IMP Series

Motion Control Command Library

Reference Manual

Version: V.1.01

Date: 2013.01

http://www.epcio.com.tw

Table of Contents

I. Motion Control Cammand Library Function Table

A. System Functions

No.	Command Name	Description
1	MCC_GetVersion()	Acquires version of command library
2	MCC_CreateGroup()	Creates a new motion group
3	MCC_CloseGroup()	Closes the indicated motion group
4	MCC_CloseAllGroups()	Closes all motion groups
5	MCC_SetMacParam()	Sets mechanism parameters
6	MCC_GetMacParam()	Acquires mechanism parameters
7	MCC_SetEncoderConfig()	Sets encoder parameters
8	MCC_GetEncoderConfig()	Acquires encoder parameters
9	MCC_SetHomeConfig()	Sets Go Home parameters
10	MCC_GetHomeConfig()	Acquires Go Home parameters
11	MCC_UpdateParam()	Makes the system become effective from updated parameters
12	MCC_SetCmdQueueSize()	Sets motion command queue size
13	MCC_GetCmdQueueSize()	Acquires motion command queue size
14	MCC_InitSystem()	Enabling the Motion Control Command Library
15	MCC_CloseSystem()	Disabling the Motion Control Command Library
16	MCC_ResetMotion()	Resets the motion control command library
17	MCC_EnableDryRun()	Enables motion dry run function
18	MCC_DisableDryRun()	Disables motion dry run function
19	MCC_CheckDryRun()	Checks motion dry run function status
20	MCC_SetSysMaxSpeed()	Sets maximum feed speed of general motion
21	MCC_GetSysMaxSpeed()	Acquires maximum feed speed of general motion

B. Local Input/Output Control

No.	Command Name	Description
1	MCC_SetServoOn()	Enables servo system
2	MCC_SetServoOff()	Disables servo system
3	MCC_EnablePosReady()	Enables position output signal to be ready
4	MCC_DisablePosReady()	Disables position output signal to be ready
5	MCC_GetEmgcStopStatus()	Acquires emergency stop switch input status

6	MCC_SetLIORoutine()	Sets customized interrupt service routine(ISR) of local input/output (LIO)
7	MCC_SetLIOTriggerType()	Sets trigger type of local input
8	MCC_EnableLIOTrigger()	Enables *local I/O interrupt* function.
9	MCC_DisableLIOTrigger()	Disables *local I/O interrupt* function.
10	MCC_SetLedLightOn()	Enables LED outputs
11	MCC_SetLedLightOff()	Disables LED outputs
12	MCC_GetLedLightStatus()	Acquires current LED output status

C. Positioning System

No.	Command Name	Description
1	MCC_SetAbsolute()	Uses absolute position
2	MCC_SetIncrease()	Uses incremental position
3	MCC_GetCoordType()	Acquires position type used
4	MCC_GetCurRefPos()	Acquires each axial position in Cartesian coordinate system (excluding compensation)
5	MCC_GetCurPos()	Acquires each axial position in Cartesian coordinate system (including compensation)
6	MCC_GetPulsePos()	Acquires each axial position in pulse count (including compensation)
7	MCC_DefineOrigin()	Defines current position as home
8	MCC_DefinePosHere()	Sets the current coordinate position as the actual machine position
9	MCC_DefinePos()	Sets current coordinate position value

D. Over-Travel Protection

No.	Command Name	Description
1	MCC_EnableLimitSwitchCheck()	Enables hardware limit switch protection function
2	MCC_DisableLimitSwitchCheck()	Disables hardware limit switch protection function
3	MCC_SetOverTravelCheck()	Sets software over-travel protection function
4	MCC_GetOverTravelCheck()	Acquires software over-travel protection settings
5	MCC_GetLimitSwitchStatus()	Acquires hardware limit switch status
6	MCC_GetLimitSwitchLatchStatus()	Acquires latch status of hardware limit switch
7	MCC_ClearLimitSwitchLatchStatus()	Deletes latch status of hardware limit switch

E. Linear, Curved, Circular and Helix Motion (General Motion)

No.	Command Name	Description
1	MCC_SetAccType()	Sets acceleration type
2	MCC_GetAccType()	Acquires acceleration type used
3	MCC_SetDecType()	Sets deceleration type
4	MCC_GetDecType()	Acquires deceleration type used
5	MCC_SetAccTime()	Sets acceleration time
6	MCC_GetAccTime()	Acquires acceleration time
7	MCC_SetDecTime()	Sets deceleration time
8	MCC_GetDecTime()	Acquires deceleration time
9	MCC_SetFeedSpeed()	Sets feed speed
10	MCC_GetFeedSpeed()	Acquires feed speed used
11	MCC_GetCurFeedSpeed()	Acquires current mechanism feed speed
12	MCC_GetSpeed()	Acquires current feed speed of each axis
13	MCC_Line()	8-axes simultaneous linear motion
14	MCC_ArcXYZ()	Three-point curve motion on XYZ plane
15	MCC_ArcXYZ_Aux()	Three-point curve motion on XYZ plane with linear motion along assistance axis
16	MCC_ArcXY()	Curve motion on XY plane
17	MCC_ArcYZ()	Curve motion on YZ plane
18	MCC_ArcZX()	Curve motion on ZX plane
19	MCC_ArcXY_Aux ()	Curve motion on XY plane with linear motion along assistance axis
20	MCC_ArcYZ_Aux ()	Curve motion on YZ plane with linear motion along assistance axis
21	MCC_ArcZX_Aux ()	Curve motion on ZX plane with linear motion along assistance axis
22	MCC_ArcThetaXY()	Curve motion on XY plane (with rotational angle as a parameter)
23	MCC_ArcThetaYZ()	Curve motion on YZ plane (with rotational angle as a parameter)
24	MCC_ArcThetaZX()	Curve motion on ZX plane (with rotational angle as a parameter)
25	MCC_CircleXY()	Complete circular motion on XY plane
26	MCC_CircleYZ()	Complete circular motion on YZ plane
27	MCC_CircleZX()	Complete circular motion on ZX plane
28	MCC_CircleXY_Aux ()	Complete circular motion on XY plane with linear motion along assistance axis
29	MCC_CircleYZ_Aux ()	Complete circular motion on YZ plane with linear motion along assistance axis
30	MCC_CircleZX_Aux ()	Complete circular motion on ZX plane with

		linear motion along assistance axis
31	MCC_HelicalXY_Z()	Helix motion with circular motion on XY plane
32	MCC_HelicalYZ_X()	Helix motion with circular motion on YZ plane
33	MCC_HelicalZX_Y()	Helix motion with circular motion on ZX plane
34	MCC_HelicalXY_Z_Aux ()	Helix motion with circular motion on XY plane with linear motion along assistance axis
35	MCC_HelicalYZ_X_Aux ()	Helix motion with circular motion on YZ plane with linear motion along assistance axis
36	MCC_HelicalZX_Y_Aux ()	Helix motion with circular motion on ZX plane with linear motion along assistance axis

F. Point-to-Point Motion

No.	Command Name	Description
1	MCC_SetPtPSpeed()	Sets point-to-point speed ratio
2	MCC_GetPtPSpeed()	Acquires point-to-point speed ratio used
3	MCC_PtP()	Point-to-point motion
4	MCC_PtPX()	Point-to-point motion on X axis
5	MCC_PtPY()	Point-to-point motion on Y axis
6	MCC_PtPZ()	Point-to-point motion on Z axis
7	MCC_PtPU()	Point-to-point motion on U axis
8	MCC_PtPV()	Point-to-point motion on V axis
9	MCC_PtPW()	Point-to-point motion on W axis
10	MCC_PtPA()	Point-to-point motion on A axis
11	MCC_PtPB()	Point-to-point motion on B axis
12	MCC_SetPtPAccType()	Sets acceleration type
13	MCC_GetPtPAccType()	Acquires acceleration type used
14	MCC_SetPtPDecType()	Sets deceleration type
15	MCC_GetPtPDecType()	Acquires deceleration type used
16	MCC_SetPtPAccTime()	Sets acceleration time
17	MCC_GetPtPAccTime()	Acquires acceleration time used
18	MCC_SetPtPDecTime()	Sets deceleration time
19	MCC_GetPtPDecTime()	Acquires deceleration time used

G. Any Curve Motion

No.	Command Name	Description
1	MCC_CustomMotion()	Customized curve motion
2	MCC_CustomMotionEx()	Customized curve motion with blending

H. JOG Motion

No.	Command Name	Description
1	MCC_JogPulse()	Jog motion(pulse)
2	MCC_JogSpace()	Jog motion (UU: user unit)
3	MCC_JogConti()	Continuous jog motion (UU: user unit)

I. Motion Status Check

No.	Command Name	Description
1	MCC_GetMotionStatus()	Acquires current motion status
2	MCC_GetCurCommand()	Acquires information related to the motion command being executed
3	MCC_GetCommandCount()	Acquires the number of motion command in storage
4	MCC_ResetCommandIndex()	Resets motion command index number to 0
5	MCC_GetCurPulseStockCount()	Acquires the current quantity of FMC(fine movement command) used in hardware FIFO
6	MCC_SetMaxPulseStockNum()	Sets the maximum quantity of FMC used in hardware FIFO
7	MCC_GetMaxPulseStockNum()	Acquires the maximum quantity of FMC used in hardware FIFO
8	MCC_GetErrorCode()	Acquires existing error codes
9	MCC_ClearError()	Deletes existing error codes

J. Go Home

No.	Command Name	Description
1	MCC_Home()	Go Home motion
2	MCC_GetGoHomeStatus()	Confirms completion of Go Home motion
3	MCC_AbortGoHome()	Stops Go Home motion
4	MCC_GetHomeSensorStatus()	Acquires home sensor status

K. In Position Control

No.	Command Name	Description
1	MCC_SetCompParam()	Sets parameters for gear backlash and gap

		compensation
2	MCC_UpdateCompParam()	Responds to updated parameters for gear backlash and gap compensation
3	MCC_SetPGain()	Sets proportional gain used in position closed loop control
4	MCC_GetPGain()	Acquires proportional gain used in position closed loop control
5	MCC_SetIGain()	Sets integral gain used in closed loop control
6	MCC_GetIGain()	Acquires integral gain used in closed loop control
7	MCC_SetDGain()	Sets differential gain used in closed loop control
8	MCC_GetDGain()	Acquires differential gain used in closed loop control
9	MCC_SetFGain()	Sets feedforward gain used in closed loop control
10	MCC_GetFGain()	Acquires feedforward gain used in closed loop control
11	MCC_SetMaxPulseSpeed()	Sets maximum pulse for each axis
12	MCC_GetMaxPulseSpeed()	Acquires maximum pulse for each axis
13	MCC_SetMaxPulseAcc()	Sets maximum pulse acceleration for each axis
14	MCC_GetMaxPulseAcc()	Acquires maximum pulse acceleration for each axis
15	MCC_SetInPosMode()	Sets the in position confirmation mode
16	MCC_SetInPosMaxCheckTime()	Sets the in position check time
17	MCC_SetInPosSettleTime()	Sets the in position settle time
18	MCC_EnableInPos()	Enables the in position confirmation function
19	MCC_DisableInPos()	Disables the in position confirmation function
20	MCC_SetInPosToleranceEx()	Sets tolerance of “in position” function
21	MCC_GetInPosToleranceEx()	Acquires tolerance of “in position” function
22	MCC_GetInPosStatus()	Check whether actual position satisfies the in position confirmation status
23	MCC_EnableTrackError()	Enables tracking error check function
24	MCC_DisableTrackError()	Disables tracking error check function
25	MCC_SetTrackErrorLimit()	Sets tolerance of tracking error
26	MCC_GetTrackErrorLimit()	Acquires tolerance of tracking error
27	MCC_SetPCLRoutine()	Sets customized ISR of PCL(Position Control Loop)
28	MCC_SetErrorCountThreshold()	Sets tolerance of pulse error count
29	MCC_GetErrorCount()	Acquires pulse error count for each axis

L. Advanced Trajectory Planning

No.	Command Name	Description
1	MCC_HoldMotion()	Pauses motion
2	MCC_ContiMotion()	Continues motion
3	MCC_AbortMotion()	Stops promptly and aborts all unexecuted motion commands
4	MCC_AbortMotionEx()	Decelerate to stop at the deceleration time and aborts all unexecuted motion commands
5	MCC_EnableBlend()	Enables path blending
6	MCC_DisableBlend()	Disables path blending
7	MCC_CheckBlend()	Checks whether path blending has been enabled
8	MCC_DelayMotion()	Sets motion delay time
9	MCC_CheckDelay()	Checks motion delay status
10	MCC_OverrideSpeed()	Sets the override speed rate for general motion
11	MCC_OverrideSpeedEx()	Sets the override speed rate for general motion with updated timing
12	MCC_GetOverrideRate()	Acquires override speed rate for general motion

M. Encoder Control

No.	Command Name	Description
1	MCC_SetENCRoutine()	Sets customized ISR of ENC(Encoder)
2	MCC_SetENCInputRate()	Sets encoder feedback rate
3	MCC_ClearENCCounter()	Resets encoder counter
4	MCC_GetENCValue()	Acquires encoder counter
5	MCC_SetENCLatchType()	Sets trigger type of encoder counter latch
6	MCC_SetENCLatchSource()	Sets trigger source of encoder counter latch
7	MCC_GetENCLatchValue()	Acquires latch value recorded in the register
8	MCC_EnableENCIndexTrigger()	Enables *encoder index interrupt* function
9	MCC_DisableENCIndexTrigger()	Disables *encoder index interrupt* function
10	MCC_GetENCIndexStatus()	Acquires current encoder index signal status
11	MCC_SetENCCompValue()	Sets encoder comparative value
12	MCC_EnableENCCompTrigger()	Enables *encoder comparative value interrupt* function
13	MCC_DisableENCCompTrigger()	Disables *encoder comparative value interrupt* function

N. Timer and Watchdog Control

No.	Command Name	Description
1	MCC_SetTMRRoutine()	Sets customized ISR of timer
2	MCC_SetTimer()	Sets timer
3	MCC_EnableTimer()	Enables timer
4	MCC_DisableTimer()	Disables timer
5	MCC_EnableTimerTrigger()	Enables *timer interrupt* function
6	MCC_DisableTimerTrigger()	Disables *timer interrupt* function
7	MCC_SetWatchDogTimer()	Sets watchdog timer
8	MCC_SetWatchDogResetPeriod()	Sets watchdog reset signal period
9	MCC_EnableWatchDogTimer()	Enables watchdog
10	MCC_DisableWatchDogTimer()	Disables watchdog
11	MCC_RefreshWatchDogTimer()	Resets watchdog timer

O. Remote Input/Output Control

No.	Command Name	Description
1	MCC_EnableARIOSetControl ()	Enables indicated remote I/O set control
2	MCC_DisableARIOSetControl ()	Disables indicated remote I/O set control
3	MCC_EnableARIOSlaveControl ()	Enables indicated remote I/O slave data transmission
4	MCC_DisableARIOSlaveControl ()	Disables indicated remote I/O slave data transmission
5	MCC_GetARIOTransStatus()	Acquires current remote I/O data transmission status
6	MCC_GetARIOMasterStatus()	Acquires current status of remote I/O master data transmission to slave
7	MCC_GetARIOSlaveStatus()	Acquires current status of remote I/O slave data reception from master
8	MCC_GetARIOInputValue()	Acquires 16-Bit digital input value of indicated set and slave
9	MCC_SetARIOOutputValue()	Sets 16-Bit digital output value indicated set and slave

P. Digital to Analog Converter (DAC) Control

No.	Command Name	Description
1	MCC_SetDACOutput()	Outputs indicated voltage
2	MCC_SetDACTriggerOutput()	Sets preprogrammed voltage output
3	MCC_SetDACTriggerSource()	Sets source to trigger preprogrammed voltage output
4	MCC_EnableDACTriggerMode()	Enables function of triggering output preprogrammed voltage
5	MCC_DisableDACTriggerMode()	Disables function of triggering output preprogrammed voltage
6	MCC_StartDACConv()	Enables voltage output
7	MCC_StopDACConv()	Disables voltage output

Q. Analog to Digital Converter (ADC) Control

No.	Command Name	Description
1	MCC_SetADCRoutine()	Sets customized ISR of ADC
2	MCC_SetADCConvType()	Sets voltage conversion type to unipolar or bipolar
3	MCC_GetADCInput()	Acquires DC input
4	MCC_SetADCCompType()	Sets comparative type of voltage conversion
5	MCC_SetADCCompValue()	Sets comparative value of voltage conversion
6	MCC_GetADCCompValue()	Acquires comparative value used
7	MCC_EnableADCConvChannel ()	Enables selected channel of voltage conversion
8	MCC_DisableADCConvChannel ()	Disables selected channel of voltage conversion
9	MCC_StartADCConv()	Starts voltage conversion
10	MCC_StopADCConv()	Stops voltage conversion

II. Motion Control Command Library

A. System Functions

---

1. void MCC_GetVersion(

char* ***strVersion***

)

Description Acquires version of command library

Parameters *strVersion* A pointer to char memory is used to receive the command library version

---

2. int MCC_CreateGroup(

int ***xMapToCh*****,**

int ***yMapToCh*****,**

int ***zMapToCh*****,**

int ***uMapToCh*****,**

int ***vMapToCh*****,**

int ***wMapToCh*****,**

int ***aMapToCh*****,**

int ***bMapToCh*****,**

int ***nCardIndex***

)

Description This command is used to establish a new motion group.

