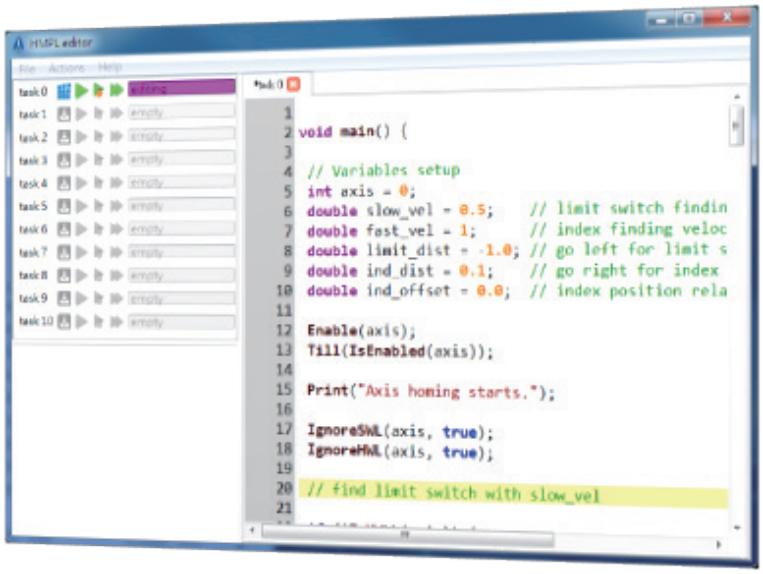
HIWIN®

Print(“Hello World”);



HIMC

HMPL 使用手冊

www.hiwinmikro.tw

修訂紀錄

發行日期	版次	適用軟體	更新內容
2019/04/02	0.1	iA Studio 1.1.3772.0	第一版發行。

目錄

目錄

目錄

目錄

目錄

目錄

目錄

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。

註 1：在 HMPL 中，所有運動相關變數均採用 MKS 單位系統（公尺–公斤–秒）。

註 2：圖示  代表該函式可在 iA Studio 的 Message Window 或自行安裝的終端機中使用。

1.2 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。

1.3 數據型態

在 HMPL 中，數據型態用於聲明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。

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	用來聲明計數器物件的型態，用於 TON 和 TOF 函式。	8	--

表 1-1

1.4 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。

類型	可視範圍	聲明處	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 task
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

表 1-2

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
//  聲明一個全域變數
int global_var = 100;

//  聲明一個全域函式
void GlobalFunction1() {
    Print("%d", global_var);
}
```

2.

```
//  撰寫於 task 1
//  聲明一個 task 變數
int task_var = 0;

//  聲明一個 task 函式
void TaskFunction1() {
    //  聲明一個區域變數
    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 IsSystemOper

用途

詢問 HIMC 系統是否處於運行（operation）狀態。若是，則可使用運動相關的函式。

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 1.1.3761.0

2.2 IsSystemPreOp

用途

詢問 HIMC 系統是否處於預運行（pre-operation）狀態。若是，HIMC 與從站間的連線已建立完成，但無法使用運動相關的函式。

語法

```
int IsSystemPreOp();
```

參數

無

回傳值

若 HIMC 系統處於預運行狀態，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 1.1.3761.0

2.3 IsSystemError

用途

詢問 HIMC 系統是否處於錯誤（error）狀態。若是，HIMC 與從站間的連線發生錯誤。

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 1.1.3761.0

2.4 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畫。

需求版本

最低支援版本	iA Studio 0.23.2087.0

2.5 StopAll

用途

以急剎加速度停止全部的軸和軸群組。

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畫。

需求版本

最低支援版本	iA Studio 0.23.1892.0

2.6 EStop

用途

緊急停止控制器。

語法

```
int EStop();
```

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 0.23.2156.0

2.7 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 1.1.3761.0

2.8 RescanMoE

用途

當 HIMC 與從站失去連線時，重新掃描 MoE。

語法

```
int RescanMoE();
```

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 1.1.3761.0

2.9 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**。

需求版本

最低支援版本	iA Studio 1.1.3761.0

2.10 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）

範例

```
int main() {
    Till(IsEnabled(0) && IsEnabled(1));

    //  做點事
}
```

需求版本

最低支援版本	iA Studio 0.23.1892.0

2.11 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。

需求版本

最低支援版本	iA Studio 0.23.1892.0

2.12 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 0.11.1555.0

2.13 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

需求版本

最低支援版本	iA Studio 0.23.1892.0

2.14 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 0.23.2033.0

2.15 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 0.23.2033.0

2.16 TON

用途

正緣觸發計數器。

當 IN 條件成立（參數轉換為 1），此函式會在 PT 毫秒後回傳回傳值。

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：millisecond（毫秒）

回傳值

若輸出信號為低態，將回傳 **int** 型態的值 **0**。若為高態，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，輸出信號被設為 1。當輸入命令下降，輸出信號被重置為 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

```
int main() {

    Timer timer1 = TimerInit;
    Timer timer2 = TimerInit;
    int var1 = 0;
    int var2 = 0;
    int var3 = 0;
    int var4 = 0;
    for (;;) {

      var1 = ...;
      var2 = ...;
      var3 = TON(&timer1, var1 > var2, 100);

      if (TON(&timer2, var3, 500)) {
        var4 = var1 + var2 + var3;
      }
    }
}
```

需求版本

最低支援版本	iA Studio 0.23.1892.0

2.17 TOF

用途

負緣觸發計數器。

當 IN 條件成立（參數轉換為 1），此函式會在 PT 毫秒後回傳回傳值。

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：millisecond（毫秒）

回傳值

若輸出信號為低態，將回傳 **int** 型態的值 **0**。若為高態，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，輸出信號被設為 1。當執行時間已經過，輸出信號被重置為 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

```
int main() {

    Timer timer1 = TimerInit;
    Timer timer2 = TimerInit;
    int var1 = 0;
    int var2 = 0;
    int var3 = 0;
    int var4 = 0;
    for (;;) {

      var1 = ...;
      var2 = ...;
      var3 = TOF(&timer1, var1 > var2, 100);

      if (TOF(&timer2, var3, 500)) {
        var4 = var1 + var2 + var3;
      }
    }
}
```

需求版本

最低支援版本	iA Studio 0.23.1892.0

2.18 _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(0)) {
         StopTask( _TASKID_ );  //  停止當前的 task
    }
  }
}

void main() {
    test();
}
```

需求版本

最低支援版本	iA Studio 0.23.2033.0

2.19 _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 0.22.1850.0

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。
以下聲明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。

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

表 3-1

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

```
void 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 0.23.1948.0

3.3 StingPrint

用途

將格式化的字串寫入記憶體中。

語法

```
void 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。 **請參閱節 3.2 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 0.23.1948.0

3.4 StringLen

用途

取得字串的長度。

語法

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

參數

str [in]

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 0.23.2151.0

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]

str2 [in]

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 0.23.2151.0

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 0.23.2151.0

3.7 StrFindCharEx

用途

找出字元集合中，任一字元或其補集合在字串中第一個出現之處。

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 0.25.2340.0

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 0.23.2151.0

3.9 StringCopy

用途

複製字串。

語法

```
void 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

無

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 0.23.2151.0

3.10 StringCopyEx

用途

複製子字串。

語法

```
void 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

無

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 0.25.2340.0

3.11 StringCat

用途

連接兩字串。

語法

```
void 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

無

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 0.23.2151.0

3.12 StringCatEx

用途

連接子字串。

語法

```
void 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

無

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 0.25.2340.0

3.13 StringToDouble

用途

將字串轉換為 double 型態。

語法

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

參數

str [in]

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 0.23.2164.0

3.14 MemoryCopy

用途

將數據從 source 記憶體複製到 destination 記憶體。

語法

```
void 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

無

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 0.23.1948.0

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

```
void 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

無

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 = 0x1111
}
```

需求版本

最低支援版本	iA Studio 0.23.1948.0

3.16 IsMemoryEqual

用途

檢查兩記憶塊是否相同。

語法

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

參數

memory_ptr1 [in]

memory_ptr2 [in]

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 0.23.1948.0

3.17 START_ASCII_AGENT

用途

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」。
}
```

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 0.23.1948.0

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 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

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，請參閱節 4.7 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 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

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 %d.", fabs(-35.91));
    //  -35.91 的絕對值為 35.91。
}
```

需求版本

最低支援版本	iA Studio 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

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），請參閱節 4.16 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 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

