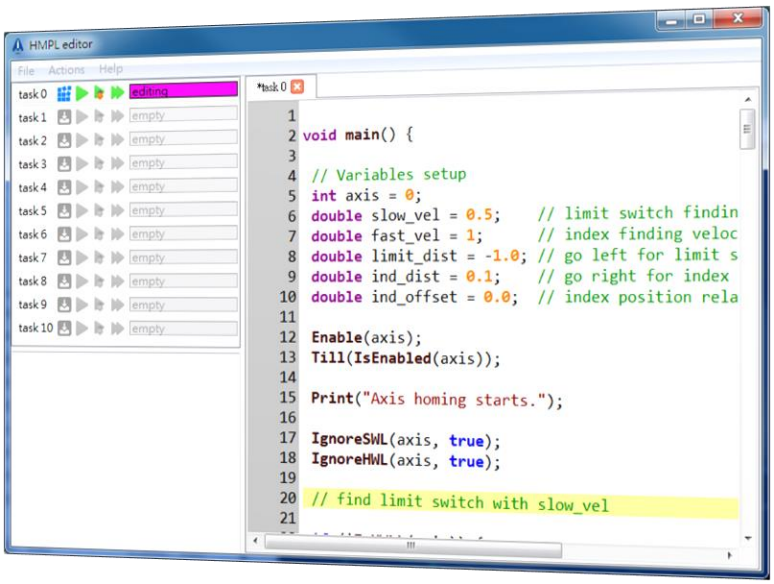
HIMC

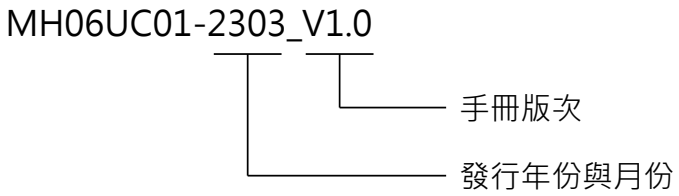
HMPL 使用手冊

```
Print("Hello World");
```



修訂紀錄

手冊版次資訊亦標記於手冊封面右下角。



發行日期	版次	適用軟體	更新內容
2023/03/15	1.0	iA Studio 3.0.0	1. HIMC 支援 CoE 通訊：新增 Read/Write SDO 與 PDO 函式，移除 Get/Set Slave Var 與 Run PDL 相關函式 2. 新增函式 ■ 第 2 章 IsSystemInit、GetECATSt、GetSlvECATSt、ScanNetwork ■ 第 5 章 SetOpMode、SetBufferMode ■ 第 9 章 SetSlvAOHex、GetSlvAOHex ■ 第 13 章 SetTouchProbeFunc ■ 第 18 章 GetDriveErr ■ 第 20 章 MoveHome、SetHomeSwitchVel、SetHomeZeroVel、SetHomeAcc、IsHomed、IsHoming 3. 新增範例：8.1.6 節 4. 移除第 12 章，Random PT 相關功能。
2022/06/30	0.9	iA Studio 2.0	1. 新增章節： ■ 第 10 章—AIO 函式 ■ 21.3 節—Modbus 通訊 2. 新增函式： ■ 第 2 章 SendMsgEvent ■ 第 5 章

發行日期	版次	適用軟體	更新內容
			MoveTrq、MovePVT、IsAcc ■ 第 8 章 ■ ArcAngle2D、SetGrpAngMotionProfile、GetGrpCoordTrans、SetGrpCoordTrans、GetGrpPoseCmd、GetGrpPoseFb、CircleRel ■ 第 9 章 SetGPIInvert、SetGPOInvert、BindEMO、GetAllGPIInvertSt、GetAllGPOInvertSt ■ 第 14 章 SetCompAlgType ■ 第 20 章 AxisHome、SetHomeType、SetHomeMethod、SetHomeProfile、SetHomeOffset、SetHomeTimeout、SetEndStopPosErr、SetEndStopDist 3. 新增範例：8.1.6 節 4. 新增變數：5.1.1 節、8.1.1 節 5. 新增錯誤訊息：18.1.1 節、18.1.2 節、18.1.3 節
2021/12/24	0.8	iA Studio 1.4	1. 新增函式： ■ 第 11 章 SetPT_PosArray、SetPT_StateArray、SetPT_StartIndex、SetPT_EndIndex 2. 新增範例：11.1.3 節 3. 移除範例：8.1.6 節
2021/09/15	0.7	iA Studio 1.4	1. 新增章節： ■ 9.1.1 節—GPIO 變數 ■ 第 20 章—通訊函式 2. 新增函式： ■ 第 2 章 GetFirmwareVer ■ 第 5 章 GetVelFb、GetVelErr、GetCurrFb、SetVelScale、GetVelScale、SetRollover、GetRolloverTurns、IsDriveErr、IsPosErr ■ 第 8 章 JogGroup、JogGroupAxis、SetGrpVelScale、GetGrpVelScale

發行日期	版次	適用軟體	更新內容
			■ 第 14 章 SetupComp3D 3. 新增範例：SetHMIScope、8.1.6 節、9.1.2 節、13.1.1 節、19.3.1 節 4. 新增變數： 5.1.1 節、8.1.1 節、16.1.2 節 5. 新增錯誤訊息： 17.1.1 節、17.1.2 節、17.1.3 節 6. 新增歸原點方法： 19.1 節
2020/09/16	0.6	iA Studio 1.3	1. 13.1.1 節：修改圖片與程式碼。 2. 15.1 節：新增敘述。 3. 16.1.2 節：修改表 16.1.2.3 的註解。
2020/06/30	0.5	iA Studio 1.3	1. 單位系統變更： 2. 公尺-弧度-秒 → 毫米-角度-毫秒。 3. 新增章節： ■ 第 18 章 巨集定義與函式 ■ 第 19 章 歸原點 4. 新增 / 修改第 5、7、8、9、10、11、12、13、15、16、17 章之概述。 5. 新增函式： ■ 第 5 章 Halt、Resume、SetAccTime、SetDecTime、IgnorePE ■ 第 6 章 GetGearRatio、IsInGear、IsGearMaster、IsGearSlave ■ 第 7 章 GetGantryPairID、IsGantryPair ■ 第 8 章 ■ HaltGroup、ResumeGroup、ArcCW2D、ArcCCW2D、GetGrpKin、GetGrpMaxVel、SetGrpVel、GetGrpMaxAcc、SetGrpAcc、SetGrpAccTime、GetGrpMaxDec、SetGrpDec、SetGrpDecTime、GetGrpSMTime、SetGrpSMTime、GetGrpCoordSys、SetGrpCoordSys、GetGrpBufferMode、SetGrpBufferMode、

發行日期	版次	適用軟體	更新內容
			GetGrpTransMode、SetGrpTransMode、SetGrpTransPrm、GetGrpCmdNum ■ 第 9 章 SetAllGPO、SetSlvAllGPO、GetAllGPI、GetAllGPO、GetSlvAllGPI、GetSlvAllGPO ■ 第 13 章 GetCompPos ■ 第 16 章 RunSlvPdlFunc、StopSlvPdlFunc、IsSlvPdlFuncRunning ■ 第 19 章 Home 6. 移除函式： ■ 第 8 章 Bezier ■ 第 12 章 SetPT_Polarity 7. 函式更名： ■ 第 8 章 GroupReset → ResetGroup
2020/04/28	0.4	iA Studio 1.2.4107.1	1. **第 18 章**：修改基本版、進階版與 E1 系列驅動器龍門模式歸原點範例。
2019/11/29	0.3	iA Studio 1.2.4032.0	1. **2.10 節**：Till 函式新增備註。 2. **第 11 章**：新增 PT 功能使用流程。 3. **第 18 章**：新增 E1 系列驅動器龍門模式歸原點範例。
2019/04/02	0.1	iA Studio 1.1.3772.0	第一版發行。

相關文件

產品相關文件的關係圖表如下，請視需要參閱文件。

產品		文件名稱	文件編號	內容
控制器	HIMC 運動控制器	HIMC 安裝指南	MH07UC01-□□□□	詳細說明 HIMC 運動控制器的安裝、連接等。
		HIMC iA Studio 軟體使用手冊	MH01UC01-□□□□	詳細說明 HIMC 運動控制器的人機介面操作。
		HIMC Modbus TCP 使用手冊	MH02UC01-□□□□	詳細說明 Modbus TCP 通訊協定應用於 HIMC 運動控制器的方式。
		HIMC HMPL 使用手冊	MH06UC01-□□□□	詳細說明 HIMC 運動控制器的 HMPL 函式庫。
		HIMC API 參考指南	MH05UC01-□□□□	詳細說明 HIMC 運動控制器的 API 函式庫。
		HIOM 安裝指南	MH03UC01-□□□□	詳細說明 HIOM（HIWIN mega-ulink IO 模組）的安裝、連接等。
		ETA3 安裝指南	MH09UC01-□□□□	詳細說明 ETA3（HIMC 遠端模組）的安裝、連接等。
驅動器	E 系列 驅動器	E1 系列驅動器使用者操作手冊	MD09UC01-□□□□	詳細說明 E1 系列驅動器的選擇、安裝、連接、設定、試運轉、調機、監控等。
		E2 系列驅動器使用者操作手冊	MD28UC01-□□□□	詳細說明 E2 系列驅動器的選擇、安裝、連接、設定、試運轉、調機、監控等。
		E1 系列驅動器 Thunder 軟體操作手冊	MD12UC01-□□□□	詳細說明 E1 系列驅動器的人機介面操作。
		E1 系列驅動器 EtherCAT(CoE) 通訊命令手冊	MD08UC01-□□□□	詳細說明 EtherCAT 通訊協定應用於 E1 系列驅動器的方式。
		E1 系列驅動器 MECHATROLINK-III 通訊命令手冊	MD24UC01-□□□□	詳細說明 MECHATROLINK-III 通訊協定應用於 E1 系列驅動器的方式。
		E1 系列驅動器 PROFINET 通訊命令手冊	MD02UC01-□□□□	詳細說明 PROFINET 通訊協定應用於 E1 系列驅動器的方式。
		E1 系列驅動器龍門控制系統 使用者操作手冊	MD22UC01-□□□□	詳細說明 E1 系列驅動器龍門控制系統的使用方式。
		E1 系列驅動器電子凸輪控制系統 使用者操作手冊	MD27UC01-□□□□	詳細說明 E1 系列驅動器電子凸輪控制系統的使用方式。
		E1 系列驅動器多工位功能 使用者操作手冊	MD32UC01-□□□□	詳細說明 E1 系列驅動器多工位功能的使用方式。
		MPI 函式庫參考手冊	MD19UC01-□□□□	詳細說明 E1 系列驅動器與 D 系列驅動器的 MPI 函式庫。
		MPI 範例程式	MD18UC01-□□□□	詳細說明 E1 系列驅動器與 D 系列驅動器的 MPI 範例程式。
		驅動器 API 函式庫參考手冊	MD23UC01-□□□□	詳細說明 E1 系列驅動器與 D 系列驅動器的 API 函式庫。
		E1 系列驅動器 PDL 範例程式	MD25UC01-□□□□	詳細說明 E1 系列驅動器的 PDL 範例程式。
		Application Note E1 PROFINET 驅動器搭配 Siemens TIA Portal	MD30UC01-□□□□	詳細說明 E1 PROFINET 驅動器搭配 Siemens S7 系列 PLC 時，PLC 軟體 TIA Portal 的操作。
		Application Note E1 MECHATROLINK-III 驅動器搭配 YASKAWA MPE720	MD31UC01-□□□□	詳細說明 E1 MECHATROLINK-III 驅動器搭配 YASKAWA MP3000 系列運動控制器時，運動控制器軟體 MPE720 的操作。
	D 系列 驅動器	D1 驅動器使用者操作手冊	MD20UC01-□□□□	詳細說明 D1 驅動器的選擇、安裝、連接、設定、試運轉、調機、監控等。
		D2 驅動器使用者操作手冊	MD07UC01-□□□□	詳細說明 D2T 驅動器的選擇、安裝、連接、設定、試運轉、調機、監控等。
		D2T-LM 系列驅動器使用者操作手冊	MD11UC01-□□□□	詳細說明 D2T-LM 驅動器的選擇、安裝、連接、設定、試運轉、調機、監控等。

產品		文件名稱	文件編號	內容
		MPI 函式庫參考手冊	MD19UC01-□□□□	詳細說明 E1 系列驅動器與 D 系列驅動器的 MPI 函式庫。
		MPI 範例程式	MD18UC01-□□□□	詳細說明 E1 系列驅動器與 D 系列驅動器的 MPI 範例程式。
		驅動器 API 函式庫參考手冊	MD23UC01-□□□□	詳細說明 E1 系列驅動器與 D 系列驅動器的 API 函式庫。
		PDL 範例程式使用者操作手冊	MD13UC01-□□□□	詳細說明 D 系列驅動器的 PDL 範例程式。
馬達	線性馬達	線性馬達使用者操作手冊	MP99UC01-□□□□	詳細說明線性馬達的選擇、安裝、連接等。
	直驅馬達	DMN 系列直驅馬達使用者操作手冊	MR01UC01-□□□□	詳細說明 DMN 系列直驅馬達的選擇、安裝、連接等。
		DMT 系列直驅馬達使用者操作手冊	MR03UC01-□□□□	詳細說明 DMT 系列直驅馬達的選擇、安裝、連接等。
		DMY 系列直驅馬達使用者操作手冊	MR04UC01-□□□□	詳細說明 DMY 系列直驅馬達的選擇、安裝、連接等。
		DMS 系列直驅馬達使用者操作手冊	MR05UC01-□□□□	詳細說明 DMS 系列直驅馬達的選擇、安裝、連接等。
		DMR 系列直驅馬達使用者操作手冊	MR06UC01-□□□□	詳細說明 DMR 系列直驅馬達的選擇、安裝、連接等。
	力矩馬達	力矩馬達使用者操作手冊	MW99UC01-□□□□	詳細說明力矩馬達的選擇、安裝、連接等。
	伺服馬達	伺服馬達使用者操作手冊	MC03UC01-□□□□	詳細說明伺服馬達的選擇、安裝、連接等。
	IM-1 系列高速主軸馬達	IM-1 系列高速主軸馬達使用者操作手冊	MS01UC01-□□□□	詳細說明 IM-1 系列高速主軸馬達的選擇、安裝等。
線性馬達定位平台	單軸線性馬達定位平台	單軸線性馬達定位平台使用者操作手冊	MM06UC01-□□□□	詳細說明單軸線性馬達定位平台的選擇、安裝、連接等。
致動器	線性致動器	線性致動器使用者操作手冊	MA99UC01-□□□□	詳細說明線性致動器的選擇、安裝、連接等。

目錄

1. 序言

1.1 HMPL 如何運作

HIWIN Motion Programming Language(HMPL)使用類似 C 語言的語法建構獨立 task，供使用者使用。使用者可透過 iA Studio 的 HMPL Editor 撰寫應用所需的運動控制邏輯與程式，經由 HMPL 編譯器將程式轉譯並載入到 HIMC 中。HIMC 的即時程序會於每個通訊週期內，執行固定數量的基礎命令單位。

註：圖示 代表該函式可透過 ASCII TCP 通訊（請參閱第 21 章）在 iA Studio 的 Message Window 或自行安裝的終端機應用程式中使用。

1.2 版本說明

HIMC 控制器搭配軟體版本 iA Studio 3.0（含）以上，支援具 CoE 通訊功能的 HIMC 控制器（產品型號 MC-XX-XX-01-XX），但並不與支援 MoE 通訊的 HIMC 控制器相容（產品型號 MC-XX-XX-00-XX）。搭配 MoE 通訊的 HIMC 控制器，需使用 iA Studio 2.X（含）以下的軟體版本。

iA Stduio 1.3（含）以上所採用的運動變數單位：線性運動（mm）、旋轉運動（deg）、時間（ms）; iA Studio 1.2（含）以下所採用的運動變數單位：線性運動（m）、旋轉運動（rad）、時間（s）。

1.3 法律免責聲明

使用者可因特定用途，採用或修改本使用手冊所提供的任一示例代碼。但是，在不同的應用場景下，無法保證其正確性、有效性及安全性。使用者須為軟體執行的安全性及有效性負全責。

1.4 數據型態

在 HMPL 中，數據型態用於宣告變數或取得函式之回傳值。變數的型態決定其佔據儲存空間的容量及有效值。

表 1.4.1

型態	描述	位元組（Byte）	有效值
char int8_t	8 位元整數	1	-128 ~ 127
unsigned char uint8_t	8 位元正整數	1	0 ~ 255
short int16_t	16 位元整數	2	-32768 ~ 32767
unsigned short uint16_t	16 位元正整數	2	0 ~ 65535
int int32_t	32 位元整數	4	-2147483648 ~ 2147483647
unsigned int uint32_t	32 位元正整數	4	0 ~ 4294967295
long long int64_t	64 位元整數	8	-9223372036854775808 ~ 9223372036854775807
unsigned long long uint64_t	64 位元正整數	8	0 ~ 18446744073709551615
float	32 位元浮點數型（精確到小數點後 6 位）	4	1.17549e-38 ~ 3.40282e+38
double	64 位元浮點數型（精確到小數點後 15 位）	8	2.225074e-308 ~ 1.797693e+308
int* char* double* ...	指標型態，含特定型態變數之儲存位置地址。	8	--
void	具 void 回傳型態的函式不回傳任何值。	--	--
void*	泛指標型態，含任意型態變數之儲存位置地址。	8	--
Timer	用來宣告計數器物件的型態，用於 TON 和 TOF 函式。	8	--

1.5 可視範圍規則

已定義變數或函式之可視範圍，即為其在 HMPL task 中的存在區域。若超出此範圍，則無法使用該變數或該函式。

表 1.5.1

類型	可視範圍	宣告處	描述
全域函式	全域可視範圍	在 **task 0** 內	皆可使用
task 函式	task 可視範圍	不在 **task 0** 內	僅能用於該 task
全域變數	全域可視範圍	在所有的函式外，但在 **task 0** 內	皆可使用
task 變數	task 可視範圍	在所有的函式及 **task 0** 外	僅能用於該函式或 task 中
區域變數	區塊可視範圍	在一個區塊內	僅能用於該區塊
	函式可視範圍	在一個函式內	僅能用於該函式

註：全域變數與 task 變數只在編譯時初始化。

範例 1

```
//  撰寫於 task 0
//  宣告一個全域變數
int global_var = 100;

//  宣告一個全域函式
void GlobalFunction1() {
    Print("%d", global_var);
}
```

範例 2

```
//  撰寫於 task 1
//  宣告一個 task 變數
int task_var = 0;

//  宣告一個 task 函式
void TaskFunction1() {
    //  宣告一個區域變數
    int local_var = task_var;
    for (int i = 0; i < local_var; ++i) {  //  區塊開始
        global_var += i;  //  i 為具有區塊可視範圍的區域變數
    }  //  區塊結束
    global_var += local_var;
}
void main() {
    task_var = 10;
    TaskFunction1();
    GlobalFunction1();  //  輸出為 155
}
```

(此頁有意留白。)

2. HIMC 系統函式

2.1 IsSystemInit

用途

詢問 HIMC 系統是否處於初始（initializing）狀態。

語法

```
int IsSystemInit();
```

參數

無

回傳值

若 HIMC 系統處於初始狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 3.0

2.2 IsSystemOper

用途

詢問 HIMC 系統是否處於運行（operation）狀態。若是，HIMC 與從站間的連線已建立完成。

語法

```
int IsSystemOper();
```

參數

無

回傳值

若 HIMC 系統處於運行狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 3.0

2.3 IsSystemError

用途

詢問 HIMC 系統是否處於錯誤（error）狀態。

語法

```
int IsSystemError();
```

參數

無

回傳值

若 HIMC 系統處於錯誤狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 3.0

2.4 GetECATSt

用途

取得控制器的通訊狀態(EtherCAT State Machine)。

語法

```
int GetECATSt();
```

參數

無

回傳值

控制器通訊狀態，1：Init、2：Pre-Operation、4：Safe-Operation、8：Operation。

需求版本

最低支援版本	iA Studio 3.0

2.5 GetSlvECATSt

用途

取得從站的通訊狀態(EtherCAT State Machine)。

語法

```
int GetSlvECATSt(
    int slv_id
);
```

參數

無

回傳值

從站通訊狀態，1：Init、2：Pre-Operation、4：Safe-Operation、8：Operation。

需求版本

最低支援版本	iA Studio 3.0

2.6 DisableAll

用途

解激磁全部的軸和軸群組。

語法

```
int DisableAll();
```

參數

無

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

此函式會清除軸和軸群組原有的路徑規劃。

需求版本

最低支援版本	iA Studio 0.23

2.7 StopAll

用途

以緊急減速度停止全部的軸和軸群組，並讓軸和軸群組維持在激磁狀態。

語法

```
int StopAll();
```

參數

無

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

此函式會清除軸和軸群組原有的路徑規劃。

需求版本

最低支援版本	iA Studio 0.23

2.8 EStop

用途

緊急停止控制器內部的所有執行程序（包含全部的 HMPL task），並解激磁全部的軸和軸群組。

語法

```
void EStop();
```

參數

無

回傳值

無

需求版本

最低支援版本	iA Studio 0.23

2.9 GetSlaveNum

用途

取得連接至控制器的從站數量。

語法

```
int GetSlaveNum();
```

參數

無

回傳值

連接至控制器的從站數量。

需求版本

最低支援版本	iA Studio 1.1

2.10 GetFirmwareVer

用途

取得控制器的韌體版本。

語法

```
int GetFirmwareVer(
    char *ver_buf
);
```

參數

ver_buf [out]　　指標型態的記憶體，用來儲存回傳的韌體版本字串。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
void main() {
    char ver_buf[30] = {0};
    GetFirmwareVer(ver_buf);
    Print("%s", ver_buf);
}
```

需求版本

最低支援版本	iA Studio 1.4

2.11 ScanNetwork

用途

重新掃描控制器與從站的連線。

語法

```
int ScanNetwork();
```

參數

無

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

2.12 SetHMIScope

用途

開始或停止執行示波器。

語法

```
int SetHMIScope(
    int start
);
```

參數

start [in] 設為 1：開始記錄數據；設為 0：停止記錄數據。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
void main() {
    //  軸激磁
    int axis_id = 0;
    Enable(axis_id);        Till(IsEnabled(axis_id));
    //  開始執行示波器
    SetHMIScope(1);
    //  P2P 運動
    MoveAbs(axis_id, 100);  Till(IsInPos(axis_id));
    MoveAbs(axis_id, 0);    Till(IsInPos(axis_id));
    //  停止執行示波器
    SetHMIScope(0);
}
```

需求版本

最低支援版本	iA Studio 1.1

2.13 Sleep

用途

暫停執行 HMPL task 一段時間。

語法

```
void Sleep(
    int ms
);
```

參數

ms [in]	時間以毫秒為單位。

回傳值

無

需求版本

最低支援版本	iA Studio 0.23

2.14 SendEvent

用途

藉 ID 傳送事件至主機 PC。

語法

```
int SendEvent(
    unsigned short evt_id
);
```

參數

evt_id [in]　　使用者自定義事件 ID。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1) 主機 PC 可藉《HIMC API 參考指南》中的 HIMC_SetHmplEvtCallback 函式來設置回調功能，以取得事件 ID。

(2) 不可太常呼叫此函式 (通常為 1KHz)。若太常呼叫，此函式會被擋掉，直到平均呼叫頻率低於 1KHz。

需求版本

最低支援版本	iA Studio 0.11

2.15 SendMsgEvent

用途

傳送字串訊息至主機 PC。

語法

```
int SendMsgEvent(
    char *format,
    …
);
```

參數

format [in] 指標型態的記憶體，內含欲寫入訊息視窗的正文。

可選擇性地包含以「%指示字」為原型的嵌入式格式指示字。

指示字定義了其型態與相對應的陳述。

指示字	輸出	範例
d 或 i	十進位整數	589
u	十進位正整數	589
x	十六進位正整數	24d
c	字元	M
s	字串	Hello world
f	十進位浮點數（精確到小數點後 6 位）	589.000000
e	科學記號（精確到小數點後 6 位）	5.890000e+02
g	%e 或%f 的最短表現方式	589
%	兩個%相連代表一個%。	%

… [in] 附加參數。

每個參數都包含一個值，用來替換格式字串中的格式指示符。這些參數的數量至少要和格式指示符中指定值的數量一樣多。函式會忽略多出來的參數。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1) 主機 PC 可藉《HIMC API 參考指南》中的 HIMC_SetHmplMsgEvtCallback 函式來設置回調功能，以取得字串訊息。

(2) 不可太常呼叫此函式 (通常為 1KHz)。若太常呼叫，此函式會被擋掉，直到平均呼叫頻率低於 1KHz。

(3) 字串訊息的長度上限為 128 個字元。

範例

```
void main()
{
    SendMsgEvent("variable: %d", 88);
}
```

需求版本

最低支援版本	iA Studio 2.0

2.16 RunScheduler

用途

使呼叫的 task 釋放 CPU 資源，給其他已準備運行的 task。

語法

```
void RunScheduler();
```

參數

無

回傳值

無

需求版本

最低支援版本	iA Studio 0.23

2.17 MutexLock

用途

藉 ID 鎖定互斥鎖物件。

語法

```
void MutexLock(
    int mutex_id
);
```

參數

mutex_id [in]　　互斥鎖物件 ID。有 8 個可用的互斥鎖物件，所以 ID 編號可為 0~7。

回傳值

無

備註

(1) 若當前並無任何 task 鎖定此互斥鎖物件，此互斥鎖物件將透過此函式被鎖定。互斥鎖物件由鎖定它的 task 所擁有，並在鎖定它的 task「呼叫 MutexUnlock 函式」或「停止」前保持鎖定狀態。

(2) 若此互斥鎖物件當前被其他 task 鎖定，此函式會被擋掉，直到互斥鎖物件解鎖。

(3) 若此互斥鎖物件已被呼叫此函式的同一 task 鎖定，task 將停止，並出現運行錯誤的訊息。

需求版本

最低支援版本	iA Studio 0.23

2.18 MutexUnlock

用途

藉 ID 解鎖互斥鎖物件。

語法

```
void MutexUnLock(
    int mutex_id
);
```

參數

mutex_id [in]　　互斥鎖物件 ID。有 8 個可用的互斥鎖物件，所以 ID 編號可為 0~7。

回傳值

無

備註

若呼叫的 task 當前未鎖定互斥鎖物件，則不會有任何事發生。

需求版本

最低支援版本	iA Studio 0.23

2.19 TON

用途

正緣觸發計數器。

當 IN 輸入條件成立，並經過 PT 毫秒後，函式回傳值會由 0 變成非零值。

語法

```
int TON(
    Timer *timer_p,
    int IN,
    int PT
);
```

參數

timer_p [in]	指標型態的記憶體，用來儲存計數器物件。
IN [in]	計數器命令。
PT [in]	執行時間。 參數單位：ms（毫秒）

回傳值

若輸出訊號為 low，將回傳 **int** 型態的值 **0**。若為 high，則回傳**非零值**。

備註

(1) 計數器以 IN 輸入的正緣脈波為始，在經過時間與執行時間相同時停止。IN 輸入的負緣脈波將計數器重置為 0。當執行時間已經過，輸出訊號被設為 1。當輸入命令下降，輸出訊號被重置為 0。

(2) 欲重新啟動計數器，請透過 TimerInit（計數器初始化程序）來初始化計數器物件。

範例

```
void main()
{
    //  初始化計數器
    Timer timer1 = TimerInit;
    Timer timer2 = TimerInit;

    int counter = 0;
    int target_cnt = 1000;
    int cnt_reach_delay_1s = 0;
    int cnt_reach_delay_3s = 0;

    for (;;) {
        Sleep(1);
        counter = counter + 1;
        //  滿足 TON 條件後，經過 1s，cnt_reach_delay_1s 由 0 → 1。
        cnt_reach_delay_1s = TON(&timer1, counter >= target_cnt, 1000);
        //  滿足 TON 條件後，經過 2s，cnt_reach_delay_3s 由 0 → 1。
        cnt_reach_delay_3s = TON(&timer2, cnt_reach_delay_1s, 2000);

        //  存入系統變數，用於 Scope Manager 觀察數值變化。
        system_dtest0 = counter;
        system_dtest1 = target_cnt;
        system_dtest2 = cnt_reach_delay_1s;
        system_dtest3 = cnt_reach_delay_3s;

        //  滿足條件後，經過 0.5s，重置 counter。
        if(cnt_reach_delay_3s){
            Sleep(500);
            counter = 0;
        }
    }
}
```

需求版本

最低支援版本	iA Studio 0.23

2.20 TOF

用途

負緣觸發計數器。

當 IN 輸入條件成立，並經過 PT 毫秒後，函式回傳值會由非零值變成 0。

語法

```
int TOF(
    Timer *timer_p,
    int IN,
    int PT
);
```

參數

timer_p [in]	指標型態的記憶體，用來儲存計數器物件。
IN [in]	計數器命令。
PT [in]	執行時間。 參數單位：ms（毫秒）

回傳值

若輸出訊號為 low，將回傳 **int** 型態的值 **0**。若為 high，則回傳**非零值**。

備註

(1) 計數器以 IN 輸入的負緣脈波為始，在經過時間與執行時間相同時停止。IN 輸入的正緣脈波將計數器重置為 0。當 IN 輸入升至 TRUE，輸出訊號被設為 1。當執行時間已經過，輸出訊號被重置為 0。

(2) 欲重新啟動計數器，請透過 TimerInit（計數器初始化程序）來初始化計數器物件。

範例

```
void main()
{
    //  初始化計數器
    Timer timer1 = TimerInit;
    Timer timer2 = TimerInit;

    int counter = 10000;
    int target_cnt = 8000;
    int cnt_reach_delay_1s = 1;
    int cnt_reach_delay_3s = 1;

    for (;;) {
        Sleep(1);
        counter = counter - 1;
        //  滿足 TOF 條件後，經過 1s，cnt_reach_delay_1s 由 1 → 0。
        cnt_reach_delay_1s = TOF(&timer1, counter >= target_cnt, 1000);
        //  滿足 TOF 條件後，經過 2s，cnt_reach_delay_3s 由 1 → 0。
        cnt_reach_delay_3s = TOF(&timer2, cnt_reach_delay_1s, 2000);

        //  存入系統變數，用於 Scope Manager 觀察數值變化。
        system_dtest0 = counter;
        system_dtest1 = target_cnt;
        system_dtest2 = cnt_reach_delay_1s;
        system_dtest3 = cnt_reach_delay_3s;

        //  滿足條件後，經過 0.5s，重置 counter。
        if(!cnt_reach_delay_3s){
        Sleep(500);
            counter = 10000;
        }
    }
}
```

需求版本

最低支援版本	iA Studio 0.23

3. 字串函式

3.1 概述

在 HMPL 中，字串為字元的一維陣列，以空字元\0（截止位）結尾。
以下宣告與初始化即創建了字串「HIMC」。

```
char str[5] = {‘H’, ‘I’, ‘M’, ‘C’, ‘\0’};
```

因字串以空字元結尾，字串的大小應為正文中的字元數量再加 1。因此，以上範例的大小為 5。

以上陳述有另一種表現方式。使用者不須將空字元放在字串的結尾。HMPL 編譯器會自動將字串的大小設為 5，並在初始化陣列時自動將\0 放在字串的結尾。

```
char str[] = “HIMC”;
```

以上兩個字串的記憶體呈現方式是一樣的，如表 3.1.1。

表 3.1.1

str[0]	str[1]	str[2]	str[3]	str[4]
H	I	M	C	\0

註：在 HMPL 中，字串的最大長度為 512。使用者可藉 HMPL_STR_MAX_LEN 得到此值。

3.2 Print

用途

將格式化的字串寫入訊息視窗。

語法

```
int Print(
    char *format,
    …
);
```

參數

format [in] 指標型態的記憶體，內含欲寫入訊息視窗的正文。

可選擇性地包含以「%指示字」為原型的嵌入式格式指示字。

指示字定義了其型態與相對應的陳述。

指示字	輸出	範例
d 或 i	十進位整數	589
u	十進位正整數	589
x	十六進位正整數	24d
c	字元	M
s	字串	Hello world
f	十進位浮點數（精確到小數點後 6 位）	589.000000
e	科學記號（精確到小數點後 6 位）	5.890000e+02
g	%e 或%f 的最短表現方式	589
%	兩個%相連代表一個%。	%

… [in] 附加參數。

每個參數都包含一個值，用來替換格式字串中的格式指示符。這些參數的數量至少要和格式指示符中指定值的數量一樣多。函式會忽略多出來的參數。

回傳值

字元總數。若發生錯誤，將回傳-1。

範例

```
void main() {

    char str[] = "hello world";
    int var1 = 321;
    double var2 = 1428.57;

    Print("var1: %d, var2: %f, str: %s", var1, var2, str);
    //  var1: 321, var2: 1428.570000, str: hello world

    Print("var2: %e", var2);
    //  var2: 1.428570e+03

    Print("var2: %g", var2);
    //  var2: 1428.57
}
```

需求版本

最低支援版本	iA Studio 0.23

3.3 StringPrint

用途

將格式化的字串寫入記憶體中。

語法

```
int StringPrint(
    char *destination,
    char *format,
    …
);
```

參數

destination [out] 指標型態的記憶體，用來儲存字串結果。

format [in] 指標型態的記憶體，內含欲寫入訊息視窗的正文。
請參閱 3.2 節 Print。

… [in] 附加參數。
每個參數都包含一個值，用來替換格式字串中的格式指示符。這些參數的數量至少要和格式指示符中指定值的數量一樣多。函式會忽略多出來的參數。

回傳值

字元總數。若發生錯誤，將回傳-1。

備註

(1) 此函式與 Print 相似，其不同之處在於輸出字串被寫入記憶體中，而非展示於訊息視窗。

(2) source 字串的截止位也會被複製。
為避免超出範圍，destination 字串的陣列大小應大到足以容納 source 字串（包括截止位）。

範例

```
void main() {

    char dest[80];
    char str[] = "hello world";
    int var1 = 321;
    double var2 = 1428.57;

    StringPrint(dest, "var1: %d, var2: %f, str: %s", var1, var2, str);
    Print("%s", dest);  //  var1: 321, var2: 1428.570000, str: hello world

    StringPrint(dest, "var2: %e", var2);
    Print("%s", dest);  //  var2: 1.428570e+03

    StringPrint(dest, "var2: %g", var2);
    Print("%s", dest);  //  var2: 1428.57
}
```

需求版本

最低支援版本	iA Studio 0.23

3.4 StringLen

用途

取得字串的長度。

語法

```
int StringLen(
    char *str
);
```

參數

str [in] 字串。

回傳值

字串的長度（不包括截止位）。

範例

```
void main() {

    char str[] = "hello world";
    int len = StringLen(str);  //  len = 11
}
```

需求版本

最低支援版本	iA Studio 0.23

3.5 IsStringEqual

用途

檢查兩字串是否相同。

語法

```
int IsStringEqual(
    char *str1,
    char *str2
);
```

參數

str1 [in]　　字串 1。

str2 [in]　　字串 2。

回傳值

若兩字串相同，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

範例

```
void main() {

    char str1[] = "hello world";
    char str2[] = "hello world";
    char str3[] = "hello worldd";

    int is_equal = IsStringEqual(str1, str2);  //  is_equal = 1
    is_equal = IsStringEqual(str1, str3);  //  is_equal = 0
}
```

需求版本

最低支援版本	iA Studio 0.23

3.6 StrFindChar

用途

找出某字元在字串中第一個出現之處。

語法

```
int StrFindChar(
    char *str,
    char character
);
```

參數

str [in] 字串。

character [in] 字元。

回傳值

此字元在字串中第一個出現之處 (offset)。

若字串中無此字元，將回傳-1。

範例

```
void main() {

    char str[] = "hello world";

    int offset = StrFindChar(str, 'h');  //  offset = 0
    offset = StrFindChar(str, 'l');  //  offset = 2
    offset = StrFindChar(str, 'z');  //  offset = -1
}
```

需求版本

最低支援版本	iA Studio 0.23

3.7 StrFindCharEx

用途

找出字串中，字元集合或其補集合的第一個出現之處。

語法

```
int StrFindCharEx(
    char *str,
    char *char_set,
    int   complement_set
);
```

參數

str [in]	字串。
char_set [in]	字元集合。
complement_set [in]	找尋對象。 False (**0**) : 找字元集合。 True (**非零值**) : 找字元集合的補集合。

回傳值

此字元集合中的任一字元或其補集合在字串中第一個出現之處 (offset)。

若未找到任何字元，將回傳-1。

範例

```
void main() {

    char str[] = "hello world";

    int offset = StrFindCharEx(str, "lo ", false);  //  offset = 2
    offset = StrFindCharEx(str, "lo ", true);  //  offset = 0
    offset = StrFindCharEx(str, "zx!c", false);  //  offset = -1
    offset = StrFindCharEx(str, "leh", true);  //  offset = 4
}
```

需求版本

最低支援版本	iA Studio 0.25

3.8 StrFindStr

用途

找出某子字串在字串中第一個出現之處。

語法

```
int StrFindStr(
    char *str1,
    char *str2
);
```

參數

str1 [in] 字串。

str2 [in] 子字串。

回傳值

此子字串在字串中第一個出現之處（offset）。

若字串中無此子字串，將回傳-1。

範例

```
void main() {

    char str[] = "hello world";

    int offset = StrFindStr(str, "hel");  //  offset = 0
    offset = StrFindStr(str, "wor");  //  offset = 6
    offset = StrFindStr(str, "wol");  //  offset = -1
}
```

需求版本

最低支援版本	iA Studio 0.23

3.9 StringCopy

用途

複製字串。

語法

```
int StringCopy(
    char *destination,
    char *source
);
```

參數

destination [out]　指標型態的記憶體，用來儲存字串結果。

source [in]　欲複製的字串。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

source 字串的截止位也會被複製。

為避免超出範圍，destination 字串的陣列大小應大到足以容納 source 字串（包括截止位）。

範例

```
void main() {

    char source[] = "hello world";
    char destination[80];

    StringCopy(destination, source);
    Print("%s", destination);  //  輸出為 hello world
}
```

需求版本

最低支援版本	iA Studio 1.3

3.10 StringCopyEx

用途

複製子字串。

語法

```
int StringCopyEx(
    char *destination,
    char *source,
    int start_pos,
    int copy_len
);
```

參數

destination [out]	指標型態的記憶體，用來儲存字串結果。
source [in]	欲複製的字串。
start_pos [in]	欲複製的子字串偏移量。
copy_len [in]	欲複製的子字串長度。若為-1，則複製字串結尾前的所有字元。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

source 字串的截止位也會被複製。

為避免超出範圍，destination 字串的陣列大小應大到足以容納 source 字串（包括截止位）。

範例

```
void main() {

    char source[] = "hello world";
    char destination[80];

    StringCopyEx(destination, source, 6, 3);
    Print("%s", destination);  //  輸出為 wor

    StringCopyEx(destination, source, 6, -1);
    Print("%s", destination);  //  輸出為 world

    StringCopyEx(destination, source, 0, -1);
    Print("%s", destination);  //  輸出為 hello world
}
```

需求版本

最低支援版本	iA Studio 1.3

3.11 StringCat

用途

連接兩字串。

語法

```
int StringCat(
    char *destination,
    char *source
);
```

參數

destination [out]	指標型態的記憶體，用來儲存字串結果。
source [in]	欲連接的字串。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

將 source 字串附加到 destination 字串上。source 字串的第一個字元會覆寫在 destination 字串的截止位上。為避免超出範圍，destination 字串的陣列大小應大到足以容納 source 字串（包括截止位）。

範例

```
void main() {

    char str[80] = "hello";
    StringCat(str, " world");
    Print("%s", str);  //  輸出為 hello world
}
```

需求版本

最低支援版本	iA Studio 1.3

3.12 StringCatEx

用途

連接子字串。

語法

```
int StringCatEx(
    char *destination,
    char *source,
    int start_pos,
    int copy_len
);
```

參數

destination [out]	指標型態的記憶體，用來儲存字串結果。
source [in]	欲連接的字串。
start_pos [in]	欲複製的子字串偏移量。
copy_len [in]	欲複製的子字串長度。若為-1，則複製字串結尾前的所有字元。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

將 source 字串附加到 destination 字串上。source 字串的第一個字元會覆寫在 destination 字串的截止位上。為避免超出範圍，destination 字串的陣列大小應大到足以容納 source 字串 (包括截止位)。

範例

```
void main() {

    char source[] = "friendsmy ";
    char destination[80] = "hello ";

    StringCatEx(destination, source, 7, -1);
    Print("%s", destination);  //  輸出為 hello my

    //  此時 destination 為 hello my
    StringCatEx(destination, source, 0, 7);
    Print("%s", destination);  //  輸出為 hello my friends
}
```

需求版本

最低支援版本	iA Studio 1.3

3.13 StringToDouble

用途

將字串轉換為 double 型態。

語法

```
double StringToDouble(
    char *str
);
```

參數

str [in] 字串。

回傳值

轉換後的浮點數值。

範例

```
void main() {
    double v = StringToDouble("1.234");  //  v = 1.234
}
```

需求版本

最低支援版本	iA Studio 0.23

3.14 MemoryCopy

用途

將數據從 source 記憶體複製到 destination 記憶體。

語法

```
int MemoryCopy(
    void *destination,
    void *source,
    int byte_num
);
```

參數

destination [out]	指標型態的記憶體，用來儲存數據結果。
source [in]	欲複製的數據。
byte_num [in]	欲複製的位元組數。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
void main() {

    int array1[5] = {1, 2, 3, 4, 5};
    int array2[5] = {11, 22, 33, 44, 55};
    int array3[5] = {345, 456, 567, 678, 789};

    MemoryCopy(array1, array2, sizeof(array2));
    //  此時 array1 裡的值為 11、22、33、44、55

    MemoryCopy(array1, array3, sizeof(int)*3);
    //  此時 array1 裡的值為 345、456、567、44、55

    MemoryCopy(&array1[3], &array3[3], sizeof(int)*2);
    //  此時 array1 裡的值為 345、456、567、678、789
}
```

需求版本

最低支援版本	iA Studio 1.3

3.15 MemorySet

用途

將 destination 記憶體的第一個位元組設為一特定值。

語法

```
int MemorySet(
    void *destination,
    int value,
    int byte_num
);
```

參數

destination [out]	指標型態的記憶體，用來儲存數據結果。
value [in]	欲設置的值。雖以 **int** 型態傳輸，但此函式會將其轉換成 **char** 型態的值存入記憶體。
byte_num [in]	欲設置的位元組數。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
void main() {

    int array1[5] = {1, 2, 3, 4, 5};

    MemorySet(array1, 0, sizeof(array1));
    //  此時 array1 裡的值為 0、0、0、0、0

    MemorySet(array1, 1, sizeof(int));
    //  此時 array1 裡的值為 16843009、0、0、0、0
    //  16843009 = 0x01010101
}
```

需求版本

最低支援版本	iA Studio 1.3

3.16 IsMemoryEqual

用途

檢查兩記憶塊是否相同。

語法

```
int IsMemoryEqual(
    void *memory_ptr1,
    void *memory_ptr2,
    int byte_num
);
```

參數

memory_ptr1 [in]　記憶塊 1。

memory_ptr2 [in]　記憶塊 2。

byte_num [in]　欲設置的位元組數。

回傳值

若兩記憶塊相同，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

範例

```
void main() {

    int array1[5] = {1, 2, 3, 4, 5};
    int array2[5] = {1, 2, 3, 44, 55};
    int is_equal = false;
    is_equal = IsMemoryEqual(array1, array2, sizeof(array2));
    //  is_equal = false

    is_equal = IsMemoryEqual(array1, array2, sizeof(int)*3);
    //  is_equal = true
}
```

需求版本

最低支援版本	iA Studio 0.23

(此頁有意留白。)

4. 數學函式

4.1 sin

用途

取得 x 弧度的正弦值。

語法

```
double sin(
    double x
);
```

參數

x [in]　　一個以弧度表示角度的值。1 弧度等於 180/π角度。

回傳值

x 弧度的正弦值。

範例

```
void main() {
    Print("sine of 30.0 degrees is %f.", sin(30.0 * PI / 180));
    //  30.0 度角的正弦值為 0.5。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.2 cos

用途

取得 x 弧度的餘弦值。

語法

```
double cos(
    double x
);
```

參數

x [in]　　一個以弧度表示角度的值。1 弧度等於 180/π角度。

回傳值

x 弧度的餘弦值。

範例

```
void main() {
    Print("cosine of 60.0 degrees is %f.", cos(60.0 * PI / 180));
    //  60.0 度角的餘弦值為 0.5。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.3 tan

用途

取得 x 弧度的正切值。

語法

```
double tan(
    double x
);
```

參數

x [in]　　一個以弧度表示角度的值。1 弧度等於 180/π角度。

回傳值

x 弧度的正切值。

範例

```
void main() {
    Print("tangent of 45.0 degrees is %f.", tan(45.0 * PI / 180));
    //  45.0 度角的正切值為 1。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.4 asin

用途

取得 x 的反正弦值。在三角函數中，反正弦為正弦的逆運算。

語法

```
double asin(
    double x
);
```

參數

x [in]　　介於[-1, +1]區間的值。

回傳值

x 的反正弦值，介於[-π/2, +π/2]弧度區間。1 弧度等於 180/π角度。

備註

若 x 超出區間，則無法定義回傳值。

範例

```
void main() {
    Print("arc sine of 0.5 is %f degrees", asin(0.5) * 180.0 / PI);
    //  0.5 的反正弦值為 30.0 度角。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.5 acos

用途

取得 x 的反餘弦值。在三角函數中，反餘弦為餘弦的逆運算。

語法

```
double acos(
    double x
);
```

參數

x [in]　　介於[-1, +1]區間的值。

回傳值

x 的反餘弦值，介於[0, π]弧度區間。1 弧度等於 180/π角度。

備註

若 x 超出區間，則無法定義回傳值。

範例

```
void main() {
    Print("arc cosine of 0.5 is %f degrees", acos(0.5) * 180.0 / PI);
    //  0.5 的反餘弦值為 60.0 度角。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.6 atan

用途

取得 x 的反正切值。在三角函數中，反正切為正切的逆運算。

語法

```
double atan(
    double x
);
```

參數

x [in]

回傳值

x 的反正切值，介於[-π/2, +π/2]弧度區間。1 弧度等於 180/π角度。

備註

由於符號模糊性，此函式無法透過其正切值來完全確定角度會落在哪個象限中。有關採用小數參數的替代方法，請參閱 4.7 節 atan2。

範例

```
void main() {
    Print("arc tangent of 1.0 is %f degrees", atan(1.0) * 180.0 / PI);
    //  1.0 的反正切值為 45.0 度角。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.7 atan2

用途

取得 y/x 的反正切值。

語法

```
double atan2(
    double y,
    double x
);
```

參數

y [in]　　表示 Y 座標比例的值。

x [in]　　表示 X 座標比例的值。

回傳值

y/x 的反正弦值，介於[-π, +π]弧度區間。1 弧度等於 180/π角度。

範例

```
void main() {
    Print("arc tangent for (x=-10, y=10) is %f degrees",
            atan2(10, -10) * 180.0 / PI);
    //  ( x=-10, y=10 ) 的反正切值為 135 度角。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.8 abs

用途

取得整數 x 的絕對值：|x|。

語法

