FATEK

M Series

Programmable Controller

結構化語言 ST 使用手冊

因手冊內容會隨著版本變更而做修改，此版本不一定會是最終版本。
若要下載最新版的手冊請到 www.fatek.com 的技術支援專區。

永宏電機股份有限公司

目錄

前言

請在使用本產品前閱讀並確實瞭解本手冊之內容後再行使用。如若有任何疑問或意見，請洽詢 FATEK 代理經銷商保固內容和責任限制。

使用本產品注意事項

符合應用之條件
FATEK 的產品適用評估並安裝於經過全面設計的設備或系統。 請使用者自行確認目前所使用的系統、機械或是裝置是否適用於 FATEK 產品。如未確認是否符合或適用時，本公司無須對產品的適用性負責。 如客戶要求，FATEK 將提供相應的第三方認證來明確適用於產品的額定值和使用限制。該認證信息本身不足以完全決定 FATEK 產品與最終產品、機器、系統及其它應用或組合的適用性。 以下為一些必須引起特別注意的應用場合，但下述內容並非為包括所有可能的產品用途，也不表示所列用途對產品均適用：戶外使用、在遭受潛在化學污染或電氣干擾處使用、或未在本手冊中提及的條件或用途。可能對生命或財產造成風險的系統、機器和設備。 務必事先確認系統整體是否有危險告示、並採用備援設計等可確保安全性的設計，否則不得將產品用於與人身財產安全密切相關的場合。FATEK 對於客戶在其應用中的產品組合或產品使用的規格、法規或限制等，不承擔任何責任。 使用本產品時，FATEK 不對用戶編輯的程式或其引起的後果承擔任何責任。

免責聲明

尺寸和重量
手冊記載的尺寸和重量僅為名義值，即使已說明了公差，也不能用於製造用途。

性能數據
本手冊中給出的這些數據僅表示在 FATEK 測試條件下的性能數據僅供用戶作為確定符合應用的參考，用戶必須將其與實際應用條件互相考慮。實際性能遵從 FATEK 保證內容和責任限制。

錯誤和疏忽
本手冊中的內容已仔細核對並認為是準確的；但對於文字、印刷和校對錯誤或疏忽不承擔任何責任。

規格變更
產品規格和附件可能會因技術改進或其它原因而隨時變更。當公佈的規格、性能改變，或者進行過重大的結構改變時，FATEK 通常會改變型號。若產品的某些規格發生變更時，以下情況不另行通知：根據客戶的要求，對客戶的應用指定特別的型號或設定特定的規格。歡迎隨時洽詢 FATEK 代理經銷商，確認所購產品的實際規格。

安全注意事項

安全注意的標示與意義

以下標示用於本手冊中，以提供 M 系列 PLC 安全使用所需的注意事項。安全注意事項對於安全使用產品至關重要。因此請務必閱讀、瞭解並遵守安全注意事項中的內容及意義。

警告	表示潛在的危險狀況，如不加以避免，將會造成死亡或嚴重傷害。 此外，還可能導致嚴重的財產損失。

注意	表示潛在的危險狀況，如不加以避免，可能會造成輕度或中度傷害 或財產損失。

	該標誌表示表示應禁止執行的操作。
	該標誌表示表示應強制執行的操作。
	該標誌表示一般注意事項。
	該標誌表示與灼熱表面相關的注意事項。
	該標誌表示與電力系統接線、接地，觸電相關的注意事項。

警告	
請勿在通電狀態下試圖拆卸任何模組或接觸模組內部，否則可能會導致觸電	
請勿在通電狀態下接觸任何端子或端子台，否則可能會導致觸電。	
為了確保系統安全，避免因人為外部因素或 PLC 故障誤動作引起異常動作，應在外部電路中(非 PLC 程序內部)設置以下安全措施，否則可能導致嚴重事故。 外部控制電路中必須設有緊停電路、互鎖電路、限位電路及類似的安全措施。 在執行中遇到嚴重故障報警時，PLC 會將所有輸出停止。但是，I/O 控制和 I/O 暫存器中的錯誤及其它無法檢測的錯誤仍然會引發意外動作。為應對上述錯誤，必須設置外部安全措施以確保系統安全。 若輸出繼電器卡死、燒毀或輸出晶體管毀損，PLC 輸出可能會保持在 ON 或 OFF 狀態。 為應對上述問題，必須設置外部安全措施以確保系統安全。在系統和設備中採取相應的安全措施，即使在使用中發生通訊錯誤或誤動作，也能確保整體系統的安全。	
用戶必須採取相應的故障安全措施，即使在因信號線路損壞、瞬時斷電或其它原因導致信號錯誤、丟失或異常的情況下，也能確保安全。若不採取適當的措施，則可能會因操作不當而導致嚴重事故。	

注意	
請勿在通電狀態下或在關閉電源後立即觸摸電源模組。此時電源模組溫度可能很高，會導致灼傷。	
與電源模組端子台座連接時，請確實壓接合適尺寸的歐式端子，電纜線材鬆動時可能會導致電源模組燒毀或故障。	
確認延長 PLC 週期時間不會造成任何負面影響後，方可執行線上編輯。否則可能會導致輸入訊號無法讀取。	
對目的 I/O 端點進行安全確認後，方可向其它端點傳送 PLC 設定、I/O 表、I/O 暫存器數據等參數，傳送、修改以上的數據內容可能會導致意外動作。	

使用注意事項

使用 M 系列 PLC 時，應遵循以下注意事項。

電源使用

- 請使用手冊中規定的電源電壓。電源電壓錯誤會導致誤動作或設備燒毀。
- 連接模組數量若超過電源模組的額定電流，可能會導致 CPU 模組或其它模組無法啟動。
- 確保使用指定的電源以額定電壓和頻率進行供電。請特別注意供電不穩定的場所，供電錯誤可能會導致誤動作。
- 著手進行以下任何事項前，請務必關閉 PLC 的電源。否則，可能會導致誤動作或觸電。
  (1) 安裝或拆卸電源模組、I/O 模組、CPU 模組或其它任何模組
  (2) 連接電纜或對系統配線
  (3) 連接或斷開連接線
- 使用電源模組時，請務必遵循以下注意事項。
  (1) 在設備輸出處施加的電壓或連接的負載不得超過電源模組額定規格。
  (2) 若電源模組閒置三個月以上的時間，則應將其保存在陰涼乾燥的環境中，以保持其功能的正常性。
  (3) 如果電源模組安裝不當，則會使熱量聚積，從而可能會造成內部元件老化或損壞。請確實連接並使用標準安裝方法。

安裝

- 請勿將 PLC 安裝在高頻噪音干擾源附近。
- 確認端子台、連接線、記憶卡、周邊通訊連接線和其它帶卡扣裝置的部件均嵌合到位，嵌合不當會導致誤動作。
- 連接鄰近模組後，頂部或底部的卡扣必須完全鎖定(即卡入到位)。如果卡扣沒有鎖定確實，可能無法實現正確的功能。

配線

- 請遵循本手冊中的說明以正確執行配線作業。
- 在接通電源前，應仔細檢查所有的配線及開關設定。配線錯誤可能會導致設備燒毀。
- 對安裝位置進行徹底檢查後，方可安裝端子台和連接線。
- 配線時應將標籤保留在模組上。若撕去標籤，可能會因異物落入模組導致誤動作。
- 為保證散熱正常，請在配線完成後撕去標籤。保留標籤可能會導致誤動作。
- 請使用歐式端子進行配線。請勿用裸絞合線直接連接端子，線材老化斷裂可能會導致設備燒毀。
- 施加在輸入模組上的電壓不得超過額定輸入電壓，否則可能會導致設備燒毀。
- 請勿將超出最大開關容量的電壓或負載施加到輸出模組。過電壓或過載，可能會導致設備燒毀。
- 請勿過度拽拉或彎曲電纜。上述動作均可能導致電纜斷裂。
- 請勿在電纜或其它配線上放置物品，否則可能會導致電纜斷裂。
- 電源模組及通訊埠口請正確設置接地線，避免雜訊干擾造成通訊錯誤及設備誤動作。
- 建議使用 M 系列專用 AC 電源模組供給 MPLC 相關模組電力。
- 建議通訊電纜使用雙絞屏蔽線，並正確接地。

使用

- 開始 MPLC 上電操作前，應確保資料數據暫存區的設定正確無誤。
- 在著手執行以下事項前，請確認其不會對系統造成任何負面影響，否則可能會導致意外動作。

  (1) 改變 PLC 的操作模式(RUN 模式/STOP 模式) ；

  (2) 對暫存器中的任一位進行強制致能 / 強制抑能；

  (3) 改變暫存器中的任一字或設定值的當前值。
- 請勿試圖拆解、修理或改裝任何模組，否則可能會導致誤動作、火災或觸電。
- 請勿使 PLC 墜落或使其遭受過度振動或衝擊。
- 若 I/O 保持位置 ON，則當從 RUN 模式切換到 STOP 模式時，PLC 的輸出將會置於

OFF，並將解除所有輸出動作。請確保外部負載不會在上述過程中構成危險因素。

- 當因致命錯誤導致 CPU 模組停止運轉時，部分類型輸出模組能夠在配置階段設定停止運轉時的輸出處理方式，並非一律 OFF。
- 若狀態監視頁或參數設定不當，可能會導致意外動作。即使狀態監視頁或參數設定正確，也應在啟動之前確認受控系統不會受到負面影響。
- 若在絕緣強度試驗中施加了最大電壓或使用開關突然關閉電源，可能會導致 CPU 模組損壞。請使用可變電阻器逐漸調高或調低電壓。
- 在執行耐壓測試或絕緣電阻測試前，應將電源模組上的線路接地端子和功能接地端子分開，否則可能會導致設備燒毀。

運行環境注意事項

- 請遵循本手冊中的說明以正確執行安裝作業。
- 請勿在下列場所運行控制系統：
  (1) 陽光直射處
  (2) 溫度或濕度超出規格中規定範圍的場所
  (3) 由於溫度急劇變化易造成結露現象的場所
  (4) 存在腐蝕性氣體或易燃性氣體的場所
  (5) 存在粉塵(尤其是鐵屑)或煙霧的場所
  (6) 暴露於水、油類或化學品的場所
  (7) 易受衝擊或振動的場所
- 將系統安裝在下列場所時，應採取適當和有效的預防措施：
  (1) 存在靜電或其它形式噪音的場所
  (2) 存在強電磁場的場所
  (3) 可能暴露於放射性污染的場所
  (4) 靠近動力電源的場所

1

認識結構化語言 ST

1-1 ST 語言的特點

早期的自動化控制在編輯可程式邏輯控制器邏輯時，須透過書寫器將類似組合語言的程式簡碼指令 (Mnemonic)輸入控制器中，已達到專案所需求的動作，後續隨著工業環境演變，開發出了控制迴路階梯圖 (LD) 來表達程序的邏輯，讓不擅於撰寫程式的作業人員能以圖像化的方式來撰寫操控 PLC。

現在的 PLC 可以操控運行的邏輯日愈複雜，再加上撰寫程式已經愈來愈普及，使用文字撰寫的 PLC 程式漸漸變為流行，因而催生了類似 Pascal, C 程式語法來撰寫 ，只要是學習過資訊領域的人員，便可以容易上手編程。使用結構化語言 (ST) 開發可程式邏輯控制器的人員越來越多，使得結構化語言 (ST) 成為現今熱門的自動化開發工具之一。

算式的運算處理

算術指令可以和比較指令等一般的表達式一樣進行敘述。

■多個運算可以寫進同一行中

可以使用運算式 （ + 、 -等 ）簡潔地進行敘述，因此 ST 程式比梯形圖更易於理解。

程式範例

在 R3 中代入 R0~R2 的平均值。

R3=(R0+R1+R2)÷3

ST

```
R3:=(R0+R1+R2)/3;
```

LD

複雜的資訊處理

可以通過 if 語法或 for.while 等語法編寫控制程式。與階梯圖相比，能夠更簡潔明瞭的敘述根據不同條件對執行內容進行複雜分支或循環處理等。

程式範例

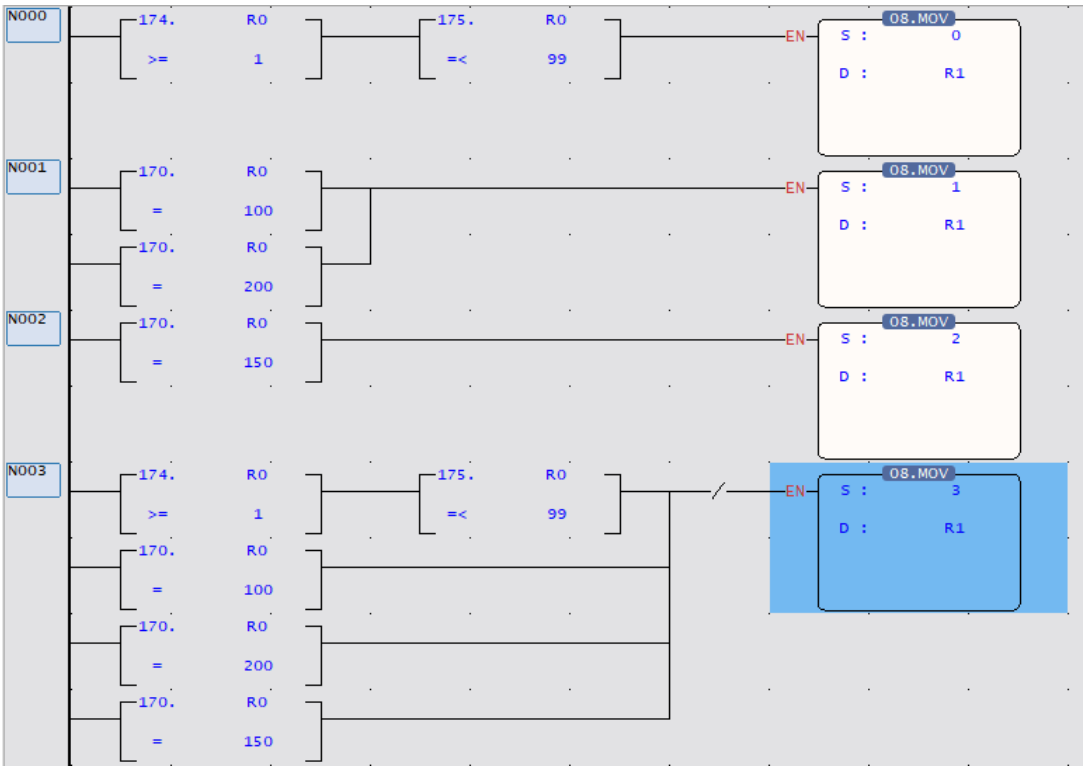
根據 R0 值在不同條件下， R1 設置為 0～3

- **當 R 值為 1～99 時:　　R1=0**
- **當 R 值為 100 或 200 時:　R1=1**
- **當 R 值為 150 時:　　R1=2**
- **滿足上述情形以外時:　　R1=3**

ST

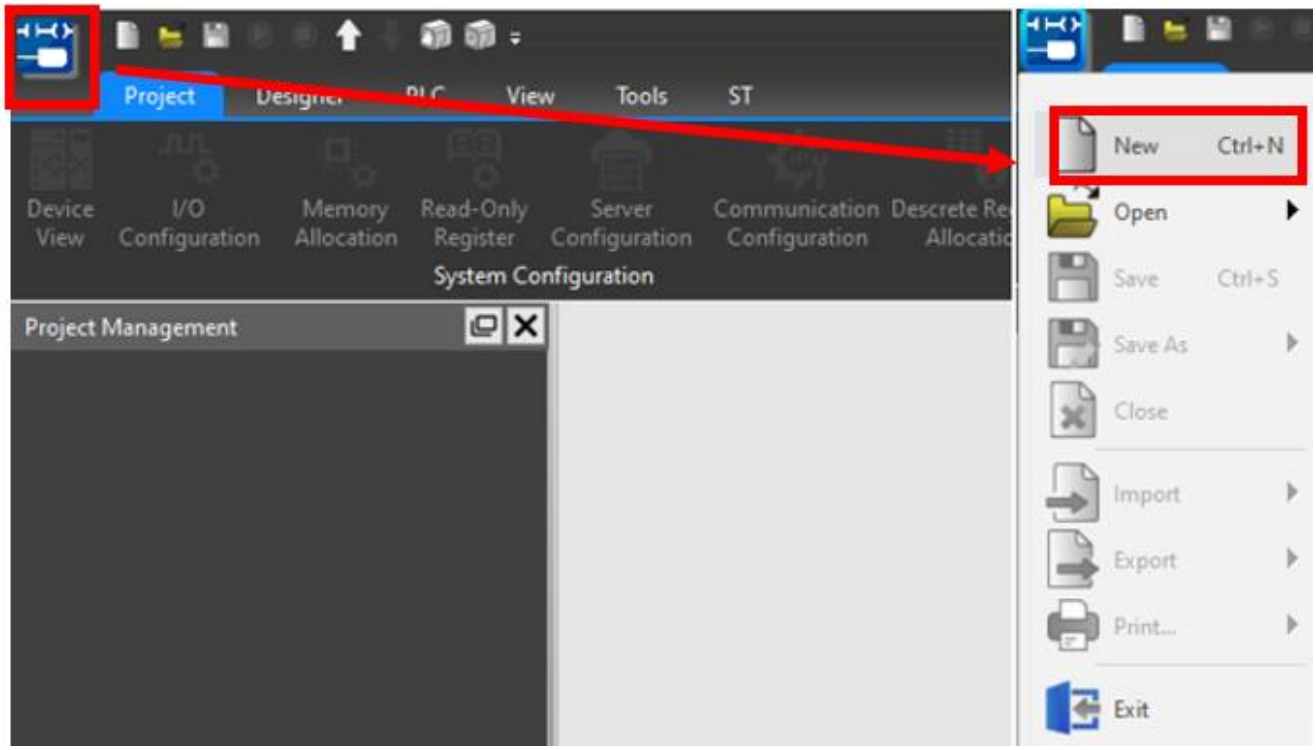
```
IF R0 >= 1   & R0   <= 99   THEN
    R1:=0;
ELSEIF R0 = 100 or R0 = 200 THEN
    R1:=1;
ELSEIF R0 = 150   THEN
    R1:=2;
ELSE
     R1 :=3;
END_IF
```