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 0.23.2005.0

5. 軸函式

5.1 概述

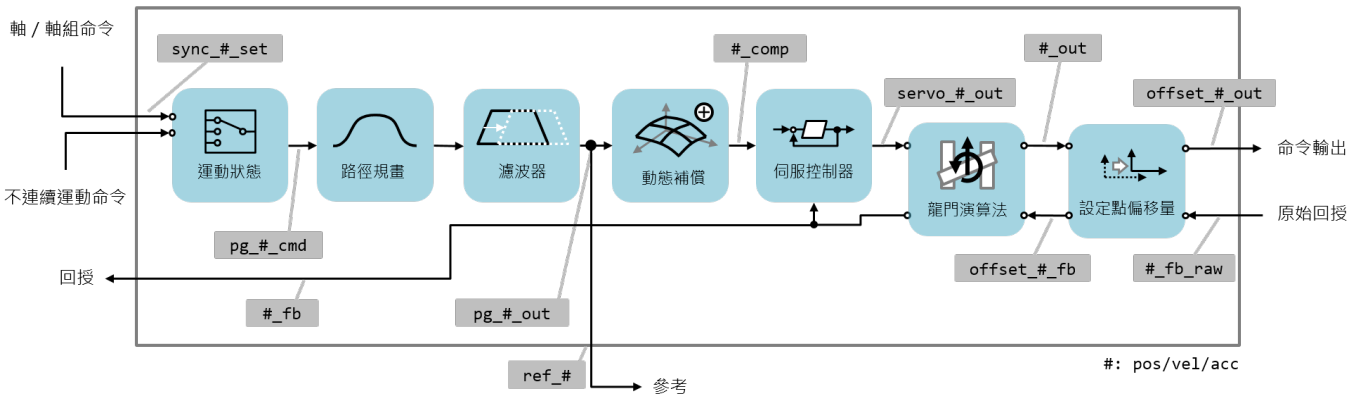


圖 5-1

5.1.1 軸變數

5.1.1.1 運動相關

表 5-1 為常見的軸變數。請在編輯框中打上**“hcv.axis[&].(變數名稱)”**以取得 iA Studio 的變數。其中，**“&”**為軸編號。

名稱	型態	單位	描述
sync_pos_set	double	公尺 或 弧度	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。
pg_pos_cmd	double	公尺 或 弧度	軌跡規畫位置命令。當軸處於不連續運動模式(點對點)時，此點為其目標位置。
ref_pos	double	公尺 或 弧度	參考位置。根據預先定義好的運動軌跡，經軌跡規畫器生產而成的位置設定點。
ref_vel	double	公尺/秒 或 弧度/秒	參考速度。根據預先定義好的運動軌跡，經軌跡規畫器生產而成的速度設定點。
ref_acc	double	公尺/秒^2 或 弧度/秒^2	參考加速度。根據預先定義好的運動軌跡，經軌跡規畫器生產而成的加速度設定點。
pos_comp_set	double	公尺 或 弧度	位置誤差補償設定點。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 。

pos_comp	double	公尺 或 弧度	補償位置。此為經動態誤差補償校正而成的位置命令。
pos_offset	double	公尺 或 弧度	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。
pos_out	double	公尺 或 弧度	位置輸出。此為無位置偏移量的軸位置命令。
offset_pos_out	double	公尺 或 弧度	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。
pos_fb_raw	double	公尺 或 弧度	原始回授位置。此為從從站讀取、並從編碼器轉換而來的回授位置。
offset_pos_fb	double	公尺 或 弧度	被偏移的回授位置。此為具位置偏移量的回授位置。
pos_fb	double	公尺 或 弧度	回授位置。位於軸座標系統中。
pos_err	double	公尺 或 弧度	跟隨誤差。此為位置輸出與原始回授位置間的差值。

表 5-1

5.1.1.2 軌跡規畫相關

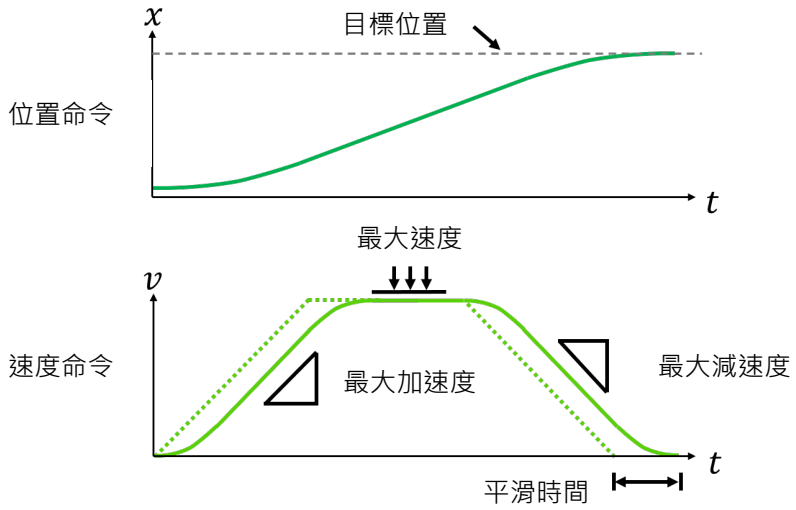


圖 5-2

表 5-2 為軌跡規畫相關變數。

名稱	型態	單位	描述
max_vel	double	公尺/秒 或 弧度/秒	最大速度。不一定要達到。
max_acc	double	公尺/秒 ^{2} 或 弧度/秒 ^{2}	最大加速度。不一定要達到。
max_dec	double	公尺/秒 ^{2} 或 弧度/秒 ^{2}	最大減速度。不一定要達到。
sm_time	double	秒	平滑時間。其輸入範圍為 0.0~0.5。增加該值以減少運動期間的機械振動。請注意，該值會影響總運動時間。

表 5-2

5.1.2 軸錯誤

變數：**fault_status**、**fault_resp**。

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。	控制器解激磁軸。
2	Position error	跟隨誤差超過保護範圍。	控制器解激磁軸。
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 5-3

5.1.3 軸狀態圖

單軸的有限狀態機與相關命令如圖 5-3。

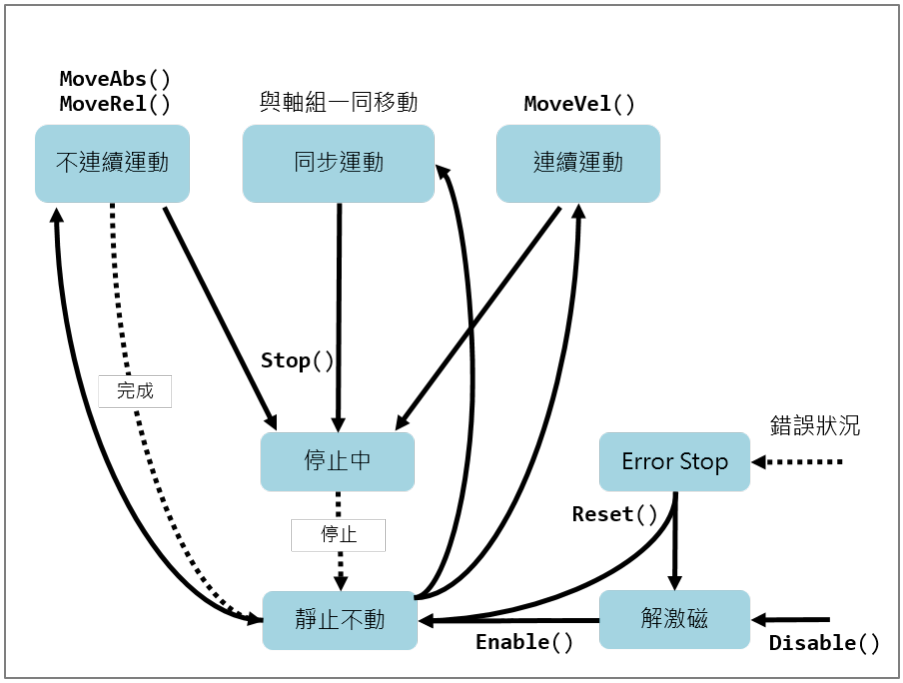


圖 5-3

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**。

需求版本

最低支援版本	iA Studio 0.23.1892.0

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

此函式會清除軸原有的路徑規畫。

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.2.3 Reset

用途

當軸處於 ErrorStop 狀態時，重新設定軸。

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

當軸處於 error stop 狀態時，執行此函式。

需求版本

最低支援版本	iA Studio 0.23.1892.0

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：meter（公尺）或 radian（弧度）

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 0.23.1892.0

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：meter（公尺）或 radian（弧度）

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 0.23.1892.0

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。 可為正也可為負，以表示運動的方向。 參數單位：m/s (公尺/秒) 或 rad/s (弧度/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 0.23.1892.0

5.2.7 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畫。

需求版本

最低支援版本	iA Studio 0.23.1892.0

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：m/s（公尺/秒）或 rad/s（弧度/秒）

需求版本

最低支援版本	iA Studio 0.23.1892.0

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：m/s (公尺/秒) 或 rad/s (弧度/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 0.23.1892.0

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：m/s^2（公尺/秒 ^{2}）或 rad/s^2（弧度/秒 ^{2}）

