PCI バス対応　高機能・4軸/8軸モーションコントロールボード

MC8541P
MC8581P

デバイスドライバ
取扱説明書

２０１６．　９．　７　　　　　　初版

NOVA electronics　　　　株式会社 ノヴァ電子

はじめに

このたびは、ノヴァ電子製ＰＣＩバス対応モータコントロールボードをご検討いただきまして、ありがとうございます。

■安全にお使いいただくために

本製品を安全にお使いいただくために､本書に記述されている内容を必ずお守りください。
なお、注意事項をお守りいただかない場合、製品の故障､瑕疵担保責任､その他一切の保証をできかねる場合があります。
本製品をご使用いただく前に、必ず本書を熟読し理解した上でご使用ください。

また、本書の記載内容は、今後、機能の向上などのため予告なしに変更する場合があります。
最新の取扱説明書､ソフトウェアは、弊社ホームページ（URL: http://www.novaelec.co.jp/ ）からダウンロードできます。

■マニュアルの併用

ボードの仕様などについては、取扱説明書を参照してください。

■本書で使用する特殊用語

ＩＣ　　ＩＣとはMCX514の事を指します。

ＩＣ－Ａ、ＩＣ－Ｂ　　ボードに複数ＩＣを搭載している場合、１つ目のＩＣを「ＩＣ－Ａ」、２つ目のＩＣを「ＩＣ－Ｂ」と表現します。ボードに搭載しているＩＣが１つの場合、そのＩＣを「ＩＣ－Ａ」と表現します。

目　次

１．概要

本デバイスドライバは、ノヴァ電子製ＰＣＩバス対応モータコントロールボードデバイスドライバです。
ボードの仕様は、ボードの取扱説明書とボードに搭載しているＩＣの取扱説明書を参照して下さい。

1.1　対応ボード

本デバイスドライバは下記のボードに対応しています。

対応ボード
MC8541P
MC8581P

注）現バージョン(デバイスドライバVer.7.2.1.0)では本ボードをインストールしたＰＣに他のMC8000Pボードはインストールできません。詳細は、1.6節を参照してください。

1.2　対応ＯＳ

本デバイスドライバは下記のＯＳに対応しています。

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

1.3　対応言語

本デバイスドライバは下記の言語に対応しています。
アプリケーションから弊社が提供するＤＬＬを呼び出す事でボードを制御できます。

対応言語
Microsoft Visual C++
Microsoft Visual Basic

注）対応しているMicrosoft Visual Studioのバージョンおよび、言語に対応する開発環境とＯＳとの関係はMicrosoftのホームページおよび当社ホームページを参照ください。

1.4　最大ボード数

本デバイスドライバは対応ボードを同時に１６枚まで認識します。

1.5　ボード搭載ＩＣ

対応ボードに搭載しているＩＣは下記の通りです。

ボード	搭載ＩＣ	ＩＣ数	軸数
MC8541P	MCX514	1	4
MC8581P	MCX514	2	8

1.6　他のMC8000Pボードとの共用について

現バージョン(デバイスドライバVer.7.2.1.0)では、本ボードをインストールするPCには、次の表にあるボードとは共用できません(同時にインストール出来ません)。次回バージョンアップ時にこれらのボードと共用できるようにする予定です。

ボード
MC8082P / MC8082Pe
MC8042P
MC8022P
MC8080P
MC8043P / MC8043Pe

２．インストール

この章では、本ボードのパソコンへの組込みとデバイスドライバのインストール方法について説明します。

2.1　ドライバソフトウェアの準備

ＣＤ－ＲＯＭからデバイスドライバをインストールする場合は、MC8541P_8581PデバイスドライバのＣＤ－ＲＯＭを用意して下さい。
ホームページからダウンロードしたデバイスドライバをインストールする場合は、ダウンロードしたソフトウェアを解凍して下さい。

2.2　本ボードを複数枚使用する場合の設定

本デバイスドライバは本ボードを同時に１６枚まで認識します。

本ボードを１つのシステム（ＰＣ）で複数枚使用する時は、ボードを個別に認識させる為に、２枚目以降のボードはボード番号をボード上のロータリースイッチで設定して下さい。
ロータリースイッチ（ＳＷ１）の位置は、ボードの取扱説明書「基板外形」の章を参照してください。
ロータリースイッチは０～Ｆのいずれかを設定できます。ロータリースイッチの番号は他のボードと重複しないように設定して下さい。

出荷時は、ボード上のロータリースイッチに０が設定されています。

2.3　パソコンへの本ボードの組込み

注意：パソコンへの取り付け作業は必ずパソコンの電源を切断してから行ってください。さもないと回路素子を破壊する原因となります。

① パソコン本体の電源がＯＦＦであることを確認してから、外装カバー、スロットカバー等を外します。
② 空いている拡張スロットへ本製品を差し込みます。基板のエッジコネクタをパソコンのＰＣＩバスコネクタに正しく挿入してください。
③ 取付金具をネジ止めしてください。この時キチンとねじを締めないと後で抜け落ちたりするなどして、ショートや故障、誤動作の原因となります。
④ パソコン本体の外装カバーを元通りに取り付けます。

2.4　デバイスドライバのインストール

インストール手順を説明します。

2.4.1　他のMC8000Pボードがインストールされている場合

1.6節にあるように、現バージョン(デバイスドライバVer.7.2.1.0)では、本ボードは他のMC8000P対応ボード（**注１**）と共用できません。本ボードをインストールするPCに他のMC8000P対応ボードがインストールされている場合は、本ボードをインストールする前に、必ず他のMC8000P対応ボードのアンインストールを行ってください。アンインストールの手順については、2.5節を参照してください。このとき、アンインストーラーの実行ファイルは、インストールしているMC8000P対応ボードの実行ファイル「uninstall XXXX INF.exe」を使用して行ってください。(XXXX にはお使いのモデル名が入ります)

注１：他のMC8000P対応ボードとは、MC8082P、MC8042P、MC8022P、MC8080P、MC8043P、MC8082Pe、MC8043Peを示します。

2.4.2　インストール手順

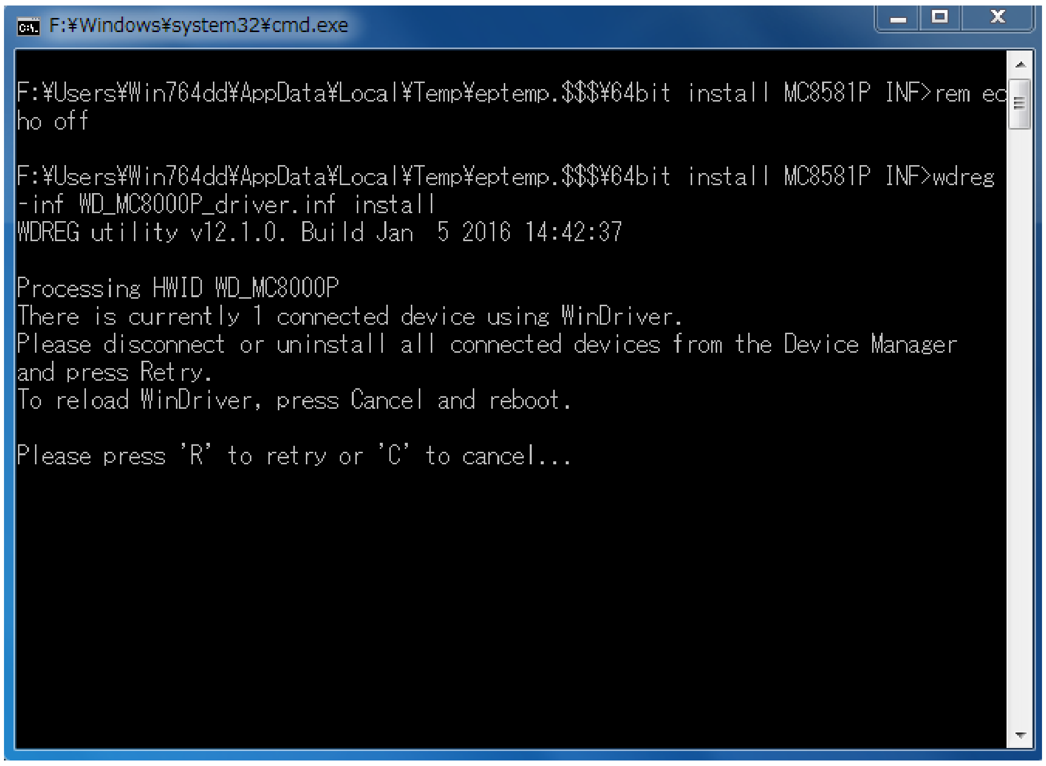
起動中の他のアプリケーションは終了させてください。
デバイスドライバのインストールは必ずアドミニストレーター権限をもったユーザーログインで行ってください。アドミニストレーター権限以外でインストールをした場合正常にインストールされません。
下記の手順はWindows 7(64bit)を例にしていますが、Windows 7(32bit)やWindows 8.1、Windows 10でもインストールの進め方は同じです。

①　2.1の方法でインストールするデバイスドライバを準備して下さい。
②　2.2, 2.3 を実行し、本ボードが確実にパソコンに組み込まれているか確認してください。
③　パソコン本体の電源をＯＮし、Windows 7を起動します。
④　アドミニストレーター権限を持ったユーザーでログインしてください。
⑤　提供ＣＤ－ＲＯＭのDriverフォルダ（提供ＣＤ－ＲＯＭがＤドライブにある場合は、D:¥Driver）、あるいはダウンロードしたソフトウェアのDriverフォルダを選択し、フォルダ内の「64bit install XXXX INF.exe」をダブルクリックしてください。
(32bitの場合は「install XXXX INF.exe」をダブルクリックしてください。)
(XXXX にはお使いのモデル名が入ります)

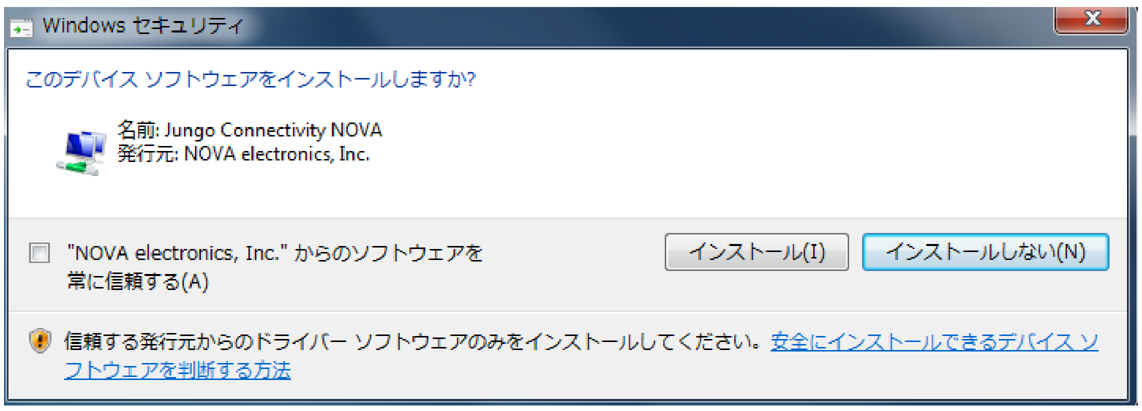
⑥　”ユーザーアカウント制御” の画面が出てきた場合は、【許可】ボタンをクリックして操作を進めてください。
(画面上のMC8541Pはお使いのモデル名に置き換えてご覧下さい。)

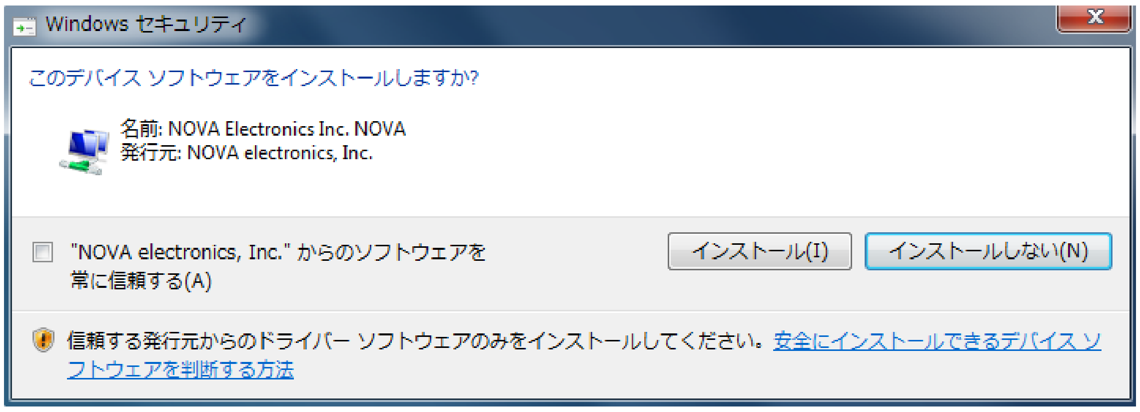
⑦　デバイスドライバのインストール画面
【次へ】のボタンをクリック　（画面上のMC8541Pはお使いのモデル名に置き換えてご覧下さい。）

