HIWIN PCI-4P 運動函式庫範例手冊

第 1.1 版
2016 年 8 月 1 日

目錄

修訂記錄

版次	日期	適用範圍	註記
1.0	2015/11/05	PCI-4P 四軸運動控制卡	初版發行
1.1	2016/08/01	PCI-4P 四軸運動控制卡	函式庫更新

（此頁保留空白）

1. PCI-4P 四軸運動控制卡範例程式介紹

本手冊介紹之範例程式提供 console mode(VC++版本)與人機介面模式(VB 版本)，使用者須要自行整合文件中的範例程式到開發的應用程式。PCI-4P 運動控制函式庫(MCCL)最多可支援 12 張 PCI-4P 運動控制卡和 72 個 Group。為了簡化範例程式，本文僅使用 1 張 PCI-4P 運動控制卡，程式中所使用到的 wCardIndex 皆預設為 0；同樣的，僅使用到一組 Group，nGroupIndex 預設為 0。

2. 運動群組、機構參數設定

相關函式

MCC_SetSysMaxSpeed()
MCC_SetMacParam()
MCC_GetMacParam()
MCC_CreateGroup()
MCC_CloseGroup()
MCC_CloseAllGroups()
MCC_UpdateParam()

範例程式

InitSys.cpp (VC++)
InitSys.vb (VB)

說明

此範例說明如何設定運動群組與機構參數,使用 VC++開發請參考範例程式 InitSys.cpp (VC++)，若使用的是 VB 開發請參考範例程式 InitSys.vb (VB)，往後範例遵照此規範不再贅述。先使用 MCC_SetSysMaxSpeed()來設定最大進給速度；使用 MCC_SetMacParam()設定各軸的機構參數；最後使用 MCC_CreateGroup()來設定運動群組。

