FATEK

M Series

Programmable Controller

Function/Function Block 使用手冊

NEXT Level SOLUTION

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 http://www.fatek.com 的技術支援專區。

永宏電機股份有限公司

目錄

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。

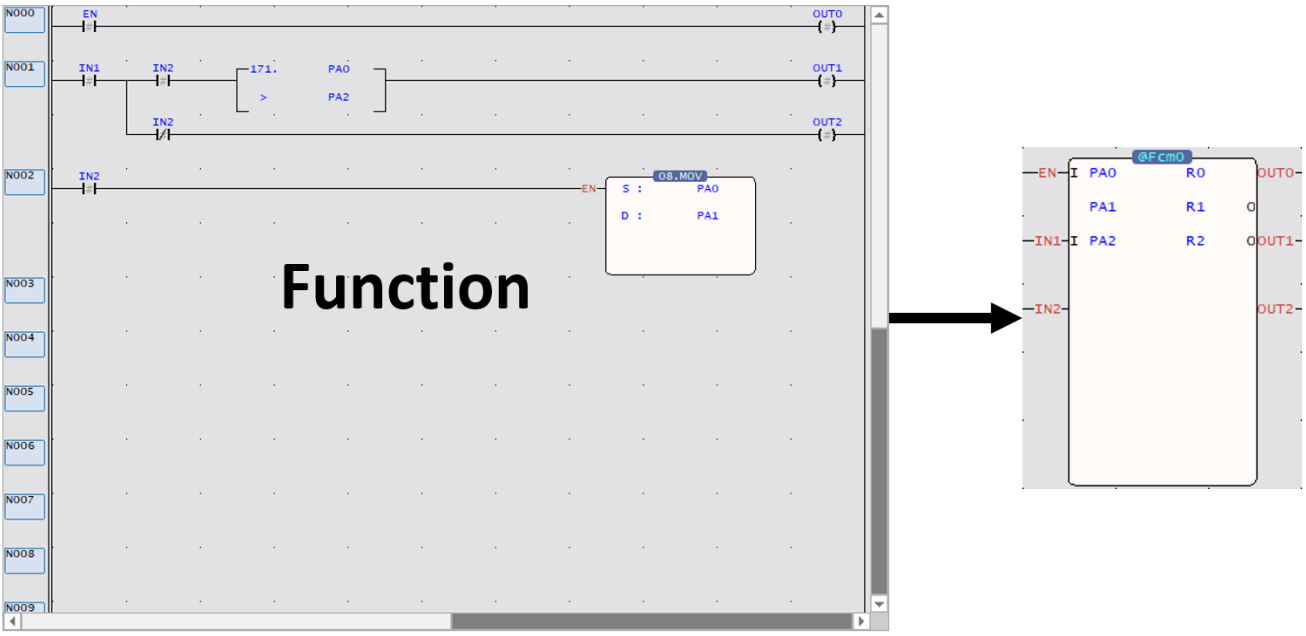
規格變更
產品規格和附件可能會因技術改進或其它原因而隨時變更。當公佈的規格、性能改變，或者進行過重大的結構改變時，FATEK 通常會改變型號。若產品的某些規格發生變更時，以下情況不另行通知：根據客戶的要求，對客戶的應用指定特別的型號或設定特定的規格。歡迎隨時洽詢 FATEK 代理經銷商，確認所購產品的實際規格。

1

使用 Function/Function Block 的好處

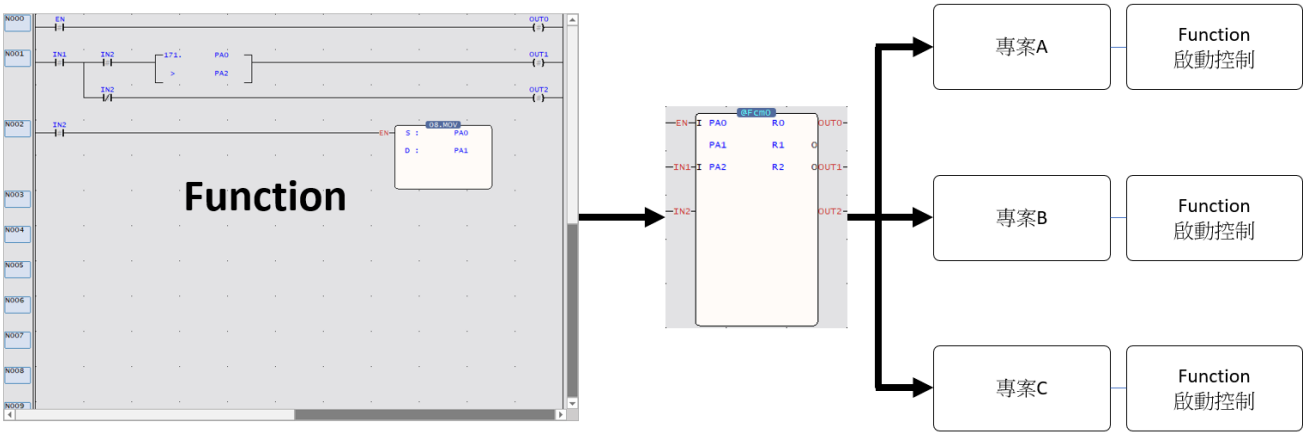
1-1 易於閱讀

Function/Function Block(使用 FCM)能夠將複雜的程式包裝成一個 Function 指令，使用者只需要了解 Function 指令輸入輸出的動作，而不需要深入研究 Function 指令中的程式內容如何達成特定的功能。



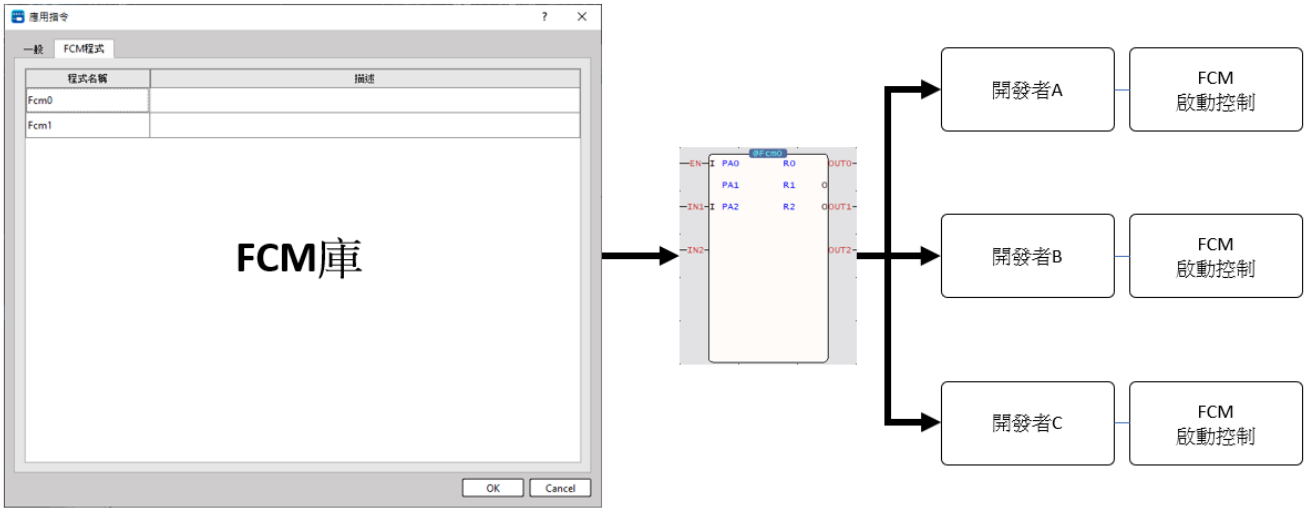
1-2 重複使用

Function/Function Block(使用 FCM)提供使用者將程式包裝成一個 Function 指令，重複使用 Function 指令不需要對 Function/Function Block(使用 FCM)的內部暫存器重新編號，如果累積了大量的 Function/Function Block(使用 FCM)，對於不同專案也能重複利用，加速使用者對專案的編程。



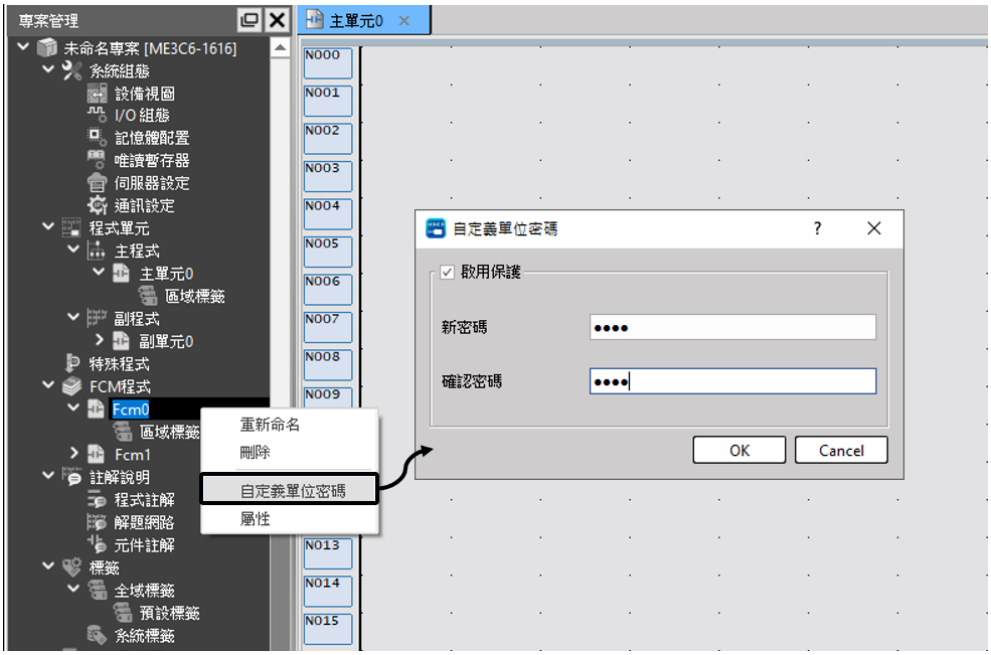
1-3 提高編程品質

前面提到 Function/Function Block(使用 FCM)能夠重複使用，在專案開發者之間共享 FCM 庫，可以提高因為不同開發者編輯造成的功能穩定上的差異。



1-4 程式保護

對 Function/Function Block(使用 FCM)設定自定義單位密碼，能夠在共享 Function/Function Block 指令時有效保護 Function/Function Block 指令不被其他使用者任意查看和編輯內容，進而保護程式設計者的智慧財產。



2

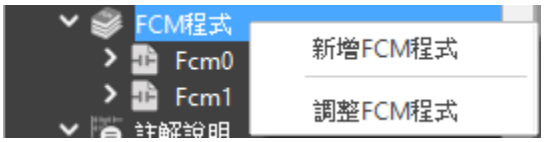
FCM 程式

本章節將介紹如何在程式單元中新增並使用 FCM 程式。

2-1 FCM 庫

新增的 FCM 程式儲存在【專案管理】>【程式單元】>【FCM 程式】。跟程式單元一樣能夠調整順序,重新命名和自定義密碼。FCM 程式有別於一般程式單元，FCM 程式需要設定屬性。(FCM 屬性請參考 3-4 章節)

在【FCM 程式】使用滑鼠右鍵點擊能夠【新增 FCM 程式】或是【調整 FCM 程式】順序。



在新增的【FCM 程式單元】使用滑鼠右鍵點擊能夠【重新命名】,【刪除】,【自定義單元密碼】,【屬性】設定。

2-2 匯入/匯出 FCM 程式

透過 UperLogic，使用者可以將編寫好的 FCM 程式匯入或匯出，在不同專案中重複利用。

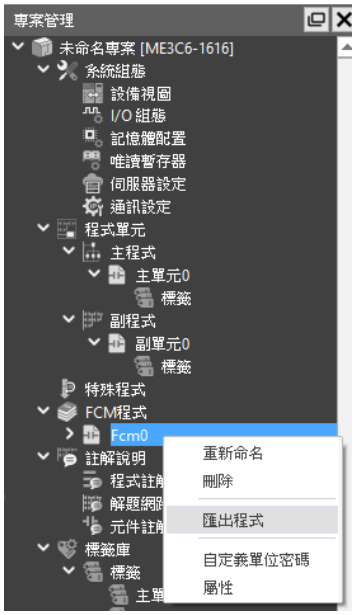
匯入:

