HIWIN PCI-4P Motion Library
Example Manual

Mar. 3. 2005

Contents

1. Introduction

The examples provided are for console mode, user may integrate these examples into his own application program. MCCL can support up to 12 PCI-4P cards and 72 groups. For the purpose of increasing the readability of these examples, it is assumed to use 1 PCI-4P card only, in other words the motion control card used in the example program, their CARD_INDEX are defined to 0. In the example program we defined 1 group only, it is group 0, thus GROUP_INDEX is defined to 0.

2. Group, mechanism parameter setting

Related functions

MCC_SetSysMaxSpeed()
MCC_GetSysMaxSpeed()
MCC_SetGroupConfig()
MCC_SetMachParam()
MCC_GetMachParam()
MCC_UpdateMachParam()

Example program

InitSys.cpp

Explanation

This example explains the setup process of group parameter and mechanism parameter. Use MCC_SetSysMaxSpeed() to set the upper limit of feed speed first ,then use MCC_SetMachParam() to set mechanism parameter of each axis. Finally use MCC_SetGroupConfig() to set group parameter.

The more detailed explanation of related group parameter and mechanism parameter, please refer to "**PCI-4P motion library user's manual**".

3. Initialize and close motion library

Related functions

MCC_InitSystem()
MCC_CloseSystem()
MCC_GetMotionStatus()

Example program

InitSys.cpp

Explanation

After finish setting group parameter and mechanism parameter, have to use MCC_InitSystem() to start the motion library. The parameters needed, please refer to "**PCI-4P motion library user's manual**". Used example is as follows:

```
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 PCI4P card only.
if (nRet == NO_ERR)//    Initialization succeed.
    {
    /*
     User can do other initialization here, for example set feeding speed, set
     displacement unit.
    */
    }
```

MCC_CloseSystem() is used to close MCCL and driver function library. There are two ways to close the system:

A. Finish executing all motion command then close system.

Check if system is in stop condition, if the returned value from MCC_GetMotionStatus() is 1, then is in stop condition.

```
while ((nRet = MCC_GetMotionStatus(GROUP_INDEX)) != 1)
    {
    MCC_TimeDelay(1);     //  Sleep 1 ms
                          //  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.
```

B. Close motion library directly

Call MCC_CloseSystem() only, system will stop operation right away.

4. Set system status

Related functions

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_GePtPDecType()
MCC_SetServoOn()
MCC_SetServoOff()
MCC_EnablePosReady()
MCC_DisablePosReady()

Example program

SetStatus.cpp

Explanation

If system's status is not set specially, system will operate according to it's preset status, please refer to " **PCI-4P motion library user's manual** ". Used example is shown as follows:

```
MCC_SetUnit(_MM_, 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', 0, 0, GROUP_INDEX);

//      Use ‘S’ curve as the deceleration type of point to point motion.
MCC_SetPtpDecType('S', 'S', 'S', 'S', 0, 0, GROUP_INDEX);

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

//      Set on position ready output
MCC_EnablePosReady(CARD_INDEX);
```

The maximum value of feeding speed can not be greater than RPM × pitch / gear_ratio, otherwise MCCL will use RPM × pitch / gear_ratio as the maximum value for each axis’s feeding speed automatically in it’s trajectory planning. RPM , pitch , gear_ratio are each axis’s mechanism parameters.

5. Set feeding speed

Related functions

MCC_SetFeedSpeed()
MCC_GetFeedSpeed()
MCC_SetPtPSpeed()
MCC_GetPtpSpeed()

Example program

SetSpeed.cpp

Explanation

It is necessary set feeding speed before doing any motion. Special notice must be taken that the feeding speed can not be greater than the value set by MCC_SetSysMaxSpeed().

Use MCC_SetFeedSpeed() to set the feeding speed of line, arc, circular motion. For example when call MCC_SetFeedSpeed (20, GROUP_INDEX), means feeding speed is 20 mm/sec or 20 inch /sec, depends on the unit used.