機構參數

本運動控制函式庫利用機構參數來定義使用者的機構平台特性,並利用機構參數規劃相對於邏輯原點的座標系統、座標邊界和各軸的最大安全進給速度。

機構參數說明：

wRPM：馬達最大安全轉速

各軸馬達最大安全轉速。各軸進行點對點移動時，各軸馬達的最大轉速將不會超過 *wRPM* 設定的值。

dwPPR：馬達旋轉一圈編碼器的計數值，此值相當於每旋轉一圈所需的 pulse 數

當使用閉迴路控制，此值為馬達旋轉一圈編碼器的計數值；如果操作在開迴路，此值為馬達旋轉一圈所需的 pulse 數。

dfPitch：導螺桿間隙

單位為 mm 或 inch。如果沒有配置導螺桿則此值應設為 1。

dfGearRatio：減速箱齒輪比

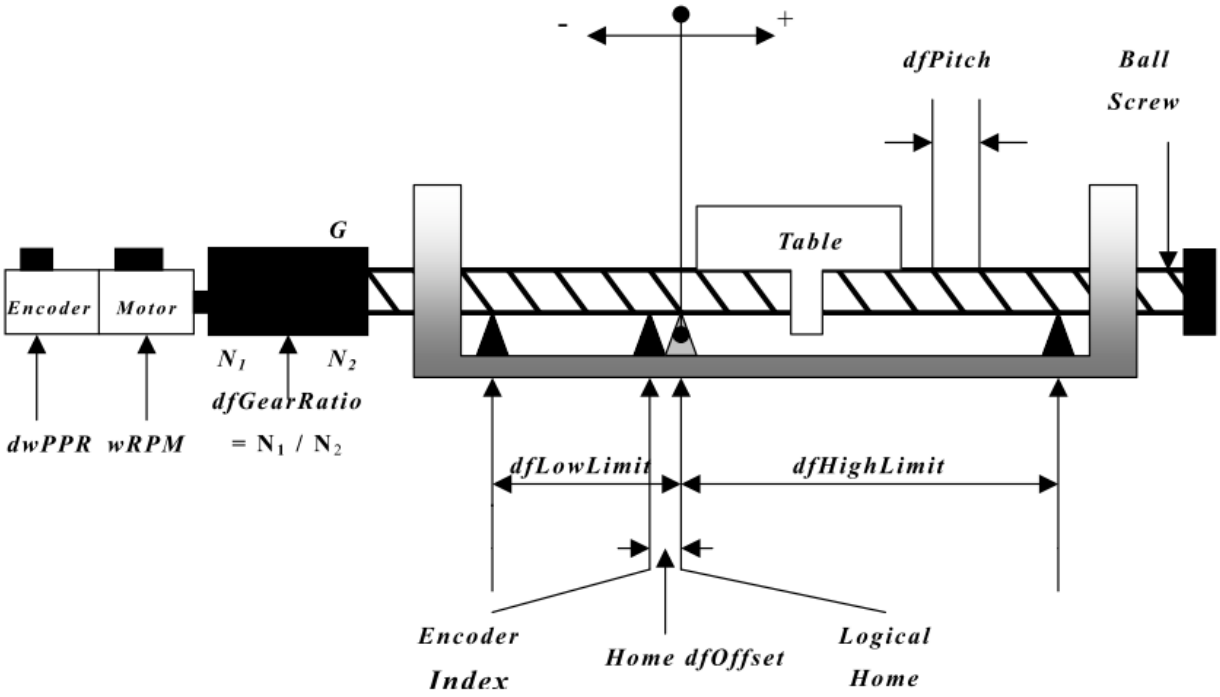
如果沒有配置減速箱此值應設為 1。

dfHighLimit：軟體過行程保護上限

此值為相對於邏輯原點正方向的最大位移量，單位為 mm 或 inch。

dfLowLimit：軟體過行程保護下限

此值為相對於邏輯原點負方向的最大位移量，單位為 mm 或 inch。



圖一 機構參數

3. 初始化與關閉運動控制函式庫

相關函式

MCC_InitSystem()
MCC_CloseSystem()
MCC_GetMotionStatus()

範例程式

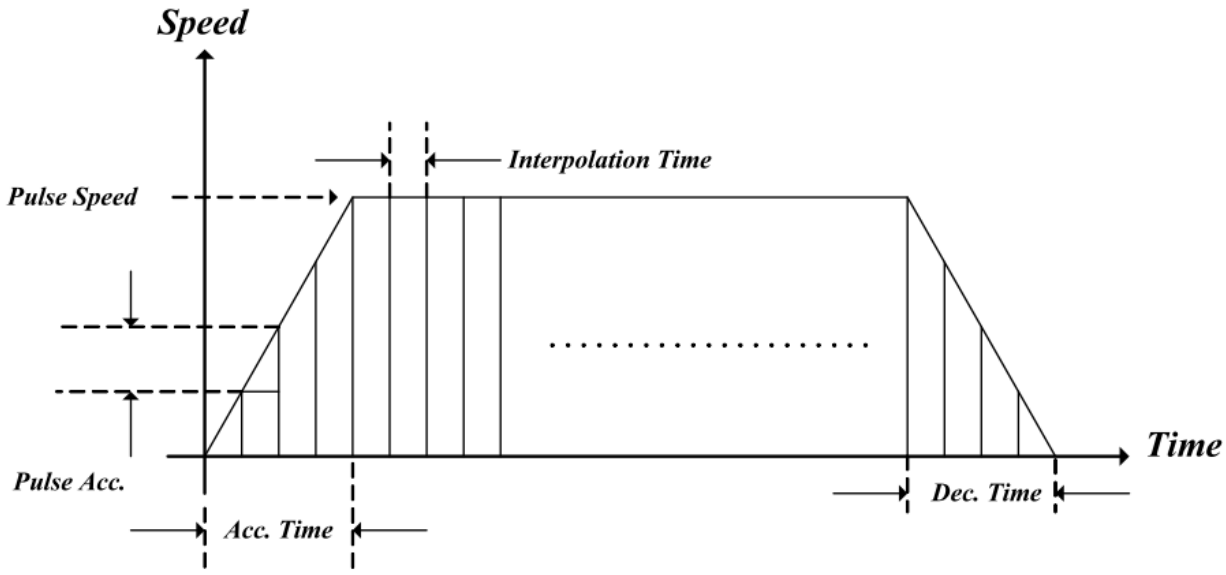
InitSys.cpp (VC++)
InitSys.vb (VB)

說明

在完成運動群組與機構參數設定之後，使用 MCC_InitSystem()啟動運動控制函式庫(MCCL)。

插值時間(Interpolation Time)

插值時間是指距離下一個插值點的時間。



圖二 插值時間

範例：

```
SYS_CARD_CONFIG    stCardConfig[MAX_CARD_NUM];

stCardConfig[CARD_INDEX].wCardType       = wCardType;
//      Start motion library
nRet = MCC_InitSystem(INTERPOLATION_TIME, stCardConfig, 1);
//interpolation time set to 10 ms; hardware parameter, use 1 PCI-4P card only.
if (nRet == NO_ERR)//    Initialization succeed
{
      /*
      User can do other initialization here, for example set feeding speed, set displacement unit.
      */
}
```

使用 MCC_CloseSystem()來關閉運動控制函式庫(MCCL)和驅動函式庫。下面提供兩種關閉系統的方法：

1. 完成所有的運動指令後關閉系統

檢查系統是否在停止狀態，若 MCC_GetMotionStatus()的回傳值為 GMS_STOP 則系統在停止狀態。

```
while ((nRet = MCC_GetMotionStatus(GROUP_INDEX)) != 1)
{
      MCC_TimeDelay(1);
                  //   Sleep 1ms
                  //   Using ”while” command, it is necessary to avoid system’s being lockup.
                  //   Call MCC_TimeDelay() to release CPU.
}

MCC_CloseSystem();//    Close MCCL and driver function library.
```