需求版本

最低支援版本	iA Studio 0.23.1892.0

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：m/s^2（公尺/秒 ^{2}）或 rad/s^2（弧度/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 0.23.1892.0

5.3.5 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：m/s^2（公尺/秒 ^{2}）或 rad/s^2（弧度/秒 ^{2}）

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.6 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：m/s^2（公尺/秒 ^{2}）或 rad/s^2（弧度/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 0.23.1892.0

5.3.7 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：meter（公尺）或 radian（弧度）

需求版本

最低支援版本	iA Studio 1.1.3761.0

5.3.8 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：meter（公尺）或 radian（弧度）

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 1.1.3761.0

5.3.9 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：meter（公尺）或 radian（弧度）

需求版本

最低支援版本	iA Studio 1.1.3761.0

5.3.10 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：meter（公尺）或 radian（弧度）

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 1.1.3761.0

5.3.11 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：second (秒)

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.12 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：second（秒） 輸入範圍：0.0~0.5

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 0.23.1892.0

5.3.13 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：second (秒)

需求版本

最低支援版本	iA Studio 0.24.2310.0

5.3.14 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：second (秒)

需求版本

最低支援版本	iA Studio 0.24.2310.0

5.3.15 SetPos

用途

標出軸目前位置的值。

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：meter（公尺）或 radian（弧度）

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

當軸處於同步模式、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 0.23.1892.0

5.3.16 GetPosFb

用途

取得軸的回授位置。

語法

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

參數

axis_id [in] 軸編號。

回傳值

軸的回授位置。

單位：meter（公尺）或 radian（弧度）

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.17 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：meter（公尺）或 radian（弧度）

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.18 GetPosErr

用途

取得軸的跟隨誤差。

語法

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

參數

axis_id [in] 軸編號。

回傳值

軸的跟隨誤差。

單位：meter（公尺）或 radian（弧度）

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.19 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：meter（公尺）或 radian（弧度）

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.20 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：m/s（公尺/秒）或 rad/s（弧度/秒）

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.21 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：m/s^2（公尺/秒 ^{2}）或 rad/s^2（弧度/秒 ^{2}）

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.22 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：meter（公尺）或 radian（弧度）

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.23 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：m/s（公尺/秒）或 rad/s（弧度/秒）

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.24 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：m/s^2（公尺/秒 ^{2}）或 rad/s^2（弧度/秒 ^{2}）

需求版本

最低支援版本	iA Studio 0.23.1892.0

5.3.25 IgnoreHWL

用途

忽略觸碰到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 0.23.1892.0

5.3.26 IgnoreSWL

用途

忽略觸碰到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 0.23.1892.0

5.3.27 GetAxisNum

用途

取得連接至控制器的軸數量。

語法

```
int GetAxisNum(
    void
);
```

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 0.23.2096.0

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

若已激磁軸，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 0.23.1892.0

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 0.23.1892.0

5.4.3 IsInPos

用途

詢問軸的到位狀態。若軸已到位，跟隨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 0.23.1892.0

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 0.23.1919.0

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 0.23.1892.0

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 0.23.2096.0

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，軸會遵從主站的命令。

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 0.23.1892.0

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）。

需求版本

最低支援版本	iA Studio 0.23.1892.0

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）。

需求版本

最低支援版本	iA Studio 0.23.1892.0

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 0.23.1892.0

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 0.23.1892.0

5.4.12 IsCompActive

用途

詢問補償功能是否有效。

語法

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

參數

axis_id [in] 軸編號。

回傳值

若軸處於補償功能有效狀態，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 0.23.1892.0

6. 同步運動函式

6.1 概述

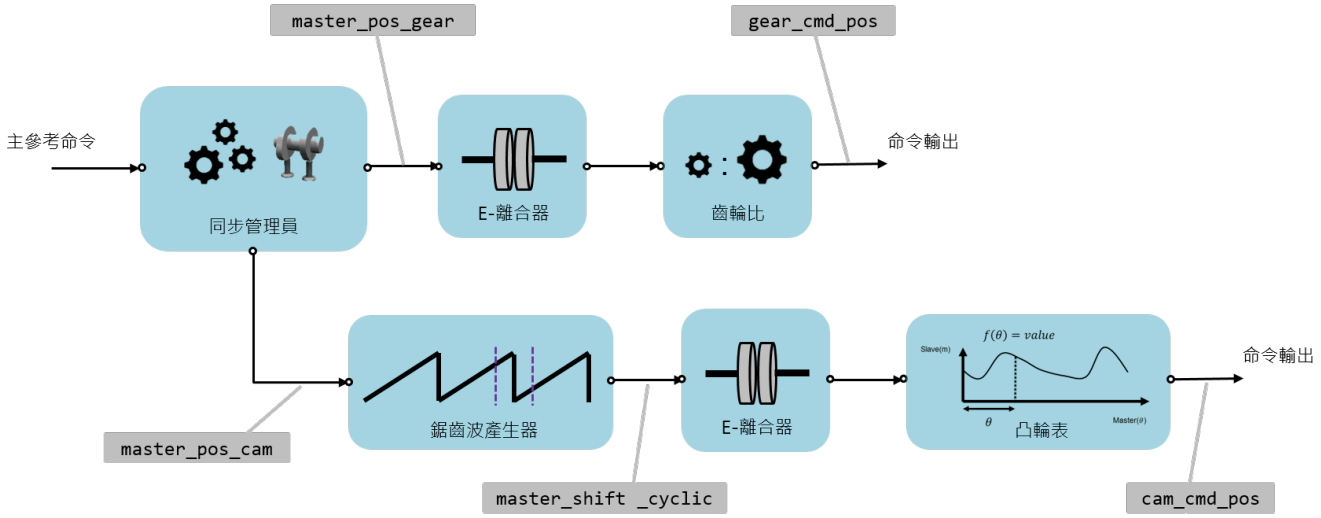


圖 6-1

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 6-2 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。

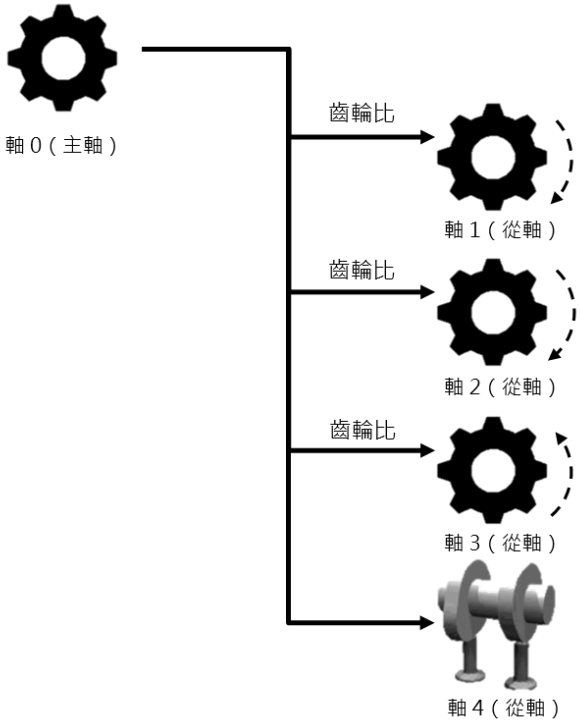


圖 6-2

6.1.1 同步運動變數

表 6-1 為常見的同步運動變數。請在編輯框中打上**“hcv.axis[&].(變數名稱)”**以取得 iA Studio 的變數。其中，**“&”**為軸編號。

名稱	型態	單位	描述
master_pos_gear	double	公尺 或 弧度	來自主軸的位置命令。
gear_cmd_pos	double	公尺 或 弧度	從軸輸出位置命令。
gear_ratio	double	公尺 或 弧度	齒輪比。