```
int abs(
    int x
);
```

參數

x [in]　　一個整數。

回傳值

整數 x 的絕對值。

範例

```
void main() {
    Print("absolute value of -3591 is %d.", abs(-3591));
    //  -3591 的絕對值為 3591。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.9 fabs

用途

取得雙精度浮點數 x 的絕對值：|x|。

語法

```
double fabs(
    double x
);
```

參數

x [in]　　一個雙精度浮點數。

回傳值

雙精度浮點數 x 的絕對值。

範例

```
void main() {
    Print("absolute value of -35.91 is %f.", fabs(-35.91));
    //  -35.91 的絕對值為 35.91。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.10 ceil

用途

將 x 無條件進位為整數。

語法

```
double ceil(
    double x
);
```

參數

x [in]

回傳值

不小於 x 的最小整數。

範例

```
void main() {
    Print("ceil of 2.3 is %g", ceil(2.3));  //  2.3 無條件進位為 3.0。
    Print("ceil of 3.8 is %g", ceil(3.8));  //  3.8 無條件進位為 4.0。
    Print("ceil of -2.3 is %g", ceil(-2.3));  //  -2.3 無條件進位為-2.0。
    Print("ceil of -3.8 is %g", ceil(-3.8));  //  -3.8 無條件進位為-3.0。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.11 floor

用途

將 x 無條件捨去為整數。

語法

```
double floor(
    double x
);
```

參數

x [in]

回傳值

不大於 x 的最大整數。

範例

```
void main() {
    Print("floor of 2.3 is %g", floor(2.3));  //  2.3 無條件捨去為 2.0。
    Print("floor of 3.8 is %g", floor(3.8));  //  3.8 無條件捨去為 3.0。
    Print("floor of -2.3 is %g", floor(-2.3));  //  -2.3 無條件捨去為-3.0。
    Print("floor of -3.8 is %g", floor(-3.8));  //  -3.8 無條件捨去為-4.0。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.12 ldexp

用途

取得 x 乘以 2 的 y 次方的值：x * 2^y。

語法

```
double ldexp(
    double x,
    int    y
);
```

參數

x [in]

y [in]　　一個整數。

回傳值

x * 2^y。

若結果太大，將回傳最大可表示的 double 型態值。

範例

```
void main() {
    Print("0.95 * 2^4 = %f", ldexp(0.95, 4 ));  //  0.95 * 2^4 = 15.20
}
```

需求版本

最低支援版本	iA Studio 0.23

4.13 exp

用途

取得 e 的 x 次方的值：e^x。

e 為自然對數的基數，其值大約等於 2.71828。

語法

```
double exp(
    double x
);
```

參數

x [in]

回傳值

e^x。

若結果太大，將回傳最大可表示的 double 型態值。

範例

```
void main() {
    Print("The exponential value of 5.0 is %f.", exp(5.0));
    //  e 的 5.0 次方為 148.413159。
}
```

需求版本

最低支援版本	iA Studio 0.23

4.14 pow

用途

取得 x 的 y 次方的值：x^y。

語法

```
double pow(
    double x,
    double y
);
```

參數

x [in]

y [in]

回傳值

x^y。

若結果太大，將回傳最大可表示的 double 型態值。

備註

(1) 若 x 為有限負數、y 為有限非整數，則無法定義回傳值。

(2) 若 x 與 y 皆為 0，則無法定義回傳值。

(3) 若 x 為 0、y 為負數，則無法定義回傳值。

範例

```
void main() {
    Print("7.0 ^ 3.0 = %f", pow(7.0, 3.0));  //  7.0 ^ 3.0 = 343.0
    Print("4.73 ^ 12.0 = %f", pow(4.73, 12.0));  //  4.73 ^ 12.0 = 125410439.217423
    Print("32.01 ^ 1.54 = %f", pow(32.01, 1.54));  //  32.01 ^ 1.54 = 208.036691
}
```

需求版本

最低支援版本	iA Studio 0.23

4.15 log

用途

取得 x 的自然對數（基數為 e）。

語法

```
double log(
    double x
);
```

參數

x [in]

回傳值

x 的自然對數（基數為 e）。

備註

(1) 若 x 為負數或 0，則無法定義回傳值。

(2) 有關常用的對數（基數為 10），請參閱 4.16 節 log10。

範例

```
void main() {
    Print("log(5.5) = %f", log(5.5));  //  log(5.5) = 1.704748
}
```

需求版本

最低支援版本	iA Studio 0.23

4.16 log10

用途

取得 x 的對數（基數為 10）。

語法

```
double log10(
    double x
);
```

參數

x [in]

回傳值

x 的對數（基數為 10）。

備註

若 x 為負數或 0，則無法定義回傳值。

範例

```
void main() {
    Print("log10(1000.0) = %f", log10(1000.0));  //  log10(1000.0) = 3.0
}
```

需求版本

最低支援版本	iA Studio 0.23

4.17 sqrt

用途

取得 x 的平方根。

語法

```
double sqrt(
    double x
);
```

參數

x [in]

回傳值

x 的平方根。

備註

若 x 為負數，則無法定義回傳值。

範例

```
void main() {
    Print("sqrt(1024.0) = %f", sqrt(1024.0));  //  sqrt(1024.0) = 32.0
}
```

需求版本

最低支援版本	iA Studio 0.23

4.18 cbrt

用途

取得 x 的立方根。

語法

```
double cbrt(
    double x
);
```

參數

x [in]

回傳值

x 的立方根。

範例

```
void main() {
    Print("cbrt (27.0) = %f", cbrt(27.0));  //  cbrt (27.0) = 3.0
}
```

需求版本

最低支援版本	iA Studio 0.23

4.19 hypot

用途

取得直角三角形 (直角邊為 x 和 y) 的斜邊。

語法

```
double hypot(
    double x,
    double y
);
```

參數

x [in]　　直角三角形的其中一邊。

y [in]　　直角三角形的另一邊。

回傳值

(x^2+y^2) 的平方根。

若結果太大，將回傳最大可表示的 double 型態值。

範例

```
void main() {
    Print("hypot(3.0, 4.0) = %f.", hypot(3.0, 4.0));
    //  hypot(3.0, 4.0) = 5
}
```

需求版本

最低支援版本	iA Studio 0.23

5. 軸函式

5.1 概述

HIMC 提供單軸運動命令，包含點到點 (P2P)、吋動 (JOG) 與同步運動命令，使用者可依應用需求使用相關的運動函式功能。圖 5.1.1 為 HIMC 軸運動控制流程圖，運動命令會經過內建的軌跡規劃器 (PG，Profile Generator)產生軸的參考位置命令(Reference Position)；而輸出的參考位置命令疊加上軸的誤差補償值，即產生給驅動器的位置輸出命令 (Position Output)。

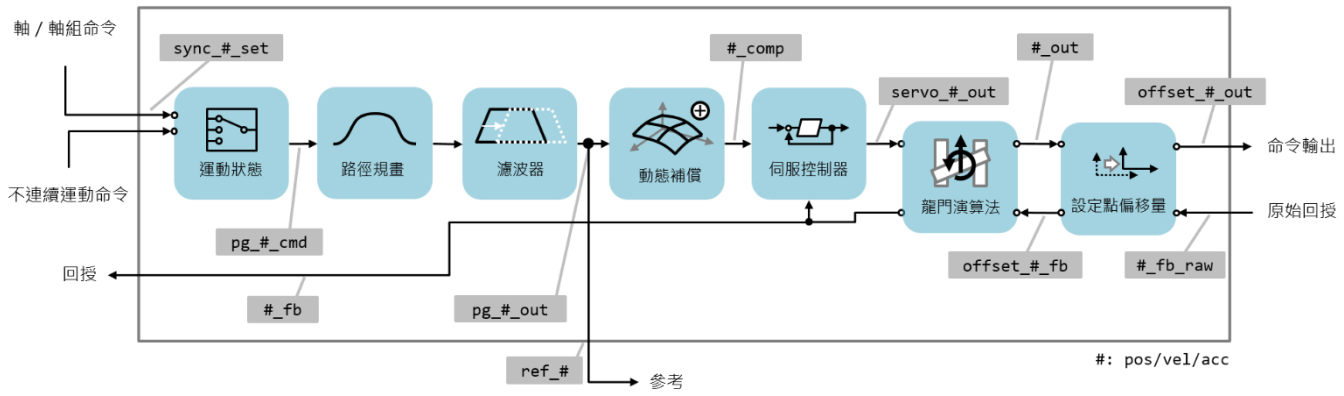


圖 5.1.1

HIMC 的軌跡規劃器內建 S-Curve 速度規劃，如圖 5.1.2 所示。使用者可設定軌跡規劃器的最大速度、最大加速度、最大減速度與平滑時間。

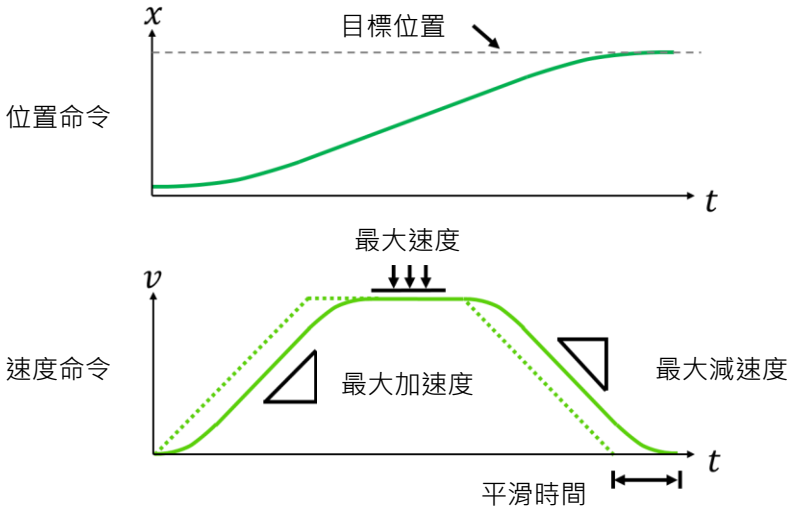


圖 5.1.2

如圖 5.1.3 所示，加入平滑時間會造成軸速度命令延遲，但可以有效降低高加減速所產生的急跳度 (Jerk)，以增加系統的穩定性。急跳度、最大加速度與平滑時間的關係如下：

急跳度 = 最大加速度/平滑時間 Ts

總加速度時間則可由下列關係式得到：

總加速度時間 T = T-Curve 加速度時間 Ta + 平滑時間 Ts

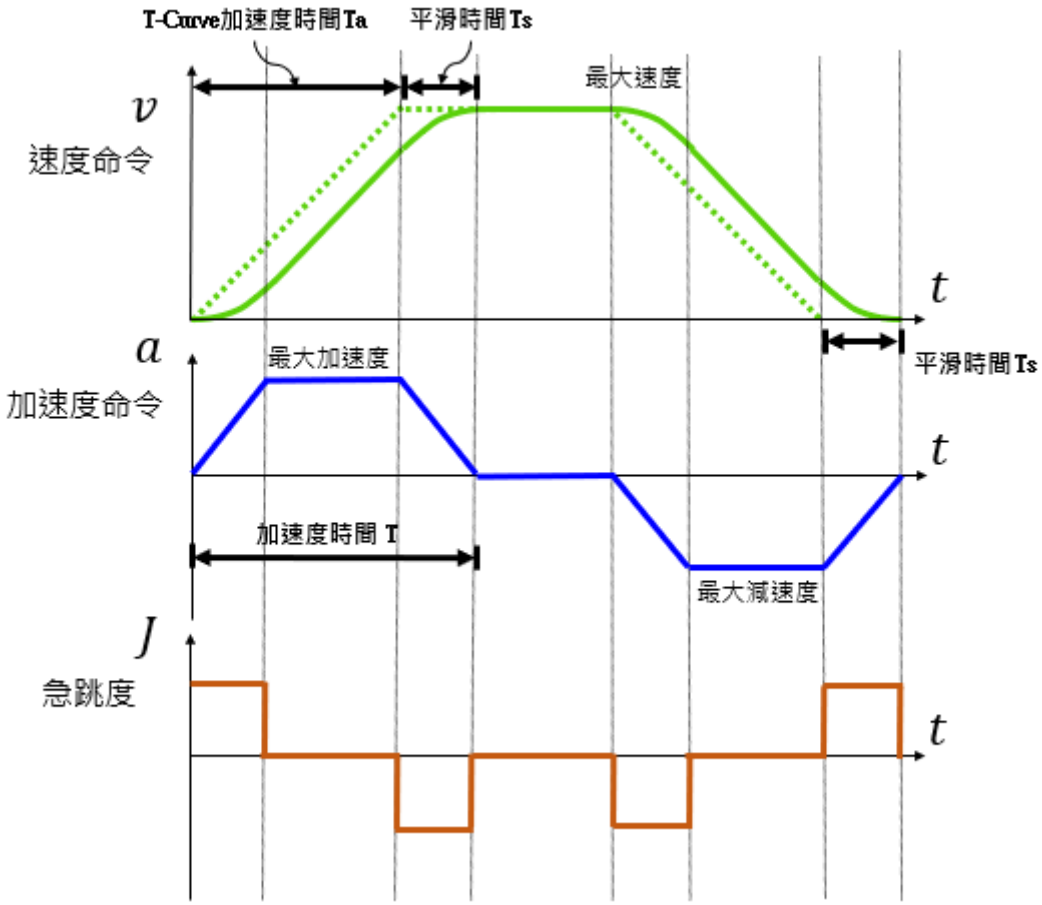


圖 5.1.3

此外，HIMC 軸運動命令支援動態地變更目標位置與速度規劃 (On the Fly Modification)。使用者可以在軸運動的過程中，改變軸的目標位置命令、最大速度與最大加 / 減速度；HIMC 的軌跡規劃器會依據新的命令與軸運動參數，移動到新的目標位置。

軸的運動狀態分成移動中（Moving）與是否到位（In-Position），如圖 5.1.4 所示。在區域 I 中持續送出軸位置規劃命令，並在進入區域 II 前結束；控制器會依據所設定的目標框半徑（Target Radius）與反彈跳時間（Debounce Time），判斷軸是否已經到位。

軸位置回授若在軸位置命令的目標框內，經過反彈跳時間後，會被視為軸位置已經到位，此時控制器內部的軸到位狀態成立；若在反彈跳時間內的任一時刻，軸位置回授超出位置命令的目標框，則整定時間的計算將被重設，等待下一次軸位置回授進入目標框內後，才會重新計算是否滿足反彈跳時間的到位條件。

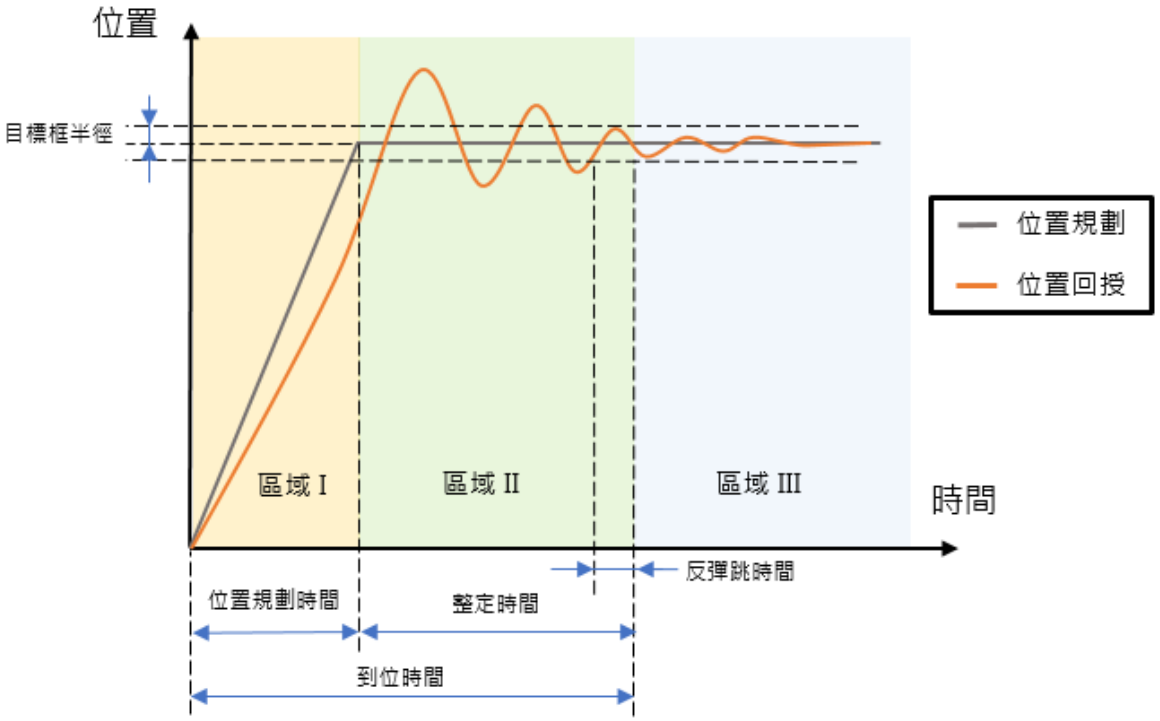


圖 5.1.4

參考圖 5.1.4，軸運動狀態說明如下：

1. 區域 I：軸運動規劃中（Moving），尚未到位（Not In-Position）。
2. 區域 II：軸運動規劃停止（Not Moving），但尚未到位（Not In-Position）。
3. 區域 III：軸運動規劃停止（Not Moving），已到位（In-Position）。

而若軸在同步運動狀態時，軸運動會交由軸群組或主軸來產生規劃的運動軌跡，軸本身不規劃運動軌跡，僅追隨由軸群組或主軸規劃的位置命令。

5.1.1 軸變數

軸變數分成運動命令變數、運動規劃變數與狀態變數，使用者可利用 iA Studio 的 Scope Manager（請參閱《iA Studio 軟體使用手冊》4.8 節）選擇欲觀測的變數。詳細說明如表 5.1.1.1 至表 5.1.1.5。

表 5.1.1.1 軸運動命令變數

名稱	變數	單位	描述
Sync Position Setpoint	sync_pos_set	毫米 或 角度	同步位置設定點。當軸處於同步運動模式（如軸群組、凸輪或齒輪傳動操作）時，此點為其目標位置。
Position Command	pg_pos_cmd	毫米 或 角度	軌跡規劃位置命令。當軸處於不連續運動模式（點對點）時，此點為其目標位置。
Reference Position	ref_pos	毫米 或 角度	參考位置。根據預先定義好的運動軌跡，經軌跡規劃器生產而成的位置設定點。
Reference Velocity	ref_vel	毫米/秒 或 角度/秒	參考速度。根據預先定義好的運動軌跡，經軌跡規劃器生產而成的速度設定點。
Reference Acceleration	ref_acc	毫米/秒 ^{2} 或 角度/秒 ^{2}	參考加速度。根據預先定義好的運動軌跡，經軌跡規劃器生產而成的加速度設定點。
Position Compensation	comp_pos_set	毫米 或 角度	位置誤差補償值。此為動態誤差補償功能的補償輸出。若不使用此功能，其值為 0 。
Compensated Position	pos_comp	毫米 或 角度	補償後的位置命令值。此為參考位置與位置誤差補償值的加總。
Position Offset	pos_offset	毫米 或 角度	位置偏移量。預設值為 0。若使用者在馬達未移動時設一個新的軸位置，其值則不是 0。
Position Output	pos_out	毫米 或 角度	位置輸出。此為無位置偏移量的軸位置命令。
Velocity Output	vel_out	毫米/秒 或 角度/秒	速度輸出。此為無速度偏移量的軸速度命令。
Acceleration Output	acc_out	毫米/秒 ^{2} 或 角度/秒 ^{2}	加速度輸出。此為無加速度偏移量的軸加速度命令。
Offsetted Position Output	offset_pos_out	毫米 或 角度	具位置偏移量的位置輸出。此為最終計算而成軸位置命令。該值將轉換為單位 count 並傳送到相應的從站。
Raw Position Feedback	pos_fb_raw	毫米 或 角度	原始位置回授。由從站讀取該站收到的編碼器回授。

名稱	變數	單位	描述
Offsetted Position Feedback	offset_pos_fb	毫米 或 角度	被偏移的位置回授。此為具位置偏移量的位置回授。
Position Feedback	pos_fb	毫米 或 角度	位置回授。位於軸座標系統中。
Velocity Feedback	vel_fb	毫米/秒 或 角度/秒	速度回授。位於軸座標系統中。
Position Error	pos_err	毫米 或 角度	位置誤差。此為位置輸出與原始位置回授間的差值。
Velocity Error	vel_err	毫米/秒 或 角度/秒	速度誤差。此為速度輸出與原始速度回授間的差值。
Move Time	movetime	毫秒	路徑規劃時間。
Settling Time	settlingtime	毫秒	整定時間。

表 5.1.1.2 軸運動規劃變數

名稱	變數	單位	描述
Max. Profile Velocity	max_vel	毫米/秒 或 角度/秒	最大速度。不一定會達到。
Max. Profile Acceleration	max_acc	毫米/秒 ^{2} 或 角度/秒 ^{2}	最大加速度。不一定會達到。
Profile Deceleration	max_dec	毫米/秒 ^{2} 或 角度/秒 ^{2}	最大減速度。不一定會達到。
Smooth Time	sm_factor	毫秒	平滑時間。輸入範圍為 0 ~ 500。增加該值可減少運動期間的機械振動，但會影響總運動時間。

表 5.1.1.3 軸狀態變數

名稱	變數	單位	描述
Fault Status	fault_status	無	軸錯誤狀態，位元定義請參閱表 5.1.1.4。
Motion Status	motion_status	無	軸運動狀態，位元定義請參閱表 5.1.1.5。

表 5.1.1.4 軸錯誤狀態位元定義

位元	名稱	描述	預設反應
0	Error Stop	軸處於 error stop 狀態。	無。
1	Drive fault	從站驅動器的錯誤。	控制器將軸解激磁。
2	Position error	位置誤差超過保護範圍。	控制器將軸解激磁。
3	Hardware right limit	觸發軸的硬體右極限。	控制器停止軸的運動。
4	Hardware left limit	觸發軸的硬體左極限。	控制器停止軸的運動。
5	Software right limit	觸發軸的軟體右極限。	控制器停止軸的運動。
6	Software left limit	觸發軸的軟體左極限。	控制器停止軸的運動。

表 5.1.1.5 軸運動狀態位元定義

位元	名稱	描述	備註
0	Enabled	軸激磁狀態。	無。
1	Moving	軸移動中。	請參閱 5.1 節。
2	In Position	軸到位。	請參閱 5.1 節。
3	Synchronous	軸在同步狀態。	軸在軸群組中或為電子齒輪從軸。
4	Group	軸在軸群組中。	請參閱 8.1 節。
5	Gantry	軸為龍門軸。	請參閱 7.1 節。
6	Input Shape	開啟 Input Shape 濾波器。	請參閱 15.1 節。
7	VSF	開啟 VSF 濾波器。	請參閱 15.1 節。
8	Gear	軸為電子齒輪從軸。	請參閱 6.1 節。
9	Cam	軸為電子凸輪從軸。	目前不支援。
10	Accelerating	軸加速中。	請參閱 5.1 節。

5.2 軸運動控制

5.2.1 Enable

用途

激磁軸。

語法

```
int Enable(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將物件 0x6040(Control word)配置為 PDO。

需求版本

最低支援版本	iA Studio 0.23

5.2.2 Disable

用途

解激磁軸。

語法

```
int Disable(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1) 此函式會清除軸原有的路徑規劃。

(2) 使用此函式需將物件 0x6040(Control word)配置為 PDO。

需求版本

最低支援版本	iA Studio 0.23

5.2.3 Reset

用途

當軸處於 ErrorStop 狀態時，清除錯誤。

語法

```
int Reset(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1) 當軸處於 error stop 狀態時，執行此函式。

(2) 使用此函式需將物件 0x6040(Control word)配置為 PDO。

需求版本

最低支援版本	iA Studio 0.23

5.2.4 MoveAbs

用途

將軸移動至絕對目標位置。

語法

```
int MoveAbs(
    int    axis_id,
    double pos
);
```

參數

axis_id [in]	軸編號。
pos [in]	絕對目標位置的值。 參數單位：mm（毫米）或 deg（角度）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將對應的命令配置為 PDO，例如 CSP 模式為 0x607A(Target position)。

需求版本

最低支援版本	iA Studio 0.23

5.2.5 MoveRel

用途

將軸移動相對距離。

語法

```
int MoveRel(
    int    axis_id,
    double rel_dist
);
```

參數

axis_id [in]	軸編號。
rel_dist [in]	相對距離的值。 參數單位：mm（毫米）或 deg（角度）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將對應的命令配置為 PDO，例如 CSP 模式為 0x607A(Target position)。

需求版本

最低支援版本	iA Studio 0.23

5.2.6 MoveVel

用途

以特定速度持續移動。

語法

```
int MoveVel(
    int    axis_id,
    double vel
);
```

參數

axis_id [in]	軸編號。
vel [in]	移動速度的值。 數值的正負值表示運動方向。 參數單位：mm/s（毫米/秒）或 deg/s（角度/秒）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將對應的命令配置為 PDO，例如 CSP 模式為 0x607A(Target position)。

需求版本

最低支援版本	iA Studio 0.23

5.2.7 MoveTrq

用途

以特定轉矩持續移動。

語法

```
int MoveTrq(
    int    axis_id,
    double torque_cmd
);
```

參數

axis_id [in]	軸編號。
torque_cmd [in]	轉矩命令。 參數單位：N-m (牛頓-米)

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1) 此函式僅適用於 Profile Torque 模式。

(2) 若轉矩命令大於馬達的連續轉矩，馬達會以連續轉矩的值移動。

(3) 使用此函式需將物件 0x6071(Target torque)配置為 PDO，且需設定馬達的力矩常數(force constant)。

需求版本

最低支援版本	iA Studio 2.0

5.2.8 MovePVT

用途

依給定的位置（P）、速度（V）與時間（T），將軸移動至指定位置。

語法

```
int MovePVT(
    int axis_id,
    double *time,
    double *pos,
    double *vel,
    int num_pt
);
```

參數

axis_id [in] 軸編號。

time [in] 指向使用者給定的時間陣列的指標。
參數單位：ms（毫秒）

pos [in] 指向使用者給定的位置陣列的指標。
參數單位：mm（毫米）或 deg（角度）

vel [in] 指向使用者給定的速度陣列的指標。
參數單位：mm/s（毫米/秒）或 deg/s（角度/秒）

num_pt [in] PVT 運動點的數量，其最大值為 50。
時間、位置與速度陣列的長度皆須與此參數一致。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將對應的命令配置為 PDO，例如 CSP 模式為 0x607A(Target position)。

範例

```
void main() {
    int axis_id = 0;
    //  PVT 運動的位置、速度與時間陣列
    double point[6] = {0.0, 153.333, 42.123, 161.21, 177.0, 83.333};
    double velocity[6] = {0.0, 1000.0, 800.0, 1660.0, 450.0, 0.0};
    double time[6] = {0.0, 2000.0, 3000.0, 4000.0, 6000.0, 11000.0};
    Enable(axis_id);
    Till(IsEnabled(axis_id));
    //  啟動 PVT 運動
    MovePVT(axis_id, time, point, velocity, 6);
}
```

需求版本

最低支援版本	iA Studio 2.0

5.2.9 Stop

用途

停止軸的運動。

語法

```
int Stop(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

此函式會清除軸原有的路徑規劃。

需求版本

最低支援版本	iA Studio 0.23

5.2.10 Halt

用途

暫停軸的運動，軸運動速度被設定為 0。

語法

```
int Halt(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

在 PP/PV/PT 模式下，使用此函式需將物件 0x6040(Control word)配置為 PDO。

需求版本

最低支援版本	iA Studio 1.3

5.2.11 Resume

用途

由軸暫停的狀態中恢復軸的運動。

語法

```
int Resume(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

在 PP/PV/PT 模式下，使用此函式需將物件 0x6040(Control word)配置為 PDO。

需求版本

最低支援版本	iA Studio 1.3

5.3 軸設定

5.3.1 GetMaxVel

用途

取得軸的最大速度。

語法

```
double GetMaxVel(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的最大速度。

單位：mm/s（毫米/秒）或 deg/s（角度/秒）

需求版本

最低支援版本	iA Studio 0.23

5.3.2 SetVel

用途

設置軸的最大速度。

語法

```
int SetVel(
    int    axis_id,
    double vel
);
```

參數

axis_id [in]	軸編號。
vel [in]	軸的新最大速度。 參數單位：mm/s（毫米/秒）或 deg/s（角度/秒） 輸入範圍：非零正值

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.23

5.3.3 GetMaxAcc

用途

取得軸的最大加速度。

語法

```
double GetMaxAcc(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的最大加速度。

單位：mm/s^2（毫米/秒 ^{2}）或 deg/s^2（角度/秒 ^{2}）

需求版本

最低支援版本	iA Studio 0.23

5.3.4 SetAcc

用途

設置軸的最大加速度。

語法

```
int SetAcc(
    int    axis_id,
    double acc
);
```

參數

axis_id [in]	軸編號。
acc [in]	軸的新最大加速度。 參數單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2） 輸入範圍：非零正值

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.23

5.3.5 SetAccTime

用途

設置軸的加速度時間。

語法

```
int SetAccTime(
    int    axis_id,
    double acc_time
);
```

參數

axis_id [in]	軸編號。
acc_time [in]	軸的加速度時間。 參數單位：ms（毫秒） 輸入範圍：非零正值

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

5.3.6 GetMaxDec

用途

取得軸的最大減速度。

語法

```
double GetMaxDec(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

軸的最大減速度。

單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2）

需求版本

最低支援版本	iA Studio 0.23

5.3.7 SetDec

用途

設置軸的最大減速度。

語法

```
int SetDec(
    int    axis_id,
    double dec
);
```

參數

axis_id [in]	軸編號。
dec [in]	軸的新最大減速度。 參數單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2） 輸入範圍：非零正值

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.23

5.3.8 SetDecTime

用途

設置軸的減速度時間。

語法

```
int SetDecTime(
    int    axis_id,
    double dec_time
);
```

參數

axis_id [in]	軸編號。
dec_time [in]	軸的減速度時間。 參數單位：ms（毫秒） 輸入範圍：非零正值

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

5.3.9 GetKillDec

用途

取得軸的緊急減速度。

語法

```
double GetKillDec(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的緊急減速度。

單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2）。

需求版本

最低支援版本	iA Studio 1.3

5.3.10 SetKillDec

用途

設置軸的緊急減速度。

語法

```
int SetKillDec(
    int    axis_id,
    double kill_dec
);
```

參數

axis_id [in]	軸編號。
kill_dec [in]	軸的新緊急減速度。 參數單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2） 輸入範圍：非零正值

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

5.3.11 GetSWRL

用途

取得軸的軟體右極限位置。

語法

```
double GetSWRL(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的軟體右極限位置。

單位：mm（毫米）或 deg（角度）

需求版本

最低支援版本	iA Studio 1.1

5.3.12 SetSWRL

用途

設置軸的軟體右極限位置。

語法

```
int SetSWRL(
    int    axis_id,
    double right_limit_pos
);
```

參數

axis_id [in]	軸編號。
right_limit_pos [in]	軸的新軟體右極限位置。 參數單位：mm（毫米）或 deg（角度）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.1

5.3.13 GetSWLL

用途

取得軸的軟體左極限位置。

語法

```
double GetSWLL(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的軟體左極限位置。

單位：mm（毫米）或 deg（角度）

需求版本

最低支援版本	iA Studio 1.1

5.3.14 SetSWLL

用途

設置軸的軟體左極限位置。

語法

```
int SetSWLL(
    int    axis_id,
    double left_limit_pos
);
```

參數

axis_id [in]	軸編號。
left_limit_pos [in]	軸的新軟體左極限位置。 參數單位：mm（毫米）或 deg（角度）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.1

5.3.15 GetSMTime

用途

取得軸的平滑時間。

語法

```
double GetSMTime(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的平滑時間。

單位：ms (毫秒)

需求版本

最低支援版本	iA Studio 0.23

5.3.16 SetSMTime

用途

設置軸的平滑時間。

語法

```
int SetSMTime(
    int    axis_id,
    double smooth_time
);
```

參數

axis_id [in]	軸編號。
smooth_time [in]	軸的新平滑時間。 參數單位：ms（毫秒） 輸入範圍：0 ~ 500

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當軸正在移動時，此函式不適用。

需求版本

最低支援版本	iA Studio 0.23

5.3.17 GetMoveTime

用途

取得軸的路徑規劃時間。

語法

```
double GetMoveTime(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的路徑規劃時間。

單位：ms (毫秒)

需求版本

最低支援版本	iA Studio 0.24

5.3.18 GetSettlingTime

用途

取得軸的整定時間。

語法

```
double GetSettlingTime(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的整定時間。

單位：ms (毫秒)

需求版本

最低支援版本	iA Studio 0.24

5.3.19 SetPos

用途

設定軸位置，改變原點偏移量。

語法

```
int SetPos(
    int    axis_id,
    double pos
);
```

參數

axis_id [in]	軸編號。
pos [in]	軸目前位置的值。 參數單位：mm（毫米）或 deg（角度）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當軸處於同步狀態、加入軸群組或處於錯誤狀態時，此功能不適用。

需求版本

最低支援版本	iA Studio 0.23

5.3.20 GetPosFb

用途

取得軸的位置回授。

語法

```
double GetPosFb(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的位置回授。

單位：mm（毫米）或 deg（角度）

備註

使用此函式需將物件 0x6064(Position actual value)配置為 PDO。

需求版本

最低支援版本	iA Studio 0.23

5.3.21 GetPosOffset

用途

取得軸的位置偏移量。

語法

```
double GetPosOffset(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的位置偏移量。

單位：mm（毫米）或 deg（角度）

需求版本

最低支援版本	iA Studio 0.23

5.3.22 GetPosErr

用途

取得軸的位置誤差。

語法

```
double GetPosErr(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的位置誤差。

單位：mm（毫米）或 deg（角度）

需求版本

最低支援版本	iA Studio 0.23

5.3.23 GetVelFb

用途

取得軸的速度回授。

語法

```
double GetVelFb(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的速度回授。

單位：mm/s（毫米/秒）或 deg/s（角度/秒）

需求版本

最低支援版本	iA Studio 1.4

5.3.24 GetVelErr

用途

取得軸的速度誤差。

語法

```
double GetVelErr(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

軸的速度誤差。

單位：mm/s（毫米/秒）或 deg/s（角度/秒）

需求版本

最低支援版本	iA Studio 1.4

5.3.25 GetCurrFb

用途

取得軸的電流回授。

語法

```
int GetCurrFb(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的電流回授。

參數單位：0.1%額定電流。

備註

使用此函式需將物件 0x6077(Torque actual value)配置為 PDO。

需求版本

最低支援版本	iA Studio 3.0

5.3.26 GetRefPos

用途

取得軸的參考位置。

語法

```
double GetRefPos(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的參考位置。

單位：mm（毫米）或 deg（角度）

需求版本

最低支援版本	iA Studio 0.23

5.3.27 GetRefVel

用途

取得軸的參考速度。

語法

```
double GetRefVel(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的參考速度。

單位：mm/s（毫米/秒）或 deg/s（角度/秒）

需求版本

最低支援版本	iA Studio 0.23

5.3.28 GetRefAcc

用途

取得軸的參考加速度。

語法

```
double GetRefAcc(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的參考加速度。

單位：mm/s^2（毫米/秒 ^{2}）或 deg/s^2（角度/秒 ^{2}）

需求版本

最低支援版本	iA Studio 0.23

5.3.29 GetPosOut

用途

取得由控制器送至驅動器的位置命令輸出。

語法

```
double GetPosOut(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的位置命令輸出。

單位：mm（毫米）或 deg（角度）

需求版本

最低支援版本	iA Studio 0.23

5.3.30 GetVelOut

用途

取得由控制器送至驅動器的速度命令輸出。

語法

```
double GetVelOut(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的速度命令輸出。

單位：mm/s（毫米/秒）或 deg/s（角度/秒）

需求版本

最低支援版本	iA Studio 0.23

5.3.31 GetAccOut

用途

取得由控制器送至驅動器的加速度命令輸出。

語法

```
double GetAccOut(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的加速度命令輸出。

單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2）

需求版本

最低支援版本	iA Studio 0.23

5.3.32 IgnoreHWL

用途

忽略硬體極限的警告通知。

語法

```
int IgnoreHWL(
    int axis_id,
    int cmd
);
```

參數

axis_id [in]　　軸編號。

cmd [in]　　設為 1：忽略通知；設為 0：恢復通知（預設值）。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.23

5.3.33 IgnoreSWL

用途

忽略軟體極限的警告通知。

語法

```
int IgnoreSWL(
    int axis_id,
    int cmd
);
```

參數

axis_id [in]　　軸編號。

cmd [in]　　設為 1：忽略通知；設為 0：恢復通知（預設值）。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.23

5.3.34 IgnorePE

用途

忽略位置誤差極限的警告通知。

語法

```
int IgnorePE(
    int axis_id,
    int cmd
);
```

參數

axis_id [in] 軸編號。

cmd [in] 設為 1：忽略通知；設為 0：恢復通知（預設值）。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

5.3.35 GetAxisNum

用途

取得連接至控制器的軸數量。

語法

```
int GetAxisNum();
```

參數

無

回傳值

連接至控制器的軸數量。

需求版本

最低支援版本	iA Studio 0.23

5.3.36 SetVelScale

用途

設置軸運動的速度百分比。

語法

```
int SetVelScale(
    int    axis_id,
    double vel_scale
);
```

參數

axis_id [in]	軸編號。
vel_scale [in]	軸運動的新速度百分比。 輸入範圍：0 ~ 100

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.4

5.3.37 GetVelScale

用途

取得軸運動的速度百分比。

語法

```
double GetVelScale(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸運動的速度百分比，數值範圍為 0 ~ 100。

需求版本

最低支援版本	iA Studio 1.4

5.3.38 SetRollover

用途

設定軸在單圈模式下的位置上限值。

語法

```
int SetRollover(
    int    axis_id,
    double rollover_val
);
```

參數

axis_id [in]	軸編號。
rollover_val [in]	軸在單圈模式下的位置上限值。 參數單位：mm（毫米）或 deg（角度） 輸入範圍：非零正值

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1) 若參數 rollover_val 設定為 0，即關閉此功能。

(2) 當軸處於解激磁狀態時，此函式才適用。

(3) 當軸加入軸群組時，此功能不適用。

需求版本

最低支援版本	iA Studio 1.4

5.3.39 GetRolloverTurns

用途

取得軸在單圈模式下的翻轉圈數。

語法

```
int GetRolloverTurns(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸在單圈模式下的翻轉圈數。

需求版本

最低支援版本	iA Studio 1.4

5.3.40 SetOpMode

用途

設置軸的操作模式。

語法

```
int SetOpMode(
    int    axis_id,
    int    op_mode
);
```

參數

axis_id [in]　軸編號。

op_mode [in]　軸的新操作模式。

輸入範圍：1(位置控制)、3(速度控制)、4(轉矩控制)
8(週期同步位置)、9(週期同步速度)、10(週期同步轉矩)

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將物件 0x6060(Mode of operation)配置為 PDO。

需求版本

最低支援版本	iA Studio 3.0

5.3.41 SetBufferMode

用途

設置軸的速度緩衝模式。

語法

```
int SetBufferMode(
    int    axis_id,
    int    buf_mode
);
```

參數

axis_id [in] 軸編號。

buf_mode [in] 軸的新速度緩衝模式。
輸入範圍：0 (立即停止模式)、1(緩衝模式)、2(連續運動模式)

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

5.4 軸狀態

5.4.1 IsEnabled

用途

詢問軸的激磁狀態。

語法

```
int IsEnabled(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸為激磁狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

備註

使用此函式需將物件 0x6041(Status word)配置為 PDO。

需求版本

最低支援版本	iA Studio 0.23

5.4.2 IsMoving

用途

詢問軸的移動狀態。若軸正在移動，軌跡規劃器（PG）會持續輸出新的位置。

語法

```
int IsMoving(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若軸正在移動，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

5.4.3 IsInPos

用途

詢問軸的到位狀態。若軸已到位，位置誤差會小於所設定的目標框，並維持一段時間（反彈跳時間）。

語法

```
int IsInPos(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若軸已到位，則回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

5.4.4 IsErrorStop

用途

詢問軸是否處於 error stop 狀態。

語法

```
int IsErrorStop(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸處於 error stop 狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

5.4.5 IsGantry

用途

詢問軸是否處於龍門狀態。

語法

```
int IsGantry(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸處於龍門狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

5.4.6 IsGrouped

用途

詢問軸是否被歸類至一個軸群組。

語法

```
int IsGrouped(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸被歸類至一個軸群組，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

5.4.7 IsSync

用途

詢問軸是否處於同步狀態。若軸處於同步狀態，軸會追隨主站的命令。

語法

```
int IsSync(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若軸處於同步狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

5.4.8 IsHWLL

用途

詢問軸是否觸發硬體左極限（HWLL）。

語法

```
int IsHWLL(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸處於 HWLL 狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

備註

使用此函式需將物件 0x60FD(Digital inputs)配置為 PDO，指定 bit 0 為左極限輸入。

需求版本

最低支援版本	iA Studio 0.23

5.4.9 IsHWRL

用途

詢問軸是否觸發硬體右極限（HWRL）。

語法

```
int IsHWRL(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若軸處於 HWRL 狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

備註

使用此函式需將物件 0x60FD(Digital inputs)配置為 PDO，指定 bit 1 為右極限輸入。

需求版本

最低支援版本	iA Studio 0.23

5.4.10 IsSWLL

用途

詢問軸是否觸發軟體左極限（SWLL）。

語法

```
int IsSWLL(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸處於 SWLL 狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

5.4.11 IsSWRL

用途

詢問軸是否觸發軟體右極限（SWRL）。

語法

```
int IsSWRL(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸處於 SWRL 狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

5.4.12 IsDriveErr

用途

詢問軸是否觸發驅動器警報。

語法

```
int IsDriveErr(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸觸發驅動器警報，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

備註

使用此函式需將物件 0x6041(Status word)配置為 PDO。

需求版本

最低支援版本	iA Studio 1.4

5.4.13 IsPosErr

用途

詢問軸的位置誤差是否超過保護範圍。

語法

```
int IsPosErr(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸的位置誤差超過保護範圍，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

備註

誤差保護範圍係指控制器內的軸位置誤差容許值。

需求版本

最低支援版本	iA Studio 1.4

5.4.14 IsCompActive

用途

詢問補償功能是否啟動。

語法

```
int IsCompActive(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸已啟動補償功能，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

5.4.15 IsAcc

用途

詢問軸是否正在加速。

語法

```
int IsAcc(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸正在加速，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 2.0

6. 同步運動函式

6.1 概述

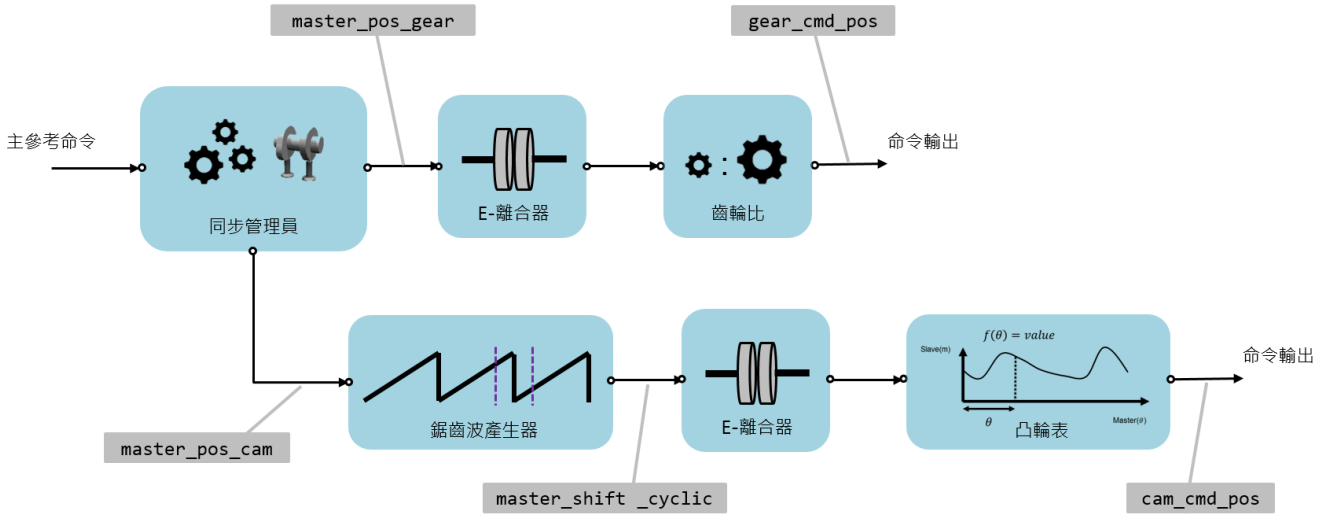


圖 6.1.1

使用者可定義兩軸間的同步運動。作為引導軸的主軸生成位置命令後，從軸會依運動配置參考主軸。若主從關係固定不變，則為電子齒輪傳動；若從軸須遵循某個模式，則為電子凸輪傳動。圖 6.1.2 中，軸 0 作為主軸，引導軸 1、軸 2、軸 3 與軸 4。軸 1、軸 2 與軸 3 採電子齒輪傳動，軸 4 則採電子凸輪傳動。

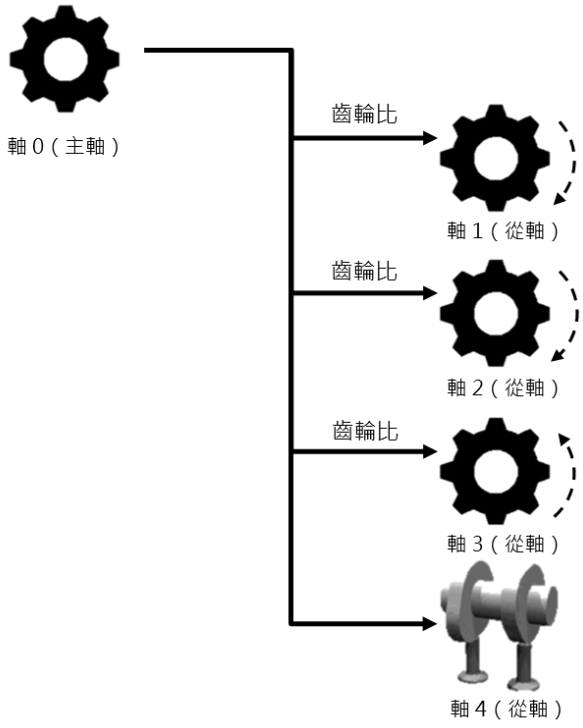


圖 6.1.2

6.1.1 同步運動變數

表 6.1.1.1 為常見的同步運動變數，使用者可利用 iA Studio 的 Scope Manager（請參閱《iA Studio 軟體使用手冊》4.8 節）選擇欲觀測的變數。

表 6.1.1.1

名稱	變數	單位	描述
Raw Master Position	master_pos_gear	毫米 或 角度	主軸的位置命令。
Gear Command Position	gear_cmd_pos	毫米 或 角度	從軸輸出位置命令。
Gear Ratio	gear_ratio	毫米 或 角度	齒輪比。

6.1.2 範例