⑧　既に別のボードがインストールされている場合、次のような画面が表示されます。
キーボードで[C]を入力します。

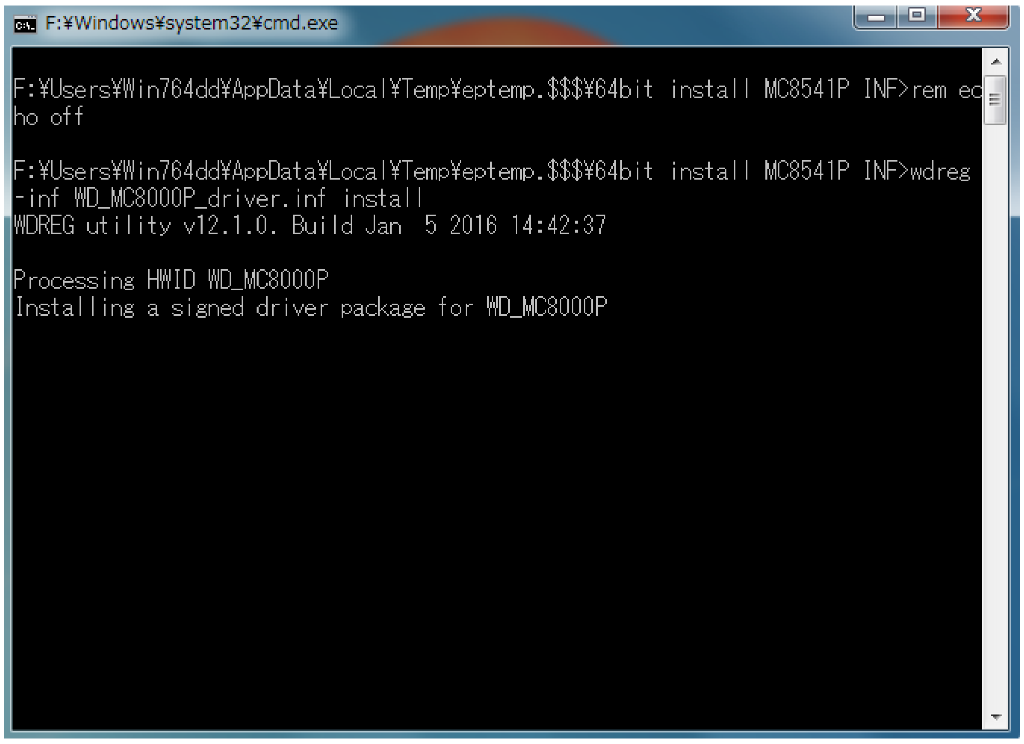


⑨　【インストール】をクリックします。

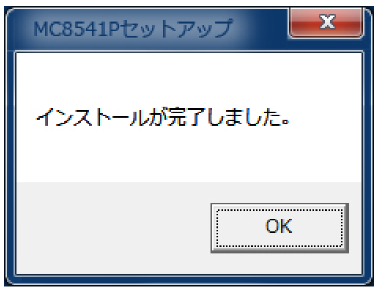




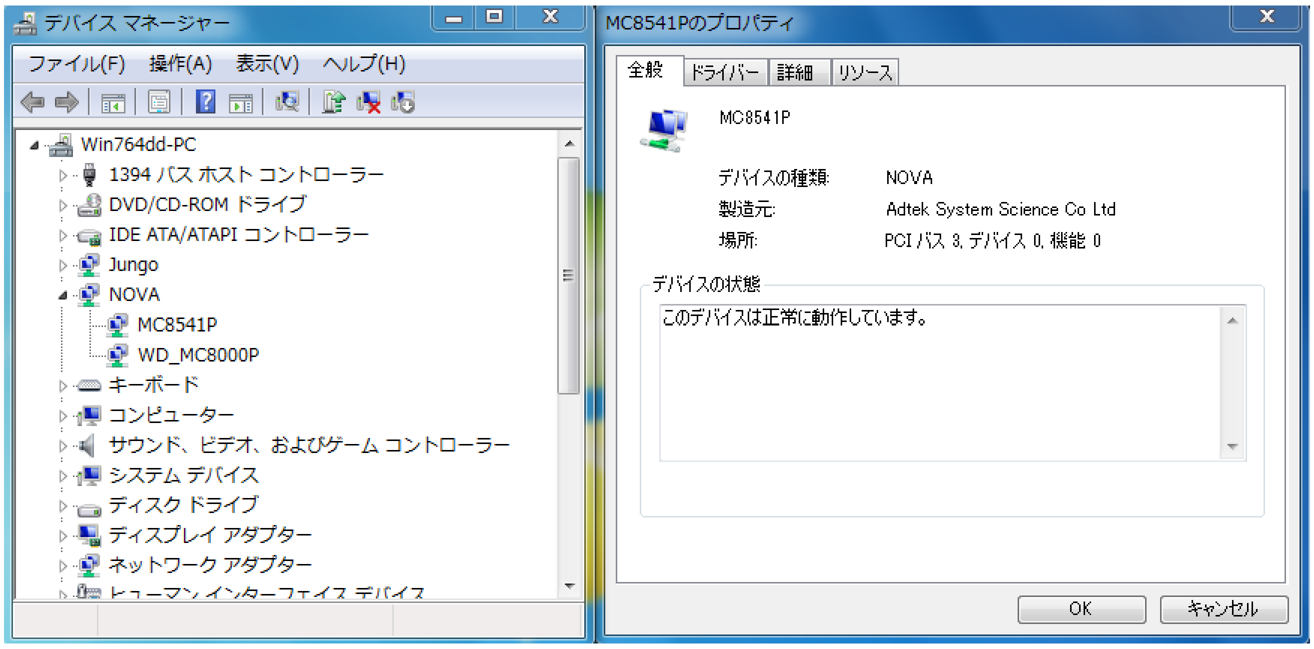
⑩　処理が終了すると自動的に画面が閉じられます。（インストールに時間がかかることがあります。完了のメッセージがあるまでお待ちください。）



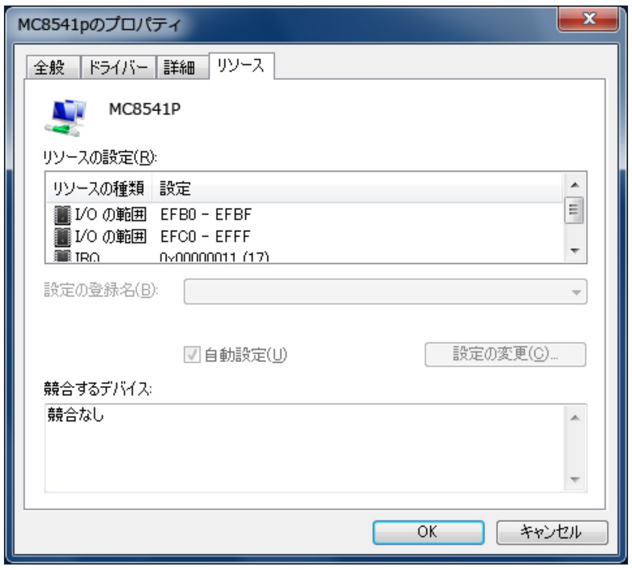
⑪　【ＯＫ】ボタンをクリックしてデバイスドライバのインストールは完了です。



⑫　以上でデバイスドライバのインストールは完了です。
次の方法で正しくインストールされたかどうかを確認して下さい。
「コントロールパネル」－「システムとメンテナンス」－「デバイスマネージャ」画面（下記左画面）を開き、「ＮＯＶＡ」の下のボード名（MC xxxx P など）をダブルクリックし、「全般」タブで下記右画面を表示します。
この画面でデバイスの状態に「このデバイスは正常に動作しています」と記載されていたらインストールは正常終了です。
（画面上のMC8541Pはお使いのモデル名に置き換えてご覧下さい。）



インストールを完了したらリソース（I／Oアドレス、割り込みレベル）の設定､競合の有無を［コントロールパネル］－［システムとメンテナンス］－［デバイスマネージャ］でボード名をダブルクリックし、［リソース］タブで確認してください。

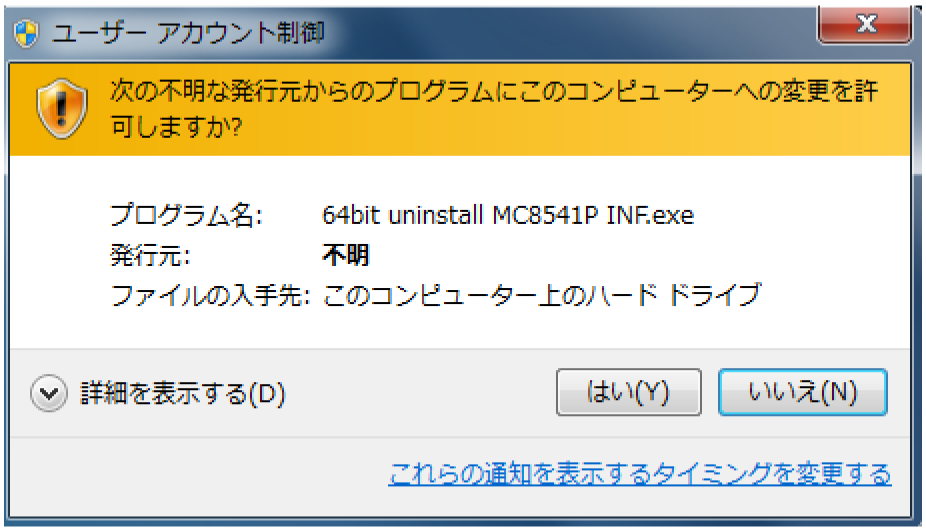


2.5 取り外し

ボードをPCから外す場合、外す前に必ず下記の手順にそって、デバイスドライバを削除してください。
下記の手順はWindows 7(64bit)を例にしていますが、Windows 7(32bit)やWindows 8.1、Windows 10でもインストールの進め方は同じです。

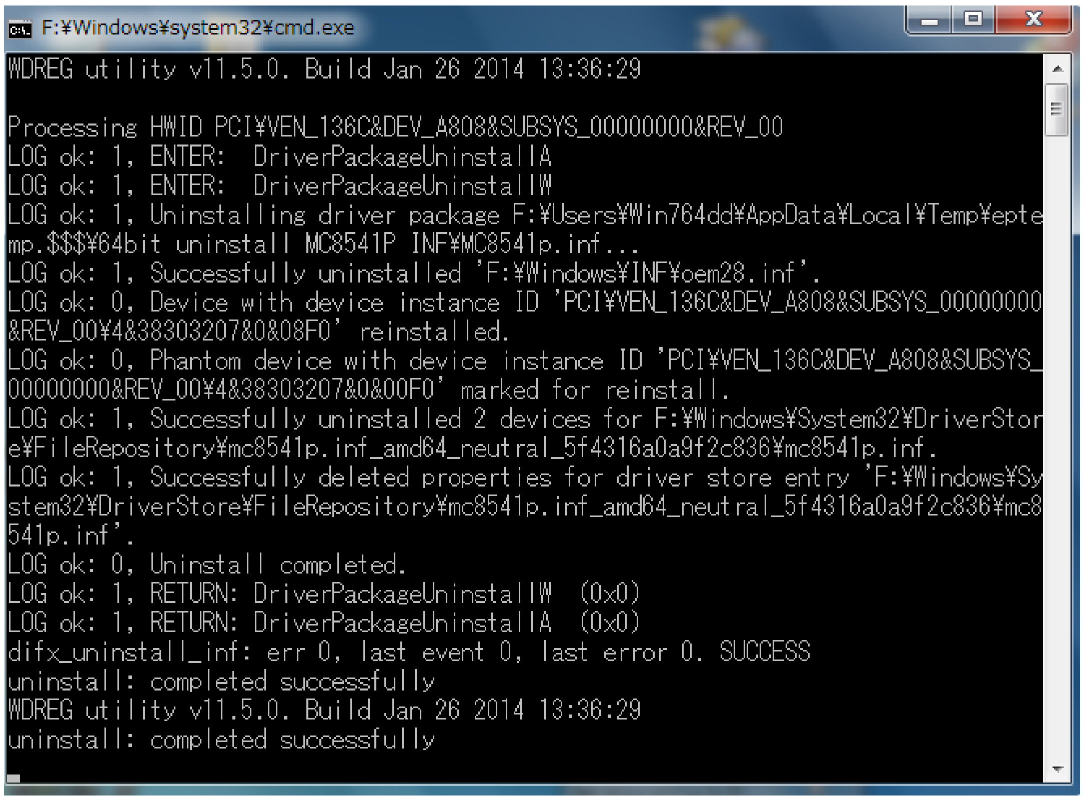
① 提供ＣＤ－ＲＯＭのDriverフォルダ（提供ＣＤ－ＲＯＭがＤドライブにある場合は、D:¥Driver）、あるいはダウンロードしたソフトウェアのDriverフォルダを選択し、フォルダ内の「64bit uninstall XXXX INF.exe」をダブルクリックしてください。
（32bitの場合は「uninstall XXXX INF.exe」をダブルクリックしてください。）
（画面上のMC8541Pはお使いのモデル名に置き換えてご覧下さい。）