於【專案】>【程式單元】>【FCM 程式】>【匯入程式】。

或於【專案管理】>【FCM 程式】使用滑鼠右鍵點擊>【匯入程式】。

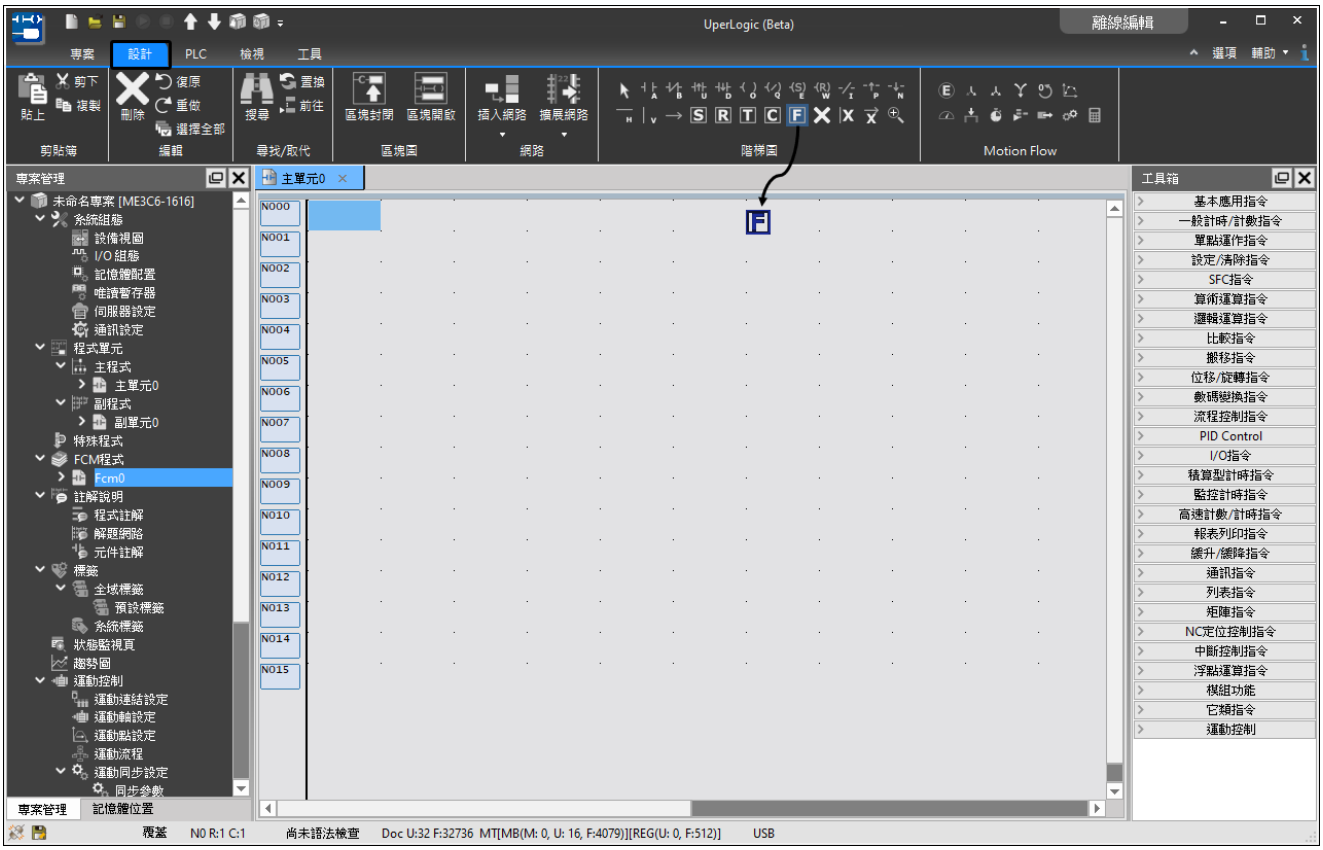
匯出:

於【專案管理】>【FCM 程式】>對欲匯出的 FCM 程式使用滑鼠右鍵點擊>【匯出程式】。

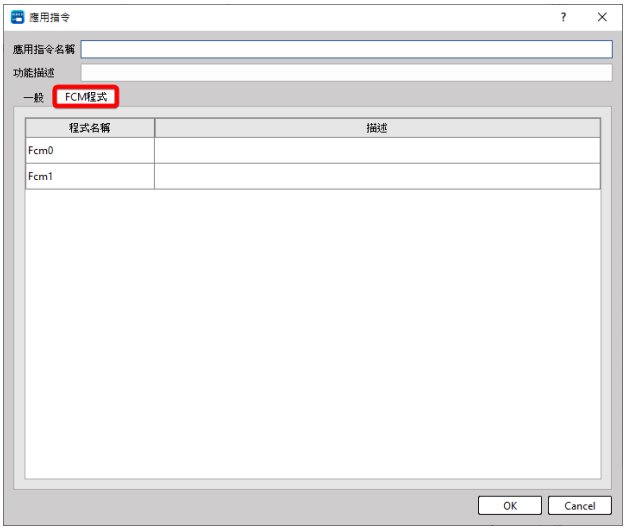


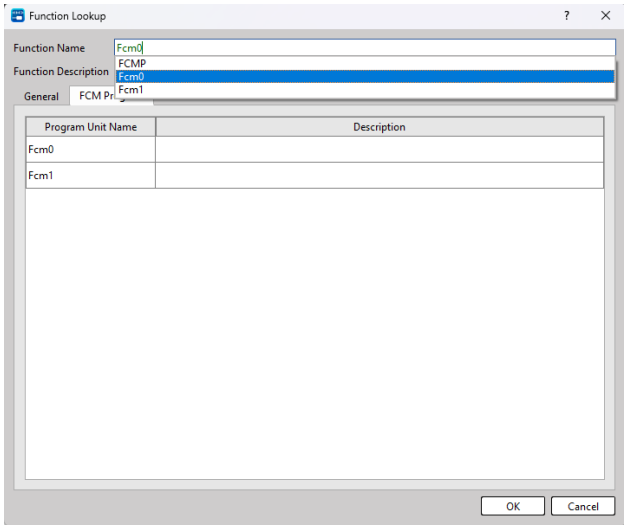
2-3 Function/Function Block 使用方式

Step 1. 使用【設計】>【階梯圖】>按鈕【F】>新增到程式單元編輯畫面中，或是在程式單元編輯畫面中使用快捷鍵” F” 呼叫【應用指令】庫

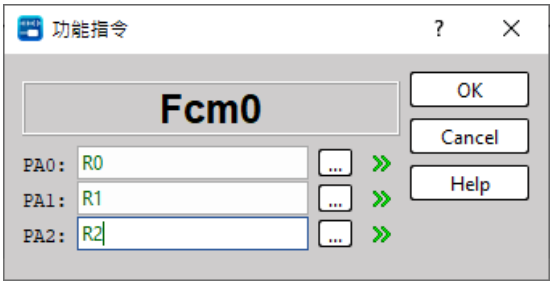


Step 2. 接著彈出【應用指令】視窗 > 切換到 FCM 程式分頁 > 選擇要新增到程式單元中的 FCM 程式(下圖中有 2 個 FCM 程式)，或直接於應用指令名稱欄位輸入 FCM 程式名稱。

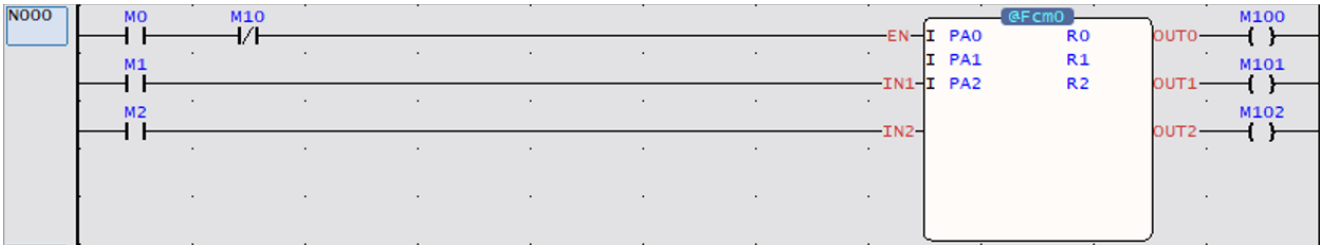




Step 3.　選擇 FCM 程式後，自動彈出 FCM 功能指令設定視窗，跟一般的 Function 指令一樣，在參數中填入暫存器或常數。



Step 3.　FCM 程式跟一般 Function 指令一樣，使用 A 接點和 B 接點...等元件監控輸入腳位和輸出腳位。



3

Function/Function Block 程式

本章節將介紹如何在 UperLoigc 中新增 FCM 程式,設定 FCM 參數以及屬性。

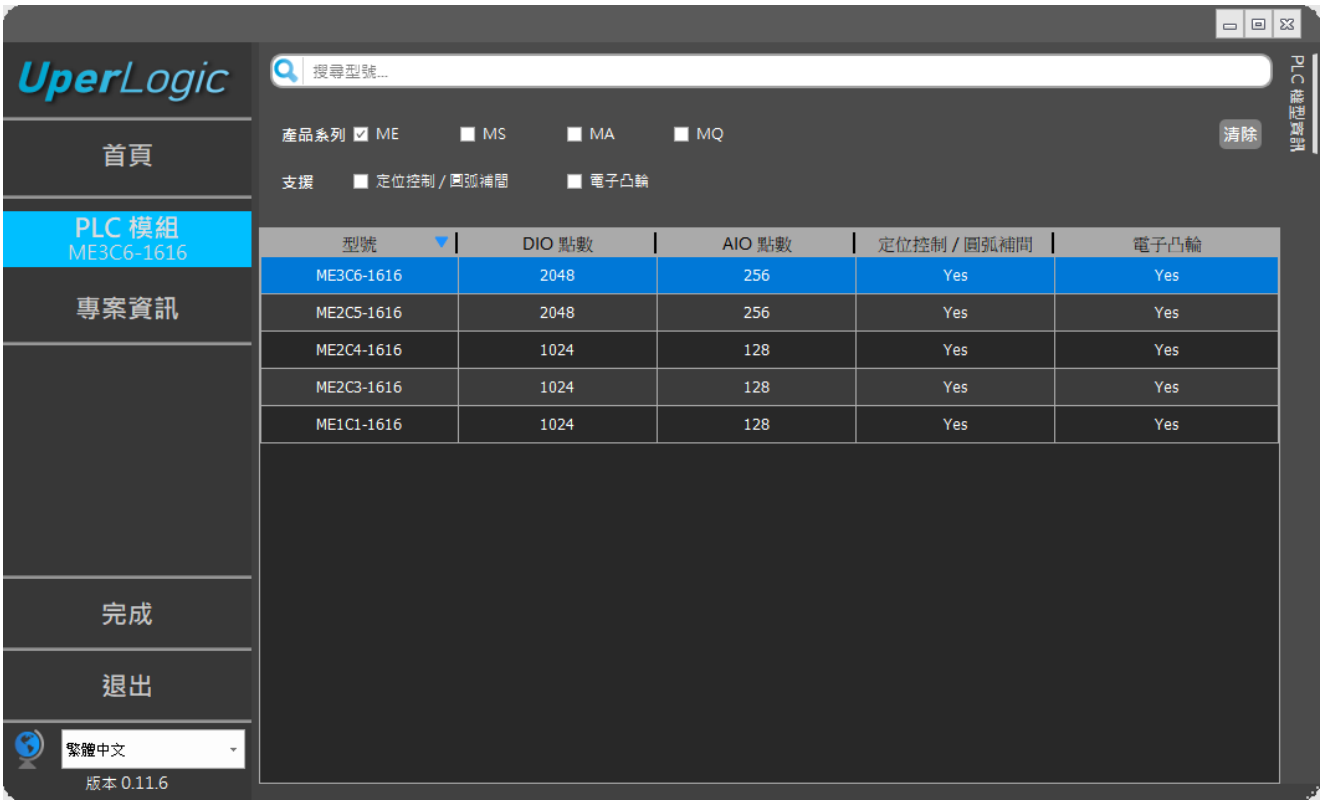
3-1 開啟新專案

Step 1. 選擇 UperLogic 左上角【菜單】 > 【開新專案】

Step 2. 接著彈出 PLC 模組選型視窗

① 選擇產品系列 (下圖選擇 ME 系列)

② 選擇型號 (下圖選擇 ME3C6-1616)

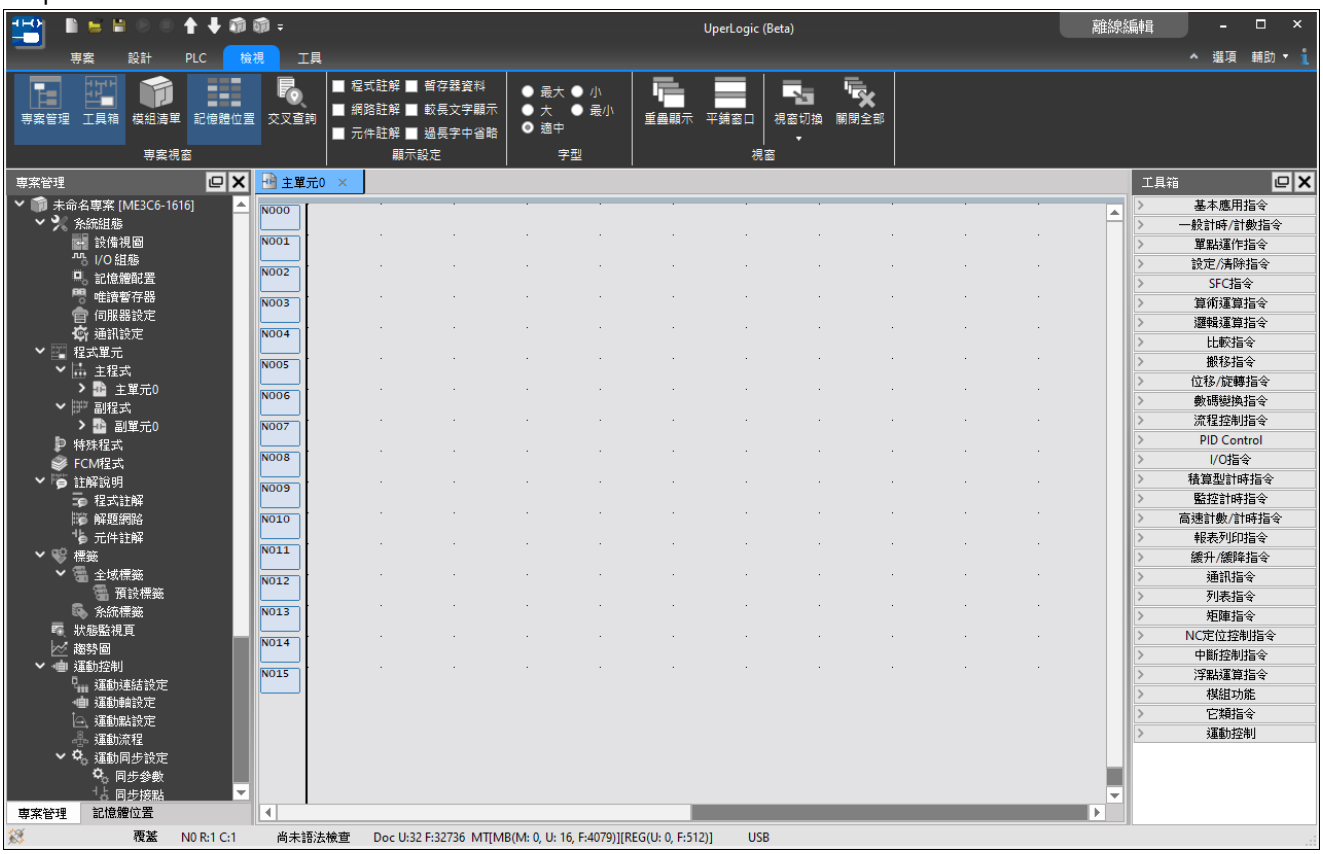


Step 3.　設定專案資訊

③　選擇常用程式語言(下圖選擇階梯圖 LD)