```
void main()
{

    double target = 100;
    double Gear_ratio[4]={1.0, 2.0, 4.0, -1.0};
    int master = 0;
    int slave[4]={1, 2, 3, 4};

    Enable(master);
    Enable(slave[0]);
    Enable(slave[1]);
    Enable(slave[2]);
    Enable(slave[3]);
    Till(IsEnabled(slave[0]) && IsEnabled(slave[1]) &&
    IsEnabled(slave[2]) && IsEnabled(slave[3]) && IsEnabled(master))

    //  結合兩軸，使其成為主從關係。
    EnableGear(master, slave[0]);
    EnableGear(master, slave[1]);
    EnableGear(master, slave[2]);
```

```
    EnableGear(master, slave[3]);

    //  更改從軸的狀態：脫離→咬合
    GearIn(master, slave[0], Gear_ratio[0]);
    GearIn(master, slave[1], Gear_ratio[1]);
    GearIn(master, slave[2], Gear_ratio[2]);
    GearIn(master, slave[3], Gear_ratio[3]);

    MoveAbs(master, target);
    Till(IsInPos(master));

    //  更改從軸的狀態：咬合→脫離
    GearOut(slave[0]);
    GearOut(slave[1]);
    GearOut(slave[2]);
    GearOut(slave[3]);

}
```

6.2 EnableGear

用途

結合兩軸，使其成為主從關係。

語法

```
int EnableGear(
    int axis_master_id,
    int axis_slave_id
);
```

參數

axis_master_id [in]　　主軸編號。

axis_slave_id [in]　　從軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當兩軸都已激磁，此函式才適用。

需求版本

最低支援版本	iA Studio 1.1

6.3 DisableGear

用途

解除兩軸的主從關係，使其恢復兩獨立軸。

語法

```
int DisableGear(
    int axis_slave_id
);
```

參數

axis_slave_id [in]　　從軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.1

6.4 GearIn

用途

更改從軸的狀態：脫離→咬合。

語法

```
int GearIn(
    int axis_master_id,
    int axis_slave_id,
    double gear_ratio
);
```

參數

axis_master_id [in]	主軸編號。
axis_slave_id [in]	從軸編號。
gear_ratio [in]	齒輪比的值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當兩軸都已激磁，此函式才適用。

需求版本

最低支援版本	iA Studio 1.1

6.5 GearOut

用途

更改從軸的狀態：咬合→脫離。

語法

```
int GearOut(
    int axis_slave_id
);
```

參數

axis_slave_id [in]　　從軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.1

6.6 GetGearRatio

用途

取得從軸的電子齒輪比。

語法

```
double GetGearRatio(
    int axis_slave_id
);
```

參數

axis_slave_id [in]　　從軸編號。

回傳值

從軸的電子齒輪比。

需求版本

最低支援版本	iA Studio 1.3

6.7 IsInGear

用途

詢問從軸是否處於咬合狀態。

語法

```
int IsInGear(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若從軸處於咬合狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 1.3

6.8 IsGearMaster

用途

詢問軸是否為主軸。

語法

```
int IsGearMaster(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸為主軸，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 1.3

6.9 IsGearSlave

用途

詢問軸是否為從軸。

語法

```
int IsGearSlave(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸為從軸，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 1.3

7. 龍門函式

7.1 概述

龍門配置將一對右側軸（RHS）和左側軸（LHS）轉換為一對假想的線性軸（Linear）和旋轉軸（Yaw），如圖 7.1.1。使用者在建立龍門配置之後，可對右側軸下達線性軸方向的命令，以相同方向驅動右側軸和左側軸；對左側軸則下達旋轉軸方向的旋轉運動命令。

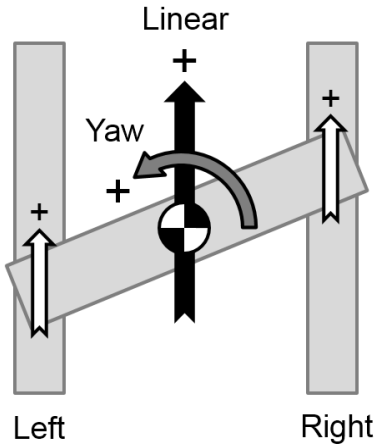


圖 7.1.1

在龍門配置中，線性軸與旋轉軸的位置回授定義如下：

$$Pos_{linear} = \frac{Pos_{RHS} + Pos_{LHS}}{2}; \qquad Pos_{yaw} = \frac{Pos_{RHS} - Pos_{LHS}}{2}$$

Pos_{linear}	：線性軸的位置回授	Pos_{yaw}	：旋轉軸的位置回授
Pos_{RHS}	：右側軸的位置回授	Pos_{LHS}	：左側軸的位置回授

圖 7.1.2 為線性軸、旋轉軸、右側軸與左側軸配置的位置回授示意圖。

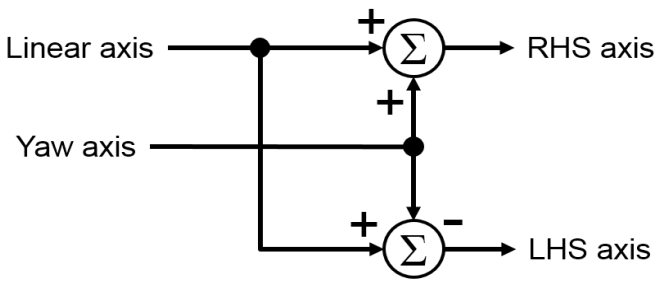


圖 7.1.2

7.1.1 範例

設置龍門的方式如以下 HMPL task 所示。

```
void main() {

    int axis_0 = 0;  //  使用者自定義
    int axis_1 = 1;

    DisableGantryPair(axis_0);  //  解激磁現有的龍門設定
    Till(!IsGantry(axis_0) && !IsGantry(axis_1));

    Enable(axis_0);
    Till(IsEnabled(axis_0));
    Disable(axis_0);
    Till(!IsEnabled(axis_0));

    Enable(axis_1);
    Till(IsEnabled(axis_1));
    Disable(axis_1);
    Till(!IsEnabled(axis_1));

    EnableGantryPair(axis_0, axis_1);
    Enable(axis_0);

    Till(IsEnabled(axis_0) && IsEnabled(axis_1));
    Till(IsGantry(axis_0) && IsGantry(axis_1));
}
```

7.2 EnableGantryPair

用途

建立一對龍門。

語法

```
int EnableGantryPair(
    int rhs_axis_id,
    int lhs_axis_id
);
```

參數

rhs_axis_id [in]　　右側軸之編號。

lhs_axis_id [in]　　左側軸之編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當兩軸都處於解激磁狀態時，此函式才適用。

需求版本

最低支援版本	iA Studio 0.23

7.3 DisableGantryPair

用途

分開一對龍門。

語法

```
int DisableGantryPair(
    int axis_id
);
```

參數

axis_id [in] 龍門中任一軸之編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當兩軸都處於解激磁狀態時，此函式才適用。

需求版本

最低支援版本	iA Studio 0.23

7.4 GetGantryPairID

用途

取得任意龍門軸的龍門 ID。

語法

```
int GetGantryPairID(
    int axis_id
);
```

參數

axis_id [in]　　龍門中任一軸之編號。

回傳值

龍門 ID。

若輸入軸本身不是龍門軸，將回傳-1。

需求版本

最低支援版本	iA Studio 1.3

7.5 IsGantryPair

用途

詢問任意兩軸是否為一對龍門。

語法

```
int IsGantryPair(
    int axis_id_1,
    int axis_id_2
);
```

參數

axis_id_1 [in]　　軸編號 1。

axis_id_2 [in]　　軸編號 2。

回傳值

若兩軸為一對龍門，則回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 1.3

(此頁有意留白。)

8. 軸群組函式

8.1 概述

HIMC 提供多軸直線與圓弧同動插補功能的軸群組運動命令，包含 LineAbs2D / 3D、LineRel2D / 3D、Arc2D、Circle2D 等。與軸運動命令相較，軸群組運動命令保證群組內各軸的同動性，各軸運動的起始與停止時間一致，控制器會依使用者給定的速度規劃命令對各軸運動速度進行調配。HIMC 控制器的基礎功能，支援最多 4 軸的軸群組運動命令（產品型號 MC-XX-XX-XX-00），若有 5 軸同動(或以上)的軸群組運動功能需求，請向本公司或當地經銷商諮詢相關資訊。

圖 8.1.1 為 HIMC 軸群組運動命令的參數流程圖。由各軸的位置回授（Axis Position Feedback），經過正向運動學計算，得到軸群組在機器座標系統中的位置回授（Cartesian Position Feedback）；而根據使用者所給的目標命令，控制器依軸群組的運動軌跡（Motion Profile）規劃空間中的插補命令（Cartesian Position Command），如圖 8.1.2 所示，並由反向運動學計算各軸馬達的對應位置命令（Axis Position Command）。

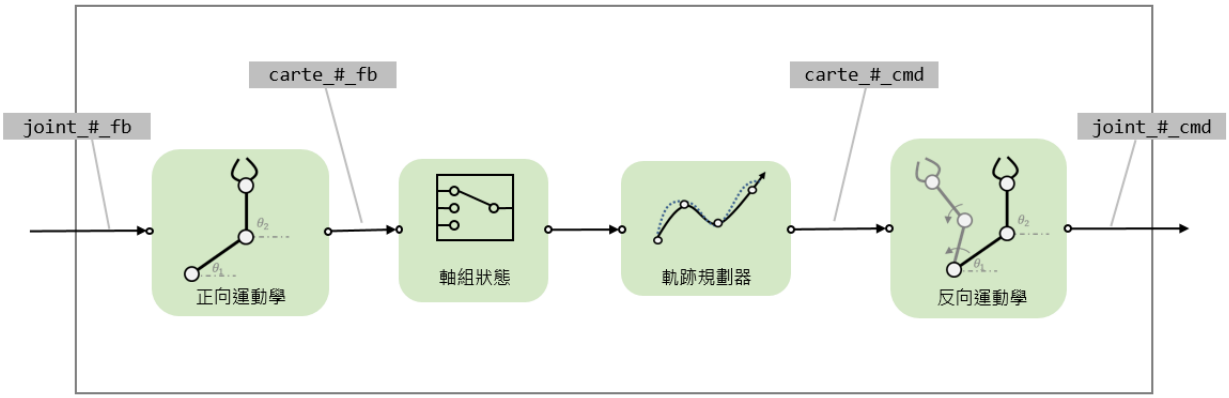


圖 8.1.1

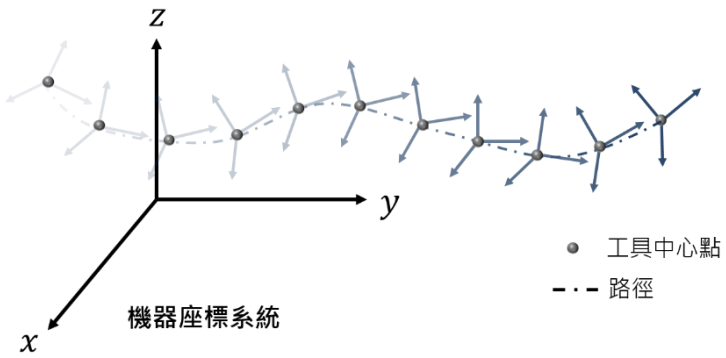


圖 8.1.2

HIMC 會在軸群組運動命令中計算各線段在空間中移動的距離。與軸運動命令不同，其速度規劃是沿著軸群組在空間中移動的方向進行規劃，其移動方向會隨著運動命令的方向而改變。

軸群組運動命令與軸運動命令（請參閱第 5 章）相似，採用 S-Curve 速度規劃，如圖 8.1.3 所示。軸群組在空間中的運動包含平移（Translation）與旋轉（Rotation）兩部分。平移命令由 XYZ 的位置命令所組成；旋轉命令則由 ABC 的旋轉命令所組成。透過軸群組，使用者可設置平移與旋轉的速度規劃參數，包含軌跡規劃器的最大速度、最大加速度、最大減速度與平滑時間。

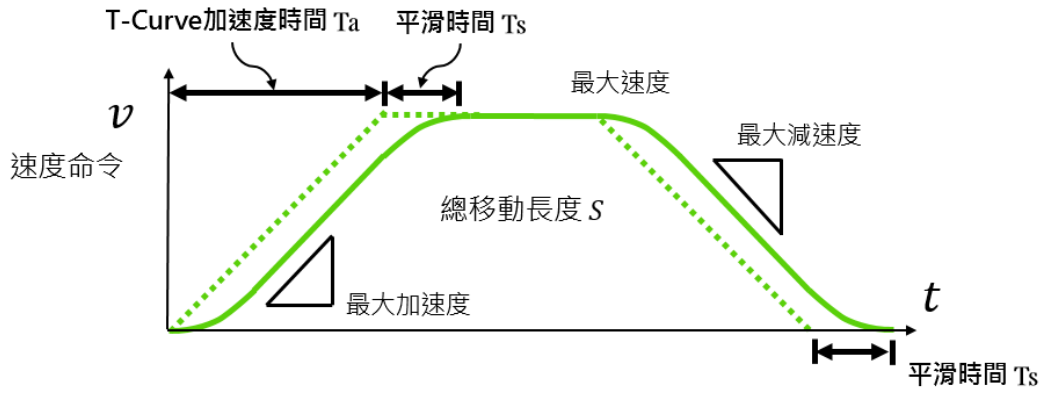


圖 8.1.3

每一個軸群組的運動命令會被視為個別線段（Segment），如圖 8.1.4 所示。在運動過程中，HIMC 會依各個別線段的平移命令與旋轉命令，以及使用者設置的速度規劃參數，計算平移命令與旋轉命令的移動時間，並取移動時間較長者的速度規劃參數作為軸群組的進給速度（Feed Rate）；移動時間較短者，則依此進給速度命令分配後的運動命令移動。

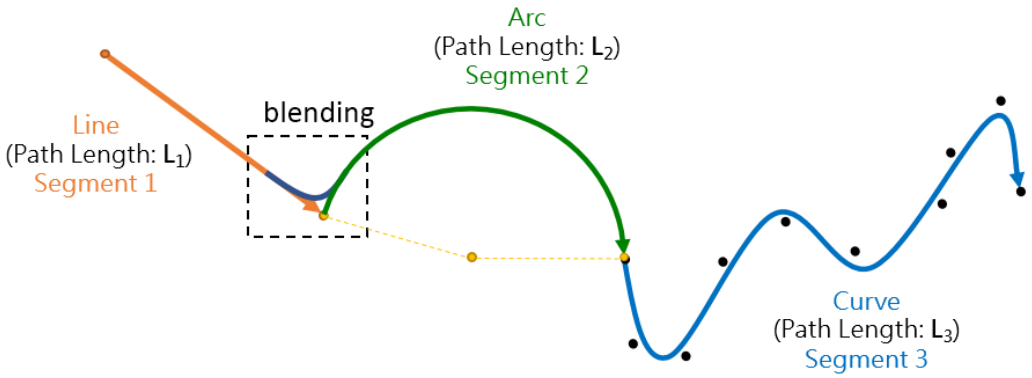


圖 8.1.4

HIMC 內建軸群組命令緩衝空間，每個運動命令的線段都會被放置在此空間，最多能同時接受 512 個運動命令。大於此容量限制的運動命令則會被控制器捨去，並提示錯誤訊息。各個線段的運動命令之間會根據使用者所設定的速度緩衝模式與路徑過渡模式進行速度與路徑的規劃，規劃後的速度曲線與空間路徑有可能會依選擇的模式不同而改變。以圖 8.1.4 為例，若使用速度緩衝模式設定各線段的速度交接，此軸群組的運動總長度為 $S = L_1(Line) + L_2(Arc) + L_3(Curve)$，詳細說明請參閱 8.1.4 與 8.1.5 節。

軸群組的運動狀態與軸的運動狀態相似，分成移動中（Moving）與是否到位（In-Position）。在運動過程中，有如圖 5.1.4 的三個階段，包含：

1. 軸群組運動規劃中（Moving），尚未到位（Not In-Position）。
2. 軸群組運動規劃停止（Not Moving），但尚未到位（Not In-Position）。
3. 軸群組運動規劃停止（Not Moving），已到位（In-Position）。

與軸運動命令利用目標框半徑（Target Radius）與反彈跳時間（Debounce Time）計算軸運動是否到位不同，軸群組命令藉由判斷軸群組底下所有的軸是否已經到位，來決定軸群組的運動是否已經到位；意即當軸群組已經到位時，軸群組底下的各軸也皆在到位的運動狀態。

8.1.1 軸群組變數

軸群組變數分成運動命令變數、運動規劃變數與狀態變數，使用者可利用 iA Studio 的 Scope Manager(請參閱《iA Studio 軟體使用手冊》4.8 節) 選擇欲觀測的變數。詳細說明如表 8.1.1.1 至表 8.1.1.5。

表 8.1.1.1 軸群組運動命令變數

名稱	變數	單位	描述
Cartesian Position Command	carte_pose_cmd	毫米 或 角度	軸群組在機器座標系統（MCS）中的空間位置命令。為一陣列變數，存放 [X Y Z A B C] 的數值。
Cartesian Velocity Command	carte_vel_cmd	毫米/秒 或 角度/秒	軸群組在機器座標系統（MCS）中的空間速度命令。為一陣列變數，存放 [X Y Z A B C] 的數值。
Cartesian Position Feedback	carte_pose_fb	毫米 或 角度	軸群組在機器座標系統（MCS）中的空間位置回授。為一陣列變數。存放 [X Y Z A B C] 的數值。
Axis Position Command	joint_pos_cmd	毫米 或 角度	軸群組在軸座標系統（ACS）中的軸位置命令。為一陣列變數。
Axis Velocity Command	joint_vel_cmd	毫米/秒 或 角度/秒	軸群組在軸座標系統（ACS）中的軸速度命令。為一陣列變數。
Axis Acceleration Command	joint_acc_cmd	毫米/秒 ^{2} 或 角度/秒 ^{2}	軸群組在軸座標系統（ACS）中的軸加速度命令。為一陣列變數。
Axis Position Feedback	joint_pos_fb	毫米 或 角度	軸群組在軸座標系統（ACS）中的軸位置回授。為一陣列變數。
Cartesian Position Error	carte_pose_err	毫米 或 角度	軸群組在機器座標系統（MCS）中的空間位置誤差。為一陣列變數，存放 [X Y Z A B C] 的數值。
Reference Group Position	grp_pg_pos	毫米 或 角度	軸群組參考位置。根據軸群組命令的運動軌跡，經軌跡規劃器生產而成的位置設定點。
Reference Group Velocity	grp_pg_vel	毫米/秒 或 角度/秒	軸群組參考速度。根據軸群組命令的運動軌跡，經軌跡規劃器生產而成的速度設定點。
Reference Group Acceleration	grp_pg_acc	毫米/秒 ^{2} 或 角度/秒 ^{2}	軸群組參考加速度。根據軸群組命令的運動軌跡，經軌跡規劃器生產而成的加速度設定點。

表 8.1.1.2 軸群組運動規劃變數

名稱	變數	單位	描述
Group Max. Linear Profile Velocity	grp_lin_vel	毫米/秒	軸群組線性運動最大速度。不一定會達到。
Group Max. Linear Profile Acceleration	grp_lin_acc	毫米/秒 ^{2}	軸群組線性運動最大加速度。不一定會達到。
Group Max. Linear Profile Deceleration	grp_lin_dec	毫米/秒 ^{2}	軸群組線性運動最大減速度。不一定會達到。
Group Linear Smooth Time	grp_lin_sf	毫秒	軸群組線性運動平滑時間。輸入範圍為 0 ~ 500。增加該值可減少運動期間的機械振動，但會影響總運動時間。
Group Max. Angular Profile Velocity	grp_ang_vel	角度/秒	軸群組旋轉運動最大速度。不一定會達到。
Group Max. Angular Profile Acceleration	grp_ang_acc	角度/秒 ^{2}	軸群組旋轉運動最大加速度。不一定會達到。
Group Max. Angular Profile Deceleration	grp_ang_dec	角度/秒 ^{2}	軸群組旋轉運動最大減速度。不一定會達到。
Group Angular Smooth Time	grp_ang_sf	毫秒	軸群組旋轉運動平滑時間。輸入範圍為 0 ~ 500。增加該值可減少運動期間的機械振動，但會影響總運動時間。

表 8.1.1.3 軸群組狀態變數

名稱	變數	單位	描述
Group Fault Status	grp_fault_status	無	軸群組錯誤狀態，位元定義請參閱表 8.1.1.4。
Group Motion Status	grp_motion_status	無	軸群組運動狀態，位元定義請參閱表 8.1.1.5。

表 8.1.1.4 軸群組錯誤狀態位元定義

位元	名稱	描述	預設反應
0	Error Stop	軸群組處於 error stop 狀態。	無。
1	Axis Fault	從站驅動器的錯誤。	控制器將軸解激磁，軸群組脫離同步。
2	Software Limit	觸發軸的軟體極限。	控制器停止軸的運動，軸群組脫離同步。

表 8.1.1.5 軸群組運動狀態位元定義

位元	名稱	描述	備註
0	Enabled	軸群組激磁狀態。	無。
1	Moving	軸群組移動中。	無。
2	In Position	軸群組到位。	軸群組內所有軸到位。
3	Input Shape	開啟軸群組 Input Shape 濾波器。	請參閱 15.1 節。

8.1.2 座標系統

表 8.1.2.1 為 HIMC 的座標系統定義與說明，包含軸座標系統（Axis Coordinate System，ACS）、機器座標系統（Machine Coordinate System，MCS）、產品座標系統（Product Coordinate System，PCS）、工件座標系統（Workpiece Coordinate System，WCS）、全局座標系統（Global Coordinate System）與座標偏移量（Coordinate Offset）。

表 8.1.2.1

HMPL 定義	描述
CS_ACS	軸座標系統，與個別馬達運動有關。
CS_MCS	機器座標系統（又稱「大地座標系統」或「基座標系統」）。 在機器上具有固定原點的座標系統，藉運動學轉換（參閱 8.1.3 節）與 ACS 連接。 共 6 個維度來表示空間中的位置與方位角（ 3 平移、 3 旋轉）。
CS_PCS	產品座標系統（在 CNC 程式中稱「程式座標系統」），依附在產品或工件上。 可設定座標轉換參數。
CS_WCS# (#=1~15)	工件座標系統，用來設定工件零點。最多提供 15 個獨立的工件座標系統，預設無偏移量。 相依於產品座標系統，可設定座標轉換參數。
CS_GLOBAL	全局座標系統，用來設定全局零點。可建立各個軸群組的全局空間關係。 目前不支援。
CS_OFFSET	座標偏移量，用來設定暫時零點。預設無偏移量，即偏移量座標原點為機器座標原點。 相依於產品座標系統，可設定座標轉換參數。

圖 8.1.2.1 以具有兩旋轉軸的 SCARA 機器人為例，說明 ACS、MCS 與 PCS 之間的關係。ACS 與 MCS 之間透過正逆向運動學轉換（參閱 8.1.3 節），而 MCS 與 PCS 之間則存在相對座標的轉換關係，透過座標的平移與旋轉得到在座標系上的位置。

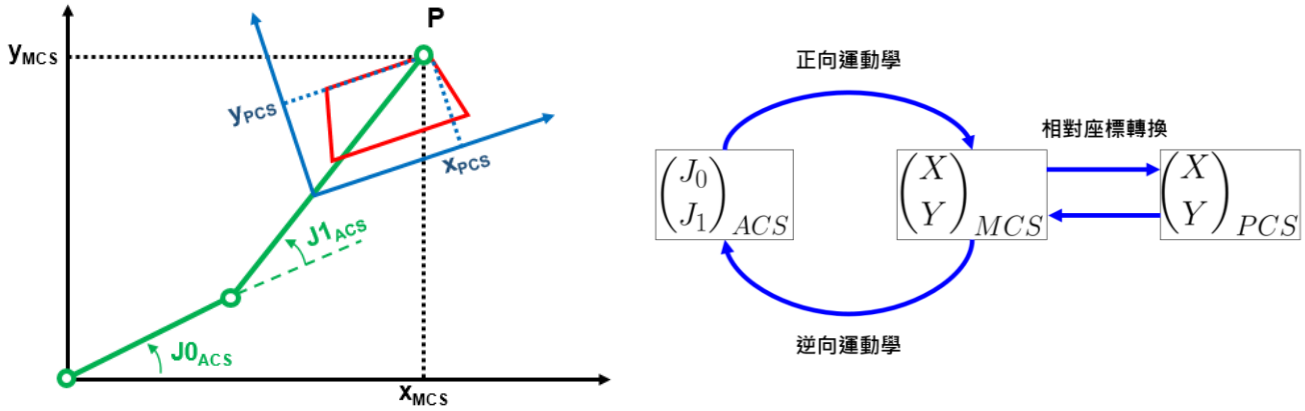


圖 8.1.2.1

將 MCS 轉換至 PCS 時，HIMC 可依需求設定機台的工件座標（WCS1~15）與座標偏移量（OFFSET）。其中，座標系統的設定使用 3 個平移自由度（X、Y、Z）與 3 個旋轉自由度（A、B、C）來表示空間中的姿態（Pose）。

HIMC 採用固定座標系下的 Roll-Pitch-Yaw 旋轉慣例（Rotation Convention）。如圖 8.1.2.2 所示，沿 X 軸旋轉的自由度為 Roll，為角度 A；沿 Y 軸旋轉的自由度為 Pitch，為角度 B；沿 Z 軸旋轉的自由度為 Yaw，為角度 C。此旋轉慣例等同於使用 Tait-Bryan angles 的 ZYX 順序來表示物體在空間中的方位角（Orientation），如圖 8.1.2.3 所示。

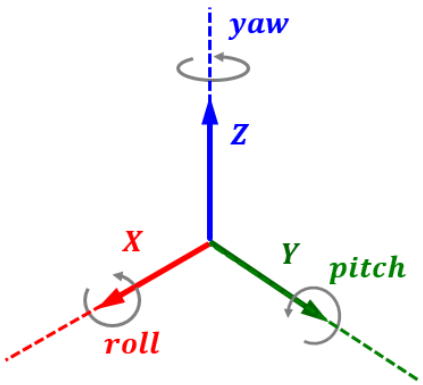


圖 8.1.2.2

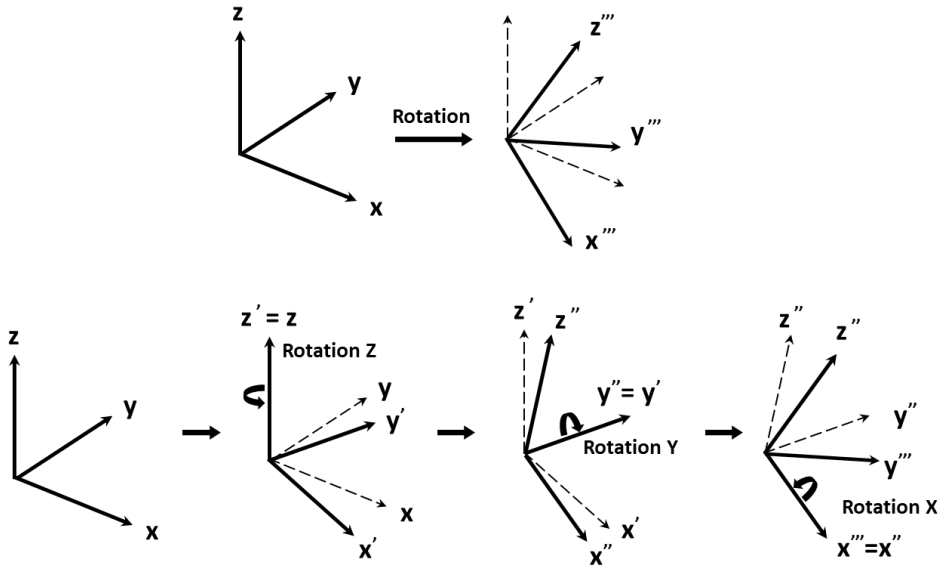


圖 8.1.2.3

假設無座標偏移量，各個 WCS 與 MCS 的對應關係如圖 8.1.2.4 所示。

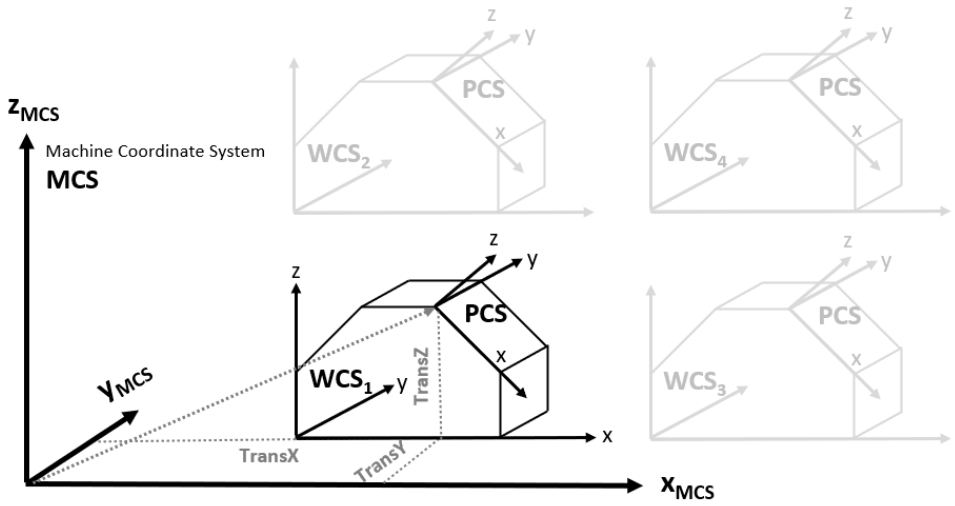


圖 8.1.2.4

若加入座標偏移量，WCS 與 MCS 的對應關係則如圖 8.1.2.5 所示，兩者之間會加入座標偏移量的轉換。

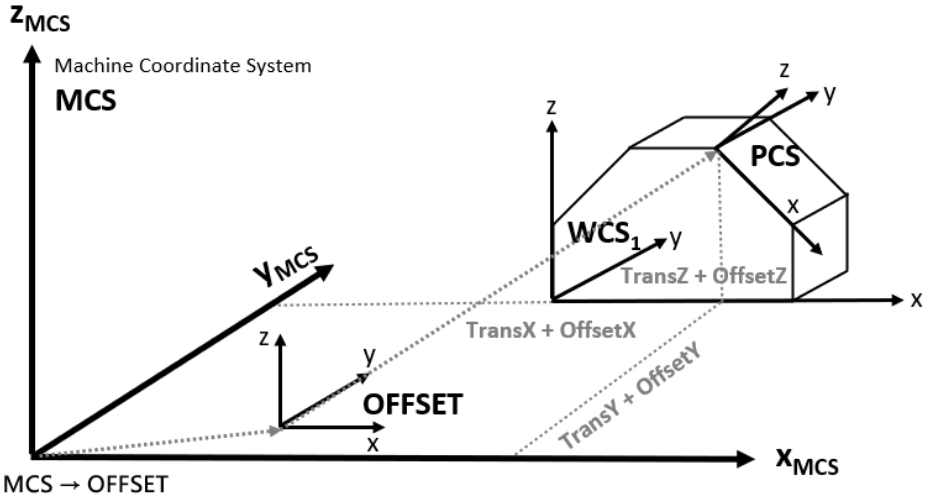


圖 8.1.2.5

依上述座標系統的功能，使用者可在 HIMC 定義座標轉換的參數，依應用需求建立各座標系統之間的轉換關係。圖 8.1.2.6 為各個座標系統之間的關係示意圖，為了讓使用者容易理解，圖中僅採用 XY 平面的座標作為示意，實際應用可設定 6 個自由度的座標轉換參數（X、Y、Z、A、B、C）。

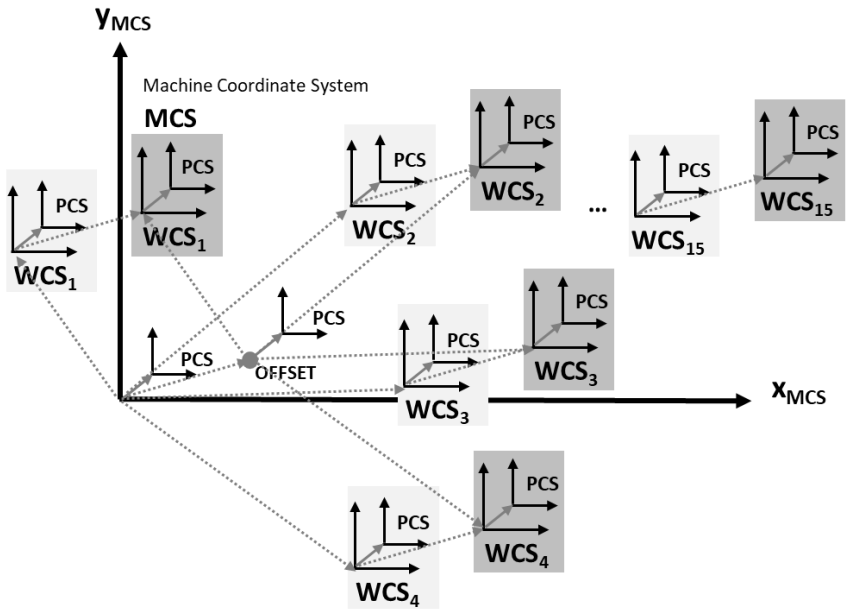


圖 8.1.2.6

8.1.3 運動學

運動學主要處理 ACS（軸座標系統）與 MCS（機器座標系統）之間的轉換關係。由 ACS 中各軸的位置回授計算在 MCS 的空間座標位置，為正向運動學（Forward Kinematics）; 反之，由 MCS 的空間座標位置計算在 ACS 中各軸的位置，則為逆向運動學（Inverse Kinematics）。表 8.1.3.1 為 HIMC 提供的運動學組態定義。

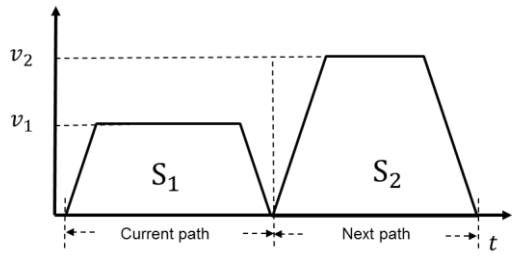
表 8.1.3.1

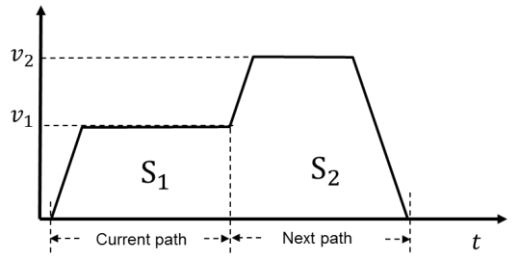
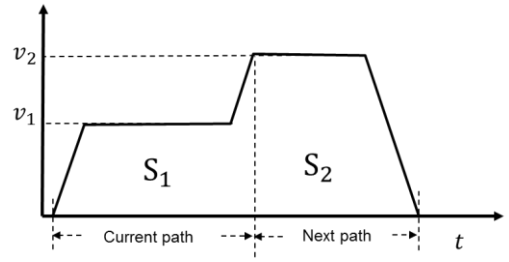
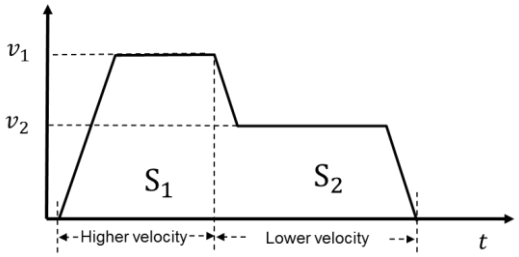
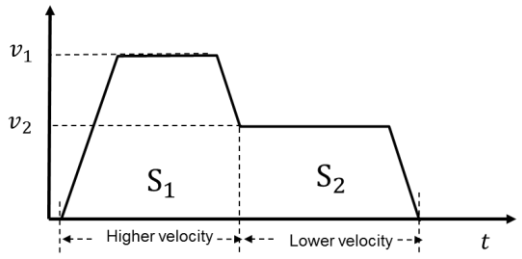
ID	名稱	描述
1	Cartesian	將協調運動中的每個軸分別映射到笛卡爾坐標系的 X、Y、Z、A、B、C 軸。關節空間中，最多可允許 6 個軸。(軸群組的預設值)
2	SCARA	(未開放)
3	WAFER	(未開放)
4	6-Axis Articulated Robot	(未開放)

8.1.4 速度緩衝模式

速度緩衝模式決定了相鄰路徑端點處的速度，使用者可透過此設置規劃相鄰兩段路徑的軌跡速度。表 8.1.4.1 為 HIMC 提供的速度緩衝模式定義。

表 8.1.4.1

HMPL 定義	描述
BM_ABORT	中止當前的運動，並立即執行下一個運動。
BM_BUFF	完成當前的路徑（current path）後，再開始下一個路徑（next path）。 

BM_PREV	其交接速度為當前路徑的速度。(Blending) 
BM_NEXT	其交接速度為下一路徑的速度。(Blending) 
BM_HIGH	其交接速度為兩路徑中較高的速度。(Blending) 
BM_LOW	其交接速度為兩路徑中較低的速度。(Blending) 

8.1.5 路徑過渡模式

路徑過渡模式決定了相鄰路徑間的過渡曲線類型。透過此設置，HIMC 會依據使用者所設定的參數 (請參考表 8.1.5.1) 在兩直線運動命令之間進行轉角平滑化的計算。為達到更佳的規劃路徑，此規劃方式會影響原先的運動軌跡。

表 8.1.5.1

HMPL 定義	描述	相關參數	單位
TM_NONE	無：不插入過渡曲線 (預設模式)	無	無
TM_START_VEL (未開放)	Start velocity	TPStartVelocity	毫米/秒 或 角度/秒
TM_CONST_VEL	Constant velocity	TPVelocity	毫米/秒 或 角度/秒
TM_CORNER_DIST	Corner distance	TPCornerDistance	毫米 或 角度
TM_MAX_CORNER_DEV	Max. corner deviation	TPCornerDeviation	毫米 或 角度
TM_MAX_CORNER_CURV	Max. corner curvature	TPCornerCurv	曲率

使用進階軸群組運動控制命令 LinAbs 和 LinRel 函式，路徑過渡模式選擇 TM_CONST_VEL 指定圓弧運動速度時，會依 8.3.26 節所設定的過渡模式距離參數決定圓弧起點到角落的距離，如圖 8.1.5.1 所示；選擇 TM_CORNER_DIST 指定圓弧起點到角落的距離時，會依 8.3.26 節所設定的過渡模式速度參數決定圓弧運動速度。

若路徑過渡模式計算出的平滑化圓弧半徑超過任一線段的長度，該線段間的過渡模式功能將會被忽略。

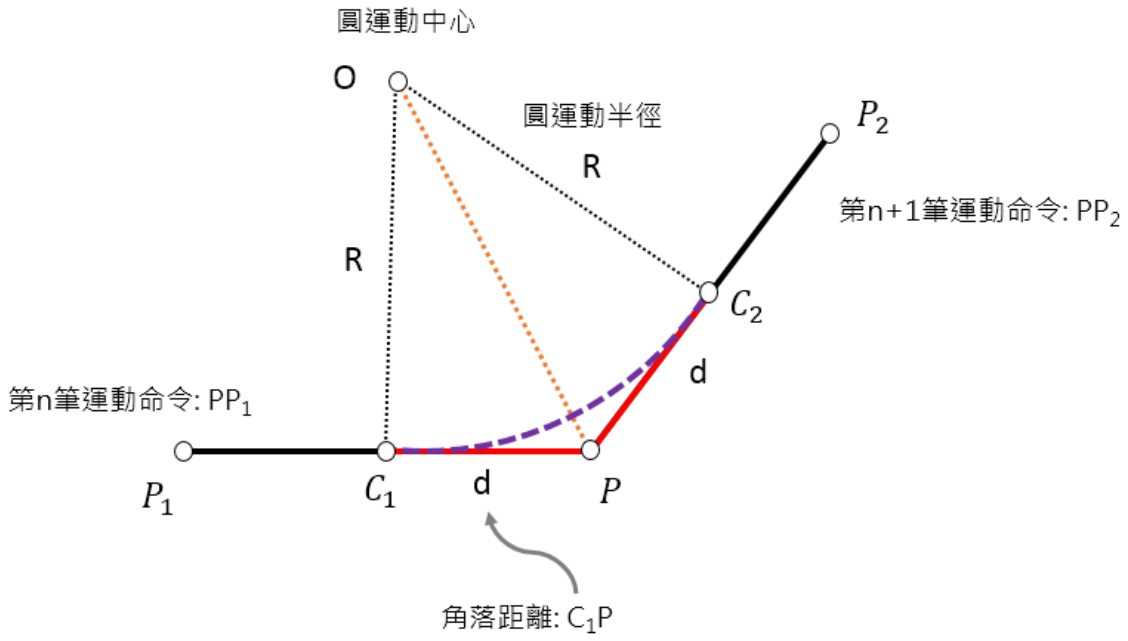

圓運動中心
O
圓運動半徑
R
R
P_2
第n+1筆運動命令: PP_2
C_2
d
第n筆運動命令: PP_1
P_1
C_1
d
P
角落距離: C_1P

圖 8.1.5.1

8.1.6 範例

範例 1：基本軸群組設置與直線運動

建立、致能軸群組並執行協調運動命令的方式如以下 HMPL task 所示。

```
void main() {

    int gid = 0;  //  軸群組編號

    UngrpAllAxes(gid);
    //  移除現有軸群組中所有的軸，並解致能此軸群組。(非強制)

    Enable(0);
    Enable(1);

    Till(IsEnabled(0) && IsEnabled(1));  //  等到所有的軸都被激磁

    SetGrpMotionProfile(gid, 100, 5000, 5000, 200);
    //  為LineAbs2D設定TCP的最大切向運動

    SetupGroup(gid, 0, 1);  //  將軸0和軸1加入軸群組，並致能軸群組。

    LineAbs2D(gid, 100, 100);  //  絕對線性運動
    Till(IsGrpInPos(gid));

    LineAbs2D(gid, 0.0, 0.0);  //  絕對線性運動
    Till(IsGrpInPos(gid));
}
```

此概念與 PLCopen®運動控制第 4 部分《協調運動》中的概念相似。請參閱 PLCopen® 4.1 節 Creating and using an AxisGroup 以獲更多資訊。

註：PLCopen®為 PLCopen 協會授權的註冊商標。

範例 2：進階軸群組設置與速度交接

沿著圖 8.1.6.1 的二維路徑移動工具中心點之方式如以下 HMPL task 所示。此運動軌跡由 **BM_PREV**、**BM_NEXT** 與 **BM_BUFF** 所組成。p_1的協調速度因 **BM_PREV** 而與路徑 1 的最大速度相關；p_2的協調速度則因 **BM_NEXT** 而與路徑 3 的最大速度相關。

註：請參閱 8.1.4 節來了解 BM_PREV 與 BM_NEXT。

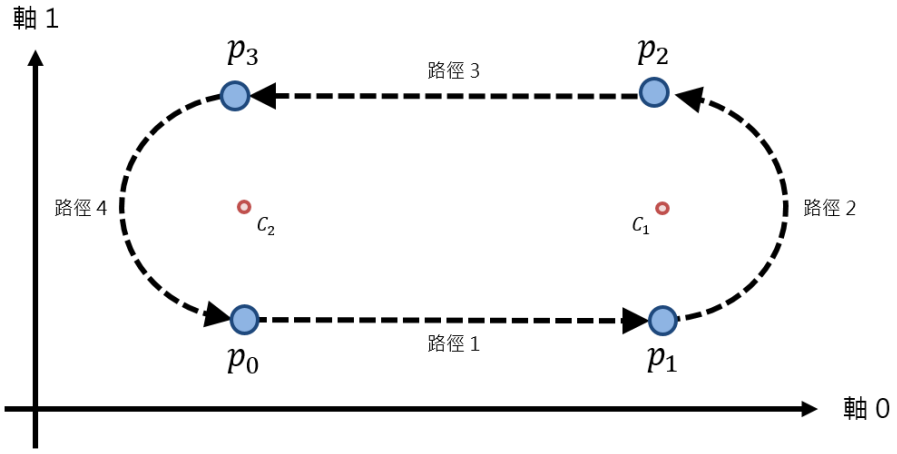


圖 8.1.6.1

```
void main() {

    int axis[2] = {0, 1};  //  軸編號
    int gid = 0;  //  軸群組編號

    UngrpAllAxes(gid);
    //  移除現有軸群組中所有的軸，並解致能此軸群組。（非強制）

    AddAxisToGrp(gid, axis[0]);  //  將軸加入軸群組 0 中
    AddAxisToGrp(gid, axis[1]);

    Enable(axis[0]);  //  激磁軸群組 0 中所有的軸
    Enable(axis[1]);

    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    //  等到所有的軸都被激磁

    EnableGroup(gid);  //  致能軸群組 0
```

```
    double c1[3] = {100, 50, 0};
    double c2[3] = {0, 50, 0};
    double p0[6] = {0, 0, 0, 0, 0, 0};
    double p1[6] = {100, 0, 0, 0, 0, 0};
    double p2[6] = {100, 100, 0, 0, 0, 0};
    double p3[6] = {0, 100, 0, 0, 0, 0};

    double norm_ccw[3] = {0, 0, 1};
    double vel[4] = {100, 5000, 5000, 50};

    double trans_para[4] = {0, 0, 0, 0};

    LineAbs(gid, p0, vel, CS_MCS, BM_BUFF, TM_NONE, trans_para);
    Till(IsGrpInPos(gid));

    //  Blending Next 與 Blending Previous
    LineAbs(gid, p1, vel, CS_MCS, BM_PREV, TM_NONE, trans_para);
    //  路徑 1
    CircleAbs(gid, c1, norm_ccw, 0, p2, vel, CS_MCS, BM_NEXT, TM_NONE, trans_para);
    //  路徑 2
    LineAbs(gid, p3, vel, CS_MCS, BM_PREV, TM_NONE, trans_para);
    //  路徑 3
    Till(IsGrpInPos(gid));

    //  Buffered
    CircleAbs(gid, c2, norm_ccw, 0, p0, vel, CS_MCS, BM_BUFF, TM_NONE, trans_para);
    //  路徑 4
    Till(IsGrpInPos(gid));
}
```

範例 3-1：路徑過渡（直線對直線）

沿著圖 8.1.6.2 的二維路徑移動工具中心點之方式如以下 HMPL task 所示。此運動軌跡由p_0經p_1、p_2、p_3回到p_0後，設定路徑過渡模式為 **TM_CORNER_DIST**，並設定其速度與距離；第二次運動經p_1、p_2、p_3、p_0時，路徑會自動修改成圖 8.1.6.2 中的紅色實線。

