PCI Bus/PCI Express
Motor Control Board

MC8000P Device Driver User's Manual

2018-01-30 Ver. 8.0.0.0

Supported board	
MC8000P series	MC8043P/ MC8043Pe
	MC8082P/ MC8082Pe
	MC8022P
	MC8042P
MC8500P series	MC8541P/ MC8541Pe
	MC8581P/ MC8581Pe

NOVA electronics

Introduction

■ Before You Begin

Before using MC8000P and MC8500P, please read this manual carefully to fully understand for correct use and observe all the instructions given in this manual. We shall be exempted from taking responsibility and held harmless for damage or losses incurred by the user if the user fails to observe the instructions.

Information in this manual is subject to change without notice.
You can download the latest manual and software from our web site: http://www.novaelec.co.jp/eng

■ Board Manual

Regarding the board specifications, please refer to the User's Manual of each board.

■ Terms in this Manual

IC	IC indicates IC chips such as MCX314As, MCX304 or MCX514.
IC-A, IC-B	When a board has multiple IC chips on it, the first IC is described as "IC-A", and the second IC is described as "IC-B". When a board has only one IC chip on it, the IC is described as "IC-A".

1. Outline

This device driver is a common device driver for the PCI-bus/PCI Express motor control board of NOVA electronics.
Regarding the specifications of the board, please refer to the user's manuals of each board and IC.

1.1 Boards

This device driver supports the following boards. Multiple boards can be installed into a single PC.

Supported Boards	
MC8000P series	MC8043P / MC8043Pe
	MC8082P / MC8082Pe
	MC8022P
	MC8042P
MC8500P series	MC8541P/MC8541Pe
	MC8581P/MC8581Pe

1.2 Operating Systems

This device driver supports the following OS.

Supported OS
Windows7(32bit,64bit)
Windows8.1(32bit,64bit)
Windows10(32bit,64bit)

1.3 Languages

This device driver supports the following languages.
The user can control each board from an application by calling DLL provided by NOVA electronics.

Supported Languages
Microsoft Visual C++
Microsoft Visual Basic
Microsoft Visual C#

Note: Supported version of Microsoft Visual Studio and the latest information about development environment and OS for each language, see the Microsoft website and our website.

1.4 Maximum Number of Boards

This device driver can recognize the boards up to 16 simultaneously.
When using multiple boards in a single PC, the device driver also can recognize the boards up to 16 simultaneously.

1.5 IC on the Board

Each board is equipped with 1 or more ICs, as shown in the table below.

Board	IC	Number of IC	Number of Axis
MC8043P / MC8043Pe	MCX314As	1	4
MC8082P / MC8082Pe	MCX304	2	8
MC8022P	MCX304	1	2
MC8042P	MCX304	1	4
MC8541P / MC8541Pe	MCX514	1	4
MC8581P / MC8581Pe	MCX514	2	8

2. Installation

This chapter describes how to install the board into your PC and install the device driver.

2.1 Preparation of Driver Software

When installing the driver from CD-ROM, prepare MC8000P device driver CD-ROM.
When installing the driver from the downloaded file from our homepage, extract the file.

2.2 Settings When Using Multiple Boards

This device driver can recognize the boards up to 16 simultaneously.

When using multiple boards on a system (PC), in order to individually recognize each board, set the board number of second or later board by the rotary switch on the board. For the location of the rotary switch (SW1), see chapter “Board Dimensions” in the user’s manual of each board.
The rotary switch can be set from 0 to F. Make sure that there is no duplicate number among multiple boards and the different boards.
The rotary switch of factory default is set to 0.

2.3 Installation of the Board into your PC

Note: **Make sure the PC’s power is shut off before installing the board.** Otherwise, the circuit elements may be damaged.

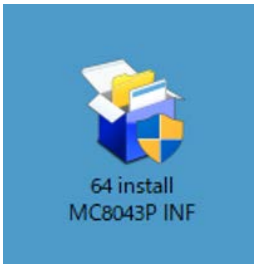
(1) Make sure that the PC is powered OFF, and then remove the external cover and slot cover.
(2) Insert the board into an empty expansion slot. Be sure that the board’s edge connector fits into the PC’s PCI bus/PCI Express connector.
(3) Screw the mounting bracket. Make sure that you fix the screws appropriately; otherwise, short out, breakdown or operation error may result.
(4) Replace the external cover.

2.4 Installation of Device Driver

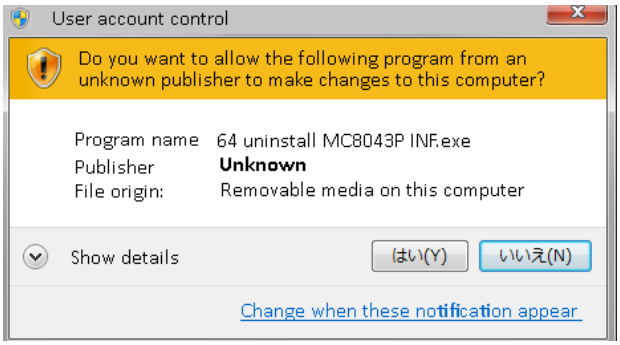
This chapter describes of how to install device driver.

Make sure to close the other application.
Before starting the installation procedure, ensure that you are logged on to Windows with a user name having administrator authority. Otherwise, the installation is not successfully completed.
There is no difference for the procedure of the installation for each Windows OS.

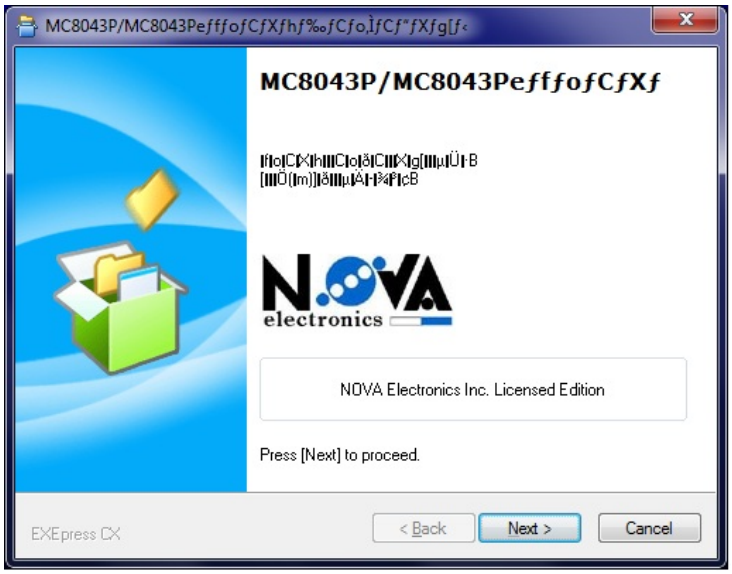
(1) Prepare the device driver by chapter 2.1.
(2) Make sure that the board is seated properly in the PC by chapter 2.2 and 2.3.
(3) Turn on the PC and start Windows 7
(4) Log on to Windows with a user name having administrator authority.
(5) Select Driver folder in the attached CD-ROM (When the attached CD-ROM in D drive, D:\Driver) or the downloaded software, double click [64 install XXXX INF.exe] in 64Driver folder. (For 32bit, double click [32 install XXXX INF.exe] in 32driver folder.)
(XXXX indicates the model name.)



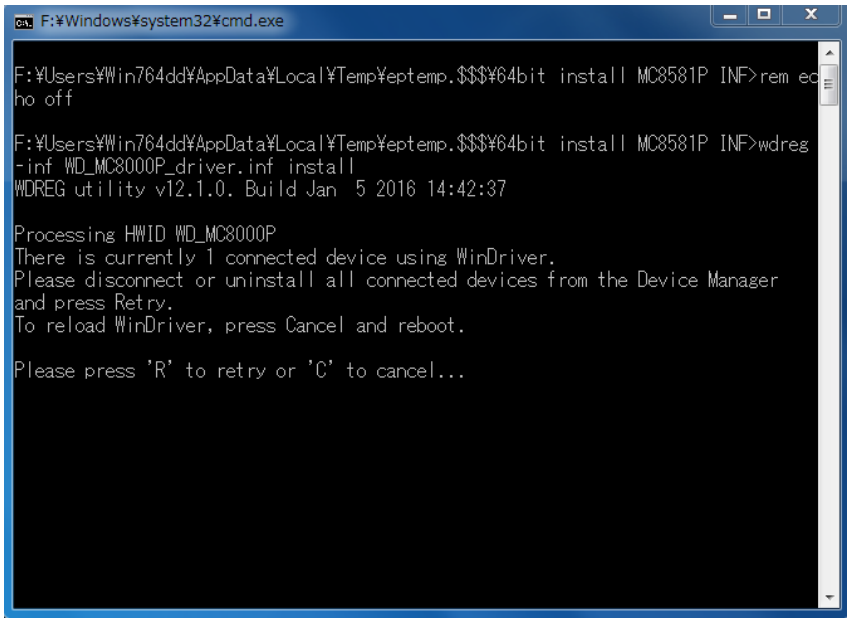
(6) When the following wizard appears. Click “Yes”. (Read by replacing MC8043P in the dialog box with your model name.)



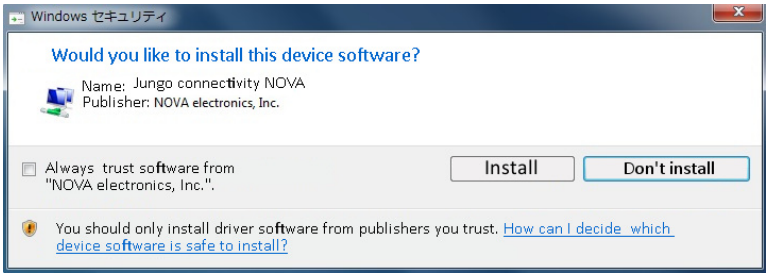
(7) Device driver installation window appears. Click Next. (Read by replacing MC8043P in the dialog box with your model name.)

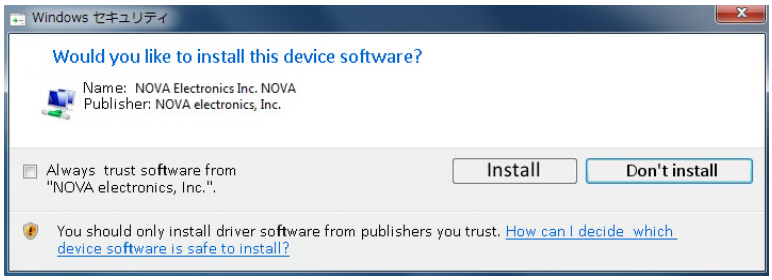


(8) If the other board has already been installed, the following window appears. Enter [C] (This window does not appear for the first installation.

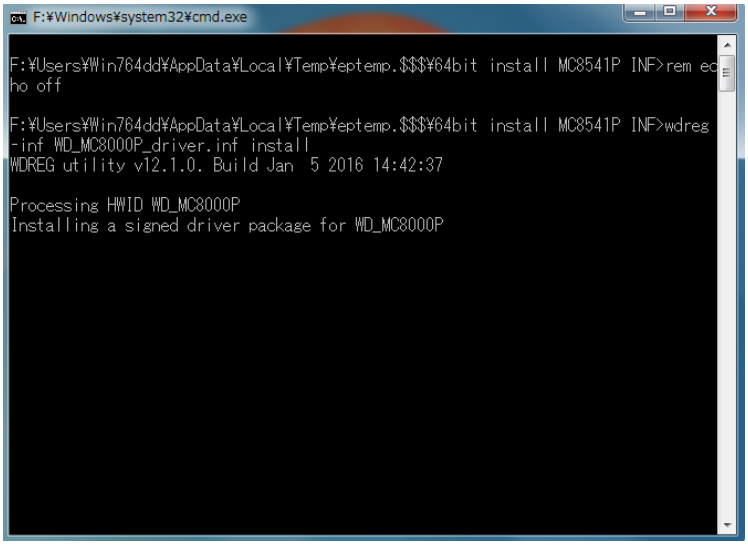


(9) The following 2 dialogs appear, click [Install].

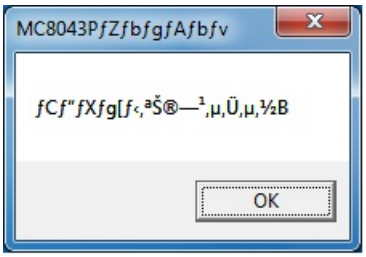




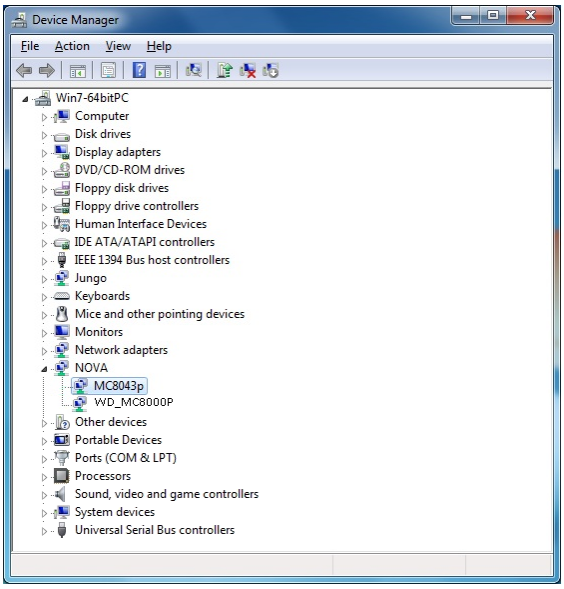
(10) When installation has completed, the window will automatically close. (Installation may take time.)

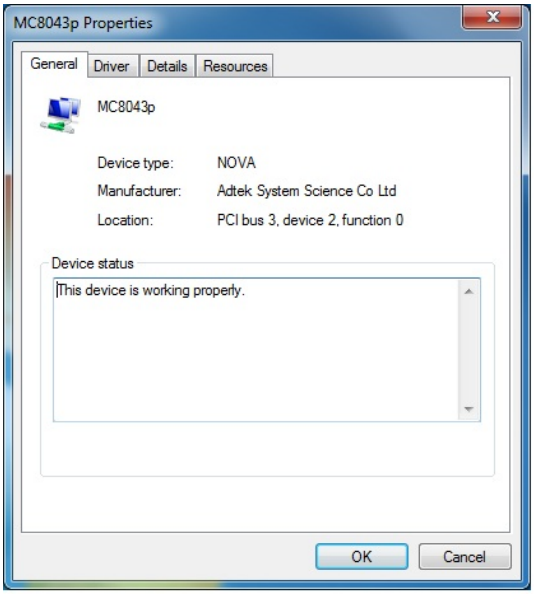


(11) Click OK to complete the driver installation.

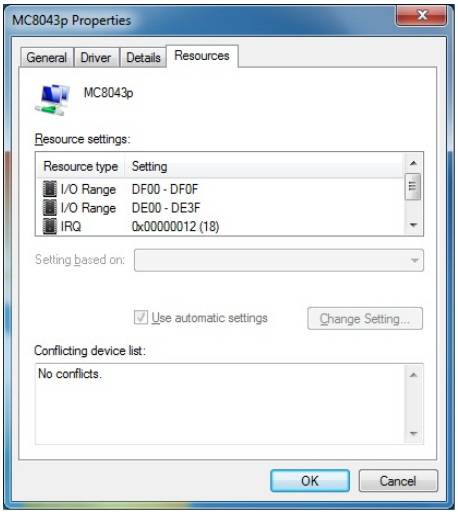


(12) The installation has finished. Check the installation is successfully completed by the following steps:
[Control Panel] → [System and security] → [Device Manager] (shown on the left below). [MC XXXX P] and [WD_MC8000P」 is shown under "**NOVA**", and then click the "**General**" tab to display the window shown on the right below. If the driver is correctly installed, you can see "**This device is working properly**" in the Device status field. (Read by replacing MC8043P in the dialog box with your model name.)
(If PCI-express board is installed, board name is the same as PCI. For example, MC8082Pe is installed, its name is "MC8082P".)





After the installation is successfully completed, check the resource settings and No conflicts.



2.5 Uninstallation of Device Driver

This chapter describes how to uninstall device driver.

There is no difference for the procedure of the installation for each Windows OS.

2.5.1 Uninstallation of Device Driver

This chapter describes how to uninstall device driver.

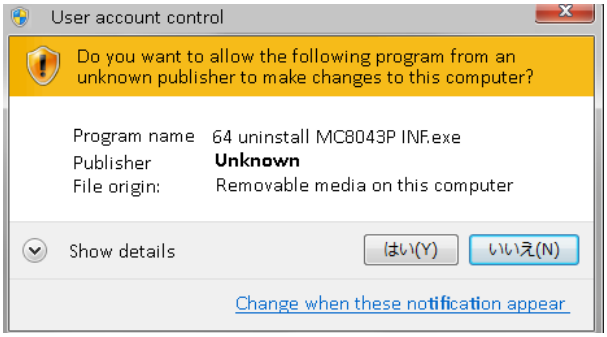
When the different kinds of boards are installed (For example, MC8043P and MC8541P are installed.), refer to the chapter 2.5.2 Uninstallation When the Different kinds of the Boards are installed.

2.5.1.1 Execute Uninstall XXXX INF.exe

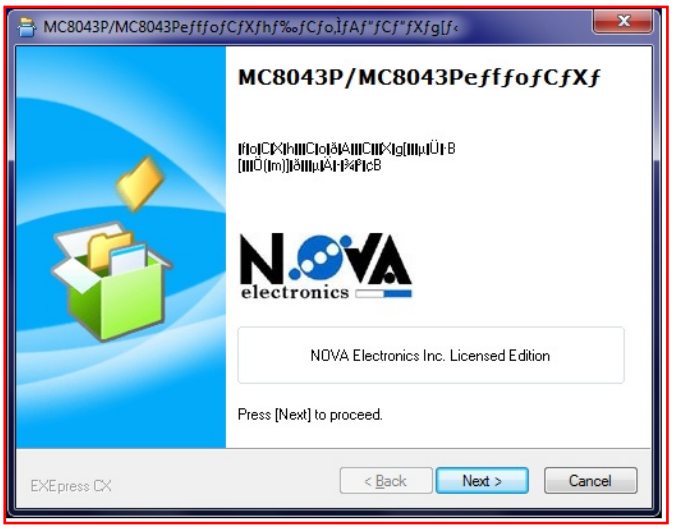
Execute uninstall XXXX INF.exe (XXXX indicates the model name) as below.

(1) Select Driver folder in the attached CD-ROM (When the attached CD-ROM in D drive, D:\Driver) or the downloaded software, double click [64 uninstall XXXX INF.exe] in 64Driver folder.
When using the device driver before Ver.8.0.0.0, delete it using the software Ver.8.0.0.0 or later.
(For 32bit, double click [32 uninstall XXXX INF.exe] in 32driver folder.)
(Read by replacing MC8043P in the dialog box with your model name.)

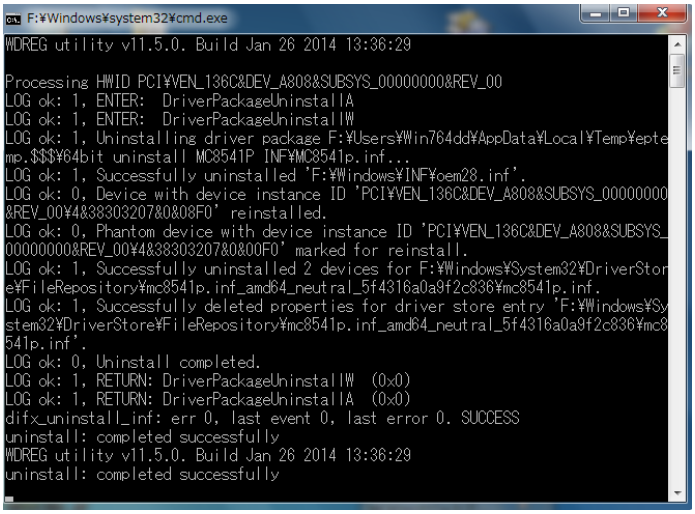
(2) When the following wizard appears. Click “Yes”. (Read by replacing MC8043P in the dialog box with your model name.)



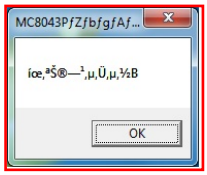
(3) Uninstallation window, click “Next”. (Read by replacing MC8541P in the dialog box with your model name.)



(4) When installation has completed, the window will automatically close. (Installation may take time.)



(5) Click OK to complete the driver installation.

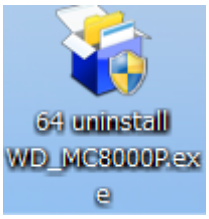


Next, 2.5.1.2 Execute Uninstall WD_MC8000P.exe.
(When not uninstalling a board, do not execute 2.5.1.2 Execute Uninstall WD_MC8000P.exe., proceed 2.5.1.3 Confirm Device Manager.)

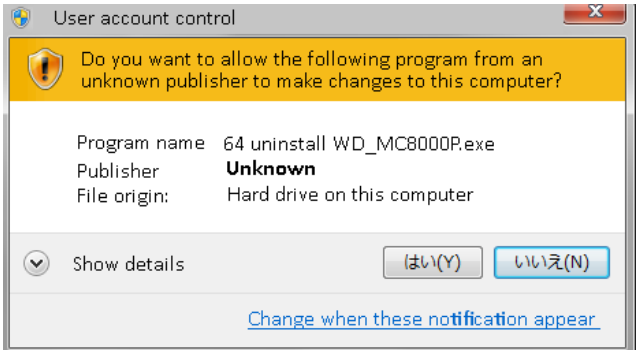
2.5.1.2 Execute Uninstall WD_MC8000P.exe.

Execute Uninstall WD_MC8000P.exe as below.

(1) Select Driver folder in the attached CD-ROM (When the attached CD-ROM in D drive, D:\Driver) or the downloaded software, double click [uninstall WD_MC8000P.exe] in 64Driver folder.
(For 32bit, double click [32 uninstall WD_MC8000P.exe] in 32driver folder.)

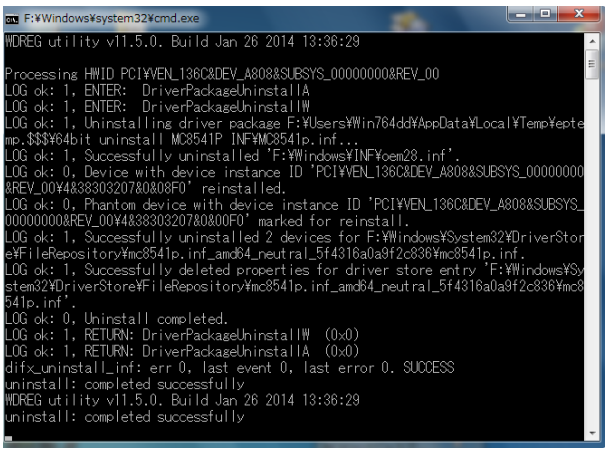


(2) When the following wizard appears. Click “Yes”.

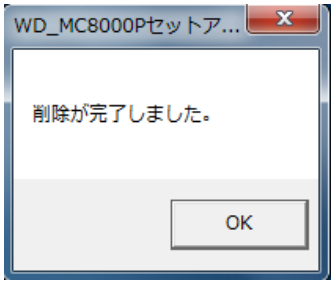


(3) Uninstallation window, click “Next”.

(4) When uninstallation has completed, the window will automatically close. (Uninstallation may take time.)

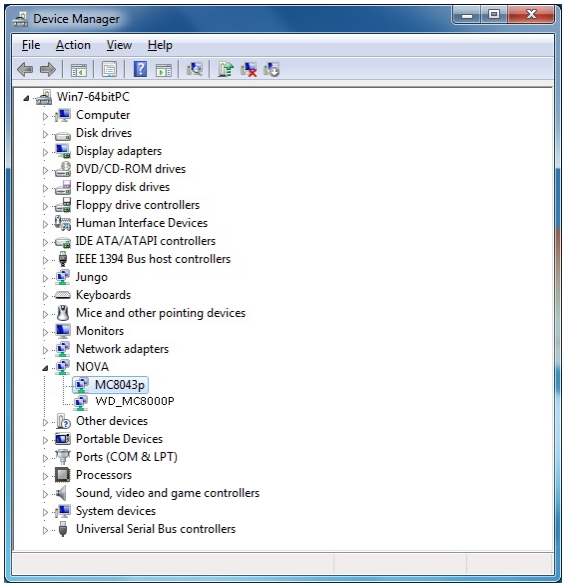


(5) Click OK to complete the driver uninstallation.

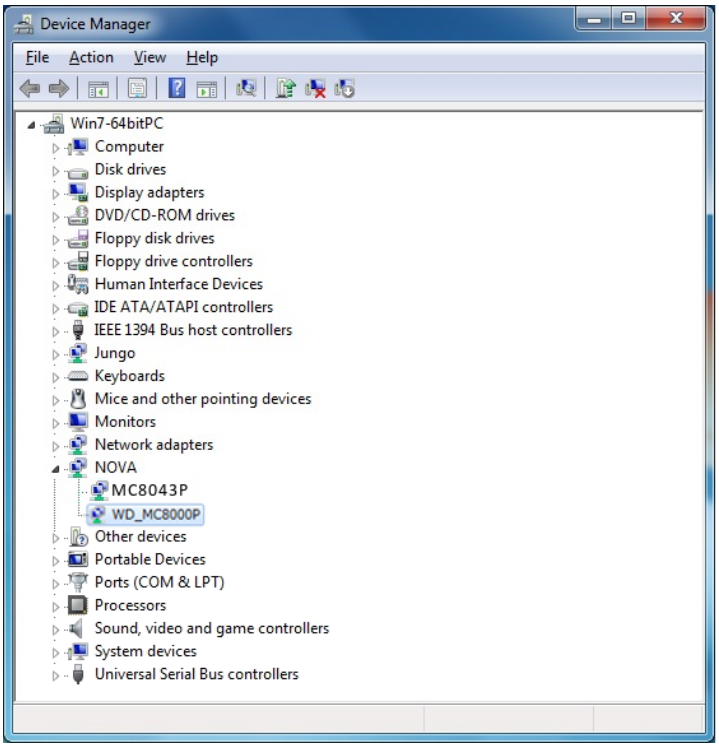


2.5.1.3 Confirm Device Manager

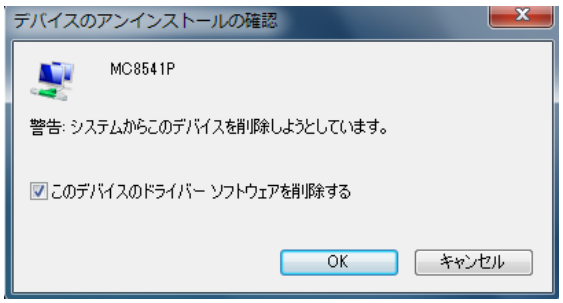
(1) Make sure that the board is deleted from [Control Panel] → [System and security] → [System] → [Device Manager].
For example, if you have deleted MC8043P board, ensure that the highlighted text (MC8043P) in the window is also deleted.



(2) When execute 2.5.1.2 Execute Uninstall WD_MC8000P.exe, ensure that the highlighted text (WD_MC8000P) in the window is also deleted.



(3) When not deleted, click the undeleted driver, select [Delete] from [Operation] menu and check [Delete this driver software]. Click [OK] and delete driver.



(4) Make sure that the PC is powered OFF and then remove the external cover and slot cover.
(5) Unscrew the mounting bracket.
(6) Remove the board by lifting steadily.

2.5.2 Uninstallation When the Different kinds of the Boards are installed.

This chapter describes how to uninstall when the different kind boards are installed. (For example, MC8043P and MC8541P are installed.),

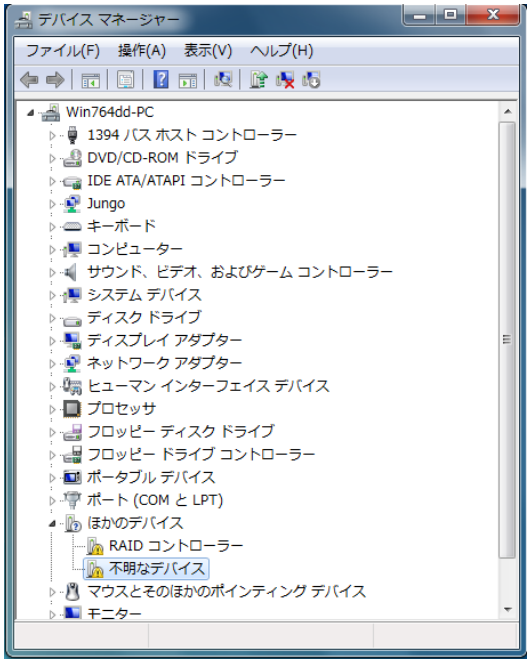
(1) Refer to 2.5.1.1 Execute Uninstall XXXX INF.exe and execute [uninstall XXXX INF.exe] of the board to be uninstalled.
(2) Execute other than (2) of 2.5.13 Confirm Device Manager.
If delete the multiple boards, execute (1) and (2) repeatedly.
(3) If uninstall all the boards, execute [Uninstall WD_MC8000P.exe] referring to 2.5.1.2 Execute Uninstall WD_MC8000P.exe.
(4) Execute 2.5.1.3 Confirm Device Manager

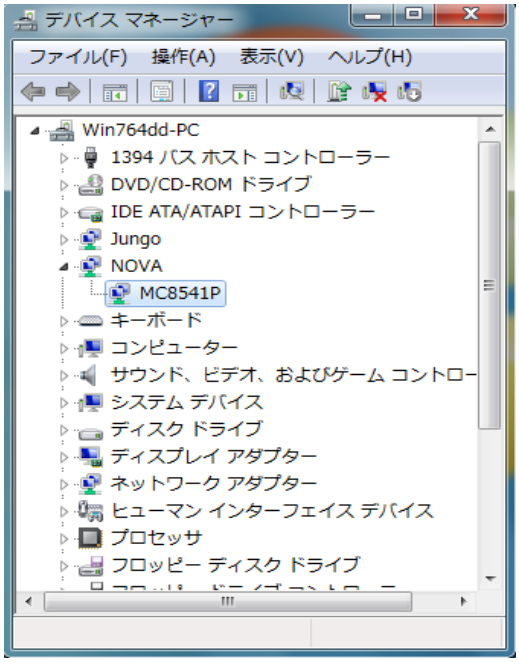
2.6 Updating Device Driver

To update the driver, follow the steps below.

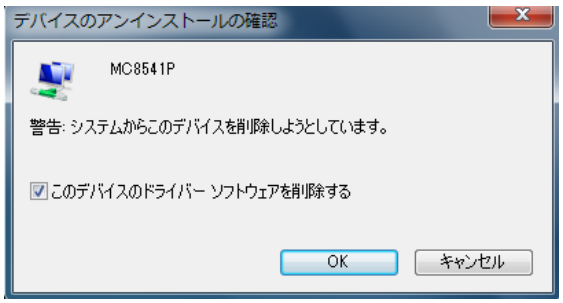
Make sure to close the other application.
Before starting the installation procedure, ensure that you are logged on to Windows with a user name having administrator authority. Otherwise, the installation is not successfully completed.
There is no difference for the procedure of the installation for each Windows OS.

(1) Delete device driver as 2.5.
(2) Reboot PC.
(3) Make sure that the board is deleted from [Control Panel] → [System and security] → [System] → [Device Manager]. When deleted, [Unknow device] is shown as the highlighted text in the left lower. When undeleted, board name and WD_MC8000P is shown as the highlighted text in the right lower.

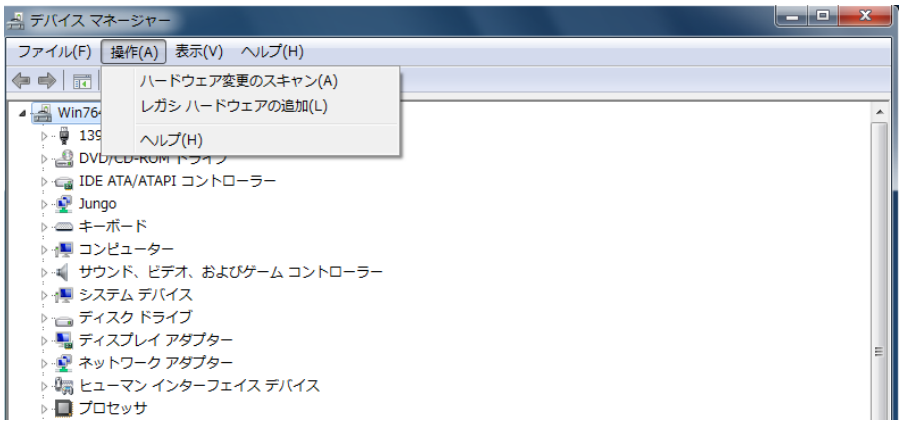




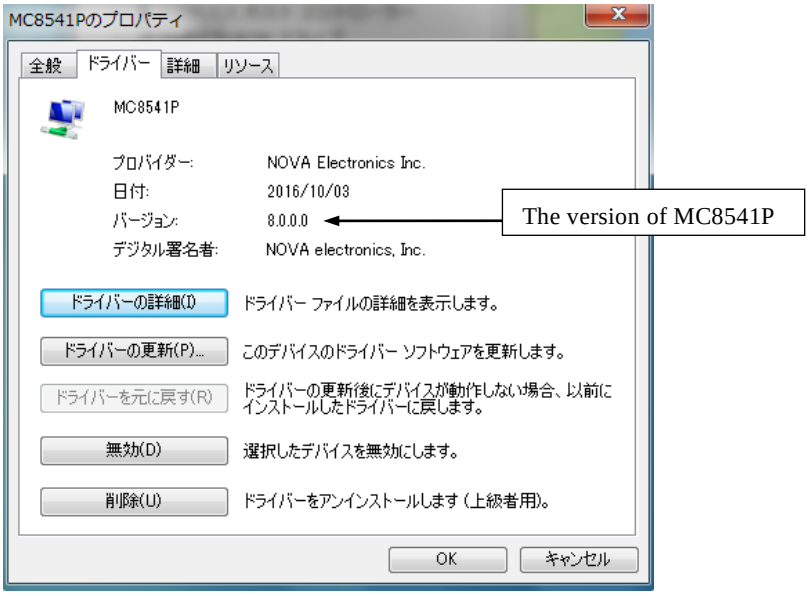
When not deleted, click the undeleted driver, select [Delete] from [Operation] menu and check [Delete this driver software]. Click [OK] and delete driver.

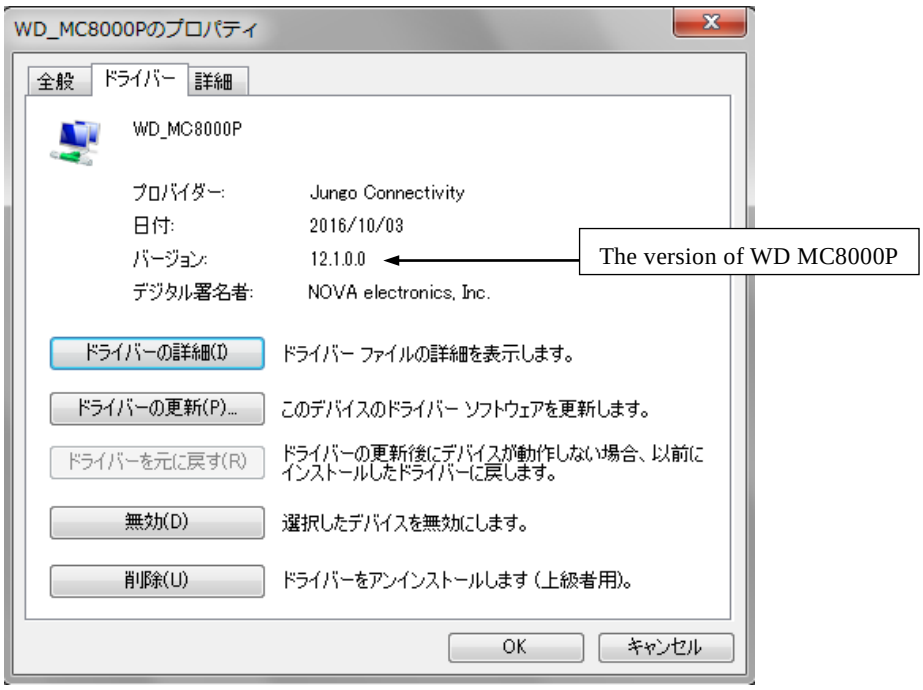


(4) [Operation] → [Scan for hardware changes], execute (3) again. Execute (3) → (4) repeatedly until correctly deleted .



(5) Prepare the device driver to be installed as the chapter 2.1 after confirming correctly deleted.
(6) Install it referring to (5) ~ (10) of the chapter 2.4.
(7) Confirm whether the version of [1. Driver version] of \64 Driver\Version.txt of driver folder in CD-ROM (When CD-ROM is in D drive, select D:\Driver), or select the downloaded driver folder on hard disc, and then double click “64bit uninstall xxxxx INF.exe” in the driver folder is the same with the following
(For 32bit, confirm 32 Driver\Version.txt file.)
(Read by replacing MC8541P in the dialog box with your model name.)




WD_MC8000Pのプロパティ
全般
ドライバー
詳細
WD_MC8000P
プロバイダー:
Jungo Connectivity
日付:
2016/10/03
バージョン:
12.1.0.0
デジタル署名者:
NOVA electronics, Inc.
The version of WD MC8000P
ドライバーの詳細(I)
ドライバー ファイルの詳細を表示します。
ドライバーの更新(P)...
このデバイスのドライバー ソフトウェアを更新します。
ドライバーを元に戻す(R)
ドライバーの更新後にデバイスが動作しない場合、以前にインストールしたドライバーに戻します。
無効(D)
選択したデバイスを無効にします。
削除(U)
ドライバーをアンインストールします (上級者用)。
OK
キャンセル

3. Programming

This chapter describes software specifications and how to program applications.
Applications can be developed with Microsoft Visual C++ (VC++), Microsoft Visual Basic (VB) or Microsoft Visual C# (C#).

3.1 Operating Environment

Operating Systems	Windows 7(32bit,64bit), Windows 8.1(32bit,64bit) & Windows 10(32bit,64bit)
Support Languages	Microsoft Visual C++ 2008 or later
	Microsoft Visual Basic 2008 or later
	Microsoft Visual C# 2008 or later

Notes: When developing applications, refer to the MSDN website for more information about development tools.
When you run a sample program in Visual Studio.2010 or later, migrate a sample program with reference to MSDN documentation. 64bit device driver only applies to 64bit application, 32bit device driver to 32bit application.
For MC8082Pe, use the same file with MC8082P, for MC8043Pe same one with MC8043P.
For MC8541Pe, use the same file with MC8541P, for MC8581Pe same one with MC8581P.
For MC8022P/MC8042P, use the file for MC8082P.

3.2 Software Configuration

3.2.1 Software List

Software	Folder	File Name, Folder	Description
Device driver	Driver\ 32 Driver	32 install MC8043P INF.exe	32bit device driver installer for MC8043P.
		32 install MC8082P INF.exe	32bit device driver installer for MC8082P
		32 install MC8541P INF.exe	32bit device driver installer for MC8541P
		32 install MC8581P INF.exe	32bit device driver installer for MC8581P
		32 uninstall MC8043P INF.exe	32bit device driver uninstaller for MC8043P
		32 uninstall MC8082P INF.exe	32bit device driver uninstaller for MC8082P
		32 uninstall MC8541P INF.exe	32bit device driver uninstaller for MC8541P
		32 uninstall MC8581P INF.exe	32bit device driver uninstaller for MC8581P
		uninstall WD_MC8000P.exe	32bit device driver uninstaller for WD_MC8000P

Software	Folder	File Name, Folder	Description
Device driver	Driver\ 64 Driver	64 install MC8043P INF.exe	64bit device driver installer for MC8043P
		64 install MC8082P INF.exe	64bit device driver installer for MC8082P
		64 install MC8541P INF.exe	64bit device driver installer for MC8541P
		64 install MC8581P INF.exe	64bit device driver installer for MC8581P
		64 uninstall MC8043P INF.exe	64bit device driver uninstaller for MC8043P
		64 uninstall MC8082P INF.exe	64bit device driver uninstaller for MC8082P
		64 uninstall MC8541P INF.exe	64bit device driver uninstaller for MC8541P
		64 uninstall MC8581P INF.exe	64bit device driver uninstaller for MC8581P
		64 uninstall WD_MC8000P.exe	64bit device driver uninstaller for WD_MC8000P
Library	Lib\VC \32 lib	MC8000P.LIB	Library to use MC8000P.DLL 32bit & VC++ only
		MC8000P_DLL.h	Header definition file to use MC8000P.DLL VC++ only
	Lib\VC \64 lib	MC8000P.LIB	Library to use MC8000P.DLL 64bit & VC++ only
		MC8000P_DLL.h	Header definition file to use MC8000P.DLL VC++ only
	Lib\VB.NET	MC8000P_DLL.vb	Declare definition to use MC8000P.DLL VB.NET only
	Lib\C#	Mc8000pWrap.dll	Class library to use MC8000P.DLL C# only
Sample program	Please download it from our website URL: http://www.novaelec.co.jp/eng/		
Evaluation tool for the board equipped with MCX304 ※MC8082P	Tool \MCX304 Board \64 Tool (\32 Tool)	MCX304-A.exe	Evaluation tool for the board with MCX304
		MCX304-B.exe	Application to execute each command with settings such as parameters an mode in the window.
		ParameterSample	Parameter sample file: The Action of each file can be checked in the plot pane by loading this file i MCX304-x.exe and executing the plot.
Evaluation tool for the board equipped with MCX314As ※MC8043P	Tool \MCX314As Board \64 Tool (\32 Tool)	MCX314As-A.exe	Evaluation tool for the board with MCX314As Application to execute each command with settings such as parameters an mode in the window.
		ParameterSample	Parameter sample file: The Action of each file can be checked in the plot pane by loading this file i MCX314As-x.exe and executing the plot. Parameter sample file: The Action of each file can be checked in the plot pane by loading this file i MCX314As-x.exe and executing the plot.
Evaluation tool for the board equipped with MCX514 ※MC8541P & MC8581P	Tool \MCX514 Board \64 Tool (\32 Tool)	MCX514.exe	Evaluation tool for the board with MCX514 Application to execute each command with settings such as parameters an mode in the window.

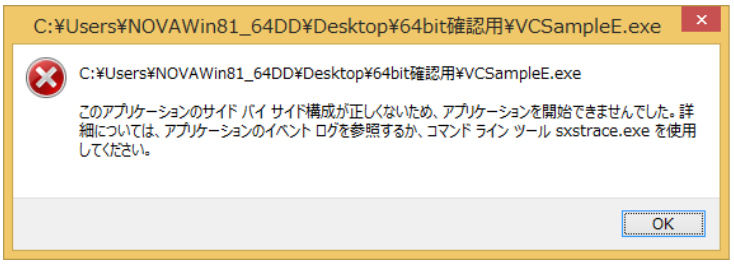
Note: Description about files automatically created by VC++ MFC AppWizard is omitted.
For the evaluation tool, use exe.file in 32bit folder (\32 Tool) for 32bit OS, 64bit folder (\64 Tool) for 64bit OS.

3.2.2 When error occurs in executing sample program and evaluation tool

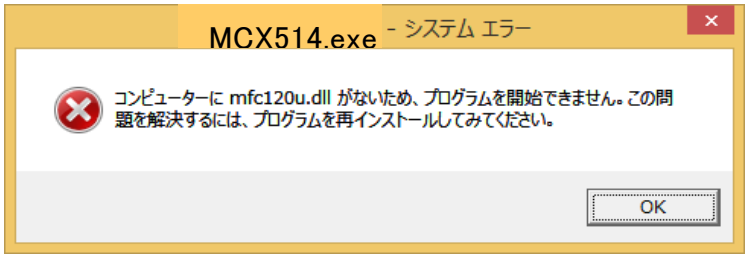
■ Error when executing exe create by VC++.

When Visual Studio is not installed on PC which is not installed MC board, the error as below occurs in executing exe file. When the error occurs, install Redistributable Package from the website of Microsoft. Install the sample program for Visual Studio 2008 and the evaluation tool for Visual Studio 2013. There are 32bitOS and 64bitOS version, install either according to your PC.
For example, to release the error of the evaluation tool on 64bitOS, install Redistributable Package for 64bit version of Visual Studio 2013. And, to release the error of the sample program, install Redistributable Package for 64bit version of Visual Studio 2008.

Note : The details of errors and how to release them, see the support page of Microsoft and release the error according to Microsoft support.



Example of error when executing VC++ sample program.

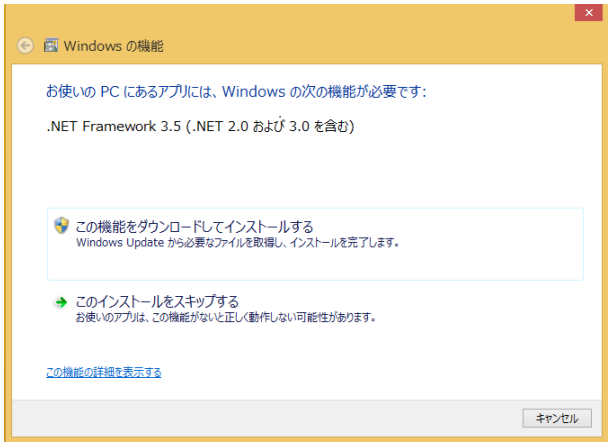


Example of error when executing the evaluation tool.

■ Error when executing exe create by VB.NET.

When NET Framework 3.5 is not installed on PC which is not installed MC board, the error as below occurs in executing exe file. When the error occurs, install NET Framework 3.5 according to the window or enable NET Framework 3.5 as follows. [Control panel]- [Program]-[turn windows features on or off].

Note : The details of errors and how to release them, see the support page of Microsoft and release the error according to Microsoft support



Example of error when executing VB.NET sample program.

3.3 Development Procedure

3.3.1 When using VC++

MC8000P.lib and MC8000P_DLL.h files are used in applications. These two files are for VC++2008or later.

(1) Copy two files, MC8000P.lib and MC8000P_DLL.h in \Lib\VC\32 lib folder for 62bitOS or \Lib\VC\64 lib for 64bitOS to the application's folder to develop.
(2) Add MC8000P_DLL.h to your project in VC++. Also, include MC8000P_DLL.h to the source file using API function.
(3) [Project] → [Property] → select [Linker] → [Input], add MC8000P.lib to [Additional dependent file]
(4) Program by using functions of "4.1 API" or "5.1 API".

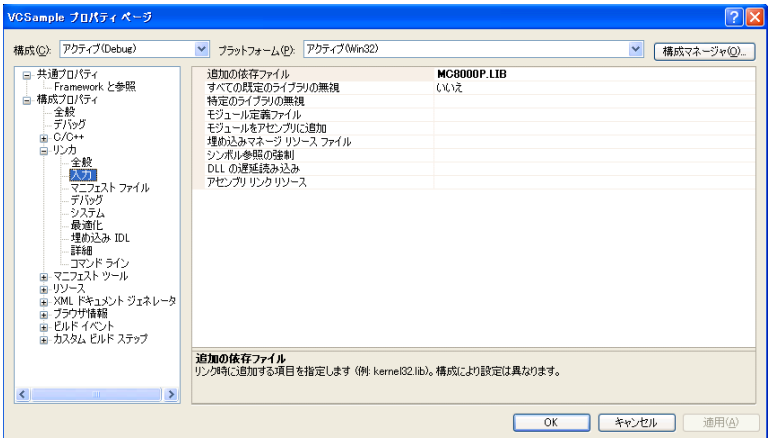


Fig.3.3-1 VC++ 2008 Property Pages

3.3.2 When using VB.NET

MC8000P_DLL.vb file is used in applications. This file is for VB2008 or later.

(1) Add MC8000P_DLL.vb in \Lib\VB.NET folder to your project to the application's folder to develop.
(2) Program by using functions of "4.1 API" or "5.1 API".

Note: The user cannot use the interrupt in VB.NET applications.

3.3.3 When using C#

.NET C# class library MC8000pWrap.dll is used in MC8000P applications.

(1) Add MC8000pWrap.dll in \Lib\Csharp folder to the application's folder to develop.
(2) Go to [Project] – [Add Reference], and then add a reference to "MC8000pWrap.dll".
(3) Add the namespace "MC8000pWrap" with a using statement to the source file.
(2) Program by using functions of "4.1 API" or "5.1 API".

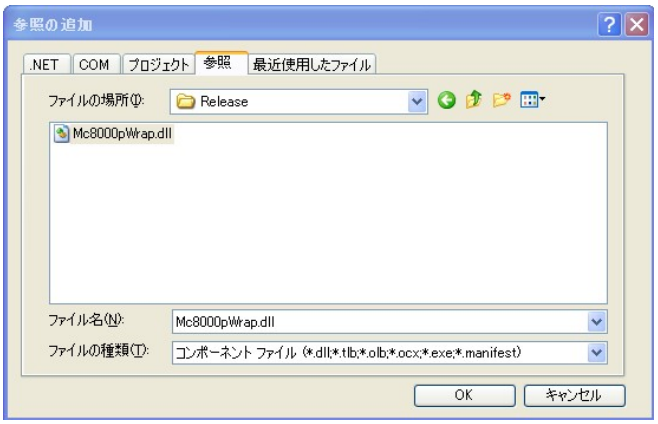


Fig.3.3-2 Add C# 2008 reference

4. MC8000P series board

This chapter describes API used for the boards as below.

Board	IC	Number of IC	Number of axis
MC8043P / MC8043Pe	MCX314As	1	4
MC8082P / MC8082Pe	MCX304	2	8
MC8022P	MCX304	1	2
MC8042P	MCX304	1	4

4.1 API

API provided by MC8000P.SYS and MC8000P.DLL.

4.1.1. Function list

The following table is the API function list.
The column of VC, VB.NET, C# indicates the availability of each function in each language.
VC++ See VC column.
VB.NET See VB.NET column
C#. See C# columns.

○ is available and × is not.

(1) Basic Function

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_Open	Start to use the board	○	○	○	20	
Nmc_Close	Stop to use the board	○	○	○	〃	
Nmc_CloseAll	Stop to use all the boards	○	○	○	21	
Nmc_GetBoardInfo	Get information about the opened board	○	○	○	〃	
Nmc_OutPort	Write data to output port	○	○	○	22	
Nmc_InPort	Read data from input port	○	○	○	〃	
Nmc_WriteReg	Write data to register of the board	○	○	○	23	
Nmc_ReadReg	Read data from register of the board	○	○	○	〃	
Nmc_SetEvent	Set user function to handle an interrupt.	○	×	○	24	
Nmc_ResetEvent	Release user function to handle an interrupt.	○	×	○	25	
Nmc_ReadEvent	Read RR3 value of each axis right after the interrupt generated.	○	×	○	〃	

(2) Reset, Command

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_Reset	Reset the IC on the board	○	○	○	26	
Nmc_Command	Execute the command of the specified axis	○	○	○	〃	
Nmc_Command_IP	Execute the interpolation command	○	○	○	27	*1

(3) Write Register

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_WriteReg0	WR0 (Command Register) Writing	○	○	○	27	
Nmc_WriteReg1	WR1 (Mode Register 1) Writing	○	○	○	28	
Nmc_WriteReg2	WR2 (Mode Register 2) Writing	○	○	○	〃	
Nmc_WriteReg3	WR3 (Mode Register 3) Writing	○	○	○	29	
Nmc_WriteReg4	WR4 (Output Register) Writing	○	○	○	〃	
Nmc_WriteReg5	WR5 Writing	○	○	○	30	
Nmc_WriteReg6	WR6 (Write Data Register 1) Writing	○	○	○	〃	
Nmc_WriteReg7	WR7 (Write Data Register 2) Writing	○	○	○	31	

(4) Read Register

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_ReadReg0	RR0 (Main Status Register) Reading	○	○	○	31	
Nmc_ReadReg1	RR1 (Status Register 1) Reading	○	○	○	32	
Nmc_ReadReg2	RR2 (Status Register 2) Reading	○	○	○	〃	
Nmc_ReadReg4	RR4 (Input Register 1) Reading	○	○	○	〃	
Nmc_ReadReg5	RR5 (Input Register 2) Reading	○	○	○	33	
Nmc_ReadReg6	RR6 (Read Data Register 1) Reading	○	○	○	〃	
Nmc_ReadReg7	RR7 (Read Data Register 2) Reading	○	○	○	〃	

(5) Parameter Settings

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_Range	Range Setting	○	○	○	34	
Nmc_Jerk	Jerk Setting	○	○	○	〃	
Nmc_Acc	Acceleration Setting	○	○	○	35	
Nmc_Dec	Deceleration Setting	○	○	○	〃	
Nmc_StartSpd	Initial Speed Setting	○	○	○	36	
Nmc_Speed	Drive Speed Setting	○	○	○	〃	
Nmc_Pulse	Output Pulse Number/Interpolation Finish Point Setting (For VC, C#)	○	×	○	37	
Nmc_Pulse_VB	Output Pulse Number/Interpolation Finish Point Setting (For VB)	×	○	×	〃	
Nmc_DecP	Manual Decelerating Point Setting (For VC, C#)	○	×	○	38	
Nmc_DecP_VB	Manual Decelerating Point Setting (For VB)	×	○	×	〃	
Nmc_Center	Circular Center Point Setting	○	○	○	39	*1
Nmc_Lp	Logical Position Counter Setting	○	○	○	〃	
Nmc_Ep	Real Position Counter Setting	○	○	○	40	
Nmc_CompP	COMP+ Register Setting	○	○	○	〃	
Nmc_CompM	COMP– Register Setting	○	○	○	41	
Nmc_AccOfst	Acceleration Counter Offsetting	○	○	○	〃	
Nmc_DJerk	Deceleration Increasing Rate Setting	○	○	○	42	*1
Nmc_HomeSpd	Home Search Speed Setting	○	○	○	〃	

(6) Other Mode Settings

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_ExpMode	Extension Mode Setting	○	○	○	43	*1
Nmc_SyncMode	Synchronous Action Mode Setting	○	○	○	〃	*1
Nmc_HomeMode	Automatic Home Search Mode Setting	○	○	○	44	*2

(7) Data Reading

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_ReadLp	Logical Position Counter Reading	○	○	○	44	
Nmc_ReadEp	Real Position Counter Reading	○	○	○	〃	
Nmc_ReadSpeed	Current Drive Speed Reading	○	○	○	45	
Nmc_ReadAccDec	Current Acceleration/Deceleration Reading	○	○	○	〃	
Nmc_ReadSyncBuff	Synchronous Buffer Register Reading	○	○	○	46	*1

(8) Status Reading

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_GetDriveStatus	Drive Status Reading	○	○	○	46	
Nmc_GetCNextStatus	The Status Reading of Ready Signal for Writing of Continuous Interpolation	○	○	○	47	*1
Nmc_GetBpSc	BP Interpolation Stack Counter Reading	○	○	○	〃	*1

(9) Writing / Reading

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_WriteRegSetAxis	Axis Assignment Write Register Writing (WR1~3)	○	○	○	48	
Nmc_ReadRegSetAxis	Axis Assignment Read Register Reading (RR1~2)	○	○	○	〃	
Nmc_WriteData	Data Writing (Parameter)	○	○	○	49	
Nmc_WriteData2	Data Writing (Extension Mode, Synchronous Action Mode)	○	○	○	〃	*1
Nmc_ReadData	Data Reading	○	○	○	50	

(10) Interpolation Execution

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_2BPExec	2-axis BP Interpolation Execution	○	○	○	50	*1
Nmc_3BPExec	3-axis BP Interpolation Execution	○	○	○	52	*1
Nmc_2BPExec_BG	2-axis BP Interpolation Execution (run in the background)	○	○	○	53	*1
Nmc_3BPExec_BG	3-axis BP Interpolation Execution (run in the background)	○	○	○	56	*1
Nmc_2CIPExec	2-axis Continuous Interpolation Execution	○	○	○	58	*1
Nmc_3CIPExec	3-axis Continuous Interpolation Execution	○	○	○	60	*1
Nmc_2CIPExec_BG	2-axis Continuous Interpolation Execution (run in the background)	○	○	○	63	*1
Nmc_3CIPExec_BG	3-axis Continuous Interpolation Execution (run in the background)	○	○	○	66	*1
Nmc_IPStop	Stop the Interpolation Execution	○	○	○	69	*1
Nmc_IPGetMsgNo	Get board and IC number from received message at the end of the Interpolation	○	○	○	70	*1

*1: Function for MC8043P/MC8043Pe only

*2: Function for MC8082P/MC8082Pe, MC8022P and MC8042P only

4.1.2. Function Specifications

For VC++ users See **VC** and [VC]
For VB.NET users See **VB.NET** and [VB.NET]
For C# users See **C#** and [C#]
And others are common to each language.

Function Name	Function and Content

Nmc_Open	Start the board.
	
	VC BOOL Nmc_Open(int No, BOOL IntrptFlg);
	VB.NET Function Nmc_Open(ByVal No As Integer, ByVal IntrptFlg As Integer) As Integer
	C# bool MC8000P.Nmc_Open(int No, bool IntrptFlg);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IntrptFlg Set the flag to use interrupt.
	[VC] TRUE: use interrupt. FALSE: not use.
	[VB.NET] FALSE only. Interrupt cannot be used in VB.
	[C#] true: use interrupt. false: not use.
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[VB.NET] If the function succeeds, the return value is nonzero.
	If the function fails, the return value is 0.
	[C#] If the function succeeds, the return value is true.
	If the function fails, the return value is false.
	
	Example
	[VC] status = Nmc_Open(0, FALSE); // Open the board 0 and not use the interrupt.
	[VB.NET] status = Nmc_Open(0, False)
	[C#] status = MC8000P.Nmc_Open(0, false);

Nmc_Close	Terminate the board.
	
	VC BOOL Nmc_Close(int No);
	VB.NET Function Nmc_Close(ByVal No As Integer) As Integer
	C# bool MC8000P.Nmc_Close(int No);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[VB.NET] If the function succeeds, the return value is nonzero.
	If the function fails, the return value is 0.
	[C#] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	
	Example
	[VC] status = Nmc_Close(0); // Close the board 0.
	[VB.NET] status = Nmc_Close(0)
	[C#] status = MC8000P.Nmc_Close(0);

Function Name	Function and Content
Nmc_CloseAll	Terminate all the boards. **VC** BOOL Nmc_CloseAll(void); **VB.NET** Function Nmc_CloseAll() As Integer **C#** bool MC8000P.Nmc_CloseAll(); **Input Parameter** None **Return Value** [VC] If the function succeeds, the return value is TRUE. If the function fails, the return value is FALSE. [VB.NET] If the function succeeds, the return value is nonzero. If the function fails, the return value is 0. [C#] If the function succeeds, the return value is true. If the function fails, the return value is false. **Example** [VC] status = Nmc_CloseAll(); // Close all the boards. [C#] status = MC8000P.Nmc_CloseAll();

Nmc_GetBoardInfo	Get Device ID of the opened board.
	
	VC BOOL Nmc_GetBoardInfo(int No, USHORT* DeviceID);
	VB.NET Function Nmc_GetBoardInfo(ByVal No As Integer, ByRef DeviceID As Short) As Integer
	C# bool MC8000P.Nmc_GetBoardInfo(int No, out ushort DeviceID);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	DeviceID [VC] Address of a variable to store the Device ID obtained from the board.
	[VB.NET] [C#] Address of a variable to store the Device ID obtained from the board.
	See 4.1.3 (1) ③ regarding Device ID of each board.
	[VC][VB.NET] MC8082P/MC8082Pe,42P,22P are ID_MC8082P,
	MC8043P/MC8043Pe are ID_MC8043P,
	[C#] MC8082P/MC8082Pe,42P,22P are Dev_ID.MC8082P, MC8080P is
	Dev_ID.MC8080P, MC8043P/MC8043Pe are Dev_ID.MC8043P.
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[VB.NET] If the function succeeds, the return value is nonzero.
	If the function fails, the return value is 0.
	[C#] If the function succeeds, the return value is true.
	If the function fails, the return value is false.
	Example
	[VC] USHORT DecviceID;
	status = Nmc_GetBoardInfo(No, &DeviceID); // Get Device ID.
	if(DeviceID == ID_MC8082P) // When the board is MC8082P.
	[VB.NET] Dim DeviceID As Short
	status = Nmc_GetBoardInfo(No, DeviceID)
	If DeviceID = ID_MC8082P Then
	[C#] ushort DeviceID;
	status = Nmc_GetBoardInfo(No, out DeviceID);
	if(DeviceID == Dev_ID.MC8082P)

Function Name	Function and Content

Nmc_OutPort	Write 2-byte data into output port.
	
	VC void Nmc_OutPort(int No, long Adr, long Data);
	VB.NET Sub Nmc_OutPort(ByVal No As Integer, ByVal Adr As Integer, ByVal Data As Integer)
	C# void MC8000P.Nmc_OutPort(int No, REG_MCX Adr, int Dat);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	Adr Address to be written. I/O address described in User's Manual of each board.
	See 4.1.3 (1)①, (6) for more details.
	Data Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_OutPort(No, 0, 0x8000); // Soft reset IC-A.(Write into WR0)
	[VB.NET] Call Nmc_OutPort(No, 0, &H8000)
	[C#] MC8000P.Nmc_OutPort(No, REG_MCX.WR0_A, 0x8000);

Nmc_InPort	Read out 2-byte data from input port.
	
	VC long Nmc_InPort(int No, long Adr);
	VB.NET Function Nmc_InPort(ByVal No As Integer, ByVal adr As Integer) As Integer
	C# int MC8000P.Nmc_InPort(int No, REG_MCX Adr);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	Adr Address to be read. I/O address described in User's Manual of each board.
	See 4.1.3 (1)①, (6) for more details.
	
	Return Value
	Data read out from input port.
	
	Example
	[VC] data = Nmc_InPort(No, 0); // Read out the read register RR0 of IC-A.
	[VB.NET] data = Nmc_InPort(No, 0)
	[C#] data = MC8000P.Nmc_InPort(No, REG_MCX.RR0_A);
	
	Note
	[VC] [C#] Regarding reading RR3 register data, refer to Nmc_ReadEvent function.

Function Name	Function and Content

Nmc_WriteReg	Write data into write register (WR0~WR7) of the board.
	
	VC void Nmc_WriteReg(int No, int IcNo, long RegNum, long Dat);
	VB.NET Sub Nmc_WriteReg(ByVal No As Integer, ByVal IcNo As Integer, ByVal RegNum As Integer ByVal Dat As Integer)
	C# void MC8000P.Nmc_WriteReg(int No, int IcNo, int RegNum, int Dat);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1).
	Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	RegNum Register to be written (MCX_WR0~MCX_WR7). See 4.1.3 (1) for more details.
	e.g. [VC][VB.NET] Specify MCX_WR0 for WR0 and MCX_WR1 for WR1.
	[C#] Specify REG_MCX.WR0 for WR0 and REG_MCX.WR1 for WR1.
	Dat Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg(No, 0, MCX_WR0, 0x8000); // Soft reset IC-A.(Write into WR0)
	[VB.NET] Call Nmc_WriteReg(No, 0, MCX_WR0, &H8000)
	[C#] MC8000P.Nmc_WriteReg(No, 0, REG_MCX.WR0, 0x8000);

Nmc_ReadReg	Read out data from read register (RR0~RR7) of the board.
	
	VC long Nmc_ReadReg(int No, int IcNo, long RegNum);
	VB.NET Function Nmc_ReadReg(ByVal No As Integer, ByVal IcNo As Interger, ByVal RegNum As Integer) As Integer
	C# void MC8000P.Nmc_ReadReg(int No, int IcNo, int RegNum);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1).
	Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	RegNum Register to be read (MCX_RR0~MCX_RR7). See 4.1.3 (1) for more details.
	e.g. [VC][VB._NET] Specify MCX_RR0 for RR0 and MCX_RR1 for RR1.
	[C#] Specify REG_MCX.RR0 for RR0 and REG_MCX.RR1 for RR1.
	
	Return Value
	Data read out from read register.
	
	Example
	[VC] data = Nmc_ReadReg(No, 0, MCX_RR0); // Read out the read register RR0 of IC-A.
	[VB.NET] data = Nmc_ReadReg(No, 0, MCX_RR0)
	[C#] data = MC8000P.Nmc_ReadReg(No, 0, REG_MCX.RR0);
	
	Note
	[VC] [C#] Regarding reading the RR3 register data, refer to Nmc_ReadEvent function.

Function Name	Function and Content

Nmc_SetEvent	Set user function to handle an interrupt.
	By executing this function, the user function is called when an interrupt occurs and then one specified argument is passed. This user function is run as one thread.
	To release user function to handle an interrupt, execute Nmc_ResetEvent.
	