For point to point motion instead of calling MCC_SetFeedSpeed(), use MCC_SetPtPSpeed() to set speed, but it's values are set as percent of maximum speed for each axis., range is 0 ~ 100. For example when execute MCC_SetPtPSpeed (50, GROUP_INDEX), the speed for each axis of point to point motion is (wRPM × dfPitch /dfGearRatio) × 50%.

6. Line, arc, circular motion

Related functions

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

Example program

GeneralMotion.cpp

Explanation

After finish setting mechanism parameter, starting system, setting maximum feeding speed, and setting feed speed it becomes ready to execute line, arc, circular motion. When executing arc function, there is a special notice that the parameters needed are within the reasonable range. Example is as follows:

```
MCC_SetAbsolute(GROUP_INDEX);          // Use absolute coordinate to express
                                          the position of each axis
MCC_SetFeedSpeed(20, GROUP_INDEX); //  Set feeding speed
MCC_Line(10, 10, 0, 0, 0, 0, GROUP_INDEX);
                          // Notice that start point, reference point and end point should not
                                                be colinear
 nRet = MCC_ArcXY(10, 20, 20, 20, GROUP_INDEX);
if (nRet != NO_ERR)
    {
    /*
    Use returned value to identify the reason of error, if parameter error then send back
```

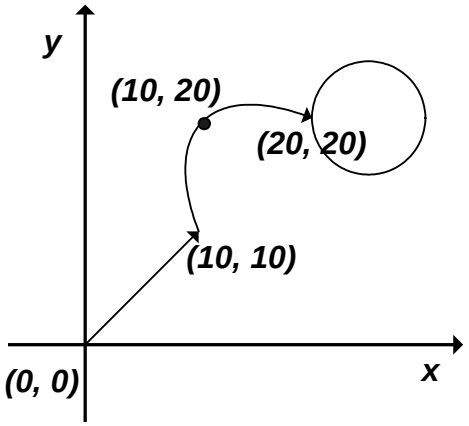
value PARAMETER_ERR is returned.

*/

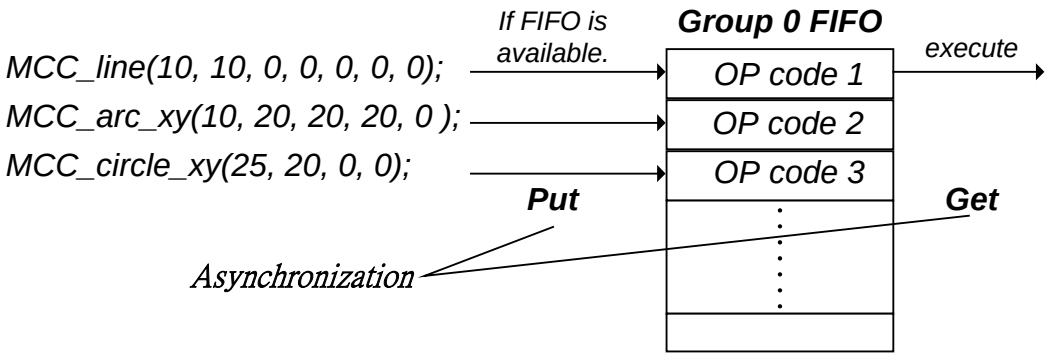
}

MCC_CircleXY(25, 20, 0, GROUP_INDEX);

Use returned value to identify the reason of error. Regarding the meaning of returned error, please refer to "**PCI-4P motion library user's manual"**. Trajectory of motion is shown as follows:



The process of executing motion function is that motion function put op code in each FIFO of each group first, then MCCL get the commands from FIFO's of different groups simultaneously and execute sequentially. It doesn't have to wait for one motion to finish to execute another motion. If FIFO is full, the returned value of function is COMMAND_BUFFER_FULL_ERR,the size of each FIFO buffer is 10000. The figure shown below is the operation of FIFO for a certain group.