表 6-1

6.1.2 範例

```
void main()
{

    double target = 0.1;
    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 1.1.3761.0

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 1.1.3761.0

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 1.1.3761.0

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 1.1.3761.0

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）和偏擺軸（Yaw）。

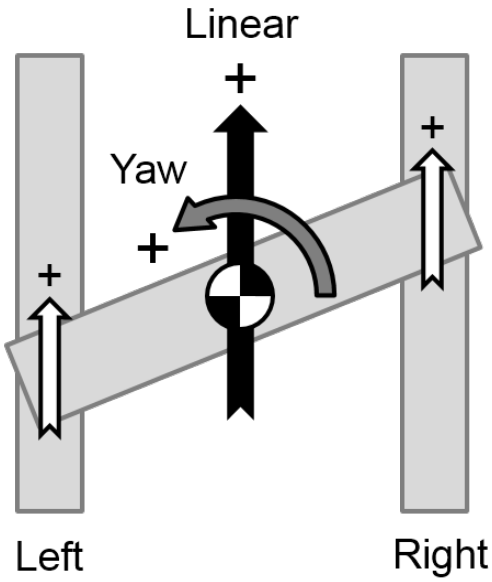


圖 7-1

線性軸以相同方向驅動右側軸和左側軸，偏擺軸則下達旋轉運動的命令。

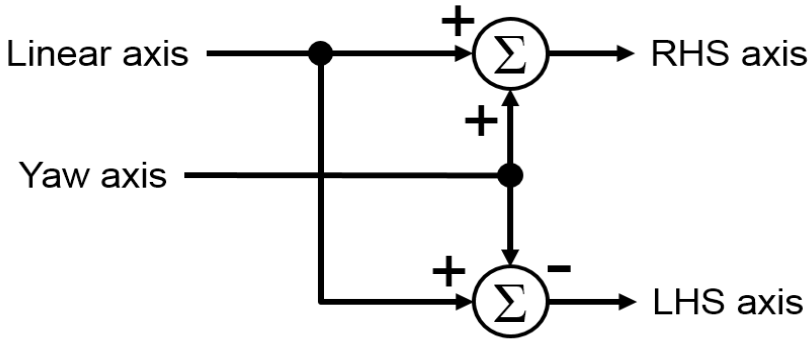


圖 7-2

線性軸的回授位置 $= \frac{(\mathrm{RHS}+\mathrm{LHS})}{2}$；偏擺軸的回授位置 $= \frac{(\mathrm{RHS}-\mathrm{LHS})}{2}$。

7.1.1 龍門設置範例

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 lhs_axis_id,
    int rhs_axis_id
);
```

參數

lhs_axis_id [in]　　左側軸之編號。

rhs_axis_id [in]　　右側軸之編號。

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

當兩個軸都處於激磁狀態時，此函式才適用。

需求版本

最低支援版本	iA Studio 0.23.1892.0

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

當兩個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 0.23.1892.0

(此頁有意留白。)

8. 軸群組函式

8.1 概述

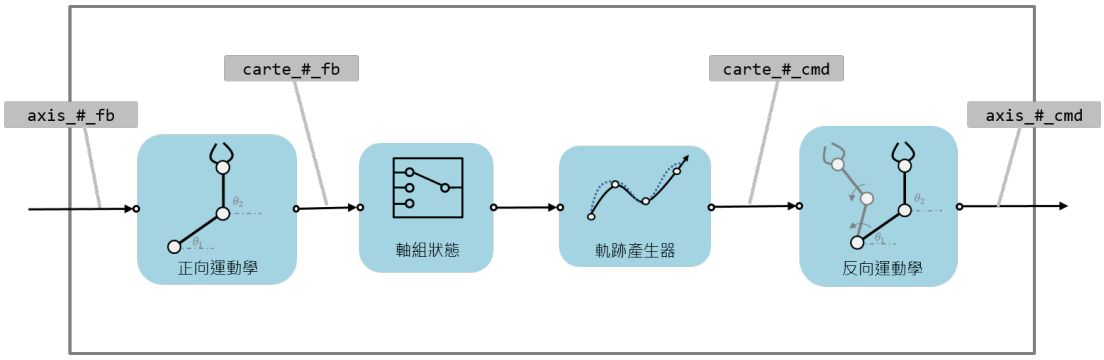

axis_#_fb
carte_#_fb
carte_#_cmd
axis_#_cmd
正向運動學
軸組狀態
軌跡產生器
反向運動學

圖 8-1

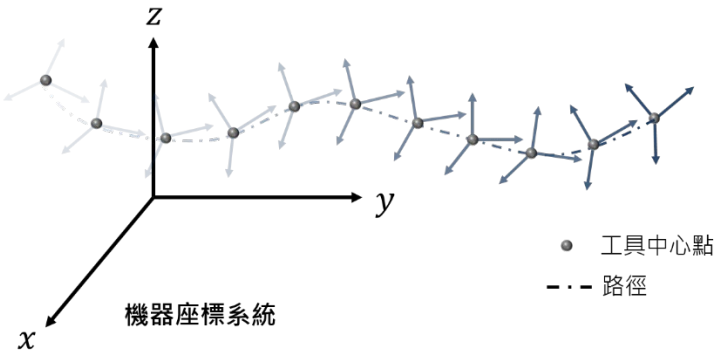

z
y
x
機器座標系統
工具中心點
路徑

圖 8-2

8.1.1 軸群組運動變數

表 8-1 為常見的軸群組變數。請在編輯框中打上**“hcv.axis_group[&].(變數名稱)”**以取得 iA Studio 的變數。其中，**“&”**為軸群組編號。

名稱	型態	單位	描述
grp_tcp_vel_norm	double	公尺/秒	工具中心點速度向量範數。為一絕對值，與機器座標系統（MCS）中的工具中心點之線性速度相符。
grp_tcp_acc_norm	double	公尺/秒 ^{2}	工具中心點加速度向量範數。為一絕對值，與機器座標系統（MCS）中的工具中心點之線性加速度相符。
axis_pos_cmd	double	公尺 或 弧度	軸空間位置命令。為一陣列，內含軸座標系統（ACS）中的位置命令。這些來自反向運動學的值將被各軸採用，並視為其目標設定點。
axis_vel_cmd	double	公尺/秒 或 弧度/秒	軸空間速度命令。為一陣列，內含軸座標系統（ACS）中的速度命令。這些來自反向運動學的值將被各軸採用，並視為其目標設定點。
axis_acc_cmd	double	公尺/秒 ^{2} 或 弧度/秒 ^{2}	軸空間加速度命令。為一陣列，內含軸座標系統（ACS）中的加速度命令。這些來自反向運動學的值將被各軸採用，並視為其目標設定點。

表 8-1

8.1.2 軸群組狀態圖

軸群組的有限狀態機與相關命令如圖 8-3。

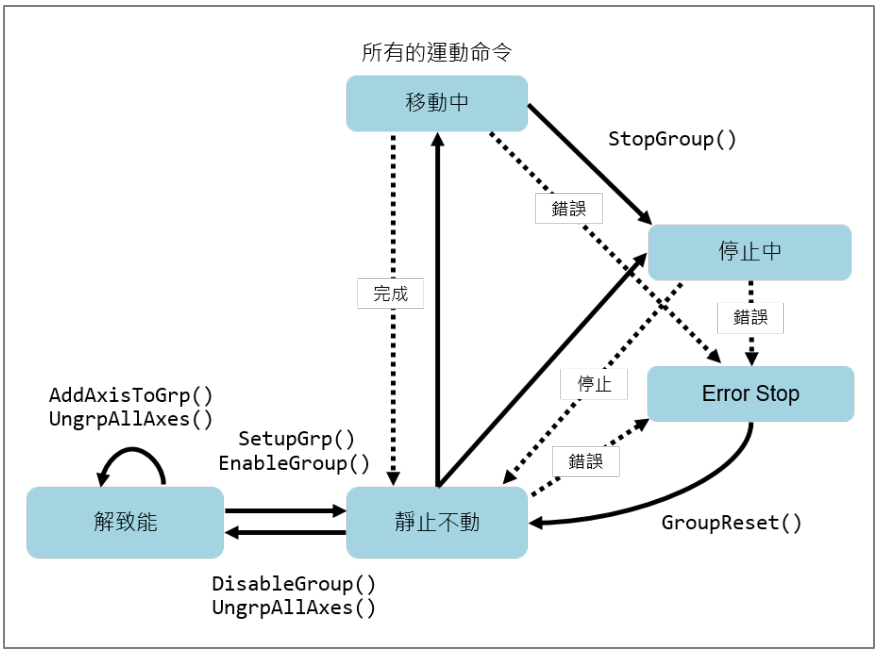


圖 8-3

8.1.3 範例

8.1.3.1 基本軸群組設置

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。

```
int 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, 0.5, 5, 5, 0.2);
    //  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, 0.1, 0.1);  //  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®節 4.1 Creating and using an AxisGroup 以獲更多資訊。

註：PLCopen®為 PLCopen 協會授權的註冊商標。

8.1.3.2 進階軸群組設置與速度交接

沿著圖 8-4 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：請參閱節 19.1 來了解 BM_PREV 與 BM_NEXT。

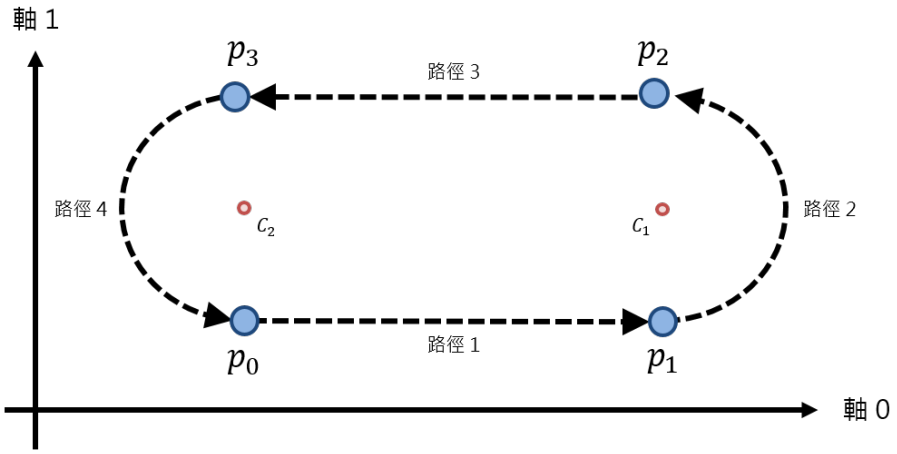


圖 8-4

