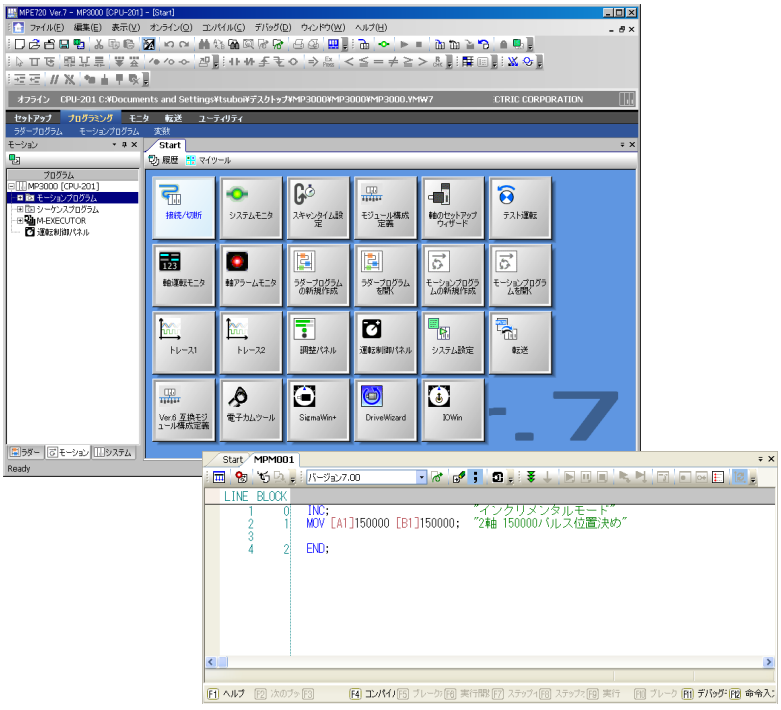
YASKAWA

運動控制器　MP3000系列

運動程式
程式編寫手冊



資料編號　YTWMNCO-15005A

前言

本手冊將針對 MP3000 系列運動控制器的階梯圖程式等相關資訊進行說明。

使用前請詳閱本手冊之說明，以期能正確使用本運動控制器系統，並善用系統功能以作為控制生產線系統之用。

請妥善保管本手冊，以便在需要的時候能隨時參考。

本手冊的使用方式

◆ 本手冊適用對象

本手冊的適用對象如下所示。

・MP3000 系列的系統設計人員

・負責編寫 MP3000 系列運動程式及序列程式的人員

◆ 簡稱及代號

本手冊使用下述簡稱及代號。

・固定參數：固定運動參數

・設定參數：設定運動參數

・監控參數：運動監控參數

・運動控制器：MP3000 系列運動控制器

・MP3200：電源元件、CPU 單元、基本元件、擴充槽 I/F 元件等的總稱

・MP3300：CPU 模組、基本元件等的總稱

・MP720：可程式裝置專用軟體或已安裝該軟體之可程式化裝置 (如電腦等)

・運動控制功能：運動模組內置的功能及 CPU 單元 /CPU 模組內建的 SVC/SVR、SVC 32/SVR 32 功能

◆ 本手冊中所使用的開發工具

本手冊引用 MPE720 Ver. 7 的畫面作為說明之用。

◆ 關於專有名詞「轉矩」

旋轉型伺服馬達通常使用「轉矩」，而線性伺服馬達則使用「推力」，本手冊則統一採用「轉矩」一詞來標示 (參數名稱除外)。

◆ 圖示符號

本手冊採用下列圖示符號，並標示在文章中重要的位置，目的在於明確地區別所說明的內容。


重要

包含務必遵守的注意事項或限制事項等。
代表雖然會發出警報，但還不至於造成裝置損壞程度之注意事項。


註記

包含注意或是避免操作錯誤之註記事項。

範例　包含操作或設定範例等。

補充　包含補充說明或是有用的輔助資訊等。


專有名詞解說

說明範圍包含不容易瞭解的專有名詞或是文章中所出現未預先說明的名詞。

相關使用手冊

下表為相關手冊。

使用本產品前，請確實瞭解使用限制條件，以期完全活用本產品。

分類	資料名稱	資料編號	內容
基本功能	運動控制器 MP2000/MP3000 系列 運動控制器系統 設定手冊	SIJP C880725 00	說明 MP2000/MP3000 系列　運動控制器安裝 / 連接、設定、試運轉、程式編寫與除錯以及各項功能。
	運動控制器 MP3000 系列 MP3200/MP3300 故障排除手冊	SIJP C880725 01	說明 MP3000 系列　運動控制器的問題排除事項。
	運動控制器 MP3000 系列 MP3200 使用手冊	SIJP C880725 10	說明 MP3000 系列　MP3200 規格、系統架構及 CPU 單元功能。
	運動控制器 MP3000 系列 MP3300 產品手冊	YTWNCO-14008A	說明 MP3000 系列　MP3300 的規格、系統架構及 CPU 模組功能。
通訊功能	運動控制器 MP3000 系列 通訊功能 使用手冊	YTWNCO-15003A	包含 MP3000 系列乙太網路通訊規格、系統架構及通訊連線方法等相關內容。
運動控制功能	運動控制器 MP3000 系列 運動控制功能 使用手冊	YTWNCO-14013A	包含 MP3000 系列運動控制功能 (SVC/SVC32、SVR/SVR32) 之規格、系統架構及使用方法等相關內容。
	運動控制器 MP2000 系列 脈衝輸出運動模組 PO-01 使用手冊	SIJP C880700 28	針對 MP2000 系列運動模組 PO-01 功能、規格、使用方法等進行更詳盡的說明。
	運動控制器 MP2000 系列 運動模組 SVA-01 使用手冊	SIJP C880700 32	針對 MP2000 系列運動模組 SVA-01 功能、規格、使用方法等進行更詳盡的說明。
	運動控制器 MP2000 系列 運動模組 內置型 SVB/SVB-01 使用手冊	SIJP C880700 33	針對 MP2000 系列運動模組 (內置 SVB、SVB-01、SVR) 功能、規格及使用方法進行更詳細的說明。
	運動控制器 MP2000 系列 運動模組 SVC-01 使用手冊	SIJP C880700 41	針對 MP2000 系列運動模組 SVC-01 功能、規格、使用方法等進行更詳盡的說明。
程式	運動控制器 MP3000 系列 階梯圖程式 程式編寫手冊	YTWMNCO-15004A	說明 MP3000 系列的階梯圖程式規格及指令。
開發工具	運動控制器 MP2000/MP3000 系列 系統整合開發工具 MPE720 Ver. 7 使用手冊	SIJP C880761 03	說明 MPE720 Ver. 7 的操作方法。

安全注意事項

本手冊採用下列標誌，提醒使用者注意使用安全。
攸關安全的標誌上所編寫的文字係為重要內容，請務必嚴格遵守之。

⚠ 警告 若未遵守本書所示之指示，恐將造成死亡、嚴重人身傷害等意外發生。

⚠ 注意 若未遵守本書所示之指示，恐將造成輕度 ~ 中度人身傷害，或是財物損壞等意外發生。
此外，即使是已刊載於 ⚠ 注意 上之事項，在某些狀況下亦有可能導致嚴重的後果。

禁止 ⃠內已載明了具體的內容，代表嚴禁 (嚴格禁止) 圖中所述之事項。
例如，⃠符號表示嚴禁煙火。

強制 ●內已載明了具體的內容，代表強制 (絕對必要) 圖中所述之事項。
例如，⏚符號表示強制接地。

以下所示為存放、搬運、安裝、配線、運轉、維護、檢查及報廢時務必遵守之重要注意事項。

◆ 一般

⚠ 警告

- **本產品應由專業技術人員妥善進行設置。**
  否則恐將造成觸電或人身傷害等意外發生。
- **連接裝置運轉時，應保持隨時可以緊急停止之狀態。**
  否則恐將造成人身傷害等意外發生。
- **運轉時因瞬間停電而重置時，裝置有可能突然重新啟動，此時請勿靠近裝置。重新啟動時，請採取安全防護措施，以維護人身安全。**
  否則恐將造成人身傷害等意外發生。
- **嚴禁碰觸產品內部。**
  否則恐將造成觸電意外。
- **請勿在通電狀態下卸除前方護蓋、纜線、接頭及配件等。**
  否則恐將造成觸電、故障或產品損壞等意外發生。
- **請勿刮傷纜線，強力拉扯、不當施力、在纜線上放置重物或是擠壓纜線。**
  否則恐將造成觸電或產品停止動作，甚至損毀等意外發生。
- **嚴禁改造本產品。**
  否則恐將造成裝置的損壞。

◆ 存放 / 搬運

⚠ 注意

- **本產品必須存放於下述環境。**
  - 不會受到陽光直射之場所
  - 環境溫度未超過所規定的存放溫度條件之場所
  - 相對濕度未超過所規定的存放濕度條件之場所
  - 溫度不會急遽變化或結露之場所
  - 不含腐蝕性或可燃性氣體之場所
  - 粉塵、灰塵、鹽分或金屬粉末含量較低之場所
  - 不會碰觸水、油或藥品等之場所
  - 產品不會受到震動或撞擊力影響之場所

  否則恐將造成火災、觸電或裝置損壞等意外發生。
- **搬運時，請搬妥產品的主體部分。**
  搬運產品時，若僅搬運纜線或接頭部位，可能會造成接頭損壞或纜線斷線，並成人身傷害等意外發生。
- **請勿重覆堆疊本產品 (請依照標示)。**
  否則恐將造成產品的故障。
- **運送時，請勿讓本產品暴露於含有鹵素 (氟素、氯、溴或碘) 等氣體的環境。**
  可能會造成產品的故障或損壞。
- **如需針對包裝用的木質材料 (木框、合板或棧板等) 進行消毒或除蟲時，請務必採用煙燻以外的方式。**
  例：熱處理 (中心溫度大於 56°C，持續 30 分鐘以上)
  此外，請在包裝前的材料階段進行處理，而非包裝完成後再整體處理。
  若使用經過煙燻處裡的木質材料包裝電子產品 (單機或是裝載在機器等的產品上)，材料所散發出來的氣體或蒸氣，恐將嚴重損壞電子零件。尤其像是鹵素類消毒劑 (氟素、氯、溴或碘等) 可能會造成電容器內部的腐蝕。

◆ 安裝

⚠ 注意

- **請勿將本產品設置於下列環境中。**
  - 不會受到陽光直射之場所
  - 環境溫度不會超過所規定的設置溫度條件之場所
  - 相對濕度不會超過所規定的存放濕度條件之場所
  - 溫度不會急遽變化或結露之場所
  - 不含腐蝕性或可燃性氣體之場所
  - 粉塵、灰塵、鹽分或金屬粉末含量較低之場所
  - 不會碰觸水、油或藥品等之場所
  - 產品不會受到震動或撞擊力影響之場所

  否則恐將造成火災、觸電或裝置損壞等意外發生。
- **設置時，請勿讓本產品暴露於含有鹵素 (氟素、氯、溴或碘) 等氣體的環境。**
  可能會造成產品的故障或損壞。
- **請勿坐在本產品上，或是在產品上放置重物。**
  否則恐將造成產品的故障。
- **請勿覆蓋住進氣及排氣孔，此外，也請勿讓異物進入產品內部。**
  否則恐將造成內部元件品質劣化，甚至造成火災或產品故障。
- **請務必遵守本手冊所規定之安裝方向。**
  否則恐將造成產品故障。
- **設置產品、控制面板內面及其他裝置時，必須依照規定的間隔來設置。**
  否則恐將造成火災或產品故障。
- **請勿強力撞擊本產品。**
  否則恐將造成產品故障。
- **安裝電池時，應由專業的技術人員妥善進行安裝作業。**
  否則恐將造成觸電、人身傷害或裝置損壞等意外發生。
- **請勿碰觸電池的電極部分。**
  否則靜電恐將造成產品損壞。

◆ 配線

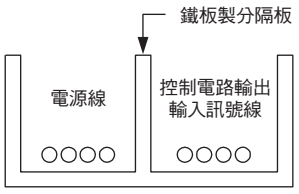
⚠ 注意

- **配線作業需正確且確實實施。**
  否則恐將造成馬達爆震，或是人身傷害、產品故障等意外發生。
- **使用時，請依指定的電源電壓正確使用。**
  否則恐將造成火災或產品故障。
- **在電源狀態不佳的場所使用本產品時，請於輸入功率在規定的電壓變動範圍內可供電的狀態下使用。**
  否則恐將造成裝置損壞。
- **請設置斷路器等安全裝置，以利外部配線短路時之用。**
  否則恐將引發火災等意外發生。
- **在以下場所使用本產品時，請分別採取適當的遮蔽措施。**
  - 因靜電而發生干擾之場所
  - 產生強力電場及磁場之場所
  - 有可能暴露於幅射源之場所
  - 電源線從附近經過之場所

  否則恐將造成裝置損壞。
- **相較於輸出輸入專用的 24 V 電源，本產品採用 CPU 單元 /CPU 模組必須先通電的電路架構。如需進一步瞭解電路相關資訊，請參閱以下的使用手冊。**
  MP3000 系列　CPU 單元　操作手冊 (資料編號：TOBP C880725 16)
  MP3000 系列 MP3300　CPU 模組　操作手冊 (資料編號：SIJP C880725 23)

  若先將輸出輸入專用的 24 V 電源等外部電源通電後，再對 CPU 單元 /CPU 模組通電，恐造成 CPU 單元 / CPU 模組的輸出瞬間啟動，並因無法預期的動作，造成人身傷害或裝置損壞等意外發生。
- **請利用產品外部的控制電路來架構攸關安全防護的緊急停止電路、互鎖電路及限制電路等。**
  否則恐將造成裝置的損壞。
- **使用 MECHATROLINK 輸出輸入模組時，必須先建立 MECHATROLINK 通訊，以作為互鎖輸出的條件。**
  否則恐將造成裝置損壞。
- **連接電池時，必須連接至正確的極性。**
  否則恐將造成電池損壞或爆炸。
- **選擇用來連接產品及外部裝置的輸出輸入訊號線 (外部配線) 時，必須考量以下事項。**
  - 機械強度
  - 干擾影響
  - 配線距離
  - 訊號電壓
- **如欲降低電源纜線所造成之干擾影響，控制面板內部和外部在進行控制電路的輸出輸入訊號纜線配線及設置時，均必須和電源線分隔。**
  若纜線未確實分隔時，可能會造成本產品的故障。

  配線分隔範例



◆ 運轉

⚠ 注意

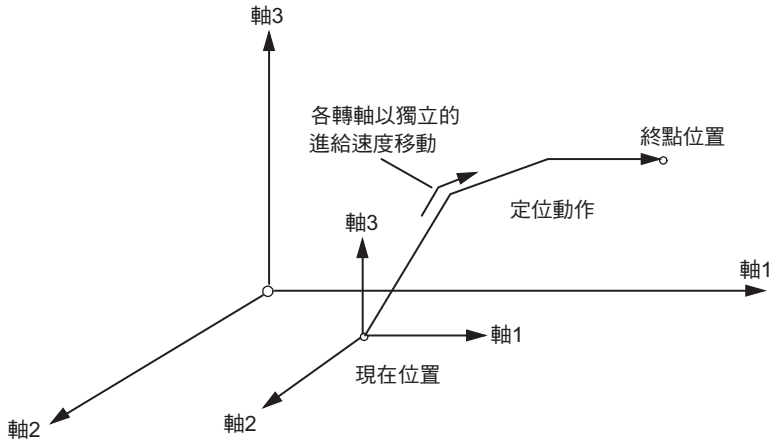
- **請依照產品相對應的使用手冊上所刊載之步驟及指示來進行運轉及試運轉作業。**
  若在伺服馬達與機器連接狀態下發生操作錯誤，不但將造成機器損壞，有時甚至會導致人身傷害等意外發生。
- **請在產品外部加裝互鎖訊號等安全電路，如此即使在下述狀況發生時，仍能確保整個系統的安全性。**
  - 產品故障或因外部因素而造成異常發生時
  - 本產品藉由自動診斷功能檢測異常發生，停止運轉，並將輸出訊號關閉 (或是維持) 時
  - 因輸出繼電器熔接、燒毀或電晶體損壞，使得產品的輸出變成開啟或關閉時
  - 本產品的 DC 24 V 輸出因過負載或短路狀態，以致電壓降低，而無法輸出訊號時
  - 因本產品的自動診斷功能無法檢測電源、輸出入部位或記憶體異常，而發生無預期輸出時

  以上狀況皆有可能導致人身傷害、裝置損壞或燒毀等意外。
- **設定下列參數時，必須依照本手冊所規定之方法來設定。**
  - 使用轉軸型作為有限長軸時所使用之絕對位置檢測參數
  - 使用轉軸型作為無限長軸時，用來設定簡易 ABS 無限長軸位置管理參數

  MP3000 系列　運動控制功能　使用手冊 (資料編號：YTWMNCO-14013A)

  若使用本手冊所未刊載的方法來進行設定，當電源啟動時，有可能造成目前位置偏移，因而導致裝置損壞。
- **以轉軸型作為有限長軸時之設定參數 OL□□□48 (機械座標系統原點偏移) 隨時有效。請勿在產品運轉狀態下，變更 OL□□□48 的設定內容。**
  否則恐將造成裝置損壞，甚至導致事故發生。
- **利用以下的轉軸移動指令來編寫程式時，務必先確認軌跡，以確認系統是否安全進行動作。**
  - 定位 (MOV)
  - 線性內插 (MVS)
  - 循環內插 (MCC、MCW)
  - 螺旋內插 (MCC、MCW)
  - 指定時間定位 (MVT)
  - 附略過功能線性內插 (SKP)
  - 原點復歸 (ZRN)
  - 外部定位 (EXM)

範例



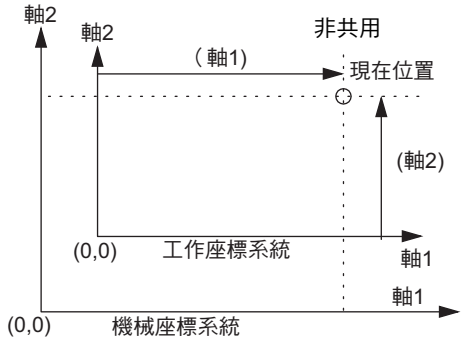
定位 (MOV) 基本軌跡範例

否則恐將造成裝置的損壞。

⚠ 注意

- **若在絕對模式下對相同的座標語下指令，此時的移動動作將和增量模式下執行指令完全不同。運轉前務必確認 ABS/INC 指令是否正確執行。**
  否則恐將造成裝置的損壞。
- **使用定位 (MOV) 功能時之移動軌跡將無法保持直線。使用本功能來編寫程式前，請先確認軌跡，然後再檢查系統是否安全執行動作。**
  否則恐將造成裝置的損壞。
- **線性內插 (MVS) 同時適用於直線軸或旋轉軸。但是當本功能用於旋轉軸時，線性內插的軌跡將無法保持直線。使用本功能來編寫程式前，請先確認軌跡，再確認系統是否安全執行動作。**
  否則恐將造成裝置的損壞。
- **執行螺旋內插 (MCW、MCC) 時，亦可同時將線性內插功能用於直線軸及旋轉軸。不過，線性內插部位的轉軸使用方式不同，將使得螺旋內插的軌跡無法形成螺旋狀。使用本功能來編寫程式前，請先確認軌跡，然後再檢查系統是否安全執行動作。**
  否則恐將造成裝置的損壞。
- **一旦下達下列錯誤的座標指令，恐將造成無法預期的意外發生。執行以下指令前，請先確認指令編寫是否正確後，再讓機器運轉。**
  - 絕對模式 (ABS)
  - 增量模式 (INC)
  - 變更現在值 (POS)

  範例



變更現在值 (POS) 後之全新工作座標系統範例

否則恐將造成裝置的損壞。

- **變更現在值 (POS) 可用來建立新的工作座標值。因此，一旦下達錯誤的 POS 指令，恐將造成無法預期的意外發生。下達 POS 指令時，請先確認工作座標系統的位置是否正確後，再讓機器開始運轉。**
  否則恐將造成裝置的損壞。
- **機械座標指令 (MVM) 可為機械座標上的座標位置暫時進行定位。因此，下達指令前，若未先確認好機械座標系統的原點位置，恐將造成無法預期的意外發生。下達 MVM 指令前，請先確認機械原點位置是否正確，再讓機器開始運轉。**
  否則恐將造成裝置的損壞。

◆ 維護 / 檢查

⚠ 注意

- **請勿自行拆解或維修本產品。**
  否則恐將造成觸電、人身傷害或裝置損壞等意外發生。
- **請勿在通電的狀態下更改配線。**
  否則恐將造成觸電、人身傷害或裝置損壞等意外發生。
- **更換電池時，應由專業的技術人員進行。**
  否則恐將造成觸電、人身傷害或裝置損壞等意外發生。
- **更換電池時，MP3□00 務必處於供電狀態。**
  在 MP3□00 供電中斷的狀態下交換電池時，儲存於 MP3□00 記憶體內的資料可能會被刪除。
- **更換電池時，請勿碰觸電極部位。**
  否則靜電恐將造成產品損壞。
- **更換 CPU 單元 /CPU 模組時，請勿遺漏下述作業。**
  ・ 為即將更換的 CPU 單元 /CPU 模組的程式及參數進行備份。
  ・ 將備份好的程式及參數傳送到新的 CPU 單元 /CPU 模組。
  未先傳送資料而運作 CPU 單元 /CPU 模組時，可能會因無法預期的動作而造成人身傷害或裝置損壞等意外發生。
- **通電過程中或電源關閉後請勿立刻碰觸 CPU 單元 /CPU 模組的散熱裝置。**
  若散熱裝置處於高溫狀態，恐將造成燙傷等意外發生。

◆ 廢棄

⚠ 注意

- **本產品應依照一般工業廢棄物標準進行處理。**
- **使用過的電池必須依照當地的規定進行處理。**

◆ 一般注意事項

使用時請注意。

- **本手冊上所刊載的插圖之目的在於說明，因此所顯示的圖面有可能為已卸除護蓋或安全防護品之狀態。操作本產品前，請務必依規定將護蓋或防護品恢復原狀，並依照使用手冊上之說明進行操控。**
- **本手冊上所刊載的插圖僅為代表性範例，與實際交貨的產品有可能會有部分差異。**
- **損壞或遺失本手冊時，如欲重新訂購，請聯絡本公司經銷商或本手冊封底所刊載本公司最近的業務單位，並告知欲訂購之資料編號。**

關於保固

◆ 保固內容

■ 保固期間

本公司對客戶所購買的產品 (以下簡稱交貨產品) 自交貨至客戶指定的地點該日起提供 1 年的產品保固，或是自本公司工廠出貨後 18 個月，以先到期者為準。

■ 保固範圍

保固期限內一旦發生故障，且可歸咎於本公司之責時，本公司將免費提供替代品或故障品維修的服務。
如因交貨產品的使用壽命已到而造成產品故障或需更換消耗品、週期性零件時，則不在保固範圍內。

此外，倘若故障原因符合下列因素時，亦不在本公司保固範圍內。

- 未依照型錄、手冊或另行取得的規格書上所刊載的適當條件、環境或操作方式來使用本產品
- 非交貨產品所造成時
- 產品改造或維修並非出自本公司時
- 產品的使用方式不符規定時
- 依本公司出貨時之科技、技術水準所無法預知之事由時
- 其他如天災、災害等非可歸咎於本公司責任之原因時

◆ 免責事項

- 對於因交貨產品故障所造成之損害以及客戶在機會上的損失等，本公司一律不負相關責任。
- 由非本公司人員對可程式化產品執行程式 (包含各種參數設定) 所導致的任何結果，本公司一律不負相關責任。
- 型錄或手冊上所刊載的所有資訊係以客戶能依用途而選擇適合的產品為目的。不保證其使用上對本公司及第三人之智慧財產權或其他權利不會造成侵害或表示同意其實施。
- 因使用型錄或手冊上的資訊而發生侵害第三人智慧財產權或其他相關權利，本公司一律不負責任。

◆ 確認適用用途及條件

- 如將本公司產品搭配其他產品使用時，請客戶先行確認適用規格、應遵守之法律或規定等。
- 請客戶自行確認您所使用的系統、機器、裝置與本公司產品之間之適用性。
- 本產品如欲作為下述用途，請先洽詢本公司後，再判斷是否可行。如判斷可行，建議採用額定規格、性能餘裕下使用，並採行安全措施，以便將故障時所發生的危險降至最低。
    - 在戶外或是受到潛在化學污染、電磁波干擾，或型錄、手冊上所未刊載的條件或環境下使用
    - 核能控制設備、燃燒設備、鐵路、航空、車輛設備、醫療裝置、娛樂設備，或是需要遵守行政機構或各業界規定之設備
    - 有危害生命或財產之虞的系統、機器或裝置
    - 瓦斯、自來水、電力供應系統或 24 小時連續運轉系統等需要高可靠性之系統
    - 其他符合上述條件且需要極高安全性之系統
- 若將本產品用於對生命或財產有重大危害之虞的用途時，請利用危險警告標誌或冗餘設計方式，來確保必要之安全性，此外，也必須事先確認本公司產品的配電和設置是否適當。
- 型錄或手冊上所刊載的電路實例或其他應用實例僅為參考之用，引用前，必須先確認您所要使用的機器、裝置的功能及安全性。
- 請正確瞭解所有使用時之禁止事項及注意事項，並正確使用本公司的產品，避免造成第三人意外損害。

◆ 規格變更

本公司因改善產品或其他事由有權變更型錄或手冊上所刊載的產品品名、規格、外觀或附件，且不另行通知。變更後，本公司將更新型錄或手冊上的資料編號，並發行修訂版。當您詢問或訂購型錄上所刊載的產品時，請事先和本公司業務人員確認。

目　錄

2 序列程式概述

3 程式開發流程

4 暫存器

運動程式概述

1

本章將以運動程式初次使用者為對象，說明程式概要、特性及其使用方法。

1.1 何謂運動程式

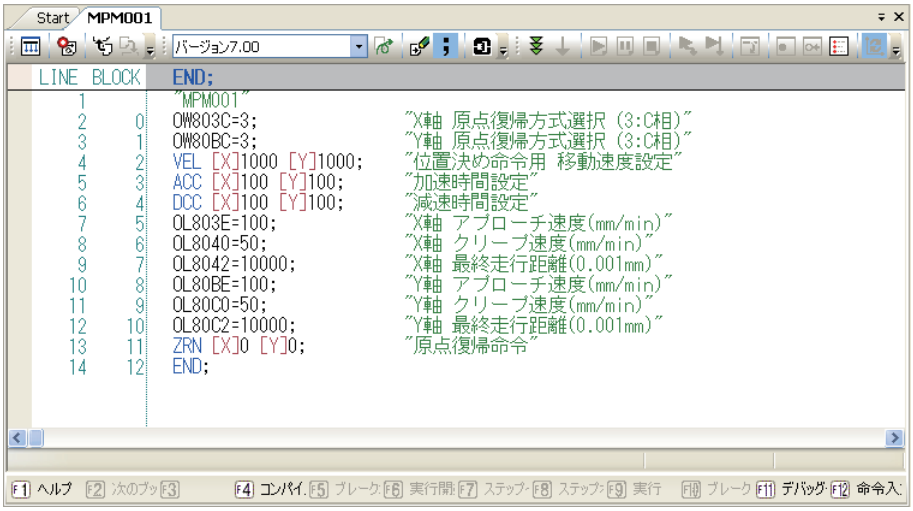
所謂的運動程式就是利用本公司獨創的文字形式語言，也就是以運動語言編寫而成的程式。

不同於階梯圖程式，運動程式的的語言指令 1 行就能執行各種動作。而且，和階梯圖程式完全不同的還有一點，那就是只要在插補指令中設定好目標位置、加速檢速時間或插補進給速度，系統就會根據所設定的參數，自動計算每次掃描的移動量，以執行運動控制。

運動程式可利用階梯圖程式的 MSEE 指令來編寫，或是叫出 M-EXECUTOR 程式的執行定義後即可執行。

有別於階梯圖程式，運動程式最多可編寫 512 個。

下圖為運動程式範例。

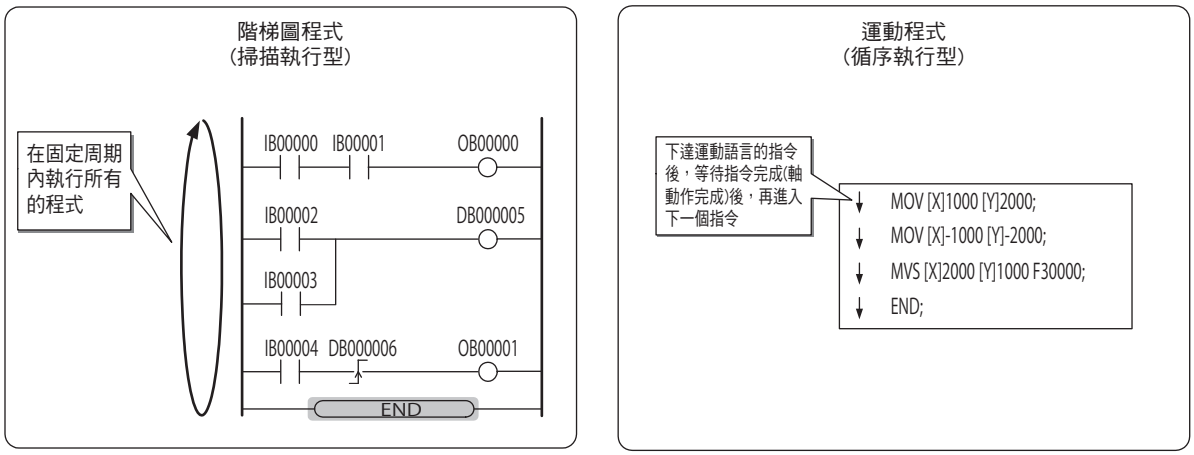


1.2 運動程式特色

本節將說明運動程式的特色。

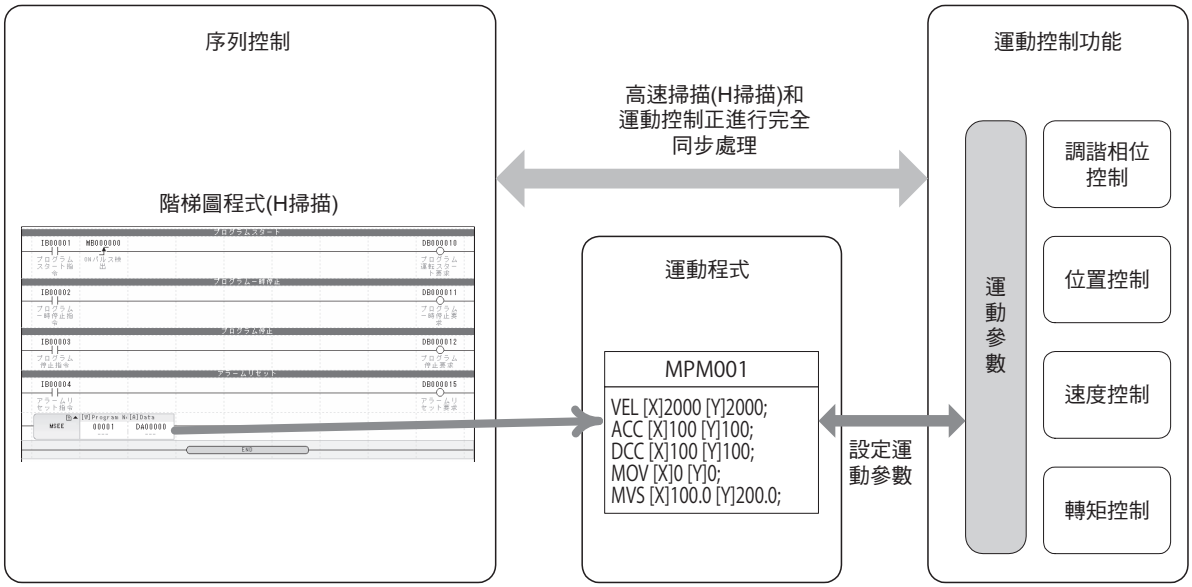
運動程式執行方式

運動程式採用了和階梯圖程截然不同的執行方式。
階梯圖程式會在一次掃描內將程式的起始到 END 指令為止處理完成。
而在運動程式中，即使只有 1 個指令，也會執行多次掃描來進行處理作業。運動程式會先完成 1 個指令，然後再進入下一個指令，循序執行處理作業。
本手冊將各個執行方式分為「掃描執行型」和「循序執行型」。



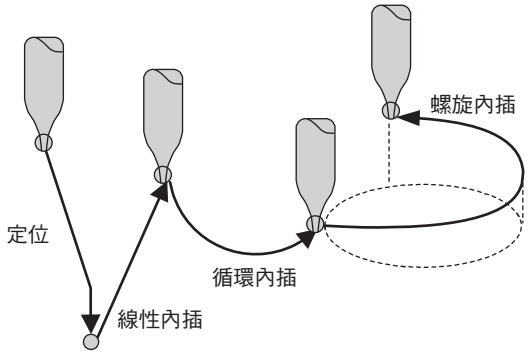
序列控制 & 運動控制完全同步化

利用運動程式編寫而成的處理作業和 MP3000 系列的高速掃描 (H 掃描) 完全進行同步。運動程式無啟動的時間延遲，從階梯圖程式下達啟動要求到運動程式啟動，皆會在 1 次掃描內執行。



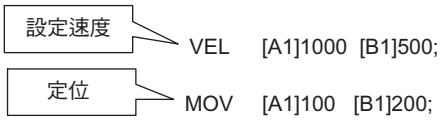
高階運動控制

除了基本的運動控制外，甚至是需要複雜動作的運動控制，也能透過運動程式輕鬆完成。



簡單易懂的指令語言

運動程式採用直覺式、簡單易懂的指令語言，像是利用下圖所示的「VEL」來設定速度，或是「MOV」來定位等。



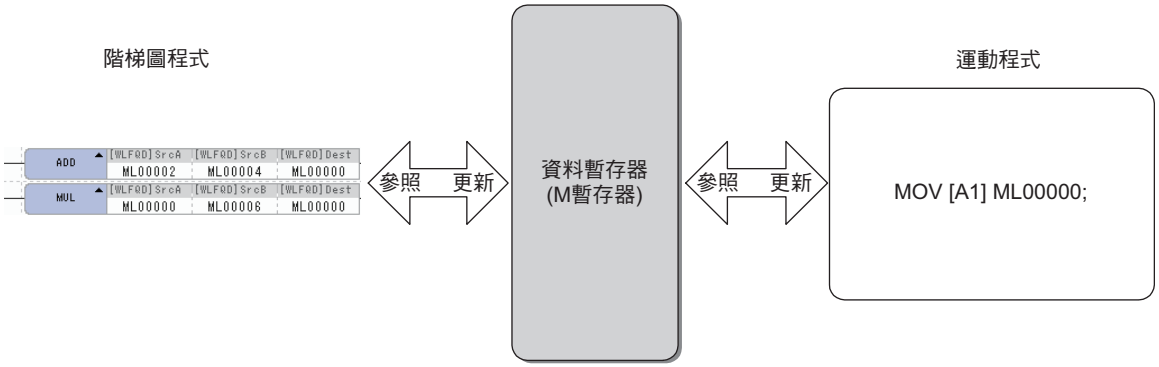
運動程式裡的算式運算

運動語言備有四則運算及邏輯運算等指令。
利用這些運算，就能在運動程式裡計算出目標位置。

```
DL00000 = DL00002 + DW00004;
DL00000 = DW00002 * DL00004;
MW00000 = MW00000 & 00FFH;
MF00000 = SIN(30.0);
```

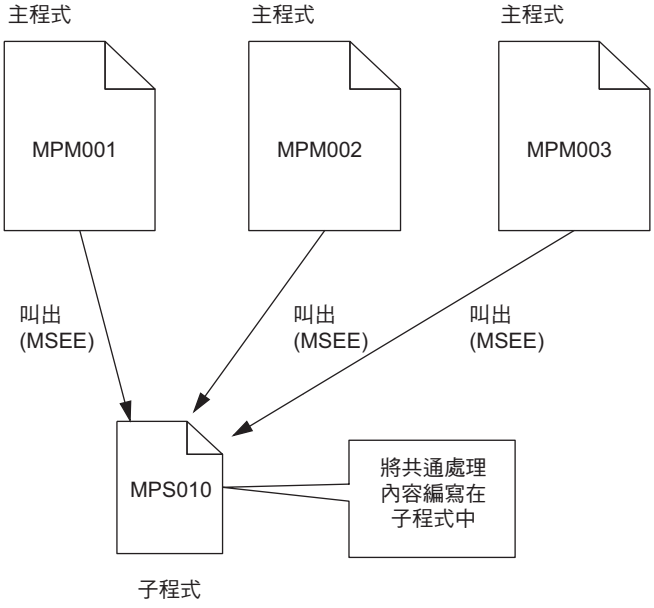
接收來自階梯圖程式的資料

階梯圖程式和運動程式之間可互相進行資料傳收。
資料傳收必須透過資料暫存器 (M 暫存器)。
運動程式可使用階梯圖程式裡更新後的數值，反之，階梯圖程式也可使用運動程式已更新的數值。



善用子程式以提高記憶體效率

運動程式可用來製作子程式 (子程式)。
只要讓共通的動作以子程式化 (共通化) 來執行，就能將程式的步進控制在最小化，並提高記憶體效率。



程式的並列執行

一台 MP3000 系列最多可同時處理 32 個任務。讓程式並列執行後，系統就能同時控制多個不同的動作。

利用主程式 / 子程式來執行 PFORK 指令時，每個任務最多可並列 8 個作業。

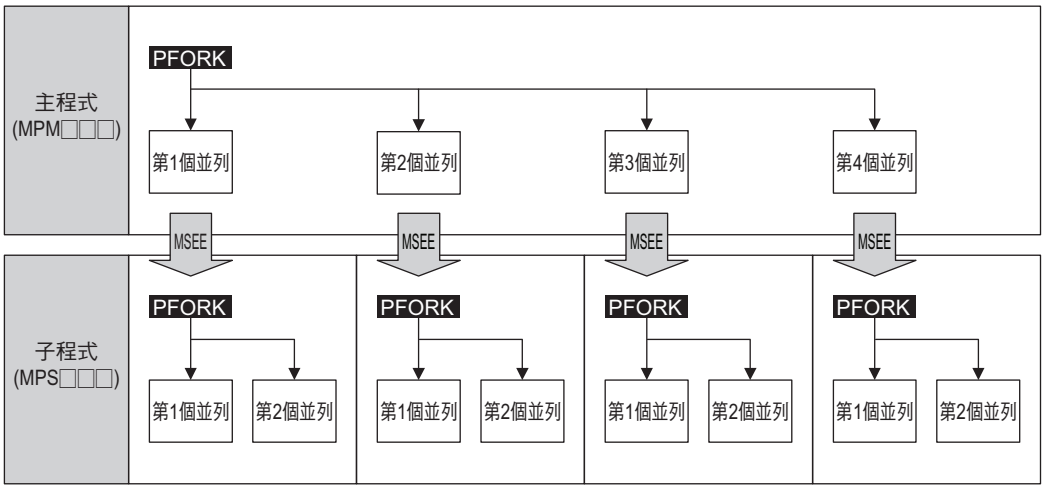
進入 [程式性質] 對話框，即可設定為並列模式。

PFORK 指令的並列模式包含 4 種模式，詳情如下圖所示。

主程式 4 x 子程式 2 (MP2000 互換模式)

主程式最多可執行 4 項並列，子程式則為 2 項。

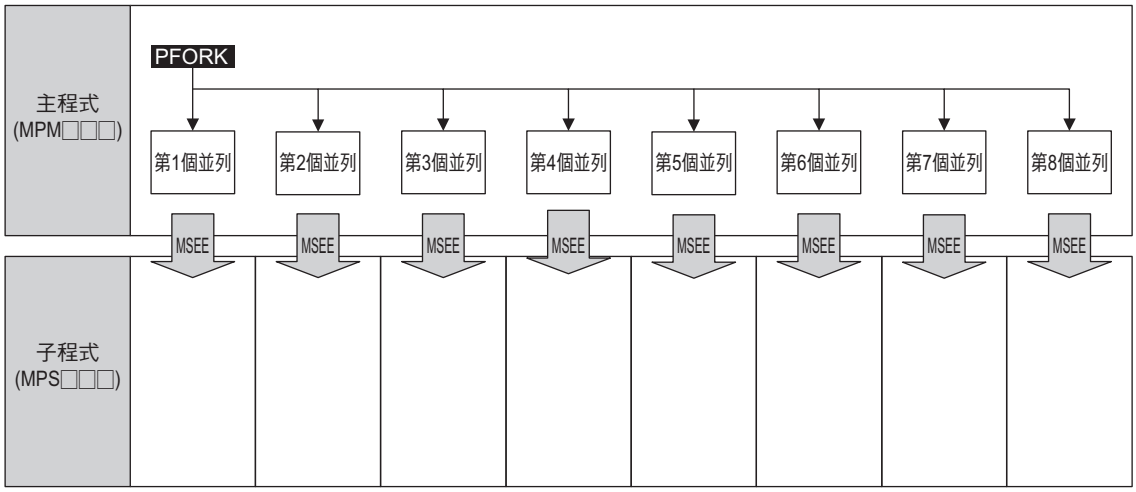
預設值為下圖所示的模式。



主程式 8 x 子程式 1

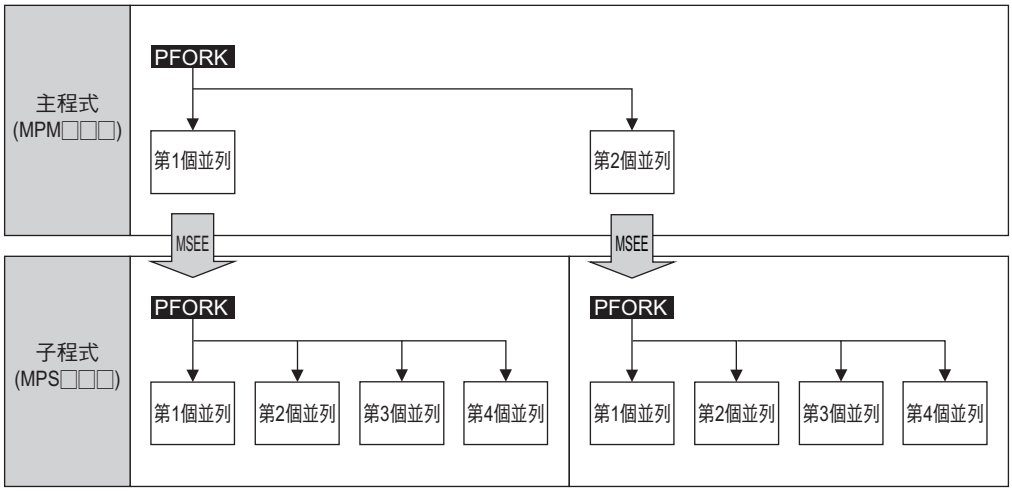
主程式最多可執行 8 項並列。

但子程式無法並列執行。



主程式 2 x 子程式 4

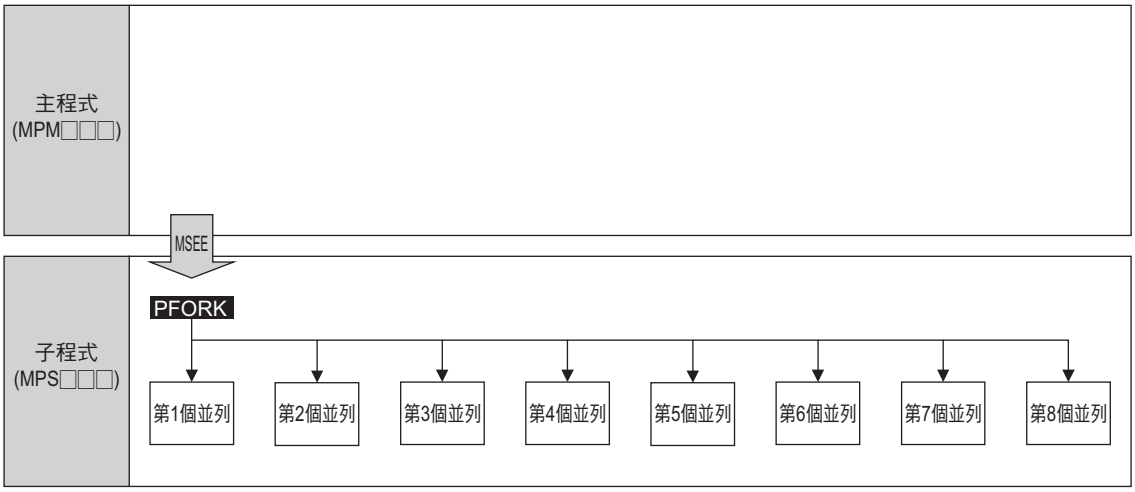
在此模式下，主程式最多可執行 2 項並列，而子程式則最多可執行 4 項並列。



主程式 1 x 子程式 8

子程式最多可執行 8 項並列。

但主程式無法並列執行。

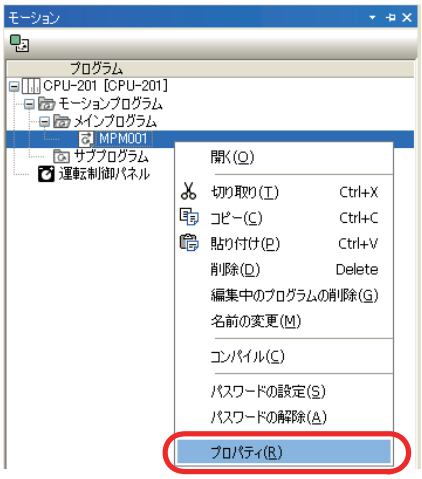


PFORK 並列模式設定方法

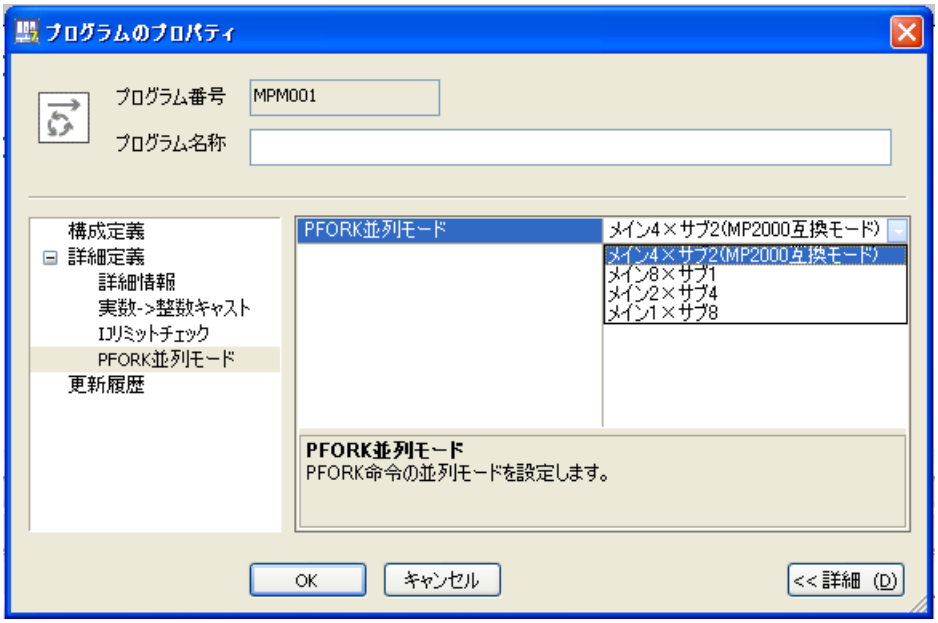
接下來將說明如何設定 PFORK 並列模式。

進入主程式中的 [程式性質] 對話框，即可開始設定並列模式。初始設定值為主程式 4 x 子程式 2。

1. 進入 [運動] 子視窗，並依序選擇 [運動程式] － [主程式] － [MPM001] － [性質]。



2. 進入 [程式性質] 對話框，並從 [PFORK 並列模式] 方塊中選擇。



■ 並列模式被設定為開啟之時間點

進入 [程式性質] 對話框，並點擊 [OK] 鍵，即可將並列模式設定為開啟。

檢查轉軸警報

本功能可用來確認 MP3000 系列運動程式所下達移動指令的轉軸是否發生警報 (IL □□□ 04)。

欲開啟 / 關閉本功能，只要進入 MPE720 Ver. 7 的 [環境設定] 即可。

MPE720 Ver 7[環境設定] 的設定方法請參閱下述手冊。

🕮 **MP2000/MP3000 系列 系統綜合開發工具　MPE720 Ver. 7 使用手冊**
(資料編號：SIJP C880761 03)

如欲進一步了解本功能，請參閱下列章節。

☞ **附錄 C MP2000 系列和 MP3000 系列間的差異**

檢查轉軸警報 (MP3000 系列標準功能)

一旦被下達移動指令的轉軸發生警報 (IL□□□04 ≠ 0) 時，運動程式就會發出警報，此時所有被下指令的轉軸將停止動作 (OW□□□09 Bit 1 =ON)，且下達 NOP (OW□□□08 = 0) 的運動指令。

MP3000 系列在出廠時已預設好此模式。

不檢查轉軸警報 (和 MP2000 系列互換)

即使被下達移動指令的轉軸發生警報 (IL□□□04 ≠ 0)，系統仍會持續對未發生警報的轉軸下達指令。

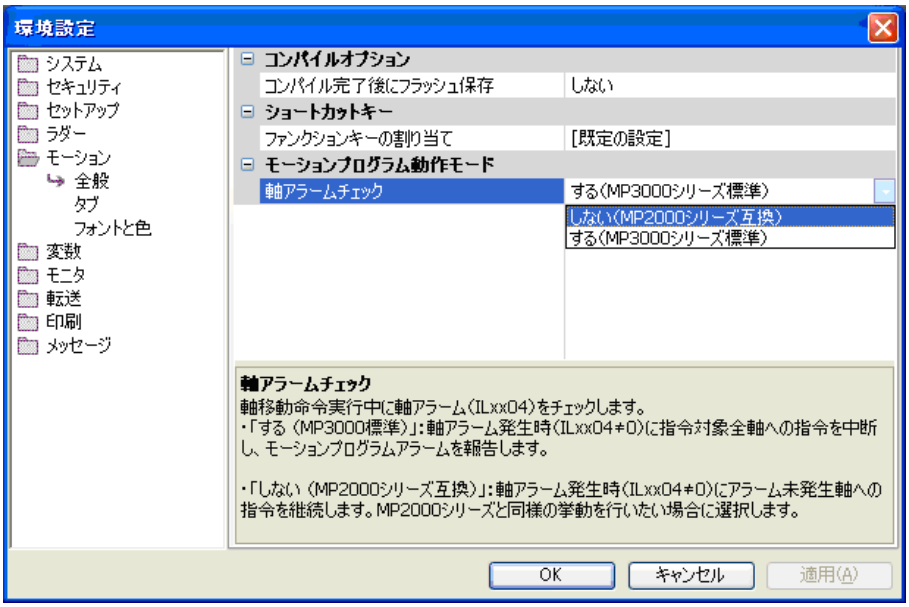
使用此模式時，必須和外部裝置互鎖。

補充　本模式將和 MP2000 系列執行相同的動作。若要以 MP3000 系列取代 MP2000 系列，或是和 MP2000 系列設定為相同的動作時，請選擇本模式。

檢查轉軸的方法

接下來將說明如何設定模式，以確認轉軸警報。

1. 進入 [MPE720 Ver. 7] 視窗並從檔案選單中選擇 [環境設定]。
2. 從 [環境設定] 對話框中，選擇 [動作] － [一般]。
3. 從 [運動程式動作模式] 方塊中，依序選擇 [檢查轉軸警報] － [執行 (MP3000 系列標準功能)]。



MP 控制器和 MPE720 Ver. 7 的版本組合

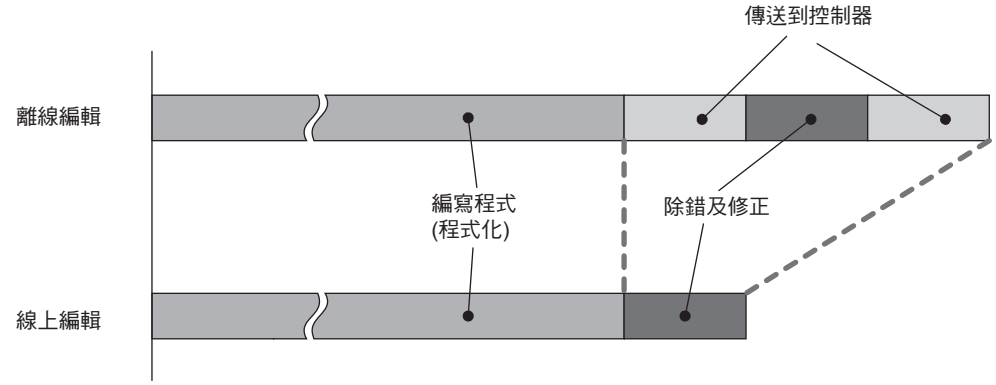
下表係用來說明 MP3000 系列與 MPE720 Ver 7 版本之各種組合。

MP3000 系列軟體版本	MPE720 Ver. 7 (Ver 7.21.0100 以後版本)		MPE720 Ver. 7 (Ver 7. 20.0100 以前版本)	
	選擇檢查轉軸警報	檢查轉軸警報 (是 / 否)	選擇檢查轉軸警報	檢查轉軸警報 (是 / 否)
MP3000 系列 (Ver 1.05 以前版本)	無法選擇	是 (固定)	無法選擇	是 (固定)
MP3000 系列 (Ver 1.06 以後版本)	可選擇	預設值為 「是」	無法選擇	是 (固定)

補充　利用 MP3000 系列 (Ver 1.06 以後版本) 和 MPE720 Ver. 7 (Ver 7.21.0100 以後版本) 組合表，設定並選擇是否檢查轉軸警報，接著再將設定內容儲存於快閃記憶體後，設定內容就會從下次開始生效。

線上編輯程式

運動程式和階梯圖程式一樣，可在線上進行程式編輯。
所謂的線上編輯就是讓控制器進入登入狀態後，再進行程式編輯。進行連線編輯時，僅需存取程式即可傳送至控制器，因此不需要再對控制器執行傳送，提升程式開發效率。



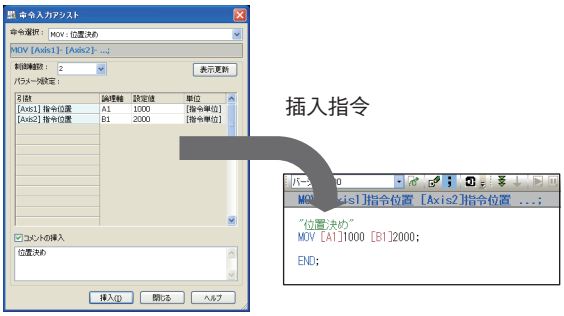
補充　執行運動程式時，無法同時進行線上編輯。

增加更多簡易程式功能 (MPE720 Ver. 7.0 以後版本)

MP3000 系列的開發工具「MPE720 Ver. 7.0」，包含了以下簡易程式功能。

● 指令輸入輔助功能

進入以下的對話框並選擇指令，接著只要設定好資料，
即可將指令插入編輯器中。



● 測試運轉功能

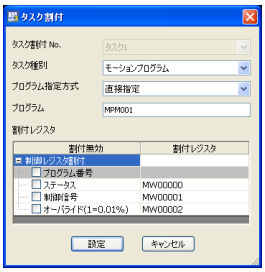
利用以下對話框，即可控制轉軸。

● 轉軸運轉監控功能

讓各轉軸的運轉狀況更一目瞭然。

● 任務配置

輕鬆即可將您所要執行的程式登錄於系統中。

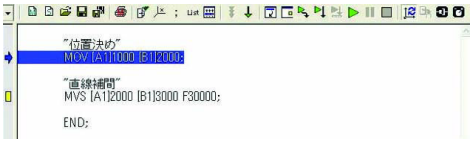


● 運轉控制面板功能

利用運動編輯器視窗，即可啟動運動程式。

● 除錯運轉功能

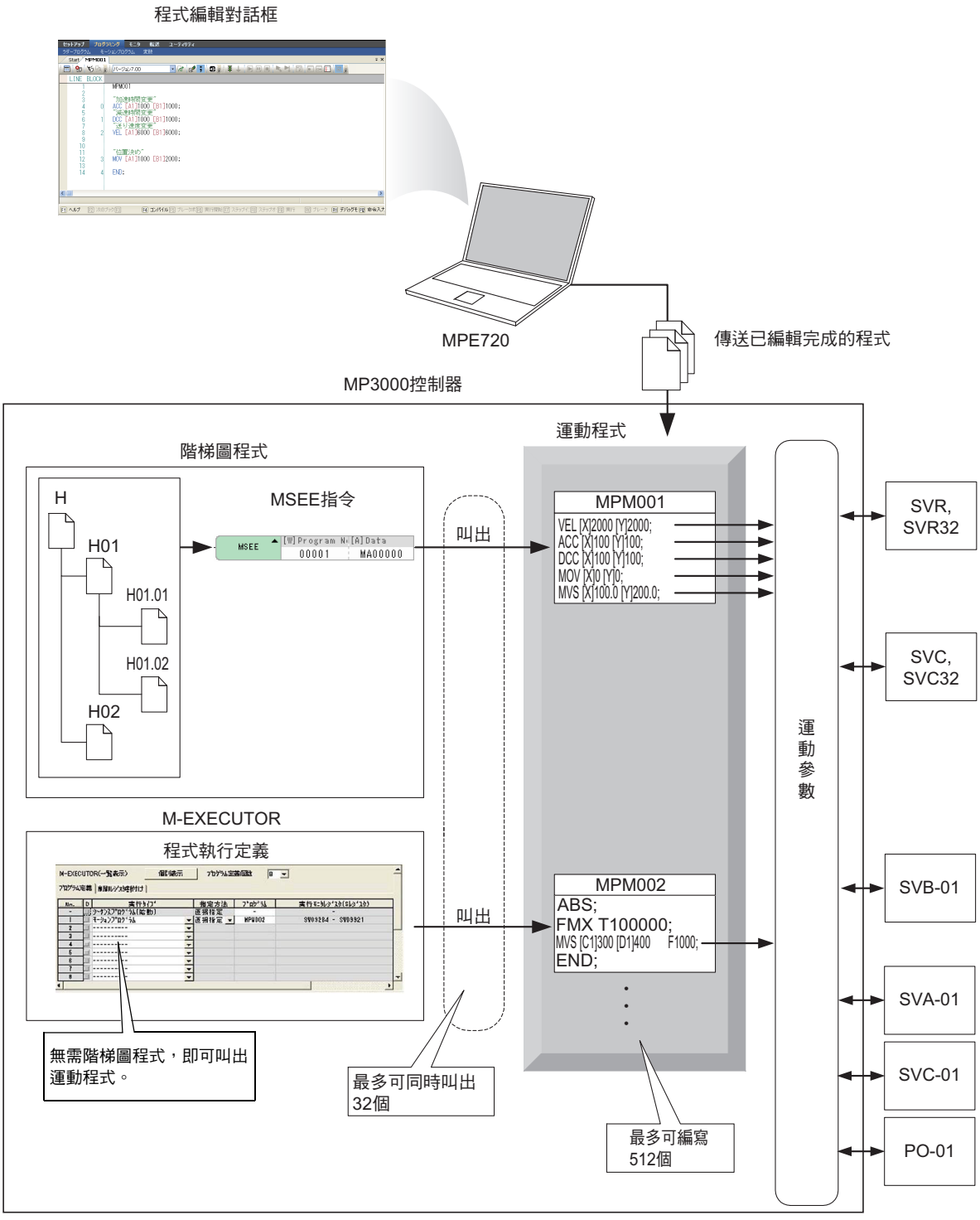
本功能可用來為運動程式除錯。
利用執行步進、設定斷點等的除錯指令，
即可為運動程式的動作進行除錯。



1.3 運動程式系統架構

當您完成 MPE720 運動編輯器的編輯後，運動程式的資料就會被傳送到 MP3000 控制器中。您所傳送的運動程式可利用階梯圖程式所編寫的 MSEE 指令來叫出，或是 M-EXECUTOR 的執行定義來叫出。運動語言指令係透過運動參數，對運動控制功能下達指令，然後再讓轉軸執行動作。

下圖為運動程式的系統架構。



1.4 運動程式的類型

運動程式包含下表所示的 2 種類型。

分類	指定方法	特徵	程式數
主程式	MPM□□□ (□□□= 1 ～ 512)	・從 M-EXECUTOR 程式的執行定義叫出 ・從 DWG.H 叫出	以下所示之程式最多可編寫 512 個 ・運動主程式 ・運動子程式 ・序列主程式 ・序列子程式
子程式	MPS□□□ （□□□= 1 ～ 512）	・從主程式叫出	


重要

1. 運動程式和序列程式必須分別使用不同的程式編號。若編號相同，將使得 MPE720 出現錯誤。
2. MP3000 系列可同時執行的運動程式數為 32 個。
   若同時執行 33 個以上的程式，系統將發出運動程式警報 (無系統工作錯誤)。
   無系統工作錯誤：運動程式狀態旗標 -Bit E

補充　本手冊係利用 DWG.H 來表示階梯圖程式高速處理圖面。

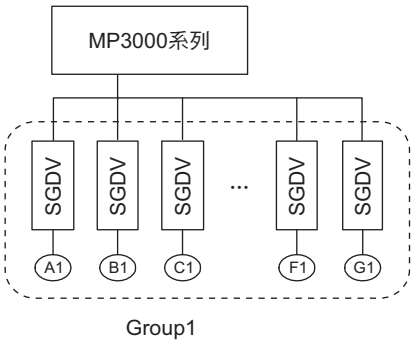
1.5 運動程式群組

運動程式提供了將動作相關轉軸整合為同一個群組，而且可依群組別編寫程式的功能。因此，僅需 1 台運動控制器，即可獨立控制多台機器。群組運轉包含單一群組運轉及多群組運轉。

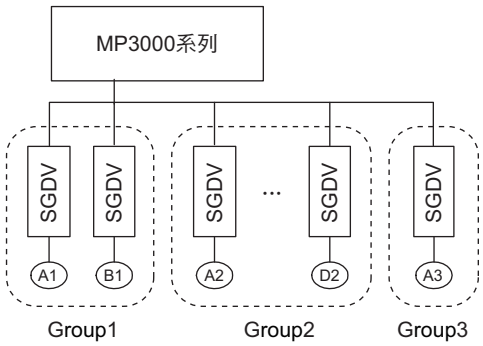
轉軸如何整合為群組，請利用「群組定義」功能加以定義。如欲設定群組定義，請參閱以下章節。

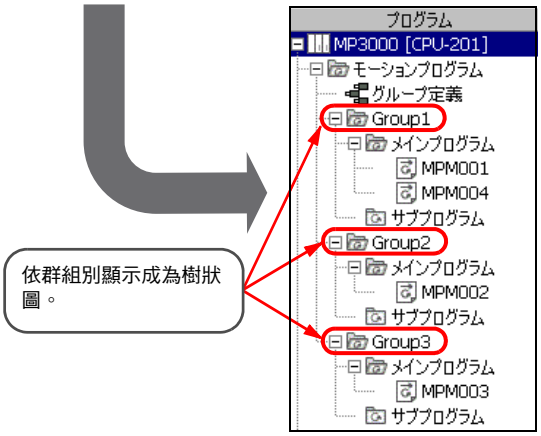
☞ 5.2 群組定義說明 (第 5-9 頁)

■ 單一群組運轉



■ 多群組運轉





1.6 運動程式執行時間

運動程式可先和 MP3000 系列的高速掃描完全同步後再進行處理作業。

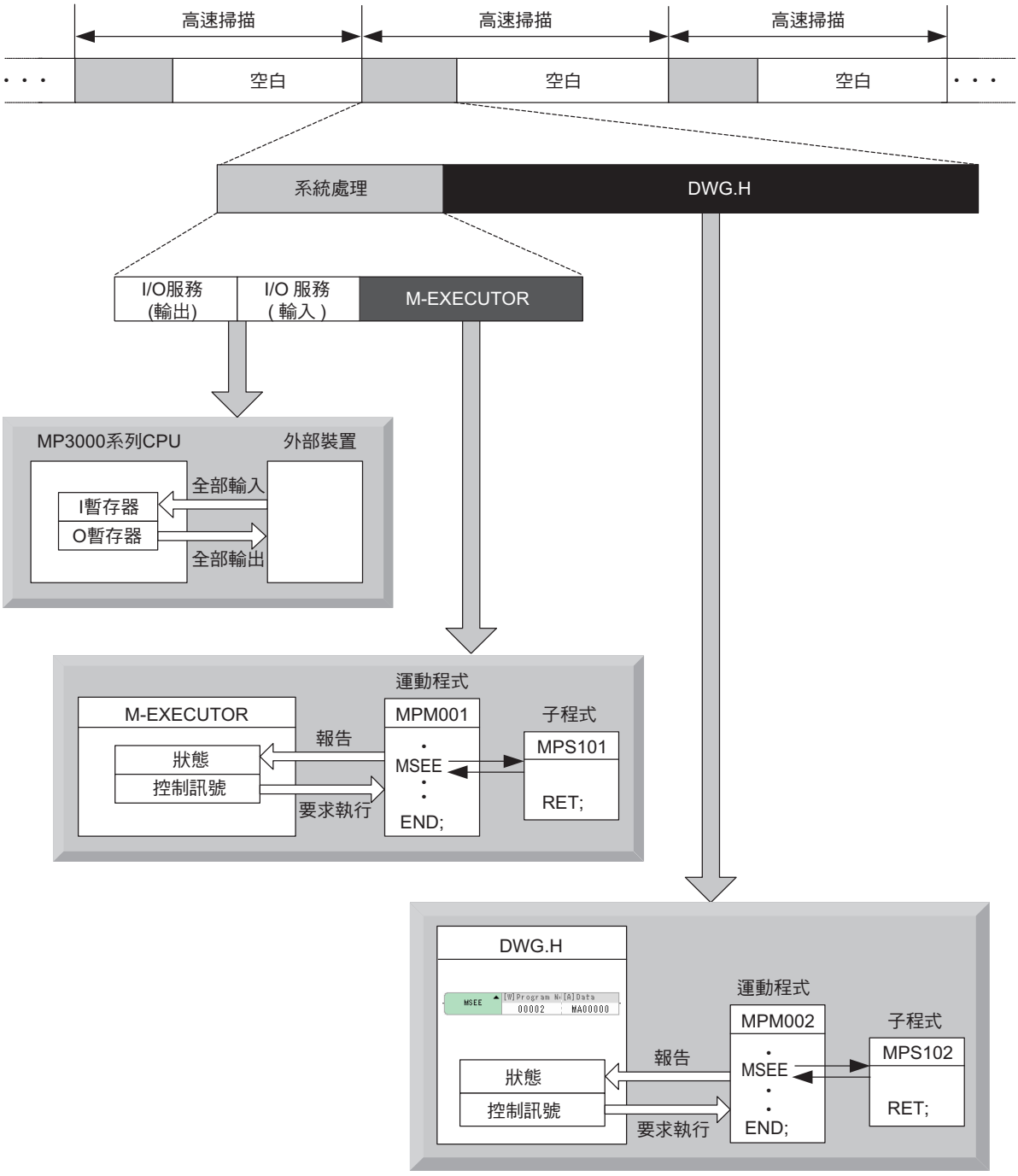
高速掃描時，系統會先執行 I/O 服務，接著再執行已登錄於 M-EXECUTOR 的運動程式，最後會在執行已嵌入 DWG.H 的 MSEE 指令時，同時執行運動程式。

專有名詞解說

I/O 服務

I/O服務係指利用MP3000系列的CPU單元/CPU模組和外部裝置(各種選配模組)，來執行資料輸出入的處理作業。

下圖為運動程式的執行時間。



1.7 執行運動程式

接下來將說明運動程式的執行方法。

執行方式

執行運動程式前，必須將您所編寫的程式登錄到系統中。登錄完成後，運動程式就會依照 H 掃描的週期進行參照。

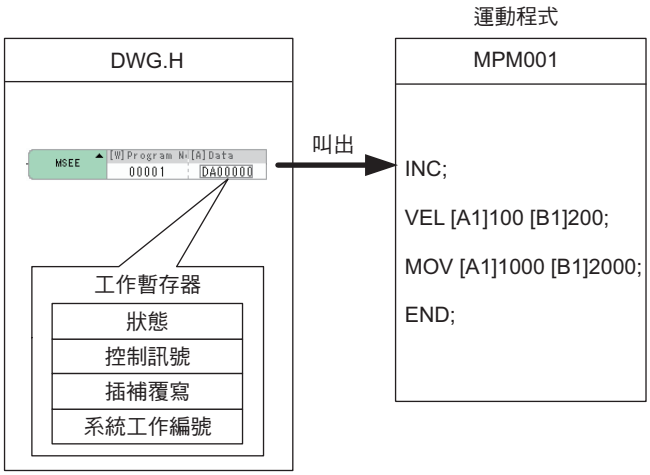
系統登錄方式包含以下 2 種。

・利用階梯圖程式的 MSEE 指令來叫出

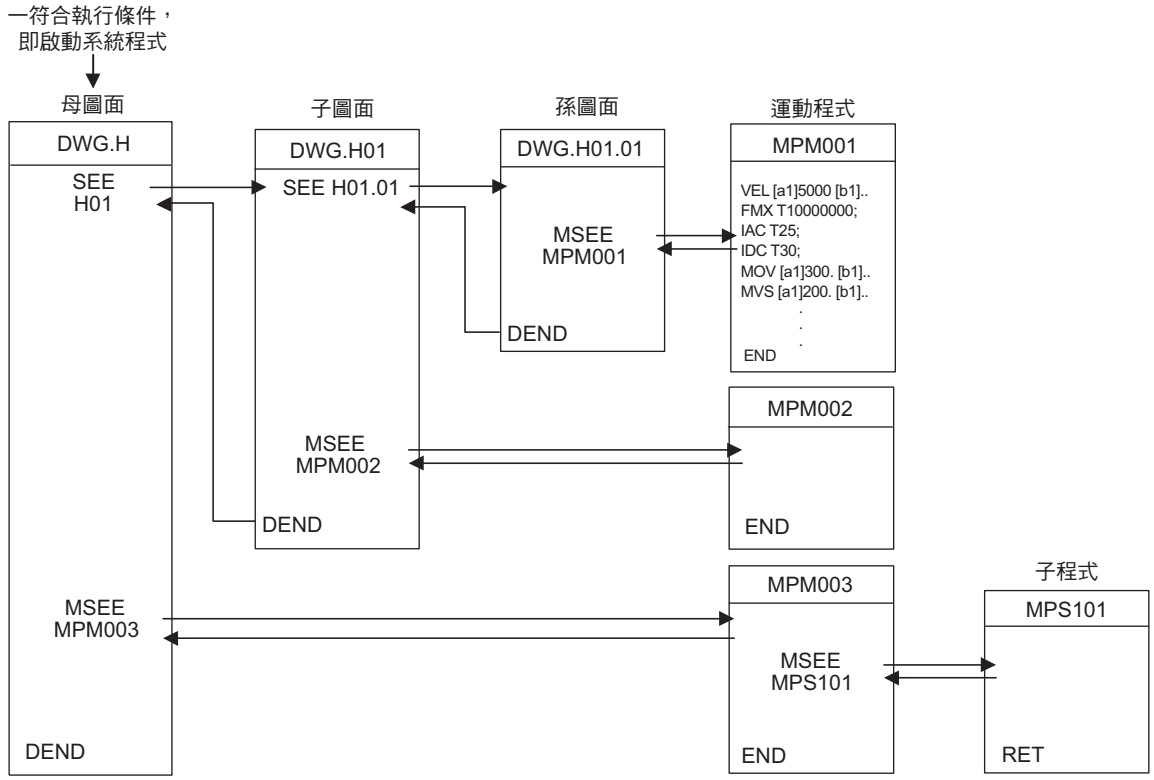
・利用 M-EXECUTOR 程式的執行定義來叫出

利用階梯圖程式的 MSEE 指令來叫出

運動程式編寫完成後，可將 MSEE 指令 (運動程式叫出指令) 嵌入 DWG.H 中。



若使用 H 圖面，則可由母圖面、子圖面、孫圖面等其中任一個圖面來執行。
下圖為執行範例。



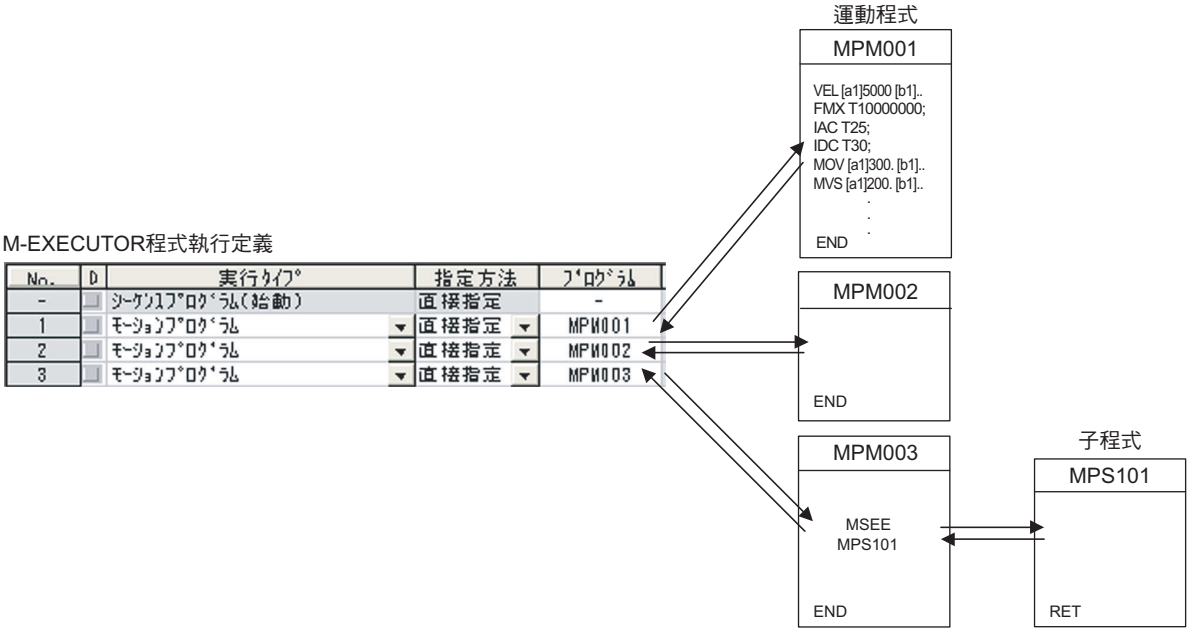
H 圖面的階梯圖指令係依照高速掃描週期，以及母圖面 - 子圖面 - 孫圖面的階層架構依序執行。
上述方法為運動程式執行前的準備程序。當您嵌入 MSEE 指令時，運動程式將不會啟動。嵌入 MSEE 指令後，只要利用控制訊號將要求程式開始運轉設定為開啟，即可啟動運動程式。
運動程式會依照掃描週期來執行動作，而階梯圖程式則是在每次掃描時執行所有的程式。運動程式係透過系統來執行控制。

註記

執行運動程式時，必須注意以下幾點。
- 登錄於 M-EXECUTOR 的運動程式，無法透過 MSEE 指令來執行。
- MSEE 指令無法執行多個編號相同的運動程式。
- 階梯圖程式的 MSEE 指令無法用來執行子程式 (MPS□□□)。
  僅能利用運動程式 (MPM□□□、MPS□□□) 進行參照。
- 階梯圖程式的 MSEE 指令無法用來執行序列程式 (SPM□□□、SPS□□□)。

利用 M-EXECUTOR 程式執行定義的叫出方式

運動程式編寫完成後，必須登錄於 M-EXECUTOR 程式的執行定義中。
登錄完成的運動程式若要執行開始 / 停止要求等操作動作，必須透過 M-EXECUTOR 控制暫存器 (I/O 暫存器)。
依照編號由小到大的順序來執行登錄於 M-EXECUTOR 程式執行定義中的程式。
下圖為執行範例。



完成 M-EXECUTOR 程式執行定義的登錄後，這時候只要利用控制訊號將要求程式開始運轉設定為開啟，即可啟動運動程式。
登錄於 M-EXECUTOR 的運動程式會依照掃描週期來執行動作，而階梯圖程式則是在每次掃描時執行所有的程式。運動程式係透過系統來執行控制。

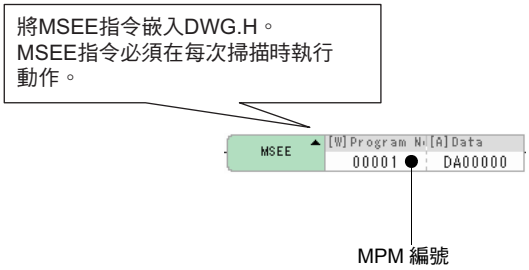
註記

將程式登錄於 M-EXECUTOR 時，必須注意以下幾點。
- 請勿登錄多個編號相同的運動程式。
- 系統無法使用間接指定功能，參照多個編號相同的運動程式。

程式執行登錄的方法

程式執行登錄的方法有 2 種。
本手冊係以 MPM001 執行登錄為例。

將 MSEE 指令嵌入階梯圖程式



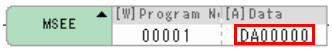
M-EXECUTOR 登錄方法

工作暫存器

完成程式登錄後，運動程式為了執行控制 / 執行監控，會對工作暫存器進行配置。工作暫存器的功用就是讓運動控制器控制專用程式對運動程式下達指令，或是取得運動程式狀態。

■ 從階梯圖程式利用 MSEE 指令叫出運動程式時

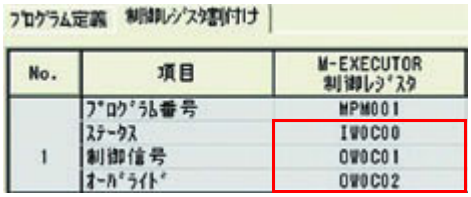
指定為 MSEE 指令的 Data 的暫存器 (MA□□□□□ 或 DA□□□□□) 其起始的 4 個字元將被當作工作暫存器使用。



工作暫存器	左圖所示的範例	內容	輸出入
第1個字元	DW00000	狀態旗標	OUT
第2個字元	DW00001	控制訊號	IN
第3個字元	DW00002	插補用覆寫	IN
第4個字元	DW00003	系統工作編號	IN

■ 若運動程式已被登錄於 M-EXECUTOR 程式的執行定義中

M-EXECUTOR 控制暫存器將被當作工作暫存器使用。
M-EXECUTOR 控制暫存器將由系統自動進行配置。



工作暫存器 (M-EXECUTOR 控制暫存器)	左圖所示的範例	內容	輸出入
狀態	IW0C00	狀態旗標	OUT
控制訊號	OW0C01	控制訊號	IN
覆寫	OW0C02	插補用覆寫	IN

狀態旗標

運動程式的狀態旗標可用來瞭解運動程式的執行狀態。

下表為狀態旗標之詳細說明。

Bit No	名稱	內容
Bit 0	程式運轉中	運動程式運轉時，此位元將變為 1。 0：運動程式停止中 1：運動程式運轉中
Bit 1	程式暫停中	利用程式暫停狀態要求，讓運動程式進入暫停狀態時，此位元將變為 1。 當程式暫停狀態要求的控制訊號輸入後，系統會先確認轉軸停止減速並停止動作後，再將狀態旗標設定為開啟。 0：因暫停要求而進入停止以外狀態 1：因暫停要求而進入停止狀態中
Bit 2	因程式要求停止而進入停止狀態	當運動程式因停止要求而停止執行動作時，此位元將變為 1。 0：因停止要求而進入停止以外狀態 1：因停止要求而進入停止中
Bit 3	(系統預約)	–
Bit 4	程式單一區塊停止運轉中	利用除錯運轉讓單一區進入停止狀態時，此位元將變為 1。 0：單一區塊停止以外狀態 1：單一區塊停止中
Bit 5	(系統預約)	–
Bit 6	(系統預約)	–
Bit 7	(系統預約)	–
Bit 8	程式目前發生錯誤	當程式發生警報時，此位元將變為 1。 當此位元為 1 時，錯誤詳細內容將被反映在錯誤資訊畫面及 S 暫存器中。 0：未發生程式錯誤 1：程式目前發生錯誤
Bit 9	因斷點而進入停止狀態	在除錯運轉中因斷點而入停止狀態時，此位元將變為 1。 0：斷點進入停止以外狀態 1：斷點停止狀態
Bit A	(系統預約)	–
Bit B	除錯模式中	利用除錯運轉來執行程式運轉時，此位元將變為 1。 0：進入除錯模式運轉以外狀態 (一般運轉狀態) 1：除錯模式運轉狀態
Bit C	程式類別	回報目前正在執行的程式為運動程式或序列程式。 0：運動程式 1：序列程式
Bit D	要求啟動之歷程紀錄	當程式要求開始運轉被設定為開啟時，此位元將變為 1。 0：要求程式開始運轉 OFF 1：要求程式開始運轉 ON
Bit E	無系統工作錯誤、執行掃描異常	若無法取得執行運動程式所需的系統工作，或是將 MSEE 指令嵌入 DWG.H 以外位置時，此位元將變為 1。 0：未發生 「無系統工作錯誤、執行掃描異常」 1：目前發生 「無系統工作錯誤、執行掃描異常」
Bit F	主程式編號錯誤	當您所指定的運動程式編號超出範圍時，此位元將變為 1。 運動程式編號範圍：1 ~ 512 0：未發生運動程式編號指定異常 1：運動程式編號指定異常

(註) 一旦發生運動程式警報，程式錯誤內容就會被反映在錯誤資訊畫面及 S 暫存器中。

控制訊號

執行運動程式前，必須先輸入程式控制訊號 (要求程式啟動運轉或要求程式停止動作)。下表為運動程式控制訊號。

⊣ ⊢：此標誌的訊號表示系統必須在接受訊號前維持開啟狀態。

⊣ ⊢：此標誌的訊號表示只要在進行 1 次高速掃描時開啟即可。

Bit No	名稱	內容
Bit 0 ⊣ ⊢ ⊣ ⊢	要求程式開始運轉	可用來要求運動程式啟動。當此位元由 0 變為 1 時，運動程式就會啟動。不過，當運動程式發生警報時，此位元的操作動作將被視為無效。 0：要求程式開始運轉 OFF 1：要求程式開始運轉 ON
Bit 1 ⊣ ⊢	程式要求暫停	可用來要求運轉中的運動程式暫停動作。 進入暫停狀態後，只要解除程式的暫停要求，即可讓停止狀態下的程式重新執行動作。 0：要求程式暫停 OFF (解除暫停) 1：程式要求暫停 ON
Bit 2 ⊣ ⊢	程式要求停止動作	可用來要求運轉中的運動程式停止動作。 若在轉軸執行動作時，將此位元設定為 1，運動程式將發生警報。 0：要求停止程式動作 OFF 1：要求停止程式動作 ON
Bit 3 ⊣ ⊢	選擇程式單一區塊模式	可用來要求選擇程式單一區塊模式。 可取代除錯運轉功能。 0：選擇程式單一區塊模式 OFF 1：選擇程式單一區塊模式 ON
Bit 4 ⊣ ⊢ ⊣ ⊢	要求程式單一區塊啟動	當此位元由 0 變為 1 時，程式將執行單一區塊 (步進執行)。本位元僅在控制訊號 Bit 3 (選擇程式單一區塊模式) 為 1 時有效。 0：要求程式單一區塊啟動 OFF 1：要求程式單一區塊啟動 ON
Bit 5 ⊣ ⊢	要求重置程式及警報	可用來重置運動程式及警報。 0：要求重置程式及警報 OFF 1：要求重置程式及警報 ON
Bit 6 ⊣ ⊢ ⊣ ⊢	要求程式開始持續運轉	利用程式停止要求，即可讓目前停止的程式持續運轉。 0：要求程式開始持續運轉 OFF 1：要求程式開始持續運轉 ON
Bit 7	(系統預約)	–
Bit 8 ⊣ ⊢	略過 (Skip)1 資訊	利用 SKP 指令 (對略過輸入訊號選擇下達 「SS1」指令) 讓轉軸在移動中時，只要此位元變為 1，移動中的轉軸就會減速停止，並且取消剩餘的移動量指令。 0：略過 1 訊號 OFF 1：略過 1 訊號 ON
Bit 9 ⊣ ⊢	略過 (Skip)2 資訊	利用 SKP 指令 (對略過輸入訊號選擇下達 「SS2」指令) 讓轉軸在移動中時，只要此位元變為 1，移動中的轉軸就會減速停止，並且取消剩餘的移動量指令。 0：略過 2 訊號 OFF 1：略過 2 訊號 ON
Bit A, B	(系統預約)	–
Bit C	(系統預約)	–
Bit D ⊣ ⊢	設定系統工作編號 *1	指定系統工作編號時，此位元將變為 1。 0：不指定系統工作編號 1：指定系統工作編號

(續下頁)

(續上頁)

Bit No	名稱	內容
Bit E 	設定插補覆寫 *2	指定插補覆寫時，此位元將變為 1。 0：不指定插補用覆寫 1：指定插補用覆寫
Bit F	(系統預約)	–

*1. 設定系統工作編號
- 若運動程式已被登錄在 M-EXECUTOR
  無法指定。使用與系統定義編號相同的系統工作編號。
- 利用階梯圖程式的 MSEE 指令來叫出運動程式
  關閉：使用系統自動取得的系統工作。系統工作的編號每次各不相同。
  開啟：使用已設定的系統工作編號之工件。
  不過，若指定由 M-EXECUTOR 佔用的工作，系統將回報「Bit E：無系統工作錯誤」的狀態。

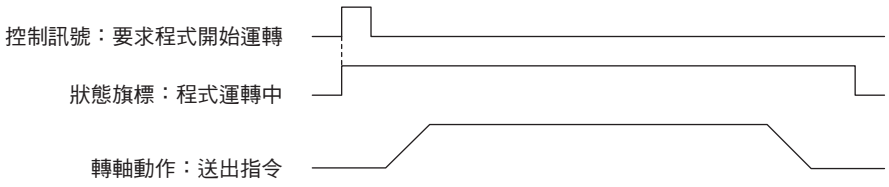
*2. 設定插補覆寫
  關閉：插補用覆寫被固定為 100%
  開啟：依照您所設定的插補用覆寫。

(註)1. 輸入階梯圖程式時必須使用適合的訊號類型。
2. 開啟電源後，程式將會執行要求程式運轉被設定為開啟狀態時之編號。

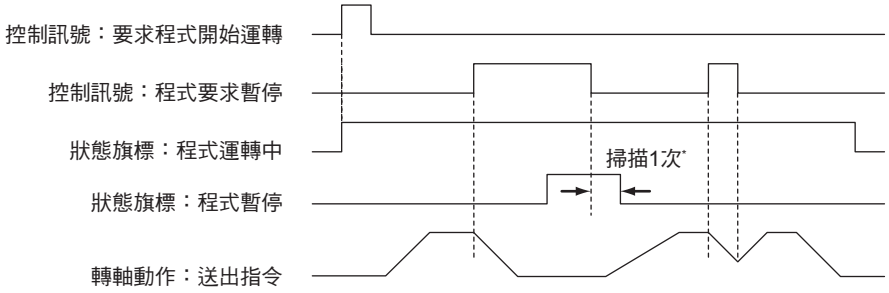
■ 運動程式控制訊號時間圖

下圖為控制訊號輸入後的狀態旗標及轉軸動作時間圖範例。

· 要求程式開始運轉時

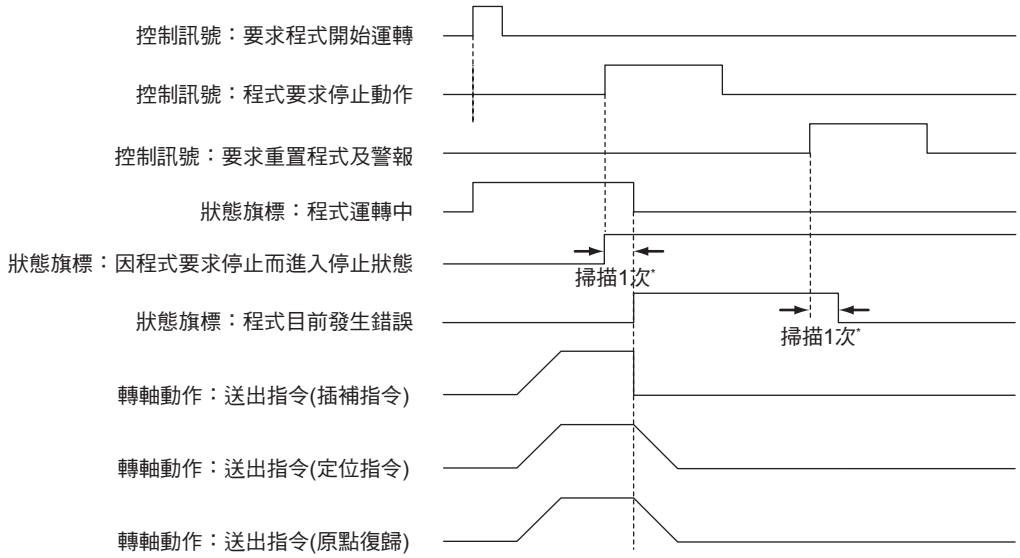


· 要求暫停時



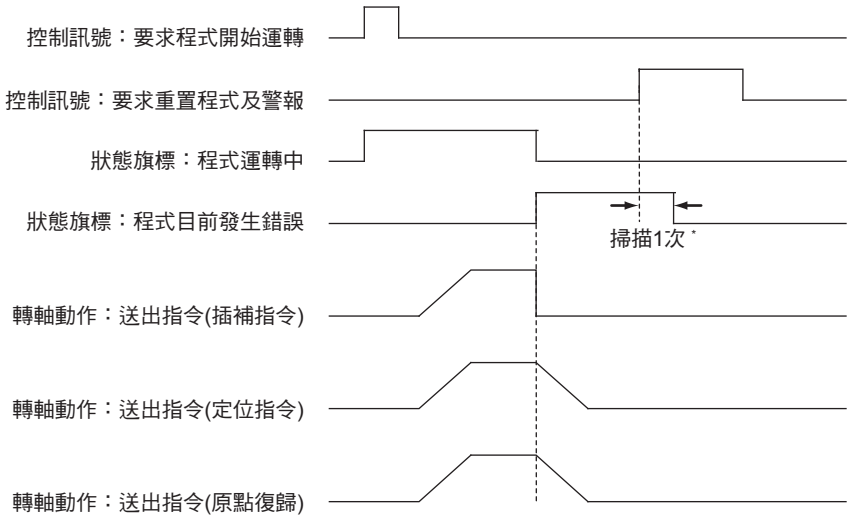
* 當控制訊號輸入時，狀態旗標會在每次掃描結束後更新。

· 要求停止動作時



* 當控制訊號輸入時，狀態旗標會在每次掃描結束後更新。

· 運動程式發生警報時



* 當控制訊號輸入時，狀態旗標會在每次掃描結束後更新。

重要

1. 利用運動語言指令來啟動轉軸時，一旦要求停止的控制訊號被設定為開啟，就會發出警報。
2. 若插補類運動語言指令在移動時送出要求停止訊號，轉軸將立刻停止動作。
   若想進行減速停止，必須使用要求暫停訊號。
3. 執行原點復歸 (ZRN) 時，無法同時接受暫停訊號。
   若要讓機器停止動作，請使用要求停止訊號。
4. 一旦運動語言指令在轉軸移動時發出運動程式警報，轉軸將立刻停止動作。

運動程式控制專用的程式範例請參閱以下章節。

☞ **B.1 運動程式控制專用程式 (第 B-2 頁)**

插補用覆寫

所謂的插補覆寫係指針對插補類運動語言指令所下達的轉軸移動速度指令，變更其輸出比率。

設定插補指令 (MVS、MCW、MCC、SKP) 執行時之覆寫。

只有當控制訊號 Bit E (設定插補用覆寫) 被設定為開啟時，插補用覆寫才有效。

插補覆寫值的指令範圍：0 ~ 32767

單位：1 = 0.01%

系統工作編號

利用階梯圖程式 MSEE 指令來執行運動程式時，必須先設定好用來叫出運動程式的系統工作編號。系統工作編號僅在控制訊號 Bit D (設定系統工作編號) 被設定為開啟時有效。

設定範圍：1 ~ 32

重要

同時使用階梯圖程式的 MSEE 指令和 M-EXECUTOR 時，請勿在階梯圖程式的 MSEE 指令中指定 M-EXECUTOR 專用的系統工作編號。否則，將發生無系統工作錯誤。
M-EXECUTOR 專用的系統工作編號：0 ~ 程式定義數量的設定值

補充　使用 M-EXECUTOR 時，無法再設定系統工作編號。所使用的系統工作編號需和定義編號一致。

1.8 高階使用方法

接下來將說明運動程式實際的執行方法。

利用暫存器間接指定程式編號

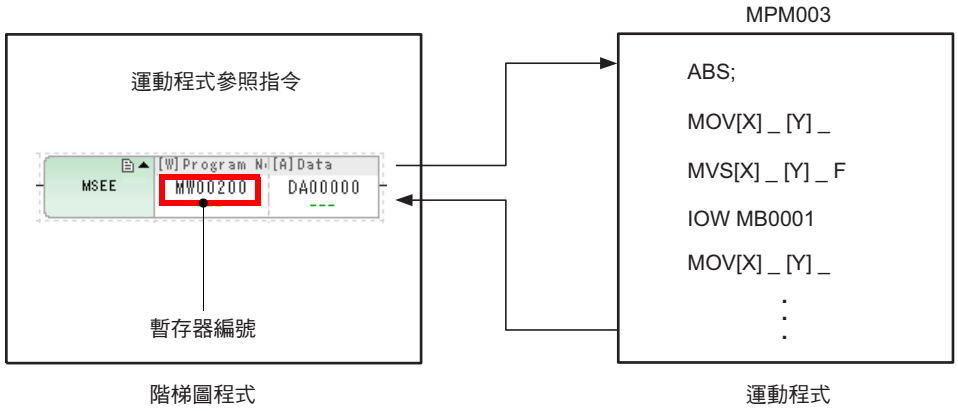
使用暫存器，即可叫出和暫存器所儲存數值一致的運動程式。

叫出運動程式的方法有 2 種。

利用階梯圖程式的 MSEE 指令叫出運動程式時

指定使用間接指定為 MSEE 指令的 Program No. 之任意暫存器 (M 暫存器、G 暫存器、或 D 暫存器)。

將 3 儲存在 MW00200，即可叫出程式 (MPM003)。



利用 M-EXECUTOR 程式執行定義來叫出運動程式

選擇 「間接指定」作為指定方法。系統就會自動配置適用於間接指定的暫存器。

將 3 儲存在 OW0C00，即可叫出程式 (MPM003)。

利用外部裝置直接控制運動程式

M-EXECUTOR 內置了可將 M-EXECUTOR 控制用暫存器配置到任一個暫存器的功能。

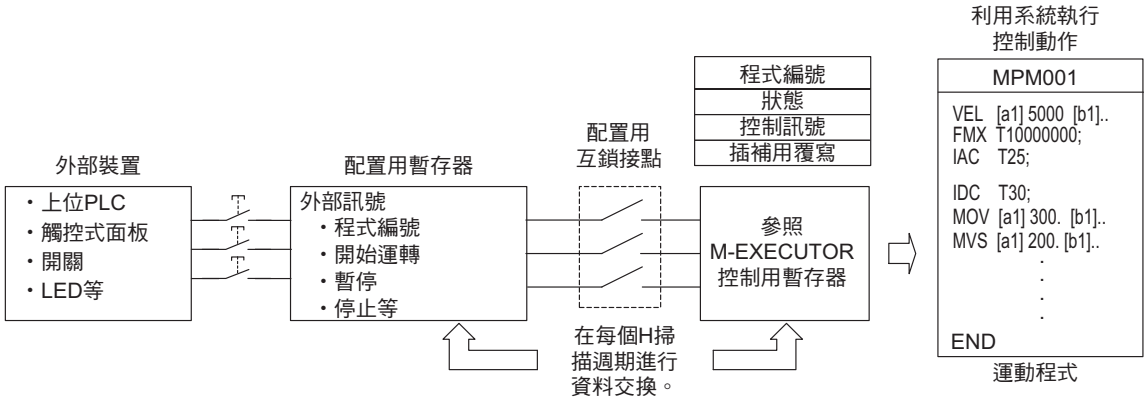
只要使用此功能，即可讓 M-EXECUTOR 控制用暫存器和連接到外部裝置的 I/O 暫存器，彼此自動進行資料交換。

接下來將說明如何利用外部裝置直接控制運動程式之設定範例。

<M-EXECUTOR暫存器配置視窗>

No.	項目	M-EXECUTOR 制御レジスタ	割付け DISABLE	方向	割付けレジスタ	割付け インターロック接点
	プログラム番号	MPM001				
	ステータス	IW0C00		->	OW0000	IB00020
1	制御信号	OW0C01		<-	IW0000	IB00020
	オーバライド	OW0C02		<-	IW0001	IB00020

可為配置用暫存器、配置用互鎖接點設定任意的暫存器。



配置用互鎖接點具有運動程式的動作互鎖功能。設定配置用暫存器前，必須先設定號配置用互鎖接點。

當配置用互鎖的接點被設定為開啟 / 關閉後，即可執行下列處理作業。

- 當配置用互鎖接點被設定為開啟時，配置用暫存器和 M-EXECUTOR 控制用暫存器就會在每個 H 掃描週期進行資料交換，此時運動程式將進入執行可能狀態。
- 當配置用互鎖接點被設定為關閉時，配置用暫存器和 M-EXECUTOR 控制用暫存器無法進行資料交換，此時系統將無法執行運動程式。
- 在運動程式執行狀態下，將配置用互鎖接點從開啟切換為關閉後，正在執行的運動程式就會停止動作，而動作中的轉軸也會停止。此時運動程式警報將進入 1Bh (緊急停止指令執行中) 狀態，而狀態旗標 Bit 8 (目前發生程式警報) 則會變為開啟。

若要重新執行運動程式，請使用以下操作步驟。

1. **將配置用互鎖接點由關閉設定為開啟。**
2. **將控制訊號 Bit 5 (要求重置程式及警報) 設定為開啟。**
3. **確認狀態旗標 Bit 8 (目前發生程式警報) 是否關閉。**
4. **將控制訊號 Bit 5 (要求重置程式及警報) 設定為關閉。**
5. **將控制訊號 Bit 0 (要求程式開始運轉) 設定為開啟。**

監控運動程式執行資訊

利用 S 暫存器 (SW03200 ~ SW05119、SW08192 ~ SW09215)，即可監控運動程式的執行資訊。

執行資訊的監控方法依用途而異，第一種是利用 MSEE 指令來叫出運動程式，另一種則是利用 M-EXECUTOR 程式執行定義來登錄運動程式。
接下來將說明不同用途的監控方法。

利用階梯圖程式的 MSEE 指令叫出運動程式時

利用階梯圖程式的 MSEE 指令來叫出運動程式時，運動程式控制訊號 Bit D (設定系統工作編號) 的設定不同，將出現以下差異。

- **運動程式的控制訊號 Bit D (設定系統工作編號) 被設定為開啟時**
  執行資訊會被回報到 SW03264 ~ SW05119、SW08192 ~ SW09215 (工作 n 使用程式資訊)。

  例如，當系統工作編號為 1 時，只要利用 SW03264 ~ SW03321 (工作 1 使用程式資訊)，即可監控運動程式的執行資訊。

- **運動程式的控制訊號 Bit D (設定系統工作編號) 被設定為關閉時**
  系統將自動決定適用之系統工作。
  請參照 SW03200 ~ SW03231 (執行中的程式編號)，即可確認適用的工作為何。

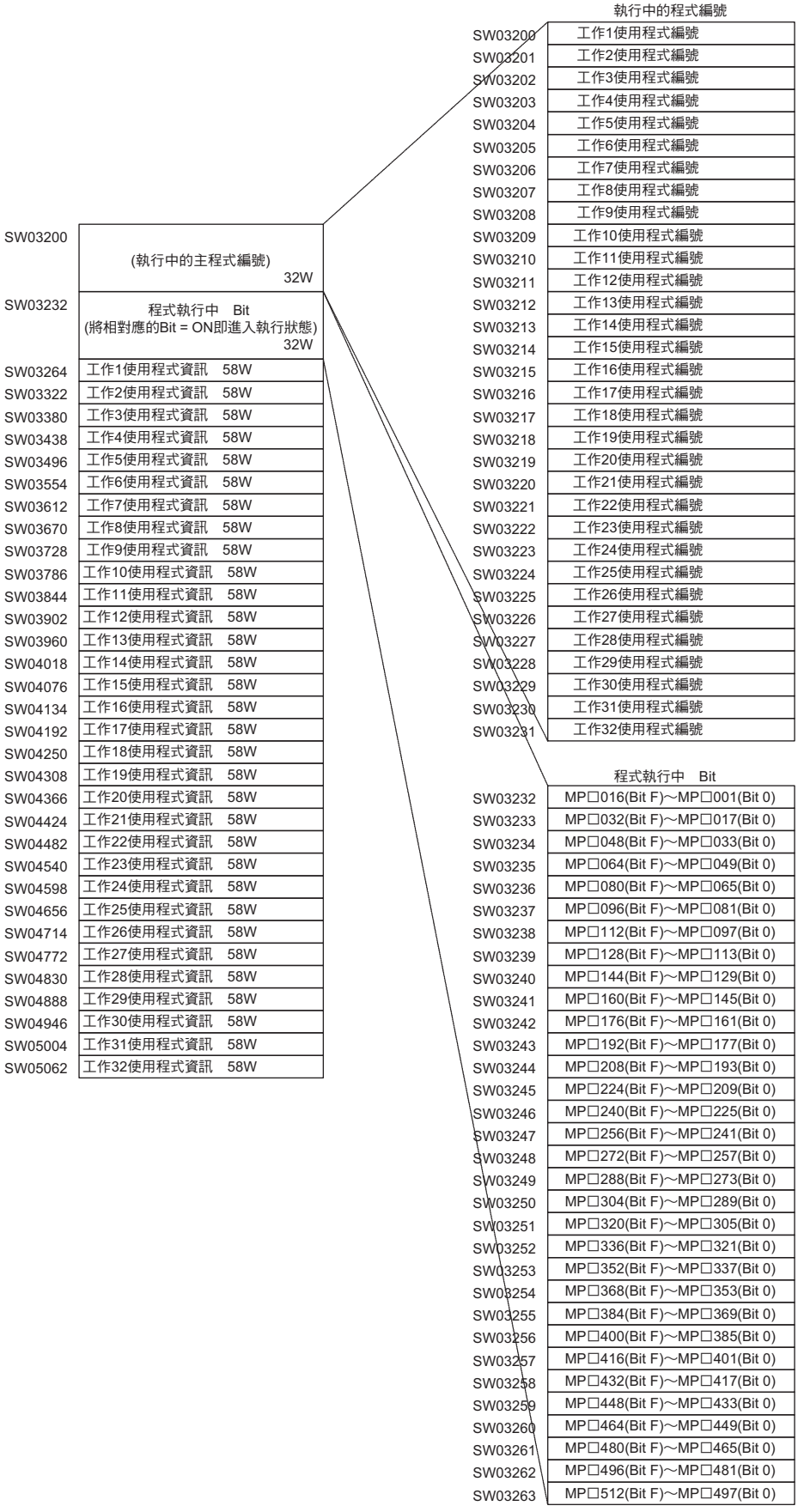
  例如，若您想要監控的運動程式為 MPM001，且 SW03202 為 1，那麼適用的暫存器工作編號則為 3。此時，SW03380 ~ SW03437 (工作 3 使用程式資訊) 即可用來監控運動程式的執行資訊。

若運動程式已被登錄於 M-EXECUTOR 程式的執行定義中

若將運動程式登錄於 M-EXECUTOR 程式執行定義中，那麼適用的系統工作編號將和登錄於 M-EXECUTOR 裡的程式登錄編號相同。

例如，假設運動程式被登錄在程式登錄編號 3，那麼適用的系統工作編號就是 3。此時，SW03380 ~ SW03437 (工作 3 使用程式資訊) 即可用來監控運動程式的執行資訊。

◆ 運動程式執行資訊中的暫存器區



(註)SW03232 ~ SW03263 的 MP□ 的□代表「M」或「S」。

◆ 系統工作編號 1 ~ 32 的使用暫存器清單

以下為系統工作編號 1 ~ 32 的使用暫存器清單。
暫存器清單的警報碼中記載了 2 個系統暫存器，建議您最好利用 SL26□□□ 系統暫存器來做確認。( ) 符號的系統暫存器雖然能用來確認警報，不過部分警報可能會出現無法確認的情形。
如欲進一步了解警報碼，請參閱以下手冊。

