



HIWIN PCI-4P Motion Library

Reference Manual

PPH *WObit* mgr inż. Witold Ober

61-474 Poznań, ul. Gruszkowa 4

tel.061/8350-620, -621 fax. 061/8350704

e-mail: wobit@wobit.com.pl. <http://www.wobit.com.pl>

Ver. 1.0

2003

Contents

<u>I. MCCL FUNCTION LIST</u>	3
<u>II. GROUP PARAMETER AND MECHANISM PARAMETER</u>	7
<u>III. MCCL MOTION LIBRARY</u>	10
A. <u>SYSTEM FUNCTIONS</u>	10
B. <u>LOCAL I/O CONTROL</u>	14
C. <u>COORDINATE SYSTEM</u>	15
D. <u>LIMIT PROTECTION</u>	17
E. <u>TIME, SPEED AND ACCELERATION</u>	20
F. <u>POINT TO POINT MOTION</u>	22
G. <u>GENERAL MOTION</u>	26
H. <u>JOG MOTION</u>	33
I. <u>MOTION STATUS</u>	35
J. <u>HOMING</u>	37
K. <u>VELOCITY BLENDING AND SPEED OVERRIDE</u>	40
L. <u>ENCODER</u>	44
<u>IV. ERROR CODE</u>	46
<u>V. FUNCTION RETURN VALUES</u>	47
<u>VI. MCCL INITIAL SETTINGS</u>	49

I. MCCL Function list

A. System functions

Function	Explanation	NO.
MCC_SetGroupConfig()	Set group parameter(for C language)	1
MCC_SetGroupContent()	Set group parameter (for C, BASIC language)	2
MCC_SetMachParam()	Set system mechanism parameter	3
MCC_GetMachParam()	Get system mechanism parameter	4
MCC_UpdateMachParam()	Respond updated system mechanism parameter	5
MCC_InitSystem()	Open motion control library	6
MCC_CloseSystem()	Close motion control library	7
MCC_ResetMotion()	Reset motion control library	8
MCC_ResetDevice()	Reset motion control card	9
MCC_EnableDryRun()	Enable motion dry run	10
MCC_DisableDryRun()	Disable motion dry run	11
MCC_CheckDryRun()	Check motion dry run status	12
MCC_SetSysMaxSpeed()	Set system's maximum feeding speed	13
MCC_GetSysMaxSpeed()	Get system's maximum feeding speed	14

B. Local I/O control

Function	Explanation	NO.
MCC_SetServoOn()	Set servo on (SVN signal)	1
MCC_SetServoOff()	Set servo off (SVN signal)	2
MCC_EnablePosReady()	Turn on PRDY signal	3
MCC_DisablePosReady()	Turn off PRDY signal	4

C. Coordinate system

Function	Explanation	NO.
MCC_SetUnit()	Set displacement unit	1

MCC_GetUnit()	Get displacement unit	2
MCC_SetIncrease()	Set incremental mode	3
MCC_SetAbsolute()	Set absolute mode	4
MCC_GetCoordType()	Get absolute or incremental mode	5
MCC_GetCurPos()	Get current position in cartesian coordinate of each axis	6
MCC_GetPulsePos()	Get current position in motor coordinate of each axis	7

D. Limit Protection

Function	Explanation	NO.
MCC_EnableLimitSwitchCheck()	Enable limit switch check	1
MCC_DisableLimitSwitchCheck()	Disable limit switch check	2
MCC_SetOverTravelCheck()	Set software limit check	3
MCC_GetOverTravelCheck()	Get software limit check	4
MCC_GetLimitSwitchStatus()	Get limit switch status	5

E. Time, Speed and Acceleration

Function	Explanation	NO.
MCC_SetInterpolateTime()	Set interpolation time	1
MCC_GetInterpolateTime()	Get interpolation time	2
MCC_SetMaxPulseSpeed()	Set maximum pulse speed for each axis	3
MCC_GetMaxPulseSpeed()	Get maximum pulse speed for each axis	4
MCC_SetMaxPulseAcc()	Set maximum pulse acceleration for each axis	5
MCC_GetMaxPulseAcc()	Get maximum pulse acceleration for each axis	6

F. Point to point motion

Function	Explanation	NO.
MCC_SetPtPAccType()	Set acceleration type for each axis	1

MCC_GetPtPAccType()	Get acceleration type for each axis	2
MCC_SetPtPDecType()	Set deceleration type for each axis	3
MCC_GetPtPDecType()	Get deceleration type for each axis	4
MCC_SetPtPAccStep()	Set acceleration steps for each axis	5
MCC_GetPtPAccStep()	Get acceleration steps for each axis	6
MCC_SetPtPDecStep()	Set deceleration steps for each axis	7
MCC_GetPtPDecStep()	Get deceleration steps for each axis	8
MCC_SetPtPSpeed()	Set feeding speed	9
MCC_GetPtPSpeed()	Get feeding speed	10
MCC_PtP()	Execute point to point motion	11

G. General motion

Function	Explanation	NO.
MCC_SetAccType()	Set acceleration type for each axis	1
MCC_GetAccType()	Get acceleration type for each axis	2
MCC_SetDecType()	Set deceleration type for each axis	3
MCC_GetDecType()	Get deceleration type for each axis	4
MCC_SetAccStep()	Set acceleration steps for each axis	5
MCC_GetAccStep()	Get acceleration steps for each axis	6
MCC_SetDecStep()	Set deceleration steps for each axis	7
MCC_GetDecStep()	Get deceleration steps for each axis	8
MCC_SetFeedSpeed()	Set feeding speed	9
MCC_GetFeedSpeed()	Get feeding speed	10
MCC_GetCurFeedSpeed()	Get current feeding speed	11
MCC_GetSpeed()	Get speed component of each axis in a group	12
MCC_Line()	Linear general motion	13
MCC_ArcXYZ()	Arc general motion for X-Y-Z space	14
MCC_ArcXY()	Arc general motion for X-Y plane	15
MCC_ArcYZ()	Arc general motion for Y-Z plane	16
MCC_ArcZX()	Arc general motion for Z-X plane	17
MCC_CircleXY()	Circular general motion for X-Y plane	18

MCC_CircleYZ()	Circular general motion for Y-Z plane	19
MCC_CircleZX()	Circular general motion for Z-X plane	20

H. Jog motion

Function	Explanation	NO.
MCC_JogPulse()	Pulse jog	1
MCC_JogSpace()	Short stroke jog	2
MCC_JogConti()	Continuous jog	3

I. Motion Status

Function	Explanation	NO.
MCC_GetMotionStatus()	Get motion status	1
MCC_GetCurCommand()	Get information of motion command in execution	2
MCC_GetCommandCount()	Get count of command in stock	3
MCC_ResetCommandIndex()	Reset command index	4
MCC_GetErrorCode()	Get error code	5
MCC_ClearError()	Clear system's error record	6

J. Homing

Function	Explanation	NO.
MCC_SetGoHomeAccStep()	Set acceleration, deceleration steps	1
MCC_GetGoHomeAccStep()	Get acceleration, deceleration steps	2
MCC_GoHome()	Execute homing	3
MCC_GetGoHomeStatus()	Check if homing is finished	4
MCC_AbortGoHome()	Abort homing	5
MCC_GetHomeSensorStatus()	Get home sensor status	6