註：請參閱 8.1.5 節來了解 TM_CORNER_DIST。

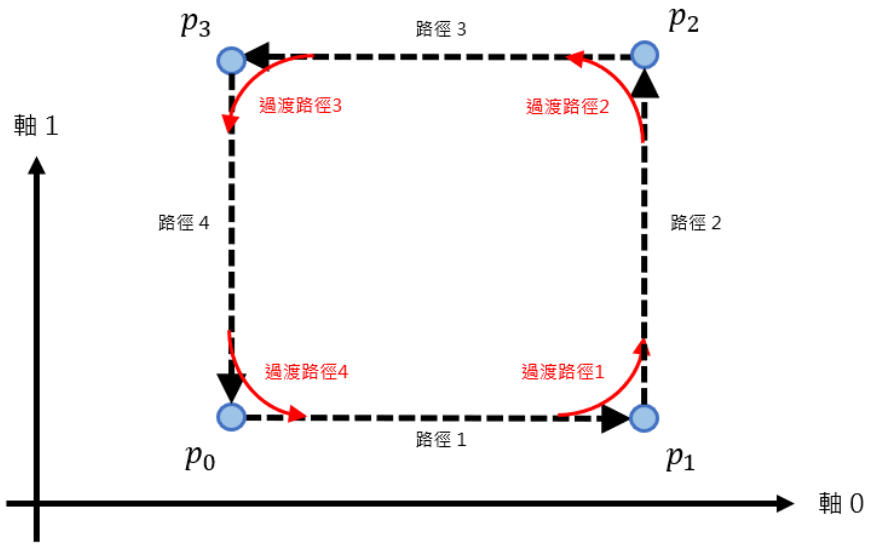


圖 8.1.6.2

```
typedef struct {
    double x, y;
} Point2D;

//  設置空間點位
Point2D p0 = {.x = 0, .y = 0};
Point2D p1 = {.x = 100, .y = 0};
Point2D p2 = {.x = 100, .y = 100};
Point2D p3 = {.x = 0, .y = 100};

void RectangularMotion(int gid) {
    LineAbs2D(gid, p0.x, p0.y);
    LineAbs2D(gid, p1.x, p1.y);
    LineAbs2D(gid, p2.x, p2.y);
    LineAbs2D(gid, p3.x, p3.y);
    LineAbs2D(gid, p0.x, p0.y);
}
```

```
void main() {
    int axis[2] = {0, 1};   //  軸編號
    int gid = 0;            //  軸群組編號
    double round_vel = 50;  //  路徑過渡速度
    double round_dis = 20;  //  路徑過渡距離
    double round_dev = 1;   //  路徑過渡最大誤差
    double round_curv = 5;  //  路徑過渡曲率


    //  建立並致能軸群組
    UngrpAllAxes(gid);
    Enable(axis[0]);       Enable(axis[1]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    SetupGroup(gid, axis[0], axis[1]);

    //  矩形運動軌跡
    RectangularMotion(gid);
    Till(IsGrpInPos(gid));

    //  開啟路徑過渡功能，並設定其速度與距離。
    SetGrpTransMode(gid, TM_CORNER_DIST);
    SetGrpTransPrm(gid, round_vel, round_dis, round_dev, round_curv);


    //  矩形運動軌跡
    RectangularMotion(gid);

    //  給定終點位置
    LineAbs2D(gid, p0.x + round_dis, p0.y);
    Till(IsGrpInPos(gid));

    //  關閉路徑過渡功能
    SetGrpTransMode(gid, TM_NONE);
}
```

範例 3-2：路徑過渡（圓弧對直線、直線對圓弧與圓弧對圓弧）

沿著圖 8.1.6.3 的二維路徑移動工具中心點之方式如以下 HMPL task 所示。此運動軌跡由p_0經p_1、p_2、p_3、p_4回到p_0後，設定路徑過渡模式為 **TM_CORNER_DIST**，並設定其速度與距離；第二次運動經p_1、p_2、p_3、p_4、p_0時，路徑會自動修改成圖 8.1.6.3 中的紅色實線。

註：請參閱 8.1.5 節來了解 TM_CORNER_DIST。

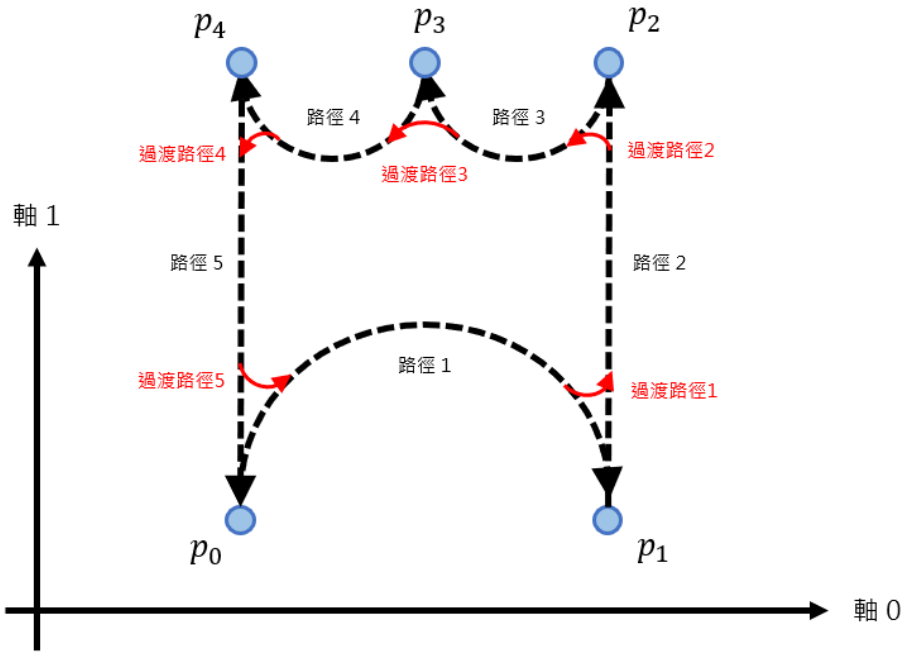


圖 8.1.6.3

```
typedef struct {
    double x, y;
} Point2D;

//  設置空間點位
Point2D pt0 = {.x = 0, .y = 0};
Point2D pt1 = {.x = 100, .y = 0};
Point2D pt2 = {.x = 100, .y = 100};
Point2D pt3 = {.x = 50, .y = 100};
Point2D pt4 = {.x = 0, .y = 100};

Point2D c1 = {.x = 50, .y = 0};
Point2D c2 = {.x = 75, .y = 100};
Point2D c3 = {.x = 25, .y = 100};
```

```
void Motion(int gid) {
    LineAbs2D(gid, pt0.x, pt0.y);
    Circle2D(gid, c1.x, c1.y, pt1.x, pt1.y, -1);
    LineAbs2D(gid, pt2.x, pt2.y);
    Circle2D(gid, c2.x, c2.y, pt3.x, pt3.y, -1);
    Circle2D(gid, c3.x, c3.y, pt4.x, pt4.y, -1);
    LineAbs2D(gid, pt0.x, pt0.y);
}

void main() {
    int axis[2] = {0, 1};   //  軸編號
    int gid = 0;            //  軸群組編號
    double round_vel = 50;  //  路徑過渡速度
    double round_dis = 20;  //  路徑過渡距離

    //  建立並致能軸群組
    UngrpAllAxes(gid);
    Enable(axis[0]); Enable(axis[1]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    SetupGroup(gid, 0, 1);

    //  關閉路徑過渡功能
    SetGrpTransMode(gid, TM_NONE);

    //  運動軌跡
    Motion(gid);
    Till(IsGrpInPos(gid))

    //  開啟路徑過渡功能，並設定其速度與距離。
    SetGrpTransMode(gid, TM_CORNER_DIST);
    SetGrpTransPrm(gid, round_vel, round_dis, 0, 0);

    //  運動軌跡
    Motion(gid);
    Circle2D(gid, c1.x, c1.y, 0.5*(pt0.x + pt1.x), 0.5*(pt0.x + pt1.x), -1);
    Till(IsGrpInPos(gid))
```

```
    //  關閉路徑過渡功能
    SetGrpTransMode(gid, TM_NONE);
}
```

範例 4：三維圓弧運動與螺旋運動

```
void main() {

    int axis[3] = {0, 1, 2};  //  軸編號
    int gid = 0;  //  軸群組編號

    double center1[3] = {0, 50, 0};           //  圓心 1
    double center2[3] = {0, 0, 50};           //  圓心 2
    double end_pos[6] = {0, 0, 0, 0, 0, 0};   //  終點位置
    double norm_x[3] = {1, 0, 0};             //  法向量 x
    double norm_y[3] = {0, 1, 0};             //  法向量 y
    double norm_z[3] = {0, 0, 1};             //  法向量 z
    double vel[4] = {100, 5000, 5000, 50};

    //  建立並致能軸群組
    UngrpAllAxes(gid);
    Enable(axis[0]); Enable(axis[1]); Enable(axis[2]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]) && IsEnabled(axis[2]));
    SetupGroup(gid, axis[0], axis[1], axis[2]);

    //  移動到座標(0, 0, 0)
    LineAbs3D(gid, 0, 0, 0);

    //  三維圓弧運動
    CircleAbs(gid, center1, norm_z, 1, end_pos, vel, CS_MCS, BM_BUFF, TM_NONE, 0);
    CircleAbs(gid, center2, norm_y, 1, end_pos, vel, CS_MCS, BM_BUFF, TM_NONE, 0);
    CircleAbs(gid, center2, norm_x, 1, end_pos, vel, CS_MCS, BM_BUFF, TM_NONE, 0);
    end_pos[2] = 100;

    //  螺旋運動
    CircleAbs(gid, center1, norm_z, 5, end_pos, vel, CS_MCS, BM_BUFF, TM_NONE, 0);

    //  移動到座標(0, 0, 0)
    LineAbs3D(gid, 0, 0, 0);
    Till(IsGrpInPos(gid));
}
```

範例 5-1：座標轉換（MCS→PCS）

產生一個箭頭運動軌跡在機器座標系統（MCS）上，使用者只須設定參考座標為產品座標系統（PCS）與其轉換參數，不須改變原運動軌跡數值，即可將 MCS 轉換至 PCS，如圖 8.1.6.4 所示。

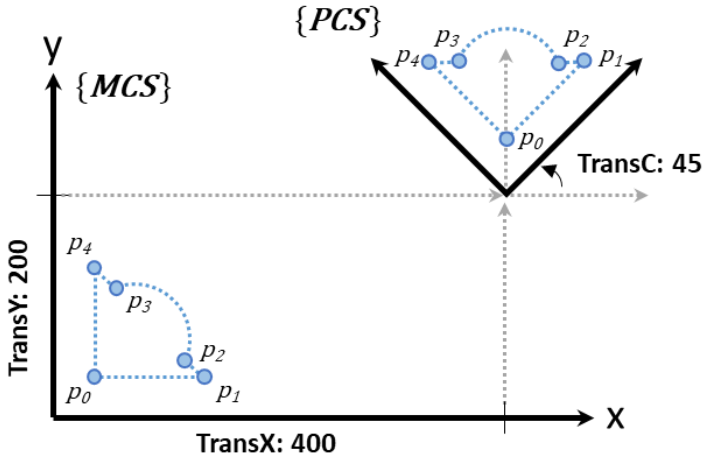


圖 8.1.6.4

```
typedef struct {
    double x, y;
} Point2D;

//  設置空間點位
Point2D p0 = {.x = 10, .y = 10};
Point2D p1 = {.x = 110, .y = 10};
Point2D p2 = {.x = 100, .y = 40};
Point2D p3 = {.x = 40, .y = 100};
Point2D p4 = {.x = 10, .y = 110};
Point2D center = {.x = 60, .y = 60};

void ArrowMotion(int gid) {
    LineAbs2D(gid, p0.x, p0.y);
    LineAbs2D(gid, p1.x, p1.y);
    LineAbs2D(gid, p2.x, p2.y);
    Circle2D(gid, center.x, center.y, p3.x, p3.y, 0);
    LineAbs2D(gid, p4.x, p4.y);
    LineAbs2D(gid, p0.x, p0.y);
}
```

```
void main() {
    int axis[2] = {0, 1};   //  軸編號
    int gid = 0;            //  軸群組編號

    //  建立並致能軸群組
    UngrpAllAxes(gid);
    Enable(axis[0]);       Enable(axis[1]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    SetupGroup(gid, axis[0], axis[1]);   //  軸 0 為 X 軸，軸 1 為 Y 軸

    //  箭頭運動軌跡
    ArrowMotion(gid);
    Till(IsGrpInPos(gid));

    //  設定產品座標轉換參數：X 平移 400 mm、Y 平移 200 mm、Z 旋轉 45 deg
    double transfer[6] = {400, 200, 0, 0, 0, 45};
    SetGrpCoordTrans(gid, CS_PCS, transfer);

    //  參考產品座標系統
    SetGrpCoordSys(gid, CS_PCS);

    //  箭頭運動軌跡
    ArrowMotion(gid);

    //  參考機器座標系統
    SetGrpCoordSys(gid, CS_MCS);

    //  給定終點位置
    LineAbs2D(gid, 0, 0);
    Till(IsGrpInPos(gid));

    //  清除座標轉換參數
    double zeros[6] = {0, 0, 0, 0, 0, 0};
    SetGrpCoordTrans(gid, CS_PCS, zeros);
}
```

範例 5-2：座標轉換（MCS→WCSn→PCS）

同範例 5-1 概念，產生一個箭頭運動軌跡在機器座標系統(MCS)上，使用者只須將工件座標系統(WCS)加入參考座標中，不須改變原運動軌跡數值，即可將原產品座標系統(PCS)的運動軌跡轉移至各個工件座標上，如圖 8.1.6.5 所示。

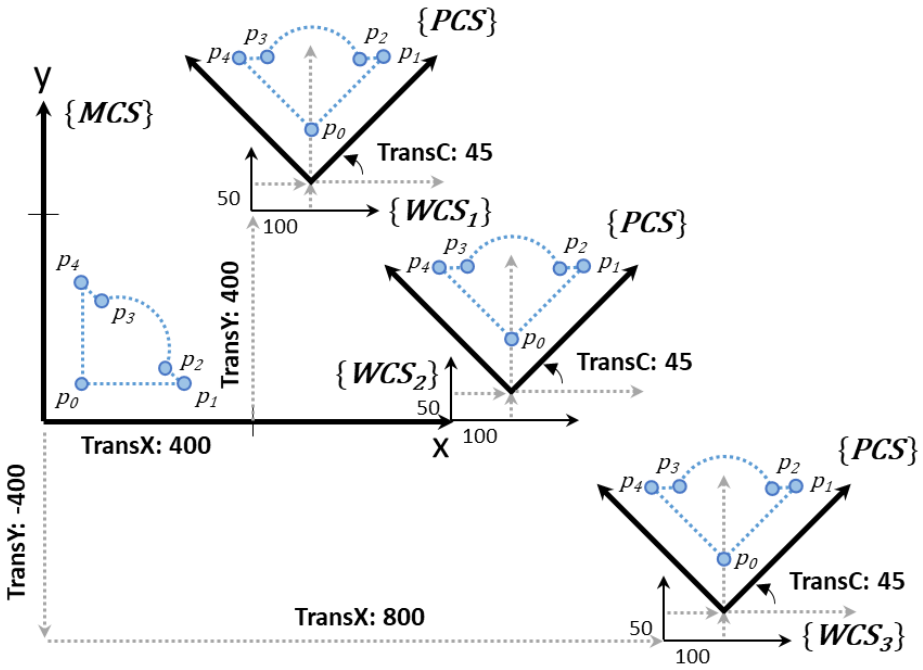


圖 8.1.6.5

```
typedef struct {
    double x, y;
} Point2D;

//  設置空間點位
Point2D p0 = {.x = 10, .y = 10};
Point2D p1 = {.x = 110, .y = 10};
Point2D p2 = {.x = 100, .y = 40};
Point2D p3 = {.x = 40, .y = 100};
Point2D p4 = {.x = 10, .y = 110};
Point2D center = {.x = 60, .y = 60};

void ArrowMotion(int gid) {
    LineAbs2D(gid, p0.x, p0.y);
    LineAbs2D(gid, p1.x, p1.y);
    LineAbs2D(gid, p2.x, p2.y);
```

```
    Circle2D(gid, center.x, center.y, p3.x, p3.y, 0);
    LineAbs2D(gid, p4.x, p4.y);
    LineAbs2D(gid, p0.x, p0.y);
}

void main() {
    int axis[2] = {0, 1};   //  軸編號
    int gid = 0;            //  軸群組編號

    //  建立並致能軸群組
    UngrpAllAxes(gid);
    Enable(axis[0]);       Enable(axis[1]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    SetupGroup(gid, axis[0], axis[1]);   //  軸 0 為 X 軸，軸 1 為 Y 軸

    //  箭頭運動軌跡
    ArrowMotion(gid);
    Till(IsGrpInPos(gid));

    //  設定產品座標轉換參數：X 平移 100 mm、Y 平移 50 mm、Z 旋轉 45 deg
    double transfer[6] = {100, 50, 0, 0, 0, 45};

    //  設定工件座標 1~3 的轉換參數
    double wcs1[6] = {400, 400, 0, 0, 0, 0};
    double wcs2[6] = {600, 0, 0, 0, 0, 0};
    double wcs3[6] = {800, -400, 0, 0, 0, 0};

    SetGrpCoordTrans(gid, CS_PCS, transfer);
    SetGrpCoordTrans(gid, CS_WCS1, wcs1);
    SetGrpCoordTrans(gid, CS_WCS2, wcs2);
    SetGrpCoordTrans(gid, CS_WCS3, wcs3);

    //  參考工件座標系統 1 & 產品座標系統
    SetGrpCoordSys(gid, CS_WCS1 | CS_PCS);
    ArrowMotion(gid);

    //  參考工件座標系統 2 & 產品座標系統
    SetGrpCoordSys(gid, CS_WCS2 | CS_PCS);
```

```
    ArrowMotion(gid);

    //  參考工件座標系統 3 & 產品座標系統
    SetGrpCoordSys(gid, CS_WCS3 | CS_PCS);
    ArrowMotion(gid);

    //  參考機器座標系統
    SetGrpCoordSys(gid, CS_MCS);

    //  給定終點位置
    LineAbs2D(gid, 0, 0);
    Till(IsGrpInPos(gid));

    //  清除座標轉換參數
    double zeros[6] = {0, 0, 0, 0, 0, 0};
    SetGrpCoordTrans(gid, CS_PCS, zeros);
    SetGrpCoordTrans(gid, CS_WCS1, zeros);
    SetGrpCoordTrans(gid, CS_WCS2, zeros);
    SetGrpCoordTrans(gid, CS_WCS3, zeros);
}
```

範例 5-3：座標轉換（MCS→OFFSET→WCSn→PCS）

延續範例 5-2，利用 HIMC 提供的座標偏移量（OFFSET），可偏移範例 5-2 的結果，如圖 8.1.6.6 所示。

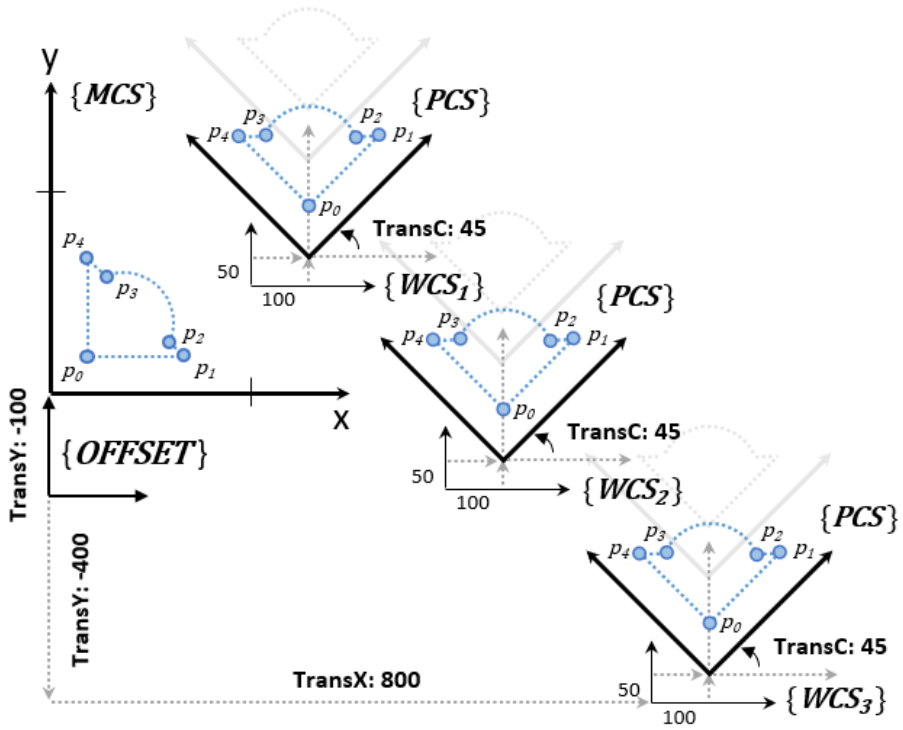


圖 8.1.6.6

```
typedef struct {
    double x, y;
} Point2D;

//  設置空間點位
Point2D p0 = {.x = 10, .y = 10};
Point2D p1 = {.x = 110, .y = 10};
Point2D p2 = {.x = 100, .y = 40};
Point2D p3 = {.x = 40, .y = 100};
Point2D p4 = {.x = 10, .y = 110};
Point2D center = {.x = 60, .y = 60};

void ArrowMotion(int gid) {
    LineAbs2D(gid, p0.x, p0.y);
    LineAbs2D(gid, p1.x, p1.y);
    LineAbs2D(gid, p2.x, p2.y);
    Circle2D(gid, center.x, center.y, p3.x, p3.y, 0);
```

```
    LineAbs2D(gid, p4.x, p4.y);
    LineAbs2D(gid, p0.x, p0.y);
}

void main() {
    int axis[2] = {0, 1};   //  軸編號
    int gid = 0;            //  軸群組編號

    //  建立並致能軸群組
    UngrpAllAxes(gid);
    Enable(axis[0]);       Enable(axis[1]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    SetupGroup(gid, axis[0], axis[1]);   //  軸 0 為 X 軸，軸 1 為 Y 軸

    //  箭頭運動軌跡
    ArrowMotion(gid);
    Till(IsGrpInPos(gid));

    //  設定產品座標轉換參數：X 平移 100 mm、Y 平移 50 mm、Z 旋轉 45 deg
    double transfer[6] = {100, 50, 0, 0, 0, 45};

    //  設定工件座標 1~3 的轉換參數
    double wcs1[6] = {400, 400, 0, 0, 0, 0};
    double wcs2[6] = {600, 0, 0, 0, 0, 0};
    double wcs3[6] = {800, -400, 0, 0, 0, 0};

    //  設定座標偏移量的轉換參數
    double offset[6] = {0, -100, 0, 0, 0, 0};

    SetGrpCoordTrans(gid, CS_PCS, transfer);
    SetGrpCoordTrans(gid, CS_WCS1, wcs1);
    SetGrpCoordTrans(gid, CS_WCS2, wcs2);
    SetGrpCoordTrans(gid, CS_WCS3, wcs3);
    SetGrpCoordTrans(gid, CS_OFFSET, offset);

    //  參考座標偏移量 & 工件座標系統 1 & 產品座標系統
    SetGrpCoordSys(gid, CS_OFFSET | CS_WCS1 | CS_PCS);
    ArrowMotion(gid);
```

```
    //  參考座標偏移量 & 工件座標系統 2 & 產品座標系統
    SetGrpCoordSys(gid, CS_OFFSET | CS_WCS2 | CS_PCS);
    ArrowMotion(gid);

    //  參考座標偏移量 & 工件座標系統 3 & 產品座標系統
    SetGrpCoordSys(gid, CS_OFFSET | CS_WCS3 | CS_PCS);
    ArrowMotion(gid);

    //  參考機器座標系統
    SetGrpCoordSys(gid, CS_MCS);

    //  給定終點位置
    LineAbs2D(gid, 0, 0);
    Till(IsGrpInPos(gid));

    //  清除座標轉換參數
    double zeros[6] = {0, 0, 0, 0, 0, 0};
    SetGrpCoordTrans(gid, CS_PCS, zeros);
    SetGrpCoordTrans(gid, CS_WCS1, zeros);
    SetGrpCoordTrans(gid, CS_WCS2, zeros);
    SetGrpCoordTrans(gid, CS_WCS3, zeros);
    SetGrpCoordTrans(gid, CS_OFFSET, zeros);
}
```

8.2 軸群組運動控制

8.2.1 EnableGroup

用途

致能軸群組。

語法

```
int EnableGroup(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

在執行此函式前，須激磁軸群組內的所有軸。

需求版本

最低支援版本	iA Studio 0.23

8.2.2 DisableGroup

用途

解致能軸群組。

語法

```
int DisableGroup(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.23

8.2.3 ResetGroup

用途

更改軸群組的狀態：Group Error Stop → Group Standby。

語法

```
int ResetGroup(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

清除所有的錯誤之後，才可使用此函式。

需求版本

最低支援版本	iA Studio 1.3

8.2.4 StopGroup

用途

停止軸群組的運動。

語法

```
int StopGroup(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

此函式會清除軸群組原有的路徑規劃。

需求版本

最低支援版本	iA Studio 0.23

8.2.5 HaltGroup

用途

暫停軸群組的運動，軸運動速度被設定為 0。

語法

```
int HaltGroup(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

若軸群組尚未到位，軸群組會繼續移動。

需求版本

最低支援版本	iA Studio 1.3

8.2.6 ResumeGroup

用途

由軸群組暫停的狀態中恢復軸群組的運動。

語法

```
int ResumeGroup(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.2.7 JogGroup

用途

命令一個軸群組，在機器座標系統中的指定方向以特定速度持續移動。

語法

```
int JogGroup(
    int    group_id,
    int    carte_dir,
    double jog_vel
);
```

參數

group_id [in]	軸群組編號。
carte_dir [in]	機器座標系統中的運動方向。 數字 0 ~ 5 依序代表機器座標系統中的 6-DOF { X, Y, Z, A, B, C } 。
jog_vel [in]	移動速度的值。 數值的正負值表示往運動方向的同方向或反方向移動。 參數單位：mm/s (毫米/秒) 或 deg/s (角度/秒)

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.4

8.2.8 JogGroupAxis

用途

命令一個軸群組中的特定軸，在軸座標系統中以特定速度持續移動。

語法

```
int JogGroupAxis(
    int    group_id,
    int    grp_axis,
    double jog_vel
);
```

參數

group_id [in]	軸群組編號。
grp_axis [in]	軸群組中的軸編號。 數字 0 ~ 8 依序代表各軸加入軸群組的順序。
jog_vel [in]	移動速度的值。 數值的正負值表示往運動方向的同方向或反方向移動。 參數單位：mm/s（毫米/秒）或 deg/s（角度/秒）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.4

8.2.9 LineAbs2D

用途

命令一個軸群組，二維內插線性移動至機器座標系統中的絕對位置。

語法

```
int LineAbs2D(
    int    group_id,
    double end_x,
    double end_y
);
```

參數

group_id [in]	軸群組編號。
end_x [in]	絕對目標位置在 X 軸的值。
end_y [in]	絕對目標位置在 Y 軸的值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
void main() {
    //  假設已致能內含兩正交軸的軸群組 0
    double target_x = 100;
    double target_y = 100;
    LineAbs2D (0 /* group_id */, target_x, target_y);
    //  移動至 ( target_x, target_y )
    //  也就是 ( 100, 100 )
}
```

需求版本

最低支援版本	iA Studio 0.24

8.2.10 LineAbs3D

用途

命令一個軸群組，三維內插線性移動至機器座標系統中的絕對位置。

語法

```
int LineAbs3D(
    int    group_id,
    double end_x,
    double end_y,
    double end_z
);
```

參數

group_id [in]	軸群組編號。
end_x [in]	絕對目標位置在 X 軸的值。
end_y [in]	絕對目標位置在 Y 軸的值。
end_z [in]	絕對目標位置在 Z 軸的值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.24

8.2.11 LineRel2D

用途

命令一個軸群組，二維內插線性移動至機器座標系統中的相對位置。

語法

```
int LineRel2D(
    int    group_id,
    double distance_x,
    double distance_y
);
```

參數

group_id [in]	軸群組編號。
distance_x [in]	相對距離在 X 軸的值。
distance_y [in]	相對距離在 Y 軸的值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
void main() {
    //  假設已致能內含兩正交軸的軸群組 0
    //  且兩軸的起點為 ( 100, 200 )
    double distance_x = 100;
    double distance_y = 100;
    LineRel2D (0 /* group_id */, distance_x, distance_y);
    //  移動至 ( X 起點 + distance_x, Y 起點 + distance_y )
    //  也就是 ( 200, 300 )
}
```

需求版本

最低支援版本	iA Studio 0.24

8.2.12 LineRel3D

用途

命令一個軸群組，三維內插線性移動至機器座標系統中的相對位置。

語法

```
int LineRel3D(
    int    group_id,
    double distance_x,
    double distance_y,
    double distance_z
);
```

參數

group_id [in]	軸群組編號。
distance_x [in]	相對距離在 X 軸的值。
distance_y [in]	相對距離在 Y 軸的值。
distance_z [in]	相對距離在 Z 軸的值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.24

8.2.13 Arc2D

用途

命令一個軸群組，二維內插圓弧移動至機器座標系統中的絕對位置。

語法

```
int Arc2D(
    int    group_id,
    double border_x,
    double border_y,
    double end_x,
    double end_y
);
```

參數

group_id [in]　　軸群組編號。
border_x [in]　　絕對中繼位置在 X 軸的值。
border_y [in]　　絕對中繼位置在 Y 軸的值。
end_x [in]　　絕對終點位置在 X 軸的值。
end_y [in]　　絕對終點位置在 Y 軸的值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
void main() {
    //  假設已致能內含兩正交軸的軸群組 0
    //  且兩軸的起點為 ( 0, 0 )
    double border_x = 100;
    double border_y = 100;
    double end_x = 200;
    double end_y = 0;
    Arc2D(0 /* group_id */, border_x, border_y, end_x, end_y);
    //  經 ( border_x , border_y ) 圓弧移動至 ( end_x, end_y )
    //  也就是經 ( 100, 100 ) 圓弧移動至 ( 200, 0 )
}
```

需求版本

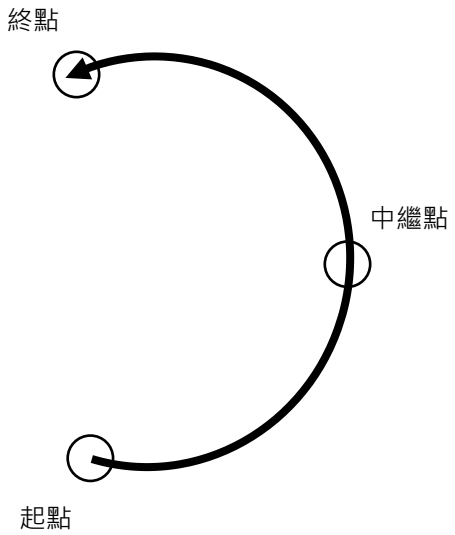
最低支援版本	iA Studio 0.24

優點

使用者可指定中繼點 (運動中的最遠點) ，
並確保機器可取得此點。

缺點

在單一命令中，其角度被限制為<2π。



8.2.14 ArcCW2D

用途

命令一個軸群組，二維內插圓弧順時針移動至機器座標系統中的絕對位置。

語法

```
int ArcCW2D(
    int    group_id,
    double center_x,
    double center_y,
    double end_x,
    double end_y
);
```

參數

group_id [in]	軸群組編號。
center_x [in]	絕對中心位置在 X 軸的值。
center_y [in]	絕對中心位置在 Y 軸的值。
end_x [in]	絕對終點位置在 X 軸的值。
end_y [in]	絕對終點位置在 Y 軸的值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.2.15 ArcCCW2D

用途

命令一個軸群組，二維內插圓弧逆時針移動至機器座標系統中的絕對位置。

語法

```
int ArcCCW2D(
    int    group_id,
    double center_x,
    double center_y,
    double end_x,
    double end_y
);
```

參數

group_id [in]	軸群組編號。
center_x [in]	絕對中心位置在 X 軸的值。
center_y [in]	絕對中心位置在 Y 軸的值。
end_x [in]	絕對終點位置在 X 軸的值。
end_y [in]	絕對終點位置在 Y 軸的值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.2.16 ArcAngle2D

用途

命令一個軸群組依給定角度，二維內插圓弧移動至機器座標系統中的絕對位置。

語法

```
int ArcAngle2D(
    int    group_id,
    double center_x,
    double center_y,
    double angle
);
```

參數

group_id [in]	軸群組編號。
center_x [in]	絕對中心位置在 X 軸的值。
center_y [in]	絕對中心位置在 Y 軸的值。
angle [in]	起點與終點相對於絕對中心位置的夾角，決定了圓弧移動的方向及旋轉角度。 參數單位：deg (角度)

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

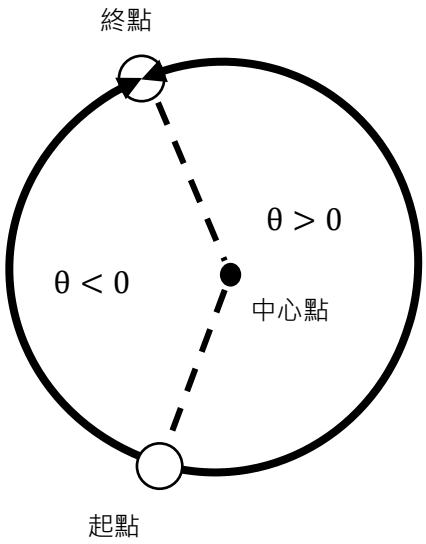
參數 angle 代表圓軌跡旋轉的方向，若 angle > 0：逆時針移動；若 angle < 0：順時針移動。

範例

```
void main() {
    //  假設已致能內含兩正交軸的軸群組 0
    //  且兩軸的起點為 ( 0, 0 )
    double center_x = 10;
    double center_y = 10;
    double angle = 45;
    ArcAngle2D(0 /* group_id */, center_x, center_y, angle);
    //  以 ( center_x, center_y ) 為中心旋轉 45 度
    //  也就是以 ( 10, 10 ) 為中心，從 ( 0, 0 ) 圓弧移動至 ( 10, 10-10√2 )
}
```

需求版本

最低支援版本	iA Studio 2.0



θ 的值決定了圓弧移動的方向。

θ > 0：逆時針移動

θ < 0：順時針移動

8.2.17 Circle2D

用途

命令一個軸群組，二維內插圓周運動至機器座標系統中的絕對位置。

語法

```
int Circle2D(
    int    group_id,
    double center_x,
    double center_y,
    double end_x,
    double end_y,
    int    turns
);
```

參數

group_id [in]	軸群組編號。
center_x [in]	絕對中心位置在 X 軸的值。
center_y [in]	絕對中心位置在 Y 軸的值。
end_x [in]	絕對終點位置在 X 軸的值。
end_y [in]	絕對終點位置在 Y 軸的值。
turns [in]	相對於起點的圓周運動圈數，決定了圓周運動的方向及總角度。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1) 參數 turns 代表圓軌跡旋轉的方向，若
turns >= 0：逆時針移動；
turns < 0：順時針移動。

(2) 當 ||turns|| <= 1 時，圓軌跡的總移動角度<360°；
若圓軌跡總移動角度>=360°（即一圈，或一圈以上），||turns|| 須 >= 2。

(3) 使用此函式時，turns = 0 與 turns = 1 的行為定義相同。

範例

```
void main() {
    //  假設已致能內含兩正交軸的軸群組 0
    //  且兩軸的起點為 ( 0, 0 )
    double center_x = 100;
    double center_y = 0;
    double end_x = 200;
    double end_y = 0;
    int turns = 1;
    Circle2D(0 /* group_id */, center_x, center_y, end_x, end_y, turns);
    //  以 ( center_x, center_y ) 為中心做圓周運動，並移動至 ( end_x, end_y )
    //  也就是以 ( 100, 0 ) 為中心做圓周運動，並移動至 ( 200, 0 )
}
```

需求版本

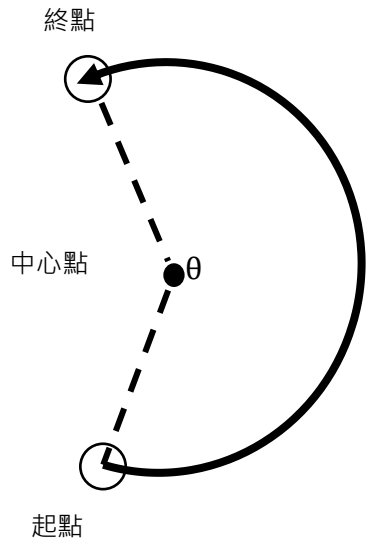
最低支援版本	iA Studio 1.1

優點

無角度限制。

缺點

使用者無法指定中繼點 (運動中的最遠點)。因此，機器不一定可以取得此點。



圈數的值決定了圓周運動的方向。

※ 角度 = θ + 圈數 × 360

圈數 ≥ 0 為正轉，圈數< 0 為反轉。(注意：圈數 = 0 與 圈數 = 1 的運動軌跡相同。)

下表以 θ = 210° 為例。

圈數	計算	角度
-2	$210 - 2 \times 360°$	$-510°$
-1	$210 - 1 \times 360°$	$-150°$
0	$210 + 0 \times 360°$	$210°$
1	$210 + 0 \times 360°$	$210°$
2	$210 + 1 \times 360°$	$570°$

8.3 軸群組設定

8.3.1 AddAxisToGrp

用途

將軸加入一個具有特定序列的軸群組中。

語法

```
int AddAxisToGrp(
    int group_id,
    int axis_id
);
```

參數

group_id [in]　　軸群組編號。

axis_id [in]　　軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1)　最大軸數為 9。

(2)　須依 { X, Y, Z, A, B, C } 的順序加入軸。

需求版本

最低支援版本	iA Studio 0.23

8.3.2 RemoveAxisFromGrp

用途

從一個具有特定序列的軸群組中移除最後一軸。

語法

```
int RemoveAxisFromGrp(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.24

8.3.3 SetupGroup

用途

設定並致能一個具有特定序列的軸群組。

語法

```
int SetupGroup(
    int group_id,
    int axis_id,
    int axis_id,
    …
    int axis_id
);
```

參數

group_id [in]　　軸群組編號。

axis_id [in]　　軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

最大軸數為 9。

需求版本

最低支援版本	iA Studio 0.25

8.3.4 UngrpAllAxes

用途

拆散軸群組。

語法

```
int UngrpAllAxes(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.23

8.3.5 GetGroupID

目的

取得軸所屬的軸群組 ID。

語法

```
int GetGroupID(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸所屬的軸群組 ID。

若此軸不屬於任何軸群組，其值為-1。

需求版本

最低支援版本	iA Studio 0.23

8.3.6 SetGrpMotionProfile

用途

設置軸群組的 TCP 線性運動參數。

語法

```
int SetGrpMotionProfile(
    int    group_id,
    double max_velocity,
    double max_acceleration,
    double max_deceleration,
    double smooth_time
);
```

參數

group_id [in] 軸群組編號。

max_velocity [in] 軸群組線性運動的最大速度。
參數單位：mm/s（毫米/秒）
輸入範圍：0 ~ 5000

max_acceleration [in] 軸群組線性運動的最大加速度。
參數單位：mm/s^2（毫米/秒 ^{2}）
輸入範圍：>0 ~ 50000（加速度不能為 0）

max_deceleration [in] 軸群組線性運動的最大減速度。
參數單位：mm/s^2（毫米/秒 ^{2}）
輸入範圍：>0 ~ 50000（減速度不能為 0）

smooth_time [in] 軸群組線性運動的平滑時間。
參數單位：ms（毫秒）
輸入範圍：0 ~ 500

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

軸群組線性運動的參數預設值分別為 [100（速度）, 500（加速度）, 500（減速度）, 50（平滑時間）]。

需求版本

最低支援版本	iA Studio 0.24

8.3.7 SetGrpAngMotionProfile

用途

設置軸群組的 TCP 旋轉運動參數。

語法

```
int SetGrpAngMotionProfile(
    int    group_id,
    double max_velocity,
    double max_acceleration,
    double max_deceleration,
    double smooth_time
);
```

參數

group_id [in] 軸群組編號。

max_velocity [in] 軸群組旋轉運動的最大速度。
參數單位：deg/s（角度/秒）
輸入範圍：0 ~ 7200

max_acceleration [in] 軸群組旋轉運動的最大加速度。
參數單位：deg/s^2（角度/秒 ^{2}）
輸入範圍：>0 ~ 72000（加速度不能為 0）

max_deceleration [in] 軸群組旋轉運動的最大減速度。
參數單位：deg/s^2（角度/秒 ^{2}）
輸入範圍：>0 ~ 72000（減速度不能為 0）

smooth_time [in] 軸群組旋轉運動的平滑時間。
參數單位：ms（毫秒）
輸入範圍：0 ~ 500

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

軸群組旋轉運動的參數預設值分別為 [360 (速度) , 1800 (加速度) , 1800 (減速度) , 50 (平滑時間)]。

需求版本

最低支援版本	iA Studio 2.0

8.3.8 GetGrpKin

用途

取得軸群組的運動學模式。

語法

```
int GetGrpKin(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

軸群組的運動學模式，請參閱 8.1.3 節。

需求版本

最低支援版本	iA Studio 1.3

8.3.9 SetGrpKin

用途

設置軸群組的運動學模式。

語法

```
int SetGrpKin(
    int group_id,
    int kin_type
);
```

參數

group_id [in]　　軸群組編號。

kin_type [in]　　軸群組的新運動學模式，請參閱 8.1.3 節。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.23

8.3.10 GetGrpMaxVel

用途

取得軸群組的最大速度。

語法

```
double GetGrpMaxVel(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

軸群組的最大速度。

單位：mm/s（毫米/秒）或 deg/s（角度/秒）

需求版本

最低支援版本	iA Studio 1.3

8.3.11 SetGrpVel

用途

設置軸群組的最大速度。

語法

```
int SetGrpVel(
    int    group_id,
    double vel
);
```

參數

group_id [in]	軸群組編號。
vel [in]	軸群組的新最大速度。 參數單位：mm/s（毫米/秒）或 deg/s（角度/秒） 輸入範圍：0 ~ 5000

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.3.12 GetGrpMaxAcc

用途

取得軸群組的最大加速度。

語法

```
double GetGrpMaxAcc(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

軸群組的最大加速度。

單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2）

需求版本

最低支援版本	iA Studio 1.3

8.3.13 SetGrpAcc

用途

設置軸群組的最大加速度。

語法

```
int SetGrpAcc(
    int    group_id,
    double acc
);
```

參數

group_id [in]	軸群組編號。
acc [in]	軸群組的新最大加速度。 參數單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2） 輸入範圍：>0 ~ 50000（加速度不能為 0）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.3.14 SetGrpAccTime

用途

設置軸群組的加速度時間。

語法

```
int SetGrpAccTime(
    int    group_id,
    double acc_time
);
```

參數

group_id [in] 軸群組編號。

acc_time [in] 軸群組的加速度時間。
參數單位：ms（毫秒）
輸入範圍：非零正值

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.3.15 GetGrpMaxDec

用途

取得軸群組的最大減速度。

語法

```
double GetGrpMaxDec(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

軸群組的最大減速度。

單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2）

需求版本

最低支援版本	iA Studio 1.3

8.3.16 SetGrpDec

用途

設置軸群組的最大減速度。

語法

```
int SetGrpDec(
    int    group_id,
    double dec
);
```

參數

group_id [in] 軸群組編號。

dec [in] 軸群組的新最大減速度。

參數單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2）

輸入範圍：>0 ~ 50000（減速度不能為 0）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.3.17 SetGrpDecTime

用途

設置軸群組的減速度時間。

語法

```
int SetGrpDecTime(
    int    group_id,
    double dec_time
);
```

參數

group_id [in] 軸群組編號。

dec_time [in] 軸群組的減速度時間。
參數單位：ms（毫秒）
輸入範圍：非零正值

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.3.18 GetGrpSMTime

用途

取得軸群組的平滑時間。

語法

```
double GetGrpSMTime(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

軸群組的平滑時間。

單位：ms (毫秒)

需求版本

最低支援版本	iA Studio 1.3

8.3.19 SetGrpSMTime

用途

設置軸群組的平滑時間。

語法

```
int SetGrpSMTime(
    int    group_id,
    double smooth_time
);
```

參數

group_id [in]	軸群組編號。
smooth_time [in]	軸群組的新平滑時間。 參數單位：ms（毫秒） 輸入範圍：0 ~ 500

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.3.20 GetGrpCoordSys

用途

取得軸群組的座標系統。

語法

```
int GetGrpCoordSys(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

軸群組的座標系統，請參閱 8.1.2 節。

需求版本

最低支援版本	iA Studio 1.3

8.3.21 SetGrpCoordSys

用途

設置軸群組的座標系統。

語法

```
int SetGrpCoordSys(
    int group_id,
    int coord_sys
);
```

參數

group_id [in]	軸群組編號。
coord_sys [in]	軸群組的新座標系統，請參閱 8.1.2 節。 範例：1. CS_MCS 2. CS_WCS1 | CS_PCS 3. CS_OFFSET | CS_WCS2 | CS_PCS

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 2.0

8.3.22 GetGrpBufferMode

用途

取得軸群組的速度緩衝模式。

語法

```
int GetGrpBufferMode(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

軸群組的速度緩衝模式，請參閱 8.1.4 節。

需求版本

最低支援版本	iA Studio 1.3

8.3.23 SetGrpBufferMode

用途

設置軸群組的速度緩衝模式。

語法

```
int SetGrpBufferMode(
    int group_id,
    int buffer_mode
);
```

參數

group_id [in]	軸群組編號。
buffer_mode [in]	軸群組的新速度緩衝模式，請參閱 8.1.4 節。 輸入範圍：0 ~ 5

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.3.24 GetGrpTransMode

用途

取得軸群組的路徑過渡模式

語法

```
int GetGrpTransMode(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

軸群組的路徑過渡模式，請參閱 8.1.5 節。

需求版本

最低支援版本	iA Studio 1.3

8.3.25 SetGrpTransMode

用途

設置軸群組的路徑過渡模式。

語法

```
int SetGrpTransMode(
    int group_id,
    int trans_mode
);
```

參數

group_id [in]	軸群組編號。
trans_mode [in]	軸群組的新路徑過渡模式，請參閱 8.1.5 節。 輸入範圍：0 ~ 4

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.3.26 SetGrpTransPrm

用途

設置軸群組的路徑過渡模式參數。

語法

```
int SetGrpTransPrm(
    int    group_id,
    double trans_vel,
    double trans_dis,
    double trans_dev,
    double trans_curv
);
```

參數

group_id [in]	軸群組編號。
trans_vel [in]	軸群組新的路徑過渡模式速度參數，請參閱 8.1.5 節。
trans_dis [in]	軸群組新的路徑過渡模式距離參數，請參閱 8.1.5 節。
trans_dev [in]	軸群組新的路徑過渡模式最大誤差參數，請參閱 8.1.5 節。
trans_curv [in]	軸群組新的路徑過渡模式曲率參數，請參閱 8.1.5 節。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.3

8.3.27 GetGrpCmdNum

用途

取得軸群組在命令緩衝區的命令數量。

語法

```
int GetGrpCmdNum(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

軸群組在命令緩衝區的命令數量。

需求版本

最低支援版本	iA Studio 1.3

8.3.28 SetGrpVelScale

用途

設置軸群組運動的速度百分比。

語法