📖 MP3000 系列　MP3200/MP3300　故障排除手冊 (資料編號：SIJP C880725 01)

· 系統工作編號 1 ~ 8

系統工作編號		工作 1	工作 2	工作 3	工作 4	工作 5	工作 6	工作 7	工作 8
執行中的主程式編號		SW03200	SW03201	SW03202	SW03203	SW03204	SW03205	SW03206	SW03207
狀態		SW03264	SW03322	SW03380	SW03438	SW03496	SW03554	SW03612	SW03670
控制訊號		SW03265	SW03323	SW03381	SW03439	SW03497	SW03555	SW03613	SW03671
並列 0	程式編號	SW03266	SW03324	SW03382	SW03440	SW03498	SW03556	SW03614	SW03672
	區塊編號	SW03267	SW03325	SW03383	SW03441	SW03499	SW03557	SW03615	SW03673
	警報碼	SL26000 (SW03268)	SL26016 (SW03326)	SL26032 (SW03384)	SL26048 (SW03442)	SL26064 (SW03500)	SL26080 (SW03558)	SL26096 (SW03616)	SL26112 (SW03674)
並列 1	程式編號	SW03269	SW03327	SW03385	SW03443	SW03501	SW03559	SW03617	SW03675
	區塊編號	SW03270	SW03328	SW03386	SW03444	SW03502	SW03560	SW03618	SW03676
	警報碼	SL26002 (SW03271)	SL26018 (SW03329)	SL26034 (SW03387)	SL26050 (SW03445)	SL26066 (SW03503)	SL26082 (SW03561)	SL26098 (SW03619)	SL26114 (SW03677)
並列 2	程式編號	SW03272	SW03330	SW03388	SW03446	SW03504	SW03562	SW03620	SW03678
	區塊編號	SW03273	SW03331	SW03389	SW03447	SW03505	SW03563	SW03621	SW03679
	警報碼	SL26004 (SW03274)	SL26020 (SW03332)	SL26036 (SW03390)	SL26052 (SW03448)	SL26068 (SW03506)	SL26084 (SW03564)	SL26100 (SW03622)	SL26116 (SW03680)
並列 3	程式編號	SW03275	SW03333	SW03391	SW03449	SW03507	SW03565	SW03623	SW03681
	區塊編號	SW03276	SW03334	SW03392	SW03450	SW03508	SW03566	SW03624	SW03682
	警報碼	SL26006 (SW03277)	SL26022 (SW03335)	SL26038 (SW03393)	SL26054 (SW03451)	SL26070 (SW03509)	SL26086 (SW03567)	SL26102 (SW03625)	SL26118 (SW03683)
並列 4	程式編號	SW03278	SW03336	SW03394	SW03452	SW03510	SW03568	SW03626	SW03684
	區塊編號	SW03279	SW03337	SW03395	SW03453	SW03511	SW03569	SW03627	SW03685
	警報碼	SL26008 (SW03280)	SL26024 (SW03338)	SL26040 (SW03396)	SL26056 (SW03454)	SL26072 (SW03512)	SL26088 (SW03570)	SL26104 (SW03628)	SL26120 (SW03686)
並列 5	程式編號	SW03281	SW03339	SW03397	SW03455	SW03513	SW03571	SW03629	SW03687
	區塊編號	SW03282	SW03340	SW03398	SW03456	SW03514	SW03572	SW03630	SW03688
	警報碼	SL26010 (SW03283)	SL26026 (SW03341)	SL26042 (SW03399)	SL26058 (SW03457)	SL26074 (SW03515)	SL26090 (SW03573)	SL26106 (SW03631)	SL26122 (SW03689)
並列 6	程式編號	SW03284	SW03342	SW03400	SW03458	SW03516	SW03574	SW03632	SW03690
	區塊編號	SW03285	SW03343	SW03401	SW03459	SW03517	SW03575	SW03633	SW03691
	警報碼	SL26012 (SW03286)	SL26028 (SW03344)	SL26044 (SW03402)	SL26060 (SW03460)	SL26076 (SW03518)	SL26092 (SW03576)	SL26108 (SW03634)	SL26124 (SW03692)
並列 7	程式編號	SW03287	SW03345	SW03403	SW03461	SW03519	SW03577	SW03635	SW03693
	區塊編號	SW03288	SW03346	SW03404	SW03462	SW03520	SW03578	SW03636	SW03694
	警報碼	SL260014 (SW03289)	SL26030 (SW03347)	SL26046 (SW03405)	SL26062 (SW03463)	SL26078 (SW03521)	SL26094 (SW03579)	SL26110 (SW03637)	SL26126 (SW03695)
邏輯軸 #1 程式現在位置		SL03290	SL03348	SL03406	SL03464	SL03522	SL03580	SL03638	SL03696
邏輯軸 #2 程式現在位置		SL03292	SL03350	SL03408	SL03466	SL03524	SL03582	SL03640	SL03698
邏輯軸 #3 程式現在位置		SL03294	SL03352	SL03410	SL03468	SL03526	SL03584	SL03642	SL03700
邏輯軸 #4 程式現在位置		SL03296	SL03354	SL03412	SL03470	SL03528	SL03586	SL03644	SL03702
邏輯軸 #5 程式現在位置		SL03298	SL03356	SL03414	SL03472	SL03530	SL03588	SL03646	SL03704

(續下頁)

(續上頁)

系統工作編號	工作 1	工作 2	工作 3	工作 4	工作 5	工作 6	工作 7	工作 8
邏輯軸 #6 程式現在位置	SL03300	SL03358	SL03416	SL03474	SL03532	SL03590	SL03648	SL03706
邏輯軸 #7 程式現在位置	SL03302	SL03360	SL03418	SL03476	SL03534	SL03592	SL03650	SL03708
邏輯軸 #8 程式現在位置	SL03304	SL03362	SL03420	SL03478	SL03536	SL03594	SL03652	SL03710
邏輯軸 #9 程式現在位置	SL03306	SL03364	SL03422	SL03480	SL03538	SL03596	SL03654	SL03712
邏輯軸 #10 程式現在位置	SL03308	SL03366	SL03424	SL03482	SL03540	SL03598	SL03656	SL03714
邏輯軸 #11 程式現在位置	SL03310	SL03368	SL03426	SL03484	SL03542	SL03600	SL03658	SL03716
邏輯軸 #12 程式現在位置	SL03312	SL03370	SL03428	SL03486	SL03544	SL03602	SL03660	SL03718
邏輯軸 #13 程式現在位置	SL03314	SL03372	SL03430	SL03488	SL03546	SL03604	SL03662	SL03720
邏輯軸 #14 程式現在位置	SL03316	SL03374	SL03432	SL03490	SL03548	SL03606	SL03664	SL03722
邏輯軸 #15 程式現在位置	SL03318	SL03376	SL03434	SL03492	SL03550	SL03608	SL03666	SL03724
邏輯軸 #16 程式現在位置	SL03320	SL03378	SL03436	SL03494	SL03552	SL03610	SL03668	SL03726
邏輯軸 #17 程式現在位置	SL08192	SL08224	SL08256	SL08288	SL08320	SL08352	SL08384	SL08416
邏輯軸 #18 程式現在位置	SL08194	SL08226	SL08258	SL08290	SL08322	SL08354	SL08386	SL08418
邏輯軸 #19 程式現在位置	SL08196	SL08228	SL08260	SL08292	SL08324	SL08356	SL08388	SL08420
邏輯軸 #20 程式現在位置	SL08198	SL08230	SL08262	SL08294	SL08326	SL08358	SL08390	SL08422
邏輯軸 #21 程式現在位置	SL08200	SL08232	SL08264	SL08296	SL08328	SL08360	SL08392	SL08424
邏輯軸 #22 程式現在位置	SL08202	SL08234	SL08266	SL08298	SL08330	SL08362	SL08394	SL08426
邏輯軸 #23 程式現在位置	SL08204	SL08236	SL08268	SL08300	SL08332	SL08364	SL08396	SL08428
邏輯軸 #24 程式現在位置	SL08206	SL08238	SL08270	SL08302	SL08334	SL08366	SL08398	SL08430
邏輯軸 #25 程式現在位置	SL08208	SL08240	SL08272	SL08304	SL08336	SL08368	SL08400	SL08432
邏輯軸 #26 程式現在位置	SL08210	SL08242	SL08274	SL08306	SL08338	SL08370	SL08402	SL08434
邏輯軸 #27 程式現在位置	SL08212	SL08244	SL08276	SL08308	SL08340	SL08372	SL08404	SL08436
邏輯軸 #28 程式現在位置	SL08214	SL08246	SL08278	SL08310	SL08342	SL08374	SL08406	SL08438
邏輯軸 #29 程式現在位置	SL08216	SL08248	SL08280	SL08312	SL08344	SL08376	SL08408	SL08440
邏輯軸 #30 程式現在位置	SL08218	SL08250	SL08282	SL08314	SL08346	SL08378	SL08410	SL08442
邏輯軸 #31 程式現在位置	SL08220	SL08252	SL08284	SL08316	SL08348	SL08380	SL08412	SL08444
邏輯軸 #32 程式現在位置	SL08222	SL08254	SL08286	SL08318	SL08350	SL08382	SL08414	SL08446

· 系統工作編號 9 ～ 16

系統工作編號		工作 9	工作 10	工作 11	工作 12	工作 13	工作 14	工作 15	工作 16
執行中的主程式編號		SW03208	SW03209	SW03210	SW03211	SW03212	SW03213	SW03214	SW03215
狀態		SW03728	SW03786	SW03844	SW03902	SW03960	SW04018	SW04076	SW04134
控制訊號		SW03729	SW03787	SW03845	SW03903	SW03961	SW04019	SW04077	SW04135
並列 0	程式編號	SW03730	SW03788	SW03846	SW03904	SW03962	SW04020	SW04078	SW04136
	區塊編號	SW03731	SW03789	SW03847	SW03905	SW03963	SW04021	SW04079	SW04137
	警報碼	SL26128 (SW03732)	SL26144 (SW03790)	SL26160 (SW03848)	SL26176 (SW03906)	SL26192 (SW03964)	SL26208 (SW04022)	SL26224 (SW04080)	SL26240 (SW04138)
並列 1	程式編號	SW03733	SW03791	SW03849	SW03907	SW03965	SW04023	SW04081	SW04139
	區塊編號	SW03734	SW03792	SW03850	SW03908	SW03966	SW04024	SW04082	SW04140
	警報碼	SL26130 (SW03735)	SL26146 (SW03793)	SL26162 (SW03851)	SL26178 (SW03909)	SL26194 (SW03967)	SL26210 (SW04025)	SL26226 (SW04083)	SL26242 (SW04141)
並列 2	程式編號	SW03736	SW03794	SW03852	SW03910	SW03968	SW04026	SW04084	SW04142
	區塊編號	SW03737	SW03795	SW03853	SW03911	SW03969	SW04027	SW04085	SW04143
	警報碼	SL26132 (SW03738)	SL26148 (SW03796)	SL26164 (SW03854)	SL26180 (SW03912)	SL26196 (SW03970)	SL26212 (SW04028)	SL26228 (SW04086)	SL26244 (SW04144)
並列 3	程式編號	SW03739	SW03797	SW03855	SW03913	SW03971	SW04029	SW04087	SW04145
	區塊編號	SW03740	SW03798	SW03856	SW03914	SW03972	SW04030	SW04088	SW04146
	警報碼	SL26134 (SW03741)	SL26150 (SW03799)	SL26166 (SW03857)	SL26182 (SW03915)	SL26198 (SW03973)	SL26214 (SW04031)	SL26230 (SW04089)	SL26246 (SW04147)
並列 4	程式編號	SW03742	SW03800	SW03858	SW03916	SW03974	SW04032	SW04090	SW04148
	區塊編號	SW03743	SW03801	SW03859	SW03917	SW03975	SW04033	SW04091	SW04149
	警報碼	SL26136 (SW03744)	SL26152 (SW03802)	SL26168 (SW03860)	SL26184 (SW03918)	SL26200 (SW03976)	SL26216 (SW04034)	SL26232 (SW04092)	SL26248 (SW04150)
並列 5	程式編號	SW03745	SW03803	SW03861	SW03919	SW03977	SW04035	SW04093	SW04151
	區塊編號	SW03746	SW03804	SW03862	SW03920	SW03978	SW04036	SW04094	SW04152
	警報碼	SL26138 (SW03747)	SL26154 (SW03805)	SL26170 (SW03863)	SL26186 (SW03921)	SL26202 (SW03979)	SL26218 (SW04037)	SL26234 (SW04095)	SL26250 (SW04153)
並列 6	程式編號	SW03748	SW03806	SW03864	SW03922	SW03980	SW04038	SW04096	SW04154
	區塊編號	SW03749	SW03807	SW03865	SW03923	SW03981	SW04039	SW04097	SW04155
	警報碼	SL26140 (SW03750)	SL26156 (SW03808)	SL26172 (SW03866)	SL26188 (SW03924)	SL26204 (SW03982)	SL26220 (SW04040)	SL26236 (SW04098)	SL26252 (SW04156)
並列 7	程式編號	SW03751	SW03809	SW03867	SW03925	SW03983	SW04041	SW04099	SW04157
	區塊編號	SW03752	SW03810	SW03868	SW03926	SW03984	SW04042	SW04100	SW04158
	警報碼	SL26142 (SW03753)	SL26158 (SW03811)	SL26174 (SW03869)	SL26190 (SW03927)	SL26206 (SW03985)	SL26222 (SW04043)	SL26238 (SW04101)	SL26254 (SW04159)
邏輯軸 #1 程式現在位置		SL03754	SL03812	SL03870	SL03928	SL03986	SL04044	SL04102	SL04160
邏輯軸 #2 程式現在位置		SL03756	SL03814	SL03872	SL03930	SL03988	SL04046	SL04104	SL04162
邏輯軸 #3 程式現在位置		SL03758	SL03816	SL03874	SL03932	SL03990	SL04048	SL04106	SL04164
邏輯軸 #4 程式現在位置		SL03760	SL03818	SL03876	SL03934	SL03992	SL04050	SL04108	SL04166
邏輯軸 #5 程式現在位置		SL03762	SL03820	SL03878	SL03936	SL03994	SL04052	SL04110	SL04168
邏輯軸 #6 程式現在位置		SL03764	SL03822	SL03880	SL03938	SL03996	SL04054	SL04112	SL04170
邏輯軸 #7 程式現在位置		SL03766	SL03824	SL03882	SL03940	SL03998	SL04056	SL04114	SL04172
邏輯軸 #8 程式現在位置		SL03768	SL03826	SL03884	SL03942	SL04000	SL04058	SL04116	SL04174
邏輯軸 #9 程式現在位置		SL03770	SL03828	SL03886	SL03944	SL04002	SL04060	SL04118	SL04176

(續下頁)

(續上頁)

系統工作編號	工作 9	工作 10	工作 11	工作 12	工作 13	工作 14	工作 15	工作 16
邏輯軸 #10 程式現在位置	SL03772	SL03830	SL03888	SL03946	SL04004	SL04062	SL04120	SL04178
邏輯軸 #11 程式現在位置	SL03774	SL03832	SL03890	SL03948	SL04006	SL04064	SL04122	SL04180
邏輯軸 #12 程式現在位置	SL03776	SL03834	SL03892	SL03950	SL04008	SL04066	SL04124	SL04182
邏輯軸 #13 程式現在位置	SL03778	SL03836	SL03894	SL03952	SL04010	SL04068	SL04126	SL04184
邏輯軸 #14 程式現在位置	SL03780	SL03838	SL03896	SL03954	SL04012	SL04070	SL04128	SL04186
邏輯軸 #15 程式現在位置	SL03782	SL03840	SL03898	SL03956	SL04014	SL04072	SL04130	SL04188
邏輯軸 #16 程式現在位置	SL03784	SL03842	SL03900	SL03958	SL04016	SL04074	SL04132	SL04190
邏輯軸 #17 程式現在位置	SL08448	SL08480	SL08512	SL08544	SL08576	SL08608	SL08640	SL08672
邏輯軸 #18 程式現在位置	SL08450	SL08482	SL08514	SL08546	SL08578	SL08610	SL08642	SL08674
邏輯軸 #19 程式現在位置	SL08452	SL08484	SL08516	SL08548	SL08580	SL08612	SL08644	SL08676
邏輯軸 #20 程式現在位置	SL08454	SL08486	SL08518	SL08550	SL08582	SL08614	SL08646	SL08678
邏輯軸 #21 程式現在位置	SL08456	SL08488	SL08520	SL08552	SL08584	SL08616	SL08648	SL08680
邏輯軸 #22 程式現在位置	SL08458	SL08490	SL08522	SL08554	SL08586	SL08618	SL08650	SL08682
邏輯軸 #23 程式現在位置	SL08460	SL08492	SL08524	SL08556	SL08588	SL08620	SL08652	SL08684
邏輯軸 #24 程式現在位置	SL08462	SL08494	SL08526	SL08558	SL08590	SL08622	SL08654	SL08686
邏輯軸 #25 程式現在位置	SL08464	SL08496	SL08528	SL08560	SL08592	SL08624	SL08656	SL08688
邏輯軸 #26 程式現在位置	SL08466	SL08498	SL08530	SL08562	SL08594	SL08626	SL08658	SL08690
邏輯軸 #27 程式現在位置	SL08468	SL08500	SL08532	SL08564	SL08596	SL08628	SL08660	SL08692
邏輯軸 #28 程式現在位置	SL08470	SL08502	SL08534	SL08566	SL08598	SL08630	SL08662	SL08694
邏輯軸 #29 程式現在位置	SL08472	SL08504	SL08536	SL08568	SL08600	SL08632	SL08664	SL08696
邏輯軸 #30 程式現在位置	SL08474	SL08506	SL08538	SL08570	SL08602	SL08634	SL08666	SL08698
邏輯軸 #31 程式現在位置	SL08476	SL08508	SL08540	SL08572	SL08604	SL08636	SL08668	SL08700
邏輯軸 #32 程式現在位置	SL08478	SL08510	SL08542	SL08574	SL08606	SL08638	SL08670	SL08702

・系統工作編號 17 ～ 24

系統工作編號		工作 17	工作 18	工作 19	工作 20	工作 21	工作 22	工作 23	工作 24
執行中的主程式編號		SW03216	SW03217	SW03218	SW03219	SW03220	SW03221	SW03222	SW03223
狀態		SW04192	SW04250	SW04308	SW04366	SW04424	SW04482	SW04540	SW04598
控制訊號		SW04193	SW04251	SW04309	SW04367	SW04425	SW04483	SW04541	SW04599
並列 0	程式編號	SW04194	SW04252	SW04310	SW04368	SW04426	SW04484	SW04542	SW04600
	區塊編號	SW04195	SW04253	SW04311	SW04369	SW04427	SW04485	SW04543	SW04601
	警報碼	SL26256 (SW04196)	SL26272 (SW04254)	SL26288 (SW04312)	SL26304 (SW04370)	SL26320 (SW04428)	SL26336 (SW04486)	SL26352 (SW04544)	SL26368 (SW04602)
並列 1	程式編號	SW04197	SW04255	SW04313	SW04371	SW04429	SW04487	SW04545	SW04603
	區塊編號	SW04198	SW04256	SW04314	SW04372	SW04430	SW04488	SW04546	SW04604
	警報碼	SL26258 (SW04199)	SL26274 (SW04257)	SL26290 (SW04315)	SL26306 (SW04373)	SL26322 (SW04431)	SL26338 (SW04489)	SL26354 (SW04547)	SL26370 (SW04605)
並列 2	程式編號	SW04200	SW04258	SW04316	SW04374	SW04432	SW04490	SW04548	SW04606
	區塊編號	SW04201	SW04259	SW04317	SW04375	SW04433	SW04491	SW04549	SW04607
	警報碼	SL26260 (SW04202)	SL26276 (SW04260)	SL26292 (SW04318)	SL26308 (SW04376)	SL26324 (SW04434)	SL26340 (SW04492)	SL26356 (SW04550)	SL26372 (SW04608)
並列 3	程式編號	SW04203	SW04261	SW04319	SW04377	SW04435	SW04493	SW04551	SW04609
	區塊編號	SW04204	SW04262	SW04320	SW04378	SW04436	SW04494	SW04552	SW04610
	警報碼	SL26262 (SW04205)	SL26278 (SW04263)	SL26294 (SW04321)	SL26310 (SW04379)	SL26326 (SW04437)	SL26342 (SW04495)	SL26358 (SW04553)	SL26374 (SW04611)
並列 4	程式編號	SW04206	SW04264	SW04322	SW04380	SW04438	SW04496	SW04554	SW04612
	區塊編號	SW04207	SW04265	SW04323	SW04381	SW04439	SW04497	SW04555	SW04613
	警報碼	SL26264 (SW04208)	SL26280 (SW04266)	SL26296 (SW04324)	SL26312 (SW04382)	SL26328 (SW04440)	SL26344 (SW04498)	SL26360 (SW04556)	SL26376 (SW04614)
並列 5	程式編號	SW04209	SW04267	SW04325	SW04383	SW04441	SW04499	SW04557	SW04615
	區塊編號	SW04210	SW04268	SW04326	SW04384	SW04442	SW04500	SW04558	SW04616
	警報碼	SL26266 (SW04211)	SL26282 (SW04269)	SL26298 (SW04327)	SL26314 (SW04385)	SL26330 (SW04443)	SL26346 (SW04501)	SL26362 (SW04559)	SL26378 (SW04617)
並列 6	程式編號	SW04212	SW04270	SW04328	SW04386	SW04444	SW04502	SW04560	SW04618
	區塊編號	SW04213	SW04271	SW04329	SW04387	SW04445	SW04503	SW04561	SW04619
	警報碼	SL26268 (SW04214)	SL26284 (SW04272)	SL26300 (SW04330)	SL26316 (SW04388)	SL26332 (SW04446)	SL26348 (SW04504)	SL26364 (SW04562)	SL26380 (SW04620)
並列 7	程式編號	SW04215	SW04273	SW04331	SW04389	SW04447	SW04505	SW04563	SW04621
	區塊編號	SW04216	SW04274	SW04332	SW04390	SW04448	SW04506	SW04564	SW04622
	警報碼	SL26270 (SW04217)	SL26286 (SW04275)	SL26302 (SW04333)	SL26318 (SW04391)	SL26334 (SW04449)	SL26350 (SW04507)	SL26366 (SW04565)	SL26382 (SW04623)
邏輯軸 #1 程式現在位置		SL04218	SL04276	SL04334	SL04392	SL04450	SL04508	SL04566	SL04624
邏輯軸 #2 程式現在位置		SL04220	SL04278	SL04336	SL04394	SL04452	SL04510	SL04568	SL04626
邏輯軸 #3 程式現在位置		SL04222	SL04280	SL04338	SL04396	SL04454	SL04512	SL04570	SL04628
邏輯軸 #4 程式現在位置		SL04224	SL04282	SL04340	SL04398	SL04456	SL04514	SL04572	SL04630
邏輯軸 #5 程式現在位置		SL04226	SL04284	SL04342	SL04400	SL04458	SL04516	SL04574	SL04632
邏輯軸 #6 程式現在位置		SL04228	SL04286	SL04344	SL04402	SL04460	SL04518	SL04576	SL04634
邏輯軸 #7 程式現在位置		SL04230	SL04288	SL04346	SL04404	SL04462	SL04520	SL04578	SL04636
邏輯軸 #8 程式現在位置		SL04232	SL04290	SL04348	SL04406	SL04464	SL04522	SL04580	SL04638
邏輯軸 #9 程式現在位置		SL04234	SL04292	SL04350	SL04408	SL04466	SL04524	SL04582	SL04640

(續下頁)

(續上頁)

系統工作編號	工作 17	工作 18	工作 19	工作 20	工作 21	工作 22	工作 23	工作 24
邏輯軸 #10 程式現在位置	SL04236	SL04294	SL04352	SL04410	SL04468	SL04526	SL04584	SL04642
邏輯軸 #11 程式現在位置	SL04238	SL04296	SL04354	SL04412	SL04470	SL04528	SL04586	SL04644
邏輯軸 #12 程式現在位置	SL04240	SL04298	SL04356	SL04414	SL04472	SL04530	SL04588	SL04646
邏輯軸 #13 程式現在位置	SL04242	SL04300	SL04358	SL04416	SL04474	SL04532	SL04590	SL04648
邏輯軸 #14 程式現在位置	SL04244	SL04302	SL04360	SL04418	SL04476	SL04534	SL04592	SL04650
邏輯軸 #15 程式現在位置	SL04246	SL04304	SL04362	SL04420	SL04478	SL04536	SL04594	SL04652
邏輯軸 #16 程式現在位置	SL04248	SL04306	SL04364	SL04422	SL04480	SL04538	SL04596	SL04654
邏輯軸 #17 程式現在位置	SL08704	SL08736	SL08768	SL08800	SL08832	SL08864	SL08896	SL08928
邏輯軸 #18 程式現在位置	SL08706	SL08738	SL08770	SL08802	SL08834	SL08866	SL08898	SL08930
邏輯軸 #19 程式現在位置	SL08708	SL08740	SL08772	SL08804	SL08836	SL08868	SL08900	SL08932
邏輯軸 #20 程式現在位置	SL08710	SL08742	SL08774	SL08806	SL08838	SL08870	SL08902	SL08934
邏輯軸 #21 程式現在位置	SL08712	SL08744	SL08776	SL08808	SL08840	SL08872	SL08904	SL08936
邏輯軸 #22 程式現在位置	SL08714	SL08746	SL08778	SL08810	SL08842	SL08874	SL08906	SL08938
邏輯軸 #23 程式現在位置	SL08716	SL08748	SL08780	SL08812	SL08844	SL08876	SL08908	SL08940
邏輯軸 #24 程式現在位置	SL08718	SL08750	SL08782	SL08814	SL08846	SL08878	SL08910	SL08942
邏輯軸 #25 程式現在位置	SL08720	SL08752	SL08784	SL08816	SL08848	SL08880	SL08912	SL08944
邏輯軸 #26 程式現在位置	SL08722	SL08754	SL08786	SL08818	SL08850	SL08882	SL08914	SL08946
邏輯軸 #27 程式現在位置	SL08724	SL08756	SL08788	SL08820	SL08852	SL08884	SL08916	SL08948
邏輯軸 #28 程式現在位置	SL08726	SL08758	SL08790	SL08822	SL08854	SL08886	SL08918	SL08950
邏輯軸 #29 程式現在位置	SL08728	SL08760	SL08792	SL08824	SL08856	SL08888	SL08920	SL08952
邏輯軸 #30 程式現在位置	SL08730	SL08762	SL08794	SL08826	SL08858	SL08890	SL08922	SL08954
邏輯軸 #31 程式現在位置	SL08732	SL08764	SL08796	SL08828	SL08860	SL08892	SL08924	SL08956
邏輯軸 #32 程式現在位置	SL08734	SL08766	SL08798	SL08830	SL08862	SL08894	SL08926	SL08958

・系統工作編號 25 ～ 32

系統工作編號		工作 25	工作 26	工作 27	工作 28	工作 29	工作 30	工作 31	工作 32
執行中的主程式編號		SW03224	SW03225	SW03226	SW03227	SW03228	SW03229	SW03230	SW03231
狀態		SW04656	SW04714	SW04772	SW04830	SW04888	SW04946	SW05004	SW05062
控制訊號		SW04657	SW04715	SW04773	SW04831	SW04889	SW04947	SW05005	SW05063
並列 0	程式編號	SW04658	SW04716	SW04774	SW04832	SW04890	SW04948	SW05006	SW05064
	區塊編號	SW04659	SW04717	SW04775	SW04833	SW04891	SW04949	SW05007	SW05065
	警報碼	SL26384 (SW04660)	SL26400 (SW04718)	SL26416 (SW04776)	SL26432 (SW04834)	SL26448 (SW04892)	SL26464 (SW04950)	SL26480 (SW05008)	SL26496 (SW05066)
並列 1	程式編號	SW04661	SW04719	SW04777	SW04835	SW04893	SW04951	SW05009	SW05067
	區塊編號	SW04662	SW04720	SW04778	SW04836	SW04894	SW04952	SW05010	SW05068
	警報碼	SL26386 (SW04663)	SL26402 (SW04721)	SL26418 (SW04779)	SL26434 (SW04837)	SL26450 (SW04895)	SL26466 (SW04953)	SL26482 (SW05011)	SL26498 (SW05069)
並列 2	程式編號	SW04664	SW04722	SW04780	SW04838	SW04896	SW04954	SW05012	SW05070
	區塊編號	SW04665	SW04723	SW04781	SW04839	SW04897	SW04955	SW05013	SW05071
	警報碼	SL26388 (SW04666)	SL26404 (SW04724)	SL26420 (SW04782)	SL26436 (SW04840)	SL26452 (SW04898)	SL26468 (SW04956)	SL26484 (SW05014)	SL26500 (SW05072)
並列 3	程式編號	SW04667	SW04725	SW04783	SW04841	SW04899	SW04957	SW05015	SW05073
	區塊編號	SW04668	SW04726	SW04784	SW04842	SW04900	SW04958	SW05016	SW05074
	警報碼	SL26390 (SW04669)	SL26406 (SW04727)	SL26422 (SW04785)	SL26438 (SW04843)	SL26454 (SW04901)	SL26470 (SW04959)	SL26486 (SW05017)	SL26502 (SW05075)
並列 4	程式編號	SW04670	SW04728	SW04786	SW04844	SW04902	SW04960	SW05018	SW05076
	區塊編號	SW04671	SW04729	SW04787	SW04845	SW04903	SW04961	SW05019	SW05077
	警報碼	SL26392 (SW04672)	SL26408 (SW04730)	SL26424 (SW04788)	SL26440 (SW04846)	SL26456 (SW04904)	SL26472 (SW04962)	SL26488 (SW05020)	SL26504 (SW05078)
並列 5	程式編號	SW04673	SW04731	SW04789	SW04847	SW04905	SW04963	SW05021	SW05079
	區塊編號	SW04674	SW04732	SW04790	SW04848	SW04906	SW04964	SW05022	SW05080
	警報碼	SL26394 (SW04675)	SL26410 (SW04733)	SL26426 (SW04791)	SL26442 (SW04849)	SL26458 (SW04907)	SL26474 (SW04965)	SL26490 (SW05023)	SL26506 (SW05081)
並列 6	程式編號	SW04676	SW04734	SW04792	SW04850	SW04908	SW04966	SW05024	SW05082
	區塊編號	SW04677	SW04735	SW04793	SW04851	SW04909	SW04967	SW05025	SW05083
	警報碼	SL26396 (SW04678)	SL26412 (SW04736)	SL26428 (SW04794)	SL26444 (SW04852)	SL26460 (SW04910)	SL26476 (SW04968)	SL26492 (SW05026)	SL26508 (SW05084)
並列 7	程式編號	SW04679	SW04737	SW04795	SW04853	SW04911	SW04969	SW05027	SW05085
	區塊編號	SW04680	SW04738	SW04796	SW04854	SW04912	SW04970	SW05028	SW05086
	警報碼	SL26398 (SW04681)	SL26414 (SW04739)	SL26430 (SW04797)	SL26446 (SW04855)	SL26462 (SW04913)	SL26478 (SW04971)	SL26494 (SW05029)	SL26510 (SW05087)
邏輯軸 #1 程式現在位置		SL04682	SL04740	SL04798	SL04856	SL04914	SL04972	SL05030	SL05088
邏輯軸 #2 程式現在位置		SL04684	SL04742	SL04800	SL04858	SL04916	SL04974	SL05032	SL05090
邏輯軸 #3 程式現在位置		SL04686	SL04744	SL04802	SL04860	SL04918	SL04976	SL05034	SL05092
邏輯軸 #4 程式現在位置		SL04688	SL04746	SL04804	SL04862	SL04920	SL04978	SL05036	SL05094
邏輯軸 #5 程式現在位置		SL04690	SL04748	SL04806	SL04864	SL04922	SL04980	SL05038	SL05096
邏輯軸 #6 程式現在位置		SL04692	SL04750	SL04808	SL04866	SL04924	SL04982	SL05040	SL05098
邏輯軸 #7 程式現在位置		SL04694	SL04752	SL04810	SL04868	SL04926	SL04984	SL05042	SL05100
邏輯軸 #8 程式現在位置		SL04696	SL04754	SL04812	SL04870	SL04928	SL04986	SL05044	SL05102
邏輯軸 #9 程式現在位置		SL04698	SL04756	SL04814	SL04872	SL04930	SL04988	SL05046	SL05104

(續下頁)

(續上頁)

系統工作編號	工作 25	工作 26	工作 27	工作 28	工作 29	工作 30	工作 31	工作 32
邏輯軸 #10 程式現在位置	SL04700	SL04758	SL04816	SL04874	SL04932	SL04990	SL05048	SL05106
邏輯軸 #11 程式現在位置	SL04702	SL04760	SL04818	SL04876	SL04934	SL04992	SL05050	SL05108
邏輯軸 #12 程式現在位置	SL04704	SL04762	SL04820	SL04878	SL04936	SL04994	SL05052	SL05110
邏輯軸 #13 程式現在位置	SL04706	SL04764	SL04822	SL04880	SL04938	SL04996	SL05054	SL05112
邏輯軸 #14 程式現在位置	SL04708	SL04766	SL04824	SL04882	SL04940	SL04998	SL05056	SL05114
邏輯軸 #15 程式現在位置	SL04710	SL04768	SL04826	SL04884	SL04942	SL05000	SL05058	SL05116
邏輯軸 #16 程式現在位置	SL04712	SL04770	SL04828	SL04886	SL04944	SL05002	SL05060	SL05118
邏輯軸 #17 程式現在位置	SL08960	SL08992	SL09024	SL09056	SL09088	SL09120	SL09152	SL09184
邏輯軸 #18 程式現在位置	SL08962	SL08994	SL09026	SL09058	SL09090	SL09122	SL09154	SL09186
邏輯軸 #19 程式現在位置	SL08964	SL08996	SL09028	SL09060	SL09092	SL09124	SL09156	SL09188
邏輯軸 #20 程式現在位置	SL08966	SL08998	SL09030	SL09062	SL09094	SL09126	SL09158	SL09190
邏輯軸 #21 程式現在位置	SL08968	SL09000	SL09032	SL09064	SL09096	SL09128	SL09160	SL09192
邏輯軸 #22 程式現在位置	SL08970	SL09002	SL09034	SL09066	SL09098	SL09130	SL09162	SL09194
邏輯軸 #23 程式現在位置	SL08972	SL09004	SL09036	SL09068	SL09100	SL09132	SL09164	SL09196
邏輯軸 #24 程式現在位置	SL08974	SL09006	SL09038	SL09070	SL09102	SL09134	SL09166	SL09198
邏輯軸 #25 程式現在位置	SL08976	SL09008	SL09040	SL09072	SL09104	SL09136	SL09168	SL09200
邏輯軸 #26 程式現在位置	SL08978	SL09010	SL09042	SL09074	SL09106	SL09138	SL09170	SL09202
邏輯軸 #27 程式現在位置	SL08980	SL09012	SL09044	SL09076	SL09108	SL09140	SL09172	SL09204
邏輯軸 #28 程式現在位置	SL08982	SL09014	SL09046	SL09078	SL09110	SL09142	SL09174	SL09206
邏輯軸 #29 程式現在位置	SL08984	SL09016	SL09048	SL09080	SL09112	SL09144	SL09176	SL09208
邏輯軸 #30 程式現在位置	SL08986	SL09018	SL09050	SL09082	SL09114	SL09146	SL09178	SL09210
邏輯軸 #31 程式現在位置	SL08988	SL09020	SL09052	SL09084	SL09116	SL09148	SL09180	SL09212
邏輯軸 #32 程式現在位置	SL08990	SL09022	SL09054	SL09086	SL09118	SL09150	SL09182	SL09214

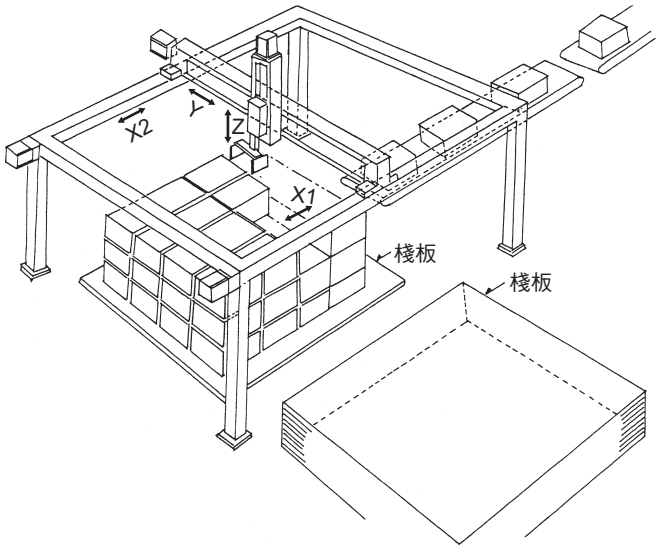
1.9 應用實例

讓運動程式可適用於各式各樣的裝置。
以下為應用實例。

搬運裝置

本範例係依照指定的數量來堆疊瓦楞紙箱，然後再送到下一個製程。

可用來控制棧板裝載的 3 軸動作，以及棧板自動供料的序列。

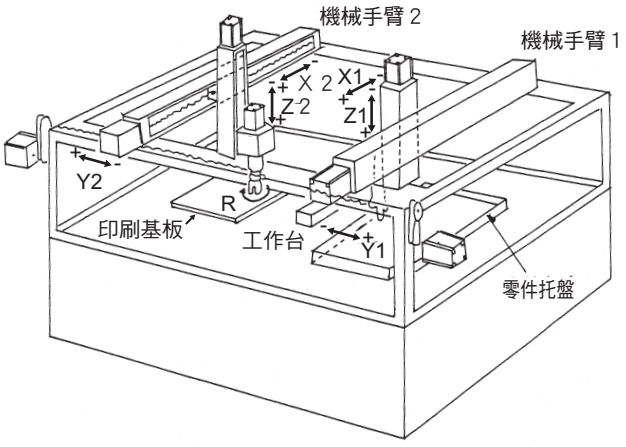


控制重點
- 使用虛擬軸，讓 X1 軸、X2 軸同步運轉。
- 利用插補動作，讓動作更順暢。
- 運動程式將根據預設條件 (包裝箱尺寸、水平列列數、垂直列列數、堆疊層數)，計算位置資料，以執行棧板堆疊。

零件插入機

本範例所介紹的裝置可用來將連接器等的零件插入印刷基板。

可利用搬運用機械手臂將零件從托盤取出，並搬運到工作台上，再依照插入用機械手臂所指定的位置及角度，插入至基板。

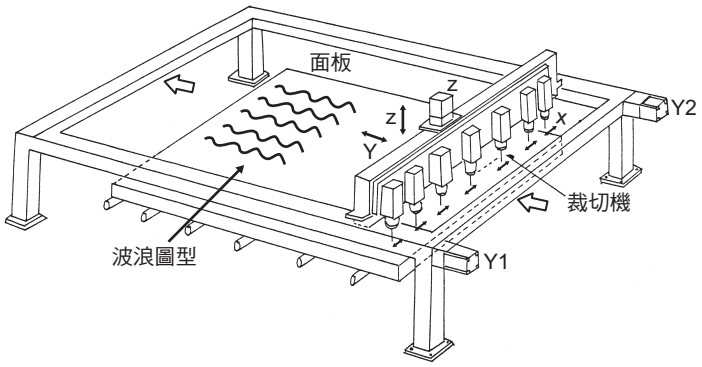


控制重點
- 利用群組運轉功能，讓兩組機械手臂控制程式可單獨執行。
- 利用 2 軸或 3 軸線性內插動作，以縮短作業時間。

面板加工機

本範例為將圖型放入建材用平板的裝置。

X 軸的垂直方向配置了 10 軸以上的裁切機，即可任意改變圖型的寬度。



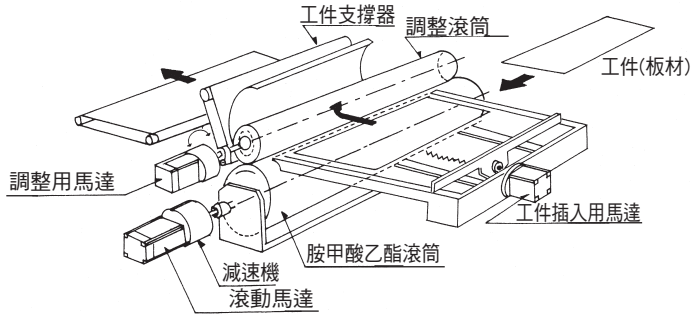
控制重點

- 對 X 軸、Y 軸進行循環內插，以描繪出波浪造型。
- 使用虛擬軸，讓 Y1 軸、Y2 軸同步運轉。

板材成型裝置

本範例所介紹的是板材彎曲成型裝置。

利用滾輪軸一面運送板材，同時移動調整軸，即可將板材成型為任意形狀。



控制重點

- 對直線軸和旋轉軸的 2 個轉軸進行線性內插控制。
- 可依照加工類型不同，切換您所要叫出的運動程式。

序列程式概述

2

本章將以序列程式初次使用者為對象，說明程式概要、特性及其使用方法。

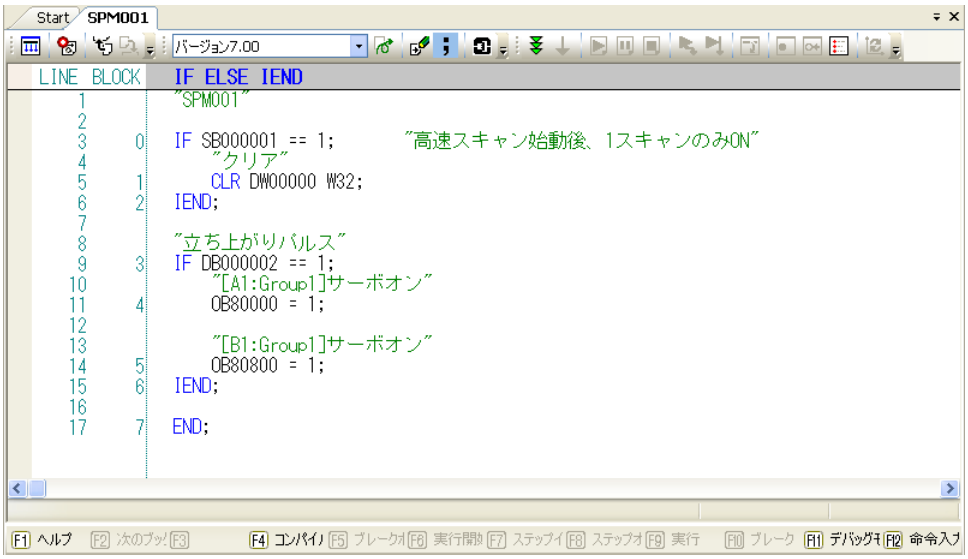
2.1 何謂「序列程式」

序列程式就是一種利用和運動程式共用的語言來編寫的掃描執行型程式。

使用序列程式，即可架構一套可在固定週期檢測互鎖等的狀態的應用程式。

利用 M-EXECUTOR 程式執行定義，即可叫出並執行序列程式。

最多共可編寫 512 個序列程式和運動程式。
以下為序列程式範例。

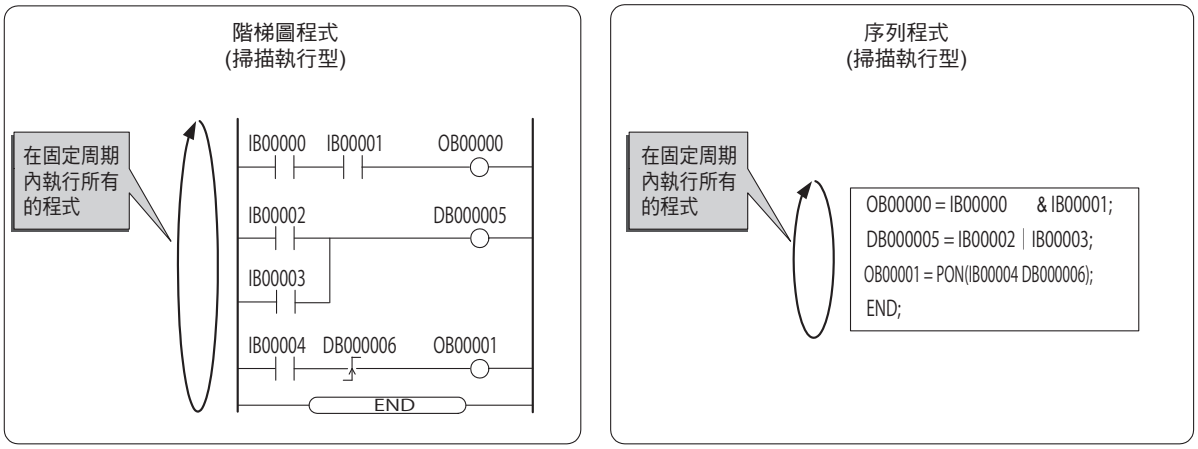


2.2 序列程式特色

序列程式執行方式

序列程式所採用的執行方式和階梯圖程式相同。

若序列程式在固定週期執行動作，並且在掃描週期結束前，完成程式起始到 END 指令所有的處理作業，即稱之為「掃描執行型」程式。只要叫出 M-EXECUTOR 程式執行定義，即可執行序列程式。



和運動程式採用相同的語言

序列程式所採用的運動語言和運動程式相同。

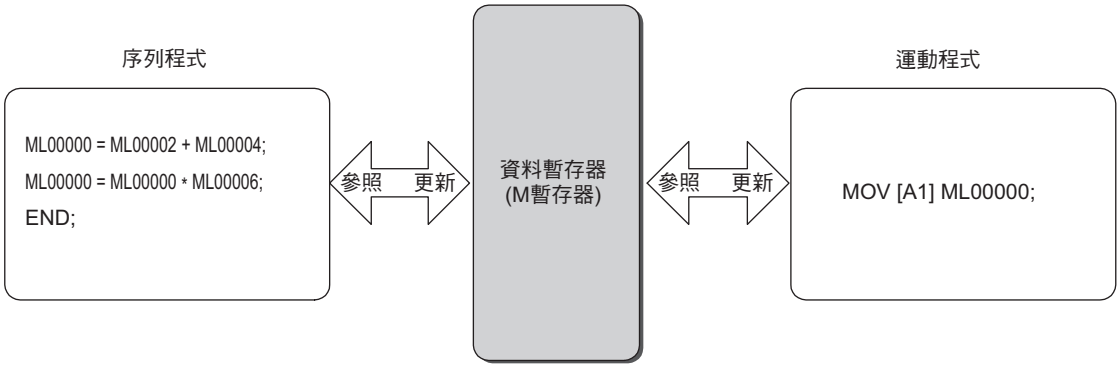
於序列程式，運動語言指令中可使用的指令僅限為運算指令等的序列語言指令。軸移動指令等運動指言指令則不適用。

使用序列程式後，不需要階梯圖程式，也能編寫序列控置專用的應用程式。

可和運動程式互相進行資料收送

序列程式和運動程式之間可互相進行資料傳收。
資料傳收必須透過資料暫存器 (M 暫存器)。

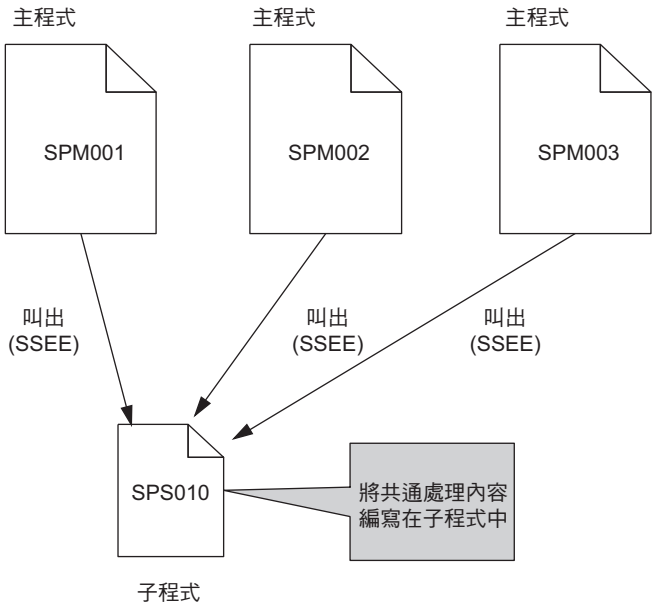
運動程式可使用序列程式裡更新後的數值，反之，序列程式也可使用運動程式已更新的數值。



善用子程式以提高記憶體效率

序列程式可用來編寫子程式 (子程式)。

只要讓共通的動作以子程式化 (共通化) 來執行，就能將程式的步進控制在最小化，並提高記憶體效率。



適用簡易程式功能

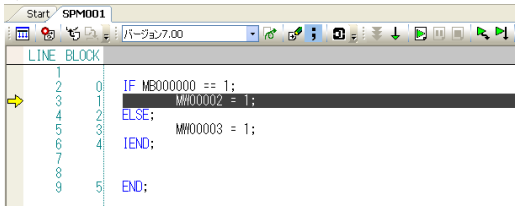
序列程式亦可使用以下的簡易程式功能。

● 指令輸入輔助功能

在[指令輸入輔助]對話框中選擇指令，並將資料設定完成後，即可將指令插入編輯器中。

● 除錯運轉功能

本功能可用來為序列程式除錯。
利用步進執行、設定斷點等除錯指令，即可為序列程式的動作進行除錯。



2.3 序列程式的類型

序列程式包含下表所示的 2 種類型。

分類	指定方法	特徵	程式數
主程式	SPM□□□ （□□□= 1～512）	・利用 M-EXECUTOR 程式執行定義叫出	以下所示之程式最多共編寫 512 個 ・運動主程式 ・運動子程式 ・序列主程式 ・序列子程式
子程式	SPS□□□ （□□□= 1～512）	・由主程式叫出	


重要

序列程式的程式編號和運動程式編號可採用統一管理方式。
程式必須使用各自不同的編號。
・ 運動程式 MPM□□□、MPS□□□ 的程式編號
・ 序列程式 SPM□□□、SPS□□□ 的程式編號

2.4 執行序列程式

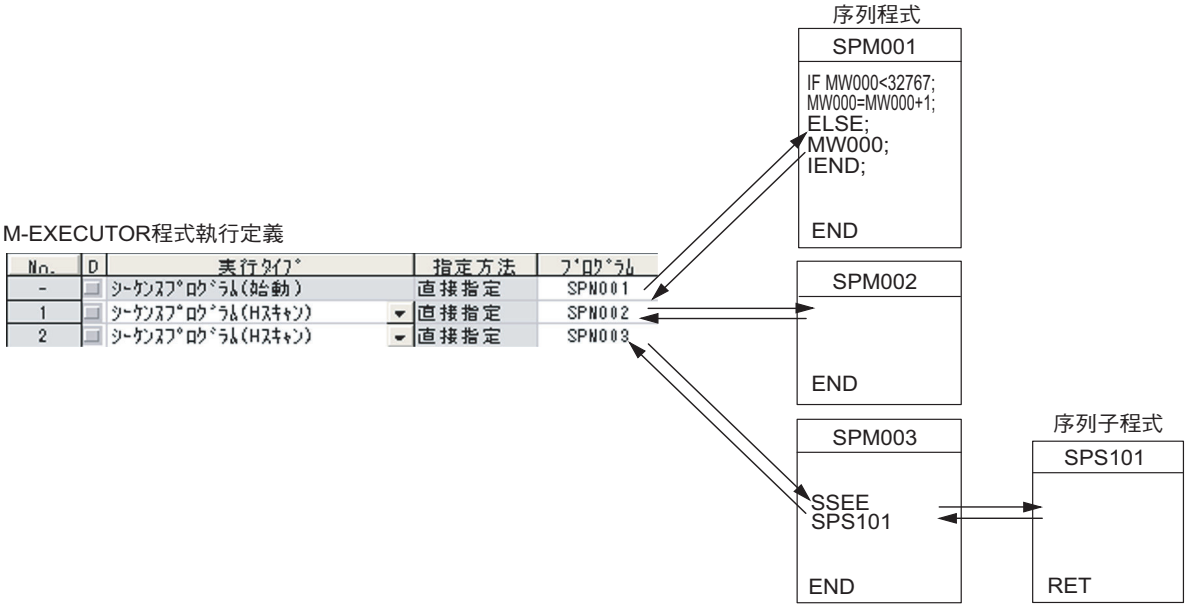
接下來將說明序列程式的執行方法。

執行方式

叫出 M-EXECUTOR 程式執行定義，即可執行序列程式。

系統將依照編號順序，由小到大依序執行序列程式。

下圖為執行範例。



若執行類型已設定為序列程式 (H 掃描) 或序列程式 (L 掃描)，就會在儲存定義時同時執行程式。如果執行類型被設定為「序列程式 (啟動)」，就會在下一次電源重新啟動時執行程式。

M-EXECUTOR 程式執行定義

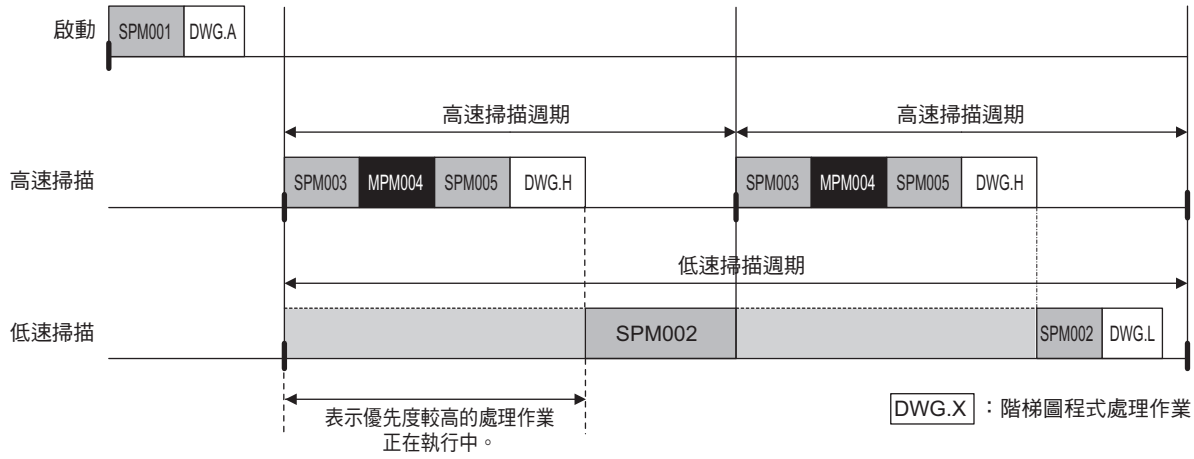
範例

序列程式執行範例

以下視窗為序列程式被登錄至 M-EXECUTOR 程式執行定義之範例。

執行時間

接下來將說明在上述視窗所述的設定條件下的執行時間。
系統將依照下圖所示的 M-EXECUTOR 程式執行定義的登錄順序來執行動作。



程式登錄

以下所示為執行登錄的方法。下圖係以 SPM001 被登錄到 H 掃描處理作業為例。

補充　序列程式僅適用於直接指定方法。不適用於間接指定。

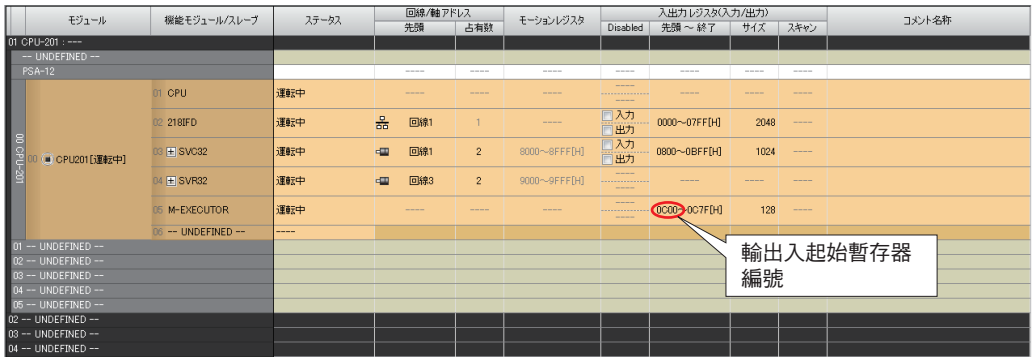
工作暫存器

執行程式登錄後，系統就會開始配置狀態旗標，以監控序列程式的狀態。利用以下公式即可求出序列程式的狀態旗標。

IW□□□□□ + 4 ×(程式登錄No. 1)

└ M-EXECUTOR輸出入起始暫存器編號 *

*輸出入起始暫存器編號可利用模組組成定義視窗來確認。



狀態旗標

序列程式的狀態旗標可用來瞭解序列程式的執行狀態。

下表為狀態旗標之詳細說明。

Bit No		名稱	內容
0 ～ 3	Bit 0	程式運轉中	當序列程式進入運轉狀態，此位元將變為 1。 0：序列程式停止中 1：序列程式運轉狀態
	Bit 1	(系統預約)	–
	Bit 2	(系統預約)	–
	Bit 3	(系統預約)	–
4 ～ 7	Bit 4	(系統預約)	–
	Bit 5	(系統預約)	–
	Bit 6	(系統預約)	–
	Bit 7	(系統預約)	–
8 ～ B	Bit 8	程式目前發生錯誤	參照序列子程式 (執行 SSEE 指令) 時，一旦發生以下異常，此位元將變為 1。當異常解除後，就會變為 0。 ・尚未登錄您所要叫出的目標程式 ・叫出序列程式以外的程式 ・所要叫出的目標程式為子程式以外程式 (已叫出主程式) ・所要叫出的目標程式發生編號錯誤 ・Nest 錯誤 0：未發生程式警報 1：程式目前發生警報
	Bit 9	因斷點而進入停止狀態	利用除錯運轉，讓斷點進入停止狀態時，此位元將變為 1。 0：斷點進入停止以外狀態 1：斷點停止狀態
	Bit A	(系統預約)	–
	Bit B	除錯模式中	利用除錯運轉來執行程式運轉時，此位元將變為 1。 0：進入除錯模式運轉以外狀態 (一般運轉狀態) 1：除錯模式運轉狀態

(續下頁)

(接上頁)

Bit No		名稱	內容
C ～ F	Bit C	程式類型	系統將會回報目前正在執行的程式為運動程式或序列程式。 0：運動程式 1：序列程式
	Bit D	要求啟動之歷程紀錄	當序列程式進入運轉狀態，此位元將變為 1。 0：序列程式停止中 1：序列程式運轉狀態
	Bit E	(系統預約)	–
	Bit F	(系統預約)	–

註記

序列程式警報

參照序列子程式 (執行 SSEE 指令) 時，一旦系統檢測到異常，狀態旗標 Bit 8 (程式警報發生狀態) 就會變為 1。當異常解除後，就會變為 0。

異常內容包含以下幾種。

- 尚未登錄您所要叫出的目標程式
- 所要叫出的標的程式為序列程式以外程式
- 所要叫出的目標程式為子程式以外程式 (已叫出主程式)
- 所要叫出的目標程式發生編號超過範圍錯誤
- Nest 錯誤

程式開發流程

3

本章將說明如何利用開發工具 MPE720 Ver. 7 來執行啟動系統，到實機運轉等所有的步驟。

3.1 程式開發流程

以下流程為本章所介紹的運動程式開發步驟。

① 備妥連線裝置

請將 MPE720 安裝至電腦中。
備妥連線裝置並加以配線。

☞ 系統架構 (第 3-3 頁)
☞ 安裝 MPE720 Ver. 7(第 3-3 頁)

② 製作專案

製作新的專案以作為開發程式之前置準備工作。

☞ 製作專案 (第 3-4 頁)

③ 自動配置

執行自動配置，以啟動系統。

☞ 自動配置 (第 3-6 頁)

④ 在線連線

對運動控制器和電腦進行通訊設定。

☞ 在線連線 (第 3-6 頁)

⑤ 群組定義設定

根據機械架構，將轉軸群組化。

☞ 群組定義設定 (第 3-6 頁)

⑥ 編寫程式

利用運動編輯器來編寫程式。

☞ 編寫程式 (第 3-8 頁)

⑦ 程式登錄

請先將程式登錄至系統中，如此就能利用高速掃描處理來執行編寫完成的程式。

☞ 程式登錄 (第 3-10 頁)

⑧ 傳送程式

將編寫完成的程式傳送到 MP3000 系列。

☞ 傳送程式 (第 3-13 頁)

⑨ 程式除錯

對已製作完成的程式進行除錯。

☞ 程式除錯 (第 3-16 頁)

⑩ 程式儲存至快閃記憶體

將除錯完成的程式儲存於快閃記憶體中。

☞ 程式儲存至快閃記憶體 (第 3-17 頁)

⑪ 執行程式

使用暫存器清單，並執行已編寫完成的程式。

☞ 執行程式 (第 3-18 頁)

(註)1. 基本上來說，運動程式和序列程式的開發步驟是一樣的。
接下來的章節將說明運動程式的開發流程。
2. 以上所示的流程僅為程式開發的其中一個例子。將程式安裝到實際的裝置後，必須同時對外部裝置進行設定。

3.2 程式的開發步驟

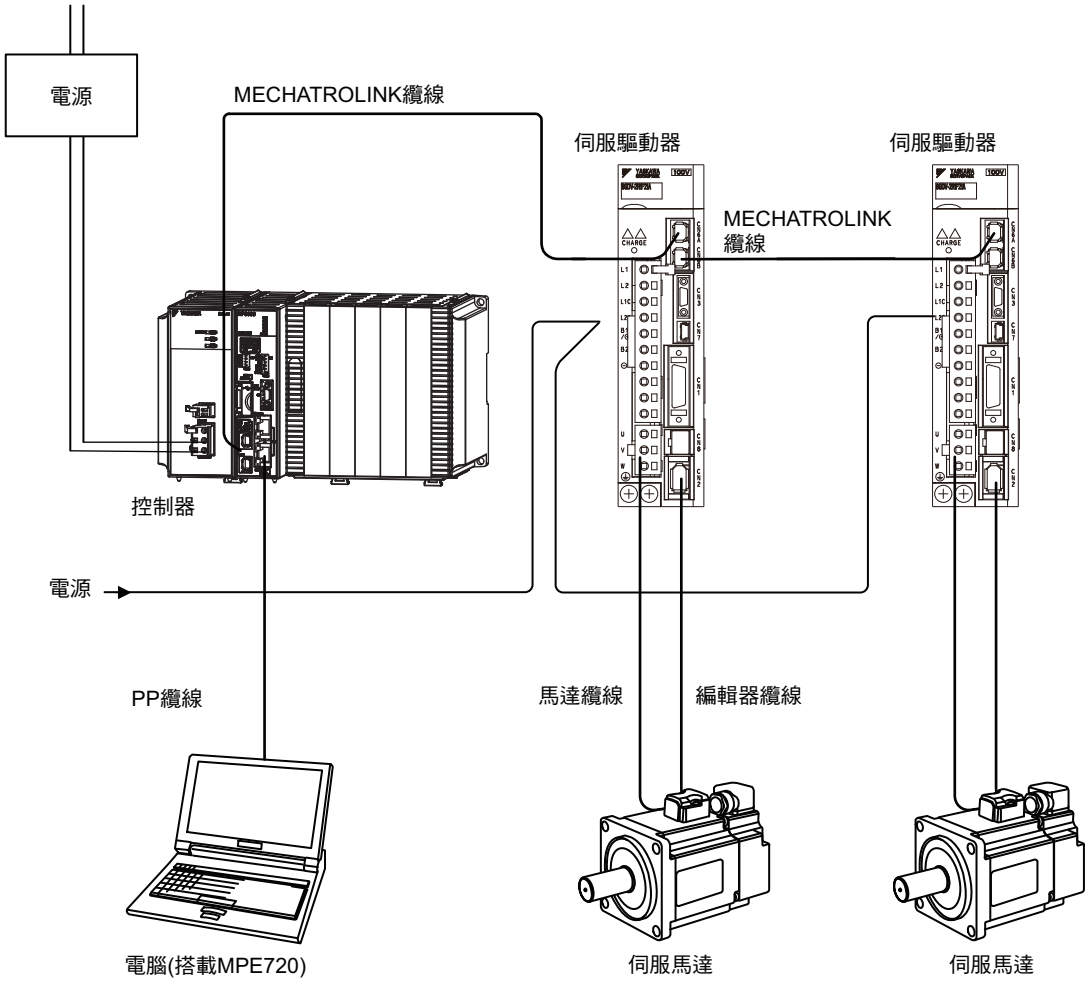
接下來將以系統為範例，說明程式的開發步驟。

備妥連線裝置

首先，先針對和運動控制器互相連接的裝置系統架構，以及系統啟動前的準備工作進行說明。

系統架構

下圖所示為系統架構範例。



(註)在本範例中，伺服驅動器的站號分別被設定為 1 和 2。

安裝 MPE720 Ver. 7

將 MPE720 Ver. 7 安裝至電腦中。

安裝步驟請參閱以下手冊。

MP2000/3000 系列　運動控制器系統　安裝手冊 (資料編號：SIJP C880725 00)

製作專案

專案檔為 MPE720 Ver. 7 專用的應用程式檔案，包含下表所示的資訊。

系統架構	・系統定義 ・掃描時間定義 ・模組架構定義 ・資料追蹤
程式	・階梯圖程式 (高速、低速、啟動、配置 / 函數) ・運動程式 (主程式、子程式、群組定義) ・表格資料 ・變數 (軸、輸出入、總體、常數、使用者自定結構體) ・註解 (輸出入、總體、常數)
暫存器	・M (資料暫存器) ・D (內部暫存器) ・C (常數暫存器) ・S (系統暫存器) ・I (輸入暫存器) ・O (輸出暫存器) ・G (資料暫存器)

專案檔結合了上表所示的各種檔案，在 Windows 環境下會被當作單一檔案來處理，擴充檔名為「.YMW7」。

開啟專案檔後，即可開始編輯上述各種檔案。

1 個 MPE720 Ver. 7 視窗只能同時開啟 1 個專案檔 ; 多個 MPE720 Ver. 7 視窗無法開啟同一個專案檔。當您所要開啟的專案檔已經被打開了，這時候已經開啟該專案檔的 MPE720 Ver. 7 視窗將會被顯示在最上層。

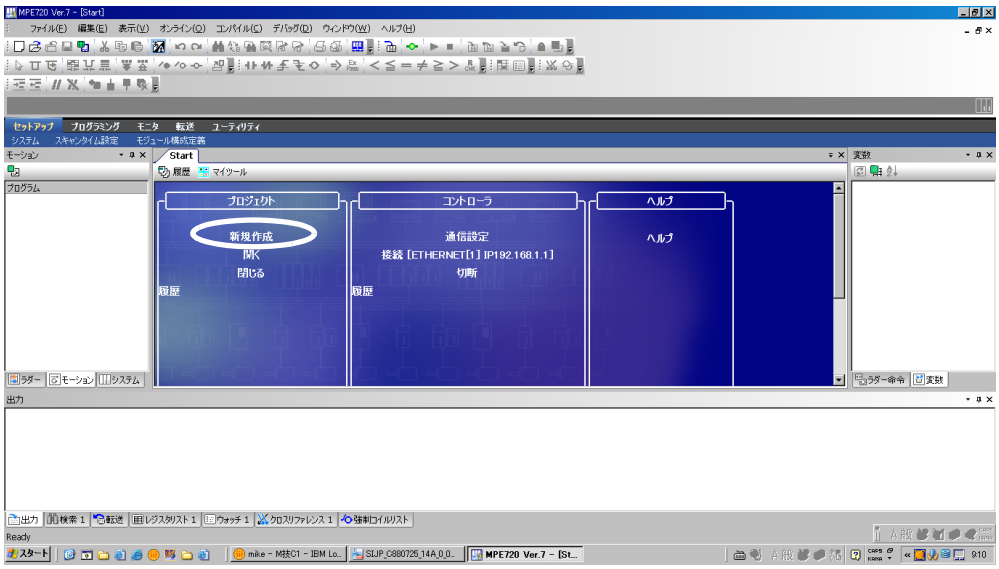

重要

MPE720 Ver 6.0 所編寫的專案檔 (擴充檔名為 .YMW) 亦適用於本程式，不過，該檔案將無法使用 MP3000 系列的擴充功能。

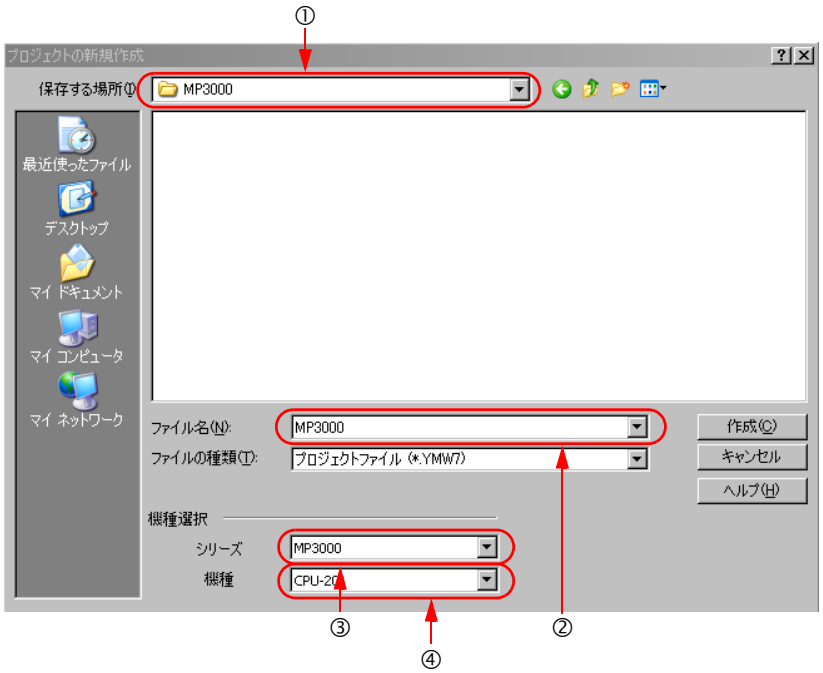
接下來將說明新專案的製作步驟。

1. 在電腦桌面上雙擊以下的圖示，以啟動 MPE720 Ver. 7。

2. 點擊 [製作新專案]。



3. 指定檔案名稱、檔案儲存位置、運動控制器的系列及機型。



①在 [儲存位置] 窗格中指定檔案的儲存位置。
②在 [檔案名稱 (N)：] 窗格中輸入檔案名稱。
③ 從 [系列] 窗格中選擇相對應的系列。
④從 [機型] 窗格中選擇相對應的機型。

4. 按下 [製作] 鍵。

自動配置

使用自動配置功能來啟動系統。所謂「自動配置」就是系統可自動辨識 MP3000 系列運動控制器所安裝的模組以及 MECHATROLINK 連接器所連接的從屬台資訊 (伺服驅動資訊) ，並自動建立模組架構定義檔的一種功能。如此一來，系統就能夠輕鬆地在短時間內執行啟動作業。「自動配置」功能可在運動控制器開啟電源時執行，或是利用 MPE720 來執行。

自動配置步驟請參閱以下手冊。

MP3000 系列　MP3200 使用手冊 (資料編號：SIJP C880725 10)

MP3000 系列　MP3300 產品手冊 (資料編號：YTWMNCO-14008A)

在線連線

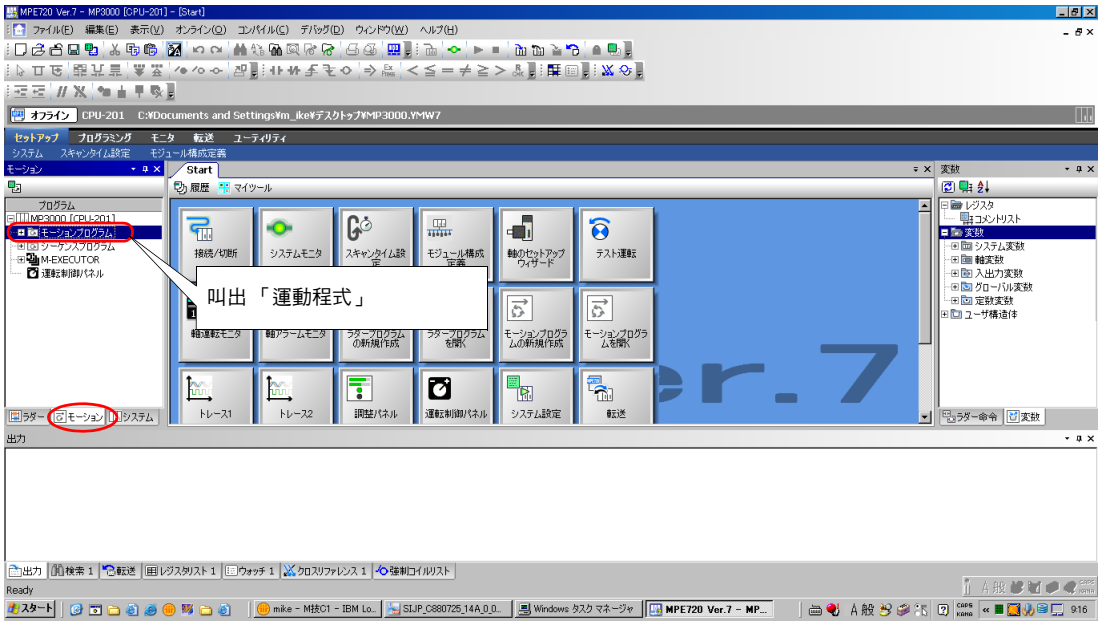
設定通訊條件的目的是為了讓安裝 MPE720 Ver. 7 的電腦和運動控制器之間能夠互相進行通訊。
通訊設定步驟請參閱以下手冊。

MP2000/3000 系列　運動控制器系統　安裝手冊 (資料編號：SIJP C880725 00)

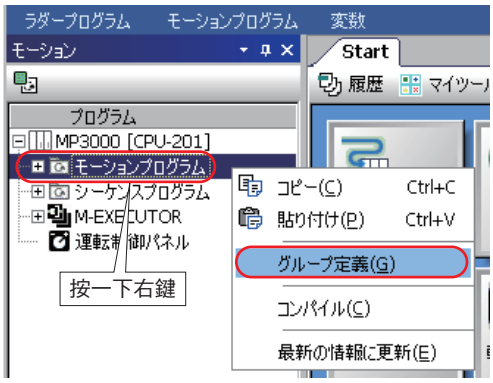
群組定義設定

編寫運動程式前，必須依照機器架構，將轉軸群組化。

1. **點擊子視窗中的 [運動] 索引標籤。**
   接著子視窗中就會顯示 [運動程式]。



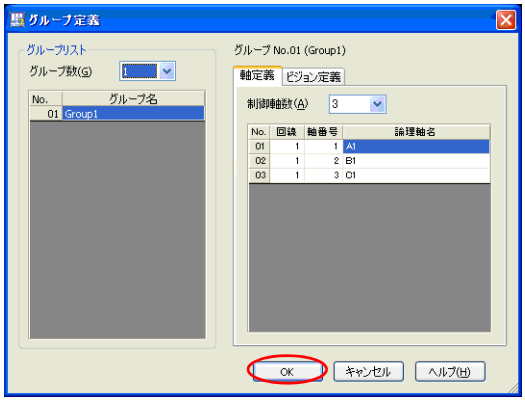
2. 在子視窗的 [運動程式] 上按一下右鍵，當選單出現後，請選擇群組定義。



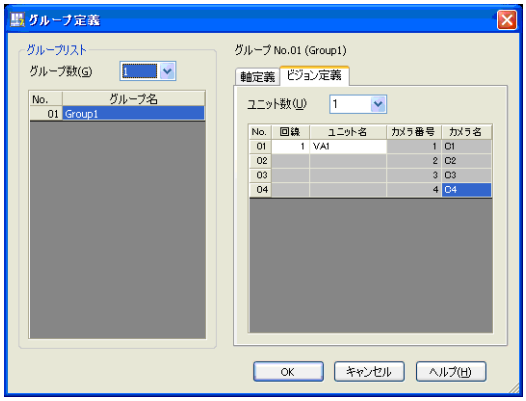
3. 選擇 [軸定義] 索引標籤並設定您所要使用的轉軸內容，然後再按下 [OK] 鍵。

(註)群組定義的相關說明，請參閱以下章節。

☞ **5.2 群組定義說明 (第 5-9 頁)**



補充　從 [群組定義] 對話框中，還可以看到 [視覺定義] 的分頁。



編寫程式

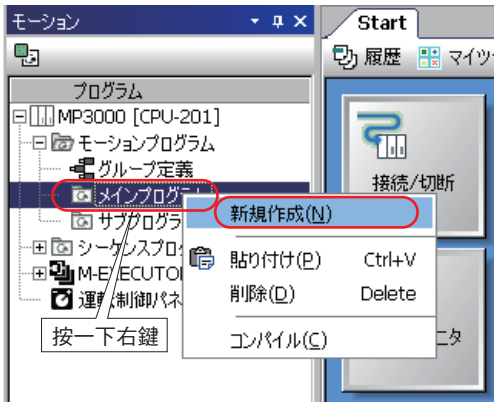
接下來將說明如何啟動運動編輯器，並利用運動程式來編寫以下條件之範例。

條件：讓伺服馬達移動 150000 脈衝量後即停止動作

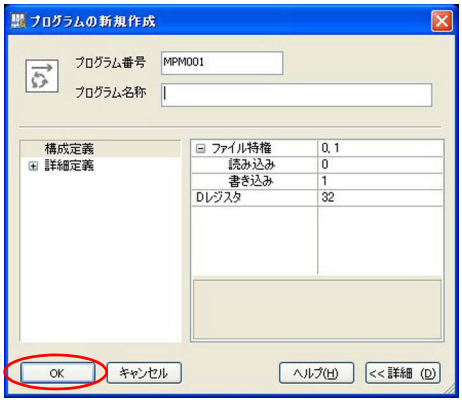

註記

實際啟動馬達時，必須為速度、加速時間及移動量設定適當的數值。

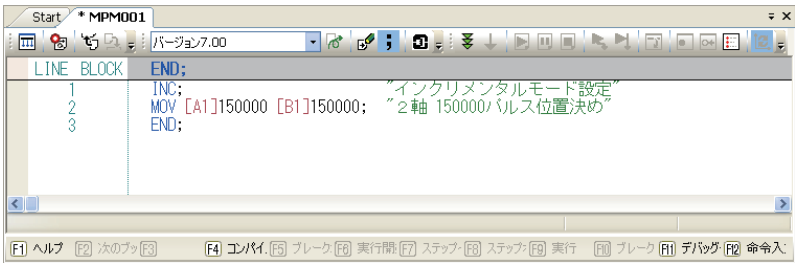
1. 在子視窗的 [主選單] 上按一下右鍵，接著在從選單上選擇 [開新程式]。



2. 請點擊 [OK] 鍵。



3. 輸入運動程式。



4. 在選單列上依序選擇 [編譯] － [編譯] 後，系統就會開始進行編譯。

編譯完成後，運動程式將會自動被儲存。

重要

進行編譯時，若畫面上出現 [錯誤清單] 對話框，表示運動程式將不會被儲存起來。

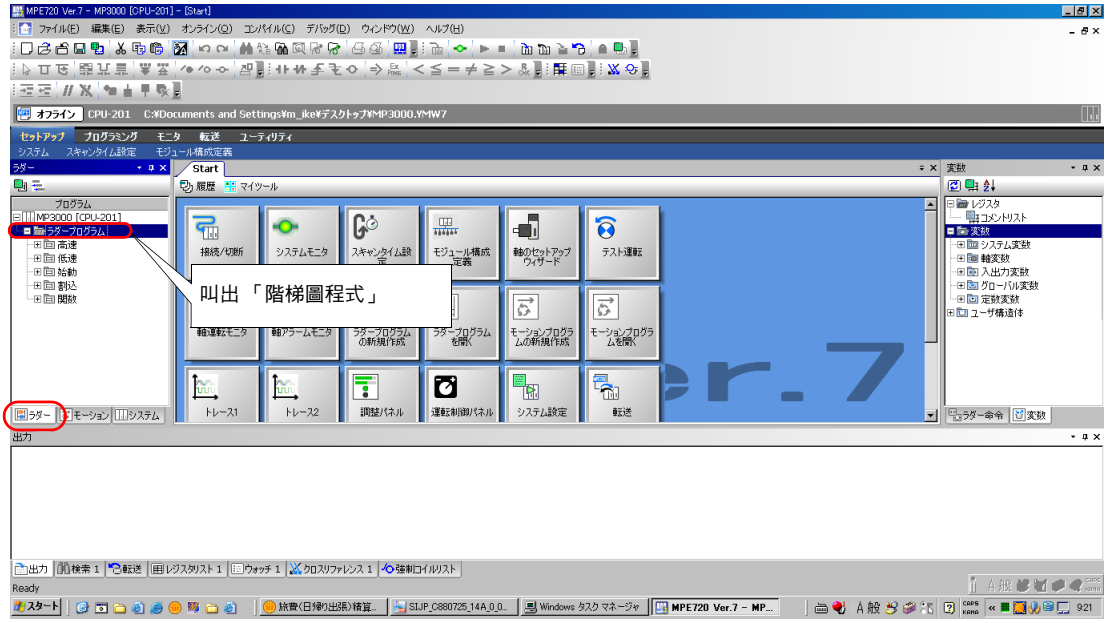
程式登錄

若要叫出編寫完成的運動程式，有以下 2 種方法，第一種是利用階梯圖程式的 MSEE 指令來叫出，另一種則是利用 M-EXECUTOR 程式執行定義來叫出。如欲瞭解程式執行登錄的方法，請參閱以下章節。

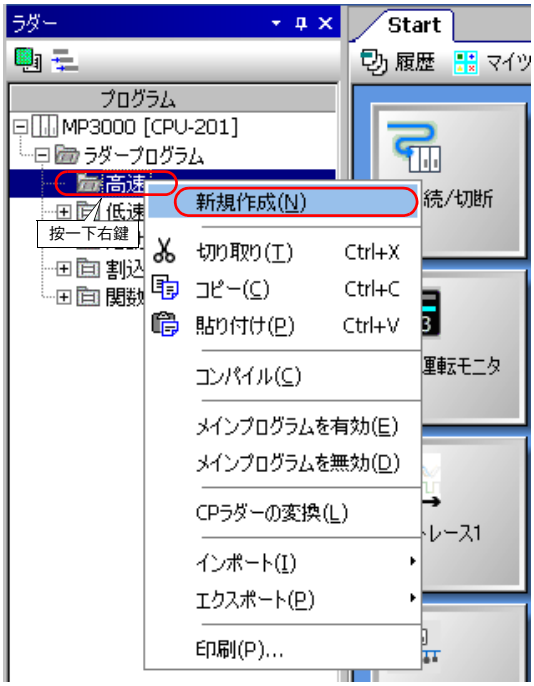
☞ 程式執行登錄的方法 (第 1-21 頁)

利用階梯圖程式來叫出運動程式

1. **點擊子視窗中的 [階梯圖] 索引標籤。**
   接著子視窗中就會顯示 [階梯圖程式]。

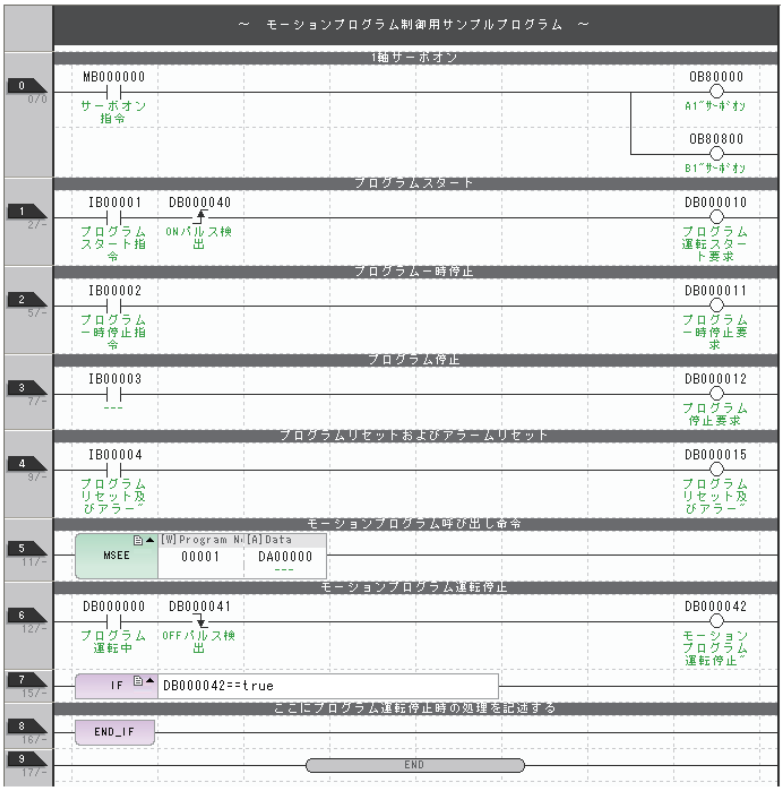


2. **在子視窗的 [高速] 上按一下右鍵，接著在從選單上選擇 [開新程式]。**



3. **請點擊 [OK] 鍵。**

4. **製作下圖所示的階梯圖程式。編寫完成後，就會開始進行編譯 (F4 鍵或 [ ] 圖示)。**



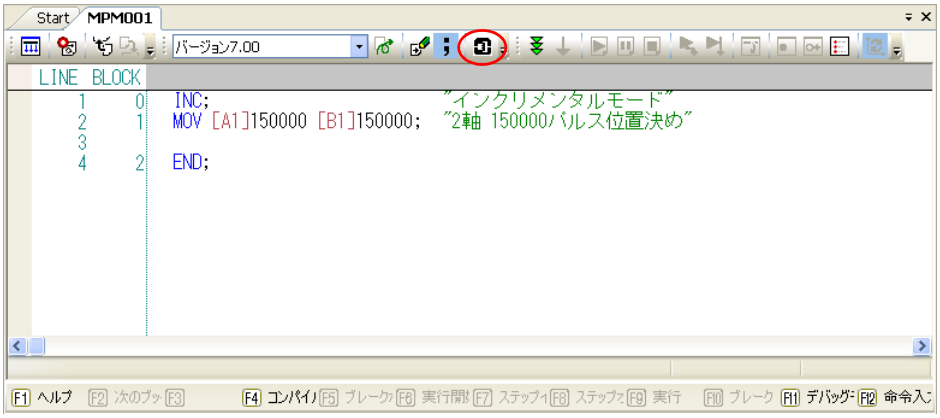
補充

・ 請先確認馬達參數 IW□□□00 Bit 0「運動控制器運轉準備完成」已經開啟後，再將伺服器開啟指令「MB000000」設定為開啟。
・ 當運動控制器運轉準備完成被設定為關閉時，表示系統無法再接受任何伺服器開啟指令。

利用 M-EXECUTOR 來叫出運動程式的方法

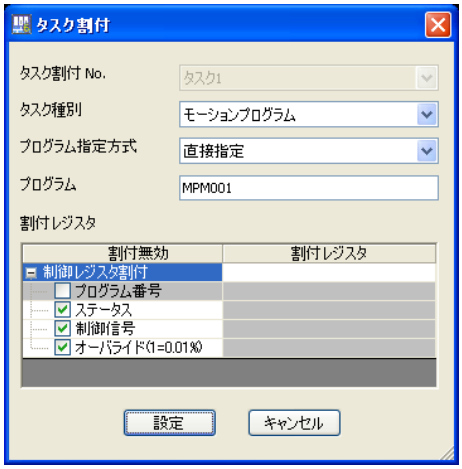
接下來將介紹 M-EXECUTOR 程式執行定義之登錄步驟。不過，執行以下操作動作前，請先傳送程式。

1. 進入已完成程式編寫的運動編輯器視窗，並點擊任務配置圖示 [ ]。



畫面上將出現 [任務配置] 對話框。

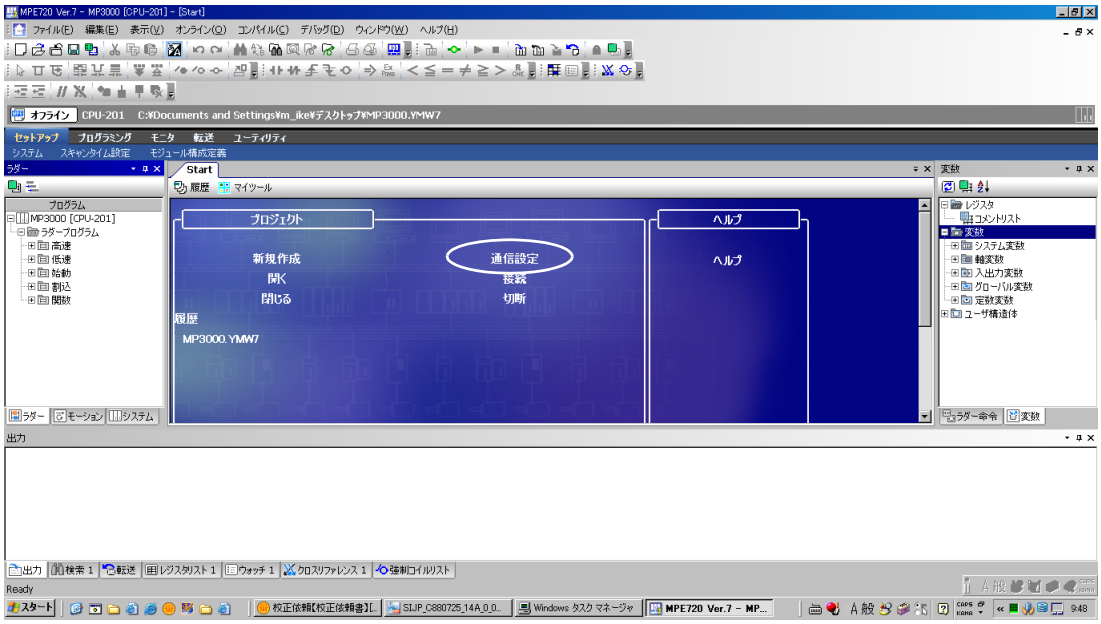
2. 點擊 [設定] 鍵後，系統就會開始進行程式登錄。



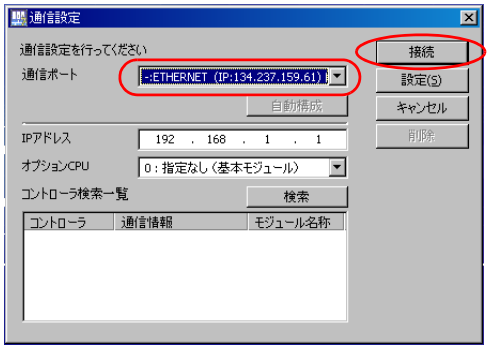
傳送程式

將運動程式傳送到 MP3000 系列。若在連線狀態下編寫運動程式時，則不需要此步驟。

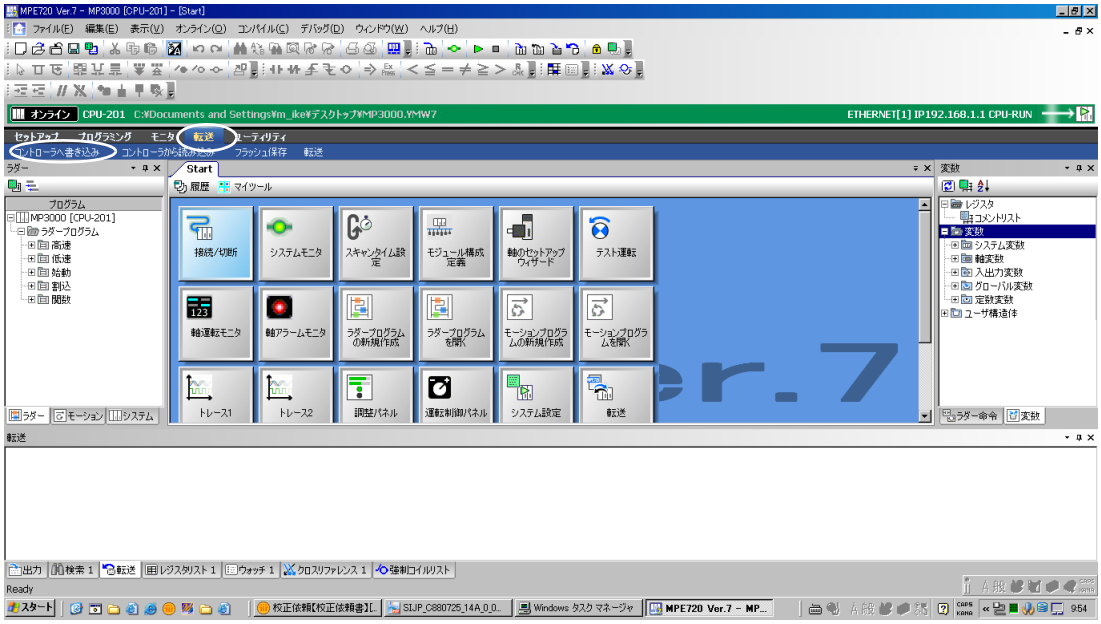
1. 點擊主視窗中的 [通訊設定]。



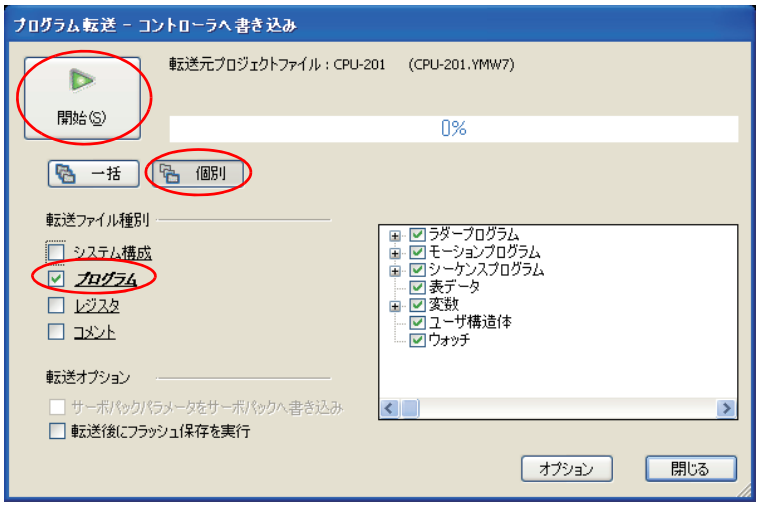
2. 從 [通訊設定] 對話框的 [通訊埠] 方塊中，選擇您所設定的通訊埠，然後再按一下 [連接] 鍵。



3. **從快捷功能列上選擇 [傳送] － [將資料寫入控制器]。**

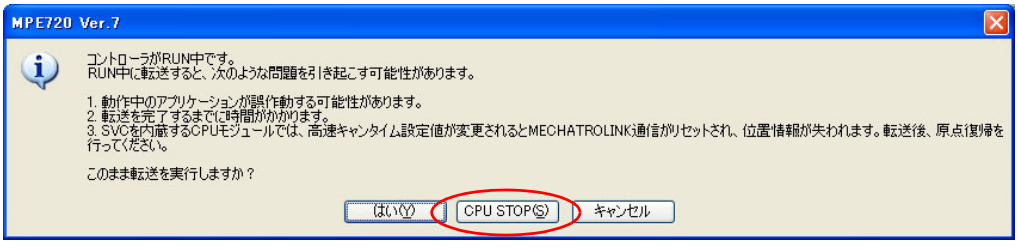


4. **點擊 [個別] 鍵，並僅選取 [程式] 核取方塊。接著，按下 [開始鍵]。**



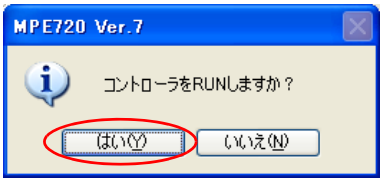
(註)1. 選擇個別傳送時，控制器裡的同一個檔案會被您所選擇的專案檔案資料所覆寫上去。
2. 若選擇全部傳送，運動控制器的 RAM 資料會在傳送前被清除，而所有的專案檔案資料則會同時被寫入。

5. **請點擊 [CPU STOP] 鍵。**



即開始傳送資料。

6. 進入 [MPE720 Ver. 7] 對話框，並點擊 [是] 鍵。

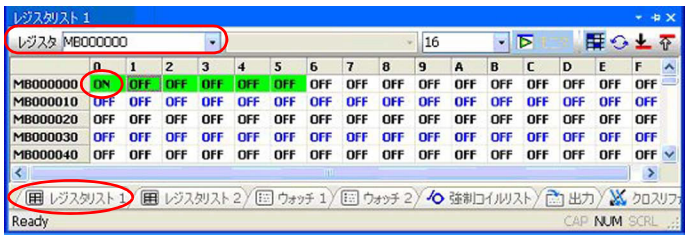


控制器將會開始運轉。

程式除錯

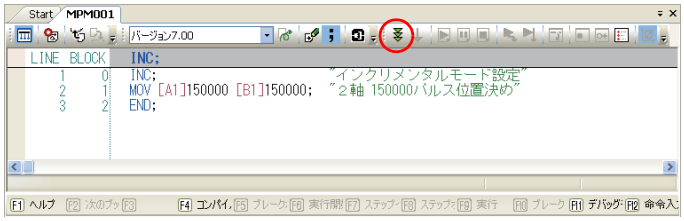
對已製作完成的程式進行除錯。

1. **點擊 [暫存器清單 1] 索引標籤。**
   畫面上將出現暫存器清單。
   請指定 MB000000 作為暫存器。請依下圖所示，在 MB000000 編寫 「ON」，以啟動伺服器。

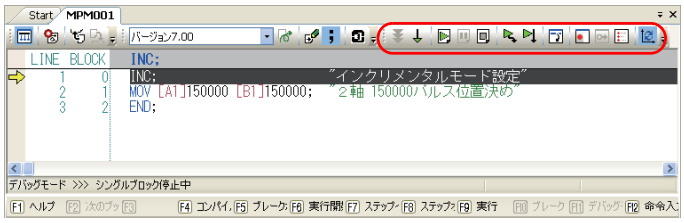


(註)利用 M-EXECUTOR 來執行程式登錄時，只要使用設定參數，即可啟動伺服器。

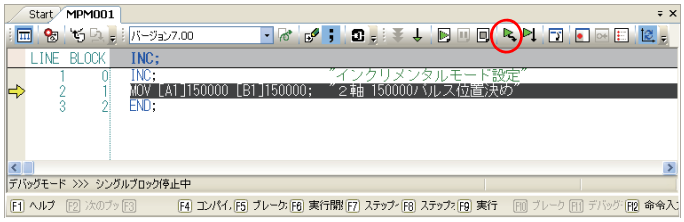
2. **點擊 [ ] 圖示。**



3. **即可切換至除錯模式。**



4. **點擊 [ ] 圖示，即可逐行執行程式，並確認程式的動作。**

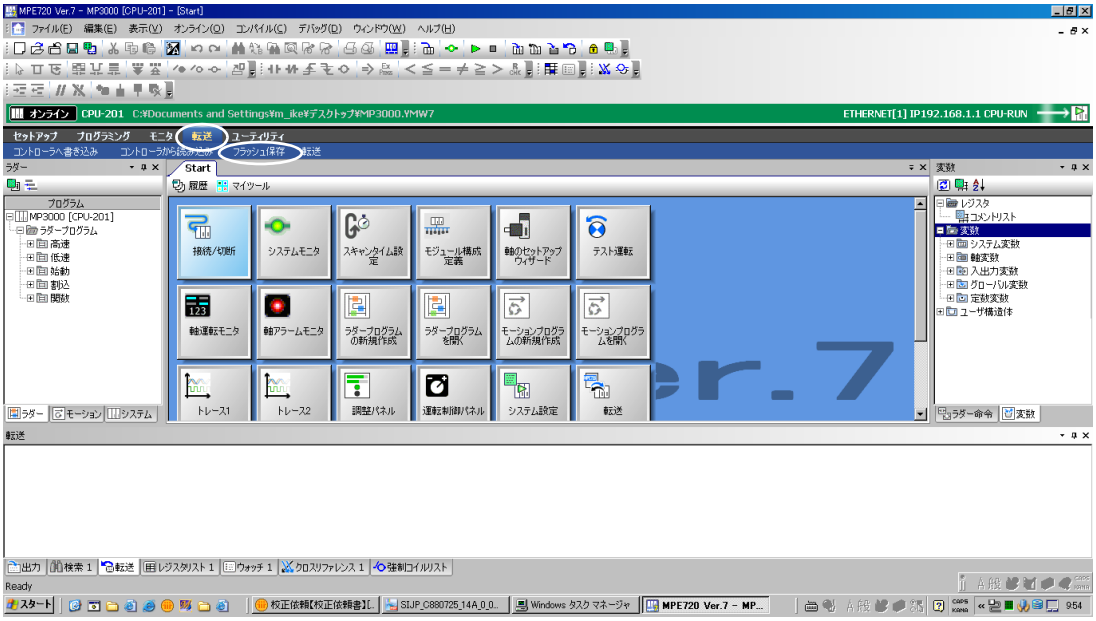


5. **執行 END 指令後，即完成除錯運轉，接著伺服器就會被關閉。**

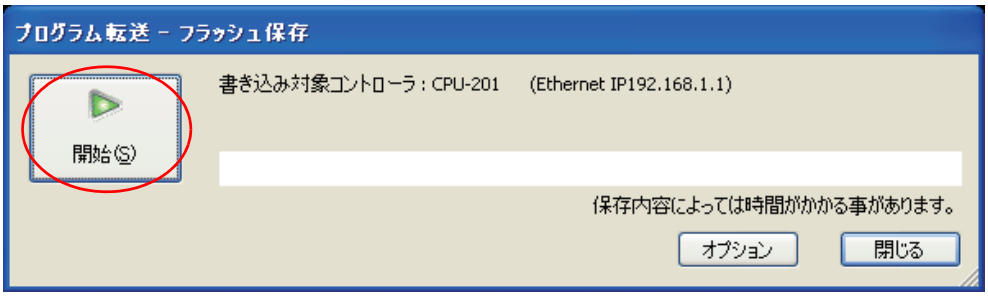
程式儲存至快閃記憶體

將運動控制器 RAM 裡的資料儲存到運動控制器的快閃記憶體中。

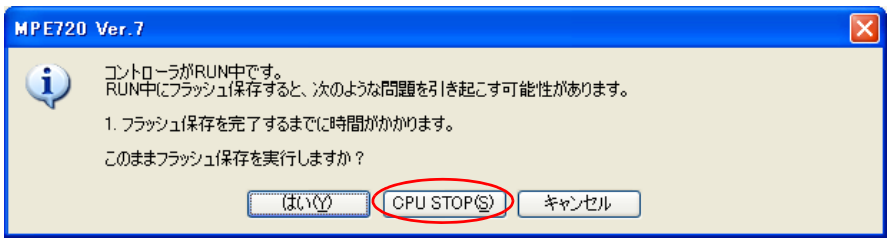
1. **選擇快捷功能列上的 [傳送] － [儲存到快閃記憶體]。**



2. **請點擊 [開始] 鍵。**

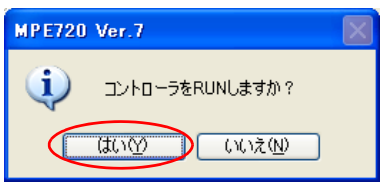


3. **請點擊 [CPU STOP] 鍵。**



程式就會被儲存在快閃記憶體中。

4. **點擊 [是] 鍵。**

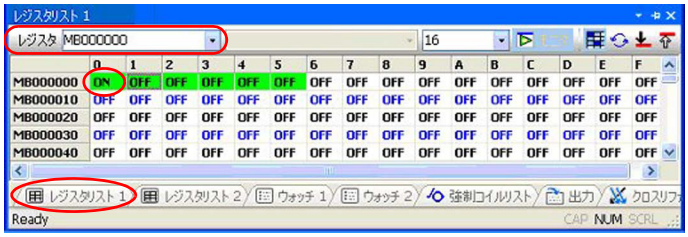


運動控制器將會開始運轉。

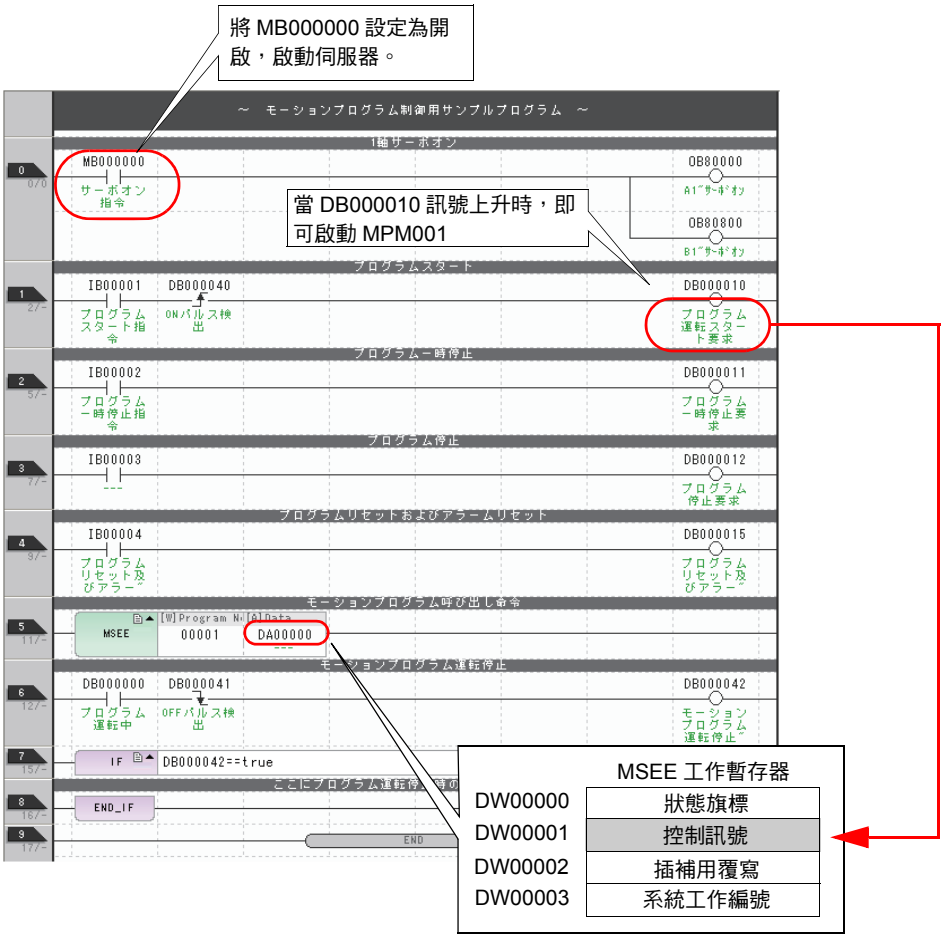
執行程式

利用實際裝置來執行編寫完成的程式。執行運動程式時，系統會操作控制訊號，並將要求程式開始運轉設定為開啟。

1. **點擊 [暫存器清單 1] 索引標籤。**
   畫面上將出現暫存器清單。
   請指定 MB000000 作為暫存器。請依下圖所示，在 MB000000 編寫 「ON」，以啟動伺服器。