Thus, commands of the same group will be executed sequentially; but commands of different groups can be executed concurrently.

7. Point to point motion

Related functions

MCC_SetAbsolute()
MCC_SetPtPSpeed()
MCC_PtP();

Example program

PtPMotion.cpp

Explanation

After finish setting mechanism parameter, starting system, setting maximum feeding speed, and setting feeding speed, it is possible to execute point to point motion. Used function example is as follows:

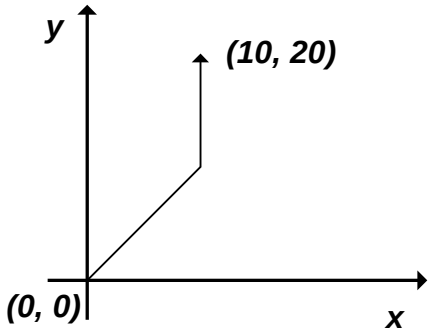
```
MCC_SetAbsolute(GROUP_INDEX);                    // Use absolute coordinate to express
                                                    the position of each axis
MCC_SetFeedSpeed(20, GROUP_INDEX); //  Set feeding speed.

                    //  Motion with 20% of the maximum speed for each axis i.e. (RPM
                        × pitch / gear_ratio) × 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);
```

For point to point motion, each axis has independent motion profile. They

won't reach the target position at the same time though they start simultaneously, this is different from general motion. General motion are synchronized, it starts and ends a motion at the same time. The figure below is a point to point motion trajectory, assuming the maximum speed are the same for each axis.



8. JOG motion

Related functions

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

Example program

JogMotion.cpp

Explanation

MCC_JogPulse() uses pulse as unit, do small stroke motion for specific axis, but pulse value can not exceed 2048. MCC_JogSpace() uses actual distance as unit(mm or inch), do also small stroke motion for specific axis. MCC_JogConti() move continuously to the right/left limit position set by mechanism parameter. The speed parameters of MCC_JogSpace() and MCC_JogConti() are set as velocity percent. Setting is similar to that of point to point motion. Used example is as follows:

```
MCC_SetUnit(_MM_, GROUP_INDEX);        //    Use mm as displacement unit.
MCC_JogPulse(0, 100, 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. Getting speed, position and command position

Related functions

MCC_GetCurFeedSpeed()
MCC_GetFeedSpeed()
MCC_GetCurPos()
MCC_GetPulsePos()
MCC_GetCurCommand()
MCC_GetCommandCount()

Example program

GetStatus.cpp

Explanation

MCC_GetCurFeedSpeed() is used to get the current feeding speed, MCC_GetSpeed() is used to get the feeding speeds of each axis. MCC_GetCurPos() is used to get the cartesian coordinates of current position of each axis, MCC_GetPulsePos() is used to get the motor coordinates(or pulse coordinates) of current position of each axis. The relationship between cartesian coordinates and motor coordinates is motor coordinate ＝ cartesian coordinate × (dfGearRatio /dfPitch) × dwPPR。Use MCC_GetCurPos and MCC_GetPulsePos() only when the channel of the group designated by GROUP_INDEX is enabled，i.e. not -1, for the coordinate value of that axis to be effective. For example channel of a certain group is disabled when

group_config.stGroupInfo[GROUP_INDEX].nchannel[2] = -1;

In this case z axis coordinate got via MCC_GetCurPos() or MCC_GetPulsePos() will always be 0. Used example is shown as follows:

double dfCurPosX, dfCurPosY, dfCurPosZ, dfCurPosU ,dfCurSpeed, dfNOT_USED;

```
double dfCurSpeedX, dfCurSpeedY, dfCurSpeedZ, dfCurSpeedU;
long lCurPulseX, lCurPulseY, lCurPulseZ, lCurPulseU, lNOT_USED;

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

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

// Get cartesian coordinates of current position of each axis
MCC_GetCurPos(&dfCurPosX, &dfCurPosY, &dfCurPosZ, &dfCurPosU,
&dfNOT_USED, &dfNOT_USED, GROUP_INDEX);

