創博股份有限公司

IoT 智動化解決方案

NexMotion 函式庫

使用者手冊

版本：1.6

日期：2018-11-15

版權與免責聲明

本檔內的所有資料屬創博股份有限公司(以下簡稱本公司)專屬財產,均受智慧財產權相關法規（包括但不限於著作權法 ）所保障。任何未經本公司授權的使用均屬侵權行為。若未經本公司事先書面同意，本檔資料之全部或部份均不可被複印、銷售、散佈、修改、發表、儲存或以其他方式作不當利用。

為力求檔之正確性及完整性，本公司保留在任何時間、不另行通知之情況下，變更或修改本檔之權利。

運行中的機械或設備具有一定的危險性，使用者在做任何操作前，應特別注意並應做好安全防護措施，本公司不承擔因不當使用本檔所述設備所造成的直接或間接損失。

檔版本紀錄

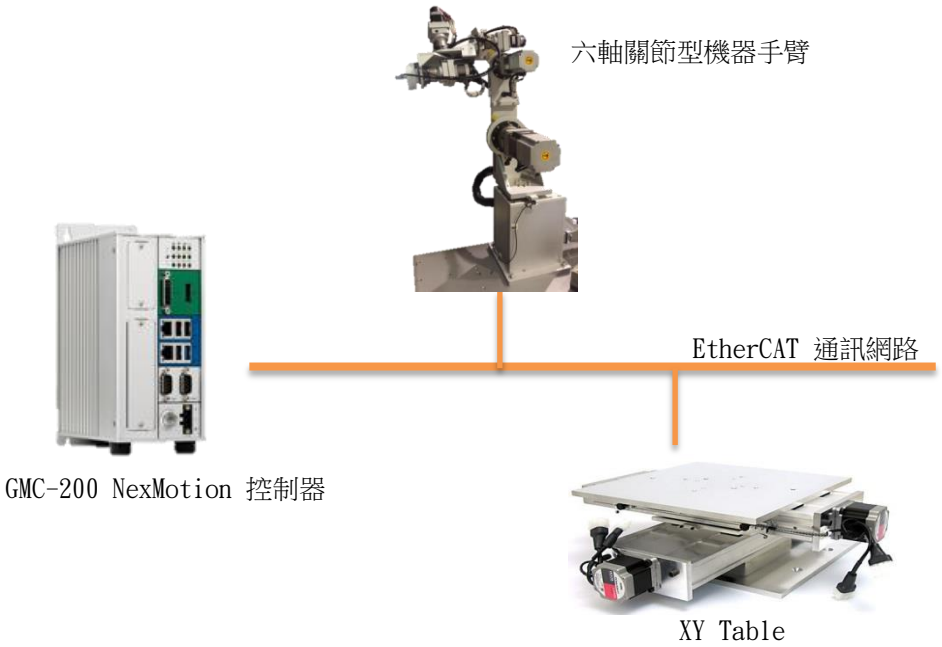
版本	修改紀錄
1.0	First released.
1.1	Add Programming theory
1.2	1. Insert a chapter 3.2.8 get description APIs. 2. Add an axis parameter. Chapter 2.2
1.3	Modify description of NMC_GroupGetMotionBuffSpace()
1.4	1.Add Ch1.4.2.4~7 for tool/base setting and teaching 2.Add Ch2.3 Tool/Base parameters for group 3.Add Ch3.4.13~14 tool/base teaching APIs
1.5	1.Add Ch3.4.15 3D simulation view APIs
1.6	1.Add Ch3.2.6 new I/O functions 2.Enhance I/O function descriptions 3.Correct the descriptions of group parameters

目錄

1. 程式設計原理

NexMotion 應用程式的基本開發流程如下:

1. 控制系統規劃
2. 利用 NexMotion Studio 對受控系統進行設定與測試
3. 利用 NexMotion Studio 產生受控系統組態檔(NCF 檔案)
4. 控制程式開發
5. 編譯與除錯
6. 測試與微調控制程式



控制系統範例（控制器 + 受控系統）

前 3 個步驟主要目的為確認:

1. 受控系統的安裝與配線是否妥善? 比如伺服馬達是否已正確安裝? I/O 裝置是可正確運作?等

2. 軸控參數是否正確，是否可以透過 NexMotion studio 進行設備操作，比如說受控系統為一隻六軸關節式機器手臂(Articulated robot)和一組兩軸之 XY 平臺系統(XY Table)，可以透過 NexMotion studio 進行機構參數之設定,單位設定以及加減速參數等設定，並使用 NexMotion studio 進行歸原點操作，點對點運動操作等確認運行的方向，移動的距離單位等是否符合系統規劃。NexMotion Studio 的使用方法可參考”NexMotion studio 使用者手冊”。

待前述步驟完成後，會自動在系統預設路徑中(C:\Nexcom\)產生組態檔案，使用者應避免更動該路徑底下之檔案，避免不預期之系統錯誤產生。

第 4 步驟:控制程式開發，意即透過 NexMotion 所提供之函式庫進行控制程式之開發，透過呼叫 API 對控制器下達命令以完成各式各樣之應用。在接下來的章節中將分別函式庫的功能作一系統性的說明包含

1. 系統操作
2. I/O 控制
3. 單軸控制
4. 群組控制

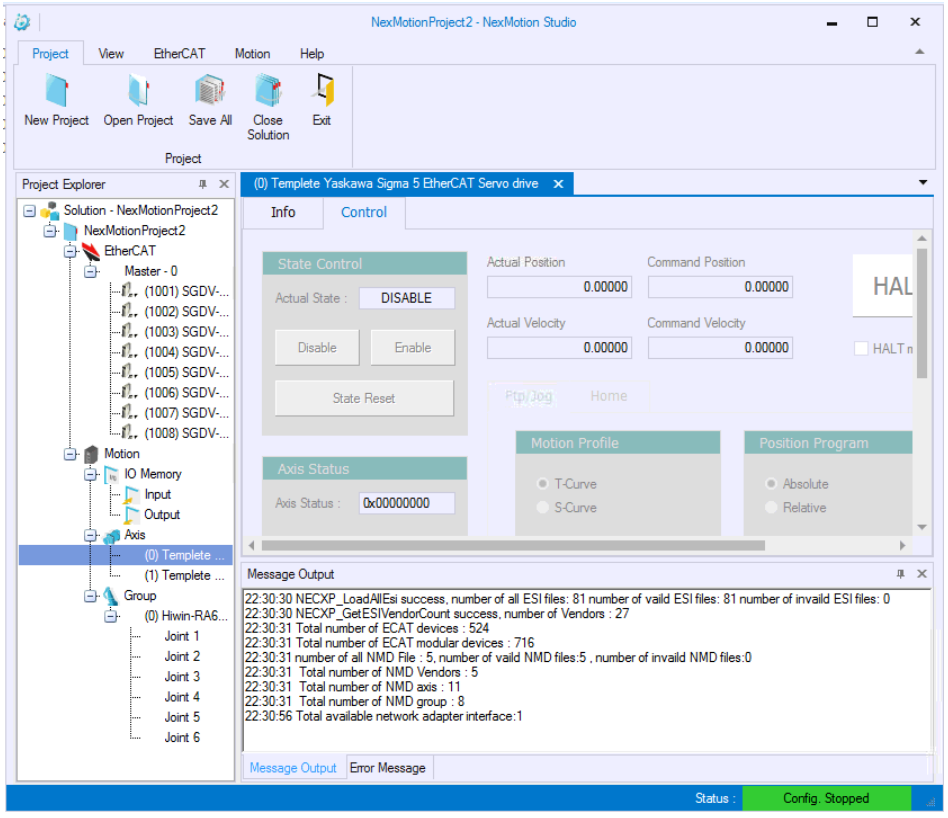
可以透過 API 命名來瞭解其分類，比如說

以 NMC_Device…開頭的 API 為「系統操作」類的 API，

以 NMC_Axis…開頭的 API 為「單軸控制」相關 API，

以 NMC_Group…開頭的 API 為「群組控制」相關 API，以此類推

亦可搭配參考產品安裝目錄下所提供之範例程式作為應用開發之基礎進行延伸變化。



NexMotion Studio 開發設定工具畫面

1.1. 系統操作

系統相關操作章節描述包括:

1. 系統初始化
2. 看門狗
3. 除錯功能

1.1.1. 系統初始化

開始使用函式庫之前必須先進行初始化的流程，最簡單的方式就是呼叫 NMC_DeviceOpenUp() 進行啟動，若成功啟動會回傳控制器的識別字(ID)，爾後可使用此識別字來對控制器下達命令。

下面為一個系統啟動和系統關閉的 C 程式範例:

```
#include "NexMotion.h"
#include "NexMotionError.h"

int main()
{
    RTN_ERR ret;
    I32_T   devType  = NMC_DEVICE_TYPE_ETHERCAT;
    I32_T   devIndex = 0;
    I32_T   devID;

    ret = NMC_DeviceOpenUp( devType, devIndex, &devID );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    // Controller is start up successfully.
    // Do something...
    // 可撰寫控制相關程式…

    NMC_DeviceShutdown( devID );

    return 0;
}
```

NMC_DeviceOpenUp() 是一個阻塞式(Blocked) 函式，該函式會進行所有系統初始化的工作包括

1. 建立控制器，產生控制器識別字
2. 載入控制器組態檔（NCF 檔案）
3. 啟動控制器通訊

上述這些工作將會花費數秒鐘，若函式成功返回代表控制器完成啟動，接下來的程式便可開始對單軸或群組等下達運動控制命令。另外，若不希望使用阻塞式函式，本函式庫也另提供非阻塞式

(Non-blocked)的呼叫方式 NMC_DeviceOpenUpRequest()，此函式呼叫後立即返回，控制器收到命令後同樣開始進行上述啟動工作，待過一段時間後控制器完成啟動工作其裝置狀態(Device state)會切換至「OPERATION」，可使用 NMC_DeviceGetState()進行輪詢(Polling)狀態或利用 NMC_DeviceWaitOpenUpRequest() 等待啟動完成。

1.1.2. 關閉控制器

關閉控制器只需要呼叫 NMC_DeviceShutdown()，當使用者呼叫此函式後，控制器會立即關閉通訊，因此使用者必須注意應用程式的關閉程式，例如，確認相關設備之運動狀態皆已停妥後再呼叫 NMC_DeviceShutdown()將控制器徹底關閉。

1.1.3. 進階系統初始化

一般初始化，可採用前述初始化流程較為簡便快速，但某些應用情況下希望啟動前先進行部分參數設定，可採用下述分段的方式進行控制器初始化:

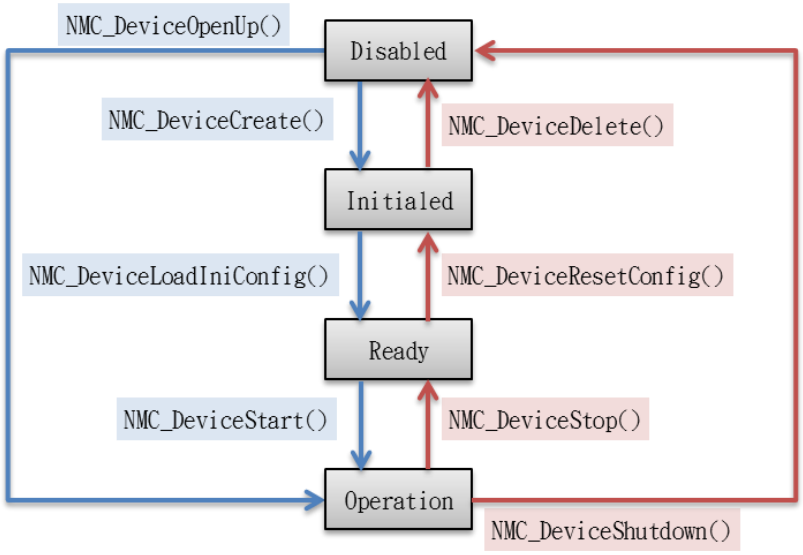
```
#include "NexMotion.h"
#include "NexMotionError.h"


int main()
{
    RTN_ERR ret;
    I32_T   devType  = NMC_DEVICE_TYPE_ETHERCAT;
    I32_T   devIndex = 0;
    I32_T   devID;

    ret = NMC_DeviceCreate( devType, devIndex, &devID );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    ret = NMC_DeviceLoadIniConfig( devID );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    // Do someting here...
    // Parameter setting...

    ret = NMC_DeviceStart( devID );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    // Controller is start up successfully.
    // Do something...
```

```
    // ...

    NMC_DeviceShutdown( devID );

    return 0;
}
```

若採用此種分段的啟動方式，下列 3 個 API 必須依序被呼叫使用

1. NMC_DeviceCreate()
2. NMC_DeviceLoadIniConfig()
3. NMC_DeviceStart()

目的是將控制器之控制器狀態(Device state)會切換至「OPERATION」，下圖為相關函式的呼叫與控制器狀態變化的關係圖:



控制器狀態變化

上述的控制器狀態可使用 NMC_DeviceGetState()讀取之。

1.1.4. 看門狗計時器(Watch Dog Timer)

看門狗計時器(Watch dog timer)為控制器的一個專用計時器，當計時器被起啟動後且在規定的時間內應用程式必須清除計時器的計數值,否則當計時器到達設定的時間後控制器會自動進行相對應的關閉動作。

此功能主要是為防範應用程式當機或者系統其他程式造成系統當機情況下的一種保護措施，當系統當機時因控制器的計數器沒有辦法再被應用程式清除時而發生過時(Timeout)而啟動關閉程式。

在開發除錯階段若開啟此功能需特別注意，常發生除錯過程因中斷程式(中斷點)，導致計時器過時系統關閉而發生不預期狀況，因此建議開發階段可以不啟用此功能，待開發完畢後再加上此功能做額外強化的安全保護。

看門狗計時器(Watch dog timer)之使用的方式一般為啟動一個系統 timer 中斷或者建立一個獨立的執行序(Thread)，透過 NMC_DeviceWatchdogTimerEnable()啟動看門狗計時器，並使用 NMC_DeviceWatchdogTimerReset()來定時清除計時器,清除的頻率通常比設定的時間要快上一倍甚至更多。若要停止看門狗計時器使用 NMC_DeviceWatchdogTimerDisable()

下面為一個例子:

```
#include "NexMotion.h"
#include "NexMotionError.h"
#include <Windows.h>

I32_T gThreadCtrl = 1;

DWORD WINAPI WatchDogTimerResetThread(_In_ LPVOID lpParameter )
{
    I32_T   devId      = *(I32_T *)lpParameter;
    I32_T   timeoutMs  = 1000;
    DWORD   sleepMs    = timeoutMs / 2;

    NMC_DeviceWatchdogTimerEnable( devId, timeoutMs, 0 );

    while( gThreadCtrl == 1 )
    {
        NMC_DeviceWatchdogTimerReset( devId );
        Sleep( sleepMs );
    }

    NMC_DeviceWatchdogTimerDisable( devId );

    return 0;
}

int main()
{
    RTN_ERR ret;
    I32_T   devType  = NMC_DEVICE_TYPE_ETHERCAT;
    I32_T   devIndex = 0;
    I32_T   devID;
```

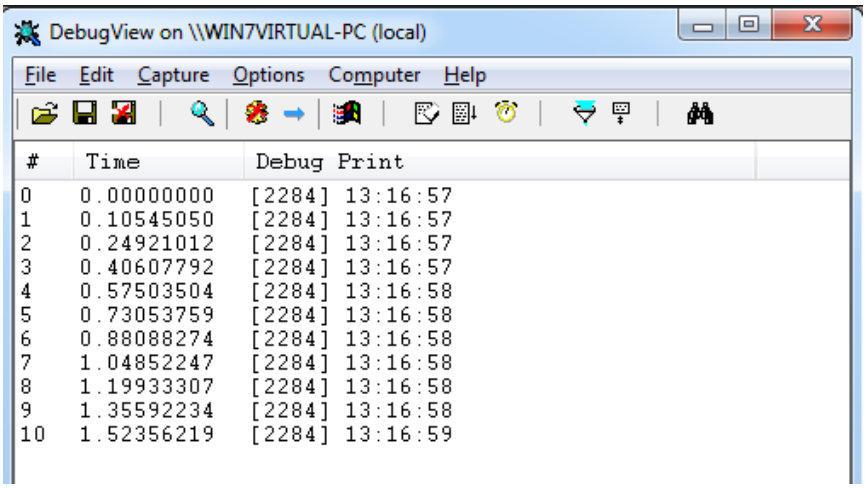
```
    HANDLE  threadHandle;

    ret = NMC_DeviceOpenUp( devType, devIndex, &devID );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    threadHandle = CreateThread( NULL, 0, WatchDogTimerResetThread, &devID, 0, NULL );
    if( threadHandle == NULL )
    {
        // Error handling...
    }

    // Controller is start up successfully.
    // Do something...
    // ...

    // Try to stop WDT thread
    gThreadCtrl = 0;
    WaitForSingleObject( threadHandle, INFINITE );

    NMC_DeviceShutdown( devID );

    return 0;
}
```

1.1.5. 除錯相關

協助用戶除錯相關功能有兩類，第一類是系統訊息，第二類是API trace。

1.1.5.1. 系統訊息

系統訊息是指控制器在運行過程中會發出訊息存入到控制器的訊息佇列(Message queue)，可透過NMC_MessagePopFirst() 將佇列中之訊息讀出，當訊息從佇列中讀出後訊息既從佇列中刪除。為方便除錯，可呼叫NMC_MessageOutputEnable()將系統訊息複製轉發一份至MS Windows系統訊息(OutputDebugString)之中，使用者可下載DebugView小工具來攔截Windows之系統訊息。DebugView 工具可至 https://docs.microsoft.com/zh-tw/sysinternals/downloads/debugview 下載



DebugView 畫面

1.1.5.2. 函式追蹤(API Trace)

函式追蹤(API Trace)主要是追蹤用戶之應用程序呼叫NexMotion 函式庫的情況。透過使用NMC_DebugSetTraceMode()可開啟API Trace的功能，根據設定的Trace模式當NexMotion 所提供之API被呼叫時可將呼叫的函式名稱跟輸入的參數值和錯誤回傳值輸出到MS Windows系統訊息(OutputDebugString)之中，使用者可利用DebugView視窗軟體攔截觀察NexMotion函式呼叫之情況。

另外，更進一步可使用嵌入函式(Hooked function或稱Callback function)，將自訂的嵌入函式嵌入到NexMotion 的函數之中。將使用者自訂義的函數可透過 NMC_DebugSetHookFunction() 註冊到系統之中，當NexMotion 函式庫中的API被呼叫執行後要返回前會自動呼叫使用者所嵌入的函式(Hooked function)。

註冊到系統中的嵌入函數是全域的，亦即一旦註冊成功，所有被呼叫的NexMotion API都會呼叫同一個嵌入函式，可透過NMC_DebugGetApiAddress()查詢呼叫的API名稱。

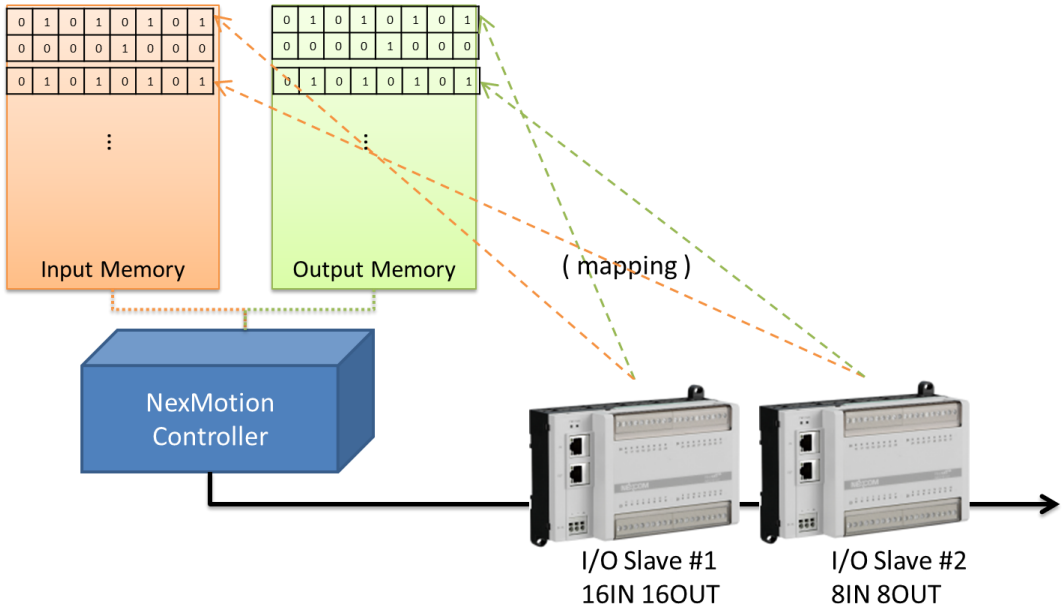
下面為一個使用範例:

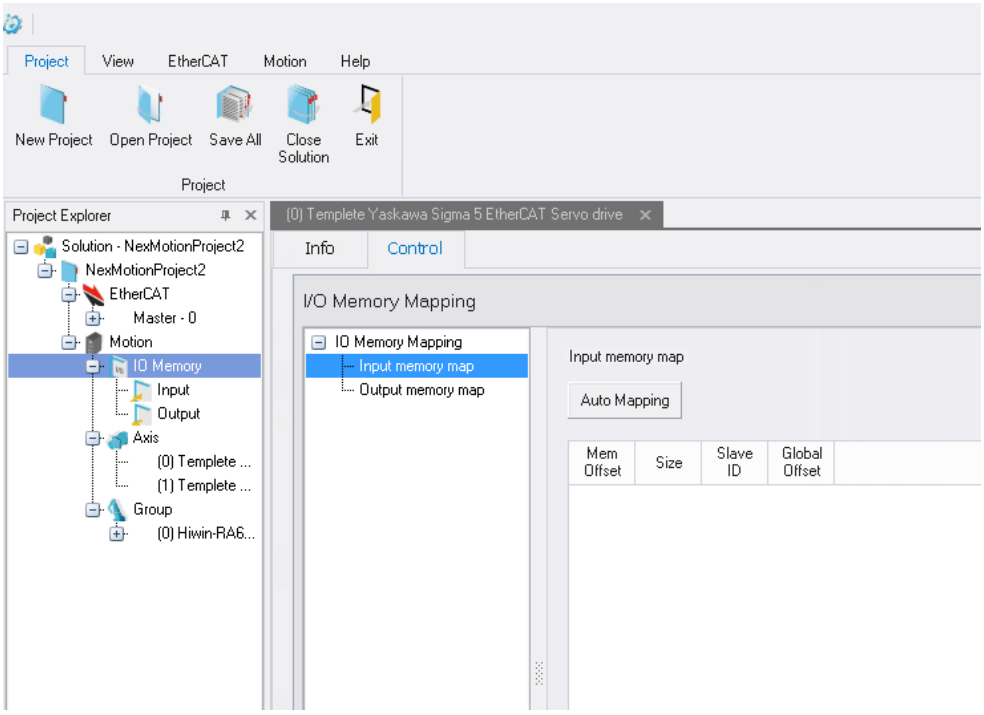
```
#include "NexMotion.h"
#include "NexMotionError.h"
#include <stdio.h>

void MyHookFunction(
      const void   *PFuncAddress  // [i] Function name call this hook funciton.
    , const char   *PFuncName     // [i] Pass API Name to hook function.
    , RTN_ERR       ReturnCode    // [i] function return call
    , void         *PUserData     // [i] User data.
)
{
    const void *pFunAddr = NMC_DebugGetApiAddress( "NMC_DeviceOpenUp" );
    I32_T *pCounter      = (I32_T *)PUserData;
```

```
    if( PFuncAddress == pFunAddr )
    {
        (*pCounter)++;
        printf( " NMC_DeviceOpenUp is called %d times \n", *pCounter );
    }

    printf( "Hook: %s is called.\n", PFuncName );
}


int main()
{
    RTN_ERR ret;
    I32_T   devType  = NMC_DEVICE_TYPE_ETHERCAT;
    I32_T   devIndex = 0;
    I32_T   devID;

    I32_T   counter  = 0;

    // Set Hook function
    NMC_DebugSetHookData( &counter );
    NMC_DebugSetHookFunction( MyHookFunction );

    ret = NMC_DeviceOpenUp( devType, devIndex, &devID );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    // Controller is start up successfully.
    // Do something...
    // ...

    // Disable hooked function
    NMC_DebugSetHookFunction( 0 );
    NMC_DebugSetHookData( 0 );

    NMC_DeviceShutdown( devID );

    return 0;
}
```

1.2. I/O 控制

NexMotion 提供彈性且高速的 I/O 控制，存取 I/O 如同存取記憶體一般的。使用上，在存取 I/O 前須先使用 NexMotion studio 來設定 I/O 裝置的記憶體映射位置，將 I/O 映射到虛擬記憶體後，存取該位置記憶體時既可控制或讀取 I/O，其設定方式請參考 NexMotion Studio 使用手冊。





NexMotion Studio I/O Memory 設定畫面

設定後，即可使用下列函式來存取虛擬記憶體(控制或讀取 I/O 裝置)

- NMC_ReadInputMemory()
- NMC_ReadOutputMemory()
- NMC_WriteOutputMemory()

須注意，I/O 記憶體的更新率規格是每 10 ms 更新一次(100 Hz)，因此若呼叫 Digital output 的控制頻率高於 100Hz 將無法及時回應。

1.3. 單軸(Axis)控制

有別於機構中協同作動的軸(group axis)，在本章節所介紹的單軸控制功能，其適用範圍對應到機械結構端乃為獨立作動(Independent)的軸(single axis)。就功能面來看，本章節的內容主要區分為三大部分，第一部分為單軸的設定，包含：單位設定、軟體極限保護設定；第二部分則為運動控制的功能，包含:激磁、點對點運動、JOG 運動、停止運動與 Change on fly 的功能;第三部分主要著重在讀取單軸運動相關資訊的功能,包含:讀取單軸目前的狀態、運動資訊。

1.3.1. 單軸單位設定

由於單軸運動控制 API 所需輸入的參數,譬如:點對點運動的位置,或是 JOG 運動的速度,都是運行在使用者定義的單位(user unit)。為了建立使用者定義的單位與實際設定到驅動器的命令脈波數(Command count)之間的關係，用戶必須先設定單位相關參數。下表列出與單位設定有關之單軸參數:

Param. Num.	Sub. Index	說明	備註
0x00	0	Mechanical pitch (unit/rev)	(*1)
0x01	0	Mechanical revolution	(*1)
0x02	0	Motor revolution	(*1)
0x03	0	Encoder resolution (pulse/rev)	(*1)

(*1) 啟動後無法修改參數

- 0x00: Mechanical pitch (user unit/rev)

  此參數主要描述機構端轉一圈等於多少個 user unit

- 0x01: Mechanical revolution

  0x02: Motor revolution

  上述兩個參數必須搭配設定，主要用來描述馬達與機構端之間的齒輪比關係。以旋轉機構來說，馬達與機構之間通常存在減速機，假設減速機的齒輪比等於 80，代表馬達端轉動 80 圈，機構端會相對應轉 1 圈，則 0x01 設定值為 1，0x02 設定值為 80。以直線機構使用導螺杆來說，若馬達端與機構端之間只存在聯軸器，中間沒有減速機構，則 0x01 與 0x02 的設定值皆為 1。但若馬達端與機構端之間存在減速機構，則必須依據減速機構的齒輪比設定 0x01 與 0x02。

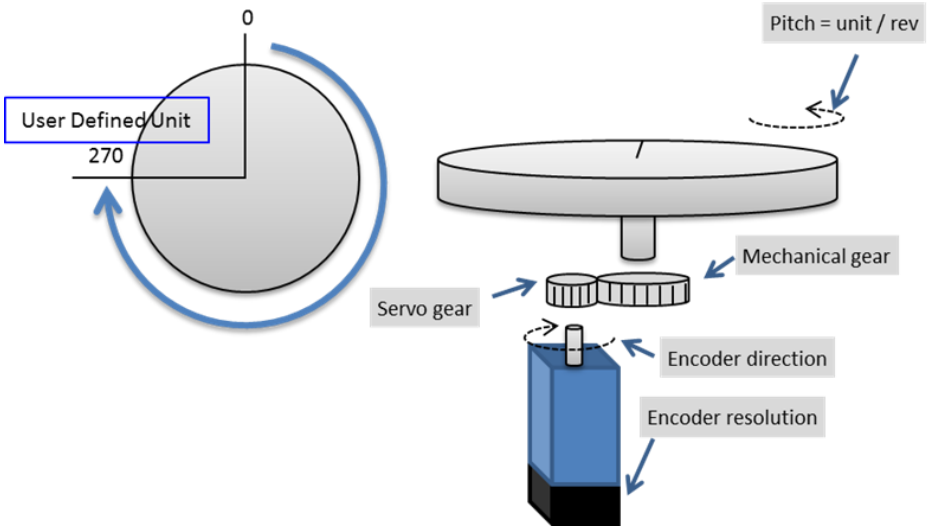
- 0x03: Encoder resolution (pulse/rev)

  此參數主要設定編碼器的解析度，代表馬達轉一圈，編碼器會回傳多少個脈波數(pulse count)。以 20 bit 解析度的編碼器來說，2 的 20 次方等於 1048576，則此參數的設定值為 1048576。

單位轉換公式如下:

$$1\ \text{User unit} = \ \text{EncoderResolution} \times \frac{Motor\ revolution}{Mechanical\ revolution} \div Pitch\ \ (\text{Pulse})$$

單位設定範例一: 圓盤機構



旋轉型電機 + 減速機 + 旋轉機構

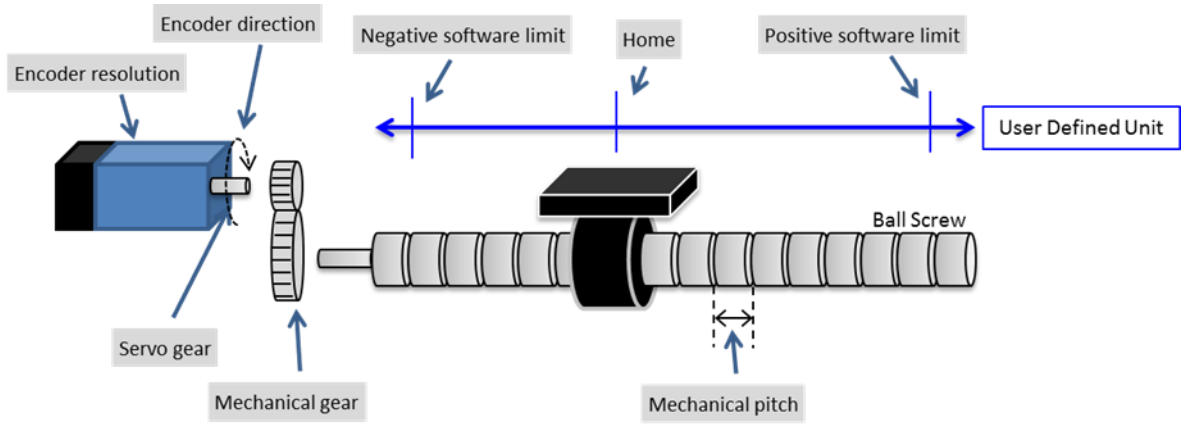
當減速機減速比 為 1:5 (馬達轉 5 圈時機構轉 1 圈)，Encoder 解析度為 1,048,576 pulse/rev
吾假設使用者單位(user unit)訂為每單位 0.001 degree ，參數設定如下:

0x00 (pitch)= 360,000
0x01 (Mechanical revolution) = 1
0x02 (Motor revolution) = 5,
0x03 (Encoder resolution)= 1048576

User unit(um) 和 pulse 之間的關係為:

1 user unit (0.001 degree) = 1048576 x (5/1) / 360000 = 14.5636 pulse

單位設定範例二: 螺杆機構



旋轉型電機 + 減速機 + 線性螺杆

當減速機減速比為 1:2（馬達轉 2 圈時機構轉 1 圈），Encoder 解析度為 131,072 pulse/rev，螺杆導程為 5 mm/rev，吾假設使用者單位(user unit)訂為每單位 1 mm ，參數設定如下:

0x00 (pitch)= 5
0x01 (Mechanical revolution) = 1
0x02 (Motor revolution) = 10
0x03 (Encoder resolution)= 131072

User unit(mm) 和 pulse 之間的關係為:
1 mm = 131072 x (10/1) / 5 = 262144 pulse

其他相關參數說明:

- 0x04: Encoder direction

此參數主要提供單軸反向的功能，default 值為 0。由於在單軸運動控制單元中，會依據編碼器回傳的脈波數計算單軸的位置座標(user unit)，若單軸運動方向對應到位置座標值的改變，與使用者預期的相反，可以將此參數設定為 1，則單軸運動方向對應到位置座標值的改變將會反向。此功能主要是讓用戶可以在控制器端直接可以進行運動反向的設定。

- 0x05: Encoder type

此參數主要設定編碼器的類型，若編碼器為增量型，則設定值為 0；若為絕對型，則設定值為 1。

- 0x06，SubIndex = 0: Enable external encoder
  0x06，SubIndex = 1: External Encoder ratio

此參數主要用來設定另一個編碼器，或是當輸入到驅動器的命令脈波數與編碼器回傳的脈波數不一致時，可以將 Enable 的參數(SubIndex 0)設定為 1，則單軸的位置座標值會利用編碼器回傳的讀值乘上 External encoder ratio 進行計算，此 ratio 代表的是編碼器讀值轉換為單軸位置座標必須乘以的比例。

- 0x07: Cancel synchronized actual position to command position when servo enable

此參數主要用來設定在激磁單軸時，設定到驅動器的命令脈波數是否解除與編碼器回傳的脈波數同步。一般來說，對於伺服馬達，在激磁單軸時，由於處在位置閉回路控制模式，因此，設定到驅動器的命令脈波數需要與編碼器的脈波數同步；但對於步進馬達來說，由於沒有位置閉回路控制，因此，在激磁時，設定到驅動器的命令脈波數必須解除與編碼器的脈波數同步，若強制兩者同步，甚至可能發生在激磁的當下，馬達隨即高速運轉，造成錯誤發生。

除了單軸(Axis)外，對於群組(Group)來說，各群組軸在 ACS 坐標系下，由於座標值對應的單位為 user unit，因此，在系統讀取控制器組態檔（NCF 檔案)之前，也必須利用 NexMotion Studio 設定與單位相關的參數，設定方式與單軸設定相同。

上述這些參數因為與機構元件的選用與配置有關，因此，在呼叫 NMC_DeviceOpenUp()之前，必須先利用 NexMotion Studio 設定相關的參數，以產生相對應的控制器組態檔（NCF 檔案)。若使用者已經成功呼叫 NMC_DeviceOpenUp()，若控制器狀態處於「OPERATION」，則無法再更改與單位設定相關的參數。

若使用者使用進階系統初始化的方式初始化控制器，因為在 NMC_DeviceLoadIniConfig()的階段，系統會讀取控制器組態檔（NCF 檔案)，所以，一旦控制器的狀態處於「READY」，則無法再更改與單位設定相關的參數。

1.3.2. 軟體極限保護

在實際的機構中，單軸往往有行程上的限制，且因為馬達有最大可輸出的力矩與速度限制，因此，在單軸運動的參數中，有相對應的正負方向之位置極限、最大可允許速度與最大可允許加速度。在運動控制單元中，有提供相對應的軟體保護機制，確保單軸在運動的過程中，機構在允許的限制條件下作動。對於單軸運動來說，當啟動軟體極限保護功能，在呼叫單軸運動的函式時，會檢查相關的運動參數，若不滿足設定的極限條件，則函式會回傳相對應的錯誤碼。對於群組的單軸來說，當群組正在進行軌跡運動，若各別的單軸有超出設定的極限保護值，控制器會進行停止運動。以下，分別說明與極限保護相關的參數，

1.3.2.1. 位置極限保護

Param. Num.	Sub. Index	說明	備註

0x10	0	Positive software limit (user unit)	(*2)
	1	Enable positive software limit	
	2	Negative software limit (user unit)	(*2)
	3	Enable negative software limit	

(*2): Enable 該功能時參數生效, 其他時間修改不會立刻變更

上述的參數主要與軟體位置極限有關，若要啟動單軸軟體位置保護功能，必須先設定極限位置(SubIndex 0 或 2)，設定完位置後，再設定 SubIndex 1 或 3 為 1(Enable)，以啟動軟體保護功能。啟動後，若欲更改極限位置，必須先關閉此功能(將 SubIndex 1 或 3 設定為 0)，待設定好新的極限位置後(SubIndex 0 或 2)，再啟用(將 SubIndex 1 或 3 設定為 1)方可運作。

1.3.2.2. 速度極限保護

Param. Num.	Sub. Index	說明	備註
0x11	0	Maximum velocity limit (unit/sec)	(*2)
	1	Enable max. velocity limit	

(*2): Enable 該功能時參數生效, 其他時間修改不會立刻變更

上述參數主要與單軸的速度限制保護有關，若要啟動單軸最大速度保護功能，必須先設定極限速度(SubIndex 0)，設定完速度後，再設定 SubIndex 1 為 1(Enable)，以啟動單軸最大速度限制保護功能。啟動後，若欲更改極限速度，必須先關閉此功能(將 SubIndex 1 設定為 0)，待設定好新的極限速度後(SubIndex 0)，再啟用(將 SubIndex 1 設定為 1)方可運作。

1.3.2.3. 加速度限制保護

Param. Num.	Sub. Index	說明	備註
0x12	0	Maximum acceleration limit (unit/sec2)	(*2)
	1	Enable max. acceleration limit	

(*2): Enable 該功能時參數生效, 其他時間修改不會立刻變更

上述參數主要與單軸的最大加速度限制保護有關，若要啟動單軸最大加速度保護功能，必須先設定極限加速度(SubIndex 0)，設定完加速度後，再設定 SubIndex 1 為 1(Enable)，以啟動單軸最大加速度限制保護功能。啟動後，若欲更改極限加速度，必須先關閉此功能(將 SubIndex 1 設定為 0)，待設定好新的極限加速度後(SubIndex 0)，再啟用(將 SubIndex 1 設定為 1)方可運作。請注意，單軸在運動過程中，啟動緊急停止運動 NMC_AxisStop()所使用的減加速度並不受限於此保護功能所制定的最大加速度(SubIndex 1)。

1.3.3. 單軸啟動與狀態

在啟動單軸進行運動之前，必須先呼叫 NMC_AxisEnable()，讓馬達進入激磁狀態。由於此函式為非阻塞式呼叫，當呼叫此函式並成功回傳，並不代表馬達已經進入激磁狀態，使用者可以呼叫 NMC_AxisGetState() 取得單軸目前的狀態，當單軸狀態切換到「NMC_AXIS_STATE_STAND_STILL」，代表馬達已經成功進入激磁狀態。

除了透過單軸狀態(State of axis)確認馬達是否已經進入激磁狀態，使用者也可以呼叫 NMC_AxisGetStatus()讀取單軸運動資訊(Status of axis)，當 ENABLE (ENA,bit 6)轉變為 1，代表馬達已經成功進入激磁狀態。

在呼叫 NMC_AxisEnable()之前，可以先呼叫 NMC_AxisGetState()，確認馬達驅動器是否有報警(Alarm)，當馬達驅動器有報警，單軸狀態會切換到「NMC_AXIS_STATE_ERROR」。使用者也可以呼叫 NMC_AxisGetStatus()，當單軸運動資訊(status of axis)的 ALM(bit 1)與 ERR(bit 7)轉變為 1，代表馬達驅動器有報警的情況，此時，必須呼叫 NMC_AxisResetDriveAlm()，以清除驅動器的報警。請注意，並不是所有的驅動器報警都可以透過此函式清除，有些驅動器報警必須藉由斷電再上電的方式重置。當清除驅動器報警後，必須再呼叫 NMC_AxisResetState()將單軸的狀態從「NMC_AXIS_STATE_ERROR」切換到「NMC_AXIS_STATE_DISABLE」，此時，單軸運動資訊的 ALM(bit 1)與 ERR(bit 7) 會清除為 0。在呼叫 NMC_AxisResetState()時，若驅動器有報警，此函式會先清除驅動器報警後再將單軸狀態回復到「NMC_AXIS_STATE_DISABLE」。

當單軸成功進入激磁狀態後，即可以呼叫單軸運動相關函式進行運動。當運動結束，或是在運動中發生突發狀況，使用者可以直接呼叫 NMC_AxisDisable()，將馬達解除激磁狀態。當成功解除激磁狀態，單軸狀態會切換到「NMC_AXIS_STATE_DISABLE」，單軸運動資訊的 ENA(bit 6)會清除為 0。

1.3.4. 單軸速度百分比設定

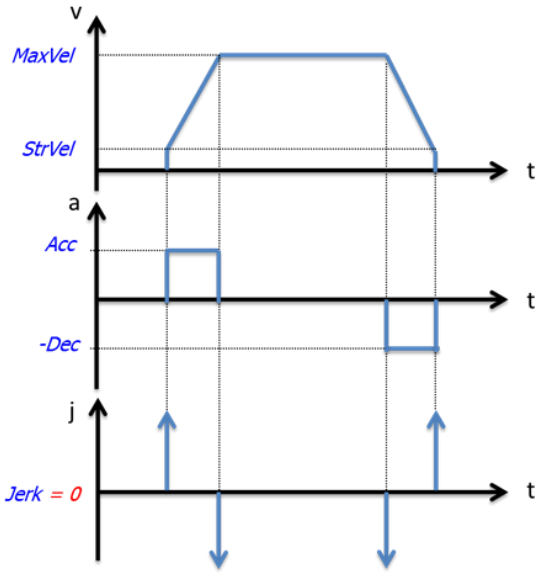
在單軸的運動中，包括點對點運動與 JOG 運動，都會依據單軸參數:0x32 的設定值作為目標速度。在某些應用情境中，通常是在手動操作模式下，為了確認點位元或運作程式是否滿足需求，會希望能夠降低運動的目標速度，此時，使用者可以呼叫 NMC_AxisSetVelRatio()設定速度百分比，成功設定後，運動控制模組將會依據單軸參數:0x32 (Max. velocity)乘上設定的速度百分比做為單軸運動的目標速度。此外，呼叫 NMC_AxisGetVelRatio()可以取得目前單軸運動的速度百分比。

此函式除了可以在單軸尚未進行任何運動前呼叫，也支持線上呼叫，也就是當單軸正在進行點對點運動或是 JOG 運動時，呼叫此函式除了會依據設定的百分比自動改變目前運動的目標速度，對於接下來輸入的單軸點對點運動與 JOG 運動命令，其目標速度也都會依據此設定的百分比進行規劃。

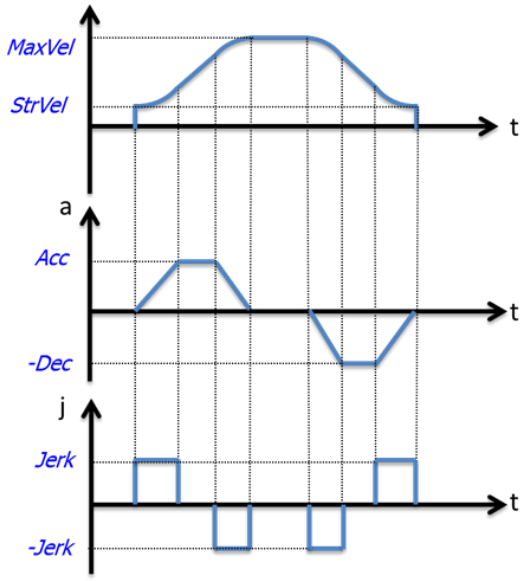
有關速度百分比設定的數值，範圍從 0(包含) ~ 1(包含)，設定超出此範圍的數值，在呼叫 NMC_AxisSetVelRatio()時將會回傳錯誤碼。若單軸在進行點對點或 JOG 運動時，將速度百分比設為 0，則單軸將降速到 0，且單軸運動資訊 (Status of axis)的 CSTP(bit 9)會轉變為 1。雖然在此狀況下，單軸處於靜止狀態，但原先執行的點對點運動或 JOG 運動仍然在運作，因此單軸運動資訊的 OP(bit 13)會保持為 1，當速度百分比再被設定為大於 0 的值，單軸將會自動運動到目標位置(點對點運動)或目標速度(JOG 運動)。

1.3.5. 單軸點對點(PTP)運動

在單軸運動中，點對點(PTP)運動是一個很常使用到的運動形式，使用者可以呼叫 NMC_AxisPtp()，給定目標位置或相對位移，運動控制模組將依據相關的單軸參數進行速度曲線規劃，將單軸移動到用戶指定的位置，如下圖所示。



梯形速度曲線



S 形速度曲線

點對點運動相關之相關單軸參數:

Param. Num.	Sub. Index	說明	備註
0x28	0	Base velocity (unit/sec)	
0x31	0	Profile type	
0x32	0	Max. velocity (unit/sec)	
0x33	0	Acceleration (unit/sec2)	
0x34	0	Deceleration (unit/sec2)	
0x35	0	Jerk (unit/sec3)	

以下分別說明相關參數的意義與使用方式：

- 0x28： Base velocity (unit/sec)

當單軸在靜止狀態下啟動點對點運動，一般來說，速度曲線會從初速為 0 開始加速到目標速度，但在步進馬達的應用中，常需要設定初始的起跳速度，在此需求下，使用者可以將參數 0x28 設定為起跳速度，則在啟動點對點運動與 JOG 運動時，初始速度會依據參數 0x28 的設定值進行規劃。除了初始速度，當單軸在進行點對點運動或是停止運動時，末速度也會設定為參數 0x28 的設定值。

- 0x30： Absolute or relative programming

在點對點運動 NMC_AxisPtp()中，使用者輸入的參數 TargetPos，可以是絕對位置座標，也可以是相對位移。當 0x30 參數設定為 1，則點對點運動輸入的參數 TargetPos 為相對位移；當 0x30 參數設定為 0，則 TargetPos 為絕對位置座標。