2. 直接關閉運動控制函式庫

呼叫 MCC_CloseSystem()，系統將立即停止運動。

4. 設定系統狀態

相關函式

MCC_SetUnit()
MCC_GetUnit()
MCC_SetAbsolute()
MCC_SetIncrease()
MCC_GetCoordType()
MCC_SetAccType()
MCC_GetAccType()
MCC_SetDecType()
MCC_GetDecType()
MCC_SetPtPAccType()
MCC_GetPtPAccType()
MCC_SetPtPDecType()
MCC_GetPtPDecType()
MCC_SetServoOn()
MCC_SetServoOff()
MCC_EnablePosReady()
MCC_DisablePosReady()

範例程式

SetStatus.cpp (VC++)
SetStatus.vb (VB)

說明

若沒有特別設定系統狀態，將使用系統預設狀態，系統預設狀態可參考”附錄三 系統初始設定”。

範例：

```
MCC_SetUnit(1, GROUP_INDEX);          //  Use mm as the unit of displacement.

MCC_SetAbsolute(GROUP_INDEX);         //  Use absolute coordinate type to show the position of each axis.

//  Use ’T ‘curve as the acceleration type of line, arc, circular motion.
MCC_SetAccType('T', GROUP_INDEX);
```

```
//Use ‘S’ curve as the deceleration type of line, arc, circular motion.
MCC_SetDecType('S', GROUP_INDEX);

//      Use ‘T’ curve as the acceleration type of point to point motion.
MCC_SetPtPAccType('T', 'T', 'T', 'T', 'T', 'T', GROUP_INDEX);

//  Use ‘S’ curve as the deceleration type of point to point motion.
MCC_SetPtPDecType('S', 'S', 'S', 'S', 'S', 'S', GROUP_INDEX);

MCC_SetServoOn(0, CARD_INDEX);//    Set axis 0 servo on

//    Enable position ready output connection function.
MCC_EnablePosReady(CARD_INDEX);
```

最大進給速度不能大於(*wRPM* / 60) × *dfPitch* / *dfGearRatio*，若設定的進給速度大於此值，運動控制函式庫(MCCL)會自動使用(*wRPM* / 60) × *dfPitch* / *dfGearRatio* 為各軸的最大進給速度來規劃軌跡。其中 *wRPM*、*dfPitch*、*dfGearRatio* 為各軸的機構參數，請參考”圖一 機構參數”。

5. 設定進給速度

相關函式

MCC_SetFeedSpeed()
MCC_GetFeedSpeed()
MCC_SetPtPSpeed()
MCC_GetPtPSpeed()

範例程式

SetSpeed.cpp (VC++)
SetSpeed.vb (VB)

說明

在進行直線、圓弧、圓運動前要先設定進給速度。特別注意，進給速度不能大於MCC_SetSysMaxSpeed()所設定的值。

MCC_SetFeedSpeed()所設定的進給速度會使用在直線、圓弧、圓和螺線運動(一般運動)。當呼叫函式 MCC_SetFeedSpeed(20, GROUP_INDEX)，進給速度可能為 20 mm/sec 或 20 inch/sec，須視所設定的單位而定。

使用 MCC_SetPtPSpeed()來設定點對點運動的進給速度。MCC_SetPtPSpeed()設定的不是最大進給速度，而是各軸最大進給速度的比例。此值必須大於 0 且小於等於 100。舉例來說，當設定 MCC_SetPtPSpeed(50, GROUP_INDEX)，則各軸點對點運動的最大速度為((*wRPM* / 60) × *dfPitch* / *dfGearRatio*) x 50%。

6. 直線、圓弧、圓運動

相關函式

MCC_SetAbsolute()
MCC_SetFeedSpeed()
MCC_Line()
MCC_ArcXY()
MCC_CircleXY()

範例程式

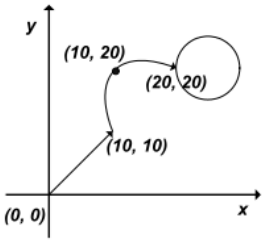
GeneralMotion.cpp (VC++)
GeneralMotion.vb (VB)

說明

在設定完機構參數、啟動系統、設定最大進給速度和設定各軸進給速度後可以開始執行直線、圓弧和圓運動。特別注意，設定圓弧運動時參數須要在合理的範圍。
範例：

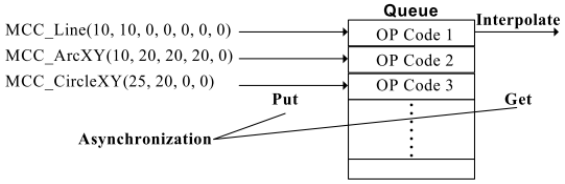
```
MCC_SetAbsolute(GROUP_INDEX);     //    Use absolute coordinate to express the position of each axis

MCC_SetFeedSpeed(5, GROUP_INDEX);//    Set feeding speed
MCC_Line(10, 10, 0, 0, 0, 0, GROUP_INDEX);

nRet = MCC_ArcXY(10, 20, 20, 20, GROUP_INDEX);// Notice that start point, reference point and end point should not be colinear

if (nRet < NO_ERR)
/*
Use returned value to identicfy the reason of error, if parameter error then send back value PARAMETER_ERR is returned.
*/
MCC_CircleXY(25, 20, 0, GROUP_INDEX);
```