	VC BOOL Nmc_SetEvent
	(int No, LPTHREAD_START_ROUTINE UserThread, LPVOID lpParameter);
	VB.NET cannot be used.
	C# bool MC8000P.Nmc_SetEvent(int No, UserThread Callback, int param);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	[VC] UserThread User function address
	[VC] lpParameter Assign one argument to be pass to user function thread.
	Set the available pointer for the thread.
	When not using the argument, set NULL.
	When using the pointer, set the available pointer when the user function is called.
	[C#] Callback User method (delegate type)
	See 4.1.4 Usage for more details.
	param Assign one parameter (int only, such as numeric type) to pass to user function after an interrupt occurs.
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[C#] If the function succeeds, the return value is true.
	If the function fails, the return value is false.
	Example
	[VC]
	(When the board number is 0)
	status = Nmc_SetEvent(0, MC_EventFunc0, lpParam);
	// Set the function address and argument
	Nmc_WriteReg1(0, 0, AXIS_ALL, 0x8000); // Interrupt occurs at the stop of IC-A (All axes)
	(When the board number is 1)
	status = Nmc_SetEvent(1, MC_EventFunc1, NULL);
	// Set the function address and argument
	Nmc_WriteReg1(1, 0, AXIS_ALL, 0x8000); // Interrupt occurs at the stop of IC-A (All axes)
	
	■ Example of interrupt user function
	DWORD WINAPI MC_EventFunc0(LPVOID lpParam)
	{
	long Rr3X, Rr3Y, Rr3Z, Rr3U;
	Nmc_ReadEvent(0, 0, &Rr3X,&Rr3Y,&Rr3Z,&Rr3U);
	//Board 0, read out RR3 interrupt data of IC-A
	. . . .
	return 0;
	}
	DWORD WINAPI MC_EventFunc1(LPVOID lpParam)
	{
	long Rr3X, Rr3Y, Rr3Z, Rr3U;
	Nmc_ReadEvent(1, 0, &Rr3X, &Rr3Y, &Rr3Z, &Rr3U);
	//Board 1, read RR3 interrupt data of IC-A
	. . . .
	return 0;
	}
	[C#]
	// Assign interrupt method isr to a delegate type variable
	MC8000P.callback[0] = new MC8000P.UserThread(isr);
	//Set user method to handle an interrupt
	MC8000P.Nmc_SetEvent(no, MC8000P.callback[0], param);

Function Name	Function and Content

Nmc_ResetEvent	Release user function to handle an interrupt.
	By executing this function, the user function is not called when an interrupt occurs.
	
	VC BOOL Nmc_ResetEvent(int No);
	VB.NET cannot be used.
	C# bool MC8000P.Nmc_ResetEvent(int No);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[C#] If the function succeeds, the return value is true.
	If the function fails, the return value is false.
	Example
	[VC] status = Nmc_ResetEvent(No);
	[C#] MC8000P.Nmc_WriteReg1(No, AXIS.ALL, 0x0000); // No interrupt (All axes)
	status = MC8000P.Nmc_ResetEvent(No);

Nmc_ReadEvent	Read the value of RR3 of each axis right after an interrupt generation. (RR3 will be cleared after reading.)
	
	VC BOOL Nmc_ReadEvent(int No, int IcNo, long* RrIrX, long* RrIrY, long* RrIrZ, long* RrIrU);
	VB.NET cannot be used.
	C# bool MC8000P.Nmc_ReadEvent(int No, int IcNo, out int RrIrX, out int RrIrY, out int RrIrZ, out int RrIrU);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1).
	Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	RrIrX Pointer to a variable that receives the X axis RR3 value.
	RrIrY Pointer to a variable that receives the Y axis RR3 value.
	RrIrZ Pointer to a variable that receives the Z axis RR3 value.
	RrIrU Pointer to a variable that receives the U axis RR3 value.
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[C#] If the function succeeds, the return value is true.
	If the function fails, the return value is false.
	Example
	[VC] long Rr3X[2], Rr3Y[2], Rr3Z[2], Rr3U[2];
	Nmc_ReadEvent(No, 0, &Rr3X[0], &Rr3Y[0], &Rr3Z[0], &Rr3U[0]);
	// Read out RR3 data of IC-A
	Nmc_ReadEvent(No, 1, &Rr3X[1], &Rr3Y[1], &Rr3Z[1], &Rr3U[1]);
	// Read out RR3 data of IC-B
	[C#] int Rr3X,Rr3Y,Rr3Z,Rr3U ; // X-axis, Y-axis, Z-axis, Axis
	MC8000P.Nmc_ReadEvent(No, IcNo, out Rr3X, out Rr3Y, out Rr3Z, out Rr3U);
	Note
	The RR3 value is cleared due to the driver operation, just after the interrupt occurs. The user must use this function in order to know the interrupt factor.
	In addition, when the interrupt occurs, the driver certainly reads and saves RR3 data regardless of the execution of Nmc_SetEvent or Nmc_ResetEvent functions. RR3 data saved in the driver can be cleared after reading by execution of Nmc_ReadEvent function.
	To clear the RR3 data of the driver, execute Nmc_ReadEvent function.

Function Name	Function and Content

Nmc_Reset	Reset the IC on the board.
	
	VC void Nmc_Reset(int No, int IcNo);
	VB.NET Sub Nmc_Reset(ByVal No As Integer, ByVal IcNo As Integer)
	C# void MC8000P.Nmc_Reset(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1).
	Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Reset(1, 0); // Reset IC-A of the board 1.
	[VB.NET] Call Nmc_Reset(1, 0)
	[C#] MC8000P.Nmc_Reset(1, 0);

Nmc_Command	Execute the command of a specified axis. (Write the command of a specified axis into WR0.) **VC** void Nmc_Command(int No, int IcNo, int axis, int cmd); **VB.NET** Sub Nmc_Command(ByVal No As Integer, ByVal IcNo As Integer, ByVal Axis As Integer, ByVal cmd As Integer) **C#** void MC8000P.Nmc_Command(int No, int IcNo, AXIS axis, CMD cmd); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. Axis Axis to execute the command. Multiple axes can be assigned. See 4.1.3 (2) for more details. cmd Command number. Specify one command from “Driving commands, Other commands” of “Command Definition” described in definition file (*1). For + direction fixed pulse drive, specify CMD_F_DRV_P. See 4.1.3 (4) for C#. *1: [VC] MC8000P_DLL.H, [VB.NET] MC8000P_DLL.vb **Return Value** None **Example** [VC] Nmc_Command(No, IcNo, AXIS_X, CMD_F_DRV_P); // Execute the + direction fixed drive of X axis. [VB.NET] Call Nmc_Command(No, IcNo, AXIS_X, CMD_F_DRV_P) [C#] MC8000P.Nmc_Command(No, IcNo, AXIS.X, CMD.CMD_F_DRV_P);

Function Name	Function and Content

Nmc_Command_IP	Execute interpolation command. (Write interpolation command into WR0.) *MCX314As only
	
	VC void Nmc_Command_IP(int No, int IcNo, int cmd);
	VB.NET Sub Nmc_Command_IP(ByVal No As Integer, ByVal IcNo As Integer, ByVal cmd As Integer
	C# void MC8000P.Nmc_Command_IP(int No, int IcNo, CMD cmd);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1).
	Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	cmd Command number. Specify one command from “Interpolation Commands” of “Command Definition” described in definition file (*1).
	For 2-axis linear interpolation drive, specify CMD_IP_2ST. See 4.1.3 (4) for C#.
	*1: [VC] MC8000P_DLL.H, [VB.NET] MC8000P_DLL.vb
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg5(No, IcNo, 0x0004);
	// Set the interpolation axis (Main axis: X, Second axis: Y).
	Nmc_Command_IP(No, IcNo, CMD_IP_2ST); // Execute 2-axis linear interpolation drive.
	[VB.NET] Call Nmc_WriteReg5(No, IcNo, &H0004)
	Call Nmc_Command_IP(No, IcNo, CMD_IP_2ST)
	[C#] MC8000P.Nmc_WriteReg5(No, IcNo, 0x0004);
	MC8000P.Nmc_Command_IP(No, IcNo, CMD.CMD_IP_2ST);

Nmc_WriteReg0	Write data into WR0 (Command register).
	
	VC void Nmc_WriteReg0(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_WriteReg0(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg0(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg0(No, IcNo, 0x0120); // Execute the + direction fixed drive of X axis.
	[VB.NET] Call Nmc_WriteReg0(No, IcNo, &H120)
	[C#] MC8000P.Nmc_WriteReg0(No, IcNo, 0x0120);

Function Name	Function and Content

Nmc_WriteReg1	Write data into WR1 (Mode register 1).
	
	VC void Nmc_WriteReg1(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_WriteReg1(ByVal No As Integer, ByVal IcNo As Integer, ByVal Axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg1(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to write data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg1(No, IcNo, AXIS_X, 0x8000); // Interrupt occurs at the stop (X axis).
	[VB.NET] Call Nmc_WriteReg1(No, IcNo, AXIS_X, &H8000)
	[C#] MC8000P.Nmc_WriteReg1(No, IcNo, AXIS.X, 0x8000);

Nmc_WriteReg2	Write data into WR2 (Mode register 2).
	
	VC void Nmc_WriteReg2(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_WriteReg2(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg2(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to write data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg2(No, IcNo, AXIS_Y, 0x2000); // Enable ALARM (Y axis).
	[VB.NET] Call Nmc_WriteReg2(No, IcNo, AXIS_Y, &H2000)
	[C#] MC8000P.Nmc_WriteReg2(No, IcNo, AXIS.Y, 0x2000);

Function Name	Function and Content

Nmc_WriteReg3	Write data into WR3 (Mode register 3).
	
	VC void Nmc_WriteReg3(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_WriteReg3(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg3(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to write data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg3(No, IcNo, AXIS_ALL, 0x0001); // Manual decelerating for all axes.
	[VB.NET] Call Nmc_WriteReg3(No, IcNo, AXIS_ALL, &H1)
	[C#] MC8000P.Nmc_WriteReg3(No, IcNo, AXIS.ALL, 0x0001);

Nmc_WriteReg4	Write data into WR4 (Output register).
	
	VC void Nmc_WriteReg4(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_WriteReg4(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg4(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg4(No, IcNo, 0x0001); // Hi level output for X axis general output OUT0.
	[VB.NET] Call Nmc_WriteReg4(No, IcNo, &H0001)
	[C#] MC8000P.Nmc_WriteReg4(No, IcNo, 0x0001);

Function Name	Function and Content

Nmc_WriteReg5	Write data into WR5.
	
	VC void Nmc_WriteReg5(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_WriteReg5(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg5(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	■ MCX304
	[VC] Nmc_WriteReg5(No, IcNo, 0x0001); // Enable X axis general output OUT0.
	[VB.NET] Call Nmc_WriteReg5(No, IcNo, &H1)
	[C#] MC8000P.Nmc_WriteReg5(No, IcNo, 0x0001);
	■ MCX314As
	[VC] Nmc_WriteReg5(No, IcNo, 0x0024);
	// Set the interpolation axis (Main axis: X, Second axis: Y, Third axis: Z).
	[VB.NET] Call Nmc_WriteReg5(No, IcNo, &H0024)
	[C#] MC8000P.Nmc_WriteReg5(No, IcNo, 0x0024);

Nmc_WriteReg6	Write data into WR6 (Write data register 1).
	
	VC void Nmc_WriteReg6(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_WriteReg6(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg6(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg6(No, IcNo, 0x1234); // Write data (1234)H into write data register 1.
	[VB.NET] Call Nmc_WriteReg6(No, IcNo, &H1234)
	[C#] MC8000P.Nmc_WriteReg6(No, IcNo, 0x1234);

Function Name	Function and Content

Nmc_WriteReg7	Write data into WR7 (Write data register 2).
	
	VC void Nmc_WriteReg7(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_WriteReg7(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg7(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1).
	Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg7(No, IcNo, 0x5678); // Write data (5678)H into write data register 2.
	[VB.NET] Call Nmc_WriteReg7(No, IcNo, &H5678)
	[C#] MC8000P.Nmc_WriteReg7(No, IcNo, 0x5678);

Nmc_ReadReg0	Read out data from RR0 (Main status register).
	
	VC long Nmc_ReadReg0(int No, int IcNo);
	VB.NET Function Nmc_ReadReg0(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg0(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	
	Return Value
	The data of RR0 (Main status register)
	
	Example
	[VC] Data = Nmc_ReadReg0(No, IcNo); // Read out RR0.
	[VB.NET] Data = Nmc_ReadReg0(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadReg0(No, IcNo);

Function Name	Function and Content

Nmc_ReadReg1	Read out data from RR1 (Status register 1).
	
	VC long Nmc_ReadReg1(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadReg1(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer As Integer
	C# int MC8000P.Nmc_ReadReg1(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X-axis, AXIS_Y for Y-axis, AXIS_Z for Z-axis and AXIS_U for U-axis.See 4.1.3 (2) for more details.
	
	Return Value
	The data of RR1 (Status register 1)
	
	Example
	[VC] Data = Nmc_ReadReg1(No, IcNo, AXIS_X); // Read out RR1 of X axis.
	[VB.NET] Data = Nmc_ReadReg1(No, IcNo, AXIS_X)
	[C#] Data = MC8000P.Nmc_ReadReg1(No, IcNo, AXIS.X);

Nmc_ReadReg2	Read out data from RR2 (Status register 2).
	
	VC long Nmc_ReadReg2(int No, int IcNo, int Axis);
	VB.NET Function Nmc_ReadReg2(ByVal No As Integer, ByVal IcNo As Integer, ByVal Axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg2(int No, int IcNo, AXIS Axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Specify AXIS_X for X-axis, AXIS_Y for Y-axis, AXIS_Z for Z-axis and AXIS_U for U-axis. See 4.1.3 (2) for more details.
	
	Return Value
	The data of RR2 (Status register 2)
	
	Example
	[VC] Data = Nmc_ReadReg2(No, IcNo, AXIS_Y); // Read out RR2 of Y axis.
	[VB.NET] Data = Nmc_ReadReg2(No, IcNo, AXIS_Y)
	[C#] Data = MC8000P.Nmc_ReadReg2(No, IcNo, AXIS.Y);

Nmc_ReadReg4	Read out data from RR4 (Input register 1).
	
	VC long Nmc_ReadReg4(int No, int IcNo);
	VB.NET Function Nmc_ReadReg4(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg4(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	
	Return Value
	The data of RR4 (Input register 1)
	
	Example
	[VC] Data = Nmc_ReadReg4(No, IcNo); // Read out RR4.
	[VB.NET] Data = Nmc_ReadReg4(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadReg4(No, IcNo);

Function Name	Function and Content

Nmc_ReadReg5	Read out data from RR5 (Input register 2).
	
	VC long Nmc_ReadReg5(int No, int IcNo);
	VB.NET Function Nmc_ReadReg5(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg5(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	
	Return Value
	The data of RR5 (Input register 2)
	
	Example
	[VC] Data = Nmc_ReadReg5(No, IcNo); // Read out RR5.
	[VB.NET] Data = Nmc_ReadReg5(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadReg5(No, IcNo);

Nmc_ReadReg6	Read out data from RR6 (Read data register 1).
	
	VC long Nmc_ReadReg6(int No, int IcNo);
	VB.NET Function Nmc_ReadReg6(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg6(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	
	Return Value
	The data of RR6 (Read data register 1)
	
	Example
	[VC] Data = Nmc_ReadReg6(No, IcNo); // Read out RR6.
	[VB.NET] Data = Nmc_ReadReg6(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadReg6(No, int IcNo);

Nmc_ReadReg7	Read out data from RR7 (Read data register 2).
	
	VC long Nmc_ReadReg7(int No, int IcNo);
	VB.NET Function Nmc_ReadReg7(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg7(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	
	Return Value
	The data of RR7 (Read data register 2)
	
	Example
	[VC] Data = Nmc_ReadReg7(No, IcNo); // Read out RR7.
	[VB.NET] Data = Nmc_ReadReg7(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadReg7(No, IcNo);

Function Name	Function and Content
Nmc_Range	Set the range. **VC** void Nmc_Range(int No, int IcNo, int axis, long wdata); **VB.NET** Sub Nmc_Range(ByVal No As Integer, ByVal IcNo As Integer, ByVal Axis As Integer, ByVal wdata As Integer) **C#** void MC8000P.Nmc_Range(int No, int IcNo, AXIS axis, int wdata); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details. wdata Data to be set. **Return Value** None **Example** [VC] Nmc_Range(No, IcNo, AXIS_ALL, 800000); // Set 800000 (Multiple = 10) to range (All axes). [VB.NET] Call Nmc_Range(No, IcNo, AXIS_ALL, 800000) [C#] MC8000P.Nmc_Range(No, IcNo, AXIS.ALL, 800000);

Nmc_Jerk	Set jerk.
	
	VC void Nmc_Jerk(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Jerk(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Jerk(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Jerk(No, IcNo, AXIS_X, 1000); // Set 1000 as jerk (X axis).
	[VB.NET] Call Nmc_Jerk(No, IcNo, AXIS_X, 1000)
	[C#] MC8000P.Nmc_Jerk(No, IcNo, AXIS.X, 1000);

Function Name	Function and Content

Nmc_Acc	Set acceleration.
	
	VC void Nmc_Acc(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Acc(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Acc(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Acc(No, IcNo, AXIS_Y, 100); // Set 100 as acceleration (Y axis).
	[VB.NET] Call Nmc_Acc(No, IcNo, AXIS_Y, 100)
	[C#] MC8000P.Nmc_Acc(No, IcNo, AXIS.Y, 100);

Nmc_Dec	Set deceleration.
	
	VC void Nmc_Dec(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Dec(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Dec(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Dec(No, IcNo, AXIS_Z, 100); // Set 100 as deceleration (Z axis).
	[VB.NET] Call Nmc_Dec(No, IcNo, AXIS_Z, 100)
	[C#] MC8000P.Nmc_Dec(No, IcNo, AXIS.Z, 100);

Function Name	Function and Content

Nmc_StartSpd	Set initial speed.
	
	VC void Nmc_StartSpd(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_StartSpd(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_StartSpd(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_StartSpd(No, IcNo, AXIS_U, 100); // Set 100 as initial speed (U axis).
	[VB.NET] Call Nmc_StartSpd(No, IcNo, AXIS_U, 100)
	[C#] MC8000P.Nmc_StartSpd(No, IcNo, AXIS.U, 100);

Nmc_Speed	Set drive speed.
	
	VC void Nmc_Speed(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Speed(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Speed(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Speed(No, IcNo, AXIS_X | AXIS_Y, 1000); // Set 1000 as drive speed (X/Y axes).
	[VB.NET] Call Nmc_Speed(No, IcNo, AXIS_X Or AXIS_Y, 1000)
	[C#] MC8000P.Nmc_Speed(No, IcNo, AXIS.X | AXIS.Y, 1000);

Function Name	Function and Content

Nmc_Pulse	Set output pulse number or interpolation finish point. (VC, C# only) *MCX304 has no interpolation function.
	
	The number of output pulses indicates the total number of pulses that are output in fixed pulse driving.
	For linear and circular interpolation driving, set the finish point of each axis.
	The finish point should be set the relative numbers to the current position.
	The output pulse number is unsigned 32-bit value. The interpolation finish point is signed 32-bit value.
	
	VC void Nmc_Pulse(int No, int IcNo, int axis, long wdata);
	VB.NET cannot be used.
	C# To set the interpolation finish point (P)
	void MC8000P.Nmc_Pulse(int No, int IcNo, AXIS axis, int wdata);
	To set the output pulse number
	void MC8000P.Nmc_Pulse(int No, int IcNo, AXIS axis, uint wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Pulse(No, IcNo, AXIS_X, 2000); // Set 2000 as output pulse number (X axis).
	Nmc_Pulse(No, IcNo, AXIS_Y, 300); // Set 300 as interpolation finish point (Y axis).
	Nmc_Pulse(No, IcNo, AXIS_Z, -400); // Set -400 as interpolation finish point (Z axis).
	[C#] MC8000P.Nmc_Pulse(No, IcNo, AXIS.X, 2000);
	MC8000P.Nmc_Pulse(No, IcNo, AXIS.Y, 300);
	MC8000P.Nmc_Pulse(No, IcNo, AXIS.Z, -400);

Nmc_Pulse_VB	Set output pulse number or interpolation finish point. (VB.NET only) *MCX304 has no interpolation function. The number of output pulses indicates the total number of pulses that are output in fixed pulse driving. For linear and circular interpolation driving, set the finish point of each axis. The finish point should be set the relative numbers to the current position. **VC** cannot be used. **VB.NET** Sub Nmc_Pulse_VB(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Double) **C#.NET** cannot be used. **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details. wdata Data to be set. **Return Value** None **Example** [VB.NET] Call Nmc_Pulse_VB(No, IcNo, AXIS_X, 2000) ‘ Set 2000 as output pulse number (X axis). Call Nmc_Pulse_VB(No, IcNo, AXIS_Y, 300) ‘ Set 300 as interpolation finish point (Y axis). Call Nmc_Pulse_VB(No, IcNo, AXIS_Z, -400) ‘ Set -400 as interpolation finish point (Z axis)

Function Name	Function and Content

Nmc_DecP	Set manual decelerating point. (VC, C# only)
	
	VC void Nmc_DecP(int No, int IcNo, int axis, ULONG wdata);
	VB.NET cannot be used.
	C# void MC8000P.Nmc_DecP(int No, int IcNo, AXIS axis, uint wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_DecP(No, IcNo, AXIS_U, 30000); // Set 30000 as manual decelerating point (U axis).
	[C#] MC8000P.Nmc_DecP(No, IcNo, AXIS.U, 30000);

Nmc_DecP_VB	Set manual decelerating point. (VB.NET only) **VC** cannot be used. **VB.NET** Sub Nmc_DecP_VB(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Double) **C#** cannot be used. **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details. wdata Data to be set. **Return Value** None **Example** [VB.NET] Call Nmc_DecP_VB(No, IcNo, AXIS_X, 40000) ‘ Set 40000 as manual decelerating point (X axis).

Function Name	Function and Content

Nmc_Center	Set circular center point. *MCX314As only
	
	VC void Nmc_Center(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Center(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Center(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Center(No, IcNo, AXIS_Y, 1500); // Set 1500 as circular center point (Y axis).
	[VB.NET] Call Nmc_Center(No, IcNo, AXIS_Y, 1500)
	[C#] MC8000P.Nmc_Center(No, IcNo, AXIS.Y, 1500);

Nmc_Lp	Set logical position counter.
	
	VC void Nmc_Lp(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Lp(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Lp(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Lp(No, IcNo, AXIS_ALL, 0); // Set 0 to logical position counter of all axes.
	[VB.NET] Call Nmc_Lp(No, IcNo, AXIS_ALL, 0)
	[C#] MC8000P.Nmc_Lp(No, IcNo, AXIS.ALL, 0);

Function Name	Function and Content

Nmc_Ep	Set real position counter.
	
	VC void Nmc_Ep(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Ep(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Ep(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Ep(No, IcNo, AXIS_ALL, 0); // Set 0 to real position counter of all axes.
	[VB.NET] Call Nmc_Ep(No, IcNo, AXIS_ALL, 0)
	[C#] MC8000P.Nmc_Ep(No, IcNo, AXIS.ALL, 0);

Nmc_CompP	Set COMP+ register.
	
	VC void Nmc_CompP(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_CompP(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_CompP(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_CompP(No, IcNo, AXIS_X, 50000); // Set 50000 as COMP+ register (X axis).
	[VB.NET] Call Nmc_CompP(No, IcNo, AXIS_X, 50000)
	[C#] MC8000P.Nmc_CompP(No, IcNo, AXIS.X, 50000);

Function Name	Function and Content

Nmc_CompM	Set COMP– register.
	
	VC void Nmc_CompM(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_CompM(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_CompM(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_CompM(No, IcNo, AXIS_X, -50000); // Set -50000 as COMP- register (X axis).
	[VB.NET] Call Nmc_CompM(No, IcNo, AXIS_X, -50000)
	[C#] MC8000P.Nmc_CompM(No, IcNo, AXIS.X, -50000);

Nmc_AccOfst	Set acceleration counter offsetting.
	
	VC void Nmc_AccOfst(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_AccOfst(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_AccOfst(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_AccOfst(No, IcNo, AXIS_Y, 20); // Set 20 as acceleration counter offsetting (Y axis).
	[VB.NET] Call Nmc_AccOfst(No, IcNo, AXIS_Y, 20)
	[C#] MC8000P.Nmc_AccOfst(No, IcNo, AXIS.Y, 20);

Function Name	Function and Content

Nmc_DJerk	Set deceleration increasing rate. *MCX314As only
	
	VC void Nmc_DJerk(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_DJerk(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_DJerk(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_DJerk(No, IcNo, AXIS_Z, 1000); // Set 1000 as deceleration increasing rate (Z axis).
	[VB.NET] Call Nmc_DJerk(No, IcNo, AXIS_Z, 1000)
	[C#] MC8000P.Nmc_DJerk(No, IcNo, AXIS.Z, 1000);

Nmc_HomeSpd	Set home search speed.
	
	VC void Nmc_HomeSpd(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_HomeSpd(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_HomeSpd(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_HomeSpd(No, IcNo, AXIS_U, 200); // Set 200 as home search speed (U axis).
	[VB.NET] Call Nmc_HomeSpd(No, IcNo, AXIS_U, 200)
	[C#] MC8000P.Nmc_HomeSpd(No, IcNo, AXIS.U, 200);

<table>
<tr><th>Function Name</th><th>Function and Content</th></tr>
<tr><td>Nmc_ExpMode</td><td>Set extension mode. *MCX314As only

VC void Nmc_ExpMode(int No, int IcNo, int axis, long EM6_data, long EM7_data);

VB.NET Sub Nmc_ExpMode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal EM6_data As Integer, ByVal EM7_data As Integer)
C# void MC8000P.Nmc_ExpMode(int No, int IcNo, AXIS axis, int EM6_data, int EM7_data);

Input Parameter
No Board number (setting value of rotary switch (0~15) on the board)
IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
EM6_data Data to be set in extension mode register EM6.
EM7_data Data to be set in extension mode register EM7.

Return Value
None

Example Set enable for all filters, delay time 512μs and execution of automatic home search step1, 2 and 4 to X axis extension mode.
[VC] Nmc_ExpMode(No, IcNo, AXIS_X, 0x5F00, 0x0045);
[VB.NET] Call Nmc_ExpMode(No, IcNo, AXIS_X, &H5F00, &H0045)
[C#] MC8000P.Nmc_ExpMode(No, IcNo, AXIS.X, 0x5F00, 0x0045);</td></tr>
<tr><td>Nmc_SyncMode</td><td>Set synchronous action mode. *MCX314As only

VC void Nmc_SyncMode(int No, int IcNo, int axis, long SM6_data, long SM7_data);
VB.NET Sub Nmc_SyncMode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal SM6_data As Integer, ByVal SM7_data As Integer)
C# void MC8000P.Nmc_SyncMode(int No, int IcNo, AXIS axis, int SM6_data, int SM7_data);

Input Parameter
No Board number (setting value of rotary switch (0~15) on the board)
IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details.
SM6_data Data to be set in synchronous action mode register SM6.
SM7_data Data to be set in synchronous action mode register SM7.

Return Value
None

Example Set “start fixed pulse driving in the + direction of Y axis at the termination of X axis driving”.
(1) Set Y axis activation by the activation factor D-END to X axis synchronous action mode.
(2) Set fixed pulse driving in the + direction (FDRV+) as action to Y axis synchronous action mode.

[VC] (1) Nmc_SyncMode(No, IcNo, AXIS_X, 0x2020, 0);
(2) Nmc_SyncMode(No, IcNo, AXIS_Y, 0, 0x0001);
[VB.NET] (1) Call Nmc_SyncMode(No, IcNo, AXIS_X, &H2020, 0)
(2) Call Nmc_SyncMode(No, IcNo, AXIS_Y, 0, &H0001)
[C#] (1) MC8000P.Nmc_SyncMode(No, IcNo, AXIS.X, 0x2020, 0);
(2) MC8000P.Nmc_SyncMode(No, IcNo, AXIS.Y, 0, 0x0001);</td></tr>
</table>

Function Name	Function and Content
Nmc_HomeMode	Set automatic home search mode. *MCX304 only **VC** void Nmc_HomeMode(int No, int IcNo, int axis, long WR6_data); **VB.NET** Sub Nmc_HomeMode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer) **C#** void MC8000P.Nmc_HomeMode(int No, int IcNo, AXIS axis, int WR6_data); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See 4.1.3 (2) for more details. WR6_data Data to be set to WR6. **Return Value** None **Example** Set execution of automatic home search step1, 2 and 4 to X axis automatic home search mode. [VC] Nmc_HomeMode(No, IcNo, AXIS_X, 0x0045); [VB.NET] Call Nmc_HomeMode(No, IcNo, AXIS_X, &H0045) [C#] MC8000P.Nmc_HomeMode(No, IcNo, AXIS.X, 0x0045);

Nmc_ReadLp	Read out logical position counter.
	
	VC long Nmc_ReadLp(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadLp(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadLp(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X-axis, AXIS_Y for Y-axis, AXIS_Z for Z-axis and AXIS_U for U-axis.
	
	Return Value
	The value of the current logical position counter
	Example
	[VC] Data = Nmc_ReadLp(No, IcNo, AXIS_X); // Read out logical position counter of X axis.
	[VB.NET] Data = Nmc_ReadLp(No, IcNo, AXIS_X)
	[C#] Data = MC8000P.Nmc_ReadLp(No, IcNo, AXIS.X);

Nmc_ReadEp	Read out real position counter.
	
	VC long Nmc_ReadEp(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadEp(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadEp(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X-axis, AXIS_Y for Y-axis, AXIS_Z for Z-axis and AXIS_U for U-axis.
	
	Return Value
	The value of the current real position counter
	Example
	[VC] Data = Nmc_ReadEp(No, IcNo, AXIS_Y); // Read out real position counter of Y axis.
	[VB.NET] Data = Nmc_ReadEp(No, IcNo, AXIS_Y)
	[C#] Data = MC8000P.Nmc_ReadEp(No, IcNo, AXIS.Y)

<table>
<tr><th>Function Name</th><th>Function and Content</th></tr>
<tr><td>Nmc_ReadSpeed</td><td>Read out the current drive speed.

VC long Nmc_ReadSpeed(int No, int IcNo, int axis);
VB.NET Function Nmc_ReadSpeed(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer As Integer
C# int MC8000P.Nmc_ReadSpeed(int No, int IcNo, AXIS axis);

Input Parameter
No Board number (setting value of rotary switch (0~15) on the board)
IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
Axis Axis to read data. Specify AXIS_X for X-axis, AXIS_Y for Y-axis, AXIS_Z for Z-axis, AXIS_U for U-axis.

Return Value
The current drive speed.

Example
[VC] Data = Nmc_ReadSpeed(No, IcNo, AXIS_Z); // Read out the current drive speed of Z axis.
[VB.NET] Data = Nmc_ReadSpeed(No, IcNo, AXIS_Z)
[C#] Data = MC8000P.Nmc_ReadSpeed(No, IcNo, AXIS.Z);</td></tr>
<tr><td>Nmc_ReadAccDec</td><td>Read out the current acceleration/deceleration.

Read out the value of the current acceleration or deceleration during driving.
When the driving stops, the read data is random number.

VC long Nmc_ReadAccDec(int No, int IcNo, int Axis);
VB.NET Function Nmc_ReadAccDec(ByVal No As Integer, ByVal IcNo As Integer,
ByVal Axis As Integer) As Integer
C# int MC8000P.Nmc_ReadAccDec(int No, int IcNo, AXIS Axis);

Input Parameter
No Board number (setting value of rotary switch (0~15) on the board)
IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
Axis Axis to read data. Specify AXIS_X for X-axis, AXIS_Y for Y-axis, AXIS_Z for Z-axis, AXIS_U for U-axis. See 4.1.3 (2) for more details.

Return Value
The current acceleration/deceleration

Example
[VC] Data = Nmc_ReadAccDec(No, IcNo, AXIS_U);
// Read out the current acceleration/deceleration of U axis.
[VB.NET] Data = Nmc_ReadAccDec(No, IcNo, AXIS_U)
[C#] Data = MC8000P.Nmc_ReadAccDec(No, IcNo, AXIS.U);</td></tr>
</table>

Function Name	Function and Content
Nmc_ReadSyncBuff	Read out synchronous action buffer register. *MCX314As only **VC** long Nmc_ReadSyncBuff(int No, int IcNo, int axis); **VB.NET** Function Nmc_ReadSyncBuff(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer **C#** int MC8000P.Nmc_ReadSyncBuff(int No, int IcNo, AXIS axis); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. Axis Axis to read data. Specify AXIS_X for X-axis, AXIS_Y for Y-axis, AXIS_Z for Z-axis, AXIS_U for U-axis. See 4.1.3 (2) for more details. See 4.1.3 (2) for more details. **Return Value** The value of synchronous buffer register **Example** [VC] Data = Nmc_ReadSyncBuff(No, IcNo, AXIS_X); // Read out synchronous action buffer register of X axis. [VB.NET] Data = Nmc_ReadSyncBuff(No, IcNo, AXIS_X) [C#] Data = MC8000P.Nmc_ReadSyncBuff(No, IcNo, AXIS.X);

Nmc_GetDriveStatus	Read drive status.
	The user can use to check whether the driving of the specified axis has finished or not.
	
	VC int Nmc_GetDriveStatus(int No, int IcNo, int axis);
	VB.NET Function Nmc_GetDriveStatus(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P. Nmc_GetDriveStatus(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	Axis Axis to read drive status. Multiple axes can be assigned. See 4.1.3 (2) for more details.
	Return Value
	If all the specified axes have finished driving, the return value is 0.
	If more than one of the specified axes is/are driving, the return value is nonzero.
	
	Example
	[VC] if(Nmc_GetDriveStatus(No, IcNo, AXIS_X) == 0) // When X axis has finished driving.
	AfxMessageBox("X axis has finished driving");
	else
	AfxMessageBox("X axis is driving");
	[VB.NET] If Nmc_GetDriveStatus(No, IcNo, AXIS_X) = 0 Then
	Call MsgBox("X axis has finished driving")
	Else
	Call MsgBox("X axis is driving")
	End If
	[C#] if(MC8000P.Nmc_GetDriveStatus(No, IcNo, AXIS.X) == 0)
	MessageBox.Show("X axis has finished driving ");
	else
	MessageBox.Show("X axis is driving");

Function Name	Function and Content

Nmc_GetCNextStatus	Read the status of ready signal for writing of continuous interpolation. *MCX314As only
	The user can use to check whether the signal for the writing of continuous interpolation is ready or not during continuous interpolation execution.
	
	VC int Nmc_GetCNextStatus(int No, int IcNo);
	VB.NET Function Nmc_GetCNextStatus(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_GetCNextStatus(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	
	Return Value
	If the signal for the writing of continuous interpolation is ready, the return value is nonzero.
	If the signal for the writing of continuous interpolation is not ready, the return value is 0.
	
	Example
	[VC] if(Nmc_GetCNextStatus(No, IcNo) != 0) // When the signal for the writing is ready.
	AfxMessageBox("The signal for the writing of continuous interpolation is ready");
	else
	AfxMessageBox("The signal for the writing of continuous interpolation is not ready");
	[VB.NET] If Nmc_GetCNextStatus(No, IcNo) <> 0 Then
	Call MsgBox("The signal for the writing of continuous interpolation is ready")
	Else
	Call MsgBox("The signal for the writing of continuous interpolation is not ready")
	End If
	[C#] if(MC8000P.Nmc_GetCNextStatus(No, IcNo) != 0)
	MessageBox.Show("The signal for the writing of continuous interpolation is ready");
	else
	MessageBox.Show("The signal for the writing of continuous interpolation is not ready");

Nmc_GetBpSc	Read the value of BP interpolation stack counter. *MCX314As only
	
	VC int Nmc_GetBpSc(int No, int IcNo);
	VB.NET Function Nmc_GetBpSc(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_GetBpSc(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	
	Return Value
	The value of the current bit pattern interpolation stack counter.
	
	Example
	[VC] Data = Nmc_GetBpSc(No, IcNo); // Read the value of BP interpolation stack counter.
	[VB.NET] Data = Nmc_GetBpSc(No, IcNo)
	[C#] Data = MC8000P.Nmc_GetBpSc(No, IcNo);

Function Name	Function and Content
Nmc_WriteRegSetAxis	Write data into one specified write register of WR1~WR3 for the specified axis. **VC** void Nmc_WriteRegSetAxis(int No, int IcNo, int axis, int RegNumber, long wdata); **VB.NET** Sub Nmc_WriteRegSetAxis(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer,ByVal RegNumber As Integer, ByVal wdata As Integer) **C#** void MC8000P.Nmc_WriteRegSetAxis(int No, int IcNo, AXIS axis, int RegNumber, int wdata); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. Axis Axis to write data. Multiple axes can be assigned. See 4.1.3 (2) for more details. RegNumber Write register number to write data. [VC][VB.NET] Assign MCX_RR1 for RR1 and MCX_RR2 for RR2. [C#] Assign REG_MCX.RR1 for RR1 and REG_MCX.RR2 for RR2. See 4.1.3 (1). wdata Data to be written. **Return Value** None **Example** Write ALARM Enable (2000)H into WR2 of all axes. [VC] Nmc_WriteRegSetAxis(No, IcNo, AXIS_ALL, MCX_WR2, 0x2000); [VB.NET] Call Nmc_WriteRegSetAxis(No, IcNo, AXIS_ALL, MCX_WR2, &H2000) [C#] MC8000P.Nmc_WriteRegSetAxis(No, IcNo, AXIS.ALL, REG_MCX.WR2, 0x2000);
Nmc_ReadRegSetAxis	Read out data from the specified read register (either RR1 or RR2) for the specified axis. **VC** long Nmc_ReadRegSetAxis(int No, int IcNo, int axis, int RegNumber); **VB.NET** Function Nmc_ReadRegSetAxis(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal RegNumber As Integer) As Integer **C#** int MC8000P.Nmc_ReadRegSetAxis(int No, int IcNo, AXIS axis, int RegNumber); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. Axis Axis to read data. Specify AXIS_X for X-axis, AXIS_Y for Y-axis, AXIS_Z for Z-axis, AXIS_U for U-axis. See 4.1.3 (2) for more details. RegNumber Read register number to read data. [VC][VB.NET] Assign MCX_RR1 for RR1 and MCX_RR2 for RR2. [C#] Assign REG_MCX.RR1 for RR1 and REG_MCX.RR2 for RR2. See 4.1.3 (1). **Return Value** The data of the specified read register for the specified axis. **Example** Read out the data of X axis RR1. [VC] Data = Nmc_ReadRegSetAxis(No, IcNo, AXIS_X, MCX_RR1); [VB.NET] Data = Nmc_ReadRegSetAxis(No, IcNo, AXIS_X, MCX_RR1) [C#] Data = MC8000P.Nmc_ReadRegSetAxis(No, IcNo, AXIS.X, REG_MCX.RR1);

Function Name	Function and Content
Nmc_WriteData	Write the specified parameter into the specified axis. (Execute commands for data writing) **VC** void Nmc_WriteData(int No, int IcNo, int axis, int cmd, long wdata); **VB.NET** Sub Nmc_WriteData(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal cmd As Integer, ByVal wdata As Integer) **C#** void MC8000P.Nmc_WriteData(int No, int IcNo, AXIS axis, CMD cmd, int wdata); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. Axis Axis to write data. Multiple axes can be assigned. See 4.1.3 (2) for more details. cmd Commands for data writing ((00)H~(0E)H, (61)H). e.g. Range setting is (00)H. *(08)H and (0E)H are excluded for MCX304. wdata Data to be written. **Return Value** None **Example** Set 1000 as drive speed of all axes. Drive speed command code is (05)H. [VC] Nmc_WriteData(No, IcNo, AXIS_ALL, 0x05, 1000); [VB.NET] Call Nmc_WriteData(No, IcNo, AXIS_ALL, &H05, 1000) [C#] MC8000P.Nmc_WriteData(No, IcNo, AXIS.ALL, 0x05, 1000);
Nmc_WriteData2	Write the data of extension mode or synchronous action mode into the specified axis. *MCX314As only (Execute commands for data writing) **VC** void Nmc_WriteData2(int No, int IcNo, int axis, int cmd, long WR6_data, long WR7_data); **VB.NET** Sub Nmc_WriteData2(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal cmd As Integer, ByVal WR6_data As Integer, ByVal WR7_data As Integer) **C#** void MC8000P.Nmc_WriteData2(int No, int IcNo, AXIS axis, CMD cmd, int WR6_data, int WR7_data); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. Axis Axis to be written data into. Multiple axes can be assigned. See 4.1.3 (2) for more details. cmd Commands for data writing. Specify (60)H for extension mode, and (64)H for synchronous action mode. WR6_data Data to be written into EM6 in extension mode and into SM6 in synchronous action mode. WR7_data Data to be written into EM7 in extension mode and into SM7 in synchronous action mode. **Return Value** None **Example** Write EM6 data (5F00)H and EM7 data (45)H into extension mode of X axis. [VC] Nmc_WriteData2(No, IcNo, AXIS_X, 0x60, 0x5F00, 0x0045); [VB.NET] Call Nmc_WriteData2(No, IcNo, AXIS_X, &H60, &H5F00, &H45) [C#] MC8000P.Nmc_WriteData2(No, IcNo, AXIS.X, 0x60, 0x5F00, 0x0045);

<table>
<tr><th>Function Name</th><th>Function and Content</th></tr>
<tr><td>Nmc_ReadData</td><td>
Read out data by executing commands for reading data.

VC long Nmc_ReadData(int No, int IcNo, int Axis, int cmd);
VB.NET Function Nmc_ReadData(ByVal No As Integer, ByVal IcNo As Integer,
ByVal Axis As Integer, ByVal cmd As Integer) As Integer
C# int MC8000P.Nmc_ReadData(int No, int IcNo, AXIS Axis, CMD cmd);

Input Parameter
No Board number (setting value of rotary switch (0~15) on the board)
IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
Axis Axis to read data. Specify AXIS_X for X-axis, AXIS_Y for Y-axis, AXIS_Z for Z-axis, AXIS_U for U-axis. See 4.1.3 (2) for more details.
cmd Commands for reading data ((10)H~(14)H).
e.g. Logical position counter reading is (10)H. *(14)H is excluded for MCX304.

Return Value
Data to be read.

Example
[VC] Data = Nmc_ReadData(No, IcNo, AXIS_X, 0x10);
// Read logical position counter of X axis.

[VB.NET] Data = Nmc_ReadData(No, IcNo, AXIS_X, &H10)
[C#] Data = MC8000P.Nmc_ReadData(No, IcNo, AXIS.X, 0x10);
</td></tr>
<tr><td>Nmc_2BPExec</td><td>
Execute 2-axis bit pattern interpolation using the specified interpolation data. *MCX314As only
This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread.

VC DWORD Nmc_2BPExec(int No, int IcNo, DATA_2BP* pData2Bp, int DataCnt, int IpAxis,
BOOL ContinueFlg = FALSE);

VB.NET Function Nmc_2BPExec(ByVal No As Integer, ByVal IcNo As Integer,
ByRef pData2Bp As DATA_2BP, ByVal DataCnt As Integer,
ByVal IpAxis As Integer, ByVal ContinueFlg As Integer) As Integer

C# Nmc_Status MC8000P.Nmc_2BPExec(int No, int IcNo, DATA_2BP[] pData2Bp, int DataCnt,
int IpAxis, bool ContinueFlg);

Input Parameter
No Board number (setting value of rotary switch (0~15) on the board)
IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
pData2Bp Pointer to an array of DATA_2BP structures (user-defined type in VB).
Set the 2-axis BP interpolation data to DATA_2BP.
See 4.1.3 (3) for DATA_2BP.
DataCnt The number of 2-axis BP interpolation data. Specify the number of the DATA_2BP structure (user-defined type) array.
IpAxis Axis to execute interpolation. Specify the same value as the setting value of WR5 D0~D5 (Axis assignment). See 4.1.3 (4).
ContinueFlg Set the flag to continue when BP interpolation stopped during driving (because the driving speed is too fast to stack the next data).
[VC] TRUE: Continue, FALSE: Not continue Can be omitted. Default is FALSE.
[VB.NET] True: Continue, False: Not continue
[C#] true: Continue, false: Not continue

Return Value
If the function succeeds, the return value is BP_END.
If the function fails, the return value is the following Error code.
For C#, See 4.1.3 (1).

■ Normal end
BP_END BP interpolation has been successfully completed.

■ Error code
BP_CNT_ERR The number of the specified data is out of range.
BP_ALREADY_EXEC BP interpolation or continuous interpolation is already running.
BP_PARAM_ERR The parameter is incorrect.
BP_NOT_OPEN_ERR The specified board is not opened.
BP_OTHER_ERR Other errors.
</td></tr>
</table>

	BP_STOP BP interpolation stopped during driving (too fast to stack next data).
	BP_USER_STOP The user aborted BP interpolation.
	BP_DRIVE_ERR Error occurred in the board during BP interpolation.
	(When the error status was set to RR0.)
	
	Example
	[VC] // 2-axis BP interpolation data BP1P, BP1M, BP2P, BP2M
	DATA_2BP Data2Bp[2] = {{0x0000, 0x2BFF, 0xFFD4, 0x0000},
	{0xF6FE, 0x0000, 0x000F, 0x3FC0}};
	
	Nmc_WriteReg5(No, IcNo, 0x04);
	// Axis assignment for interpolation. Main axis: X, Second axis: Y
	Ret = Nmc_2BPExec(No, IcNo, Data2Bp, 2, 0x04);
	// Execute 2-axis BP interpolation. The number of data is 2, X, Y axes.
	if(Ret == BP_END) AfxMessageBox ("Successful completion");
	// Return value is correct.
	
	[VB.NET] Dim Data2Bp(1) As DATA_2BP ' 2-axis BP interpolation data
	' 2-axis BP interpolation data setting
	Data2Bp(0).Bp1p = &H0: Data2Bp(0).Bp1m = &H2BFF
	Data2Bp(0).Bp2p = &HFFD4: Data2Bp(0).Bp2m = &H0
	Data2Bp(1).Bp1p = &HF6FE: Data2Bp(1).Bp1m = &H0
	Data2Bp(1).Bp2p = &HF: Data2Bp(1).Bp2m = &H3FC0
	Call Nmc_WriteReg5(No, IcNo, &H4)
	' Axis assignment for interpolation. Main axis: X, Second axis: Y
	Ret = Nmc_2BPExec(No, IcNo, Data2Bp(0), 2, &H4, False)
	' Execute 2-axis BP interpolation. The number of data is 2, X, Y axes.
	If Ret = BP_END Then ' Return value is correct.
	Call MsgBox("Successful completion")
	End If
	
	[C#]
	DATA_2BP [] Data2Bp = new DATA_2BP[1]; // 2-axis BP interpolation data
	// Set interpolation data
	Data2Bp[0].Bp1p = 0xFF30; // 1111 1111 0011 0000 BP1 + direction10 pulse
	Data2Bp[0].Bp1m = 0; // 0000 0000 0000 0000 BP1 - direction0 pulse
	Data2Bp[0].Bp2p = 0; // 0000 0000 0000 0000 BP2 + direction0 pulse
	Data2Bp[0].Bp2m = 0x84FF; // 1000 0100 1111 1111 BP2 - direction10 pulse
	
	IpAxis = IP_AXIS.IP_X | IP_AXIS.IP_Y << 2; // Axis for interpolation
	
	MC8000P.Nmc_WriteReg5(No, IcNo, IpAxis); // Axis assignment for interpolation.
	MC8000P.Nmc_StartSpd(No, IcNo, AXIS.ALL, 1); // Set initial speed
	MC8000P.Nmc_Speed(gBoardNo, IcNo, AXIS.ALL, 1); // Set drive speed
	// Execute 2-axis BP interpolation.
	Ret = MC8000P.Nmc_2BPExec(No, IcNo, Data2Bp, DataCnt, Axis, ContinueFlg);
	
	if(Ret == Nmc_Status.BP_END) // Return value is correct.

Function Name	Function and Content

Nmc_3BPExec	Execute 3-axis bit pattern interpolation using the specified interpolation data. *MCX314As only
	This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread.
	
	VC DWORD Nmc_3BPExec(int No, int IcNo, DATA_3BP* pData3Bp, int DataCnt, int IpAxis, BOOL ContinueFlg = FALSE);
	
	VB.NET Function Nmc_3BPExec(ByVal No As Integer, ByVal IcNo As Integer, ByRef pData3Bp As DATA_3BP, ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal ContinueFlg As Integer) As Integer
	
	C# Nmc_Status MC8000P.Nmc_3BPExec(int No, int IcNo, DATA_3BP[] pData3Bp, int DataCnt, int IpAxis, bool ContinueFlg);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	pData3Bp Pointer to an array of DATA_3BP structures (user-defined type). Set the 3-axis BP interpolation data to DATA_3BP. See 4.1.3 (3) for DATA_3BP.
	DataCnt The number of 3-axis BP interpolation data. Specify the number of the DATA_3BP structure (user-defined type) array.
	IpAxis Axis to execute interpolation. Specify the same value as the setting value of WR5 D0~D5 (Axis assignment). See 4.1.3 (4).
	ContinueFlg Set the flag to continue when BP interpolation stopped during driving (because the driving speed is too fast to stack the next data).
	[VC] TRUE: Continue, FALSE: Not continue Can be omitted. Default is FALSE.
	[VB.NET] True: Continue, False: Not continue
	[C#] true: Continue, false: Not continue
	
	Return Value
	If the function succeeds, the return value is BP_END.
	If the function fails, the return value is the following Error code.
	For C#, See 4.1.3 (1).
	
	■ Normal end
	BP_END BP interpolation has been successfully completed.
	
	■ Error code
	BP_CNT_ERR The number of the specified data is out of range.
	BP_ALREADY_EXEC BP interpolation or continuous interpolation is already running.
	BP_PARAM_ERR The parameter is incorrect.
	BP_NOT_OPEN_ERR The specified board is not opened.
	BP_OTHER_ERR Other errors.
	BP_STOP BP interpolation stopped during driving (too fast to stack next data).
	BP_USER_STOP The user aborted BP interpolation.
	BP_DRIVE_ERR Error occurred in the board during BP interpolation. (When the error status was set to RR0.)
	
	Example
	[VC] // 3-axis BP interpolation data BP1P, BP1M, BP2P, BP2M, BP3P, BP3M
	DATA_3BP Data3Bp[2] = {{0xFF30, 0, 0, 0x84FF, 0, 0xAC35},
	{0xAC35, 0, 0xC000, 0x36E7, 0xC000, 0x3F3F}};
	Nmc_WriteReg5(No, IcNo, 0x24);
	// Axis assignment for interpolation. Main axis: X, Second axis: Y, Third axis: Z
	Ret = Nmc_3BPExec(No, IcNo, Data3Bp, 2, 0x24);
	// Execute 3-axis BP interpolation. The number of data is 2, X, Y, Z axes.
	if(Ret == BP_END) AfxMessageBox ("Successful completion");
	// Return value is correct.
	
	[VB.NET] Dim Data3Bp(1) As DATA_3BP ' 3-axis BP interpolation data
	' 3-axis BP interpolation data setting
	Data3Bp(0).Bp1p = &HFF30: Data3Bp(0).Bp1m = &H0
	Data3Bp(0).Bp2p = &H0: Data3Bp(0).Bp2m = &H84FF
	Data3Bp(0).Bp3p = &H0: Data3Bp(0).Bp3m = &HAC35
	Data3Bp(1).Bp1p = &HAC35: Data3Bp(1).Bp1m = &H0
	Data3Bp(1).Bp2p = &HC000: Data3Bp(1).Bp2m = &H36E7
	Data3Bp(1).Bp3p = &HC000: Data3Bp(1).Bp3m = &H3F3F
	Call Nmc_WriteReg5(No, IcNo, &H24)
	' Axis assignment for interpolation. Main axis: X, Second axis: Y, Third axis: Z
	Ret = Nmc_3BPExec(No, IcNo, Data3Bp(0), 2, &H24, False)

	' Execute 3-axis BP interpolation. The number of data is 2, X, Y, Z axes.
	If Ret = BP_END Then ' Return value is correct.
	Call MsgBox("Successful completion")
	End If
	
	[C#]
	DATA_3BP [] Data3Bp = new DATA_3BP[1]; // 3-axis BP interpolation data
	// Set interpolation data
	Data3Bp[0].Bp1p = 0xFF30; // 1111 1111 0011 0000 BP1 + direction 10 pulse
	Data3Bp[0].Bp1m = 0; // 0000 0000 0000 0000 BP1 - direction 0 pulse
	Data3Bp[0].Bp2p = 0; // 0000 0000 0000 0000 BP2 + direction 0 pulse
	Data3Bp[0].Bp2m = 0x84FF; // 1000 0100 1111 1111 BP2- direction 10 pulse
	Data3Bp[0].Bp3p = 0; // 0000 0000 0000 0000 BP3 + direction 0 pulse
	Data3Bp[0].Bp3m = 0xAC35; // 1010 1100 0011 0101 BP3 - direction 8 pulse
	
	IpAxis = IP_AXIS.IP_X | IP_AXIS.IP_Y << 2 | IP_AXIS.IP_Z << 4; // Axis for interpolation
	
	MC8000P.Nmc_WriteReg5(No, IcNo, IpAxis); // Axis assignment for interpolation.
	MC8000P.Nmc_StartSpd(No, IcNo, AXIS.ALL, 1); // Set initial speed
	MC8000P.Nmc_Speed(No, IcNo, AXIS.ALL, 1); // Set drive speed
	// Execute 3-axis BP interpolation.
	Ret = MC8000P.Nmc_3BPExec(No, IcNo, Data3Bp, DataCnt, IpAxis, ContinueFlg);
	if(Ret == Nmc_Status.BP_END) // Return value is correct.

Nmc_2BPExec_BG	Execute 2-axis bit pattern interpolation in the background using the specified interpolation data. This function returns control right after the interpolation process started and executes the interpolation in the background. WM_BP_END message is sent to the specified window at the end of the interpolation and finishing status is passed. *MCX314As only **VC** DWORD Nmc_2BPExec_BG(HWND User_hWnd, int No, int IcNo, DATA_2BP* pData2Bp, int DataCnt, int IpAxis, BOOL ContinueFlg = FALSE); **VB.NET** Function Nmc_2BPExec_BG(ByVal User_hWnd As Integer, ByVal No As Integer, ByVal IcNo As Integer, ByRef pData2Bp As DATA_2BP, ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal ContinueFlg As Integer) As Intege **C#** Nmc_Status MC8000P.Nmc_2BPExec_BG(System.IntPtr User_hWnd, int No, int IcNo, DATA_2BP[] pData2Bp, int DataCnt, int IpAxis, bool ContinueFlg); **Input Parameter** User_hWnd Window handle of the user application No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. pData2Bp Pointer to an array of DATA_2BP structures (user-defined type). Set the 2-axis BP interpolation data to DATA_2BP. See 4.1.3 (3) for DATA_2BP. DataCnt The number of 2-axis BP interpolation data. Specify the number of the DATA_2BP structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of WR5 D0~D5 (Axis assignment). See 4.1.3 (4). ContinueFlg Set the flag to continue when BP interpolation stopped during driving (because the driving speed is too fast to stack the next data). [VC] TRUE: Continue, FALSE: Not continue Can be omitted. Default is FALSE. [VB.NET] True: Continue, False: Not continue [C#] true: Continue, false: Not continue **Return Value** If the interpolation process has been successfully started in the background, the return value is BP_START. If an error occurred before starting the interpolation process, the return value is the following Error code (errors before starting the interpolation). For C#, See 4.1.3 (1). ■ Normal start BP_START BP interpolation has been successfully started in the background. ■ Error code (errors before starting the interpolation)

	BP_CNT_ERR The number of the specified data is out of range.
	BP_ALREADY_EXEC BP interpolation or continuous interpolation is already running.
	BP_THREAD_ERR Thread cannot be started.
	BP_MALLOC_ERR Memory cannot be allocated.
	BP_PARAM_ERR The parameter is incorrect.
	BP_NOT_OPEN_ERR The specified board is not opened.
	BP_OTHER_ERR Other errors.
	
	After the interpolation process has been successfully started in the background, WM_BP_END message is sent to the specified window at the end of the interpolation. The board number is passed to the first argument received in WM_BP_END message received function and finishing status is passed to the second argument.
	If the interpolation has been successfully completed, the finishing status is BP_END.
	If an error occurred during the interpolation process, the following Error code (errors after starting the interpolation) returns.
	
	■ Normal end
	BP_END BP interpolation has been successfully completed.
	
	■ Error code (errors after starting the interpolation)
	BP_STOP BP interpolation stopped during driving (too fast to stack next data).
	BP_USER_STOP The user aborted BP interpolation.
	BP_DRIVE_ERR Error occurred in the board during BP interpolation.
	(When the error status was set to RR0.)
	
	Example
	[VC]
	{
	// 2-axis BP interpolation data BP1P, BP1M, BP2P, BP2M
	DATA_2BP Data2Bp[2] = {{0x0000, 0x2BFF, 0xFFD4, 0x0000},
	{0xF6FE, 0x0000, 0x000F, 0x3FC0}};
	
	Nmc_WriteReg5(No, IcNo, 0x04);
	// Axis assignment for interpolation. Main axis: X, Second axis: Y
	Ret = Nmc_2BPExec_BG(hWnd, No, IcNo, Data2Bp, 2, 0x04);
	// Execute 2-axis BP interpolation. The number of data is 2, X, Y axes.
	if(Ret == BP_START) AfxMessageBox ("Interpolation has started");
	// Return value is correct. (Interpolation has started)
	}
	BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)
	// WM_BP_END message received function setting
	ON_MESSAGE(WM_BP_END, OnMsg_BP)
	END_MESSAGE_MAP()
	
	// WM_BP_END message received function
	afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_BP(WPARAM BoardNo, LPARAM Status)
	{
	if(Status == BP_END) AfxMessageBox("Interpolation has been successfully completed");
	// Return value is correct. (Interpolation completed.)
	
	return 0;
	}
	
	[VB.NET]
	Dim Data2Bp(1) As DATA_2BP ' 2-axis BP interpolation data
	
	' 2-axis BP interpolation data setting
	Data2Bp(0).Bp1p = &H0: Data2Bp(0).Bp1m = &H2BFF
	Data2Bp(0).Bp2p = &HFFD4: Data2Bp(0).Bp2m = &H0
	Data2Bp(1).Bp1p = &HF6FE: Data2Bp(1).Bp1m = &H0
	Data2Bp(1).Bp2p = &HF: Data2Bp(1).Bp2m = &H3FC0
	
	Call Nmc_WriteReg5(No, IcNo, &H4)
	' Axis assignment for interpolation. Main axis: X, Second axis: Y.
	Ret = Nmc_2BPExec_BG(hWnd,No,IcNo, Data2Bp(0), 2, &H4, False)
	' Execute 2-axis BP interpolation. The number of data is 2, X, Y axes.
	If Ret = BP_START Then
	' Return value is correct. (Interpolation has started)

```
            Call MsgBox("Interpolation has started")
        End If
    End Sub

    ' WM_BP_END message received function
    Protected Overrides Sub WndProc(ByRef m As Message)

        If m.Msg = WM_BP_END Then    ' BP interpolation finishing message
            If lParam = BP_END Then    ' Return value is correct. (Interpolation has finished)
                Call MsgBox("Interpolation has been successfully completed")
            End If
        End If
        MyBase.WndProc(m)
    End Sub
```

```
[C#]
    DATA_2BP [] Data2Bp = new DATA_2BP[1];                  // 2-axis BP interpolation data
    // Set interpolation data
    Data2Bp[0].Bp1p = 0xFF30;       // 1111 1111 0011 0000   BP1 + direction     10 pulse
    Data2Bp[0].Bp1m = 0;            // 0000 0000 0000 0000   BP1 - direction      0 pulse
    Data2Bp[0].Bp2p = 0;            // 0000 0000 0000 0000   BP2 + direction      0 pulse
    Data2Bp[0].Bp2m = 0x84FF;       // 1000 0100 1111 1111   BP2 - direction     10 pulse

    IpAxis = IP_AXIS.IP_X | IP_AXIS.IP_Y << 2;      // Axis for interpolation

    MC8000P.Nmc_WriteReg5(No, IcNo, IpAxis);              // Axis assignment for interpolation.
    MC8000P.Nmc_StartSpd(No, IcNo, AXIS.ALL, 1);          // Set initial speed
    MC8000P.Nmc_Speed(gBoardNo, IcNo, AXIS.ALL, 1);   // Set drive speed
     // Execute 2-axis BP interpolation in the background.
    Ret = MC8000P.Nmc_2BPExec_BG((System.IntPtr)parent.Handle, No, IcNo, Data2Bp,
                                  DataCnt, IpAxis, ContinueFlg);
    if(Ret == Nmc_Status.BP_START)      // Interpolation has been successfully started.

    // WM_BP_END message received function
    protected override void WndProc(ref Message m)
    {
       // Call original WndProc (call a constructor explicitly)
       base.WndProc ( ref m );
       if ( m.Msg == (int)MSG_ID.WM_BP_END )
       {                                // BP interpolation has been successfully completed.
            if((uint)m.LParam == Nmc_Status.BP_END)

       }
    }
```

Function Name	Function and Content
Nmc_3BPExec_BG	Execute 3-axis bit pattern interpolation in the background using the specified interpolation data. This function returns control right after the interpolation process started and executes the interpolation in the background. WM_BP_END message is sent to the specified window at the end of the interpolation and finishing status is passed. *MCX314As only **VC** DWORD Nmc_3BPExec_BG(HWND User_hWnd, int No, int IcNo, DATA_3BP* pData3Bp, int DataCnt, int IpAxis, BOOL ContinueFlg = FALSE); **VB.NET** Function Nmc_3BPExec_BG(ByVal User_hWnd As Integer, ByVal No As Integer, ByVal IcNo As Integer, ByRef pData3Bp As DATA_3BP, ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal ContinueFlg As Integer) As Intege **C#** Nmc_Status MC8000P.Nmc_3BPExec_BG(System.IntPtr User_hWnd, int No, int IcNo, DATA_3BP[] pData3Bp, int DataCnt, int IpAxis, bool ContinueFlg); **Input Parameter** User_hWnd Window handle of the user application No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. pData3Bp Pointer to an array of DATA_3BP structures (user-defined type in VB). Set the 3-axis BP interpolation data to DATA_3BP. See 4.1.3 (3) for DATA_3BP. DataCnt The number of 3-axis BP interpolation data. Specify the number of the DATA_3BP structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of WR5 D0~D5 (Axis assignment). See 4.1.3 (4). ContinueFlg Set the flag to continue when BP interpolation stopped during driving (because the driving speed is too fast to stack the next data). [VC] TRUE: Continue, FALSE: Not continue Can be omitted. Default is FALSE. [VB.NET] True: Continue, False: Not continue [C#] true: Continue, false: Not continue **Return Value** If the interpolation process has been successfully started in the background, the return value is BP_START. If an error occurred before starting the interpolation process, the return value is the following Error code (errors before starting the interpolation). For C#, See 4.1.3 (1). ■ Normal start BP_START BP interpolation has been successfully started in the background. ■ Error code (errors before starting the interpolation) BP_CNT_ERR The number of the specified data is out of range. BP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. BP_THREAD_ERR Thread cannot be started. BP_MALLOC_ERR Memory cannot be allocated. BP_PARAM_ERR The parameter is incorrect. BP_NOT_OPEN_ERR The specified board is not opened. BP_OTHER_ERR Other errors. After the interpolation process has been successfully started in the background, WM_BP_END message is sent to the specified window at the end of the interpolation. The board number is passed to the first argument received in WM_BP_END message received function and finishing status is passed to the second argument. If the interpolation has been successfully completed, the finishing status is BP_END. If an error occurred during the interpolation process, the following Error code (errors after starting the interpolation) returns. ■ Normal end BP_END BP interpolation has been successfully completed. ■ Error code (errors after starting the interpolation) BP_STOP BP interpolation stopped during driving (too fast to stack next data). BP_USER_STOP The user aborted BP interpolation.

BP_DRIVE_ERR Error occurred in the board during BP interpolation.
(When the error status was set to RR0.)