This command is required to establish a motion group and acquires the new motion group's number as (one of) its input parameter before calling a command related to motion groups in MCCL (Ex: MCC_Line).

This command must be called before MCCL initialization (MCC_InitSystem); meanwhile, please call MCC_CloseAllGroups before calling this command for the first time. Note: Any two motion axes cannot respond to the same physical output channel.

Parameters *xMapToCh* Assigns the physical output channel (0~7) that

		corresponds to X axis in this group
	yMapToCh	Assigns the physical output channel (0~7) that corresponds to Y axis in this group
	zMapToCh	Assigns the physical output channel (0~7) that corresponds to Z axis in this group
	uMapToCh	Assigns the physical output channel (0~7) that corresponds to U axis in this group
	vMapToCh	Assigns the physical output channel (0~7) that corresponds to V axis in this group
	wMapToCh	Assigns the physical output channel (0~7) that corresponds to W axis in this group
	aMapToCh	Assigns the physical output channel (0~7) that corresponds to A axis in this group
	bMapToCh	Assigns the physical output channel (0~7) that corresponds to B axis in this group
	nCardIndex	Assigns the motion control card number (0~5) that corresponds to this group
	***AXIS_INVALID* must be input if the motion axis does not correspond to a physical axis**	
Return Value	>=0	Group number of the newly established group
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_CloseGroup(

int *nGroupIndex*

)

Description	Closes indicated group	
Parameters	*wGroupIndex*	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_CloseAllGroups()

Description	Closes all groups in the system. Please call this command before calling MCC-CreateGroup.	
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_SetMacParam(

SYS_MAC_PARAM* ***pstMacParam*****,**

WORD ***wChannel*****,**

WORD ***wCardIndex,***

)

Description	Sets the mechanism parameter of each axis	
Parameters	*pstMacParam*	A pointer to SYS_MAC_PARAM structure is used to contain desired mechanism parameters
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_GetMacParam(

SYS_MAC_PARAM* ***pstMacParam*****,**

WORD ***wChannel*****,**

WORD ***wCardIndex***

)

Description	Acquires the mechanism parameter content of indicated axis	
Parameters	*pstMacParam*	A pointer to SYS_MAC_PARAM structure is used to receive the desired mechanism parameter
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_SetEncoderConfig(

SYS_ENCODER_CONFIG* ***pstEncoderConfig*****,**

WORD ***wChannel*****,**

WORD ***wCardIndex***

)

Description	Sets encoder parameter content	
Parameters	*pstEncoderConfig*	A pointer to SYS_ENCODER_CONFIG structure is used to contain the desired encoder parameter
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_GetEncoderConfig(

SYS_ENCODER_CONFIG* ***pstEncoderConfig*****,**

WORD ***wChannel*****,**

WORD ***wCardIndex***

)

Description	Acquires the encoder parameter content of indicated axis	
Parameters	*pstEncoderConfig*	A pointer to SYS_ENCODER_CONFIG structure is used to receive the desired encoder parameter content
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

9. int MCC_SetHomeConfig(

SYS_HOME_CONFIG* ***pstHomeConfig*****,**

WORD ***wChannel*****,**

WORD ***wCardIndex***

)

Description	Sets Go Home parameter content	
Parameters	*pstHomeConfig*	A pointer to SYS_HOME_CONFIG structure is used to contain the desired Go Home parameter
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

10. int MCC_GetHomeConfig(

SYS_HOME_CONFIG* ***pstHomeConfig*****,**

WORD ***wChannel*****,**

WORD ***wCardIndex***

)

Description	Acquires Go Home parameter content	
Parameters	*pstHomeConfig*	A pointer to SYS_HOME_CONFIG structure is used to receive Go Home parameters
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

11. int MCC_UpdateParam()

Description	Responds to updated mechanism, encoder and Go Home parameters. If MCC_SetMacParam and MCC_SetEncoderConfig are used again to change related parameters after MCC_InitSystem has been called, the system will only respond to updated settings by using MCC_UpdateParam(). Please note that similar to MCC_ResetMotion, the system will reset to its initial status when this command is called.	
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return**

		Values for the meaning of return value

---

12. int MCC_SetCmdQueueSize(

int ***nSize*****,**

WORD ***wGroupIndex***

)

Description	Sets the size of motion command queue	
Parameters	*nSize*	Size of motion command queue (unit: motion command)
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

13. int MCC_GetCmdQueueSize(

WORD ***wGroupIndex***

)

Description	Acquires the size of motion command queue	
Parameters	*wGroupIndex*	Group number
Return Value	>0 or =0	Size of motion command queue (unit: motion command)
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

14. int MCC_InitSystem(

int ***nInterpolateTime*****,**

SYS_CARD_CONFIG* ***pstCardConfig*****,**

WORD ***wCardNo***

)

Description	Enabling the Motion Control Command Library. It is required to call **MCC_InitSystem** before using other commands in MCCL with exception of the next ones: MCC_CreateGroup、MCC_SetMacParam、MCC_SetEncoderConfig、MCC_SetHomeConfig、MCC_SetCompParam. This command only needs to be called once.	
Parameters	*nInterpolateTime*	Interpolation time; unit: ms; ranging from 1ms ~

	50ms.	
		Shorter interpolation time result in better operational performance, but this depends on the load capacity of CPU. For general system, the setting of 2ms will be appropriate.
	pstCardConfig	Motion control card hardware parameter: for details of hardware parameters, please refer to “IMP Series Motion Control Command Library User Manual”.
	wCardNo	Number of motion control card used (1~6)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

15. int MCC_CloseSystem()

Description	Disables the motion control command library	
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

16. int MCC_ResetMotion()

Description	Resets the motion control command library. This command will clear error status and restore Cartesian coordinate and motor coordinate to zero; finally, the system will return to the initial status after MCC_InitSystem is called.	
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

17. int MCC_EnableDryRun()

Description	Enables motion dry run function. Motion command will still be calculated after this function is enabled but the calculated result will not be output. At this time, MCC_GetCurPos and MCC_GetPulsePos can be used to get positions required for analysis or drawing.	
Return Value	0	Successful

	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

18. int MCC_DisableDryRun()

Description	Disables motion dry run function	
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

19. int MCC_CheckDryRun()

Description	Check motion dry run function status	
Return Value	0	Motion dry run has been enabled
	1	Motion dry run has been disabled
	Other	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

20. int MCC_SetSysMaxSpeed(

double *dfMaxSpeed*

)

Description	Sets the maximum feed speed of general motion (linear, curve, circular and helix) to prevent the feed speed set by using MCC_SetFeedSpeed from exceeding system work limitations. Unit: User Unit/sec (*)	
Parameters	*dfMaxSpeed*	Maximum feed speed
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

21. double MCC_GetSysMaxSpeed()

Description	Acquires the maximum feed speed of general motion (linear, curve, circular and helix); unit: User Unit/sec
Return Value	Maximum feed speed

---

*Note: User Unit (hereafter referred to as UU) is the unit of length (angle) selected (e.g. dfPitch,

dfHighLimit, dfLowLimit) when the user sets mechanism parameters. Once selected, the same unit of length (angle) will be used throughout the MCCL.

B. Local Input/Output Control

---

1. int MCC_SetServoOn(

WORD ***wChannel*****,**

WORD ***wCardIndex*****,**

)

Description	Enables servo system	
Parameters	*wChannel*	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_SetServoOff(

WORD ***wChannel*****,**

WORD ***wCardIndex***

)

Description	Disables servo system	
Parameters	*wChannel*	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_EnablePosReady(

WORD ***wCardIndex***

)

Description	Enables position output signal to be ready	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return**

		Values for the meaning of return value

---

4. int MCC_DisablePosReady(

WORD ***wCardIndex***

)

Description	Disables position output signal to be ready	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_GetEmgcStopStatus(

WORD* ***pwStatus*****,**

WORD ***wCardIndex***

)

Description	Acquires the input status of emergency stop switch. To enable this function, please refer to the setting method of Emergency Stop in the relevant hardware user manual. JP5 is preset to short when delivered from the factory. When Emergency Stop wiring is complete, JP5 must be open to enable Emergency Stop funtion.	
Parameters	*pwStatus*	A pointer to WORD variable is used to receive the emergency stop switch input status
		0 Not triggered
		1 Emergency stop switch triggered
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_SetLIORoutine(

LIOISR ***pfnLIORoutine*****,**

WORD ***wCardIndex***

)

Description	Sets customized ISR of LIO ; for details, please refer to "**IMP Series Motion Control Command Library User Manual**".	
Parameters	*pfnLIORoutine*	Command index for customized ISR of LIO
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_SetLIOTriggerType(

WORD *wTriggerType*,

WORD *wPoint*,

WORD *wCardIndex*

)

Description	Sets trigger type of local input signal to trigger customized ISR . The trigger type can be set as Rising Edge, Falling Edge and Level Change.	
Parameters	*wTriggerType* The trigger type can be set as	
	LIO_INT_RISE	Rising edge
	LIO_INT_FALL	Falling edge
	LIO_INT_LEVEL	Level change
	wPoint	Input number ranges from LIO_LDI_OTP0 ~ LIO_LDI_OTP7, LIO_LDI_OTN0 ~ LIO_LDI_OTN7 and LIO_LDI_HOME0 ~ LIO_LDI_HOME7; for meanings of input signal, please refer to "**IMP Series Motion Control Command Library User Manual**".
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_EnableLIOTrigger(

WORD *wPoint*,

WORD *wCardIndex*

)

Description	Enables *local I/O interrupt* function. The *local I/O interrupt* trigger ISR of LIO	
Parameters	*wPoint*	Local input number ranges from LIO_LDI_OTP0 ~ LIO_LDI_OTP7, LIO_LDI_OTN0 ~ LIO_LDI_OTN7 and LIO_LDI_HOME0 ~ LIO_LDI_HOME7; for meanings of input signal, please refer to "**IMP Series Motion Control Command Library User Manual".**
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

9. int MCC_DisableLIOTrigger(

WORD *wPoint*,

WORD *wCardIndex*

)

Description	Disables *local I/O interrupt* function. The *local I/O interrupt* trigger ISR of LIO	
Parameters	*wPoint*	Local input number ranges from LIO_LDI_OTP0 ~ LIO_LDI_OTP7, LIO_LDI_OTN0 ~ LIO_LDI_OTN7 and LIO_LDI_HOME0 ~ LIO_LDI_HOME7; for meanings of input signal, please refer to "**IMP Series Motion Control Command Library User Manual".**
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

10. int MCC_SetLedLightOn(

WORD *wBit*,

WORD *wCardIndex*

)

Description	Enables LED outputs	
Parameters	*wBit*	Motion Control card LED output bit (1~7)
		Bit 0 represents system indicator number
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

11. int MCC_SetLedLightOff(

WORD *wBit*,

WORD *wCardIndex*

)

Description	Disables LED outputs	
Parameters	*wBit*	Motion Control card LED output bit (1~7)
		Bit 0 represents system indicator number
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

12. int MCC_GetLedLightStatus(

WORD *wBit*,

WORD* *pwStatus*,

WORD *wCardIndex*

)

Description	Acquires current LED output status	
Parameters	*wBit*	Motion control card LED output channel (1~7)
		Bit 0 represents system indicator number
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

C. Positioning System

---

1. int MCC_SetAbsolute(

WORD ***wGroupIndex***

)

Description	Adopts absolute position mode	
Parameters	*wGroupIndex*	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_SetIncrease(

WORD ***wGroupIndex***

)

Description	Adopts incremental position mode	
Parameters	*wGroupIndex*	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_GetCoordType(

WORD ***wGroupIndex***

)

Description	Acquires position mode used	
Parameters	*wGroupIndex*	Group number
Return Value	0	Incremental position
	1	Absolute position
	Other	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_GetCurRefPos(

double* ***pdfX*****,**
double* ***pdfY*****,**
double* ***pdfZ*****,**
double* ***pdfU*****,**
double* ***pdfV*****,**
double* ***pdfW*****,**
double* ***pdfA*****,**
double* ***pdfB*****,**
WORD ***wGroupIndex***

)

Description	Acquires each axial position in Cartesian coordinate system ***(excluding compensation)***	
Parameters	*pdfX ~ pdfB*	A pointer to double variable is used to store the current position for axis X to B in Cartesian coordinate system (excluding compensation)
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_GetCurPos(

double* ***pdfX*****,**
double* ***pdfY*****,**
double* ***pdfZ*****,**
double* ***pdfU*****,**
double* ***pdfV*****,**
double* ***pdfW*****,**
double* ***pdfA*****,**
double* ***pdfB*****,**
WORD ***wGroupIndex***

)

Description	Acquires each axial position in Cartesian coordinate system ***(including compensation)***

Parameters	*pdfX ~ pdfB*	A pointer to double variable is used to store the current position for axis X to B in Cartesian coordinate system (including compensation)
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_GetPulsePos(

long* *plX*,
long* *plY*,
long* *plZ*,
long* *plU*,
long* *plV*,
long* *plW*,
long* *plA*,
long* *plB*,
WORD *wGroupIndex*

)

Description	Acquires each current axial position in pulse count ***(including compensation)***	
Parameters	*plX ~ plB*	A pointer to long variable is used to store the current axial pulse count for axis X to B (including compensation)
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_DefineOrigin(

WORD *wAxis*,
WORD *wGroupIndex*

)

Description	Resets the indicated motion axis position value in a specified group to home; the indicated group motion must be stopped to use this command

Parameters	*wAxis*	Indicated motion axes 0 to 7 respectively represent axes X to B
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_DefinePosHere(

WORD *wGroupIndex* ,

DWORD *dwAxisMask*

)

Description: Sets the current coordinate position as the actual machine position.

The machine can be moved manually under certain circumstances; at this point, a discrepancy will be created between the actual machine position and system position value in the motion control command library. ***If an encoder is installed*** in the system, use the encoder count after successfully using this command to correct the system position. The system position value will show the actual machine position.

Parameters:

wGroupIndex	Group number
dwAxisMask	Indicates the axis that performs the desired action; the assigned parameters can be:

IMP_AXIS_X	X axis
IMP_AXIS_Y	Y axis
IMP_AXIS_Z	Z axis
IMP_AXIS_U	U axis
IMP_AXIS_V	V axis
IMP_AXIS_W	W axis
IMP_AXIS_A	A axis
IMP_AXIS_B	B axis
IMP_AXIS_ALL	All motion axes

The above parameters can be combined freely. Take the operation on X, Z and V axes for an example:

(IMP_AXIS_X | IMP_AXIS_Z | IMP_AXIS_V)

Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

9. int MCC_DefinePos(

WORD ***wAxis,***

DOUBLE ***dfPos*** **,**

WORD ***wGroupIndex***

)

Description	Sets current system position value. The machine may need to be rebooted under certain circumstances; at this point, the encoder position on the control card will be reset to 0. When the MCCL restarts initialization , a discrepancy will be created between the actual machine position and system position value in the motion control command library. ***If an encoder is installed*** in the system, use the absolute encoder count after successfully using this command to correct the system position. The system position value will show the actual machine position.	
Parameters	*wAxis*	Indicated motion axes 0 to 7 respectively represent axes X to B
	dfPos	Sets system position value
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

D. Over-Travel Protection

---

1. int MCC_EnableLimitSwitchCheck(

int *nMode*

)

Description

Enables limit switch protection function. Mechanism parameters *wOverTravelUpSensorMode* and *wOverTravelDownSensorMode* must be set as 0 (Normal Open) or 1 (Normal Close).Once this function is enabled, group motion command output will be stopped (and produced an error record) once the limit switch in the direction of a given axis is triggered (for example, when traveling in a forward direction and triggering a positive limit switch or when traveling in the reverse direction and triggering a negative limit switch). MCC_EnableLimitSwitchCheck() is generally used with MCC_GetErrorCode(). Through continuously calling MCC_GetErrorCode (), the user can find out if the error record is produced when the limit switch is triggered (code 0xF701 to 0xF708 respectively represents the limit switch is triggered by axis X to B). When the error of triggering limit switch is discovered, the general approach would be showing a message on the screen to notify the operator and then call MCC_ClearError() to clear the error so that the system can travel in the opposite direction to move away from the limit switch.

Parameters

nMode Hardware limit switch protection mode. It can be set as:

0 Stops to output motion command of the axis once limit switch is triggered.

1 Stops to output motion command of the axis only when the limit switch in the same direction as the system is triggered (for example, when traveling in a forward direction and triggering a positive limit switch or when traveling in the reverse direction and triggering a negative limit switch).

2 Stops to output motion command of the axis once the

limit switch is triggered; an error record will be produced as well.

3 Stops to output motion command of the axis only when the limit switch in the same direction as the system is triggered (for example, when traveling in a forward direction and triggering a positive limit switch or when traveling in the reverse direction and triggering a negative limit switch) ; an error record will be produced as well.

Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_DisableLimitSwitchCheck()

Description	Disables limit switch protection	
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_SetOverTravelCheck(

int *nCheck0*,
int *nCheck1*,
int *nCheck2*,
int *nCheck3*,
int *nCheck4*,
int *nCheck5*,
int *nCheck6*,
int *nCheck7*,
WORD *wGroupIndex*

)

Description Sets software over-travel protection function

Parameters *nCheck0*, *nCheck1*, *nCheck2*, *nCheck3*, *nCheck4*, *nCheck5*, *nCheck6* and *nCheck7* are setting parameters. 1 indicates to enable the software over-travel

protection of this axis; 0 indicates to disable to this function.