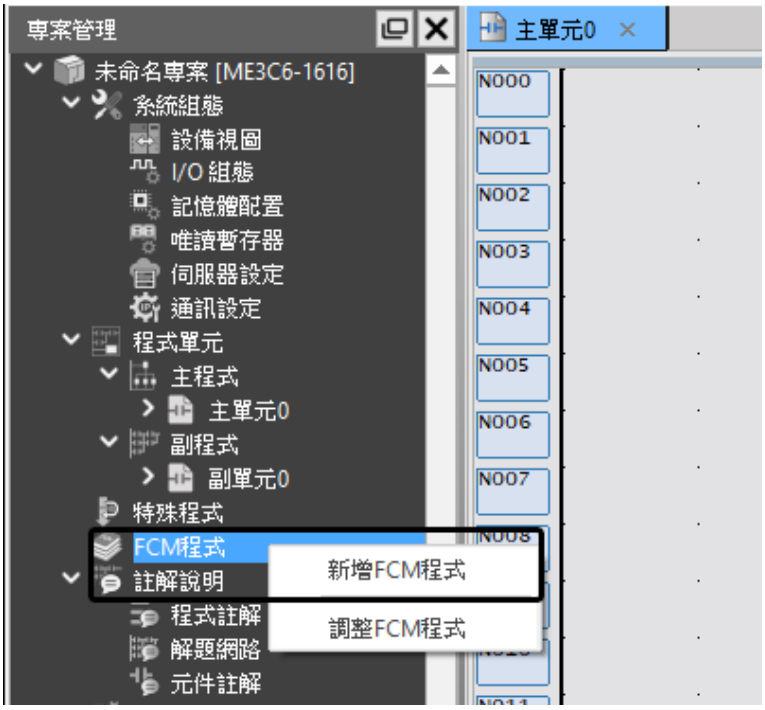
①　完成

Step 4.　確認完成後會顯示主程式單元編輯畫面

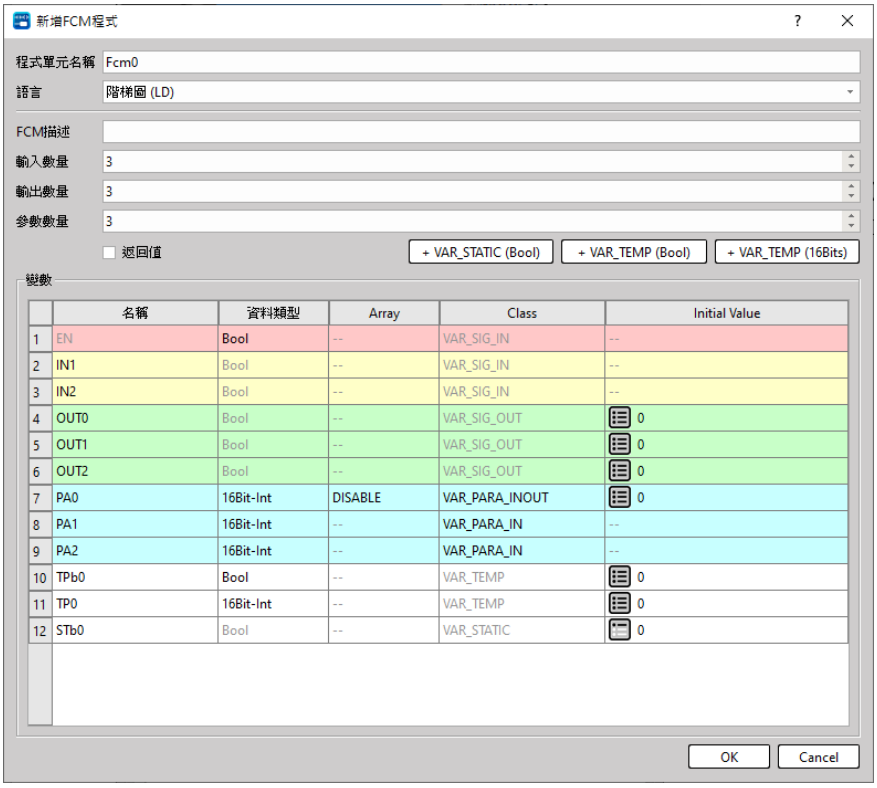


3-2 新增 FCM

Step 1. 選擇【UperLogic】左邊的【專案管理】>【程式單元】>【FCM 程式】>點擊滑鼠右鍵>【新增 FCM 程式】



Step 2. 接著彈出【FCM 屬性】設定視窗 > 設定【FCM 屬性】 > 設定完成後按下【OK】來完成新增 FCM 程式。(FCM 屬性請參考 3-4 章節)



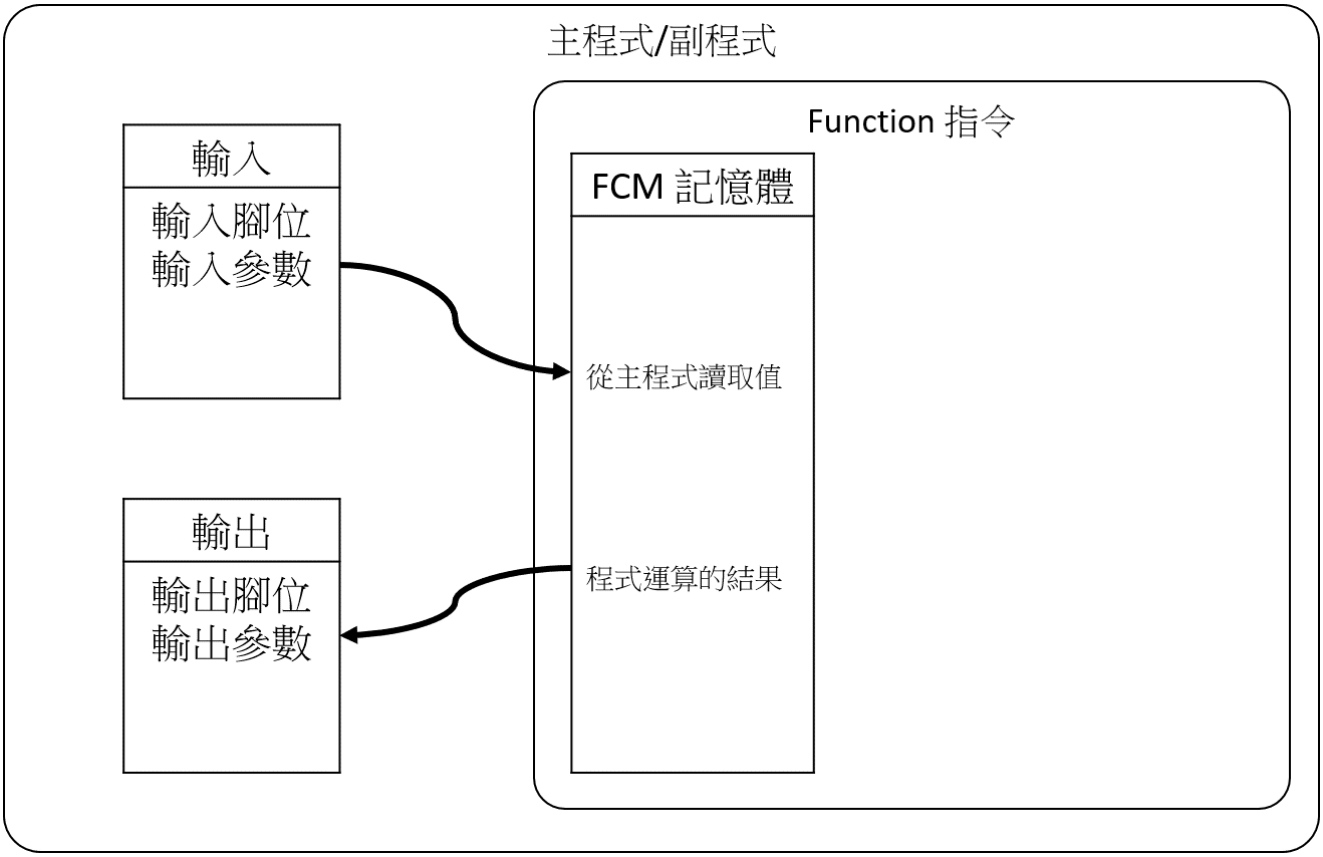
3-3 FCM 內部變數

FCM 擁有獨立的內部繼電器和內部暫存器，內部變數只能作為 FCM 程式運算使用。部分的內部繼電器和內部暫存器可以跟外面的程式單元暫存器作交換資料。這些參數屬於區域變數，僅在該功能塊（FCM）內部有效。每次執行時，Function Block 都有自己的獨立變數空間，不會受到其他部分的影響，這確保了功能塊可以重複使用而不會干擾其他邏輯。

[注意]進行 FCM 功能塊封裝，需注意全域暫存器類型的使用，例如: R/D/M。避免呼叫時，在其他程式誤用與操作到重覆的暫存器位置，導致邏輯功能異常。若需完整封裝該 FCM，必須在 FCM 程式中只使用下述各類型變數。

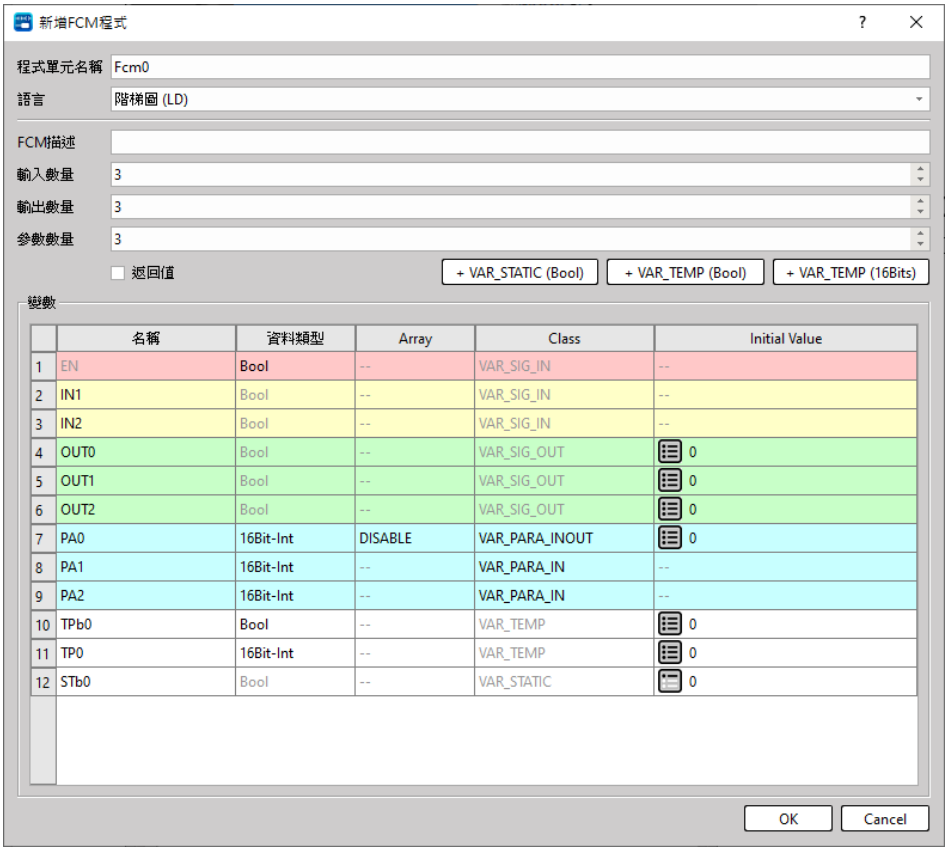
FCM 內部變數

項目		
Bit	VAR_SIG_IN	輸入腳位
	VAR_SIG_OUT	輸出腳位
	VAR_TEMP	內部自動變數
	VAR_STATIC	內部靜態變數
WORD	VAR_PARA_IN	輸入參數
	VAR_PARA_OUT	輸出參數
	VAR_PARA_INOUT	輸入輸出參數/內部靜態變數
	VAR_TEMP	內部自動變數