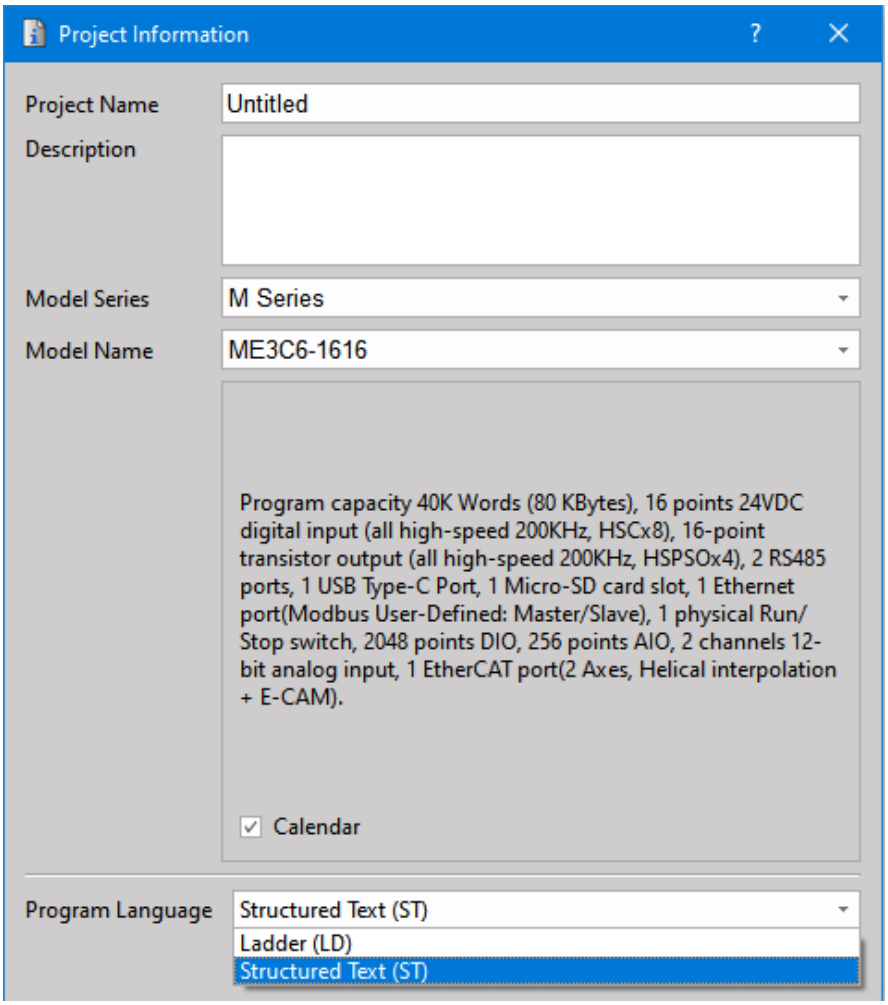
LD



1-2 新增 ST 語言程式

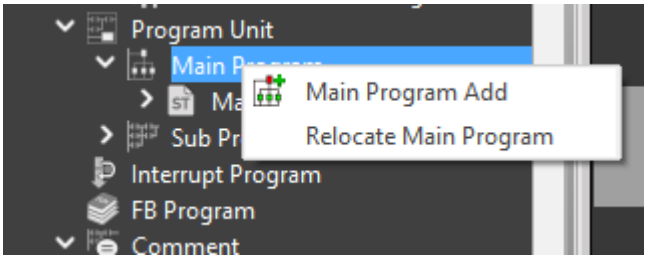
建立新專案

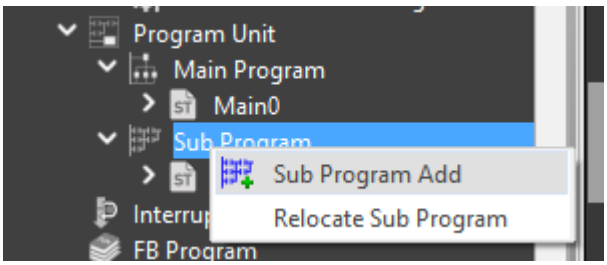




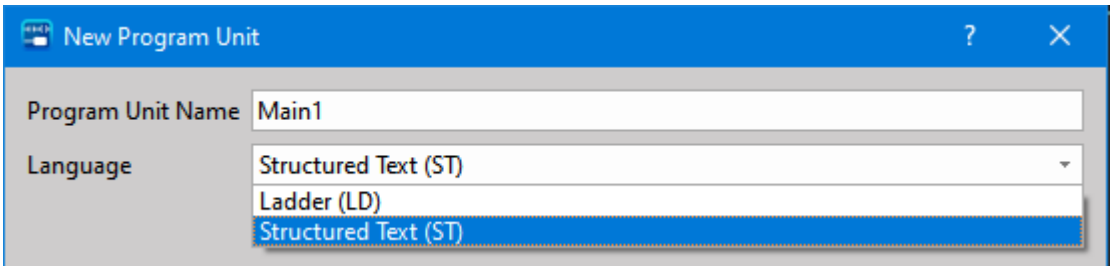
建立主程式/ 副程式(ST)的範例

除了在新專案建立時可選擇編程語言，也可在”主程式”，或是 “副程式”上面按滑鼠右鍵選擇新增主/副 程式時會跳出選擇視窗





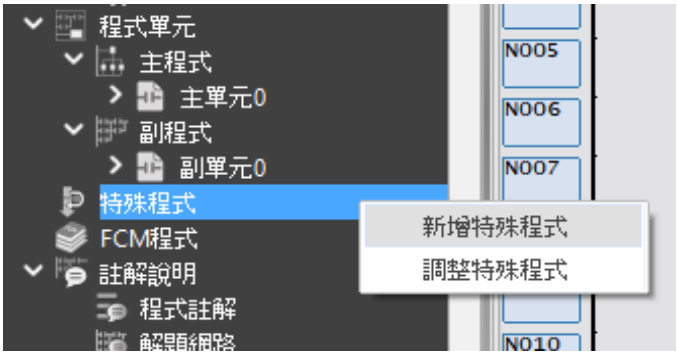
選擇下結構化文字(ST) 選項已新增對應的程式



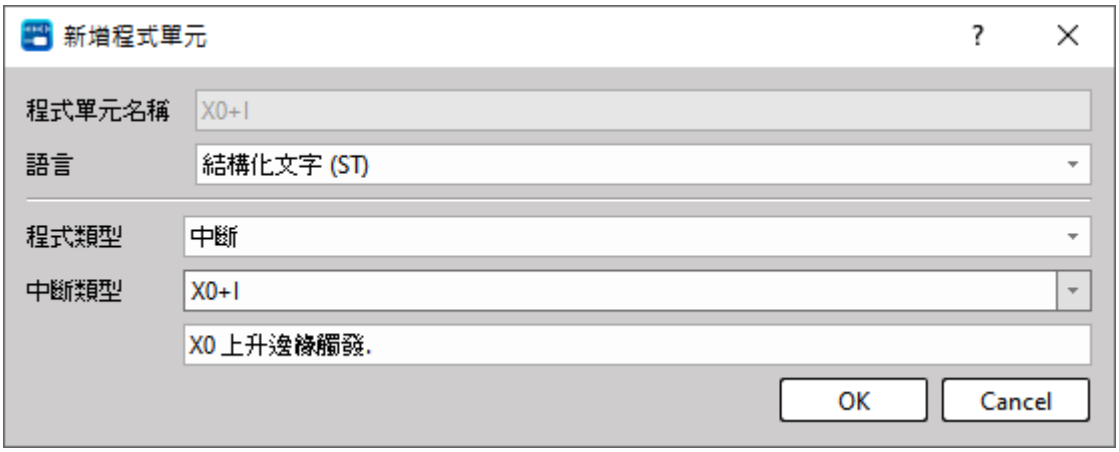
建立特殊程式/中斷程式(ST)的範例

在特殊程式的節點上面點及滑鼠右鍵

新增特殊程式



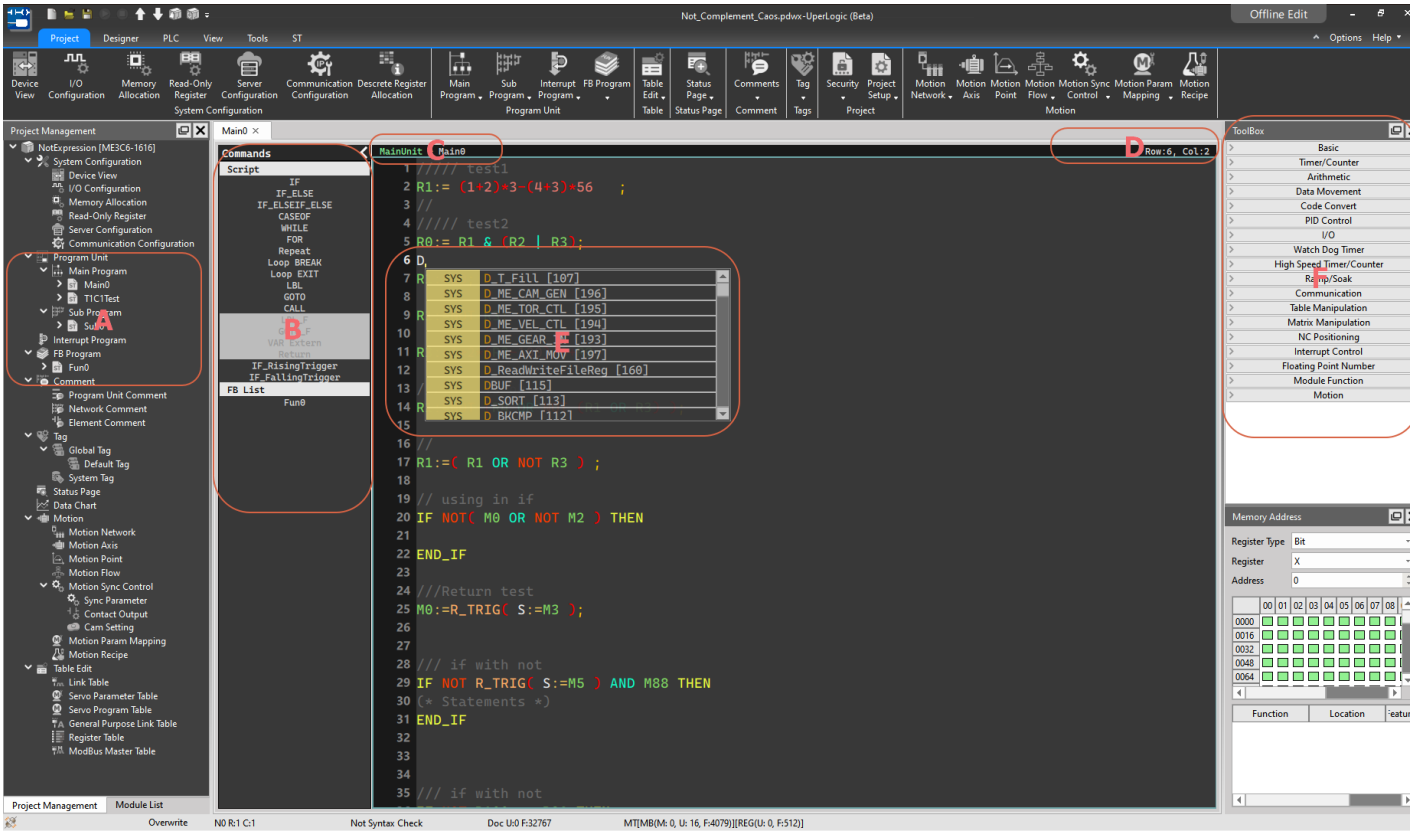
選取想要處理的中斷訊號,以及程式類別 (ST)



2

Uperlogic ST 使用者介面

2-1 介面概觀



A:　雙擊程式名稱會跳出編輯畫面

B:　1.　顯示目前可以使用的 ST 語法,以及可呼叫使用的 FCM 程式，雙擊該欄位，會插入對應的樣板在程式裡頭

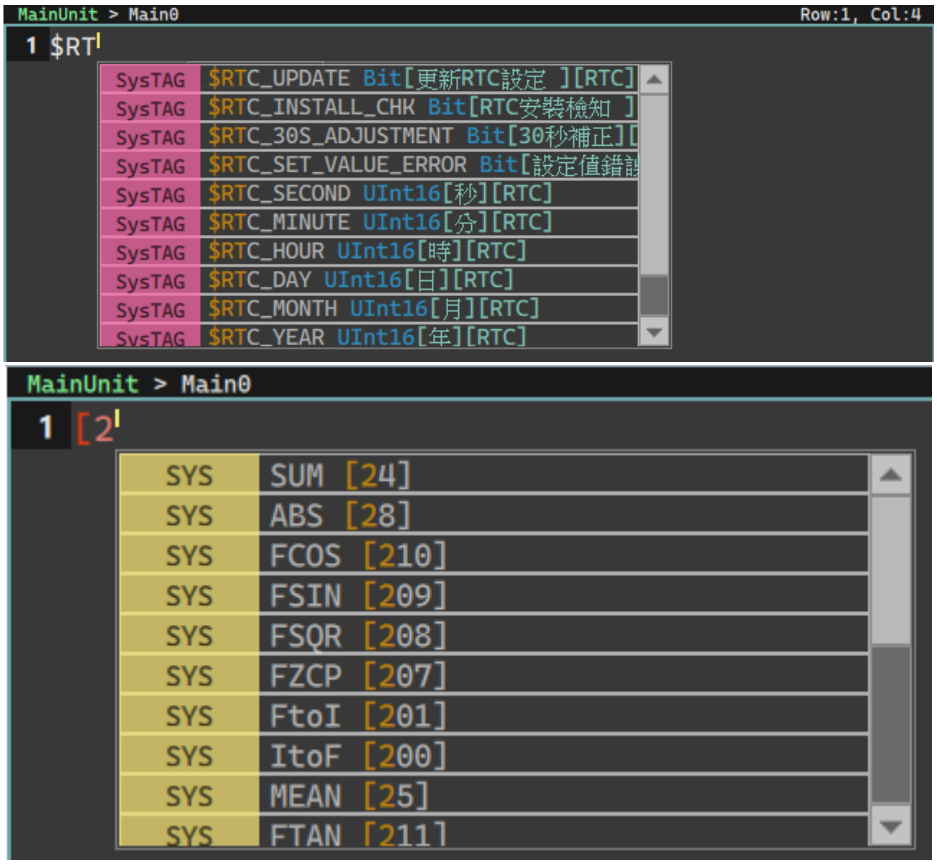
2.　顯示的如果呈現反灰，代表該指令不能用在當前的文本中

3.　右側邊界可以拖動調整寬度，
也可以點擊右上角的按鈕 ❮縮到最小或是還原 ❯

C:　目前 ST 的類別跟名稱　`MainUnit > Main0`

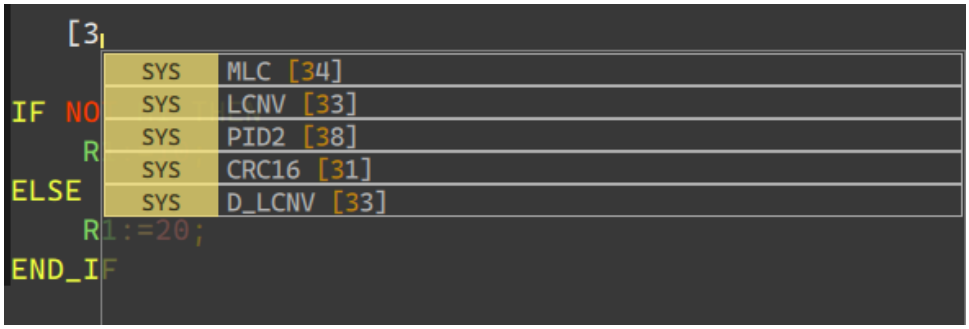
D:　目前游標在的行列資訊　`Row:17, Col:7`

E:　智能彈跳提示視窗，方便使用者在輸入時便捷提示



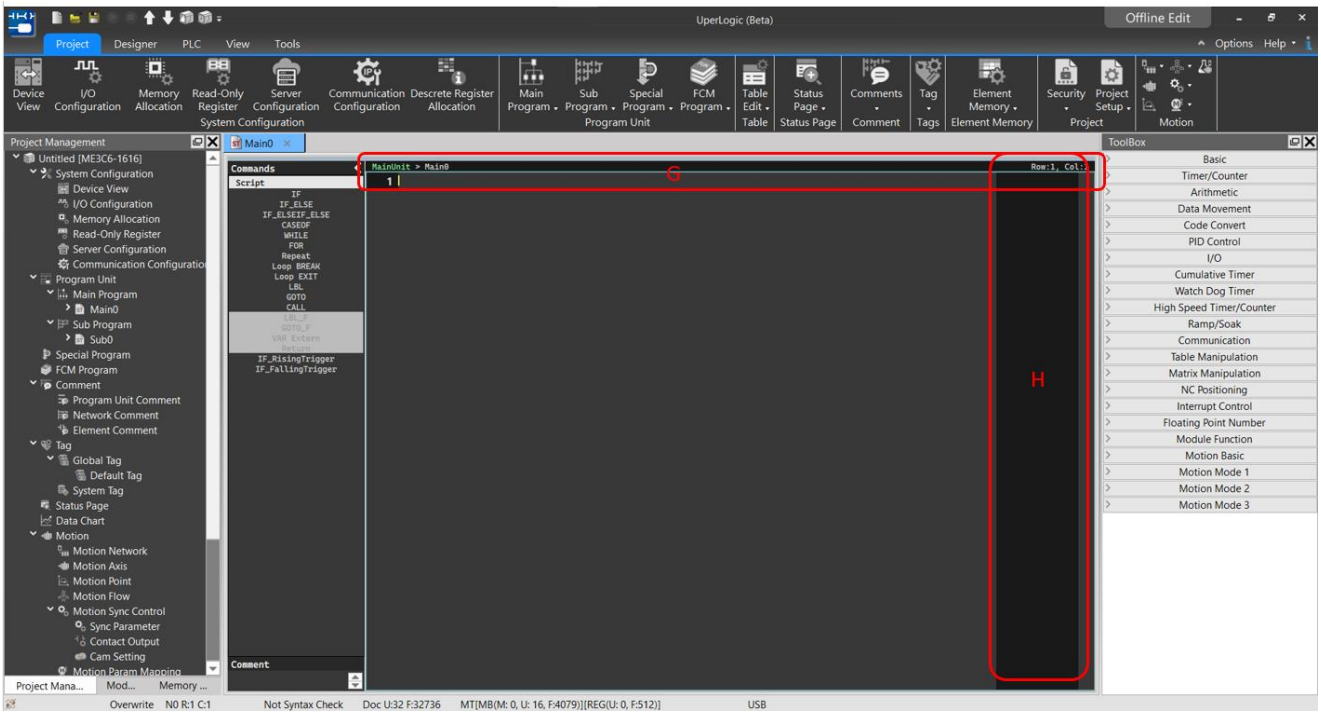
這個彈跳視窗會隨著使用者輸入，(比對 3 個字元以上或[+數字)即時的顯示自動偵測的提示程式片段，包含可使用的標籤，變數，呼叫的函式等等。其中有符合的字串會用橘色顯示在視窗內(如上圖)。

由於不只名稱會搜尋，也會依同搜尋後面的註解(顯示優先權較低)，因此除了直接搜尋該函示名稱之外，也可以輸入註解相關文字，如函式 ID (如下圖)

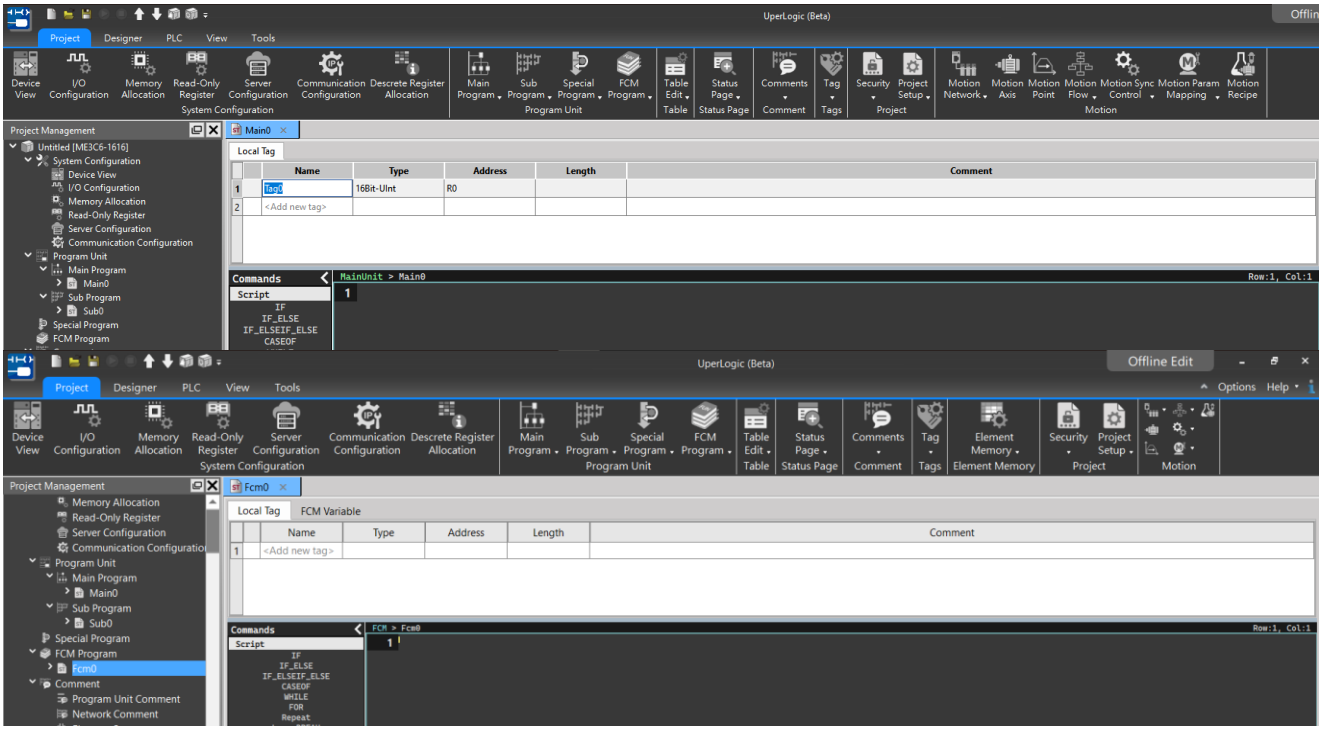


這樣便可以使用滑鼠選擇雙擊，或是上下鍵選取想要插入的程式片段。

F: 可呼叫系統指令的工具箱,滑鼠雙擊，或是拖曳到文字編輯器中



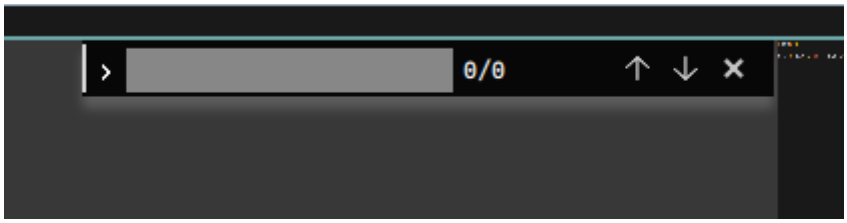
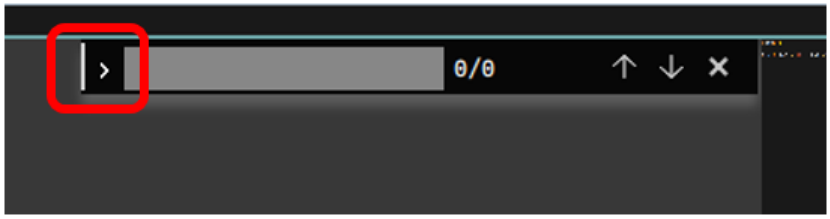
G:　拖曳此區可顯示區域標籤，在編寫 ST/FCM 程式時對照標籤修改程式將更方便



此區支援 ST/FCM 線上編輯模式，可供程式除錯修正使用

H:　鳥瞰圖滾輪視窗，方便簡略查看程式概況

2-2 支援的鍵盤指令

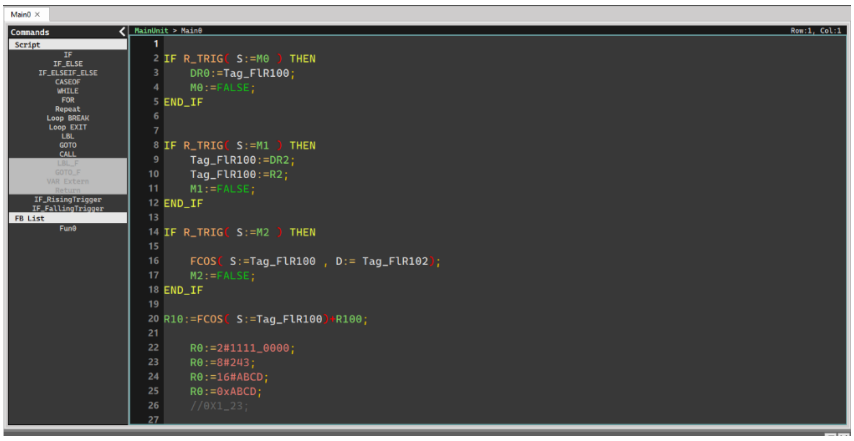
鍵盤	功能
Ctrl + C	複製
Ctrl + V	貼上
Ctrl + A	全選
Ctrl + X	剪下
Tab	插入 tab
選取多行+Tab	多行同時加 Tab
Shift + Tab	根據游標或是選取的行數同時 Back Tab
Ctrl　+ F	出現搜尋視窗在畫面右上方 可以搜尋目前文本文字  按下”＞”按鈕可以使用文本取代功能  取代功能分別為”單項取代”和”全部取代” (只能取代當前程式單元) 
Ctrl + /	根據所選取的行數,批次進行註解/反註解
Ctrl + ‘+’	放大整個 ST 區域字體
Ctrl + ‘-’	縮小整個 ST 區域字體

2-3 系統模式

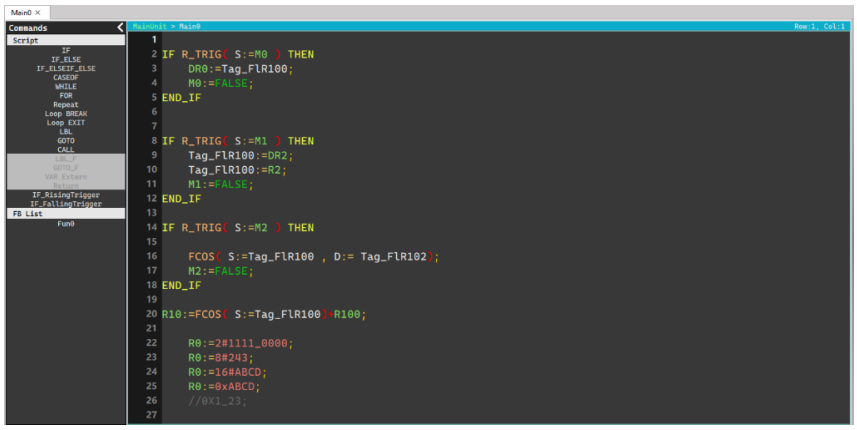
目前系統軟體有三種模式

ST 編輯器也對應會呈現三種狀態

1. 離線編輯(ST 編輯視窗上方顯示黑色底)



2. 線上監測 (唯讀 不可編輯) (ST 編輯視窗上方顯示青色底)

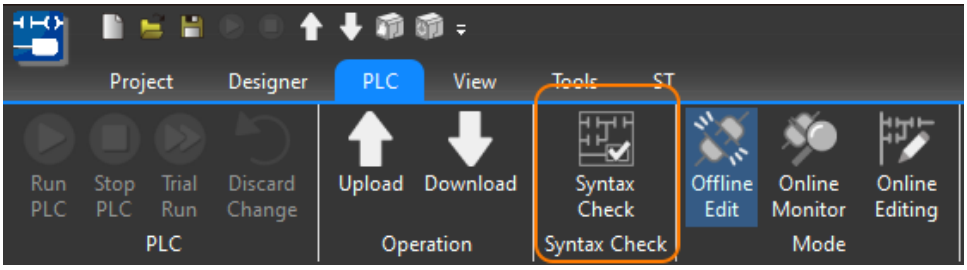


3. 線上編輯(ST 編輯視窗上方顯示橘色底)