	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_GetOverTravelCheck(

int* ***pnChk0*****,**
int* ***pnChk1*****,**
int* ***pnChk2*****,**
int* ***pnChk3*****,**
int* ***pnChk4*****,**
int* ***pnChk5*****,**
int* ***pnChk6*****,**
int* ***pnChk7*****,**
WORD ***wGroupIndex***

)

Description	Acquires software over-travel protection settings	
Parameters	*pnChk0 ~ pnChk7*	A pointer to int variable is used to store current software over-travel protection settings of axes X to B;1 indicates it has been enabled and 0 indicates disabled.
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_GetLimitSwitchStatus(

WORD* ***pwStatus*****,**
WORD ***wUpDown*****,**
WORD ***wChannel*****,**
WORD ***wCardIndex***

)

Description	Acquires the limit switch status. The limit switch wiring must be accurately defined before using this command. The wiring is defined in the mechanism parameters *wOverTravelUpSensorMode* and *wOverTravelDownSensorMode*。	
Parameters	*pwStatus*	A pointer to WORD variable is used to store the limit switch status. 1 indicates that the limit switch is currently triggered and 0 indicates not triggered.
	wUpDown	0 indicates the negative limit switch status and 1 indicates the positive one
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_GetLimitSwitchLatchStatus(

WORD* *pwStatus*,
WORD *wUpDown*,
WORD *wChannel*,
WORD *wCardIndex*

)

Description	Acquires the limit switch latch status. The limit status will be latched when the limit switch is triggered. This command can be used to acquire the limit switch latch status. The limit switch wiring must be accurately defined before using this command. The wiring is defined in mechanism parameters *wOverTravelUpSensorMode* and *wOverTravelDownSensorMode.*	
Parameters	*pwStatus*	A pointer to WORD variable is used to store limit switch latch status. 1 indicates the limit switch was triggered and 0 indicated not triggered
	wUpDown	0 indicates the negative limit switch latch status and 1 indicates the positive one
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)

Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_ClearLimitSwitchLatchStatus(

WORD *wUpDown*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Deletes the limit switch latch status. The limit status will be latched when the limit switch is triggered. This command can be used to delete the limit switch latch status.	
Parameters	*wUpDown*	0 indicates to delete the negative limit switch latch status and 1 indicates to delete the positive one
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

E. Linear, Curve, Circular and Helix motion (General Motion)

---

1. int MCC_SetAccType(

char ***cAccType*****,**

WORD ***wGroupIndex***

)

Description	Sets acceleration types of general motion	
Parameters	*cAccType*	Acceleration types for each axis; types can be set as: 'T' to use trapezoidal acceleration curve 'S' to use S acceleration curve
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_GetAccType(

WORD ***wGroupIndex***

)

Description	Acquires acceleration types used in general motion	
Parameters	*wGroupIndex*	Group number
Return Value	0	Currently uses trapezoidal acceleration curve
	1	Currently uses S acceleration curve

---

3. int MCC_SetDecType(

char ***cDecType*****,**

WORD ***wGroupIndex***

)

Description	Sets deceleration types of general motion	
Parameters	*cDecType*	Deceleration types for each axis; types can be set as: 'T' to use trapezoidal deceleration curve 'S' to use S deceleration curve

	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_GetDecType(

WORD *wGroupIndex*

)

Description: Acquires deceleration types used in general motion

Parameters: *wGroupIndex* Group number

Return Value	0	Currently uses trapezoidal deceleration curve
	1	Currently uses S deceleration curve

---

5. int MCC_SetAccTime(

double *dfAccTime*,

WORD *wGroupIndex*

)

Description: Sets the time required for general motion to accelerate to a stable speed

Parameters	*dfAccTime*	Required acceleration time; must be greater than 0. Unit: ms.
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. double MCC_GetAccTime(

WORD *wGroupIndex*

)

Description: Acquires the time required for general motion to accelerate to a stable speed

Parameters: *wGroupIndex* Group number

Return Value: The time required for general motion to accelerate to a stable speed; unit: ms.

---

7. int MCC_SetDecTime(

double *dfDecTime*,

WORD ***wGroupIndex***

)

Description	Sets the time required for general motion to decelerate from a stable speed to stop	
Parameters	*dfDecTime*	Required deceleration time; must be greater than 0. Unit: ms.
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. double MCC_GetDecTime(

WORD ***wGroupIndex***

)

Description	Acquires the time required for general motion to decelerate from a stable speed to stop	
Parameters	*wGroupIndex*	Group number
Return Value	Time required for general motion to decelerate from a stable speed to stop ; unit: ms.	

---

9. double MCC_SetFeedSpeed(

double ***dfFeedSpeed*****,**

WORD ***wGroupIndex***

)

Description	Sets the feed speed of general motion; unit: UU/sec. This value shall not be 0. The feed speed for general motion under actual operation (can be obtained by using MCC_GetCurFeedSpeed() must consider whether MCC_OverrideSpeed() is used to set feed speed ratio. For example, if MCC_OverrideSpeed(10) is called when the last feed speed ratio was set using MCC_SetFeedSpeed(150), then the actual feed speed for general motion used is 10×150% = 15	
Parameters	*dfFeedSpeed*	Required feed speed
	wGroupIndex	Group number
Return Value	Actual feed speed set	

---

10. double MCC_GetFeedSpeed(

WORD *wGroupIndex*

)

Description Acquires the feed speed set for general motion. The feed speed obtained through this command is simply the return value of MCC_SetFeedSpeed(), and excludes the impact of MCC_OverrideSpeed() on the actual feed speed. For this part, please refer to the description of MCC_SetFeedSpeed().

Parameters *wGroupIndex* Group number

Return Value Current feed speed set

---

11. double MCC_GetCurFeedSpeed(

WORD *wGroupIndex*

)

Description Acquires the current actual feed speed of machine

Parameters *wGroupIndex* Group number

Return Value The current actual feed speed of machine

---

12. int MCC_GetSpeed(

double* *pdfV0*,

double* *pdfV1*,

double* *pdfV2*,

double* *pdfV3*,

double* *pdfV4*,

double* *pdfV5*,

double* *pdfV6*,

double* *pdfV7*,

WORD *wGroupIndex*

)

Description Acquires current feed speed of each axis

Parameters *pdfV0 ~ pdfV7* A pointer to double variable is used to store current feed speed of each axis

wGroupIndex Group number

Return Value 0 Successful

not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

13. int MCC_Line(

double ***dfX0*****,**
double ***dfX1*****,**
double ***dfX2*****,**
double ***dfX3*****,**
double ***dfX4*****,**
double ***dfX5*****,**
double ***dfX6*****,**
double ***dfX7*****,**
WORD ***wGroupIndex*** **,**
DWORD ***dwAxisMask***

)

Description: Moves from current position to the indicated end point in linear motion Calling this command successfully will increase the number of stored motion commands.

Parameters:

dfX0 ~ dfX7	End point position value
wGroupIndex	Group number
dwAxisMask	Indicates the axis that performs the desired action; the assigned parameters can be:

IMP_AXIS_X	X axis
IMP_AXIS_Y	Y axis
IMP_AXIS_Z	Z axis
IMP_AXIS_U	U axis
IMP_AXIS_V	V axis
IMP_AXIS_W	W axis
IMP_AXIS_A	A axis
IMP_AXIS_B	B axis
IMP_AXIS_ALL	All motion axes

The above parameters can be combined freely. Take the operation on X, Z and V axes for an example:

(IMP_AXIS_X | IMP_AXIS_Z | IMP_AXIS_V)

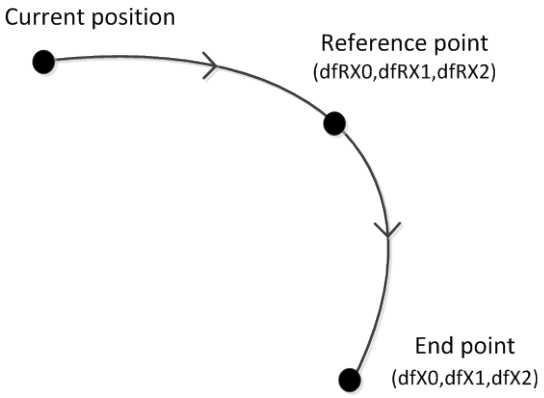
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

14. int MCC_ArcXYZ(

double ***dfRX0*****,**
double ***dfRX1*****,**
double ***dfRX2*****,**
double ***dfX0*****,**
double ***dfX1*****,**
double ***dfX2*****,**
WORD ***wGroupIndex***

)

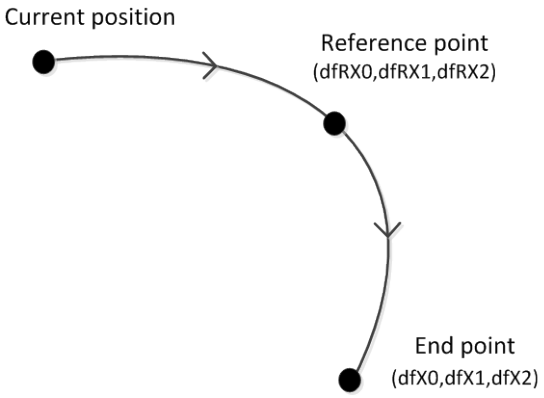


Description	Moves in a curve from current position to the end point through the indicated reference point within the space constructed by XYZ axes. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfRX0 ~dfRX2*	XYZ position of the reference point
	dfX0~dfX2	XYZ position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

15. int MCC_ArcXYZ_Aux(

double ***dfRX0*****,**
double ***dfRX1*****,**
double ***dfRX2*****,**
double ***dfX0*****,**
double ***dfX1*****,**
double ***dfX2*****,**
double ***dfX3*****,**

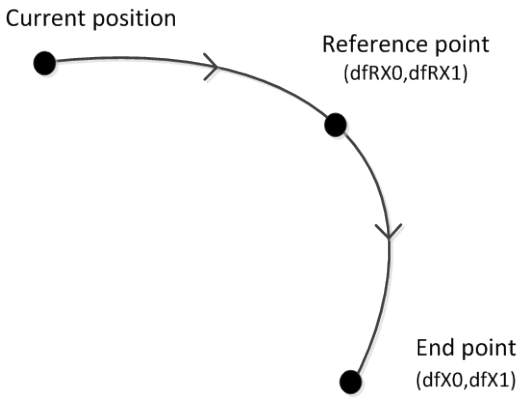


double ***dfX4*****,**
double ***dfX5*****,**
double ***dfX6*****,**
double ***dfX7*****,**
WORD ***wGroupIndex***
)

Description	Moves in a curve from current position to the end point through the indicated reference point within the space constructed by XYZ axes while simultaneously executing linear motion on U, V, W, A and B axes. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfRX0 ~dfRX2*	XYZ position of the reference point
	dfX0~dfX2	XYZ position of the end point
	dfX3~dfX7	UVWAB position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

16. int MCC_ArcXY(
double ***dfRX0*****,**
double ***dfRX1*****,**
double ***dfX0*****,**
double ***dfX1*****,**
WORD ***wGroupIndex***
)



Description	Moves from current position, through indicated reference point to the end point within the plane constructed by XY axes in curve motion. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfRX0, dfRX1*	XY position of the reference point
	dfX0, dfX1	XY position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL

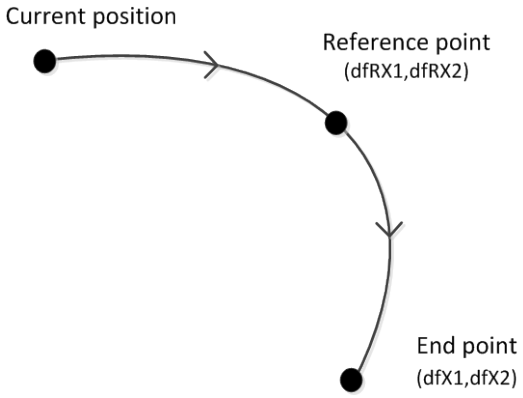
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

17. int MCC_ArcYZ(

double *dfRX1*,
double *dfRX2*,
double *dfX1*,
double *dfX2*,
WORD *wGroupIndex*

)



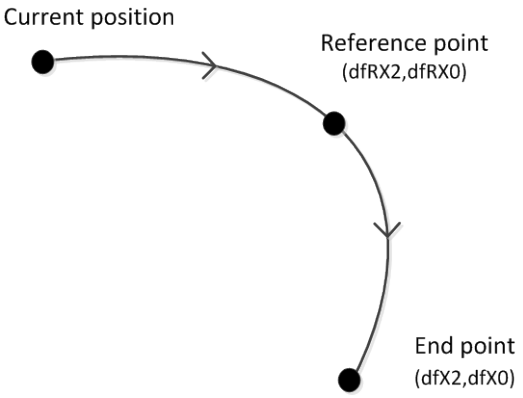
Description	Moves from current position, through indicated reference point to the end point within the plane constructed by YZ axes in curve motion. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfRX1*, *dfRX2*	YZ position of the reference point
	dfX1, *dfX2*	YZ position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

18. int MCC_ArcZX(

double *dfRX2*,
double *dfRX0*,
double *dfX2*,
double *dfX0*,
WORD *wGroupIndex*

)



Description	Moves from current position, through indicated reference point to the end point within the plane constructed by ZX axes in curve motion. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfRX2*, *dfRX0*	ZX position of the reference point

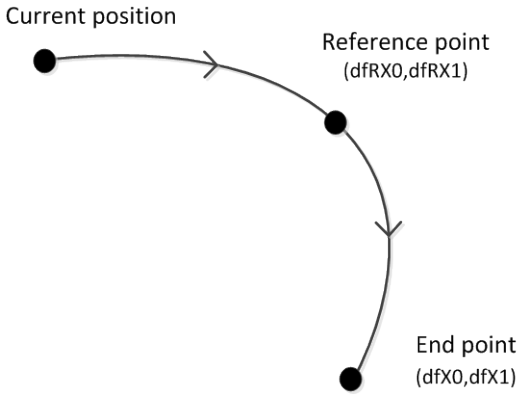
	dfX2, *dfX0*	ZX position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

19. int MCC_ArcXY_Aux(

double *dfRX0*,
double *dfRX1*,
double *dfX0*,
double *dfX1*,
double *dfX3*,
double *dfX4*,
double *dfX5*,
double *dfX6*,
double *dfX7*,
WORD *wGroupIndex*

)



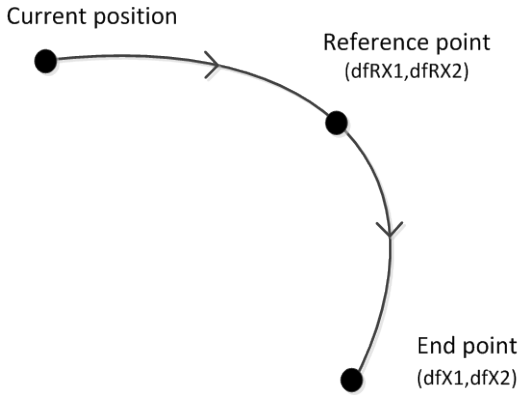
Description	Moves in a curve from current position to the end point through the indicated reference point within the plane constructed by XY axes while simultaneously executing linear motion on U, V, W, A and B axes. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfRX0*, *dfRX1*	XY position of the reference point
	dfX0, *dfX1*	XY position of the end point
	dfX3~dfX7	UVWAB position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

20. int MCC_ArcYZ_Aux(

double *dfRX1*,

double ***dfRX2*****,**
double ***dfX1*****,**
double ***dfX2*****,**
double ***dfX3*****,**
double ***dfX4*****,**
double ***dfX5*****,**
double ***dfX6*****,**
double ***dfX7*****,**
WORD ***wGroupIndex***



)

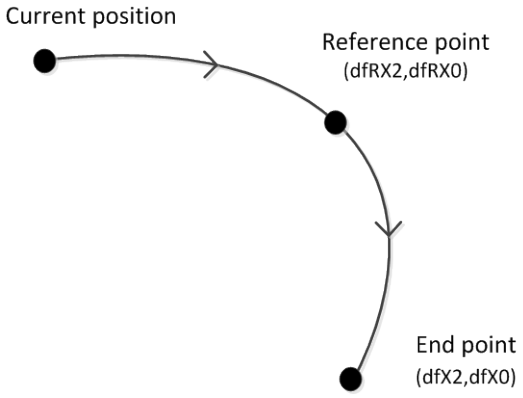
Description: Moves in a curve from current position to the end point through the indicated reference point within the plane constructed by YZ axes while simultaneously executing linear motion on U, V, W, A and B axes. Calling this command successfully will increase the number of stored motion commands.