Example

[VC]

```
{
    // 3-axis BP interpolation data     BP1P,     BP1M,   BP2P,     BP2M,     BP3P,     BP3M
    DATA_3BP Data3Bp[2] = {{0xFF30,     0,              0, 0x84FF,          0, 0xAC35},
                           {0xAC35,     0,     0xC000, 0x36E7, 0xC000, 0x3F3F}};
    Nmc_WriteReg5(No, IcNo, 0x24);
        // Axis assignment for interpolation. Main axis: X, Second axis: Y. Third axis: Z
    Ret = Nmc_3BPExec_BG(hWnd, No, IcNo, Data3Bp, 2, 0x24);
        // Execute 3-axis BP interpolation. The number of data is 2, X, Y, Z axes.
    if(Ret == BP_START)     AfxMessageBox("Interpolation has started");
        // Return value is correct. (Interpolation has started)
}
BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)
        // WM_BP_END message received function setting
    ON_MESSAGE( WM_BP_END, OnMsg_BP )
END_MESSAGE_MAP()

// WM_BP_END message received function
afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_BP(WPARAM BoardNo, LPARAM Status)
{
    if(Status == BP_END)     AfxMessageBox("Interpolation has been successfully completed ");
        // Return value is correct. (Interpolation has finished)
     return 0;
}
```

[VB.NET]

```
    Dim Data3Bp(1) As DATA_3BP          ' 3-axis BP interpolation data

    ' 3-axis BP interpolation data setting
     Data3Bp(0).Bp1p = &HFF30: Data3Bp(0).Bp1m = &H0
     Data3Bp(0).Bp2p = &H0:      Data3Bp(0).Bp2m = &H84FF
     Data3Bp(0).Bp3p = &H0:      Data3Bp(0).Bp3m = &HAC35
     Data3Bp(1).Bp1p = &HAC35: Data3Bp(1).Bp1m = &H0
     Data3Bp(1).Bp2p = &HC000: Data3Bp(1).Bp2m = &H36E7
     Data3Bp(1).Bp3p = &HC000: Data3Bp(1).Bp3m = &H3F3F

    Call Nmc_WriteReg5(No, IcNo, &H24)
        ' Axis assignment for interpolation. Main axis: X, Second axis: Y, Third axis: Z
    Ret = Nmc_3BPExec_BG(hWnd,No,IcNo,Data3Bp(0),2,&H24,False)
        ' Execute 3-axis BP interpolation. The number of data is 2, X, Y, Z axes.
    If Ret = BP_START Then
        ' Return value is correct. (Interpolation has started)
       Call MsgBox("Interpolation has started")
    End If
End Sub

' WM_BP_END message received function
Protected Overrides Sub WndProc(ByRef m As Message)

     If m.Msg = WM_BP_END Then    ' BP interpolation finishing message
         If lParam = BP_END Then    ' Return value is correct. (Interpolation has finished)
             Call MsgBox("Interpolation has been successfully completed ")
        End If
    End If
    MyBase.WndProc(m)
End Sub
```

[C#]

```
DATA_3BP [] Data3Bp = new DATA_3BP[1];                  // 3-axis BP interpolation data
// Set interpolation data
Data3Bp[0].Bp1p = 0xFF30;          // 1111 1111 0011 0000   BP1 + direction     10 pulse
Data3Bp[0].Bp1m = 0;               // 0000 0000 0000 0000   BP1 - direction     0 pulse
Data3Bp[0].Bp2p = 0;               // 0000 0000 0000 0000   BP2 + direction     0 pulse
Data3Bp[0].Bp2m = 0x84FF;          // 1000 0100 1111 1111   BP2 - direction     10 pulse
Data3Bp[0].Bp3p = 0;               // 0000 0000 0000 0000   BP3 + direction     0 pulse
```

```
Data3Bp[0].Bp3m = 0xAC35;          // 1010 1100 0011 0101   BP3 - direction      8 pulse

IpAxis = IP_AXIS.IP_X | IP_AXIS.IP_Y << 2 | IP_AXIS.IP_Z << 4;      // Axis for interpolation

MC8000P.Nmc_WriteReg5(No, IcNo, IpAxis);              // Axis assignment for interpolation.
MC8000P.Nmc_StartSpd(No, IcNo, AXIS.ALL, 1);          // Set initial speed
MC8000P.Nmc_Speed(gBoardNo, IcNo, AXIS.ALL, 1);       // Set drive speed
// Execute 3-axis BP interpolation in the background.
Ret = MC8000P.Nmc_3BPExec_BG((System.IntPtr)parent.Handle, No, IcNo, Data3Bp,
                                        DataCnt, IpAxis, ContinueFlg);
if(Ret == Nmc_Status.BP_START)     // Interpolation has been successfully started.

// WM_BP_END message received function
protected override void WndProc(ref Message m)
{
   // Call original WndProc (call a constructor explicitly)
   base.WndProc ( ref m );
   if ( m.Msg == (int)MSG_ID.WM_BP_END )
   {                                    // BP interpolation has been successfully completed.
         if((uint)m.LParam == Nmc_Status.BP_END)
   }
}
```

Nmc_2CIPExec

Execute 2-axis continuous interpolation using the specified interpolation data. *MCX314As only
This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread.

VC DWORD Nmc_2CIPExec(int No, int IcNo, DATA_2CIP* pData2Cip, int DataCnt, int IpAxis, BOOL SpdChgFlg = FALSE, BOOL ContinueFlg = FALSE);

VB.NET Function Nmc_2CIPExec(ByVal No As Integer, ByVal IcNo As Integer, ByRef pData2Cip As DATA_2CIP, ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal SpdChgFlg As Integer,ByVal ContinueFlg As Integer) As Integer

C# Nmc_Status MC8000P. Nmc_2CIPExec(int No, int IcNo, DATA_2CIP[] pData2Cip, int DataCnt, int IpAxis, bool SpdChgFlg, bool ContinueFlg);

Input Parameter

No	Board number (setting value of rotary switch (0~15) on the board)
IcNo	IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
pData2Cip	Pointer to an array of DATA_2CIP structures (user-defined type in VB). Set the 2-axis continuous interpolation data to DATA_2CIP. See 4.1.3 (3) for DATA_2CIP.
DataCnt	The number of 2-axis continuous interpolation data. Specify the number of the DATA_2CIP structure (user-defined type) array.
IpAxis	Axis to execute interpolation. Specify the same value as the setting value of WR5 D0~D5 (Axis assignment). See 4.1.3 (4).
SpdChgFlg	Set the flag to change the speed during the interpolation process. If you change the speed, See 4.1.3 (5). [VC] TRUE: Change, FALSE: Not change Can be omitted. Default is FALSE. [VB.NET] True: Change, False: Not change [C#] true: Change, false: Not change **When selecting Change:** Refers to the setting value of DATA_2CIP Speed. Set 1～8000 to Speed・・・Changes to the setting speed. Set 0 to Speed・・・・・・ Not change the speed. **When selecting Not change:** Not refers to the setting value of DATA_2CIP Speed.
ContinueFlg	Set the flag to continue when continuous interpolation stopped during driving (because the driving speed is too fast to set the next data). [VC] TRUE: Continue, FALSE: Not continue Can be omitted. Default is FALSE. [VB.NET] True: Continue, False: Not continue [C#] true: Continue, false: Not continue

Return Value

If the function succeeds, the return value is CIP_END.

If the function fails, the return value is the following Error code.
For C#, See 4.1.3 (1).

■ Normal end

CIP_END	Continuous interpolation has been successfully completed.

■ Error code

CIP_CNT_ERR	The number of the specified data is out of range.
CIP_ALREADY_EXEC	BP interpolation or continuous interpolation is already running.
CIP_CMD_ERR	The wrong command was specified.
CIP_PARAM_ERR	The parameter is incorrect.
CIP_NOT_OPEN_ERR	The specified board is not opened.
CIP_OTHER_ERR	Other errors.
CIP_STOP	Continuous interpolation stopped during driving (too fast to set next data).
CIP_USER_STOP	The user aborted continuous interpolation.
CIP_DRIVE_ERR	Error occurred in the board during continuous interpolation. (When the error status was set to RR0.)

NOTE
Before using this function, set 8000 to initial speed. (Don't change initial speed during executing this function.) For the details, please see chapter 4.1.4.

Example

[VC]
```
// 2-axis continuous interpolation data   Command, Speed, Finishing point 1,Finishing point 2,Center point 1,Center point 2
DATA_2CIP Data2Cip[2]={{CMD_IP_2ST, 0,   4500,      0,      0,      0},
                                    // 2-axis linear interpolation
                      {CMD_IP_CCW, 0,   1500,   1500,      0,   1500}};
                                    // CCW circular interpolation

Nmc_WriteReg5(No, IcNo, 0x04);
      // Axis assignment for interpolation. Main axis: X, Second axis: Y.
Ret = Nmc_2CIPExec(No, IcNo, Data2Cip, 2, 0x04);
      // Execute 2-axis continuous interpolation. The number of data is 2, X, Y axes.
if(Ret == CIP_END)    AfxMessageBox ("Successful completion");
      // Return value is correct.
```

[VB.NET]
```
Dim Data2Cip(1) As DATA_2CIP              ' 2-axis continuous interpolation data

' 2-axis continuous interpolation data setting
Data2Cip(0).Command = CMD_IP_2ST       ' 2-axis linear interpolation
Data2Cip(0).EndP1 = 4500
Data2Cip(0).EndP2 = 0

Data2Cip(1).Command = CMD_IP_CCW       ' CCW circular interpolation
Data2Cip(1).EndP1 = 1500
Data2Cip(1).EndP2 = 1500
Data2Cip(1).Center1 = 0
Data2Cip(1).Center2 = 1500

Call Nmc_WriteReg5(No, IcNo, &H4)
      ' Axis assignment for interpolation. Main axis: X, Second axis: Y.
R Ret = Nmc_2CIPExec(No, IcNo, Data2Cip(0), 2, &H4, False, False)
      ' 2-axis continuous interpolation. The number of data is 2, X, Y axes.
If Ret = CIP_END Then              ' Return value is correct.
   Call MsgBox("Successful completion")
End If
```

[C#]
```
DATA_2CIP [] Data2Cip = new DATA_2CIP[1];      // 2-axis continuous interpolation data
```

	// Set interpolation data
	Data2Cip[0].Command = (ushort)CMD.CMD_IP_CW; // CW circular interpolation
	Data2Cip[0].EndP1 = 2000;
	Data2Cip[0].EndP2 = 2000;
	Data2Cip[0].Center1 = 2000;
	Data2Cip[0].Center2 = 0;
	Data2Cip[0].Speed = 200; // Change speed (200)
	
	IpAxis = IP_AXIS.IP_X | IP_AXIS.IP_Y << 2; // Axis for interpolation
	
	MC8000P.Nmc_WriteReg5(No, IpAxis); // Axis assignment for interpolation.
	
	// Execute 2-axis continuous interpolation.
	// This function will not return control unless the interpolation process ends.
	Ret = MC8000P.Nmc_2CIPExec(No, int IcNo, Data2Cip, DataCnt, IpAxis, SpdChgFlg, ContinueFlg);
	if(Ret == Nmc_Status.CIP_END)
	// Continuous interpolation has been successfully completed.

Nmc_3CIPExec	Execute 3-axis continuous interpolation using the specified interpolation data. *MCX314As only This function returns control after the interpolation process has finished. That is the control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread. **VC** DWORD Nmc_3CIPExec(int No, int IcNo, DATA_3CIP* pData3Cip, int DataCnt, int IpAxis, BOOL SpdChgFlg = FALSE, BOOL ContinueFlg = FALSE); **VB.NET** Function Nmc_3CIPExec(ByVal No As Integer, ByVal IcNo As Integer, ByRef pData3Cip As DATA_3CIP, ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal SpdChgFlg As Integer,ByVal ContinueFlg As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_3CIPExec(int No, IcNo, DATA_3CIP[] pData3Cip, int DataCnt, int IpAxis, bool SpdChgFlg, bool ContinueFlg); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. pData3Cip Pointer to an array of DATA_3CIP structures (user-defined type in VB). Set the 3-axis continuous interpolation data to DATA_3CIP. See 4.1.3 (3) for DATA_3CIP. DataCnt The number of 3-axis continuous interpolation data. Specify the number of the DATA_3CIP structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of WR5 D0~D5 (Axis assignment). See 4.1.3 (4). SpdChgFlg Set the flag to change the speed during the interpolation process. If you change the speed, See 4.1.3 (5). [VC] TRUE: Change, FALSE: Not change Can be omitted. Default is FALSE. [VB] True: Change, False: Not change [C#] true: Change, false: Not change **When selecting Change:** Refers to the setting value of DATA_3CIP Speed. Set 1～8000 to Speed・・・Changes to the setting speed. Set 0 to Speed・・・・・・ Not change the speed. **When selecting Not change:** Not refers to the setting value of DATA_3CIP Speed. ContinueFlg Set the flag to continue when continuous interpolation stopped during driving (because the driving speed is too fast to set the next data). [VC] TRUE: Continue, FALSE: Not continue Can be omitted. Default is FALSE. [VB] True: Continue, False: Not continue [C#] true: Continue, false: Not continue **Return Value** If the function succeeds, the return value is CIP_END. If the function fails, the return value is the following Error code. For C#, See 4.1.3 (1).

■ Normal end

CIP_END	Continuous interpolation has been successfully completed.

■ Error code

CIP_CNT_ERR	The number of the specified data is out of range.
CIP_ALREADY_EXEC	BP interpolation or continuous interpolation is already running.
CIP_CMD_ERR	The wrong command was specified.
CIP_PARAM_ERR	The parameter is incorrect.
CIP_NOT_OPEN_ERR	The specified board is not opened.
CIP_OTHER_ERR	Other errors.
CIP_STOP	Continuous interpolation stopped during driving (too fast to set next data).
CIP_USER_STOP	The user aborted continuous interpolation.
CIP_DRIVE_ERR	Error occurred in the board during continuous interpolation. (When the error status was set to RR0.)

NOTE

Before using this function, set 8000 to initial speed.(Don't change initial speed during executing this function.) For the details, please see chapter 4.1.4.

Example

```
[VC]   DATA_3CIP Data3Cip[2];              //   3-axis continuous interpolation data

       // 3-axis continuous interpolation data setting
       Data3Cip[0].EndP1 = 1000;
       Data3Cip[0].EndP2 = 2000;
       Data3Cip[0].EndP3 = 3000;
       Data3Cip[0].Speed = 0;

       Data3Cip[1].EndP1 = 2000;
       Data3Cip[1].EndP2 = -1000;
       Data3Cip[1].EndP3 = 3000;
       Data3Cip[1].Speed = 0;

       Nmc_WriteReg5(No, IcNo, 0x24);
            // Axis assignment for interpolation. Main axis: X, Second axis: Y, Third axis: Z.
       Ret = Nmc_3CIPExec(No, IcNo, Data3Cip, 2, 0x24);
            // Execute 3-axis continuous interpolation. The number of data is 2, X, Y, Z axes.
       if(Ret == CIP_END)     AfxMessageBox ("Successful completion");
            // Return value is correct.

[VB.NET]   Dim Data3Cip(1) As DATA_3CIP      ' 3-axis continuous interpolation data

       ' 3-axis continuous interpolation data setting
       Data3Cip(0).EndP1 = 1000
       Data3Cip(0).EndP2 = 2000
       Data3Cip(0).EndP3 = 3000
       Data3Cip(0).Speed = 0

       Data3Cip(1).EndP1 = 2000
       Data3Cip(1).EndP2 = -1000
       Data3Cip(1).EndP3 = 3000
       Data3Cip(1).Speed = 0

       Call Nmc_WriteReg5(No, IcNo, &H24)
            ' Axis assignment for interpolation. Main axis: X, Second axis: Y, Third axis: Z.
       Ret = Nmc_3CIPExec(No,IcNo, Data3Cip(0), 2, &H24, False,False)
            ' 3-axis continuous interpolation. The number of data is 2, X, Y, Z axes.
       If Ret = CIP_END Then
            ' Return value is correct.
```

```
            Call MsgBox("Successful completion")
        End If

[C#]
    DATA_3CIP [] Data3Cip = new DATA_3CIP[1];       // 3-axis continuous interpolation data
    // Set interpolation data
    Data3Cip[0].EndP1 = 1000;
    Data3Cip[0].EndP2 = 2000;
    Data3Cip[0].EndP3 = 3000;
    Data3Cip[0].Speed = 0;

    IpAxis = IP_AXIS.IP_X | IP_AXIS.IP_Y << 2 | IP_AXIS.IP_Z << 4;    // Axis for interpolation


    MC8000P.Nmc_WriteReg5(gBoardNo, IcNo, IpAxis);    // Axis assignment for interpolation.
    MC8000P.Nmc_StartSpd(gBoardNo, IcNo, AXIS.ALL, 8000);   // Set initial speed
    MC8000P.Nmc_Speed(gBoardNo, IcNo, AXIS.ALL, 100);        // Set drive speed

    // Execute 3-axis continuous interpolation.
    // This function will not return control unless the interpolation process ends.
    Ret = MC8000P.Nmc_3CIPExec(No, IcNo, Data3Cip, DataCnt, IpAxis, SpdChgFlg,
                                        ContinueFlg);
    if(Ret == Nmc_Status.CIP_END)
                                // Continuous interpolation has been successfully completed.
```

Function Name	Function and Content
Nmc_2CIPExec_BG	Execute 2-axis continuous interpolation in the background using the specified interpolation data. This function returns control right after the interpolation process started and executes the interpolation in the background. WM_CIP_END message is sent to the specified window at the end of the interpolation and finishing status is passed. *MCX314As only **VC** DWORD Nmc_2CIPExec_BG(HWND User_hWnd, int No, int IcNo, DATA_2CIP* pData2Cip, int DataCnt,int IpAxis, BOOL SpdChgFlg = FALSE, BOOL ContinueFlg = FALSE); **VB.NET** Function Nmc_2CIPExec_BG(ByVal User_hWnd As Integer, ByVal No As Integer, ByVal IcNo As Integer, ByRef pData2Cip As DATA_2CIP, ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal SpdChgFlg As Integer,ByVal ContinueFlg As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_2CIPExec_BG(System.IntPtr User_hWnd, int No, int IcNo, DATA_2CIP[] pData2Cip, int DataCnt, int IpAxis, bool SpdChgFlg, bool ContinueFlg); **Input Parameter** User_hWnd Window handle of the user application No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. pData2Cip Pointer to an array of DATA_2CIP structures (user-defined type). Set the 2-axis continuous interpolation data to DATA_2CIP. See 4.1.3 (3) for DATA_2CIP. DataCnt The number of 2-axis continuous interpolation data. Specify the number of the DATA_2CIP structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of WR5 D0~D5 (Axis assignment). See 4.1.3 (4). SpdChgFlg Set the flag to change the speed during the interpolation process. If you change the speed, See 4.1.3 (5). [VC] TRUE: Change, FALSE: Not change Can be omitted. Default is FALSE. [VB] True: Change, False: Not change [C#] true: Change, false: Not change **When selecting Change:** Refers to the setting value of DATA_2CIP Speed. Set 1～8000 to Speed・・・Changes to the setting speed. Set 0 to Speed・・・・・・ Not change the speed. **When selecting Not change:** Not refers to the setting value of DATA_2CIP Speed. ContinueFlg Set the flag to continue when continuous interpolation stopped during driving (because the driving speed is too fast to set the next data). [VC] TRUE: Continue, FALSE: Not continue Can be omitted. Default is FALSE. [VB] True: Continue, False: Not continue [C#] true: Continue, false: Not continue **Return Value** If the interpolation process has been successfully started in the background, the return value is CIP_START. If an error occurred before starting the interpolation process, the return value is the following Error code (errors before starting the interpolation). For C#, See 4.1.3 (1). ■ Normal start CIP_START Continuous interpolation has been successfully started in the background. ■ Error code (errors before starting the interpolation) CIP_CNT_ERR The number of the specified data is out of range. CIP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. CIP_THREAD_ERR Thread cannot be started. CIP_MALLOC_ERR Memory cannot be allocated. CIP_CMD_ERR The wrong command was specified. CIP_PARAM_ERR The parameter is incorrect. CIP_NOT_OPEN_ERR The specified board is not opened. CIP_OTHER_ERR Other errors. After the interpolation process has been successfully started in the background, WM_CIP_END message is sent to the specified window at the end of the interpolation. The board number is passed to the first argument received in WM_CIP_END message received function and finishing status is passed to the second argument.

If the interpolation has been successfully completed, the finishing status is CIP_END.
If an error occurred during the interpolation process, the following Error code (errors after starting the interpolation) returns.

■ Normal end

CIP_END	Continuous interpolation has been successfully completed.

■ Error code (errors after starting the interpolation)

CIP_STOP	Continuous interpolation stopped during driving (too fast to set next data).
CIP_USER_STOP	The user aborted continuous interpolation.
CIP_DRIVE_ERR	Error occurred in the board during continuous interpolation. (When the error status was set to RR0.)

NOTE
Before using this function, set 8000 to initial speed. (Don't change initial speed during executing this function.) For the details, please see chapter 4.1.4.

Example

[VC] {

```
// 2-axis continuous interpolation data Command, Speed, Finishing point 1,Finishing point
   2, Center point 1, Center point 2
DATA_2CIP Data2Cip[2]= {{CMD_IP_2ST, 0,   4500,       0,         0,         0},
                                    // 2-axis linear interpolation
                                   {CMD_IP_CCW, 0,   1500,   1500,        0,    1500}};
                                    // CCW circular interpolation
Nmc_WriteReg5(No, IcNo, 0x04);
      // Axis assignment for interpolation. Main axis: X, Second axis: Y.
Ret = Nmc_2CIPExec_BG(hWnd, No, IcNo, Data2Cip, 2, 0x04);
      // Execute 2-axis continuous interpolation. The number of data is 2, X, Y axes.
if(Ret == CIP_START)    AfxMessageBox("Interpolation has started");
      // Return value is correct. (Interpolation has started)
}
BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)
      // WM_CIP_END message received function setting
 ON_MESSAGE( WM_CIP_END, OnMsg_CIP )
END_MESSAGE_MAP()
// WM_CIP_END message received function
afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_CIP(WPARAM BoardNo, LPARAM Status)
{
 if(Status == CIP_END)   AfxMessageBox("Interpolation has been successfully completed");
       // Return value is correct. (Interpolation has finished)
 return 0;
}
```

[VB.NET]

```
Dim Data2Cip(1) As DATA_2CIP          ' 2-axis continuous interpolation data
' 2-axis continuous interpolation data setting
Data2Cip(0).Command = CMD_IP_2ST      ' 2-axis linear interpolation
Data2Cip(0).EndP1 = 4500
Data2Cip(0).EndP2 = 0

Data2Cip(1).Command = CMD_IP_CCW      ' CCW circular interpolation
Data2Cip(1).EndP1 = 1500
Data2Cip(1).EndP2 = 1500
Data2Cip(1).Center1 = 0
Data2Cip(1).Center2 = 1500

Call Nmc_WriteReg5(No, IcNo, &H4)
      ' Axis assignment for interpolation. Main axis: X, Second axis: Y.
      ' 2-axis continuous interpolation. The number of data is 2, X, Y axes.
Ret = Nmc_2CIPExec_BG(hWnd, No, IcNo, Data2Cip(0), 2, &H4, False, False)
If Ret = CIP_START Then   ' Return value is correct. (Interpolation has started)
   Call MsgBox("Interpolation has started")
End If
End Sub

' WM_CIP_END message received function
Protected Overrides Sub WndProc(ByRef m As Message)
   If m.Msg = WM_CIP_END Then   ' Continuous interpolation finishing message
      If lParam = CIP_END Then   ' Return value is correct. (Interpolation has finished)
         Call MsgBox("Interpolation has been successfully completed")
```

```
            End If
        End If
        MyBase.WndProc(m)
    End Sub


[C#]
    DATA_2CIP [] Data2Cip = new DATA_2CIP[1];   // 2-axis continuous interpolation data

    // Set interpolation data
    Data2Cip[0].Command = (ushort)CMD.CMD_IP_CW;          // CW circular interpolation
    Data2Cip[0].EndP1 = 2000;
    Data2Cip[0].EndP2 = 2000;
    Data2Cip[0].Center1 = 2000;
    Data2Cip[0].Center2 = 0;
    Data2Cip[0].Speed = 200;                                        // Change speed (200)
    IpAxis = IP_AXIS.IP_X | IP_AXIS.IP_Y << 2;      // Axis for interpolation
    MC8000P.Nmc_WriteReg5(No, int IcNo, IpAxis);    // Axis assignment for interpolation.

    // Execute 2-axis continuous interpolation.
    // Execute in the background.
    Ret = MC8000P.Nmc_2CIPExec_BG((System.IntPtr)parent.Handle, dNo, int IcNo, Data2Cip,
                                    DataCnt, IpAxis, SpdChgFlg, ContinueFlg);
    if(Ret == Nmc_Status.CIP_START)
                        // Continuous interpolation has been successfully started.

     // WM_BP_END message received function
     protected override void WndProc(ref Message m)
     {
        // Call original WndProc (call a constructor explicitly)
        base.WndProc ( ref m );
         if ( m.Msg == (int)MSG_ID.WM_CIP_END )
        {
            if((uint)m.LParam == Nmc_Status.CIP_END)
                        // Continuous interpolation has been successfully completed.
        }
     }
```

Function Name	Function and Content
Nmc_3CIPExec_BG	Execute 3-axis continuous interpolation in the background using the specified interpolation data. This function returns control right after the interpolation process started and executes the interpolation in the background. WM_CIP_END message is sent to the specified window at the end of the interpolation and finishing status is passed. *MCX314As only **VC** DWORD Nmc_3CIPExec_BG(HWND User_hWnd, int No, int IcNo, DATA_3CIP* pData3Cip,int DataCnt, int IpAxis, BOOL SpdChgFlg = FALSE, BOOL ContinueFlg = FALSE); **VB.NET** Function Nmc_3CIPExec_BG(ByVal User_hWnd As Integer, ByVal No As Integer, ByVal IcNo As Integer, ByRef pData3Cip As DATA_3CIP, ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal SpdChgFlg As Integer,ByVal ContinueFlg As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_3CIPExec_BG(System.IntPtr User_hWnd, int No, int IcNo, DATA_3CIP[] pData3Cip, int IpAxis, bool SpdChgFlg, bool ContinueFlg); **Input Parameter** User_hWnd Window handle of the user application No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details. pData3Cip Pointer to an array of DATA_3CIP structures (user-defined type in VB). Set the 3-axis continuous interpolation data to DATA_3CIP. See 4.1.3 (3) for DATA_3CIP. DataCnt The number of 3-axis continuous interpolation data. Specify the number of the DATA_3CIP structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of WR5 D0~D5 (Axis assignment). See 4.1.3 (4). SpdChgFlg Set the flag to change the speed during the interpolation process. If you change the speed, See 4.1.3 (5). [VC] TRUE: Change, FALSE: Not change Can be omitted. Default is FALSE. [VB] True: Change, False: Not change [C#] true: Change, false: Not change **When selecting Change:** Refers to the setting value of DATA_3CIP Speed. Set 1～8000 to Speed・・・Changes to the setting speed. Set 0 to Speed・・・・・・ Not change the speed. **When selecting Not change:** Not refers to the setting value of DATA_3CIP Speed. ContinueFlg Set the flag to continue when continuous interpolation stopped during driving (because the driving speed is too fast to set the next data). [VC] TRUE: Continue, FALSE: Not continue Can be omitted. Default is FALSE. [VB] True: Continue, False: Not continue [C#] true: Continue, false: Not continue **Return Value** If the interpolation process has been successfully started in the background, the return value is CIP_START. If an error occurred before starting the interpolation process, the return value is the following Error code (errors before starting the interpolation). For C#, See 4.1.3 (1). ■ Normal start CIP_START Continuous interpolation has been successfully started in the background. ■ Error code (errors before starting the interpolation) CIP_CNT_ERR The number of the specified data is out of range. CIP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. CIP_THREAD_ERR Thread cannot be started. CIP_MALLOC_ERR Memory cannot be allocated. CIP_CMD_ERR The wrong command was specified. CIP_PARAM_ERR The parameter is incorrect. CIP_NOT_OPEN_ERR The specified board is not opened. CIP_OTHER_ERR Other errors. After the interpolation process has been successfully started in the background, WM_CIP_END message is sent to the specified window at the end of the interpolation. The board number is passed to the first argument received in WM_CIP_END message received function and finishing status is passed to the second argument. If the interpolation has been successfully completed, the finishing status is CIP_END.

If an error occurred during the interpolation process, the following Error code (errors after starting the interpolation) returns.

■ Normal end

CIP_END	Continuous interpolation has been successfully completed.

■ Error code (errors after starting the interpolation)

CIP_STOP	Continuous interpolation stopped during driving (too fast to set next data).
CIP_USER_STOP	The user aborted continuous interpolation.
CIP_DRIVE_ERR	Error occurred in the board during continuous interpolation. (When the error status was set to RR0.)

NOTE

Before using this function, set 8000 to initial speed.(Don't change initial speed during executing this function.) For the details, please see chapter 4.1.4.

Example

[VC]

```
{
        DATA_3CIP Data3Cip[2];              // 3-axis continuous interpolation data
        // 3-axis continuous interpolation data setting
         Data3Cip[0].EndP1 = 1000;
         Data3Cip[0].EndP2 = 2000;
         Data3Cip[0].EndP3 = 3000;

         Data3Cip[1].EndP1 = 2000;
         Data3Cip[1].EndP2 = -1000;
         Data3Cip[1].EndP3 = 3000;

        Nmc_WriteReg5(No, IcNo, 0x24);
            // Axis assignment for interpolation. Main axis: X, Second axis: Y, Third axis: Z.
        Ret = Nmc_3CIPExec_BG(hWnd,No, IcNo, Data3Cip, 2, 0x24);
            // Execute 3-axis continuous interpolation. The number of data is 2, X, Y, Z axes.
        if(Ret == CIP_START)    AfxMessageBox ("Interpolation has started");
            // Return value is correct. (Interpolation has started)
    }
    BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)
            // WM_CIP_END message received function setting
        ON_MESSAGE( WM_CIP_END, OnMsg_CIP )
    END_MESSAGE_MAP()

    // WM_CIP_END message received function
    afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_CIP(WPARAM BoardNo, LPARAM Status)
    {
        if(Status == CIP_END)   AfxMessageBox("Interpolation has been successfully completed");
            // Return value is correct. (Interpolation has finished)
        return 0;
    }
```

[VB.NET]

```
        Dim Data3Cip(1) As DATA_3CIP      ' 3-axis continuous interpolation data
        ' 3-axis continuous interpolation data setting
         Data3Cip(0).EndP1 = 1000
         Data3Cip(0).EndP2 = 2000
         Data3Cip(0).EndP3 = 3000

         Data3Cip(1).EndP1 = 2000
         Data3Cip(1).EndP2 = -1000
         Data3Cip(1).EndP3 = 3000

        Call Nmc_WriteReg5(No, IcNo, &H24)
            ' Axis assignment for interpolation. Main axis: X, Second axis: Y, Third axis: Z
            ' 3-axis continuous interpolation. The number of data is 2, X, Y, Z axes.
        Ret = Nmc_3CIPExec_BG(hWnd, No, IcNo, Data3Cip(0), 2, &H24, False, False)
        If Ret = CIP_START Then   ' Return value is correct. (Interpolation has started)
            Call MsgBox("Interpolation has started")
        End If
    End Sub

    ' WM_CIP_END message received function
```

```
Protected Overrides Sub WndProc(ByRef m As Message)
    If m.Msg = WM_CIP_END Then    ' Continuous interpolation finishing message
        If lParam = CIP_END Then    ' Return value is correct. (Interpolation has finished)
            Call MsgBox("Interpolation has been successfully completed ")
        End If
    End If
    MyBase.WndProc(m)
End Sub

[C#]
    DATA_3CIP [] Data3Cip = new DATA_3CIP[1];   // 3-axis continuous interpolation data
    // Set interpolation data
    Data3Cip[0].EndP1 = 1000;
    Data3Cip[0].EndP2 = 2000;
    Data3Cip[0].EndP3 = 3000;
    Data3Cip[0].Speed = 0;

    IpAxis = IP_AXIS.IP_X | IP_AXIS.IP_Y << 2 | IP_AXIS.IP_Z << 4;    // Axis for interpolation

    MC8000P.Nmc_WriteReg5(gBoardNo, int IcNo, IpAxis); // Axis assignment for interpolation.
    MC8000P.Nmc_StartSpd(gBoardNo, int IcNo, AXIS.ALL, 8000);    // Set initial speed
    MC8000P.Nmc_Speed(gBoardNo, int IcNo, AXIS.ALL, 100);          // Set drive speed

    // Execute 3-axis continuous interpolation.
    // Execute in the background.
    Ret = MC8000P.Nmc_3CIPExec_BG((System.IntPtr)parent.Handle, gBoardNo, int IcNo,
                                    Data3Cip, DataCnt, IpAxis, SpdChgFlg, ContinueFlg);
    if(Ret == Nmc_Status.CIP_START)
                            // Continuous interpolation has been successfully started.

     // WM_BP_END message received function
     protected override void WndProc(ref Message m)
     {
        // Call original WndProc (call a constructor explicitly)
        base.WndProc ( ref m );
        if ( m.Msg == (int)MSG_ID.WM_CIP_END )
        {
                if((uint)m.LParam == Nmc_Status.CIP_END)
                            // Continuous interpolation has been successfully completed.
        }
     }
```

Function Name	Function and Content

Nmc_IPStop	Stop the interpolation process during driving. *MCX314As only
	The interpolation driving stops immediately and terminates the executed interpolation process in Nmc_xxx interpolation function.
	
	When stopping the interpolation process using Nmc_IPStop, the return value of each interpolation function is the following error code.
	◆ BP interpolation: BP_USER_STOP
	◆ Continuous interpolation: CIP_USER_STOP
	
	VC BOOL Nmc_IPStop(int No int IcNo,);
	VB.NET Function Nmc_IPStop(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# bool MC8000P.Nmc_IPStop(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 4.1.3 (7) for more details.
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[VB.NET] If the function succeeds, the return value is nonzero.
	If the function fails, the return value is 0.
	[C#] If the function succeeds, the return value is true.
	If the function fails, the return value is false.
	
	Example
	[VC] Nmc_IPStop(No, IcNo); // Stop the interpolation process during driving.
	[VB.NET] Call Nmc_IPStop(No, IcNo)
	[C#] MC8000P.Nmc_IPStop(No, IcNo);

Function Name	Function and Content

Nmc_IPGetMsgNo	Read the board and IC number from the parameter wParam of the following received message at the end of the Interpolation. *MCX314As only
	
	◆ BP interpolation: WM_BP_END
	◆ Continuous interpolation: WM_CIP_END
	
	VC void Nmc_IPGetMsgNo(int wParam, int* No, int* IcNo);
	VB.NET Sub Nmc_IPGetMsgNo(ByVal wParam As Integer, ByRef No As Integer, ByRef IcNo As Integer)
	C# bool MC8000P.Nmc_IPGetMsgNo(int wParam, out int No, out int IcNo)
	
	Input Parameter
	wParam The parameter wParam of received message at the end of the Interpolation.
	No [VC] Address of a variable to store the board number.
	[VB.NET] [C#] Variable to store the board number.
	IcNo [VC] Address of a variable to store the IC number.
	[VB.NET] [C#] Variable to store the IC number.
	
	Return Value
	None
	
	Example
	[VC] int BdNo; // Board number
	int IcNo; // IC number
	Nmc_IPGetMsgNo(wParam, &BdNo, &IcNo);
	
	[VB.NET] Dim BdNo As Integer ' Board number
	Dim IcNo As Integer ' IC number
	Call Nmc_IPGetMsgNo(m.WParam.ToInt32(), BdNo, IcNo)
	
	[C#] int BdNo; // Board number
	int IcNo; // IC number (0~1)
	MC8000P.Nmc_IPGetMsgNo(WParam, out BdNo, out IcNo)

4.1.3. Footnote

(1) Each definition is defined in the following files.

VC	MC8000P_DLL.H
VB.NET	MC8000P_DLL.vb
C#	Users can refer to or input each definition through the IntelliSense.

VC++, VB.NET and C# definitions are as follows:

①Register number

[VC]

```
#define MCX_WR0          0x0000    // WR0
#define MCX_WR1          0x0001    // WR1
#define MCX_WR2          0x0002    // WR2
#define MCX_WR3          0x0003    // WR3
#define MCX_WR4          0x0004    // WR4
#define MCX_WR5          0x0005    // WR5
#define MCX_WR6          0x0006    // WR6
#define MCX_WR7          0x0007    // WR7

#define MCX_RR0          0x0000    // RR0
#define MCX_RR1          0x0001    // RR1
#define MCX_RR2          0x0002    // RR2
#define MCX_RR3          0x0003    // RR3
#define MCX_RR4          0x0004    // RR4
#define MCX_RR5          0x0005    // RR5
#define MCX_RR6          0x0006    // RR6
#define MCX_RR7          0x0007    // RR7
```

[C#]

```
// ■Nmc_OutPort／Nmc_InPort
// When the board has one IC, use WR0_A ~ WR7_A／RR0_A ~ RR7_A
// When the board has 2 ICs, use WR0_A~WR7_A/RR0_A~RR7_A for IC_A, and
//                              WR0_B~WR7_B/RR0_B~RR7_B for IC_B.
//■Nmc_WriteReg／Nmc_ReadReg
// Use WR0~WR7/RR0~RR7.

  public enum REG_MCX : int
  {
        //■Nmc_OutPort
        // Write register address
        // WR0~WR7 for IC-A
        WR0_A   =0x0000,
        WR1_A   =0x0001,
        WR2_A   =0x0002,
        WR3_A   =0x0003,
        WR4_A   =0x0004,
        WR5_A   =0x0005,
        WR6_A   =0x0006,
        WR7_A   =0x0007,

        // WR0~WR7 for IC-B
        WR0_B   =0x0008,
        WR1_B   =0x0009,
        WR2_B   =0x000A,
        WR3_B   =0x000B,
        WR4_B   =0x000C,
        WR5_B   =0x000D,
        WR6_B   =0x000E,
        WR7_B   =0x000F,
```

```
        // PIX132 address
        WR14    =0x0014,
        WR15    =0x0015,
        WR16    =0x0016,

        //■Nmc_InPort
        // Read register address
        // RR0～RR7 for IC-A
        RR0_A   =0x0000,
        RR1_A   =0x0001,
        RR2_A   =0x0002,
        RR3_A   =0x0003,
        RR4_A   =0x0004,
        RR5_A   =0x0005,
        RR6_A   =0x0006,
        RR7_A   =0x0007,

        // RR0～RR7 for IC-B
        RR0_B   =0x0008,
        RR1_B   =0x0009,
        RR2_B   =0x000A,
        RR3_B   =0x000B,
        RR4_B   =0x000C,
        RR5_B   =0x000D,
        RR6_B   =0x000E,
        RR7_B   =0x000F,

        // PIX132 address
        RR14    =0x0014,
        RR15    =0x0015,
        RR16    =0x0016,

        //■Nmc_WriteReg
        // Write register address
        WR0     =0x0000,
        WR1     =0x0001,
        WR2     =0x0002,
        WR3     =0x0003,
        WR4     =0x0004,
        WR5     =0x0005,
        WR6     =0x0006,
        WR7     =0x0007,

        //■Nmc_ReadReg
        // Read register address
        RR0     =0x0000,
        RR1     =0x0001,
        RR2     =0x0002,
        RR3     =0x0003,
        RR4     =0x0004,
        RR5     =0x0005,
        RR6     =0x0006,
        RR7     =0x0007
}
```

Example) REG_MCX of WR0 REG_MCX.WR0

②Axis assignment

[VC]

```
#define AXIS_ALL        0xF     // All axes
#define AXIS_X          0x1     // X axis
#define AXIS_Y          0x2     // Y axis
#define AXIS_Z          0x4     // Z axis
#define AXIS_U          0x8     // U axis
#define AXIS_NONE       0       // No axis assignment
```

[C#]

```
  public enum AXIS : int
  {
        // Axis assignment
        ALL     =0xF,   // All axes
        X       =0x1,   // X axis
        Y       =0x2,   // Y axis
        Z       =0x4,   // Z axis
        U       =0x8,   // U axis
        NONE    =0      // No axis assignment
  }
```

Example) To assign all axes, AXIS.ALL

③Device ID

Device Id is the fixed number assigned to a board.
Number of each board is as below.

Board	Device ID
MC8043P / MC8043Pe	0xA0A2
MC8082P / MC8082Pe	0xA0D0

[Note] Device ID of MC8022P and MC8042P are the same with that of MC8082P.

[VC]

```
#define ID_MC8043P      0xA0A2  // MC8043P and MC8043Pe
#define ID_MC8082P      0xA0D0  // MC8082P, 42P, 22P and MC8082Pe
```

Example) MC8082P, 42P, 22P and MC8082Pe are ID_MC8082P, MC8043P and MC8082Pe are ID_MC8043P.

[C#]

```
  public enum Dev_ID : ushort
  {
        MC8043P  =0xA0A2, // MC8043P and MC8043Pe
        MC8082P  =0xA0D0  // MC8082P, 42P, 22P and MC8082Pe
  }
```

Example) MC8082P, 42P, 22P and MC8082Pe are Dev_ID.MC8082P, MC8043P and MC8043Pe are Dev_ID.MC8043P.

④Command definition *Refer to the User's Manual of each IC for available commands.
Commands of both (Ⅰ) and (Ⅱ) are available.

[VC]

```
// Driving commands
//( Ⅰ )
#define CMD_F_DRV_P     0x20    // + direction fixed pulse driving
#define CMD_F_DRV_M     0x21    // - direction fixed pulse driving
#define CMD_C_DRV_P     0x22    // + direction continuous pulse driving
#define CMD_C_DRV_M     0x23    // - direction continuous pulse driving
#define CMD_START_HOLD  0x24    // Drive start holding
#define CMD_START_FREE  0x25    // Drive start holding release/ Drive finish status clear
#define CMD_STP_STS_CLR 0x25    // Drive start holding release/ Drive finish status clear
#define CMD_STOP_DEC    0x26    // Decelerating stop
```

```
#define CMD_STOP_SUDDEN  0x27        // Sudden stop
//(Ⅱ)
#define MC8000P_CMD_DRVFP       CMD_F_DRV_P             // + direction fixed pulse driving
#define MC8000P_CMD_DRVFM       CMD_F_DRV_M             // - direction fixed pulse driving
#define MC8000P_CMD_DRVVP       CMD_C_DRV_P             // + direction continuous pulse driving
#define MC8000P_CMD_DRVVM       CMD_C_DRV_M               // - direction continuous pulse driving
#define MC8000P_CMD_DHOLD       CMD_START_HOLD          // Drive start holding
#define MC8000P_CMD_DFREE       CMD_START_FREE          // Drive start holding release
#define MC8000P_CMD_STSCLR      CMD_STP_STS_CLR           // Drive finish status clear
#define MC8000P_CMD_DRVSBRK     CMD_STOP_DEC              // Decelerating stop
#define MC8000P_CMD_DRVFBRK     CMD_STOP_SUDDEN // Sudden stop

// Interpolation commands          *MCX314As only
//(Ⅰ)
#define CMD_IP_2ST              0x30        // 2-axis linear interpolation
#define CMD_IP_3ST              0x31        // 3-axis linear interpolation
#define CMD_IP_CW               0x32        // CW circular interpolation
#define CMD_IP_CCW              0x33        // CCW circular interpolation
#define CMD_IP_2BP              0x34        // 2-axis bit pattern interpolation
#define CMD_IP_3BP              0x35        // 3-axis bit pattern interpolation
#define CMD_BP_ENABLED          0x36        // BP register data writing enabling
#define CMD_BP_DISABLED         0x37        // BP register data writing disabling
#define CMD_BP_STACK            0x38        // BP data stack
#define CMD_BP_CLR              0x39        // BP data clear
#define CMD_IP_1STEP            0x3A        // Single step interpolation
#define CMD_IP_DEC_VALID        0x3B        // Decelerating enabling
#define CMD_IP_DEC_INVALID              0x3C        // Decelerating disabling
#define CMD_IP_INTRPT_CLR 0x3D        // Interpolation interrupt clear
(Ⅱ)
#define MC8000P_CMD_2CIP                CMD_IP_2ST                      // 2-axis linear interpolation
#define MC8000P_CMD_3CIP                CMD_IP_3ST                      // 3-axis linear interpolation
#define MC8000P_CMD_CIPCW               CMD_IP_CW                       // CW circular interpolation
#define MC8000P_CMD_CIPCCW              CMD_IP_CCW                      // CCW circular interpolation
#define MC8000P_CMD_BPIP2               CMD_IP_2BP                      // 2-axis bit pattern interpolation
#define MC8000P_CMD_BPIP3               CMD_IP_3BP                      // 3-axis bit pattern interpolation
#define MC8000P_CMD_BPENABLED   CMD_BP_ENABLED          // BP register data writing enabling
#define MC8000P_CMD_BPDISABLED  CMD_BP_DISABLED         // BP register data writing disabling
#define MC8000P_CMD_BPSTACK             CMD_BP_STACK                    // BP data stack
#define MC8000P_CMD_BPCLR               CMD_BP_CLR                      // BP data clear
#define MC8000P_CMD_IP1STEP             CMD_IP_1STEP                    // Single step interpolation
#define MC8000P_CMD_DECEN               CMD_IP_DEC_VALID                // Decelerating enabling
#define MC8000P_CMD_DECDIS              CMD_IP_DEC_INVALID              // Decelerating disabling
#define MC8000P_CMD_CLRINTRPT   CMD_IP_INTRPT_CLR       // Interpolation interrupt clear

// Other commands
//(Ⅰ)
#define CMD_HOME_EXEC     0x62        // Automatic home search execution
#define CMD_DEVCTR_CLR    0x63        // Stack counter clear output
#define CMD_SYNC_ACTIVE   0x65        // Synchronous action activation     *MCX314As only
#define CMD_NOP           0x0F        // NOP (for axis switching)
//(Ⅱ)
#define MC8000P_CMD_HMSRC       CMD_HOME_EXEC       // Automatic home search execution
#define MC8000P_CMD_DCC         CMD_DEVCTR_CLR      // Stack counter clear output
#define MC8000P_CMD_SYNCACT     CMD_SYNC_ACTIVE     // Synchronous action activation    *MCX314As only
#define MC8000P_CMD_NOP         CMD_NOP             // NOP (for axis switching)
[C#]
   public enum CMD : int
   {
// Driving commands
```

```
//(Ⅰ)
CMD_F_DRV_P       = 0x20,      // + direction fixed pulse driving
CMD_F_DRV_M       = 0x21,      // – direction fixed pulse driving
CMD_C_DRV_P       = 0x22,      // + direction continuous pulse driving
CMD_C_DRV_M       = 0x23,      // – direction continuous pulse driving
CMD_START_HOLD= 0x24,      // Drive start holding
CMD_START_FREE  = 0x25,      // Drive start holding release/ Drive finish status clear
CMD_STP_STS_CLR= 0x25,      // Drive start holding release/ Drive finish status clear
CMD_STOP_DEC     = 0x26,      // Decelerating stop
CMD_STOP_SUDDEN= 0x27,      // Sudden stop
//(Ⅱ)
MC8000P_CMD_DRVFP          = CMD_F_DRV_P,                // + direction fixed pulse driving
MC8000P_CMD_DRVFM          = CMD_F_DRV_M,                // - direction fixed pulse driving
MC8000P_CMD_DRVVP          = CMD_C_DRV_P,                // + direction continuous pulse driving
MC8000P_CMD_DRVVM          = CMD_C_DRV_M,                // - direction continuous pulse driving
MC8000P_CMD_DHOLD          = CMD_START_HOLD,          // Drive start holding
MC8000P_CMD_DFREE          = CMD_START_FREE,          // Drive start holding release
MC8000P_CMD_STSCLR         = CMD_STP_STS_CLR,         // Drive finish status clear
MC8000P_CMD_DRVSBRK        = CMD_STOP_DEC,            // Decelerating stop
MC8000P_CMD_DRVFBRK        = CMD_STOP_SUDDEN,      // Sudden stop

// Interpolation commands       *MCX314As only.
//(Ⅰ)
CMD_IP_2ST          = 0x30,      // 2-axis linear interpolation
CMD_IP_3ST          = 0x31,      // 3-axis linear interpolation
CMD_IP_CW           = 0x32,      // CW circular interpolation
CMD_IP_CCW          = 0x33,      // CCW circular interpolation
CMD_IP_2BP          = 0x34,      // 2-axis bit pattern interpolation
CMD_IP_3BP          = 0x35,      // 3-axis bit pattern interpolation
CMD_BP_ENABLED              = 0x36,      // BP register data writing enabling
CMD_BP_DISABLED             = 0x37,      // BP register data writing disabling
CMD_BP_STACK      = 0x38,      // BP data stack
CMD_BP_CLR          = 0x39,      // BP data clear
CMD_IP_1STEP        = 0x3A,      // Single step interpolation
CMD_IP_DEC_VALID            = 0x3B,      // Decelerating enabling
CMD_IP_DEC_INVALID          = 0x3C,      // Decelerating disabling
CMD_IP_INTRPT_CLR           = 0x3D,      // Interpolation interrupt clear
//(Ⅱ)
MC8000P_CMD_2CIP            = CMD_IP_2ST,              // 2-axis linear interpolation
MC8000P_CMD_3CIP            = CMD_IP_3ST,              //    3-axis linear interpolation
MC8000P_CMD_CIPCW           = CMD_IP_CW,               // CW circular interpolation
MC8000P_CMD_CIPCCW          = CMD_IP_CCW,              // CCW circular interpolation
MC8000P_CMD_BPIP2           = CMD_IP_2BP,              // 2-axis bit pattern interpolation
MC8000P_CMD_BPIP3           = CMD_IP_3BP,              // 3-axis bit pattern interpolation
MC8000P_CMD_BPENABLED = CMD_BP_ENABLED,          // BP register data writing enabling
MC8000P_CMD_BPDISABLED= CMD_BP_DISABLED,         // BP register data writing disabling
MC8000P_CMD_BPSTACK       = CMD_BP_STACK,            // BP data stack
MC8000P_CMD_BPCLR           = CMD_BP_CLR,              // BP data clear
MC8000P_CMD_IP1STEP         = CMD_IP_1STEP,            // Single step interpolation
MC8000P_CMD_DECEN           = CMD_IP_DEC_VALID,        // Decelerating enabling
MC8000P_CMD_DECDIS          = CMD_IP_DEC_INVALID,      // Decelerating disabling
MC8000P_CMD_CLRINTRPT   = CMD_IP_INTRPT_CLR,       // Interpolation interrupt clear

// Other commands
//(Ⅰ)
CMD_HOME_EXEC               = 0x62,      // Automatic home search execution
CMD_DEVCTR_CLR              = 0x63,      // Stack counter clear output
CMD_SYNC_ACTIVE             = 0x65,      // Synchronous action activation
   }
```

```
//(Ⅱ)
MC8000P_CMD_HMSRC       = CMD_HOME_EXEC,        // Automatic home search execution
MC8000P_CMD_DCC         = CMD_DEVCTR_CLR,       // Deviation counter clear pulse output
MC8000P_CMD_SYNCACT     = CMD_SYNC_ACTIVE,      // Synchronous action activation  ※MCX314As only
MC8000P_CMD_NOP         = CMD_NOP,              // NOP (For axis switching)
}
```

Example) To specify 2-axis liner interpolation drive, CMD.CMD_IP_2ST

⑤Interpolation finishing message, finishing status *The board with MCX314As only

```
[VC]
// Interpolation finishing message
#define WM_BP_END           (WM_USER + 1)   // BP interpolation finishing message
#define WM_CIP_END          (WM_USER + 2)   // Continuous interpolation finishing message

//***** BP Interpolation     Finishing Status *****
// ■ Normal
#define BP_START    0x101   // BP interpolation has started in the background.
#define BP_END      0x102   // BP interpolation has been successfully completed.

// ■ Errors before starting the interpolation
#define BP_CNT_ERR          0x111   // The number of the specified data is out of range.
#define BP_ALREADY_EXEC     0x112   // BP interpolation or continuous interpolation is already running.
#define BP_THREAD_ERR       0x113   // Thread cannot be started.
#define BP_MALLOC_ERR       0x114   // Memory cannot be allocated.
#define BP_PARAM_ERR        0x116   // The parameter is incorrect.
#define BP_NOT_OPEN_ERR     0x117   // The specified board is not opened.
#define BP_OTHER_ERR        0x118   // Other errors.

// ■ Errors during the interpolation driving
#define BP_STOP             0x121   // BP interpolation stopped during driving. (too fast to stack next data)
#define BP_USER_STOP        0x122   // The user aborted BP interpolation.
#define BP_DRIVE_ERR        0x123   // Error occurred in the board during BP interpolation.
                                    (When the error status was set to RR0.)

//***** Continuous Interpolation     Finishing Status *****
// ■ Normal
#define CIP_START           0x201   // Continuous interpolation has started in the background.
#define CIP_END             0x202   // Continuous interpolation has been successfully completed.

// ■ Errors before starting the interpolation
#define CIP_CNT_ERR         0x211   // The number of the specified data is out of range.
#define CIP_ALREADY_EXEC    0x212   // BP interpolation or continuous interpolation is already running.
#define CIP_THREAD_ERR      0x213   // Thread cannot be started.
#define CIP_MALLOC_ERR      0x214   // Memory cannot be allocated.
#define CIP_CMD_ERR         0x215   // Command error (The wrong command was specified by the user.)
#define CIP_PARAM_ERR       0x216   // The parameter is incorrect.
#define CIP_NOT_OPEN_ERR    0x217   // The specified board is not opened.
#define CIP_OTHER_ERR       0x218   // Other errors.

// ■ Errors during the interpolation driving
#define CIP_STOP            0x221   // Continuous interpolation stopped during driving. (too fast to set next data)
#define CIP_USER_STOP       0x222   // The user aborted Continuous interpolation.
#define CIP_DRIVE_ERR       0x223   // Error occurred in the board during Continuous interpolation.
                                    (When the error status was set to RR0.)

 [C#]
  public enum MSG_ID : int
  {         // Interpolation finishing message
```

```
        WM_USER         =0x0400,
        WM_BP_END       =WM_USER+1,     // (WM_USER + 1) BP interpolation finishing message
        WM_CIP_END      =WM_USER+2      // (WM_USER + 2) Continuous interpolation finishing message
  }
    Example) MSG_ID and WM_BP_END MSG_ID.WM_BP_END

  public enum Nmc_Status : uint
  {
//***** BP Interpolation    Finishing Status *****
// ■ Normal
BP_START        = 0x101   ,     // BP interpolation has started in the background.
BP_END          = 0x102   ,     // BP interpolation has been successfully completed.

// ■ Errors before starting the interpolation
BP_CNT_ERR      = 0x111   ,     // The number of the specified data is out of range.
BP_ALREADY_EXEC         = 0x112   ,     // BP interpolation or continuous interpolation is already running.
BP_THREAD_ERR   = 0x113   ,     // Thread cannot be started.
BP_MALLOC_ERR   = 0x114   ,     // Memory cannot be allocated.

// ■ Errors during the interpolation driving
BP_STOP         =0x121    ,     // BP interpolation stopped during driving. (too fast to stack next data)
BP_USER_STOP    = 0x122   ,     // The user aborted BP interpolation.
BP_DRIVE_ERR    = 0x123   ,     // Error occurred in the board during BP interpolation.
                                  (When the error status was set to RR0.)

//***** Continuous Interpolation    Finishing Status *****
// ■ Normal
CIP_START       = 0x201   ,     // Continuous interpolation has started in the background.
CIP_END         = 0x202   ,     // Continuous interpolation has been successfully completed.

// ■ Errors before starting the interpolation
CIP_CNT_ERR     = 0x211   ,     // The number of the specified data is out of range.
CIP_ALREADY_EXEC        = 0x212   ,     // BP interpolation or continuous interpolation is already running.
CIP_THREAD_ERR  = 0x213   ,     // Thread cannot be started.
CIP_MALLOC_ERR  = 0x214   ,     // Memory cannot be allocated.
CIP_CMD_ERR     = 0x215   ,     // Command error (The wrong command was specified by the user.)

// ■ Errors during the interpolation driving
CIP_STOP        = 0x221   ,     // Continuous interpolation stopped during driving. (too fast to set next data)
CIP_USER_STOP   = 0x222   ,     // The user aborted Continuous interpolation.
CIP_DRIVE_ERR   = 0x223   ,     // Error occurred in the board during Continuous interpolation.
                                    (When the error status was set to RR0.)
  }
    Example) When BP interpolation has started in the background,       Nmc_Status.BP_START

  public enum IP_AXIS : int
  {     // Axis for interpolation
        IP_X    =0,     // Assign X axis for interpolation.
        IP_Y    =1,     // Assign Y axis for interpolation.
        IP_Z    =2,     // Assign Z axis for interpolation.
        IP_U    =3      // Assign U axis for interpolation.
  }
    Example) To assign X axis for interpolation,        IP_AXIS.IP_X

⑥Constant definition
[C#]
  public enum CONST : int
  {
        MAX_BOARD_MC8000P       =16     // The maximum number of boards MC8000P device driver can recognize
```

simultaneously.
}
Example) To specify the maximum number of MC8000P boards, CONST.MAX_BOARD_MC8000P

public enum MaxValue : uint
{
MCX304 =268435455, // The maximum value of MCX304 output pulse
MCX314As =4294967295 // The maximum value of MCX314As output pulse
}
Example) To specify the maximum value of MCX304 output pulse, MaxValue.MCX304

⑦IC
[C#]
public enum IC : int
{
A =0, // The first IC
B =1, // The second IC
MCX304 =0, // MCX304
MCX314AS =1 // MCX314AS
}
Example) To specify IC A, IC.A

(2) The method of axis assignment is as follows:

Axis	VC	C#
X	AXIS_X	AXIS.X
Y	AXIS_Y	AXIS.Y
Z	AXIS_Z	AXIS.Z
U	AXIS_U	AXIS.U
All	AXIS_ALL	AXIS.ALL

①To assign 1 axis
Specify one of the following axes: AXIS_X, AXIS_Y, AXIS_Z, AXIS_U.

Example) Set 1000 as the drive speed of X axis.
[VC] Nmc_Speed(No, IcNo, AXIS_X, 1000);
[VB] Call Nmc_Speed(No, IcNo, AXIS_X, 1000)
[C#] MC8000P.Nmc_Speed(No, IcNo, AXIS.X, 1000);

②To assign 2 axes
Use Bit OR operator.
For instance, if the user tries to assign X and Y axes simultaneously,
[VC]・・・Specify AXIS_X | AXIS_Y.
[VB]・・・Specify AXIS_X Or AXIS_Y.
[C#]・・・Specify AXIS.X | AXIS.Y.

Example) Set 1000 as the drive speed of X and Y axes.
[VC] Nmc_Speed(No, IcNo, AXIS_X | AXIS_Y, 1000);
[VB] Call Nmc_Speed(No, IcNo, AXIS_X Or AXIS_Y, 1000)
[C#] MC8000P.Nmc_Speed(No, IcNo, AXIS.X | AXIS.Y, 1000);

③To assign 3 axes
Use Bit OR operator.
For instance, if the user tries to assign X, Y and Z axes simultaneously,
[VC]・・・Specify AXIS_X | AXIS_Y | AXIS_Z.
[VB]・・・Specify AXIS_X Or AXIS_Y Or AXIS_Z.
[C#]・・・Specify AXIS.X | AXIS.Y | AXIS.Z.

Example) Set 1000 as the drive speed of X, Y and Z axes.

[VC] Nmc_Speed(No, IcNo, AXIS_X | AXIS_Y | AXIS_Z, 1000);
[VB] Call Nmc_Speed(No, IcNo, AXIS_X Or AXIS_Y Or AXIS_Z, 1000)
[C#] MC8000P.Nmc_Speed(No, IcNo, AXIS.X | AXIS.Y | AXIS.Z, 1000);

④To assign all axes
Specify AXIS_ALL.

Example) Set 1000 as the drive speed of all axes.
[VC] Nmc_Speed(No, IcNo, AXIS_ALL, 1000);
[VB] Call Nmc_Speed(No, IcNo, AXIS_ALL, 1000)
[C#] MC8000P.Nmc_Speed(No, IcNo, AXIS.ALL, 1000);

(3) The structure (user-defined type in VB.NET) used in the interpolation function is defined as follows:
*The board with MCX314As only

①VC

```
// 2-axis BP interpolation
typedef struct _DATA_2BP
{
        USHORT Bp1p;            // BP1P data
        USHORT Bp1m;            // BP1M data
        USHORT Bp2p;            // BP2P data
        USHORT Bp2m;            // BP2M data
} DATA_2BP;

// 3-axis BP interpolation
typedef struct _DATA_3BP
{
        USHORT Bp1p;            // BP1P data
        USHORT Bp1m;            // BP1M data
        USHORT Bp2p;            // BP2P data
        USHORT Bp2m;            // BP2M data
        USHORT Bp3p;            // BP3P data
        USHORT Bp3m;            // BP3M data
} DATA_3BP;

// 2-axis continuous interpolation
typedef struct _DATA_2CIP
{
        USHORT  Command;// Command number (Set one of CMD_IP_2ST, CMD_IP_CW, CMD_IP_CCW.)
        USHORT  Speed;  // Speed (When changing the speed, set 1~8000. When not changing, set 0.)
        long    EndP1;  // Finishing point (The first axis)
        long    EndP2;  // Finishing point (The second axis)
        long    Center1;// Circular center point (The first axis)
        long    Center2;// Circular center point (The second axis)
} DATA_2CIP;            // Note: The first or second axis must be specified by WR5.

// 3-axis continuous interpolation
typedef struct _DATA_3CIP
{
        long    EndP1;  // Finishing point (The first axis)
        long    EndP2;  // Finishing point (The second axis)
        long    EndP3;  // Finishing point (The third axis)
        USHORT  Speed;  // Speed (When changing the speed, set 1~8000. When not changing, set 0.)
} DATA_3CIP;            // Note: The first or second axis must be specified by WR5.
```

②VB.NET

```
' 2-axis BP interpolation
Structure DATA_2BP
```

```
        Dim Bp1p As Short    ' BP1P data
        Dim Bp1m As Short    ' BP1M data
        Dim Bp2p As Short    ' BP2P data
        Dim Bp2m As Short    ' BP2M data
End Structure

' 3-axis BP interpolation
Structure DATA_3BP
        Dim Bp1p As Short    ' BP1P data
        Dim Bp1m As Short    ' BP1M data
        Dim Bp2p As Short    ' BP2P data
        Dim Bp2m As Short    ' BP2M data
        Dim Bp3p As Short    ' BP3P data
        Dim Bp3m As Short    ' BP3M data
End Structure

' 2-axis continuous interpolation
Structure DATA_2CIP
        Dim Cmd As Short              ' Command number (Set one of CMD_IP_2ST, CMD_IP_CW, CMD_IP_CCW.)
        Dim Speed As Short            ' Speed (When changing the speed, set 1~8000. When not changing, set 0.)
        Dim EndP1 As Integer          ' Finishing point (The first axis)
        Dim EndP2 As Integer          ' Finishing point (The second axis)
        Dim Center1 As Integer        ' Circular center point (The first axis)
        Dim Center2 As Integer        ' Circular center point (The second axis)
End Structure        ' Note: The first or second axis must be specified by WR5.

' 3-axis continuous interpolation
Structure DATA_3CIP
        Dim EndP1 As Integer          ' Finishing point (The first axis)
        Dim EndP2 As Integer          ' Finishing point (The second axis)
        Dim EndP3 As Integer          ' Finishing point (The third axis)
        Dim Speed As Short            ' Speed (When changing the speed, set 1~8000. When not changing, set 0.)
        End Structure                 ' Note: The first, second or third axis must be specified by WR5.
```

③C#

```
// 2-axis BP interpolation
[StructLayout(LayoutKind.Sequential)]
public struct DATA_2BP
{
        public    DATA_2BP(ushort bp1p,ushort bp1m,ushort bp2p,ushort bp2m)
        {
                this.Bp1p  = bp1p;
                this.Bp1m = bp1m;
                this.Bp2p  = bp2p;
                this.Bp2m = bp2m;
        }
        public    ushort   Bp1p;        // BP1P data
        public    ushort   Bp1m;        // BP1M data
        public    ushort   Bp2p;        // BP2P data
        public    ushort   Bp2m;        // BP2M data
}

// 3-axis BP interpolation
[StructLayout(LayoutKind.Sequential)]
public struct DATA_3BP
{
        public    DATA_3BP(ushort bp1p,ushort bp1m,ushort bp2p,ushort bp2m,ushort bp3p,ushort bp3m)
        {
                this.Bp1p  = bp1p;
```

```
                this.Bp1m = bp1m;
                this.Bp2p  = bp2p;
                this.Bp2m = bp2m;
                this.Bp3p  = bp3p;
                this.Bp3m = bp3m;
        }
        public    ushort   Bp1p;        // BP1P data
        public    ushort   Bp1m;        // BP1M data
        public    ushort   Bp2p;        // BP2P data
        public    ushort   Bp2m;        // BP2M data
        public    ushort   Bp3p;        // BP3P data
        public    ushort   Bp3m;        // BP3M data
}

// 2-axis continuous interpolation
[StructLayout(LayoutKind.Sequential)]
public struct DATA_2CIP
{
        public    DATA_2CIP(ushort command,ushort speed,int endp1,int endp2,int center1,int center2)
        {
                this.Command        = command;
                this.Speed = speed;
                this.EndP1 = endp1;
                this.EndP2 = endp2;
                this.Center1        = center1;
                this.Center2        = center2;
        }
        public    ushort   Command;   // Command number (Set one of CMD_IP_2ST, CMD_IP_CW, CMD_IP_CCW.)
        public    ushort   Speed;     // Speed (When specifying the speed, specify 1~8000. When not specifying, set 0.)
        public    int   EndP1;        // Finishing point (The first axis)
        public    int   EndP2;        // Finishing point (The second axis)
        public    int   Center1;      // Circular center point (The first axis)
        public    int   Center2;      // Circular center point (The second axis)
}                           // Note: The first or second axis must be specified by WR5.

// 3-axis continuous interpolation
 [StructLayout(LayoutKind.Sequential)]
public struct DATA_3CIP
{
        public    DATA_3CIP(int endp1,int endp2,int endp3,ushort speed)
        {
                this.EndP1 = endp1;
                this.EndP2 = endp2;
                this.EndP3 = endp3;
                this.Speed = speed;
        }
        public    int   EndP1;        // Finishing point (The first axis)
        public    int   EndP2;        // Finishing point (The second axis)
        public    int   EndP3;        // Finishing point (The third axis)
        public    ushort   Speed;     // Speed (When specifying the speed, specify 1~8000. When not specifying, set 0.)
        }                   // Note: The first, second or third axis must be specified by WR5.
```

Examples of structure are as follows:

Example 1) Not define and initialize at the same time

```
DATA_2BP [] Data2Bp = new DATA_2BP[4];          // 2-axis BP interpolation

// Interpolation data setting
```

```
Data2Bp[0].Bp1p = 0xFF30;    // 1111 1111 0011 0000   BP1 + direction      10 pulse
Data2Bp[0].Bp1m = 0;         // 0000 0000 0000 0000   BP1 - direction       0 pulse
Data2Bp[0].Bp2p = 0;         // 0000 0000 0000 0000   BP2 + direction       0 pulse
Data2Bp[0].Bp2m = 0x84FF;    // 1000 0100 1111 1111   BP2 - direction      10 pulse

Data2Bp[1].Bp1p = 0xAC35;    // 1010 1100 0011 0101   BP1 + direction       8 pulse
Data2Bp[1].Bp1m = 0;         // 0000 0000 0000 0000   BP1 - direction       0 pulse
Data2Bp[1].Bp2p = 0xC000;    // 1100 0000 0000 0000   BP2 + direction       2 pulse
Data2Bp[1].Bp2m = 0x36E7;    // 0011 0110 1110 0111   BP2 - direction      10 pulse

Data2Bp[2].Bp1p = 0x3F3F;    // 0011 1111 0011 1111   BP1 + direction      12 pulse
Data2Bp[2].Bp1m = 0xC000;    // 1100 0000 0000 0000   BP1 - direction       2 pulse
Data2Bp[2].Bp2p = 0xFBDA;    // 1111 1011 1101 1010   BP2 + direction      12 pulse
Data2Bp[2].Bp2m = 0;         // 0000 0000 0000 0000   BP2 - direction       0 pulse

Data2Bp[3].Bp1p = 0;         // 0000 0000 0000 0000   BP1 + direction       0 pulse
Data2Bp[3].Bp1m = 0x1CF2;    // 0001 1100 1111 0010   BP1 - direction       8 pulse
Data2Bp[3].Bp2p = 0xFFFF;    // 1111 1111 1111 1111   BP2 + direction      16 pulse
Data2Bp[3].Bp2m = 0;         // 0000 0000 0000 0000   BP2 - direction       0 pulse
```

Example 2) Define and initialize at the same time

```
// 2-aixs BP interpolation data      BP1P,    BP1M,    BP2P,    BP2M      (Data for Fig.2.32 in MCX314As Manual)
DATA_2BP[] Data2Bp = new DATA_2BP[]
{
        new DATA_2BP(0x0000, 0x2BFF, 0xFFD4, 0x0000),
        new DATA_2BP(0xF6FE, 0x0000, 0x000F, 0x3FC0),
        new DATA_2BP(0x1FDB, 0x0000, 0x00FF, 0xFC00),
        new DATA_2BP(0x4000, 0x7FF5, 0x0000, 0x0AFF),
};
```

(4) The interpolation axis (IpAxis) specified by the interpolation function is as follows: *The board with MCX314As only
Set the interpolation axis data from X, Y, Z and U axis to the lower 6-bit of 16-bit data. For 2-axis interpolation, set to the first and second axis, for 3-axis interpolation, set to the first, second and third axis.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	AX31	AX30	AX21	AX20	AX11	AX10

Third axis Second axis First axis

◆ Description of each bit

D1, 0 AX11, 10 Specify the first axis (main axis) for interpolation driving. Axis codes are as follows:

Axis	Code (Binary)
X	00
Y	01
Z	10
U	11

Example of the first axis: X, Second axis: Y, Third axis: Z

D5	D4	D3	D2	D1	D0
1	0	0	1	0	0

D3, 2 AX21, 20 Specify the second axis using the code in the table above for interpolation driving.

