EPCIO Series
Motion Control Command Library
Reference Manual

(Applicable to Motion Control Command Library V.510)

Version: V.5.10

Date: 2009.10

http://www.epcio.com.tw

Table of Contents

I. Motion Control Command 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 configuration
8	MCC_SetHomeConfig()	Sets Go Home configuration
9	MCC_UpdateParam()	Makes the system respond to updated parameters
10	MCC_SetCmdQueueSize()	Sets motion command queue size
11	MCC_GetCmdQueueSize()	Acquires command queue size
12	MCC_InitSystem()	Initiates motion control command library
13	MCC_CloseSystem()	Closes motion control command library
14	MCC_ResetMotion()	Resets motion control command library
15	MCC_EnableDryRun()	Enables motion dry run function
16	MCC_DisableDryRun()	Disables motion dry run function
17	MCC_CheckDryRun()	Checks motion dry run function status
18	MCC_SetSysMaxSpeed()	Sets maximum feed rate speed for general motion
19	MCC_GetSysMaxSpeed()	Acquires maximum feed rate speed for general motion

B. Local Input/Outpu Control

No.	Command Name	Description
1	MCC_SetServoOn()	Enables servo
2	MCC_SetServoOff()	Disables servo
3	MCC_EnablePosReady()	Enables outputs position ready signal
4	MCC_DisablePosReady()	Disables outputs position ready signal
5	MCC_GetEmgcStopStatus()	Acquires emergency stop switch input status
6	MCC_SetLIORoutinEx()	Serially connects customized interrupt service routine (ISR) of local I/O
7	MCC_SetLIOTriggerType()	Sets local I/O interruption trigger type

8	MCC_EnableLIOTrigger()	Enables ISR function to trigger local I/O signal
9	MCC_DisableLIOTrigger()	Disables ISR function to trigger local I/O signal

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 positions for each axial position (excluding compensation)
5	MCC_GetCurPos()	Acquires positions for each axial position (including compensation)
6	MCC_GetPulsePos()	Acquires motor positions (pulse) for each axial position (including compensation)
7	MCC_DefineOrigin()	Defines current position as origin
8	MCC_DefinePosHere()	Justifies the current coordinate position and the actual machine position
9	MCC_DefinePos ()	Sets current system positions

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

E. Line, Curve, Circular, and Helix Motion (General Motion)

No.	Command Name	Description

1	MCC_SetAccDecMode()	Sets acceleration/deceleration mode for general motion
2	MCC_GetAccDecMode()	Acquires acceleration/deceleration mode for general motion
3	MCC_SetAccType()	Sets acceleration type
4	MCC_GetAccType()	Acquires acceleration type used
5	MCC_SetDecType()	Sets deceleration type
6	MCC_GetDecType()	Acquires deceleration type used
7	MCC_SetAccTime()	Sets acceleration time
8	MCC_GetAccTime()	Acquires acceleration time used
9	MCC_SetDecTime()	Sets deceleration time
10	MCC_GetDecTime()	Acquires deceleration time used
11	MCC_SetFeedSpeed()	Sets feed rate speed
12	MCC_GetFeedSpeed()	Acquires feed rate speed used
13	MCC_GetCurFeedSpeed()	Acquires current machine feed rate speed
14	MCC_GetSpeed()	Acquires current speed for each axis
15	MCC_Line()	Six-axis simultaneous linear motion
16	MCC_ArcXYZ()	Three-point curve motion on the XYZ plane
17	MCC_ArcXYZUVW()	Three-point curve motion on the XYZ plane and executing linear motion on the U,V and W axes
18	MCC_ArcXY()	Curve motion on XY plane
19	MCC_ArcYZ()	Curve motion on YZ plane
20	MCC_ArcZX()	Curve motion on ZX plane
21	MCC_ArcXYUVW()	Curve motion on XY plane and executing linear motion on the U,V and W axes
22	MCC_ArcYZUVW()	Curve motion on YZ plane with linear motion along assistance axis
23	MCC_ArcZXUVW()	Curve motion on ZX plane with linear motion along assistance axis
24	MCC_Arc Theta XY()	Curve motion on XY plane (with rotational angle as a parameter)
25	MCC_Arc Theta YZ()	Curve motion on YZ plane (with rotational angle as a parameter)
26	MCC_Arc Theta ZX()	Curve motion on ZX plane (with rotational angle as a parameter)
27	MCC_CircleXY()	Complete circular motion on XY plane
28	MCC_CircleYZ()	Complete circular motion on YZ plane
29	MCC_CircleZX()	Complete circular motion on ZX plane
30	MCC_CircleXYUVW()	Complete circular motion on XY plane and executing linear motion on the U,V and W axes

31	MCC_CircleYZUVW()	Complete circular motion on YZ plane and executing linear motion on the U,V and W axes
32	MCC_CircleZXUVW()	Complete circular motion on ZX plane and executing linear motion on the U,V and W axes
33	MCC_HelicaXY_Z()	Helix motion with circular motion on XY plane
34	MCC_HelicaYZ_X()	Helix motion with circular motion on YZ plane
35	MCC_HelicaZX_Y()	Helix motion with circular motion on ZX plane

F. Point-to-Point Motion

No.	Command Name	Description
1	MCC_SetPtPSpeed()	Sets speed ratio
2	MCC_GetPtPSpeed()	Acquires speed ratio used
3	MCC_PtP()	Point-to-point motion
4	MCC_SetPtPAccType()	Sets acceleration type
5	MCC_GetPtPAccType()	Acquires acceleration type used
6	MCC_SetPtPDecType()	Sets deceleration type
7	MCC_GetPtPDecType()	Acquires deceleration type used
8	MCC_SetPtPAccTime()	Sets acceleration time
9	MCC_GetPtPAccTime()	Acquires acceleration time used
10	MCC_SetPtPDecTime()	Sets deceleration time
11	MCC_GetPtPDecTime()	Acquires deceleration time used

G. JOG Motion

No.	Command Name	Description
1	MCC_JogPulse()	Pulse motion
2	MCC_JogSpace()	Inch motion
3	MCC_JogConti()	Continuous inch motion

H. Motion Status Check

No.	Command Name	Description
1	MCC_GetMotionStatus()	Acquires current motion status

2	MCC_GetCurCommand()	Acquires information related to motion commands currently being executed
3	MCC_GetCommandCount()	Acquires the amount of motion commands in storage
4	MCC_ResetCommandIndex()	Resets motion command index number
5	MCC_GetCurPulseStockCount()	Acquires amount of pulse commands stored in current hardware
6	MCC_GetErrorCode()	Acquires existing error codes
7	MCC_ClearError()	Deletes existing error codes

I. Go Home

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

J. Position Control

No.	Command Name	Description
1	MCC_SetCompParam()	Sets parameters for gear backlash and backlash compensation
2	MCC_UpdateCompParam()	Responds to updated parameters for gear backlash and backlash 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_SetMaxPulseSpeed()	Sets maximum pulse speed for each axis
6	MCC_GetMaxPulseSpeed()	Acquires maximum pulse speed for each axis
7	MCC_SetMaxPulseAcc()	Sets maximum pulse acceleration for each axis
8	MCC_GetMaxPulseAcc()	Acquires maximum pulse acceleration for each axis
9	MCC_SetInPosMode()	Sets in position mode
10	MCC_SetInPosMaxCheckTime()	Sets in position check time
11	MCC_SetInPosSettleTime()	Sets in position settle time
12	MCC_EnableInPos()	Enables in position function

13	MCC_DisableInPos()	Disables in position function
14	MCC_SetInPosToleranceEx()	Sets extent of in position error tolerance
15	MCC_GetInPosToleranceEx()	Acquires extent of in position error tolerance used
16	MCC_GetInPosStatus()	Confirms whether actual position satisfies in position status
17	MCC_EnableTrackError()	Enables error tracking function
18	MCC_DisableTrackError()	Disables error tracking function
19	MCC_SetTrackErrorLimit()	Sets the maximum error tolerance range of tracking limits
20	MCC_GetTrackErrorLimit()	Acquires the maximum error tolerance range of tracking limits
21	MCC_SetPCLRoutine()	Serially connects customized position control loop ISR

K. Advanced Trajectory Planning

No.	Command Name	Description
1	MCC_HoldMotion()	Pauses motion
2	MCC_ContiMotion()	Continues motion
3	MCC_AbortMotionEx()	Decelerates to a stop at the set deceleration time and aborts unexecuted motion commands
4	MCC_EnableBlend()	Enables path blending
5	MCC_DisableBlend()	Disables path blending
6	MCC_CheckBlend()	Checks whether path blending has been enabled
7	MCC_DelayMotion()	Sets motion delay time
8	MCC_CheckDelay()	Checks motion delay status
9	MCC_OverrideSpeed()	Sets general motion override speed rate
10	MCC_GetOverrideRate()	Acquires general motion override speed rate
11	MCC_OverridePtPSpeed()	Sets point-to-point override speed rate
12	MCC_GetPtPOverrideRate()	Acquires point-to-point override speed rate

L. Encoder Control

No.	Command Name	Description
1	MCC_SetENCRoutineEx()	Serially connects customized encoder ISR
2	MCC_SetENCInputRate()	Sets encoder feedback rate
3	MCC_ClearENCCounter()	Resets encoder counter

4	MCC_GetENCValue()	Acquires encoder count
5	MCC_SetENCLatchType()	Sets triggering type for latch encoder counter
6	MCC_SetENCLatchSource()	Sets triggering signal source of latch encoder counter
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

M. Timer and Watchdog Control

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

N. Remote Input/Output Control

No.	Command Name	Description
1	MCC_SetRIORoutineEx()	Serially connects customized Remote I/O ISR
2	MCC_EnableRIOSetControl()	Enables indicated Remote I/O Set control
3	MCC_DisableRIOSetControl()	Disables indicated Remote I/O Set control
4	MCC_EnableRIOSlaveControl()	Enables indicated Remote I/O Slave data

		transmission
5	MCC_DisableRIOSlaveControl()	Disables indicated Remote I/O Slave data transmission
6	MCC_GetRIOTransStatus()	Acquires current Remote I/O data transmission status
7	MCC_GetRIOMasterStatus()	Acquires current status of Remote I/O Master data transmission to Slave
8	MCC_GetRIOSlaveStatus()	Acquires current status of Remote I/O Slave, which received data from Master
9	MCC_GetRIOInputValue()	Acquires indicated Set and Port 16-Bit Digital Input signal status val ue
10	MCC_SetRIOOutputValue()	Sets the indicated Set and Port 16-Bit Digital Output signal status value
11	MCC_EnquRIOOutputValue()	Sets the indicated Set and Port 16-Bit Digital Output signal status value (This command will be send to the command queue)
12	MCC_SetRIOTransError()	Sets the number times to resend when a Remote I/O data transmission error occurs
13	MCC_SetRIOTriggerType()	Sets the method for triggering ISR with the Remote I/O Digital Input signal
14	MCC_EnableRIOInputTrigger()	Enables the method for triggering ISR with the Remote I/O Digital Input signal
15	MCC_DisableRIOInputTrigger()	Disables the method for triggering ISR with the Remote I/O Digital Input signal
16	MCC_EnableRIOTransTrigger()	Enables function triggering ISR with Remote I/O “Transmission Error”
17	MCC_DisableRIOTransTrigger()	Disables function triggering ISR with Remote I/O “Transmission Error”

O. Digital to Analog Converter Control

No.	Command Name	Description
1	MCC_SetDACOutput()	Outputs indicated voltage
2	MCC_SetDACTriggerOutput()	Sets pre-programmed voltage output
3	MCC_SetDACTriggerSource()	Sets source for output to trigger pre-programmed voltage
4	MCC_EnableDACTriggerMode()	Enables source for output to trigger pre-programmed voltage
5	MCC_DisableDACTriggerMode()	Disables source for output to trigger pre-programmed voltage
6	MCC_StartDACConv()	Enables voltage output
7	MCC_StopDACConv()	Disables voltage output

P. Analog to Digital Converter Control

No.	Command Name	Description
1	MCC_SetADCRoutine()	Serially connects customized ISR of ADC
2	MCC_SetADCConvType()	Sets voltage conversion type to unipolar or bipolar
3	MCC_GetADCConvType()	Acquires voltage conversion type used
4	MCC_SetADCConvMode()	Sets voltage conversion mode to Single or Free Running Mode
5	MCC_GetADCInput()	Acquires DC input
6	MCC_SetADCSingleChannel()	Sets single voltage conversion channel
7	MCC_GetADCWorkStatus()	Acquires single conversion channel work status
8	MCC_EnableADCConvTrigger()	Enables function to trigger customized ADC ISR when any channel voltage conversion is completed
9	MCC_DisableADCConvTrigger()	Disables function to trigger customized ADC ISR when any channel voltage conversion is completed
10	MCC_SetADCTagChannel()	Sets voltage conversion tag channel
11	MCC_EnableADCTagTrigger()	Enables function to trigger customized ADC ISR when tag channel voltage conversion is completed
12	MCC_DisableADCTagTrigger()	Disables function to trigger customized ADC ISR when tag channel voltage conversion is completed
13	MCC_SetADCCompMask()	Sets masking bit of voltage conversion
14	MCC_SetADCCompType()	Sets comparative voltage of conversion conditions
15	MCC_SetADCCompValue()	Sets conversion value of comparative voltage
16	MCC_GetADCCompValue()	Acquires comparative value used
17	MCC_EnableADCCompTrigger()	Enables function to trigger ADC ISR when ADC comparative conditions are met
18	MCC_DisableADCCompTrigger()	Disables function to trigger ADC ISR when ADC comparative conditions are met
19	MCC_EnableADCConvChannel()	Enables selected channel of voltage conversion
20	MCC_DisableADCConvChannel()	Disables selected channel of voltage conversion
21	MCC_StartADCConv()	Starts voltage conversion
22	MCC_StopADCConv()	Stops voltage conversion

II. Motion Control Command Library

A. System Functions

1. void MCC_GetVersion(

char* ***strVersion***

)

Description	Acquires command library version	
Parameters	*strVersion*	Indicates a memory buffer used to receive the command library version
Return Value	0	Command successful
	≠0	Command failed; for the specific meaning of return values, please refer to **Section IV. Command Return Values**

2. int MCC_CreateGroup(

int ***xMapToCh***,

int ***yMapToCh***,

int ***zMapToCh***,

int ***uMapToCh***,

int ***vMapToCh***,

int ***wMapToCh***,

int ***nCardIndex***

)

Description	This command establishes a new motion group. This command is required to establish a motion group and to acquire the new motion group’s number as (one of) its input parameter before calling a command related to motion groups in the motion control command library (Example: MCC_Line).