- 0x31： Profile type

當使用者呼叫 NMC_AxisPtp()，控制器的運控控制模組會先規劃一個速度曲線，並依據此速度曲線在每個通訊週期時間計算出單軸的命令位置，最後將單軸移動到指定的目標位置。通常速度曲線有兩種形式，包含梯形速度曲線與 S 形速度曲線。當 0x31 設定為 0，速度曲線為梯形曲線，加速度不連續；當 0x31 設定為 1，速度為 S 形曲線，加速度連續。在相同的目標速度與加速度的情況下，採用 S 形速度曲線需要較多的時間到達目標位置或是目標速度，但因為加速度為連續，較可避免振動的發生。若採用梯形速度曲線，到達目標位置或目標速度所需要的時間較少，但因為加速度不連續，容易引起振動的產生。

- 0x32： Max. velocity (unit/sec)

此參數設定值為單軸點對點運動與 JOG 運動所採用的目標速度，如上圖的 MaxVel。

- 0x33： Acceleration (unit/sec2)

此參數設定值為單軸點對點運動、JOG 運動從初始速度到目標速度所採用的加速度，如上圖的 Acc。。

- 0x34： Deceleration (unit/sec2)

此參數設定值為單軸點對點運動從目標速度到末速度或是 Halt 運動所採用的減加速度，如上圖的 Dec。

- 0x35： Jerk (unit/sec3)

此參數設定值為 S 形速度曲線的 Jerk，乃加速度曲線圖(a-t)的斜率。

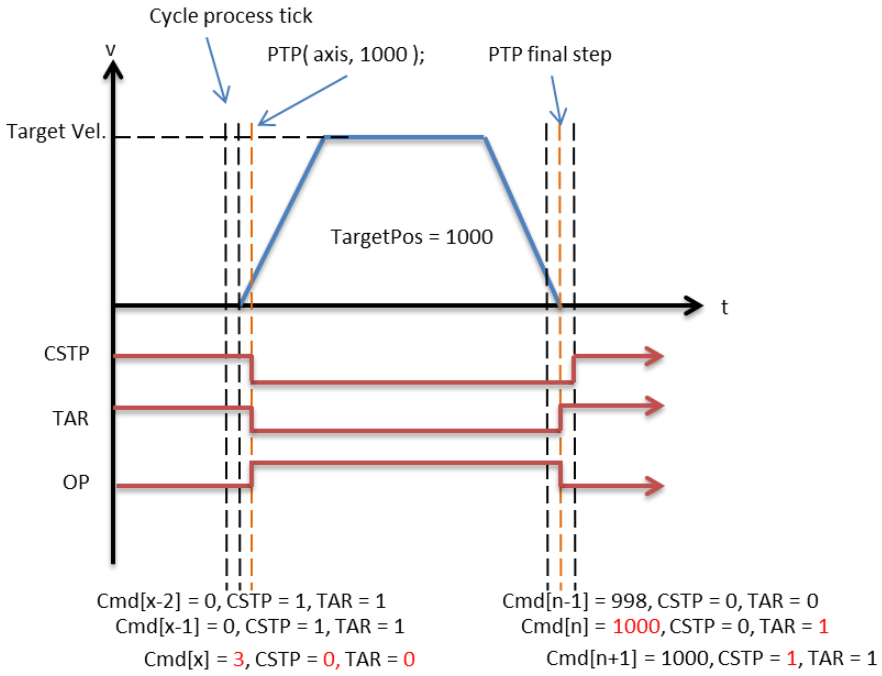
- 0x36： Buffer mode

單軸除了可以進行單節運動，也可以輸入一連串的運動命令儲存到單軸的運動隊列(Motion Queue)中，透過參數 0x36 的設定，定義新輸入的運動命令與前一個運動命令之間的連接方式。有關單軸運動隊列的運作行為，以及參數 0x36 的說明，將在單軸運動隊列章節進行說明，以下，先針對單一個點對點運動的相關內容進行說明。

當成功呼叫NMC_AxisPtp()後，單軸軸狀態會切換到「NMC_AXIS_STATE_DISCRETE_MOTION」。有關呼叫 NMC_AxisPtp()之前與呼叫後的單軸狀態改變之行為，請參考下列表格：

運動命令 / 目前狀態	NMC_AxisPtp()
NMC_AXIS_STATE_DISABLE	禁止，回傳錯誤
NMC_AXIS_STATE_STAND_STILL	狀態改變至 AXIS_STATE_DISCRETE_MOTION
NMC_AXIS_STATE_HOMING	禁止，回傳錯誤，正在進行的 homing 不會被影響
NMC_AXIS_STATE_DISCRETE_MOTION	依照單軸參數:0x36(Buffer mode)設定值儲存到運動隊列或立即執行
NMC_AXIS_STATE_CONTINUOUS_MOTION	依照單軸參數:0x36(Buffer mode)設定值儲存到運動隊列或立即執行 當 Buffer mode 為 Blending，禁止，回傳錯誤
NMC_AXIS_STATE_STOPPING	禁止，回傳錯誤
NMC_AXIS_STATE_STOPPED	禁止，回傳錯誤
NMC_AXIS_STATE_WAIT_SYNC	依照單軸參數:0x36(Buffer mode)決定 PTP 命令被儲存到運動隊列 1. Abort: 清除運動隊列後存入 2. Buffered: 存入運動隊列 3. Blending: 存入運動隊列 若 Buffer mode 為 Blending，而儲存在運動隊列的上一個運動命令為 JOG，則回傳錯誤。
NMC_AXIS_STATE_ERROR	禁止，回傳錯誤

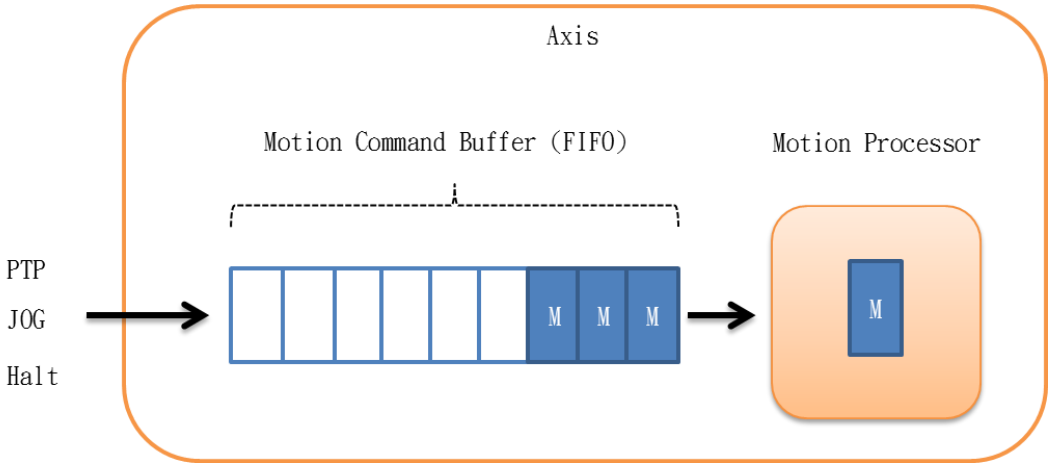
成功呼叫 NMC_AxisPtp()且開始進行 PTP 運動後，過程中的運動狀態可以透過呼叫 NMC_AxisGetStatus()取得單軸運動資訊，其中 bit 8 ~ bit 13 與運動狀態相關，以下圖表示：



PTP 運動過程中的運動狀態

1.3.6. 單軸運動隊列

每個單軸皆具有獨立之運動命令佇列(Motion command buffer)，因此在進行 PTP 或 JOG 運動過程中，若再輸入新的運動命令，則依據單軸參數:0x36 (Buffer mode)的設定值，會有不同的行為模式:



單軸運動隊列

- 每個單軸各有獨立之運動隊列(Queue)，單軸的處理單元會至運動隊列中取出運動命令執行。
- Buffer size = 32
- NMC_AxisPtp()、NMC_AxisJog()和 NMC_AxisHalt()運動可被填入運動隊列之中

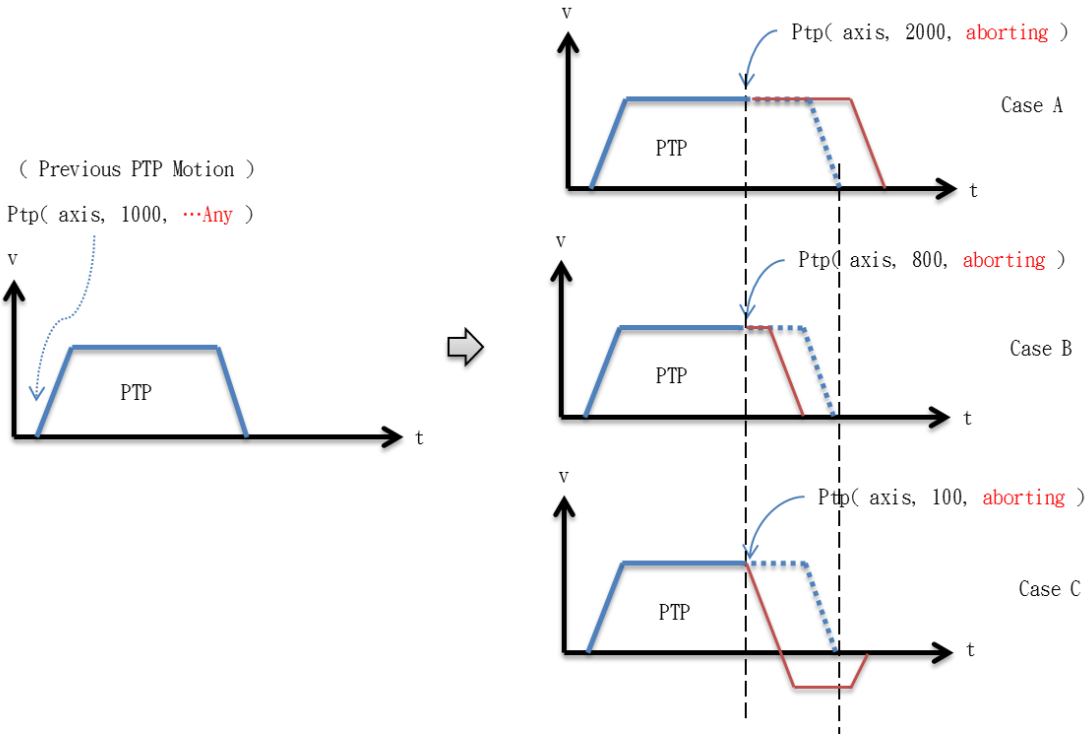
- 可透過 NMC_AxisGetMotionBuffSpace()查看目前佇列的容量
- NMC_AxisHalt()會停止目前的運動但不會清空佇列，如果佇列中有其他運動命令會接續執行
- NMC_AxisStop()會停止目前的運動並且清空佇列
- NMC_AxisDisable()會連帶清空佇列

以下說明單軸參數:0x36 (Buffer mode)的使用行為：

- 當單軸狀態在「NMC_AXIS_STATE_STAND_STILL」，呼叫單軸運動函式時，不管 0x36(Buffer mode)的參數設定值為何，會立即啟動。
- 當單軸正在進行運動的過程中，使用者呼叫新的運動命令，則新的運動命令以何種方式與目前的運動或前一次運動命令(儲存在單軸運動隊列中的運動命令)連結起來，乃依據單軸參數:0x36 的設定值，說明如下：

1.3.6.1. Buffer Mode: Aborting (0x36 = 0)

儲存在單軸運動隊列中的運動命令將會被清除，而原本正在進行的單軸運動將會被捨棄，由新的運動命令接管單軸的運動。以下面範例進行說明：

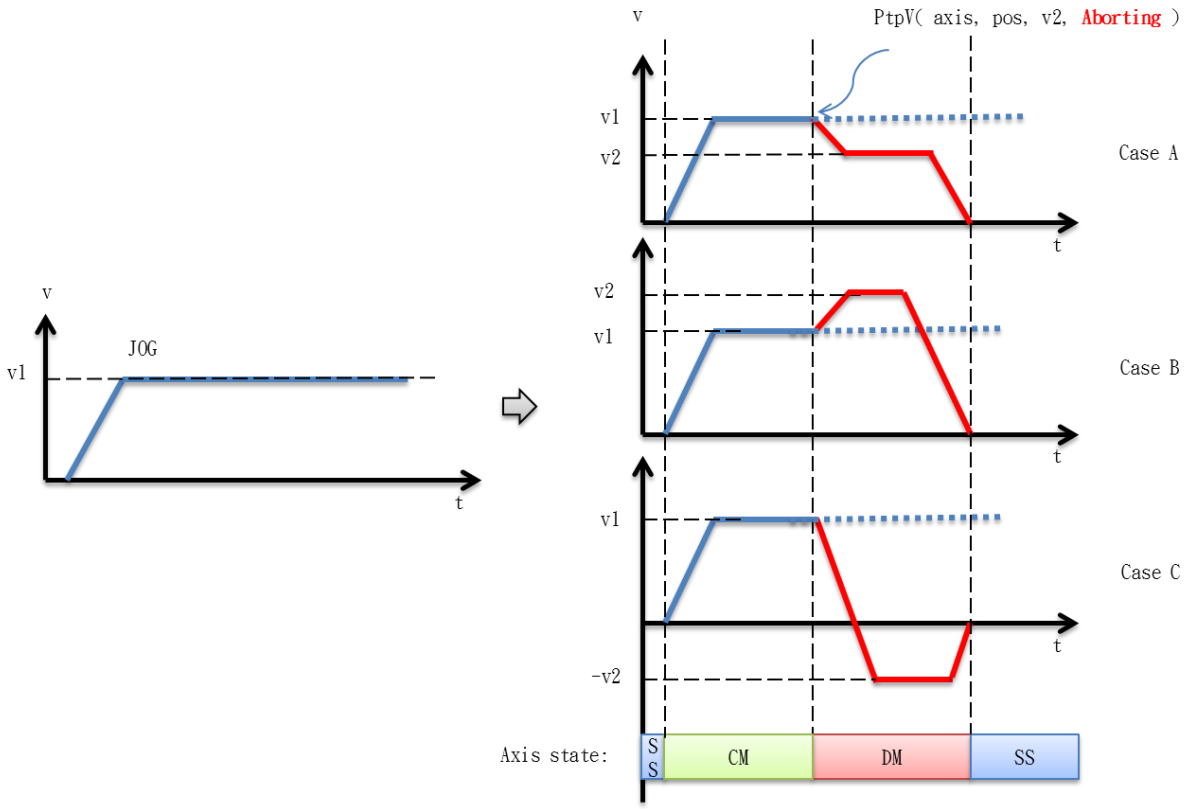


PTP 運動中由另一 PTP 運動中斷(Aborting)之三種情況

在上圖的範例中，原先單軸啟動了點對點運動，目標位置為 1000，當正在運動的過程中，又呼叫了新的點對點運動，且 0x36 設定值為 0 (Aborting)，則依據新的目標位置會有圖中所示

之三種可能情況，其中，當單軸往正方向運動，而新的目標位置小於目前的位置時，會產生反轉運動。

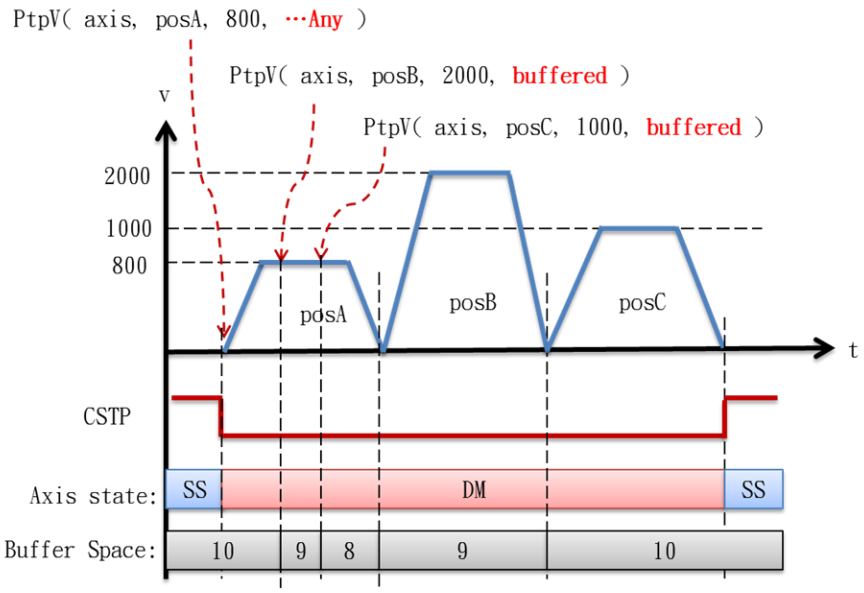
另一個範例如下圖所示，原先單軸正在進行 JOG 運動，目標速度設定為 v1，又呼叫了點對點運動，目標速度為 v2，且 0x36 設定值為 0（Aborting），當目標位置與目前位置的相對方向與目前運動方向相反時，會產生反轉運動，如圖中 Case C 所示。當目標位置與目前位置的相對方向與目前運動方向相同時，且相對距離足夠進行加減速，則目標速度會改為 v2，並運動到目標位置，如圖中 Case A 與 B 所示。



原為 JOG 運動，由另一 PTP 運動中斷(Aborting)之三種情況

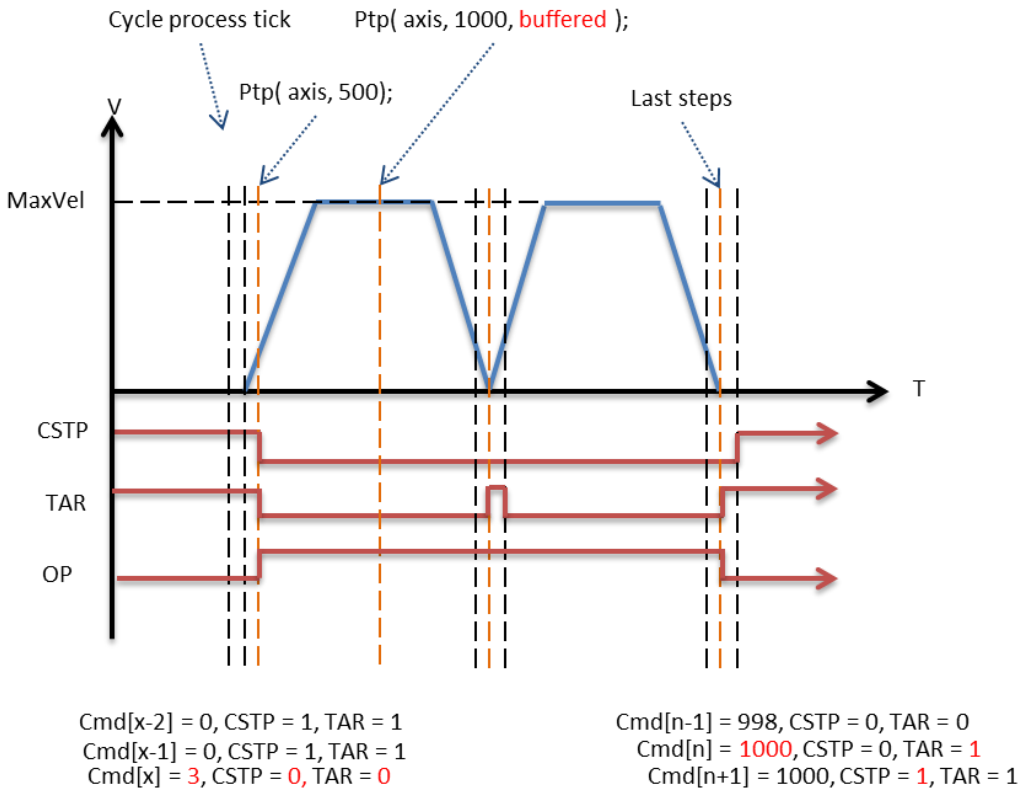
1.3.6.2. Buffer Mode: Buffered (0x36 = 1)

新輸入的運動命令將被儲存在運動隊列中，待前一個輸入的運動命令完成後才會被啟動。所謂的”運動命令完成”，若運動命令為點對點運動，代表到達目標位置；若運動命令為 JOG 運動，代表達到目標速度。



PTP + PTP(Buffered) + PTP(Buffered)

以上圖為例，在進行第一個 PTP 運動過程中，又連續下了兩個 PTP 運動且各自有不同的目標速度，由於 0x36 參數設定為 Buffered，這兩個 PTP 運動會被儲存到單軸運動隊列中，等到單軸到達目前運動的目標位置後，儲存在佇列中的運動命令將會被自動執行。確切的說，所謂到達目標位置，主要以單軸運動資訊(Status of axis)中，TAR(bit 8)為 1 作為判斷基準，如下圖所示。



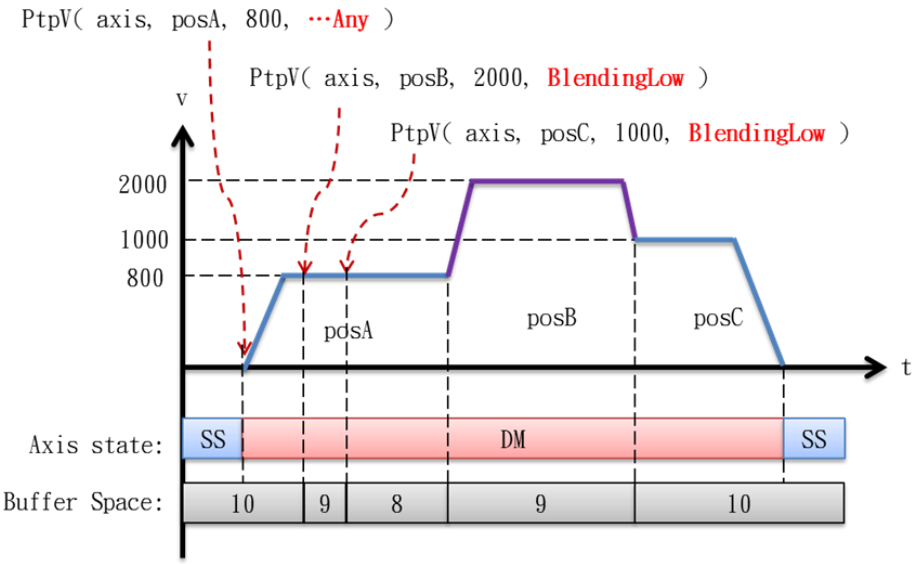
Buffered PTP 之運動狀態圖

除了前述的 Abort 與 Buffered，有另一種行為稱為 Blending，主要是新的運動命令與前一個運動命令會以指定的速度連接起來，有四種指定速度的方式，包含: BlendingLow、BlendingHigh、BlendingPrevious 與 BlendingNext，以下分別說明:

1.3.6.3. Buffer Mode: BlendingLow (0x36 = 2)

新的運動命令將被儲存在運動隊列中，並與前一個輸入的運動命令以較低的目標速度連結起來。以下圖來看，一開始從靜止狀態下啟動點對點運動，目標速度為 800，到達目標點位元 posA 時，末速度為 0。然而由於在運動過程中，新輸入另一個點對點運動，目標速度為 2000，且以 BlendingLow 的方式與前一個點對點運動連接，因此，原先的點對點運動末速度由 0 變為 800(2000 > 800)，而產生出如圖 1-3-9 中所示的速度曲線圖，從目標速度 800 一直運動到目標點位元 posA，末速度仍為 800。到達目標點位元 posA 後，運動控制模組會自動啟動第二個點對點運動，目標點位元為 posB，因此，速度曲線圖由初速度 800(等於上一個點對點運動的末速度)加速到目標速度 2000。若接下來沒有其他的運動儲存在運動隊列中，則移動到目標點位元 posB 時，末速度為 0。然而由於有另一個點對點運動(目標速度 1000)也以 BlendingLow 的方式與目前的點對點運動

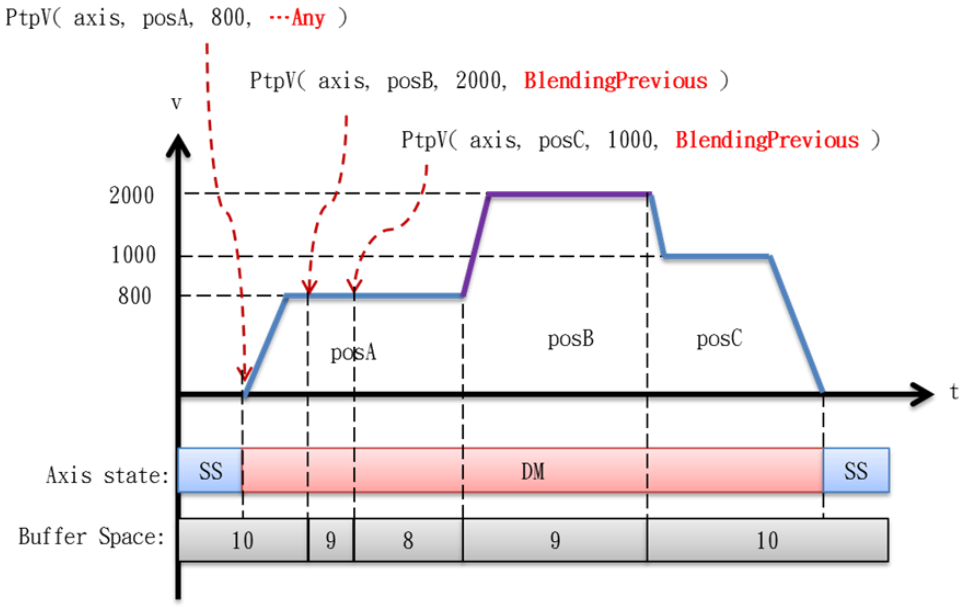
相連，因此，到達目標點位元 posB 時，末速度為 1000(2000 > 1000)，此時，運動控制模組會再啟動第三個點對點運動，將單軸移動到目標點位元 posC。



PTP + PTP(BlendingLow) + PTP(BlendingLow)

1.3.6.4. Buffer Mode: BlendingPrevious (0x36 = 3)

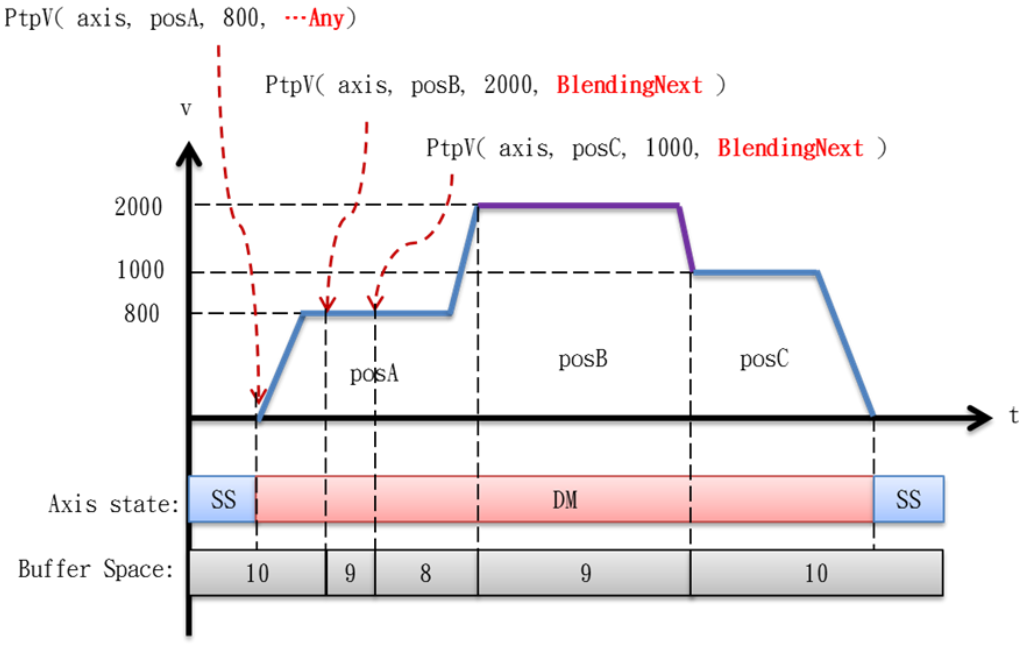
新的運動命令將被儲存在運動隊列中，並與前一個輸入的運動命令以前一個運動命令的目標速度連結起來。以下圖來看，新輸入的點對點運動與前一個運動的連接，皆以前一個運動的目標速度相連，因此，在移動到目標點位元 posA 與 posB 時，點對點運動的末速度皆等於目標速度。



PTP + PTP(BlendingPrevious) + PTP(BlendingPrevious)

1.3.6.5. Buffer Mode: BlendingNext (0x36 = 4)

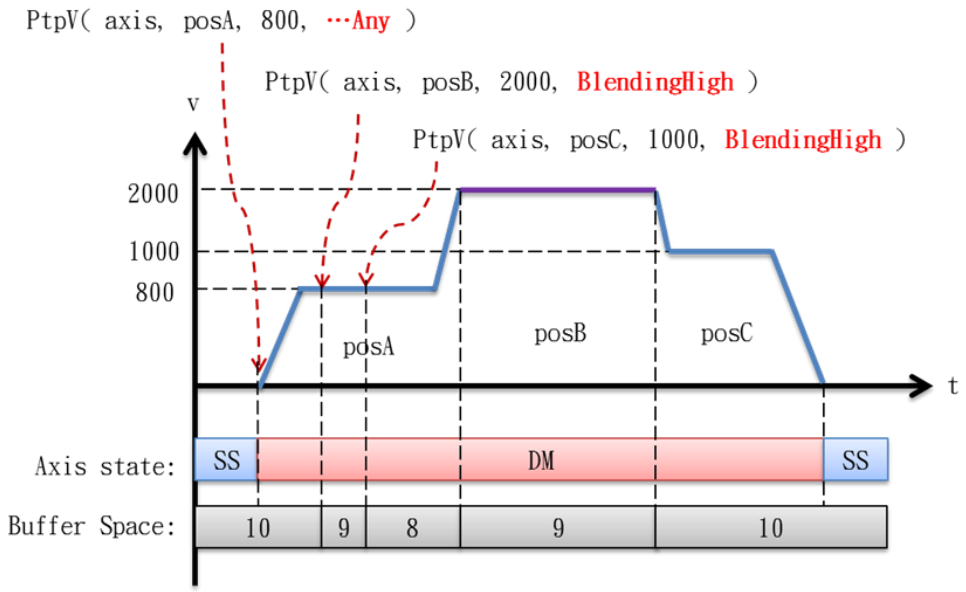
新的運動命令將被儲存在運動隊列中，並與前一個輸入的運動命令以新的運動命令之目標速度連結起來。以下圖來看，新輸入的點對點運動與前一個運動的連接，皆以新的運動之目標速度相連，因此，在移動到目標點位元 posB 與 posC 時，點對點運動的初速度皆已經到達目標速度。



PTP + PTP(BlendingNext) + PTP(BlendingNext)

1.3.6.6. Buffer Mode: BlendingHigh (0x36 = 5)

新的運動命令將被儲存在運動隊列中，並與前一個輸入的運動命令以較高的目標速度連結起來。以下圖來看，由於第二個點對點運動的目標速度大於第一個點對點運動，因此，到達目標點位元 posA 時，末速度會加速到 2000，並以 2000 為初始速度移動到目標點位元 posB，由於第三個點對點運動的目標速度也小於 2000，因此，到達目標點位元 posB 時，末速度為 2000。



PTP + PTP(BlendingHigh) + PTP(BlendingHigh)

1.3.7. 單軸 JOG 運動

在單軸運動中,JOG 運動也是一個很常使用到的運動形式,使用者可以呼叫NMC_AxisJog(),運動控制模組將依據相關單軸參數進行速度曲線規劃,將單軸加減速到目標速度,並以此目標速度持續運動下去。因此,當單軸在此運動模式下,其狀態稱為連續運動(NMC_AXIS_STATE_CONTINUOUS_MOTION)。

JOG 運動相關之相關單軸參數:

Param. Num.	Sub. Index	說明	備註
0x28	0	Base velocity (unit/sec)	
0x31	0	Profile type	
0x32	0	Max. velocity (unit/sec)	
0x33	0	Acceleration (unit/sec2)	
0x35	0	Jerk (unit/sec3)	

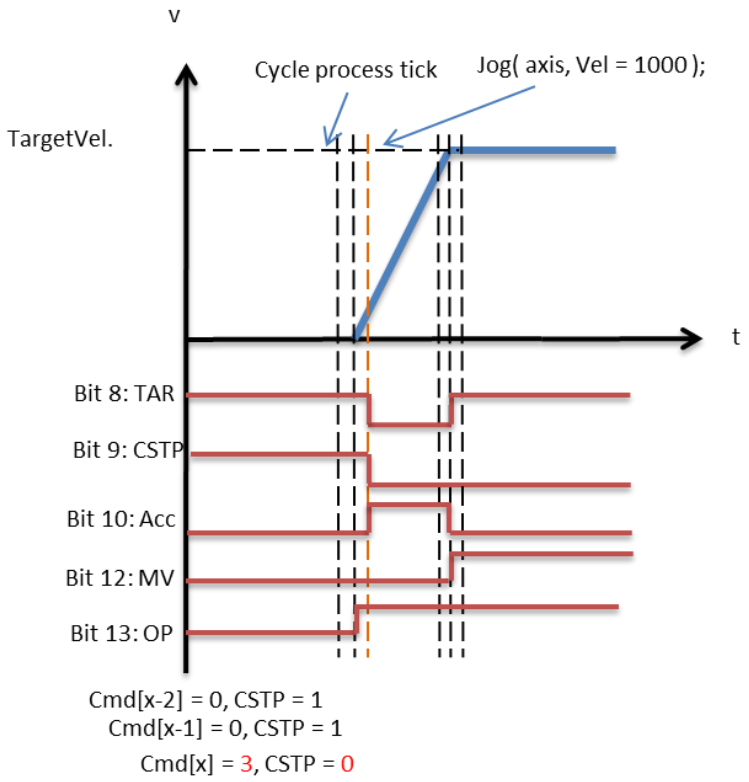
與 JOG 運動相關的單軸參數包括:目標速度(0x32)、加速度(0x33)、速度曲線形式(0x31)、Jerk(0x35)。若使用者想要指定不同於 0x32 設定的目標速度,也可以透過指標 PMaxVel 將所欲的目標速度傳入,參數 0x32 的設定值也會連動改變。

成功啟動 JOG 運動後,單軸狀態會切換到「NMC_AXIS_STATE_CONTINUOUS_MOTION」,有關啟動 JOG 運動前與啟動後的單軸狀態變化,請參閱下列表格:

運動命令 / 目前狀態	NMC_AxisJog()

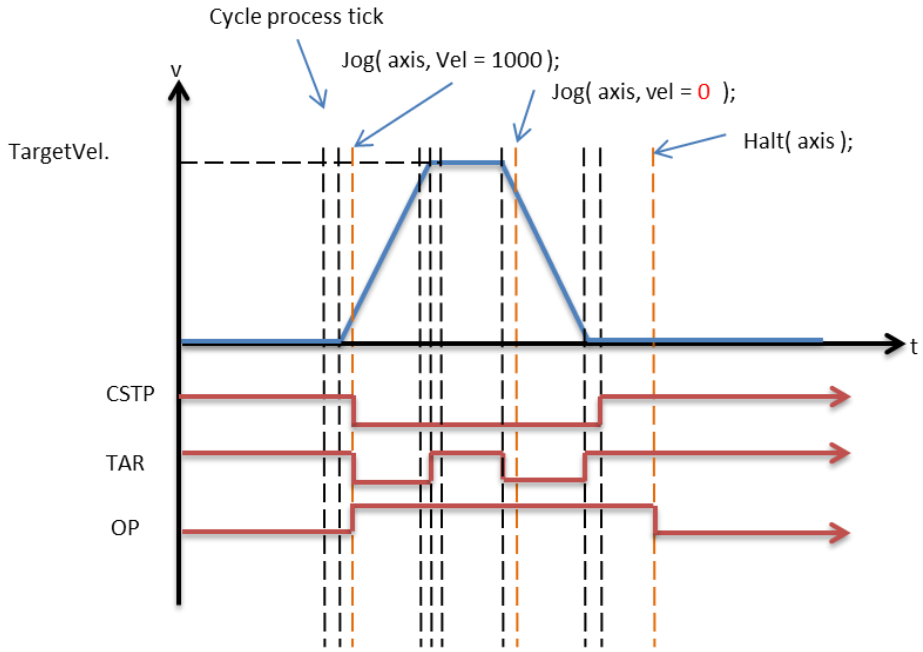
NMC_AXIS_STATE_DISABLE	禁止，回傳錯誤
NMC_AXIS_STATE_STAND_STILL	狀態改變至 NMC_AXIS_STATE_CONTINUOUS_MOTION
NMC_AXIS_STATE_HOMING	禁止，回傳錯誤，正在進行的 homing 不會被影響
NMC_AXIS_STATE_DISCRETE_MOTION	依照單軸參數:0x36(Buffer mode)設定儲存到運動隊列或立即執行。若為 Aborting mdoe 狀態改變為 NMC_AXIS_STATE_CONTINUOUS_MOTION
NMC_AXIS_STATE_CONTINUOUS_MOTION	依照單軸參數:0x36(Buffer mode)設定值儲存到運動隊列或立即執行。當 Buffer mode 為 Blending，禁止，回傳錯誤
NMC_AXIS_STATE_STOPPING	禁止，回傳錯誤
NMC_AXIS_STATE_STOPPED	禁止，回傳錯誤
NMC_AXIS_STATE_WAIT_SYNC	依照單軸參數:0x36(Buffer mode)決定 JOG 命令被儲存到運動隊列 1. Abort: 清除運動隊列後存入 2. Buffered: 存入運動隊列 3. Blending: 存入運動隊列 若 Buffer mode 為 Blending，而儲存在運動隊列的上一個運動命令為 JOG，則回傳錯誤。
NMC_AXIS_STATE_ERROR	禁止，回傳錯誤

對於 JOG 運動來說，當單軸到達目標速度，其運動資訊(Axis Status)中的 TAR(bit 8)轉變為 1，代表目標速度已經到達。此外，由於到達目標速度後，單軸仍然以此速度繼續運動，所以運動資訊(Axis Status)中的 MV(bit 12)會保持為 1。



JOG 運動資訊(Axis Status)時序圖

當成功啟動 JOG 運動後,若將 JOG 速度重設為 0 (Aborting)或利用 NMC_AxisSetVelRatio() 將速度百分比設定為 0，則單軸將降速到 0，相關的運動狀態圖，如下圖所示:



啟動 JOG 運動後將速度百分比設定為 0 之運動狀態圖

當速度百分比設定為 0，目標速度將重新設定為 0，因此，單軸將由目前的速度減速到 0。當速度降為 0 之後，因為已經到達目標速度，所以運動狀態中的 TAR(bit 8) 轉為 1，且由於單軸已經靜止，因此運動狀態中的 CSTP (bit 9)會轉為 1。雖然單軸已經降速為 0，但是 JOG 的運動命令仍然作用在單軸上，一旦速度百分比設定為大於 0 的某個數值，單軸將再重新運動起來，因此(OP) bit 13 仍然保持為 1，因為單軸仍在 JOG 的運作中。若使用者呼叫停止運動 NMC_AxisHalt()，單軸將從 JOG 的運作脫離出來，因此，OP(bit 13)會轉變為 0。有關停止運動的相關行為，請參閱下面章節。

1.3.8. 運動停止

單軸在運動的過程中，若想終止目前正在進行的運動，可以呼叫 NMC_AxisHalt()或是 NMC_AxisStop()，兩者的差別，主要是 NMC_AxisHalt()是在正常狀況下呼叫，其降速的減加速度與速度曲線的形式乃依據單軸參數:0x34(Deceleration)與單軸參數:0x31(Profile type)的設定值，這些參數與單軸進行點對點(PTP)運動和 JOG 運動中所使用到的參數是一樣的。但若在緊急狀況下，欲使單軸停止目前正在進行的運動，則必須呼叫 NMC_AxisStop()，在降速的過程中所依據的減加速度與速度曲線形式為另一組參數，說明如下：

Param. Num.	Sub. Index	說明	備註
0x20	0	Profile type for AxisStop command	
0x21	0	Deceleration for AxisStop command (unit/sec2)	
0x22	0	Jerk for AxisStop command (unit/sec3)	

- 0x20: Profile type for AxisStop command

  呼叫 NMC_AxisStop()後，單軸在降速的過程中所依據的速度曲線形式。

- 0x21: Deceleration for AxisStop command (unit/sec2)

  呼叫 NMC_AxisStop()後，單軸在降速的過程中所使用的減加速度。此參數的設定值，不受限於軟體極限保護中的加速度極限值(0x12, SubIndex = 0)。一般來說，若希望單軸在緊急狀況下可以快一點降速到 0，0x21 的設定值會大於 0x34 的設定值。

- 0x22: Jerk for AxisStop command (unit/sec3)

  當參數 0x20 設定為 S 形速度曲線時，此參數可以指定 Jerk，與參數 0x35 可以區別開來。

以下，分別說明 NMC_AxisHalt()和 NMC_AxisStop()的使用方式：

- NMC_AxisHalt():

單軸在進行點對點運動、JOG 運動或是 Homing 運動，皆可呼叫此函式讓單軸進行正常停止運動，停止後，單軸狀態回復到「NMC_AXIS_STATE_STAND_STILL」。由於此函式視為單軸在正常情況下的運動形式，有關呼叫前與呼叫後的單軸狀態變化，如下表所示：

運動命令 / 目前狀態	NMC_AxisHalt()
NMC_AXIS_STATE_DISABLE	禁止，回傳錯誤
NMC_AXIS_STATE_STAND_STILL	狀態無變化
NMC_AXIS_STATE_HOMING	當運動結束後，狀態轉變為 STAND_STILL
NMC_AXIS_STATE_DISCRETE_MOTION	依照單軸參數:0x36(Buffer mode)設定儲存到運動隊列或立即執行，當運動結束後，狀態轉變為 STAND_STILL
NMC_AXIS_STATE_CONTINUOUS_MOTION	依照單軸參數:0x36(Buffer mode)設定儲存到運動隊列或立即執行 當運動結束後，狀態轉變為 STAND_STILL
NMC_AXIS_STATE_STOPPING	禁止，回傳錯誤
NMC_AXIS_STATE_STOPPED	禁止，回傳錯誤
NMC_AXIS_STATE_ERROR	禁止，回傳錯誤

- NMC_AxisStop()

單軸在進行點對點(PTP)運動、JOG 運動或是 Homing 運動，皆可呼叫此函式讓單軸進行緊急停止運動，當成功啟動此函式後，儲存於運動隊列中的運動命令會被清除，在停止過程中，單軸狀態會切換到「NMC_AXIS_STATE_STOPPING」；待單軸完全停止後，單軸狀態會切換到「NMC_AXIS_STATE_STOPPED」。當異常狀況排除後，在重新啟動單軸運動之前，必須呼叫 NMC_AxisResetState()，成功執行後，若單軸此時處於激磁狀態，則單軸狀態會回復到「NMC_AXIS_STATE_STAND_STILL」；若單軸處於非激磁狀態，則單軸狀態會切換到「NMC_AXIS_STATE_DISABLE」。有關呼叫 NMC_AxisStop()前與呼叫後的單軸狀態變化，如下表所示：

運動命令 / 目前狀態	NMC_AxisStop()
NMC_AXIS_STATE_DISABLE	狀態無變化
NMC_AXIS_STATE_STAND_STILL	當運動結束後，狀態轉變為 NMC_AXIS_STATE_STOPPED
NMC_AXIS_STATE_HOMING	當運動結束後，狀態轉變為 NMC_AXIS_STATE_STOPPED
NMC_AXIS_STATE_DISCRETE_MOTION	當運動結束後，狀態轉變為 NMC_AXIS_STATE_STOPPED
NMC_AXIS_STATE_CONTINUOUS_MOTION	當運動結束後，狀態轉變為 NMC_AXIS_STATE_STOPPED
NMC_AXIS_STATE_STOPPING	狀態無變化
NMC_AXIS_STATE_STOPPED	狀態無變化
NMC_AXIS_STATE_ERROR	狀態無變化

上述兩個函式僅適用于單一個單軸的停止運動，因此，必須輸入單軸的索引號碼。若使用者欲將整個系統中所有的單軸停止下來，可以呼叫相對應的 NMC_AxisHaltAll()或是 NMC_AxisStopAll()。

1.3.9. 運行中速度改變

當單軸在進行運動的過程中，可以呼叫相關函式改變原先設定的目標速度與加減速度，以下說明相關函式的使用方式與限制。

- NMC_AxisVelOverride()

只有當單軸正在進行點對點(PTP)運動或是 JOG 運動時，才可以呼叫此函式改變目標速度，否則將會回傳錯誤碼。此函式輸入的參數為目標速度的絕對量值，並不受到速度百分比的影響，且不會改變單軸參數:0x32 (Max. velocity)的設定值。舉例來說，假設單軸正在進行 JOG 運動，成功呼叫此函式後，不管此時的速度百分比為何，目標速度的量值都將更改為輸入此函式的參數(TargetVel)，但單軸參數:0x32 (Max. velocity) 仍維持原先的設定值。由於 JOG 運動的主要目標乃是讓單軸以輸入的目標速度持續運動，因此，成功呼叫 NMC_AxisVelOverride()後，單軸必定會運動到指定的目標速度。但若單軸進行的是點對點(PTP)運動，且已經進入到降速階段-單軸運動資訊 (Status of axis)的 DEC(bit 11)為 1，則呼叫此函式對於目前正在進行的點對點運動不會有任何影響；若單軸是在加速或等速度區間，則會依據目前速度、指定的目標速度以及殘餘距離，判定是否可以到達使用者指定的目標速度，若無法到達使用者指定的目標速度，則會依據所需的加減速距離，合理修改目標速度。

- NMC_AxisAccOverride()

只有當單軸正在進行點對點(PTP)運動或是 JOG 運動時，才可以呼叫此函式改變目標加速度，否則將會回傳錯誤碼。此函式輸入的參數為目標加速度的絕對量值，且會連動改變單軸參數:0x33 (Acceleration)的設定值。就單軸運動狀態來看，只有當單軸處於加速到目標速度的狀態下(Axis status 的 ACC-bit 10 變為 1)，呼叫此函式才會因為改變加速度而對目前的運動有所作用；若單軸已經到達目標速度(Axis status 的 MV-bit 12 變為 1)或是進入到降速階段(Axis status 的 DEC-bit 11 變為 1)，則呼叫此函式對於目前的運動不會有任何作用。