D5, 4 AX31, 30 Specify the third axis using the code in the table above for 3-axis interpolation driving.
This is not used in 2-axis interpolation driving, so it doesn't matter to set any code.

◆ For C#, specify the axis for interpolation by using the following default value.

```
public enum IP_AXIS : int
{
        IP_X    =0,     // Assign X axis for interpolation.
        IP_Y    =1,     // Assign Y axis for interpolation.
        IP_Z    =2,     // Assign Z axis for interpolation.
        IP_U    =3,     // Assign U axis for interpolation.
}
```

Axis	Example
X	IP_AXIS.IP_X
Y	IP_AXIS.IP_Y
Z	IP_AXIS.IP_Z
U	IP_AXIS.IP_U

To specify the axis for interpolation, set it by using bit operations as follows.

Example) For 3-axis interpolation of the first axis: X, the second axis: Y and the third axis: Z, set them by using bit operations as follows.

```
IpAxis = IP_AXIS.IP_X | IP_AXIS.IP_Y<<2 | IP_AXIS.IP_Z<<4 ;
```

(5) About the speed change when continuous interpolation function is executed. *The board with MCX314As only
Continuous interpolation function can change the speed during interpolation driving. The user can set the speed to each segment. To change the speed during interpolation driving, set TRUE(True) to the function parameter SpdChgFlg.

◆ How to change the speed for each segment
Set the speed of each segment to the Speed of DATA_2CIP or the Speed of DATA_3CIP.
- When set the different speed from the previous segment, set 1~8000.
- When set the same speed as the previous segment, set 0.

◆ About the timing of changing the speed
When the Speed of DATA_2CIP or the Speed of DATA_3CIP is set to 1~8000, the process of interpolation function is described as follows:

For the first segment, set the speed before executing the segment.
For the second or later segment, set the speed right after that segment has started (when the next segment is ready to be written).
For instance, when the second segment starts and the third segment is ready to be written, set the speed of the second segment. Therefore, after the second segment has started, the speed of the first segment is applied until the speed of the second segment is set.

(6) Address assignment
The address of Nmc_OutPort and Nmc_InPort function should be assigned I/O address described in each board manual.

I/O addresses such as Write register or Read register are as follows. See each board manual for more details.

Board	Address of Write register	Address of Read register	Address of MSM82C55	Address of PIX132
MC8082P	0~F	0~F	-----	14~16
MC8042P	0~F	0~F	-----	14~16
MC8022P	0~F	0~F	-----	14~16
MC8043P	0~7	0~7	-----	-----
MC8082Pe	0~F	0~F	-----	14~16
MC8043Pe	0~7	0~7	-----	-----

Note1: The address is given in hexadecimal.
Note2: It is easy to access the write or read register by using not Nmc_OutPort or Nmc_InPort function but the functions for write or read registers.

(7) IC number assignment
①When the board has one IC, set 0.
②When the board has 2 ICs, set the following value.
・IC-A: 0
・IC-B: 1
The number of IC chips and the assignment of IC number for each board are shown in the table below.

Board	Number of IC	IC number
MC8082P	2	0, 1
MC8042P	1	0
MC8022P	1	0
MC8043P	1	0
MC8082Pe	2	0, 1
MC8043Pe	1	0

Note: IC indicates IC chips such as MCX314As or MCX304.

4.1.4. Usage

■ API Function Declaration

API function declaration is defined in the following files.

VC++, VC++.NET	MC8000P_DLL.H
VB.NET	MC8000P _DLL.vb
C#.NET	Refer to User's Manual or IntelliSense.

■ Usage

(1) Start process・・・Execute Nmc_Open once before using each function.
(2) End process・・・Execute Nmc_Close or Nmc_CloseAll at the end of program.

■ Board Number

The board number specified from an application should be as follows.

	Rotary switch value of board	Board number should be specified (decimal number)
1	0 ~ 9	0 ~ 9
2	A ~ F	10 ~ 15

■ Notes for Use of Function

(1) About VC, VB.NET, C# (all languages)
①When each function is used before executing Nmc_Open function, operation is not guaranteed.
②When the board number, which is not connected, is assigned, the operation of each function is not guaranteed.
③Do not access the one board from 2 or more applications at the same time (such as Nmc_Open).

(2) VC, C# only
①When using the interrupt handling function, the time from the interrupt generation to user-defined function is not guaranteed by the nature of Windows.
②When the user tries to perform the interrupt, do not execute the close handling (Nmc_Close or Nmc_CloseAll) while the interrupt user-defined function (the function designated by Nmc_SetEvent) is running. Before executing the close handling, make sure that the interrupt user-defined function is finished.

■ How to handle an interrupt by VC

①Set the interrupt by using Nmc_Open function.

```
Nmc_Open(No, TRUE);          // The second argument   TRUE: interrupt   FALSE: not interrupt
```

②Set the user-defined function handling an interrupt by using Nmc_SetEvent function, and set which interrupt is allowed or not.

```
Nmc_SetEvent(No, MC_EventProc, IpParam);        // Set the user function address and argument.
Nmc_WriteReg(No, IcNo, AXIS_ALL, 0x8000);       // Interrupt occurs at the stop of the specified IC (All axes).
```

③If an interrupt occurs, the interrupt user-defined function set by Nmc_SetEvent is called.
The interrupt user-defined function can check the interrupt factor. To read the interrupt factor of RR3, use Nmc_ReadEvent function.

■ The example of the interrupt user-defined function

```
DWORD WINAPI MC_EventProc(LPVOID IpParam)
{
    ・・・・
   long Rr3X, Rr3Y, Rr3Z, Rr3U;
   Nmc_ReadEvent(No, IcNo, &Rr3X, &Rr3Y, &Rr3Z, &Rr3U);
    ・・・・
   return 0;
}
```

④Use Nmc_ResetEvent to release the interrupt user-defined function. By executing this function, the user-defined function is not called if an interrupt occurs in the board.

```
Nmc_ResetEvent(No);
```

■ How to handle an interrupt by C# (C# only)

①Set the method to handle an interrupt by using MC8000P.Nmc_SetEvent function. When multiple boards are used, example is as follows.

(When the board number is 0)

```
MC8000P.callback[0] = new MC8000P.User Thread(isr);          // isr is the interrupt user-defined function.
bool ret = MC8000P.Nmc_SetEvent(0, MC8000P.callback[0]);    // Board number 0 is specified to the first argument.
                                                            // Set the function address.
MC8000P.Nmc_WriteReg1(0, (int)IC.A, AXIS.ALL, 0x8000);      // Generate an interrupt at the stop (All axes).
        ・・・
```

(When the board number is 1)

```
MC8000P.callback[1] = new MC8000P.User Thread(isr2);         // isr is the interrupt user-defined function.
bool ret = MC8000P.Nmc_SetEvent(1, MC8000P.callback[1]);    // Board number 1 is specified to the first argument.
                                                            // Set the function address.
MC8000P.Nmc_WriteReg1(1, (int)IC.A, AXIS.ALL, 0x8000);      // Generate an interrupt at the stop (All axes).
        ・・・
```

②The interrupt handling function can check the interrupt factor. To read the interrupt factor of RR3, use MC8000P.Nmc_ReadEvent function.

(Interrupt handling process of the board number 0)

```
static void isr()
    {
        int Rr3X, Rr3Y, Rr3Z, Rr3U;
        MC8000P.Nmc_ReadEvent(0, (int)IC.A, out Rr3X, out Rr3Y, out Rr3Z, out Rr3U);
        ・・・・
    }
```

(Interrupt handling process of the board number 1)

```
static void isr2()
    {
        int Rr3X, Rr3Y, Rr3Z, Rr3U;
        MC8000P.Nmc_ReadEvent(1, (int)IC.A, out Rr3X, out Rr3Y, out Rr3Z, out Rr3U);
        ・・・・
    }
```

③Use MC8000P.Nmc_ResetEvent method to release the interrupt handling function. By executing this function, the interrupt user-defined function method is not called if an interrupt occurs in the board.

■ Continuous Interpolation *The board with MCX314As only

When executing the continuous interpolation, please read the chapter “2.4.5 Continuous Interpolation” of MCX314As user’s manual carefully and execute the process described in the chapter in the application. Continuous Interpolation Functions [*1] execute some of the process by DLL. So they will be used to execute the process of the continuous interpolation. But there are some notes when using Continuous Interpolation Functions. Please note them.

[*1] Nmc_2CIPExec, Nmc_3CIPExec, Nmc_2CIPExec_BG, Nmc_3CIPExec_BG

Notes for when using Continuous Interpolation Function:
Continuous interpolation function writes the next segment data such as the finish point, the center point, etc. and writes the interpolation command, and checks the error. If the error occurs, the function returns. If not, it will check whether the data of the next segment is writable or not (check the bit D9 of RR0). When it becomes writable, it will write the data of the next segment and the command of the interpolation. This function repeats the process until the continuous interpolation is completed.
Because the loops that check the error and check whether the next segment data is writable are always executed in DLL, the use of this function isn’t suitable if you want to execute other process by the application during this function is executing. In this case the continuous interpolation function shouldn’t be used and the user must make the source code of the continuous interpolation in the applications by referring to MCX314As user’s manual. Please see the chapter “2.4.6 The Acceleration / Deceleration Control in Interpolation” of MCX314As user’s manual about the acceleration/deceleration drive of continuous interpolation.
When using continuous interpolation function, the initial speed should be set 8,000. (Don’t change the initial speed during this function is executing.) In this case the fixed speed driving mode is applied in each segment.

■ The BP (Bit Pattern) Interpolation *The board with MCX314As only

When executing the BP interpolation, please read the chapter “2.4.3 The Bit Pattern Interpolation” of MCX314As user’s manual carefully and execute the process described in the chapter in the application. The BP interpolation functions [*2] execute some of the process by DLL. So, they will be used to execute the process of the BP interpolation. But there are some notes when using the BP interpolation functions. Please note them.

[*2] Nmc_2BPExec, Nmc_3BPExec, Nmc_2BPExec_BG, Nmc_3BPExec_BG

Notes for when using the BP Interpolation Function
The BP interpolation function writes the next BP data and the interpolation command and checks the error. If the error occurs, the function returns. If not, it will check whether the stack counter becomes 2 or less (check the bit D14,13 of RR0). When it becomes 2 or less, it will write the next BP data. This function repeats the process until the BP interpolation is completed. Because the loops that check the error and check the stack counter are always executed in DLL, the use of this function isn’t suitable if you want to execute other process by the application during this function is executing. In this case the BP interpolation function shouldn’t be used and the user must make the source code of the BP interpolation in the applications by referring to MCX314As user’s manual. Please see chapter “2.4.6 The Acceleration / Deceleration Control in Interpolation” of MCX314As user’s manual about the acceleration/deceleration drive of BP interpolation.

■ Notes for Use of Interpolation Function *The board with MCX314As only

(1) Concerning the following interpolation function, the user can execute only one interpolation function at once.
While executing the interpolation function, the other interpolation function cannot be executed. If executed, an error will return.

Nmc_2BPExec	Nmc_2BPExec_BG	Nmc_2CIPExec	Nmc_2CIPExec_BG
Nmc_3BPExec	Nmc_3BPExec_BG	Nmc_3CIPExec	Nmc_3CIPExec_BG

(2) While executing the above interpolation function, do not perform the following operation.

①Execution of the interpolation command (30h~3Dh)
②Change of WR5 interpolation mode register

(3) The following interpolation function is executed in the background, so that the memory for interpolation data is allocated at the start of interpolation function and then the interpolation data specified by the user is copied. Then, when the interpolation process in the background is finished, the memory will be released and the message will be sent to the user window.
Therefore, while executing the following interpolation function in the background, do not exit the application.

Then, while executing the following interpolation function in the background, do not execute the close handling (Nmc_Close or Nmc_CloseAll).
If you want to stop the execution of interpolation function, execute the interpolation stop function (Nmc_IPStop) and make sure to receive the stop message.

Nmc_2BPExec_BG	Nmc_2CIPExec_BG
Nmc_3BPExec_BG	Nmc_3CIPExec_BG

(4) While the interpolation function is executed and when the speed is too fast, the interpolation driving may stop before setting the next data.

■ Notes for when developing multithread applications

This chapter describes the notes for developing applications which work in multithread.

In Nmc_xxx function, there are functions executing axis switching, data writing into WR6, WR7 and data reading to RR6, RR7. Each Nmc_xxx function is as follows:

◆ Functions executing axis switching

Nmc_Reset	Nmc_Command	Nmc_Command_IP			
Nmc_WriteReg0	Nmc_WriteReg1	Nmc_WriteReg2	Nmc_WriteReg3		
Nmc_ReadReg1	Nmc_ReadReg2				
Nmc_Range	Nmc_Jerk	Nmc_Acc	Nmc_Dec	Nmc_StartSpd	Nmc_Speed
Nmc_Pulse	Nmc_Pulse_VB	Nmc_DecP	Nmc_DecP_VB	Nmc_Center	Nmc_Lp
Nmc_Ep	Nmc_CompP	Nmc_CompM	Nmc_AccOfst	Nmc_DJerk	Nmc_HomeSpd
Nmc_ExpMode	Nmc_SyncMode	Nmc_HomeMode			
Nmc_ReadLp	Nmc_ReadEp	Nmc_ReadSpeed	Nmc_ReadAccDec	Nmc_ReadSyncBuff	
Nmc_2BPExec	Nmc_3BPExec	Nmc_2BPExec_BG	Nmc_3BPExec_BG		
Nmc_2CIPExec	Nmc_3CIPExec	Nmc_2CIPExec_BG	Nmc_3CIPExec_BG		
Nmc_WriteRegSetAxis	Nmc_ReadRegSetAxis		Nmc_WriteData	Nmc_WriteData2	Nmc_ReadData

◆ Functions executing data writing into WR6, WR7

Nmc_Range	Nmc_Jerk	Nmc_Acc	Nmc_Dec	Nmc_StartSpd	Nmc_Speed
Nmc_Pulse	Nmc_Pulse_VB	Nmc_DecP	Nmc_DecP_VB	Nmc_Center	Nmc_Lp
Nmc_Ep	Nmc_CompP	Nmc_CompM	Nmc_AccOfst	Nmc_DJerk	Nmc_HomeSpd
Nmc_ExpMode	Nmc_SyncMode	Nmc_HomeMode	Nmc_WriteData	Nmc_WriteData2	
Nmc_2CIPExec	Nmc_3CIPExec	Nmc_2CIPExec_BG	Nmc_3CIPExec_BG		
Nmc_WriteReg6	Nmc_WriteReg7				

When writing into WR6, WR7 by Nmc_OutPort or Nmc_WriteReg.

◆ Functions executing data reading to RR6, RR7

Nmc_ReadLp	Nmc_ReadEp	Nmc_ReadSpeed	Nmc_ReadAccDec	Nmc_ReadSyncBuff	Nmc_ReadData

To perform WR1~WR3 writing, RR1~RR2 reading, data writing command and data reading command, basically use the following Nmc_xxx function.

◆ WR1~WR3 writing

Nmc_WriteReg1	Nmc_WriteReg2	Nmc_WriteReg3	Nmc_WriteRegSetAxis

◆ RR1~RR2 reading

Nmc_ReadReg1	Nmc_ReadReg2	Nmc_ReadRegSetAxis

◆ Data writing command

Nmc_Range	Nmc_Jerk	Nmc_Acc	Nmc_Dec	Nmc_StartSpd	Nmc_Speed
Nmc_Pulse	Nmc_Pulse_VB	Nmc_DecP	Nmc_DecP_VB	Nmc_Center	Nmc_Lp
Nmc_Ep	Nmc_CompP	Nmc_CompM	Nmc_AccOfst	Nmc_DJerk	Nmc_HomeSpd
Nmc_ExpMode	Nmc_SyncMode	Nmc_HomeMode	Nmc_WriteData	Nmc_WriteData2	

◆ Data reading command
Nmc_ReadLp Nmc_ReadEp Nmc_ReadSpeed Nmc_ReadAccDec Nmc_ReadSyncBuff Nmc_ReadData

When trying to execute the commands such as WR1~WR3 data writing, RR1~RR2 data reading, data writing and data reading, the user must take care in multithread environment if the same operations are performed without these functions.

(1) For example, when writing into WR1, Nmc_WriteReg1 is used; however, there is other way as follows:

```
①Nmc_OutPort(No, MCX_WR0, 0x010F);          // Switch to X axis (IC-A).
②Nmc_OutPort (No, MCX_WR1, Data);           // Write to WR1 (IC-A).
```

Also, the following functions can perform the same operation.

```
③Nmc_WriteReg(No, IcNo, MCX_WR0, 0x010F);   // Switch to X axis.
④Nmc_WriteReg (No, IcNo, MCX_WR1, Data);    // Write to WR1.
```

In this case, if Nmc_xxx function to switch the axis is executed between ① and ② or ③ and ④, the data will be written into WR1 of a different axis.

(2) For example, when setting the speed, Nmc_Speed is used; however, there is other way as follows:

```
①Nmc_OutPort(No, MCX_WR6, Data);            // Write to WR6 (IC-A).
②Nmc_OutPort(No, MCX_WR0, 0x0105);          // Set WR6 data to the speed of X axis (IC-A).
```

Also, the following functions can perform the same operation.

```
③Nmc_WriteReg6(No, IcNo, Data);             // Write to WR6.
④Nmc_Command(No, IcNo, AXIS_X, 0x05);       // Set WR6 data as the speed of X axis.
```

In this case, if Nmc_xxx function to write data into WR6, WR7 is executed between ① and ② or ③ and ④, the other data will be set as the speed.

(3) For example, when reading logical position counter, Nmc_ReadLp is used; however, there is other way as follows:

```
①Nmc_OutPort(No, MCX_WR0, 0x0110);          // Read logical position counter of X axis to RR6, RR7 (IC-A).
②d6 = Nmc_InPort(No, MCX_RR6);              // Read from RR6 (IC-A).
③d7 = Nmc_InPort(No, MCX_RR7);              // Read from RR7 (IC-A).
```

In this case, if Nmc_xxx function to read data to RR6, RR7 is executed between ① and ② or ② and ③, the different data will be read.

Thus, in multithread environment, when calling API function more than twice to execute the objective operation, the user needs not to perform such an operation or needs to take exclusive control.

When the operation is finished by calling Nmc_xxx function once, it properly works in multithread environment.
Each function of Nmc_xxx takes exclusive control each other.

4.2 Notes on Programming

(1) Initial setting of input signal filter

■ The board equipped with MCX314As

Each input signal of the board, for example, a limit signal, uses the built-in integral filter of MCX314As. The device driver provided by NOVA electronics sets the filter as shown below for each input signal by writing extension mode setting command (60h) to MCX314As by default when PC is powered on.

Filter delay time: 512 μ sec

Each Input Signal Filter Enable/Disable:

Signal Name	Enable / Disable
EMG, nLMT+, nLMT-, nIN0, nIN1	Enable
nIN2	Enable
nINPOS, nALARM	Enable
nEXOP+, nEXOP-	Enable
nIN3	Enable

To switch Enable/Disable of these input signal filters on the application, see chapter 6.16 of MCX314As user's manual. It can be changed by extension mode setting command (60h). The following example shows that all ICs of all axes (X, Y, Z and U axes) of the board number 0 are set to the same setting as the table above. Nmc_ExpMode executes extension mode setting command (60h).

Example 1)

```
Nmc_ExpMode(0, 0, AXIS_ALL, 0x5F00, 0x0000);
```

Example 2)

```
Nmc_WriteReg6(0, 0, 0x5F00);
Nmc_WriteReg7(0, 0, 0x0000);
Nmc_WriteReg0(0, 0, 0x0F60);
```

Notes:

①Extension mode setting command (60h) also sets the automatic home search (WR7) setting with input signal filter (WR6) setting. If the user tries to set either, be sure to set the proper value to both WR6 and WR7.

②When the user executes soft reset (set 1 to WR0/D15) in the application, the device driver sets the above setting, the same setting as the above table, to the filter of each input signal.

■ The board equipped with MCX304

Each input signal of the board, for example, a limit signal, uses the built-in integral filter of MCX304. The device driver provided by NOVA electronics sets the filter as shown below for each input signal by writing the data into mode register 3 (WR3) of MCX304 by default when PC is powered on.

Filter delay time: 512 μ sec

Each Input Signal Filter Enable/Disable:

Signal Name	Enable / Disable
EMG, nLMT+, nLMT-, nSTOP0, nSTOP1	Enable
nSTOP2	Enable
nINPOS, nALARM	Enable
nEXOP+, nEXOP-	Enable

To switch Enable/Disable of these input signal filters on the application, see chapter 2.6.9 and 4.6 of MCX304 user's manual. It can be changed by writing the data into mode register 3 (WR3). The following example shows that all ICs of all axes (X, Y, Z and U axes) of the board number 0 are set to the same setting as the table above.

Example)

```
Nmc_WriteReg3(0, 0, AXIS_ALL, 0x4F00);      // Setting for IC-A
Nmc_WriteReg3(0, 1, AXIS_ALL, 0x4F00);      // Setting for IC-B (For the board with multiple ICs)
```

Note:
When the user executes soft reset (set 1 to WR0/D15) in the application, the device driver sets the above setting, the same setting as the above table, to the filter of each input signal.

(2) PC standby mode and hibernation mode
In this device driver, the operation after standby or hibernation mode is not guaranteed.
When the user tries to access the board after standby or hibernation mode, be sure to restart PC before access.

(3) Interrupt support
The user can use the interrupt in the application only developed in VC++ and C#.
And the user cannot use the interrupt in the application developed in VB.

Supported interrupts are as follows:
■ The board equipped with MCX314As
- All interrupts reported by RR3 register

■ The board equipped with MCX304
- All interrupts reported by RR3 register
- The interrupt that occurs when the next segment data and interpolation command is writable in continuous interpolation.
- The interrupt that occurs when the value of stack counter changes from 2 to 1 in bit pattern interpolation.

(4) Interrupt clearing
①The interrupt reported by RR3 register
The interrupt is cleared after the driver read RR3, just after the interrupt occurs in the board.
Then, user-defined function of the application for interrupt is called. (Only when user-defined function has been set.)

②The interrupt that occurs when the next segment data and interpolation command is writable in continuous interpolation.
*The board with MCX314As only
The interpolation interrupt is cleared in the driver, just after the interrupt occurs in the board.
Then, user-defined function of the application for interrupt is called. (Only when user-defined function has been set.)

③The interrupt that occurs when the value of stack counter changes from 2 to 1 in bit pattern interpolation.
*The board with MCX314As only
The interpolation interrupt is cleared in the driver, just after the interrupt occurs in the board.
Then, user-defined function of the application for interrupt is called. (Only when user-defined function has been set.)

(5) When using both RR3 interrupt and the Interpolation Interrupt *The board with MCX314As only
When both the interrupt reported by register RR3 and the interpolation interrupt [*1] are enabled, do as follows. When checking the factor of the interrupt in the interrupt user function[*2], read the factor of RR3 interrupt first and after that check whether the interpolation interrupt occurs or not.

Example) The Interrupt User Function Process when the interrupt occurs
1. Read the factor of RR3 interrupt by using Nmc_ReadEvent. And check if there is the interrupt in RR3 or not.
2. Check if there is the interpolation interrupt or not. (Check the bit CNEXT of RR0 or the bit BPSC1, 0 of RR0)

[*1] The interrupt that occurs when the next segment data and interpolation command is writable in continuous interpolation.
Or the interrupt that occurs when the value of the stack counter has changed into 1 from 2 during the BP interpolation.

[*2] The user function specified by Nmc_SetEvent function.

4.3 Evaluation Tool of MCX304

MCX304 evaluation tool program is the tool to evaluate the board equipped with MCX304 (such as MC8082P, MC8082Pe, MC8042P, MC8022P).
You can download it from our web site: http://www.novaelec.co.jp/eng (attached to MC8000P device driver)
Before executing the evaluation tool, install MC8000P device driver.

Note: This chapter describes the outline of MCX304 evaluation tool. For more details, see ReadMe.txt in Tool\MCX304 Board folder.

4.3.1 Execution Program

There are 2 types of execution program, MCX304-A.exe and MCX304-B.exe.

■ Regarding MCX304
Each evaluation tool evaluates the MCX304 described below.

- MCX304-A.exe
  ① MCX304 of the board equipped with only one MCX304 (such as MC8042P, MC8022P).
  ② The first MCX304 (MCX304-A) of the board equipped with 2 or more MCX304 (such as MC8082P, MC8082Pe).

- MCX304-B.exe
  ①The second MCX304 (MCX304-B) of the board equipped with 2 or more MCX304 (such as MC8082P, MC8080P, MC8082Pe).

■ Regarding program execution
MCX304-A and MCX304-B can be executed to the same board simultaneously. However, the user cannot use or set the interrupt in this case.
When either MCX304-A.exe or MCX304-B.exe is executed to the one board, the user can use the interrupt.
MCX304-A and MCX304-B can be executed to the different board simultaneously. And the user can use the interrupt in this case.

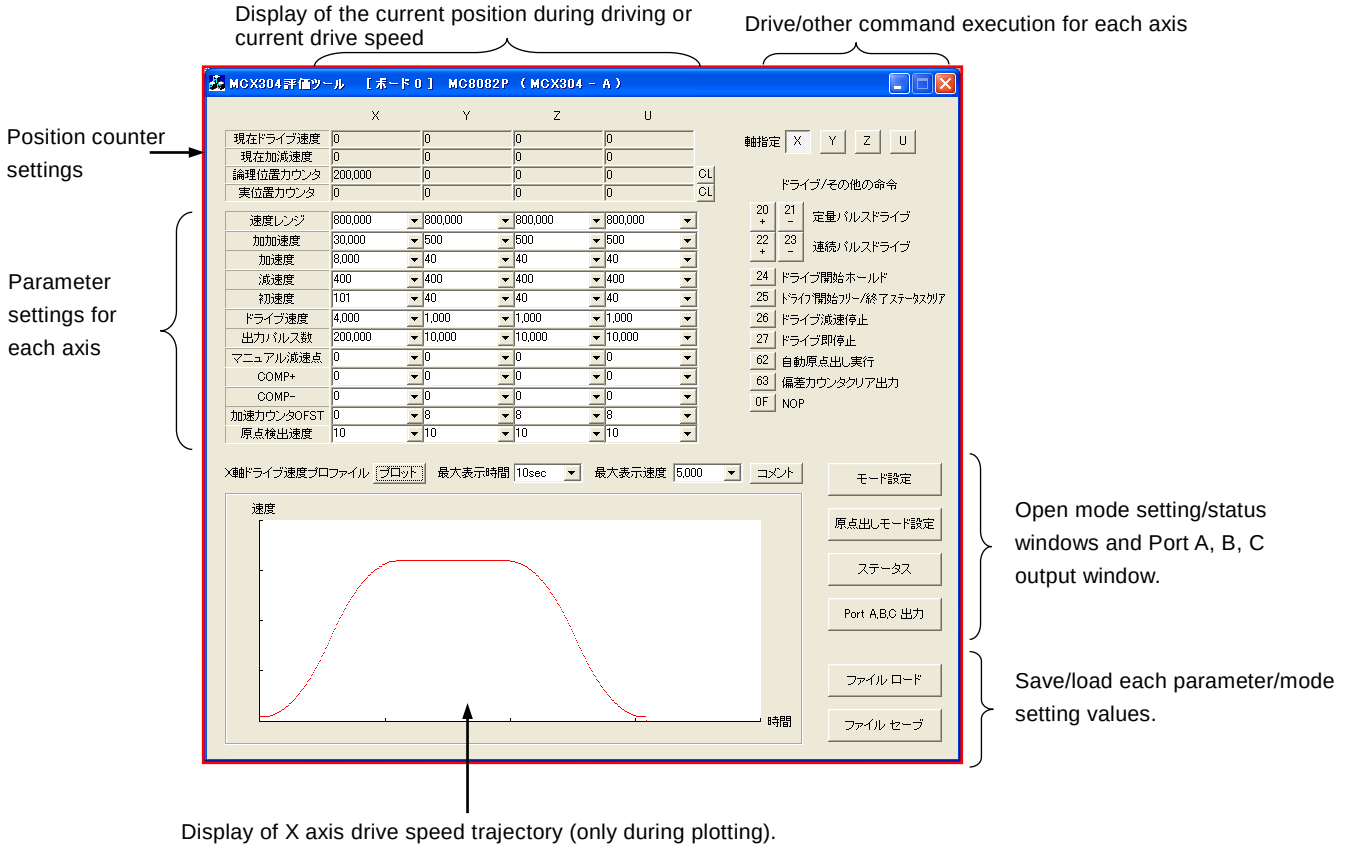
4.3.2 Function Overview

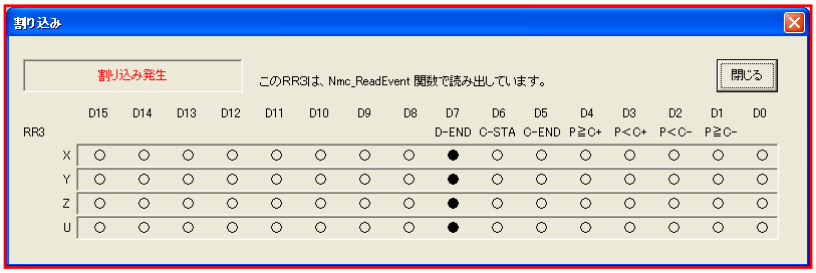
Start the evaluation tool, and the Board Number Select Window appears. The user can select the board number (setting value of rotary switch (0~F) on the board). And if the Board Name Display button is pressed, the board name will be shown at the side of the board number (only the board using this device driver). After selecting the board number, press OK button, and the Main Window appears.

In the Main Window, the user can perform parameter settings for each axis, drive/other command execution, current position/speed display, interrupt window display and save/load parameter/mode setting values. In the Mode Setting Window or Status Window, the user can also set the mode or check the status.
The general output of port A, B and C can be performed in the Port A, B, C Output Window.

4.3.3 Main Window

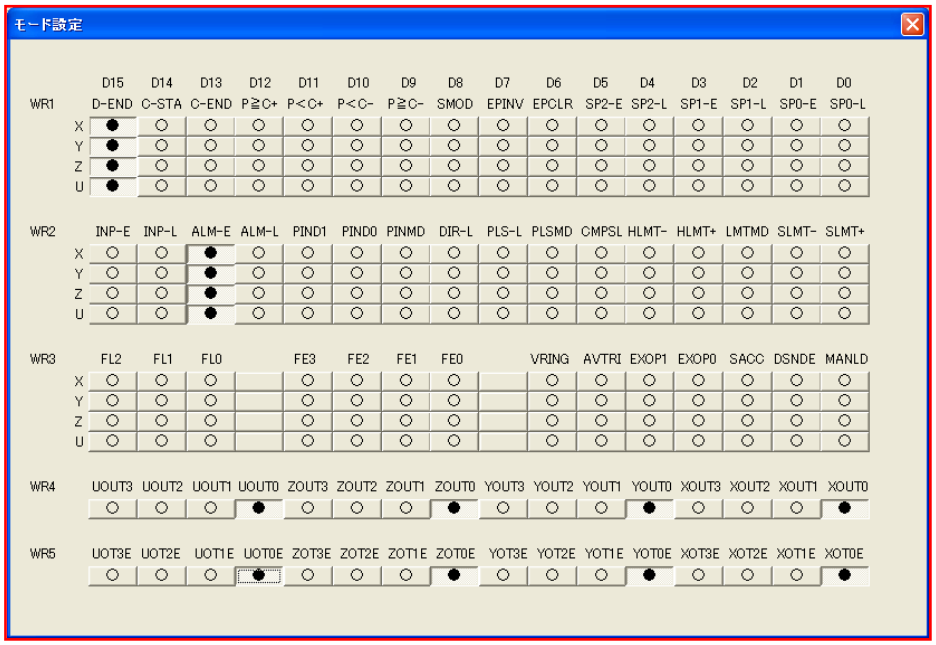
In the main window, the user can do the following operations as shown below.





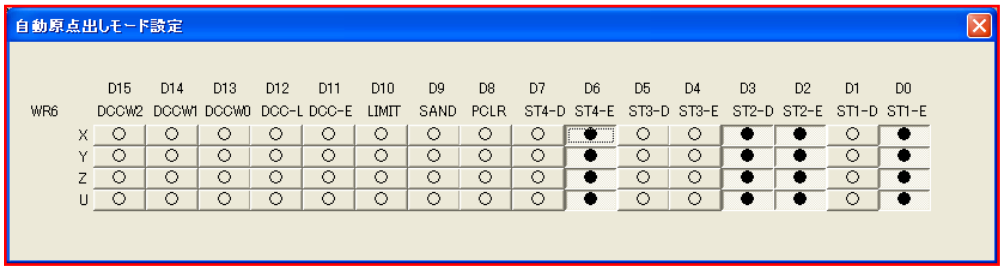
4.3.4 Mode Setting Window

In the mode setting window, the user can set WR1~WR5 of MCX304 (mode register, output register).



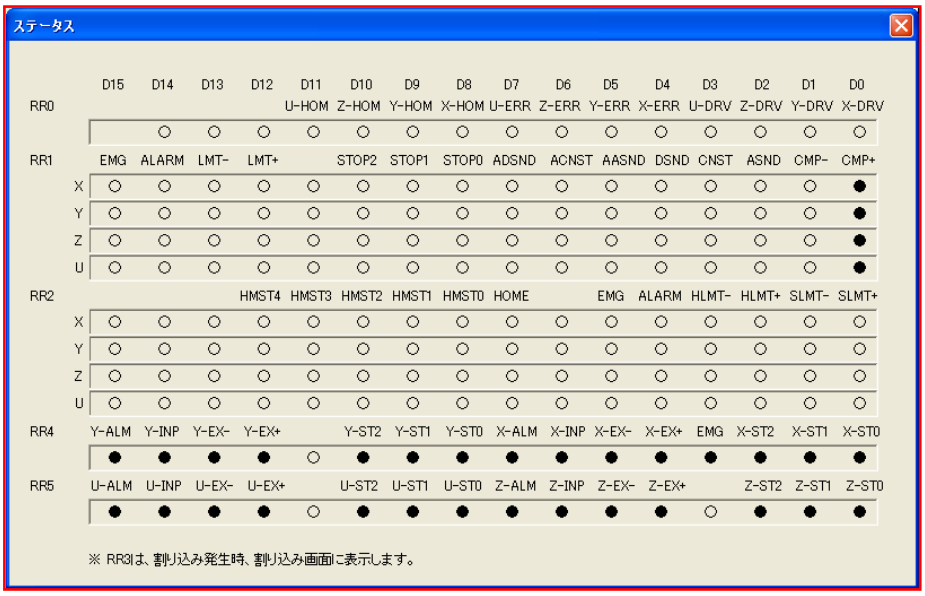
4.3.5 Automatic Home Search Mode Setting Window

In the automatic home search mode setting window, the user can set the automatic home search mode of MCX304.



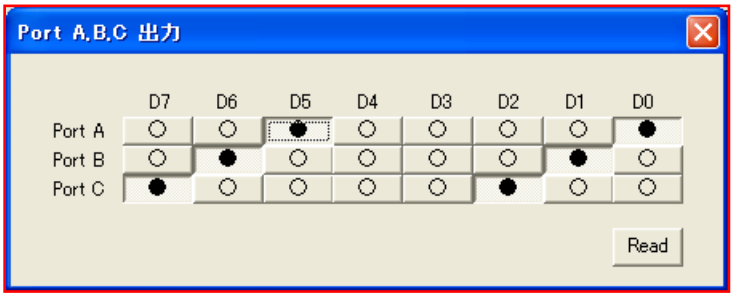
4.3.6 Status Window

The status window shows the current values of RR0, RR1, RR2, RR4 and RR5 (status register, input register) which are read out from MCX304 at regular intervals. This data reading interval is 50 milliseconds.
RR3 is displayed in the interrupt window when an interrupt occurs.



4.3.7 Port A, B, C Output Window

In the port A, B, C output window, the user can set the general output for general output port A, B and C of MC8082P or MC8082Pe. This is only available to MC8082P and MC8082Pe.



4.4 MCX314As Evaluation Tool

MCX314As evaluation tool program is the tool to evaluate the board equipped with MCX314As (such as MC8043P, MC8043Pe). You can download it from our web site: http://www.novaelec.co.jp/eng (attached to MC8000P device driver)
Before executing the evaluation tool, install MC8000P device driver.

Note:
This chapter describes the outline of MCX314As evaluation tool. For more details, see ReadMe.txt in Tool\MCX314As Board folder.

4.4.1 Execution Program

MCX314As-A.exe is execution program.

■ Regarding MCX314As
Each evaluation tool evaluates the MCX314As described below.

- MCX314As-A.exe
  ① MCX314As of the board equipped with only one MCX314As (such as MC8043P, MC8043Pe).

■ Regarding program execution

MCX314As-A can be executed to the different board simultaneously. Before executing, change the different name between the boards. (For instance, such as MCX314As-A.exe and MCX314As-B.exe)

Multiple MCX314As-A cannot be executed to the same board simultaneously.

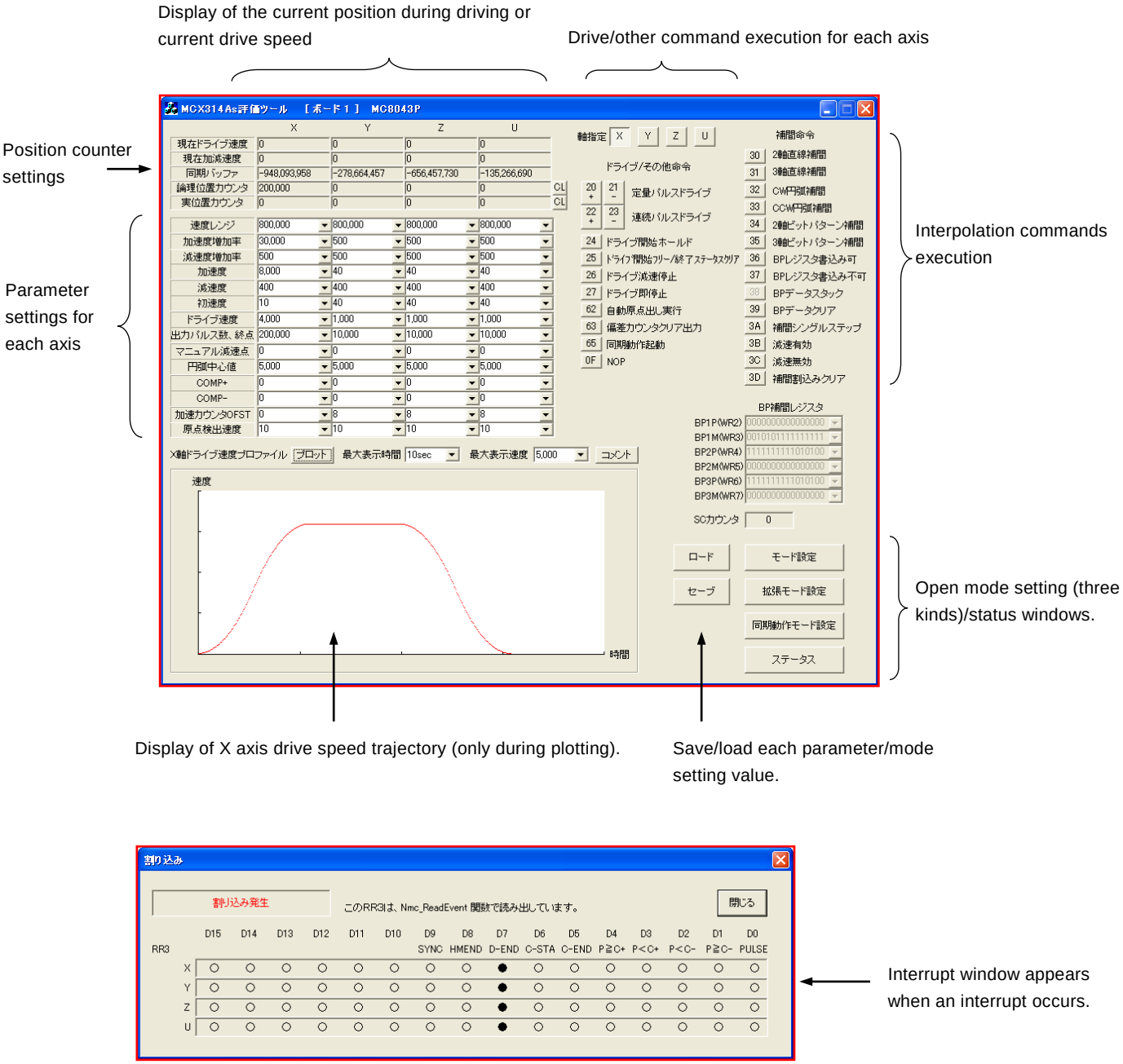
4.4.2 Function Overview

Start the evaluation tool, and the Board Number Select Window appears. The user can select the board number (setting value of rotary switch (0~F) on the board). And if the Board Name Display button is pressed, the board name will be shown at the side of the board number (only the board using this device driver). After selecting the board number, press OK button, and the Main Window appears.

In the Main Window, the user can perform parameter settings for each axis, drive/other command execution, current position/speed display, interrupt window display and save/load parameter/mode setting values. In the 3 kinds of Mode Setting Window or Status Window, the user can also set the mode or check the status.

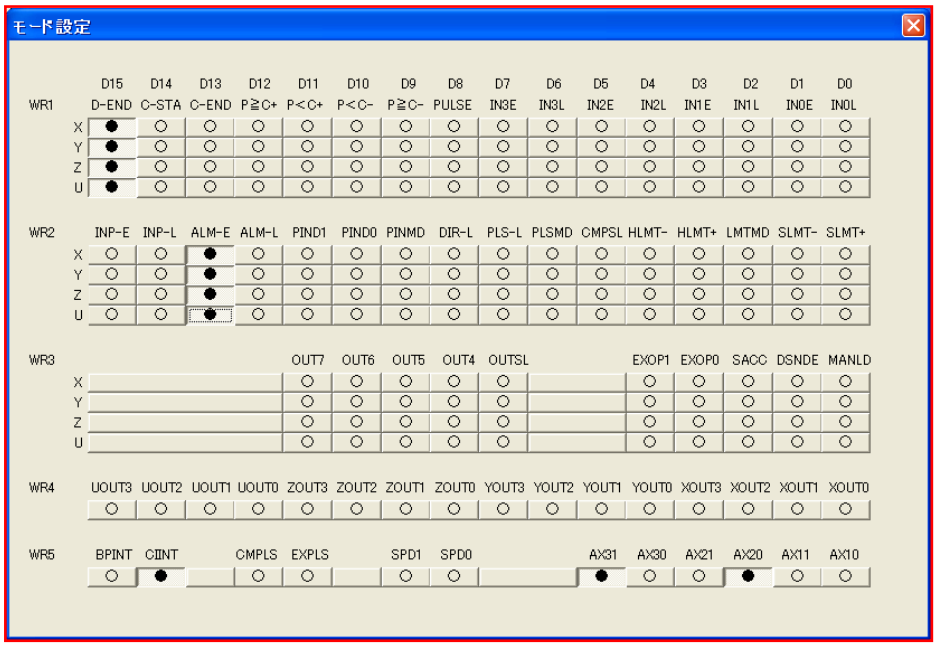
4.4.3 Main Window

In the main window, the user can do the following operations as shown below.



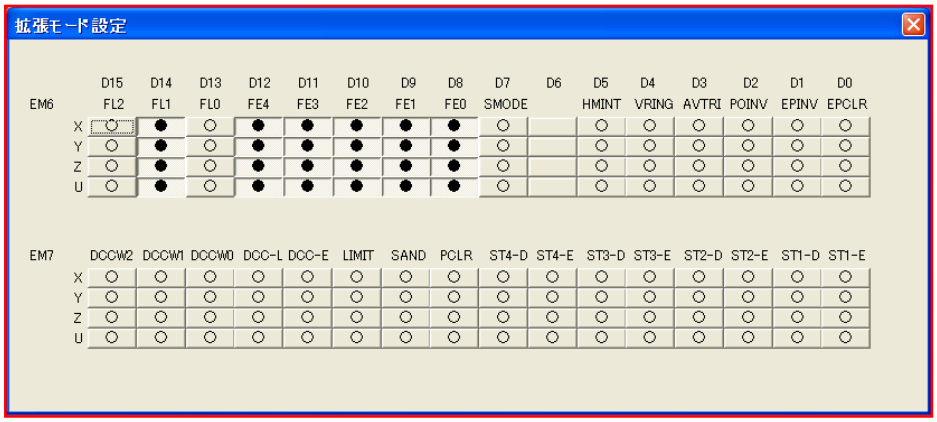
4.4.4 Mode Setting Window

In the mode setting window, the user can set WR1~WR5 of MCX314As (mode register, output register).



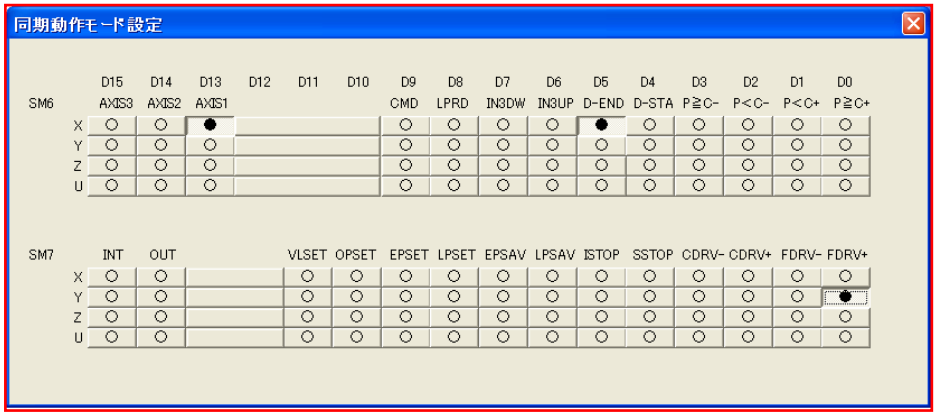
4.4.5 Expansion Mode Setting Window

In the expansion mode setting window, the user can set the extension mode register of MCX314As (EM6, EM7).



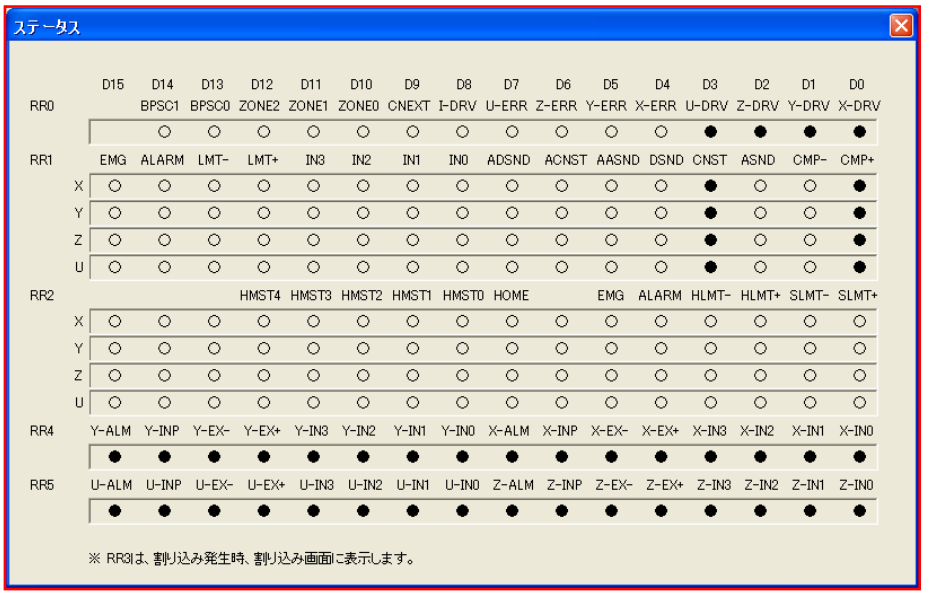
4.4.6 Synchronous Action Mode Setting Window

In the synchronous action mode setting window, the user can set the synchronous action mode register of MCX314As (SM6, SM7).



4.4.7 Status Window

The status window shows the current values of RR0, RR1, RR2, RR4 and RR5 (status register, input register) which are read out from MCX314As at regular intervals. This data reading interval is 50 milliseconds.
RR3 is displayed in the interrupt window when an interrupt occurs.



5. API for MC8500P series

This chapter describes API function which can be used with the following boards

Board	IC	Number of IC	Axis
MC8541P / MC8541Pe	MCX514	1	4
MC8581P / MC8581Pe	MCX514	2	8

5.1 API

API provided by MC8000P.SYS and MC8000P.DLL.

5.1.1 Function List

The following table is the API function list.

The column of VC, VB.NET indicates the availability of each function in each language.

VC++ See VC column.

VB.NET See VB.NET column

C# See C# column.

○ is available and × is not.

(1) Basic Function

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_Open	Start to use the board	○	○	○	106	
Nmc_Close	Stop to use the board	○	○	○	〃	
Nmc_CloseAll	Stop to use all the boards	○	○	○	107	
Nmc_GetBoardInfo	Get information about the opened board	○	○	○	〃	
Nmc_OutPort	Write data to output port	○	○	○	108	
Nmc_InPort	Read data from input port	○	○	○	〃	
Nmc_WriteReg	Write data to register of the board	○	○	○	109	
Nmc_ReadReg	Read data from register of the board	○	○	○	〃	
Nmc_SetEvent	Set user function to handle an interrupt.	○	×	○	110	
Nmc_ResetEvent	Release user function to handle an interrupt.	○	×	○	111	
Nmc_ReadEvent	Read RR1 value of each axis right after the interrupt generated.	○	×	○	〃	

(2) Reset, Command

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_Reset	Reset the IC on the board	○	○	○	112	
Nmc_Command	Execute the command of the specified axis	○	○	○	〃	
Nmc_Command_IP	Execute the interpolation command	○	○	○	113	

(3) Write register

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_WriteReg0	WR0 (Command Register) Writing	○	○	○	113	
Nmc_WriteReg1	WR1 (Mode Register 1) Writing	○	○	○	114	
Nmc_WriteReg2	WR2 (Mode Register 2) Writing	○	○	○	〃	
Nmc_WriteReg3	WR3 (Mode Register 3) Writing	○	○	○	115	
Nmc_WriteReg4	WR4 (Output Register1) Writing	○	○	○	〃	
Nmc_WriteReg5	WR5 (Output Register1) Writing	○	○	○	116	
Nmc_WriteReg6	WR6 (Write Data Register 1) Writing	○	○	○	〃	
Nmc_WriteReg7	WR7 (Write Data Register 2) Writing	○	○	○	〃	

(4) Read Register

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_ReadReg0	RR0(Main Status Register)	○	○	○	117	
Nmc_ReadReg2	RR2(Status Register 2)	○	○	○	〃	
Nmc_ReadReg3	RR3(Status Register 3)	○	○	○	118	
Nmc_ReadReg3P	(Page not specified.)	○	○	○	〃	
Nmc_ReadReg4	RR3(Status Register 3)	○	○	○	119	
Nmc_ReadReg5	(Page specified.)	○	○	○	〃	
Nmc_ReadReg6	RR4(Input Register 1)	○	○	○	〃	
Nmc_ReadReg7	RR5(Input Register 2)	○	○	○	120	

(5) Parameter Settings

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_Jerk	Jerk Setting	○	○	○	121	
Nmc_DJerk	Deceleration increasing rate setting	○	○	○	〃	
Nmc_Acc	Acceleration Setting	○	○	○	122	
Nmc_Dec	Deceleration Setting	○	○	○	123	
Nmc_StartSpd	Initial Speed Setting	○	○	○	〃	
Nmc_Speed	Drive Speed Setting	○	○	○	124	
Nmc_Pulse	Moving Pulse Number/Interpolation Finish Point Setting (For VC, C#)	○	×	○	〃	
Nmc_Pulse_VB	Moving Pulse Number/Interpolation Finish Point Setting (For VB.NET)	×	○	×	125	
Nmc_DecP	Manual Decelerating Point Setting (For VC, C#)	○	×	○	〃	
Nmc_DecP_VB	Manual Decelerating Point Setting (For VB.NET)	×	○	×	126	
Nmc_Center	Circular Center Point Setting	○	○	○	〃	
Nmc_Lp	Logical Position Counter Setting	○	○	○	127	
Nmc_Ep	Real Position Counter Setting	○	○	○	〃	
Nmc_CompP	Software limit + register setting	○	○	○	128	
Nmc_CompM	Software limit - register setting	○	○	○	〃	
Nmc_AccOfst	Acceleration counter offset setting	○	○	○	129	
Nmc_HomeSpd	Home search speed setting	○	○	○	〃	
Nmc_LpMax	Logical Position Counter Maximum Value Setting	○	○	○	130	
Nmc_RpMax	Real Position Counter Maximum Value Setting	○	○	○	〃	
Nmc_MR0	Multi-purpose register 0 setting	○	○	○	131	
Nmc_MR1	Multi-purpose register 1 setting	○	○	○	〃	
Nmc_MR2	Multi-purpose register 2 setting	○	○	○	132	
Nmc_MR3	Multi-purpose register 3 setting	○	○	○	133	
Nmc_SpeedInc	Speed increase/decrease value setting	○	○	○	〃	
Nmc_Timer	Timer value setting	○	○	○	134	
Nmc_TPMax	Interpolation/finish point Maximum Value Setting	○	○	○	135	
Nmc_HLNumber	Helical rotation number setting	○	○	○	〃	
Nmc_HLValue	Helical calculation value setting	○	○	○	〃	

(6) Other Mode Settings

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_MRmMode	Multi-purpose register mode setting	○	○	○	136	
Nmc_PIO1Mode	PIO signal setting1	○	○	○	〃	
Nmc_PIO2Mode	PIO signal setting1･other setting	○	○	○	137	
Nmc_HMSrch1Mode	Automatic Home Search Mode Setting1	○	○	○	138	
Nmc_HMSrch2Mode	Automatic Home Search Mode Setting2	○	○	○	〃	
Nmc_FilterMode	Input signal filter mode setting	○	○	○	139	
Nmc_Sync0Mode	Synchronous ActionSYNC0 Setting	○	○	○	〃	
Nmc_Sync1Mode	Synchronous ActionSYNC1 Setting	○	○	○	140	
Nmc_Sync2Mode	Synchronous ActionSYNC2 Setting	○	○	○	141	
Nmc_Sync3Mode	Synchronous ActionSYNC3 Setting	○	○	○	〃	
Nmc_IPMode	Interpolation Mode Setting	○	○	○	142	

(7) Data Reading

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_ReadLp	Logical Position Counter Reading	○	○	○	143	
Nmc_ReadEp	Real Position Counter Reading	○	○	○	〃	
Nmc_ReadSpeed	Current Drive Speed Reading	○	○	○	144	
Nmc_ReadAccDec	Current Acceleration/Deceleration Reading	○	○	○	145	
Nmc_ReadMR0	Multi-purpose register 0 Reading	○	○	○	〃	
Nmc_ReadMR1	Multi-purpose register 1 Reading	○	○	○	146	
Nmc_ReadMR2	Multi-purpose register 2 Reading	○	○	○	147	
Nmc_ReadMR3	Multi-purpose register 3 Reading	○	○	○	〃	
Nmc_ReadCT	Current timer value Reading	○	○	○	148	
Nmc_ReadTX	Interpolation/finish point Maximum Value reading	○	○	○	149	
Nmc_ReadCHLN	Current Helical rotation number reading	○	○	○	〃	
Nmc_ReadHLV	Helical calculation value reading	○	○	○	150	
Nmc_ReadWR1	WR1 setting value reading	○	○	○	151	
Nmc_ReadWR2	WR2 setting value reading	○	○	○	〃	
Nmc_ReadWR3	WR3 setting value reading	○	○	○	152	
Nmc_ReadMRM	Multi-purpose register setting reading	○	○	○	153	
Nmc_ReadP1M	PIO signal setting1 reading	○	○	○	〃	
Nmc_ReadP2M	PIO signal setting and other setting reading	○	○	○	154	
Nmc_ReadAc	Acceleration setting value reading	○	○	○	155	
Nmc_ReadStartSpd	Initial speed setting value reading	○	○	○	〃	
Nmc_ReadSetSpeed	Drive speed setting value reading	○	○	○	156	
Nmc_ReadPulse	Moving Pulse Number/ Finish Point Setting value reading	○	○	○	157	

(8) Status reading

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_GetDriveStatus	Drive Status Reading	○	○	○	157	
Nmc_GetCNextStatus	The Status Reading of Ready Signal for Writing of Continuous Interpolation	○	○	○	158	
Nmc_GetSc	Continuous interpolation pre-buffer Stack Counter Reading	○	○	○	〃	

(9) Writing / Reading

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_WriteRegSetAxis	Axis Assignment Write Register Writing (WR1~3)	○	○	○	159	
Nmc_ReadRegSetAxis	Axis Assignment Read Register Reading (RR1~2)	○	○	○	〃	
Nmc_WriteData	Data Writing (Parameter)	○	○	○	160	
Nmc_ReadData	Data Reading (4 bytes reading)	○	○	○	〃	

(10) BP Interpolation/ continuous interpolation Execution

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_2BPExecMC8500P	2-axis BP Interpolation Execution	○	○	○	161	
Nmc_3BPExecMC8500P	3-axis BP Interpolation Execution	○	○	○	163	
Nmc_4BPExecMC8500P	4-axis BP Interpolation Execution	○	○	○	166	
Nmc_2BPExecMC8500P_BG	2-axis BP Interpolation Execution (run in the background)	○	○	○	169	
Nmc_3BPExecMC8500P_BG	3-axis BP Interpolation Execution (run in the background)	○	○	○	172	
Nmc_4BPExecMC8500P_BG	4-axis BP Interpolation Execution (run in the background)	○	○	○	175	
Nmc_2CIPExecMC8500P	2-axis Continuous Interpolation Execution	○	○	○	179	
Nmc_3CIPExecMC8500P	3-axis Continuous Interpolation Execution	○	○	○	182	
Nmc_4CIPExecMC8500P	4-axis Continuous Interpolation Execution	○	○	○	185	
Nmc_2CIPExecMC8500P_BG	2-axis Continuous Interpolation Execution (run in the background)	○	○	○	188	
Nmc_3CIPExecMC8500P_BG	3-axis Continuous Interpolation Execution (run in the background)	○	○	○	192	
Nmc_4CIPExecMC8500P_BG	4-axis Continuous Interpolation Execution (run in the background)	○	○	○	196	
Nmc_IPStop	Stop the Interpolation Execution	○	○	○	200	
Nmc_IPGetMsgNo	Get board and IC number from received message at the end of the Interpolation	○	○	○	〃	

(11) Helical interpolation Execution

Function Name	Description	VC	VB.NET	C#	Page	Note
Nmc_HLValueExe	Helical calculation Execution	○	○	○	202	
Nmc_HLExec	Helical interpolation Execution	○	○	○	205	

5.1.2 Function Specifications

For VC++ : See **VC** and [VC]
For VB.NET : See **VB.NET** and [VB.NET]
For C# : See **C#** and [C#]

And others are common to each language.

Function Name	Function and Content

Nmc_Open	Start the board.
	
	VC BOOL Nmc_Open(int No, BOOL IntrptFlg);
	VB.NET Function Nmc_Open(ByVal No As Integer, ByVal IntrptFlg As Integer) As Integer
	C# bool MC8000P.Nmc_Open(int No, bool IntrptFlg);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IntrptFlg Set the flag to use interrupt.
	[VC] TRUE: use interrupt. FALSE: not use.
	[VB.NET] FALSE only. Interrupt cannot be used in VB.
	[C#] true: use interrupt. false: not use.
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[VB.NET] If the function succeeds, the return value is nonzero.
	If the function fails, the return value is 0.
	[C#] If the function succeeds, the return value is true.
	If the function fails, the return value is false.
	