```
int SetGrpVelScale(
    int    group_id,
    double vel_scale
);
```

參數

group_id [in] 軸群組編號。

vel_scale [in] 軸群組運動的新速度百分比。
輸入範圍：0 ~ 100

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.4

8.3.29 GetGrpVelScale

用途

取得軸群組運動的速度百分比。

語法

```
double GetGrpVelScale(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

軸群組運動的速度百分比，數值範圍為 0 ~ 100。

需求版本

最低支援版本	iA Studio 1.4

8.3.30 GetGrpCoordTrans

用途

取得軸群組座標系統的轉換參數。

語法

```
int GetGrpCoordTrans(
    int group_id,
    int coord_sys,
    double *trans_param
);
```

參數

group_id [in]	軸群組編號。
coord_sys [in]	座標系統，請參閱 8.1.2 節。
trans_param [out]	指向六元素陣列的指標，內含 6-DOF { X, Y, Z, A, B, C } 中的轉換參數。 參數單位：X, Y, Z 為 mm (毫米) ; A, B, C 為 deg (角度)。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 2.0

8.3.31 SetGrpCoordTrans

用途

設置軸群組座標系統的轉換參數。

語法

```
int SetGrpCoordTrans(
    int group_id,
    int coord_sys,
    double *trans_param
);
```

參數

group_id [in]	軸群組編號。
coord_sys [in]	座標系統，請參閱 8.1.2 節。
trans_param [in]	指向六元素陣列的指標，內含 6-DOF { X, Y, Z, A, B, C } 中的轉換參數。 參數單位：X, Y, Z 為 mm (毫米) ; A, B, C 為 deg (角度)。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 2.0

8.3.32 GetGrpPoseCmd

用途

取得軸群組座標系統的姿態命令。

語法

```
int GetGrpPoseCmd(
    int group_id,
    int coord_sys,
    double *pose_cmd
);
```

參數

group_id [in]	軸群組編號。
coord_sys [in]	座標系統，請參閱 8.1.2 節。
pose_cmd [out]	指向六元素陣列的指標，內含 6-DOF { X, Y, Z, A, B, C } 中的姿態命令。 參數單位：X, Y, Z 為 mm（毫米）; A, B, C 為 deg（角度）。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 2.0

8.3.33 GetGrpPoseFb

用途

取得軸群組座標系統的姿態回授。

語法

```
int GetGrpPoseFb(
    int group_id,
    int coord_sys,
    double *pose_fb
);
```

參數

group_id [in]	軸群組編號。
coord_sys [in]	座標系統，請參閱 8.1.2 節。
pose_fb [out]	指向六元素陣列的指標，內含 6-DOF { X, Y, Z, A, B, C } 中的姿態回授。 參數單位：X, Y, Z 為 mm（毫米）; A, B, C 為 deg（角度）。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 2.0

8.4 軸群組狀態

8.4.1 IsGrpEnabled

用途

詢問軸群組的致能狀態。

語法

```
int IsGrpEnabled(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若已致能軸群組，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

8.4.2 IsGrpMoving

用途

詢問軸群組的移動狀態。若軸群組正在移動，軌跡規劃器（PG）會持續輸出新的位置。

語法

```
int IsGrpMoving(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若軸群組正在移動，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

8.4.3 IsGrpInPos

用途

詢問軸群組的到位狀態。若軸群組已到位，軸群組底下的各軸皆到位。

語法

```
int IsGrpInPos(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若軸群組已到位，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.24

8.4.4 IsGrpErrorStop

用途

詢問軸群組是否處於 error stop 狀態。

語法

```
int IsGrpErrorStop(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若軸群組處於 error stop 狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 1.3

8.5 進階軸群組運動控制

8.5.1 LineAbs

用途

命令一個軸群組，內插線性移動至特定座標系統中的絕對位置。

語法

```
int LineAbs(
    int group_id,
    double *target_pos,
    double *motion_profile,
    int coord_sys,
    int buff_mode,
    int trans_mode,
    double *trans_par
);
```

參數

group_id [in] 軸群組編號。

target_pos [in] 指向六元素陣列的指標，內含 6-DOF { X, Y, Z, A, B, C } 中的絕對目標位置和方向。
參數單位：X, Y, Z 為 mm（毫米）; A, B, C 為 deg（角度）。

motion_profile [in] 指向四元素陣列的指標，內含路徑上 TCP 的最大切線方向運動。
{max_velocity, max_acceleration, max_deceleration, smooth_time}
請參閱 8.3.6 節 SetGrpMotionProfile。

coord_sys [in] 指定合適的座標系統，請參閱 8.1.2 節。

buff_mode [in] 指定路徑緩衝模式，請參閱 8.1.4 節。

trans_mode [in] 指定路徑過渡模式，請參閱 8.1.5 節。

trans_par [in] 指定特定過渡模式的指標，請參閱 8.1.5 節。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

8.5.2 LineRel

用途

命令一個軸群組，內插線性移動至特定座標系統中的相對位置。

語法

```
int LineRel(
    int group_id,
    double *relative_dist,
    double *motion_profile,
    int coord_sys,
    int buff_mode,
    int trans_mode,
    double *trans_par
);
```

參數

group_id [in] 軸群組編號。

relative_dist [in] 指向六元素陣列的指標，內含 6-DOF { X, Y, Z, A, B, C } 中的相對距離。
參數單位：X, Y, Z 為 mm (毫米)；A, B, C 為 deg (角度)。

motion_profile [in] 指向四元素陣列的指標，內含路徑上 TCP 的最大切線方向運動。
{max_velocity, max_acceleration, max_deceleration, smooth_time}
請參閱 8.3.6 節 SetGrpMotionProfile。

coord_sys [in] 指定合適的座標系統，請參閱 8.1.2 節。

buff_mode [in] 指定路徑緩衝模式，請參閱 8.1.4 節。

trans_mode [in] 指定路徑過渡模式，請參閱 8.1.5 節。

trans_par [in] 指定特定過渡模式的指標，請參閱 8.1.5 節。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

8.5.3 CircleAbs

用途

命令一個軸群組，內插圓周運動至特定座標系統中的絕對位置。

語法

```
int CircleAbs(
    int group_id,
    double *center_pos,
    double *normal_vector,
    int turns,
    double *target_pos,
    double *motion_profile,
    int coord_sys,
    int buff_mode,
    int trans_mode,
    double *trans_par
);
```

參數

group_id [in] 軸群組編號。

center_pos [in] 指向三元素陣列的指標，內含圓周運動中的絕對中心點 { X, Y, Z }。
參數單位：mm（毫米）

normal_vector [in] 指向三元素陣列的指標，內含以右手規則旋轉的法向量 { X, Y, Z }。
參數單位：mm（毫米）

turns [in] 相對於起點的圓周運動圈數，決定了圓周運動的方向及總角度。

target_pos [in] 指向六元素陣列的指標，內含 6-DOF { X, Y, Z, A, B, C } 中的絕對目標位置和方向。
參數單位：X, Y, Z 為 mm（毫米）; A, B, C 為 deg（角度）。

motion_profile [in] 指向四元素陣列的指標，內含路徑上 TCP 的最大切線方向運動。
{max_velocity, max_acceleration, max_deceleration, smooth_time}
請參閱 8.3.6 節 SetGrpMotionProfile。

coord_sys [in] 指定合適的座標系統，請參閱 8.1.2 節。

buff_mode [in] 指定路徑緩衝模式，請參閱 8.1.4 節。

trans_mode [in] 指定路徑過渡模式，請參閱 8.1.5 節。

trans_par [in] 指定特定過渡模式的指標，請參閱 8.1.5 節。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

8.5.4 CircleRel

用途

命令一個軸群組，內插圓周運動至特定座標系統中的相對位置。

語法

```
int CircleRel(
    int group_id,
    double *center_pos,
    double *normal_vector,
    int turns,
    double *relative_dist,
    double *motion_profile,
    int coord_sys,
    int buff_mode,
    int trans_mode,
    double *trans_par
);
```

參數

group_id [in] 軸群組編號。

center_pos [in] 指向三元素陣列的指標，內含圓周運動中的絕對中心點 { X, Y, Z }。
參數單位：mm（毫米）

normal_vector [in] 指向三元素陣列的指標，內含以右手規則旋轉的法向量 { X, Y, Z }。
參數單位：mm（毫米）

turns [in] 相對於起點的圓周運動圈數，決定了圓周運動的方向及總角度。

relative_dist [in] 指向六元素陣列的指標，內含 6-DOF { X, Y, Z, A, B, C } 中的相對距離。
參數單位：X, Y, Z 為 mm（毫米）; A, B, C 為 deg（角度）。

motion_profile [in] 指向四元素陣列的指標，內含路徑上 TCP 的最大切線方向運動。
{max_velocity, max_acceleration, max_deceleration, smooth_time}
請參閱 8.3.6 節 SetGrpMotionProfile。

coord_sys [in] 指定合適的座標系統，請參閱 8.1.2 節。

buff_mode [in] 指定路徑緩衝模式，請參閱 8.1.4 節。

trans_mode [in] 指定路徑過渡模式，請參閱 8.1.5 節。

trans_par [in] 指定特定過渡模式的指標，請參閱 8.1.5 節。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

9. GPIO 函式

9.1 概述

HIMC 提供各 8 組通用輸入 / 輸出 (GPIO) 腳位，硬體延遲時間 1ms 以內，24V。從站裝置可透過 CoE 通訊連結控制器並更新從站上的 IO 狀態，IO 的數量依從站裝置而定。使用者可利用本章提供的函式，如 SetGPO 與 SetSlvGPO 來分別設定 HIMC 與從站輸出腳位的訊號，也可以詢問輸入 / 輸出腳位的訊號狀態。另可利用 iA Studio 功能模組 Digital IO (請參閱《iA Studio 軟體操作手冊》4.4 節) 來觀察與設定 HIMC 與從站的輸入 / 輸出狀態。

HIMC 的數位輸入 I8 為 E-Stop 的訊號源 (請參閱《HIMC 安裝指南》3.3 節)，接收到正緣訊號時會被觸發，此時所有軸都會被解激磁，且所有 HMPL task 都會被停止執行。

註：觸發正緣訊號後，使用者可重新激磁軸或重新執行 HMPL task。

9.1.1 控制器 GPIO 變數

使用者可利用 iA Studio 的 Scope Manager (請參閱《iA Studio 軟體使用手冊》4.8 節) 選擇欲觀測的控制器通用輸入 / 輸出系統變數。詳細說明如表 9.1.1.1。

表 9.1.1.1 控制器通用輸入 / 輸出變數

名稱	變數	單位	描述
HIMC GPO	himc_gpo_bits	無	控制器通用輸出腳位狀態。
HIMC GPI	himc_gpi_bits	無	控制器通用輸入腳位狀態。

9.1.2 範例

範例 1

利用控制器第 4 個通用輸入腳位，當偵測到輸入的正緣訊號時，解激磁全部的軸，並將控制器第 3 個通用輸出腳位狀態設置為 1。

```
void main() {
    int last_state = 0;
    while (1) {
        int in = GetGPI(4);
        //  偵測控制器第 4 個通用輸入腳位訊號
        if ( (in ^ last_state) & in) {
            DisableAll();  //  解激磁全部的軸
            SetGPO(3, 1);  //  將控制器第 3 個通用輸出腳位狀態設置為 1
        }
        last_state = in;   //  記錄控制器通用輸入的狀態
    }
}
```

範例 2

利用從站 1 第 3 個通用輸入腳位，當偵測到輸入的負緣訊號時，切換從站 1 第 2 個通用輸出腳位狀態，並將控制器所有的通用輸出腳位狀態設置為 1。

```
void main() {
    int last_state = 0;
    while (1) {
        int in = GetSlvGPI(1, 3);
        //  偵測從站 1 第 3 個通用輸入腳位訊號
        if ( (in ^ last_state) && !in) {
            ToggleSlvGPO(1, 2);  //  切換從站 1 第 2 個通用輸出腳位狀態
            SetAllGPO(0xff);     //  將控制器所有的通用輸出腳位狀態設置為 1
        }
        last_state = in;  //  記錄從站通用輸入的狀態
    }
}
```

範例 3

設定從站 1 上插槽 2 的第 3 個通用輸出腳位，切換從站其腳位輸出狀態。

```
void main() {
    int last_state = 0;
    while (1) {
        SetSlvGPO(1 | HMPL_SLOT_2, 3);
        //  設定從站 1 上插槽 2 的第 3 個通用輸出腳位
        Sleep(1000);
        //  等待 1 秒
        ToggleSlvGPO(1 | HMPL_SLOT_2, 3);
        //  切換從站 1 上插槽 2 的第 3 個通用輸出腳位
    }
}
```

9.2 控制器 IO 設定

9.2.1 SetGPO

用途

設置控制器通用輸出的狀態。

語法

```
int SetGPO(
    int gpo_idx,
    int on_off
);
```

參數

gpo_idx [in]　　通用輸出編號。

on_off [in]　　欲設置的狀態。1 為導通狀態，0 為不導通狀態。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳 **int** 型態的值**-1**。

需求版本

最低支援版本	iA Studio 1.1

9.2.2 ToggleGPO

用途

切換控制器通用輸出的狀態。

語法

```
int ToggleGPO(
    int gpo_idx
);
```

參數

gpo_idx [in]　　通用輸出編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳 **int** 型態的值**-1**。

需求版本

最低支援版本	iA Studio 1.1

9.2.3 SetAllGPO

用途

切換控制器所有通用輸出的狀態。

語法

```
int SetAllGPO(
    int all_gpo_state
);
```

參數

all_gpo_state [in]　　所有通用輸出的狀態值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳 **int** 型態的值**-1**。

需求版本

最低支援版本	iA Studio 1.3

9.2.4 SetGPIInvert

用途

設置控制器通用輸入的反相狀態。

語法

```
int SetGPIInvert(
    int gpi_idx,
    int invert
);
```

參數

gpi_idx [in] 通用輸入編號。

invert [in] 欲設置的反相狀態。1 為反相狀態，0 為不反相狀態。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳 **int** 型態的值**-1**。

需求版本

最低支援版本	iA Studio 2.0

9.2.5 SetGPOInvert

用途

設置控制器通用輸出的反相狀態。

語法

```
int SetGPOInvert(
    int gpo_idx,
    int invert
);
```

參數

gpo_idx [in] 通用輸出編號。

invert [in] 欲設置的反相狀態。1 為反相狀態，0 為不反相狀態。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳 **int** 型態的值**-1**。

需求版本

最低支援版本	iA Studio 2.0

9.2.6 BindEMO

用途

設置欲綁定 E-Stop 的通用輸入腳位。

語法

```
int BindEMO(
    int gpi_idx
);
```

參數

gpi_idx [in] 通用輸入編號，預設值為 8。
若設為 0，則所有通用輸入腳位皆不綁定 E-Stop。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳 **int** 型態的值**-1**。

需求版本

最低支援版本	iA Studio 2.0

9.3 從站 IO 設定

9.3.1 SetSlvGPO

用途

設置從站通用輸出的狀態。

語法

```
int SetSlvGPO(
    int slv_slot_id,
    int gpo_idx,
    int on_off
);
```

參數

slv_slot_id [in] 從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

gpo_idx [in] 通用輸出編號。

on_off [in] 欲設置的狀態。1 為導通狀態，0 為不導通狀態。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳 **int** 型態的值**-1**。

備註

使用此函式需將 Digital output 物件配置為 PDO，例如驅動器需設定 0x60FE(Digital outputs)為 PDO。

需求版本

最低支援版本	iA Studio 1.1

9.3.2 ToggleSlvGPO

用途

切換從站通用輸出的狀態。

語法

```
int ToggleSlvGPO(
    int slv_slot_id,
    int gpo_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

gpo_idx [in]　通用輸出編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳 **int** 型態的值**-1**。

備註

使用此函式需將 Digital output 物件配置為 PDO，例如驅動器需設定 0x60FE(Digital outputs)為 PDO。

需求版本

最低支援版本	iA Studio 1.1

9.3.3 SetSlvAllGPO

用途

切換從站所有通用輸出的狀態。

語法

```
int SetSlvAllGPO(
    int slv_slot_id,
    int all_gpo_state
);
```

參數

slv_slot_id [in] 從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

all_gpo_state [in] 所有通用輸出的狀態值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳 **int** 型態的值**-1**。

備註

使用此函式需將 Digital output 物件配置為 PDO，例如驅動器需設定 0x60FE(Digital outputs)為 PDO。

需求版本

最低支援版本	iA Studio 2.0

9.4 控制器 IO 狀態

9.4.1 GetGPI

用途

詢問控制器通用輸入的狀態。

語法

```
int GetGPI(
    int gpi_idx
);
```

參數

gpi_idx [in]　　通用輸入編號。

回傳值

若輸入為導通狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 3.0

9.4.2 GetGPO

用途

詢問控制器通用輸出的狀態。

語法

```
int GetGPO(
    int gpo_idx
);
```

參數

gpo_idx [in]　　通用輸出編號。

回傳值

若輸出為導通狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 3.0

9.4.3 GetAllGPI

用途

取得控制器所有通用輸入的狀態。

語法

```
int GetAllGPI();
```

參數

無

回傳值

控制器所有通用輸入的狀態值。

若第 1 與第 4 個 GPI 腳位為 TRUE，則回傳值為 9。

需求版本

最低支援版本	iA Studio 1.3

9.4.4 GetAllGPO

用途

取得控制器所有通用輸出的狀態。

語法

```
int GetAllGPO();
```

參數

無

回傳值

控制器所有通用輸出的狀態值。

若第 2 與第 3 個 GPO 腳位為 TRUE，則回傳值為 6。

需求版本

最低支援版本	iA Studio 1.3

9.4.5 GetAllGPIInvertSt

用途

取得控制器所有通用輸入的反相狀態。

語法

```
int GetAllGPIInvertSt();
```

參數

無

回傳值

控制器所有通用輸入的反相狀態值。

若第 1 與第 4 個 GPI 反相腳位為 TRUE，則回傳值為 9。

需求版本

最低支援版本	iA Studio 2.0

9.4.6 GetAllGPOInvertSt

用途

取得控制器所有通用輸出的反相狀態。

語法

```
int GetAllGPOInvertSt();
```

參數

無

回傳值

控制器所有通用輸出的反相狀態值。

若第 2 與第 3 個 GPO 反相腳位為 TRUE，則回傳值為 6。

需求版本

最低支援版本	iA Studio 2.0

9.5 從站 IO 狀態

9.5.1 GetSlvGPI

用途

詢問從站通用輸入的狀態。

語法

```
int GetSlvGPI(
    int slv_slot_id,
    int gpi_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

gpi_idx [in]　通用輸入編號。

回傳值

若輸入為導通狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

備註

使用此函式需將 Digital input 物件配置為 PDO，例如驅動器需設定 0x60FD(Digital inputs)為 PDO。

需求版本

最低支援版本	iA Studio 3.0

9.5.2 GetSlvGPO

用途

詢問從站通用輸出的狀態。

語法

```
int GetSlvGPO(
    int slv_slot_id,
    int gpo_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

gpo_idx [in]　通用輸出編號。

回傳值

若輸入為導通狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

備註

使用此函式需將 Digital output 物件配置為 PDO，例如驅動器需設定 0x60FE(Digital outputs)為 PDO。

需求版本

最低支援版本	iA Studio 3.0

(此頁有意留白。)

10. AIO 函式

10.1 概述

利用 AIO 函式，具類比輸入（AI）或類比輸出（AO）功能的從站可讀取和設定相關參數。其中 HMPL 提供使用者指定數位類比轉換形式設定，說明如表 10.1。

表 10.1.1 類比資料轉換形式定義

轉換形式	轉換說明	描述	附註
HMPL_DAC_NONE	無轉換	此轉換形式只能使用 Hex 函式操作。	-
HMPL_DAC_N10_P10	-10~10	將類比 Hex 值轉換為-10~10。	常見於類比電壓模組
HMPL_DAC_P0_P10	0~10	將類比 Hex 值轉換為 0~10。	
HMPL_DAC_N5_P5	-5~5	將類比 Hex 值轉換為-5~10。	
HMPL_DAC_P0_P5	0~5	將類比 Hex 值轉換為-0~10。	
HMPL_DAC_P0_P20	0~20	將類比 Hex 值轉換為 0~20。	常見於類比電流模組
HMPL_DAC_P4_P20	4~20	將類比 Hex 值轉換為 4~20。	
HMPL_DAC_N20_P20	-20~20	將類比 Hex 值轉換為-20~20。	
HMPL_DAC_SIGNED	高位元為正負號定義	高位元 0 為正數，1 為負數。	-

10.1.1 範例

範例 1

設定與讀取類比輸出 Hex 值。

```
void main() {
    int slv_id = 0;
    int ao_index = 1;          //  類比輸出通道位置
    int ao_hex_val = 0xFFFF;  //  類比 Hex 輸出：0xFFFF = 65535
    SetSlvAOHex(slv_id, ao_index, ao_hex_val );
    Print("%l", GetSlvAOHex(slv_id, ao_index));
}
```

範例 2

讀取類比輸入 Hex 值。

```
void main() {
    int slv_id = 1;
    int ai_index = 1;          //  類比輸入通道位置
    Print("%l", GetSlvAIHex(slv_id, ai_index));
}
```

範例 3

設定類比輸入/輸出轉換形式。

```
void main() {
    int slv_id_ao = 1;
    int ao_index = 1;          //  類比輸出通道位置
    int slv_id_ai = 2;
    int ai_index = 1;          //  類比輸入通道位置

    SetSlvAOType(slv_id_ao, ao_index, HMPL_DAC_N10_P10 );
    //  設定類比輸出轉換為-10~10
    SetSlvAO(slv_id_ao, ao_index, 10)
    //  設定類比輸出 10
    SetSlvAIType(slv_id_ai, ai_index, HMPL_DAC_N10_P10 );
    //  設定類比輸入轉換為-10~10
    Print("%f", GetSlvAI(slv_id, ai_index));
}
```

10.2 從站 AIO 設定

10.2.1 SetSlvAIType

用途

設置從站的類比輸入值轉換形式。

語法

```
int SetSlvAIType(
    int slv_slot_id,
    int ai_idx,
    int range_type
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。
ai_idx [in]　類比輸入通道位置。
range_type [in]　類比資料轉換形式，參考表 10.1。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

10.2.2 SetSlvAOType

用途

設置從站的類比輸出值轉換形式。

語法

```
int SetSlvAOType(
    int slv_slot_id,
    int ao_idx,
    int range_type
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ao_idx [in]　類比輸出通道位置。

range_type [in]　類比資料轉換形式，參考表 10.1。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

10.2.3 SetSlvAOHex

用途

設置從站的類比輸出 Hex 值。

語法

```
int SetSlvAOHex(
    int slv_slot_id,
    int ao_idx,
    long long ao_hex_val
);
```

參數

slv_slot_id [in] 從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ao_idx [in] 類比輸出通道位置。

ao_hex_val [in] 類比 64bit 輸出值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將 Analog output 物件配置為 PDO。

需求版本

最低支援版本	iA Studio 3.0

10.2.4 SetSlvAO

用途

設置從站的類比輸出值。

語法

```
int SetSlvAO(
    int slv_slot_id,
    int ao_idx,
    double ao_val
);
```

參數

slv_slot_id [in] 從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ao_idx [in] 類比輸出通道位置。

ao_val [in] 類比輸出值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將 Analog output 物件配置為 PDO。

需求版本

最低支援版本	iA Studio 3.0

10.3 從站 AIO 狀態

10.3.1 GetSlvAIType

用途

取得從站的類比輸入類型。

語法

```
int GetSlvAIType(
    int slv_slot_id,
    int ai_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ai_idx [in]　類比輸入通道位置。

回傳值

從站的類比輸入類型，請參考表 10.1。

需求版本

最低支援版本	iA Studio 3.0

10.3.2 GetSlvAOType

用途

取得從站的類比輸出類型。

語法

```
int GetSlvAOType(
    int slv_slot_id,
    int ao_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ao_idx [in]　類比輸出通道位置。

回傳值

從站的類比輸出類型，請參考表 10.1。

需求版本

最低支援版本	iA Studio 3.0

10.3.3 GetSlvAIHex

用途

取得從站類比輸入 Hex 值。

語法

```
int GetSlvAIHex(
    int slv_slot_id,
    int ai_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。
ai_idx [in]　類比輸入通道位置。

回傳值

類比輸入 Hex 值。

備註

使用此函式需將 Analog input 物件配置為 PDO。

需求版本

最低支援版本	iA Studio 3.0

10.3.4 GetSlvAI

用途

取得從站類比輸入值。

語法

```
double GetSlvAI(
    int slv_slot_id,
    int ai_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。
ai_idx [in]　類比輸入通道位置。

回傳值

類比輸入值。

備註

使用此函式需將 Analog input 物件配置為 PDO。

需求版本

最低支援版本	iA Studio 2.0

10.3.5 GetSlvAOHex

用途

取得從站的類比輸出 Hex 值。

語法

```
int GetSlvAOHex(
    int slv_slot_id,
    int ao_idx
);
```

參數

slv_slot_id [in] 從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ao_idx [in] 類比輸出通道位置。

回傳值

類比輸出 Hex 值。

備註

使用此函式需將 Analog output 物件配置為 PDO。

需求版本

最低支援版本	iA Studio 3.0

10.3.6 GetSlvAO

用途

取得從站的類比輸出值。

語法

```
double GetSlvAO(
    int slv_slot_id,
    int ao_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ao_idx [in]　類比輸出通道位置。

回傳值

類比輸出值。

備註

使用此函式需將 Analog output 物件配置為 PDO。

需求版本

最低支援版本	iA Studio 2.0

10.4 從站 AO 綁定 HIMC 內部記憶體變數

10.4.1 SetSlvAOMonitor

用途

設置欲與類比輸出綁定的控制器變數。

語法

```
int SetSlvAOMonitor(
    int slv_slot_id,
    int ao_idx,
    int var_id
);
```

參數

slv_slot_id [in] 從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ao_idx [in] 類比輸出通道位置。

var_id [in] 控制器變數與軸 ID。

控制器變數與軸 ID 的定義如下表，更多的控制器變數請參閱 17.1.1.1 節。

範例：HMPL_AXIS_0 | HMPL_REF_VEL

控制器變數	定義說明	控制器變數	定義說明
HMPL_REF_POS	軸參考位置	HMPL_POS_FB	軸位置回授
HMPL_REF_VEL	軸參考速度	HMPL_VEL_FB	軸速度回授
HMPL_REF_ACC	軸參考加速度	HMPL_ACC_FB	軸加速度回授
		HMPL_CUR_FB	軸電流回授

軸 ID	定義說明
HMPL_AXIS_0	軸 0
HMPL_AXIS_1	軸 1
…	…
HMPL_AXIS_15	軸 15

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 2.0

10.4.2 SetSlvAOParam

用途

設置從站的類比輸出轉換控制器參數。

語法

```
int SetSlvAOParam(
    int slv_slot_id,
    int ao_idx,
    int ao_en_bind,
    double ao_scale,
    double ao_offset
);
```

參數

slv_slot_id [in]	從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。
ao_idx [in]	類比輸出通道位置。
ao_en_bind [in]	0：關閉類比輸出綁定控制器變數的功能（預設值）
	1：開啟類比輸出綁定控制器變數的功能
ao_scale [in]	類比輸出控制器變數的比例增益。
ao_offset [in]	類比輸出控制器變數的偏移量。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 2.0

10.4.3 GetSlvAOScale

用途

取得從站類比輸出與控制器變數的比例增益。

語法

```
double GetSlvAOScale(
    int slv_slot_id,
    int ao_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ao_idx [in]　類比輸出通道位置。

回傳值

類比輸出控制器變數的比例增益。

需求版本

最低支援版本	iA Studio 2.0

10.4.4 GetSlvAOOffset

用途

取得從站類比輸出與控制器變數的偏移量。

語法

```
double GetSlvAOOffset(
    int slv_slot_id,
    int ao_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ao_idx [in]　類比輸出通道位置。

回傳值

類比輸出控制器變數的偏移量。

需求版本

最低支援版本	iA Studio 2.0

10.4.5 IsSlvAOBound

用途

詢問從站的類比輸出是否綁定控制器變數。

語法

```
int IsSlvAOBound(
    int slv_slot_id,
    int ao_idx
);
```

參數

slv_slot_id [in]　從站 ID 與其插槽 ID，若從站無插槽則可忽略插槽 ID。

ao_idx [in]　類比輸出通道位置。

回傳值

若從站的類比輸出綁定控制器變數，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 2.0

(此頁有意留白。)

11. User Table 函式

11.1 概述

HIMC 提供使用者可自由使用的記憶體空間，最多可儲存 512,000 個 double 型態的變數資料(500K Bytes)。使用者可利用本章提供的函式存取記憶體空間，寫入的值會存放在控制器的隨機存取記憶體（RAM）中。透過 SaveUserTable 函式，將 User Table 在記憶體空間的資料儲存到 HIMC 的硬碟空間中；在 HIMC 斷電重開之後，透過 LoadUserTable 函式，將保存的資料重新複製到 User Table 的記憶體空間。

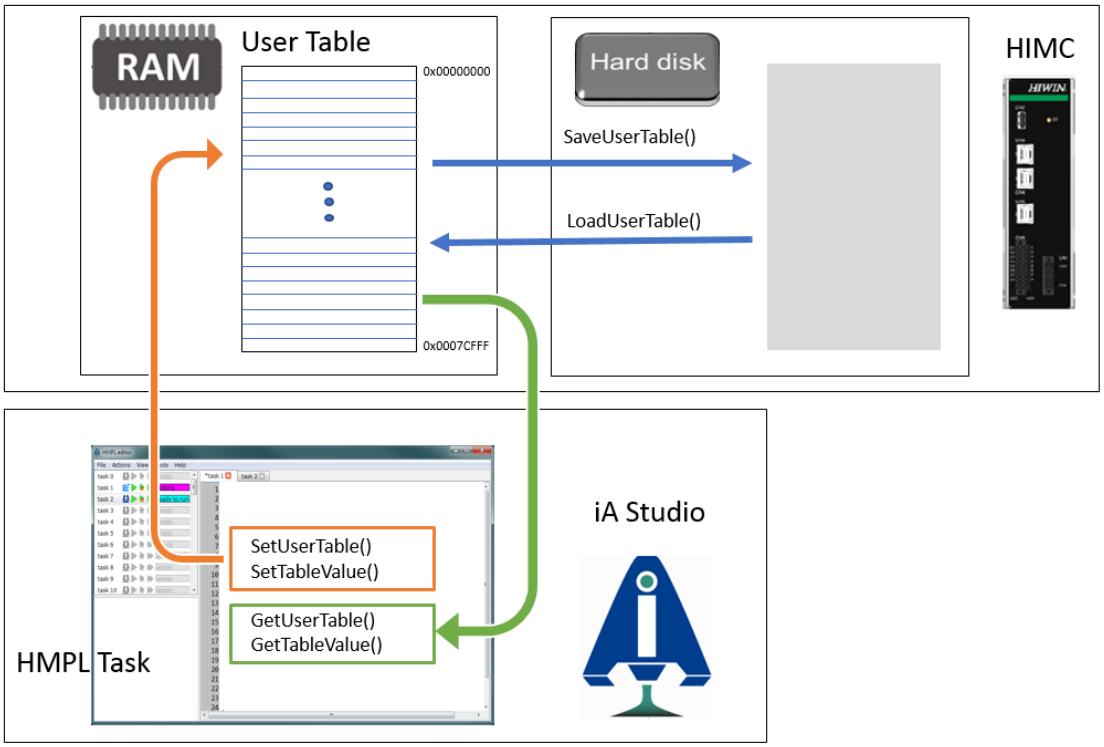


圖 11.1.1

註：使用者可利用 iA Studio 的 Table Viewer (請參閱《iA Studio 軟體使用手冊》4.11 節) 來存取 User Table 的變數值，包含載入與存入 HIMC 的記憶體與硬碟之中。

注意：

動態誤差補償函式所使用的誤差補償表 (Error map) 為儲存在 User Table 的記憶體空間。啟動動態誤差補償時，使用者需自行確保其他 User Table 數值的存取不會影響到所建立的誤差補償值。

11.2 SetUserTable

用途

將數據寫入 User Table。

語法

```
int SetUserTable(
    int    start_idx,
    int    num_data,
    double *input
);
```

參數

start_idx [in]　User Table 的起始編號。

num_data [in]　元素的數量。

input [in]　指向陣列的指標，內含輸入數據。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
int main() {
    //  將數據寫入 User Table
    double data[5] = {-2.0, 0.0, 2.0, 6.0, 4.0};
    SetUserTable(
      1,  //  User Table 的起始編號
      5,  //  元素的數量
      data  //  指向輸入數據陣列的指標
    );
    //  以上文字與下方內容相同
    system_user_table[1] = -2.0;
    system_user_table[2] = 0.0;
    system_user_table[3] = 2.0;
    system_user_table[4] = 6.0;
    system_user_table[5] = 4.0;
}
```

需求版本

最低支援版本	iA Studio 0.23

11.3 GetUserTable

用途

取回 User Table 數據。

語法

```
int GetUserTable(
    int    start_idx,
    int    num_data,
    double *output
);
```

參數

start_idx [in]　User Table 的起始編號。

num_data [in]　元素的數量。

output [out]　指向陣列的指標，內含輸出數據。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
int main() {
    //  將數據寫入 User Table
    double input[5] = {-2.0, 0.0, 2.0, 6.0, 4.0};
    SetUserTable(
      1,  //  User Table 的起始編號
      5,  //  元素的數量
      input  //  指向輸入數據陣列的指標
    );
    //  現在 User Table 裡有"-2.0"、"0.0"、"2.0"、"6.0"、"4.0"這些值
    //  從 index 1 開始

    //  讀取 User Table
    double output[3];
    GetUserTable(
      3,  //  User Table 的起始編號
      3,  //  元素的數量
      output  //  指向輸出數據陣列的指標
    );
    //  現在 output[0]為 2.0;
    //  output[1]為 6.0;
    //  output[2]為 4.0;
}
```

需求版本

最低支援版本	iA Studio 0.23

11.4 SetTableValue

用途

將數據寫入 User Table 的特定編號中。

語法

```
int SetTableValue(
    int    index,
    double value
);
```

參數

index [in] User Table 的編號。

value [in] 輸入數據。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.1

11.5 GetTableValue

用途

從 User Table 的特定編號中取得數據。

語法

```
double GetTableValue(
    int index
);
```

參數

index [in] User Table 的編號。

回傳值

特定編號中的數據。

需求版本

最低支援版本	iA Studio 1.1

11.6 SaveUserTable

用途

將在 RAM 中的 User Table 數據存入永久記憶體中。

語法

```
int SaveUserTable(
    int start_idx,
    int num_data
);
```

參數

start_idx [in]　User Table 的起始編號。

num_data [in]　被儲存元素的數量。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
int main() {
    //  將數據寫入 User Table
    system_user_table[3] = 2.0;
    system_user_table[4] = 6.0;
    system_user_table[5] = 4.0;

    SaveUserTable(
      3,  //  User Table 的起始編號
      3  //  元素的數量
    );
    //  重新啟動控制器
    //  system_user_table[3]的值為 2.0
    //  system_user_table[4]的值為 6.0
    //  system_user_table[5]的值為 4.0
}
```

需求版本

最低支援版本	iA Studio 0.23

11.7 LoadUserTable

用途

將永久記憶體中的 User Table 數據載入至 RAM。

語法

```
int LoadUserTable(
    int start_idx,
    int num_data
);
```

參數

start_idx [in]　User Table 的起始編號。

num_data [in]　被載入元素的數量。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
int main() {
    //  將數據寫入 User Table
    system_user_table[3] = 2.0;
    system_user_table[4] = 6.0;
    system_user_table[5] = 4.0;
    SaveUserTable(
      3,  //  User Table 的起始編號
      3  //  元素的數量
    );
    /* 於 table[3]、table[4]和 table[5]內填入任意數值 */
    system_user_table[3] = 999.0;
    system_user_table[4] = 777.0;
    system_user_table[5] = 888.0;
    LoadUserTable(3, 3);
    //  system_user_table[3]的值為 2.0
    //  system_user_table[4]的值為 6.0
    //  system_user_table[5]的值為 4.0
}
```

需求版本

最低支援版本	iA Studio 0.23

12. 位置觸發函式

12.1 概述

HIMC 位置觸發函式僅適用於搭配大銀驅動器，使用者可利用 HMPL 命令操作 PT (位置觸發) 相關功能。操作 PT 相關功能前，請向本公司或當地經銷商諮詢相容的驅動器。

註：大銀驅動器使用 PT 相關功能的條件為 (1) 須為數位式編碼器 (2) 須先執行完歸原點流程。

12.1.1 PT 變數

表 12.1.1.1 為操作 PT 相關功能的變數介紹。

表 12.1.1.1

名稱	型態	單位	描述	HMPL 函式
狀態	int	true / false	PT 功能的狀態，顯示 PT 是否仍在運行。	EnablePT DisablePT IsPTEnabled
起點	double	毫米 或 角度	PT 功能的起點。PT 輸出訊號序列從此點開始。	SetPT_StartPos
終點	double	毫米 或 角度	PT 功能的終點。此點後不再發送 PT 輸出訊號。	SetPT_EndPos
間距	double	毫米 或 角度	連續 PT 輸出的位置間距。	SetPT_Interval
脈波寬度	int	奈秒	每個 PT 輸出訊號的寬度。 範圍： 1. E1 系列驅動器為 20 奈秒至 80,000 奈秒，20 奈秒為最小增加單位。例如，20、40、... 80,000。	SetPT_PulseWidth

圖 12.1.1.1 中，極性被設定為 active high。

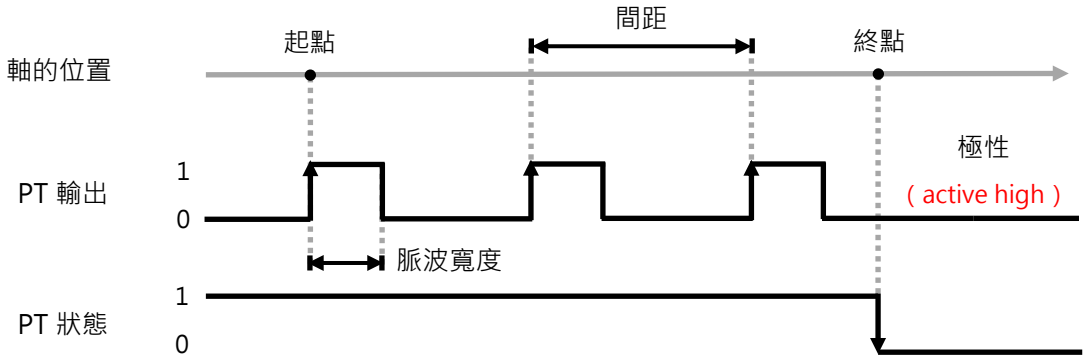


圖 12.1.1.1

圖 12.1.1.2 中，極性被設定為 active low。

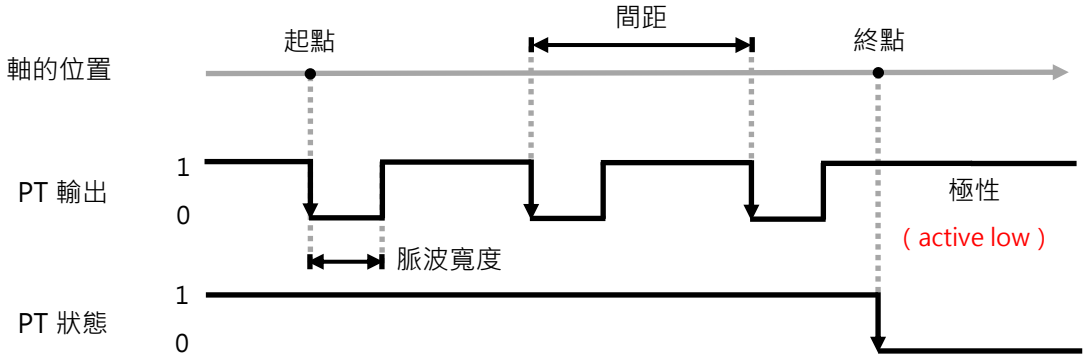


圖 12.1.1.2

限制條件：

PT 功能的間距與軸的移動速度須滿足『速度 < 間距 x 位置取樣頻率』。

若間距設定為 100 um，位置取樣頻率為 16 K，則移動速度須小於 1600 mm/s。

註：欲調整 PT 功能的輸出極性 (active high / low)，請至驅動器人機介面設定，儲存設定後斷電重開驅動器，使輸出極性生效。

12.1.2 PT 功能使用流程

◆ 等間距 PT 功能

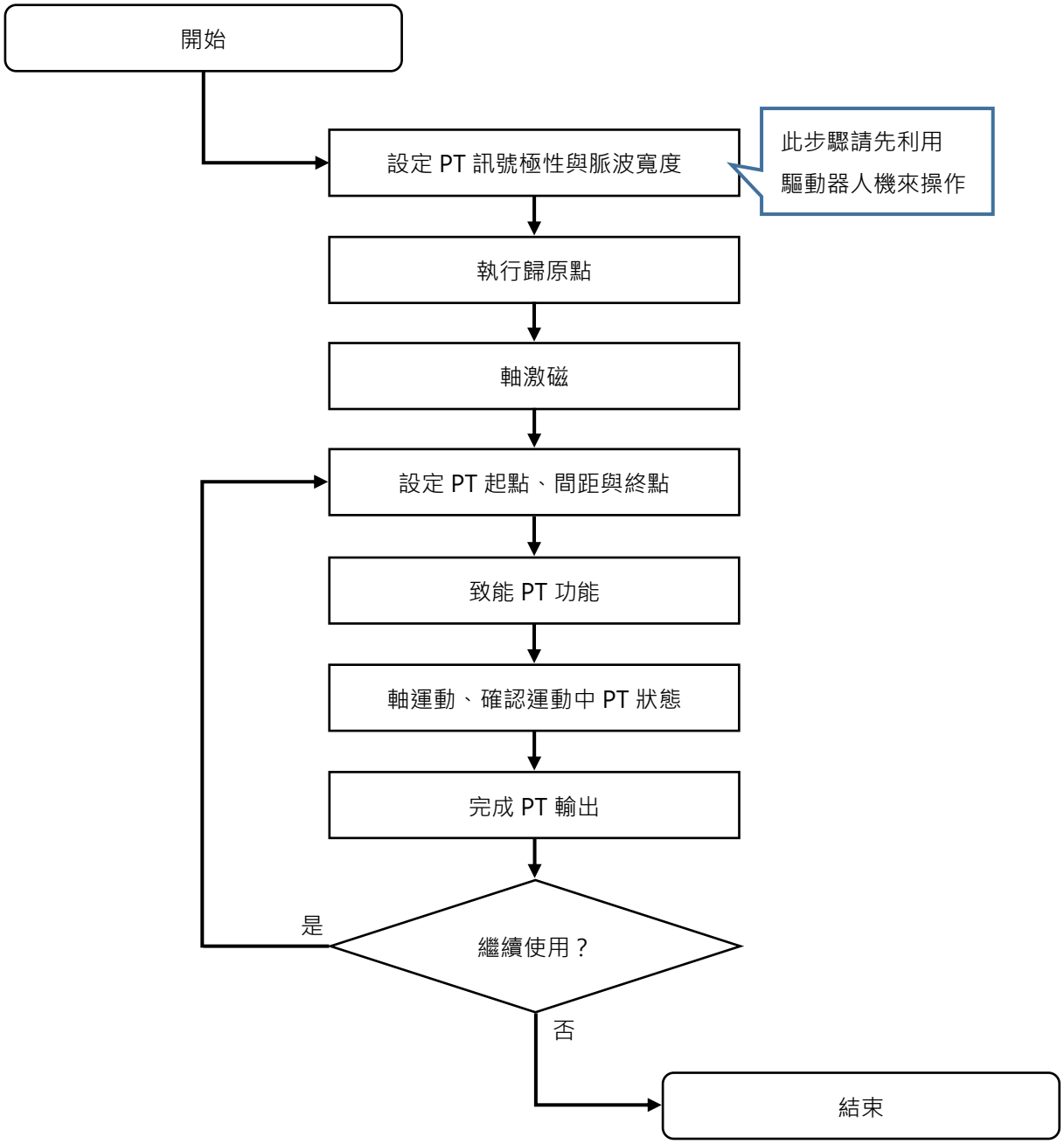


圖 12.1.2.1

12.1.3 範例

範例：等間距 PT 功能

```
void main()
{
    int axis_id = 1;
    double start_pos = 10;    //  PT功能的起點為 10 mm
    double end_pos = 20;      //  PT功能的終點為 20 mm
    double interval = 1;      //  PT功能的間距為 1 mm

    //  脈波寬度為 1000 ns
    SetPT_PulseWidth(axis_id, 1000);
    //  歸原點方法33 - 負方向找Index訊號
    Home(HOME_METHOD_33, axis_id, 0, 20, 0, 10000);
    //  激磁軸
    Enable(axis_id);
    Till(IsEnabled(axis_id));
    //  移動至 0 mm
    MoveAbs(axis_id, 0);
    Till(IsInPos(axis_id));

    // ------ 往正方向移動 ------
    //  設定PT功能的起點、間距與終點
    SetPT_StartPos(axis_id, start_pos);
    SetPT_Interval(axis_id, interval);
    SetPT_EndPos(axis_id, end_pos);

    //  致能PT功能
    EnablePT(axis_id);

    //  軸從 0 mm 移動至 30 mm
    MoveAbs(axis_id, 30);
    Till(IsInPos(axis_id));

    // ------ 往負方向移動 ------
    //  設定PT功能的起點與終點
    SetPT_StartPos(axis_id, end_pos);
```

```
    SetPT_EndPos(axis_id, start_pos);

    //  致能PT功能
    EnablePT(axis_id);

    //  軸從 30 mm 移動至 0 mm
    MoveAbs(axis_id, 0.0);
    Till(IsInPos(axis_id));
}
```

12.2 EnablePT

用途

致能軸的位置觸發功能。

語法

```
int EnablePT(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

iA Studio 1.2（含）以上支援 E1 系列驅動器設定。

需求版本

最低支援版本	iA Studio 0.23

12.3 DisablePT

用途

解致能軸的位置觸發功能。

語法

```
int DisablePT(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

iA Studio 1.2（含）以上支援 E1 系列驅動器設定。

需求版本

最低支援版本	iA Studio 0.23

12.4 IsPTEnabled

用途

詢問是否已致能位置觸發功能。

語法

```
int IsPTEnabled(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸處於 PT enabled 狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

備註

iA Studio 1.2（含）以上支援 E1 系列驅動器設定。

需求版本

最低支援版本	iA Studio 0.23

12.5 SetPT_StartPos

用途

設置位置觸發功能的起點。

語法

```
int SetPT_StartPos(
    int    axis_id,
    double start_pos
);
```

參數

axis_id [in]	軸編號。
start_pos [in]	PT 功能的起點。 參數單位：mm（毫米）或 deg（角度）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

iA Studio 1.2（含）以上支援 E1 系列驅動器設定。

需求版本

最低支援版本	iA Studio 0.23

12.6 SetPT_EndPos

用途

設置位置觸發功能的終點。

語法