K. Velocity blending and speed override

Function	Explanation	NO.
MCC_HoldMotion()	Hold motion	1

MCC_ContiMotion()	Continue motion	2
MCC_AbortMotion()	Abort motion	3
MCC_EnableBlend()	Enable velocity blend	4
MCC_DisableBlend()	Disable velocity blend	5
MCC_CheckBlend()	Check whether velocity blend is enabled or not	6
MCC_DelayMotion()	Set motion delay time	7
MCC_CheckDelay()	Check whether in delay motion or not	8
MCC_SetOverSpeed()	Set speed override ratio for general motion	9
MCC_GetOverSpeed()	Get speed override ratio for general motion	10
MCC_SetPtPOverSpeed()	Set speed override ratio for point to point motion	11
MCC_GetPtPOverSpeed()	Get speed override ratio for point to point motion	12

L. Encoder

Function	Explanation	NO.
MCC_SetENCInputRate()	Set encoder mode(1x,2x,4x)	1
MCC_ClearENCCounter()	Reset encoder counter	2
MCC_GetENCValue()	Read encoder counter	3
MCC_GetENCIndexStatus()	Read index signal	4

II. Group parameter and mechanism parameter

Before motion could be executed, it is necessary to set group parameter and mechanism parameters. Then use MCC_InitSystem() to initialize. Regarding detailed explanation of parameters, please refer to "PCI4P Motion Library User's Manual".

□ Group parameter

```
typedef struct _SYS_GROUP_INFO
{
    int nCardIndex;
    int nChannel[6];
} SYS_GROUP_INFO;
```

```
typedef struct _SYS_GROUP_CONFIG
{
    int nGroupUsed[72];
    SYS_GROUP_INFO stGroupInfo[72];
} SYS_GROUP_CONFIG;
```

□ Mechanism parameter

```
typedef struct _SYS_MACH_PARAM
{
    WORD wPosToEncoderDir;
    WORD wRPM;
    DWORD dwPPR;
    double dfPitch;
    double dfGearRatio;
    double dfHighLimit;
    double dfLowLimit;
    double dfHighLimitOffset;
    double dfLowLimitOffset;

    WORD wPulseMode;
    WORD wPulseWidth;
    WORD wCommandMode;
    WORD wPaddle;
```



```
HOME_CONFIG          stHome;

ENCODER_CONFIG       stEncoder;

WORD                 wOverTravelUpSensorMode;
WORD                 wOverTravelDownSensorMode;
} SYS_MACH_PARAM;
```

```
typedef struct HOME_CONFIG
{
    WORD  wType;
    WORD  wPhase0Dir;
    WORD  wPhase1Dir;
    WORD  wSensorMode;
    double dfOffset;
} HOME_CONFIG;
```

```
typedef struct _ENCODER_CONFIG
{
    WORD  wType;
    WORD  wAInverse;
    WORD  wBInverse;
    WORD  wCInverse;
    WORD  wABSwap;
    WORD  wPaddle[3];
} ENCODER_CONFIG;
```

□ Hardware parameter

```
typedef struct _SYS_CARD_CONFIG
{
    WORD  wCardType;          // Value is fixed : 2
```

```
WORD    wCardAddress;    // Not used
WORD    wIRQ_No;         // Not used
WORD    wPaddle;         // Not used
} SYS_CARD_CONFIG;
```

III. MCCL motion library

A. System functions

1. int MCC_SetGroupConfig(SYS_GROUP_CONFIG *pstConfig)

Description Set group parameter. Before calling any other function in the function library, run this function first. Regarding, detail information of group parameter, please refer to "PCI4P Motion Library User's Manual". **This function is only used by C language. If BASIC language is used, please use MCC_SetGroupContent().**

Parameters *pstconfig group parameters

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer to

V. Function return value.

2. int MCC_SetGroupContent(WORD wGroupIndex, WORD wCardIndex, int nChannel0, int nChannel1, int nChannel2, int nChannel3, int nChannel4, int nChannel5)

Description Set group parameter before calling any other function in the function library. **This function is especially for BASIC programming language.**

Parameters wGroupIndex Designate the group number whose content is to be set. Range of

value 0 ~ 71

wCardIndex

Set card number(0 ~ 11) to be used by the designated group.

nChannel0~nChannel3

The suffixes 0~3 correspond to X,Y,Z,U accordingly in a group. These parameters dispatch channel number on the card of *wCardIndex* to X,Y,Z,U in a group. A value of -1 means disable.

nChannel4~nChannel5 Not used.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to

V. Function return value.

3. int MCC_SetMachParam(SYS_MACH_PARAM *pstMachParam, WORD wChannel, WORD wCardIndex)

Description Set system mechanism parameter.

Parameters **pstMachParam* mechanism parameter

wChannel number (0 ~ 3) of motion axis on a card

wCardIndex number (0 ~ 11)of motion control card

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer to

V. Function return value.

4. int MCC_GetMachParam(SYS_MACH_PARAM *pstMachParam, WORD wChannel, WORD wCardIndex)

Description Get system mechanism parameter.

Parameters **pstMachParam* mechanism parameter

wChannel number (0 ~ 3) of motion axis on a card

wCardIndex number (0 ~ 11)of motion control card

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer to

V. Function return value.



5. int MCC_UpdateMachParam()

Description Update system mechanism parameter. Once after MCC_InitSystem(), if there is any change in mechanism parameters(using MCC_SetMachPara()), run this function to update system mechanism parameter.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to

V. Function return value.

6. int MCC_InitSystem(int *nInterpolateTime*, SYS_CARD_CONFIG **psCardConfig*, WORD *wCardNo*)

Description After calling MCC_SetMachParam(), MCC_SetGroupConfig() (or MCC_SetGroupContent()), use this function to initialize motion library. This function has only to be called once.

nInterpolateTime Interpolation time in ms. Value range 1~200. Smaller value results in better operation, however increase computer load. It is suggested to use 5 ms.

PsCardConfig Hardware parameter of motion control card. For detail information, please refer to "PCI4P Motion Library User's Manual".

wCardNo Number(1 ~ 12)of motion control card to be used

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to

V. Function return value.

7. int MCC_CloseSystem()

Description Close motion library. After executing this function, unless calling MCC_InitSystem() again, library can not be used.



Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to
 V. Function return value.

8. int MCC_ResetMotion()

Description Reset motion library. After executing this function, all error status will be cleared, and cartesian coordinate and motor coordinate are set to zero, system will return to the state when MCC_InitSystem() was called.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to
 V. Function return value.

9. int MCC_ResetDevice()

Description Reset all modules in motion control card, and return to the state of turn on hardware. But this function won't clear error status, and neither set cartesian coordinate and motor coordinate to zero.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to
 V. Function return value.

10. int MCC_EnableDryRun()

Description Enable dry run function. After enabling this function, the calculation of motion command is still doned, but no pulse is output. In this case MCC_GetCurPos() is applicable for analysis or graph.