	Example
	[VC] status = Nmc_Open(0, FALSE); // Open the board 0 and not use the interrupt.
	[VB.NET] status = Nmc_Open(0, false)
	[C#] status = MC8000P.Nmc_Open(0, false):

Nmc_Close	Terminate the board.
	
	VC BOOL Nmc_Close(int No);
	VB.NET Function Nmc_Close(ByVal No As Integer) As Integer
	C# bool MC8000P.Nmc_Close(int No);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[VB.NET] If the function succeeds, the return value is nonzero.
	If the function fails, the return value is 0.
	[C#] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	
	Example
	[VC] status = Nmc_Close(0); // Close the board 0.
	[VB.NET] status = Nmc_Close(0)
	[C#] status = MC8000P.Nmc_Close(0);

Function Name	Function and Content

Nmc_CloseAll	Terminate all the boards.
	
	VC BOOL Nmc_CloseAll(void);
	VB.NET Function Nmc_CloseAll() As Integer
	C# bool MC8000P.Nmc_CloseAll();
	
	Input Parameter
	None
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[VB.NET] If the function succeeds, the return value is nonzero.
	If the function fails, the return value is 0.
	[C#] If the function succeeds, the return value is true.
	If the function fails, the return value is false.
	
	Example
	[VC] status = Nmc_CloseAll(); // Close all the boards.
	[VB.NET] status = Nmc_CloseAll()
	[C#] status = MC8000P.Nmc_CloseAll(); [

Nmc_GetBoardInfo	Get Device ID of the opened board.
	
	VC BOOL Nmc_GetBoardInfo(int No, USHORT* pDeviceID);
	VB.NET Function Nmc_GetBoardInfo(ByVal No As Integer, ByRef DeviceID As Short) As Integer
	C# bool MC8000P.Nmc_GetBoardInfo(int No, out ushort DeviceID);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	DeviceID [VC] Address of a variable to store the Device ID obtained from the board.
	[VB.NET][C#] Address of a variable to store the Device ID obtained from the board.
	See [4.1.3] (1)③ regarding Device ID of each board.
	[VC][VB] MC8541P/MC8541Pe are ID_MC8541P, MC8581P/MC8581Pe are ID_MC8581P.
	[C#] MC8541P/MC8541Pe are Dev_ID.MC8541P, MC8581P/MC8581Pe are Dev_ID.MC8581P.
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[VB.NET] If the function succeeds, the return value is nonzero.
	If the function fails, the return value is 0.
	[C#] If the function succeeds, the return value is true.
	If the function fails, the return value is false.
	
	Example
	[VC] USHORT DeviceID;
	status = Nmc_GetBoardInfo(No, &DeviceID); // Get Device ID.
	if(DeviceID == ID_MC8581P) // When the board is MC8581P.
	[VB.NET] Dim DeviceID As Short
	status = Nmc_GetBoardInfo(No, DeviceID)
	If DeviceID = ID_MC8581P Then
	[C#] ushort DeviceID;
	status = Nmc_GetBoardInfo(No, out DeviceID);
	if(DeviceID == Dev_ID.MC8581P)

Function Name	Function and Content

Nmc_OutPort	Write 2-byte data into output port.
	
	VC void Nmc_OutPort(int No, long Adr, long Dat);
	VB.NET Sub Nmc_OutPort(ByVal No As Integer, ByVal adr As Integer, ByVal Dat As Integer)
	C# void MC8000P.Nmc_OutPort(int No, REG_MCX Adr, int Dat);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	Adr Address to write. I/O address described in User's Manual of each board. See [5.1.3] (1)①, (6) for more details.
	Data Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_OutPort(No, 0, MC8500P_CMD_RST); // Soft reset IC-A.(Write into WR0)
	[VB.NET] Call Nmc_OutPort(No, 0, MC8500P_CMD_RST)
	[C#] MC8000P.Nmc_OutPort(No, REG_MCX.WR0_A, MC8500P_CMD_RST);

Nmc_InPort	Read out 2-byte data from input port.
	
	VC long Nmc_InPort(int No, long Adr);
	VB.NET Function Nmc_InPort(ByVal No As Integer, ByVal adr As Integer) As Integer
	C# int MC8000P.Nmc_InPort(int No, REG_MCX Adr);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	Adr Address to read. I/O address described in User's Manual of each board. See [5.1.3] (1)①, (6) for more details.
	
	Return Value
	Data read out from input port.
	
	Example
	[VC] data = Nmc_InPort(No, 0); // Read out the read register RR0 of IC-A.
	[VB.NET] data = Nmc_InPort(No, 0)
	[C#] data = MC8000P.Nmc_InPort(No, REG_MCX.RR0_A);
	
	Note
	[VC][C#] Regarding reading the RR1 register data, refer to Nmc_ReadEvent function.

Function Name	Function and Content
Nmc_WriteReg	Write data into write register (WR0~WR7) of the board. **VC** void Nmc_WriteReg(int No, int IcNo, long RegNum, long Dat); **VB.NET** Sub Nmc_WriteReg(ByVal No As Integer, ByVal IcNo As Integer, ByVal RegNum As Integer, ByVal Dat As Integer) **C#** void MC8000P.Nmc_WriteReg(int No, int IcNo, int RegNum, int Dat); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. RegNum Register to be written (MCX_WR0~MCX_WR7). See [5.1.3] (1) for more details. e.g. [VC][VB.NET] Specify MCX_WR0 for WR0 and MCX_WR1 for WR1. [C#] Specify REG_MCX.WR0 for WR0 and REG_MCX.WR1 for WR1. Dat Data to be written. **Return Value** None **Example** [VC] Nmc_WriteReg(No, 0, MCX_WR0, MC8500P_CMD_RST); // Soft reset IC-A.(Write into WR0) [VB.NET] Call Nmc_WriteReg(No, 0, MCX_WR0, MC8500P_CMD_RST) [C#] MC8000P.Nmc_WriteReg(No, 0, REG_MCX.WR0, MC8500P_CMD_RST);

Nmc_ReadReg	Read out data from read register (RR0~RR7) of the board.
	
	VC long Nmc_ReadReg(int No, int IcNo, long RegNum);
	VB.NET Function Nmc_ReadReg(ByVal No As Integer, ByVal IcNo As Integer, ByVal RegNum As Intege
	) As Intege
	C# void MC8000P.Nmc_ReadReg(int No, int IcNo, int RegNum);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	RegNum Register to read (MCX_RR0~MCX_RR7). See Footnote (1) for more details.
	e.g. [VC], [VB.NET] Specify MCX_RR0 for RR0 and MCX_RR1 for RR1.
	[C#] Specify REG_MCX.RR0 for RR0 and REG_MCX.RR1 for RR1.
	Return Value
	Data read out from read register.
	
	Example
	[VC] data = Nmc_ReadReg(No, 0, MCX_RR0); // Read out the read register RR0 of IC-A.
	[VB.NET] data = Nmc_ReadReg(No, 0, MCX_RR0)
	[C#] data = MC8000P.Nmc_ReadReg(No, 0, REG_MCX.WR0);
	
	Note
	[VC][C#] Regarding reading the RR1 register data, refer to Nmc_ReadEvent function.

Function Name	Function and Content

Nmc_SetEvent	Set user function to handle an interrupt.
	By executing this function, the user function is called when an interrupt occurs and then one specified argument is passed. (C# cannot specify an argument.) This user function is run as one thread.
	To release user function to handle an interrupt, execute Nmc_ResetEvent.
	
	VC BOOL Nmc_SetEvent(int No, LPTHREAD_START_ROUTINE UserFunc, LPVOID lpParameter);
	VB.NET Cannot be used.
	C# bool MC8000P.Nmc_SetEvent(int No, UserThread Callback, int param);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	[VC]UserFunc User function address.
	[VC]lpParameter Assign one argument to be pass to user function thread. Set the available pointer for the thread. When not using the argument, set NULL. When using the pointer, set the available pointer when the user function is called.
	[C#] Callback User method (delegate type). See 4.1.4 Usage for more details.
	[C#] param Assign one parameter (int only, such as numeric type) to pass to user function after an interrupt occurs.
	
	Return Value
	If the function succeeds, the return value is TRUE. If the function fails, the return value is FALSE.
	Example
	[VC]
	(When the board number is 0)
	status = Nmc_SetEvent(0, MC_EventFunc0, lpParam); // Set the function address and argument
	Nmc_WriteReg1(0, 0, AXIS_ALL, 0x0080); // Interrupt occurs at the stop of IC-A (All axes)
	
	(When the board number is 1)
	status = Nmc_SetEvent(1, MC_EventFunc1, NULL); // Set the function address and argument
	Nmc_WriteReg1(1, 0, AXIS_ALL, 0x0080); // Interrupt occurs at the stop of IC-A (All axes)
	
	■Example of interrupt user function
	DWORD WINAPI MC_EventFunc0(LPVOID lpParam)
	{
	long Rr1X, Rr1Y, Rr1Z, Rr1U;
	Nmc_ReadEvent(0, 0, &Rr1X,&Rr1Y,&Rr1Z,&Rr1U); // Board 0, read out RR1 interrupt dat of IC-A
	. . . .
	return 0;
	}
	
	DWORD WINAPI MC_EventFunc1(LPVOID lpParam)
	{
	long Rr1X, Rr1Y, Rr1Z, Rr1U;
	Nmc_ReadEvent(1, 0, &Rr1X,&Rr1Y,&Rr1Z,&Rr1U); // Board 1, read RR3 interrupt data of IC-A
	. . . .
	return 0;
	}
	[C#]
	// Assign interrupt method is to a delegate type variable
	MC8000P.callback[0] = new MC8000P.UserThread(isr);
	//Set user method to handle an interrupt
	MC8000P.Nmc_SetEvent(no, MC8000P.callback[0], param);

Nmc_ResetEvent	Release user function to handle an interrupt. By executing this function, the user function is not called when an interrupt occurs. **VC** BOOL Nmc_ResetEvent(int No); **VB.NET** cannot be used. **C#** bool MC8000P.Nmc_ResetEvent(int No); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) **Return Value** [VC] If the function succeeds, the return value is TRUE. If the function fails, the return value is FALSE. [C#] If the function succeeds, the return value is true. If the function fails, the return value is false. **Example** [VC] status = Nmc_ResetEvent(No); [C#] status = MC8000P.Nmc_ResetEvent(No);

Nmc_ReadEvent	Read the value of RR1 of each axis right after an interrupt generation. (RR1 will be cleared after reading.)
	
	VC BOOL Nmc_ReadEvent(int No, int IcNo, long* RrIrX, long* RrIrY, long* RrIrZ, long* RrIrU);
	VB.NET cannot be used.
	C# bool MC8000P.Nmc_ReadEvent(int No, int IcNo, out int RrIrX, out int RrIrY, out int RrIrZ, out int RrIrU);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See 5.1.3 (7) for more details.
	RrIrX Pointer to a variable that receives the X axis RR1 value.
	RrIrY Pointer to a variable that receives the Y axis RR1 value.
	RrIrZ Pointer to a variable that receives the Z axis RR1 value.
	RrIrU Pointer to a variable that receives the U axis RR1 value.
	
	Return Value
	[VC] If the function succeeds, the return value is TRUE. If the function fails, the return value is FALSE.
	[C#] If the function succeeds, the return value is true. If the function fails, the return value is false.
	
	Example
	[VC] long Rr1X[2], Rr1Y[2], Rr1Z[2], Rr1U[2];
	Nmc_ReadEvent(No, 0, &Rr1X[0], &Rr1Y[0], &Rr1Z[0], &Rr1U[0]); // Read out RR1 data of IC-A
	Nmc_ReadEvent(No, 1, &Rr1X[1], &Rr1Y[1], &Rr1Z[1], &Rr1U[1]); // Read out RR1 data of IC-B
	[C#] int Rr1X,Rr1Y,Rr1Z,Rr1U ; // X-axis, Y-axis, Z-axis, U-Axis
	MC8000P.Nmc_ReadEvent(No, IcNo, out Rr1X, out Rr1Y, out Rr1Z, out Rr1U);
	
	Note
	The RR1 value is cleared due to the driver operation, just after the interrupt occurs. The user must use this function in order to know the interrupt factor.
	In addition, when the interrupt occurs, the driver certainly reads and saves RR1 data regardless of the execution of Nmc_SetEvent or Nmc_ResetEvent functions. RR1 data saved in the driver can be cleared after reading by execution of Nmc_ReadEvent function.
	To clear the RR1 data of the driver, execute Nmc_ReadEvent function.

Function Name	Function and Content

Nmc_Reset	Reset the IC on the board.
	
	VC void Nmc_Reset(int No, int IcNo);
	VB.NET Sub Nmc_Reset(ByVal No As Integer, ByVal IcNo As Integer)
	C# void MC8000P.Nmc_Reset(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Reset(1, 0); // Reset IC-A of the board 1.
	[VB.NET] Call Nmc_Reset(1, 0)
	[C#] MC8000P.Nmc_Reset(1, 0);

Nmc_Command	Execute the command of a specified axis. (Write the command of a specified axis into WR0.) **VC** void Nmc_Command(int No, int IcNo, int axis, int cmd); **VB.NET** Sub Nmc_Command(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal cmd As Integer) **C#** void MC8000P.Nmc_Command(int No, int IcNo, AXIS axis, CMD cmd); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. axis Axis to execute the command. Multiple axes can be assigned. See [5.1.3] (2) for more details. cmd Command number. Specify one command from “Driving commands, Other commands of “Command Definition” described in definition file (*1). For relative position drive, specify MC8500P_CMD_DRVRL. See [5.1.3] (4) for C#. *1 : [VC]MC8000P_DLL.H / [VB.REMAINING]MC8000P_DLL.vb **Return Value** None **Example** [VC] Nmc_Command(No, IcNo, AXIS_X, MC8500P_CMD_DRVRL); // Execute the relative position drive of X axis. [VB.NET] Call Nmc_Command(No, IcNo, AXIS_X, MC8500P_CMD_DRVRL) [C#] MC8000P.Nmc_Command(No, IcNo, AXIS.X, CMD.MC8500P_CMD_DRVRL);

Function Name	Function and Content

Nmc_Command_IP	Execute interpolation command. (Write interpolation command into WR0.)
	
	VC void Nmc_Command_IP(int No, int IcNo, int cmd);
	VB.NET Sub Nmc_Command_IP(ByVal No As Integer, ByVal IcNo As Integer, ByVal cmd As Integer)
	C# void MC8000P.Nmc_Command_IP(int No, int IcNo, CMD cmd);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See Footnote [5.1.3] (7) for more details.
	cmd Command number. Specify one command from “Interpolation Commands” of “Command Definition” described in definition file (*1).
	
	For 2-axis linear interpolation drive, specify MC8500P_CMD_2CIP. See [5.1.3] (1) ④ for [C#]
	*1 : [VC]MC8000P_DLL.H / [VB.NET]MC8000P_DLL.vb
	
	Return Value
	None
	
	Example
	[VC] Nmc_IPMode(No, IcNo, 0x0003); // Set the interpolation axis (Main axis X, Second axis: Y).
	Nmc_Command_IP(No, IcNo, MC8500P_CMD_2CIP); // Execute 2-axis linear interpolation drive.
	[VB.NET] Call Nmc_IPMode(No, IcNo, &H3)
	Call Nmc_Command_IP(No, IcNo, MC8500P_CMD_2CIP)
	[C#] MC8000P.Nmc_IPMode(No, IcNo, 0x0003);
	MC8000P.Nmc_Command_IP(No, IcNo, CMD.MC8500P_CMD_2CIP);

Nmc_WriteReg0	Write data into WR0 (Command register).
	
	VC void Nmc_WriteReg0(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_WriteReg0(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg0(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg0(No, IcNo, 0x0150); // Execute the relative position drive of X axis.
	[VB.NET] Call Nmc_WriteReg0(No, IcNo, &H150)
	[C#] MC8000P.Nmc_WriteReg0(No, IcNo, 0x0150);

Function Name	Function and Content

Nmc_WriteReg1	Write data into WR1 (Mode register 1).
	
	VC void Nmc_WriteReg1(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_WriteReg1(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg1(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	axis Axis to write data. Axis to write data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg1(No, IcNo, AXIS_X, 0x0080); // Interrupt occurs at the stop (X axis).
	[VB.NET] Call Nmc_WriteReg1(No, IcNo, AXIS_X, &H80)
	[C#] MC8000P.Nmc_WriteReg1(No, IcNo, AXIS.X, 0x0080);

Nmc_WriteReg2	Write data into WR2 (Mode register 2).
	
	VC void Nmc_WriteReg2(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_WriteReg2(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg2(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	axis Axis to write data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg2(No, IcNo, AXIS_Y, 0x0200); // Enable ALARM (Y axis)
	[VB.NET] Call Nmc_WriteReg2(No, IcNo, AXIS_Y, &H200)
	[C#] MC8000P.Nmc_WriteReg2(No, IcNo, AXIS.Y, 0x0200);

Function Name	Function and Content

Nmc_WriteReg3	Write data into WR3 (Mode register 3).
	
	VC void Nmc_WriteReg3(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_WriteReg3(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg3(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	axis Axis to write data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg3(No, IcNo, AXIS_ALL, 0x0004); // Set S-curve acceleration/deceleration to the all axes.
	[VB.NET] Call Nmc_WriteReg3(No, IcNo, AXIS_ALL, &H4)
	[C#] MC8000P.Nmc_WriteReg3(No, IcNo, AXIS.ALL, 0x0004);

Nmc_WriteReg4	Write data into WR4 (Output register).
	
	VC void Nmc_WriteReg4(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_WriteReg4(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg4(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg4(No, IcNo, 0x0001); // Hi level output for X axis general output PIO0.
	[VB.NET] Call Nmc_WriteReg4(No, IcNo, &H1)
	[C#] MC8000P.Nmc_WriteReg4(No, IcNo, 0x0001);

Function Name	Function and Content

Nmc_WriteReg5	Write data into WR5 (Output register2).
	
	VC void Nmc_WriteReg5(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_WriteReg5(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg5(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1).Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	wdata Data to be written.
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg5(No, IcNo, 0x0001); // Hi level output for Z axis general output PIO0.
	[VB.NET] Call Nmc_WriteReg5(No, IcNo, &H1)
	[C#] MC8000P.Nmc_WriteReg5(No, IcNo, 0x0024);

Nmc_WriteReg6	Write data into WR6 (Write data register 1).
	
	VC void Nmc_WriteReg6(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_WriteReg6(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg6(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details
	wdata Data to be written.
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg6(No, IcNo, 0x1234); // Write data (1234)H into write data register 1.
	[VB.NET] Call Nmc_WriteReg6(No, IcNo, &H1234)
	[C#] MC8000P.Nmc_WriteReg6(No, IcNo, 0x1234);

Nmc_WriteReg7	Write data into WR7 (Write data register 2).
	
	VC void Nmc_WriteReg7(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_WriteReg7(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_WriteReg7(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	wdata Data to be written
	
	Return Value
	None
	
	Example
	[VC] Nmc_WriteReg7(No, IcNo, 0x5678); // Write data (5678) into write register2.
	[VB.NET] Call Nmc_WriteReg7(No, IcNo, &H5678)
	[C#] MC8000P.Nmc_WriteReg7(No, IcNo, 0x5678);

Function Name	Function and Content

Nmc_ReadReg0	Read out data from RR0 (Main status register).
	
	VC long Nmc_ReadReg0(int No, int IcNo);
	VB.NET Function Nmc_ReadReg0(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg0(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	wdata Data to be written
	
	Return Value
	The data of RR0 (Main status register)
	
	Example
	[VC] Data = Nmc_ReadReg0(No, IcNo); // Read out RR0.
	[VB.NET] Data = Nmc_ReadReg0(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadReg0(No, IcNo);

Nmc_ReadReg2	Read out data from RR2 (Status register 2).
	
	VC long Nmc_ReadReg2(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadReg2(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg2(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	See [5.1.3] (2) for more details.
	
	Return Value
	The data of RR2 (Status register 2)
	
	Example
	[VC] Data = Nmc_ReadReg2(No, IcNo, AXIS_Y); // Read out RR2 Y axis.
	[VB.NET] Data = Nmc_ReadReg2(No, IcNo, AXIS_Y)
	[C#] Data = MC8000P.Nmc_ReadReg2(No, IcNo, AXIS.Y);

Function Name	Function and Content

Nmc_ReadReg3	Read out data from RR3 (Status register 3).
	
	VC long Nmc_ReadReg3(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadReg3(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000.Nmc_ReadReg3(int No, int IcNo, AXIS axis);
	
	Input parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis, See [5.1.3] (2) for more details.
	
	Return Value
	The data of RR3 (Status register 3)
	
	Example
	[VC] Nmc_Command(No, IcNo, AXIS_Y, MC8500P_CMD_RR3P0); // Indicate page0 of RR3 o Y axis.
	Data = Nmc_ReadReg3(No, IcNo, AXIS_Y); // Read out RR3 of Y axis.
	[VB.NET] Call Nmc_Command(No, IcNo, AXIS_Y, MC8500P_CMD_RR3P0)
	Data = Nmc_ReadReg3(No, IcNo, AXIS_Y)
	[C#] MC8000P.Nmc_Command(No, IcNo, AXIS.Y, CMD.MC8500P_CMD_RR3P0);
	Data = MC8000P.Nmc_ReadReg3(No, IcNo, AXIS.Y);
	
	Note
	Regarding RR3 register, there are 2 kinds: Page 0 and Page1. The page can be specified by writing RR3 page display command (7Ah, 7Bh). It will be Page 0 at reset.

Nmc_ReadReg3P	Read out data from RR3 (Status register 3. Specify page.).
	
	VC long Nmc_ReadReg3P(int No, int IcNo, int axis, int Page);
	VB.NET Function Nmc_ReadReg3P(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal Page As Integer) As Integer
	C# int MC8000.Nmc_ReadReg3P(int No, int IcNo, AXIS axis, int Page);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	See [5.1.3] (2) for more details.
	Page Page to read data. Pge0 is 0, page1 is 1.
	
	Return Value
	RR3 (Status register3) data of the specified page.
	
	Example
	[VC] Data = Nmc_ReadReg3P(No, IcNo, AXIS_Y, 0); // Read out page0 of RR3 of Y axis.
	[VB.NET] Data = Nmc_ReadReg3P(No, IcNo, AXIS_Y, 0)
	[C#] Data = MC8000P.Nmc_ReadReg3P(No, IcNo, AXIS.Y, 0);

Function name	Function and Content

Nmc_ReadReg4	Read out data from RR4 (PIO read register 1).
	
	VC long Nmc_ReadReg4(int No, int IcNo);
	VB.NET Function Nmc_ReadReg4(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg4(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return Value
	The data of RR4 (PIO read register 1).
	
	Example
	[VC] Data = Nmc_ReadReg4(No, IcNo); // Read out RR4.
	[VB.NET] Data = Nmc_ReadReg4(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadReg4(No, IcNo);

Nmc_ReadReg5	Read out data from RR5 (PIO read register 2).
	
	VC long Nmc_ReadReg5(int No, int IcNo);
	VB.NET Function Nmc_ReadReg5(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg5(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return Value
	The data of RR5 (PIO read register 2).
	
	Example
	[VC] Data = Nmc_ReadReg5(No, IcNo); // Read out RR5.
	[VB.NET] Data = Nmc_ReadReg5(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadReg5(No, IcNo);

Nmc_ReadReg6	Read out data from RR6 (Read data register 1).
	
	VC long Nmc_ReadReg6(int No, int IcNo);
	VB.NET Function Nmc_ReadReg6(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg6(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return Value
	The data of RR6 (Read data register 1)
	
	Example
	[VC] Data = Nmc_ReadReg6(No, IcNo); // Read out RR6.
	[VB.NET] Data = Nmc_ReadReg6(No, IcNo) ' Read out RR6.
	[C#] Data = MC8000P.Nmc_ReadReg6(No, int IcNo);

Function Name	Function and Content

Nmc_ReadReg7	Read out data from RR7 (Read data register 2).
	
	VC long Nmc_ReadReg7(int No, int IcNo);
	VB.NET Function Nmc_ReadReg7(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadReg7(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return Value
	The data of RR7 (Read data register 2)
	
	Example
	[VC] Data = Nmc_ReadReg7(No, IcNo); // Read out RR7.
	[VB.NET] Data = Nmc_ReadReg7(No, IcNo) ' Read out RR7.
	[C#] Data = MC8000P.Nmc_ReadReg7(No, IcNo);

Function Name	Function and Content

Nmc_Jerk	Set jerk .
	
	VC void Nmc_Jerk(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Jerk(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Jerk(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Jerk(No, IcNo, AXIS_X, 1000); // Set 1000 to jerk (X axis).
	[VB.NET] Call Nmc_Jerk(No, IcNo, AXIS_X, 1000)
	[C#] MC8000P.Nmc_Jerk(No, IcNo, AXIS.X, 1000);

Function Name	Function and Content

Nmc_DJerk	Set deceleration increasing rate.
	
	VC void Nmc_DJerk(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_DJerk(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_DJerk(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_DJerk(No, IcNo, AXIS_X, 1000); // Set 1000 to deceleration increasing rate (Z axis).
	[VB.NET] Call Nmc_DJerk(No, IcNo, AXIS_X, 1000)
	[C#] MC8000P.Nmc_DJerk(No, IcNo, AXIS.X, 1000);

Nmc_Acc	Set acceleration.
	
	VC void Nmc_Acc(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Acc(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Acc(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Acc(No, IcNo, AXIS_Y, 100); // Set 100 to acceleration (Y axis).
	[VB.NET] Call Nmc_Acc(No, IcNo, AXIS_Y, 100)
	[C#] MC8000P.Nmc_Acc(No, IcNo, AXIS.Y, 100);

Function Name	Function and Content

Nmc_Dec	Set deceleration.
	
	VC void Nmc_Dec(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Dec(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Dec(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Dec(No, IcNo, AXIS_Z, 100); // Set 100 to deceleration (Z axis).
	[VB.NET] Call Nmc_Dec(No, IcNo, AXIS_Z, 100)
	[C#] MC8000P.Nmc_Dec(No, IcNo, AXIS.Z, 100);

Nmc_Speed	Set drive speed.
	
	VC void Nmc_Speed(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Speed(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Speed(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Speed(No, IcNo, AXIS_X | AXIS_Y, 1000); // Set 1000 to drive speed (X/Y axes).
	[VB.NET] Call Nmc_Speed(No, IcNo, AXIS_X or AXIS_Y, 1000)
	[C#] MC8000P.Nmc_Speed(No, IcNo, AXIS.X | AXIS.Y, 1000);

Function Name	Function and Content

Nmc_Pulse	Set output pulse number or interpolation finish point. (VC/C# only)
	
	The number of output pulses indicates the total number of pulses that are output in fixed pulse driving.
	For linear and circular interpolation driving, set the finish point of each axis.
	For helical interpolation, set the feed amount of Z and U axis.
	The finish point should be specified relative value to the current position.
	
	VC void Nmc_Pulse(int No, int IcNo, int axis, long wdata);
	VB.NET cannot be used.
	C# void MC8000P.Nmc_Pulse(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Pulse(No, IcNo, AXIS_X, 2000); // Set 2000 to output pulse number (X axis).
	Nmc_Pulse(No, IcNo, AXIS_Y, 300); // Set 300 to interpolation finish point (Y axis)
	Nmc_Pulse(No, IcNo, AXIS_Z, -400); // Set -400 to interpolation finish point (Z axis)
	[C#] MC8000P.Nmc_Pulse(No, IcNo, AXIS.X, 2000);
	MC8000P.Nmc_Pulse(No, IcNo, AXIS.Y, 300);
	MC8000P.Nmc_Pulse(No, IcNo, AXIS.Z, -400);

Nmc_Pulse_VB	Set output pulse number or interpolation finish point. (VB/VB.NET only). The number of output pulses indicates the total number of pulses that are output in fixed pulse driving. For linear and circular interpolation driving, set the finish point of each axis. For helical interpolation, set the feed amount of Z and U axis. The finish point should be specified relative value to the current position. **VC** cannot be used. **VB.NET** Sub Nmc_Pulse_VB(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Double) **C#** cannot be used. **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. **Return Value** None **Example** [VB.NET] Call Nmc_Pulse_VB(No, IcNo, AXIS_X, 2000) ' Set 2000 to output pulse number (X axis). Call Nmc_Pulse_VB(No, IcNo, AXIS_Y, 300) ' Set 300 to interpolation finish point (Y axis). Call Nmc_Pulse_VB(No, IcNo, AXIS_Z, -400) ' Set -400 to interpolation finish point (Z axis).

Function Name	Function and Content

Nmc_DecP	Set manual decelerating point. (VC, C# only)
	
	VC void Nmc_DecP(int No, int IcNo, int axis, ULONG wdata);
	VB.NET cannot be used.
	C# void MC8000P.Nmc_DecP(int No, int IcNo, AXIS axis, uint wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_DecP(No, IcNo, AXIS_U, 30000); // Set 30000 to manual decelerating point (U axis)
	[C#] MC8000P.Nmc_DecP(No, IcNo, AXIS.U, 30000);

Nmc_DecP_VB	Set manual decelerating point. (VB.NET only) **VC** cannot be used. **VB.NET** Sub Nmc_DecP_VB(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Double) **C#** cannot be used. **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. **Return Value** None **Example** [VB.NET] Call Nmc_DecP_VB(No, IcNo, AXIS_X, 40000) ' Set 40000 to manual decelerating point (X axis).

Nmc_Center	Set circular center point.
	VC void Nmc_Center(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Center(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Center(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Center(No, IcNo, AXIS_Y, 1500); // Set 1500 to circular center point (Y axis).
	[VB.NET] Call Nmc_Center(No, IcNo, AXIS_Y, 1500)
	[C#] MC8000P.Nmc_Center(No, IcNo, AXIS.Y, 1500);

Function Name	Function and Content

Nmc_Lp	Set logical position counter.
	
	VC void Nmc_Lp(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Lp(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Lp(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Lp(No, IcNo, AXIS_ALL, 0); // Set 0 to logical position counter of all axes.
	[VB.NET] Call Nmc_Lp(No, IcNo, AXIS_ALL, 0)
	[C#] MC8000P.Nmc_Lp(No, IcNo, AXIS.ALL, 0);

Nmc_Ep	Set real position counter.
	
	VC void Nmc_Ep(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Ep(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Ep(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Ep(No, IcNo, AXIS_ALL, 0); // Set 0 to real position counter of all axes.
	[VB.NET] Call Nmc_Ep(No, IcNo, AXIS_ALL, 0)
	[C#] MC8000P.Nmc_Ep(No, IcNo, AXIS.ALL, 0);

Function Name	Function and Content

Nmc_CompP	Set software limit +register.
	
	VC void Nmc_CompP(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_CompP(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_CompP(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_CompP(No, IcNo, AXIS_X, 50000); // Set 50000 to software limit +register (X axis)
	[VB.NET] Call Nmc_CompP(No, IcNo, AXIS_X, 50000)
	[C#] MC8000P.Nmc_CompP(No, IcNo, AXIS.X, 50000);

Nmc_CompM	Set software limit -register.
	
	VC void Nmc_CompM(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_CompM(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_CompM(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_CompM(No, IcNo, AXIS_X, -50000); // Set -50000 to software limit +register (X axis).
	[VB.NET] Call Nmc_CompM(No, IcNo, AXIS_X, -50000)
	[C#] MC8000P.Nmc_CompM(No, IcNo, AXIS.X, -50000);

Function Name	Function and Content

Nmc_AccOfst	Set acceleration counter offsetting.
	
	VC void Nmc_AccOfst(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_AccOfst(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_AccOfst(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_AccOfst(No, IcNo, AXIS_Y, 20); // Set 20 to acceleration counter offsetting (Y axis).
	[VB.NET] Call Nmc_AccOfst(No, IcNo, AXIS_Y, 20)
	[C#] MC8000P.Nmc_AccOfst(No, IcNo, AXIS.Y, 20);

Nmc_HomeSpd	Set home search speed.
	
	VC void Nmc_HomeSpd(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_HomeSpd(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_HomeSpd(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_HomeSpd(No, IcNo, AXIS_Z, 200); // Set 200 to home search speed (Z axis).
	[VB.NET] Call Nmc_HomeSpd(No, IcNo, AXIS_Z, 200)
	[C#] MC8000P.Nmc_HomeSpd(No, IcNo, AXIS.U, 200);

Function Name	Function and Content

Nmc_LpMax	Set logical position counter maximum value.
	
	VC void Nmc_LpMax(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_LpMax(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_LpMax(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_LpMax(No, IcNo, AXIS_U, 2000); // Set 2000 to logical position counter maximum value (U axis).
	[VB.NET] Call Nmc_LpMax(No, IcNo, AXIS_U, 2000)
	[C#] MC8000P.Nmc_LpMax(No, IcNo, AXIS.U, 2000);

Nmc_RpMax	Set real position counter maximum value.
	
	VC void Nmc_RpMax(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_RpMax(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_RpMax(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_RpMax(No, IcNo, AXIS_X, 1000); // Set 1000 to real position counter maximum value (X axis).
	[VB.NET] Call Nmc_RpMax(No, IcNo, AXIS_X, 1000)
	[C#] MC8000P.Nmc_RpMax(No, IcNo, AXIS.X, 1000);

Function Name	Function and Content

Nmc_MR0	Set multi-purpose register 0.
	
	VC void Nmc_MR0(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_MR0(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_MR0(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_MR0(No, IcNo, AXIS_Z, 500000); // Set 500000 to multi-purpose register 0 (Z axis).
	[VB.NET] Call Nmc_MR0(No, IcNo, AXIS_Z, 500000)
	[C#] MC8000P.Nmc_MR0(No, IcNo, AXIS.Z, 500000);

Nmc_MR1	Set multi-purpose register 1.
	
	VC void Nmc_MR1(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_MR1(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_MR1(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_MR1(No, IcNo, AXIS_U, 5000); // Set 5000 to multi-purpose register 0(U axis).
	[VB.NET] Call Nmc_MR1(No, IcNo, AXIS_U, 5000)
	[C#] MC8000P.Nmc_MR1(No, IcNo, AXIS.U, 5000);

Function Name	Function and Content

Nmc_MR2	Set multi-purpose register 2.
	
	VC void Nmc_MR2(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_MR2(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_MR2(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_MR2(No, IcNo, AXIS_X, 5000000); // Set 5000000 to multi-purpose register 2 (X axis).
	[VB.NET] Call Nmc_MR2(No, IcNo, AXIS_X, 5000000)
	[C#] MC8000P.Nmc_MR2(No, IcNo, AXIS.X, 5000000);

Nmc_MR3	Set multi-purpose register 3.
	
	VC void Nmc_MR3(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_MR3(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_MR3(int No, int IcNo, AXIS axis, int wdata);
	
	In No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_MR3(No, IcNo, AXIS_Y, 30000); // Set 30000 to multi-purpose register 2(Y axis).
	[VB.NET] Call Nmc_MR3(No, IcNo, AXIS_Y, 30000)
	[C#] MC8000P.Nmc_MR3(No, IcNo, AXIS.Y, 30000);

Function Name	Function and Content

Nmc_SpeedInc	Set speed increase/decrease value setting
	
	VC void Nmc_SpeedInc(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_SpeedInc(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_SpeedInc(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_SpeedInc(No, IcNo, AXIS_Z, 100); // Set 100 to speed increase/decrease value(Z-axis).
	[VB.NET] Call Nmc_SpeedInc(No, IcNo, AXIS_Z, 100)
	[C#] MC8000P.Nmc_SpeedInc(No, IcNo, AXIS.Z, 100);

Nmc_Timer	Set timer value
	
	VC void Nmc_Timer(int No, int IcNo, int axis, long wdata);
	VB.NET Sub Nmc_Timer(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_Timer(int No, int IcNo, AXIS axis, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Specify AXIS_X, AXIS_Y and so on. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_Timer(No, IcNo, AXIS_U, 10000); // Set 10000 to timer value (U axis)
	[VB.NET] Call Nmc_Timer(No, IcNo, AXIS_U, 10000)
	[C#] MC8000P.Nmc_Timer(No, IcNo, AXIS.U, 10000);

Function Name	Function and Content

Nmc_TPMax	Set Interpolation/finish point Maximum Value Setting
	
	VC void Nmc_TPMax(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_TPMax(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_TPMax(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_TPMax(No, IcNo, 10000); // Set 10000 to Interpolation/finish point Maximum Value.
	[VB.NET] Call Nmc_TPMax(No, IcNo, 10000)
	[C#] MC8000P.Nmc_TPMax(No, IcNo, 10000);

Nmc_HLNumber	Set helical rotation number.
	
	VC void Nmc_HLNumber(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_HLNumber(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer
	C# void MC8000P.Nmc_HLNumber(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_HLNumber(No, IcNo, 7); // Set 7 to helical rotation number.
	[VB.NET] Call Nmc_HLNumber(No, IcNo, 7)
	[C#] MC8000P.Nmc_HLNumber(No, IcNo, 7);

Nmc_HLValue	Set helical calculation value.
	
	VC void Nmc_HLValue(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_HLValue(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P.Nmc_HLValue(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_HLValue(No, IcNo, 36799); // Set 36799 to helical calculation value.
	[VB.NET] Call Nmc_HLValue(No, IcNo, 36799)
	[C#] MC8000P.Nmc_HLValue(No, IcNo, 36799);

Function Name	Function and Content
Nmc_MRmMode	Set multi-purpose register mode. **VC** void Nmc_MRmMode(int No, int IcNo, int axis, long WR6_data); **VB.NET** Sub Nmc_MRmMode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer) **C#** void MC8000P.Nmc_MRmMode(int No, int IcNo, AXIS axis, int WR6_data); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. **Return Value** None **Example** // Set comparative object (current speed) and comparison condition (>) of MR1 to X axis multi-purpose register mode. [VC] Nmc_MRmMode(No, IcNo, AXIS_X, 0x0060); [VB.NET] Call Nmc_MRmMode(No, IcNo, AXIS_X, &H60) [C#] MC8000P.Nmc_MRmMode(No, IcNo, AXIS.X, 0x0060);

Nmc_PIO1Mode	Set PIO signal setting 1 (Function of nPIO7～0).
	
	VC void Nmc_PIO1Mode(int No, int IcNo, int axis, long WR6_data);
	VB.NET Sub Nmc_PIO1Mode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer)
	C# void MC8000P.Nmc_PIO1Mode(int No, int IcNo, AXIS axis, int WR6_data);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to set data. Multiple axes can be assigned. See [5.1.3] (2) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_PIO1Mode(No, IcNo, AXIS_X, 0x0055); //Set general purpose output by X-axis PIO signal 0~3.
	[VB.NET] Call Nmc_PIO1Mode(No, IcNo, AXIS_X, &H55)
	[C#] MC8000P.Nmc_PIO1Mode(No, IcNo, AXIS.X, 0x0055);

Function Name	Function and Content
Nmc_PIO2Mode	Set PIO signal setting 2/others (synchronous pulse output logic, pulse width and so on.) **VC** void Nmc_PIO2Mode(int No, int IcNo, int axis, long WR6_data); **VB.NET** Sub Nmc_PIO2Mode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer) **C#** void MC8000P.Nmc_PIO2Mode(int No, int IcNo, AXIS axis, int WR6_data); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. **Return Value** None **Example** //Set positive to X axis pulse signal and 1msec to pulse width. [VC] Nmc_PIO2Mode(No, IcNo, AXIS_X, 0x0070); [VB.NET] Call Nmc_PIO2Mode(No, IcNo, AXIS_X, &H70) [C#] MC8000P.Nmc_PIO2Mode(No, IcNo, AXIS.X, 0x0070);
Nmc_HMSrch1Mode	Set automatic home search mode setting 1. **VC** void Nmc_HMSrch1Mode(int No, int IcNo, int axis, long WR6_data); **VB.NET** Sub Nmc_HMSrch1Mode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer) **C#** void MC8000P.Nmc_HMSrch1Mode(int No, int IcNo, AXIS axis, int WR6_data); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. **Return Value** None **Example** Set as follows to automatic home search mode setting 1. ・Step 1,2,3,4：Execute ・Direction of Step 1,2,3 ：－ direction ・Search signal of Step1,2：STOP1 ・DCC output, RP clear, LP clear of STEP2：Disable ・DCC output, RP clear, LP clear of STEP3：Enable [VC] Nmc_HMSrch1Mode(No, IcNo, AXIS_X, 0xFC37); [VB.NET] Call Nmc_HMSrch1Mode(No, IcNo, AXIS_X, &HFC37) [C#] MC8000P.Nmc_HMSrch1Mode(No, IcNo, AXIS.X, 0xFC37);

Function Name	Function and Content
Nmc_HMSrch2Mode	Set automatic home search mode setting 2. **VC** void Nmc_HMSrch2Mode(int No, int IcNo, int axis, long WR6_data); **VB.NET** Sub Nmc_HMSrch2Mode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer) **C#** void MC8000P.Nmc_HMSrch2Mode(int No, int IcNo, AXIS axis, int WR6_data); Input Parameter No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. Return Value None Example Set as follows to automatic home search mode setting 2. ・Step 2&3：Disable ・LP clear at automatic home search termination：Enable ・RP clear at automatic home search termination：Enable ・DCC pulse logic：10μsec ・DCC pulse width：Hi pulse ・Timer between steps：Disable ・Timer value：0 [VC] Nmc_HMSrch2Mode(No, IcNo, AXIS_X, 0x0006); [VB.NET] Call Nmc_HMSrch2Mode(No, IcNo, AXIS_X, &H6) [C#] MC8000P.Nmc_HMSrch2Mode(No, IcNo, AXIS.X, 0x0006);
Nmc_FilterMode	Set input signal mode. **VC** void Nmc_FilterMode(int No, int IcNo, int axis, long WR6_data); **VB.NET** Sub Nmc_FilterMode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer) **C#** void MC8000P.Nmc_FilterMode(int No, int IcNo, AXIS axis, int WR6_data); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. **Return Value** None **Example** Set all the filters enable and delay 512μs to X axis input signal mode. [VC] Nmc_FilterMode(No, IcNo, AXIS_X, 0xAAFF); [VB.NET] Call Nmc_FilterMode(No, IcNo, AXIS_X, &HAAFF) [C#] MC8000P.Nmc_FilterMode(No, IcNo, AXIS.X, 0xAAFF);

Function Name	Function and Content
Nmc_Sync0Mode	Set synchronous action SYNC0. **VC** void Nmc_Sync0Mode(int No, int IcNo, int axis, long WR6_data); **VB.NET** Sub Nmc_Sync0Mode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer) **C#** void MC8000P.Nmc_Sync0Mode(int No, int IcNo, AXIS axis, int WR6_data); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. **Return Value** None **Example** Set [Save logical position counter to MR0 after starting constant speed area of acceleration drive during X axis driving.]. [VC] Nmc_Sync0Mode(No, IcNo, AXIS_X, 0x0054); [VB.NET] Call Nmc_Sync0Mode(No, IcNo, AXIS_X, &H54) [C#] MC8000P.Nmc_Sync0Mode(No, IcNo, AXIS.X, 0x0054);
Nmc_Sync1Mode	Set synchronous action SYNC1. **VC** void Nmc_Sync1Mode(int No, int IcNo, int axis, long WR6_data); **VB.NET** Sub Nmc_Sync1Mode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer) **C#** void MC8000P.Nmc_Sync1Mode(int No, int IcNo, AXIS axis, int WR6_data); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. **Return Value** None **Example** Set [Save current timer value to MR1 after MR1 object changes to positive during X axis driving.]. [VC] Nmc_Sync1Mode(No, IcNo, AXIS_X, 0x0071); [VB.NET] Call Nmc_Sync1Mode(No, IcNo, AXIS_X, &H71) [C#] MC8000P.Nmc_Sync1Mode(No, IcNo, AXIS.X, 0x0071);

Function Name	Function and Content
Nmc_Sync2Mode	Set synchronous action SYNC2. **VC** void Nmc_Sync2Mode(int No, int IcNo, int axis, long WR6_data); **VB.NET** Sub Nmc_Sync2Mode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer) **C#** void MC8000P.Nmc_Sync2Mode(int No, int IcNo, AXIS axis, int WR6_data); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. **Return Value** None **Example** Set [Activate X axis SYNC3 after internal timer is expired during X axis driving and then save logical position counter to MR2.]. [VC] Nmc_Sync2Mode(No, IcNo, AXIS_X, 0x0252); [VB.NET] Call Nmc_Sync2Mode(No, IcNo, AXIS_X, &H252) [C#] MC8000P.Nmc_Sync2Mode(No, IcNo, AXIS.X, 0x0252);
Nmc_Sync3Mode	Set synchronous action SYNC3. **VC** void Nmc_Sync3Mode(int No, int IcNo, int axis, long WR6_data); **VB.NET** Sub Nmc_Sync3Mode(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal WR6_data As Integer) **C#** void MC8000P.Nmc_Sync3Mode(int No, int IcNo, AXIS axis, int WR6_data); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to set data. Multiple axes can be assigned. See [5.1.3] (2) for more details. wdata Data to be set. **Return Value** None **Example** Set [Activate Y-axis SYNC0 after MR3 object changes to positive during X axis driving and then sav logical position counter to MR3.]. [VC] Nmc_Sync3Mode(No, IcNo, AXIS_X, 0x1051); [VB.NET] Call Nmc_Sync3Mode(No, IcNo, AXIS_X, &H1051) [C#] MC8000P.Nmc_Sync3Mode(No, IcNo, AXIS.X, 0x1051);

Function Name	Function and Content

Nmc_IPMode	Set interpolation mode.
	
	VC void Nmc_IPMode(int No, int IcNo, long wdata);
	VB.NET Sub Nmc_IPMode(ByVal No As Integer, ByVal IcNo As Integer, ByVal wdata As Integer)
	C# void MC8000P. Nmc_IPMode(int No, int IcNo, int wdata);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	wdata Data to be set.
	
	Return Value
	None
	
	Example
	[VC] Nmc_IPMode(No, IcNo, 0x0007); //Specify interpolation axis, X,Y,Z.
	[VB.NET] Call Nmc_IPMode(No, IcNo, &H7)
	[C#] MC8000P. Nmc_IPMode(No, IcNo, 0x0007);

Function Name	Function and Content

Nmc_ReadLp	Read out logical position counter.
	
	VC long Nmc_ReadLp(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadLp(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadLp(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,.
	wdata Data to be set.
	
	Return Value
	Current value of the logical position counter.
	
	Example
	[VC] Data = Nmc_ReadLp(No, IcNo, AXIS_X); // Read out logical position counter of X axis.
	[VB.NET] Data = Nmc_ReadLp(No, IcNo, AXIS_X)
	[C#] Data = MC8000P.Nmc_ReadLp(No, IcNo, AXIS.X);

Nmc_ReadEp	Read out real position counter.
	
	VC long Nmc_ReadEp(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadEp(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadEp(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	
	Return Value
	Current value of the real position counter.
	
	Example
	[VC] Data = Nmc_ReadEp(No, IcNo, AXIS_Y); // Read out real position counter of Y axis.
	[VB.NET] Data = Nmc_ReadEp(No, IcNo, AXIS_Y) ' Read out real position counter of Y axis.
	[C#] Data = MC8000P.Nmc_ReadEp(No, IcNo, AXIS.Y)

Function Name	Function and Content

Nmc_ReadSpeed	Read out the current drive speed.
	
	VC long Nmc_ReadSpeed(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadSpeed(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer As Intege
	C# int MC8000P.Nmc_ReadSpeed(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	Return Value
	The current drive speed.
	
	Example
	[VC] Data = Nmc_ReadSpeed(No, IcNo, AXIS_Z); // Read out the current drive speed of Z-axis.
	[VB.NET] Data = Nmc_ReadSpeed(No, IcNo, AXIS_Z)
	[C#] Data = MC8000P.Nmc_ReadSpeed(No, IcNo, AXIS.Z);

Nmc_ReadAccDec	Read out the current acceleration/deceleration.
	
	Read out the value of the current acceleration or deceleration during driving.
	When the driving stops, the read data is random number.
	
	VC long Nmc_ReadAccDec(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadAccDec(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadAccDec(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis.
	
	Return Value
	Current value of acceleration/deceleration speed.
	
	Example
	[VC] Data = Nmc_ReadAccDec(No, IcNo, AXIS_U); // Read out the current acceleration/deceleration of U axis.
	[VB.NET] Data = Nmc_ReadAccDec(No, IcNo, AXIS_U)
	[C#] Data = MC8000P.Nmc_ReadAccDec(No, IcNo, AXIS.U);

Function Name	Function and Content

Nmc_ReadMR0	Read multi-purpose register 0.
	
	VC long Nmc_ReadMR0(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadMR0(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer)As Intege
	C# int MC8000P.Nmc_ReadMR0(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	
	Return Value
	Value of multi-purpose register 0.
	
	Example
	[VC] Data = Nmc_ReadMR0(No, IcNo, AXIS_X); // Read multi-purpose register 0 of X axis
	[VB.NET] Data = Nmc_ReadMR0(No, IcNo, AXIS_X)
	[C#] Data = MC8000P.Nmc_ReadMR0(No, IcNo, AXIS.X);

Nmc_ReadMR1	Read multi-purpose register 1.
	
	VC long Nmc_ReadMR1(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadMR1(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Intege
	
	C# int MC8000P.Nmc_ReadMR1(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for
	U axis,
	
	Return Value
	Value of multi-purpose register 1.
	
	Example
	[VC] Data = Nmc_ReadMR1(No, IcNo, AXIS_Y); // Read multi-purpose register 1of Y axis
	[VB.NET] Data = Nmc_ReadMR1(No, IcNo, AXIS_Y)
	[C#] Data = MC8000P.Nmc_ReadMR1(No, IcNo, AXIS.Y);

Function Name	Function and Content

Nmc_ReadMR2	Read multi-purpose register 2.
	
	VC long Nmc_ReadMR2(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadMR2(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadMR2(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	
	Return Value
	Value of multi-purpose register 2.
	
	Example
	[VC] Data = Nmc_ReadMR2(No, IcNo, AXIS_Z); // Read multi-purpose register 2 of Z axis
	[VB.NET] Data = Nmc_ReadMR2(No, IcNo, AXIS_Z)
	[C#] Data = MC8000P.Nmc_ReadMR2(No, IcNo, AXIS.Z);

Nmc_ReadMR3	Read multi-purpose register 3.
	
	VC long Nmc_ReadMR3(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadMR3(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadMR3(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	
	Return Value
	Value of multi-purpose register 3.
	
	Example
	[VC] Data = Nmc_ReadMR3(No, IcNo, AXIS_U); // Read multi-purpose register 3 of U axis
	[VB.NET] Data = Nmc_ReadMR3(No, IcNo, AXIS_U)
	[C#] Data = MC8000P.Nmc_ReadMR3(No, IcNo, AXIS.U);

Function Name	Function and Content

Nmc_ReadCT	Read current timer value.
	VC long Nmc_ReadCT(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadCT(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadCT(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	
	Return Value
	Current value of timer.
	
	Example
	[VC] Data = Nmc_ReadCT(No, IcNo, AXIS_X); // Read current timer value of X axis.
	[VB.NET] Data = Nmc_ReadCT(No, IcNo, AXIS_X)
	[C#] Data = MC8000P.Nmc_ReadMR3(No, IcNo, AXIS.X);

Nmc_ReadTX	Read interpolation/finish point Maximum Value.
	
	VC long Nmc_ReadTX(int No, int IcNo);
	VB.NET Function Nmc_ReadTX(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadTX(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return Value
	Value of interpolation/finish point maximum value.
	
	Example
	[VC] Data = Nmc_ReadTX(No, IcNo); // Read interpolation/finish point Maximum Value.
	[VB.NET] Data = Nmc_ReadTX(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadTX(No, IcNo);
	
	Note
	Value to be read is different from that of before interpolation driving and during driving.
	Finish maximum value of interpolation segment which is being input is read before interpolation driving. And, finish point maximum value of the current interpolation driving segment is read during interpolation driving.

Function Name	Function and Content

Nmc_ReadCHLN	Read current helical rotation number.
	
	VC long Nmc_ReadCHLN(int No, int IcNo);
	VB.NET Function Nmc_ReadCHLN(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadCHLN(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return Value
	Current number of helical rotation.
	
	Example
	[VC] Data = Nmc_ReadCHLN(No, IcNo); // Read current helical rotation number.
	[VB.NET] Data = Nmc_ReadCHLN(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadCHLN(No, IcNo);

Nmc_ReadHLV	Read helical calculation value.
	Use this function when reading the result of helical calculation command(6Bh,6Ch)
	
	VC long Nmc_ReadHLV(int No, int IcNo);
	VB.NET Function Nmc_ReadHLV(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_ReadHLV(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return Value
	Value of helical calculation.
	
	Example
	[VC] Data = Nmc_ReadHLV(No, IcNo); // Read helical calculation value.
	[VB.NET] Data = Nmc_ReadHLV(No, IcNo)
	[C#] Data = MC8000P.Nmc_ReadHLV(No, IcNo);

Nmc_ReadWR1	Read WR1 setting value.
	
	VC long Nmc_ReadWR1(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadWR1(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadWR1(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	
	Return Value
	WR1 value.
	
	Example
	[VC] Data = Nmc_ReadWR1(No, IcNo, AXIS_Y); // Read WR1 setting value of Y axis.
	[VB.NET] Data = Nmc_ReadWR1(No, IcNo, AXIS_Y)
	[C#] Data = MC8000P. Nmc_ReadWR1(No, IcNo, AXIS.Y);

Function Name	Function and Content

Nmc_ReadWR2	Read WR2 setting value.
	
	VC long Nmc_ReadWR2(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadWR2(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadWR2(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	
	Return Value
	WR2 value.
	
	Example
	[VC] Data = Nmc_ReadWR2(No, IcNo, AXIS_Z); // Read WR2 setting value of Z axis.
	[VB.NET] Data = Nmc_ReadWR2(No, IcNo, AXIS_Z) ' Read WR2 setting value of Z axis.
	[C#] Data = MC8000P. Nmc_ReadWR2(No, IcNo, AXIS.Z);

Nmc_ReadWR3	Read WR3 setting value.
	
	VC long Nmc_ReadWR3(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadWR3(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadMR3(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	
	Return Value
	WR3 value.
	
	Example
	[VC] Data = Nmc_ReadWR3(No, IcNo, AXIS_U); // Read WR3 setting value of U axis.
	[VB.NET] Data = Nmc_ReadWR3(No, IcNo, AXIS_U)
	[C#] Data = MC8000P.Nmc_ReadWR3(No, IcNo, AXIS.U);

Function Name	Function and Content

Nmc_ReadMRM	Read multi-purpose mode register setting value.
	
	VC long Nmc_ReadMRM(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadMRM(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P. Nmc_ReadMRM(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis,
	
	Return Value
	Multi-purpose register mode value.
	
	Example
	[VC] Data = Nmc_ReadMRM(No, IcNo, AXIS_X); // Read multi-purpose mode register setting value of X axis.
	[VB.NET] Data = Nmc_ReadMRM(No, IcNo, AXIS_X)
	[C#] Data = MC8000P.Nmc_ReadMRM(No, IcNo, AXIS.X);

Nmc_ReadP1M	Read PIO signal setting value 1.
	
	VC long Nmc_ReadP1M(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadP1M(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadP1M(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis.
	
	Return Value
	PIO signal setting 1 value.
	
	Example
	[VC] Data = Nmc_ReadP1M(No, IcNo, AXIS_Y); // Read PIO signal setting value 1 of Y axis.
	[VB.NET] Data = Nmc_ReadP1M(No, IcNo, AXIS_Y)
	[C#] Data = MC8000P.Nmc_ReadP1M(No, IcNo, AXIS.Y);

Function Name	Function and Content
Nmc_ReadP2M	Read PIO signal setting value1 / other setting. **VC** long Nmc_ReadP2M(int No, int IcNo, int axis); **VB.NET** Function Nmc_ReadP2M(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer As Integer **C#** int MC8000P.Nmc_ReadP2M(int No, int IcNo, AXIS axis); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis. **Return Value** PIO signal setting 1/other setting value. **Example** // Read PIO signal setting value1 / other setting of Z axis. [VC] Data = Nmc_ReadP2M(No, IcNo, AXIS_Z); [VB.NET] Data = Nmc_ReadP2M(No, IcNo, AXIS_Z) [C#] Data = MC8000P.Nmc_ReadP2M(No, IcNo, AXIS.Z);

Nmc_ReadAc	Read out the current acceleration/deceleration.
	
	VC long Nmc_ReadAc(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadAc(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadAc(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis.
	
	Return Value
	Acceleration speed value to be set.
	
	Example
	[VC] Data = Nmc_ReadAc(No, IcNo, AXIS_U); // Read out the current acceleration/deceleration of U axis. [VB.NET] Data = Nmc_ReadAc(No, IcNo, AXIS_U)
	[C#] Data = MC8000P.Nmc_ReadAc(No, IcNo, AXIS.U);

Function Name	Function and Content

Nmc_ReadSetSpeed	Read out drive speed setting value.
	
	VC long Nmc_ReadSetSpeed(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadSetSpeed(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadSetSpeed(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis.
	
	Return Value
	Drive speed value to be set.
	
	Example
	[VC] Data = Nmc_ReadSetSpeed(No, IcNo, AXIS_Y); // Read out drive speed setting value of Y axis.
	[VB.NET] Data = Nmc_ReadSetSpeed(No, IcNo, AXIS_Y)
	[C#] Data = MC8000P.Nmc_ReadSetSpeed(No, IcNo, AXIS.Y);

Nmc_ReadPulse	Read moving pulse number/ Finish Point Setting value.
	
	VC long Nmc_ReadPulse(int No, int IcNo, int axis);
	VB.NET Function Nmc_ReadPulse(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P.Nmc_ReadPulse(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis.
	
	Return Value
	Moving Pulse Number/ Finish Point Setting value to be set.
	
	Example
	[VC] Data = Nmc_ReadPulse(No, IcNo, AXIS_Z); // Read Moving Pulse Number/ Finish Point Setting value of Z axis.
	[VB.NET] Data = Nmc_ReadPulse(No, IcNo, AXIS_Z)
	[C#] Data = MC8000P.Nmc_ReadSetSpeed(No, IcNo, AXIS.Z);

Function Name	Function and Content

Nmc_GetDriveStatus	Drive Status Reading. Use this function to check whether drive of specified axis finishes.
	
	VC int Nmc_GetDriveStatus(int No, int IcNo, int axis);
	VB.NET Function Nmc_GetDriveStatus(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer
	C# int MC8000P. Nmc_GetDriveStatus(int No, int IcNo, AXIS axis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	Axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis.
	
	Return value
	If all the specified axes have finished driving, the return value is 0.
	If more than one of the specified axes is/are driving, the return value is nonzero.
	
	Example
	[VC] if(Nmc_GetDriveStatus(No, IcNo, AXIS_X) == 0) // When X axis has finished driving.
	AfxMessageBox("X axis has finished driving");
	else
	AfxMessageBox("X axis is driving ");
	[VB.NET] If Nmc_GetDriveStatus(No, IcNo, AXIS_X) = 0 Then
	Call MsgBox("X axis has finished driving")
	Else
	Call MsgBox("X axis is driving")
	End If
	[C#] if(MC8000P.Nmc_GetDriveStatus(No, IcNo, AXIS.X) == 0)
	MessageBox.Show(" X axis has finished driving ");
	else
	MessageBox.Show(" X axis is driving ");

Function Name	Function and Content

Nmc_GetCNextStatus	Read the status of ready signal for writing of continuous interpolation.
	The user can use to check whether the signal for the writing of continuous interpolation is ready or not during continuous interpolation execution.
	
	VC int Nmc_GetCNextStatus(int No, int IcNo);
	VB.NET Function Nmc_GetCNextStatus(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P.Nmc_GetCNextStatus(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return Value
	If the signal for the writing of continuous interpolation is ready, the return value is nonzero.
	If the signal for the writing of continuous interpolation is not ready, the return value is 0.
	
	Example
	[VC] if(Nmc_GetCNextStatus(No, IcNo) != 0) // When the signal for the writing is ready.
	AfxMessageBox("The signal for the writing of continuous interpolation is ready ");
	else
	AfxMessageBox("The signal for the writing of continuous interpolation is not ready ");
	[VB.NET] If Nmc_GetCNextStatus(No, IcNo) <> 0 Then
	Call MsgBox("The signal for the writing of continuous interpolation is ready ")
	Else
	Call MsgBox("The signal for the writing of continuous interpolation is not ready ")
	End If
	[C#] if(MC8000P.Nmc_GetCNextStatus(No, IcNo) != 0)
	MessageBox.Show(" The signal for the writing of continuous interpolation is ready ");
	else
	MessageBox.Show(" The signal for the writing of continuous interpolation is not ready ");
	");
	Nmc_GetSc
	Read the value of Continuous interpolation pre-buffer Stack Counter.
	
	VC int Nmc_GetSc(int No, int IcNo);
	VB.NET Function Nmc_GetSc(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# int MC8000P. Nmc_GetSc(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return Value
	The value of the current Continuous interpolation pre-buffer Stack Counter.
	
	Example
	[VC] Data = Nmc_GetSc(No, IcNo); // Read the value of BP interpolation stack counter.
	[VB.NET] Data = Nmc_GetSc(No, IcNo)
	[C#] Data = MC8000P.Nmc_GetSc(No, IcNo)

Function Name	Function and Content
Nmc_WriteRegSetAxis	Write data into one specified write register of WR1~WR3 for the specified axis. **VC** void Nmc_WriteRegSetAxis(int No, int IcNo, int axis, int RegNumber, long wdata); **VB.NET** Sub Nmc_WriteRegSetAxis(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal RegNumber As Integer, ByVal wdata As Integer) **C#** void MC8000P.Nmc_WriteRegSetAxis(int No, int IcNo, AXIS axis, int RegNumber, int wdata); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. axis Axis to write data. Multiple axes can be assigned. See [5.1.3] (2) for more details. RegNumber Write register number to write data. [VC][VB.NET] Assign MCX_WR1 for WR1, MCX_WR2 for WR2, MCX_WR3 for WR3 [C#] Assign REG_MCX.RR1 for RR1, REG_MCX.RR2 for RR2. See [5.1.3] (1) for more detail. wdata Data to be written **Return Value** None **Example** Set hardware limit enable. [VC] Nmc_WriteRegSetAxis(No, IcNo, AXIS_ALL, MCX_WR2, 0x0800); [VB.NET] Call Nmc_WriteRegSetAxis(No, IcNo, AXIS_ALL, MCX_WR2, &H800) [C#] MC8000P.Nmc_WriteRegSetAxis(No, IcNo, AXIS.ALL, REG_MCX.WR2, 0x0800);
Nmc_ReadRegSetAxis	Read out data from the specified read register (either RR2 or RR3) for the specified axis. **VC** long Nmc_ReadRegSetAxis(int No, int IcNo, int axis, int RegNumber); **VB.NET** Function Nmc_ReadRegSetAxis(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal RegNumber As Integer) As Integer **C#** int MC8000P.Nmc_ReadRegSetAxis(int No, int IcNo, AXIS axis, int RegNumber); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis. See [5.1.3] (2) for more details. RegNumber Read register number to read data. [VC] [VB.NET] Assign MCX.RR1 for RR1, MCX.RR2 for RR2. [C#] Assign REG_MCX.RR1 for RR1, REG_MCX.RR2 for RR2. See [5.1.3] (1) for more detail. **Return Value** The data of the specified read register for the specified axis. **Example** Read out the data of X axis RR2. [VC] Data = Nmc_ReadRegSetAxis(No, IcNo, AXIS_X, MCX_RR2); [VB.NET] Data = Nmc_ReadRegSetAxis(No, IcNo, AXIS_X, MCX_RR2) [C#] Data = MC8000P.Nmc_ReadRegSetAxis(No, IcNo, AXIS.X, REG_MCX.RR2); **Note** Regarding RR3 register, there are 2 kinds: Page 0 and Page1. The page can be specified by writing RR3 page display command (7Ah, 7Bh). It will be Page 0 at reset.