主程式/副程式
Function 指令
輸入
輸入腳位
輸入參數
FCM 記憶體
從主程式讀取值
輸出
輸出腳位
輸出參數
程式運算的結果

3-4 FCM 屬性

FCM 屬性



- 程式單元名稱 ：Function/Function Block 指令的名稱。(僅能輸入英文和數字)
- 語言 ： 用來編寫 Function/Function Block 指令內部程式的語言，例如 LD(Ladder),ST...等。
- 輸入數量 : Function/Function Block 指令的輸入腳位數量。
- 輸出數量 : Function/Function Block 指令的輸出線圈數量。
- 參數數量 : Function/Function Block 指令的輸入/輸出參數數量。
- +VAR_STATIC : 新增內部靜態變數
- +VAR_TEMP : 新增內部自動變數
- 變數 :設定前面設定的輸入腳位變數,輸出腳位變數, 輸入輸出參數變數和內部變數的標籤名稱和類型。

3-4-1 變數名稱

除了 EN 作為 Function 指令觸發腳位所以固定名稱，其他變數都可以修改成英文和數字混合的名稱。針對不同類型的變數使用不同的背景顏色作區分。

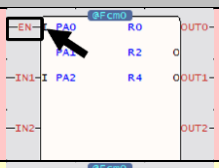
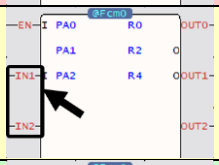
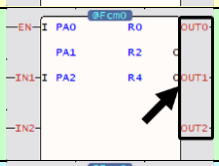
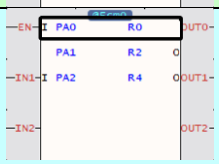
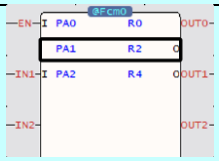
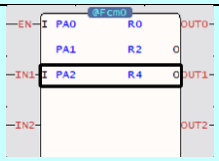
變數名稱

變數	名稱長度	背景顏色
EN	20 個英文和數字混合字元	紅色
輸入	20 個英文和數字混合字元	黃色
輸出	20 個英文和數字混合字元	綠色
參數	20 個英文和數字混合字元	藍色
內部靜態變數	20 個英文和數字混合字元	白色
內部自動變數	20 個英文和數字混合字元	白色

3-4-2 變數類別 Class

這些參數屬於區域變數，僅在該功能塊 (Function Block) 內部有效。每次執行時，FCM 都有自己的獨立變數空間，不會受到其他部分的影響，這確保了功能塊可以重複使用而不會干擾其他邏輯。如果有兩個 FCM 呼叫相同的邏輯，內部參數是不互相影響的。

Function 變數跟主程式單元交互傳輸的類別

變數	類別	說明	Function 圖示
EN	VAR_SIG_IN	程式單元觸發 FCM 執行的腳位 FCM 執行時接收程式單元給的 Bool 狀態	
輸入	VAR_SIG_IN	FCM 執行時接收程式單元給的 Bool 狀態	
輸出	VAR_SIG_OUT	輸出 FCM 的運算結果 Bool 狀態給程式單元	
參數	VAR_PARA_IN	FCM 執行時接收程式單元給的數值	 *前面標示【I】表示 IN
	VAR_PARA_OUT	輸出 FCM 運算結果給程式單元	 *後面標示【O】表示 OUT
	VAR_PARA_INOUT	FCM 執行時先接收程式單元給的數值 FCM 運算完成時輸出運算結果給程式單元	 *前面標示【I】表示 IN *後面標示【O】表示 OUT
內部靜態變數	VAR_STATIC	FCM 結束仍保留數值給下一次掃描週期的同個 FCM 執行時繼續使用	
內部自動變數	VAR_TEMP	隨 FCM 運算結束而自動消滅	

[注意]進行 FCM 功能塊封裝，需注意全域暫存器類型的使用，例如: R/D/M。避免呼叫時，在其他程式誤用與操作到重覆的暫存器位置，導致邏輯功能異常。若需完整封裝該 FCM，必須在 FCM 程式中只使用上述各類型變數。

STATIC BIT【內部靜態變數】:

- FCM 結束仍保留數值給下一個掃描週期的同個 FCM 執行時繼續使用
- 只有被執行的第一次執行初始化

VAR_STATIC BIT : VAR_STATIC (Bool)

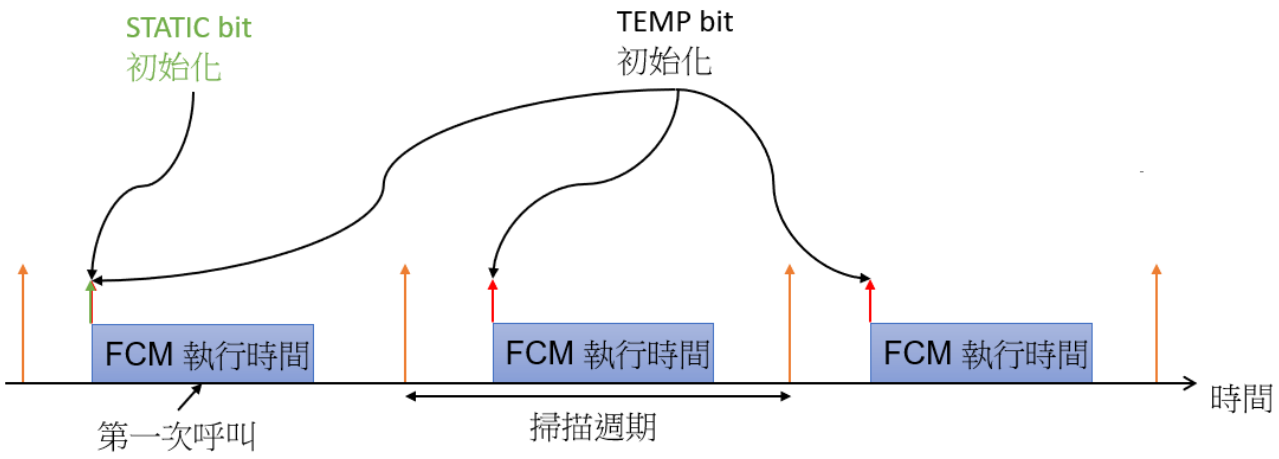
VAR_ STATIC WORD : 使用參數的 VAR_PARA_INOUT 取代

TEMP BIT【內部自動變數】:

- 隨 FCM 結束而自動消滅
- 每次執行 FCM 時都先被初始化

VAR_TEMP BIT : VAR_TEMP (Bool)

VAR_TEMP WORD : VAR_TEMP (16Bit-Int, 16Bit-Uint, 32Bit-Int, 32Bit-Uint, Float)



STATIC BIT & TEMP BIT 初始化行為時序圖

3-4-3 變數資料類型

Function 變數的資料類型

變數	資料類型	說明
EN	Bool	準位觸發
	Bool, Pulse Rising	上緣觸發
輸入	Bool	
輸出	Bool	
參數	16Bit-Int, 16Bit-Uint, 32Bit-Int, 32Bit-Uint, Float	
	Pointer	參數 Class 為 VAR_PARA_INOUT 才能選擇
內部靜態變數	Bool	
內部自動變數	Bool, 16Bit-Int, 16Bit-Uint, 32Bit-Int, 32Bit-Uint, Float	

EN 訊號觸發類型

準位觸發(Bool) : EN 訊號為 ON 時，每次掃描週期掃描到都執行 Function 指令。

上緣觸發(Bool, Pulse Rising) : EN 訊號的上升沿執行一次 Function 指令。

上緣觸發比準位觸發更有效率，因為上緣觸發僅在 EN 訊號的上升沿執行一次。這可以顯著的減少 PLC 負載。

範例: 例如 3-5 章節的 Function 範例，當使用者想要計算水位深度時，上緣觸發只需執行 Container 函數一次即可。如果使用準位觸發，Container 函數在每次掃描周期都會執行一次，在沒有改變輸入水量的情況下，不需要一直重複計算水位深度。

【指標型】與【一般型】的差異

【指標型】VAR_PARA_INOUT(資料類型為 Pointer)

1. 將參數的記憶體**位址**傳遞到 FCM 內部（傳址呼叫）。
2. 不需要初始值，因為 FCM 總是從記憶體位址讀取值。
3. FCM 程式直接改變參數的值。

*參數作為 VAR_PARA_INOUT 才能選擇指標型 Pointer。

【一般型】VAR_PARA_INOUT(資料類型為 16Bit-Int, 16Bit-Uint, 32Bit-Int, 32Bit-Uint, Float)

1. 僅將參數值複製到 FCM 內部的暫存器（傳值呼叫）。
2. 可以使用初始值或從程式單元傳遞的值進行初始化。
3. FCM 程式內部變更從參數傳遞的值不會影響外部參數的值。(FCM 運行期間不會變更外部參數的值，只有 FCM 運算完成時將結果輸出才修改外部參數的值)

【複週期指令】

複週期指令在背景跨掃描週期運行，並不以掃描週期結束而結束。 FCM 結束後 FCM 記憶體將被系統釋放。INOUT 類型參數將從這些沒被配置的記憶體中取得不確定的值。所以複週期指令要用【指標型】來跨掃描週期運行。

*請參考附錄一(複週期指令表)

3-4-4 陣列變數 Array

Array 為陣列變數，能夠將一段長度的暫存器傳遞到 Function 指令內部。

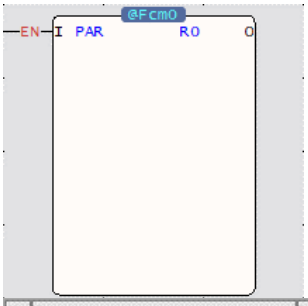
Array

變數	類別	Array	說明
參數數量	VAR_PARA_INOUT	PAR [0...199]	最大長度為 200

*除了陣列屬性，Array 跟【指標型】的行為一致。(請參考 3-4-3 變數_資料類型)

範例:

設定一個參數(VAR_PARA_INOUT)，Array 設為長度 5，Function 指令 PAR 填入 R0。



	名稱	資料類型	Array	Class	Initial Value
1	EN	Bool	--	VAR_SIG_IN	--
2	PAR	16Bit-Int	PAR[5]	VAR_PARA_INOUT	DISABLE

使用 EN 觸發 Function 指令時，Function 指令會將 R0 的數值傳遞到 Function 指令內部的 PAR[0]，R1 的數值傳遞到 Function 指令內部的 PAR[1] ，R2 的數值傳遞到 Function 指令內部的 PAR[2]...依此類推。因為是 VAR_PARA_INOUT 類別，所以 Function 指令運算完成時，會將 Function 指令內部的 PAR[0]傳出回 R0。

EN 觸發 Function 指令時，將 ARRAY 傳入 Function。

主程式		Function 指令
R0	→	PAR[0]
R1	→	PAR[1]
R2	→	PAR[2]
R3	→	PAR[3]
R4	→	PAR[4]

Function 指令運算完成時，將 ARRAY 傳出 Function。

主程式		Function 指令
R0	←	PAR[0]
R1	←	PAR[1]
R2	←	PAR[2]
R3	←	PAR[3]
R4	←	PAR[4]

3-4-5 變數初始值 Variable Initial Value

FCM 變數初始值的設定

- 輸入變數不需要有初始值，因為主程式單元總是傳遞一個數值給輸入變數（暫存器或常數）
- 輸出變數必須使用初始值進行初始化
- 輸入輸出變數可以用主程式傳遞的數值或使用初始值進行初始化
- 靜態變數只在第一次執行 FCM 時使用初始值初始化