This command must be called before Motion Control Command Library initialization. Call MCC_CloseAllGroups before calling this command for the first time.

Note: Any two motion axes cannot correspond to the same physical output channel.

Parameters

xMapToCh Assigns the physical output channels (0 - 5) that corresponds to the X axis in this group

yMapToCh Assigns the physical output channels (0 - 5) that corresponds to the Y axis in this group

zMapToCh Assigns the physical output channels (0 - 5) that corresponds to the Z axis in this group

uMapToCh Assigns the physical output channels (0 - 5) that corresponds to the U axis in this group

vMapToCh Assigns the physical output channels (0 - 5) that corresponds to the V axis in this group

wMapToCh Assigns the physical output channels (0 - 5) that corresponds to the W axis in this group

nCardIndex Assigns the motion control card numbers (0 - 11) 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 for the newly established group

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

---

3. int MCC_CloseGroup(

int *nGroupIndex*

)

Description Closes the group indicated

Parameters *wGroupIndex* Group number

Return Value 0 Command successful

	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

4. int MCC_CloseAllGroups()

Description	Closes all of the groups in the system. Call this command before calling MCC_CreateGroup for the first time.	
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

5. int MCC_SetMacParam(

SYS_MAC_PARAM* *pstMacParam*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets the mechanism parameters for each axis	
Parameters	*pstMacParam*	Indicates a SYS_MAC_PARAM structure containing the mechanism parameters with the desired settings
	wChannel	Motion control card output channel (0 - 5)
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

6. int MCC_GetMacParam(

SYS_MAC_PARAM* *pstMacParam*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Acquires the mechanism parameter content for the axis indicated	
Parameters	*pstMacParam*	Indicates a SYS_MAC_PARAM structure used to receive the desired mechanism parameter content
	wChannel	Motion control card output channel (0 - 5)
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

7. int MCC_SetEncoderConfig(

SYS_ENCODER_CONFIG* ***pstEncoderConfig***,
WORD ***wChannel***,
WORD ***wCardIndex***
)

Description	Sets the encoder configuration	
Parameters	*pstEncoderConfig*	Indicates a SYS_ENCODER_CONFIG structure containing the encoder parameters desired
	wChannel	Motion control card output channel (0 - 5)
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

8. int MCC_SetHome Config(

SYS_HOME_CONFIG* ***pstHomeConfig***,
WORD ***wChannel***,
WORD ***wCardIndex***
)

Description	Sets the Go Home configuration	
Parameters	*pstHomeConfig*	Indicates a SYS_HOME_CONFIG structure

		containing a Go Home configuration with the desired settings
	wChannel	Motion control card output channel (0 - 5)
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

9. int MCC_ UpdateParam()

Description	Responds to updated mechanisms, encoders, and Go Home parameters. If MCC_SetMacParam and MCC_SetEncoderConfig are used again to change related parameters after MCC_InitSystem has been called, this command is required to allow the system to respond to the updated settings. Please note that similar to MCC_ResetMotion, the system will reset to its initial status when this command is called.

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

10. int MCC_SetCmdQueueSize(

int *nSize*,

WORD *wGroupIndex*

)

Description	Sets the size of the motion command queue	
Parameters	*nSize*	Motion command queue size (in units of motion commands)
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

11. int MCC_GetCmdQueueSize(

WORD *wGroupIndex*

)

Description	Acquires the size of the motion command queue	
Parameters	*wGroupIndex*	Group number
Return Value	≥0	Motion command queue size (in terms of motion commands)
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

12. int MCC_InitSystem(

int *nInterpolateTime*,

SYS_CARD_CONFIG* *pstCardConfig*,

WORD *wCardNo*

)

Description	Enables motion control command library Excluding MCC_CreateGroup, MCC_SetMacParam, MCC_SetEncoderConfig, MCC_SetHomeConfig, and MCC_SetCompParam, calling this command is required prior to using other commands in the motion control command library. This command only needs to be used once.	
Parameters	*nInterpolateTime*	Interpolation time in the unit of ms, ranging from 1 ms to 50 ms. Shorter interpolation times create better operational capacity in the motion control command library, but the PC's load capacity must first be confirmed. Generally, the PCs can be set to 5 ms.
	pstCardConfig	Motion control card hardware parameters; for a detailed description of hardware parameters, please refer to the “EPCIO

		Series Motion Control Command Library User Manual.”
	wCardNo	Number of motion control cards used (1 - 12)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

13. int MCC_CloseSystem()

Description Disables the motion control command library

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

14. int MCC_ ResetMotion()

Description Resets the motion control command library. This command will clear the error status, restore the Cartesian or pulse position to zero, and return the system to the initial status as it was after MCC_InitSystem was called.

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

15. int MCC_ EnableDryRun()

Description Enables the dry run function. Motion command calculation will still occur after this function is enabled, but the calculated results will not be output. Instead, MCC_GetCurPos and MCC_GetPulsePos can be used to acquire the positions required for analysis or drawing.

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

16. int MCC_DisableDryRun()

Description	Disables the dry run function	
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

17. int MCC_CheckDryRun()

Description	Checks the status of the dry run function	
Return Value	0	Dry run has been enabled
	1	Dry run has been disabled
	Other	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

18. int MCC_SetSysMaxSpeed(
double *dfMaxSpeed*
)

Description	Sets the maximum feed rate speed for general motion (line, curve, circular, and helix) to prevent the feed rate speed that was set using MCC_SetFeedSpeed from exceeding the system work limitations; the unit used is User Unit/sec*	
Parameters	*dfMaxSpeed*	Maximum feed rate speed
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

19. double MCC_GetSysMaxSpeed()

Description	Acquires the maximum feed rate speed for general motion (line, curve, circular, and helix); the unit used is User Unit/sec
Return Value	Maximum feed rate speed

***NOTE: User Unit (hereafter referred to as UU) is the unit of length (angle) selected (i.e. dfPitch, dfHighLimit, dfLowLimit) when the user sets the mechanism parameters. Once selected, the same unit of length (angle) will be used throughout the motion control command library.**

B. Local Input/Output Control

---

1. int MCC_SetServoOn(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Enables the servo	
Parameters	*wChannel*	Motion control card output channel (0 - 5)
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

2. int MCC_SetServoOff(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Disables the servo	
Parameters	*wChannel*	Motion control card output channel (0 - 5)
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

3. int MCC_EnablePosReady(

WORD *wCardIndex*

)

Description	Output signal from Position Ready output on the motion control card	
Parameters	*wCardIndex*	Motion control card number (0 - 11)

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

4. int MCC_DisablePosReady(
WORD *wCardIndex*
)

Description	Stops output signal from Position Ready output on the motion control card	
Parameters	*wCardIndex*	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

5. int MCC_GetEmgcStopStatus(
WORD* *pwStatus*,
WORD *wCardIndex*
)

Description: Acquires the emergency stop switch status. To enable this function, please refer to the method for setting the emergency stop in the relevant hardware user manual. For example, the JP1 is default short circuit in an EPCIO-4000/4005 motion control card. When the system is connected to the Emergency Stop circuit, the JP1 must be used the open circuit to avoid vibration in the Emergency Stop section.

Parameters	*pwStatus*	Indicates WORD value representing the emergency stop switch input status 0 Disabled 1 Enabled
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful

≠0 Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

6. int MCC_SetLIORoutineEx(

LIOISR_EX *pfnLIORoutine*,

WORD *wCardIndex*

)

Description Serially connects the Local I/O ISR; for a detailed description, please refer to "**EPCIO Series Motion Control Command Library User Manual.**"

Parameters *pfnLIORoutine* Index for the customized Local I/O ISR command

wCardIndex Motion control card number (0 - 11)

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

7. int MCC_SetLIOTriggerType(

WORD *wTriggerType*,

WORD *wPoint*,

WORD *wCardIndex*

)

Description Sets the trigger type of the Local I/O signal for ISR as rising edge, falling edge, or level change.

Parameters *wTriggerType* Trigger type can be set as

LIO_INT_RISE	Rising edge
LIO_INT_FALL	Falling edge
LIO_INT_LEVEL	Level change

wPoint Local I/O number ranges from LIO_LDI0 to LIO_LDI6 (0-6); for input signal meanings, please refer to "**EPCIO Series Motion Control Command Library User Manual.**"

	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

8. int MCC_EnableLIOTrigger(

WORD *wPoint*,

WORD *wCardIndex*

)

Description	Enables function triggering Local I/O customized ISR by signal	
Parameters	*wPoint*	Local I/O number ranges from LIO_LDI0 to LIO_LDI6 (0~6); for input signal meanings, please refer to "**EPCIO Series Motion Control Command Library User Manual.**"
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

9. int MCC_DisableLIOTrigger(

WORD *wPoint*,

WORD *wCardIndex*

)

Description	Disables function triggering Local I/O customized ISR by signal	
Parameters	*wPoint*	Local I/O number ranges from LIO_LDI0 to LIO_LDI6 (0~6); for input signal meanings, please refer to "EPCIO Series Motion Control Command Library User Manual."
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful

≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

C. Positioning System

1. in t MCC_SetAbsolute(

WORD *wGroupIndex*

)

Description	Adopts absolute position mode	
Parameters	*wGroupIndex*	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

2. int MCC_SetIncrease(

WORD *wGroupIndex*

)

Description	Adopts incremental position mode	
Parameters	*wGroupIndex*	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

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; for the meaning of return values, please refer to **Section IV. Command Return Values**

4. int MCC_GetCurRefPos(

double* ***pdfX***,
double* ***pdfY***,
double* ***pdfZ***,
double* ***pdfU***,
double* ***pdfV***,
double* ***pdfW***,
WORD ***wGroupIndex***
)

Description	Acquires the current Cartesian position for each axis ***(excluding compensation)***	
Parameters	*pdfX-pdfW*	Indicates a double value used to store the current Cartesian position for each axis X to W (excluding compensation)
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

5. int MCC_GetCurPos(

double* ***pdfX***,
double* ***pdfY***,
double* ***pdfZ***,
double* ***pdfU***,
double* ***pdfV***,
double* ***pdfW***,
WORD ***wGroupIndex***
)

Description	Acquires the current Cartesian position for each axis ***(including compensation)***	
Parameters	*pdfX-pdfW*	Indicates a double value used to store the current Cartesian position values for each axis X to W (including compensation)
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

6. int MCC_GetPulsePos(

long* ***plX***,
long* ***plY***,
long* ***plZ***,
long* ***plU***,
long* ***plV***,
long* ***plW***,
WORD ***wGroupIndex***
)

Description	Acquires the current motor position (unit: pluse) for each axis ***(also termed the pulse position, including compensation)***	
Parameters	*pdfX-pdfW*	Indicates a long value used to store the current motor position (unit: pluse) values for each axis X to W (including compensation)
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

7. int MCC_DefineOrigin(
WORD *wAxis*,
WORD *wGroupIndex*
)

Description	Resets the indicated motion axis position value in a specified group to zero; the indicated group motion must be stopped to use this command	
Parameters	*wAxis*	Indicated motion axes 0 to 5 represent axes X to W, respectively
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

8. int MCC_DefinePosHere(
WORD *wGroupIndex*,
DWORD *dwAxisMask*
)

Description

Revises the current system position value to align with the actual machine position. Under certain circumstances, the machine can be moved manually, creating a discrepancy between the machine's actual position and the system position value in the motion control command library. If an encoder is installed in the system, use the encoder counter after successfully using this command to revise the system position. The system position value will react to the actual position of the machine.

Parameters	*wGroupIndex*	Group number
	dwAxisMask	Set the axis that performs the desired action; the assigned parameters could be:

EPCIO_AXIS_X	X axis
EPCIO_AXIS_Y	Y axis
EPCIO_AXIS_Z	Z axis
EPCIO_AXIS_U	U axis
EPCIO_AXIS_V	V axis

EPCIO_AXIS_W	W axis
EPCIO_AXIS_ALL	All motion axes

The above parameters can be combined. For example, X, Z, and V:

(EPCIO_AXIS_X | EPCIO_AXIS_Z | EPCIO_AXIS_V)

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

9. int MCC_DefinePos (

double *dfCart,*

WORD *wAxis,*

WORD *wGroupIndex*

)

Description	Defines the current system position value. The user can reset the current position value. After this command is successfully called, the system position will update to the newly set position.	
Parameters	*dfCart*	Currently set position (mm)
	wAxis	Set the axis that performs the desired action; the assigned parameters could be:
		0 X axis
		1 Y axis
		2 Z axis
		3 U axis
		4 V axis
		5 W axis
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

D. Over-Travel Protection

1. int MCC_EnableLimitSwitchCheck(

int *nMode*

)

Description

Enables limit switch protection. Mechanism parameters *wOverTravelUpSensorMode* and *wOverTravelDownSensorMode* must be set to 0 (Normal Open) or 1 (Normal Close) to accurately execute this function.

Once this command is enabled, output group motion command will be terminated (and an error recorded) if a limit switch in the direction of the given axis is triggered (for example, traveling in a forward direction and triggering a forward directional limit switch, or traveling in a reverse direction and triggering a reverse directional limit switch).

