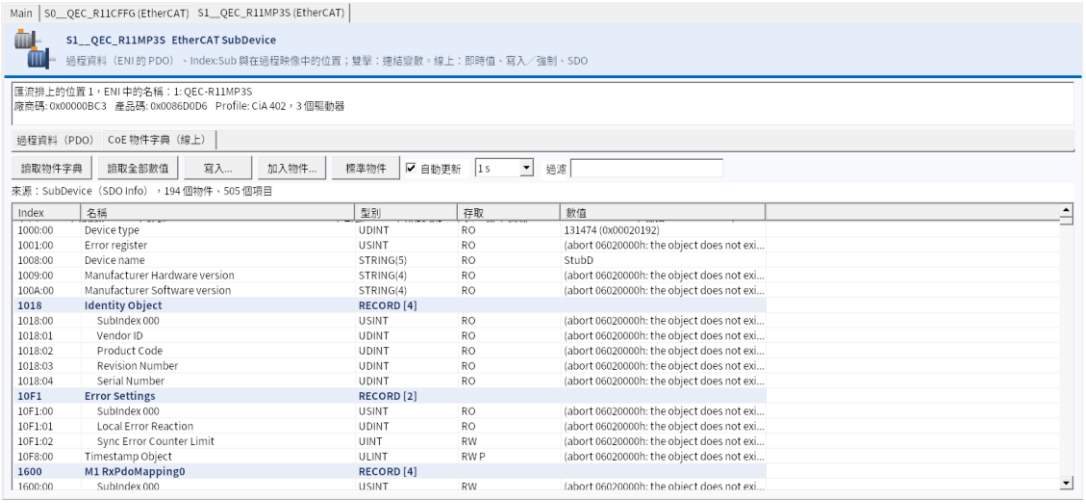


圖 11-56　CoE 物件字典（線上）：QEC-R11MP3S 的物件、型別、存取與數值

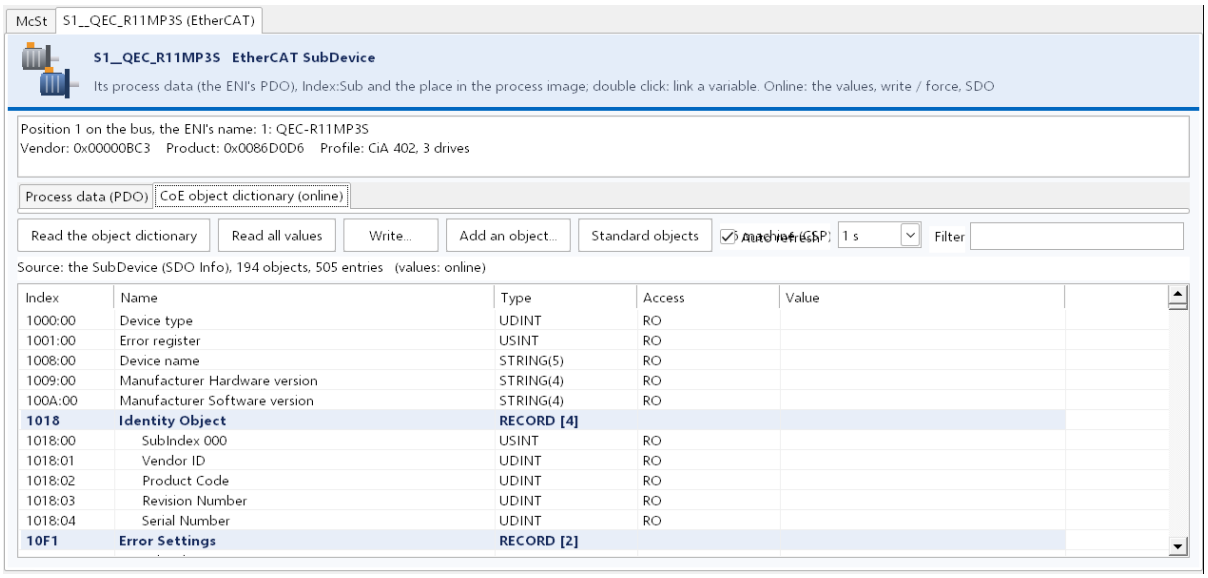


圖 11-57　未連線時：顯示已存入專案的物件清單（數值需連線）

- 存取欄：RO 唯讀、RW 可讀寫；「(PRE-OP)」表示只能在 PRE-OP 狀態寫入；「P」表示可映射到 PDO。
- 從站不支援 SDO Info 時改列**標準物件**（1000h、1018h、CiA 402 常用物件）與 PDO 項目；**加入物件...** 可手動加入其他 Index。
- 讀寫在控制器的掃描之間以非阻塞方式進行，**不會讓 PLC 掃描停住**。
- 常見 abort code：06020000h 物件不存在、06010002h 唯讀、06090030h 數值超出範圍、08000022h 目前狀態不允許（PRE-OP 才能寫）。
- 命令列：`86ldmicro-cli online <專案> <埠> sdo …、… coe <SubDevice>`（15.5 節）。

11.7 疑難排解

EtherCAT 沒有作用時，先開「序列埠監看」（不需上線）看主站狀態，再依下表處理：

狀況	原因與處理
EtherCAT 頁顯示沒有 Configurator 回應	尚未上傳 Configurator，或 PLC 程式已取代它：按 **上傳 Configurator**；USB 時確認工具列選的是控制器的埠；網路時確認電腦連到控制器的 LAN 埠與 IP
`EtherCAT: begin failed`	EtherCAT 線未接到主站的 EtherCAT 埠、從站未送電；.ino 仍呼叫 1.x 的 `EVA.begin()`（11.1 節）
`the ENI … cannot be used`	控制器上沒有專案的 ENI（`C:/eni/…`）或與實際網路不符，主站改用掃描到的網路：重新 Scan、匯出並上傳 ENI、讀回專案
`no slave N`	從站數量或順序與 ENI 不符、線接在從站的 OUT 而非 IN：檢查接線，重新 Scan、匯出 ENI、讀回
輸入永遠是 FALSE	變數沒有對應（建置時「訊息」會警告）；或該 PDO 不在 ENI 的指派中（Process Data 分頁勾選後重新匯出）
EtherCAT 完全沒有作用	移除 1.x 的 `EVA.begin()` 與 `myeva.*`；確認 EtherCAT 裝置的 master 為 `plc`（11.4.3）
CSP 驅動器不動	PDO 需有 6040h / 607Ah / 6041h / 6064h（11.3.6）；在 CoE 物件字典看 Statusword、錯誤碼 603Fh；機台與軸的設定見第 12 章
Configurator 上傳後 PLC 程式不見了	正常現象：Configurator 暫時取代 PLC 程式；設定完成後按 **上傳**（F8）
上傳或線上時 Configurator 斷線	USB 模式下上傳與線上監看優先，Configurator 自動暫停；完成後回到 EtherCAT 頁即恢復

第 12 章　Motion86 運動控制

86LDmicro 以 TwinCAT 式的「機台 + 軸」管理 EtherCAT 伺服驅動器。本章說明 Motion86 機台與軸的建立與設定、不寫程式的試車、以 PLCopen MC 功能塊與 M86 機台功能塊寫運動程式，以及安全機制。開始前請先完成第 11 章，讓 CiA 402 驅動器出現在專案樹「裝置 > EtherCAT」。

12.1　概念

一台 **Motion86 機台**是 Motion86 函式庫的一個軸群組，最多 **9 軸**（X、Y、Z、U、V、W、A、B、C）。每個**軸**是一台 EtherCAT CiA 402 驅動器，以 **CSP（週期同步位置）**模式控制：Motion86 在每個 EtherCAT 週期計算各軸的目標位置，送給驅動器。範例 `tools\example\ex30_dispense` 的機台 **Gantry** 使用 QEC-R11MP3S 的三個軸（AxisX、AxisY、AxisZ）。

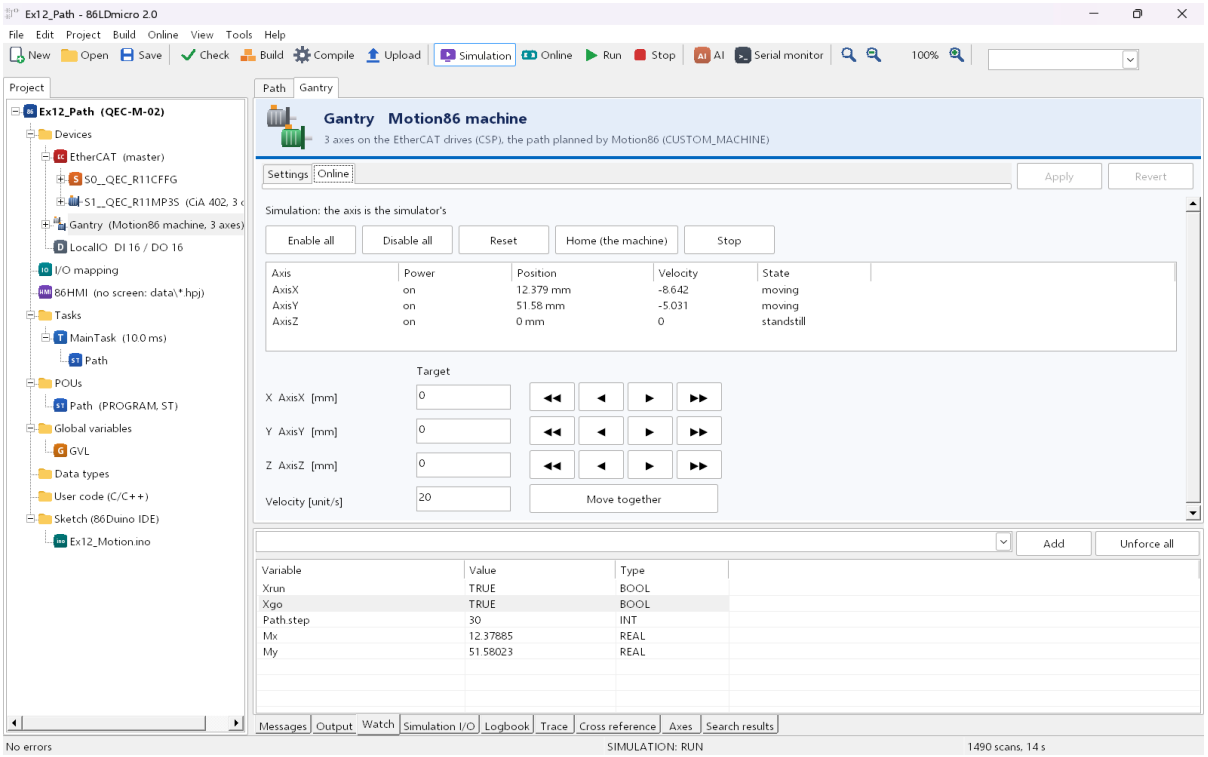


圖 12-1　Gantry 機台的三個軸在模擬中移動：專案樹的機台與軸、機台頁「線上」

名詞	說明
機台	一組一起規劃的軸；程式中以 `Machine := Gantry` 指定
軸	一台驅動器（或虛擬軸）；軸名就是程式中的常數，例如 `Axis := AxisX`
軸字母	機台的第 1～9 軸依序是 X、Y、Z、U、V、W、A、B、C；G-code 與 M86_* 功能塊用這些字母指定軸
使用者單位	程式中的位置用 mm、deg 等單位，速度為 unit/s；由軸的 PPU 換算成驅動器的 counts
PPU	每單位的脈衝數 = 每轉脈衝數 ÷ 每轉移動量；例如 6400 counts/rev、螺桿導程 10 mm → 640 counts/mm
插補	整台機台一條路徑（直線、圓弧、G-code），各軸同時到達
獨立	每軸各自規劃，可同時各跑各的移動
CSP	Cyclic Synchronous Position：主站每個週期送目標位置（607Ah），驅動器回報實際位置（6064h）

12.2 建立機台

三種方式都會開啟「新增 Motion86 機台」視窗：

- 專案樹「裝置」按右鍵 >「新增裝置」> `motion86`。
- 在專案樹的 CiA 402 驅動器（馬達圖示）上按右鍵 >「作為 Motion86 機台的軸」。
- 在驅動器的 SubDevice 頁按 **作為 Motion86 機台的軸（CSP）...**（11.5 節）。

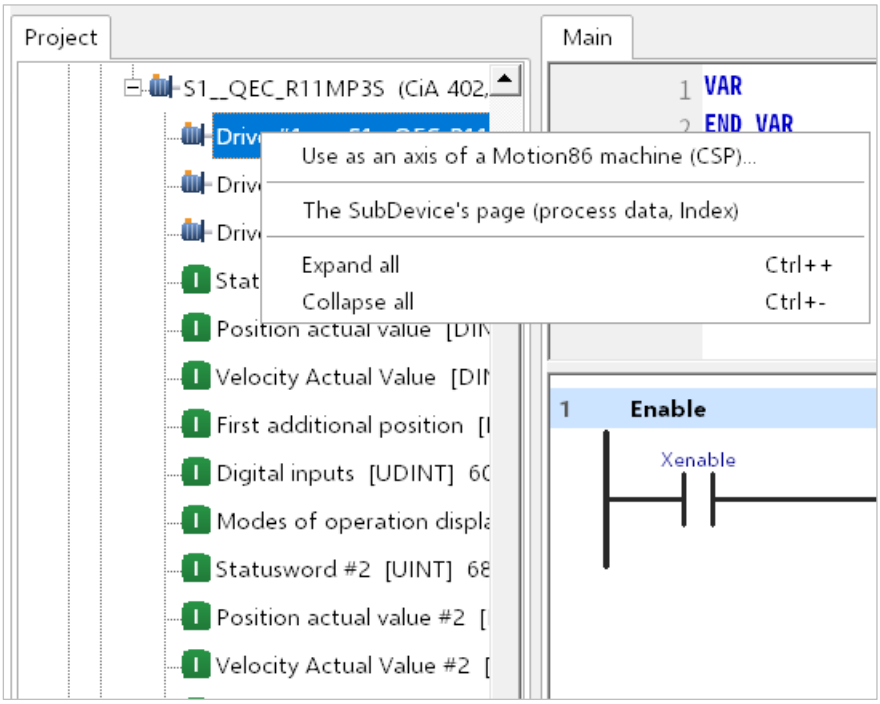


圖 12-2　在驅動器上按右鍵：作為 Motion86 機台的軸

1. 輸入機台名稱（例如 `Gantry`；英文字母、數字、_，不可與其他裝置或軸同名）。
2. 選軸數（1～9）。可用的 CiA 402 驅動器依序成為 X、Y、Z、U、V、W、A、B、C 軸；沒有可用的驅動器時先建立為**虛擬軸**，之後在軸頁的「驅動器」再選。
3. 按 **確定**。專案樹出現機台與各軸（馬達圖示），軸後面顯示軸字母與對應的驅動器，例如 `AxisX (X) -> S1__QEC_R11MP3S #1`。

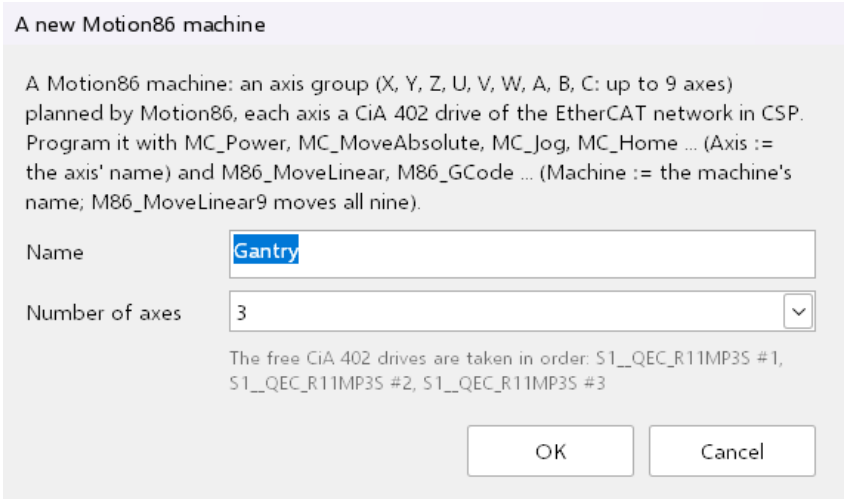


圖 12-3　新增 Motion86 機台：名稱與軸數（1～9）

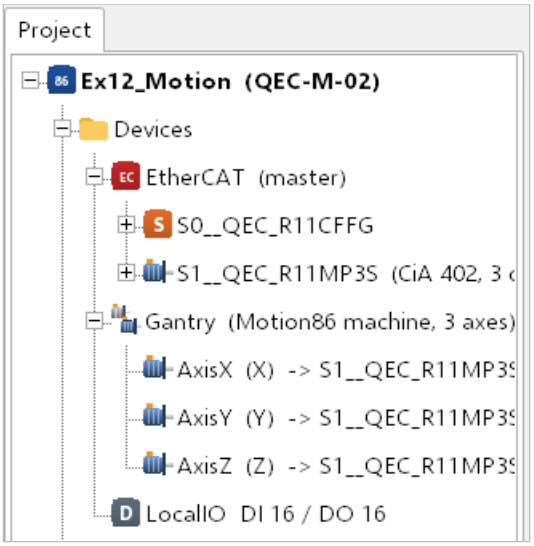


圖 12-4　專案樹：機台 Gantry 與三個軸，軸後面顯示對應的驅動器

*提示：軸名與機台名就是程式中的常數（DINT），例如 `Axis := AxisX`、`Machine := Gantry`。改名後請一併修改程式，按 **檢查**（Shift+F7）找出遺漏。

12.3 機台設定

雙擊專案樹的機台，開啟機台頁的「**設定**」分頁。修改後按 **套用**（存入專案；建置並上傳後才會套用到控制器），**還原** 放棄修改。

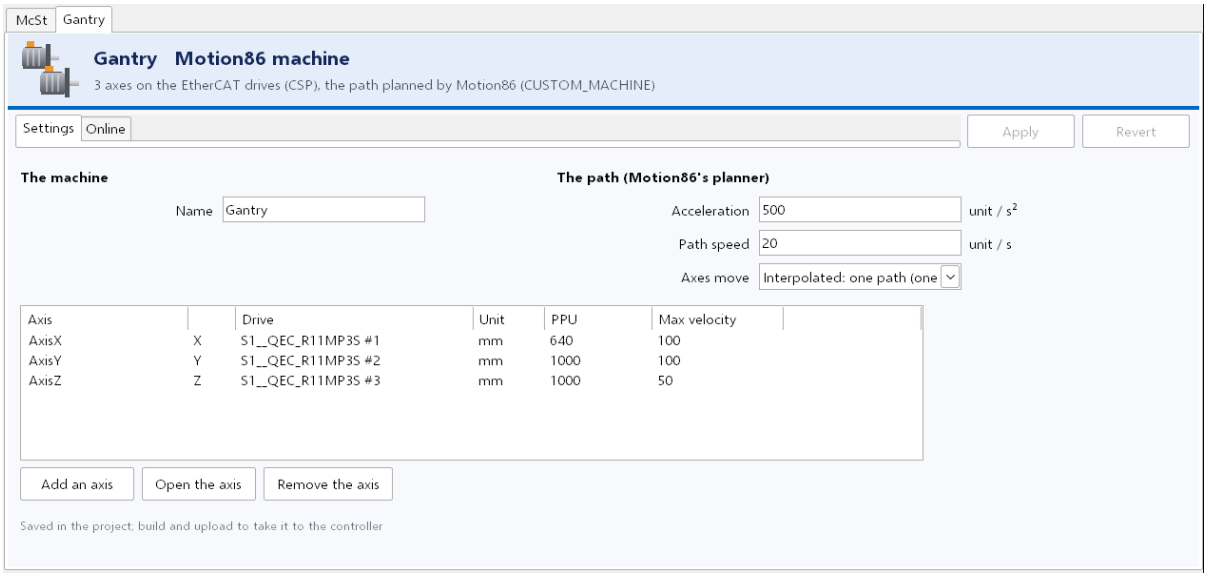


圖 12-5　機台頁「設定」：名稱、路徑加速度、路徑速度、軸運動方式與軸清單

欄位	說明
名稱	機台名稱，程式中 `Machine := Gantry`
加速度（unit/s²）	Motion86 規劃器的路徑加減速度；MC_* 的 Acceleration / Deceleration 由此決定
路徑速度（unit/s）	M86_MoveLinear 的 Feedrate 為 0 時使用
軸運動方式	**插補**：單一路徑（一台 Motion86 機台），同一機台的移動依序排隊；**獨立**：各軸各自運動（每軸一台 Motion86 機台），可同時移動
軸清單	軸、字母、驅動器、單位、PPU、最大速度；按 **新增軸**、**開啟軸**、**移除軸**（驅動器仍保留在 EtherCAT 網路中）

*__注意__：EtherCAT 驅動器上的 Motion86 機台最多 **3 台**；獨立模式每軸占一台，因此整個專案最多 3 個獨立軸（或 1 台插補機台 + 2 個獨立軸）。超過 3 軸的機台請用**插補**模式（一台機台就能帶 9 軸）。超過時「套用」與建置會提示錯誤。獨立模式下 M86_MoveArcXY / M86_GCode 不執行。

12.3.1　多軸機台（最多 9 軸）

一台機台可以有 1～9 個軸，依序對應 X、Y、Z、U、V、W、A、B、C。專案樹與軸清單都會顯示每個軸的字母。

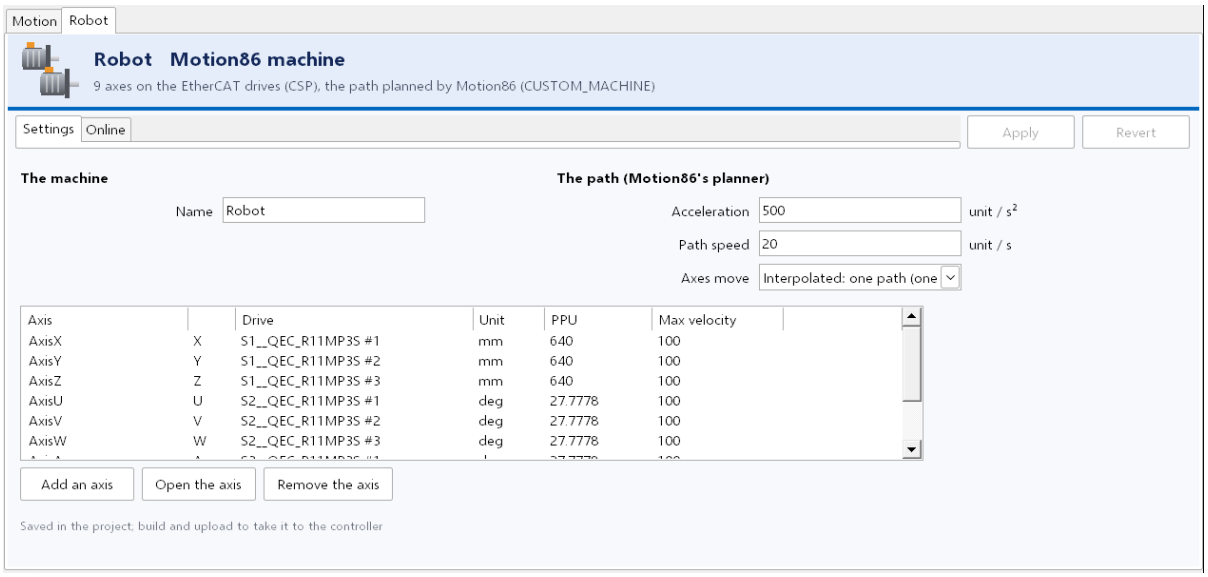


圖 12-6　9 軸機台 Robot 的「設定」：軸清單的第二欄是軸字母（X～C），清單可上下捲動

1. 建立機台時在「軸數」選 9；或在機台頁按 **新增軸**，一次加一軸（到 9 軸為止）。
2. 雙擊每一個軸，在軸頁「驅動器」選它的 CiA 402 驅動器，設定單位、每轉脈衝數與每轉移動量（12.4 節）。直線軸用 mm、旋轉軸用 deg。
3. 「軸運動方式」保持 **插補**，按 **套用**，再建置、上傳。

12.4 軸設定

雙擊專案樹的軸，開啟軸頁的「**設定**」分頁；修改後按 **套用**。

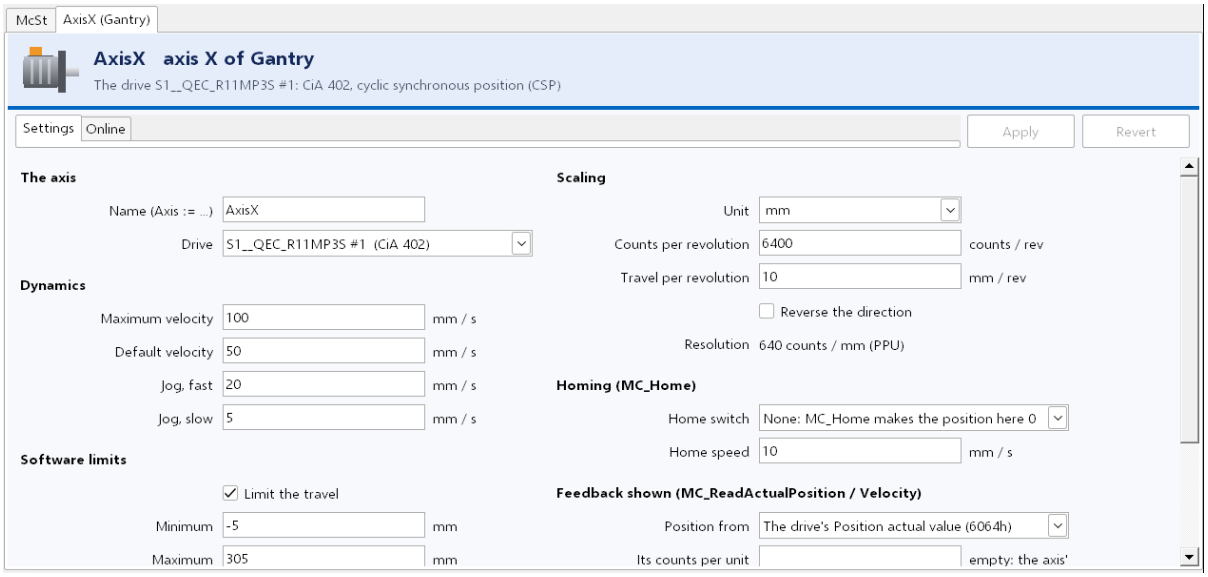


圖 12-7　軸頁「設定」：驅動器、動態參數、軟體極限、比例換算與原點復歸

區塊	欄位	說明
軸	名稱（Axis := ...）	程式中的軸名
軸	驅動器	例如 `S1__QEC_R11MP3S #1`；多軸驅動器選 #1 / #2 / #3；也可選虛擬軸（沒有驅動器，位置即命令）
動態參數	最大速度、預設速度	unit/s；MC_* 的 Velocity 為 0 時用預設速度
動態參數	寸動（快）、寸動（慢）	軸頁寸動按鈕與 MC_Jog（Velocity 為 0）的速度
軟體極限	限制行程、最小值、最大值	勾選後，超出範圍的移動會被拒絕
比例換算	單位、每轉脈衝數（counts）、每轉移動量	解析度（PPU）自動計算顯示
比例換算	反轉方向	馬達方向與座標相反時勾選
原點復歸（MC_Home）	原點開關	**無**：MC_Home 將目前位置設為 0；**驅動器的原點開關（60FDh bit 2）**；或 **GVL 中的 BOOL**
原點復歸（MC_Home）	原點復歸速度	找原點開關的速度（離開開關時為其 1/10）

***注意：模擬器不套用軟體極限：模擬中 MC_MoveAbsolute 等命令超出極限仍會照常移動。建置到控制器時，極限由 Motion86 的位置極限處理（依「限制行程」的最小、最大值）。**

超出極限時控制器的實際反應，請在實機以低速、短距離確認後，再依此設計錯誤處理。

12.4.1 顯示的回授

「**顯示的回授（MC_ReadActualPosition / Velocity）**」區塊決定軸頁顯示的位置、速度，以及 MC_ReadActualPosition / MC_ReadActualVelocity 讀到的值從哪裡來：

欄位	說明
位置來源	預設：**驅動器的實際位置（6064h）**。可改選 EtherCAT 網路中任一 TxPDO 項目（例如 QEC-R11MP3S 的編碼器位置），顯示值 = 項目值 ÷ 其每單位計數 + 其偏移量
其每單位計數	所選項目每個單位的計數；**空白 = 使用軸設定**（本軸的 PPU，含反轉方向）；填負數可反向
其偏移量	加到顯示值的偏移，單位同軸的單位
速度來源	預設：**軸本身的（Motion86，依運動計算）**。可改選 TxPDO 項目（例如 606Ch 實際速度），MC_ReadActualVelocity = 項目值 ÷ 其每 單位/s 的 計數/s
其每 單位/s 的 計數/s	速度項目的換算；空白 = PPU

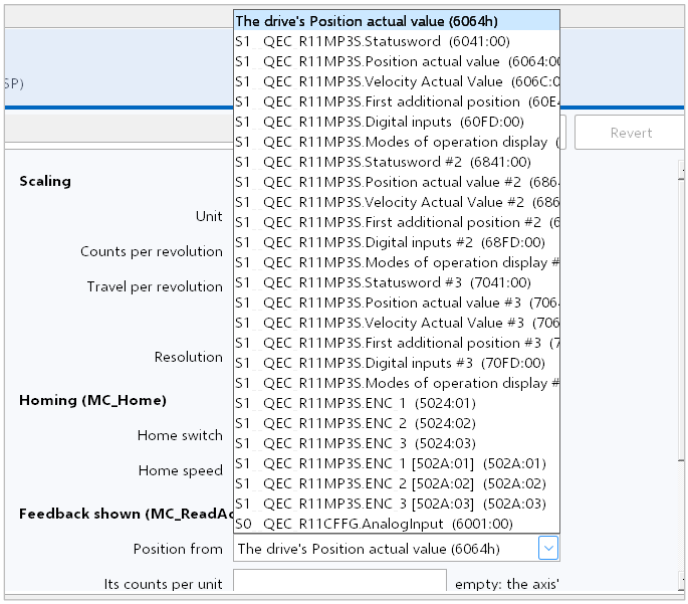


圖 12-8　軸頁「設定」的「顯示的回授」：位置來源選擇 TxPDO 項目

*__重要__：位置 / 速度來源**只影響顯示**與 **MC_ReadActualPosition / MC_ReadActualVelocity**。CSP 控制、到位判斷、原點復歸、軟體極限、M86_ReadPosition 與機台頁的座標仍使用驅動器的 6064h。

MC_Home、MC_SetPosition 不會改變外部項目的讀值：原點復歸後，顯示的回授值不會歸零（86LDmicro 2.0.0 的限制）。需要時以「其偏移量」對齊。

外部編碼器（例如 60E4h）的解析度通常與馬達不同，「其每單位計數」請填該編碼器每單位的計數；空白會用馬達的 PPU，換算錯誤時數值看起來會放大、跳動。

模擬時沒有 EtherCAT 的值，顯示驅動器（模擬器）的位置。所選項目若已不在 ENI 中，建置時會警告並改顯示驅動器的值。

12.5 機台試車

模擬（F5）或連線（Ctrl+F5）後，切到機台頁的「**線上**」分頁，不寫程式就能操作整台機台。未模擬、未連線時，頁面提示「離線：請先開始模擬（F5）或連線（Ctrl+F5）再試車」。