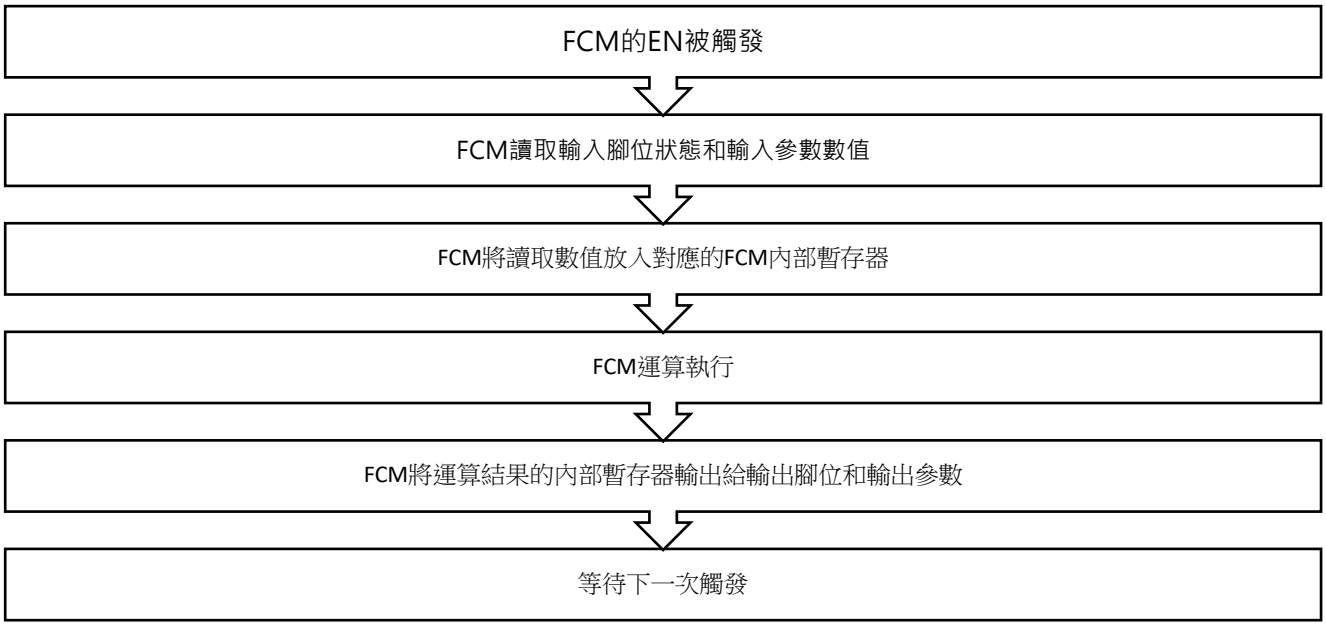
變數初始值

變數	類別	初始值圖示	初始化的數值
EN	VAR_SIG_IN	--	
輸入	VAR_SIG_IN	--	
輸出	VAR_SIG_OUT	 每次執行 FCM 都初始化	依據變數的資料類型決定初始值的資料類型
參數	VAR_PARA_IN	--	
	VAR_PARA_OUT	 每次執行 FCM 都初始化	依據變數的資料類型決定初始值的資料類型
	VAR_PARA_INOUT	 每次執行 FCM 都初始化	依據變數的資料類型決定初始值的資料類型或是 關閉初始化功能由主程式單元傳遞的數值進行初始化 *填入空白值表示[DISABLE], 不進行初始化。
內部靜態變數	VAR_STATIC	 只有第一次執行 FCM 時初始化	依據變數的資料類型決定初始值的資料類型
內部自動變數	VAR_TEMP	 每次執行 FCM 都初始化	依據變數的資料類型決定初始值的資料類型

3-5 Function/Function Block 指令(使用 FCM)運作方式

FCM 指令基本跟一般 Function 指令一樣，透過不同條件觸發 EN 腳位執行 FCM 指令，FCM 指令會依據輸入腳位狀態和輸入參數數值執行 FCM 內部的 Ladder 程式，接著將運算結果放到輸出參數和輸出腳位，給程式單元做後續的控制。

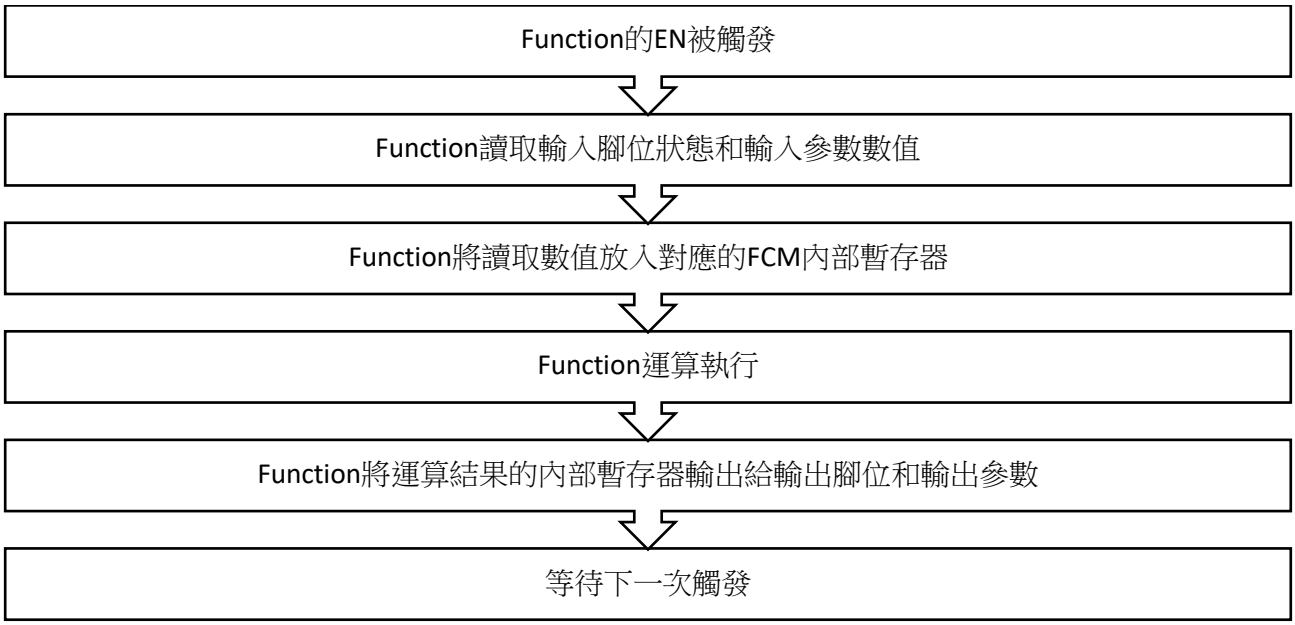
*PA0~PA2 依據 FCM 的屬性設定可以作為輸入參數或輸出參數，或是同時作為輸入參數和輸出參數。



3-5-1 Function Block vs Function

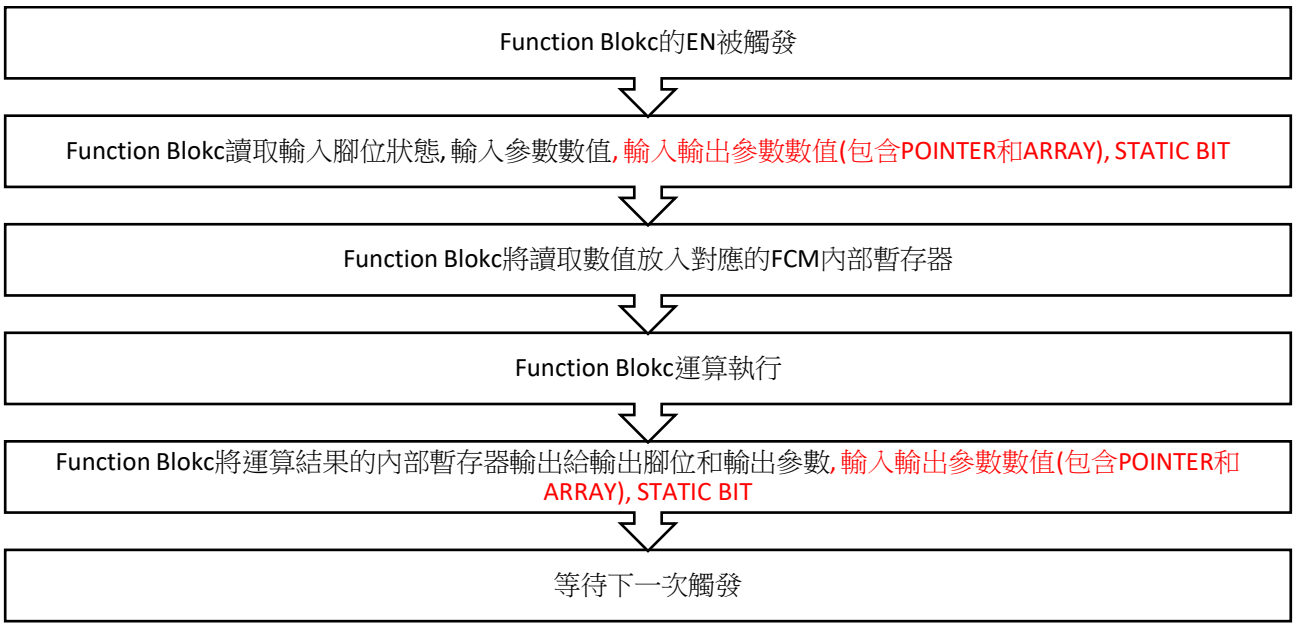
Function

Function 沒有專用記憶體的程式區塊，每次執行都是從初始值開始。



Function Block

Function Block 能夠將其值永久儲存在專用記憶體的程式區塊，因此在執行 Function Blokc 後它們仍然可用，下一次執行可以延續上一次的結果繼續執行。專用記憶體為 STATIC 參數 *請參考 3-4-2 章節



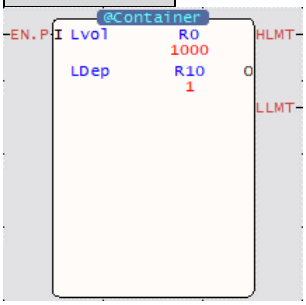
3-6 Function 程式範例

3-6-1 Water Container

應用功能

在內部長度和寬度固定的立方體水容器，一個能夠根據輸入水量計算出水深的 Function 指令。若輸入水量超過 6000 單位，上限警報開啟。若輸入水量小於 1000 單位，下限警報開啟。