11. int MCC_DisableDryRun()

Description Disable dry run.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to
 V. Function return value.

12. int MCC_CheckDryRun()

Description Check whether set dry run or not.

Return Value 0 means set.

1 means not set.

others Fail , meaning of returned value, please refer to V.

Function return value.

13. int MCC_SetSysMaxSpeed(double *dfMaxSpeed*)

Description Set maximum feeding speed of general motion. The speed set by MCC_SetFeedSpeed() never exceed this value.

Parameters *dfMaxSpeed* maximum feeding speed.

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer to

V. Function return value.

14. int MCC_GetSysMaxSpeed()

Description Get maximum feeding speed of general motion.

Return Value Maximum feeding speed of general motion.

B. Local I/O control

1. int MCC_SetServoOn(WORD *wChannel*, WORD *wCardIndex*)

Description Set servo on signal.(SVN0~SVN1~SVN2~SVN3)

Parameters *wChannel* Number(0 ~ 3) of motion axis.

wCardIndex Number(0 ~ 11) of motion control card.

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer to

V. Function return value.

2. int MCC_SetServoOff(WORD *wChannel*, WORD *wCardIndex*)

Description Turn off servo on signal. (SVN0~SVN1~SVN2~SVN3)

Parameters *wChannel* Number(0 ~ 3) of motion axis.
wCardIndex Number(0 ~ 11) of motion control card.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to
V. Function return value.

3. int MCC_EnablePosReady(WORD *wCardIndex*)

Description Turn on PRDY signal.

Parameters *wCardIndex* Number(0 ~ 11) of motion control card.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to
V. Function return value.

4. int MCC_DisablePosReady(WORD *wCardIndex*)

Description Turn off PRDY signal.

Parameters *wCardIndex* Number(0 ~ 11) of motion control card.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to
V. Function return value.

C. Coordinate system

1. int MCC_SetUnit(int *nUnitMode*, WORD *wGroupIndex*)

Description Set displacement unit

Parameters *nUnitMode* Set unit type

1	use metric system, mm
2	use British system, inch

wGroupIndex group number(0 ~ 71)

Return Value *nUnitMode* is returned.

2. int MCC_GetUnit(WORD *wGroupIndex*)

Description Get displacement unit

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value 1: metric , mm
 2: British , inch

3. int MCC_SetIncrease(WORD *wGroupIndex*)

Description Set incremental mode for positioning.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to V.
Function return value.

4. int MCC_SetAbsolute(WORD *wGroupIndex*)

Description Set absolute mode for positioning.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to V.
Function return value.

5. int MCC_GetCoordType(WORD *wGroupIndex*)

Description Get positioning mode(incremental or absolute).

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value 0 Incremental mode
 1 Absolute mode
 others Fail , meaning of returned value, please refer to
V. Function return value.

6. int MCC_GetCurPos(double *pdfX, double *pdfY, double *pdfZ, double *pdfU, double *pdfV, double *pdfW,

WORD *wGroupIndex*)

Description	Get current position in cartesian coordinate for each axis. If in the group the axis is disabled, the position is of no meaning.
Parameters	<i>*pdfX</i> , <i>*pdfY</i> , <i>*pdfZ</i> , <i>*pdfU</i> save each axis's position in cartesian coordinate. <i>*pdfV</i> , <i>*pdfW</i> are not used. <i>wGroupIndex</i> group number(0 ~ 71)
Return Value	0 Succeed Non zero Fail , meaning of returned value, please refer to

V. Function return value.

**7. int MCC_GetPulsePos(long **plX*, long **plY*, long **plZ*,
 long **plU*, long **plV*, long **plW*,
 WORD *wGroupIndex*)**

Description	Get current position in motor coordinate for each axis. If in the group the axis is disabled, the position is of no meaning.
Parameters	<i>*plX</i> , <i>*plY</i> , <i>*plZ</i> , <i>*plU</i> save each axis's position in pulse coordinate (or motor coordinate). <i>*pdfV</i> , <i>*pdfW</i> are not used. <i>wGroupIndex</i> group number(0 ~ 71)
Return Value	0 Succeed Non zero Fail , meaning of returned value, please refer to

V. Function return value.

D. Limit protection

1. int MCC_EnableLimitSwitchCheck(int *nMode*)

Description	Enable limit switch check function. Notice that mechanism parameters : <i>wOverTravelUpSensorMode</i> and <i>wOverTravelDownSensorMode</i> have to be set to 0 (normal open) or 1 (normal close), thus limit switch check function could work correctly. After enable this function and set proper check mode of limit switch, when system touches limit switch,
-------------	--



motion control card will stop outputting pulses, but motion command is still in calculation internally, user need to call MCC_AbortHome() or MCC_AbortMotion() to stop the internal motion command in execution. This function is always used together with MCC_GetLimitSwitchStatus(), which provide status, whether limit switch is engaged or not.

Parameters	<i>nMode</i>	Limit switch check mode
	0	Limit switches are direction-sensitive. (For example move in the positive direction and touch the positive limit switch, or move in the negative direction and touch the negative limit switch, it will stop outputting pulses.)
	1	Limit switches are not direction-sensitive. So long as any limit switch is engaged, it stop outputting pulses.
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to V. Function return value.

2. int MCC_DisableLimitSwitchCheck()

Description Disable limit switch check function.

Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to V. Function return value.

3. int MCC_SetOverTravelCheck(int *nCheck0*, int *nCheck1*, int *nCheck2*, int *nCheck3*, int *nCheck4*, int *nCheck5*, WORD *wGroupIndex*)

Description Set software limit check function.

Parameters	<i>nCheck0</i> , <i>nCheck1</i> , <i>nCheck2</i> , <i>nCheck3</i> 1 means check, 0 means no check.
------------	--

nCheck4, nCheck5 Not used.

	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to

V. Function return value.

**4. int MCC_GetOverTravelCheck(int *pnChk0, int *pnChk1, int *pnChk2,
 int *pnChk3, int *pnChk4, int *pnChk5,
 WORD wGroupIndex)**

Description	Get software limit check setting condition.	
Parameters	<i>*pnChk0, *pnChk1, *pnChk2, *pnChk3</i> 1 means check, 0 means no check.	
	<i>*pnChk4, *pnChk5</i> Not used.	

	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to V.

Function return value.

**5. int MCC_GetLimitSwitchStatus(WORD *pwStatus, WORD wUpDown,
 WORD wChannel, WORD wCardIndex)**

Description	Get the status of limit switch. Before using this function, <i>wOverTravelUpSensorMode</i> and <i>wOverTravelDownSensorMode</i> in mechanism parameter must be set correctly(normal open or normal close).	
Parameters	<i>*pwStatus</i>	limit switch status,1 means engaged 0 means not.
	<i>wUpDown</i>	Set 0 to read negative limit switch set 1 to read positive limit switch.
	<i>wChannel</i>	Number (0 ~ 3) of motion axis.
	<i>wCardIndex</i>	Number(0 ~ 11) of motion control card.
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to

V. Function return value.

E. Time, Speed and Acceleration

1. int MCC_SetInterpolateTime(double *dfTime*)

Description Set interpolation time. Smaller value results in better operation, however increase computer load. For detail information, please refer to "PCI4P Motion Library User's Manual".