Parameters		
	dfRX1, *dfRX2*	YZ position of the reference point
	dfX1, *dfX2*	YZ position of the end point
	dfX3~dfX7	UVWAB position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

21. int MCC_ArcZX_Aux(

double ***dfRX2*****,**
double ***dfRX0*****,**
double ***dfX2*****,**
double ***dfX0*****,**
double ***dfX3*****,**
double ***dfX4*****,**
double ***dfX5*****,**
double ***dfX6*****,**
double ***dfX7*****,**
WORD ***wGroupIndex***



)

Description	Moves in a curve from current position to the end point through the indicated reference point within the plane constructed by ZX axes while simultaneously executing linear motion on U, V, W, A and B axes. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfRX2*, *dfRX0*	ZX position of the reference point
	dfX2, *dfX0*	ZX position of the end point
	dfX3~dfX7	UVWAB position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

22. int MCC_ArcThetaXY(

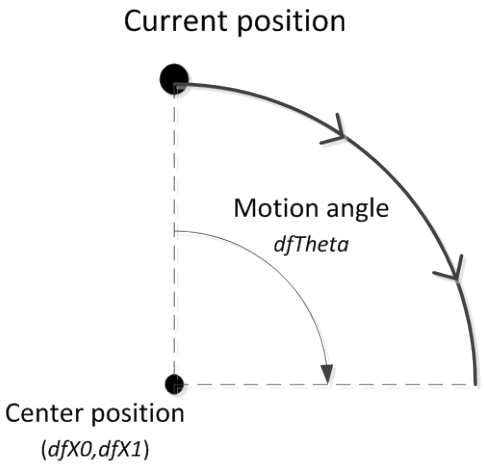
double *dfX0*,

double *dfX1*,

double *dfTheta*,

WORD *wGroupIndex*

)



Description	Moves in a curve around the indicated center with the indicated angle in the plane constructed by XY axes. A negative angle indicates operating clockwise motion; a positive angle indicates operating counter-clockwise motion. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfX0*, *dfX1*	Center position in XY cooidinate
	dfTheta	Motion angle
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

23. int MCC_ArcThetaYZ(

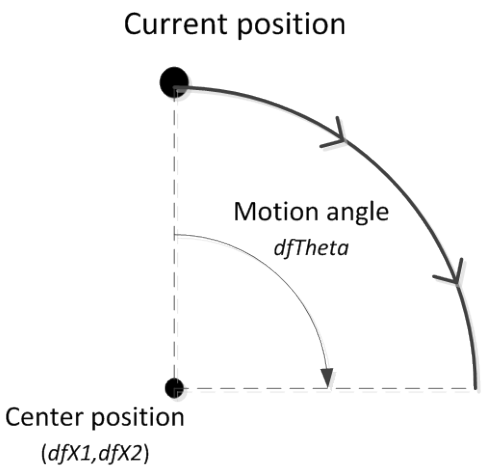
double *dfX1*,

double *dfX2*,

double *dfTheta*,

WORD *wGroupIndex*

)



Description	Moves in a curve around the indicated center with the indicated angle in the plane constructed by YZ axes. A negative angle indicates operating clockwise motion; a positive angle indicates operating counter-clockwise motion. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfX1*, *dfX2*	Center position in YZ cooidinate
	dfTheta	Motion angle
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

24. int MCC_ArcThetaZX(

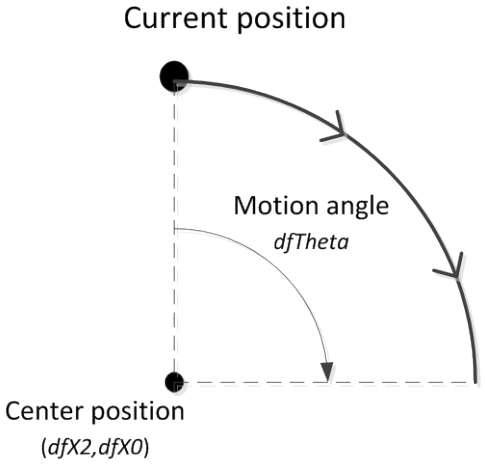
double *dfX2*,

double *dfX0*,

doubled *dfTheta*,

WORD *wGroupIndex*

)



Description	Moves in a curve around the indicated center with the indicated angle in the plane constructed by ZX axes. A negative angle indicates operating clockwise

motion; a positive angle indicates operating counter-clockwise motion. Calling this command successfully will increase the number of stored motion commands.

Parameters	*dfX2*, *dfX0*	Center position in ZX cooidinate
	dfTheta	Motion angle
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

25. int MCC_CircleXY(

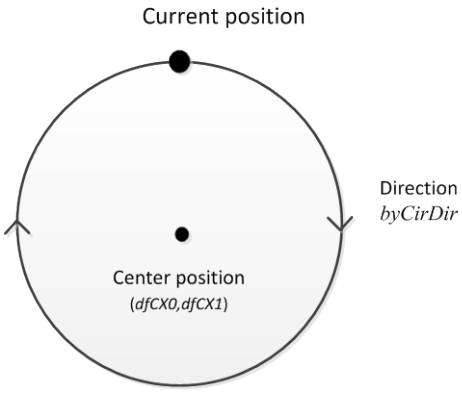
double *dfCX0*,

double *dfCX1*,

BYTE *byCirDir*,

WORD *wGroupIndex*

)



Description Moves in a complete circular trajectory around the indicated center in the plane constructed by XY axes. Calling this command successfully will increase the number of stored motion commands.

Parameters	*dfCX0*, *dfCX1*	Center position in XY cooidinate
	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

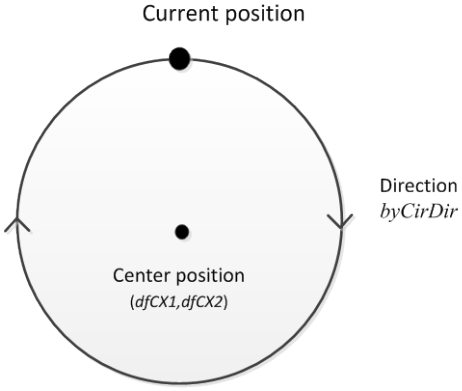
---

26. int MCC_CircleYZ(

double *dfCX1*,

double *dfCX2*,

BYTE *byCirDir*,



WORD ***wGroupIndex***

)

Description	Moves in a complete circular trajectory around the indicated center in the plane constructed by YZ axes. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfCX1*, *dfCX2*	Center position in YZ cooidinate
	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

27. int MCC_CircleZX(

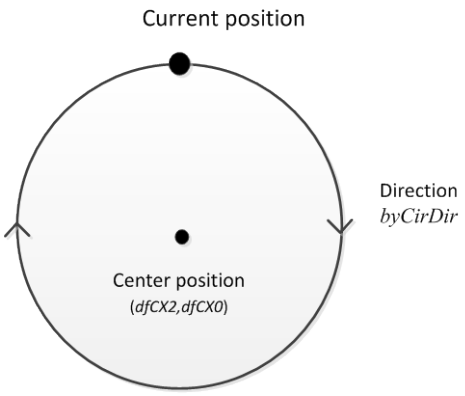
double ***dfCX2*****,**

double ***dfCX0*****,**

BYTE ***byCirDir*****,**

WORD ***wGroupIndex***

)



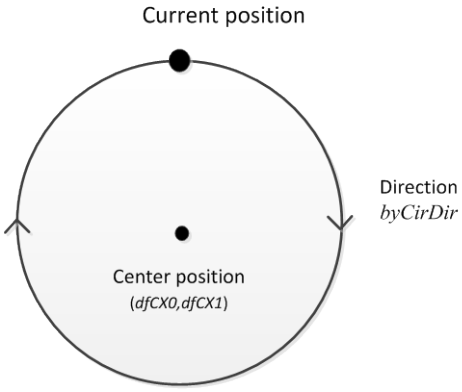
Description	Moves in a complete circular trajectory around the indicated center in the plane constructed by ZX axes. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfCX2*, *dfCX0*	Center position in ZX cooidinate
	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

28. int MCC_CircleXY_Aux(

double *dfCX0*,
double *dfCX1*,
double *dfU*,
double *dfV*,
double *dfW*,
double *dfA*,
double *dfB*,
BYTE *byCirDir*,
WORD *wGroupIndex*

)

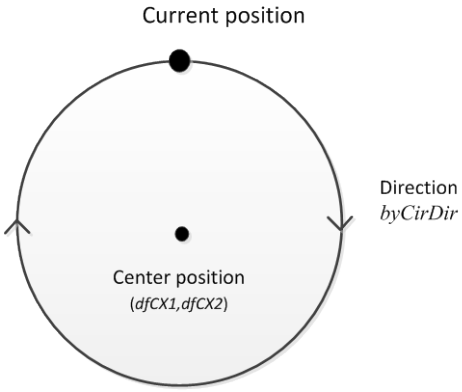


Description	Moves in a complete circular trajectory around the indicated center in the plane constructed by XY axes while simultaneously executing linear motion on U, V, W, A and B axes. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfCX0*, *dfCX1*	Center position in XY cooidinate
	dfU, *dfV*, *dfW*, *dfA*, *dfB*	UVWAB position of the end point
	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

29. int MCC_CircleYZ_Aux(

double *dfCX1*,
double *dfCX2*,
double *dfU*,
double *dfV*,
double *dfW*,
double *dfA*,
double *dfB*,
BYTE *byCirDir*,



WORD ***wGroupIndex***

)

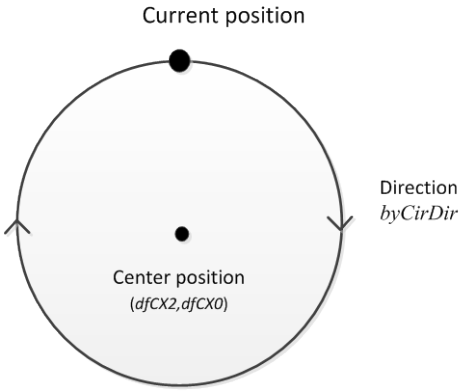
Description	Moves in a complete circular trajectory around the indicated center in the plane constructed by YZ axes while simultaneously executing linear motion on U, V, W, A and B axes. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfCX1*, *dfCX2*	Center position in YZ cooidinate
	dfU, *dfV*, *dfW*, *dfA*, *dfB*	UVWAB position of the end point
	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

30. int MCC_CircleZX_Aux(

double ***dfCX2*****,**
double ***dfCX0*****,**
double ***dfU*****,**
double ***dfV*****,**
double ***dfW*****,**
double ***dfA*****,**
double ***dfB*****,**
BYTE ***byCirDir*****,**
WORD ***wGroupIndex***

)



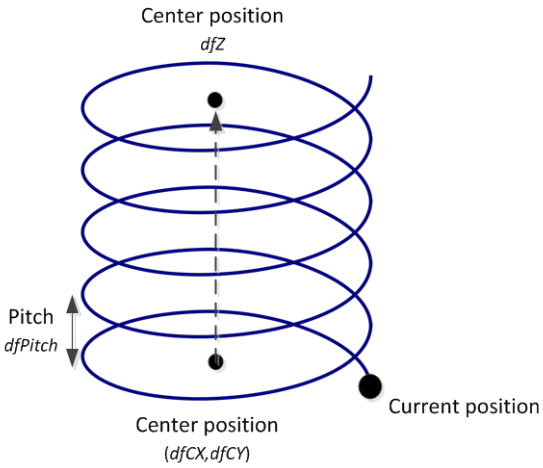
Description	Moves in a complete circular trajectory around the indicated center in the plane constructed by ZX axes while simultaneously executing linear motion on U, V, W, A and B axes. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfCX2*, *dfCX0*	Center position in ZX cooidinate
	dfU, *dfV*, *dfW*, *dfA*, *dfB*	UVWAB position of the end point

	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

31. int MCC_HelicaXY_Z(

double *dfCX*,
double *dfCY*,
double *dfZ*,
double *dfPitch*,
BYTE *byCirDir*,
WORD *wGroupIndex*
)

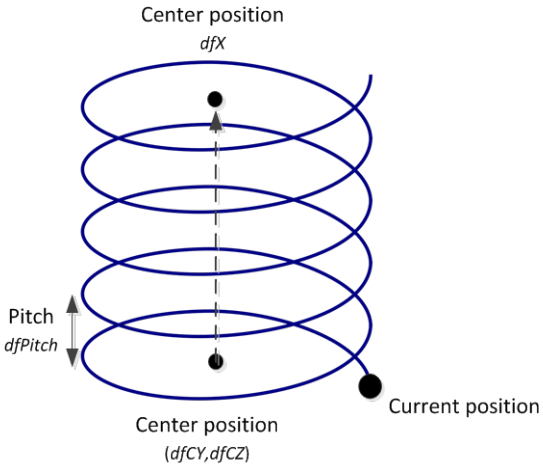


Description: Operates helix motion from current position. This is a circular motion along the XY plane; MCC_SetFeedSpeed() can be used to set the feed speed of this circular motion. When using this command, it is required to indicate the center position of circular motion on the XY plane; the radius is determined by the distance between current position and the center. It is also required to indicate the end point position on the Z axis. Calling this command successfully will increase the number of stored motion commands.

Parameters	*dfCX*, *dfCY*	Center position in XY cooidinate
	dfZ	Z position of the end point
	dfPitch	Distance moved on the Z axis after one complete circular motion on the XY plane; the value must be greater than 0.
	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

32. nt MCC_HelicaYZ_X(

double *dfCY*,
double *dfCZ*,
double *dfX*,
double *dfPitch*,
BYTE *byCirDir*,
WORD *wGroupIndex*
)

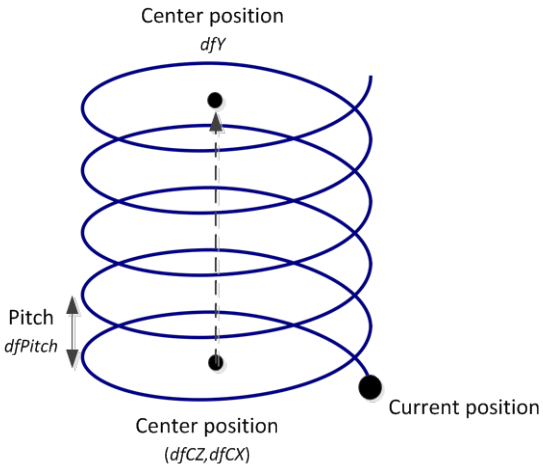


Description: Operates helix motion from current position. This is a circular motion along the YZ plane; MCC_SetFeedSpeed() can be used to set the feed speed of this circular motion. When using this command, it is required to indicate the center position of circular motion on the YZ plane; the radius is determined by the distance between current position and the center. It is also required to indicate the end point position on the X axis. Calling this command successfully will increase the number of stored motion commands.

Parameters		
	dfCY, *dfCZ*	Center position in YZ cooidinate
	dfX	X position of the end point
	dfPitch	Distance moved on the X axis after one complete circular motion on the YZ plane; the value must be greater than 0.
	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

33. int MCC_HelicaZX_Y(

double *dfCZ*,
double *dfCX*,
double *dfY*,

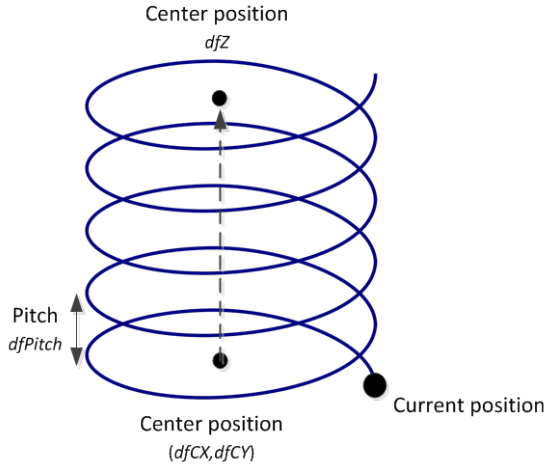


double ***dfPitch*****,**
BYTE ***byCirDir*****,**
WORD ***wGroupIndex***
)

Description	Operates helix motion from current position. This is a circular motion along the ZX plane; MCC_SetFeedSpeed() can be used to set the feed speed of this circular motion. When using this command, it is required to indicate the center position of circular motion on the ZX plane; the radius is determined by the distance between current position and the center. It is also required to indicate the end point position on the Y axis. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfCZ, dfCX*	Center position in ZX cooidinate
	dfY	Y position of the end point
	dfPitch	Distance moved on the Y axis after one complete circular motion on the ZX plane; the value must be greater than 0.
	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

34. int MCC_HelicaXY_Z_Aux(
double ***dfCX*****,**
double ***dfCY*****,**
double ***dfZ*****,**
double ***dfU*****,**
double ***dfV*****,**
double ***dfW*****,**
double ***dfA*****,**



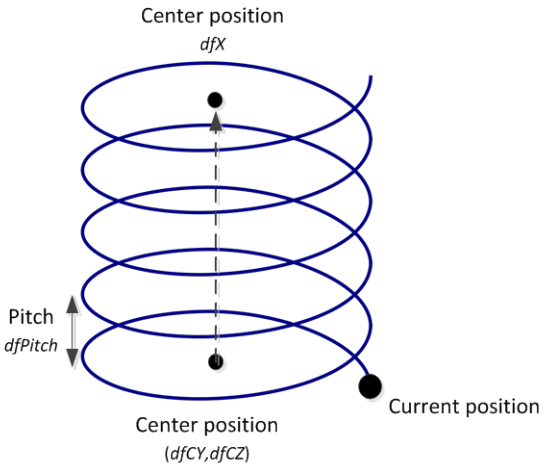
double ***dfB*****,**
double ***dfPitch*****,**
BYTE ***byCirDir*****,**
WORD ***wGroupIndex***
)

Description	Operates helix motion from current position. This is a circular motion along the XY plan while simultaneously executing linear motion on U, V, W, A and B axes. MCC_SetFeedSpeed() can be used to set the feed speed of this circular motion. When using this command, it is required to indicate the center position of circular motion on the XY plane; the radius is determined by the distance between current position and the center. It is also required to indicate the end point position on the Z axis. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfCX*, *dfCY*	Center position in XY cooidinate
	dfZ	Z position of the end point
	dfU, *dfV*, *dfW*, *dfA*, *dfB*	UVWAB position of the end point
	dfPitch	Distance moved on the Z axis after one complete circular motion on the XY plane; the value must be greater than 0.
	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

35. int MCC_HelicaYZ_X_Aux(
double ***dfCY*****,**
double ***dfCZ*****,**
double ***dfX*****,**
double ***dfU*****,**

double ***dfV*****,**
double ***dfW*****,**
double ***dfA*****,**
double ***dfB*****,**
double ***dfPitch*****,**
BYTE ***byCirDir*****,**
WORD ***wGroupIndex***
)



Description: Operates helix motion from current position. This is a circular motion along the YZ plane while simultaneously executing linear motion on U, V, W, A and B axes. MCC_SetFeedSpeed() can be used to set the feed speed of this circular motion. When using this command, it is required to indicate the center position of circular motion on the YZ plane; the radius is determined by the distance between current position and the center. It is also required to indicate the end point position on the X axis. Calling this command successfully will increase the number of stored motion commands.