② ”ユーザーアカウント制御” の画面が出てきた場合は、【許可】ボタンをクリックして操作を進めてください。
（画面上のMC8541Pはお使いのモデル名に置き換えてご覧下さい。）



③ デバイスドライバのアンインストール画面
【次へ】のボタンをクリック （画面上のMC8541Pはお使いのモデル名に置き換えてご覧下さい。）

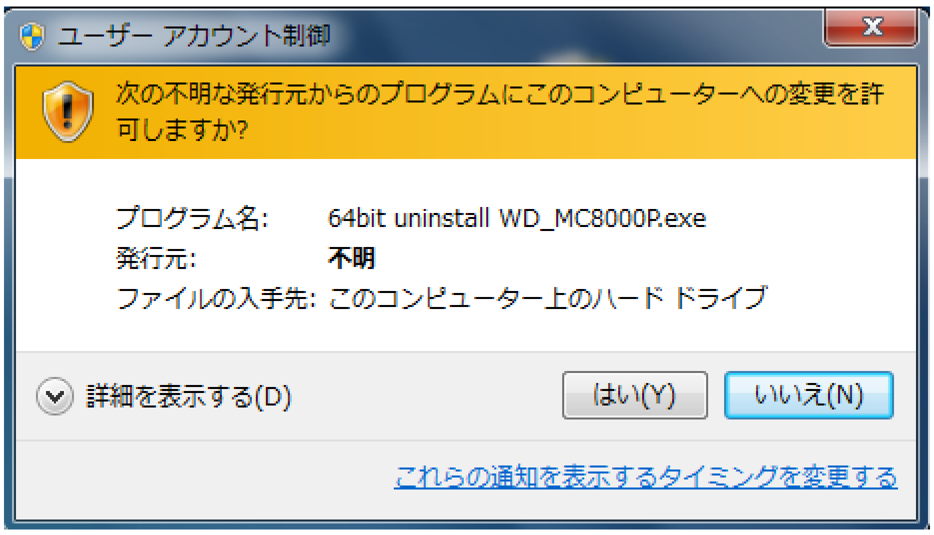
④　処理が終了すると自動的に画面が閉じられます。（アンインストールに時間がかかることがあります。完了のメッセージがあるまでお待ちください。）



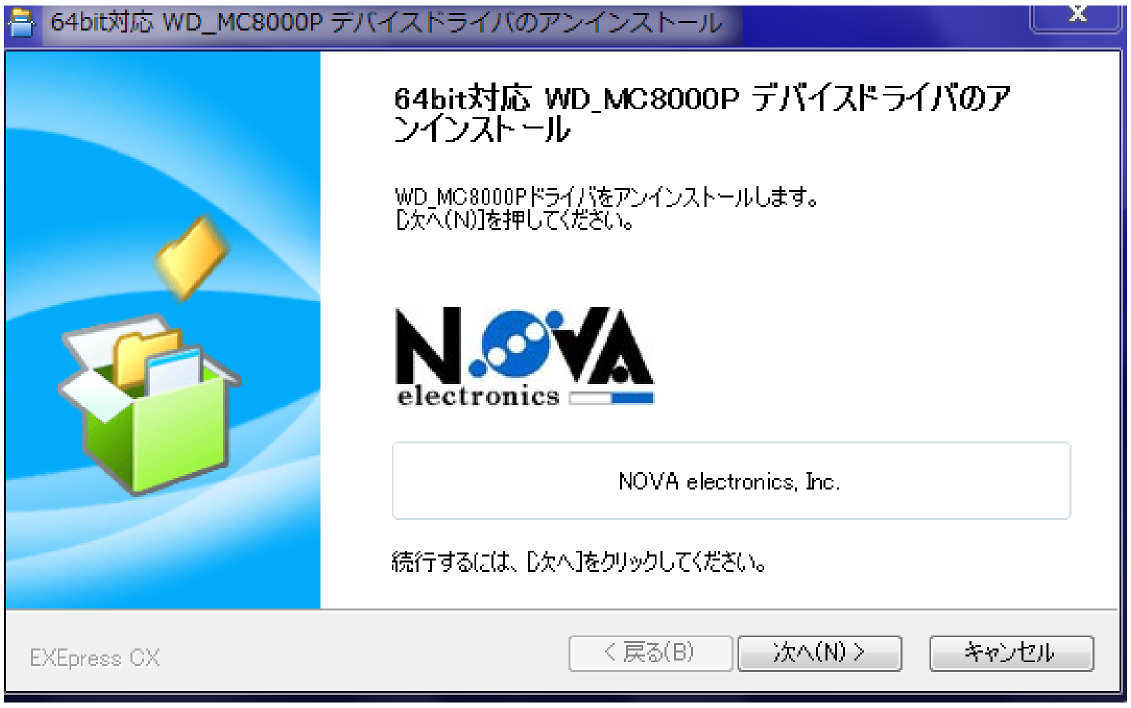
⑤　【ＯＫ】ボタンをクリックしてデバイスドライバのアンインストールは完了です。

⑥　ＰＣから全てのボード(MC8541PおよびMC8581P)を取り外す場合、フォルダ内の「64bit uninstall WD_MC8000P.exe」をダブルクリックしてください。MC8541Pは取り外し、MC8581Pは取り外さないなど、取り外さないボードがある場合は⑥～⑩の作業は必要ありません。

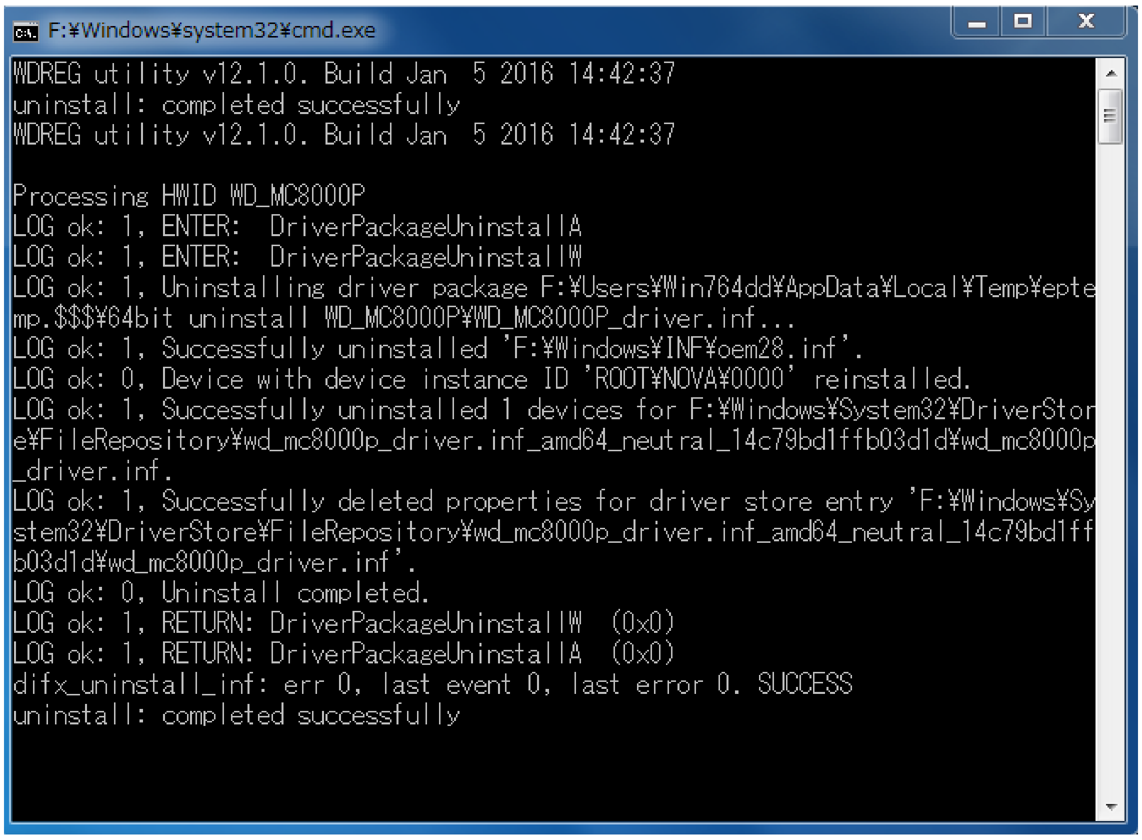
⑦　”ユーザーアカウント制御”　の画面が出てきた場合は、【許可】ボタンをクリックして操作を進めてください。



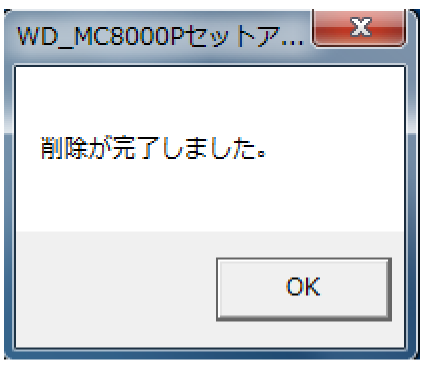
⑧　デバイスドライバのアンインストール画面
　　【次へ】のボタンをクリック



⑨　処理が終了すると自動的に画面が閉じられます。（アンインストールに時間がかかることがあります。完了のメッセージがあるまでお待ちください。）

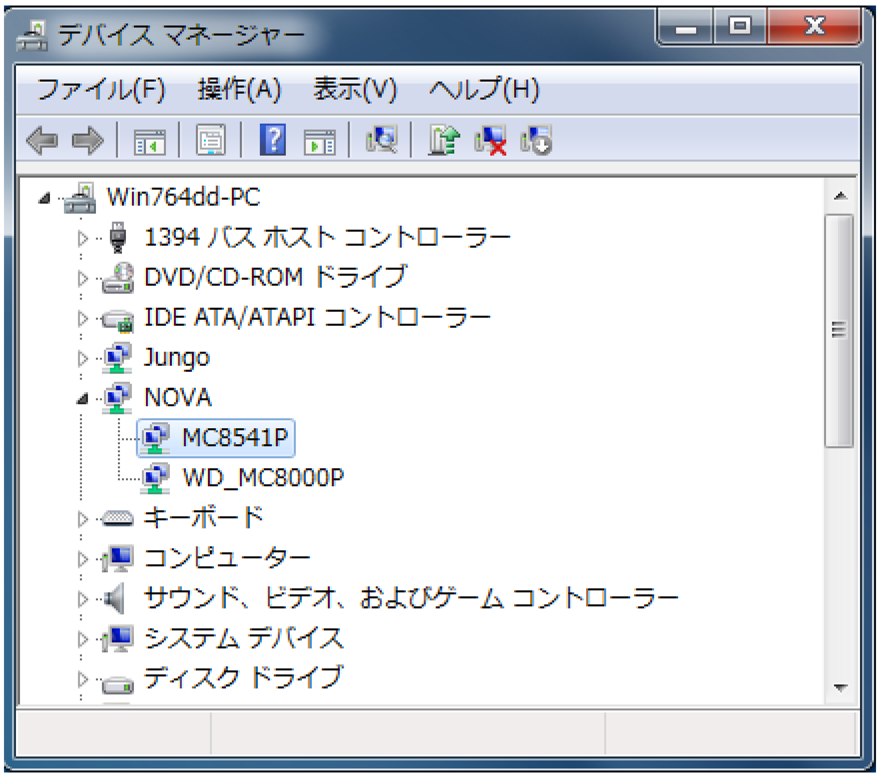


⑩　【ＯＫ】ボタンをクリックしてデバイスドライバのアンインストールは完了です。

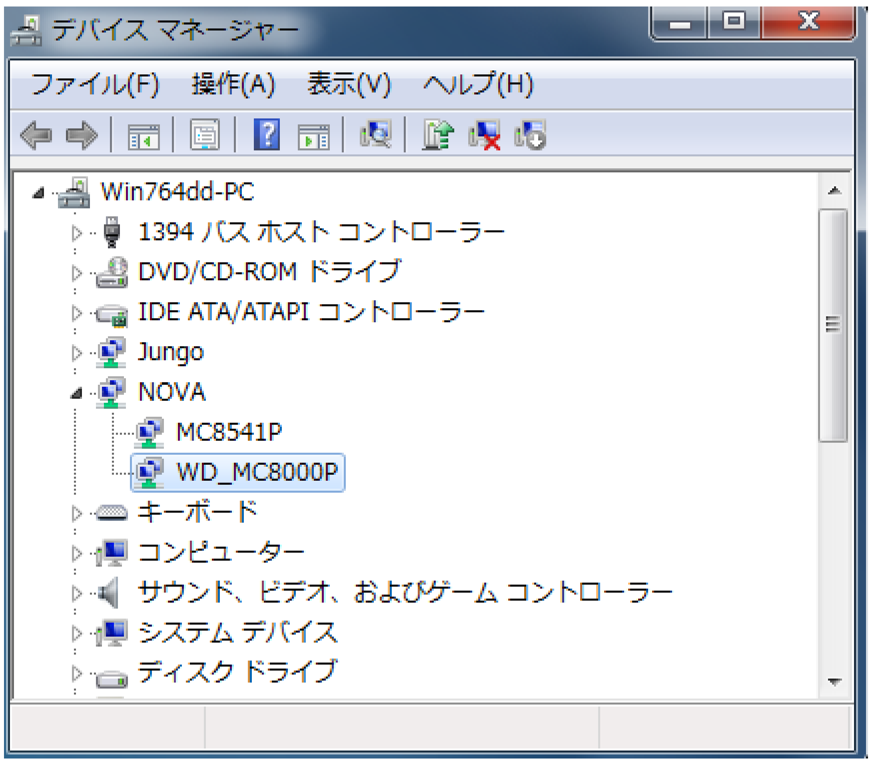


⑪　パソコン本体の電源がＯＦＦであることを確認してから、外装カバー、スロットカバー等を外します。
⑫　ボードを止めているビスを外します。
⑬　本ボードを指先でつまんで軽く左右にゆするようにしながら引き出します。