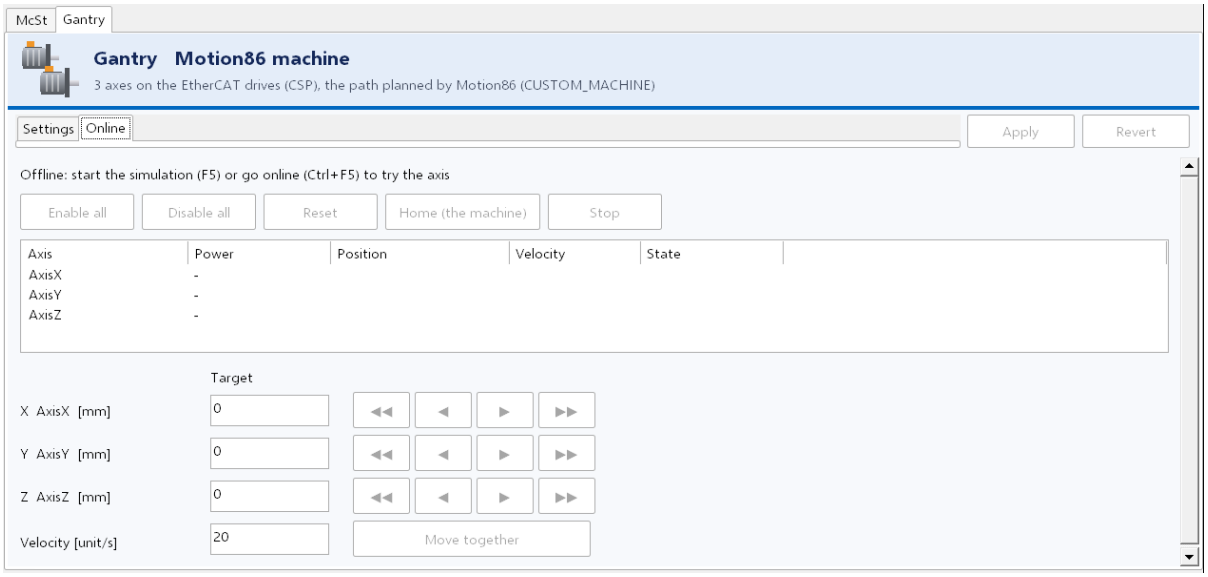


圖 12-9　機台頁的「線上」分頁：上方為各軸狀態，下方為目標、速度與寸動

按鈕 / 欄位	說明
全部激磁	所有軸的驅動器進入 Operation Enabled，「激磁」欄顯示 ON
全部解除激磁	先停止再解除所有驅動器
寸動 ◀◀ ◀ ▶ ▶▶	每軸一列：快退、慢退、慢進、快進；**按住才動、放開即停**
目標、速度、一起移動	在每軸的「目標」欄輸入位置、「速度」欄輸入 unit/s，按 **一起移動**：插補模式下各軸做一次直線移動、同時到達；獨立模式下各軸同時出發、各自到位
原點復歸（整台機台）	有設定原點開關的軸逐一找原點；沒有開關的軸位置不變
STOP	立即停止所有軸
重置	清除錯誤

1. 按 **全部激磁**，確認每軸的「激磁」為 ON。
2. 以**慢速寸動**（◀ ▶）確認每軸的方向。
3. 在「目標」輸入 X 80、Y 40、Z 20，「速度」輸入 20，按 **一起移動**。
4. 結束時按 **全部解除激磁**。

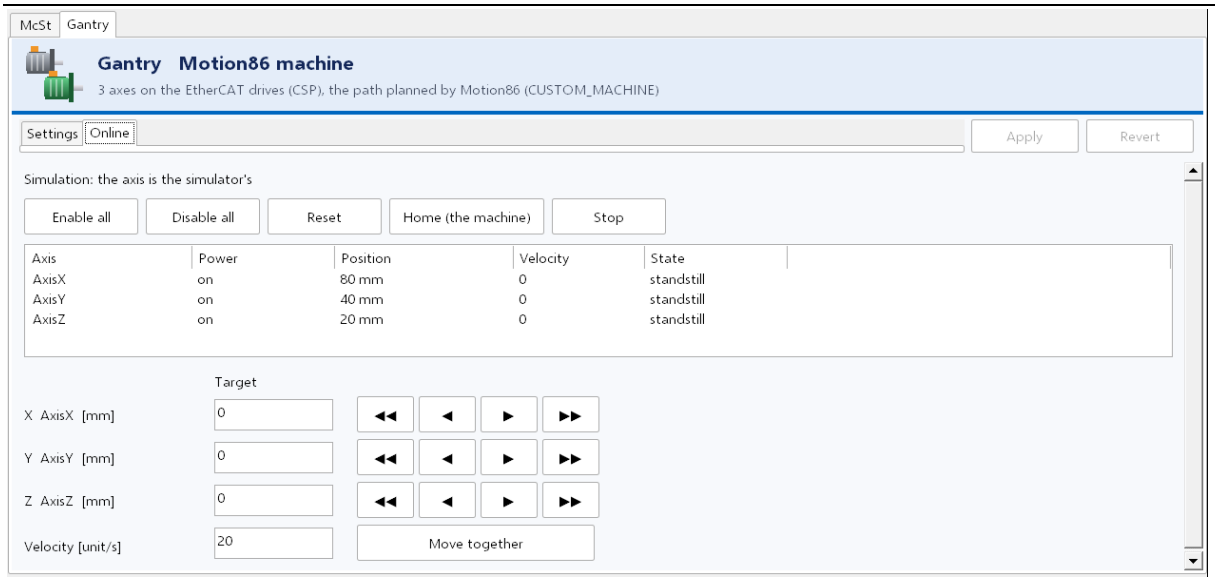


圖 12-10　Gantry 線上：三軸已激磁並移動到 X 80、Y 40、Z 20 mm

超過 3 軸時，每一軸有一列目標與寸動按鈕，列數依軸數增加（排列依視窗大小而定）；畫面放不下時用滑鼠滾輪或右側捲軸往下捲，找到「速度」與 一起移動：

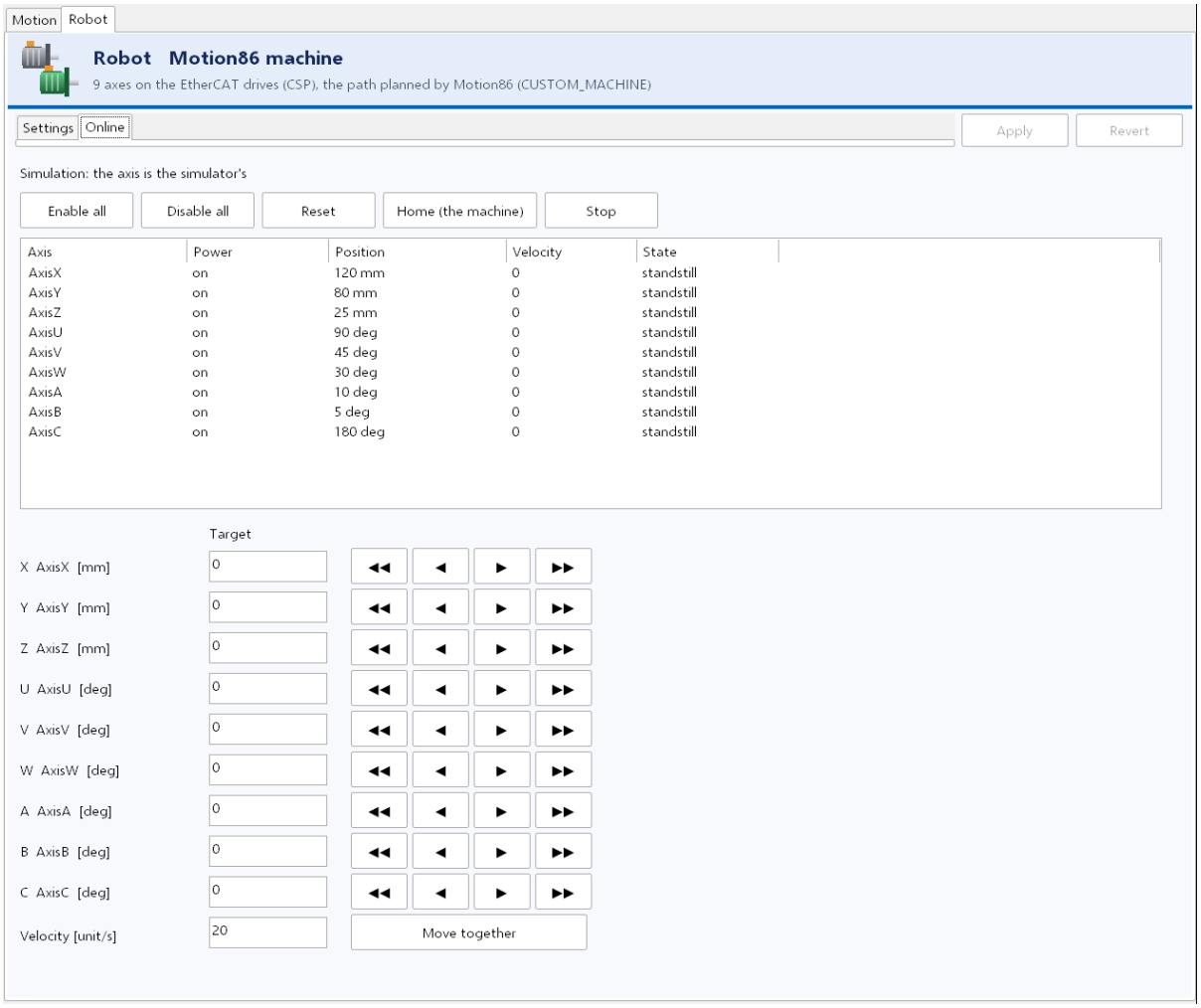


圖 12-11　9 軸機台的「線上」：上方為各軸狀態，下方為各軸的目標、寸動與「一起移動」

12.6 單軸試車

軸頁的「**線上**」分頁操作單一軸，並顯示位置、速度、追隨誤差、命令與最近 30 秒的趨勢：

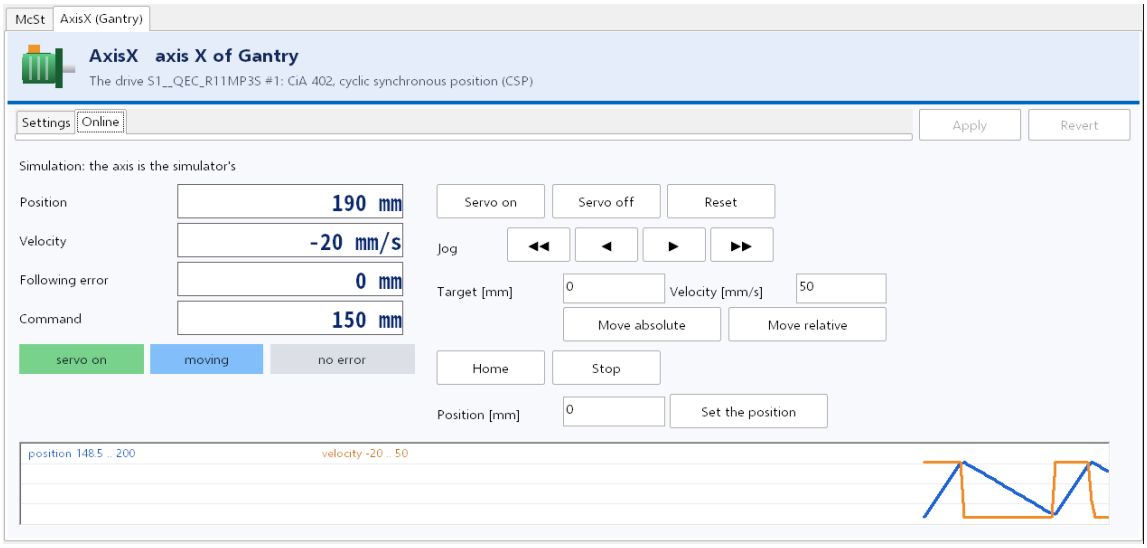


圖 12-12　軸頁「線上」：位置、速度、追隨誤差、命令、狀態燈、激磁、寸動、移動、原點、設定位置與趨勢圖

按鈕 / 欄位	說明
激磁 / 解除激磁 / 重置	驅動器激磁、解除；重置清除錯誤
狀態燈	已激磁、靜止 / 運動中、錯誤
寸動 ◀◀ ◀ ▶ ▶▶	按住才動（每 200 ms 更新命令，控制器 600 ms 沒收到就自動停止）
絕對移動 / 相對移動	輸入目標（或距離）與速度
原點復歸	依軸的原點開關設定（12.4 節）
STOP	立即停止
設定目前位置	把目前位置設為輸入的值，驅動器不動（同 MC_SetPosition）
趨勢	位置（藍）與速度（橘），最近 30 秒，模擬或線上時顯示

12.6.1 軸（試車）視窗

「檢視 > 軸（試車）」列出專案所有軸的激磁、位置與命令狀態，可直接激磁、寸動、移動、原點復歸、STOP；雙擊 Motion86 軸開啟它的軸頁。在專案樹的機台上按右鍵 >「軸面板（試車）」也可開啟。

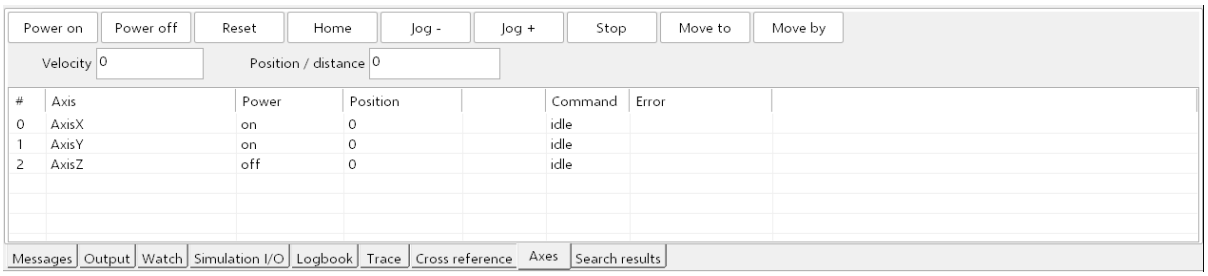


圖 12-13　軸（試車）視窗：所有軸的狀態與操作

12.7 PLCopen MC 功能塊

程式中以 PLCopen 的 MC_* 功能塊控制**單一軸**。在 ST、SFC 中直接呼叫；在階梯圖中以 **CALL** 插入（7.8、7.11 節）。

12.7.1 基本規則

- `Axis :=` 填軸名（例如 `AxisX`）；軸名就是 DINT 常數。
- 位置與距離為軸的單位（mm…），速度為 unit/s；參數型別都是 **REAL**。Velocity 為 0 時用軸的預設速度。
- Acceleration / Deceleration 由機台的加速度決定（傳入的值保留給相容）。
- 功能塊實例要在 VAR 宣告，且**每個掃描都要呼叫**（或至少在 Execute 改變與等待 Done 期間呼叫）。
- 插補模式下，同一台機台的 MC_* 移動依序排隊；要多軸同時直線移動，請用 M86_MoveLinear（12.8 節）。

12.7.2 功能塊的共通行為

類型	觸發	輸出
Execute 類（MoveAbsolute、MoveRelative、MoveVelocity、Home、Stop、Halt、Reset、SetPosition）	`Execute` 的**上升緣**送出命令；要再送一次，先把 Execute 設回 FALSE	`Busy`（執行中）→ `Done`（完成）或 `Error`（`ErrorID`）；被新命令取代時 `CommandAborted`。Done / Error 在 Execute 為 TRUE 期間保持
Enable 類（Power、ReadActualPosition、ReadActualVelocity、ReadStatus、ReadAxisError）	`Enable` 為 TRUE 期間持續有效	`Valid`（或 MC_Power 的 `Status`）與讀到的數值
MC_Jog	`JogForward` / `JogBackward` 為 TRUE 期間移動（兩者同時或都 FALSE 則停止）	`Busy`、`Done`、`Error`

12.7.3 功能塊一覽

功能塊	輸入	輸出	說明
MC_Power	Axis, Enable	Status, Valid, Error, ErrorID	激磁；Status 為 TRUE 表示已激磁
MC_MoveAbsolute	Axis, Execute, Position, Velocity, Acceleration, Deceleration	Done, Busy, Active, CommandAborted, Error, ErrorID	移動到絕對位置
MC_MoveRelative	Axis, Execute, Distance, Velocity, Acceleration, Deceleration	同上	移動一段距離
MC_MoveVelocity	Axis, Execute, Velocity, Acceleration, Deceleration	InVelocity, Busy, Active, CommandAborted, Error, ErrorID	以速度持續移動，直到 MC_Stop
MC_Jog	Axis, JogForward, JogBackward, Velocity, Acceleration, Deceleration	Done, Busy, CommandAborted, Error, ErrorID	寸動（需持續呼叫）
MC_Home	Axis, Execute, Method, Velocity, Acceleration	Done, Busy, Active, CommandAborted, Error, ErrorID	原點復歸（依軸的原點開關設定）
MC_Stop	Axis, Execute, Deceleration	Done, Busy, CommandAborted, Error, ErrorID	停止（立即停止）
MC_Halt	Axis, Execute, Deceleration	Done, Busy, Active, CommandAborted, Error, ErrorID	暫停（立即停止）
MC_Reset	Axis, Execute	Done, Busy, Error, ErrorID	清除錯誤
MC_SetPosition	Axis, Execute, Position	Done, Busy, Error, ErrorID	把目前位置設為 Position（驅動器不動）
MC_ReadActualPosition	Axis, Enable	Valid, Busy, Error, ErrorID, Position	目前位置（REAL）
MC_ReadActualVelocity	Axis, Enable	Valid, Busy, Error, ErrorID, Velocity	目前速度
MC_ReadStatus	Axis, Enable	Valid, Busy, Error, ErrorID, ErrorStop, Disabled, Stopping, Homing, StandStill, DiscreteMotion, ContinuousMotion, SynchronizedMotion	軸的狀態
MC_ReadAxisError	Axis, Enable	Valid, Busy, Error, ErrorID, AxisErrorID	軸的錯誤碼

12.7.4 範例：單軸激磁、原點、往返三次

情境：Gantry 的 X 軸：激磁開關 Xenable 打開後，按 Xstart：X 軸原點復歸，移動到 200 mm，再後退 50 mm，往返 3 次後停在 150 mm；Xstop 停止軸並回到待機（再按 Xstart 從原點復歸重新開始）；發生錯誤時停在錯誤步驟、Yerror 亮，按 Xreset 重置後回到待機。按住 Xstart 不放也只執行一次。

變數	型別	I/O 對應	用途
Xenable	BOOL	數位輸入	激磁開關
Xstart	BOOL	數位輸入	啟動按鈕
Xstop	BOOL	數位輸入	停止按鈕
Xreset	BOOL	數位輸入	錯誤重置按鈕
Ybusy	BOOL	數位輸出	動作中指示燈
Yerror	BOOL	數位輸出	錯誤指示燈
Mpos	REAL	—	X 軸目前位置（mm）
step、count	INT	—（McSt 的區域變數）	步驟、已往返次數

```
PROGRAM McSt
VAR
    pwr  : MC_Power;
    home : MC_Home;
    mv   : MC_MoveAbsolute;
    rel  : MC_MoveRelative;
    stp  : MC_Stop;
    rst  : MC_Reset;
    rd   : MC_ReadActualPosition;
    st   : MC_ReadStatus;
    ae   : MC_ReadAxisError;
    startEdge : R_TRIG;      // Xstart 的新按壓
    resetEdge : R_TRIG;      // Xreset 的新按壓
    resetReq  : BOOL;        // 已要求 MC_Reset
    step  : INT;             // 0 待機、10 原點、20 去程、30 回程、90 停止中、99 錯誤
    count : INT;             // 已往返次數
    lastError : DINT;        // 最後一次錯誤的 ErrorID
END_VAR
pwr(Axis := AxisX, Enable := Xenable);
rd(Axis := AxisX, Enable := TRUE);
Mpos := rd.Position;
st(Axis := AxisX, Enable := TRUE);
ae(Axis := AxisX, Enable := TRUE);
startEdge(CLK := Xstart);
resetEdge(CLK := Xreset);

(* 1. 移動中，錯誤與停止優先：結束目前的命令 *)
IF step >= 10 AND step <= 30 THEN
    IF home.Error OR mv.Error OR rel.Error OR st.ErrorStop THEN
        IF home.Error THEN lastError := home.ErrorID;
        ELSIF mv.Error THEN lastError := mv.ErrorID;
        ELSIF rel.Error THEN lastError := rel.ErrorID;
        ELSE lastError := ae.AxisErrorID;     (* 只有軸本身進入 ErrorStop *)
        END_IF;
```

```
        step := 99;
    ELSIF Xstop THEN
        step := 90;
    END_IF;
END_IF;

(* 2. 步驟 *)
CASE step OF
0:  IF pwr.Status AND startEdge.Q AND NOT Xstop THEN   (* 新的按壓，且沒有按住停止 *)
        count := 0;
        step := 10;
    END_IF;
10: IF home.Done THEN step := 20; END_IF;
20: IF mv.Done THEN step := 30; END_IF;
30: IF rel.Done THEN
        count := count + 1;
        IF count < 3 THEN step := 20; ELSE step := 0; END_IF;
    END_IF;
90: IF stp.Error THEN
        lastError := stp.ErrorID;
        step := 99;
    ELSIF stp.Done AND NOT Xstop THEN    (* 已停止：回待機，再按 Xstart 從原點復歸開始 *)
        step := 0;
    END_IF;
99: IF resetEdge.Q THEN resetReq := TRUE; END_IF;   (* 由操作員重置 *)
    IF rst.Error THEN                    (* 重置失敗：記下錯誤，可再按 Xreset *)
        lastError := rst.ErrorID;
        resetReq := FALSE;
    ELSIF rst.Done THEN
        resetReq := FALSE;
        IF NOT st.ErrorStop THEN step := 0; END_IF;
    END_IF;
END_CASE;

(* 3. 命令：Execute 只在自己的步驟為 TRUE，離開步驟就結束命令，
      回到該步驟時有新的上升緣 *)
home(Axis := AxisX, Execute := step = 10);
mv(Axis := AxisX, Execute := step = 20, Position := 200.0, Velocity := 50.0);
rel(Axis := AxisX, Execute := step = 30, Distance := -50.0, Velocity := 20.0);
stp(Axis := AxisX, Execute := step = 90);
rst(Axis := AxisX, Execute := resetReq AND step = 99);
Ybusy := step >= 10 AND step <= 30;
Yerror := step = 99;
END_PROGRAM
```

#	操作
1	依第 11 章讀入 ENI（例如匯入 ex30_dispense 的 `data\86ld\ethercat\QEC-ENI-1.xml`），依 12.2 節建立 3 軸機台 Gantry（AxisX、AxisY、AxisZ）。
2	在 GVL 宣告 Xenable、Xstart、Xstop、Xreset、Ybusy、Yerror（BOOL）與 Mpos（REAL）。
3	「專案 > 新增 PROGRAM（ST）...」，名稱 `McSt`，輸入上面的程式。
4	按 **檢查**（Shift+F7）確認沒有錯誤。
5	按 **模擬**（F5）。監看視窗加入 Xenable、Xstart、Mpos、Ybusy、McSt.step、McSt.count。
6	把 Xenable 寫成 TRUE，再把 Xstart 寫成 TRUE（約 1 秒後寫回 FALSE）。開啟 AxisX 的軸頁「線上」觀察。

預期結果：X 軸原點復歸（沒有原點開關時，目前位置設為 0），以 50 mm/s 移動到 200 mm（約 4 秒），再以 20 mm/s 後退到 150 mm；重複 3 次後停在 150 mm，McSt.step 回到 0、Ybusy 熄滅。按住 Xstart 不放，完成後也不會自動再執行；按住 Xstop 時按 Xstart 不會啟動。移動中把 Xstop 寫成 TRUE：軸停止，step 經 90 回到 0；放開 Xstop 後再按 Xstart，從原點復歸重新開始。

***注意：錯誤時程式停在步驟 99（Yerror 亮、lastError 記下 ErrorID），要按 Xreset 才重置，重置成功後回到步驟 0；錯誤只在移動步驟中判斷，所以不會一直回到 99；重置失敗（rst.Error）時記下 ErrorID，可再按 Xreset 重試。**

模擬器不會產生軸的錯誤，也不套用軟體極限（12.4 節）；錯誤與重置的流程請在實機上以可控制的方式確認（例如低速、短距離），並先確認停止與緊急停止有效。

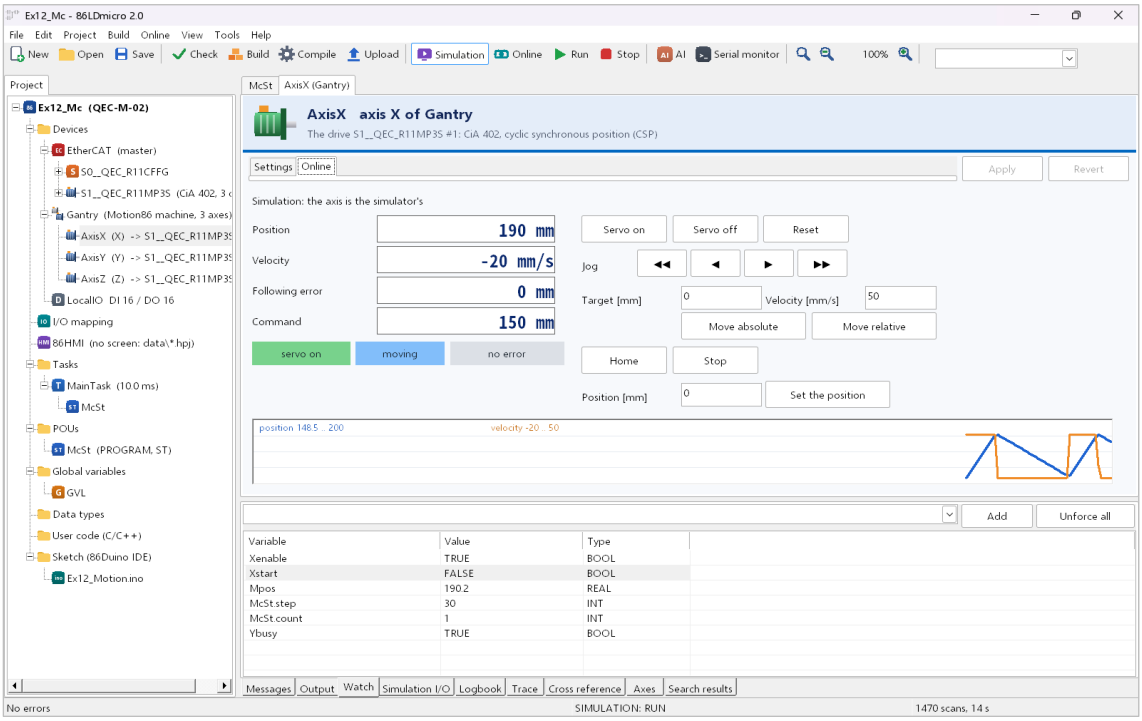


圖 12-14　模擬中的 McSt：監看視窗的 step、count、Mpos 與 AxisX 軸頁的趨勢

***提示：**SFC 中使用 MC 時，步驟離開後其動作就不再執行；請在下一個步驟把前一個移動的 Execute 設回 FALSE，下次回到該步驟時才有新的上升緣。MC_Power 建議放在另一個 ST 或 LD 程式中常駐呼叫（8.6 節）。

12.8 M86 機台功能塊

M86_* 功能塊控制**整台機台**：多軸同時的直線、圓弧插補與 G-code。`Machine :=` 填機台名稱。

功能塊	輸入	輸出	說明
M86_Power	Machine, Enable	Status	整台機台全部軸激磁
M86_MoveLinear	Machine, Execute, X, Y, Z, Feedrate	Done, Busy, Error	直線插補到 X、Y、Z（Feedrate 為 unit/min，0 用機台的路徑速度）；U ~ C 軸不動
M86_MoveLinear9	Machine, Execute, X, Y, Z, U, V, W, A, B, C, Feedrate	Done, Busy, Error	機台全部軸（最多 9 軸）一條直線路徑、同時到達；軸數較少時多出的輸入不使用
M86_MoveArcXY	Machine, Execute, CX, CY, X, Y, Clockwise, Feedrate	Done, Busy, Error	XY 平面圓弧（圓心 CX、CY，終點 X、Y）
M86_Home	Machine, Execute	Done, Busy, Error	整台機台原點復歸
M86_GCode	Machine, Execute, Line	Done, Busy, Error	執行一行 G-code，例如 `'G1 X50 Y0 F3000'`（沒寫到的軸不動）
M86_Stop	Machine, Execute, Emergency	Done	停止機台
M86_ReadPosition	Machine, Enable	Valid, X ~ C, Moving	各軸目前位置與是否移動中
M86_Override	Machine, Ratio	—	速度倍率（0 ~ 2）

- `Feedrate` 為 **unit/min**（例如 1200 = 20 mm/s）；0 表示使用機台的路徑速度。
- `Execute` 由 FALSE 變 TRUE 時才送出一次移動；要再移動，先把 Execute 設回 FALSE。
- M86_* 的座標與 MC_* 一致（含 MC_SetPosition 的偏移）；G-code 字串為機台座標。

  M86_MoveLinear、M86_MoveLinear9、M86_MoveArcXY 執行前會切回絕對座標（G90）；M86_GCode 不改變模式：曾送出 G91 之後，G-code 會以相對座標執行，請先送 G90。
- 獨立模式的機台不執行 M86_MoveArcXY 與 M86_GCode（12.3 節）。

12.8.1 範例：直線、圓弧與 G-code

情境：Gantry 激磁（`Xrun`）後按 `Xgo`：XY 直線移動到 (50, 30)，以圓心 (25, 30) 逆時針畫半圓到 (0, 30)，再以 G-code 移動到 (10, 10)；`Mx`、`My` 顯示目前位置。

變數	型別	I/O 對應	用途
Xrun	BOOL	數位輸入	機台激磁
Xgo	BOOL	數位輸入	執行路徑
Mx、My	REAL	—	目前 X、Y（mm）

```
PROGRAM Path
VAR
    mp   : M86_Power;
    lin : M86_MoveLinear;
    arc : M86_MoveArcXY;
    gc   : M86_GCode;
    rp   : M86_ReadPosition;
    step : INT;
END_VAR
mp(Machine := Gantry, Enable := Xrun);
rp(Machine := Gantry, Enable := TRUE);
CASE step OF
0:  IF mp.Status AND Xgo THEN step := 20; END_IF;
20: lin(Machine := Gantry, Execute := TRUE, X := 50.0, Y := 30.0, Z := 0.0, Feedrate := 1200.0);
    IF lin.Done THEN lin(Machine := Gantry, Execute := FALSE); step := 30; END_IF;
30: arc(Machine := Gantry, Execute := TRUE, CX := 25.0, CY := 30.0, X := 0.0, Y := 30.0,
        Clockwise := FALSE, Feedrate := 600.0);                    (* 半圓：(50,30) → (0,30) *)
    IF arc.Done THEN arc(Machine := Gantry, Execute := FALSE); step := 40; END_IF;
40: gc(Machine := Gantry, Execute := TRUE, Line := 'G1 X10 Y10 F3000');   (* 前面的 lin、arc 已切回
G90：絕對座標 *)
    IF gc.Done THEN gc(Machine := Gantry, Execute := FALSE); step := 0; END_IF;
END_CASE;
Mx := rp.X;
My := rp.Y;
END_PROGRAM
```

#	操作
1	使用 12.7.4 的專案（機台 Gantry 為插補模式）；在 GVL 宣告 Xrun、Xgo（BOOL）與 Mx、My（REAL）。
2	「專案 > 新增 PROGRAM（ST）...」，名稱 `Path`，輸入上面的程式；在工作設定中移除 McSt（兩個程式不要同時控制 Gantry）。
3	按 **檢查**，再按 **模擬**（F5）；監看視窗加入 Xrun、Xgo、Mx、My、Path.step。
4	把 Xrun 寫成 TRUE，再把 Xgo 寫成 TRUE；開啟 Gantry 機台頁「線上」觀察。

預期結果：直線移動到 (50, 30) 約 3 秒（20 mm/s），半圓到 (0, 30) 約 8 秒（10 mm/s），G-code 移動到 (10, 10) 後 Path.step 回到 0。Xgo 保持 TRUE 時會再次執行同一路徑。