Parameters *dfTime* Interpolation time, unit is ms, range is 1ms ~ 200ms. Usually, 5ms of setting value is OK.

Return Value Greater than zero Actual by set interpolation time.

Others Fail , meaning of returned value, please refer to **V. Function return value.**

2. int MCC_GetInterpolateTime()

Description Get interpolation time.

Return Value Greater than zero Actual by set interpolation time.

Others Fail , meaning of returned value, please refer to **V. Function return value.**

3. int MCC_SetMaxPulseSpeed(int *nPulse0*, int *nPulse1*, int *nPulse2*, int *nPulse3*, int *nPulse4*, int *nPulse5*)

Description Set the maximum pulse speed for each axis. This is to set the maximum pulses which could be sent within one interpolation time. For detail information, please refer to "PCI4P Motion Library User's Manual".

Parameters *nPulse0* ~ *nPulse3* Set the maximum pulse speed for each axis. Range is 1~32767.

nPulse4 ~ *nPulse5* Not used.

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer



to V. Function return value.

**4. int MCC_GetMaxPulseSpeed(int **pnPulse0*, int **pnPulse1*, int **pnPulse2*,
 int **pnPulse3*, int **pnPulse4*, int **pnPulse5*)**

Description Get maximum pulse speed set for each axis.

Parameters **pnPulse0* ~ **pnPulse3* Maximum pulse speed for each axis.
 **pnPulse4* ~ **pnPulse5* Not used.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer
 to V. Function return value.

**5. int MCC_SetMaxPulseAcc(int *nPulse0*, int *nPulse1*, int *nPulse2*,
 int *nPulse3*, int *nPulse4*, int *nPulse5*)**

Description Set the maximum pulse acceleration for each axis. This sets the
 maximum difference of sent pulses between two contiguous
 interpolation time. For detail information, please refer to
 "PCI4P Motion Library User's Manual".

Parameters *nPulse0* ~ *nPulse3* the maximum difference of sent
 pulses between two contiguous
 interpolation time for each
 axis.Range is 1~32767.
 nPulse4 ~ *nPulse5* Not used.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer
 to V. Function return value.

**6. int MCC_GetMaxPulseAcc(int **pnPulse0*, int **pnPulse1*, int **pnPulse2*,
 int **pnPulse3*, int **pnPulse4*, int **pnPulse5*)**

Description Get maximum pulse acceleration setting for each axis.

Parameters **pnPulse0* ~ **pnPulse3* Maximum pulse acceleration for each axis.
 **pnPulse4* ~ **pnPulse5* Not used.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to

V. Function return value.

F. Point to point motion

**1. int MCC_SetPtPAccType(char *cType0*, char *cType1*, char *cType2*,
 char *cType3*, char *cType4*, char *cType5*, WORD
wGroupIndex)**

Description Set acceleration type in point to point motion for each axis individually.

Parameters *cType0* ~ *cType3* Acceleration type for each axis□

'T' Trapezoidal curve

'S' S curve

cType4 ~ *cType5* Not used.

wGroupIndex group number(0 ~ 71)

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer to **V. Function return value.**

**2. int MCC_GetPtPAccType(char **pcType0*, char **pcType1*, char **pcType2*,
 char **pcType3*, char **pcType4*, char **pcType5*,
 WORD *wGroupIndex*)**

Description Get acceleration type in point to point motion for each axis.

Parameters **pcType0* ~ **pcType3* Acceleration type for each axis, 0: trapezoidal curve, 1: S curve.

**pcType4* ~ **pcType5* Not used.

wGroupIndex group number(0 ~ 71)

Return Value 0 Succeed

Non zero Fail , meaning of returned value,

please refer to **V. Function return value.**

**3. int MCC_SetPtPDecType(char *cType0*, char *cType1*, char *cType2*,
 char *cType3*, char *cType4*, char *cType5*, WORD *wGroupIndex*)**

Description Set deceleration type in point to point motion for each axis individually.

Parameters *cType0* ~ *cType3* Deceleration type for each axis□

'T' Trapezoidal curve

'S' S curve

cType4 ~ *cType5* Not used.

wGroupIndex group number(0 ~ 71)

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer to

V. Function return value.

**4. int MCC_GetPtPDecType(char **pcType0*, char **pcType1*, char **pcType2*,
 char **pcType3*, char **pcType4*, char **pcType5*,
 WORD *wGroupIndex*)**

Description Get deceleration type in point to point motion for each axis.

Parameters **pcType0* ~ **pcType3* Deceleration type for each axis, 0: trapezoidal curve, 1: S curve.

**pcType4* ~ **pcType5* Not used.

wGroupIndex group number(0 ~ 71)

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer to

V. Function return value.

**5. int MCC_SetPtPAccStep(int *nStep0*, int *nStep1*, int *nStep2*,int *nStep3*,
 int *nStep4*, int *nStep5*, WORD *wGroupIndex*)**

Description Set acceleration steps for each axis in point to point

motion.<acceleration time> = <interpolation time> ×
 <acceleration steps>.Each axis uses their own acceleration
 type.

Parameters *nStep0~ nStep3* Acceleration steps for each axis.
nStep4~ nStep5 Not used.
wGroupIndex group number(0 ~ 71)
 Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please
 refer to **V. Function return value.**

**6. int MCC_GetPtPAccStep(int *pnStep0, int *pnStep1, int *pnStep2,
 int *pnStep3, int *pnStep4, int *pnStep5, WORD wGroupIndex)**

Description Get acceleration steps for each axis in point to point motion.

Parameters *pnStep0~*pnStep3 Acceleration steps for each axis
 *pnStep4~*pnStep5 Not used.

wGroupIndex group number(0 ~ 71)
 Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please
 refer to **V. Function return value.**

**7. int MCC_SetPtPDecStep(int nStep0, int nStep1, int nStep2,
 int nStep3, int nStep4, int nStep5, WORD wGroupIndex)**

Description Set deceleration steps for each axis in point to point
 motion.<deceleration time> = <interpolation time> ×
 <deceleration steps>.Each axis uses their own deceleration
 type.

Parameters *nStep0~ nStep 3* Deceleration steps for each axis
nStep4~ nStep 5 Not used.
wGroupIndex group number(0 ~ 71)
 Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please
 refer to **V. Function return value.**

**8. int MCC_GetPtPDecStep(int *pnStep0, int *pnStep1, int *pnStep2,
 int *pnStep3, int *pnStep4, int *pnStep5, WORD wGroupIndex)**

Description Get deceleration steps for each axis in point to point motion.

Parameters *pnStep0~*pnStep3 Deceleration steps for each axis

*pnStep4~*pnStep5 Not used.

wGroupIndex group number(0 ~ 71)

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer to

V. Function return value.

9. int MCC_SetPtPSpeed(int nRatio, WORD wGroupIndex)

Description Set speed ratio for point to point motion. This speed is

$$(wRPM \times dfPitch / dfGearRatio) \times (nRatio / 100)$$

in which *wRPM*, *dfPitch*, *dfGearRatio* are defined in mechanism parameter.