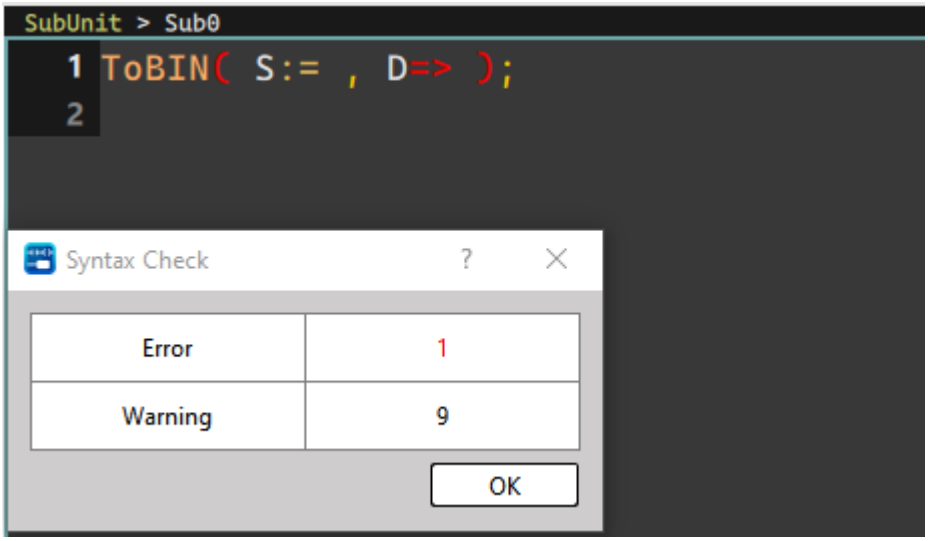
2-4 編譯文本 / 語法檢查

ST 的文件都必須要經由編譯語法的檢查轉換成 PLC 可以運行的程序

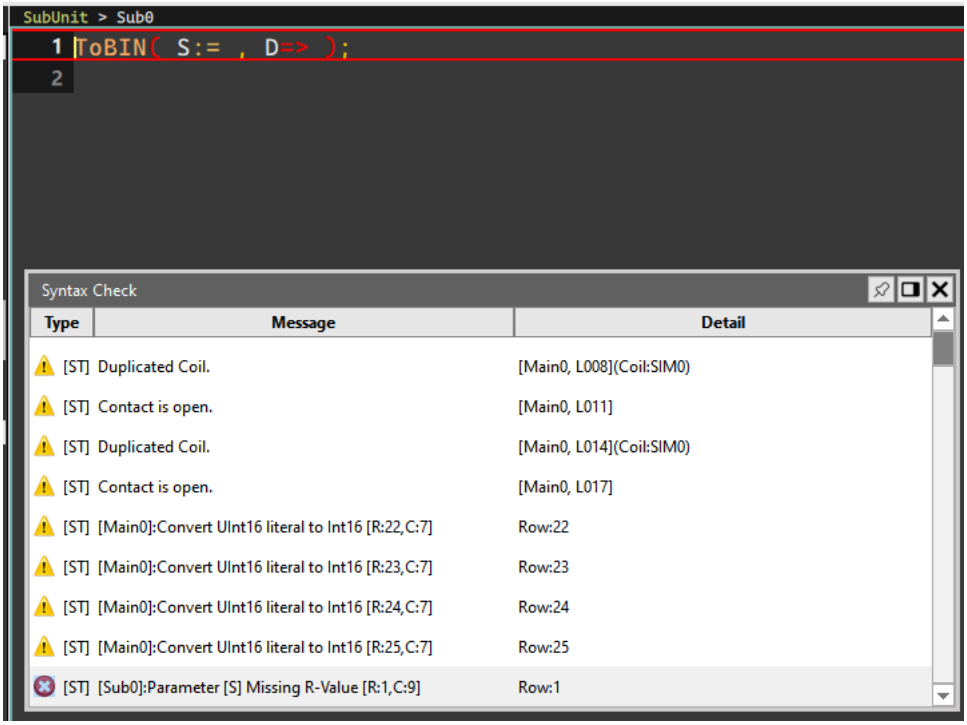
平時在撰寫的時候可以點擊下方圖示的按鈕進行編譯，確認語法是否正確



假如有錯誤或是有警告提示，會有視窗彈跳提示



細部內容則會顯示在如下圖的視窗內，雙擊該欄位後編輯游標會跳移至錯誤行



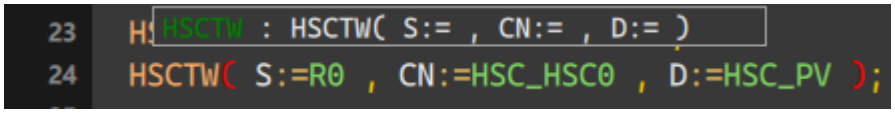
每次系統下載或試運行時都會自動進行編譯以確保語法正確，當檢測有”錯誤”時將會無法完成指示動作。

2-5 滑鼠懸浮提示

為了除錯以及撰寫程式方便，支援滑鼠在變數或是函式文字上面懸浮停留會出現提示視窗的介面，給予使用者撰寫程式或除錯的提示

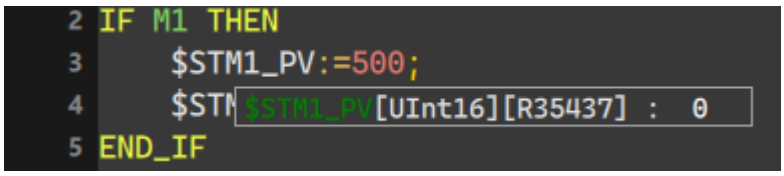
A. 一般函式

會顯示該函示的完整呼叫參數 如:



B. 變數

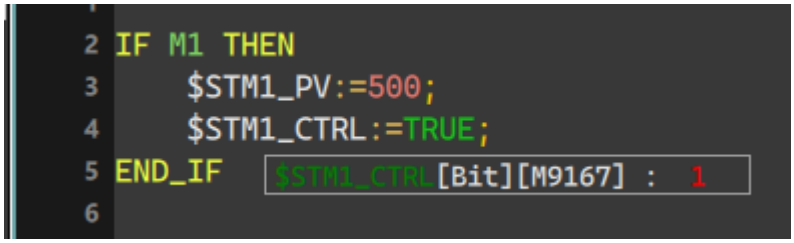
會顯示該變數的型態跟數值 如:



後面顯示的數值，可以直接用滑鼠點擊後面的數值進行修改

Bool 型別點擊 On/Off 切換

數字型別: 則是直接輸入對應的數字修改



!注意!

要得到上述 B.變數功能的提示，該程式文本需要語法檢查成功過後才能取得正確提示。

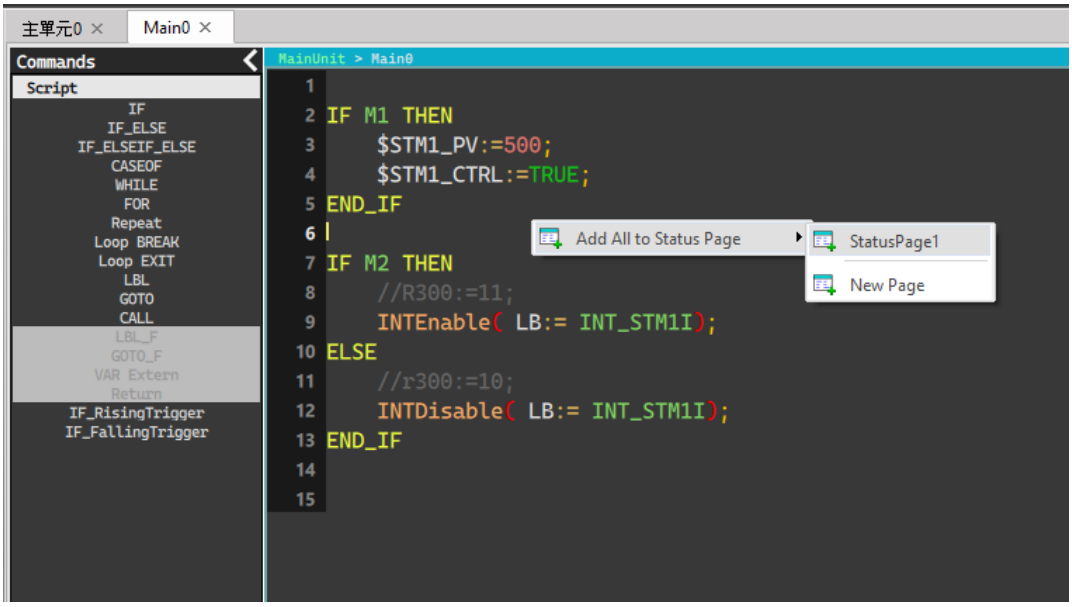
2-6 加入觀測變數

在線上編輯跟線上監測模式中，在文字編輯器上面按滑鼠右鍵會跳出選單可以將滑鼠游標對準的變數，或是目前文本所有變數自動加到狀態觀測的頁面中。方便設計者監視當前值並進行除錯。

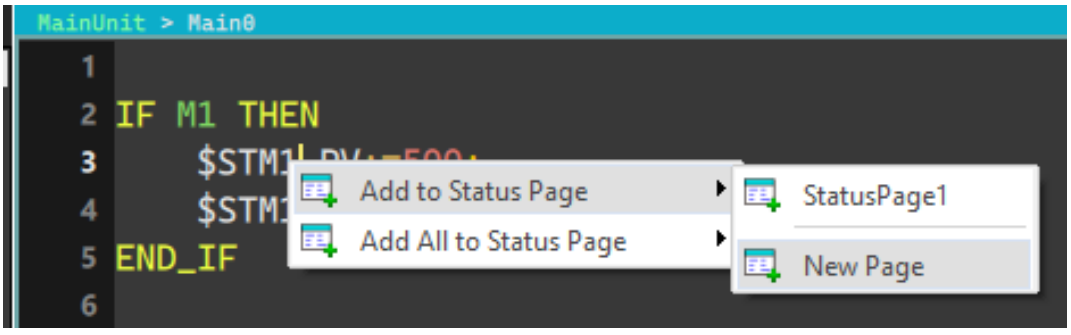
如下圖:

在空白區域點擊滑鼠右鍵會跳出自動加入所有變數到監視頁中的選項，

並且可以在子選單選擇加入到哪一頁，或是創建新的一頁



在變數上面按滑鼠右鍵則會多出新增當前變數到監視頁面中

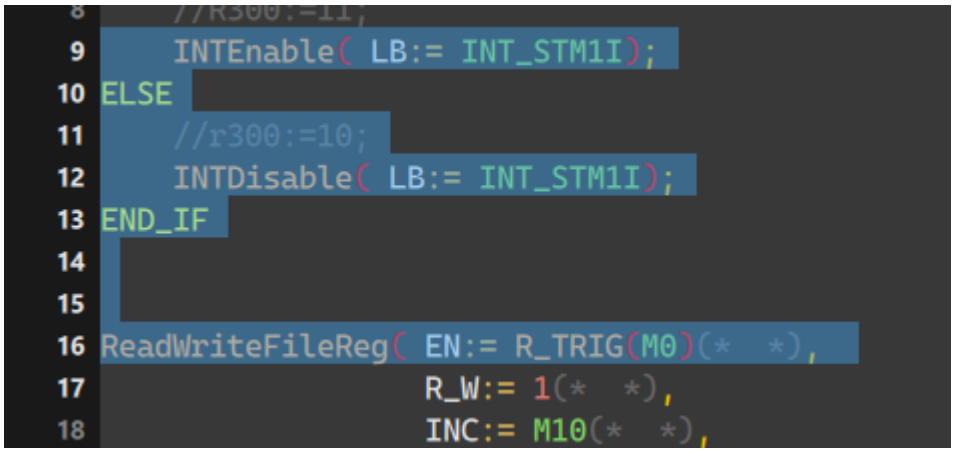


2-7 快捷滑鼠鍵盤操作

ST 文字編輯提供許多提升程式編輯便捷性之常見功能操作

A. 選取整行

滑鼠點選行數顯示的地方會選取該行，並且可以往上或往下連續選取整行



B. 批次加 Tab / 移除 Tab

單行起始加 Tab:

游標移到該行的起始按 Tab 按鍵

多行批次加 Tab:

選取多行程式碼後再按 Tab，則所有行都會統一加一個 Tab 在開頭

單/多 行移除 Tab:

游標選取該行任意位置，或是選取多行，再按 Shift + Tab 則會統一往左縮排(Back-Tab)

C. 註解反註解

多行註解除了可以使用 "(*" , "*)" 這兩種符號來作之外，支援 Ctrl+/ 來針對游標所在的行，或是目前所選取的行來批次新增 // 註解在每一行文字開頭。

新增邏輯:

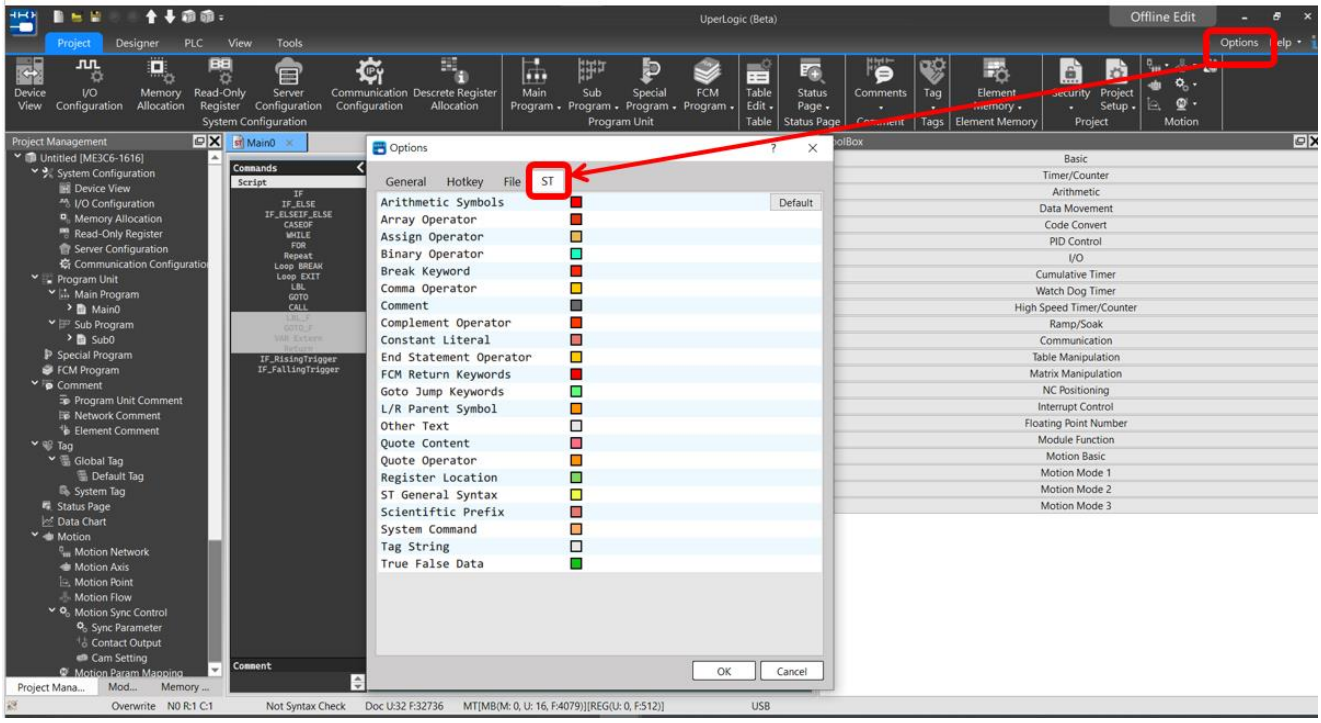
只要選取行數範圍內有一行開頭沒有// 符號，則所有行數統一新增//註解

取消邏輯:

所選取的行數每一行開頭都要有//符號，就會運行批次移除註解

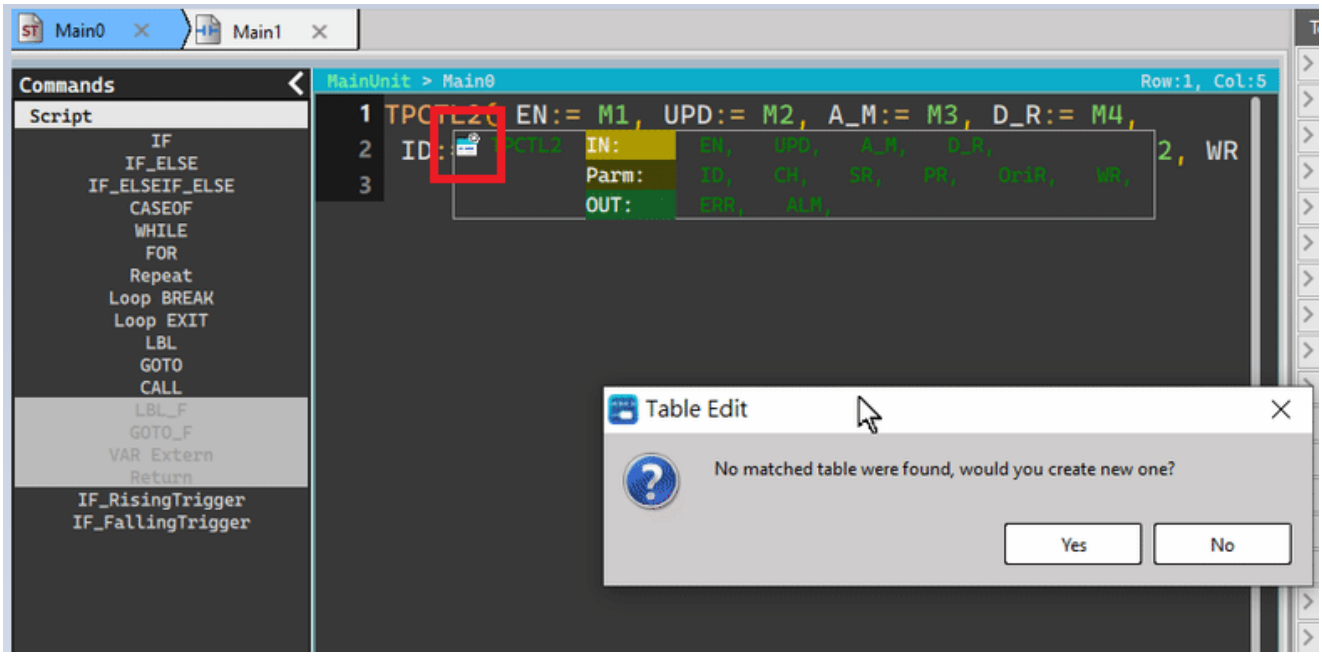
2-8 結構化語言顏色調整

配合使用者習慣，可開放調整 ST 編輯環境字體顏色



2-9 快速呼叫/建立表格

部分指令可點擊彈跳視窗的左上角的按鈕，可快速呼叫/建立對應的表格



支援列表	指令 ID
PID2	38
ASCWR	94
TPCTL2	99
HSPSO	140
MPARA	141
MHSPO	147
AHSPO	149
ModBUS	150
CLINK	151
NCR	152
NSSEND	154
NSRCV	155

3

Uperlogic ST 的程式基本架構

3-1 簡介

本章對使用基本 ST 語言編寫程式的方法進行說明。

3-1-1 字元編碼

ST 編輯器支持 Unicode(encode in UTF-8)多國語言。支援程式編輯中出現的日語、英語、中文等多種語言的基本字元及大部分符號，除了可以用于註解外，還可以用于標籤或程式、表格名中。

3-1-2 構成單位

ST 語言使用以下記號的組合敘述程式。後面的章節會在細部解說

類型		範例	參考
運算符號		+、-、*、/、AND、&、OR、|、XOR、MOD、 <=、 >=、<>、++、--、<<、>>、NOT、:=、(、)	3-4 運算元（Operand）與運算子（Operator）
控制語法的關鍵字 (定義的標準識別符號)		IF、ELSEIF、END_IF CASE、OF、END_CASE WHILE、FOR、END_WHILE END_FOR LBL GOTO、CALL、 LBL_F、GOTO_F IF_RISINGTRIGGER IF_FALLINGTRIGGER	3-6 流程控制與迴圈
識別符號	變數, 硬體暫存器,IO 點位,間接定址....etc (標籤、元件等)	X0、Y0、M100、R0、DR0、D0、DD0、IM0...etc Tag(註冊任意的名稱)	

	函式呼叫(FCM)	1.系統內建 Function 2.FCM 函式庫	3-14 呼叫 FCM 函數
常數		整數: 預設為 signed int Ex: **R0**:=1; **R0**:=-2; Unsigned int Ex: **Tag0**:=0xFF; 字串: 只能使用在 Label 上 Goto, LBL...etc Bit (Bool) Type: TRUE、FALSE Float: 32 位元浮點常數 ex: TagFloat1:=1.1;	3-7-4 常數
分隔符號		分號『:』 **出現在 CASE OF 中** 分隔『,』 多個函式參數傳遞使用	3-6-3 Case Of
左右小括弧		『(』 ,『)』 1. 函式呼叫 的參數開始結尾 ex: Fun0 (S:=R0, D:=R1); 2. 強制普通運算子的優先順序 R0:= (1+R0)*R2;	3-4-1 ( )
RETURN		可以立即離開該程式區段	3-6-6 Label & Jump/Call
『;』 描述句結尾		用於標明一段程式敘述句的結束	-

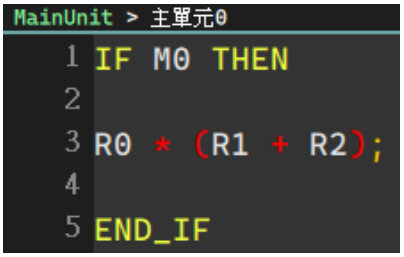
各記號之間可以自由插入空白、換行及注釋。

類型	範例	參考
空白	空格(全形/半形)、TAB	-
換行	換行代碼	-
註解	//、(* .... *)	

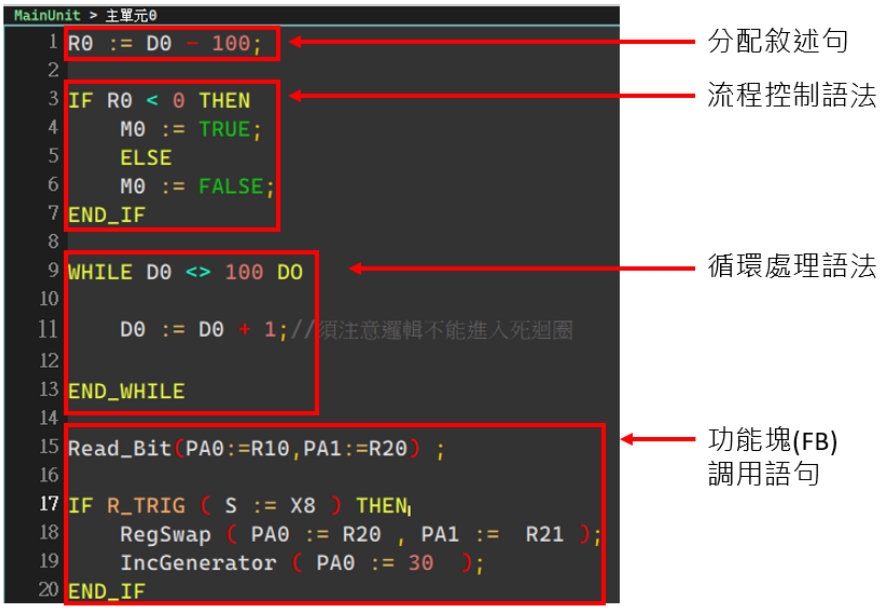
3-2 敘述句

敘述句是 ST 語言中最基本的執行單位，它代表的是一個可被完整執行的工作。一個完整敘述句的呈現可能並不侷限在同一行文字當中，但它必定是以分號『；』來做為該敘述句的結尾。另外，一個敘述句中也允許再包含多個或多層的子敘述句，且不論其敘述句所處的位置為何，只要是一個敘述句的結尾，最後必定會跟隨著『；』符號。