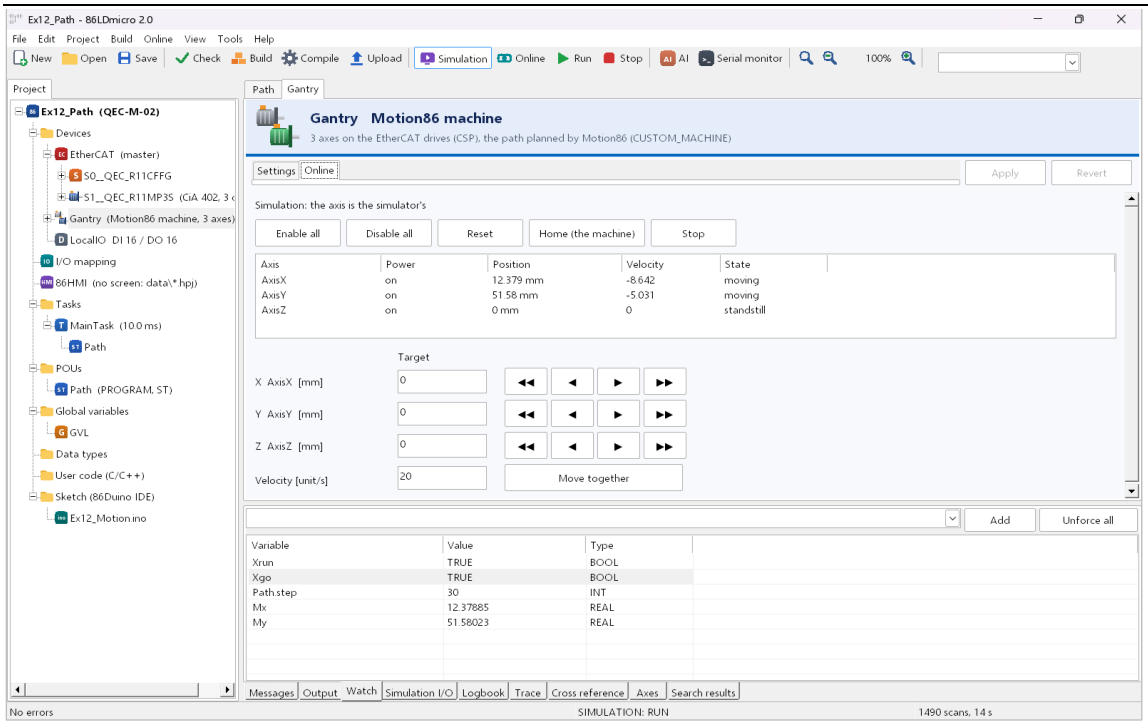


圖 12-15　模擬中的 Path：畫圓弧時機台頁顯示的 X、Y 位置

12.8.2　範例：9 軸一起移動

9 軸機台 Robot（AxisX ~ AxisC）：Xpower 為 TRUE 時激磁，按 Xgo9 用 M86_MoveLinear9 讓 9 軸同時到達目標，並用 M86_ReadPosition 讀回各軸位置。Xpower、Xgo9 為 GVL 的 BOOL（初值 FALSE），程式載入後不會自行激磁或移動。目標座標是各軸的單位（直線軸 mm、旋轉軸 deg），第一次執行前請確認目標在行程內並降低 Feedrate；要停止時用 M86_Stop 或把 Xpower 設為 FALSE。

```
PROGRAM Motion
VAR
    pw  : M86_Power;
    all : M86_MoveLinear9;
    rd  : M86_ReadPosition;
END_VAR
pw(Machine := Robot, Enable := Xpower);                    (* 激磁：Xpower *)
all(Machine := Robot, Execute := Xgo9 AND pw.Status,     (* 按 Xgo9 送出一次移動 *)
    X := 120.0, Y := 80.0, Z := 25.0, U := 90.0, V := 45.0, W := 30.0,
    A := 10.0, B := 5.0, C := 180.0, Feedrate := 6000.0);
rd(Machine := Robot, Enable := TRUE);   (* rd.X .. rd.C：目前位置 *)
END_PROGRAM
```

G-code 也可以指定 U ~ C 軸，例如 `M86_GCode` 的 `Line := 'G1 X10 U90 C180 F3000'`。

12.8.3 在 C/C++ 中控制軸

使用者 C/C++ 程式（第 14 章）也可以控制軸：

函式	說明
`AXIS(AxisX)`	軸的編號（傳給下列函式）
`mc_power(ax, on)`、`mc_is_on(ax)`	激磁 / 解除、是否已激磁
`mc_move_abs(ax, pos, vel)`、`mc_move_rel(ax, dist, vel)`	絕對 / 相對移動（vel 為 0 用預設速度）
`mc_jog(ax, dir, vel)`	寸動：dir 為 +1 / −1 / 0；需持續呼叫，最後一次呼叫 0.6 秒後自動停止
`mc_stop(ax)`、`mc_home(ax)`、`mc_set_position(ax, pos)`、`mc_reset(ax)`	停止、原點復歸、設定位置、清除錯誤
`mc_position(ax)`、`mc_velocity(ax)`	目前位置、速度
`mc_state(ax)`、`mc_done`、`mc_busy`、`mc_error`、`mc_moving`、`mc_error_code`	狀態（0 閒置、1 執行中、2 完成、3 錯誤）
`M86_MACHINE(Gantry)`	Motion86 的 `Machine *`（不經 86LDmicro 的座標偏移與寸動管理）

*__注意__：mc_* 請在 `plc86_user_loop` / `plc86_user_before` / `plc86_user_after` 中呼叫，不要在 `plc86_user_ethercat`（中斷）中呼叫。

12.9 安全機制

86LDmicro 與 Motion86 內建下列保護，但它們不能取代硬體的緊急停止與安全迴路。

情況	動作
PLC STOP、看門狗	程式不再執行；所有機台停止、寸動結束；一般輸出保持最後的值（10.4、10.5 節）
暖重啟 / 冷重啟	暖重啟：RETAIN 變數保留，其他變數（包含輸出用的變數）回到初值；冷重啟：所有變數（含 RETAIN）回到初值。兩者都解除全部強制，RUN / STOP 狀態不變：下一個掃描起輸出依初值寫出（RUN 時再由程式重新計算）。重啟前請先停止機台（STOP 或 MC_Stop / M86_Stop）
寸動沒有持續更新	軸頁按鈕、MC_Jog、`mc_jog` 600 ms 沒有更新就自動停止
驅動器在激磁中離開 Operation Enabled（異常、Quick stop、STO、網路中斷）	整台機台停止，且**不會自行再激磁**；事件紀錄寫入 `Motion86: an axis' drive left operation enabled: the machine stopped (MC_Reset, MC_Power again)`，需 MC_Reset 再 MC_Power
解除激磁	先停止機台，再解除驅動器
驅動器未激磁時被手動移動	位置參考會偏移，事件紀錄寫入 `Motion86: an axis moved while its drive was off: …`；請重新原點復歸或 MC_SetPosition
原點復歸找不到原點開關	事件紀錄寫入 `Motion86: homing found no home switch`
軟體極限	勾選「限制行程」時，超出最小值 / 最大值的移動會被拒絕

***安全：MC_Stop / MC_Halt 為 Motion86 的立即停止，沒有減速斜坡。** 實機請設計外部緊急停止與安全迴路（切斷驅動器主電源或 STO）。

第一次在實機執行時，請先用低速、確認方向與行程，並準備緊急停止。

12.9.1 首次試車檢查表

實機第一次讓馬達轉動前，請逐項確認並打勾。建議兩人作業：一人操作電腦，一人站在緊急停止旁。

#	通電前	確認
A1	緊急停止接在硬體安全迴路（切斷驅動器主電源或 STO），按下後馬達確定無法出力	☐
A2	機構可移動範圍內無人員、工具、異物；行程兩端有機械擋塊或極限開關	☐
A3	馬達動力線、編碼器線、EtherCAT 線依驅動器手冊接妥，接地確實	☐
A4	驅動器參數（馬達型號、電子齒輪、極限）已依驅動器手冊設定	☐

#	通電、通訊	確認
B1	序列埠監看顯示 `EtherCAT: running, N slaves, … sync`，N 與實際從站數一致	☐
B2	SubDevice 頁的線上值會更新；驅動器的 Statusword（6041h）可讀	☐
B3	ENI 的 PDO 包含 6040h / 607Ah 輸出與 6041h / 6064h 輸入（11.3.6）	☐
B4	機台與軸的設定：單位、每轉脈衝數、每轉移動量、驅動器對應正確	☐

1. 先在**模擬**（F5）中用機台頁「線上」操作一次，熟悉按鈕位置與方向。
2. 上傳程式並連線，開啟「軸（試車）」視窗或軸頁「線上」。
3. **單軸**激磁，確認「已激磁」燈亮、位置不跳動。
4. 以**慢速寸動**（◄►）每次移動一小段，確認正方向；相反時解除激磁，勾選軸頁「反轉方向」，重新建置上傳。
5. 用小距離（例如 5 mm）的**相對移動**確認實際移動量與命令相同；不同時檢查每轉移動量。
6. 執行**原點復歸**，確認停在正確位置。
7. 每軸都確認後，才在機台頁做「一起移動」，速度由低往高逐步提高。
8. 最後才執行自己的 MC / M86 程式。

#	第一次動作	確認
C1	每軸方向正確	☐
C2	每軸移動量正確（命令 5 mm，實際 5 mm）	☐
C3	按 STOP、PLC STOP、拔 EtherCAT 線時，軸都會停止	☐
C4	硬體緊急停止可停止所有軸	☐
C5	原點復歸正常	☐

第 13 章　86HMI 人機介面

86HMI 是 86Duino 的觸控人機畫面。畫面（頁面、按鈕、燈號、數值……）以 86Duino IDE 的 **86HMI 編輯器**製作；86LDmicro 負責把畫面上的元件與 PLC 變數**綁定**，並在模擬與線上時讓畫面與程式連動。本章依序說明啟用、綁定、各元件的互動方式、文字 / 趨勢 / 警報 / 換頁、模擬，以及資源上傳與語系按鈕。

工作	在哪裡做	本章
畫頁面、放元件、設定外觀與語系	86HMI 編輯器（86Duino IDE「工具 > 86HMI」，或 86HMI 頁的 **86HMI 編輯器**）	13.1、13.6
元件 ↔ 變數的綁定	86LDmicro 的 86HMI 頁、專案樹「86HMI」、階梯圖元件的右鍵	13.2 ~ 13.4
在電腦上試操作	86HMI 模擬視窗（模擬 F5 時自動開啟）	13.5
把畫面送到控制器	上傳（F8），或只上傳資源	13.6

13.1　啟用 86HMI

86HMI 的畫面存在 sketch 的 `data\_project.hpj`。86HMI 編輯器按 **產生程式碼**（Code Generation）時，會寫出 `myhmi.cpp` / `myhmi.h`、畫面用到的圖片與字型（`data\Resource`），並在 `.ino` 加上 `#include "myhmi.h"` 與 `Hmi.begin();`。之後 86LDmicro 讀取這個 `.hpj`，列出所有頁面與元件。

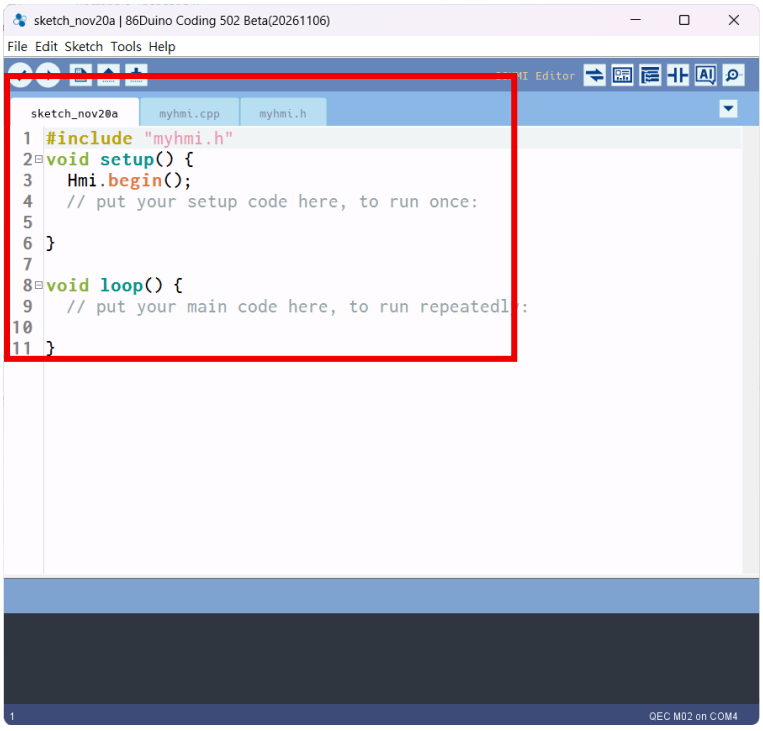


圖 13-0　86HMI 產生程式碼

13.1.1　從 86Duino IDE 開啟 86HMI 編輯器

在 86Duino IDE 開啟 sketch 後按「**工具 > 86HMI**」即開啟 86HMI 編輯器。這與 1.1 時代的做法相同。

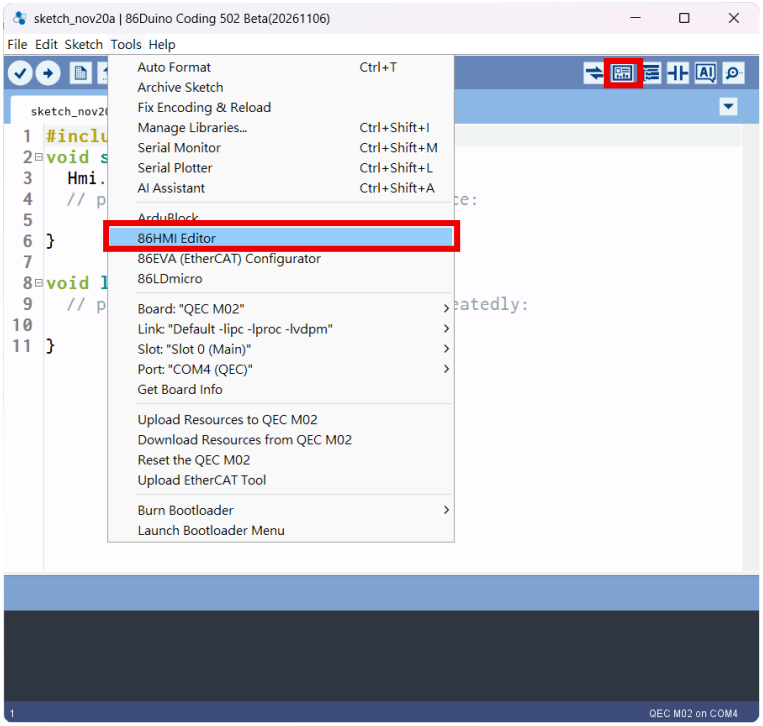


圖 13-1　86Duino IDE 的「工具」選單：86HMI 與 86LDmicro

13.1.2　從 86LDmicro 開啟 86HMI 編輯器（建議）

1. 專案樹雙擊「**86HMI**」，或「**專案 > 86HMI 畫面與綁定**」，開啟 86HMI 頁。

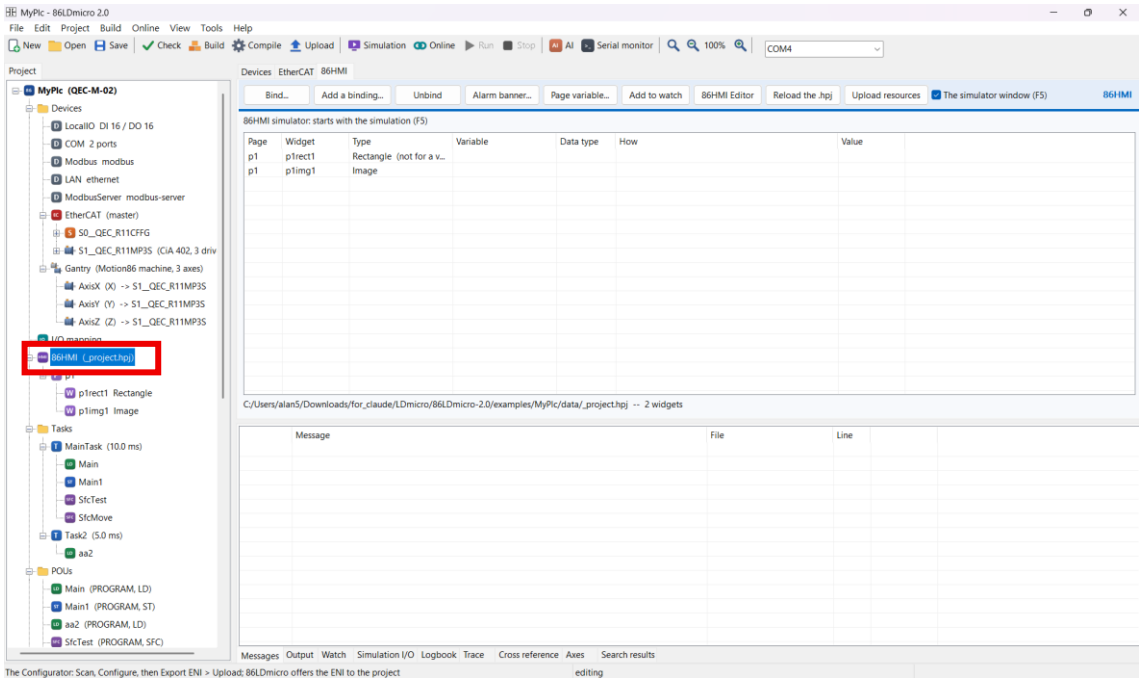


圖 13-2-1　86LDmicro 從專案 > 86HMI 畫面與綁定

2. 按 **86HMI 編輯器**：以與 86Duino IDE「工具 > 86HMI」相同的方式開啟編輯器，開的是本專案 sketch 的 `data\_project.hpj`；sketch 還沒有畫面時，建立一個新畫面。

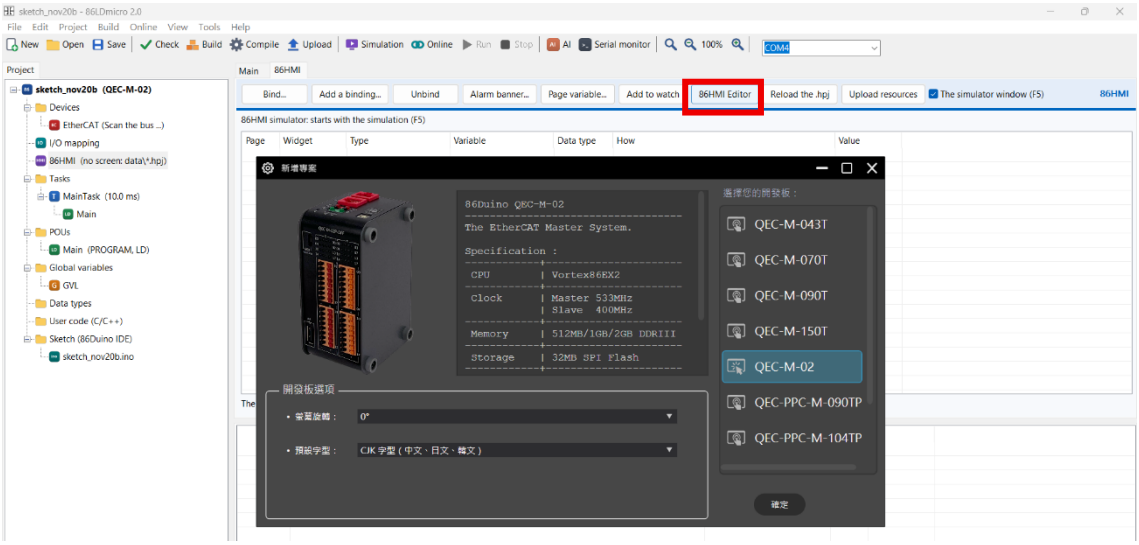


圖 13-2-2　86LDmicro 和 86HMI 編輯器，開啟新專案

3. 在編輯器中新增頁面，放置元件（Button、Switch、LED、Number、Label、Image……），設定名稱（例如 `p1b1`、`p1led1`）與外觀。

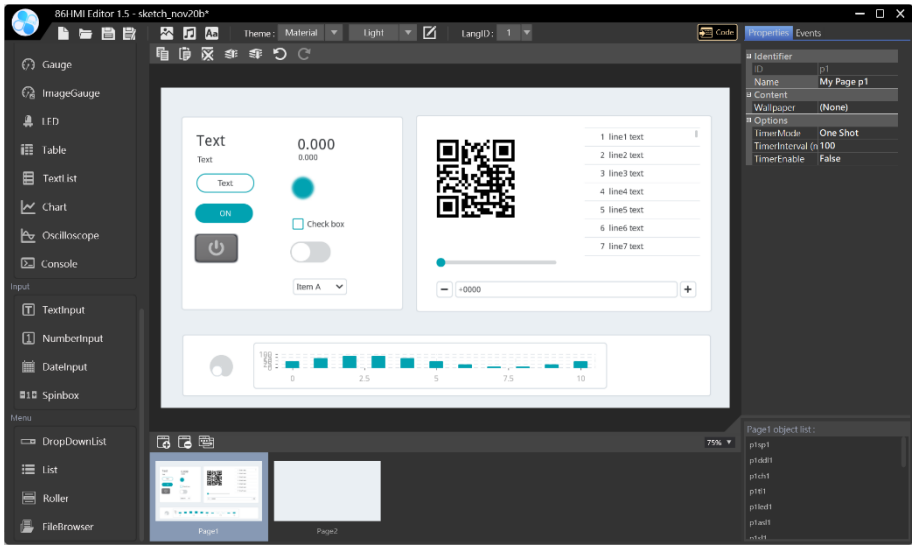


圖 13-2-3　86HMI 編輯器

4. 按 **產生程式碼**：存回 `data\_project.hpj`、`myhmi.cpp`、`myhmi.h` 與 `data\Resource`，並修補 `.ino`。

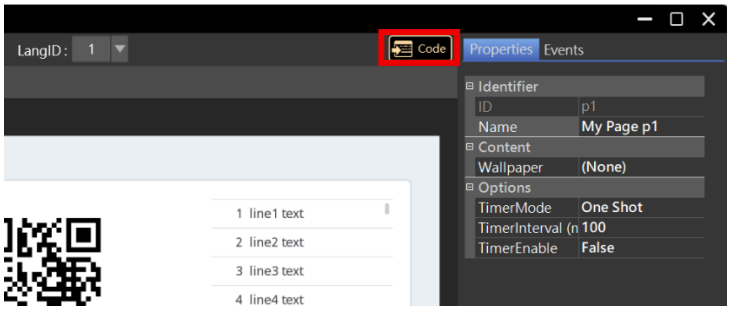


圖 13-2-4　86HMI 編輯器 4. 按 產生程式碼

5. 回到 86LDmicro：86HMI 頁與專案樹自動重新讀取（狀態列顯示「86HMI：畫面已存回 sketch（data_project.hpj、myhmi.cpp / myhmi.h）：重新建置即可使用」）。

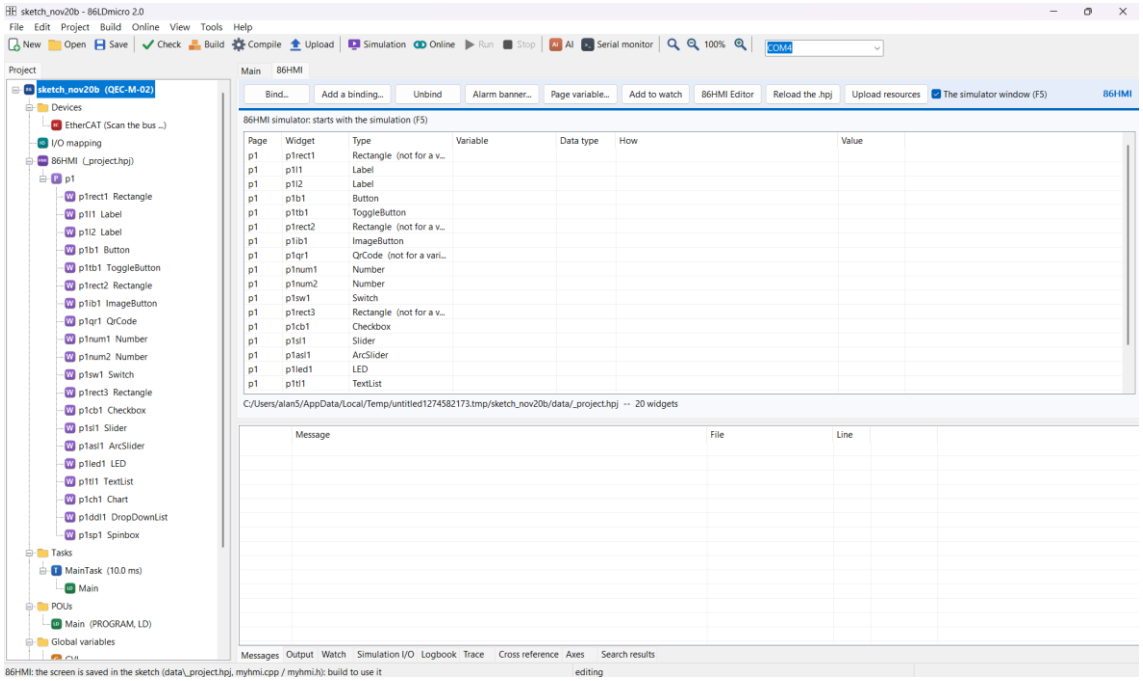


圖 13-2-5　回到 86LDmicro：86HMI 頁與專案樹自動重新讀取

若沒有更新，按 **重新讀取 .hpj**。

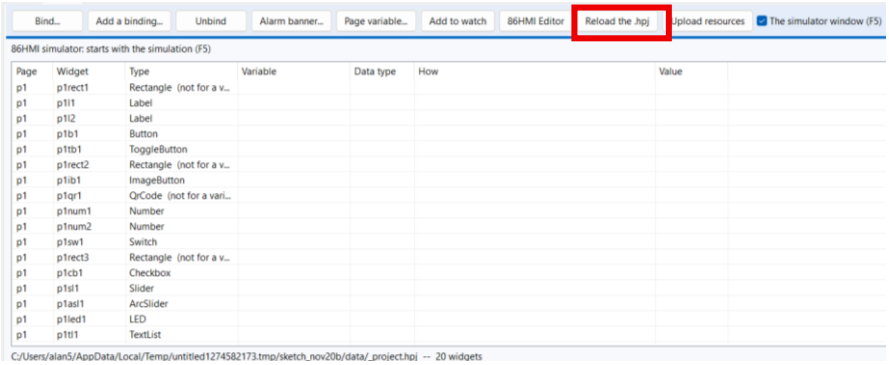


圖 13-2-6　回到 86LDmicro：86HMI 若沒有更新，按 重新讀取 .hpj

***提示**：86LDmicro 啟動 86HMI 編輯器時使用 86Duino IDE 內附的編輯器，所以需要正確設定 IDE 資料夾（2.6 節「選項」）。從 86Duino IDE 啟動 86LDmicro 時會自動使用該 IDE。

13.1.3 86HMI 頁

86HMI 頁列出 `.hpj` 中的每一個元件；已綁定的元件**每個綁定一列**。上方按鈕與表格欄位如下：

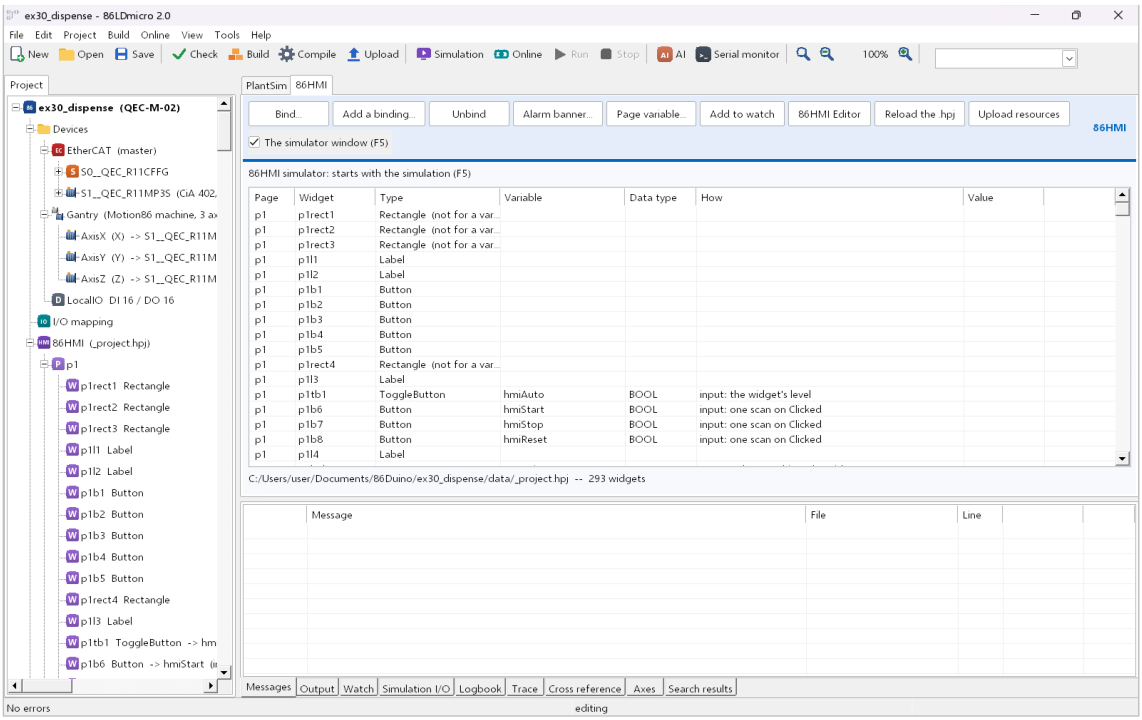


圖 13-3　86HMI 頁：按鈕列與元件 / 綁定清單

按鈕	用途
綁定...	修改選取的那一列綁定；元件尚未綁定時新增一個（13.2 節）
新增綁定...	在同一個元件再加一個綁定（13.2.3 節）
解除綁定	只移除選取的那一列綁定
警報橫幅...	BOOL 變數控制的警報橫幅（13.4.3 節）
頁面變數...	整數變數決定目前顯示的頁面（13.4.4 節）
加入監看	把選取列的變數加入「監看」視窗（9.3 節）
86HMI 編輯器	開啟本專案的 `data\_project.hpj`（13.1.2 節）
重新讀取 .hpj	在外部改過畫面後重新讀取頁面與元件
上傳資源	只上傳 `data\Resource`（13.6 節）
模擬視窗（F5）（勾選框）	勾選時，模擬（F5）會自動開啟 86HMI 模擬視窗（13.5 節）

欄位	內容
頁面	元件所在的頁面（p1、p2…）
元件	86HMI 編輯器中的元件名稱（例如 p1b1）
型別	元件類型（Button、Switch、LED、Number、Image (x)…）；不能綁定變數的元件顯示「（不可綁定變數）」
變數	綁定的全域變數
資料型別	變數的型別（BOOL、INT、REAL…）
用法	輸入或輸出及其方式（13.3 節）
數值	模擬或線上時變數的即時值

頁面最下方顯示 `.hpj` 的路徑與元件數。sketch 沒有畫面時，專案樹顯示「86HMI（無畫面：data*.hpj）」，請先依 13.1.2 節建立畫面。

13.2 綁定元件

綁定就是指定一個元件與哪一個全域變數連動，以及資料往哪個方向流：**輸入**由畫面設定變數（按鈕、開關、輸入數值），**輸出**把變數顯示在畫面（燈號、數值、文字、圖片位置）。綁定存在 `project.86ld` 的 `bindings`，上傳到控制器的程式與模擬視窗使用同一份設定。

13.2.1 開啟綁定表單的方法

- 86HMI 頁選一個元件（或它的某一列綁定），按 **綁定...**，或直接雙擊該列。
- 專案樹「86HMI > 頁面 > 元件」雙擊。
- 從變數開始：在階梯圖元件、監看列或 ST 中的變數上按右鍵 > 「**<變數> 的 86HMI 綁定...**」，再選元件、警報橫幅或顯示的頁面（13.4.5 節）。

13.2.2 綁定表單（依元件類型）

表單的標題為「86HMI 綁定」，第一行說明是哪一個元件（名稱、類型、頁面）。欄位依元件類型而不同：

欄位	說明
變數	全域變數。可從下拉選，或輸入新名稱：新名稱會自動加入 GVL（按鈕 / LED 為 BOOL，數值為 INT）
用法	輸入（由畫面設定變數）或輸出（在畫面顯示變數）；選項依元件類型而定（13.3 節）。事件只在一個掃描週期為 TRUE
比例	數值元件：變數 = 元件值 × 比例（輸出時 ÷ 比例）；1 表示不換算
文字（Label / Table / TextList）	範本（溫度 `%d C`，%d 代表數值），或每個數值一則訊息，一行一則：`0`=停止、`1`=運轉、`2`=警報（13.4.1 節）
欄，列 / 行（Table / TextList）	要寫入的儲存格（Table）或行（TextList）
通道（1..4），週期（ms）（Chart / Oscilloscope）	趨勢：變數取樣到哪一個通道、每隔多少 ms 取樣一次（13.4.2 節）
啟用（BOOL，可省略）	文字與趨勢元件：空白 = 一直更新；填一個 BOOL 時，只有它為 TRUE 才更新元件