Parameters *nRatio* Set speed ratio; if *ratio* < 0, set to 0 automatically; if *ratio* > 100, then set to 100 automatically.

wGroupIndex group number(0 ~ 71)

Return Value Actual set speed ratio.

10. int MCC_GetPtPSpeed(WORD wGroupIndex)

Description Get speed ratio for point to point motion.

Parameters wGroupIndex group number(0 ~ 71)

Return Value Greater than or equal to zero Actual set speed ratio

Less than zero Fail , meaning of returned value, please refer to **V. Function return value.**

**11. int MCC_PtP(double dfX, double dfY, double dfZ,
 double dfU, double dfV, double dfW, WORD wGroupIndex)**



Description	Execute point to point motion with set feeding speed.	
Parameters	<i>dfX, dfY, dfZ, dfU</i>	Target position or displacement for X, Y, Z, U axis.
	<i>dfV, dfW</i>	Not used.
	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	Greater than or equal to zero	Command Index value set by MCCL.
	Less than zero	Fail , meaning of returned value, please refer to V. Function return value.

G. General motion

1. int MCC_SetAccType(char *cAccType*, WORD *wGroupIndex*)

Description	Set acceleration type of each group in line, arc, circular motion.	
Parameters	<i>cAccType</i>	Acceleration type for each axis may be set up as follows
	'T'	Trapezoidal curve
	'S'	S curve
	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to V. Function return value.

2. int MCC_GetAccType(WORD *wGroupIndex*)

Description	Get acceleration type of each group in line, arc or circular motion.	
Parameters	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	0	Trapezoidal curve.

1 S curve.

3. int MCC_SetDecType(char *cDecType*, WORD *wGroupIndex*)

Description Set deceleration type of each axis in line, arc, circular motion.
 Parameters *cAccType* Deceleration type for each group may be set up as follows□

'T'	Trapezoidal curve
'S'	S curve

wGroupIndex group number(0 ~ 71)
 Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to
V. Function return value.

4. int MCC_GetDecType(WORD *wGroupIndex*)

Description Get deceleration type of each group in line, arc or circular motion.
 Parameters *wGroupIndex* group number(0 ~ 71)
 Return Value 0 Trapezoidal curve.
 1 S curve.

5. int MCC_SetAccStep(int *nStep*, WORD *wGroupIndex*)

Description Set acceleration steps for each group in general motion.<acceleration time> = <interpolation time> × <acceleration steps>.
 Parameters *nStep* Set acceleration steps
wGroupIndex group number(0 ~ 71)
 Return Value Greater than or equal to zero Autual set acceleration steps.
 Less than zero Fail , meaning of returned value, please refer to **V. Function return value.**

6. int MCC_GetAccStep(int *nStep*, WORD *wGroupIndex*)

Description Get acceleration steps for each group in general motion.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value Greater than or equal to zero Acceleration steps value.

Less than zero Fail , meaning of returned value, please refer to **V. Function return value.**

7. int MCC_SetDecStep(int *nStep*, WORD *wGroupIndex*)

Description Set deceleration steps for each group in general motion.<deceleration time> = <interpolation time> × <deceleration steps>.

Parameters *nStep* Set deceleration steps
wGroupIndex group number(0 ~ 71)

Return Value Greater than or equal to zero Actual set deceleration steps.

Less than zero Fail , meaning of returned value, please refer to **V.Function return value.**

8. int MCC_GetDecStep(int *nStep*, WORD *wGroupIndex*)

Description Get deceleration steps for each group in general motion.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value Greater than or equal to zero Actual set deceleration steps.

Less than zero Fail , meaning of returned value, please refer to **V. Function return value.**

9. int MCC_SetFeedSpeed(double *dfFeedSpeed*, WORD *wGroupIndex*)

Description Set feeding speed of line, arc, circular motion, unit is **mm/sec**;
 if *dfFeedSpeed* < 0 then set to 0 automatically.

Parameters *dfFeedSpeed* Feeding speed.
wGroupIndex group number(0 ~ 71)

Return Value Actual set feeding speed.

10. double MCC_GetFeedSpeed(WORD *wGroupIndex*)

Description Get feeding speed of line, arc, and circular motion.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value Current feeding speed.

11. double MCC_GetCurFeedSpeed(WORD *wGroupIndex*)

Description Get current feeding speed of line, arc and circular motion.
 This speed at constant velocity region should equal to the
 feeding speed(Get by using MCC_GetFeedSpeed()).

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value Current actual feeding speed.

12. double MCC_GetSpeed(double *pdfV0, double *pdfV1, double *pdfV2, double *pdfV3, double *pdfV4, double *pdfV5, WORD *wGroupIndex*)

Description For general motion, get current speed component of each axis in
 a group.

Parameters *pdfV0~*pdfV3 the current feeding speeds of each axis.
 *pdfV4~*pdfV5 Not used

wGroupIndex group number(0 ~ 71)

Return Value Current actual feeding speed.

13. int MCC_Line(double dfX0, double dfX1, double dfX2, double dfX3, double dfX4, double dfX5, WORD *wGroupIndex*)

Description Linear general motion. If this function is called successfully,
 command counter value will increase by 1.

Parameters dfX0 ~ dfX3 each axis's coordinate of target position or
 displacement.

dfX4 ~ dfX5 Not used.

wGroupIndex group number(0 ~ 71)

Return Value Greater than or equal to zero CommandIndex value set by MCCL.
 Less than zero Fail , meaning of returned
 value, please refer to V.

Function return value.

**14. int MCC_ArcXYZ(double *dfRX0*, double *dfRX1*, double *dfRX2*,
 double *dfDX0*, double *dfDX1*, double *dfDX2*,
 WORD *wGroupIndex*)**

Description In the space defined by **X-Y-Z** axis, execute arc general motion, starting from current position using reference point as center move to target. If this function is called successfully, command counter value will increase by 1.

Parameters *dfRX0 ~dfRX2* **X-Y-Z** coordinates of reference point.
dfDX0~dfDX2 **X-Y-Z** coordinates of target position.
wGroupIndex group number(0 ~ 71)

Return Value Greater than or equal to zero CommandIndex value set by MCCL.

Less than zero Fail , meaning of returned value, please refer to **V. Function return value.**

**15. int MCC_ArcXY(double *dfRX*, double *dfRY*, double *dfDX*, double *dfDY*,
 WORD *wGroupIndex*)**

Description On the plane of **X-Y** axis, execute arc general motion, starting from current position using reference point as center move to target. If this function is called successfully, command counter value will increase by 1.

Parameters *dfRX, dfRY* **X-Y** coordinates of reference position.
dfDX, dfDY **X-Y** coordinates of target position.
wGroupIndex group number(0 ~ 71)

Return Value Greater than or equal to zero CommandIndex value set by MCCL.