Function 指令

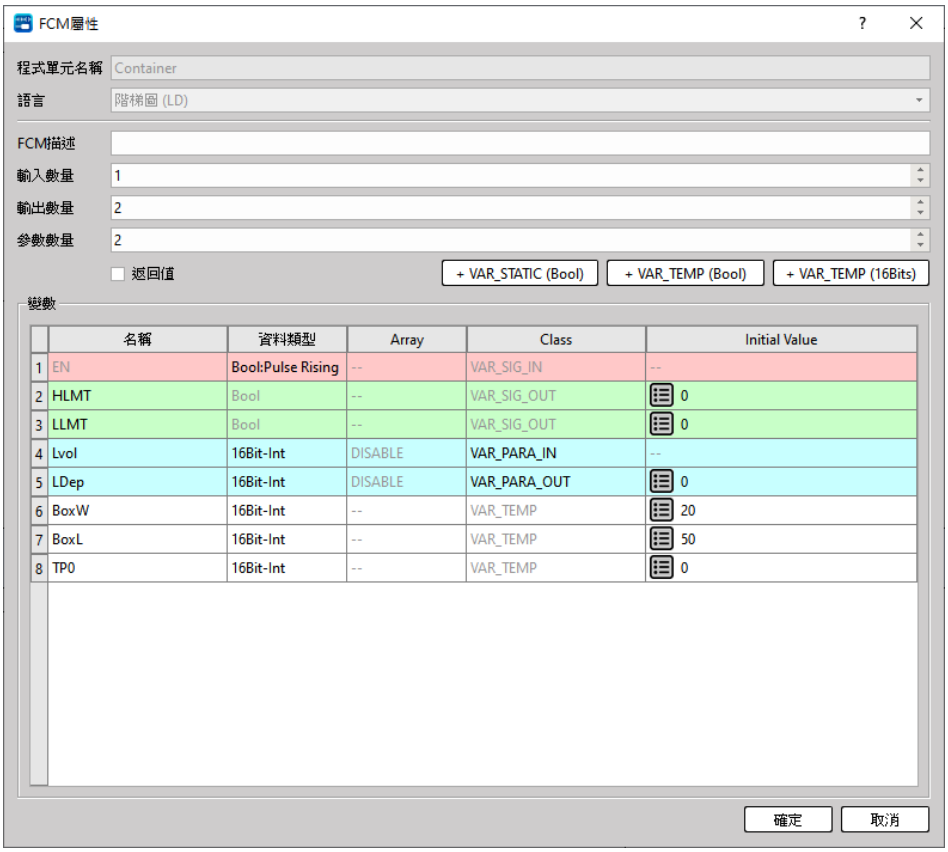


輸入/輸出腳位和參數

參數	名稱	說明
輸入腳位	EN.P	上緣觸發
輸出腳位	HLMT	上限警報
	LLMT	下限警報
輸入參數	Lvol	輸入水量
輸出參數	LDep	計算的水深

Function 屬性

1 個輸入數量(輸入腳位)，2 個輸出數量(輸出腳位)，2 個參數數量，3 個內部自動變數(VAR_TEMP)。

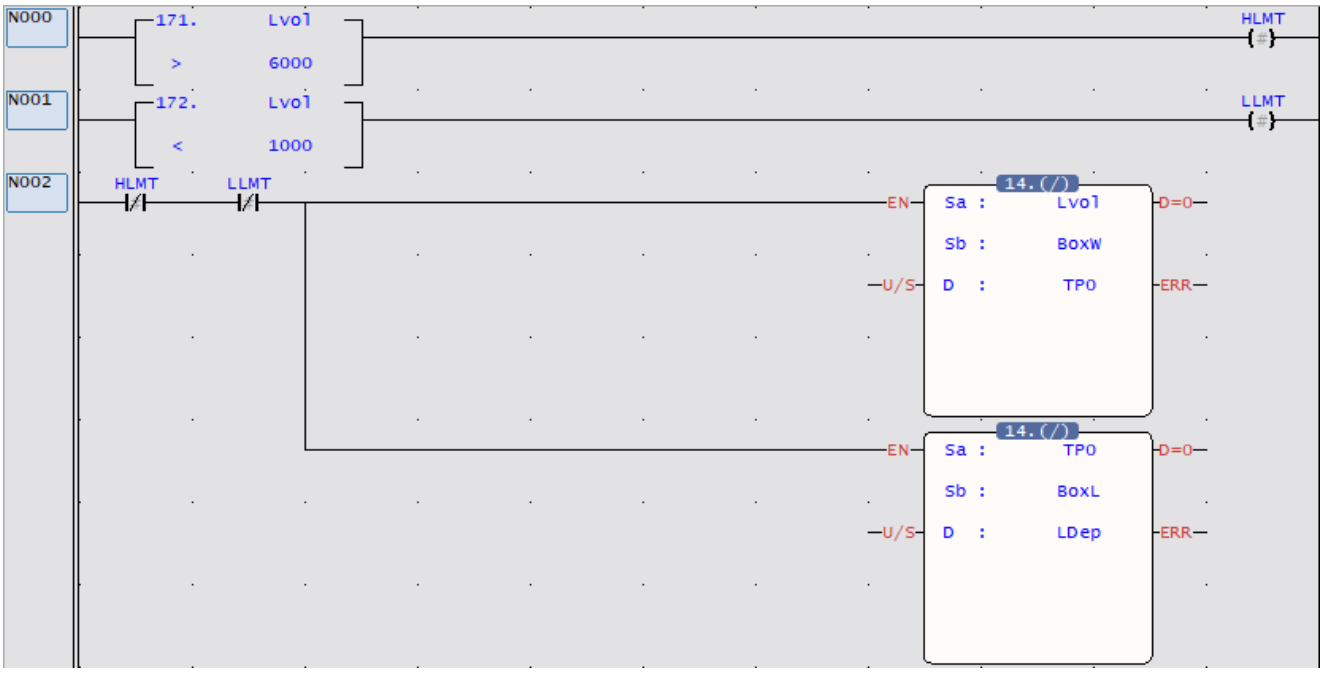


變數

名稱	資料類型	Array	Class	初始值	說明
EN.P	Bool; Pulse Rising	--	VAR_SIG_IN	--	上緣觸發
HLMT	Bool	--	VAR_SIG_OUT	每次都初始化為 0	水量上限警報
LLMT	Bool	--	VAR_SIG_OUT	每次都初始化為 0	水量下限警報
Lvol	16Bit-Int	DISABLE	VAR_PARA_IN	--	輸入的水量
LDep	16Bit-Int	DISABLE	VAR_PARA_OUT	每次都初始化為 0	計算的水深
BoxW	16Bit-Int	--	VAR_TEMP	每次都初始化為 20	容器的寬度固定為 20 單位
BoxL	16Bit-Int	--	VAR_TEMP	每次都初始化為 50	容器的長度固定為 50 單位
TP0	16Bit-Int	--	VAR_TEMP	每次都初始化為 0	計算過程中暫存

Function 內部程式說明

Function 指令 【Container】內部 FCM 程式



N000　　使用 Fun171 將輸入的 Lvol(水量)作比較，超過 6000 單位時觸發 HLMT(輸出上限警報)。

N001　　使用 Fun172 將輸入的 Lvol(水量)作比較，低於 1000 單位時觸發 LLMT(輸出下限警報)。

N002　　如果輸入的 Lvol(水量)沒有超過 HLMT(上限警報)和低於 LLMT(下限警報)，將 Lvol(水量)除以容器的寬度和長度計算出 LDep(水深)，並輸出給外面。

Function 內部程式_ST 語言

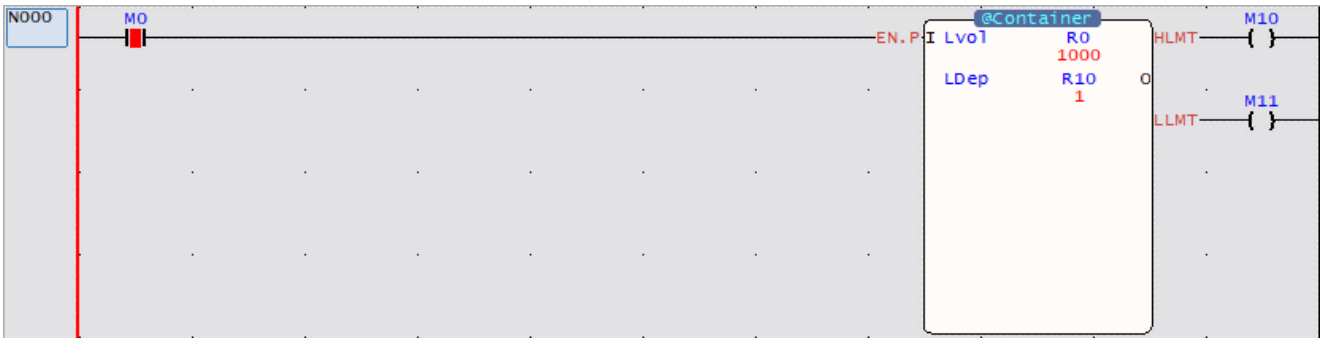
```
IF Lvol > 6000 THEN
    HLMT := TRUE;
END_IF

IF Lvol < 1000 THEN
    LLMT := TRUE;
END_IF

IF NOT HLMT AND NOT LLMT THEN
    TP0:= Lvol / BoxW;
    LDep:= TP0 / BoxL;
END_IF
```

動作說明

Function 指令 【Container】這個 Function 的使用說明



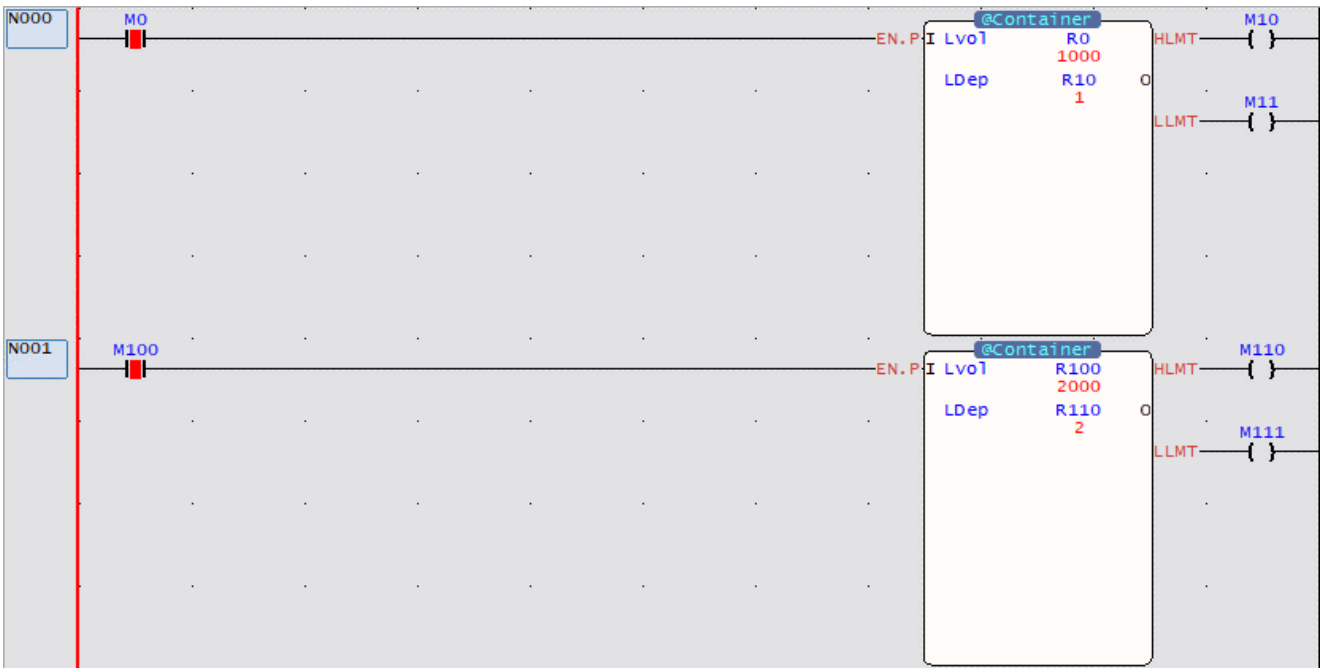
Step 1. 在 R0 輸入 Lvol(水量)

Step 2. 使用 M0 觸發 Function 指令的 EN.P (上緣觸發只計算一次)

Step 3. R10 將會顯示計算出的 LDep(水深)或是上下限警報(HLMT/LLMT)輸出警報

重複使用

在主程式重複使用 Function 指令 【Container】



N000 輸入 1000 單位水量，輸出 1 單位水深

N001 輸入 2000 單位水量，輸出 2 單位水深

互不相干擾運算結果

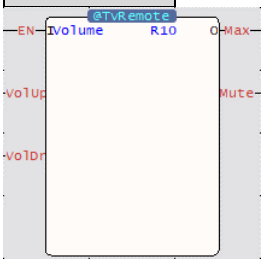
3-7 Function Block 程式範例

3-7-1 TV Remote

應用功能

一個使用遙控器控制電視機音量的 Function Block 指令，能夠記憶上次調整的音量繼續調整音量，音量值範圍為 0~10。

Function 指令

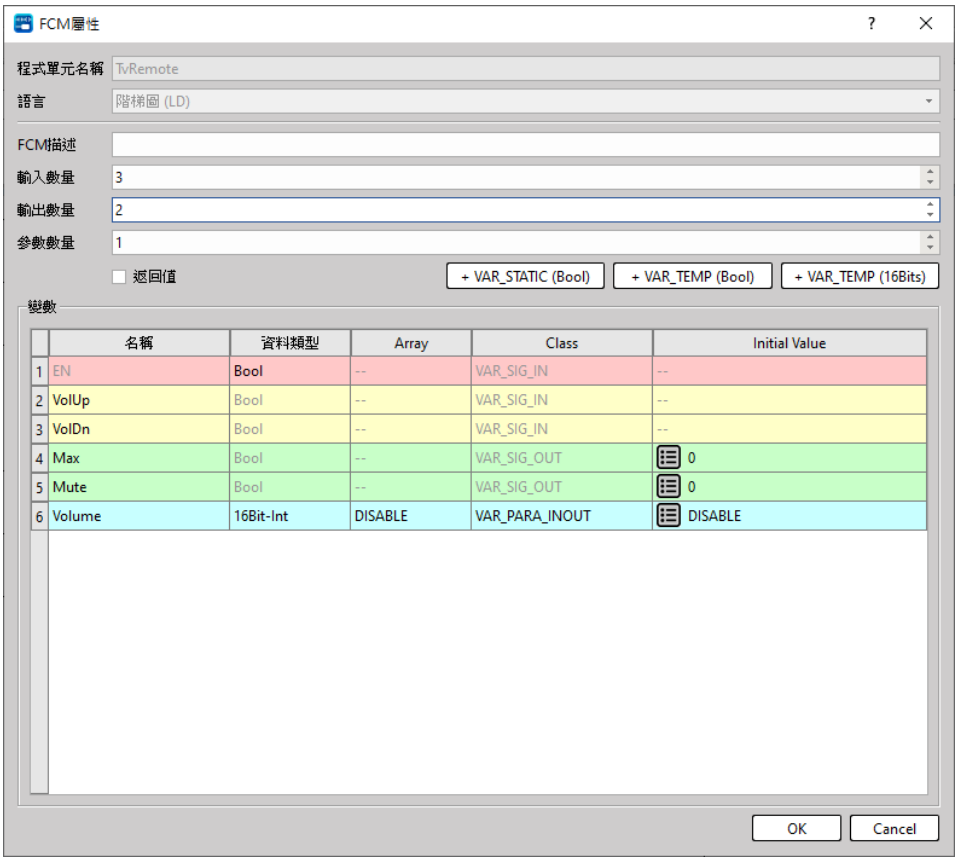


輸入輸出腳位和參數

參數	名稱	說明
輸入腳位	EN	準位觸發
	VolUp	增加音量
	VolDn	將低音量
輸出腳位	Max	音量最大值
	Mute	音量靜音
輸入輸出參數	Volume	音量