Button（按鈕）：表單只有「變數」「用法」「比例」。新增時「變數」空白，用法預設為「輸入：變數跟隨元件狀態」。

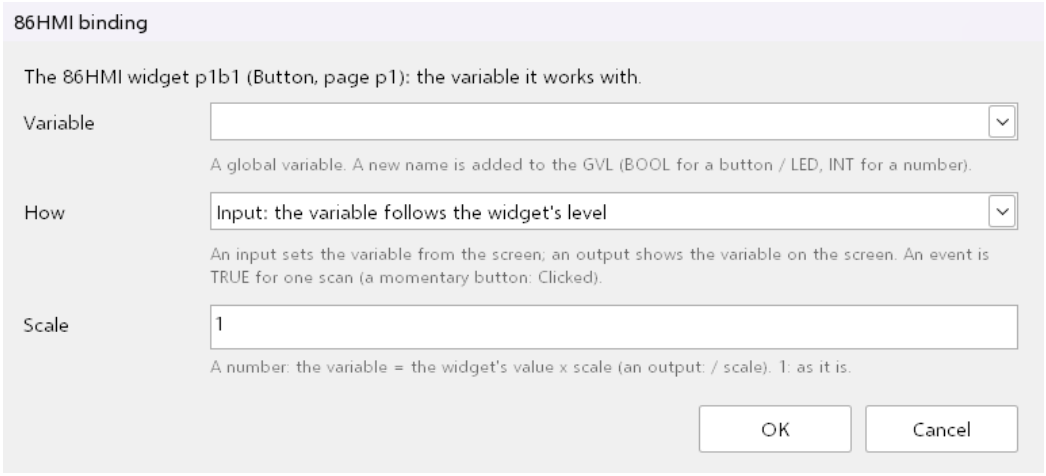


圖 13-4　綁定 Button：變數、用法、比例

Switch（開關）：切換 ON / OFF 後保持，通常綁定一個內部旗標（例如 `Rstart`）。

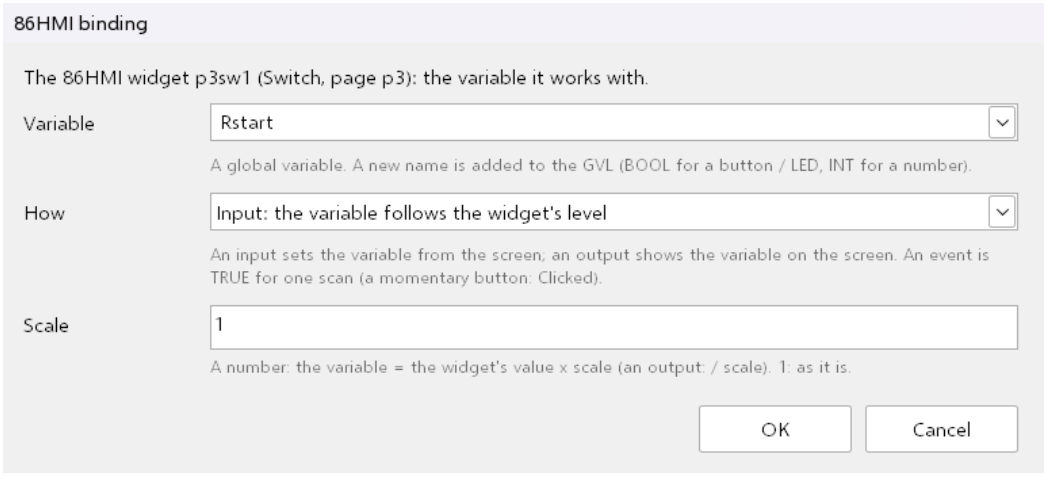


圖 13-5　綁定 Switch：p1sw1 → Rstart，輸入：變數跟隨元件狀態

Number（數值）：只能輸出，把變數的值顯示在畫面上；比例可做單位換算。

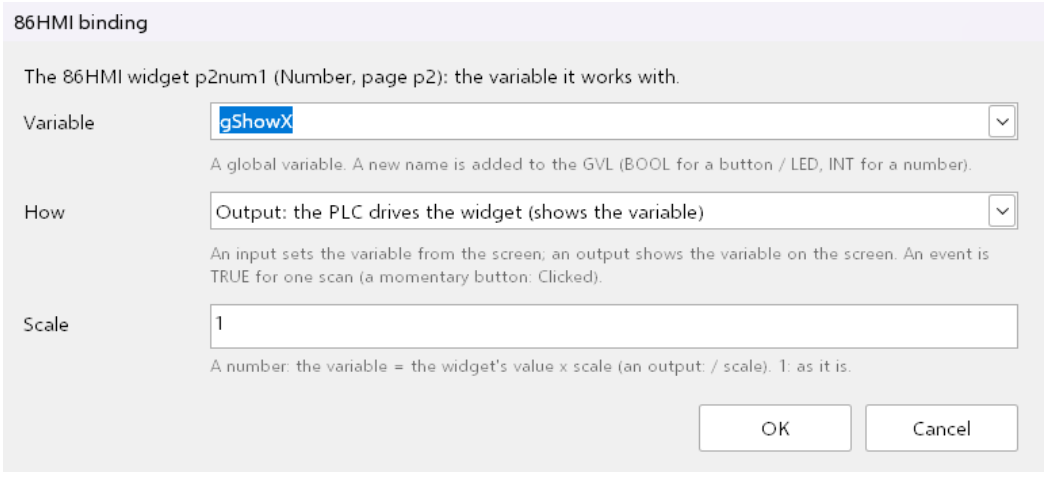


圖 13-6　綁定 Number：輸出，PLC 驅動元件（顯示變數）

13.2.3　圖解：把按鈕 p1b1 綁到 Rstop

1. 在 86HMI 頁選元件 **p1b1**（Button），按 **綁定...**。
2. 「變數」輸入或選擇 `Rstop`；「用法」選 **輸入：變數跟隨元件狀態**（按住時 TRUE）。
3. 按 **確定**。表格中 p1b1 的「變數」欄顯示 Rstop，專案樹的 86HMI 下也同步顯示（例：`p1b1 Button -> Rstop`）。
4. 在階梯圖中把 `Rstop` 與實體停止按鈕一起使用（見 13.3 節的注意）。

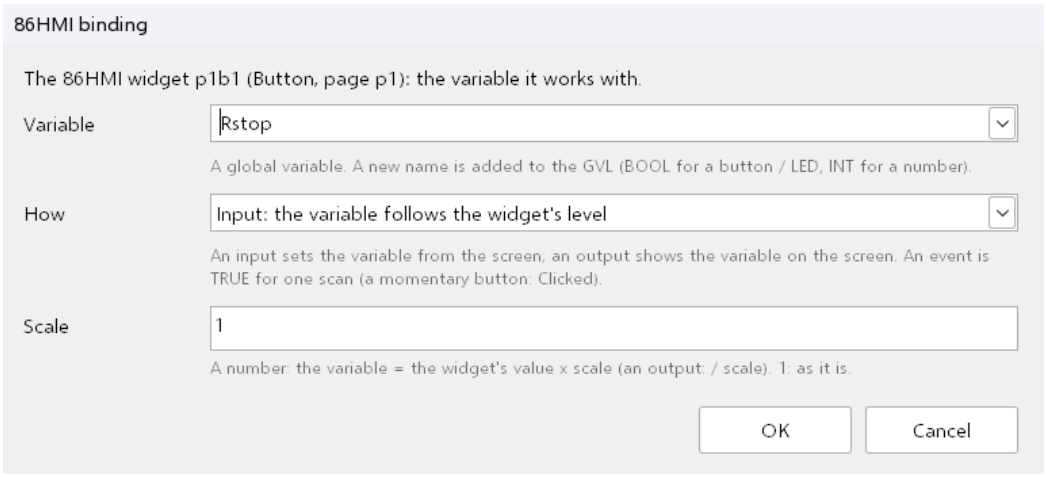


圖 13-7　綁定表單：p1b1 → Rstop，輸入：變數跟隨元件狀態

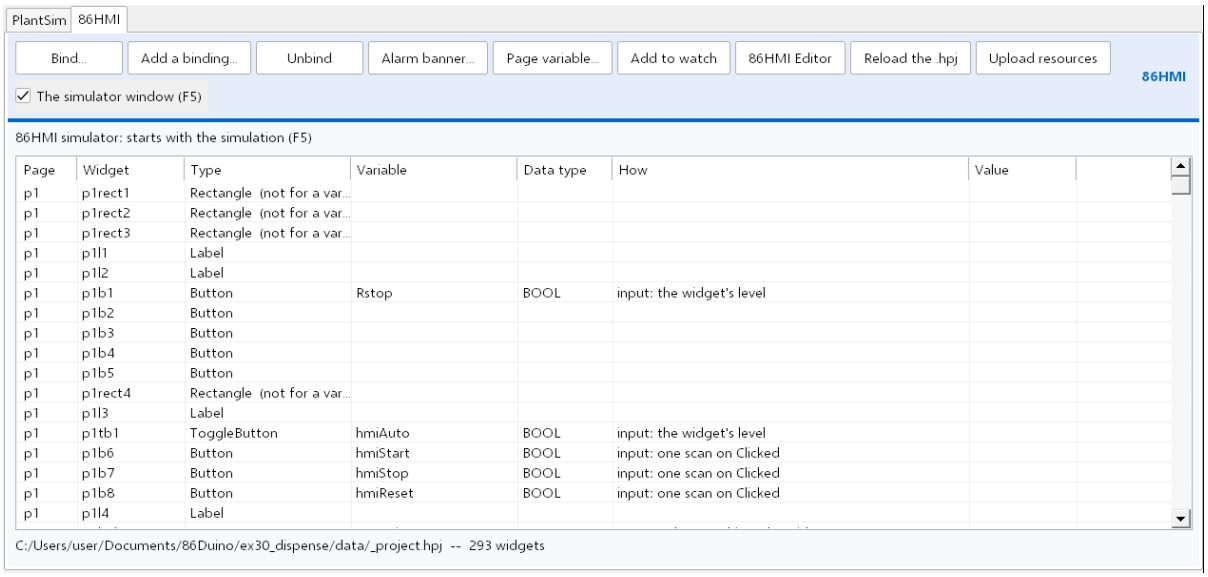
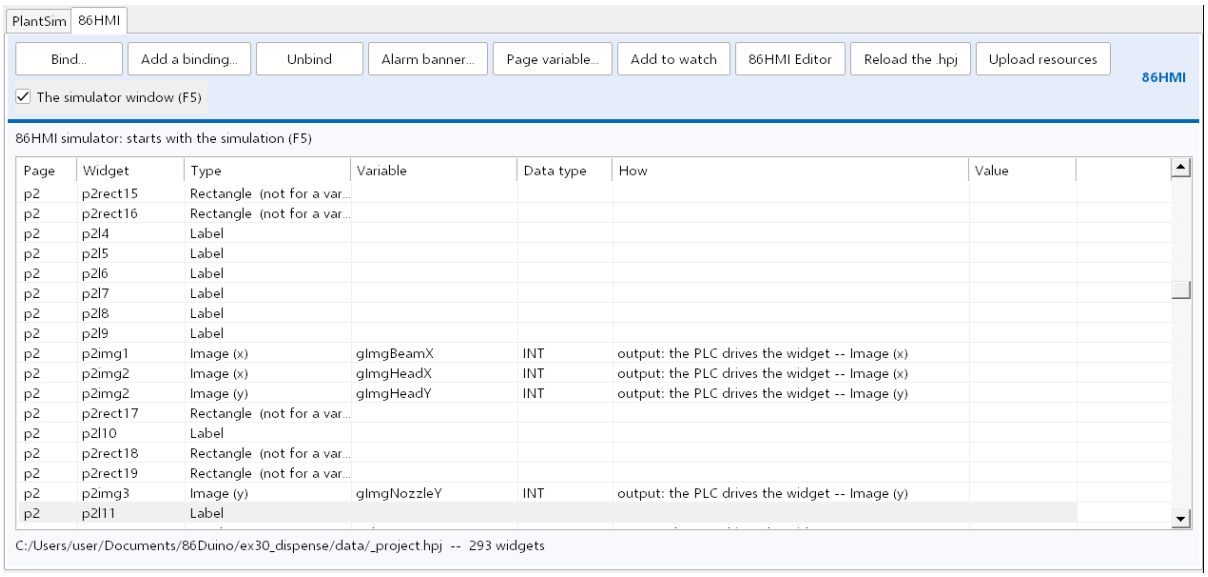


圖 13-8　綁定完成：表格與專案樹同時顯示 p1b1 → Rstop、p1led1 → Yout、p1sw1 → Rstart、p2num1 → Mposition

13.2.4 一個元件多個綁定

一個元件可以同時綁定好幾個變數，例如一張 Image 同時有「顯示 / 隱藏」「x 位置」「y 位置」；一個 Chart 的每個通道；一個 Table 的每個儲存格；任何元件再加一個「啟用」。86HMI 頁中**每個綁定一列**，同一個元件會出現好幾列。



Page	Widget	Type	Variable	Data type	How	Value
p2	p2rect15	Rectangle (not for a var...				
p2	p2rect16	Rectangle (not for a var...				
p2	p2l4	Label				
p2	p2l5	Label				
p2	p2l6	Label				
p2	p2l7	Label				
p2	p2l8	Label				
p2	p2l9	Label				
p2	p2img1	Image (x)	gImgBeamX	INT	output: the PLC drives the widget -- Image (x)	
p2	p2img2	Image (x)	gImgHeadX	INT	output: the PLC drives the widget -- Image (x)	
p2	p2img2	Image (y)	gImgHeadY	INT	output: the PLC drives the widget -- Image (y)	
p2	p2rect17	Rectangle (not for a var...				
p2	p2l10	Label				
p2	p2rect18	Rectangle (not for a var...				
p2	p2rect19	Rectangle (not for a var...				
p2	p2img3	Image (y)	gImgNozzleY	INT	output: the PLC drives the widget -- Image (y)	
p2	p2l11	Label				

圖 13-9　ex30 的 86HMI 頁：p2img1、p2img2 各有 Image (x) 與 Image (y) 兩個綁定，每個綁定一列

1. 選該元件的任一列，按 **新增綁定...**。
2. 選「變數」與「用法」。表單上方說明這個元件已有幾個其他綁定（「已有 N 個其他綁定：新的綁定會加在它們旁邊」）。
3. 按 **確定**，清單多出一列；新變數自動加入 GVL。
4. 要修改某一個綁定：選那一列按 **綁定...**（或雙擊）。要移除：選那一列按 **解除綁定**，只會移除那一個。

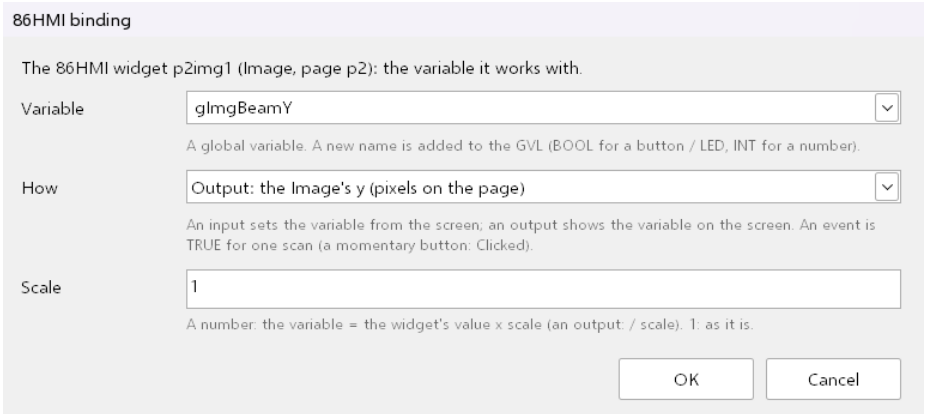


圖 13-10　在 p2img1 新增「輸出：Image 的 y 位置（頁面上的像素）」綁定（變數 gImgBeamY）

為了避免互相衝突，以下情況會被擋下並提示：

- 同一個元件重複相同的用法（例如兩個 x 位置、Chart 的同一個通道、Table 的同一個儲存格）。
- 同一個值同時是輸入又是輸出（例如 Switch 綁成輸入，又綁一個輸出）。
- 一個變數已經綁在別的元件上：請先到那個元件解除（啟用用的 BOOL 例外，可綁在多個元件上）。

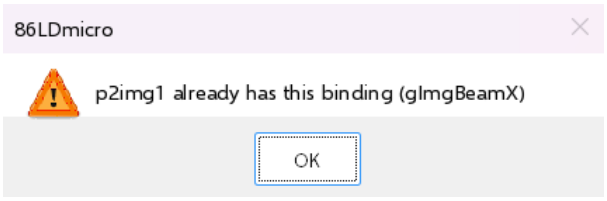


圖 13-11　p2img1 已經有 x 位置綁定時，再新增一個會被擋下

*提示：專案樹中，元件後面列出它所有的綁定；雙擊有多個綁定的元件，會讓你選要修改哪一個或「（新增一個綁定）」；在專案樹按「解除綁定」會詢問是否移除全部。

13.3 元件與互動方式

「用法」清單只列出該元件支援的方式。下表整理常用元件（元件名稱為 86HMI 編輯器中的類型）：

元件	可選的用法	互動
Button、ImageButton	輸入：變數跟隨元件狀態 輸入：一個掃描週期 TRUE，當 Clicked（事件） 輸入：一個掃描週期 TRUE，當 Pressing（事件）	跟隨狀態：按住 TRUE、放開 FALSE（像實體按鈕；ImageButton 只有事件） 事件：按一下時變數 TRUE **一個掃描週期**，適合「啟動」「重置」等瞬間指令
ToggleButton、ImageToggleButton、Checkbox	輸入：變數跟隨元件狀態 輸入：狀態反相 輸入：事件（Checked / Unchecked） 輸出：PLC 驅動元件	切換後保持；也可由 PLC 設定狀態
Switch	輸入：變數跟隨元件狀態 輸入：狀態反相（Off） 輸入：事件（On / Off） 輸出：PLC 驅動元件	切換 ON / OFF 後保持（例：p1sw1 → Rstart）
LED	輸出：PLC 驅動元件（顯示變數）	變數 TRUE 時亮（例：p1led1 → Yout）
Number	輸出：PLC 驅動元件	顯示變數的值；比例可做換算
NumberInput、Spinbox	輸入：變數跟隨元件狀態 輸入：事件（Applied / Canceled，僅 NumberInput） 輸出：PLC 驅動元件	可同時有一個輸出與一個輸入（兩個變數）：PLC 改變輸出變數時寫入畫面（例如載入配方值），操作員輸入的值由輸入變數讀回；輸出只在值改變時寫入，不會蓋掉正在輸入的值
Slider、ArcSlider、ImageSlider	輸入：變數跟隨元件狀態、事件（Dropped / Dragging） 輸出：PLC 驅動元件	拖曳設定數值
Bar、ArcBar、Gauge、ImageBar、ImageGauge	輸出：PLC 驅動元件 輸入：變數跟隨元件狀態	以長條、弧形或指針顯示數值（通常用輸出）
DropDownList、List、Roller	輸入：變數跟隨元件狀態（選取的項目，整數）、事件（Selected）	操作員選擇一個項目

元件	可選的用法	互動
Image	輸出：PLC 驅動元件（顯示／隱藏） 輸出：Image 的 x 位置（頁面上的像素） 輸出：Image 的 y 位置（頁面上的像素）	顯示：變數 TRUE 時顯示、FALSE 時隱藏 x / y：變數（INT）是圖片在頁面上的像素位置，PLC 改變數圖片就移動（例：ex30 的位置頁）
Label、Table、TextList	輸出：以文字顯示數值	範本或每個數值一則訊息（13.4.1 節）
Chart、Oscilloscope	輸出：趨勢圖（變數取樣到通道）	每個通道一個綁定（13.4.2 節）
任何元件（含 Label、Table、Chart）	輸出：元件啟用（TRUE）或變灰（FALSE）	BOOL 為 FALSE 時元件變灰、不能操作，讓 PLC 決定目前哪些按鈕能按；同一個 BOOL 可以綁在多個元件上；只在值改變時寫入

13.3.1 輸入的三種方式

用法	變數的值	適合
輸入：變數跟隨元件狀態	與元件的狀態相同（按住 TRUE、放開 FALSE；開關 ON = TRUE）	寸動、需要「按住才動作」、開關型的選擇
輸入：狀態反相	與元件狀態相反	常閉型的按鈕（按下 = FALSE）
輸入：一個掃描週期 TRUE，當 Clicked（事件）	事件發生的那一次，TRUE 一個掃描週期，之後自動回到 FALSE	啟動、重置、確認警報等瞬間指令；不需要自己做上升緣

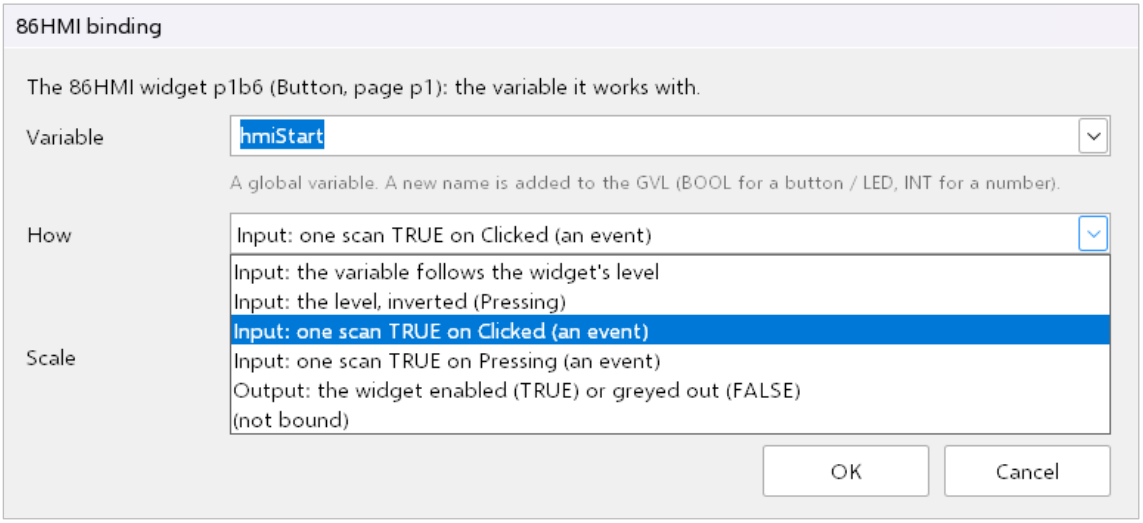


圖 13-12　Button 的「用法」清單：跟隨狀態、Clicked 事件、Pressing 事件

*提示：事件（一個掃描週期 TRUE）在**讀取它的那個工作**掃描過一次後清除，所以放在較慢的工作（例如 100 ms 的 SlowTask）也讀得到。同一個事件被兩個工作讀取時只保證第一個工作，建置時會警告：請只在一個工作讀取，必要時用 SET 鎖存。

*注意：已對應實體輸入的變數不能當 HMI 的輸入（下一個掃描會被實體輸入覆蓋）。請綁定內部變數（例如 `Rstart`、`Rstop`），再於程式中與實體按鈕一起使用。

例：實體啟動 `Xstart`（本機 X00）與 HMI 開關 `Rstart` 並聯，實體停止 `Xstop` 與 HMI 停止按鈕 `Rstop` 串聯：

```
NETWORK 'start/stop'
    [ NO(Xstart) | NO(Rstart) | NO(Rrun) ] NC(Xstop) NC(Rstop) OUT(Rrun)
NETWORK 'lamp'
    NO(Rrun) OUT(Yrun)
```

模擬確認：`Rstart` 為 TRUE 時 `Rrun`、`Yrun` 變 TRUE 並保持；`Rstop` 為 TRUE 時 `Rrun` 變 FALSE。

13.3.2 啟用（Enable）：讓 PLC 決定哪些按鈕能按

任何元件都可以加一個「**輸出：元件啟用（TRUE）或變灰（FALSE）**」綁定。例如運轉中把「原點復歸」按鈕變灰，避免誤按：

1. 選該元件按 **新增綁定...**（元件已有其他綁定時）或 **綁定...**。
2. 「變數」輸入一個 BOOL（例如 `gCanHome`），「用法」選 **輸出：元件啟用（TRUE）或變灰（FALSE）**，按確定。
3. 在程式中寫 `gCanHome := NOT gRunning;`（或對應的階梯圖）。同一個 BOOL 可以綁在多個元件上。

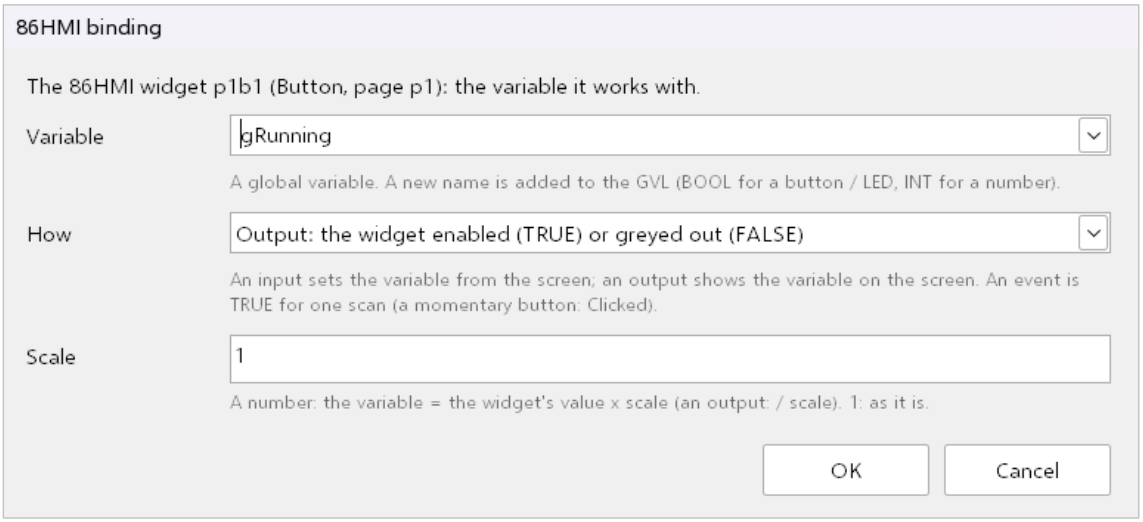


圖 13-13　啟用綁定：BOOL 為 FALSE 時元件變灰

13.4 文字、趨勢、警報與換頁

13.4.1 文字：Label、Table、TextList

文字元件的用法為「**輸出：以文字顯示數值**」。「文字」欄有兩種寫法：

寫法	範例	畫面顯示
範本：`%d` 代表數值	`溫度 %d C`	溫度 25 C

每個數值一則訊息，一行一則	`0=停止`
	`1=運轉`
	`2=警報`
		變數為 1 時顯示「運轉」

- **Table**：另填「欄，列」指定儲存格；每個儲存格一個綁定。
- **TextList**：另填「行」。僅實機：86HMI 模擬器沒有 TextList 的 API，模擬時不會更新。
- 「啟用（BOOL，可省略）」：空白 = 一直更新；填一個 BOOL 時，只有它為 TRUE 才更新文字。

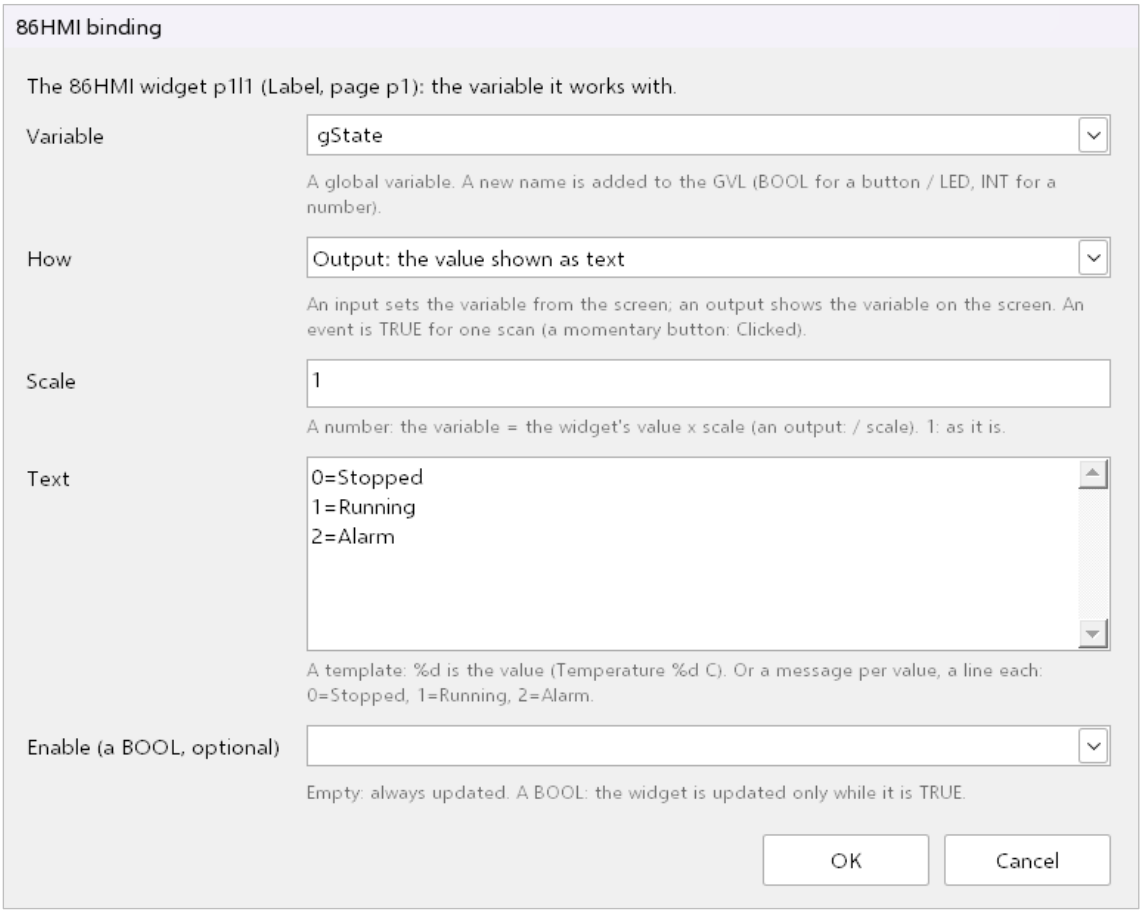


圖 13-14　Label 的文字綁定：每個數值一則訊息

13.4.2 趨勢：Chart、Oscilloscope

用法為「**輸出：趨勢圖（變數取樣到通道）**」；「通道（1..4），週期（ms）」例如 `1, 100` 表示每 100 ms 把變數取樣到第 1 通道。一個 Chart 最多 4 個通道，每個通道一個綁定（用 **新增綁定...** 加第二個通道）。表單上方會顯示這個 Chart 的點數與通道數。

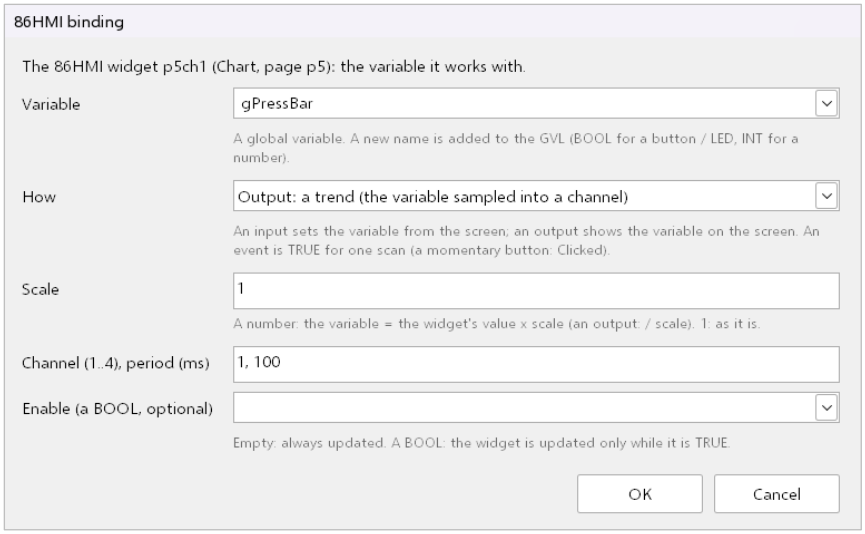


圖 13-15　Chart 的趨勢綁定：通道與取樣週期

13.4.3 警報橫幅

按 86HMI 頁的 **警報橫幅...**（或專案樹「86HMI」右鍵）：一個 BOOL 變數為 TRUE 時顯示橫幅、FALSE 時移除。

欄位	說明
變數（BOOL；空白：移除）	控制橫幅的變數；新名稱以 BOOL 加入 GVL
位置（0..15）	橫幅的位置：不同位置的橫幅可同時顯示
文字	橫幅上的訊息，例如「緊急停止已按下」
啟用（BOOL，可省略）	空白 = 一直有效

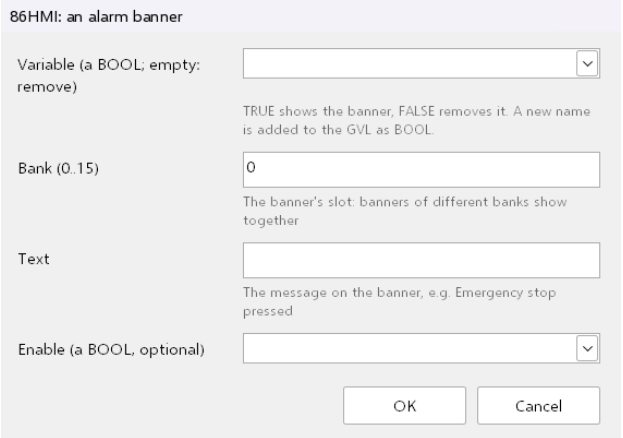


圖 13-16　「86HMI：警報橫幅」：變數、位置、文字

13.4.4　換頁：頁面變數

按 **頁面變數...**：一個整數變數決定目前顯示的頁面（1 = p1、2 = p2...），讓 PLC 主動切換畫面（例如發生警報時切到警報頁）。新名稱以 INT 加入 GVL。