Parameters

dfCY, dfCZ	Center position in YZ cooidinate
dfX	X position of the end point
dfU, dfV, dfW, dfA, dfB	UVWAB position of the end point
dfPitch	Distance moved on the X axis after one complete circular motion on the YZ plane; the value must be greater than 0.
byCirDir	Direction; 0=clockwise, 1=counter-clockwise
wGroupIndex	Group number

Return Value

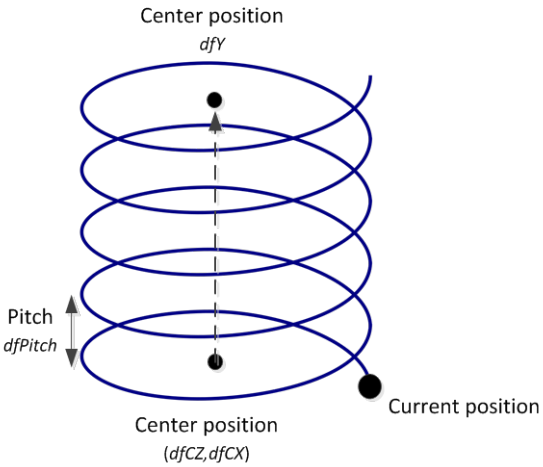
>0 or =0	The motion command index given by the MCCL
<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

36. int MCC_HelicaZX_Y_Aux(

double ***dfCZ*****,**
double ***dfCX*****,**
double ***dfY*****,**
double ***dfU*****,**

double ***dfV*****,**
double ***dfW*****,**
double ***dfA*****,**
double ***dfB*****,**
double ***dfPitch*****,**
BYTE ***byCirDir*****,**
WORD ***wGroupIndex***
)



Description	Operates helix motion from current position. This is a circular motion along the ZX plane while simultaneously executing linear motion on U, V, W, A and B axes. MCC_SetFeedSpeed() can be used to set the feed speed of this circular motion. When using this command, it is required to indicate the center position of circular motion on the ZX plane; the radius is determined by the distance between current position and the center. It is also required to indicate the end point position on the Y axis. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfCZ, dfCX*	Center position in ZX cooidinate
	dfY	Y position of the end point
	dfU, dfV, dfW, dfA, dfB	UVWAB position of the end point
	dfPitch	Distance moved on the Y axis after one complete circular motion on the ZX plane; the value must be greater than 0.
	byCirDir	Direction; 0=clockwise, 1=counter-clockwise
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

F. Point-to-Point Motion

---

1. double MCC_SetPtPSpeed(

double *dfRatio*,

WORD *wGroupIndex*

)

Description	Sets the point-to-point speed ratio. The speed (UU/s) of each axis during point-to-point motion equals ((Maximum rotation speed/60) × pitch number / gear deceleration ratio) × (speed ratio / 100) where the maximum rotation speed (*wRPM*), pitch number (*dfPitch*) and gear deceleration ratio (*dfGearRatio*) are all defined in mechanism parameters. The point-to-point speed ratio is equal to point-to-point speed divided by the maximum rotation speed of motor, and multiply the obtained value by 100. .	
Parameters	*dfRatio*	point-to-point speed ratio; the value must be larger than 0 and smaller than or equal 100
	wGroupIndex	Group number
Return Value	Speed ratio actually set	

---

2. double MCC_GetPtPSpeed(

WORD *wGroupIndex*

)

Description	Acquires the point-to-point speed ratio ratio used during point-to-point motion	
Parameters	*wGroupIndex*	Group number
Return Value	>0 or =0	Point-to-point speed ratio ratio used during point-to-point motion
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_PtP(

double ***dfX*****,**
double ***dfY*****,**
double ***dfZ*****,**
double ***dfU*****,**
double ***dfV*****,**
double ***dfW*****,**
double ***dfA*****,**
double ***dfB*****,**
WORD ***wGroupIndex*****,**
DWORD ***dwAxisMask***

)

Description: Uses point-to-point motion to move from the current position to the indicated end point by the set point-to-point speed ratio. Calling this command successfully will increase the number of stored motion commands.

Parameters:

dfX, dfY, dfZ	XYZ position of the end point
dfU, dfV, dfW	UVW position of the end point
dfA, dfB	AB position of the end point
wGroupIndex	Group number
dwAxisMask	Indicates the axis that performs the desired action; the assigned parameters can be:

IMP_AXIS_X	X axis
IMP_AXIS_Y	Y axis
IMP_AXIS_Z	Z axis
IMP_AXIS_U	U axis
IMP_AXIS_V	V axis
IMP_AXIS_W	W axis
IMP_AXIS_A	A axis
IMP_AXIS_B	B axis
IMP_AXIS_ALL	All motion axes

The above parameters can be combined freely. Take the operation on X, Z and V axes for an example:

```
(IMP_AXIS_X | IMP_AXIS_Z | IMP_AXIS_V)
```

Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_PtPX(

double ***dfX*****,**

WORD ***wGroupIndex***

)

Description	Uses point-to-point motion to move from the current position to the indicated end point by the set point-to-point speed ratio. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfX*	X position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_PtPY(

double ***dfY*****,**

WORD ***wGroupIndex***

)

Description	Uses point-to-point motion to move from the current position to the indicated end point by the set point-to-point speed ratio. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfY*	Y position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_PtPZ(

double ***dfZ*****,**

WORD ***wGroupIndex***

)

Description	Uses point-to-point motion to move from the current position to the indicated end point by the set point-to-point speed ratio. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfZ*	Z position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_PtPU(

double *dfU*,

WORD *wGroupIndex*

)

Description	Uses point-to-point motion to move from the current position to the indicated end point by the set point-to-point speed ratio. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfU*	U position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_PtPV(

double *dfV*,

WORD *wGroupIndex*

)

Description	Uses point-to-point motion to move from the current position to the indicated end point by the set point-to-point speed ratio. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfV*	V position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL

	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

9. int MCC_PtPW(

double *dfW*,

WORD *wGroupIndex*

)

Description	Uses point-to-point motion to move from the current position to the indicated end point by the set point-to-point speed ratio. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfW*	W position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

10. int MCC_PtPA(

double *dfA*,

WORD *wGroupIndex*

)

Description	Uses point-to-point motion to move from the current position to the indicated end point by the set point-to-point speed ratio. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfA*	A position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

11. int MCC_PtPB(

double *dfB*,

WORD *wGroupIndex*

)

Description	Uses point-to-point motion to move from the current position to the indicated end point by the set point-to-point speed ratio. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dfB*	B position of the end point
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

12. int MCC_SetPtPAccType(char *cType0*, char *cType1*, char *cType2*,
char *cType3*, char *cType4*, char *cType5*,
char *cType6*, char *cType7*, WORD *wGroupIndex*
)

Description	Sets the acceleration type used for point-to-point motion; each axis uses an independent acceleration type.	
Parameters	cType0 ~ cType7	Acceleration type for each axis; it can be set as : 'T' to use trapezoidal acceleration curve 'S' to use S acceleration curve
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

13. int MCC_GetPtPAccType(char **pcType0*, char **pcType1*, char **pcType2*,
char **pcType3*, char **pcType4*, char **pcType5*,
char **pcType6*, char **pcType7*, WORD *wGroupIndex*)

Description	Acquires the acceleration type used during point-to-point motion	
Parameters	*pcType0 ~ *pcType7	Acceleration type for each axis 0 indicates using trapezoidal acceleration curve 1 indicates using S acceleration curve
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command**

Return Values for the meaning of return value

---

14. int MCC_SetPtPDecType(char *cType0*, char *cType1*, char *cType2*,
char *cType3*, char *cType4*, char *cType5*,
char *cType6*, char *cType7*, WORD *wGroupIndex*)

Description	Sets the deceleration type used for point-to-point motion; each axis uses an independent deceleration type.	
Parameters	cType0 ~ cType7	Deceleration type for each axis; it can be set as : 'T' to use trapezoidal deceleration curve 'S' to use S deceleration curve
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

15. int MCC_GetPtPDecType(char **pcType0*, char **pcType1*, char **pcType2*,
char **pcType3*, char **pcType4*, char **pcType5*,
char **pcType6*, char **pcType7*, WORD *wGroupIndex*)

Description	Acquires deceleration type used in point-to-point motion	
Parameters	*pcType0 ~ *pcType7	Deceleration type of each axis; 0 indicates to use trapezoidal deceleration curve 1 indicates to use S deceleration curve
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

16. int MCC_SetPtPAccTime(double *dfTime0*, double *dfTime1*, double *dfTime2*,
double *dfTime3*, double *dfTime4*, double *dfTime5*,
double *dfTime6*, double *dfTime7*, WORD *wGroupIndex*)

Description	Sets the time required for point-to-point motion to accelerate to a stable speed; each axis uses an independent acceleration time.

Parameters	dfTime0 ~ dfTime7	Acceleration time of each axis; this value must be greater than 0. Unit: ms.
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

17. int MCC_GetPtPAccTime(double **pdfTime0*, double **pdfTime1*, double **pdfTime2*, double **pdfTime3*, double **pdfTime4*, double **pdfTime5*, double **pdfTime6*, double **pdfTime7*, WORD *wGroupIndex*)

Description	Acquires the time required for point-to-point motion to accelerate to a stable speed; each axis uses an independent acceleration time.	
Parameters	pdfTime0 ~ pdfTime7	Acceleration time of each axis; unit: ms.
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

18. int MCC_SetPtPDecTime(double *dfTime0*, double *dfTime1*, double *dfTime2*, double *dfTime3*, double *dfTime4*, double *dfTime5*, double *dfTime6*,double *dfTime7*,WORD *wGroupIndex*)

Description	Sets the time required for point-to-point motion to decelerate from a stable speed to stop; each axis uses an independent deceleration time.	
Parameters	dfTime0 ~ dfTime7	Deceleration time of each axis; this value must be greater than 0. Unit: ms.
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command**

		Return Values for the meaning of return value

---

19. int MCC_GetPtPDecTime(double **pdfTime0*, double **pdfTime1*,
double **pdfTime2*, double **pdfTime3*,
double **pdfTime4*, double **pdfTime5*,
double **pdfTime6*, double **pdfTime7*,
WORD *wGroupIndex*)

Description	Acquires the time required for point-to-point motion to decelerate from a stable speed to stop; each axis uses an independent deceleration time.	
Parameters	pdfTime0 ~ pdfTime7	Deceleration time of each axis; unit: ms.
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

G. Any curve motion

---

1. int MCC_CustomMotion(

CUSTOM_START_MOTION *pfnStartMotion*,

CUSTOM_INTERPOLATION *pfnInterpolation*,

CUSTOM_CLEANUP *pfnCleanUp*,

void **pvBuffer*, WORD *wGroupIndex*, DWORD *dwAxisMask*

)

Description: The approach to use the random curve motion is to run the trajectory motion from current position through customized interpolation points with the set feed speed. Before using this command, the user must write three CallBack commands that to be called by the MCCL during motion process. Calling this command successfully will increase the number of stored motion commands.

Parameters:

pfnStartMotion: This command will be called when the MCCL initiates one any curve motion command. This command must send back the total offset travel of the given random curve command to be executed.

pfnInterpolation: During this process, the command will be called continuously when the MCCL runs one random curve interpolation motion. This command must send back the next interpolation offset of motion process.

pfnCleanUp: This command will be called when the MCCL completes one random curve motion. This command is responsible for the system source release used by the random curve, such as memory release.

**pvBuffer*: A pointer to pvBuffer structure is used to indicates the initiation position of memory. The user can store the information required by the random curve into a memory block and send the initiation position of this memory to the MCCL. When the MCCL calls these callback

		commands, the memory position will also be sent to callback commands.
	wGroupIndex	Group number
	dwAxisMask	Indicates the axis that performs the desired action; the assigned parameters can be: IMP_AXIS_X X axis IMP_AXIS_Y Y axis IMP_AXIS_Z Z axis IMP_AXIS_U U axis IMP_AXIS_V V axis IMP_AXIS_W W axis IMP_AXIS_A A axis IMP_AXIS_B B axis IMP_AXIS_ALL All motion axes The above parameters can be combined freely. Take the operation on X, Z and V axes for an example: (IMP_AXIS_X | IMP_AXIS_Z | IMP_AXIS_V)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_CustomMotionEx(

CUSTOM_START_MOTION *pfnStartMotion*,

CUSTOM_INTERPOLATION *pfnInterpolation*,

CUSTOM_CLEANUP *pfnCleanUp*,

CUSTOM_BLENDING_START *pfnBlendingStart*,

CUSTOM_BLENDING *pfnBlending*,

CUSTOM_BLENDING_END *pfnBlendingEnd*,

void **pvBuffer*, WORD *wGroupIndex*, DWORD *dwAxisMask*

)

Description The approach to use the random curve motion is to run the trajectory motion from current position through customized interpolation points with the set

feed speed. Before using this command, the user must write six CallBack commands that to be called by the MCCL during motion process. Calling this command successfully will increase the number of stored motion commands.

Parameters

pfnStartMotion	This command will be called when the MCCL initiates one any curve motion command. This command must send back the total travel of the given random curve command to be executed.
pfnInterpolation	During this process, the command will be called continuously when the MCCL runs one random curve interpolation motion. This command must send back the next interpolation offset of motion process.
pfnCleanUp	This command will be called when the MCCL completes one random curve motion. This command is responsible for the system source release used by the random curve, such as memory release.
pfnBlendingStart	This command will be called first when the MCCL initiates one random curve continuous path planning. The users can preprogramme the information required for the continuous path in this command.
pfnBlending	This command will be called continuously when the MCCL runs one random curve continuous path. This command must send back the next interpolation offset of motion process.
pfnBlendingEnd	This command will be called when the MCCL ends one random curve continuous path motion. The user can release back the system source used in this command.
**pvBuffer*	A pointer to pvBuffer structure is used to indicates the initiation position of memory. The user can store the information required by the random curve into a memory block and send the initiation position of this memory to the MCCL. When the MCCL calls these callback commands, the memory position will also be sent to callback commands.

	wGroupIndex	Group number
	dwAxisMask	Indicates the axis that performs the desired action; the assigned parameters can be: IMP_AXIS_X X axis IMP_AXIS_Y Y axis IMP_AXIS_Z Z axis IMP_AXIS_U U axis IMP_AXIS_V V axis IMP_AXIS_W W axis IMP_AXIS_A A axis IMP_AXIS_B B axis IMP_AXIS_ALL All motion axes The above parameters can be combined freely. Take the operation on X, Z and V axes for an example: (IMP_AXIS_X | IMP_AXIS_Z | IMP_AXIS_V)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

H. JOG Motion

---

1. int MCC_JogPulse(

int *nPulse*,

char *cAxis*,

WORD *wGroupIndex*

)

Description	Jog motion (pulse). After all other motion commands have been executed (the return value obtained by calling MCC_GetMotionStatus() should be GMS_STOP at this point), the specified axis is driven according to the indicated displacement (pulses) and direction. This command is manually fine-tuning process which requires the motion status to be at "stop" to be effective. Jog motion does not consist of acceleration or deceleration; therefore, the set displacement should not be overly large to avoid excessive machine vibration.	
Parameters	*nPulse*	Displacement; unit: pulse. Within the given range of -2048 to 2048.
	cAxis	The motion axis number required for jog motion (0 ~ 7 represents X ~ B axes)
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_JogSpace(

double *dfOffset*,

int *nRatio*,

char *cAxis*,

WORD *wGroupIndex*

)

Description	Jog motion (UU: user unit). After all other motion commands have been executed (the return value obtained by calling MCC_GetMotionStatus()	

should be GMS_STOP at this point), the specified axis is driven according to the indicated displacement (increment) and speed ratio (same meaning as point-to-point speed ratio). Calling this command successfully will increase the number of stored motion commands.

Parameters	*dfOffset*	Displacement; unit: UU
	nRatio	Speed ratio; the value must be larger than 0 and smaller than or equal 100
	cAxis	The motion axis number required for step motion (0 ~ 7 represents X ~ B axes)
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_JogConti(

int *nDir*,

int *nRatio*,

char *cAxis*,

WORD *wGroupIndex*

)

Description: Continuous jog motion (UU: user unit). After all other motion commands have been executed (the return value obtained by calling MCC_GetMotionStatus() should be GMS_STOP at this point), the specified axis is driven according to the indicated direction and speed ratio (same meaning as point-to-point speed ratio) and stops until moving to the edge of effective work zone (the limits of effective work zone is defined by machine parameters). Calling this command successfully will increase the number of stored motion commands.