MCC_EnableLimitSwitchCheck() will generally be used in combination with MCC_GetErrorCode(). Continuously calling MCC_GetErrorCode () provides information as to whether a limit switch has been triggered and an error recorded (codes 0xF701 to 0xF706 represent limit switches triggered by axes X to W, respectively). When an error from triggering a limit switch is discovered, a common process might first involve a message displayed on the screen alerting the operator that an error was found. Then call MCC_ClearError during programming to clear the error record, at which point the system will reverse direction away from the limit switch.

Parameters

nMode Hardware limit switch protection mode

Possible settings:

0 Output axis motion command will be stopped once a limit switch is triggered.

1 Output axis motion command will be stopped only when a limit switch in the same direction as the system is triggered (for example, traveling in a forward direction and

triggering a forward directional limit switch, or traveling in a reverse direction and triggering a reverse directional limit switch).

2 Output axis motion command will be stopped and an error recorded once a limit switch is triggered.

3 Output axis motion command will be stopped and an error recorded only when a limit switch in the same direction as the system is triggered (for example, traveling in a forward direction and triggering a forward directional limit switch, or traveling in a reverse direction and triggering a reverse directional limit switch).

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

2. int MCC_DisableLimitSwitchCheck()

Description Disables limit switch protection

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

3. int MCC_SetOverTravelCheck(

int *nCheck0*,

int *nCheck1*,

int *nCheck2*,

int *nCheck3*,

int *nCheck4*,

int *nCheck5*,

WORD *wGroupIndex*

)

Description Sets software over-travel protection

Parameters *nCheck0, nCheck1, nCheck2, nCheck3, nCheck4, nCheck5* are the setting parameters; 1 indicates that the software over-travel protection for this axis should be enabled; 0 indicates that it should be disabled.

	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

4. int MCC_GetOverTravelCheck(

int* *pnChk0*,

int* *pnChk1*,

int* *pnChk2*,

int* *pnChk3*,

int* *pnChk4*,

int* *pnChk5*,

WORD *wGroupIndex*

)

Description Acquires the settings for software over-travel protection

Parameters	*pnChk0-pnChk5*	Indicates an int value used to store the current software over-travel protection settings for each axis X~W 1 indicates that it has been enabled 0 indicates that it has been disabled
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

5. int MCC_GetLimitSwitchStatus(

WORD* ***pwStatus***,

WORD ***wUpDown***,

WORD ***wChannel***,

WORD ***wCardInde*** **x**

)

Description	Acquires the limit switch status. The limit switch wiring method must be accurately defined before this command can be used. The wiring method is defined in the mechanism parameters *wOverTravelUpSensorMode* and *wOverTravelDownSensorMode*.	
Parameters	*pwStatus*	Indicates WORD value used to store the limit switch status; 1 indicates that a limit switch has currently been triggered; 0 indicates it has not
	wUpDown	0 indicates reverse limit switch status, 1 indicates forward limit switch status
	wChannel	Motion control card output channel (0 - 5)
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

E. Line, Curve, Circular, and Helix Motion (General Motion)

1. **int MCC_SetAccDecMode(**
   char *cAccDecMode*,
   WORD *wGroupIndex*
)

Description Sets the acceleration and deceleration modes for general motion

Parameters *cAccDecMode* Acceleration modes for each axis

Possible settings:

‘A’ Post-acceleration and deceleration mode

‘B’ Pre-acceleration and deceleration mode

wGroupIndex Group number

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

2. **int MCC_GetAccDecMode(**
   WORD *wGroupIndex*
)

Description Acquires acceleration and deceleration modes for general motion

Parameters *wGroupIndex* Group number

Return Value 0 Currently uses Post-acceleration and deceleration mode

1 Currently uses Pre-acceleration and deceleration mode

3. **int MCC_SetAccType(**
   char *cAccType*,
   WORD *wGroupIndex*

)

Description	Sets acceleration mode for general motion type	
Parameters	*cAccType*	Acceleration type for each axis Possible settings: 'T' to use trapezoidal acceleration curve 'S' to use S acceleration curve
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

4. int MCC_GetAccType(

WORD *wGroupIndex*

)

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

---

5. int MCC_SetDecType(

char *cDecType*,

WORD *wGroupIndex*

)

Description	Sets deceleration mode for general motion type	
Parameters	*cDecType*	Deceleration type for each axis Possible settings: 'T' to use trapezoidal deceleration curve 'S' to use S deceleration curve
	wGroupIndex	Group number
Return Value	0	Command successful

	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

6. int MCC_GetDecType(

WORD *wGroupIndex*

)

Description	Acquires deceleration type used for general motion	
Parameters	*wGroupIndex*	Group number
Return Value	0	Currently uses trapezoidal deceleration curve
	1	Currently uses S deceleration curve

7. int MCC_SetAccTime(

double *dfAccTime*,

WORD *wGroupIndex*

)

Description	Sets the time required for general motion to accelerate to a stable speed	
Parameters	*dfAccTime*	Time required for acceleration, greater than 0 ms.
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

8. 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	Time required for general motion to accelerate to a stable speed, in units of ms.	

9. int MCC_SetDecTime(

double *dfDecTime*,
WORD *wGroupIndex*
)

Description	Sets the time required for general motion to decelerate to a stop	
Parameters	*dfDecTime*	Time required to decelerate, greater than 0 ms.
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

10. double MCC_GetDecTime(

WORD *wGroupIndex*
)

Description	Acquires the time required for general motion to decelerate to a stop	
Parameters	*wGroupIndex*	Group number
Return Value	Time required for general motion to decelerate to a stop, in units of ms.	

11. double MCC_SetFeedSpeed(

double *dfFeedSpeed* ,
WORD *wGroupIndex*
)

Description

Sets the feed rate speed for general motion in UU/sec, but the value cannot equal to zero. The feed rate speed for general motion during actual operation (obtained using MCC_GetCurFeedSpeed()) must consider whether MCC_OverrideSpeed is used to set the feed speed rate ratio. For example, if MCC_SetFeedSpeed(10) is called when the last feed rate speed ratio was set using MCC_OverrideSpeed(150), the actual feed rate speed for general motion used is 10×150 % = 15

Parameters	*dfFeedSpeed*	Required feed rate speed

wGroupIndex Group number

Return Value Actual feed rate speed set

---

12. double MCC_GetFeedSpeed(

WORD *wGroupIndex*

)

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

Parameters *wGroupIndex* Group number

Return Value Current set return value

---

13. double MCC_GetCurFeedSpeed(

WORD *wGroupIndex*

)

Description Acquires the machine's current actual feed rate speed

Parameters *wGroupIndex* Group number

Return Value Machine's current actual feed rate speed

---

14. int MCC_GetSpeed(

double* *pdfV0*,

double* *pdfV1*,

double* *pdfV2*,

double* *pdfV3*,

double* *pdfV4*,

double* *pdfV5*,

WORD *wGroupIndex*

)

Description Acquires current feed rate speed for each axis

Parameters *pdfV0~pdfV5* Indicates a double value used to store the current feed rate speed for each axis

	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

15. int MCC_Line(

double ***dfX0***,

double ***dfX1***,

double ***dfX2***,

double ***dfX3***,

double ***dfX4***,

double ***dfX5***,

WORD ***wGroupIndex***,

DWORD ***dwAxisMask***

)

Description	Moves the current position in a line to the destination indicated. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dfX0-dfX5*	Destination position value
	wGroupIndex	Group number
	dwAxisMask	Axis that performs the desired action Possible parameters:

EPCIO_AXIS_X	X AXIS
EPCIO_AXIS_Y	Y AXIS
EPCIO_AXIS_Z	Z AXIS
EPCIO_AXIS_U	U AXIS
EPCIO_AXIS_ V	V AXIS
EPCIO_AXIS_ W	W AXIS
EPCIO_AXIS_ALL	ALL MOTION AXES

The above parameters can be combined. For example, X, Z, and V:

(EPCIO_AXIS_X | EPCIO_AXIS_Z | EPCIO_AXIS_V)

Return Value	≥0	Command index given to this motion command in the motion control command library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

16. int MCC_ArcXYZ(

double *dfRX0*,

double *dfRX1*,

double *dfRX2*,

double *dfX0*,

double *dfX1*,

double *dfX2*,

WORD *wGroupIndex*

)

Description Moves in a curve from the current position through the indicated reference point to the destination within the space constructed by the XYZ axes. Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfRX0 -dfRX2*	XYZ axis position values of the reference point
	dfDX0-dfDX2	XYZ axis position values of the destination point
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control command library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

17. int MCC_ArcXYZUVW(

double *dfRX0*,

double *dfRX1*,

double *dfRX2*,

double *dfX0*,

double *dfX1*,

double ***dfX2***,

double ***dfX3***,

double ***dfX4***,

double ***dfX5*****,**

WORD ***wGroupIndex***

)

Description Moves in a curve from the current position through the indicated reference point to the destination within the space constructed by the XYZ axes, while simultaneously executing linear movement on the U, V, and W axes. Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfRX0 -dfRX2*	XYZ axis position values of the reference point
	dfX0-dfX2 dfDX0-dfDX2	XYZ axis position values of the destination point
	dfX3-dfX5	UVW axis position values of the destination point
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control command library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

18. int MCC_ArcXY(

double ***dfRX0***,

double ***dfRX1***,

double ***dfX0***,

double ***dfX1***,

WORD ***wGroupIndex***

)

Description Moves in a curve from the current position through the indicated reference point to the destination within the plane constructed by the XY axes. Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfRX0, dfRX1*	XY axis position values of the reference point

	dfX0, *dfX1*	XY axis position values of the destination point
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

19. int MCC_ArcYZ(

double *dfRX1*,

double *dfRX2*,

double *dfX1*,

double *dfX2*,

WORD *wGroupIndex*

)

Description: Moves in a curve from the current position through the indicated reference point to the destination within the plane constructed by the YZ axes. Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfRX1*, *dfRX2*	YZ axis position values of the reference point
	dfX0, *dfX1*	YZ axis position values of the destination point
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

20. int MCC_ArcZX(

double *dfRX2*,

double *dfRX0*,

double *dfX2*,

double *dfX0*,

WORD *wGroupIndex*

)

Description Moves in a curve from the current position through the indicated reference point to the destination within the plane constructed by the ZX axes. Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfRX2, dfRX0*	ZX axis position values of the reference point
	dfX2, dfX0	ZX axis position values of the destination point
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

21. int MCC_ArcXYUVW(

double *dfRX0*,

double *dfRX1*,

double *dfX0*,

double *dfX1*,

double *dfX3*,

double *dfX4*,

double *dfX5*,

WORD *wGroupIndex*

)

Description Moves in a curve from the current position through the indicated reference point to the destination within the plane constructed by the XY axes, while simultaneously executing linear movement on the U, V, and W axes. Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfRX0, dfRX1*	XY axis position values of the reference point
	dfX0, dfX1	XY axis position values of the destination point
	dfX3-dfX5	UVW axis position values of the destination point

	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

22. int MCC_ArcYZUVW(

double ***dfRX1***,
double ***dfRX2***,
double ***dfX1***,
double ***dfX2***,
double ***dfX3***,
double ***dfX4***,
double ***dfX5***,
WORD ***wGroupIndex***

)

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

Parameters	*dfRX1*, *dfRX2*	YZ axis position values of the reference point
	dfX1, *dfX2*	YZ axis position values of the destination point
	dfX3-*dfX5*	UVW axis position values of the destination point
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

23. int MCC_ArcZXUVW(

double *dfRX2*,

double *dfRX0*,

double *dfX2*,

double *dfX0*,

double *dfX3*,

double *dfX4*,

double *dfX5*,

WORD *wGroupIndex*

)

Description	Moves in a curve from the current position through the indicated reference point to the destination within the plane constructed by the ZX axes, while simultaneously executing linear movement on the U, V, and W axes. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dfRX2, dfRX0*	ZX axis position values of the reference point
	dfX2, dfX0	ZX axis position values of the destination point
	dfX3-dfX5	UVW axis position values of the destination point
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

24. int MCC_ArcThetaXY(

double *dfX0*,

double *dfX1*,

double *dfTheta*,

WORD *wGroupIndex*

)

Description Moves in a curve around the indicated epicenter at the indicated angle in the plane constructed by the XY axes. A negative angle translates to clockwise motion, while a positive angle translates to counter-clockwise motion. Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfX0*, *dfX1*	Position values of the indicated epicenter
	dfTheta	Motion angle
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

25. int MCC_ArcThetaYZ(

double ***dfX1***,

double ***dfX2***,

double ***dfTheta***,

WORD ***wGroupIndex***

)

Description: Moves in a curve around the indicated epicenter at the indicated angle in the plane constructed by the YZ axes. A negative angle translates to clockwise motion, while a positive angle translates to counter-clockwise motion. Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfX1*, *dfX2*	Position values of the indicated epicenter
	dfTheta	Motion angle
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

26. int MCC_ArcThetaZX(

double ***dfX2***,

double ***dfX0***,

doubled ***dfTheta***,

WORD ***wGroupIndex***

)

Description	Moves in a curve around the indicated epicenter at the indicated angle in the plane constructed by the ZX axes. A negative angle translates to clockwise motion, while a positive angle translates to counter-clockwise motion. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dfX2*, *dfX0*	Position values of the indicated epicenter
	dfTheta	Motion angle
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

27. int MCC_CircleXY(

double ***dfCX0***,

double ***dfCX1***,

BYTE ***byCirDir***,

WORD ***wGroupIndex***

)

Description	Moves in a complete circular trajectory around the indicated epicenter in the plane constructed by the XY axes. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dfCX0, dfCX1*	XY axis position values of the epicenter
	byCirDir	Direction: 0 = clockwise; 1 = counter-clockwise
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in

		the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

28. int MCC_CircleYZ(

double *dfCX1*,

double *dfCX2*,

BYTE *byCirDir*,

WORD *wGroupIndex*

)