2. **在暫存器清單的 MB000001 編寫 「ON」, 就會執行運動程式 MPM001。**



暫存器

4

本章將針對適用於運動程式及序列程式的暫存器做進一步的說明。

4.1 暫存器

本節將針對暫存器進行說明。

使用運動程式和序列程式時，除了可直接將「數值」寫入外，還可利用「暫存器」來編寫。實際使用「暫存器」時，系統會擷取儲存於「暫存器區」的「數值」來使用。

暫存器類型

暫存器共有 11 種，每種程式適用的暫存器各不相同。

運動程式和序列程式適用下表所示的 7 種 (S、M、G、I、O、C、D) 暫存器。

S、M、G、I、O、C 暫存器為所有程式皆適用的總體暫存器。D 暫存器屬於局部暫存器，僅適用於單一圖面，無法在其他程式當中使用。

暫存器類型

類型	名稱	指定方法	適用範圍	內容	特性
S	系統暫存器 (S 暫存器)	SBnnnnnh， SWnnnnn， SLnnnnn， SQnnnnn， SFnnnnn， SDnnnnn， SAnnnnn	SW00000 ~ SW65534	系統所預設之暫存器，可用來回報運動控制器的狀態等。 系統啟動時，SW00000 ~ SW00049 將會被歸零。 將以電池組備用。	適用於所有程式
M	資料暫存器 (M 暫存器)	MBnnnnnnnh， MWnnnnnnn， MLnnnnnnn， MQnnnnnnn， MFnnnnnnn， MDnnnnnnn， MAnnnnnnn	MW0000000 ~ MW1048575	使用於程式間的 I/F 等的暫存器。 將以電池組備用。	
G	G 暫存器	GBnnnnnnnh, GWnnnnnnn, GLnnnnnnn, GQnnnnnnn, GFnnnnnnn, GDnnnnnnn, GAnnnnnnn,	GW0000000 ~ GW2097151	使用於程式間的 I/F 等的暫存器。 並未以電池組備用。	
I	輸入暫存器 (I 暫存器)	IBhhhhhh， IWhhhhh， ILhhhhh， IQhhhhh， IFhhhhh， IDhhhhh， IAhhhhh，	IW00000 ~ IW07FFF, IW10000 ~ IW17FFF	適用於輸入資料的暫存器。	
			IW08000 ~ IW0FFFF	為運動監控參數。 此類暫存器適用於運動控制功能。	
			IW20000 ~ IW23FFF	適用於 CPUIF 輸入資料的暫存器。	

(續下頁)

暫存器類型（續上頁）

類型	名稱	指定方法	適用範圍	內容	特性
O	輸出暫存器 (O 暫存器)	OBhhhhhh， OWhhhhh， OLhhhhh， OQhhhhh， OFhhhhh， ODhhhhh， OAhhhhh，	OW00000 ~ OW07FFF, OW10000 ~ OW17FFF	適用於輸出資料的暫存器。	適用於所有程式
			OW08000 ~ OW0FFFF	為運動設定參數。 此類暫存器適用於運動控制功能。	
			OW20000 ~ OW23FFF	適用於 CPUIF 輸出資料的暫存器。	
C	常數暫存器 (C 暫存器)	CBnnnnnh， CWnnnnn， CLnnnnn， CQnnnnn, CFnnnnn， CDnnnnn, CAnnnnn	CW00000 ~ CW16383	僅能在程式內部參照的暫存器。 利用 MPE720 即可設定數值。	
D	D 暫存器	DBnnnnnh， DWnnnnn, DLnnnnn， DQnnnnn， DFnnnnn， DDnnnnn， DAnnnnn	DW00000 ~ DW16383	程式內部所通用的暫存器。 初始值為每個圖面可使用 32 個字元。	單一程式
#	# 暫存器	#Bnnnnnh， #Wnnnnn， #Lnnnnn， #Qnnnnn， #Fnnnnn， #Dnnnnn， #Annnnn	#W00000 ~ #W16383	此類暫存器僅適合用來參照。 只有相對應的 DWG 才能參照。	單一函數
X	函數輸入暫存器	XBnnnnnh， XWnnnnn， XLnnnnn， XQnnnnn， XFnnnnn， XDnnnnn	XW00000 ~ XW00016	輸入至函數 ・輸入位元：XB000000 ~ XB00000F ・輸入整數：XW00001 ~ XW00016 ・長整數：XL00001 ~ XL00015 ・4 長整數：XQ00001 ~ XQ00013 ・實數：XF00001 ~ XF00015 ・倍精度實數：XD00001 ~ XD00013	
Y	函數輸出暫存器	YBnnnnnh， YWnnnnn, YLnnnnn， YQnnnnn， YFnnnnn， YDnnnnn	YW00000 ~ YW000016	輸出至函數 ・輸出位元：YB000000 ~ YB00000F ・輸出整數：YW00001 ~ YW00016 ・長整數：YL00001 ~ YL00015 ・4 長整數：YQ00001 ~ YQ00013 ・實數：YF00001 ~ YF00015 ・倍精度實數：YD00001 ~ YD00013	
Z	函數內部暫存器	ZBnnnnnh， ZWnnnnn, ZLnnnnn， ZQnnnnn， ZFnnnnn， ZDnnnnn	ZW00000 ~ ZW00016	每個函數所預設的內部暫存器。 適合作為函數內部處理之用。	

(註)n：10 進位制、h：16 進位制

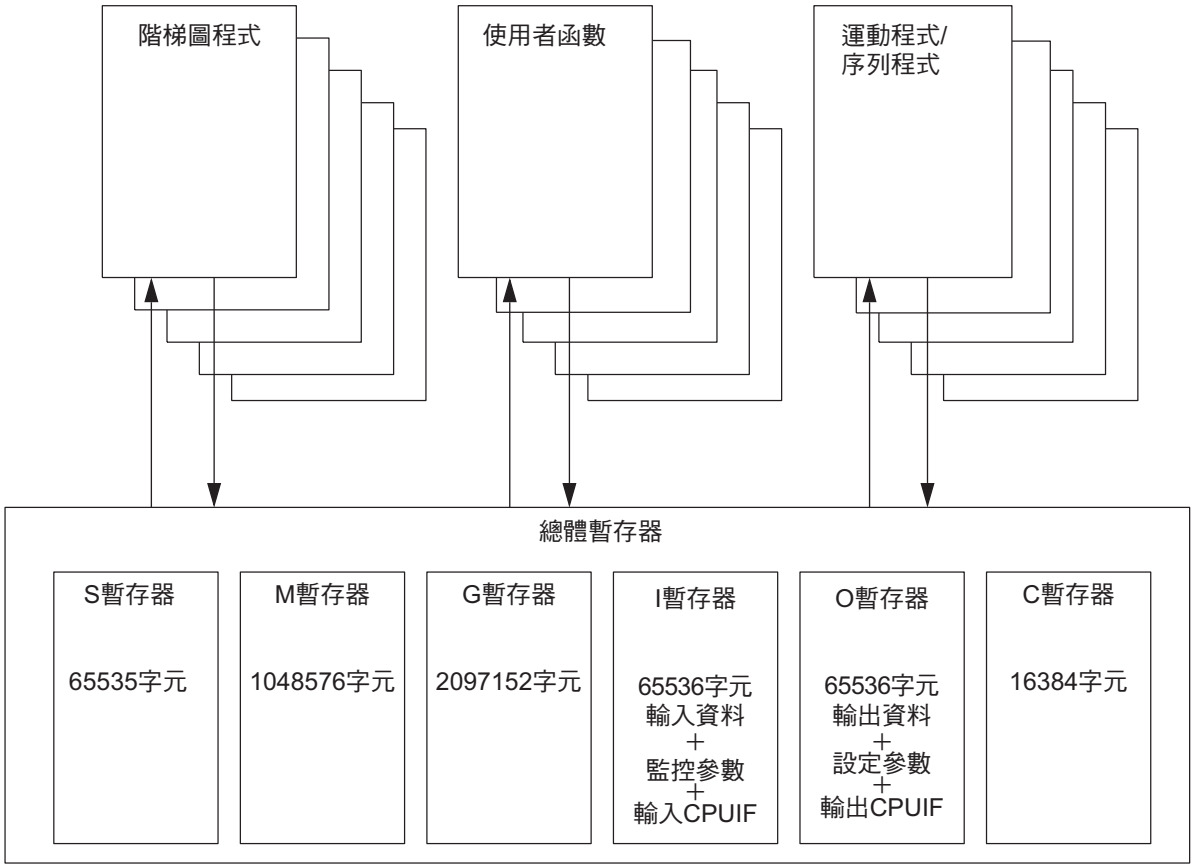

重要

暫存器不適用於運動程式和序列程式。若使用 # 暫存器，儲存程式時，系統將顯示 「語法錯誤」。

總體暫存器

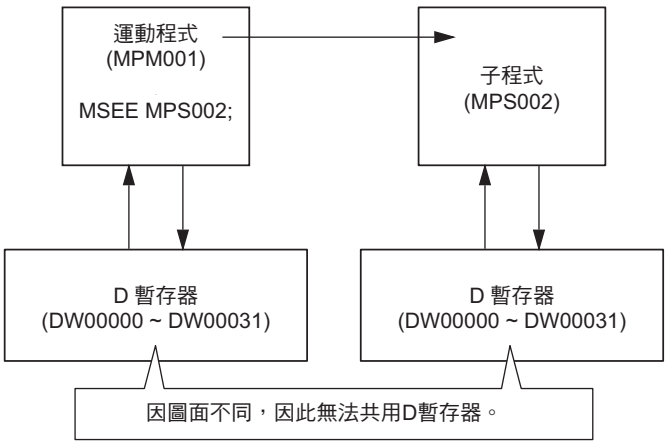
總體暫存器是一種適用於階梯圖程式、使用者函數、運動程式及序列程式各種程式的暫存器。

階梯圖程式運算出來的結果同樣適用於其他使用者函數、運動程式及序列程式。系統已經為每種暫存器預留了總體暫存器所需的空間。



局部暫存器

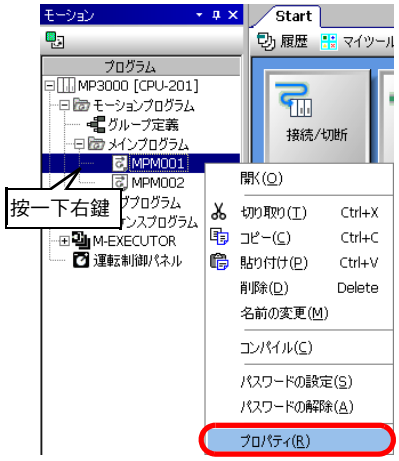
局部暫存器是一種適用於單一圖面的暫存器，無法與其他圖面共用。每個圖面的程式記憶體中已預留了局部暫存器所需的空間。



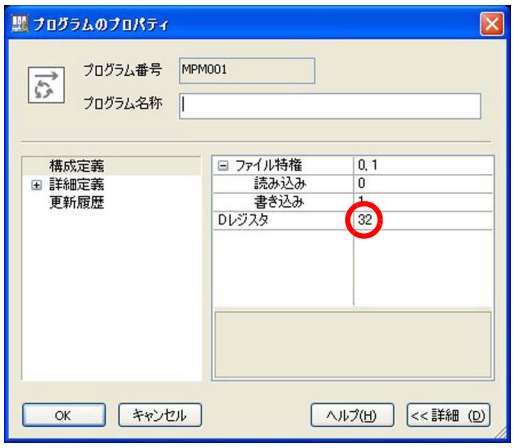
(註)D 暫存器的初始值為每個圖面可使用 32 個字元。

進入 [程式性質] 對話框，即可指定適用於每個圖面的暫存器範圍。
每個圖面最多可為局部暫存器預留 16384 個字元。
若要擴大 D 暫存器的範圍，請依照下列步驟來執行。

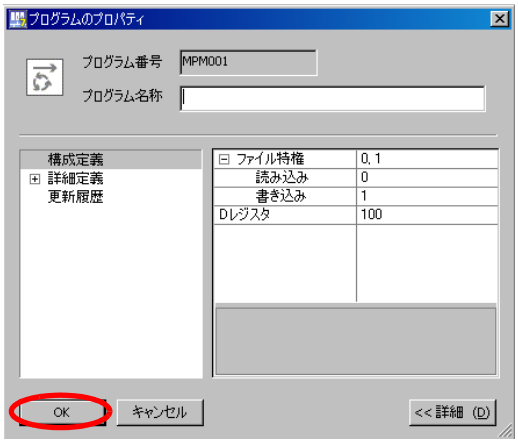
1. **在運動子視窗中的 [MPM001] 按一下右鍵，當選單出現後，請選擇 [性質]。**



2. **進入 [程式性質] 對話框，並將 D 暫存器的範圍從 32 改為 100。**



3. **請點擊 [OK] 鍵。**



如此即完成 D 暫存器擴充範圍的所有步驟。

資料類型

資料類型包含下表所示的位元型、整數型、長整數型、4 長整數型、實數型、倍精度實數型等，請依使用目的選擇適合的類型。

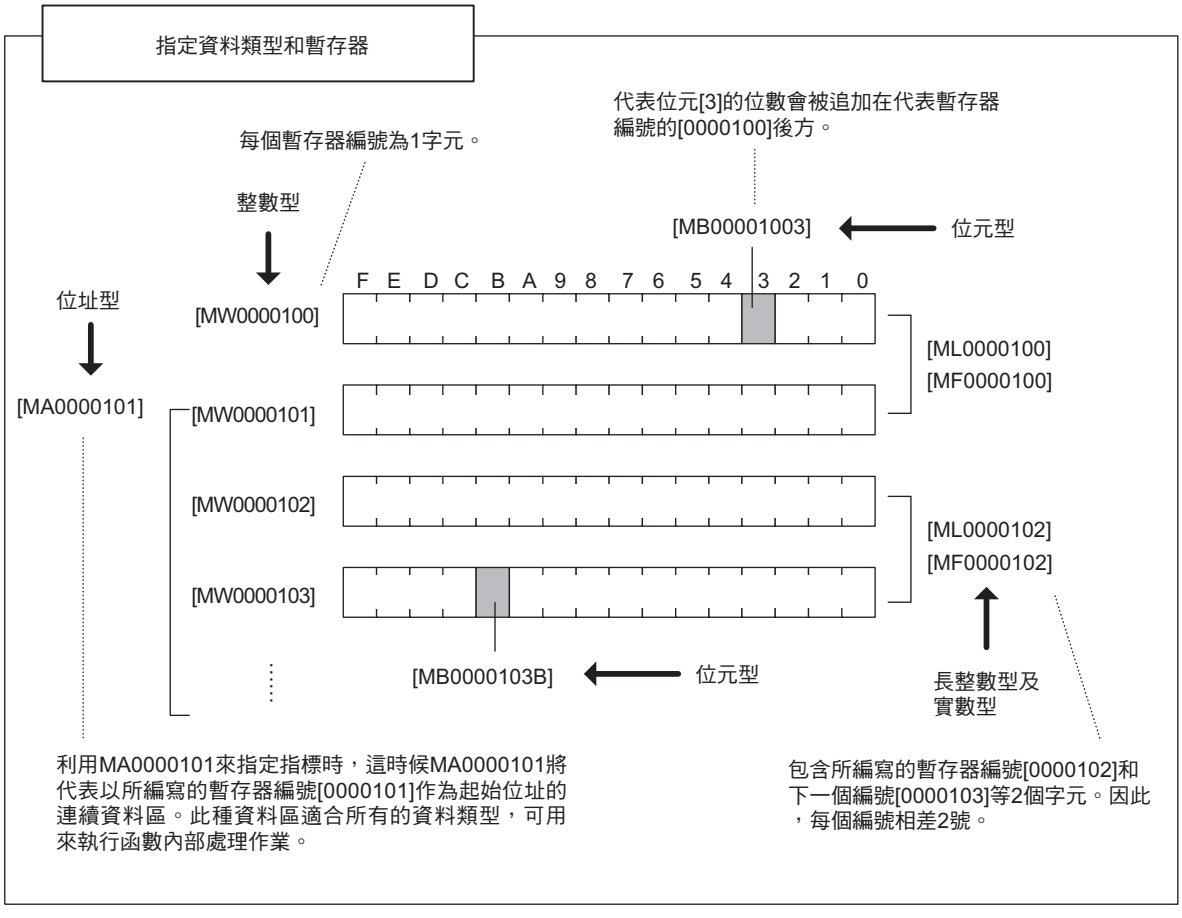
表 4.1 資料類型

符號	資料類型	數值範圍	資料大小	備註
B	位元	1 (開啟) (關閉)	–	適合用來判定繼電器序列或開啟 / 關閉條件。
W	整數	-32768 ~ +32767 (8000H ~ 7FFFH)	1 字元	作為數值演算之用。 左欄 (　) 內所示為使用邏輯演算方式時之數值。
L	長整數	-2147483648 ~ +2147483647 （80000000H ~ 7FFFFFFFH）	2 字元	作為數值演算之用。 左欄 (　) 內所示為使用邏輯演算方式時之數值。
Q	4 長整數 *1	-9223372036854775800 ~ 9223372036854775807 (8000000000000000H ~ 7FFFFFFFFFFFFFFFH)	4 字元	作為數值演算之用。 左欄 (　) 內所示為使用邏輯演算方式時之數值。
F	實數	±(1.175E-38 ~ 3.402E+38)、0	2 字元	適合高階的數值演算用途。*2
D	倍精度實數 *1	±(2.225E-308 ~ 1.798E+308)、0	4 字元	適合高階的數值演算用途。*2
A	位址	0 ~ 2097152	–	僅適合用來指定指標。

*1. 不適合用來間接指定運動程式。
*2. 符合 IEEE754 規範。

重要

MP3000 系列並未根據不同的資料類型內置暫存器。如下圖所示，只要位址相同，即使資料類型不同，也能存取同一個暫存器。
例如，位元型 MB00001003 和整數型 MW0000100 的資料類型雖然不同，卻同樣能存取 MW0000100。




專有名詞解說

指定指標

將位址當作引數傳給函數，就稱為指定指標。

指定指標後，以所編寫的暫存器編號作為起始位址的連續資料區將適合所有的資料類型，作為函數內部處理之用。


重要

暫存器運算注意事項

若資料被儲存在不同類型的暫存器時，將產生以下結果。

- 格式

  數值運算指令 = 使用 (代入)。

  暫存器放在左方，運算式則編寫在右方。

  MW00100 = MW00101 + MW00102;

- 暫存器運算

  將實數型資料儲存於整數型暫存器

  MW00100 = MF00200; 將實數值轉換為整數並加以儲存。

  (00001) (1.234)

  將實數資料儲存於整數型暫存器時，需特別注意捨入誤差。

  進入 [程式性質] 對話框，即可設定實數換算為整數時之捨入方法 (無條件捨去 / 四捨五入)。

  MW00100 = MF00200 + MF00202;

  (0124) (123.48) (0.02) 運算結果依您所要運算的暫存器數值而異。

  (0123) (123.49) (0.01)

  將實數型資料儲存於長整數型暫存器

  ML00100 = MF00200; 將實數值轉換為整數並加以儲存。

  (65432) (65432.1)

  將長整數型資料儲存於整數型暫存器

  MW00100 = ML00200; 長整數型資料的低階 16 位元將會直接被儲存起來。

  (-00001) (65535)

  將整數型資料儲存於長整數型暫存器

  ML00100 = MW00200; 整數型資料會先被轉換為長整數型資料後再儲存起來。

  (0001234) (1234)

- 語法錯誤範例

  整數資料被儲存在位元型暫存器中

  MB000100 = 123; ⇒ 語法錯誤

  MB000100 = MW00100; ⇒ 語法錯誤

4.2 暫存器的使用方法

以下將說明各種暫存器的使用方法。

系統暫存器 (S 暫存器)

系統暫存器 (S 暫存器) 係原本內置於運動控制器系統中的暫存器，可用來讀取系統錯誤資訊或運轉狀況。是一種同時適用於運動程式和序列程式的總體暫存器。

說明

S 暫存器的指定方法如下。

SB000000 ~ SB65534F
SW00000 ~ SW65534
SL00000 ~ SL65532
SQ00000 ~ SQ65528
SF00000 ~ SF65532
SD00000 ~ SD65528

您可利用 10 進位制格式來指定暫存器編號，但是指定位元時，則必須以 16 進位制格式，並從暫存器編號的下一位數開始指定。

程式範例

- **指定位元**
  OB00010 = SB000402 | SB000403;

- **指定整數**
  MW00100 = SW00041;

- **指定長整數**
  ML00100 = SL00062;


註記

系統暫存器 (S) 為讀取專用。若要寫入資料，系統將無法保證所執行的動作完全正確。

資料暫存器 (M 暫存器)

M 暫存器是一種適合階梯圖程式、使用者函數、各種運動程式及序列程式使用的通用型暫存器。

此種總體暫存器適用於運動程式、序列程式及階梯圖程式之間的介面。

說明

M 暫存器的指定方法如下。

MB00000000 ~ MB1048575F
MW0000000 ~ MW1048575
ML0000000 ~ ML1048574
MQ0000000 ~ MQ1048572
MF0000000 ~ MF1048574
MD0000000 ~ MD1048572

在各種運算過程中 M 暫存器扮演了 「暫存器」的角色，可代入運算結果，或是利用 「暫存器」來指定定位座標值或速度。您可利用 10 進位制格式來指定暫存器編號。

程式範例

◆ 以軸移動指令來指定位置，以暫存器來指定速度。

以下係指令單位為 mm，小數點以下位數有 3 位時之程式範例。

```
ML0000100 = 100000;                                    → 100.000 mm
ML0000102 = 200000;                                    → 200.000 mm
ML0000104 = 300000;                                    → 300.000 mm
ML0000106 = 500000;                                    → 500.000 mm/min
MVS [X]ML0000100 [Y]ML0000102 [Z]ML0000104 FML0000106;
```

◆ 將暫存器用於運算過程中

・指定位元

```
MB00001001 = IB0000100 & IB0000201;
```

・指定整數

```
MW0000101 = (MW0000101 | MW0000102) & FF0CH;
```

・指定長整數

```
ML0000200 = ((ML0000202 * ML0000204) / ML0000206) * 3;
```

・指定實數

```
MF0000200 = ((MF0000202 * MF0000204) / MF0000206) * 3.14;
```

重要

若要利用以下的運動語言指令來指定移動量座標值或是以暫存器來指定速度，則必須使用長整數型。
MOV，MVS，MCW/MCC，ZRN，SKP，MVT，EXM，POS，ACC，DCC，SCC，IAC，IDC，IFP，FMX，INP，IDH

資料暫存器 (G 暫存器)

資料暫存器 (G 暫存器) 是一種適合階梯圖程式、使用者函數、各種運動程式及序列程式使用的通用型暫存器。

運動程式和序列程式雖然能共用此種資料暫存器，不過它卻是一種無法以電池組備用的總體暫存器。

說明

G 暫存器的指定方法如下。

GB00000000 ~ GB2097151F
GW0000000 ~ GW2097151
GL0000000 ~ GL2097150
GQ0000000 ~ GQ2097148
GF0000000 ~ GF2097150
GD0000000 ~ GD2097148

您可利用 10 進位制格式來指定暫存器編號，不過，指定位元時，則必須以 16 進位制格式，並從暫存器編號的下一位數開始指定。

程式範例

以下為運算過程中所使用的暫存器範例。

- **指定位元**
  GB00001001 = IB0000100 & IB0000201;

- **指定整數**
  GW0000101 = (GW0000101 | GW0000102) & FF0CH;

- **指定長整數**
  GL0000200 = ((GL0000202 * GL0000204) / GL0000206) * 3;

- **指定實數**
  GF0000200 = ((GF0000202 * GF0000204) / GF0000206) * 3.14;


重要

若要利用以下的運動語言指令來指定移動量座標值或是以暫存器來指定速度，則必須使用長整數型。
MOV、MVS、MCW/MCC、ZRN、SKP、MVT、EXM、POS、ACC、DCC、SCC、IAC、IDC、IFP、FMX、INP、IDH

輸入暫存器 (I 暫存器)

適用於輸入資料及監控參數的暫存器。不過，僅能用來讀取監控參數。若要用來寫入資料，無法保證所執行的動作完全正確。

說明

I 暫存器的指定方法如下。

IB000000 ~ IB23FFFF
IW00000 ~ IW07FFF, IW10000 ~ IW17FFF … 輸入資料
IW08000 ~ IW0FFFF … 監控參數
IW20000 ~ IW23FFF … CPUIF 的輸入資料
IL00000 ~ IL23FFF
IQ00000 ~ IQ23FFC
IF00000 ~ IF23FFE
ID00000 ~ ID23FFC

輸入資料的暫存器編號取決於模組組成定義所設定的位址。

您可利用 16 進位制格式來指定暫存器編號。

程式範例

讀取輸入資料及監控參數。

- **指定位元**
  MB00001000 = IB0000010 & IB0000105;

- **指定整數**
  MW0000100 = IW08008;

- **指定長整數**
  ML0000100 = IL08004;

輸出暫存器 (O 暫存器)

適用於輸出資料及設定參數的暫存器。

說明

O 暫存器的指定方法如下。

OB000000 ~ OB23FFFF
OW00000 ~ OW07FFF，OW10000 ~ OW17FFF … 輸出資料
OW08000 ~ OW0FFFF … 設定參數
OW20000 ~ OW23FFF … CPUIF 的輸出資料
OL00000 ~ OL23FFF
OQ00000 ~ OQ23FFC
OF00000 ~ OF23FFE
OD00000 ~ OD23FFC

輸出資料的暫存器編號取決於模組組成定義所設定的位址。

您可利用 16 進位制格式來指定暫存器編號。

程式範例

寫入輸出資料及設定參數。

- **指定位元**
  OB01000 = MB00001000 & IB0000100;

- **指定整數**
  OW08008 = MW0000100;

- **指定長整數**
  OL08010 = ML0000100+ML0000200;

C 暫存器 (C 暫存器)

C 暫存器為程式參照時所使用的暫存器，無法寫入。

說明

C 暫存器的指定方法如下。

CB000000 ~ CB16383F
CW00000 ~ CW16383
CL00000 ~ CL16382
CQ00000 ~ CQ16380
CF00000 ~ CF16382
CD00000 ~ CF16380

C 暫存器不具備從程式將資料寫入的功能。

您可利用 10 進位制格式來指定暫存器編號。

程式範例

以下係在運算過程中使用暫存器之範例。

- **指定位元**
  MB00001000 = CB001001;
- **指定整數**
  MW0000100 = CW00100;
- **指定長整數**
  ML0000100 = CL00100;
- **指定為 4 長整數**
  MQ0000100 = CQ00100;
- **指定實數**
  MF0000100 = CF00100;
- **指定為倍精度實數**
  MD0000100 = CD00100;

D 暫存器 (D 暫存器)

為運動程式及序列程式原本內置的暫存器，僅可適用於特定程式的暫存器。

說明

D 暫存器的指定方法如下。

DB000000 ~ DB16383F
DW00000 ~ DW16383 (最大)
DL00000 ~ DL16382
DQ00000 ~ DQ16380
DF00000 ~ DF16382
DD00000 ~ DD16380

利用上述暫存器將運算結果代入各種運算的暫存器中，或是利用暫存器來指定定位座標值及速度。您可利用 10 進位制格式來指定暫存器編號，不過，指定位元時，則必須以 16 進位制格式，並從暫存器編號的下一位數開始指定。進入程式架構定義 ([程式性質]) 對話框，即可設定大小 (初始值 : 32 字元)。

程式範例

◆ 以軸移動指令來指定位置，以暫存器來指定速度時

以下範例係以 mm 為指令單位，並假設小數點以下位數為 3。

```
DL00100 = 100000;                                   → 100.000 mm
DL00102 = 200000;                                   → 200.000 mm
DL00104 = 300000;                                   → 300.000 mm
DL00106 = 500000;                                   → 500.000 mm/min
MVS [A1]DL00100 [B1]DL00102 [C1]DL00104 FDL00106;
```

◆ 將暫存器用於運算過程中時

· 指定位元

```
DB001000 = IB0001001 & MB00000101;
```

· 指定整數

```
DW00102 = ( CW00103 | DW00104 ) & DW00105;
```

· 指定長整數

```
DL00106 = ( DL00108 * ML0000011 ) / ML0000200;
```

· 指定實數

```
DF00200 = ( MF0000202 * DF00202 ) * 3.14;
```


重要

若要利用以下的運動語言指令來指定移動量座標值或是以暫存器來指定速度，則必須使用長整數型。
MOV、MVS、MCW/MCC、ZRN、SKP、MVT、EXM、POS、ACC、DCC、SCC、IAC、IDC、IFP、FMX、INP、IDH

4.3 索引 i、j 的使用方法

這是一種用來修飾繼電器編號及暫存器編號的專用暫存器，本系統內置 i 和 j 2 種暫存器。i 和 j 的功能完全相同。暫存器編號可當作變數使用。

指定索引 i、j 作為變數時，必須以 10 進位制的格式來指定。

接下來將以暫存器的各種資料類型為例，進行說明。

■ 位元型附加索引

就像暫存器編號被加上 i 或 j 的數值一樣。

例如，當 i = 2 時，MB00000000i 和 MB00000002 相同。

```
i = 2;
DB000000 = MB00000000i;   <--- 等價 --->   DB000000 = MB00000002;
```

■ 整數型附加索引

就像暫存器編號被加上 i 或 j 的數值一樣。

例如，當 j = 30 時，MW0000001i 和 MW0000031 相同。

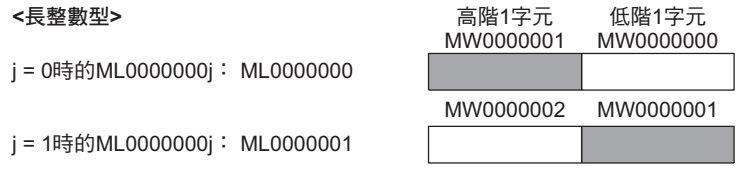
```
j = 30;
DW00000 = MW0000001j;   <--- 等價 --->   DW00000 = MW0000031;
```

■ 長整數型及實數型附加索引

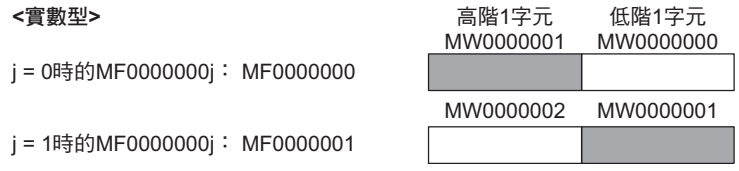
就像暫存器編號被加上 i 或 j 的數值一樣。

例如，當 j = 1 時，ML0000000j 和 ML0000001 相同。此外，當 j = 1 時，MF0000000j 和 MF0000001 相同。

<長整數型>



<實數型>



註記

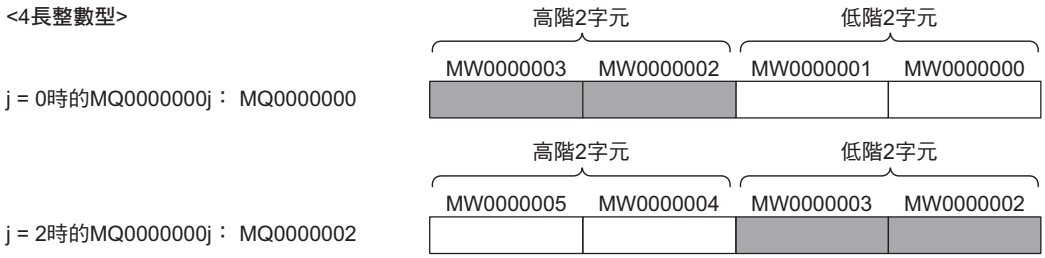
若為長整數型或實數型時，系統所使用的區域相當於 2 字元。例如，j = 0 時的 ML0000000j 和 j = 1 時的 ML0000000j，其中 MW0000001 的 1 字元的區域重複。若長整數型和實數型附加索引，則必須注意區域是否重複。

■ 4 長整數型及倍精度實數型附加索引

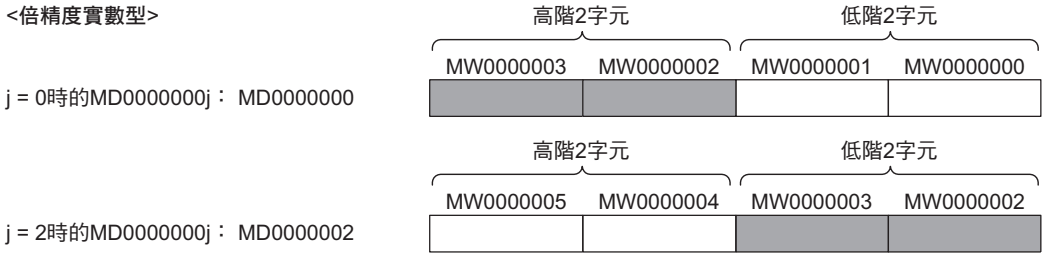
就像暫存器編號被加上 i 或 j 的數值一樣。

例如，當 j = 2 時，MQ0000000j 和 MQ0000002 相同。此外，當 j = 2 時，MD0000000j 和 MD0000002 相同。

<4長整數型>



<倍精度實數型>



註記 若為 4 長整數型或倍精度實數型時，系統所使用的區域相當於 4 字元。例如，j = 0 時的 MQ0000000j 和 j = 2 時的 MQ0000000j，其中 MW0000002、MW0000003 的 2 字元的區域重複。若 4 長整數型和倍精度實數型附加索引，則必須注意區域是否重複。

■ 程式範例

以下為使用索引之程式範例。

使用索引 j，並將 ML0000100~ML0000198 共 50 個暫存器的總和加到 ML0000200 中。

```
   :
   :
ML0000200 = 0 ;
J = 0 ;
WHILE J < 100 ;
   ML0000200 = ML0000200 + ML0000100J ;
   J = J + 2 ;
WEND ;
   :
   :
```

補充 索引 i、j 可用大寫 / 小寫其中一種形式來編寫。

```
j = 0;
J = 0;
DW00000 = MW0000000j ;
DW00000 = MW0000000J ;
```

4.4 陣列暫存器的使用方法

所謂「陣列暫存器」是一種用來修飾暫存器編號的專用暫存器。

適合以暫存器編號為變數時使用。

和索引一樣，亦可將偏移值加到暫存器編號中。

■ 位元型附有陣列時

就像暫存器編號被加上陣列數值一樣。

例如，當 DW00000 = 2 時，MB00000000 [DW00000] 和 MB00000002 相同。

```
DW00000 = 2;
DB000020 = MB00000000[DW00000];   等價   DB000020 = MB00000002;
```

■ 位元型以外類型附加陣列時

就像暫存器編號被加上 (陣列數值 x 資料型字元大小) 一樣。

例如，當 DW00000 = 30 時，ML0000002 [DW00000] 和 ML0000062 相同。

DL00002 = ML00000 (30×2 + 2) = ML0000062

```
DW00000 = 30;
DL00002 = ML0000002[DW00000];   等價   DL00002 = ML0000062;
```

■ 格式

下表可用來說明陣列暫存器的格式。

MOV[A1]ML00000[MW00100];
①　②

內容	使用用途	適用之暫存器
①	陣列名稱	・所有資料類型的暫存器 (#、C 暫存器除外)
②	陣列要素	・整數型、長整數型的所有暫存器 (#、C 暫存器除外) ・常數 ・索引暫存器

■ 程式範例

以下程式範例係使用陣列暫存器，並將 ML0000100~ML0000198 共 50 個暫存器的總和加到 ML0000200 中。

```
ML0000200 = 0;
DW00000 = 0;
WHILE DW00000 < 50;
  ML0000200 = ML0000200 + ML0000100[DW00000];
  DW00000 = DW00000 + 1;
WEND;

END;
```

程式編寫規則

5

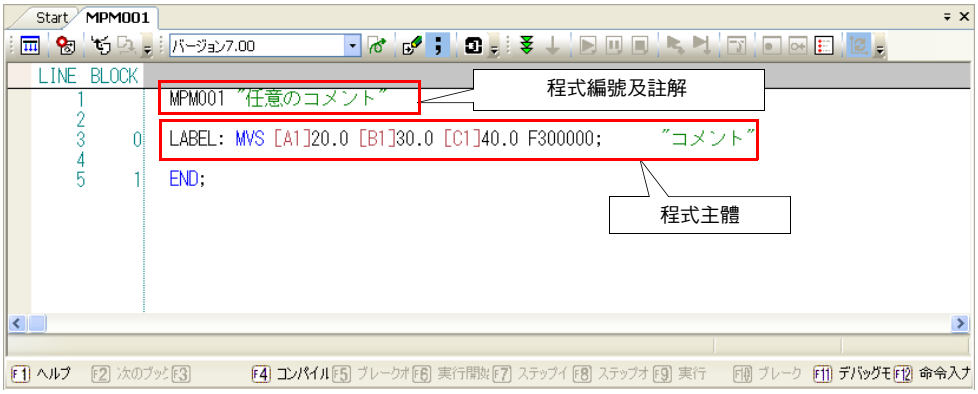
本章將針對運動程式及序列程式的編寫規則進行說明。

5.1 程式的編寫

接下來將說明如何編寫運動程式及序列程式。
運動程式和序列程式的編寫方式相同。

運動程式的編寫結構

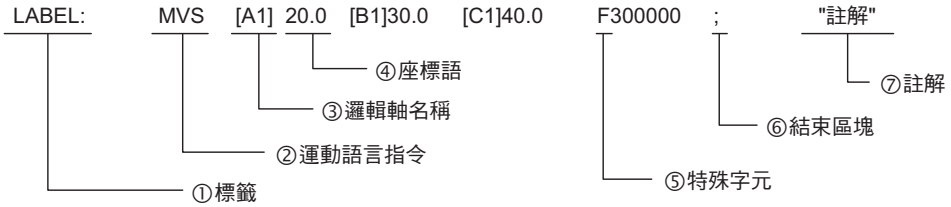
運動程式係由程式編號、註解、程式主體及 END 指令等所組成。程式主體必須編寫運動程式的處理作業。
下圖為運動程式的編寫結構。



補充　編寫時，不一定非得指定「程式編號及註解」行。

區塊的格式

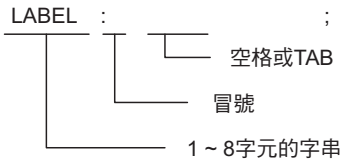
所謂「區塊」就是程式進行處理時的執行單位。程式主體係由 1 個以上的區塊所組成。
運動程式區塊的格式如下。



No.	項目	代表意義
①	標籤	用來表示 PFORK 指令、SFORK 指令的分歧標的。
②	運動語言指令	用來指定運動程式的指令。
③	邏輯軸名稱	用來指定群組定義時所設定之邏輯軸名稱。
④	座標語	用來指定軸座標值或軸累加移動量。
⑤	特殊字元	用來指定運動語言指令的輔助資料。
⑥	結束區塊	用來指定區塊的結束點。
⑦	註解	用來編寫程式註解。

標籤

標籤係由使用英文數字 / 符號的 1 ~ 8 字元的字串以及冒號 「：」、空格 (Space) 或 TAB 所組成。



類型	適合當作標籤的英文字母、數字 / 符號
英文字母	A ~ Z ，a ~ z
數字 *	0 ~ 9
符號	– (連字號)

* 標籤首字禁止使用數字。

使用平行處理 (PFORK)、選擇 (SFORK) 等功能時，需加上標籤。
若不使用平行處理 (PFORK)、選擇 (SFORK)，則不需要編寫標籤。

範例 **標籤編寫範例**

```
PFORK LAB1, LAB2;
LAB1: ZRN [A1]0 [B1]0 [C1]0;
JOINTO LAB3;
LAB2: MVS [D1]100.0 [E1]200.0 [F1]300.0;
JOINTO LAB3;
LAB3: PJOINT;
```

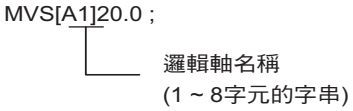
運動語言指令

編寫運動語言指令。
如欲進一步瞭解運動語言指令，請參閱以下章節。

☞ **第 6 章 運動語言指令**

邏輯軸名稱

編寫群組定義所設定的邏輯軸名稱時，需加上括號 ([ ])。



類型	適合當作邏輯軸名稱的字元
英文字母	A ~ Z ，a ~ z
數字	0 ~ 9

座標語

所謂座標語係指編寫在軸名稱後方的數值及變數。可用來指定指令位置、速度或加減速的速度。

◆ 利用數字來編寫座標語。

在軸名稱後方直接指定數值。
雖然整數、實數皆可被指定為數值，但指定整數時需特別注意。
將指令單位設定為 0.001 mm 時，只要在座標語輸入指令位置 「1000」，控制器內部就會解讀為 1.000 mm。輸入實數 「1.000」後，將直接解讀為 1.000 mm。

MVS [A1]1000; → 1.000 mm
或是
MVS [A1]1.000; → 1.000 mm
或是
MVS [A1]1.; → 1.000 mm

◆ 利用暫存器來編寫座標語。

利用長整數型暫存器，即可在軸名稱後方間接指定暫存器。

間接指定暫存器時，只要將指令單位設定為 0.001 mm，並設定為和上述範例所示的整數值輸入值相同的暫存值 = 1000，控制器內部將解讀為 1,000mm。

ML00000 = 1000;
MVS [A1]ML00000; → 1.000 mm

補充　座標語的單位依運動語言指令或運動參數的設定而異。

◆ 特殊字元

以下為各種特殊字元的意義及使用範例一覽表。

字元	代表意義	使用範例	參考標的
M	加減速模式	ACCOMDE M2;	設定插補加減速模式 (ACC-MODE)
F	插補進給速度	MVS [A1]1000 [B1]2000 F3000000; MVS [A1]1000 [B1]2000 FML00000;	線性內插 (MVS)
FW	連結處理控制訊號	MVS [A1]1000 FWMB00000000;	線性內插 (MVS)
T	插補進給最高速度	FMX T30000000; FMX TML00000;	設定插補進給最高速度 (FMX)
	時間設定值	TIM T100; TIM TML00000; TIM1MS T100; TIM1MS TMW0000000; MVT [A1]1000 [B1]2000 T100; MVT [A1]1000 [B1]2000 TML00000; IAC T100; IAC TML00000; IDC T100; IDC TML00000; IDH T100; IDH TML00000;	等待時間 (TIM) 等待時間 (TIM1MS) 指定時間定位 (MVT) 變更插補加速時間 (IAC) 變更插補減速時間 (IDC) 設定暫停時的插補減速時間 (IDH)
	循環轉數	MCW [A1]1000 [B1]2000 U500 V500 T2 F3000000; MCW [A1]1000 [B1]2000 U500 V500 TML00000 F3000000;	循環內插 < 指定中心位置 > (MCW、MCC) 循環內插 < 指定半徑 > (MCW、MCC)
TW	連結處理控制訊號	MVS [A1]1000 TWMB00000000;	線性內插 (MVS)
R	圓弧半徑	MCW [A1]1000 [B1]2000 R500 F3000000; MCW [A1]1000 [B1]2000 RML00000 F3000000;	循環內插 < 指定中心位置 > (MCW、MCC) 循環內插 < 指定半徑 > (MCW、MCC)
U	圓弧中心座標 1 (橫軸)	MCW [A1]1000 [B1]2000 U500 V500 T2 F3000000; MCW [A1]1000 [B1]2000 UML00000 V500 T2 F3000000;	循環內插 < 指定中心位置 > (MCW、MCC) 循環內插 < 指定半徑 > (MCW、MCC)
V	圓弧中心座標 2 (縱軸)	MCW [A1]1000 [B1]2000 U500 V500 T2 F3000000; MCW [A1]1000 [B1]2000 U500 VML00000 T2 F3000000;	循環內插 < 指定中心位置 > (MCW、MCC) 循環內插 < 指定半徑 > (MCW、MCC)
P	不同比率之插補進給速度	IFP P50; IFP PML00000;	設定插補進給速度比率 (IFP)
SS	選擇略過訊號	SKP [A1]1000 [B1]2000 F3000000 SS1; SKP [A1]1000 [B1]2000 F3000000 SS2;	附略過功能線性內插 (SKP)
D	插補重疊距離	MVS [A1]1000 D1000; MVS [A1]1000 DML0000000;	線性內插 (MVS)
	外部定位移動量	EXM [A1]1000 D1000; EXM [A1]1000 DML00000;	外部定位 (EXM)
N	位移數	SFR MB001000 N5 W10; SFR MB001000 NMW00000 W10;	位元右移 (SFR) 位元左移 (SFL)
W	位元寬度	BLK MW00100 DW00100 W10; BLK MW00100 DW00100 WMW00000;	位元右移 (SFR) 位元左移 (SFL) 傳送區塊 (BLK) 清除 (CLR)
MPS	運動子程式編號	MSEE MPS002;	叫出運動子程式 (MSEE)

(續下頁)

(接上頁)

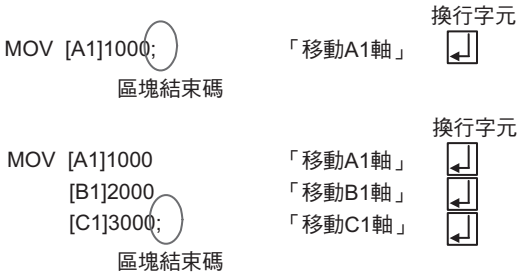
字元	代表意義	使用範例	參考標的
SPS	序列子程式編號	SSEE SPS002;	叫出序列子程式 (SSEE)

結束區塊

結束區塊需寫上分號 (;)。區塊可由多行所組成。編寫結束區塊，以指定區塊結束點。

結束區塊的後面必須換行。

範例 **結束區塊編寫範例**



註解

註解符號包含 「”」和 「//」 2 種格式。

◆ 使用雙斜線來編寫字串

從雙斜線到後面的換行 (Enter) 所有的字元皆被視為註解。

換行字元

// 文字列 ↵

範例

註解編寫範例 2

```
// 所有軸原點復歸
ZRN [A1]0 [B1]0 [C1]0;

//3 軸線性內插
MVS [A1]100.0 [B1]200.0 [C1]300.0;
```

補充　除了半形英文字母或數字外，亦可使用包含漢字的全形文字作為字串 (註解)。

◆ 使用雙引號來編寫字串

· 在兩個雙引號間編寫所要的字串。

雙引號之間的字串即被視為 「註解」。

“ 字串 ”

範例

註解編寫範例 1

```
ZRN [A1]0 [B1]0 [C1]0;                  " 所有軸原點復歸 "
MVS [A1]100.0 [B1]200.0 [C1]300.0;   "3 軸線性內插 "
```

· 在一個雙引號後面加上字串 (註解)

從一個雙引號到後面的換行符號 (Enter) 所有的字元皆被視為註解。

換行文字

" 文字列 ↵

範例

註解編寫範例 2

```
" 所有軸原點復歸
ZRN [A1]0 [B1]0 [C1]0;

"3 軸線性內插
MVS [A1]100.0 [B1]200.0 [C1]300.0;
```

補充　除了半形英文字母或數字外，亦可使用包含漢字的全形文字作為字串 (註解)。

常數和暫存器的標記方法

接下來將說明常數和暫存器的標記方法。

常數的標記方法

以下為運動程式所適用的常數。

分類	範圍	編寫範例
10 進位制整數	-9223372036854775808 ~ 9223372036854775807	823，-2493，123k，123K
16 進位制整數	0000000000000000H ~ FFFFFFFFFFFFFFFFH	FFFABCDEH，2345H，FH
實數	-9223372036854775808 ~ 9223372036854775807 依小數點位置而異	763.0，824.2，234.56，-321.12345

補充

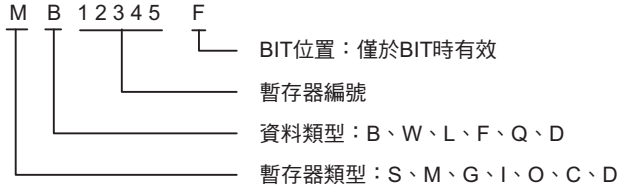
1. - (負) 號不可省略，但 + (正) 號則可省略不寫。
   [A1]+123 ⇒ [A1]123
   [A1]-123 ⇒ [A1]-123
2. 10 進制整數加上 K 後，數值就會變為原來的 1000 倍。可避免因位置指令值的 0 太多，而不易判讀。
   [A1]123k ⇒ [A1]123000
   [A1]123K ⇒ [A1]123000

暫存器的標記方法

以下為運動程式所適用的暫存器。

分類	暫存器類型	資料類型					
		BIT	WORD	LONG	FLOAT	QUAD	DOUBLE
總體暫存器	S 暫存器	SB	SW	SL	SF	SQ	SD
	M 暫存器	MB	MW	ML	MF	MQ	MD
	G 暫存器	GB	GW	GL	GF	GQ	GD
	I 暫存器	IB	IW	IL	IF	IQ	ID
	O 暫存器	OB	OW	OL	OF	OQ	OD
	C 暫存器	CB	CW	CL	CF	CQ	CD
局部暫存器	D 暫存器	DB	DW	DL	DF	DQ	DD

範例 暫存器編寫範例



標記常數和暫存器時，有時可省略 0，有時則否。

範例 **可省略 0**

[A1]00123 ⇒ [A1]123
[A1]MW00010 ⇒ [A1]MW10
[A1]100.000 ⇒ [A1]100.

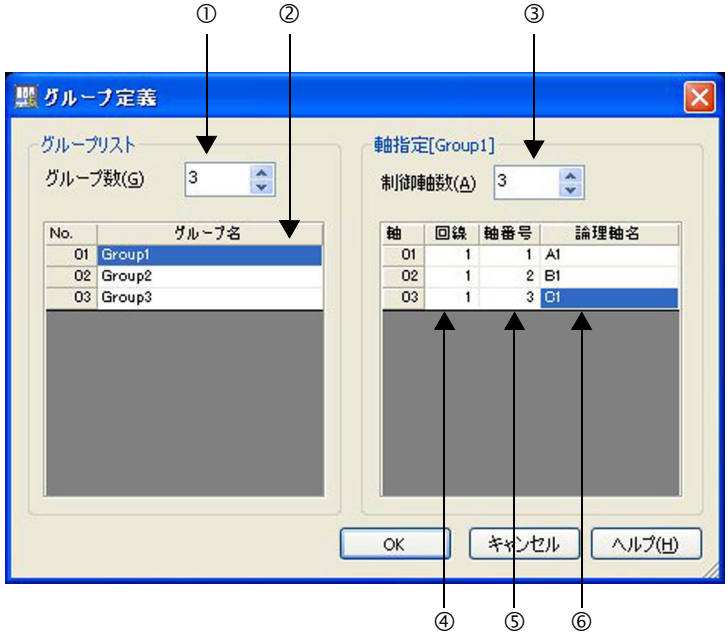
範例 **不可省略 0**

MPM001; (編寫於程式起始的程式編號)
MSEE MPS002;

5.2 群組定義說明

所謂「群組定義」就是在定義時將多個軸彙整為同一個群組。

接下來，將說明 [群組定義] 對話框。



① 群組數

設定群組運轉數量。
使用 1 個群組來運轉時，請設定為「1」。
使用多個群組來運轉時，請設定群組運轉的數量。

② 群組名稱

定義群組名稱。

③ 控制軸數

設定群組所能控制的轉軸數。

④ 線路

設定您所要使用的運動控制功能的線路編號。
利用模組架構定義即可確認線路編號。

線路編號

⑤ 軸編號

設定您所要使用的轉軸編號。
進入模組架構定義並點擊 SVC32 的 + 鍵，即可確認轉軸編號。

⑥ 邏輯軸名稱

定義指定的轉軸編號名稱。
此處所定義的名稱適合在運動程式編寫時使用。

```
MVS  [A1]1000  [B1]2000  [C1]3000  F1000;
```

邏輯軸名稱

5.3 運算時的優先順序

運動語言指令在運算時具有優先順序。
若要同時執行 3 項以上的運算，請使用 (　)，以指定運算的優先順序。
下表為運算的優先順序。

高 ⟶ 低

運算子	優先順序				
	1	2	3	4	5
括弧	()				
NOT		!			
AND			&		
OR				|	
XOR				^	
數值運算				* /	+ -

範例 **數值運算範例**

・ 運算範例
MW00100 = 1 + 2;
在上述算式中，程式先運算 1 + 2 = 3，再將結果代入 MW00100 中

・ 3 項以上之運算範例
MW00100 = 1 + (2 * 3);
在上述算式中，程式先將 2 x 3 = 6 的運算結果和 1 相加，然後再將結果代入 MW00100。
因此得到 MW00100 = 7 的結果。

註記 **3 項以上之演算的注意重點**

假設我們編寫了一個算式如下 :
MW00100 = 1 + 2 * 3;
運算時需依照上表所示的優先順序，因此第一步先運算 2 x 3 = 6，再計算 1 + 6 = 7。
最後再將結果代入 MW00100。因此得到 MW00100 = 7 的結果。

範例 **邏輯運算範例**

・ 運算範例
MW00100 = 0001H | 0002H;
上述算式係先運算 0001H 和 0002H 的邏輯和 (OR)，然後再將結果代入 MW00100 中。

・ 3 項以上的運算範例
MW00100 = (1111H | 2222H) & 00FFH;
上述算式係先運算 1111H 和 2222H 的邏輯和 (OR) 以及 00FFH 的邏輯積 (AND)，接著再將結果代入 MW00100 中。
因此得到 MW00100 = 0033H 的結果。

註記 **3 項以上之演算的注意重點**

假設我們編寫了一個算式如下。
MW00100 = 1111H | 2222H & 00FFH;
上述算式係先運算 2222H 和 00FFH 的邏輯積 (AND) 以及 1111H 的邏輯和 (OR)，再將結果代入 MW00100 中。
因此得到 MW00100 = 1133H 的結果。

註記

編譯器版本「版本 6 互換」注意重點

以下為編譯器版本「版本 6 互換」運算時之優先順序。

運算子	優先順序			
	1	2	3	4
括弧	()			
NOT		!		
AND			&	
OR				|
XOR				^
四則運算				+ - * /

假設我們編寫了一個算式如下。

MW00100 = 1 + 2 * 3;

運算時需依照上表所示的優先順序，因此第一步先運算 1 + 2 = 3，再計算 3 x 3 = 9。

最後再將結果代入 MW00100。因此得到 MW00100 = 9 的結果。

5.4 指令類型和執行掃描動作

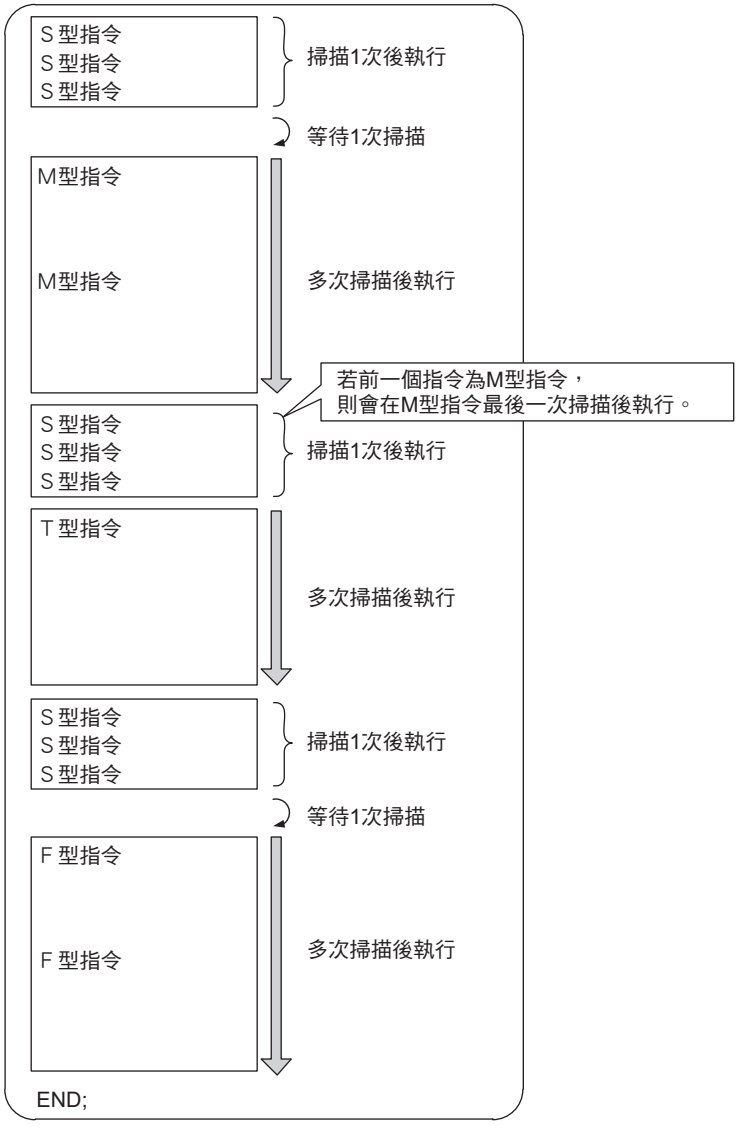
本節將針對指令類型及執行掃描動作進行說明。

指令類型

運動語言指令包含 4 種指令類型。指令類型不同，所需執行的掃描次數亦各異。下表為指令類型的種類及所需執行的掃描次數。

指令類型	指令	所需要的掃描次數
S 型	運算指令	掃描 1 次
M 型	軸移動相關指令	多次掃描
T 型	計時器相關指令	
F 型	傳送指令相關指令	

以下為各指令類型的動作示意圖。



■ S 型指令

S 型指令包含運算指令，是一種掃描 1 次後就結束的指令。
S 型指令為連續型程式，因此會在同一次掃描時進行處理。

■ M 型指令

M 型指令包含軸移動指令，因此必須等待好幾次掃描才能執行指令。
從 S 型指令切換到 M 型指令約需等待 1 次掃描的時間。

■ T 型指令

T 型指令包含計時器指令，因此必須等待好幾次掃描才能執行指令。

■ F 型指令

將指令從 CPU 單元 /CPU 模組傳送到選購元件時，必須等待好幾次掃描的時間。
從 S 型指令切換到 F 型指令約需等待 1 次掃描的時間。

指令類型一覽表

下表為指令類型一覽表。

分類	指令	S 型	M 型	T 型	F 型
軸設定指令	ABS	○			
	INC	○			
	ACC		○		
	DCC		○		
	SCC		○		
	VEL		○		
	FMX	○			
	IFMX	○			
	IFP	○			
	FUT	○			
	IAC	○			
	IDC	○			
	IDH	○			
	IUT	○			
	＋、-		○		
	ACCMODE	○			
軸移動指令	MOV		○		
	MVS		○		
	MCW		○		
	MCC		○		
	ZRN		○		
	DEN		○		
	SKP		○		
	MVT		○		
	EXM		○		
控制指令	POS	○			
	MVM	○			
	PLD	○			
	PFN		○		
	INP		○		
	PFP		○		
	PLN	○			
程式控制指令	IF ELSE IEND	○			
	WHILE WEND	○			
	WHILE WENDX			○	
	PFORK JOINTO PJOINT	○			
	SFORK JOINTO SJOINT	○			
	MSEE	○			
	SSEE	○			
	UFC		○		
	FUNC	○			
	END	○			
	RET	○			
	TIM			○	

(續下頁)

指令類型一覽表

(接上頁)

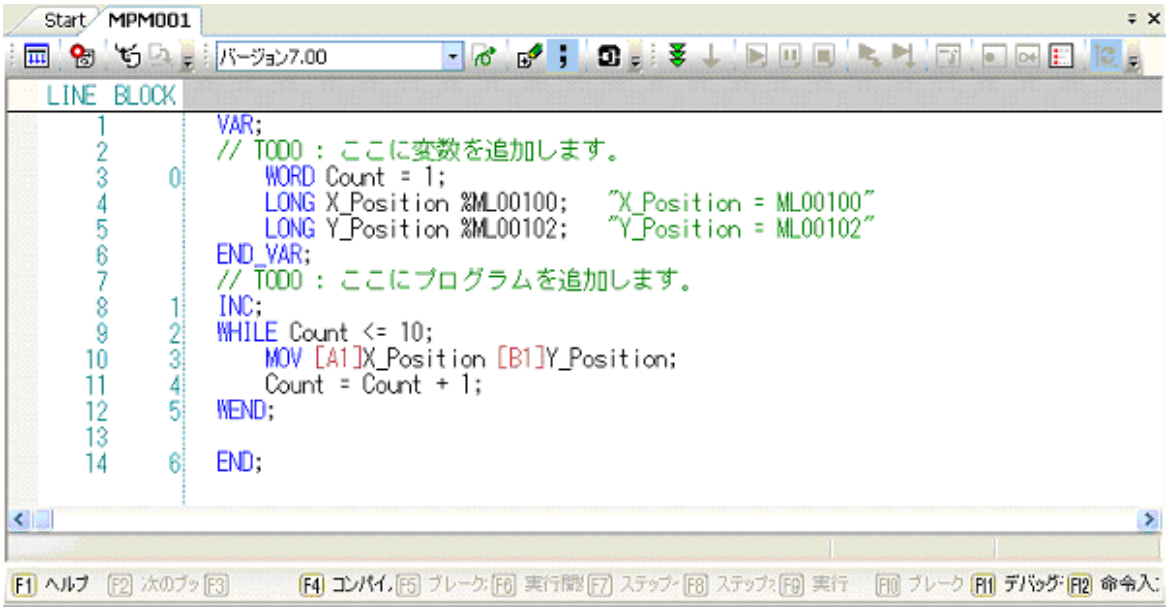
分類	指令	S 型	M 型	T 型	F 型
程式控制指令	TIM1MS			○	
	IOW			○	
	EOX			○	
	SNGD/SNGE	○			
數值運算	=	○			
	+	○			
	-	○			
	+ +	○			
	- -	○			
	*	○			
	/	○			
	MOD	○			
邏輯運算	|	○			
	&	○			
	^	○			
	!	○			
數值比較	= =	○			
	<>	○			
	>	○			
	<	○			
	>=	○			
	<=	○			
資料操作	SFR	○			
	SFL	○			
	BLK	○			
	CLR	○			
	SETW	○			
	ASCII	○			
基本函數	SIN	○			
	COS	○			
	TAN	○			
	ASN	○			
	ACS	○			
	ATN	○			
	SQT	○			
	BIN	○			
	BCD	○			
	S{ }	○			
	R{ }	○			
	PON	○			
	NON	○			
	TON	○			
	TON1MS	○			
	TOF	○			
	TOF1MS	○			
影像指令	VCAPI				○
	VCAPS				○
	VFIL				○
	VANA				○
	VRES				○

5.5 變數編寫

所謂「變數編寫」就是利用使用者所宣告的任一個字串 (變數) 來編寫程式的一種方式。

宣告變數後即可進行程式編寫，完全不需要考慮暫存器，因此大大提升了程式通用性及擴充性。

宣告過的變數僅適用於同一個程式。



註記

變數編寫僅適用於編譯器版本 「版本 7.00」。

若編譯器版本為 「版本 6 互換」時，將會造成編譯錯誤。

變數宣告

請將您所要宣告的變數編寫在 VAR 和 END_VAR 間的區塊中。

每個程式可宣告變數之數量為 1000。

宣告過的變數，在 END_VAR 之後可與暫存器相等使用。

```
VAR;
  編寫您所要宣告的變數
END_VAR;
```

變數的格式

變數係由資料類型、半形英文字母 / 數字、使用符號的 1 ~ 255 個字元的字串、分號「;」所組成。

每個程式可宣告的變數大小為 16384 個字元。

```
VAR;
 LONG  Data  ;
  資料類型   變數名稱   程式結束
END_VAR;
```

下表為適用的資料類型。

資料類型	內容
BOOL	位元
WORD/SINT	1 字元帶符號整數
LONG/DINT	2 字元帶符號整數
QUAD/LONGLONG/LINT	4 字元帶符號整數
FLOAT/REAL	單精度浮動小數點
DOUBLE/LREAL	倍精度浮動小數點
ADDRESS	位址
結構體名稱	結構體

(註)位址型不適用陣列。

下表為適合當作變數名稱的字元。

變數名稱類型	適用的字元
英文字母	A ~ Z, a ~ z
數字	0 ~ 9
符號	_ (底線)

(註)數字不適用於變數名稱的起始。

如何指定為初始值

格式如下。

```
VAR;
 資料類型   變數名稱 = 初始值
END_VAR;
```

補充　暫存器不得被指定為初始值。

範例　**指定初始值時編寫**

```
VAR;
 BOOL  Complete      = 1;
 LONG  Vel           = 1000 ;
 LONG  Position[3]   = {1000, 2000, 3000} ;
END_VAR;
```

如何和暫存器建立相關性

可讓宣告過的變數和您所指定的暫存器數值一致。

格式如下。

```
VAR;
  資料類型   變數名稱   % 暫存器 = 初始值；
END_VAR;
```

補充

1. 亦可選擇省略初始值的格式。
2. #、C 暫存器以外的所有暫存器皆適用。

範例 **編寫一個讓宣告過的變數和所指定的暫存器數值一致的程式**

```
VAR;
  BOOL  Complete      %OB00010;
  LONG  Vel           %ML00200  =  1000 ;
  LONG  Position[3]   %ML00300  =  {1000, 2000, 3000} ;
END_VAR;
```

如何指定常數值

以下格式可用來指定常數值。

```
VAR;
 CONST   資料類型      變數名稱 = 常數值；
END_VAR;
```

補充 上述格式無法和暫存器建立相關性。

範例 **指定常數值時編寫**

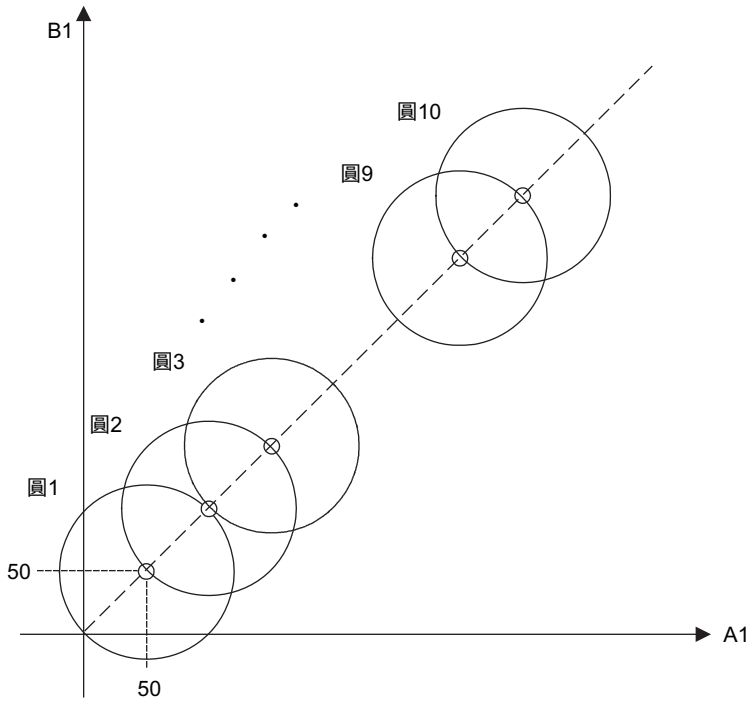
```
VAR;
  CONST  WORD  MotionCMD_NOP      =  0 ;
  CONST  WORD  MotionCMD_HOME     =  9 ;
  CONST  LONG  MaxSpeed           =  6000 ;
END_VAR;
```

程式範例

以下為使用變數之程式範例。

本程式範例係讓 X 軸每次移動 50 (指令單位)，Y 軸每次移動 50 (指令單位)，並畫出半徑為 50 (指令單位) 的圓 10 次。

```
VAR;
      WORD Count;                              " 計數器 "
 CONST WORD CountNum = 10;                     " 迴圈次數 "
      LONG X_radius %ML00100;                  "A1 軸半徑 "
      LONG Y_radius %ML00102;                  "B1 軸半徑 "
      LONG Speed = 8000;                       " 插補進給速度 "
END_VAR;

ZRN [A1]0 [B1]0;
Count = 1;                                     " 計數器預設 "
INC;
PLN [A1][B1];
FMX T80000;
WHILE Count <= CountNum;                       " 指定次數迴圈 "
 MCW [A1]0 [B1]0 U X_radius V Y_radius F Speed; " 循環內插 "
 MOV [A1]X_radius [B1]Y_radius;                " 定位 "
 Count = Count+1;                              " 計數器增量 "
WEND;

END;
```



運動語言指令

6

本章將說明運動語言指令的參考。

6.1 軸設定指令

所謂軸設定指令係指一種用來設定轉軸移動時加減速或速度的指令。
軸設定指令包含 16 種指令，僅適用於運動程式。

下表為軸設定指令一覽表。

指令	名稱	格式	內容	運動程式	序列程式
ABS	絕對模式	ABS; 或是 ABS MOV [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 ;	接下來的座標將被視為絕對值 (Absolute value) 處理。	○	×
INC	增量模式	INC; 或是 INC MOV [邏輯軸名稱 1] － [邏輯軸名稱 2] － ;	接下來的座標將被視為增量值 (Incremental value) 處理。	○	×
ACC	變更加速時間	ACC [邏輯軸名稱 1] 加速時間 [邏輯軸名稱 2] 加速時間 [邏輯軸名稱 3] 加速時間・・・;	用來變更定位類指令的加速時間。 最多可同時指定 32 個軸。	○	×
DCC	變更減速時間	DCC [邏輯軸名稱 1] 減速時間 [邏輯軸名稱 2] 減速時間 [邏輯軸名稱 3] 減速時間・・・;	用來變更定位類指令的減速時間。 最多可同時指定 32 個軸。	○	×
SCC	變更 S 型時間常數	SCC [邏輯軸名稱 1]S 型時間常數 [邏輯軸名稱 2]S 型時間常數・・・;	用來設定移動平均濾波器的時間常數。 最多可同時指定 32 個軸。 濾波器同時適用於定位類指令 / 插補指令。	○	×
VEL	變更進給速度	VEL [邏輯軸名稱 1] 進給速度 [邏輯軸名稱 2] 進給速度 [邏輯軸名稱 3] 進給速度・・・;	用來設定定位指令的速度。 最多可同時指定 32 個軸。	○	×
FMX	設定插補進給最高速度	FMX T 插補進給最高速度；	用來設定插補指令的最高速度。 插補加速時間就是從0速到目前速度所需的時間，插補減速時間為目前速度回到 0 速所需的時間。	○	×
IFMX	設定軸別插補進給最高速度	IFMX [邏輯軸名稱 1] 設定軸別插補進給最高速度 [邏輯軸名稱 2] 設定軸別插補進給最高速度	為插補指令所指定的轉軸設定最高速度。 每個軸可分別設定速度限制值。	○	×
FUT	變更插補進給速度單位	FUT U 插補進給速度單位編號；	變更插補進給速度單位(FUT)可用來變更插補指令的速度單位。	○	×
IFP	設定插補進給速度比率	IFP P 插補進給速度比率；	用來設定插補指令的速度。 指定最高速度相對應的 %。	○	×
IAC	變更插補加速時間	IAC T 插補加速時間；	用來變更插補指令的加速時間。 用來指定0速到最高速度所需的加速時間。	○	×

(續下頁)

(接上頁)

指令	名稱	格式	內容	運動程式	序列程式
IDC	變更插補減速時間	IDC　T 插補減速時間；	用來變更插補指令的減速時間。 用來指定最高速度到0速所需的減速時間。	○	×
IDH	變更暫停時的插補減速時間	IDH　T 暫停時的插補減速時間；	用來變更插補指令暫停時的減速時間。 用來指定從最高速度到0速所需的減速時間。	○	×
IUT	變更插補加減速單位	IUT　U 插補加減速單位編號；	變更插補加減速單位 (IUT) 指令可用來在插補指令 (MVS、SKP、MCW/MCC) 中變更加減速度之單位。	○	×
(+、-)	插補進給速度標的軸設定功能	MVS　[+ 邏輯軸名稱 1] 指令位置 [+ 邏輯軸名稱 2] 指令位置 [- 邏輯軸名稱 3] 指令位置・・・;	指定要用來架構插補進給速度的轉軸。 只要邏輯軸名稱的起始未編寫「+」，即可指定架構插補進給速度的對象。 編寫「-」的轉軸將以插補進給速度的同步速度執行動作。	○	×
ACCMODE	設定插補加減速模式	ACCMODE　M 模式編號；	用來設定插補指令的加減速模式。 可指定連續型插補指令的後續處理動作。	○	×

絕對模式 (ABS)

絕對值模式 (ABS) 是一種將軸移動指令的座標語視為最終目標位置的指令。
下達絕對值模式 (ABS) 後，將維持 ABS 模式，直到下達增量模式 (INC) 為止。程式開始運轉後，即進入 ABS 模式。

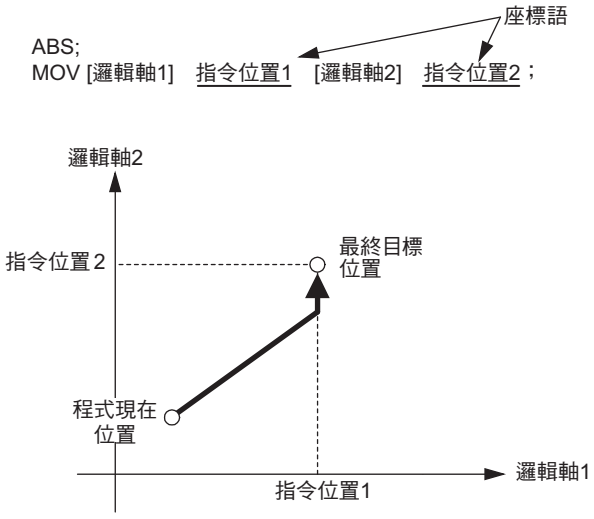


圖 6.1 絕對值模式 (ABS) 之移動模式

本手冊係以指令位置或位置指令值來表示軸移動指令的邏輯軸名稱後方的座標語。

⚠ 注意

- **若在絕對模式下對相同的座標語下指令，此時的移動動作將和增量模式下執行指令完全不同。運轉前務必確認 ABS/INC 指令是否正確執行。**
  否則恐將造成裝置的損壞。

專有名詞解說

程式現在位置
所謂程式現在位置就是軸移動指令開始執行軸動作時，工作座標上的現在位置。

格式

以下為 ABS 指令的格式。

・ 單獨指定
  ABS;

・ 將軸移動指令指定為相同的區塊
  ABS MOV [邏輯軸名稱 1] － [邏輯軸名稱 2] －；

程式範例

以下為 ABS 指令的程式範例。

```
ABS;                                  " 絕對模式
MOV [A1]10000 [B1]40000;              " 定位
MOV [A1]50000 [B1]20000;              " 定位
END;
```

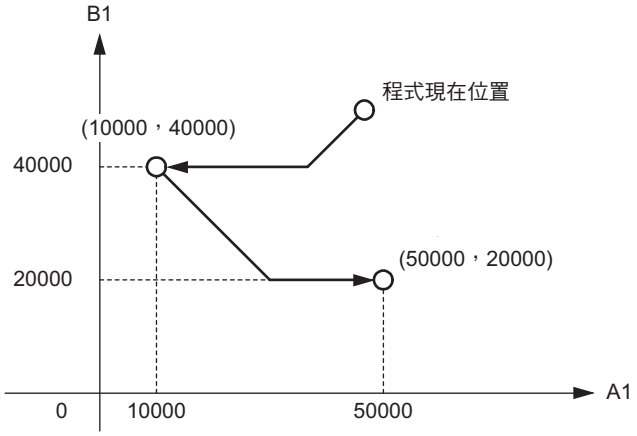


圖 6.2 ABS 指令的程式範例

ABS 指令補充事項

◆ 相關的運動參數

ABS 指令和設定參數之間並無任何相關性。
軸移動指令的移動模式 (ABS 模式 /INC 模式) 為運動程式專用的控制資料，無法利用設定參數來指定。

註記

軸移動指令的移動模式 (ABS 模式 /INC 模式) 和設定參數 OW □□□ 9 Bit 5 (位置指令型) 的設定內容各不相同，使用時需特別注意。

◆ 有限長軸和無限長軸

有限長軸和無限長軸對於座標語位置指令值的處理方式各不相同。
下表將說明有限長軸和無限長軸對於位置指令值的處理方式。

軸類型	指定軸移動指令的移動模式	位置指令值的指定方法
有限長軸	ABS 模式	在位置指令值指定最終目標位置。
	INC 模式	在位置指令值指定相對移動量。
無限長軸	ABS 模式	在 0 ~ POSMAX 的範圍內，指定位置指令值的最終目標位置。 利用位置指令值的正負號來指定移動方向。指定為正值表示朝正方向移動，指定為負值則朝負方向移動。
	INC 模式	在位置指令值指定相對移動量。

補充

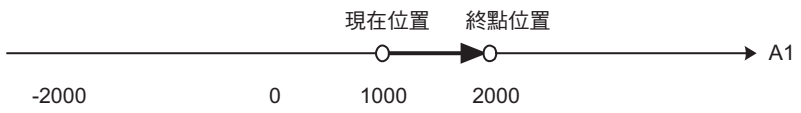
1. 利用固定參數 No. 1 (功能選擇旗標 1) 的「Bit 0：選擇軸類型」，即可設定有限長軸 / 無限長軸。設定有限長軸 / 無限長軸時必須考慮機器架構。若要瞭解如何設定適合機器的運動參數，請參閱以下手冊。
   MP3000 系列　運動控制功能　使用手冊 **(** 資料編號：**YTWNCO-14013A)**
2. 利用固定參數 No. 10 (無限長軸重置位置 (POSMAX)) 即可設定 POSMAX。

下圖為 ABS 模式下的有限長軸 / 無限長軸動作。
如欲瞭解 INC 模式下的動作，請參閱以下章節。
☞ **增量模式 (INC)(第 6-10 頁)**

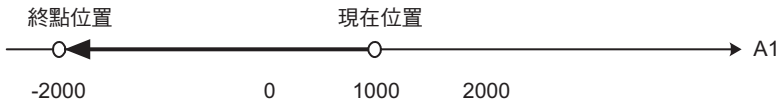
■ 指定適合有限長軸的 ABS 模式時

在位置指令值指定最終目標位置。
舉例來說，以下係為將最終目標位置從現在位置 1000 指定移動到 2000 或 -2000 時之動作。

```
ABS;
MOV [A1]2000;
```



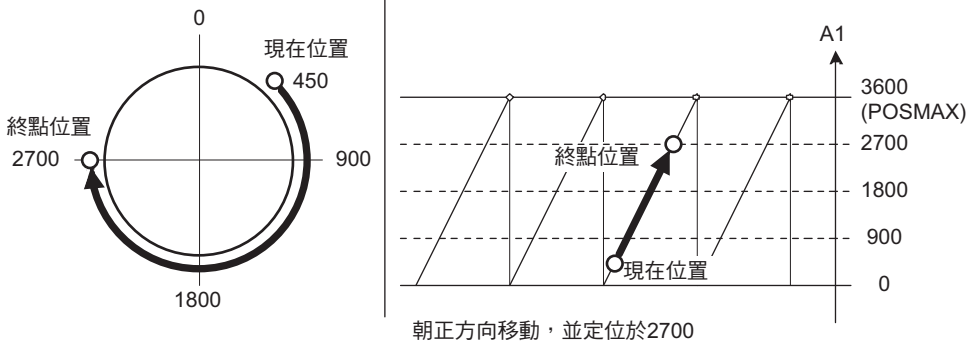

```
ABS;
MOV [A1]-2000;
```



■ 指定適合無限長軸的 ABS 模式

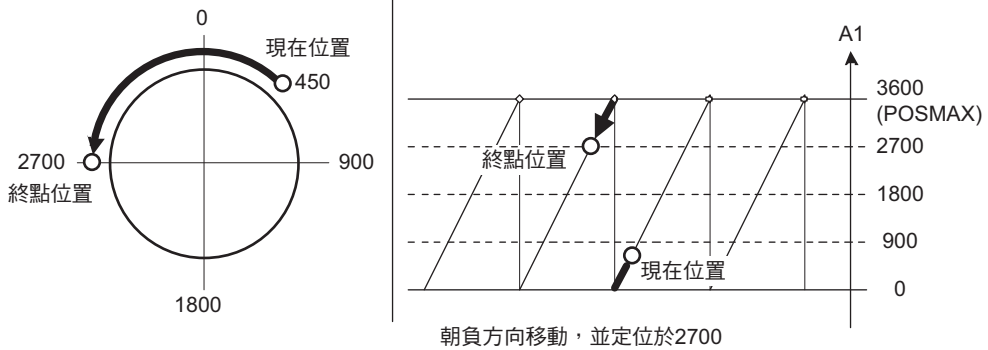
在 0 ~ POSMAX 的範圍內，指定位置指令值的最終目標位置。
利用位置指令值的正負號來指定移動方向。指定為正值表示朝正方向移動，指定為負值則朝負方向移動。
舉例來說，以下係為相對於 POSMAX = 3600 無限長軸而將最終目標位置從現在位置 450 指定移動到 2700 或 -2700 時之動作。

```
ABS;
MOV [A1]2700;
```



朝正方向移動，並定位於2700

```
ABS;
MOV [A1]-2700;
```



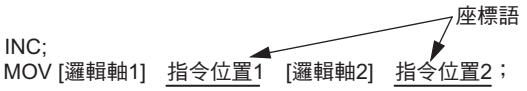
朝負方向移動，並定位於2700

註記

1. 當無限長軸被指定為 ABS 模式並移動到位置 0 時，這時候即使指定為 +0，機器也會朝負方向移動。若要朝正方向移動，則必須指定 POSMAX 值。
2. 若無限長軸被指定為 ABS 模式時的最終目標位置 (位置指令值的絕對值) 超過 POSMAX，運動程式警告就會發出。

增量模式 (INC)

增量模式 (INC) 是一種用來將軸移動的指令座標語作為相對移動量的指令。
下達增量模式 (INC) 的指令後，系統將維持 INC 模式，直到下一個絕對值模式 (ABS) 指令被下達為止。程式開始運轉後，即進入 ABS 模式。



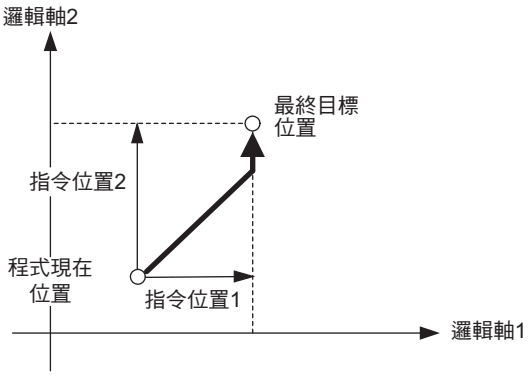


圖 6.3 增量模式 (INC) 的移動模式

本手冊係以指令位置或位置指令值來表示軸移動指令的邏輯軸名稱後方的座標語。

⚠ 注意

- **若在絕對模式下對相同的座標語下指令，此時的移動動作將和增量模式下執行指令完全不同。運轉前務必確認 ABS/INC 指令是否正確執行。**
  否則恐將造成裝置的損壞。

專有名詞解說

程式現在位置
所謂程式現在位置就是軸移動指令開始執行軸動作時，工作座標上的現在位置。

格式

以下為 INC 指令的格式。

・單獨指定
INC;

・將軸移動指令指定為相同的區塊
INC MOV [邏輯軸名稱 **1]** － **[** 邏輯軸名稱 **2]** －；

程式範例

以下為 INC 指令的程式範例。

```
INC;                              " 增量模式
MOV [A1]20000 [B1]30000;          " 定位
MOV [A1]20000 [B1]10000;          " 定位
END;
```

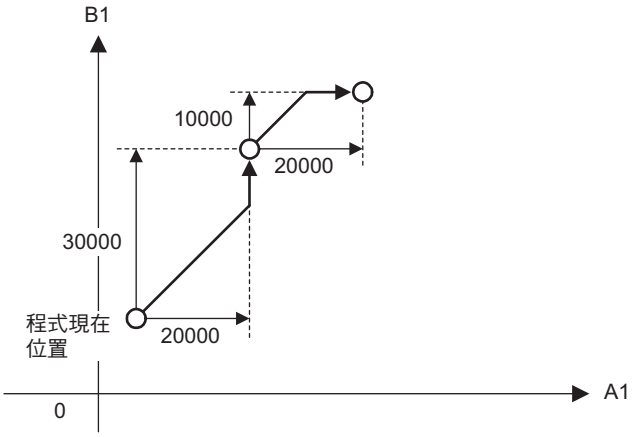


圖 6.4 INC 指令的程式範例

INC 指令補充事項

◆ 相關的運動參數

INC 指令和設定參數之間並無任何相關性。
軸移動指令的移動模式 (ABS 模式 /INC 模式) 為運動程式專用的控制資料，無法利用設定參數來指定。

◆ 有限長軸和無限長軸

有限長軸和無限長軸對於座標語位置指令值的處理方式各不相同。
下表將分別說明有限長軸和無限長軸對於位置指令值不同的處理方式。

軸類型	指定軸移動指令的移動模式	位置指令值的指定方法
有限長軸	ABS 模式	在位置指令值指定最終目標位置。
	INC 模式	在位置指令值指定相對移動量。
無限長軸	ABS 模式	在 0 ~ POSMAX 的範圍內，指定位置指令值的最終目標位置。 利用位置指令值的正負號來指定移動方向。指定為正值表示朝正方向移動，指定為負值則朝負方向移動。
	INC 模式	在位置指令值指定相對移動量。

補充

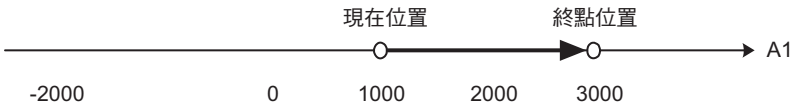
1. 利用固定參數 No. 1 (功能選擇旗標 1) 的「Bit 0：選擇軸類型」，即可設定有限長軸 / 無限長軸。設定有限長軸 / 無限長軸時必須考慮機器架構。若要瞭解如何設定適合機器的運動參數，請參閱以下手冊。
   📖 **MP3000** 系列　運動控制功能　使用手冊 **(** 資料編號：**YTWNCO-14013A)**
2. 利用固定參數 No. 10 (無限長軸重置位置 (POSMAX)) 即可設定 POSMAX。

下圖為 INC 模式下的有限長軸 / 無限長軸動作。
如欲瞭解 ABS 模式下的動作，請參閱以下章節。
☞ **絕對模式 (ABS)(第 6-6 頁)**

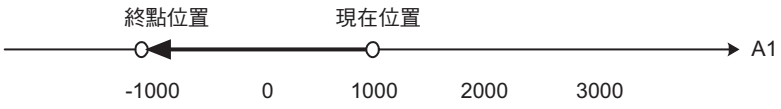
■ 指定適合有限長軸的 INC 模式

在位置指令值指定相對移動量。
舉例來說，以下係為將最終目標位置從現在位置 1000 指定移動到 2000 或 -2000 時之動作。

```
INC;
MOV [A1]2000;
```




```
INC;
MOV [A1]-2000;
```

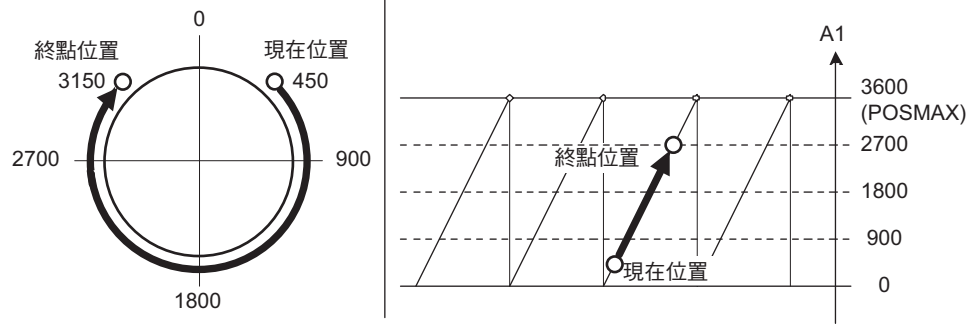


■ 指定適合無限長軸的 INC 模式

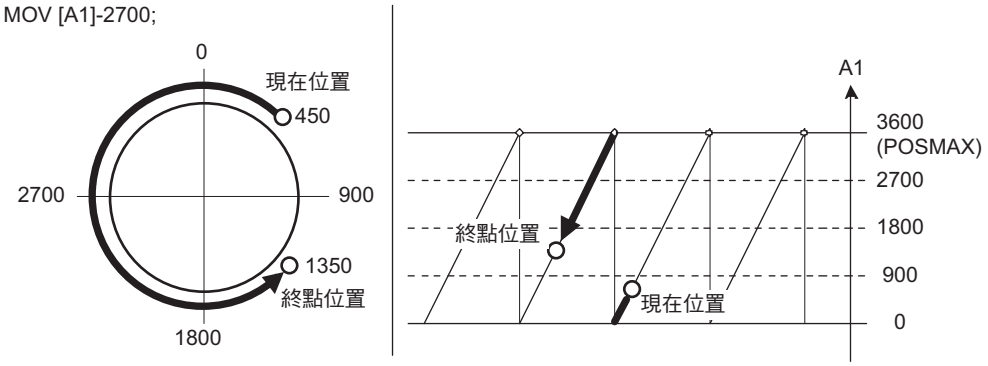
在位置指令值指定相對移動量。

舉例來說，以下係為相對於 POSMAX = 3600 無限長軸而將最終目標位置從現在位置 450 指定移動到 2700 或 -2700 時之動作。

```
INC;
MOV [A1]2700;
```

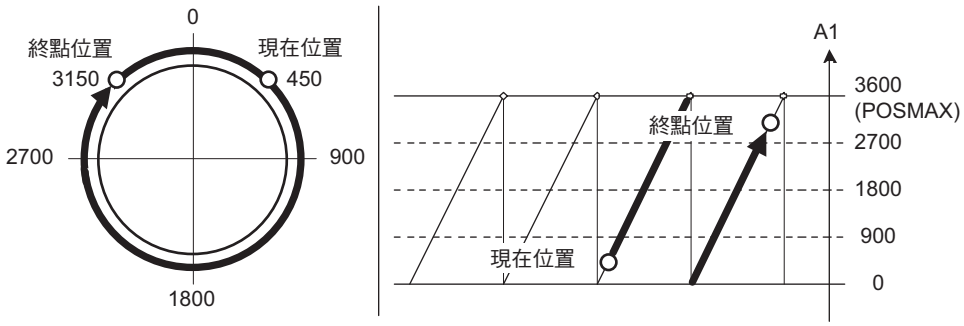



```
INC;
MOV [A1]-2700;
```



補充　當位置指令值 (座標語) 的絕對值超過 POSMAX 時，INC 模式下的位置指令值 (座標語) 就會被當作相對移動量來移動轉軸。

```
INC;
MOV [A1]6300;          "|6300|>3600(POSMAX)
```



變更加速時間 (ACC)

變更加速時間 (ACC) 是一種針對以下的軸移動指令來變更各轉軸加速時間或加速度的指令。

・定位 (MOV)

・指定時間定位 (MVT)

・外部定位 (EXM)

最多可同時變更 32 個軸。省略指令的轉軸其加速時間將維持不變。

變更後的軸加速時間將維持不變，直到變更加速時間 (ACC) 指令被重新設定為止。

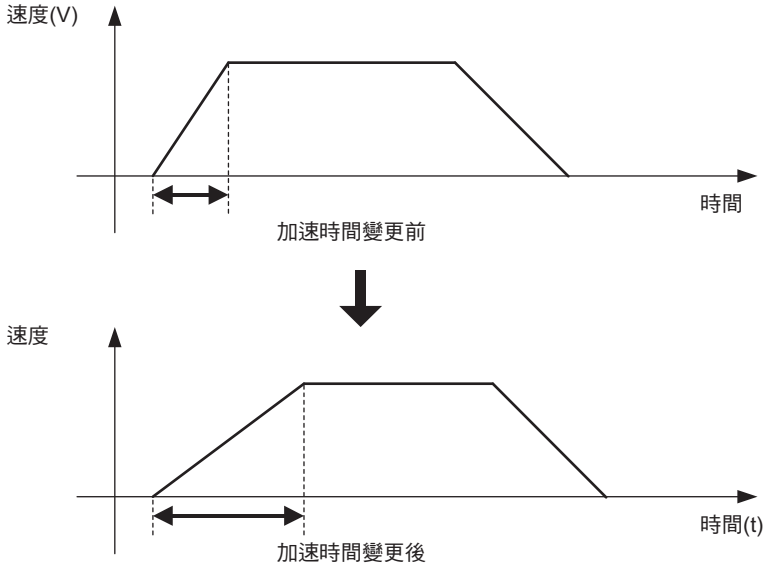


圖 6.5 變更加速時間

補充

1. 變更加速時間 (ACC) 是一種用來設定定位指令 (MOV、EXM、MVT) 加速時間的指令。利用 IAC 指令即可設定插補指令 (MVS、MCW、MCC、SKP) 的加速時間。
2. ACC/DCC/SCC 指令適用於所有的運動控制功能。

格式

以下為 ACC 指令的格式。

ACC [邏輯軸名稱 1] 加速時間 [邏輯軸名稱 2] 加速時間 [邏輯軸名稱 3] 加速時間・・・;

項目	單位	適用的資料
加速時間或加速度	ms 或指令單位 /s^2	・利用立即值直接指定 ・利用長整數型暫存器間接指定

(註) 利用設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 即可設定。

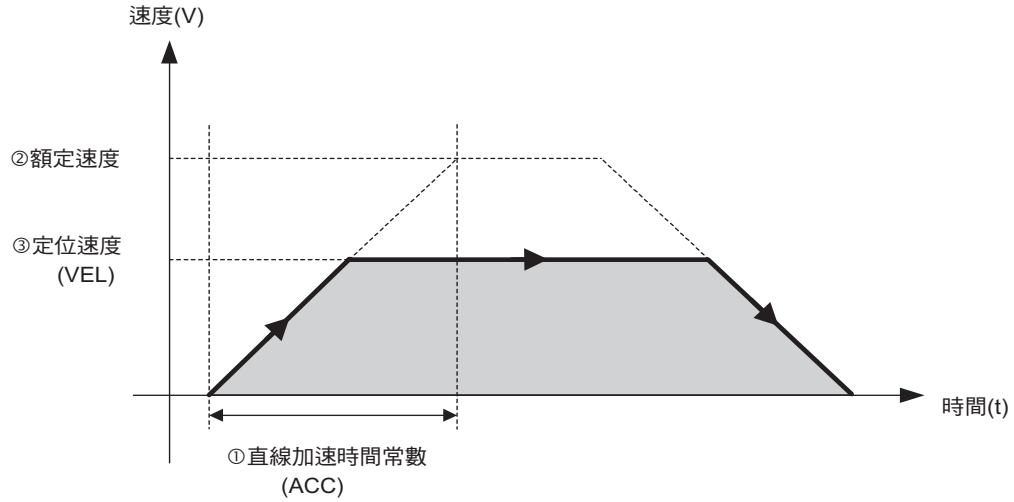
ACC 指令的設定項目

接下來將說明 ACC 指令的設定項目。
您可選擇加速時間 (ms) 或加速度 (指令單位 /s^2) 作為 ACC 指令設定值的單位。

利用設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 即可設定您所要使用的單位。

參數名稱	加減速的速度單位
功能設定 1 選擇加減速的速度單位	0：指令單位 /s^2 1：ms (初始值)

· 當設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 被設定為 「1：ms」時



① 加速時間

ACC 指令的設定值為加速時間 (從速度 0 到達額定速度所需的時間)。
指令範圍為 1 ~ 32767 ms。

② 額定速度

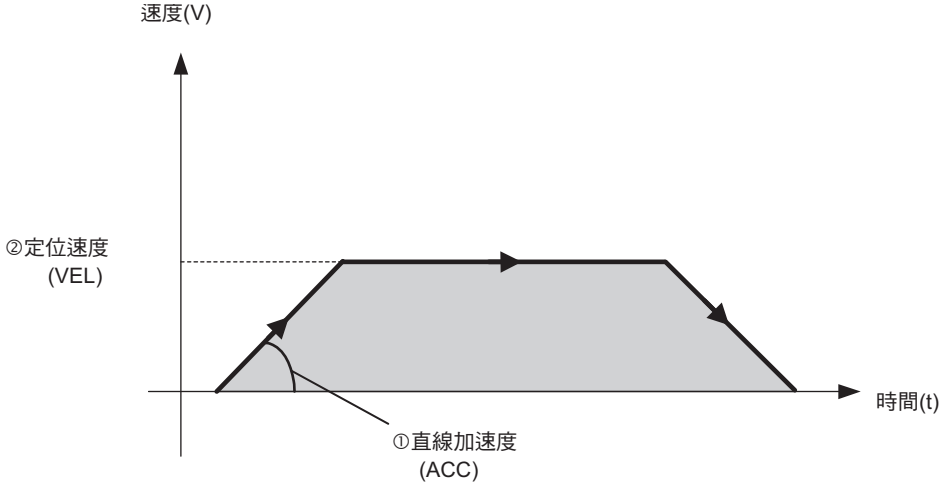
利用固定參數 No. 34 (額定轉數)，即可設定每個軸的額定速度。

③ 定位速度

亦即定位類指令 (MOV、MVT、EXM) 的速度。
利用變更進給速度 (VEL) 指令，即可設定每個軸的定位速度。

補充　指定時間定位 (MVT) 時，定位速度並非 VEL 指令的指令值。
指定時間定位 (MVT) 係根據您所設定的定位時間及移動量來改變定位速度。

・當設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 被設定為「0：指令單位 /s^2」時



① 直線加速度

ACC 指令的設定值為「直線加速度」。
指令範圍為 1 ~ 2^{31} - 1 (指令單位 /s^2)。

② 定位速度

亦即定位類指令 (MOV、MVT、EXM) 的速度。
利用變更進給速度 (VEL)，即可設定每個軸的定位速度。

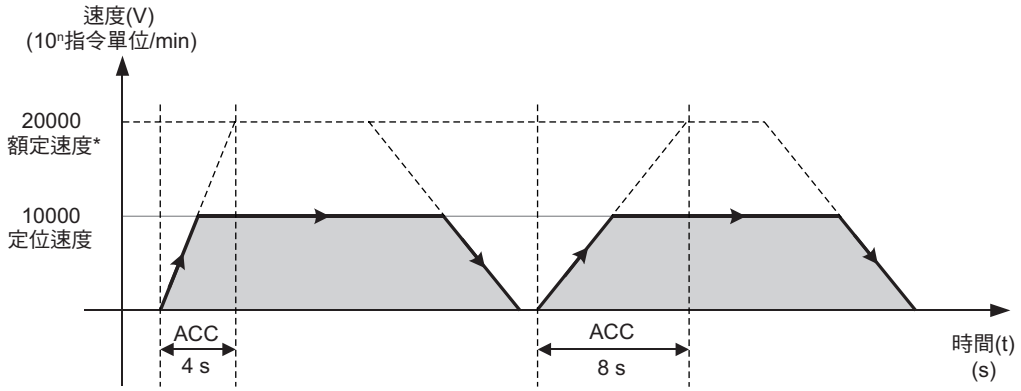
補充　指定時間定位 (MVT) 時，定位速度並非 VEL 指令的指令值。
指定時間定位 (MVT) 係根據您所設定的定位時間及移動量來改變定位速度。

程式範例

以下為 ACC 指令的程式範例。

・ **當設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 被設定為「1：ms」時**
以下範例係執行 4s 內從速度 0 加速到額定速度的 MOV 指令，以及執行 8s 內加速的 MOV 指令。

```
INC;                    " 增量模式
VEL [A1]10000;          " 變更進給速度 [10**n 指令單位 /min]
DCC [A1]8000;           " 變更減速時間 [ms]
ACC [A1]4000;           " 變更加速時間 [ms]
MOV [A1]5000000;        " 定位
DL00000 = 8000;         " 加速時間 [ms]
ACC [A1]DL00000;        " 變更加速時間 [ms]
MOV [A1]5000000;        " 定位
END;
```



* 額定速度的單位必須和定位速度轉換為相同的單位 (10^n 指令單位 /min)，而非 min-1。

圖 6.6 ACC 指令的程式範例 1

・當設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 被設定為 「0：指令單位 /s^2」時

以下範例係執行以加速度 60.000 (mm/s^2) 加速的 MOV 指令以及以加速度 100.000 (mm/s^2) 加速的 MOV 指令。但每個指令單位為 0.001 mm。

```
INC;                        " 增量模式
VEL [A1]18000;              " 變更進給速度 [10**n 指令單位 /min]
DCC [A1]100000;             " 變更減速時間 [ 指令單位 /(S*S)]
ACC [A1]60000;              " 變更加速度 [ 指令單位 /(S*S)]
MOV [A1]5000000;            " 定位
DL00000 = 100000;           " 加速度 [ 指令單位 /(S*S)]
ACC [A1]DL00000;            " 變更加速度 [ 指令單位 /(S*S)]
MOV [A1]5000000;            " 定位
END;
```