Less than zero Fail , meaning of returned value, please refer to **V. Function return value.**



16. int MCC_ArcYZ(double *dfRY*, double *dfRZ*, double *dfDY*, double *dfDZ*, WORD *wGroupIndex*)

Description	On the plane of Y-Z axis, execute arc general motion, starting from current position using reference point as center move to target. If this function is called successfully, command counter value will increase by 1.	
Parameters	<i>dfRY</i> , <i>dfRZ</i>	Y-Z coordinates of reference position.
	<i>dfDY</i> , <i>dfDZ</i>	Y-Z coordinates of target position.
	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	Greater than or equal to zero	CommandIndex value set by MCCL.
	Less than zero	Fail , meaning of returned value, please refer to V. Function return value.

17. int MCC_ArcZX(double *dfRZ*, double *dfRX*, double *dfDY*, double *dfDZ*, WORD *wGroupIndex*)

Description	On the plane of Z-X axis, execute arc general motion, starting from current position using reference point as center move to target. If this function is called successfully, command counter value will increase by 1.	
Parameters	<i>dfRZ</i> , <i>dfRX</i>	Z-X coordinates of reference position.
	<i>dfDZ</i> , <i>dfDX</i>	Z-X coordinates of target position.
	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	Greater than or equal to zero	CommandIndex value set by MCCL.
	Less than zero	Fail , meaning of returned value, please refer to V. Function return value.

18. int MCC_CircleXY(double *dfCX*, double *dfCY*, BYTE *byCirDir*, WORD *wGroupIndex*)

Description	On the plane of X-Y axis, from current position execute a	
-------------	--	--

circular motion with the center designated by its coordinates.
 If this function is called successfully, command counter value will increase by 1.

Parameters *dfCX, dfCY* **X-Y** axis coordinates of designated circle center.
byCirDir motion direction, 0: CW , 1: CCW
wGroupIndex group number(0 ~ 71)

Return Value Greater than or equal to zero CommandIndex value set by MCCL.
 Less than zero Fail , meaning of returned value, please refer to **V. Function return value.**

**19. int MCC_CircleYZ(double *dfCY*, double *dfCZ*, BYTE *byCirDir*,
 WORD *wGroupIndex*)**

Description On the plane of **Y-Z** axis, from current position execute a circular motion with the center designated by its coordinates.
 If this function is called successfully, command counter value will increase by 1.

Parameters *dfCY, dfCZ* **Y-Z** axis coordinates of designated circle center.
byCirDir motion direction, 0: CW , 1: CCW
wGroupIndex group number(0 ~ 71)

Return Value Greater than or equal to zero CommandIndex value set by MCCL.
 Less than zero Fail , meaning of returned value, please refer to **V. Function return value.**

**20. int MCC_CircleZX(double *dfCZ*, double *dfCX*, BYTE *byCirDir*,
 WORD *wGroupIndex*)**

Description On the plane of **Z-X** axis, from current position execute a circular motion with the center designated by its coordinates.
 If this function is called successfully, command counter



value will increase by 1.

Parameters	<i>dfCZ, dfCX</i>	Z-X axis coordinates of designated circle center.
	<i>byCirDir</i>	motion direction, 0: CW , 1: CCW
	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	Greater than or equal to zero	CommandIndex value set by MCCL.
	Less than zero	Fail , meaning of returned value, please refer to V. Function return value.

H. Jog motion

1. int MCC_JogPulse((int *nPulse*, char *cChannel*, WORD *wGroupIndex*)

Description Pulse jog(pulse motion).This function works only after other motion stops.(Use the return value of MCC_GetMotionStatus() to check, 1=motion stop). Since the execution of this motion function has no acceleration or deceleration, displacement setting of pulse cannot be too large.

Parameters	<i>nPulse</i>	Set pulse displacement,unit is pulse,range is -2048~2048.
	<i>nChannel</i>	Number(0 ~ 3)of designated jog axis.
	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to V. Function return value.

2. int MCC_JogSpace(double *dfOffset*, int *nRatio*, char *cChannel*, WORD *wGroupIndex*)

Description Short stroke jog. The way to set speed ratio is similar to that of point to point motion. If this function is called successfully,

command counter value will increase by 1.

Parameters	<i>dfOffset</i>	Designated displacement, displacement unit is mm or inch (Defined by MCC_SetUnit()).
	<i>nRatio</i>	Maximum speed percentage. If <i>ratio</i> < 0, set to 0 automatically; if <i>ratio</i> > 100, then set to 100 automatically.
	<i>nChannel</i>	Number(0 ~ 3) of designated jog axis.
	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	Greater than or equal to zero	CommandIndex value set by MCCL.
	Less than zero	Fail , meaning of returned value, please refer to V. Function return value.

3. int MCC_JogConti(int *nDir*, int *nRatio*, char *cChannel*, WORD *wGroupIndex*)

Description	Continuous jog. The way to set speed ratio is similar to that of point to point motion. When it moves to boundary position, it stops(boundary position is defined in mechanism parameter). If this function is called successfully, command counter value will increase by 1.	
Parameters	<i>nDir</i>	Jog direction□ 1 positive 2 negative
	<i>nRatio</i>	Maximum speed ratio. If <i>ratio</i> < 0, set to 0 automatically; if <i>ratio</i> > 100, then set to 100 automatically.
	<i>nChannel</i>	Number(0 ~ 3) of designated jog axis.
	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	Greater than or equal to zero	CommandIndex value set by MCCL.
	Less than zero	Fail , meaning of returned value, please refer to V.

Function return value.

I. Motion status

1. int MCC_GetMotionStatus(WORD *wGroupIndex*)

Description	Get motion status.		
Parameters	<i>wGroupIndex</i> group number(0 ~ 71)		
Return Value	0	System in motion. There are still unexecuted motion commands.	
	1	System stops. There is no unexecuted motion commands.	
	2	System pauses due to MCC_HoldMotion()	
	Others	Fail , meaning of returned value, please refer to V. Function return value.	

2. int MCC_GetCurCommand(COMMAND_INFO **pstCurCmdInfo*, WORD *wGroupIndex*)

Description	Get information of motion command in execution. Use this to get command type, index, feeding speed and target position.		
Parameters	* <i>pstCurCmdInfo</i>	This is used to save command information, and is defined as	

```
typedef struct _COMMAND_INFO
{
    int      nType;
    int      nCommandIndex;
    double   dfFeedSpeed;
    double   dfPos[6];
} COMMAND_INFO;
```

in which

nType—motion command type

- 0 point to point motion
- 1 linear motion
- 2 CW arc or circular motion
- 3 CCW arc or circular motion

nCommandIndex—motion command index

dfFeedSpeed—For general motion—feeding speed.