Description	Moves in a complete circular trajectory around the indicated epicenter in the plane constructed by the YZ axes. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dfCX1*, *dfCX2*	YZ axis position values of the epicenter
	byCirDir	Direction: 0 = clockwise; 1 = counter-clockwise
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

29. int MCC_CircleZX(

double *dfCX2*,

double *dfCX0*,

BYTE *byCirDir*,

WORD *wGroupIndex*

)

Description	Moves in a complete circular tracjetory around the indicated epicenter in the plane constructed by the ZX axes. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dfCX2*, *dfCX0*	ZX axis position value of the epicenter

	byCirDir	Direction: 0 = clockwise; 1 = counter-clockwise
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

30. int MCC_CircleXYUVW(

double ***dfCX0***,

double ***dfCX1***,

double ***dfU***,

double ***dfV***,

double ***dfW***,

BYTE ***byCirDir***,

WORD ***wGroupIndex***

)

Description	Moves in a complete circular trajectory around the indicated epicenter in the plane constructed by the XY axes, while simultaneously executing linear movement on the U, V, and W axes. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dfCX0, dfCX1*	XY axis position values of the epicenter
	dfU, dfV, dfW	UVW axis position values of the destination
	byCirDir	Direction: 0 = clockwise; 1 = counter-clockwise
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

31. int MCC_CircleYZUVW(

double *dfCX1*,

double *dfCX2*,

double *dfU*,

double *dfV*,

double *dfW*,

BYTE *byCirDir*,

WORD *wGroupIndex*

)

Description	Moves in a complete circular trajectory around the indicated epicenter in the plane constructed by YZ axes, while simultaneously executing linear movement on the U, V, and W axes. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dfCX1*, *dfCX2*	YZ axis position values of the epicenter
	dfU, *dfV*, *dfW*	UVW axis position values of the destination
	byCirDir	Direction: 0 = clockwise; 1 = counter-clockwise
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

32. int MCC_CircleZXUVW(

double *dfCX2*,

double *dfCX0*,

double *dfU*,

double *dfV*,

double *dfW*,

BYTE *byCirDir*,

WORD *wGroupIndex*

)

Description

Moves in a complete circular trajectory around the indicated epicenter in the plane constructed by ZX axes, while simultaneously executing linear movement on the U, V, and W axes. Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfCX2, dfCX0*	ZX axis position values of the epicenter
	dfU, dfV, dfW	UVW axis position values of the destination
	byCirDir	Direction: 0 = clockwise; 1 = counter-clockwise
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

33. int MCC_HelicaXY_Z(

double *dfCX*,

double *dfCY*,

double *dfZ*,

double *dfPitch*,

BYTE *byCirDir*,

WORD *wGroupIndex*

)

Description

Moves in a helix motion from the current position. This is a circular motion along the XY plane, and the speed is set using MCC_SetFeedSpeed(). Before this command can be used, the epicenter position, the radius (determined by the distance between the current position and the epicenter), and the destination position on the Z axis must all be indicated. Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfCX, dfCY*	XY axis position values of the epicenter
	dfZ	Z axis position value of the destination point

dfPitch Distance moved on the Z axis after one complete circular motion on the XY plane; must be greater than 0.

byCirDir Direction: 0 = clockwise; 1 = counter-clockwise

wGroupIndex Group number

Return Value ≥0 Command index given to this motion command in the motion control library

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

---

34. int MCC_HelicaYZ_X(

double *dfCY*,

double *dfCZ*,

double *dfX*,

double *dfPitch*,

BYTE *byCirDir*,

WORD *wGroupIndex*

)

Description Moves in a helix motion from the current position. This is a circular motion along the YZ plane, and the speed is set using MCC_SetFeedSpeed(). Before this command can be used, the epicenter position, the radius (determined by the distance between the current position and the epicenter), and the destination position on the X axis must all be indicated. Successfully calling this command will increase the number of stored motion commands.

Parameters *dfCY*, *dfCZ* YZ axis position values of the epicenter

dfX X axis position value of the destination point

dfPitch Distance moved on the X axis after one complete circular motion on the YZ plane; must be greater than 0.

byCirDir Direction: 0 = clockwise; 1 = counter-clockwise

wGroupIndex Group number

Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

35. int MCC_HelicaZX_Y(

double *dfCZ*,

double *dfCX*,

double *dfY*,

double *dfPitch*,

BYTE *byCirDir*,

WORD *wGroupIndex*

)

Description	Moves in a helix motion from the current position. This is a circular motion along the ZX plane, and the speed is set using MCC_SetFeedSpeed(). Before this command can be used, the epicenter position, the radius (determined by the distance between the current position and the epicenter), and the destination position on the Y axis must all be indicated. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dfCZ, dfCX*	ZX axis position values of the epicenter
	dfY	Y axis position value of the destination point
	dfPitch	Distance moved on the Y axis after one complete circular motion on the ZX plane; must be greater than 0.
	byCirDir	Direction: 0 = clockwise; 1 = counter-clockwise
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

F. Point-to-Point Motion

1. double MCC_SetPtPSpeed(
double *dfRatio*,
WORD *wGroupIndex*
)

Description

Sets the point-to-point speed ratio. Each axis speed (mm/s) for point-to-point motion equals

((Maximum rotational speed /60) x pitch number/gear deceleration ratio) x (speed ratio/100)

where the maximum rotational speed (*wRPM*), pitch number (*dfPitch*), and gear deceleration ratio (*dfGearRatio*) are all defined in the mechanism parameters. Therefore, the speed ratio can be obtained by dividing the ratio of the required speed by the maximum speed at which the maximum rotational speed can drive the machine, and multiplying the obtained percentage by 100.

However, the feed rate speed during actual operation of point-to-point motion requires consideration of the use of MCC_OverrideSpeed() to set the motion feed rate speed ratio.

Parameters

dfRatio	Speed ratio; must range between 0 and 100
wGroupIndex	Group number

Return Value

Actual speed ratio set

2. double MCC_GetPtPSpeed(
WORD *wGroupIndex*
)

Description

Acquires the speed ratio used during point-to-point motion

Parameters

wGroupIndex	Group number

Return Value

≥0	Speed ratio used during point-to-point motion

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

3. int MCC_PtP(

double *dfX*,

double *dfY*,

double *dfZ*,

double *dfU*,

double *dfV*,

double *dfW*,

WORD *wGroupIndex*,

DWORD *dwAxisMask*

)

Description	Moves from the current position to the indicated destination using point-to-point motion at the set feed rate speed ratio. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dfX, dfY, dfZ*	XYZ axis position values of the destination position
	dfU, dfV, dfW	UVW axis position values of the destination position
	wGroupIndex	Group number
	dwAxisMask	Axis performing the desired action Possible parameters EPCIO_AXIS_X X axis EPCIO_AXIS_Y Y axis EPCIO_AXIS_Z Z axis EPCIO_AXIS_U U axis EPCIO_AXIS_ V V axis EPCIO_AXIS_ W W axis EPCIO_AXIS_ALL All motion axes The above parameters can be combined. For example, X, Z, and V: (EPCIO_AXIS_X | EPCIO_AXIS_Z | EPCIO_AXIS_V)
Return Value	≥0	Command index given to this motion command in

		the motion control library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

4. int MCC_SetPtPAccType(char *cType0*, char *cType1*, char *cType2*, charc*Type3*,charc*Type4*,charc*Type5*,WORD *wGroupIndex*)

Description	Sets the acceleration type for point-to-point motion; each axis uses an independent acceleration type.	
Parameters	cType0~cType5	Acceleration type for each axis Possible settings: **'T' to use trapezoidal acceleration curve** **'S' to use S acceleration curve**
	wGroupIndex	**Group number (WINDOWS: 0-71**; **DOS: 0-3)**
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

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

Description	Acquires the acceleration type for point-to-point motion	
Parameters	*pcType0-*pcType5	Acceleration type for each axis 0 indicates use of Trapezoidal acceleration curve 1 indicates use of acceleration curve
wGroupIndex	**Group number (WINDOWS: 0~71; DOS: 0~3)**	
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

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

Description	Sets the deceleration type for point-to-point motion; each axis uses an independent deceleration type.	
Parameters	cType0~cType5	Deceleration type for each axis Possible settings: **'T' to use trapezoidal deceleration curve** **'S' to use S deceleration curve**
	wGroupIndex	**Group number (WINDOWS: 0~71; DOS: 0~3)**
Return Value	**0**	**Command successful**
	≠0	**Command failed; for the meaning of return values, please refer to Section IV. Command Return Values**

---

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

Description	Acquires the deceleration type for point-to-point motion	
Parameters	*pcType0~*pcType5	Deceleration type for each axis 0 indicates use of trapezoidal deceleration curve 1 indicates use of deceleration curve
	wGroupIndex	**Group number (WINDOWS: 0~71; DOS: 0~3)**
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

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

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

Parameters	dfTime0~dfTime5	Acceleration time for each axis in ms; must be greater than zero
	wGroupIndex	**Group number (WINDOWS: 0~71; DOS: 0~3)**
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

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

Description	Acquires the acceleration time required for point-to-point motion to achieve stable speed; each axis uses an independent acceleration time.	
Parameters	pdfTime0~pdfTime5	Acceleration time for each axis in ms; must be greater than zero
	wGroupIndex	**Group number (WINDOWS: 0~71; DOS: 0~3)**
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

10. int MCC_SetPtPDecTime(double dfTime0, double dfTime1, double dfTime2, double dfTime3, double dfTime4, double dfTime 5, WORD wGroupIndex)

Description	Sets the deceleration time required for point-to-point motion from stable speed to a stop; each axis uses an independent deceleration time.	
Parameters	dfTime0~dfTime5	Deceleration time for each axis in ms
	wGroupIndex	**Group number (WINDOWS: 0~71; DOS: 0~3)**
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

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

Description	Acquires the deceleration time required for point-to-point motion from stable speed to a stop; each axis uses an independent deceleration time.	
Parameters	pdfTime0~pdfTime5	Deceleration time for each axis in ms
	wGroupIndex	**Group number (WINDOWS: 0~71; DOS: 0~3)**
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

G. JOG Motion

---

1. int MCC_JogPulse(

int *nPulse*,

char *cAxis*,

WORD *wGroupIndex*

)

Description

Pulse motion. After all other motion commands have been executed (at which the return value from calling MCC_GetMotionStatus() should be GMS_STOP), the specified axis is driven according to the indicated displacement (pulses) and direction.

This command is a manually programmed fine-tuning mode which requires the motion status to be at “stop” to be effective. Pulse motion lacks acceleration or deceleration; therefore, to avoid excessive machine vibration, the set displacement should not be overly large.

Parameters

nPulse	Displacement in pulses within a given range of -2048 to 2048.
cAxis	The motion axis number required for pulse motion (0~5 represents axes X~W)
wGroupIndex	Group number

Return Value

0	Command successful
≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

2. int MCC_JogSpace(

double *dfOffset*,

int *nRatio*,

char *cAxis*,

WORD *wGroupIndex*

)

Description

Inch motion (step motion). After all other motion commands have been executed (at which the return value from calling MCC_GetMotionStatus() should be GMS_STOP), the specified axis is driven according to the indicated displacement (increment) and speed ratio (same meaning as point-to-point speed ratio). Successfully calling this command will increase the number of stored motion commands.

Parameters	*dfOffset*	Displacement in UU units
	nRatio	Speed ratio; must range between 0 to 100
	cAxis	The motion axis number required for step motion (0~5 represents axes X to W)
	wGroupIndex	Group number
Return Value	≥0	Motion command index in motion control command library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

3. int MCC_JogConti(

int *nDir*,

int *nRatio*,

char *cAxis*,

WORD *wGroupIndex*

)

Description

Continuous inch motion. After all other motion commands have been executed (at which the return value from calling MCC_GetMotionStatus() should be GMS_STOP), the specified axis is driven according to the indicated direction and speed ratio (same meaning as point-to-point speed ratio) to the edge of the effective work zone before stopping (the mechanism parameters define the limits of the effective work zone). Successfully calling this command will increase the number of stored motion commands.

Parameters	*nDir*	Continuous motion direction Possible settings: 1 Forward motion -1 Reverse motion
	nRatio	Speed ratio; must range between 0 to 100
	cAxis	The motion axis number required for step motion (0~ 5 represents axes X to W)
	wGroupIndex	Group number
Return Value	≥0	Motion command index in motion control command library
	<0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

H. Motion Status Check

---

1. int MCC_GetMotionStatus(
WORD *wGroupIndex*
)

Description	Checks the current motion status of the system	
Parameters	*wGroupIndex*	Group number
Return Value	GMS_RUNNING	Running status; motion command has yet to be completed
	GMS_STOP	Stop status; no stored motion commands
	GMS_HOLD	Hold status (if the user called MCC_HoldMotion)
	GMS_DELAYING	Delaying status (if the user called MCC_DelayMotion)
	Other	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

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

Description: Acquires information about the motion commands being executed; including motion command type, motion command index, required feed rate speed, and destination.

Parameters: *pstCurCmdInfo* Indicates a COMMAND_INFO structure used to store content about motion commands being executed, and 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	Enable path blending
8	Disable path blending
9	Enable position confirmation
10	Disable position confirmation

nCommandIndex: Motion command index

dfFeedSpeed:

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

dfPos[]: Absolute position of destination

	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

3. **int MCC_GetCommandCount(**

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

   WORD ***wGroupIndex***

)

Description: Acquires the amount of yet unexecuted motion commands stored in the motion command queue.

Regarding which commands will increase the commands stored, please refer to the "**Command Library Operational Properties**" section in the "**EPCIO Series Motion Control Command Library User Manual.**"

Parameters	*pnCmdCount*	Indicates the int value used to store the number of motion commands
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

4. **int MCC_ResetCommandIndex(**

WORD *wGroupIndex*

)

Description: Resets the motion command index to zero. The motion command index is the relative identifying data given to each motion command in the motion control command library. By using this command, the motion command index can be counted from 0.