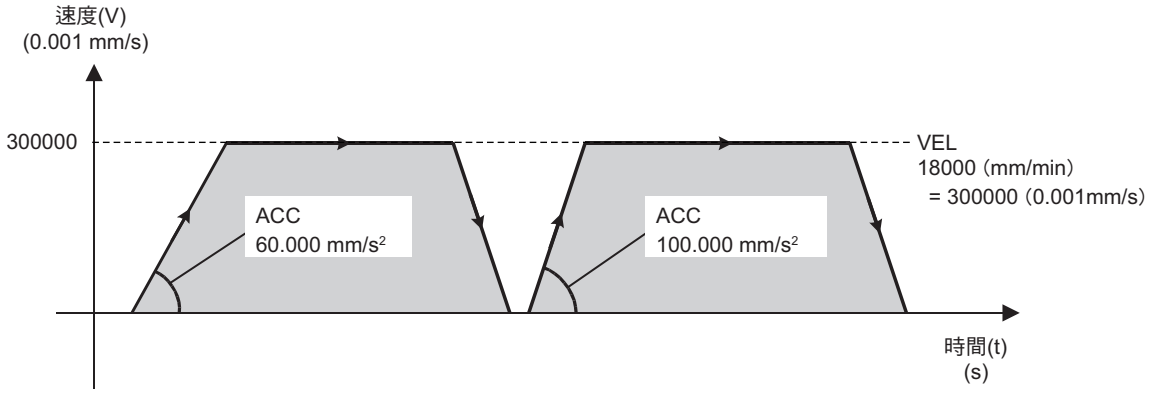


圖 6.7 ACC 指令的程式範例 2

ACC 指令補充事項

◆ 相關的運動參數

ACC 指令可用來變更設定參數的加速時間。

參數名稱	暫存器編號	內容
直線加速度 / 加速時間常數	OL□□□36	設定直線加速度或直線加速時間常數。

除了 ACC 指令外，即使直接變更設定參數 OL□□□36 (直線加速度 / 加速時間常數)，也可變更加速時間。直接變更加速時間的步驟，請參閱下表。

運動控制功能	規格	設定步驟
SVR，SVR32，SVA-01，PO-01	轉軸將依照設定參數 OL□□□36 (直線加速度 / 加速時間常數) 的加速時間來執行動作。	在設定參數 OL□□□36 (直線加速度 / 加速時間常數) 中設定加速時間。
SVC，SVC32，SVC-01，SVB-01	轉軸將依照伺服驅動器的使用者常數的加速度來執行動作。	在設定參數 OL□□□36 (直線加速度 / 加速時間常數) 中設定加速時間。若要讓加速時間反映在伺服驅動器上，請利用設定參數 OW□□□08 (運動指令) 來執行 「10：變更加速時間常數」。

(註) SVC、SVC32、SVC-01、SVB-01 內置可將設定參數 OL□□□36 (直線加速度 / 加速時間常數) 自動反映到伺服驅動器使用者常數的加速度的功能。當自動反映功能開啟後，就不需要再利用設定參數 OW□□□08 (運動指令) 來執行 「10：變更直線加速時間常數」。
如欲使用自動反映功能，請參閱以下手冊。

🕮 **MP3000** 系列　運動控制功能　使用手冊 **(** 資料編號：**YTWNCO-14013A)**

◆ 加速時間和減速時間

若同時使用下表所示的運動控制功能及伺服驅動器時，則無法再單獨設定加速時間和減速時間。您只要設定好加速時間，即同時完成加速時間和減速時間的設定。SGD-N 和 SGDB-N 以外的伺服驅動器只要使用 ACC 指令和 DCC 指令，即可單獨設定加速時間和減速時間。

運動控制功能	伺服驅動器	備註
SVB-01	SGD-N	・SVB-01 的情況，將依照伺服驅動器使用者常數的加減速速度來執行轉軸動作。 ・SGD-N、SGDB-N 伺服驅動器可共用加速時間 / 減速時間的使用者常數。
	SGDB-N	

變更減速時間 (DCC)

變更減速時間 (DCC) 是一種針對以下的軸移動指令來變更各轉軸減速時間或減速度的指令。

・定位 (MOV)

・指定時間定位 (MVT)

・外部定位 (EXM)

最多可同時變更 32 個軸。省略指令的轉軸其減速時間將維持不變。

變更後的軸減速時間將維持不變，直到變更減速時間 (DCC) 指令被重新設定為止。

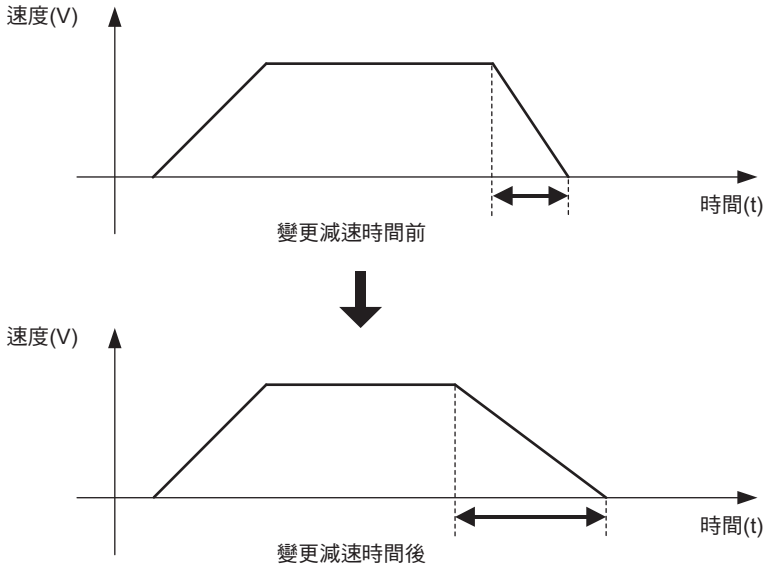


圖 6.8 變更減速時間

補充

1. 變更減速時間 (DCC) 是一種用來設定定位指令 (MOV、EXM、MVT) 減速時間的指令。利用 IDC 指令即可設定插補指令 (MVS、MCW、MCC、SKP) 的減速時間。
2. ACC/DCC/SCC 指令適用於所有的運動控制功能。

格式

以下為 DCC 指令的格式。

DCC [邏輯軸名稱 1] 減速時間 [邏輯軸名稱 2] 減速時間 [邏輯軸名稱 3] 減速時間・・・;

項目	單位	適用的資料
減速時間或 減速速度	ms 或指令單位 /s^2	・利用立即值直接指定 ・利用長整數型暫存器間接指定

(註) 利用設定參數 OW□□□03 Bit 4 ~ 7 即可設定單位。

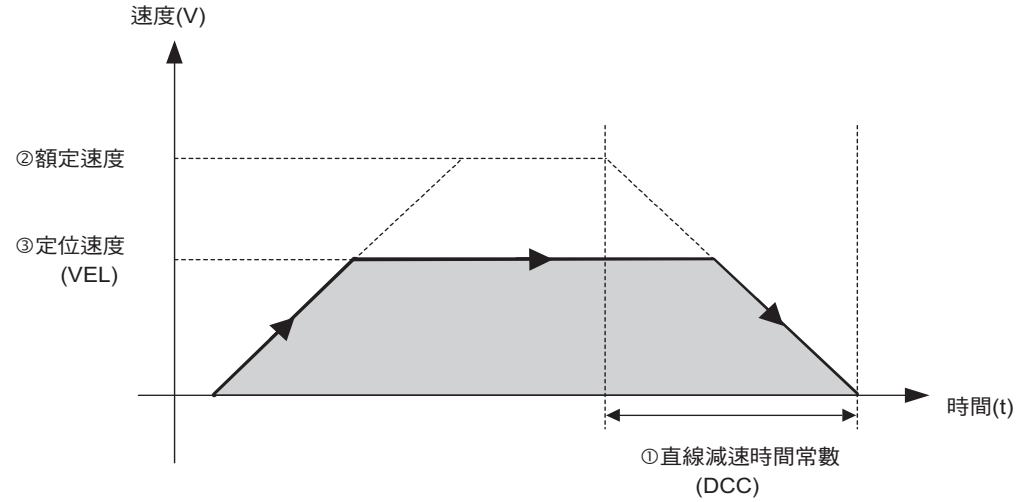
DCC 指令的設定項目

接下來將說明 DCC 指令的設定項目。
您可選擇減速時間 (ms) 或減速度 (指令單位 /s^2) 作為 DCC 指令設定值的單位。

利用設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 即可設定您所要使用的單位。

參數名稱	加減速的速度單位
功能設定 1 選擇加減速的速度單位	0：指令單位 /s^2 1：ms (初始值)

· 當設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 被設定為 「1：ms」時



① 直線減速時間常數

DCC 指令的設定值即為直線減速時間常數 (從額定速度直線減速到速度 0 所需的時間)。
指令範圍為 1 ~ 32767 ms。

② 額定速度

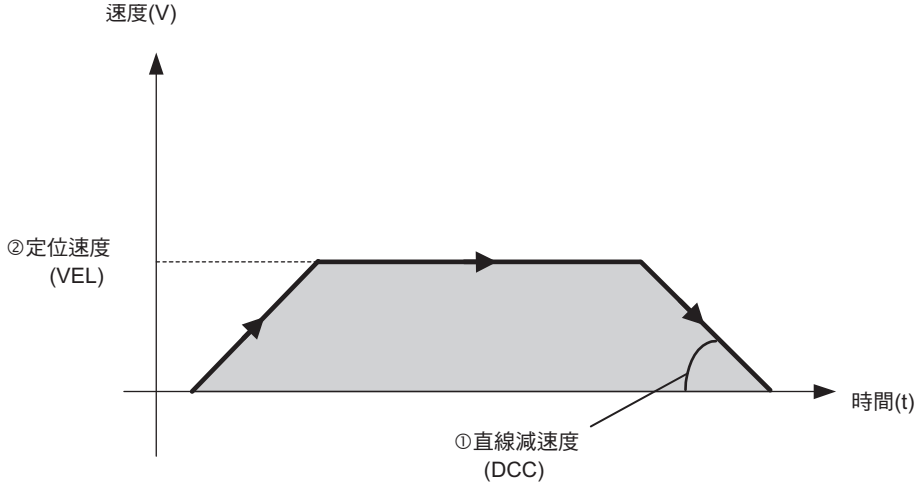
利用固定參數 No. 34 (額定轉數)，即可設定每個軸的額定速度。

③ 定位速度

亦即定位類指令 (MOV、MVT、EXM) 的速度。
利用變更進給速度 (VEL)，即可設定每個軸的定位速度。

補充　指定時間定位 (MVT) 時，定位速度並非 VEL 指令的指令值。
指定時間定位 (MVT) 係根據您所設定的定位時間及移動量來改變定位速度。

・當設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 被設定為「0：指令單位 /s^2」時



① 直線減速度

DCC 指令的設定值為直線減速度。
指令範圍為 1 ~ 2^{31} - 1 (指令單位 /s^2)。

② 定位速度

亦即定位類指令 (MOV、MVT、EXM) 的速度。
利用變更進給速度 (VEL)，即可設定每個軸的定位速度。

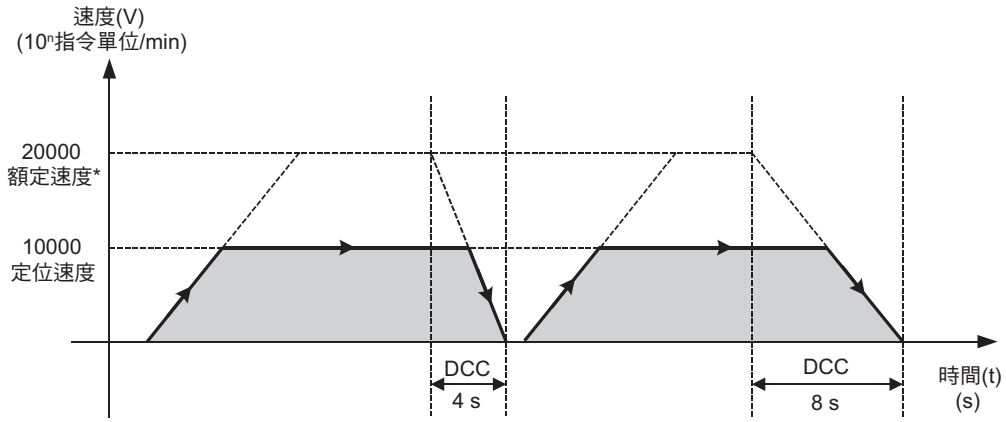
補充　指定時間定位 (MVT) 時，定位速度並非 VEL 指令的指令值。
指定時間定位 (MVT) 係根據您所設定的定位時間及移動量來改變定位速度。

程式範例

以下為 DCC 指令的程式範例。

・ **當設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 被設定為 「1：ms」時**
以下係在 4s 內從額定速度減速到速度 0 的 MOV 指令，以及在 8s 內減速的 MOV 指令之執行範例。

```
INC;                          " 增量模式
VEL [A1]10000;                " 變更進給速度 [10**n 指令單位 /min]
ACC [A1]8000;                 " 變更加速時間 [ms]
DCC [A1]4000;                 " 變更減速時間 [ms]
MOV [A1]5000000;              " 定位
DL00000 = 8000;               " 減速時間 [ms]
DCC [A1]DL00000;              " 變更減速時間 [ms]
MOV [A1]5000000;              " 定位
END;
```



* 額定速度的單位必須和定位速度轉換為相同的單位 (10^n 指令單位 /min)，而非 min^{-1}。

圖 6.9 DCC 指令的程式範例 1

· **當設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 被設定為 「0：指令單位 /s^2」時**
以下係以減速度 60.000 (mm/s^2) 減速的 MOV 指令，以及以減速度 100.000 (mm/s^2) 減速的 MOV 指令之執行範例。但每個指令單位為 0.001 (mm)。

```
INC;                        " 增量模式
VEL [A1]18000;              " 變更進給速度 [10**n 指令單位 /min]
ACC [A1]100000;             " 變更加速度 [ 指令單位 /(S*S)]
DCC [A1]60000;              " 變更減速度 [ 指令單位 /(S*S)]
MOV [A1]5000000;            " 定位
DL00000 = 100000;           " 減速度 [ 指令單位 /(S*S)]
DCC [A1] DL00000;           " 變更減速度 [ 指令單位 /(S*S)]
MOV [A1]5000000;            " 定位
END;
```

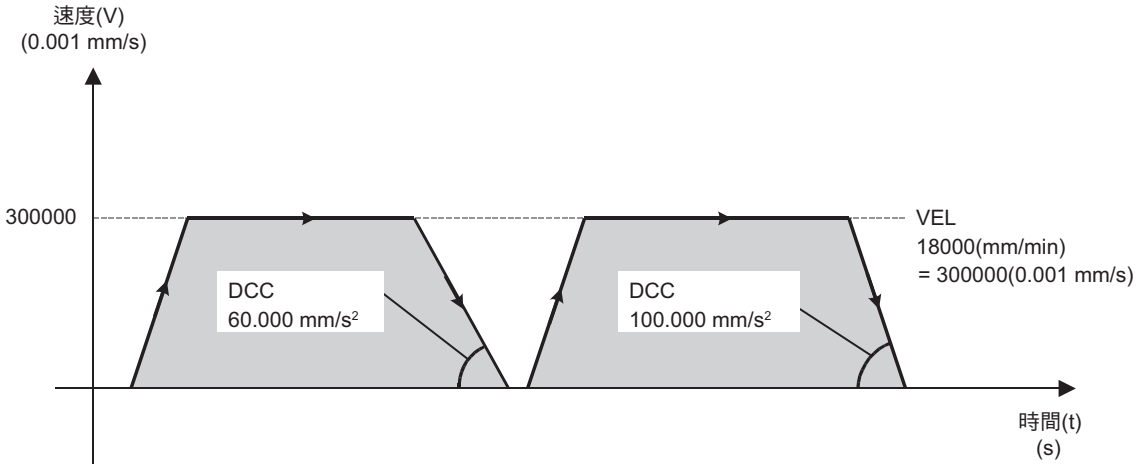


圖 6.10 DCC 指令的程式範例 2

DCC 指令補充事項

◆ 相關的運動參數

DCC 指令可用來變更設定參數的減速時間。

參數名稱	暫存器編號	說明
直線減速度 / 減速時間常數	OL□□□38	設定直線減速度或直線減速時間常數。

除了 DCC 指令外，即使直接變更設定參數 OL□□□38 (直線減速度 / 減速時間常數)，也可變更減速時間。直接變更減速時間的步驟，請參閱下表。

運動控制功能	規格	設定步驟
SVR, SVR32, SVA-01, PO-01	轉軸將依照設定參數 OL□□□38 (直線減速度 / 減速時間常數) 的減速時間來執行動作。	在設定參數 OL□□□38 (直線減速度 / 減速時間常數) 中設定減速時間。
SVC, SVC32, SVC-01, SVB-01	轉軸將依照伺服驅動器的使用者常數的減速度來執行動作。	在設定參數 OL□□□38 (直線減速度 / 減速時間常數) 中設定減速時間。若要讓減速時間反映在伺服驅動器上，請利用設定參數 OW□□□08 (運動指令) 來執行「11：變更減速時間常數」。

(註) SVC、SVC32、SVC-01 和 SVB-01 內置了可將設定參數 OL□□□38 (直線減速度 / 減速時間常數) 自動反映到伺服驅動器使用者常數的減速度的功能。當自動反映功能開啟後，就不需要再利用設定參數 OW□□□08 (運動指令) 來執行「11：變更直線減速時間常數」。
如欲使用自動反映功能，請參閱以下手冊。

📖 **MP3000 系列　運動控制功能　使用手冊 (資料編號：YTWNCO-14013A)**

◆ 加速時間和減速時間

若同時使用下表所示的運動控制功能及伺服驅動器時，則無法再單獨設定加速時間和減速時間。您只要設定好加速時間，即同時完成加速時間和減速時間的設定。SGD-N 和 SGDB-N 以外的伺服驅動器只要使用 ACC 指令和 DCC 指令，即可單獨設定加速時間和減速時間。

運動控制功能	伺服驅動器	備註
SVB-01	SGD-N	・SVB-01 將依照伺服驅動器使用者常數的加減速速度來執行轉軸動作。 ・SGD-N、SGDB-N 伺服驅動器可共用加速時間 / 減速時間的使用者常數。
	SGDB-N	

變更 S 型時間常數 (SCC)

變更 S 型時間常數 (SCC) 就是一種可用來變更軸移動指令中各轉軸 S 型時間常數的指令。
S 型時間常數係指可用來抑制加減速時機械系統震動的 S 型加減速功能參數。
最多可同時變更 32 個軸。省略指令的該轉軸 S 型時間常數將維持不變。
變更後的軸 S 型時間常數將維持不變，直到 變更 S 型時間常數 (SCC) 被重新設定為止。

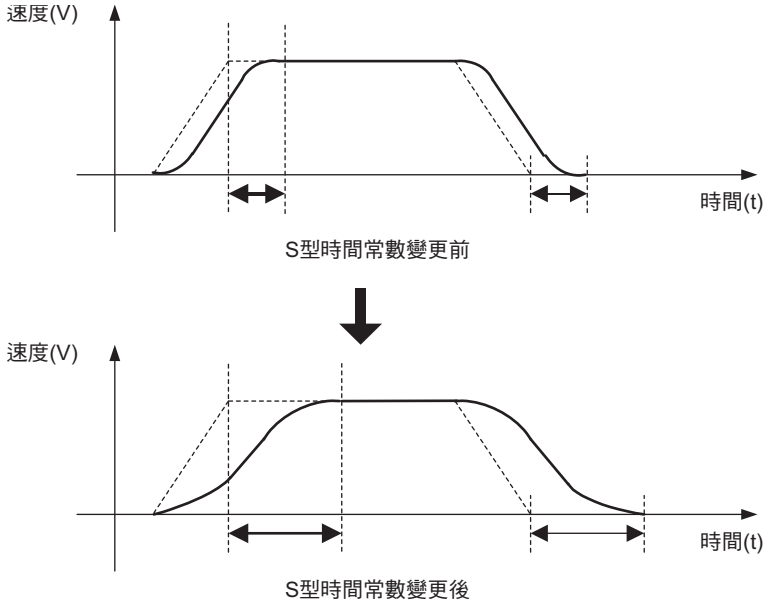


圖 6.11 變更 S 型時間常數

格式

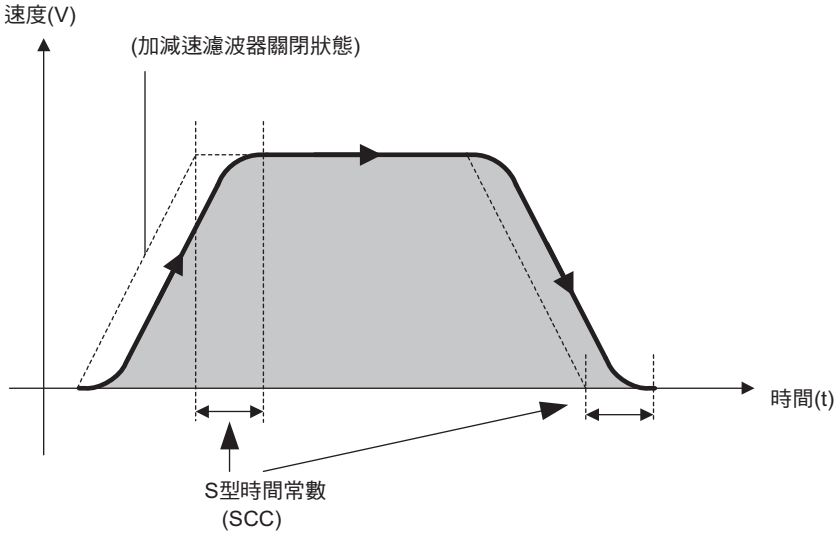
以下為 SCC 指令的格式。

SCC [邏輯軸名稱 1] S 型時間常數 [邏輯軸名稱 2] S 型時間常數・・・;

項目	單位	適用的資料
S 型時間常數	ms	・利用立即值直接指定 ・利用長整數型暫存器間接指定

SCC 指令的設定項目

接下來將說明 SCC 指令的設定項目。



利用 SCC 指令的數值或暫存器，即可對各轉軸的 S 型時間常數下達指令。
S 型時間常數的指令範圍依下表所使用的運動控制功能而異。

・SVR、SVR32、PO-01 和 SVA-01 不得超過設定參數 OW□□□3A (濾波器時間常數) 的指令範圍。

・SVC、SVC32、SVC-01 和 SVB-01 不得超過伺服驅動器使用者常數的平均移動時間的指令範圍。

下表為 S 型時間常數的指令範圍。

運動控制功能	SCC 指令的指令範圍 [ms]	備註
SVA-01	0 ～ 6553	–
SVC、SVC32、SVC-01、SVB-01	0 ～ 510	使用 SGD-N、SGDB-N、SGDH+NS110/NS115、SGDS、SGDX、SGDV 伺服驅動器
	–	使用 SGDJ 伺服驅動器時，由於伺服驅動器並無 「平均移動時間」這項使用者常數，因此無法使用 S 型加減速功能。
PO-01	0 ～ 6553	–
SVR、SVR32	0 ～ 6553	–

註記

1. 無論運動控制功能為何，只要您所下達的指令超過 6553 ms，運動程式警告就會發生。
2. 一旦 SVC、SVC32、SVC-01 和 SVB-01 所下達的指令超過上限值 (511 ~ 6553 ms)，就會出現監控參數 IL□□□02 Bit 1 = 1 (設定參數異常)，此時伺服驅動器使用者常數的平均移動時間就會被設定上限值 (510 ms)。

程式範例

以下為 SCC 指令的程式範例。
以下為 S 型時間常數 250 (ms)MOV 指令和 S 型時間常數 500 (ms)MOV 指令之執行範例。
不過，前提必須為設定參數採用以下設定。

・ 在設定參數 OW□□□03 Bit 0 ~ 3 (選擇速度單位) 中設定 「0: 指令單位 /s」

・ 在設定參數 OW□□□03 Bit 4 ~ 7 (選擇加減速的速度單位) 中設定 「0：指令單位 /s^2」

```
INC;                        " 增量模式
VEL [A1]10000;              " 變更進給速度 [ 指令單位 /S]
ACC [A1]20000;              " 變更加速度 [ 指令單位 /S * S]
DCC [A1]20000;              " 變更減速度 [ 指令單位 /S * S]
SCC [A1]250;                " 變更 S 型時間常數 [ms]
MOV [A1]20000;              " 定位
DL00000 = 500;              "S 型時間常數 [ms]
SCC [A1]DL00000;            " 變更 S 型時間常數 [ms]
MOV [A1]20000;              " 定位
END;
```

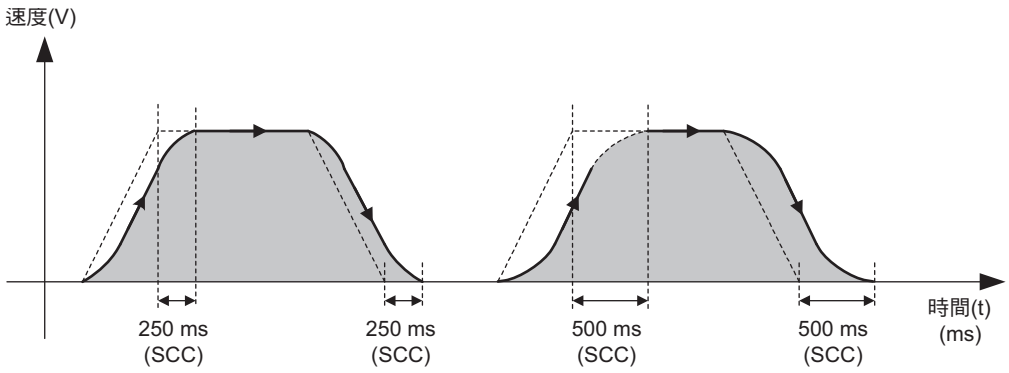


圖 6.12 SCC 指令的程式範例

SCC 指令補充事項

◆ 相關的運動參數

SCC 指令可用來變更設定參數中的 S 型常數。

參數名稱	暫存器編號	說明
濾波器時間常數	OW□□□3A	設定加減速濾波器的時間常數 (1 = 0.1 ms)。 ・請確認目前為「送出指令完成狀態 (IW□□□0C Bit 0 = 1)」後，再變更濾波器時間常數。 ・利用選擇濾波器類型 (OW□□□03 Bit 8 ~ B) 來選擇您所要使用的濾波器類型，即可變更時間常數。

除了 SCC 指令外，您還可以直接變更設定參數 OW□□□3A (濾波器時間常數)，來變更 S 型時間常數。下表為直接變更 S 型時間常數的步驟。

運動控制功能	規格	設定步驟
SVR, SVR32, SVA-01, PO-01	將 S 型加減速設定為開啟後，系統就會依照設定參數 OW□□□3A (濾波器時間常數) 中的 S 型時間常數來執行轉軸動作。	請在設定參數 OW□□□3A (濾波器時間常數) 中設定 S 型時間常數。
SVC, SVC32, SVC-01, SVB-01	將 S 型加減速設定為開啟後，就會依照伺服驅動器使用者常數的平均移動濾波器時間常數，來執行轉軸動作。	請在設定參數 OW□□□3A (濾波器時間常數) 中設定 S 型時間常數。接著，請利用設定參數 OW□□□08 (運動指令) 來執行 「12：變更濾波器時間常數」，如此就能將 S 型時間常數反映在伺服驅動器上。(*)

* SVC、SVC32、SVC-01 和 SVB-01 內置了可將設定參數 OW□□□3A (濾波器時間常數) 自動反映在伺服驅動器使用者常數的平均移動濾波器時間常數的功能。
當自動反映功能開啟後，就不需要再利用設定參數 OW□□□08 (運動指令) 來執行 「12：變更濾波器時間常數」。
如欲使用自動反映功能，請參閱以下手冊。
MP3000 系列　運動控制功能　使用手冊 (資料編號：YTWNCO-14013A)

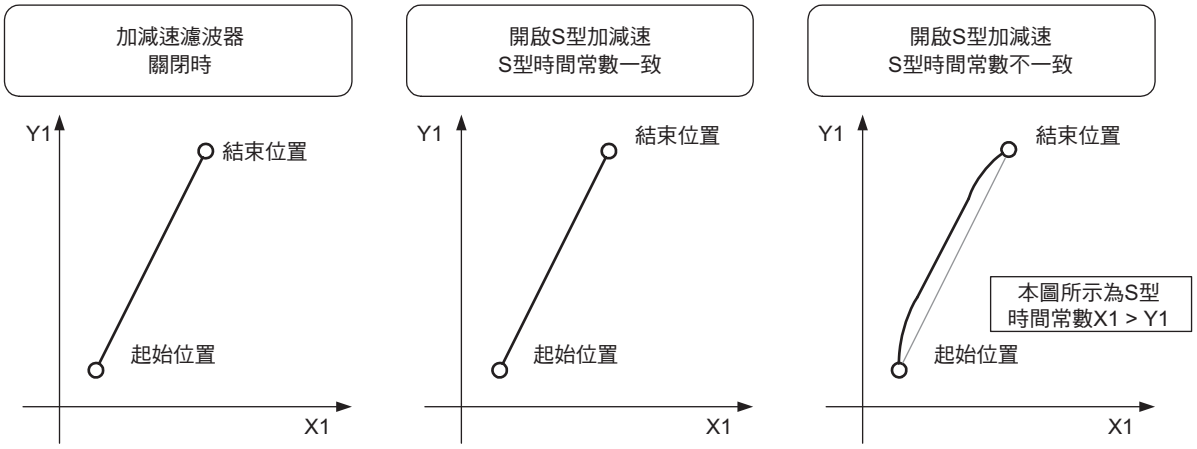
◆ 插補移動軌跡及 S 型加減速

S 型加減速會影響插補指令 (MVS、MCW、MCC、SKP) 的移動軌跡。

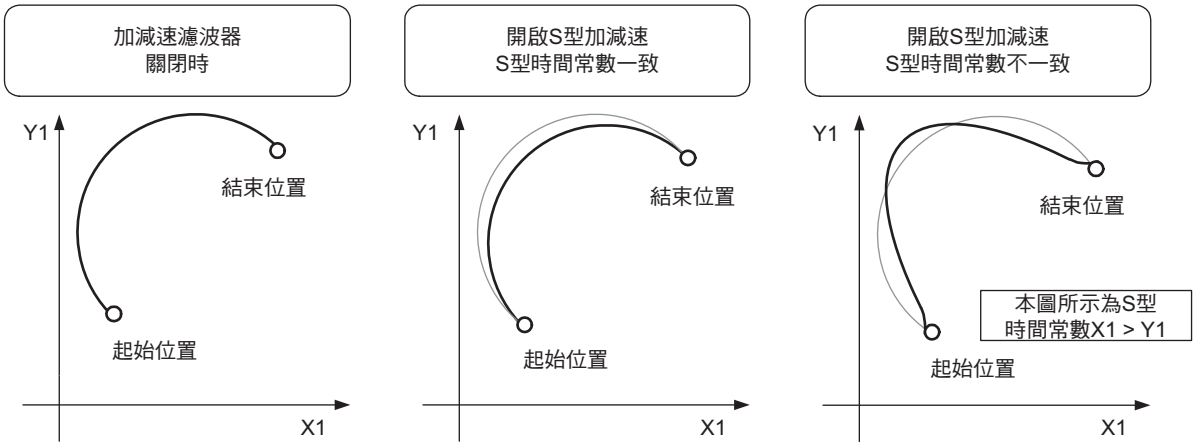
・若希望利用線性內插，讓移動軌跡和 S 型加減速關閉狀態時一樣，插補的對象軸的 S 型時間常數必須一致。

・利用循環內插來開啟 S 型加減速時，移動軌跡將和 S 型加減速被關閉時不同。

< 線性內插的移動軌跡 >



< 循環內插的移動軌跡 >



◆ 選擇濾波器類型

將 S 型加減速設定為開啟前，請先利用設定參數 OW□□□03 Bit 8 ~ B (選擇濾波器類型)，將各轉軸濾波器類型設定為 「2：平均移動濾波器」。

參數名稱	暫存器編號	濾波器類型
功能設定 1 選擇濾波器類型	OW□□□03 Bit 8 ～ B	0：無濾波器 (初始值) 1：指數函數加減速濾波器 2：平均移動濾波器

若要使用 SVC、SVC32、SVC-01 或 SVB-01，並將自動寫入伺服驅動器設定為關閉時，請利用設定參數 OW□□□08 (運動指令) 來執行 「13：變更濾波器類型」，以便反映在伺服驅動器上。
以下程式範例係利用運動程式來變更濾波器類型。

```
:
:

" 確認目前狀態是否適合更新濾波器類型
IOW IW8008 = = 0;                    " 等待無運動指令狀態
IOW IB800C0 = = 1;                   " 等待送出指令完成狀態

" 選擇平均移動濾波器作為濾波器類型
DW00000 = OW8003 & F0FFH;            " 儲存 「選擇濾波器類型」以外的資訊
OW8003  = DW00000 | 0200H;           " 濾波器類型 = 平均移動濾波器

" 利用內置的 SVB/SVB-01 模組將濾波器類型反映在伺服驅動器上
OW8008 = 13;                         " 要求 「變更濾波器類型」
IOW IW8008 = = 13;                   " 等待進入 「變更濾波器類型」的處理狀態
IOW IB80098 = = 1;                   " 等待運動指令執行完成
OW8008 = 0；                         " 清除要求
IOW IW8008 = = 0;                    " 等待無運動指令狀態

:
:
```

補充　使用 SVR、SVR32、PO-01 或 SVA-01 時，不需要編寫上述程式。
或是，即使使用 SVC、SVC32、SVC-01 或 SVB-01，只要將自動寫入伺服驅動器功能設定為啟動，即不需要上述程式。

補充　如需利用 SVC、SVC32、SVC-01 或 SVB-01 對伺服驅動器執行 「自動寫入」功能時，請參閱下述手冊。

- 📖 **MP2000 系列運動模組 SVB/SVB-01 內置型　使用手冊 (資料編號：SIJPC880700 33)**
- 📖 **MP2000 系列運動模組 SVC/SVC-01 內置型　使用手冊 (資料編號：SIJPC880700 41)**
- 📖 **MP3000 系列　運動控制功能　使用手冊 (資料編號：YTWNCO-14013A)**

變更進給速度 (VEL)

變更進給速度 (VEL) 是一種針對以下的軸移動指令變更各軸進給速度的指令。

・定位 (MOV)

・外部定位 (EXM)

本手冊將上述的軸移動指令和指定時間定位 (MVT) 加以整合成為定位類指令，因此進給速度就稱為定位速度。

最多可同時變更 32 個軸。省略指令的轉軸其定位速度不變。

變更後的軸定位速度將維持不變，直到變更進給速度 (VEL) 指令被重新設定，或是指定時間定位 (MVT) 的速度被變更為止。

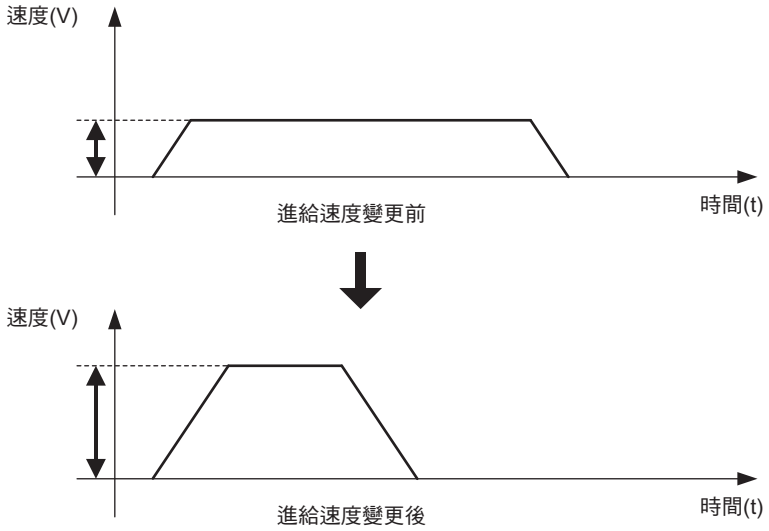


圖 6.13　變更進給速度

補充　變更進給速度 (VEL) 是一種用來設定定位類指令 (MOV、EXM) 定位速度的指令。利用 F 指令設定或 IFP 指令，即可設定插補指令 (MVS、MCW、MCC、SKP) 的進給速度。

格式

以下為 VEL 指令的格式。

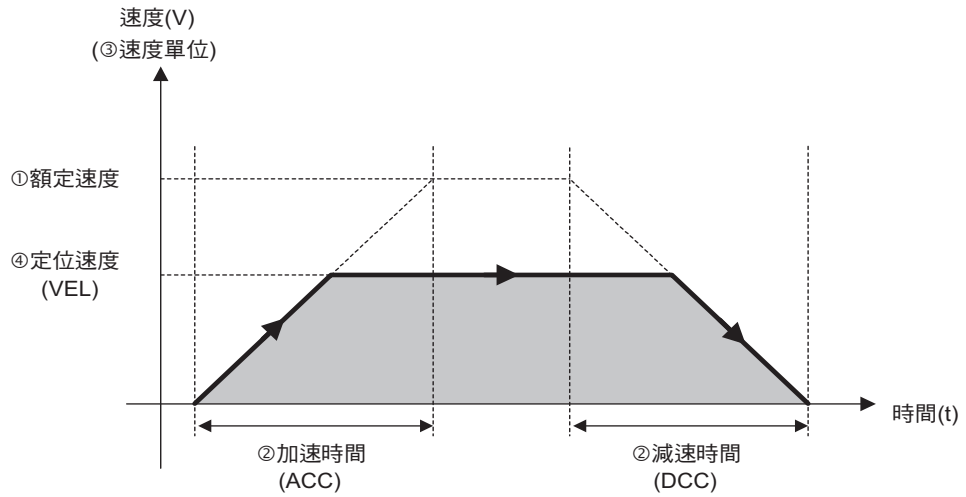
VEL　[邏輯軸名稱 1] 定位速度　　[邏輯軸名稱 2] 定位速度　・・・;

項目	單位	適用的資料
定位 速度	10^n 指令單位 /min， 指令單位 /s， 0.01% (指定額定速度的比率)， 0.0001% (指定額定速度的比率)	・利用立即值直接指定 ・利用長整數型暫存器間接指定

(註) 利用設定參數 OW□□□03 Bit 0 ~ 3 (選擇速度單位) 即可設定單位。

VEL 指令的設定項目

接下來將說明 VEL 指令的設定項目。



① 額定速度

利用固定參數 No. 34 「額定轉數」，即可設定每個軸的額定速度。

② 加速時間／減速時間

利用變更加速時間 (ACC)/ 變更減速時間 (DCC) 指令，即可設定各軸的加減速時間。
ACC 指令所設定的時間即為到達額定速度前的加減速時間。

③ 速度單位

利用設定參數 OW□□□03 Bit 0 ~ 3 (選擇速度單位)，即可設定各軸的速度單位。初始設定為 1 (10^n 指令單位 /min)。

參數名稱	暫存器編號	速度單位	指令範圍
功能設定 1 選擇速度單位	OW□□□03 Bit 0 ～ 3	0：指令單位 /s	0 ～ 2^{31}-1 (指令單位 /s)
		1：10^n 指令單位 /min	0 ～ 2^{31}-1 (10^n 指令單位 /min)
		2：指定為 0.01%	0 ～ 32767 (0.01%)
		3：指定為 0.0001%	0 ～ 3276700 (0.0001%)

補充　選擇設定參數 OW□□□03 Bit 1 時 VEL 指令的設定單位將依固定參數 No. 4 (選擇指令單位) 而異。

固定參數 No. 4 (選擇指令單位)	速度單位 (10^n 指令單位 /min)	備註
pulse	1 = 1000 pulse/min	・選擇指令單位被設定為 pulse 時， n = 3 ・選擇指令單位被設定為 pulse 以外數值時，n = 固定參數 No. 5 (小數點以下位數)
mm	1 = 1 mm/min	
deg	1 = 1 deg/min	
inch	1 = 1 inch/min	
μm	1 = 1 μm/min	

④ 定位速度

各軸的定位速度可利用 VEL 指令的數值或是暫存器來下指令。

程式範例

以下為 VEL 指令的程式範例。
以下係以相對於額定速度 40% 的定位速度來執行 MOV 指令，以及以相對於額定速度 20% 的定位速度來執行 MOV 指令時之範例。

```
INC;                    " 增量模式
ACC [A1]5000;           " 變更加速時間 [ms]
DCC [A1]5000;           " 變更減速時間 [ms]
VEL [A1]4000;           " 變更進給速度 [0.01%]
MOV [A1]3000000;        " 定位
VEL [A1]2000;           " 變更進給速度 [0.01%]
MOV [A1]3000000;        " 定位
END;
```

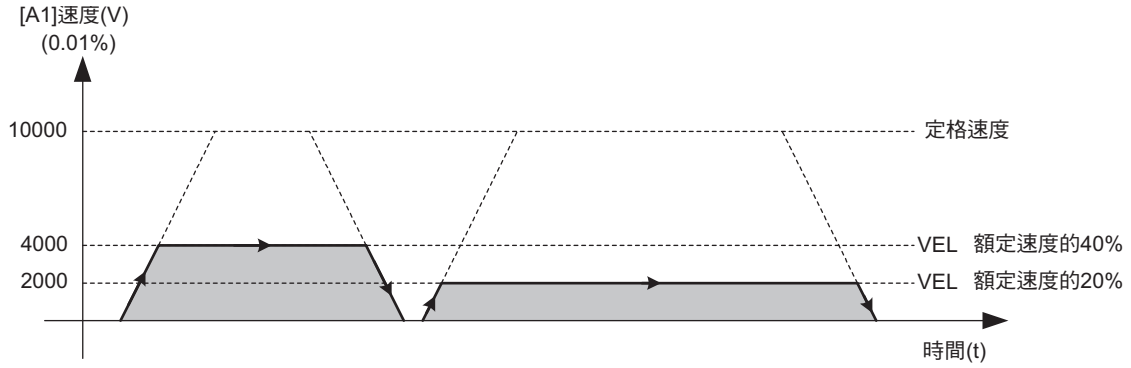


圖 6.14 VEL 指令的程式範例

VEL 指令補充事項

接下來將說明和 VEL 指令相關的 3 項追加事項。

◆ 相關的運動參數

VEL 指令可用來變更設定參數的定位速度。

參數名稱	暫存器編號	說明
速度指令設定	OL□□□10	可用來設定速度指令值。

除了 VEL 指令外，即使變更直接設定參數 OL□□□10 (設定速度指令)，也可變更定位速度。

◆ 覆寫

以 0.01% 為單位，並利用設定參數 OW□□□18 (覆寫)，即可設定 VELL 指令下達定位速度指令時所要執行的百分比 (輸出比率)。
設定參數 OW□□□18 (覆寫) 初始值為 10000(100.00%)。

VEL指令的指令值 × 覆寫(OW□□□18) 0 ~ 327.67% = 定位速度

專有名詞解說

覆寫
覆寫就是對插補類運動語言指令所下達的軸移動速度指令變更輸出比率。

轉軸移動時亦可變更設定參數 OW□□□18 (覆寫)。

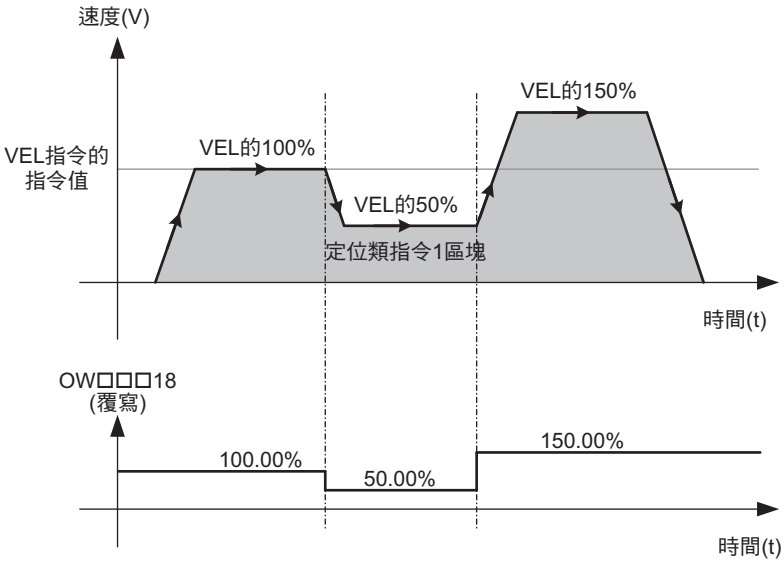


圖 6.15 OW□□□18 (覆寫) 和定位類指令

補充

1. SVR、SVR32 未內置設定參數 OW□□□18 (覆寫)。
2. 使用指定時間定位 (MVT) 時，覆寫的基準，也就是定位速度並非 VEL 指令的指令值。利用指定時間定位 (MVT) 所變更的定位速度即為覆寫的速度標準。
3. 若利用指定時間定位 (MVT) 來使用覆寫，將不會在指定時間內完成定位。
   執行指定時間定位 (MVT) 時的定位速度運算將以 100% 為覆寫標準。
4. 利用固定參數所指定的額定速度和運動程式所執行的定位速度 (VEL)，兩者的速度單位各不相同。

速度	速度單位
固定參數 No. 34 (額定轉數)	旋轉 /min
定位速度 (VEL)	指令單位 /s、10^n 指令單位 /min、0.01%、0.0001%

若要利用算式算出符合定位速度 (VEL) 速度單位的額定速度時，請參閱以下章節。

☞ **馬達速度規格 (第 6-36 頁)**

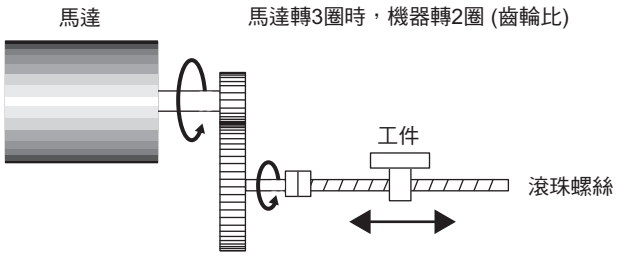
◆ 馬達速度規格

檢討 VEL 指令的設定值時，除了 VEL 指令的指令範圍外，還必須考量馬達的額定速度 / 最高速度。設定 VEL 指令的設定值前，請先確認您所使用的馬達速度規格，以避免所設定的速度值過大。

補充　若馬達屬於旋轉型，則馬達的速度規格以單位時間的旋轉數來表示。
利用固定參數，計算出如下的速度單位 (10n 指令單位 /min) 條件下的額定速度。

· 參數設定範例 (電子齒輪啟動條件下)
只要利用固定參數 No. 4 (選擇指令單位)，選擇 pulse 以外，即可啟動電子齒輪。

固定參數
· No. 4：選擇指令單位 = mm
· No. 5：小數點以下位數 = 3 位數
· No. 6：機器每轉 1 圈的移動量 = 指令單位 10000
· No. 8：馬達端齒輪比 = 3
· No. 9：機器端齒輪比 = 2
· No. 34：額定轉數 = 3000 轉 /min



旋轉1圈時，工作台將移動10 mm
(機器每轉1圈的移動量)

當電子齒輪啟動時，速度單位 (10^n 指令單位 /min) 中的「n」表示小數點以下位數，因此速度單位將變為

(10^n 指令單位 /min) = (10^3 • 0.001 mm/min) = (mm/min)

馬達依額定轉數動作時，機器端的旋轉速度即為

額定轉數 (轉數 /min) x 齒輪比

= 3000 x (2/3) = 2000 (轉 /min)

將機器轉數更改為指令單位 (0.001 mm) 後，

機器每轉 1 圈的移動量 (0.001 mm/ 轉) x 2000 (轉 /min)

＝ 10000×2000 ＝ 20000000 (0.001 mm/min)

假設速度單位為 (mm/min)，

20000000 (0.001 mm/min) ＝ 20000 (mm/min)

－續下頁－

－續前頁－

· 參數設定範例 (關閉電子齒輪－ SVA-01)
利用固定參數 No. 4 (選擇指令單位)，選擇 pulse，即可關閉電子齒輪。

固定參數
· No. 4：選擇指令單位 = pulse
· No. 22：脈衝計數方式 = A/B x 4 (-> 4 倍數)
· No. 34：額定轉數 = 3000 轉 /min
· No. 36：馬達每轉 1 圈的脈衝數 (倍數前) = 16384 pulse/ 轉

關閉電子齒輪時，速度單位 (10^n 指令單位 /min) 中的「n」為 3，因此速度單位將變為

(10^n 指令單位 /min) = (10^3 pulse/min) = (1000 pulse/min)

若將馬達額定轉數更改為 pulse 單位後，

額定轉數 (轉數 /min) x (馬達每轉 1 圈的脈衝數 (pulse/ 轉) x 倍數)

＝ 3000×(16384×4) ＝ 196608000 (pulse/min)

假設速度單位為 (1000 pulse/min)，

196608000 (pulse/min) ＝ 196608 (1000 pulse/min)

· 參數設定範例 (關閉電子齒輪－ SVC、SVC32、SVR、SVR32、SVC-01、SVB-01、PO-01 時)
利用固定參數 No. 4 (選擇指令單位)，選擇 pulse，即可關閉電子齒輪。

固定參數
· No. 4：選擇指令單位 = pulse
· No. 34：額定轉數 = 3000 轉 /min
· No. 36：馬達每轉 1 圈的脈衝數 = 65536 pulse/ 轉

關閉電子齒輪時，速度單位 (10^n 指令單位 /min) 中的「n」為 3，因此速度單位將變為

(10^n 指令單位 /min) = (10^3 pulse/min) = (1000 pulse/min)

若將馬達額定轉數更改為 pulse 單位後，

額定轉數 (轉數 /min) x (馬達每轉 1 圈的脈衝數 (pulse/ 轉)

＝ 3000×65536 ＝ 196608000 (pulse/min)

假設速度單位為 (1000 pulse/min)，

196608000 (pulse/min) ＝ 196608 (1000 pulse/min)

若要讓轉軸正確動作，即使參數設定範例中所未提到的固定參數，也必須正確進行設定。

如欲進一步瞭解運動參數，請參閱以下手冊。

MP3000 系列　運動控制功能　使用手冊 (資料編號：YTWNCO-14013A)

設定插補進給最高速度 (FMX)

設定插補進給最高速度 (FMX) 是一種用來設定插補指令 (MVS、MCW、MCC、SKP) 最高速度的指令。
設定的插補進給最高速度將維持不變，直到 FMX 指令被重新設定為止。
程式開始運轉時，FMX 指令處於未設定狀態。
當您執行以下插補相關指令前，必須先執行 FMX 指令。

- 線性內插 (MVS)
- 循環內插 (MCW、MCC)
- 螺旋內插 (MCW、MCC)
- 附略過功能線性內插 (SKP)
- 設定插補進給速度比率 (IFP)
- 變更插補加速時間 (IAC)
- 變更插補減速時間 (IDC)
- 變更暫停時的插補減速時間 (IDH)

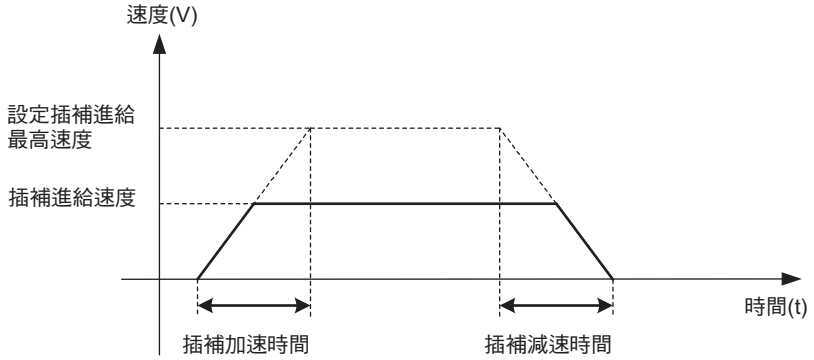


圖 6.16 設定插補進給最高速度

註記

若未下達 FMX 指令，即先行執行各種插補相關指令 (MVS、MCW、MCC、SKP、IFP、IAC、IDC、IDH)，運動程式警告將會發生。

補充

1. 關於插補的各種指令，必須以事先設定插補進給最高速度為前提來進行處理。變更插補加速時間 (IAC)、變更插補減速時間 (IDC) 或變更暫停時的插補減速時間 (IDH) 的指令皆可用來指定速度 0 到插補進給最高速度所需的時間，因此必須事先進行設定。
2. FMX 指令和設定參數之間並無任何相關性。
   FMX 指令所指定的插補進給最高速度為運動程式專用的控制資料，因此無法透過設定參數來指定。

格式

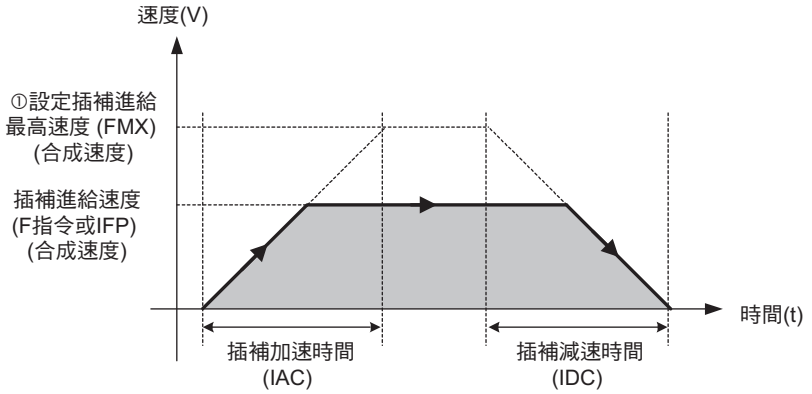
以下為 FMX 指令的格式。

FMX 設定 T 插補進給最高速度；

項目	單位	適用的資料
設定插補進給最高速度	需符合 FUT 指令的設定單位 指令單位 /min 或指令單位 /s	・利用立即值直接指定 ・利用長整數型暫存器間接指定

FMX 指令的設定項目

接下來將說明 FMX 指令的設定項目。



① 設定插補進給最高速度

設定插補進給最高速度指令就是利用 FMX 指令來指定接在字元「T」後面的數值，或是利用暫存器進行設定。設定插補進給最高速度的指令範圍為 1 ～ 2^{31}-1(指令單位 /min)。

設定插補進給最高速度指令適用於和插補有關的所有指令。

使用插補指令 (MVS、MCW、MCC、SKP) 時，必須在運動程式的起始位置下達 FMX 指令。

程式範例

以下為 FMX 指令的程式範例。

```
INC;                              " 增量模式
FMX T300000;                      " 設定插補進給最高速度
IAC T4000;                        " 變更插補加速時間 [ms]
IDC T4000;                        " 變更插補減速時間 [ms]
IFP P75;                          " 設定插補進給速度比率定 [%]
MVS [A1]30000 [B1]30000;          " 線性內插
MVS [A1]30000 [B1]30000 F150000;  " 線性內插 (F 指令 )
END;
```

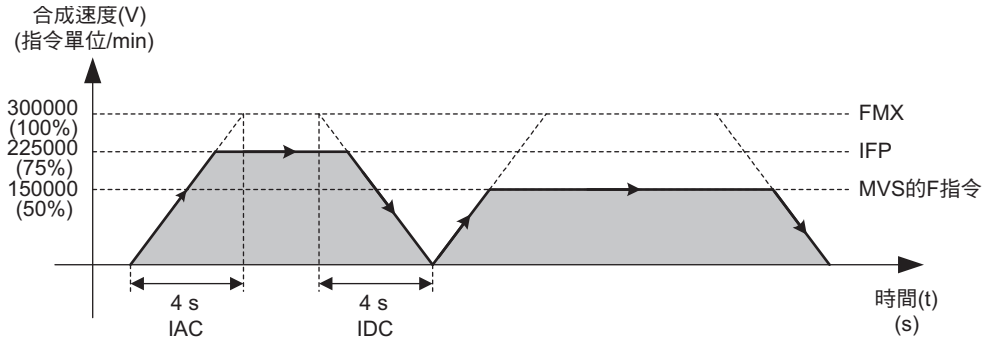


圖 6.17 FMX 指令的程式範例

設定軸別插補進給最高速度 (IFMX)

設定軸別插補進給最高速度 (IFMX) 指令可在使用插補指令 (MVS、SKP、MCW、MCC) 時，依不同轉軸設定插補進給最高速度。

設定的軸別插補進給最高速度將維持不變，直到 IFMX 指令被重新設定為止。

一旦實際的插補進給速度超過 IFMX 指令所設定的數值時，運動程式警告就會發生，所有轉軸亦將立刻停止動作。

程式開始運轉時，IFMX 指令將被視為尚未設定完成，此時轉軸將在無速度限制的條件下執行動作。

以下係對 2 個軸 (A1 軸和 B1 軸) 進行線性內插時，僅對 A1 軸設定 IFMX 指令之時間圖。

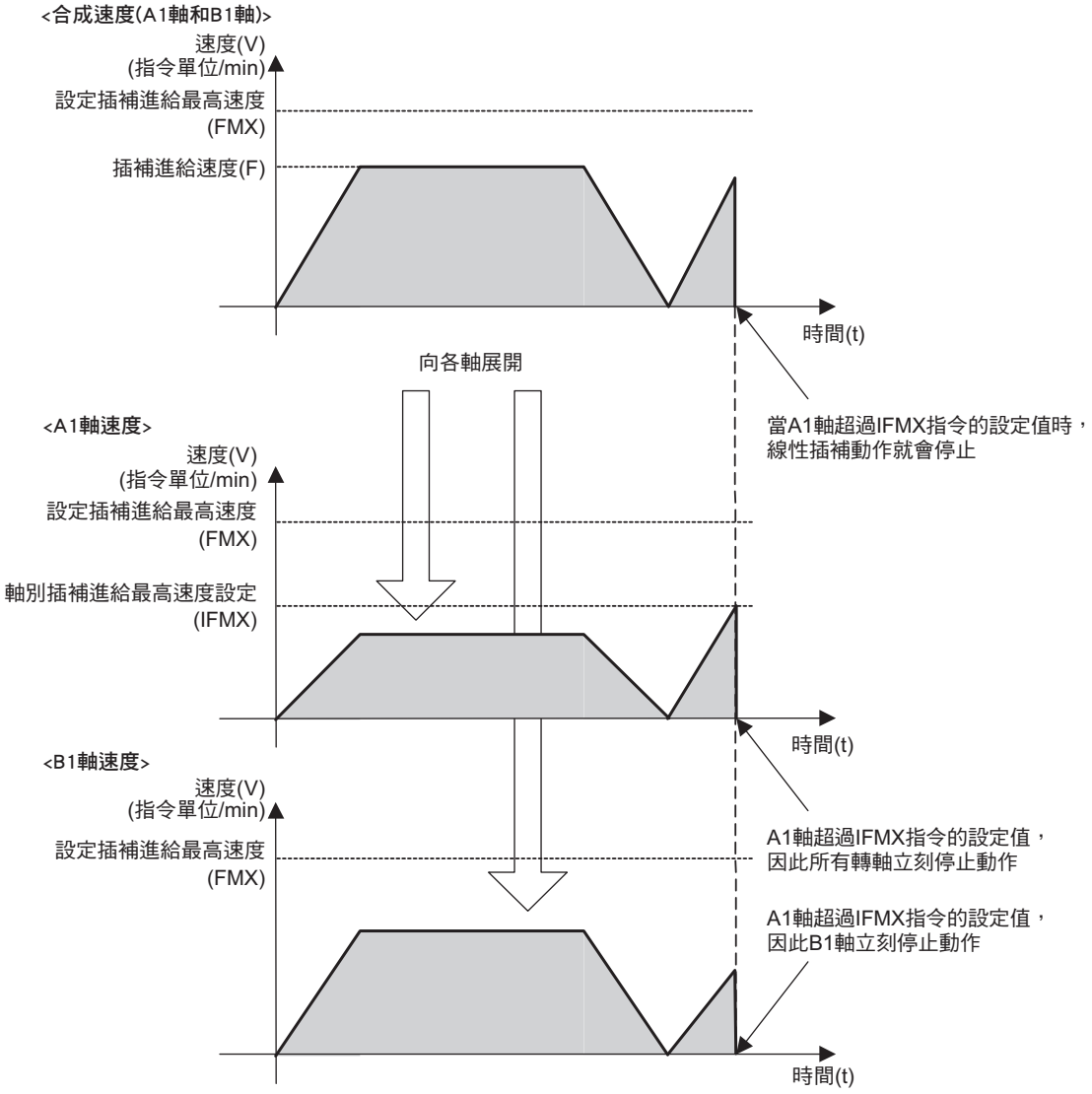


圖 6.18 設定軸別插補進給最高速度

補充

1. 若未設定 IFMX 指令，或是設定為 0，轉軸就會在無速度限制的條件下執行動作。
2. 在運動程式裡，IFMX 指令的設定值 [指令單位 /min] 會被轉換為 [指令單位 /scan]。和插補進給速度 (F) 不同，一旦轉換單位後，小數點以下的數值就會被進位，並以整數值的形式來判定速度。因此，即使是在根據高速掃描及插補進給速度「插補進給速度 (F) > 速度限制值 (IFMX 指令)」時轉軸執行動作，但是仍然不會超過插補進給速度。
   利用下列算式即可轉換插補進給速度及 IFMX 指令設定值 [指令單位 /min] 的單位。
   插補進給速度 (速度限制值) [指令單位 /scan] = F 值 (IFMX 指令設定值)/60 (s)/1000 (ms) x Ts (高速掃描)

格式

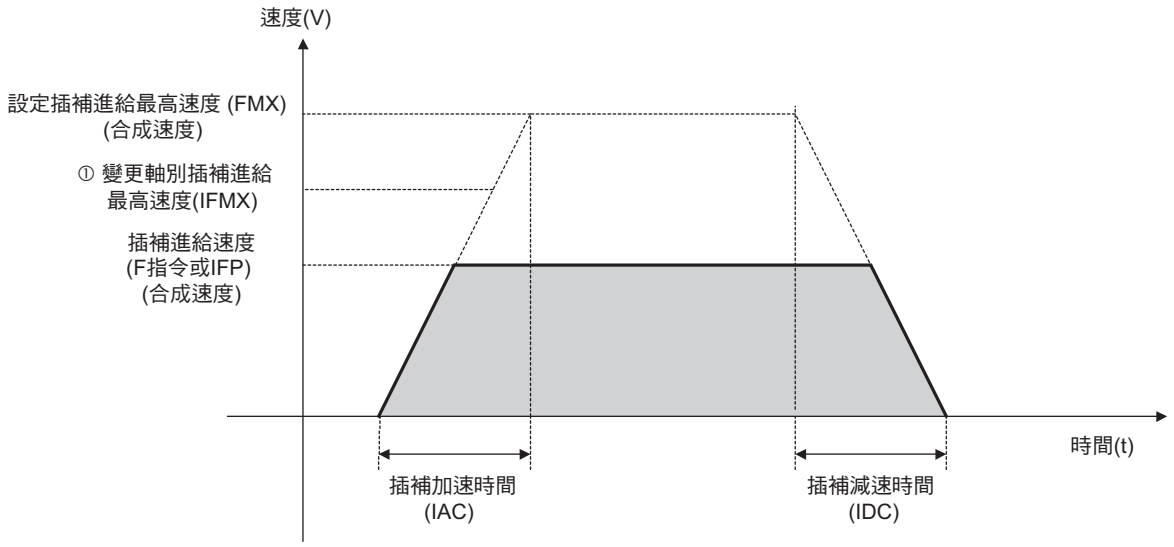
以下為 IFMX 指令的格式。

IFMX [邏輯軸 1] 設定軸別插補進給最高速度 [邏輯軸 2] 設定軸別插補進給最高速度 …；

項目	單位	適用資料
設定軸別插補進給最高速度	依照 FUT 指令的設定單位 指令單位 /min 或指令單位 /s	・利用立即值直接指定 ・利用長整數型暫存器間接指定

IFMX 指令的設定項目

接下來將說明 IFMX 指令的設定項目。



① 設定軸別插補進給最高速度

如欲設定軸別插補進給最高速度時，可利用 IFMX 指令來設定數值，或是利用暫存器來設定。

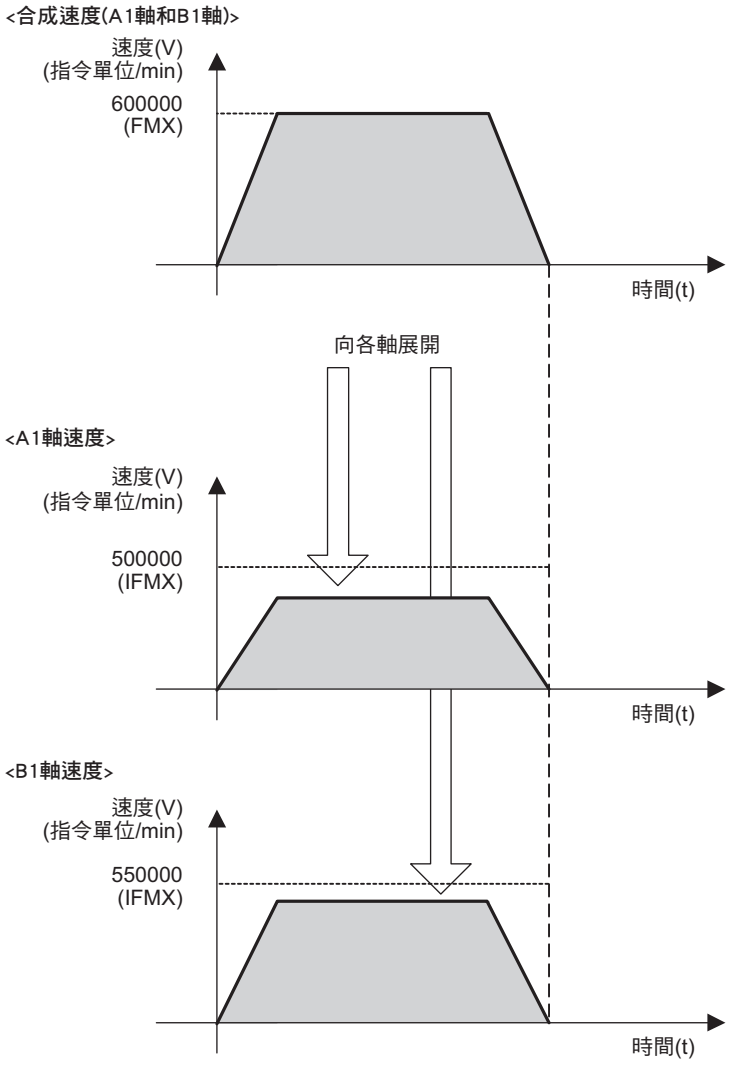
IFMX 指令的設定範圍為 0 ～ 2^{31}-1 (指令單位 /min)。

當 IFMX 指令被設定為 0 時，IFMX 指令將被視為尚未設定完成，此時轉軸將在無速度限制的條件下執行動作。

程式範例

以下為 IFMX 指令的程式範例。

```
FMX T600000;                        " 設定插補進給最高速度 "
IFMX [A1]500000 [B1]550000;         " 設定軸別插補進給最高速度 "
INC;                                " 增量模式 "
IAC T500;                           " 插補加速時間  = 500 ms"
IDC T500;                           " 插補減速時間 = 500 ms"
MVS [A1]30000 [B1]40000 F600000;    " 線性內插指令 "

END;
```



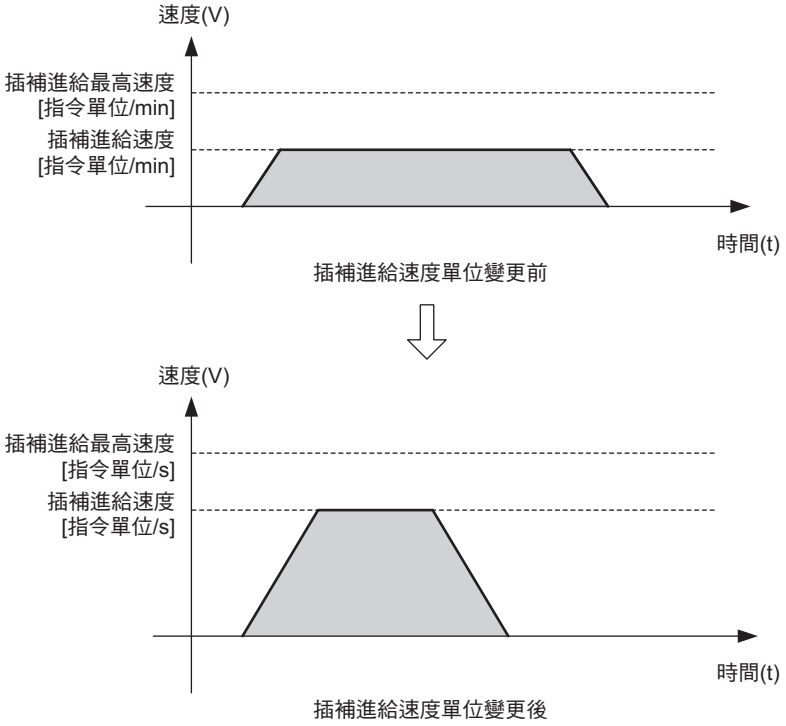
變更插補進給速度單位 (FUT)

變更插補進給速度單位 (FUT) 是一項可用來變更下列插補指令速度單位的指令。

- 設定插補進給最高速度 (FMX)
- 設定軸別插補進給最高速度 (IFMX)
- 線性內插 (MVS)
- 循環內插 (MCW/MCC)
- 螺旋內插 (MCW/MCC)
- 附略過功能線性內插 (SKP)

選擇的插補進給速度的單位將維持不變，直到 FUT 指令被重新設定為止。

程式開始運轉後，插補進給速度單位將變成 [指令單位 /min]。



補充

1. 若不執行 FUT 指令，插補進給速度的單位將變為 [指令單位 /min]。
2. 指定 FUT 指令時，若超出規定的設定範圍，編譯器就會發生錯誤。
3. FUT 指令適用於以下版本。

控制器及 MPE720	適用版本
MP3000 系列	Ver 1.08 以後版本
MPE720 Ver.7	Ver 7.23 以後版本

格式

以下為 FUT 指令的格式格式。

FUT　U 插補進給速度單位編號；

項目	單位	適用的資料
插補進給速度單位編號	–	利用立即值直接指定 0：指令單位 /min 1：指令單位 /s

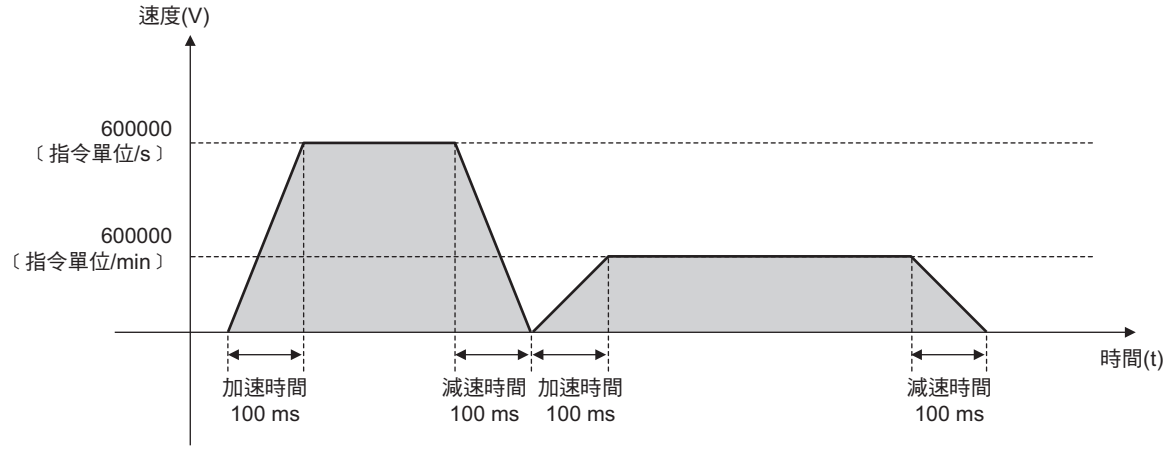
註記

執行 FUT 指令且變更單位時，FMX、IFMX、F 和 IFP 的數值將會被設定為初始值 (0)。請在變更單位後，依照設定單位，重新設定插補進給速度。

程式範例

以下為 FUT 指令的程式範例。

```
FUT U1;                    " 插補進給速度單位變更 [ 指令單位 /min] → [ 指令單位 /s]"
INC;                       " 增量模式
FMX T600000;               " 插補進給最高速度 [ 指令單位 /s]"
IAC T100;                  " 加速時間 100 ms"
IDC T100;                  " 減速時間 100 ms"
MVS [A1]10000 F600000;     " 線性內插　插補進給速度 600000 指令單位 /s"
FUT U0;                    " 插補進給速度單位變更 [ 指令單位 /s] → [ 指令單位 /min]"
FMX T600000;               " 插補進給最高速度 [ 指令單位 /min]"
MVS [A1]10000 F600000;     " 線性內插　插補進給速度 600000 指令單位 /min"
END;
```



設定插補進給速度比率 (IFP)

設定插補進給速度比率 (IFP) 是一項用來設定下列軸移動指令進給速度的指令。指定進給速度時，需指定為插補進給最高速度相對應的比率。

- 線性內插 (MVS)
- 循環內插 (MCW、MCC)
- 螺旋內插 (MCW、MCC)
- 附略過功能線性內插 (SKP)

本手冊將上述軸移動指令統稱為 「插補指令」，進給速度則稱為 「插補進給速度」。設定的插補進給速度將維持不變，系統將維持該速度，直到設定插補進給速度比率 (IFP) 指令被重新設定，或是利用插補指令來執行 F 指令為止。
當程式開始運轉時，並未設定插補進給速度。執行插補指令前，請執行設定插補進給速度比率 (IFP) 指令或 F 指令，來設定插補進給速度。

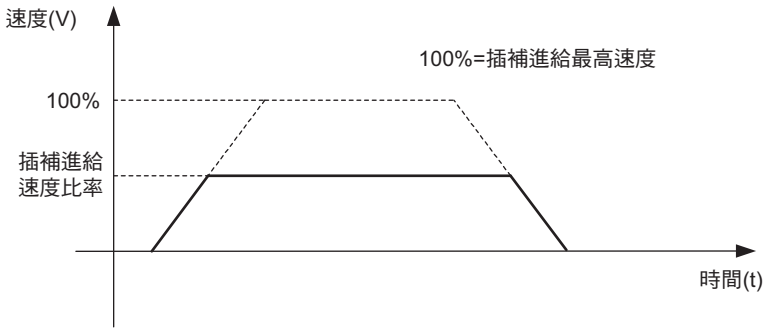


圖 6.19 設定插補進給速度比率

重要

1. 設定 IFP 指令前，請先指定插補進給最高速度 (FMX)。執行 IFP 指令前，若未先指定好 FMX 指令，恐將使得運動程式警告發生。
2. 若在執行插補指令前，不曾指定過插補進給速度，運動程式將發出警告。

補充

1. F 指令就是一項利用插補指令來指定接在字元 「F」後面的數值，或是利用暫存器來指定插補進給速度的方法。利用 「指令單位 /min」，即可指定插補進給速度。
2. 若先執行 F 指令再執行 IFP 指令，F 指令所指定的插補進給速度就會被取消。
   反之，若先執行 IFP 指令再執行 F 指令，IFP 指令所指定的插補進給速度則會被取消。
3. IFP 指令是一項用來設定插補指令 (MVS、MCW、MCC、SKP) 進給速度的指令。利用 VEL 指令即可進行定位類指令 (MOV、EXM) 進給速度的設定。
4. IFP 指令和設定參數之間並無任何相關性。
   IFP 指令所指定的插補進給速度比係為運動程式專用的控制資料，因此無法利用設定參數來指定。

格式

以下為 IFP 指令的格式。

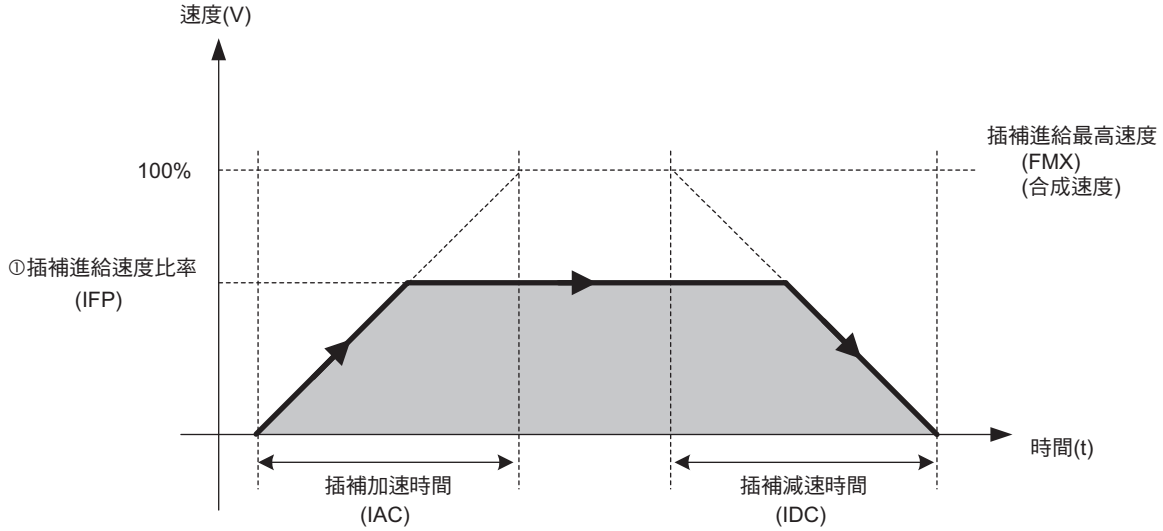
IFP　P 插補進給速度比率；

項目	單位	適用的資料
插補進給速度比率	%	・利用立即值直接指定 ・利用長整數型暫存器間接指定

補充　請勿在插補指令 (MVS、MCW、MCC、SKP) 相同的區塊內指定 IFP 指令。

IFP 指令的設定項目

接下來將說明 IFP 指令的設定項目。



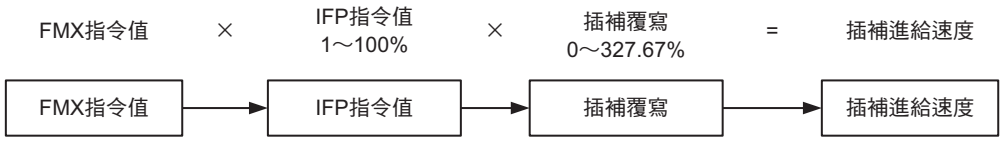
① 插補進給速度比率

若要指定插補進給速度比，可利用 IFP 指令指定接在字元「P」後面的數值，或是以暫存器來下指令。
利用 IFP 指令所設定的時間即為插補進給最高速度的插補進給速度比率。
插補進給速度是一項利用插補指令 (MVS、MCW、MCC、SKP) 所指定的所有轉軸的合成速度。
插補進給速度比率的指令範圍為 1~100%。
您可以選擇是 / 否使用插補覆寫作為插補進給速度。
插補覆寫的相關使用方法，請參閱以下章節。

☞ **工作暫存器 (第 1-22 頁)**

範例 **不使用插補覆寫**

範例 **使用插補覆寫時**



重要

當您所指定的 IFP 指令值 (%) 超過 100%，運動程式警告就會發生。

補充

1. 指定插補進給速度時，除了可利用 IFP 指令，另外還有其他指定方法 (利用 F 指令來指定)。
   如欲進一步瞭解插補進給速度，請參閱以下章節。
   ☞ **線性內插 (MVS)–MVS 指令的設定項目 (第 6-81 頁)**
2. 使用插補覆寫來下指令時，一旦您所指定的插補進給速度超過 FMX 指令值，插補進給速度的輸出值將變為 FMX 指令值。

程式範例

以下為 IFP 指令的程式範例。

```
INC;                          " 增量模式
FMX T300000;                  " 設定插補進給最高速度 [ 指令單位 /min]
IAC T4000;                    " 變更插補加速時間 [ms]
IDC T4000;                    " 變更插補減速時間 [ms]
IFP P75;                      " 設定插補進給速度比率 [%]
MVS [A1]30000 [B1]30000;      " 線性內插
DL00000 = 50;                 " 插補進給速度比率 [%]
IFP PDL00000;                 " 設定插補進給速度比率 [%]
MVS [A1]30000 [B1]30000;      " 線性內插
END;
```

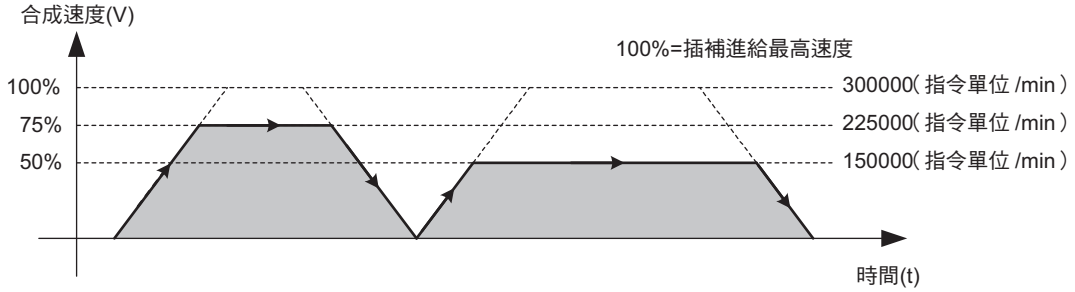


圖 6.20 IFP 指令的程式範例

變更插補加速時間 (IAC)

變更插補加速時間 (IAC) 是一項用來變更下述軸移動指令加速時間的指令。

- 線性內插 (MVS)
- 循環內插 (MCW、MCC)
- 螺旋內插 (MCW、MCC)
- 附略過功能線性內插 (SKP)

執行 IAC 指令前，必須先執行設定插補進給最高速度 (FMX)。
變更的加速時間將維持不變，直到變更插補加速時間 (IAC) 重新被設定為止。
當程式開始運轉時，插補加速時間為 0 ms。

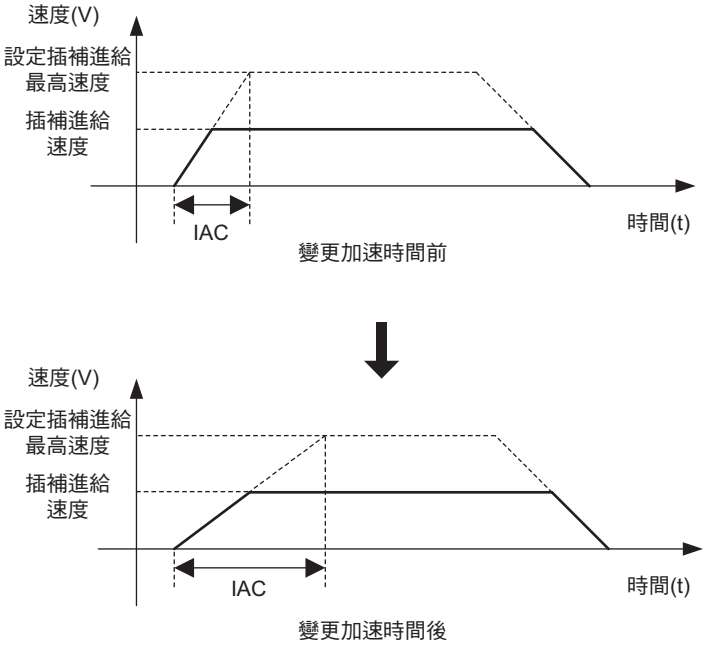


圖 6.21 變更插補加速時間

補充

1. 變更插補加速時間 (IAC) 是一項用來設定插補指令 (MVS、MCW、MCC、SKP) 加速時間的指令。
   利用 ACC 指令，即可設定定位類指令 (MOV、EXM、MVT) 的加速時間。
2. IAC 指令和設定參數之間並無任何相關性。
   利用 IAC 指令所指定的插補加速時間係為運動程式專用的控制資料，因此無法透過設定參數來指定。

格式

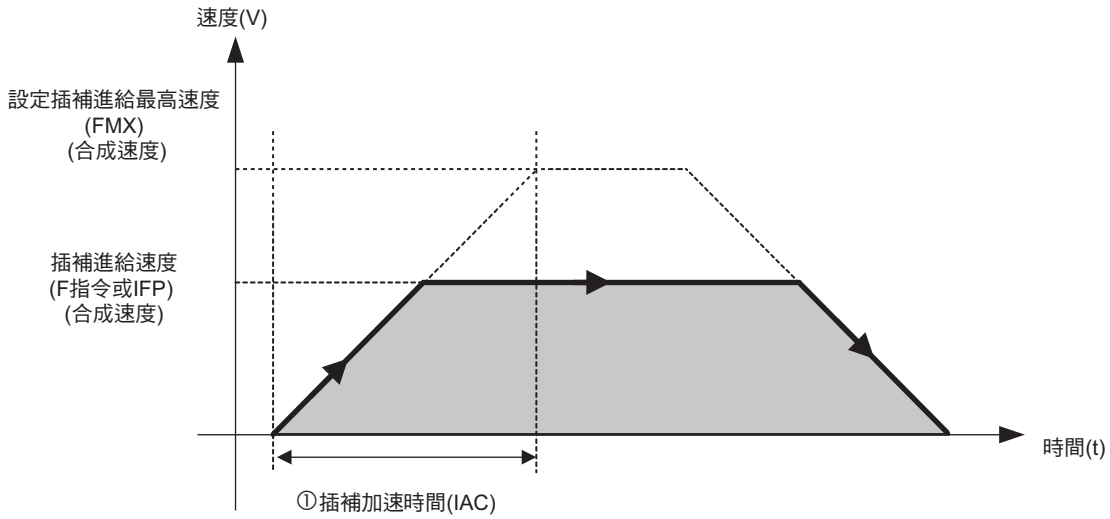
以下為 IAC 指令的格式。

IAC　T 插補加速時間；

項目	單位	適用的資料
插補加速時間	符合 IUT 指令設定單位的 ms 或指令單位 /s^2	・利用立即值直接指定 ・利用長整數型暫存器間接指定

IAC 指令的設定項目

接下來將說明 IAC 指令的設定項目。



① 插補加速時間

若要指定插補加速時間，可利用 IAC 指令指定接在字元「T」後面的數值，或是使用暫存器下指令。
IAC 指令所設定的時間就是從速度 0 提高到插補進給最高速度所需的加速時間。
插補加速時間的指令範圍為 0~32767 ms。

程式範例

以下為 IAC 指令的程式範例。

```
INC;                                  " 增量模式
FMX T300000;                          " 設定插補進給最高速度 [ 指令單位 /min]
IDC T4000;                            " 變更插補減速時間 [ms]
IAC T2000;                            " 變更插補加速時間 [ms]
MVS [A1]30000 [B1]30000 F150000;      " 線性內插
DL00000 = 4000;                       "" 補間加速時間 [ms]
IAC TDL00000;                         " 變更插補加速時間 [ms]
MVS [A1]30000 [B1]30000;              " 線性內插
END;
```

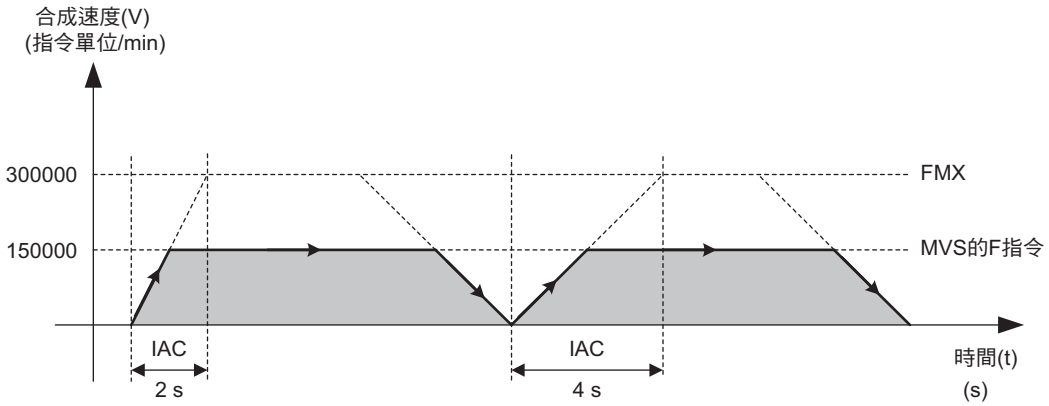


圖 6.22 IAC 指令的程式範例

變更插補減速時間 (IDC)

變更插補減速時間 (IDC) 是一項用來變更下述軸移動指令減速時間的指令。

- 線性內插 (MVS)
- 循環內插 (MCW、MCC)
- 螺旋內插 (MCW、MCC)
- 附略過功能線性內插 (SKP)

執行 IDC 指令前，必須先執行設定插補進給最高速度 (FMX)。變更的減速時間將維持不變，直到變更插補減速時間 (IDC) 重新被設定為止。
當程式開始運轉時，插補減速時間為 0 ms。

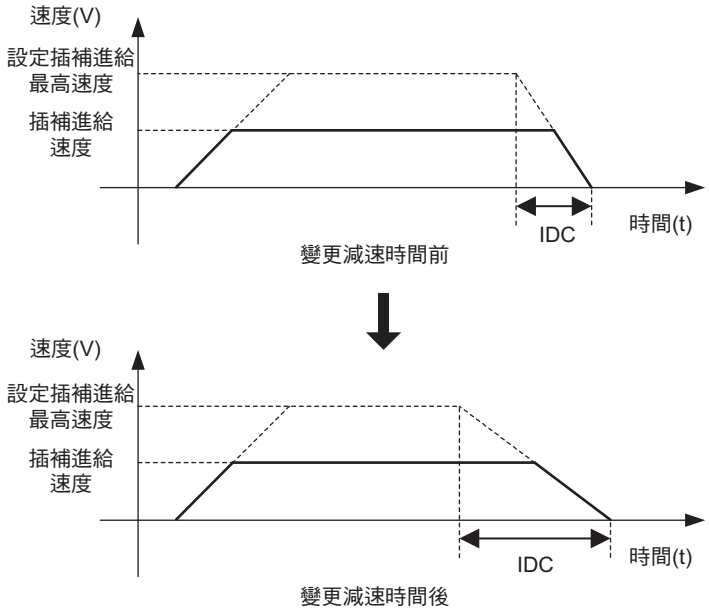


圖 6.23 變更插補減速時間

補充

1. 變更插補減速時間 (IDC) 是一項用來設定插補指令 (MVS、MCW、MCC、SKP) 減速時間的指令。
   利用 DCC 指令，即可設定定位類指令 (MOV、EXM、MVT) 的減速時間。
2. IDC 指令和設定參數之間並無任何相關性。
   利用 IDC 指令所指定的插補減速時間係為運動程式專用的控制資料，因此無法透過設定參數來指定。

格式

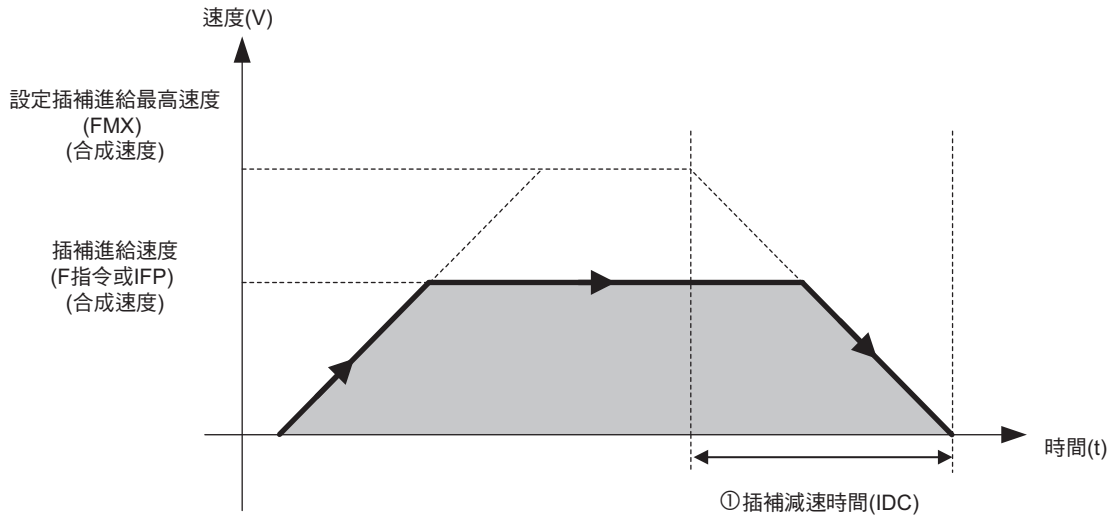
以下為 IDC 指令的格式。

IDC　T 插補減速時間 ;

項目	單位	適用的資料
插補減速時間	符合 IUT 指令設定單位的 ms 或指令單位 /s^2	・利用立即值直接指定 ・利用長整數型暫存器間接指定

IDC 指令的設定項目

接下來將說明 IDC 指令的設定項目。



① 插補減速時間

若要指定插補減速時間，可利用 IDC 指令指定接在字元「T」後面的數值，或是使用暫存器下指令。
IDC 指令所設定的時間就是從插補進給最高速度降至速度 0 所需的減速時間。
插補減速時間的指令範圍為 0~32767 ms。

程式範例

以下為 IDC 指令的程式範例。

```
INC;                                  " 增量模式
FMX T300000;                          " 設定插補進給最高速度 [ 指令單位 /min]
IAC T4000;                            " 變更插補加速時間 [ms]
IDC T2000;                            " 變更插補減速時間 [ms]
MVS [A1]30000 [B1]30000 F150000;      " 線性內插
DL00000 = 4000;                       " 插補減速時間 [ms]
IDC TDL00000;                         " 變更插補減速時間 [ms]
MVS [A1]30000 [B1]30000;              " 線性內插
END;
```

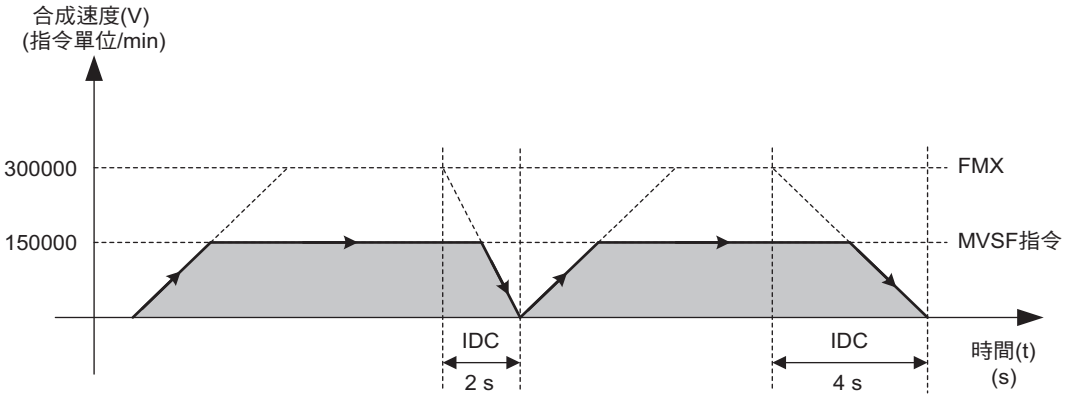


圖 6.24 IDC 指令的程式範例

變更暫停用插補減速時間 (IDH)

變更暫停用插補減速時間 (IDH) 是一項可在下述軸移動指令暫停時，用來變更減速時間的指令。

- 線性內插 (MVS)
- 循環內插 (MCW、MCC)
- 螺旋內插 (MCW、MCC)
- 附略過功能線性內插 (SKP)

當您希望系統迅速減速停止時，使用 IDH 指令，將優於以 IDC (變更插補減速時間) 指令指定減速時間的做法。

執行 IDH 指令前，必須先設定好插補進給最高速度 (FMX)。

變更的減速時間將維持不變，直到 IDH 指令重新被設定為止。

若未執行 IDH 指令，將採用 IDC 指令所設定的減速時間。

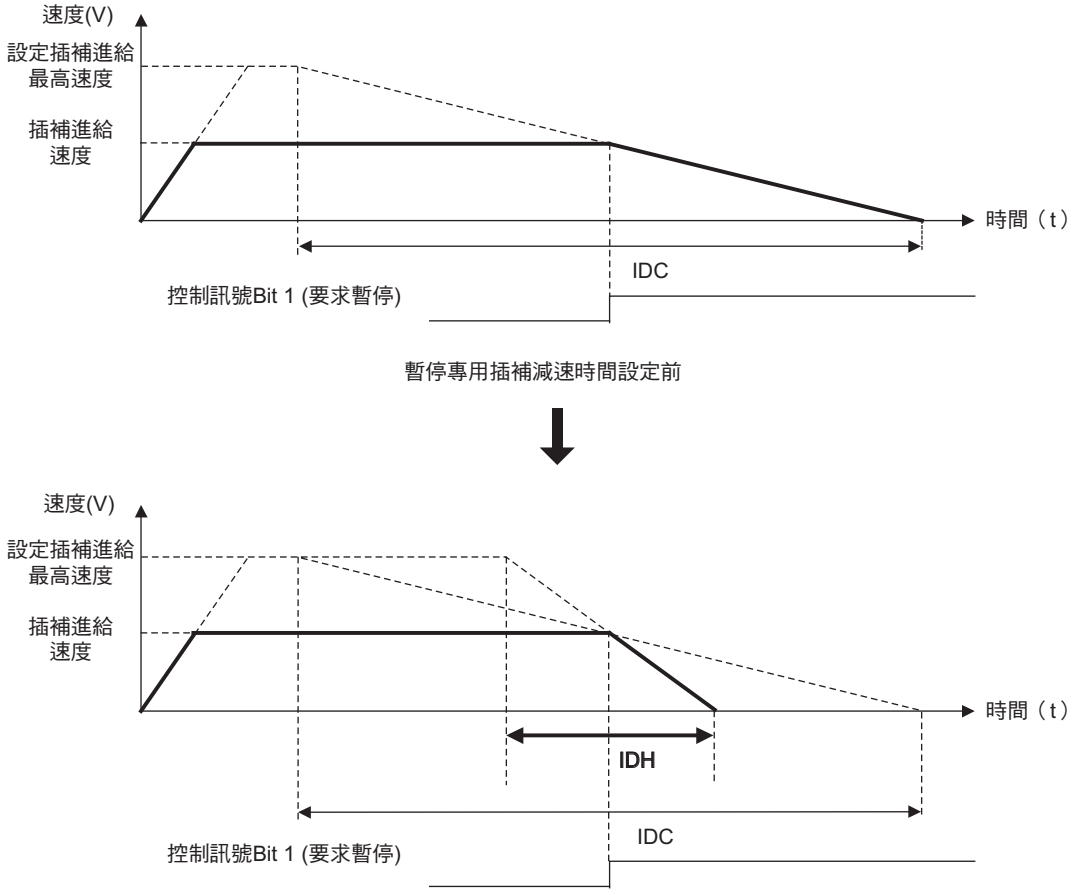


圖 6.25 變更暫停用插補減速時間

格式

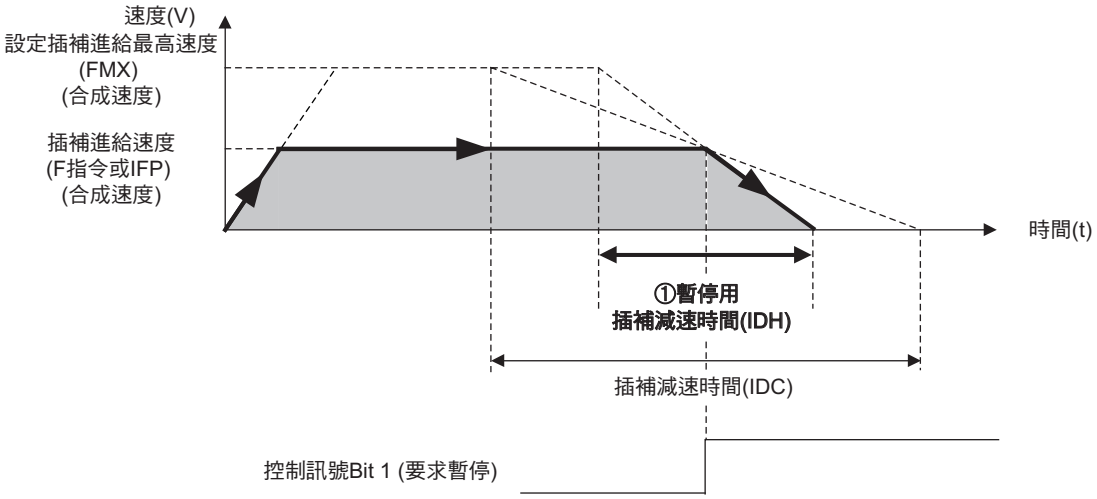
以下為 IDH 指令的格式。

IDH T 暫停用插補減速時間 ;

項目	單位	適用的資料
暫停用插補減速時間	符合 IUT 指令設定單位的 ms 或指令單位 /s^2	・利用立即值直接指定 ・利用長整數暫存器間接指定

IDH 指令的設定項目

接下來將說明 IDH 指令的設定項目。



① 暫停用插補減速時間

若要指定暫停用插補減速時間，可利用 IDH 指令來指定接在字元「T」後面的數值或是使用暫存器來指定。

IDH 指令所設定的時間就是從插補進給最高速度降至速度 0 所需的減速時間。

暫停專用插補減速時間的指令範圍為 0~32767 ms。

程式範例

以下為 IDH 指令的程式範例。

```
INC;                                  " 增量模式
FMX T300000;                          " 設定插補進給最高速度 [ 指令單位 /min]
IAC T2000;                            " 變更插補加速時間 [ms]
IDC T4000;                            " 變更插補減速時間 [ms]
IDH T100;                             " 暫停專用插補減速時間設定 [ms]
MVS [A1]30000 [B1]30000 F150000;      " 線性內插 ( 轉軸移動時，要求暫停訊號為 ON)
END;
```

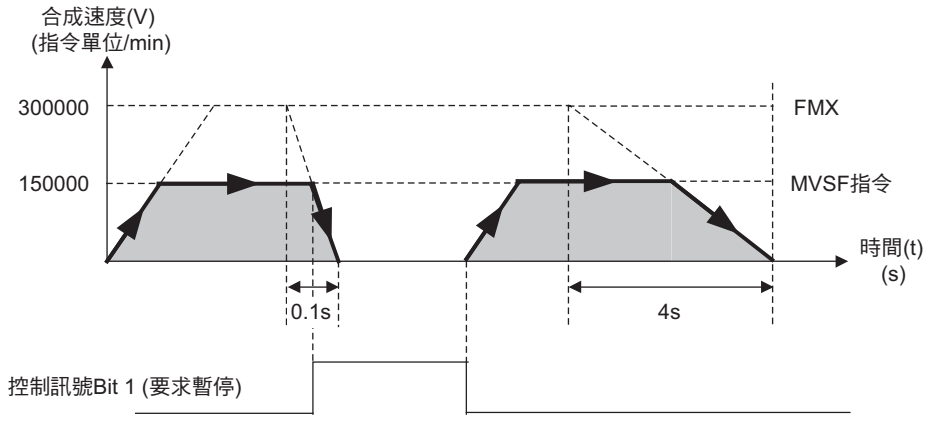


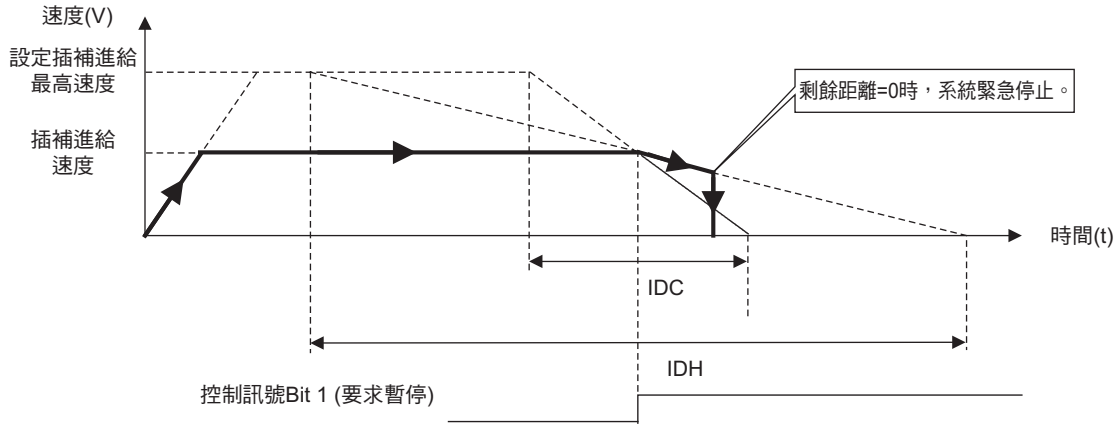
圖 6.26 IDH 指令的程式範例

IDH 指令補充事項

◆「IDH 指令所指定的減速時間＞ IDC 指令所指定的減速時間」時的動作

當 IDH 指令的減速時間大於 IDC 指令的減速時間時，將有可能出現剩餘距離 (插補指令剩餘移動量) 小於減速停止距離 (在指定的減速時間條件下，減速到 0 速所需的移動量) 的情形。

若剩餘距離小於減速停止距離，系統將在剩餘距離變為 0 時緊急停止，此點需特別注意。



◆ 略過訊號輸入時的動作

利用 IDH 指令完成減速時間設定後，一旦 SKP 指令執行狀態下有略過訊號輸入，就會依照 IDH 指令所設定的減速時間來執行。

◆ 設定加減速模式時之動作

當 ACCMODE 指令完成加減速模式的設定後，在插補指令執行中執行要求暫停時的動作如以下所示。

■ 開始進行下一個區塊的插補送出指令前，收到要求暫停訊號

將依照 IDH 指令所設定的減速時間進行減速。

當前一個區塊的剩餘距離為 0 時，下一個區塊的插補指令仍不會開始進行送出指令，此時將不會執行插補連接。

■ 開始進行下一個區塊的插補送出指令後，收到要求暫停訊號

前一個和下一個區塊皆會遵照 IDH 指令所設定的減速時間。

當要求暫停的訊號被解除後，無論前一個或下一個區塊皆會在剩餘距離進行送出指令動作。

變更插補加減速單位 (IUT)