```
int 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] = {0.1, 0.05, 0.0};
    double c2[3] = {0.0, 0.05, 0.0};
    double p0[6] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    double p1[6] = {0.1, 0.0, 0.0, 0.0, 0.0, 0.0};
    double p2[6] = {0.1, 0.1, 0.0, 0.0, 0.0, 0.0};
    double p3[6] = {0.0, 0.1, 0.0, 0.0, 0.0, 0.0};

    double norm_ccw[3] = {0, 0, 1};
    double vel[4] = {0.1, 5.0, 5.0, 0.05};

    LinAbs(gid, p0, vel, CS_MCS, BM_BUFF, TM_NONE, 0.0);
    Till(IsGrpInPos(gid));

    //  Blending Next 與 Blending Previous
    LinAbs(gid, p1, vel, CS_MCS, BM_PREV, TM_NONE, 0.0); //  路徑 1
    CircAbs(gid, c1, norm_ccw, 0, p2, vel, CS_MCS, BM_NEXT, TM_NONE, 0); //  路徑 2
    LinAbs(gid, p3, vel, CS_MCS, BM_PREV, TM_NONE, 0.0); //  路徑 3
    Till(IsGrpInPos(gid));

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

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 0.23.1892.0

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 0.23.1892.0

8.2.3 基本運動命令

8.2.3.1 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 = 0.1;
    double target_y = 0.1;
    LineAbs2D (0 /* group_id */, target_x, target_y);
    //  移動至 ( target_x, target_y )
    //  也就是 ( 0.1, 0.1 )
}
```

需求版本

最低支援版本	iA Studio 0.24.2326.0

8.2.3.2 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 0.24.2326.0

8.2.3.3 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 ( 0.1, 0.2 )
    double distance_x = 0.1;
    double distance_y = 0.1;
    LineRel2D (0 /* group_id */, distance_x, distance_y);
    //  移動至 ( X 起點 + distance_x, Y 起點 + distance_y )
    //  也就是 ( 0.2, 0.3 )
}
```

需求版本

最低支援版本	iA Studio 0.24.2326.0

8.2.3.4 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 0.24.2326.0

8.2.3.5 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

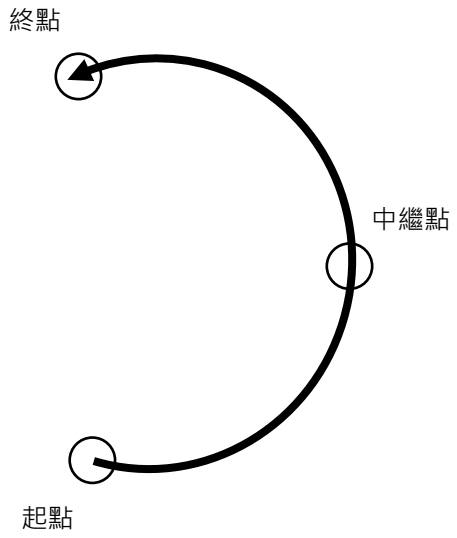
```
void main() {
    //  假設已致能內含兩正交軸的軸群組 0
    //  且兩軸的起點為 ( 0, 0 )
    double border_x = 0.1;
    double border_y = 0.1;
    double end_x = 0.2;
    double end_y = 0;
    Arc2D(0 /* group_id */, border_x, border_y, end_x, end_y);
    //  經 ( border_x , border_y ) 圓弧移動至 ( end_x, end_y )
    //  也就是經 ( 0.1, 0.1 ) 圓弧移動至 ( 0.2, 0 )
}
```

需求版本

最低支援版本	iA Studio 0.24.2326.0

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。

8.2.3.6 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**。

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 = 0.1;
    double center_y = 0;
    double end_x = 0.2;
    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 ( 0.1, 0 ) 為中心做圓周運動，並移動至 ( 0.2, 0 )
}
```

需求版本

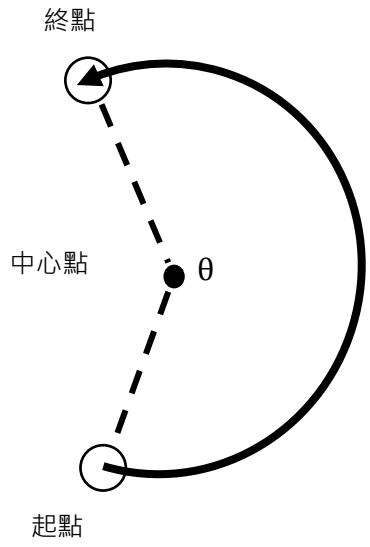
最低支援版本	iA Studio 1.1.3761.0

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	$210 + 1 \times 360°$	$570°$

8.2.4 進階運動命令

8.2.4.1 Bezier

用途

命令一個軸群組，內插貝茲曲線移動至特定座標系統中的絕對位置。

語法

```
int Bezier(
    int group_id,
    int num_point,
    double *x,
    double *y,
    double *z,
    double *abc,
    double *motion_profile
);
```

參數

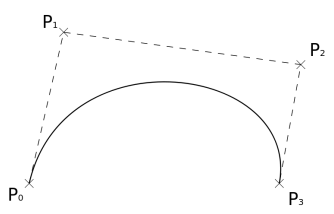
group_id [in]　　軸群組編號。

num_point [in]　　貝茲曲線的控制點數量。

輸入範圍：2~32

x [in]　　指向陣列的指標，內含控制點在 X 軸的值。

參數單位：meter (公尺)



y [in]　　指向陣列的指標，內含控制點在 Y 軸的值。

參數單位：meter (公尺)

z [in] 指向陣列的指標，內含控制點在 Z 軸的值。
參數單位：meter（公尺）

abc [in] 指向三元素陣列的指標，內含工具中心點的終點方向 { A, B, C } 。
參數單位：radian（弧度）

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

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

第一個控制點的位置須與軸群組目前的位置一致。

範例

```
void main() {
    //  假設已致能內含三正交軸的軸群組 0
    //  且三軸的起點為 ( 0, 0, 0 )
    double x[4] = {0, 0.1, 0.3, 0.5};
    double y[4] = {0, 0.5, 0, 0.2};
    double z[4] = {0, 0.1, 0.2, 0.5};
    double abc[3] = {0, 0, 0};
    double profile[4] = {0.1, 5, 5, 0.5};
    Bezier (0 /* group_id */, 4, x, y, z, abc, profile);
}
```

得到的路徑如圖 8-5。

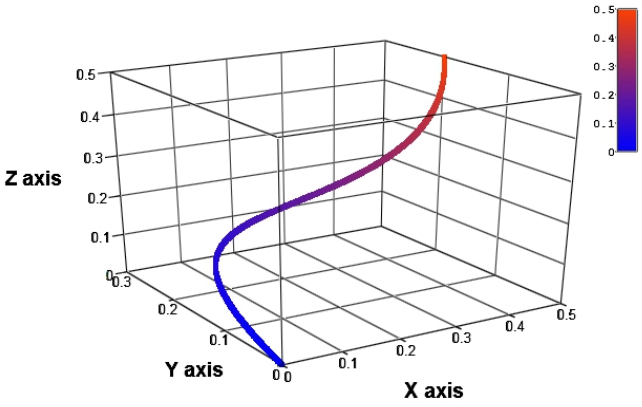


圖 8-5

需求版本

最低支援版本	iA Studio 0.23.1892.0

8.2.4.2 LinAbs

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 LinAbs(
    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 meter（公尺）；A, B, C 為 radian（弧度）。

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

coord_sys [in] 指定合適的座標系統。**請參閱節 19.3 座標系統。**

buff_mode [in] 指定路徑緩衝模式。**請參閱節 19.1 路徑緩衝模式。**

trans_mode [in] 指定路徑過渡模式。**請參閱節 19.2 路徑過渡模式。**

trans_par [in] 指定特定過渡模式的參數。若不使用，請填上 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 0.23.2170.0

8.2.4.3 LinRel

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 LinRel(
    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 meter（公尺）；A, B, C 為 radian（弧度）。

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

coord_sys [in] 指定合適的座標系統。**請參閱節 19.3 座標系統。**

buff_mode [in] 指定路徑緩衝模式。**請參閱節 19.1 路徑緩衝模式。**

trans_mode [in] 指定路徑過渡模式。**請參閱節 19.2 路徑過渡模式。**

trans_par [in] 指定特定過渡模式的參數。若不使用，請填上 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 0.23.2170.0

8.2.4.4 CircAbs

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 CircAbs(
    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：meter (公尺)

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：meter (公尺)

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 meter (公尺) ；A, B, C 為 radian (弧度) 。

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

coord_sys [in] 指定合適的座標系統。**請參閱節 19.3 座標系統。**

buff_mode [in] 指定路徑緩衝模式。**請參閱節 19.1 路徑緩衝模式。**

trans_mode [in] 指定路徑過渡模式。**請參閱節 19.2 路徑過渡模式。**

trans_par [in] 指定特定過渡模式的參數。若不使用，請填上 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 0.23.2170.0

8.2.5 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畫。

需求版本

最低支援版本	iA Studio 0.23.1960.0

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 0.23.1892.0

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 0.24.2302.0

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 0.25.2348.0

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 0.23.1892.0

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 0.23.2096.0

8.3.6 SetGrpMotionProfile

用途

在路徑上為一個群組運動設定 TCP 的最大切向運動。

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] 軸的新最大速度。
參數單位：m/s（公尺/秒）或 rad/s（弧度/秒）
輸入範圍：非零正值

max_acceleration [in] 軸的新最大加速度。
參數單位：m/s^2（公尺/秒 ^{2}）或 rad/s^2（弧度/秒 ^{2}）
輸入範圍：非零正值

max_deceleration [in] 軸的新最大減速度。
參數單位：m/s^2（公尺/秒 ^{2}）或 rad/s^2（弧度/秒 ^{2}）
輸入範圍：非零正值

smooth_time [in] 軸的新平滑時間。
參數單位：second（秒）
輸入範圍：0.0~0.5

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**。

備註

軸群組運動的參數預設值分別為 [0.1（速度）, 0.5（加速度）, 0.5（減速度）, 0.05（平滑時間）]。

需求版本

最低支援版本	iA Studio 0.24.2302.0

8.3.7 SetGrpKin

用途

設置軸群組的運動學模式（在 MCS 和 ACS 間轉換）。

語法

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

參數

group_id [in]　軸群組編號。

kin_type [in]　運動學模式（在 MCS 和 ACS 間轉換）。

請參閱節 19.3 座標系統與 19.4 運動學。

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 0.23.1892.0

8.3.8 GroupReset

用途

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

語法