Parameters	*wGroupIndex*	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

5. int MCC_GetCurPulseStockCount(

WORD* *pwStockCount*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description: Acquires the number of hardware pulses in the command library. To guarantee stable motion control, the number of pulses stored in the command library should not be less than 60 during motion. If this number is unattainable, please increase the interpolation time (recall MCC_InitSystem).

Parameters	*pwStockCount*	Indicates a WORD value used to store the stock pulse count
	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

6. 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 100 ms) during system operation to confirm that the system is currently operating normally. If a record of an error is found, please perform the corresponding error recovery process.

Parameters	*wGroupIndex*	Group number
Return Value	0	No error found
	Other	Error code (please refer to **Section IV. Error Codes**)

---

7. int MCC_ClearError (

WORD *wGroupIndex*

)

Description: After an error has occurred during system operation, if the error has been removed, this command is required to clear the record of the error in the system; otherwise, the system will be unable to operate normally.

Parameters	*wGroupIndex*	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

I. Go Home

1. int MCC_Home(

int *nOrder0*,

int *nOrder1*,

int *nOrder2*,

int *nOrder3*,

int *nOrder4*,

int *nOrder5*,

WORD *wCardIndex*

)

Description	Performs the Go Home motion. For the Go Home settings, please refer to MCC_SetHomeConfig(). This command can be used in conjunction with MCC_GetGoHomeStatus() to check whether the action has finished. Once the Go Home motion has finished, the position for each axis should be set to zero.	
Parameters	*nOrder0-nOrder5*	The Go Home order for each axis. Can be set between 0 and 5, where the smaller numbers are executed first. The Go Home order for axes of motion that will not perform the Go Home action must be set to 255. **NOTE: These parameters correspond to the outlet axes (0~5) on the *wCardIndex* control card number, not the axes of motion in the group. For a detailed description, please refer to the "Go Home" section in the "EPCIO Series Motion Control Command Library User Manual."**
	wCardIndex	Motion control card number (0~ 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

2. int MCC_GetGoHomeStatus()

Description: After MCC_Home() has been called, use this command to check the Go Home status.

Return Value

0	Go Home has yet to finish
1	Go Home has finished
Other	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

3. int MCC_AbortGoHome()

Description: After MCC_Home() has been called, use this command to stop the Go Home action.

Return Value

0	Command successful
≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

4. int MCC_GetHomeSensorStatuss(

WORD* *pwStatus*,

WORD *wChannel*,

WORD *wCardInde* x

)

Description: Acquires the home sensor status. Defining the home sensor wiring (normal open or normal closed) is required prior to using this command. Wiring is defined in the Go Home parameters.

Parameters

pwStatus	Indicates a WORD value used to store home sensor status: 1 = Home sensor triggered 0 = No home sensor triggered
wChannel	Motion control card output channel (0~5)
wCardIndex	Motion control card number (0~11)

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

J. Position Control

1. int MCC_SetCompParam(

SYS_COMP_PARAM* ***pstCompParam***,
WORD ***wChannel***,
WORD ***wCardIndex***
)

Description	Sets the parameters for gear backlash and backlash compensation. The user can first set the compensation parameters, then input the parameters using this command, and finally call MCC_UpdatetCompParam(). Compensation parameters must cover the machine's entire course of work to avoid abnormal operations. For a detailed description, please refer to the section on "**Gear Backlash and Backlash Compensation"** in the "**EPCIO Series Motion Control Command Library User Manual.**"	
Parameters	*pstCompParam*	Indicates a SYS_COMP_PARAM structure, used to describe compensation parameters
	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

2. int MCC_UpdateCompParam()

Description	Responds to updated parameters for gear backlash and backlash compensation. This command is required to respond to the new settings after MCC_SetCompParam() has been called.	
Return Value	0	Command successful

	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

3. int MCC_SetPGain(

WORD ***wGain0***,
WORD ***wGain1***,
WORD ***wGain2***,
WORD ***wGain3***,
WORD ***wGain4***,
WORD ***wGain5***,
WORD ***wCardIndex***
)

Description	Sets proportional gain used in position closed loop control	
Parameters	*wGain0~wGain5*	Proportional gain used in each axis, set between 1 to 16256
	wCardIndex	Motion control card number (0~ 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

4. int MCC_GetPGain(

WORD* ***pwGain0***,
WORD* ***pwGain1***,
WORD* ***pwGain2***,
WORD* ***pwGain3***,
WORD* ***pwGain4***,
WORD* ***pwGain5***,
WORD ***wCardIndex***
)

Description	Acquires the proportional gain used in position closed circuit control

Parameters	*pwGain0~p wGain5*	Indicates a WORD value used to store the proportional gain used in each axis
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

5. int MCC_SetMaxPulseSpeed(

int *nPulse0*,
int *nPulse1*,
int *nPulse* 2,
int *nPulse3*,
int *nPulse4*,
int *nPulse5*,
WORD *wCardIndex*

)

Description	Sets the maximum pulse speed for each axis. Maximum pulse speed prevents the machine speed from exceeding operating parameters by limiting the number of pulses that each axis can send within one unit of interpolation time. For a detailed description, please refer to the section on "**Interpolation Time and Deceleration Time**" in the "**EPCIO Series Motion Control Command Library User Manual.**"	
Parameters	*nPulse0~nPulse5*	Maximum pulse speed for each axis. Set between 1 to 32767; the appropriate value is determined by considering machine properties and interpolation time.
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

6. int MCC_GetMaxPulseSpeed(

int* *pnPulse0*,

int* *pnPulse1*,

int* *pnPulse* 2,

int* *pnPulse3*,

int* *pnPulse4*,

int* *pnPulse5*,

WORD *wCardIndex*

)

Description: Acquires the maximum pulse speed for each axis

Parameters: *pnPulse0~pnPulse5* indicates a int value used to store the maximum pulse speed for each axis

wCardIndex Motion control card number (0~11)

Return Value: 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

7. int MCC_SetMaxPulseAcc(

int *nPulse0*,

int *nPulse1*,

int *nPulse* 2,

int *nPulse3*,

int *nPulse4*,

int *nPulse5*,

WORD *wCardIndex*

)

Description: Sets the maximum pulse acceleration for each axis. Maximum pulse acceleration prevents the machine acceleration (deceleration) from exceeding operating parameters by limiting the change in the number of pulses that each axis can send between any two continuous interpolation times. For a detailed description, please refer to the section on "**Interpolation Time and Deceleration Time**" in the "**EPCIO Series Motion Control Command Library User Manual.**"

Parameters *nPulse0~nPulse5* Maximum pulse acceleration for each axis.

Set between 1 to 32767; the appropriate value is determined by considering machine properties and interpolation time.

wCardIndex Motion control card number (0~11)

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

8. int MCC_GetMaxPulseAcc(

int* ***pnPulse0***,

int* ***pnPulse1***,

int* ***pnPulse* 2,**

int* ***pnPulse3***,

int* ***pnPulse4***,

int* ***pnPulse5***,

WORD ***wCardIndex***

)

Description Acquires the maximum pulse acceleration for each axis

Parameters *pnPulse0~pnPulse5* Indicates a int value used to store the maximum pulse acceleration for each axis

wCardIndex Motion control card number (0~11)

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

9. int MCC_SetInPosMode(

WORD ***wMode***,

WORD ***wGroupIndex***

)

Description	Sets the in position mode used. For a detailed description, please refer to the section on "**In Position**" in the "**EPCIO Series Motion Control Command Library User Manual.**"	
Parameters	*wMode*	In position mode
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

10. int MCC_SetInPosMaxCheckTime(

WORD ***wMaxCheckTimet***,

WORD ***wGroupIndex***

)

Description	Sets the in position maximum check time. For a detailed description, please refer to the section on "**In Position**" in the "**EPCIO Series Motion Control Command Library User Manual.**"	
Parameters	*wMaxCheckTime*	In position maximum check time, in ms
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

11. int MCC_ SetInPosSettleTime(

WORD ***wSettleTimet***,

WORD ***wGroupIndex***

)

Description	Sets the in position settle time. For a detailed description, please refer to the section on "**In Position**" in the "**EPCIO Series Motion Control Command Library User Manual.**"	
Parameters	*wSettleTIme*	In position settle time, in ms
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

12. int MCC_EnableInPos(

WORD *wGroupIndex*

)

Description	Enables the in position function. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*wGroupIndex*	Group number
Return Value	≥0	Command index given to this motion command in the motion control command library
	<0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

13. int MCC_DisableInPos(

WORD *wGroupIndex*

)

Description	Disables the in position function. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*wGroupIndex*	Group number
Return Value	≥0	Command index given to this motion command in the motion control command library
	<0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

14. int MCC_SetInPosToleranceEx(

double *dfTol0*,

double *dfTol1*,

double *dfTol2*,

double *dfTol3*,

double *dfTol4*,

double *dfTol5*,

WORD *wGroupIndex*

)

Description	Sets the extent of in position error tolerance	
Parameters	*dfTol0~dfTol5*	Extent of in position error tolerance in UU
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

15. int MCC_GetInPosToleranceEx(

double* *pdfTol0*,

double* *pdfTol1*,

double* *pdfTol2*,

double* *pdfTol3*,

double* *pdfTol4*,

double* *pdfTol5*,

WORD *wGroupIndex*

)

Description	Acquires the extent of in position error tolerance	
Parameters	*pdfTol0~pdfTol5*	Indicates a double value used to store extent of in position error tolerance in UU
	wGroupIndex	Group number
Return Value	0	Command successful

	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

16. int MCC_GetInPosStatus(

BYTE* ***pbyInPos0***,

BYTE* ***pbyInPos1***,

BYTE* ***pbyInPos2***,

BYTE* ***pbyInPos3***,

BYTE* ***pbyInPos4***,

BYTE* ***pbyInPos* 5,**

WORD ***wGroupIndex***

)

Description	Acquires the in position status for each axis.		
Parameters	*pbyInPos0~pbyInPos5*	Indicates a BYTE value used to store the in Position status for each axis:	
		0xFf(255)	In position requirements satisfied
		0	In position requirements not satisfied
	wGroupIndex	Group number	
Return Value	0	Command successful	
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**	

---

17. int MCC_EnableTrackError(

WORD ***wGroupIndex***,

DWORD ***dwAxisMask***

)

Description	Enables error tracking. For a detailed description, please refer to the section on “**Error Tracking**” in the “**EPCIO Series Motion Control Command Library User Manual.**”	
Parameters	*wGroupIndex*	Group number
	dwAxisMask	Axis perfoming the desired action

Possible parameters:

EPCIO_AXIS_X	X axis
EPCIO_AXIS_Y	Y axis
EPCIO_AXIS_Z	Z axis
EPCIO_AXIS_U	U axis
EPCIO_AXIS_V	V axis
EPCIO_AXIS_W	W axis
EPCIO_AXIS_ALL	All axes of motion

The above parameters can be combined. For example, X, Z, and V:

(EPCIO_AXIS_X | EPCIO_AXIS_Z | EPCIO_AXIS_V)

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

18. int MCC_DisableTrackError(

WORD wGroupIndex

)

Description	Disables error tracking check	
Parameters	*wGroupIndex*	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

19. int MCC_SetTrackErrorLimit(

double *dfLimit*,

char *cAxis*,

WORD *wGroupIndex*

)

Description	Sets the error tracking limits	
Parameters	*dfLimit*	Error tracking limits in UU

cAxis Number of axis of motion (0~5 correspond to axes X~W)

wGroupIndex Group number

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

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

Description Acquires the error tracking limits

Parameters *pdfLimit* Indicates a double value used to store the error tracking limits, in units of UU

cAxis Number of axis of motion (0~5 correspond to axes X~W)

wGroupIndex Group number

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

21. int MCC_SetPCLRoutine(
PCLISR ***pfnENCRoutine***,
WORD ***wCardIndex***
)

Description Serially connects the customized position control loop ISR. The system will automatically call this ISR when the position control loop fails. For a detailed description, please refer to the section on “**Position Control Loop Failure Treatment”** in “**EPCIO Series Motion Control Command Library User Manual.**”

Parameters	*pfnPCLRoutine*	Command index for customized position control loop ISR
	wCardIndex	Motion control card number (0~ 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

K. Advanced Trajectory Planning

---

1. int MCC_HoldMotion(
WORD *wGroupIndex*
)

Description	Pauses motion. This command must be used during motion to have any effect. After this command is called, the motion speed will decelerate to a stop. If the return value for MCC_GetMotionStatus() is GMS_RUNNING, the motion must completely stop before the return value GMS_HOLD can be obtained.	
Parameters	*wGroupIndex*	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

2. int MCC_ContiMotion(
WORD *wGroupIndex*
)

Description	Continues performing unfinished motion commands. This command must be used when the motion is paused to have any effect.	
Parameters	*wGroupIndex*	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

3. int MCC_AbortMotionEx(
double *dfDecTime*,
WORD *wGroupIndex*
)

Description	Decelerates within the set deceleration time to a stop and aborts all subsequent motion commands. After this command is called, if the return value for MCC_GetMotionStatus() is GMS_RUNNING, the motion must completely stop before the return value GMS_HOLD can be obtained. NOTE: ***After this command is used, the system must enter GMS_STOP status before subsequent motion commands can be achieved;*** otherwise the value will return ABORT_NOT_FINISH_ERR(-15).	
Parameters	*dfDecTime*	Deceleration time required
	wGroupIndex	Group number
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

4. int MCC_EnableBlend(

WORD *wGroupIndex*

)

Description	Enables path blending. Trajectory planned as on continuous path after calling this command. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*wGroupIndex*	Group number
Return Value	≥0	Command index given to this motion command in the motion control library
	<0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

5. int MCC_DisableBlend(

WORD *wGroupIndex*

)

Description	Disables path blending. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*wGroupIndex*	Group number

Return Value	≥0	Command index given to this motion command in the motion control command library
	<0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

6. int MCC_CheckBlend(

WORD *wGroupIndex*

)