變更插補加減速單位 (IUT) 指令是一項用來變更插補指令 (MVS、SKP、MCW/MCC) 加速度單位的指令。

IUT 指令所選擇的單位僅適用於以下指令。

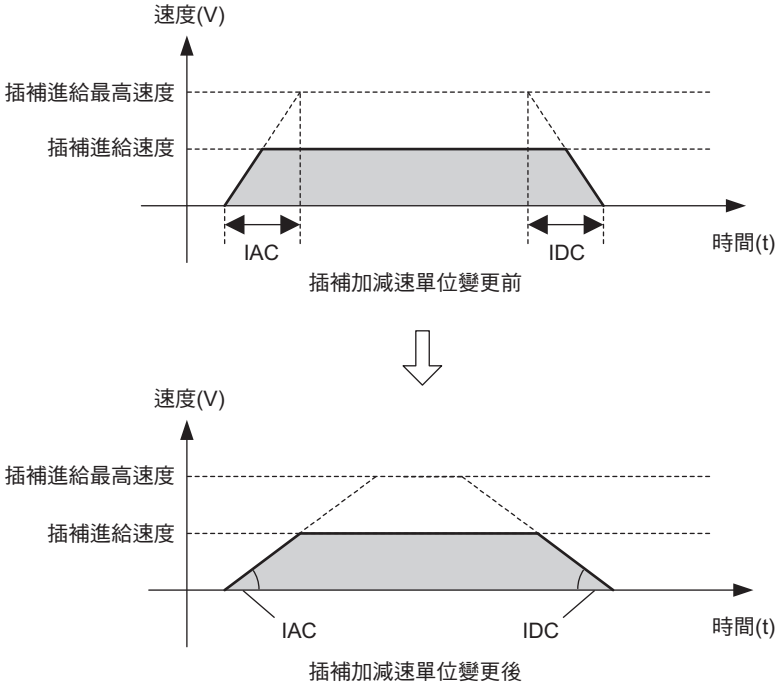
・變更插補加速時間指令 (IAC)

・變更插補加速時間指令 (IDC)

・變更暫停專用插補減速時間指令 (IDH)

選擇的插補加減速單位將維持不變，直到 IUT 指令被重新設定為止。

程式開始運轉後，加減速單位將變為 [ms]。



補充

1. 若不執行 IUT 指令，插補加減速的速度單位將變為 [ms]。
2. 指定 IUT 指令時，若超出規定的設定範圍，編譯器就會發生錯誤。
3. IUT 指令適用於以下版本。

控制器及 MPE720	適用版本
MP3000 系列	Ver 1.08 以後版本
MPE720 Ver.7	Ver 7.23 以後版本

格式

以下為 IUT 指令的格式。

IUT　U 插補加減速單位編號；

項目	單位	適用的資料
插補加減速單位編號	–	利用立即值直接指定 0：ms (預設值) 1：指令單位 /s^2

重要

1. 當您利用 IUT 指令來變更插補加減速單位後，系統為了安全性考量，將自動設定為最和緩的加減速速度。單位變更完成後，請依照設定單位，重新設定插補加減速度。

IUT 設定值	IAC、IDC、IDH 的設定值
U0 → U1	1 [指令單位 /s^2]
U1 → U0	32767 [ms]

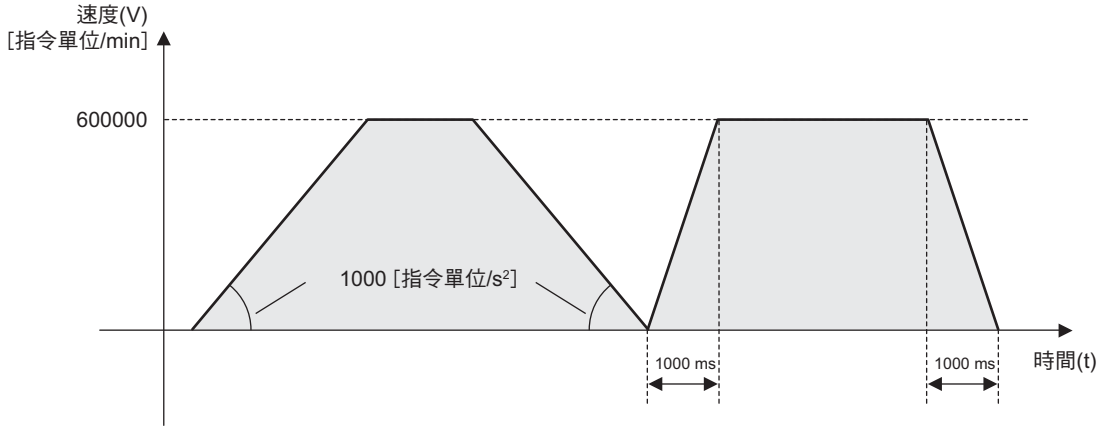
2. IAC、IDC 和 IDH 指令的設定範圍依加減速單位而異。

IUT 設定值	IAC、IDC、IDH 的設定值
U0	0 ～ 32767 [ms]
U1	1 ～ 2147483647 [指令單位 /s^2]

程式範例

以下為 IUT 指令的程式範例。

```
INC;                          " 增量模式
FMX T600000;                  " 插補進給最高速度 "
IUT U1;                       " 插補加減速單位變更 [ms] → [ 指令單位 /s²]"
IAC T1000;                    " 加速度 1000 的指令單位 /s²"
IDC T1000;                    " 減速度 1000 的指令單位 /s²"
MVS [A1]1000000 F600000;      "MVS①"
IUT U0;                       " 插補加減速單位變更 [ 指令單位 /s²] → [ms]"
IAC T1000;                    " 加速時間 1000 ms"
IDC T1000;                    " 減速時間 1000 ms"
MVS [A1]1000000 F600000;      "MVS②"
END;
```



插補進給速度標的軸設定功能 (+、-)

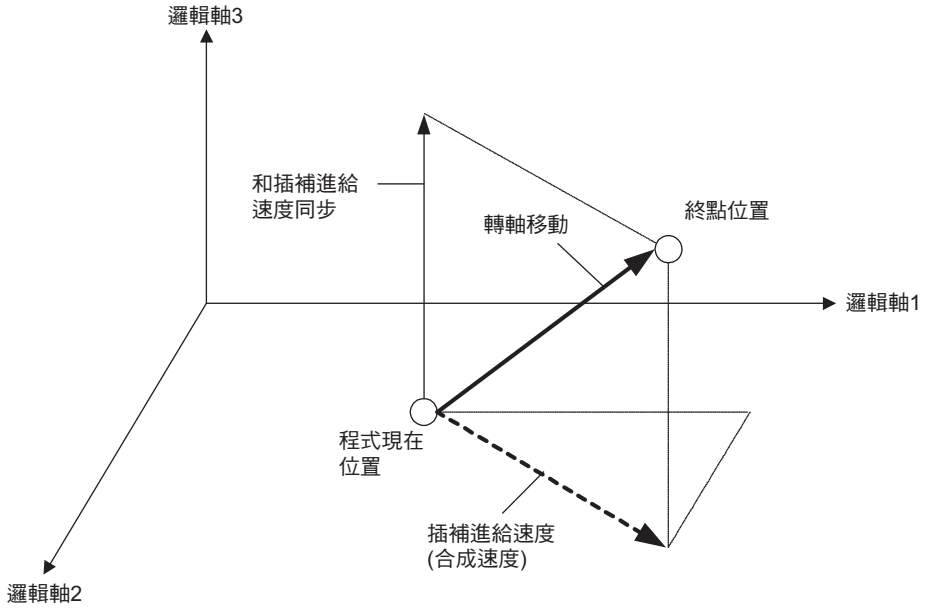
插補進給速度標的軸設定功能 (+、-) 指令是一項使用者可任意指定插補進給速度組成軸的功能。

本指令僅適用於插補指令 (MVS、SKP、MCW、MCC (螺旋內插) 執行狀態。

請將邏輯軸名稱的起始位置編寫「+」，或是完全不編寫，即可指定插補進給速度的組成軸的對象。

若該軸的邏輯軸名稱起始位置被編寫了「-」，表示將依照插補進給速度同步的速度來執行動作。

下圖係對 3 個軸執行線性內插時，假設邏輯軸 1、2 為插補進給速度標的軸設定功能的對象 (加上「+」號)，邏輯軸 3 並非為插補進給速度的對象 (加上「-」號) 時之線性內插動作圖。



補充

一般來說，當多個軸在進行插補動作時，若希望特定軸能維持一定的速度，就必須計算插補進給速度 (合成速度)。

但若使用本功能，則不需計算也能讓特定軸維持一定的速度，並進行同步控制。

主軸 (+) 將依照您所設定的插補進給速度 (合成速度) 來執行動作，而從屬軸 (-) 則依照不同的移動量所對應之速度，和主軸進行同步動作。

格式

以下為插補進給速度標的軸設定功能之格式。

・使用 MVS 指令時之格式
MVS [(+) 邏輯軸名稱 1] 指令位置　[(+) 邏輯軸名稱 2] 指令位置　[- 邏輯軸名稱 3] 指令位置 …F 插補進給速度；

・使用 SKP 指令時之格式
SKP [(+) 邏輯軸名稱 1] 指令位置　[(+) 邏輯軸名稱 2] 指令位置　[- 邏輯軸名稱 3] 指令位置 …F 插補進給速度　SS 略過輸入訊號；

・使用 MCW、MCC (螺旋內插 < 指定中心位置 >) 指令時之格式
MCW(MCC) [邏輯軸名稱1]終點位置 [邏輯軸名稱2]終點位置 U中心位置 V中心位置 [-邏輯軸名稱3]線性內插終點位置 T 轉數 F 插補進給速度；

・使用 MCW、MCC (螺旋內插 < 指定半徑 >) 指令時之格式
MCW(MCC) [邏輯軸名稱1]終點位置 [邏輯軸名稱2]終點位置 R半徑 [-邏輯軸名稱3]線性內插終點位置 F插補進給速度；

項目	單位	適用資料
指令位置	指令單位	・利用立即值直接指定 ・利用長整數型暫存器間接指定
插補進給速度	指令單位 /min	
略過輸入訊號	–	
終點位置	指令單位	
中心位置	指令單位	
線性內插的終點位置	指令單位	
轉數	次數	
半徑	指令單位	

重要

MVS 指令或 SKP 指令注意事項
・ 若在所有邏輯軸名稱的起始位置編寫 「+」號，將執行一般的插補動作。
・ 若在所有邏輯軸名稱的起始位置編寫 「-」號，將發生編譯器錯誤。
・ 在部分設定條件下，若該轉軸並非為插補進給速度標的軸設定功能，將有可能發生轉軸速度超過 FMX 指令設定值的情形。為了安全考量，使用本指令前，必須先設定好 IFMX 指令。
・ 若該轉軸為插補進給速度標的軸設定功能，此時只要該轉軸的合成移動量為 0，運動程式就會發出警報，且轉軸不執行動作。

重要

MCW、MCC (螺旋內插) 指令注意事項
・ 使用 MCW、MCC (螺旋內插) 指令時，只要在線性內插軸的邏輯軸名稱起始位置編寫 「+」號，就會執行一般的螺旋內插動作。
・ 若圓弧製作軸被指定為 「+」或 「-」，就會發生編譯器錯誤。
・ MCW、MCC (循環內插 < 指定半徑 > 或循環內插 < 指定中心位置 >) 指令不適用於本功能。
・ 在部分設定條件下，若該轉軸並非為插補進給速度軸設定功能，將有可能發生轉軸速度超過 FMX 指令設定值的情形。為了安全考量，使用本指令前，必須先設定好 IFMX 指令。
・ 若該轉軸為插補進給速度標的軸設定功能，此時只要該轉軸的合成移動量為 0，運動程式就會發出警報，且轉軸不執行動作。

程式範例

以下為插補進給速度標的軸設定功能之程式範例。

◆ 指定為 MVS 指令時

```
INC;                                          " 增量模式 "
FMX T1000000;                                 " 設定插補進給最高速度 "
IAC T100;                                     " 插補加速時間 =100 ms"
IDC T100;                                     " 插補減速時間 =100 ms"
MVS [+A1]10000 [-B1]20000 [-C1]30000 F1000000;  內置插補進給速度標的軸設定功能之線性內插 "
END;
```

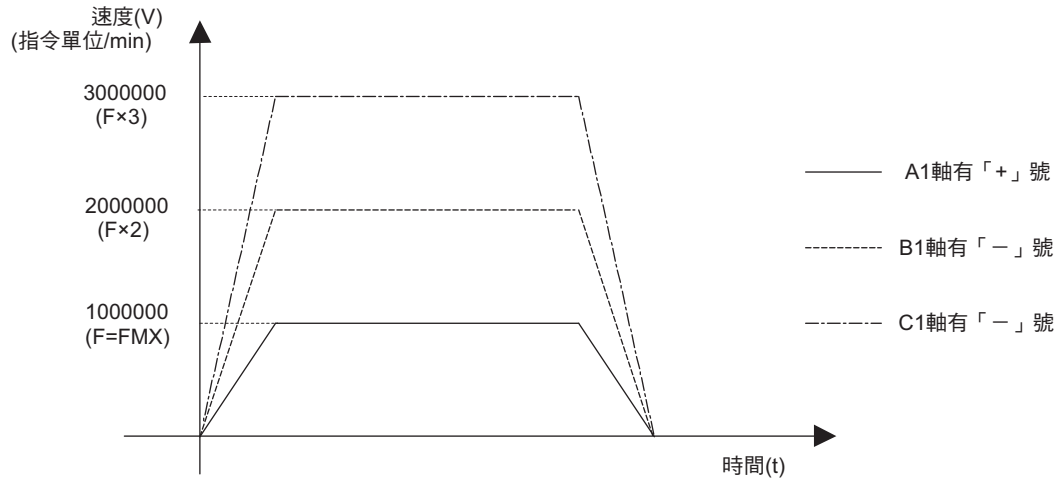


圖 6.27 插補進給速度標的軸設定功能開啟時之 MVS 指令程式範例

◆ 指定為 MCW、MCC (螺旋內插) 指令時

```
ABS;                                          " 絕對模式 "
FMX T30000000;                                " 設定插補進給最高速度 "
PLN [A1][B1];                                 指定座標平面 "
MCC [A1]1000 [B1]0 R1000 [-C1]500 F2000;      " 內置插補進給速度標的軸設定功能之螺旋內插 "
END;
```

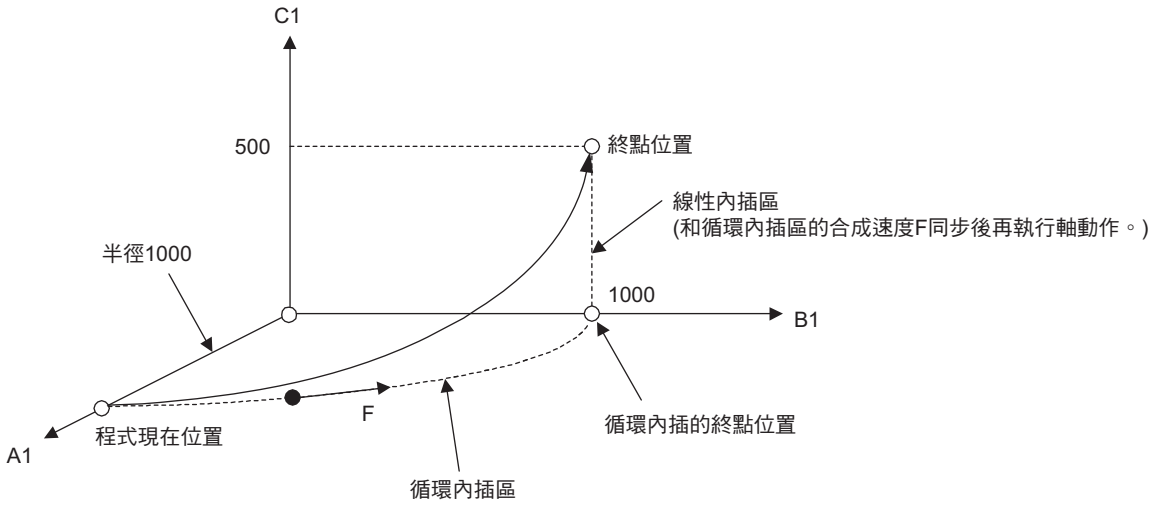


圖 6.28 插補進給速度標的軸設定功能開啟狀態下之 MCW、MCC (螺旋內插) 指令程式範例

設定插補加減速模式 (ACCMODE)

設定插補加減速模式 (ACCMODE) 是一項用來設定下述插補指令的加減速模式的指令。
使用 ACCMODE 指令，可將連續的插補指令速度互相連結。

- 線性內插 (MVS)
- 循環內插 (MCW、MCC)
- 螺旋內插 (MCW、MCC)
- 附略過功能線性內插 (SKP)

設定的插補加減速模式皆維持不變，直到 ACCMODE 指令的設定變更為止。

程式開始運轉後，插補加減速模式將變為預設模式 (插補加減速模式為 0)。

補充

1. 無法變更位於連續插補區塊之間的插補加減速模式。
   請務必在停止減速後才變更插補加減速模式。
2. 當插補加減速模式超過設定範圍時，不同的 CPU 版本將會出現不同的動作。

軟體版本	MPE720 (Ver 7.24) 以後	MPE720 (Ver 7.23) 以前
CPU 版本 (Ver 1.09) 以後	執行插補指令時，運動程式將發出「31h：位址 M 超出範圍」的警報。	執行插補指令時，運動程式將發出「31h：尚未登錄完成」的警報。
CPU 版本 (Ver 1.08) 以前	執行插補指令時，系統不會發出警報。 並且繼續維持目前的插補加減速模式。	

3. 使用 PFORK 指令時，分歧前的插補加減速模式設定將會被引用到所有的行。分歧後，可並列獨立設定插補加減速模式。

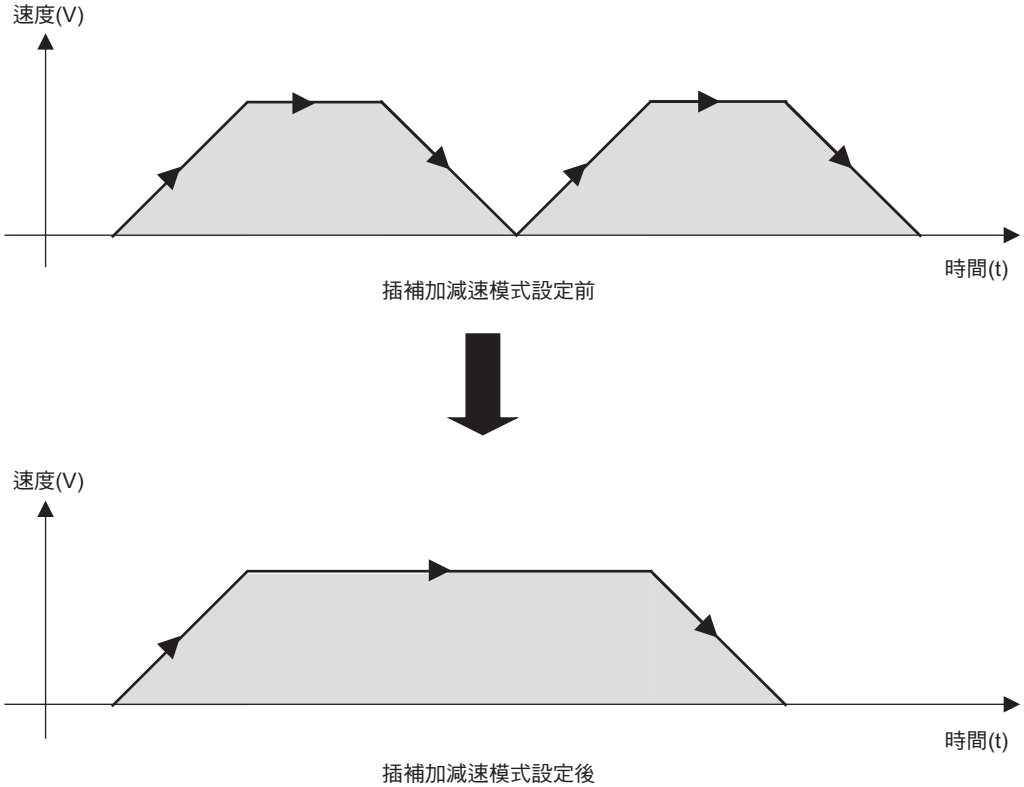


圖 6.29 設定插補加減速模式

格式

以下為 ACCMODE 指令的格式。

ACCMODE　M 插補加減速模式；

項目	單位	適用的資料
插補加減速模式	–	利用立即值直接指定 (0~4)

ACCMODE 設定項目

接下來將說明 ACCMODE 指令的設定項目。

插補加減速模式是一項利用 ACCMODE 指令來指定接在字元「M」後面數值的指令。

以下 5 種插補加減速模式可供選擇。

- 預設模式 (插補加減速模式 0)
- 附連結處理控制訊號監控功能加減速模式 (插補加減速模式 1)
- 附插補重疊功能加減速模式 (插補加減速模式 2)
- 附連結處理控制訊號監控功能加減速模式 (內置插補加減速模式 3、微小區塊減速連結功能)
- 附下個區塊速度指定的加減速模式 (插補加減速模式 4)

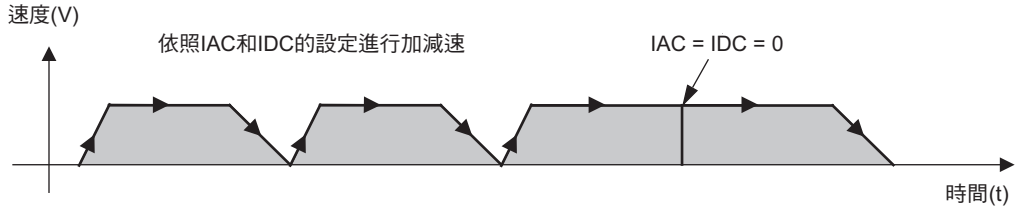
ACCMODE 說明

接下來將說明 ACCMODE 指令的 5 種插補加減速模式。

◆ 預設模式 (插補加減速模式 0) 說明

此模式將依照 IAC 指令及 IDC 指令所設定的加減速時間來執行加減速動作。

當程式開始運轉後，就會進入預設模式。



■ 格式

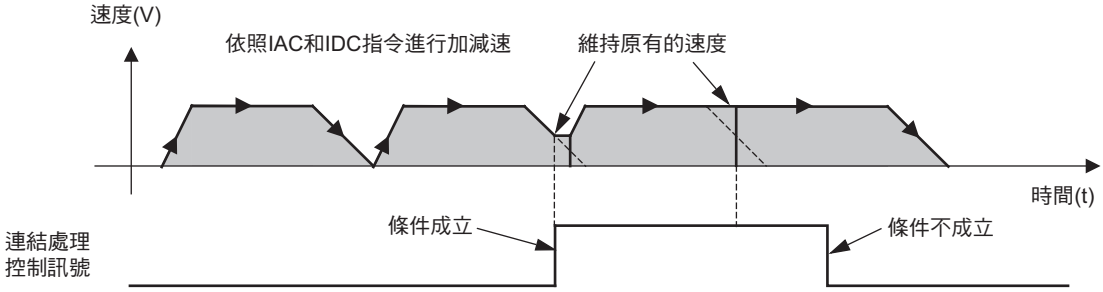
選擇插補加減速模式 0 時之格式如下。

ACCMODE M0;

◆ 附連結處理控制訊號監控功能加減速模式 (插補加減速模式 1) 之說明

此模式可用來監控連結處理控制訊號，一旦條件成立，就會在連續的插補區塊之間進行連結處理作業。

本模式僅適用於所有連續的插補區塊，且使用相同的轉軸的條件下。



■ 格式

選擇插補加減速模式 1 時之格式如下。

ACCMODE M1;
MVS [邏輯軸名稱 1] 指令位置 F 插補進給速度 TW 連結處理控制訊號；

或是

ACCMODE M1;
MVS [邏輯軸名稱 1] 指令位置 F 插補進給速度 FW 連結處理控制訊號；

項目	單位	適用的資料
連結處理控制訊號	–	所有位元型暫存器 (#、C、D 暫存器除外)

(註) 使用 MCC、MCW、SKP 指令時的格式亦同。

插補指令只要加上字元 TW 或 FW，即可用來監控 「連結處理控制訊號」。字元 TW 或 FW 所指定的位元型暫存器將作為 「連結處理控制訊號」之用。

若插補指令未加上字元 TW 或 FW，或者是條件不成立，將依照 IAC 指令和 IDC 指令所設定的加減速時間來進行加減速，且不執行 「連結處理控制訊號」監控。

補充

字元 TW 或 FW 僅適用於附連結處理控制訊號監控功能加減速模式 (插補加減速模式 1 及模式 3)。用於其他模式時，部分 CPU 軟體版本將出現不同的動作。

軟體版本	MPE720 (Ver 7.24) 以後版本	MPE720 (Ver 7.23) 以前版本
CPU 版本 (Ver 1.09) 以後	執行插補指令時，運動程式就會發出「32h：指定位址錯誤」的警報。	執行插補指令時，運動程式將發出「32h：尚未登錄完成」的警報。
CPU 版本 (Ver 1.08) 以前	執行插補指令時，系統將不會發出警報。 此時，位址 TW/FW 會被忽略，且執行插補指令。	

字元 TW 將使用正邏輯來監控連結處理控制訊號。

連結處理控制訊號	動作概述
開啟	忽略 IDC 指令所指定的減速時間，並在維持原有速度的條件下，以減速時間 =0 ms 完成送出指令動作。
關閉	依照 IDC 指令所指定的減速時間進行減速。

字元 FW 將使用負邏輯來監控連結處理控制訊號。

連結處理控制訊號	動作概述
開啟	依照 IDC 指令所指定的減速時間進行減速。
關閉	忽略 IDC 指令所指定的減速時間，並在維持原有速度的條件下，以減速時間 =0 ms 完成送出指令動作。

重要

若指定的移動量無法滿足您設定的減速時間，而執行連結處理時，系統將發生無法預期的動作，因此使用時必須確保充裕的移動量。

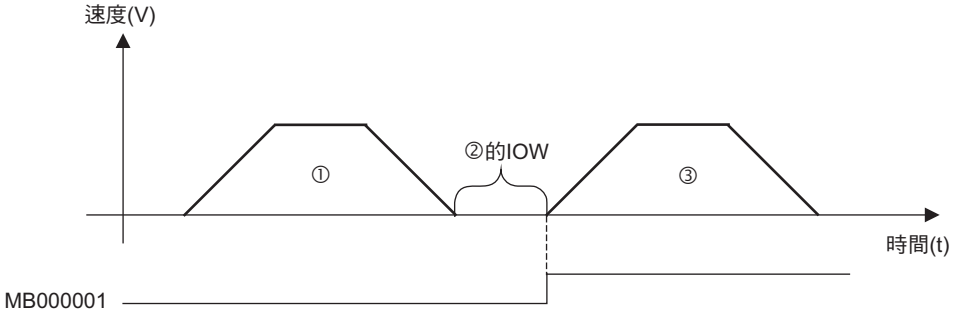
■ 程式範例

以下為附連結處理控制訊號監控功能加減速模式 (插補加減速模式 1) 之程式範例。

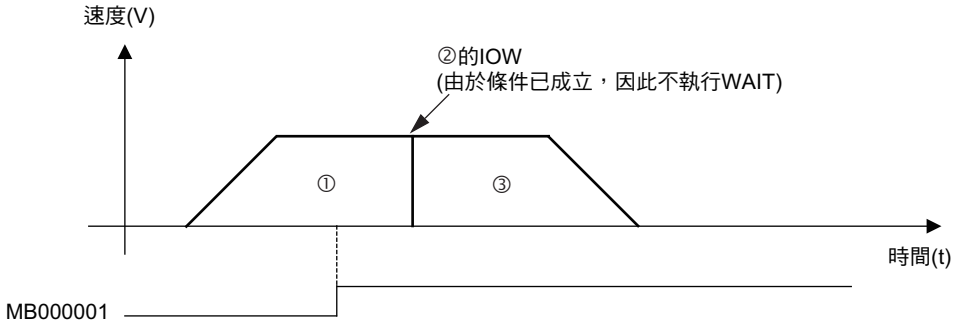
```
FMX T30000000;
ABS;
IAC T1000;
IDC T1000;

ACCMODE M1;

MVS [A1] 4000 F50000 TWMB000001;   "①"
IOW MB000001 = = 1;                "②"
MVS [A1] 8000;                     "③"

END;
```

接下來將說明 MVS 指令與插補加減速模式 1 互相搭配時之範例。

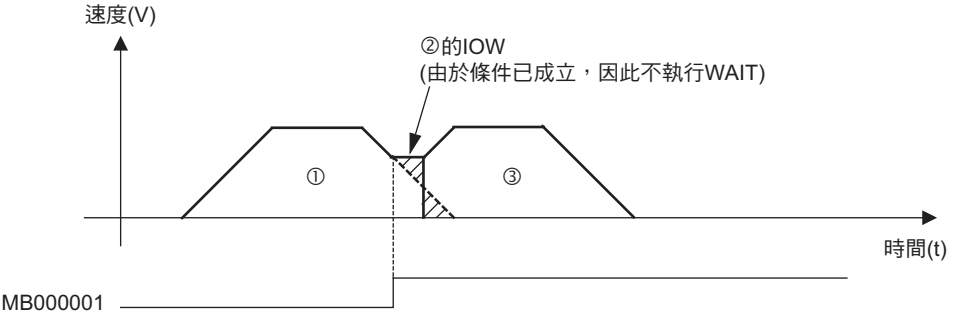
・**當 ① 的 MVS 指令完成送出指令動作後，連結處理控制訊號變為開啟時**
當 ① 的 MVS 指令停止減速後，執行以下區塊。
執行 ③ 的 MVS 時，從速度 =0 (指令單位 /min) 開始加速。



・**當 ① 的 MVS 進行送出指令時 (開始減速前)，連結處理訊號變為 ON 時**
在不減速的條件下，維持 ① 的 MVS 速度，並執行 ③ 的 MVS。



・**當 ① 的 MVS 進行送出指令時 (減速狀態)，連結處理控制訊號變為開啟時**
在維持連結處理控制訊號開啟的速度條件下，執行 ③ 的 MVS。



補充

1. 若 ③ 的 MVS 指令速度大於 ① 的 MVS，就會以 ① 的結束速度作為 ③ 的開始速度，並且持續加速直到到達您所指定的速度為止。
2. 若 ③ 的 MVS 指令速度小於 ① 的 MVS，就會以 ① 的結束速度作為 ③ 的開始速度，並且持續減速直到到達您所指定的速度為止。
3. ③ 的 MVS 移動距離小於減速距離時，③ 就會在減速過程中完成送出指令動作。

■ 補充事項

以下為附連結處理控制訊號監控功能加減速模式動作時之相關補充事項。

- **要求暫停時的動作**
  開始進行下一個區塊的插補送出指令前，收到要求暫停訊號
  依照 IDH 指令所指定的插補減速時間，進行減速。
  將不會到下一個插補區塊進行連結處理。

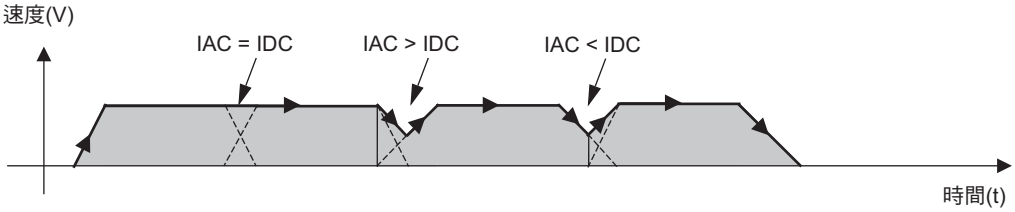
  開始進行下一個區塊的插補送出指令後，收到要求暫停訊號
  無論上一個或下一個區塊，皆會依照 IDH 指令所指定的插補減速時間來減速。
  當要求暫停的訊號被解除後，無論上一個或下一個區塊，皆會在剩餘的距離範圍內進行送出指令動作。
- **要求停止時的動作**
  轉軸移動中的插補區塊將立刻停止動作。
- **程式單一區塊運轉模式下的動作**
  將不會到下一個插補區塊進行連結處理。
- **除錯運轉模式下的動作**
  將不會到下一個插補區塊進行連結處理。
- **下一個區塊使用非插補指令時的動作**
  將不會進入下一個區塊進行連結處理。
  當下一個區塊進入停止狀態後，即開始加速。
- **插補減速時間 (IDC) 被設定為 0 ms 時的動作**
  無論連結處理控制訊號的狀態為何，將會進入下一個插補區塊進行連結處理。
- **並列執行 (PFORK) 時的連結動作**
  跨越 PFORK 指令時，將不會執行連結處理。
  請設定為在各項並列中完成本模式的處理程序。

◆ 附插補重疊功能加減速模式 (插補加減速模式 2) 說明

附插補重疊功能加減速模式 (插補加減速模式 2) 係在插補區塊減速時，讓下一個插補區塊開始加速，如此就能讓每個插補區塊的送出指令作業互相重疊，並以連續方式進行插補區塊之間的連結處理。

插補區塊係利用 IAC 指令及 IDC 指令所設定的加減速時間，來進行加減速。

本模式僅適用於 MVS 指令及 MCW/MCC 指令。



■ 格式

ACCMODE M2;
MVS [邏輯軸名稱 1] 指令位置 F 插補進給速度 D 插補重疊距離；

項目	單位	適用的資料
插補重疊距離	指令單位	・利用立即值直接指定 ・利用長整數型暫存器間接指定

(註) 插補重疊距離可省略。
使用 MCC、MCW 指令時的格式亦同。

附插補重疊功能加減速模式係利用將字元 D 附加在插補指令，以便指定插補送出指令重疊距離的最大值。

若在插補指令加上字元 D，這時候只要插補區塊的剩餘移動距離小於插補重疊距離，接著就會開始下一個插補區塊的送出指令。當插補重疊距離為 0 (指令單位) 時，插補區塊會先減速，再開始下一個插補區塊的送出指令作業。

若插補指令未加上字元 D，將依照前一次運動程式所指定的插補重疊距離來執行動作。

程式開始運轉後，插補重疊距離將變為 0 (指令單位)。

補充

1. 字元 D 僅適用於本模式。
CPU 版本不同，其他插補加減速模式下的動作亦各異。

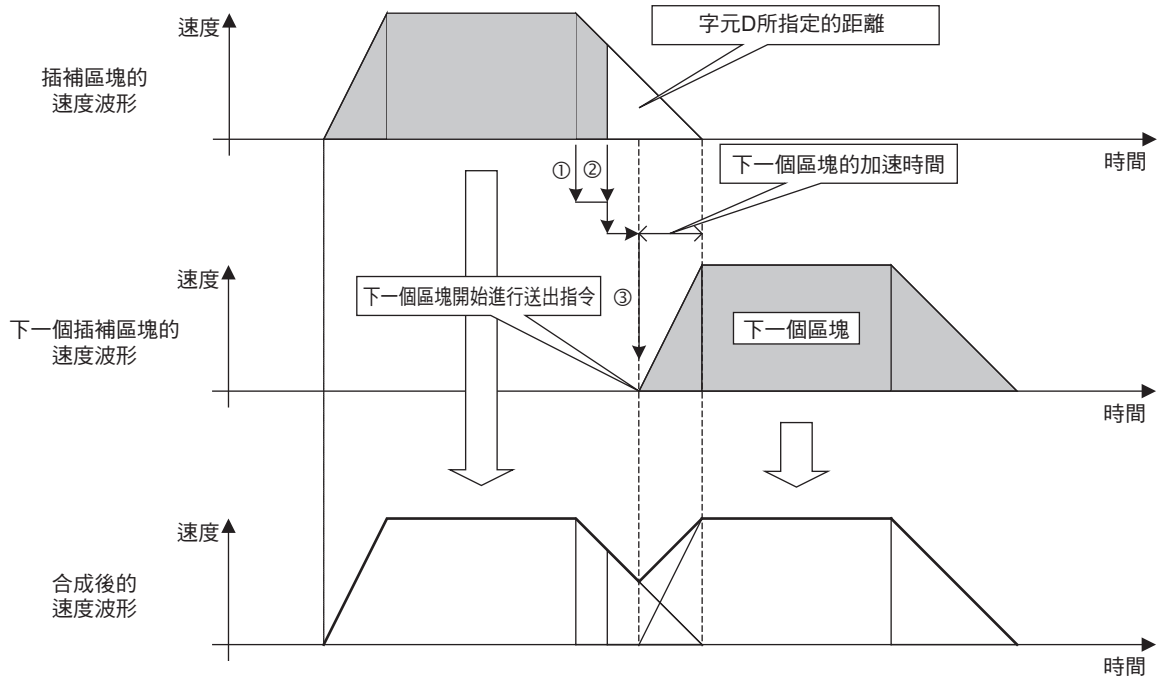
軟體版本	MPE720 (Ver 7.24) 以後版本	MPE720 (Ver 7.23) 以前版本
CPU 版本 (Ver 1.09) 以後	執行插補指令時，運動程式就會發出「32h：指定位址錯誤」的警報。	執行插補指令時，運動程式將發出「32h：尚未登錄完成」的警報。
CPU 版本 (Ver 1.08) 以前	執行插補指令時，系統將不會發出警報。 此時，位址 D 會被忽略，且執行插補指令。	

2. 插補重疊距離的指令範圍為 0~2147483647 (指令單位)。
設定為負值時，系統將依照絕對值動作。

■ 下一個插補區塊的開始送出指令條件

只要符合以下條件，下一個插補區塊就會開始執行送出指令動作。

No.	條件
1	程式單一區塊並非處於運轉模式。
2	控制訊號 (Bit 1：要求暫停) 變為關閉。
3	插補類指令並未加上 PFN 指令及 PFP 指令。
4	插補區塊由定速狀態進入減速狀態後。(下圖 ① 所示的時間點)
5	插補區塊的剩餘距離需小於字元 D 所指定的插補重疊距離。(下圖 ② 所示的時間點)
6	插補區塊的剩餘減速時間需小於下一個插補區塊的加速時間。(下圖 ③ 所示的時間點)



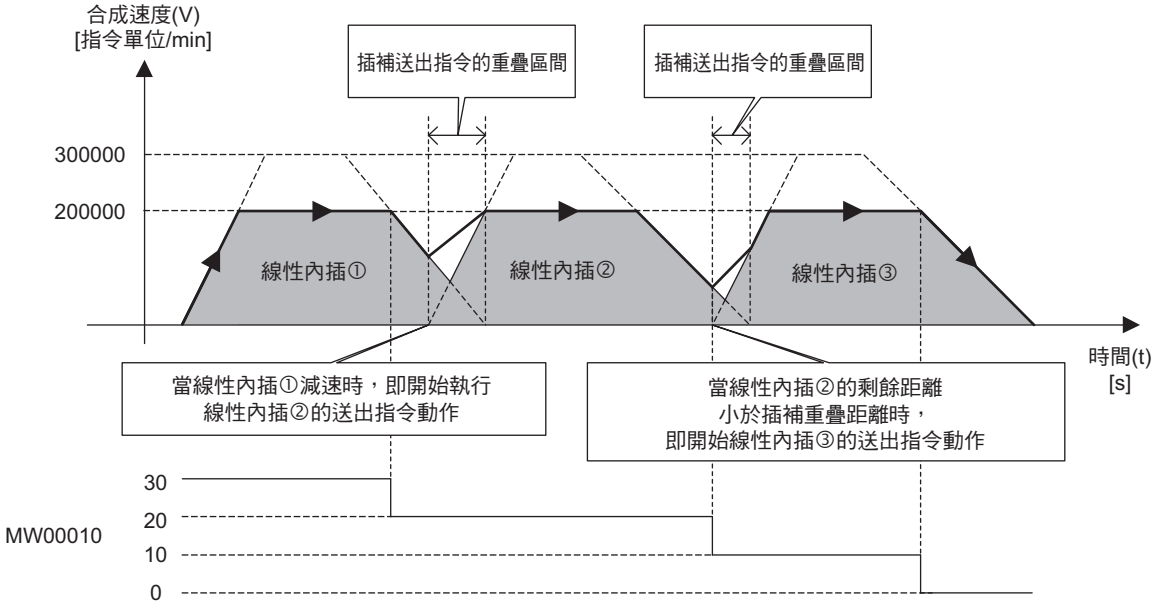
■ 程式範例

以下為附插補重疊功能加減速模式下之程式範例。

```
FMX T300000;
INC;
IAC T1000;
IDC T2000;

ACCMODE M2;

MW00010 = 30;
MVS [A1] 20000  [B1] 10000 F200000;          " 線性內插 ①"
MW00010 = 20;
MVS [A1] 10000  [B1] -20000 D100;            " 線性內插 ②"
MW00010 = 10;
MVS [A1] 20000  [B1] 10000;                  " 線性內插 ③"
MW00010 = 0;

END;
```

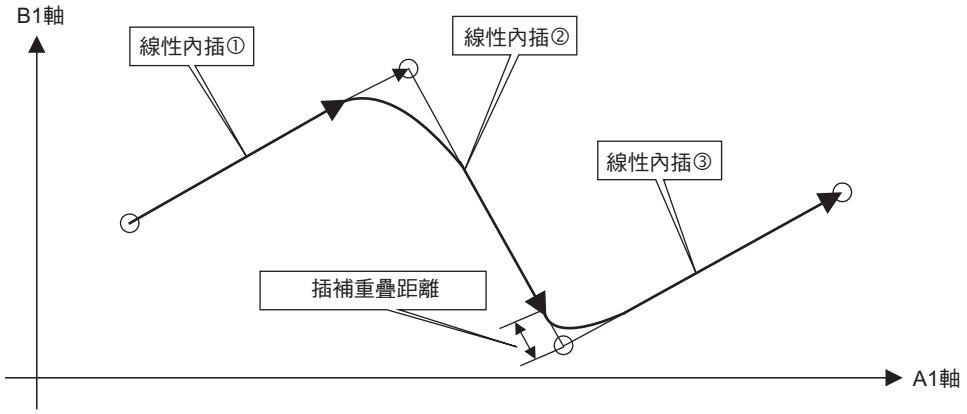
當插補加減速模式 2 在進行處理作業時，只要插補區塊進入減速狀態，或是小於您所設定的插補重疊距離，程式的執行區塊就會變化。
當程式的執行區塊變化的時間點，就會開始執行插補區塊連結處理中的 S 型指令 (運算指令等)。

下圖為程式範例的速度波形。



下圖為上述程式範例的插補軌跡。

當插補區塊減速時，下一個插補區塊就會開始執行送出指令動作，因此如下圖所示，軌跡並不會通過插補區塊的終點 (= 下一個插補區塊的起點)。



■ 補充事項

以下為附插補重疊功能加減速模式動作時之相關補充事項。

- **要求暫停時的動作**
  當下一個插補區塊開始執行送出指令動作時，轉軸移動狀態下的插補區塊將依照 IDH 指令所指定的減速時間來減速。
  但是，並不會進入下一個插補區塊進行連結處理。
  當下一個插補區塊開始執行送出指令動作時，每個插補區塊皆會依照 IDH 指令所指定的減速時間來減速。
- **要求停止時的動作**
  所有插補區塊皆立刻停止動作。
- **程式單一區塊運轉模式下的動作**
  不會到下一個插補區塊進行連結處理。
- **除錯運轉模式下的動作**
  不會到下一個插補區塊進行連結處理。
- **下一個區塊使用非插補指令時的動作**
  不會進入下一個區塊進行連結處理。
  當下一個區塊進入停止狀態後，即開始加速。
- **並列執行 (PFORK) 時的連結動作**
  如要跨越 PFORK 指令，則本模式不適用。
  請利用 PFN 指令或 PFP 指令等來調整插補區塊的送出指令時間點，如此就能讓所有的並列中的本模式 (送出指令) 的處理作業完成。
- **執行 T 型指令時的動作**
  若在插補區塊進行連結處理時，同時執行 T 型指令 (計時器指令等)，就會造成下一個插補區塊錯過正確的送出指令時間點。

◆ 附連結處理控制訊號監控功能加減速模式 (插補加減速模式 3) 的詳細說明

附連結處理控制訊號監控功能加減速模式 (插補加減速模式 3) 和附連結處理控制訊號監控功能加減速模式 (插補加減速模式 1) 一樣，可用來監控連結處理控制訊號，一旦條件成立，就會在連續的插補區塊內進行連結處理。

與附連結處理控制訊號監控功能加減速模式 (插補加減速模式) 的差異在於，若您所設定的移動量較小，並在微小區塊內進行連結處理時，將在到達設定速度前盡量減速，以執行插補區塊內的連結處理。

補充　所謂微小區塊，即為從連結處理動作中的速度，根據指定的減速度減速停止到 0 速時所需的距離不足的移動量的小插補區塊。

■ 格式

以下為設定附連結處理控制訊號監控功能加減速模式 (插補加減速模式 3) 時之格式。如欲進一步瞭解連結處理控制訊號，請參閱以下章節。

☞ ◆ 附連結處理控制訊號監控功能加減速模式 (插補加減速模式 1) 之說明 (第 6-62 頁)

```
ACCMODE 3;
MVS  [ 邏輯軸名稱 1] 指令位置 …F 插補進給速度  TW 連結處理控制訊號；

或是

ACCMODE 3;
MVS  [ 邏輯軸名稱 1] 指令位置 …F 插補進給速度  FW 連結處理控制訊號；
```

(註)使用 MCC、MCW、SKP 指令時的格式亦同。

補充　字元 TW、FW 僅適用於附連結處理控制訊號監控加減速模式 (插補加減速模式 1) 及本模式。在其他模式下指定字元 TW、FW 時，不同的 CPU 版本將會出現不同的動作。

軟體版本	MPE720 (Ver 7.24) 以後版本	MPE720 (Ver 7.23) 以前版本
CPU 版本 (Ver 1.09) 以後	執行插補指令時，發生「32h：指定位址錯誤」。	執行插補指令時，發生「32h：未登錄完成警報」。
CPU 版本 (Ver 1.08) 以前	執行插補指令時，系統將不會發出警報。 此時，位址 TW、FW 會被忽略，且執行插補指令。	

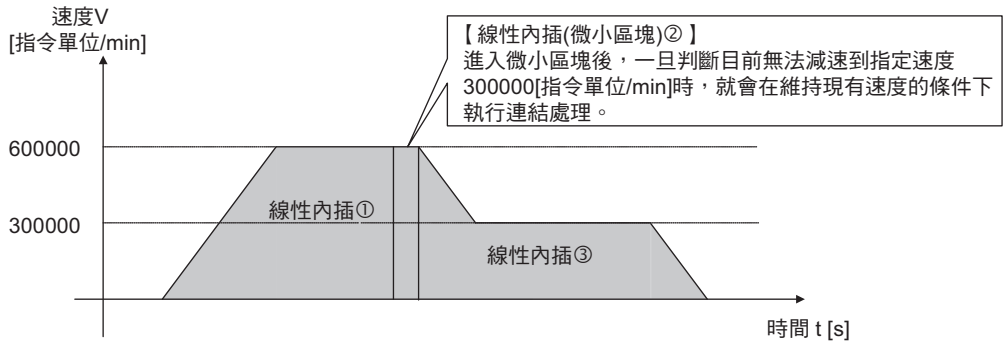
■ 程式範例

接下來將說明 MVS 指令執行插補加減速模式 1 和插補加減速模式 3 時之動作差異。

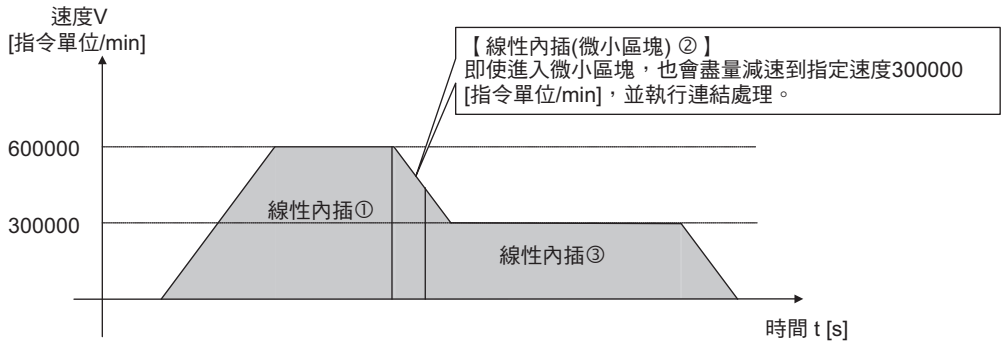
```
INC;
FMX T1000000;
IAC T5000;
IDC T5000;
MB1000=1;                  // 連結用控制訊號位元
// 插補加減速模式 ( 利用 ACCMODE M1 和 M3 來執行 )
ACCMODE M1;                                  // ACCMODE M1  or  ACCMODE M3
MVS [A1]100000 F600000 TWMB1000;             // 線性內插 ①
MVS [A1]5000 F300000 TWMB1000;               // 線性內插 ( 微小區塊 )②
MVS [A1]100000 F300000 FWMB1000;             // 線性內插 ③

END;
```

下圖為程式範例的速度波形。

・利用插補加減速模式 1 執行動作



・利用插補加減速模式 3 執行動作



■ 補充事項

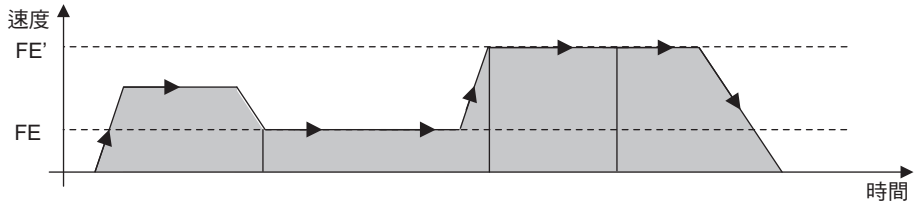
如欲瞭解附連結處理控制訊號監控功能加減速模式 (插補加減速模式 3) 的相關補充事項，請參閱以下章節。

☞ ■ 補充事項 (第 6-65 頁)

◆ 附下個區塊速度指定功能加減速模式 (插補加減速模式 4) 說明

附下個區塊速度指定功能加減速模式 (插補加減速模式 4) 可用來指定每個插補區塊的最終速度，並且依照您所指定的速度，進入下一個插補區塊進行連結處理。

本模式僅適用於所有連續的插補區塊，且使用相同的轉軸的條件下。



■ 格式

設定插補加減速模式 4 時之格式如下。

ACCMODE 4;
MVS [邏輯軸名稱 1] 指令位置 …F 插補進給速度 FE 插補進給最終速度；

項目	單位	適用的資料
插補進給最終速度	符合 FUT 設定單位之指令單位 /min 或指令單位 /s	・利用長整數型暫存器間接指定 ・利用立即值直接指定

(註) 插補進給最終速度可省略不設定。使用 MCC、MCW、SKP 指令時的格式亦同。

附下個區塊速度指定功能加減速模式可在插補指令中加上字元 FE，以指定插補區塊的最終速度。

當插補指令被加上字元 FE 後，就會調整送出指令動作，以便能依照您所指定的插補進給最終速度，來結束插補區塊作業。

當插補進給最終速度為 0 (速度單位) 時，將停止減速，且不會執行連結處理。

若插補指令未被加上字元 FE，就會依照上一次運動程式所指定的插補進給最終速度來執行動作。

當程式開始運轉後，插補進給最終速度就會變為 0 (速度單位)。

重要

1. 字元 FE 僅適用於本模式。在其他插補加減速模式下，運動程式就會發出警報。
2. 插補進給最終速度的指令範圍為 0~2147483647 (速度單位)。若設定為負值時，將會發生編譯器錯誤。

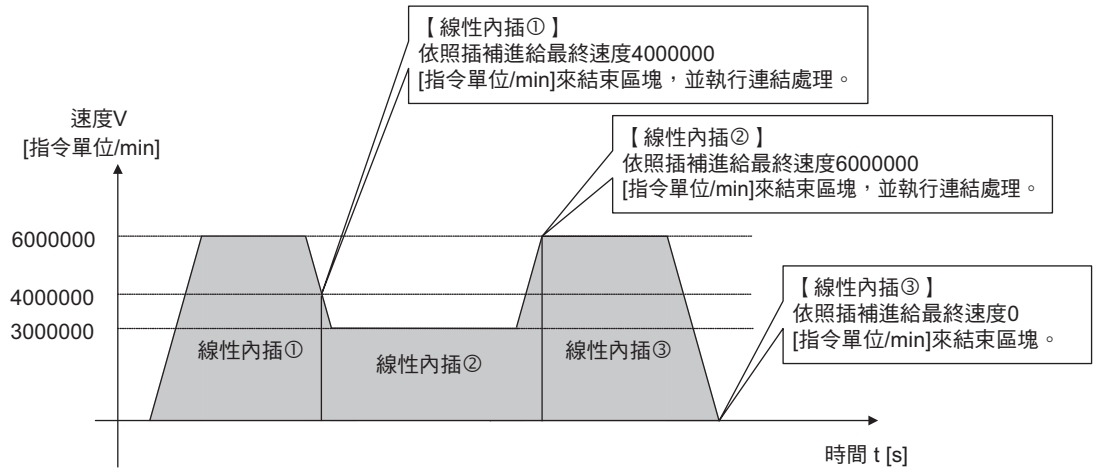
補充

本模式適用於以下的版本組合。
・ CPU 版本：Ver 1.09
・ MPE720 Ver 7 版本：Ver 7.24

■ **程式範例**

```
FMX T6000000;
IAC T1000;
IDC T1000;
INC;
ACCMODE M4;
MVS [A1]300000 F6000000 FE4000000;          " 線性內插 ①"
MVS [A1]300000 F3000000 FE6000000;          " 線性內插 ②"
MVS [A1]300000 F6000000 FE0;                " 線性內插 ③"

END;
```

下圖為程式範例的速度波形。



■ **補充事項**

以下為附下個區塊速度指定功能加減速模式動作時之相關補充事項。

- **要求暫停時的動作**
  一旦要求暫停的訊號出現時，將依照您所設定的減速停止至 0 速。若移動量不足，系統就會在到達目標位置時緊急停止。
- **要求停止時的動作**
  插補區塊立刻停止動作。
- **未到達插補進給最終速度時的動作**
  持續加速 (減速) 直到到達插補進給最終速度為止，並且在移動量變為 0 時立刻停止動作。
  若下個區塊為插補指令，就會在立刻停止時執行連結處理。
- **程式單一區塊運轉模式下的動作**
  進入下個插補區塊時，將不執行連結處理。
- **除錯運轉模式下的動作**
  進入下個插補區塊時，將不執行連結處理。
- **下一個區塊使用非插補指令時的動作**
  進入下個區塊時，將不執行連結處理。
  下個區塊會從「0」開始加速。
- **並列執行 (PFORK) 時的連結動作**
  跨越 PFORK 指令時，將不執行連結處理。
  請設定為在各項並列中完成本模式的處理程序。

6.2 軸移動指令

軸移動指令可用來移動和運動控制功能互相連接的轉軸。
軸移動指令包含 11 種指令，而且僅適用於運動程式。
以下為軸移動指令一覽表。

指令	名稱	格式		內容	運動程式	序列程式
MOV	定位	MOV [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 [邏輯軸名稱 3] 指令位置・・・;		利用定位速度功能，最多可同時對 32 個轉軸進行定位。	○	×
MVS	線性內插	MVS [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 [邏輯軸名稱 3] 指令位置・・・ F 插補進給速度;		利用插補進給速度 F，最多可同時讓 32 個轉軸線性移動。	○	×
MCW	循環內插 (順時鐘方向)	指定中心位置時	MCW [邏輯軸名稱 1] 終點位置 [邏輯軸名稱 2] 終點位置 U 中心位置 V 中心位置 T 轉數 F 插補進給速度;	利用接線速度 F，即可沿著半徑 R (或指定的中心座標) 的圓弧，同時對 2 軸進行插補。 指定中心座標時，利用 T- 即可指定多個圓周。(T － 亦可省略)	○	×
		指定半徑時	MCW [邏輯軸名稱 1] 終點位置 [邏輯軸名稱 2] 終點位置 R 半徑 F 插補進給速度;			
MCC	循環內插 (逆時鐘方向)	指定中心位置時	MCC [邏輯軸名稱 1] 終點位置 [邏輯軸名稱 2] 終點位置 U 中心位置 V 中心位置 T 轉數 F 插補進給速度;			
		指定半徑時	MCC [邏輯軸名稱 1] 終點位置 [邏輯軸名稱 2] 終點位置 R 半徑 F 插補進給速度;			

(續下頁)

(接上頁)

指令	名稱	格式		內容	運動程式	序列程式
MCW	螺旋內插 (順時鐘方向)	指定中心位置時	MCW [邏輯軸名稱 1] 終點位置 [邏輯軸名稱 2] 終點位置 U 中心位置　V 中心位置 [邏輯軸名稱 3] 線性內插的終點位置 T 轉數　F 插補進給速度；	可將循環內插和循環內插平面外的線性內插互相合成，並同時讓 3 軸移動。速度 F 為 3 軸的接線速度。 指定中心座標時，T －可用來指定轉數。(T －亦可省略)	○	×
		指定半徑時	MCW [邏輯軸名稱 1] 終點位置 [邏輯軸名稱 2] 終點位置 R 半徑 [邏輯軸名稱 3] 線性內插的終點位置 F 插補進給速度；			
MCC	螺旋內插 (順時鐘方向)	指定中心位置時	MCC [邏輯軸名稱 1] 終點位置 [邏輯軸名稱 2] 終點位置 U 中心位置　V 中心位置 [邏輯軸名稱 3] 線性內插的終點位置 T 轉數　F 插補進給速度；			
		指定半徑時	MCC [邏輯軸名稱 1] 終點位置 [邏輯軸名稱 2] 終點位置 R 半徑 [邏輯軸名稱 3] 線性內插的終點位置 F 插補進給速度；			
ZRN	原點復歸	ZRN [邏輯軸名稱 1]0 [邏輯軸名稱 2]0 [邏輯軸名稱 3]0・・・；		復歸至各軸的原點。	○	×
DEN	送出指令完成後進行步進定位	MOV [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 [邏輯軸名稱 3] 指令位置・・・DEN；		只要完成送出指令後即進入下個區塊定位，不需等待定位完成指令。	○	×
SKP	附略過功能線性內插	SKP [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 [邏輯軸名稱 3] 指令位置・・・ F 插補進給速度 選擇 SS 略過輸入訊號；		在線性內插動作狀態下，只要略過訊號開啟，就會跳過剩餘的移動量，並進入下個區塊。	○	×
MVT	指定時間定位	MVT [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 [邏輯軸名稱 3] 指令位置・・・ T 定位時間 (ms)；		會先調整進給速度，再進行定位，以便在指定時間內完成移動動作。	○	×
EXM	外部定位	EXM [邏輯軸名稱 1] 指令位置 D 外部定位訊號輸入後之移動量；		執行定位時，只要有外部定位訊號輸入，將以 D- 所指定的移動量作為增量值並進行定位，然後再執行下一個指令。	○	×

定位 (MOV)

定位 (MOV) 指令是一項可依照定位速度，讓每個軸各自從程式現在位置移動到終點位置的指令。
本指令最多可同時移動 32 個轉軸。若省略本指令，將無法移動轉軸。
MOV 指令的移動軌跡不同於線性內插指令，並不會線性移動。

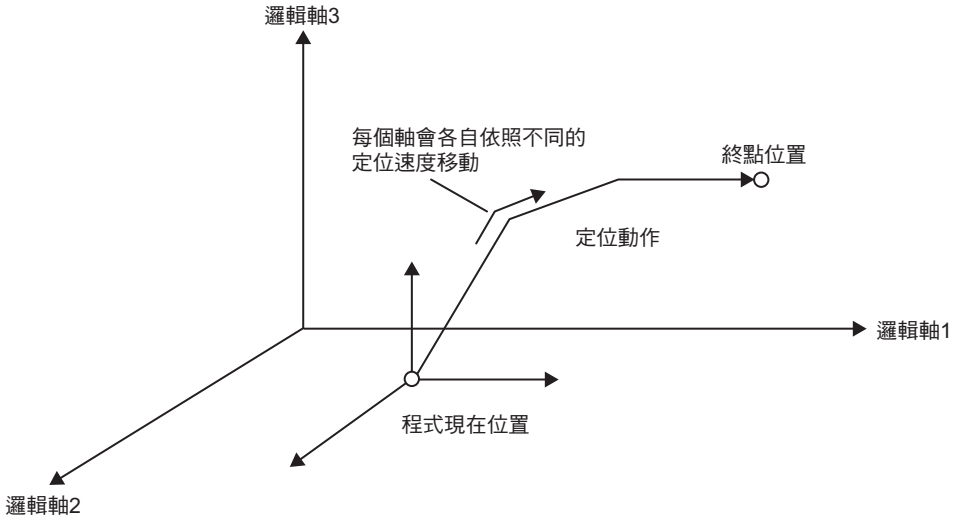


圖 6.30 MOV 指令的移動軌跡

⚠ 注意

- **使用定位 (MOV) 功能時之移動軌跡將無法保持直線。使用本功能來編寫程式前，請先確認軌跡，然後再檢查系統是否安全執行動作。**
  否則恐將造成裝置的損壞。

重要

當 MOV 指令所指定的任一個轉軸發出警報時，運動程式將發出警報並停止轉軸動作。

格式

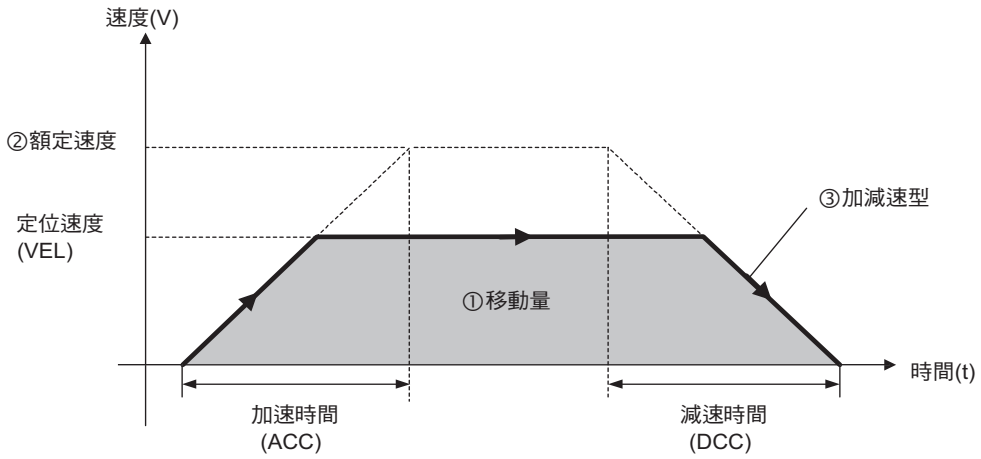
以下為 MOV 指令的格式。

MOV [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 [邏輯軸名稱 3] 指令位置・・・;

項目	單位	適用的資料
指令位置	指令單位	・利用立即值直接指定 ・利用長整數暫存器間接指定

MOV 指令的設定項目

接下來將說明 MOV 指令的設定項目。



① 移動量

每個軸的移動量在 ABS 模式或 INC 模式下各不相同。

- ABS 模式的移動量
  在 ABS 模式下，程式現在位置與指令位置之間的差值將被視為移動量。
- INC 模式的移動量
  在 INC 模式下，指令位置將直接被視為移動量。

② 額定速度

利用固定參數 No. 34 (額定轉數)，即可設定每個軸的額定速度。

③ 加減速型

MOV 指令可從以下 3 種加減速類型中選擇。
利用 ACC/DCC/SCC 指令、設定參數 OW□□□03 Bit 4~7 (選擇加減速度單位) 及 OW□□□03 Bit 8~B (選擇濾波器類型) 的組合，即可設定加減速類型。

・無加減速
此種方法在加速時間、減速時間皆為 0 的狀態下動作。

設定方法	動作範例
・OW□□□03 Bit 4~7 (選擇加減速度單位) 設定為「1：ms」 ・OW□□□03 Bit 8~B (選擇濾波器類型) 設定為「0：無濾波器」 ・ACC 指令設定為「0」 ・DCC 指令設定為「0」	

・1 段直線加減速
此種方法可依照固定的加速度、減速度執行動作。

設定方法	動作範例
・OW□□□03 Bit 4~7 (選擇加減速度單位) 設定為「1：ms」 ・OW□□□03 Bit 8~B (選擇濾波器類型) 設定為「0：無濾波器」 ・ACC 指令設定為 0 以外數值 ・DCC 指令設定為 0 以外數值	

・S 型加減速
此方法可依照 S 型加速度、減速度執行動作。

設定方法	動作範例
・OW□□□03 Bit 4~7 (選擇加減速度單位) 設定為「1：ms」 ・OW□□□03 Bit 8~B (選擇濾波器類型) 設定為「2：平均移動濾波器」 ・ACC 指令設定為 0 以外數值 ・DCC 指令設定為 0 以外數值 ・SCC 指令設定為 0 以外數值	

補充　依 MOV 指令移動轉軸後，一旦確認目前已進入定位完成範圍後，就會執行到位確認 (PFN)。完成到位確認後，就會執行下一個移動指令的區塊。
下圖為到位確認所執行的動作。

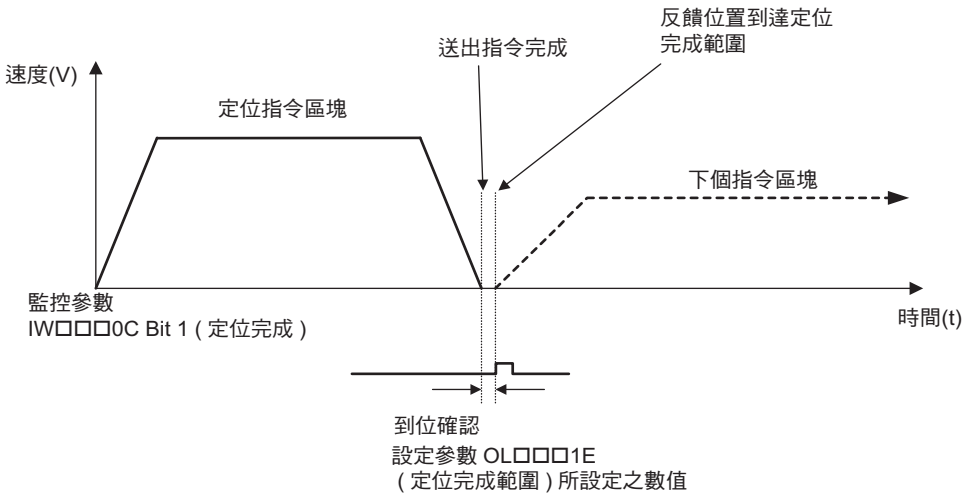


圖 6.31 到位確認動作

程式範例

以下為 ABS 模式下執行 MOV 指令之程式範例。

```
ABS;
ACC [A1]1000 [B1]1000 [C1]1000;
DCC [A1]1000 [B1]1000 [C1]1000;
VEL [A1]2000 [B1]2000 [C1]2000;
MOV [A1]4000 [B1]3000 [C1]2000;
END;
```

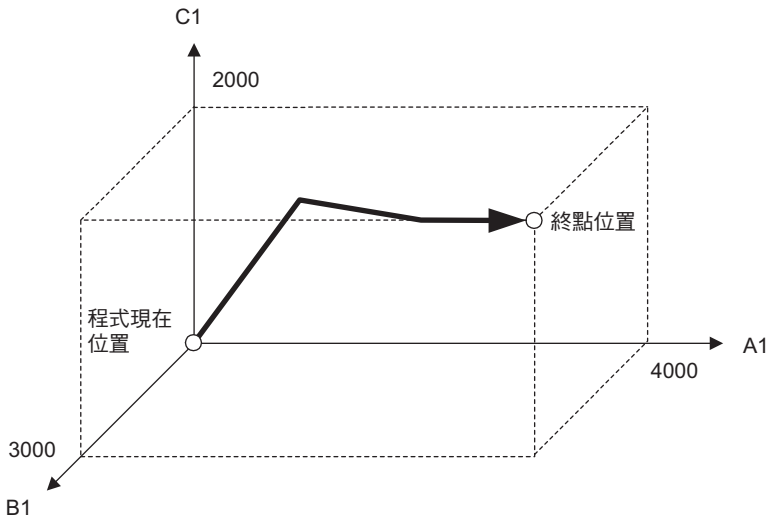


圖 6.32 MOV 指令的程式範例

線性內插 (MVS)

線性內插 (MVS) 是一項依照插補進給速度，並以直線方式將各轉軸從程式現在位置移動到終點位置的指令。本指令最多可同時移動 32 個轉軸。
省略邏輯軸格式的轉軸則不會移動。

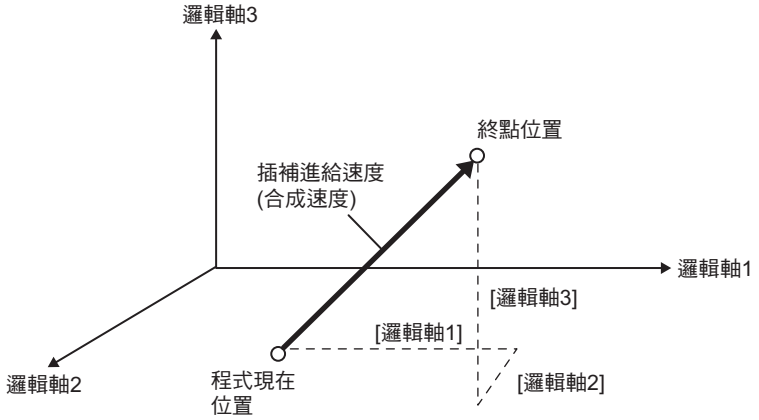


圖 6.33 MVS 指令的移動軌跡

⚠ 注意

- **線性內插 (MVS) 同時適用於直線軸或旋轉軸。但是當本功能用於旋轉軸時，線性內插的軌跡將無法保持直線。使用本功能來編寫程式前，請先確認軌跡，然後再檢查系統是否安全執行動作。**
  否則恐將造成裝置的損壞。

重要

當 MVS 指令所指定的任一個轉軸發出警報時，運動程式將發出警報並停止轉軸動作。

補充

依 MVS 指令移動轉軸後，確認目前已進入定位完成範圍後，不會執行到位確認 (PFN)。若要確認目前是否已進入定位完成範圍，請使用 PFN 指令來確認。

格式

以下為 MVS 指令的格式。

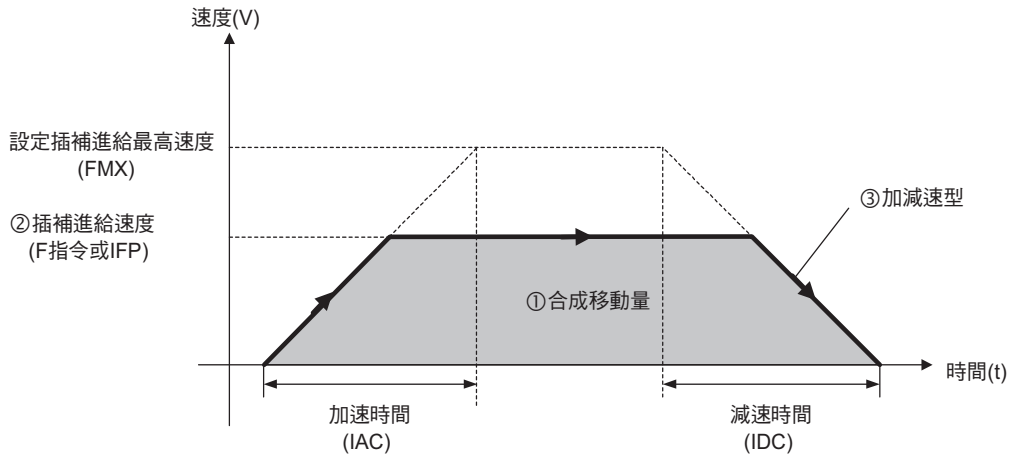
MVS [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 [邏輯軸名稱 3] 指令位置・・・ F 插補進給速度 ;

項目	單位	適用的資料
指令位置	指令單位	・利用長整數型暫存器間接指定 ・利用立即值直接指定
插補進給速度	需符合 FUT 指令的設定單位的指令單位 /min 或指令單位 /s	

(註) 插補進給速度可省略格式。

MVS 指令的設定項目

接下來將說明 MVS 指令的設定項目。



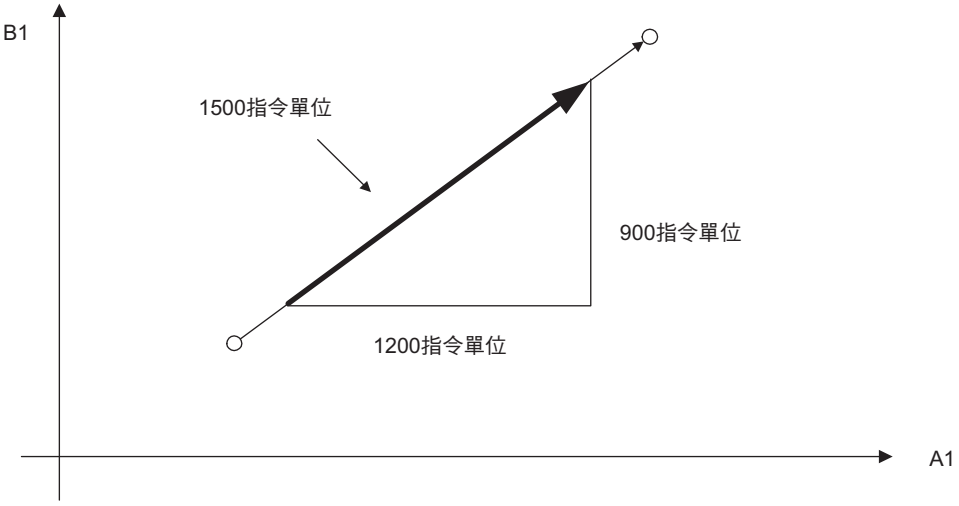
① 合成移動量

ABS 模式或 INC 模式下的合成移動量各不相同。

- ABS 模式下的合成移動量
  在 ABS 模式下，程式現在位置與指令位置之間的差值將被視為合成移動量。
- IINC 模式下的合成移動量
  在 INC 模式下，指令位置將直接被視為合成移動量。

範例 **INC MVS[A1]1200[B1]900；時**

合成移動量為 $\sqrt{1200^2 + 900^2} = 1500$ 。

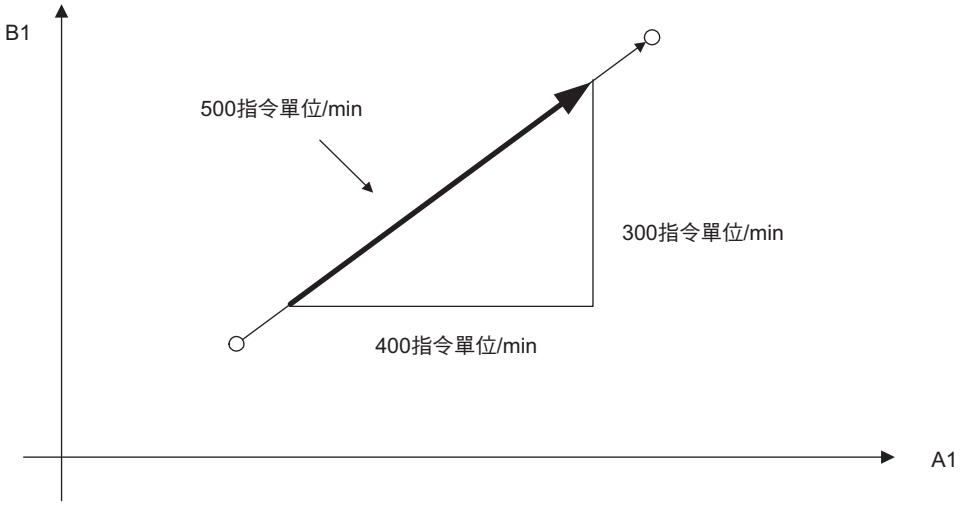


② 插補進給速度 (F 指令或 IFP)

利用 MVS 指令來指定接在字元「F」後面的數值或利用暫存器，即可指定 (F 指令) 插補進給速度。
插補進給速度可用來指定所有轉軸的合成速度。
指令範圍為 1~ 設定插補進給最高速度 (FMX) (指令單位 /min)。

範例 **INC MVS[A1]1200 [B1]900 F500；時**

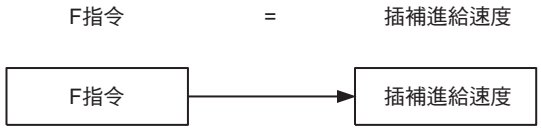
插補進給速度 $\sqrt{400^2 + 300^2} = 500$ (指令單位 /min)。



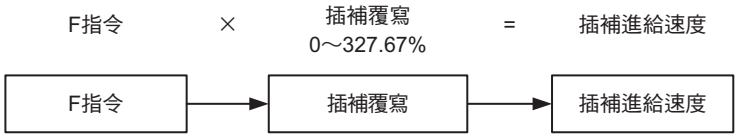
F 指令可用來選擇是 / 否使用插補覆寫動作。
插補覆寫的相關使用方法，請參閱以下章節。

☞ **工作暫存器 (第 1-22 頁)**

・不使用插補覆寫時



・使用插補覆寫時



亦可利用設定插補進給最高速度 (FMX) 相對應的比率，來指定插補進給速度。
關於插補進給速度指令的比率，請參照 IFP 指令。

重要

若您在 F 指令 (指令單位 /min) 所指定的數值超過 FMX 指令值 (指令單位 /min) 時，將會發出運動程式警報。

補充

1. 使用插補覆寫來下指令時，一旦插補進給速度超過 FMX 指令值，插補進給速度的輸出值將變為 FMX 指令值。
2. 若省略、不指定插補進給速度，就會依照最後指定的插補進給速度來執行動作。

插補覆寫指令可在轉軸移動時變更。

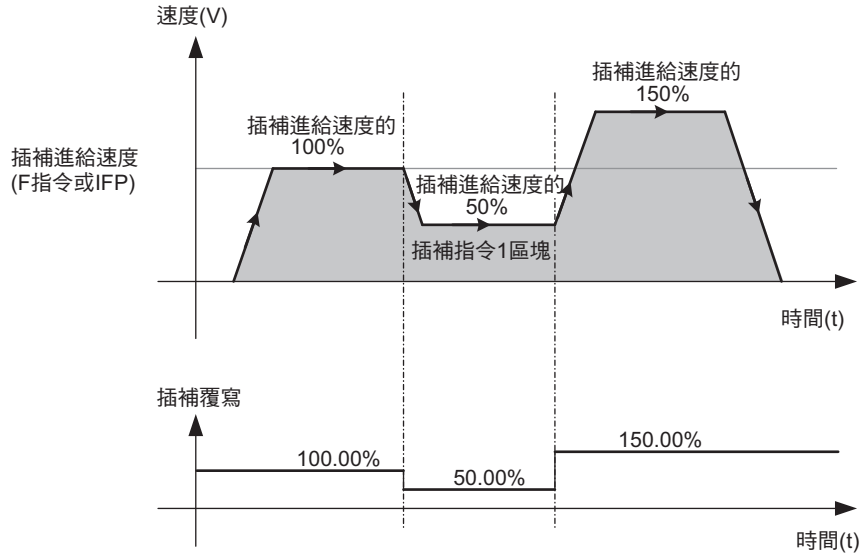


圖 6.34 插補覆寫及插補指令

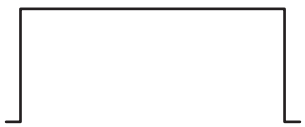
③ 加減速型

利用 IAC/IDC/SCC 指令和設定參數 OW□□□03 Bit 8~B (選擇濾波器類型) 的組合，即可設定加減速類型。

MVS 指令的加減速類型可從以下 3 種加減速類型中選擇。

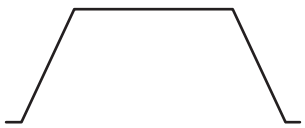
・無加減速

此種方法可讓加速時間、減速時間皆為 0 的狀態下動作。

設定方法	動作示意圖
・OW□□□03 Bit 8~B (選擇濾波器類型) 設定為「0：無濾波器」 ・IAC 指令設定為「0」 ・IDC 指令設定為「0」	

・1 段直線加減速

此種方法可依照固定的加速度、減速度執行動作。

設定方法	動作示意圖
・OW□□□03 Bit 8~B (選擇濾波器類型) 設定為「0：無濾波器」 ・IAC 指定設定為 0 以外數值 ・IDC 指定設定為 0 以外數值	

・S 型加減速

此方法可依照 S 型加速度、減速度執行動作。

設定方法	動作示意圖
・OW□□□03 Bit 8~B (選擇濾波器類型) 設定為「2：平均移動濾波器」 ・IAC 指定設定為 0 以外數值 ・IDC 指定設定為 0 以外數值 ・SCC 指定設定為 0 以外數值	

重要

運動程式的起始位置必須指定為設定插補進給最高速度 (FMX) 指令。
否則，MVS 指令將使得運動程式發出警報。

補充

1. 若不指定加減速時間，將依照加減速時間的初始值 (0 ms) 來執行動作。
2. 當 MVS 指令移動轉軸後，確認目前已進入定位完成範圍後，就不會執行到位確認 (In position check)。若要確認目前是否已經進入定位完成範圍，請使用 PFN 指令。

程式範例

以下為 ABS 模式下執行 MVS 指令之程式範例。

```
FMX T30000000;
ABS;
IAC T1000;
IDC T1000;
MVS [A1]4000 [B1]3000 [C1]2000 F50000;
END;
```

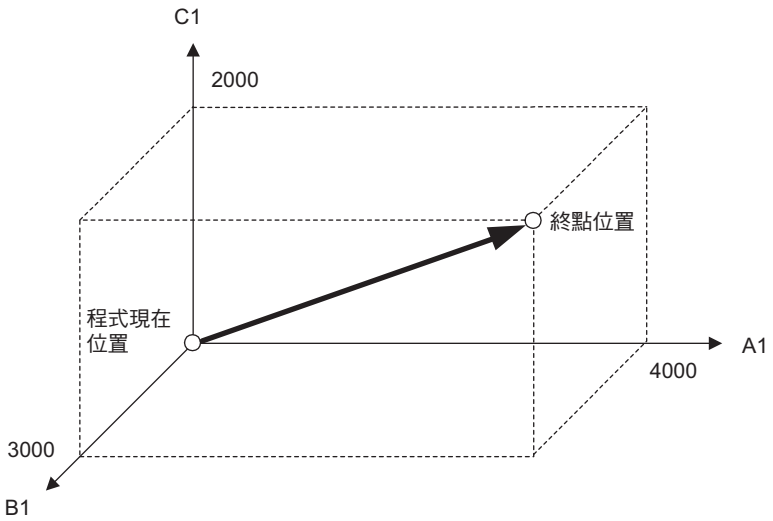


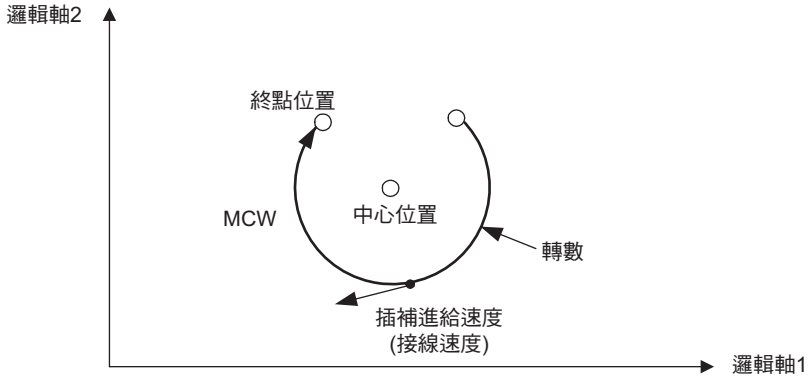
圖 6.35 MVS 指令的程式範例

循環內插 < 中心位置指定 > (MCW、MCC)

循環內插 < 指定中心位置 >(MCW、MCC) 是一項可依照插補進給速度，讓指定平面上的 2 軸同時由程式現在位置移動到終點位置，其中心位置所指定的圓弧上的指令。

・MCW：順時鐘方向 (CW)

・MCC：逆時鐘方向 (CCW)



重要

1. 下達循環內插 < 指定中心位置 >(MCW、MCC) 指令前，必須利用指定平面座標 (PLN) 來指定循環內插的平面。
   若未指定 PLN 指令，循環內插 < 指定中心位置 >(MCW、MCC) 指令將使得運動程式發出警報。
2. 指定終點位置及圓弧中心時，請依照 PLN 指令所指定的橫軸名稱、縱軸名稱相對應的順序來指定。
3. 程式的起始位置必須指定為設定插補進給最高速度 (FMX) 指令。
   若未指定，循環內插 < 指定中心位置 >(MCW、MCC) 指令將使得運動程式發出警報。
4. 當循環內插 < 指定中心位置 >(MCW、MCC) 指令所指定的任一個轉軸發出警報時，運動程式也會發出警報，並停止轉軸動作。

補充

1. 若不指定加減速時間，將依照加減速時間的初始值 (0 ms) 來執行動作。
2. 利用循環內插 < 指定中心位置 >(MCW、MCC) 指令來移動轉軸時，將不執行到位確認，來確認目前是否進入定位完成範圍。若要確認目前是否已進入定位完成範圍，請使用 PFN 指令來確認。

格式

以下為 MCW (循環內插 < 指定中心位置 >) 指令的格式。

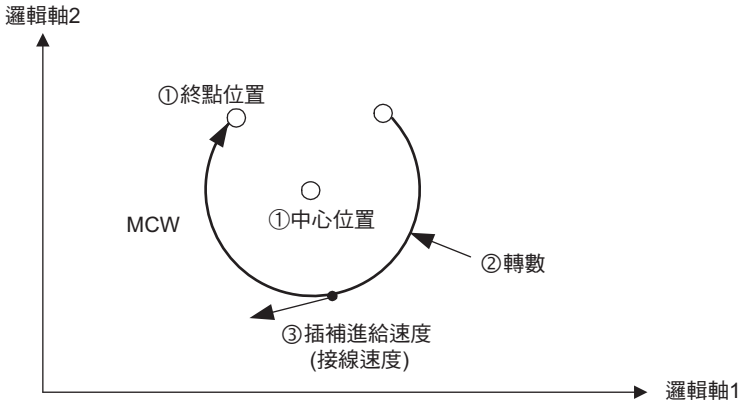
MCW [邏輯軸名稱 1] 終點位置 [邏輯軸名稱 2] 終點位置 U 中心位置 V 中心位置 T 轉數 F 插補進給速度 ;

項目	單位	適用的資料
終點位置	指令單位	・利用立即值直接指定 ・利用長整數型暫存器間接指定
中心位置	指令單位	
轉數	次數	
插補進給速度	需符合 FUT 指令的設定單位 指令單位 /min 或指令單位 /s	

(註) 轉數、插補進給速度可以從格式中省略不寫。

循環內插＜中心位置指定＞ (MCW、MCC) 指令設定項目

接下來將說明循環內插 < 指定中心位置 >(MCW、MCC) 指令設定項目。



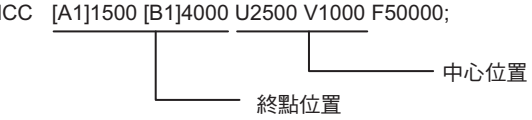
① 終點位置及中心位置

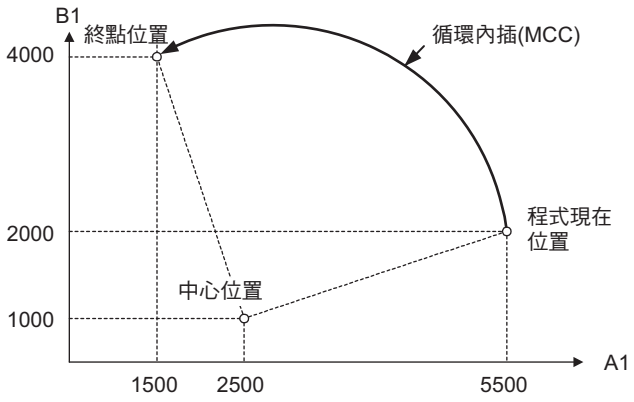
終點位置可利用接在邏輯軸名稱後面的數值或暫存器來指定。
利用循環內插 < 指定中心位置 >(MCW、MCC) 指令中接在字元「U」、「V」後面的數值或是暫存器來指定中心位置。
在 ABS 模式或 INC 模式下，指令位置相對應的實際終點位置及中心位置將各不相同。

範例 **ABS 模式時**

在 ABS 模式下，中心位置及終點位置將被視為絕對位置。

```
FMX T30000000;
ABS;
PLN[A1][B1];
MCC  [A1]1500 [B1]4000 U2500 V1000 F50000;
```





範例

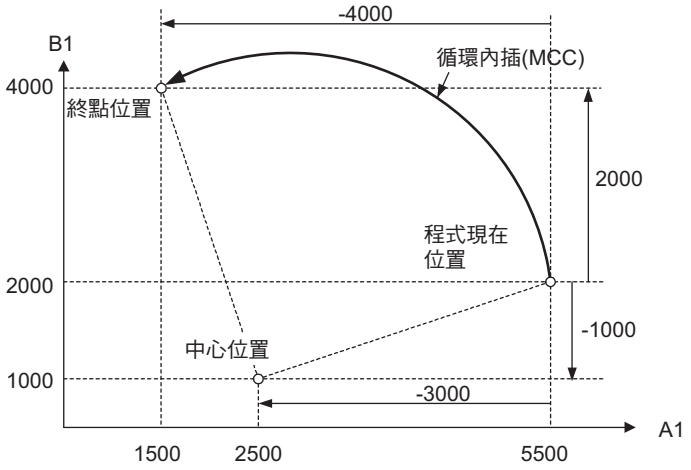
INC 模式時

在 INC 模式下，中心位置及終點位置將被視為程式現在位置的相對位置。

```
FMX T30000000;
INC;
PLN[A1][B1];
MCC [A1]-4000 [B1]2000 U-3000 V-1000 F50000;
```

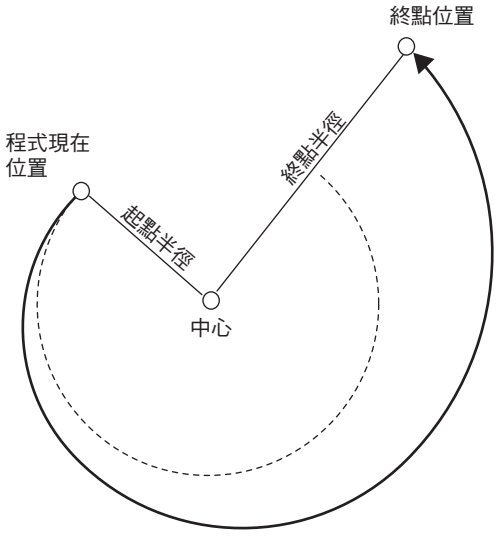
終點位置
(相對位置)

中心位置
(相對位置)



註記

請注意，當起點半徑≠終點半徑時，循環內插的軌跡將如下圖所示。



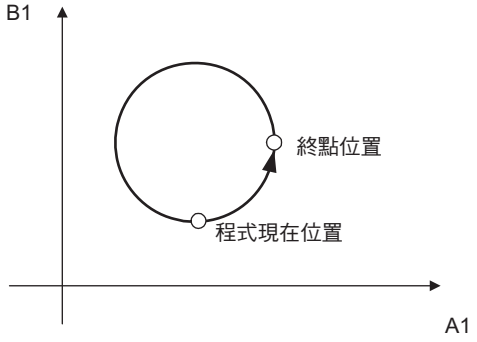
② 轉數

利用循環內插 < 指定中心位置 >(MCW、MCC) 指令中接在字元「T」後面的數值或是暫存器來指定轉數。

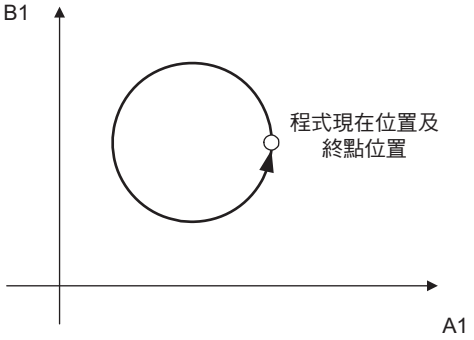
指定轉數後，即可執行多個圓周的動作。若轉數指定為負值，運動程式將會發出警報。指定轉數和多個圓周動作的次數與程式現在位置、終點位置之間的關係如下。

範例 **轉數設定為 2 次**

・ 程式現在位置≠終點位置時，
即為 2 圈 +1/4 圈的圓弧。



・ 程式現在位置 = 終點位置時，
即為 3 周的圓弧。



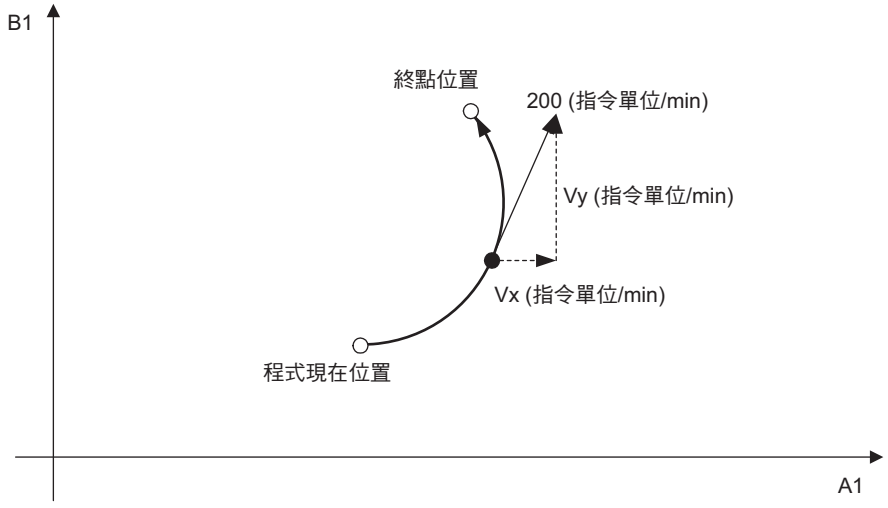
③ 插補進給速度

循環內插 < 指定中心位置 >(MCW、MCC) 指令的插補進給速度即為接線方向的速度。

指令範圍為 1~ 設定插補進給最高速度 (指令單位 /min)。

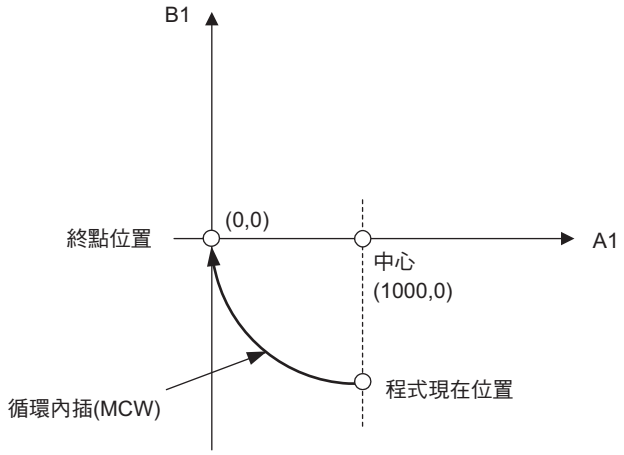
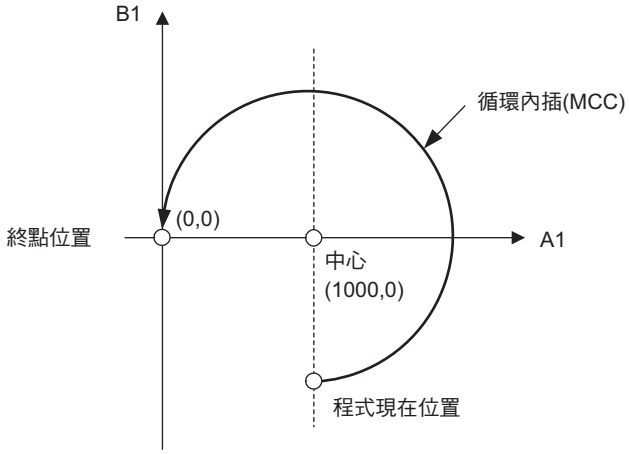
範例 **MCC[A1]- [B1]- F200；時**

插補進給速度 $\sqrt{Vx^2 + Vy^2}$ = 200 (指令單位 /min)。



程式範例

以下為 ABS 模式下執行循環內插 < 指定中心位置 >(MCW、MCC) 指令時之程式範例。
利用 MCW (順時鐘方向) 或 MCC (逆時鐘方向) 即可指定旋轉方向。

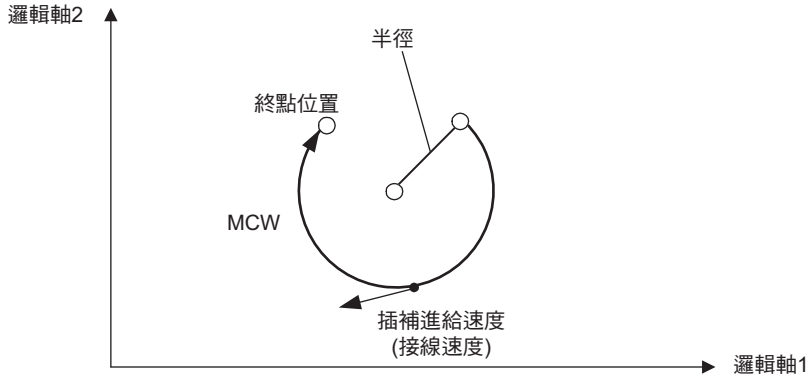
旋轉方向	程式範例
順時鐘方向 (MCW)	**ABS;** **FMX T30000000;** **PLN [A1][B1];** **MCW [A1]0 [B1]0 U1000 V0 F2000; "MCW (順時鐘方向)"** **END;**  **圖 6.36 MCW (指定中心位置) 指令之程式範例**
逆時鐘方向 (MCC)	**ABS;** **FMX T30000000;** **PLN [A1][B1];** **MCC [A1]0 [B1]0 U1000 V0 F2000; "MCC (逆時鐘方向)"** **END;**  **圖 6.37 MCC (指定中心位置) 指令之程式範例**

循環內插 < 指定半徑 >(MCW、MCC)

循環內插 < 指定半徑 >(MCW、MCC) 是一項可依照插補進給速度，讓指定平面上的 2 軸同時由程式現在位置移動到終點位置，其半徑所指定的圓弧上的指令。

・ MCW：順時鐘方向 (CW)

・ MCC：逆時鐘方向 (CCW)



重要

1. 執行循環內插指令前，必須先利用指定座標平面 (PLN) 指令來指定要進行循環內插的平面。
   若未指定 PLN 指令，循環內插 < 指定半徑 >(MCW、MCC) 指令將使得運動程式發出警報。
2. 指定終點位置及圓弧中心時，請依照 PLN 指令所指定的橫軸名稱、縱軸名稱相對應的順序來指定。
3. 程式的起始位置必須指定為設定插補進給最高速度 (FMX) 指令。
   未指定時，循環內插 < 指定半徑 >(MCW、MCC) 指令將使得運動程式發出警報。
4. 當循環內插 < 指定半徑 >(MCW、MCC) 指令所指定的任一個轉軸發出警報時，運動程式也會發出警報，並停止轉軸動作。

補充

1. 若不指定加減速時間，將依照加減速時間的初始值 (0 ms) 來執行動作。
2. 利用循環內插 < 指定半徑 >(MCW、MCC) 指令來移動轉軸後，將不會再執行到位確認 (PFN) 指令，來確認目前是否進入定位完成範圍。若要確認目前是否已進入定位完成範圍，請使用 PFN 指令來確認。

格式

以下為 MCW (循環內插 < 指定半徑 >) 指令的格式。

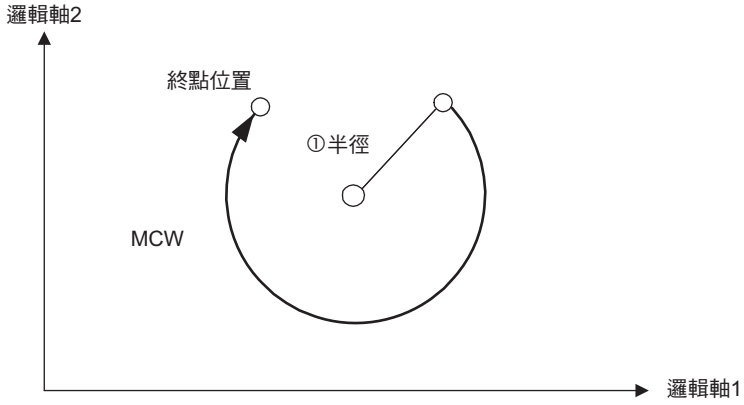
MCW [邏輯軸名稱 1] 終點位置 [邏輯軸名稱 2] 終點位置 R 半徑 F 插補進給速度 ;

項目	單位	適用的資料
終點位置	指令單位	・ 利用立即值直接指定 ・ 利用長整數型暫存器間接指定
半徑	指令單位	
插補進給速度	需符合 FUT 指令的設定單位 指令單位 /min 或指令單位 /s	

(註)1. 指定半徑功能無法用來指定轉數。
2. 插補進給速度可以從格式中省略。

循環內插 < 指定半徑 >(MCW、MCC) 設定項目

接下來將說明循環內插 < 指定半徑 >(MCW、MCC) 指令設定項目。



① 半徑

利用循環內插 < 指定半徑 >(MCW、MCC) 指令的接在字元「R」後面的數值或是暫存器來指定半徑。使用半徑指令值符號時的循環內插軌跡如下所示。

範例

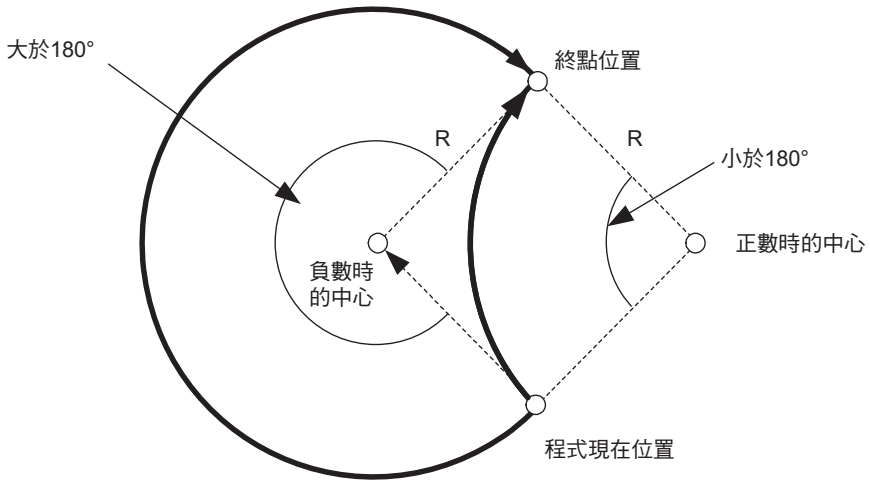
循環內插 < 指定半徑 >(MCW、MCC) 軌跡

使用 MCW　[A1] － [B1] － R －；指令，

R>0 時：在圓弧角小於 180° 的循環內插

R<0 時：在圓弧角大於 180° 的循環內插

R=0 時：發出運動程式將警報。



補充　指定半徑時，無法再對一個或多個圓周下達指令。

程式範例

以下為 ABS 模式下執行循環內插 < 指定半徑 >(MCW、MCC) 指令時之程式範例。
利用 MCW (順時鐘方向) 或 MCC (逆時鐘方向)，即可指定旋轉方向，圓弧角則請利用半徑指令值的符號來指定。