一個完整的敘述句相當於一個具備完整功能的階梯圖區段（NETWORK），它必須能夠明確的表達一件工作。以上述程式中的敘述句為例，它所執行的工作便是將各裝置的內容值依照數學式表達的順序 R0*（R1+R2） 來進行運算，並將運算的結果指定給裝置 R10；但如下圖紅色框線中的內容雖然合法，但不是一個完整的敘述句，它只是一個表達式（Expression），代表的只是一個數學運算的數值，卻不是一件具體的工作。

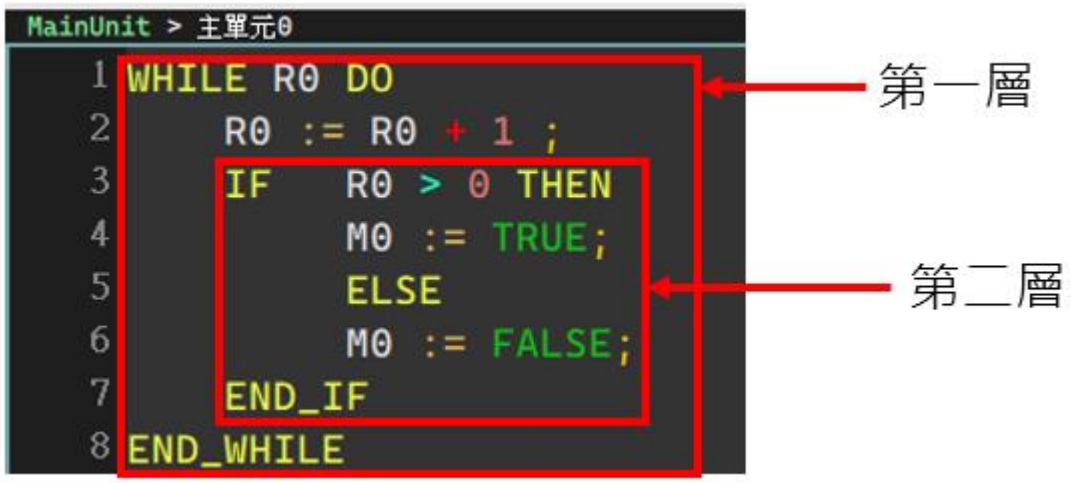


以下將提供幾個 Uperlogic 結構化語言的敘述句:



類型		內容	範例
分配敘述句		將右邊的結果代入左邊的變數	:=
流程控制語法	選擇敘述句(IF、CASE)	根據條件選擇執行的語法	3-6-1
	循環處理敘述句(FOR、WHILE)	根據結束條件，多次執行	3-6-2 3-6-4
子程式語句	FCM 調用語句	調用 FB(FCM)	3-14
	標籤語句	調用 LBL	3-6-6
工具箱語句		將工具箱指令調用的方便語句	

流程控制語法可分層。(可以搭配選擇語句、循環處理語句混合多層使用)



3-3 表達式（Expression）

表達式在敘述句結構當中是一個相當重要的組成元素，它代表的是一個「值」，例如 TRUE 或 FALSE 的布林值，亦或是 20 或 -5 的整數值；而表達式視使用的場合，可以是一個運算式或是一個常數， 當然也可以是一個變數符號或裝置。下列便是一些表達式的例子。

■ M0 & M1（布林值的表達式） 此表達式所代表的是 M0 與 M1 進行 & 運算後的布林值。

■ M0 = FALSE（布林值的表達式） 此表達式所代表的是「M0 = FALSE」這個條件是否成立。

■ M0 （布林值的表達式） 此表達式則直接以 M0 的值當作其代表的布林值。

■ D1 + D2（數值的表達式） 此表達式代表的數值為 D1 與 D2 相加後的運算結果。

■ D0（數值的表達式） 此表達式則直接以 D0 的現在值做為其代表的數值。

■ D2 = D0 + D1（布林值的表達式） 它代表的是「D2=D0+D1」這個條件是否成立。也就是說當 D0 與 D1 相加的結果等於 D2 的現在值時，表達式代表的便是布林值 TRUE；相反的，如果 D0 與 D1 相加的結果不等於 D2 的現在值時，表達式所代表的即是布林值 FALSE。

■ D2 := D0 + D1；（敘述句，非表達式。） 此為一個完整的敘述句而非表達式，其代表的意義是「將 D0 與 D1 的相加結果指定給 D2」的工作；不過這個敘述句也是由「D2」「D0+D1」這兩個表達式所組成的。

表達式的使用位置

下表所示為表達式的類型

類型		表達式(運算結果)的資料類型	範例
運算表達式	算術表達式	整數、實數等(依據運算物件)	R0+R2
	邏輯表達式	布林值(TRUE/FALSE)	R0 AND R2
	比較表達式	布林值(TRUE/FALSE)	R0 > R2
基本表達式	變數、常數	定義的資料類型	M0.R0.123.TRUE
	函數調用表達式	返回值的資料類型	Fcm0(PA0:= ,OUT0=>);

3-3-1 運算表達式

本節以範例說明，運算表達式在 ST 環境和 LD 環境的呈現方式

3-3-2 四則運算(+、-、*、/)

四則運算使用與一般的算術符號相同的運算符號(+、-、*、/)敘述。

對于使用 LD 圖無法一次性敘述完整的運算，可以通過單行表達式簡潔地敘述出来。

程式範例

在 R3 中代入 R0~R2 的和。

R3=R0+R1+R2

ST

R3:=R0+R1+R2;

LD

■用一個敘述句加入多個運算表達式時，將從優先級最高的運算符號開始處理。

四則運算符號的優先級請參考運算元章節，有多個優先級相同的運算符號時，從最左邊的運算符號開始運算。

高級運算(指數)

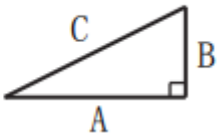
指數運算或三角函數運算使用通用函數。

類型		函數名	範例	
			一般算式表述	ST
絕對值		ABS	|X|	ABS(D:=);
平方根		SQRT	$\sqrt{X}$	FSQR(S:= ,D:=);
三角函數	正弦、反正弦	SIN、ASIN	B=SIN A	FSIN(S:= , D:=);
	餘弦、反餘弦	COS、ACOS	B=COS A	FCOS(S:= , D:=);
	正切、反正切	TAN、ATAN	B=TAN A	FTAN(S:= , D:=);

程式範例

求直角三角形的斜邊長

$$C = \sqrt{(A^2 + B^2)}$$



ST

(使用浮點數需先註冊 Tag 改變資料型別

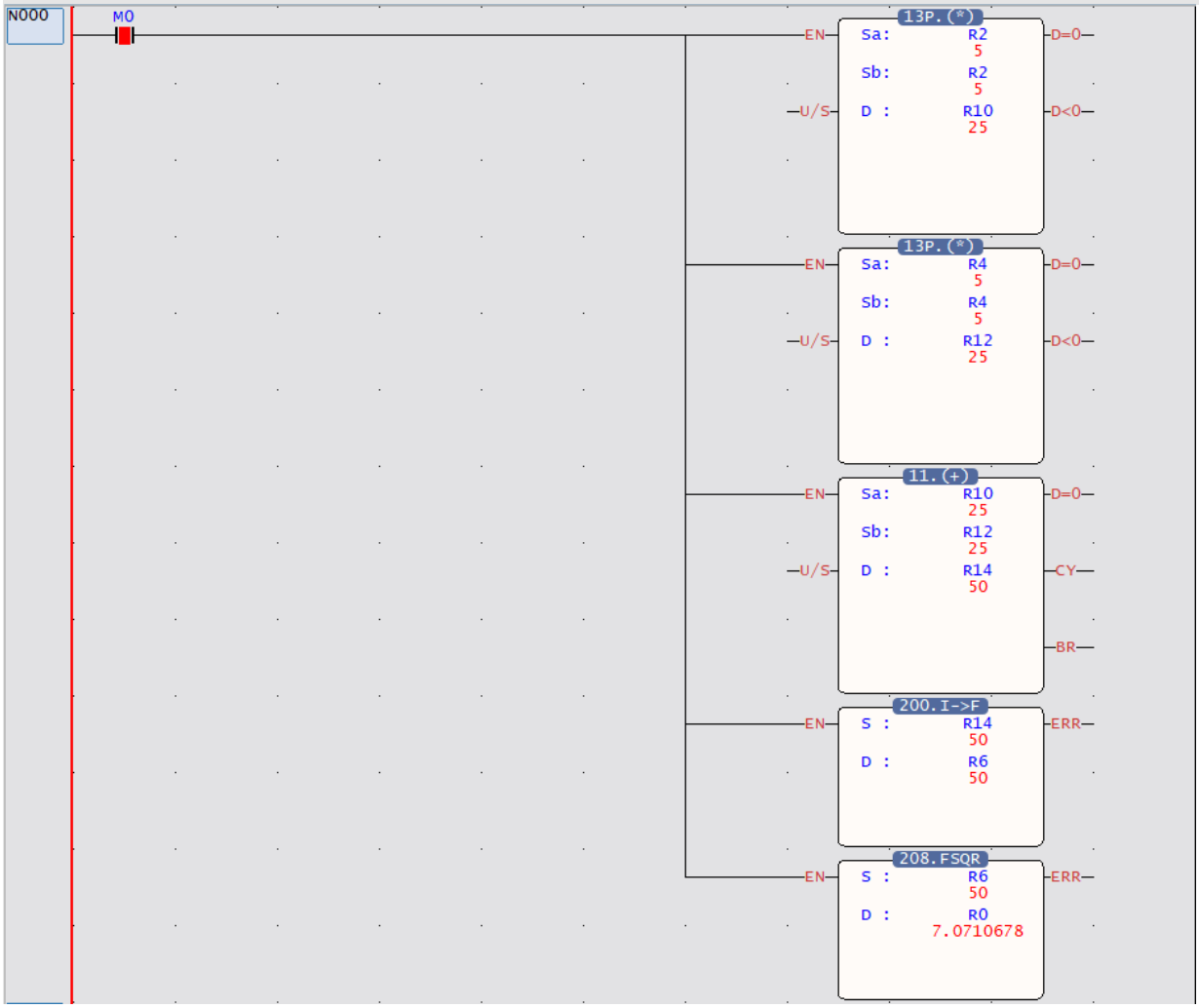
ex: A= Float DR0、B= Float DR2、C= Float DR4)

FSQR(S:= A*A+B*B, D:= C);

區域標籤

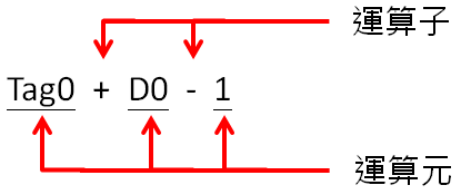
		名稱	類型	位址	長度
1		A	Float	DR0	1
2		B	Float	DR2	1
3		C	Float	DR4	1

LD


N000
M0
13P.(*)
EN
Sa: R2 5
Sb: R2 5
U/S
D : R10 25
D=0
D<0
13P.(*)
EN
Sa: R4 5
Sb: R4 5
U/S
D : R12 25
D=0
D<0
11.(+)
EN
Sa: R10 25
Sb: R12 25
U/S
D : R14 50
D=0
CY
BR
200.I->F
EN
S : R14 50
D : R6 50
ERR
208.FSQR
EN
S : R6 50
D : R0 7.0710678
ERR

3-4 運算元（Operand）與運算子（Operator）

運算元與運算子為組成一個表達式的基本元素。運算元指的是參與運算的對象，而運算子則代表所執行的運算動作，例如表達式 D0 + D1 中，D0 與 D1 都是運算元，而 + 號則是運算子。 由上一節的各個例子可知，表達式可為一群運算元與運算子的組合，但也可單獨的以一個運算元來呈現一個表達式，而其中做為運算元的可為裝置、變數符號或是常數。



如同數學式一般，運算子本身也有執行運算的優先順序，而當優先的等級相同時，進行運算的順序便會是由左至右。下表即為 Uperlogic 中 ST 語法的運算子一覽表。

符號	功能	資料格式		表達式範例		優先等級
		運算元	運算結果 (表達式的值)	表達式	值	最高
( )	區塊優先	不限	不限	(D0 + 6) * 3		
++,--	快速加,減 1	任何數值	任何數值	++D0　　D0++		
-	數值負號	任何數值	任何數值	-D0		
NOT	邏輯反向	布林	布林	NOT M0	TRUE	
*	乘法	任何數值	任何數值	D0*3		
/	除法	任何數值	任何數值	15/D0		
+,-	加法,減法	任何數值	任何數值	D0+3		
< , >,<=, >=	數值比較	任何數值	布林	D0>2		
=,<>	等於,不等於	任何數值	布林	D0<>2		
		布林		M0=TRUE		
AND,&	"及" 運算	布林 or 數值	布林 or 數值	M0&M1		
OR, |	"或" 運算	布林 or 數值	布林 or 數值	M0 OR M1		
XOR XNOR	"互斥或" 運算 除法取整數	布林 or 數值	布林 or 數值 整數	M0 XOR M1		
MOD		數值	數值	D0 MOD 3		
>>,<<	右移 1,左移 1	任何數值	任何數值	D0>>1	2	
R_TRIG, F_TRIG	上沿 下沿					最低

■有底色的部分同區段底色的會是同樣的優先度

■AND OR　XOR XNOR 基本上會做 BitWise 的作用(ex R0 AND R1) 結果就會是數值

除非左右兩邊都是 Bool(bit) (ex. M0 AND M1) 的型別就會做邏輯的運算,結果就會是 Bool(bit) 值

■沒有支援&&、||、==語法

各小節針對個別運算子做介紹

3-4-1 ( )

作用跟數學式子一樣，針對括弧內的表達式來優先進行運算

EX:
R0:= (R0+2)*3-(R2+3)*56 ; // R0=1;R2=2
// R0= - 271

3-4-2 四則運算 +, -,* , /

針對兩邊的運算元進行加減乘除的計算

EX:
R0:= 5+4-3*2/1 ;
// R0=3

3-4-3 遞增/遞減 ++, --

單獨對運算元 + 1 或是 -1

前綴語法:

++(Register)
--(Register)

效果等同於 (Register):=(Register)+1;

但是比較簡短整潔

後綴語法:

(Register)++
(Register)--

會先取出(Register)數值,並且 return ,之後再運行(Register):=(Register)+1 的行為

ex:

R0:=10;
R2:=10;
R100:=++R0; // R100 為 11, R0 為 11
R102:=R2++; // R102 為 10, R2 為 11

3-4-4 數值比較 >=,<=,>,<

會比較運算子左右兩側的運算元大小

運算子左右兩側需要相同型別才可以比較大小

否則得到的比較結果可能會有誤 (float <->int)

EX:

M0:=R1 >= R2; //如果 R1 大於等於 R2 的話則 M0 為 TRUE (1)

3-4-5 相等或不等 = ,<>

比較左右兩個運算元是否相等

得到的結果則為 TRUE/ FALSE (1/ 0)

EX:

M0 := R0=100; //假如 R0 的值為 100 的話 M0 為 TRUE (1)

3-4-6 位元左右平移 <<, >>

『>>』,『<<』

針對左側運算元 左/右偏移右側的運算元

EX: R10:=R0<<4; //R0=1(0b0_0001)

// R0 的數值會被往左偏移 4 個 bit, 並且分配到 R10

// R10=16(0b1_0000)

P.S. R0 並不會改變

3-4-7 反相運算元 NOT

針對右邊的運算元依位元反相

EX:

R0:=NOT R2; // 對 R2 的數值做反相存到 R0

// R2=1(0b0_0001) => NOT R2= -2(0b1111_1110)

M0:= NOT (M0 OR M2); // M0 跟 M2 做完或閘之後 再反相 存到 M0

3-4-8 負號運算元 '-'

『-』

針對右邊的運算元變為負數

EX:

```
IF R_TRIG( S:=M1 ) THEN
R0:=-R2;     //R2=1 (0b0_0000_0000_0001)=> R0=-1(0b1111_1111_1111_1111)
END_IF
```

3-4-9 賦予數值 :=

將運算子的右邊運算元(變數),指派到左邊的變數

EX:

```
R1:=R0;   //   R1=R0
R1:=1;    //   R1=1
```

3-4-10 邏輯運算元 AND (&)、OR (|)、XOR、XNOR

對運算子左右兩邊的運算元進行邏輯運算

運算結果型別會跟運算元的左邊第一位型別是一樣的

EX:

```
R0:=   R1 AND R2;    //   對 R1, R2 的數值進行位元 及 運算存到 R0
 //   假如 R1=1(0b0_0001)   R2=15(0b0_1111)   則 R0 =1(0b0_0001)
DR0:=   R10 & DR12;    //   對 R10, DR12 的數值進行位元 及 運算存到 DR0
          //   假如 R10=1118481(0x11_1111H)   DR12=69905(0x1_1111H)
          //   則 DR0 =4369(0x1111H)
R0:=   R1 | R2;    //   對 R1, R2 的數值進行位元 或 運算存到 R0
          //   假如 R1=1(0b0_0001)   R2=15(0b0_1111)   則 R0 =15(0b0_1111)
DR0:=   DR10 OR R12;    //   對 R10, DR12 的數值進行位元 或 運算存到 DR0
          // 假如 DR10=4369(0x1111H)   R12=69905(0x1_1111H)
          // 則 DR0 =4369(0x1111H)
R0:=   R1 XOR R2;   //   對 R1, R2 的數值進行位元 反或 運算存到 R0
          //   假如 R1=1(0b0_0001)   R2=15(0b0_1111)   則 R0 =14(0b0_1110)
R0:=   R1 XNOR R2;    //   對 R1, R2 的數值進行位元 反互斥或 運算存到 R0
        //   假如 R1=1(0b0_0001)   R2=15(0b0_1111)   則 R0 =-
15(0b1111_0001)
```

3-4-11 MOD 餘數

計算左右運算元的餘數 (等同於 C 語言的 %符號)

EX:

```
R0:= R1 MOD R2 ;   // R0 為 R1 除以 R2 之後的餘數
//   R1=10,R2=3   10/3 商數等於 3 餘數等於 1,R0=1
```

3-5 註解（Comment）

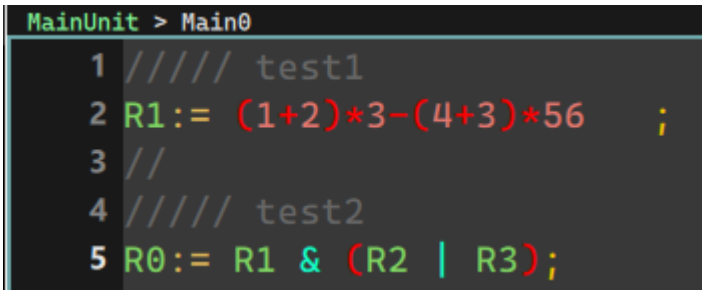
註解是用來讓程式開發者日後容易維護的註解

有分為單行或是連續註解

會呈現淺灰色字樣，這些部分會被編譯器忽略不會產生運行資料

3-5-1 單行註解　//

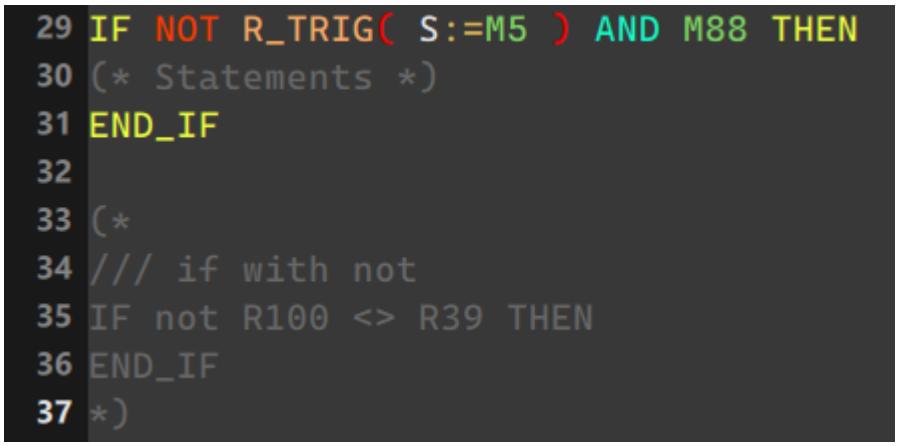
這個符號之後會呈現淺灰,並且不會產生運行的程式碼



3-5-2 多行註解 (*　　*)

可以跨多行(單行也行) 的連續註解

在這兩個符號中間的文字會被當成註解



P.S. 在操作上也可以透過快捷鍵 (Ctrl+ /) 來針對所選取的行列

批次 (加 / 移除) 註解

3-6 流程控制與迴圈

撰寫 ST 的時候通常都會需要一些條件或是迴圈控制便於設計，參考以下介紹。

3-6-1 IF ELSE

```
IF (* bool exp *) THEN
(* Statements *)
ELSE
(* Statements *)
END_IF
```

當 IF 後面的表達式最後結果為 TRUE 或是 1 的時候

會緊接著運行 IF 後的描述

否則會運行 ELSE 後的描述

EX:

```
IF R_TRIG( S:=M5 ) AND M88 THEN
    R100:=10;
    R10:=12;
END_IF
```

假如 M5 上沿訊號觸發，並且 M88 也為 TRUE (1) 的時候

R100 為 10，R10 為 12

如果有多個條件則可以用 ELSEIF

EX:

```
IF (* bool exp *) THEN
(* Statements *)
ELSEIF (* bool exp *) THEN
(* Statements *)
ELSE
(* Statements *)
END_IF
```

3-6-2 For Loop

```
FOR (*Reg*) := (*Start*) TO (*End*) BY  (*<inc>*) DO
(* Statements *)
END_FOR
```

參數：

(* Reg *)：計數暫存器

重複運行 FOR 到 END_FOR 裏頭的指令，直到計數暫存器達到達到目標值

每重複一次運行就會對計數暫存器值加上遞增值或遞減值

(* Start *)：計數初始值

(* End *)：計數目標值

(* < inc > *)：遞增值或遞減值

(* Statements *)：欲迴圈執行之程式

P.S. BY 遞增值這個欄位可以省略 (預設為 1) 如下圖

```
FOR (*Reg*) := (*Start*) TO (*End*)  DO
(* Statements *)
END_FOR
```

	計數暫存器	計數初始值	計數目標值	遞增值或遞減值
16 位元暫存器	○	○	○	
32 位元暫存器	○			
常數		○	○	○

P.S. 若迴圈計數為遞增，則 計數初始值 < 計數目標值；

若迴圈計數為遞減，則 計數初始值 > 計數目標值。

EX:

```
FOR DR0 := R2 TO R4 BY 2 DO //DR0  會從 R2, R2+2, R2+4,....etc 直到 R4
    R10++;                                  // DR0 = R4, R10 = R10 + (R4-R2) / 2
END_FOR
```

3-6-3 Case Of

整數條件選擇

```
CASE (* Integer target *) OF
(* int literal *) : (* single statement *)
ELSE
(* Statements *)
END_CASE
```