⑭　パソコン本体の電源をＯＮし、Windows 7を起動します。
⑮　[コントロールパネル]-[システムとセキュリティ]-[システム]-[デバイスマネージャ]タブで本製品が削除されていることを確認して下さい。
取り外すボードがMC8541Pとすると、図中の反転部分（MC8541P）が削除されていることを確認してください。



⑯　全てのボードを取り外す場合は、⑮と同様にして、図中の反転部分、WD_MC8000Pが削除されていることを確認してください。



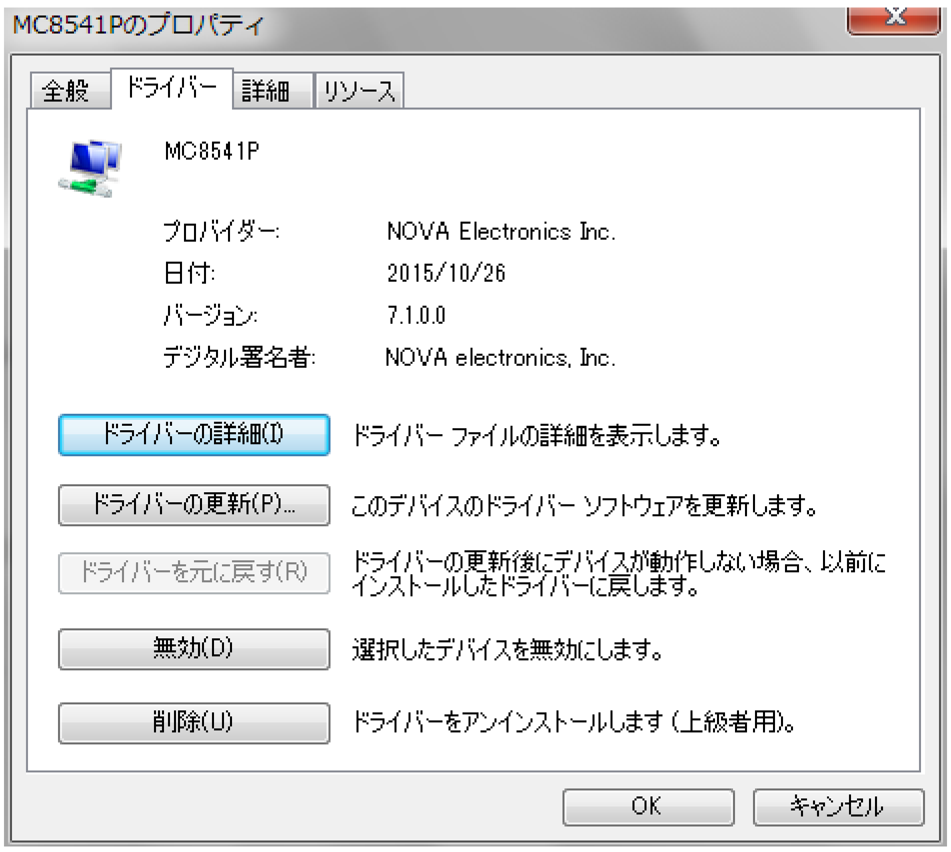
2.6　デバイスドライバの更新

デバイスドライバのバージョンが新しくなった場合、次の手順でドライバの更新を行って下さい。
以下に、更新手順を説明します。

起動中の他のアプリケーションは終了させてください。
デバイスドライバのインストールは必ずアドミニストレーター権限をもったユーザーログインで行ってください。アドミニストレーター権限以外でインストールをした場合正常にインストールされません。
下記の手順はWindows 7を例にしていますが、Windows 8.1、Windows 10でもデバイス更新の進め方は同じです。

① 2.1の方法でインストールするデバイスドライバを準備して下さい。
② 2.4.2の⑤から⑪を参照してインストールします。
③ 途中で次のような画面が表示されたら、【全て上書き(A)】をクリックします。

④ 下記画面（デバイスマネージャの更新したボードのプロパティ画面）で、更新したドライバのバージョンを確認します。
¥Driver¥Driver(VerX.X.X.X)¥Version.txt (あるいは¥Driver¥64Driver(VerX.X.X.X)¥Version.txt)ファイルの「１．ドライババージョン」のバージョンが下記画面に表示されますので正しいか確認して下さい。



３．プログラミング

この章では、アプリケーション開発のためのソフトウェア仕様とプログラミング方法について説明します。
アプリケーション開発は、Microsoft Visual C++ (以下VC++)、Microsoft Visual Basic (以下VB.NET)のどちらかを使用して行います。

3.1 動作環境

対応ＯＳ　　Windows 7(32bit,64bit) Windows 8.1(32bit,64bit) Windows 10(32bit,64bit)

対応言語

Microsoft Visual C++ 2008　以降
Microsoft Visual Basic 2008　以降

注）アプリケーション開発を行う場合は、開発ツールのサポート状況などMSDNのサポートページを参考にソフトウェアを開発してください。サンプルプログラムをVisual Studio 2010以降で動作させる場合は、MSDNのサポートページを参考にサンプルプログラムの移行を行ってください。64bitのデバイスドライバは64bitアプリにのみ対応しています。32bitアプリには対応しておりません。

3.2 ソフトウェア構成

3.2.1 ソフトウェア一覧

項目	フォルダ	ファイル名	説　　明
デバイスドライバ	Driver¥Driver(VerX.X.X.X)	install MC8541P INF.exe	32bit用MC8541Pデバイスドライバインストールプログラム
		install MC8581P INF.exe	32bit用MC8581Pデバイスドライバインストールプログラム
		uninstall MC8541P INF.exe	32bit用MC8541Pデバイスドライバアンインストールプログラム
		uninstall MC8581P INF.exe	32bit用MC8581Pデバイスドライバアンインストールプログラム
		uninstall WD_MC8000P	32bit用WD_MC8000Pアンインストールプログラム
	Driver¥64Driver(VerX.X.X.X)	64 install MC8541P INF.exe	64bit用MC8541Pデバイスドライバインストールプログラム
		64 install MC8581P INF.exe	64bit用MC8581Pデバイスドライバインストールプログラム
		64 uninstall MC8541P INF.exe	64bit用MC8541Pデバイスドライバアンインストールプログラム
		64 uninstall MC8581P INF.exe	64bit用MC8581Pデバイスドライバアンインストールプログラム
		64bit uninstall WD_MC8000P	64bit用WD_MC8000Pアンインストールプログラム
	Lib¥VC ¥lib	MC8000P.LIB	MC8000P.DLLを使用するためのライブラリ　32bit、VC++専用
		MC8000P_DLL.h	MC8000P.DLLを使用するためのヘッダ定義ファイル　VC++専用
	Lib¥VC ¥64 lib	MC8000P.LIB	MC8000P.DLLを使用するためのライブラリ　64bit、VC++専用
		MC8000P_DLL.h	MC8000P.DLLを使用するためのヘッダ定義ファイル　VC++専用
	Lib¥VB.NET	MC8000P_DLL.vb	MC8000P.DLLを使用するためのDeclare宣言ファイル VB.NET用
VC++サンプルプログラム (VC2008)	Sample¥ VC2008¥64(¥32)	Sample A	リミット入力表示、論理位置カウンタ表示、定量ドライブ操作例
		Sample E	割込みを用いた連続補間のプログラム例
VB.NETサンプルプログラム (VB2008)	Sample¥ VB2008	Sample A	リミット入力表示、論理位置カウンタ表示、定量ドライブ操作例
MCX514搭載ボード 評価ツール ※MC8541P、MC8581P	Tool ¥MCX514 Board ¥64(¥32)	MCX514_Tool.exe	MCX514搭載ボード評価ツール。画面からパラメータ、モード等を設定し、各コマンドを実行するアプリケーション。

備考：VC++のMFC AppWizerdが自動的に作成するファイルに関しては説明を省略します。

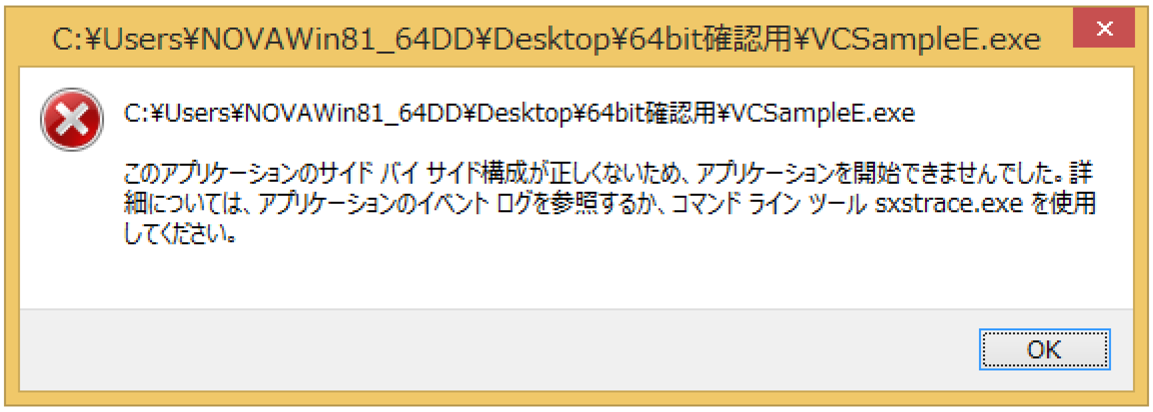
評価ツールについては、32bitOSでは32bitフォルダ(¥32)、64bitOSでは64bitフォルダ(¥64)の実行ファイルをご使用ください。
サンプルプログラムは、VC++は32bitOSでは32bitフォルダ(¥32)、64bitOSでは64bitフォルダ(¥64)のプロジェクトをご使用ください。VB.NETは32bitOS、64bitOS共通です。

3.2.2　サンプルプログラムおよび評価ツールの実行時にエラーが発生する場合の対応

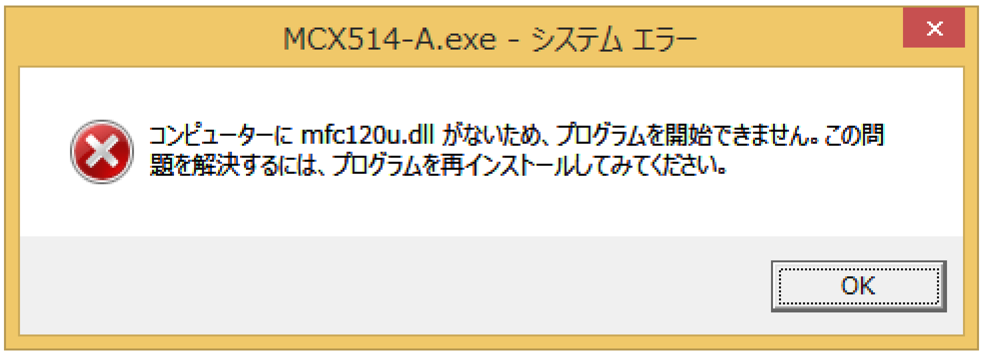
■　VC++で作成されたexe実行時のエラー

本ボードをインストールしたPCにVisualStudioがインストールされていない場合、exeファイル実行時に下図のようなエラーが発生し、実行できない場合があります。エラーが発生した場合、マイクロソフトホームページから「再配布可能パッケージ」をインストールしてください。サンプルプログラムはVisualStudio2008用、評価ツールはVisualStudio2013用のものをインストールします。また、32bitOS版、64bitOS版がありますので、それぞれに対しPCに対応したものをインストールしてください。
例えば、64bitOSで評価ツールのエラーを解除する場合は、VisualStudio2013の64bit版の「再配布可能パッケージ」をインストールすることになります。また、サンプルプログラムのエラーを解除する場合は、VisualStudio2008の64bit版の「再配布可能パッケージ」をインストールする必要があります。