For point to point motion—speed ratio.

dfPos[]—target position in absolute coordinate

wGroupIndex group number(0 ~ 71)

Return Value 0 Succeed

Non zero Fail , meaning of returned value, please refer to

V. Function return value.

3. int MCC_GetCommandCount(int *pnCmdCount, WORD wGroupIndex)

Description Read unexecuted command count in buffer. The buffer size is currently 10000. Using this function also provides how much vacant space in the buffer. When motion functions are called, command index is dispatched. This includes all general motions(linear, arc, circular motion)

MCC_PtP()

MCC_JogSpace()

MCC_JogConti()

MCC_EnableBlend()

MCC_DisableBlend()

MCC_DelayMotion()



Parameters **pnCmdCount* unexecuted command count
 wGroupIndex group number(0 ~ 71)
 Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to

V. Function return value.

4. int MCC_ResetCommandIndex(WORD *wGroupIndex*)

Description Reset command index. Command index is a ID-number for each motion. Use this function to restart index from 0.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to

V. Function return value.

5. int MCC_GetErrorCode(WORD *wGroupIndex*)

Description Get current error code to know if any error in machine 's operation.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value Error code (Please refer to **IV.message code**)

6. int MCC_ClearError(WORD *wGroupIndex*)

Description When system error occurred, please remove the error cause and call this function to clear the error record in the system; otherwise system does not resume to normal situation.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to **V. Function return value.**

J. Homing



1. int MCC_SetGoHomeAccStep(int *nStep0*, int *nStep1*, int *nStep2*, int *nStep3*, int *nStep4*, int *nStep5*)

Description Set acceleration, deceleration steps in homing for each axis, i.e.
 <acceleration, deceleration time > = <interpolation time> *
 <acceleration, deceleration steps>.

Parameters *nStep0~nStep3* Set acceleration, deceleration steps values
 for each axis. Values must be greater than
 or equal to 1.

nStep4~nStep5 Not used.

wGroupIndex group number(0 ~ 71)

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please
 refer to **V. Function return value.**

2. int MCC_GetGoHomeAccStep(int **pnStep0*, int **pnStep1*, int **pnStep2*, int **pnStep3*, int **pnStep4*, int **pnStep5*)

Description Get current acceleration, deceleration steps in homing of each
 axis.

Parameters **pnStep0~pnStep3* acceleration, deceleration steps in
 homing of each axis.

**pnStep4~pnStep5* Not used.

wGroupIndex group number(0 ~ 71)

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please
 refer to **V. Function return value.**

3. int MCC_GoHome(double *dfXSpeed*, double *dfYSpeed*, double *dfZSpeed*, double *dfUSpeed*, double *dfVSpeed*, double *dfWSpeed*, int *nXOrder*, int *nYOrder*, int *nZOrder*, int *nUOrder*, int *nVOrder*, int *nWOrder*, WORD *wCardIndex*)

Description Execute homing. Call MCC_GetGoHomeStatus() to check if
 homing is done. After homing is finished, the cartesian



coordinates of each axis will be set to zero.

Parameters *dfXSpeed~dfUSpeed* Feeding speed of each axis, unit is mm/sec.

dfVSpeed~dfWSpeed Not used.

nXOrder~nUOrder Homing order (0~5). If no homing is needed for the axis, please set 0xff.

nVOrder~nWOrder Not used.

wCardIndex Number(0 ~ 11) of motion control card.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to **V. Function return value.**

4. int MCC_GetGoHomeStatus()

Description After call MCC_GoHome(), check with this function if machine has finished homing process.

Return Value 0 Not finished homing yet.
 1 Finished homing process.
 Non zero Fail , meaning of returned value, please refer to **V. Function return value.**

5. int MCC_AbortGoHome()

Description After call MCC_GoHome(), call this function to abort homing.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to **V. Function return value.**

6. int MCC_GetHomeSensorStatus(WORD *pwStatus, WORD wChannel, WORD wCardIndex)

Description Get home sensor status. Notice that home sensor type (normal open or normal close) have to be set properly in mechanism parameter(*stHome.wSensorMode*).(HOM0~HOM1~HOM2~HOM3)

Parameters	<i>*pwStatus</i>	Home sensor status 1 means home sensor is engaged, 0 means not.
	<i>wChannel</i>	Number (0 ~ 3) of motion axis.
	<i>wCardIndex</i>	Number(0 ~ 11) of motion control card.
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to
V. Function return value.		

K. Velocity blending and speed override

1. int MCC_HoldMotion(WORD *wGroupIndex*)

Description	Hold motion. Use this function only when system in motion. After calling this, system brakes with constant deceleration.	
Parameters	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to
V. Function return value.		

2. int MCC_ContiMotion(WORD *wGroupIndex*)

Description	After motion is on hold, use this function to continue unfinished motion command. This function is only of meaning, when a motion was hold.	
Parameters	<i>wGroupIndex</i>	group number(0 ~ 71)
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to
V. Function return value.		

3. int MCC_AbortMotion (WORD *wGroupIndex*)

Description	Abort current motion command in execution.	
Parameters	<i>wGroupIndex</i>	group number(0 ~ 71)

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to
 V. Function return value.

4. int MCC_EnableBlend(WORD *wGroupIndex*)

Description Enable velocity blend function. If this function is called successfully, command counter value will increase by 1.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value Greater than or equal to zero CommandIndex value set by MCCL.
 Less than zero Fail , meaning of returned value, please refer to **V. Function return value.**

5. int MCC_DisableBlend(WORD *wGroupIndex*)

Description Disable velocity blend function. If this function is called successfully, command counter value will increase by 1.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value Greater than or equal to zero CommandIndex value set by MCCL.
 Less than zero Fail , meaning of returned value, please refer to **V. Function return value.**

6. int MCC_CheckBlend(WORD *wGroupIndex*)

Description Check if set velocity blend function.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value 0 Blend enabled.
 1 Blend disabled.
 Others Fail , meaning of returned value, please refer to
 V. Function return value.

7. int MCC_DelayMotion(unsigned int *dwTime*, WORD *wGroupIndex*)

Description	Force to delay before next motion command is executed. If this function is called successfully, command counter value will increase by 1.		
Parameters	<i>dwTime</i>	Delay time. Unit is one interpolation time. For example, if <i>dwTime</i> = 50 and interpolation time = 5ms, then delay time is 250 ms.	
	<i>wGroupIndex</i>	group number(0 ~ 71)	
Return Value	Greater than or equal to zero	CommandIndex	value set by MCCL.
	Less than zero	Fail , meaning of returned value, please refer to V. Function return value.	

8. int MCC_CheckDelay(WORD *wGroupIndex*)

Description	Check if now in delay phase.		
Parameters	<i>wGroupIndex</i>	group number(0 ~ 71)	
Return Value	0	Not in delay phase.	
	1	In delay phase.	
	Others	Fail , meaning of returned value, please refer to V. Function return value.	

9. int MCC_SetOverSpeed(int *nRate*, WORD *wGroupIndex*)

Description	Set speed override ratio of line, arc or circular motion. This function must be called in constant speed region.		
Parameters	<i>nRate</i>	Speed override ratio. New feeding speed becomes ($dfFeedSpeed \times nRate / 100$) in which <i>dfFeedSpeed</i> is the original feeding speed set by <i>MCC_SetFeedSpeed()</i> . <i>nRate</i> must be ≥ 1 . However, new feeding speed will never be greater than the speed set by <i>MCC_SetSysMaxSpeed()</i> .	
	<i>wGroupIndex</i>	group number(0 ~ 71)	
Return Value	Greater than or equal to 1	Actually set speed override ratio.	