```
int GroupReset(
    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 1.1.3761.0

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 0.23.1892.0

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 0.23.1892.0

8.4.3 IsGrpInPos

用途

詢問軸群組的到位狀態。若軸群組已到位，跟隨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 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 0.24.2302.0

(此頁有意留白。)

9. GPIO 函式

9.1 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]	要設置為特定輸出的狀態。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 1.1.3761.0

9.2 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 1.1.3761.0

9.3 IsGPI_On

用途

詢問控制器通用輸入的狀態。

語法

```
int IsGPI_On(
    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 0.23.1892.0

9.4 IsGPO_On

用途

詢問控制器通用輸出的狀態。

語法

```
int IsGPO_On(
    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 0.23.1892.0

9.5 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

```
int 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 0.23.1892.0

9.6 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

```
int 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 0.23.1892.0

9.7 SetSlvGPO

用途

設置從站通用輸出的狀態。

語法

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

參數

slave_id [in]	從站 ID。
gpo_idx [in]	通用輸出編號。
on_off [in]	要設置為特定輸出的狀態。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 1.1.3761.0

9.8 ToggleSlvGPO

用途

切換從站通用輸出的狀態。

語法

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

參數

slave_id [in] 從站 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**。

需求版本

最低支援版本	iA Studio 1.1.3761.0

9.9 IsSlvGPI_On

用途

詢問從站通用輸入的狀態。

語法

```
int IsSlvGPI_On(
    int slave_id,
    int gpi_idx
);
```

參數

slave_id [in]　　從站 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）。

需求版本

最低支援版本	iA Studio 1.1.3761.0

9.10 IsSlvGPO_On

用途

詢問從站通用輸出的狀態。

語法

```
int IsSlvGPO_On(
    int slave_id,
    int gpo_idx
);
```

參數

slave_id [in] 從站 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）。

需求版本

最低支援版本	iA Studio 1.1.3761.0

9.11 SLV_GPI

用途

詢問從站通用輸入的狀態。

語法

```
SLV_GPI(
    int slave_id,
    int gpi_idx
);
```

參數

slave_id [in] 從站 ID。

gpi_idx [in] 通用輸入編號。

範例

```
int main() {
    //  取得特定通用輸入的狀態
    if (SLV_GPI(0, 4) && SLV_GPI(0, 6)) {
    //  若 SLV_GPI(0, 4)與 SLV_GPI(0, 6)皆為導通狀態
      //  做點事
    }
}
```

需求版本

最低支援版本	iA Studio 0.23.1892.0

9.12 SLV_GPO

用途

詢問從站通用輸出的狀態。

語法

```
SLV_GPO(
    int slave_id,
    int gpo_idx
);
```

參數

slave_id [in] 從站 ID。

gpo_idx [in] 通用輸出編號。

範例

```
int main() {
    //  取得特定通用輸出的狀態
    if (SLV_GPO(0, 1)) {  //  若 SLV_GPO(0, 1)為導通狀態
      //  設定特定通用輸出
      SLV_GPO(0, 1) = 0;
    }
}
```

需求版本

最低支援版本	iA Studio 0.23.1892.0

(此頁有意留白。)

10. User Table 函式

10.1 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 0.23.1892.0

10.2 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 0.23.1892.0

10.3 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 1.1.3761.0

10.4 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 1.1.3761.0

10.5 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 0.23.2359.0

10.6 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 0.23.2359.0

(此頁有意留白。)

11. 位置觸發函式

11.1 概述

使用者可利用 HMPL 命令，在部分 HIWIN 驅動器中操作 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。

11.1.1 PT 變數

表 11-1 為操作 PT 相關功能的變數介紹。

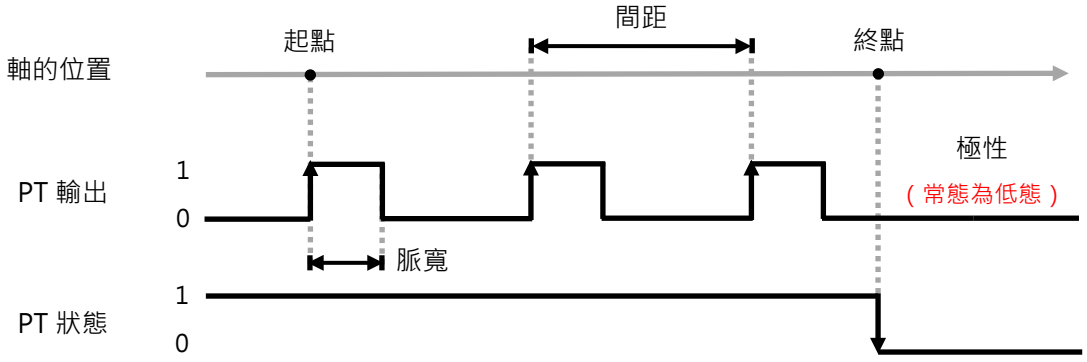


圖 11-1

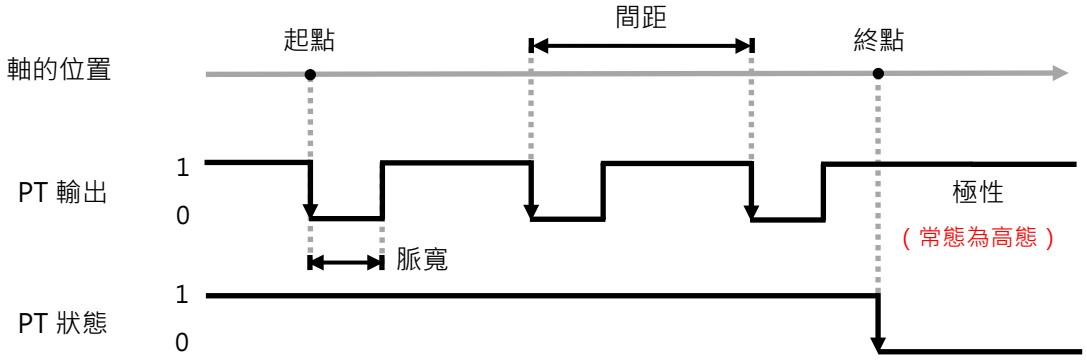


圖 11-2

使用者可自定義極性。圖 11-1 中，極性被設定為「常態為低態」。圖 11-2 中，極性被設定為「常態為高態」。節 11.9 SetPT_Polarity 以圖 11-1 為例。