Description	Checks if 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	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

7. int MCC_DelayMotion(

DWORD *dwTime*,

WORD *wGroupIndex*

)

Description	Sets the motion delay, delaying execution of the next motion command. Successfully calling this command will increase the number of stored motion commands.	
Parameters	*dwTime*	Time of delay in ms
	wGroupIndex	Group number
Return Value	≥0	Command index given to this motion command in the motion control command library
	<0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

8. int MCC_CheckDelay(

WORD *wGroupInde* x

)

Description	Checks motion delay status (at this point, GMS_DELAYING is the return value when MCC_GetMotionStatus() is called)	
Parameters	*wGroupIndex*	Group number
Return Value	0	Not in a status of motion delay
	1	In a status of motion delay
	Other	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

9. double MCC_OverrideSpeed(

double *dfRate*,

WORD *wGroupIndex*

)

Description	Sets the current general motion override speed rate	
Parameters	*wGroupIndex*	Group number
Return Value	> 0	Actual set override speed rate
	Other	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

10. double MCC_GetOverrideRate(

WORD *wGroupIndex*

)

Description	Acquires the current general motion override speed rate	
Parameters	*wGroupIndex*	Group number
Return Value	> 0	General motion override speed rate
	Other	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

---

11. double MCC_OverridePtPSpeed(double *dfRate*, WORD *wGroupIndex*)

Description	Sets the point-to-point override speed rate. This command is used to vary the speed of each axis.	
Parameters	dfRate	dfRate is the rate between the altered speed ratio and the original speed ratio multiplied by 100. In other words, the point-to-point altered speed ratio is equal to (original speed ratio x dfRate/100). dfRate is an integer greater than or equal to 1. The dfRate will automatically be set to 1 if it drops below 1. If the updated speed exceeds the MCC_SetSysMaxSpeed() settings, the new feed speed will simply equal these settings.
	wGroupIndex	Group number (WINDOWS: 0~71; DOS: 0~3)
Return Value	≥1	Actual override speed rate set
	Other	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

12. double MCC_GetPtPOverrideRate(WORD *wGroupIndex*)

Description	Acquires the current point-to-point override speed rate	
Parameters	*wGroupIndex*	Group number (WINDOWS: 0~71; DOS: 0~3)
Return Value	≥1	Current point-to-point override speed rate
	Other	Command failed; for the meaning of return values, please refer to **Section IV. Command Return Values**

L. Encoder Control

This section primarily describes the functions provided by, and the methods for use of, the encoder module in the EPCIO Series control card. Users should read this section in conjunction with the section on **"Encoder Control"** in the "**EPCIO Series Motion Control Command Library User Manual.**"

---

1. int MCC_SetENCRoutineEx(

ENCISR_EX *pfnENCRoutine*,

WORD *wCardIndex*

)

Description	Serially connects the customized encoder ISR	
Parameters	*pfnENCRoutine*	Command index for customized encoder ISR
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

2. int MCC_SetENCInputRate(

WORD *wInputRate*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets the encoder feedback rate. Calling this command has an effect on the feedback rate identical to that of the mechanism parameter *wInputRate*.	
Parameters	*wInputRate*	Encoder feedback rate; can be set as 1, 2, 4
	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

3. int MCC_ClearENCCounter(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Resets encoder counter to zero	
Parameters	*wChannel*	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

4. int MCC_GetENCValue(

long* *plValue*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Acquires encoder count	
Parameters	*plValue*	Indicates the long value used to store encoder count
	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

5. int MCC_SetENCLatchType(

WORD *wType*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets the latch encoder counter trigger method	
Parameters	*wType*	Latch encoder counter trigger method Possible settings: *ENC_TRIG_FIRST* When the first trigger condition is met, the count will be latched and will not change *ENC_TRIG_LAST* When the trigger conditions are met, the new count will be latched an unlimited number of times
	wChannel	Motion axis number (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

6. int MCC_SetENCLatchSource(

WORD *wSource*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description — Sets the latch encoder counter trigger signal source. 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 is input and the Channel 0 forward limit is triggered, the encoder count will be recorded in the latch register for Channel 0 in Card 0.

Parameters	*wSource*	Signal source; could be set as:
	ENC_TRIG_NO	No trigger signal source was 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_LIO0	Interrupt DI 0 in Local I/O
	ENC_TRIG_LIO1	Interrupt DI 1 in Local I/O
	ENC_TRIG_RDI0	Interrupt DI 0 in Remote I/O Set 0
	ENC_TRIG_RDI1	Interrupt DI 1 in Remote I/O Set 0
	ENC_TRIG_ADC0	Establish Channel 0 ADC comparative conditions
	ENC_TRIG_ADC1	Establish Channel 1 ADC comparative conditions
	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

7. int MCC_GetENCLatchValue(

long* *plValue*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Acquires the latch value recorded in register	
Parameters	*plValue*	Indicates a long value used to store a latch value(s) recorded in register
	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

8. int MCC_ EnableENCIndexTrigger (

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Enables the function triggering encoder ISR with encoder index signal	
Parameters	*wChannel*	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

9. int MCC_Dis ableENCIndexTrigger (

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Disables the function triggering encoder ISR with encoder index signal	
Parameters	*wChannel*	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

10. int MCC_GetENCIndexStatus(

WORD* *pwStatus*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Confirms whether the current position is located at the index signal input

Parameters	*pwStatus*	Indicates a WORD value used to store the index signal input status 1 Currently located at the index 0 Not located at the index
	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

11. int MCC_SetENCCompValue(

long *lValue*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets the comparative encoder value	
Parameters	*lValue*	Comparative encoder value
	wChannel	Motion control card output channel (0 - 5)
	wCardIndex	Motion control card number (0 - 11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

12. int MCC_EnableENCCompTrigger(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Enables the function triggering encoder ISR when the encoder count is equal to the comparative value	
Parameters	*wChannel*	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

13. int MCC_DisableENCCompTrigger(
WORD *wChannel*,
WORD *wCardIndex*
)

Description: Disables the function triggering encoder ISR when the encoder count is equal to the comparative value

Parameters	*wChannel*	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

M. Timer and Watchdog Control

This section primarily describes the functions provided by, and methods for use of, the timer and watchdog in the EPCIO Series motion control card. Users should read this section in conjunction with the section on **"Timer and Watchdog Control"** in the "**EPCIO Series Motion Control Command Library User Manual.**"

---

1. int MCC_SetTimer(

DWORD *dwValue*,

WORD *wCardIndex*

)

Description Sets the time cycle on the timer. The ISR for the customized Local I/O is triggered during each time cycle.

Parameters	*dwValue*	Time cycle in 25 ns; set between 1 to 16777215
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

2. int MCC_EnableTimer(

WORD *wCardIndex*

)

Description Enables timer

Parameters	*wCardIndex*	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

3. int MCC_DisableTimer(

WORD *wCardIndex*

)

Description	Disables timer	
Parameters	*wCardIndex*	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

4. int MCC_EnableTimerTrigger(

WORD *wCardIndex*

)

Description	Enables the function triggering the ISR for the customized Local I/O during each time cycle	
Parameters	*wCardIndex*	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

5. int MCC_DisableTimerTrigger(

WORD *wCardIndex*

)

Description	Disables the function triggering the ISR for the customized Local I/O during each time cycle	
Parameters	*wCardIndex*	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

6. int MCC_SetWatchDogTimer(

WORD *wValue*,

WORD *wCardInde* x

)

Description	Sets the watchdog timer. A hardware reset signal will be produced once the watchdog timer ends. If users do not want a reset signal, use MCC_RefreshWatchDogTimer() before the timer ends to reset the watchdog timer.	
Parameters	*dwValue*	Watchdog timer value. The units are in time cycles set by MCC_SetTimer(), ranging from 1 to 65535
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

7. int MCC_SetWatchDogResetPeriod(

WORD *wValue*,

WORD *wCardIndex*

)

Description	Sets the duration of the hardware reset signal generated once the watchdog timer ends	
Parameters	*wValue*	Reset period in 25 ns
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

8. int MCC_EnableWatchDogTimer(

WORD *wCardIndex*

)

Description	Enables the watchdog function	
Parameters	*wCardIndex*	Motion control card number (0~11)
Return Value	0	Command successful

	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

9. int MCC_DisableWatchDogTimer(
WORD *wCardIndex*
)

Description	Disables the watchdog function	
Parameters	*wCardIndex*	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

10. int MCC_RefreshWatchDogTimer(
WORD *wCardIndex*
)

Description	Resets the watchdog timer to avoid the generation of a hardware reset signal when the timer ends.	
Parameters	*wCardIndex*	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

N. Remote Input/Output Control

This section primarily describes the functions provided by, and methods for use of, the Remote I/O module in the EPCIO Series control card. Users should read this section in conjunction with the section on **"Remote Input/Output Control"** in the "**EPCIO Series Motion Control Command Library User Manual.**"

---

1. int MCC_SetRIORoutineEx(
RIOISR_EX *pfnRIORoutine*,
WORD *wCardIndex*
)

Description	Serially connects the customized Remote I/O ISR		
Parameters	*pfnRIORoutine*	Command index for customized Remote I/O ISR	
	wSet	Remote I/O set number	
		RIO_SET0	Remote I/O Set 0
		RIO_SET1	Remote I/O Set 1
	wCardIndex	Motion control card number (0~11)	
Return Value	0	Command successful	
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**	

---

2. int MCC_EnableRIOSetControl(
WORD *wSet*,
WORD *wCardIndex*
)

Description	Enables the indicated Remote I/O Set data transfer. The slave data transfer function for a given set can be enabled by calling MCC_EnableRIOSlaveControl().	
Parameters	*wSet*	Remote I/O Set number

RIO_SET0 Remote I/O Set 0

RIO_SET1 Remote I/O Set 1

wCardIndex Motion control card number (0~11)

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

3. int MCC_DisableRIOSetControl(
WORD *wSet*,
WORD *wCardIndex*
)

Description Disables the indicated Remote I/O Set data transfer. The slave data transfer function for a given set will also be disabled.

Parameters *wSet* Remote I/O Set number

RIO_SET0 Remote I/O Set 0

RIO_SET1 Remote I/O Set 1

wCardIndex Motion control card number (0~11)

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

4. int MCC_EnableRIOSlaveControl(
WORD *wSet*,
WORD *wCardIndex*
)

Description Enables the indicated Remote I/O Slave data transfer. Once the slave data transfer function is enabled, MCC_EnableRIOSetControl() is required to enable the set data transfer function, allowing the Remote I/O module to begin transmitting and receiving.

Parameters *wSet* Remote I/O Set number

RIO_SET0 Remote I/O Set 0

		RIO_SET1 Remote I/O Set 1
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

5. int MCC_DisableRIOSlaveControl(

WORD *wSet*,

WORD *wCardIndex*

)

Description	Disables the indicated Remote I/O Slave data transfer	
Parameters	*wSet*	Remote I/O Set number
		RIO_SET0 Remote I/O Set 0
		RIO_SET1 Remote I/O Set 1
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

6. int MCC_GetRIOTransStatus(

WORD* *pwStatus*,

WORD *wSet*,

WORD *wCardIndex*

)

Description: Acquires the current Remote I/O data transmission status. If transmission has stopped, it calls MCC_GetRIOMasterStatus() and MCC_GetRIOSlaveStatus() to distinguish between errors produced on the Master side or the Slave side.

Parameters	*pwStatus*	Indicates a WORD value used to store Remote I/O data transmission status

1 Remote I/O Set working normally

0 Remote I/O Set not working normally

wSet Remote I/O Set number

RIO_SET0 Remote I/O Set 0

RIO_SET1 Remote I/O Set 1

wCardIndex Motion control card number (0~11)

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

7. int MCC_GetRIOMasterStatus(

WORD* *pwStatus*,

WORD *wSet*,

WORD *wCardIndex*

)

Description Acquires the current status of Remote I/O Master data transmission to Slave

Parameters *pwStatus* Indicates a WORD value used to store Remote I/O data transmission status

1 Remote I/O Master terminal receiving signal normally

0 Remote I/O Master terminal not receiving signal normally

wSet Remote I/O Set number

RIO_SET0 Remote I/O Set 0

RIO_SET1 Remote I/O Set 1

wCardIndex Motion control card number (0~11)

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

8. int MCC_GetRIOSlaveStatus(

WORD* *pwStatus*,

WORD *wSet*,

WORD *wCardIndex*

)

Description	Acquires the current status for Remote I/O Slave reception of Master data		
Parameters	*pwStatus*	Indicates a WORD value used to store Remote I/O data transmission status	
		1	Remote I/O Slave terminal receiving signal normally
		0	Remote I/O Slave terminal not receiving signal normally
	wSet	Remote I/O Set number	
		RIO_SET0	Remote I/O Set 0
		RIO_SET1	Remote I/O Set 1
	wCardIndex	Motion control card number (0~11)	
Return Value	0	Command successful	
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**	

---

9. int MCC_GetRIOInputValue(

WORD* *pwValue*,

WORD *wSet*,

WORD *wPort*,

WORD *wCardIndex*

)

Description	Sets the indicated Set and Port 16-Bit Digital Input signal status value	
Parameters	*pwValue*	Indicates a WORD value used to store the 16 bit digital input signal status in the indicated position (Set, Port) (bit 0 to bit 15 represent the status for point 0 to point 15 in the Port)

wSet Remote I/O Set Number

RIO_SET0 Remote I/O Set 0

RIO_SET1 Remote I/O Set 1

wPort Digital Input Port No.