Parameters	*nDir*	Continuous jog motion direction; it can be set as: 1 Forward motion -1 Reverse motion
	nRatio	Speed ratio; the value must be larger than 0 and smaller than or equal 100

	cAxis	The motion axis number required for jog motion (0 ~ 7 represents X ~ B axes)
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

I. Motion Status Check

---

1. int MCC_GetMotionStatus(

WORD ***wGroupIndex***

)

Description	Checks the current motion status of system	
Parameters	*wGroupIndex*	Group number
Return Value	0	Under running status (GMS_RUNNING); there are still motion commands waiting to be completed
	1	Under stop status (GMS_STOP); no stored motion commands
	2	Under hold status (GMS_HOLD); if the user called MCC_HoldMotion
	3	Under delay status(GMS_DELAYING); if the user called MCC_DelayMotion
	Other	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_GetCurCommand(

COMMAND_INFO* ***pstCurCmdInfo*****,**

WORD ***wGroupIndex***

)

Description	Acquires the information related to motion commands being executed, including motion command type, motion command index, required feed speed and end point.	
Parameters	*pstCurCmdInfo*	A pointer to COMMAND_INFO structure is used to store the content of motion commands being executed is defined as:

```
typedef struct _COMMAND_INFO
{
    int     nType;
    int     nCommandIndex;
    double  dfFeedSpeed;
```

```
    double    dfPos[MAX_AXIS_NUM];
} COMMAND_INFO;
```

nType：Motion command type

0	Point-to-point motion
1	Linear motion
2	Clockwise curve and circular motion
3	Counter-clockwise curve and circular motion
4	Clockwise helix motion
5	Counter-clockwise helix motion
6	Motion delay command
7	Enabling path blending
8	Disabling path blending
9	Enabling In Position Confirmation
10	Disabling In Position Confirmation

nCommandIndex: Motion command index

dfFeedSpeed：

General motion	Programmed feed speed
Point-to-point motion	Programmed point-to-point speed ratio
Motion delay	Delay time currently remaining (unit: ms)

dfPos[]: Absolute position of end point

	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. **int MCC_GetCommandCount(**

int* ***pnCmdCount*****,**

WORD ***wGroupIndex***

)

Description Acquires the amount of unexecuted motion commands stored in the motion command queue. As for which commands will increase the commands stored, please refer to "**Command Library Operational Properties**" section in

	"**IMP Series Motion Control Command Library User Manual**".	
Parameters	*pnCmdCount*	A pointer to int variable is used to store the number of motion commands
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. **int MCC_ResetCommandIndex(**
   WORD ***wGroupIndex***
)

Description	Reset the motion command index to zero. The motion command index is a identifying data given to each motion command in the MCCL. The motion command index can be counted from 0 by using this command.	
Parameters	*wGroupIndex*	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_GetCurPulseStockCount(
WORD* ***pwStockCount*****,**
WORD ***wChannel*****,**
WORD ***wCardIndex***
)

Description	Acquires the current quantity of fine movement command (FMC) used in hardware FIFO for each exis. To achieve stable motion control, the FMC stored in the command library should not be smaller than 60 during motion. Please increase the interpolation time when this condition cannot be met (recall MCC_InitSystem).	
Parameters	*pwStockCount*	A pointer to WORD variable is used to store the quantity of FMC
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)

Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_SetMaxPulseStockNum(

WORD ***nMaxStockNum*****,**

)

Description	Sets the maximum quantity of FMC used in hardware FIFO. The quantity of FMC uses should be evaluated with the real-time performance of operating system. The less quantity it sets, the higher real-time performance the operating system should possess; the more quantity it sets, a larger tolerance to real-time performance and a higher stability to motion control functions. Please set more usage quantities or increase the interpolation time if conditions of stable motion control cannot be met.	
Parameters	*wMaxStockNum*	Sets the maximum quantity of FMC used in hardware FIFO
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_GetMaxPulseStockNum(

WORD ****wMaxStockNum***

)

Description	Acquires the maximum quantity of FMC used in hardware FIFO	
Parameters	*wMaxStockNum*	A pointer to WORD variable is used to store maximum quantity of FMC in hardware FIFO
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_GetErrorCode(

WORD ***wGroupIndex***

)

Description	Acquires current error record to check if an error occurred during system operation. This command should be called periodically (for example, every 100ms) during system operation to confirm the system is currently operating normally. If an error record is produced, the user should conduct the corresponding error recovery process.	
Parameters	*wGroupIndex*	Group number
Return Value	0	No error
	Other	Error code (please refer to **Section IV. Error Codes**)

---

9. int MCC_ClearError(

WORD *wGroupIndex*

)

Description	After an error has occurred during system operation, if the error has been removed, it is required to use this command to delete the error record in the system to ensure normal system operation.	
Parameters	*wGroupIndex*	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

J. Go Home

---

1. int MCC_Home(

int ***nOrder0*****,**

int ***nOrder1*****,**

int ***nOrder2*****,**

int ***nOrder3*****,**

int ***nOrder4*****,**

int ***nOrder5*****,**

int ***nOrder6*****,**

int ***nOrder7*****,**

WORD ***wCardIndex***

)

Description	Performs the Go Home motion. Please refer to MCC_SetHomeConfig() for related settings. This command can be used with calling MCC_GetGoHomeStatus() to check if the action has been completed. Once the Go Home motion has been completed, the Cartesian coordinates for each axis will be set as 0.	
Parameters	*nOrder0 ~ nOrder7*	The Go Home order for each axis. The order can be set between 0 and 7; the smaller the number is, the earlier the Go Home motion will be executed. For motion axes that do not run the Go Home motion, the order should be set as 255. Note: These parameters correspond to outlet axes 0~7 on *wCardIndex* control card instead of motion axes in the group. For a detailed description, please refer to the "**Go Home" section** in "**IMP Series Motion Control Command Library User Manual"**
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section V. Commands** for

		the meaning of return value

---

2. int MCC_GetGoHomeStatus()

Description	After calling MCC_Home(), this command can be used to check if the Go Home motion has been completed.	
Return Value	0	Go Home motion has not completed
	1	Go Home motion has completed
	Other	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_AbortGoHome()

Description	After calling MCC_Home(), this command can be used to stop the Go Home motion.	
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_GetHomeSensorStatuss(

WORD* ***pwStatus*****,**

WORD ***wChannel*****,**

WORD ***wCardIndex***

)

Description	Acquires the home sensor status. Before using this command, it is necessary to accurately define the home sensor wiring (normal open or normal close). The wiring is defined in the Go Home parameters.	
Parameters	*pwStatus*	A pointer to WORD variable is used to store the home sensor status. 1 indicates the home sensor is currently triggered and 0 indicates not triggered.
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

K. In Position Control

1. int MCC_SetCompParam(

SYS_COMP_PARAM* ***pstCompParam*****,**

WORD ***wChannel*****,**

WORD ***wCardIndex***

)

Description	Sets parameters for gear backlash and gap compensation. User sets compensation parameters by using this command and call MCC_UpdatetCompParam()to ensure new parameters to be updated. Compensation parameters must be assigned for the total travel in each axis. For a detailed description, please refer to the "**Gear Backlash and Gap Compensation"** Section in **"IMP Series Motion Control Command Library User Manual**".	
Parameters	*pstCompParam*	A pointer to SYS_COMP_PARAM structure is used to describe compensation parameters
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

2. int MCC_UpdateCompParam()

Description	Responds to updated parameters for gear backlash and gap compensation. After calling MCC_SetCompParam (), this command must be executed for the system to respond to new settings.	
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

3. int MCC_SetPGain(

WORD ***wGain0*****,**
WORD ***wGain1*****,**
WORD ***wGain2*****,**
WORD ***wGain3*****,**
WORD ***wGain4*****,**
WORD ***wGain5*****,**
WORD ***wGain6*****,**
WORD ***wGain7*****,**
WORD ***wCardIndex***

)

Description	Sets the proportional gain (P Gain) used in position closed loop control	
Parameters	*wGain0~wGain7*	Proportional gain used in each axis; the setting range is between 1 ~ 127
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_GetPGain(

WORD* ***pwGain0*****,**
WORD* ***pwGain1*****,**
WORD* ***pwGain2*****,**
WORD* ***pwGain3*****,**
WORD* ***pwGain4*****,**
WORD* ***pwGain5*****,**
WORD* ***pwGain6*****,**
WORD* ***pwGain7*****,**
WORD ***wCardIndex***

)

Description	Acquires the proportional gain (P Gain) used in position closed loop control	
Parameters	*pwGain0~pwGain7*	A pointer to WORD variable is used to store the proportional gain used in each axis

	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_SetIGain(

WORD ***wGain0*****,**
WORD ***wGain1*****,**
WORD ***wGain2*****,**
WORD ***wGain3*****,**
WORD ***wGain4*****,**
WORD ***wGain5*****,**
WORD ***wGain6*****,**
WORD ***wGain7*****,**
WORD ***wCardIndex***

)

Description	Sets the integral gain (I Gain) used in position closed loop control	
Parameters	*wGain0~wGain7*	Integral gain used in each axis; the setting range is between 1 ~ 127
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_GetIGain(

WORD* ***pwGain0*****,**
WORD* ***pwGain1*****,**
WORD* ***pwGain2*****,**
WORD* ***pwGain3*****,**
WORD* ***pwGain4*****,**
WORD* ***pwGain5*****,**
WORD* ***pwGain6*****,**
WORD* ***pwGain7*****,**

WORD ***wCardIndex***

)

Description	Acquires the integral gain (I Gain) used in position closed loop control	
Parameters	*pwGain0~pwGain7*	A pointer to WORD variable is used to store the integral gain used in each axis
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_SetDGain(

WORD ***wGain0*****,**
WORD ***wGain1*****,**
WORD ***wGain2*****,**
WORD ***wGain3*****,**
WORD ***wGain4*****,**
WORD ***wGain5*****,**
WORD ***wGain6*****,**
WORD ***wGain7*****,**
WORD ***wCardIndex***

)

Description	Sets the differential gain (D Gain) used in position closed loop control	
Parameters	*wGain0~wGain7*	Differential gain used in each axis; the setting range is between 1 ~ 127
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_GetDGain(

WORD* ***pwGain0*****,**
WORD* ***pwGain1*****,**
WORD* ***pwGain2*****,**

WORD* *pwGain3*,
WORD* *pwGain4*,
WORD* *pwGain5*,
WORD* *pwGain6*,
WORD* *pwGain7*,
WORD *wCardIndex*
)

Description	Acquires the differential gain (D Gain) used in position closed loop control	
Parameters	*pwGain0~pwGain7*	A pointer to WORD variable is used to store the differential gain used in each axis
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

9. int MCC_SetFGain(
WORD *wGain0*,
WORD *wGain1*,
WORD *wGain2*,
WORD *wGain3*,
WORD *wGain4*,
WORD *wGain5*,
WORD *wGain6*,
WORD *wGain7*,
WORD *wCardIndex*
)

Description	Sets the feed forward gain used in position closed loop control	
Parameters	*wGain0~wGain7*	Feed forward gain used in each axis; the setting range is between 1 ~ 127
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

10. int MCC_GetFGain(

WORD* ***pwGain0*****,**

WORD* ***pwGain1*****,**

WORD* ***pwGain2*****,**

WORD* ***pwGain3*****,**

WORD* ***pwGain4*****,**

WORD* ***pwGain5*****,**

WORD* ***pwGain6*****,**

WORD* ***pwGain7*****,**

WORD ***wCardIndex***

)

Description	Acquires the feed forward gain used in position closed loop control	
Parameters	*pwGain0~pwGain7*	A pointer to WORD variable is used to store the feed forward gain used in each axis
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

11. int MCC_SetMaxPulseSpeed(

int ***nPulse0*****,**

int ***nPulse1*****,**

int ***nPulse2*****,**

int ***nPulse3*****,**

int ***nPulse4*****,**

int ***nPulse5*****,**

int ***nPulse6*****,**

int ***nPulse7*****,**

WORD ***wCardIndex***

)

Description Sets the maximum pulse for each axis. Maximum pulse prevents the machine speed from exceeding operating parameters by limiting pulses each axis can

	send within one unit of interpolation time. For a detailed description, please refer to the "**Interpolation Time and Acceleration/Deceleration Time**" in **"IMP Series Motion Control Command Library User Manual"**.	
Parameters	*nPulse0~nPulse7*	Maximum pulse speed for each axis. The setting range is between 1~32767; the appropriate value should be determined by machine properties and interpolation time.
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

12. int MCC_GetMaxPulseSpeed(

int* *pnPulse0*,
int* *pnPulse1*,
int* *pnPulse2*,
int* *pnPulse3*,
int* *pnPulse4*,
int* *pnPulse5*,
int* *pnPulse6*,
int* *pnPulse7*,
WORD *wCardIndex*

)

Description	Acquires the maximum pulse speed for each axis	
Parameters	*pnPulse0~pnPulse7*	A pointer to int variable is used to store the maximum pulse for each axis
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

13. int MCC_SetMaxPulseAcc(

int *nPulse0*,

int ***nPulse1*****,**
int ***nPulse2*****,**
int ***nPulse3*****,**
int ***nPulse4*****,**
int ***nPulse5*****,**
int ***nPulse6*****,**
int ***nPulse7*****,**
WORD ***wCardIndex***
)

Description: Sets the maximum pulse acceleration for each axis. Maximum pulse acceleration prevents the machine acceleration (deceleration) from exceeding work range by limiting the change of pulses in two continuous interpolation time. For a detailed description, please refer to the "**Interpolation Time and Acceleration/Deceleration Time**" in **"IMP Series Motion Control Command Library User Manual"**.

Parameters		
	nPulse0~nPulse7	The maximum pulse acceleration for each axis. The setting range is between 1~32767; the appropriate value should be determined by machine properties and interpolation time.
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

14. int MCC_GetMaxPulseAcc(
int* ***pnPulse0*****,**
int* ***pnPulse1*****,**
int* ***pnPulse2*****,**
int* ***pnPulse3*****,**
int* ***pnPulse4*****,**
int* ***pnPulse5*****,**
int* ***pnPulse6*****,**
int* ***pnPulse7*****,**

WORD ***wCardIndex***

)

Description	Acquires the maximum pulse acceleration for each axis	
Parameters	*pnPulse0~pnPulse7*	A pointer to int variable is used to store the maximum acceleration of pulse for each axis
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

15. int MCC_SetInPosMode(

WORD ***wMode*****,**

WORD ***wGroupIndex***

)

Description	Sets the in position confirmation mode. For a detailed description, please refer to the "**In Position confirmation" section** in **"IMP Series Motion Control Command Library User Manual"**	
Parameters	*wMode*	In position confirmation mode
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

16. int MCC_SetInPosMaxCheckTime(

WORD ***wMaxCheckTimet*****,**

WORD ***wGroupIndex***

)

Description	Sets the in position confirmation maximum check time. For a detailed description, please refer to the "**In Position confirmation" section** in **"IMP Series Motion Control Command Library User Manual"**	
Parameters	*wMaxCheckTime*	In position confirmation maximum check time; unit: ms
	wGroupIndex	Group number
Return Value	0	Successful

	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

17. int MCC_ SetInPosSettleTime(

WORD *wSettleTime*,

WORD *wGroupIndex*

)

Description	Sets the in position confirmation settle time. For a detailed description, please refer to the "**In Position confirmation" section** in **"IMP Series Motion Control Command Library User Manual"**	
Parameters	*wSettleTime*	In position confirmation settle time; unit: ms.
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

18. int MCC_EnableInPos(

WORD *wGroupIndex*

)

Description	Enables the in position confirmation function. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*wGroupIndex*	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

19. int MCC_DisableInPos(

WORD *wGroupIndex*

)

Description	Disables the in position confirmation function. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*wGroupIndex*	*g*roup number
Return Value	>0 or =0	The motion command index given by the MCCL

	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

20. int MCC_SetInPosToleranceEx(

double ***dfTol0*****,**
double ***dfTol1*****,**
double ***dfTol2*****,**
double ***dfTol3*****,**
double ***dfTol4*****,**
double ***dfTol5*****,**
double ***dfTol6*****,**
double ***dfTol7*****,**
WORD ***wGroupIndex***

)

Description	Sets tolerance of “in position” function for each axis.	
Parameters	*dfTol0 ~ dfTol7*	In position error tolerance of each axis; unit: UU
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

21. int MCC_GetInPosToleranceEx(

double* ***pdfTol0*****,**
double* ***pdfTol1*****,**
double* ***pdfTol2*****,**
double* ***pdfTol3*****,**
double* ***pdfTol4*****,**
double* ***pdfTol5*****,**
double* ***pdfTol6*****,**
double* ***pdfTol7*****,**
WORD ***wGroupIndex***

)

Description	Acquires tolerance of “in position” function for each axis

Parameters	*pdfTol0~pdfTol7*	A pointer to double variable is used to store the in position error tolerance for each axis; unit: UU
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

22. int MCC_GetInPosStatus(

BYTE* *pbyInPos0*,
BYTE* *pbyInPos1*,
BYTE* *pbyInPos2*,
BYTE* *pbyInPos3*,
BYTE* *pbyInPos4*,
BYTE* *pbyInPos5*,
BYTE* *pbyInPos6*,
BYTE* *pbyInPos7*,
WORD *wGroupIndex*

)

Description	Acquires the in position confirmation status for each axis.	
Parameters	*pbyInPos0~pbyInPos7*	A pointer to BYTE variable is used to store the in position confirmation status for each axis. 0xff (255) indicates in position confirmation requirements satisfied; 0 indicates in position confirmation requirements not satisfied
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

23. int MCC_EnableTrackError(

WORD *wGroupIndex*,
DWORD *dwAxisMask*

)

Description	Enables tracking error check function . For a detailed description, please refer to the "**Error Tracking" section** in "**IMP Series Motion Control Command Library User Manual"**	
Parameters	*wGroupIndex*	Group number
	dwAxisMask	Indicates the axis that performs the desired action; the assigned parameters can be:
		IMP_AXIS_X X axis
		IMP_AXIS_Y Y axis
		IMP_AXIS_Z Z axis
		IMP_AXIS_U U axis
		IMP_AXIS_V V axis
		IMP_AXIS_W W axis
		IMP_AXIS_A A axis
		IMP_AXIS_B B axis
		IMP_AXIS_ALL All motion axes
		The above parameters can be combined freely. Take the operation on X, Z and V axes for an example: (IMP_AXIS_X | IMP_AXIS_Z | IMP_AXIS_V)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

24. int MCC_DisableTrackError(

WORD wGroupIndex

)

Description	Disables tracking error check function.	
Parameters	*wGroupIndex*	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

25. int MCC_SetTrackErrorLimit(

double ***dfLimit*****,**
char ***cAxis*****,**
WORD ***wGroupIndex***
)

Description	Sets tolerance of tracking error.	
Parameters	*dfLimit*	Tolerance of tracking error; unit: UU
	cAxis	Motion axis number (0~7 represents to X~B axis)
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

26. int MCC_GetTrackErrorLimit(
double* ***pdfLimit*****,**
char ***cAxis*****,**
WORD ***wGroupIndex***
)

Description	Acquires tolerance of tracking error.	
Parameters	*pdfLimit*	A pointer to double variable is used to store tolerance of tracking error; unit: UU
	cAxis	Motion axis number (0~7 represents to X~B axis)
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

27. int MCC_SetPCLRoutine(
PCLISR ***pfnPCLRoutine*****,**
WORD ***wCardIndex***
)

Description Sets customized ISR of PCL. The system will automatically call this ISR when the position closed loop fails. For a detailed description, please refer to the "**Position Closed Loop Failure Treatment" section** in **"Handling**

Positional Closed Loop Control Failure"

Parameters	*pfnPCLRoutine*	Command index for the customized position closed loop ISR
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

28. int MCC_SetErrorCountThreshold (

WORD *wChannel*,

WORD *wPlusThreshold*,

WORD *wMinusThreshold*,

WORD *wCardIndex*

)

Description Sets tolerance of pulse error count.