Function Block 屬性

3 個輸入數量(輸入腳位)，2 個輸出數量(輸出腳位)，1 個參數數量。



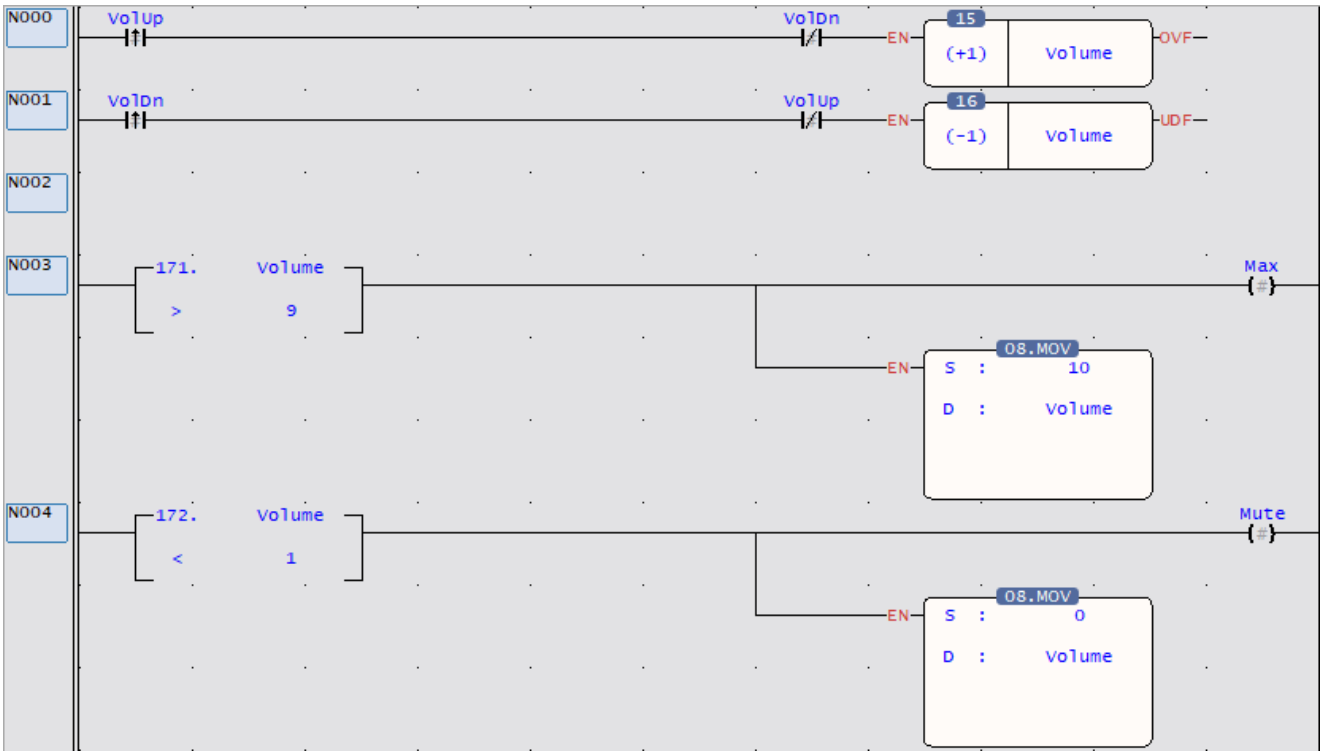
變數

名稱	資料類型	Array	Class	初始值	說明
EN	Bool	--	VAR_SIG_IN	--	上緣觸發
VolUp	Bool	--	VAR_SIG_IN	--	增加音量
VolDn	Bool	--	VAR_SIG_IN	--	將低音量
Max	Bool	--	VAR_SIG_OUT	每次都初始化為 0	音量最大值
Mute	Bool	--	VAR_SIG_OUT	每次都初始化為 0	音量靜音
Volume	16Bit-Int	DISABLE	VAR_PARA_INOUT	DISABLE	音量

*為了能夠記憶上次音量，這裡的音量使用 VAR_PARA_INOUT，執行 Function Block 時從外部暫存器將上一次的音量輸入 Function Block，Function Block 結束時將音量輸出給外部暫存器，給下一次執行 Function Block 時使用。

Function Block 內部程式說明

Function 指令 【TV Remote】內部 FCM 程式



N000　VolUp(音量增加)上緣觸發 Fun15 增加音量，包含互鎖機制。

N001　VolDn(音量降低)上緣觸發 Fun16 降低音量，包含互鎖機制。

N002　使用 Fun171 將 Volume(音量)作比較，超過 9 時觸發 Max(音量最大值)，並維持音量在 10。

N003　使用 Fun172 將 Volume(音量)作比較，低於 1 單位時觸發 Mute(音量靜音)，並維持音量在 0。

Function Block 內部程式_ST 語言

```
IF R_TRIG( S:=VolUp ) THEN
    IF NOT VolDn THEN
        Volume:= Volume+1;
    END_IF
END_IF

IF R_TRIG( S:=VolDn ) THEN
    IF NOT VolUp THEN
        Volume:= Volume-1;
    END_IF
END_IF

if Volume > 9 THEN
    Max:=TRUE;
    Volume:=10;
ELSEIF Volume < 1 THEN
    Mute:=FALSE;
    Volume:=0;
END_IF
```

動作說明

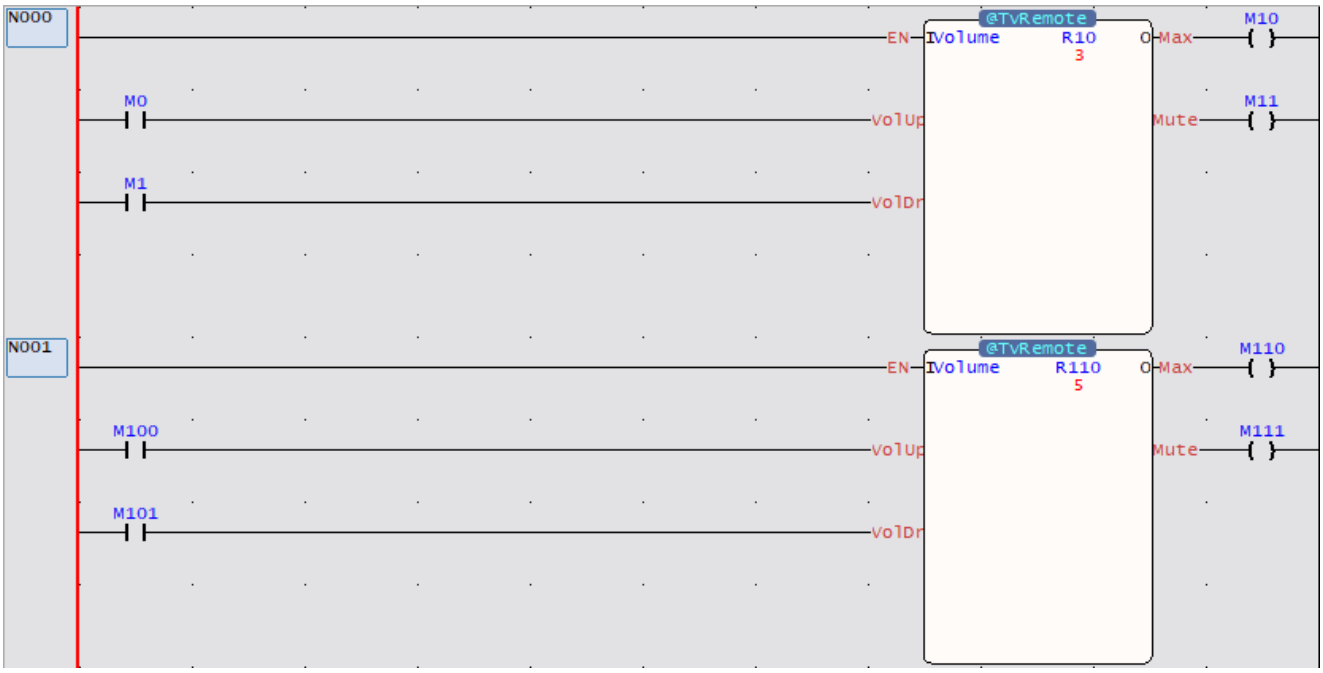
Step 1.　使用 VolUp(音量增加)或 VolDn(音量降低)增減音量

Step 2.　音量增加到 10 時，無法再繼續增加。

Step 3.　音量降低到 0 時，無法再繼續降低。

重複使用

在主程式重複使用 Function 指令 【TV Remote】



N000　控制 R10 音量

N001　控制 R110 音量

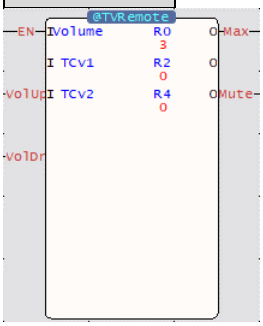
互不相干擾運算結果

3-7-2 TV Remote Timer

應用功能

使用遙控器控制電視機音量的 Function Block 指令，能夠記憶上一次的音量繼續調整音量，要跟真正的電視遙控器一樣，允許使用者隨著按鈕按下時間的延長而更快地增加音量，音量值範圍為 0~500。

Function 指令

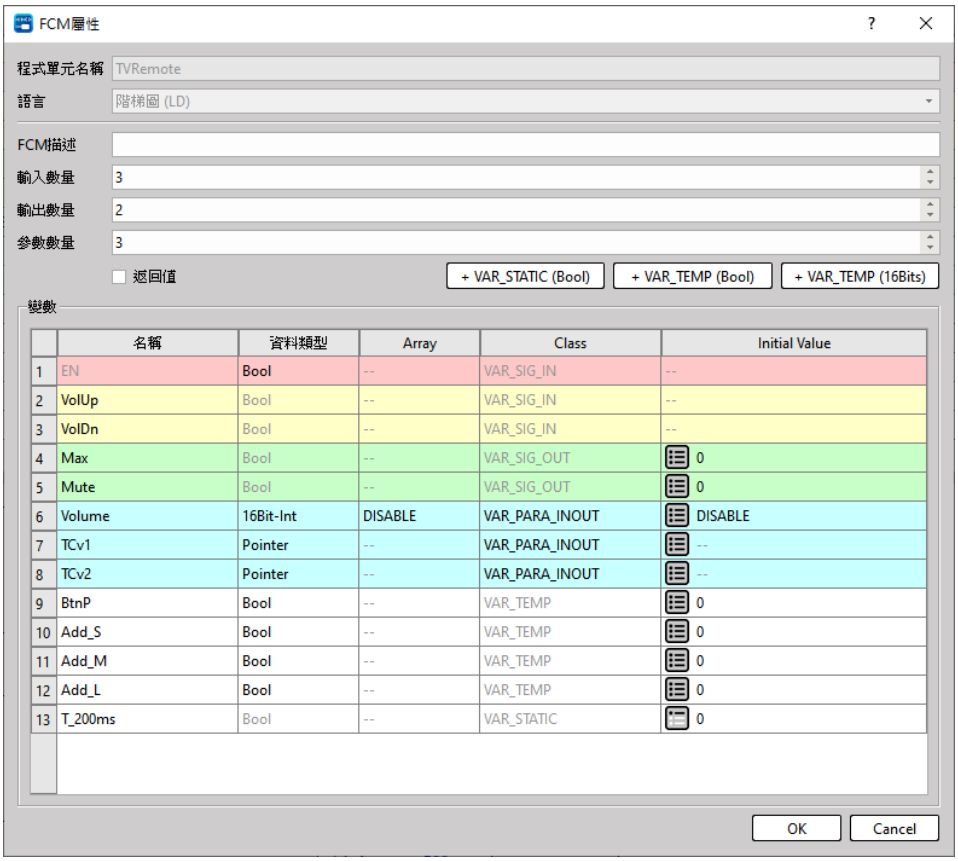


輸入輸出腳位和參數

參數	名稱	說明
輸入腳位	EN	準位觸發
	VolUp	增加音量
	VolDn	將低音量
輸出腳位	Max	音量最大值
	Mute	音量靜音
輸入輸出參數	Volume	音量
	TCv1	Fun87 Timer 計時用
	TCv2	Fun87 Timer 計時用

Function Block 屬性

3 個輸入數量(輸入腳位)，2 個輸出數量(輸出腳位)，3 個參數數量。



變數

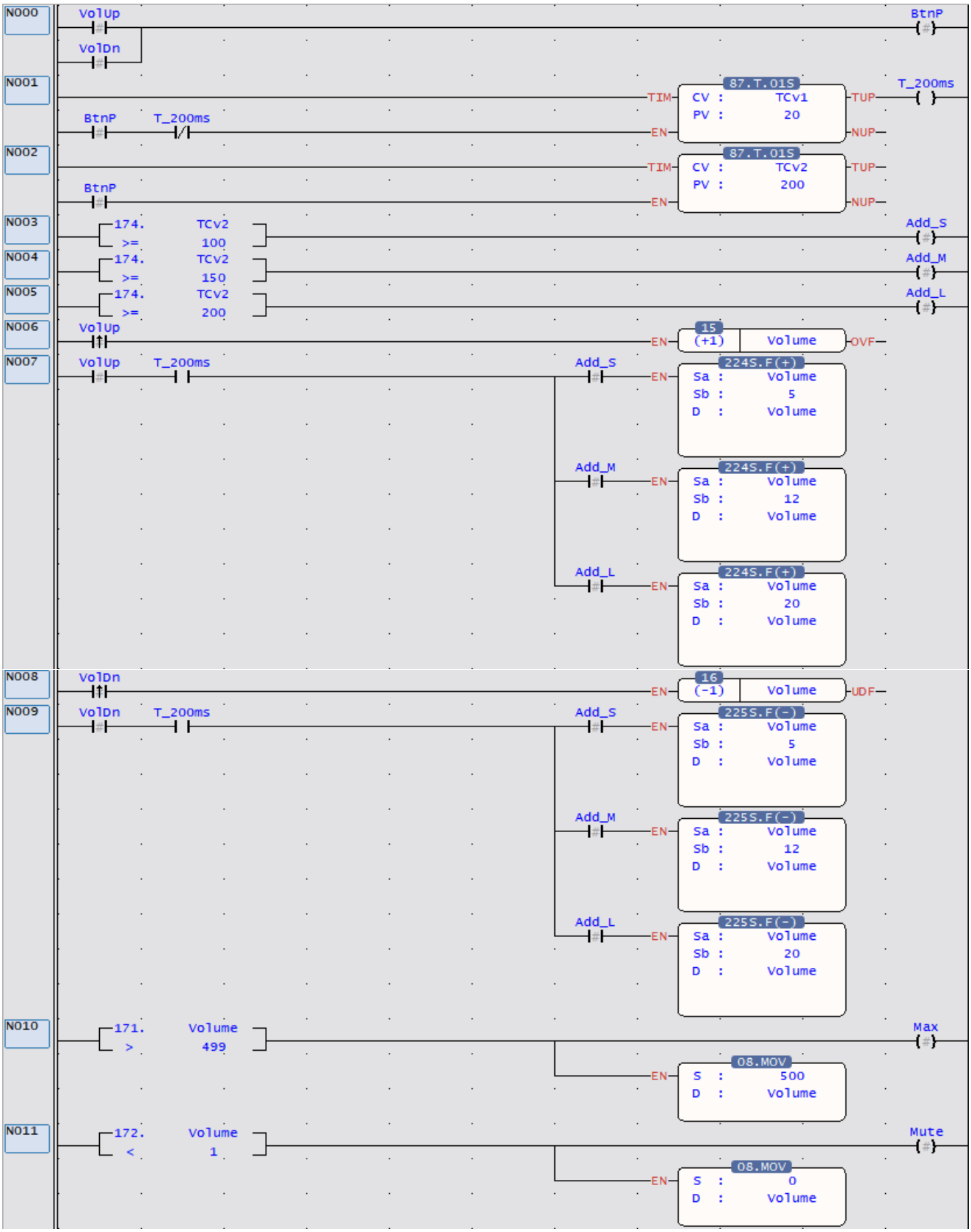
名稱	資料類型	Array	Class	初始值	說明
EN	Bool	--	VAR_SIG_IN	--	上緣觸發
VolUp	Bool	--	VAR_SIG_IN	--	增加音量
VolDn	Bool	--	VAR_SIG_IN	--	將低音量
Max	Bool	--	VAR_SIG_OUT	每次都初始化為 0	音量最大值
Mute	Bool	--	VAR_SIG_OUT	每次都初始化為 0	音量靜音
Volume	16Bit-Int	DISABLE	VAR_PARA_INOUT	DISABLE	音量
TCv1	Pointer	--	VAR_PARA_INOUT	--	Fun87 Timer 計時用
TCv2	Pointer	--	VAR_PARA_INOUT	--	Fun87 Timer 計時用
BtnP	Bool	--	VAR_TEMP	每次都初始化為 0	
Add_S	Bool	--	VAR_TEMP	每次都初始化為 0	
Add_M	Bool	--	VAR_TEMP	每次都初始化為 0	
Add_L	Bool	--	VAR_TEMP	每次都初始化為 0	
T_200ms	Bool	--	VAR_STATIC	第一次初始化為 0	Fun87 Timer 計時輸出用