RIO_PORT0 Slave DI 0~DI 15

RIO_PORT1 Slave DI 16~DI 31

RIO_PORT2 Slave DI 32~DI 47

RIO_PORT3 Slave DI 48~DI 63

wCardIndex Motion control card number (0~11)

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

10. int MCC_SetRIOOutputValue(

WORD *wValue*,

WORD *wSet*,

WORD *wPort*,

WORD *wCardIndex*

)

Description Sets the indicated Set and Por t 16-bit digital output signal status value

Parameters *pwValue* Indicates a WORD value used to store the 16 bit digital output signal status in the indicated position (Set, Port) (bit 0 to bit 15 represent the status for point 0 to point 15 in the Port)

wSet Remote I/O Set Number

RIO_SET0 Remote I/O Set 0

RIO_SET1 Remote I/O Set 1

wPort Digital Input Port Number

RIO_PORT0 Slave DI 0~DI 15

RIO_PORT1 Slave DI 16~DI 31

RIO_PORT2 Slave DI 32~DI 47

RIO_PORT3 Slave DI 48~DI 63

	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

11. int MCC_EnquRIOOutputValue (

WORD *wValue*,

WORD *wSet*,

WORD *wPort*,

WORD *wCardIndex*

)

Description	Sets the indicated Set and Port 16-bit digital output signal status value	
Parameters	*pwValue*	Indicates a WORD value used to store the 16 bit digital output signal status in the indicated position (Set, Port) (bit 0 to bit 15 represent the status for point 0 to point 15 in the Port)
	wSet	Remote I/O Set Number
		RIO_SET0 Remote I/O Set 0
		RIO_SET1 Remote I/O Set 1
	wPort	Digital Input Port Number
		RIO_PORT0 Slave DI 0~DI 15
		RIO_PORT1 Slave DI 16~DI 31
		RIO_PORT2 Slave DI 32~DI 47
		RIO_PORT3 Slave DI 48~DI 63
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

NOTE: MCC_SetRIOOutputValue() is an immediate response, while MCC_EnquRIOOutputValue () will be entered in the motion command register.

12. int MCC_SetRIOTransError(

WORD *wTime*,

WORD *wSet*,

WORD *wCardIndex*

)

Description	Sets the maximum number of times Remote I/O transmission can retransmit. This setting is preset to 16. When data is unable to transmit correctly, the EPCIO Series motion control card will retransmit the data. If the data is still unable to be transmitted correctly when the set number of retransmissions is reached, a data transmission error will be produced (at which point MCC_GetRIOTransStatus() can be used to obtain the abnormal results of data transmission).		
Parameters	*wTime*	Number of times erroneous data is retransmitted (0~16)	
	wSet	Remote I/O Set number	
		RIO_SET0	Remote I/O Set 0
		RIO_SET1	Remote I/O Set 1
	wCardIndex	Motion control card number (0~11)	
Return Value	0	Command successful	
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**	

13. int MCC_SetRIOTriggerType (

WORD *wType*,

WORD *wSet*,

WORD *wDigitalInput*,

WORD *wCardIndex*

)

Description

Sets the method for triggering ISR with the Remote I/O Digital Input signal as “rising edge,” “falling edge,” or “level change.” The first four digital inputs in each Slave group (RIO_DI0, RIO_DI1, RIO_DI2, and RIO_DI3) can trigger the customized ISR. MCC_EnableRIOInputTrigger() is required to enable the intterupt function after this command has been set.

Parameters	wType	Remote I/O digital input signal interruption triggering method	
		RIO_INT_RISE	Rising Edge Trigger
		RIO_INT_FALL	Falling Edge Trigger
		RIO_INT_LEVEL	Level Change Trigger
	wSet	Remote I/O Set Number	
		RIO_SET0	Remote I/O Set 0
		RIO_SET1	Remote I/O Set 1
	wDigitalInput	Slave Digital Input Number	
		RIO_DI0	Remote I/O Slave Input 0
		RIO_DI1	Remote I/O Slave Input 1
		RIO_DI2	Remote I/O Slave Input 2
		RIO_DI3	Remote I/O Slave Input 3
	wCardIndex	Motion control card number (0~11)	
Return Value	0	Command successful	
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**	

---

14. int MCC_EnableRIOInputTrigger(

WORD *wSet*,

WORD *wDigitalInput*,

WORD *wCardIndex*

)

Description

The first four digital input points in each Slave group (RIO_DI0, RIO_DI1, RIO_DI2, and RIO_DI3) can trigger the customized ISR. This command enables the RIO_DI0 to RIO_DI3 interrupt function.

Parameters	*wSet*	Remote I/O Set Number	
		RIO_SET0	Remote I/O Set 0
		RIO_SET1	Remote I/O Set 1
	wDigitalInput	Slave Digital Input Number	

		RIO_DI0	Remote I/O Slave Input 0
		RIO_DI1	Remote I/O Slave Input 1
		RIO_DI2	Remote I/O Slave Input 2
		RIO_DI3	Remote I/O Slave Input 3
	wCardIndex	Motion control card number (0~11)	
Return Value	0	Command successful	
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**	

---

15. int MCC_DisableRIOInputTrigger(

WORD *wSet*,

WORD *wDigitalInput*,

WORD *wCardIndex*

)

Description: The first four digital inputs in each Slave group (RIO_DI0, RIO_DI1, RIO_DI2, and RIO_DI3) can trigger the user-customized ISR. This command disables the RIO_DI0 to RIO_DI3 interrupt function.

Parameters	*wSet*	Remote I/O Set Number	
		RIO_SET0	Remote I/O Set 0
		RIO_SET1	Remote I/O Set 1
	wDigitalInput	Slave Digital Input Number	
		RIO_DI0	Remote I/O Slave Input 0
		RIO_DI1	Remote I/O Slave Input 1
		RIO_DI2	Remote I/O Slave Input 2
		RIO_DI3	Remote I/O Slave Input 3
	wCardIndex	Motion control card number (0~11)	
Return Value	0	Command successful	
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**	

---

16. int MCC_EnableRIOTransTrigger(

WORD *wSet*,

WORD *wCardIndex*

)

Description	Enables the "Transmission Error" interrupt function of the Remote I/O		
Parameters	*wSet*	Remote I/O Set Number	
		RIO_SET0	Remote I/O Set 0
		RIO_SET1	Remote I/O Set 1
	wCardIndex	Motion control card number (0~11)	
Return Value	0	Command successful	
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**	

17. int MCC_DisableRIOTransTrigger(

WORD *wSet*,

WORD *wCardIndex*

)

Description	Disables the "Transmission Error" interrupt function of the Remote I/O		
Parameters	*wSet*	Remote I/O Set Number	
		RIO_SET0	Remote I/O Set 0
		RIO_SET1	Remote I/O Set 1
	wCardIndex	Motion control card number (0~11)	
Return Value	0	Command successful	
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**	

O. Digital to Analog Converter Control

This section primarily describes the functions provided by, and methods for use of, the DAC module in the EPCIO Series motion control card. Users should read this section in conjunction with the section on **"Analog Voltage Output Control"** in the "**EPCIO Series Motion Control Command Library User Manual.**"

---

1. int MCC_SetDACOutput(

Float *fVoltage*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Outputs indicated voltage	
Parameters	*fVoltage*	Voltage output (-10 V to 10 V)
	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

2. int MCC_SetDACTriggerOutput(

float *fVoltage*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	When the output axes (0~5) in the motion control card are not using voltage command operation mode (in other words, mechanism parameter *wCommandMode* is set at OCM_PULSE), the voltage in the DAC mode can be preprogrammed in the DAC module. When trigger conditions are met, the hardware can immediately output this pre-programmed voltage.	
Parameters	*fVoltage*	Preprogrammed voltage output (-10 V to 10 V)

	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

3. int **MCC_SetDACTriggerSource(**

DWORD *dwSource*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description

This command can be used when the axes of motion are not using the voltage command operation mode to set the source to trigger the preprogrammed voltage output. Various sources can be set for each DAC channel. MCC_EnableDACTriggerMode() is required to enable the trigger mode after the command has been set.

Parameters *dwSource* DAC trigger source, could be:

DAC_TRIG_ENC0	Specific count for encoder Channel 0
DAC_TRIG_ENC1	Specific count for encoder Channel 1
DAC_TRIG_ENC2	Specific count for encoder Channel 2
DAC_TRIG_ENC3	Specific count for encoder Channel 3
DAC_TRIG_ENC4	Specific count for encoder Channel 4
DAC_TRIG_ENC5	Specific count for encoder Channel 5
DAC_TRIG_ADC0	Specific voltage for ADC 0
DAC_TRIG_ADC1	Specific voltage for ADC 1
DAC_TRIG_ADC2	Specific voltage for ADC 2
DAC_TRIG_ADC3	Specific voltage for ADC 3
DAC_TRIG_ADC4	Specific voltage for ADC 4
DAC_TRIG_ADC5	Specific voltage for ADC 5
DAC_TRIG_ADC6	Specific voltage for ADC 6
DAC_TRIG_ADC7	Specific voltage for ADC 7
DAC_TRIG_LDI0	Channel 0 Limit + Signal Input

	DAC_TRIG_LDI1	Channel 1 Limit + Signal Input
	DAC_TRIG_LDI2	Channel 2 Limit + Signal Input
	DAC_TRIG_LDI3	Channel 3 Limit + Signal Input
	DAC_TRIG_R0DI0	Remote Set 0 DI0 Signal Input
	DAC_TRIG_R0DI1	Remote Set 0 DI1 Signal Input
	DAC_TRIG_R0DI2	Remote Set 0 DI2 Signal Input
	DAC_TRIG_R0DI3	Remote Set 0 DI3 Signal Input
	DAC_TRIG_R1DI0	Remote Set 1 DI0 Signal Input
	DAC_TRIG_R1DI1	Remote Set 1 DI1 Signal Input
	DAC_TRIG_R1DI2	Remote Set 1 DI2 Signal Input
	DAC_TRIG_R1DI3	Remote Set 1 DI3 Signal Input
	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~1 1)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

4. int MCC_EnableDACTriggerMode(

WORD *wChannel*,

WORD *wCardIndex*

)

Description: This command can be used when the axes of motion are not using the voltage command operation mode to enable the function triggering the preprogrammed voltage output. Before enabling the trigger mode, please set the trigger source.

Parameters	*wChannel*	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~1 1)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

5. int MCC_DisableDACTriggerMode(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	This command can be used when the axes of motion are not using the voltage command operation mode to disable the function triggering the preprogrammed voltage output.	
Parameters	*wChannel*	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

6. int MCC_StartDACConv(

WORD *wCardIndex*

)

Description	This command enables voltage output when no output channels in the motion control cards indicated by *wCardIndex* use the voltage command operation mode.	
	wChannel	Motion control card output channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

7. int MCC_StopDACConv(WORD *wCardIndex*)

)

Description	This command disables voltage output when no output channels in the control cards indicated by *wCardIndex* use the voltage command operation mode.	
	wChannel	Motion control card output channel (0~5)

	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

P. Analog to Digital Converter Control

This section primarily describes the functions provided by, and methods for use of, the ADC module in the EPCIO Series control card. Users should read this section in conjunction with the section on **"Analog Voltage Input Control"** in the "**EPCIO Series Motion Control Command Library User Manual.**"

---

1. int MCC_SetADCRoutine(

ADCISR *pfnADCRoutine*,

WORD *wCardIndex*

)

Description	Serially connects customized ADC ISR	
Parameters	*pfnADCRoutine*	Command index for customized ADC interrupt service routine
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

2. int MCC_SetADCConvType(

WORD *wConvType*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets the ADC voltage conversion type as unipolar or bipolar		
Parameters	*wConvType*	Type of voltage conversion	
		ADC_TYPE_BIP	Bipolar Converter Type
		ADC_TYPE_UNI	Unipolar Converter Type
	wChannel	A/D converter channel (0~5)	
	wCardIndex	Motion control card number (0~11)	

Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

3. int MCC_GetADCConvType(

WORD* *pwConvType*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Acquires ADC type	
Parameters	*p wConvType*	Indicates a WORD value used to store voltage conversion type. Possible values are: ADC_TYPE_BIP Bipolar Converter Type ADC_TYPE_UNI Unipolar Converter Type
	wChannel	A/D converter channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

4. int MCC_SetADCConvMode(

WORD *wConvMode*,

WORD *wCardIndex*

)

Description	Sets the ADC voltage conversion mode to Single or Free Running Mode	
Parameters	*wConvMode*	Voltage conversion mode; this model provides the following settings: ADC_MODE_SINGLE Single Conversion ADC_MODE_FREE Free Running Conversion

wCardIndex Motion control card number (0~11)

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

5. int MCC_GetADCInput(

float* ***pfInput***,

WORD ***wChannel***,

WORD ***wCardIndex***

)

Description Acquires the DC voltage input for the indicated ADC channel. If the ADC is indicated as “unipolar,” EPCIO-400 and EPCIO-601 rms voltage input is 0 to 20 V, and EPCIO-4000 and EPCIO-6000 rms voltage input is 0 to 10 V. If the ADC is set to “bipolar,” EPCIO-400 and EPCIO-601 rms voltage input is -10 V to +10 V, and EPCIO-4000 and EPCIO-6000 rms voltage input is -5 to +5 V.

Parameters *pfInput* Indicates a float value used to store ADC channel DC input

wChannel A/D converter channel (0~5)

wCardIndex Motion control card number (0~11)

Return Value 0 Command successful

≠0 Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

6. int MCC_SetADCSingleChannel(

WORD ***wChannel***,

WORD ***wCardIndex***

)

Description Sets an indicated ADC channel as “Single Channel.” Combined with using MCC_SetADCConvMode() to set the conversion mode to single mode, when MCC_StartADCConv() is called, the channel selected will directly convert electricity once. Conversion will not occur again once it is finished; the user must call MCC_StartADCConv() again for another conversion. While converting, MCC_GetADCWorkStatus() can be used to check the conversion progress.