會讀取對目標暫存器的數值，滿足指定的條件後運行 : 後面的敘述句

假如都不滿足，則會運行 ELSE 後面的敘述句

條件只支援整數常數 或是 範圍 .. 符號內的常數

EX:

```
CASE R0 OF
    1:R100:=10;
    3..9:R100:=11;
END_CASE
```

R0 等於 1 // R100 等於 10

//3, 4, 5, 6, 7, 8, 9 ,R100 等於 11

3-6-4 While Loop & Repeat

重複運行描述 直到條件(不)滿足

While Loop:

```
WHILE (* bool exp *) DO
(* Statements *)
END_WHILE
```

EX:

```
WHILE R0<10 DO
    ++R0;
END_WHILE
```

Repeat:

```
REPEAT
(* Statements *)
UNTIL (* Stop Condition *)
END_REPEAT
```

EX:

```
REPEAT
    ++R0;
UNTIL R0=10
END_REPEAT
```

Notice:

因為硬體運行邏輯的關係，While 如果持續太久時間沒有跳開迴圈，會使 PLC 設備無法處理其他 IO 狀態，造成系統錯誤，在使用的時候要注意

3-6-5 Break / Exit 跳開迴圈

在 FOR , WHILE ， REPEAT 的迴圈狀況下，可以在適當時間搭配 break 或是 exit 提前離開迴圈。

EX:

```
WHILE M0 DO
    ++R0;
IF R0>=100 THEN
    M0:=FALSE;
    BREAK;
    END_IF

END_WHILE
```

當 R0 大於等於 100 的時候 會運行 break 跳開 while loop

Notice:

迴圈系列如 For While Repeat，有可能會因為迴圈運行次數過久，會導致 PLC 在運行的時候產生 Error，使用上請特別注意，可以搭配中斷訊號處理迴圈處理過久的問題。

3-6-6 Label & Jump/Call

LBL 指令能達到 LD FUN_65 相同效果(詳細說明可以參考 FUN 66)，最多可輸入 6 個 ASCII 字元當作標籤名稱。

EX:

```
3  LBL ( "R65" )
4      R65 := R65 + 1;
5  LBL ( "123456" )
6      R66 := R66 + 1;
7  LBL_65 ( "ABCD" )//等同於LBL
8
9      R67 := R67 + 1;
```

能夠在 ST 環境呼叫 Label 的關鍵字有以下幾種

JMP_66/GOTO (詳細說明可以參考 FUN 66)

EX:

```
JMP_66 ("123")

GOTO ("456")
R66:=R68+1;

LBL ("123")
R2:=123;

LBL ("456")
R4:=456;
```

JMP_66/GOTO 只能在當前單元執行跳轉

CALL/CALL_67 (詳細說明可以參考 FUN 67)

EX:

MainUnit > 主單元0

```
1 CALL ("123")
2 CALL_67 ("456")
3
```

SubUnit > 副單元0

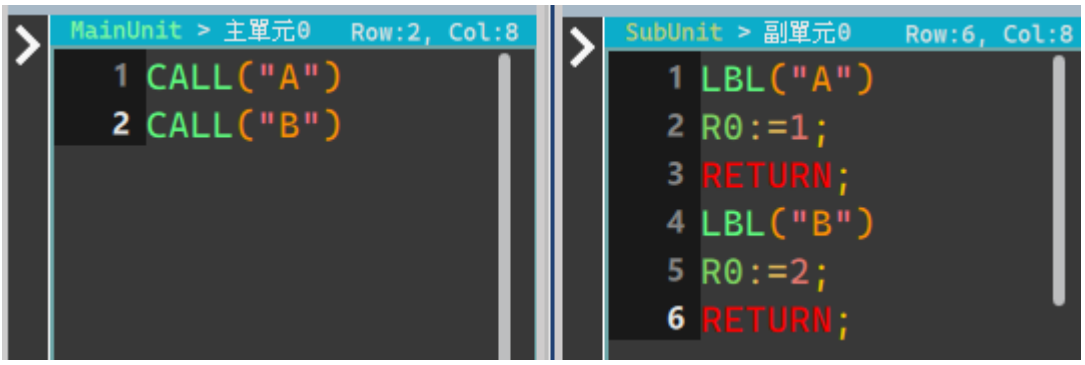
```
1 LBL ("123")
2 R2:=123;
3
```

SubUnit > 副單元1

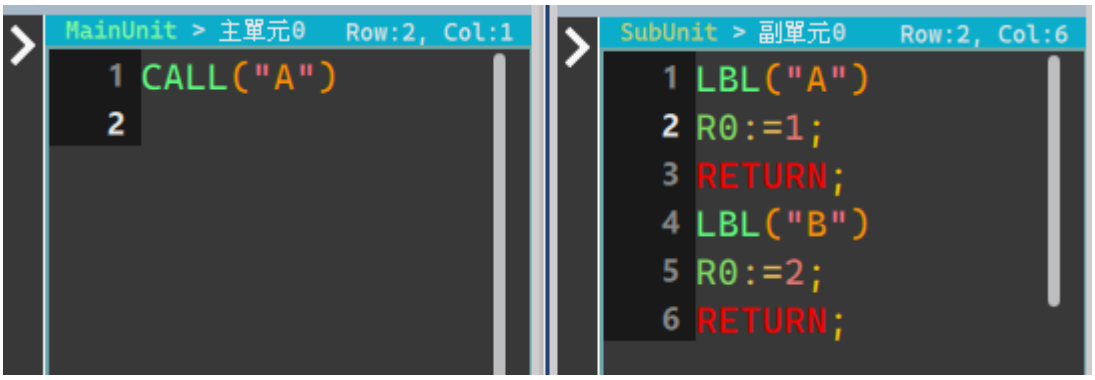
```
1 LBL ("456")
2 R4:=456;
3
```

在副程式使用 Label 功能時，LBL 指令搭配 RETURN 會有不同的結果

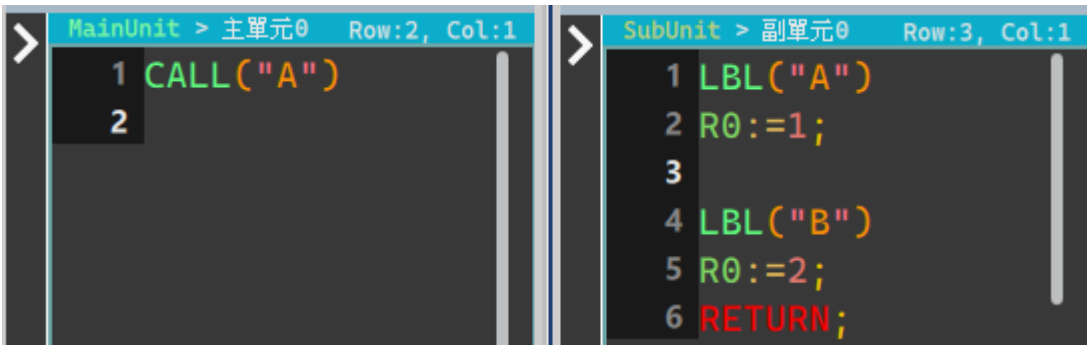
EX:



R0=2,因為同時呼叫 A.B 的 Label,並將結果 RETURN



R0=1,因為只呼叫了 A 的 Label,並將結果 RETURN



R0=2,因為呼叫了 A 的 Label,但 RETURN 在 B 的 Label 後

R0=3, 因為呼叫了"副單元 0"A 的 Label,但 RETURN 在"副單元 1"C 的 Label 後(需特別注意)

如需在 FCM 環境

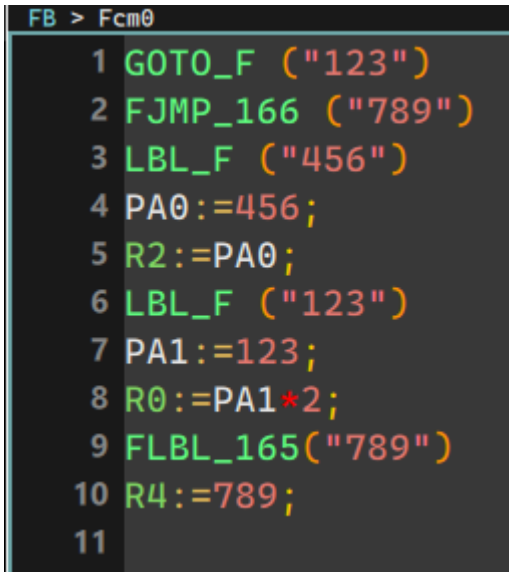
使用標籤則需要利用 LBL_F、FLBL_165 指令

只能在 FCM 呼叫 Label 的關鍵字

GOTO_F(只能在 FB 使用的 Label)

FJMP_166

EX:



3-7 暫存器和資料型別

在程式語言中，變數使用跟資料型別是不可或缺的一部分為了要賦予暫存器型別的特性，便於程式撰寫者觀看除錯，ST 都使用標籤(全域、區域) 來賦予暫存器具有特定的型別，以下一一介紹 ST 中所支援的資料型態跟用法。

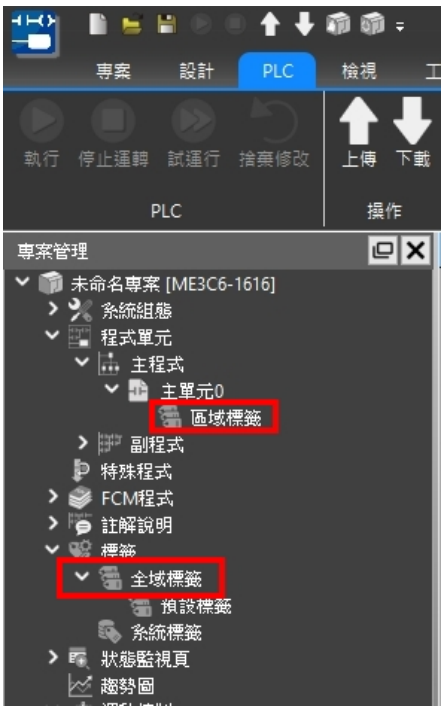
注意事項

運算結果超出了資料類型所能處理的值的範圍時，在以後的處理中將無法反映正確的結果（數值）。

應事先將運算物件變數的資料類型轉換成可對運算結果進行處理的範圍的資料類型。

標籤建立範例

標籤分為全域標籤及區域標籤，全域標籤所建立的標籤名稱無法與區域標籤共用，為整個專案可共用的標籤名稱。區域標籤所建立的標籤名稱可在不同主程式、副程式、Fcm 程式間使用相同的名字，且不相互影響使用(但使用位址須不相同)。

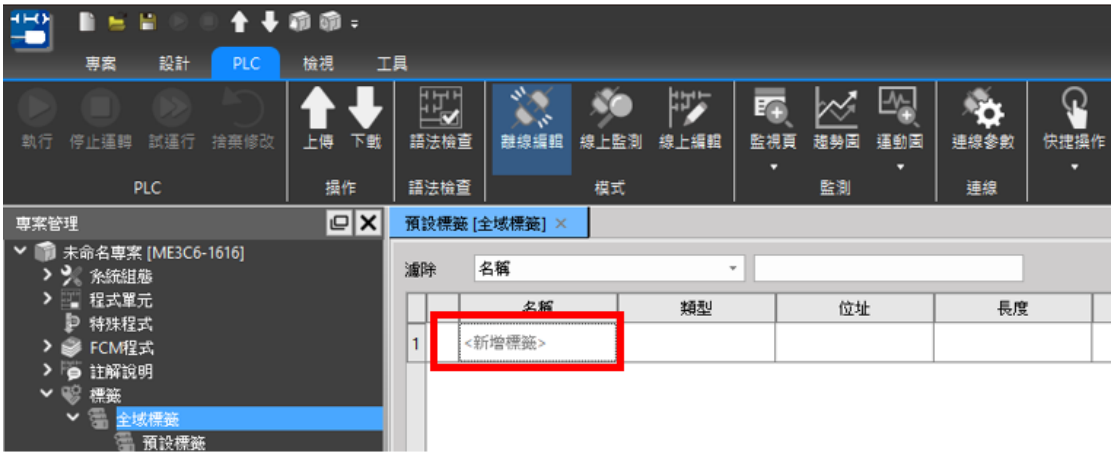


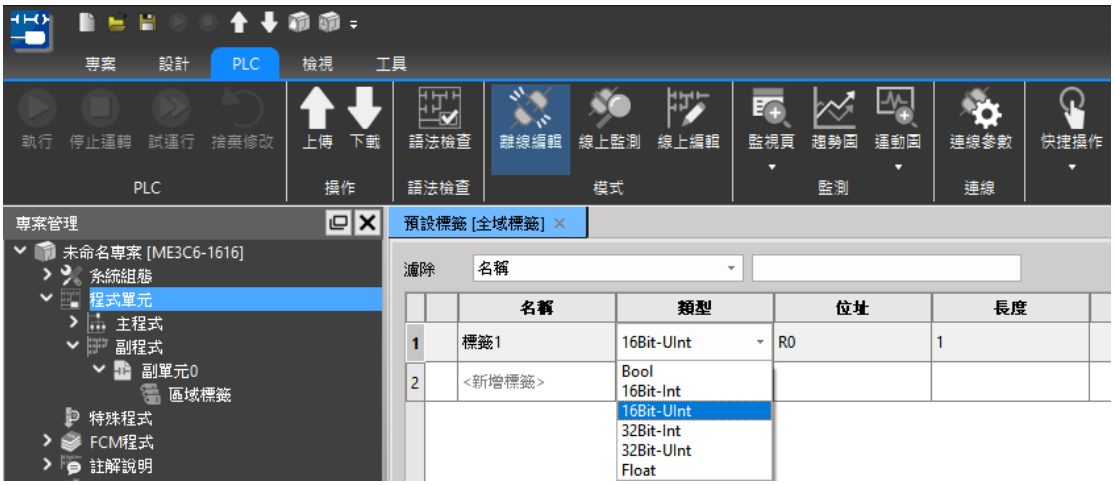
為配合不同資料型別的程式運算使用，以下將示範如何建立不同型別的標籤

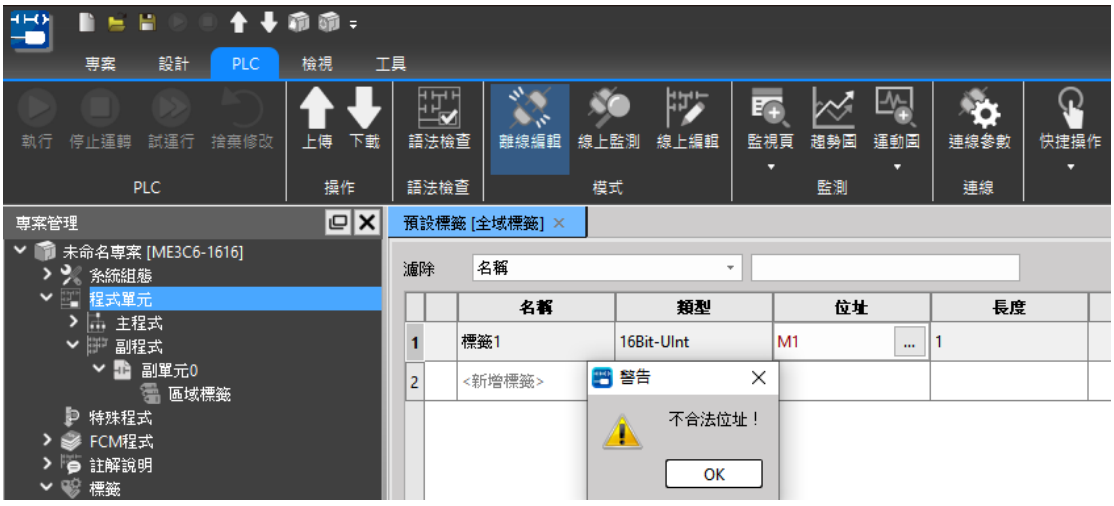
Rx.Dx 暫存器預設為 INT16, DRx.DDx 暫存器預設 INT32 整數型變數

EX:

雙擊點選全域標籤->新增標籤







如果選取的位址與資料型別不支援，則會跳出提醒阻擋

3-7-1 Bool / Bit

只要是代表的資料為 bit 或是數值比較運算 ex. <, >, <> ...等, 這些結果都為 Bool (Bit) 型別，在這裡可以使用 0 或是 1 , TRUE, FALSE 都可以接受為表現 Bool 型別的常數。其中繼電器的開關也都是用 Bool / Bit 來表示 (ex. M0, X0, Y0...etc)

EX:

```
1 M0:=1;
2
3 M0:=TRUE;
4
5 M1:=R0<10;
6
```

3-7-2 Integer 整數型別

ST 有四種整數類型 分別為 INT16, UINT16, INT32, UINT32，

型別名稱	說明
INT16	**16 位元的整數**
UINT16	**16 位元的正整數**
INT32	**32 位元的整數**
UINT32	**32 位元的正整數**

如果使用到 16 bit 的暫存器位置 (ex. R0, R1..etc) 系統會預設為 INT16 的資料型別處理，如果是 32 bit 的暫存器 (ex. DR0, DR2...etc)則會以 INT32 的方式對待

3-7-3 Float 浮點數

IEEE-754 定義的浮點數

ST 針對浮點數的操作, 都需要先產生對應的 Tag 才可以讓系統知道哪個變數指的暫存器位置代表著浮點數.

其中當使用者在做浮點數的操作的時候會有一些預設隱含的行為出現，以下圖的範例來說明。

Line 1: 表示將標籤 0 的浮點數數值賦值為 69905.14159

Line 3: 表示把標籤 0 的浮點數數值轉換成 int32 的整數數值
並且存在 DR10 的位置,小數點的部分則會被移除
(ex. 標籤 0 為 69905.141 則 DR10 內容則為 69905)

Line 5: 表示會把標籤 0 的浮點數數值轉換成 int16 的整數數值
並且存在 R12 的位置,超過 int16 的整數數值以及小數點的部分則會被 移除
(ex. 標籤 0 為 69905.141 則 R12 內容則為 4369)

Line 2: DR14(INT32 **)轉換成浮點數的資料型別，並且存進標籤 0 這個標籤中**

(可觀察浮點數的數位表示跟整數不同，可當作 Fun 200 I -> F 運用)

EX:

3-7-4 常數

以下為 ST 所支援輸入的常數類型

Type	格式及樣本
Bool (Bit)	**0, 1, TRUE, FALSE** **----------** **M0:=1;** **M0:=TRUE;**
整數	**整數:** **R0:=1;** **R0:=-10;** **二進位:** 2# **後面接上 0,1 的數值,** **可以使用底線分割群組** **R0:=2#1111_0000;** **八進位:** 8# **後面接上數值,** **可以使用底線分割群組** **R0:=8#243;** **十六進位:** 16# **後面接上數值,** **可以使用底線分割群組** 0x **後面接上數值,** **可以使用底線分割群組**

科學記號	**Tag_FlR100:=1E3; //1000**
	Tag_FlR100:=1.2E+3; //1200
	Tag_FlR100:=1E-3; //0.001

單位符號	**G : 1e+9** **M : 1e+6** **K,k : 1e+3** **m : 1e-3** **u : 1e-6** **n : 1e-9** **----------** **DR0:=1G;** **R0:=1K;**

	R0:=1k; **DR0:=1M;** **Tag_FlR100:=1m;** **Tag_FlR100:=1n;** **Tag_FlR100:=1u;**
字串 (目前只有 Label 指令可以使用)	**用 “ quote 符號 包起來的 ASCII 字串** **----------** **LBL ("my_lab")**

3-8 使用 PLC 的暫存器跟記憶體

除了使用建立 Tag 來調用對應的暫存器之外，也可以使用間接暫存器名來進行運算或是流程控制。

只要是該暫存器的資料型態為 single word 16bit ---> INT16
double word 32bit ---> INT32

EX:

```
R29:=100;
V:=19; //間接定址
R200:= R10V+100;     R200=200
```

比較特別的 T, C 這兩個位元只會根據程式的邏輯會有 bit 或是 int 的特性

EX:

```
/// T1 表示為 bit 型別
/// 代表是否 timeout
IF T1 THEN
   R10:=20;

/// T1 表示當前 timer 的數值
/// 為 int16 的整數
ELSEIF T1>100 THEN
    R10:=30;
END_IF

///這時候會把 T1 當成 Bit 類型來做反向
M0:=NOT T1;
```

3-9 呼叫系統內建的函式

內建函式表達結構會如下所示

<函式名稱> (　<參數名稱 1> := <輸入參數 1>,　....);

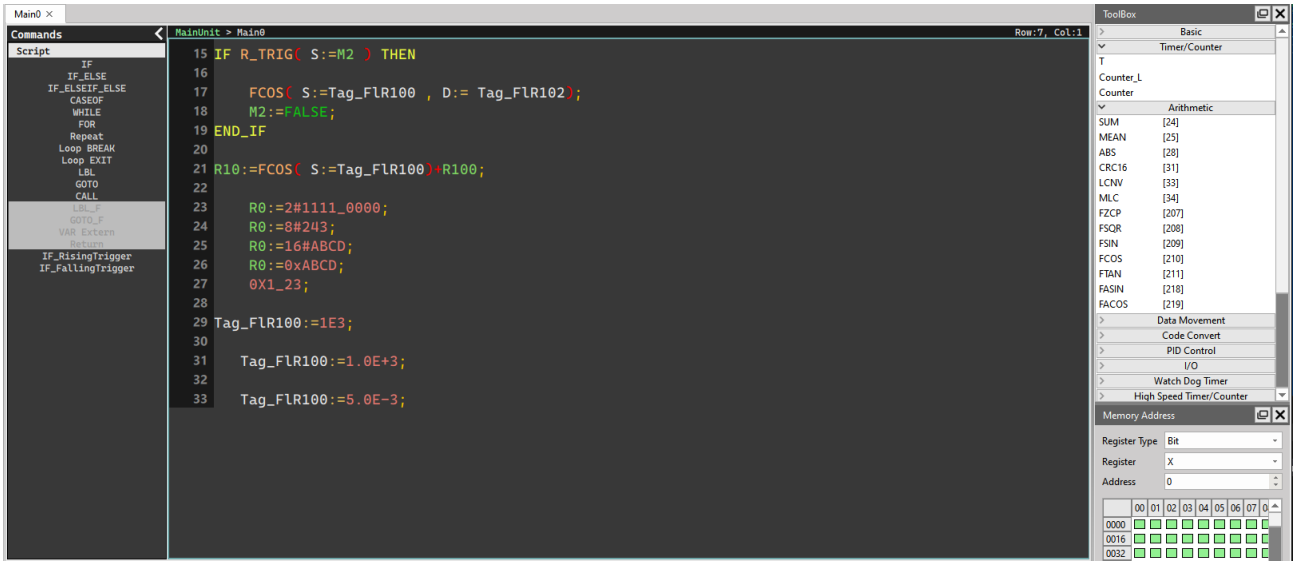
和階梯圖一樣，ST 就帶有一些內建的函式供使用者呼叫，

會出現在右側的工具箱欄位中。

可以從工具箱拖曳 或是雙擊該欄位，便會新增至文本中。

在呼叫的時候，參數的順序可以隨意變更，但是不能缺少，

除非一些特殊數允許省略

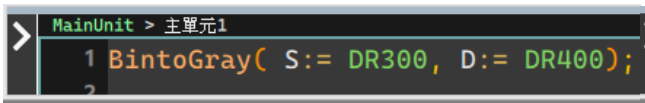


其中某些函式會有分 32 bit 模式 (預設都是 16 bit 模式)

會需要再該函式後面加上_D 明確指示該函示為 32 bit 模式

EX:二進制碼轉換格雷碼

錯誤轉換

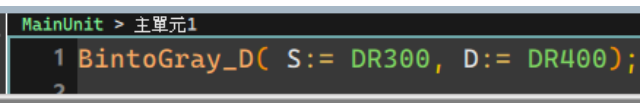


監視頁

檢視 編輯 檔案

名稱	狀態	資料	註解	名稱	狀
DR300	十六進制	FFFFFFFFH	[R300]		
DR400	十六進制	00008000H	[R400]		

正確轉換



監視頁

檢視 編輯 檔案

名稱	狀態	資料	註解	名稱	狀態
DR300	十六進制	FFFFFFFFH	[R300]		
DR400	十六進制	80000000H	[R400]		

大部分的函式都是呼叫即執行，如使用時希望條件達成後只執行一次動作，參考寫法

EX:

IF R_TRIG(S:=”條件”) then

“函式”

END_IF

!注意!