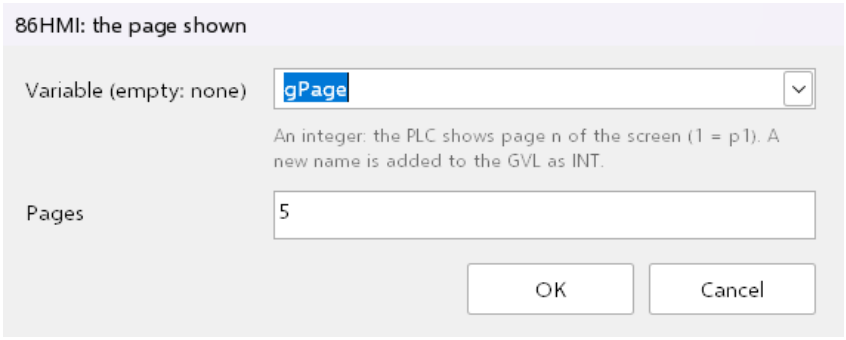


圖 13-17　「86HMI：顯示的頁面」：頁面變數與頁數

*提示：一般的換頁建議直接用 86HMI 編輯器的按鈕腳本「Page Change」（不經過 PLC）。「頁面變數」用於 PLC 主動切頁；切頁後請把變數設回 0，下次才能再切。

13.4.5　從變數綁定

在階梯圖元件（或監看列、ST 中的變數）上按右鍵 >「**<變數> 的 86HMI 綁定...**」，在清單中選元件（後面顯示它目前的綁定）、「警報橫幅」或「顯示的頁面」，按確定後再填用法。選「（無：移除這個變數的 86HMI 綁定）」可移除該變數的所有 86HMI 綁定。

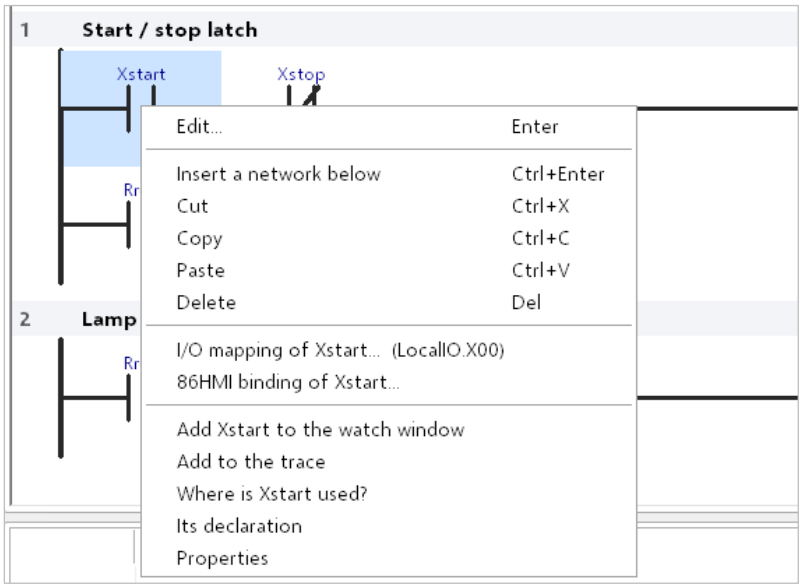


圖 13-18　階梯圖元件的右鍵選單：「Rstop 的 86HMI 綁定...」

*提示：86HMI 畫面的更新在第一個工作的掃描之後執行，不算在掃描時間與看門狗內（標籤文字排版、換頁可能要數十 ms）。

13.5 模擬與線上

模擬（F5）時，sketch 有 86HMI 畫面就自動開啟 **86HMI 模擬視窗**：在電腦上直接按按鈕、切換開關，PLC 程式即時反應；LED、數值、文字、圖片位置、警報、頁面與趨勢顯示 PLC 的值。線上（Ctrl+F5）時，模擬視窗顯示控制器上的值。



圖 13-19　86HMI 模擬視窗：在電腦上直接操作畫面，PLC 程式即時反應（範例：加壓站監控）

13.5.1 圖解：模擬中操作 86HMI

1. 確認 86HMI 頁的「**模擬視窗（F5）**」已勾選。

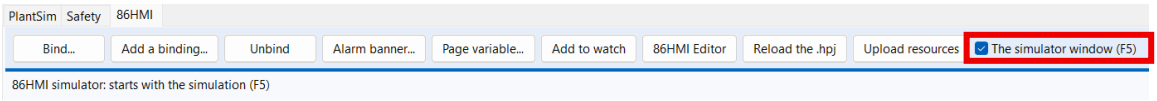


圖 13-20-1　模擬視窗（F5）」已勾選

2. 按 **模擬**（F5），模擬視窗自動開啟；86HMI 頁上方顯示模擬視窗的連線狀態。

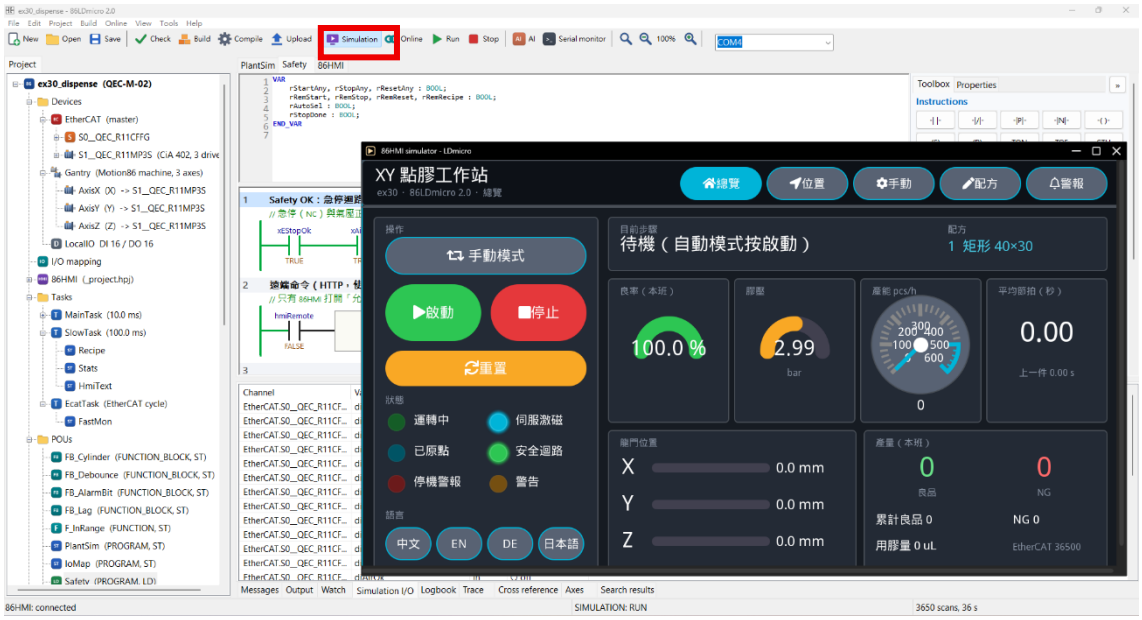


圖 13-20-2　模擬中：86LDmicro 的階梯圖與 86HMI 模擬視窗並排

3. 按住畫面上的按鈕：綁定的變數變 TRUE，階梯圖中對應的接點變綠；86HMI 頁「數值」欄同步更新。

4. PLC 輸出變數改變時，綁定的 LED、數值、圖片同步更新。

5. 再按 F5 結束模擬，模擬視窗一起關閉。

要做的事	**操作**
關閉（不要自動開啟）模擬視窗	取消 86HMI 頁的「模擬視窗（F5）」勾選
模擬中重新開啟模擬視窗	「線上 > 86HMI 模擬視窗（sketch 的 .hpj）」
改了綁定	綁定在下次啟動模擬（F5）時生效：先結束模擬再按 F5
線上時看控制器的值	Ctrl+F5 連線，開啟模擬視窗

***注意：已知問題**（2.0.0）：86HMI 模擬時，ToggleButton 切換狀態正常，但按鈕上的文字不會在「未按下」與「按下」兩組文字之間切換（實機上由 86HMI 產生的程式負責更新文字）。不影響 PLC 變數。

13.6 資源上傳與語系

13.6.1 上傳 86HMI 資源

86HMI 編輯器「產生程式碼」時，畫面用到的圖片、字型與畫面資料寫到 sketch 的 `data\Resource`。這些**資源**存在控制器上，和程式分開（一般的上傳說明見 10.6 節）。

情況	做法
第一次上傳、或程式與畫面都改了	按 **上傳**（F8）：程式與 `data\Resource` 一起上傳
只改了圖片、字型、畫面外觀，PLC 程式沒改	按 **上傳資源**：只送資源，不編譯，控制器上的程式不變（比 F8 快）

1. 工具列確認 COM 埠（例如 COM4）。
2. 在 86HMI 頁按 **上傳資源**，或「**建置 > 上傳 86HMI 資源（data\Resource）**」。

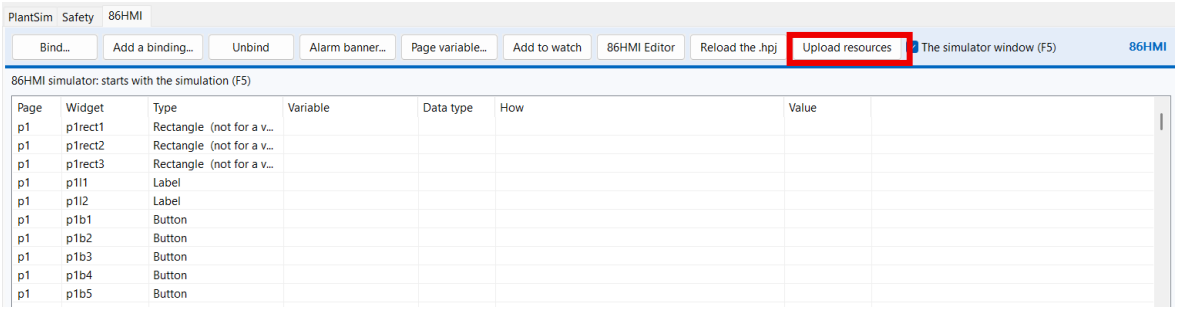


圖 13-21-1　上傳 86HMI 資源按鈕

3. 「輸出」視窗出現「== 上傳資源到 COM4: ...\data\Resource ...」與進度，最後「資源上傳完成」。
4. 原本在線上時，上傳前會自動離線，完成後自動重新連線。

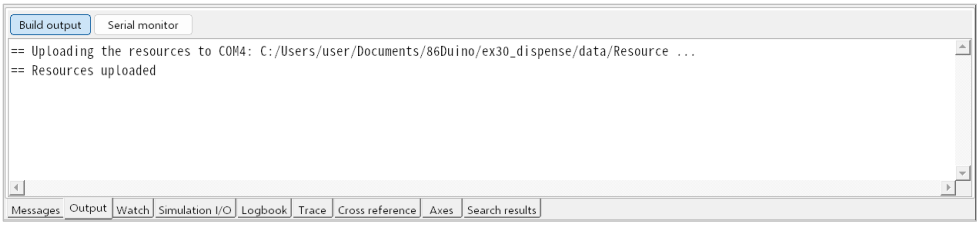


圖 13-21-2　上傳 86HMI 資源：「輸出」視窗的過程與「資源上傳完成」

*提示：這個功能與 86Duino IDE 的「工具 > 上傳資源檔至 QEC M02」（英文介面為 Upload Resources to QEC M02）相同。命令列：`86ldmicro-cli upload-res <專案> <COM>`。

出現「沒有可上傳的資源：... 不存在或是空的」時，表示 sketch 沒有 `data\Resource` 或是空的：先在 86HMI 編輯器開啟 .hpj 按「產生程式碼」。

13.6.2　語系切換按鈕

86HMI 畫面可以有多個語系（在 86HMI 編輯器中設定）。要讓操作員在控制器上按按鈕切換語系，**必須在 86HMI 編輯器中為按鈕寫切換腳本**；86LDmicro 的綁定只處理變數，不會替你切換語系。

1. 在 86HMI 編輯器選語系按鈕（例如「中文」），在它的事件（Clicked）加入腳本 `Hmi.setLanguage(1);`（數字為語系編號，依編輯器中語系的順序 1、2、3...）。
2. 其他語系按鈕依序設定 `Hmi.setLanguage(2);`、`Hmi.setLanguage(3);`...
3. 按 **產生程式碼**，回到 86LDmicro 按 **上傳**（F8）。



圖 13-22　86HMI 編輯器：語系按鈕的 Clicked 腳本 Hmi.setLanguage(1);

***注意：**腳本留空時，模擬視窗可能看起來正常，但上傳到控制器後按按鈕不會切換語系。ex30 範例已使用正確的腳本。

第 14 章　C/C++ 與 86Duino 整合

86LDmicro 的 PLC 程式最後會變成 86Duino 的 C/C++ 程式（`plc86.cpp`），由 86Duino IDE 的編譯器編譯。因此在同一個專案中可以加入自己的 C/C++：序列通訊協定、網路、特殊演算法、86Duino / Arduino 的函式庫，都能與 LD、ST、SFC 共用同一份變數。

寫在哪裡	適合	本章
使用者程式檔 `usercode/*.cpp86`	與 PLC 變數密切相關的程式：通訊、換算、每個 EtherCAT 週期的同步控制	14.1 ~ 14.4
sketch 的 `.ino` 與其他 `.cpp` / `.h`	與 PLC 無關或已有的 Arduino 程式	14.5

14.1　使用者程式檔

「**專案 > 新增使用者程式檔（C/C++）...**」（或專案樹「使用者程式（C/C++）」右鍵）輸入名稱（預設 `MyCode`），建立 `usercode/<名稱>.cpp86`。建置時它直接編進 `plc86.cpp`，可以使用 86Duino / Arduino 的所有 API 與函式庫（`#include <...>` 照常寫）。

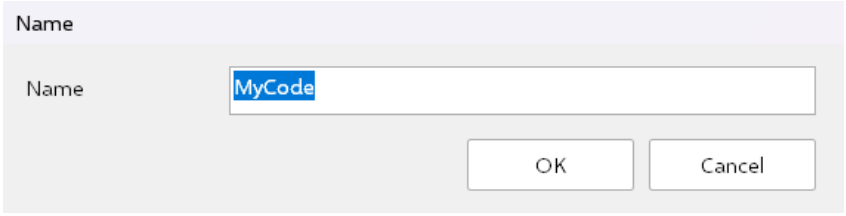


圖 14-1　新增使用者程式檔：輸入名稱

新檔案附有範本：註解說明存取 PLC 變數的寫法，並列出各個掛勾函式（14.3 節）。範本會挑選 GVL 中第一個 `X...` 與 `Y...` 開頭的 BOOL 變數當作範例名稱；專案沒有 EtherCAT 裝置時，`plc86_user_ethercat()` 以註解列出：

```
// MyCode: C/C++ of the project, compiled into plc86.cpp with the PLC (Help > C/C++ user code).
//
// The PLC's variables -- not by their bare names, through these (Ctrl+Space lists them):
//   GVL(Xstart)              a global variable (GVL), read and written: GVL(Ylamp) = true;
//   GVL(table[2]), GVL(rec.x)    an array element (the ST indexes), a structure's field
//   ST(Main.count)               a PROGRAM's variable (ST(Main.t1.Q): a function block's output)
//   FUN(Scale)(x, 10)            an ST FUNCTION of the project
//   LD_READ(x), LD_WRITE(y, v)   as in LDmicro 1.1
// The hooks: the PLC calls the ones defined here (delete the ones not needed).
// The USB Serial is the upload and online port: print to Serial1..3, not to Serial.

// in setup(), after the PLC's: begin libraries (Serial1, Ethernet, CAN, Modbus ...)
void plc86_user_setup(void) {
}

// every scan of the first task, before its programs (the inputs are read)
void plc86_user_before(void) {
}
```

```
// every scan of the first task, after its programs (before the outputs are written)
void plc86_user_after(void) {
    // e.g.: GVL(Ylamp) = GVL(Xstart);
}

// loop(), every pass: non-blocking work only (no delay())
void plc86_user_loop(void) {
}

// every EtherCAT cycle (the master's cyclic callback, an interrupt): synchronous control.
// Only with an EtherCAT device (Project > Devices). Short: no delay(), no Serial, no malloc.
// void plc86_user_ethercat(void) {
//     if(!ECAT_OK)
//         return;
// }
```

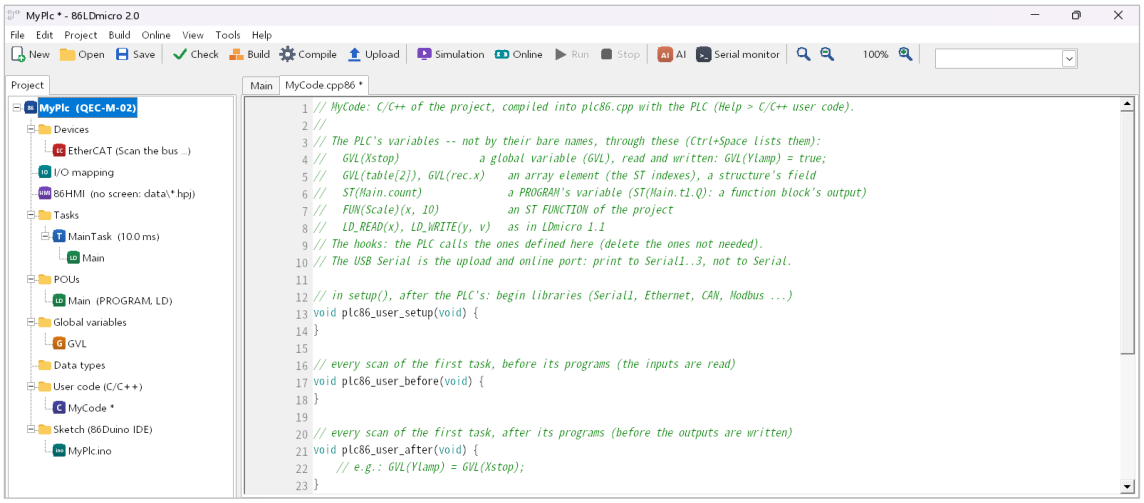


圖 14-2　新增的使用者程式 MyCode.cpp86：範本與專案樹「使用者程式（C/C++）」

使用者程式編輯器與 ST 編輯器相同，有語法顏色、Ctrl+T 自動排版、錯誤底線；另提供 86Duino / Arduino 全部 API 的補全與參數提示（約 7,600 項），選到尚未 include 的函式庫物件或類別時會自動在檔頭加上 `#include`。

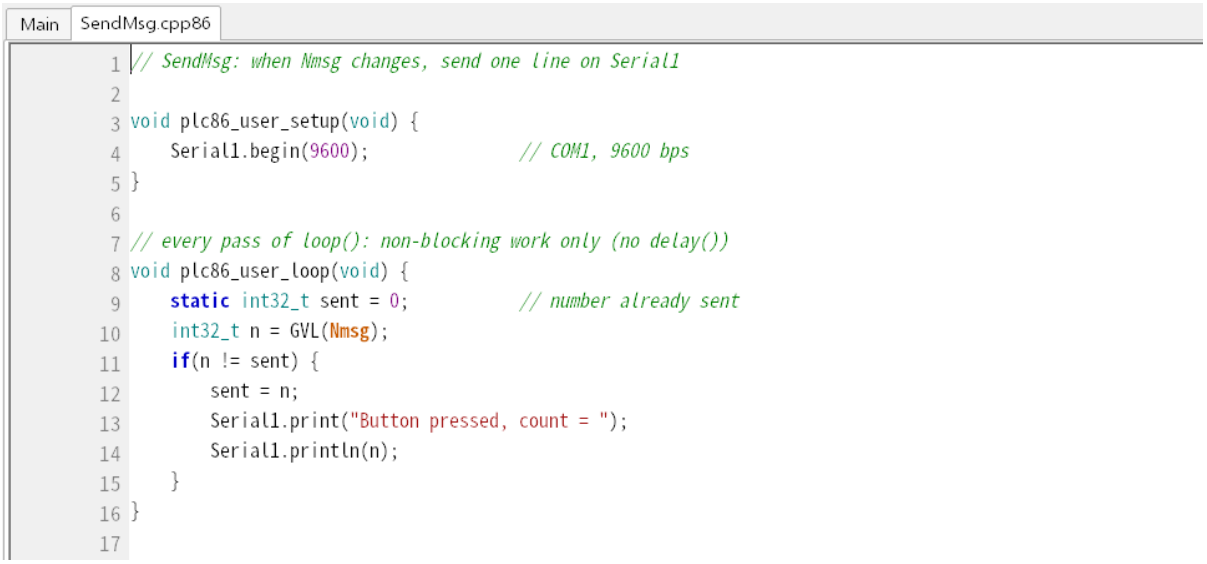


圖 14-3　使用者程式編輯器：掛勾函式與 GVL() 存取 PLC 變數

14.2 存取 PLC 變數

*__重要__：PLC 變數__不能直接寫名稱__（C 裡實際的名稱是 `ldst_P_GVL__Xstart` 之類），要寫 `GVL(Xstart)`。直接寫名稱時，編譯錯誤會提示正確寫法。

寫法	意義	範例
`GVL(名稱)`	全域變數（讀寫皆可）	`if(GVL(Xstart)) GVL(Ylamp) = true;`
`GVL(陣列[i])`	全域陣列元素（索引用 ST 的範圍）	`ARRAY[1..5]` 的第 3 個：`GVL(Recipe[3]) = 100;`
`GVL(結構.欄位)`	全域結構欄位	`GVL(Axis1.pos) = 12.5f;`
`ST(程式.變數)`	PROGRAM 的區域變數、FB 實例的成員、陣列、結構	`ST(Main.tLamp.Q)`、`ST(Filling.step)`
`FUN(名稱)(參數)`	呼叫專案的 ST FUNCTION	`float v = FUN(Scale)(raw, 0.0f, 100.0f);`
`LD_READ(x)`、`LD_WRITE(y, v)`、`LD(x)`	LDmicro 1.1 寫法，同 `GVL()`	`LD_WRITE(Ylamp, LD_READ(Xstart));`
`ECAT_MASTER`、`ECAT_OK`、`ECAT_SLAVE(從站)`	EtherCAT 主站物件、主站已啟動、從站的驅動物件（專案有 EtherCAT 裝置時）	`ECAT_SLAVE(S0__QEC_R11CFFG).digitalWrite(1, HIGH);`
`AXIS(軸名)`、`mc_*`、`M86_MACHINE(機台)`	運動控制（第 12 章）	`mc_move_abs(AXIS(AxisX), 100.0f, 50.0f);`

名稱、欄位、索引在建置時檢查，錯誤指回 `.cpp86` 的行號（例如 `usercode/MyCode.cpp86:11: error: GVL(Xstpo): there is no such global variable`）。在「訊息」雙擊該列即跳到那一行。

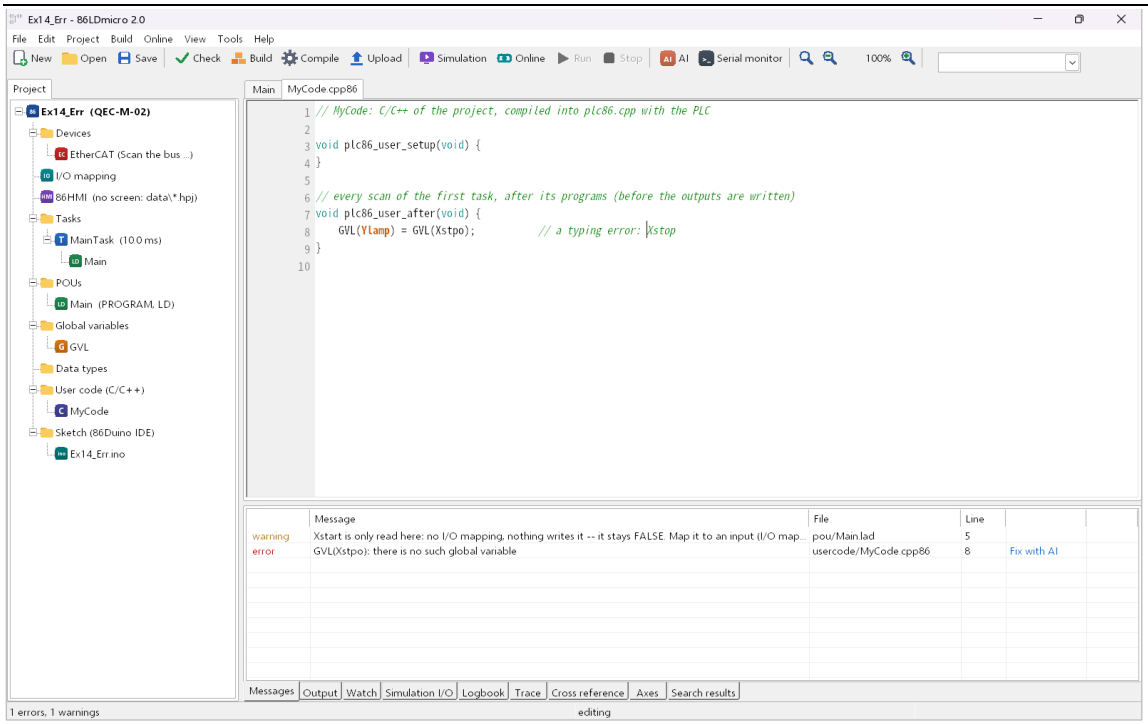


圖 14-4　打錯的變數名稱：「訊息」指出 .cpp86 的行號

輸入 GVL(、ST(、FUN(時會列出可用的名稱（型別與註解），(之後顯示 FUNCTION 的參數；滑鼠停在名稱上可看到型別，模擬或線上時也顯示數值。

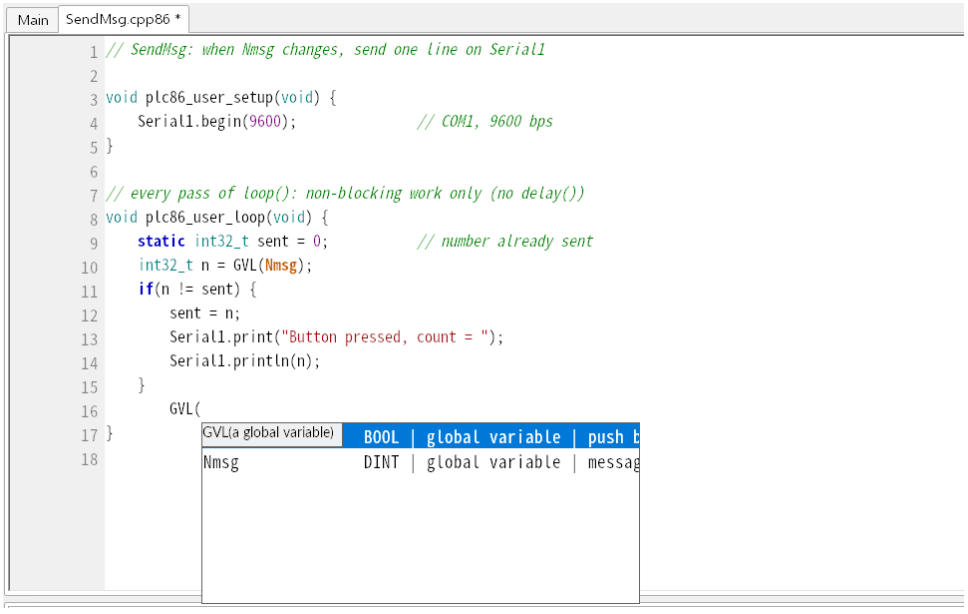


圖 14-5　輸入 GVL(列出全域變數（型別與註解）

14.2.1 型別對應

PLC 型別	C 型別	PLC 型別	C 型別
BOOL	`bool`	REAL	`float`
SINT / USINT	`int8_t` / `uint8_t`	LREAL	`double`
INT / UINT	`int16_t` / `uint16_t`	TIME	`int32_t`（毫秒）
DINT / UDINT	`int32_t` / `uint32_t`	列舉	`int32_t`
LINT / ULINT	`int64_t` / `uint64_t`	STRING[n]	`char[n+1]`（`STRING` 為 `char[81]`）
BYTE / WORD	`uint8_t` / `uint16_t`	DWORD / LWORD	`uint32_t` / `uint64_t`

*提示：STRING 在 C 中是以 0 結尾的字元陣列：讀取直接當 const char * 使用；寫入請用 strncpy(GVL(Msg), "text", sizeof(GVL(Msg)) - 1); GVL(Msg)[sizeof(GVL(Msg)) - 1] = '\0';，不超過宣告的長度並確保以 0 結尾。

14.3 掛勾函式

掛勾函式（hook）**有定義就會被呼叫**，沒用到的可以刪除。它們讓你的 C/C++ 在 PLC 執行的固定時點介入：

函式	呼叫時機	適合
`plc86_user_setup()`	`setup()` 中、PLC 初始化之後	`Serial1.begin()`、Ethernet、CAN、Modbus、SCoop 等初始化
`plc86_user_before()`	第一個工作每次掃描：讀入輸入之後、程式執行之前	把外部資料交給 PLC（前處理輸入）
`plc86_user_after()`	第一個工作每次掃描：程式執行之後、寫出輸出之前	讀 PLC 的結果（後處理輸出、回報狀態）
`plc86_user_loop()`	`loop()` 每一圈（掃描之間）	非阻塞通訊；不要 `delay()`
`plc86_user_ethercat()`	每個 EtherCAT 週期（主站的週期 callback，中斷中）	同步控制；保持簡短
`plc86_user_ecat_config()`	EtherCAT 從站 attach 之後、start 之前	SDO 初始參數、主站設定

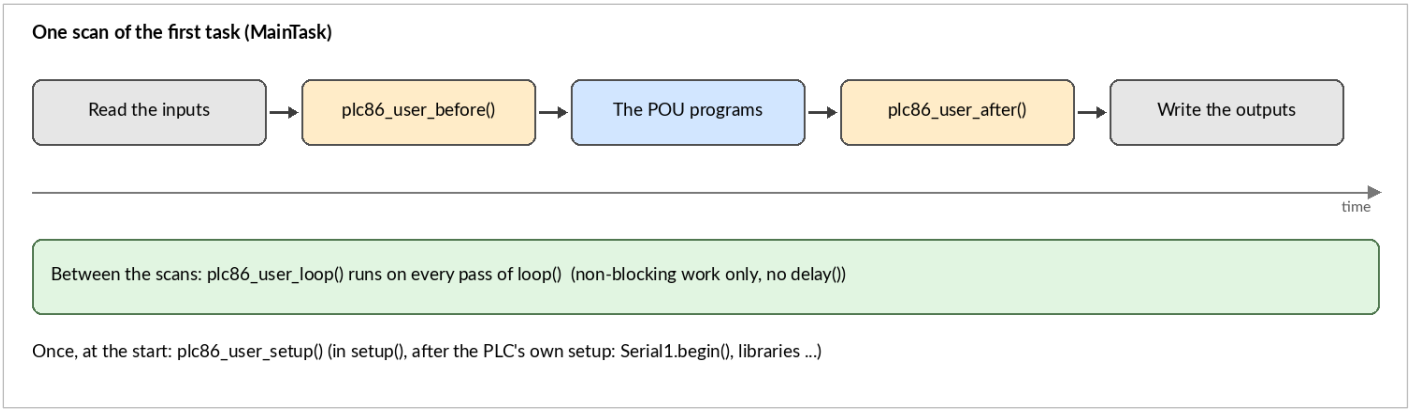


圖 14-6　一個掃描中掛勾函式的位置：讀入輸入 → before → 程式 → after → 寫出輸出；loop 在掃描之間

- `plc86_user_ethercat()`：專案有 EtherCAT 裝置即可使用，不需要建立 EtherCAT 工作；有 EtherCAT 工作時，在該工作的程式之後呼叫。在 `plc86_setup()` 完成前不會被呼叫。
- `plc86_user_ecat_config()` 不在範本中，需要時自行加入（寫在使用者程式或 `.ino` 都可以）。
- 補全清單中可以直接插入這些函式的空白定義。

14.4 範例

14.4.1 Serial1：收到指令就啟動，狀態改變時回報

情境：外部設備經控制器的 COM1（RS-232 / 485）送出字元 S（啟動）、T（停止）；PLC 狀態改變時回傳 RUN / STOP。實體按鈕仍可操作。（以 Serial1 傳送訊息的入門範例見 6.7 節。）

變數	型別	I/O 對應	用途
Xstart	BOOL	LocalIO.X00	實體啟動按鈕
Xstop	BOOL	LocalIO.X01	實體停止按鈕
Rstart	BOOL	（無）	Serial1 的啟動指令（一個掃描週期）
Rstop	BOOL	（無）	Serial1 的停止指令（一個掃描週期）
Rrun	BOOL	（無）	運轉中（自保持）
Yrun	BOOL	LocalIO.Y00	運轉指示燈

階梯圖（Main）：

```
NETWORK 'start/stop'
    [ NO(Xstart) | NO(Rstart) | NO(Rrun) ] NC(Xstop) NC(Rstop) OUT(Rrun)
NETWORK 'lamp'
    NO(Rrun) OUT(Yrun)
```