注：エラーの詳細と解除方法の詳細については、必ずマイクロソフトのサポートページを参照し、マイクロソフトサポートの指示に従ってエラーを解除してください。



VC++サンプルプログラム実行時のエラー例

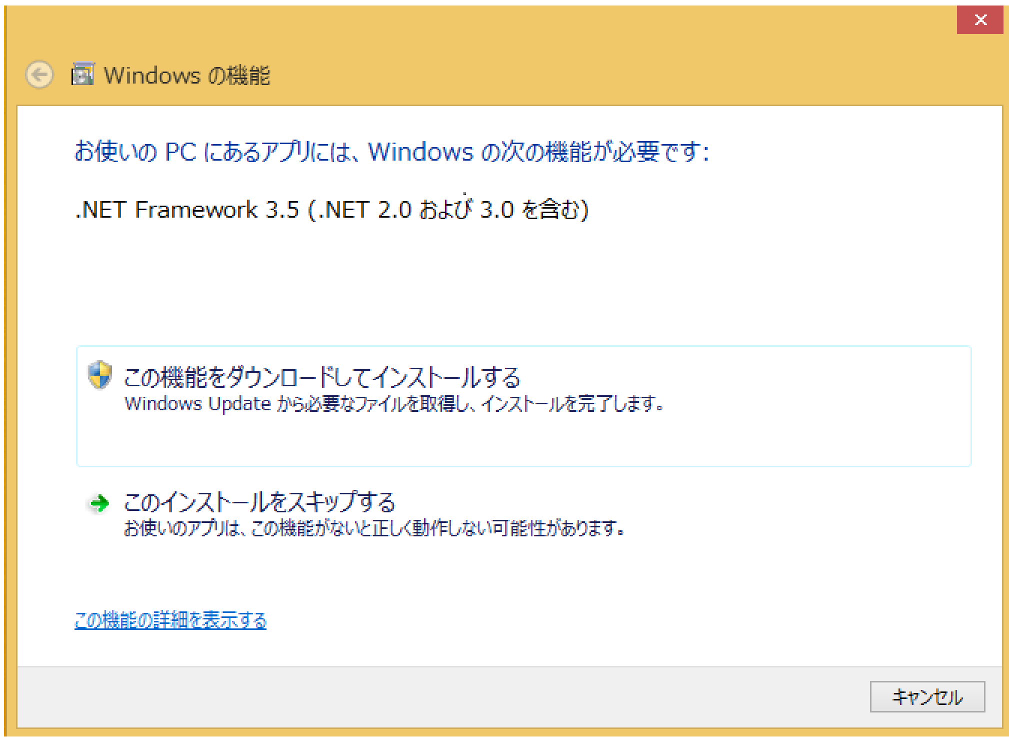


評価ツール実行時のエラー例

■ VB.NETで作成されたexe実行時のエラー

本ボードをインストールしたPCに.NET Framework 3.5がインストールされていない場合、exeファイル実行時に下図のようなエラーが発生し、実行できない場合があります。エラーが発生した場合、画面に従って.NET Framework 3.5 をインストールするか、[コントロールパネル]-[プログラム]-[Windowsの機能と有効化または無効化]で.NET Framework 3.5を有効にして下さい。

注：エラーの詳細と解除方法の詳細については、必ずマイクロソフトのサポートページを参照し、マイクロソフトサポートの指示に従ってエラーを解除してください。



VB.NETサンプルプログラム実行時のエラー例

3.3 開発手順

3.3.1 ＶＣ＋＋の場合

アプリケーションはMC8000P.libとMC8000P_DLL.hファイルを使用します。この２ファイルは VC++ 2008 以降対応です。

① 32bitOSの場合は¥Lib¥VC¥libフォルダ、64bitOSの場合は¥Lib¥VC¥64libに入っている２つのファイルMC8000P.libとMC8000P_DLL.hを開発するアプリケーションのフォルダにコピーしてください。

② VC++の総合開発環境にてMC8000P_DLL.h をご使用のプロジェクトに追加登録してください。
また、API 関数を使用するソースファイルにMC8000P_DLL.h をincludeして下さい。

③ [プロジェクト]－[プロパティ]画面で[リンカ]－[入力]を選択し「追加の依存ファイル」にMC8000P.libを追加して下さい。

④ 3.4 ＡＰＩの関数を使用してプログラミングを行って下さい。

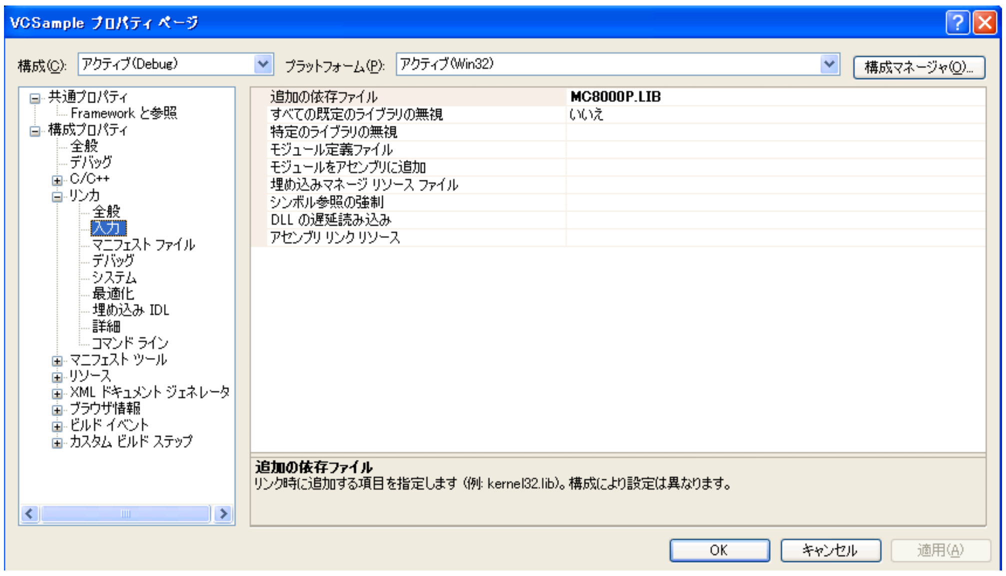


図3.3-1 VC++ 2008 プロジェクトのプロパティ

3.3.2 ＶＢ．ＮＥＴの場合

アプリケーションはMC8000P_DLL.vbファイルを使用します。このファイルは VB2008 以降対応です。

① ¥Lib¥VB.NET フォルダに入っているMC8000P_DLL.vbファイルを開発するアプリケーションのプロジェクトに追加してください。

② 3.4 ＡＰＩの関数を使用してプログラミングを行って下さい。

注意：VB.NET では、本ボードに関する割り込みを使用することはできません。

3.4　ＡＰＩ

MC8000P.SYS、MC8000P.DLL がアプリケーションに提供するＡＰＩ

3.4.1　関数一覧

下表は、API関数の一覧表です。
「VC」「VB.NET」の欄は、各言語において各関数が使用できるかどうかを記載しています。
VC++の場合　　・・・・・・・［ VC ］の項目を参照して下さい。
VB.NETの場合　・・・・・・・［ VB.NET ］の項目を参照して下さい。

○は使用できます。×は使用できません。

（１）基本関数

関数名	説明	VC	VB.NET	ページ	備考
Nmc_Open	ボードの使用を開始する	○	○	19	
Nmc_Close	ボードの使用を終了する	○	○	〃	
Nmc_CloseAll	全てのボードの使用を終了する	○	○	〃	
Nmc_GetBoardInfo	オープンしたボードの情報を取得する	○	○	20	
Nmc_OutPort	出力ポートにデータを書く	○	○	〃	
Nmc_InPort	入力ポートからデータを読む	○	○	〃	
Nmc_WriteReg	ボードのレジスタにデータを書き込む	○	○	21	
Nmc_ReadReg	ボードのレジスタからデータを読み込む	○	○	〃	
Nmc_SetEvent	割込みを処理するユーザー関数を設定する	○	×	22	
Nmc_ResetEvent	割込みを処理するユーザー関数の設定を解除する	○	×	〃	
Nmc_ReadEvent	割込み発生直後の各軸ＲＲ１の値を取得する	○	×	23	

（２）リセット、命令

関数名	説明	VC	VB.NET	ページ	備考
Nmc_Reset	ボードに搭載しているＩＣをリセットする	○	○	23	
Nmc_Command	指定軸の命令を実行する	○	○	24	
Nmc_Command_IP	補間命令を実行する	○	○	〃	

（３）ライトレジスタ

関数名	説明	VC	VB.NET	ページ	備考
Nmc_WriteReg0	ＷＲ０(コマンドレジスタ)　　書き込み	○	○	25	
Nmc_WriteReg1	ＷＲ１(モードレジスタ１)　　書き込み	○	○	〃	
Nmc_WriteReg2	ＷＲ２(モードレジスタ２)　　書き込み	○	○	〃	
Nmc_WriteReg3	ＷＲ３(モードレジスタ３)　　書き込み	○	○	26	
Nmc_WriteReg4	ＷＲ４(アウトプットレジスタ１)書き込み	○	○	〃	
Nmc_WriteReg5	ＷＲ５(アウトプットレジスタ２)書き込み	○	○	〃	
Nmc_WriteReg6	ＷＲ６(ライトデータレジスタ１)書き込み	○	○	27	
Nmc_WriteReg7	ＷＲ７(ライトデータレジスタ２)書き込み	○	○	〃	

（４）リードレジスタ

関数名	説明	VC	VB.NET	ページ	備考
Nmc_ReadReg0	ＲＲ０(主ステータスレジスタ)　読み出し	○	○	27	
Nmc_ReadReg2	ＲＲ２(ステータスレジスタ２)　読み出し	○	○	〃	
Nmc_ReadReg3	ＲＲ３(ステータスレジスタ３)　読み出し (ページ指定なし)	○	○	28	
Nmc_ReadReg3P	ＲＲ３(ステータスレジスタ３)　読み出し (ページ指定あり)	○	○	〃	
Nmc_ReadReg4	ＲＲ４(インプットレジスタ１)　読み出し	○	○	〃	
Nmc_ReadReg5	ＲＲ５(インプットレジスタ２)　読み出し	○	○	29	
Nmc_ReadReg6	ＲＲ６(リードデータレジスタ１)読み出し	○	○	〃	
Nmc_ReadReg7	ＲＲ７(リードデータレジスタ２)読み出し	○	○	〃	

（５）パラメータ設定

関数名	説明	VC	VB.NET	ページ	備考
Nmc_Jerk	加速度増加率設定（加加速度）	○	○	29	
Nmc_DJerk	減速度増加率設定	○	○	30	
Nmc_Acc	加速度設定	○	○	〃	
Nmc_Dec	減速度設定	○	○	〃	
Nmc_StartSpd	初速度設定	○	○	31	
Nmc_Speed	ドライブ速度設定	○	○	〃	
Nmc_Pulse	移動パルス数／補間終点設定(VC用)	○	×	〃	
Nmc_Pulse_VB	移動パルス数／補間終点設定(VB.NET用)	×	○	32	
Nmc_DecP	マニュアル減速点設定(VC用)	○	×	〃	
Nmc_DecP_VB	マニュアル減速点設定(VB.NET用)	×	○	〃	
Nmc_Center	円弧中心点設定	○	○	33	
Nmc_Lp	論理位置カウンタ設定	○	○	〃	
Nmc_Ep	実位置カウンタ設定	○	○	〃	
Nmc_CompP	ソフトリミット＋レジスタ設定	○	○	34	
Nmc_CompM	ソフトリミット－レジスタ設定	○	○	〃	
Nmc_AccOfst	加速カウンタオフセット設定	○	○	〃	
Nmc_HomeSpd	原点検出速度設定	○	○	35	
Nmc_LpMax	論理位置カウンタ最大値設定	○	○	〃	
Nmc_RpMax	実位位置カウンタ最大値設定	○	○	〃	
Nmc_MR0	多目的レジスタ０設定	○	○	36	
Nmc_MR1	多目的レジスタ１設定	○	○	〃	
Nmc_MR2	多目的レジスタ２設定	○	○	〃	
Nmc_MR3	多目的レジスタ３設定	○	○	37	
Nmc_SpeedInc	速度増減値設定	○	○	〃	
Nmc_Timer	タイマー値設定	○	○	〃	
Nmc_TPMax	補間・終点最大値設定	○	○	38	
Nmc_HLNumber	ヘリカル回転数設定	○	○	〃	
Nmc_HLValue	ヘリカル演算値設定	○	○	〃	

（６）その他のモード設定

関数名	説明	VC	VB.NET	ページ	備考
Nmc_MRmMode	多目的レジスタモード設定	○	○	38	
Nmc_PIO1Mode	ＰＩＯ信号設定１	○	○	39	
Nmc_PIO2Mode	ＰＩＯ信号設定２･その他設定	○	○	〃	
Nmc_HMSrch1Mode	自動原点出しモード設定１	○	○	〃	
Nmc_HMSrch2Mode	自動原点出しモード設定２	○	○	40	
Nmc_FilterMode	入力信号フィルタモード設定	○	○	〃	
Nmc_Sync0Mode	同期動作ＳＹＮＣ０設定	○	○	41	
Nmc_Sync1Mode	同期動作ＳＹＮＣ１設定	○	○	〃	
Nmc_Sync2Mode	同期動作ＳＹＮＣ２設定	○	○	42	
Nmc_Sync3Mode	同期動作ＳＹＮＣ３設定	○	○	〃	
Nmc_IPMode	補間モード設定	○	○	〃	