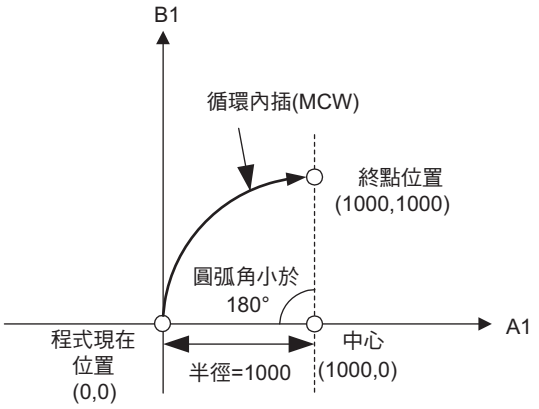
<table>
<tr><th>旋轉方向</th><th>圓弧角</th><th>程式範例</th></tr>
<tr><td rowspan="2">順時鐘方向
(MCW)</td><td>小於 180°
(半徑指令值 >0)</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCW [A1]1000 [B1]1000 R1000 F2000;　"MCW (順時鐘方向)"
END;
 

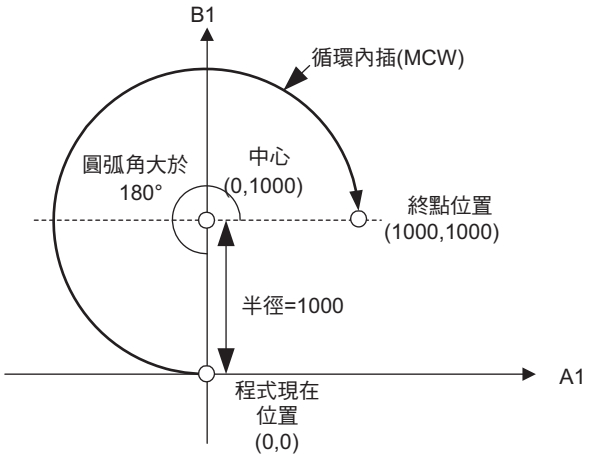
圖 6.38　指定半徑時之程式範例 (MCW)</td></tr>
<tr><td>大於 180°
(半徑指令值 <0)</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCW [A1]1000 [B1]1000 R-1000 F2000;　"MCW (順時鐘方向)"
END;
 

圖 6.39　指定半徑時之程式範例 (MCW)</td></tr>
</table>

(續下頁)

(接上頁)

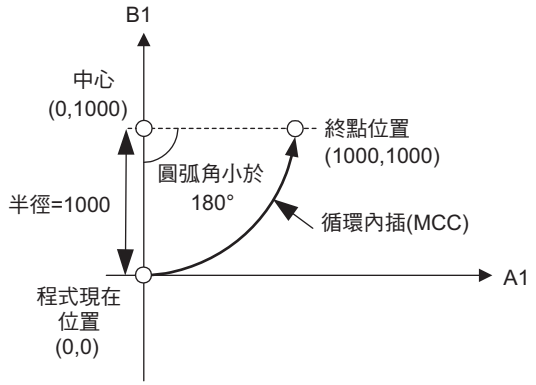
<table>
<tr><th>旋轉方向</th><th>圓弧角</th><th>程式範例</th></tr>
<tr><td rowspan="2">逆時鐘方向
(MCC)</td><td>小於 180°
(半徑指令值 >0)</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]1000 R1000 F2000; "MCC (逆時鐘方向)"
END;
 

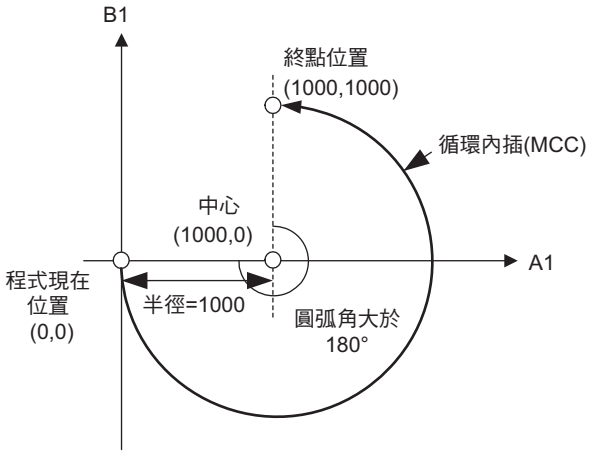
圖 6.40 指定半徑時之程式範例 (MCC)</td></tr>
<tr><td>大於 180°
(半徑指令值 <0)</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]1000 R-1000 F2000; "MCC (逆時鐘方向)"
END;
 

圖 6.41 指定半徑時之程式範例 (MCC)</td></tr>
</table>

螺旋內插 < 指定中心位置 >(MCW、MCC)

螺旋內插 < 指定中心位置 >(MCW、MCC) 是一項利用您所指定的中心位置且在所決定的圓弧上移動 (循環內插)，然後再加以同步，並執行移動線性內插的指令。
插補進給速度即為循環內插的接線速度和線性內插的合成速度。

・ MCW：順時鐘方向 (CW)

・ MCC：逆時鐘方向 (CCW)

⚠ 注意

- **執行螺旋內插 (MCW、MCC) 時，亦可同時將線性內插功能用於直線軸及旋轉軸。不過，線性內插部位的轉軸使用方式不同，將使得螺旋內插的軌跡無法形成螺旋狀。使用本功能來編寫程式前，請先確認軌跡，然後再檢查系統是否安全執行動作。**
  否則恐將造成裝置的損壞。

重要

1. 執行螺旋內插指令前，請先利用指定座標平面 (PLN) 指令來指定循環內插的平面。
   利用 [邏輯軸 1]、[邏輯軸 2]，即可對您所指定的平面橫軸、縱軸終點位置及圓弧中心下達指令。
2. 指定終點位置及圓弧中心時，請依照 PLN 指令所指定的橫軸名稱、縱軸名稱相對應的順序來指定。
3. 線性內插的軸字元可用來指定平面所未指定的 1 個軸。插補平面不一定要有直角。
4. 利用螺旋內插 < 指定中心位置 >(MCW、MCC) 指令來指定任一個轉軸時，一旦發生警報，運動程式亦將發出警報並停止轉軸動作。

補充

利用螺旋內插 < 指定中心位置 >(MCW、MCC) 指令來移動轉軸後，將不會再執行到位確認 (PFN) 指令，來確認目前是否進入定位完成範圍。
若要確認目前是否已進入定位完成範圍，請使用 PFN 指令來確認。

格式

以下為 MCW (螺旋內插 < 中心位置指令 >) 指令的格式。

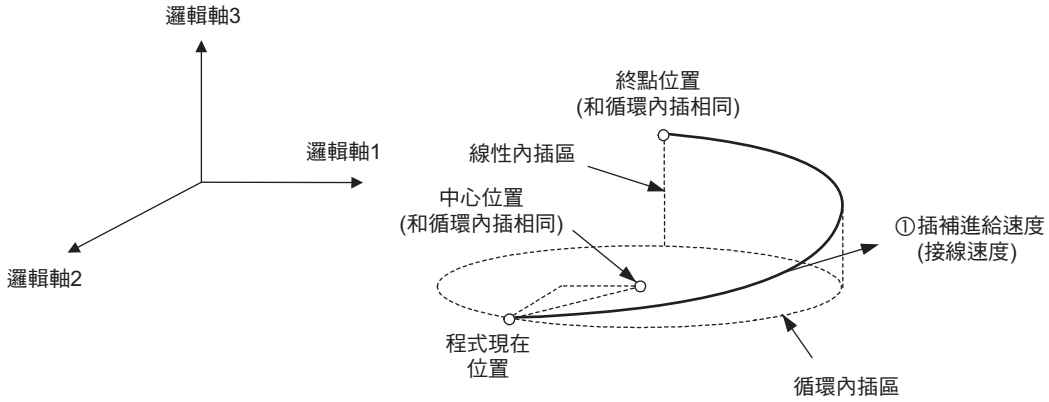
MCW [邏輯軸名稱 1 終點位置] [邏輯軸名稱 2] 終點位置 U 中心位置 V 中心位置
[邏輯軸名稱 3] 線性內插終點位置 T 轉數 F 插補進給速度 ;

項目	單位	適用的資料
終點位置	指令單位	・利用立即值直接指定 ・利用長整數型暫存器間接指定
中心位置	指令單位	
轉數	次數	
插補進給速度	需符合 FUT 指令的設定單位 指令單位 /min 或指令單位 /s	

(註)1. 指定半徑功能無法用來指定轉數。
2. 插補進給速度可以從格式中省略。

螺旋內插 < 指定中心位置 >(MCW、MCC) 設定項目

接下來將說明螺旋內插 < 指定中心位置 >(MCW、MCC) 指令設定項目。

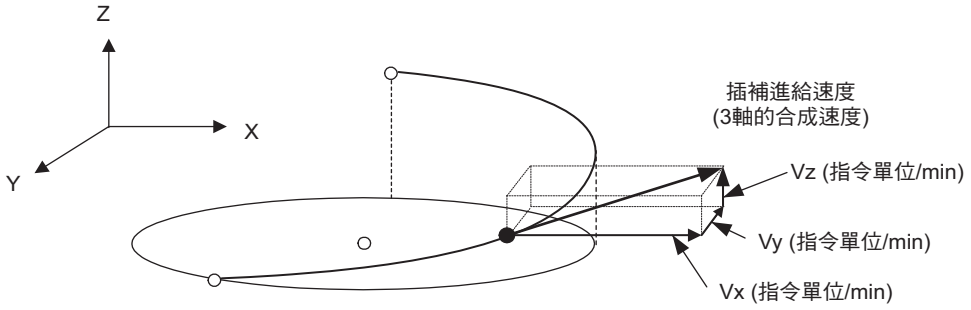


① 插補進給速度

螺旋內插 < 指定中心位置 >(MCW、MCC) 指令的插補進給速度即為循環內插的接線方向速度及線性內插軸的合成速度。

範例 **MCC[X]- [Y]- U- V- [Z]- F300；時**

插補進給速度 $\sqrt{Vx^2 + Vy^2 + Vz^2}$ = 300 (指令單位 /min)。



程式範例

以下為 ABS 模式下螺旋內插 < 指定中心位置 >(MCC) 指令時之程式範例。

```
ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]0 U0 V0 [C1]500 F2000;
END;
```

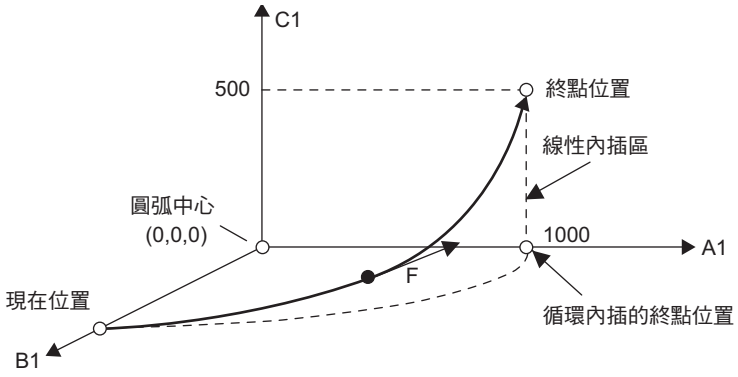


圖 6.42 螺旋內插 < 指定中心位置 >(MCC) 指令程式範例

螺旋內插 < 指定半徑 >(MCW、MCC)

螺旋內插 < 指定半徑 >(MCW、MCC) 是一項利用您所指定的半徑值且在所決定的圓弧上移動 (循環內插) ，然後再加以同步，再執行移動線性內插。
插補進給速度即為循環內插的接線速度和線性內插的合成速度。

・ MCW：順時鐘方向 (CW)

・ MCC：逆時鐘方向 (CCW)

⚠ 注意

- **執行螺旋內插 (MCW、MCC) 時，亦可同時將線性內插功能用於直線軸及旋轉軸。不過，線性內插部位的轉軸使用方式不同，將使得螺旋內插的軌跡無法形成螺旋狀。使用本功能來編寫程式前，請先確認軌跡，然後再檢查系統是否安全執行動作。**
  否則恐將造成裝置的損壞。

重要

1. 執行螺旋內插指令前，請先利用指定座標平面 (PLN) 指令來指定循環內插的平面。
   利用 [邏輯軸 1]、[邏輯軸 2]，即可對您所指定的平面橫軸、縱軸終點位置及圓弧中心下達指令。
2. 指定終點位置及圓弧中心時，請依照 PLN 指令所指定的橫軸名稱、縱軸名稱相對應的順序來指定。
3. 線性內插的軸字元可用來指定平面所未指定的 1 個軸。插補平面不一定要有直角。
4. 利用螺旋內插 < 指定半徑 >(MCW、MCC) 指令來指定任一個轉軸時，一旦發生警報，運動程式亦將發出警報並停止轉軸動作。

補充

利用螺旋內插 < 指定半徑 >(MCW、MCC) 指令來移動轉軸後，將不會再執行到位確認指令，來確認目前是否進入定位完成範圍。
若要確認目前是否已進入定位完成範圍，請使用 PFN 指令來確認。

格式

以下為 MCW (螺旋內插 < 指定半徑 >) 指令的格式。

MCW **[邏輯軸名稱 1] 終點位置** **[邏輯軸名稱 2] 終點位置** **R 半徑**
[邏輯軸名稱 3] 線性內插終點位置 **F 插補進給速度 ;**

項目	單位	適用的資料
終點位置	指令單位	・利用立即值直接指定 ・利用長整數型暫存器間接指定
中心位置	指令單位	
半徑	指令單位	
插補進給速度	需符合 FUT 指令的設定單位 指令單位 /min 或指令單位 /s	

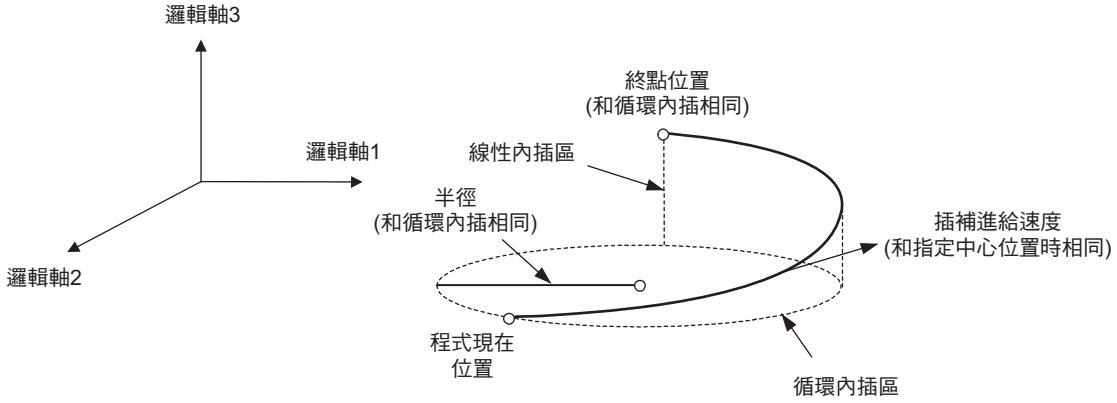
(註)1. 指定半徑功能無法用來指定轉數。
2. 插補進給速度可以從格式中省略。

螺旋內插 < 指定半徑 >(MCW、MCC) 設定項目

接下來將說明螺旋內插 < 指定半徑 >(MCW、MCC) 指令設定項目。

螺旋內插 < 指定半徑 > 指令的半徑、終點位置的指定方法和循環內插 < 指定半徑 > 指令相同。

此外，插補進給速度的指定方法亦和螺旋循環內插 < 指定中心位置 > 相同。



程式範例

以下為 ABS 模式下執行螺旋內插 < 指定半徑 >(MCC) 指令時之程式範例。

```
ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]0 R1000 [C1]500 F2000;
END;
```

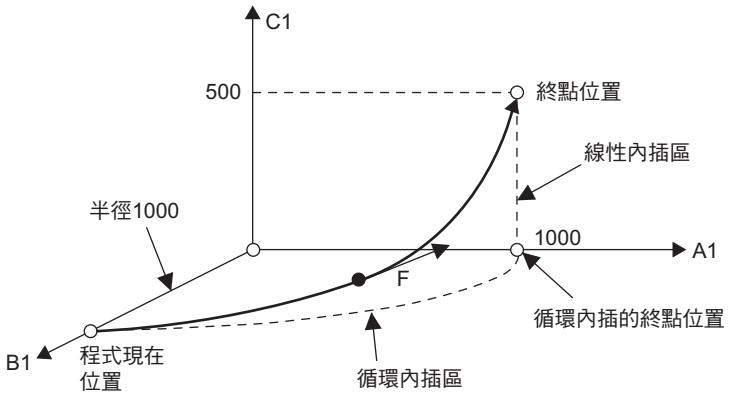


圖 6.43 螺旋內插 < 指定半徑 >(MCC) 指令程式範例

原點復歸 (ZRN)

原點復歸 (ZRN) 是一項用來執行原點復歸動作的指令。
本指令最多可同時移動 32 個轉軸。若省略本指令，將無法移動轉軸。當您所指定的所有轉軸完成原點復歸動作後，就會進入下一個區塊。

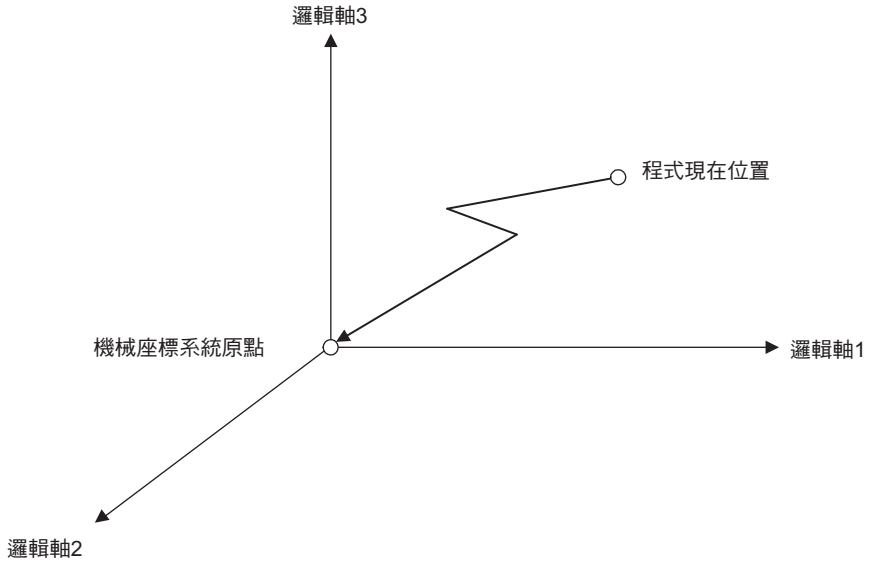


圖 6.44 ZRN 指令的移動軌跡

執行 ZRN 指令後，復歸後的位置會被設定為機械座標系統的原點。同時，您還可以利用變更現在值 (POS) 指令，來取消上一次設定的工作座標系統。
執行 ZRN 指令後，機械座標系統將和工作座標系統一致。接著，機械座標指令 (MVM) 將變為無效，直到變更現在值 (POS) 指令被執行為止。
如欲進一步瞭解機械座標系統和工作座標系統，請參閱以下章節。

☞ **變更現在值 (POS)(第 6-110 頁)**

重要

當 ZRN 指令所指定的任一個轉軸發出警報時，運動程式將發出警報並停止轉軸動作。

註記

當程式被要求暫停時，ZRN 指令就會變為無效。
若要中途停止動作，就必須對程式送出停止要求。
如欲進一步瞭解如何要求程式暫停、要求程式停止，請參閱以下章節。

☞ **工作暫存器 (第 1-22 頁)**

格式

以下為 ZRN 指令的格式。

ZRN [邏輯軸名稱 1] 0 [邏輯軸名稱 2] 0 [邏輯軸名稱 3] 0・・・;

(註) 請務必指定邏輯軸名稱後面的資料 0。

ZRN 指令的設定項目

接下來將說明 ZRN 指令的設定項目。

◆ 原點復歸方式

利用動作設定參數 OW□□□3C (原點復歸方式)，即可設定每個軸的原點復歸方式。
以下為轉軸適用之原點復歸方式。
如欲進一步瞭解每種方式，請參閱以下使用手冊。

📖 **MP3000 系列　運動控制功能　使用手冊 (資料編號：YTWNCO-14013A)**

名稱	設定原點復歸方式 (OW□□□3C)	SVA-01	SVB-01/ SVC-01/ SVC/ SVC32	PO-01
DEC1+C 相脈波	0	○	○	×
ZERO 訊號	1	○	○	×
DEC1+ZERO 訊號	2	○	○	○
C 相脈波	3	○	○	×
DEC2+ZERO 訊號	4	○	×	○
DEC1+LMT+ZERO 訊號	5	○	×	○
DEC2+C 相訊號	6	○	×	×
DEC1+LMT+C 相訊號	7	○	×	×
單 C 脈波	11	○	○	×
P-OT&C 相脈波	12	○	○	×
P-OT	13	○	○	×
HOME LS&C 相脈波	14	○	○	×
HOME LS	15	○	○	×
N-OT&C 相脈波	16	○	○	×
N-OT	17	○	○	×
INPUT&C 相脈波	18	○	○	×
INPUT	19	○	○	×

○：適用、×：不適用

◆ 原點復歸速度

原點復歸速度依您所使用的原點復歸方式而異。

程式範例

以下為 ABS 模式下執行 ZRN 指令之程式範例。

停止位置將被設定為機械座標系統原點 (0,0)。

```
ZRN [A1]0 [B1]0;
END;
```

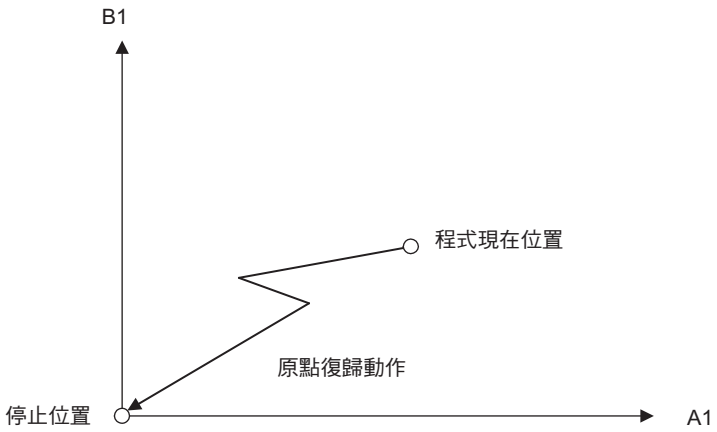


圖 6.45 ZRN 指令之程式範例

送出指令完成後步進定位 (DEN)

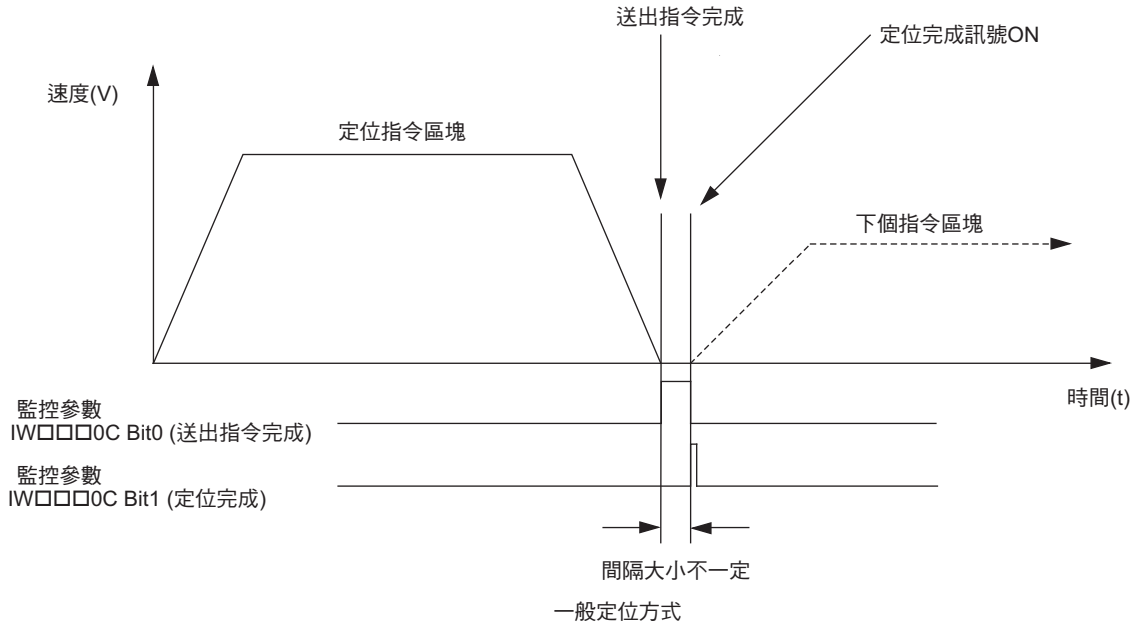
送出指令完成後步進定位 (DEN) 是一項用來擴充定位 (MOV) 的指令。

可同時移動 32 個轉軸。若省略本指令，將無法移動轉軸。

DEN 指令係利用監控參數 IW□□□0C Bit 1 (送出指令完成) 來執行下一個指令區塊，不需要等待監控參數 IW□□□0C Bit 0 (定位完成)。

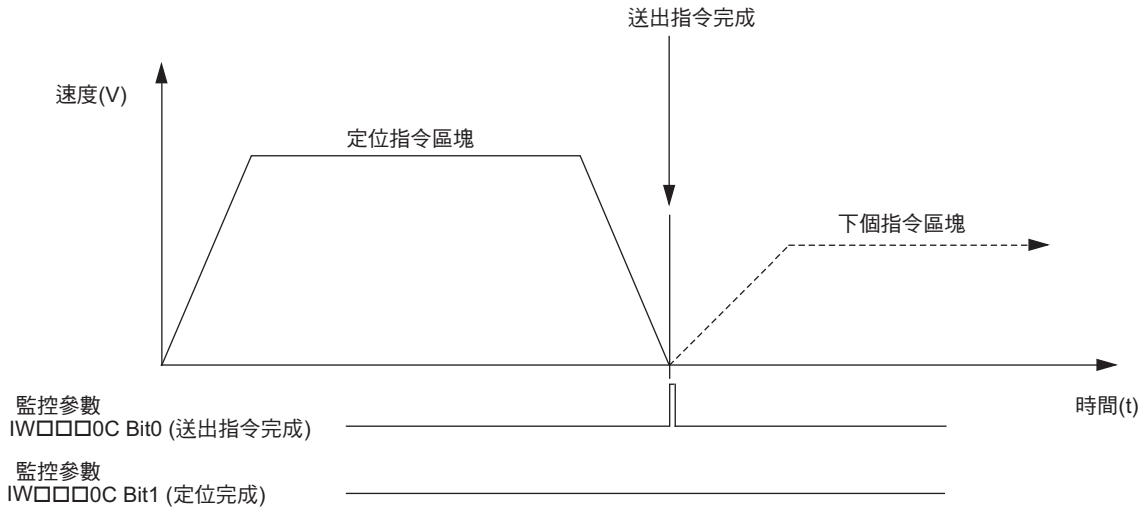
DEN 指令不同於一般的定位動作。

以下為一般的定位動作。



一般定位方式

以下為 DEN 指令所執行之定位動作。



送出指令完成後進行步進定位

圖 6.46 送出指令完成後進行步進定位

格式

以下為 DEN 指令的格式。

MOV [邏輯軸名稱 1] － [邏輯軸名稱 2] － [邏輯軸名稱 3]・・・・・DEN；

項目	單位	適用的資料
指令位置	指令單位	・利用量測值直接指定 ・利用長整數型暫存器間接指定

程式範例

以下為 DEN 指令的程式範例及定位軌跡。

```
ABS;
MOV [A1]10000 DEN;
MOV [B1]10000 DEN;
MOV [A1]20000 DEN;
END;
```

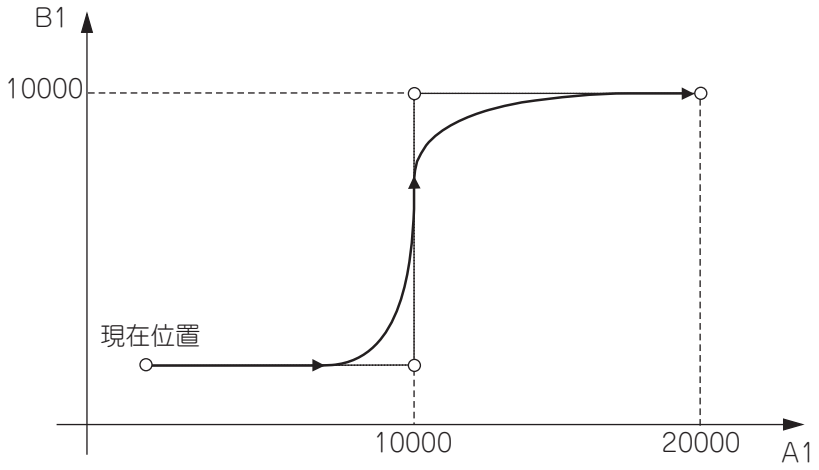
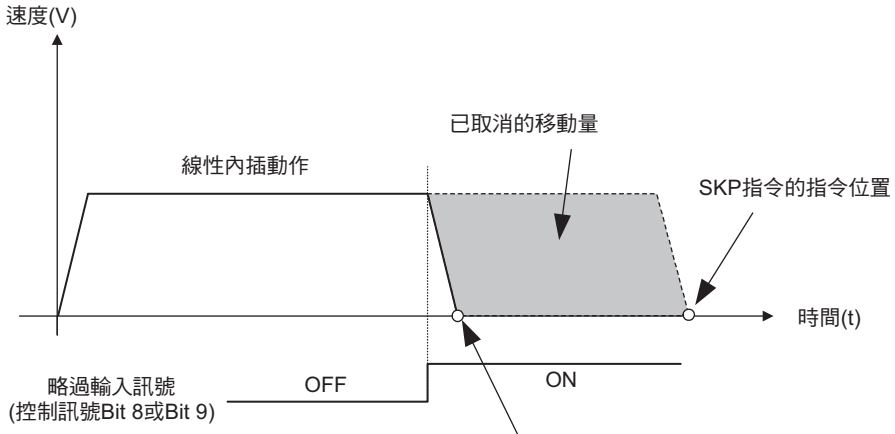


圖 6.47 DEN 指令的程式範例

附略過功能線性內插 (SKP)

附略過功能線性內插 (SKP) 是一項用來擴充線性內插 (MVS) 的指令。若在轉軸移動時，利用 SKP 指令讓略過輸入訊號 ON，此時正在移動的轉軸將停止減速，並取消剩餘移動量。
使用 SKP 指令，即可根據外部狀況，來編寫運動控制程式。
略過輸入訊號會被輸入 MSEE 指令、或是 M-EXECUTOR 控制暫存器的控制訊號中。



選擇略過輸入訊號	略過輸入訊號
略過輸入訊號1(SS1)	控制訊號Bit 8
略過輸入訊號2(SS2)	控制訊號Bit 9

圖 6.48 SKP 指令之動作範例

重要

當 SKP 指令所指定的任一個轉軸發出警報時，運動程式將發出警報並停止轉軸動作。

註記

當略過輸入訊號開啟後，轉軸就會停止減速，不過 SKP 指令則會繼續執行直到定位完成訊號開啟為止。

格式

以下為 SKP 指令的格式。

SKP [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 [邏輯軸名稱 3] 指令位置・・・
F 插補進給速度 選擇 SS 略過輸入訊號 ;

項目	單位	適用的資料
指令位置	指令單位	・利用立即值直接指定 ・利用長整數型暫存器間接指定
插補進給速度	需符合 FUT 指令的設定單位 指令單位 /min 或指令單位 /s	
選擇略過輸入訊號	–	・利用立即值直接指定 (1 或 2) ・利用長整數型暫存器間接指定

(註) 插補進給速度的格式可省略不寫。

程式範例

以下為 ABS 模式下執行 SKP 指令之程式範例。

```
FMX T30000000;
ABS;
IAC T1000;
IDC T1000;
SKP [A1]4000 [B1]3000 [C1]2000 F50000 SS1;
END;
```

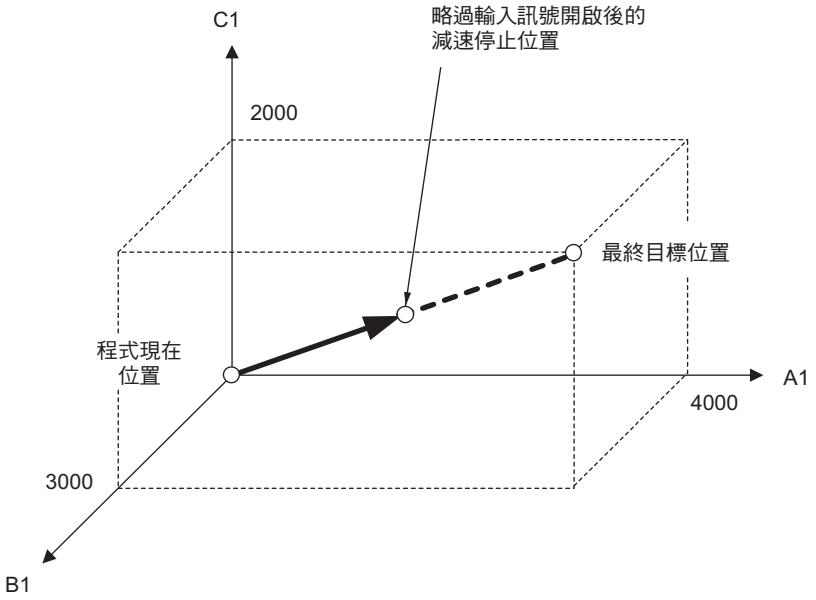


圖 6.49 SKP 指令的程式範例

指定時間定位 (MVT)

指定時間定位 (MVT) 是一項可用來擴充定位 (MOV) 的指令。
本指令最多可同時移動 32 個轉軸。若省略本指令，將無法移動轉軸。
MVT 指令可調整每個軸的進給速度並進行定位，以便在指定時間內定位完成。本指令不會執行插補動作，因此所有指令軸並不一定會同時完成定位動作。
加減速時間的設定值不同，恐將造成時間差。

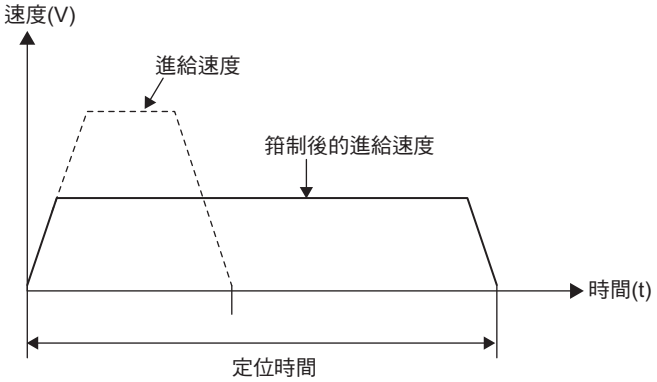


圖 6.50 MVT 指令動作示意圖

使用插補專用覆寫指令時，將無法在指定時間內完成定位。
使用濾波器時，僅濾波器時間常數的部分的定位時間延遲。

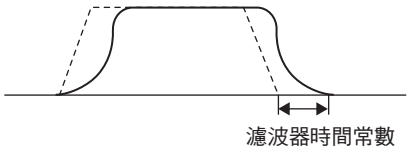


圖 6.51 使用濾波器時發生定位時間延遲

重要

1. MVT 指令可用來覆寫您所使用的轉軸 VEL 指令設定值。請先執行 MVT 指令，然後再利用 VEL 指令來設定進給速度。
2. 當定位時間被設定為 0 後，運動程式將發出警報，並停止轉軸動作。
3. 當轉軸的移動量被設定為 0 時，運動程式將發出警報，並停止轉軸動作。
4. 當 MVT 指令所指定的任一個轉軸發出警報時，運動程式將發出警報並停止轉軸動作。

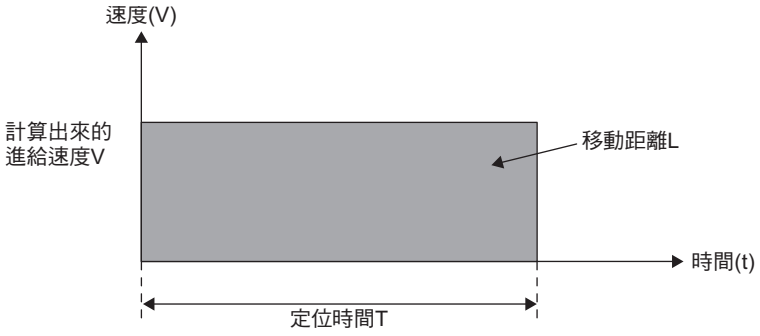
格式

以下為 MVT 指令的格式。

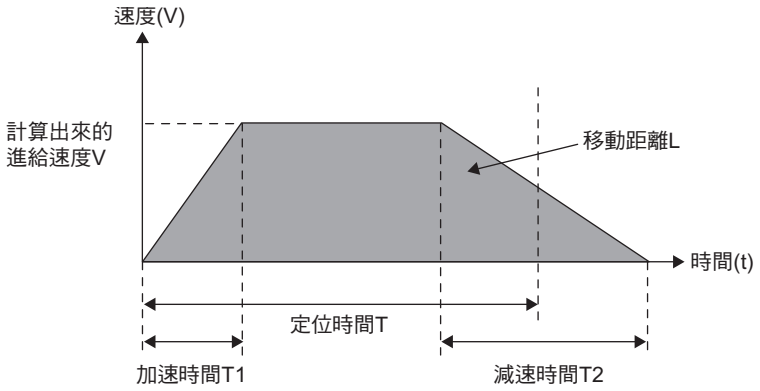
MVT [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 [邏輯軸名稱 3] 指令位置・・・
T 定位時間 ;

項目	單位	適用的資料
指令位置	指令單位	・利用立即值直接指定 ・利用長整數型暫存器間接指定
定位時間	ms	

定位時間的指令範圍為 1~2147483647 ms。
MVT 指令的進給速度係依照定位時間及移動量，在運動控制器內部進行計算。
如下所示，此種計算方式係以加速度 0 作為計算標準。



以下為實際的動作 (加速時間 T1< 減速時間 T2 的條件下)。



利用 MVT 指令即可覆寫您所使用的各軸 VEL 指令設定值，因此當您執行 MVT 指令後，必須再次利用 VEL 指令來設定進給速度。

補充　利用 MVT 指令來移動轉軸時，和 MOV 指令一樣會執行到位確認 (PFN) 指令，以確認目前是否進入定位完成範圍。

程式範例

以下為 ABS 模式下執行 MVT 指令之程式範例。

```
ABS;
ACC [A1]1000;
DCC [A1]1000;
MVT [A1]4000 T1000;
END;
```

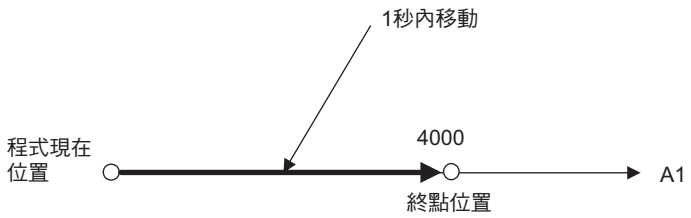


圖 6.52 MVT 指令的程式範例

外部定位 (EXM)

外部定位 (EXM) 指令是一項用來擴充定位 (MOV) 的指令。
EXM 指令會在外部定位訊號開啟後，依照您所指定移動量作為增量值，並進行定位。若外部定位訊號並未變成開啟，將會在 EXM 指令位置結束定位。
EXM 指令僅能指定 1 個轉軸。

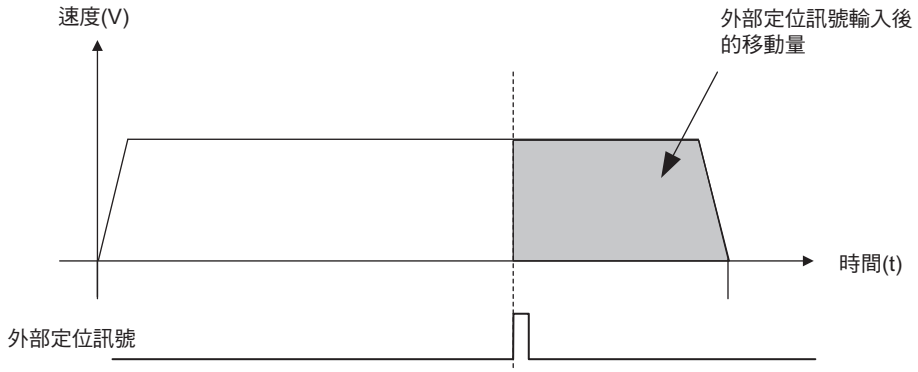


圖 6.53 EXM 指令動作示意圖

若移動量被指定為負數值，將在停止減速後，朝負方向移動。

重要

1. PO-01 模組不適用 EXM 指令。
   PO-01 模組一旦使用 EXM 指令，運動程式就會發出警報。
2. 外部閂鎖輸入訊號可能會被用來執行原點復歸，使用時請特別注意。
3. 當 EXM 指令所指定的任一個轉軸發出警報時，運動程式將發出警報並停止轉軸動作。

格式

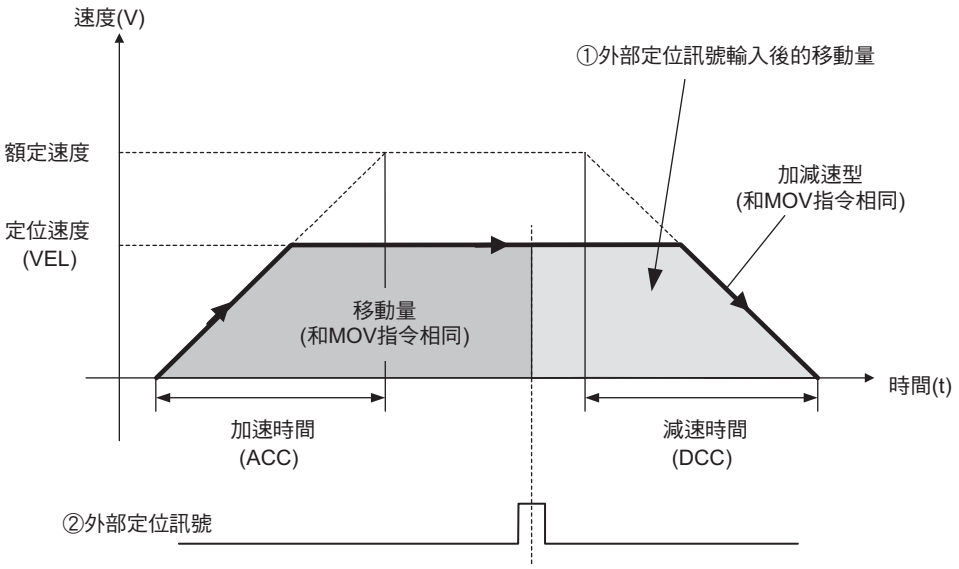
以下為 EXM 指令的格式。

EXM [邏輯軸名稱 1] 指令位置 D 外部定位訊號輸入後的移動量 ;

項目	單位	適用的資料
指令位置	指令單位	・利用立即值直接指定 ・利用長整數型暫存器間接指定
外部定位訊號輸入後的移動量	指令單位	

EXM 指令的設定項目

接下來將說明 EXM 指令的設定項目。



① 外部定位訊號輸入後的移動量

設定時係以外部定位訊號開啟後的移動量作為增量值。
指令範圍為 -2147483648~2147483647 (指令單位)。

② 外部定位訊號

利用設定參數 OW□□□04 Bit 4~7 (設定功能 2)，即可設定外部定位訊號。

程式範例

以下為 ABS 模式下執行 EXM 指令之程式範例。

```
ABS;
ACC [A1]1000;
DCC [A1]1000;
VEL [A1]2000;
DL00000 = 1000;
EXM [A1]4000 DDL00000;
END;
```

6.3 軸控制指令

「軸控制指令」是一種用來控制配置完成之轉軸位置或座標的指令。
軸控制指令包含 7 種指令，僅適用於運動程式。

下表為控制指令一覽表。

指令	名稱	格式	內容	運動程式	序列程式
POS	變更現在值	POS [邏輯軸名稱 1] 欲變更的座標值 [邏輯軸名稱 2] 欲變更的座標值 … ;	最多 32 軸同時，將想要變更的現在值變更至座標值。以後的移動指令將會在新的座標系統上移動。	○	×
MVM	機械座標指令	MVM MOV [邏輯軸名稱 1] 指令位置 [邏輯軸名稱 2] 指令位置 [邏輯軸名稱 3] 指令位置 ・・・;	可指定在機械座標系統上移動。完成原點復歸後，自動設定的座標系統即稱之為「機械座標系統」，是一種不會受到 POS 指令所影響的座標系統。	○	×
PLD	更新程式目前位置	PLD [邏輯軸名稱 1] [邏輯軸名稱 2]…;	利用手動方式，即可更新為位移後的程式現在位置，最多可指定 32 軸。	○	×
PFN	到位確認	MVS [邏輯軸名稱 1] － [邏輯軸名稱 2] －・・・PFN; 或是 MVS [邏輯軸名稱 1] － [邏輯軸名稱 2] －・・・; PFN [邏輯軸名稱 1] [邏輯軸名稱 2]; MVS [邏輯軸名稱 1] － [邏輯軸名稱 2] －・・・;	當相同區塊或前一個區塊的插補移動指令進入到位確認範圍後，就會進入下一個區塊。	○	×
INP	設定到位確認範圍	INP [邏輯軸名稱 1] 定位點附近檢測範圍 [邏輯軸名稱 2] 定位點附近檢測範圍 ・・・;	設定定位點附近檢測範圍。插補移動指令會和接下來所要下達的 PFN 指令進入定位點附近檢測範圍後，就會進入下一個區塊。	○	×
PFP	定位完成檢查	MVS [邏輯軸名稱 1] － [邏輯軸名稱 2] －・・・PFP; 或是 MVS [邏輯軸名稱 1] － [邏輯軸名稱 2] －・・・; PFP [邏輯軸名稱 1] [邏輯軸名稱 2]; MVS [邏輯軸名稱 1] － [邏輯軸名稱 2] －・・・;	定位完成後，相同區塊或前一個區塊的插補移動指令就會進入下一個區塊。	○	×
PLN	指定座標平面	PLN [邏輯軸名稱 1 (縱軸)] [邏輯軸名稱 2 (橫軸)];	指定平面可指定使用於必要指令的平面座標。	○	×

變更現在值 (POS)

變更現在值 (POS) 可用來建立全新的座標系統，並覆寫在希望變更現在位置的座標值上。
本手冊稱新設定的座標系統稱為「工作座標系統」，而機械既有的座標則稱之為「機械座標系統」。
POS 指令之後所下達的移動指令會在工作座標系統上執行動作。

座標系統	說明	備註
機械座標系統	機械既有的座標	原點復歸時的位置即為原點 (0)
工作座標系統	使用者於任意位置所設定的座標	利用 POS 指令設定全新的座標

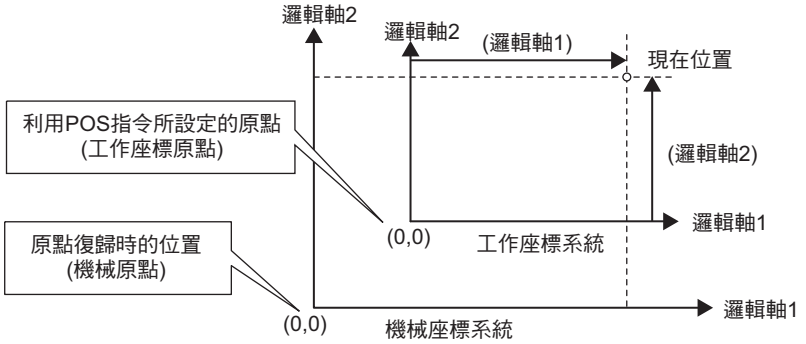


圖 6.54 利用 POS 指令設定工作座標系統

⚠ 注意

- **變更現在值 (POS) 指令可用來建立新的工作座標值。因此，一旦下達錯誤的 POS 指令，恐將造成無法預期的意外發生。下達 POS 指令時，請先確認工作座標系統的位置是否正確後，再讓機器開始運轉。**
  否則恐將造成裝置的損壞。

POS 指令可任意切換工作座標系統。但機械座標系統必須事先設定完成。
POS 指令不會對機械座標系統造成任何影響。
POS 指令最多可同時對 32 軸下達指令。省略指令的軸，則其工作座標系統將不會被更新。
工作座標系統上的移動指令在機械座標系統上換算後，一旦超過最大指令值，將無法下達指令。

下表為機械座標系統和工作座標系統的設定狀態。

表 6.1 座標系統設定時間點

座標系統設定時間點	固定參數 No. 30 (選擇編碼器)	
	增量式編碼器 / 絕對值編碼器 (使用增量)	絕對值編碼器
開啟電源後	機械座標系統：暫時設定 *1 工作座標系統：取消 *3	機械座標系統：有 *2 工作座標系統：取消
執行原點復歸 (ZRN) 後	機械座標系統：設定 工作座標系統：取消	工作座標系統：取消
執行 POS 指令後	工作座標系統：設定	工作座標系統：設定
執行原點設定後	機械座標系統：設定	機械座標系統：設定

*1. 暫時設定就是設定以開啟電源時的現在位置作為原點的機械座標系統。
接著，若不執行原點復歸，則無法開啟軟體 LS 功能。
*2. 若機械座標系統被設定為「有」，就會利用絕對值檢測編碼器所提供的位置資訊，來建立機械座標原點。
*3. 若工作座標系統被設定為「取消」，則先前設定的工作座標系統就會被取消，並與機械座標系統相等。

重要

1. 對於設定無限長的轉軸，設定範圍僅可為 0~POSMAX。
若設定值超過此範圍，運動程式就會發出警報。
2. 若要利用階梯圖程式進行原點復歸，或是不使用 ZRN 指令來進行原點復歸，此時工作座標系統將不會被取消。

格式

以下為 POS 指令的格式。

POS [邏輯軸名稱 1] 座標軸 [邏輯軸名稱 2] 座標軸・・・;

項目	單位	適用的資料
座標軸	指令單位	・利用立即值直接指定 ・利用長整數型暫存器間接指定

程式範例

以下為 POS 指令的程式範例。

```
ABS;                              " 絕對模式

MOV [A1]1000 [B1]2000;            " 定位

POS [A1]0 [B1]0;                  " 更新工作座標系統
MOV [A1]3000 [B1]4000;            " 定位

DL00000 = IL8010;                 " 取得 A1 軸機械座標系統計算位置 (CPOS)
DL00002 = IL8090;                 " 取得 B1 軸機械座標系統計算位置 (CPOS)
POS [A1]DL00000 [B1]DL00002;      " 取消工作座標系統

END;
```

機械座標指令 (MVM)

機械座標指令 (MVM) 係利用變更現在值 (POS)，設定與機械座標系統不同的工作座標系統後，欲暫時在機械座標系統移動時使用的指令。
當機械座標指令 (MVM) 被指定為軸移動指令後，則轉軸將暫時被移動到機械座標系統的絕對座標位置。
此無論 ABS/INC 模式的設定狀態為何，皆必須在 ABS 模式下執行動作。
MVM 指令僅適用於您所指定的區塊。例如，下一個區塊以後所執行的線性內插 (MVS) 將會在工作座標系統上移動。

⚠ 注意

· **機械座標指令 (MVM) 可為機械座標上的座標位置暫時進行定位。因此，下達指令前，若未先確認好機械座標系統的原點位置，恐將造成無法預期的意外發生。下達 MVM 指令前，請先確認機械原點位置是否正確，再讓機器開始運轉。**
否則恐將造成裝置的損壞。

格式

以下為 MVM 指令的格式。

MVM MOV .....；
或是
MVM MVS .....；

程式範例

以下為 MVM 指令的程式範例。

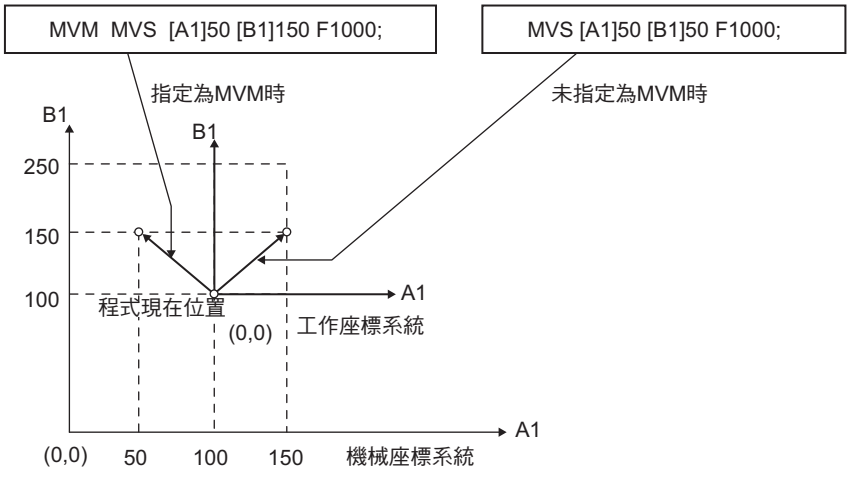


圖 6.55 MVM 指令的程式範例

更新程式現在位置 (PLD)

更新程式現在位置 (PLD) 是一項透過手動方式來更新程式位移後的程式現在位置之指令，本指令最多可同時設定 32 軸。
若運動程式移動過程中，其他程式 (階梯圖程式或執行中以外的運動程式) 也對同一個轉軸下達移動指令，程式將不會更新現在位置。若在此狀態下繼續執行運動程式，只會依照手動的移動量，移動至位移後的位置。若要解決此問題，請使用 PLD 指令，即可更新程式現在位置。

補充

1. 部分用途可由使用者自行來執行 PLD 指令。即是在運動程式運轉中手動介入，亦可不使用 PLD 指令。
2. 若該轉軸未設定 PLD 指令，則程式現在位置將不會被更新。
3. PLD 指令請在標的軸的停止狀態下使用。

格式

以下為 PLD 指令的格式。

PLD [邏輯軸名稱 1] [邏輯軸名稱 2] [邏輯軸名稱 3]...;

程式範例

以下為 PLD 指令的程式範例。

◆ 在運動程式運轉中以手動方式執行時

```
MOV [A1]1000;
                    " 在其間利用 JOG 移動 [A1] 軸
PLD [A1];           " 更新「程式現在位置」
MOV [A1]2000;
```

◆ 利用運動程式的使用者函數來移動轉軸時

```
MOV [A1]1000;
UFC  FNC10 MB000000 IW0100 MB000020;    " 利用使用者函數移動 [A1] 軸
PLD [A1];                               " 更新「程式現在位置」
MOV [A1]2000;
```

◆ 注意事項

先利用運動模組 (SVA-01、SVB-01、SVC-01、PO-01) 對您所指定的轉軸執行插補指令 (MVS、SKP、MCW 或 MCC 指令)，再執行 PLD 指令時，請在執行 PLD 指令前，執行 EOX 指令 (1 次掃描 Wait)。
若因掃描以致資料延遲更新，而無法執行 EOX 指令時，恐將無法更新為正確的程式位置。

範例 **在執行 PLD 指令前先執行 EOX 指令之範例**

```
MVS [A1]1000;                   " 對選購模組的配置軸執行插補指令
EOX;                            "1 次掃描 Wait
PLD [A1];                       " 更新「程式現在位置」
MOV [A1]1000;
```

到位確認 (PFN)

到位確認 (PFN) 是一種用來確認插補移動時是否進入定位點附近之指令。
利用插補指令 (MVS、MCW、MCC、SKP) 來移動轉軸時，一般來說，不會執行到位確認指令來確認目前是否已進入定位完成範圍。若要確認目前是否已經進入定位完成範圍，請使用到位確認。

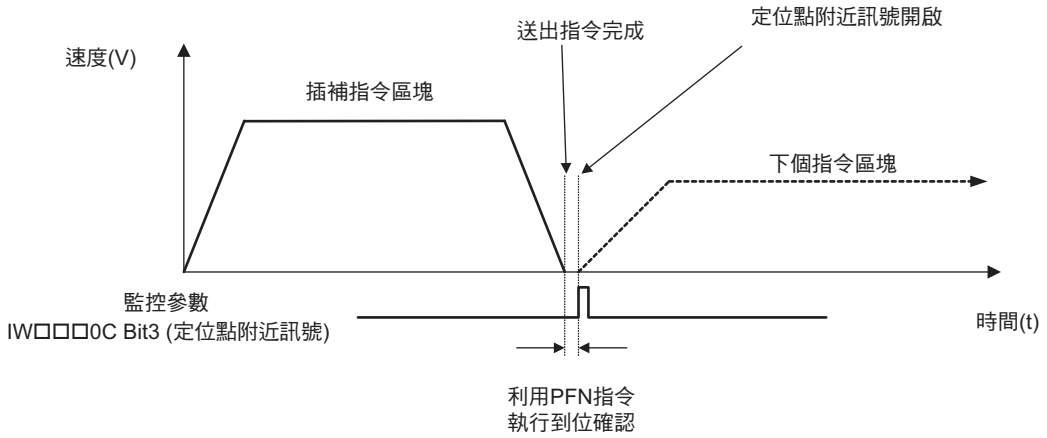


圖 6.56 PFN 指令的動作

只要監控參數 IW□□□0C Bit 3 (定位點附近信號) 符合 |MPOS - APOS| ≦ 定位點附近檢測範圍的條件，就會變為開啟。
利用 INP 指令即可指定定位點附近檢測範圍。

補充　當定位點附近檢測範圍變為 0 時，濾波器等送出指令完成訊號將變為開啟。

格式

以下為 PFN 指令的格式。

・ 所指定的區塊與插補指令相同
MVS [邏輯軸名稱 1] － [邏輯軸名稱 2] － [邏輯軸名稱 3]... PFN;

・ 單獨指定
PFN [邏輯軸名稱 1] [邏輯軸名稱 2] [邏輯軸名稱 3]... ;

程式範例

以下為 PFN 指令的程式範例。

◆ 所指定的區塊與插補指令相同時

```
MVS [A1]1000 F20000 PFN;
MOV [A1]3000;
END;
```

◆ 單獨指定時

```
MVS [A1]1000 F20000;
PFN [A1];
MOV [A1]3000;
END;
```

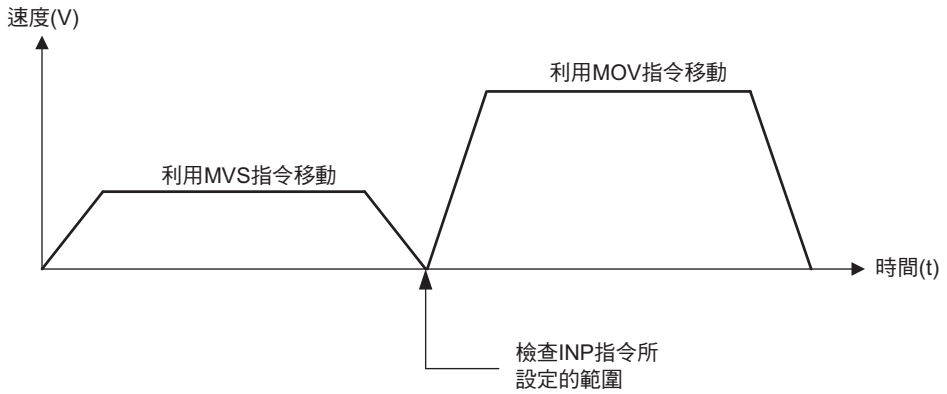


圖 6.57 PFN 指令的程式範例

到位確認範圍設定 (INP)

設定到位確認範圍 (INP) 是一種用來設定定位點附近範圍 (到位確認範圍) 的指令。
本指令最多可同時設定 32 個轉軸。被下達指令的轉軸，會更新各軸的設定參數 OL□□□20 (定位點附近檢測範圍)。
指令範圍為 1~65535 (指令單位)。

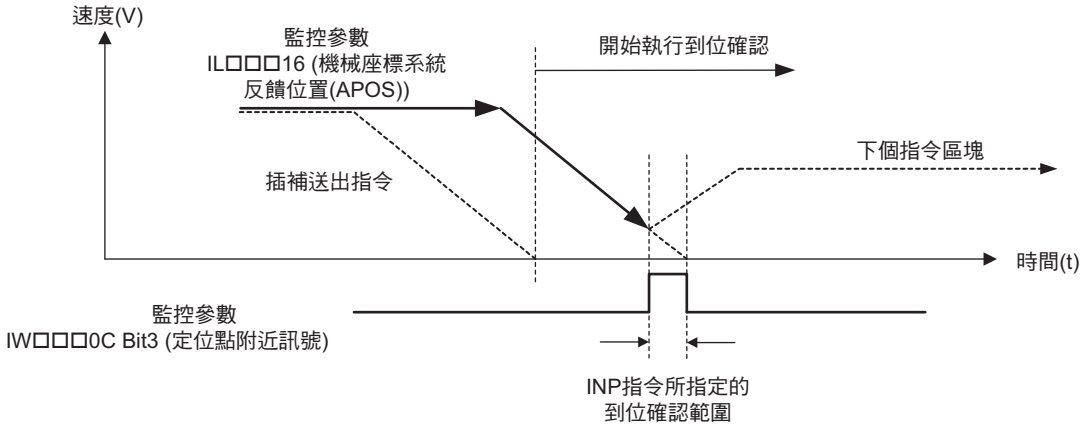


圖 6.58 下達 INP 指令的方法

補充　SVR、SVR32 未內置設定參數 OL□□□20 (定位點附近檢測範圍)。
SVR、SVR32 將以 0 來處理附近檢測範圍。

格式

以下為 INP 指令的格式。

INP　[邏輯軸名稱 1] 定位點附近檢測範圍　[邏輯軸名稱 2] 定位點附近檢測範圍・・・;

項目	單位	適用的資料
定位點附近檢測範圍	指令單位	・利用立即值直接指定 ・利用長整數型暫存器間接指定

程式範例

以下為 INP 指令的程式範例。

```
ABS;
MOV [A1]0 [B1]0;          " 定位至原點
INP [A1]100 [B1]200;      " 設定到位確認範圍
MVS [A1]1000 PFN;
MVS [B1]1000 PFN;
MVS [A1]-1000 ;
END;
```

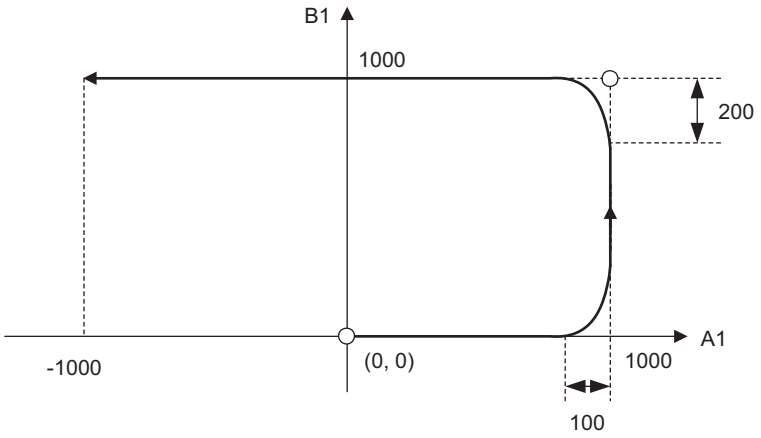


圖 6.59 INP 指令的程式範例

定位完成檢查 (PFP)

定位完成檢查 (PFP) 是一項在插補指令執行移動動作時，確認定位是否完成的指令。

利用插補指令 (MVS、MCW、MCC、SKP) 來移動轉軸時，將直接進入下一個指令區塊，而不執行定位完成檢查。

若要確認定位是否完成，請使用定位完成檢查。

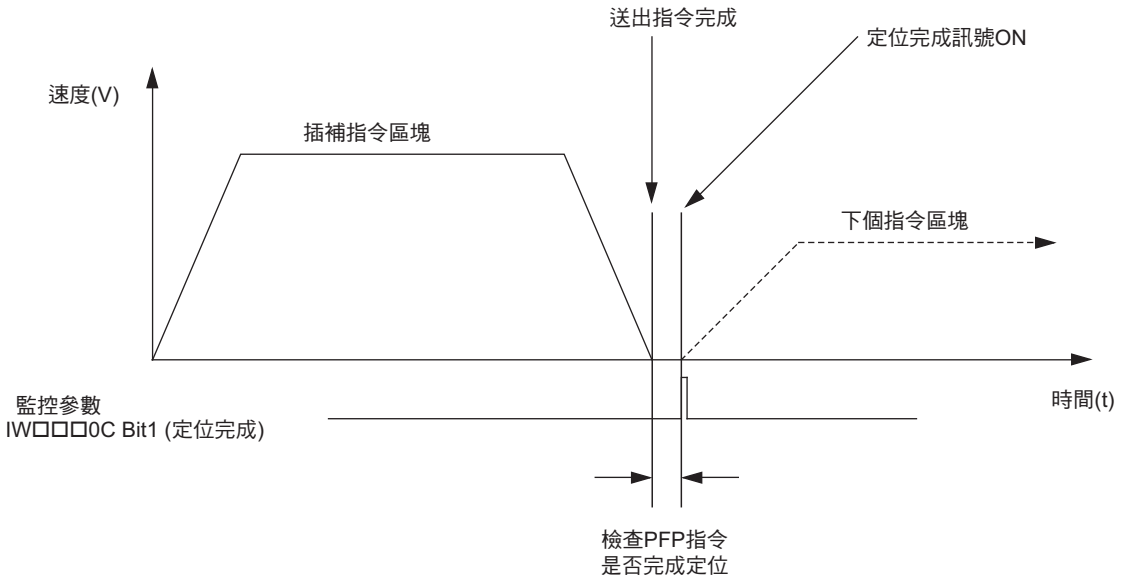


圖 6.60 定位完成檢查

當監控參數 IW□□□0C Bit 1 (定位完成訊號) 完成送出指令，且現在位置已經進入定位完成範圍時，就會變為開啟。

利用 OW□□□1E 即可指定定位完成範圍。

格式

以下為 PFP 指令的格式。

・ 所指定的區塊與插補指令相同

MVS [邏輯軸名稱 1] － [邏輯軸名稱 2] － [邏輯軸名稱 3] ・・・PFP;

・ 單獨指定時

PFP [邏輯軸名稱 1][邏輯軸名稱 2][邏輯軸名稱 3] ・・・;

程式範例

以下為 PFP 指令的程式範例。

◆ 所指定的區塊與插補指令相同時

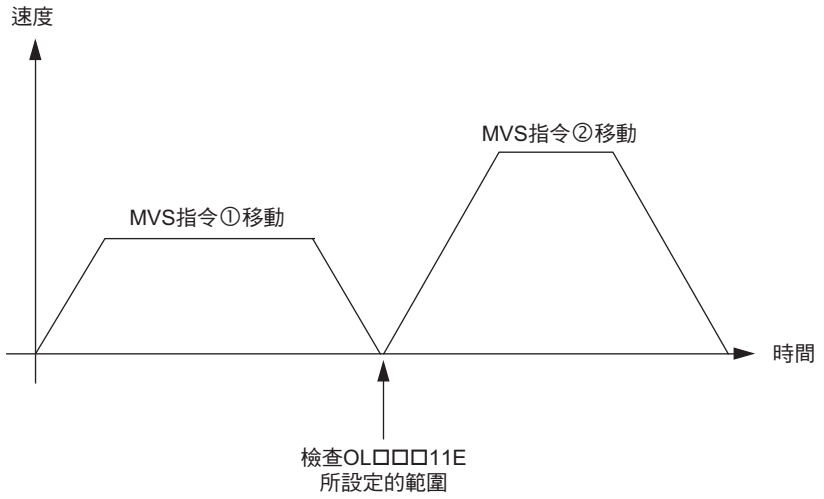
```
MVS [A1]1000 F20000 PFP;          "MVS 指令 ①
MVS [A1]3000 F50000;              "MVS 指令 ②
END;
```

◆ 單獨指定時

```
MVS [A1]1000 F20000;                    "MVS 指令 ①
PFP [A1];
MVS [A1]3000 F50000;                    "MVS 指令 ②
END;
```

註記

使用 PFP 指令時，無法同時再利用 ACCMODE 指令來執行插補連結動作。



指定座標平面 (PLN)

指定座標平面 (PLN) 是一項用來指定參數所設定的 2 個邏輯軸，然後在定義為座標平面的指令。執行循環內插 (MCW、MCC)、螺旋內插 (MCW、MCC) 前，必須先執行本指令。
若該座標平面先前已經被指定過，必須重新定義或是等到程式執行到結束點後才能啟動。

格式

以下為 PLN 指令的格式。

```
                    橫軸名稱        縱軸名稱
PLN  [ 邏輯軸名稱 1] [ 邏輯軸名稱 2];

     指定 2 軸的座標平面
```

程式範例

以下為 PLN 指令的程式範例。

```
PLN[A1][B1];                          " 指定 A1、B1 軸所組成的平面。
MCW  [A1]50 [B1]50 R50 F1000;
```

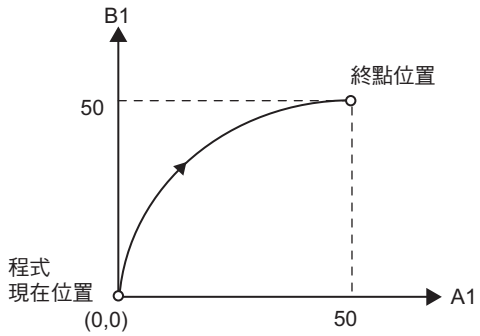
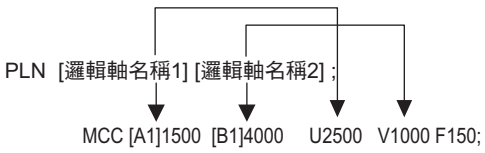


圖 6.61 PLN 指令的程式範例

註記

指定時，須依照循環內插及螺旋內插的終點位置、指定中心位置時利用 PLN 指令所指定的橫軸名稱、縱軸名稱等之相對應順序。



6.4 程式控制指令

所謂「程式控制指令」就是一項用來控制程式執行序列的指令。
程式控制指令包含 16 種指令。

下表為程式控制指令一覽表。

指令	名稱	格式	內容	運動程式	序列程式

IF
ELSE
IEND
	分歧	IF (條件式);
		(處理 1)；
		ELSE;
		(處理 2)；
		IEND;
			滿足條件式時，執行 (處理 1)，若否，則執行 (處理 2)。	○	○

WHILE WEND	循環	WHILE (條件式); ・・・； WEND;	在持續滿足條件式的期間循環執行 WHILE~WEND 的處理。	○	○
WHILE WENDX	包含 1 次掃描 WAIT 的循環	WHILE (條件式); ・・・； WENDX;	在持續滿足條件式的期間循環執行 WHILE~WENDX 的處理。 掃描 1 次後，執行 1 次迴圈的處理作業。	○	○
PFORK JOINTO PJOINT	並列執行	PFORK 標籤 1， 標籤 2， 標籤 3・・・; 標籤 1: 處理 1； JOINTO 標籤 X; 標籤 2: 處理 2； JOINTO 標籤 X; 標籤 3: 處理 3； JOINTO 標籤 X; 標籤 X: PJOINT;	並列執行標籤所指定的區塊。 處理並列執行時，無法使用 END、RET 指令。	○	×
SFORK JOINTO SJOINT	選擇執行	SFORK 條件式 1? 標籤 1， 條件式 2? 標籤 2， 條件式 3? 標籤 3， 條件式 4? 標籤 4; 標籤 1: 處理 1； JOINTO 標籤 X； 標籤 2: 處理 2； JOINTO 標籤 X； 標籤 3: 處理 3； JOINTO 標籤 X； 標籤 4: 處理 4； JOINTO 標籤 X； ・・・； 標籤 X: SJOINT;	滿足條件式 1 時，執行處理 1，滿足條件式 2 時，則執行處理 2。	○	○
MSEE	叫出運動子程式	MSEE MPS□□□;	執行 MPS□□□ 子程式。	○	×
SSEE	叫出序列子程式	SSEE SPS□□□;	執行 SPS□□□ 子程式。	×	○

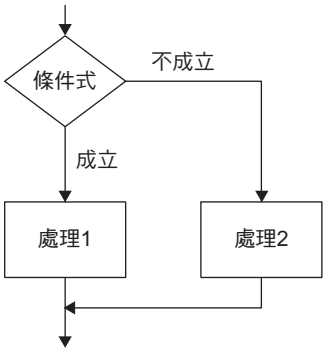
(續下頁)

(接上頁)

指令	名稱	格式	內容	運動程式	序列程式
UFC	從運動程式叫出使用者函數	UFC 使用者函數名稱 輸入資料、輸入位址、輸出資料；	利用運動程式叫出使用者自行編寫的函數。	○	×
FUNC	從序列程式叫出使用者函數	FUNC 使用者函數名稱 輸入資料、輸入位址、輸出資料；	利用序列程式叫出使用者自行編寫的函數。	×	○
END	程式結束	END;	結束程式。	○	○
RET	子程式結束	RET;	結束子程式。	○	○
TIM	等待時間	TIM T－;	等待 T 所指定的時間後，即進入下一個區塊。	○	×
TIM1MS	等待時間 (1 ms)	TIM1MS T－;	等待 T 所指定的時間後，即進入下一個區塊。	○	×
IOW	等待輸出入變數	IOW MB－==・・・;	運動程式將停止執行，直到條件式成立為止。	○	×
EOX	1 次掃描 WAIT	EOX;	此指令可用來分割連續執行的序列指令。 執行 EOX 以後的指令，即可開始下一次的掃描。	○	×
SNGD/ SNGE	單一區塊關閉 (SNGD)/ 單一區塊開啟 (SNGE)	SNGD; ・・・; SNGE;	此指令可用來指定是 / 否執行除錯運轉的步進動作。	○	×

分岐 (IF ELSE IEND)

分岐 (IF ELSE IEND) 指令就是當條件式滿足時，即執行 「IF~ELSE」間的區塊，若否，則執行「ELSE~IEND」間的區塊。
「ELSE」可省略。此時，若條件式無法滿足，就會從 「IEND」的下一個區塊來繼續執行動作。



補充　分岐 (IF ELSE IEND) 最多可執行 8 個巢狀式。

格式

以下為 IF ELSE IEND 指令的格式。

```
IF ( 條件式 );
  …( 處理 1)
ELSE;
  …( 處理 2)
IEND;
```

以下為適用於分歧指令的條件式。

◆ 比較位元型資料

■ 格式

數值比較指令以 ==(等於) 來表示。

式子左邊編寫暫存器，右邊則寫 0 或 1。

```
IF MB000000 = = 0;   "MB000000 = 0
IF MB000000 = = 1;   "MB000000 = 1
```

■ 條件式運算

編寫 &、|、! (邏輯積、邏輯和、反轉) 邏輯演算。

```
IF (MB000000 &  MB000001) = = 1;   "MB000000=1 且MB000001 = 1
IF (MB000000 & !MB000001) = = 1;   "MB000000=1 且MB000001 = 0
IF (MB000000 ｜ MB000001) = = 1;   "MB000000=1 或是MB000001 = 1
IF (MB000000 ｜ !MB000001) = = 1;   "MB000000=1 或是MB000001 = 0
```

■ **語法錯誤範例**

發生下述情況時，將出現語法錯誤。

・**在數值比較指令中編寫時 <>(不等於)**

```
IF MB000000 <> 0;                    ⇒ 語法錯誤
```

・於式子左邊編寫數值，或於右邊編寫暫存器時

```
IF 1 = = MB000000;                   ⇒ 語法錯誤
IF MB000000 = = MB000001;            ⇒ 語法錯誤
```

・未加上數值比較指令時

```
IF MB000000;                         ⇒ 語法錯誤
IF (0);                              ⇒ 語法錯誤
```

・編寫多個數值比較指令時

```
IF (MB000000 = = 0) & (MB000001 = = 1);      ⇒ 語法錯誤
```

◆ 比較整數 / 長整數 / 實數型資料

■ **格式**

可使用所有的數值比較指令 (==、<>、>、<、>=、<=)。

暫存器可編寫在式子左邊或右邊。

```
IF MW00000 = = 3;                    "MW00000 ＝ 3
IF ML00000 <> ML00002;               "ML00000 ≠ ML00002
IF 1.23456 >= MF00000;               "1.23456 ≧ MF00000
```

■ **條件式運算**

可編寫數值運算或邏輯運算。

```
IF MW00000 = = (MW00001/3);          "MW00000 ＝ (MW00001÷3)
IF (ML00000 & F0000000H) <> ML00002; "(ML00000 Λ F0000000H) ≠ ML00002
IF 1.23456 >= (MF00000 * MF00002);   "1.23456 ≧ (MF00000×MF00002)
```

■ **語法錯誤範例**

發生下述情況時，將出現語法錯誤。

・式子兩邊皆只有編寫常數時

```
IF 0 = = 3;                          ⇒ 語法錯誤
IF (3.14 * 2 * 1000) > 9000.0;       ⇒ 語法錯誤
```

・未加上數值比較指令時

```
IF MW000000;                         ⇒ 語法錯誤
IF (-1);                             ⇒ 語法錯誤
```

・編寫多個數值比較指令時

```
IF (MW00000 < 0) & (MW000001 > 0);           ⇒ 語法錯誤
```

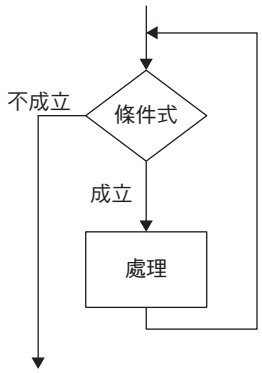
程式範例

以下為 IF ELSE IEND 指令的程式範例。

```
IF MB000000 = = 1;
MOV [A1] 10000;          " 當 MB000000 變為開啟時，執行 A1 定位。
ELSE;
MOV [B1] 10000;          " 當 MB000000 變為關閉時，執行 B1 定位。
IEND;
```

循環 (WHILE WEND)

循環 (WHILE WEND) 指令就是當條件式滿足時，即循環執行 「WHILE~WEND」區塊，反之，若 < 條件式 > 不成立，則跳躍到 「WEND」的下一個區塊。



重要

若您在編寫循環處理作業時，僅讓指令執行 1 次掃描，將因加重掃描處理的負擔，而造成掃描逾時，或是發生監視逾時的情形。
只要執行 1 次掃描時，請變更為 WHILE WENDX 指令 (循環 1 次掃描 WAIT)，或是加上 EOX 指令 (1 次掃描 WAIT 指令) 或是 TIM 指令 (等待時間)。
關於執行 1 次掃描的指令，請參閱以下章節。
☞ **5.4 指令類型和執行掃描動作 (第 5-13 頁)**

補充　循環 (WHILE WEND) 指令最多可執行 8 個巢狀式。

格式

以下為 WHILE WEND 指令的格式。

```
WHILE ( 條件式 ) ;
  ...;
  ( 處理 );
  ...;
WEND   ;        " 結束循環指令
```

以下為循環指令所適用之條件式。

◆ 比較位元型資料

■ 格式

數值比較指令以 ==(等於) 來表示

式子左邊編寫暫存器，右邊則寫 0 或 1。

```
WHILE MB000000 = = 0;   "MB000000 = 0
WHILE MB000000 = = 1;   "MB000000 = 1
```

■ 條件式運算

編寫 &、|、! (邏輯積、邏輯和、反轉) 邏輯演算。

```
WHILE (MB000000 &  MB000001) = = 1;    "MB000000=1 且MB000001 = 1
WHILE (MB000000 & !MB000001) = = 1;    "MB000000=1 且MB000001 = 0
WHILE (MB000000 ｜ MB000001) = = 1;    "MB000000=1 或是MB000001 = 1
WHILE (MB000000 ｜ !MB000001) = = 1;   "MB000000=1 或是MB000001 = 0
```

■ 語法錯誤範例

發生下述情況時，將出現語法錯誤。

・在數值比較指令中編寫 <>(不等於) 時

```
WHILE MB000000 <> 0;                    ⇒ 語法錯誤
```

・於式子左邊編寫數值，或於右邊編寫暫存器時

```
WHILE 1 = = MB000000;                   ⇒ 語法錯誤
WHILE MB000000 = = MB000001;            ⇒ 語法錯誤
```

・未加上數值比較指令時

```
WHILE MB000000;                         ⇒ 語法錯誤
WHILE (0);                              ⇒ 語法錯誤
```

・編寫多個數值比較指令時

```
WHILE (MB000000 = = 0) & (MB000001 = = 1);      ⇒ 語法錯誤
```

◆ 比較整數 / 長整數 / 實數型資料

■ 格式

可使用所有的數值比較指令 (==、<>、>、<、>=、<=) 等。

暫存器可編寫在式子左邊或右邊。

```
WHILE MW00000 = = 3;            "MW00000 ＝ 3
WHILE ML00000 <> ML00002;       "ML00000 ≠ ML00002
WHILE 1.23456 >= MF00000;       "1.23456 ≧ MF00000
```

■ 條件式運算

可編寫數值運算或邏輯運算。

```
WHILE MW00000 = = (MW00001/3);          "MW00000 ＝ (MW00001÷3)
WHILE (ML00000 & F0000000H) <> ML00002; "(ML00000 Λ F0000000H) ≠ ML00002
WHILE 1.23456 >= (MF00000 * MF00002);   "1.23456 ≧ (MF00000×MF00002)
```

■ 語法錯誤範例

發生下述情況時，將出現語法錯誤。

・式子兩邊皆只有編寫常數時

```
WHILE 0 = = 3;                          ⇒ 語法錯誤
WHILE (3.14 * 2 * 1000) > 9000.0;       ⇒ 語法錯誤
```

・未加上數值比較指令時

```
WHILE MW000000;                         ⇒ 語法錯誤
WHILE (-1);                             ⇒ 語法錯誤
```

・編寫多個數值比較指令時

```
WHILE (MW00000 < 0) & (MW000001 > 0);   ⇒ 語法錯誤
```

程式範例

以下係利用 WHILE WEND 指令畫出 10 個半徑為 50 的圓之程式範例。

```
MOV [A1] 0 [B1] 0;                      " 定位
MW00100 = 1;                            " 計數器預設
INC;                                    " 指定為增量模式
PLN [A1] [B1];                          " 指定座標平面
WHILE MW00100 <= 10 ;                   " 循環指令
   MCW [A1]0 [B1]0 U50. V50. F8000 ;    " 循環內插
   MOV [A1]50. [B1]50.;                 " 定位
   MW00100 = MW00100 + 1;               " 計數器累加
WEND ;                                  " 結束循環指令
```

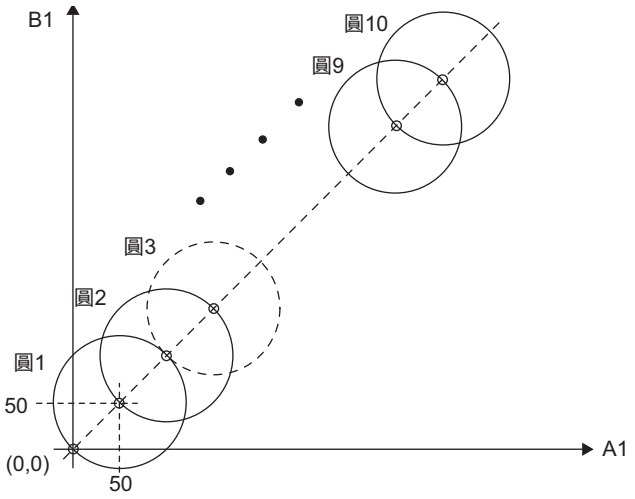
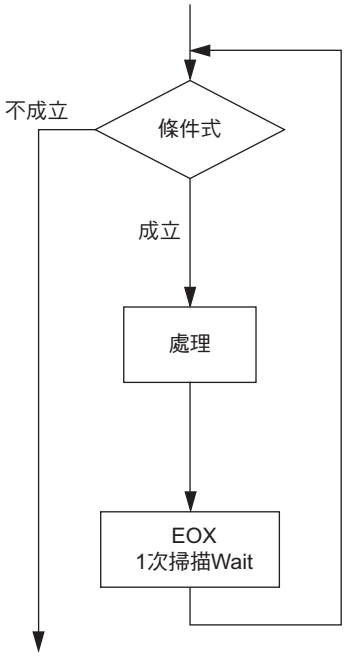


圖 6.62 循環 (WHILE WEND) 指令之程式範例

包含 1 次掃描 WAIT 的循環 (WHILE WENDX)

包含 1 次掃描 WAIT 的循環 (WHILE WENDX) 是一種組合了循環 (WHILE WEND) 與 1 次掃描 WAIT(EOX) 的指令。若條件式滿足時，就會循環執行 「WHILE~WENDX」區塊，若不成立，則跳躍至 「WENDX」的下一個區塊。

在 WENDX 的上一個區塊時，必須等待 1 次掃描，然後再執行 1 次掃描 1 次迴圈的處理作業。



格式

以下為包含 1 次掃描 WAIT 的循環 (WHILE、WENDX) 指令之格式。

```
WHILE ( 條件式 )；
        ・・・；
      ( 處理 )；
        ・・・；
WENDX；                          "1 次掃描 WAIT 後，結束循環指令 "
```

以下為循環指令所適用之條件式。

◆ 比較位元型資料

■ 格式

數值比較指令以 ==(等於) 來表示。

式子左邊編寫暫存器，右邊則寫 0 或 1。

```
WHILE MB000000 = = 0;   "MB000000 = 0
WHILE MB000000 = = 1;   "MB000000 = 1
```

■ 條件式運算

編寫 &、|、! (邏輯積、邏輯和、反轉) 邏輯演算。

```
WHILE (MB000000 &  MB000001) = = 1;      "MB000000=1 且MB000001 = 1
WHILE (MB000000 & !MB000001) = = 1;      "MB000000=1 且MB000001 = 0
WHILE (MB000000 ｜ MB000001) = = 1;      "MB000000=1 或是MB000001 = 1
WHILE (MB000000 ｜ !MB000001) = = 1;     "MB000000=1 或是MB000001 = 0
```

■ 語法錯誤範例

發生下述情況時，將出現語法錯誤。

・在數值比較指令中編寫 <>(不等於)

```
WHILE MB000000 <> 0;                     ⇒ 語法錯誤
```

・於式子左邊編寫數值，或於右邊編寫暫存器時

```
WHILE 1 = = MB000000;                    ⇒ 語法錯誤
WHILE MB000000 = = MB000001;             ⇒ 語法錯誤
```

・未加上數值比較指令時

```
WHILE MB000000;                          ⇒ 語法錯誤
WHILE (0);                               ⇒ 語法錯誤
```

・編寫多個數值比較指令時

```
WHILE (MB000000 = = 0) & (MB000001 = = 1);      ⇒ 語法錯誤
```

◆ 比較整數 / 長整數 / 實數型資料

■ 格式

可使用所有的數值比較指令 (==、<>、>、<、>=、<=)。

暫存器可編寫在式子左邊或右邊。

```
WHILE MW00000 = = 3;             "MW00000 ＝ 3
WHILE ML00000 <> ML00002;        "ML00000 ≠ ML00002
WHILE 1.23456 >= MF00000;        "1.23456 ≧ MF00000
```

■ **條件式運算**

可編寫數值運算或邏輯運算。

```
WHILE MW00000 = = (MW00001/3);              "MW00000 ＝ (MW00001÷3)
WHILE (ML00000 & F0000000H) <> ML00002;     "(ML00000 Λ F0000000H) ≠ ML00002
WHILE 1.23456 >= (MF00000 * MF00002);       "1.23456 ≧ (MF00000×MF00002)
```

■ **語法錯誤範例**

發生下述情況時，將出現語法錯誤。

・式子兩邊皆只有編寫常數時

```
WHILE 0 = = 3;                              ⇒ 語法錯誤
WHILE (3.14 * 2 * 1000) > 9000.0;           ⇒ 語法錯誤
```

・未加上數值比較指令時

```
WHILE MW000000;                             ⇒ 語法錯誤
WHILE (-1);                                 ⇒ 語法錯誤
```

・編寫多個數值比較指令時

```
WHILE (MW00000 < 0) & (MW000001 > 0);       ⇒ 語法錯誤
```

程式範例

以下為包含 1 次掃描 WAIT 的循環 (WHILE、WENDX) 指令的程式範例。

下述程式係以暫存器 ML00000 累加至 100 為例。

```
ML00000 = 0
WHILE ML00000 == 100;          " 循環指令 "
ML00000 = ML00000 + 1;         "ML00000 增量 "
WENDX;                         "1 次掃描 WAIT 後，結束循環指令 "
END;
```

並列執行 (PFORK、JOINTO、PJOINT)

並列執行 (PFORK) 是一項可針對您所指定的標籤區塊並列執行的指令。所有作業並列處理完成後，就會和 JOINTO 指令所指定的標籤互相組合。並列處理指令最多可指定 8 個並列。如欲進一步瞭解標籤，請參閱以下章節。

☞ 區塊的格式 (第 5-2 頁)

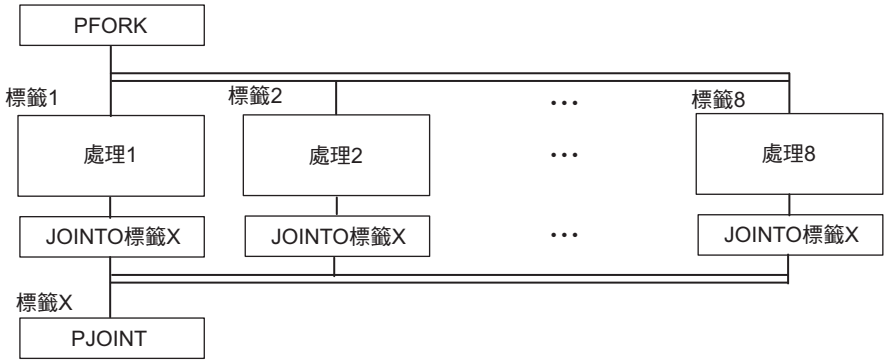


圖 6.63 下達並列執行 (PFORK、JOINTO、PJOINT) 指令的方法

利用上述指令，即可並列執行 PFOTK 指令所指定的標籤區塊 (處理 1、處理 2、處理 3、…)。所有作業並列處理完成後，就會和 JOINTO 指令所指定的標籤互相組合。PFORK 指令可並列執行軸移動指令及序列，或者可任意指定軸移動指令和並列執行軸移動指令。

■ PFORK 指令的前一個指令

PFORK 指令的前一個指令：FMX、ABS/INC、F 指令、IFP、PLN、IDH、IAC/IDC… 等數值將會被並列執行指令所執行的並列執行作業所引用。或者，可依不同的並列來指定指令。當式子組合後，最左側的處理數值就會被引用。

■ 子程式中的並列執行指令

若要在子程式內並列執行執行指令，需遵守以下條件。

- 子程式最多可編寫 8 項並列。
  可執行的並列數量依主程式所設定的並列模式而異。
  可執行的並列數過多時，運動程式就會發出警告。
- MSEE 指令只能在起始標籤所指定的區塊中編寫。

```
      PFORK 0001 0002;
0001:MVS [A1]100.[C1]100.;
       JOINTO 0003;
0002:IOW MW10000= =1;
       JOINTO 0003;
0003:PJOINT;
```

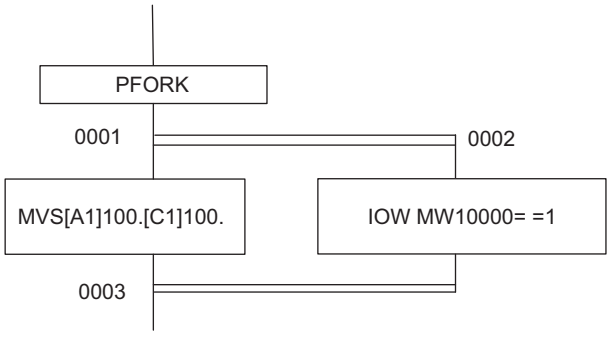


圖 6.64 子程式中的並列執行指令

重要

- 若同一個程式包含多個標籤時，就會發生錯誤 (標籤被重覆定義)。
- 若 PFORK 的分歧數和標籤數不同，就會發生錯誤。

格式

以下為並列執行 (PFORK、JOINTO、PJOINT) 指令之格式。

```
PFORK  標籤 1  標籤 2  標籤 3・・・・・

標籤 1：處理 1
        JOINTO  標籤 X；
標籤 2：處理 2
        JOINTO  標籤 X；
標籤 3：處理 3
        JOINTO  標籤 X；
標籤 X：PJOINT
```

程式範例

以下為並列執行 (PFORK、JOINTO、PJOINT) 指令的程式範例。

```
   MOV [A1]100. [B1]150.;
   MVS [A1]200. [B1]250. F1000;
   PFORK 0001 0002 0003;
0001:MVS [A1]300. [B1]100.
   JOINTO 0004;
0002:MW12345=MW10000+MW10002;
   IOW MB120001= =1;
   JOINTO 0004;
0003:MVS [C1]100. [D1]100. F3000;
   JOINTO 0004;
0004:PJOINT;
   MOV [A1]500. [B1]500. [C1]500.;
        .
        .
```

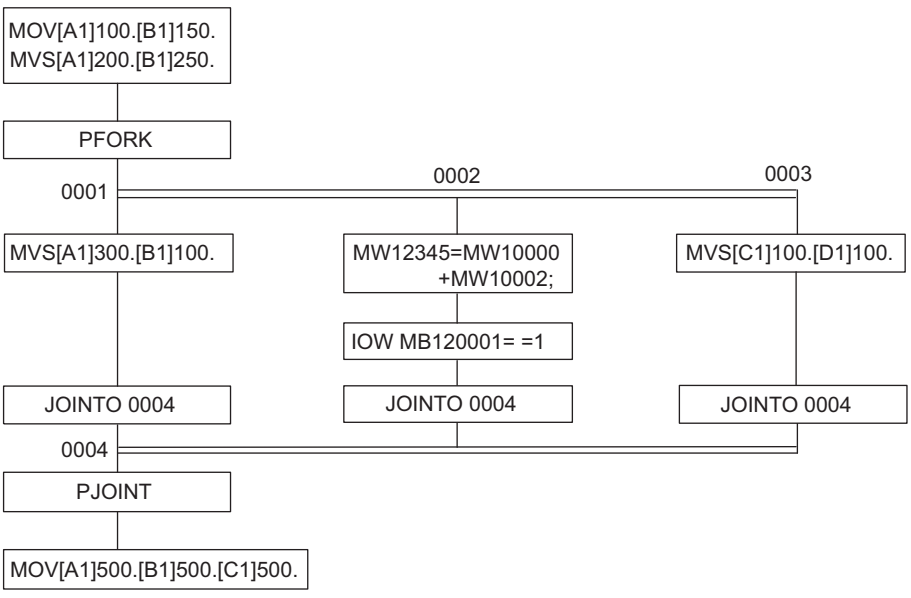


圖 6.65 並列執行 (PFORK、JOINTO、PJOINT) 指令的程式範例

選擇執行 (SFORK、JOINTO、SJOINT)

選擇執行 (SFORK、JOINTO、SJOINT) 就是當您所指定的條件式滿足時，就會執行接在 「?」後面的標籤區塊之指令。當完成各項處理作業後，就會和 JOINTO 指令所指定的標籤區塊組合。指定條件式時，最多可指定 16 個 (包含 DEFAULT)。
若所有的條件式皆滿足，即執行 「DEFAULT?」後面的標籤區塊。
DEFAULT 僅可指定為最後一個條件式。
運動程式可省略 DEFAULT 指令之指定內容，不過序列程式則無法省略。

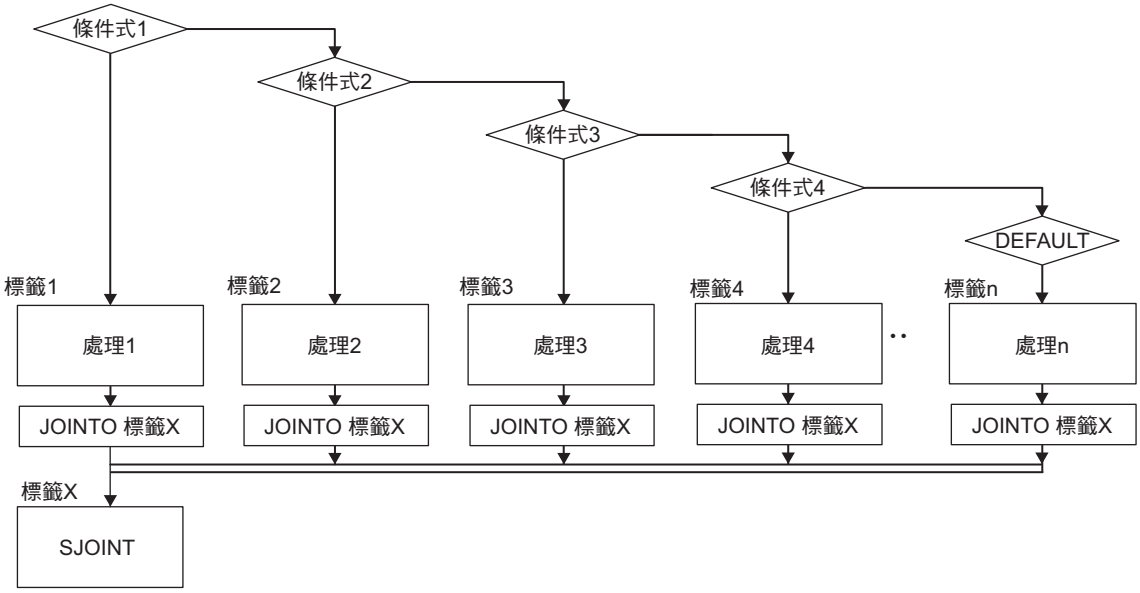


圖 6.66 選擇執行 (SFORK、JOINTO、SJOINT) 的指令方法

補充

1. 條件式將從 「條件式 1」開始依序進行判斷。因此，即使多個條件式同時成立，會先執行第一個成立的條件式的標籤處理作業。
2. 運動程式若要使用 SFORK，條件式則一定要必寫成立條件。條件若不成立，將在 SFORK 指令的區塊等待，直到條件成立為止。

格式

以下為選擇執行 (SFORK、JOINTO、SJOINT) 指令的格式。

SFORK 條件式 1？標籤 1，條件式 2？標籤 2，條件式 3？標籤 3，條件式 4？標籤 4，
・・・，DEFAULT? 標籤 n；

標籤 1：處理 1
JOINTO 標籤 X
標籤 2：處理 2
JOINTO 標籤 X
標籤 3：處理 3
JOINTO 標籤 X
標籤 4：處理 4
JOINTO 標籤 X
.
.
標籤 n：處理 n
JOINTO 標籤 X
標籤 X：SJOINT

以下為適用於選擇執行 (SFORK) 指令之條件式。

◆ 比較位元型資料

■ 格式

數值比較指令以 == (等於) 來表示。

式子左邊編寫暫存器，右邊則寫 0 或 1。

MB000000 = = 0? 標籤　"MB000000 = 0
MB000000 = = 1? 標籤　"MB000000 = 1

■ 條件式運算

編寫 &、|、! (邏輯積、邏輯和、反轉) 邏輯演算。

(MB000000 & MB000001) = = 1? 標籤　　"MB000000 = 1 且　　MB000001 = 1
(MB000000 & !MB000001) = = 1? 標籤　　"MB000000 = 1 且　　MB000001 = 0
(MB000000 ｜ MB000001) = = 1? 標籤　　"MB000000 = 1 或是　MB000001 = 1
(MB000000 ｜ !MB000001) = = 1? 標籤　　"MB000000 = 1 或是　MB000001 = 0

■ 語法錯誤範例

發生下述情況時，將出現語法錯誤。

・在數值比較指令中編寫 <>(不等於)

MB000000 <> 0? 標籤　⇒ 語法錯誤

・於式子左邊編寫數值，或於右邊編寫暫存器時

1 = = MB000000? 標籤　⇒ 語法錯誤
MB000000 = = MB000001? 標籤　⇒ 語法錯誤

・未加上數值比較指令時

MB000000? 標籤　⇒ 語法錯誤
(0)? 標籤　⇒ 語法錯誤

・編寫多個數值比較指令時

(MB000000 = = 0) & (MB000001 = = 1)? 標籤　⇒ 語法錯誤

◆ 比較整數 / 長整數 / 實數型資料

■ 格式

可使用所有的數值比較指令 (==、<>、>、<、>=、<=) 等。

暫存器可編寫在式子左邊或右邊。

MW00000 = = 3? 標籤　"MW00000 ＝ 3
ML00000 <> ML00002? 標籤　"ML00000 ≠ ML00002
1.23456 >= MF00000? 標籤　"1.23456 ≧ MF00000

■ 條件式運算

可編寫數值運算或邏輯運算。

MW00000 = = (MW00001/3)? 標籤	**"MW00000 ＝ (MW00001÷3)**
(ML00000 & F0000000H) <> ML00002? 標籤	**"(ML00000 Λ F0000000H) ≠ ML00002**
1.23456 >= (MF00000 * MF00002)? 標籤	**"1.23456 ≧ (MF00000×MF00002)**

■ 語法錯誤範例

發生下述情況時，將出現語法錯誤。

・式子兩邊皆只有編寫常數時

0 = = 3? 標籤	**⇒ 語法錯誤**
(3.14 * 2 *1000) > 9000.0? 標籤	**⇒ 語法錯誤**

・未加上數值比較指令時

MW000000? 標籤	**⇒ 語法錯誤**
(-1)? 標籤	**⇒ 語法錯誤**

・編寫多個數值比較指令時

(MW00000 < 0) & (MW000001 > 0)? 標籤	**⇒ 語法錯誤**

程式範例

以下為選擇執行 (SFORK、JOINTO、SJOINT) 指令的程式範例。

```
   MOV [A1]100.[B1]150.;
   MVS [A1]200.[B1]250.F1000;
   SFORK  MW00100= =1 ? 0001,MW00100= =2 ? 0002,MW00100= =3 ? 0003,DEFAULT ? 0004;
0001:MVS [A1]300.[B1]100.F3000;
   JOINTO 0005
0002:MVS [A1]300.[C1]100.F3000;
   JOINTO 0005
0003:MVS [C1]300.[S]100.F3000;
   JOINTO 0005
0004:JOINTO 0005;
0005:SJOINT;
MOV[A1]500.[B1]500.[C1]500.
```

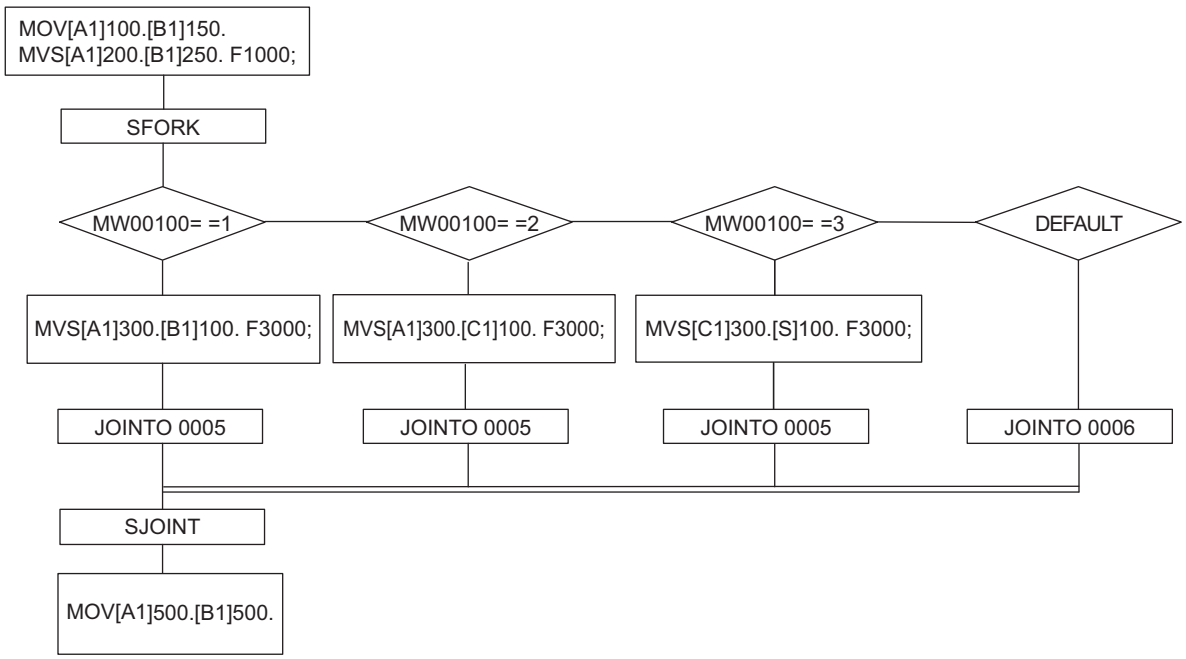


圖 6.67 選擇執行 (SFORK、JOINTO、SJOINT) 指令的程式範例

叫出運動子程式 (MSEE)