使用者程式（MyCode.cpp86）：

```
static bool cmdStart = false, cmdStop = false;

void plc86_user_setup(void)
{
    Serial1.begin(9600);                    // 控制器的 COM1
}

void plc86_user_loop(void)                  // loop() 每一圈：不要 delay()
{
    while(Serial1.available()) {            // 只讀已經收到的字元
        char c = Serial1.read();
        if(c == 'S') cmdStart = true;
        if(c == 'T') cmdStop = true;
    }
}

void plc86_user_before(void)                // 程式執行前：指令交給 PLC 一個掃描週期
{
    GVL(Rstart) = cmdStart;
    GVL(Rstop) = cmdStop;
    cmdStart = cmdStop = false;
}

void plc86_user_after(void)                 // 程式執行後：狀態改變就回報
```

```
{
    static bool last = false;
    if(GVL(Rrun) != last) {
        last = GVL(Rrun);
        Serial1.println(last ? "RUN" : "STOP");
    }
}
```

#	操作
1	在 GVL 宣告上表的變數；依 4.6 節把 Xstart、Xstop、Yrun 對應到本機 I/O
2	Main 以文字模式（Ctrl+T）貼上階梯圖，按「回到圖形」
3	「專案 > 新增使用者程式檔（C/C++）...」名稱 MyCode，貼上程式
4	按 **檢查**（Shift+F7），再 **上傳**（F8）
5	COM1 接到電腦的 USB 轉 RS-232 / 485，在「序列埠監看」選該埠、9600，按 **連線**，送出 `S`

預期結果：送出 `S` 後 `Rrun`、`Yrun` 變 TRUE，序列埠監看收到 `RUN`；送出 `T` 或按實體停止，收到 `STOP`。`Rstart` / `Rstop` 只在一個掃描週期為 TRUE，所以不會「卡住」。

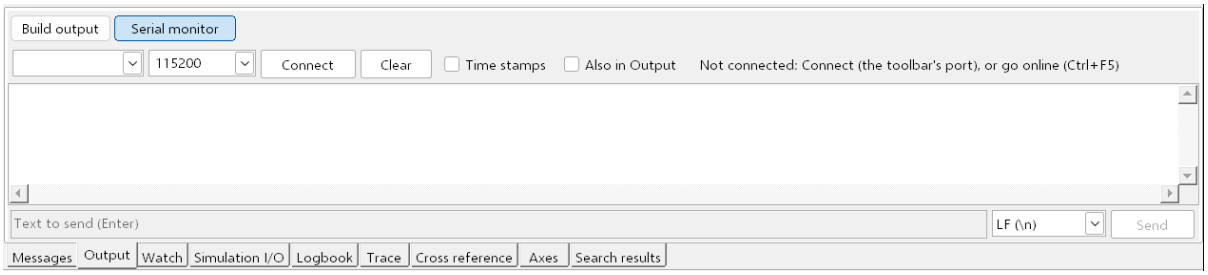


圖 14-7　序列埠監看：選埠、鮑率、連線、時間戳記、傳送

14.4.2 HTTP 狀態頁（構想）

用 86Duino 的 Ethernet 函式庫做一個最簡單的 HTTP 伺服器：瀏覽器開 `http://192.168.1.50/` 就看到 PLC 變數的 JSON。以下示範做法，IP 位址等請依現場網路修改：

```
#include <Ethernet.h>

static byte mac[] = {0x02, 0x86, 0x00, 0x00, 0x00, 0x01};
static IPAddress ip(192, 168, 1, 50);
static EthernetServer web(80);

void plc86_user_setup(void)
{
    Ethernet.begin(mac, ip);
    web.begin();
}

void plc86_user_loop(void)                  // 不等待：沒有連線就立刻返回
{
    EthernetClient client = web.available();
    if(!client)
        return;
    while(client.available())               // 讀掉請求（這裡不解析）
        client.read();
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: application/json");
    client.println("Connection: close");
    client.println();
    client.print("{\"run\":");
    client.print(GVL(Rrun) ? "true" : "false");
    client.print(",\"level\":");
    client.print(GVL(Level), 1);
    client.print(",\"count\":");
    client.print(GVL(Ncount));
    client.println("}");
    client.stop();
}
```

- 通訊寫在 plc86_user_loop()：它在掃描之間執行，不放進 PLC 工作；但它花的時間會延後下一次掃描，每次處理要很短（不要等待連線或資料），耗時請依下一點單獨量測。本例是做法示範：網路壅塞時 print()、client.stop() 也可能花一段時間。
- plc86_user_loop() 的時間不算在工作的掃描時間內，看門狗與狀態列的掃描時間都看不到它造成的延後。請在 plc86_user_loop() 的開頭與結尾用 micros() 量測，把最大值寫到 GVL 變數（例如 GVL(LoopMaxUs)）在監看視窗觀察；它應遠小於工作的週期。
- 回應中讀取的變數（`Rrun`、`Level`、`Ncount`）要在 GVL 宣告。要從網頁**寫入** PLC，建議同 14.4.1 節：先存在 C 的變數，再於 `plc86_user_before()` 交給 PLC。

14.4.3 每個 EtherCAT 週期的同步輸出

```
void plc86_user_ethercat(void)           // 每個 EtherCAT 週期（中斷中）
{
    if(!ECAT_OK) return;
    static uint32_t n = 0;
    n++;
    GVL(Ncycle) = (int32_t)n;                // 給 LD / ST 看
    // DQ01 只由這段程式寫入（不要再把它對應到變數，也不要由其他程式寫入）
    bool on = GVL(Rrun) && ((n / 500) & 1);       // 每 500 個週期反轉一次：週期 1 ms 時每 0.5 秒（一明一暗約 1 秒）
    ECAT_SLAVE(S0__QEC_R11CFFG).digitalWrite(1, on ? HIGH : LOW);   // Rrun 為 FALSE 時寫 LOW
}

void plc86_user_ecat_config(void)         // attach 之後、start 之前
{
    // ECAT_SLAVE(S0__QEC_R11CFFG).sdoDownload8(index, subindex, value);   // 例：SDO 設定；物件、有效值請依模組的物件字典填入
}
```

從站名稱（例如 `S0__QEC_R11CFFG`）與專案樹「裝置 > EtherCAT」下的名稱相同（第 11 章）。DMP 的 QEC 模組會使用其驅動類別（例如 `EthercatDevice_QECR11CFFG`），可呼叫 `digitalWrite`、`digitalRead`、`analogRead` 等；CiA 402 驅動器為 `EthercatDevice_CiA402`，其他從站為 `EthercatDevice_Generic`（`pdoGetInputBuffer()` / `pdoGetOutputBuffer()`、SDO）。

14.5 sketch 的 .ino

sketch 的 `.ino` 只需要三行（建置會自動補上），其餘可以放自己的程式：

```
#include "plc86.h"
void setup() { plc86_setup(); /* 你自己的初始化 */ }
void loop()  { plc86_loop();  /* 你自己的程式，不要 delay() 太久 */ }
```

- EtherCAT 主站、序列埠、86HMI、Modbus、線上監看都由 `plc86_setup()` 依專案的裝置啟動，`.ino` 不需要也不可以再呼叫 `EVA.begin()`、`EthercatMaster::begin()`。使用 86HMI 時，建置會加上 `#include "myhmi.h"` 與 `Hmi.begin();`。
- `.ino` 與 sketch 的 `.cpp` 在 `#include "plc86.h"` 後可用 `GVL(名稱)`、`LD_READ`、`LD_WRITE` 存取 GVL 的基本型別變數（BOOL、整數、REAL、TIME），也可使用 EtherCAT 物件與 `mc_*`；陣列、結構、`ST()`、`FUN()`、`ECAT_SLAVE()`、`M86_MACHINE()` 只在使用者程式（.cpp86）中可用。
- 專案樹「Sketch（86Duino IDE）」或「專案 > Sketch 的 .ino（86Duino IDE 的主程式）」可直接在 86LDmicro 中編輯 `.ino` 與其他 `.cpp` / `.h`。儲存、建置、編譯前會一併存檔；建置修補 `.ino` 後分頁自動更新。

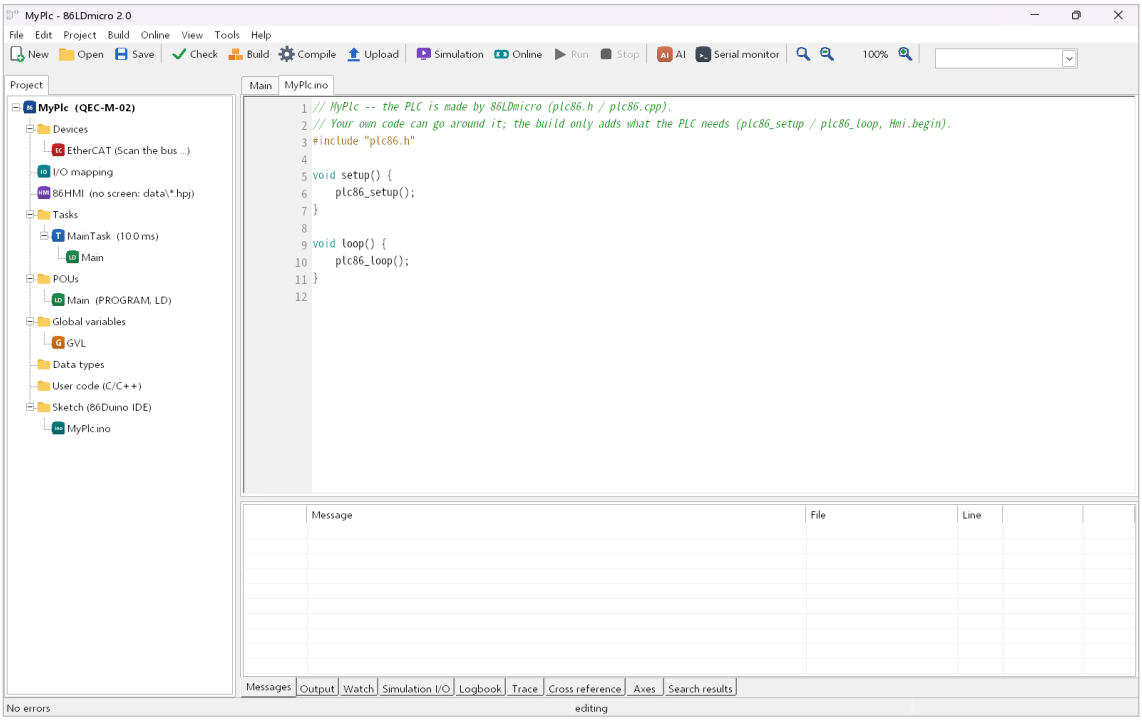


圖 14-8　在 86LDmicro 中編輯 sketch 的 .ino

建置後，sketch 中多出下列檔案（每次建置重寫，**請勿手改**）：

檔案	內容
`plc86.h` / `plc86.cpp`	PLC 程式：變數、POU、工作、I/O 對應、86HMI 綁定、使用者程式
`plc86_rt.h` / `plc86_rt.cpp`	執行環境（每個專案相同）：線上監看（讀寫、強制、RUN / STOP）、RETAIN、事件紀錄
`plc86_ecat.h` / `plc86_ecat.cpp`	EtherCAT 主站與從站物件、啟動流程（`begin` → 各從站 `attach` → `plc86_user_ecat_config()` → `start`）

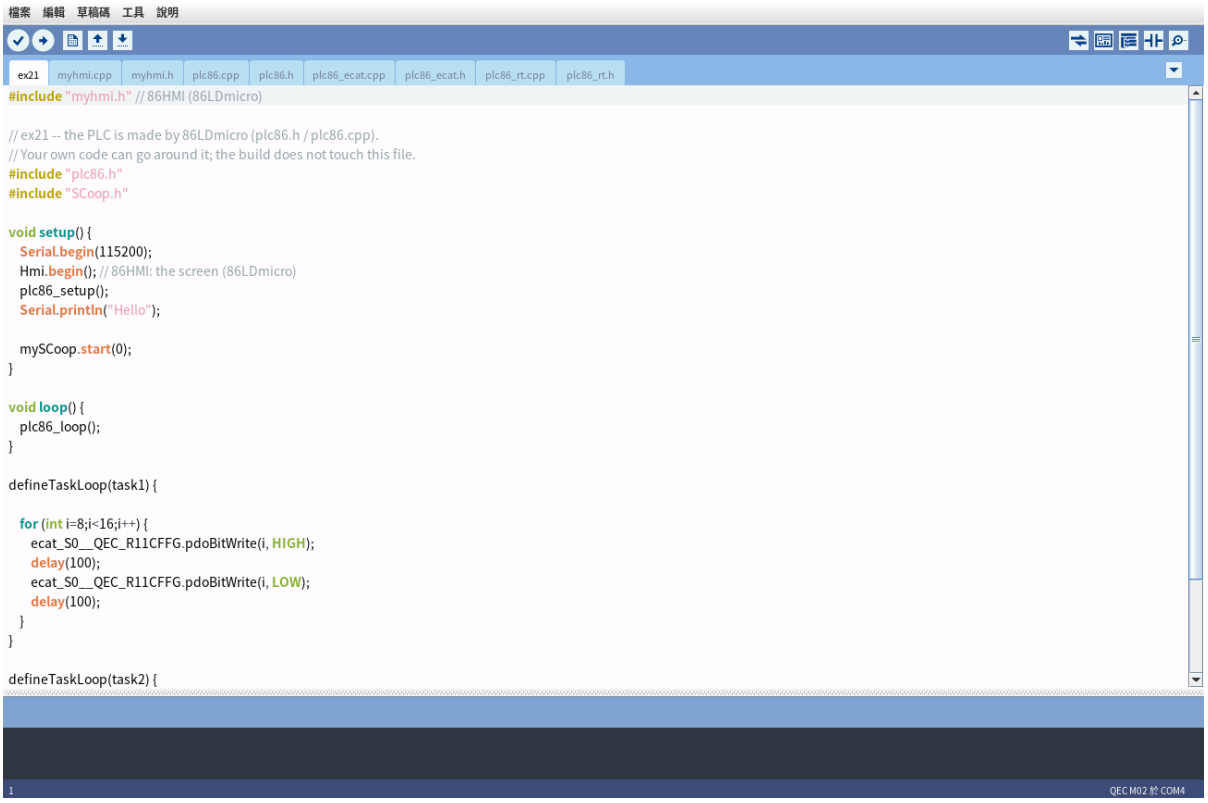


圖 14-9　建置後的 86Duino IDE：plc86.cpp、plc86.h、plc86_ecat.cpp 等分頁

14.6 注意事項

注意	原因與做法
不要對 USB `Serial` 持續列印	USB 的 `Serial` 用於上傳與線上監看。電腦沒有讀取時（86LDmicro 未連線、序列埠監看關閉），`Serial.print()` 寫滿緩衝區後會等待，**整個程式（包括 PLC 掃描）跟著停住**。與外部設備通訊請用 `Serial1`～`Serial3`；除錯列印請先判斷 `Serial.availableForWrite()` 足夠再印
`plc86_user_ethercat()` 在中斷中執行	不可 `delay()`、`Serial.print()`、動態配置記憶體（malloc / new）；先確認 `ECAT_OK`；浮點運算可用。處理時間要遠小於 EtherCAT 週期
`plc86_user_loop()` 不要等待	不要 `delay()`、不要 `while` 等外部回應；一次只處理已收到的資料
不要手改 `plc86*` 檔案	每次建置都會重寫；要調整 EtherCAT 啟動流程請用 `plc86_user_ecat_config()`
已對應實體輸入的變數	C 寫入的值會在下一個掃描讀入輸入時被覆蓋；請寫內部變數（例如 `Rstart`），再在程式中與實體輸入並聯
C++ 標準標頭	`#include <vector>` 等會自動放到 Arduino.h 之前（避開 min / max 巨集衝突）
Serial 的初始化	啟用線上監看時執行環境已開啟 `Serial`；未啟用時請在 `.ino` 的 `setup()` 或 `plc86_user_setup()` 自行 `Serial.begin(115200);`

*注意：1.1 的 `.ino` 若還有 `#include "myeva.h"`、`EVA.begin()`，使用 2.0 的 EtherCAT 時請移除，並移走 sketch 中的 `myeva.cpp` / `myeva.h`（建置會自動註解 `EVA.begin()` 並提醒，見 15.1 節）。

第 15 章　匯入、AI、命令列與專案保護

本章說明幾個輔助功能：把 LDmicro 1.1 或其他 IEC 61131-3 工具的程式匯入 86LDmicro、使用 AI 助理與 MCP、以命令列工具做批次處理，以及用密碼保護專案。

15.1　匯入 LDmicro 1.1 專案

LDmicro 1.1 的程式（`.ld`，通常是 sketch 的 `data\myLD.ld`）可以轉成 86LDmicro 2.0 的專案。專案寫到同一個 sketch 的 `data\86ld`，**原本的 `.ld` 不會被修改**，需要時仍可用 1.1 開啟。

15.1.1　從 86Duino IDE 匯入

1. 在 86Duino IDE 開啟含有 `data\myLD.ld` 的 sketch，按「**工具 > 86LDmicro**」。
2. 86LDmicro 偵測到 1.1 的程式，詢問是否匯入，按 **是**。
3. 匯入完成後開啟專案：1.1 的階梯圖成為 POU，變數放進 GVL，I/O 腳位成為 I/O 對應。

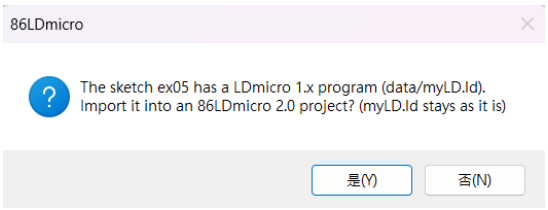


圖 15-1　偵測到 1.1 專案時的詢問

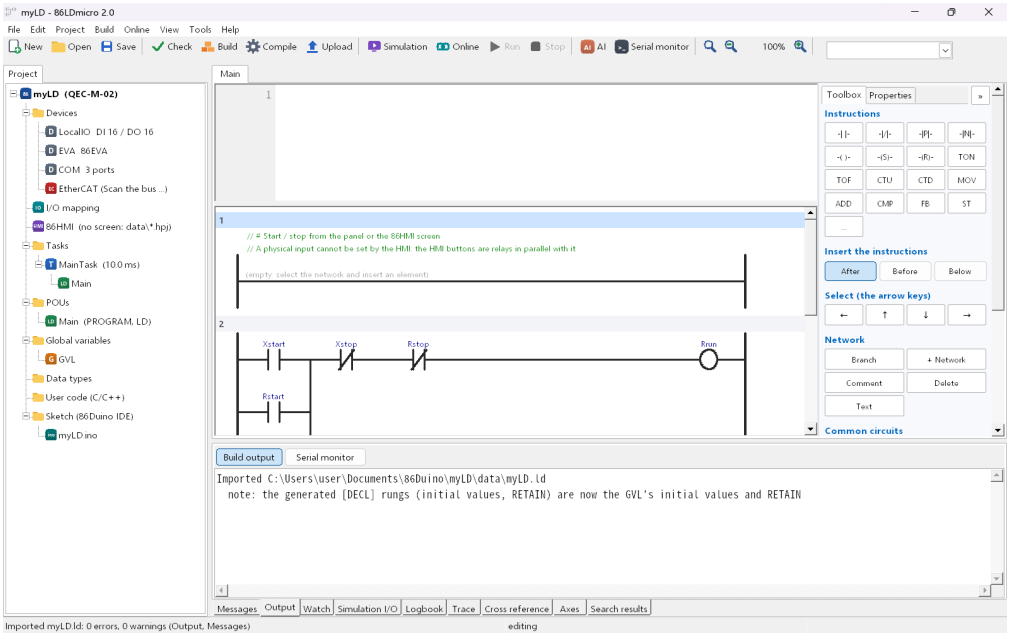


圖 15-2　匯入後的專案：1.1 的階梯圖與 ST 變成 POU，依原順序放進 MainTask

15.1.2 從 86LDmicro 匯入

1. 「**檔案 > 匯入 LDmicro 1.x 程式或 PLCopen XML (.ld, .xml)...**」選 `.ld` 檔；或把 `.ld` 檔直接拖放到 86LDmicro 視窗（會先詢問「要將 … 匯入為新的 86LDmicro 專案嗎？」）。
2. 「匯入」表單的「**sketch 資料夾**」：`.ld` 在 sketch 的 `data` 資料夾中時，預設為該 sketch；否則為 `.ld` 旁、與檔名同名的資料夾。專案寫入 `<sketch>\data\86ld`。按 **確定**。
3. 狀態列顯示「已匯入 …：N 個錯誤、M 個警告（見「輸出」、「訊息」）」，並開啟第一個工作的第一個程式。
4. `.ino` 已存在時不覆寫；其中 1.1 的 `setupPlc()` / `loopPlc()` 請改成 `plc86_setup()` / `plc86_loop()`（14.5 節），其他程式碼不變。
5. 按 **檢查**（Shift+F7）確認沒有錯誤，再 **模擬**（F5）驗證行為，最後上傳。

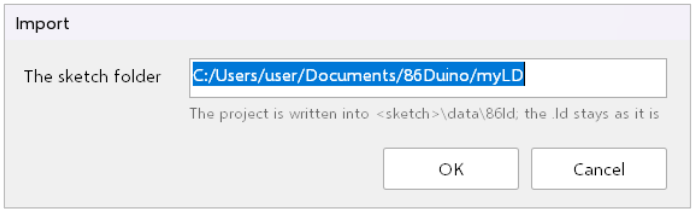


圖 15-3　「匯入」表單：sketch 資料夾（專案寫入 <sketch>\data\86ld；原 .ld 檔不變）

15.1.3 什麼變成什麼

LDmicro 1.1	86LDmicro 2.0
階梯圖（一個程式）	`pou/Main.lad`（若 ST 程式已叫 Main，則為 `Ladder`）
ST 程式（POU、before / after、task=ecat）	`pou/*.st`；階梯圖前後的順序成為工作（Task）中的順序，EtherCAT 週期成為 EtherCAT 工作
函式庫 FB、ST block	`pou/*.st` 的 FUNCTION_BLOCK / FUNCTION
資料型別	`dut/*.dut`
VAR LIST、DECLARATIONS（初值、RETAIN、反相、濾波、換算）	`gvl/GVL.gvl`（型別依 1.1 的大小規則）＋ I/O 對應的反相、濾波、比例
IO LIST（腳位、Modbus）	I/O 對應（LocalIO、Pin、Modbus）
BINDINGS（EVA 點、伺服、HMI）	I/O 對應（EVA.*）與綁定（伺服、86HMI）
使用者程式碼（global / setup / beforescan / afterscan / loop / ethercat）	`usercode/*.cpp86` 的掛勾函式 `plc86_user_setup`、`_before`、`_after`、`_loop`、`_ethercat`；`LD_READ` / `LD_WRITE` 照常可用（14.2 節）
86EVA、COM 對應、Modbus、Ethernet、Modbus server	裝置
線上監看	86LDmicro 的「線上」（Ctrl+F5，10.4 節）

15.1.4 注意事項

- 1.x 的計時器、計數器等指令保留 1.x 語意（以掃描週期計數），行為與 1.1 相同；新程式建議改用 IEC 功能塊（第 7.8 節）。
- 與 IEC 保留字同名的變數會加上 `_v`（例如 `TON` → `TON_v`）；`__` 開頭的名稱保留給 2.0 內部使用。
- 改用 2.0 的 EtherCAT（第 11 章）時，移除 `.ino` 中 1.x 的 `#include "myeva.h"` 與 `EVA.begin()`，以及 sketch 中的 `myeva.cpp` / `myeva.h`（建置會自動註解 `EVA.begin()` 並提醒）。
- `MC_ReadActualPosition.Position` 在 2.0 為 REAL，原本接 DINT 的程式請改為 REAL 或使用 `REAL_TO_DINT`。
- 編譯錯誤不再自動跳出 AI 視窗；需要時在「訊息」按「用 AI 修正」（15.3 節）。

15.2 匯入 PLCopen XML

PLCopen XML（TC6）是 IEC 61131-3 的標準交換格式。OpenPLC Editor、Beremiz、CODESYS 等工具匯出的 `.xml` 可以匯入成新的 86LDmicro 專案，操作與 15.1.2 節相同。

1. 在原工具中匯出 PLCopen XML（例如 CODESYS：「專案 > 匯出 PLCopenXML」；OpenPLC Editor / Beremiz 的專案檔 `plc.xml` 即為此格式）。
2. 86LDmicro「**檔案 > 匯入 LDmicro 1.x 程式或 PLCopen XML (.ld, .xml)...**」選 `.xml`；或把 `.xml` 直接**拖放**到 86LDmicro 視窗，在詢問時按 **是**。
3. 在「匯入」表單選擇 sketch 資料夾，按 **確定**。
4. 看狀態列的錯誤、警告數；「輸出」列出匯入的 POU 數（LD、ST、SFC 各幾個）與未匯入的部分。
5. 按 **檢查**、**模擬** 確認，再對應 I/O（4.6 節）並上傳。

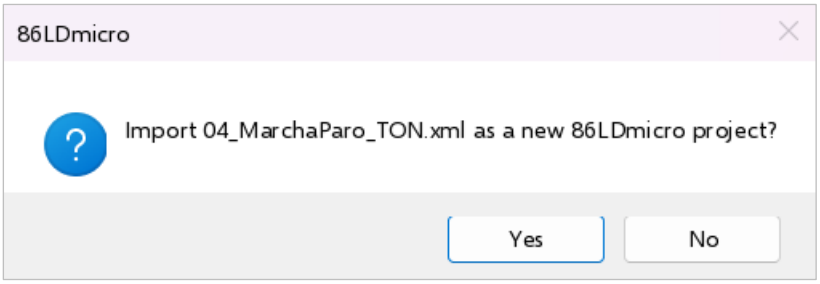


圖 15-4　把 .xml 拖放到視窗：詢問是否匯入為新的 86LDmicro 專案

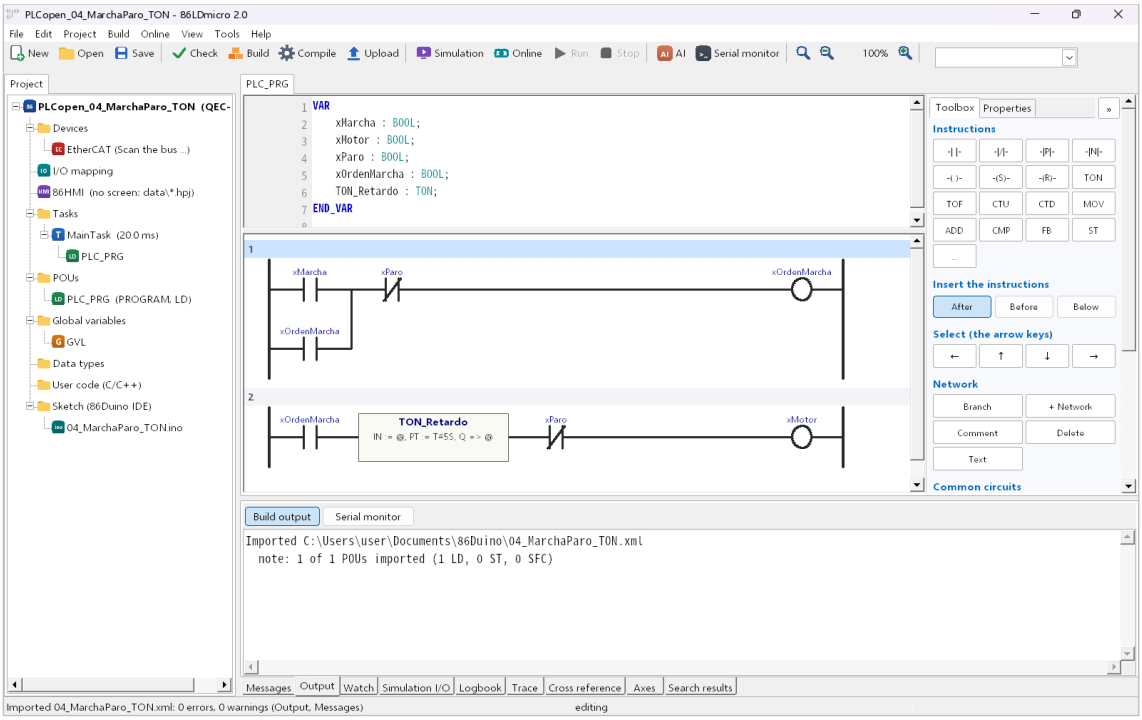


圖 15-5　PLCopen XML 匯入後：POU、GVL 與「輸出」中的匯入摘要

PLCopen 內容	**匯入結果**
LD（階梯圖）	`.lad` 程式（接點、線圈、功能塊、分支）
ST	`.st` 程式
SFC	`.sfc` 程式（步驟、轉移、動作、分支）
資料型別（STRUCT、列舉、陣列）	`.dut`
全域變數（configuration / resource）	`.gvl`；`AT %IX…` 位址只保留為註解，請在 I/O 對應中重新連結
工作（task）與程式實例	專案的工作與指派
PROGRAM 的 VAR_INPUT / VAR_OUTPUT	改為 VAR（86LDmicro 的 PROGRAM 沒有參數），訊息中說明
VAR_EXTERNAL	省略（每個 POU 都看得到全域變數），訊息中說明
FBD、IL	**本版不匯入**，於訊息中列出；請在原工具中改寫為 ST 或 LD 後再匯出

命令列：`86ldmicro-cli import <檔案.xml> <sketch 資料夾> [名稱]`；MCP 的 `ld86_import_11` 也接受 PLCopen XML。

15.3 AI 助理

AI 助理可以撰寫或修正 ST / LD / C 程式、解釋錯誤、規劃 I/O。它使用**你自己的 API 金鑰**（或本機的模型），請求直接從這台電腦送到你設定的伺服器；按「傳送」（或「測試連線」）之前不會送出任何資料。

15.3.1 AI 助理分頁

按工具列 **AI**，或「**工具 > AI 助手（MCP）...**」，開啟「AI 助理」分頁：

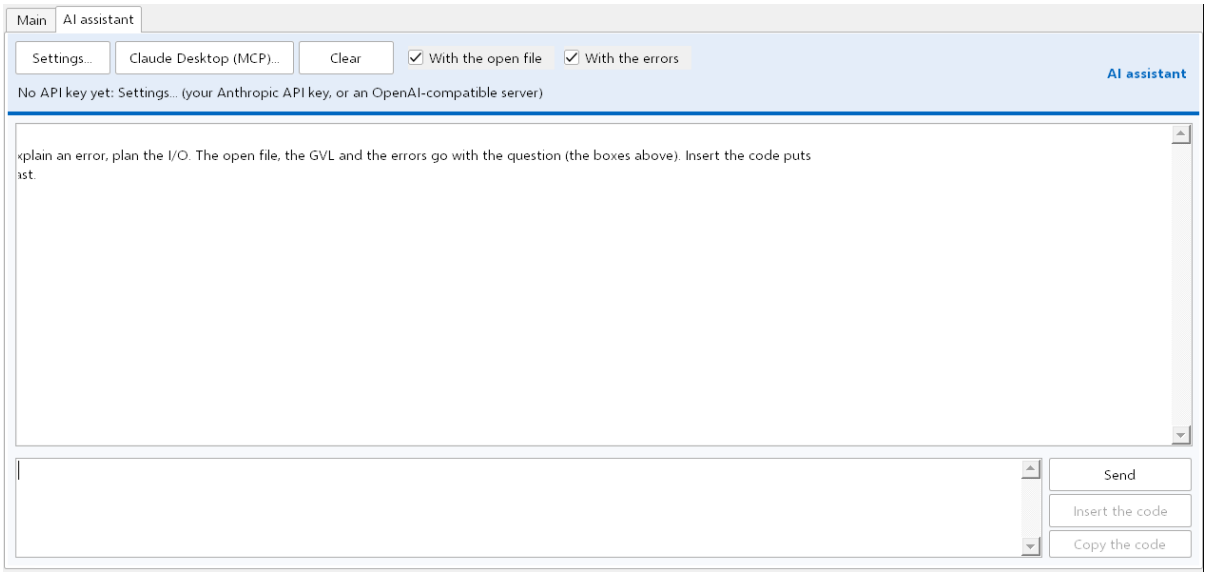


圖 15-6　AI 助理分頁：設定、Claude Desktop（MCP）、清除、附上開啟中的檔案與錯誤

項目	用途
設定...	選服務、輸入 API 金鑰與型號（15.3.2 節）
Claude Desktop（MCP）...	設定 MCP（15.4 節）
清除	清除對話
附上開啟中的檔案、附上錯誤（勾選框）	勾選時，問題連同目前開啟的檔案、「訊息」的錯誤一起送出（GVL 與工作一律附上）
傳送（Ctrl+Enter）	送出下方輸入的問題
插入程式碼	把最後一次回答中的程式碼放進編輯器：ST、C 插在最後使用的編輯器游標處；LD 的 NETWORK 插在選取的網路之後（VAR 區塊中的新變數自動宣告）
複製程式碼	把回答中的程式碼複製到剪貼簿