工具箱中的”Motion Basic”.”Motion Mode1”. ”Motion Mode2”. ”Motion Mode3”函式除非條件狀態有變化，否則都是只執行一次。

有關於每個呼叫函數的細部參數內容，則請參考 FUN 的文件，下面就針對幾個不同的 FUN 來介紹

3-9-1 Timer

Prototype:

```
Timer( Trigger:=    (* Trigger Bit Rising Edge ->ON, Falling Edge ->OFF *),
       T:=          (* Timer 0~1023 *),
       PV:=         (* preset value (0~32767) *),
       IsTimeout=> (* time out bit *);
```

參數:

Trigger (bool) : Rising Edge 則為 Timer 啟動，
Falling Edge 停止運行

T (int) : 0~1023 的整數，分別代表 T0~T1023 的 Timer //0,T0…etc

PV (int) : Timer 的設定值

IsTimeout (bool) : Timer 達到設定值後輸出的線圈 //M0,T0…etc

詳細參數說明請對應 LD Timer 的說明文件

3-9-2 Counter / Counter_L

這裡分成兩組 Counter 函式，讓使用者能容易分別目前呼叫的是

single word (Counter) 的 counter 還是 double words (Counter_L) 的版本

Prototype:

```
Counter( Pulse:= (* Pulse Signal *),
         Clr:= (* CLR Signal *),
         C:= (* counter number (0~1023) *),
         PV:= (* Preset value (Single Word 0~65535) *),
         IsUp=> (* Counter Is Up *);

Counter_L( Pulse:= (* Pulse Signal *),
           Clr:= (* CLR Signal *),
           C:= (* counter number (1024~1279 Long counter) *),
           PV:= (* Preset value (Double Words) *),
           IsUp=> (* Counter Is Up *);
```

參數:

Pulse	(bool)	:	Off->On 則會計數一次	//M0,M9129…etc
Clr	(bool)	:	是否清除 counter	
C	(int)	:	Counter id 0~1279	
PV	(int)	:	Counter 設定值	
IsUp	(bool)	:	Counter 達到設定值後輸出的線圈	//M0,C0…etc

3-9-3 R_TRIG / F_TRIG

上沿訊號和下沿訊號的偵測 function

Prototype:

Mode 1:

R_TRIG(S:= , D=>)

F_TRIG(S:= , D=>)

Mode 2:

R_TRIG(S:=)

F_TRIG(S:=)

隱含 Return Value D

參數:

S (bool) : Rising / Falling Edge 偵測來源

D (bool) : 偵測結果

Example:

Mode 1: R_TRIG(S:=M0, D:=M5);

Mode 2:

```
IF R_TRIG(S:=M0) THEN
    ++R100;
END_IF
```

3-9-4 TARESUB and TAREZEOFFSET

這兩個函式是由原本的 FUN258 MODCONF 皮重偏移指令 模式拆出來的，詳細參數設定可以參考原本 FUN258 的文件。拆成兩個獨立的函式也是便於使用者在撰寫的時候能夠一目了然呼叫的功能。

Prototype:

TAREZEOFFSET(EN:= , MD:= , ID:= , CH:= , WR:= , ERR=> , DN=>)

TARESUB(EN:= , RST:= , ID:= , CH:= , SB:= , ERR=>)

3-10 具有多重呼叫模式的函式

如前面 3-9-3 章節中的觸發偵測指令，這些指令都支援兩種型態，一種是直接完整呼叫，另一種是省略存放結果的暫存器位置(D)，下表則為目前具有該特性的函式。

函式名稱	具有回傳特性 參數名稱	函數格式
FSQR	D	FSQR(S:= ,D:=)
FSIN	D	FSIN(S:= ,D:=)
FCOS	D	FCOS(S:= ,D:=)
FTAN	D	FTAN(S:= ,D:=)
FASIN	D	FASIN(S:= ,D:= ,MD:=)
FACOS	D	FACOS(S:= ,D:= ,MD:=)
R_TRIG	D	R_TRIG(S:= ,D:=)
F_TRIG	D	F_TRIG(S:= ,D:=)
ItoF	D	ItoF(S:= ,D:=)
FtoI	D	FtoI(S:= ,D:=)

也就是說上述這些函式具有回傳特性的參數在呼叫的時候可以省略
並且直接用另一個參數承接或是直接來運算

EX:

沒有省略:

```
FSIN(S:=Tag_Float_DR100, D:=Tag_Float_DR102);
```

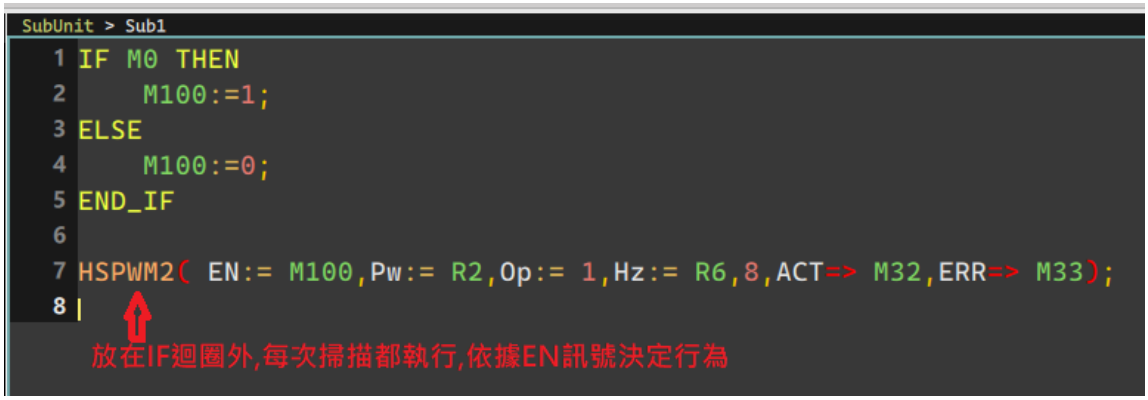
省略回傳參數: (由於回傳數值及代表結果，則可以直接帶入一些四則運算或是條件是判斷)

```
Tag_Float_DR200 := FSIN(S:=Tag_Float_DR100) + FSIN(S:=Tag_Float_DR102);
```

3-11 複週期指令集(Multi-cycle instructions)

以下指令列表為持續在背景運作型指令，使用方式為，放在 IF 迴圈外，每次掃描週期都執行，會以 EN 訊號是否成立，決定該函式的運行狀態。

如下圖的範例:



這些指令的類別，因為呼叫後，執行時間通常都超過一個掃描週期。

下表為目前所有的複週期指令

函式名稱	函式 ID
LCNV	33
MLC	34
TPCTL2	99
RAMP2	98
HSPWM	139
HSPSO	140
MPARA	141
PSOFF	142
PSCNV	143
MPG	148

ModBUS	**150**
CLINK	**151**
ReadWriteFileReg	**160**
WriteSDMem	**161**
ReadSDMem	**162**
PID2	**38**
DBUF	**115**
ICA	**137**
ICF	**138**
NCR	**152**
CMCTL	**156**
HSPWM2	**144**
Timer	**T**
T.01s	**87**
T.1s	**88**
T1s	**89**
MFSysInit	**187**
MFSysStop	**177**
MFSysRstAlm	**185**
MFSysRCPR	**188**
MFSysRCPW	**189**
MFSysCAMR	**191**
MFSysCAMW	**192**
MFSysCAMGen	**196**
MFSysSetVirt	**235**
MFSDOR	**236**

MFSDOW	**237**
MFFlowStart	**176**
MFFlowStop	**186**
MFFlowHalt	**184**
MFFlowResume	**183**
MFChgTbPrm	**181**
MFMapTbPrm	**198**
MFPointMov	**179**
MFHome	**178**
MFJog	**180**
MFGearMPG	**193**
MFVelCtl	**194**
MFTorqCtl	**195**
MFAxMov	**197**
MFAxLMov	**238**
MFAxCirMov	**239**
MFPathMov	**240**

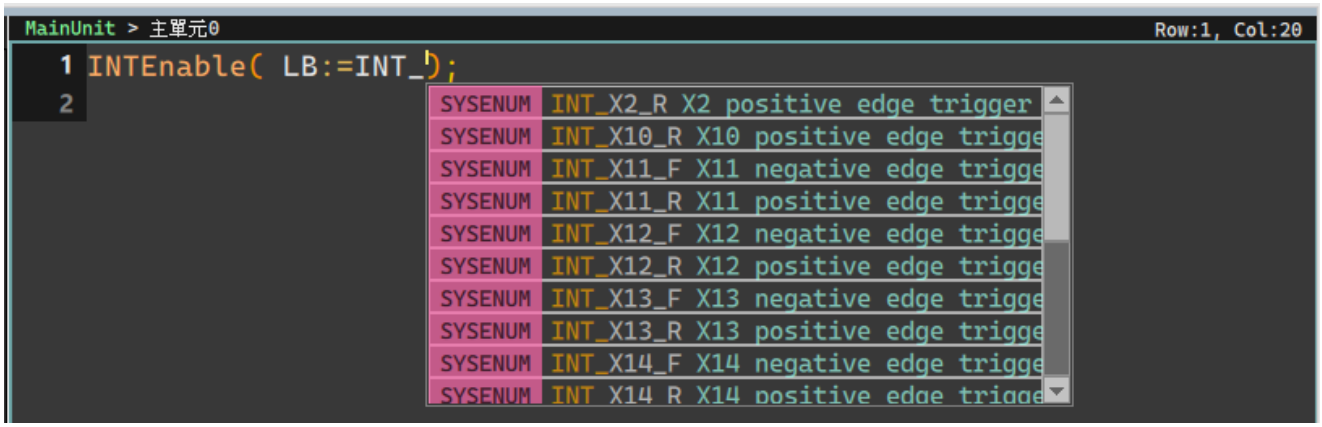
3-12 中斷,特殊指令啟動/關閉指令

請參照 FUN 145, 146

```
INTEnable( LB:= (* Interrupt number (Range 1~48) *));
INTDisable( LB:= (* Interrupt number (Range 1~48) *));
```

LB 參數可輸入 1~49 的整數，分別對應中斷的類型

也可藉由智能彈跳提示視窗選取對應中斷的類型



No.	中斷源	優先度	中斷標記	條件	備註
1	內建數位輸入		X0+I (INT0+)	X0 上沿觸發	軟體高速計數器 HSC4~HSC7 可以指定為任意中斷 X0~X15 的觸發源。 因此，軟體高速計數器的中斷優先權取決於 X0~X15 的優先權。
2			X0–I (INT0-)	X0 下沿觸發	
3			X1+I (INT1+)	X1 上沿觸發	
4			X1–I (INT1-)	X1 下沿觸發	

5			X2+I (INT2+)	X2 上沿觸發	
6			X2–I (INT2-)	X2 下沿觸發	
7			X3+I (INT3+)	X3 上沿觸發	
8			X3–I (INT3-)	X3 下沿觸發	
9			X4+I (INT4+)	X4 上沿觸發	
10			X4–I (INT4-)	X4 下沿觸發	
11			X5+I (INT5+)	X5 上沿觸發	
12			X5–I (INT5-)	X5 下沿觸發	
13			X6+I (INT6+)	X6 上沿觸發	
14			X6–I (INT6-)	X6 下沿觸發	
15			X7+I (INT7+)	X7 上沿觸發	
16			X7–I (INT7-)	X7 下沿觸發	
17	內部定時時基	3	STM0I	間隔 1ms~60000ms	時基單位 1ms，值=1~60000
18		3	STM1I	間隔 1ms~60000ms	
19		3	STM2I	間隔 1ms~60000ms	
20		3	STM3I	間隔 1ms~60000ms	

21		3	LTM0I	間隔 10ms~60000ms	時基單位 10ms，Value=1~6000
22		3	LTM1I	間隔 10ms~60000ms	
23		3	LTM2I	間隔 10ms~60000ms	
24		3	LTM3I	間隔 10ms~60000ms	
25	HST	1	HST0I	間隔 0.1ms to 6000ms	時基單位 100us，Value=1~60000
26		1	HST1I	間隔 0.1ms to 6000ms	
27		1	HST2I	間隔 0.1ms to 6000ms	
28		1	HST3I	間隔 0.1ms to 6000ms	
33	HSC	2	HSC0I	間隔 HSC0 to (CV=PV)	
34		2	HSC1I	間隔 HSC1 to (CV=PV)	
35		2	HSC2I	間隔 HSC2 to (CV=PV)	
36		2	HSC3I	間隔 HSC3 to (CV=PV)	
37		2	HSC4I	間隔 HSC4 to (CV=PV)	
38		2	HSC5I	間隔 HSC5 to (CV=PV)	
39		2	HSC6I	間隔 HSC6 to (CV=PV)	

40		2	HSC7I	間隔 HSC7 to (CV=PV)	
66	內建數位輸入 (MA 系列)		X8+I (INT8+)	X8 上沿觸發	軟體高速計數器 HSC4~HSC7 可以指定為任意中斷 X0~X15 的觸發源。 因此，軟體高速計數器的中斷優先權取決於 X0~X15 的優先權。
67			X8–I (INT8-)	X8 下沿觸發	
68			X9+I (INT9+)	X9 上沿觸發	
69			X9–I (INT9-)	X9 下沿觸發	
70			X10+I (INT10+)	X10 上沿觸發	
71			X10–I (INT10-)	X10 下沿觸發	
72			X11+I (INT11+)	X11 上沿觸發	
73			X11–I (INT11-)	X11 下沿觸發	
74			X12+I (INT12+)	X12 上沿觸發	
75			X12–I (INT12-)	X12 下沿觸發	
76			X13+I (INT13+)	X13 上沿觸發	
77			X13–I (INT13-)	X13 下沿觸發	
78			X14+I (INT14+)	X14 上沿觸發	

79			X14–I (INT14-)	X14 下沿觸發	
80			X15+I (INT15+)	X15 上沿觸發	
81			X15–I (INT15-)	X15 下沿觸發	
82	MQ model DI Plug-in		X16+I (INT15+)	X16 上沿觸發	
83			X16–I (INT15-)	X16 下沿觸發	
84			X17+I (INT15+)	X17 上沿觸發	
85			X17–I (INT15-)	X17 下沿觸發	
86			X18+I (INT15+)	X18 上沿觸發	
87			X18–I (INT15-)	X18 下沿觸發	
88			X19+I (INT15+)	X19 上沿觸發	
89			X19–I (INT15-)	X19 下沿觸發	
90			X20+I (INT15+)	X20 上沿觸發	
91			X20–I (INT15-)	X20 下沿觸發	
92			X21+I (INT15+)	X21 上沿觸發	
93			X21–I (INT15-)	X21 下沿觸發	
94			X22+I (INT15+)	X22 上沿觸發	
95			X22–I (INT15-)	X22 下沿觸發	

96			**X23+I (INT15+)**	**X23 上沿觸發**	
97			**X23–I (INT15-)**	**X23 下沿觸發**	

3-13 具有特殊意義的變數名稱

A. Timer, Counter

T0~TN, C0~CN

在 ST 環境中賦予這兩個變數同時具有 Bool 的特性跟 Integer 整數的特性

例如:

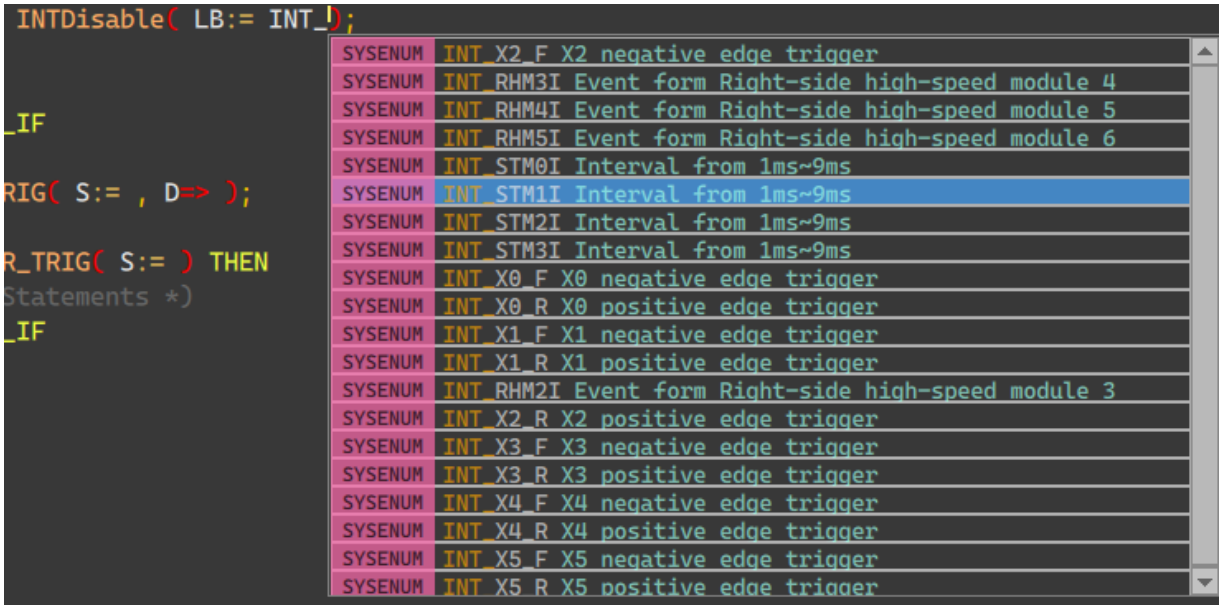
```
R0:=T1;   //會把 Timer1 目前計算到的數值存到 R0(Integer)
M0:=T1;   // 會把 Timer1 目前是否在運行的狀態存到 M0 (Bool)
M3:=C1>10; // 則會判斷 Counter 1 是否計數超過 10

IF R_TRIG( T1 ) THEN
   /// 當 Timer1 從 0 變 1 的時候 (開始運行的瞬間),則會執行

END_IF
```

B. **中斷 Enum**

原本 Enable / Disable 中斷只能輸入常數，為了讓程式看起來可讀性提高，修改為可以輸入系統預設的 Enum 參數，可以打 INT_ 的字樣就會顯示對應的一系列可用的 Enum。



例如:

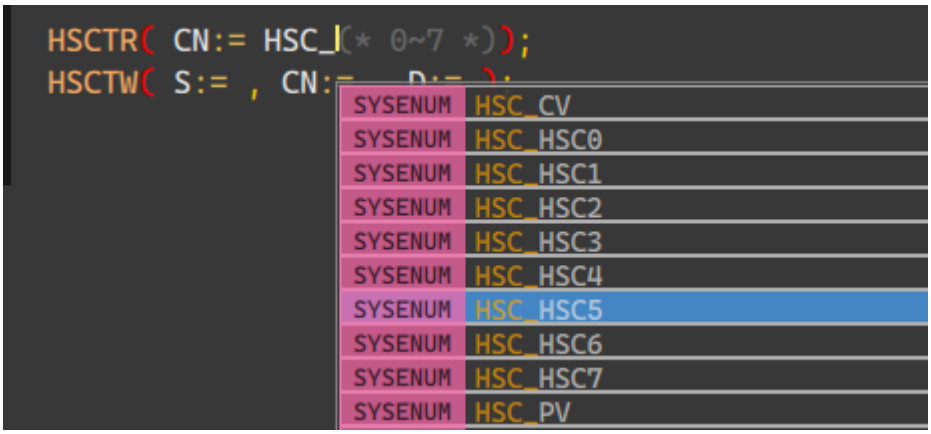
```
IF M2 THEN
    INTEnable( LB:= INT_STM1I);
ELSE
    INTDisable( LB:= INT_STM1I);
END_IF
```

C. 高速計數 Enum “HSC_” Prefix

呼叫相關高速指令如

```
HSCTR( CN:= (* 0~7 *));
HSCTW( S:= , CN:= , D:= );
```

其中 CN , 跟 D 的參數式支援常數整數，為了閱讀方便 也導入 HSC_ 開頭的 Enum



讓程式更具可讀性

例如:

```
HSCTR( CN:= HSC_HSC0(* 0~7 *));
HSCTW( S:=R0 , CN:=HSC_HSC0 , D:=HSC_PV );
```

D. **File Register Enum　F0~F32767**

針對 FUN 160 的 Sb 參數，除了使用 0~32767 之外，可以使用 F0~F32767 的 Enum 來代表對應的位置.