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	公尺 或 弧度	PT 功能的起點。PT 輸出信號序列從此點開始。	SetPT_StartPos
終點	double	公尺 或 弧度	PT 功能的終點。此點後不再發送 PT 輸出信號。	SetPT_EndPos
間距	double	公尺 或 弧度	連續 PT 輸出的位置間距。	SetPT_Interval
脈寬	int	奈秒	每個 PT 輸出信號的寬度。 範圍為 25 奈秒至 100,000 奈秒，25 奈秒為最小增加單位。例如，25、50、… 100,000。	SetPT_PulseWidth
極性	int	0 / 1	電子信號極性輸出。 0 表示常態為低態，1 表示常態為高態。	SetPT_Polarity

表 11-1

11.2 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**。

需求版本

最低支援版本	iA Studio 0.23.1993.0

11.3 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**。

需求版本

最低支援版本	iA Studio 0.23.1993.0

11.4 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）。

需求版本

最低支援版本	iA Studio 0.23.1993.0

11.5 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：meter（公尺）或 radian（弧度）

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 0.23.1993.0

11.6 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：meter（公尺）或 radian（弧度）

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 0.23.1993.0

11.7 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：meter（公尺）或 radian（弧度）

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 0.23.1993.0

11.8 SetPT_PulseWidth

用途

設置位置觸發功能的脈寬。

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 輸出信號的寬度。 參數單位：nanosecond（奈秒） 輸入範圍：25~100,000（25 為最小增加單位）

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 0.23.1993.0

11.9 SetPT_Polarity

用途

設置位置觸發功能的極性。

語法

```
int SetPT_Polarity(
    int axis_id,
    int is_normal_high
);
```

參數

axis_id [in] 軸編號。

is_normal_high [in] 電子信號極性輸出。0 表示常態為低態，1 表示常態為高態。

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**非零值**。

需求版本

最低支援版本	iA Studio 0.23.1993.0

(此頁有意留白。)

12. Touch Probe 函式

12.1 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**。

需求版本

最低支援版本	iA Studio 0.23.1892.0

12.2 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**。

需求版本

最低支援版本	iA Studio 0.23.1892.0

12.3 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）。

需求版本

最低支援版本	iA Studio 0.23.1892.0

12.4 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）。

需求版本

最低支援版本	iA Studio 0.23.1892.0

12.5 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：meter（公尺）或 radian（弧度）

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 gantry 模式，此函式不適用。

需求版本

最低支援版本	iA Studio 0.23.1892.0

13. 動態誤差補償函式

13.1 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 0.23.1892.0

13.2 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。

需求版本

最低支援版本	iA Studio 0.23.1892.0

13.3 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]　　基值（補償輸入的最小值）。
參數單位：meter（公尺）或 radian（弧度）

interval [in]　　相鄰補償點的間距。
參數單位：meter（公尺）或 radian（弧度）

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**。

備註

當被補償的軸處於解激磁狀態，此函式不適用。

範例

在此範例中，軸 0 的位置（輸出）依軸 1 的位置設定點（輸入）進行補償。其關係如圖 13-1。

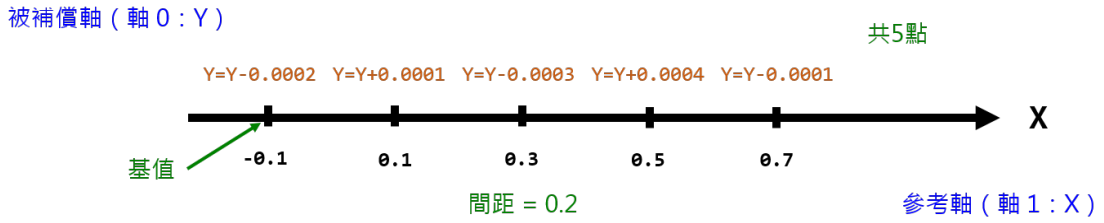


圖 13-1

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] = {-2e-4, 1e-4, -3e-4, 4e-4, -1e-4};
    SetUserTable(1, 5, data);

    SetupComp(
      0,    //  被補償軸
      1,    //  User Table 中，補償點的起始編號
      -0.1, //  基值
      0.2,  //  間距
      5,    //  補償點的數量（含基值）
      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() {
    DisableComp(0);  //  取消對軸 0 的補償
}
```

需求版本

最低支援版本	iA Studio 0.23.1892.0

13.4 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，內含各維的基值（補償輸入的最小值）。
參數單位：meter（公尺）或 radian（弧度）

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：meter（公尺）或 radian（弧度）

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**。

備註

當被補償的軸處於解激磁狀態，此函式不適用。

範例

在此範例中，軸 2 的位置（輸出）依軸 0 與軸 1 的位置設定點（輸入）進行補償。其關係如圖 13-2。

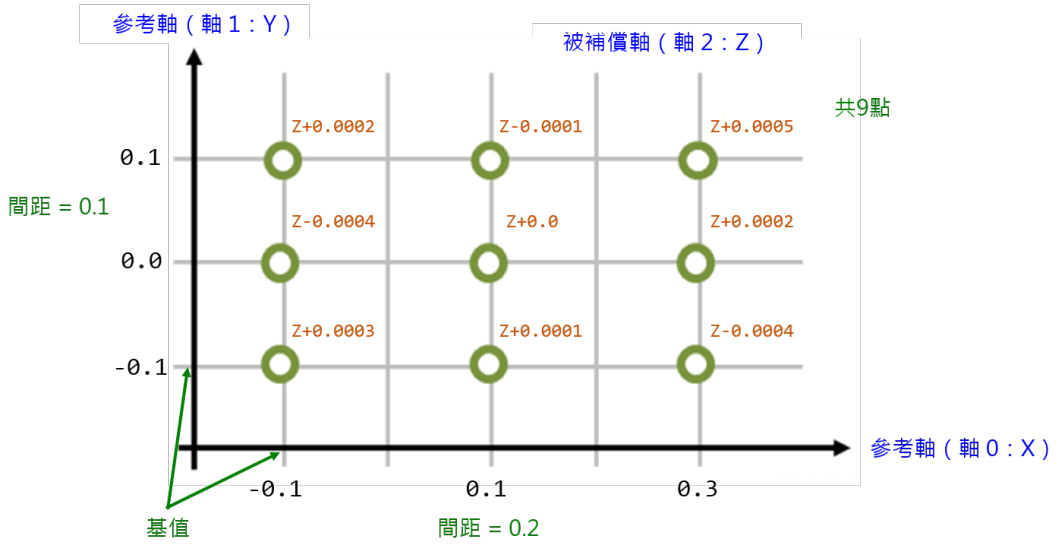


圖 13-2

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] = {
      3e-4, 1e-4,-4e-4,
      -4e-4,  0.0, 2e-4,
      2e-4,-1e-4, 5e-4
    };
    SetUserTable(3, 9, data);

    double base[2] = {-0.1, -0.1};
    double interval[2] = {0.2, 0.1};
    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,  //  基值
        interval,    //  間距
        num_pt,      //  補償點的數量（含基值）
        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() {
    DisableComp(2);  //  取消對軸 2 的補償
}
```

需求版本

最低支援版本	iA Studio 0.23.1892.0

(此頁有意留白。)

14. 濾波器函式

14.1 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畫時間，減少整定時間。

14.1.1 範例

設置 InShape 濾波器的方式如以下 HMPL task 所示。

```
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
}
```

14.2 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 1.1.3761.0

14.3 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 1.1.3761.0

14.4 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 1.1.3761.0

14.5 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 1.1.3761.0

14.6 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 1.1.3761.0

14.7 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 1.1.3761.0

14.8 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 1.1.3761.0

14.9 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 1.1.3761.0

14.10 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 1.1.3761.0

(此頁有意留白。)

15. HMPL Task 函式

15.1 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 0.22.1862.0

15.2 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 0.22.1862.0

15.3 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 0.22.1862.0

15.4 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 0.23.2114.0

15.5 IsTaskStop

用途

詢問是否已停止執行 HMPL task。

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 0.23.1919.0

16. 變數操作函式

16.1 GetSlvVar

用途

取得從站的變數值。

語法

```
double GetSlvVar(
    int slave_id,
    const char *var_name
);
```

參數

slave_id [in]　　從站編號。

var_name [in]　　指標型態的記憶體，用來儲存變數名稱。

回傳值

變數值。

需求版本

最低支援版本	iA Studio 0.23.1892.0

16.2 GetSlvVarEx

用途

取得從站的變數值與執行結果。

語法

```
double GetSlvVarEx(
    int slave_id,
    const char *var_name,
    int *result
);
```

參數

slave_id [in]	從站編號。
var_name [in]	指標型態的記憶體，用來儲存變數名稱。
result [out]	指標型態的記憶體，用來儲存執行結果。 若執行成功，將回傳 **int** 型態的值 **0**。若失敗，則回傳**非零值**。

回傳值

變數值。

需求版本

最低支援版本	iA Studio 0.23.1892.0

16.3 SetSlvVar

用途

設置從站的變數值。

語法

```
int SetSlvVar(
    int slave_id,
    const char *var_name,
    double value
);
```