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

Use MCC_GetCurCommand() can obtain the information of motion command in execution, including motion command type、motion command encode、feed speed and target position etc. Use MCC_GetCommandCount() can obtain the number of motion command which is saved and has not been executed in motion buffer area. User can compare motion command encode to the number of saved motion command. Used example is as follows:

```
Int                    nCommandCount;
```

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

        /*
            stCommandInfo.nType;          motion commmand type
                                      0   point to point motion
                                      1   linear motion
                                      2   clockwise arc, circular motion
                                      3   counterclockwise arc,circular motion

            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. Motion hold, continue, abort

Related functions

MCC_HoldMotion()
MCC_ContiMotion()
MCC_AbortMotion()

Example program

CtrlMotion.cpp

Explanation

MCC_HoldMotion() is used to hold the current command in execution; MCC_ContiMotion() is used to continue executing the current hold command. Use MCC_ContiMotion() together with MCC_HoldMotion() for using and they must be used for the same group. MCC_AbortMotion() is used to abort the current command in execution ,it also clear rest motion commands in the buffer.

If there is no motion in execution, when call MCC_AbortMotion(), the return value of the function will be ABORT_ILLGEGAL_ERR. If there is no motion in execution, when call MCC_HoldMotion(), the return value of the function will be HOLD_ILLGEGAL_ERR; if calling MCC_HoldMotion() before unsuccessfully, call MCC_ContiMotion() the return value of the function will be CONTI_ILLGEGAL_ERR. Usually we will use the value returned by the function to judge if these functions are used not in the right time. Used example is shown as follows:

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

```
nRet = MCC_HoldMotion(GROUP_INDEX);
if (nRet != NO_ERR)
    {
    nRet = MCC_motion_abort(GROUP_INDEX);
    if (nRet != NO_ERR)
        printf("Erroneous return value : %d ", nRet);      // Use the function not at the
                                                           right time.
    }
else
        printf("Erroneous return value : %d ", nRet);// Use the function not
                                                                at the right time.
```

11. Motion status

Related functions

MCC_GetMotionStatus()

Example program

MotionFinished.cpp

Explanation

Use return value of MCC_GetMotionStatus() to check motion status. If returned value is 0, it means in motion; if returned value is 1, machine is in stop status, and there is no command in the motion buffer; After a successful call to MCC_HoldMotion(), the returned value of MCC_GetMotionStatus() is 2, which means machine is in hold status, and there are still unfinished commands in the buffer. Used example is shown as follows:

```
int nStatus;

MCC_Line(50, 50, 0, 0, 0, 0, GROUP_INDEX);

//      Wait for motion finish
  while (MCC_GetMotionStatus(GROUP_INDEX) != 1);

MCC_Line(20, 20, 0, 0, 0, 0, GROUP_INDEX);
nStatus = MCC_GetMotionStatus(GROUP_INDEX);
printf("Motion status : %d \r", nStatus);
```

12. Speed override

Related functions

MCC_SetOverSpeed()
MCC_GetOverSpeed()
MCC_SetPtPOverSpeed()
MCC_GetPtPOverSpeed()