```
int SetPT_EndPos(
    int    axis_id,
    double end_pos
);
```

參數

axis_id [in]	軸編號。
end_pos [in]	PT 功能的終點。 參數單位：mm（毫米）或 deg（角度）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

iA Studio 1.2（含）以上支援 E1 系列驅動器設定。

需求版本

最低支援版本	iA Studio 0.23

12.7 SetPT_Interval

用途

設置位置觸發功能的位置間距。

語法

```
int SetPT_Interval(
    int    axis_id,
    double interval
);
```

參數

axis_id [in]	軸編號。
interval [in]	連續 PT 輸出的位置間距。 參數單位：mm（毫米）或 deg（角度）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

iA Studio 1.2（含）以上支援 E1 系列驅動器設定。

需求版本

最低支援版本	iA Studio 0.23

12.8 SetPT_PulseWidth

用途

設置位置觸發功能的脈波寬度。

語法

```
int SetPT_PulseWidth(
    int axis_id,
    int width_ns
);
```

參數

axis_id [in]	軸編號。
width_ns [in]	每個 PT 輸出訊號的寬度。 參數單位：ns（奈秒）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

iA Studio 1.2（含）以上支援 E1 系列驅動器設定。

需求版本

最低支援版本	iA Studio 0.23

(此頁有意留白。)

13. Touch Probe 函式

13.1 概述

Touch Probe 函式為位置閂鎖（Latch）功能，透過編碼器輸入訊號的邊緣觸發來補捉編碼器的位置回授數值，用於歸原點程序中（適用於 AC、DM、LM 馬達）。如圖 13.1.1 所示，驅動器經過其編碼器 Index 訊號時，會觸發 Touch Probe 功能並記錄此 Index 訊號的位置。使用者可透過 Touch Probe 函式詢問控制器的 Touch Probe 功能是否已被觸發，控制器亦可獲得被 Touch Probe 閂鎖記錄的 Index 訊號位置。

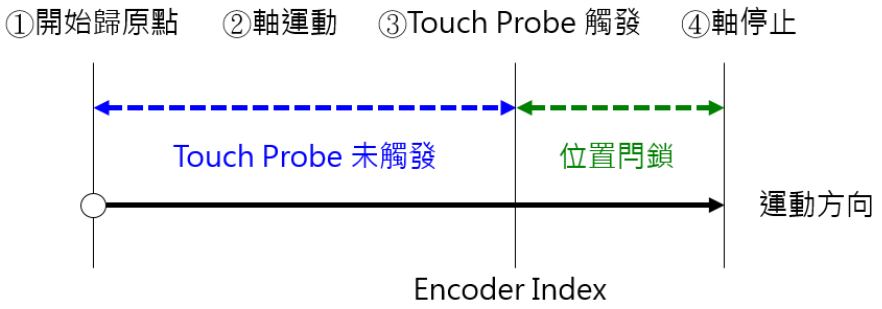


圖 13.1.1

13.2 EnableTouchProbe

用途

致能軸的 Touch Probe 功能。

語法

```
int EnableTouchProbe(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將物件 0x60B8(Touch probe function)配置為 PDO。

需求版本

最低支援版本	iA Studio 0.23

13.3 DisableTouchProbe

用途

解致能軸的 Touch Probe 功能。

語法

```
int DisableTouchProbe(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將物件 0x60B8(Touch probe function)配置為 PDO。

需求版本

最低支援版本	iA Studio 0.23

13.4 IsTouchProbeEnabled

用途

詢問是否已致能 Touch Probe 功能。

語法

```
int IsTouchProbeEnabled(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若軸處於 touch probe enabled 狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

備註

使用此函式需將物件 0x60B9(Touch probe status)配置為 PDO。

需求版本

最低支援版本	iA Studio 0.23

13.5 IsTouchProbeTriggered

用途

詢問是否已觸發 Touch Probe 功能。

語法

```
int IsTouchProbeTriggered(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸處於 touch probe enabled 狀態，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

備註

使用此函式需將物件 0x60B9(Touch probe status)配置為 PDO。

需求版本

最低支援版本	iA Studio 0.23

13.6 GetTouchProbePos

用途

取得軸的 touch probe 位置。

語法

```
int GetTouchProbePos(
    int    axis_id,
    double *output
);
```

參數

axis_id [in]	軸編號。
output [out]	指標型態的記憶體，用來儲存軸的 touch probe 位置。 參數單位：mm（毫米）或 deg（角度）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將物件對應的 Touch probe 物件配置為 PDO，例如 0x60BA(Touch probe 1 positive edge)。

需求版本

最低支援版本	iA Studio 0.23

13.7 SetTouchProbeFunc

用途

設定 touch probe 功能。

語法

```
int SetTouchProbeFunc(
    int    axis_id,
    int    tp_source,
    int    cont_trigger,
    int    detect_edge
);
```

參數

axis_id [in] 軸編號。

tp_source [in] Touch probe 來源。
輸入範圍：0(touch probe 1、預設值)、1(touch probe 2)。

cont_trigger [in] 觸發模式。
輸入範圍：0(單次觸發、預設值)、1(連續觸發)。

detect_edge [in] 邊緣偵測模式。
輸入範圍：0(正緣觸發、預設值)、1(負緣觸發)。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

14. 動態誤差補償函式

14.1 概述

HIMC 提供動態 1D / 2D / 3D 誤差補償功能，依據相關誤差量測與計算結果，使用者可建立誤差補償表並在 HIMC 進行設定。設定的參數包含被補償軸、參考軸、補償點的間距(Interval)、補償點的起始位置(Base)、補償點的數量 (Number) 與各補償點的補償值 (Value)；其中補償值的設定須使用 HIMC User Table 的記憶體空間，並提供誤差補償表的第一個 ID 位置於 User Table 中。

註 1：有關 User Table 的使用與詳細說明，請參閱第 11 章。
註 2：啟動動態誤差補償前，軸須先完成歸原點以固定被補償軸與參考軸的座標位置。

至於被補償軸與參考軸，可選擇同一軸，既是被補償軸，也是參考軸；亦可選擇多個不同的軸作為被補償軸的參考軸，如被補償軸為 Z 軸，參考軸為 X 和 Y 軸；而被補償軸的補償值會依據參考軸的運動位置而變化。建立的誤差補償表會在軸運動過程中，以線性插補的方式計算補償點之間的補償命令值，使運動過程中的補償值能夠保持連續，而不會產生命令跳躍的情形。而當軸位置超過補償表所建立的範圍時，HIMC 會使用最近補償點的補償值作為其補償命令，如圖 14.1.1 所示。

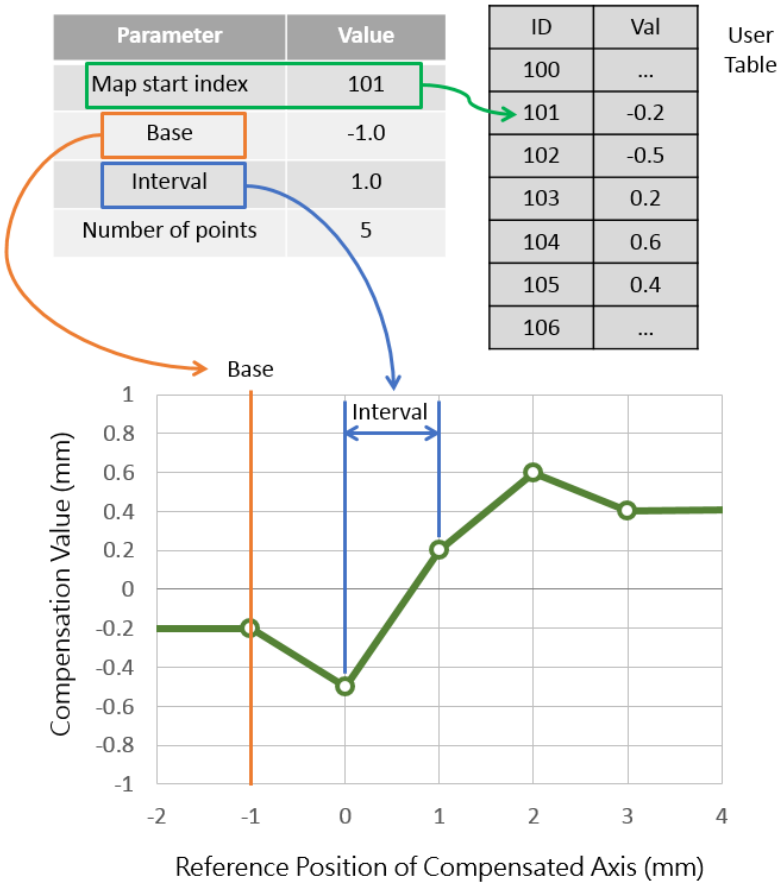


圖 14.1.1

啟動動態誤差補償後，控制器的軸控制命令中，輸出的位置命令會疊加上欲補償的位移量以消除量測已知的誤差量，如圖 5.1.1 所示，其關係為：

參考位置命令 (Reference Position) + 補償值 (Position Compensation) = 位置命令輸出 (Position Output)

啟動動態誤差補償後，可利用 iA Studio 的 Scope Manger 來觀察變數。

- 補償值：Axis → Motion Variable → Position Compensation
- 位置命令 (不含補償) : Axis → Motion Variable → Reference Position
- 位置命令 (含補償) : Axis → Motion Variable → Position Output

限制：

補償命令在 HIMC 中沒有經過路徑的軌跡規劃器，控制器預設最大的誤差補償值為 1 mm。若補償值大於 1 mm，系統會顯示錯誤訊息以提示使用者。

啟動動態誤差補償時，須固定補償的參考座標，故無法變更軸的原點偏移量。

14.1.1 範例

範例 1：一維動態誤差補償

在此範例中，軸 0 為被補償軸，軸 1 為參考軸，被補償軸的補償值會隨著參考軸的位置而變化。其關係如圖 14.1.1.1。

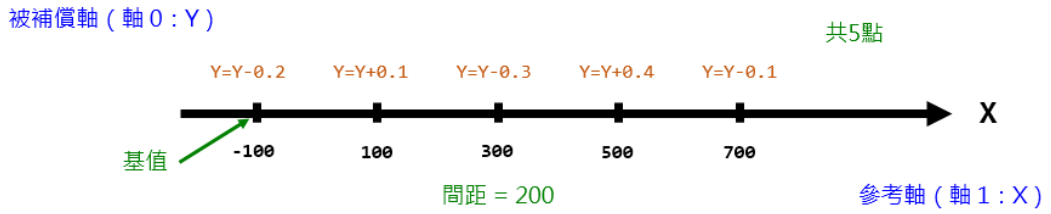


圖 14.1.1.1

利用以下 HMPL 設定並啟動補償。接著，激磁被補償軸（軸 0）、移動參考軸（軸 1）來觀察補償結果。

```
void main() {
    //  設定用戶補償概述表
    double data[5] = {-0.2, 0.1, -0.3, 0.4, -0.1};
    SetUserTable(1, 5, data);

    SetupComp(
      0,    //  被補償軸
      1,    //  User Table 中，補償點的起始編號
      -100, //  起始位置
      200,  //  間距
      5,    //  補償點的數量（含起始位置）
      1     //  參考軸（輸入）
    );
    EnableComp(0);  //  啟動對軸 0 的補償
    Enable(0);  //  激磁軸 0 以啟動補償
}
```

使用者取消補償時，軸的位置命令將被重新設定為目前位置。

```
void main() {
    Disable(0);
    Till(!IsEnabled(0));
    DisableComp(0);  //  取消對軸 0 的補償
```

```
}
```

範例 2：二維動態誤差補償

此範例為被補償軸與參考軸不同軸的應用，大多用於 XYZ 平台上的 Z 軸補償，因 Z 軸在不同的 XY 位置會有幾微米的高度誤差。在此範例中，軸 2 為被補償軸，軸 0 與軸 1 為參考軸，軸 2 的補償值會隨著軸 0 與軸 1 的位置而變化。其關係如圖 14.1.1.2。

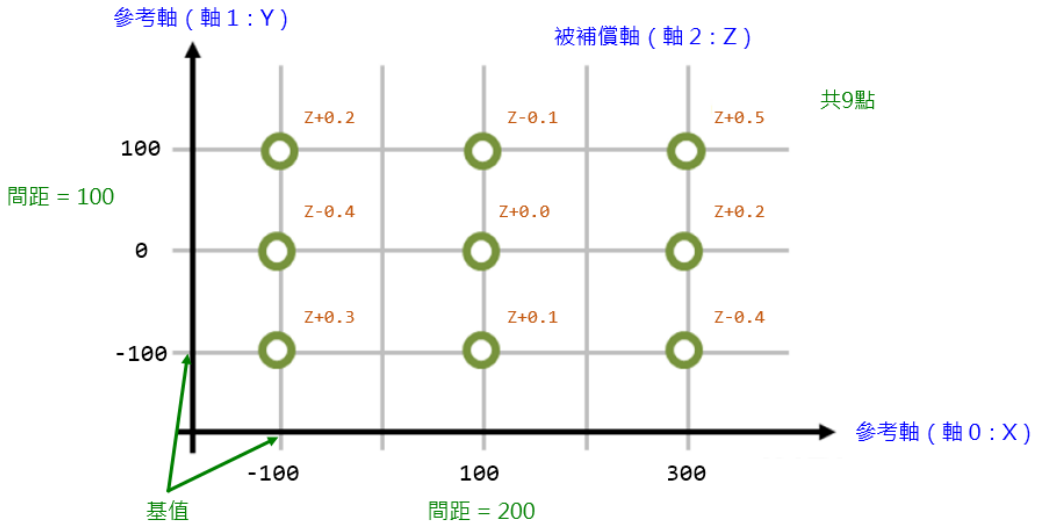


圖 14.1.1.2

利用以下 HMPL 設定並啟動補償。
接著，激磁被補償軸 (軸 2)、移動參考軸 (軸 0 與軸 1) 來觀察補償結果。

```
void main() {
    //  設定用戶補償概述表
    double data[9] = {0.3, 0.1, -0.4, -0.4, 0.0, 0.2, 0.2, -0.1, 0.5};
    SetUserTable(3, 9, data);

    double base[2] = {-100, -100};
    double interval[2] = {200, 100};
    int num_pt[2] = {3, 3};
    int ref_axis[2] = {0, 1};

    SetupComp2D(
      2,     //  被補償軸
      3,     //  User Table 中，補償點的起始編號
      base,  //  起始位置
```

```
        interval,     //  間距
        num_pt,       //  補償點的數量（含起始位置）
        ref_axis      //  參考軸（輸入）
    );
    EnableComp(2);  //  啟動對軸 2 的補償
    Enable(2);  //  激磁軸 2 以啟動補償
}
```

使用者取消補償時，軸的位置命令將被重新設定為目前位置。

```
void main() {
    Disable(2);
    Till(!IsEnabled(2));
    DisableComp(2);  //  取消對軸 2 的補償
}
```

範例 3：三維動態誤差補償

此範例為三維動態誤差補償，應用於精密的 XYZ 平台上。在此範例中，軸 2 為被補償軸，軸 0、軸 1 與軸 2 為參考軸，軸 2 的補償值會隨著軸 0、軸 1 與軸 2 的位置而變化。圖 14.1.1.3 說明參數輸入順序與其補償值，而用戶補償概述表的詳細說明則如圖 14.1.1.4 至圖 14.1.1.6 所示。

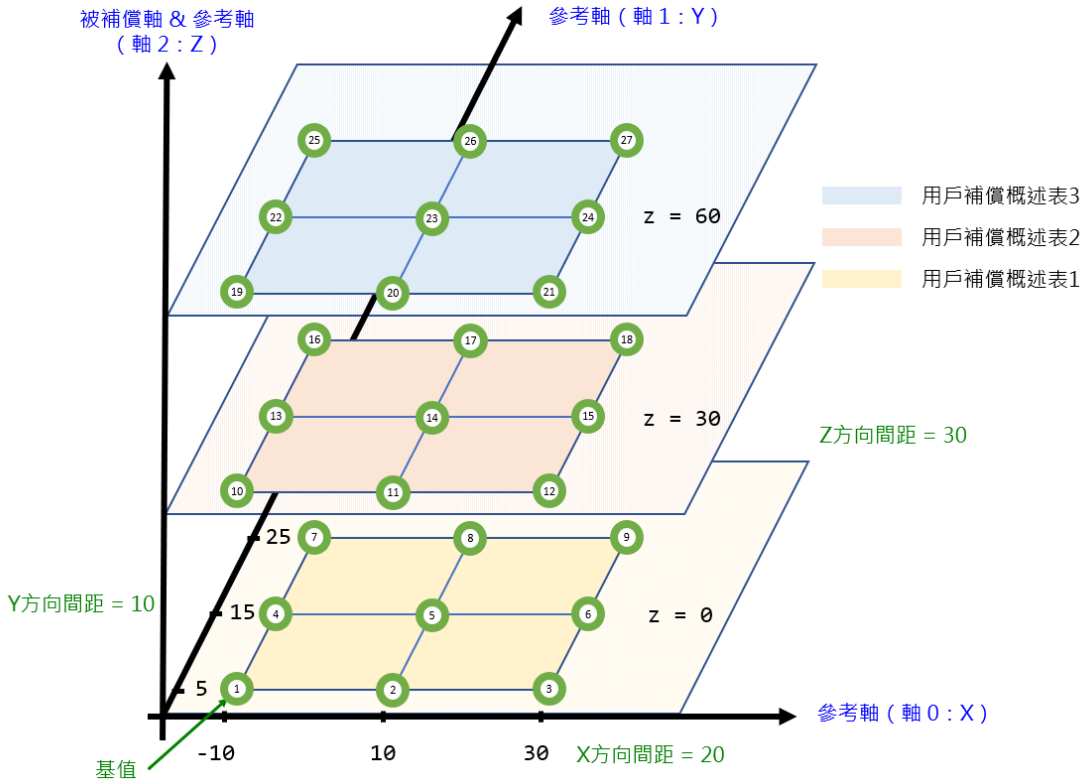


圖 14.1.1.3

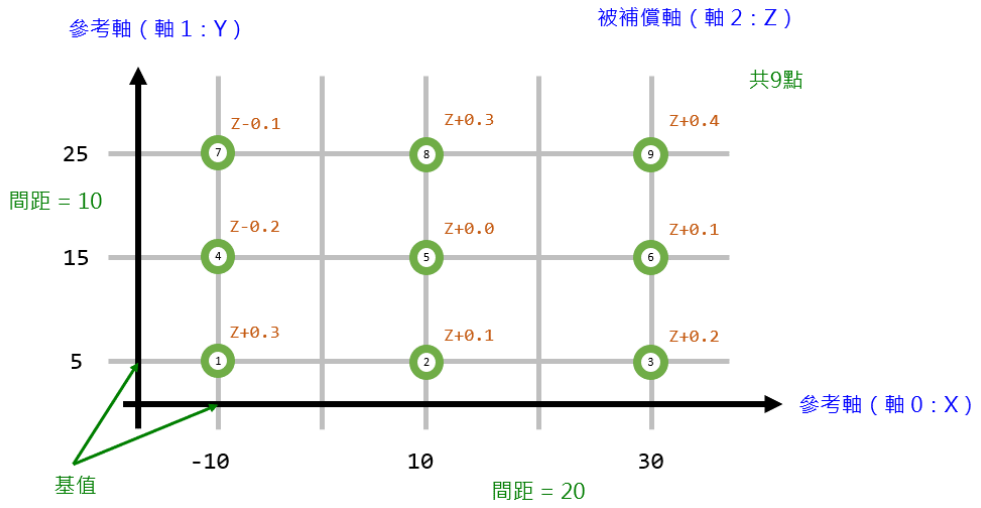


圖 14.1.1.4 用戶補償概述表 1

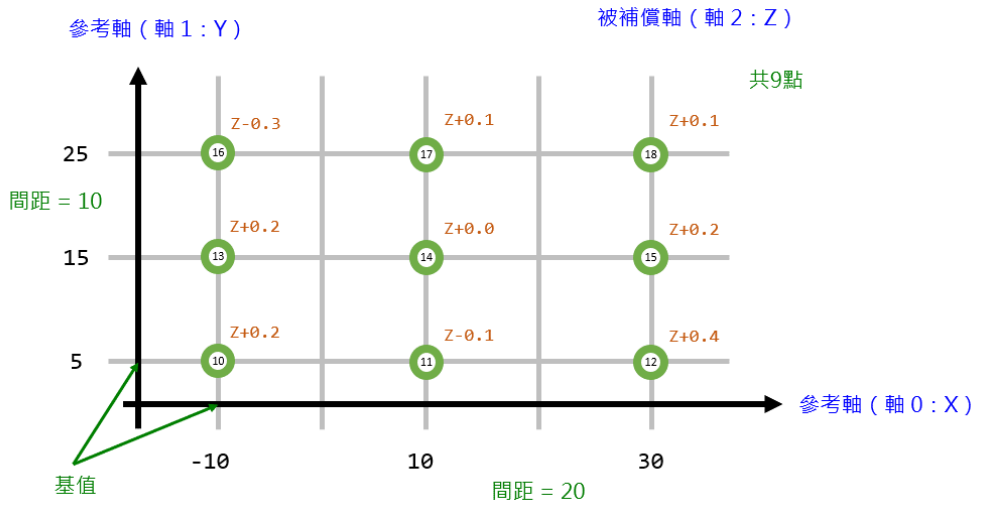


圖 14.1.1.5 用戶補償概述表 2

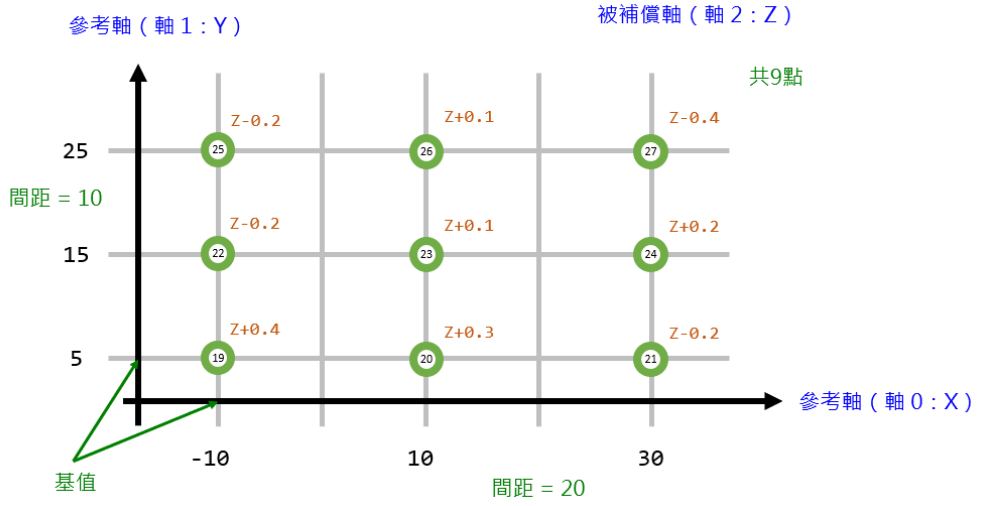


圖 14.1.1.6 用戶補償概述表 3

利用以下 HMPL 設定並啟動補償。

接著，激磁被補償軸（軸 2）、移動參考軸（軸 0、軸 1 與軸 2）來觀察補償結果。

```
void main() {
    //  設定用戶補償概述表
    double data[27] = {0.3, 0.1, 0.2, -0.2, 0.0, 0.1, -0.1, 0.3, 0.4,  //  表 1
                       0.2, -0.1, 0.4, 0.2, 0.0, 0.2, 0.3, 0.1, 0.1,  //  表 2
                       0.4, 0.3, -0.2, -0.2, 0.1, 0.2, -0.2, 0.1, -0.4};  //  表 3
    SetUserTable(3, 27, data);

    double base[3] = {-10, 5, 0};
    double interval[3] = {20, 10, 30};
    int num_pt[3] = {3, 3, 3};
    int ref_axis[3] = {0, 1, 2};

    SetupComp3D(
      2,     //  被補償軸
      3,     //  User Table 中，補償點的起始編號
      base,  //  起始位置
      interval,    //  間距
      num_pt,      //  補償點的數量（含起始位置）
      ref_axis     //  參考軸（輸入）
    );
    EnableComp(2);  //  啟動對軸 2 的補償
    Enable(2);  //  激磁軸 2 以啟動補償
}
```

使用者取消補償時，軸的位置命令將被重新設定為目前位置。

```
void main() {
    Disable(2);
    Till(!IsEnabled(2));
    DisableComp(2);  //  取消對軸 2 的補償
}
```

14.2 EnableComp

用途

啟動軸的動態誤差補償。

語法

```
int EnableComp(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當軸處於激磁狀態，此函式不適用。

需求版本

最低支援版本	iA Studio 0.23

14.3 DisableComp

用途

取消軸的動態誤差補償。

語法

```
int DisableComp(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1) 軸的參考位置將被重新設定為目前位置。

(2) 當軸處於激磁狀態，此函式不適用。

(3) 此函式會清除原有的動態誤差補償設定。欲再啟動動態誤差補償，須重新設定。

需求版本

最低支援版本	iA Studio 0.23

14.4 SetupComp

用途

設定軸的一維動態誤差補償。

語法

```
int SetupComp(
    int    axis_id,
    int    start_idx,
    double base_val,
    double interval,
    int    num_pt,
    int    ref_axis_id
);
```

參數

axis_id [in]　　軸編號。

start_idx [in]　　User Table 中，補償點的起始編號。

base_val [in]　　起始位置（補償輸入的最小值）。
參數單位：mm（毫米）或 deg（角度）

interval [in]　　相鄰補償點的間距。
參數單位：mm（毫米）或 deg（角度）

num_pt [in]　　補償點的數量。

ref_axis_id [in]　　參考軸的編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.23

14.5 SetupComp2D

用途

設定軸的二維動態誤差補償。

語法

```
int SetupComp2D(
    int    axis_id,
    int    start_idx,
    double *base_val,
    double *interval,
    int    *num_pt,
    int    *ref_axis_id
);
```

參數

axis_id [in]　　軸編號。

start_idx [in]　　User Table 中，補償點的起始編號。

base_val [in]　　指向兩元素陣列的指標，內含各維的起始位置（補償輸入的最小值）。
參數單位：mm（毫米）或 deg（角度）

interval [in]　　指向兩元素陣列的指標，內含各維相鄰補償點的間距。
參數單位：mm（毫米）或 deg（角度）

num_pt [in]　　指向兩元素陣列的指標，內含各維補償點的數量。

ref_axis_id [in]　　指向兩元素陣列的指標，內含各維參考軸的編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.23

14.6 SetupComp3D

用途

設定軸的三維動態誤差補償。

語法

```
int SetupComp3D(
    int    axis_id,
    int    start_idx,
    double *base_val,
    double *interval,
    int    *num_pt,
    int    *ref_axis_id
);
```

參數

axis_id [in] 軸編號。

start_idx [in] User Table 中，補償點的起始編號。

base_val [in] 指向三元素陣列的指標，內含各維的起始位置（補償輸入的最小值）。
參數單位：mm（毫米）或 deg（角度）

interval [in] 指向三元素陣列的指標，內含各維相鄰補償點的間距。
參數單位：mm（毫米）或 deg（角度）

num_pt [in] 指向三元素陣列的指標，內含各維補償點的數量。

ref_axis_id [in] 指向三元素陣列的指標，內含各維參考軸的編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.4

14.7 GetCompPos

用途

取得由控制器送至驅動器的軸誤差補償值。

語法

```
double GetCompPos(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

軸的誤差補償值。

單位：mm（毫米）或 deg（角度）

需求版本

最低支援版本	iA Studio 1.3

14.8 SetCompAlgType

用途

設置軸的動態誤差補償插補方式。

語法

```
int SetCompAlgType(
    int axis_id,
    int alg_type
);
```

參數

axis_id [in] 軸編號。

alg_type [in] 動態誤差補償插補方式。

0：一階線性插補（預設值）

1：三階樣條插補

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

三維動態誤差補償不支援三階樣條插補。

需求版本

最低支援版本	iA Studio 2.0

15. 濾波器函式

15.1 概述

利用濾波器函式，修正軌跡規劃器的位置命令。目前，HMPL 提供三種濾波器：平滑時間、VSF 與 InShape。

平滑時間讓馬達平穩地加速，以實現平穩運動；而 VSF 與 InShape 在運動期間抑制馬達的振動（尤其當機構的負載為懸臂時）。透過調整「頻率」和「阻尼比」，即可達到抑制震動的效果。

不能同時使用 VSF 及 InShape，但兩者皆可與平滑時間搭配。

此外，Axis InShape 函式無法對協調運動產生作用，使用者須採用 Group InShape 函式來抑制振動。

註：使用濾波器會增加路徑規劃時間，減少整定時間。

15.1.1 範例

設置 InShape 濾波器的方式如以下 HMPL task 所示。

範例 1：單軸 InShape 濾波器應用

```
void main()
{
    SetAxisInShape(0, 5.5, 0.03, SHAPER_NORMAL);
    //  axis_id, frequency, damping_ratio, shaper_type
    EnableAxisInShape(0);  //  啟動軸0的InShape濾波器
}
```

範例 2：軸群組 InShape 濾波器應用

```
void main()
{
    int gid = 0;          //  設定gid為group id 0
    int axis1 = 0;        //  設定axis1為軸0
    int axis2 = 1;        //  設定axis2為軸1

    Enable(axis1);        //  激磁軸0
    Enable(axis2);        //  激磁軸1
    Till(IsEnabled(axis1) && IsEnabled(axis2));

    AddAxisToGrp(gid, axis1);        //  將axis1加入軸群組gid
```

```
    AddAxisToGrp(gid, axis2);         //  將axis2加入軸群組gid
    EnableGroup(gid);                 //  致能軸群組gid

    SetGrpInShape(gid, 10, 0.15, 1); //  設置軸群組gid的InShape濾波器參數
    EnableGrpInShape(gid);            //  啟動軸群組gid的InShape濾波器
}
```

15.2 EnableAxisVsf

用途

啟動軸的 VSF 濾波器。

語法

```
int EnableAxisVsf(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當馬達正在移動時，此函式不適用。否則，馬達將產生非預期的振動。

需求版本

最低支援版本	iA Studio 1.1

15.3 DisableAxisVsf

用途

取消軸的 VSF 濾波器。

語法

```
int DisableAxisVsf(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.1

15.4 SetAxisVsf

用途

設置軸的 VSF 濾波器參數。

語法

```
int SetAxisVsf(
    int axis_id,
    double frequency,
    double damping_ratio
);
```

參數

axis_id [in] 軸編號。

frequency [in] 系統頻率。
參數單位：Hz（赫茲）
輸入範圍：0.1 ~ 200

damping_ratio [in] 阻尼比。
輸入範圍：0.7 ~ 1.5（建議輸入 **1.0**）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當馬達正在移動時，此函式不適用。否則，馬達將產生非預期的振動。

需求版本

最低支援版本	iA Studio 1.1

15.5 EnableAxisInShape

用途

啟動軸的 InShape 濾波器。

語法

```
int EnableAxisInShape(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當馬達正在移動時，此函式不適用。否則，馬達將產生非預期的振動。

需求版本

最低支援版本	iA Studio 1.1

15.6 DisableAxisInShape

用途

取消軸的 InShape 濾波器。

語法

```
int DisableAxisInShape(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.1

15.7 SetAxisInShape

用途

設置軸的 InShape 濾波器參數。

語法

```
int SetAxisInShape(
    int axis_id,
    double frequency,
    double damping_ratio,
    int shaper_type
);
```

參數

axis_id [in] 軸編號。

frequency [in] 系統頻率。
參數單位：Hz（赫茲）
輸入範圍：1.5 ~ 300

damping_ratio [in] 阻尼比。
輸入範圍：0.0 ~ 0.3

shaper_type [in] Shaper 類型。設為 1：**SHAPER_NORMAL**；設為 0：**SHAPER_ROBUST**。
SHAPER_ROBUST 的效果比 SHAPER_NORMAL 更強，但 SHAPER_NORMAL 的強度足以抑制振動。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1) 當馬達正在移動時，此函式不適用。否則，馬達將產生非預期的振動。

(2) 系統頻率與阻尼比的預設值分別為 5.5Hz 與 0.03。

需求版本

最低支援版本	iA Studio 1.1

15.8 EnableGrpInShape

用途

啟動軸群組的 InShape 濾波器。

語法

```
int EnableGrpInShape(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

當馬達正在移動時，此函式不適用。否則，馬達將產生非預期的振動。

需求版本

最低支援版本	iA Studio 1.1

15.9 DisableGrpInShape

用途

取消軸群組的 InShape 濾波器。

語法

```
int DisableGrpInShape(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.1

15.10 SetGrpInShape

用途

設置軸群組的 InShape 濾波器參數。

語法

```
int SetGrpInShape(
    int group_id,
    double frequency,
    double damping_ratio,
    int shaper_type
);
```

參數

group_id [in] 軸群組編號。

frequency [in] 系統頻率。
參數單位：Hz（赫茲）
輸入範圍：3.0 ~ 300

damping_ratio [in] 阻尼比。
輸入範圍：0.0 ~ 0.3

shaper_type [in] Shaper 類型。設為 1：**SHAPER_NORMAL**；設為 0：**SHAPER_ROBUST**。
SHAPER_ROBUST 的效果比 SHAPER_NORMAL 更強，但 SHAPER_NORMAL 的強度足以抑制振動。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

(1) 當馬達正在移動時，此函式不適用。否則，馬達將產生非預期的振動。

(2) 系統頻率與阻尼比的預設值分別為 5.5Hz 與 0.03。

需求版本

最低支援版本	iA Studio 1.1

16. HMPL Task 函式

16.1 概述

HIMC 內建 64 個 HMPL task 讓使用者能實作應用所需的運動規劃命令。在任一 HMPL task 中，使用者可透過 HMPL task 函式來啟動或停止其他的 HMPL task。當 HMPL task 正在執行中，則無法重複要求已在執行中的 HMPL task 重新執行，須等到 task 完成執行、進入停止狀態為止。但使用者可以詢問 HMPL task 目前是否處於正在執行的狀態，並依此對多個 HMPL task 進行應用所需的順序控制。

16.2 StartTask

用途

開始執行 HMPL task。

語法

```
int StartTask(
    int task_id
);
```

參數

task_id [in] HMPL task ID。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.22

16.3 StartTaskFunc

用途

開始執行 HMPL task 中的一個函式。

語法

```
int StartTaskFunc(
    int   task_id,
    char *func_name
);
```

參數

task_id [in]　　HMPL task ID。

func_name [in]　　指標型態的記憶體，用來儲存 HMPL task 中的函式名稱。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.22

16.4 StopTask

用途

停止執行 HMPL task。

語法

```
int StopTask(
    int task_id
);
```

參數

task_id [in] HMPL task ID。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 0.22

16.5 StopAllTask

用途

停止執行所有 HMPL task（包含呼叫者）。

語法

```
void StopAllTask();
```

參數

無

回傳值

無

需求版本

最低支援版本	iA Studio 0.23

16.6 IsTaskStop

用途

詢問 HMPL task 是否已停止執行。

語法

```
int IsTaskStop(
    int task_id
);
```

參數

task_id [in] HMPL task ID。

回傳值

若已停止執行 HMPL task，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 0.23

(此頁有意留白。)

17. 變數與函式操作函式

17.1 概述

HIMC 提供使用者驅動器與控制器的變數操作函式。對於驅動器的變數操作，可透過 CoE 通訊函式，依據 CoE 裝置的物件字典，對相關變數進行資料交換；而控制器的變數操作，使用者須根據控制器的變數 ID 列表，給定特定變數的定址 ID 來做存取。17.1.1 節為控制器的變數定義說明。

注意：
若無特定目的需求，建議使用者利用相關人機介面與函式來存取相關的系統變數。使用變數操作函式時，使用者須自行確保存取變數與輸入數值的安全性。

17.1.1 控制器變數列表

HIMC 使用 32 個位元做為控制器變數的定址 ID，其型式為 0x□□□□□□□□，其中 0x 表示數值為十六進位制。透過變數操作函式，使用者可存取 HIMC 提供的系統變數、軸變數與軸群組變數。定址 ID 的規則說明如下：

1. 定址 ID 的第 1~2 個數值表示『控制器變數的類別』，系統變數為 0x00□□□□□□、軸變數為 0x83□□□□□□、軸群組變數為 0x82□□□□□□。

2. 定址 ID 的第 3~4 個數值表示『軸 ID 或軸群組 ID』。例如：軸變數 0x8302□□□□為存取軸編號 02 的變數；軸群組變數 0x8201□□□□為存取軸群組編號 01 的變數。

3. 定址 ID 的第 5~8 個數值表示『控制器系統、軸或軸群組變數的定址位置』，變數列表與說明請參考表 17.1.1.1 至表 17.1.1.3。

表 17.1.1.1

系統變數		
定址 ID	變數名稱	說明
0x0000012c	HCV_ID_fclk	系統執行 Clock（每 250 us 增加 1 count）
0x0000012e	HCV_ID_timeInMs	系統執行時間（ms）
0x000007d0	HCV_ID_user_table	使用者可自由使用的 double[512000]陣列變數
0x00002328	HCV_ID_ltest0	使用者可自由使用的 int 變數
0x00002329	HCV_ID_ltest1	使用者可自由使用的 int 變數
0x0000232a	HCV_ID_ltest2	使用者可自由使用的 int 變數
0x0000232b	HCV_ID_ltest3	使用者可自由使用的 int 變數
0x0000232c	HCV_ID_ltest4	使用者可自由使用的 int 變數
0x0000232d	HCV_ID_ltest5	使用者可自由使用的 int 變數
0x0000232e	HCV_ID_ltest6	使用者可自由使用的 int 變數
0x0000232f	HCV_ID_ltest7	使用者可自由使用的 int 變數
0x00002330	HCV_ID_ltest8	使用者可自由使用的 int 變數
0x00002331	HCV_ID_ltest9	使用者可自由使用的 int 變數
0x0000235a	HCV_ID_dtest0	使用者可自由使用的 double 變數
0x0000235b	HCV_ID_dtest1	使用者可自由使用的 double 變數
0x0000235c	HCV_ID_dtest2	使用者可自由使用的 double 變數
0x0000235d	HCV_ID_dtest3	使用者可自由使用的 double 變數
0x0000235e	HCV_ID_dtest4	使用者可自由使用的 double 變數
0x0000235f	HCV_ID_dtest5	使用者可自由使用的 double 變數
0x00002360	HCV_ID_dtest6	使用者可自由使用的 double 變數
0x00002361	HCV_ID_dtest7	使用者可自由使用的 double 變數

系統變數		
定址 ID	變數名稱	說明
0x00002362	HCV_ID_dtest8	使用者可自由使用的 double 變數
0x00002363	HCV_ID_dtest9	使用者可自由使用的 double 變數
0x0000238c	HCV_ID_mtest	使用者可自由使用的 double[10]陣列變數

表 17.1.1.2

軸變數		
定址 ID	變數名稱	說明
0x83□□0015	HCV_ID_motion_type	運動型態
0x83□□0033	HCV_ID_pos_tr	到位收斂半徑
0x83□□0034	HCV_ID_pos_tr_t	到位整定時間
0x83□□0065	HCV_ID_sw_RL	軟體右極限
0x83□□0066	HCV_ID_sw_LL	軟體左極限
0x83□□0067	HCV_ID_vel_lim	最大速度限制
0x83□□0068	HCV_ID_acc_lim	最大加速度限制
0x83□□0069	HCV_ID_dec_lim	最大減速度限制
0x83□□0079	HCV_ID_max_pos_err	位置誤差限制
0x83□□007a	HCV_ID_max_comp_lim	位置補償限制
0x83□□00a0	HCV_ID_home_status	歸原點狀態
0x83□□00a1	HCV_ID_home_method	歸原點方法
0x83□□00a2	HCV_ID_home_fast_vel	快速歸原點速度
0x83□□00a3	HCV_ID_home_slow_vel	慢速歸原點速度
0x83□□00a4	HCV_ID_home_timeout	歸原點逾時時間
0x83□□00a5	HCV_ID_home_acc	歸原點加速度
0x83□□00a6	HCV_ID_home_offset	歸原點位置偏移量
0x83□□00d3	HCV_ID_max_vel	目標速度
0x83□□00d4	HCV_ID_max_acc	目標加速度
0x83□□00d5	HCV_ID_max_dec	目標減速度
0x83□□00d7	HCV_ID_sm_factor	平滑時間
0x83□□00db	HCV_ID_vel_scale	速度百分比（0~100）
0x83□□00dd	HCV_ID_p2p_del	P2P 運動等待時間
0x83□□00de	HCV_ID_p2p_pos1	P2P 位置 1
0x83□□00df	HCV_ID_p2p_pos2	P2P 位置 2
0x83□□00e0	HCV_ID_p2p_repeat	重覆 P2P 運動
0x83□□00e1	HCV_ID_rlt_dist	相對移動距離
0x83□□00e2	HCV_ID_en_motionManager	Motion Manager 運動軸選擇
0x83□□00e3	HCV_ID_acc_time	加速度時間

軸變數		
定址 ID	變數名稱	說明
0x83□□00e4	HCV_ID_dec_time	減速度時間
0x83□□00e9	HCV_ID_map_io_type	誤差補償類型
0x83□□0117	HCV_ID_rollover_turns	單圈模式 rollover 次數
0x83□□0119	HCV_ID_rollover_val	單圈模式 rollover 設定值
0x83□□0193	HCV_ID_gant_pair	龍門配置的龍門 ID
0x83□□01f7	HCV_ID_en_delay	軸激磁 Time Out 時間
0x83□□01ff	HCV_ID_fb_ratio_pos	驅動器位置解析度，長度單位（分母）
0x83□□0200	HCV_ID_fb_ratio_cnt	驅動器位置解析度，單位 count（分子）
0x83□□0209	HCV_ID_fb_curr_ratio_curr	驅動器電流解析度，電流單位（分母）
0x83□□020a	HCV_ID_fb_curr_ratio_cnt	驅動器電流解析度，單位 count（分子）
0x83□□0213	HCV_ID_rotor_inertia	馬達轉子慣量比
0x83□□0214	HCV_ID_force_constant	馬達力矩常數
0x83□□0263	HCV_ID_last_err	軸錯誤代碼
0x83□□03c2	HCV_ID_gear_ratio	電子齒輪比

註：符號□□會是該軸的 ID，ID 會以十六進位制表示。例如：01 為軸編號 01；0f 為軸編號 15。

表 17.1.1.3

軸群組變數		
定址 ID	變數名稱	說明
0x82□□0002	HCV_ID_grp_num_axis	軸群組軸數
0x82□□00ca	HCV_ID_grp_lin_vel_lim	軸群組線性運動速度限制
0x82□□00cb	HCV_ID_grp_lin_acc_lim	軸群組線性運動加速度限制
0x82□□00cc	HCV_ID_grp_lin_dec_lim	軸群組線性運動減速度限制
0x82□□00d4	HCV_ID_grp_ang_vel_lim	軸群組旋轉運動速度限制
0x82□□00d5	HCV_ID_grp_ang_acc_lim	軸群組旋轉運動加速度限制
0x82□□00d6	HCV_ID_grp_ang_dec_lim	軸群組旋轉運動減速度限制
0x82□□00dd	HCV_ID_grp_lin_vel	軸群組線性運動目標速度
0x82□□00de	HCV_ID_grp_lin_acc	軸群組線性運動目標加速度
0x82□□00df	HCV_ID_grp_lin_dec	軸群組線性運動目標減速度
0x82□□00e0	HCV_ID_grp_lin_sf	軸群組線性運動平滑時間
0x82□□00e1	HCV_ID_grp_lin_acc_time	軸群組線性運動目標加速度時間
0x82□□00e2	HCV_ID_grp_lin_dec_time	軸群組線性運動目標減速度時間
0x82□□00e7	HCV_ID_grp_ang_vel	軸群組旋轉運動目標速度
0x82□□00e8	HCV_ID_grp_ang_acc	軸群組旋轉運動目標加速度
0x82□□00e9	HCV_ID_grp_ang_dec	軸群組旋轉運動目標減速度
0x82□□00ea	HCV_ID_grp_ang_sf	軸群組旋轉運動平滑時間

軸群組變數		
定址 ID	變數名稱	說明
0x82□□00eb	HCV_ID_grp_ang_acc_time	軸群組旋轉運動目標加速度時間
0x82□□00ec	HCV_ID_grp_ang_dec_time	軸群組旋轉運動目標減速度時間
0x82□□00f1	HCV_ID_grp_coord_sys	軸群組座標系統
0x82□□00f2	HCV_ID_grp_buffer_mode	軸群組速度緩衝模式
0x82□□00f3	HCV_ID_grp_trans_mode	軸群組路徑過渡模式
0x82□□00f4	HCV_ID_grp_trans_vel	軸群組路徑過渡速度
0x82□□00f5	HCV_ID_grp_trans_dis	軸群組路徑過渡距離
0x82□□00f6	HCV_ID_grp_trans_dev	軸群組路徑過渡偏移量
0x82□□00f7	HCV_ID_grp_trans_curvature	軸群組路徑過渡曲率
0x82□□0104	HCV_ID_grp_vel_scale	軸群組速度百分比（0~100）
0x82□□0119	HCV_ID_grp_shaper_fr	軸群組 InShape 濾波器頻率
0x82□□011a	HCV_ID_grp_shaper_xi	軸群組 InShape 濾波器阻尼比
0x82□□038f	HCV_ID_grp_last_err	軸群組錯誤代碼

註：符號□□會是該軸群組的 ID，ID 會以十六進位制表示。例如：01 為軸群組編號 01；0f 為軸群組編號 15。

17.2 驅動器變數操作

17.2.1 ReadSDO

用途

透過 SDO 讀取從站的物件數值。

語法

```
double ReadSDO(
    int slv_id,
    int obj_index,
    int obj_subindex,
    int obj_length
);
```

參數

slv_id [in] 從站編號。
obj_index [in] 從站物件的索引。
obj_subindex [in] 從站物件的子索引。
obj_length [in] 從站物件的 Byte 長度。

回傳值

若執行成功，將回傳 **double** 型態的 **SDO** 物件數值，若失敗，則回傳 **double** 型態的數值**-1**。

範例

```
void main()
{
    double value= ReadSDO(0, 0x6041, 0 , 2);
    Print("value = %f", value);
}
```

需求版本

最低支援版本	iA Studio 3.0

17.2.2 ReadSDOEx

用途

透過 SDO 讀取從站的物件數值，並將讀取的數值放置於指標記憶體中。

語法

```
int ReadSDOEx(
    int slv_id,
    int obj_index,
    int obj_subindex,
    int obj_length,
    double* obj_value
);
```

參數

slv_id [in] 從站編號。

obj_index [in] 從站物件的索引。

obj_subindex [in] 從站物件的子索引。

obj_length [in] 從站物件的 Byte 長度。

obj_value [out] 指標型態的記憶體，用來儲存回傳的 SDO 物件數值。

回傳值

若執行成功，將回傳 **int** 型態的數值 **0**，若失敗，則回傳數值**-1**。

範例

```
void main()
{
    double value;
    int rtn;
    rtn = ReadSDOEx(0, 0x6041, 0 , 2, &value);
    Print("return = %d, value = %f", rtn, value);
}
```