叫出運動子程式 (MSEE) 就是一項利用運動程式叫出已經預先記憶在運動程式記憶體中子程式的指令。
子程式最多可叫出 8 個巢狀式。

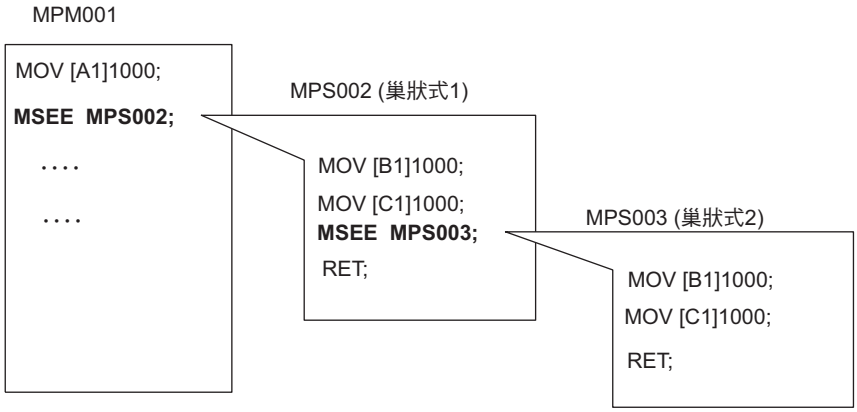


圖 6.68 叫出子程式

子程式的末尾必須指定為結束子程式 (RET) 指令。

重要

關於子程式的限制條件
利用 MSEE 指令叫出主程式時，則無法被執行。

格式

以下為 MSEE 指令的格式。

MSEE MPS 子程式編號；

項目	適用的資料
子程式編號	利用數值 001~512 來指定

程式範例

以下為 MSEE 指令的程式範例 (叫出運動子程式 MPS101 時)。

MSEE MPS101;

叫出序列子程式 (SSEE)

叫出序列子程式 (SSEE) 就是一項利用序列程式叫出已經預先記憶在序列程式記憶體中子程式的指令。
子程式最多可叫出 8 個巢狀式。

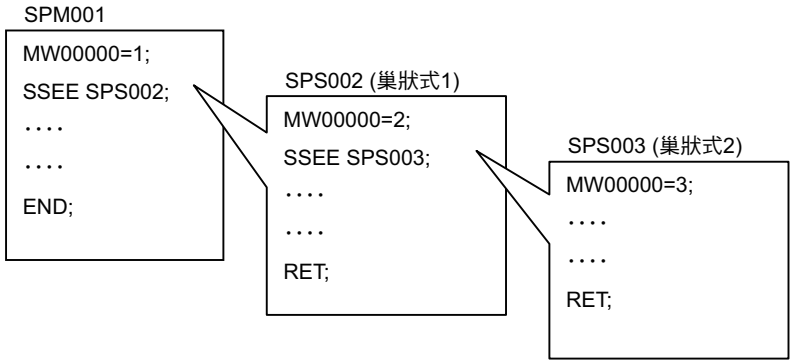


圖 6.69 叫出子程式

子程式的末尾必須指定為結束子程式 (RET) 指令。

註記

子程式的限制條件
在子程式中編寫序列程式時，需遵守以下的限制條件。
利用 SSEE 指令叫出主程式時，則無法被執行。

格式

以下為 SSEE 指令的格式。

SSEE SPS 子程式編號；

項目	適用的資料
子程式編號	利用數值 001~512 來指定

程式範例

以下為 SSEE 指令的程式範例 (叫出序列子程式 SPS101)。

SSEE SPS101;

從運動程式中叫出使用者函數 (UFC)

從運動程式叫出使用者函數 (UFC) 是一項利用運動程式來叫出使用者函數 (階梯圖程式) 的指令。
當使用者函數執行完成後，就會進入 UFC 指令的下一個區塊。

重要

從運動程式叫出使用者函數指令可用來判定輸出位元 YB000000 是否為使用者函數的結束 (Complete Bit)。

- 當 YB000000 變為關閉，且使用者函數已經結束時
  使用者函數將被視為尚未完成，當執行下一次掃描時將再次叫出使用者函數。
- 當 YB000000 變為開啟，且使用者函數已經結束時
  使用者函數被視為已完成，就會進入 UFC 指令的下一個區塊。

格式

以下為 UFC 指令的格式。

UFC 函數名稱 輸入資料，輸入位址，輸出資料 ;

項目	適用的資料
函數名稱	ASCII 8 位元組
輸入資料	最多 16 筆資料 (至少 1 筆資料)
輸入位址	最多 1 個位址
輸出資料 *	最多 16 筆資料 (至少 1 筆資料)

* 輸入位址的格式可省略。[輸入資料、輸出資料] 代表無輸入位址。
輸入資料、輸出資料均最少需要 1 筆。

程式範例

以下為 UFC 指令的程式範例。

```
UFC KANSUU  MB000000 IW0010 MB000002 ,  MA00100  ,
     函數名稱          輸入資料                輸入位址
MB000001 MW00200 ML00201;
          輸出資料
```

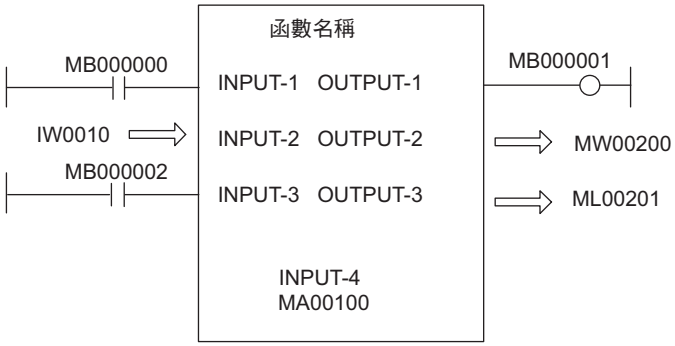
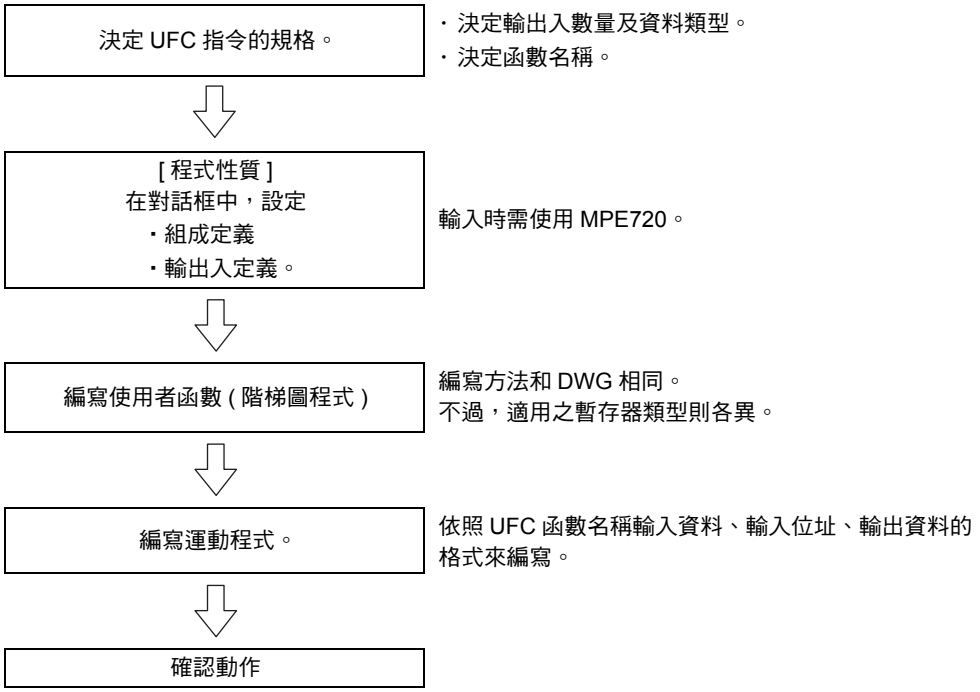


圖 6.70 叫出使用者函數 (UFC) 指令的程式範例

UFC 指令編寫步驟

以下為 UFC 指令之編寫步驟。



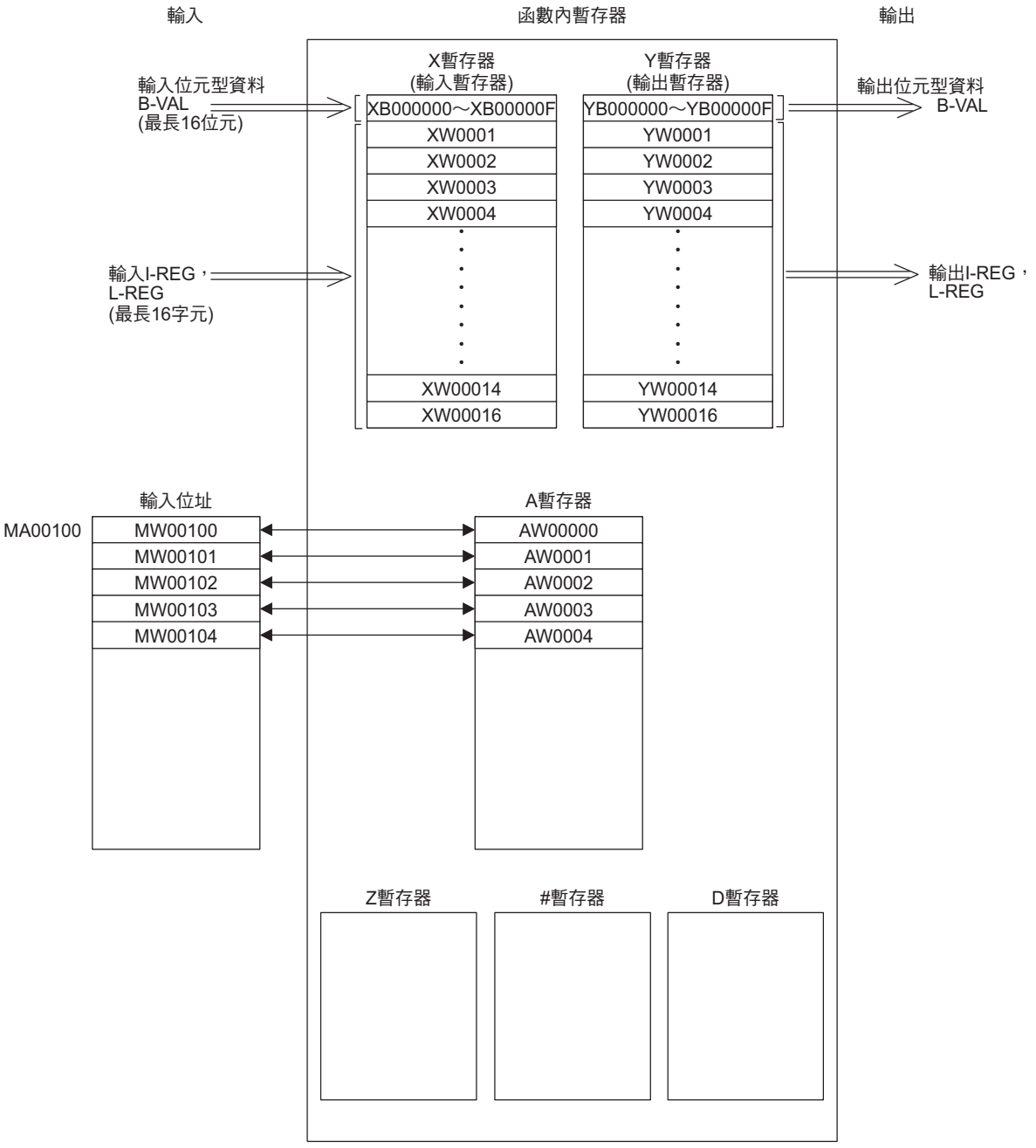
適用於使用者函數之暫存器資料類型

下表為適用之資料類型。

資料類型	類型
B	位元型
W	整數型
L	長整數型
Q	4 長整數型
F	實數型
D	倍精度實數型

輸出入暫存器和函數內暫存器之間的關係

以下為 UFC 指令所指定的輸出入暫存器和函數暫存器之間的對應關係。



下表為各種函數所適用之 12 種暫存器。

表 6.2 函數暫存器

種別	名稱	指定方法	內容	特性
X	**函數輸入暫存器**	XB,XW,XL,XQ,XF, XDnnnnn	輸入至函數 ・輸入位元：XB000000~XB00000F ・輸入整數：XW00001~XW00016 ・長整數：XL00001~XL00015 ・4 長整數：XQ00001~XQ00013 ・實數：XF00001~XF00015 ・倍精度實數：XD00001~XD00013	個別函數
Y	**函數輸出暫存器**	YB,YW,YL,YQ,YF, Ydnnnnn	輸入至函數 ・輸入位元：YB000000~YB00000F ・輸入整數：YW00001~YW00016 ・長整數：YL00001~YL00015 ・4 長整數：YQ00001~YQ00013 ・實數：YF00001~YF00015 ・倍精度實數：YD00001~YD00013	
Z	**函數內部暫存器**	ZB,ZW,ZL,ZQ,ZF, ZDnnnnn	每個函數所預設的內部暫存器。 適合作為函數內部處理之用。	
A	**函數外部暫存器**	AB,AW,AL,AQ,AF, ADnnnnn	以位址輸入值為基本位址之外部暫存器。 用來連結 (S、M、I、O、#、DAnnnnn)。 暫存器編號 nnnnn 為 10 進位制格式。	
#	**# 暫存器**	#B,#W,#L,#Q,#F, #Dnnnnn	僅能利用程式參照的暫存器。 只有符合的 DWG 能夠參照。 使用者可利用 MPE720 來指定實際的使用範圍。 暫存器編號 nnnnn 為 10 進位制格式。	
D	**D 暫存器**	DB,DW,DL,DQ,DF, DDnnnnn	每個 DWG 預設的暫存器。 只有符合的 DWG 能夠參照。 使用者可利用 MPE720 來指定實際的使用範圍。 暫存器編號 nnnnn 為 10 進位制格式。	
S	**系統暫存器**	SB,SW,SL,SQ,SF, SDnnnnn	與 DWG 暫存器相同 此種暫存器與 DWG/ 函數共用，因此如果不同優先位準的 DWG 要參照同一個函數時，需特別注意使用方法。 暫存器編號 nnnnn 為 10 進位制格式。 暫存器編號 hhhhh 為 16 進位制格式。	共用DWG
M	**資料暫存器**	MB,MW,ML,MQ,MF, MDnnnnnnn		
G	**暫存器**	GB,GW,GL,GQ,GF, GDnnnnnnn		
I	**輸入暫存器**	IB,IW,IL,IQ,IF, IDhhhhh		
O	**輸出暫存器**	OB,OW,OL,OQ,OF, ODhhhhh		
C	**常數暫存器**	CB,CW,CL,CQ,CF, CDnnnnn		

(註) SA、MA、IA、OA、DA、#A、CA 等亦適用於函數內部。

以下為輸出入暫存器之資料收受範例。

運動程式編寫方法

```
UFC  TESTFUNC  DB000000 DB000001 MW00030 MW00032, MA00100,DB000002 MW00040
```

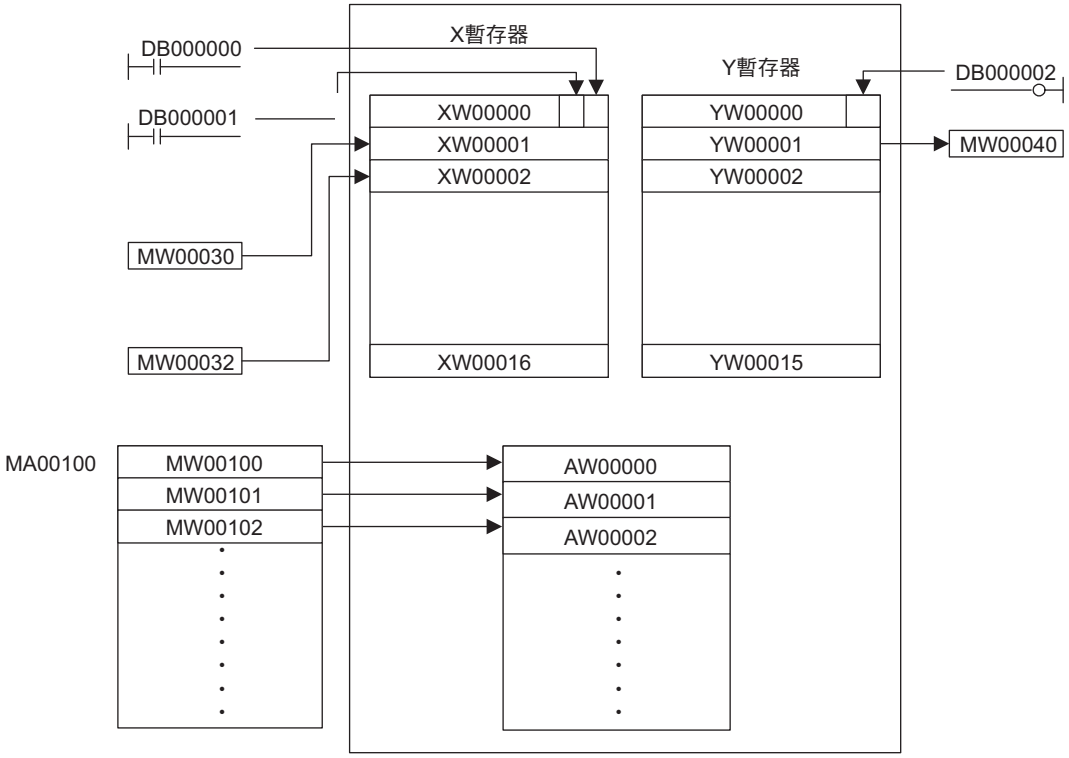


圖 6.71 運動程式編寫方法

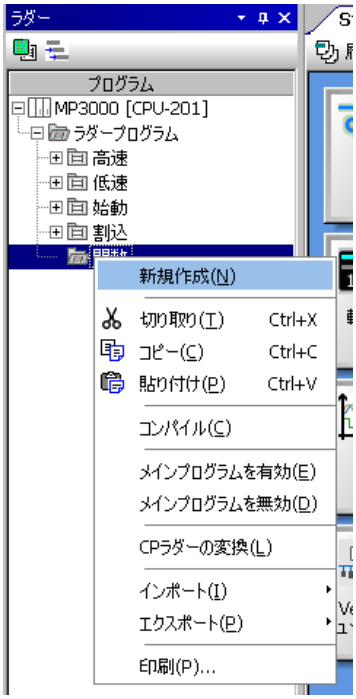
使用者函數之編寫

以下表所示的使用者函數規格為例，說明使用者函數之編寫步驟。

規格	運動程式
指定伺服器軸編號和速度資料，然後再設定參數 (快速進給速度 OL□□□10)。	MW00030= 伺服器軸 No. (1 or 2) ML00032= 快速進給速度 UFC FUNC-T1 MW00030 ML00032,,DB000001;

以下為使用者函數編寫步驟。

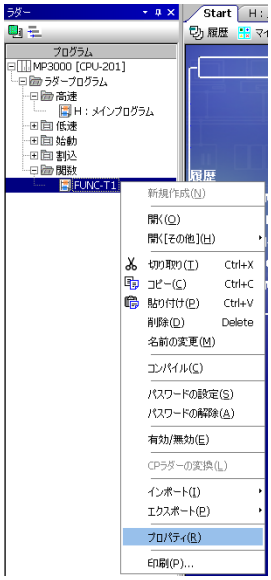
1. 叫出 [階梯圖程式] 子視窗，將游標指到函數上，按一下滑鼠右鍵，並點擊 [製作新專案 (N)]。



2. 進入 [製作新程式] 對話框中的 [程式編號] 方塊，輸入 「FUNC-T1」，並點擊 [OK] 鍵。

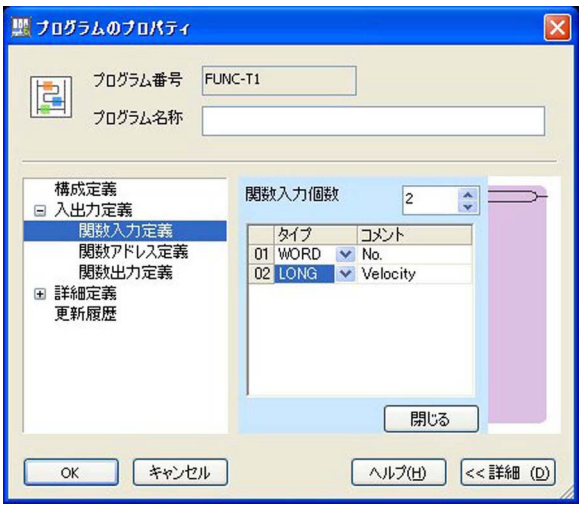
即可叫出 [階梯圖程式] 子視窗 (無任何程式的空白視窗)。

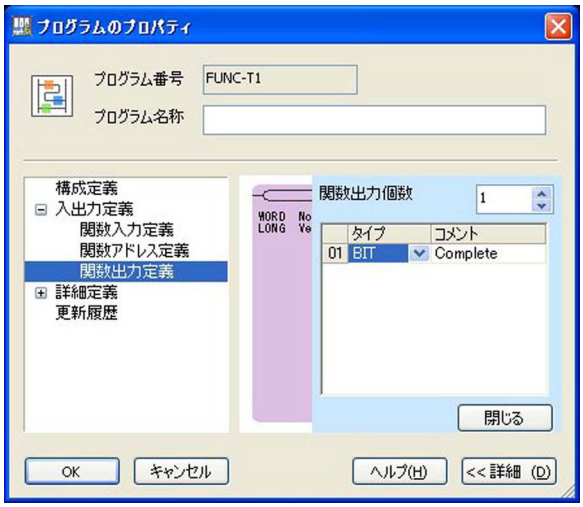
3. 將游標指到 [FUNC-T1]，按一下滑鼠右鍵，並選擇 [性質 (R)]。



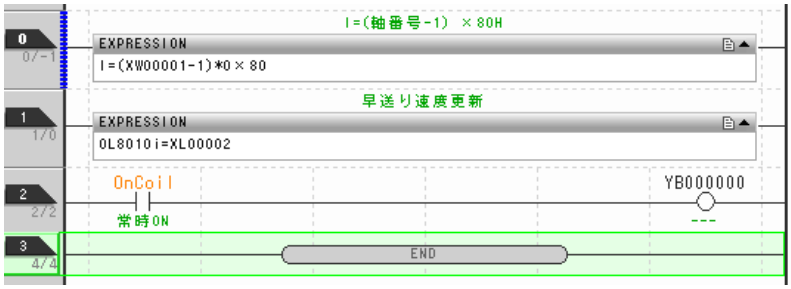
4. 點擊 [輸出入定義] 索引標籤，即可設定函數的輸出入數量及資料類型。

出現「UFC FUNC-T1 MW00030 ML00032,,DB000001;」時，請依下圖所示進行設定。

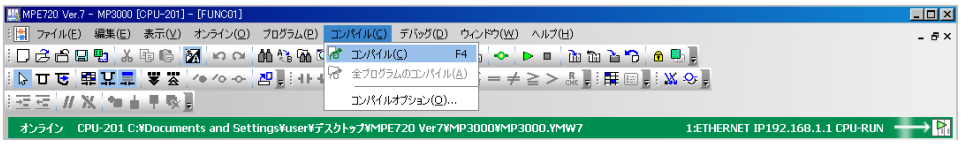




5. 關閉 DWG 組成定義視窗，在階梯圖畫面中編輯使用者程式。



6. 請從選單上選擇 [編譯 (C)] － [編譯 (C) F8]。



7. 進入運動程式編輯視窗，即可編寫可用來叫出使用者函數的程式。

```
MW00030 = 1;
ML00032 = 500;
UFC FUNC-T1 MW00030 ML00032, , DB000001;
END;
```

從運動程式叫出使用者函數的程式已全部編寫完成。
執行運動程式，以確認動作是否正確。

從序列程式叫出使用者函數 (FUNC)

從序列程式叫出使用者函數 (FUNC) 是一項利用序列程式叫出使用者函數 (階梯圖程式) 的指令。

格式

以下為 FUNK 指令的格式。

UFC 函數名稱 輸入資料 1 輸入資料 2 輸入資料 3・・・、輸入位址、
輸出資料 1 輸出資料 2 輸出資料 3・・・;

項目	適用的資料
函數名稱	ASCII 8 位元組
輸入資料	最多 16 筆資料 (至少 1 筆資料)
輸入位址	最多 1 個位址
輸出資料	最多 16 筆資料 (至少 1 筆資料)

(註)1. 上述的輸入資料、輸出資料皆可同時編寫多筆 (但最少為 1 筆)。輸入位址可省略。
省略輸入位址時，只要編寫逗號 "，"。
2. 上述指令可用來叫出使用者函數。在 FUNC 指令中，無論使用者函數是否已經執行結束，皆會進入「FUNC」的下一個區塊。

程式範例

以下為 FUNC 指令的程式範例。

本範例使用了 3 筆輸入資料、3 個輸入位址以及 3 筆輸出資料。

```
FUNC KANSUU  MB000000 IW0010 MB000020 ,  MA00100  ,
      函數名稱          輸入資料              輸入位址
MB000001 MW00201 ML00202;
        輸出資料
```

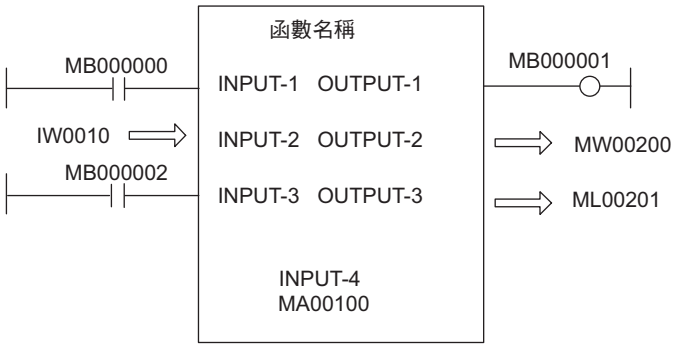


圖 6.72 叫出使用者函數 (FUNC) 指令的程式範例

結束程式 (END)

結束程式 (END) 是一項用來結束程式運轉的指令。
無法對此區塊重覆下達其他指令。
當執行完此區塊後，就會利用本指令來結束程式運轉。
若前一個區塊為移動指令，就會在執行到位確認後結束。

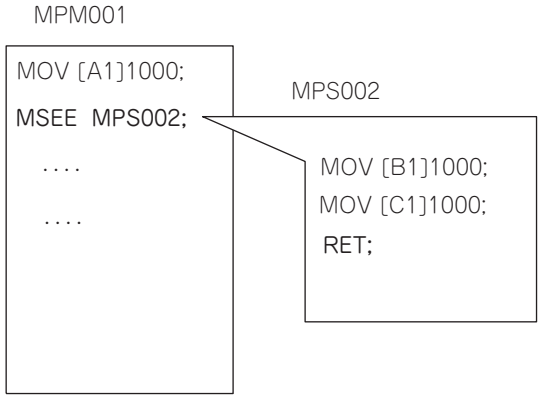
格式

以下為 END 指令的格式。

END;

結束子程式 (RET)

結束子程式 (RET) 是一種用來結束子程式的指令。
將進入叫出子程式的程式 (主程式或子程式) 的叫出運動子程式 (MSEE) 或叫出序列子程式 (SSEE) 的下一個區塊。



格式

以下為 RET 指令的格式。

RET;

等待時間 (TIM)

等待 (TIM) 是一種依照指定時間等待完成後即進入下一個區塊的指令。
可指定的時間範圍為 0.00~600.00 s。

格式

以下為 TIM 指令的格式。

TIM　　T 等待時間 ;

項目	單位	適用的資料
等待時間	0.01s	・利用立即值直接指定 ・利用整數型暫存器間接指定

程式範例

以下為 TIM 指令的程式範例。

```
MOV [A1]100;
TIM T250 ;
```

定位完成後，執行 TIM 指令。

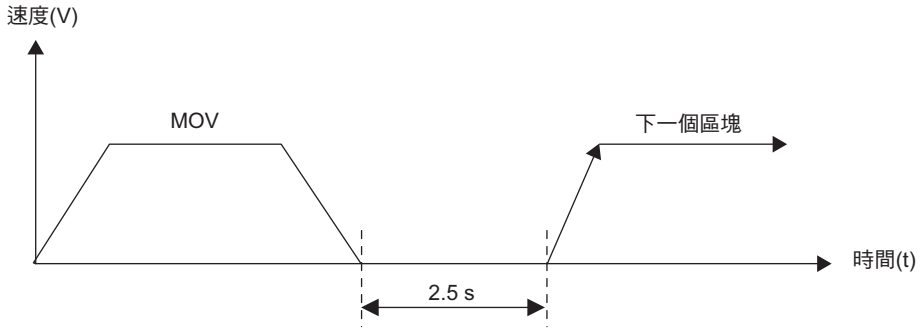


圖 6.73 TIM 指令的程式範例

等待時間 (TIM1MS)

等待 (TIM1MS) 時間是一種依照指定時間等待完成後即進入下一個區塊的指令。
時間設定單位為 1 ms，而時間指定範圍則為 0.000~60.000 s。

格式

以下為 TIM1MS 指令的格式。

TIM1MS　　T 等待時間 ;

項目	單位	適用的資料
等待時間	1ms	・利用立即值直接指定 ・整數型暫存器 (#、C 暫存器除外)

程式範例

以下為 TIM1MS 指令的程式範例。

```
MOV   [A1]1000;        " 定位
TIM1MS T5;             " 定位完成後，等待 5 ms
MOV [A1]1000;          " 定位

END;
```

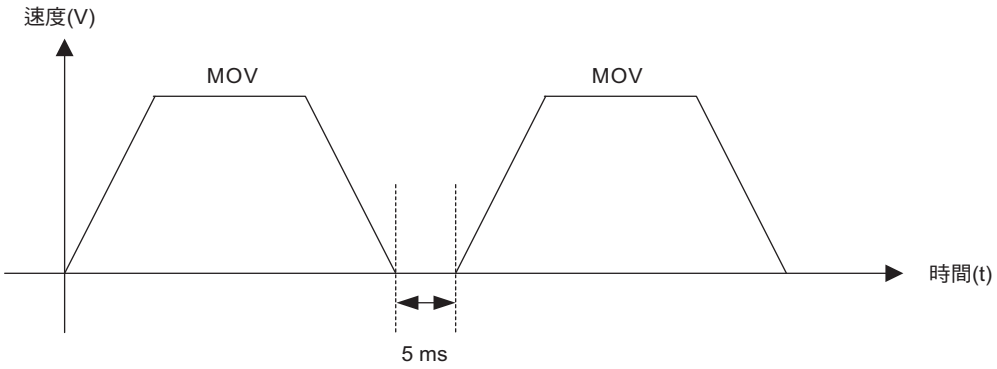


圖 6.74 TIM1MS 指令的程式範例

等待輸出入變數 (IOW)

等待輸出入變數 (IOW) 為待機直到條件式到達指定的狀態，若條件成立則進入下一個區塊的指令。

格式

以下為 IOW 指令的格式。

```
IOW     IB00001&IB00002 = = 1;
        ①
```

內容	使用用途	適用的資料
①	條件式	・整數型、長整數型、實數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

以下為 IOW 指令適用之條件式。

◆ 比較位元型資料

■ 格式

數值比較指令以 ==(等於) 來表示。

式子左邊編寫暫存器，右邊則寫 0 或 1。

```
IOW MB000000 = = 0;    "MB000000 = 0
IOW MB000000 = = 1;    "MB000000 = 1
```

■ 條件式運算

編寫 &、|、! (邏輯積、邏輯和、反轉) 邏輯演算。

```
IOW (MB000000 &  MB000001) = = 1;     "MB000000 = 1 且MB000001 = 1
IOW (MB000000 & !MB000001) = = 1;     "MB000000 = 1 且MB000001 = 0
IOW (MB000000 ｜ MB000001) = = 1;     "MB000000 = 1 或是MB000001 = 1
IOW (MB000000 ｜ !MB000001) = = 1;    "MB000000 = 1 或是MB000001 = 0
```

■ 語法錯誤範例

發生下述情況時，將出現語法錯誤。

・在數值比較指令中編寫 <>(不等於)

```
IOW MB000000 <> 0;                    ⇒ 語法錯誤
```

・ 於式子左邊編寫數值，或於右邊編寫暫存器時

```
IOW 1 = = MB000000;                   ⇒ 語法錯誤
IOW MB000000 = = MB000001;            ⇒ 語法錯誤
```

・未加上數值比較指令時

```
IOW MB000000;                         ⇒ 語法錯誤
IOW (0);                              ⇒ 語法錯誤
```

・編寫多個數值比較指令時

IOW (MB000000 = = 0) & (MB000001 = = 1);　　⇒ 語法錯誤

◆ 比較整數 / 長整數 / 實數型資料

■ 格式

可使用所有的數值比較指令 (==、<>、>、<、>=、<=) 等。

暫存器可編寫在式子左邊或右邊。

```
IOW MW00000 = = 3;          "MW00000 ＝ 3
IOW ML00000 <> ML00002;     "ML00000 ≠ ML00002
IOW 1.23456 >= MF00000;     "1.23456 ≧ MF00000
```

■ 條件式運算

可編寫數值運算或邏輯運算。

```
IOW MW00000 = = (MW00001/3);              "MW00000 ＝ (MW00001÷3)
IOW (ML00000 & F0000000H) <> ML00002;     "(ML00000 Λ F0000000H) ≠ ML00002
IOW 1.23456 >= (MF00000 * MF00002);       "1.23456 ≧ (MF00000×MF00002)
```

■ 語法錯誤範例

發生下述情況時，將出現語法錯誤。

・式子兩邊皆只有編寫常數時

```
IOW 0 = = 3;                              ⇒ 語法錯誤
IOW (3.14 * 2 * 1000) > 9000.0;           ⇒ 語法錯誤
```

・未加上數值比較指令時

```
IOW MW000000;                             ⇒ 語法錯誤
IOW (-1);                                 ⇒ 語法錯誤
```

・編寫多個數值比較指令時

```
IOW (MW00000 < 0) & (MW000001 > 0);       ⇒ 語法錯誤
```

程式範例

以下為 IOW 指令的程式範例。

```
IOW (MB001001&MB001002) = = 1;
MOV [A1]1000;
```

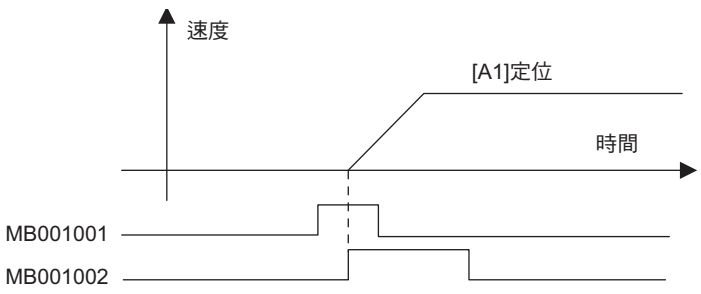


圖 6.75 IOW 指令的程式範例

1 次掃描 WAIT(EOX)

1 次掃描 WAIT(EOX) 是一種讓程式的執行在 1 次掃描的時間範圍內維持待機的指令。
EOX 指令以後的區塊，就會開始執行下一次掃描。

格式

以下為 EOX 指令的格式。

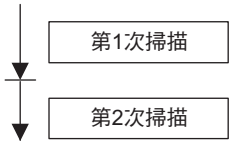
```
EOX;
```

程式範例

以下為 EOX 指令的程式範例。

・搭配使用序列指令

```
MW00000 = 100;
OB00010 = 1;
EOX;
OB00011 = 0;
```



・WHILE 指令使用範例

```
WHILE  OB00010 = = 1;
EOX;
WEND;
```

單一區塊關閉 (SNGD)/ 單一區塊開啟 (SNGE)

單一區塊關閉 (SNGD)/ 單一區塊開啟 (SNGE) 是一項用來指定除錯運轉時所執行的步進動作有效 / 無效的指令。
即使為單一區塊運轉模式，SNGD 和 SNGE 之間的區塊仍會持續運轉，且不會停止單一區塊的動作。


專有名詞解說

單一區塊運轉模式
單一區塊運轉模式可依區塊別分別執行單一區塊停止動作。

格式

以下為 SNGD 指令的格式。

```
SNGD;
想要連續執行的部分
SNGE;
```

程式範例

以下為 SNGD/SNGE 指令的程式範例。
在以下的程式範例中，即使為單一區塊運轉模式，SNGD 和 SNGE 之間的區塊 ①~③ 將進行連續運轉，而不執行單一區塊停止動作。

```
MVS  [A1]0  [B1]0;
SNGD;
MVS  [A1]100  [B1]200;   "①"
MB000101 = 1;            "②"
MB000102 = 1;            "③"
SNGE;
MB000103 = 1;
```

6.5 數值運算指令

數值運算指令包含 8 種指令，運動程式和序列程式皆適用。

下表為數值運算指令一覽表。

指令	名稱	格式	內容	運動程式	序列程式
=	代入	(結果)=(算式)	代入運算結果。此外，運算時的順序為由左而右 (無優先順序)。	○	○
+	加法	MW□ = MW□ + MW□;	執行整數 / 實數的加法。若為整數 / 實數混合，則會全部被當作實數來運算。	○	○
-	減法	MW□ = MW□ - MW□;	執行整數 / 實數的減法。若為整數 / 實數混合，則會全部被當作實數來運算。	○	○
+ +	加法擴充	MW□ = MW□ + + MW□;	執行整數的加法擴充。	○	○
- -	減法擴充	MW□ = MW□ - - MW□;	執行整數的減法擴充。	○	○
*	乘法	MW□ = MW□ * MW□;	執行整數 / 實數的乘法。若為整數 / 實數混合，則會全部被當作實數來運算。	○	○
/	除法	MW□ = MW□ / MW□;	執行整數 / 實數的除法。若為整數 / 實數混合，則會全部被當作實數來運算。	○	○
MOD	除法餘數運算	MW□ = MW□ / MW□; MW□ = MOD;	若在除法運算的下一個區塊下達指令，則餘數會被儲存在指定的暫存器中。	○	○

(註) 在格式的 □ 中加入暫存器編號。

如欲瞭解數值運算的優先順序，請參閱以下章節。

☞ **5.3 運算時的優先順序 (第 5-11 頁)**

代入 (=)

代入 (=) 就是將右邊的運算結果套用到式子左邊的暫存器中。

格式

以下為代入 (=) 的格式。

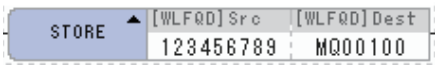
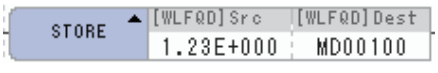
結果 = 算式 ;
① ②

內容	使用用途	適用之暫存器
①	結果	・位元型、整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	算式	・位元型、整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

程式範例

代入 (=) 指令適用於運動程式、序列程式以及階梯圖程式。

以下為代入 (=) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	MB001000 = 1;	
W	MW00100 = 12345;	
L	ML00100 = 1234567;	
F	MF00100 = 1.2345;	
Q	MQ00100 = 123456789;	
D	MD00100 = 1.234567;	

加法 (+)

加法 (+) 就是將式子右邊的整數或實數相加，然後再將結果儲存在左邊的暫存器中。能被放在式子右邊相加的除了暫存器，還有常數。即使為整數或實數混合，也會在左邊的資料類型進行儲存。

格式

以下為加法 (+) 指令的格式。

MW00101 = **MW00100** + **12345** ;
①　②　③

內容	使用用途	適用之暫存器
①	輸出資料	・整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入資料	・整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數
③	加法資料	

程式範例

加法 (+) 指令適用於運動程式、序列程式以及階梯圖程式。
以下為加法 (+) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00101 = MW00100+12345;	
L	ML00106 = ML00102+ML00104;	
F	MF00202 = MF00200+1.23456;	

補充　若您所運算的暫存器資料類型不同時，實際結果將取決於左邊的資料類型。如欲瞭解資料類型之相關說明，請參閱以下項目。

☞ 總體暫存器 (第 4-4 頁)
☞ 局部暫存器 (第 4-4 頁)

減法 (-)

減法 (-) 就是將式子右邊的整數或實數相減，然後再將結果儲存在左邊的暫存器中。能被放在式子右邊相加的除了暫存器，還有常數。即使為整數或實數混合，也會在左邊的資料類型進行儲存。

格式

以下為減法 (-) 指令的格式。

MW00101 = **MW00100** − **12345** ;
①　②　③

內容	使用用途	適用之暫存器
①	輸出資料	・整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入資料	・整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數
③	減法資料	

程式範例

減法 (-) 指令適用於運動程式、序列程式以及階梯圖程式。
以下為減法 (-) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00101 = MW00100-12345;	
L	ML00106 = ML00102-ML00104;	
F	MF00202 = MF00200-1.23456;	

加法擴充 (+ +)

加法擴充 (++) 是一項用來執行整數型數值加法運算的指令。
即使運算結果溢位，將回到負方向最大值繼續運算，且不會發生運算錯誤。
即使運算結果欠位，將回到正方向最大值繼續運算，且不會發生運算錯誤。
其他皆和加法 (+) 相同。

■ 整數型

10 進位制 ：0 → 1 ・・・32767 → -32768 ・・・-1 → 0
16 進位制 ：0000 → 0001 ・・・7FFF → 8000 ・・・FFFF → 0000

■ 長整數型

10 進位制 ：0 → 1 ・・・2147483647 → -2147483648 ・・・-1 → 0
16 進位制 ：00000000 → 00000001 ・・・7FFFFFFF → 80000000 ・・・FFFFFFFF → 00000000

■ 4 長整數型

10 進位制 ：0 → 1 ・・・9223372036854775807 → -9223372036854775808 ・・・-1 → 0
16 進位制 ：0000000000000000 → 0000000000000001 ・・・7FFFFFFFFFFFFFFF →
8000000000000000 ・・・FFFFFFFFFFFFFFFF → 0000000000000000

格式

以下為加法擴充 (++) 指令的格式。

MW00101 = MW00100 + + 12345;
① ② ③

內容	使用用途	適用之暫存器
①	輸出資料	・整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入資料	・整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數
③	加法資料	

(註) 利用實數型資料來編寫時，將造成編譯器錯誤。

程式範例

加法擴充 (++) 指令適用於運動程式、序列程式以及階梯圖程式。
以下為加法擴充 (++) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00101 = MW00100+ +1;	
L	ML00106 = ML00102+ +ML00104;	
F	–	–
Q	MQ00116 = MQ00108+ +MQ00112;	

減法擴充 (- -)

減法擴充 (--) 是一項用來執行整數型數值減法運算的指令。
即使運算結果溢位，將回到負方向最大值繼續運算，且不會發生運算錯誤。
即使運算結果為欠位，將回到正方向最大值繼續運算，而不會發生運算錯誤。
其他皆和減法 (-) 相同。

■ 整數型

10 進位制 ：0 → -1 ・・・-32768 → 32767 ・・・1 → 0
16 進位制 ：0000 → FFFF ・・・8000 → 7FFF ・・・0001 → 0000

■ 長整數型

10 進位制 ：0 → -1 ・・・-2147483648 → 2147483647 ・・・1 → 0
16 進位制 ：00000000 → FFFFFFFF ・・・80000000 → 7FFFFFFF ・・・00000001 → 00000000

■ 4 長整數型

10 進位制 ：0 → -1 ・・・-9223372036854775808 → 9223372036854775807 ・・・1 → 0
16 進位制 ：0000000000000000 → FFFFFFFFFFFFFFFF ・・・8000000000000000 →
7FFFFFFFFFFFFFFF ・・・0000000000000001 → 0000000000000000

格式

以下為減法擴充 (--) 指令的格式。

```
MW00101 = MW00100 - - 12345;
   ①        ②        ③
```

內容	使用用途	適用之暫存器
①	輸出資料	・整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入資料	・整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數
③	加法資料	

(註) 利用實數型資料來編寫時，將造成編譯器錯誤。

程式範例

減法擴充 (--) 指令適用於運動程式、序列程式以及階梯圖程式。
以下為減法擴充 (--) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00101 = MW00100- -1;	
L	ML00106 = ML00102- -ML00104;	
F	–	–
Q	MQ00116 = MQ00108- -MQ00112	

乘法 (*)

乘法 (*) 就是將式子右邊的整數或實數相乘，然後再將結果儲存在左邊的暫存器中。能被放在式子右邊相乘的除了暫存器，還有常數。即使為整數或實數混合，也會在左邊的資料類型進行儲存。

格式

以下為乘法 (*) 指令的格式。

MW00101 = **MW00100** * **12345** ;
①　②　③

內容	使用用途	適用之暫存器
①	輸出資料	・整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入資料	・整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數
③	乘法資料	

程式範例

乘法 (*) 指令適用於運動程式、序列程式以及階梯圖程式。
以下為乘法 (*) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00102 = MW00100 * MW00101	
L	ML00106 = ML00102 * ML00104;	
F	MF00202 = MF00200 * 1.23456;	

除法 (/)

除法 (/) 就是將式子右邊的整數或實數相除，然後再將結果儲存在左邊的暫存器中。能被放在式子右邊相除的除了暫存器，還有常數。即使為整數或實數混合，也會在左邊的資料類型進行儲存。

格式

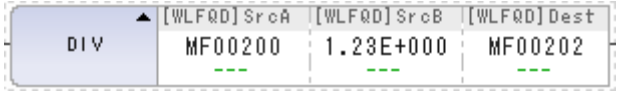
以下為除法 (/) 指令的格式。

MW00101 = **MW00100** / **12345** ;
①　②　③

內容	使用用途	適用之暫存器
①	輸出資料	・整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入資料	・整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數
③	除法資料	

程式範例

除法 (/) 指令適用於運動程式、序列程式以及階梯圖程式。
以下為除法 (/) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00102 = MW00100/MW00101;	
L	ML00106 = ML00102/ML00104;	
F	MF00202 = MF00200/1.23456;	

除法餘數 (MOD)

除法餘數 (MOD) 就是指定為除法運算指令的下一個區塊時，MOD 將被視為除法餘數儲存於您所指定的變數中。

格式

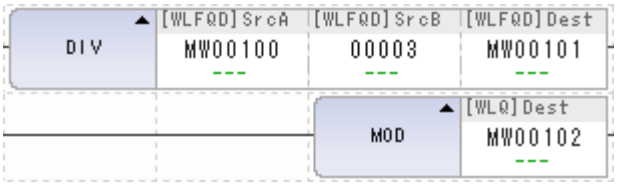
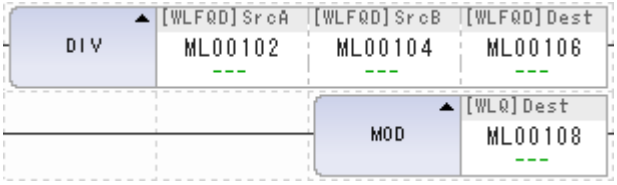
以下為 MOD 指令的格式。

```
MW00001 = 1000 / 999;
MW00002 = MOD;
    ①
```

內容	使用用途	適用之暫存器
①	輸出資料	・整數型、長整數型的所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器

程式範例

MOD 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 MOD 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00101 = MW00100/3; MW00102 = MOD;	
L	ML00106 = ML00102/ML00104; ML00108 = MOD;	
F	–	–

範例

MOD 指令的程式範例 (長整數)

```
ML00106 = ML00100 * ML00102/ML00104;
(173575)  (100000)  (60000)  (34567)
ML00108 = MOD;
(32975)
```


重要

MOD 指令必須指定在除法運算指令的下一個區塊。若在其他區塊執行，將無法保證運算結果一定正確。

6.6 邏輯運算指令

「邏輯運算指令」是一種用來運算數值真偽的指令。
邏輯運算指令包含 4 種指令，適用於運動程式及序列程式的任一種程式。

下表為邏輯運算指令一覽表。

指令	名稱	格式	內容	運動程式	序列程式
|	OR (邏輯和)	MB□ = MB□ | MB□; MB□ = MB□ | 1; MW□ = MW□ | MW□; MW□ = MW□ | 00FFH; ML□ = ML□ | ML□; ML□ = ML□ | 00FF00FFH; MQ□ = MQ□ | MQ□; MQ□ = MQ□ | 00FF00FF 00FF00FFH;	編寫位元 / 整數的邏輯和。	○	○
&	AND (邏輯積)	MB□ = MB□ & MB□; MB□ = MB□ & 1; MW□ = MW□ & MW□; MW□ = MW□ & 00FFH; ML□ = ML□ & ML□; ML□ = ML□ & 00FF00FFH; MQ□ = MQ□ & MQ□; MQ□ = MQ□ & 00FF00FF 00FF00FFH;	編寫位元 / 整數的邏輯積。	○	○
^	XOR (互斥或)	MW□ = MW□ ^ MW□; MW□ = MW□ ^ 00FFH; ML□ = ML□ ^ ML□; ML□ = ML□ ^ 00FF00FFH; MQ□ = MQ□ ^ MQ□; MQ□ = MWQ□ ^ 00FF00FF 00FF00FFH;	編寫整數的互斥或。	○	○
!	NOT (反轉)	MB□ = !MB□; MB□ = !1; MW□ = !MW□; MW□ = !00FFH; ML□ = !ML□; ML□ = !00FF00FFH; MQ□ = !MQ□; MQ□ = !00FF00FF 00FF00FFH;	編寫位元反轉後的數值。	○	○

(註) 在格式的□中加入暫存器編號。

雖然可使用與數值運算混合的算式，但無法執行實數運算。
如欲瞭解數值運算的優先順序，請參閱以下章節。

☞ **5.3 運算時的優先順序 (第 5-11 頁)**

邏輯和 (|)

邏輯和 (|) 可用來計算目前的運算結果和您所指定暫存器的邏輯和，而得到運算結果。實數暫存器無法使用。

表 6.3 邏輯和真值表 (A=B|C)

B	C	A
0	0	0
0	1	1
1	0	1
1	1	1

格式

以下為邏輯和 (|) 指令的格式。

MW00100 = DW00102 | AAAAH;
① ② ③

內容	使用用途	適用之暫存器
①	輸出資料	・位元型、整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②，③	輸入資料	・位元型、整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

程式範例

邏輯和 (|) 指令適用於運動程式、序列程式以及階梯圖程式。
以下為邏輯和 (|) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	MB001000 = MB001010 | MB001011;	
W	MW00100 = MW00101 | MW00102	
L	ML00106 = ML00102 | ML00104;	
F	–	–

邏輯積 (&)

邏輯積 (&) 可用來計算目前的運算結果與您所指定暫存器的邏輯積，而得到運算結果。實數暫存器無法使用。

表 6.4 邏輯積真值表 (A=B&C)

B	C	A
0	0	0
0	1	0
1	0	0
1	1	1

格式

以下為邏輯積 (&) 指令的格式。

MW00100 = DW00102 & AAAAH;
①　②　③

內容	使用用途	適用之暫存器
①	輸出資料	・位元型、整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②，③	輸入資料	・位元型、整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

程式範例

邏輯積 (&) 指令適用於運動程式、序列程式以及階梯圖程式。
以下為邏輯積 (&) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	MB001000 = MB001010&MB001011;	
W	MW00101 = MW00100& 00FFH;	
L	ML00106 = ML00102& ML00104;	
F	–	–

互斥或 (^)

互斥或 (^) 可用來計算目前的運算結果和您所指定暫存器的互斥或，而得到運算結果。實數暫存器無法使用。

表 6.5 互斥或的真值表 (A=B^C)

B	C	A
0	0	0
0	1	1
1	0	1
1	1	0

格式

以下為互斥或 (^) 指令的格式。

MW00100 = DW00102 ^ AAAAH;
①　②　③

內容	使用用途	適用之暫存器
①	輸出資料	・整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②，③	輸入資料	・整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

程式範例

互斥或 (^) 指令適用於運動程式、序列程式以及階梯圖程式。
以下為互斥或 (^) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00101 = MW00100 ^ 00FFH;	
L	ML00106 = ML00102 ^ ML00104;	
F	–	–

反轉 (!)

反轉 (!) 指令可將您所指定的暫存器資料反轉，而得到運算結果。實數暫存器不適用。

格式

以下為反轉 (!) 指令的格式。

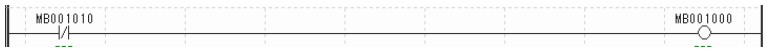
MB001000 = ! **MB001010**;
①　②

內容	使用用途	適用之暫存器
①	輸出資料	・位元型、整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入資料	・位元型、整數型、長整數型、4 長整數型所有的暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數 *

* 無法指定位元型的常數。

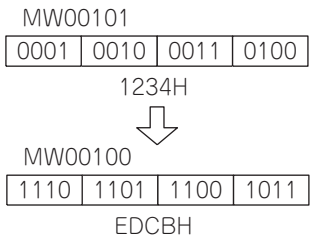
程式範例

反轉 (!) 指令適用於運動程式、序列程式以及階梯圖程式。
以下為反轉 (!) 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	MB001000 = !MB001010;	
W	MW00100 = !MW00101;	
L	ML00100 = !ML00102	
F	–	–

範例　**反轉 (!) 指令的程式範例**

MW00100 = !MW00101;



6.7 數值比較

接下來將說明用來判別條件式的數值比較指令。
數值比較指令包含 6 種指令，運動程式或序列程式皆可適用。

下表為數值比較指令一覽表。

指令	名稱	格式	內容	運動程式	序列程式
= =	等於	IF MB□ = = MB□; WHILE MB□ = = MB□; IF MW□ = = MW□; WHILE MW□ = = MW□; IF ML□ = = ML□; WHILE ML□ = = ML□; IF MF□ = = MF□; WHILE MF□ = = MF□; IF MQ□ = = MQ□; WHILE MQ□ = = MQ□; IF MD□ = = MD□; WHILE MD□ = = MD□;	適用於 IF 或 WHILE 條件式。若式子的左邊和右邊相等時，即為「真」。	○	○
<>	不等於	IF MW□ <> MW□; WHILE MW□ <> MW□; IF ML□ <> ML□; WHILE ML□ <> ML□; IF MF□ <> MF□; WHILE MF□ <> MF□; IF MQ□ <> MQ□; WHILE MQ□ <> MQ□; IF MD□ <> MD□; WHILE MD□ <> MD□;	適用於 IF 或 WHILE 條件式。若式子左邊和右邊不相等時，即為「真」。	○	○
>	大於	IF MW□ > MW□; WHILE MW□ > MW□; IF ML□ > ML□; WHILE ML□ > ML□; IF MF□ > MF□; WHILE MF□ > MF□; IF MQ□ > MQ□; WHILE MQ□ > MQ□; IF MD□ > MD□; WHILE MD□ > MD□;	適用於 IF 或 WHILE 條件式。若式子左邊大於右邊時，即為「真」。	○	○

(續下頁)

(接上頁)

指令	名稱	格式	內容	運動程式	序列程式
<	小於	IF MW□ < MW□; WHILE MW□ < MW□; IF ML□ < ML□; WHILE ML□ < ML□; IF MF□ < MF□; WHILE MF□ < MF□; IF MQ□ < MQ□; WHILE MQ□ < MQ□; IF MD□ < MD□; WHILE MD□ < MD□;	適用於 IF 或 WHILE 條件式。若式子左邊小於右邊時，即為「真」。	○	○
>=	大於等於	IF MW□ >= MW□; WHILE MW□ >= MW□; IF ML□ >= ML□; WHILE ML□ >= ML□; IF MF□ >= MF□; WHILE MF□ >= MF□; IF MQ□ >= MQ□; WHILE MQ□ >= MQ□; IF MD□ >= MD□; WHILE MD□ >= MD□;	適用於 IF 或 WHILE 條件式。若式子左邊大於等於右邊時，即為「真」。	○	○
<=	小於等於	IF MW□ <= MW□; WHILE MW□ <= MW□; IF ML□ <= ML□; WHILE ML□ <= ML□; IF MF□ <= MF□; WHILE MF□ <= MF□; IF MQ□ <= MQ□; WHILE MQ□ <= MQ□; IF MD□ <= MD□; WHILE MD□ <= MD□;	適用於 IF 或 WHILE 條件式。若式子左邊小於等於右邊時，即為「真」。	○	○

(註) 在格式的□中加入暫存器編號。

數值比較 (= =，<>，>，<，>=，<=)

數值比較 (==、<>、>、<、>=、<=) 指令可用來判別分歧 (IF ELSE IEND)、循環 (WHILE WEND)、包含 1 次掃描 WAIT 的循環 (WHILE WENDX) 或是等待輸出入 (IOW) 等「條件式」。比較指令包含以下 6 種指令。

比較指令	內容
= =	等於
< >	不等於
>	大於
<	小於
> =	大於等於
< =	小於等於

格式

以下為數值比較 (==、<>、>、<、>=、<=) 指令的格式。

IF MB001000 = = 1;
①

內容	使用用途	適用之暫存器
①	條件式	・位元型 *、整數型、長整數型、4 長整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器

* 位元型條件式只能使用等於 (==)。

程式範例

數值比較指令適用於運動程式、序列程式以及階梯圖程式。
以下為數值比較指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	IF MB001000 = = 1;	
W	IF MW00100<>10;	
L	IF ML00100>10000;	
F	IF MF00100>= 3.0;	

接下來要介紹數值比較指令所適用的條件式。

◆ 比較位元型資料

■ 格式

數值比較指令以 ==(等於) 來表示。

式子左邊編寫暫存器，右邊則寫 0 或 1。

```
IF MB000000 = = 0;      "MB000000 = 0
IF MB000000 = = 1;      "MB000000 = 1
```

■ 條件式運算

編寫 &、|、! (邏輯積、邏輯和、反轉) 邏輯演算。

```
IF (MB000000 &  MB000001) = = 1;    "MB000000 = 1 且MB000001 = 1
IF (MB000000 & !MB000001) = = 1;    "MB000000 = 1 且MB000001 = 0
IF (MB000000 ｜ MB000001) = = 1;    "MB000000 = 1 或是MB000001 = 1
IF (MB000000 ｜ !MB000001) = = 1;   "MB000000 = 1 或是MB000001 = 0
```

■ 語法錯誤範例

發生下述情況時，將出現語法錯誤。

・在數值比較指令中編寫 <>(不等於)

```
IF MB000000 <> 0;                   ⇒ 語法錯誤
```

・於式子左邊編寫數值，或於右邊編寫暫存器時

```
IF 1 = = MB000000;                  ⇒ 語法錯誤
IF MB000000 = = MB000001;           ⇒ 語法錯誤
```

・未加上數值比較指令時

```
IF MB000000;                        ⇒ 語法錯誤
IF (0);                             ⇒ 語法錯誤
```

・編寫多個數值比較指令時

```
IF (MB000000 = = 0) & (MB000001 = = 1);     ⇒ 語法錯誤
```

◆ 比較整數 / 長整數 / 實數型資料

■ 格式

可使用所有的數值比較指令 (==、<>、>、<、>=、<=) 等。

暫存器可編寫在式子左邊或右邊。

```
IF MW00000 = = 3;           "MW00000 ＝ 3
IF ML00000 <> ML00002;      "ML00000 ≠ ML00002
IF 1.23456 >= MF00000;      "1.23456 ≧ MF00000
```

■ 條件式運算

可編寫數值運算或邏輯運算。

```
IF MW00000 = = (MW00001/3);              "MW00000 ＝ (MW00001÷3)
IF (ML00000 & F0000000H) <> ML00002;     "(ML00000 Λ F0000000H) ≠ ML00002
IF 1.23456 >= (MF00000 * MF00002);       "1.23456 ≧ (MF00000×MF00002)
```

■ 語法錯誤範例

發生下述情況時，將出現語法錯誤。

・式子兩邊皆只有編寫常數時

```
IF 0 = = 3;                              ⇒ 語法錯誤
IF (3.14 * 2 * 1000) > 9000.0;           ⇒ 語法錯誤
```

・未加上數值比較指令時

```
IF MW000000;                             ⇒ 語法錯誤
IF (-1);                                 ⇒ 語法錯誤
```

・編寫多個數值比較指令時

```
IF (MW00000 < 0) & (MW000001 > 0);       ⇒ 語法錯誤
```

6.8 資料操作

「資料操作」指令可用來複製或移動您所指定的暫存器資料。
資料操作指令包含 6 種指令，適用於運動程式及序列程式。

下表為資料操作指令一覽表。

指令	名稱	格式	內容	運動程式	序列程式
SFR	向右移動	SFR MB□ N□ W□;	依照所指定的數量，將位元變數向右移動。	○	○
SFL	向左移動	SFL MB□ N□ W□;	依照所指定的數量，將位元變數向左移動。	○	○
BLK	傳送區塊	BLK MW□ MW□ W□;	以您所指定的傳送來源端為起始，將您指定的區塊區域複製到指定的傳送目的端。	○	○
CLR	清除	CLR MW□ W□;	以您所指定的暫存器為起始，將您所指定大小的區域清除為 0。	○	○
SETW	資料表初始化	SETW MW□ DW□; W□;	所要傳送的資料將被儲存在傳送目的端暫存器以及符合所指定的傳送數的所有暫存器中。	○	○
ASCII	ASCII 轉換 1	ASCII ' 文字列 ' MW□;	您所指定的字串將被轉換為 ASCII 碼，並儲存在指定的儲存目的端中。	○	○

(註) 在格式的 □ 中加入暫存器編號。

位元右移 (SFR)

位元右移 (SFR) 就是根據起始位元編號和位元範圍所指定的位元列，僅將指定的位移數向右移動。

格式

以下為 SFR 指令的格式。

SFR MB001000 N5 W10 ;
① ② ③

內容	使用用途	適用之暫存器
①	起始位元	・所有位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	位移數	・所有整數型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數
③	位元範圍	

程式範例

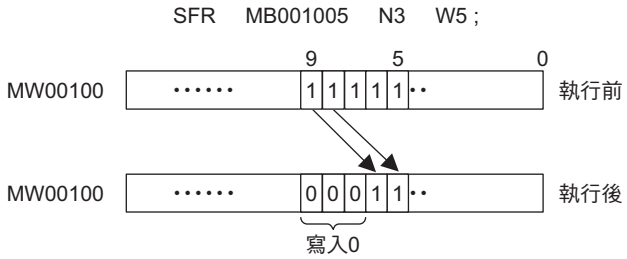
SFR 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 SFR 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	SFR MB001000 N5 W10;	
L	–	–
F	–	–

範例

SFR 指令的程式範例

以 MB001005 (MW00100 的位元 5) 為起始，將位元範圍 5 的資料向右移動 3 個位元。



補充

利用 SFR 指令讓位移數 ≥ 位元範圍時，您所指定的位元幅度資料將全部變為 0。

位元左移 (SFL)

位元左移 (SFL) 就是根據起始位元編號及位元範圍所指定的位元列，僅將指定的位移數向左移動。

格式

以下為 SFL 指令的格式。

SFL MB001000 N5 W10 ;
①　②　③

內容	使用用途	適用之暫存器
①	起始位元	・所有位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	位移數	・所有整數型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數
③	位元範圍	

程式範例

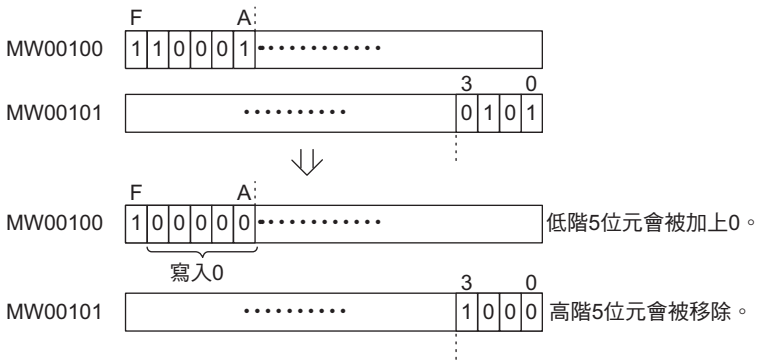
SFL 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 SFL 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	SFL MB001000 N5 W10;	
L	–	–
F	–	–

範例　**SFL 指令的程式範例**

以 MB00100A (MW00100 的位元 A) 為起始，將位元範圍 10 的資料向左移動 5 個位元。

SFL MB00100A N5 W10 ;



補充　利用 SFL 指令讓位移數 ≥ 位元範圍時，您所指定的位元範圍資料將全部變為 0。

傳送區塊 (BLK)

傳送區塊 (BLK) 就是從傳送來源端暫存器的起始位置開始，將指定數量的字元資料傳送到傳送目的端暫存器的起始位置。

格式

以下為 BLK 指令的格式。

BLK MW00100 DW00100 W10 ;
① ② ③

內容	使用用途	適用之暫存器
①	傳送來源端的起始暫存器	・所有整數型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	傳送目的端起始暫存器	・所有整數型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數
③	傳送區塊數	

程式範例

BLK 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 BLK 指令的各種程式範例。

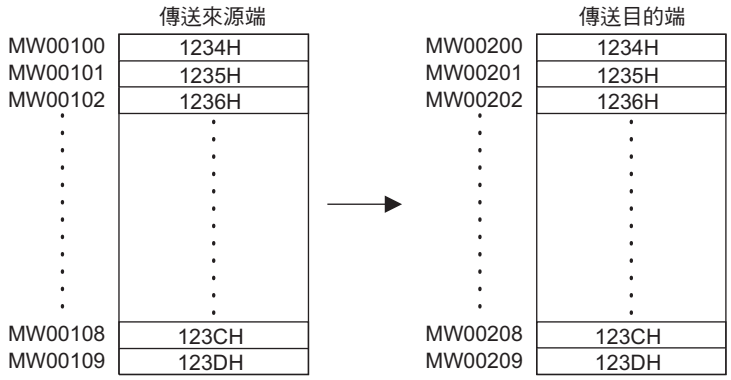
資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	BLK MW00100 DW00100 W10;	
L	–	–
F	–	–

範例

BLK 指令的程式範例

將 MW00100~MW00109 傳送到 MW00200~MW00209。

BLK MW00100 MW00200 W10;



補充

只要傳送來源端和目的端沒有重覆，傳送來源端資料就會直接被傳送到目的端。傳送來源端和目的端重覆時，傳送來源端資料可能無法直接被傳送到目的端。

清除 (CLR)

清除 (CLR) 指令可從清除資料起始暫存器開始，將指定的區塊數字元資料清除為 0。

格式

以下為 CLR 指令的格式。

CLR MW00100 W10 ;
① ②

內容	使用用途	適用之暫存器
①	清除資料起始位置	・所有整數型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	區塊數	・所有整數型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

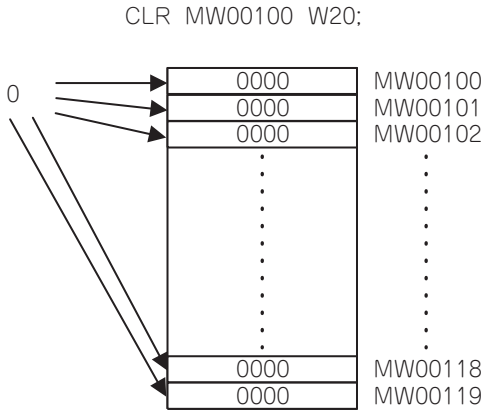
程式範例

CLR 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 CLR 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	CLR MW00100 W10;	
L	–	–
F	–	–

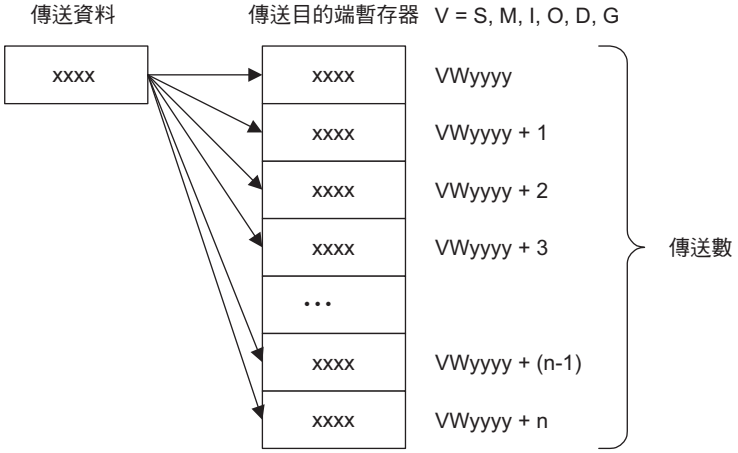
範例 **CLR 指令的程式範例**

將 MW00100~MW00119 的內容清除為 0。



資料表初始化 (SETW)

資料表初始化 (SETW) 是一種用來將傳送的資料儲存在傳送目的端暫存器和傳送數量所指定的所有暫存器中的指令。暫存器的編號往增加方向對每個字元進行儲存作業。



格式

以下為 SETW 指令的格式。

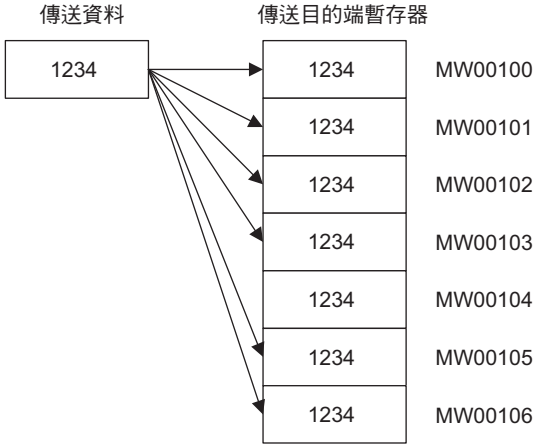
SETW MW00100 DW00100 W10;
① ② ③

內容	使用用途	適用之暫存器
①	傳送目的端暫存器	・所有整數型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	傳送資料	・所有整數型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數
③	傳送數	

程式範例

以下為 SETW 指令的程式範例。

```
DW00100 = 1234;
SETW MW00100 DW00100 W7;   " 利用 DW00100 的數值，將 MW00100~MW00106 進行初始化 "
END;
```



ASCII 轉換 1 (ASCII)

ASCII 轉換 1 (ASCII) 可將指令所指定的字元 (字串) 轉換為 ASCII 碼，然後再儲存到您所指定的儲存目的端暫存器 (整數型暫存器) 中。可區分大小寫。
依照第 1 個字元：第 1 個字元的低位元組、第 2 個字元 : 第 1 個字元的高位元組的順序進行儲存。若字串的長度為奇數時，則儲存目的端的最後一個字元的高位元組將變為 0。最多可輸入 32 個字。

格式

以下為 ASCII 指令的格式。

ASCII 'ABCDEFG' MW00200;
①　②

內容	使用用途	適用之暫存器
①	字串	ASCII 字元
②	儲存目的端暫存器編號	整數型暫存器 (#、C 暫存器除外)

接下來將介紹 ASCII 指令適用 / 不適用之字元。

◆ 適用之字元

下表為 ASCII 指令適用之字元。

項目	ASCII 字元
半形英文數字	a ～ z，A ～ Z，0 ～ 9
半形符號	空格字元， ! # $ % & ( ) * + , - . / : ; < = > ? @ [ ] ¥] ^ _ ` { | } ~

◆ 無法使用之字元

下表為 ASCII 指令不適用之字元。

項目	ASCII 字元
單引號	‘
雙引號	“
雙斜線	//
全形字	所有全形字
半形假名	所有半形假名

程式範例

以下為 ASCII 指令的程式範例。

◆ 將字串 "ABCD" 儲存到 MW00100~MW00101 時

ASCII 'ABCD' MW00100;

	高位元組	低位元組	
MW00100	42H('B')	41H('A')	MW00100 = 4241H
MW00101	44H('D')	43H('C')	MW00101 = 4443H

◆ 將字串 "ABCDEFG" 儲存到 MW00100~MW00103 時

ASCII 'ABCDEFG' MW00100;

	高位元組	低位元組	
MW00100	42H('B')	41H('A')	MW00100 = 4241H
MW00101	44H('D')	43H('C')	MW00101 = 4443H
MW00102	46H('F')	45H('E')	MW00102 = 4645H
MW00103	00H	47H('G')	MW00103 = 0047H

↑剩餘的位元組加上 0。

6.9 基本函數

「基本函數指令」是一種將數值運算和邏輯運算互相組合的特殊運算指令。
基本函數指令包含了 17 種指令。

下表為基本函數指令一覽表。

指令	名稱	格式	內容	運動程式	序列程式
SIN	正弦	SIN (MW□); SIN (90)；	求出正弦值。 實際規格將依指定為整數型或實數型而異。	○	○
COS	餘弦	COS (MW□); COS (90)；	求出餘弦值。 實際規格將依指定為整數型或實數型而異。	○	○
TAN	正切	TAN (MF□); TAN (45.0)；	求出正切值。 僅能指定為實數型。	○	○
ASN	反正弦	ASN (MF□); ASN (90.0)；	求出反正弦值。 僅能指定為實數型。	○	○
ACS	反餘弦	ACS (MF□); ACS (90.0);	求出反餘弦值。 僅能指定為實數型。	○	○
ATN	反正切	ATN (MW□); ATN (45)；	求出反正切值。 實際規格將依指定為整數型或實數型而異。	○	○
SQT	平方根	SQT (MW□); SQT (100)；	求出平方根值。 實際規格將依指定為整數型或實數型而異。	○	○
BIN	BCD → BIN	BIN (MW□);	將 BCD 資料轉換為 BIN 資料。	○	○
BCD	BIN → BCD	BCD (MW□);	將 BIN 資料轉換為 BCD 資料。	○	○
S{ }	指定位元 ON	S {MB□} = MB□ & MB□;	若邏輯運算結果為「真」，就會開啟指定的位元。 即使邏輯運算結果為「偽」，也不會關閉指定的位元。	○	○
R{ }	指定位元 OFF	R {MB□} = MB□ & MB□;	若邏輯運算結果為「真」，就會關閉指定的位元。即使邏輯運算結果為「偽」，也不會開啟指定的位元。	○	○
PON	上升脈衝	MB□ = PON (MB□ MB□); 或是 IF PON (MB□ MB□) = = 1; ・・・; IEND;	當位元輸入狀態由關閉變為開啟時，位元輸出訊號便會在 1 次掃描期間開啟。	×	○
NON	下降脈衝	MB□ = NON (MB□ MB□); 或是 IF NON (MB□ MB□) = = 1; ・・・; IEND;	當位元輸入狀態由開啟變為關閉時，位元輸出訊號便會在 1 次掃描期間開啟。	×	○

(續下頁)

(接上頁)

指令	名稱	格式	內容	運動程式	序列程式
TON	通電延遲計時器	MB□ = MB□ & TON (□ MB□);	當位元輸入開啟時，將開始計算時間。 當「計數值 = 設定值」時，位元輸出訊號將變為開啟。 計數時間係以 10 ms 為單位。	×	○
TON1MS	通電延遲計時器 (1 ms)	DB□ = DB□ & TON1MS (□ DB□);	當位元輸入開啟時，將開始計算時間。 當「計數值 = 設定值」時，位元輸出訊號將變為開啟。 計數時間係以 1 ms 為單位。	×	○
TOF	斷電延遲計時器	MB□ = MB□ & TOF (□ MB□);	當位元輸入關閉時，將開始計算時間。 當「計數值 = 設定值」時，位元輸出訊號將變為關閉。 計數時間係以 10 ms 為單位。	×	○
TOF1MS	斷電延遲計時器 (1 ms)	DB□ = DB□ & TOF1MS (□ DW□);	當位元輸入關閉時，將開始計算時間。 當「計數值 = 設定值」時，位元輸出訊號將變為關閉。 計數時間係以 1 ms 為單位。	×	○

(註) 在格式的 □ 中加入暫存器編號。

正弦 (SIN)

正弦 (SIN) 可將整數型或實數型資料的正弦值儲存作為運算結果。長整數型資料無法使用本指令。

格式

以下為 SIN 指令的格式。

MW00100 = **SIN (3000)**;
①　　②

內容	使用用途	單位	適用之暫存器
①	輸出正弦值	–	・整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入角度	度 (°)*	・整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

* 輸入單位和輸出結果依整數型、實數型而異。

・整數型時
適用範圍為 -327.68 ~ 327.67°。使用先前的運算結果 (整數型資料) 作為輸入資料，再將運算結果儲存至整數型暫存器中 (輸入單位 1 = 0.01°)。將輸出 x 10000 倍的數值作為運算結果。

・實數型時
輸入並使用先前的運算結果 (實數型資料)，然後再將正弦值儲存至實數型暫存器中 (單位 = 度)。

整數型資料		整數型資料
MW00102 = SIN (MW00100) ; (05000)　　(03000)	等價 ⇒　0.5=SIN30°	MF00102 = SIN (MF00100) ; (0.5)　　(30.0)


註記

若資料為整數型時，只要所輸入的數值超過 -327.68 ~ 327.67° 的範圍，將無法產生正確的結果。

程式範例

SIN 指令適用於運動程式、序列程式以及階梯圖程式。

以下為 SIN 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00102 = SIN(MW00100);	
L	–	–
F	DF00202 = SIN(DF00200);	

餘弦 (COS)

餘弦 (COS) 可將整數型或實數型資料的餘弦值儲存作為運算結果。
長整數型資料不適用本指令。

格式

以下為 COS 指令的格式。

MW00100 = COS (3000);
①　　②

內容	使用用途	單位	適用之暫存器
①	輸出餘弦值	–	・整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入角度	度 (°)*	・整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

* 輸入單位和輸出結果依整數型、實數型而異。
・整數型資料
適用範圍為 -327.68 ~ 327.67°。使用先前的運算結果 (整數型資料) 作為輸入資料，然後再將運算結果儲存至整數型暫存器中 (輸入單位 1 = 0.01°)。
將輸出 x 10000 倍的數值作為運算結果。
・實數型資料
輸入並使用先前的運算結果 (實數型資料)，再將餘弦值儲存至實數型暫存器中 (單位 = 度)。

整數型資料		實數型資料
MW00102 = COS (MW00100) ; (05000)　　(06000)	等價 ⇒ 0.5=COS60°	MF00102 = COS (MF00100) ; (0.5)　　(60.0)


註記

若資料為整數型時，只要所輸入的數值超過 -327.68 ~ 327.67° 的範圍，將無法產生正確的結果。

程式範例

COS 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 COS 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00102 = COS(MW00100);	
L	–	–
F	DF00202 = COS(DF00200);	

正切 (TAN)

正切 (TAN) 指令可將您所指定的變數或常數 (單位 = 度) 當作輸入資料，並將其正切值儲存在實數型暫存器中。

格式

以下為 TAN 指令的格式。

MW00100 = TAN (1.0);
①　　②

內容	使用用途	單位	適用之暫存器
①	輸出正切值	–	・實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入角度	度 (°)*	・實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

* (範例) 輸入值 (θ = 45.0°) 的正切值：計算 TAN (θ) = 1.0。

DF00102=TAN(DF00100);
（1.0）　　（45.0）

重要

TAN 指令僅適用實數型資料。若指定位元 / 整數 / 長整數時，編譯時將發生錯誤。

程式範例

TAN 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 TAN 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	–	–
L	–	–
F	DF00202 = TAN(DF00200);	

反正弦 (ASN)

反正弦 (ASN) 指令可將您所指定的變數或常數當作輸入資料，並將其反正弦值 (單位 = 度) 儲存在實數型暫存器中。

格式

以下為 ASN 指令的格式。

MF00100 = ASN (0.5);
①　　②

內容	使用用途	單位	適用之暫存器
①	輸出角度	度 (°)*	・實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入正弦值	–	・實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

* (範例) 輸入值 (0.5) 的反正弦值：計算 ASN (0.5) = 30.0°。

MF00202=ASN(MF00200);
(30.0)　　(0.5)

重要

ASN 指令僅適用實數型資料。若指定位元 / 整數 / 長整數時，編譯時將發生錯誤。

程式範例

ASN 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 ASN 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	–	–
L	–	–
F	DF00202 = ASN(DF00200);	

反餘弦 (ACS)

反餘弦 (ACS) 指令可將您所指定的變數或常數當作輸入資料，並將其反餘弦值 (單位 = 度) 儲存在實數型暫存器中。

格式

以下為 ACS 指令的格式。

MF00100 = ACS (0.5);
①　　②

內容	使用用途	單位	適用之暫存器
①	輸出角度	度 (°)*	・實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入餘弦值	–	・實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

* (範例) 輸入值 (0.5) 的反餘弦值：計算 ACS (0.5) = 60.0°。

MF00100 = ACS (MF00102) ;
（60.0）　　（0.5）

重要

ACS 指令僅適用實數型資料。若指定位元 / 整數 / 長整數時，編譯時將發生錯誤。

程式範例

ACS 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 ACS 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	–	–
L	–	–
F	DF00202 = ACS(DF00200);	

反正切 (ATN)

反正切 (ATN) 指令可將整數型或實數型資料的反正切值當作運算結果儲存起來。
長整數型資料不適用本指令。

格式

以下為 ATN 指令的格式。

MW00100 = ATN (100);
①　　②

內容	使用用途	單位	適用之暫存器
①	輸出角度	度 (°)*	・整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入正切值	–	・整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

* 輸入單位和輸出結果依整數型、實數型而異。

・整數型資料
適用範圍為 -327.68 ~ 327.67。將使用先前的運算結果 (整數型資料) 作為輸入資料，再將運算結果儲存至整數型暫存器中 (輸入 1 = 0.01、運算結果將輸出數值 100)。

・實數型資料
將使用先前的運算結果 (實數型資料) 作為輸入資料，再將反正切值儲存至實數型暫存器中。

整數型資料		實數型資料
MW00100 = ATN (MW00102) ; (04500)　　(00100)	等價 ⇒　45=ATN(1.0)	MF00100 = ATN (MF00102) ; (45.0)　　(1.0)

程式範例

ATN 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 ATN 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00102 = ATN(MW00100);	
L	–	–
F	DF00202 = ATN(DF00200);	

平方根 (SQT)

平方根 (SQT) 為將整數或實數資料的平方根作為演算結果。
長整數型資料不適用本指令。

格式

以下為 SQT 指令的格式。

MW00100 = SQT (100);
①　　②

內容	使用用途	適用之暫存器
①	輸出平方根	・整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	輸入資料	・整數型、實數型、倍精度實數型所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

(註) 輸入單位和輸出結果依整數型、實數型而異。

・整數型資料
不同於數學上的平方根，必須利用下述計算公式來進行運算。

$$\text{sign (輸入資料)} \times \sqrt{|\text{輸入資料}|} \times 32768$$

sign (輸入資料)：輸入資料的符號
｜輸入資料｜：輸入資料的絕對值

也就是將所謂數學的平方根結果乘以$\sqrt{32768}$ 後所得到之數值。此外，若輸入資料為負數時，運算絕對值的平方根，然後再將負數當作運算結果。運算的最大誤差值為 ± 2。

・實數型資料
將先前的運算結果 (實數型資料) 當作輸入資料，然後再將其平方根儲存為實數型資料。

	整數型資料	實數型資料
輸入值為正數時	MW00100 = SQT (MW00102) ; $\sqrt{64} \times \sqrt{32768} = 1448$ (01448)　(00064)　(8)　(181)	MF00100 = SQT (MF00102) ; (8.0)　(64.0)
輸入值為負數時	MW00100 = SQT (MW00102) ; $-\left(\sqrt{64} \times \sqrt{32768}\right) = -1448$ (-01448)　(-00064)　(8)　(181)	MF00100 = SQT (MF00102) ; (-8.0)　(-64.0)

程式範例

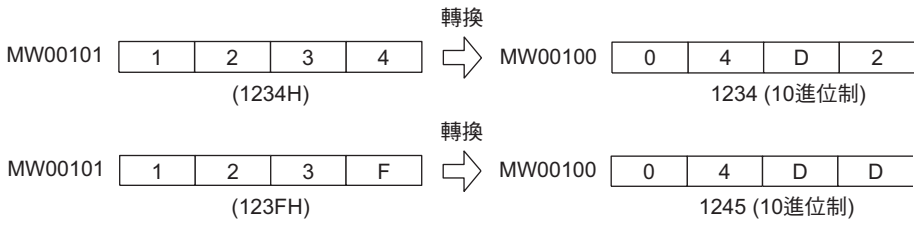
SQT 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 SQT 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00102 = SQT(MW00100);	
L	–	–
F	DF00202 = SQT(DF00200);	

BCD → BIN(BIN)

BCD → BIN (BIN) 為將 BCD 編碼的資料轉換為 2 進位制 (BIN 編碼)。
僅適用於整數資料。若您所下達的指令並非 BCD 編碼的數值，將無法產生正確的 2 進位制轉換 (BIN) 結果。

範例 **將 BCD 轉為 BIN 的轉換範例**



註記 若您所下達的指令為 BCD 碼以外資料，將無法得到正確的結果。

格式

以下為 BIN 指令的格式。

MW00100 = BIN (1234H);
① ②

內容	使用用途	適用之暫存器
①	BIN 輸出	・整數型、長整數型的所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
②	BCD 輸入	・整數型、長整數的所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

程式範例

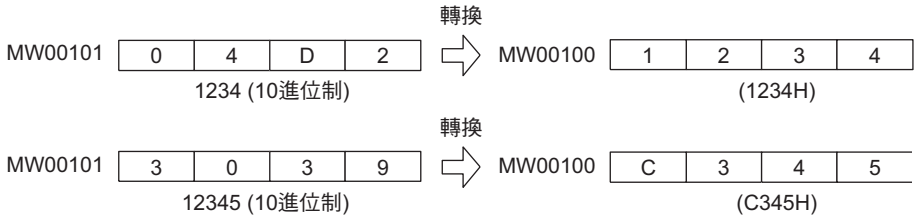
BIN 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 BIN 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00101 = BIN(MW00100);	
L	ML00102 = BIN(ML00100);	
F	–	–

BIN → BCD(BCD)

BIN → BCD (BCD) 為將 2 進位制 (BIN 編碼) 資料轉換為 BCD 編碼。
僅適用於整數資料。若 BIN 資料超過 270F，或是資料為負數值時，將無法產生正確的結果。

範例 **BIN → BCD 的轉換範例**



註記 若 BIN 資料大於 270F 時，將無法得到正確的結果。

格式

以下為 BCD 指令的格式。

MW00100 = BCD (1234);
A B

內容	使用用途	適用之暫存器
A	BCD 輸出	・整數型、長整數型的所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器
B	BIN 輸入	・整數型、長整數的所有暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・索引暫存器 ・常數

程式範例

BCD 指令適用於運動程式、序列程式以及階梯圖程式。
以下為 BCD 指令的各種程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	–	–
W	MW00101 = BCD(MW00100);	
L	ML00102 = BCD(ML00100);	
F	–	–

指定位元 ON(S{ })

指定位元 ON (S{ }) 就是當邏輯運算結果為 「真」時，將指定位元設定為 1 (開啟)。即使邏輯運算結果為「偽」，也不會將指定位元設定為 0 (關閉)。

格式

以下為指定位元 ON (S{ }) 指令的格式。

S { MB001000 } = MB001010 & MB001011;
①　　②

內容	使用用途	適用之暫存器
①	指定位元	・所有位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引
②	邏輯運算式	・所有位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・常數

程式範例

指定位元 ON (S{ }) 指令適用於運動程式、序列程式及階梯圖程式。
以下為指定位元 ON (S{ }) 指令各種不同的程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	S{MB001000} = MB001010&MB001011;	
W	–	–
L	–	–
F	–	–

指定位元 OFF(R{ })

指定位元 OFF (R{ }) 就是當邏輯運算結果為「真」時，將指定位元設定為 0 (關閉)。即使邏輯運算結果為「偽」，也不會將指定位元設定為 1 (開啟)。

格式

以下為指定位元 OFF (R{ }) 指令的格式。

R { MB001000 } = MB001010 & MB001011;
①　　②

內容	使用用途	適用之暫存器
①	指定位元	・所有位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引
②	邏輯運算式	・所有位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引 ・常數

程式範例

指定位元 OFF (R{ }) 指令適用於運動程式、序列程式及階梯圖程式。
以下為指定位元 OFF (R{ }) 指令各種不同的程式範例。

資料類型	運動程式 / 序列程式	階梯圖程式
B	R{MB001000} = MB001010&MB001011;	
W	–	–
L	–	–
F	–	–

上升脈衝 (PON)

上升脈衝 (PON) 為位元輸入狀態由 0 變為 1 時，位元輸出在 1 次掃描過程中開啟。位元輸出時用來儲存前次值的暫存器將被視為 PON 處理的工作。請設定未使用在其他處理的暫存器。

格式

以下為 PON 指令的格式。

DB000002 = PON (DB000000 DB000001) ;
①　　②　　③

內容	使用用途	適用之暫存器
①	位元輸出	・位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引
②	位元輸入	・所有位元型暫存器 ・同上述且附加索引
③	用來儲存位元輸出的前次值	・位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引

程式範例

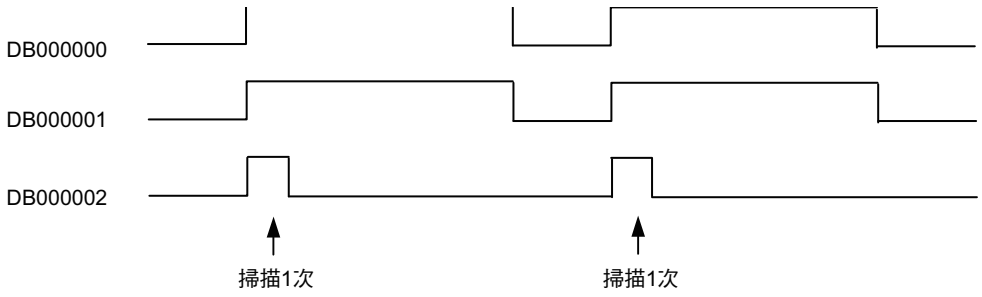
以下為 PON 指令的程式範例。

◆ 輸出至線圈時

```
DB000002 = PON(DB000000   DB000001);
```

・階梯圖程式的等價範例

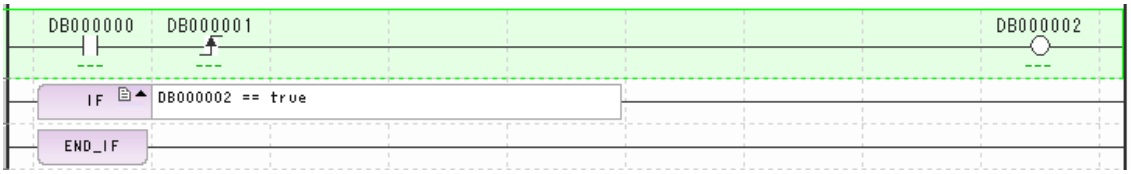
・時間圖



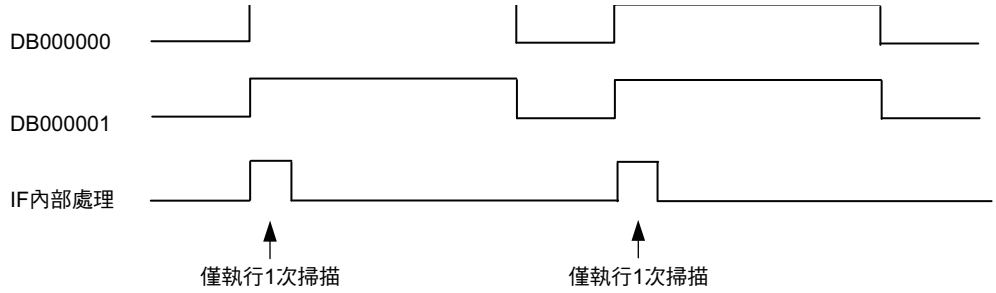
◆ 搭配 IF 指令使用時

```
IF PON(DB000000   DB000001) = = 1;
          .
          .
IEND;
```

· 階梯圖程式等價範例



· 時間圖



下降脈衝 (NON)

下降脈衝 (NON) 為位元輸入狀態由 1 變為 0 時，位元輸出在 1 次掃描過程中開啟。位元輸出時用來儲存前次值的暫存器將被視為 NON 處理的工作。
請設定未使用在其他處理的暫存器。

格式

以下為 NON 指令的格式。

DB000002 = NON (DB000000 DB000001) ;
①　　②　　③

內容	使用用途	適用之暫存器
①	位元輸出	・位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引
②	位元輸入	・所有位元型暫存器 ・同上述且附加索引
③	用來儲存位元輸出的前次值	・位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引

程式範例

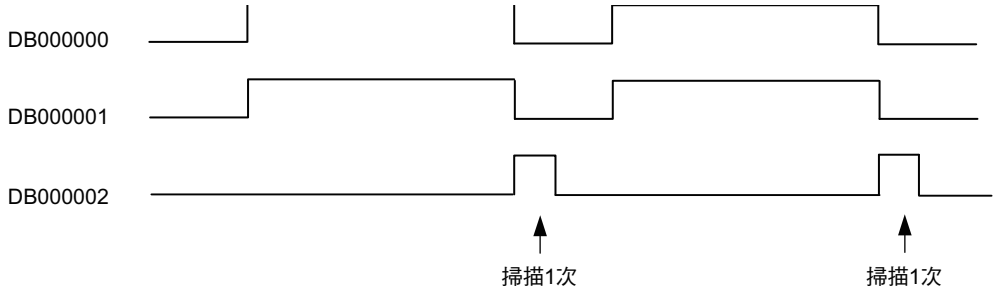
以下為 NON 指令的程式範例。

◆ 輸出至線圈時

DB000002 = NON (DB000000　DB000001);

・階梯圖程式等價範例

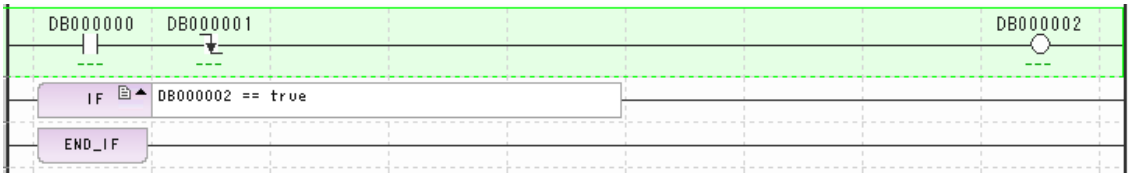
・時間圖



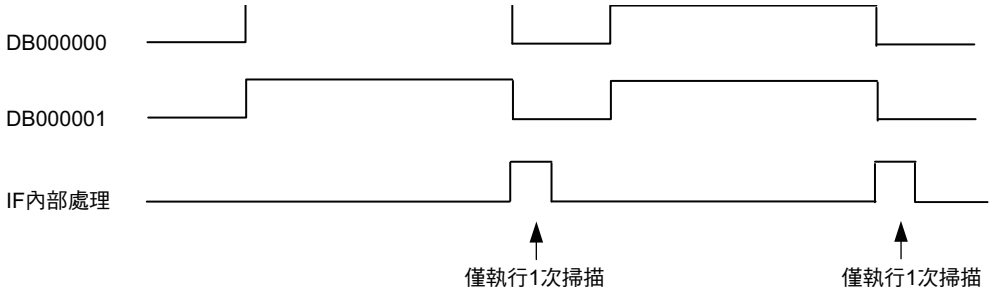
◆ 搭配 IF 指令使用時

```
IF NON(DB000000   DB000001) = = 1;
          .
          .
IEND;
```

・階梯圖程式等價範例



・時間圖



通電延遲計時器：測量單位 = 0.01 秒 (TON)

通電延遲計時器：測量單位 = 0.01 秒 (TON) 可在位元輸入 ON 時，進行時間計算。
當「計數值 = 設定值」時，位元輸出訊號將變為開啟。
若在計算過程中，位元輸入變為關閉，計時器就會停止動作。當位元輸入再次變為開啟時，就會從頭 (0) 開始進行計算。此外，實際的計數時間 (單位為 10 ms) 之值將會被儲存在計數專用暫存器中。

格式

以下為 TON 指令的格式。

DB000001 = DB000000 & TON (500 DW00001);
①　　②　　③　④

內容	使用用途	適用之暫存器
①	位元輸出	・位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引
②	位元輸入	・所有位元型暫存器 ・同上述且附加索引
③	設定值	・所有整數型暫存器 ・同上述且附加索引 ・常數 (0 ~ 65535 (655.35S)：單位為 10 ms)
④	計時器計數用暫存器	・所有整數型暫存器 ・同上述且附加索引

重要

1. 當除錯運轉停止時，將不會進行時間的計數。
   當停止除錯運轉被解除後，計數就會從目前的計數值開始計算。
2. 位元輸入必須指定為 "DB□□□□□□ &"。

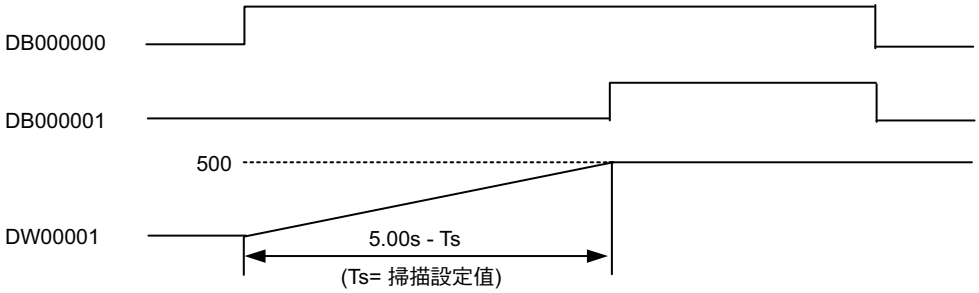
程式範例

以下為 TON 指令的程式範例。

DB000001 = DB000000 & TON (500　DW00001);
↑設定為 5 秒

・階梯圖程式等價範例

・時間圖



通電延遲計時器 (1 = 1 ms) (TON1MS)

通電延遲計時器：測量單位 = 0.001 秒 (TON1MS) 就是一項當位元輸入開啟時，進行時間測量，一旦「計數值 = 設定值」時將位元輸出開啟的指令。
若在計算過程中，位元輸入變為關閉，計時器就會停止動作。當位元輸入再次變為開啟時，就會從頭 (0) 開始進行計算。此外，實際的計數時間 (單位為 1 ms) 之值將會被儲存在計數專用暫存器中。

格式

以下為 TON1MS 指令的格式。

DB000001 = DB000000 & TON1MS(500 DW00001);
①　　②　　③　④

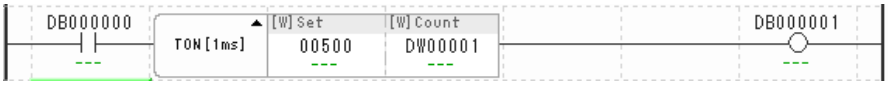
內容	使用用途	適用的資料
①	位元輸出	・位元型的暫存器 (#、C 暫存器除外) ・同上述且附加索引
②	位元輸入	・位元型的暫存器 (#、C 暫存器除外) ・同上述且附加索引
③	設定值	・所有整數型的暫存器 ・同上述且附加索引 ・常數 (0 ~ 65535 (65.535 s)：單位為 1 ms)
④	計時器計數用暫存器	・所有整數型暫存器 ・同上述且附加索引

程式範例

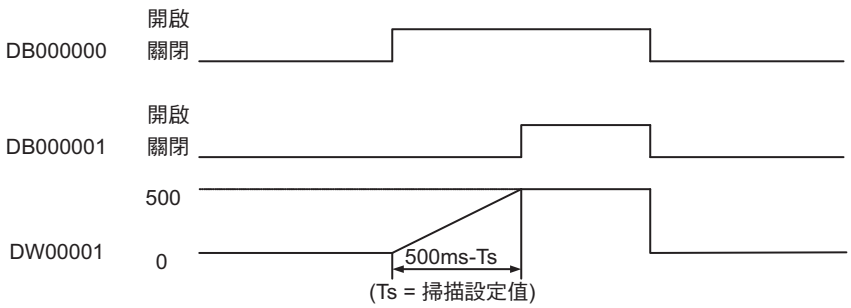
以下為使用 TON1MS 指令的序列程式及階梯圖程式範例。

```
DB000001 = DB000000 & TON1MS(500 DW00001);
```

・階梯圖程式等價範例



・時間圖



斷電延遲計時器：測量單位 = 0.01 秒 (TOF)

斷電延遲計時器：測量單位 = 0.01 秒 (TOF) 可在位元輸入關閉時，進行時間計算。
當「計數值 = 設定值」時，位元輸出訊號將變為關閉。
若在計算過程中，位元輸入變為開啟，計時器就會停止動作。當位元輸入再次變為關閉時，就會從頭 (0) 開始進行計算。此外，實際的計數時間 (單位為 10 ms) 之值將會被儲存在計數專用暫存器中。

格式

以下為 TOF 指令的格式。

DB000001 = DB000000 & TOF (500 DW00001);
①　　②　　③　④

內容	使用用途	適用之暫存器
①	位元輸出	・位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引
②	位元輸入	・所有位元型暫存器 ・同上述且附加索引
③	設定值	・所有整數型暫存器 ・同上述且附加索引 ・常數 (0 ~ 65535 (655.35S)：單位為 10 ms)
④	計時器計數用暫存器	・所有整數型暫存器 ・同上述且附加索引

重要

1. 當除錯運轉停止時，將不會進行時間的計數。
   當停止除錯運轉被解除後，就會從目前的計數值重新開始計算。
2. 位元輸入必須指定為 "DB□□□□□ &"。

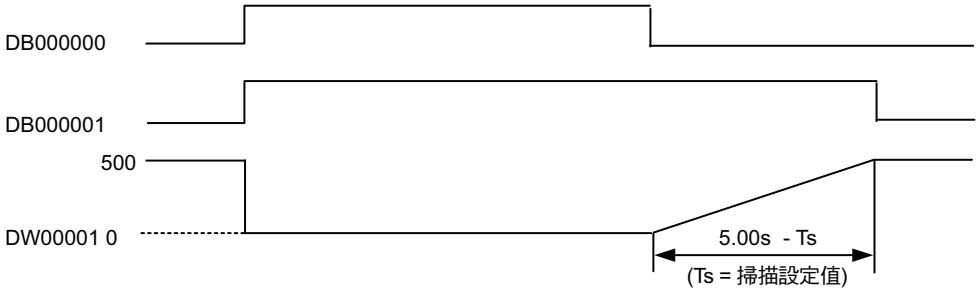
程式範例

以下為 TOF 指令的程式範例。

DB000001 = DB000000 & TOF (500　DW00001);

・階梯圖程式等價範例

・時間圖



斷電延遲計時器 (1 = 1 ms)(TOF1MS)

斷電延遲計時器：測量單位 = 0.001 秒 (TOF1MS) 就是一項當位元輸入關閉時，進行時間測量，一旦「計數值 = 設定值」時將位元輸出關閉的指令。
若在計算過程中，位元輸入變為開啟，計時器就會停止動作。當位元輸入再次變為關閉時，就會從頭 (0) 開始進行計算。此外，實際的計數時間 (單位為 1 ms) 之值將會被儲存在計數專用暫存器中。

格式

以下為 TOF1MS 指令的格式。

DB000001 = DB000000 & TOF1MS(500 DW00001);
①　　②　　③　④

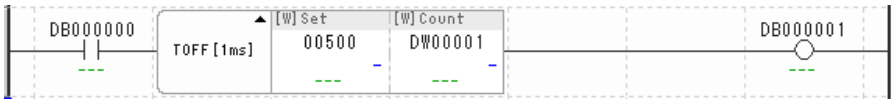
內容	使用用途	適用的資料
①	位元輸出	・位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引
②	位元輸入	・位元型暫存器 (#、C 暫存器除外) ・同上述且附加索引
③	設定值	・所有整數型暫存器 ・同上述且附加索引 ・常數 (0 ~ 65535 (65.535 s)：單位為 1 ms)
④	計時器計數用暫存器	・所有整數型暫存器 ・同上述且附加索引

程式範例

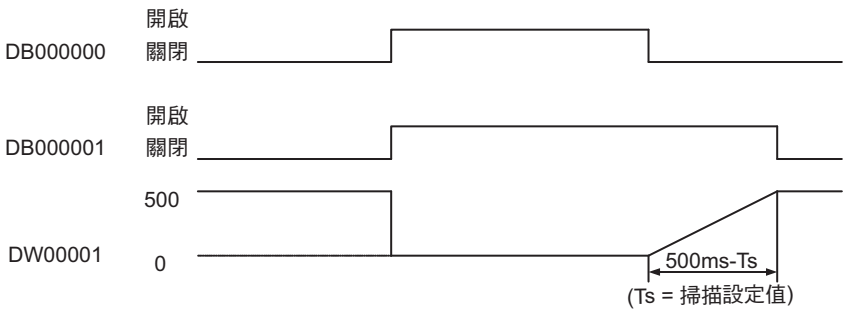
以下為使用 TOF1MS 指令之序列程式及階梯圖程式範例。

```
DB000001 = DB000000 & TOF1MS(500 DW00001);
```