15.3.2 設定：服務與金鑰

按 **設定...** 開啟「AI 助理：設定」：

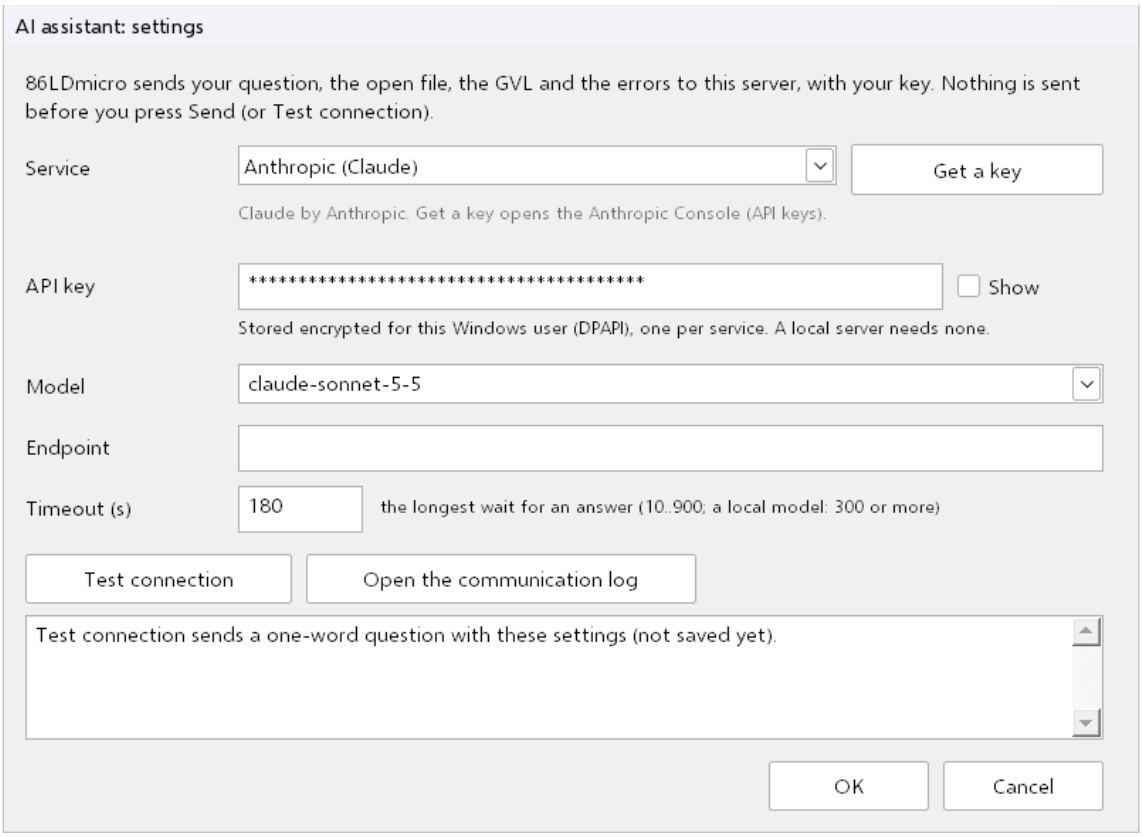


圖 15-7 「AI 助理：設定」：服務、取得金鑰、API 金鑰、型號、端點、逾時、測試連線、開啟通訊紀錄

欄位	說明
服務	Anthropic（Claude）、OpenAI、Google Gemini、Ollama（本機）、LM Studio（本機）、其他 OpenAI 相容伺服器。選擇後端點與型號自動填入
取得金鑰（本機服務為 **下載**）	以瀏覽器開啟該服務的金鑰頁面（Anthropic Console、OpenAI、Google AI Studio）或本機軟體的下載頁
API 金鑰	以此 Windows 使用者加密儲存（DPAPI），每個服務各一組；本機伺服器不需要。勾「顯示」可看到內容
型號	可從清單選，或輸入伺服器列出的模型名稱
端點 (Endpoint)	空白：使用服務的預設；Azure、公司閘道、vLLM 等請填伺服器網址
逾時（秒）	等待回應的最長時間（10..900，預設 180；本機模型建議 300 以上）

***提示：**金鑰、模型與端點只能使用 ASCII 字元；出現相關錯誤時，檢查是否貼上了全形字元。

15.3.3 測試連線與通訊紀錄

1. 在設定視窗按 **測試連線**：以目前設定（尚未儲存）送出一個簡短問題，在背景執行；測試中按鈕變成 **STOP**，可按它停止。
2. 成功時顯示 `OK: <型號> answered in N ms (...)`；失敗時以英文說明原因（見下表）。
3. 按 **開啟通訊紀錄** 查看每次請求的網址、狀態、耗時與回應開頭（不含金鑰）。紀錄檔在 `%LOCALAPPDATA%\86LDmicro\ai_comm.log`。
4. 確認無誤後按 **確定** 儲存。

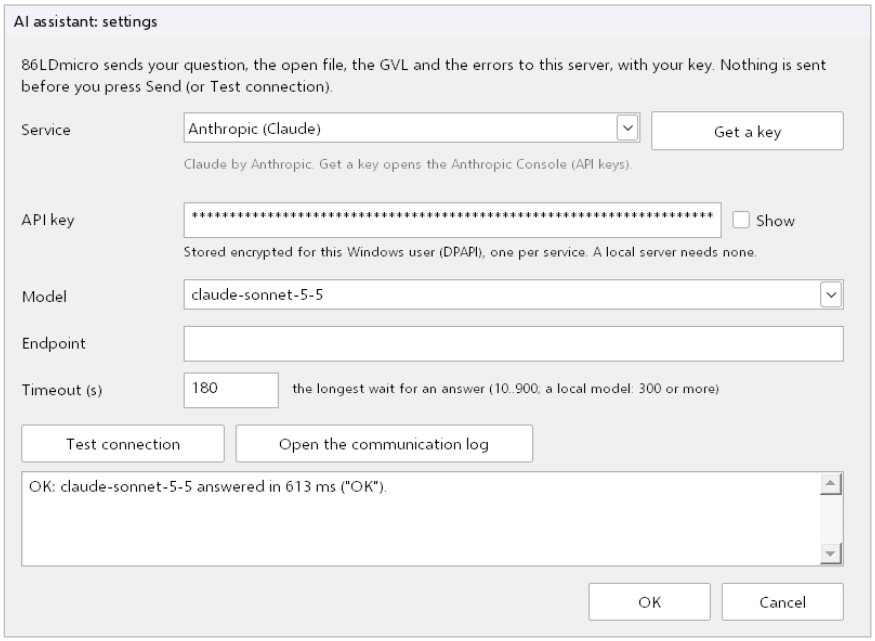


圖 15-8　測試連線的結果：OK: … answered in N ms

測試連線的訊息	意義與處理
401 Unauthorized	API 金鑰錯誤、過期、被撤銷或屬於其他服務：重新貼上金鑰（取得金鑰）
403 Forbidden	金鑰有效，但不能使用此模型或端點（方案、組織、地區、專案停用）
404 Not Found	端點網址或模型名稱錯誤；本機伺服器：模型是否已載入 / 下載（ollama pull）
408 Request Timeout	伺服器放棄等待：再試一次
429 Too Many Requests	超過速率限制或額度用完：稍候再試，或檢查帳號計費
5xx Server error	服務端的問題：稍後再試
Timeout	在設定的秒數內沒有回應：伺服器忙或本機模型仍在載入，加大逾時或再試一次
Network: …	無法連線：沒有網路、DNS、端點打錯、伺服器未啟動（Ollama / LM Studio）、防火牆、Proxy、http / https 錯誤、TLS 連線失敗（Proxy 檢查 HTTPS、電腦日期時間）

15.3.4　提問與插入程式碼

1. 開啟要討論的 POU（例如 Main），在 AI 分頁下方輸入問題，例如「把 Rrun 改成按 Xstart 兩秒後才啟動」，按 **傳送**（Ctrl+Enter）。
2. 回答顯示在上方。內容含程式碼時，按 **插入程式碼** 放進編輯器，或 **複製程式碼**。
3. 插入後狀態列提示「AI 的程式碼已放入編輯器：請檢查（檢查，Shift+F7）」：務必按 **檢查** 並 **模擬** 確認。

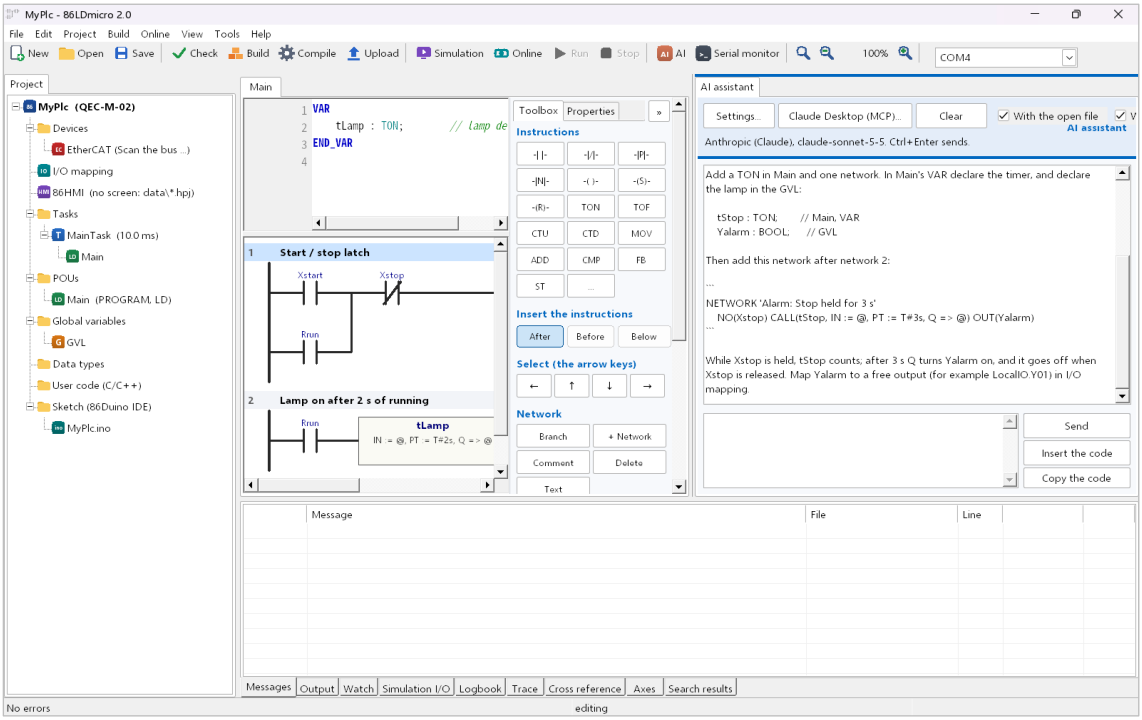


圖 15-9　AI 的回答與「插入程式碼」

15.3.5　用 AI 修正錯誤

「訊息」視窗每個錯誤列的最右邊有 **用 AI 修正**：已設定 AI 時，把該錯誤與附近的程式碼直接在 AI 分頁提問；尚未設定時，把提問內容複製到剪貼簿，可貼到其他 AI 工具。

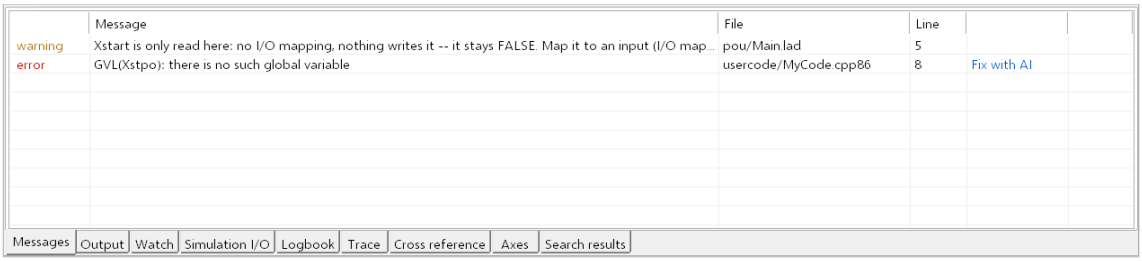


圖 15-10　「訊息」視窗的「用 AI 修正」

> ***注意**：AI 的回答可能有錯。插入的程式碼請自己看懂，並經過檢查、模擬，再上傳到實機。

15.4 MCP

MCP（Model Context Protocol）server 讓 Claude 桌面版、Cowork、Claude Code 等支援 MCP 的 AI 工具，直接讀取、修改、檢查、模擬 86LDmicro 專案。MCP server 安裝在 86LDmicro 的 `mcp` 資料夾（`mcp\ld86_mcp.py`），需要 Python 3。

15.4.1 設定 Claude 桌面版

1. 安裝 Python 3（python.org，安裝時勾選「Add to PATH」）。
2. 在 AI 助理分頁按 **Claude Desktop（MCP）...**，開啟「AI 助手（MCP）」視窗。視窗上方顯示 MCP server、Python 與目前專案的狀態。
3. 按 **設定 Claude 桌面版（MCP）**：把 86ldmicro 加入 `claude_desktop_config.json`（原檔備份為 `.bak`）。
4. 重新啟動 Claude 桌面版，然後直接問它，例如：「開啟 <資料夾> 的 86LDmicro 專案，說明 MainTask」。
5. 其他 MCP 用戶端（Cowork、Claude Code…）：按 **複製 MCP 設定（JSON）**，貼到該工具的 MCP 設定中。

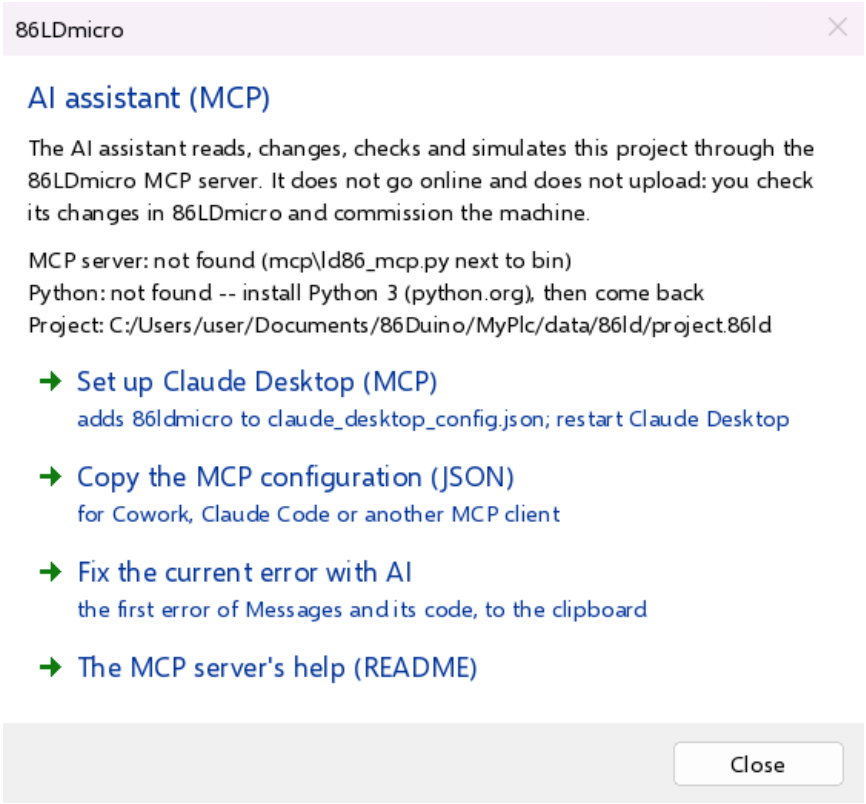


圖 15-11 「AI 助手（MCP）」：設定 Claude 桌面版、複製 MCP 設定、用 AI 修正目前的錯誤、MCP server 說明

視窗中的項目	用途
設定 Claude 桌面版（MCP）	把 86ldmicro 加入 claude_desktop_config.json；之後重新啟動 Claude 桌面版
複製 MCP 設定（JSON）	給 Cowork、Claude Code 或其他 MCP 用戶端
用 AI 修正目前的錯誤	把「訊息」的第一個錯誤與程式碼複製到剪貼簿
MCP server 說明（README）	開啟 `mcp\README.md`

15.4.2 MCP 工具

MCP 工具	用途
ld86_describe	專案概要：工作、POU、GVL、資料型別、使用者程式、裝置、I/O 對應
ld86_read_file / ld86_write_file	讀取 / 新增或取代 POU、GVL、資料型別、使用者程式（寫入後自動編譯並回傳訊息）
ld86_set_project	設定工作、裝置、I/O 對應、綁定等，接著檢查
ld86_check / ld86_simulate	編譯列出錯誤；模擬 N 個掃描並列出變數
ld86_build	寫出 sketch 的產生檔（不編譯、不上傳）
ld86_new_project / ld86_import_11 / ld86_ethercat_eni	新專案、匯入 1.x 或 PLCopen XML、由 ENI 建立 EtherCAT 裝置
ld86_lower / ld86_instructions	LD 轉成的 ST、指令集說明

15.4.3 安全

- MCP server **不會連到控制器、也不會上傳**：AI 改過的專案，請在 86LDmicro 中確認、模擬、試車後再自己上傳。
- AI 透過 MCP 寫入檔案時，86LDmicro 若開著同一個專案，請按「**檢視 > 重新整理：從磁碟重新讀取專案**」（Ctrl+R）載入修改；有未儲存的修改時會先詢問。
- 建議使用版本控制或先備份 sketch 的 `data\86ld`，方便比對與還原 AI 的修改。

15.5 命令列工具

`tool\bin\86ldmicro-cli.exe` 提供與 86LDmicro 相同的檢查、建置、模擬、上傳與線上功能，可用於批次檔或自動化測試。不帶參數執行即列出所有指令：

```
> 86ldmicro-cli
86ldmicro-cli 2.0
  import <file.ld | file.xml> <sketch folder> [name]
  check  <project>
  search <project> <text> [--case] [--word]
  build  <project>
  lower  <project> <POU>
  new    <sketch folder> [name]
  compile <project> [--ide <86Duino IDE folder>]
  upload <project> <port> [--ide <dir>]
  upload-res <project> <port> [--ide <dir>]
  iotable export|import <project> <file.xlsx|.csv>
  format <file>
  ports
  instructions
  online <project> <port> status | read <path>... | write <path> <value> | ...
  eni    <project> <eni.xml> [name on the controller (C:/eni/)]
  sim    <project> <scans> [path...] [--set path=value]... [--diff-inputs]
```

指令	用途
`new <sketch 資料夾> [名稱]`	建立新專案（MainTask 10 ms、階梯圖 Main、GVL）
`import <myLD.ld | 檔案.xml> <sketch 資料夾> [名稱]`	匯入 LDmicro 1.x 程式或 PLCopen XML（15.1、15.2 節）
`check <專案>`	檢查（同 Shift+F7）
`build <專案>`	建置：寫出 plc86.cpp 等到 sketch（同 Ctrl+B）
`compile <專案> [--ide <資料夾>]`	編譯，不上傳（同 F7）
`upload <專案> <COM> [--ide <資料夾>]`	編譯並上傳（同 F8）
`upload-res <專案> <COM>`	只上傳 86HMI 資源 `data\Resource`（13.6 節）
`sim <專案> <掃描數> [變數...] [--set 變數=值]...`	模擬 N 個掃描並列出變數值
`search <專案> <文字> [--case] [--word]`	在專案中搜尋（同 Ctrl+Shift+H）
`lower <專案> <POU>`	列出 POU 轉成的 ST
`iotable export|import <專案> <檔案.xlsx|.csv>`	I/O 對應表匯出 / 匯入 Excel
`eni <專案> <eni.xml>`	匯入 ENI，建立 EtherCAT 裝置（第 11 章）
`format <檔案>`	自動排版（同編輯器的 Ctrl+T）
`ports`	列出序列埠
`instructions`	列出階梯圖指令（JSON）

指令	用途
`online <專案> <COM> status|read|write|force|unforce|unforce-all|run|stop|reset warm|cold|log|watch`	線上操作（10.4 節）
`online <專案> <COM> sdo|coe <SubDevice> ...`	EtherCAT SDO 讀寫、讀取 CoE 物件字典（11.6 節）

`<專案>` 可以是 `project.86ld`、它所在的資料夾或 `sketch` 資料夾。範例：以第 5 章的 `MyPlc` 專案（`C:\work\MyPlc`）檢查，再模擬 `250` 個掃描（`MainTask 10 ms`，約 `2.5` 秒）：第一個掃描前把 `Rrun` 設為 `TRUE`，列出 `Rrun` 與 `Ylamp`；第 `201` 個掃描（`2` 秒）起 `Ylamp` 變為 `TRUE`。

```
86ldmicro-cli check C:\work\MyPlc
86ldmicro-cli sim C:\work\MyPlc 250 Rrun Ylamp --set Rrun=TRUE
```

--set 只在第一個掃描前寫入一次，不是強制（Force）：之後程式或模擬 I/O 仍會改寫它。已對應到輸入通道的變數（例如 Xstart）每個掃描都會被模擬的輸入覆蓋，所以本例設定內部變數 Rrun。建立新專案用 86ldmicro-cli new C:\work\Test Test（MainTask、空的階梯圖 Main、GVL）。

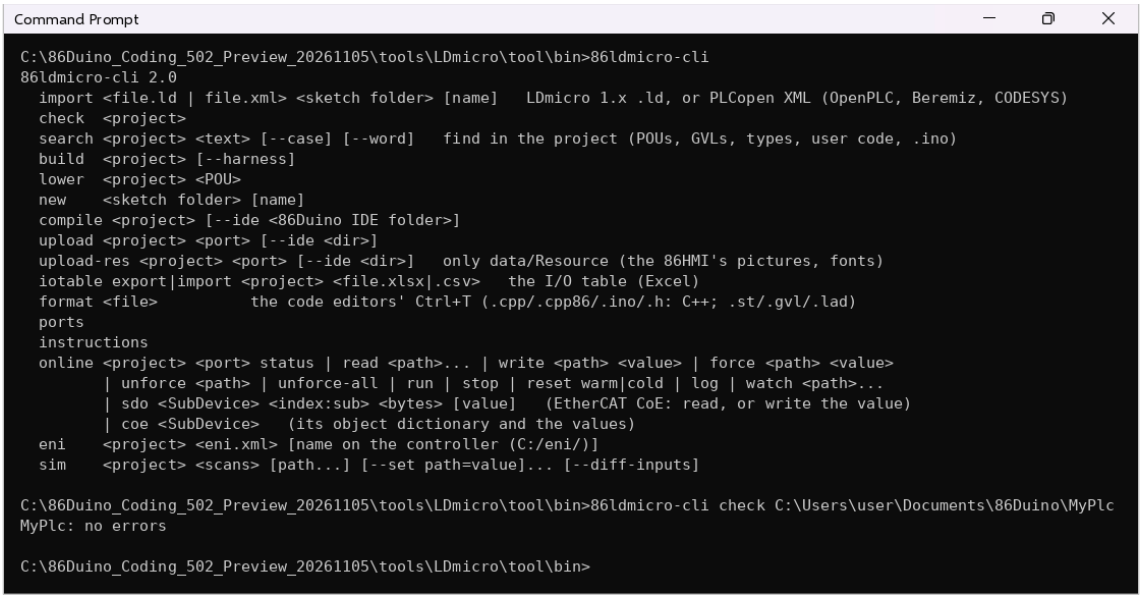


圖 15-12　命令提示字元中執行 86ldmicro-cli

15.6 專案保護

交付給客戶的專案，可以用密碼保護部分程式：「**專案 > 專案保護（密碼）...**」設定密碼後，逐一指定每個 POU 與使用者程式的保護方式。



圖 15-13 「專案保護」：密碼與每個檔案的保護方式

保護方式	效果
無	一般檔案
唯讀	可以開啟、模擬、建置，沒有密碼不能修改
隱藏（加密）	在磁碟上以 AES-256 加密（金鑰由密碼經 PBKDF2-SHA256 導出），磁碟上只保留 PROGRAM 那一行；沒有密碼時不顯示內容，也不能開啟、檢查、模擬或建置

15.6.1 設定保護

1. 「專案 > 專案保護（密碼）...」。
2. 輸入 **密碼** 與 **再輸入一次密碼**（至少 4 個字元，兩次要相同）。
3. 在每個檔案（例如 `Main (pou/Main.lad)`）旁選 **無**、**唯讀** 或 **隱藏（加密）**，按 **確定**。
4. 狀態列提示「已設定保護：可用 專案 > 專案保護 > 立即鎖定（下次開啟專案時會自動鎖定）」。

15.6.2 解鎖、變更與移除

開啟有保護的專案時為鎖定狀態。再按「專案 > 專案保護（密碼）...」輸入專案密碼即解鎖（本次工作階段有效），之後同一選單提供：

選項	說明
變更保護設定	各檔案的保護方式、密碼（新密碼空白：不變）
立即鎖定	受保護的檔案再次需要密碼
移除保護	所有檔案改回未加密，不再需要密碼（會再確認一次）

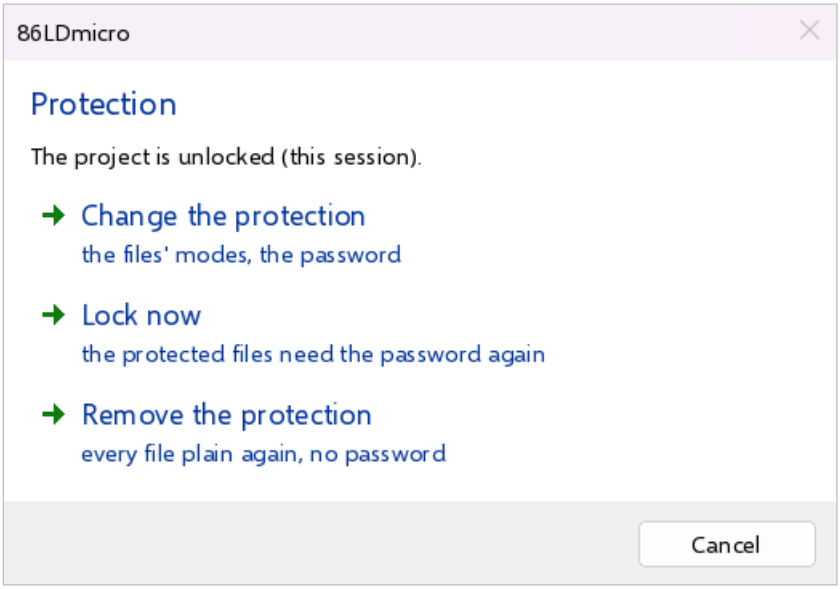


圖 15-14 解鎖後的「專案保護」：變更保護設定、立即鎖定、移除保護

*注意：密碼遺失無法復原，請自行妥善保管。隱藏（加密）的檔案沒有密碼就無法建置，交付前請確認客戶端不需要修改這些程式。

第 16 章　常見問題

本章整理安裝、上傳、程式、模擬與線上、86HMI、EtherCAT 與運動控制最常遇到的問題。錯誤訊息的完整對照見 9.6 節。

Q1：86Duino IDE 的「工具」選單沒有 86LDmicro？

確認 86LDmicro 的檔案解壓在 IDE 的 tools\LDmicro\tool（其中有 LDmicro.jar），然後重新啟動 86Duino IDE。若執行時被 Windows 封鎖，表示解壓前未「解除封鎖」：對 zip 檔按右鍵 > 內容 > 解除封鎖後重新解壓（2.3 節）。

Q2：按「編譯」或「上傳」時出現「找不到 86Duino IDE」？

單獨啟動 86LDmicro（不是從 86Duino IDE 的「工具 > 86LDmicro」開啟）時，需要指定 IDE 的位置：「**工具 > 選項...**」設定 86Duino IDE 資料夾（2.6 節）。

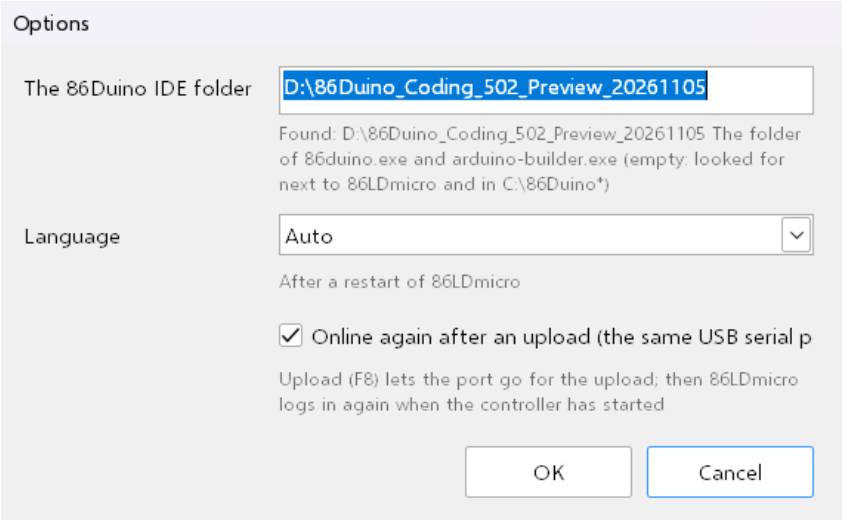


圖 16-1　選項：86Duino IDE 資料夾、語言、上傳後自動重新連線

Q3：EtherCAT 或 Motion86 的專案編譯失敗（找不到 sdoUpload_Begin、IKinematicsCB、expected class-name 等）？

86Duino IDE 的版本不符。請使用 **86Duino Coding 502 Preview（20261105）**整包 IDE：它內附新版 Motion86（9 軸）與它需要的底層函式庫。只把新版 Motion86 函式庫複製到舊版 IDE 會編譯失敗。

Q4：上傳失敗：無法開啟 COMx（序列埠被其他程式占用？）

同一個 COM 埠被其他程式開著，最常見的是 86Duino IDE 的序列埠監控視窗或終端機程式。關閉它們後再按 **上傳**（F8）。86LDmicro 自己的「序列埠監看」與線上監看在上傳時會自動讓出序列埠，不需要手動關閉。

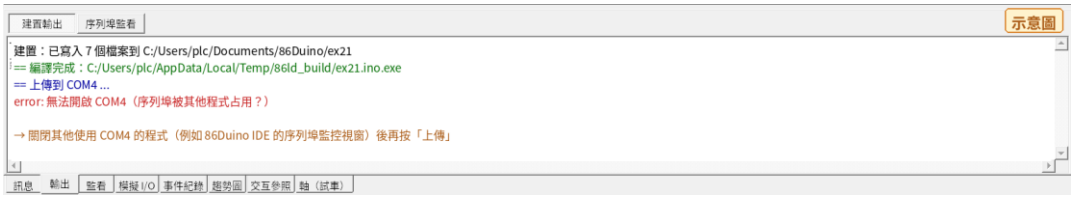


圖 16-2　上傳失敗：無法開啟 COM4（序列埠被其他程式占用）

輸出中的訊息	處理
找不到 COMx	USB 線拔掉、選錯埠或控制器未送電：依 2.4 節確認 COM 埠，在工具列重新選擇
上傳到一半逾時	USB 線或 HUB 不穩、控制器重新開機中：直接接電腦的 USB 孔，重新上傳
編譯錯誤（.cpp 檔名與行號）	C/C++ 使用者程式或函式庫錯誤：在「訊息」雙擊，跳到 .cpp86 的行號修正

Q5：上傳成功，但按下實體按鈕沒有反應？

最常見的原因是**變數沒有 I/O 對應**。PLC 程式只讀寫變數，變數與實體點（例如 `LocalIO.X00`）的連結在「I/O 對應」頁設定（4.6 節）。建置時「訊息」會出現警告：

「Xstart 在這裡只被讀取：沒有 I/O 對應，也沒有任何地方寫入它 —— 它永遠是 FALSE。請把它對應到輸入（I/O 對應，或 EtherCAT > 變數對應），或在程式 / 86HMI 按鈕中設定它」

1. 專案樹雙擊「I/O 對應」，確認每個輸入、輸出變數都有通道。
2. 沒有時按 **新增...**，選變數與通道（例如 `Xstart` → `LocalIO.X00`）。
3. 重新 **上傳**（F8）。