需求版本

最低支援版本	iA Studio 3.0

17.2.3 WriteSDO

用途

透過 SDO 寫入從站的物件數值。

語法

```
int WriteSDO(
    int slv_id,
    int obj_index,
    int obj_subindex,
    int obj_length,
    double obj_value
);
```

參數

slv_id [in]	從站編號。
obj_index [in]	從站物件的索引。
obj_subindex [in]	從站物件的子索引。
obj_length [in]	從站物件的 Byte 長度。
obj_value [in]	欲寫入的 SDO 物件數值。

回傳值

若執行成功，回傳值為 **int** 型態的數值 **0**，若失敗，則回傳 **int** 型態的數值**-1**。

需求版本

最低支援版本	iA Studio 3.0

17.2.4 ReadPDO

用途

透過 PDO 讀取從站已配置成 PDO 的物件數值。

語法

```
double ReadPDO(
    int slv_id,
    int obj_index,
    int obj_subindex,
);
```

參數

slv_id [in] 從站編號。

obj_index [in] 從站物件的索引。

obj_subindex [in] 從站物件的子索引。

obj_value [out] 指標型態的記憶體，用來儲存回傳的 PDO 物件數值。

回傳值

若執行成功，將回傳 **int** 型態的物件數值 0，若失敗，則回傳數值**-1**。

範例

```
void main()
{
    double value;
    int rtn;
    rtn = ReadPDO(0, 0x6041, 0, &value);
    Print("value = %f", value);
}
```

需求版本

最低支援版本	iA Studio 3.0

17.2.5 ReadPDOEx

用途

透過 PDO 讀取從站已配置成 PDO 的物件數值。

語法

```
int ReadPDOEx(
    int slv_id,
    int obj_index,
    int obj_subindex,
    double* obj_value
);
```

參數

slv_id [in] 從站編號。

obj_index [in] 從站物件的索引。

obj_subindex [in] 從站物件的子索引。

obj_value [out] 指標型態的記憶體，用來儲存回傳的 PDO 物件數值。

回傳值

若執行成功，將回傳 **int** 型態的物件數值 0，若失敗，則回傳數值**-1**。

範例

```
void main()
{
    double value;
    value = ReadPDOEx(0, 0x6041, 0, &value);
    Print("value = %f", value);
}
```

需求版本

最低支援版本	iA Studio 3.0

17.2.6 WritePDO

用途

透過 PDO 將數值寫入從站已配置成 PDO 的物件。

語法

```
int WritePDO(
    int slv_id,
    int obj_index,
    int obj_subindex,
    double obj_value
);
```

參數

slv_id [in] 從站編號。
obj_index [in] 從站物件的索引。
obj_subindex [in] 從站物件的子索引。
obj_value [in] 欲寫入的 PDO 物件數值。

回傳值

若執行成功，回傳值為 **int** 型態的數值 **0**，若失敗，則回傳 **int** 型態的數值**-1**。

備註

若有其他來源同時寫入該物件，會有被其他來源覆寫的風險。

需求版本

最低支援版本	iA Studio 3.0

17.2.7 ForceWritePDO

用途

透過 PDO 將數值強制寫入從站已配置成 PDO 的物件。

語法

```
int ForceWritePDO(
    int slv_id,
    int obj_index,
    int obj_subindex,
    double obj_value
);
```

參數

slv_id [in] 從站編號。
obj_index [in] 從站物件的索引。
obj_subindex [in] 從站物件的子索引。
obj_value [in] 欲寫入的 PDO 物件數值。

回傳值

若執行成功，回傳值為 **int** 型態的數值 **0**，若失敗，則回傳 **int** 型態的數值**-1**。

需求版本

最低支援版本	iA Studio 3.0

17.2.8 ReleasePDO

用途

釋放被強制寫入的 PDO 物件，與 ForceWritePDO 搭配使用。

語法

```
int ReleasePDO(
    int slv_id,
    int obj_index,
    int obj_subindex
);
```

參數

slv_id [in] 從站編號。

obj_index [in] 從站物件的索引。

obj_subindex [in] 從站物件的子索引。

回傳值

若執行成功，回傳值為 **int** 型態的數值 **0**，若失敗，則回傳 **int** 型態的數值**-1**。

需求版本

最低支援版本	iA Studio 3.0

17.3 控制器變數操作

17.3.1 GetConfigVar

用途

取得控制器的變數值。

語法

```
double GetConfigVar(
    int hcv_id,
    int *result
);
```

參數

hcv_id [in]	HIMC 控制器變數 ID，HCV ID 定義請參閱 17.1.1 節。
result [out]	若函式執行成功，將回傳 **int** 型態的值 **0**； 若函式執行失敗，將回傳**-1**。

回傳值

變數值。

範例

```
void main()
{
    int result = 0;
    //  0x83020065為軸2的軟體右極限
    double SW_RL = GetConfigVar(0x83020065, &result);
    Print("SW_RL = %f", SW_RL);
}
```

需求版本

最低支援版本	iA Studio 1.1

17.3.2 SetConfigVar

用途

設置控制器的變數值。

語法

```
int SetConfigVar(
    int hcv_id,
    double value
);
```

參數

hcv_id [in]	HIMC 控制器變數 ID，HCV ID 定義請參閱 17.1.11 節。
value [in]	新的變數值。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

範例

```
void main()
{
    int result = 0;
    SetConfigVar(0x83000065, 0.0);  //  0x83000065為軸0的軟體右極限
    Print("SW_RL = %f", GetConfigVar(0x83000065, &result));
}
```

需求版本

最低支援版本	iA Studio 1.1

18. 錯誤函式

18.1 概述

HIMC 使用 32 位元的錯誤代碼來表示相關的錯誤訊息。透過本章提供的函式，使用者可取得或清除系統、軸與軸群組的錯誤代碼 (各類別之錯誤代碼、名稱與說明分別條列於 18.1.1 節至 18.1.3 節)。錯誤代碼的型式為 0x□□□□□□□□，其中 0x 表示數值為十六進位制。其規則與控制器變數定址 ID 相同，說明如下：

1. 錯誤代碼的第 1~2 個數值表示『控制器變數的類別』，系統變數為 0x00□□□□□□、軸變數為 0x83□□□□□□、軸群組變數為 0x82□□□□□□。
2. 錯誤代碼的第 3~4 個數值表示『軸 ID 或軸群組 ID』。例如：軸變數 0x8302□□□□為存取軸編號 02 的變數；軸群組變數 0x8201□□□□為存取軸群組編號 01 的變數。
3. 錯誤代碼的第 5~8 個數值表示『變數 ID』，詳細說明請參閱 18.1.1 節至 18.1.3 節。

註：由於函式回傳值為十進位制，使用者須自行轉換成十六進位制，才能對應到正確的錯誤代碼。

18.1.1 系統錯誤訊息

表 18.1.1.1

系統錯誤代碼		
錯誤代碼	錯誤名稱	說明
0x00000001	eERR_HCV_ID_NOT_FOUND	無法找到此變數 ID。
0x00000002	eERR_DATA_EXCEEDED	要求的資料超出範圍。
0x00000003	eERR_HCV_IS_READ_ONLY	此參數唯讀。
0x00000004	eERR_HCV_VALUE_OUT_OF_RANGE	輸入值超出範圍限制。
0x00000064	eERR_EMERGENCY_STOP	控制器已緊急停止。
0x00000100	eERR_MAIL_BOX_BUSY	控制器及從站間的 mailbox 忙碌。
0x00000101	eERR_VAR_NOT_IN_SLV_DB	無法找到此從站變數。
0x00000102	eERR_VAR_NOT_REGYET	無法讀取此從站變數。
0x00000103	eERR_READ_VAR_NO_RECV	從站無回應。
0x00000104	eERR_PREV_SLV_CMD_NOT_FIN	發送至從站的前一命令尚未執行完畢。
0x00000105	eERR_SLV_ID_INVALID	此從站 ID 無效。
0x00000106	eERR_PDO_NUM_EXCEED	PDO 數量超出範圍。
0x00000107	eERR_NOT_VALID_TASKID	此 task ID 無效。
0x00000108	eERR_TASK_IS_RUNNING	此 task 正在執行中。
0x00000109	eERR_FUNC_NOT_IN_TASK	此 task 未包含此函式。
0x0000010a	eERR_TASK_EMPTY	此 task 沒有內容。
0x0000010b	eERR_TASK_NOT_RUNNING	此 task 未在執行。
0x0000012c	eERR_NIC_INIT_TOUT	mega-ulink 通訊用的網路埠未備妥。
0x0000012d	eERR_HARDWARE_MISMATCH	無法辨識的硬體。
0x0000012e	eERR_SLAVE_NUM_MISMATCH	從站數量與組態不符。
0x0000012f	eERR_INVALID_PDO	此 PDO 無效。
0x00000130	eERR_INVALID_MCK_CNFG	運動核心 (motion kernel) 組態無效。
0x00000138	eERR_HIMC_LOAD_CONFIG_FAIL	無法從 SSD 下載組態，請再儲存一次。
0x00000139	eERR_HIMC_SAVE_CONFIG_FAIL	無法將組態存至 HIMC，請再儲存一次。
0x0000013a	eERR_HIMC_SAVE_CONFIG_COPY_FAIL	無法將組態存至 HIMC。無法將檔案存至 SAVE 資料夾。
0x0000013b	eERR_HIMC_SAVE_UPDATE_PRM_TIMEOUT	無法將組態存至 HIMC。Prm 數值更新逾時。
0x0000013c	eERR_ETHERCAT_LICENSE_MISMATCH	EtherCAT 授權無效。
0x000001f4	eERR_ISR_NOT_STABLE	中斷週期不穩定。
0x000001f5	eERR_MCK_OVERLOAD	運動核心 (motion kernel) 過載。
0x000001f6	eERR_ISR_OVERLOAD	CPU 過載。
0x00001388	eERR_HMPL_INVALID_ARG	HMPL 內的自變數無效。

系統錯誤代碼		
錯誤代碼	錯誤名稱	說明
0x00001389	eERR_HMPL_INVALID_PTR	HMPL 內的指標 (pointer) 無效。
0x0000138a	eERR_HMPL_STACK_OVERFLOW	HMPL 內堆疊溢位。
0x0000138b	eERR_HMPL_ILLEGAL_MEM_OP	此記憶體操作在 HMPL 內是非法的。
0x0000138c	eERR_HMPL_MOTION_NOT_READY	控制器狀態未就緒，無法執行此 HMPL。
0x0000138d	eERR_HMPL_STR_TOO_LONG	字串長度超過限制。
0x0000138e	eERR_HMPL_INVALID_STR_FORMAT	字串格式錯誤。
0x0000138f	eERR_HMPL_ARG_OUT_OF_RANGE	輸入的引數超過範圍。
0x00001392	eERR_HMPL_ASCII_AGENT_RUNNING	ASCII agent 執行中，無法同時執行多個 ASCII agent。
0x0000139c	eERR_HMPL_CANNOT_RUN_IN_DEBUG	此 HMPL 函式無法在偵錯模式中執行。
0x000013a6	eERR_HMPL_TOO_MANY_BRK_POINT	此 task 有太多中斷點。
0x000013ec	eERR_HMPL_MUTEX_LOCK_TWICE	重複鎖定已鎖定的互斥鎖。
0x00001450	eERR_HMPL_INVALID_SYS_TIME_MEMORY	記憶體太小，至少要 30 Bytes。
0x00001451	eERR_HMPL_NOT_SUPPORTED	此平台不支援此 HMPL 函式。
0x00001452	eERR_HMPL_CLIENT_NOT_CONNECTED	無法連線至客戶端。
0x0000176f	eERR_HMPL_INTERNAL_ERROR	HMPL 內部錯誤。
0x00001770	eERR_HMPL_EXEC_FAILED	HMPL 執行失敗。
0x00001771	eERR_HMPL_ASM_LOAD_FAILED	HMPL 編譯失敗。組合語言檔案不存在或未產生。
0x00001772	eERR_HMPL_STARTTASK_TIMEOUT	HMPL StartTask 函式逾時。
0x00001773	eERR_HMPL_STOPTASK_TIMEOUT	HMPL StopTask 函式逾時。
0x000017d4	eERR_ASCII_CONNECT_TIMEOUT	ASCII 用戶端連線逾時。
0x000017d5	eERR_ASCII_CONNECT_FAILED	ASCII 用戶端連線失敗，請檢查 ip 及連接埠。
0x000017d6	eERR_ASCII_MULTI_CONNECTING	多個 ASCII 用戶端同時連線。
0x000017d7	eERR_ASCII_MULTI_DISCONNECTING	多個 ASCII 用戶端同時斷線。
0x000017d8	eERR_ASCII_DISCONNECT_TIMEOUT	ASCII 用戶端斷線逾時。
0x000017de	eERR_ASCII_RECV_TIMEOUT	ASCII 用戶端接收資料逾時，請稍候再試。
0x000017df	eERR_ASCII_RECV_FAIL	ASCII用戶端接收資料失敗,請確認連線是否正常。
0x000017e0	eERR_ASCII_MULTI_RECVING	多個 ASCII 用戶端同時接受資料。
0x000017e8	eERR_ASCII_SEND_TIMEOUT	ASCII 用戶端傳送資料逾時，請稍候再試。
0x000017e9	eERR_ASCII_SEND_FAIL	ASCII用戶端傳送資料失敗,請確認連線是否正常。
0x000017ea	eERR_ASCII_MULTI_SENDING	多個 ASCII 用戶端同時傳送資料。
0x00001838	eERR_MODBUS_CONNECT_TIMEOUT	Modbus 用戶端連線逾時。
0x00001839	eERR_MODBUS_CONNECT_FAILED	Modbus 用戶端連線失敗，請檢查 ip。
0x0000183a	eERR_MODBUS_MULTI_CONNECTING	多個 Modbus 用戶端同時連線。
0x0000183b	eERR_MODBUS_MULTI_DISCONNECTING	多個 Modbus 用戶端同時斷線。

系統錯誤代碼		
錯誤代碼	錯誤名稱	說明
0x0000183c	eERR_MODBUS_DISCONNECT_TIMEOUT	Modbus 用戶端斷線逾時。
0x0000183d	eERR_MODBUS_DATALENGTH_ERR	Modbus 用戶端的讀寫資料數量超過上限。
0x0000183e	eERR_MODBUS_SOCKET_BUSY	Modbus 用戶端同時處理兩個以上的命令。
0x0000183f	eERR_MODBUS_JOB_TIMEOUT	Modbus 用戶端執行任務逾時，請稍候再試。
0x00001840	eERR_MODBUS_JOB_FAIL	Modbus 用戶端執行任務失敗,請確認連線是否正常。

18.1.2 軸錯誤訊息

以下錯誤代碼會出現在軸發生錯誤或操作無效時。錯誤發生時，符號□□會是該軸的 ID，ID 會以十六進位制表示。例如：01 為軸編號 01；0f 為軸編號 15。

表 18.1.2.1

軸錯誤代碼		
錯誤代碼	錯誤名稱	說明
0x83□□000a	eERR_AXIS_CMD_UNKOWN	未知的命令名稱。
0x83□□001e	eERR_AXIS_CMD_QUEUE_FULL	軸命令佇列 (queue) 已滿。
0x83□□0064	eERR_AXIS_CMD_INVALID_STATE	軸在當前的狀態下無法執行此命令。
0x83□□006e	eERR_AXIS_CMD_INVALID_ENABLED	軸激磁時無法使用此命令。
0x83□□0078	eERR_AXIS_CMD_INVALID_DISABLED	軸解激磁時無法使用此命令。
0x83□□0082	eERR_AXIS_CMD_INVALID_MOVING	軸移動時無法執行此命令。
0x83□□008c	eERR_AXIS_CMD_INVALID_STOPPING	軸停止時無法執行此命令。
0x83□□0096	eERR_AXIS_CMD_INVALID_ERROR_STATE	軸發生錯誤時無法執行此命令。
0x83□□00a0	eERR_AXIS_CMD_INVALID_IN_SYNC	軸處於同步運動狀態時，此命令無效。
0x83□□00aa	eERR_AXIS_CMD_INVALID_GEAR_MASTER	軸為電子齒輪主軸時，此命令無效。
0x83□□00b4	eERR_AXIS_CMD_INVALID_PP_MODE	軸在 PP 模式下，此命令無效。
0x83□□00be	eERR_AXIS_CMD_INVALID_MAP_SWITCHING	軸切換補償表時，此命令無效。
0x83□□00c8	eERR_AXIS_CMD_INVALID_INPUTSHAPING_ENABLED	軸位置命令塑型功能開啟時，此命令無效。
0x83□□00d2	eERR_AXIS_CMD_INVALID_COMP_ENABLED	動態補償功能開啟時，此命令無效。
0x83□□00dc	eERR_AXIS_CMD_INVALID_GANTRY_MODE	軸處於龍門模式時，此命令無效。
0x83□□00e6	eERR_AXIS_CMD_INVALID_GROUPED	若軸已在群組內，此命令無效。
0x83□□00f0	eERR_AXIS_CMD_INVALID_CONTROL_MODE	在當前的控制模式下，此命令無效。
0x83□□00fa	eERR_AXIS_CMD_INVALID_OP_MODE	運行模式無效。
0x83□□0104	eERR_AXIS_CMD_INVALID_BUFFER_MODE	軸緩衝模式無效。
0x83□□010e	eERR_AXIS_CMD_INVALID_TP_ENABLED	Touch Probe 致能時，無法使用此命令。
0x83□□012c	eERR_AXIS_CMD_INVALID_PARAMETER	命令參數無效。
0x83□□0136	eERR_AXIS_CMD_INVALID_POS	軸目標位置超出允許範圍。
0x83□□0140	eERR_AXIS_CMD_INVALID_VEL	軸速度設定超出允許範圍。
0x83□□014a	eERR_AXIS_CMD_INVALID_ACC	軸加速度設定超出允許範圍。
0x83□□0154	eERR_AXIS_CMD_INVALID_DEC	軸減速度設定超出允許範圍。
0x83□□015e	eERR_AXIS_CMD_INVALID_JERK	軸急跳度 (jerk) 設定超出允許範圍。
0x83□□0168	eERR_AXIS_CMD_INVALID_SM_TIME	軸平滑時間設定超出允許範圍。
0x83□□0172	eERR_AXIS_CMD_INVALID_KILL_DEC	軸緊急減速度設定超出允許範圍。
0x83□□017c	eERR_AXIS_CMD_INVALID_VEL_SCALE	軸速度百分比設定超出允許範圍。
0x83□□0190	eERR_AXIS_COMP_NOT_CNFG	軸動態補償設定未妥善配置。

軸錯誤代碼		
錯誤代碼	錯誤名稱	說明
0x83□□01c2	eERR_AXIS_CMD_INVALID_MASTER_SLAVE_CONNECTION	主從軸關係設定無效。
0x83□□01cc	eERR_AXIS_CMD_INVALID_SLAVE_ID	從軸 ID 設定無效。
0x83□□01d6	eERR_AXIS_CMD_INVALID_GEAR_RATIO	從軸齒輪比設定超出允許範圍。
0x83□□01f4	eERR_AXIS_CMD_INVALID_ROLLOVER_POS	無效的軸單圈模式位置設置。應調整為正值。
0x83□□03f2	eERR_AXIS_DRIVE_FAULT	驅動器發生錯誤。
0x83□□03fc	eERR_AXIS_DRIVE_ABNORMAL_DISABLE	驅動器不正常解激磁。
0x83□□0406	eERR_AXIS_DRIVE_ENABLE_TOUT	激磁驅動器的時間過長。
0x83□□0410	eERR_AXIS_DRIVE_CLEAR_ERROR_TOUT	清除驅動器錯誤的時間過長。
0x83□□041a	eERR_AXIS_DRIVE_DISABLE_TOUT	解激磁驅動器的時間過長。
0x83□□0424	eERR_AXIS_DRIVE_HOME_TOUT	軸歸原點的時間過長。
0x83□□042e	eERR_AXIS_DRIVE_HOME_FAILED	軸歸原點錯誤。請檢查驅動器錯誤代碼。
0x83□□0456	eERR_AXIS_VEL_LIMIT	軸速度超過速度極限。
0x83□□0460	eERR_AXIS_ACC_LIMIT	軸加速度超過加速度極限。
0x83□□046a	eERR_AXIS_CURR_LIMIT	軸電流超過電流極限。
0x83□□0474	eERR_AXIS_DAMPINGRATIO_LIMIT	軸的阻尼比設定超出允許範圍。
0x83□□047e	eERR_AXIS_FREQUENCY_LIMIT	軸的頻率設定超出允許範圍。
0x83□□07da	eERR_AXIS_SWRL	軸位置命令碰觸到右側軟體極限。
0x83□□07e4	eERR_AXIS_SWLL	軸位置命令碰觸到左側軟體極限。
0x83□□07ee	eERR_AXIS_HWRL	軸的右側硬體極限訊號被觸發。
0x83□□07f8	eERR_AXIS_HWLL	軸的左側硬體極限訊號被觸發。
0x83□□0802	eERR_AXIS_COMP_LIMIT	軸誤差補償位置超出最大誤差極限。
0x83□□083e	eERR_AXIS_PERR	軸位置誤差超出位置誤差限制。請檢查是否有機構干涉。
0x83□□0848	eERR_AXIS_VERR	軸速度誤差超出速度誤差限制。請檢查是否有機構干涉。
0x83□□08a2	eERR_AXIS_PVT_MOTION_VEL_LIMIT	軸 PVT 運動的速度超過保護範圍。請檢查給定的參數是否有效。
0x83□□08ac	eERR_AXIS_PVT_MOTION_ACC_LIMIT	軸 PVT 運動的加速度超過保護範圍。請檢查給定的參數是否有效。
0x83□□08b6	eERR_AXIS_PVT_MOTION_INVALID_TIME	軸 PVT 運動的時序無效。請檢查給定的參數是否有效。
0x83□□0bb8	eERR_AXIS_CTRL_ERR	軸內部錯誤。
0x83□□0fa0	eERR_AXIS_CMD_GEAR_DISABLED	當電子齒輪解激磁時，電子齒輪命令不被允許執行。
0x83□□0fa1	eERR_AXIS_CMD_INVALID_AXIS_IN_CAM	當軸為電子凸輪時，電子齒輪命令無效。

軸錯誤代碼		
錯誤代碼	錯誤名稱	說明
0x83□□1388	eERR_CAM_CMD_INVALID_ENGAGE_WINDOW	電子凸輪咬合視窗超出允許範圍。
0x83□□1389	eERR_CAM_CMD_INVALID_ENGAGE_POSITION	電子凸輪咬合位置超出電子凸輪表範圍。
0x83□□138a	eERR_CAM_CMD_INVALID_MASTER_SCALE_FACTOR	電子凸輪主軸比例因子超出允許範圍。
0x83□□138b	eERR_CAM_CMD_INVALID_CAM_SCALE_FACTOR	電子凸輪比例因子超出允許範圍。
0x83□□138c	eERR_CAM_CMD_INVALID_CAMTABLE_ID	電子凸輪表 ID 超出允許範圍。
0x83□□138d	eERR_CAM_CMD_INVALID_DISENGAGE_WINDOW	電子凸輪脫離視窗超出允許範圍。
0x83□□138e	eERR_CAM_CMD_INVALID_DISENGAGE_POSITION	電子凸輪脫離位置超出允許範圍。
0x83□□138f	eERR_CAM_CMD_INVALID_OPERATION_IN_ENGAGED_STATE	咬合狀態下，電子凸輪命令無效。
0x83□□1390	eERR_CAM_CMD_INVALID_START_MODE	電子凸輪起始模式不符合有效列舉數值。
0x83□□1391	eERR_CAM_CMD_INVALID_MOVE_MODE	電子凸輪移動模式不符合有效列舉數值。
0x83□□1392	eERR_CAM_ENGAGED_FAILED	由於窗口過小，電子凸輪主軸可能通過咬合視窗。
0x83□□1393	eERR_CAM_CMD_NOTINUSE	目前不支援軌跡模式的終止。
0x83□□1394	eERR_CAM_CMD_CAM_DISABLED	電子凸輪未致能時，電子凸輪命令不被允許。
0x83□□1395	eERR_CAM_CMD_INVALID_AXIS_NOT_INCAM	當軸不為電子凸輪時，電子凸輪命令無效。
0x83□□1396	eERR_CAM_CMD_INVALID_AXIS_NOT_IN_DISENGAGED	當軸未脫離時，電子凸輪命令無效。
0x83□□1397	eERR_CAM_CMD_INVALID_AXIS_IN_GEAR	當軸為電子齒輪時，電子凸輪命令無效。

18.1.3 軸群組錯誤訊息

以下錯誤代碼會出現在軸群組發生錯誤或操作無效時。錯誤發生時，符號□□會是該軸群組的 ID，ID 會以十六進位制表示。例如：01 為軸群組編號 01；0f 為軸群組編號 15。

表 18.1.3.1

軸群組錯誤代碼		
錯誤代碼	錯誤名稱	說明
0x82□□000a	eERR_CRD_CMD_UNKNOWN	未知的軸群組命令。
0x82□□0028	eERR_CRD_CMD_AXIS_DUPLICATED	無法新增軸，因該軸已在群組內。
0x82□□0032	eERR_CRD_CMD_GRP_SIZE_EMPTY	此軸群組已空。
0x82□□003c	eERR_CRD_CMD_GRP_SIZE_FULL	此軸群組已滿，無法新增軸。
0x82□□0046	eERR_CRD_CMD_INVALID_MOVING	軸群組移動時，此命令無效。
0x82□□0050	eERR_CRD_CMD_INVALID_DISABLED	軸群組未激磁，此命令無效。
0x82□□005a	eERR_CRD_CMD_INVALID_INPUTSHAPING_PARAMETER_INCOMPLETE	軸群組塑型功能的參數不完全。
0x82□□001e	eERR_CRD_CMD_INVALID_KIN_SETTING	無效的運動類型。
0x82□□001f	eERR_CRD_CMD_INVALID_SPECIFIC_KIN	軸群組處於特殊運動學模式，此命令無效。
0x82□□006e	eERR_CRD_CMD_INVALID_STATE	軸群組在目前的運動狀態無法執行此命令。
0x82□□0078	eERR_CRD_CMD_QUEUE_FULL	請待前一命令執行完畢。
0x82□□0082	eERR_CRD_CMD_GRP_AXIS_INVALID	軸群組無效。
0x82□□00d2	eERR_CRD_CMD_INVALID_POS	軸群組的目標位置或方向超出允許範圍。
0x82□□00dc	eERR_CRD_CMD_INVALID_LIN_VEL	軸群組的線性速度設定超出允許範圍。
0x82□□00e6	eERR_CRD_CMD_INVALID_LIN_ACC	軸群組的線性加速度設定超出允許範圍。
0x82□□00f0	eERR_CRD_CMD_INVALID_LIN_DEC	軸群組的線性減速度設定超出允許範圍。
0x82□□00fa	eERR_CRD_CMD_INVALID_LIN_JERK	軸群組的線性急跳度 (jerk) 設定超出允許範圍。
0x82□□0104	eERR_CRD_CMD_INVALID_LIN_SM_TIME	軸群組的線性平滑時間設定超出允許範圍。
0x82□□010e	eERR_CRD_CMD_INVALID_DAMPINGRATIO	軸群組的阻尼比設定超出允許範圍。
0x82□□0118	eERR_CRD_CMD_INVALID_FREQUENCY	軸群組的頻率設定超出允許範圍。
0x82□□0140	eERR_CRD_CMD_INVALID_ANG_VEL	軸群組的旋轉速度設定超出允許範圍。
0x82□□014a	eERR_CRD_CMD_INVALID_ANG_ACC	軸群組的旋轉加速度設定超出允許範圍。
0x82□□0154	eERR_CRD_CMD_INVALID_ANG_DEC	軸群組的旋轉減速度設定超出允許範圍。
0x82□□015e	eERR_CRD_CMD_INVALID_ANG_JERK	軸群組的旋轉急跳度 (jerk) 設定超出允許範圍。
0x82□□0168	eERR_CRD_CMD_INVALID_ANG_SM_TIME	軸群組的旋轉平滑時間設定超出允許範圍。
0x82□□0190	eERR_CRD_CMD_INVALID_VEL_SCALE	軸群組的速度百分比超出允許範圍。
0x82□□019a	eERR_CRD_CMD_INVALID_TRANS_VEL	軸群組的路徑過渡速度無效。

軸群組錯誤代碼		
錯誤代碼	錯誤名稱	說明
0x82□□01a4	eERR_CRD_CMD_INVALID_TRANS_DIS	軸群組的路徑過渡距離無效。
0x82□□01a5	eERR_CRD_CMD_INVALID_TRANS_DEV	軸群組過渡誤差無效。
0x82□□01a6	eERR_CRD_CMD_INVALID_TRANS_CURVE	軸群組過渡曲率誤差無效。
0x82□□01b8	eERR_CRD_CMD_TRANS_MODE_UNKNOWN	未知的路徑過渡模式。
0x82□□01c2	eERR_CRD_CMD_COORD_SYS_UNKNOWN	未知的座標系統。
0x82□□01cc	eERR_CRD_CMD_BLEND_MODE_UNKNOWN	未知的 Blending 模式。
0x82□□01fe	eERR_CRD_CMD_LIN_INVALID_PARAM	線性路徑規劃的參數無效。
0x82□□0262	eERR_CRD_CMD_CIRC_INVALID_PARAM	圓弧路徑規劃的參數無效。
0x82□□026c	eERR_CRD_CMD_CIRC_INVALID_CENTER	圓弧路徑的中心點太接近起點或終點。
0x82□□0276	eERR_CRD_CMD_CIRC_ANGLE_SMALL	圓弧路徑的中心角度太小。
0x82□□0280	eERR_CRD_CMD_CIRC_INVALID_RADIUS	圓弧路徑的半徑無效。
0x82□□028a	eERR_CRD_CMD_CIRC_INVALID_COORD	圓弧路徑的座標系統無效。
0x82□□02c6	eERR_CRD_CMD_BEZIER_INVALID_PARAM	貝茲路徑規劃的參數無效。
0x82□□02d0	eERR_CRD_CMD_BSPLINE_INVALID_PARAM	BSpline 曲線路徑規劃的參數無效。
0x82□□02da	eERR_CRD_CMD_COORD_INVALID_PARAM	曲線路徑規劃的起始位置無效。
0x82□□02e4	eERR_CRD_CMD_COORD_INVALID_PARAM	座標轉換的參數無效。
0x82□□02ee	eERR_CRD_CMD_NURBS_INVALID_PARAM	NURBS 曲線路徑規劃的參數無效。
0x82□□03f2	eERR_CRD_AXIS_ABNORMALLY_DISABLED	軸群組的一軸或多軸不正常解激磁。
0x82□□03fc	eERR_CRD_AXIS_SWL	軸群組的一軸超出軟體極限。

18.2 GetSystemLastErr

用途

取得控制器的最新錯誤代碼。

語法

```
int GetSystemLastErr();
```

參數

無

回傳值

控制器的最新錯誤代碼，代碼定義請參閱 18.1.1 節。

需求版本

最低支援版本	iA Studio 1.1

18.3 GetAxisLastErr

用途

取得軸的最新錯誤代碼。

語法

```
int GetAxisLastErr(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

軸的最新錯誤代碼，代碼定義請參閱 0 節。

需求版本

最低支援版本	iA Studio 1.1

18.4 ClearAxisLastErr

用途

清除軸的最新錯誤代碼。

語法

```
int ClearAxisLastErr(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.1

18.5 GetGrpLastErr

用途

取得軸群組的最新錯誤代碼。

語法

```
int GetGrpLastErr(
    int group_id
);
```

參數

group_id [in] 軸群組編號。

回傳值

軸群組的最新錯誤代碼，代碼定義請參閱 18.1.3 節。

需求版本

最低支援版本	iA Studio 1.1

18.6 ClearGrpLastErr

用途

清除軸群組的最新錯誤代碼。

語法

```
int ClearGrpLastErr(
    int group_id
);
```

參數

group_id [in]　　軸群組編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 1.1

18.7 GetDriveErr

用途

取得驅動器的錯誤代碼。

語法

```
int GetDriveErr(
    int    axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

返回驅動器的錯誤代碼。

備註

使用此函式需將物件 0x603F(Error code)配置為 PDO。

需求版本

最低支援版本	iA Studio 3.0

19. 巨集定義與函式

19.1 _TASKID_

用途

詢問當前的 HMPL task ID。

範例

```
#if _TASKID_ == 0
int global_var = 0;  //  當前 task ID 為 0 才會被編譯
#endif

void test(){

    for (;;) {
      if (HIMC_GPI(1)) {
        StopTask( _TASKID_ );  //  停止當前的 task
      }
    }
}

void main() {
    test();
}
```

需求版本

最低支援版本	iA Studio 0.23

19.2 _AUTORUN_

用途

開機時，自動執行某 task。

範例

```
_AUTORUN_ void main() {
    Till(IsSystemOper());
    //  做點事
}
```

需求版本

最低支援版本	iA Studio 0.22

19.3 Till

用途

暫停執行 HMPL task，直到滿足特定條件。

語法

```
Till(
    condition
);
```

參數

condition [in] 型態：**int**
條件評估的結果 → **true**（非零值）或 **false**（0）

備註

呼叫此函式時，使用者須自行負責因 Till 參數的條件無法滿足，導致 HMPL 應用程式失效，引發 HIMC 的異常行為。

範例

```
void main() {
    Till(IsEnabled(0) && IsEnabled(1));

    //  做點事
}
```

需求版本

最低支援版本	iA Studio 0.23

19.4 HIMC_GPI

用途

詢問控制器通用輸入的狀態。

語法

```
HIMC_GPI(
    int gpi_idx
);
```

參數

gpi_idx [in] 通用輸入編號。

範例

```
void main() {
    //  取得特定通用輸入的狀態
    if (HIMC_GPI(4) && HIMC_GPI(6)) {
    //  若 HIMC_GPI(4)與 HIMC_GPI(6)皆為導通狀態
      //  做點事
    }
}
```

需求版本

最低支援版本	iA Studio 0.23

19.5 HIMC_GPO

用途

詢問控制器通用輸出的狀態。

語法

```
HIMC_GPO(
    int gpo_idx
);
```

參數

gpo_idx [in]　　通用輸出編號。

範例

```
void main() {
    //  取得特定通用輸出的狀態
    if (HIMC_GPO(5)) {  //  若 HIMC_GPO(5)為導通狀態
      //  做點事
    }
    HIMC_GPO(1) = HIMC_GPI(4) && HIMC_GPI(6);  //  設置特定通用輸出
}
```

需求版本

最低支援版本	iA Studio 0.23

20. 歸原點函式

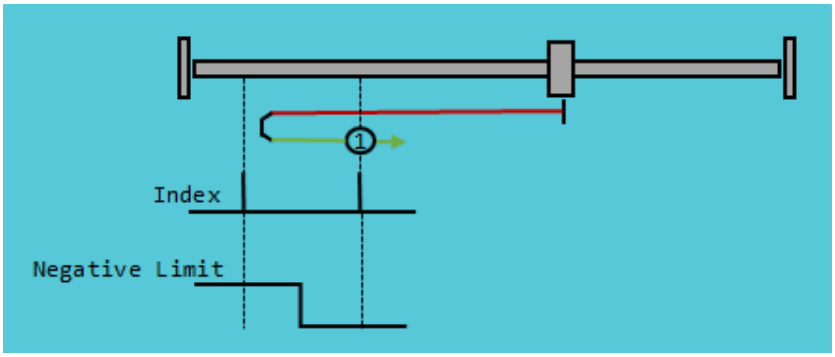
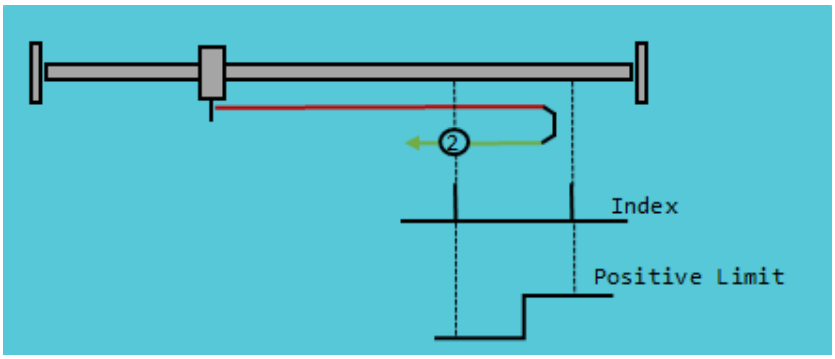
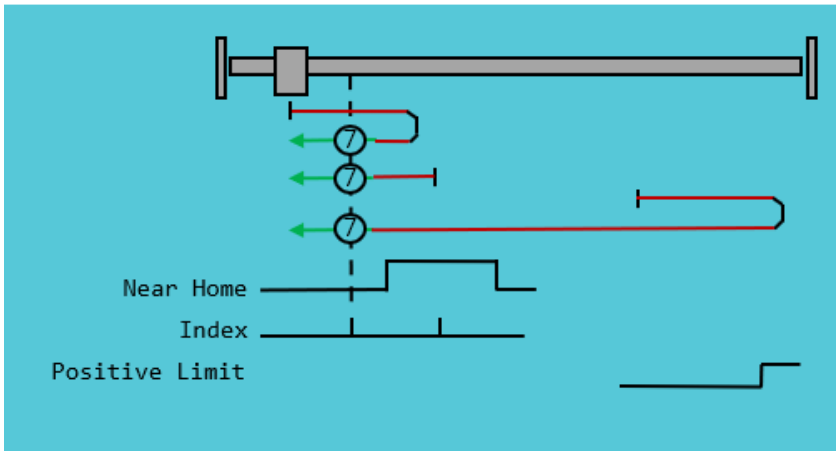
20.1 概述

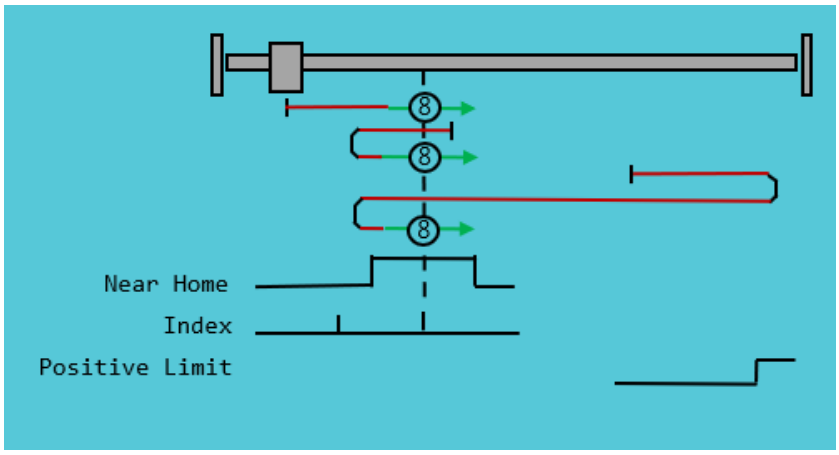
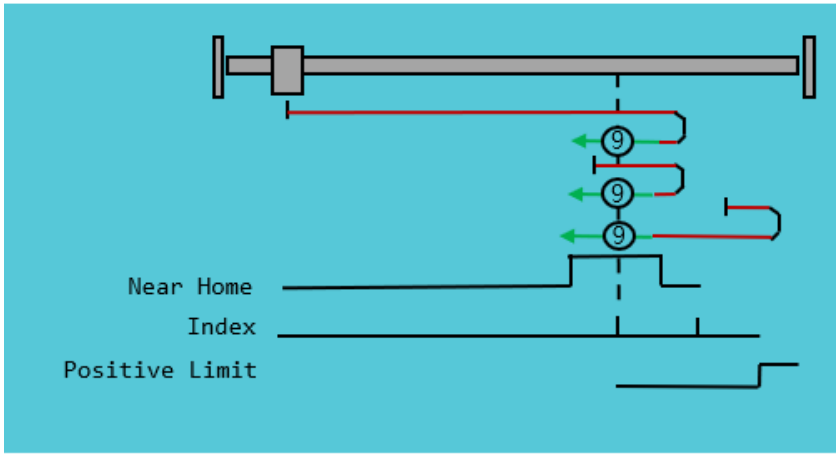
HIMC 支援 CiA 402 歸原點模式，使用者可依據機台配置設定各軸的歸原點方法，表 20.1.1 列出所有歸原點方法，詳細圖示與歸原點流程則如表 20.1.2 所示。

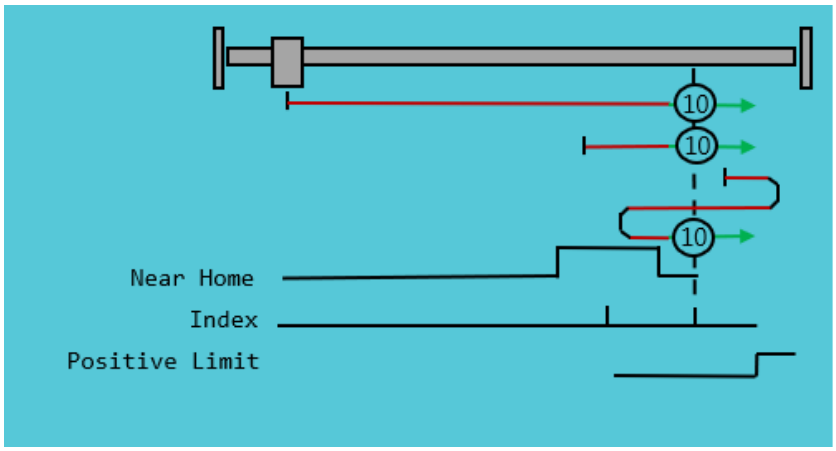
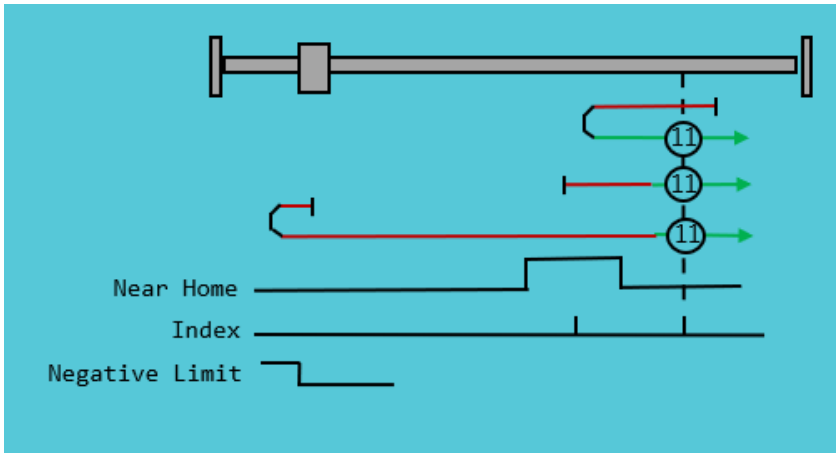
表 20.1.1

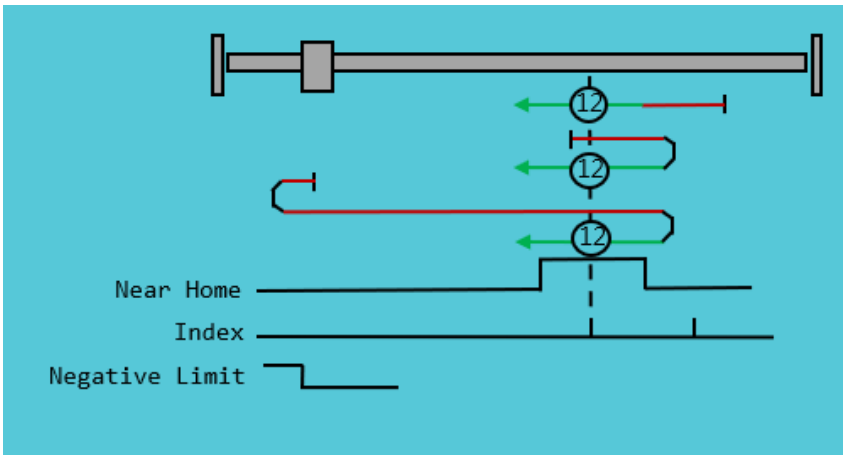
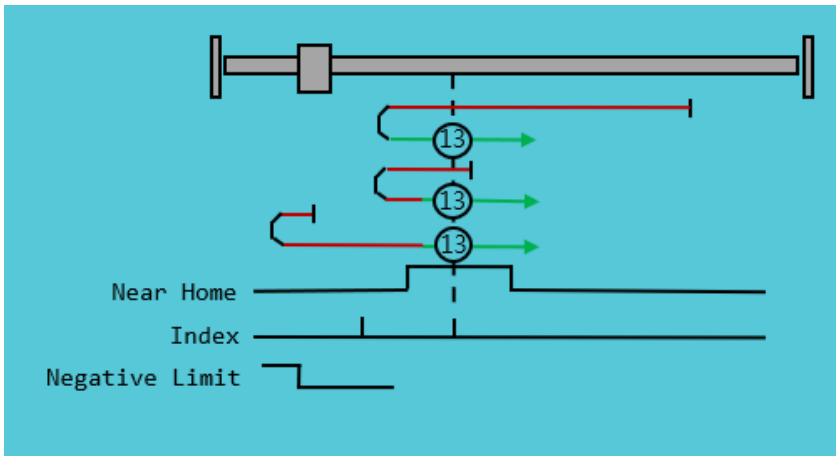
HMPL 定義	描述
HOME_METHOD_1	往負方向找負極限，再往正方向找 Index
HOME_METHOD_2	往正方向找正極限，再往負方向找 Index
HOME_METHOD_7	往正方向找近原點開關正緣左側的 Index
HOME_METHOD_8	往正方向找近原點開關正緣右側的 Index
HOME_METHOD_9	往正方向找近原點開關負緣左側的 Index
HOME_METHOD_10	往正方向找近原點開關負緣右側的 Index
HOME_METHOD_11	往負方向找近原點開關正緣右側的 Index
HOME_METHOD_12	往負方向找近原點開關正緣左側的 Index
HOME_METHOD_13	往負方向找近原點開關負緣右側的 Index
HOME_METHOD_14	往負方向找近原點開關負緣左側的 Index
HOME_METHOD_17	往負方向找負極限，再移動到原點偏移量的位置
HOME_METHOD_18	往正方向找正極限，再移動到原點偏移量的位置
HOME_METHOD_19	往負方向找負極限，再往正方向找正極限，最後移動到正負極限的中間
HOME_METHOD_33	往負方向找 Index，再移動到原點偏移量的位置
HOME_METHOD_34	往正方向找 Index，再移動到原點偏移量的位置
HOME_METHOD_37	將馬達當下位置設為原點

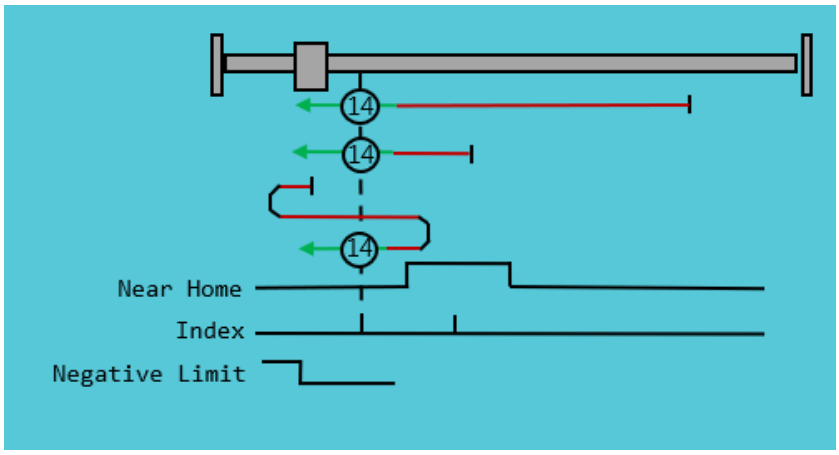
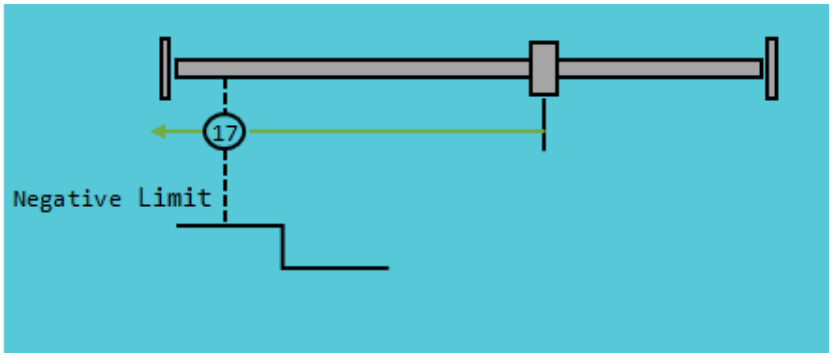
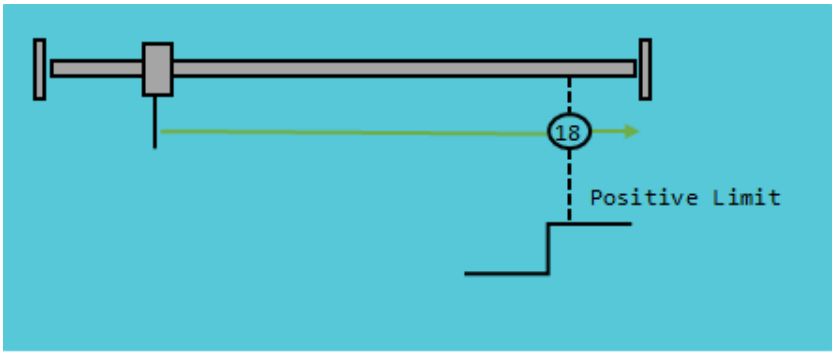
表 20.1.2

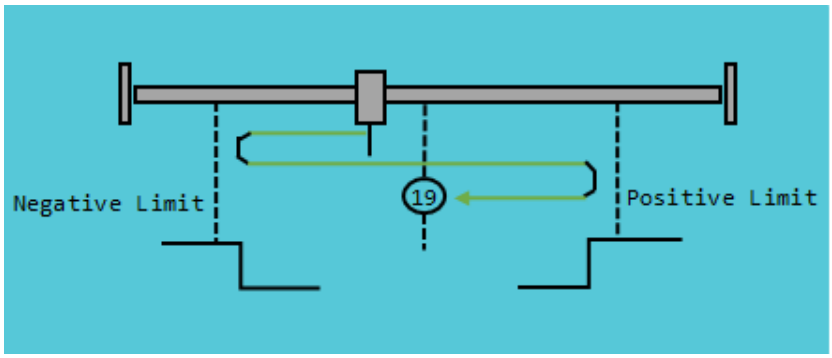
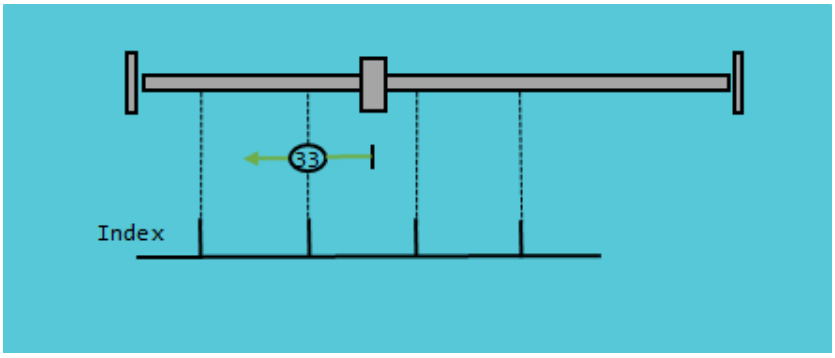
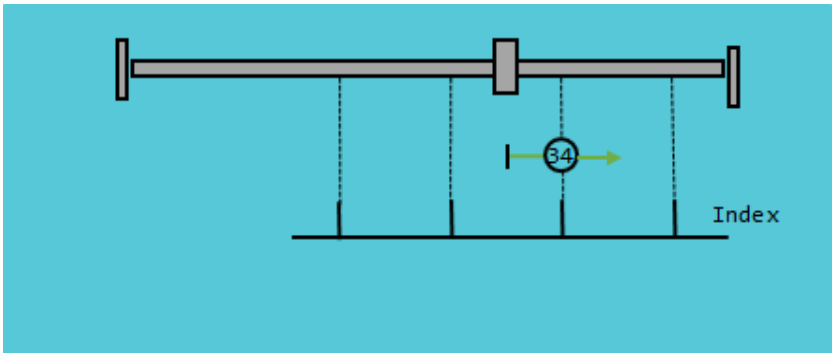
HMPL 定義	歸原點流程
HOME_METHOD_1	
	快速往負方向移動，找負極限，再慢速往正方向移動找 Index。 以 Index 位置當作原點，慢速移動到原點偏移量。
HOME_METHOD_2	
	快速往正方向移動，找正極限，再慢速往負方向移動找 Index。 以 Index 位置當作原點，慢速移動到原點偏移量。
HOME_METHOD_7	
	1. (不在近原點開關上) 往正方向移動，找近原點開關正緣，再往負方向找 Index。 2. (在近原點開關上) 往負方向移動，找近原點開關負緣，再往負方向找 Index。 3. (不在近原點開關上) 往正方向移動，找正極限，再往負方向移動，找近原點開關負緣，最後往負方向找 Index。

HMPL 定義	歸原點流程
HOME_METHOD_8	
	1. (不在近原點開關上) 往正方向移動，找近原點開關正緣，再往正方向找尋 Index。 2. (在近原點開關上) 往負方向移動，找近原點開關負緣，再往正方向找尋 Index。 3. (不在近原點開關上) 往正方向移動，找正極限，再往負方向移動，找近原點開關負緣，最後往正方向找尋 Index。
HOME_METHOD_9	
	1. (不在近原點開關上) 往正方向移動，找近原點開關負緣，再往負方向找尋 Index。 2. (在近原點開關上) 往正方向移動，找近原點開關負緣，再往負方向找尋 Index。 3. (不在近原點開關上) 往正方向移動，找正極限，再往負方向移動，找近原點開關正緣，最後往負方向找尋 Index。

HMPL 定義	歸原點流程
HOME_METHOD_10	
	1. (不在近原點開關上) 往正方向移動，找近原點開關負緣，再往正方向找尋 Index。 2. (在近原點開關上) 往正方向移動，找近原點開關負緣，再往正方向找尋 Index。 3. (不在近原點開關上) 往正方向移動，找正極限，再往負方向移動，找近原點開關正緣，最後往正方向找尋 Index。
HOME_METHOD_11	
	1. (不在近原點開關上) 往負方向移動，找近原點開關正緣，再往正方向找尋 Index。 2. (在近原點開關上) 往正方向移動，找近原點開關負緣，再往正方向找尋 Index。 3. (不在近原點開關上) 往負方向移動，找負極限，再往正方向移動，找近原點開關負緣，最後往正方向找尋 Index。

HMPL 定義	歸原點流程
HOME_METHOD_12	
	1. (不在近原點開關上) 往負方向移動，找近原點開關正緣，再往負方向找尋 Index。 2. (在近原點開關上) 往正方向移動，找近原點開關負緣，再往負方向找尋 Index。 3. (不在近原點開關上) 往負方向移動，找負極限，再往正方向移動，找近原點開關負緣，最後往負方向找尋 Index。
HOME_METHOD_13	
	1. (不在近原點開關上) 往負方向移動，找近原點開關負緣，再往正方向找尋 Index。 2. (在近原點開關上) 往負方向移動，找近原點開關負緣，再往正方向找尋 Index。 3. (不在近原點開關上) 往負方向移動，找負極限，再往正方向移動，找近原點開關正緣，最後往正方向找尋 Index。

HMPL 定義	歸原點流程
HOME_METHOD_14	
	1. (不在近原點開關上) 往負方向移動，找近原點開關負緣，再往負方向找尋 Index。 2. (在近原點開關上) 往負方向移動，找近原點開關負緣，再往負方向找尋 Index。 3. (不在近原點開關上) 往負方向移動，找負極限，再往正方向移動，找近原點開關正緣，最後往負方向找尋 Index。
HOME_METHOD_17	
	慢速往負方向移動，找負極限。負極限位置當作原點，慢速移動到原點偏移量。
HOME_METHOD_18	
	慢速往正方向移動，找正極限。正極限位置當作原點，慢速移動到原點偏移量。

HMPL 定義	歸原點流程
HOME_METHOD_19	
	慢速往負方向移動，找負極限，再慢速往正方向移動，找正極限。慢速移動到中間，中間位置當作原點。
HOME_METHOD_33	
	慢速往負方向移動，找 Index。Index 位置當作原點，慢速移動到原點偏移量。
HOME_METHOD_34	
	慢速往正方向移動，找 Index。Index 位置當作原點，慢速移動到原點偏移量。

註：歸原點程序不支援模擬器。

20.1.1 範例

範例：單軸歸原點使用 HOME_METHOD_33