- NMC_AxisDecOverride()

只有當單軸正在進行點對點運動或是停止運動(呼叫 NMC_AxisHalt)時，才可以呼叫此函式改變減加速度(Dec)，否則，將會回傳錯誤碼。此函式輸入的參數為目標減加速度(Dec)的絕對量值，且會連動改變單軸參數:0x34 的設定值。就單軸運動形式來看，當單軸正在進行點對點運動，且運動狀態處在加速到目標速度或是已經到達目標速度，呼叫此函式對於目前運動可能會有作用，主要是因為加減速距離改變而連帶產生的作用，譬如：原先到達不了指定的目標速度，但因為呼叫此函式，新設定的 Dec 使得需要的加速度距離變短，因而可以到達指定的目標速度。但是，若單軸運動狀態已經進入降速到末速度(Axis status 的 DEC-bit 11 變為 1)，則呼叫此函式對於目前的運動將不會有影響。當單軸因為呼叫 NMC_AxisHalt()正在進行停止運動，呼叫此函式將會改變目前的 Dec，因而影響停止所需的時間。

1.3.10. 單軸歸零運動

一般來說，設定單軸歸零的方式有兩種應用情境，第一種應用情境為，將單軸用任何方式移動至原點位置(或參考位置)，然後在目前的位置直接設定座標值。另一種為在單軸開始運作前，使用單軸的歸原點程式，選擇合適的歸零模式回到原點位置，以此位置作為座標基準點，再進行後續的運動；以下針對兩種歸零方式分別說明：

1.3.10.1. 手動設定原點位置

透過NMC_AxisSetHomePos()設定原點位置，只有在單軸狀態為「NMC_AXIS_STATE_DISABLE」或是「NMC_AXIS_STATE_STAND_STILL」時，使用者才可以透過呼叫此函式，將單軸目前的位置指定為某座標值，否則將回傳錯誤碼。成功呼叫此函式後，單軸的實際位置(Actual position)與命令位置(Command position)都會變更為指定的座標值，且運動控制模組會自動計算伺服馬達位置偏移量(position offset)並自動儲存於單軸參數:0x08 (Position offset)之中。對於使用絕對式編碼器的單軸(單軸參數:0x05 設定為 Absolute)來說，可以在系統開機後，讀取到編碼器回傳的數值時，透過此 position offset 計算出單軸此時的位置座標。

1.3.10.2. 單軸歸零運動

使用 NMC_AxisHomeDrive()（？）所提供之歸零運動(Homing)功能，該 API 只有當單軸狀態為「NMC_AXIS_STATE_STAND_STILL」時，才可以呼叫此函式，啟動單軸進行回原點的程式，否則將回傳錯誤碼。

驅動器歸原點相關的單軸參數如下:

Param. Num.	Sub. Index	說明	備註
0x80	0	Number of home parameters	(*3)(*4)
	1	EtherCAT CiA HOME method	(*5)
	2	EtherCAT CiA HOME speed search switch	(*5)
	3	EtherCAT CiA HOME speed search zero	(*5)
	4	EtherCAT CiA HOME acceleration	(*5)
	5	EtherCAT CiA HOME offset	(*5)

(*3): 可設定範圍根據控制器的規格而不同

(*4): Read only

(*5): 可設定範圍根據受控裝置的規格而不同

由於回原點的運動程式，主要由驅動器提供的功能規格為主，因此，使用者在呼叫此函式進行歸零運動時，必須先詳閱驅動器使用手冊，確認需要輸入的參數，以及提供那些回原點模式(設定於上述單軸參數:0x80:1~5 中)。

若歸零運動(Homing)使用到原點訊號或正負極限訊號，必須先確認感測器(Home sensor)、正負極限(Limit sensor)是否已正確連接到驅動器的 I/O 接點，可透過讀取單軸運動資訊:RPEL(bit16)、RNEL(bit17)和 RHOM(bit18)訊號之變化確定感測器已經正確連接與設定。

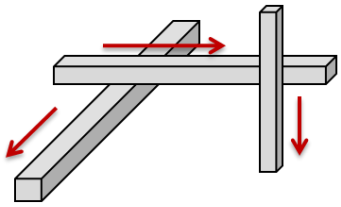
在正常情況下，當成功呼叫 NMC_AxisHomeDrive()函式後，控制器會將上述歸原點相關參數傳送給驅動器並自動進行用戶設定的回原點程式，在運動的過程中，單軸狀態將切換到「NMC_AXIS_STATE_HOMING」。一旦歸零程式成功完成後，單軸狀態將回復到「NMC_AXIS_STATE_STAND_STILL」，且單軸運動資訊中的 RHOM (bit 18)將轉變為 1。在進行歸零運動的過程中，若有發生不預期的錯誤，則單軸將停止歸零運動，且單軸狀態會切換到「NMC_AXIS_STATE_ERROR」，Axis status 的 ERR(bit 7) 會轉變為 1。舉例來說，若使用者輸入的歸零模式，驅動器本身並不提供，則呼叫此函式並成功執行一段時間後，單軸狀態會切換到「NMC_AXIS_STATE_ERROR」。另外，若歸零運動的參數設定不當，例如：加速度設定過小，以至於當觸發到外部極限感測器(Limit Sensor)後，因為減速距離過大，以至於在停止的過程中又脫離外部極限感測器(Limit Sensor)觸發的位置範圍，造成錯誤的狀況。當單軸在歸零過程中發生錯誤的狀況，單軸狀態進入「NMC_AXIS_STATE_ERROR」，使用者必須呼叫 NMC_AxisResetState()將單軸的狀態回復到「NMC_AXIS_STATE_STAND_STILL」後，才可以再執行其他的運動命令。

歸零運動視為單軸運動的一種，因此，在運動的過程中，可以呼叫 NMC_AxisHalt()、NMC_AxisDisable()或是 NMC_AxisStop()終止歸零程式，呼叫後的單軸行為，與一般單軸在運動狀況下呼叫相關函式的行為一樣。

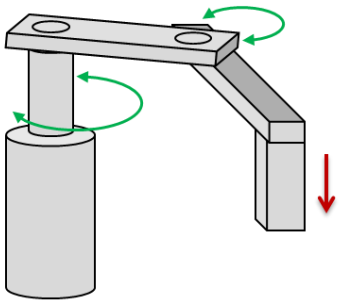
1.4. 群組(Group)控制

NexMotion 可定義若干軸為一個群組(Group)，而一群組代表一個有特定結構關係的機構或機器人(Robot)，目前已支持下列常見的工業型機器人:

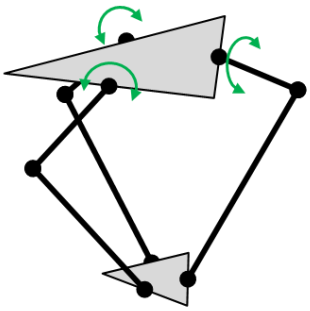
1. 直角坐標機構(Robot)
2. SCARA Robot
3. Delta Robot
4. 關節型(Arteculated) Robot



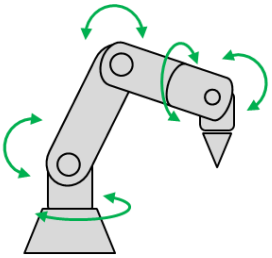
Linear Robot



SCARA Robot



Delta Robot



Articulated Robot

1.4.1. 設定群組

可透過 NexMotion Studio 匯入機構之設定，其步驟如下:

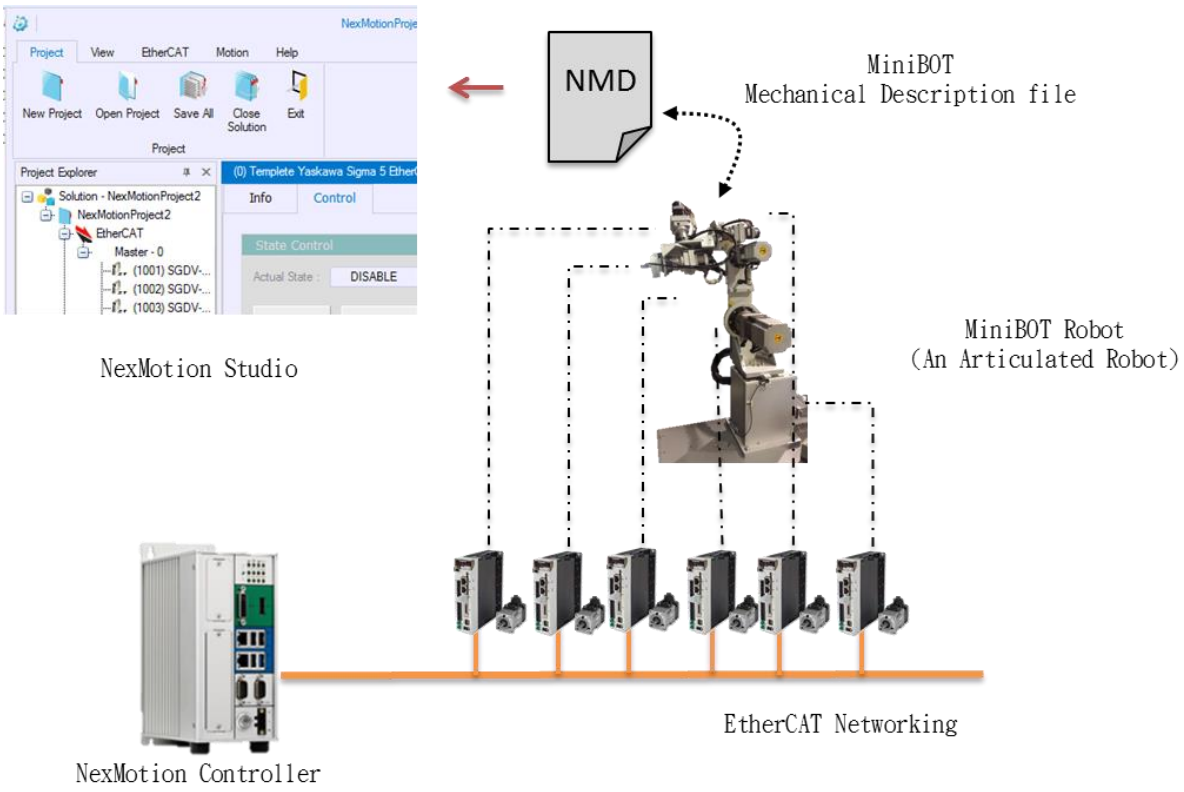
1. 透過 NexMotion Studio 匯入一個機構描述(Mechnical Description, NMD)檔案
2. 映射伺服馬達到群組各軸
3. 測試群組，同時產生受控系統組態檔(NCF 檔案)

詳細步驟可參考 NexMotion Studio 使用手冊。

也可以手動設定一個群組，可先利用 NexMotion Studio 先匯入一個類似的機構當作樣板(Templete)，從現有樣板再去修改調整成符合實際的機構型式，其步驟如下:

1. 透過 NexMotion Studio 匯入一個機構描述(Mechnical Description, NMD)檔案，作為基礎
2. 修改各軸單位參數(參考單軸單位設定)

3. 修改運動學參數(參考運動學設定)
4. 映射伺服馬達到群組各軸
5. 測試群組，同時產生受控系統組態檔(NCF 檔案)



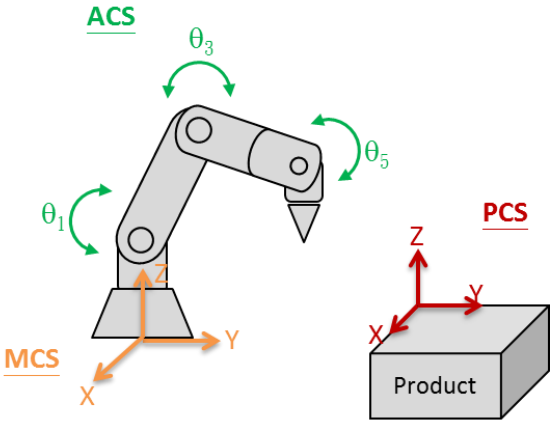
透過 NMD 檔案匯入一機構(Robot)資料

1.4.2. 坐標系說明

吾欲對群組(或稱 Robot)的各軸或其末端點(Tool center point,TCP)位置進行控制，NexMotion 提供了三種坐標系來描述 Robot 的位置，其中有

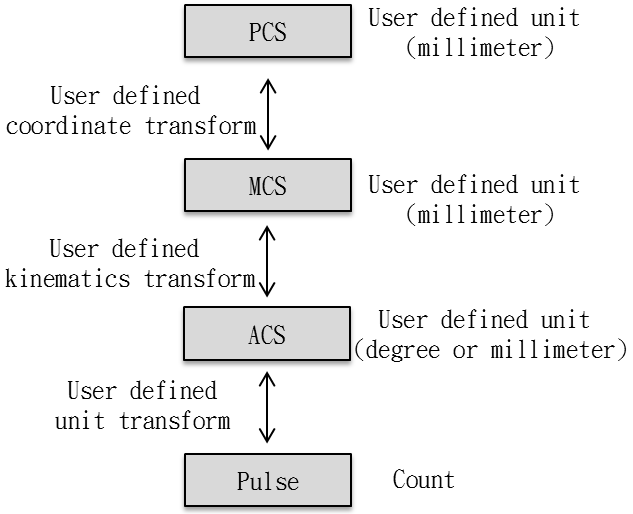
1. 軸坐標系(Axis Coordinate System, ACS)
2. 機械坐標系(Machine Coordinate system, MCS)或俗稱大地坐標系
3. 工件坐標系(Product Coordinate System, PCS)

下圖為各坐標系之示意圖;



ACS、MCS 和 PCS 坐標系之定義

NexMotion 已經內建各坐標系間之單位轉換，使用者可透過參數設定來定義之轉換關係，下圖為各坐標系之單位轉換關係圖:



單位轉換關係圖

轉換關係相關參數

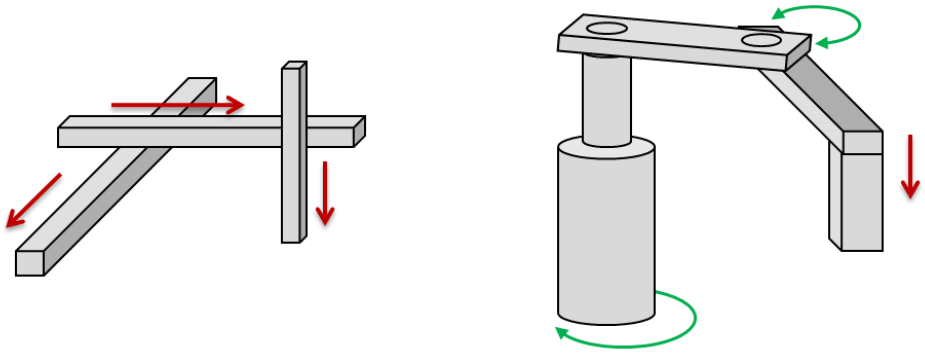
轉換類別	參數類別	說明
單位轉換	各群組軸參數 0x00~0x06	設定伺服的脈波數和 ACS 坐標系物理單位之轉換
運動學轉換	群組參數 0x00	設定 ACS 坐標系和 MCS 坐標系之轉換
坐標系轉換	群組參數 0xC0~DF	設定 MCS 和 PCS 坐標系之轉換
TCP 座標轉換	群組參數	設定 Tool 之 TCP 座標轉換

	0x80~8F	

在程式方面，NexMotion 定義了「Pos_T 結構」和「Xyz_T 結構」來描述座標資料。Pos_T 結構可用來描述 ACS, MCS, 或 PCS 座標資料，而 Xyz_T 結構專門用來描述 MCS 和 PCS 的 x, y, z 軸位置。

1.4.2.1. ACS 坐標系(Axis Coordinate System)

ACS 坐標系為描述群組機構(或稱 Robot)中各關節軸的位置，一 Robot 系統可能具有各種不同的活動關節，我們可將各關節視為一個軸(Axis)，並透過軸坐標系(ACS)來描述該軸的位置。較常見的關節可概分為直線關節(Linear joint)和旋轉關節(Rotary joint)兩種，使用者可以根據其機構形式來決定位置的單位定義，舉例來說，若為直線關節可定義其物理單位為毫米(mm)，若為旋轉關節可定義為角度(Degree)單位。

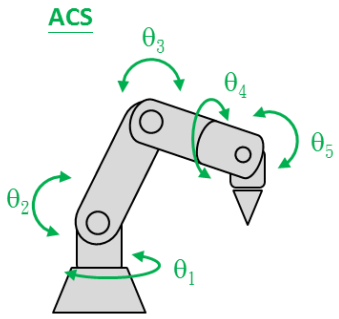


Robot 的關節型式

控制器提供從伺服的脈波單位(Pulse count)到機構之物理單位之轉換，其設定方式請參考 單軸單位設定

一個群組最多可以有 8 個軸，其實際軸數依照機構的型態決定

下圖為一關節型六軸機械手臂之 ACS 坐標系描述方式



Name	Unit
θ_1	Degree
θ_2	Degree
θ_3	Degree
θ_4	Degree
θ_5	Degree
θ_6	Degree

範例:關節型機械手臂之

ACS 座標參數標記法

下面是一個使用「Pos_T 結構」來表示軸座標(ACS)的範例:

```
#include "NexMotion.h"
#include "NexMotionError.h"

int main()
{
    Pos_T point;

    // "point" represents as ACS position.
    point.pos[0] = 0.0;     // Degree of joint 1
    point.pos[1] = 90.0;    // Degree of Joint 2
    point.pos[2] = 0.0;     // Degree of Joint 3
    point.pos[3] = 0.0;     // Degree of Joint 4
    point.pos[4] = 0.0;     // Degree of Joint 5
    point.pos[5] = 0.0;     // Degree of Joint 6

}
```

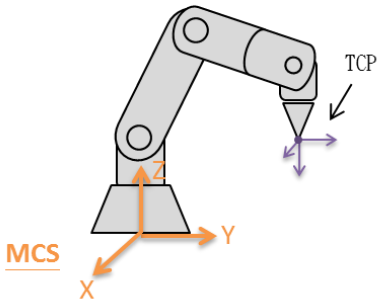
要讀取群組之 ACS 座標位置使用「Pos_T 結構」可用下列 API 達成

NMC_GroupGetCommandPosAcs() => 讀取 ACS 命令位置

NMC_GroupGetActualPosAcs() => 讀取ACS實際位置

1.4.2.2. MCS 坐標系(Machine Coordinate system)

MCS 座標是用來表示為工具末端點(Tool Center Point, TCP)相對於機械原點的位置姿態，其座標原點通常定義在機台安裝點 ，下圖為一關節型六軸機械手臂之 ACS 坐標系描述方式



Name	Unit
X	mm
Y	mm
Z	mm
A	Degree
B	Degree
C	Degree

範例:關節型機械手臂之
MCS 座標參數標記法

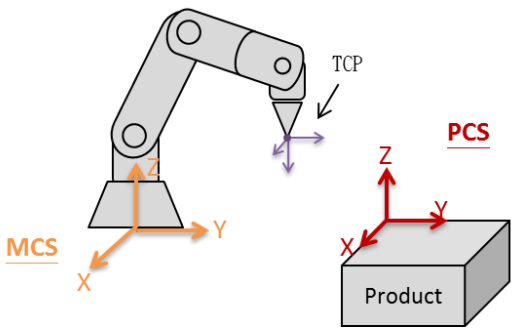
描述末端點最高為六個自由度，其標記法: X,Y,Z,A,B,C，

- 位置: X,Y,Z 為直角坐標系標記法，遵循右手定則(Right-hand rule)
- 姿態: A,B,C 為尤拉角 intrinsic Z-Y-X 標記法。分別依序對 Z，Y 和 X 軸旋轉。

MCS 坐標系通常使用在歸原點的階段，歸完原點後在應用時，習慣上直接使用 PCS 坐標系來表示群組在直角坐標系下的位置。

1.4.2.3. PCS 坐標系(Product Coordinate System):

PCS 坐標系是用來表示為工具末端點(Tool Center Point, TCP)相對於工件座標原點的位置姿態，其座標原點通常定義在待加工之物件上 ，下圖為一關節型六軸機械手臂之 PCS 坐標系描述方式:



Name	Unit
X	mm
Y	mm
Z	mm
A	Degree
B	Degree
C	Degree

範例:關節型機械手臂之
PCS 座標參數標記法

如同 MCS 坐標系，PCS 坐標系描述末端點最高為六個自由度，其標記法: X,Y,Z,A,B,C

- 位置: X,Y,Z 為直角坐標系標記法，遵循右手定則(Right-hand rule)
- 姿態: A,B,C 為尤拉角 intrinsic Z-Y-X 標記法。分別依序對 X，Y 和 X 軸旋轉。

PCS 坐標系與 MCS 坐標系之差別在於座標轉換之關係，因此當轉換關係為初始值 PCS 坐標系和 MCS 坐標系為重迭，換言之，當我們不特別設定工件坐標系時，PCS 坐標系就等於 MCS 坐標系。

下面是一個使用「Pos_T 結構」來表示機械座標(PCS)的範例:

```
#include "NexMotion.h"
#include "NexMotionError.h"

int main()
{
    Pos_T point;

    // "point" represents as PCS position.
    point.pos[0] = 454.5;   // X axis
    point.pos[1] =   0.0;   // Y axis
    point.pos[2] = 755.0;   // Z axis
    point.pos[3] = -90.0;   // A axis
    point.pos[4] = -90.0;   // B axis
    point.pos[5] = -90.0;   // C axis

}
```

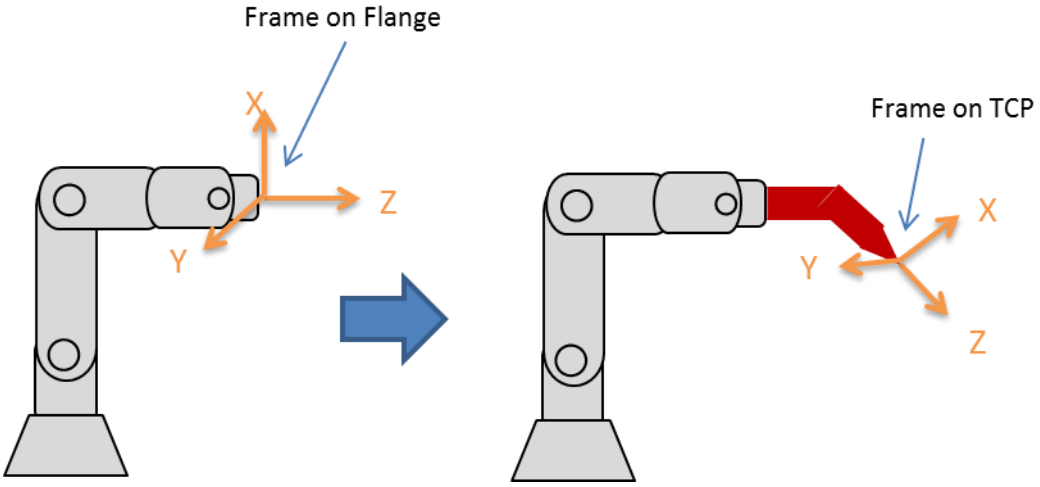
要讀取群組之 PCS 座標位置使用「Pos_T 結構」可用下列 API 達成

NMC_GroupGetCommandPosPcs() => 讀取PCS命令位置

NMC_GroupGetActualPosPcs()=> 讀取PCS實際位置

1.4.2.4. Tool 設定

當使用者尚未設定 Tool 時，運動指令中的目標位置(Target position)代表機器人末端點法蘭面(Flange)的位置與姿態，如下圖。當 Tool 參數被設定使用後，TCP(Tool center point)坐標系根據轉換關係附著於 Tool 之末端點。

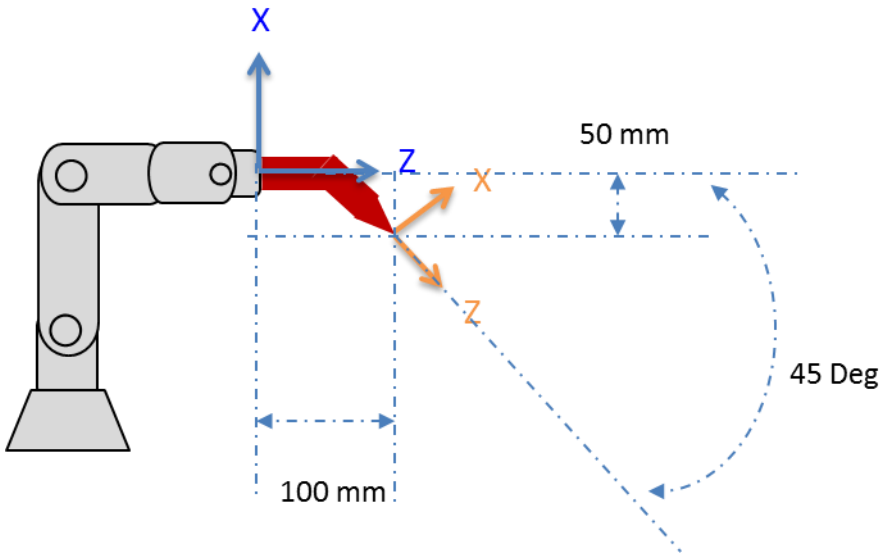


目前系統最多可支援 16 組 Tool 設定，其設定方式為設定群組參數 0x80~0x8F。定義如下:

Param. Num.	Sub. Index	資料型態	說明
0x80~8F	0	F64_T	Offset along flange x-axis
	1	F64_T	Offset along flange y-axis
	2	F64_T	Offset along flange z-axis
	3	F64_T	Rotation angle about flange z-axis
	4	F64_T	Rotation angle about flange y-axis
	5	F64_T	Rotation angle about flange x-axis

設定範例:

吾欲設定一組工具 Tool (Index = 0)如下圖，其 TCP 座標原點相對於 Flange 坐標系 x 軸為-50 單位，y 軸為 0 單位，Z 軸為 100 單位，另外坐標系相對於 Flange 之 y 軸旋轉-45 度:



其群組設定參數如下:

設定值如下:

NUM	Sub	Value	Description
0x80	0	-50	Offset along flange x-axis
	1	0	Offset along flange y-axis
	2	100	Offset along flange z-axis
	3	0	Rotation angle about flange z-axis
	4	-45	Rotation angle about flange y-axis
	5	0	Rotation angle about flange x-axis

使用方式為當運動指令的目標位置(Target postion)若該運動指令沒有特別指定 Tool index，則系統會參考群組參數 0x40:0 之設定當作目前系統所使用的 Tool

Param. Num.	Sub. Index	資料型態	說明
0x40	0	I32_T	Tool index selection for motion target

1.4.2.5. Tool 教導

設定 Tool 的方式除了直接輸入參數外，另外也提供數種使用姿態點位元方式換算 Tool 的參數 APIs:

- TCP 平移教導法
- TCP 平移與 Z 方向設定教導法
- TCP 平移與姿態設定教導法
- 姿態設定教導法

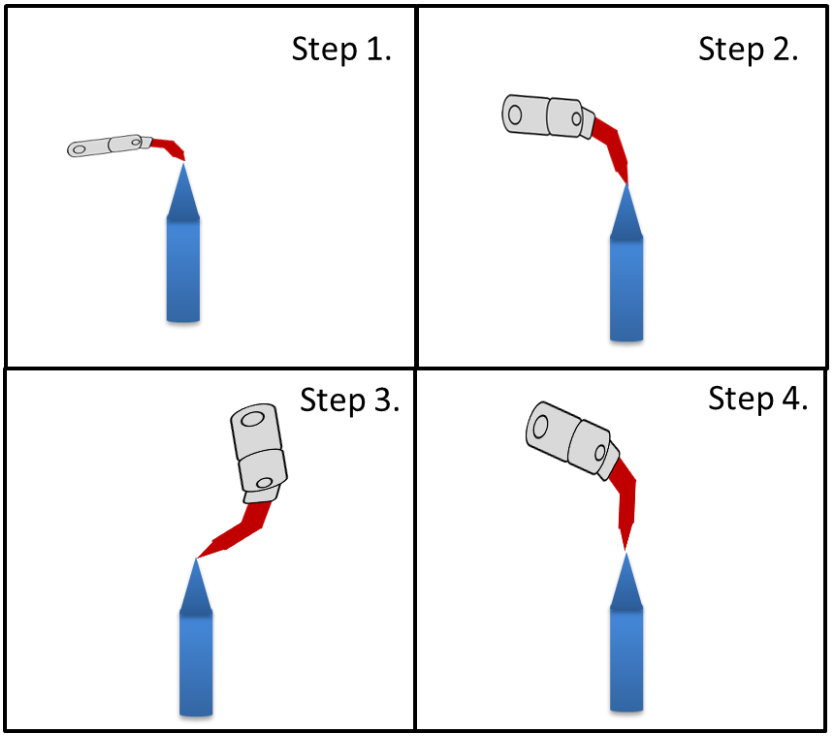
所對應使用的 API 如下:

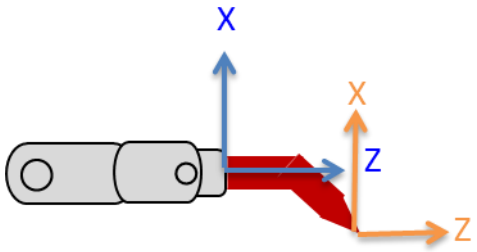
函數名稱	說明
NMC_ToolCalib_4p	Tool 教導- TCP 平移教導法
NMC_ToolCalib_4pWithZ	Tool 教導- TCP 平移與 Z 方向設定教導法
NMC_ToolCalib_4pWithOri	Tool 教導- TCP 平移與姿態設定教導法
NMC_ToolCalib_Ori	Tool 教導- TCP 姿態設定教導法

注意,設定 Tool 過程中,可以選擇在 MCS 坐標系中進行教導,或者在任何一個 Base 中進行教導。切勿教導過程中變更 Base 設定以避免教導錯誤。

- TCP 平移教導法

透過 TCP 位置固定,給定四個不同姿態找出 TCP 和手臂法蘭面(Flange)之間的關係,此方法所找出的 TCP 只有平移關係,ABC 角無變化(為 0)

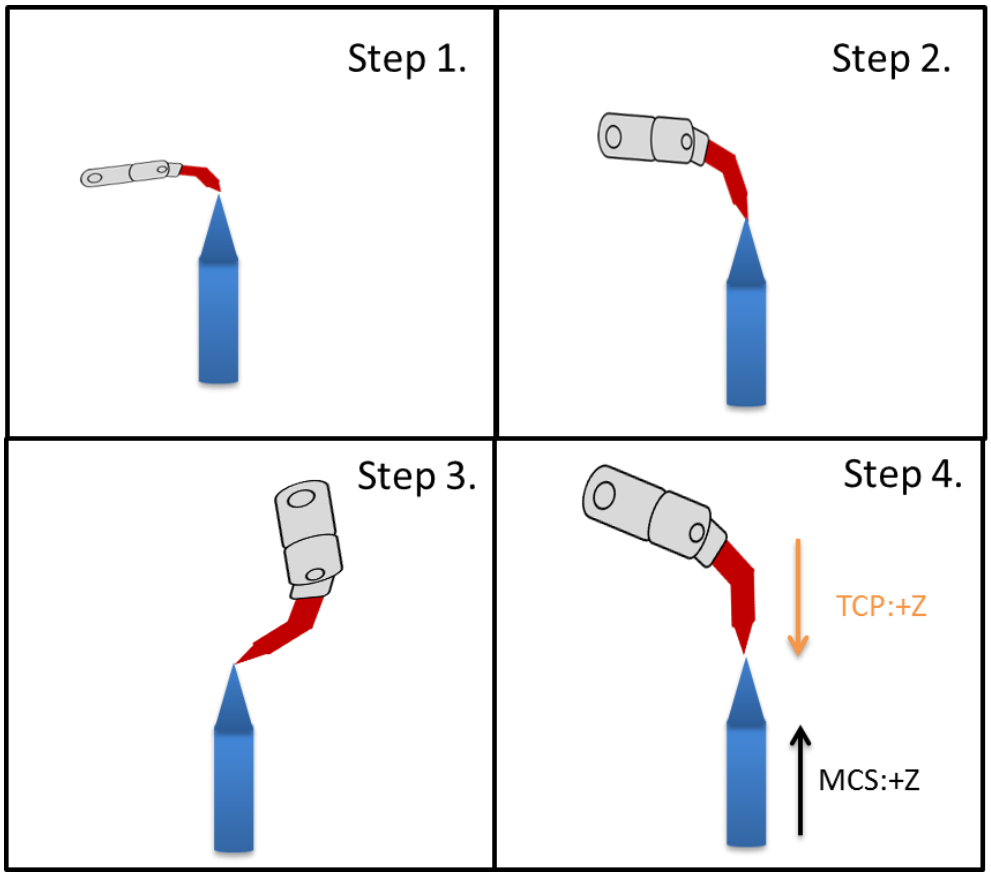


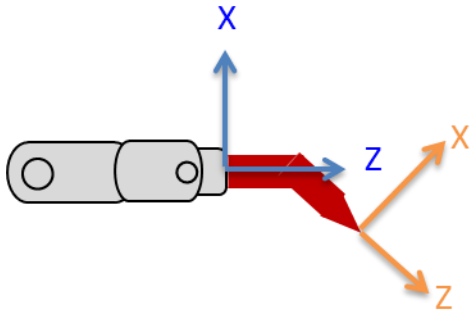


TCP 座標為 Flange 座標平移

- TCP 平移與 Z 方向設定教導法

透過 TCP 位置固定，給定四個不同姿態找出 TCP 和手臂法蘭面(Flange)之間的關係，第四步驟 TCP 的 Z 方向必須指向 MCS 的負 Z 方向。此方法所找出的 TCP 有平移關係和對 Flange-Y 軸旋轉

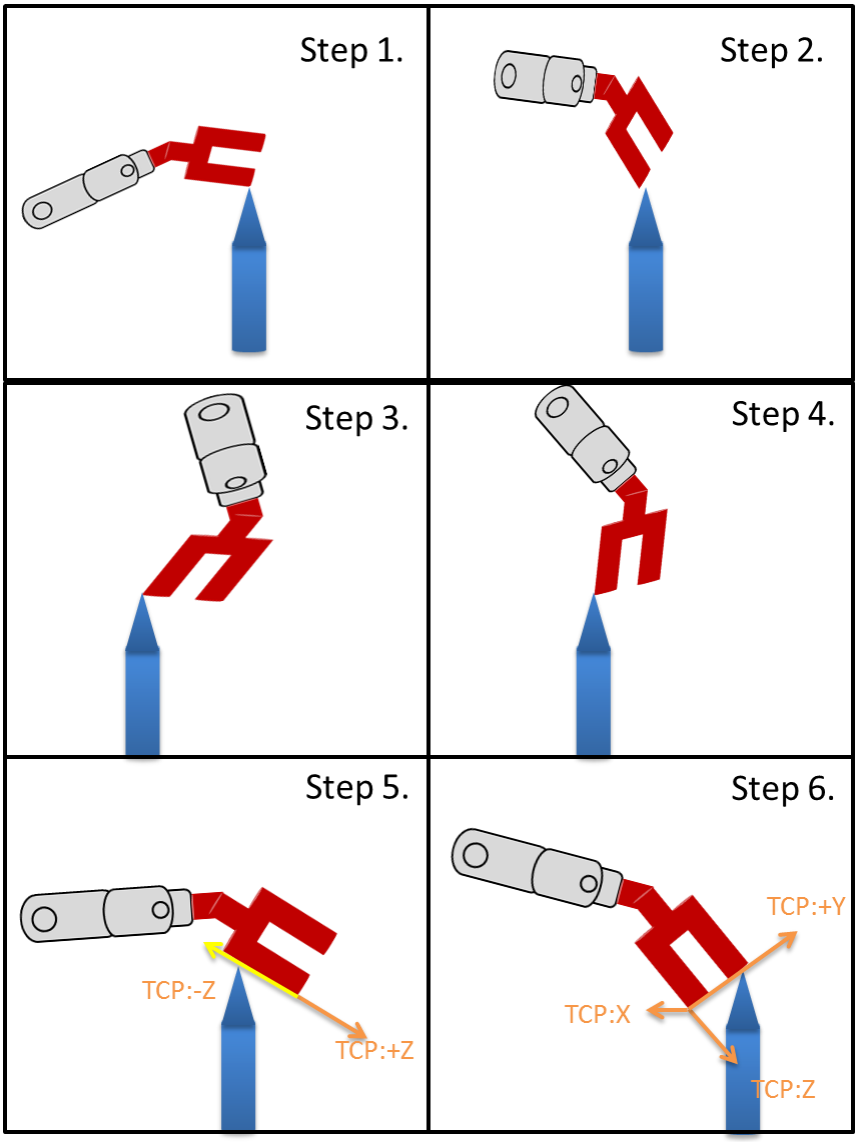


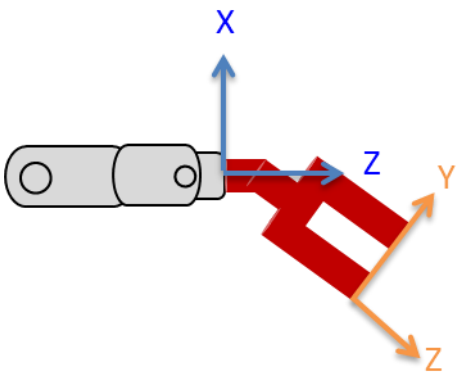


TCP 座標為 Flange 座標平移與對 Flange-y 軸旋轉之方向

- TCP 平移與姿態設定教導法

  此方法之前面四個步驟與「TCP 平移教導法」相同，主要是找出 TCP 與 Flange 座標之平移關係，步驟五與步驟六則是找出 TCP 座標的姿態。

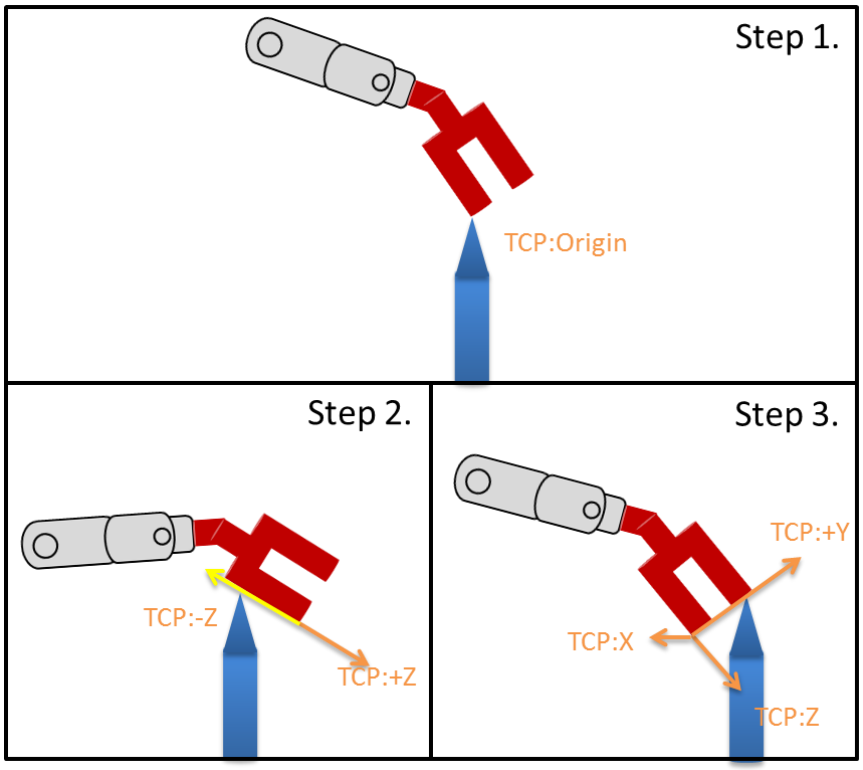




TCP 座標與 Flange 座標之關係

- 姿態設定教導法

這個方法是用來教導 TCP 姿態，可以配合「TCP 平移教導法」先找出平移關係，再使用本方法找出姿態設定。或者只想重新校正姿態時可使用本方法。一般不會獨立使用。

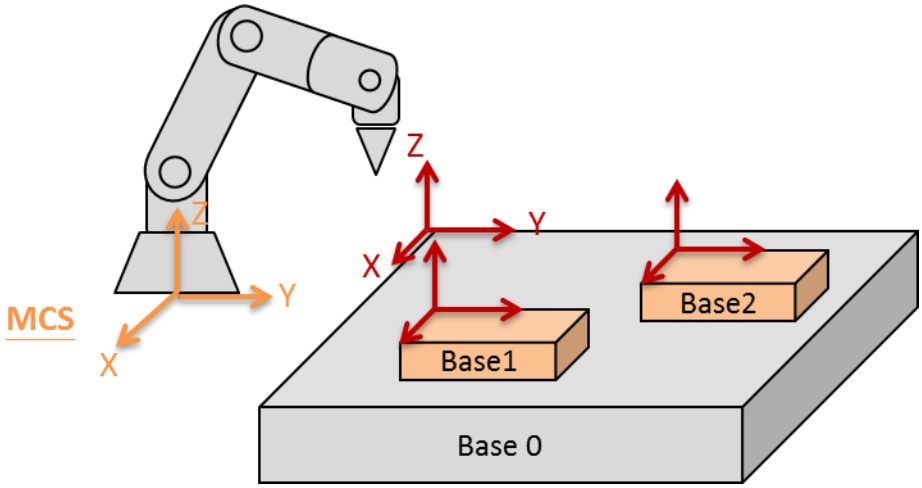


1.4.2.6. Base 設定

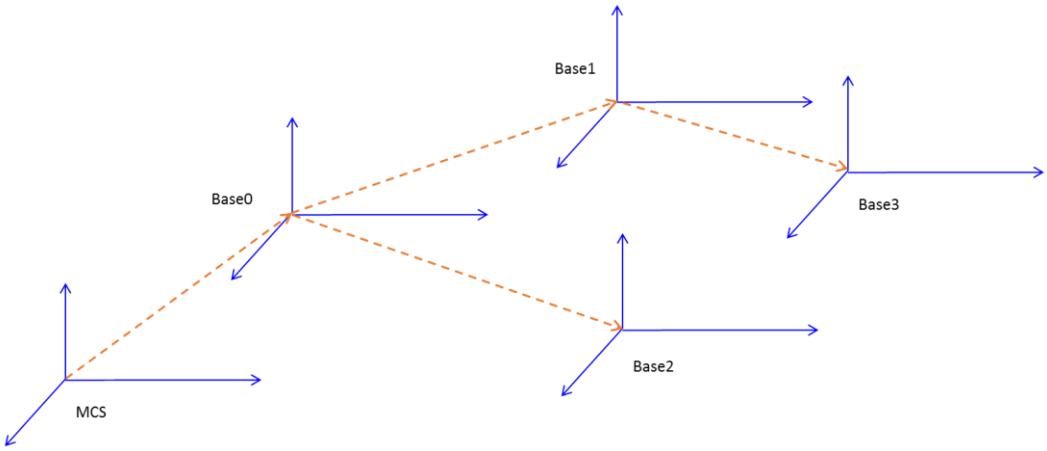
設定 Base 就是定義 PCS 坐標系，其最大好處在於，所有目標點位元相對於參考坐標系(Base)，當參考坐標系變動時，只需從新設定參考坐標系，目標點位元不需要從新教導即可利用。

NexMotin 之 PCS 座標轉換特色如下:

- 最高支持 32 組 Base
- 具階層設定, 最高 3 階



Base 的設定具有階層性，如上圖, Base0 相對於 MCS，而 Base1 和 Base2 則相對於 Base0。當 Base0 有改變，Base1 和 Base2 也會跟著改變。



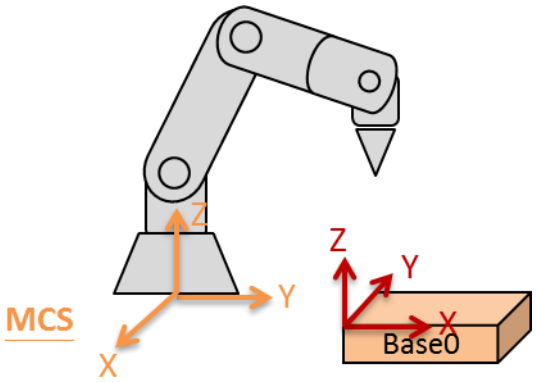
其設定方式為設定群組參數 0xC0~0xDF 共 32 組。定義如下:

Param. Num.	Sub. Index	資料型態	說明
0xC0~DF	0	F64_T	Offset along reference base x-axis
	1	F64_T	Offset along reference base y-axis
	2	F64_T	Offset along reference base z-axis
	3	F64_T	Rotation angle about reference base z-axis
	4	F64_T	Rotation angle about reference base y-axis
	5	F64_T	Rotation angle about reference PCS x-axis
	6	I32_T	Reference base index

設定範例:

吾欲設定 Base（Index = 0）之座標原點相對於 MCS 坐標系 x 軸為 10 單位,y 軸為 15 單位，

Z 軸為 5 單位，另外 Base0 坐標系相對於 MCS 之 Z 軸旋轉 90 度



xyz@MCS =(10, 15, 5)

設定值如下:

NUM	Sub	Value	Description
0xC0	0	10	Offset along reference base x-axis
	1	15	Offset along reference base y-axis
	2	5	Offset along reference base z-axis
	3	90	Rotation angle about reference base z-axis
	4	0	Rotation angle about reference base y-axis
	5	0	Rotation angle about reference PCS x-axis
	6	-1	Reference base index

使用方式為當運動指令的目標位置(Target postion)若該運動指令沒有特別指定 Base index，則系統會參考群組參數 0x48:0 之設定當作目前系統所使用的 Base

Param. Num.	Sub. Index	資料型態	說明
0x48	0	I32_T	Base index selection for motion target

1.4.2.7. Base 教導

設定 Base 的方式除了直接輸入外，另外提供教點位元自動校正的方式來設定 Base，自動校正的方式包括:

1. 一點(1P)教導法

2. 兩點(2P)教導法
3. 三點(3P)教導法

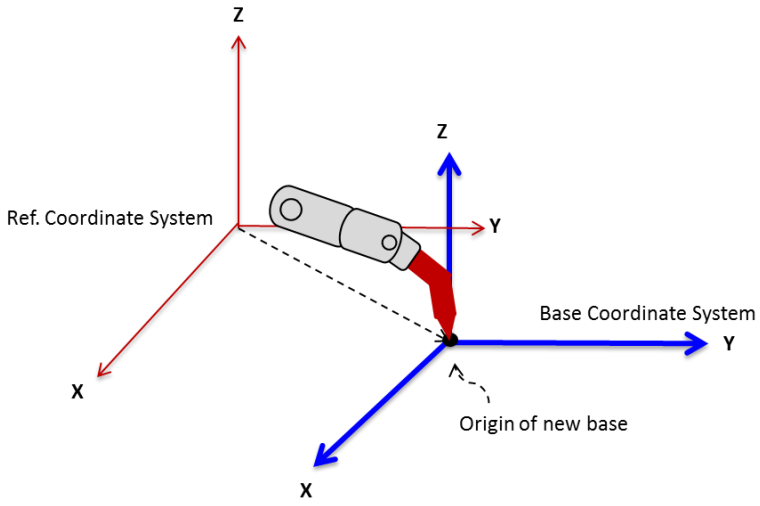
所對應使用的 API 如下:

函數名稱	說明
NMC_BaseCalib_1p	Base 教導-1p 法
NMC_BaseCalib_2p	Base 教導-2p 法
NMC_BaseCalib_3p	Base 教導-3p 法

顧名思義，一點法只需要教導一點，兩點法只需要交兩點，三點法則需要教三點。，其中，兩點法跟一點法的差別在於，兩點法所教導的 base 多了 Z 軸的旋轉，而三點法則可任意定義 X-Y-Z 軸。

- 「一點教導法」之步驟:

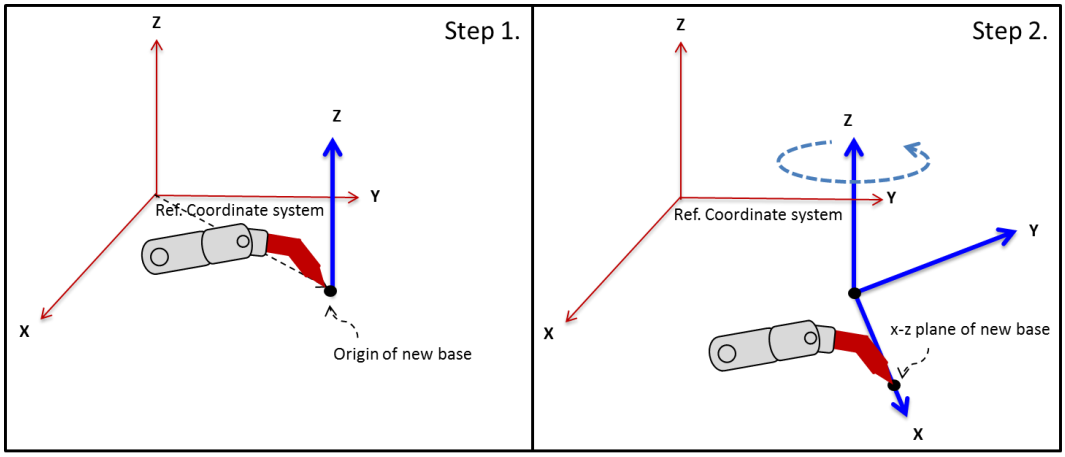
Step1: 定義 Base 座標原點



此方法為相當於參考坐標系平移到新的座標點上。

- 「兩點教導法」之步驟:

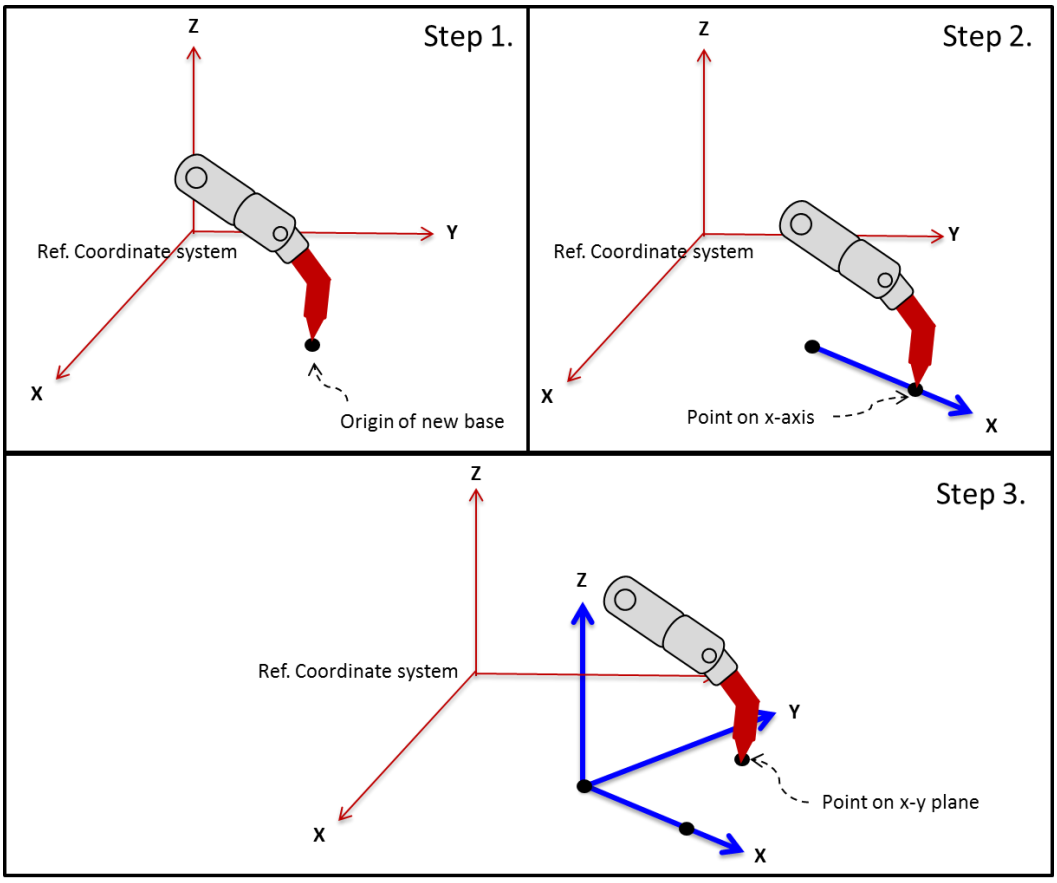
Step1: 定義 Base 座標原點
Step2: 定義 X-Z 平面上一點，X 軸為正方向



此方法為相當於參考坐標系平移到新的座標點上後再對 Z 軸做旋轉。

- 「三點教導法」之步驟:

Step1: 定義 Base 座標原點
Step2: 定義 X 軸上一點，X 軸為正方向
Step3: 定義 X-Y 平面上一點，Y 軸為正方向



1.4.3. 機構運動學設定

NexMotion 目前支持下列四種類型工業機器人

1. 線性型機器人(2~8 軸)
2. 六軸關節型(Arteculated) 機器人
3. Delta 型機器人
4. SCARA 型機器人

其設定方式於群組參數 0x00:0~48 中設定，其中定義如下:

參數 0x00:0 為構型選擇

參數 0x00:1~24 為機構尺寸參數

參數意義如下表:

構型 / Sub	線性型	六軸關節型	Delta 型	SCARA 型
0	0	1	2	3
1	Axis number (value=2~8 integer)	0 (*1)	f(mm)	a1 (mm)
2	N/A	a2 (mm)	e(mm)	a2 (mm)
3	N/A	a3 (mm)	rf(mm)	0 (*1)
4	N/A	a4 (mm)	re(mm)	0 (*1)
5	N/A	0 (*1)	hf(mm)	d1 (mm)
6	N/A	0 (*1)	he(mm)	0 (*1)
7	N/A	d1 (mm)	PEL-J1 (Deg)	d3 (mm)
8	N/A	0 (*1)	PEL-J2 (Deg)	d4 (mm)
9	N/A	0 (*1)	PEL-J3 (Deg)	PEL-J1 (Deg)
10	N/A	d4 (mm)	PEL-J4 (Deg)	PEL-J2 (Deg)
11	N/A	0 (*1)	MEL-J1 (Deg)	PEL-J3 (mm)
12	N/A	d6 (mm)	MEL-J2 (Deg)	PEL-J4 (Deg)
13	N/A	PEL-J1 (Deg)	MEL-J3 (Deg)	MEL-J1 (Deg)
14	N/A	PEL-J2 (Deg)	MEL-J4 (Deg)	MEL-J2 (Deg)
15	N/A	PEL-J3 (Deg)	Theta disable Value = 0(Delta4) Value = 1(Delta3)	MEL-J3 (mm)
16	N/A	PEL-J4 (Deg)	N/A	MEL-J4 (Deg)
17	N/A	PEL-J5 (Deg)	N/A	N/A
18	N/A	PEL-J6 (Deg)	N/A	N/A

19	N/A	MEL-J1 (Deg)	N/A	N/A
20	N/A	MEL-J2 (Deg)	N/A	N/A
21	N/A	MEL-J3 (Deg)	N/A	N/A
22	N/A	MEL-J4 (Deg)	N/A	N/A
23	N/A	MEL-J5 (Deg)	N/A	N/A
24	N/A	MEL-J6 (Deg)	N/A	N/A

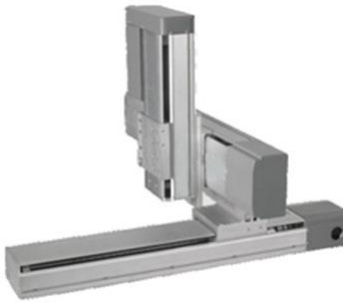
(*1) 保留，必須設定為0

- PEL: 正機構極限 (Positive End Limit)
- MEL: 負機構極限 (Minus End Limit)

1.4.3.1. 線性型機器人(2~8 軸)

線性型機器人為馬達機構配置為線性軸，可定義軸數為 2~8 軸。線性機器人其特性為 ACS, MCS 和 PCS 坐標系為重迭。

最常見的為 X-Y Table (2 軸)或 X-Y-Z 直角坐標機器人(3 軸)如下圖，直角坐標中的每一個方向都可以配置一顆馬達，其馬達運動則直接對應至卡式座標中的方向運動。



X-Y-Z Linear Axis

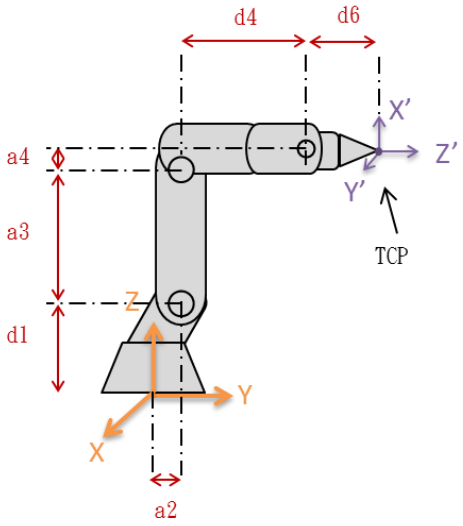
運動學參數(群組參數 0x00)定義如下表:

構型 / Sub	線性型 參數定義
0	0
1	軸數目 (value=2~8 integer)

1.4.3.2. 六軸關節型(Arteculated) 機器人

六軸關節型(Arteculated) 機器人其馬達機構配置與運動學參數如下圖，此系統可提供三維空間中的六個自由度。

運動學參數(群組參數 0x00)定義如下表:

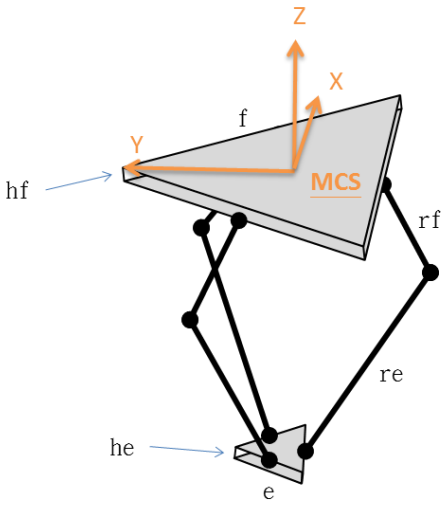
0x00 Sub. NO.	參數意義	圖示
1	0 (*1)	
2	a2 (mm)	
3	a3 (mm)	
4	a4 (mm)	
5	0 (*1)	
6	0 (*1)	
7	d1 (mm)	
8	0 (*1)	
9	0 (*1)	
10	d4 (mm)	
11	0 (*1)	
12	d6 (mm)	
13	PEL-J1 (Deg)	Joint 1 正極限
14	PEL-J2 (Deg)	Joint 2 正極限
15	PEL-J3 (Deg)	Joint 3 正極限
16	PEL-J4 (Deg)	Joint 4 正極限
17	PEL-J5 (Deg)	Joint 5 正極限
18	PEL-J6 (Deg)	Joint 6 正極限
19	MEL-J1 (Deg)	Joint 1 負極限
20	MEL-J2 (Deg)	Joint 2 負極限
21	MEL-J3 (Deg)	Joint 3 負極限
22	MEL-J4 (Deg)	Joint 4 負極限
23	MEL-J5 (Deg)	Joint 5 負極限
24	MEL-J6 (Deg)	Joint 6 負極限

(*1) 保留，必須設定為0

1.4.3.3. Delta 型機器人

Delta Robot 為並聯式機械手臂，其馬達機構配置與運動學參數如下圖，此系統可提供三維空間中的 XYZ 平移和 1 個 Z 軸方向旋轉。

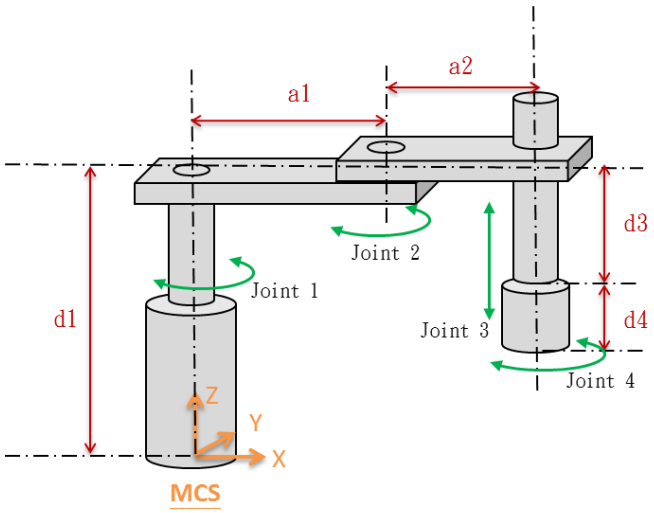
運動學參數(群組參數 0x00)定義如下表:

0x00 Sub. NO.	參數意義	圖示
1	f 上三角形邊長	 哪一個是第一軸，哪一個是第二軸?...
2	e 下三角形邊長	
3	rf 上杆長	
4	re 下杆長	
5	hf 上三角形厚度	
6	he 下三角形厚度	
7	PEL-J1 (Deg)	Joint 1 正極限
8	PEL-J2 (Deg)	Joint 2 正極限
9	PEL-J3 (Deg)	Joint 3 正極限
10	PEL-J4 (Deg)	Joint 4 正極限
11	MEL-J1 (Deg)	Joint 1 負極限
12	MEL-J2 (Deg)	Joint 2 負極限
13	MEL-J3 (Deg)	Joint 3 負極限
14	MEL-J4 (Deg)	Joint 4 負極限

1.4.3.4. SCARA 型機器人

Scara Robot 為水準關節機器，其馬達機構配置與運動學參數如下圖，此系統可提供三維空間中的 XYZ 平移和 1 個 Z 軸方向旋轉。

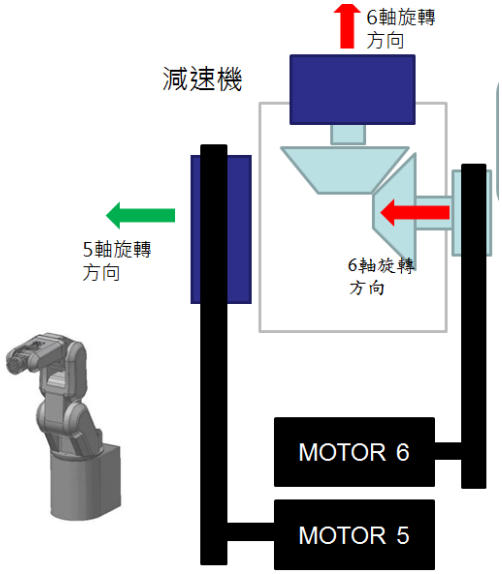
運動學參數(群組參數 0x00)定義如下表:

0x00 Sub. NO.	參數意義	圖示
1	a1 (mm)	
2	a2 (mm)	
3	0 (*1)	
4	0 (*1)	
5	d1 (mm)	
6	0 (*1)	
7	d3 (mm)	
8	d4 (mm)	
9	PEL-J1 (Deg)	Joint 1 正極限
10	PEL-J2 (Deg)	Joint 2 正極限
11	PEL-J3 (mm)	Joint 3 正極限
12	PEL-J4 (Deg)	Joint 4 正極限
13	MEL-J1 (Deg)	Joint 1 負極限
14	MEL-J2 (Deg)	Joint 2 負極限
15	MEL-J3 (mm)	Joint 3 負極限
16	MEL-J4 (Deg)	Joint 4 負極限

(*1) 保留，必須設定為0

1.4.4. 結構耦合補償功能

在群組的機構中，常見到軸於軸之間由於機構設計而產生耦合的情況，例如:六軸關節型機械手臂中第五與第六軸之間，由於傳動機構的連動，造成第五軸馬達在轉動時，第六軸外部關節結構也會隨之產生相對應的轉動，如下圖所示：



六軸關節行手臂第 5、6 軸結構耦合示意圖

（圖表文字修改成英文）

另一個範例為四軸 SCARA 機構，若其垂直軸是利用滾珠螺杆將馬達轉動變為垂直運動，則在末端的機構也會產生相對應的轉動。為了補償因為結構耦合產生的連動效應，使用者可以透過群組參數:0x01，將耦合的資訊設定到運動控制核心模組，以計算所需的運動補償命令，相關參數說明如下：

相關群組參數:0x01 說明如下:

Sub index	說明
0	Mechanical couple master axis
1	Mechanical couple slave axis
2	Compensation source type
3	Reference master original position
4	Coupling ratio numerator
5	Coupling ratio denominator
6	Enable mechanical couple compensation

- 0x01, SubIndex = 0:Mechanical couple master axis
  指定結構耦合關係中的主動軸，以上述六軸關節型手臂的例子來看，第 5 軸為主動軸。

- 0x01, SubIndex = 1:Mechanical couple slave axis

  指定結構耦合關係中的從動軸，以上述六軸關節型手臂的例子來看，第 6 軸為從動軸。

- 0x01, SubIndex = 2:Compensation source type

  用戶可以指定從動軸的位置補償命令是依據主動軸的 actual position 或是 command position。

- 0x01, SubIndex = 3:Reference master original position

  用戶可以指定主動軸在甚麼位置作為從動軸位置命令補償的計算基準。

- 0x01, SubIndex = 4:Coupling ratio numerator(分子)
- 0x01, SubIndex = 5:Coupling ratio denominator(分母)

  上述兩個參數必須搭配設定，主要利用分子分母的形式來設定從動軸位置命令補償量計算所需之 Coupling ratio。分子，不可以為 0，負號代表反向補償；分母，必須大於 0。以上圖六軸關節型手臂來說，當第五軸(主動軸)機構端往正方向轉動 90 度時，會造成第 6 軸(從動軸)馬達端往負方向轉動 90 度，假設第 6 軸馬達減速比為 50，則第 6 軸機構端將轉動-1.8 度。為了補償因結構耦合造成的轉動，則第 6 軸機構端的位置命令需要加上 1.8 度(從動軸位置命令補償量)，對應到 Coupling ratio，則分母(SubIndex)設定為 50，分子(SubIndex 4)設定為 1，相關公式如下所示：

從動軸機構端位置命令補償量 =(主動軸機構端位置 - 主動軸機構端位置基準(SubIndex 2))*(SubIndex 4) / (SubIndex 5)

- 0x01, SubIndex = 6:Enable mechanical couple compensation

  啟動結構耦合補償之功能，開啟後，控制器將依據相關結構耦合資訊自動計算位置補償命令到從動軸。

上述的參數當系統進入 OPERATION 的狀態後，將無法更改。

1.4.5. 群組啟動與狀態

欲使某群組(Group)中所有軸向進行伺服啟動(Servo enable)，則可使用NMC_GroupEnable()達成。因各廠牌驅動器之啟動(Servo enable)時間不一，因此可撰寫等待時間待驅動器狀態切換至「NMC_AXIS_STATE_DISABLE」狀態。若需要查看各群組狀態(State of group)，可使用NMC_GroupGetState()來查看。

當所有軸向皆已啟動，則此群組的狀態會切換為「NMC_GROUP_STATE_STAND_STILL」；若群組中某一軸以上的狀態為 DISABLE，則群組的總體狀態會切換為「NMC_GROUP_STATE_DISABLE」；

若群組中有一軸以上出現錯誤，則群組的狀態會切換為「NMC_GROUP_STATE_ERROR」。若欲使某群組中所有軸向進行伺服關閉(Servo disable)，則可使用 NMC_GroupDisable()達成。

```
#include "NexMotion.h"
#include "NexMotionError.h"
#include <Windows.h>

int main()
{
    RTN_ERR ret       = 0;
    I32_T   devType   = NMC_DEVICE_TYPE_ETHERCAT;
    I32_T   devIndex  = 0;
    I32_T   groupIndex = 0;
    I32_T   devID     = 0;
    I32_T   state     = 0;

    ret = NMC_DeviceOpenUp( devType, devIndex, &devID );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }
    // Controller is start up successfully.

    ret = NMC_GroupEnable( devID, groupIndex );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    // To make sure state is transer to ENABLE
    Sleep( 3000 );  //3 secounds is a try value. Depends on servo drives.

    ret = NMC_GroupGetState( devID, groupIndex, &state );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    if( state != NMC_GROUP_STATE_STAND_STILL )
    {
        // Error handling...
    }

    // The group is enable successfully.
    // Do something...
    // ...


    ret = NMC_GroupDisable( devID, groupIndex );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    // To make sure state is transer to DISABLE
    Sleep( 1000 );  //1 second is a try value. Depends on servo drives.
```

```
    NMC_DeviceShutdown( devID );
    return 0;
}
```

若控制器使用多個軸或群組，欲使所有軸或群組所有軸向同時伺服啟動(Servo enable)，則可使用 NMC_DeviceEnableAll()。若需要查看群組的數量，可使用 NMC_DeviceGetGroupCount()來得知此控制器使用多少個群組。若欲使所有群組中的所有軸向一起 DISABLE，則可使用 NMC_DeviceDisableAll()即可關閉系統中所有軸向。

```
#include "NexMotion.h"
#include "NexMotionError.h"
#include <Windows.h>

int main()
{
    RTN_ERR ret       = 0;
    I32_T   devType   = NMC_DEVICE_TYPE_ETHERCAT;
    I32_T   devIndex  = 0;
    I32_T   groupIndex = 0;
    I32_T   devID     = 0;
    I32_T   groupCount = 0;
    I32_T   state     = 0;
    I32_T   i;

    ret = NMC_DeviceOpenUp( devType, devIndex, &devID );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    // Controller is start up successfully.

    ret = NMC_DeviceEnableAll( devID );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    // To make sure the state is transer to ENABLE
    Sleep( 3000 );  //3 secounds is a try value. Depends on servo drives.

    // Get group count in device
    ret = NMC_DeviceGetGroupCount( devID, &groupCount );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    for( i = 0; i < groupCount; i++ )
    {
        ret = NMC_GroupGetState( devID, groupIndex, &state );
```

```
        if( ret != ERR_NEXMOTION_SUCCESS )
        {
            // Error handling...
        }

        if( state != NMC_GROUP_STATE_STAND_STILL )
        {
            // Error handling...
        }
    }

    NMC_DeviceDisableAll( devID );

    // To make sure state is transer to DISABLE
    Sleep( 1000 );  //1 second is a try value. Depends on servo drives.

    NMC_DeviceShutdown( devID );

    return 0;
}
```

除群組狀態(State of group)外，若需查看更多群組的運動資訊 (Status of group)，可使用NMC_GroupGetStatus()達成。目前提供14種狀態資訊供參考，包含外部急停訊號、警報、正向硬體極限、負向硬體極限、正向軟體極限、負向軟體極限、ENABLE、錯誤狀態、命令停止、加速狀態、減速狀態、等速狀態、運動中狀態與停止狀態。

有兩個時機點需要使用NMC_GroupResetState()來清除群組狀態:

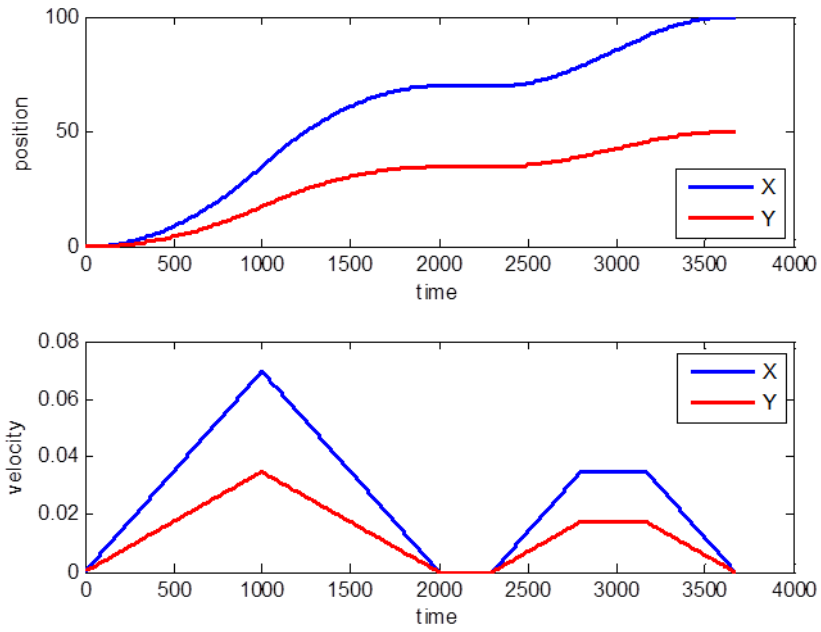
1. 於任何情況(或緊急情況)下當使用者呼叫NMC_GroupStop()使群組停止運動，則群組狀態(STATE)的變化為STOPPED。當群組在STOPPED狀態時，控制器將拒絕執行任何新的運動命令。當使用者確認狀況排除後可讓控制器接受新的運動命令時使用NMC_GroupResetState()來使群組狀態切換為STAND STILL。
2. 若群組有任一軸驅動器發生警報，若此警報可由控制器清除，則亦可使用此API將警報清除，此時群組狀態(STATE)的變化為ERROR STOP→DISABLE。

若群組有任一軸驅動器有警報，且此警報有開放許可權給控制器來清除，則可使用NMC_GroupResetDriveAlm()或NMC_GroupResetDriveAlmAll()將此軸向的警報清除。

1.4.6. 群組速度百分比設定

每個群組有設置一速度百分比設定，可透過NMC_GroupSetVelRatio()進行設定，該群組的任何運動最大速度皆會參考此速度百分比的數值，例如，若最大速度為100；速度百分比為70%，則運動的最大速度將以 100 * 70% = 70的上限進行速度規劃。如下圖為一群組具有X和Y兩軸執行

一點對點運動(PTP)，於時間為1000時設定速度百分比為0，設定後則開始減速為零；于時間為2300時設定速度百分比為50，設定後則開始加速，直至到達目標位置。



(擺放上圖之範例程式)

```
#include "NexMotion.h"
#include "NexMotionError.h"

int main()
{

}
```

若需得知目前群組的速度百分比的數值，則可使用NMC_GroupGetVelRatio()達成。

1.4.7. 群組點對點運動

與群組軸座標之點對點運動(PTP)相關的API如下:

- 軸坐標系:

  NMC_GroupPtpAcs()

  NMC_GroupPtpAcsAll()

- 卡式坐標系:

  NMC_GroupPtpCart()

NMC_GroupPtpCartAll()

上述API之差別在於可控制的軸數與目標位置(Target position)所使用的坐標系不同，但其運動行為是一致的，換言之無論使用那一種API，當執行點對點(PTP)運動時，控制器會以各軸(軸坐標系)之最短路徑進行運動規劃，並自動以所需運動時間最長的軸為基準降低其他軸之速度使之所有運動軸向會同一時間到達目標位置。各軸之運動速度會以各群組軸之軸參數進行運算，與PTP相關之速度參數如下:

Param. Num.	Sub. Index	資料型態	說明
0x30	0	I32_T	Absolute or relative programming
0x31	0	I32_T	Profile type
0x32	0	F64_T	Max. velocity (unit/sec)
0x33	0	F64_T	Acceleration (unit/sec^2)
0x34	0	F64_T	Deceleration (unit/sec^2)
0x35	0	F64_T	Jerk (unit/sec^3)

設定或讀取群組軸參數相關之API如下:

NMC_GroupAxSetParamI32()

NMC_GroupAxGetParamI32()

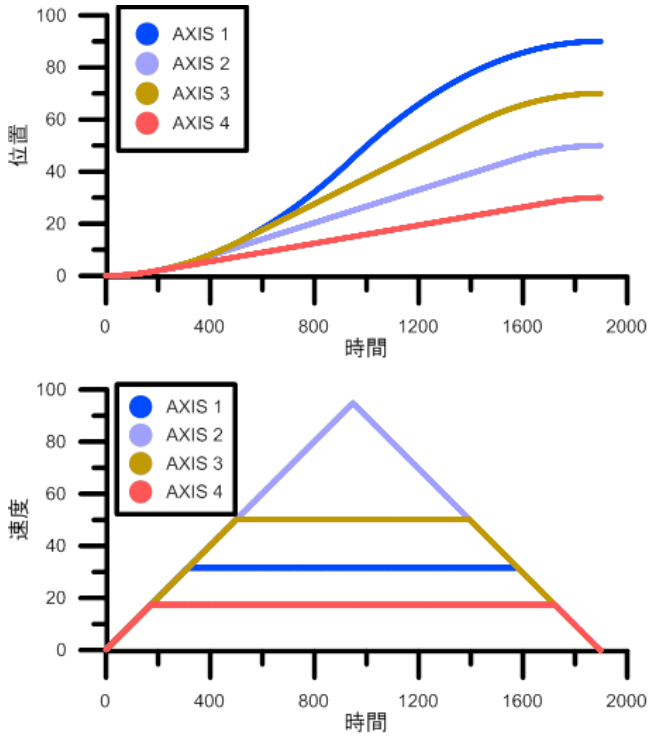
NMC_GroupAxSetParamF64()

NMC_GroupAxGetParamF64()

使用軸坐標系為(?)，若僅需要移動群組中某一軸向到指定的軸座標位置，則可使用NMC_GroupPtpAcs()達成。若需要移動一個以上的軸向到指定的軸座標位置，則可使用NMC_GroupPtpAcsAll()達成。需指定欲移動軸向的遮罩(Mask)與軸座標位置，

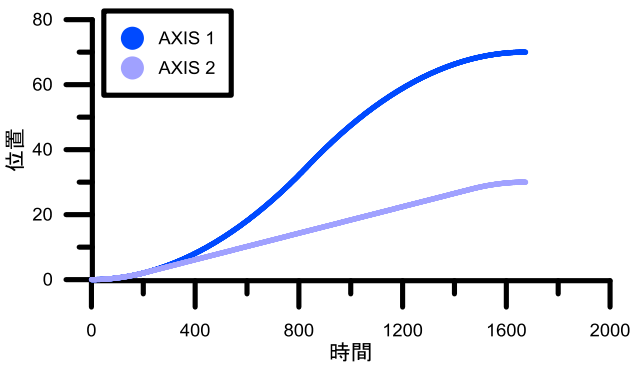
若使用空間坐標系為單位，需要延著移動某個卡氏空間坐標軸方向到指定的位置(於MCS或PCS空間中操作)，則可使用NMC_GroupPtpCart()達成。若需要移動一個以上的卡式坐標軸，則可使用NMC_GroupPtpCartAll()來達成，需指定卡氏空間坐標軸的遮罩與位置。

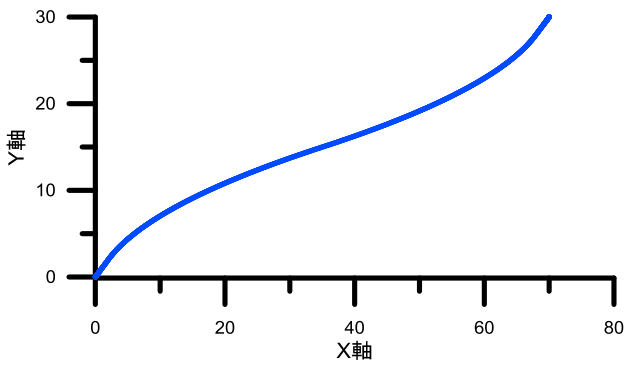
下圖與範例程式為一使用NMC_GroupPtpAcsAll()之範例，使第1軸移動至90的位置；第2軸移動至50的位置；第3軸移動至70的位置；第4軸移動至30的位置，呼叫完此API後，群組將需要移動的軸向進行插補。




```
#include "NexMotion.h"
#include "NexMotionError.h"

int main()
{
    RTN_ERR ret              = 0;
    I32_T   groupIndex        = 0;
    I32_T   groupAxesIdxMask = 15; //移動第1軸、第2軸、第3軸與第4軸
    I32_T   devID            = 0;
    Pos_T   acsPos           = { 0 };

    acsPos[0] = 90; //第1軸的目標位置
    acsPos[1] = 50; //第2軸的目標位置
    acsPos[2] = 70; //第3軸的目標位置
    acsPos[3] = 30; //第4軸的目標位置

    ret = NMC_GroupPtpAcsAll( devID, groupIndex, groupAxesIdxMask, &acsPos );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    return 0;
}
```

下圖與範例程式為一使用NMC_GroupPtpCartAll()之範例，使群組移動X方向至70的位置；Y方向至30的位置，呼叫完此API後，群組將需要移動的軸向進行插補。






```
#include "NexMotion.h"
#include "NexMotionError.h"

int main()
{
    RTN_ERR ret          = 0;
    I32_T   groupIndex   = 0;
    I32_T   cartAxesMask = 5; //移動的座標為X軸與Z軸
    I32_T   devID        = 0;
    Pos_T   targetPos    = { 0 };

    targetPos[0] = 10; //X軸的目標位置
    targetPos[2] = 20; //Z軸的目標位置

    ret = NMC_GroupPtpCartAll( devID, groupIndex, cartAxesMask, &targetPos );
    if( ret != ERR_NEXMOTION_SUCCESS )
    {
        // Error handling...
    }

    return 0;
}
```

1.4.8. 群組 Jog 運動

若需要將群組中某一軸向以最大速度移動(於ACS空間中操作),則可使用NMC_GroupJogAcs()達成。當指令下達成功後，該群組軸將以設定之最大速運轉。如要停止可呼叫NMC_GroupHalt()將或呼叫NMC_GroupHaltAll()將整個群組暫停。Jog的速度規劃依下列軸參數定義:

Param. Num.	Sub. Index	資料型態	說明
0x31	0	I32_T	Profile type
0x32	0	F64_T	Max. velocity (unit/sec)
0x33	0	F64_T	Acceleration (unit/sec^2)
0x34	0	F64_T	Deceleration (unit/sec^2)
0x35	0	F64_T	Jerk (unit/sec^3)

設定或讀取群組軸參數相關之API如下:

NMC_GroupAxSetParamI32()

NMC_GroupAxGetParamI32()

NMC_GroupAxSetParamF64()

NMC_GroupAxGetParamF64()

若移動過程中遇到硬體或軟體極限，則該軸會自動停止，且狀態(STATE)的變化為MOVING→STOPPING→STOPPED，若需要將狀態切換為STAND STILL，請使用NMC_GroupResetState()。

1.4.9. 群組停止運動

任何運動命令皆可透過下列指令將運動停止:

API	說明
NMC_GroupHalt() NMC_GroupStop()	停止單一群組
NMC_GroupHaltAll() NMC_GroupStopAll()	停止所有群組
NMC_DeviceHaltAll() NMC_DeviceStopAll()	停止所由軸與群組

Halt 與 Stop的行為比較:

	Halt	Stop
行為	將目前正在運行中的運動中止，若運動隊列(Motion buffer)中無其他指運動指令則群組狀態將回到 NMC_GROUP_STATE_STAND_STILL，反之若運動隊列(Motion buffer)中尚有其他的運動指令則佇列中之指令將會繼續被執行。	將目前正在運行中的運動中止，並將運動隊列(Motion buffer)中所有運動指令清空，群組狀態會切換至 NMC_GROUP_STATE_STOPPED，直到使用者呼叫 NMC_GroupResetState()才能執行新的運動指令
參數	0x31:Halt profile type 0x34:Halt deceleration (unit/sec2) 0x35:Halt jerk (unit/sec3) 0x36-0:Buffer mode	0x20 :Stop profile type 0x21 :Stop deceleration (unit/sec2) 0x22 :Stop jerk (unit/sec3)
是否可佇列?	指令可被放入運動隊列中(Motion buffer)。配合群組參數 0x36 sub 0(Buffer Mode)來使用，若使用 0 的數值(Aborting)，則會立刻停止運動；若使用 1 的數值(Buffered)，則會將此命令加入至排序中。	無

1.4.10.群組直線補間

若要執行卡式坐標系(Cartisian space)下的直線運動，可使用下列 API:

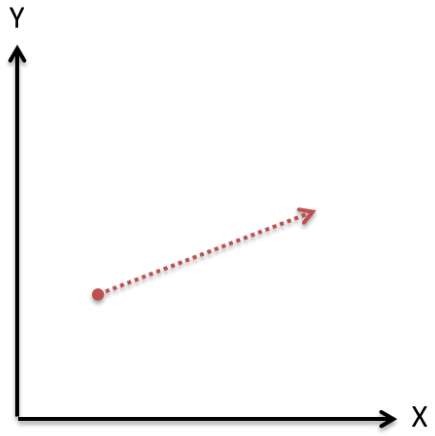
API	說明
NMC_GroupLineXY()	2D 直線補間(XY軸)
NMC_GroupLine()	3D 直線補間與3D姿態(Orientation)控制(最高六自由度)

直線補間運動功能可分為

1. 線性軸之直線補間
2. 線性軸之直線補間 + 姿態軸(ABC 軸)姿態控制

1.4.10.1. 多維度直線補間

一般二維的機構(如XY Table)或機構超過二維可使用NMC_GroupLineXY()下二維直線補間運動，如下圖所示:



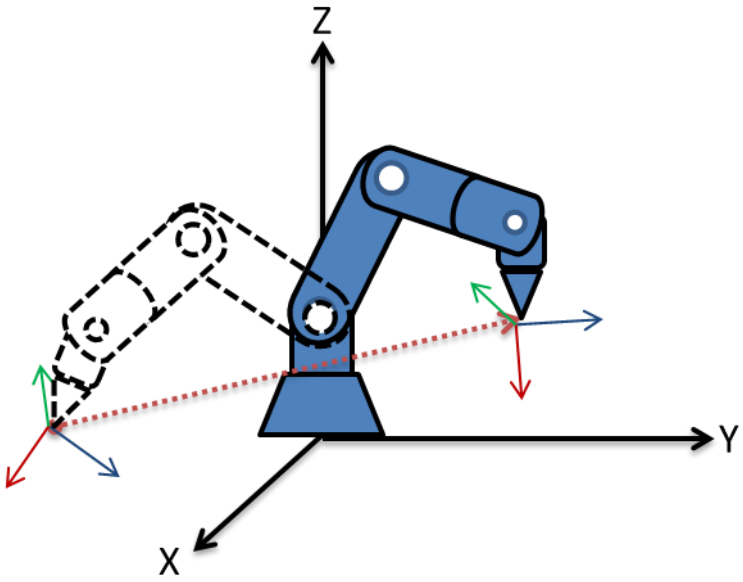
若為2軸以上的線性機構要進行補間運動則可採用NMC_GroupLine()下達，群組直線補間運動使用下列群組參數進行規劃:

定義	常數值	說明
目標位置型式	0x30	0: 絕對位置; 1: 增量位置
速度規劃方式	0x31	0: T curve; 1: S curve
最大速度	0x32	速度規劃的最大速度(unit/sec)
加速度	0x33	速度規劃的加速度(unit/sec^2)
減速度	0x34	速度規劃的減速度(unit/sec^2)

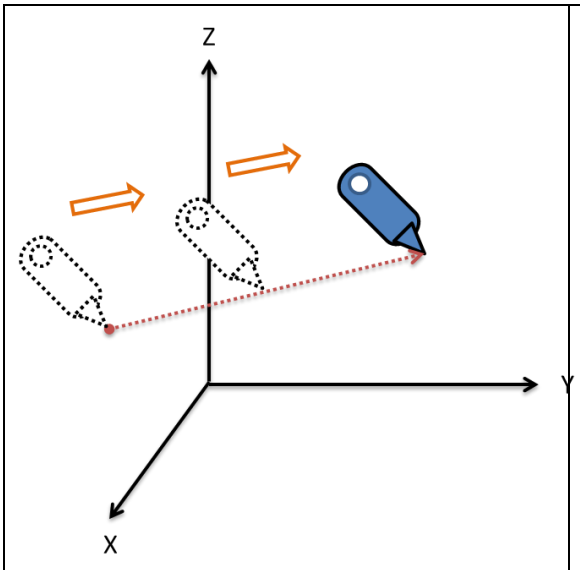
急衝量	0x35	速度規劃的急衝量(unit/sec^3)(僅 S curve 有效)

1.4.10.2. 3D 直線補間與姿態控制

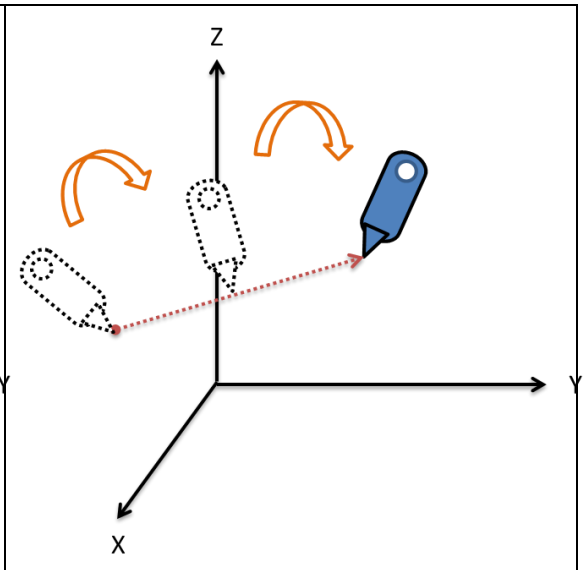
若設定的群組機構形式為機器人(Robot)，其ABC值則定義為姿態角(Oreintation angle)，下圖為一六軸機械手臂執行直線補間伴隨姿態控制



下圖說明有無姿態控制之示意圖:



直線補間(無姿態控制)



直線補間 + 姿態控制

群組直線補間與姿態控制使用下列群組參數進行規劃:

- 線性軸直線補間速度參數:

定義	常數值	說明
目標位置型式	0x30	0: 絕對位置; 1: 增量位置
速度規劃方式	0x31	0: T curve; 1: S curve
最大速度	0x32	速度規劃的最大速度(unit/sec)
加速度	0x33	速度規劃的加速度(unit/sec^2)
減速度	0x34	速度規劃的減速度(unit/sec^2)
急衝量	0x35	速度規劃的急衝量(unit/sec^3)(僅 S curve 有效)

- 姿態控制速度參數:

定義	常數值	說明
姿態角獨立插補	0x3A	0: 獨立插補; 1: 非獨立插補
姿態角最大速度	0x3B	姿態角的最大速度(unit/sec)
姿態角加速度	0x3C	姿態角的加速度(unit/sec^2)
姿態角減速度	0x3D	姿態角的減速度(unit/sec^2)
姿態角急衝量	0x3E	姿態角的急衝量(unit/sec^3)(僅 S curve 有效)

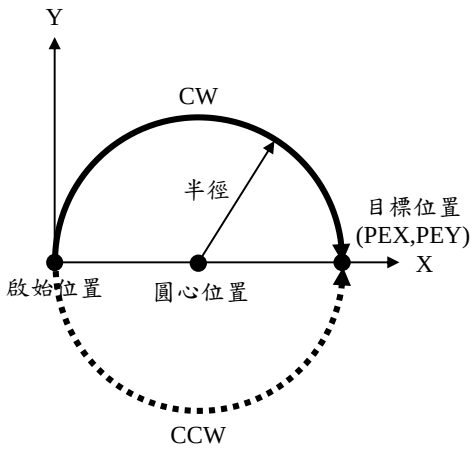
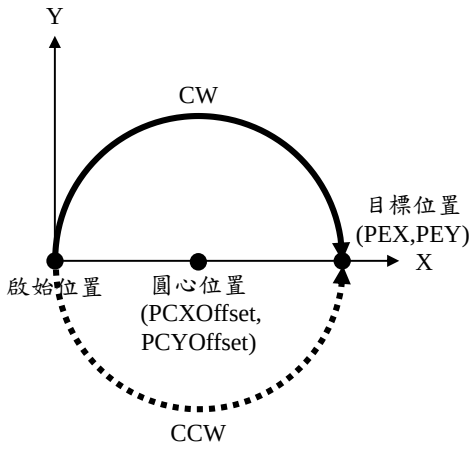
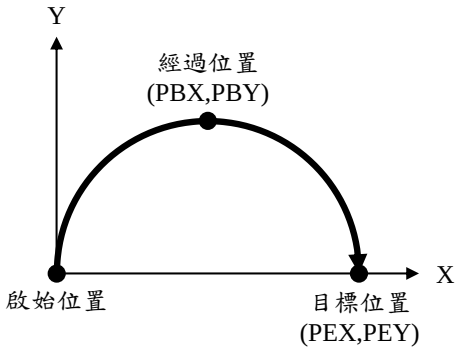
0x3A 參數決定姿態角運動是否使用獨立之插補器運算:

- 當 0x3A 設定為 0(獨立插補)，表示姿態控制採用獨立插補器進行速度規劃，其姿態角的速度規劃參考 0x3B~0x3E;
- 當 0x3A 設定為 1(非獨立差補)，表示姿態控角之變化量等比例跟隨直線運動。

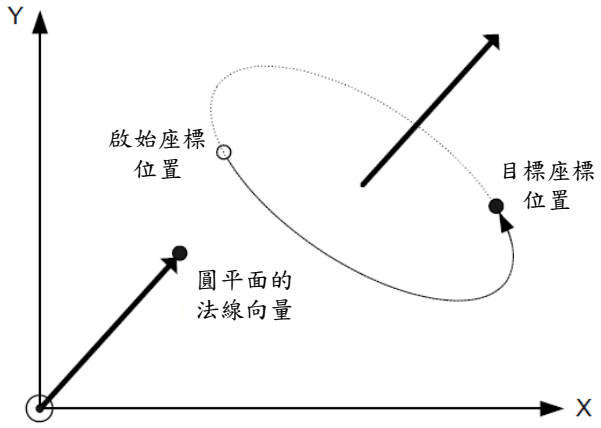
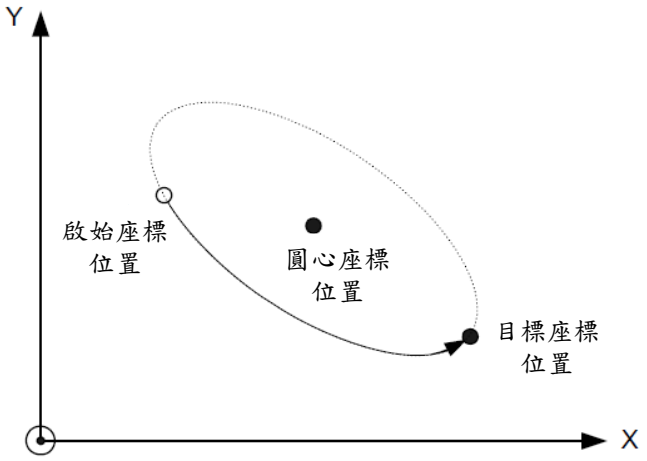
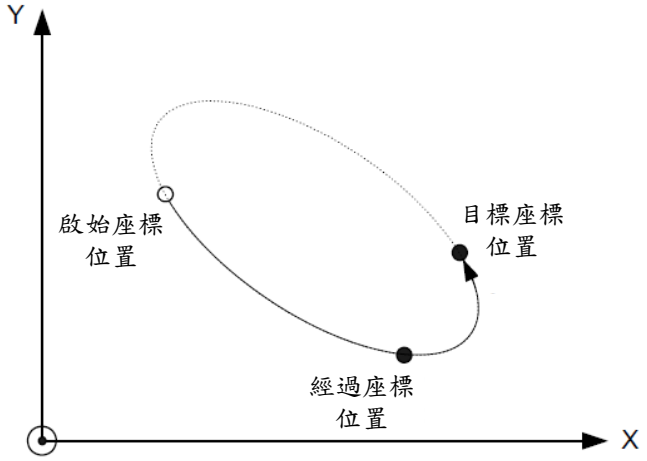
1.4.11. 群組圓弧補間

執行卡式坐標系(Cartisian space)下的圓弧運動，本控制器提供下列方式定義圓弧運動:

- 2D 圓弧

定義方式	相關 API	示意圖
終點位置 半徑 方向	NMC_GroupCirc2R()	
終點位置 圓心位置 方向	NMC_GroupCirc2C()	
終點位置 經過點位置	NMC_GroupCirc2B()	

- 3D 圓弧(含姿態)

定義方式	相關 API	示意圖
終點位置 姿態 半徑 方向 平面法向量	NMC_GroupCircR()	
終點位置 姿態 圓心位置 方向	NMC_GroupCircC()	
終點位置 姿態 經過點位置	NMC_GroupCircB()	

其相關速度規劃參數如下:

- 圓弧切線速度參數:

定義	常數值	說明
目標位置型式	0x30	0: 絕對位置; 1: 增量位置
速度規劃方式	0x31	0: T curve; 1: S curve

最大速度	0x32	圓弧切線最大速度(unit/sec)
加速度	0x33	圓弧切線加速度(unit/sec^2)
減速度	0x34	圓弧切線減速度(unit/sec^2)
急衝量	0x35	圓弧切線急衝量(unit/sec^3)(僅 S curve 有效)

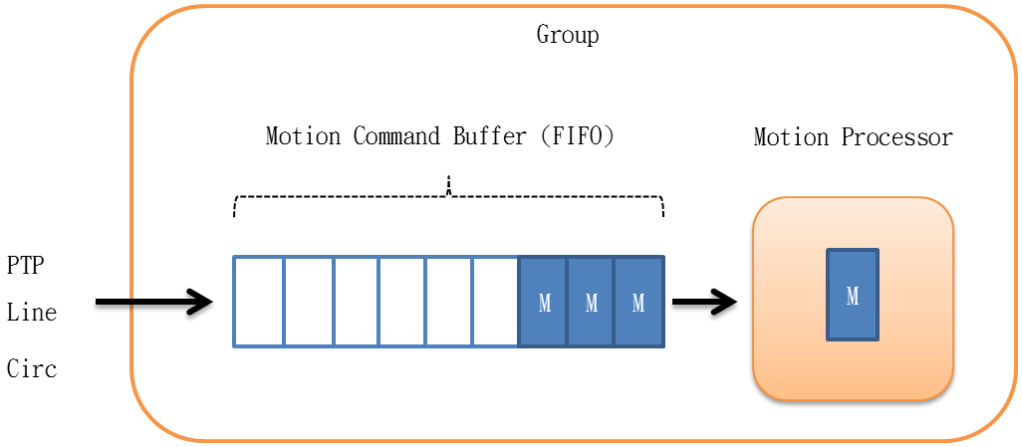
- 姿態控制速度參數:

定義	常數值	說明
姿態角獨立插補	0x3A	0: 獨立插補; 1: 非獨立插補
姿態角最大速度	0x3B	姿態角的最大速度(unit/sec)
姿態角加速度	0x3C	姿態角的加速度(unit/sec^2)
姿態角減速度	0x3D	姿態角的減速度(unit/sec^2)
姿態角急衝量	0x3E	姿態角的急衝量(unit/sec^3)(僅 S curve 有效)

0x3A 參數決定姿態角運動是否使用獨立之插補器運算:

- 當 0x3A 設定為 0(獨立插補)，表示姿態控制採用獨立插補器進行速度規劃，其姿態角的速度規劃參考 0x3B~0x3E;
- 當 0x3A 設定為 1(非獨立差補)，表示姿態控角之變化量等比例跟隨直線運動。

每個群組設置有命令佇列(Motion command buffer)，可以連續將運動命令設定存入佇列中，此佇列為先進先出(FIFO)，控制器會依序將運動隊列中之運動命令依序執行，可透過NMC_GroupGetMotionBuffSpace()讀取目前還有多少佇列空間。預設的群組佇列大小為 32。



可被暫存的運動命令有下列:

1. 點對點運動(PTP)
2. 直線補間運動(Line)
3. 圓弧補間運動(Circ)

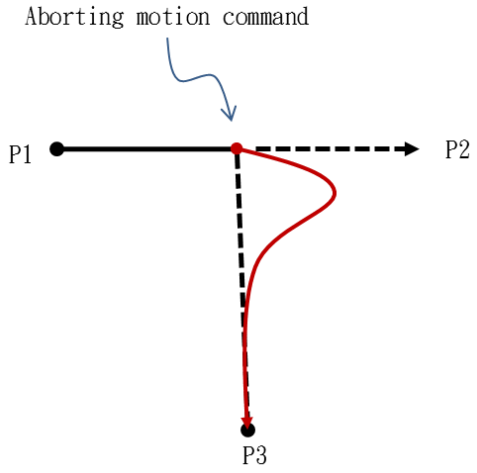
當使用者呼叫上述運動指令時控制器會參考群組參數 0x36:0 (Buffer mode)來判斷如何處裡新的運動命令和當前(或前一個)運動命令的連接方式，下表為 Buffer Mode 的參數意義:

群組參數 0x36:0 (Buffer Mode)

數值	Buffer Mode	說明
0	Aborting	中斷目前運動並立即執行此運動命令。
1	Buffered	暫存此運動命令待前一運動命令結束後再執行
2	Blending	混合此運動命令與前一個運動命令，使軌跡與速度平滑化， 平滑程度透過群組參數 0x36:1~3 設定之。

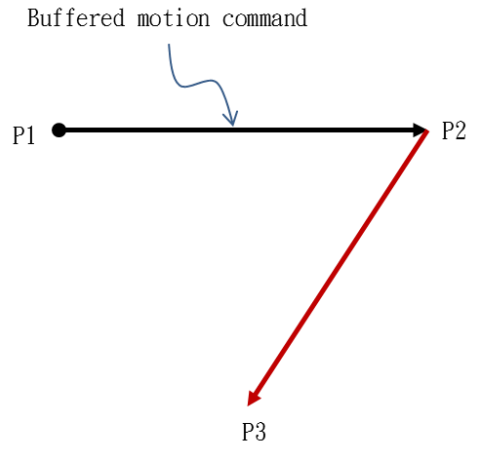
1.4.12.1. Buffer Mode: Aborting

系統預設為參數為 Aborting 模式，亦即當運動命令下達後會中斷目前運動立刻執行，若佇列中有其他運動命令也會一併清空。下圖為 Aborting 之示意圖:



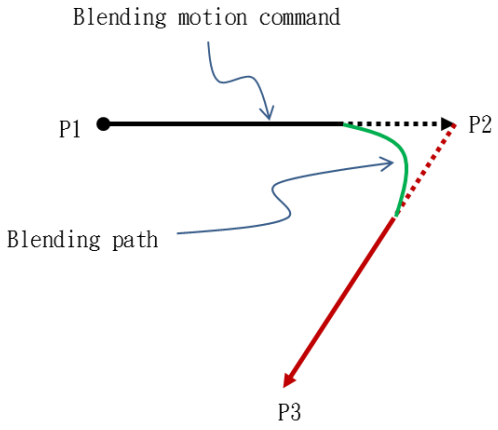
1.4.12.2. Buffer Mode: Buffered

若設定為 Buffered 模式，運動命令將被暫存至佇列中待前一命令執行完畢後再被執行，前一個運動指令會減速至零，此模式通常被應用於希望軌跡規劃能確實通過目標點位元，下圖為 Buffered 命令之示意圖:



1.4.12.3. Buffer Mode: Blending

若設定路徑連接方式為混合模式(Blending)，可將接續的兩個路徑進行位置與速度的混合，透過此功能可提升加工路徑軌跡之平滑性，提升運動速率，縮短加工時間，但路徑不通過設定點位元。下圖為 Blending 命令之示意圖:



路徑混合模式(Buffer Mode)包括: BlendingLow, BlendingPrevious, BlendingNext, BlendingHigh 和 Blending，而路徑混合功能可另外設定混合方式 Transition Mode(群組參數 0x36 Sub 1)和 Transition Paratemer(群組參數 0x36 Sub 2)來微調其混合時的行為，參數意義如下表:

Transition Mode(參數 0x36 Sub 1)：

數值	Transition Mode	說明
0	None	速度連續運動的速度由 Buffer Mode 決定，如 BlendingLow、BlendingHigh、BlendingPrevious、BlendingNext。
1	Start Velocity	速度連續運動的速度由當前運動的速度百分比決定
2	Constant Velocity	速度連續運動的速度由下一個運動的速度百分比決定
3	Corner Distance	速度連續運動的速度由轉角的距離決定。
4	Corner Deviation	速度連續運動的速度由轉角的誤差決定。

Transition paratemer(參數 0x36 Sub 2)：

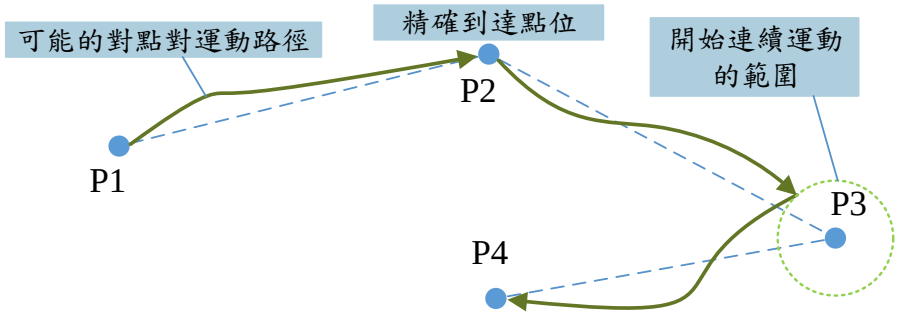
依 Buffer Mode 與 Transition Mode 設定的參數值不同，此參數可設定為速度百分比、轉角的距離或誤差。

範例:點對點運動(PTP)之間的連續運動

下表為一連續運動指令:

順序	運動指令	連接方式 Buffer Mode	混合方式 Blending Mode	混合參數 Blending Parameter
1	PTP(P2)	忽略 (目前無運動)	忽略 (目前無運動)	忽略 (目前無運動)
2	PTP(P3)	Buffered	忽略	忽略

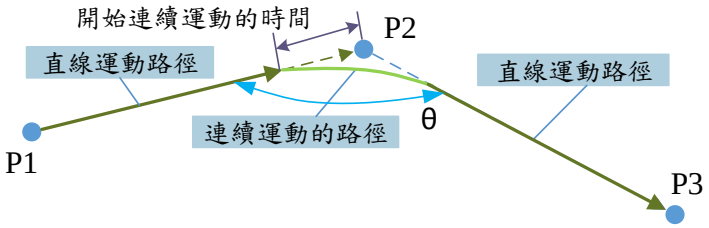
3	PTP(P4)	Blending	Corner Distance	Distance Value



當 PTP(P3)無使用連續運動功能，則前一運動指令會精確地到達 P2 點位;當 PTP(P4)使用連續運動功能，當插補點位接近 P3 點位時(到達指定的 Corner Distance 範圍)，會立刻下達新的位置命令(P4)，且末端點的速度不會減速為零。

範例:兩個直線補間(Line)的連續運動：

順序	運動指令	連接方式 Buffer Mode	混合方式 Blending Mode	混合參數 Blending Parameter
1	Line(P2)	忽略 (目前無運動)	忽略 (目前無運動)	忽略 (目前無運動)
2	Line(P3)	Blending	Corner Distance	Distance value

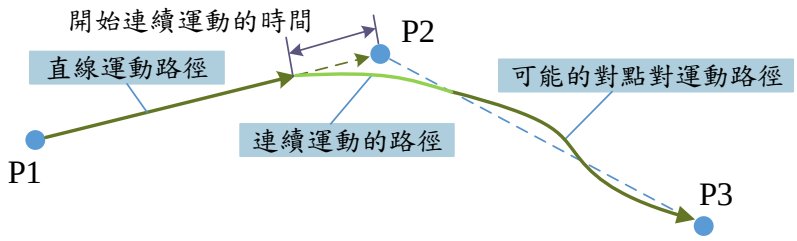


若 Line(P3) 使用路徑混合模式(Buffer mode = Blendig)，當插補點位到達指定的 Corner Distance 時，控制器會將 Line(P2) 的路徑與 Line(P3)之路徑進行速度與路徑平滑化，而末端點的速度不會減速為零，路徑規劃將不會通過 P2 點位。

範例:直線補間(Line)和點對點運動(PTP)的連續運動：如下圖，

順序	運動指令	連接方式 Buffer Mode	混合方式 Blending Mode	混合參數 Blending Parameter
1	Line(P2)	忽略	忽略	忽略

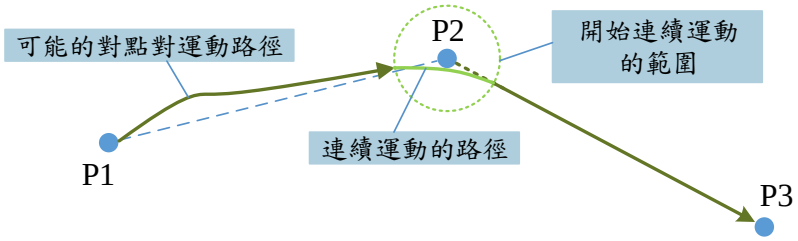
		(目前無運動)	(目前無運動)	(目前無運動)
2	PTP(P3)	Blending	Corner Distance	Distance value



若 PTP(P3) 使用路徑混合模式(Buffer mode = Blendig)，當插補點位到達指定的 Corner Distance 時，控制器會將 Line(P2) 的路徑與 PTP(P3)之路徑進行速度與路徑平滑化，而末端點的速度不會減速為零，路徑規劃將不會通過 P2 點位。

範例:點對點運動(PTP)與直線補間(Line)的連續運動：

順序	運動指令	連接方式 Buffer Mode	混合方式 Blending Mode	混合參數 Blending Parameter
1	PTP(P2)	忽略 (目前無運動)	忽略 (目前無運動)	忽略 (目前無運動)
2	Line(P3)	Blending	Corner Distance	Distance value



若 Line(P3) 使用路徑混合模式(Buffer mode = Blendig)，當 PTP 的路徑進入到達指定的 Corner Distance 範圍內，控制器會將 Line(P3) 的路徑與 PTP(P2)之路徑進行速度與路徑平滑化，路徑規劃將不會通過 P2 點位元且末端點的速度不會減速為零。

不同的運動命令的組合有其使用規則，請參考下表(○代表支持；✕代表不支持)。

直線補間-直線補間(或 直線補間-圓弧補間 / 圓弧補間-直線補間 / 圓弧補間-圓弧補間)

Buffer Mode / Transition Mode	Aborting	Buffered	Blending Low	Blending Previous	Blending Next	Blending High	Blending
None	○	○	○	○	○	○	○
Start Velocity	○	✕	○	○	✕	○	✕
Constant Velocity	○	✕	○	✕	○	○	✕

Corner Distance	×	×	×	×	×	×	○
Corner Deviation	×	×	×	×	×	×	○

直線補間-點對點運動(或 圓弧補間-點對點運動)

Buffer Mode / Transition Mode	Aborting	Buffered	Blending Low	Blending Previous	Blending Next	Blending High	Blending
None	○	○	○	○	○	○	○
Start Velocity	×	×	×	×	○	×	×
Constant Velocity	×	×	×	×	×	×	×
Corner Distance	×	×	○	×	×	×	×
Corner Deviation	×	×	×	×	×	×	×

點對點運動-直線補間(或 點對點運動-圓弧補間)

Buffer Mode / Transition Mode	Aborting	Buffered	Blending Low	Blending Previous	Blending Next	Blending High	Blending
None	○	○	○	×	×	×	×
Start Velocity	×	×	×	×	×	×	×
Constant Velocity	×	×	×	×	×	×	×
Corner Distance	×	×	○	×	×	×	×
Corner Deviation	×	×	×	×	×	×	×

點對點運動-點對點運動

Buffer Mode / Transition Mode	Aborting	Buffered	Blending Low	Blending Previous	Blending Next	Blending High	Blending
None	○	○	○	×	×	×	×
Start Velocity	×	×	×	×	×	×	×
Constant Velocity	×	×	×	×	×	×	×
Corner Distance	×	×	○	×	×	×	×
Corner Deviation	×	×	×	×	×	×	×

Halt()/Stop() command 的作用?

Motion state/Motion satus 的變化?

簡易版(所有)

Buffer Mode / Transition Mode	Aborting	Buffered	Blending Low	Blending Previous	Blending Next	Blending High	Blending
None	○	○	○	○	○	○	○

Start Velocity	×	×	○	○	×	○	×
Constant Velocity	×	×	○	×	○	○	×
Corner Distance	×	×	×	×	×	×	○
Corner Deviation	×	×	×	×	×	×	○

1.4.13. 群組歸零運動

由於群組歸零運動和單軸歸零運動功能相同，相關的使用行為與使用時機，請參考單軸歸零運動章節中的說明。

與群組歸零運動相關的函式，說明如下：

- NMC_GroupSetHomePos():

  在此函式中，使用者必須輸入在目前各軸的位置下，所欲設定的各軸之座標位置(ACS)。

- NMC_GroupAxesHomeDrive():

在此函式中，使用者可以指定要進行單軸歸零運動的群組軸，成功呼叫後，指定的群組軸將依據單軸參數中與歸零運動相關的參數，進行各自的單軸歸零程式。只要有一軸仍在進行歸零程式時，群組的狀態會停留在HOMING。在歸零程式中，若有任一軸發生錯誤的狀況，群組狀態將切換到ERROR。

2. 控制器參數

2.1. 系統參數

Param. Num.	Sub. Index	資料型態	說明	設定範圍	備註
0x00	0	I32_T	Motion cycle time (us).	1000~4000 microsecond	(*1)(*3)
0x01	0	I32_T	Number of activated axis	0~64	(*1)
0x02	0	I32_T	Number of activated group	0~64	(*1)

(*1): 系統啟動後參數生效，啟動中無法修改該參數(Return error)

(*2): Enable 該功能時參數生效, 其他時間修改不會立刻變更

(*3): 可設定範圍根據控制器的規格而不同

(*4): Read only

2.2. 單軸參數

Param. Num.	Sub. Index	資料型態	說明	設定範圍	備註
0x00	0	I32_T	Mechanical pitch (unit/rev)	1 ~ 2147483647	(*1)
0x01	0	I32_T	Mechanical revolution	1 ~ 2147483647	(*1)
0x02	0	I32_T	Motor revolution	1 ~ 2147483647	(*1)
0x03	0	I32_T	Encoder resolution (pulse/rev)	1 ~ 2147483647	(*1)
0x04	0	I32_T	Encoder direction	0:Not inverse, 1:Inverse	(*1)
0x05	0	I32_T	Encoder type	0:Incremental, 1:Absolute	(*1)
0x06	0	I32_T	Enable external encoder	0:Disable, 1:Enable	(*1)
0x06	1	F64_T	External Encoder ratio	(0+) ~ 2147483647	(*1)
0x07	0	I32_T	Cancel synchronized actual position to command position when servo enable	0:Not Cancel, 1:Cancel	(*1)
0x08	0	F64_T	Position offset	-2147483648 ~ 2147483647	(*1)
0x09	0	I32_T	Digital input configuration	該值以 bit 方式設定, 各 bit 設定意義如下: Bit 0: Set 1 to ignore PEL signal Bit 1: Set 1 to ignore MEL signal Bit 2~15: Reserved, set 0 Bit 16: Set 1 to inverse PEL signal Bit 17: Set 1 to inverse MEL signal Bit 18~31: Reserved, set 0	(*1)
0x0A	0	I32_T	Motor ID assignment	0~63, -1: No assignment	(*1)
0x10	0	F64_T	Positive software limit (user unit)	-2147483648 to 2147483647	(*2)
	1	I32_T	Enable positive software limit	0:Disable, 1:Enable	
	2	F64_T	Negative software limit (user unit)	-2147483648 ~ 2147483647	(*2)
	3	I32_T	Enable negative software limit	0:Disable, 1:Enable	
0x11	0	F64_T	Maximum velocity limit (unit/sec)	1 ~ 2147483647	(*2)
	1	I32_T	Enable max. velocity limit	0:Disable, 1:Enable	
0x12	0	F64_T	Maximum acceleration limit (unit/sec^2)	1 ~ 2147483647	(*2)
	1	I32_T	Enable max. acceleration limit	0:Disable, 1:Enable	
0x20	0	I32_T	Profile type for AxisStop command	0: T-Curve, 1:S-Curve	
0x21	0	F64_T	Deceleration for AxisStop command (unit/sec^2)	1 ~ 2147483647	
0x22	0	F64_T	Jerk for AxisStop command (unit/sec^3)	1 ~ 2147483647	
0x28	0	F64_T	Base velocity (unit/sec)	0 ~ 2147483647	
0x30	0	I32_T	Absolute or relative programming	0: Absolute, 1:Relative	
0x31	0	I32_T	Profile type	0: T-Curve, 1:S-Curve	

0x32	0	F64_T	Max. velocity (unit/sec)	1 ~ 2147483647	
0x33	0	F64_T	Acceleration (unit/sec^2)	1 ~ 2147483647	
0x34	0	F64_T	Deceleration (unit/sec^2)	1 ~ 2147483647	
0x35	0	F64_T	Jerk (unit/sec^3)	1 ~ 2147483647	
0x36	0	I32_T	Buffer mode	0:Aborting 1:Buffered 2:BlendingLow 3:BlendingPrevious 4:BlendingNext 5:BlendingHigh	
0x80	0	I32_T	Number of home parameters	5	(*3)(*4)
	1	I32_T	EtherCAT CiA HOME method	Refer to drive user manual	(*5)
	2	F64_T	EtherCAT CiA HOME speed search switch	Refer to drive user manual	(*5)
	3	F64_T	EtherCAT CiA HOME speed search zero	Refer to drive user manual	(*5)
	4	F64_T	EtherCAT CiA HOME acceleration	Refer to drive user manual	(*5)
	5	F64_T	EtherCAT CiA HOME offset	Refer to drive user manual	(*5)

(*1): 系統啟動後參數生效，啟動中無法修改該參數(Return error)

(*2): Enable 該功能時參數生效, 其他時間修改不會立刻變更

(*3): 可設定範圍根據控制器的規格而不同

(*4): Read only

(*5): 可設定範圍根據受控裝置的規格而不同

2.3. 群組參數

Num.	Sub	Type	Description	Value Description	Common
0x00	0	I32_T	Kinematics type	0:Linear (2~8 Axes) 1:Articulated Robot (AR6) 2:Delta 3:SCARA	(*1)
	1	F64_T	Kinematics parameter 1~48	參考機構運動學設定章節	(*1)
0x01	0	I32_T	Mechanical couple master axis	0~7 group axis	(*1)
	1	I32_T	Mechanical couple slave axis	0~7 group axis	(*1)
	2	I32_T	Compensation source type	0:Actual position, 1:Command position	(*1)
	3	F64_T	Reference master original position	unit, Value -2147483648 to 2147483647	(*1)
	4	F64_T	Coupling ratio numerator	Value != 0 (Negative value represents the opposite direction compensation)	(*1)
	5	F64_T	Coupling ratio denominator	Value > 0	(*1)
	6	I32_T	Enable mechanical couple compensation	0:Disable, 1:Enable	(*1)
0x15	0	F64_T	Max. velocity reference. (PCS,TCP)	unit/sec, value = 1 ~ 2147483647	
0x16	0	F64_T	Safety velocity limit. (PCS,TCP)	unit/sec, value = 1 ~ 2147483647	(*1)
0x20	0	I32_T	STOP profile type	0: T-Curve, 1:S-Curve	
0x21	0	F64_T	STOP deceleration	unit/sec^2,value = 1 ~ 2147483647	
0x22	0	F64_T	STOP jerk	unit/sec^3,value = 1 ~ 2147483647	
0x23	0	I32_T	Halt buffer mode	0:Aborting, 1:Buffered	
0x30	0	I32_T	Cart. relative positioning	0: Absolute positioning, 1:Relative Positioning	
0x31	0	I32_T	Cart. profile type	0: T-Curve, 1:S-Curve	
0x32	0	F64_T	Cart. max. velocity	unit/sec, value = 1 ~ 2147483647	
0x33	0	F64_T	Cart. acceleration	unit/sec^2,value = 1 ~ 2147483647	
0x34	0	F64_T	Cart. deceleration	unit/sec^2,value = 1 ~ 2147483647	
0x35	0	F64_T	Cart. jerk	unit/sec^3,value = 1 ~ 2147483647	
0x36	0	I32_T	Buffer mode	0:Aborting, 1:Buffered, 2:Blending	
	1	I32_T	Blending mode	0: Corner distance 1: MaxCorner deviation	
	2	F64_T	Blending corner distance	pos unit, Value 0 to 2147483647	
	3	F64_T	Blending corner deviation	pos unit, Value 0 to 2147483647	
0x3A	0	I32_T	Orientation motion independent	0: Not independent 1:Independent	
0x3B	0	F64_T	Max. orientation velocity	unit/sec, value = 1 ~ 2147483647	
0x3C	0	F64_T	Orientation acceleration	unit/sec^2,value = 1 ~ 2147483647	
0x3D	0	F64_T	Orientation deceleration	unit/sec^2,value = 1 ~ 2147483647	
0x3E	0	F64_T	Orientation jerk	unit/sec^3,value = 1 ~ 2147483647	

0x40	0	I32_T	Tool index selection for motion target	value = -1 :No tool assignment (Flange) value = 0~15: tool 0~15	
	1	I32_T	Tool index selection for read position	value = -2 : Parameter disable value = -1 :No tool assignment (Flange) value = 0~15: tool 0~15	
0x48	0	I32_T	Base index selection for motion target	value = -1 :No base assignment (MCS) value = 0~31: base 0~31	
	1		Base index selection for read position	value = -2 : Parameter disable value = -1 :No base assignment value = 0~31: base 0~31	
0x80~8F	0	F64_T	Offset along flange x-axis	unit, Value -2147483648 to 2147483647	
	1	F64_T	Offset along flange y-axis	unit, Value -2147483648 to 2147483647	
	2	F64_T	Offset along flange z-axis	unit, Value -2147483648 to 2147483647	
	3	F64_T	Rotation angle about flange z-axis	degree	
	4	F64_T	Rotation angle about flange y-axis	degree	
	5	F64_T	Rotation angle about flange x-axis	degree	
0xC0~DF	0	F64_T	Offset along reference base x-axis	unit, Value -2147483648 to 2147483647	
	1	F64_T	Offset along reference base y-axis	unit, Value -2147483648 to 2147483647	
	2	F64_T	Offset along reference base z-axis	unit, Value -2147483648 to 2147483647	
	3	F64_T	Rotation angle about reference base z-axis	degree	
	4	F64_T	Rotation angle about reference base y-axis	degree	
	5	F64_T	Rotation angle about reference PCS x-axis	degree	
	6	I32_T	Reference base index	value = -1: reference to MCS value = 0~31:reference to base 0~31	

(*1): 系統啟動後參數生效，啟動中無法修改該參數

(*2): Enable 該功能時參數生效, 其他時間修改不會立刻變更

3. C/C++函式庫

3.1. APIs 總覽

下表列出 NexMotion 函式庫 API 的列表，所有 API 定義於 NexMotion.h 標頭檔之中。

版本及錯誤資訊讀取相關函式

函數名稱	說明
NMC_GetLibVersion	回傳函式庫(Library)的版本編號
NMC_GetLibVersionString	回傳函式庫(Library)的版本編號 c 字串
NMC_GetErrorDescription	回傳錯誤碼描述 c 字串

控制器啟動相關函式

函數名稱	說明
NMC_DeviceOpenUp	開啟控制器 (阻塞式呼叫)
NMC_DeviceShutdown	關閉控制器 (阻塞式呼叫)
NMC_DeviceOpenUpRequest	送出開啟控制器請求(非阻塞式呼叫)
NMC_DeviceWaitOpenUpRequest	等待控制器啟動程序(阻塞式呼叫)
NMC_DeviceShutdownRequest	送出關閉控制器請求(非阻塞式呼叫)
NMC_DeviceWaitShutdownReque	等待控制器關閉程序(阻塞式呼叫)

進階控制器啟動相關函式

函數名稱	說明
NMC_DeviceCreate	取得控制器識別碼
NMC_DeviceDelete	移除控制器識別碼
NMC_DeviceLoadIniConfig	載入控制器之設定參數
NMC_DeviceResetConfig	清除控制器之設定參數
NMC_DeviceStart	啟動控制器(阻塞式呼叫)
NMC_DeviceStop	停止控制器(阻塞式呼叫)
NMC_DeviceStartRequest	送出啟動控制器的請求(非阻塞式呼叫)
NMC_DeviceStopRequest	送出停止控制器的請求(非阻塞式呼叫)
NMC_DeviceGetState	讀取控制器的狀態

看門狗相關函式

函數名稱	說明
NMC_DeviceWatchdogTimerEnable	啟動看門狗計時器
NMC_DeviceWatchdogTimerDisable	停止看門狗計時器

NMC_DeviceShutdownRequest

NMC_DeviceWatchdogTimerReset	清除看門狗計數器

系統參數設定相關函式

函數名稱	說明
NMC_DeviceSetParam	設定控制器之系統參數
NMC_DeviceGetParam	讀取控制器之系統參數
NMC_SetIniPath	設定 INI 檔案所在位置

I/O 控制相關函式

函數名稱	說明
NMC_GetInputMemorySize	讀取數位輸入(Input)映射記憶體之大小
NMC_GetOutputMemorySize	讀取數位輸出(Output)映射記憶體之大小
NMC_ReadInputMemory	讀取數位輸入(Input)映射記憶體
NMC_ReadOutputMemory	讀取數位輸出(Output)映射記憶體
NMC_WriteOutputMemory	寫入數位輸出(Output)映射記憶體
NMC_ReadInputBit	讀取數位輸入(Input)映射記憶體 bit 值
NMC_ReadInputI8	讀取數位輸入(Input)映射記憶體 byte 值
NMC_ReadInputI16	讀取數位輸入(Input)映射記憶體 word 值
NMC_ReadInputI32	讀取數位輸入(Input)映射記憶體 double-word 值
NMC_ReadOutputBit	讀取數位輸出(Output)映射記憶體 bit 值
NMC_ReadOutputI8	讀取數位輸出(Output)映射記憶體 byte 值
NMC_ReadOutputI16	讀取數位輸出(Output)映射記憶體 word 值
NMC_ReadOutputI32	讀取數位輸出(Output)映射記憶體 double-word 值
NMC_WriteOutputBit	寫入數位輸出(Output)映射記憶體 bit 值
NMC_WriteOutputI8	寫入數位輸出(Output)映射記憶體 byte 值
NMC_WriteOutputI16	寫入數位輸出(Output)映射記憶體 word 值
NMC_WriteOutputI32	寫入數位輸出(Output)映射記憶體 double-word 值

軸或群組數量資訊

函數名稱	說明
NMC_DeviceGetAxisCount	讀取軸數量資訊
NMC_DeviceGetGroupCount	讀取群組數量資訊
NMC_DeviceGetGroupAxisCount	讀取群組中軸數量資訊

軸或群組名稱資訊

函數名稱	說明
NMC_AxisGetDescription	讀取指定軸的名稱描述資訊

NMC_GroupGetDescription	讀取指定群組的名稱描述資訊

所有軸與群組啟用/禁用函式

函數名稱	說明
NMC_DeviceEnableAll	啟用系統中所有軸與群組
NMC_DeviceDisableAll	停用系統中所有軸與群組

所有軸與群組運動中止函式

函數名稱	說明
NMC_DeviceHaltAll	系統中所有軸與群組運動終止 (Stand still 狀態)
NMC_DeviceStopAll	系統中所有軸與群組運動終止 (Stopped 狀態)

系統訊息相關

函數名稱	說明
NMC_MessagePopFirst	讀取系統訊息佇列
NMC_MessageOutputEnable	訊息轉發一份至 MS Windows 系統訊息

函式追蹤相關

函數名稱	說明
NMC_DebugSetTraceMode	設定 API 追蹤模式
NMC_DebugSetHookData	設定傳入嵌入函式(Hook function)料結構指標
NMC_DebugSetHookFunction	設定嵌入函式(Hook function)
NMC_DebugGetApiAddress	讀取 API 的 Address

單軸參數設定相關函式

函數名稱	說明
NMC_AxisSetParamI32	設定軸參數(I32_T 資料型態)
NMC_AxisGetParamI32	讀取軸參數(I32_T 資料型態)
NMC_AxisSetParamF64	設定軸參數(F64_T 資料型態)
NMC_AxisGetParamF64	讀取軸參數(F64_T 資料型態)

單軸狀態控制相關函式

函數名稱	說明
NMC_AxisEnable	單軸啟用
NMC_AxisDisable	單軸停用
NMC_AxisGetStatus	讀取單軸運動資訊
NMC_AxisGetState	讀取單軸狀態

NMC_AxisResetState	清除單軸錯誤狀態
NMC_AxisResetDriveAlm	清除單軸伺服警報

單軸運動資訊讀取相關函式

函數名稱	說明
NMC_AxisGetCommandPos	讀取單軸命令位置
NMC_AxisGetActualPos	讀取單軸實際位置
NMC_AxisGetCommandVel	讀取單軸命令速度
NMC_AxisGetActualVel	讀取單軸實際速度
NMC_AxisGetMotionBuffSpace	讀取單軸運動指令暫存空間大小

單軸運動控制相關函式

函數名稱	說明
NMC_AxisPtp	啟動單軸點對點運動
NMC_AxisJog	啟動單軸速度運動

單軸歸原點運動函式

函數名稱	說明
NMC_AxisSetHomePos	設定原點位置
NMC_AxisHomeDrive	啟動單軸歸原點運動(驅動器執行)

單軸運動中止函式

函數名稱	說明
NMC_AxisHalt	單軸運動終止 (Stand still 狀態)
NMC_AxisStop	單軸運動終止 (Stopped 狀態)
NMC_AxisHaltAll	所有單軸運動終止 (Stand still 狀態)
NMC_AxisStopAll	所有單軸運動終止 (Stopped 狀態)

單軸運行中變動相關函式

函數名稱	說明
NMC_AxisVelOverride	單軸運行中速度改變
NMC_AxisAccOverride	單軸運行中加速度改變
NMC_AxisDecOverride	單軸運行中減速度改變

單軸總體速率設定相關函式

函數名稱	說明
NMC_AxisSetVelRatio	設定單軸速度百分比

NMC_AxisGetVelRatio	讀取單軸速度百分比

群組參數設定相關函式

函數名稱	說明
NMC_GroupSetParamI32	設定群組參數(I32_T 資料型態)
NMC_GroupGetParamI32	讀取群組參數(I32_T 資料型態)
NMC_GroupSetParamF64	設定群組參數(F64_T 資料型態)
NMC_GroupGetParamF64	讀取群組參數(F64_T 資料型態)
NMC_GroupAxSetParamI32	設定群組軸參數(I32_T 資料型態)
NMC_GroupAxGtParamI32	讀取群組軸參數(I32_T 資料型態)
NMC_GroupAxSetParamF64	設定群組軸參數(F64_T 資料型態)
NMC_GroupAxGetParamF64	讀取群組軸參數(F64_T 資料型態)

群組狀態控制相關函式

函數名稱	說明
NMC_GroupEnable	群組啟用
NMC_GroupDisable	群組停用
NMC_GroupGetStatus	讀取群組運動資訊
NMC_GroupGetState	讀取群組狀態
NMC_GroupResetState	清除群組錯誤狀態
NMC_GroupResetDriveAlm	清除群組軸伺服警報
NMC_GroupResetDriveAlmAll	清除群組中所有軸伺服警報

群組總體速率設定相關函式

函數名稱	說明
NMC_GroupSetVelRatio	設定單軸速度百分比
NMC_GroupGetVelRatio	讀取單軸速度百分比

群組點對點運動(軸座標系)相關函式

函數名稱	說明
NMC_GroupPtpAcs	啟動群組單軸之點對點運動
NMC_GroupPtpAcsAll	啟動群組所有軸之點對點運動

群組軸 Jog 運動(軸座標系)相關函式

函數名稱	說明
NMC_GroupJogAcs	啟動群組軸之速度運動

群組軸 Jog 運動(卡氏座標)相關函式

函數名稱	說明
NMC_GroupJogTcpFrame	啟動群組軸沿 TCP 座標之速度運動
NMC_GroupJogPcsFrame	啟動群組軸沿 PCS 座標之速度運動

群組點對點運動(卡氏座標)相關函式

函數名稱	說明
NMC_GroupPtpCart	啟動群組卡氏座標單軸之點對點運動
NMC_GroupPtpCartAll	啟動群組卡氏座標之點對點運動

群組運動中止函式

函數名稱	說明
NMC_GroupHalt	群組運動終止 (Stand still 狀態)
NMC_GroupStop	群組運動終止 (Stopped 狀態)
NMC_GroupHaltAll	所有群組運動終止 (Stand still 狀態)
NMC_GroupStopAll	所有群組運動終止 (Stopped 狀態)

群組運動資訊讀取相關函式

函數名稱	說明
NMC_GroupGetCommandPosAcs	讀取群組軸座標(ACS)命令位置
NMC_GroupGetActualPosAcs	讀取群組軸座標(ACS)實際位置
NMC_GroupGetCommandPosPcs	讀取群組卡式座標(PCS)命令位置
NMC_GroupGetActualPosPcs	讀取群組卡式座標(PCS)實際位置
NMC_GroupGetMotionBuffSpace	讀取群組運動指令暫存空間大小

群組歸原點運動函式

函數名稱	說明
NMC_GroupSetHomePos	設定群組軸原點位置
NMC_GroupAxesHomeDrive	啟動群組軸歸原點運動(驅動器執行)

群組 2D 直線圓弧補間運動相關函式

函數名稱	說明
NMC_GroupLineXY	群組 2D 直線補間運動
NMC_GroupCirc2R	群組 2D 圓弧補間運動給定終點和半徑
NMC_GroupCirc2C	群組 2D 圓弧補間運動給定終點和圓心
NMC_GroupCirc2B	群組 2D 圓弧補間運動給定終點和經過點

群組 3D 直線圓弧補間運動相關函式

函數名稱	說明
NMC_GroupLine	群組 3D 直線補間運動
NMC_GroupCircR	群組 3D 圓弧補間運動給定終點和半徑
NMC_GroupCircC	群組 3D 圓弧補間運動給定終點和圓心
NMC_GroupCircB	群組 3D 圓弧補間運動給定終點和經過點

Tool 教導相關函式

函數名稱	說明
NMC_ToolCalib_4p	Tool 教導- TCP 平移教導法
NMC_ToolCalib_4pWithZ	Tool 教導- TCP 平移與 Z 方向設定教導法
NMC_ToolCalib_4pWithOri	Tool 教導- TCP 平移與姿態設定教導法
NMC_ToolCalib_Ori	Tool 教導- TCP 姿態設定教導法

Base 教導相關函式

函數名稱	說明
NMC_BaseCalib_1p	Base 教導- 1p 法
NMC_BaseCalib_2p	Base 教導- 2p 法
NMC_BaseCalib_3p	Base 教導- 3p 法

3D Simulation 相關函式

函數名稱	說明
NMC_Group3DShow	創建 3D 模擬視窗或顯示視窗
NMC_Group3DHide	隱藏 3D 模擬視窗
NMC_Group3DAlwaysTop	設置 3D 模擬視窗永遠在最上層
NMC_Group3DDrawPath	啟用/禁用 3D 視窗中的繪製 TCP 路徑的功能
NMC_Group3DClearPath	清除 3D 模擬視窗中的所有 TCP 路徑

3.2. 系統相關 APIs

3.2.1. 版本及錯誤資訊讀取相關函式

3.2.1.1. NMC_GetLibVersion

回傳函式庫(Library)的版本編號: (Major.Minor.Stage.Build)

- **C/C++語法:**

I32_T NMC_GetLibVersion(I32_T *PRetMajor, I32_T *PRetMinor, I32_T *PRetStage, I32_T *PRetBuild);

- **參數:**

I32_T *PRetMajor: 輸入一指標變數，回傳主版本號碼, 可以輸入 NULL (0)忽略此參數

I32_T *PRetMinor: 輸入一指標變數，回傳次版本號碼, 可以輸入 NULL (0)忽略此參數

I32_T *PRetStage: 輸入一指標變數，回傳階段版本號碼, 可以輸入 NULL (0)忽略此參數

1~4:測試階段版本，5:正式發行版本

I32_T *PRetBuild: 輸入一指標變數，回傳編譯版本號碼, 可以輸入 NULL (0)忽略此參數

- **回傳值:**

以I32_T 資料型態回傳版本編號，回傳值意義如下:

Version = (Major x 10,000,000) + (Minor x 100,000) + (Stage x 10,000) + Build

- **用法:**

- **範例:**

```
U32_T   version;
I32_T   major, minor, stage, build;

version = NMC_GetLibVersion( &major, &minor, &stage, &build );
printf( "Library version = %d, (%d,%d,%d,%d) \n", version, major, minor, stage, build );
```

- **參閱:**

<無>

3.2.1.2. NMC_GetLibVersionString

回傳函式庫(Library)的版本編號 C-Style 字串: “Major.Minor.Stage.Build”

- **C/C++語法:**

void NMC_GetLibVersionString(char *PRetVersionString, U32_T StringSize);

- **參數:**

char *PRetVersionString: 輸入 C-Style 字元陣列，建議大小至少超過 32 char

U32_T StringSize: char 字元陣列大小

- **回傳值:**

無回傳值

- **用法:**

- **範例:**

```
char versionString[32];

NMC_GetLibVersionString( versionString, 32 );
printf( "Library version = (%s) \n", versionString );
```

- **參閱:**

<無>

3.2.1.3. NMC_GetErrorDescription

回傳錯誤碼描述 c 字串

- **C/C++語法:**

const char* NMC_GetErrorDescription(RTN_ERR ErrorCode, _opt_null_ char *PRetErrorDesc, U32_T StringSize);

- **參數:**

RTN_ERR ErrorCode: 欲查詢之錯誤碼

_opt_null_ char *PRetErrorDesc: 輸入 char 陣列，大小建議 512 以上

U32_T StringSize: char 陣列大小

- **回傳值:**

Const char* : 回傳錯誤代碼描述，若 ErrorCode 無定義則回傳 NULL.