使用函式傳回值來判斷錯誤原因。請參考"附錄一 函式傳回值"的說明。運動軌跡：



圖三 運動軌跡

呼叫運動函式後會將相關的運動命令先存放(Put)在各運動群組(Group)專屬的運動命令緩衝區(Motion Command Queue)中，而非立即執行。然後運動控制函式庫使用先入先出(First In First Out, FIFO)的方式，從緩衝區中抓取(Get)運動命令執行。但這兩個行為並非順序與同步動作，也就是說並不需要等到運動命令執行完成，即可將新的運動命令送到運動命令衝區中。



圖四 運動命令緩衝區

各運動群組之運動命令緩衝區預設可儲存 10000 筆命令，可利用 MCC_SetCmdQueueSize()改變其大小；注意，改變緩衝區大小的動作必須在無庫存命令時始可執行。

7. 點對點運動

相關函式

MCC_SetAbsolute()
MCC_SetPtPSpeed()
MCC_PtP()

範例程式

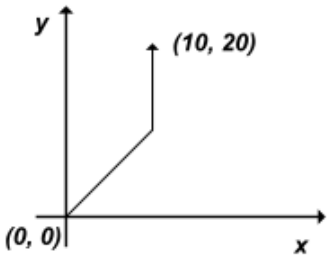
PtpMotion.cpp (VC++)
PtpMotion.vb (VB)

說明

在設定完機構參數、啟動系統、設定最大進給速度和說定各軸進給速度後可以開始執行點對點運動。
範例：

```
MCC_SetAbsolute(GROUP_INDEX);       //    Use absolute coordinate to express the position of each axis

MCC_SetFeedSpeed(10, GROUP_INDEX);//    set feeding speed

//Motion with 20% of the maximum speed for each axis i.e. ((wRPM / 60 ) × dfPitch / dfGearRatio) × 20%
MCC_SetPtPSpeed(20, GROUP_INDEX);

//Move to (10,20), however motion of each axis start at same time and could arrive at target at different time.
MCC_PtP(10, 20, 0, 0, 0, 0, GROUP_INDEX);
```

針對點對點運動，各軸有獨立的運動軌跡規劃。點對點運動中，各軸同時開始，但不一定會同時結束，端看所給定的參數而定。此差異為點對點運動與一般運動主要不同處。一般運動中，各軸會同時開始並在同一時間結束。點對點運動運動軌跡(假定各軸的最大速度相同)：



圖五 點對點運動軌跡

8. JOG 運動

相關函式

MCC_SetUnit()
MCC_JogPulse()
MCC_JogSpace()
MCC_JogConti()

範例程式

JogMotion.cpp (VC++)
JogMotion.vb (VB)

說明

MCC_JogPulse()要求特定軸移動指定的 pulse 量，可輸入範圍為-2048~2048 個 pulse。MCC_JogSpace()要求特定軸依指定的進給速度比例移動指定的位移量，單位為 mm 或 inch。MCC_JogConti()則要求特定軸依指定的進給速度比例與方向，移動到使用者設定的有效工作區間邊界才停止。在使用以上函式時，運動狀態須要處於靜止狀態，也就是 MCC_GetMotionStatus()的傳回值應為 GMS_STOP。

範例：

```
MCC_SetUnit(1, GROUP_INDEX);            //  Use mm as the unit of displacement.

MCC_JogPulse(100, 0, GROUP_INDEX);      //  Axis 0 move 100 pulses

// Move axis 0 a distance with speed of (wRPM × dfPitch/dfGearRation) ×10%
MCC_JogSpace(-1, 10, 0, GROUP_INDEX);

// Move axis 0 continuously with speed of (wRPM × dfPitch/dfGearRation) ×10% to right limit.
MCC_JogConti(1, 10, 0, GROUP_INDEX);
```

9. 讀取速度、位置、指令位置

相關函式

MCC_GetCurFeedSpeed()
MCC_GetSpeed()
MCC_GetCurPos()
MCC_GetPulsePos()
MCC_GetCurCommand()
MCC_GetCommandCount()

範例程式

GetStatus.cpp (VC++)
GetStatus.vb (VB)