Parameters	*wChannel*	Motion axis number (0~7 represents to X~B axis)
	wPlusThreshold	Sets the error tolerance of pulse count in a positive value; unit: pulse
	wMinusThreshold	Sets the error tolerance of pulse count in a negative value; unit: pulse
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

29. int MCC_GetErrorCount (

int *pnErrCount0,

int *pnErrCount1,

int *pnErrCount2,

int *pnErrCount3,

int *pnErrCount4,

int *pnErrCount5,

int *pnErrCount6,

int **pnErrCount7,*

WORD *wCardIndex*

)

Description	Acquires pulse error count for each axis	
Parameters	*pnErrCount0~ pnErrCount7*	A pointer to int variable is used to store pulse error count for each axis; unit: pulse
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

L. Advanced Trajectory Planning

---

1. int MCC_HoldMotion(

WORD *wGroupIndex*

)

Description: Pauses motion. This command must be used during motion to be effective. The motion speed will decrease to zero after this command is called. Before decreasing to a zero, the return value of calling MCC_GetMotionStatus() is still GMS_RUNNING, the return value of GMS_HOLD can only be obtained when the motion completely executed.

Parameters	*wGroupIndex*	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_ContiMotion(

WORD *wGroupIndex*

)

Description: Continues running unfinished motion commands. This command must be used when the motion is paused to be effective.

Parameters	*wGroupIndex*	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_AbortMotion(

WORD *wGroupIndex*

)

Description: Stops promptly and aborts all unexecuted motion commands. Note: ***After this command is applied, users can not send motion commands until motion status is GMS_STOP status;*** otherwise the return value of

ABORT_NOT_FINISH_ERR(-15) will be obtained.

Parameters	*wGroupIndex*	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_AbortMotionEx(

double *dfDecTime*,

WORD *wGroupIndex*

)

Description: Sets the deceleration time to pause motion and current remain motion commands will abort. The motion speed will decrease to zero after calling this command. Before decreasing to a zero, the return value of calling MCC_GetMotionStatus() is still GMS_RUNNING, The return value of GMS_STOP can only be obtained when the motion completely stops. Note: ***After this command is applied, users can not send motion commands until motion status is GMS_STOP status;*** otherwise the return value of ABORT_NOT_FINISH_ERR(-15) will be obtained.

Parameters	*dfDecTime*	Required deceleration time
	wGroupIndex	Group number
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_EnableBlend(

WORD *wGroupIndex*

)

Description: Enables path blending. Trajectory planning is conducted by the continuous path method after this command is called. Calling this command successfully will increase the number of stored motion commands.

Parameters	*wGroupIndex*	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return**

Values for the meaning of return value

---

6. int MCC_DisableBlend(

WORD ***wGroupIndex***

)

Description	Disables path blending. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*wGroupIndex*	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_CheckBlend(

WORD ***wGroupIndex***

)

Description	Checks whether path blending has been enabled	
Parameters	*wGroupIndex*	Group number
Return Value	0	Path blending has been enabled
	1	Path blending has not been enabled
	Other	Pommand failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_DelayMotion(

DWORD ***dwTime*****,**

WORD ***wGroupIndex***

)

Description	Sets the motion delay time. Forcibly delays the execution of the next motion command. Calling this command successfully will increase the number of stored motion commands.	
Parameters	*dwTime*	Delay time; unit: ms
	wGroupIndex	Group number
Return Value	>0 or =0	The motion command index given by the MCCL
	<0	Command failed; refer to **Section IV. Command Return**

		Values for the meaning of return value

---

9. int MCC_CheckDelay(

WORD *wGroupIndex*

)

Description	Check current motion delay status (GMS_DELAYING is the return value when MCC_GetMotion Status() is called.)	
Parameters	*wGroupIndex*	Group number
Return Value	0	Not in motion delay status
	1	In motion delay status
	Other	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

10. double MCC_OverrideSpeed(

double *dfRate*,

WORD *wGroupIndex*

)

Description	Sets the override speed rate for general motion. The feed speed of general motion will be promptly altered when this command is used.	
Parameters	*dfRate*	The override speed rate is equal to altered speed divided by original feed speed, and multiply the obtained value by 100. In other words, the new feed speed of general motion will equal (dfFeedSpeed × *dfRate* / 100). dfFeedSpeed is the feed speed set by using MCC_SetFeedSpeed() originally. The setting of *dfRate* must be greater than 0. If the updated feed speed exceeds settings of MCC_SetSysMaxSpeed(), then the new feed speed will equal these settings.
	wGroupIndex	Group number
Return Value	>0	Override speed rate actually set
	Other	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

11. double MCC_OverrideSpeedEx(

double *dfRate*,

BOOL *bInstant*,

WORD *wGroupIndex*

)

Description	Sets the override speed rate for general motion. The feed speed of general motion will be promptly altered when this command is used.	
Parameters	*dfRate*	The override speed rate is equal to altered speed divided by original feed speed, and multiply the obtained value by 100. In other words, the new feed speed of general motion will equal (dfFeedSpeed × *dfRate* / 100). dfFeedSpeed is the feed speed set by using MCC_SetFeedSpeed() originally. The setting of *dfRate* must be greater than 0. If the updated feed speed exceeds settings of MCC_SetSysMaxSpeed(), then the new feed speed will equal these settings.
	bInstant	Override speed rate update timing. 1 (True) indicates to update speed immediately; 0 (False) indicates to update speed with the next command.
	wGroupIndex	Group number
Return Value	>0	Override speed rate actually set
	Other	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

12. double MCC_GetOverrideRate(

WORD *wGroupIndex*

)

Description	Acquires current override speed rate used by general motion	
Parameters	*wGroupIndex*	Group number
Return Value	>0	Current override speed rate used by general motion
	Other	Command failed; refer to **Section IV. Command Return**

Values for the meaning of return value

M. Encoder Control

This section primarily introduces the functions and their usages provided by the encoder module in IMP Series motion control card. The user should read this section along with the one "**Encoder Control**" in "**IMP Series Motion Control Command Library User Manual"**.

---

1. int MCC_SetENCRoutine(
ENCISR_EX *pfnENCRoutine*,
WORD *wCardIndex*
)

Description	Sets customized ISR of ENC	
Parameters	*pfnENCRoutine*	Command index for customized ISR of ENC
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_SetENCInputRate(
WORD *wInputRate*,
WORD *wChannel*,
WORD *wCardIndex*
)

Description	Sets the encoder feedback rate. The impacts of calling this command and the mechanism parameter *wInputRate* are the same to the feedback rate.	
Parameters	*wInputRate*	Encoder feedback rate; can be set as 1, 2, 4
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_ClearENCCounter(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Resets encoder counter	
Parameters	*wChannel*	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. **int MCC_GetENCValue(**

long* *plValue*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Acquires encoder counter value	
Parameters	*plValue*	A pointer to long variable is used to store the encoder counter value
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_SetENCLatchType(

WORD *wType*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets trigger type of encoder counter latch	
Parameters	*wType*	Trigger type; it can be set as: *ENC_TRIG_FIRST*

		When the first triggering condition is met, the count will be latched and not changed *ENC_TRIG_LAST* When the triggering condition is met, the new count will be latches (with unlimited times)
	wChannel	Motion axis number (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_SetENCLatchSource(

WORD *wSource*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description: Sets trigger source of encoder counter latch. Multiple signal sources can be set simultaneously.

For example:

MCC_SetENCLatchSource(ENC_TRIG_INDEX0 | ENC_TRIG_LIO0, 0, 0) means that when the encoder Channel 0 index signal or the Channel 0 positive limit is triggered, the encoder count will be recorded in the latch register for Channel 0 in Card 0.

Parameters:

	wSource	Signal source; can be set as:
	ENC_TRIG_NO	No trigger signal source selected
	ENC_TRIG_INDEX0	Index signal in encoder Channel 0
	ENC_TRIG_INDEX1	Index signal in encoder Channel 1
	ENC_TRIG_INDEX2	Index signal in encoder Channel 2
	ENC_TRIG_INDEX3	Index signal in encoder Channel 3
	ENC_TRIG_INDEX4	Index signal in encoder Channel 4
	ENC_TRIG_INDEX5	Index signal in encoder Channel 5
	ENC_TRIG_INDEX6	Index signal in encoder Channel 6

	ENC_TRIG_INDEX7	Index signal in encoder Channel 7
	ENC_TRIG_OTP0	Positive limit signal in Channel 0
	ENC_TRIG_OTP1	Positive limit signal in Channel 1
	ENC_TRIG_OTP2	Positive limit signal in Channel 2
	ENC_TRIG_OTP3	Positive limit signal in Channel 3
	ENC_TRIG_OTP4	Positive limit signal in Channel 4
	ENC_TRIG_OTP5	Positive limit signal in Channel 5
	ENC_TRIG_OTP6	Positive limit signal in Channel 6
	ENC_TRIG_OTP7	Positive limit signal in Channel 7
	ENC_TRIG_OTN0	Negative limit signal in Channel 0
	ENC_TRIG_OTN1	Negative limit signal in Channel 1
	ENC_TRIG_OTN2	Negative limit signal in Channel 2
	ENC_TRIG_OTN3	Negative limit signal in Channel 3
	ENC_TRIG_OTN4	Negative limit signal in Channel 4
	ENC_TRIG_OTN5	Negative limit signal in Channel 5
	ENC_TRIG_OTN6	Negative limit signal in Channel 6
	ENC_TRIG_OTN7	Negative limit signal in Channel 7
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_GetENCLatchValue(

long* *plValue*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Acquires latch value recorded in the register	
Parameters	*plValue*	A pointer to long variable is used to store latch values recorded in the register
	wChannel	Motion control card output channel (0 ~ 7)

	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_EnableENCIndexTrigger(

WORD *wChannel*,

WORD *w CardIndex*

)

Description Enables *encoder index interrupt function*. The *encoder index interrupt* trigger ISR of ENC

Parameters	*wChannel*	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

9. int MCC_DisableENCIndexTrigger(

WORD *wChannel*,

WORD *wCardIndex*

)

Description Disables *encoder index interrupt function*. The *encoder index interrupt* trigger ISR of ENC

Parameters	*wChannel*	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

10. int MCC_GetENCIndexStatus(

WORD* *pwStatus*,

WORD *wChannel*,

WORD ***wCardIndex***

)

Description	Acquires encoder index status by confirming if the current position is located at the position of index input signal	
Parameters	*pwStatus*	A pointer to WORD variable is used to store the encoder index status. 1 indicates that current located at the index and 0 indicates not at the index
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

11. int MCC_SetENCCompValue(

long ***lValue*,**

WORD ***wChannel*,**

WORD ***wCardIndex***

)

Description	Sets encoder comparative value	
Parameters	*lValue*	Comparative encoder value
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

12. int MCC_EnableENCCompTrigger(

WORD ***wChannel*,**

WORD ***wCardIndex***

)

Description	Enables *encoder compasirson interrupt function*. The *encoder compasirson interrupt* trigger ISR of ENC	
Parameters	*wChannel*	Motion control card output channel (0 ~ 7)

	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

13. int MCC_DisableENCCompTrigger(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Disables *encoder compasirson interrupt function*. The *encoder compasirson interrupt* trigger ISR of ENC	
wChannel	Motion control card output channel (0 ~ 7)	
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

N. Timer and Watchdog Control

This section primarily introduces the functions and their usages provided by the timer and watchdog in IMP Series motion control card. The user should read this section along with the one "**Timer and Watchdog Control**" in **"IMP Series Motion Control Command Library User Manual"**.

---

1. int MCC_SetTimer(
DWORD *dwValue*,
WORD *wCardIndex*
)

Description	Sets the timing cycle of timer. The customized ISR of timer is triggered during each timing cycle.	
Parameters	*dwValue*	Timing cycle; unit: 1us. The setting range is between 1 to 2^{31}
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_EnableTimer(
WORD *wCardIndex*
)

Description	Enables timer	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_DisableTimer(
WORD *wCardIndex*

)

Description	Disables timer	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_EnableTimerTrigger(

WORD *wCardIndex*

)

Description	Enables *timer interrupt* function. The *timer interrupt* trigger ISR of timer during each timing cycle	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_DisableTimerTrigger(

WORD *wCardIndex*

)

Description	Disables *timer interrupt* function. The *timer interrupt* trigger ISR of timer during each timing cycle	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_SetWatchDogTimer(

WORD *wValue*,

WORD *wCardIndex*

)

Description	Sets the countdown time of watchdog. A hardware reset signal will be produced once the watchdog timing ends. Using

MCC_RefreshWatchDogTimer() before the timing ends to reset the watchdog ti

0me, the reset signal would not be produced

Parameters	*dwValue*	Countdown time of the watchdog; unit: 1us. The setting range is between 1 to 2^{31}
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_SetWatchDogResetPeriod(

WORD *wValue*,

WORD *wCardIndex*

)

Description	Sets the duration of hardware reset signal produced when the watchdog timing ends	
Parameters	*wValue*	Duration of hardward reset signal; unit: 10ns
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_EnableWatchDogTimer(

WORD *wCardIndex*

)

Description	Enables the watchdog function	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

9. int MCC_DisableWatchDogTimer(

WORD *wCardIndex*

)

Description	Disables the watchdog function	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

10. int MCC_RefreshWatchDogTimer(

WORD *wCardIndex*

)

Description	Resets the watchdog timer to avoid the production of hardware reset signal when the watchdog timing ends.	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

O. Remote Input and Output Control

This section primarily introduces the functions and their usages provided by the Remote I/O module in IMP Series motion control card. The user should read this section along with the one "**Remote I/O Control**" in **"IMP Series Motion Control Command Library User Manual"**.