（７）データ読み出し

関数名	説明	VC	VB.NET	ページ	備考
Nmc_ReadLp	論理位置カウンタ読み出し	○	○	43	
Nmc_ReadEp	実位置カウンタ読み出し	○	○	〃	
Nmc_ReadSpeed	現在ドライブ速度読み出し	○	○	〃	
Nmc_ReadAccDec	現在加／減速度読み出し	○	○	44	
Nmc_ReadMR0	多目的レジスタ０読み出し	○	○	〃	
Nmc_ReadMR1	多目的レジスタ１読み出し	○	○	〃	
Nmc_ReadMR2	多目的レジスタ２読み出し	○	○	45	
Nmc_ReadMR3	多目的レジスタ３読み出し	○	○	〃	
Nmc_ReadCT	現在タイマー値 読み出し	○	○	〃	
Nmc_ReadTX	補間・終点最大値読み出し	○	○	46	
Nmc_ReadCHLN	現在ヘリカル回転数読み出し	○	○	〃	
Nmc_ReadHLV	ヘリカル演算値読み出し	○	○	〃	
Nmc_ReadWR1	WR1設定値読み出し	○	○	47	
Nmc_ReadWR2	WR2設定値読み出し	○	○	〃	
Nmc_ReadWR3	WR3設定値読み出し	○	○	〃	
Nmc_ReadMRM	多目的レジスタモード設定読み出し	○	○	48	
Nmc_ReadP1M	PIO信号設定１読み出し	○	○	〃	
Nmc_ReadP2M	PIO信号設定２その他設定読み出し	○	○	〃	
Nmc_ReadAc	加速度設定値読み出し	○	○	49	
Nmc_ReadStartSpd	初速度設定値読み出し	○	○	〃	
Nmc_ReadSetSpeed	ドライブ速度設定値読み出し	○	○	〃	
Nmc_ReadPulse	移動パルス数/終点設定値読み出し	○	○	50	

（８）状態取得

関数名	説明	VC	VB.NET	ページ	備考
Nmc_GetDriveStatus	ドライブ状態取得	×	○	50	
Nmc_GetCNextStatus	連続補間次データ書込み可能状態取得	×	○	51	
Nmc_GetSc	連続補間プリバッファスタックカウンタ取得	×	○	〃	

（９）書き込み・読み出し

関数名	説明	VC	VB.NET	ページ	備考
Nmc_WriteRegSetAxis	軸指定ライトレジスタ書き込み(WR1～3)	○	○	51	
Nmc_ReadRegSetAxis	軸指定リードレジスタ読み出し(RR2～3)	○	○	52	
Nmc_WriteData	データ書き込み（パラメータ）	○	○	〃	
Nmc_ReadData	データ読み出し(4バイト読み出し)	○	○	53	

（１０）ＢＰ補間・連続補間実行

関数名	説明	VC	VB.NET	ページ	備考
Nmc_2BPExecMC8500P	２軸ＢＰ補間実行	○	○	54	
Nmc_3BPExecMC8500P	３軸ＢＰ補間実行	○	○	56	
Nmc_4BPExecMC8500P	４軸ＢＰ補間実行	○	○	58	
Nmc_2BPExecMC8500P_BG	２軸ＢＰ補間実行（バックグラウンドで実行）	○	○	60	
Nmc_3BPExecMC8500P_BG	３軸ＢＰ補間実行（バックグラウンドで実行）	○	○	62	
Nmc_4BPExecMC8500P_BG	４軸ＢＰ補間実行（バックグラウンドで実行）	○	○	64	
Nmc_2CIPExecMC8500P	２軸連続補間実行	○	○	66	
Nmc_3CIPExecMC8500P	３軸連続補間実行	○	○	68	
Nmc_4CIPExecMC8500P	４軸連続補間実行	○	○	70	
Nmc_2CIPExecMC8500P_BG	２軸連続補間実行（バックグラウンドで実行）	○	○	72	
Nmc_3CIPExecMC8500P_BG	３軸連続補間実行（バックグラウンドで実行）	○	○	74	
Nmc_4CIPExecMC8500P_BG	４軸連続補間実行（バックグラウンドで実行）	○	○	76	
Nmc_IPStop	補間実行を中断する	○	○	78	
Nmc_IPGetMsgNo	補間終了時の受信ﾒｯｾｰｼﾞからﾎﾞｰﾄﾞ番号とＩＣ番号取得	○	○	〃	

（１１）ヘリカル補間実行

関数名	説明	VC	VB.NET	ページ	備考
Nmc_HLValueExe	ヘリカル演算実行	○	○	79	
Nmc_HLExec	ヘリカル補間実行	○	○	81	

3.4.2 関数仕様

VC++の場合 : **VC** と[VC]の項目を参照して下さい。
VB.NETの場合 : **VB.NET** と[VB.NET]の項目を参照して下さい。
指定のない項目は、各言語共通の内容です。

<table>
<tr><th>関数名</th><th>機能　及び　内容</th></tr>
<tr><td>Nmc_Open</td><td>ボードの使用を開始する。

VC　BOOL　Nmc_Open(int No, BOOL IntrptFlg);
VB.NET　Function Nmc_Open(ByVal No As Integer, ByVal IntrptFlg As Integer) As Integer

入力パラメータ
No　ボード番号（ボード上のロータリースイッチの値(0～15)）
IntrptFlg　割り込みを使用するかどうかを指定する。
[VC] TRUE: 使用する。FALSE: 使用しない。
[VB.NET] False固定。割り込みを使用しない設定。（VBでは割り込みを使用できません）

戻り値
[VC]　オープンに成功するとTRUE、失敗するとFALSE
[VB.NET]　オープンに成功すると０以外、失敗すると０

使用例
[VC]　status = Nmc_Open(0, FALSE);　// ボード番号０をオープン、割り込みを使用しない
[VB.NET]　status = Nmc_Open(0, False)　'ボード番号０をオープン、割り込みを使用しない</td></tr>
<tr><td>Nmc_Close</td><td>ボードの使用を終了する。

VC　BOOL　Nmc_Close(int No);
VB.NET　Function Nmc_Close(ByVal No As Integer) As Integer

入力パラメータ
No　ボード番号（ボード上のロータリースイッチの値(0～15)）

戻り値
[VC]　クローズに成功するとTRUE、失敗するとFALSE
[VB.NET]　クローズに成功すると０以外、失敗すると０

使用例
[VC]　status = Nmc_Close(0);　// ボード番号０をクローズ
[VB.NET]　status = Nmc_Close(0)　'ボード番号０をクローズ</td></tr>
<tr><td>Nmc_CloseAll</td><td>全てのボードの使用を終了する。

VC　BOOL　Nmc_CloseAll(void);
VB.NET　Function Nmc_CloseAll() As Integer

入力パラメータ
なし

戻り値
[VC]　クローズに成功するとTRUE、失敗するとFALSE
[VB.NET]　クローズに成功すると０以外、失敗すると０

使用例
[VC]　status = Nmc_CloseAll();　// 全てのボードをクローズ
[VB.NET]　status = Nmc_CloseAll()　'全てのボードをクローズ</td></tr>
</table>

関数名	機能　及び　内容
Nmc_GetBoardInfo	オープンしたボードの情報としてデバイスＩＤを取得する。 **VC**　　BOOL　　Nmc_GetBoardInfo(int No, USHORT* pDeviceID); **VB.NET**　Function Nmc_GetBoardInfo(ByVal No As Integer, ByRef DeviceID As Short) As Integer **入力パラメータ** 　No　　　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　pDeviceID　[VC/VB.NET]: 取得したボードのデバイスＩＤを格納する変数のアドレス 　　　　　　各ボードのデバイスＩＤは補足説明(1)③参照。 　　　　　　[VC/VB.NET]　MC8541PはID_MC8541P、MC8581PはID_MC8581P。 **戻り値** 　[VC]　　　取得が成功するとTRUE、失敗するとFALSE 　[VB.NET]　取得が成功すると０以外、失敗すると０ **使用例** 　[VC]　　　USHORT DeviceID; 　　　　　　status = Nmc_GetBoardInfo(No, &DeviceID);　// デバイスＩＤ取得 　　　　　　if(DeviceID == ID_MC8581P)　　　　　　　// MC8581Pの場合 　[VB.NET]　Dim DeviceID As Short 　　　　　　status = Nmc_GetBoardInfo(No, DeviceID)　　'デバイスＩＤ取得 　　　　　　If DeviceID = ID_MC8581P Then　　　　　　'MC8581Pの場合
Nmc_OutPort	出力ポートに２バイトデータを書き込む。 **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) **入力パラメータ** 　No　ボード番号（ボード上のロータリースイッチの値(0～15)） 　Adr　書き込むアドレス。各ボード取扱説明書に記載しているＩ／Ｏアドレス。 　　　詳細は補足説明(1)①、(6)参照。 　Dat　書き込むデータ **戻り値** 　なし **使用例** 　[VC]　　　Nmc_OutPort(No, 0, MC8500P_CMD_RST);　//　ＩＣ－Ａのコマンドリセット (WR0書き込み) 　[VB.NET]　Call Nmc_OutPort(No, 0, MC8500P_CMD_RST)　'ＩＣ－Ａのコマンドリセット (WR0書き込み)
Nmc_InPort	入力ポートから２バイトデータを読み出す。 **VC**　　long　　Nmc_InPort(int No, long Adr); **VB.NET**　Function Nmc_InPort(ByVal No As Integer, ByVal adr As Integer) As Integer **入力パラメータ** 　No　ボード番号（ボード上のロータリースイッチの値(0～15)） 　Adr　読み出すアドレス。各ボード取扱説明書に記載しているＩ／Ｏアドレス。 　　　詳細は補足説明(1)①、(6)参照。 **戻り値** 　入力ポートから読み込んだデータ **使用例** 　[VC]　　　data = Nmc_InPort(No, 0);　//　ＩＣ－Ａのリードレジスタ RR0 の読み出し 　[VB.NET]　data = Nmc_InPort(No, 0)　　'ＩＣ－Ａのリードレジスタ RR0 の読み出し **注意** 　[VC]　ＲＲ１レジスタ(ステータスレジスタ：割り込みが発生した要因を表示するレジスタ)のデータ読み出しに関しては、Nmc_ReadEvent関数の説明を参考にして下さい。

<table>
<tr><th>関数名</th><th>機能　及び　内容</th></tr>
<tr><td>Nmc_WriteReg</td><td>ボードのライトレジスタ(WR0～WR7)にデータを書き込む。

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)

入力パラメータ
　No　　　ボード番号（ボード上のロータリースイッチの値(0～15)）
　IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。
　　　　　詳細は補足説明(7)参照。

　RegNum　書き込むレジスタ(MCX_WR0～MCX_WR7)。詳細は補足説明(1)参照。
　　　　　例）[VC/VB.NET]WR0の場合は MCX_WR0 を、WR1の場合は MCX_WR1 を指定する。

　Dat　　　書き込むデータ

戻り値
　なし

使用例
　[VC]　　　Nmc_WriteReg(No, 0, MCX_WR0, MC8500P_CMD_RST);
　　　　　　　　　　　　　　　　　　　　// ＩＣ－Ａのコマンドリセット(WR0書き込み)
　[VB.NET]　Call Nmc_WriteReg(No, 0, MCX_WR0, MC8500P_CMD_RST)
　　　　　　　　　　　　　　　　　　　　’ＩＣ－Ａのコマンドリセット(WR0書き込み)</td></tr>
<tr><td>Nmc_ReadReg</td><td>ボードのリードレジスタ(RR0～RR7)からデータを読み出す。

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 Integer)
　　　　　　As Integer

入力パラメータ
　No　　　ボード番号（ボード上のロータリースイッチの値(0～15)）
　IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。
　　　　　詳細は補足説明(7)参照。

　RegNum　読み出すレジスタ(MCX_RR0～MCX_RR7)。詳細は補足説明(1)参照。
　　　　　例）[VC/VB.NET]RR0の場合は MCX_RR0 を、RR2の場合は MCX_RR2 を指定する。

戻り値
　リードレジスタから読み出したデータ

使用例
　[VC]　　　data = Nmc_ReadReg(No, 0, MCX_RR0);　// ＩＣ－Ａのリードレジスタ RR0 の読み出し
　[VB.NET]　data = Nmc_ReadReg(No, 0, MCX_RR0)　　’ＩＣ－Ａのリードレジスタ RR0 の読み出し

注意
　[VC]　ＲＲ１レジスタのデータ読み出しに関しては、Nmc_ReadEvent関数の説明を参考にして下さい。</td></tr>
</table>

関数名	機能　及び　内容

Nmc_SetEvent	割込みを処理するユーザー関数を設定する。
	この関数を実行すると、割り込みが発生した時にユーザー関数が呼び出され、指定した引数が１つ渡されます。このユーザー関数は１つのスレッドとして起動されます。
	割込み処理する関数の設定を解除する場合は Nmc_ResetEvent を実行して下さい。
	