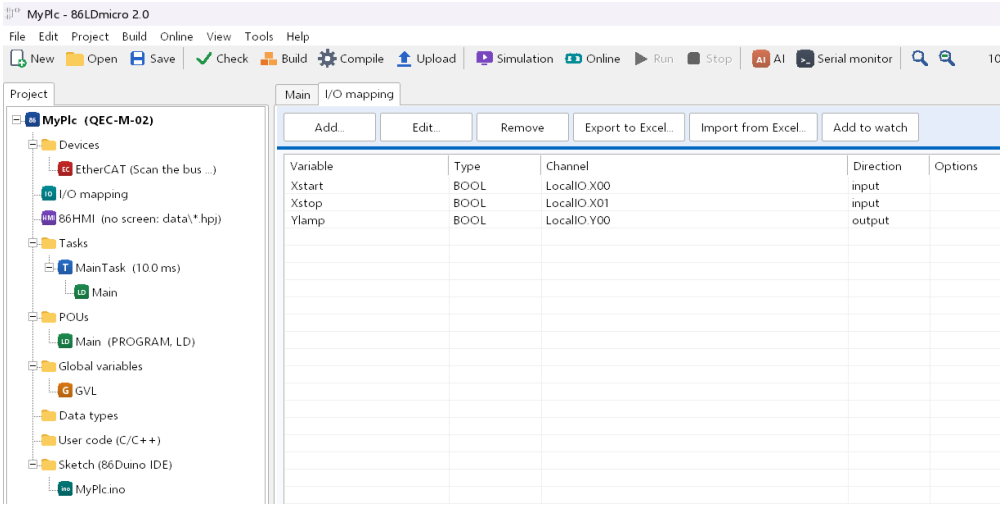


圖 16-3　I/O 對應頁：每個變數對應一個實體通道

Q6：階梯圖的元件變成紅底？

該元件的變數未宣告、名稱打錯，或型別不符（例如把 REAL 當接點）。切到下方「訊息」，雙擊錯誤列就會選到該元件；修正名稱或在 GVL / VAR 宣告後按 **檢查**（Shift+F7），直到沒有錯誤。常見訊息與處理見 9.6 節。

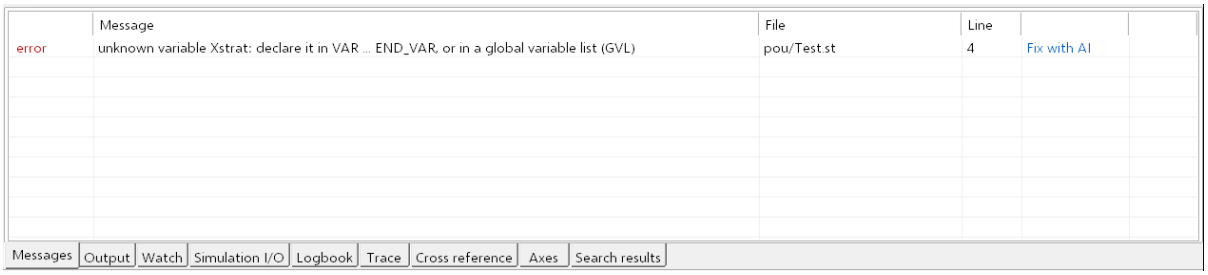


圖 16-4　「訊息」視窗：每列一個錯誤，雙擊跳到該元件

Q7：線上（Ctrl+F5）連線被拒絕，或一直沒有回應？

- **被拒絕**（控制器上的程式與這次建置不同）：控制器跑的不是目前專案這次建置的程式，請重新 **上傳**（F8）。
- **沒有回應**：確認「專案設定」的「線上監看」已啟用、埠為 Serial（USB）；使用者程式與 `.ino` 不要占用 USB `Serial`；改了設定要重新建置上傳。
- 連線中斷 3 秒，控制器會自動解除所有強制。

Q8：程式跑一陣子就停住（或沒接電腦時停住），接上 86LDmicro 又恢復？

通常是程式對 USB `Serial` 持續列印。電腦沒有讀取時（86LDmicro 未連線、序列埠監看關閉），`Serial.print()` 寫滿緩衝區後會等待，整個程式（包括 PLC 掃描）跟著停住。正式運轉時不要持續列印到 `Serial`；與外部設備通訊用 `Serial1`～`Serial3`；除錯列印前先判斷 `Serial.availableForWrite()` 是否足夠（14.6 節）。

Q9：強制了變數，卻沒有作用？

強制值在每次掃描**開始**（讀入輸入後）與**結束**（寫出輸出前）寫入。掃描中被程式重寫的變數，後面的程式讀到的是程式寫的值。所以強制只由 I/O 對應寫入的變數（實體輸入）與輸出最有效。

- 用交互參照（Ctrl+Shift+F）查這個變數被哪些程式寫入（9.5 節）。
- 例：ex30 展示模式（`gDemo = TRUE`）時，PlantSim 會重寫 `sim*`、IoMap 會重寫 `x*`，強制它們會被覆寫；請先關閉展示模式（`gDemo = FALSE`），再強制 `diEntry` 等實體輸入。

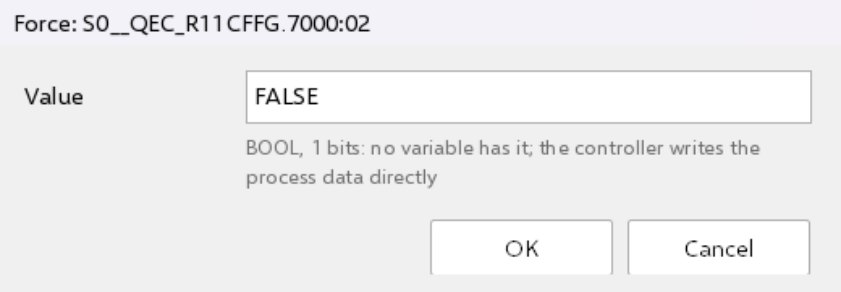


圖 16-5　強制一個變數的值

Q10：模擬時在監看視窗寫入輸入變數，值馬上又變回去？

已對應實體輸入的變數，模擬時由「**模擬 I/O**」決定（每個掃描都重新讀入）。請到下方「模擬 I/O」雙擊該通道切換 ON / OFF（或輸入類比值），或使用強制（9.2、9.3 節）。

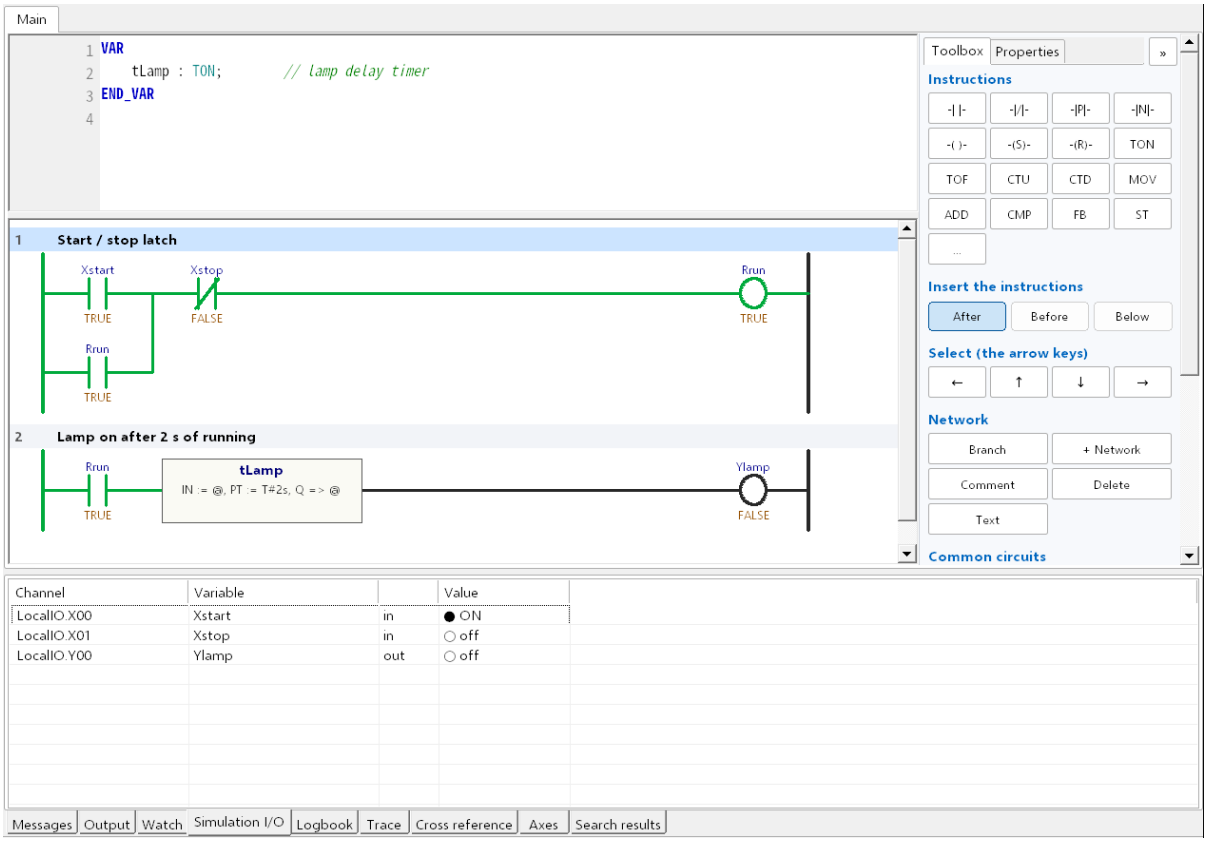


圖 16-6　「模擬 I/O」：雙擊通道切換模擬的實體輸入

Q11：86HMI 畫面上的按鈕按了，PLC 沒有反應？

- 按鈕綁定的是**已對應實體輸入的變數**：下一個掃描會被實體輸入覆蓋。請改綁內部變數（例如 `Rstart`），再在程式中與實體按鈕並聯（13.3 節）。
- 綁定的用法是「事件」，但同一個變數在兩個工作中讀取：只保證第一個工作讀得到，建置時有警告。請只在一個工作讀取，必要時用 SET 鎖存。
- 模擬中才改了綁定：綁定在下次啟動模擬（F5）時生效，先結束模擬再按 F5。
- 86HMI 頁表格中該元件的「變數」欄是空的：還沒有綁定（13.2 節）。

Q12：86HMI 模擬時，ToggleButton 的文字不會在「未按下」與「按下」之間切換？

這是 2.0.0 的**已知問題**：ToggleButton 的狀態切換正常（綁定的變數也正確），只有模擬視窗中按鈕上的文字不會變。實機上由 86HMI 產生的程式負責更新文字，不受影響。

Q13：86HMI 的語系按鈕在模擬時正常，上傳到控制器後按了卻不切換語系？

語系按鈕必須在 **86HMI 編輯器**中寫切換腳本：在按鈕的 Clicked 事件加入 `Hmi.setLanguage(1);`（數字為語系編號），按「產生程式碼」後重新上傳（13.6.2 節）。86LDmicro 的綁定只處理變數，不會替你切換語系。

Q14：EtherCAT 沒有作用（從站的輸入輸出不動）？

1. 開「**序列埠監看**」（不需上線）：正常時每 5 秒顯示 `EtherCAT: running, N slaves, ...`，否則顯示失敗原因。
2. `begin failed`：EtherCAT 線未接到主站的 EtherCAT 埠，或從站未送電。
3. `no slave`：線未接、從站未送電，或接在從站的 OUT 而不是 IN。
4. `the ENI … cannot be used`：ENI 未上傳到主站或與實際網路不符：重新 Scan、匯出並上傳 ENI（第 11 章）。
5. sketch 的 `.ino` 不可再呼叫 1.x 的 `EVA.begin()`；移除 `#include "myeva.h"` 與 sketch 中的 `myeva.cpp` / `myeva.h`。
6. 線上時看「事件紀錄」的 EtherCAT 訊息。



圖 16-7　事件紀錄：EtherCAT 主站啟動訊息與程式開始執行

Q15：軸不動、方向相反，或原點復歸後軸頁顯示的位置不對？

現象	處理
軸不動	確認已激磁（`MC_Power.Status`）、驅動器沒有警報、ENI 的 PDO 有 607Ah / 6064h；機台頁「線上」看狀態（第 12 章）
方向相反	軸頁勾選「反轉方向」，重新建置上傳
原點復歸後，顯示的位置與預期有固定差	軸頁「顯示的回授」改用其他來源（例如 QEC-R11MP3S 的編碼器）時，本版的原點復歸**不會重新對齊該來源**。請用「顯示的回授」的偏移量對齊；CSP 控制、到位、原點、極限仍使用 6064h，不受影響
驅動器未激磁時被手動推動	位置參考會偏移（事件紀錄會提示）：再做一次原點復歸或 `MC_SetPosition`

Q16：MC 移動功能塊（MC_MoveAbsolute 等）只執行一次？

PLCopen MC 功能塊在 `Execute` 的**上升緣**才啟動。要再執行一次，`Execute` 必須先回到 FALSE、再設為 TRUE。同一台機台的移動會依序排隊（Buffered），新命令不會中斷正在進行的移動；`MC_Stop` / `MC_Halt` 為立即停止，沒有減速斜坡（12.7、12.9 節）。

Q17：ST 編輯器按 Ctrl+Space 沒有出現補全清單？

本版 ST 編輯器的補全是**輸入變數或關鍵字的前幾個字時自動列出**（顯示型別與註解），Ctrl+Space 在 ST 編輯器中不會開啟清單。C/C++ 使用者程式與 sketch 檔案的編輯器可以用 Ctrl+Space。

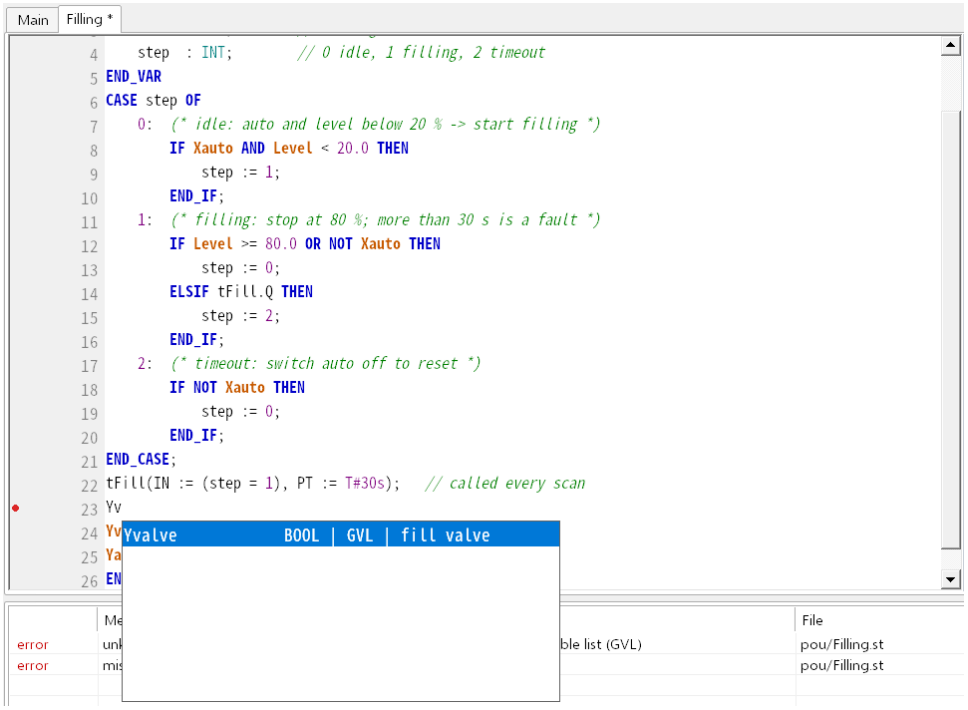


圖 16-8　ST 的補全：輸入 Yv 即列出 Yvalve（BOOL，GVL，註解）

Q18：匯入 PLCopen XML 後少了部分程式？或在外部改了檔案，86LDmicro 沒有更新？

- PLCopen XML 中的 **FBD 與 IL** 程式本版不匯入，會在「輸出」與「訊息」中列出。請在原工具中改寫為 ST 或 LD 後再匯出（15.2 節）。
- 用外部編輯器、AI / MCP 或版本控制改過專案的檔案後，按「**檢視 > 重新整理：從磁碟重新讀取專案**」（Ctrl+R）。有未儲存的修改時會先詢問；模擬或線上時請先停止。

回報問題時請附上：

- 86LDmicro 版本（`tool\VERSION.txt`，例如 `2.0.0 2026-11-11`）與硬體組合（控制器、模組、驅動器型號）。
- 操作步驟與畫面截圖；「輸出」視窗的建置訊息、「事件紀錄」內容。
- 必要時附上專案資料夾（sketch 內的 `data\86ld`）的壓縮檔。聯絡方式見本手冊最後的「聯絡我們」。

附錄 A　LDmicro 1.1 對照

熟悉 LDmicro 1.1（使用手冊 v1.6.3）的使用者，可用下表對照 86LDmicro 2.0 的做法。1.1 的專案匯入方式見 15.1 節。

項目	LDmicro 1.1	86LDmicro 2.0
啟動	86Duino Coding「工具」選單啟動 LDmicro	86Duino IDE「工具 > 86LDmicro」或 IDE 工具列的 86LDmicro 按鈕（2.5 節）
程式檔	一個 `.ld` 檔（`data\myLD.ld`）	專案資料夾 `data\86ld`：`project.86ld`、`pou`、`gvl`、`dut`、`usercode`（4.1 節）
程式語言	階梯圖（含 ST block）	LD、ST、SFC（IEC 61131-3）與 C/C++ 使用者程式
選 MCU / 板子	Settings → General（PLC Configuration）	「專案 > 設定」的控制器（預設 QEC-M-02）
掃描週期	Settings → General 的 PLC Cycle Time	「工作（Task）」的週期（預設 MainTask 10 ms），可有多個工作（4.3 節）
COM 埠與鮑率	Settings → General 的 COM Mapping、Baudrate	「裝置」的 COM 裝置（4.4 節）
Modbus	Settings → Modbus，變數字首 M / I / H	「裝置」的 modbus、modbus-server 與 I/O 對應
EtherCAT	Settings → Virtual Arduino + 86EVA Configurator（`myeva.h` / `myeva.cpp`、`EVA.begin()`）	EtherCAT Configurator → 匯出 ENI → 86LDmicro 讀回 ENI，從站成為裝置（第 11 章）；`.ino` 不再呼叫 `EVA.begin()`
變數與型別	字首決定類型（X 輸入、Y 輸出、R 內部、T 計時器、C 計數器），IO List 雙擊改大小	在 GVL / VAR 宣告 IEC 型別（BOOL、INT、REAL、TIME…）；字首只是命名習慣（附錄 C）
I/O 腳位	IO List 的 Pin Name 欄雙擊選腳位	「I/O 對應」頁：變數 ↔ 通道（例如 `LocalIO.X00`），含反相、濾波、比例（4.6 節）
計時器、計數器	TON、TOF、CTU… 指令（以週期計數）	IEC 功能塊 TON、TOF、TP、CTU…（`CALL`，以時間計）；1.x 指令仍可用（第 7 章）
產生 C 程式	Code Generation 按鈕	**建置**（Ctrl+B）
產生的檔案	`myplc.cpp` / `myplc.h`、`myeva.h` / `myeva.cpp`、`ladder.h`	`plc86.cpp` / `plc86.h`、`plc86_rt.*`、`plc86_ecat.*`（14.5 節）
`.ino` 的進入點	`EVA.begin()` → `setupPlc()` / `loopPlc()`	`plc86_setup()` / `plc86_loop()`（建置自動補上）
編譯與上傳	回到 86Duino Coding 按「上傳」	86LDmicro 直接 **編譯**（F7）、**上傳**（F8）（10.1 節）
模擬	F7 模擬模式、F8 即時模擬、Space 單一週期	**模擬**（F5），速度 x1 / x10；模擬 I/O、86HMI 模擬視窗（第 9 章）

項目	LDmicro 1.1	86LDmicro 2.0
線上監看	線上監看（ldmon）	**線上**（Ctrl+F5）：能流、監看、寫入、強制、RUN / STOP（10.4 節）
86HMI	86HMI 綁定對話框	86HMI 頁與綁定表單（同樣的概念），一個元件可多個綁定（第 13 章）
使用者 C 程式碼	User Code（global / setup / beforescan / afterscan / loop / ethercat）	使用者程式檔 `.cpp86` 的掛勾函式，`GVL()` 存取變數（第 14 章）
AI	編譯錯誤時跳出 AI 視窗	AI 助理分頁、「訊息」的「用 AI 修正」、MCP（15.3、15.4 節）
說明	F1 或 File → Manual...（內建 manual.txt）	「說明」選單開啟線上文件網站；`tool\docs` 為離線版

附錄 B　快捷鍵

以下為 86LDmicro 2.0.0 的全部快捷鍵（主視窗的加速鍵、各編輯器的按鍵）。

B.1　主視窗（任何分頁）

按鍵	功能	選單
Ctrl+N	新增專案	檔案
Ctrl+O	開啟專案	檔案
Ctrl+S	儲存	檔案
Ctrl+Z / Ctrl+Y	復原 / 重做	編輯
Ctrl+X / Ctrl+C / Ctrl+V	剪下 / 複製 / 貼上	編輯
Del	刪除	編輯
Ctrl+A	全選	編輯
Ctrl+F	在此分頁尋找	編輯
F3 / Shift+F3	尋找下一個 / 上一個	編輯
Ctrl+Shift+F	在專案中尋找變數（交互參照）	編輯
Ctrl+Shift+H	在專案中搜尋（搜尋結果）	編輯
Shift+F7	檢查	建置
Ctrl+B	建置（寫出 sketch）	建置
F7	編譯（86Duino）	建置
F8	上傳到控制器	建置
F5	模擬（開始 / 結束）	線上
Ctrl+F5	登入控制器（線上）	線上
F9 / Shift+F9	RUN / STOP	線上
Ctrl+ +（或數字鍵盤 +）	放大	檢視
Ctrl+ -（或數字鍵盤 -）	縮小	檢視
Ctrl+0（或數字鍵盤 0）	100%	檢視
Ctrl+滑鼠滾輪	放大 / 縮小	—
Ctrl+\	左右並排：把前面的分頁移到右側 / 結束	檢視
Ctrl+R	重新整理：從磁碟重新讀取專案	檢視

按鍵	功能	選單
Ctrl+W、滑鼠中鍵	關閉分頁	—
Ctrl+Tab / Ctrl+Shift+Tab	下一個 / 上一個分頁	—

B.2 尋找列與專案樹

按鍵	功能
Enter 或 F3	尋找列：下一個
Shift+Enter 或 Shift+F3	尋找列：上一個
Esc	關閉尋找列
Ctrl+ + / Ctrl+ - （專案樹有焦點時）	全部展開 / 全部收合
Enter（搜尋結果）	跳到選取的那一處（同雙擊）

B.3 階梯圖編輯器

按鍵	功能
← / → / ↑ / ↓	移動選取
Ctrl+← / Ctrl+→	把選取的元件往左 / 右移
Ctrl+↑ / Ctrl+↓（選取網路時）	把網路往上 / 下移
Home / End	選取第一個 / 最後一個網路
Enter、雙擊	編輯選取的元件參數；選取網路時編輯網路標題與註解
Del、Backspace	刪除選取的元件或網路
Ctrl+Enter / Ctrl+Shift+Enter	在選取的網路之後 / 之前插入網路
Ctrl+C / Ctrl+X / Ctrl+V	複製 / 剪下 / 貼上
Ctrl+Z / Ctrl+Y（或 Ctrl+Shift+Z）	復原 / 重做
Ctrl+T	文字模式（.lad 文字）
Shift + 指令鍵	插在選取元件之前
B	在選取元件下方並聯一個常開接點

插入指令的按鍵（插在選取元件之後，或依右側欄「指令插入位置」）：

鍵	指令	鍵	指令	鍵	指令
C	常開接點 NO	S	置位線圈 SET	D	CTD
N	常閉接點 NC	R	復位線圈 RST	M	MOV
P	正緣接點 PT	O	TON	A	ADD
L	線圈 OUT	F	TOF	E	條件接點 CMP
K	功能塊 CALL	U	CTU	T	ST 敘述

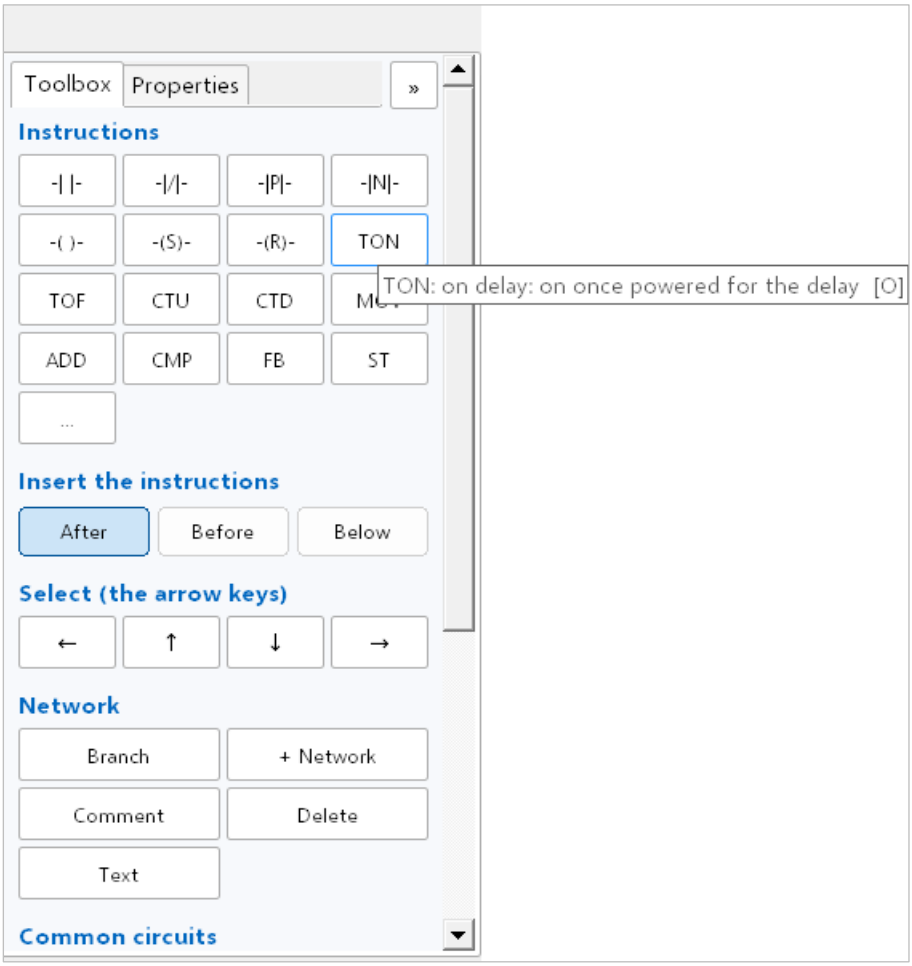


圖 B-1　右側欄「工具箱」：滑鼠停在指令按鈕上顯示說明與按鍵

B.4 SFC 編輯器

按鍵	功能
Insert	+ 步驟：在選取處之後加一個轉移條件與步驟
Enter、雙擊	編輯選取的步驟或轉移條件
Del	刪除（步驟連同其轉移條件）
↑ / ↓	同一序列的前一個 / 後一個項目
← / →	相鄰的分支
Ctrl+Z / Ctrl+Y（或 Ctrl+Shift+Z）	復原 / 重做
Ctrl+T	文字模式（.sfc 文字）

B.5 程式碼編輯器（ST、GVL、資料型別、C/C++、sketch）

按鍵	功能
Ctrl+T	自動排版（只調整行首縮排與多餘空白行，Ctrl+Z 可復原）
Ctrl+Space	開啟補全清單（C/C++ 編輯器）。ST 編輯器：輸入前幾個字時自動列出，本版 Ctrl+Space 不開啟清單
↑ / ↓、PgUp / PgDn	補全清單中移動
Enter 或 Tab	補全清單：插入選取的項目
Esc	關閉補全清單或參數提示
Tab / Shift+Tab	縮排 4 格 / 取消縮排
Enter	換行並保持縮排（{、THEN、DO 等之後多縮一層）
Ctrl+Z / Ctrl+Y	復原 / 重做

B.6 AI 助理

按鍵	功能
Ctrl+Enter	傳送問題
Esc	關閉「AI 助理：設定」

附錄 C　變數命名與型別

C.1　命名習慣

86LDmicro 2.0 的變數型別由宣告決定，名稱沒有限制。為了閱讀與維護方便，建議沿用 LDmicro 1.1 的字首習慣（本手冊的範例都照此命名）：

字首	用途	宣告範例	說明
X	輸入	`Xstart : BOOL;`	對應實體輸入（I/O 對應）
Y	輸出	`Ylamp : BOOL;`	對應實體輸出
R	內部旗標	`Rrun : BOOL;`	自保持、HMI 按鈕、步驟旗標
T	計時器實例	`Tdelay : TON;`	每個計時器一個實例
C	計數器實例	`Cparts : CTU;`	每個計數器一個實例
M	數值	`Mposition : INT;`	位置、設定值、量測值

C.2　命名規則

- 只能使用英文字母、數字與底線 `_`，不能以數字開頭；不區分大小寫（`Xstart` 與 `XSTART` 是同一個變數）。
- 不能與 IEC 保留字同名（`TON`、`IF`、`END_VAR`...）；從 1.1 匯入時這類名稱會加上 `_v`。`__` 開頭的名稱保留給 86LDmicro 內部使用。
- 宣告在 **GVL** 的變數所有 POU、86HMI、C/C++ 都看得到；宣告在 POU 的 `VAR` 中只有該 POU 可用（其他 POU 以 `程式名.變數` 讀取）。
- 同一個名稱在 GVL 中只能宣告一次。

C.3 資料型別與範圍

型別	大小	範圍	C 型別
BOOL	1 位元	FALSE / TRUE	`bool`
SINT	8 位元	-128 ~ 127	`int8_t`
USINT	8 位元	0 ~ 255	`uint8_t`
INT	16 位元	-32,768 ~ 32,767	`int16_t`
UINT	16 位元	0 ~ 65,535	`uint16_t`
DINT	32 位元	-2,147,483,648 ~ 2,147,483,647	`int32_t`
UDINT	32 位元	0 ~ 4,294,967,295	`uint32_t`
LINT	64 位元	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807	`int64_t`
ULINT	64 位元	0 ~ 18,446,744,073,709,551,615	`uint64_t`
BYTE / WORD / DWORD / LWORD	8 / 16 / 32 / 64 位元	位元字串（無號）	`uint8_t`～`uint64_t`
REAL	32 位元	IEEE 754 單精度，約 ±3.4E38，有效位數約 7 位	`float`
LREAL	64 位元	IEEE 754 倍精度，約 ±1.8E308，有效位數約 15 位	`double`
TIME	32 位元	毫秒，±2,147,483,647 ms（約 ±24.8 天），常數最大 T#24d20h31m23s647ms	`int32_t`
STRING / STRING[n]	n + 1 位元組	字串，預設 80 個字元，最多 254 個字元	`char[n+1]`
列舉（TYPE … (A, B, C)）	32 位元	宣告的值	`int32_t`

陣列（ARRAY[1..5] OF INT）、結構（STRUCT）、列舉在「資料型別」中定義（4.5 節）。86LDmicro 2.0.0 的 IEC 執行環境尚未支援 WSTRING、DATE、TIME_OF_DAY（TOD）、DATE_AND_TIME（DT）、POINTER、REF_TO。

*__注意：__數值超出型別範圍時，檢查會出現「數值 … 超出 INT 範圍」之類的錯誤；計數、累加可能超過 32,767 時請使用 DINT。

C.4 常數與 TIME 寫法

種類	寫法	值
TIME	`T#500ms`、`T#2s`、`T#1.5s`、`T#1m30s`、`T#1h30m`、`T#1d`	單位 d、h、m、s、ms 可組合，數字可有小數點與底線
TIME（完整寫法）	`TIME#2m15s`	同 `T#2m15s`
16 進位、2 進位、8 進位	`16#FFFF`、`2#1010_1010`、`8#777`	65535、170、511
指定型別	`INT#-5`、`REAL#1.5`	該型別的常數
實數	`1.5`、`1.5E3`	1.5、1500.0
字串	`'hello'`	單引號
布林	`TRUE`、`FALSE`	

TIME 與整數的轉換用 `TIME_TO_DINT(t)`（毫秒）與 `DINT_TO_TIME(ms)`；比較時間要與時間比較，例如 `IF Fill.T >= T#5s THEN`。宣告範例（以上寫法皆經檢查與模擬確認）：

```
VAR
    tFill   : TIME := T#1m30s;     (* 90000 ms *)
    tPulse  : TIME := T#1.5s;      (* T#1s500ms *)
    Mask    : WORD := 16#FFFF;
    Pattern : BYTE := 2#1010_1010;
    Total   : LINT;
    Ratio   : LREAL := 1.5E3;
    Msg     : STRING[20] := 'hello';
END_VAR
```

聯絡我們

若您在依照本手冊操作時遇到任何技術問題，或對 QEC 系列產品有採購、客製化需求，請透過以下管道與我們聯絡：

項目	聯絡方式
業務 / 採購	info@qec.tw（QEC 產品線）
技術支援 / 研發	info@icop.com.tw（ICOP Technology Inc.）
線上文件	https://www.qec.tw/86duino/86ldmicro/

86LDmicro 的「說明」選單會直接開啟線上文件網站；最新的使用手冊與說明文件請以網站為準。