Example program

OverSpeed.cpp

Explanation

MCC_SetOverSpeed() is used to set speed override for general motion. It is set is percent of former speed. MCC_GetOverSpeed() is used to get the current over speed ratio. For point to point motion speedoverride ratio need to call MCC_SetPtPOverSpeed() and MCC_GetPtPOverSpeed(). Used example is as follows:

```
//      Set general motion’s feeding speed to 20 mm/ sec
MCC_SetFeedSpeed(20, GROUP_INDEX);
MCC_Line(10, 10, 0 ,0 ,0 ,0 ,0, GROUP_INDEX)

//      Set speed override ratio, i.e. current speed
//      will become 20 X150% = 30 mm/ sec
MCC_SetOverSpeed(150, GROUP_INDEX);

nRate = MCC_GetOverSpeed( GROUP_INDEX);//    nRate should be 150
//          Set point to point motion’s feeding speed to 50 % of maximum speed of each axis.
//      i.e. equal to (wRPM * dfPitch / dfGearRatio)  ×50 %
```

```
MCC_SetPtPSpeed(50, GROUP_INDEX);


MCC_PtP(10, 10, 0 ,0 ,0 ,0 ,0, GROUP_INDEX)


//      Set point to point motion speed override ratio
  //    i. e. current speed will become (wRPM * dfPitch / dfGearRatio) 50 % × 150%
MCC_SetPtPOverSpeed(150, GROUP_INDEX);


nRate = MCC_GetPtPOverSpeed(GROUP_INDEX);//    nRate should be 150
```

13. Software limit check and limit switch check

Related functions

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

Example program

CheckOT.cpp

Explanation

MCCL provides software limit check function. When software limit check is enabled, if travel range of any axis passes limit value, system will stop movement. To resume from this situation, it is necessary to disable software limit check, and clear system's error record (refer to the explanation later). dfHighLimit, dfLowLimit, dfHighLimitOffset, dfLowLimitOffset of mechanism parameters to set software limit's ; MCC_SetOverTravelCheck() is used to enable and disable this, MCC_GetOverTravelCheck() is used to check current status. Used example is shown as follows:

```
//    Enable X axis software limit check.
MCC_SetOverTravelCheck(1, 0, 0, 0, 0, 0, GROUP_INDEX);

//    For OT0 ~ OT3 a value of 1 means software limit check is enabled for the axis, 0 means disabled.
MCC_GetOverTravelCheck(&OT0, &OT1, &OT2, &OT3, &NOT_USED, &NOT_USED, GROUP_INDEX);
```

nErrCode = MCC_GetErrCode(GROUP_INDEX);// Get error code

Use MCC_GetErrorCode() to judge if software limit is reached it cause system not to move. The returned value of 0xf301~0xf304correspond to X axis ~ U axis. If this situation appears, please follow the example showed below to put system back to normal motion, but be very careful to avoid motor's collision.

```
//  Disable software limit check
MCC_SetOverTravelCheck(0, 0, 0, 0, 0, 0, GROUP_INDEX);

//  Clear error record in system , let it continue to do the motion normally
MCC_ClearError(GROUP_INDEX);
```

MCCL provides hardware limit switch check also, use MCC_EnableLimitSwitchCheck() to enable it.If wOverTravelUpSensor and wOverTravelDownSensor mode are set 2, it does not make sense to call MCC_EnableLimitSwitchCheck(), Calling MCC_EnableLimitSwitchCheck(0) or MCC_EnableLimitSwitchCheck(1) enable limit switch check . When limit switch is touched, motion control card will stop outputting pulse, but the motion command will keep execution internally, user must call MCC_AbortMotion() to stop current motion command in execution. If limit switch check function is not needed, call MCC_DisableLimitSwitchCheck() Call MCC_GetLimitSwitchStatus() to check if limit switch is touched.

14. Speed blend function

Related functions

MCC_EnableBlend()
MCC_DisableBlend()
MCC_CheckBlend()

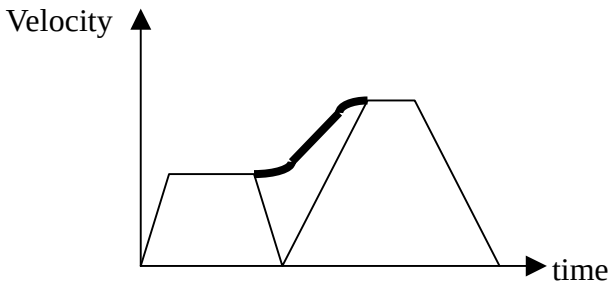
Example program

SetBlend.cpp

Explanation

MCCL provides speed blend function, the description of trajectory and speed blend in motion, please refer to "**PCI-4P Motion Library User's Manual**".