例如:

```
15
16 ReadWriteFileReg( EN:= R_TRIG(M0)(*  *),
17                   R_W:= 1(*  *),
18                   INC:= M10(*  *),
19                   Sa:=  R11(*  *),
20                   Sb:=  F10(* 0~32767 => F0~F32767 *),
21                   Pr:=  R100(*  *),
22                   L:=   10(*  *));
23
```

3-14 呼叫 FCM 函數

呼叫 FCM 的方式與呼叫系統函式類似，不過這些 FCM 函式都是使用者自行創建的

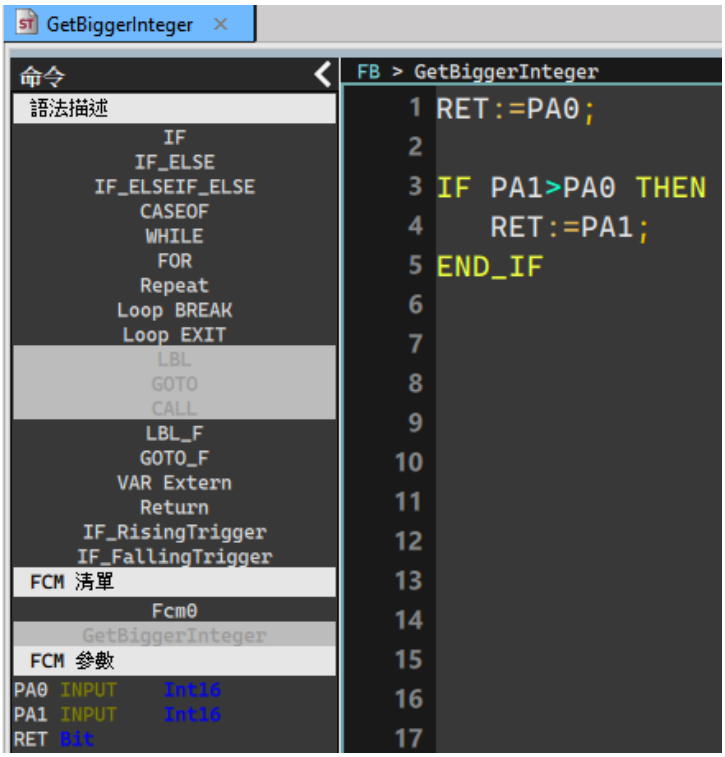
創建的 FCM 會放在左側的命令欄位中的 FCM list

可以直接雙擊 該欄位，插入所選取的函數到文本

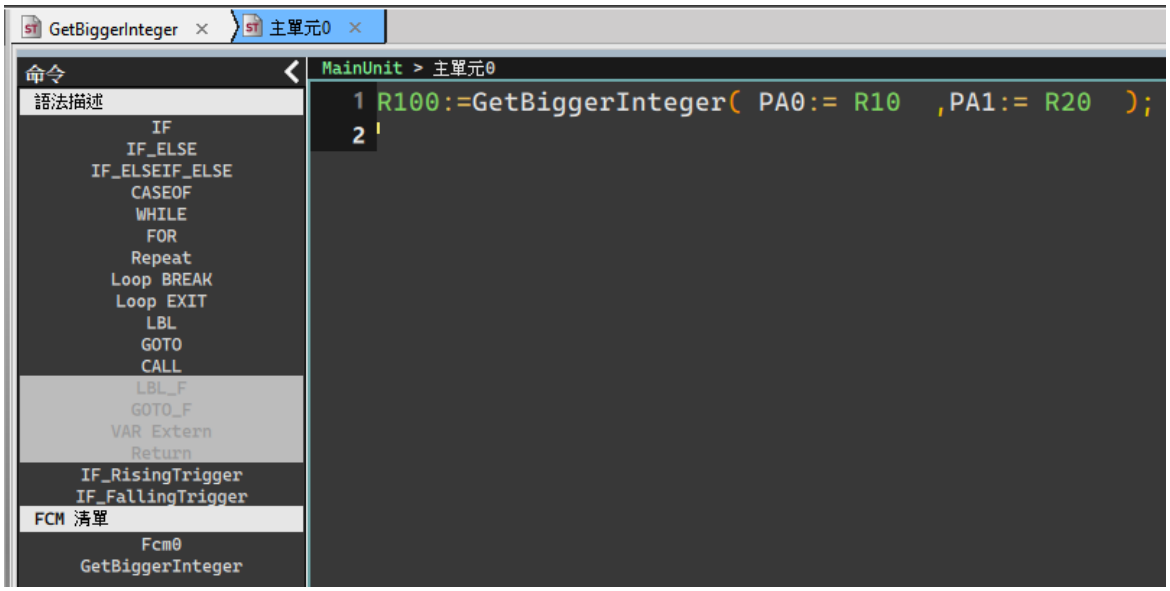
其中 FCM 可以指定一個 Return Value, 用來使用在程式撰寫的時候可以直接調用

EX:

FCM:



呼叫端:



這樣可以適時地減少一些暫時不需要的變數宣告

3-15 函數注意事項

無論是呼叫系統內建的或是 FCM 均有以下共同特點

A. 呼叫方式 ：

1. **完整帶入參數名稱**
2. **省略參數名稱**
3. **輸出參數(=>)可以直接忽略不傳遞**

範例:

以 R_TRIG 這個函式為例

完整表現 ： R_TRIG(S:= M0, D=>M1);

用參數帶入的話允許不按照預設的順序

上述的函式可以改成

R_TRIG(D=>M1, S:= M0);
R_TRIG(D=>M1);
都是合法輸入

省略參數名稱 ： R_TRIG(M0, M1);

帶有 Return 效果或是輸出的可以不傳遞 ：

```
R_TRIG( S:= M0 );

R_TRIG( M0 );
```

有 Return 特性的可以與其他變數或是描述句混合運用

```
IF R_TRIG(M0) THEN
   //// some statements
END_IF

M10:= R_TRIG(M0) AND R_TRIG(M1);
```

！注意 ！ 帶入參數跟不帶入參數的呼叫方式不能混用如:

R_TRIG(S:=M0, M1);
R_TRIG(M1,D:=M2);
以上兩個都是不合法的輸入

4

ST 編輯環境支援函示列表

4-1 Ladder 與 ST 編輯環境函示的名稱對照

ST 指令名稱	參數	指令手冊對應資訊		備註
		編號	名稱	
ToBCD ToBCD_D	S,D	Fun20	BCD	
ToBIN ToBIN_D	S,D	Fun21	BIN	
F_TRIG	S,D	Fun4	DIFU	
R_TRIG	S,D	Fun5	DIFD	
Timer	EN,T,PV,IsTimeout	T	TIMER	
Counter Counter_L	Pulse,clr,C,PV,IsUp	C	COUNTER	
ABS ABS_D	D	Fun28	ABS	
CRC16	S,N,D	Fun31	CRC16	
DIV	Sa,Sb,D	Fun14	DIVISION	
FtoI FtoI_D	S,D	Fun201	F->I	
FACOS_deg	S,D	Fun219	FACOS	
FACOS_rad	S,D	Fun219	FACOS	
FASIN_deg	S,D	Fun218	FASIN	
FASIN_rad	S,D	Fun218	FASIN	
FCOS	S,D	Fun210	FCOS	
FDIV	Sa,Sb,D	Fun205	FDIV	
FMUL	Sa,Sb,D	Fun204	FMUL	
FSIN	S,D	Fun209	FSIN	
FSQR	S,D	Fun208	FSQR	
FTAN	S,D	Fun211	FTAN	
FZCP	S,Su,S1,INZ,ERR S_Gt_U, S_Less_L	Fun207	FZCP	
ItoF ItoF_D	S,D	Fun200	I->F	
LCNV LCNV_D	EN,Md,S,Ts,D,L	Fun33	LCNV	

MEAN MEAN_D	S,N,D	Fun25	MEAN	
MLC	EN,XY,Rs,Sl,Tx,Ty,Tl,D,OVR	Fun34	MLC	
MUL	Sa,Sb,D	Fun13	MULTIPLICATION	
SUM SUM_D	S,N,D	Fun24	SUM	
BITRD BITRD_D	S,N,OTB,ERR	Fun40	BITRD	
BITWR BITWR_D	INB,D,N,ERR	Fun41	BITWR	
DBUF	EN,ID,CH,D,DN	Fun115	DBUF	
ReadSDMem	EN,INC,Bk,Os,Pr,L,D,ERR	Fun162	RD-MP	
ReadWriteFileReg ReadWriteFileReg_D	EN,R_W,INC,Sa,Sb,Pr,L	Fun160	RW-FR	
SWAP	D	Fun46	SWAP	
WriteSDMem	EN,INC,S,Bk,Os,Pr,L,WR,ACT,ERR,DN	Fun161	WR-MP	
ToASCII	S,N,D	Fun64	->ASCII	
ToBCD ToBCD_D	S,D	Fun20	->BCD	
ToBIN ToBIN_D	S,D	Fun21	->BIN	
ToHEX	S,N,D	Fun63	->HEX	
ToHMS	S,D	Fun62	->HMS	
ToSEC	S,D	Fun61	->SEC	
BintoGray BintoGray_D	S,D	Fun55	B->G	
DECOD	S,Ns,Nl,D,D,ERR	Fun57	DECOD	
ENCOD	H_L,S,Ns,Nl,D,D_0,ERR	Fun58	ENCOD	
GraytoBin GraytoBin_D	S,D	Fun56	G->B	
PID PID_D	AM,BUM,D_R,Ts,SR,O_R,PR,WR,CC	Fun30	PID	

	ERR,HAL,LAL			
PID2	EN,UPD,AM,D_R, ID,CH,SR,OutR,PR, WR,ERR	Fun38	PID2	
TPCTL2	EN,UPD,A_M,D_R, ID,CH,SR,PR,OriR, WR,ERR,ALM	Fun99	TPCTL2	
IMDIO	D,N	Fun74	IMDIO	
SPD	EN,S,TI,D,OVF	Fun83	SPD	
ACTimer_10MS ACTimer_10MS_D	TIM,EN,CV,PV,TUP, NUP	Fun87	T.01S	
ACTimer_100MS ACTimer_100MS_D	TIM,EN,CV,PV,TUP, NUP	Fun88	T.1S	
ACTimer_1S ACTimer_1S_D	TIM,EN,CV,PV,TUP, NUP	Fun89	T1S	
RSWDT		Fun91	RSWDTF	
WDT	N	Fun90	WDT	
HSCTR	CN	Fun92	HSCTR	
HSCTW	S,CN,D	Fun93	HSCTW	
RAMP2	EN,Om,Ta,Td,Rt,Rc, WR,ACC,DEC	Fun98	RAMP2	
CLINK	EN,PAU,ABT,Pt,MD ,WR,ERR,DN	Fun151	CLINK	
CMCTL	EN,PAU,ID,Pt,Ts, MD,WR,ERR	Fun156	CMCTL	
ModBUS	EN,A_R,ABT,PT,SR WR,ACT,ERR,DN,	Fun150	M-Bus	
NCR	NE,SR,MD,WR,AC T,ERR,DN	Fun152	NCR	
BKCMP BKCMP_D	Rs,Ts,L,D	Fun112	BKCMP	
BT_M BT_M_D	Ts,Td,L	Fun103	BT_M	
QUEUE QUEUE_D	InOut,IW,Qu,L,Pr, OW,EPT,FUL	Fun110	QUEUE	
T_Search T_Search_D	FHD,DS,Rs,Ts,L,Pr,F ND,END,ERR	Fun105	R-T_S	

SORT SORT_D	AD,S,D,L	Fun113	SORT	
STACK STACK_D	InOut,IW,ST,L,Pr, OW,EPT,FUL	Fun111	STACK	
T_Compare T_Compare_D	FHD,DS,Ta,Tb,L,Pr, FND,END,ERR	Fun106	T-T_C	
T_Fill T_Fill_D	Rs,Td,L	Fun107	T_FIL	
ZoneWR	Val,D,N	Fun114	Z-WR	
MAND	Ma,Mb,Md,L	Fun120	MAND	
MBCNT	OnOff,Ms,L,D	Fun130	MBCNT	
MBRD	EN,INC,CLR,Ms,L, Pr,OTB,END,ERR	Fun126	MBRD	
MBROT	L_R,Ms,Md,L,OTB	Fun129	MBROT	
MBSHF	INB,L_R,Ms,Md,L, OTB	Fun128	MBSHF	
MINV	Ms,Md,L	Fun124	MINV	
MOR	Ma,Mb,Md,L	Fun121	MOR	
MXNR	Ma,Mb,Md,L	Fun123	MXNR	
MXOR	Ma,Mb,Md,L	Fun122	MXOR	
HSPSO	EN,PAU,ABT,Ps,SR, WR,ACT,ERR,DN	Fun140	HSPSO	
HSPWM	EN,Pw,Op,Rs,Pn, OutR,WR,ACT	Fun139	HSPWM	
HSPWM2	EN,Pw,Op,Hz,O_R, ACT,ERR	Fun144	HSPWM2	
ICA	EN,D_R,Ps,Is,Fo,Ag ,ACT,ERR,DN	Fun137	ICA	
ICF	EN,D_R,Ps,Is,Fo,Fd ,ACT,ERR,DN	Fun138	ICF	
MHSPO	EN,PAU,ABT,Gp,SR ,WR,ACT,ERR,DN	Fun147	MHSPO	
MPARA	EN,Ps,SR	Fun141	MPARA	
MPG	EN,Sc,Ps,Fo,Mr,WR ,ACT	Fun148	MPG	
PSCNV	EN,Ps,D	Fun143	PSCNV	
PSOFF	EN,Ps	Fun142	PSOFF	

INTDisable	LB	Fun146	DIS	
INTEnable	LB	Fun145	EN	
TARESUB	EN,RST,ID,CH,SB, ERR	Fun258	MODCONF	
TAREZEOFFSET	EN,MD,ID,CH,WR, ERR	Fun258	MOD	
MFSysCAMR	EN,Md,D,ID,L,ACT, ERR,DN	Fun191	MFSysCAMR	
MFSysCAMW	EN,Md,D,ID,L,ACT, ERR,DN	Fun192	MFSysCAMW	
MFSysInit	EN,ACT,ERR,DN	Fun187	MFSysInit	
MFSysRCPR	EN,Md,D,Gp,ACT,E RR,DN	Fun188	MFSysRCPR	
MFSysRCPW	EN,MD,D,GP,ACT, ERR,DN	Fun189	MFSysRCPW	
MFSysRstAlm	EN,ACT,ERR,DN	Fun185	MFSysRstAlm	
MFSysSetVirt	EN,AX,ACT,ERR,D N	Fun235	MFSysSetVirt	
MFSysStop	EN,ACT,ERR,DN	Fun177	MFSysStop	
MFChgTbPrm	EN,GP,ID,N,D,ERR, DN	Fun181	MFChgTbPrm	
MFFlowHalt	EN,ID,ACT,ERR,DN	Fun184	MFFlowHalt	
MFFlowPause	EN,ID,ACT,ERR,DN	Fun182	MFFlowPause	
MFFlowStart	EN,ID,ACT,ERR,DN	Fun176	MFFlowStart	
MFFlowStop	EN,ID,ACT,ERR,DN	Fun186	MFFlowStop	
MFMapTbPrm	EN,GP,ID,N,D,ERR, DN	Fun198	MFMapTbPrm	
MFHome	EN,AX,ACT,ERR, DN	Fun178	MFHome	
MFJog	EN,D_R, AX, MD,ACT,ERR ,DN	Fun180	MFJog	
MFPointMov	EN,PT,AX,ACT,ERR, DN	Fun178	MFPointMov	
MFAxCirMov	EN,UPD,TAX,MD, DR,BF,D,EC,ACT,ER R,DN,UPD	Fun239	MFAxCirMov	
MFAxLMov	EN,UPD,TAX,MD,	Fun238	MFAxLMov	

	DR,BF,D,EC,ACT, ERR,DN,UPD			
MFAxMov MFAxMov_D	EN,UPD_in,AX,MD ,Ps,V_n,A,D,SA,SD, DR,BF,ACT,ERR,DN ,UPD_out	Fun197	MFAxMov	
MFGearMPG MFGearMPG_D	EN,UPD_in,M,S,N, D,T,ACT,ERR,DN,U PD_out	Fun193	MFGearMPG	
MFPathMov	EN,ACT,UPD,AX1, AX2,AX3,PT,V,TA,T D,SA,SD,EC,ACT,ER R,DN,UPD	Fun240	MFPathMov	
MFTorqCtl MFTorqCtl_D	EN,UPD_in,AX,T,M X,ACT,ERR,DN, UPD_out	Fun195	MFTorqCtl	
MFVelCtl MFVelCtl_D	EN,UPD_in,AX,T,M X,ACT,ERR,DN, UPD_out	Fun194	MFVelCtl	

4-2 ST 編輯環境支援函示的參數說明

ST 指令名稱	參數	參數說明
ToBCD ToBCD_D	S,D	S：被轉換之資料或其暫存器號碼。 D：存放轉換結果(BCD 碼)之暫存器號碼。
ToBIN ToBIN_D	S,D	S：被轉換之資料或其暫存器號碼。 D：存放轉換結果(BIN 碼)之暫存器號碼。
F_TRIG	S,D	S：被轉換之資料或其暫存器號碼。 D：存放上微分結果之繼電器線圈號碼
R_TRIG	S,D	S：被轉換之資料或其暫存器號碼。 D：存放上微分結果之繼電器線圈號碼
Timer	EN,T,PV,IsTimeout	Tn：計時器號碼，為儲存累計之計時時間（即現 在值 CV）。 PV：（Preset Value）為計時器之設定值。
Counter Counter_L	Pulse,clr,C,PV,IsUp	Cn：計數器號碼，為儲存累計之計數值（即現在 值 CV）。 PV：（Preset Value）為計數器之設定值。
ABS ABS_D	D	D：取絕對值之暫存器號碼
CRC16	S,N,D	MD：0，計算 CRC 時，祇計算暫存器之低位元組，暫存器之高位元組不計算 ：1，保留 S：需計算 CRC 之起始暫存器號碼 N：需計算 CRC 之資料長度，單位為 Byte D：存放 CRC 計算結果之暫存器號碼，暫存器 D 存放 CRC 運算結果之 Upper Byte 暫存器 D＋1 存放 CRC 運算結果之 Lower Byte S，N，D 運算元可結合 V、Z、P0~P9 指標作間接定 址應用
DIV	Sa,Sb,D	Sa：被除數或其暫存器號碼。 Sb：除數或其暫存器號碼。 D ：存放結果(商和餘數)之暫存器號碼。 Sa，Sb，D 可結合 V、Z、P0~P9 作間接定址應用。
FtoI FtoI_D	S,D	S：來源暫存器之起始號碼 D：存放結果(整數)之暫存器起始號碼運算元使用之暫存器必

		須是偶數位址，例 R8 是合法 的，R7 是不合法的。 S、D 運算元可結合 V、Z、P0~P9 指標作間接定址應 用
FACOS_deg	S,D	S：求反正弦函數值之來源數值或暫存器號碼。 運算元使用之暫存器必須是偶數位址，例 R8 是合法 的，R7 是不合法的。 D：存放結果(反正弦函數值)之暫存器號碼。 S、D 可結合 V、Z、P0~P9 作間接定址應用。
FACOS_rad	S,D	
FASIN_deg	S,D	S：求反正弦函數值之來源數值或暫存器號碼。 運算元使用之暫存器必須是偶數位址，例 R8 是合法 的，R7 是不合法的。 D：存放結果(反正弦函數值)之暫存器號碼。 S、D 可結合 V、Z、P0~P9 作間接定址應用。
FASIN_rad	S,D	
FCOS	S,D	S：求 COS 值之來源數值或暫存器號碼。 D：存放結果之 暫存器號碼。 運算元使用之暫存器必須是偶數位址，例 R8 是合法 的，R7 是不合法的。 S、D 運算元可結合 V、Z、P0~P9 指標作間接定址應用
FDIV	Sa,Sb,D	Sa：被除數或其暫存器號碼。 Sb：除數或其暫存器號碼。 D：存放結果(商)之暫存器起始號碼 運算元使用之暫存器必須是偶數位址，例 R8 是合法 的，R7 是不合法的。 S、D 運算元可結合 V、Z、P0~P9 指標作間接定址應用
FMUL	Sa,Sb,D	Sa：被乘數或其暫存器號碼。 Sb：乘數或其暫存器號碼。 D：存放結果(積)之暫存器起始號碼 運算元使用之暫存器必須是偶數位址，例 R8 是合法 的，R7 是不合法的。 S、D 運算元可結合 V、Z、P0~P9 指標作間接定址 應用
FSIN	S,D	S：求 SIN 值之來源數值或暫存器號碼。 D：存放結果之暫存器號碼。 運算元使用之暫存器必須是偶數位址，例 R8 是合法 的，R7 是不合法的。 S、D 運算元可結合 V、Z、P0~P9 指標作間接定址 應用
FSQR	S,D	S：求平方根之來源數值或暫存器號碼。 D：存放結果(平方根值)之暫存器號碼。 運算元使用之暫存器必須是偶數位址，例 R8 是合法 的，R7 是不合法的。 S、D 運算元可結合 V、Z、P0~P9 指標作間接定址 應用

FTAN	S,D	S：求 TAN 值之來源數值或暫存器號碼。 D：存放結果之暫存器號碼。 運算元使用之暫存器必須是偶數位址，例 R8 是合法 的，R7 是不合法的。 S、D 運算元可結合 V、Z、P0~P9 指標作間接定址應用
FZCP	S,Su,S1,INZ,ERR S_Gt_U, S_Less_L	S ：存放比較資料(浮點數)之暫存器號碼。 SU：區域上限值或上限值暫存器號碼。 SL：區域下限值或下限值暫存器號碼。 運算元使用之暫存器必須是偶數位址，例 R8 是合法 的，R7 是不合法的。 S、D 運算元可結合 V、Z、P0~P9 指標作間接定址應用
ItoF ItoF_D	S,D	S ：來源暫存器之起始號碼 D ：存放結果(浮點數)之暫存器起始號碼 運算元使用之暫存器必須是偶數位址，例 R8 是合法 的，R7 是不合法的。 S、D 運算元可結合 V、Z、P0~P9 指標作間接定址應用
LCNV LCNV_D	EN,Md,S,Ts,D,L	Md：運算模式選擇，0～3 S ：欲轉換之來源暫存器起始號碼 Ts：轉換表格起始暫存器起始號碼 D：存放轉換結果之起始暫存器號碼 L：欲轉換之長度，1～64
MEAN MEAN_D	S,N,D	S ：來源暫存器之起頭號碼 N ：欲平均之暫存器個數 （由 S 開始連續 N 個） D ：存放結果（平均值）之暫存器號碼 S，N，D 可結合 V、Z、P0~P9 作間接定址應用
MLC	EN,XY,Rs,Sl,Tx,Ty,Tl,D,OVR	Rs：來源資料起始暫存器號碼 Sl：欲轉換之來源資料長度，1～64 Tx：X 轉換表格起始暫存器起始號碼 Ty：Y 轉換表格起始暫存器起始號碼 Tl：轉換表格長度，2～255 D：存放轉換結果之起始暫存器號碼
MUL	Sa,Sb,D	Sa：被乘數或其暫存器號碼。 Sb：乘數或其暫存器號碼。 D ：存放結果(積)之暫存器號碼。 Sa，Sb，D 可結合 V、Z、P0~P9 作間接定址應用。
SUM SUM_D	S,N,D	S：來源暫存器之起頭號碼 N：欲總和之暫存器個數（由 S 開始連續 N 個）

		D：存放結果（總和）之暫存器號碼 S，N，D 可結合 V、Z、P0~P9 作間接定址應用
BITRD BITRD_D	S,N,OTB,ERR	S ：欲讀取位元之資料或其暫存器號碼 N：指定 S 資料中第 N 個位元資料被讀出 S，N 可結合 V、Z、P0~P9 作間接定址應用
BITWR BITWR_D	INB,D,N,ERR	D：欲寫入位元之暫存器號碼 N：指定將寫入位元 INB 之狀態寫入 D 中第 N 個位元處 D，N 可結合 V、Z、P0~P9 作間接定址應用
DBUF	EN,ID,CH,D, DN	ID：擴充模組 ID 編號(0 ~ N) CH：擴充模組通道索引 (0 ~ N) D：存放緩存數據起始暫存器
ReadSDMem	EN,INC,Bk,O s,Pr,L,D,ERR	BK：SD 卡之區塊號碼，0 ~ 1 Os：分區資料起始位置 Pr：指標暫存器號碼 L：讀取資料長度 1 ~ 128 D：存放讀取資料之暫存器起始號碼
ReadWriteFileR eg ReadWriteFileR eg_D	EN,R_W,INC, Sa,Sb,Pr,L	Sa：暫存器列表之起始暫存器號碼 Sb：檔案暫存器之起始號碼 Pr：指標暫存器號碼 L：列表之長度 1~511 Sa 可結合 V、Z、P0~P9 作間接定址應用
SWAP	D	D：執行位元組資料對換之暫存器號碼 D 可結合 V、Z、P0~P9 作間接定址應用
WriteSDMem	EN,INC,S,Bk, Os,Pr,L,WR,A CT,ERR,DN	S：寫入資料之來源起始暫存器號碼 BK：SD 卡之區塊號碼，0 ~ 1 Os：分區資料起始位置 Pr：指標暫存器號碼 L：寫入資料長度 1 ~ 128 WR：工作暫存器起始號碼，共佔用 2 個暫存器 S 可結合 V、Z、P0 ~ P9 作間接定址應用
ToASCII	S,N,D	S ：來源暫存器之起始號碼 N ：欲轉十六進制值為 ASCII 碼之個數 D ：存放結果（ASCII 碼）之暫存器起始號碼 S，N，D 可結合 V、Z、P0~P9 作間接定址應用
ToBCD	S,D	S：被轉換之資料或其暫存器號碼。