Others Fail , meaning of returned value, please refer
 to **V. Function return value.**

10. int MCC_GetOverSpeed(WORD *wGroupIndex*)

Description Get speed override ratio of line, arc, or circular motion.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value Greater than or equal to 1 Actually set speed override ratio.
 Others Fail , meaning of returned value, please refer
 to **V. Function return value.**

11. int MCC_SetPtPOverSpeed(int *nRate*, WORD *wGroupIndex*)

Description Set speed override ratio for point to point motion. This function
 must be called in constant speed region.

Parameters *nRate* Speed override ratio for point to point motion.
 New feeding speed becomes ($dfFeedSpeed \times nRate / 100$) in which *dfFeedSpeed* is the
 original feeding speed set by
 MCC_SetPtPSpeed(). *nRate* must be ≥ 1 .
 However, new feeding speed will never be
 greater than the speed set by
 MCC_SetSysMaxSpeed().
wGroupIndex group number(0 ~ 71)

Return Value Greater than or equal to 1 Actually set speed override ratio.
 Others Fail , meaning of returned value,
 please refer to **V. Function return
 value.**

12. int MCC_GetPtPOverSpeed(WORD *wGroupIndex*)

Description Get speed override ratio in point to point motion.

Parameters *wGroupIndex* group number(0 ~ 71)

Return Value Greater than or equal to 1 Actually set speed override ratio
 of point to point motion.
 Others Fail , meaning of returned



value, please refer to **V.**
Function return value.

L. Encoder

1. int MCC_SetENCInputRate(WORD *wInputRate*,

WORD *wChannel*, WORD *wCardIndex*)

Description	Set encoder mode (1x,2x,4x). The default is 4x.	
Parameters	<i>wInputRate</i>	Set encoder mode. Valid values are 1~2~4~
	<i>wChannel</i>	Number (0 ~ 3) of motion axis.
	<i>wCardIndex</i>	Number(0 ~ 11) of motion control card.
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to V. Function return value.

2. int MCC_ClearENCCounter(WORD *wChannel*, WORD *wCardIndex*)

Description	Reset encoder counter.	
Parameters	<i>wChannel</i>	Number (0 ~ 3) of motion axis.
	<i>wCardIndex</i>	Number(0 ~ 11) of motion control card.
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to V. Function return value.

3. int MCC_GetENCValue(long **plValue*, WORD *wChannel*,

WORD *wCardIndex*)

Description	Read encoder counter.	
Parameters	* <i>plValue</i>	Encoder counter.
	<i>wChannel</i>	Number (0 ~ 3) of motion axis.
	<i>wCardIndex</i>	Number(0 ~ 11) of motion control card.
Return Value	0	Succeed
	Non zero	Fail , meaning of returned value, please refer to

V. Function return value.

**4. int MCC_GetENCIndexStatus(WORD *pwStatus,
WORD wChannel, WORD wCardIndex)**

Description Read index signal.

Parameters **pwStatus* Index signal status.
 wChannel Number (0 ~ 3) of motion axis.
 wCardIndex Number(0 ~ 11) of motion control card.

Return Value 0 Succeed
 Non zero Fail , meaning of returned value, please refer to

V. Function return value.

IV. ERROR CODE

error code	explanation
0xf104	Illegal parameter in arc motion command.
0xf202	Error counter overflow.
0xf301	X axis coordinate out of range in mechanism parameter.
0xf302	Y axis coordinate out of range in mechanism parameter.
0xf303	Z axis coordinate out of range in mechanism parameter.
0xf304	U axis coordinate out of range in mechanism parameter.
0xf401	Error occurred in execution of arc command.
0xf203	Feeding speed too fast, exceed the set pulse maximum speed.
0xf204	Acceleration too big.

V. FUNCTION RETURN VALUES

Symbol	Value	Explanation
NO_ERR	0	Call function succeeded.
INITIAL_MOTION_ERR	-1	System failed to open properly, please call MCC_InitSystem() again.
COMMAND_BUFFER_FULL_ERR	-2	Command buffer is full, cannot receive this command.
COMMAND_NOTACCEPTED_ERR	-3	System in busy state, cannot receive this command.
COMMAND_NOTFINISHED_ERR	-4	Former command not finished, cannot receive this command.
PARAMETER_ERR	-5	Format error in passing parameter to function.
GROUP_PARAMETER_ERR	-6	Error in given group parameter, group designated is invalid.
FEED_RATE_ERR	-7	Feeding speed not set or wrongly set. Please recall MCC_SetFeedSpeed() function.
BLEND_COMMAND_NOTCALLED_ERR	-8	Call to MCC_DisableBlend() not successful.
HOME_COMMAND_NOTCALLED_ERR	-10	Call to MCC_GoAbortHome() not successful.
HOLD_ILLEGAL_ERR	-11	Issuing hold command at wrong time.
CONTI_ILLEGAL_ERR	-12	Issuing continue command at wrong time.

ABORT_ILLEGAL_ERR	-13	Issuing abort command at wrong time.
RUN_TIME_ERR	-14	Run time error. Use MCC_GetErrorCode() to check further error causes.

VI. MCCL INITIAL SETTINGS

After calling MCC_InitSystem(), the initial settings of MCCL are as below. If these initial settings cannot satisfy user's need, please call related function to change.

Contents	Settings	Related Function
Type of system coordinate	Absolute coord.	MCC_SetAbsolute() MCC_SetIncrease()
Unit of displacement	mm	MCC_SetUnit() MCC_GetCoordType()
Max pulse speed	30000 Pulses	MCC_SetMaxPulseSpeed() MCC_GetMaxPulseSpeed()
Max pulse acceleration	30000 Pulses	MCC_SetMaxPulseAcc() MCC_GetMaxPulseAcc()
System max feeding speed	0	MCC_SetSysMaxSpeed() MCC_GetSysMaxSpeed()
Software limit check	enabled	MCC_SetOverTravelCheck() MCC_GetOverTravelCheck
Limit switch check	disabled	MCC_EnableLimitSwitchCheck() MCC_DisableLimitSwitchCheck
Type of acceleration/deceleration for each axis in general motion	S curve	MCC_SetAccType() MCC_GetAccType() MCC_SetDecType() MCC_GetDecType()
Acceleration/deceleration steps for each axis in general motion (Time for each step is 1 interpolation time)	20	MCC_SetAccStep() MCC_GetAccStep() MCC_SetDecStep() MCC_GetDecStep()
Feeding speed for general motion	0	MCC_SetfeedSpeed()

		MCC_GetFeedSpeed()
Type of acceleration/deceleration for each axis in point to point motion	S curve	MCC_SetPtPAccType() MCC_GetPtPAccType() MCC_SetPtPDecType() MCC_GetPtPDecType()
Acceleration/deceleration steps for each axis in point to point motion (Time for each step is 1 interpolation time)	20	MCC_SetPtPAccStep() MCC_GetPtPAccStep() MCC_SetPtPDecStep() MCC_GetPtPDecStep()
Maximum speed ratio for point to point motion	50	MCC_SetPtPSpeed() MCC_GetPtPSpeed()
Acceleration/deceleration steps for each axis in homing (Time for each step is 1 interpolation time)	10	MCC_SetGoHomeAccStep() MCC_GetGoHomeAccStep()
Velocity blending	disabled	MCC_EnableBlend() MCC_DisnableBlend()