說明

使用MCC_GetCurFeedSpeed()獲得目前實際的切線速度，MCC_GetSpeed()則可以獲得目前各軸實際的進給速度。MCC_GetCurPos()可以獲得目前的命令位置，單位為mm或inch。MCC_GetPulsePos()可獲得目前已從控制卡輸出的pulse量。兩者的關係為pulse = (UU)(*dfGearRatio* / *dfPitch*) x *dwPPR*，其中UU為User Unit即使用者在設定機構參數時，自行選定之長度單位(mm或inch)。使用MCC_GetCurPos()和MCC_GetPulsePos()指令前需先呼叫MCC_CreateGroup()建立運動群組，讀到的傳回值才有意義。
範例：

```
double dfCurSpeed;
double dfCurPosX, dfCurPosY, dfCurPosZ, dfCurPosU, dfCurPosV, dfCurPosW;
double dfCurSpeedX, dfCurSpeedY, dfCurSpeedZ, dfCurSpeedU, dfCurSpeedV, dfCurSpeedW;
long lCurPulseX, lCurPulseY, lCurPulseZ, lCurPulseU, lCurPulseV, lCurPulseW;

dfCurSpeed = MCC_GetCurFeedSpeed(GROUP_INDEX);//Get current feeding speed.

//Get feeding speeds of each axis.
MCC_GetSpeed(&dfCurSpeedX, &dfCurSpeedY, &dfCurSpeedZ, &dfCurSpeedU, &dfCurSpeedV,
&dfCurSpeedW, GROUP_INDEX);

//Get Cartesian coordinates of current position of each axis
MCC_GetCurPos(&dfCurPosX, &dfCurPosY, &dfCurPosZ, &dfCurPosU, &dfCurPosV, &dfCurPosW,
```

```
GROUP_INDEX);

// Get motor coordinates of current position of each axis
MCC_GetPulsePos(&lCurPulseX, &lCurPulseY, &lCurPulseZ, &lCurPulseU, &lCurPulseV,
&lCurPulseW, GROUP_INDEX);
```

使用MCC_GetCurCommand()可以獲得目前正在執行的運動命令相關的資訊，包括運動命令之類型、運動命令編碼、要求的進給速度與目的點位置等。使用MCC_GetCommandCount()可以獲得目前尚未被執行的運動命令之庫存量，此庫存量不包括正在執行的運動命令。

範例：

```
COMMAND_INFO stCommandInfo = {0};
int nCommandCount = 0;

//   Get motion command being executed
MCC_GetCurCommand(   &stCommandInfo, GROUP_INDEX);

 /*
      stCommandInfo representive meanings is as follows :

      stCommandInfo.nType;              motion commmadn type
                                              0: Pont to Point
                                              1: Line
                                              2: Clockwise Arc/Circle
                                              3: Counterclockwise Arc/Circle
                                              6: Delay
                                              7: Enable Blend
                                              8: Disable Blend
                                              9: Enable In-Position
                                             10: Disable In-Position

      stCommandInfo.nCommandIndex;    Motion command encode
      stCommandInfo. DfFeedSpeed      feed spped
      stCommandInfo. dfPos[]          Target position coordinate
*/

//   Get the number of saved motion command
MCC_GetCommandCount(&nCommandCount, GROUP_INDEX);
```

10. 運動暫停、持續、棄置

相關函式

MCC_HoldMotion()
MCC_ContiMotion()
MCC_AbortMotionEx()

範例程式

CtrlMotion.cpp (VC++)
CtrlMotion.vb (VB)

說明

可使用 MCC_AbortMotionEx()捨棄目前正在執行中與庫存的所有運動命令，也可以使用 MCC_HoldMotion()暫停執行中的運動命令，此時將以等減速的方式停止運動，待使用 MCC_ContiMotion()後，才繼續執行該筆命令尚未完成的部分，但此時也可以使用 MCC_AbortMotionEx()，捨棄尚未完成的部分。

若呼叫 MCC_AbortMotionEx()時已經沒有執行中與庫存的運動命令，則傳回 ABORT_ILLEGAL_ERR；若呼叫 MCC_HoldMotion()時已經沒有執行中與庫存的運動命令，則傳回 HOLD_ILLEGAL_ERR；若呼叫 MCC_ContiMotion()時 MCC_HoldMotion()還沒有回傳成功，則傳回 CONTI_ILLEGAL_ERR。使用者可以利用傳回值來判斷是否正確使用函式。

範例：

```
int        nRet;
nRet = MCC_HoldMotion(GROUP_INDEX);
if (nRet != NO_ERR)    // Use the function not at the right time.
      printf("\n\nErroneous return value : %d\n\n", nRet);

nRet = MCC_ContiMotion(GROUP_INDEX);
if (nRet != NO_ERR)    // Use the function not at the right time.
      printf("\n\nErroneous return value : %d\n\n", nRet);

nRet = MCC_AbortMotion(GROUP_INDEX);
if (nRet != NO_ERR)    // Use the function not at the right time.
      printf("\n\nErroneous return value : %d\n\n", nRet);
```

11. 運動狀態

相關函式

MCC_GetMotionStatus()

範例程式

MotionFinished.cpp (VC++)
MotionFinished.vb (VB)