ToBCD_D		D：存放轉換結果(BCD 碼)之暫存器號碼。 S，D 可結合 V、Z、P0~P9 作間接定 址應用。
ToBIN ToBIN_D	S,D	S：被轉換之資料或其暫存器號碼。 D：存放轉換結果(BIN 碼)之暫存器號碼。 S，D 可結合 V、Z、P0~P9 作間接定址應用
ToHEX	S,N,D	S ：來源暫存器之起始號碼 N ：欲轉 ASCII 碼為十六進制值之個數 D ：存放結果（十六進制值）之暫存器起始號碼 S，N，D 可結合 V、Z、P0~P9 作間接定址應用
ToHMS	S,D	S ：欲變換之秒數資料暫存器起頭號碼 D ：變換結果〔時：分：秒〕存放之暫存器起頭號碼
ToSEC	S,D	S ：欲變換之時間資料暫存器起頭號碼 D ：存放結果之暫存器起頭號碼
BintoGray BintoGray_D	S,D	S ：來源暫存器之起始號碼 D ：存放結果（格雷碼）之暫存器起始號碼 S，D 運算元可結合 V、Z、P0~P9 指標作間接定址應用
DECOD	S,Ns,Nl,D,D,ERR	S ：解碼之來源資料暫存器號碼（16 位元） NS：S 中欲被解碼之起始位元 NL：解碼值之長度（1～8 位元） D ：存放解碼結果之暫存器起頭號碼 （2～256 點=1～16 Words） S，NS，NL、D 可結合 V、Z、P0~P9 作間接 定址應用
ENCOD	H_L,S,Ns,Nl,D,D_0,ERR	S ：被編碼之暫存器起頭號碼 NS：指定 S 中之一點為編碼起始點 NL：編碼之單點數目（2～256 點） D ：存放編碼結果之暫存器號碼 （1 個 Word） S，NS，NL，D 可結合 V、Z、P0~P9 作間 接定址應用
GraytoBin GraytoBin_D	S,D	S ：來源暫存器之起始號碼 D ：存放結果（格雷碼）之暫存器起始號碼 S，D 運算元可結合 V、Z、P0~P9 指標作間接定址應用
PID PID_D	AM,BUM,D_R,Ts,SR,O_R,PR,WR,CCERR,HA	Ts ：PID 運算間隔時間 SR ：程控設定值起始暫存器號碼， 共佔用 8 個暫存器 OR ：PID 輸出暫存器號碼 PR ：參數設定值起始暫存器號碼， 共佔用 7 個暫存器

	L,LAL	WR ：本指令所需使用之工作暫存器起 始號碼，共佔用 5 個暫存器，其它地方不可重覆使用。
PID2	EN,UPD,AM,D_R,ID,CH,SR,OutR,PR,WR,ERR	ID ：作輸入訊號的擴充模組的編號 CH : 作輸入訊號的通道編號 SR ：程控設定值起始暫存器號碼， 共佔用 8 個暫存器 OR ：PID 輸出暫存器號碼 PR ：參數設定值起始暫存器號碼， 共佔用 7 個暫存器 WR ：本指令所需使用之工作暫存器起始號碼，共佔用 5 個暫存器，其它地方不可重覆使用。
TPCTL2	EN,UPD,A_M,D_R,ID,CH,SR,PR,OriR,WR,ERR,ALM	ID：作輸入訊號的擴充模組的編號 CH: 作輸入訊號的通道編號 SR : 程控設定起始暫存器 PR : 增益設定起始暫存器 OR :輸出起始暫存器 WR: 工作起始暫存器
IMDIO	D,N	D：欲更新之 I/O 點起頭號碼 N：欲更新之 I/O 點數
SPD	EN,S,TI,D,OVF	S ：欲偵測速度之脈波輸入點 TI：偵測之取樣時間（單位為 mS） D ：存放結果之暫存器號碼
ACTimer_10MS ACTimer_10MS_D	TIM,EN,CV,PV,TUP,NUP	CV：存放計時時間（即現在值）之暫存器號碼 PV：計時器之設定或存放設定值之暫存器號碼範圍運算元 WX WY WM WS TMR CTR HR IR OR SR ROR DR
ACTimer_100MS ACTimer_100MS_D	TIM,EN,CV,PV,TUP,NUP	
ACTimer_1S ACTimer_1S_D	TIM,EN,CV,PV,TUP,NUP	
RSWDT		本指令無運算元
WDT	N	N：監控定時器之設定時間。 其值祇能為 50、 60、 70… …120， 單位為 10mS，即設定之時間範圍為 (50 ~ 120)×10mS， 即 0.5 秒 ~ 1.2 秒
HSCTR	CN	CN：硬體高速計數器號碼 0：HSC0 1：HSC1 2：HSC2 3：

		HSC3 4：HSC4 5：HSC5 6：HSC6 7：HSC7
HSCTW	S,CN,D	CN：硬體高速計數器號碼 0：HSC0 1：HSC1 2：HSC2 3：HSC3 4：HST4 2：HSC2 3：HSC3 4：HST4 D：寫入對象（0：表示 CV，1：表示 PV）
RAMP2	EN,Om,Ta,Td,Rt,Rc,WR,ACC,DEC	OM：最大輸出設定；設定範圍 0~65535 TA ：由 0 到最大輸出之上升時間設定； 設定範圍 0~65000，單位為 MS TD ：由最大輸出到 0 之下降時間設定； 設定範圍 0~65000，單位為 MS RT ：目標輸出量設定暫存器； 設定範圍 0~65535 RC ：目前輸出量暫存器，用來作 D/A 輸出 WR：工作暫存器起始位址，共佔用 4 個暫存器
CLINK	EN,PAU,ABT,Pt,MD,WR,ERR,DN	Pt ：指定通訊埠，1、2 MD：通訊模式選擇(MD0~MD3) SR ：存放通訊程式起始暫存器 WR：指令運作起始暫存器，共佔用 8 個暫存器，其它程式不 可重複使用。
CMCTL	EN,PAU,ID,Pt,Ts,MD,WR,ERR	ID:使用的模組編號 Pt:指定 COMA/COMB (A= 0;B=1) Ts:通訊表格遮罩 Bit 1:表格 0 ，Bit 2:表格 1 ~ Bit 15:表格 15 Bit 16~Bit 31: 保留，不可使用 MD:設定模式 0:RUN ONCE 1:RUN CYCLING 2:STOP WR:存放工作狀態 Bit0~Bit1:表格 0 state Bit2~Bit3:表格 1 state ~ Bit30~Bit31:表格 15 state (= 0:RUN_ONCE = 1:RUN_CYCLING = 2:STOP)
ModBUS	EN,A_R,ABT,PT,SRWR,ACT,ERR,DN,	PT ：1、2，透過該通訊埠,以 MODBUS RTU 通訊協定作資料傳輸 SR ：通訊程式起始暫存器（見範例說明） WR ：指令運作起始暫存器（見範例說明），共佔用 8 個暫存器，其它程式不可重複使用
NCR	NE,SR,MD,WR,ACT,ERR,DN	SR: 表格起始暫存器位址。 MD: Modbus TCP 主動通訊(=1) 。 WR: 工作暫存器。

BKCMP BKCMP_D	Rs,Ts,L,D	RS ：被比較之資料或其暫存器號碼 TS ：上下限值暫存器區塊之起頭號碼 L ：上下限值之組數 D ：存放比較結果之繼電器起頭號碼
BT_M BT_M_D	Ts,Td,L	TS ：來源列表之起頭暫存器號碼 TD ：目的列表之起頭暫存器號碼 L ：來源和目的列表之長度 TS，TD 可結合 V、Z、P0~P9 作間接定址應用
QUEUE QUEUE_D	InOut,IW,Qu, L,Pr,OW,EPT, FUL	IW ：擠入貯列之資料或其暫存器號碼 QU ：貯列之起頭暫存器號碼 L ：貯列之長度 PR ：指標暫存器號碼 OW ：接收自貯存器移出資料之暫存器號碼 QU 可結合 V、Z、P0~P9 作間接定址應用
T_Search T_Search_D	FHD,DS,Rs,Ts ,L,Pr,FND,EN D,ERR	Rs：找尋之來源（樣本）資料或其暫存器號碼 Ts：被找尋之列表起頭暫存器號碼 L：列表長度 Pr：指標，用以記錄目標所在之位置值 Rs，Ts 可結合 V、Z、P0~P9 作間接定址應用
SORT SORT_D	AD,S,D,L	S ：欲排序之來源資料起頭暫存器號碼 D ：排序後之資料起頭暫存器號碼 L ：排序之資料長度
STACK STACK_D	InOut,IW,ST, L,Pr, OW,EPT,FUL	IW ：塞入堆疊之資料或其暫存器號碼 ST ：堆疊之起頭暫存器號碼 L ：堆疊之長度 PR ：指標暫存器號碼 OW ：接收堆疊移出資料之暫存器號碼 ST 可結合 V、Z、P0~P9 作間接定址應用
T_Compare T_Compare_D	FHD,DS,Ta,T b,L,Pr,FND,E ND,ERR	Ta：列表 a 之暫存器起頭號碼 Tb：列表 b 之暫存器起頭號碼 L：列表 a 和 b 之長度 Pr：指標，用以記錄目標所在之位置值 Ta，Tb 可結合 V、Z、P0~P9 作間接定址應用

T_Fill T_Fill_D	Rs,Td,L	RS：欲填入列表之來源資料或其暫存器號碼 TD ：列表之起頭暫存器號碼 L ：列表之長度 RS，TD 可結合 V、Z、P0~P9 作間接定址應用
ZoneWR	Val,D,N	D ：欲寫入或清除區域之起始位址 N ：欲寫入或清除區域之長度:1~511 D、N 可結合 V、Z、P0~P9 作間接定址應用
MAND	Ma,Mb,Md,L	Ma：來源矩陣 a 之起頭暫存器號碼 Mb：來源矩陣 b 之起頭暫存器號碼 Md：存放結果之目的矩陣起頭暫存器號碼 L：矩陣（Ma、Mb 和 Md）之長度 Ma，Mb，Md 可結合 V、Z、P0~P9 作間接定址應用
MBCNT	OnOff,Ms,L, D	MS ：矩陣之起頭暫存器號碼 L ：矩陣之長度 D ：存放數量結果之暫存器號碼 MS 可結合 V、Z、P0~P9 作間接定址應用
MBRD	EN,INC,CLR, Ms,L, Pr,OTB,END,E RR	Ms：矩陣之起頭暫存器號碼 L ：矩陣之長度 Pr ：指標暫存器號碼 Ms 可結合 V、 Z、 P0~P9 作間接定 址應用
MBROT	L_R,Ms,Md,L, OTB	Ms ：來源矩陣之起頭暫存器號碼 Md ：目的矩陣之起頭暫存器號碼 L ：矩陣（ Ms 和 Md）長度 MS， Md 可結合 V、 Z、 P0~P9 作間接 定址應用
MBSHF	INB,L_R,Ms, Md,L,OTB	Ms ：來源矩陣之起頭暫存器號碼 Md ：目的矩陣之起頭暫存器號碼 L ：矩陣（ Ms 和 Md）長度 Ms， Md 可結合 V、 Z、 P0~P9 作間接 定址應用
MINV	Ms,Md,L	Ms ：來源矩陣之起頭暫存器號碼 Md ：存放結果之目的矩陣起頭暫存器號碼 L ：矩陣（ Ms 和 Md）之長度 Ms，Md 可結合 V、Z、 P0~P9 作間接定址應 用
MOR	Ma,Mb,Md,L	Ma：來源矩陣 a 之起頭暫存器號碼

		Mb：來源矩陣 b 之起頭暫存器號碼 Md：存放結果之目的矩陣起頭暫存器號碼 L ：矩陣（ Ma、 Mb 和 Md）之長度 Ma， Mb， Md 可結合 V、 Z、 P0~P9 作間接 定址應用
MXNR	Ma,Mb,Md,L	Ma ：來源矩陣 a 之起頭暫存器號碼 Mb ：來源矩陣 b 之起頭暫存器號碼 Md ：存放結果之目的矩陣起頭暫存器號碼 L ：矩陣（ Ma， Mb， Md）之長度 Ma， Mb， Md 可結合 V、 Z、 P0~P9 作間接 定址應用
MXOR	Ma,Mb,Md,L	Ma ：來源矩陣 a 之起頭暫存器號碼 Mb ：來源矩陣 b 之起頭暫存器號碼 Md ：存放結果之目的矩陣起頭暫存器號碼 L ：矩陣（ Ma， Mb， Md）之長度 Ma， Mb， Md 可結合 V、 Z、 P0~P9 作間接 定址應用
HSPSO	EN,PAU,ABT, Ps,SR,WR,AC T,ERR,DN	Ps ：第幾組 Pulse 輸出 （0～7） 0：Y0 & Y1 1：Y2 & Y3 2：Y4 & Y5 3：Y6 & Y7 4：Y8 & Y9 5： Y10 & Y11 6：Y12 & Y13 7：Y14 & Y15 SR ：定位程式起始暫存器 WR：指令運作起始暫存器，共佔用 7 個暫存器，其它程式 不可重覆使用
HSPWM	EN,Pw,Op,Rs ,Pn, OutR,WR,AC T	Pw： 高速脈波寬度調變輸出點 (0=Y0, 1=Y2, 2=Y4, 3=Y6, 4=Y8, 5=Y10, 6=Y12, 7=Y14) Op： 輸出極性； 0=輸出不倒相 1=輸出倒相 Rs： 解析度； 0=1/100 (1%) 1=1/1000 (0.1%) Pn： 輸出頻率參數設定(0~255) OR : PWM 輸出寬度設定暫存器 0~100 或 0~1000 WR： 指令運作工作暫存器，其他程式不可重複使用
HSPWM2	EN,Pw,Op,Hz ,O_R, ACT,ERR	PW：脈波寬度調變輸出點(0=Y0, 1=Y2, 2=Y4, 3=Y6, 4=Y8, 5=Y10, 6=Y12, 7=Y14) Op：輸出極性(0=正相, 1=倒相) Hz：輸出頻率(1~100000000 or 1~200000000，單位 0.001Hz) OR：脈波輸出寬度(0~100，單位%)
ICA	EN,D_R,Ps,Is, Fo,Ag,ACT,E	Ps：第幾組 Pulse 輸出 （0～7） 0：Y0 & Y1 1：Y2 & Y3 2：

	RR,DN	Y4 & Y5 3：Y6 & Y7 4：Y8 & Y9 5：Y10 & Y11 6：Y12 & Y13 7：Y14 & Y15 Ls：外界輸入 X 點索引號碼(0~15) Fo：目標工作速度(1~100000 or 1~200000) Ag：中斷時固定角度(0~36000)
ICF	EN,D_R,Ps,Is,Fo,Fd,ACT,ERR,DN	Ps：第幾組 Pulse 輸出 （0～7） 0：Y0 & Y1 1：Y2 & Y3 2：Y4 & Y5 3：Y6 & Y7 4：Y8 & Y9 5：Y10 & Y11 6：Y12 & Y13 7：Y14 & Y15 Ls：外界輸入 X 點索引號碼(0~15) Fo：目標工作速度(1~100000 or 1~200000) Fd：中斷捕捉後輸出之脈波移動量
MHSPO	EN,PAU,ABT,Gp,SR,WR,ACT,ERR,DN	Gp ：第幾個群組（0～1） SR ：定位程式起始暫存器 WR：指令運作起始暫存器，共佔用 9 個暫存器，其它程式不可重覆使用
MPARA	EN,Ps,SR	Ps ：第幾組 Pulse Output（0～3） SR ：參數表起始暫存器，共 18 個參數，佔用 24 個暫存器
MPG	EN,Sc,Ps,Fo,Mr,WR,ACT	Sc ：指定接手搖輪之來源高速計數器； 0~7 Ps ：指定反應手搖輪之脈波輸出軸； 0~3 Fo ：輸出頻率設定暫存器 (2 個暫存器) Mr ：倍率設定暫存器 (2 個暫存器) Mr+0：倍率被乘數 (Fa) Mr+1：倍率被除數 (Fb) WR：工作暫存器起始位址，共佔用 4 個暫存器 .輸出脈波數=(手搖輪輸入脈波數×Fa)/Fb * PLC OS V4.60(含)以後支援此命令
PSCNV	EN,Ps,D	Ps：0～3；將第幾組脈波位置（PS）轉換為與設定值同單位之 mm（Deg，Inch，PS）， 以作為目前位置顯示. D ：儲存轉換後目前位置之暫存器,共需使用二個暫存器；例如 D10，即代表 D10（Low Word）與 D11（High Word）二個暫存器。
PSOFF	EN,Ps	Ps：0～3 強制第幾組 Pulse Output 停止輸出
INTDisable	LB	LBL：禁止作動之外界輸入或週邊之中斷標記名稱。

INTEnable	LB	LBL：允許中斷作動之外界輸入或週邊標記名稱。
TARESUB	EN,RST,ID,CH,SB, ERR	ID:擴充模組 ID 編號(0 ~ N) CH:通道索引(0 ~ N) SB: 為欲扣除之皮重 32BIT 數值
TAREZEOFFSET	EN,MD,ID,CH,WR,ERR	
MFSysCAMR	EN,Md,D,ID,L,ACT,ERR,DN	讀取凸輪資料 路線 1：從 PLC Md：0 D：凸輪資料讀取後放在 PLC 暫存器的起始位址 ID：凸輪編號 L：凸輪解析度 路線 2：從 SD 卡 Md：1 D：SD 卡凸輪資料檔案編號 ID：凸輪編號 L：凸輪解析度
MFSysCAMW	EN,Md,D,ID,L,ACT,ERR,DN	寫入凸輪資料 路線 1：從 PLC Md：0 D：寫入的凸輪資料在 PLC 暫存器的起始位址 ID：凸輪編號 L：凸輪解析度 路線 2：從 SD 卡 Md：1 D：SD 卡凸輪資料檔案編號 ID：凸輪編號 L：凸輪解析度
MFSysInit	EN,ACT,ERR,DN	啟用 Motion 功能 / 開啟驅動器連線
MFSysRCPR	EN,Md,D,Gp,ACT,ERR,DN	按照運動配方設定的內容讀取配方 路線 1：從 PLC Md：0 D：無意義 Gp：讀取所有設定配方 = 0，讀取第一項設定配方 = 1，以此類推 路線 2：從 SD 卡

		Md：1 D：SD 卡配方檔案編號 Gp：讀取所有設定配方 = 0，讀取第一項設定配方 = 1，以此類推
MFSysRCPW	EN,MD,D,GP, ACT, ERR,DN	按照運動配方設定的內容寫入配方 路線 1：從 PLC Md：0 D：無意義 Gp：寫入所有設定配方 = 0，寫入第一項設定配方 = 1，以此類推 路線 2：從 SD 卡 Md：1 D：SD 卡配方檔案編號 Gp：寫入所有設定配方 = 0，寫入第一項設定配方 = 1，以此類推
MFSysRstAlm	EN,ACT,ERR, DN	重製運動控制錯誤
MFSysSetVirt	EN,AX,ACT,E RR,DN	實體軸轉虛擬軸 AX：軸編號
MFSysStop	EN,ACT,ERR, DN	停止 Motion 功能 / 關閉驅動器連線
MFChgTbPrm	EN,GP,ID,N,D ,ERR,DN	單筆資料映射 TM：表格類型 PN：表格編號 S：項目 PV：映射值
MFFlowHalt	EN,ID,ACT,ER R,DN	立刻暫停運動流程 ID：運動流程編號
MFFlowPause	EN,ID,ACT,ER R,DN	完成當前動作塊後，暫停運動流程 ID：運動流程編號
MFFlowStart	EN,ID,ACT,ER R,DN	啟動運動流程 ID：運動流程編號
MFFlowStop	EN,ID,ACT,ER R,DN	中止運動流程 ID：運動流程編號
MFMapTbPrm	EN,GP,ID,N,D ,ERR,DN	按照運動參數映射表映射資料 Gp：映射表分頁編號

		N：起始項目編號 L：映射項目數量
MFHome	EN,AX,ACT,ERR, DN	執行軸回原點模式 AX：軸編號
MFJog	EN,D_R, AX, MD,ACT,ERR,DN	執行吋動模式 AX：軸編號 MD：以啟動速度連續運動 (0)，以啟動速度吋動 (1)，以 JOG 速度連續運動 (2)，以 JOG 速度吋動 (3) D/R：正方向 (ON) / 負方向 (OFF)
MFPointMov	EN,PT,AX,ACT,ERR,DN	執行軸回原點模式 AX：軸編號
MFAxCirMov	EN,UPD,TAX, MD, DR,BF,D,EC,ACT,ERR,DN,UPD	總軸數 (TAX)： 模式 (MD)： 0: 絕對, 1: 相對 方向 (DR)： 1: 正向, 2: 反向
MFAxLMov	EN,UPD,TAX, MD, DR,BF,D,EC,ACT, ERR,DN,UPD	連續模式 (BF)： 0 = 立即執行當前指令 1 = 等待前一指令結束 2 = 選擇較低速度連續 3 = 選擇前一指令速度 4 = 選擇當前指令速度 5 = 選擇較高速度連續 點資料 (D)： 暫存器起始位置 錯誤碼 (EC)： 暫存器儲存位置
MFAxMov MFAxMov_D	EN,UPD_in,AX,MD,Ps,V_n,A,D,SA,SD,DR,BF,ACT,ERR,DN,UPD_out	執行單軸定位模式 S：主軸編號 MD：絕對座標 (0) / 相對座標 (1) / 無限距離模式 (2) Ps：目標座標 V：最大線速度 A：最大加速度 D：最大減速度 SA：加速段 S 曲線比例 (單位：0.1%) SD：減速段 S 曲線比例 (單位：0.1%) DR：正方向 (1) / 負方向 (2) BF：連續點模式

MFGearMPG MFGearMPG_D	EN,UPD_in, M,S,N,D,T,ACT,ERR,DN,UPD_out	M: (主軸) 1~16: 軸編號 100: 格雷碼 (X8~15) 101~108: HSC0~7 S: (從軸) 1~16: 軸編號 N: (轉換分子) 整數，包含從軸小數點 D: (轉換分母) 正整數，包含主軸小數點 T: (平滑時間) 正整數 單位: ms
MFPathMov	EN,ACT,UPD, AX1,AX2,AX3,PT,V,TA,TD, SA,SD,EC,ACT,ERR,DN,UPD	AX1: 軸 1 AX2: 軸 2 AX3: 軸 3 PT: 路徑編號 V: Vel 正整數 包含主軸小數點位 TA, TD: 加速時間 正整數 (ms) SA, SD: (S 比例 %) 0.0% ~ 100.0% 單位: 0.1% EC: 暫存器儲存位置
MFTorqCtl MFTorqCtl_D	EN,UPD_in,AX,T,MX,ACT,ERR,DN, UPD_out	AX: 1~16: 軸編號 T: 單位: 0.1% MX: 最大轉速
MFVelCtl MFVelCtl_D	EN,UPD_in,AX,T,MX,ACT,ERR,DN, UPD_out	AX: 1~16: 軸編號 V : 單位: 看驅動器 MX:

		最大扭矩