- **用法:**

呼叫本函式成功後存入錯誤代碼描述。

- **範例:**

```
```

- **參閱:**

<無>

3.2.2.1. NMC_DeviceOpenUp

開啟控制器（阻塞式呼叫）

- **C/C++語法:**

```
RTN_ERR NMC_DeviceOpenUp( I32_T DevType, I32_T DevIndex, I32_T *PRetDevID );
```

- **參數:**

I32_T DevType: 指定驅動裝置型態(Device Type)

0:啟動為模擬器(Simulator)

1:啟動為 EtherCAT 裝置

I32_T DevIndex: 指定裝置序號，設定為 0

I32_T *PRetDevID:呼叫成功後回傳裝置識別號(DevID)

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

呼叫其他控制 API 前必須先呼叫 NMC_DeviceOpenUp()來啟動控制器，本函數為阻塞式(blocked)呼叫，函數會進行一系列初始化動作包含載入設定檔案等需耗費一段時間，待控制器狀態進入操作狀態(Operation state)後返回。

- **範例:**

- **參閱:**

NMC_DeviceShutdown()

3.2.2.2. NMC_DeviceShutdown

關閉控制器（阻塞式呼叫）

- **C/C++語法:**

```
RTN_ERR NMC_DeviceShutdown( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”（0），反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

整個應用程式結束前必須呼叫本函式來關閉控制器並釋放系統資源，呼叫本函式後會阻塞(blocked)直到關閉程式結束。

- **範例:**

```
```

- **參閱:**

NMC_DeviceOpenUp()

3.2.2.3. NMC_DeviceOpenUpRequest

送出開啟控制器請求(非阻塞式呼叫)

- **C/C++語法:**

```
RTN_ERR NMC_DeviceOpenUpRequest( I32_T DevType, I32_T DevIndex );
```

- **參數:**

I32_T DevType: 指定驅動裝置型態(Device Type)

I32_T DevIndex: 指定裝置序號，設定為 0

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

本函式呼叫後立即返回，必須使用 NMC_DeviceWaitOpenUpRequest()或使用 NMC_DeviceGetState()讀取控制器狀態確認是否進入可操作狀態。

- **範例:**

```
```

- **參閱:**

NMC_DeviceWaitOpenUpRequest()

3.2.2.4. NMC_DeviceWaitOpenUpRequest

等待控制器啟動程式(阻塞式呼叫)

- **C/C++語法:**

```
RTN_ERR NMC_DeviceWaitOpenUpRequest( U32_T WaitMs, I32_T *PRetDevID  );
```

- **參數:**

U32_T WaitMs: 等待啟動時間，單位ms，可設定NMC_WAIT_TIME_INFINITE等待程式結束

I32_T *PRetDevID:呼叫成功後回傳裝置識別號(DevID)

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

本函是一般用於確認控制器是否已成功啟動，通常在 NMC_DeviceOpenUpRequest()之後呼叫使用。

呼叫後會阻塞(Blocked)直到控制器進入「OPERATION」或者是「ERROR」狀態。

- **範例:**

```
```

- **參閱:**

NMC_DeviceOpenUpRequest()

3.2.2.5. NMC_DeviceShutdownRequest

送出關閉控制器請求(非阻塞式呼叫)

- **C/C++語法:**

```
RTN_ERR NMC_DeviceShutdownRequest( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

本函用於送出關閉控制器請求，呼叫後立即返回，必須使用 NMC_DeviceWaitShutdownRequest() 或使用 NMC_DeviceGetState()讀取控制器狀態確認是否已離開可操作狀態。

- **範例:**

- **參閱:**

3.2.2.6. NMC_DeviceWaitShutdownRequet

等待控制器關閉程式(阻塞式呼叫)

- **C/C++語法:**

```
RTN_ERR NMC_DeviceWaitShutdownRequest( I32_T DevID, U32_T WaitMs );
```

- **參數:**

I32_T DevID: 裝置識別號

U32_T WaitMs: 等待啟動時間，單位ms，可設定NMC_WAIT_TIME_INFINITE (0xFFFFFFFF)等待程式結束

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

本函是一般用於確認控制器是否已成功關閉，通常在 NMC_DeviceShutdownRequest()之後呼叫使用。呼叫後會阻塞(Blocked)直到控制器進入「OPERATION」或者是「ERROR」狀態。

- **範例:**

```
```

- **參閱:**

NMC_DeviceShutdownRequest()

3.2.3.1. NMC_DeviceCreate

取得控制器識別字

- **C/C++語法:**

```
RTN_ERR NMC_DeviceCreate( I32_T DevType, I32_T DevIndex, I32_T *PRetDevID );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T DevIndex: 指定裝置序號，設定為 0

I32_T *PRetDevID: 指呼叫成功後回傳裝置識別號(DevID)

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

參考進階系統初始化章節

- **範例:**

- **參閱:**

3.2.3.2. NMC_DeviceDelete

移除控制器識別字

- **C/C++語法:**

```
RTN_ERR NMC_DeviceDelete( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”（0），反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

參考進階系統初始化章節

- **範例:**

- **參閱:**

3.2.3.3. NMC_DeviceLoadIniConfig

載入控制器之設定參數

- **C/C++語法:**

```
RTN_ERR NMC_DeviceLoadIniConfig( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

本函式會根據「NexMotionLibConfig.ini」檔案內容載入控制器相關設定。

「NexMotionLibConfig.ini」預設安裝為 C:\Nexcom，使用者不需要自行建立此檔案，也應避免修改檔案名和其內容避免檔案載入錯誤。

NMC_DeviceLoadIniConfig()會依下列目錄順序搜尋「NexMotionLibConfig.ini」檔案:

1. NexMotion.dll 資料夾
2. C:\Nexcom
3. 系統目錄(C:\Windows\System32)

若使用者想要自訂「NexMotionLibConfig.ini」的路徑位置可另外可呼叫 NMC_SetIniPath()來設定 ini 檔案之路徑。

當函數 NMC_DeviceLoadIniConfig()成功呼叫後，控制器狀態進入「READY」狀態，可準備啟動。

- **範例:**

```
```

- **參閱:**

NMC_SetIniPath()

3.2.3.4. NMC_DeviceResetConfig

清除控制器之設定參數

- **C/C++語法:**

```
RTN_ERR NMC_DeviceResetConfig( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

NMC_DeviceResetConfig()用於清除控制器之設定，成功呼叫後控制器狀態返回「INIT」狀態。

若控制器處於操作中「OPERATION」狀態，不可呼叫本函式。

- **範例:**

```
```

- **參閱:**

3.2.3.5. NMC_DeviceStart

啟動控制器(阻塞式呼叫)

- **C/C++語法:**

```
RTN_ERR NMC_DeviceStart( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

當控制器狀態處於「READY」狀態時，呼叫本函式啟動控制器，當函式成功返回後控制器進入可操作「OPERATION」狀態。

參考進階系統初始化章節

- **範例:**

- **參閱:**

3.2.3.6. NMC_DeviceStop

停止控制器(阻塞式呼叫)

- **C/C++語法:**

```
RTN_ERR NMC_DeviceStop( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

當控制器狀態處於於「OPERATION」狀態，呼叫本函數來停止控制器，當函式成功返回控制器始進入「READY」狀態。

- **範例:**

```

```

- **參閱:**

3.2.3.7. NMC_DeviceStartRequest

送出啟動控制器的請求(非阻塞式呼叫)

- **C/C++語法:**

```
RTN_ERR NMC_DeviceStartRequest( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

當控制器狀態於「READY」狀態，呼叫本函式送出啟動控制器之請求，送出請求後立刻返回，可使用 NMC_DeviceGetState()讀取控制器狀態以確認控制器是否進入「OPERATION」狀態

- **範例:**

- **參閱:**

3.2.3.8. NMC_DeviceStopRequest

送出停止控制器的請求(非阻塞式呼叫)

- **C/C++語法:**

```
RTN_ERR NMC_DeviceStopRequest( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

當控制器狀態於「OPERATION」狀態，呼叫本函式送出停止控制器之請求，送出請求後立刻返回，可使用 NMC_DeviceGetState()讀取控制器狀態以確認控制器是否進入「READY」狀態

- **範例:**

```
```

- **參閱:**

3.2.3.9. NMC_DeviceGetState

讀取控制器的狀態

- **C/C++語法:**

```
RTN_ERR NMC_DeviceGetState( I32_T DevID, I32_T *PRetDeviceState );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T *PRetDeviceState:呼叫成功後回傳裝置狀態(Device State)

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

- **範例:**

```

```

- **參閱:**

3.2.4.1. NMC_DeviceWatchdogTimerEnable

啟動看門狗計時器

- **C/C++語法:**

```
RTN_ERR NMC_DeviceWatchdogTimerEnable( I32_T DevID, I32_T TimeoutMs, I32_T WDTMode );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T TimeoutMs: 設定看門狗計數器過時(Timeout)值，單位 milliseconds.

設定範圍 20~200000 ms

I32_T WDTMode:計數器到時(Timeout)模式

0:系統狀態切換至 READY(若系統狀態為 OPERATION)

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

參考看門狗計時器(Watch Dog Timer)章節說明

- **範例:**

參閱:

3.2.4.2. NMC_DeviceWatchdogTimerDisable

停止看門狗計時器

- **C/C++語法:**

```
RTN_ERR NMC_DeviceWatchdogTimerDisable( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”（0），反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

參考看門狗計時器(Watch Dog Timer)章節說明

- **範例:**

參閱:

3.2.4.3. NMC_DeviceWatchdogTimerReset

清除看門狗計數器

- **C/C++語法:**

```
RTN_ERR NMC_DeviceWatchdogTimerReset( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”（0），反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

參考看門狗計時器(Watch Dog Timer)章節說明

- **範例:**

參閱:

3.2.5.1. NMC_DeviceSetParam

3.2.5.2. NMC_DeviceGetParam

設定/讀取控制器之系統參數

- **C/C++語法:**

```
RTN_ERR NMC_DeviceSetParam( I32_T DevID, I32_T ParamNum, I32_T SubIndex, I32_T ParaValue );
RTN_ERR NMC_DeviceGetParam( I32_T DevID, I32_T ParamNum, I32_T SubIndex, I32_T *PRetParaValue );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T ParamNum: 參數類別編號

I32_T SubIndex: 參數序號

I32_T ParaValue: 設定之參數數值

I32_T *PRetParaValue: 呼叫成功後回傳

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

參數內容請參閱:系統參數章節

- **範例:**

- **參閱:**

系統參數章節

3.2.5.3. NMC_SetIniPath

設定 INI 檔案所在位置

- **C/C++語法:**

RTN_ERR NMC_SetIniPath(_opt_null_ const char *PIniPath);

- **參數:**

I32_T DevID: 裝置識別號

const char *PIniPath: 指定 ini 檔案之路徑，c-style 字串，可給定 NULL(0)代表清除路徑設定使用系統預設路徑

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

系統啟動過程中，會根據「NexMotionLibConfig.ini」載入設定值，使用者可以透過呼叫本函式來指定 INI 檔的路徑。系統預設路徑為 C:\NEXCOM\。

- **範例:**

```
```

- **參閱:**

NMC_DeviceOpenUp()

NMC_DeviceOpenUpRequest()

3.2.6.1. NMC_GetInputMemorySize

讀取數位輸入(Input)映射記憶體之大小

- **C/C++語法:**

```
RTN_ERR NMC_GetInputMemorySize ( I32_T DevID, U32_T *PRetSizeByte );
```

- **參數:**

I32_T DevID: 裝置識別號

U32_T *PRetSizeByte: 呼叫成功後回傳 I/O 輸入記憶體大小，單位 Byte

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

NMC_GetInputMemorySize() 用來讀取控制器中 I/O 記憶體的可存取範圍(0 到回傳 Size，單位 byte)。不同的控制器(不同型號)其可控制的 I/O 記憶體大小可能不同，因此，可使用此 API 來確認實際可存取之範圍。

控制器啟動後，可調用此函式。

- **範例:**

```

```

- **參閱:**

3.2.6.2. NMC_GetOutputMemorySize

讀取數位輸出(Output)映射記憶體之大小

- **C/C++語法:**

RTN_ERR NMC_GetOutputMemorySize(I32_T DevID, U32_T *PRetSizeByte);

- **參數:**

I32_T DevID: 裝置識別號

U32_T *PRetSizeByte: 呼叫成功後回傳 I/O 輸出記憶體大小，單位 Byte

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

NMC_GetOutputMemorySize () 用來讀取控制器中 I/O 記憶體的可存取範圍(0 到回傳 Size，單位 byte)。不同的控制器(不同型號)其可控制的 I/O 記憶體大小可能不同，因此，可使用此 API 來確認實際可存取之範圍。

控制器啟動後，可調用此函式。

- **範例:**

- **參閱:**

3.2.6.3. NMC_ReadInputMemory

讀取數位輸入(Input)映射記憶體

- **C/C++語法:**

RTN_ERR NMC_ReadInputMemory(I32_T DevID, U32_T OffsetByte, U32_T SizeByte, void *PRetValues);

- **參數:**

I32_T DevID: 裝置識別號

U32_T OffsetByte: 記憶體偏移量，從 0 開始，單位 byte

U32_T SizeByte: 欲讀取記憶體之大小，單位 byte

void *PRetValues:成功呼叫後存入指定的記憶體區段值於變數中

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

請參考 I/O 控制章節，了解如何將 I/O 映射到控制器內部 I/O 記憶體

控制器啟動後，可調用此函式。

假設一 I/O 裝置其 DI-0 到 DI-15 設定到 I/O Input 記憶體第一個 Byte 跟第二個 Byte，吾欲讀取第四個輸入點位 (DI-3)的值，請參考下面範例:

- **範例:**

```
RTN_ERR ret;
U32_T offsetByte = 0;
U32_T sizeByte   = 2;
U16_T diValue    = 0;
U16_T di_03;

ret = NMC_ReadInputMemory( devId, offsetByte, sizeByte, &diValue );
di_03 = ( diValue >> 3 ) & 1;
```

- **參閱:**

3.2.6.4. NMC_ReadOutputMemory

讀取數位輸出(Output)映射記憶體

- **C/C++語法:**

RTN_ERR NMC_ReadOutputMemory(I32_T DevID, U32_T OffsetByte, U32_T SizeByte, void *PRetValues);

- **參數:**

I32_T DevID: 裝置識別號

U32_T OffsetByte: 記憶體偏移量，從 0 開始

U32_T SizeByte: 欲讀取記憶體之大小

void *PRetValues:成功呼叫後存入指定的記憶體區段值於變數中

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

請參考 I/O 控制章節，了解如何將 I/O 映射到控制器內部 I/O 記憶體

控制器啟動後，可調用此函式。

假設一 I/O 裝置其 DO-0 到 DO-15 設定到 I/O Output 記憶體第一個 Byte 跟第二個 Byte，吾欲讀取第四個輸出點位 (DO-3)的值，請參考下面範例:

- **範例:**

```
RTN_ERR ret;
U32_T offsetByte = 0;
U32_T sizeByte   = 2;
U16_T doValue    = 0;
U16_T do_03;

ret = NMC_ReadOutputMemory( devId, offsetByte, sizeByte, &doValue );
do_03 = ( doValue >> 3 ) & 1;
```

- **參閱:**

3.2.6.5. NMC_WriteOutputMemory

寫入數位輸出(Output)映射記憶體

- **C/C++語法:**

RTN_ERR NMC_WriteOutputMemory(I32_T DevID, U32_T OffsetByte, U32_T SizeByte, const void *PValues);

- **參數:**

I32_T DevID: 裝置識別號

U32_T OffsetByte: 記憶體偏移量，從 0 開始

U32_T SizeByte: 欲寫入記憶體之大小

void *PRetValues:成功呼叫後將此變數的值存入指定的記憶體區段中

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

請參考 I/O 控制章節，了解如何將 I/O 映射到控制器內部 I/O 記憶體

控制器啟動後，可調用此函式。

假設一 I/O 裝置其 DO-0 到 DO-15 設定到 I/O Output 記憶體第一個 Byte 跟第二個 Byte，吾欲設定第四個輸出點位 (DO-3)的值，請參考下面範例:

- **範例:**

```
RTN_ERR ret;
U32_T offsetByte = 0;
U32_T sizeByte   = 2;
U16_T doValue    = 0;
U16_T do_03;

NMC_ReadOutputMemory( devId, offsetByte, sizeByte, &doValue );
doValue = doValue | ( ( 1 << 3 ) ); // Set bit 3 ON
//doValue = doValue & (~( 1 << 3 ) ); // Set bit 3 OFF
NMC_WriteOutputMemory( devId, offsetByte, sizeByte, &doValue );
```

- **參閱:**

3.2.6.6. NMC_ReadInputBit

3.2.6.7. NMC_ReadInputI8

3.2.6.8. NMC_ReadInputI16

3.2.6.9. NMC_ReadInputI32

讀取數位輸入(Input)映射記憶體 bit 值/byte 值/word 值/double-word 值

- **C/C++語法:**

```
RTN_ERR NMC_ReadInputBit( I32_T DevID, U32_T OffsetByte, U32_T BitIndex, BOOL_T *PRetBitValue );
RTN_ERR NMC_ReadInputI8( I32_T DevID, U32_T OffsetByte, I8_T *PRetI8Value );
RTN_ERR NMC_ReadInputI16( I32_T DevID, U32_T OffsetByte, I16_T *PRetI16Value );
RTN_ERR NMC_ReadInputI32( I32_T DevID, U32_T OffsetByte, I32_T *PRetI32Value );
```

- **參數:**

I32_T DevID: 裝置識別號

U32_T OffsetByte: 記憶體偏移量，從 0 開始，單位 byte

U32_T BitIndex: 欲讀取 Byte 中第幾個 Bit 號碼,讀取範圍為 0~7

BOOL_T *PRetBitValue: 回傳 bit 值(0 or 1)

I8_T *PRetI8Value: 回傳 byte（I8_T）值

I16_T *PRetI16Value: 回傳 word（I16_T）值

I32_T *PRetI32Value: 回傳 double-word(I32_T) 值

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

請參考 I/O 控制章節，了解如何將 I/O 映射到控制器內部 I/O 記憶體

控制器啟動後，可調用此函式。

- **範例:**

- **參閱:**

3.2.6.10. NMC_ReadOutputBit

3.2.6.11. NMC_ReadOutputI8

3.2.6.12. NMC_ReadOutputI16

3.2.6.13. NMC_ReadOutputI32

讀取數位輸出(Output)映射記憶體 bit 值/byte 值/word 值/double-word 值

- **C/C++語法:**

```
RTN_ERR NMC_ReadOutputBit( I32_T DevID, U32_T OffsetByte, U32_T BitIndex, BOOL_T
*PRetBitValue );
RTN_ERR NMC_ReadOutputI8( I32_T DevID, U32_T OffsetByte, I8_T *PRetI8Value );
RTN_ERR NMC_ReadOutputI16( I32_T DevID, U32_T OffsetByte, I16_T *PRetI16Value );
RTN_ERR NMC_ReadOutputI32( I32_T DevID, U32_T OffsetByte, I32_T *PRetI32Value );
```

- **參數:**

I32_T DevID: 裝置識別號

U32_T OffsetByte: 記憶體偏移量，從 0 開始，單位 byte

U32_T BitIndex: 欲讀取 Byte 中第幾個 Bit 號碼,讀取範圍為 0~7

BOOL_T *PRetBitValue: 回傳 bit 值(0 or 1)

I8_T *PRetI8Value: 回傳 byte（I8_T）值

I16_T *PRetI16Value: 回傳 word（I16_T）值

I32_T *PRetI32Value: 回傳 double-word(I32_T) 值

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”（0），反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

請參考 I/O 控制章節，了解如何將 I/O 映射到控制器內部 I/O 記憶體

控制器啟動後，可調用此函式。

- **範例:**

- **參閱:**

3.2.6.14. NMC_WriteOutputBit

3.2.6.15. NMC_WriteOutputI8

3.2.6.16. NMC_WriteOutputI16

3.2.6.17. NMC_WriteOutputI32

寫入數位輸出(Output)映射記憶體 bit 值/byte 值/word 值/double-word 值

- **C/C++語法:**

```
RTN_ERR NMC_WriteOutputBit( I32_T DevID, U32_T OffsetByte, U32_T BitIndex, BOOL_T BitValue );
RTN_ERR NMC_WriteOutputI8( I32_T DevID, U32_T OffsetByte, I8_T I8Value );
RTN_ERR NMC_WriteOutputI16( I32_T DevID, U32_T OffsetByte, I16_T I16Value );
RTN_ERR NMC_WriteOutputI32( I32_T DevID, U32_T OffsetByte, I32_T I32Value );
```

- **參數:**

I32_T DevID: 裝置識別號

U32_T OffsetByte: 記憶體偏移量，從 0 開始，單位 byte

U32_T BitIndex: 欲讀取 Byte 中第幾個 Bit 號碼,讀取範圍為 0~7

BOOL_T BitValue: 設定輸出 output bit 值(0 or 1)

I8_T I8Value: 設定輸出 output byte (I8_T) 值

I16_T I16Value: 設定輸出 output word (I16_T) 值

I32_T I32Value: 設定輸出 output double-word(I32_T) 值

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

請參考 I/O 控制章節，了解如何將 I/O 映射到控制器內部 I/O 記憶體

控制器啟動後，可調用此函式。

- **範例:**

- **參閱:**

3.2.7.1. NMC_DeviceGetAxisCount

讀取軸數量資訊

- **C/C++語法:**

```
RTN_ERR NMC_DeviceGetAxisCount( I32_T DevID, I32_T *PRetAxisCount );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T *PRetAxisCount: 成功呼叫回傳啟用單軸數量

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

載入設定檔後，可呼叫此函數確認啟用單軸數量。呼叫清除控制器函式 NMC_DeviceResetConfig() 後單軸數量為 0

- **範例:**

```

```

- **參閱:**

3.2.7.2. NMC_DeviceGetGroupCount

讀取群組數量資訊

- **C/C++語法:**

```
RTN_ERR NMC_DeviceGetGroupCount( I32_T DevID, I32_T *PRetGroupCount );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T *PRetGroupCount: 成功呼叫回傳啟用群組數量

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

載入設定檔後，可呼叫此函數確認啟用群組數量。呼叫清除控制器函式 NMC_DeviceResetConfig() 後群組數量為 0

- **範例:**

```
```

- **參閱:**

3.2.7.3. NMC_DeviceGetGroupAxisCount

讀取群組中軸數量資訊

- **C/C++語法:**

RTN_ERR NMC_DeviceGetGroupAxisCount(I32_T DevID, I32_T GroupIndex, I32_T *PRetGroupAxisCount);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 欲取得資訊之群組序號

I32_T *PRetGroupAxisCount:成功呼叫回傳該群組的軸數

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

載入設定檔後，可呼叫此函數確認某一啟用群組中實際的軸數。

- **範例:**

- **參閱:**

3.2.8. 軸或組名信息

3.2.8.1. NMC_AxisGetDescription

讀取指定軸的名稱描述資訊

- **C/C++語法:**

RTN_ERR NMC_AxisGetDescription(I32_T DevID, I32_T AxisIndex, U32_T DescStrSize, char *PRetAxisDescription);

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

U32_T DescStrSize: C-字串緩存空間大小, byte 單位

char *PRetAxisDescription: 回傳軸描述 C-字串

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

本函數用於讀取指定單軸之名稱描述，其資訊來源根據系統所載入之 NCF 檔案，因此必須在載入後方才能正常讀取，換言之系統狀態必須為 READY 狀態。

- **範例:**

```
```

- **參閱:**

3.2.8.2. NMC_GroupGetDescription

讀取指定群組的名稱描述資訊

- **C/C++語法:**

RTN_ERR NMC_GroupGetDescription(I32_T DevID, I32_T GroupIndex, U32_T DescStrSize, char *PRetGroupDescription);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

U32_T DescStrSize: C-字串緩存空間大小, byte 單位

char *PRetGroupDescription: 回傳軸描述 C-字串

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

本函數用於讀取指定群組之名稱描述，其資訊來源根據系統所載入之 NCF 檔案，因此必須在載入後方才能正常讀取，換言之系統狀態必須為 READY 狀態。

- **範例:**

- **參閱:**

3.2.9.1. NMC_DeviceEnableAll

啟用系統中所有軸與群組

- **C/C++語法:**

```
RTN_ERR NMC_DeviceEnableAll( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

控制器啟動後，方可調用此函式。

呼叫此函數會將系統中所有單軸與所有群組設定為啟動狀態(伺服啟動 Servo On)。啟動過程中會依序先由單軸先啟動之後接續各群組之啟動,若過程中如遇啟動失敗則程式停止並回傳錯誤。

- **範例:**

- **參閱:**

3.2.9.2. NMC_DeviceDisableAll

停用系統中所有軸與群組

- **C/C++語法:**

```
RTN_ERR NMC_DeviceDisableAll( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

控制器啟動後，方可調用此函式。

呼叫此函數會將系統中所有單軸與所有群組設定為停用狀態(伺服啟動 Servo Off)。停用過程中會依序先由單軸先停用之後接續各群組之停用,若過程中如遇停用失敗則程式停止並回傳錯誤。

- **範例:**

- **參閱:**

3.2.10.1. NMC_DeviceHaltAll

系統中所有軸與群組運動終止（Stand still 狀態）

- **C/C++語法:**

```
RTN_ERR NMC_DeviceHaltAll( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”（0），反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

控制器啟動後，方可調用此函式。

呼叫此函數會將系統中所有單軸與所有群組下達減速停止(Halt)指令，其效果等同單一 Halt 指令，過程中會依序先由單軸命令下達之後接續各群組，若過程中如命令下達失敗則程式停止並回傳錯誤。

- **範例:**

```
```

- **參閱:**

3.2.10.2. NMC_DeviceStopAll

系統中所有軸與群組運動終止（Stopped 狀態）

- **C/C++語法:**

```
RTN_ERR NMC_DeviceStopAll( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”（0），反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

控制器啟動後，方可調用此函式。

呼叫此函數會將系統中所有單軸與所有群組下達減速停止(Stop)指令，其效果等同單一 Stop 指令，過程中會依序先由單軸命令下達之後接續各群組，若過程中如命令下達失敗則程式停止並回傳錯誤。

- **範例:**

```
```

- **參閱:**

3.2.11.1. NMC_MessagePopFirst

讀取系統訊息佇列

- **C/C++語法:**

RTN_ERR NMC_MessagePopFirst(_opt_null_ NmcMsg_T *PRetMsg);

- **參數:**

_opt_null_ NmcMsg_T *PRetMsg: 取回系統訊息資料結構，本參數可設定為 NULL 表示僅從訊息佇列中移除訊息不取回該訊息

注意,若參數不為 NULL,呼叫 NMC_MessagePopFirst()前必須先初始化結構中”sizeOfStruct”成員變數，見下面範例程式

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

若系統中已無訊息，回傳 ERR_NEXMOTION_QUEUE_EMPTY

- **用法:**

NMC_MessagePopFirst()讀取訊息佇列中之訊息,會依最早(最舊)之訊息依序讀取,讀取後刪除。

- **範例:**

```
RTN_ERR  ret;
NmcMsg_T msg;

msg.sizeOfStruct = sizeof(msg);
do{
    ret = NMC_MessagePopFirst( &msg );
    if( ret == ERR_NEXMOTION_SUCCESS )
    {
        printf( " Msg:Type=%d, code=%d, src=%s, text=%s\n", msg.type, msg.code,
msg.source, msg.text );
    }
}while( ret == ERR_NEXMOTION_SUCCESS );
```

- **參閱:**

NMC_MessageOutputEnable()

3.2.11.2. NMC_MessageOutputEnable

訊息轉發一份至 MS Windows 系統訊息

- **C/C++語法:**

```
void NMC_MessageOutputEnable( BOOL_T Enable );
```

- **參數:**

BOOL_T Enable: False (0) 不轉發，True(1)轉發

- **回傳值:**

無回傳

- **用法:**

當呼叫NMC_MessageOutputEnable(True) 後系統訊息將轉發至MS Windows之系統訊息中

- **範例:**

```

```

- **參閱:**

NMC_MessagePopFirst()

3.2.12.1. NMC_DebugSetTraceMode

設定 API 追蹤模式

- **C/C++語法:**

```
void NMC_DebugSetTraceMode( I32_T TraceMode );
```

- **參數:**

I32_T TraceMode:

0:關閉 trace 功能,不輸出

1:只輸出發生錯誤之 API

2:輸出所有 API

- **回傳值:**

無回傳

- **用法:**

參考函式追蹤章節

3.2.12.2. NMC_DebugSetHookData

設定傳入嵌入函數(Hook function)料結構指標

- **C/C++語法:**

```
void NMC_DebugSetHookData( void *PHookUserData );
```

- **參數:**

void *PHookUserData: 傳入指標(Pointer)指向使用者定義之變數或資料結構

- **回傳值:**

無回傳

- **用法:**

參考函式追蹤章節

3.2.12.3. NMC_DebugSetHookFunction

設定嵌入函式(Hook function)

- **C/C++語法:**

```
void    NMC_DebugSetHookFunction( PF_NmcHookAPI PFHookFuncPtr );
```

- **參數:**

PF_NmcHookAPI PFHookFuncPtr: 設定嵌入函式(Hook function)之函式指標，

- **回傳值:**

無回傳

- **用法:**

參考函式追蹤章節

3.2.12.4. NMC_DebugGetApiAddress

讀取 API 的 Address

- **C/C++語法:**

```
const void* NMC_DebugGetApiAddress( const char *PApiName );
```

- **參數:**

const char *PApiName: 輸入 NexMotion API 之函式名稱

- **回傳值:**

const void*: 回傳 NexMotion API 之函式指標

- **用法:**

參考函式追蹤章節

3.3. 單軸相關 APIs

3.3.1. 單軸參數設定相關函式

3.3.1.1. NMC_AxisSetParamI32

3.3.1.2. NMC_AxisGetParamI32

3.3.1.3. NMC_AxisSetParamF64

3.3.1.4. NMC_AxisGetParamF64

設定/讀取軸參數(I32_T/F64_T 資料型態)

- **C/C++語法:**

```
RTN_ERR NMC_AxisSetParamI32( I32_T DevID, I32_T AxisIndex, I32_T ParamNum, I32_T SubIndex, I32_T ParaValueI32 );
RTN_ERR NMC_AxisGetParamI32( I32_T DevID, I32_T AxisIndex, I32_T ParamNum, I32_T SubIndex, I32_T *PRetParaValueI32 );
RTN_ERR NMC_AxisSetParamF64( I32_T DevID, I32_T AxisIndex, I32_T ParamNum, I32_T SubIndex, F64_T ParaValueF64 );
RTN_ERR NMC_AxisGetParamF64( I32_T DevID, I32_T AxisIndex, I32_T ParamNum, I32_T SubIndex, F64_T *PRetParaValueF64 );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

I32_T ParamNum: 參數組別索引編號

I32_T SubIndex: 個別參數索引編號

I32_T ParaValueI32: 設定之參數數值(有號整數)

F64_T ParaValueF64: 設定之參數數值(雙精浮點數)

I32_T *PRetParaValueI32: 回傳之參數數值(有號整數)

F64_T *PRetParaValueF64: 回傳之參數數值(雙精浮點數)

- **回傳值:**

回傳錯誤代碼。

呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

參數內容請參閱: 單軸參數章節。

控制器啟動過程中會依照設定檔案載入參數值，可調用本函式設定或讀取部分單軸參數，在呼叫函數進行參數設定與讀取參數值時，必須先確認參數之資料型態為有號整數(I32_T)或雙精浮點數(F64_T)，以選用適當之函數。

- **範例:**

- **參閱:**

單軸參數章節

3.3.2.1. NMC_AxisEnable

單軸啟用

- **C/C++語法:**

```
RTN_ERR NMC_AxisEnable( I32_T DevID, I32_T AxisIndex );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 在進行單軸運動前，必須先呼叫此函數讓單軸進入激磁狀態。
2. 若單軸已經處於激磁狀態，呼叫此函數會成功回傳。
3. 若單軸處於錯誤狀態，呼叫此函數會回傳錯誤代碼。此時，必須先解除錯誤狀況後，呼叫 NMC_AxisResetState()後，方可呼叫此函數讓單軸重新進入激磁狀態。

- **範例:**

```
RTN_ERR  ret = 0;
ret = NMC_AxisEnable( 0, 0 );
```

- **參閱:**

NMC_AxisDisable()

NMC_AxisGetState()

NMC_AxisResetState()

3.3.2.2. NMC_AxisDisable

單軸停用

- **C/C++語法:**

```
RTN_ERR NMC_AxisDisable( I32_T DevID, I32_T AxisIndex );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 單軸在激磁狀態下，可以呼叫此函數解除激磁狀態。
2. 單軸在運動狀態中，也可以呼叫此函數，立即中止目前正在進行的運動，並解除激磁狀態。此時，若有運動儲存在 Motion Queue 中，將會被清除。

- **範例:**

```
RTN_ERR  ret = 0;
ret = NMC_AxisDisable( 0, 0 );
```

- **參閱:**

NMC_AxisEnable()

NMC_AxisGetState()

3.3.2.3. NMC_AxisGetStatus

讀取單軸運動資訊

- **C/C++語法:**

```
RTN_ERR NMC_AxisGetStatus( I32_T DevID, I32_T AxisIndex, I32_T *PRetAxisStatus );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

I32_T *PRetAxisStatus: 回傳單軸運動資訊(Status of axis)

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

- **範例:**

```
RTN_ERR  ret    = 0;
I32_T    status = 0;
ret = NMC_AxisGetStatus( 0, 0, &status );
```

- **參閱:**

NMC_AxisGetState()

3.3.2.4. NMC_AxisGetState

讀取單軸狀態

- **C/C++語法:**

```
RTN_ERR NMC_AxisGetState( I32_T DevID, I32_T AxisIndex, I32_T *PRetAxisState );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

I32_T *PRetAxisState: 回傳單軸狀態

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 此函數可以在週期性的處理常式中呼叫，以確認單軸目前的狀態。
2. 單軸在進行運動或是 Homing 程式前，必須先處於激磁狀態，當呼叫 NMC_AxisEnable()後，可以透過呼叫 NMC_AxisGetState()確認單軸是否已進入激磁狀態。
3. 當單軸處於錯誤狀態或非正常停止狀態，呼叫 NMC_AxisResetState()後，可以透過呼叫 NMC_AxisGetState()確認單軸是否已解除錯誤狀態。

- **範例:**

```
RTN_ERR  ret   = 0;
I32_T    state = 0;
ret = NMC_AxisGetState( 0, 0, &state );
```

- **參閱:**

1. NMC_AxisGetStatus()
2. NMC_AxisEnable()
3. NMC_AxisResetState()

3.3.2.5. NMC_AxisResetState

清除單軸錯誤狀態

- **C/C++語法:**

RTN_ERR NMC_AxisResetState(I32_T DevID, I32_T AxisIndex);

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 當單軸處於錯誤狀態，或因啟動非正常停止運動且已停止狀態下，可以呼叫此函數將單軸回復到非激磁或激磁等正常狀態，使用者可以呼叫 NMC_AxisGetState()確認單軸是否已回復到正常狀態。
2. 呼叫此函數時，若單軸驅動器處於報警狀態(Alarm)，控制器會自動清除驅動器錯誤，當錯誤清除後，會將單軸狀態從錯誤狀態(AXIS_STATE_ERROR)回復到非激磁狀態(AXIS_STATE_DISABLE)。
3. 若單軸因啟動非正常停止運動且已停止狀態下，呼叫此函數可將單軸回復到激磁狀態(AXIS_STATE_STAND_STILL)。

- **範例:**

```
RTN_ERR  ret = 0;
ret = NMC_AxisResetState( 0, 0 );
```

- **參閱:**

1. NMC_AxisGetState()
2. NMC_AxisResetDriveAlm()

3.3.2.6. NMC_AxisResetDriveAlm

清除單軸伺服警報

- **C/C++語法:**

```
RTN_ERR NMC_AxisResetDriveAlm( I32_T DevID, I32_T AxisIndex );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 當單軸驅動器發生報警(Alarm)，通常驅動器會自動解除激磁，使用者可以在排除錯誤狀況後，呼叫此函數清除驅動器報警，並呼叫 NMC_AxisGetStatus()讀取單軸 Status，確認驅動器報警是否已經清除成功。
2. 並非所有驅動器報警都可以透過此函數加以清除，某些驅動器報警必須利用重新上電的方式加以清除。
3. 呼叫此函數成功清除驅動器報警後，單軸狀態並不會自動從錯誤狀態中回復，必須接著呼叫 NMC_AxisResetState()將單軸回復到非激磁狀態(AXIS_STATE_DISABLE)。

- **範例:**

```
RTN_ERR  ret = 0;
ret = NMC_AxisResetDriveAlm( 0, 0 );
```

- **參閱:**

1. NMC_AxisGetStatus()
2. NMC_AxisResetState()

3.3.2.7. NMC_AxisGetDriveAlmCode

3.3.3.1. NMC_AxisGetCommandPos

3.3.3.2. NMC_AxisGetActualPos

- **C/C++語法:**

```
TN_ERR NMC_AxisGetCommandPos( I32_T DevID, I32_T AxisIndex, F64_T *PRetCmdPos );
RTN_ERR NMC_AxisGetActualPos( I32_T DevID, I32_T AxisIndex, F64_T *PRetActPos );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

1. 回傳錯誤代碼， 若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。
2. F64_T *PRetCmdPos: 回傳命令位置，單位為 user unit
3. F64_T *PRetActPos: 回傳編碼器位置，單位為 user unit。

- **用法:**

呼叫此函數可以讀取單軸的命令位置與編碼器的回授位置。

- **範例:**

```
RTN_ERR  ret = 0;
F64_T    cmdPos = 0.0;
F64_T    actPos = 0.0;
ret = NMC_AxisGetCommandPos( 0, 0, &cmdPos );
ret = NMC_AxisGetActualPos( 0, 0, &actPos );
```

- **參閱:**

3.3.3.3. NMC_AxisGetCommandVel

3.3.3.4. NMC_AxisGetActualVel

- **C/C++語法:**

```
RTN_ERR NMC_AxisGetCommandVel( I32_T DevID, I32_T AxisIndex, F64_T *PRetCmdVel );
RTN_ERR NMC_AxisGetActualVel( I32_T DevID, I32_T AxisIndex, F64_T *PRetActVel );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

1. 回傳錯誤代碼， 若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。
2. F64_T *PRetCmdVel: 回傳單軸命令速度，單位為 user unit/sec。
3. F64_T *PRetActVel: 回傳單軸編碼器回授速度，單位為 user unit/sec。

- **用法:**

呼叫相關函數以取得單軸命令速度與編碼器回授速度。

- **範例:**

```
RTN_ERR  ret = 0;
F64_T    cmdVel = 0.0;
F64_T    actVel = 0.0;
ret = NMC_AxisGetCommandVel( 0, 0, &cmdVel );
ret = NMC_AxisGetActualVel( 0, 0, &actVel );
```

- **參閱:**

3.3.3.5. NMC_AxisGetMotionBuffSpace

- **C/C++語法:**

RTN_ERR NMC_AxisGetMotionBuffSpace(I32_T DevID, I32_T AxisIndex, I32_T *PRetFreeSpace);

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

1. 回傳錯誤代碼， 若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。
2. I32_T *PRetFreeSpace: 回傳單軸 Motion Queue 中尚可以儲存的運動數量。

- **用法:**

由於單軸提供 Motion Queue 的功能，使用者可以呼叫此函數確認尚可存入 Motion Queue 的運動數量。

- **範例:**

```
RTN_ERR  ret   = 0;
I32_T    space = 0;
ret = NMC_AxisGetMotionBuffSpace( 0, 0, &space );
```

- **參閱:**

1. NMC_AxisPtp()
2. NMC_AxisJog()
3. NMC_AxisHalt()

3.3.4.1. NMC_AxisHomeDrive

- **C/C++語法:**

```
RTN_ERR NMC_AxisHomeDrive( I32_T DevID, I32_T AxisIndex );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 在進行單軸運動前，使用者通常會呼叫此函數啟動單軸 Homing 程式，讓單軸回到定義的原點位置。
2. 與 Homing 相關的參數定義于單軸參數表中，包含：Homing method、加速度、速度與 Offset 等。其中，有關 Homing 程式的方法，必須視驅動器提供的方法而定。
3. 當單軸在進行 Homing 程式的過程中發生錯誤，單軸 Status 的 Bit 7 會變為 1，且單軸的狀態會切換到錯誤狀態(AXIS_STATE_ERROR)，此時，當單軸完全停止後(可讀取單軸 Status 確認 Bit 9 是否為 1)，可以呼叫 NMC_AxisResetDriveAlm()或是 NMC_AxisResetState()以回復到正常狀態。
4. 當單軸順利完成 Homing 程式，單軸 Status 的 Bit 18 會變為 1，且單軸的狀態回切換到正常激磁狀態(AXIS_STATE_STAND_STILL)。

- **範例:**

```
RTN_ERR  ret = 0;
ret = NMC_AxisHomeDrive( 0, 0 );
```

- **參閱:**

3.3.4.2. NMC_AxisSetHomePos

3.3.4.3. NMC_AxisPtp

- **C/C++語法:**

RTN_ERR NMC_AxisPtp(I32_T DevID, I32_T AxisIndex, F64_T TargetPos, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

F64_T TargetPos: 目標位置(單位: user unit)，本設定值將根據軸參數 0x30 (Absolute or relative programming)解釋為絕對位置或相對距離。

F64_T *PMaxVel: 利用指標的方式輸入目標速度，或直接輸入 0，代表不輸入目標速度，此時，控制器會以單軸參數 AXP_VM 的速度設定值進行運動規劃。

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 單軸必須處於激磁狀態下，方可呼叫此函數進行點對點運動。
2. 單軸正在進行 Homing 程式時，呼叫此函數將回傳錯誤碼。
3. 當單軸處於 AXIS_STATE_STOPPING、AXIS_STATE_STOPPED 與 AXIS_STATE_ERROR 等狀態下，呼叫此函數將回傳錯誤碼。使用者必須呼叫 NMC_AxisResetState()將單軸回復到正常激磁狀態(AXIS_STATE_STAND_STILL)後，才可順利呼叫此函數啟動點對點運動。
4. 若單軸正在進行其他運動，呼叫此函數後，會依據單軸參數 AXP_BUFF_PARAM 的設定，而有相對應的行為。
5. 呼叫此函數可啟動單軸進行點對點運動，運動結束後，單軸將運動到輸入的目標位置。若單軸參數 0x30 (Absolute or relative programming)設定為 1，則輸入此函數的目標位置為與目前位置的相對距離；若設定為 0，則輸入此函數的目標位置為絕對座標位置。
6. 當單軸在進行點對點運動時，單軸狀態會切換到 AXIS_STATE_DISCRETE_MOTION；到達目標位置後，若沒有接續新的運動，則單軸 Status 的 Bit 9 會變為 1，狀態會回復到正常激磁狀態(AXIS_STATE_STAND_STILL)。
7. 控制器會依照單軸參數中的 AXP_PROF_TYPE、AXP_ACC、AXP_DEC 與 AXP_JERK 進行速度曲線規劃。

8. 使用者可以透過 PMaxVel 利用指標方式輸入最大速度，此時，相對應的單軸參數 AXP_VM 會自動更改為輸入的最大速度，並依此進行速度規劃。
9. 若輸入的指標 PMaxVel 為 0，控制器將徑行依照單軸參數 AXP_VM 做為目標速度進行速度規劃。
10. 若單軸狀態處於 AXIS_STATE_STAND_STILL，則不管單軸參數 AXP_BUFF_PARAM 為何，呼叫此函數後都將立即啟動。
11. 若單軸狀態處於 AXIS_STATE_WAIT_SYNC，若單軸參數 AXP_BUFF_PARAM 為 Aborting，呼叫此函數後，原先儲存在 Motion Queue 的運動將被清除，而此次呼叫的點對點運動將被儲存到 Motion Queue，等待 Trigger 訊號的啟動。
12. 在進行點對點運動的過程中，在尚未到達目標位置前，可以呼叫 NMC_AxisHalt()以停止運動。

- **範例:**

```
RTN_ERR  ret = 0;
ret = NMC_AxisPtp( 0, 0, 100, 0 ) // 依照單軸參數AXP_VM作為目標速度進行速度規劃
```

- **參閱:**

1. NMC_AxisSetParamI32()、NMC_AxisSetParamF64()
2. NMC_AxisResetState()
3. NMC_AxisGetStatus()
4. NMC_AxisHalt()

3.3.4.4. NMC_AxisJog

- **C/C++語法:**

RTN_ERR NMC_AxisJog(I32_T DevID, I32_T AxisIndex, I32_T Dir, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

I32_T Dir: 目標速度方向，1 代表往正方向；-1 代表往負方向。

F64_T *PMaxVel: 利用指標的方式輸入目標速率，或直接輸入 0，代表不輸入目標速率，此時，控制器會以單軸參數中 AXP_VM 的設定值做為目標速率。

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 單軸必須處於激磁狀態下，方可呼叫此函數進行點對點運動。
2. 單軸正在進行 Homing 程式時，呼叫此函數將回傳錯誤碼。
3. 當單軸處於 AXIS_STATE_STOPPING、AXIS_STATE_STOPPED 與 AXIS_STATE_ERROR 等狀態下，呼叫此函數將回傳錯誤碼。使用者必須呼叫 NMC_AxisResetState()將單軸回復到正常激磁狀態(AXIS_STATE_STAND_STILL)後，才可順利呼叫此函數啟動點對點運動。
4. 呼叫此函數後，控制器將依據單軸參數 AXP_ACC 設定的加速度，藉以加速或減速到目標速度，單軸狀態會切換到 AXIS_STATE_CONTINUOUS_MOTION。
5. 利用指標PMaxVel輸入的目標速率會連帶修改單軸參數AXP_VM。輸入的目標速率可以為0。
6. 呼叫此函數後，當到達目標速度，此時單軸 Status 的 Bit 8 會變為 1，且以目標速度持續運動。
7. 若單軸正在進行其他運動，呼叫此函數後，會依據單軸參數 AXP_BUFF_PARAM 的設定，而有相對應的行為。
8. 呼叫此函數後，可以呼叫 NMC_AxisHalt()停止單軸運動。

- **範例:**

```
RTN_ERR  ret = 0;
F64_T    maxVel = 100;
ret = NMC_AxisJog( 0, 0, -1, &maxVel ); // 設定單軸往負方向運動；單軸參數AXP_VM
                                        修改為100，且作為目標速率進行速度規劃。
```