Parameter	*wChannel*	A/D converter channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

7. int MCC_GetADCWorkStatus(

WORD* ***pwStatus***,

WORD ***wCardIndex***

)

Description	Acquires current ADC work status	
Parameters	*pwStatus*	Indicates WORD value used to store ADC work status 1 Converting 0 Not converting
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

8. int MCC_EnableADCConvTrigger(

WORD ***wCardIndex***

)

Description	Enables the function triggering the customized ADC ISR when any channel voltage conversion is complete	
Parameters	*wCardIndex*	Motion control card number (0~11)
Return Value	0	Command successful

	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

9. int MCC_DisableADCConvTrigger(

WORD *wCardIndex*

)

Description	Disables the function triggering the customized ADC ISR when any channel voltage conversion is complete	
Parameters	*wCardIndex*	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

10. int MCC_SetADCTagChannel(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets an indicated ADC as the tag channel. Used in combination with MCC_EnableADCTagTrigger(), the customized ISR will be triggered when the tag channel voltage completes conversion.	
Parameter	*wChannel*	A/D converter channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

11. int MCC_EnableADCTagTrigger(

WORD *wCardIndex*

)

Description	Enables the function triggering the customized ISR when the tag channel voltage completes conversion.	
Parameter	*wChannel*	A/D converter channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

12. int MCC_DisableADCTagTrigger(WORD *wCardIndex*)

)

Description	Disables the function triggering the customized ISR when the tag channel voltage completes conversion.	
Parameter	*wChannel*	A/D converter channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

13. int MCC_SetADCCompMask(

WORD *wMask*,

WORD *wCardIndex*

)

Description When the voltage input is compared to the set comparative value, the smallest few bits can be masked for comparison, reducing the sensitivities of the comparator to prevent interruptions due to input voltage vibrations

Parameters	*wMask*	Voltage mask bit, flag bit can be set as:	
		ADC_MASK_NO	No voltage mask bit
		ADC_MASK_BIT1	1 voltage mask bit

		ADC_MASK_BIT2 2 voltage mask bit
		ADC_MASK_BIT3 3 voltage mask bit
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

14. int MCC_SetADCCompType(

WORD *wCompType*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description

Sets the ADC voltage comparative type. Calling MCC_EnableADCCompTrigger() after this command produces an ADC hardware interruption signal when the comparative conditions are established. In addition to triggering customized ISR, this signal also triggers the DAC module preprogrammed voltage output. The first two groups of the ADC channel trigger signal can be used simultaneously to latch the encoder count.

Parameters	*wCompType*	Voltage comparative type; possible settings:
	ADC_COMP_RISE	Input voltage is compared from least to greatest
	ADC_COMP_FALL	Input voltage is compared from greatest to least
	ADC_COMP_LEVEL	Input voltage is changed and compared
	wChannel	ADC channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

15. int MCC_SetADCCompValue(

float *fValue*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Sets the ADC comparative channel voltage output value in bipolar mode. This command does not supply a comparative function for unipolar mode. MCC_Set ADCCompType() and MCC_EnableADCCompTrigger() must be used after this command has been set to produce an ADC hardware interruption signal when this ADC channel voltage input meets the conditions for comparison.	
Parameters	*fValue*	Voltage input comparative value (EPCIO-400/601 control card can be set between -10 V to 10 V, and the EPCIO-4000/6000 motion control card can be set between -5 V to 5 V)
	wChannel	ADC channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

16. int MCC_GetADCCompValue(

float * *pfValue*,

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Acquires voltage comparative value used	
Parameters	*pfValue*	Indicates a float value used to store the voltage input comparative value
	wChannel	A/D converter channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful

	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

17. int MCC_EnableADCCompTrigger(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Enables the function triggering the customized ISR when voltage comparative conditions are met	
Parameters	*wChannel*	A/D converter channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

18. int MCC_DisableADCCompTrigger(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Disables the function triggering the customized ISR when voltage comparative conditions are met	
Parameters	*wChannel*	ADC channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

19. int MCC_EnableADCConvChannel(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Enables the ADC of voltage input in the ADC channel. The conversion channel set in this command must use the free run mode. MCC_StartADCConvChannel must be called after the channel is set to initiate ADC conversion.	
Parameters	*wChannel*	ADC channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

20. int MCC_DisableADCConvChannel(

WORD *wChannel*,

WORD *wCardIndex*

)

Description	Disables the ADC of voltage input in the ADC channel	
Parameters	*wChannel*	A/D converter channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

---

21. int MCC_StartADCConv(

WORD *wCardIndex*

)

Description	Starts ADC channel analog voltage conversion. This command must be combined with MCC_EnableADCConvChannel().	
Parameters	*wChannel*	A/D converter channel (0~5)

	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

22. int MCC_StopADCConv(

WORD *wCardIndex*

)

Description	Stops all ADC channel analog voltage conversion.	
Parameters	*wChannel*	A/D converter channel (0~5)
	wCardIndex	Motion control card number (0~11)
Return Value	0	Command successful
	≠0	Command failed; for the meaning of return values, please consult **Section IV. Command Return Values**

III. Error Codes

Error Code	Description
0xF101	Motion control command library has yet to be initialized
0xF104	Given parameters during curve motion commands are unreasonable
0xF203	Feed rate speed exceeds the allowed pulse output for each interpolation time
0xF204	Feed acceleration exceeds the allowed pulse output increment for each interpolation time
0xF301	X axis position exceeds the work limits set by mechanism parameters
0xF302	Y axis position exceeds the work limits set by mechanism parameters
0xF303	Z axis position exceeds the work limits set by mechanism parameters
0xF304	U axis position exceeds the work limits set by mechanism parameters
0xF305	V axis position exceeds the work limits set by mechanism parameters
0xF306	W axis position exceeds the work limits set by mechanism parameters
0xF401	An error occurred during motion command execution
0xF501	In position 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
0xF801	X error tracking has exceeded the set tolerance range
0xF802	Y error tracking has exceeded the set tolerance range
0xF803	Z error tracking has exceeded the set tolerance range
0xF804	U error tracking has exceeded the set tolerance range
0xF805	V error tracking has exceeded the set tolerance range
0xF806	W error tracking has exceeded the set tolerance range

IV. Command Return Values

Return Value Definition	Number Value	Description
NO_ERR	0	Command successful
INITIAL_MOTION_ERR	-1	System has yet to be initialized; please recall MCC_InitSystem()
COMMAND_BUFFER_FULL_ERR	-2	Motion command queue is full; unable to receive this command at this time
COMMAND_NOTACCEPTED_ERR	-3	System is busy; unable to receive this command at this time
COMMAND_NOTFINISHED_ERR	-4	Motion command execution unfinished; unable to receive this command at this time
PARAMETER_ERR	-5	Incoming parameter format error when command was called
GROUP_PARAMETER_ERR	-6	Error given by group parameters; invalid group indicated
FEED_RATE_ERR	-7	Feed rate speed not set or set incorrectly; please recall MCC_SetFeedSpeed()
VOLTAGE_COMMAND_NOTCALLED_ERR	-9	Use of this command is inhibited because either the system or this axis of motion are using the V Command operation mode
HOME_COMMAND_NOTCALLED_ERR	-10	Currently not at 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 from calling MCC_GetErrorCode() to understand the content of the error
ABORT_NOT_FINISH_ERR	-15	Command abortion not finished
GROUP_RAN_OUT_ERR	-16	No groups remaining for use

V. Motion Control Command Library Default Settings

The following table lists the default settings for the motion control command library after MCC_InitSystem() has been called. If these default settings are unable to satisfy user needs, the related commands can be called to alter the settings.

Setting Content	Default Setting	Related Commands

Command queue size	10000 commands	MCC_SetCmdQueueSize()
		MCC_GetCmdQueueSize()

Dry run function	Disabled	MCC_EnableDryRun()
		MCC_DisableDryRun()

Maximum feed rate 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 loop	64	MCC_SetPGain()
		MCC_GetPGain()

Acceleration and deceleration type for each axis during line, curve, circular, and helix motions	S curve	MCC_SetAccType()
		MCC_GetAccType()
		MCC_SetDecType()
		MCC_GetDecType()

Acceleration and deceleration time for each axis during line, curve, circular, and helix motions	300 ms	MCC_SetAccTime()
		MCC_GetAccTime()
		MCC_SetDecTime()
		MCC_GetDecTime()

Feed rate speed used for line, 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 maximum check time	1000 ms	MCC_SetInPosMaxCheckTime()
In position settling time	100 ms	MCC_SetInPosSettleTime()

In position error tolerance value	∞	MCC_SetInPosToleranceEx()
		MCC_GetInPosToleranceEx()

In position function	Disabled	MCC_EnableInPos()
		MCC_DiableInPos()

Path blend function	Disabled	MCC_EnableBlend()
		MCC_DisnableBlend()

Track error function	Disabled	MCC_EnableTrackError()
		MCC_DisnableTrackError()

Track error permissible limit value	∞	MCC_SetTrackErrorLimit()
		MCC_GetTrackErrorLimit()

VI. Changes to Motion Control Command Library

This section lists the differences between Motion Control Command Library V5.0 (or higher) and the previous version. First time users can skip over this section entirely. Users who initially used earlier versions of Motion Control Command Library should read the descriptions in this section carefully.

A. Removed Commands

Command Name	Reason and Approach
MCC_RedefineCoord	This version of the motion control command library does not allow different axes of motion to correspond to the same actual channel; therefore this command is unnecessary
MCC_SetInterpolateTime	Interpolation time should not be actively modified while the system is operating normally. To maintain system stability, this version of the motion control command library no longer supports this command. If the user needs to actively change the interpolation time, please execute MCC_CloseSystem(), then recall MCC_InitSystem()
MCC_GetInterpolateTime	Interpolation time is a parameter in MCC_InitSystem(). If the user needs this value while programming other areas, the user should save the value manually
MCC_GetErrorCount	While the error count is necessary for internal use, it is meaningless to the user. Therefore, this version of the motion control command library no longer supports this command
MCC_GetInPosStableTime	Three new types of the in position modes (for a total of four types) were added to this version; therefore this motion control command library no longer supports this command
MCC_ChangeFeedSpeed	This command is similar to the function of MCC_OverrideSpeed(). To avoid user confusion, this version of the motion control command library no longer supports this command. Instead, please use MCC_OverrideSpeed()
MCC_ChangePtPSpeed	This command is similar to the function of MCC_OverrideSpeed(). To avoid user confusion, this version of the motion control command library no

	longer supports this command. Instead, please use MCC_OverrideSpeed()
MCC_SetCycleInterruptRoutine	To prevent customized commands from interfering with the internal interpolation time, this version of the motion control command library no longer supports this command; please use the timer function provided

MCC_SetAccStep
MCC_GetAccStep
MCC_SetDecStep
MCC_GetDecStep
	Please refer to MCC_SetAccTime()
	MCC_GetAccTime()
	MCC_SetDecTime()
	MCC_GetDecTime()

MCC_SetPtPAccStep
MCC_GetPtPAccStep
MCC_SetPtPDecStep
MCC_GetPtPDecStep
	Please refer to MCC_SetAccTime()
	MCC_GetAccTime()
	MCC_SetDecTime()
	MCC_GetDecTime()

MCC_SetGoHomeAccTime MCC_GetGoHomeAccTime MCC_SetGoHomeDecTime MCC_GetGoHomeDecTime MCC_SetGoHomeAccStep MCC_GetGoHomeAccStep MCC_SetGoHomeDecStep MCC_GetGoHomeDecStep MCC_SetLeaveHomeSensorSpeed	Please refer to MCC_SetHomeConfig()

B. Obsolete Commands

The following is a list of obsolete commands in this version of the motion control command library, and exists only to be compatible with earlier versions. Though these commands can still be used normally in this version, users should try to avoid them as they may be removed in future versions.

Command Name	Replacement Command(s)

MCC_SetGroupConfig	MCC_CreateGroup()
	MCC_CloseGroup()
	MCC_CloseAllGroups()

MCC_SetInPosCheckTime	MCC_SetInPosMaxCheckTime()

MCC_SetInPosTolerance
MCC_GetInPosTolerance
	MCC_SetInPosToleranceEx()
	MCC_GetInPosToleranceEx()

MCC_AbortMotion	MCC_AbortMotionEx()
MCC_SetDACClockDivider MCC_SetADCClockDivider MCC_SetRIOClockDivider	N/A (unnecessary)

MCC_SetMachParam
MCC_GetMachParam
MCC_UpdateMachParam
	MCC_SetMacParam()
	MCC_GetMacParam()
	MCC_SetEncoderConfig()
	MCC_SetHomeConfig()
	MCC_UpdateParam()

MCC_GoHome	MCC_SetHomeConfig()
	MCC_Home()

MCC_LineX MCC_LineY MCC_LineZ MCC_LineU MCC_LineV MCC_LineW	MCC_Line()
MCC_PtPX MCC_PtPY MCC_PtPZ MCC_PtPU MCC_PtPV MCC_PtPW	MCC_PtP()

C. Commands with Actions that Differ from those of Earlier Versions

Command Name	Difference in Action
MCC_EnableLimitSwitchCheck	1. In earlier versions, a triggered limit switch would only stop output commands, but would not produce an error record (Jog commands could be used immediately to retreat from the limit switch region). The user had to call MCC_GetLimitSwitchStatus() to know whether this error had occurred. In this new version, an error record is produced. As long as the user calls MCC_GetErrorCode(), he or she will know whether this error occurred. MCC_ClearError() must first be called to remove the error before the Jog method can be

	used to retreat from the limit switch. 2. In this new version, if this command is called successfully, an error record will be recorded and the motion stopped only if the triggered limit switch is aligned with the direction of the motion, regardless of input parameters.
MCC_Home MCC_GoHome	In earlier versions, a result similar to MCC_ResetMotion() would occur once the Go Home action was finished, resetting the system to a default status. In this new version, it only resets the axis of motion executing the Go Home action.
MCC_AbortGoHome	In earlier versions, calling this command would not only stop the Go Home action, but would produce a result similar to MCC_ResetMotion(). In this new version, it simply stop Go Home.
MCC_DelayMotion	Timing unit changed from interpolation time to ms
MCC_AbortMotionEx	Please refer to the description of this command