Function Name	Function and Content
Nmc_WriteData	Write the specified parameter into the specified axis. (Execute commands for data writing) **VC** void Nmc_WriteData(int No, int IcNo, int axis, int cmd, long wdata); **VB.NET** Sub Nmc_WriteData(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal cmd As Integer, ByVal wdata As Integer) **C#** void MC8000P.Nmc_WriteData(int No, int IcNo, AXIS axis, CMD cmd, int wdata); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. axis Axis to write data. Multiple axes can be assigned. See [5.1.3] (2) for more details. cmd Commands for data writing ((00)H～(0E)H, (61)H). E.g. Jerk setting is (00)H. wdata Data to be written **Return Value** None **Example** Set 1000 to the drive speed of all axes. Drive speed command code is (05)H. [VC] Nmc_WriteData(No, IcNo, AXIS_ALL, 0x05, 1000); [VB.NET] Call Nmc_WriteData(No, IcNo, AXIS_ALL, &H5, 1000) [C#] MC8000P.Nmc_WriteData(No, IcNo, AXIS.ALL, 0x05, 1000);
Nmc_ReadData	Read out data by executing commands for reading data. **VC** long Nmc_ReadData(int No, int IcNo, int axis, int cmd); **VB.NET** Function Nmc_ReadData(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal cmd As Integer) As Integer **C#** int MC8000P.Nmc_ReadData(int No, int IcNo, AXIS axis, CMD cmd); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. axis Axis to read data. Specify AXIS_X for X axis, AXIS_Y for Y axis, AXIS_Z for Z axis, AXIS_U for U axis. See [5.1.3] (2) for more details. cmd Commands for reading data ((10)H～(14)H). E.g. Logical position counter reading is (30)H. **Return Value** Data to be read. **Example** // Read the logical position counter of X axis. [VC] Data = Nmc_ReadData(No, IcNo, AXIS_X, 0x30); [VB.NET] Data = Nmc_ReadData(No, IcNo, AXIS_X, &H30) [C#] Data = MC8000P.Nmc_ReadData(No, IcNo, AXIS.X, 0x30);

Function Name	Function and Content

Nmc_2BPExecMC8500P	Execute 2-axis bit pattern interpolation using the specified interpolation data.
	This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread.
	
	VC DWORD Nmc_2BPExecMC8500P(int No, int IcNo, DATA_2BP* pData2Bp, int DataCnt, int IpAxis);
	
	VB.NET Function Nmc_2BPExecMC8500P(ByVal No As Integer, ByVal IcNo As Integer, ByVal pData2Bp As DATA_2BP(), ByVal DataCnt As Integer, ByVal IpAxis As Integer) As Integer
	
	C# Nmc_Status MC8000P.Nmc_2BPExecMC8500P(int No, int IcNo, DATA_2BP[] pData2Bp, int DataCnt, int IpAxis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	pData2Bp Pointer to an array of DATA_2BP structures (user-defined type in VB). Set the 2-axis BP interpolation data to DTA_2BP. See [5.1.3] (3) for DATA_2BP.
	DataCnt The number of 2-axis BP interpolation data. Specify the number of the DATA_2BP structure (user-defined type) array.
	IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4).
	Return Value
	If the function succeeds, the return value is BP_END. If the function fails, the return value is the following Error code. For C#, See [5.1.3] (1).
	
	■ Normal end
	BP_END BP interpolation has been successfully completed.
	
	■ Error code
	BP_CNT_ERR The number of the specified data is out of range.
	BP_ALREADY_EXEC BP interpolation or continuous interpolation is already running.
	BP_MALLOC_ERR Memory cannot be collocated.
	BP_PARAM_ERR The parameter is incorrect.
	BP_NOT_OPEN_ERR The specified board is not opened.
	BP_OTHER_ERR Other errors.
	BP_FUNC_ERR Unusable function is used.
	BP_STOP BP interpolation stopped during driving (too fast to stack next data).
	BP_USER_STOP The user aborted BP interpolation.
	BP_DRIVE_ERR Error occurred in the board during BP interpolation. (When the error status was set to RR0.)
	BP_STOP_CERR BP interpolation stopped during driving. (Due to RR2 error.)
	BP_GETSC_ERR Stack counter reading error
	
	When interpolation is terminated if end code is written, error code, BP_END or BP_STOP, is returned.
	
	Example
	[VC] // 2 axes BP interpolation data BP1P, BP1M, BP2P, BP2M
	DATA_2BP Data2Bp[2] = {{0x0000, 0x2BFF, 0xFFD4, 0x0000},
	{0xF6FE, 0x0000, 0x000F, 0x3FC0}};
	
	Nmc_IPMode(No, IcNo, 0x0003); //Interpolation mode setting. Assign X and Y axis.
	
	// Parameter setting related to speed(The actual setting description is omitted)
	Ret = Nmc_2BPExecMC8500P(No, IcNo, Data2Bp, 2, 0x3);
	// Execute 2-axis BP interpolation. The number of data is 2, X, Y axes.
	if(Ret == BP_END) AfxMessageBox("Successful completion"); // Return value is correct.

[VB.NET]

```
'2-axis BP interpolation data setting
  Dim Data2Bp(1) As DATA_2BP
  Data2Bp(0).Bp1p = 0
  Data2Bp(0).Bp1m = &H2BFFS
  Data2Bp(0).Bp2p = &HFFD4S
  Data2Bp(0).Bp2m = 0

  Data2Bp(1).Bp1p = &HF6FES
  Data2Bp(1).Bp1m = 0
  Data2Bp(1).Bp2m = &HFS
  Data2Bp(1).Bp2m = &H3FC0S

  Call Nmc_IPMode(No, IcNo, &H3)                    ' Interpolation mode setting.
                                                    Assign X and Y axis.

 ' Parameter setting related to speed(The actual setting description is omitted)

Ret = Nmc_2BPExecMC8500P(No, IcNo, Data2Bp, 2, &H3)
' Execute 2-axis BP interpolation. The number of data is 2, X, Y axes.
  If Ret = BP_END Then       ' Return value is correct
    Call MsgBox("Successful completion")
  End If
```

[C#]

```
 DATA_2BP [] Data2Bp = new DATA_2BP[2];
// 2-axis BP interpolation data
// Interpolation data setting
 Data2Bp[0].Bp1p = 0;              // 0000 0000 0000 0000   BP1 +direction 0pulse
 Data2Bp[0].Bp1m = 0x2BFF;         // 0010 1011 1111 1111   BP1 -direction 12pulse
 Data2Bp[0].Bp2p = 0xFFD4;         // 1111 1111 1101 0100   BP2 +direction 12pulse
 Data2Bp[0].Bp2m = 0;              // 0000 0000 0000 0000   BP2 -direction 0pulse

 Data2Bp[1].Bp1p = 0xF6FE;         // 1111 0110 1111 1110   BP1 +direction 13pulse
 Data2Bp[1].Bp1m = 0;              // 0000 0000 0000 0000   BP1 -direction 0pulse
 Data2Bp[1].Bp2p = 0xF;            // 0000 0000 0000 1111   BP2 +direction 4pulse
 Data2Bp[1].Bp2m = 0x3FC0;         // 0011 1111 1100 0000   BP2 +direction 8pulse

  MC8000P.Nmc_IPMode(No, IcNo, 0x0003);
  // Interpolation mode setting. Assign X and Y axis.

// Parameter setting related to speed(The actual setting description is omitted)

// Execute 2-axis BP interpolation. The number of data is 2, X, Y axes.

 Ret = MC8000.Nmc_2BPExecMC8500P(No, IcNo, Data2Bp, 2, 0x3);

if(Ret == Nmc_Status.BP_END)                        // Return value is
correct.
```

Note

Make sure to set interpolation axis and so on in interpolation mode setting (2A)H before executin this function. Set the same value to axis to be set in interpolation mode setting and axis specifie by argument IpAxis.

This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content

Nmc_3BPExecMC8500P	Execute 3-axis bit pattern interpolation using the specified interpolation data.
	This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread.
	
	VC DWORD Nmc_3BPExecMC8500P(int No, int IcNo, DATA_3BP* pData3Bp, int DataCnt, int IpAxis);
	VB.NET Function Nmc_3BPExecMC8500P(ByVal No As Integer, ByVal IcNo As Integer, ByVal pData3Bp As DATA_3BP(), ByVal DataCnt As Integer ByVal IpAxis As Integer) As Integer
	C# Nmc_Status MC8000P.Nmc_3BPExecMC8500P(int No, int IcNo, DATA_3BP[] pData3Bp, int DataCnt, int IpAxis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	pData3Bp Pointer to an array of DATA_3BP structures (user-defined type in VB). Set the 3-axis BP interpolation data to DATA_3BP. See [5.1.3] (3) for DATA_3BP.
	DataCnt The number of 3-axis BP interpolation data. Specify the number of the DATA_3BP structure (user-defined type) array.
	IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4).
	
	Return Value
	If the function succeeds, the return value is BP_END. If the function fails, the return value is the following Error code.
	For C#, See [5.1.3] (1).
	
	■ Normal end
	BP_END BP interpolation has been successfully completed.
	
	■ Error code
	BP_CNT_ERR The number of the specified data is out of range.
	BP_ALREADY_EXEC BP interpolation or continuous interpolation is already running.
	BP_MALLOC_ERR Memory cannot be collocated.
	BP_PARAM_ERR The parameter is incorrect.
	BP_NOT_OPEN_ERR The specified board is not opened.
	BP_OTHER_ERR Other errors.
	BP_FUNC_ERR Unusable function is used.
	BP_STOP BP interpolation stopped during driving (too fast to stack next data).
	BP_USER_STOP The user aborted BP interpolation.
	BP_DRIVE_ERR Error occurred in the board during BP interpolation. (When the error status was set to RR0.)
	BP_STOP_CERR BP interpolation stopped during driving. (Due to RR2 error.)
	BP_GETSC_ERR Stack counter reading error
	
	When interpolation is terminated if end code is written, error code, BP_END or BP_STOP, is returned.
	
	Example
	[VC]
	// 3 axis BP interpolation data BP1P, BP1M, BP2P, BP2M, BP3P, BP3M
	DATA_3BP Data3Bp[2] = {{0xFF30, 0, 0, 0x84FF, 0, 0xAC35},
	{0xAC35, 0, 0xC000, 0x36E7, 0xC000, 0x3F3F}};
	
	Nmc_IPMode(No, IcNo, 0x0007); // Interpolation mode setting. Assign X, Y and Z axis.
	
	// Parameter setting related to speed(The actual setting description is omitted)

```
                    Ret = Nmc_3BPExecMC8500P(No, IcNo, Data3Bp, 2, 0x7); // Execute 3-axis BP
                                                                          interpolation. The
                                                                          number of data is 2, X,
                                                                          Y, Z axes.

                    if(Ret == BP_END)    AfxMessageBox("Successful completion ");          // Return
value is correct.

    [VB.NET]
        '3-axis BP interpolation data
        'Interpolation data setting
         Dim Data3Bp(1) As DATA_3BP
         Data3Bp(0).Bp1p = &HFF30S
         Data3Bp(0).Bp1m = 0
         Data3Bp(0).Bp2p = 0
         Data3Bp(0).Bp2m = &H84FFS
         Data3Bp(0).Bp3p = 0
         Data3Bp(0).Bp3m = &HAC35S

        Data3Bp(1).Bp1p = &HAC35S
         Data3Bp(1).Bp1m = 0
         Data3Bp(1).Bp2p = &HC000S
         Data3Bp(1).Bp2m = &H36E7S
         Data3Bp(1).Bp3p = &HC000S
         Data3Bp(1).Bp3m = &H3F3FS

            Call Nmc_IPMode(No, IcNo, &H7)                    ' Interpolation mode setting.
                                                               Assign X, Y and Z axis.

          ' Parameter setting related to speed(The actual setting description is omitted)


          Ret = Nmc_3BPExecMC8500P(No, IcNo, Data3Bp, 2, &H7)
          ' Execute 2-axis BP interpolation. The number of data is 2, X, Y, Z axes.

          If Ret = BP_END Then      ' Return value is correct
               Call MsgBox("Successful completion")
          End If

    [C#]
          DATA_3BP [] Data3Bp = new DATA_3BP[2];
           // 3-axis BP interpolation data
          // Interpolation data setting
           Data3Bp[0].Bp1p = 0xFF30;        // 1111 1111 0011 0000        BP1 +direction   10pulse
           Data3Bp[0].Bp1m = 0;             // 0000 0000 0000 0000        BP1 –direction    0pulse
           Data3Bp[0].Bp2p = 0;             // 0000 0000 0000 0000        BP2 +direction    0pulse
           Data3Bp[0].Bp2m = 0x84FF;        // 1000 0100 1111 1111        BP2 -direction   10pulse
           Data3Bp[0].Bp3p = 0;             // 0000 0000 0000 0000        BP3 +direction    0pulse
           Data3Bp[0].Bp3m = 0xAC35;        // 1010 1100 0011 0101        BP3 -direction    8pulse

           Data3Bp[1].Bp1p = 0xAC35;        // 1010 1100 0011 0101        BP1 +direction   8pulse
           Data3Bp[1].Bp1m = 0;             // 0000 0000 0000 0000        BP1 -direction   0pulse
           Data3Bp[1].Bp2p = 0xC000;        // 1100 0000 0000 0000        BP2 +direction   2pulse
           Data3Bp[1].Bp2m = 0x36E7;        // 0011 0110 1110 0111        BP2 -direction   10pulse
           Data3Bp[1].Bp3p = 0xC000;        // 1100 0000 0000 0000        BP3 +direction   2pulse
           Data3Bp[1].Bp3m = 0x3F3F;        // 0011 1111 0011 1111        BP3 -direction   12pulse

            MC8000P.Nmc_IPMode(No, IcNo,0x0007);              // Interpolation mode setting.
                                                               Assign X, Y and Z axis.

          // Parameter setting related to speed(The actual setting description is omitted)

           Ret = MC8000P.Nmc_3BPExecMC8500P(No, IcNo, Data3Bp, 2, 0x7); // Execute 3-axis BP
interpolation.
```

	if(Ret == Nmc_Status.BP_END) // Return value is correct **Note** Make sure to set interpolation axis and so on in interpolation mode setting (2A)H before executing this function. Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis. This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content

Nmc_4BPExecMC8500P	Execute 4-axis bit pattern interpolation using the specified interpolation data.
	This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread.
	
	VC DWORD Nmc_4BPExecMC8500P(int No, int IcNo, DATA_4BP* pData4Bp, int DataCnt, int IpAxis);
	VB.NET Function Nmc_4BPExecMC8500P(ByVal No As Integer, ByVal IcNo As Integer, ByVal pData4Bp As DATA_4BP(), ByVal DataCnt As Integer, ByVal IpAxis As Integer) As Integer
	C# Nmc_Status MC8000P.Nmc_4BPExecMC8500P(int No, int IcNo, DATA_4BP[] pData4Bp, int DataCnt, int IpAxis);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	pData4Bp Pointer to an array of DATA_4BP structures (user-defined type in VB). Set the 4-axis BP interpolation data to DATA_4BP. See [5.1.3] (3) for DATA_4BP.
	DataCnt The number of 4-axis BP interpolation data. Specify the number of the DATA_4BP structure (user-defined type) array.
	IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4).
	
	Return Value
	If the function succeeds, the return value is BP_END. If the function fails, the return value is the following Error code. For C#, See [5.1.3] (1).
	
	■ Normal end
	BP_END BP interpolation has been successfully completed.
	
	■Error code
	BP_CNT_ERR The number of the specified data is out of range.
	BP_ALREADY_EXEC BP interpolation or continuous interpolation is already running.
	BP_MALLOC_ERR Memory cannot be collocated.
	BP_PARAM_ERR The parameter is incorrect.
	BP_NOT_OPEN_ERR The specified board is not opened.
	BP_OTHER_ERR Other errors.
	BP_STOP BP interpolation stopped during driving (too fast to stack next data).
	BP_USER_STOP The user aborted BP interpolation.
	BP_DRIVE_ERR Error occurred in the board during BP interpolation. (When the error status was set to RR0.)
	BP_STOP_CERR BP interpolation stopped during driving. (Due to RR2 error.)
	BP_GETSC_ERR Stack counter reading error
	
	When interpolation is terminated if end code is written, error code, BP_END or BP_STOP, is returned.
	
	Example
	[VC]
	// 4 axis BP interpolation data BP1P, BP1M, BP2P, BP2M, BP3P, BP3M, BP4P, BP4M
	DATA_4BP Data4Bp[2] = {{0xFFE4, 0x0000, 0x03FF, 0x4000, 0x0000, 0xFFFF, 0xFE80, 0x000F},
	{0x0000, 0x03FF, 0xFFD0, 0x0000, 0x4AAB, 0x0000, 0x1FFF, 0x0000}};
	
	Nmc_IPMode(No, IcNo,0x000F); // Interpolation mode setting. Assign X, Y, Z and U axis.
	
	// Parameter setting related to speed(The actual setting description is omitted)

```
// Execute 4-axis BP interpolation. The number of data is 2, X, Y, Z, U axes.

Ret = Nmc_4BPExecMC8500P(No, IcNo, Data4Bp, 2, 0xF);        if(Ret == BP_END)

AfxMessageBox("Successful completion ");          // Return value is correct.
```

[VB.NET]

```
'4-axis BP interpolation data
Dim Data4Bp(1) As DATA_4BP
Data4Bp(0).Bp1p = &HFFE4S
Data4Bp(0).Bp1m = 0
Data4Bp(0).Bp2p = &H3FFS
Data4Bp(0).Bp2m = &H4000S
Data4Bp(0).Bp3p = 0
Data4Bp(0).Bp3m = &HFFFFS
Data4Bp(0).Bp4p = &HFE80S
Data4Bp(0).Bp4m = &H000FS

Data4Bp(1).Bp1p = 0
Data4Bp(1).Bp1m = &H3FFS
Data4Bp(1).Bp2p = &HFFD0S
Data4Bp(1).Bp2m = 0
Data4Bp(1).Bp3p = &H4AABS
Data4Bp(1).Bp3m = 0
Data4Bp(1).Bp4p = &H1FFFS
Data4Bp(1).Bp4m = 0

Call Nmc_IPMode(No, IcNo, &HF)          ' Interpolation mode setting. Assign X, Y, Z and U axis.

' Parameter setting related to speed(The actual setting description is omitted)

'Execute 4-axis BP interpolation. The number of data is 2, X, Y, Z, U axes.
Ret = Nmc_4BPExecMC8500P(No, IcNo, Data4Bp, 2, &HF)

If Ret = BP_END Then       ' Return value is correct
    Call MsgBox("Successful completion ")
End If
```

[C#]

```
DATA_4BP [] Data4Bp = new DATA_4BP[2];                          // 4-axis BP interpolation data
// Interpolation data setting
Data4Bp[0].Bp1p = 0xFFE4;          // 1111 1111 1110 0100          BP1 +direction   12pulse
Data4Bp[0].Bp1m = 0;               // 0000 0000 0000 0000          BP1 +direction    0pulse
Data4Bp[0].Bp2p = 0x03FF;          // 0000 0011 1111 1111 BP2 +direction   10pulse
Data4Bp[0].Bp2m = 0x4000;          // 0100 0000 0000 0000          BP2 -direction   1pulse
Data4Bp[0].Bp3p = 0;               // 0000 0000 0000 0000          BP3 +direction   0pulse
Data4Bp[0].Bp3m = 0xFFFF;          // 1111 1111 1111 1111  BP3 -direction   16pulse
Data4Bp[0].Bp4p = 0xFE80;          // 1111 1110 1000 0000          BP4 +direction   6pulse
Data4Bp[0].Bp4m = 0x000F;          // 0000 0000 0000 1111          BP4 -direction   4pulse

Data4Bp[1].Bp1p = 0;               // 0000 0000 0000 0000          BP1 +direction   0pulse
Data4Bp[1].Bp1m = 0x03FF;          // 0000 0011 1111 1111 BP1 -direction   10pulse
Data4Bp[1].Bp2p = 0xFFD0;          // 1111 1111 1101 0000 BP2 +direction   11pulse
Data4Bp[1].Bp2m = 0;               // 0000 0000 0000 0000          BP2 -direction   0pulse
Data4Bp[1].Bp3p = 0x4AAB;          // 0100 1010 1010 1011          BP3 +direction   8pulse
Data4Bp[1].Bp3m = 0;             // 0000 0000 0000 0000    BP3 -direction   0pulse
Data4Bp[1].Bp4p = 0x1FFF;          // 0001 1111 1111 1111 BP4 +direction   13pulse
Data4Bp[1].Bp4m = 0;             // 0000 0000 0000 0000    BP4 -direction   0pulse

  MC8000P.Nmc_IPMode(No, IcNo, 0x000F);                         // Interpolation mode setting.
                                                                  Assign X,Y,Z and U axis.

// Parameter setting related to speed(The actual setting description is omitted)

Ret = MC8000P.Nmc_4BPExecMC8500P(No, IcNo, Data4Bp, 1, 0xF); // Execute 4-axis BP
interpolation.
```

	if(Ret == Nmc_Status.BP_END) // Return value is correct **Note** Make sure to set interpolation axis and so on in interpolation mode setting (2A)H before executing this function. Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis. This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content
Nmc_2BPExecMC8500P_BG	Execute 2-axis bit pattern interpolation in the background using the specified interpolation data. This function returns control right after the interpolation process started and executes the interpolation in the background. WM_BP_END message is sent to the specified window at the end of the interpolation and finishing status is passed. **VC** DWORD Nmc_2BPExecMC8500P_BG(HWND User_hWnd, int No, int IcNo, DATA_2BP* pData2Bp, int DataCnt, int IpAxis); **VB.NET** Function Nmc_2BPExecMC8500P_BG(ByVal User_hWnd As Integer, ByVal No As Integer, ByVal IcNo As Integer, ByVal pData2Bp As DATA_2BP(), ByVal DataCnt As Integer, ByVal IpAxis As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_2BPExecMC8500P_BG(System.IntPtr User_hWnd, int No, int IcNo, DATA_2BP[] pData2Bp, int DataCnt, int IpAxis,); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. pData2Bp Pointer to an array of DATA_2BP structures (user-defined type in VB). Set the 2-axis BP interpolation data to DATA_2BP. See [5.1.3] (3) for DATA_2BP. DataCnt The number of 2-axis BP interpolation data. Specify the number of the DATA_2BP structure (user-defined type) array.

IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4).

Return Value

If the interpolation process has been successfully started in the background, the return value is BP_START.
If an error occurred before starting the interpolation process, the return value is the following Error code (errors before starting the interpolation). For C#, See [5.1.3] (1).

■ Normal start

BP_START	BP interpolation has been successfully started in the background.

■ Error code (errors before starting the interpolation)

BP_CNT_ERR	The number of the specified data is out of range.
BP_ALREADY_EXEC	BP interpolation or continuous interpolation is already running.
BP_THREAD_ERR	Thread cannot be started.
BP_MALLOC_ERR	Memory cannot be allocated.
BP_PARAM_ERR	The parameter is incorrect.
BP_NOT_OPEN_ERR	The specified board is not opened.
BP_OTHER_ERR	Other errors.
BP_FUNC_ERR	Unusable function is used.

After the interpolation process has been successfully started in the background, WM_BP_END message is sent to the specified window at the end of the interpolation. The board number is passed to the first argument received in WM_BP_END message received function and finishing status is passed to the second argument.
If the interpolation has been successfully completed, the finishing status is BP_END.
If an error occurred during the interpolation process, the following Error code (errors after starting the interpolation) returns.

■ Normal end

BP_END	BP interpolation has been successfully completed.

■ Error code (errors after starting the interpolation)

BP_STOP	BP interpolation stopped during driving (too fast to stack next data).
BP_USER_STOP	The user aborted BP interpolation.
BP_DRIVE_ERR	Error occurred in the board during BP interpolation. (When the error status was set to RR0.)
BP_STOP_CERR	BP interpolation stopped during driving. (Due to RR2 error.)
BP_GETSC_ERR	Stack counter reading error

When interpolation is terminated if end code is written, error code, BP_END or BP_STOP, is returned.

Example

```
[VC]
    {
        //  2-axis BP interpolation data    BP1P,   BP1M,   BP2P,   BP2M
        DATA_2BP Data2Bp[2] = {{0x0000, 0x2BFF, 0xFFD4, 0x0000},
                               {0xF6FE, 0x0000, 0x000F, 0x3FC0}};

        Nmc_IPMode(No, IcNo,0x0003);                    // Interpolation mode setting. Assign
X and Y axis.

        // Parameter setting related to speed(The actual setting description is omitted)

        Ret = Nmc_2BPExecMC8500P_BG(hWnd, No, IcNo, Data2Bp, 2, 0x3);
            // Execute 2-axis BP interpolation. The number of data is 2, X, Y axes

        if(Ret == BP_START)   AfxMessageBox("Interpolation has started ");
            // Return value is correct. (Interpolation has started)
    }

    BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)          // WM_BP_END message
```

received function setting

```
        ON_MESSAGE( WM_BP_END, OnMsg_BP )
    END_MESSAGE_MAP()


    // WM_BP_END message received function
    afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_BP(WPARAM BoardNo, LPARAM Status)
    {
        if(Status == BP_END)    AfxMessageBox("Interpolation has been successfully completed"");
                // Return value is correct. (Interpolation has finished)
        return 0;
    }
```

[VB.NET]

```
            '2-axis BP interpolation data
            Dim Data2Bp(1) As DATA_2BP
            Data2Bp(0).Bp1p = 0
            Data2Bp(0).Bp1m = &H2BFFS
            Data2Bp(0).Bp2p = &HFFD4S
            Data2Bp(0).Bp2m = 0

            Data2Bp(1).Bp1p = &HF6FES
            Data2Bp(1).Bp1m = 0
            Data2Bp(1).Bp2p = &HFS
            Data2Bp(1).Bp2m = &H3FC0S
            Call Nmc_IPMode(No, IcNo, &H3)                  ' Interpolation mode setting. Assign X
                                                              and Y axis.

            ' Parameter setting related to speed(The actual setting description is omitted)

         ' Execute 2-axis BP interpolation. The number of data is 2, X, Y axes
            Ret = Nmc_2BPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data2Bp, 2, &H3)
            If Ret = BP_START Then      ' Return value is correct (Interpolation has started)
                Call MsgBox("Interpolation has started ")
            End If
        End Sub

        ' WM_BP_END message received function
        Protected Overrides Sub WndProc(ByRef m As Message)
            Select Case (m.Msg)
                Case WM_BP_END                              'BP completion message.
                    If m.LParam.ToInt32 = BP_END Then   ' Return value is correct. (Interpolation
                    has started)
                        Call MsgBox("Interpolation has been successfully completed")
                    End If
                    Exit Select
            End Select
            MyBase.WndProc(m)
        End Sub
```

[C#]

```
        DATA_2BP [] Data2Bp = new DATA_2BP[2];                  // 2-axis BP interpolation
data
        // Interpolation data setting
        Data2Bp[0].Bp1p = 0;                    // 0000 0000 0000 0000  BP1 +direction   0pulse
        Data2Bp[0].Bp1m = 0x2BFF;       // 0010 1011 1111 1111 BP1 -direction   12pulse
        Data2Bp[0].Bp2p = 0xFFD4;       // 1111 1111 1101 0100 BP2 +direction   12pulse
        Data2Bp[0].Bp2m = 0;          // 0000 0000 0000 0000    BP2 -direction    0pulse

        Data2Bp[1].Bp1p = 0xF6FE;     // 1111 0110 1111 1110    BP1 +direction   13pulse
        Data2Bp[1].Bp1m = 0;            // 0000 0000 0000 0000          BP1 -direction   0pulse
        Data2Bp[1].Bp2p = 0xF;          // 0000 0000 0000 1111          BP2 +direction   4pulse
```

```
Data2Bp[1].Bp2m = 0x3FC0;       // 0011 1111 1100 0000          BP2 -direction   8pulse

MC8000P.Nmc_IPMode(No, IcNo, 0x0003);                    // Interpolation mode setting.
                                                          Assign X and Y axis.

// Parameter setting related to speed(The actual setting description is omitted)

// Execute 2-axis BP interpolation in the background.
Ret = MC8000P.Nmc_2BPExecMC8500P_BG((System.IntPtr)parent.Handle, No, IcNo,
Data2Bp,2,0x3);
if(Ret == Nmc_Status.BP_START)                    // Interpolation has successfully
                                                  started.
// WM_BP_END message received function
protected override void WndProc(ref Message m)
{
  // C Call original WndProc (call a constructor explicitly)
  base.WndProc ( ref m );
  if ( m.Msg == (int)MSG_ID.WM_BP_END )
  {
   if((uint)m.LParam == Nmc_Status.BP_END)          // BP interpolation has been
                                                    successfully completed.
  }
}
```

Note

Make sure to set interpolation axis and so on in interpolation mode setting (2A)H before executing this function. Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis.

This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content
Nmc_3BPExecMC8500P_BG	Execute 3-axis bit pattern interpolation in the background using the specified interpolation data. This function returns control right after the interpolation process started and executes the interpolation in the background. WM_BP_END message is sent to the specified window at the end of the interpolation and finishing status is passed. **VC** DWORD Nmc_3BPExecMC8500P_BG(HWND User_hWnd, int No, int IcNo, DATA_3BP* pData3Bp, int DataCnt, int IpAxis); **VB.NET** Function Nmc_3BPExecMC8500P_BG(ByVal User_hWnd As Integer, ByVal No As Integer, ByVal IcNo As Integer, ByVal pData3Bp As DATA_3BP(), ByVal DataCnt As Integer, ByVal IpAxis As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_3BPExec_BG(System.IntPtr User_hWnd, int No, int IcNo, DATA_3BP[] pData3Bp, int DataCnt, int IpAxis,); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. pData3Bp Pointer to an array of DATA_3BP structures (user-defined type in VB). Set the 3-axis BP interpolation data to DATA_3BP. See [5.1.3] (3) for DATA_3BP. DataCnt The number of 2-axis BP interpolation data. Specify the number of the DATA_3BP structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4). **Return Value** If the interpolation process has been successfully started in the background, the return value is BP_START. If an error occurred before starting the interpolation process, the return value is the following Error code (errors before starting the interpolation). For C#, See [5.1.3] (1). ■ Normal start BP_START BP interpolation has been successfully started in the background. ■ Error code (errors before starting the interpolation) BP_CNT_ERR The number of the specified data is out of range. BP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. BP_THREAD_ERR Thread cannot be started. BP_MALLOC_ERR Memory cannot be allocated. BP_PARAM_ERR The parameter is incorrect. BP_NOT_OPEN_ERR The specified board is not opened. BP_OTHER_ERR Other errors. BP_FUNC_ERR Unusable function is used. After the interpolation process has been successfully started in the background, WM_BP_END message is sent to the specified window at the end of the interpolation. The board number is passed to the first argument received in WM_BP_END message received function and finishing status is passed to the second argument. If the interpolation has been successfully completed, the finishing status is BP_END. If an error occurred during the interpolation process, the following Error code (errors after starting the interpolation) returns. ■ Normal end BP_END BP interpolation has been successfully completed. ■ Error code (errors after starting the interpolation) BP_STOP BP interpolation stopped during driving (too fast to stack next data). BP_USER_STOP The user aborted BP interpolation. BP_DRIVE_ERR Error occurred in the board during BP interpolation. (When the error status was set to RR0.)

BP_STOP_CERR BP interpolation stopped during driving. (Due to RR2 error.)
BP_GETSC_ERR Stack counter reading error

When interpolation is terminated if end code is written, error code, BP_END or BP_STOP, is returned

Example

[VC]

```
{
  //  3-axis BP interpolation data   BP1P,   BP1M,   BP2P,   BP2M,   BP3P,   BP3M
  DATA_3BP Data3Bp[2] = {{0xFF30, 0x0000, 0x0000, 0x84FF, 0x0000, 0xAC35},
                         {0xAC35, 0x0000, 0xC000, 0x36E7, 0xC000, 0x3F3F}};

  Nmc_IPMode(No, IcNo,0x0007);                    // Interpolation mode setting. Assign X, Y and Z axis.

  // Parameter setting related to speed(The actual setting description is omitted)

  Ret = Nmc_3BPExecMC8500P_BG(hWnd, No, IcNo, Data3Bp, 2, 0x7);
  // Execute 3-axis BP interpolation. The number of data is 2, X, Y,Z axes
  if(Ret == BP_START)   AfxMessageBox("Interpolation has started ");
  // Return value is correct. (Interpolation has started)
}

BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)
 // WM_BP_END message received function setting
    ON_MESSAGE( WM_BP_END, OnMsg_BP )
END_MESSAGE_MAP()

// WM_BP_END message received function
afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_BP(WPARAM BoardNo, LPARAM Status)
{
    if(Status == BP_END)   AfxMessageBox("Interpolation has been successfully completed");
    // Return value is correct. (Interpolation has finished)
      return 0;
}
```

[VB.NET]

```
'3-axis BP interpolation data
Dim Data3Bp (1) As DATA_3BP
Data3Bp(0).Bp1p = &HFF30S
Data3Bp(0).Bp1m = 0
Data3Bp(0).Bp2p = 0
Data3Bp(0).Bp2m = &H84FFS
Data3Bp(0).Bp3p = 0
Data3Bp(0).Bp3m = &HAC35S

Data3Bp(1).Bp1p = &H0xAC35S
Data3Bp(1).Bp1m = 0
Data3Bp(1).Bp2p = &HC000S
Data3Bp(1).Bp2m = &H36E7S
Data3Bp(1).Bp3p = &HC000S
Data3Bp(1).Bp3m = &H3F3FS

Call Nmc_IPMode(No, IcNo, &H7)                  ' Interpolation mode setting. Assign X, Y and Z axis

' Parameter setting related to speed(The actual setting description is omitted)

' Execute 3-axis BP interpolation. The number of data is 2, X, Y, Z axes
Ret = Nmc_3BPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data3Bp, 2, &H7)
```

```
            If Ret = BP_START Then      ' Return value is correct (Interpolation has started)
                Call MsgBox("Interpolation has started ")
            End If
        End Sub

        ' WM_BP_END message received function
        Protected Overrides Sub WndProc(ByRef m As Message)
            Select Case (m.Msg)
                Case WM_BP_END                              ' BP completion message
                    If m.LParam.ToInt32 = BP_END Then       ' Return value is correct. (Interpolation
                    has started)
                        Call MsgBox("Interpolation has been successfully completed ")
                    End If
                    Exit Select
            End Select
            MyBase.WndProc(m)
        End Sub
```

[C#]

```
        DATA_3BP [] Data3Bp = new DATA_3BP[2];                  // 3-axis BP interpolation data
        // Interpolation data setting
        Data3Bp[0].Bp1p = 0xFF30;       // 1111 1111 0011 0000      BP1 +direction   10pulse
        Data3Bp[0].Bp1m = 0;            // 0000 0000 0000 0000      BP1 -direction    0pulse
        Data3Bp[0].Bp2p = 0;            // 0000 0000 0000 0000      BP2 +direction    0pulse
        Data3Bp[0].Bp2m = 0x84FF;       // 1000 0100 1111 1111      BP2 -direction   10pulse
        Data3Bp[0].Bp3p = 0;            // 0000 0000 0000 0000      BP3 +direction    0pulse
        Data3Bp[0].Bp3m = 0xAC35;       // 1010 1100 0011 0101      BP3 -direction    8pulse

        Data3Bp[1].Bp1p = 0xAC35;       // 1010 1100 0011 0101      BP1 +direction    8pulse
        Data3Bp[1].Bp1m = 0;            // 0000 0000 0000 0000      BP1 -direction    0pulse
        Data3Bp[1].Bp2p = 0xC000;       // 1100 0000 0000 0000      BP2 +direction    2pulse
        Data3Bp[1].Bp2m = 0x36E7;       // 0011 0110 1110 0111      BP2 -direction   10pulse
        Data3Bp[1].Bp3p = 0xC000;       // 1100 0000 0000 0000      BP3 +direction    2pulse
        Data3Bp[1].Bp3m = 0x3F3F;       // 0011 1111 0011 1111      BP3 -direction   12pulse
          MC8000P.Nmc_IPMode(No, IcNo,0x0007);
          // Interpolation mode setting. Assign X, Y and Z axis
      // Parameter setting related to speed(The actual setting description is omitted)
      // Execute 3-axis BP interpolation in the background.
      Ret = MC8000P.Nmc_3BPExecMC8500P_BG((System.IntPtr)parent.Handle, No, IcNo,
Data3Bp,2,0x7);

      if(Ret == Nmc_Status.BP_START)                            // Interpolation has successfully
started.

       // WM_BP_END message received function
       protected override void WndProc(ref Message m)
       {
         // C Call original WndProc (call a constructor explicitly)
         base.WndProc ( ref m );
         if ( m.Msg == (int)MSG_ID.WM_BP_END )
         {
         if((uint)m.LParam == Nmc_Status.BP_END)            // BP interpolation has been successfull
completed.
          }
       }

      if(Ret == Nmc_Status.BP_END)                                  // Return value is correct
```

Note

Make sure to set interpolation axis and so on in interpolation mode setting (2A)H before executing thi function. Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis.

This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content
Nmc_4BPExecMC8500P_BG	Execute 4-axis bit pattern interpolation in the background using the specified interpolation data. This function returns control right after the interpolation process started and executes the interpolation i the background. WM_BP_END message is sent to the specified window at the end of the interpolation and finishing status is passed. **VC** DWORD Nmc_4BPExecMC8500P_BG(HWND User_hWnd, int No, int IcNo, DATA_4BP* pData4Bp, int DataCnt, int IpAxis); **VB.NET** Function Nmc_4BPExecMC8500P_BG(ByVal User_hWnd As Integer, ByVal No As Integer ByVal IcNo As Integer, ByVal pData4Bp As DATA_4BP(), ByVal DataCnt As Integer, ByVal IpAxis As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_4BPExec_BG(System.IntPtr User_hWnd, int No, int IcNo, DATA_4BP[] pData4Bp, int DataCnt, int IpAxis,); **Input parameter** User_hWnd Window handle of user application. No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. pData4Bp Pointer to an array of DATA_4BP structures (user-defined type in VB). Set the 4-axis BP interpolation data to DATA_4BP. See [5.1.3] (3) for DATA_4BP. DataCnt The number of 2-axis BP interpolation data. Specify the number of the DATA_4BP structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4). **Return Value** If the interpolation process has been successfully started in the background, the return value is BP_START. If an error occurred before starting the interpolation process, the return value is the following Error code (errors before starting the interpolation).For C#, See [5.1.3] (1). ■ Normal start BP_START BP interpolation has been successfully started in the background ■ Error code (errors before starting the interpolation) BP_CNT_ERR The number of the specified data is out of range. BP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. BP_THREAD_ERR Thread cannot be started. BP_MALLOC_ERR Memory cannot be allocated. BP_PARAM_ERR The parameter is incorrect. BP_NOT_OPEN_ERR The specified board is not opened. BP_OTHER_ERR Other errors. BP_FUNC_ERR Unusable function is used After the interpolation process has been successfully started in the background, WM_BP_END message is sent to the specified window at the end of the interpolation. The board number is passed to the first argument received in WM_BP_END message received function and finishing status is passed to the second argument. If the interpolation has been successfully completed, the finishing status is BP_END. If an error occurred during the interpolation process, the following Error code (errors after starting the interpolation) returns. ■ Normal end BP_END BP interpolation has been successfully completed. ■ Error code (errors after starting the interpolation) BP_STOP BP interpolation stopped during driving (too fast to stack next data).

BP_USER_STOP The user aborted BP interpolation.
BP_DRIVE_ERR Error occurred in the board during BP interpolation. (When the error status was set to RR0.)
BP_STOP_CERR BP interpolation stopped during driving. (Due to RR2 error.)
BP_GETSC_ERR Stack counter reading error

When interpolation is terminated if end code is written, error code, BP_END or BP_STOP, is returned.

Example

[VC]

```
{
 // 4-axis BP interpolation data BP1P, BP1M, BP2P, BP2M, BP3P, BP3M, BP4P, BP4M
  DATA_4BP Data4Bp[2] = {{0xFFE4, 0x0000, 0x03FF, 0x4000, 0x0000, 0xFFFF, 0xFE80, 0x000F},
                         {0x0000, 0x03FF, 0xFFD0, 0x0000, 0x4AAB, 0x0000, 0x1FFF, 0x0000}};

  Nmc_IPMode(No, IcNo,0x000F);                    // Interpolation mode setting. Assign X, Y Z and U axis.
  // Parameter setting related to speed(The actual setting description is omitted)

  Ret = Nmc_4BPExecMC8500P_BG(hWnd, No, IcNo, Data4Bp, 2, 0xF);
 // Execute 4-axis BP interpolation. The number of data is 2, X, Y,Z, U axes

  if(Ret == BP_START)  AfxMessageBox("Interpolation has started "); // Return value is correct. (Interpolation has started)
}

 BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)  // WM_BP_END message received function setting
     ON_MESSAGE( WM_BP_END, OnMsg_BP )
 END_MESSAGE_MAP()

 // WM_BP_END message received function
 afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_BP(WPARAM BoardNo, LPARAM Status)
 {
     if(Status == BP_END)  AfxMessageBox("Interpolation has been successfully completed
                                                                                    ";
    // Return value is correct. (Interpolation has finished)
    return 0;
 }
```

[VB.NET]

```
'4-axis BP interpolation data
Dim Data4Bp(1) As DATA_4BP
Data4Bp(0).Bp1p = &HFFE4S
Data4Bp(0).Bp1m = 0
Data4Bp(0).Bp2p = &H3FFS
Data4Bp(0).Bp2m = &H4000S
Data4Bp(0).Bp3p = 0
Data4Bp(0).Bp3m = &HFFFFS
Data4Bp(0).Bp4p = &HFE80S
Data4Bp(0).Bp4m = &H000FS

Data4Bp(1).Bp1p = 0
Data4Bp(1).Bp1m = &H3FFS
```

```
        Data4Bp(1).Bp2p = &HFFD0S
        Data4Bp(1).Bp2m = 0
        Data4Bp(1).Bp3p = &H4AABS
        Data4Bp(1).Bp3m = 0
        Data4Bp(1).Bp4p = &H1FFFS
        Data4Bp(1).Bp4m = 0
        Call Nmc_IPMode(No, IcNo, &HF)                ' Interpolation mode setting. Assign
        X, Y, Z and U axis

        ' Parameter setting related to speed(The actual setting description is omitted)

        'Execute 4-axis BP interpolation. The number of data is 2, X, Y, Z, U axes
        Ret = Nmc_4BPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data4Bp, 2, &HF)
        If Ret = BP_START Then      ' Return value is correct (Interpolation has started)
            Call MsgBox("Interpolation has started ")
        End If
    End Sub

    ' WM_BP_END message received function
    Protected Overrides Sub WndProc(ByRef m As Message)
        Select Case (m.Msg)
            Case WM_BP_END                                ' BP completion message
                If m.LParam.ToInt32 = BP_END Then      ' Return value is correct.
                (Interpolation has started)
                    Call MsgBox("Interpolation has been successfully completed ")
                End If
                Exit Select
        End Select
        MyBase.WndProc(m)
    End Sub
```

[C#]

```
DATA_4BP [] Data4Bp = new DATA_4BP[2];            // 4-axis BP interpolation data
// Interpolation data setting
Data4Bp[0].Bp1p = 0xFFE4;        // 1111 1111 1110 0100          BP1 +direction
12pulse
Data4Bp[0].Bp1m = 0;             // 0000 0000 0000 0000          BP1 -direction
0pulse          Data4Bp[0].Bp2p = 0x03FF;  // 0000 0011 1111 1111BP2 +direction
10pulse
Data4Bp[0].Bp2m = 0x4000;        // 0100 0000 0000 0000          BP2 -direction
1pulse
Data4Bp[0].Bp3p = 0;             // 0000 0000 0000 0000          BP3 +direction
0pulse
Data4Bp[0].Bp3m = 0xFFFF;        // 1111 1111 1111 1111  BP3 -direction      16pulse
Data4Bp[0].Bp4p = 0xFE80;        // 1111 1110 1000 0000          BP3 +direction
8pulse
Data4Bp[0].Bp4m = 0x000F;        // 0000 0000 0000 1111          BP3 -direction
4pulse

Data4Bp[1].Bp1p = 0;             // 0000 0000 0000 0000          BP1 +direction
8pulse
Data4Bp[1].Bp1m = 0x03FF;        // 0000 0011 1111 1111          BP1 -direction
10pulse
Data4Bp[1].Bp2p = 0xFFD0;        // 1111 1111 1101 0000           BP2 +direction
11pulse
Data4Bp[1].Bp2m = 0;             // 0000 0000 0000 0000          BP2 -direction
0pulse
Data4Bp[1].Bp3p = 0x4AAB;        // 0100 1010 1010 1011          BP3 +direction
8pulse
Data4Bp[1].Bp3m = 0;             // 0000 0000 0000 0000           BP3 -direction
0pulse
Data4Bp[1].Bp4p = 0x1FFF;        // 0001 1111 1111 1111          BP3 +direction
13pulse
Data4Bp[1].Bp4m = 0;            // 0000 0000 0000 0000          BP3 -direction
0pulse

  MC8000P.Nmc_IPMode(No, IcNo, 0x000F);
  // Interpolation mode setting. Assign X, Y Z and U axis
```

```
// Parameter setting related to speed(The actual setting description is omitted)

// Execute 4-axis BP interpolation in the background.
Ret = MC8000P.Nmc_4BPExecMC8500P_BG((System.IntPtr)parent.Handle,No,
IcNo,Data3Bp,2,0xF);

if(Ret == Nmc_Status.BP_START)                                    // Interpolation has
successfully started.

// WM_BP_END message received function
protected override void WndProc(ref Message m)
{
   // C Call original WndProc (call a constructor explicitly)
   base.WndProc ( ref m );
   if ( m.Msg == (int)MSG_ID.WM_BP_END )
   {
   if((uint)m.LParam == Nmc_Status.BP_END)          // BP interpolation has been
                                                    successfully completed.
    }
  }

if(Ret == Nmc_Status.BP_END)                                      // Return value is correct
```

Note

Make sure to set interpolation axis and so on in interpolation mode setting (2A)H before executing this function. Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis.

This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content
Nmc_2CIPExecMC8500P	Execute 2-axis continuous interpolation using the specified interpolation data. This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread. **VC** DWORD Nmc_2CIPExecMC8500P(int No, int IcNo, DATA_2CIP_MC8500P* pData2Cip, int DataCnt, int IpAxis, BOOL SpdChgFlg = FALSE); **VB.NET** Function Nmc_2CIPExecMC8500P(ByVal No As Integer, ByVal IcNo As Integer, ByVal pData2Cip As DATA_2CIP_MC8500P(), ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal SpdChgFlg As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_2CIPExecMC8500P(int No, int IcNo, DATA_2CIP_MC8500P[] pData2Cip, int DataCnt,int IpAxis, bool SpdChgFlg); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. pData2Cip Pointer to an array (DATA_2CIP_MC8500P address) of DATA_2CIP_MC8500P structures (user-defined type). Set the 2-axis continuous interpolation data to DATA_2CIP_MC8500P.See [5.1.3] (3) for DATA_2CIP_MC8500P. DataCnt The number of 2-axis continuous interpolation data. Specify the number of the DATA_2CIP_MC8500P structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4). SpdChgFlg Set the flag to change the speed during the interpolation process. If you change the speed, See [5.1.3] (5). [VC] TRUE: Change, FALSE: Not change Can be omitted. Default is FALSE. [VB.NET] True: Change, False: Not change [C#] true: Change, false: Not change **When selecting Change** : Refers to the setting value of DATA_2CIP_MC8500P Speed. Set 1～4000000 to Speed・・・Changes to the setting speed. Set 0 to Speed・・・・・・ Not change the speed. **When selecting Not change:** Not refers to the setting value of DATA_2CIP_MC8500 Speed. **Return Value** If the function succeeds, the return value is CIP_END. If the function fails, the return value is the following Error code. For C#, See [5.1.3] (1). ■ Normal end CIP_END Continuous interpolation has been successfully completed. ■ Error code CIP_CNT_ERR The number of the specified data is out of range. CIP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. CIP_MALLOC_ERR Memory cannot be allocated. CIP_CMD_ERR The wrong command was specified. CIP_PARAM_ERR The parameter is incorrect. CIP_NOT_OPEN_ER The specified board is not opened. CIP_OTHER_ERR Other errors. CIP_FUNC_ERR Unusable function is used. CIP_STOP Continuous interpolation stopped during driving CIP_USER_STOP The user aborted continuous interpolation. CIP_DRIVE_ERR Error occurred in the board during continuous interpolation. (When the error status was set to RR0.) CIP_STOP_CERR Continuous interpolation stopped during driving. (Due to RR2 error.) CIP_GETSC_ERR Stack counter reading error

Note

Set 4,000,000 to initial speed value before executing this function. (Don't change initial speed during executing this function). See [5.1.4] for more details.

Example

[VC]

```
// 2-axis continuous interpolation data    Command, Speed, Finishing point 1, Finishing point 2, Center point 1, Center point 2
    DATA_2CIP_MC8500P Data2Cip[2]={{MC8500P_CMD_2CIP,  0,  4500,  0,  0,  0},
    //2-axis linear interpolation
                                   {MC8500P_CMD_CIPCCW, 0,  1500,  1500,  0,  1500}};
                                   // CCW circular interpolation

    Nmc_IPMode(No, IcNo, 0x0003);                          //Interpolation mode setting. Assign X Y axes.

  // Parameter setting related to speed
    Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);               // Set 4M to execute constant speed drive
                                                           // The other setting description is omitted
Ret = Nmc_2CIPExecMC8500P(No, IcNo, Data2Cip, 2, 0x3);
// Execute 2-axis continuous interpolation. The number of data is 2, X, Y axes


    if(Ret == CIP_END)  AfxMessageBox("Interpolation has finished");   // Return value is correct.
```

[VB.NET]

```
        '2-axis continuous interpolation data
        Dim Data2Cip(1) As DATA_2CIP_MC8500P
        Data2Cip(0).Command = MC8500P_CMD_2CIP   '2-axis linear interpolation
        Data2Cip(0).Speed = 0
        Data2Cip(0).EndP1 = 4500
        Data2Cip(0).EndP2 = 0
        Data2Cip(0).Center1 = 0
        Data2Cip(0).Center2 = 0

        Data2Cip(1).Command = MC8500P_CMD_CIPCCW 'CCW circular interpolation
        Data2Cip(1).Speed = 0
        Data2Cip(1).EndP1 = 1500
        Data2Cip(1).EndP2 = 1500
        Data2Cip(1).Center1 = 0
        Data2Cip(1).Center2 = 1500

        Call Nmc_IPMode(No, IcNo, &H3)          ' Interpolation mode setting. Assign X and Y axis

       ' Parameter setting related to speed
       Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)   ' Set 4M to execute constant speed drive
                                                       'The other setting description is omitted.
       'Execute 2-axis continuous interpolation. The number of data is 2, X, Y axes.

        Ret = Nmc_2CIPExecMC8500P(No, IcNo, Data2Cip, 2, &H3, False)

        If Ret = CIP_END Then      ' Return value is correct.
            Call MsgBox("Interpolation has finished")
        End If
```

[C#]

```
    DATA_2CIP_MC8500P [] Data2Cip = new DATA_2CIP_MC8500P[2];
```

	// 2-axis continuous interpolation data
	// Interpolation data setting.
	Data2Cip[0].Command = (ushort)CMD.MC8500P_CMD_2CIP; // 2-axis linear interpolation
	Data2Cip[0].EndP1 = 0;
	Data2Cip[0].EndP2 = 4500;
	Data2Cip[0].Center1 = 0;
	Data2Cip[0].Center2 = 0;
	Data2Cip[0].Speed = 0;
	
	Data2Cip[1].Command = (ushort)CMD.MC8500P_CMD_CIPCCW; // CCW circular interpolation
	Data2Cip[1].EndP1 = 1500;
	Data2Cip[1].EndP2 = 1500;
	Data2Cip[1].Center1 = 0;
	Data2Cip[1].Center2 = 1500;
	Data2Cip[1].Speed = 0;
	
	MC8000P.Nmc_IPMode(No, IcNo, 0x0003) // Interpolation mode setting. Assign X and Y axis.
	
	// Execute 2-axis continuous interpolation.
	// This function will not return control unless the interpolation process ends.
	Ret = MC8000P.Nmc_2CIPExecMC8500P(No, IcNo, Data2Cip, 2, 0x3, SpdChgFlg);
	if(Ret == Nmc_Status.CIP_END) // Continuous interpolation has been successfully completed
	
	Note
	Make sure to set interpolation axis in interpolation mode setting (2A)H before executing this function.
	Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis.
	This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content
Nmc_3CIPExecMC8500P	Execute 3-axis continuous interpolation using the specified interpolation data. This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread. **VC** DWORD Nmc_3CIPExecMC8500P(int No, int IcNo, DATA_3CIP_MC8500P* pData3Cip, int DataCnt, int IpAxis, BOOL SpdChgFlg = FALSE); **VB.NET** Function Nmc_3CIPExecMC8500P(ByVal No As Integer, ByVal IcNo As Integer, ByVal pData3Cip As DATA_3CIP_MC8500P(), ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal SpdChgFlg As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_3CIPExecMC8500P(int No, IcNo, DATA_3CIP_MC8500P[] pData3Cip, int DataCnt,int IpAxis, bool SpdChgFlg) **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. pData3Cip Pointer to an array (DATA_3CIP_MC8500P address) of DATA_3CIP_MC8500P structures (user-defined type). Set the 3-axis continuous interpolation data to DATA_3CIP_MC8500P.See [5.1.3] (3) for DATA_3CIP_MC8500P. DataCnt The number of 3-axis continuous interpolation data. Specify the number of the DATA_3CIP_MC8500P structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4). SpdChgFlg Set the flag to change the speed during the interpolation process. If you change the speed, See [5.1.3] (5). [VC] TRUE: Change, FALSE: Not change Can be omitted. Default is FALSE. [VB.NET] True: Change, False: Not change [C#] true: Change, false: Not change **When selecting Change** : Refers to the setting value of DATA_.3CIP_MC8500P Speed. Set 1～4000000 to Speed・・・Changes to the setting speed Set 0 to Speed・・・・・・ Not change the speed. **When selecting Not change:** Not refers to the setting value of DATA_3CIP_MC8500 Speed. **Return Value** If the function succeeds, the return value is CIP_END. If the function fails, the return value is the following Error code. For C#, See [5.1.3] (1). ■ Normal end CIP_END Continuous interpolation has been successfully completed. ■ Error code CIP_CNT_ERR The number of the specified data is out of range. CIP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. CIP_MALLOC_ERR Memory cannot be allocated. CIP_CMD_ERR The wrong command was specified. CIP_PARAM_ERR The parameter is incorrect. CIP_NOT_OPEN_ER The specified board is not opened. CIP_OTHER_ERR Other errors. CIP_FUNC_ERR Unusable function is used. CIP_STOP Continuous interpolation stopped during driving CIP_USER_STOP The user aborted continuous interpolation. CIP_DRIVE_ERR Error occurred in the board during continuous interpolation. (When the error status was set to RR0.)

Note

Set 4,000,000 to initial speed value before executing this function. (Don't change initial speed during executing this function). See [5.1.4] for more details.

Example

[VC]

```
DATA_3CIP_MC8500P Data3Cip[2];                    // 3-axis continuous interpolation data
// 3-axis continuous interpolation data setting
Data3Cip[0].EndP1 = 1000;
Data3Cip[0].EndP2 = 2000;
Data3Cip[0].EndP3 = 3000;
Data3Cip[0].Speed = 0;

Data3Cip[1].EndP1 = 2000;
Data3Cip[1].EndP2 = -1000;
Data3Cip[1].EndP3 = 3000;
Data3Cip[1].Speed = 0;

Nmc_IPMode(No, IcNo,0x0007);                      // Interpolation mode setting. Assign X, Y, Z axes.

// Parameter setting related to speed
Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);          // Set 4M to execute constant speed drive
                                                                                  // The other setting description is omitted.

Ret = Nmc_3CIPExecMC8500P(No, IcNo, Data3Cip, 2, 0x7);
// Execute 3-axis continuous interpolation. The number of data is 2, X, Y, Z axes
if(Ret == CIP_END)     AfxMessageBox("Interpolation has finished."); // Return value is correct
```

[VB.NET]

```
'3-axis continuous interpolation data setting
Dim Data3Cip(1) As DATA_3CIP_MC8500P
Data3Cip(0).EndP1 = 1000
Data3Cip(0).EndP2 = 2000
Data3Cip(0).EndP3 = 3000
Data3Cip(0).Speed = 0

Data3Cip(1).EndP1 = 2000
Data3Cip(1).EndP2 = -1000
Data3Cip(1).EndP3 = 3000
Data3Cip(1).Speed = 0

Call Nmc_IPMode(No, IcNo, &H7)                    ' Interpolation mode setting. Assign X, Y, Z axes.

' Parameter setting related to speed
Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)      ' Set 4M to execute constant speed drive
                                                  'The other setting description is omitted.
'Execute 3-axis continuous interpolation. The number of data is 2, X, Y, Z axes

Ret = Nmc_3CIPExecMC8500P(No, IcNo, Data3Cip, 2, &H7, False)

If Ret = CIP_END Then       ' Return value is correct.
    Call MsgBox("Interpolation has finished ")
End If
```

[C#]

```
DATA_3CIP_MC8500P [] Data3Cip = new DATA_3CIP_MC8500P[2];    // 3-axis continuous interpolation
// Interpolation mode setting
Data3Cip[0].EndP1 = 1000;
Data3Cip[0].EndP2 = 2000;
Data3Cip[0].EndP3 = 3000;
Data3Cip[0].Speed = 0;

Data3Cip[1].EndP1 = 2000;
Data3Cip[1].EndP2 = -1000;
Data3Cip[1].EndP3 = 3000;
Data3Cip[1].Speed = 0;

MC8000P.Nmc_IPMode(No, IcNo, 0x0007)                    // Interpolation mode setting. Assign X, Y and Z axis

// Execute 3-axis continuous interpolation.
// This function will not return control unless the interpolation process ends.
Ret = MC8000P.Nmc_3CIPExecMC8500P(No, IcNo, Data3Cip, 2, 0x7, SpdChgFlg);

if(Ret == Nmc_Status.CIP_END)                    // Continuous interpolation has been
                                                 successfully completed
```

Note

Make sure to set interpolation axis in interpolation mode setting (2A)H before executing this function. Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis.

This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content
Nmc_4CIPExecMC8500P	Execute 4-axis continuous interpolation using the specified interpolation data. This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread. **VC** DWORD Nmc_4CIPExecMC8500P(int No, int IcNo, DATA_4CIP_MC8500P* pData4Cip, int DataCnt, int IpAxis, BOOL SpdChgFlg = FALSE); **VB.NET** Function Nmc_4CIPExecMC8500P(ByVal No As Integer, ByVal IcNo As Integer, ByVal pData4Cip As DATA_4CIP_MC8500P(), ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal SpdChgFlg As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_4CIPExecMC8500P(int No, IcNo, DATA_4CIP_MC8500P[] pData4Cip, int DataCnt,int IpAxis, bool SpdChgFlg); **Input Parameter** No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. pData4Cip Pointer to an array (DATA_4CIP_MC8500P address) of DATA_4CIP_MC8500P structures (user-defined type). Set the 4-axis continuous interpolation data to DATA_4CIP_MC8500P.See [5.1.3] (3) for DATA_4CIP_MC8500P. DataCnt The number of 4-axis continuous interpolation data. Specify the number of the DATA_4CIP_MC8500P structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4). SpdChgFlg Set the flag to change the speed during the interpolation process. If you change the speed, See [5.1.3] (5). [VC] TRUE: Change, FALSE: Not change Can be omitted. Default is FALSE. [VB.NET] True: Change, False: Not change [C#] true: Change, false: Not change **When selecting Change** : Refers to the setting value of DATA_4CIP_MC8500P Speed. Set 1～4000000 to Speed・・・Changes to the setting speed. Set 0 to Speed・・・・・・ Not change the speed. **When selecting Not change:** Not refers to the setting value of DATA_4CIP_MC8500 Speed. **Return Value** If the function succeeds, the return value is CIP_END. If the function fails, the return value is the following Error code. For C#, See [5.1.3] (1). ■ Normal end CIP_END Continuous interpolation has been successfully completed. ■ Error code CIP_CNT_ERR The number of the specified data is out of range. CIP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. CIP_MALLOC_ERR Memory cannot be allocated. CIP_CMD_ERR The wrong command was specified. CIP_PARAM_ERR The parameter is incorrect. CIP_NOT_OPEN_ER The specified board is not opened. CIP_OTHER_ERR Other errors. CIP_FUNC_ERR Unusable function is used. CIP_STOP Continuous interpolation stopped during driving CIP_USER_STOP The user aborted continuous interpolation. CIP_DRIVE_ERR Error occurred in the board during continuous interpolation. (When the error status was set to RR0.) CIP_STOP_CERR Continuous interpolation stopped during driving. (Due to RR2 error.)

CIP_GETSC_ERR Stack counter reading error

Note

Set 4,000,000 to initial speed value before executing this function. (Don't change initial spee during executing this function). See [5.1.4] for more details.

Example

[VC]

```
DATA_4CIP_MC8500P Data4Cip[2];                    // 4-axis continuous interpolation data
// 4-axis continuous interpolation data setting
Data4Cip[0].EndP1 = 1000;
Data4Cip[0].EndP2 = 2000;
Data4Cip[0].EndP3 = 3000;
Data4Cip[0].EndP4 = 4000;
Data4Cip[0].Speed = 0;

Data4Cip[1].EndP1 = 2000;
Data4Cip[1].EndP2 = -1000;
Data4Cip[1].EndP3 = 3000;
Data4Cip[1].EndP4 = -2000;
Data4Cip[1].Speed = 0;

Nmc_IPMode(No, IcNo,0x000F);                      //interpolation mode setting Assign X, Y, Z, U axes

// Parameter setting related to speed
Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);  // Set 4M to execute constant speed drive
                                          // The other setting description is omitted.
Ret = Nmc_4CIPExecMC8500P(No, IcNo, Data4Cip, 2, 0xF);
//Execute 4-axis continuous interpolation. The number of data is 2, X, Y, Z, U axes

if(Ret == CIP_END)     AfxMessageBox("Interpolation has finished ");    // Return value is correct.
```

[VB.NET]

```
'4-axis continuous interpolation data setting
Dim Data4Cip(1) As DATA_4CIP_MC8500P
Data4Cip(0).EndP1 = 1000
Data4Cip(0).EndP2 = 2000
Data4Cip(0).EndP3 = 3000
Data4Cip(0).EndP4 = 4000
Data4Cip(0).Speed = 0

Data4Cip(1).EndP1 = 2000
Data4Cip(1).EndP2 = -1000
Data4Cip(1).EndP3 = 3000
Data4Cip(1).EndP4 = -2000
Data4Cip(1).Speed = 0

Call Nmc_IPMode(No, IcNo, &HF)                    ' Interpolation mode setting. Assig
X, Y, Z, U axes.

' Parameter setting related to speed
Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)              ' Set 4M to execute constant speed drive
                                                  'The other setting description is omitted.
'Execute 4-axis continuous interpolation. The number of data is 2. X, Y, X. U axes
Ret = Nmc_4CIPExecMC8500P(No, IcNo, Data4Cip, 2, &HF, False)

If Ret = CIP_END Then       ' Return value is correct.
```

```
                Call MsgBox("Interpolation has finished ")
            End If


    [C#]
        DATA_4CIP_MC8500P [] Data4Cip = new DATA_4CIP_MC8500P[2];    // 4-axis continuous
interpolation
        // Interpolation mode setting
        Data4Cip[0].EndP1 = 1000;
        Data4Cip[0].EndP2 = 2000;
        Data4Cip[0].EndP3 = 3000;
        Data4Cip[0].EndP4 = 3000;
        Data4Cip[0].Speed = 0;

        Data4Cip[1].EndP1 = 2000;
        Data4Cip[1].EndP2 = -1000;
        Data4Cip[1].EndP3 = 3000;
        Data4Cip[1].EndP4 = -2000;
        Data4Cip[1].Speed = 0;

        MC8000P.Nmc_IPMode(No, IcNo, 0x000F)                    // Interpolation mode setting. Assig
X, Y, Z and U axis

        // Execute 4-axis continuous interpolation.
        // This function will not return control unless the interpolation process ends.
        Ret = MC8000P.Nmc_4CIPExecMC8500P(No, IcNo, Data4Cip, 2, 0xF, SpdChgFlg);
         if(Ret == Nmc_Status.CIP_END)                    // Continuous interpolation has been
                                                 successfully completed
```

Note

Make sure to set interpolation axis in interpolation mode setting (2A)H before executing this function. Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis.