---

1. int MCC_EnableARIOSetControl (

WORD ***wSet*****,**

WORD ***wCardIndex***

)

Description	Enables the indicated Remote I/O Set data transfer. The slave data transfer function of a given set needs to be enabled by calling MCC_EnableARIOSlaveControl ().	
Parameters	*wSet*	I/O set number. There is only one I/O set in IMP**(OK)**
		RIO_SET0 Remote I/O set **(OK)**
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_DisableARIOSetControl (

WORD ***wSet*****,**

WORD ***wCardIndex***

)

Description	Disables the indicated Remote I/O Set data transfer. The slave data transfer function of a given set will be disabled as well.	
Parameters	*wSet*	I/O set number. There is only one I/O set in IMP**(OK)**
		RIO_SET0 remote I/O set 0**(OK)**
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return**

Values for the meaning of return value

---

3. int MCC_EnableARIOSlaveControl (

WORD ***wSet*****,**

WORD ***wSlave*****,**

WORD ***wCardIndex***

)

Description	Enables the indicated Remote I/O Slave data transfer. Once the slave data transfer function is enabled, it is necessary to call MCC_EnableARIOSetControl () again to enable the set data transfer function so that the remote I/O module will start transmission and reception.	
Parameters	*wSet*	I/O set number. There is only one I/O set in IMP(OK)
		RIO_SET0 Remote I/O set 0(OK)
	wSlave	Slave number of digital input (0~31)
		RIO_Slave0 Slave 0, the slave card whose address is 0
		RIO_Slave1 Slave 1, the slave card whose address is 1
		.....
		RIO_Slave31 Slave 31, the slave card whose address is 31
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_DisableARIOSlaveControl (

WORD ***wSet*****,**

WORD ***wSlave*****,**

WORD ***wCardIndex***

)

Description	Disables the indicated Remote I/O Slave data transfer	
Parameters	*wSet*	I/O set number. There is only one I/O set in IMP(OK)
		RIO_SET0 Remote I/O set 0(OK)
	wSlave	Slave number of digital input (0~31)

		RIO_Slave0 Slave 0, the slave card whose address is 0 RIO_Slave1 Slave 1, the slave card whose address is 1 ..... RIO_Slave31 Slave 31, the slave card whose address is 31
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_GetARIOTransStatus(

WORD* ***pwStatus*****,**

WORD ***wSet*****,**

WORD ***wCardIndex***

)

Description	Acquires current remote I/O data transfer status. If the transfer has stopped, the user can call MCC_GetARIOMasterStatus() and MCC_GetARIOSlaveStatus() to identify the error is produced on the master or slave .	
Parameters	*pwStatus*	A pointer to WORD variable is used to store remote I/O data transfer status. 1 indicates that remote I/O set is normally operating and 0 indicates not normally
	wSet	I/O Set number. There is only one I/O set in IMP(OK) RIO_SET0 Remote I/O set 0(OK)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_GetARIOMasterStatus(

WORD* ***pwStatus*****,**

WORD ***wSet*****,**

WORD ***wCardIndex***

)

Description	Acquires current status of remote I/O master data transmission to slave	
Parameters	*pwStatus*	A pointer to WORD variable is used to store remote I/O data transfer status. 1 indicates that remote I/O master terminal is receiving signals normally and 0 indicates not normally
	wSet	I/O set number. There is only one I/O set in IMP**(OK)**
		RIO_SET0 Remote I/O set 0**(OK)**
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_GetARIOSlaveStatus(

DWORD* *pwStatus*,

WORD *wSet*,

WORD *wCardIndex*

)

Description	Acquires the current status of Remote I/O slave reception from master data	
Parameters	*pwStatus*	A pointer to DWORD variable is used to store remote I/O slave reception from master data. 0 indicates that remote I/O master terminal is receiving signals from slave normally and 1 indicates not normally (bit 0 ~ bit 31 respectively represents the status of slave 0 to slave 31)
	wSet	I/O set number. There is only one I/O set in IMP**(OK)**
		RIO_SET0 Remote I/O set 0**(OK)**
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_GetARIOInputValue(

WORD* *pwValue*,

WORD ***wSet*****,**
WORD ***wSlave*****,**
WORD ***wCardIndex***
)

Description	Acquires the 16-Bit digital input value of indicated set and slave	
Parameters	*pwValue*	A pointer to WORD variable is used to store the 16-bit digital input value in the indicated position (set, slave) (bit 0 ~ bit 15 respectively represents the value of input 0 to input 15 in the port)
	wSet	I/O set number. There is only one I/O set in IMP(OK)
		RIO_SET0 Remote I/O set 0(OK)
	wSlave	Slave number of digital input (0~31)
		RIO_Slave0 Slave 0, the slave card whose address is 0
		RIO_Slave1 Slave 1, the slave card whose address is 1
		.....
		RIO_Slave31 Slave 31, the slave card whose address is 31
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

9. int MCC_SetARIOOutputValue(
WORD ***wValue*****,**
WORD ***wSet*****,**
WORD ***wSlave*****,**
WORD ***wCardIndex***
)

Description	Sets the 16-Bit digital output value of indicated set and slave	
Parameters	*wValue*	The 16-bit digital output value in the indicated position (set, slave) (bit 0 ~ bit 15 respectively represents the value of output 0 to output 15 in this slave)
	wSet	I/O set number. There is only one I/O set in IMP(OK)

		RIO_SET0 Remote I/O set 0(OK)
	wSlave	Slave number of digital input (0~31)
		RIO_Slave0 Slave 0, the slave card whose address is 0
		RIO_Slave1 Slave 1, the slave card whose address is 1
		.....
		RIO_Slave31 Slave 31, the slave card whose address is 31
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

P. Digital to Analog Converter Control

This section primarily introduces the functions and their usages provided by the DAC module in IMP Series motion control card. The user should read this section along with the one "**Analog Voltage Output Control**" in **"IMP Series Motion Control Command Library User Manual"**.

---

1. int MCC_SetDACOutput(

float *fVoltage*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Outputs indicated voltage	
Parameters	*fVoltage*	Output voltage (-10V ~ 10 V)
	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_SetDACTriggerOutput(

float *fVoltage*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	When output axes (0 ~ 7) in the motion control card are not using velocity command mode (meaning the mechanism parameter *wCommandMode* is set as OCM_PULSE), a voltage can be preprogrammed in the DAC module. When trigger conditions are met, the hardware will immediately output this preprogrammed voltage.	
Parameters	*fVoltage*	Preprogrammed voltage output (-10V ~ 10 V)

	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

3. int MCC_SetDACTriggerSource(

DWORD *dwSource*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description: This command can be used to set the source of triggering the preprogrammed output voltage when indicated axis is not in velocity command mode. Each D/A converter channel can set various trigger sources. After setting this command, it is required to use MCC_EnableDACTriggerMode() again to enable the trigger mode.

Parameters:

dwSource	D/A converter trigger source; can be set as:	
	DAC_TRIG_ENC0	Specific count in encoder channel 0
	DAC_TRIG_ENC1	Specific count in encoder channel 1
	DAC_TRIG_ENC2	Specific count in encoder channel 2
	DAC_TRIG_ENC3	Specific count in encoder channel 3
	DAC_TRIG_ENC4	Specific count in encoder channel 4
	DAC_TRIG_ENC5	Specific count in encoder channel 5
	DAC_TRIG_ENC6	Specific count in encoder channel 6
	DAC_TRIG_ENC7	Specific count in encoder channel 7

	wChannel	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_EnableDACTriggerMode(

WORD *wChannel*,

WORD *wCardIndex*

)

Description This command can be used to enable the function of triggering the preprogrammed voltage output when indicated axis is not in velocity command mode. Please set the trigger source before enabling the trigger mode.

Parameters	*wChannel*	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_DisableDACTriggerMode(

WORD *wChannel*,

WORD *wCardIndex*

)

Description This command can be used to disable the function of triggering the preprogrammed voltage output when indicated axis is not in velocity command mode.

Parameters	*wChannel*	Motion control card output channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_StartDACConv(

WORD *wCardIndex*

)

Description	This command can be used to enable voltage output when all channels in the indicated motion control card are not in velocity command mode.	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_StopDACConv(

WORD *wCardIndex*

)

Description	Disable voltage output.	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

Q. Analog to Digital Converter Control

This section primarily introduces the functions and their usages provided by the ADC module in IMP Series motion control card. The user should read this section along with the one "**Analog Voltage Input Control**" in **"IMP Series Motion Control Command Library User Manual"**.

---

1. int MCC_SetADCRoutine(

ADCISR *pfnADCRoutine*,

WORD *wCardIndex*

)

Description	Sets customized ISR of ADC	
Parameters	*pfnADCRoutine*	Command index for customized ADC ISR
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

2. int MCC_SetADCConvType(

WORD *wConvType*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets the ADC voltage conversion type as bipolar or unipolar	
Parameters	*wConvType*	Voltage conversion type setting
		ADC_TYPE_BIP Bipolar Converter Type
		ADC_TYPE_UNI Unipolar Converter Type
	wChannel	A/D converter channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return**

		Values for the meaning of return value

---

3. int MCC_GetADCInput(

float* *pfInput*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Acquires the DC input of the indicated ADC channel. When the ADC is indicated as “unipolar”, then the effective input voltage of IMP Series motion control card is 0~10V. When the ADC is set as “bipolar”, then the effective input voltage of IMP Series motion control card is -5V~+5V.	
Parameters	*pfInput*	A pointer to float variable is used to store DC input in ADC channel
	wChannel	A/D converter channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

4. int MCC_SetADCCompType(

WORD *wCompType*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets comparative type of voltage conversion. Calling MCC_EnableADCCompTrigger() after this command, an ADC interrupt signal can be triggered when this ADC channel input voltage satisfies comparative conditions. In addition to triggering customized ISR, this signal can also be used to trigger the DAC module preprogrammed voltage output. The first two groups of the ADC channel trigger signal can be used to latch the encoder count as well.	
Parameters	*wCompType*	Voltage comparative type; types can be set as:
		ADC_COMP_RISE Input voltage is compared from

		the least to the greatest *ADC_COMP_FALL* Input voltage is compared from the greatest to the least *ADC_COMP_LEVEL* Input voltage is changed and compared
	wChannel	A/D converter channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

5. int MCC_SetADCCompValue(

float *fValue*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets the ADC comparative channel voltage input value under bipolar mode. This command does not supply the comparative function under unipolar mode. After setting this command, it is required to use MCC_Set ADCCompType() and MCC_EnableADCCompTrigger(). An ADC interrupt signal can be triggered when this ADC channel input voltage satisfies comparative conditions.	
Parameters	*fValue*	Voltage input comparative value (IMP Series motion control card can be set between -5V~5V)
	wChannel	A/D converter channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

6. int MCC_GetADCCompValue(

float* *pfValue*,

WORD *wChannel*,

WORD ***wCardIndex***

)

Description	Acquires comparative value used	
Parameters	*pfValue*	A pointer to float variable is used to store the voltage input comparative value
	wChannel	A/D converter channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

7. int MCC_EnableADCConvChannel (

WORD ***wChannel*,**

WORD ***wCardIndex***

)

Description	Enables the ADC of voltage conversion in the ADC channel. The conversion channel set by this command must use a free run mode. MCC_StartADCConv() must be called after the setting is completed to initiate ADC conversion.	
Parameters	*wChannel*	A/D converter channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

8. int MCC_DisableADCConvChannel (

WORD ***wChannel*,**

WORD ***wCardIndex***

)

Description	Disables the ADC of voltage conversion in the ADC channel.	
Parameters	*wChannel*	A/D converter channel (0 ~ 7)
	wCardIndex	Motion control card number (0 ~ 5)
Return Value	0	Successful

	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

9. int MCC_StartADCConv(

WORD *wCardIndex*

)

Description	Starts ADC channel analog voltage conversion. This command must be used with MCC_EnableADCConvChannel ().	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

10. int MCC_StopADCConv(

WORD *wCardIndex*

)

Description	Stops all ADC channel analog voltage conversion	
Parameters	*wCardIndex*	Motion control card number (0 ~ 5)
Return Value	0	Successful
	not 0	Command failed; refer to **Section IV. Command Return Values** for the meaning of return value

---

III. ERROR CODES

Error code	Description
0xF101	Motion control command library has not been initialized yet
0xF104	Given parameters during curve motion commands are unreasonable
0xF203	Feed speed exceeds the allowed pulse output for each interpolation time
0xF204	Feed speed exceeds the allowed pulse increment for each interpolation time
0xF301	X axis position exceeds work limits set by mechanism parameters
0xF302	Y axis position exceeds work limits set by mechanism parameters
0xF303	Z axis position exceeds work limits set by mechanism parameters
0xF304	U axis position exceeds work limits set by mechanism parameters
0xF305	V axis position exceeds work limits set by mechanism parameters
0xF306	W axis position exceeds work limits set by mechanism parameters
0xF307	A axis position exceeds work limits set by mechanism parameters
0xF308	B axis position exceeds work limits set by mechanism parameters
0xF401	An error occurred during motion command execution
0xF501	In position confirmation error
0xF701	X axis has triggered a hardware limit switch
0xF702	Y axis has triggered a hardware limit switch
0xF703	Z axis has triggered a hardware limit switch
0xF704	U axis has triggered a hardware limit switch
0xF705	V axis has triggered a hardware limit switch
0xF706	W axis has triggered a hardware limit switch
0xF707	A axis has triggered a hardware limit switch
0xF708	B axis has triggered a hardware limit switch
0xF801	Tracking error of X axis has exceeded the set tolerance
0xF802	Tracking error of Y axis has exceeded the set tolerance
0xF803	Tracking error of Z axis has exceeded the set tolerance
0xF804	Tracking error of U axis has exceeded the set tolerance
0xF805	Tracking error of V axis has exceeded the set tolerance
0xF806	Tracking error of W axis has exceeded the set tolerance
0xF807	Tracking error of A axis has exceeded the set tolerance
0xF808	Tracking error of B axis has exceeded the set tolerance

IV. COMMAND RETURN VALUES

Return Value Definition	Number Value	Description
NO_ERR	0	Command successful
INITIAL_MOTION_ERR	-1	System has not been activated; please recall MCC_InitSystem()
COMMAND_BUFFER_FULL_ERR	-2	Motion command queue is full; unable to accept this command at this time.
COMMAND_NOTACCEPTED_ERR	-3	System is busy; unable to accept this command at this time.
COMMAND_NOTFINISHED_ERR	-4	Motion command execution unfinished; unable to accept this command at this time.
PARAMETER_ERR	-5	Incoming parameter format error when the command was called
GROUP_PARAMETER_ERR	-6	Group parameter setting error; the indicated group is invalid
FEED_RATE_ERR	-7	Feed speed not set or set incorrectly; please recall MCC_SetFeedSpeed()
VOLTAGE_COMMAND_NOTCALLED_ERR	-9	Using this command is inhibited because the system or this motion axis is using the velocity command mode
HOME_COMMAND_NOTCALLED_ERR	-10	Currently not under Go Home mode
HOLD_ILLEGAL_ERR	-11	Inappropriate time to hold command
CONTI_ILLEGAL_ERR	-12	Inappropriate time to continue command
ABORT_ILLEGAL_ERR	-13	Inappropriate time to abort command
RUN_TIME_ERR	-14	Running time error; use the error code obtained by calling MCC_GetErrorCode() to find out the error content
ABORT_NOT_FINISH_ERR	-15	Command abortion not finished
GROUP_RAN_OUT_ERR	-16	No more groups for using

V. MOTION CONTROL COMMAND LIBRARY DEFAULT SETTINGS

The following table lists default settings of MCCL after MCC_InitSystem() is called. The user can call related commands for amendment if initial setting cannot meet the user's requirements.

Setting Content	Default Setting	Related Command
Command queue size	10000 commands	MCC_SetCmdQueueSize() MCC_GetCmdQueueSize()
Dry run function	Disabled	MCC_EnableDryRun() MCC_DisableDryRun()
Maximum feed speed permitted by the machine	100	MCC_SetSysMaxSpeed() MCC_GetSysMaxSpeed()
System position type	Absolute position	MCC_SetAbsolute() MCC_SetIncrease() MCC_GetCoordType()
Maximum pulse acceleration permitted at each axis	32767	MCC_SetMaxPulseAcc() MCC_GetMaxPulseAcc()
Maximum pulse speed permitted at each axis	32767	MCC_SetMaxPulseSpeed() MCC_GetMaxPulseSpeed()
Software over-travel check	Disabled	MCC_SetOverTravelCheck() MCC_GetOverTravelCheck
Hardware limit switch check	Disabled	MCC_EnableLimitSwitchCheck() MCC_DisableLimitSwitchCheck()
Proportional gain used in position control closed loop	64	MCC_SetPGain() MCC_GetPGain()
Acceleration and deceleration types used by each axis during linear, curve, circular and helix motions	S curve	MCC_SetAccType() MCC_GetAccType() MCC_SetDecType() MCC_GetDecType()
Acceleration and deceleration times used by each axis during linear, curve, circular and helix motions	300 ms	MCC_SetAccTime() MCC_GetAccTime() MCC_SetDecTime() MCC_GetDecTime()
Feed speed used during linear, curve, circular and helix motions	1	MCC_SetFeedSpeed() MCC_GetFeedSpeed()
Point-to-point motion speed ratio for each axis	1	MCC_SetPtPSpeed() MCC_GetPtPSpeed()
In position confirmation maximum check time	1000 ms	MCC_SetInPosMaxCheckTime()
In position confirmation settling time	100 ms	MCC_SetInPosSettleTime()
In position confirmation error tolerance	∞	MCC_SetInPosToleranceEx() MCC_GetInPosToleranceEx()
In position confirmation function	Disabled	MCC_EnableInPos() MCC_DiableInPos()
Path blend function	Disabled	MCC_EnableBlend() MCC_DisnableBlend()
Tracking error function	Disabled	MCC_EnableTrackError() MCC_DisnableTrackError()

Tracking error tolerance	∞	MCC_SetTrackErrorLimit() MCC_GetTrackErrorLimit()

Revision History

Date	Version	Summary of Changes
2010/06	1.00	Newly established
2013/01	1.01	－Revised descriptions of each command and aligned the layout. －P.33，Revised the description of IMC_GetLimitSwitchLatchStatus(). －P.69，Revised the return value description of IMC_GetMotionStatus(). －P.88，Added descriptions of MCC_SetErrorCountThreshold() and MCC_GetErrorCount(). －P.92，Added the description of MCC_OverrideSpeedEx(). －P.97，Revised the description of IMC_SetENCLatchSource() external triggering signal source. －Revised descriptions of each command in section O.Remote I/O and deleted several obsolete commands. －Deleted Q. Several obsolete commands in section A/D Converter Control.