說明

利用呼叫 MCC_GetMotionStatus()所獲得的傳回值可以判斷目前的運動狀態；傳回值若為 GMS_RUNNING，表示系統正處於運動狀態；傳回值若為 GMS_STOP，表示系統處於停止狀態，已無任何未執行的庫存命令；傳回值若為 GMS_HOLD，表示系統因使用 MCC_HoldMotion()暫停中；傳回值若為 GMS_DELAYING，表示系統因使用 MCC_DelayMotion()目前正在延遲中。
範例：

```
int   nStatus;

MCC_Line(20, 20, 0, 0, 0, 0, GROUP_INDEX);

while (MCC_GetMotionStatus(GROUP_INDEX) != 1)//   Wait for motion finish
{
  nStatus = MCC_GetMotionStatus(GROUP_INDEX);
  printf("Motion nStatus : %d       \r", dfCurPosX, dfCurPosY, nStatus);
}

nStatus = MCC_GetMotionStatus(GROUP_INDEX);
//   nStatus   :   0      in motion
//   nStatus   :   1      machine is in stop status and there is no command in the motion buffer
//   nStatus   :   2      machine is in hold status and there are still unfinished commands in the buffer
printf("Motion nStatus : %d       \r", dfCurPosX, dfCurPosY, nStatus);
```

12. 速度強制功能(speed override)

相關函式

MCC_OverrideSpeed()
MCC_GetOverrideRate()
MCC_OverridePtPSpeed()
MCC_GetPtPOverrideRate()

範例程式

OverSpeed.cpp (VC++)
OverrideSpeed.vb (VB)

說明

如需在運動中動態變更進給速度時，可使用速度強制功能，此功能可將執行中運動命令的速度加速到要求的速度值，或由目前的速度減速到要求的速度值。

使用 MCC_OverrideSpeed()指定速度比例，即時強制變更切線速度。速度比例的定義為：

速度比例 = 要求變更後的進給速度 / 原進給速度 × 100

原進給速度是指使用 MCC_SetFeedSpeed()或 MCC_SetPtPSpeed()所設定的速度。

注意：使用 MCC_OverrideSpeed()後，將影響往後全部運動的速度，而不只是影響執行中的運動。

範例：

```
printf("1. Press ESCAPE key to quit\n");
printf("2. Press 'L' key to execute line motion\n");
printf("3. Press 'O' key to execute line over-speed command\n\n");

while (1)
{
  if (_kbhit())
  {
        cKey = _getch();
        iif (cKey == 'l' || cKey == 'L')
              MCC_Line(200, 0, 0, 0, 0, 0, GROUP_INDEX);
        else if (cKey == 'o' || cKey == 'O')
              MCC_OverrideSpeed(150, GROUP_INDEX);//    Set over speed ratio,i.e. current speed
```

```
    will become 20 * 150 % = 30 mm/sec
     }
     MCC_GetSpeed(&dfCurSpeedX, &dfCurSpeedY, &dfCurSpeedZ, &dfCurSpeedU, &dfCurSpeedV,
    &dfCurSpeedW, GROUP_INDEX); //    Get current feeding speeds of each axis
     nRate = MCC_GetOverSpeed(GROUP_INDEX);
     printf("Velocity : x = %6.2f       Over Speed : %d       \r", dfCurSpeedX, nRate);
    }
```

13. 軟體極限、軟體極限開關

相關函式

MCC_SetOverTravelCheck()
MCC_GetOverTravelCheck()
MCC_EnableLimitSwitchCheck()
MCC_DisableLimitSwitchCheck()
MCC_GetLimitSwitchStatus()

範例程式

CheckOT.cpp (VC++)
CheckOT.vb (VB)

說明

使用 MCC_SetOverTravelCheck()開啟此項功能後，運動控制函式庫(MCCL)在計算完每一個插值點時會檢查此插值點是否會超出各軸的有效工作區間；若判斷已超出工作區間時，則不再對運動控制卡送出命令。使用者可使用 MCC_GetErrorCode()查詢錯誤代碼(參考"附錄二 錯誤代碼")獲知是否已超出各軸的有效工作區間。

當使用 MCC_EnableLimitSwitchCheck()將極限開關功能開啟時，在碰觸到該軸運動方向的極限開關時，將會停止運動。利用呼叫 MCC_GetErrorCode()來獲知系統是否因碰觸到極限開關而無法運動，此時內部已產生錯誤記錄(代碼 0xF701~0xF703 分別代表 X~Z 軸碰觸極限開關)。

當發生錯誤時，將不再執行運動命令。此時使用者需自行使用 MCC_GetErrorCode()判斷錯誤原因並排除之，而後使用 MCC_ClearError()清除錯誤紀錄，使系統恢復至正常狀態。

範例：

```
WORD wLimitSwitchStatus;
int   nErr;
int   nOT0, nOT1, nOT2, nOT3, nOT4, nOT5;
MCC_GetOverTravelCheck(&nOT0, &nOT1, &nOT2, &nOT3, &nOT4, &nOT5, GROUP_INDEX);

nErr = MCC_GetErrorCode(GROUP_INDEX);//   Get occurred eroor code
//   The meanings of nErr (error code) please refer to the Motion Library Reference Manual
//System will get current error code regularly to check if the machine operation has any error
MCC_GetLimitSwitchStatus(&wLimitSwitchStatus, 1, 0, 0);

printf("OT check : %d   Error Code : %x Limit Switch : %d \r", nOT0, nErr, wLimitSwitchStatus);
```