參數

slave_id [in]	從站編號。
var_name [in]	指標型態的記憶體，用來儲存變數名稱。
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**。

需求版本

最低支援版本	iA Studio 0.23.1892.0

16.4 GetSlvSt

用途

取得從站的狀態。

語法

```
int GetSlvSt(
    int slave_id,
    const char *st_name
);
```

參數

slave_id [in]　　從站編號。

st_name [in]　　指標型態的記憶體，用來儲存狀態名稱。

回傳值

從站的狀態。

需求版本

最低支援版本	iA Studio 1.1.3761.0

16.5 SetSlvSt

用途

設置從站的狀態。

語法

```
int SetSlvSt(
    int slave_id,
    const char *st_name,
    int on_off
);
```

參數

slave_id [in]　　從站編號。

st_name [in]　　指標型態的記憶體，用來儲存狀態名稱。

on_off [in]　　新的從站狀態。

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**非零值**。

需求版本

最低支援版本	iA Studio 1.1.3761.0

16.6 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] hcv ID。

註：從 iA_Studio 資料夾中的 hcv_id.h 檔案取得此 ID。

result [out] 若函式執行失敗，將回傳**-1**。

回傳值

變數值。

範例

```
void main()
{
    int result = 0;
    double SW_RL = GetConfigVar(0x83000065, &result);
    Print("SW_RL = %f", SW_RL);
}
```

需求版本

最低支援版本	iA Studio 1.1.3761.0

16.7 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] hcv ID。

註：從 iA_Studio 資料夾中的 hcv_id.h 檔案取得此 ID。

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 1.1.3761.0

17. 錯誤函式

17.1 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。

需求版本

最低支援版本	iA Studio 1.1.3761.0

17.2 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。

需求版本

最低支援版本	iA Studio 1.1.3761.0

17.3 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 1.1.3761.0

17.4 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。

需求版本

最低支援版本	iA Studio 1.1.3761.0

17.5 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 1.1.3761.0

18. 歸原點範例

18.1 基本版（僅使用 index 信號）

```
//  軸歸原點的標準流程—基本版
//  單軸歸原點（僅使用 touch probe）
//  HIMC 版本 1.0

/* 設置參數 */
int axis = 1;
double Home_vel = -0.02;  //  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);
}
```

18.2 進階版（使用 index 信號與極限開關）

```
//  軸歸原點的標準流程—進階版
//  單軸與龍門歸原點（使用 touch probe 與極限開關）
//  HIMC 版本 1.0

/* 設置參數 */
int Homing_Type=0;  //  歸原點模式
int axis = 1;  //  配置軸
int axis_1 = 2;  //  配置龍門
double Home_vel = -0.02;  //  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 適用於龍門
//  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);

            //  建立龍門
            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);
```

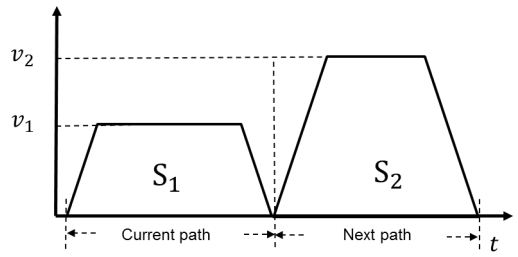
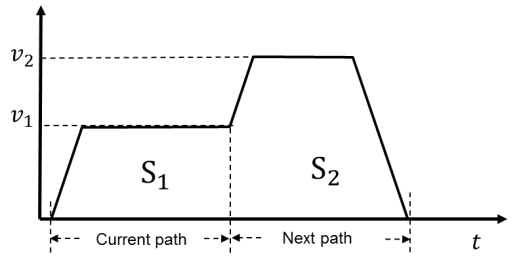
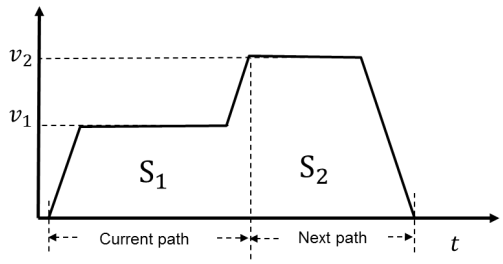
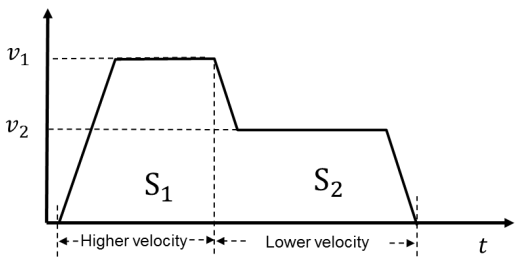
```
    Till(!IsGantry(axis) && !IsGantry(axis_1));

    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;
}
```

(此頁有意留白。)

19. 附錄

19.1 路徑緩衝模式

緩衝模式決定了相鄰路徑端點處的速度。

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） 

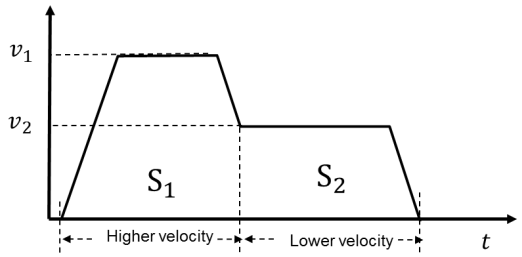
BM_LOW	其交接速度為兩路徑中較低的速度。(Blending) 

表 19-1

19.2 路徑過渡模式

過渡模式決定了相鄰路徑間的過渡曲線類型。

HMPL 定義	描述	相關參數
TM_NONE	無：不插入過渡曲線 (預設模式)	無
TM_START_VEL	Start velocity (不使用)	TPStartVelocity (單位：公尺/秒 或 弧度/秒)
TM_CONST_VEL	Constant velocity (不使用)	TPVelocity (單位：公尺/秒 或 弧度/秒)
TM_CORNER_DIST	Corner distance (不使用)	TPCornerDistance (單位：公尺 或 弧度)
TM_MAX_CORNER_DEV	Max. corner deviation (不使用)	TPCornerDeviation (單位：公尺 或 弧度)

表 19-2

19.3 座標系統

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，藉運動轉換（向前與向後轉換）與 ACS 連接。為一想像空間，最多有 6 個維度（ 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。

表 19-3

圖 19-1 以具有兩旋轉軸的 SCARA 機器人為例，展示了 ACS、MCS 與 PCS 之間的關係。

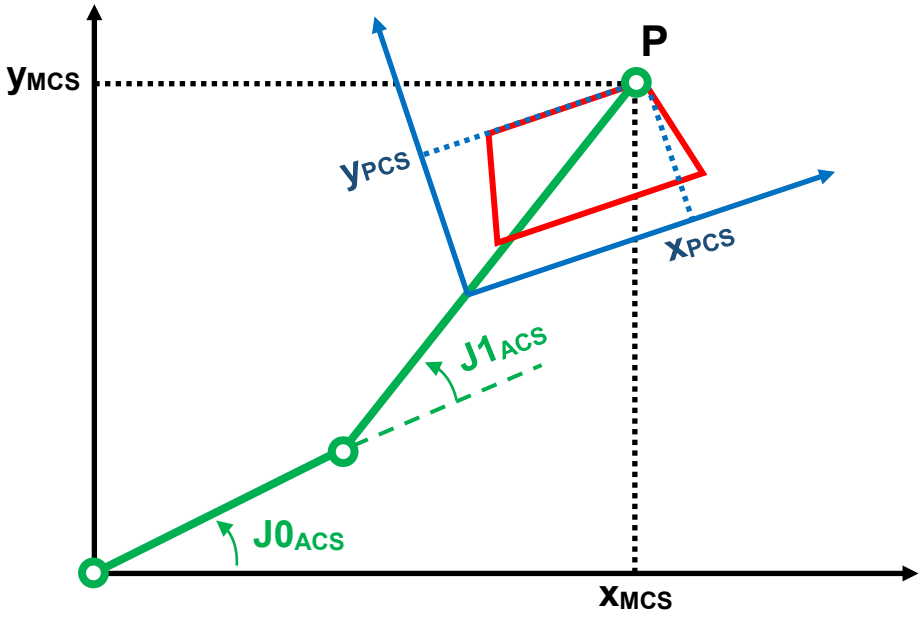


圖 19-1

19.4 運動學

ID	名稱	描述
0	Independent	單軸。
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	X-Y-C Gantry	(不使用)
3	Gantry master-master	(不使用)
4	SCARA	(不使用)
5	DELTA	(不使用)
6	6-axis articulated robot (PUMA)	(不使用)
7	Custom	(不使用)

表 19-4

19.5 數學常數

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。	0.70710678118654752401

表 19-5

19.6 系統變數

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。 **請參閱節 10.5 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。

表 19-6

19.7 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。)