・階梯圖程式等價範例



・時間圖



6.10 影像指令

所謂「影像指令」就是一項利用影像元件 (YVD-001) 的攝影機取得所拍攝到的影像或進行分析的指令

影像指令包含 5 種指令，僅適用於運動程式。

下表為影像指令一覽表。

指令	名稱	格式	內容	運動程式	序列程式
VCAPI	影像擷取	VCAPI [邏輯線路名稱/邏輯攝影機名稱] 影像記憶編號 [邏輯線路名稱 / 邏輯攝影機名稱] 影像記憶編號 …；	從攝影機擷取影像。	○	×
VCAPS	影像擷取與外部觸發訊號同步	VCAPS [邏輯線路名稱/邏輯攝影機名稱]影像記憶編號 [邏輯線路名稱/邏輯攝影機名稱] 影像記憶編號 …TW(FW) 解除訊號；	利用外部觸發訊號，從攝影機擷取影像。	○	×
VFIL	前段處理指令	VFIL [邏輯線路名稱]要求參數 [邏輯線路名稱] 要求參數 …；	進行影像分析的前段處理。	○	×
VANA	分析指令	VANA [邏輯線路名稱] 要求參數 回應參數 [邏輯線路名稱] 要求參數 回應參數 …；	進行影像分析處理。	○	×
VRES	取得分析結果指令	VRES [邏輯線路名稱] 要求參數 回應參數 [邏輯線路名稱] 要求參數 回應參數 …；	讀取影像分析處理結果。	○	×

開發工具 (MPE720) 功能介紹

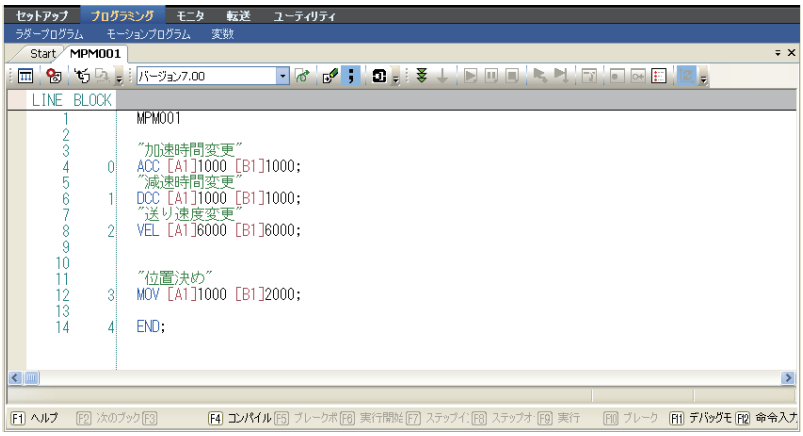
7

本章將說明利用開發工具 MPE720 的功能來編寫運動程式及序列程式的方法。

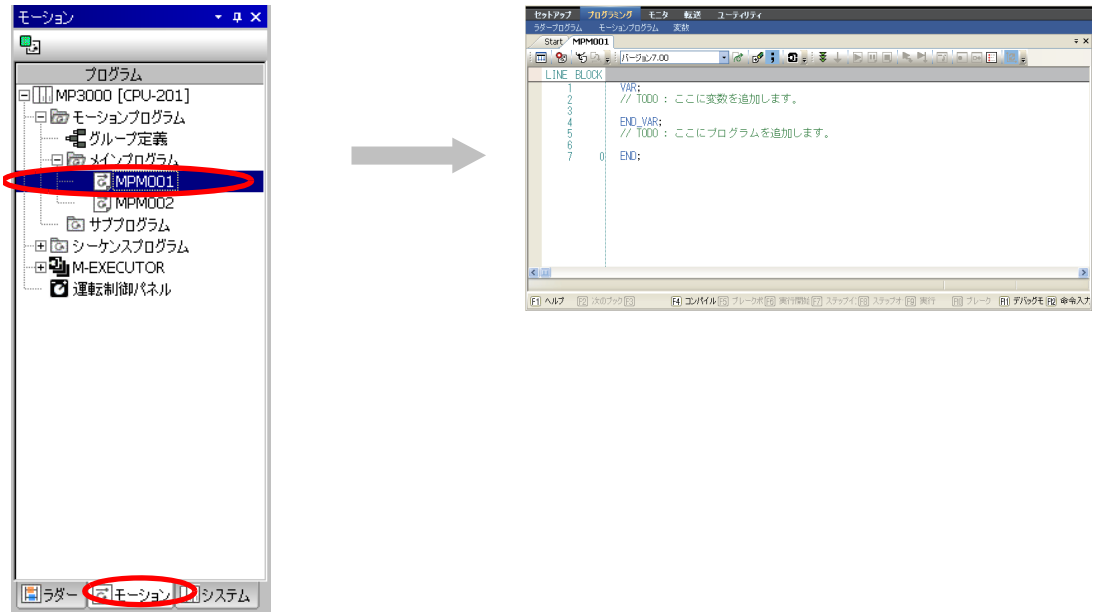
7.1 運動編輯器

「運動編輯器」係指利用運動程式和序列程式輸入或編輯時所必須使用到的程式編寫工具。具備了程式碼編輯、編譯 (儲存) 以及除錯、監控的功能。

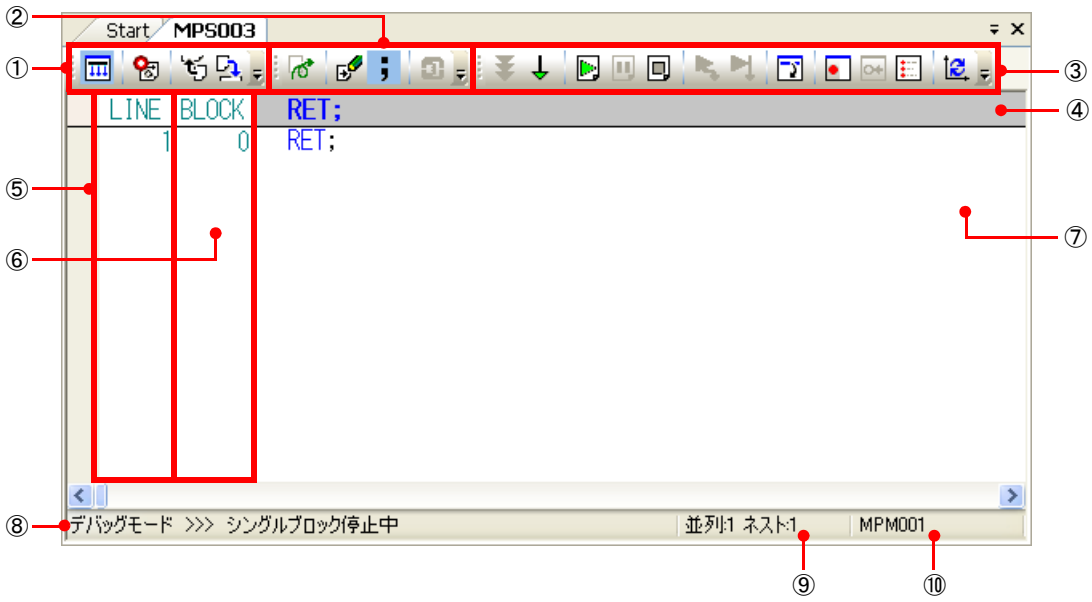
下圖為運動編輯器的視窗。



若要啟動運動編輯器，請進入 [運動] 子視窗，並選擇您所需要的程式。



■ 運動編輯器視窗說明



編號	名稱	內容	
①	監控工具列		跟著區塊監控畫面，隨時顯示在畫面上。
			可用來顯示運動警報視窗。
			可用來參照運動子程式。
			可用來顯示運動任務視窗。
②	程式工具列		可用來編譯已開啟的程式。
			可用來叫出 [指令輸入小幫手] 對話框。
			可自動加上分號。
			可用來叫出 [任務配置] 對話框。
③	運動除錯工具列		以除錯模式進行運轉。
			以一般運轉模式進行運轉。
			執行程式。
			中斷程式執行。
			強制結束目前正在執行的程式。
			STEP IN。
			STEP OVER。
			移動開始執行行。
			設定 / 解除斷點。
			斷點有效化 / 無效化。
			顯示斷點清單。
			更新至最新狀態。

(續下頁)

(接上頁)

編號	名稱	內容 (接上頁)
④	輸入指南	可在編寫運動程式的同時，一面確認運動語言指令的語法。將游標移到運動語言指令 (藍字) 上，即可叫出指令的輸入方法。
⑤	行編號	亦即所編寫的字串 (指令、註解、空白行等) 行數。
⑥	區塊編號	程式僅計算被執行的行，並顯示的行數。(註解、空白行等將不被納入計算。)
⑦	編輯 (輸入) 區	此區域可用來輸入程式。
⑧	狀態列	可用來顯示目前的運轉模式及警報發生狀況。 **一般運轉模式** 執行中 ：執行中 發生警報 ：目前發生警報 **除錯模式** 目前正在執行除錯模式：除錯模式 單一區塊停止中 ：除錯模式 >>> 單一區塊停止中 發生警報 ：除錯模式 >>> 目前發生警報
⑨	並列 / 巢狀式	可用來顯示並列編號、巢狀式編號。
⑩	主程式	可用來顯示目前參照的主程式編號。
⑪	編譯器版本	可用來指定編譯時的選項。 **版本 7.00** 支援 MP3000 運動程式所有功能。 利用 MPE720 Ver. 7 製作的新程式，為「7.00 版」。 **版本 6 互換** 僅支援 MP2000 運動程式的功能。 利用 MPE720 Ver 6 或 Ver5 所編寫的程式，為「版本 6 互換」。

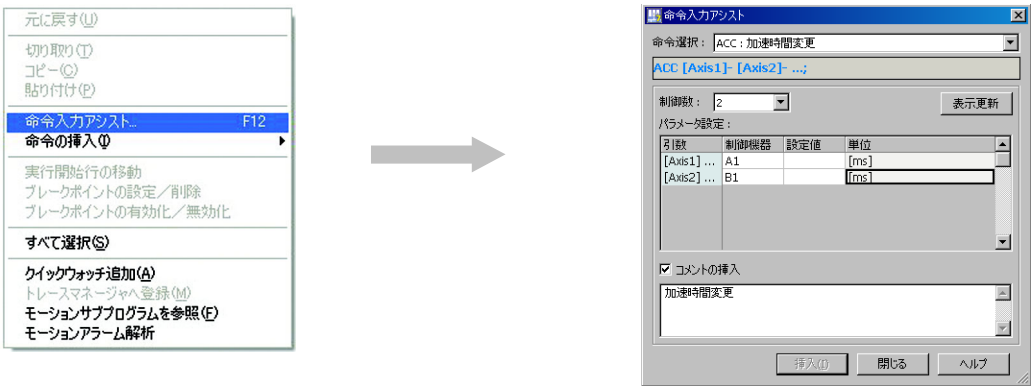
7.2 指令輸入小幫手

「指令輸入小幫手」是一種用來在編寫運動程式時協助運動語言指令輸入的功能。

運動語言指令就是利用文字形式的運動語言，必須將程式依照格式輸入。只要使用 [指令輸入小幫手] 對話框，即可輕鬆選擇您所要輸入的指令。

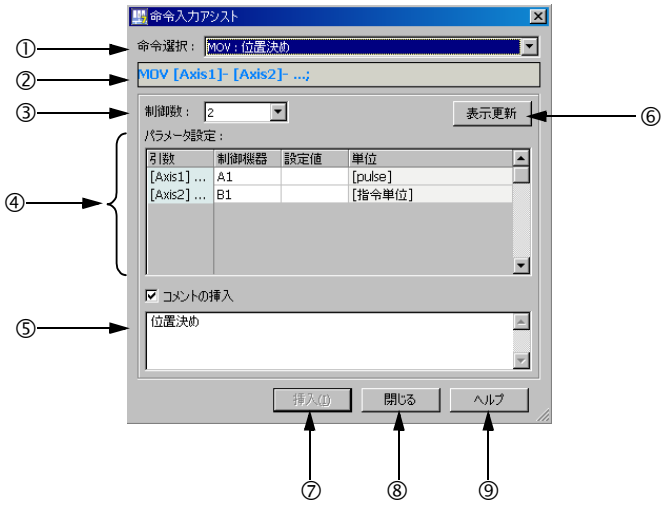
進入運動編輯器視窗，即可啟動指令輸入小幫手。啟動方法有二種。

· 從右鍵快速選單上選擇 [指令輸入小幫手]



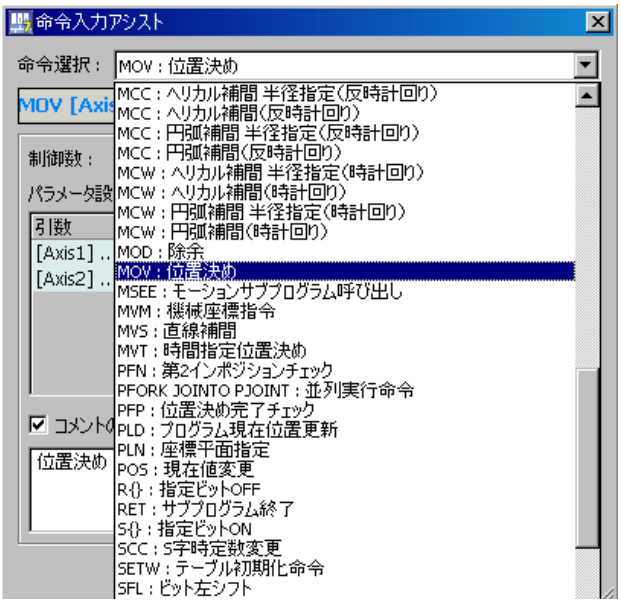
· 從右鍵快速選單上依序選擇 [插入指令] → (所要插入的指令)

■ [指令輸入小幫手] 對話框說明



① 選擇指令

下拉式選單將顯示可插入的指令清單。



② 指令輸入格式

可用來顯示您所選擇的指令輸入格式。

範例

MOV：定位

```
MOV [Axis1]- [Axis2]- ...;
```

+：加法

```
ML001 06=ML001 02 + ML001 04;
```

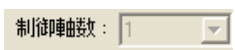
③ 控制軸數

使用軸移動類指令時，可選擇控制軸數 (1 軸 ~ 群組定義所設定的軸數)。
若控制軸數已固定，畫面上將顯示固定值並顯示為灰色 (無法使用)。

範例

MOV：定位 … 指定控制軸數

EXM：外部定位 … 控制軸數　固定



④ 設定參數

可用來設定指令的參數。設定項目如下表所示。

項目	內容
引數	顯示被設定為引數的參數名稱。無法變更。 若該引數可省略時，畫面上將顯示為 [可省略]。
邏輯軸名稱	顯示邏輯軸名稱。邏輯軸可依實際需要加以變更。
設定值	輸入作為設定值的常數或暫存器。
單位	畫面將顯示參數單位。無法變更。

利用群組定義即可定義邏輯軸名稱。

所顯示的單位將依每個軸的運動參數定義內容而定。不進行運動參數的定義時，顯示時單位就會變成黃色。將滑鼠對準該項目後，即可顯示說明圖說文字。請依照指示，為運動參數進行定義。

若該指令不需要設定控制軸數及參數，就會出現如下圖所示的 [指令輸入小幫手] 對話框的 [輸入程式] 方塊。請參照指令輸入格式並輸入指令。

⑤ 插入註解 / 註解欄

勾選 [插入註解] 的核取方塊後，即可將註解輸入到指令的上一行。若未勾選，則註解欄將顯示為灰色，此時無法插入註解。

補充　無法變更註解的插入位置。

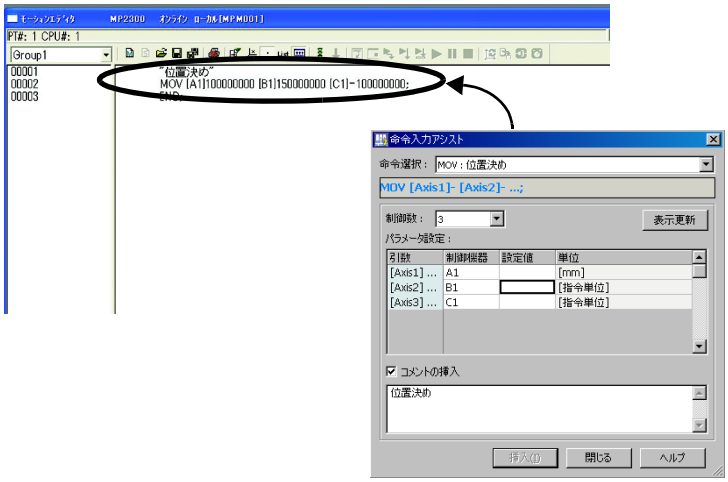
⑥ 更新畫面鍵

可用來更新 [指令輸入小幫手] 對話框的顯示內容。

補充　變更和單位有關的運動參數時，請利用 [更新畫面] 鍵，來更新目前顯示畫面。

⑦ 插入鍵

可將 [指令輸入小幫手] 對話框所編輯的指令插入運動編輯器的游標位置。

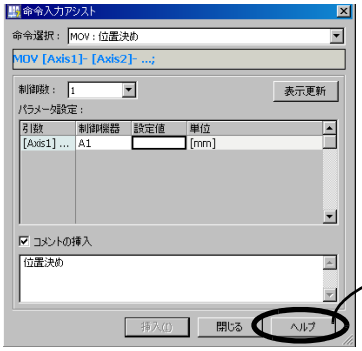


⑧ 關閉鍵

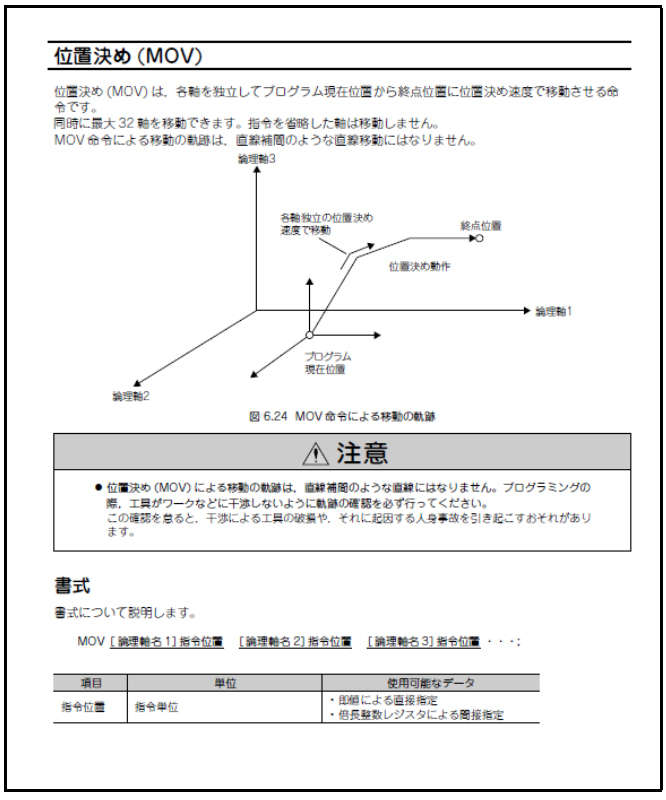
關閉 [指令輸入小幫手] 對話框。

⑨ 說明鍵

顯示相關指令的手冊。



可跳躍到您所選擇指令的說明。



位置決め (MOV)

位置決め (MOV) は，各軸を独立してプログラム現在位置から終点位置に位置決め速度で移動させる命令です。
同時に最大 32 軸を移動できます。指令を省略した軸は移動しません。
MOV 命令による移動の軌跡は，直線補間のような直線移動にはなりません。

図 6.24 MOV 命令による移動の軌跡

⚠ 注意

- 位置決め (MOV) による移動の軌跡は，直線補間のような直線にはなりません。プログラミングの際，工具がワークなどに干渉しないように軌跡の確認を必ず行ってください。
この確認を怠ると，干渉による工具の破損や，それに起因する人身事故を引き起こすおそれがあります。

書式

書式について説明します。

MOV [論理軸名 1] 指令位置 [論理軸名 2] 指令位置 [論理軸名 3] 指令位置 ・・・;

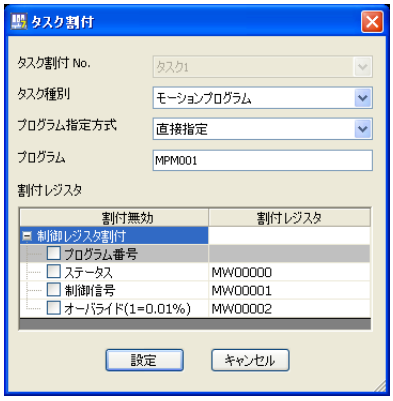
項目	単位	使用可能なデータ
指令位置	指令単位	・即値による直接指定 ・倍長整数レジスタによる間接指定

7.3 任務配置

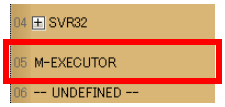
「任務配置」是一種用來叫出運動程式和序列程式的功能。

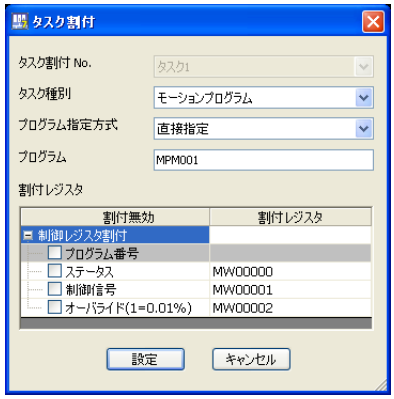
使用 [任務配置] 對話框，即可輕鬆地將編寫完成的運動程式和序列程式執行登錄到 MP3000 系統中。
啟動任務配置功能的方法有 2 種。

・點擊運動編輯器上的 [ ] 圖示的方法

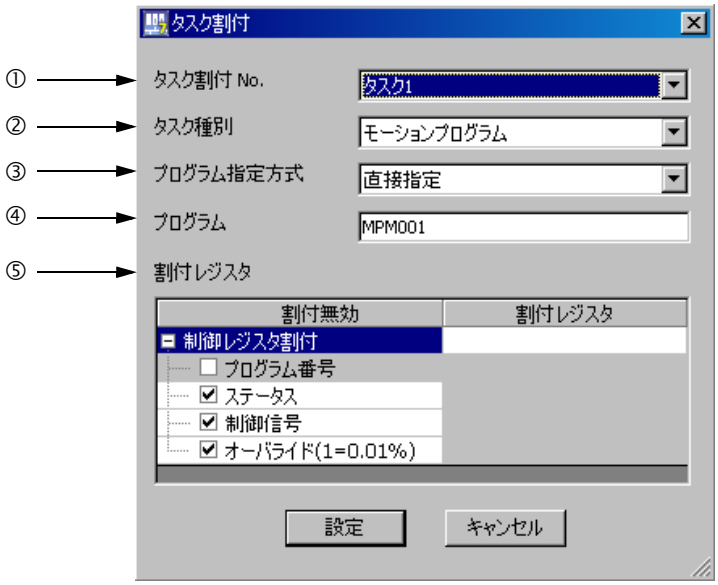


・在模組組成定義中開啟 **M-EXECUTOR** 的詳細定義，再點擊 [個別表示] 圖示的方法





■ [任務配置] 對話框說明



① 任務配置編號

畫面將會顯示用來配置程式時的任務編號。點擊運動編輯器工具列上的 [ ] 圖示，即可利用下拉式選單選擇任務編號。

② 任務類型

用來設定程式的執行類型。

執行類型	執行程式	執行條件
序列程式 (啟動)	序列程式	開啟電源 (僅在開啟電源時執行一次)
序列程式 (L 掃描)		定週期啟動 (每次進入低速掃描時序時執行)
序列程式 (H 掃描)		定週期啟動 (每次進入高速掃描時序時執行)
運動程式	運動程式	要求程式開始運轉的控制訊號 ON (要求程式開始運轉的訊號 ON 後執行)

③ 程式指定方式

設定程式的指定方法。
程式的指定方法依程式而異。

指定方法	運動程式	序列程式	備註
直接指定	可	可	指定程式編號的方法 範例：MPM001、SPM002 等
間接指定	可	不可	指定用來儲存程式編號之暫存器的方法 範例：OW0C0C 等 (將 1 儲存到 OW0C0C 後，就會開始執行 MPM001)

④ 程式

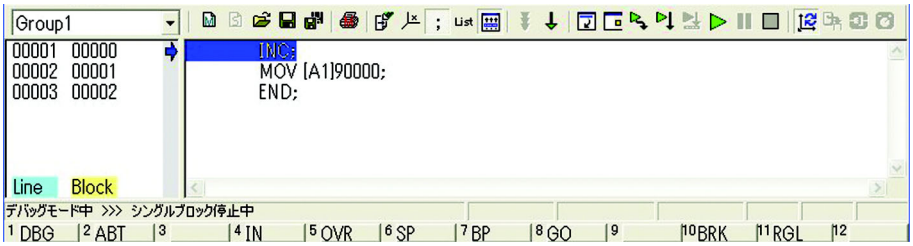
設定程式編號。

⑤ 配置用暫存器

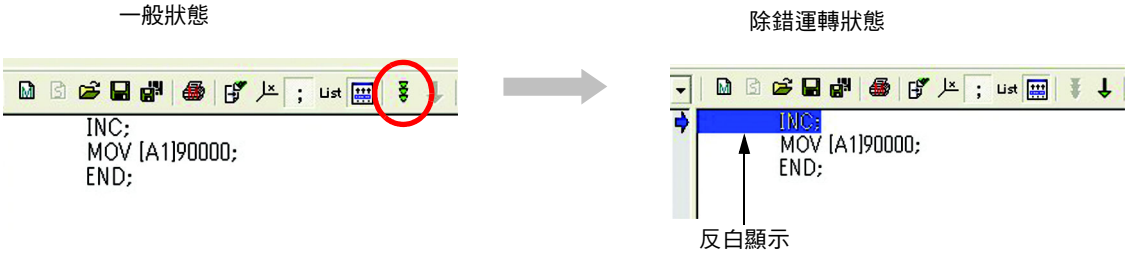
設定配置用暫存器。配置用暫存器和 M-EXECUTOR 控制用暫存器可即時進行資料交換。配置用暫存器可用來設定 I 暫存器、O 暫存器和 M 暫存器等任一種暫存器。

7.4 除錯運轉

除錯運轉係透過監控運動程式和序列程式執行中的行的方式，讓程式的錯誤更容易被發現的功能。
利用程式暫停、設定斷點、步進執行 (執行單一區塊) 等功能，即可對編寫完成的程式進行動作驗證。
進入除錯運轉狀態後，程式目前的執行行將如下圖所示，顯示在視窗上。



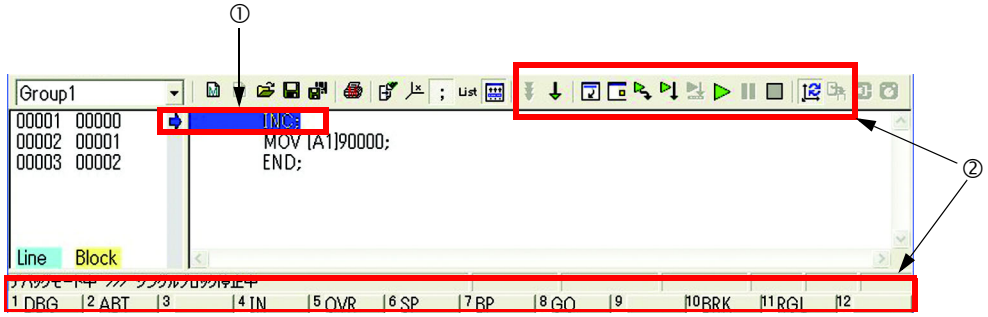
若要開始進行除錯運轉，請先連接控制器，然後點擊運動編輯器上的 [ ] 圖示。
如下圖所示，當程式進入除錯運轉狀態後，目前的執行行將會以反白方式顯示在視窗上。



註記

開始除錯運轉的前提條件為程式必須先執行登錄。

■ 除錯運轉狀態的視窗說明



① 程式執行行

目前程式的執行行將顯示為藍色。
一旦運動程式發生警報時，畫面將顯示為紅色。
如欲進一步瞭解運動程式的警報，請參閱以下手冊。

📖 **MP3000 系列　MP3200/MP3300　故障排除手冊 (資料編號：SIJP C880725 01)**

② 工具圖示和功能鍵

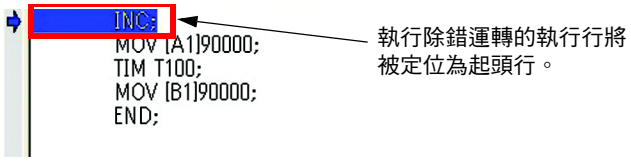
以下為除錯運轉時所使用的圖示和功能鍵。

功能	圖示	使用按鍵	說明	運動程式	序列程式
除錯模式		F1	開始進入除錯模式。	○	○
一般運轉模式		F11	結束除錯模式，並在一般運轉模式下繼續執行程式。	○	○
移動開始執行行		F6	移動開始執行行 (起始點)。	○	○
設定 / 刪除斷點		F7	設定或刪除斷點。程式畫面可用來顯示斷點。	○	○
STEP IN		F4	執行單一區塊。若使用 MSEE、SSEE 時，則會移動至子程式的起始行。	○	○
STEP OVER		F5	執行單一區塊。使用 MSEE、SSEE 時，會先執行子程式，然後再移動至 MSEE、SSEE 的下一個區塊。	○	○
執行		F8	在除錯模式下，繼續執行運動程式。	○	○
中斷		F10	在除錯模式下，暫停目前正在執行的運動程式。	○	○
強制結束		F2	強制停止運動程式執行。	○	×
更新目前位置		–	更新目前的位置座標。	○	×
設定運動任務		–	設定目前正在選擇的程式並列編號、階層編號及任務。	○	○
斷點的有效化 / 無效化	–	–	開啟或關閉斷點。可從除錯選單或使用右鍵快速選單來進行操作。	○	○
新增快速檢視	–	–	將暫存器登錄在快速參照中。或是使用右鍵快速選單來進行操作。	○	○

(註)○：可使用　×：不可使用

・**除錯模式**

「除錯模式」是一種可逐行確認程式並同時執行的一種模式。除錯運轉將從程式的起始行開始執行。



補充

切換至除錯模式時的除錯起始行，運動程式和序列程式將發生以下變化。

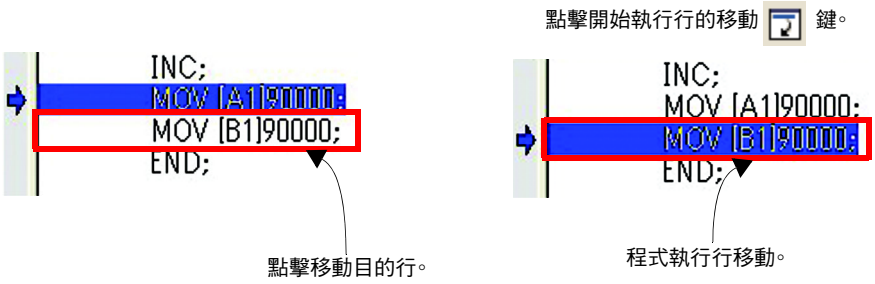
- 對於尚未啟動的運動程式操作時
  除錯運轉將從程式起始行開始執行，如上例所示。
- 要操作已經啟動的運動程式時
  在轉軸動作狀態下切換至除錯模式時，轉軸將在完成動作後，從下一個區塊位置開始進行除錯運轉。
- 對於尚未啟動的序列程式操作時
  無法執行除錯運轉。
- 要操作已經啟動的序列程式時
  除錯運轉將從程式起始行開始執行，如上例所示。

・**一般運轉模式**

「一般運轉模式」是一種程式從開始持續運轉到結束的模式。若在此時解除除錯運轉，將從程式目前的執行行開始重新運轉。所設定的斷點也會全部被解除。

・**移動開始執行行**

移動程式執行行。

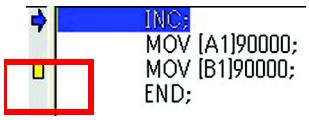


(註)不會執行「MOV[A1]90000；」行。

・**設定 / 刪除斷點**

設定斷點。斷點最多可設定 4 個。

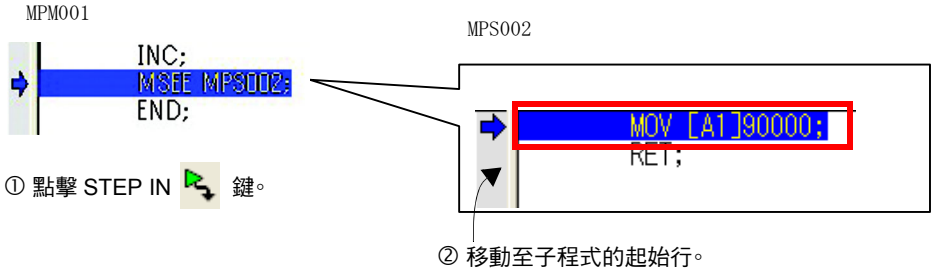
只要對已設定斷點的該行再度點擊一次，斷點便會被刪除。



・**STEP IN**

執行 1 行程式。

利用 MSEE 和 SSEE 來執行 STEP IN 時，就會移動至子程式的起始行。



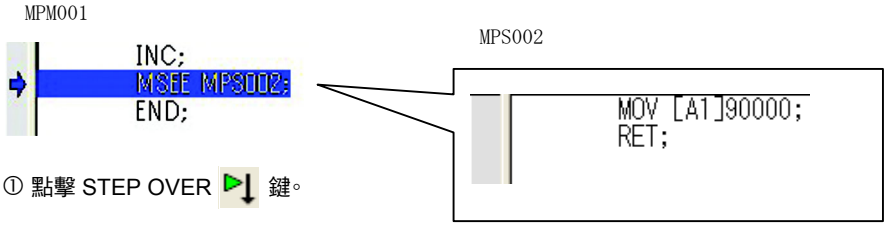
① 點擊 STEP IN 鍵。

② 移動至子程式的起始行。

・**STEP OVER**

執行 1 行程式。

利用 MSEE 和 SSEE 來執行 STEP OVER 時，將先執行子程式，然後再移動至下一行。

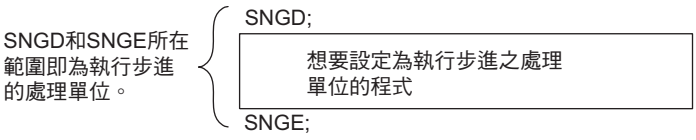


① 點擊 STEP OVER 鍵。

② 執行子程式。

③ 移動至 MSEE (SSEE) 的下一行。

補充　使用 SNGD/SNGE 指令，即可將多項處理作業視為程式執行步進的處理單位。



・**執行**

以連續方式執行程式。一旦到達斷點的該行時，就會在斷點行停止動作。

① 設定斷點。

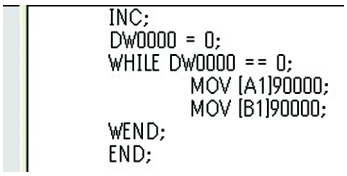
② 點擊執行 鍵。

③ 在斷點行停止。

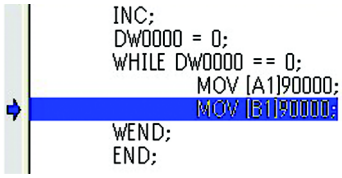
・**中斷**

暫停目前正在執行除錯運轉中的程式。要重新開啟程式時，請點擊 [ ] 圖示。

① 執行運動程式。



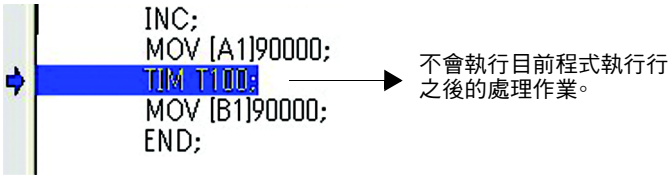
② 點擊中斷 鍵。



③ 程式停止。

・**強制結束**

強制結束目前正在進行除錯的程式執行作業。



・**更新目前位置**

此圖示和 PLD 指令具有同樣的功能。選擇此圖示時，即可用系統處理 STEP IN、STEP OVER、執行的操作下的 PLD 指令。

如欲進一步瞭解 PLD 指令，請參閱以下章節。

☞ **更新程式現在位置 (PLD)(第 6-113 頁)**

・**設定運動任務**  **(僅適用子程式)**

設定在子程式中進行程式監控、除錯運轉的子程式資訊。
顯示目前啟動中的主程式，並設定要用來叫出子程式的主程式。

・**設定呼叫堆疊**  **(僅適用子程式)**

設定比運動任務更詳細的子程式資訊。

① **主程式編號**

設定要用來叫出子程式的主程式。

② **FORK 編號**

設定要使用哪一項主程式並列來叫出子程式。

若要執行 MPS004 的程式監控或除錯運轉，請將 FORK 編號指定為 "3"。

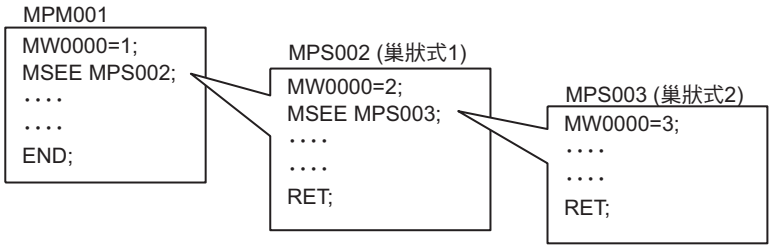
< 為 MPM001 時 >

```
PFORK Label1 Label2 Label3 Label4;
Label1: " 第 1 個並列
    MSEE MPS002;
    JOINTO LabelX;
Label2: " 第 2 個並列
    MSEE MPS003;
    JOINTO LabelX;
Label3: " 第 3 個並列
    MSEE MPS004;
    JOINTO LabelX;
Label4: " 第 4 個並列
    MSEE MPS005;
    JOINTO LabelX;
LabelX: PJOINT;
....
END;
```

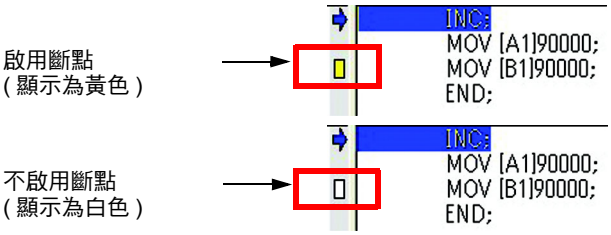
③ **NEST 階層**

設定子程式的叫出階層。

若要執行 MPS003 的程式監控或除錯運轉時，請將 NEST 階層指定為 "2"。

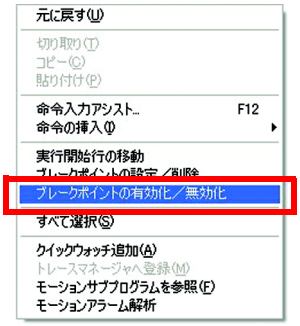


・**斷點的啟用 / 不啟用**
啟用或不啟用斷點。



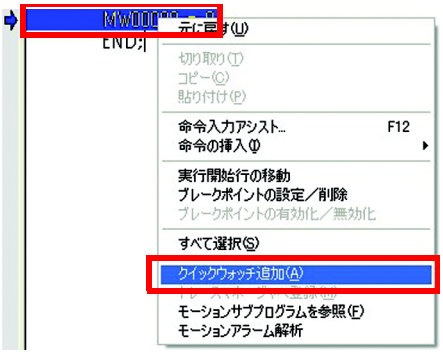
除錯 (G) 選單

運動編輯器視窗上的右鍵快速選單

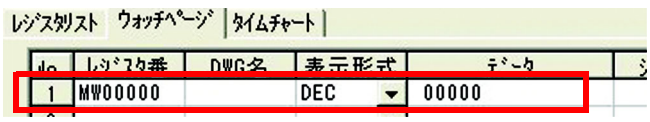


・**新增快速檢視**
可將編輯器上的暫存器登錄至快速參照中。
登錄至快速參照後，即可確認暫存器數值。

1. 在您所要監控的暫存器上叫出右鍵快速選單，接著選擇 **[新增快速檢視]**。



2. 暫存器會被登錄在快速參照的檢視頁面中。



7.5 運轉控制面板

所謂「運轉控制面板」就是一項可在程式執行進給行試運轉，或是用來監控執行中的程式運轉狀態的功能。

若要執行編寫完成的運動程式，請先執行登錄在 MP3000 系統中，然後再利用使用者程式，建立要求程式開始運轉訊號。
若想要能在編寫使用者程式前執行運動程式，可在 [運轉控制面板] 對話框中進行試運轉。
[運轉控制面板] 對話框可用來建立開始 / 停止運轉的要求、要求重置警報的指令等。

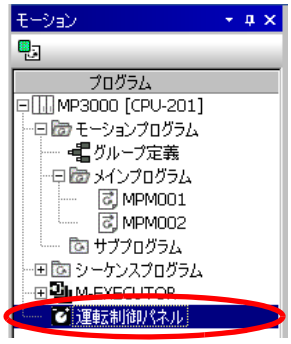
タスク	タスク1	タスク2	タスク3	タスク4
メインプログラム	MPM001	割付なし	割付なし	割付なし
制御信号	OW0C01 H0000	SW03323 H0000	SW03381 H0000	SW03439 H0000
Bit 0：運転スタート要求	○	○	○	○
Bit 1：一時停止要求	○	○	○	○
Bit 2：停止要求	○	○	○	○
Bit 3：シングルブロックモード	○	○	○	○
Bit 4：シングルブロックスタート要求	○	○	○	○
Bit 5：アラームリセット要求	○	○	○	○
Bit 6：継続運転スタート要求	○	○	○	○
Bit 8：スキップ1情報	○	○	○	○
Bit 9：スキップ2情報	○	○	○	○
Bit D：システムワーク番号設定	○	○	○	○
Bit E：補間用オーバーライド設定	○	○	○	○
ステータス	IW0C00 H0000	SW03322 H0000	SW03380 H0000	SW03438 H0000
Bit 0：運転中	○	○	○	○
Bit 1：一時停止中	○	○	○	○
Bit 2：停止中	○	○	○	○
Bit 4：シングルブロック運転停止中	○	○	○	○
Bit 8：アラーム発生中	○	○	○	○
Bit 9：ブレーク停止中	○	○	○	○
Bit B：デバッグモード中	○	○	○	○
Bit D：スタート要求信号ヒストリ	○	○	○	○
Bit E：システムワークなしエラー	○	○	○	○
Bit F：メインプログラム番号オーバーエラー	○	○	○	○

(註)運轉控制面板無法用來執行類似除錯運轉當中的設定斷點、
或是執行步進 (執行單一區塊) 的功能。

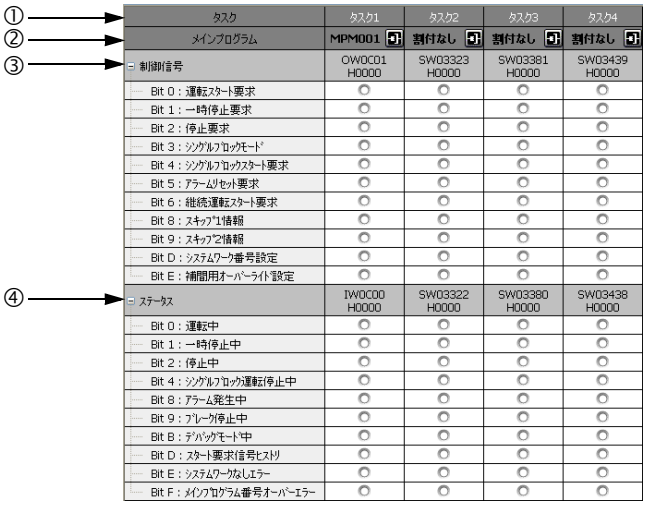

重要

1. 因可用來執行轉軸動作，因此必須仔細確認安全性之後再行使用。
2. 請不要將運動程式的控制暫存器覆寫在序列程式或階梯圖程式上。覆寫時，可能無法從本面板控制。
3. 對於同一轉軸請勿使用多個程式同時下達移動指令。否則將造成無法預期的意外動作發生。

若要啟動運轉控制面板，請點擊 [運動] 子視窗中的 [ ] 圖示。



■ 運轉控制面板視窗說明



① 任務

顯示任務編號

② 主程式

可用來顯示執行試運轉的主程式編號。

程式編號必須事先利用 M-EXECUTOR 程式執行定義來設定。

③ 控制訊號

顯示控制訊號的狀態內容。

④ 狀態

可用來顯示所執行的控制訊號狀態。

7.6 測試運轉功能

所謂「測試運轉功能」就是從畫面上為連接至控制器的轉軸進行試運轉的一項功能。

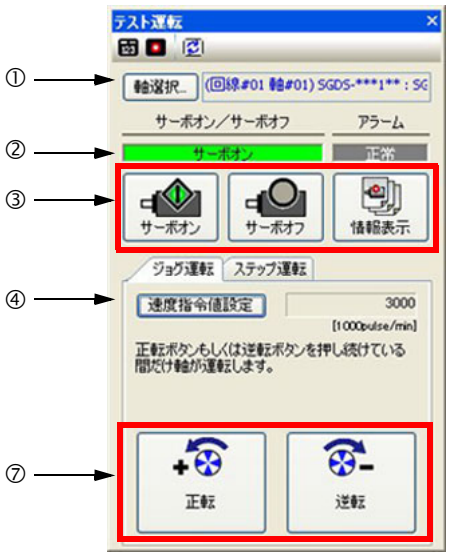
不需編寫程式即可下達執行伺服器開啟、伺服器關閉、JOG 動作或是步進動作指令。

重要

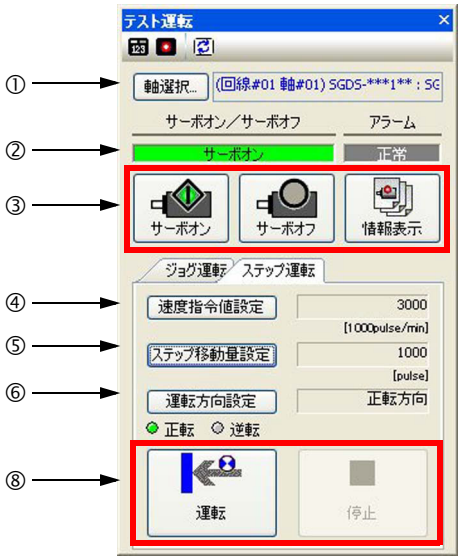
1. 因可用來執行轉軸動作，因此必須仔細確認安全性之後再行使用。
2. 運轉前，請採用即使處於一般運轉模式，仍能讓轉軸停止動作的設計方式。
3. 運轉前，請讓目前正在執行的階梯圖程式和運動程式停止。

若要啟動測試運轉功能，請進入 [系統] 子視窗，並雙擊 [測試運轉]。

■ [測試運轉] 對話框說明



利用 [JOG 運轉]、[步進運轉] 標籤進行切換



① 選擇轉軸

選擇測試運轉時所要執行動作的轉軸。

② 伺服器開啟、伺服器關閉、警報顯示

可用來顯示伺服器開啟 / 關閉狀態，以及轉軸警報狀態。

③ 伺服器開啟 / 關閉、資訊顯示

可將伺服器開啟或關閉。此設定可用來更新設定參數，使用時需特別注意。
點擊狀態顯示鍵後，即可顯示軸警報說明畫面。

④ 設定速度指令值

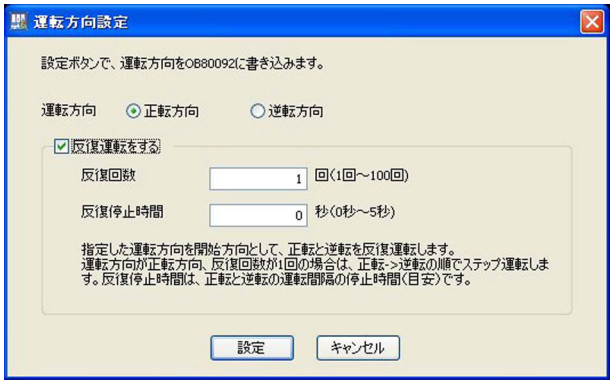
可用來設定速度指令值。此設定可用來更新設定參數，使用時需特別注意。

⑤ 設定步進移動量

可用來設定步進運轉模式下的步進移動量。此設定可用來更新設定參數，使用時需特別注意。

⑥ 設定運轉方向

畫面將顯示用來設定步進運轉模式下之軸運轉方向的 [設定運轉方向] 對話框。
進入 [設定運轉方向] 對話框，並勾選 [正轉方向] 或 [反轉方向] 其中一個核取方塊。此設定可用來更新設定參數，使用時需特別注意。
或是用來指定循環進行步進運轉。



⑦ JOG 運轉

執行 JOG 運轉。
點擊 [正轉] 鍵或 [逆轉] 鍵後，您所指定的轉軸就會開始執行動作。只要一解除點擊後，軸就會停止動作。

⑧ 步進運轉

執行步進動作。
點擊 [運轉] 鍵後，您所指定的轉軸就會執行一次步進動作。此動作不同於 JOG 運轉，無須連續點擊按鍵。
若指定為循環運轉時，依照您所指定的循環次數執行步進動作後，即停止動作。在循環運轉中，亦可用來停止轉軸動作。

7.7 軸運轉監控、軸警報監控

所謂「軸運轉監控」就是一項用來監控連接至運動控制器的轉軸運轉狀態的功能。

可顯示的運轉狀態包含狀態顯示 (準備完成 / 伺服器開啟、警報 / 警告、送出指令 / 定位完成、運動指令) 或是任一種監控參數。

「軸警報監控」是一種用來監控連接至運動控制器的轉軸警報資訊的功能。

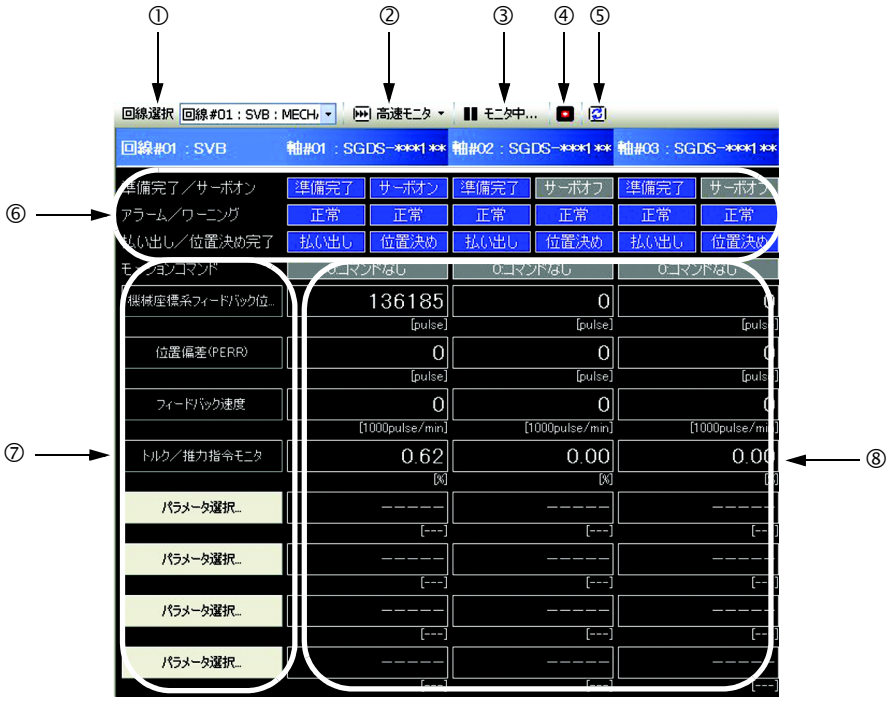
軸運轉監控

軸警報監控

若要啟動軸運轉監控、軸警報監控，請雙擊 [系統] 子視窗中的 [軸運轉監控] 或 [軸警報監控]。

■ 軸運轉監控視窗說明

接下來將說明軸運轉監控視窗畫面。

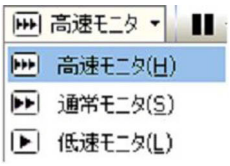


① 選擇線路

顯示您所選擇的線路的監控參數。

② 選擇監控週期

選擇監控週期。



③ 暫停 / 開始監控

暫停或開始監控動作。

④ 軸警報監控

顯示軸警報監控畫面。

⑤ 更新為最新資訊

將軸運轉監控畫面更新至最新狀態。

⑥ 狀態顯示

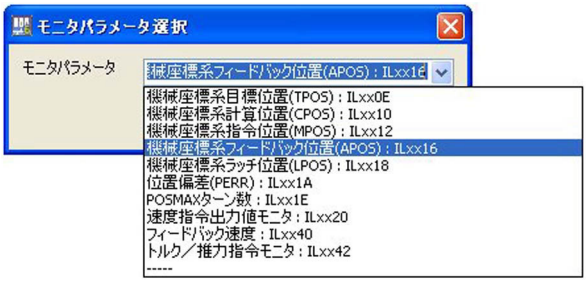
顯示準備完成 / 伺服器開啟、警報 / 警告、送出指令 / 定位完成及運動指令狀態。顯示畫面會依目前的狀態而變。

⑦ **選擇監控參數**

最多可選擇 8 個監控參數。

出廠時已預設為顯示機械座標系統反饋位置 (APOS)、位置偏差 (PERR)、反饋速度、轉矩 / 推力指令監控畫面。

點擊 [パラメータ選択...] 鍵，即可從以下的 [選擇監控參數] 對話框的下拉式選單中選擇監控參數。



顯示於下拉式選單中的監控參數

監控參數	暫存器	單位
機械座標系統目標位置 (TPOS)	IL□□□0E	指令單位
機械座標系統計算位置 (CPOS)	IL□□□10	指令單位
機械座標系統指令位置 (MPOS)	IL□□□12	指令單位
32 Bit 計算位置 (DPOS)	IL□□□14	指令單位
機械座標系統反饋位置 (APOS)	IL□□□16	指令單位
機械座標系統閂鎖位置 (LPOS)	IL□□□18	指令單位
位置偏差 (PERR)	IL□□□1A	指令單位
POSMAX 轉數	IL□□□1E	[rev]
監控速度指令輸出值	IL□□□20	[pulse/s]
反饋速度	IL□□□40	選擇速度單位
監控轉矩 / 推力指令	IL□□□42	轉矩單位選擇

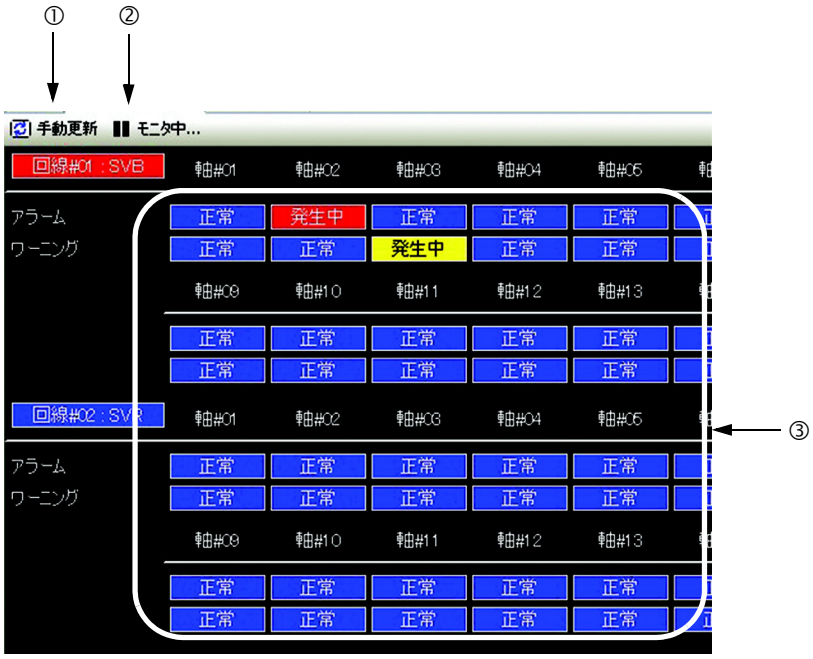
補充　若您在 [選擇監控參數] 對話框中輸入例如像 "IW08000" 的運動暫存器，即可用來指定未顯示在下拉式選單中的監控參數。

⑧ **監控參數畫面**

叫出您已指定的監控參數的狀態畫面。

■ 軸警報監控視窗說明

接下來將說明軸警報監控視窗顯示畫面。



① 手動更新

以手動方式更新警報及警告資訊。

② 暫停 / 開始監控

暫停或開始監控。

③ 警報及警告畫面

顯示警報及警告狀態畫面。

畫面	軸狀態
正常 (顯示為藍色)	未發生警報或警告
発生中 (顯示為紅色)	目前發生警報
発生中 (顯示為黃色)	警告發生中

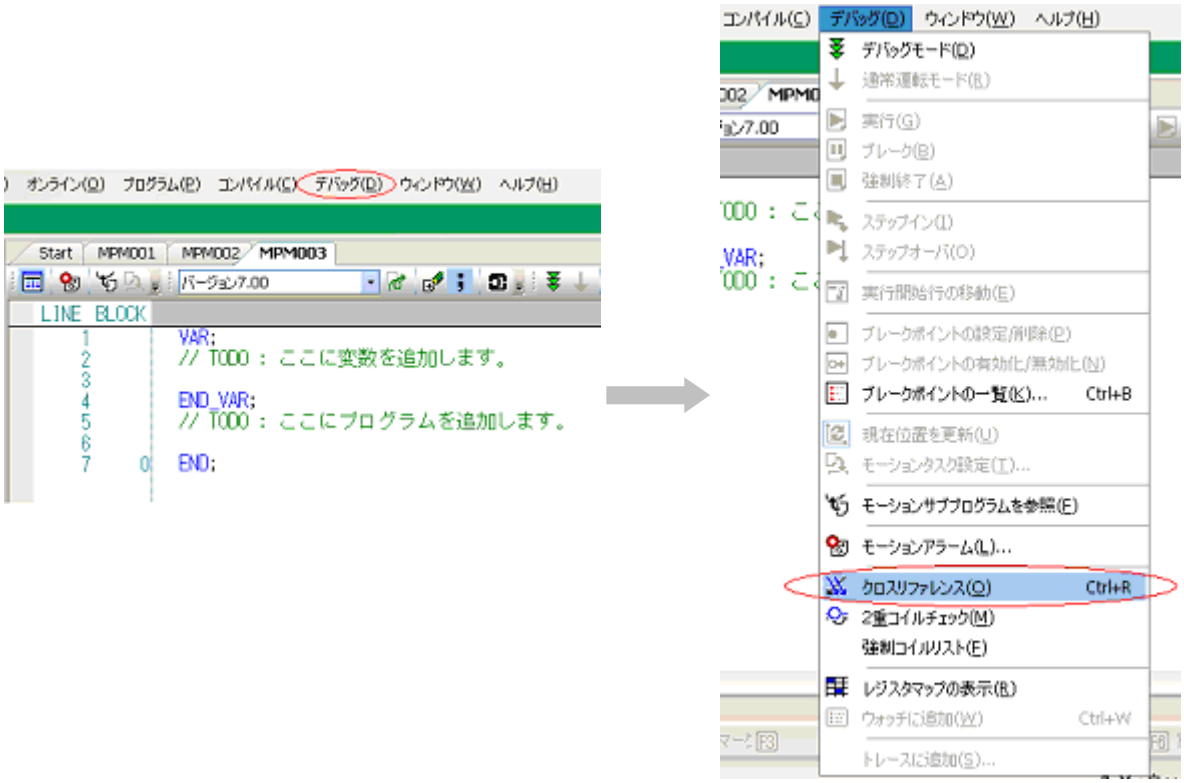
7.8 交互參照

接下來將說明交互參照。

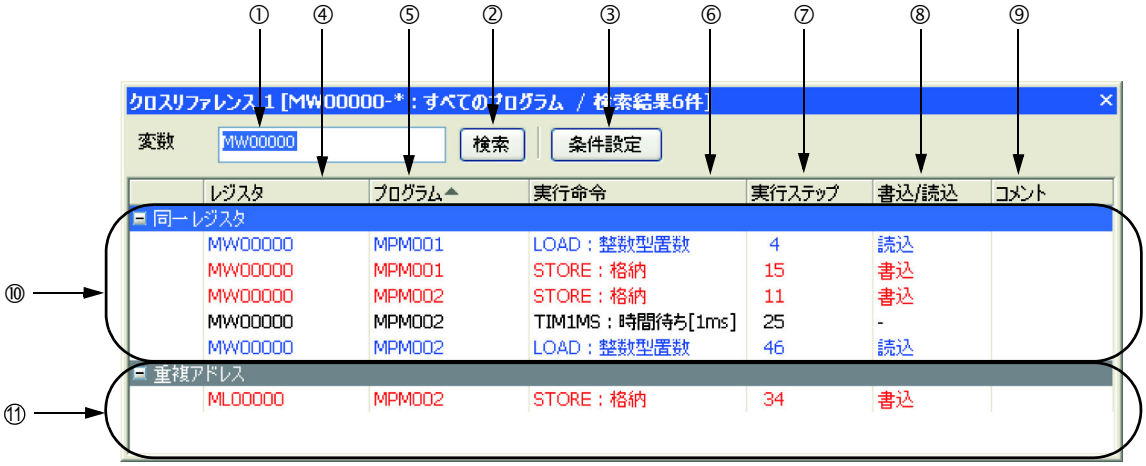
「交互參照」是一種用來檢索程式目前正在使用的變數或暫存器的功能。

執行檢索後，交互參照視窗中就會出現您檢索過的暫存器目前正在使用的程式編號和區塊編號。

若要啟動交互參照，請從主選單中依序選擇 [除錯 (D)] － [交互參照 (O)]。



■ 交互參照視窗說明



① 變數視窗

輸入您所要檢索的變數或暫存器。

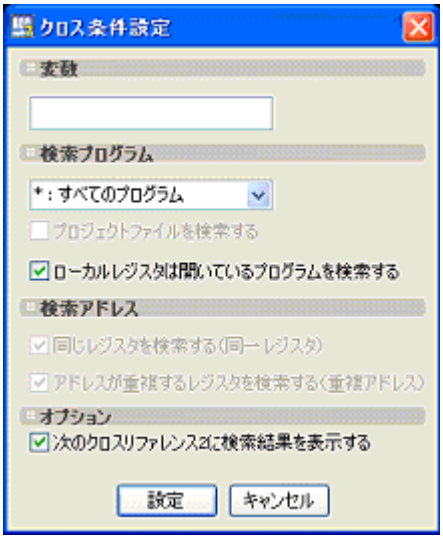
② 檢索鍵

可用來執行交互參照。

③ 條件設定鍵

可用來設定交互參照的條件。

點擊條件設定鍵後，畫面上就會出現 [設定交互條件] 對話框。



項目	內容
變數	輸入您所要檢索的變數、暫存器。
檢索程式	指定您所要檢索的程式。
檢索位址	指定檢索相同的暫存器、重複的位址。
選項	設定顯示檢索結果在下次的交互參照的選項。

④ 暫存器

畫面顯示您檢索過的變數或暫存器編號。

⑤ 程式

畫面顯示您檢索過的變數或目前使用暫存器編號的程式編號。

⑥ 執行指令

畫面顯示您檢索過的變數或目前使用暫存器編號的指令。

⑦ 有效步進

畫面顯示您檢索過的變數或目前使用暫存器編號的區塊編號。

⑧ 寫入 / 讀取

顯示您所檢索過的變數或暫存器編號目前正在執行寫入或讀取。

正在寫入時，畫面將顯示紅字。正在讀取時，則顯示為藍字。

⑨ 註解

顯示您檢索過的變數或暫存器編號的註解。

⑩ 同一個暫存器

畫面上將顯示和您所檢索過的變數或暫存器編號，其變數名稱、暫存器類型、資料類型、位址完全相同的暫存器。

⑪ 重複的位址

畫面上將顯示和您所檢索過的變數或暫存器編號，其位址重複的暫存器。

例如，檢索 MW00000 時，使用 ML00000 的地方就會被顯示在畫面上。

規格

附錄 A

本章將針對運動程式所支援的元件 / 模組及程式規格進行說明。

A.1 適用的元件 / 模組

以下的元件或模組適用於運動程式。

只要將轉軸連接至以下元件或模組，即可利用運動程式進行控制。

- MP3000/CPU-20□ SVC32
- MP3000/CPU-20□ SVR32
- MP3000/CPU-30□ SVC
- MP3000/CPU-30□ SVR
- MP2000/SVA-01
- MP2000/SVB-01
- MP2000/SVC-01
- MP2000/PO-01

A.2 控制器規格一覽表

下表將說明適用控制器的程式規格。

規格		CPU-20□, CPU-30□	備註
運動程式	程式數	最多 512 個	最多可編寫共 512 個運動程式 / 序列程式。
	群組數	16 群組	–
	任務數	最多 32 個任務 (可同時執行的運動程式數)	–
	並列處理數 (每個任務)	最多 8 個並列 (有以下 4 種模式可供選擇) ・主程式 4 並列 x 子程式 2 並列 ・主程式 8 並列 ・主程式 2 並列 x 子程式 4 並列 ・子程式 8 並列	使用 MPE720，即可切換模式。
	執行登錄	・從階梯圖程式使用 MSEE 指令 ・使用 M-EXECUTOR	–
	啟動方法	因檢測到控制訊號 Bit 0 (要求程式開始運轉) 啟動而啟動程式	–
	對定位速度進行覆寫	設定範圍為 0.01% ~ 327.67%	–
	動作模式	ABS/INC 模式	使用專用指令 (ABS/INC)，即可切換模式。
	指令單位	・SVC/SVC 32/SVC-01/SVB-01/SVR/SVR 32 pulse，mm，deg，inch，μm ・SVA-01/PO-01 pulse，mm，deg，inch	–
	最小指令單位	・pulse 1 ・mm，deg，inch，μm 1，0.1，0.01，0.001，0.0001，0.00001	–
	指令範圍	-2147483648 ~ +2147483647 (32 位元附加符號)	–
	同步控制軸數 (每個任務)	・定位、線性內插、原點復歸、附略過 (Skip) 功能線性內插、指定時間定位 最多 32 軸 ・循環內插 2 軸 ・螺旋內插 3 軸 ・外部定位 1 軸	–
序列程式	程式數	最多 512 個 (可利用啟動處理、高速掃描處理、低速掃描處理來選擇執行時間點)	最多可編寫共 512 個運動程式 / 序列程式。
	任務數	最多 32 任務 (可同時執行的序列程式數)	–
	並列處理數 (每個任務)	無法使用並列處理 (PFORK 指令)。	–
	執行登錄	使用 M-EXECUTOR	–
	啟動方法	利用系統啟動	將程式登錄至 M-EXECUTOR，即可利用系統啟動。

(續下頁)

(續上頁)

規格		CPU-20□, CPU-30□	備註
可存取之暫存器	M 暫存器	1048576 字元	電池組備用記憶體
	S 暫存器	65535 字元	電池組備用記憶體
	G 暫存器	2097152 字元	無法利用各程式共用的電池組備用的暫存器。
	I 暫存器	65536 字元 + 設定參數 + CPUIF 專用	–
	O 暫存器	65536 字元 + 監控參數 + CPUIF 專用	–
	C 暫存器	16384 字元	–
	D 暫存器	可指定 0 ~ 16384 字元	各 DWG 內固有的內部暫存器。僅可使用於相應的 DWG。

範例程式

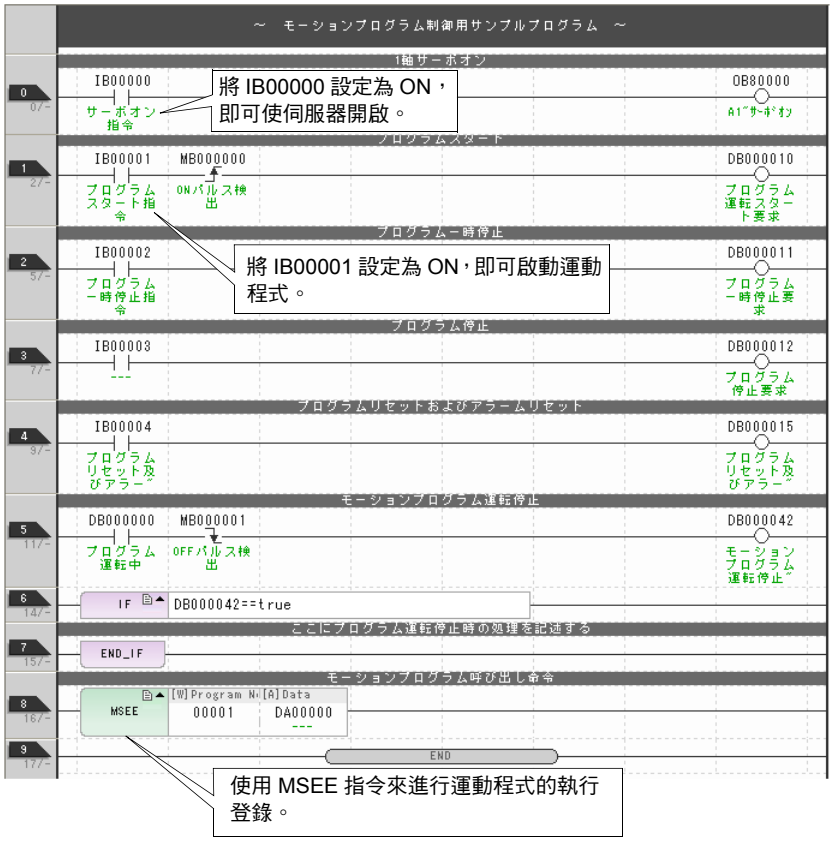
附錄 B

本章將針對運動程式和序列程式之程式範例進行說明。

B.1 運動程式控制專用程式

對運動程式進行執行控制的範例程式。

以下為階梯圖程式的程式範例。

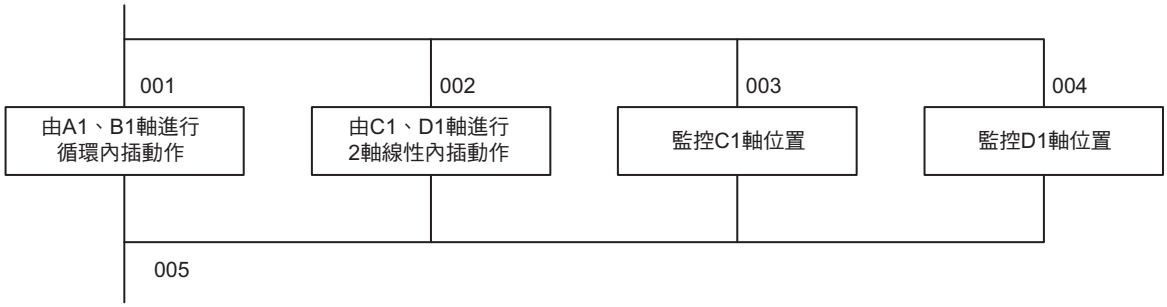


B.2 並列處理

以下係運動程式中使用 PFORK 指令來並列執行作業之範例程式。

```
      ABS;                                    " 絕對模式
      PFORK 001 002 003 004;                  ← 開始並列處理
001:FMX T10000K;                              " 設定最高插補速度
      PLN [A1] [B1];                          " 設定圓弧平面
      MCC [A1] 0 [B1]0 U100000 V0 F10000K;    " 循環內插   ← 第1個並列：執行循環內插
      JOINTO 005;
002:FMX T10000K;                              " 設定最高插補速度
      MVS [C1] 131072 [D1] 20000 F10000K;     " 線性內插   ← 第2個並列：執行線性內插
      JOINTO 005;
003:IOW IL8016>130000;                        " 監控C1軸位置
      OB00000=1;                              " 到達後，OB00000將會開啟 ← 第3個並列：監控C1軸位置
      JOINTO 005;
004:IOW IL8096>12000;                         "監控D1軸位置
      OB00001=1;                              " 到達後，OB00001將會開啟 ← 第4個並列：監控D1軸位置
      JOINTO 005;
005:PJOINT;
      END;
```

以下為範例程式的動作內容。

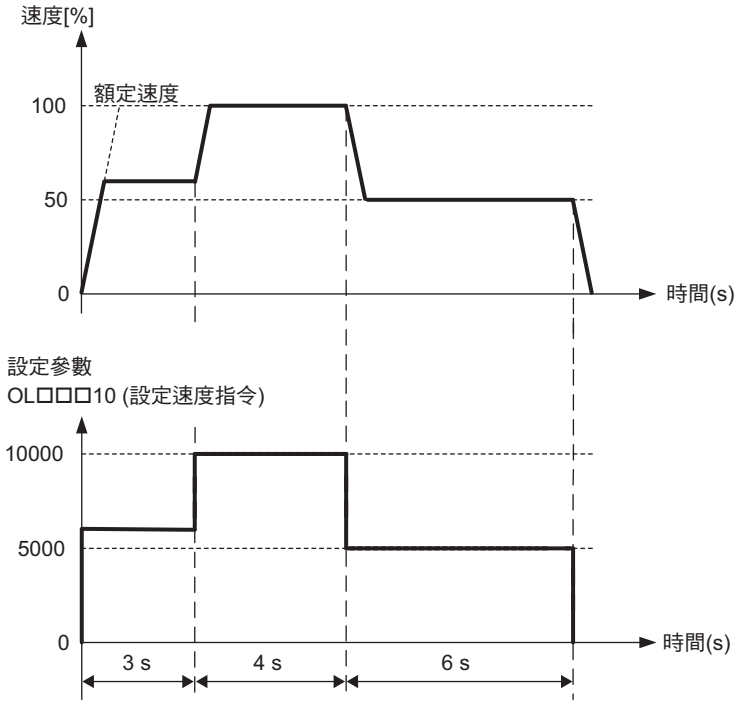


B.3 利用運動程式進行速度控制

以下係為以運動程式來執行速度控制之範例程式。
此範例係利用設定參數 OW□□□03 Bit 0 ~ 3 (選擇速度單位)，設定為 0.01%(指定額定速度的比率)。

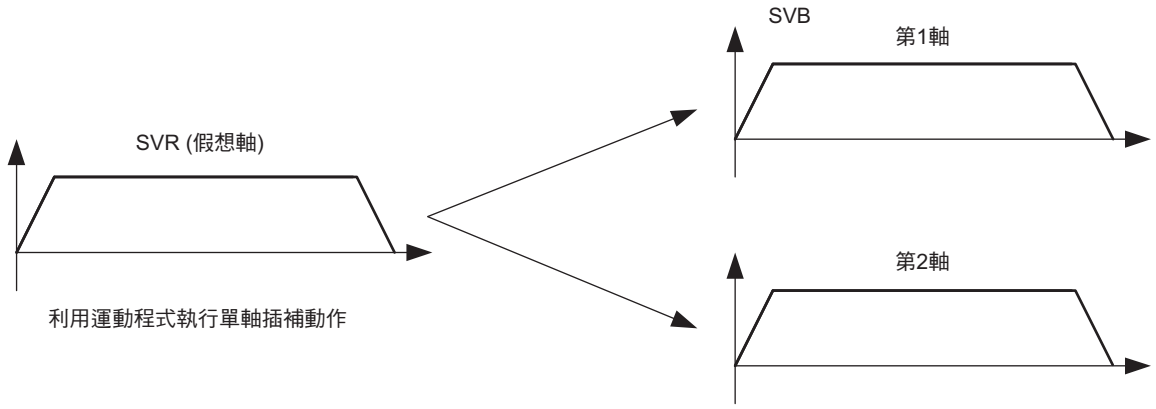
```
OW8008=23;          " 速度控制模式
OL8010=6000;        " 變更速度　額定速度的 60%
TIM T300;           " 等待 3 秒
OL8010=10000;       " 變更速度　額定速度
TIM T400;           " 等待 4 秒
OL8010=5000;        " 變更速度　額定速度的 50%
TIM T600;           " 等待 6 秒
OW8008=0;           " 停止速度控制模式
END;
```

以下為範例程式的動作類型。



B.4 使用假想軸進行簡易的同步運轉

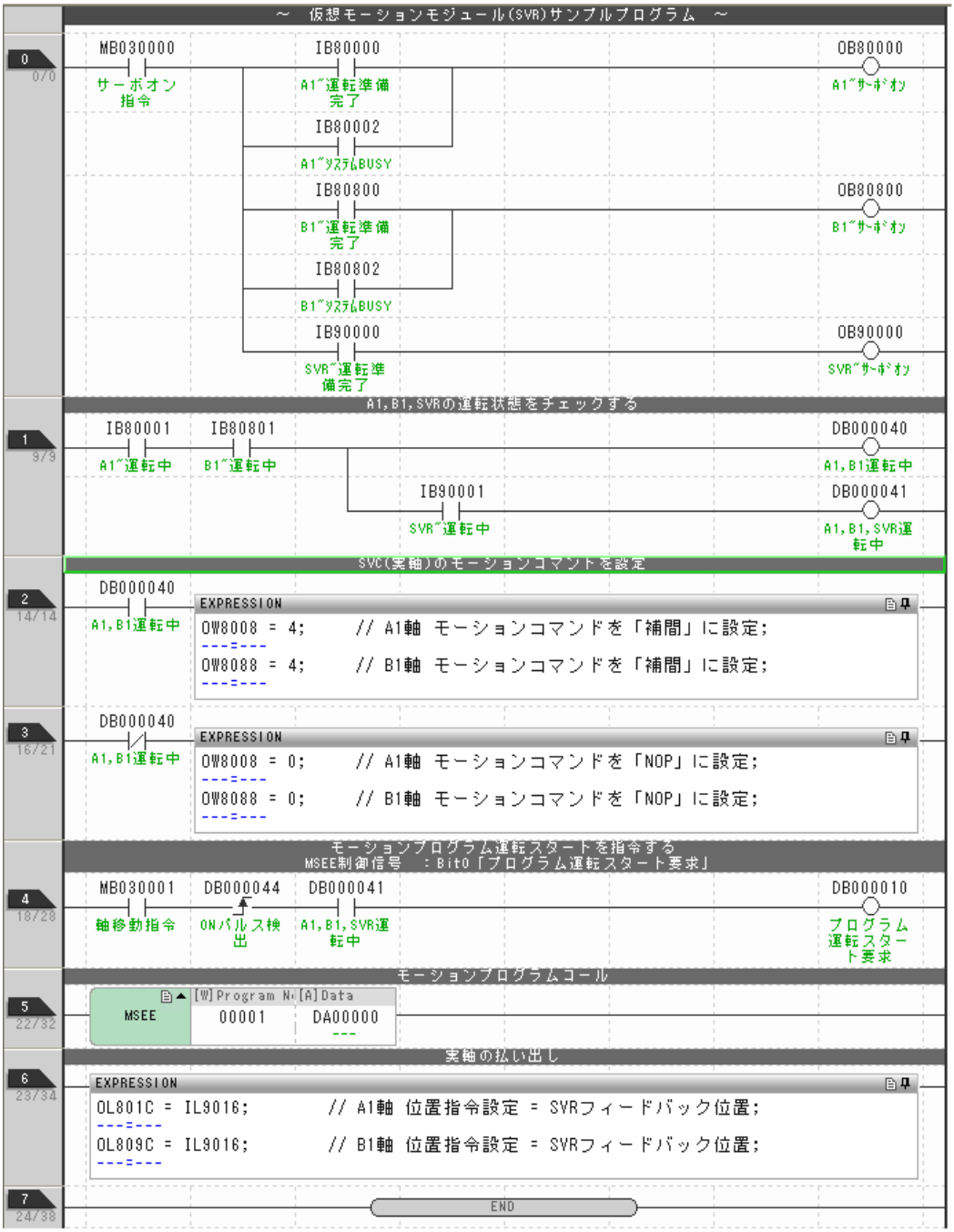
此範例程式係利用運動程式來移動 SVR (假想軸)，再將 SVR (假想軸) 的反饋位置分配給 2 個實軸，以對 2 軸進行同步運轉。



■ **運動程式**

```
FMX T10000K;                  " 設定最高插補速度 K = 1000
INC;                          " 增量模式
IAC T500;                     " 插補加速時間 = 500 ms
IDC T500;                     " 插補減速時間 = 500 ms
MVS [SVR] 1000K F10000K;      " 移動距離 1000000 的插補動作
END;
```

■ 階梯圖程式



重要

本程式省略了軸異常時之復歸方法。使用本程式前，應先預設可能會發生的異常狀況，並新增程式以達到安全運轉的目的。

B.5 序列程式

此範例程式係使用序列程式，讓 1 軸的伺服馬達執行 JOG 運轉及 STEP 運轉。

■ 序列主程式 (SPM001)

```
"SPM001：主程式 "

SSEE SPS002;              " 軸共用設定處理
SSEE SPS003;              "JOG & STEP 動作處理
END;
```

■ 序列子程式 (SPS002)

```
"SPS002：軸共用設定處理"

"-------------------------------
"  檢測到運動指令0
"-------------------------------
IF IW8008 = = 0;
        MB300010 = 1;
ELSE;
        MB300010 = 0;
IEND;

"-------------------------------
"  伺服器開啟指令
"-------------------------------
OB80000 = MB300000 & (IB80000 | IB80002);          "伺服器開啟

"-------------------------------
"  重置警報
"-------------------------------
OB8000F = MB300001;                                "重置警報

"-----------------------------------------------------------------------
"  選擇   速度單位&加減速度單位
"
"    Bit 0 ~ 3：選擇速度單位(0：指令單位/s、1：指令單位/min、2：指定%)
"    Bit 4 ~ 7：選擇加減速度單位(0：指令單位/s^2、1：ms單位)
"-----------------------------------------------------------------------
DW00010 = OW8003 & FF00H;                          "功能設定1工作
OW8003 = DW00010 | 0011H;                          "功能設定1

"-------------------------------
"  設定直線加速/減速度
"-------------------------------
IF MB300020 = = 1;
        OL8036 = 100;                              "直線加速度/加速時間常數
        OL8038 = 100;                              "直線減速度/減速時間常數
IEND;

RET;
```

將MB300000設定為開啟，即可開啟伺服器。

■ 序列子程式 (SPS003)

"SPS003：JOG & STEP 動作處理"

```
"-------------------------------------
"  JOG運轉
"-------------------------------------
IF IB80001 & ( (DB000010 & !DB000011)｜(!DB000010 & DB000011) ) = = 1;
      DB000000 = 1;
ELSE;
      DB000000 = 0;
IEND;

DB000001 = PON( DB000000 DB000050 ) & MB300010;          "開始JOG
DB000002 = NON( DB000000 DB000051 );                     "停止JOG

IF DB000001 = = 1;
      OL8010 = 1000;
      OW8008 = 7;                                        "運動指令FEED
IEND;
IF DB000002 = = 1;
      OW8008 = 0;                                        "運動指令NOP
IEND;

"-------------------------------------
"  STEP運轉
"-------------------------------------
IF IB80001 & ( (DB000012 & !DB000013)｜(!DB000012 & DB000013) ) = = 1;
      DB000008 = 1;
ELSE;
      DB000008 = 0;
IEND;

DB000009 = PON( DB000008 DB000058 ) & MB300010;          "開始STEP
DB00000A = NON( DB000008 DB000059 );                     "停止STEP

IF DB000009 = = 1;
      OL8010 = 1000;                                     "設定STEP速度
      OL8044 = 1000;                                     "設定STEP移動量(1000pulse)
      OW8008 = 8;                                        "運動指令STEP
IEND;

IF DB00000A = = 1;
      OW8008 = 0;                                        "運動指令NOP
IEND;

"-------------------------------------
"   選擇逆旋轉
"-------------------------------------
OB80092 = ( DB000000 & DB000011 )｜( DB000008 & DB000013 );          "選擇逆旋轉

RET;
```

將DB000010設定為開啟 後，即開始JOG運轉(正轉)。

將DB000011設定為開啟後，即開始JOG運轉(逆轉)。

將DB000012設定為開啟後，即開始STEP運轉(正轉)。

將DB000013設定為開啟後，即開始STEP運轉(逆轉)。

MP2000 系列和 MP3000 系列間的差異

附錄 C

本章將說明和運動程式相關的 MP2000 系列與 MP3000 系列之間的差異。

■ 運動程式

項目	MP2000 系列	MP3000 系列	備註
程式數	256 個	512 個	運動程式和序列程式的總和
群組數	8	16	–
任務數	16	32	可同時執行的程式數
每個群組最大控制軸數	16 軸	32 軸	使用「SVC、SVC32」和「SVB-01、SVC-01」時，透過 MECHATROLINK 連結，傳送到從屬端時將會出現時間差，所以無法在「SVC、SVC32」與「SVB-01、SVC-01」之間進行插補動作。
並列處理數	主程式 4 並列 x 子程式 2 並列	從以下 4 種中選擇 ・主程式 4 並列 x 子程式 2 並列 ・主程式 8 並列 ・主程式 2 並列 x 子程式 4 並列 ・子程式 8 並列	☞ **程式的並列執行 (第 1-7 頁)**
G 暫存器	不可使用	可使用	–
4 長整數	不可使用	可使用	無法使用運動語言指令間接指定
倍精度實數	不可使用	可使用	無法使用運動語言指令間接指定
陣列型	不可使用	可使用	–

■ 除錯運轉

項目	MP2000 系列	MP3000 系列	備註
斷點點數	4	8	–

■ 下達軸移動指令時若發生警報，運動程式可能出現的動作

有別於 MP2000 系列，MP3000 系列會針對下達軸移動指令的所有轉軸進行異常確認。發生警報時，會讓下達指令的所有轉軸停止動作，然後再下達運動指令 NOP。因此，當 MP3000 系列已下達指令的轉軸發生警報時，您不需要再利用應用程式進行互鎖，因此安全性比 MP2000 系列來得高。

補充

在軸移動指令執行中，指令對象的轉軸發生警報時，運動程式可能會出現的動作如下表所示。下表以後的版本，可選擇「MP2000 互換模式」作為運動程式的動作，因此可以將利用 MP2000 系列所編寫之含有互鎖動作的應用程式直接應用在 MP3000。

控制器及 MPE720	適用版本
MP3000 系列	Ver 1.08 以後版本
MPE720 Ver.7	Ver 7.21 以後版本

下表為被下達軸移動指令的轉軸一旦發生警報時，運動程式可能會出現的動作。

軸移動指令	MP2000 系列			MP3000 系列		
	發生警報的轉軸	未發生警報的轉軸	運動程式動作	發生警報的轉軸	未發生警報的轉軸	運動程式動作
定位 (MOV)、指定時間定位 (MVT)	停止	移動至目標位置	持續對未發生警報的轉軸下達指令，移動至目標位置	停止	停止	當運動程式發生警報，所有被下達指令的轉軸將中斷指令執行 (警報代碼：8Fh (軸警報發生中))
外部定位 (EXM)	停止	–	指令將持續執行動作，直到運動指令的狀態 IW□□□09 Bit 8 (指令執行完成) 成為開啟	停止	–	
原點復歸 (ZRN)	停止	移動至原點	指令將持續執行動作，直到下達指令的所有轉軸的位置管理狀態 IW□□□0C Bit 5 (原點復歸 (設定) 完成) 成為開啟	停止	停止	
線性內插 (MVS) *、循環內插、螺旋內插 (MCW、MCC) *、附略過功能線性內插 (SKP) *	停止	停止	當運動程式發生警報，所有被下達指令的轉軸將中斷指令執行 (警報代碼：84h (重複運動指令))	停止	停止	

* 當被下達插補指令的轉軸發生軟體限制警報時，MP2000 系列和 MP3000 系列動作就各異。
如欲進一步瞭解，請參閱以下項目之說明。

☞ ■ 被下達插補指令的轉軸在發生軟體限制警報時動作

(註)MP2000 系列在執行外部定位 (EXM)、原點復歸 (ZRN) 時，運動程式的執行區塊將不會改變，因此必須先執行要求程式停止，然後再執行要求程式重置及警報重置。

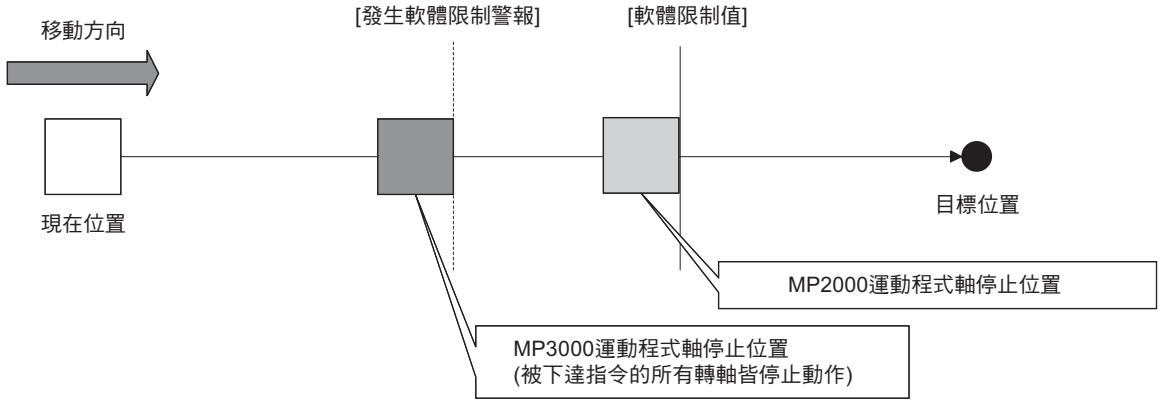
■ **被下達插補指令的轉軸在發生軟體限制警報時動作**

軟體限制警報為在到達軟體限制值之前發出警報，以避免數值超過您所設定的軟體限制值。

MP3000 系列會在發生軸警報時，停止轉軸動作，因此被下達插補指令的所有轉軸會在到達您所設定的軟體限制值之前停止動作。

MP2000 系列可讓發生軟體限制警報的轉軸在到達軟體限制值之前停止轉軸。未發生軟體限制警報的轉軸則會繼續移動至目標位置。

內容	MP2000 系列	MP3000 系列
發生軟體限制警報軸	移動轉軸直至到達軟體限制值	一發生警報，即停止轉軸動作 (到達軟體限制值前，停止轉軸動作)
未發生軟體限制警報軸	移動轉軸直至到達目標位置	一發生警報，即停止轉軸動作
運動程式警報代碼	無	8Fh (目前發生軸警報)



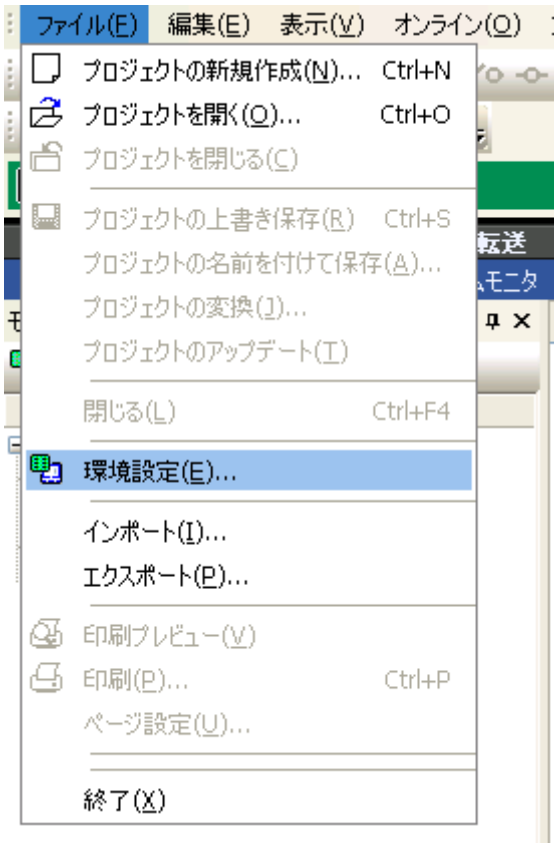
■ **檢查軸警報之設定方法**

接下來將說明如何設定檢查軸警報功能。

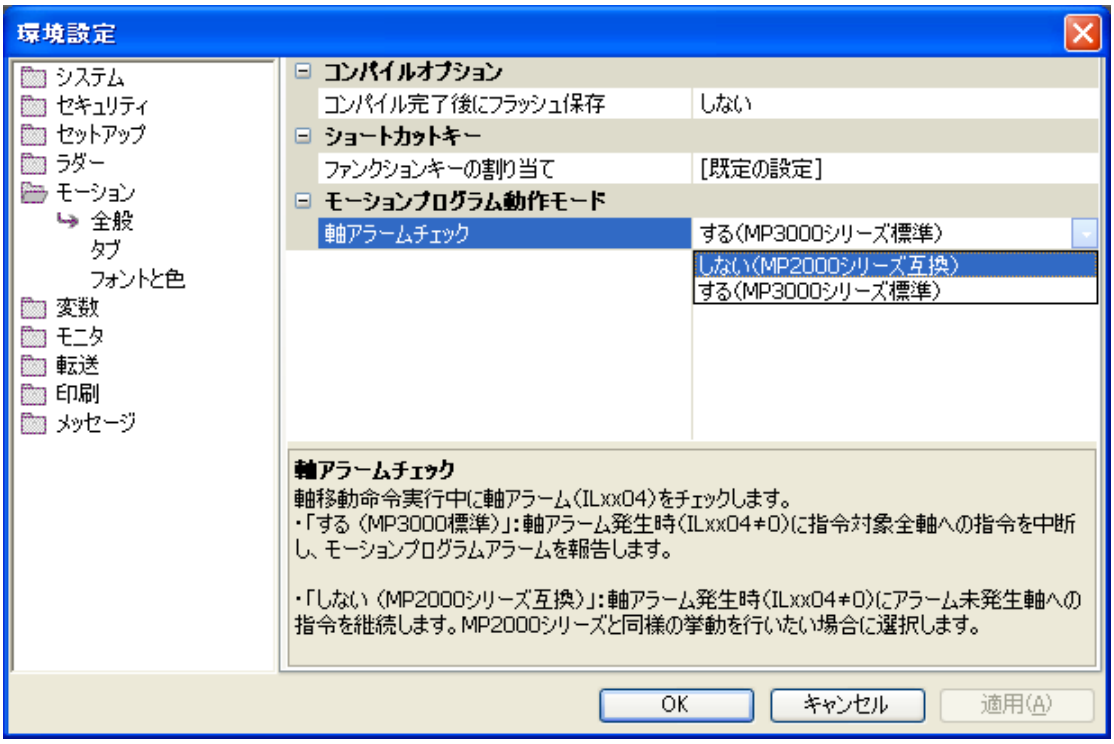
若要設定檢查軸警報功能，必須先進入 MPE720 Ver. 7 的 [環境設定] 對話框。

出廠時已將檢查軸警報功能設定為啟用。

1. 依序 [檔案] － [環境設定]。



2. 進入 [環境設定] 對話框，並從 [運動] 方塊中選擇運動程式的動作模式。



■ 軸警報檢查設定的啟用時間

進入 [環境設定] 對話框，並點擊 [OK] 鍵後，即可開啟軸警報檢查設定啟用時間。

注意事項

附錄 D

接下來將說明運動程式和序列程式之注意事項。

D.1 一般注意事項

接下來將說明運動程式和序列程式的一般注意事項。

關於在變更應用程式時儲存至快閃記憶體的方式

變更運動程式、序列程式等應用程式後，務必將變更後的內容儲存在快閃記憶體中。若未儲存至快閃記憶體就直接關閉控制器電源，變更過的應用程式所有內容將會全部消失。

對運轉中的系統進行除錯

嚴禁對運轉中的系統進行除錯運轉。執行除錯運轉，將因此造成時序改變，因而導致程式錯誤動作或裝置故障。
執行除錯運轉時，需使用除錯專用的系統。

D.2 運動參數相關注意事項

接下來將說明使用運動程式和運動參數時之注意事項。

利用運動程式對同一個軸下達軸移動指令

若在執行軸移動指令時，同時利用其他運動程式對同一個轉軸下達軸移動指令，設定參數 OW□□□09 Bit 5 (位置指令型) 將出現不同的轉軸動作。

以下將說明不同的位置指令類型之轉軸動作。

增量值加算方式

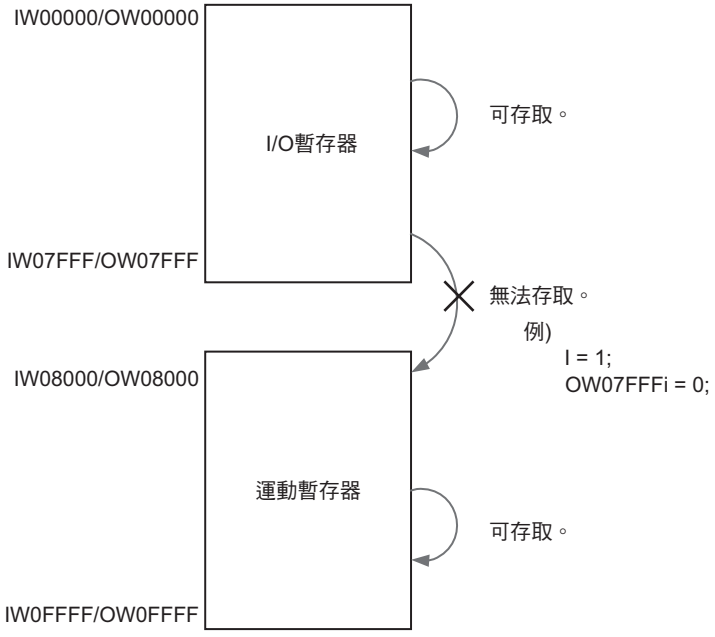
定位至加上兩方的運動程式的指令位置。最終到達位置和所有指令位置皆相異。

絕對值指令方式

定位至後來所下達指令的目標位置。

關於使用索引，並利用 I/O 暫存器來參照運動暫存器時

I/O 暫存器和運動暫存器無法被配置到連續的記憶體中。
使用索引時，必須能夠存取 I/O 暫存器或運動暫存器所在範圍內的所有暫存器。

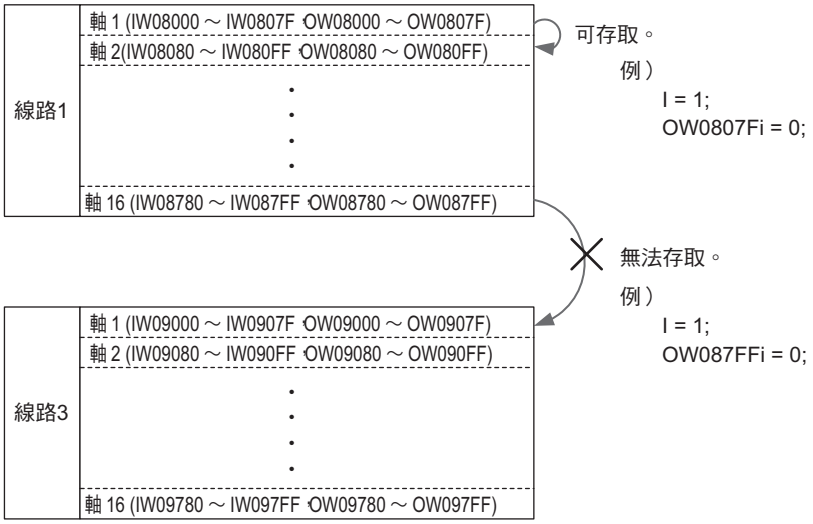


關於不同線路的運動暫存器的參照時

與 I/O 暫存器和運動暫存器之間的關係相同，不同線路的運動暫存器並不會被配置到連續的記憶體中。
使用索引時，請存取每個線路所被配置的運動暫存器範圍內的暫存器。
若線路相同，則可跨越轉軸存取運動暫存器。
以下為運動暫存器一覽表。

線路編號	1 軸	2 軸	・・・	16 軸
1	OW08000 ～ OW0807F	OW08080 ～ OW080FF	・・・	OW08780 ～ OW087FF
3	OW09000 ～ OW0907F	OW09080 ～ OW090FF	・・・	OW09780 ～ OW097FF
5	OW0A000 ～ OW0A07F	OW0A080 ～ OW0A0FF	・・・	OW0A780 ～ OW0A7FF
7	OW0B000 ～ OW0B07F	OW0B080 ～ OW0B0FF	・・・	OW0B780 ～ OW0B7FF
9	OW0C000 ～ OW0C07F	OW0C080 ～ OW0C0FF	・・・	OW0C780 ～ OW0C7FF
11	OW0D000 ～ OW0D07F	OW0D080 ～ OW0D0FF	・・・	OW0D780 ～ OW0D7FF
13	OW0E000 ～ OW0E07F	OW0E080 ～ OW0E0FF	・・・	OW0E780 ～ OW0E7FF
15	OW0F000 ～ OW0F07F	OW0F080 ～ OW0F0FF	・・・	OW0F780 ～ OW0F7FF

範例　**存取不同線路的運動暫存器**



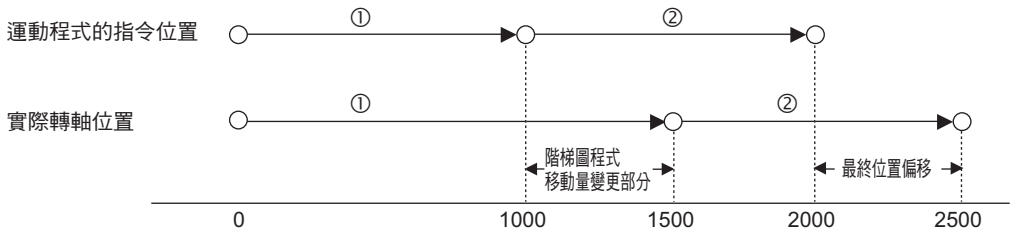
設定參數 OL□□□1C (設定位置指令)

利用運動程式來移動轉軸時，只要同時使用其他程式 (階梯圖程式等) 來變更 OL□□□1C (設定位置指令)，就會根據變更後的內容，來移動轉軸。此時運動程式所下達的指令位置將和實際的轉軸位置發生落差。

範例

利用以下的運動程式來執行 ① 時，若是同時利用階梯圖程式，將 OL□□□1C (設定位置指令) 的 A1 軸移動量由 +1000 變更為 +1500，A1 軸將會移動至 +1500 的位置。因此與運動程式的指令位置 (+1000) 之間會產生落差。
運動程式將接著執行 ②。結果連 A1 軸實際的最終位置也和運動程式的指令位置不同。

```
INC;
ZRN [A1]0;

MOV [A1]1000; ・・・①
MOV [A1]1000; ・・・②

END;
```



發生軟體限制警報時的轉軸動作

在執行插補指令時，一旦發生軟體限制警報 (IL□□□04 Bit 3、Bit 4)，根據設定速度將造成轉軸在到達軟體限制值之前停止動作。停止位置依設定速度而異。

索引

符號

數字

A

B

C

D

E

F

G

I

J

M

5 劃

6 劃

7 劃

8 劃

9 劃

10 劃

11 劃

12 劃

13 劃

14 劃

15 劃

16 劃

17 劃

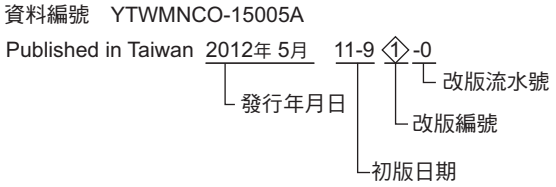
18 劃

19 劃

23 劃

修訂記錄

修訂相關資訊及資料編號等刊載於本書封底右下方。



發行年 / 月	改版編號	改版序號	項目編號	變更項目
2014 年 3 月	◇3	0	所有章節	新增：MP3300 相關資訊
			5.4	新增：FUT 指令、IUT 指令相關資訊
			6.1	新增：FUT 指令、IUT 指令、ACCMODE 指令相關資訊
			附錄 C	新增：軸警報檢查的設定方法、軸警報檢查設定的有效時機
			封底	變更：位址
2013 年 7 月	◇2	0	所有章節	部分修訂
				變更：變更專有名詞 (基本元件 → MP3200)
				新增：軸警報相關資訊
				新增：F 型指令相關資訊
				新增：影像指令相關資訊
2012 年 7 月	◇1	1	6.2	MVS 指令設定項目 (插補進給速度) 範例
2012 年 5 月		0	所有章節	部分修訂
2011 年 9 月	–	–	–	初版發行

運動控制器　MP3000系列

運動程式
程式編寫手冊

台灣安川電機股份有限公司

事務所/技術服務中心
地址：23143新北市新店區北新路3段207號12樓
TEL: (02)8913-1333　FAX: (02)8913-1513/1519

台南服務中心
地址：74144台南市新市區創業路18號2樓
TEL: (06)505-1432　FAX: (06)505-6405

代理商 / 經銷商

YASKAWA

資料編號　YTWMNCO-15005A
Published in Taiwan　2014年3月　11-9 ◇-0
13-12-9