14. 速度平滑運動

相關函式

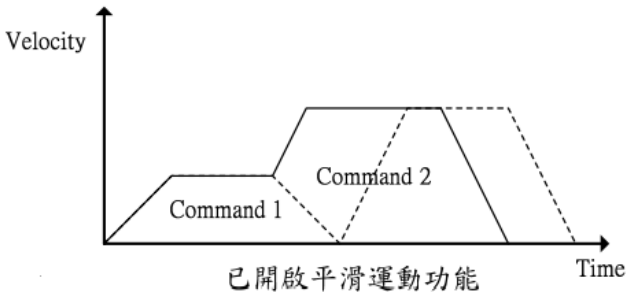
MCC_EnableBlend()
MCC_DisableBlend()
MCC_CheckBlend()

範例程式

SetBlend.cpp (VC++)
SetBlend.vb (VB)

說明

可以使用 MCC_EnableBlend()開啟平滑運動功能，此項功能可滿足不同運動命令間的等速段達到速度平滑連續的要求，也就是在完成前一段運動命令時速度不需減速到停，可直接加速或減速到下一段運動命令要求的速度，參考圖三。平滑運動功能包括直線-直線、直線-圓弧、圓弧-圓弧間的平滑運動。



圖六 平滑運動

呼叫 MCC_EnableBlend()開啟平滑運動功能。呼叫此函式後，以連續路徑方式進行軌跡規劃。呼叫 MCC_DisableBlend()關閉連續運動功能。MCC_CheckBlend()檢查是否開啟連續運動功能。

15. 讀取、清除錯誤狀態

相關函式

MCC_GetErrorCode()
MCC_ClearError()

範例程式

ErrorStatus.cpp (VC++)
ErrorStatus.vb (VB)

說明

呼叫 MCC_GetErrorCode()讀取目前錯誤記錄，檢查系統運作時是否發生錯誤。系統運作中使用者應隨時呼叫此函式確認系統工作正常，若發現錯誤記錄產生，則須採取相對應之錯誤回復處理。

在系統運作發生錯誤後，若已排除這些錯誤，仍必須呼叫 MCC_ClearError()來清除系統內的錯誤記錄，否則系統仍無法正常運作。

範例：

```
int    nErr;
nErr = MCC_GetErrorCode(GROUP_INDEX);//    Get occurred eroor code
//    To understand the meanings of nErr (error code) please refer to the appendix

printf("Error code : %x       \r", nErr);

if (nErr != NO_ERR)
{
  MCC_ClearError(GROUP_INDEX);
}
```

16. 歸原點

相關函式

MCC_SetHomeConfig()
MCC_Home()
MCC_GetGoHomeStatus()
MCC_AbortGoHome()
MCC_GetHomeSensorStatus()

範例程式

GoHome.cpp (VC++)
GoHome.vb (VB)

說明

運動控制函式庫(MCCL)利用原點復歸參數來定義原點復歸動作，包括使用模式、原點復歸運動方向、原點開關(HomeSensor)的配線方式、編碼器 INDEX 訊號計數次數、加減速度設定等。

在原點復歸過程中可以使用 MCC_AbortGoHome()停止復歸動作；也可以利用 MCC_GetGoHomeStatus()的函式傳回值獲知原點復歸的動作是否已經完成，若傳回值為 1 表示原點復歸動作已經完成，若為 0 表示原點復歸的動作尚在進行中。

範例：

```
SYS_HOME_CONFIG    stHome;
int   nStatus;

stHome.wMode              = 3;
stHome.wSensorMode        = 0;
stHome.wDirection         = 1;
stHome.dfOffset           = 0;
stHome.nIndexCount        = 2;
stHome.dfAccTime          = 300;
stHome.dfDecTime          = 300;
stHome.dfHighSpeed        = 10;
stHome.dfLowSpeed         = 2;

MCC_SetHomeConfig(&stHome, wChannel, CARD_INDEX);
MCC_Home(0, 0xff, 0xff, 0xff, 0xff, 0xff,       // Order for going home,   0xff means axis won't do
```

```
homing
                CARD_INDEX);

MCC_AbortGoHome();//    Abort homing

//    Check if finished homing procedure, return value nStatus = 1 means finished
nStatus = MCC_GetGoHomeStatus();
```

17. 加、減速設定

相關函式

MCC_SetAccTime()
MCC_GetAccTime()
MCC_SetDecTime()
MCC_GetDecTime()
MCC_SetPtPAccTime()
MCC_GetPtPAccTime()
MCC_SetPtPDecTime()
MCC_GetPtPDecTime()