	VC　　BOOL Nmc_SetEvent(int No, LPTHREAD_START_ROUTINE UserFunc, LPVOID IpParameter);
	VB.NET　使用できません
	
	入力パラメータ
	No　　　　ボード番号（ボード上のロータリースイッチの値(0～15)）
	UserFunc　　ユーザー関数のアドレス
	IpParameter　ユーザー関数スレッドに渡す１つの引数を指定する。
	引数を使用しない場合は、NULL等で良い。
	スレッドで使用可能なポインタを設定して下さい。
	ポインタの場合、ユーザー関数呼び出し時に使用可能なポインタを設定して下さい。
	
	戻り値
	成功するとTRUE、失敗するとFALSE
	
	使用例
	[VC]
	（ボード番号０の場合）
	status = Nmc_SetEvent(0, MC_EventFunc0, IpParam); // 関数のアドレスと引数を設定
	Nmc_WriteReg1(0, 0, AXIS_ALL, 0x0080);　　　　// ＩＣ－Ａのドライブ終了割込み発生（全軸）
	
	（ボード番号１の場合）
	status = Nmc_SetEvent(1, MC_EventFunc1, NULL);　　//関数のアドレスと引数を設定
	Nmc_WriteReg1(1, 0, AXIS_ALL, 0x0080);　　　　// ＩＣ－Ａのドライブ終了割込み発生（全軸）
	
	■割込みユーザー関数例
	DWORD WINAPI MC_EventFunc0(LPVOID IpParam)
	{
	　long Rr1X, Rr1Y, Rr1Z, Rr1U;
	　Nmc_ReadEvent(0, 0, &Rr1X,&Rr1Y,&Rr1Z,&Rr1U); //　ボード0,IC-AのRR1割込みデータ読み出し
	　・・・・
	　return 0;
	}
	
	DWORD WINAPI MC_EventFunc1(LPVOID IpParam)
	{
	　long Rr1X, Rr1Y, Rr1Z, Rr1U;
	　Nmc_ReadEvent(1, 0, &Rr1X,&Rr1Y,&Rr1Z,&Rr1U); // ボード1,IC-AのRR1割込みデータ読み出し
	　・・・・
	　return 0;
	}

Nmc_ResetEvent	割込みを処理するユーザー関数の設定を解除する。 この関数を実行すると、割り込みが発生してもユーザー関数は呼び出されません。 **VC**　　BOOL Nmc_ResetEvent(int No); **VB.NET**　使用できません **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） **戻り値** [VC]　成功するとTRUE、失敗するとFALSE **使用例** [VC]　status = Nmc_ResetEvent(No);

関数名	機能　及び　内容

Nmc_ReadEvent	割込み発生直後の各軸ＲＲ１の値を取得する。（ドライバ内のＲＲ１データは読み出し後クリアされる）
	
	VC　　BOOL Nmc_ReadEvent(int No, int IcNo, long* RrIrX, long* RrIrY, long* RrIrZ, long* RrIrU);
	VB.NET　使用できません
	
	入力パラメータ
	　No　　ボード番号（ボード上のロータリースイッチの値(0～15)）
	　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。
	　RrIrX　　Ｘ軸のＲＲ１を格納する為のバッファへのポインタ
	　RrIrY　　Ｙ軸のＲＲ１を格納する為のバッファへのポインタ
	　RrIrZ　　Ｚ軸のＲＲ１を格納する為のバッファへのポインタ
	　RrIrU　　Ｕ軸のＲＲ１を格納する為のバッファへのポインタ
	
	戻り値
	　[VC]　成功するとTRUE、失敗するとFALSE
	
	使用例
	　[VC]
	　long Rr1X[2], Rr1Y[2], Rr1Z[2], Rr1U[2];
	　Nmc_ReadEvent(No, 0, &Rr1X[0], &Rr1Y[0], &Rr1Z[0], &Rr1U[0]); // IC-AのRR1データを読み出す
	　Nmc_ReadEvent(No, 1, &Rr1X[1], &Rr1Y[1], &Rr1Z[1], &Rr1U[1]); // IC-BのRR1データを読み出す
	
	注意
	　ボードで割込が発生した直後ドライバ内でＲＲ１を読み出してしまうのでＲＲ１はクリアされてしまいます。割込み発生直後のＲＲ１を確認する場合はこの関数を使用してください。
	
	　また、Nmc_SetEvent、Nmc_ResetEvent関数の実行とは関係なく、割込みが発生するとドライバは必ずＲＲ１データを読み出し保存します。ドライバ内に保存されたＲＲ１データは、Nmc_ReadEvent関数を実行して読み出すとクリアされます。
	
	　ドライバ内のＲＲ１データをクリアしたい時は、Nmc_ReadEvent関数を実行して下さい。

Nmc_Reset	ボードに搭載しているＩＣをリセットする。 **VC**　　void Nmc_Reset(int No, int IcNo); **VB.NET**　Sub　Nmc_Reset(ByVal No As Integer, ByVal IcNo As Integer) **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 **戻り値** 　なし **使用例** 　[VC]　　Nmc_Reset(1, 0);　　　// ボード番号１のＩＣ－Ａをリセットする 　[VB.NET]　Call Nmc_Reset(1, 0)　　'ボード番号１のＩＣ－Ａをリセットする

関数名	機能　及び　内容
Nmc_Command	指定軸の命令を実行する。（WR0に指定軸の命令を書く） **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。 詳細は補足説明(7)参照。 axis　命令を実行する軸。複数軸指定可能。詳細は補足説明(2)参照。 cmd　命令番号。定義ファイル(※１)内のコマンド定義の「ドライブ命令、その他の命令」の中から１つを指定する。相対位置ドライブの場合はMC8500P_CMD_DRVRLを指定する。 ※１：[VC]MC8000P_DLL.H ／ [VB.NET]MC8000P_DLL.vb **戻り値** なし **使用例** [VC]　　Nmc_Command(No, IcNo, AXIS_X, MC8500P_CMD_DRVRL);　// Ｘ軸の相対位置ドライブを実行する [VB.NET]　Call Nmc_Command(No, IcNo, AXIS_X, MC8500P_CMD_DRVRL)　'Ｘ軸の相対位置ドライブを実行する
Nmc_Command_IP	補間命令を実行する。（WR0に補間命令を書く） **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。 詳細は補足説明(7)参照。 cmd　命令番号。定義ファイル(※１)内のコマンド定義の「補間命令」の中から１つを指定する。 ２軸直線補間ドライブの場合はMC8500P_CMD_2CIPを指定する。 ※１：[VC]MC8000P_DLL.H ／ [VB.NET]MC8000P_DLL.vb **戻り値** なし **使用例** [VC]　Nmc_IPMode(No, IcNo, 0x0003);　　　　// 補間軸設定（主軸：Ｘ、第２軸：Ｙ） Nmc_Command_IP(No, IcNo, MC8500P_CMD_2CIP);　// ２軸直線補間ドライブを実行する [VB.NET]　Call Nmc_IPMode(No, IcNo, &H3)　　　　'補間軸設定（主軸：Ｘ、第２軸：Ｙ） Call Nmc_Command_IP(No, IcNo, MC8500P_CMD_2CIP)　'２軸直線補間ドライブを実行する

<table>
<tr><th>関数名</th><th>機能　及び　内容</th></tr>
<tr><td>Nmc_WriteReg0</td><td>ＷＲ０(コマンドレジスタ)にデータを書き込む。

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)

入力パラメータ
　No　　ボード番号（ボード上のロータリースイッチの値(0～15)）
　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。
　　　　詳細は補足説明(7)参照。

　wdata　書き込むデータ

戻り値
　なし

使用例
　[VC]　　　Nmc_WriteReg0(No, IcNo, 0x0150);　// Ｘ軸の相対位置ドライブを実行する
　[VB.NET]　Call Nmc_WriteReg0(No, IcNo, &H150)　'Ｘ軸の相対位置ドライブを実行する</td></tr>
<tr><td>Nmc_WriteReg1</td><td>ＷＲ１(モードレジスタ１)にデータを書き込む。

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)

入力パラメータ
　No　　ボード番号（ボード上のロータリースイッチの値(0～15)）
　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。
　　　　詳細は補足説明(7)参照。

　axis　データを書き込む軸。AXIS_X, AXIS_Y 等を指定。複数軸指定可能。
　　　　詳細は補足説明(2)参照。

　wdata　書き込むデータ

戻り値
　なし

使用例
　[VC]　　　Nmc_WriteReg1(No, IcNo, AXIS_X, 0x0080);　// ドライブエンド割り込み発生(Ｘ軸)
　[VB.NET]　Call Nmc_WriteReg1(No, IcNo, AXIS_X, &H80)　'ドライブエンド割り込み発生(Ｘ軸)</td></tr>
<tr><td>Nmc_WriteReg2</td><td>ＷＲ２(モードレジスタ２)にデータを書き込む。

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)

入力パラメータ
　No　　ボード番号（ボード上のロータリースイッチの値(0～15)）
　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。
　　　　詳細は補足説明(7)参照。

　axis　データを書き込む軸。AXIS_X, AXIS_Y 等を指定。複数軸指定可能。
　　　　詳細は補足説明(2)参照。

　wdata　書き込むデータ

戻り値
　なし

使用例
　[VC]　　　Nmc_WriteReg2(No, IcNo, AXIS_Y, 0x0200);　// ALARM 有効(Ｙ軸)
　[VB.NET]　Call Nmc_WriteReg2(No, IcNo, AXIS_Y, &H200)　'ALARM 有効(Ｙ軸)</td></tr>
</table>

関数名	機能　及び　内容
Nmc_WriteReg3	ＷＲ３(モードレジスタ３)にデータを書き込む。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。 詳細は補足説明(7)参照。 axis　データを書き込む軸。AXIS_X, AXIS_Y 等を指定。複数軸指定可能。 詳細は補足説明(2)参照。 wdata　書き込むデータ **戻り値** なし **使用例** [VC]　　Nmc_WriteReg3(No, IcNo, AXIS_ALL, 0x0004);　// 全軸Ｓ字加減速設定 [VB.NET]　Call Nmc_WriteReg3(No, IcNo, AXIS_ALL, &H4)　'全軸Ｓ字加減速設定
Nmc_WriteReg4	ＷＲ４(アウトプットレジスタ１)にデータを書き込む。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 wdata　書き込むデータ **戻り値** なし **使用例** [VC]　　Nmc_WriteReg4(No, IcNo, 0x0001);　　// Ｘ軸汎用出力PIO0 Hiレベル出力 [VB.NET]　Call Nmc_WriteReg4(No, IcNo, &H1)　　'Ｘ軸汎用出力PIO0 Hiレベル出力
Nmc_WriteReg5	ＷＲ５(アウトプットレジスタ２)にデータを書き込む。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 wdata　書き込むデータ **戻り値** なし **使用例** [VC]　　Nmc_WriteReg5(No, IcNo, 0x0001);　　// Z軸汎用出力PIO0 Hiレベル出力 [VB.NET]　Call Nmc_WriteReg5(No, IcNo, &H1)　　'Z軸汎用出力PIO0 Hiレベル出力

関数名	機能　及び　内容
Nmc_WriteReg6	ＷＲ６(ライトデータレジスタ１)にデータを書き込む。 **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) 入力パラメータ No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 wdata　書き込むデータ 戻り値 なし 使用例 [VC]　　Nmc_WriteReg6(No, IcNo, 0x1234);　　// ライトデータレジスタ１にデータ(1234)Hを書く [VB.NET]　Call Nmc_WriteReg6(No, IcNo, &H1234)　'ライトデータレジスタ１にデータ(1234)Hを書く
Nmc_WriteReg7	ＷＲ７(ライトデータレジスタ２)にデータを書き込む。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 wdata　書き込むデータ **戻り値** なし **使用例** [VC]　　Nmc_WriteReg7(No, IcNo, 0x5678);　　// ライトデータレジスタ２にデータ(5678)Hを書く [VB.NET]　Call Nmc_WriteReg7(No, IcNo, &H5678)　'ライトデータレジスタ２にデータ(5678)Hを書く
Nmc_ReadReg0	ＲＲ０(主ステータスレジスタ)のデータを読み出す。 **VC**　　long　　Nmc_ReadReg0(int No, int IcNo); **VB.NET**　Function Nmc_ReadReg0(ByVal No As Integer, ByVal IcNo As Integer) As Integer **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 **戻り値** ＲＲ０(主ステータスレジスタ)のデータ **使用例** [VC]　　Data = Nmc_ReadReg0(No, IcNo);　　// ＲＲ０のデータを読む [VB.NET]　Data = Nmc_ReadReg0(No, IcNo)　　　'ＲＲ０のデータを読む
Nmc_ReadReg2	ＲＲ２(ステータスレジスタ２)のデータを読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。 詳細は補足説明(2)参照。 **戻り値** ＲＲ２(ステータスレジスタ２)のデータ **使用例** [VC]　　Data = Nmc_ReadReg2(No, IcNo, AXIS_Y);　　// Ｙ軸のＲＲ２のデータを読む [VB.NET]　Data = Nmc_ReadReg2(No, IcNo, AXIS_Y)　　　'Ｙ軸のＲＲ２のデータを読む