This function operates in constant speed driving. Set speed parameter to constant speed drive

Function Name	Function and Content
Nmc_2CIPExecMC8500P_BG	Execute 2-axis continuous interpolation in the background using the specified interpolation data. This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread. **VC** DWORD Nmc_2CIPExecMC8500P_BG(HWND User_hWnd, int No, int IcNo, DATA_2CIP_MC8500P* pData2Cip, int DataCnt,int IpAxis BOOL SpdChgFlg = FALSE); **VB.NET** Function Nmc_2CIPExecMC8500P_BG(ByVal User_hWnd As Integer, ByVal No As Integer, ByVal IcNo As Integer, ByVal pData2Cip As DATA_2CIP_MC8500P(), ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal SpdChgFlg As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_2CIPExecMC8500P_BG(System.IntPtr User_hWnd, int No, int IcNo, DATA_2CIP_MC8500P[] pData2Cip, int DataCnt, int IpAxis, bool SpdChgFlg); **Input parameter** User_hWnd Window handle of the user application No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. pData2Cip Pointer to an array (DATA_2CIP_MC8500P address) of DATA_2CIP_MC8500P structures (user-defined type). Set the 2-axis continuous interpolation data to DATA_2CIP_MC8500P.See [5.1.3] (3) for DATA_2CIP_MC8500P. DataCnt The number of 2-axis continuous interpolation data. Specify the number of the DATA_2CIP_MC8500P structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4). SpdChgFlg Set the flag to change the speed during the interpolation process. If you change the speed, See [5.1.3] (5). [VC] TRUE: Change, FALSE: Not change. Can be omitted. Default is FALSE. [VB.NET] True: Change, False: Not change [C#] true: Change, false: Not change **When selecting Change** : Refers to the setting value of DATA_2CIP_MC8500P Speed. Set 1～4000000 to Speed・・・Changes to the setting speed. Set 0 to Speed・・・・・・ Not change the speed. **When selecting Not change:** Not refers to the setting value of DATA_2CIP_MC8500 Speed. **Return Value** If the interpolation process has been successfully started in the background, the return value is CIP_START. If an error occurred before starting the interpolation process, the return value is the following Error code (errors before starting the interpolation). For C#, See [5.1.3] (1). ■ Normal start CIP_START Continuous interpolation has been successfully started in the background. ■ Error code (errors before starting the interpolation) CIP_CNT_ERR The number of the specified data is out of range. CIP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. CIP_THREAD_ERR Thread cannot be started. CIP_MALLOC_ERR Memory cannot be allocated. CIP_CMD_ERR The wrong command was specified. CIP_PARAM_ERR The parameter is incorrect. CIP_NOT_OPEN_ERR The specified board is not opened. CIP_OTHER_ERR Other errors. CIP_FUNC_ERR Unusable function is used. After the interpolation process has been successfully started in the background, WM_CIP_END message is sent to the specified window at the end of the interpolation. The board number is passed to

the first argument received in WM_CIP_END message received function and finishing status is passed to the second argument.
If the interpolation has been successfully completed, the finishing status is CIP_END.
If an error occurred during the interpolation process, the following Error code (errors after starting the interpolation) returns.

■ Normal end

CIP_END	Continuous interpolation has been successfully completed.

■ Error code (Error after starting interpolation)

CIP_USER_STOP	The user aborted continuous interpolation.
CIP_DRIVE_ERR	Error occurred in the board during continuous interpolation. (When the error status was set to RR0.)
CIP_STOP_CERR	Continuous interpolation stopped during driving. (Due to RR2 error.)
CIP_GETSC_ERR	Stack counter reading error

Note

Set 4,000,000 to initial speed value before executing this function. (Don't change initial speed during executing this function). See [5.1.4] for more details.

Example

```
[VC]
 {
// 2-axis continuous interpolation data Command, Speed, Finishing point 1, Finishing point 2, Center point 1, Center point 2
    DATA_2CIP_MC8500P Data2Cip[2]={{MC8500P_CMD_2CIP,  0,  4500,  0,  0,  0},
    //2-axis linear interpolation
                                   {MC8500P_CMD_CIPCCW, 0,  1500,  1500,  0,  1500}};
                                  // CCW circular interpolation

    Nmc_IPMode(No, IcNo,0x0003);                    // Interpolation mode setting Assign X, Y axes

 // Parameter setting related to speed
    Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);        // Set 4M to execute constant speed drive
                                                    // The other setting description is omitted.
    Ret = Nmc_2CIPExecMC8500P_BG(hWnd, No, IcNo, Data2Cip, 2, 0x3);
 // Execute 2-axis continuous interpolation. The number of data is 2, X, Y axes

    if(Ret == CIP_START)  AfxMessageBox("Interpolation has started");
    // Return value is correct.(Interpolation has started)
   }

   BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)   // WM_CIP_END message received function setting.
       ON_MESSAGE( WM_CIP_END, OnMsg_CIP )
   END_MESSAGE_MAP()

   // WM_CIP_END message received function
   afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_CIP(WPARAM BoardNo, LPARAM Status)
   {
       if(Status == CIP_END)  AfxMessageBox("Interpolation has been successfully completed ");
// Return value is correct. (Interpolation has finished)
       return 0;
   }

  [VB.NET]
        '2-axis BP interpolation data setting
        Dim Data2Cip(1) As DATA_2CIP_MC8500P
```

```
                Data2Cip(0).Command = MC8500P_CMD_2CIP     '2-axis linear interpolation data
                Data2Cip(0).Speed = 0                      'Change speed
                Data2Cip(0).EndP1 = 4500                   ' Finishing point 1
                Data2Cip(0).EndP2 = 0                      ' Finishing point 2
                Data2Cip(0).Center1 = 0                    ' Center point 1
                Data2Cip(0).Center2 = 0                    ' Center point 2

                Data2Cip(1).Command = MC8500P_CMD_CIPCCW ' CCW circular interpolation
                Data2Cip(1).Speed = 0                      ' Change speed
                Data2Cip(1).EndP1 = 1500                   ' Finishing point 1
                Data2Cip(1).EndP2 = 1500                   ' Finishing point 2
                Data2Cip(1).Center1 = 0                    ' Center point 1
                Data2Cip(1).Center2 = 1500                 ' Center point 2

                Call Nmc_IPMode(No, IcNo, &H3)             ' Interpolation mode setting. Assign X and Y
                axis

               ' Parameter setting related to speed
               Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)   ' Set 4M to execute constant speed drive
                                                               'The other setting description is omitted.
                'Execute 2-axis continuous interpolation. The number of data is 2. X, Y axes
                Ret = Nmc_2CIPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data2Cip, 2, &H3, False)

                If Ret = CIP_START Then       ' Return value is correct. (Interpolation has started.)
                    Call MsgBox("Interpolation has started ")
                End If
            End Sub

            ' WM_CIP_END message received function
            Protected Overrides Sub WndProc(ByRef m As Message)
                Select Case (m.Msg)
                    Case WM_CIP_END                        'Continuous interpolation finish
                    message.
                        If m.LParam.ToInt32 = CIP_END Then     ' Return value is correct. ("Interpolation
                        has finished")
                            Call MsgBox("Interpolation has finished")
                        End If
                        Exit Select
                End Select
                MyBase.WndProc(m)
            End Sub


    [C#]
        DATA_2CIP_MC8500P [] Data2Cip = new DATA_2CIP_MC8500P[2];
        // 2-axis continuous interpolation data
        // Interpolation data setting
        Data2Cip[0].Command = (ushort)CMD.MC8500P_CMD_2CIP;                    //   2-axis linear
interpolation
        Data2Cip[0].EndP1 = 0;
        Data2Cip[0].EndP2 = 4500;
        Data2Cip[0].Center1 = 0;
        Data2Cip[0].Center2 = 0;
        Data2Cip[0].Speed = 0;

        Data2Cip[1].Command = (ushort)CMD.MC8500P_CMD_CIPCCW;                // CCW circular
interpolation
        Data2Cip[1].EndP1 = 1500;
        Data2Cip[1].EndP2 = 1500;
        Data2Cip[1].Center1 = 0;
```

```
Data2Cip[1].Center2 = 1500;
Data2Cip[1].Speed = 0;

MC8000P.Nmc_IPMode(No, IcNo, 0x3);                    // Interpolation mode setting. Assign X and Y axis.

// Execute 2-axis continuous interpolation.
// This function will not return control unless the interpolation process ends
Ret = MC8000P.Nmc_2CIPExecMC8500P_BG(No, IcNo, Data2Cip, 2, 0x3, SpdChgFlg);

// WWM_CIP_END message received function
  protected override void WndProc(ref Message m)
  {
     // C Call original WndProc (call a constructor explicitly)
     base.WndProc ( ref m );
     if ( m.Msg == (int)MSG_ID.WM_CIP_END )
     {
          if((uint)m.LParam == Nmc_Status.CIP_END)
      // Continuous interpolation has been successfully completed.          }
  }
```

Note

Make sure to set interpolation axis and so on in interpolation mode setting before executing this function. Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis.

This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content
Nmc_3CIPExecMC8500P_BG	Execute 3-axis continuous interpolation in the background using the specified interpolation data. This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread. **VC** DWORD Nmc_3CIPExecMC8500P_BG(HWND User_hWnd, int No, int IcNo, DATA_3CIP_MC8500P* pData3Cip, int DataCnt, int IpAxis, BOOL SpdChgFlg = FALSE); **VB.NET** Function Nmc_3CIPExecMC8500P_BG(ByVal User_hWnd As Integer, ByVal No As Integer, ByVal IcNo As Integer, ByVal pData3Cip As DATA_3CIP_MC8500P(), ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal SpdChgFlg As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_3CIPExecMC8500P_BG(System.IntPtr User_hWnd, int No, int IcNo, DATA_3CIP_MC8500P[] pData3Cip, int IpAxis, bool SpdChgFlg); **Input Parameter** User_hWnd Window handle of the user application No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. pData3Cip Pointer to an array (DATA_3CIP_MC8500P address) of DATA_3CIP_MC8500P structures (user-defined type). Set the 3-axis continuous interpolation data to DATA_3CIP_MC8500P.See [5.1.3] (3) for DATA_3CIP_MC8500P. DataCnt The number of 3-axis continuous interpolation data. Specify the number of the DATA_3CIP_MC8500P structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4). SpdChgFlg Set the flag to change the speed during the interpolation process. If you change the speed, See [5.1.3] (5). [VC] TRUE: Change, FALSE: Not change. Can be omitted. Default is FALSE. [VB.NET] True: Change, False: Not change [C#] true: Change, false: Not change **When selecting Change** : Refers to the setting value of DATA_.3CIP_MC8500P Speed. Set 1～4000000 to Speed・・・Changes to the setting speed. Set 0 to Speed・・・・・・ Not change the speed. **When selecting Not change:** Not refers to the setting value of DATA_3CIP_MC8500 Speed. **Return Value** If the interpolation process has been successfully started in the background, the return value is CIP_START. If an error occurred before starting the interpolation process, the return value is the following Error code (errors before starting the interpolation). For C#, See [5.1.3] (1). ■ Normal start CIP_START Continuous interpolation has been successfully started in the background. ■ Error code (errors before starting the interpolation) CIP_CNT_ERR The number of the specified data is out of range. CIP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. CIP_THREAD_ERR Thread cannot be started. CIP_MALLOC_ERR Memory cannot be allocated. CIP_CMD_ERR The wrong command was specified. CIP_PARAM_ERR The parameter is incorrect. CIP_NOT_OPEN_ERR The specified board is not opened. CIP_OTHER_ERR Other errors.

CIP_FUNC_ERR Unusable function is used.

After the interpolation process has been successfully started in the background, WM_CIP_END message is sent to the specified window at the end of the interpolation. The board number is passed to the first argument received in WM_CIP_END message received function and finishing status is passed to the second argument.

If the interpolation has been successfully completed, the finishing status is CIP_END.

If an error occurred during the interpolation process, the following Error code (errors after starting the interpolation) returns.

■ Normal end

CIP_END Continuous interpolation has been successfully completed.

■ Error code (errors after starting the interpolation)

CIP_USER_STOP The user aborted continuous interpolation.

CIP_DRIVE_ERR Error occurred in the board during continuous interpolation. (When the error status was set to RR0.)

CIP_STOP_CERR Continuous interpolation stopped during driving. (Due to RR2 error.)

CIP_GETSC_ERR Stack counter reading error

Note

Set 4,000,000 to initial speed value before executing this function. (Don't change initial speed during executing this function). See [5.1.4] for more details.

Example

[VC]

```
{
    DATA_3CIP_MC8500P Data3Cip[2];                    // 3-axis continuous interpolation data
    // 3-axis continuous interpolation data setting
    Data3Cip[0].EndP1 = 1000;
    Data3Cip[0].EndP2 = 2000;
    Data3Cip[0].EndP3 = 3000;

    Data3Cip[1].EndP1 = 2000;
    Data3Cip[1].EndP2 = -1000;
    Data3Cip[1].EndP3 = 3000;

    Nmc_IPMode(No, IcNo,0x0007);                      // Interpolation mode setting. Assign X, Y, Z axes.

 // Parameter setting related to speed
    Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);        // Set 4M to execute constant speed drive
                                                                                    // The other setting description is omitted.
    Ret = Nmc_3CIPExecMC8500P_BG(hWnd,No, IcNo, Data3Cip, 2, 0x7);
 // Execute 3-axis continuous interpolation. The number of data is 2, X, Y, Z axes

    if(Ret == CIP_START)     AfxMessageBox("Interpolation has started");
 // Return value is correct(Interpolation has started)
}

BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)    // WM_CIP_END message received function setting
    ON_MESSAGE( WM_CIP_END, OnMsg_CIP )
END_MESSAGE_MAP()

// WM_CIP_END message received function
afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_CIP(WPARAM BoardNo, LPARAM Status)
{
```

```
            if(Status == CIP_END)     AfxMessageBox("Interpolation has finished");
            // Return value is correct. (Interpolation has finished)
            return 0;
        }

    [VB.NET]
            '3-axis BP interpolation data setting
            Dim Data3Cip(1) As DATA_3CIP_MC8500P
            Data3Cip(0).EndP1 = 10000
            Data3Cip(0).EndP2 = 20000
            Data3Cip(0).EndP3 = 30000
            Data3Cip(0).Speed = 0

            Data3Cip(1).EndP1 = 2000
            Data3Cip(1).EndP2 = -1000
            Data3Cip(1).EndP3 = 3000
            Data3Cip(1).Speed = 0

            Call Nmc_IPMode(No, IcNo, &H7)                    ' Interpolation mode setting. Assign X,
                                                                Y and Z axis.

            ' Parameter setting related to speed
            Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)   ' Set 4M to execute constant speed drive
                                                              'The other setting description is
omitted.
            'Execute 3-axis continuous interpolation. The number of data is 2, X, Y, Z axes
            Ret = Nmc_3CIPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data3Cip, 2, &H7, False)
            If Ret = CIP_START Then                           ' Return value is correct. (Interpolation
            has started.)
                Call MsgBox("Interpolation has started")
            End If
        End Sub

        ' WM_CIP_END message received function
        Protected Overrides Sub WndProc(ByRef m As Message)
            Select Case (m.Msg)
                Case WM_CIP_END                               ' Continuous interpolation finish
                message.
                    If m.LParam.ToInt32 = CIP_END Then     ' Return value is correct.
                    ("Interpolation has finished")
                        Call MsgBox("Interpolation has finished")
                    End If
                    Exit Select
            End Select
            MyBase.WndProc(m)
        End Sub
    [C#]
        DATA_3CIP_MC8500P [] Data3Cip = new DATA_3CIP_MC8500P[2];     // 3-axis continuous
        interpolation data            // Interpolation data setting
        Data3Cip[0].EndP1 = 1000;
        Data3Cip[0].EndP2 = 2000;
        Data3Cip[0].EndP3 = 3000;
        Data3Cip[0].Speed = 0;

        Data3Cip[1].EndP1 = 2000;
        Data3Cip[1].EndP2 = -1000;
        Data3Cip[1].EndP3 = 3000;
        Data3Cip[1].Speed = 0;
```

```
        MC8000P.Nmc_IPMode(No, IcNo, 0x0007);                    // Interpolation mode setting. Assign
X, Y and Z axis.


        // Execute 3-axis continuous interpolation.
        // Execute in the background.
        Ret = MC8000P.Nmc_3CIPExec_BG((System.IntPtr)parent.Handle, gBoardNo, int IcNo,
                                        Data3Cip, 2, 0x7, SpdChgFlg, ContinueFlg);
        if(Ret == Nmc_Status.CIP_START)
        // Continuous interpolation has been successfully started.


        // WM_CIP_END message received function
        protected override void WndProc(ref Message m)
        {
           // Call original WndProc (call a constructor explicitly)
            base.WndProc ( ref m );
            if ( m.Msg == (int)MSG_ID.WM_CIP_END )
            {
                    if((uint)m.LParam == Nmc_Status.CIP_END)
                    // Continuous interpolation has been successfully completed.
            }
         }
```

Note

Make sure to set interpolation axis and so on in interpolation mode setting before executing this function. Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis.

This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content
Nmc_4CIPExecMC8500P_BG	Execute 4-axis continuous interpolation in the background using the specified interpolation data. This function returns control after the interpolation process has finished. Control will not return unless the interpolation process ends, so it is recommended to create a thread in the application and call from the thread. **VC** DWORD Nmc_4CIPExecMC8500P_BG(HWND User_hWnd, int No, int IcNo, DATA_4CIP_MC8500P* pData4Cip, int DataCnt, int IpAxis, BOOL SpdChgFlg = FALSE); **VB.NET** Function Nmc_4CIPExecMC8500P_BG(ByVal User_hWnd As Integer, ByVal No As Integer, ByVal IcNo As Integer, ByVal pData4Cip As DATA_4CIP_MC8500P(), ByVal DataCnt As Integer, ByVal IpAxis As Integer, ByVal SpdChgFlg As Integer) As Integer **C#** Nmc_Status MC8000P.Nmc_4CIPExecMC8500P_BG(System.IntPtr User_hWnd, int No, int IcNo, DATA_4CIP_MC8500P[] pData4Cip, int IpAxis, bool SpdChgFlg); **Input Parameter** User_hWnd Window handle of the user application No Board number (setting value of rotary switch (0~15) on the board) IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details. pData4Cip Pointer to an array (DATA_4CIP_MC8500P address) of DATA_4CIP_MC8500P structures (user-defined type). Set the 4-axis continuous interpolation data to DATA_4CIP_MC8500P.See [5.1.3] (3) for DATA_4CIP_MC8500P. DataCnt The number of 4-axis continuous interpolation data. Specify the number of the DATA_4CIP_MC8500P structure (user-defined type) array. IpAxis Axis to execute interpolation. Specify the same value as the setting value of D0~D3 (Axis assignment) of interpolation mode setting (2A). See [5.1.3] (4). SpdChgFlg Set the flag to change the speed during the interpolation process. If you change the speed, See [5.1.3] (5). [VC] TRUE: Change, FALSE: Not change. Can be omitted. Default is FALSE. [VB.NET] True: Change, False: Not change [C#] true: Change, false: Not change **When selecting Change** : Refers to the setting value of DATA_4CIP_MC8500P Speed. Set 1～4000000 to Speed・・・Changes to the setting speed. Set 0 to Speed・・・・・・ Not change the speed. **When selecting Not change:** Not refers to the setting value of DATA_4CIP_MC8500 Speed. **Return Value** If the interpolation process has been successfully started in the background, the return value is CIP_START. If an error occurred before starting the interpolation process, the return value is the following Error code (errors before starting the interpolation). For C#, See [5.1.3] (1). ■ Normal start CIP_START Continuous interpolation has been successfully started in the background. ■ Error code (errors before starting the interpolation) CIP_CNT_ERR The number of the specified data is out of range. CIP_ALREADY_EXEC BP interpolation or continuous interpolation is already running. CIP_THREAD_ERR Thread cannot be started. CIP_MALLOC_ERR Memory cannot be allocated. CIP_CMD_ERR The wrong command was specified. CIP_PARAM_ERR The parameter is incorrect.

CIP_NOT_OPEN_ERR The specified board is not opened.
CIP_OTHER_ERR Other errors.
CIP_FUNC_ERR Unusable function is used.

After the interpolation process has been successfully started in the background, WM_CIP_END message is sent to the specified window at the end of the interpolation. The board number is passed to the first argument received in WM_CIP_END message received function and finishing status is passed to the second argument.
If the interpolation has been successfully completed, the finishing status is CIP_END. If an error occurred during the interpolation process, the following Error code (errors after starting the interpolation) returns.

■ Normal end
CIP_END Continuous interpolation has been successfully completed.

■ Error code (errors after starting the interpolation)
CIP_USER_STOP The user aborted continuous interpolation.
CIP_DRIVE_ERR Error occurred in the board during continuous interpolation. (When the error status was set to RR0.)
CIP_STOP_CERR Continuous interpolation stopped during driving. (Due to RR2 error.
CIP_GETSC_ERR Stack counter reading error

Note
Set 4,000,000 to initial speed value before executing this function. (Don't change initial speed during executing this function).See [5.1.4] for more details.

Example

```
[VC]
    {
        DATA_4CIP_MC8500P Data4Cip[2];                    // 4-axis continuous interpolation data
        // 4-axis continuous interpolation data setting
        Data4Cip[0].EndP1 = 1000;
        Data4Cip[0].EndP2 = 2000;
        Data4Cip[0].EndP3 = 3000;
        Data4Cip[0].EndP4 = 4000;

        Data4Cip[1].EndP1 = 2000;
        Data4Cip[1].EndP2 = -1000;
        Data4Cip[1].EndP3 = 3000;
        Data4Cip[1].EndP4 = 4000;

        Nmc_IPMode(No, IcNo,0x000F);                      // Interpolation mode setting. Assign X, Y, Z, U axes.

        // Parameter setting related to speed.
        Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);    // Set 4M to execute constant speed drive
                                                                                    // The other setting description is omitted.
        Ret = Nmc_4CIPExecMC8500P_BG(hWnd,No, IcNo, Data4Cip, 2, 0xF);
        // Execute 4-axis continuous interpolation. The number of data is 2. X, Y, Z, U axes
        if(Ret == CIP_START)    AfxMessageBox("Interpolation has started");
        // Return value is correct. (Interpolation has started)
      }

      BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)
      // WM_CIP_END message received function setting.
          ON_MESSAGE( WM_CIP_END, OnMsg_CIP )
      END_MESSAGE_MAP()

      // WM_CIP_END message received function.
```

```
afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_CIP(WPARAM BoardNo, LPARAM Status)
{
    if(Status == CIP_END)     AfxMessageBox("Interpolation has finished"");
    // Return value is correct (Interpolation has finished")
    return 0;
}
```

[VB.NET]

```
' 4-axis continuous interpolation data
Dim Data4Cip(1) As DATA_4CIP_MC8500P
Data4Cip(0).EndP1 = 10000
Data4Cip(0).EndP2 = 20000
Data4Cip(0).EndP3 = 30000
Data4Cip(0).EndP4 = 40000
Data4Cip(0).Speed = 0

Data4Cip(1).EndP1 = 2000
Data4Cip(1).EndP2 = -1000
Data4Cip(1).EndP3 = 3000
Data4Cip(1).EndP4 = 4000
Data4Cip(1).Speed = 0

Call Nmc_IPMode(No, IcNo, &HF)                    ' Interpolation mode setting. Assign X, Y, Z
                                                    and U axis.


' Parameter setting related to speed
Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)     ' Set 4M to execute constant speed drive
                                                    'The other setting description is omitted.

'Execute 4-axis continuous interpolation. The number of data is 2. X, Y, Z, U axes
Ret = Nmc_4CIPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data4Cip, 2, &HF, False)
If Ret = CIP_START Then       ' Return value is correct. (Interpolation has started.)
    Call MsgBox("Interpolation has started ")
End If
End Sub

' WM_CIP_END message received function.
Protected Overrides Sub WndProc(ByRef m As Message)
    Select Case (m.Msg)
        Case WM_CIP_END                                ' Continuous interpolation finish
        message.
            If m.LParam.ToInt32 = CIP_END Then     ' Return value is correct.
            ("Interpolation has finished")
                Call MsgBox("Interpolation has finished ")
            End If
            Exit Select
    End Select
    MyBase.WndProc(m)
End Sub
```

[C#]

```
DATA_4CIP_MC8500P [] Data4Cip = new DATA_4CIP_MC8500P[2];
// 4-axis continuous interpolation data
// Interpolation data setting
Data4Cip[0].EndP1 = 1000;
Data4Cip[0].EndP2 = 2000;
Data4Cip[0].EndP3 = 3000;
Data4Cip[0].EndP4 = 3000;
```

```
            Data4Cip[0].Speed = 0;

            Data4Cip[1].EndP1 = 2000;
            Data4Cip[1].EndP2 = -1000;
            Data4Cip[1].EndP3 = 3000;
            Data4Cip[1].EndP4 = -2000;
            Data4Cip[1].Speed = 0;

            MC8000P.Nmc_IPMode(No, IcNo, 0x000F);                    // Interpolation mode setting. Assign
X, ,Y, Z and U axis

            // This function will not return control unless the interpolation process ends.
            // Execute 4-axis continuous interpolation.
            // Execute in the background.
            Ret = MC8000P.Nmc_4CIPExec_BG((System.IntPtr)parent.Handle, gBoardNo, int IcNo,
                                          Data4Cip, 2, 0xF, SpdChgFlg, ContinueFlg);

            if(Ret == Nmc_Status.CIP_START)
            // Continuous interpolation has been successfully started.

             // WM_BP_END message received function
             protected override void WndProc(ref Message m)
             {
                // Call original WndProc (call a constructor explicitly)
                  base.WndProc ( ref m );
                  if ( m.Msg == (int)MSG_ID.WM_CIP_END )
                  {
                        if((uint)m.LParam == Nmc_Status.CIP_END)    // Continuous interpolation has been
successfully
                  }
              }
```

Note

Make sure to set interpolation axis and so on in interpolation mode setting before executing this function. Set the same value to axis to be set in interpolation mode setting and axis specified by argument IpAxis.

This function operates in constant speed driving. Set speed parameter to constant speed drive.

Function Name	Function and Content

Nmc_IPStop	Stop the interpolation process during driving.
	The interpolation driving stops immediately and terminates the executed interpolation process in Nmc_xxx interpolation function.
	
	When stopping the interpolation process using Nmc_IPStop, the return value of each interpolation function is the following error code.
	◆ BP interpolation: BP_USER_STOP
	◆ Continuous interpolation: CIP_USER_STOP
	
	VC BOOL Nmc_IPStop(int No, int IcNo);
	VB.NET Function Nmc_IPStop(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	C# bool MC8000P.Nmc_IPStop(int No, int IcNo);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	
	Return value
	[VC] If the function succeeds, the return value is TRUE.
	If the function fails, the return value is FALSE.
	[VB.NET] If the function succeeds, the return value is nonzero.
	If the function fails, the return value is 0.
	[C#] If the function succeeds, the return value is true.
	If the function fails, the return value is false.
	
	Example
	[VC] Nmc_IPStop(No, IcNo); // Stop the interpolation process during driving.
	[VB.NET] Call Nmc_IPStop(No, IcNo) ' Stop the interpolation process during driving.
	[C#] MC8000P.Nmc_IPStop(No, IcNo);

Nmc_IPGetMsgNo	Read the board and IC number from the parameter wParam of the following received message at the end o the interpolation.
	◆BP interpolation : WM_BP_END
	◆Continuous interpolation : WM_CIP_END
	
	VC void Nmc_IPGetMsgNo(int wParam, int* No, int* IcNo);
	VB.NET Sub Nmc_IPGetMsgNo(ByVal wParam As Integer, ByRef No As Integer, ByRef IcNo As Integer)
	C# bool MC8000P.Nmc_IPGetMsgNo(int wParam, out int No, out int IcNo)
	
	Input Parameter
	wParam The parameter wParam of received message at the end of the Interpolation.
	No [VC] Address of a variable to store the board number.
	[VB.NET][C#] Variable to store the board number.
	IcNo [VC] Address of a variable to store the IC number.
	[VB.NET][C#] Variable to store the IC number.
	
	Return Value
	None
	
	Example
	[VC] int BdNo; // Board number
	int IcNo; // IC number
	Nmc_IPGetMsgNo(wParam, &BdNo, &IcNo);
	
	[VB.NET] Dim BdNo As Integer ' Board number
	Dim IcNo As Integer ' IC number (0～1)
	Call Nmc_IPGetMsgNo(m.WParam.ToInt32(), BdNo, IcNo)

	[C#] int BdNo; // Board number
	int IcNo; // IC number (0～1)
	MC8000P.Nmc_IPGetMsgNo(WParam, out BdNo, out IcNo)

Function Name	Function and Content

Nmc_HLValueExec	Execute helical calculation using the specified interpolation data.
	
	VC DWORD Nmc_HLValueExec(int No, int IcNo, DATA_HL* pDataHl);
	VB.NET Function Nmc_HLValueExec(ByVal No As Integer, ByVal IcNo As Integer, ByRef pDataHl As DATA_HL) As Integer
	C# Nmc_Status Nmc_HLValueExec(int No, int IcNo, ref DATA_HL pDataHl);
	
	Input Parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	pDataHl Pointer to an array(DATA_HL address) of DATA_HL structures (user-defined type). Set the helical calculation data to DATA_HL. See [5.1.3] (3) for DATA_HL. Specify CW circular interpolation (HL_DIR_CW) or CW circular interpolation (HL_DIR_CCW) to Direction.
	
	Return Value
	If the calculation process has been successfully finished, the return value is HL_VALUE_END. If an error occurred before starting the interpolation process, the return value is the following Error code.For C#, See [5.1.3] (1).
	
	■ Normal end
	HL_VALUE_END Helical calculation has been successfully completed.
	
	■ Error code
	HL_PARAM_ERR The parameter is incorrect.
	HL_NOT_OPEN_ERR The specified board is not opened.
	HL_FUNC_ERR Unusable function is used
	HL_OTHER_ERR Other errors.
	
	Example
	[VC]
	DATA_IPMODE IpMode; // Interpolation mode
	
	// Interpolation mode setting
	IpMode.Cxiv = FALSE; // Interpolation axes are not changed.
	IpMode.Lmdf = TRUE; // Enable short axis pulse equalization
	IpMode.Vcnst = 3; // 2-axis high accuracy constant vector speed
	
	DATA_HL DataHL; // Helical interpolation data
	
	// Helical interpolation data setting
	DataHL.Direction = HL_DIR_CCW; // Rotation direction (CCW circular interpolation)
	DataHL.Speed = 1000; // Speed
	DataHL.CenterX = 0; // Center point of X axis
	DataHL.CenterY = 10000; // Center point of Y axis
	DataHL.EndPX = 0; // Finish point of X axis
	DataHL.EndPY = 0; // Finish point of Y axis
	DataHL.MoveZ = 3000; // Moving amount of Z axis
	DataHL.MoveU = 400; // Moving amount of U axis
	DataHL.HlNum = 1; // Helical rotation number
	DataHL.HlValue = 0; // Helical calculation value
	DataHL.IpMode = IpMode; // Interpolation mode setting
	
	// Execute helical calculation
	Ret = Nmc_HLValueExec(0, 0, &DataHL);
	if(Ret == HL_VALUE_END) AfxMessageBox("Calculation has been finished");
	// Helical calculation has been successfully completed.

```
        // Helical calculation value received
        Value = DataHL.HlValue;                    // Store the result to Hl Value after calculation.

  [VB.NET]
        Dim IpMode As DATA_IPMODE          ' Interpolation mode
        Dim DataHL As DATA_HL              ' Helical interpolation data

        ' Interpolation mode setting
        IpMode.Cxiv = False                ' Interpolation axes are not changed.
        IpMode.Lmdf = True                 ' Enable short axis pulse equalization
        IpMode.Vcnst = 3                   ' 2-axis high accuracy constant vector speed
        ' Helical interpolation data setting
        DataHL.Direction = HL_DIR_CCW      ' Rotation direction (CCW circular interpolation)
        DataHL.Speed = 1000                ' Speed
        DataHL.CenterX = 0                 'Center point of X axis
        DataHL.CenterY = 10000             ' Center point of Y axis
        DataHL.EndPX = 0                   ' Finish point of X axis
        DataHL.EndPY = 0                   ' Finish point of Y axis
        DataHL.MoveZ = 3000                'Moving amount of Z axis
        DataHL.MoveU = 400                 'Moving amount of U axis
        DataHL.HlNum = 1                   ' Helical rotation number
        DataHL.HlValue = 0                 ' Helical calculation value
        DataHL.IpMode = IpMode             ' Interpolation mode setting

        ' Execute helical calculation
        Res = Nmc_HLValueExec(0, 0, DataHL)
         If Res = HL_VALUE_END Then        ' Helical calculation has been successfully completed.
        ' Helical calculation value received
        Value = DataHL.HlValue             ' Store the result to Hl Value after calculation.
           Call MsgBox("Calculation has been finished. ")
        End If

  [C#]
         DATA_HL DataHL = new DATA_HL();

        // Helical interpolation data setting
        DataHL.Direction = (ushort)HL_DIR.HL_DIR_CCW;          // Rotation direction (CCW circular
interpolation)
        DataHL.Speed = 1000;                    // Speed
        DataHL.CenterX = 0;                     // Center point of X axis
        DataHL.CenterY = 10000;                 // Center point of Y axis
        DataHL.EndPX = 0;                       // Finish point of X axis
        DataHL.EndPY = 0;                       // Finish point of Y axis
        DataHL.MoveZ = 3000;                    // Moving amount of Z axis
        DataHL.MoveU = 400;                     // Moving amount of U axis
        DataHL.HlNum = 1;                       // Helical rotation number
        DataHL.HlValue = 0;                     // Helical calculation value

        // Interpolation mode setting
        DataHL.IpMode.Cxiv = false;             // Interpolation axes are not changed.
        DataHL.IpMode.Lmdf = true;              // Enable short axis pulse equalization
        DataHL.IpMode.Vcnst = 3;                // 2-axis high accuracy constant vector speed

        Ret = MC8000P.Nmc_HLValueExec(0, 0, ref DataHL);

        if (Ret == Nmc_Status.HL_VALUE_END)   // Helical calculation has been successfully completed.
        {
        // Helical calculation value received
           Value = DataHL.HlValue;              // Store the result to Hl Value after calculation
        }
```

	Note When executing this function, interpolation mode setting (2A)H is performed by the value of structures of helical interpolation data of argument. Interpolation mode setting which is set before executing this function, please note this. Don’t call this function to the same IC during executing this function.

Function Name	Function and Content

Nmc_HLExec	Execute helical interpolation using the specified interpolation data.
	
	VC DWORD Nmc_HLValueExec(int No, int IcNo, DATA_HL* pDataHl);
	VB.NET Function Nmc_HLValueExec(ByVal No As Integer, ByVal IcNo As Integer, ByRef pDataHl As DATA_HL) As Integer
	C# Nmc_Status Nmc_HLValueExec(int No, int IcNo, ref DATA_HL pDataHl);
	
	Input parameter
	No Board number (setting value of rotary switch (0~15) on the board)
	IcNo IC number (0~1). Set 0 when the board has one IC. Set 0 to IC-A and 1 to IC-B when the board has 2 or more ICs. See [5.1.3] (7) for more details.
	pDataHl Pointer to an array(DATA_HL address) of DATA_HL structures (user-defined type). Set the helical calculation data to DATA_HL. See [5.1.3] (3) for DATA_HL. Specify CW circular interpolation (HL_DIR_CW) or CW circular interpolation
	Return Value
	If the calculation process has been successfully finished, the return value is HL_VALUE_END.
	If an error occurred before starting the interpolation process, the return value is the following Error code.
	For C#, See [5.1.3] (1).
	
	■ Normal end
	HL_VALUE_END Helical calculation has been successfully completed.
	
	■ Error code
	HL_PARAM_ERR The parameter is incorrect.
	HL_NOT_OPEN_ERR The specified board is not opened.
	HL_FUNC_ERR Unusable function is used
	HL_OTHER_ERR Other errors.
	
	Example
	[VC]
	DATA_IPMODE IpMode; // Interpolation mode
	
	// Interpolation mode setting
	IpMode.Cxiv = FALSE; // interpolation axes are not changed.
	IpMode.Lmdf = TRUE; // Enable short axis pulse equalization
	IpMode.Vcnst = 3; // 2-axis high accuracy constant vector speed
	
	DATA_HL DataHL; // Helical interpolation data
	
	// Helical interpolation data setting
	DataHL.Direction = HL_DIR_CCW; // Rotation direction (CCW circular interpolation)
	DataHL.Speed = 1000; // Speed
	DataHL.CenterX = 0; // Center point of X axis
	DataHL.CenterY = 10000; // Center point of Y axis
	DataHL.EndPX = 0; // Finish point of X axis
	DataHL.EndPY = 0; // Finish point of Y axis
	DataHL.MoveZ = 3000; // Moving amount of Z axis
	DataHL.MoveU = 400; // Moving amount of U axis
	DataHL.HlNum = 1; // Helical rotation number
	DataHL.HlValue = 0; // Helical calculation value
	DataHL.IpMode = IpMode; // Interpolation mode setting
	
	// Execute helical calculation value
	Ret = Nmc_HLValueExec(0, 0, &DataHL);
	if(Ret == HL_VALUE_END) AfxMessageBox("Calculation has been finished ");
	// Helical calculation has been successfully completed.
	
	// Helical calculation value received

```
        Value = DataHL.HlValue;                    // Store the result to Hl Value after calculation

  [VB.NET]
        Dim IpMode As DATA_IPMODE          ' Interpolation mode
        Dim DataHL As DATA_HL              ' Helical interpolation data

        ' Interpolation mode setting
        IpMode.Cxiv = False                ' Interpolation axes are not changed.
        IpMode.Lmdf = True                 ' Enable short axis pulse equalization
        IpMode.Vcnst = 3                   '2-axis high accuracy constant vector speed
        ' Helical interpolation data setting
        DataHL.Direction = HL_DIR_CCW      ' Rotation direction (CCW circular interpolation)
        DataHL.Speed = 1000                ' Speed
        DataHL.CenterX = 0                 ' Center point of X axis
        DataHL.CenterY = 10000             ' Center point of Y axis
        DataHL.EndPX = 0                   ' Finish point of X axis
        DataHL.EndPY = 0                   ' Finish point of Y axis
        DataHL.MoveZ = 3000                ' Moving amount of Z axis
        DataHL.MoveU = 400                 ' Moving amount of U axis
        DataHL.HlNum = 1                   ' Helical rotation number
        DataHL.HlValue = 0                 ' Helical calculation value
        DataHL.IpMode = IpMode             ' Interpolation mode setting

        ' Execute helical calculation value
        Res = Nmc_HLValueExec(0, 0, DataHL)
        If Res = HL_VALUE_END Then         ' Helical calculation has been successfully completed.
          ' Helical calculation value received
          Value = DataHL.HlValue           ' Store the result to Hl Value after calculation
          Call MsgBox("Calculation has been finished ")
        End If

  [C#]
         DATA_HL DataHL = new DATA_HL();

        // Helical interpolation data setting
        DataHL.Direction = (ushort)HL_DIR.HL_DIR_CCW;          // Rotation direction (CCW circular
interpolation)

        DataHL.Speed = 1000;                    // Speed
        DataHL.CenterX = 0;                     // Center point of X axis
        DataHL.CenterY = 10000;                 // Center point of Y axis
        DataHL.EndPX = 0;                       // Finish point of X axis
        DataHL.EndPY = 0;                       // Finish point of Y axis
        DataHL.MoveZ = 3000;                    // Moving amount of Z axis
        DataHL.MoveU = 400;                     // Moving amount of U axis
        DataHL.HlNum = 1;                       // Helical rotation number
        DataHL.HlValue = 0;                     // Helical calculation value

        // Interpolation mode setting
        DataHL.IpMode.Cxiv = false;             // Interpolation axes are not changed.
        DataHL.IpMode.Lmdf = true;              // Enable short axis pulse equalization
        DataHL.IpMode.Vcnst = 3;                // 2-axis high accuracy constant vector speed

        Ret = MC8000P.Nmc_HLValueExec(0, 0, ref DataHL);

        if (Ret == Nmc_Status.HL_VALUE_END)   // Helical calculation has been successfully completed.
        {
          // Helical calculation value received
          Value = DataHL.HlValue;               // Store the result to Hl Value after calculation
```

	} **Note** When executing this function, interpolation mode setting (2A)H is performed by the value of structures of helical interpolation data of argument. Interpolation mode setting which is set before executing this function, please note this. Don’t call this function to the same IC during executing this function.

5.1.3 Footnote

(1) Each definition is defined in the following files.

VC++ ・・・MC8000P_DLL.h
VB.NET・・・MC8000P_DLL.vb
C#・・・・Users can refer to or input each definition through the IntelliSense

VC++, VB.NET, and C# definitions are as follows:

①Register number

[VC]

```
#define MCX_WR0             0x0000    // WR0
#define MCX_WR1             0x0001    // WR1
#define MCX_WR2             0x0002    // WR2
#define MCX_WR3             0x0003    // WR3
#define MCX_WR4             0x0004    // WR4
#define MCX_WR5             0x0005    // WR5
#define MCX_WR6             0x0006    // WR6
#define MCX_WR7             0x0007    // WR7

#define MCX_RR0             0x0000    // RR0
#define MCX_RR1             0x0001    // RR1
#define MCX_RR2             0x0002    // RR2
#define MCX_RR3             0x0003    // RR3
#define MCX_RR4             0x0004    // RR4
#define MCX_RR5             0x0005    // RR5
#define MCX_RR6             0x0006    // RR6
#define MCX_RR7             0x0007    // RR7
```

[C#]

```
// ■Nmc_OutPort/Nmc_InPort
// When the board has one IC, use WR0_A ~| WR7_A/RR0_A ~ RR7_A
// When the board has 2 ICs, use WR0_A ~ WR7_A/RR0_A ~ RR7_A for IC_A, WR0_B ~ WR7_B/RR0_B ~ RR7_B for IC_B.
// ■Nmc_WriteReg/Nmc_ReadReg
// Use WR0 ~ WR7/RR0 ~ RR7

   public enum REG_MCX : int
   {
                        //■Nmc_OutPort
                        // Write register address
                        // WR0～WR7 for IC-A
                        WR0_A    =0x0000,
                        WR1_A    =0x0001,
                        WR2_A    =0x0002,
                        WR3_A    =0x0003,
                        WR4_A    =0x0004,
                        WR5_A    =0x0005,
                        WR6_A    =0x0006,
                        WR7_A    =0x0007,

                        // WR0~WR7 for IC-B
                        WR0_B    =0x0008,
                        WR1_B    =0x0009,
                        WR2_B    =0x000A,
                        WR3_B    =0x000B,
                        WR4_B    =0x000C,
                        WR5_B    =0x000D,
                        WR6_B    =0x000E,
                        WR7_B    =0x000F,
```

```
        //■Nmc_InPort
        // Read register address
        // RR0～RR7 for IC-A
        RR0_A    =0x0000,
        RR1_A    =0x0001,
        RR2_A    =0x0002,
        RR3_A    =0x0003,
        RR4_A    =0x0004,
        RR5_A    =0x0005,
        RR6_A    =0x0006,
        RR7_A    =0x0007,

        // RR0～RR7 for IC-B
        RR0_B    =0x0008,
        RR1_B    =0x0009,
        RR2_B    =0x000A,
        RR3_B    =0x000B,
        RR4_B    =0x000C,
        RR5_B    =0x000D,
        RR6_B    =0x000E,
        RR7_B    =0x000F,

        //■Nmc_WriteReg
        // Write register address
        WR0      =0x0000,
        WR1      =0x0001,
        WR2      =0x0002,
        WR3      =0x0003,
        WR4      =0x0004,
        WR5      =0x0005,
        WR6      =0x0006,
        WR7      =0x0007,

        //■Nmc_ReadReg
        // Read register address
        RR0      =0x0000,
        RR1      =0x0001,
        RR2      =0x0002,
        RR3      =0x0003,
        RR4      =0x0004,
        RR5      =0x0005,
        RR6      =0x0006,
        RR7      =0x0007
    }
    Example) REG_MCX of WR0        REG_MCX.WR0
```

②Axis assignment

```
[VC]
  #define AXIS_ALL          0xF      // All axes
  #define AXIS_X            0x1      // X axis
  #define AXIS_Y            0x2      // Y axis
  #define AXIS_Z            0x4      // Z axis
  #define AXIS_U            0x8      // U axis
  #define AXIS_NONE         0        // No axis assignment

[C#]
  public enum AXIS : int
  {
```

```
            // Axis assignment
        ALL     =0xF,     // All axes
        X       =0x1,     // X axis
        Y       =0x2,     // Y axis
        Z       =0x4,     // Z axis
        U       =0x8,     // U axis
        NONE     =0       // No axis assignment
         }
    Example) To assign all axes,          AXIS.ALL
```

③Device ID

[VC]

Device ID is the unique number assigned to the board.

Number of board is as follows.

Board	Device ID
MC8541P / MC8541Pe	0xA808
MC8581P / MC8581Pe	0xA809

```
[VC]
 #define ID_MC8541P          0xA808          // MC8541P and MC8541Pe
 #define ID_MC8581P          0xA809          // MC8581P and MC8581Pe

[C#]
 public enum Dev_ID : ushort
 {
        MC8541P          =0xA808,  // MC8541P and MC8541Pe
        MC8581P          =0xA809   // MC8581P and MC8581Pe

 }
 Example) MC8541P is Dev_ID.MC8541P, MC8581P is Dev_ID.MC8581P.
```

④Command definition *Refer to MCX514 User's Manual for available commands.

```
[VC]
 // Driving commands
#define MC8500P_CMD_DRVRL          0x0050    // Relative position drive
#define MC8500P_CMD_DRVNR          0x0051    // Counter relative position drive
#define MC8500P_CMD_DRVVP          0x0052    // + direction continuous pulse drive
#define MC8500P_CMD_DRVVM          0x0053    // - direction continuous pulse drive
#define MC8500P_CMD_DRVAB          0x0054    // Absolute position drive
#define MC8500P_CMD_DRVSBRK        0x0056    // Decelerating stop
#define MC8500P_CMD_DRVFBRK        0x0057    // Sudden stop
#define MC8500P_CMD_DIRCP          0x0058    // Direction signal + setting
#define MC8500P_CMD_DIRCM          0x0059    // Direction signal - setting
#define MC8500P_CMD_HMSRC          0x005A    // Automatic home search execution

// Interpolation commands
#define MC8500P_CMD_1CIP           0x0060     // 1-axis linear interpolation drive (For multi-chip) *MC8581P only.
#define MC8500P_CMD_2CIP           0x0061    // 2-axis linear interpolation drive
#define MC8500P_CMD_3CIP           0x0062    // 3-axis linear interpolation drive
#define MC8500P_CMD_4CIP           0x0063    // 4-axis linear interpolation drive
#define MC8500P_CMD_CIPCW          0x0064    // CW circular interpolation drive
#define MC8500P_CMD_CIPCCW         0x0065    // CCW circular interpolation drive
#define MC8500P_CMD_BPIP2          0x0066    // 2-axis bit pattern interpolation drive
#define MC8500P_CMD_BPIP3          0x0067    // 3-axis bit pattern interpolation drive
#define MC8500P_CMD_BPIP4          0x0068    // 4-axis bit pattern interpolation drive
#define MC8500P_CMD_HLCW           0x0069    // CW helical interpolation drive
#define MC8500P_CMD_HLCCW          0x006A    // CCW helical interpolation drive
```

```
#define MC8500P_CMD_HLPCW          0x006B    // CW helical calculation
#define MC8500P_CMD_HLPCCW         0x006C    // CCW helical calculation
#define MC8500P_CMD_DECEN          0x006D    // Decelerating enabling
#define MC8500P_CMD_DECDIS         0x006E    // Decelerating disabling
#define MC8500P_CMD_CLRINTRPT      0x006F    // Interpolation interrupt clear
#define MC8500P_CMD_IPSTEP         0x006F    // Single step interpolation

// Other commands
#define MC8500P_CMD_VINC           0x0070    // Deceleration increase
#define MC8500P_CMD_VDEC           0x0071    // Deceleration decrease
#define MC8500P_CMD_TMSTA          0x0073    // Timer start
#define MC8500P_CMD_TMSTP          0x0074    // Timer stop
#define MC8500P_CMD_DHOLD          0x0077    // Drive start holding
#define MC8500P_CMD_DFREE          0x0078    // Drive start holding release
#define MC8500P_CMD_R2CLR          0x0079    // Error/finish status clear
#define MC8500P_CMD_RR3P0          0x007A    // RR3 page 0 display
#define MC8500P_CMD_RR3P1          0x007B    // RR3 page 1 display
#define MC8500P_CMD_TXCLR          0x007C    // Finish point maximum value clear
#define MC8500P_CMD_NOP            0x001F    // NOP
#define MC8500P_CMD_RST            0x00FF    // Command reset
```

[C#]

```
public enum CMD : int
{
// Drive commands
MC8500P_CMD_DRVRL          = 0x0050,          // Relative position drive
MC8500P_CMD_DRVNR          = 0x0051,          // Counter relative position drive
MC8500P_CMD_DRVVP          = 0x0052,          // + direction continuous pulse drive
MC8500P_CMD_DRVVM          = 0x0053,          // - direction continuous pulse drive
MC8500P_CMD_DRVAB          = 0x0054,          // Absolute position drive
MC8500P_CMD_DRVSBRK        = 0x0056,          // Decelerating stop
MC8500P_CMD_DRVFBRK        = 0x0057,          // Sudden stop
MC8500P_CMD_DIRCP          = 0x0058,          // Direction signal + setting
MC8500P_CMD_DIRCM          = 0x0059,          // Direction signal - setting
MC8500P_CMD_HMSRC          = 0x005A,          // Automatic home search execution

// Interpolation commands
MC8500P_CMD_1CIP           = 0x0060,          // 1-axis linear interpolation drive (For multi-chip)*MC8581P only.
MC8500P_CMD_2CIP           = 0x0061,          // 2-axis linear interpolation drive
MC8500P_CMD_3CIP           = 0x0062,          // 3-axis linear interpolation drive
MC8500P_CMD_4CIP           = 0x0063,          // 4-axis linear interpolation drive
MC8500P_CMD_CIPCW          = 0x0064,          // CW circular interpolation drive
MC8500P_CMD_CIPCCW         = 0x0065,          // CCW circular interpolation drive
MC8500P_CMD_BPIP2          = 0x0066,          // 2-axis bit pattern interpolation drive
MC8500P_CMD_BPIP3          = 0x0067,          // 3-axis bit pattern interpolation drive
MC8500P_CMD_BPIP4          = 0x0068,          // 4-axis bit pattern interpolation drive
MC8500P_CMD_HLCW           = 0x0069,          // CW helical interpolation drive
MC8500P_CMD_HLCCW          = 0x006A,          // CCW helical interpolation drive
MC8500P_CMD_HLPCW          = 0x006B,          // CW helical calculation
MC8500P_CMD_HLPCCW         = 0x006C,          // CCW helical calculation
MC8500P_CMD_DECEN          = 0x006D,          // Decelerating enabling
MC8500P_CMD_DECDIS         = 0x006E,          // Decelerating disabling
MC8500P_CMD_CLRINTRPT      = 0x006F,          // Interpolation interrupt clear
MC8500P_CMD_IPSTEP         = 0x006F,          // Single step interpolation

// Other commands
MC8500P_CMD_VINC           = 0x0070,          // Deceleration increase
MC8500P_CMD_VDEC           = 0x0071,          // Deceleration decrease
MC8500P_CMD_DCC            = 0x0072,          // Deviation counter clear output
```

```
MC8500P_CMD_TMSTA       = 0x0073,       // Timer start
MC8500P_CMD_TMSTP       = 0x0074,       // Timer stop
MC8500P_CMD_SPSTA       = 0x0075,       // Split pulse start
MC8500P_CMD_SPSTP       =0x0076,        // Split pulse stop
MC8500P_CMD_DHOLD       = 0x0077,       // Drive start holding
MC8500P_CMD_DFREE       = 0x0078,       // Drive start holding release
MC8500P_CMD_R2CLR       = 0x0079,       // Error / Finishing status clear
MC8500P_CMD_RR3P0       = 0x007A,       // RR3 Page0 display
MC8500P_CMD_RR3P1       = 0x007B,       // RR3 Page1 display
MC8500P_CMD_TXCLR       = 0x007C,       // Finish point maximum value clear
MC8500P_CMD_NOP         = 0x001F,       // NOP
MC8500P_CMD_RST         = 0x00FF        // Command reset
}
```

Example) To specify 2-axis liner interpolation drive, CMD.MC8500P_CMD_2CIP

⑤Interpolation finishing message, finishing status

[VC]

```
// Interpolation finishing message
#define WM_BP_END               (WM_USER + 1)   // BP interpolation finishing message
#define WM_CIP_END              (WM_USER + 2)   // Continuous interpolation finishing message

//***** BP Interpolation  Finishing Status *****
// ■Normal
#define BP_START                0x101   // BP interpolation has started in the background.
#define BP_END                  0x102   // BP interpolation has been successfully completed.

// ■Errors before starting the interpolation
#define BP_CNT_ERR              0x111   // The number of the specified data is out of range.
#define BP_ALREADY_EXEC         0x112   // BP interpolation or continuous interpolation is already running.
#define BP_THREAD_ERR           0x113   // Thread cannot be started.
#define BP_MALLOC_ERR           0x114   // Memory cannot be allocated.
#define BP_PARAM_ERR            0x116   // The parameter is incorrect.
#define BP_NOT_OPEN_ERR         0x117   // The specified board is not opened.
#define BP_OTHER_ERR            0x118   // Other errors.
#define BP_FUNC_ERR             0x119   // Unusable function is used.

//■ Errors during the interpolation driving
#define BP_STOP                 0x121   // BP interpolation stopped during driving. (Too  fast tostack next data)
#define BP_USER_STOP            0x122   // The user aborted BP interpolation.
#define BP_DRIVE_ERR            0x123   // Error occurred in the board during BP interpolation. (When
                                            the error status was set to RR0.)
#define BP_STOP_CERR            0x124   // Continuous interpolation stops during driving. (Due to RR2 error)
#define BP_GETSC_ERR            0x125   // Stack counter reading error

//***** Continuous Interpolation  Finishing Status *****
// ■Normal
#define CIP_START       0x201   // Continuous interpolation has started in the background.
#define CIP_END                 0x202   // Continuous interpolation has been successfully completed.

// ■Errors before starting the interpolation
#define CIP_CNT_ERR             0x211   // The number of the specified data is out of range.
#define CIP_ALREADY_EXEC        0x212   // BP interpolation or continuous interpolation is already running.
#define CIP_THREAD_ERR          0x213   // Thread cannot be started.
#define CIP_MALLOC_ERR          0x214   // Memory cannot be allocated.
#define CIP_CMD_ERR             0x215   // Command error (The wrong command was specified by the user.)
#define CIP_PARAM_ERR           0x216   // The parameter is incorrect.
#define CIP_NOT_OPEN_ERR        0x217   // The specified board is not opened.
#define CIP_OTHER_ERR           0x218   // Other errors.
```

```
#define CIP_FUNC_ERR                    0x219       // Unusable function is used.

// ■Errors during the interpolation driving
#define CIP_STOP                        0x221       // Continuous interpolation stopped during driving. (Too fast to set next
                                                       data)
#define CIP_USER_STOP                   0x222       // The user aborted Continuous interpolation.
#define CIP_DRIVE_ERR                   0x223       // Error occurred in the board during Continuous interpolation. (When the
                                                       error status was set to RR0.)
#define CIP_STOP_CERR                   0x224       // Continuous interpolation stops during driving. (Due to RR2 error)
#define CIP_GETSC_ERR                   0x225       // Stack counter reading error

//***** Helical Interpolation     Finishing Status *****
// ■Normal
#define HL_START                        0x301       // Helical interpolation has started.
#define HL_VALUE_END                    0x302       // Helical calculation finishes.

// ■Errors before starting helical interpolation
#define HL_CNT_ERR                      0x311       // The number of the specified data is out of range.
#define HL_PARAM_ERR                    0x312       // The parameter is incorrect.
#define HL_NOT_OPEN_ERR                 0x313       // The specified board is not opened.
#define HL_FUNC_ERR                     0x314       // Unusable function is used.
#define HL_OTHER_ERR                    0x315       // Other errors.

 [C#]
public enum MSG_ID : int
{           // Interpolation finishing message
WM_USER                                 =0x0400,
WM_BP_END                               =WM_USER+1,     // (WM_USER + 1) BP interpolation finishing message
WM_CIP_END                              =WM_USER+2,     // (WM_USER + 2) Continuous interpolation finishing
                                                           message
            }
Example) MSG_ID and WM_BP_END       MSG_ID.WM_BP_END

            public enum Nmc_Status : uint
            {

//***** BP Interpolation     Finishing Status *****
// ■Normal
BP_START                                = 0x101,            // BP interpolation has started in the background.
BP_END                                  = 0x102,            // BP interpolation has been successfully completed.

// ■Errors before starting the interpolation
BP_CNT_ERR                              = 0x111,            // The number of the specified data is out of range.
BP_ALREADY_EXEC                         = 0x112,            // BP interpolation or continuous interpolation is
                                                               already running.
BP_THREAD_ERR                           = 0x113,            // Thread cannot be started.
BP_MALLOC_ERR                           = 0x114,            // Memory cannot be allocated.
BP_PARAM_ERR                            = 0x116,            // The parameter is incorrect.
BP_NOT_OPEN_ERR                         = 0x117,            // The specified board is not opened.
BP_OTHER_ERR                            = 0x118,            // Other errors.
BP_FUNC_ERR                             = 0x119,            // Unusable function is used.

// ■Errors during the interpolation driving
BP_STOP                 = 0x121, // BP interpolation stopped during driving. (Too fast to stack next data)
BP_USER_STOP            = 0x122, // The user aborted BP interpolation. The user aborted BP interpolation
BP_DRIVE_ERR            = 0x123, // Error occurred in the board during BP interpolation. (When the error status was set to
                                    RR0.)
BP_STOP_CERR            = 0x124, // Continuous interpolation stops during driving. (Due to RR2 error)
BP_GETSC_ERR            = 0x125, // Stack counter reading error
```

```
//***** Continuous Interpolation     Finishing Status *****
// ■Normal
CIP_START              = 0x201, // Continuous interpolation has started in the background.
CIP_END                = 0x202, // Continuous interpolation has been successfully completed.

// ■Errors before starting the interpolation
CIP_CNT_ERR                = 0x211,        // The number of the specified data is out of range.
CIP_ALREADY_EXEC           = 0x212,        // BP interpolation or continuous interpolation is already running.
CIP_THREAD_ERR             = 0x213,        // Thread cannot be started.
CIP_MALLOC_ERR             = 0x214,        // Memory cannot be allocated.
CIP_CMD_ERR                = 0x215,        // Command error (The wrong command was specified by the user.)
CIP_PARAM_ERR              = 0x216,        // The parameter is incorrect.
CIP_NOT_OPEN_ERR           = 0x217,        // The specified board is not opened.
CIP_OTHER_ERR              = 0x218,        // Other errors.
CIP_FUNC_ERR               = 0x219,        // Unusable function is used.

// ■Errors during the interpolation driving
CIP_STOP                   = 0x221, // Continuous interpolation stopped during driving. (Too fast to set next
data)
CIP_USER_STOP              = 0x222, // The user aborted Continuous interpolation.
CIP_DRIVE_ERR              = 0x223, // Error occurred in the board during Continuous interpolation. (When the
                                       error status was set to RR0.)
CIP_STOP_CERR              = 0x224, // Continuous interpolation stops during driving. (Due to RR2 error)
CIP_GETSC_ERR              = 0x225, // Stack counter reading error

//***** Helical Interpolation     Finishing Status *****
// ■Normal
HL_START                   = 0x301,        // Helical interpolation has started
HL_VALUE_END               = 0x302,        // Helical calculation finishes.

// ■Errors before starting helical interpolation
HL_CNT_ERR             = 0x311,        // The number of the specified data is out of range.
HL_PARAM_ERR           = 0x312,        // The parameter is incorrect.
HL_NOT_OPEN_ERR        = 0x313,        // The specified board is not opened.
HL_FUNC_ERR            = 0x314,        // Unusable function is used. (When using the function for MC8000P to
                                          MC8500P)
HL_OTHER_ERR           = 0x315,        // Other errors.
}

⑥Helical interpolation rotation direction
[VC]
#define HL_DIR_CW       0              // CW helical interpolation
#define HL_DIR_CCW      1              // CCW helical interpolation

[C#]
public enum HL_DIR : int
        {
HL_DIR_CW          = 0,                // CW helical interpolation
HL_DIR_CCW         = 1                 // CCW helical interpolation
        }
```

(2) The method of axis assignment is as follows:

Axis	VC, VB.NET
X	AXIS_X
Y	AXIS_Y
Z	AXIS_Z
U	AXIS_U
All axes	AXIS_ALL

①To assign 1 axis
Specify one of the following axes: AXIS_X, AXIS_Y, AXIS_Z, and AXIS_U.
Example) Set 1000 to the drive speed of X axis.

```
[VC]        Nmc_Speed(No, IcNo, AXIS_X, 1000);
[VB.NET]    Call Nmc_Speed(No, IcNo, AXIS_X, 1000)
[C#]        MC8000P.Nmc_Speed(No, IcNo, AXIS.X, 1000);
```

②To assign 2 axes
Use Bit OR operator.
For instance, if the user tries to assign X and Y axes simultaneously,

```
[VC]・・・・Specify AXIS_X | AXIS_Y.
[VB.NET]・・Specify AXIS_X Or AXIS_Y.
[C#]・・・・Specify AXIS.X | AXIS.Y.
```

Example) Set 1000 to the drive speed of X and Y axes.

```
[VC]        Nmc_Speed(No, IcNo, AXIS_X | AXIS_Y, 1000);
[VB.NET]    Call Nmc_Speed(No, IcNo, AXIS_X Or AXIS_Y, 1000)
[C#]        MC8000P.Nmc_Speed(No, IcNo, AXIS.X | AXIS.Y, 1000);
```

③To assign 3 axes
Use Bit OR operator.
For instance, if the user tries to assign X, Y and Z axes simultaneously,

```
[VC]・・・・Specify AXIS_X | AXIS_Y | AXIS_Z.
[VB.NET]・・Specify AXIS_X Or AXIS_Y Or AXIS_Z.
[C#]・・・・Specify AXIS.X | AXIS.Y | AXIS.Z.
```

Example) Set 1000 to the drive speed of X, Y and Z axes.

```
[VC]        Nmc_Speed(No, IcNo, AXIS_X | AXIS_Y | AXIS_Z, 1000);
[VB.NET]    Call Nmc_Speed(No, IcNo, AXIS_X Or AXIS_Y Or AXIS_Z, 1000)
[C#]        MC8000P.Nmc_Speed(No, IcNo, AXIS.X | AXIS.Y | AXIS.Z, 1000);
```

④To assign all axes
Specify AXIS_ALL.
Example) Set 1000 to the drive speed of all axes.

```
[VC]        Nmc_Speed(No, IcNo, AXIS_ALL, 1000);
[VB.NET]    Call Nmc_Speed(No, IcNo, AXIS_ALL, 1000)

[C#]        MC8000P.Nmc_Speed(No, IcNo, AXIS.ALL, 1000);
```