範例程式

AccStep.cpp (VC++)
AccStep.vb (VB)

說明

可以設定一般運動與點對點運動加速到穩定速度所需的時間，也可以設定由穩定速度減速到停止運動所需的時間。使用 MCC_SetAccTime()與 MCC_SetDecTime()設定直線、圓弧、圓、螺線運動所需的加、減速時間；使用 MCC_SetPtPAccTime()與 MCC_SetPtPDecTime()設定點對點運動所需的加、減速時間。通常在給定較快的進給速度時會要求較長的加速度時間，因此 MCC_SetAccTime()與 MCC_SetDecTime()通常會與 MCC_SetFeedSpeed()搭配使用；同樣的，MCC_SetPtPAccTime()與 MCC_SetPtPDecTime()通常會與 MCC_SetPtPSpeed()搭配使用。
範例：

```
double dfAcc, dfSpeed, dfTime;

dfAcc = 100;          // acceleration 10mm/s^2
dfSpeed = 60;         // target speed 60mm/s

dfTime = dfSpeed / dfAcc * 1000; // ms

MCC_SetAccTime(dfTime, GROUP_INDEX);//    Set acceleration to 600 ms

MCC_SetDecTime(dfTime, GROUP_INDEX);//    Set deceleration to 600 ms
MCC_Line(30, 30, 0, 0, 0, 0, GROUP_INDEX);
```

附錄一 函式傳回值

傳回值定義	數值	說明
NO_ERR	0	函式呼叫成功
INITIAL_MOTION_ERR	-1	系統尚未啟動，請再次呼叫 MCC_InitSystem()
COMMAND_BUFFER_FULL_ERR	-2	運動命令緩衝區已滿，此時無法接受此筆命令。
COMMAND_NOTACCEPTED_ERR	-3	系統處於忙碌狀態，此時無法接受此筆命令。
COMMAND_NOTFINISHED_ERR	-4	執行中的運動命令尚未完成，此時無法接受此筆命令。
PARAMETER_ERR	-5	呼叫函式時所傳入的參數格式錯誤
GROUP_PARAMETER_ERR	-6	Group 參數給定錯誤，所指定為無效的 Group
FEED_RATE_ERR	-7	進給速度未設定或設定錯誤，請重新呼叫 MCC_SetFeedSpeed()函式
HOME_COMMAND_NOTCALLED_ERR	-10	目前並不在原點復歸模式下
HOLD_ILLEGAL_ERR	-11	不適當時機發出暫停(Hold)命令
CONTI_ILLEGAL_ERR	-12	不適當時機發出繼續(Continue)命令
ABORT_ILLEGAL_ERR	-13	不適當時機使用棄置(Abort)命令
RUN_TIME_ERR	-14	執行時期產生錯誤，利用呼叫 MCC_GetErrorCode()所獲得的錯誤訊息代碼可了解錯誤的內容
ABORT_NOT_FINISH_ERR	-15	命令棄置動作尚未完成
GROUP_RAN_OUT_ERR	-16	已無多餘 Group 可使用

表一 函式傳回值

附錄二 錯誤代碼

錯誤代碼	說明
0xF101	尚未初使化運動控制函式庫
0xF104	在使用圓弧運動命令時給定的參數不合理
0xF203	進給速度太快，超過每個插值時間內允許輸出的 Pulse 數
0xF204	進給加速度太快，超過每個插值時間內允許輸出的 Pulse 增量數
0xF301	X 軸座標值超出機構參數設定的工作範圍
0xF302	Y 軸座標值超出機構參數設定的工作範圍
0xF303	Z 軸座標值超出機構參數設定的工作範圍
0xF304	U 軸座標值超出機構參數設定的工作範圍
0xF401	運動命令在執行間發生錯誤
0xF501	定位確認錯誤
0xF701	X 軸碰觸硬體極限開關
0xF702	Y 軸碰觸硬體極限開關
0xF703	Z 軸碰觸硬體極限開關
0xF704	U 軸碰觸硬體極限開關

表二 錯誤代碼

附錄三 系統初始設定

當使用者呼叫 MCC_InitSystem()後，系統將回到初始設定，使用者需依據實際需求重新設定參數。

初始設定內容	初始設定
命令緩衝區大小	10000 筆命令
最大進給速度	100
系統座標型態	絕對座標
軟體過行程檢查	未開啟
硬體極限開關檢查	未開啟
位置控制閉迴路比例增益	64
進行直線、圓弧、圓、螺線運動時各軸使用的加、減速型式	S 形曲線
進行直線、圓弧、圓、螺線運動時各軸使用的加、減速時間	300 ms
進行直線、圓弧、圓、螺線運動時使用的進給速度	1
進行對點對運動時各軸使用的速度比例	1
定位確認最大檢查時間	1000 ms
定位確認持續時間	100 ms
定位確認容許誤差範圍	無限大
定位確認功能	未開啟
連續運動功能	未開啟

表三 系統初始設定