```
void main()
{
    int axis_id = 0;  //  軸編號 (Axis Mode)
    int home_method = HOME_METHOD_33;  //  歸原點方法
    double fast_vel = 20;  //  歸原點速度 (找 Limit Switch)
    double slow_vel = 2;   //  歸原點速度 (找 Index)
    double acc = 2000;  // 加速度時間
    double home_offsets = 0;  //  原點偏移量
    int time_out = 10000;  //  逾時時間

    SetHomeMethod(axis_id, home_method);
    SetHomeSwitchVel(axis_id, fast_vel);
    SetHomeZeroVel(axis_id, slow_vel);
    SetHomeAcc(axis_id, acc);
    SetHomeOffset(axis_id, homeoffsets);
    SetHomeTimeout(axis_id, time_out);
    int result = MoveHome(axis_id);
}
```

20.1.2 使用者自定義歸原點程序範例

範例 1：馬達往正 / 負方向找 Index 訊號

```
//  軸歸原點的標準流程—基本版
//  ( 僅使用 touch probe )

/* 設置參數 */
int axis = 0;
double Home_vel = -20;  //  方向取決於+或-
double ind_offset = 0.0;  //  完成歸原點後相對於0的Index位置

void main()
{
    DisableTouchProbe(axis);
    Till(!IsTouchProbeEnabled(axis));
    EnableTouchProbe(axis);
    Till(IsTouchProbeEnabled(axis));

    Enable(axis);
    Till(IsEnabled(axis));
    IgnoreSWL(axis, true);

    Print("Search index...");

    MoveVel(axis,-Home_vel);

    Till(IsTouchProbeTriggered(axis) || !IsMoving(axis) || IsHWLL(axis));
    Stop(axis);
    Till(!IsMoving(axis));

    if (!IsEnabled(axis)) {goto Error;}
    else if (IsHWLL(axis)) {goto Error;}
    else if (IsTouchProbeTriggered(axis)) {
            Print("Index found.");
            double pos1, pos2;
            GetTouchProbePos(axis, &pos1);
            pos2 = GetPosFb(axis) - pos1 + ind_offset;
```

```
            SetPos(axis, pos2);
            Print("Go to zero...");
            MoveAbs(axis, 0.0);  //  往0去
            Till(!IsMoving(axis));
            if (GetRefPos(axis)==0.0 && IsEnabled(axis)) {goto HomeSuccess;}
            else {goto Error;}
            }

    Error:
    Print("Axis homing fails.");
    goto RestoreFaultResponse;

    HomeSuccess:
    Print("Axis homing succeeds.");
    goto RestoreFaultResponse;

    RestoreFaultResponse:
    IgnoreSWL(axis, false);
}
```

範例 2：馬達往正 / 負方向找極限後，再找 Index 訊號

```
//  軸歸原點的標準流程—進階版
//  （使用 touch probe 與極限開關）

/* 設置參數 */
int Homing_Type=0;  //  歸原點模式
int axis = 1;  //  配置軸
int axis_1 = 2;  //  配置 HIMC 龍門模式
double Home_vel = -20;  //  方向取決於+或-

//  Homing_Type=0 : 直接找Index而不找極限開關
//  Homing_Type=1 : 找左極限開關與Index
//  Homing_Type=2 : 將左右極限的中點設為原點

//  Type 3 & 4 適用於 HIMC 龍門模式
//  Homing_Type=3 : 歸原點動作與Homing_Type=0相同
//  Homing_Type=4 : 歸原點動作與Homing_Type=1相同

double ind_offset = 0.0;  //  完成歸原點後相對於0的Index位置

int Homing_Type_0(void);
int Homing_Type_1(void);
int Homing_Type_2(double *);
int Homing_Type_3(void);
int Homing_Type_4(void);

void main()
{
    int err=0;
    if (Homing_Type==0 || Homing_Type==1)
    {
        Print("Start single axis homing...");
        if (Homing_Type==0){
            err=Homing_Type_0();
            if (err==1) {goto Error;}
            }
        if (Homing_Type==1){
```

```
            err=Homing_Type_1();
            if (err==1) {goto Error;}
        }
        if (!IsEnabled(axis)) {goto Error;}
        else if (IsHWLL(axis)) {goto Error;}
        else if (IsTouchProbeTriggered(axis)) {

            Print("Index found.");
            double pos1, pos2;
            GetTouchProbePos(axis, &pos1);
            pos2 = GetPosFb(axis) - pos1 + ind_offset;
            SetPos(axis, pos2);

            Print("Go to zero...");
            MoveAbs(axis, 0.0);  //  往0去

            Till(!IsMoving(axis));
            if (GetRefPos(axis)==0.0 && IsEnabled(axis)) {goto HomeSuccess;}
            else {goto Error;}
        }
        else {goto Error;}
    }

    if (Homing_Type==2)
    {
        double home_pos;
        Print("Start single axis homing...");
        if (Homing_Type==2){
            err=Homing_Type_2(&home_pos);
            if (err==1) {goto Error;}
        }

        if (!IsEnabled(axis)) {goto Error;}
        else {
            Print("Hardware limit found.");
            Print("Go to home position...");
            MoveAbs(axis, home_pos);  //  往0去
            Till(!IsMoving(axis));
```

```
            SetPos(axis, 0.0);
            if (GetRefPos(axis)==0.0 && IsEnabled(axis)) {goto HomeSuccess;}
            else {goto Error;}
        }
    }

    if (Homing_Type==3 || Homing_Type==4)
    {
        Print("Start gantry pair homing...");
        if (Homing_Type==3){
            err=Homing_Type_3();
            if (err==1) {goto Error;}
        }
        if (Homing_Type==4){
            err=Homing_Type_4();
            if (err==1) {goto Error;}
        }

        if (!IsEnabled(axis)) {goto Error;}
        else if (IsHWLL(axis)) {goto Error;}
        else if (IsTouchProbeTriggered(axis))
        {
            Print("Index found.");
            double pos1, pos2, pos3, pos4;
            GetTouchProbePos(axis, &pos1);
            GetTouchProbePos(axis_1, &pos3);

            pos2 = GetPosFb(axis) - pos1 + ind_offset;
            pos4 = GetPosFb(axis_1) - pos3 + ind_offset;

            SetPos(axis, pos2);
            SetPos(axis_1, pos4);

            //  建立 HIMC 龍門模式
            Disable(axis); Disable(axis_1);
            Till(!IsEnabled(axis)&& !IsEnabled(axis_1));
            EnableGantryPair(axis, axis_1);
            Till(IsGantry(axis) && IsGantry(axis_1));
```

```
            Enable(axis);
            Till(IsEnabled(axis) && IsEnabled(axis_1));

            Print("Go to zero...");
            MoveAbs(axis, 0.0);

            Till(!IsMoving(axis));
            if (0.0 == GetRefPos(axis) && IsEnabled(axis)) {goto HomeSuccess;}
            else {goto Error;}
        }
        else {goto Error;}
    }

    Error:
    Print("Axis homing fails.");
    goto RestoreFaultResponse;

    HomeSuccess:
    Print("Axis homing succeeds.");
    goto RestoreFaultResponse;

    RestoreFaultResponse:
    IgnoreSWL(axis, false);
    IgnoreSWL(axis_1, false);
}

int Homing_Type_0()
{
    DisableTouchProbe(axis);
    Till(!IsTouchProbeEnabled(axis));
    EnableTouchProbe(axis);
    Till(IsTouchProbeEnabled(axis));

    Enable(axis);
    Till(IsEnabled(axis));
    IgnoreSWL(axis, true);
```

```
    Print("Search index...");

    MoveVel(axis, -Home_vel);

    Till(IsTouchProbeTriggered(axis) || !IsMoving(axis) || IsHWLL(axis));

    Stop(axis);
    Till(!IsMoving(axis));
    return 0;
}

int Homing_Type_1()
{
    Enable(axis);
    Till(IsEnabled(axis));
    IgnoreSWL(axis, true);
    IgnoreHWL(axis, true);

    //  找極限開關
    if (!IsHWLL(axis)) {
        Print("Search hardware left limit...");
        MoveVel(axis, Home_vel);
    }
    Till(!IsMoving(axis) || IsHWLL(axis));
    Stop(axis);
    Till(!IsMoving(axis));
    if(IsHWLL(axis)==false) {return 1;}
    Print("Hardware left limit found.");

    DisableTouchProbe(axis);
    Till(!IsTouchProbeEnabled(axis));
    EnableTouchProbe(axis);
    Till(IsTouchProbeEnabled(axis));

    Print("Search Index...");

    MoveVel(axis, -Home_vel);
```

```
    Till(IsTouchProbeTriggered(axis) || !IsMoving(axis) || IsHWRL(axis));
    Stop(axis);
    Till(!IsMoving(axis));
    return 0;
}

int Homing_Type_2(double *home_pos)
{
    Enable(axis);
    Till(IsEnabled(axis));
    IgnoreSWL(axis, true);
    IgnoreHWL(axis, true);
    double pos1, pos2;

    //  找左極限開關
    if (!IsHWLL(axis)) {
        Print("Search hardware left limit...");
        MoveVel(axis, Home_vel);
    }
    Till(IsHWLL(axis)) {
      pos1 = GetPosFb(axis);
    };

    Stop(axis);
    Till(!IsMoving(axis));
    if (IsHWLL(axis)==false) {return 1;}
    Print("Hardware left limit found.");

    //  找右極限開關
    if (!IsHWRL(axis)) {
        Print("Search hardware right limit...");
        MoveVel(axis, -Home_vel);
    }
    Till(IsHWRL(axis)) {
      pos2 = GetPosFb(axis);
    };

    Stop(axis);
```

```
    Till(!IsMoving(axis));
    if (IsHWRL(axis)==false) {return 1;}
    Print("Hardware right limit found.");

    *home_pos = (pos2+pos1)/2.0;  //  pos3為原點
    return 0;
}

int Homing_Type_3()
{
    DisableGantryPair(axis);
    Till(!IsGantry(axis) && !IsGantry(axis_1));

    DisableTouchProbe(axis); DisableTouchProbe(axis_1);
    Till(!IsTouchProbeEnabled(axis) && !IsTouchProbeEnabled(axis_1));
    EnableTouchProbe(axis); EnableTouchProbe(axis_1);
    Till(IsTouchProbeEnabled(axis) && IsTouchProbeEnabled(axis_1));

    Enable(axis); Enable(axis_1);
    Till(IsEnabled(axis) && IsEnabled(axis_1));

    IgnoreSWL(axis, true); IgnoreSWL(axis_1, true);

    MoveVel(axis, -Home_vel); MoveVel(axis_1, -Home_vel);

    Till((IsTouchProbeTriggered(axis) && IsTouchProbeTriggered(axis_1)) ||
        (!IsMoving(axis) || !IsMoving(axis_1) || IsHWLL(axis) || IsHWLL(axis_1)));

    Stop(axis); Stop(axis_1);
    Till(!IsMoving(axis) && !IsMoving(axis_1));
    return 0;
}

int Homing_Type_4()
{
    DisableGantryPair(axis);
    Till(!IsGantry(axis) && !IsGantry(axis_1));
```

```
    DisableTouchProbe(axis); DisableTouchProbe(axis_1);
    Till(!IsTouchProbeEnabled(axis) && !IsTouchProbeEnabled(axis_1));
    EnableTouchProbe(axis); EnableTouchProbe(axis_1);
    Till(IsTouchProbeEnabled(axis) && IsTouchProbeEnabled(axis_1));

    Enable(axis); Enable(axis_1);
    Till(IsEnabled(axis) && IsEnabled(axis_1));

    IgnoreHWL(axis, true); IgnoreHWL(axis_1, true);
    IgnoreSWL(axis, true); IgnoreSWL(axis_1, true);

    //  找極限開關
    if (!IsHWLL(axis) || !IsHWLL(axis_1)) {
        Print("Search hardware left limit...");
        MoveVel(axis, Home_vel);
        MoveVel(axis_1, Home_vel);
    }

    Till(!IsMoving(axis) || !IsMoving(axis_1) || IsHWLL(axis) || IsHWLL(axis_1));

    Stop(axis); Stop(axis_1);

    Till(!IsMoving(axis) && !IsMoving(axis_1));
    if (IsHWLL(axis)==false && IsHWLL(axis_1)==false) {return 1;}
    Print("Hardware left limit found.");

    MoveVel(axis, -Home_vel); MoveVel(axis_1, -Home_vel);

    Till((IsTouchProbeTriggered(axis) && IsTouchProbeTriggered(axis_1)) ||
        (!IsMoving(axis) || !IsMoving(axis_1) || IsHWLL(axis) || IsHWLL(axis_1)));

    Stop(axis); Stop(axis_1);
    Till(!IsMoving(axis) && !IsMoving(axis_1));
    return 0;
}
```

20.2 MoveHome

用途

執行軸的歸原點程序。

語法

```
int MoveHome(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

備註

使用此函式需將物件 0x6060(Mode of operation)、0x6061(Mode of operation display)配置為 PDO。

需求版本

最低支援版本	iA Studio 3.0

20.3 SetHomeMethod

用途

設置歸原點程序的歸原點方法。

語法

```
int SetHomeMethod(
    int axis_id,
    int method
);
```

參數

axis_id [in] 軸編號。

method [in] 歸原點方法編號之 HMPL 定義說明，詳情請參閱表 20.1.1 與表 20.1.2。
預設值為 HOME_METHOD_33。

歸原點方法編號	HMPL 定義說明	歸原點方法編號	HMPL 定義說明
1	HOME_METHOD_1	13	HOME_METHOD_13
2	HOME_METHOD_2	14	HOME_METHOD_14
7	HOME_METHOD_7	17	HOME_METHOD_17
8	HOME_METHOD_8	18	HOME_METHOD_18
9	HOME_METHOD_9	19	HOME_METHOD_19
10	HOME_METHOD_10	33	HOME_METHOD_33
11	HOME_METHOD_11	34	HOME_METHOD_34
12	HOME_METHOD_12	37	HOME_METHOD_37

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 2.0

20.4 SetHomeSwitchVel

用途

設置歸原點程序的快速歸原點速度。

語法

```
int SetHomeSwitchVel (
    int axis_id,
    double fast_vel
);
```

參數

axis_id [in]	軸編號。
fast_vel [in]	快速歸原點速度，預設值為 20。 參數單位：mm/s（毫米/秒）或 deg/s（角度/秒）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

20.5 SetHomeZeroVel

用途

設置歸原點程序的慢速歸原點速度。

語法

```
int SetHomeZeroVel(
    int axis_id,
    double slow_vel
);
```

參數

axis_id [in]	軸編號。
slow_vel [in]	慢速歸原點速度，預設值為 5。 參數單位：mm/s（毫米/秒）或 deg/s（角度/秒）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

20.6 SetHomeAcc

用途

設置歸原點程序的歸原點加速度。

語法

```
int SetHomeAcc(
    int axis_id,
    double acc
);
```

參數

axis_id [in]	軸編號。
acc [in]	歸原點加速度，預設值為 2000。 參數單位：mm/s^2（毫米/秒^2）或 deg/s^2（角度/秒^2）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 3.0

20.7 SetHomeOffset

用途

設置歸原點程序的原點偏移量。

語法

```
int SetHomeOffset(
    int axis_id,
    double offset
);
```

參數

axis_id [in]	軸編號。
offset [in]	原點偏移量，預設值為 0。 參數單位：mm（毫米）或 deg（角度）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 2.0

20.8 SetHomeTimeout

用途

設置歸原點程序的逾時時間。

語法

```
int SetHomeTimeout(
    int axis_id,
    int timeout
);
```

參數

axis_id [in]	軸編號。
timeout [in]	逾時時間，預設值為 120,000。 參數單位：ms（毫秒）

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

需求版本

最低支援版本	iA Studio 2.0

20.9 IsHomed

用途

詢問軸是否已完成歸原點程序。

語法

```
int IsHomed(
    int axis_id
);
```

參數

axis_id [in] 軸編號。

回傳值

若軸已完成歸原點程序，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 3.0

20.10 IsHoming

用途

詢問軸是否正在執行歸原點程序。

語法

```
int IsHoming(
    int axis_id
);
```

參數

axis_id [in]　　軸編號。

回傳值

若軸正在執行歸原點程序，將回傳 **int** 型態的值 **TRUE**（1）。否則，將回傳 **FALSE**（0）。

需求版本

最低支援版本	iA Studio 3.0

21. 通訊函式

21.1 概述

HIMC 透過 TCP/IP 協定與網路介面提供 API、Modbus 與 ASCII 三種通訊方式，可與相關裝置或使用者開發的應用程式連線，並進行通訊。在網路 Socket 通訊的架構下，HIMC 可作為伺服端（Server）或用戶端（Client），接受相關裝置或應用程式使用 HIMC API(註1)、Modbus(註2)與 ASCII 進行通訊。

HIMC 作為伺服端時，HMPL 提供 ASCII 通訊相關函式，包含 Native ASCII 與 User ASCII。Native ASCII 的預設連接埠為 3999，而 User ASCII 的預設連接埠為 4000(註3)。Native ASCII 可支援大部份的 HMPL 函式，在手冊中各函式說明的上方以  標示；而 User ASCII 透過使用者自訂的語法分析程序(請參閱 21.2.1 節 START_ASCII_AGENT)，可使用自定義的字串介面，提供更多的彈性。

HIMC 作為用戶端時，可連接其他裝置。HMPL 提供 ASCII 與 Modbus 通訊函式，ASCII 用戶端函式可對通訊裝置傳送與接收 ASCII 編碼的資料；而 Modbus 用戶端函式可對 Modbus 的記憶體區塊進行讀寫，包含 Holding Registers、Input Registers、Coils 與 Discrete Inputs。

註 1：請參閱《HIMC API 參考指南》。
註 2：請參閱《Modbus TCP 使用手冊》。
註 3：使用者可透過 iA Studio 的 IP Setting 來修改連接埠，請參閱《iA Studio 軟體使用手冊》4.12 節。

21.2 ASCII 通訊

21.2.1 START_ASCII_AGENT

用途

控制器作為伺服端，啟動使用者自定義的 ASCII 命令語法分析程序代理。

語法

```
START_ASCII_AGENT(
    parser_function
);
```

參數

parser_function [in] 解析器函式的名稱。
解析器函式應為二進位制函式，可允許 ASCII 命令輸入和輸出響應。
換句話說，其原型為：
void (*ParserFunctionPrototype)(char *command, char *response)

回傳值

無

範例 1

```
void AsciiAgent(char *cmd, char *res) {

  for (int i = 0; ; ++i){

    if (cmd[i] != '\0') {
      res[i] = cmd[i] + 1;
    } else {
      res[i] = '\0';
      break;
    }
  }
}
```

```
void main() {

    START_ASCII_AGENT(AsciiAgent);
    //  取得 ASCII 回應的方式：
    //  在任何 task 中執行 START_ASCII_AGENT，並於 Message Window 中輸入文字。
    //  若 ASCII 命令為「hello」，其轉換結果為「ifmmp」。
    //  若 ASCII 命令為「asdf」，其轉換結果為「bteg」。
}
```

範例 2

```
void AsciiAgent(char *cmd, char *res) {

    char token_str[3][40];
    int token_start = 0;
    int token_num = 0;
    for (int i = 0; i < 3; ++i){
      int token_len = StrFindChar(&cmd[token_start], ' ');
      StringCopyEx(token_str[i], &cmd[token_start], 0, token_len);
      ++token_num;
      Print("%s", token_str[i]);

      if (token_len > 0) {
        int space_len = StrFindCharEx(&cmd[token_start + token_len], " ", true);
        token_start += token_len + space_len;
      } else {
        token_start = -1;
      }
      if (token_start < 0){
        break;
      }
    }
    Print("token number: %d", token_num);

    double token2_value = 0;
    double token3_value = 0;
    if (token_num >= 2){
```

```
      token2_value = StringToDouble(token_str[1]);
      if (token_num >= 3){
        token3_value = StringToDouble(token_str[2]);
      }
    }

    if (IsStringEqual(token_str[0], "ENABLE")){
      if (token_num == 2){
        Enable(token2_value);
      }
    }
    else if (IsStringEqual(token_str[0], "MOVEABS")){
      if (token_num == 3){
        MoveAbs(token2_value, token3_value);
      }
    } else if (IsStringEqual(token_str[0], "MOVEREL")){
      if (token_num == 3){
        MoveRel(token2_value, token3_value);
      }
    } else if (IsStringEqual(token_str[0], "STOP")){
      if (token_num == 2){
        Stop(token2_value);
      }
    }
}
void main() {
    Till(IsOperMode());
    START_ASCII_AGENT(AsciiAgent);
    //  其有效命令如下所示：
    //  ENABLE 0
    //  MOVEABS 0  0.05
    //  MOVEREL 0  0.01
    //  STOP 0
}
```

需求版本

最低支援版本	iA Studio 0.23

21.2.2 ASCII_ServerBroadcast

用途

控制器作為伺服端，對所有有連線的用戶端送出訊息。

語法

```
void ASCII_ServerBroadcast(
    char *message,
    int   length
);
```

參數

message [in]　　廣播訊息字串。

length [in]　　廣播訊息字串長度，其最大值為 128。

回傳值

無

備註

在執行此函式前，須完成 User ASCII 的連線。

範例

```
void main() {
    char buf[128] = {0};
    //  利用函式 StringCopy 寫入欲傳送的字串"test"，\n 為換行，\r 為到文字行首（回車）。
    StringCopy(buf, "test\n\r");
    int len = StringLen(buf);
    ASCII_ServerBroadcast(buf, len);  //  送出字串命令
}
```

需求版本

最低支援版本	iA Studio 1.4

21.2.3 ASCII_ClientConnect

用途

控制器作為用戶端，建立與伺服端的 ASCII 通訊。

語法

```
int ASCII_ClientConnect(
    char *ip,
    char *port,
    int  *socket_id
);
```

參數

ip [in]	欲連接至伺服端的 IP 位址。
port [in]	欲連接至伺服端的通訊埠。
socket_id [out]	指標型態的記憶體，用來儲存連線成功後的 socket ID。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char ip[15] = "192.168.0.2";
    char port[5] = "1234";
    int  socket_id;
    int  err = ASCII_ClientConnect(ip, port, &socket_id);
    switch (err) {
        case 0:
            Print("Connect Sucess: Sock id %d", socket_id);
            break;
        case 0x17D5:
            Print("Connect Fail: Please check ip & port or already reach limit");
            break;
        case 0x17D4:
            Print("Connect Fail: Connect timeout");
            break;
```

```
        default:
            Print("Connect Fail");
            break;
    }
}
```

需求版本

最低支援版本	iA Studio 1.4

21.2.4 ASCII_ClientRecv

用途

控制器作為用戶端，接收伺服端所傳送的 ASCII 資料。

語法

```
int ASCII_ClientRecv(
    int  socket_id,
    int  length,
    char *buffer
);
```

參數

socket_id [in]	欲接收 ASCII 資料的 socket ID。
length [in]	欲接收 ASCII 資料的字串長度，其最大值為 512。
buffer [out]	指標型態的記憶體，用來儲存欲接收的 ASCII 資料。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char buff[200] = {0};
    int  len = 100;
    int  socket_id = 10;
    int  err = ASCII_ClientRecv(socket_id, len, buff);
    switch (err) {
        case 0:
            Print("Recv = %s", buff);
            break;
        case 0x17DF:
            Print("Recv Fail: Can't Recv from this client");
            break;
        case 0x17DE:
            Print("Recv Fail: Timeout");
            break;
```

```
        default:
            Print("Recv Fail");
            break;
    }
}
```

需求版本

最低支援版本	iA Studio 1.4

21.2.5 ASCII_ClientSend

用途

控制器作為用戶端，傳送 ASCII 資料至伺服端。

語法

```
int ASCII_ClientSend(
    int  socket_id,
    int  length,
    char *buffer
);
```

參數

socket_id [in]	欲傳送 ASCII 資料的 socket ID。
length [in]	欲傳送 ASCII 資料的字串長度，其最大值為 512。
buffer [in]	指標型態的記憶體，用來儲存欲傳送的 ASCII 資料。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char msg[20] = "Test String";
    int  len = 100;
    int  socket_id = 10;
    int  err = ASCII_ClientSend(socket_id, len, msg);
    switch (err) {
        case 0:
            Print("Send OK = %s", msg);
            break;
        case 0x17E9:
            Print("Send Fail: Can't Send from this client");
            break;
        case 0x17E8:
            Print("Send Fail: Timeout");
            break;
```

```
        default:
            Print("Send Fail");
            break;
    }
}
```

需求版本

最低支援版本	iA Studio 1.4

21.2.6 ASCII_ClientDisconnect

用途

控制器作為用戶端，結束與伺服端的 ASCII 通訊。

語法

```
int ASCII_ClientDisconnect(
    int socket_id
);
```

參數

socket_id [in]　　欲解連線的 socket ID。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char ip[15] = "192.168.0.2";
    char port[5] = "1234";
    int socket_id;
    int err = ASCII_ClientConnect(ip, port, &socket_id);
    if (err) {
        Print("Connect Fail");
        return;
    }
    //  做點事：利用函式 ASCII_ClientRecv 與 ASCII_ClientSend 讀 / 寫
    err = ASCII_ClientDisconnect(socket_id);
    if (err == 0x17D8)
        Print("Disconnect Timeout");
}
```

需求版本

最低支援版本	iA Studio 1.4

21.3 Modbus 通訊

21.3.1 Modbus_ClientConnect

用途

控制器作為用戶端，建立與伺服端的 Modbus 通訊。

語法

```
int Modbus_ClientConnect(
    char *ip,
    int  *socket_id
);
```

參數

ip [in]	欲連接至伺服端的 IP 位址。
socket_id [out]	指標型態的記憶體，用來儲存連線成功後的 socket ID。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char ip[15] = "169.254.188.17";
    int  socket_id;
    int  err = Modbus_ClientConnect(ip, &socket_id);
    switch (err) {
        case 0:
            Print("Connect Sucess: Sock id %d", socket_id);
            break;
        case 0x1839:
            Print("Connect Fail: Please check ip or already reach limit");
            break;
        case 0x1838:
            Print("Connect Fail: Connect timeout");
            break;
```

```
        default:
            Print("Connect Fail");
            break;
    }
}
```

需求版本

最低支援版本	iA Studio 2.0

21.3.2 Modbus_ClientDisconnect

用途

控制器作為用戶端，結束與伺服端的 Modbus 通訊。

語法

```
int Modbus_ClientDisconnect(
    int socket_id
);
```

參數

socket_id [in]　　欲解連線的 socket ID。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char ip[15] = "169.254.188.17";
    int socket_id;
    int err = Modbus_ClientConnect(ip, &socket_id);
    if (err) {
        Print("Connect Fail");
        return;
    }
    //  做點事：利用 Modbus 相關函式進行讀 / 寫操作
    err = Modbus_ClientDisconnect(socket_id);
    if (err == 0x183C)
        Print("Disconnect Timeout");
}
```

需求版本

最低支援版本	iA Studio 2.0

21.3.3 Modbus_ClientRead_HoldReg

用途

控制器作為用戶端，讀取伺服端的 Modbus Holding Registers 資料。

語法

```
int Modbus_ClientRead_HoldReg(
    int socket_id,
    uint16_t start_addr,
    uint16_t num_regs,
    uint8_t *output_buf,
    int *use_length
);
```

參數

socket_id [in]	欲讀取 Holding Registers 資料的 socket ID。
start_addr [in]	欲讀取 Holding Registers 資料的起始位址。
num_regs [in]	欲讀取 Holding Registers 資料的數量，其最大值為 125。
output_buf [out]	指標型態的記憶體，用來儲存欲讀取的 Holding Registers 資料。
use_length [out]	指標型態的記憶體，用來儲存 output_buf 的使用長度。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char ip[15] = "169.254.188.17";
    int  socket_id;
    int  err = Modbus_ClientConnect(ip, &socket_id);
    //  等待用戶端連線
    Sleep(5000);
    if (!err) {
        uint8_t bufs[512];
        int len = 0;
        int addr = 30;
```

```
        int regs = 1;
        err = Modbus_ClientRead_HoldReg(socket_id, addr, regs, bufs, &len);
        if(!err){
            for(int i = 0; i < len; i++){
                Print("%x", bufs[i]);
            }
        }
    }
}
```

需求版本

最低支援版本	iA Studio 2.0

21.3.4 Modbus_ClientRead_InputReg

用途

控制器作為用戶端，讀取伺服端的 Modbus Input Registers 資料。

語法

```
int Modbus_ClientRead_InputReg(
    int socket_id,
    uint16_t start_addr,
    uint16_t num_regs,
    uint8_t *output_buf,
    int *use_length
);
```

參數

socket_id [in]	欲讀取 Input Registers 資料的 socket ID。
start_addr [in]	欲讀取 Input Registers 資料的起始位址。
num_regs [in]	欲讀取 Input Registers 資料的數量，其最大值為 125。
output_buf [out]	指標型態的記憶體，用來儲存欲讀取的 Input Registers 資料。
use_length [out]	指標型態的記憶體，用來儲存 output_buf 的使用長度。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char ip[15] = "169.254.188.17";
    int  socket_id;
    int  err = Modbus_ClientConnect(ip, &socket_id);
    //  等待用戶端連線
    Sleep(5000);
    if (!err) {
        uint8_t bufs[512];
        int len = 0;
        int addr = 30;
```

```
        int regs = 1;
        err = Modbus_ClientRead_InputReg(socket_id, addr, regs, bufs, &len);
        if(!err){
            for(int i = 0; i < len; i++){
                Print("%x", bufs[i]);
            }
        }
    }
}
```

需求版本

最低支援版本	iA Studio 2.0

21.3.5 Modbus_ClientRead_Coils

用途

控制器作為用戶端，讀取伺服端的 Modbus Coils 資料。

語法

```
int Modbus_ClientRead_Coils(
    int socket_id,
    uint16_t start_addr,
    uint16_t num_coils,
    uint8_t *output_buf,
    int *use_length
);
```

參數

socket_id [in]	欲讀取 Coils 資料的 socket ID。
start_addr [in]	欲讀取 Coils 資料的起始位址。
num_coils [in]	欲讀取 Coils 資料的數量，其最大值為 2000。
output_buf [out]	指標型態的記憶體，用來儲存欲讀取的 Coils 資料。
use_length [out]	指標型態的記憶體，用來儲存 output_buf 的使用長度。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char ip[15] = "169.254.188.17";
    int  socket_id;
    int  err = Modbus_ClientConnect(ip, &socket_id);
    //  等待用戶端連線
    Sleep(5000);
    if (!err) {
        uint8_t bufs[512];
        int len = 0;
        int addr = 30;
```

```
        int coils = 40;
        err = Modbus_ClientRead_Coils(socket_id, addr, coils, bufs, &len);
        if(!err){
            for(int i = 0; i < len; i++){
                Print("%x", bufs[i]);
            }
        }
    }
}
```

需求版本

最低支援版本	iA Studio 2.0

21.3.6 Modbus_ClientRead_Inputs

用途

控制器作為用戶端，讀取伺服端的 Modbus Discrete Inputs 資料。

語法

```
int Modbus_ClientRead_Inputs(
    int socket_id,
    uint16_t start_addr,
    uint16_t num_inputs,
    uint8_t *output_buf,
    int *use_length
);
```

參數

socket_id [in]	欲讀取 Discrete Inputs 資料的 socket ID。
start_addr [in]	欲讀取 Discrete Inputs 資料的起始位址。
num_inputs [in]	欲讀取 Discrete Inputs 資料的數量，其最大值為 2000。
output_buf [out]	指標型態的記憶體，用來儲存欲讀取的 Discrete Inputs 資料。
use_length [out]	指標型態的記憶體，用來儲存 output_buf 的使用長度。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char ip[15] = "169.254.188.17";
    int  socket_id;
    int  err = Modbus_ClientConnect(ip, &socket_id);
    //  等待用戶端連線
    Sleep(5000);
    if (!err) {
        uint8_t bufs[512];
        int len = 0;
        int addr = 30;
```

```
        int inputs = 40;
        err = Modbus_ClientRead_Inputs(socket_id, addr, inputs, bufs, &len);
        if(!err){
            for(int i = 0; i < len; i++){
                Print("%x", bufs[i]);
            }
        }
    }
}
```

需求版本

最低支援版本	iA Studio 2.0

21.3.7 Modbus_ClientWrite_HoldReg

用途

控制器作為用戶端，寫入伺服端的 Modbus Holding Registers 資料。

語法

```
int Modbus_ClientWrite_HoldReg(
    int socket_id,
    uint16_t start_addr,
    uint16_t num_regs,
    uint16_t *write_data,
    uint8_t *output_buf,
    int *use_length
);
```

參數

socket_id [in]	欲寫入 Holding Registers 資料的 socket ID。
start_addr [in]	欲寫入 Holding Registers 資料的起始位址。
num_regs [in]	欲寫入 Holding Registers 資料的數量，其最大值為 123。
write_data [in]	指標型態的記憶體，用來儲存欲寫入的 Holding Registers 資料。
output_buf [out]	指標型態的記憶體，用來儲存伺服端回覆的 Holding Registers 資料。
use_length [out]	指標型態的記憶體，用來儲存 output_buf 的使用長度。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char ip[15] = "169.254.188.17";
    int  socket_id;
    int  err = Modbus_ClientConnect(ip, &socket_id);
    //  等待用戶端連線
    Sleep(5000);
    if (!err) {
        uint8_t bufs[512];
```

```
        uint16_t data[1];
        //  激磁 HIMC 第 0 軸
        data[0] = 1;
        int len = 0;
        uint16_t addr = 30;
        uint16_t regs = 1;
        err = Modbus_ClientWrite_HoldReg(socket_id, addr, regs, data, bufs, &len);
        if(!err){
            for(int i = 0; i < len; i++){
                Print("%x", bufs[i]);
            }
        }
    }
}
```

需求版本

最低支援版本	iA Studio 2.0

21.3.8 Modbus_ClientWrite_Coils

用途

控制器作為用戶端，寫入伺服端的 Modbus Coils 資料。

語法

```
int Modbus_ClientWrite_Coils(
    int socket_id,
    uint16_t start_addr,
    uint16_t num_coils,
    uint16_t *write_data,
    uint8_t *output_buf,
    int *use_length
);
```

參數

socket_id [in]	欲寫入 Coils 資料的 socket ID。
start_addr [in]	欲寫入 Coils 資料的起始位址。
num_coils [in]	欲寫入 Coils 資料的數量，其最大值為 1968。
write_data [in]	指標型態的記憶體，用來儲存欲寫入的 Coils 資料。
output_buf [out]	指標型態的記憶體，用來儲存伺服端回覆的 Coils 資料。
use_length [out]	指標型態的記憶體，用來儲存 output_buf 的使用長度。

回傳值

若函式執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳控制器的錯誤代碼，代碼定義請參閱 18.1.1 節。

範例

```
void main() {
    char ip[15] = "169.254.188.17";
    int  socket_id;
    int  err = Modbus_ClientConnect(ip, &socket_id);
    //  等待用戶端連線
    Sleep(5000);
    if (!err) {
        uint8_t bufs[512];
```

```
        uint16_t data[1];
        data[0] = 15;
        int len = 0;
        uint16_t addr = 30;
        uint16_t coils = 4;
        err = Modbus_ClientWrite_Coils(socket_id, addr, coils, data, bufs, &len);
        if(!err){
            for(int i = 0; i < len; i++){
                Print("%x", bufs[i]);
            }
        }
    }
}
```

需求版本

最低支援版本	iA Studio 2.0

22. 附錄

22.1 數學常數

表 22.1.1

名稱	描述	定義值
PI	圓周長與直徑的比值。	3.14159265358979323846
SQRT2	2 的平方根。	1.41421356237309504880
SQRT1_2	2 的平方根的倒數，即 1/2 的平方根。	0.70710678118654752440 1

22.2 系統變數

表 22.2.1

名稱	型態	描述
system_timeInMs	int	儲存 HIMC 系統時間的變數，以毫秒表示。
system_fclk	int	此變數每經過一個控制器週期，就增加 1。
system_user_table[512000]	double	使用者可自由使用的陣列，可被存入永久記憶體中。 **請參閱 11.6 節 SaveUserTable。**
system_ltest0 system_ltest1 ... system_ltest9	int	使用者可自由使用的變數。
system_dtest0 system_dtest1 ... system_dtest9	double	使用者可自由使用的變數。
system_mtest[10]	double	使用者可自由使用的陣列。

22.3 位元操作

雖無設置、清除、翻轉或檢查變數內位元的內建函式，但用戶可利用以下程式來進行位元操作。

```
#define BIT_SET(a, idx)       ((a) |= (1<<(idx)))
#define BIT_CLEAR(a, idx)     ((a) &= ~(1<<(idx)))
#define BIT_FLIP(a, idx)      ((a) ^= (1<<(idx)))
#define BIT_CHECK(a, idx)     ((a) & (1<<(idx)))
```

範例

```
#define BIT_SET(a, idx)       ((a) |= (1<<(idx)))
#define BIT_CLEAR(a, idx)     ((a) &= ~(1<<(idx)))
#define BIT_FLIP(a, idx)      ((a) ^= (1<<(idx)))
#define BIT_CHECK(a, idx)     ((a) & (1<<(idx)))

void main() {
    int bits_value = 0;

    BIT_SET(bits_value, 0);  //  此時 bits_value 的值為 1
    BIT_SET(bits_value, 3);  //  此時 bits_value 的值為 9
    BIT_CLEAR(bits_value, 0);  //  此時 bits_value 的值為 8
    BIT_FLIP(bits_value, 4);  //  此時 bits_value 的值為 24

    bits_value = 684;
    if (BIT_CHECK(bits_value, 5)) {
      Print("bit 5 is 1");
    } else {
      Print("bit 5 is 0");
    }
    //  輸出為 bit 5 is 1
}
```

(此頁有意留白。)