(3) The structure used in the interpolation function is defined as follows:
①VC

```
// 2-axis BP interpolation
        typedef struct _DATA_2BP
        {
                USHORT Bp1p;            // BP1P data
                USHORT Bp1m;            // BP1M data
                USHORT Bp2p;            // BP2P data
                USHORT Bp2m;            // BP2M data
        } DATA_2BP;

         // 3-axis BP interpolation
          typedef struct _DATA_3BP
        {
           USHORT Bp1p;             // BP1P data
            USHORT Bp1m;            // BP1M data
            USHORT Bp2p;            // BP2P data
            USHORT Bp2m;            // BP2M data
            USHORT Bp3p;            // BP3P data
            USHORT Bp3m;            // BP3M data
```

```
  } DATA_3BP;

 // 4-axis BP interpolation
 typedef struct _DATA_4BP
 {
   USHORT Bp1p;              // BP1P data
    USHORT Bp1m;             // BP1M data
    USHORT Bp2p;             // BP2P data
    USHORT Bp2m;             // BP2M data
    USHORT Bp3p;             // BP3P data
    USHORT Bp3m;             // BP3M data
    USHORT Bp4p;             // BP4P data
    USHORT Bp4m;             // BP4M data
 } DATA_4BP;

// 2-axis continuous interpolation
 typedef struct _DATA_2CIP    _MC8500P
 {
        USHORT Command; // Command number (Set MC8500P_CMD_2CIP, MC8500P_CMD_CIPCW,
 MC8500P_CMD_CIPCCW)
        long    Speed;          // Speed (When changing the speed, set 1~4,000,000. When not changing,
 set 0.)
        long    EndP1;          // Finishing point (The first axis)
        long    EndP2;          // Finishing point (The second axis)
        long    Center1;        // Circular center point (The first axis)
        long    Center2;        // Circular center point (The second axis)
 } DATA_2CIP_MC8500P;     // Note: Specify interpolation axis in interpolation mode setting (2A)H.

// 3-axis continuous interpolation
 typedef struct _DATA_3CIP    _MC8500P
 {
      long    EndP1;          // Finishing point (The first axis)
      long    EndP2;          // Finishing point (The second axis)
      long    EndP3;          // Finishing point (The third axis)
        long    Speed;          // Speed (When changing the speed, set 1~4,000,000. When not changing,
 set 0.)
 } DATA_3CIP_MC8500P;     // Note: Specify interpolation axis in interpolation mode setting (2A)H.

// 4-axis continuous interpolation
 typedef struct _DATA_4CIP    _MC8500P
 {
        long    EndP1;          // Finishing point (The first axis)
        long    EndP2;          // Finishing point (The second axis)
        long    EndP3;          // Finishing point (The third axis)
       long     EndP4;          // Finishing point (The fourth axis)
        long    Speed;          // Speed (When changing the speed, set 1~4,000,000. When not changing,
 set 0.)
 } DATA_4CIP_MC8500P;     // Note: Specify interpolation axis in interpolation mode setting (2A)H.

// Interpolation mode setting
 typedef struct _DATA_IPMODE
 {
        BOOL    Cxiv;           // Interpolation axis change in circular interpolation (TURE: Change,
 FALSE: Not change)
        BOOL    Lmdf;           // Short axis pulse equalization (TRUE: Enable, FALSE: Disable)
        USHORT Vcnst;           // Constant vector speed (0:None, 1:2-axis simple, 2:3-axis simple,
 3:2-axis high accuracy)
 }
 DATA_IPMODE;
```

```
// Helical interpolation
        typedef struct _DATA_HL
        {
        USHORT Direction;           // Rotation direction (Set HL_DIR_CW: CW circular interpolation or HL_DIR_CCW:
CCW circular interpolation.)
                long    Speed;              // Speed (1~2,000,000)
                long    CenterX;            // Circular center point (X-axis)
                long    CenterY;            // Circular center point (Y-axis)
                long    EndPX;              // Finish point (X-axis)
                long    EndPY;              // Finish point (Y-axis)
                long   MoveZ;               // Moving amount (Z-axis) (0:Z-axis is not used. other than 0: Moving
        amount of Z-axis)
                long    MoveU;              // Moving amount (U-axis) (0:U-axis is not used. other than 0: Moving
        amount of U-axis)
                USHORT HlNum;               // Helical rotation number (0~65,535)
              long HlValue;                 // Helical calculation value (0: Find helical calculation value before executing
                                            interpolation, 0 or more: Execute interpolation with the calculation value which is
                                            set
                DATA_IPMODE     IpMode; // Interpolation mode setting.
        }
        DATA_HL;
```

②VB.NET

```
' 2-axis BP interpolation
Structure DATA_2BP
    Dim Bp1p As Short       ' BP1P data
    Dim Bp1m As Short       ' BP1M data
    Dim Bp2p As Short       ' BP2P data
    Dim Bp2m As Short           ' BP2M data
End Structure

' 3-axis BP interpolation
Structure DATA_3BP
   Dim Bp1p As Short        ' BP1P data
    Dim Bp1m As Short       ' BP1M data
    Dim Bp2p As Short       ' BP2P data
    Dim Bp2m As Short       ' BP2M data
    Dim Bp3p As Short       ' BP3P data
    Dim Bp3m As Short       ' BP3M data
End Structure

' 4-axis BP interpolation
Structure DATA_4BP
   Dim Bp1p As Short        ' BP1P data
    Dim Bp1m As Short       ' BP1M data
    Dim Bp2p As Short       ' BP2P data
    Dim Bp2m As Short       ' BP2M data
    Dim Bp3p As Short       ' BP3P data
   Dim Bp3m As Short        ' BP3M data
    Dim Bp4p As Short       ' BP4P data
    Dim Bp4m As Short       ' BP4M data
End Structure

' 2-axis continuous interpolation
Structure DATA_2CIP_MC8500P
    Dim Cmd As Short            ‘ Command number (Set one of CMD_IP_2ST, CMD_IP_CW, CMD_IP_CCW.)
    Dim Speed As Integer        ' Speed (When changing the speed, set 1~4,000,000. When not changing, set 0.)
   Dim EndP1 As Integer         ' Finishing point (The first axis)
```

```
        Dim EndP2 As Integer          ' Finishing point (The second axis)
        Dim Center1 As Integer        ' Circular center point (The first axis)
        Dim Center2 As Integer        ' Circular center point (The second axis)
      End Structure                   ' Note: Specify interpolation axis in interpolation mode setting (2A)H.

' 3-axis continuous interpolation
    Structure DATA_3CIP_MC8500P
        Dim EndP1 As Integer      ' Finishing point (The first axis)
        Dim EndP2 As Integer      ' Finishing point (The second axis)
        Dim EndP3 As Integer      ' Finishing point (The third axis)
        Dim Speed As Integer      ' Speed (When changing the speed, set 1~4,000,000. When not changing, set 0.)
    End Structure                 ' Note: Specify interpolation axis in interpolation mode setting (2A)H.

' 4-axis continuous interpolation
    Structure DATA_4CIP_MC8500P
        Dim EndP1 As Integer      ' Finishing point (The first axis)
        Dim EndP2 As Integer      ' Finishing point (The second axis)
        Dim EndP3 As Integer      ' Finishing point (The third axis)
        Dim EndP4 As Integer      ' Finishing point (The fourth axis)
        Dim Speed As Integer      ' Speed (When changing the speed, set 1~4,000,000. When not changing, set 0.)
    End Structure                 ' Note: Specify interpolation axis in interpolation mode setting (2A)H.

'   Interpolation mode setting
    Structure DATA_IPMODE
        Dim Cxiv As Boolean       ' Interpolation axis change in circular interpolation (TURE: Change, FALSE: Not
      change）
        Dim Lmdf As Boolean       ' Short axis pulse equalization (TRUE: Enable, FALSE: Disable)
        Dim Vcnst As UShort       ' Constant vector speed (0:None, 1:2-axis simple, 2:3-axis simple, 3:2-axis high
      accuracy)
    End Structure

‘Helical interpolation
    Structure DATA_HL
        Dim Direction As UShort     ' Rotation direction Rotation direction (Set HL_DIR_CW: CW circular interpolation
                                      or HL_DIR_CCW: CCW circular interpolation.)
        Dim Speed       As Integer        ' Speed (1~2,000,000)
        Dim CenterX As Integer    ' Circular center point (X-axis)
        Dim CenterY As Integer    ' Circular center point (Y-axis)
        Dim EndPX       As Integer    ' Finish point (X-axis)
        Dim EndPY       As Integer    ' Finish point (Y-axis)
        Dim MoveZ       As Integer    ' Moving amount (Z-axis) (0:Z-axis is not used. other than 0: Moving amount of
      Z-axis)
        Dim MoveU       As Integer    ' Moving amount (U-axis) (0: U-axis is not used. other than 0: Moving amount of
      U-axis)
        Dim HlNum       As UShort     ' Helical rotation number (0~65,535)
        Dim HlValue As Integer    ' Helical calculation value (0: Find helical calculation value before executing
                                      interpolation. 0 or more: Execute interpolation with the calculation value which is set
        Dim IpMode      As DATA_IPMODE    ' Interpolation mode setting.
    End Structure
```

③C#

```
    // 2-axis BP interpolation
    [StructLayout(LayoutKind.Sequential)]
    public struct DATA_2BP
    {
            public      DATA_2BP(ushort bp1p,ushort bp1m,ushort bp2p,ushort bp2m)
            {
                    this.Bp1p  = bp1p;
```

```
                        this.Bp1m = bp1m;
                        this.Bp2p  = bp2p;
                        this.Bp2m = bp2m;
                }
                public  ushort  Bp1p;   // BP1P data
                public  ushort  Bp1m;   // BP1M data
                public  ushort  Bp2p;   // BP2P data
                public  ushort  Bp2m;   // BP2M data
        }

        // 3-axis BP interpolation
        [StructLayout(LayoutKind.Sequential)]
        public struct DATA_3BP
        {
                public  DATA_3BP(ushort bp1p,ushort bp1m,ushort bp2p,ushort bp2m,ushort bp3p,ushort bp3m)
                {
                        this.Bp1p  = bp1p;
                        this.Bp1m = bp1m;
                        this.Bp2p  = bp2p;
                        this.Bp2m = bp2m;
                        this.Bp3p  = bp3p;
                        this.Bp3m = bp3m;
                }
                public  ushort  Bp1p;   // BP1P data
                public  ushort  Bp1m;   // BP1M data
                public  ushort  Bp2p;   // BP2P data
                public  ushort  Bp2m;   // BP2M data
                public  ushort  Bp3p;   // BP3P data
                public  ushort  Bp3m;   // BP3M data
        }

       // 4-axis BP interpolation
        [StructLayout(LayoutKind.Sequential)]
        public struct DATA_4BP
        {
                public  DATA_4BP(ushort bp1p,ushort bp1m,ushort bp2p,ushort bp2m,ushort bp3p, ushort
bp4p,ushort bp3m)
                {
                        this.Bp1p  = bp1p;
                        this.Bp1m = bp1m;
                        this.Bp2p  = bp2p;
                        this.Bp2m = bp2m;
                        this.Bp3p  = bp3p;
                        this.Bp3m = bp3m;
                        this.Bp4p  = bp4p;
                        this.Bp4m = bp4m;
                }
                public  ushort  Bp1p;   // BP1P data
                public  ushort  Bp1m;   // BP1M data
                public  ushort  Bp2p;   // BP2P data
                public  ushort  Bp2m;   // BP2M data
                public  ushort  Bp3p;   // BP3P data
                public  ushort  Bp3m;   // BP3M data
                public  ushort  Bp4p;   // BP4P data
                public  ushort  Bp4m;   // BP4M data
        }

// 2-axis continuous interpolation
        [StructLayout(LayoutKind.Sequential)]
```

```
        public struct DATA_2CIP_MC8500P
        {
                public  DATA_2CIP_MC8500P(ushort command,ushort speed,int endp1,int endp2,int center1,int
center2)
                {
                        this.Command    = command;
                        this.Speed = speed;
                        this.EndP1= endp1;
                        this.EndP2= endp2;
                        this.Center1    = center1;
                        this.Center2    = center2;
                }
                public  ushort  Command;        // Command number (Set one of CMD_IP_2ST, CMD_IP_CW,
CMD_IP_CCW.)
                public  ushort  Speed;          // Speed (When changing the speed, set 1~4,000,000. When
not changing, set 0.)
                public  int     EndP1;          // Finishing point (The first axis)
                public  int     EndP2;          // Finishing point (The second axis)
                public  int     Center1;        // Circular center point (The first axis)
                public  int     Center2;        // Circular center point (The second axis)
        }                                       // Note: Specify interpolation axis in interpolation mode setting
(2A)H.

        // 3-axis continuous interpolation
        [StructLayout(LayoutKind.Sequential)]
        public struct DATA_3CIP_MC8500P
        {
                public  DATA_3CIP_MC8500P(int endp1,int endp2,int endp3,ushort speed)
                {
                        this.EndP1= endp1;
                        this.EndP2= endp2;
                        this.EndP3= endp3;
                        this.Speed = speed;
                }
                public  int     EndP1;          // Finishing point (The first axis)
                public  int     EndP2;          // Finishing point (The second axis)
                public  int     EndP3;          // Finishing point (The third axis)
                public  ushort  Speed;          // Speed (When changing the speed, set 1~4,000,000. When not
changing, set 0.)
        }                                       // Note: Specify interpolation axis in interpolation mode setting
(2A)H.

      // 4-axis BP interpolation
        [StructLayout(LayoutKind.Sequential)]
        public struct DATA_4CIP_MC8500P
        {
                public  DATA_4CIP_MC8500P(int endp1,int endp2,int endp3,int endp4, ushort speed)
                {
                        this.EndP1= endp1;
                        this.EndP2= endp2;
                        this.EndP3= endp3;
                        this.EndP4= endp4;
                        this.Speed = speed;
                }
                public  int     EndP1;          // Finishing point (The first axis)
                public  int     EndP2;          // Finishing point (The second axis)
                public  int     EndP3;          // Finishing point (The third axis)
                public  int     EndP4;          // Finishing point (The fourth axis)
                public  ushort  Speed;          // Speed (When changing the speed, set 1~4,000,000. When not
```

```
changing, set 0.)        }                                    // Note: Specify interpolation axis in interpolation mode setting
(2A)H.

// Interpolation mode setting
        [StructLayout(LayoutKind.Sequential)]
        public struct DATA_IPMODE
        {
                public DATA_IPMODE(bool cxiv, bool lmdf, ushort vcnst)
                {
                        this.Cxiv = cxiv;
                        this.Lmdf = lmdf;
                        this.Vcnst= vcnst;
            }
             public bool        Cxiv;              // Interpolation axis change in circular interpolation (TURE: Change, FALSE: Not
change)
                public bool        Lmdf            // Short axis pulse equalization (TRUE: Enable, FALSE: Disable)
                public ushort      Vcnst;          // Constant vector speed (0:None, 1:2-axis simple, 2:3-axis simple, 3:2-axis high
accuracy)
      }

          // Helical interpolation
         [StructLayout(LayoutKind.Sequential)]
         public struct DATA_HL
         {
                public DATA_HL(ushort direction, int speed, int centerX, int centerY, int endPX, int endPY, int moveZ, int
                        moveU, ushort hlNum, int hlValue, DATA_IPMODE ipMode)
         {
              this.Direction =       direction;
              this.Speed =            speed;
              this.CenterX =          centerX;
              this.CenterY =          centerY;
              this.EndPX =             endPX;
              this.EndPY =             endPY;
              this.MoveZ =             moveZ;
              this.MoveU =             moveU;
              this.HlNum =             hlNum;
              this.HlValue =          hlValue;
              this.IpMode =            ipMode;
         }
         public ushort    Direction;    // Rotation direction Rotation direction (Set HL_DIR_CW: CW circular interpolation or
                                          HL_DIR_CCW: CCW circular interpolation.)
         public int       Speed;         // Speed (1~2,000,000)
         public int       CenterX;       // Circular center point (X-axis)
         public int       CenterY;       // Circular center point (Y-axis)
         public int       EndPX;          // Finish point (X-axis)
         public int       EndPY;          // Finish point (Y-axis)
         public int       MoveZ;          // Moving amount (Z-axis) (0:Z-axis is not used. other than 0: Moving amount of Z-axis)
         public int       MoveU;          // Moving amount (U-axis) (0:U-axis is not used. other than 0: Moving amount of
U-axis)
         public ushort    HlNum;          // Helical rotation number (0~65,535)
         public int       HlValue;        // Helical calculation value
                                          // (0: Find helical calculation value before executing interpolation. 0 or more: Execute
                                           interpolation with the calculation value which is set.)
         public DATA_IPMODE          IpMode;              // Interpolation mode setting.
      }
```

Examples of structure are as follows:

Example 1) Not define and initialize at the same time

```
DATA_2BP [] Data2Bp = new DATA_2BP[4];          // 2-axis BP interpolation

// Interpolation data setting
Data2Bp[0].Bp1p = 0xFF30;    // 1111 1111 0011 0000   BP1 + direction      10 pulse
Data2Bp[0].Bp1m = 0;         // 0000 0000 0000 0000   BP1 - direction       0 pulse
Data2Bp[0].Bp2p = 0;         // 0000 0000 0000 0000   BP2 + direction       0 pulse
Data2Bp[0].Bp2m = 0x84FF;    // 1000 0100 1111 1111   BP2 - direction      10 pulse

Data2Bp[1].Bp1p = 0xAC35;    // 1010 1100 0011 0101   BP1 + direction       8 pulse
Data2Bp[1].Bp1m = 0;         // 0000 0000 0000 0000   BP1 - direction       0 pulse
Data2Bp[1].Bp2p = 0xC000;    // 1100 0000 0000 0000   BP2 + direction       2 pulse
Data2Bp[1].Bp2m = 0x36E7;    // 0011 0110 1110 0111   BP2 - direction      10 pulse

Data2Bp[2].Bp1p = 0x3F3F;    // 0011 1111 0011 1111   BP1 + direction      12 pulse
Data2Bp[2].Bp1m = 0xC000;    // 1100 0000 0000 0000   BP1 - direction       2 pulse
Data2Bp[2].Bp2p = 0xFBDA;    // 1111 1011 1101 1010   BP2 + direction      12 pulse
Data2Bp[2].Bp2m = 0;         // 0000 0000 0000 0000   BP2 - direction       0 pulse

Data2Bp[3].Bp1p = 0;         // 0000 0000 0000 0000   BP1 + direction       0 pulse
Data2Bp[3].Bp1m = 0x1CF2;    // 0001 1100 1111 0010   BP1 - direction       8 pulse
Data2Bp[3].Bp2p = 0xFFFF;    // 1111 1111 1111 1111   BP2 + direction      16 pulse
Data2Bp[3].Bp2m = 0;         // 0000 0000 0000 0000   BP2 - direction       0 pulse
```

Example 2) Define and initialize at the same time

```
// 2-aixs BP interpolation data      BP1P,    BP1M,    BP2P,    BP2M (Data for chapter 3.4.8 in MCX514 Manual)
DATA_2BP[] Data2Bp = new DATA_2BP[]
{
        new DATA_2BP(0xFFE4, 0x0000, 0x0000, 0x03FF),
        new DATA_2BP(0x03FF, 0x4000, 0xFFD0, 0x0000),
        new DATA_2BP(0x0000, 0xFFFF, 0x4AAB, 0x0000),
        new DATA_2BP(0xFE80, 0x000F, 0x1FFF, 0x0000),
        new DATA_2BP(0x97FF, 0x8000, 0x0000, 0x7F20),
};
```

(4) The interpolation axis (IpAxis) specified by the interpolation function is as follows:
Specify interpolation axis from X, Y, Z and U axis.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0	U-EN	Z-EN	Y-EN	X-EN

◆Description of each bit

D3～0 U-EN～X-EN Specify interpolation axis from the following table.
0：Not used as interpolation axis, 1：Used as interpolation axis

Axis	Bit to specify.
X	D0
Y	D1
Z	D2
U	D3

Main axis should be specified in order of X-EN>Y-EN>Z-EN>U-EN. The axis of which bit is set 1 is selected.

(5) About the speed change when continuous interpolation function is executed.
Continuous interpolation function can change the speed during interpolation driving. The user can set the speed to each segment.
To change the speed during interpolation driving, set TRUE (True) to the function parameter SpdChgFlg.

◆ How to change the speed for each segment
Set the speed of each segment to the Speed of DATA_2CIP_MC8500P, the speed of DATA_3CIP_MC8500P, the speed ofDATA_4CIP_MC8500P.
・When set the different speed from the previous segment, set 1~4,000,000.
・When set the same speed as the previous segment, set 0.

(6) Address assignment

The address of Nmc_OutPort or Nmc_InPort function should be assigned I/O address described in each board manual.
I/O addresses such as Write register or Read register are as follows. See each board manual for more details.

Board	Address of Write register	Address of Read register
MC8541P / MC8541Pe	0~7	0~7
MC8581P / MC8581Pe	0~F	0~F

Note1: The address is given in hexadecimal.
Note2: It is easy to access the write or read register by using not Nmc_OutPort or Nmc_InPort function but the functions for write or read registers.

(7) IC number assignment

①When the board has one IC, set 0.
②When the board has 2 ICs, set the following value.
・IC-A: 0
・IC-B: 1

The number of IC chips and the assignment of IC number for each board are shown in the table below.

Board	Number of IC	IC number
MC8541P / MC8541Pe	1	0
MC8581P / MC8581Pe	2	0,1

Note: IC indicates MCX514.

5.1.4 Usage

■ API Function Declaration

API function declaration is defined in the following files.

VC++	: MC8000P_DLL.h
VB.NET	: MC8000P_DLL.vb
C#	: Refer to User's Manual or IntelliSense.

■ Usage

(1) Start process・・・Execute Nmc_Open once before using each function.
(2) End process・・・Execute Nmc_Close or Nmc_CloseAll at the end of program.

■ Board Number

The board number specified from an application should be as follows.

	Rotary switch value of board	Board number should be specified (decimal number)
1	0~9	0~9
2	A~F	10~15

■ Notes for Use of Function

(1) About VC, VB, C# (all languages)

① When each function is used before executing Nmc_Open function, operation is not guaranteed.
② When the board number, which is not connected, is assigned, the operation of each function is not guaranteed
③ Do not access the one board from 2 or more applications at the same time (such as Nmc_Open).
④ When other than Nmc_Open function is executed after executing Nmc_Close(Nmc_CloseAll) function. Operation is not guaranteed.

(2) VC, C# only

① When using the interrupt handling function, the time from the interrupt generation to user-defined function is not guaranteed by the nature of Windows.
② When the user tries to perform the interrupt, do not execute the close handling (Nmc_Close or Nmc_CloseAll) while the interrupt user-defined function (the function designated by Nmc_SetEvent) is running. Before executing the close handling, make sure that the interrupt user-defined function is finished.

■ How to handle an interrupt by VC

①Set the interrupt by using Nmc_Open function.

```
Nmc_Open(No, TRUE);          // The second argument. TRUE: interrupt, FALSE: not interrupt
```

②Set the user-defined function handling an interrupt by using Nmc_SetEvent function, and set which interrupt is allowed or not.

```
Nmc_SetEvent(No, MC_EventProc, IpParam);          // Set the user function address and argument.
Nmc_WriteReg(No, IcNo, AXIS_ALL, 0x8000);         // Interrupt occurs at the stop of the specified IC (All axes).
```

③ If an interrupt occurs, the interrupt user-defined function set by Nmc_SetEvent is called.
The interrupt user-defined function can check the interrupt factor. To read the interrupt factor of RR1, use Nmc_ReadEvent function.

■ The example of the interrupt user-defined function

```
DWORD WINAPI MC_EventProc(LPVOID lpParam)
{
        ・・・・
    long RrIrX, RrIrY, RrIrZ, RrIrU;
    Nmc_ReadEvent(No, IcNo, &RrIrX,&RrIrY,&RrIrZ,&RrIrU);
```

```
        . . . .
    return 0;
}
```

④ Use Nmc_ResetEvent to release the interrupt user-defined function. By executing this function, the user-defined function is not called if an interrupt occurs in the board.

```
Nmc_ResetEvent(No);
```

■ How to handle an interrupt by C# (C# only)

①Set the method to handle an interrupt by using MC8000P.Nmc_SetEvent function. Arguments cannot be specified. When multiple boards are used, example is as follows.

(When the board number is 0)

```
MC8000P.callback[0] = new MC8000P.UserThread(isr);              // isr is the interrupt user-defined function.
bool ret = MC8000P.Nmc_SetEvent(0, MC8000P.callback[0], param);// Board number 0 is specified to the first argument.
                                                                // Set the function address and argument.
MC8000P.Nmc_WriteReg1(0, (int)IC.A, AXIS.ALL, 0x0080);          // Generate an interrupt at the stop (All axes).
    . . .
```

(When the board number is 1)

```
MC8000P.callback[1] = new MC8000P.UserThread(isr2);             // isr is the interrupt user-defined function.
bool ret = MC8000P.Nmc_SetEvent(1, MC8000P.callback[1], param);// Board number 1 is specified to the first argument.
                                                                // Set the function address and argument.
MC8000P.Nmc_WriteReg1(1, (int)IC.A, AXIS.ALL, 0x0080);          // Generate an interrupt at the stop (All axes)
    . . .
```

②The interrupt handling function can check the interrupt factor. To read the interrupt factor of RR1, use MC8000P.Nmc_ReadEvent function.

(Interrupt user function of the board number 0)

```
static void isr(int param)
    {
        int Rr1X, Rr1Y, Rr1Z, Rr1U;
        MC8000P.Nmc_ReadEvent(0, (int)IC.A, out Rr1X, out Rr1Y, out Rr1Z, out Rr1U);
        . . . .
    }
```

(Interrupt user function of the board number 0)

```
    static void isr2(int param)
    {
        int Rr1X, Rr1Y, Rr1Z, Rr1U;
        MC8000P.Nmc_ReadEvent(1, (int)IC.A, out Rr1X, out Rr1Y, out Rr1Z, out Rr1U);
            . . . .
    }
```

③Use MC8000P.Nmc_ResetEvent method to release the interrupt handling function. By executing this function, the interrupt user-defined function method is not called if an interrupt occurs in the board.

■ Continuous Interpolation

When executing the continuous interpolation, please read the chapter “3.7 Continuous Interpolation” of MCX514user’s manual carefully and execute the process described in the chapter in the application. Continuous Interpolation Functions *1 execute some of the process by DLL. So they will be used to execute the process of the continuous interpolation. But there are some notes when using Continuous Interpolation Functions. Please note them.

*1 : Nmc_2CIPExecMC8500P, Nmc_3CIPExecMC8500P, Nmc_4CIPExecMC8500P
Nmc_2CIPExecMC8500P_BG, Nmc_3CIPExecMc8500P_BG, Nmc_4CIPExecMC8500P_BG

Notes for when using Continuous Interpolation Function:

Make sure to disable short axis pulse equalization. If enabled, unexpected operation may occur.
In Continuous interpolation function, continuous interpolation is executed by setting all the data of 1~8 segments to pre-buffer of MCX514 in advance.
Check the error and if the error occurs, function is ended. When the error is not occurred, check continuous interpolation pre-buffer stack counter (SC) of RR0, and write the data and interpolation commands of the next segment if it can be written. At this time, 6 is displayed as the writable stack counter value in this function. (Up to 6 stages of pre-buffer can be used after starting continuous interpolation drive in this function.)
This process is repeated until continuous interpolation finishes in this function. This function is not suitable to use when the user wants to execute the other process during executing this function because loop of error check and check process of writable state of next segment data are constantly performed inside DLL.
In this case, do not use continuous interpolation function but create continuous interpolation process by the application referring to MCX514 user’s manual.
About acceleration/deceleration drive in continuous interpolation, see the chapter “3.8 Acceleration / Deceleration Control in Interpolation” of MCX514 user’s manual. When using continuous interpolation function, the initial speed should be set for 4,000,000. (Don’t change the initial speed during this function is executing.) In this case the fixed speed driving mode is applied in each segment.

■The BP (Bit Pattern) Interpolation

When executing the BP interpolation, please read the chapter “3.4 The Bit Pattern Interpolation” of MCX514 user’s manual carefully and execute the process described in the chapter in the application. The BP interpolation functions *2 execute some of the process by DLL. So they will be used to execute the process of the BP interpolation. But there are some notes when using the BP interpolation functions. Please note them.

*2 : Nmc_2BPExecMC8500P, Nmc_3BPExecMC8500P, Nmc_4BPExecMC8500P
Nmc_2BPExecMC8500P_BG, Nmc_3BPExecMC8500P_BG, Nmc_4BPExecMC8500P_BG

Notes for when using the BP Interpolation Function

Make sure to disable short axis pulse equalization. If enabled, unexpected operation may occur.
In BP interpolation function, continuous interpolation is executed by setting all the data of 1~8 segments to pre-buffer of MCX514 in advance.
Check the error and if the error occurs, function is ended. When the error is not occurred, check continuous interpolation pre-buffer stack counter (SC) of RR0, and write the data and interpolation commands of the next segment if it can be written. At this time, 6 is displayed as the writable stack counter value in this function. (Up to 6 stages of pre-buffer can be used after starting continuous interpolation drive in this function.)
This process is repeated until continuous interpolation finishes in this function. This function is not suitable to use when the user wants to execute the other process during executing this function because loop of error check and check process of writable state of next segment data are constantly performed inside DLL.
In this case, do not use BP interpolation function but create BP interpolation process by the application referring to MCX514 user’s manual.
About acceleration/deceleration drive in BP interpolation, see the chapter “3.8 Acceleration / Deceleration Control in Interpolation” of MCX514 user’s manual.

■Notes for Use of Interpolation Function

(1) Concerning the following interpolation function, the user can execute only one interpolation function at once.
While executing the interpolation function, the other interpolation function cannot be executed. If executed, an error will return.

Nmc_2BPExecMC8500P Nmc_2BPExecMC8500P_BG Nmc_2CIPExecMC8500P Nmc_2CIPExecMC8500P_BG

Nmc_3BPExecMC8500P	Nmc_3BPExecMC8500P_BG	Nmc_3CIPExecMC8500P	Nmc_3CIPExecMC8500P_BG
Nmc_4BPExecMC8500P	Nmc_4BPExecMC8500P_BG	Nmc_4CIPExecMC8500P	Nmc_4CIPExecMC8500P_BG

(2) While executing the above interpolation function, do not perform the following operation.

①Execution of the interpolation command (60h～6Fh)
②Change of interpolation mode setting (2Ah)

(3) The following interpolation function is executed in the background, so that the memory for interpolation data is allocated at the start of interpolation function and then the interpolation data specified by the user is copied. Then, when the interpolation process in the background is finished, the memory will be released and the message will be sent to the user window.
Therefore, while executing the following interpolation function in the background, do not exit the application.
Then, while executing the following interpolation function in the background, do not execute the close handling (Nmc_Close or Nmc_CloseAll).
If you want to stop the execution of interpolation function, execute the interpolation stop function (Nmc_IPStop) and make sure to receive the stop message.

Nmc_2BPExecMC8500P_BG	Nmc_3BPExecMC8500P_BG	Nmc_4BPExecMC8500P_BG
Nmc_2CIPExecMC8500P_BG	Nmc_3CIPExecMC8500P_BG	Nmc_4CIPExecMC8500P_BG

■Notes for when using helical interpolation function.

(1) Concerning the following helical interpolation function, the user can execute only one helical interpolation function to one IC at once.
Don’t execute helical calculation function (Nmc_HLValueExe) during driving after executing helical interpolation function (Nmc_HLExec).
In this case, execute helical calculation function after checking that helical interpolation driving stops.
Similarly, don’t execute helical interpolation function during executing helical calculation function (Nmc_HLValueExe).

Nmc_HLValueExe Nmc_HLExec

(2) When executing the function related helical interpolation as above, interpolation mode setting (2A)H is set as the details specified by argument.

(3) Helical interpolation function (Nmc_HLExec) is executed by constant speed drive.
Therefore, when this function is executed, initial speed of X-axis is set by specified drive speed value.

(4) Don’t execute the process as follows during executing the function related helical interpolation as above.

①Execute interpolation command (60h~6Fh)
②Change interpolation mode setting (2Ah)

(5) Don’t execute continuous interpolation function as follows during executing the function related helical interpolation as above. Execute them after checking that helical calculation and helical interpolation drive finish.

Nmc_2BPExecMC8500P	Nmc_2BPExecMC8500P_BG	Nmc_2CIPExecMC8500P	Nmc_2CIPExecMC8500P_BG
Nmc_3BPExecMC8500P	Nmc_3BPExecMC8500P_BG	Nmc_3CIPExecMC8500P	Nmc_3CIPExecMC8500P_BG
Nmc_4BPExecMC8500P	Nmc_4BPExecMC8500P_BG	Nmc_4CIPExecMC8500P	Nmc_4CIPExecMC8500P_BG

■Notes for when developing multithread applications

This chapter describes the notes for developing applications which work in multithread.
In Nmc_xxx function, there are functions executing axis switching, data writing into WR6, WR7 and data reading to RR6, RR7.
Each Nmc_xxx function is as follows:

◆Functions executing axis switching

Nmc_Reset Nmc_Command Nmc_Command_IP
Nmc_WriteReg0 Nmc_WriteReg1 Nmc_WriteReg2 Nmc_WriteReg3
Nmc_ReadReg2 Nmc_ReadReg3P Nmc_ReadReg3

Nmc_Jerk Nmc_DJerk Nmc_Acc Nmc_Dec Nmc_StartSpd Nmc_Speed
Nmc_Pulse Nmc_Pulse__VB Nmc_DecP Nmc_DecP__VB Nmc_Center Nmc_Lp
Nmc_Ep Nmc_CompP Nmc_CompM Nmc_AccOfst Nmc_HomeSpd Nmc_LpMax
Nmc_RpMax Nmc_MR0 Nmc_MR1 Nmc_MR2 Nmc_MR
3Nmc_SpeedInc Nmc_Timer

Nmc_MRmMode Nmc_PIO1Mode Nmc_PIO2Mode Nmc_HMSrch1Mode Nmc_HMSrch2Mode
Nmc_FilterMode Nmc_Sync0Mode Nmc_Sync1Mode Nmc_Sync2Mode Nmc_Sync3Mode

Nmc_ReadLp Nmc_ReadEp Nmc_ReadSpeed Nmc_ReadAccDec Nmc_ReadMR0
Nmc_ReadMR1 Nmc_ReadMR2 Nmc_ReadMR3 Nmc_ReadCT Nmc_ReadWR1
Nmc_ReadWR2 Nmc_ReadWR3 Nmc_ReadMRM Nmc_ReadP1M Nmc_ReadP2M
Nmc_ReadAc Nmc_ReadStartSpd Nmc_ReadSetSpeed Nmc_ReadPulse

Nmc_2BPExecMC8500P Nmc_3BPExecMC8500P Nmc_4BPExecMC8500P
Nmc_2BPExecMC8500P_BG Nmc_3BPExecMC8500P_BG Nmc_4BPExecMC8500P_BG
Nmc_2CIPExecMC8500P Nmc_3CIPExecMC8500P Nmc_4CIPExecMC8500P
Nmc_2CIPExecMC8500P_BG Nmc_3CIPExecMC8500P_BG Nmc_4CIPExecMC8500P_BG
Nmc_HLValueExec Nmc_HLExec

Nmc_WriteRegSetAxis Nmc_ReadRegSetAxis Nmc_WriteData
Nmc_ReadData

◆Functions executing data writing into WR6, WR7

Nmc_Jerk	Nmc_DJerk	Nmc_Acc	Nmc_Dec	Nmc_StartSpd	Nmc_Speed
Nmc_Pulse	Nmc_Pulse__VB	Nmc_DecP	Nmc_DecP__VB	Nmc_Center	Nmc_Lp
Nmc_Ep	Nmc_CompP	Nmc_CompM	Nmc_AccOfst	Nmc_HomeSpd	Nmc_LpMax
Nmc_RpMax	Nmc_MR0	Nmc_MR1	Nmc_MR2	Nmc_MR3	
Nmc_SpeedInc	Nmc_Timer				

Nmc_MRmMode	Nmc_PIO1Mode	Nmc_PIO2Mode	Nmc_HMSrch1Mode	Nmc_HMSrch2Mode
Nmc_FilterMode	Nmc_Sync0Mode	Nmc_Sync1Mode	Nmc_Sync2Mode	Nmc_Sync3Mode

Nmc_2BPExecMC8500P	Nmc_3BPExecMC8500P	Nmc_4BPExecMC8500P
Nmc_2BPExecMC8500P_BG	Nmc_3BPExecMC8500P_BG	Nmc_4BPExecMC8500P_BG
Nmc_2CIPExecMC8500P	Nmc_3CIPExecMC8500P	Nmc_4CIPExecMC8500P
Nmc_2CIPExecMC8500P_BG	Nmc_3CIPExecMC8500P_BG	Nmc_4CIPExecMC8500P_BG
Nmc_HLValueExec	Nmc_HLExec	

Nmc_WriteReg6 Nmc_WriteReg7

Nmc_WriteRegSetAxis Nmc_WriteData

When writing intp WR6,WR7 by Nmc_OutPort or Nmc_WriteReg

◆Functions executing data reading to RR6, RR7

Nmc_ReadLp	Nmc_ReadEp	Nmc_ReadSpeed	Nmc_ReadAccDec	Nmc_ReadMR0	Nmc_ReadMR1
Nmc_ReadMR2	Nmc_ReadMR3	Nmc_ReadCT	Nmc_ReadTX		
Nmc_ReadCHLN	Nmc_ReadHLV	Nmc_ReadWR1	Nmc_ReadWR2	Nmc_ReadWR3	
Nmc_ReadMRM	Nmc_ReadP1M	Nmc_ReadP2M	Nmc_ReadAc	Nmc_ReadStartSpd	
Nmc_ReadSetSpeed	Nmc_ReadPulse				

Nmc_ReadData

To perform WR1~WR3 writing, RR1~RR2 reading, data writing command and data reading command, basically use the following Nmc_xxx function.

◆WR1~WR3 writing

Nmc_WriteReg1	Nmc_WriteReg2	Nmc_WriteReg3	Nmc_WriteRegSetAxis

◆RR1~RR2 reading

Nmc_ReadReg2	Nmc_ReadReg3P	Nmc_ReadReg3	Nmc_ReadRegSetAxis

◆Data writing command

Nmc_Jerk	Nmc_DJerk	Nmc_Acc	Nmc_Dec	Nmc_StartSpd
Nmc_Speed	Nmc_Pulse	Nmc_Pulse__VB	Nmc_DecP	Nmc_DecP__VB
Nmc_Center	Nmc_Lp	Nmc_Ep	Nmc_CompP	Nmc_CompM
Nmc_AccOfst	Nmc_HomeSpd	Nmc_LpMax	Nmc_RpMax	Nmc_MR0
Nmc_MR1	Nmc_MR2	Nmc_MR3	Nmc_SpeedInc	Nmc_Timer

Nmc_MRmMode	Nmc_PIO1Mode	Nmc_PIO2Mode	Nmc_HMSrch1Mode	Nmc_HMSrch2Mode
Nmc_FilterMode	Nmc_Sync0Mode	Nmc_Sync1Mode	Nmc_Sync2Mode	Nmc_Sync3Mode
Nmc_WriteData				

◆Data reading command

Nmc_ReadLp	Nmc_ReadEp	Nmc_ReadSpeed	Nmc_ReadAccDec	Nmc_ReadMR0
Nmc_ReadMR1	Nmc_ReadMR2	Nmc_ReadMR3	Nmc_ReadCT	Nmc_ReadTX
Nmc_ReadCHLN	Nmc_ReadHLV	Nmc_ReadWR1	Nmc_ReadWR2	Nmc_ReadWR3
Nmc_ReadMRM	Nmc_ReadP1M	Nmc_ReadP2M	Nmc_ReadAc	Nmc_ReadStartSpd

Nmc_ReadSetSpeed Nmc_ReadPulse
Nmc_ReadData

When trying to execute the commands such as WR1~WR3 data writing, RR1~RR2 data reading, data writing and reading, the user must take care in multithread environment if the same operations are performed without these functions.

(1) For example, when writing into WR1, Nmc_WriteReg1 is used; however, there is other way as follows:

```
① Nmc_OutPort(No, MCX_WR0,   0x011F);                    // Switch to X axis (IC-A).
② Nmc_OutPort(No, MCX_WR1,   Data);                      // Write to WR1 (IC-A).
```

Also, the following functions can perform the same operation.

```
③ Nmc_WriteReg(No, IcNo, MCX_WR0,   0x011F); // Switch to X axis.
④ Nmc_WriteReg(No, IcNo, MCX_WR1,   Data);   // Write to WR1.
```

In this case, if Nmc_xxx function to switch the axis is executed between ① and ② or ③ and ④, the data will be written into the WR1 of a different axis.

(2) For example, when setting the speed, Nmc_Speed is used; however, there is other way as follows:

```
① Nmc_OutPort( No, MCX_WR6, Data );          // Write to WR6 (IC-A).
② Nmc_OutPort( No, MCX_WR0, 0x0105 );        // Set WR6 data to the speed of X axis (IC-A).
```

Also, the following functions can perform the same operation.

```
③ Nmc_WriteReg6(No, IcNo, Data);             // Write to WR6.
④ Nmc_Command(No, IcNo, AXIS_X, 0x05);       // Set WR6 data to the speed of X axis.
```

In this case, if Nmc_xxx function to write data into WR6, WR7 is executed between ① and ② or ③ and ④, the other data will be set to the speed.

(3) For example, when reading logical position counter, Nmc_ReadLp is used; however, there is other way as follows:

```
① Nmc_OutPort( No, MCX_WR0, 0x0130 );        // Read logical position counter of X axis to RR6, RR7 (IC-A)
② d6 = Nmc_InPort( No, MCX_RR6 );            // Read from RR6 (IC-A)
③ d7 = Nmc_InPort( No, MCX_RR7 );       // Read from RR7 (IC-A)
```

In this case, if Nmc_xxx function to read data to RR6, RR7 is executed between ① and ② or ② and ③, the different data will be read.

Thus, in multithread environment, when calling API function more than twice to execute the objective operation, the user needs not to perform such an operation or needs to take exclusive control.

When the operation is finished by calling Nmc_xxx function once, it properly works in multithread environment.
Each function of Nmc_xxx takes exclusive control each other.

Also, when the used develop the application which combines multithread and interruption, avoid to generate the interrupt and access to the board at the same time.

5.2 Notes on Programming

(1) Initial setting of input signal filter

Each input signal of the board, for example, a limit signal, uses the built-in integral filter of MCX514. The device driver provided by NOVA electronics sets the filter as shown below for each input signal by writing input signal filter mode (60h) to MCX514 by default when PC is powered on.

Filter delay time: : 512μsec

Each Input Signal Filter Enable/Disable:

Signal	Enable / Disable
EMG	Enable
nLMTP, nLMTM	Enable
nSTOP0, nSTOP1	Enable
nINPOS, nALARM	Enable
nPIO6	Enable
nSTOP2	Enable
nECA, nECB	Disable

To switch Enable/Disable of these input signal filters on the application, see chapter 7.3.6 of MCX514 user's manual. It can be changed by extension mode setting command (25h). The following example shows that all ICs of all axes (X, Y, Z and U axes) of the board number 0 are set to the same setting as the table above. Nmc_ExpMode executes extension mode setting command (25h).

Example 1)

```
Nmc_FilterMode(0, 0, AXIS_ALL, 0xAA7F);    // Setting for IC-A
```

Example 2)

```
// Setting for IC-A
Nmc_WriteReg6(0, 0, 0xAA7F);
Nmc_WriteReg0(0, 0, 0x0F25);
```

Note:
①When executing command reset (00FF), the filter of each input signal is set inside the device driver same as the default when PC is powered on.

(2) PC standby mode and hibernation mode
In this device driver, the operation after standby or hibernation mode is not guaranteed.
When the user tries to access the board after standby or hibernation mode, be sure to restart PC before access

(3) Interrupt support
The user can use the interrupt in the application only developed in VC++ and C#.
Supported interrupts are as follows:

- All the interrupt reported by RR1 register.
- The interrupt that occurs when the value of stack counter changes from 8 to 7 and 4 to 3 in continuous interpolation drive bit pattern interpolation.

(4) Interrupt clearing

①The interrupt reported by RR1 register.

The interrupt is cleared after the driver read RR1, just after the interrupt occurs in the board.
Then, user-defined function of the application for interrupt is called. (Only when user-defined function has been set.)

②The interrupt that occurs when the value of stack counter changes from 8 to 7 and 4 to 3 in continuous interpolation drive bit pattern interpolation.

The interrupt is cleared inside driver, just after the interrupt occurs in the board.
Then, user-defined function of the application for interrupt is called. (Only when user-defined function has been set.)

5.3 Evaluation tool for MCX514.

MCX514 evaluation tool is used to evaluate the board equipped with MCX514 (MC8541P/MC8541Pe and MC8581P/MC8581Pe). The user can download on our website (Attached with MC8000P device driver software.).
Before executing the evaluation tool, install MC8000P device driver.

Note: This chapter describes the outline of MCX514 evaluation tool. For more details, see ReadMe.txt in Tool\MCX514 Board folder.

5.3.1 Execution Program

Execution program is MCX514.exe.
For 32bitOS is in Tool\MCX514 Board\32 Tool folder, for 64bitOS in Tool\MCX514 Board\64 Tool folder.

■Regarding MCX514.

Each evaluation tool evaluates the MCX304 described below.

●MCX514.exe

①MCX514 of the board equipped with only one MCX514 (MC8541P, MC8541Pe).
②MCX514 of the board equipped with 2 or more MCX514 (MC8581P, MC8581Pe).

■ Regarding program execution

Multiple MCX514.exe can simultaneously be executed to the multiple board. Specify the different names for these exe file before executing. (Example : MCX514-A.exe、MCX514-B.exe and so on.) Multiple MCX514.exe cannot be executed to the ICs on the same board.同

■When error occurs when activating.

When error occurs and cannot be executed in activating, process it according to the chapter [3.2.2 When error occurs in executing sample program and evaluation tool].

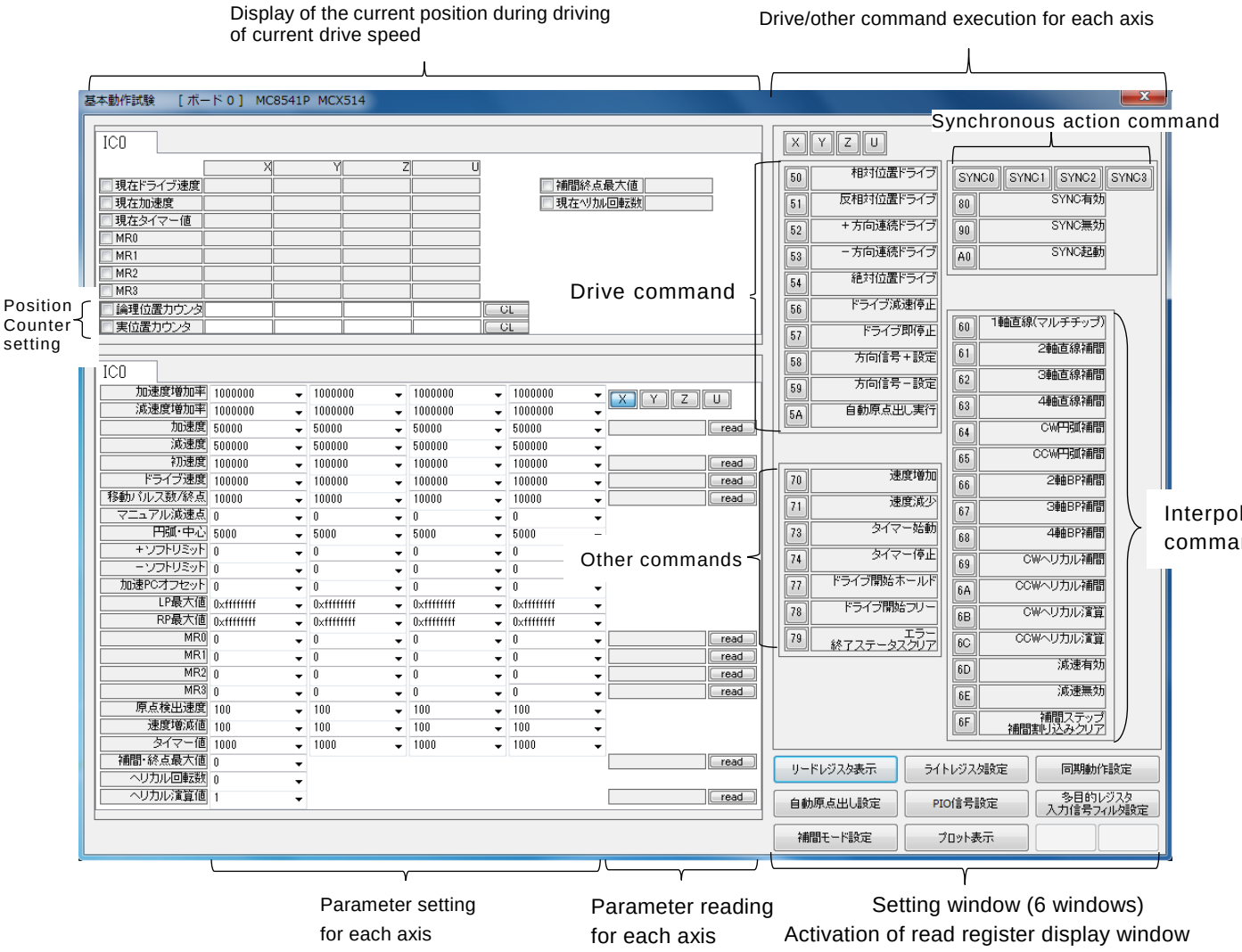
5.3.2 Function overview

Start the evaluation tool, and the Board Number Select Window appears. The user can select the board number (setting value of rotary switch (0~F) on the board).
And if the Board Name Display button is pressed, the board name will be shown at the side of the board number (only the board using this device driver). After selecting the board number, press OK button, and the Main Window appears.

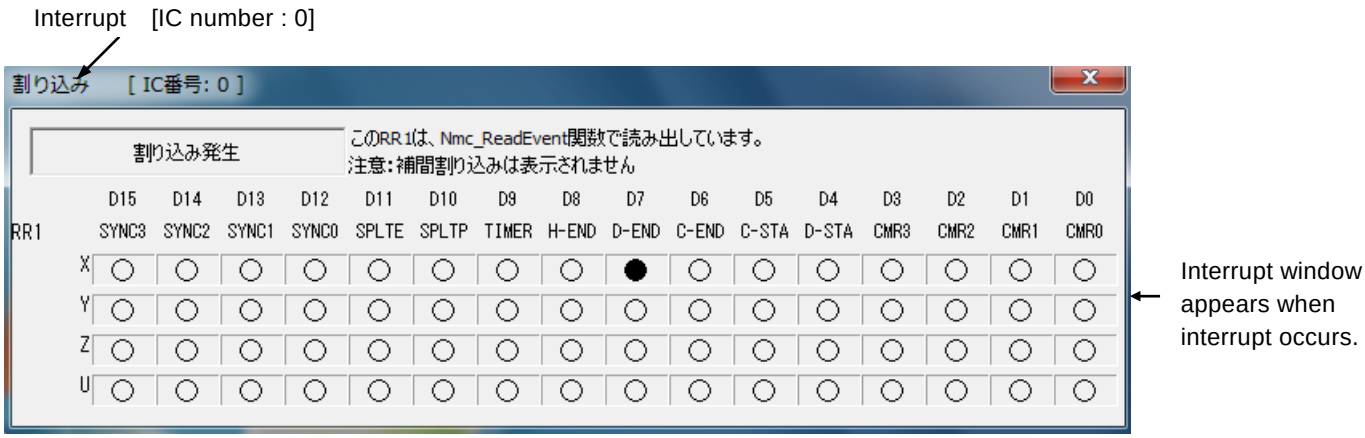
In the Main Window, the user can perform parameter settings for each axis, drive/other command execution, current position/speed display and interrupt window display. In mode setting window, (write register, synchronous action, automatic home search PIO, multi-purpose register, input signal filter and interpolation mode setting window) and read register window, mode setting and read register checking can be executed.

5.3.3 Main Window

In the main window, the user can do the following operations as shown below.

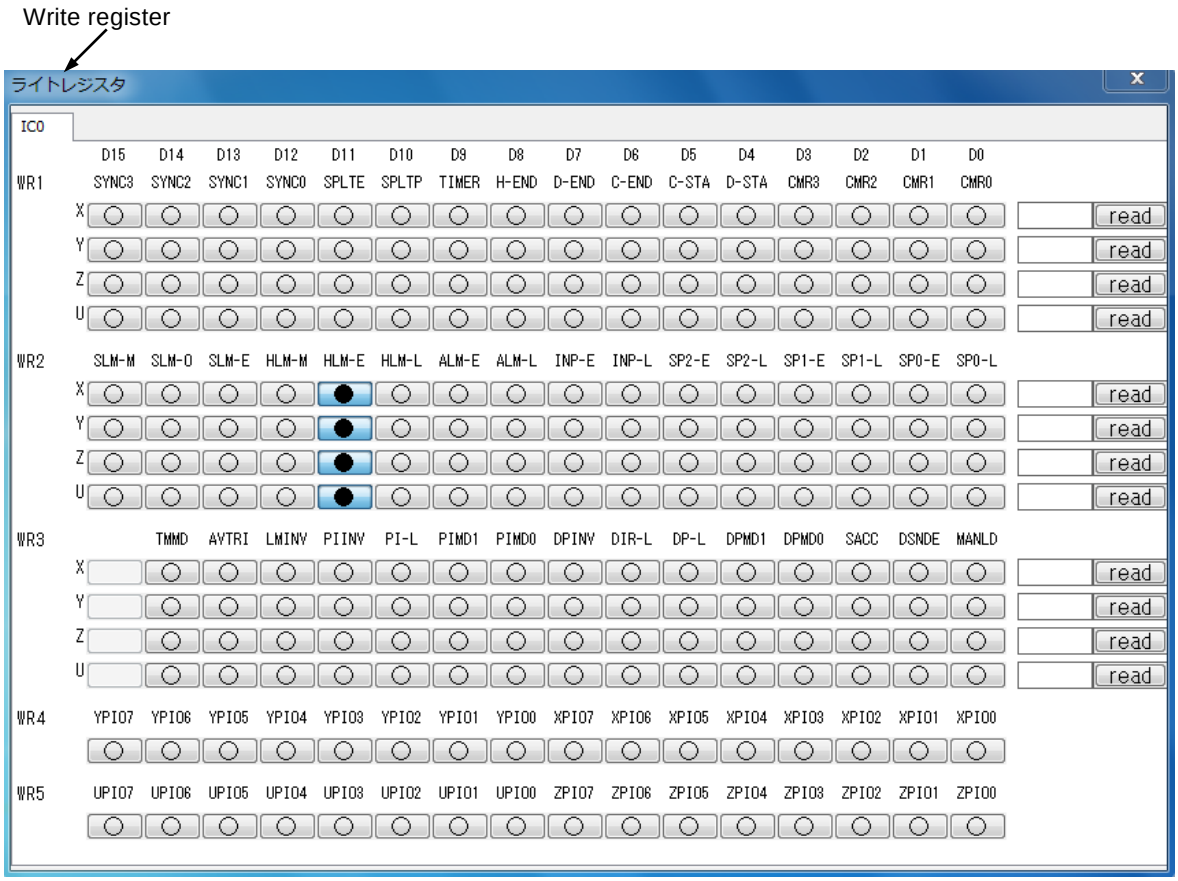


Current position and drive speed are read out and displayed at intervals by checking left side ☐ of each item.
Current drive speed of interpolation drive is displayed by reading the calculated pulse speed of the main axis.



5.3.4 Write register setting window

Set WR1~WR5 (Mode register and output register) and read WR1~WR3 (Mode register) in write register setting window.

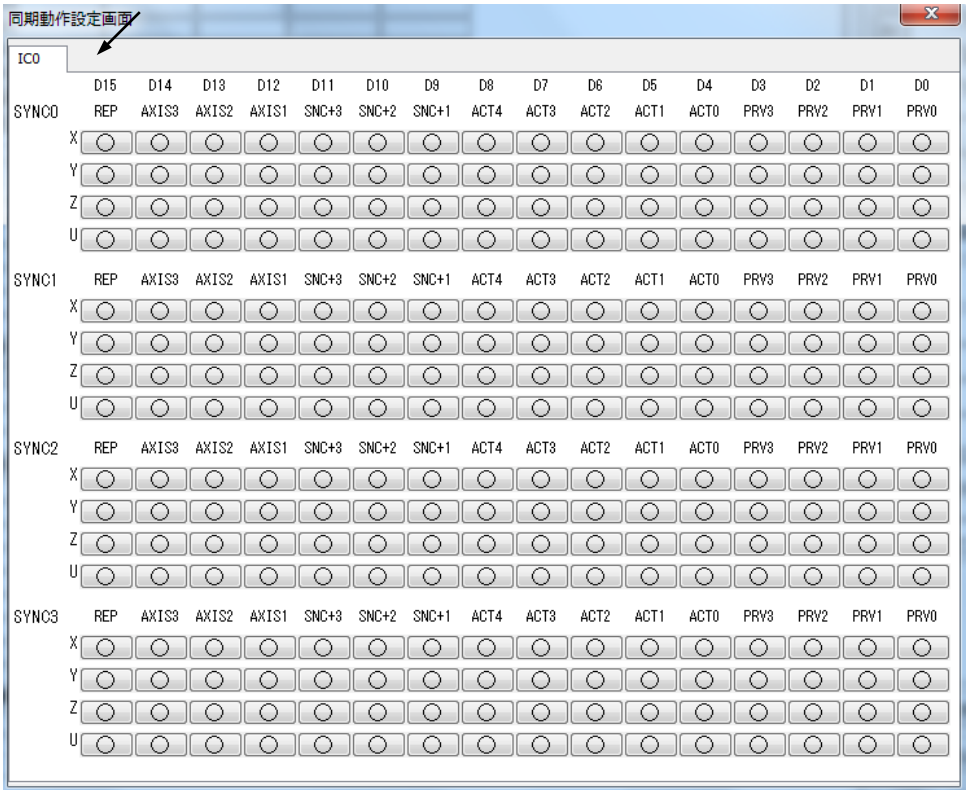


All the buttons of WR4 and WR5 can be checked. However, PIO signals of MC8541P/MC8541Pe and MC8581P/MC8581Pe which can be used are limited. Please refer to hardware user's manual to set them correctly.

5.3.5 Synchronous action setting window

Set synchronous action register (SYNC0~SYNC3) in synchronous action setting window.

Synchronous action setting window

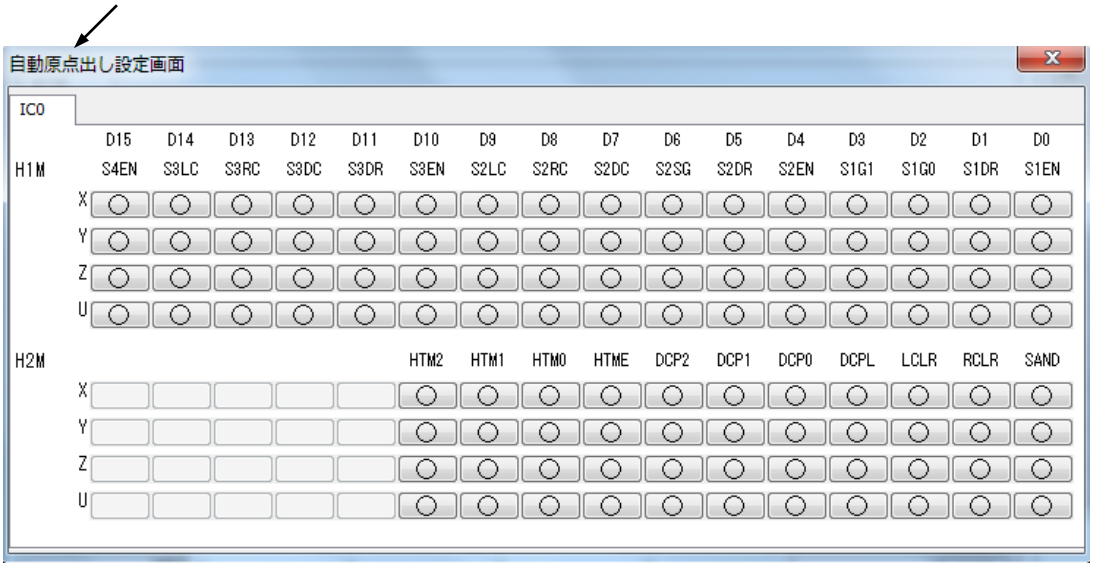


Kinds of synchronous action of MC8541P/MC8541Pe and MC8581P/MC8581Pe which can be used are limited. Please refer to hardware user's manual to set them correctly.

5.3.6 Automatic home search setting window

Set automatic home search register (H1M and H2M) in automatic home search setting window.

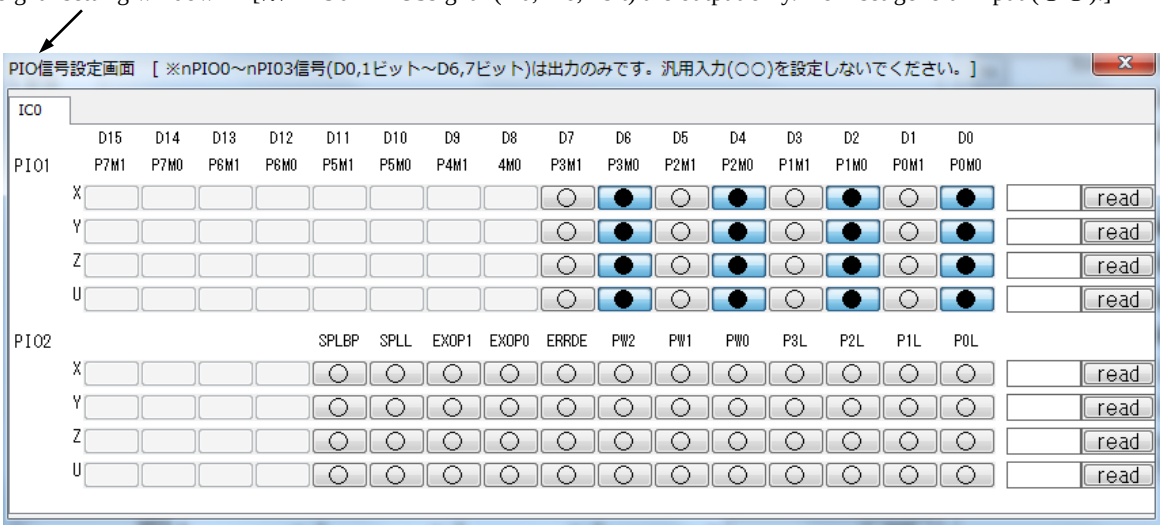
Automatic home search setting window



5.3.7 PIO signal setting window

Set and read PIO (PIO1 and PIO2) in PIO signal setting window.

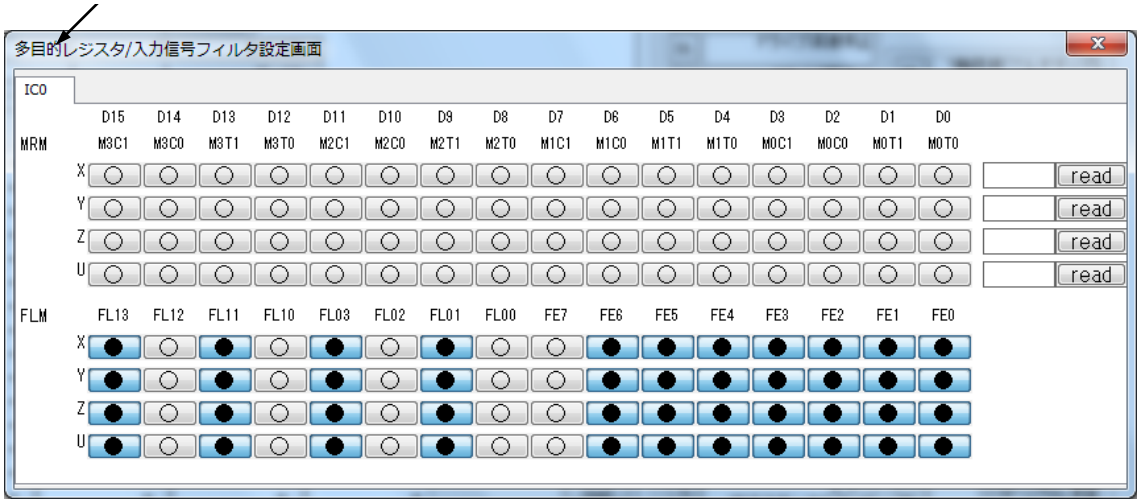
PIO signal setting window [※nPIO0~nPIO3signal (D0,1~6,7 bit) are output only. Don' set general input (○○).]



5.3.8 Multi-purpose register / input signal filter setting window

Set and read Multi-purpose register (MRM) and set input signal filter setting (FLM) in multi-purpose register / input signal filter setting window.

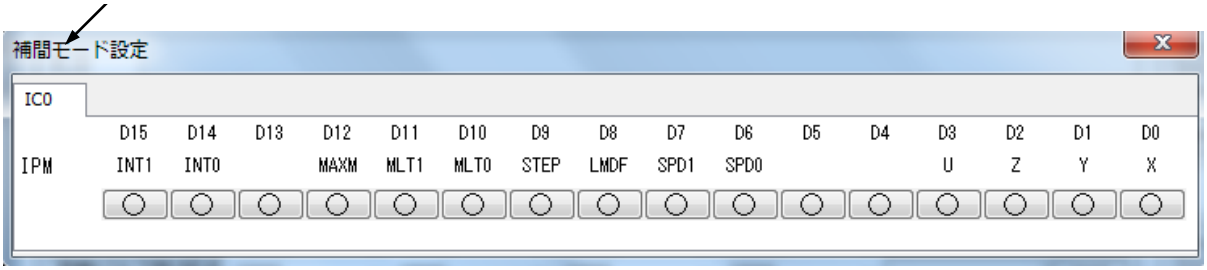
Multi-purpose register / input signal filter setting window



5.3.9 Interpolation mode setting window

Set interpolation mode (IPM) in interpolation mode setting window.

Interpolation mode setting



Please make sure to clear all the setting buttons when the evaluation of interpolation drive and then, evaluate any other function.

5.3.10 Read register window

Read and display the current value of RR0, RR2, RR3, RR4 and RR5 (Status register and PIO read register) in read regisnter window. Reading interval is 50mSEC. About RR1, display it in interrupt window when interrupt occurs.

Read register [※RR1 is displayed in interrupt window when interrupt occurs.]