- **參閱:**

1. NMC_AxisSetParamI32()、NMC_AxisSetParamF64()
2. NMC_AxisResetState()
3. NMC_AxisGetStatus()
4. NMC_AxisHalt()

3.3.5.1. NMC_AxisHalt

- **C/C++語法:**

```
RTN_ERR NMC_AxisHalt( I32_T DevID, I32_T AxisIndex );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 當單軸在進行點對點運動或是速度運動時，可以呼叫此函數停止單軸運動，控制器會依據單軸參數 AXP_DEC 降速到 AXP_V_BASE 設定的速率。
2. 單軸正在進行 Homing 程式時，呼叫此函數將回傳錯誤碼。
3. 當單軸處於 AXIS_STATE_STOPPING、AXIS_STATE_STOPPED 與 AXIS_STATE_ERROR 等狀態下，呼叫此函數將回傳錯誤碼。
4. 當單軸處於 AXIS_STATE_DISABLE 或是 AXIS_STATE_STAND_STILL 等狀態，呼叫此函數不會回傳錯誤碼。
5. 若單軸參數 AXP_BUFF_PARAM 設定為 Aborting，呼叫此函數後將立即啟動正常停止運動；若 AXP_BUFF_PARAM 設定為 Buffered，則會等單軸 Status Bit 8 變為 1 後(前一個運動完成目標後)，才會啟動正常停止運動。
6. 呼叫此函數且成功啟動正常停止運動後，單軸狀態會切換到 AXIS_STATE_DISCRETE_MOTION，當單軸停止運動後，單軸 Status Bit 8、9 變為 1，且狀態將切換到正常激磁狀態 AXIS_STATE_STAND_STILL。

- **範例:**

```
RTN_ERR  ret = 0;
ret = NMC_AxisHalt( 0, 0 );
```

- **參閱:**

3.3.5.2. NMC_AxisStop

- **C/C++語法:**

RTN_ERR NMC_AxisStop(I32_T DevID, I32_T AxisIndex);

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 單軸不管在進行任何運動時，當有非預期狀況發生，需要停止目前正在進行的運動，都可以呼叫此函數啟動單軸進行非正常停止運動，控制器會依據單軸參數 AXP_STOP_PROF_DEC 降速到 AXP_V_BASE 設定的速率。
2. 單軸正在進行 homing 程式時，也可以呼叫此函數終止 homing 程式。
3. 呼叫此函數啟動非正常停止運動後，若單軸尚未停止，此時狀態會切換到 AXIS_STATE_STOPPING；停止後，單軸 Status Bit 9 變為 1，狀態會切換到 AXIS_STATE_STOPPED。
4. 當單軸狀態處於 AXIS_STATE_STAND_STILL，呼叫此函數後，單軸狀態將切換到 AXIS_STATE_STOPPED。
5. 呼叫此函數且單軸停止運動後，不允許再啟動單軸運動，使用者必須呼叫 NMC_AxisResetState()將單軸回復到正常激磁狀態 AXIS_STATE_STAND_STILL 後，方可啟動單軸運動。

- **範例:**

```
RTN_ERR  ret = 0;
ret = NMC_AxisStop( 0, 0 );
```

- **參閱:**

NMC_AxisResetState()

3.3.5.3. NMC_AxisHaltAll

- **C/C++語法:**

```
RTN_ERR NMC_AxisHaltAll( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”（0），反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

與 NMC_AxisHalt()行為一致，呼叫此函數可以正常停止指定 Device 所有單軸之運動。

- **範例:**

```
RTN_ERR  ret = 0;
ret = NMC_AxisHaltAll( 0 );
```

- **參閱:**

NMC_AxisHalt()

3.3.5.4. NMC_AxisStopAll

- **C/C++語法:**

RTN_ERR NMC_AxisStopAll(I32_T DevID);

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

與 NMC_AxisStop()行為一致，呼叫此函數可以啟動指定 Device 所有單軸進行非正常停止運動。

- **範例:**

```
RTN_ERR  ret = 0;
ret = NMC_AxisStopAll( 0 );
```

- **參閱:**

NMC_AxisStop()

3.3.6.1. NMC_AxisVelOverride

- **C/C++語法:**

```
RTN_ERR NMC_AxisVelOverride( I32_T DevID, I32_T AxisIndex, F64_T TargetVel );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

F64_T TargetVel: 目標速度(單位: user unit / sec)

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 只有單軸啟動點對點運動(NMC_AxisPtp)或是速度運動(NMC_AxisJog)後，才可以呼叫此函數改變目標速度。
2. 呼叫此函數輸入的目標速度不會改變單軸參數 AXP_VM，只對於目前正在進行的運動有效。
3. 若啟動的是速度運動，則輸入的目標速度一定可以達到，此時單軸 Status 的 Bit 8 會變為 1。
4. 若啟動的是點對點運動，則輸入的目標速度不一定可以達到，在滿足使用者設定的目標位置、加減速度下，控制器會依據輸入的目標速度自行規畫可行的速度曲線。

- **範例:**

```
RTN_ERR  ret = 0;
F64_T    maxVel = 100.0;
ret = NMC_AxisJog( 0, 0, 1, &maxVel );  // 啟動單軸進行速度運動，目標速度為100。
Sleep(100);
ret = NMC_AxisVelOverride( 0, 0, 50.0 );  // 將目標速度調降為50。
```

- **參閱:**

1. NMC_AxisPtp()
2. NMC_AxisJog()

3.3.6.2. NMC_AxisAccOverride

- **C/C++語法:**

```
RTN_ERR NMC_AxisAccOverride( I32_T DevID, I32_T AxisIndex, F64_T TargetAcc );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

F64_T TargetAcc: 目標加速度

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 只有單軸啟動點對點運動(NMC_AxisPtp)或是速度運動(NMC_AxisJog)後，才可以呼叫此函數改變加速度。
2. 輸入的加速度將會設定到單軸參數 AXP_ACC。
3. 若單軸已經到達目標速度(單軸 Status 的 Bit 12 變為 1)，則輸入的加速度對於目前的運動將不具影響。

- **範例:**

```
RTN_ERR  ret = 0;
I32_T    status = 0;
F64_T    maxVel = 50.0;
ret = NMC_AxisJog( 0, 0, 1, &maxVel );  // 啟動單軸進行速度運動，目標速度為50。
Sleep(100);
ret = NMC_AxisGetStatus( 0, 0, &status );
if( !( status & 0x1000 ) )
{
    ret = NMC_AxisAccOverride ( 0, 0, 100.0 );  // 改變加速度為100.0
}
```

- **參閱:**

3.3.6.3. NMC_AxisDecOverride

- **C/C++語法:**

```
RTN_ERR NMC_AxisDecOverride( I32_T DevID, I32_T AxisIndex, F64_T TargetDec );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

F64_T TargetDec: 目標減加速度

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 只有單軸啟動點對點運動(NMC_AxisPtp)或是正常停止運動(NMC_AxisHalt)後，才可以呼叫此函數改變減加速度。
2. 輸入的減加速度將會設定到單軸參數 AXP_DEC。
3. 若單軸啟動點對點運動，且已從目標速度逐漸進入設定末速(單軸Status的Bit 11變為1)，則輸入的減加速度對於目前的運動將不具影響。
4. 若單軸啟動正常停止運動 NMC_AxisHalt，在單軸還沒停止前，輸入的減加速度會影響目前運動。

- **範例:**

```
RTN_ERR  ret = 0;
I32_T    status = 0;
F64_T    maxVel = 50.0;
ret = NMC_AxisPtp( 0, 0, 100, 0 );  // 啟動單軸進行點對點運動，目標速度依據AXP_VM的設定值。
ret = NMC_AxisDecOverride ( 0, 0, 100.0 );  // 改變減加速度為100.0
```

- **參閱:**

3.3.7. 單軸總體速率設定相關函式

3.3.7.1. NMC_AxisSetVelRatio

- **C/C++語法:**

```
RTN_ERR NMC_AxisSetVelRatio( I32_T DevID, I32_T AxisIndex, F64_T Percentage );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

F64_T Percentage: 輸入速度比例

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

1. 不管單軸是否有在進行運動，皆可呼叫此函數設定速度比例，此速度比例的 default 設定值為 1.0。
2. 輸入的速度比例必須滿足大於等於 0，小於等於 1。
3. 當單軸處於 AXIS_STATE_STOPPING、AXIS_STATE_STOPPED 與 AXIS_STATE_ERROR 等狀態下，呼叫此函數將回傳錯誤碼。
4. 當單軸正在進行 homing 時，呼叫此函數將回傳錯誤碼。
5. 當呼叫此函數設定速度比例後，爾後啟動的單軸運動，輸入的速度最大值將會乘上此速度比例做為目標速度。
6. 若呼叫此函數時，單軸正在進行點對點運動或是速度運動，輸入的速度比例會改變目前運動的目標速度。

- **範例:**

```
RTN_ERR  ret = 0;
F64_T    maxVel = 50.0;
ret = NMC_AxisJog( 0, 0, 1, &maxVel );  // 啟動單軸進行速度運動，目標速度為50。
ret = NMC_AxisSetVelRatio( 0, 0, 0.5 ); // 改變速度比例，目標速度變為25。
```

- **參閱:**

3.3.7.2. NMC_AxisGetVelRatio

- **C/C++語法:**

```
RTN_ERR NMC_AxisGetVelRatio( I32_T DevID, I32_T AxisIndex, F64_T *PPercentage );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T AxisIndex: 單軸識別號

- **回傳值:**

1. 回傳錯誤代碼， 若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。
2. F64_T *PPercentage: 回傳速度比例

- **用法:**

呼叫此函數讀取目前設定的速度比例。

- **範例:**

```
RTN_ERR  ret = 0;
F64_T    velRatio = 0.0;
ret = NMC_AxisGetVelRatio( 0, 0, &velRatio );
```

- **參閱:**

3.4. 群組相關 APIs

3.4.1. 群組參數設定相關函式

3.4.1.1. NMC_GroupSetParamI32

設定群組參數（I32_T 數據型態）

- **C/C++語法:**

RTN_ERR NMC_GroupSetParamI32(I32_T DevID, I32_T GroupIndex, I32_T ParamNum, I32_T SubIndex, I32_T ParaValueI32);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T ParamNum: 參數的編號

I32_T SubIndex: 參數的次編號

I32_T ParaValueI32: 設定的數值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID        = 0;
I32_T   groupIndex   = 0;
I32_T   paramNum     = 0x30; //命令為絕對式或增量式的參數
I32_T   subIndex     = 0;
I32_T   paraValueI32 = 1;    //命令設定為增量式
RTN_ERR ret          = 0;

ret = NMC_GroupSetParamI32( devID, groupIndex，paramNum, subIndex, paraValueI32 );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.1.2. NMC_GroupGetParamI32

讀取群組參數（I32_T 數據型態）

- **C/C++語法:**

RTN_ERR NMC_GroupGetParamI32(I32_T DevID, I32_T GroupIndex, I32_T ParamNum, I32_T SubIndex, I32_T *PRetParaValueI32);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T ParamNum: 參數的編號

I32_T SubIndex: 參數的次編號

I32_T *PRetParaValueI32: 輸入指標變數，回傳參數的數值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID        = 0;
I32_T   groupIndex   = 0;
I32_T   paramNum     = 0x30; //讀取命令為絕對式或增量式的參數
I32_T   subIndex     = 0;
I32_T   paraValueI32 = 0;
RTN_ERR ret          = 0;

ret = NMC_GroupGetParamI32( devID, groupIndex，paramNum, subIndex, &paraValueI32 );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.1.3. NMC_GroupSetParamF64

設定群組參數（F64_T 數據型態）

- **C/C++語法:**

RTN_ERR NMC_GroupSetParamF64(I32_T DevID, I32_T GroupIndex, I32_T ParamNum, I32_T SubIndex, F64_T ParaValueF64);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T ParamNum: 參數的編號

I32_T SubIndex: 參數的次編號

F64_T ParaValueF64: 設定的數值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID        = 0;
I32_T   groupIndex   = 0;
I32_T   paramNum     = 0x21; //設定停止運動的減速度
I32_T   subIndex     = 0;
F64_T   paraValueF64 = 80.5; //停止運動的減速度的數值
RTN_ERR ret          = 0;

ret = NMC_GroupSetParamF64( devID, groupIndex，paramNum, subIndex, paraValueF64 );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.1.4. NMC_GroupGetParamF64

設定群組參數（F64_T 數據型態）

- **C/C++語法:**

RTN_ERR NMC_GroupGetParamF64(I32_T DevID, I32_T GroupIndex, I32_T ParamNum, I32_T SubIndex, F64_T *PRetParaValueF64);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T ParamNum: 參數的編號

I32_T SubIndex: 參數的次編號

F64_T *PRetParaValueF64: 輸入指標變數，回傳參數的數值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID        = 0;
I32_T   groupIndex   = 0;
I32_T   paramNum     = 0x21; //設定停止的減速度
I32_T   subIndex     = 0;
F64_T   paraValueF64 = 0;
RTN_ERR ret          = 0;

ret = NMC_GroupGetParamF64( devID, groupIndex，paramNum, subIndex, &paraValueF64 );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.1.5. NMC_GroupAxSetParamI32

設定群組軸參數(I32_T 資料型態)

- **C/C++語法:**

RTN_ERR NMC_GroupAxSetParamI32(I32_T DevID, I32_T GroupIndex, I32_T GroupAxisIndex, I32_T ParamNum, I32_T SubIndex, I32_T ParaValueI32);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T GroupAxisIndex: 群組軸識別號

I32_T ParamNum: 參數的編號

I32_T SubIndex: 參數的次編號

I32_T ParaValueI32: 設定的數值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID          = 0;
I32_T   groupIndex     = 0;
I32_T   groupAxisIndex = 2;
I32_T   paramNum       = 0x30; //命令為絕對式或增量式的參數
I32_T   subIndex       = 0;
I32_T   paraValueI32   = 1;    //命令設定為增量式
RTN_ERR ret            = 0;

ret = NMC_GroupAxSetParamI32( devID, groupIndex, groupAxisIndex, paramNum, subIndex, paraValueI32 );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.1.6. NMC_GroupAxGtParamI32

設定群組軸參數(I32_T 資料型態)

- **C/C++語法:**

RTN_ERR NMC_GroupAxGetParamI32(I32_T DevID, I32_T GroupIndex, I32_T GroupAxisIndex, I32_T ParamNum, I32_T SubIndex, I32_T *PRetParaValueI32);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T GroupAxisIndex: 群組軸識別號

I32_T ParamNum: 參數的編號

I32_T SubIndex: 參數的次編號

I32_T *PRetParaValueI32: 輸入指標變數，回傳參數的數值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID          = 0;
I32_T   groupIndex     = 0;
I32_T   groupAxisIndex = 2
I32_T   paramNum       = 0x30; //命令為絕對式或增量式的參數
I32_T   subIndex       = 0;
I32_T   paraValueI32   = 0;
RTN_ERR ret            = 0;

ret = NMC_GroupAxGetParamI32( devID, groupIndex, groupAxisIndex, paramNum, subIndex, &paraValueI32 );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.1.7. NMC_GroupAxSetParamF64

設定群組軸參數(F64_T 資料型態)

- **C/C++語法:**

RTN_ERR NMC_GroupAxSetParamF64(I32_T DevID, I32_T GroupIndex, I32_T GroupAxisIndex, I32_T ParamNum, I32_T SubIndex, F64_T ParaValueF64);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T GroupAxisIndex: 群組軸識別號

I32_T ParamNum: 參數的編號

I32_T SubIndex: 參數的次編號

F64_T ParaValueF64: 設定的數值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID          = 0;
I32_T   groupIndex     = 0;
I32_T   groupAxisIndex = 2;
I32_T   paramNum       = 0x21; //設定停止運動的減速度
I32_T   subIndex       = 0;
F64_T   paraValueF64   = 80.5; //停止運動的減速度的數值
RTN_ERR ret            = 0;

ret = NMC_GroupAxSetParamF64( devID, groupIndex, groupAxisIndex, paramNum, subIndex, paraValueF64 );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.1.8. NMC_GroupAxGetParamF64

設定群組軸參數(F64_T 資料型態)

- **C/C++語法:**

RTN_ERR NMC_GroupAxGetParamF64(I32_T DevID, I32_T GroupIndex, I32_T GroupAxisIndex, I32_T ParamNum, I32_T SubIndex, F64_T *PRetParaValueF64);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T GroupAxisIndex: 群組軸識別號

I32_T ParamNum: 參數的編號

I32_T SubIndex: 參數的次編號

F64_T *PRetParaValueF64: 設定的數值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID          = 0;
I32_T   groupIndex     = 0;
I32_T   groupAxisIndex = 2;
I32_T   paramNum       = 0x21; //設定停止運動的減速度
I32_T   subIndex       = 0;
F64_T   paraValueF64   = 0;    //停止運動的減速度的數值
RTN_ERR ret            = 0;

ret = NMC_GroupAxGetParamF64( devID, groupIndex, groupAxisIndex, paramNum, subIndex, &paraValueF64 );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.2.1. NMC_GroupEnable

啟用(Servo On)群組中所有的群組軸，若所有的群組軸成功啟用，則群組的狀態(state)將由 GROUP_DISABLE 切換為 GROUP_ENABLE。

- **C/C++語法:**

```
RTN_ERR NMC_GroupEnable( I32_T DevID, I32_T GroupIndex );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號。

- **回傳值:**

以RTN_ERR資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
RTN_ERR ret        = 0;

ret = NMC_GroupEnable( devID, groupIndex );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.2.2. NMC_GroupDisable

停用(Servo Off)群組中所有的群組軸，若所有的群組軸成功停用，則群組的狀態(state)切換為GROUP_DISABLE。

- **C/C++語法:**

RTN_ERR NMC_GroupDisable(I32_T DevID, I32_T GroupIndex);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號。

- **回傳值:**

以RTN_ERR資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於NexMotionError.h標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
RTN_ERR ret        = 0;

ret = NMC_GroupDisable( devID, groupIndex );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.2.3. NMC_GroupGetStatus

讀取群組的以 bit 為單位的狀態(status)，各 bit 所代表的意義如下表，若讀取到的 bit 數值為 1，則觸發此事件。

bit	說明
0	觸發外部緊急開關
1	驅動器有警報
2	超出硬體的正極限
3	超出硬體的負極限
4	超出軟體的正極限
5	超出軟體的負極限
6	所有群組軸皆啟用中(即狀態(state)為 GROUP_STAND_STILL)
7	任一群組軸有發生錯誤(即狀態(state)為 GROUP_ERROR_STOP)
9	所有的群組軸為沒有位置的變動
10	卡式座標運動(直線與圓弧)中，正在加速(或減速)至最高速度的階段。 (PTP 或 JOG 運動時，此 bit 為 0)
11	卡式座標運動(直線與圓弧)中，正在減速至目標位置或停止的階段。 (PTP 或 JOG 運動時，此 bit 為 0)
12	卡式座標運動(直線與圓弧)中，正在最高速度的階段。 (PTP 或 JOG 運動時，此 bit 為 0)
13	群組正在運動中(即狀態(state)為 GROUP_MOVING、GROUP_HOMING 或 GROUP_STOPPING)
14	群組已停止下來(即狀態(state)為 GROUP_STOPPED)

- **C/C++語法:**

```
RTN_ERR NMC_GroupGetStatus( I32_T DevID, I32_T GroupIndex, I32_T *PRetStatusInBit );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號。

I32_T *PRetStatusInBit: 輸入指標變數，回傳以 bit 為單位的狀態。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID       = 0;
I32_T   groupIndex  = 0;
I32_T   statusInBit = 0;
RTN_ERR ret         = 0;

ret = NMC_GroupGetStatus( devID, groupIndex, &statusInBit );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.2.4. NMC_GroupGetState

讀取群組的狀態(state)，詳細請參考下表。

數值	狀態(state)	說明
0	GROUP_DISABLE	有任一群組軸為停用狀態
1	GROUP_STAND_STILL	所有群組軸為啟用狀態
2	GROUP_STOPPED	下達 NMC_GroupStop()後，群組處於停止階段
3	GROUP_STOPPING	下達 NMC_GroupStop()後，群組處於正在停止階段
4	GROUP_MOVING	群組運動中
5	GROUP_HOMING	群組歸零中
6	GROUP_ERROR_STOP	任一群組軸有發生錯誤

- **C/C++語法:**

```
RTN_ERR NMC_GroupGetState( I32_T DevID, I32_T GroupIndex, I32_T *PRetState );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T *PRetState: 輸入指標變數，回傳群組的狀態。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
I32_T   state      = 0;
RTN_ERR ret        = 0;

ret = NMC_GroupGetState( devID, groupIndex, &state );
if( ret != 0 ) return ret;
```

- **參閱:**

〈無〉

3.4.2.5. NMC_GroupResetState

重置群組的狀態(state)，可重置群組狀態 GROUP_STOPPED 成狀態 GROUP_STAND_STILL；若群組狀態為 GROUP_ERROR_STOP，此時下達此 API 會自動清除驅動器的警報，所有群組軸皆無警報後，會切換為 GROUP_STAND_STILL。

- **C/C++語法:**

RTN_ERR NMC_GroupResetState(I32_T DevID, I32_T GroupIndex);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
RTN_ERR ret        = 0;

ret = NMC_GroupResetState( devID, groupIndex );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.2.6. NMC_GroupResetDriveAlm

清除指定的群組軸的驅動器的警報。

- **C/C++語法:**

RTN_ERR NMC_GroupResetDriveAlm(I32_T DevID, I32_T GroupIndex, I32_T GroupAxisIndex);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T GroupAxisIndex: 欲清除警報的群組軸識別號

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID          = 0;
I32_T   groupIndex     = 0;
I32_T   groupAxisIndex = 0;
RTN_ERR ret            = 0;

ret = NMC_GroupResetDriveAlm( devID, groupIndex, groupAxisIndex );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.2.7. NMC_GroupResetDriveAlmAll

清除所有群組軸的驅動器的警報。

- **C/C++語法:**

```
RTN_ERR NMC_GroupResetDriveAlmAll( I32_T DevID, I32_T GroupIndex );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
RTN_ERR ret        = 0;

ret = NMC_GroupResetDriveAlmAll( devID, groupIndex );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.2.8. NMC_GroupGetDriveAlmCode

3.4.3.1. NMC_GroupSetVelRatio

設定群組的運動速率百分比，範圍為 0.0 ~ 100.0%。

- **C/C++語法:**

```
RTN_ERR NMC_GroupSetVelRatio( I32_T DevID, I32_T GroupIndex, F64_T Percentage );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

F64_T Percentage: 欲設定的運動速率百分比(0% ~ 100.0%)

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
F64_T   percentage = 100.0;
RTN_ERR ret        = 0;

ret = NMC_GroupSetVelRatio( devID, groupIndex, percentage );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.3.2. NMC_GroupGetVelRatio

讀取群組的運動速率百分比，範圍為 0.0 ~ 100.0%。

- **C/C++語法:**

```
RTN_ERR NMC_GroupGetVelRatio( I32_T DevID, I32_T GroupIndex, F64_T *PRetPercentage );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

F64_T *PRetPercentage: 輸入指標變數，回傳運動速率百分比

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
F64_T   percentage = 0.0;
RTN_ERR ret        = 0;

ret = NMC_GroupGetVelRatio( devID, groupIndex, &percentage );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.4.1. NMC_GroupPtpAcs

執行群組的單群組軸的點對點運動，其輸入點位的坐標系統為 ACS(Axis coordinate system)，預設的最高速度將參考參數中的數值,若需要,可在此 API 中設定,設定後將自動儲存於參數中。

- **C/C++語法:**

RTN_ERR NMC_GroupPtpAcs(I32_T DevID, I32_T GroupIndex, I32_T GroupAxisIndex, F64_T AcsPos, _opt_null_ const F64_T *PAcsMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T GroupAxisIndex: 欲執行運動的群組軸識別號

F64_T AcsPos: 欲執行的點位位置值

_opt_null_ const F64_T *PAcsMaxVel: 輸入指標變數，可指定最高速度，輸入 NULL (0)則忽略此參數

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0),反之函數呼叫失敗回傳錯誤代碼,所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID          = 0;
I32_T   groupIndex     = 0;
I32_T   groupAxisIndex = 2;
F64_T   acsPos         = 35.5;
F64_T   acsMaxVel      = 80.0;
RTN_ERR ret            = 0;

ret = NMC_GroupPtpAcs( devID, groupIndex, groupAxisIndex, acsPos, &acsMaxVel );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.4.2. NMC_GroupPtpAcsAll

執行群組的多群組軸的點對點運動，其輸入點位的坐標系統為 ACS(Axis coordinate system)，指定運動的群組軸將同時啟動同時到達。

- **C/C++語法:**

RTN_ERR NMC_GroupPtpAcsAll(I32_T DevID, I32_T GroupIndex, I32_T GroupAxesIdxMask, const Pos_T *PAcsPos);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T GroupAxesIdxMask: 欲執行運動的群組軸組合碼，請參考下表。

Pos_T *PAcsPos: 輸入指標變數，為目標座標位置

群組軸	8	7	6	5	4	3	2	1
位數	7	6	5	4	3	2	1	0
數值（次方為位數）	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

群組軸組合碼(GroupAxesIdxMask)的使用方式如下述：

若移動的群組軸有第 1 軸、第 3 軸與第 8 軸，則群組軸組合碼為 $2^0 + 2^2 + 2^7 = 133$。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID            = 0;
I32_T   groupIndex       = 0;
I32_T   groupAxesIdxMask = 133; //移動第1、3與8軸
Pos_T   acsPos           = { 10, 20, 30, 40, 50, 60, 70 };
RTN_ERR ret              = 0;

ret = NMC_GroupPtpAcsAll( devID, groupIndex, groupAxesIdxMask, &acsPos );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.5. 群組軸 Jog 運動(軸坐標系)相關函式

3.4.5.1. NMC_GroupJogAcs

執行群組的單群組軸的 Jog 運動，預設的最高速度將參考參數中的數值，若需要，可在此 API 中設定，設定後將自動儲存於參數中。

- **C/C++語法:**

RTN_ERR NMC_GroupJogAcs(I32_T DevID, I32_T GroupIndex, I32_T GroupAxisIndex, I32_T Dir, _opt_null_ const F64_T *PAcsMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T GroupAxisIndex: 群組軸識別號

I32_T Dir: 旋轉方向

_opt_null_ const F64_T *PAcsMaxVel: 輸入指標變數，可指定最高速度，輸入 NULL (0)則忽略此參數

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID          = 0;
I32_T   groupIndex     = 0;
I32_T   groupAxisIndex = 2;
I32_T   dir            = 1;
RTN_ERR ret            = 0;

ret = NMC_GroupJogAcs( devID, groupIndex, groupAxisIndex, dir, NULL );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.6.1. NMC_GroupPtpCart

執行群組的單群組軸的點對點運動，其點位元單位為卡氏空間座標值。

- **C/C++語法:**

RTN_ERR NMC_GroupPtpCart(I32_T DevID, I32_T GroupIndex, I32_T CartAxis, F64_T CartPos);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T CartAxis: 欲執行運動的卡氏空間坐標軸的代號，請參考下表

F64_T CartPos: 欲執行的點位位置值

卡氏空間座標	V	U	C	B	A	Z	Y	X
數值	7	6	5	4	3	2	1	0

卡氏空間坐標軸的代號(CartAxis)的使用方式如下述：

若移動的卡氏空間座標為 Z 軸，則卡氏空間坐標軸的代號為 2。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
I32_T   cartAxis   = 2;
F64_T   cartPos    = 55.5;
RTN_ERR ret        = 0;

ret = NMC_GroupPtpCart( devID, groupIndex, cartAxis, cartPos );
if( ret != 0 ) return ret;
```

- **參閱:**

〈無〉

3.4.6.2. NMC_GroupPtpCartAll

執行群組的多群組軸的點對點運動，其點位元單位為卡氏空間座標值。

- **C/C++語法:**

RTN_ERR NMC_GroupPtpCartAll(I32_T DevID, I32_T GroupIndex, I32_T CartAxesMask, const Pos_T *PTargetPos);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T CartAxesMask: 欲執行運動的卡氏空間坐標軸的組合碼，請參考下表

const Pos_T *PTargetPos: 輸入指標變數，為目標座標位置

卡氏空間座標	V	U	C	B	A	Z	Y	X
位數	7	6	5	4	3	2	1	0
數值（次方為位數）	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

卡氏空間坐標軸的組合碼(CartAxesMask)的使用方式如下述：

若移動的卡氏空間座標為 X，Z 與 A 軸，則卡氏空間坐標軸的代號為 $2^0 + 2^2 + 2^3 = 13$。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID        = 0;
I32_T   groupIndex   = 0;
I32_T   cartAxesMask = 13; //移動X、Z與A軸
Pos_T   targetPos    = { 10, 20, 30, 40, 50, 60, 70 };
RTN_ERR ret          = 0;

ret = NMC_GroupPtpCartAll( devID, groupIndex, cartAxesMask, &targetPos );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.7. 群組軸 Jog 運動(卡氏座標)相關函式

3.4.7.1. NMC_GroupJogCartFrame

啟動群組軸卡式座標之速度運動

- **C/C++語法:**

RTN_ERR NMC_GroupJogCartFrame(I32_T DevID, I32_T GroupIndex, I32_T CartAxis, I32_T Dir, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T CartAxis: 欲執行運動的卡氏空間坐標軸的代號，請參考下表

卡氏空間座標	V	U	C	B	A	Z	Y	X
數值	7	6	5	4	3	2	1	0

卡氏空間坐標軸的代號(CartAxis)的使用方式如下述：

若移動的卡氏空間座標為 Z 軸，則卡氏空間坐標軸的代號為 2。

I32_T Dir: 移動方向，0:正向，1:負向

_opt_null_ const F64_T *PMaxVel: 輸入指標變數，指定 Jog 最高速度，輸入 NULL (0)則忽略此參數

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
```

- **參閱:**

<無>

3.4.7.2. NMC_GroupJogTcpFrame

3.4.7.3. NMC_GroupJogPcsFrame

3.4.8. 群組運動中止函式

3.4.8.1. NMC_GroupHalt

執行群組的停止運動，此 API 會將任何的運動立即減速，而群組的狀態(state)將會由 GROUP_MOVING 變成 GROUP_STOPPING(減速中)，若速度減至零，則狀態(state)變為 GROUP_STAND_STILL。

此 API 需搭配 Buffer mode 來指定此停止運動的行為：若 Buffer mode 為 Aborting，則將目前的運動立即減速；若 Buffer mode 為 Buffered，則會將此 API 加入至 Motion Buffer 中，不會立即生效。

- **C/C++語法:**

RTN_ERR NMC_GroupHalt(I32_T DevID, I32_T GroupIndex);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
RTN_ERR ret        = 0;

ret = NMC_GroupHalt( devID, groupIndex );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.8.2. NMC_GroupStop

執行群組的停止運動，此 API 會將任何的運動立即減速，而群組的狀態(state)將會由 GROUP_MOVING 變成 GROUP_STOPPING(減速中)，若速度減至零，則狀態(state)變為 GROUP_STOPPED。

若需要執行新的運動命令時，則需使用 NMC_GroupResetState()來重置目前的 GROUP_STOPPED 狀態。

- **C/C++語法:**

RTN_ERR NMC_GroupStop(I32_T DevID, I32_T GroupIndex);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
RTN_ERR ret        = 0;

ret = NMC_GroupStop( devID, groupIndex );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.8.3. NMC_GroupHaltAll

執行所有群組的停止運動，此 API 會將任何的運動立即減速，而群組的狀態(state)將會由 GROUP_MOVING 變成 GROUP_STOPPING(減速中)，若速度減至零，則狀態(state)變為 GROUP_STAND_STILL。

此 API 需搭配 Buffer mode 來指定此停止運動的行為：若 Buffer mode 為 Aborting，則將目前的運動立即減速；若 Buffer mode 為 Buffered，則會將此 API 加入至 Motion Buffer 中，不會立即生效。

- **C/C++語法:**

```
RTN_ERR NMC_GroupHaltAll( I32_T DevID );
```

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID = 0;
RTN_ERR ret   = 0;

ret = NMC_GroupHaltAll( devID );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.8.4. NMC_GroupStopAll

執行所有群組的停止運動，此 API 會將任何的運動立即減速，而群組的狀態(state)將會由 GROUP_MOVING 變成 GROUP_STOPPING(減速中)，若速度減至零，則狀態(state)變為 GROUP_STOPPED。

若需要執行新的運動命令時，則需使用 NMC_GroupResetState()來重置目前的 GROUP_STOPPED 狀態。

- **C/C++語法:**

RTN_ERR NMC_GroupStopAll(I32_T DevID);

- **參數:**

I32_T DevID: 裝置識別號

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID = 0;
RTN_ERR ret   = 0;

ret = NMC_GroupStopAll( devID );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.9.1. NMC_GroupGetCommandPosAcs

讀取群組的各群組軸的位置命令值，其坐標系統為 ACS(Axis coordinate system)。

- **C/C++語法:**

RTN_ERR NMC_GroupGetCommandPosAcs(I32_T DevID, I32_T GroupIndex, Pos_T *PRetCmdPosAcs);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

Pos_T *PRetCmdPosAcs: 輸入指標變數，回傳各群組軸的位置命令值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
Pos_T   cmdPosAcs  = { 0 };
RTN_ERR ret        = 0;

ret = NMC_GroupGetCommandPosAcs( devID, groupIndex, &cmdPosAcs );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.9.2. NMC_GroupGetActualPosAcs

讀取群組的各群組軸的位置實際值，此實際值由編碼器的 count 值經過齒輪比或導螺杆節距轉換後得到，其坐標系統為 ACS(Axis coordinate system)。

- **C/C++語法:**

RTN_ERR NMC_GroupGetActualPosAcs(I32_T DevID, I32_T GroupIndex, Pos_T *PRetActPosAcs);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

Pos_T *PRetActPosAcs: 輸入指標變數，回傳各群組軸的位置實際值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
Pos_T   actPosAcs  = { 0 };
RTN_ERR ret        = 0;

ret = NMC_GroupGetActualPosAcs( devID, groupIndex, &actPosAcs );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.9.3. NMC_GroupGetCommandPosPcs

讀取群組的工件座標的位置命令值，其坐標系統為 PCS(Product coordinate system)。

- **C/C++語法:**

RTN_ERR NMC_GroupGetCommandPosPcs(I32_T DevID, I32_T GroupIndex, Pos_T *PRetCmdPosPcs);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

Pos_T *PRetCmdPosPcs: 輸入指標變數，回傳各群組軸的位置命令值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
Pos_T   cmdPosPcs  = { 0 };
RTN_ERR ret        = 0;

ret = NMC_GroupGetCommandPosPcs( devID, groupIndex, &cmdPosPcs );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.9.4. NMC_GroupGetActualPosPcs

讀取群組的工件座標的位置實際值，此實際值由編碼器的 count 值經過齒輪比或導螺杆節距轉換後得到，其坐標系統為 PCS(Product coordinate system)。

- **C/C++語法:**

RTN_ERR NMC_GroupGetActualPosPcs(I32_T DevID, I32_T GroupIndex, Pos_T *PRetActPosPcs);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

Pos_T *PRetActPosPcs: 輸入指標變數，回傳各群組軸的位置命令值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
Pos_T   actPosPcs  = { 0 };
RTN_ERR ret        = 0;

ret = NMC_GroupGetCommandPosPcs( devID, groupIndex, &actPosPcs );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.9.5. NMC_GroupGetCommandPos

讀取群組的位置命令值，回傳的數值由 CoordSys(坐標系統)指定。

- **C/C++語法:**

RTN_ERR NMC_GroupGetCommandPos(I32_T DevID, I32_T GroupIndex, I32_T CoordSys, Pos_T *PRetCmdPos);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T CoordSys: 指定坐標系(0:MCS, 1:PCS, 2:ACS)

Pos_T *PRetCmdPosPcs: 輸入指標變數，回傳指定的坐標系統的位置命令值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
I32_T   coordSys   = 2; //讀取ACS坐標系統的位置命令值
Pos_T   cmdPos     = { 0 };
RTN_ERR ret        = 0;

ret = NMC_GroupGetCommandPos( devID, groupIndex, coordSys, &cmdPos );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.9.6. NMC_GroupGetActualPos

讀取群組的位置實際值，回傳的數值由 CoordSys(坐標系統)指定。

- **C/C++語法:**

RTN_ERR NMC_GroupGetActualPos(I32_T DevID, I32_T GroupIndex, I32_T CoordSys, Pos_T *PRetActPos);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T CoordSys: 指定坐標系(0:MCS, 1:PCS, 2:ACS)

Pos_T *PRetActPos: 輸入指標變數，回傳指定的坐標系統的位置實際值

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
I32_T   coordSys   = 2; //讀取ACS坐標系統的位置命令值
Pos_T   actPos     = { 0 };
RTN_ERR ret        = 0;

ret = NMC_GroupGetActualPos( devID, groupIndex, coordSys, &actPos );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.9.7. NMC_GroupGetMotionBuffSpace

讀取單軸運動指令暫存空間大小

- **C/C++語法:**

RTN_ERR NMC_GroupGetMotionBuffSpace(I32_T DevID, I32_T GroupIndex, I32_T *PRetFreeSpace);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

F64_T *PRetFreeSpace: 輸入指標變數，回傳 Motion Buffer 的剩餘空間

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
I32_T   freeSpace  = 0;
RTN_ERR ret        = 0;

ret = NMC_GroupGetMotionBuffSpace( devID, groupIndex, &freeSpace );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.10.1. NMC_GroupSetHomePos

設定群組軸原點位置

- **C/C++語法:**

RTN_ERR NMC_GroupSetHomePos(I32_T DevID, I32_T GroupIndex, I32_T GroupAxesIdxMask, const Pos_T *PHomePosAcs);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T GroupAxesIdxMask: 欲執行運動的群組軸組合碼，請參考下表。

const Pos_T *PHomePosAcs: 輸入指標變數，原點座標(ACS)位置

群組軸	8	7	6	5	4	3	2	1
位數	7	6	5	4	3	2	1	0
數值（次方為位數）	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

群組軸組合碼(GroupAxesIdxMask)的使用方式如下述：

若移動的群組軸有第 1 軸、第 3 軸與第 8 軸，則群組軸組合碼為 $2^0 + 2^2 + 2^7 = 133$。

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

- **參閱:**

<無>

3.4.10.2. NMC_GroupAxesHomeDrive

啟動群組軸歸原點運動(驅動器執行)

- **C/C++語法:**

```
RTN_ERR NMC_GroupAxesHomeDrive( I32_T DevID, I32_T GroupIndex, I32_T
GroupAxesIdxMask  );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T GroupAxesIdxMask: 欲執行運動的群組軸組合碼，請參考下表。

群組軸	8	7	6	5	4	3	2	1
位數	7	6	5	4	3	2	1	0
數值 (次方為位數)	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

群組軸組合碼(GroupAxesIdxMask)的使用方式如下述：

若移動的群組軸有第 1 軸、第 3 軸與第 8 軸，則群組軸組合碼為 $2^0 + 2^2 + 2^7 = 133$。

- **回傳值:**

回傳錯誤代碼。

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS” (0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

- **參閱:**

<無>

3.4.11.1. NMC_GroupLineXY

執行 XY 平面上的直線運動，將從目前的卡氏空間座標位置直線運動至目標座標位置，而狀態(state)將會由 GROUP_STAND_STILL 變成 GROUP_MOVING(移動中)，若到達目標位置即速度減至零，則狀態(state)變為 GROUP_STAND_STILL。預設的最高速度將參考參數中的數值，若需要，可在此 API 中設定，設定後將自動儲存於參數中。

- **C/C++語法:**

RTN_ERR NMC_GroupLineXY(I32_T DevID, I32_T GroupIndex, _opt_null_ const F64_T *PX, _opt_null_ const F64_T *PY, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

_opt_null_ const F64_T *PX: 輸入指標變數，設定 X 軸目標座標值，輸入 NULL(0)則忽略此參數(不移動)

_opt_null_ const F64_T *PY: 輸入指標變數，設定 Y 軸目標座標值，輸入 NULL(0)則忽略此參數(不移動)

_opt_null_ const F64_T *PMaxVel: 輸入指標變數，設定最高速度，輸入 NULL(0)則忽略此參數

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
F64_T   posX       = 10.0;
F64_T   posY       = 20.0;
RTN_ERR ret        = 0;

ret = NMC_GroupLineXY( devID, groupIndex, &posX, &posY, NULL );
if( ret != 0 ) return ret;
```

- **參閱:**

〈無〉

3.4.11.2. NMC_GroupCirc2R

執行 XY 平面上的圓弧運動(半徑法)，將從目前的卡氏空間座標位置圓弧運動至目標座標位置，而狀態(state)將會由 GROUP_STAND_STILL 變成 GROUP_MOVING(移動中)，若到達目標位置即速度減至零，則狀態(state)變為 GROUP_STAND_STILL。預設的最高速度將參考參數中的數值，若需要，可在此 API 中設定，設定後將自動儲存於參數中。

- **C/C++語法:**

RTN_ERR NMC_GroupCirc2R(I32_T DevID, I32_T GroupIndex, _opt_null_ const F64_T *PEX, _opt_null_ const F64_T *PEY, F64_T Radius, I32_T CW_CCW, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

_opt_null_ const F64_T *PEX: 輸入指標變數，設定 X 軸目標座標值，輸入 NULL(0)則忽略此參數(不移動)

_opt_null_ const F64_T *PEY: 輸入指標變數，設定 Y 軸目標座標值，輸入 NULL(0)則忽略此參數(不移動)

F64_T Radius: 圓弧的半徑(負值代表選擇弧度較大之路徑)

I32_T CW_CCW: 圓弧的旋轉方向(0=CW; 1=CCW)

_opt_null_ const F64_T *PMaxVel: 輸入指標變數，設定最高速度，輸入 NULL(0)則忽略此參數

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
F64_T   posX       = 50.0;
F64_T   posY       = 50.0;
F64_T   radius     = 25.0;
I32_T   cwCcw      = 1;
RTN_ERR ret        = 0;
```

```
ret = NMC_GroupCirc2R( devID, groupIndex, &posX, &posX, radius, cwCcw, NULL );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.11.3. NMC_GroupCirc2C

執行 XY 平面上的圓弧運動(圓心法)，將從目前的卡氏空間座標位置圓弧運動至目標座標位置，而狀態(state)將會由 GROUP_STAND_STILL 變成 GROUP_MOVING(移動中)，若到達目標位置即速度減至零，則狀態(state)變為 GROUP_STAND_STILL。預設的最高速度將參考參數中的數值，若需要，可在此 API 中設定，設定後將自動儲存於參數中。

- **C/C++語法:**

RTN_ERR NMC_GroupCirc2C(I32_T DevID, I32_T GroupIndex, _opt_null_ const F64_T *PEX, _opt_null_ const F64_T *PEY, _opt_null_ const F64_T *PCXOffset, _opt_null_ const F64_T *PCYOffset, I32_T CW_CCW, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

_opt_null_ const F64_T *PEX: 輸入指標變數，設定 X 軸目標座標值，輸入 NULL(0)則忽略此參數(不移動)

_opt_null_ const F64_T *PEY: 輸入指標變數，設定 Y 軸目標座標值，輸入 NULL(0)則忽略此參數(不移動)

_opt_null_ const F64_T *PCXOffset:輸入指標變數，設定 X 軸圓心座標值，輸入 NULL(0)則忽略此參數(不移動)

_opt_null_ const F64_T *PCYOffset:輸入指標變數，設定 Y 軸圓心座標值，輸入 NULL(0)則忽略此參數(不移動)

I32_T CW_CCW: 圓弧的旋轉方向(0=CW; 1=CCW)

_opt_null_ const F64_T *PMaxVel: 輸入指標變數，設定最高速度，輸入 NULL(0)則忽略此參數

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
F64_T   posTarX    = 20.0;
F64_T   posTarY    =  0.0;
F64_T   posCenX    = 10.0;
```

```
F64_T   posCenY     =  0.0;
I32_T   cwCcw       = 1;
RTN_ERR ret         = 0;

ret = NMC_GroupCirc2C( devID, groupIndex, &posTarX, NULL, &posCenX, NULL, cwCcw,
NULL );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.11.4. NMC_GroupCirc2B

執行 XY 平面上的圓弧運動(經過法)，將從目前的卡氏空間座標位置圓弧運動至目標座標位置，而狀態(state)將會由 GROUP_STAND_STILL 變成 GROUP_MOVING(移動中)，若到達目標位置即速度減至零，則狀態(state)變為 GROUP_STAND_STILL。預設的最高速度將參考參數中的數值，若需要，可在此 API 中設定，設定後將自動儲存於參數中。

- **C/C++語法:**

RTN_ERR NMC_GroupCirc2B(I32_T DevID, I32_T GroupIndex, _opt_null_ const F64_T *PEX, _opt_null_ const F64_T *PEY, _opt_null_ const F64_T *PBX, _opt_null_ const F64_T *PBY, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

_opt_null_ const F64_T *PEX: 輸入指標變數，設定 X 軸目標座標值，輸入 NULL(0)則忽略此參數(不移動)

_opt_null_ const F64_T *PEY: 輸入指標變數，設定 Y 軸目標座標值，輸入 NULL(0)則忽略此參數(不移動)

_opt_null_ const F64_T *PBX: 輸入指標變數，設定 X 軸經過座標值，輸入 NULL(0)則忽略此參數(不移動)

_opt_null_ const F64_T *PBY: 輸入指標變數，設定 Y 軸經過座標值，輸入 NULL(0)則忽略此參數(不移動)

_opt_null_ const F64_T *PMaxVel: 輸入指標變數，設定最高速度，輸入 NULL(0)則忽略此參數

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID      = 0;
I32_T   groupIndex = 0;
F64_T   posTarX    = 20.0;
F64_T   posTarY    = 20.0;
F64_T   posBorX    =  0.0;
F64_T   posBorY    = 20.0;
```

```
RTN_ERR ret        = 0;

ret = NMC_GroupCirc2B( devID, groupIndex, &posTarX, &posTarY, &posBorX,
&posBorY, NULL );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.11.5. NMC_GroupCirc2BEx

3.4.12.1. NMC_GroupLine

執行卡氏空間中的直線運動，將從目前的座標位置直線運動至目標座標位置，而狀態(state)將會由 GROUP_STAND_STILL 變成 GROUP_MOVING(移動中)，若到達目標位置即速度減至零，則狀態(state)變為 GROUP_STAND_STILL。