*為了能夠記憶上一次音量，這裡的音量使用 VAR_PARA_INOUT，執行 Function Block 時從外部暫存器將上一次的音量輸入 Function Block，Function Block 結束時將音量輸出給外部暫存器，給下一次執行 Function Block 時使用。

*這邊使用了複週期指令 Fun87 累計計時器，為了讓 TIMER 的計時和輸出狀態可以在 Function Block 中跨越多個掃描時間，TIMER 的計時用了 Pointer(VAR_PARA_INOUT)，TIMER 的輸出用了 VAR_STATIC。

Function Block 內部程式說明

Function 指令 【TV Remote】內部 FCM 程式



N000 將 VolUp(音量增加)和 VolDn(音量降低)並聯成一個 BtnP 訊號
N001 在 VolUp(音量增加)或 VolDn(音量降低)被控制時，使用 Fun87 產生一個 200ms 的脈衝。
N002 在 VolUp(音量增加)或 VolDn(音量降低)被控制時，使用 Fun87 計時。
N003 Fun87_TCv2 計時到 1 秒時輸出 Add_S
N004 Fun87_TCv2 計時到 1.5 秒時輸出 Add_M
N005 Fun87_TCv2 計時到 2 秒時輸出 Add_L
N006 VolUp(音量增加)上緣增加音量
N007 VolUp(音量增加)持續按下超過 1 秒時，依據 Fun87_TCv2 不同時間用 200ms 的脈衝增加不同
N008 VolDn(音量降低)上緣降低音量
N009 VolDn(音量降低)持續按下超過 1 秒時，依據 Fun87_TCv2 不同時間用 200ms 的脈衝降低不同的音量。
N0010 使用 Fun171 將 Volume(音量)作比較，超過 499 時觸發 MAX(音量最大值)，並維持音量在 500。
N0011 使用 Fun172 將 Volume(音量)作比較，低於 1 單位時觸發 MUTE(音量靜音)，並維持音量在 0。

Function Block 內部程式_ST 語言

```
BtnP:=VolUp or VolDn;

ACTimer_10MS( TIM:= true, EN:=BtnP AND NOT T_200ms , CV:=TCv1 , PV:=20    , TUP=>T_200ms ,
NUP=>InZone);
ACTimer_10MS( TIM:= true, EN:=BtnP , CV:=TCv2 , PV:=200   , TUP=>InZone   , NUP=>InZone);

IF TCv2>=200 Then
    Add_L:=true;
elseIF TCv2>=150 Then
    Add_M:=true;
elseIF TCv2>=100 Then
    Add_S:=true;
END_IF

IF R_TRIG( S:=VolUp ) THEN
   Volume:= Volume+1;
END_IF

IF VolUp and T_200ms THEN
        IF Add_S THEN
        Volume:= Volume+5;
    ELSEIF Add_M THEN
        Volume:= Volume+12;
    ELSEIF Add_L THEN
        Volume:= Volume+20;
```

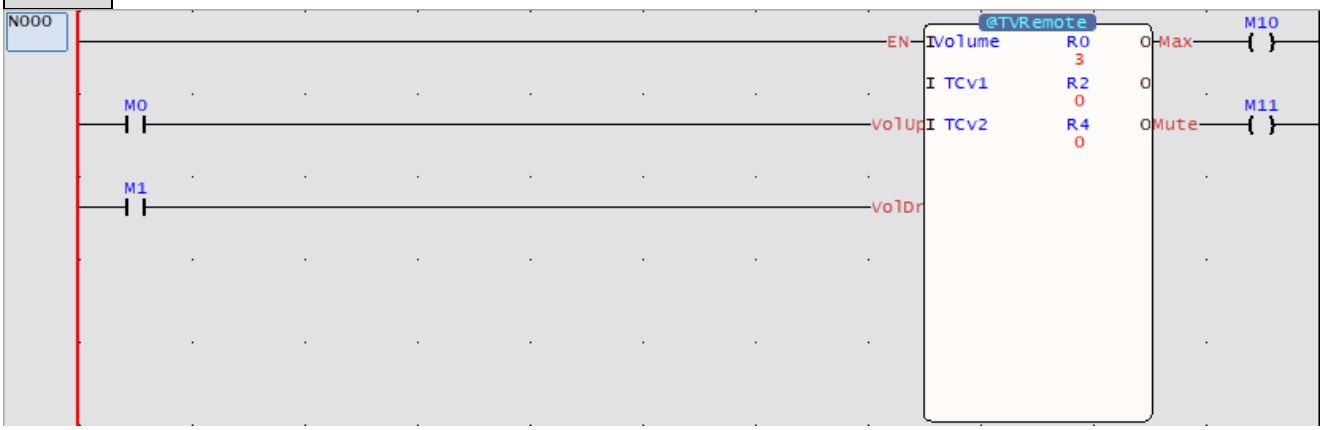
```
    END_IF
END_IF

IF R_TRIG( S:=VolDn ) THEN
    Volume:= Volume-1;
END_IF

IF VolDn and T_200ms THEN
        IF Add_S THEN
        Volume:= Volume-5;
    ELSEIF Add_M THEN
        Volume:= Volume-12;
    ELSEIF Add_L THEN
        Volume:= Volume-20;
    END_IF
END_IF

if Volume > 499 THEN
    Max:=TRUE;
    Volume:=500;
ELSEIF Volume < 1 THEN
    Mute:=TRUE;
    Volume:=0;
END_IF
```

動作說明



Step 1.　使用 VolUp(音量增加)或 VolDn(音量降低)增減音量

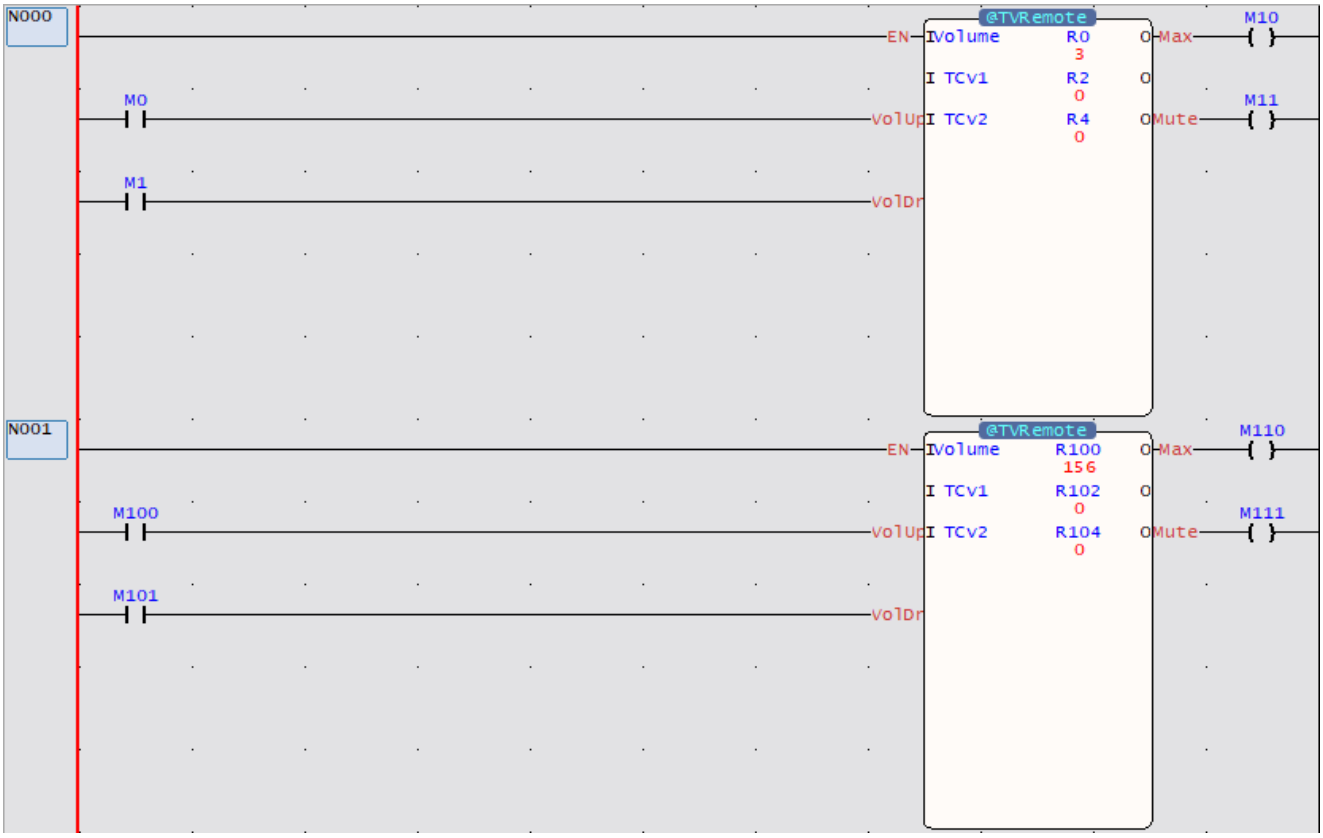
Step 2.　按壓 1 秒內增加或減少 1 格音量，按壓 1~1.5 秒每 0.2 秒增加或減少 5 格音量，按壓 1.5~2 秒每 0.2 秒增加或減少 12 格音量，按壓 2 秒以上每 0.2 秒增加或減少 20 格音量。

Step 3.　音量增加到 500 時，無法再繼續增加。

Step 4.　音量降低到 0 時，無法再繼續降低。

重複使用

在主程式重複使用 Function 指令 【TV Remote】



N002　控制 R0 音量

N003　控制 R100 音量

互不相干擾運算結果

【附錄一】 複週期指令表

複週期指令表

Function 編號	Function 名稱	Function 編號	Function 名稱
Fun33	LCNV	Fun176	MFFlowStart
Fun34	MLC	Fun177	MFSysStop
Fun38	PID2	Fun178	MFHome
Fun87	T.01S	Fun179	MFPointMov
Fun88	T.1S	Fun180	MFJog
Fun89	T1S	Fun181	MFChgTbPrm
Fun98	RAMP2	Fun183	MFFlowResume
Fun99	TPCTL2	Fun184	MFFlowHalt
Fun115	DBUF	Fun185	MFSysRstAlm
Fun137	ICA	Fun186	MFFlowStop
Fun138	ICF	Fun187	MFSysInit
Fun139	HSPWM	Fun188	MFSysRCPR
Fun140	HSPSO	Fun189	MFSysRCPW
Fun141	MPARA	Fun191	MFSysCAMR
Fun142	PSOFF	Fun192	MFSysCAMW
Fun143	PSCNV	Fun193	MFGearMPG
Fun144	HSPWM2	Fun194	MFVelCtl
Fun148	MPG	Fun195	MFTorqCtl
Fun150	MBUS	Fun196	MFSysCAMGen
Fun151	CLINK	Fun197	MFAxMov
Fun152	NCR	Fun198	MFMapTbPrm
Fun156	CMCTL	Fun235	MFSysSetVirt
Fun160	RWFR	Fun236	MFSDOR
Fun161	WRMP	Fun237	MFSDOW
Fun162	RDMP	Fun238	MFAxLMov
		Fun239	MFAxCirMov
		Fun240	MFPathMov

修訂紀錄

版本	修訂日期	修訂內容	修改者
V1.0	2022/05/31	初版	Albert