It is used to connect two different velocity profile smoothly.In the drawinf the speed of first move command transfer continuously to the second move command. Thus the motion time is shorter.



Calling MCC_EnableBlend() and MCC_DisableBlend() enables and disables speed blend function separately, call MCC_CheckBlend() get current setting. A returned value of 0 means enabled ; 1 means disabled.

15. Get and clear error status

Related functions

MCC_GetErrorCode()
MCC_ClearError()

Example function

ErrorStatus.cpp

Explanation

Error occurred in system, if its cause has been clearedremoved, it is still necessary to call MCC_ClearError()to clear error record in the system, otherwise motion won't continue. It is advised to get current error code regularly to check if there is any error. Used example is shown below. Also refer to software limit check and limit switch check.

```
nErrcode = MCC_GetErrCode(GROUP_INDEX);
if (nErrCode != 0)
    {

    /*
    Do necessary procedure
    */

    MCC_ClearError(GROUP_INDEX);//    Clear system's error record.
    }
```

Exception: while using arc motion command, a error code of 0xf104 means illegal calling parameter (points can not be collinear). In this case, there is no need to call MCC_ClearError.

16. Homing

Related functions

MCC_SetGohomeAccStep()
MCC_GetGohomeAccStep()
MCC_GoHome()
MCC_GetGohomeStatus()
MCC_AbortGoHome()

Example program

GoHome.cpp

Explanation

Homing parameters are set in HOME_CONFIG of mechanism parameter (please refer to "**PCI-4P Motion Library User's Manual"**). The default steps of homing acceleration and deceleration are 10 steps, i. e. the preset time for acceleration, deceleration are 10 × interpolation Time. To change them for smother motion use MCC_SetGohomeAccStep().

Use MCC_GetGoHomeStatus() to check if homing motion is finished, Use MCC_AbortGoHome() in the process of homing to abort it.

The homing provided by MCCL currently can only respond to one motion control card at a time. In case of homing with several cards, call MCC_GetGoHomeStatus() to check if finished , then proceed next card to call MCC_GoHome() for homing.

Used example is shown below:

```
//  Set acceleration, deceleration steps of homing procedure, if not being set, the default step is 10.
MCC_SetGoHomeAccStep(15, 15, 15, 15, 0, 0);
```

```
//      0xff means t axis won't do homing.
  MCC_GoHome(10, 10, 0, 0, 0, 0, 0, 0, 0xff, 0xff, 0xff, 0xff, CARD_INDEX);
        .
        .
MCC_AbortHome();//    If needed, use this function to abort homing.
        .
        .
nStatus = MCC_GetGoHomeStatus();
//      Use return value to check if homing procedure is done, if
//      nStatus value is 1, it has finished
```

17. Acceleration, deceleration step setting

Related functions

MCC_SetAccStep()
MCC_SetDecStep()
MCC_GetAccStep()
MCC_GetDecStep()
MCC_SetPtPAccStep()
MCC_SetPtPDecStep()
MCC_GetPtPAccStep()
MCC_GetPtPDecStep()

Example program

AccStep.cpp

Explanation

The default value is 10 steps for acceleration, deceleration of general motion (including line, arc, circular motion) and point to point motion, i.e. preset time for acceleration, deceleration is 10 × interpolation Time, Use MCC_SetAccStep(), MCC_SetDecStep(), MCC_SetPtPAccStep(), MCC_SetPtPDecStep() to change settings.

After setting feeding speed (for example call MCC_SetFeedSpeed () or MCC_SetPtPSpeed(),acceleration and deceleration steps should be adjusted ; for example when higher feeding speed is set, bigger acceleration and deceleration steps are better, especially for stepping motors.

In AccStep.cpp, as speed varies, different acceleration and deceleration steps are used. Calculate the acceleration into m/s^2 would help designing the velocity profile.