- **C/C++語法:**

RTN_ERR NMC_GroupLine(I32_T DevID, I32_T GroupIndex, I32_T CartAxisMask, const Pos_T *PCartPos, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T CartAxisMask: 欲執行運動的卡氏空間坐標軸的組合碼，請參考下表

const Pos_T *PCartPos: 輸入指標變數，為目標座標位置

_opt_null_ const F64_T *PMaxVel: 輸入指標變數，設定最高速度，輸入 NULL(0)則忽略此參數

卡氏空間座標	V	U	C	B	A	Z	Y	X
位數	7	6	5	4	3	2	1	0
數值 (次方為位數)	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

卡氏空間坐標軸的組合碼(CartAxesMask)的使用方式如下述：

若移動的卡氏空間座標為 X，Z 與 A 軸，則卡氏空間坐標軸的代號為 $2^0 + 2^2 + 2^3 = 13$。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID       = 0;
I32_T   groupIndex  = 0;
I32_T   cartAxesMask = 13; //移動X、Z與A軸
Pos_T   targetPos   = { 10, 20, 30, 40, 50, 60, 70 };
RTN_ERR ret         = 0;
```

```
ret = NMC_GroupLine( devID, groupIndex, cartAxesMask, &targetPos, NULL );
if( ret != 0 ) return ret;
```

- **參閱:**

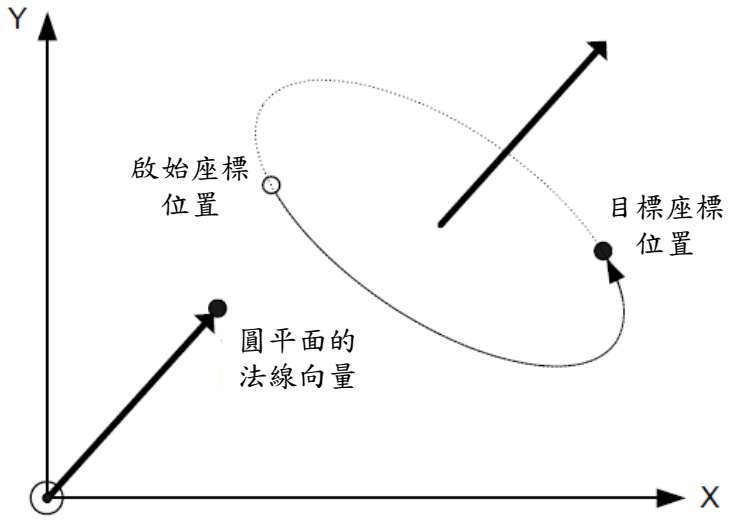
<無>

3.4.12.2. NMC_GroupCircR

執行卡氏空間中的圓弧運動(半徑法),將從目前的卡氏空間座標位置圓弧運動至目標座標位置，而狀態(state)將會由 GROUP_STAND_STILL 變成 GROUP_MOVING(移動中)，若到達目標位置即速度減至零，則狀態(state)變為 GROUP_STAND_STILL。

此圓弧運動(半徑法)的路徑演算法為參考 PLCopen 規範所製作，其注意事項如下：

- 圓平面的法線向量的起點為卡氏空間座標的原點，可不必輸入單位向量。
- 啟始座標位置與目標座標位置所形成的向量，必須與圓平面的法線向量垂直，否則會回傳錯誤代碼。
- 半徑法所規劃出來的路徑可分為弧度較大與較小的兩種，使用者需選擇其一來執行，此時則使用半徑(Radius)的正負號來決定，正號為選擇弧度較小的路徑；負號為選擇弧度較大的路徑。
- 若圓弧為 2D 路徑，可不必輸入圓平面的法線向量，但需設定旋轉方向(CW_CCW)，因可在 XY、YZ 或 ZX 平面上使用右手定則。
- 若圓弧為 3D 路徑，則不會參考旋轉方向(CW_CCW)，因 3D 路徑無法由右手定則決定。



- **C/C++語法:**

RTN_ERR NMC_GroupCircR(I32_T DevID, I32_T GroupIndex, I32_T CartAxisMask, const Pos_T *PCartPos, const Xyz_T *PNormalVector, F64_T Radius, I32_T CW_CCW, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T CartAxisMask: 欲執行運動的卡氏空間坐標軸的組合碼，請參考下表

const Pos_T *PCartPos: 輸入指標變數，為目標座標位置

const Xyz_T *PNormalVector: 輸入指標變數，為圓平面的法線向量

F64_T Radius: 圓弧的半徑(負值代表選擇弧度較大之路徑)

I32_T CW_CCW: 圓弧的旋轉方向(0=CW; 1=CCW)

_opt_null_ const F64_T *PMaxVel: 輸入指標變數，設定最高速度，輸入 NULL(0)則忽略此參數

卡氏空間座標	V	U	C	B	A	Z	Y	X
位數	7	6	5	4	3	2	1	0
數值 (次方為位數)	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

卡氏空間坐標軸的組合碼(CartAxesMask)的使用方式如下述：

若移動的卡氏空間座標為 X，Z 與 A 軸，則卡氏空間坐標軸的代號為 $2^0 + 2^2 + 2^3 = 13$。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳 “ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID        = 0;
I32_T   groupIndex   = 0;
I32_T   cartAxesMask = 3; //移動X與Y軸
Pos_T   targetPos    = { 50, 50, 0, 0, 0, 0, 0 };
Xyz_T   normalVec    = { 0, 0, 50 };
F64_T   radius       = 50.0;
I32_T   cwCcw        = 1;
RTN_ERR ret          = 0;

ret = NMC_GroupCircR( devID, groupIndex, cartAxesMask, &targetPos, &normalVec,
radius, cwCcw, NULL );
if( ret != 0 ) return ret;
```

- **參閱:**

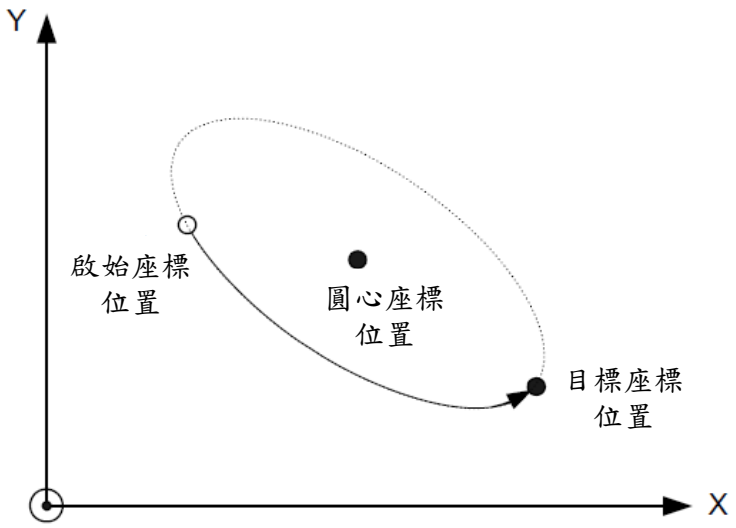
<無>

3.4.12.3. NMC_GroupCircC

執行卡氏空間中的圓弧運動(圓心法),將從目前的卡氏空間座標位置圓弧運動至目標座標位置，而狀態(state)將會由 GROUP_STAND_STILL 變成 GROUP_MOVING(移動中)，若到達目標位置即速度減至零，則狀態(state)變為 GROUP_STAND_STILL。

此圓弧運動(圓心法)的路徑演算法為參考 PLCopen 規範所製作，其注意事項如下：

- 若圓心法所規劃出來的圓弧為 3D 路徑，則可分為弧度較大與較小的兩種，使用者需選擇其一來執行，此時則使用旋轉方向(CW_CCW)來決定，CW 為選擇弧度較小的路徑；CCW 為選擇弧度較大的路徑。
- 若圓弧為 2D 路徑，因可在 XY、YZ 或 ZX 平面上使用右手定則，故可利用旋轉方向(CW_CCW)來決定弧度為選擇較大或較小的路徑。



- **C/C++語法:**

RTN_ERR NMC_GroupCircC(I32_T DevID, I32_T GroupIndex, I32_T CartAxisMask, const Pos_T *PCartPos, I32_T CenOfsMask, const Xyz_T *PCenOfs, I32_T CW_CCW, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T CartAxisMask: 欲執行運動的卡氏空間坐標軸的組合碼，請參考下表

const Pos_T *PCartPos: 輸入指標變數，為目標座標位置

I32_T CenOfsMask: 圓心座標位置的組合碼，請參考下表

const Xyz_T *PCenOfs: 輸入指標變數，為圓心座標位置

I32_T CW_CCW: 圓弧的旋轉方向(0=CW; 1=CCW)

_opt_null_ const F64_T *PMaxVel: 輸入指標變數，設定最高速度，輸入 NULL(0)則忽略此參數

卡氏空間座標	V	U	C	B	A	Z	Y	X
位數	7	6	5	4	3	2	1	0
數值（次方為位數）	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

卡氏空間坐標軸的組合碼(CartAxesMask)的使用方式如下述：
若移動的卡氏空間座標為 X，Z 與 A 軸，則卡氏空間坐標軸的代號為 $2^0 + 2^2 + 2^3 = 13$。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:
若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID        = 0;
I32_T   groupIndex   = 0;
I32_T   cartAxesMask = 7; //移動X、Y與Z軸
Pos_T   targetPos    = { 50, 50, 0, 0, 0, 0, 0 };
I32_T   cenOfsMask   = 7; //移動X、Y與Z軸
Xyz_T   cenOfs       = { 50, 0, 50 };
I32_T   cwCcw        = 1;
RTN_ERR ret          = 0;

ret = NMC_GroupCircC( devID, groupIndex, cartAxesMask, &targetPos, cenOfsMask,
&cenOfs, cwCcw, NULL );
if( ret != 0 ) return ret;
```

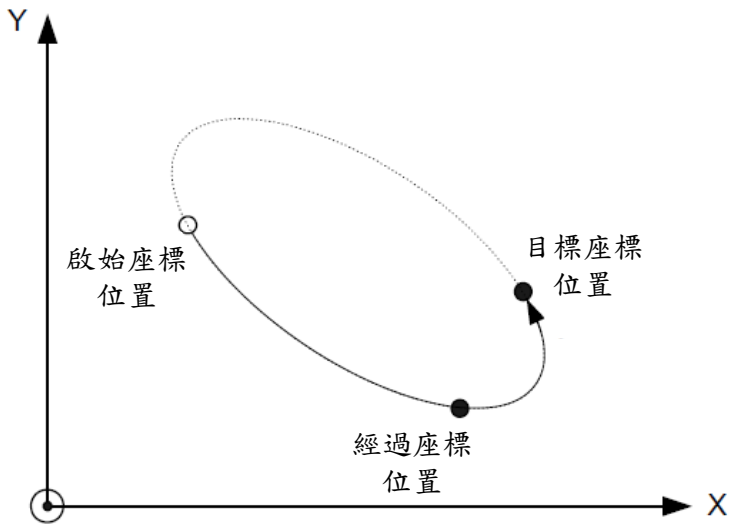
- **參閱:**

<無>

3.4.12.4. NMC_GroupCircB

執行卡氏空間中的圓弧運動(經過法),將從目前的卡氏空間座標位置圓弧運動至目標座標位置，而狀態(state)將會由 GROUP_STAND_STILL 變成 GROUP_MOVING(移動中)，若到達目標位置即速度減至零，則狀態(state)變為 GROUP_STAND_STILL。

此圓弧運動(圓心法)的路徑演算法為參考 PLCopen 規範所製作，其座標位置如下圖。



- **C/C++語法:**

RTN_ERR NMC_GroupCircB(I32_T DevID, I32_T GroupIndex, I32_T CartAxisMask, const Pos_T *PCartPos, I32_T BorPosMask, const Xyz_T *PBorPoint, _opt_null_ const F64_T *PMaxVel);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號

I32_T CartAxisMask: 欲執行運動的卡氏空間坐標軸的組合碼，請參考下表

const Pos_T *PCartPos: 輸入指標變數，為目標座標位置

I32_T BorPosMask: 經過座標位置的組合碼，請參考下表

const Xyz_T *PBorPoint: 輸入指標變數，為經過座標位置

_opt_null_ const F64_T *PMaxVel: 輸入指標變數，設定最高速度，輸入 NULL(0)則忽略此參數

卡氏空間座標	V	U	C	B	A	Z	Y	X
位數	7	6	5	4	3	2	1	0
數值 (次方為位數)	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

卡氏空間坐標軸的組合碼(CartAxesMask)的使用方式如下述：

若移動的卡氏空間座標為 X，Z 與 A 軸，則卡氏空間坐標軸的代號為 $2^0 + 2^2 + 2^3 = 13$。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **範例:**

```
I32_T   devID        = 0;
I32_T   groupIndex   = 0;
I32_T   cartAxesMask = 7; //移動X、Y與Z軸
Pos_T   targetPos    = { 50, 50, 0, 0, 0, 0, 0 };
I32_T   borPosMask   = 7; //移動X、Y與Z軸
Xyz_T   borPoint     = { 50, 0, 50 };
RTN_ERR ret          = 0;

ret = NMC_GroupCircB( devID, groupIndex, cartAxesMask, targetPos, borPosMask,
borPoint, NULL );
if( ret != 0 ) return ret;
```

- **參閱:**

<無>

3.4.12.5. NMC_GroupCircBEx

3.4.13.1. NMC_ToolCalib_4p

Tool 教導- TCP 平移教導法

- **C/C++語法:**

RTN_ERR NMC_ToolCalib_4p (const Pos_T *PMcsKinP1, const Pos_T *PMcsKinP2, const Pos_T *PMcsKinP3, const Pos_T *PMcsKinP4 , CoordTrans_T *PRetToolCoordTrans, F64_T *PRetTolerance);

- **參數:**

const Pos_T *PMcsKinP1: 第一步驟位置，TCP 必須(盡可能)落在參考位置上
const Pos_T *PMcsKinP2: 第二步驟位置，TCP 必須(盡可能)落在參考位置上
const Pos_T *PMcsKinP3: 第三步驟位置，TCP 必須(盡可能)落在參考位置上
const Pos_T *PMcsKinP4: 第四步驟位置，TCP 必須(盡可能)落在參考位置上
CoordTrans_T *PRetToolCoordTrans:回傳 Tool 座標轉換設定
F64_T *PRetTolerance:回傳重複誤差

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:
若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

- **範例:**

```
```

- **參閱:**

<無>

3.4.13.2. NMC_ToolCalib_4pWithZ

Tool 教導- TCP 平移與 Z 方向設定教導法

- **C/C++語法:**

RTN_ERR NMC_ToolCalib_4pWithZ （ const Pos_T *PMcsKinP1, const Pos_T *PMcsKinP2, const Pos_T *PMcsKinP3, const Pos_T *PMcsKinP4ZDir, CoordTrans_T *PRetToolCoordTrans, F64_T *PRetTolerance);

- **參數:**

const Pos_T *PMcsKinP1: 第一步驟位置，TCP 必須(盡可能)落在參考位置上

const Pos_T *PMcsKinP2: 第二步驟位置，TCP 必須(盡可能)落在參考位置上

const Pos_T *PMcsKinP3: 第三步驟位置，TCP 必須(盡可能)落在參考位置上

const Pos_T * PMcsKinP4ZDir: 第四步驟位置，TCP 正 Z 方向須指向 MCS 負 Z 方向

CoordTrans_T *PRetToolCoordTrans:回傳 Tool 座標轉換設定

F64_T *PRetTolerance:回傳重複誤差

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

- **範例:**

- **參閱:**

<無>

3.4.13.3. NMC_ToolCalib_4pWithOri

Tool 教導- TCP 平移與姿態設定教導法

- **C/C++語法:**

RTN_ERR NMC_ToolCalib_4pWithOri(const Pos_T *PMcsKinP1, const Pos_T *PMcsKinP2, const Pos_T *PMcsKinP3, const Pos_T *PMcsKinP4, const Pos_T *PMcsKinMinusZAxisPt, const Pos_T *PMcsKinYZPlanPt, CoordTrans_T *PRetToolCoordTrans, F64_T *PRetTolerance);

- **參數:**

const Pos_T *PMcsKinP1: 第一步驟位置，TCP 必須(盡可能)落在參考位置上

const Pos_T *PMcsKinP2: 第二步驟位置，TCP 必須(盡可能)落在參考位置上

const Pos_T *PMcsKinP3: 第三步驟位置，TCP 必須(盡可能)落在參考位置上

const Pos_T *PMcsKinP4: 第四步驟位置，TCP 必須(盡可能)落在參考位置上

const Pos_T *PMcsKinMinusZAxisPt: 第五步驟位置,參考位置必須落在 TCP 負 Z 軸上

const Pos_T *PMcsKinYZPlanPt: 第六步驟位置,參考位置必須落在 TCP 正 Y 軸上

CoordTrans_T *PRetToolCoordTrans:回傳 Tool 座標轉換設定

F64_T *PRetTolerance:回傳重複誤差

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

- **範例:**

```
```

- **參閱:**

<無>

3.4.13.4. NMC_ToolCalib_Ori

Tool 教導- TCP 平移教導法

- **C/C++語法:**

RTN_ERR NMC_ToolCalib_Ori (const Pos_T *PMcsKinOrg, const Pos_T *PMcsKinMinusZAxisPt, const Pos_T *PMcsKinYZPt, CoordTrans_T *PRetToolCoordTrans);

- **參數:**

const Pos_T * PMcsKinOrg: 第一步驟位置，TCP 必須(盡可能)落在參考位置上

const Pos_T *PMcsKinMinusZAxisPt: 第二步驟位置,參考位置必須落在 TCP 負 Z 軸上

const Pos_T *PMcsKinYZPlanPt: 第三步驟位置,參考位置必須落在 TCP 正 Y 軸上

CoordTrans_T *PRetToolCoordTrans:回傳 Tool 座標轉換設定，(只修改姿態轉換參數)

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

- **範例:**

- **參閱:**

<無>

3.4.14.1. NMC_BaseCalib_1p

Base 教導-1p 法

- **C/C++語法:**

RTN_ERR NMC_BaseCalib_1p(const Pos_T *PRefBaseP1, CoordTrans_T *PRetBaseCoordTrans);

- **參數:**

const Pos_T *PRefBaseP1: 第一步驟位置

const Pos_T *PRefBaseP2: 第二步驟位置

const Pos_T *PRefBaseP3: 第三步驟位置

CoordTrans_T *PRetBaseCoordTrans: 回傳相對於參考坐標系之座標轉換關係

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

- **範例:**

```
```

- **參閱:**

<無>

3.4.14.2. NMC_BaseCalib_2p

Base 教導-2p 法

- **C/C++語法:**

RTN_ERR NMC_BaseCalib_2p(const Pos_T *PRefBaseP1, const Pos_T *PRefBaseP2, CoordTrans_T *PRetBaseCoordTrans);

- **參數:**

const Pos_T *PRefBaseP1: 第一步驟位置

const Pos_T *PRefBaseP2: 第二步驟位置

CoordTrans_T *PRetBaseCoordTrans: 回傳相對於參考坐標系之座標轉換關係

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

- **範例:**

```
```

- **參閱:**

<無>

3.4.14.3. NMC_BaseCalib_3p

Base 教導-3p 法

- **C/C++語法:**

RTN_ERR NMC_BaseCalib_3p(const Pos_T *PRefBaseP1, const Pos_T *PRefBaseP2, const Pos_T *PRefBaseP3, CoordTrans_T *PRetBaseCoordTrans);

- **參數:**

const Pos_T *PRefBaseP1: 第一步驟位置

const Pos_T *PRefBaseP2: 第二步驟位置

const Pos_T *PRefBaseP3: 第三步驟位置

CoordTrans_T *PRetBaseCoordTrans: 回傳相對於參考坐標系之座標轉換關係

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

- **範例:**

```
```

- **參閱:**

<無>

3.4.15.1. NMC_Group3DShow

建立或顯示 3D 模擬視窗

- **C/C++語法:**

RTN_ERR NMC_Group3DShow(I32_T DevID, I32_T GroupIndex);

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號。

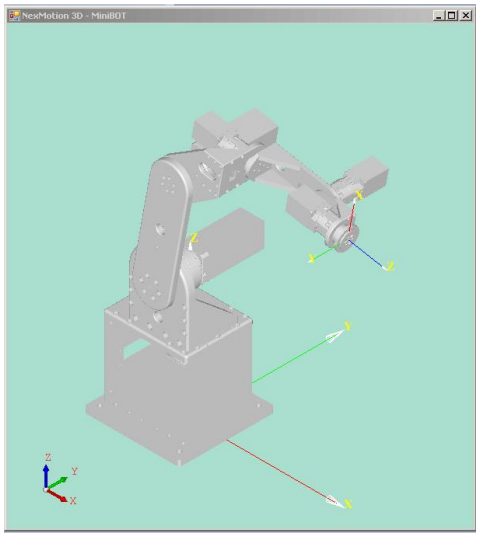
- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

系統啟動後，若第一次呼叫 NMC_Group3DShow() 是用來建立一個 3D 視窗並載入相關 3D 圖檔(如下圖)，根據不同校能硬體平台可能需花費數秒到數分鐘時間完成 3D 視窗建立。



若使用 NMC_Group3DHide()可將 3D 視窗隱藏可使用 NMC_Group3DShow()再將視窗顯示。

若無內建該群組 3D 圖檔，NMC_Group3DShow()將回傳錯誤碼。

- **範例:**

- **參閱:**

<無>

3.4.15.2. NMC_Group3DHide

隱藏 3D 模擬視窗

- **C/C++語法:**

```
RTN_ERR NMC_Group3DHide( I32_T DevID, I32_T GroupIndex );
RTN_ERR NMC_Group3DAlwaysTop( I32_T DevID, I32_T GroupIndex, BOOL_T Enable );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

系統啟動後，可呼叫 NMC_Group3DShow() 來建立一個 3D 視窗並載入相關 3D 圖檔，可使用 NMC_Group3DHide()可將 3D 視窗隱藏，另外可使用 NMC_Group3DShow()再將視窗顯示出來。

- **範例:**

```
```

- **參閱:**

<無>

3.4.15.3. NMC_Group3DAlwaysTop

設定 3D 視窗屬性永遠在最上層

- **C/C++語法:**

```
RTN_ERR NMC_Group3DAlwaysTop( I32_T DevID, I32_T GroupIndex, BOOL_T Enable );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號。

BOOL_T Enable:FALSE (0):取消最上層屬性; TRUE (1):設定最上層屬性

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

系統啟動後，可呼叫 NMC_Group3DShow() 來建立一個 3D 視窗並載入相關 3D 圖檔，可使用 NMC_Group3DAlwaysTop()將 3D 視窗設定為永遠在最上層

- **範例:**

```

```

- **參閱:**

<無>

3.4.15.4. NMC_Group3DDrawPath

於 3D 模擬視窗開啟/關閉繪製 TCP 路徑功能

- **C/C++語法:**

```
RTN_ERR NMC_Group3DDrawPath( I32_T DevID, I32_T GroupIndex, BOOL_T Enable );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號。

BOOL_T Enable:FALSE (0):取消路徑繪製; TRUE (1):啟動路徑繪製

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

系統啟動後，可呼叫 NMC_Group3DShow() 來建立一個 3D 視窗並載入相關 3D 圖檔，可使用 NMC_Group3DDrawPath()來啟動或關閉 TCP 路徑繪製於 3D 模擬視窗中，若要清除所有繪製的路徑可呼叫 NMC_Group3DClearPath()

- **範例:**

- **參閱:**

<無>

3.4.15.5. NMC_Group3DClearPath

於 3D 模擬視窗開啟/關閉繪製 TCP 路徑功能

- **C/C++語法:**

```
RTN_ERR NMC_Group3DClearPath( I32_T DevID, I32_T GroupIndex );
```

- **參數:**

I32_T DevID: 裝置識別號

I32_T GroupIndex: 群組識別號。

- **回傳值:**

以 RTN_ERR 資料型態回傳錯誤代碼，回傳值意義如下:

若呼叫函數成功回傳“ERR_NEXMOTION_SUCCESS”(0)，反之函數呼叫失敗回傳錯誤代碼，所有錯誤代碼定義於 NexMotionError.h 標頭檔中。

- **用法:**

系統啟動後，可呼叫 NMC_Group3DShow() 來建立一個 3D 視窗並載入相關 3D 圖檔，可使用 NMC_Group3DDrawPath()來啟動或關閉 TCP 路徑繪製於 3D 模擬視窗中，若要清除所有繪製的路徑可呼叫 NMC_Group3DClearPath()

- **範例:**

```

```

- **參閱:**

<無>

4. 函式庫定義

4.1. 資料型態定義

4.1.1. 基本資料型態

API 所使用的 C/C++資料型態定義於 nex_type.h 中，說明如下表:

型別	C/C++ 原型	說明	byte	範圍
BOOL_T	int	布林型別	4	0:False, 1:True
U8_T	unsigned char	無號整數	1	0 ~ 255
U16_T	unsigned short	無號整數	2	0 ~ 65535
U32_T	unsigned int	無號整數	4	0 ~ 4294967295
U64_T	unsigned __int64	無號整數	8	0 ~ 18446744073709551615
I8_T	char	有號整數	1	-128 ~ 127
I16_T	short	有號整數	2	-32768 ~ 32767
I32_T	int	有號整數	4	-2147483648 ~ 2147483647
I64_T	__int64	有號整數	8	-9223372036854775808 ~ 9223372036854775807
F32_T	float	浮點數	4	IEEE-754, 有效小數後 7 位
F64_T	double	雙精浮點數	8	IEEE-754, 有效小數後 15 位
RTN_ERR	int	錯誤代碼	4	-2147483648 ~ 2147483647

4.1.2. Motion 相關資料型態定義

Motion 相關資料型態定義於 NexMotionDef.h

4.1.2.1. Structure: Pos_T

描述群組(Group)之座標位置，視使用的 API 之定義內容可以是 ACS 座標資訊或 MCS/PCS 座標資訊

- **C/C++語法定義:**

```
typedef struct
{
    F64_T pos[MAX_POS_SIZE];
} Pos_T;
```

- **成員:**

F64_T pos[MAX_POS_SIZE]: 大小為 8 的位置陣列。

若描述 ACS 坐標系 pos[0~7]代表群組中 axis 0~7 之軸座標位置。

若描述為卡式坐標系統(MCS/PCS) pos[0~7]代表卡式坐標系統 X 軸、Y 軸、Z 軸、A 軸、B 軸、C 軸、U 軸和 V 軸之座標位置

4.1.2.2. Structure: Xyz_T

描述群組(Group)之卡式座標 X 軸、Y 軸和 Z 軸之位置

- **C/C++語法定義:**

```
typedef struct
{
    F64_T pos[MAX_XYZ_SIZE];
} Xyz_T;
```

- **成員:**

F64_T pos[MAX_XYZ_SIZE]: 大小為 3 的位置陣列。

pos[0]、pos[1]和 pos[2]分別代表卡式坐標系統 X 軸、Y 軸和 Z 軸之座標位置

4.1.2.3. Structure: CoordTrans_T

描述兩個標系之間之座標轉換關係

- **C/C++語法定義:**

```
typedef struct
{
    F64_T pose[NMC_MAX_POSE_DATA_SIZE];
} CoordTrans_T;
```

- **成員:**

F64_T pose[NMC_MAX_POSE_DATA_SIZE]: 大小為 6 的 F64_T 陣列。

pose[0]、pose[1]和 pose[2]分別代表卡式坐標系統 X 軸、Y 軸和 Z 軸之相對位置(平移)

pose[3]、pose[4]和 pose[5]分別代表卡式坐標系統 Z 軸、Y 軸和 X 軸之旋轉角度(degree)

4.1.3. 其他型態定義

4.1.3.1. 嵌入函式型態:PF_NmcHookAPI

- **C/C++語法定義:**

```
typedef void(*PF_NmcHookAPI)( const void *FPFuncAddress , const char *PFuncName, RTN_ERR
ReturnCode, void *PUserData );
```

- **成員:**

const void *FPFuncAddress: 目前被呼叫之函式指標

const char *PFuncName:目前被呼叫之函式名稱

RTN_ERR ReturnCode: 目前被呼叫之函式回傳值

void *PUserData:使用者資料指標，由NMC_DebugSetHookData()設定

4.1.3.2. Structure: NmcTime_T

描述系統時間之資料結構

- **C/C++語法定義:**

```
typedef struct
{
    U32_T year;         //  1601 through 30827
    U32_T month;        //  1 through 12.
    U32_T day;          //  1 through 31.
    U32_T hour;         //  0 through 23.
    U32_T minute;       //  0 through 59.
    U32_T second;       //  0 through 59
    U32_T milliseconds; //  0 through 999
} NmcTime_T;
```

- **成員:**

U32_T year: 西元年

U32_T month: 月, 1 ~ 12.

U32_T day:日, 1 ~ 31.

U32_T hour:時, 1~ 23.

U32_T minute:分 , 0 ~ 59.

U32_T second:秒 , 0 ~ 59

U32_T milliseconds:毫秒 , 0 ~ 999

4.1.3.3. Structure: NmcMsg_T

描述系統時間之資料結構

- **C/C++語法定義:**

```
typedef struct
{
    U32_T       sizeOfStruct;
    NmcTime_T   localTime;
    U32_T       index;
    I32_T       type;
    char        source[NMC_MAX_MSG_SOURCE_SIZE];
    I32_T       id;
    I32_T       code;
    char        text[NMC_MAX_MSG_TEXT_SIZE];
} NmcMsg_T;
```

- **成員:**

U32_T sizeOfStruct: 紀錄檢驗NmcMsg_T 資料結構之大小, 相當於sizeof(NmcMsg_T)

NmcTime_T localTime: 訊息產生當下之系統時間紀錄

U32_T index:訊息序號

I32_T type:訊息類別: 0:一般訊息(Normal), 1:警告訊息(Warning), 2:錯誤訊息(Error)

char source[NMC_MAX_MSG_SOURCE_SIZE]: 訊息來源(保留)

I32_T id;訊息識別(保留)

I32_T code;訊息代碼

char text[NMC_MAX_MSG_TEXT_SIZE]:訊息內容

4.2. 常數定義

NexMotion 相關常數定義於 NexMotionDef.h

4.2.1. 裝置型態(Device Type) 選擇

定義	常數值	說明
NMC_DEVICE_TYPE_SIMULATOR	0	模擬器裝置型態
NMC_DEVICE_TYPE_ETHERCAT	1	EtherCAT 裝置型態

4.2.2. Wait 函式 Timeout 設定

定義	常數值	說明
NMC_WAIT_TIME_INFINITE	0xFFFFFFFF	等待至該程式執行完畢

4.2.3. 控制器狀態(Device State)

定義	常數值	說明
NMC_DEVICE_STATE_INIT	1	初始化狀態(Init)，控制器參數尚未設定或已清除
NMC_DEVICE_STATE_READY	2	預備狀態(Ready)，參數已設定完成
NMC_DEVICE_STATE_ERROR	3	錯誤狀態(Error)，控制器進入重大錯誤狀態(如通訊中斷)
NMC_DEVICE_STATE_OPERATION	4	可操作狀態(Operation)

4.2.4. 坐標系統 (Coordinate system)選擇

定義	常數值	說明
NMC_COORD_MCS	0	機構坐標系(Mechanical coord. system)
NMC_COORD_PCS	1	程式坐標系(Programming coord. System)
NMC_COORD_ACS	2	軸坐標系 (Axis coord. System)

4.2.5. 單軸軸狀態(State of axis)

透過 NMC_AxisGetState()讀取單軸運動狀態，其變數值所代表意義如下示:

定義	常數值	說明
NMC_AXIS_STATE_DISABLE	0	禁用(Disable)狀態，伺服未啟動
NMC_AXIS_STATE_STAND_STILL	1	啟用(Enable)狀態，伺服啟動整定狀態
NMC_AXIS_STATE_HOMING	2	執行回原點(Homing)運動中
NMC_AXIS_STATE_DISCRETE_MOTION	3	執行點對點運動中
NMC_AXIS_STATE_CONTINUOUS_MOTION	4	執行連續運動中
NMC_AXIS_STATE_STOPPING	5	收到停止命令，減速停運動中
NMC_AXIS_STATE_STOPPED	6	收到停止命令，停止狀態
NMC_AXIS_STATE_WAIT_SYNC	7	收到 Wait 命令，等待 SYNC 訊號狀態中

NMC_AXIS_STATE_ERROR	10	發生錯誤，錯誤停止狀態

4.2.6. 單軸運動資訊(Status of Axis)位元編號

透過 NMC_AxisGetStatus()讀取單軸運動資訊，其變數之各位編號所代表意義如下示:

定義	常數值	說明
NMC_AXIS_STATUS_EMG	0	EMG 訊號發生栓鎖指示(*1)
NMC_AXIS_STATUS_ALM	1	伺服警報(Alarm)發生栓鎖指示 (*1)
NMC_AXIS_STATUS_PEL	2	正極限訊號發生栓鎖指示 (*1)
NMC_AXIS_STATUS_NEL	3	負極限訊號發生栓鎖指示 (*1)
NMC_AXIS_STATUS_PSEL	4	軟體正極限訊號發生栓鎖指示 (*1)
NMC_AXIS_STATUS_NSEL	5	軟體負極限訊號發生栓鎖指示 (*1)
NMC_AXIS_STATUS_ENA	6	單軸已啟用(Enable)停用(Disable)指示
NMC_AXIS_STATUS_ERR	7	單軸錯誤指示
NMC_AXIS_STATUS_TAR	8	單軸到達目標指示
NMC_AXIS_STATUS_CSTP	9	單軸命令停止指示
NMC_AXIS_STATUS_ACC	10	單軸于加速段指示
NMC_AXIS_STATUS_DEC	11	單軸于減速段指示
NMC_AXIS_STATUS_MV	12	單軸於最大速指示
NMC_AXIS_STATUS_OP	13	單軸於運動中指示
NMC_AXIS_STATUS_STOP	14	單軸於停止狀態中指示
NMC_AXIS_STATUS_RPEL	16	正極限訊號指示: 1:觸發, 0:未觸發
NMC_AXIS_STATUS_RNEL	17	負極限訊號指示: 1:觸發, 0:未觸發
NMC_AXIS_STATUS_RHOM	18	Home 訊號指示: 1:高准位(high), 0:低准位(Low)

(*1)栓鎖(Latched)訊號當錯誤狀態排除後，呼叫 NMC_AxisResetState()後只訊號才會清除為 0

4.2.7. 單軸運動資訊(Motion Status)位元遮罩

透過 NMC_AxisGetStatus()讀取單軸運動資訊，其變數之各位元遮罩(Bit Mask)所代表意義如下示:

定義	常數值	說明
NMC_AXIS_STATUS_MASK_EMG	0x00000001	EMG 訊號發生栓鎖指示遮罩(*1)
NMC_AXIS_STATUS_MASK_ALM	0x00000002	伺服警報(Alarm)發生栓鎖指示遮罩(*1)
NMC_AXIS_STATUS_MASK_PEL	0x00000004	正極限訊號發生栓鎖指示遮罩(*1)
NMC_AXIS_STATUS_MASK_NEL	0x00000008	負極限訊號發生栓鎖指示遮罩(*1)
NMC_AXIS_STATUS_MASK_PSEL	0x00000010	軟體正極限訊號發生栓鎖指示遮罩(*1)
NMC_AXIS_STATUS_MASK_NSEL	0x00000020	軟體負極限訊號發生栓鎖指示遮罩(*1)
NMC_AXIS_STATUS_MASK_ENA	0x00000040	單軸已啟用(Enable)停用(Disable)指示遮罩
NMC_AXIS_STATUS_MASK_ERR	0x00000080	單軸錯誤指示遮罩

NMC_AXIS_STATUS_MASK_TAR	0x00000100	單軸到達目標指示遮罩
NMC_AXIS_STATUS_MASK_CSTP	0x00000200	單軸命令停止指示遮罩
NMC_AXIS_STATUS_MASK_ACC	0x00000400	單軸于加速段指示遮罩
NMC_AXIS_STATUS_MASK_DEC	0x00000800	單軸于減速段指示遮罩
NMC_AXIS_STATUS_MASK_MV	0x00001000	單軸於最大速指示遮罩
NMC_AXIS_STATUS_MASK_OP	0x00002000	單軸於運動中指示遮罩
NMC_AXIS_STATUS_MASK_STOP	0x00004000	單軸於停止狀態中指示遮罩
NMC_AXIS_STATUS_MASK_RPEL	0x00010000	正極限訊號指示遮罩 1:觸發, 0:未觸發
NMC_AXIS_STATUS_MASK_RNEL	0x00020000	負極限訊號指示遮罩 1:觸發, 0:未觸發
NMC_AXIS_STATUS_MASK_RHOM	0x00040000	Home 訊號指示遮罩 1:高准位(high), 0:低准位(Low)

(*1)栓鎖(Latched)訊號當錯誤狀態排除後，呼叫 NMC_AxisResetState()後只訊號才會清除為 0

4.2.8. 群組坐標軸編號

定義	常數值	說明
GROUP_AXIS_X	0	坐標系統 X 軸
GROUP_AXIS_Y	1	坐標系統 Y 軸
GROUP_AXIS_Z	2	坐標系統 Z 軸
GROUP_AXIS_A	3	坐標系統 A 軸
GROUP_AXIS_B	4	坐標系統 B 軸
GROUP_AXIS_C	5	坐標系統 C 軸
GROUP_AXIS_U	6	坐標系統 U 軸
GROUP_AXIS_V	7	坐標系統 V 軸

4.2.9. 群組坐標軸號遮罩

定義	常數值	說明
GROUP_AXIS_MASK_X	0x00000001	坐標系統 X 軸遮罩
GROUP_AXIS_MASK_Y	0x00000002	坐標系統 Y 軸遮罩
GROUP_AXIS_MASK_Z	0x00000004	坐標系統 Z 軸遮罩
GROUP_AXIS_MASK_A	0x00000008	坐標系統 A 軸遮罩
GROUP_AXIS_MASK_B	0x00000010	坐標系統 B 軸遮罩
GROUP_AXIS_MASK_C	0x00000020	坐標系統 C 軸遮罩
GROUP_AXIS_MASK_U	0x00000040	坐標系統 U 軸遮罩
GROUP_AXIS_MASK_V	0x00000080	坐標系統 V 軸遮罩

4.2.10. 群組狀態(State of group)

透過 NMC_GroupGetState()讀取群組運動狀態，其變數值所代表意義如下示:

定義	常數值	說明
NMC_GROUP_STATE_DISABLE	0	群組為禁用(Disable)狀態，群組伺服未啟動
NMC_GROUP_STATE_STAND_STILL	1	群組啟用(Enable)狀態，群組伺服啟動整定狀態
NMC_GROUP_STATE_STOPPED	2	收到停止命令，停止狀態
NMC_GROUP_STATE_STOPPING	3	收到停止命令，減速運動中
NMC_GROUP_STATE_MOVING	4	執行運動命令中
NMC_GROUP_STATE_HOMING	5	執行回原點(Homing)運動中
NMC_GROUP_STATE_ERROR	6	發生錯誤，錯誤停止狀態

4.2.11. 群組運動資訊(status of group)位元編號

透過 NMC_GroupGetStatus()讀取單軸運動資訊，其變數之各位編號所代表意義如下示:

定義	常數值	說明
NMC_GROUP_STATUS_EMG	0	外部 EMG 訊號發生栓鎖指示(*1)
NMC_GROUP_STATUS_ALM	1	任意群組軸伺服警報(Alarm)發生栓鎖指示 (*1)
NMC_GROUP_STATUS_PEL	2	任意群組軸正極限訊號發生栓鎖指示 (*1)
NMC_GROUP_STATUS_NEL	3	任意群組軸負極限訊號發生栓鎖指示 (*1)
NMC_GROUP_STATUS_PSEL	4	任意群組軸軟體正極限訊號發生栓鎖指示 (*1)
NMC_GROUP_STATUS_NSEL	5	任意群組軸軟體負極限訊號發生栓鎖指示 (*1)
NMC_GROUP_STATUS_ENA	6	群組已啟用(Enable, 1)或停用(Disable, 0)指示
NMC_GROUP_STATUS_ERR	7	群組(任意群組軸)錯誤指示
NMC_GROUP_STATUS_CSTP	9	所有的群組軸無位置輸出(位置變化量為 0)
NMC_GROUP_STATUS_ACC	10	卡式座標運動(直線與圓弧)中,正在加速(或減速)至最高速度的階段，PTP 或 JOG 運動時，此位為 0
NMC_GROUP_STATUS_DEC	11	卡式座標運動(直線與圓弧)中,正在減速至目標位置或停止的階段，PTP 或 JOG 運動時，此位為 0
NMC_GROUP_STATUS_MV	12	卡式座標運動(直線與圓弧)中,正在最高速度的階段，PTP 或 JOG 運動時，此位為 0
NMC_GROUP_STATUS_OP	13	群組運動中，即狀態(state)為 GROUP_MOVING、GROUP_HOMING 或 GROUP_STOPPING
NMC_GROUP_STATUS_STOP	14	群組為停止狀態，即狀態(state)為 GROUP_STOPPED

(*1)栓鎖(Latched)訊號當錯誤狀態排除後,呼叫 NMC_GroupResetState()後只訊號才會清除為 0

4.3. 錯誤代碼(Error Code)

錯誤代碼定義於 NexMotionError.h 中，說明如下表:

定義	值	說明
ERR_NEXMOTION_SUCCESS	0	The operation completed successfully
ERR_NEXMOTION_EXTERNAL_LIBRARY_NOT_FOUND	-1	The system cannot find the externel library
ERR_NEXMOTION_API_NOT_FOUND	-2	The system cannot find an API address when specified library is loading
ERR_NEXMOTION_LOAD_EXTERNAL_LIBRARY_FAILED	-3	The system cannot load the externel library
ERR_NEXMOTION_LOAD_RUNTIME_FAILED	-4	The system cannot load the runtime library
ERR_NEXMOTION_FILE_NOT_FOUND	-5	The system cannot find the specified file
ERR_NEXMOTION_FILE_OPEN_FAILED	-6	The system cannot open the specified file
ERR_NEXMOTION_FILE_LOAD_FAILED	-7	The system cannot load the specified file
ERR_NEXMOTION_FILE_BAD_FORMAT	-8	The format of the file is bad
ERR_NEXMOTION_FILE_READ_PROHIBIT	-9	The file cannot be read
ERR_NEXMOTION_FILE_WRITE_PROHIBIT	-10	The file cannot be write
ERR_NEXMOTION_FILE_VERSION_INCOMPTIBLE	-11	The version of the file is incompatible
ERR_NEXMOTION_OPENUP_RUNTIME_FAILED	-12	The system cannot openup the runtime
ERR_NEXMOTION_OUT_OF_SYSTEM_RESOURCES	-15	The system resources is inefficient
ERR_NEXMOTION_EXTERNAL_CALL_FAILED	-16	An external call or system call has made an error
ERR_NEXMOTION_SYSTEM_NOT_INITIALIZATION	-21	The system cannot be accessed before initialization process
ERR_NEXMOTION_SYSTEM_CLOSED_DENIED	-22	The system cannot be closed before clean up process
ERR_NEXMOTION_OPERATION_DENIED	-23	In current state, the system cannot accept the operation
ERR_NEXMOTION_PERMISSION_DENIED	-24	The user does not have permission
ERR_NEXMOTION_UNEXPECTED_EXCEPTION	-25	The operation occurred unexpected exception
ERR_NEXMOTION_SYSTEM_NOT_READY	-26	The system is not ready
ERR_NEXMOTION_OPERATION_BUSY	-27	The system is busy and cannot accept the operation
ERR_NEXMOTION_WAIT_FAILED	-28	The wait function has failed. No wait condition
ERR_NEXMOTION_PROCESS_TIMEOUT	-31	System perform current process is timeout
ERR_NEXMOTION_RUNTIME_RESPONSE_TIMEOUT	-32	Runtime module does not respond in a certain time
ERR_NEXMOTION_OBJECT_ID_INVALID	-41	The system cannot find the object through specified "ID", "Index", or "Handle"
ERR_NEXMOTION_PARAMETER_NUMBER_INVALID	-42	The parameter number is invalid
ERR_NEXMOTION_PARAMETER_VALUE_INVALID	-43	The parameter value is invalid
ERR_NEXMOTION_PARAMETER_READ_ONLY	-44	The parameter is read only

ERR_NEXMOTION_ACCESS_AREA_INVALID	-45	The operation attempted to access invalid area
ERR_NEXMOTION_POINTER_NULL	-46	The input pointer variable is null
ERR_NEXMOTION_QUEUE_EMPTY	-47	The queue is empty.
ERR_NEXMOTION_STRUCT_SIZE_INCOMPTIBLE	-48	The size of structure is incomptible
ERR_NEXMOTION_INITIAL_AXIS_POSITION_INVALID	-100	The initial position of an axis is out of limit
ERR_NEXMOTION_INVERSE_KINEMATICS_FAILED	-101	Inverse kinematics process is failed
ERR_NEXMOTION_INVERSE_KINEMATICS_OVER_AXIS_LIMIT	-102	The solution of inverse kinematics is out of limit
ERR_NEXMOTION_INVERSE_KINEMATICS_SINGULAR	-103	The solution of inverse kinematics is singular
ERR_NEXMOTION_KINEMATICS_TYPE_INVALID	-104	The type of kinematics is invalid
ERR_NEXMOTION_AXIS_COUNT_INVALID	-105	The count of axis is invalid
ERR_NEXMOTION_GROUP_COUNT_INVALID	-106	The count of group is invalid
ERR_NEXMOTION_AXIS_MAPPING_INVALID	-107	The map setting of the axis is invalid
ERR_NEXMOTION_KINEMATICS_PARAMETER_INVALID	-108	The kinematics parameter of a group is invalid
ERR_NEXMOTION_EMERGEERR_NEXMOTION_EMERGENCY_STOP_ACTIVENCY_STOP_ACTIVE	-109	Emergency stop signal is detected.
ERR_NEXMOTION_ENABLE_SWITCH_FULL_PRESSED	-110	Enabling switch full pressed signal is detected.
ERR_NEXMOTION_SAFE_GUARD_ACTIVE	-111	Safe-guard signal is detected.
ERR_NEXMOTION_SAFETY_ERROR	-112	Safety error is detected.