Nmc_ReadReg3	ＲＲ３(ステータスレジスタ３)のデータを読み出す。
	
	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
	
	入力パラメータ
	No ボード番号（ボード上のロータリースイッチの値(0～15))
	IcNo IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。
	axis データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。
	詳細は補足説明(2)参照。
	
	戻り値
	ＲＲ３(ステータスレジスタ３)のデータ
	
	使用例
	[VC] Nmc_Command(No, IcNo, AXIS_Y, MC8500P_CMD_RR3P0); // Ｙ軸のＲＲ３のページ０表示
	Data = Nmc_ReadReg3(No, IcNo, AXIS_Y); // Ｙ軸のＲＲ３のデータを読む
	
	[VB.NET]
	Call Nmc_Command(No, IcNo, AXIS_Y, MC8500P_CMD_RR3P0) 'Ｙ軸のＲＲ３のページ０表示
	Data = Nmc_ReadReg3(No, IcNo, AXIS_Y) 'Ｙ軸のＲＲ３のデータを読む
	
	注意
	ＲＲ３(ステータスレジスタ３)は、ページ０およびページ１の２種類存在します。読み出す前にＲＲ３ページ表示命令(7Ah、7Bh)を書き込むことでページの指定をします。リセット時はページ０となります。

Nmc_ReadReg3P	ＲＲ３(ステータスレジスタ３)のデータを読み出す。(ページ指定あり)
	
	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
	
	入力パラメータ
	No ボード番号（ボード上のロータリースイッチの値(0～15))
	IcNo IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。
	axis データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。
	詳細は補足説明(2)参照。
	Page データを読み出すページ、ページ0は0,ページ1は1。
	
	戻り値
	指定したページのＲＲ３(ステータスレジスタ３)データ
	
	使用例
	[VC] Data = Nmc_ReadReg3P(No, IcNo, AXIS_Y, 0); // Ｙ軸のＲＲ３のページ０のデータを読む
	[VB.NET] Data = Nmc_ReadReg3P(No, IcNo, AXIS_Y, 0) 'Ｙ軸のＲＲ３のページ０のデータを読む

Nmc_ReadReg4	ＲＲ４(ＰＩＯリードレジスタ１)のデータを読み出す。
	
	VC long Nmc_ReadReg4(int No, int IcNo);
	VB.NET Function Nmc_ReadReg4(ByVal No As Integer, ByVal IcNo As Integer) As Integer
	
	入力パラメータ
	No ボード番号（ボード上のロータリースイッチの値(0～15))
	IcNo IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。
	
	戻り値
	ＲＲ４(ＰＩＯリードレジスタ１)のデータ
	
	使用例
	[VC] Data = Nmc_ReadReg4(No, IcNo); // ＲＲ４のデータを読む
	[VB.NET] Data = Nmc_ReadReg4(No, IcNo) 'ＲＲ４のデータを読む

数名	機能　及び　内容
Nmc_ReadReg5	ＲＲ５(ＰＩＯリードレジスタ２)のデータを読み出す。 **VC**　　long　　Nmc_ReadReg5(int No, int IcNo); **VB.NET**　Function Nmc_ReadReg5(ByVal No As Integer, ByVal IcNo As Integer) As Integer **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 **戻り値** ＲＲ５(ＰＩＯリードレジスタ２)のデータ **使用例** [VC]　　　Data = Nmc_ReadReg5(No, IcNo);　　　// ＲＲ５のデータを読む [VB.NET]　Data = Nmc_ReadReg5(No, IcNo)　　　　'ＲＲ５のデータを読む
Nmc_ReadReg6	ＲＲ６(リードデータレジスタ１)のデータを読み出す。 **VC**　　long　　Nmc_ReadReg6(int No, int IcNo); **VB.NET**　Function Nmc_ReadReg6(ByVal No As Integer, ByVal IcNo As Integer) As Integer **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 **戻り値** ＲＲ６(リードデータレジスタ１)のデータ **使用例** [VC]　　　Data = Nmc_ReadReg6(No, IcNo);　　　// ＲＲ６のデータを読む [VB.NET]　Data = Nmc_ReadReg6(No, IcNo)　　　　'ＲＲ６のデータを読む
Nmc_ReadReg7	ＲＲ７(リードデータレジスタ２)のデータを読み出す。 **VC**　　long　　Nmc_ReadReg7(int No, int IcNo); **VB.NET**　Function Nmc_ReadReg7(ByVal No As Integer, ByVal IcNo As Integer) As Integer **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 **戻り値** ＲＲ７(リードデータレジスタ２)のデータ **使用例** [VC]　　　Data = Nmc_ReadReg7(No, IcNo);　　　// ＲＲ７のデータを読む [VB.NET]　Data = Nmc_ReadReg7(No, IcNo)　　　　'ＲＲ７のデータを読む
Nmc_Jerk	加速度増加率を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。AXIS_X, AXIS_Y 等を指定。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　　　Nmc_Jerk(No, IcNo, AXIS_X, 1000);　　　// 加速度増加率に 1000 を設定する(Ｘ軸) [VB.NET]　Call Nmc_Jerk(No, IcNo, AXIS_X, 1000)　　'加速度増加率に 1000 を設定する(Ｘ軸)

関数名	機能　及び　内容
Nmc_DJerk	減速度増加率を設定する。 **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) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。AXIS_X，AXIS_Y 等を指定。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　Nmc_DJerk(No, IcNo, AXIS_X, 1000);　// 減速度増加率に 1000 を設定する(Ｘ軸) [VB.NET]　Call Nmc_DJerk(No, IcNo, AXIS_X, 1000)　'減速度増加率に 1000 を設定する(Ｘ軸)
Nmc_Acc	加速度を設定する。 **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) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。AXIS_X，AXIS_Y 等を指定。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　Nmc_Acc(No, IcNo, AXIS_Y, 100);　// 加速度に 100 を設定する(Ｙ軸) [VB.NET]　Call Nmc_Acc(No, IcNo, AXIS_Y, 100)　'加速度に 100 を設定する(Ｙ軸)
Nmc_Dec	減速度を設定する。 **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) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　Nmc_Dec(No, IcNo, AXIS_Z, 100);　// 減速度に 100 を設定する(Ｚ軸) [VB.NET]　Call Nmc_Dec(No, IcNo, AXIS_Z, 100)　'減速度に 100 を設定する(Ｚ軸)

関数名	機能　及び　内容
Nmc_StartSpd	初速度を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　　　Nmc_StartSpd(No, IcNo, AXIS_U, 100);　　　// 初速度に 100 を設定する(Ｕ軸) [VB.NET]　Call Nmc_StartSpd(No, IcNo, AXIS_U, 100)　　'初速度に 100 を設定する(Ｕ軸)
Nmc_Speed	ドライブ速度を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** // ドライブ速度に 1000 を設定する(Ｘ,Ｙ軸) [VC]　　　Nmc_Speed(No, IcNo, AXIS_X | AXIS_Y, 1000); 'ドライブ速度に 1000 を設定する(Ｘ,Ｙ軸) [VB.NET]　Call Nmc_Speed(No, IcNo, AXIS_X or AXIS_Y, 1000)
Nmc_Pulse	出力パルス数、あるいは補間終点を設定する。(VC専用) 出力パルス数は、定量パルスドライブの総出力パルス数です。 直線補間、円弧補間ドライブの時は、各軸の終点を設定します。 ヘリカル補間ドライブ時は、Ｚ、Ｕ軸の送り量を設定します。 終点座標は、現在位置に対する相対値を指定します。 **VC**　　void Nmc_Pulse(int No, int IcNo, int axis, long wdata); **VB.NET**　使用できません **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　Nmc_Pulse(No, IcNo, AXIS_X, 2000);　　　// 出力パルス数に 2000 を設定する(Ｘ軸) 　　　Nmc_Pulse(No, IcNo, AXIS_Y,　300);　　　// 補間終点に　300 を設定する(Ｙ軸) 　　　Nmc_Pulse(No, IcNo, AXIS_Z, -400);　　　// 補間終点に -400 を設定する(Ｚ軸)

関数名	機能　及び　内容
Nmc_Pulse_VB	出力パルス数、あるいは補間終点を設定する。(VB専用) 出力パルス数は、定量パルスドライブの総出力パルス数です。 直線補間、円弧補間ドライブの時は、各軸の終点を設定します。 ヘリカル補間ドライブ時は、Ｚ、Ｕ軸の送り量を設定します。 終点座標は、現在位置に対する相対値を指定します。 **VC**　使用できません **VB.NET**　Sub Nmc_Pulse_VB(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Double) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VB.NET]　Call Nmc_Pulse_VB (No, IcNo, AXIS_X, 2000)　'出力パルス数に 2000 を設定する(Ｘ軸) Call Nmc_Pulse_VB (No, IcNo, AXIS_Y, 300)　'補間終点に 300 を設定する(Ｙ軸) Call Nmc_Pulse_VB (No, IcNo, AXIS_Z, -400)　'補間終点に -400 を設定する(Ｚ軸)
Nmc_DecP	マニュアル減速点を設定する。(VC専用) **VC**　void Nmc_DecP(int No, int IcNo, int axis, ULONG wdata); **VB.NET**　使用できません **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　Nmc_DecP(No, IcNo, AXIS_U, 30000);　// マニュアル減速点に 30000 を設定する(Ｕ軸)
Nmc_DecP_VB	マニュアル減速点を設定する。(VB専用) **VC**　使用できません **VB.NET**　Sub Nmc_DecP_VB(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer, ByVal wdata As Double) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata　設定するデータ **戻り値** なし **使用例** 'マニュアル減速点に 40000 を設定する(Ｘ軸) [VB.NET]　Call Nmc_DecP_VB(No, IcNo, AXIS_X, 40000)

関数名	機能　及び　内容
Nmc_Center	円弧中心点を設定する。 **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) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　Nmc_Center(No, IcNo, AXIS_Y, 1500);　// 円弧中心点に 1500 を設定する（Ｙ軸） [VB.NET]　Call Nmc_Center(No, IcNo, AXIS_Y, 1500)　'円弧中心点に 1500 を設定する（Ｙ軸）
Nmc_Lp	論理位置カウンタを設定する。 **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) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　Nmc_Lp(No, IcNo, AXIS_ALL, 0);　// 全軸の論理位置カウンタを０クリアする [VB.NET]　Call Nmc_Lp(No, IcNo, AXIS_ALL, 0)　'全軸の論理位置カウンタを０クリアする
Nmc_Ep	実位置カウンタを設定する。 **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) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　Nmc_Ep(No, IcNo, AXIS_ALL, 0);　// 全軸の実位置カウンタを０クリアする [VB.NET]　Call Nmc_Ep(No, IcNo, AXIS_ALL, 0)　'全軸の実位置カウンタを０クリアする

関数名	機能　及び　内容
Nmc_CompP	ソフトリミット＋レジスタを設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** // ソフトリミット＋レジスタに 50000 を設定する(Ｘ軸) [VC]　Nmc_CompP(No, IcNo, AXIS_X, 50000); 'ソフトリミット＋レジスタに 50000 を設定する(Ｘ軸) [VB.NET]　Call Nmc_CompP(No, IcNo, AXIS_X, 50000)
Nmc_CompM	ソフトリミット－レジスタを設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** // ソフトリミット－レジスタに -50000 を設定する(Ｘ軸) [VC]　　　Nmc_CompM(No, IcNo, AXIS_X, -50000); 'ソフトリミット－レジスタに -50000 を設定する(Ｘ軸) [VB.NET]　Call Nmc_CompM(No, IcNo, AXIS_X, -50000)
Nmc_AccOfst	加速カウンタオフセットを設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** // 加速カウンタオフセットに 20 を設定する(Ｙ軸) [VC]　　　Nmc_AccOfst(No, IcNo, AXIS_Y, 20); '加速カウンタオフセットに 20 を設定する(Ｙ軸) [VB.NET]　Call Nmc_AccOfst(No, IcNo, AXIS_Y, 20)

関数名	機能　及び　内容
Nmc_HomeSpd	原点検出速度を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　　　Nmc_HomeSpd(No, IcNo, AXIS_Z, 200);　　// 原点検出速度に 200 を設定する(Ｚ軸) [VB.NET]　Call Nmc_HomeSpd(No, IcNo, AXIS_Z, 200)　'原点検出速度に 200 を設定する(Ｚ軸)
Nmc_LpMax	論理位置カウンタ最大値を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** // 論理位置カウンタ最大値に 2000 を設定する(Ｕ軸) [VC]　　　Nmc_LpMax(No, IcNo, AXIS_U, 2000); '論理位置カウンタ最大値に 2000 を設定する(Ｕ軸) [VB.NET]　Call Nmc_LpMax(No, IcNo, AXIS_U, 2000)
Nmc_RpMax	実位置カウンタ最大値を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** //実位置カウンタ最大値に 1000 を設定する(Ｘ軸) [VC]　　　Nmc_RpMax(No, IcNo, AXIS_X, 1000); '実位置カウンタ最大値に 1000 を設定する(Ｘ軸) [VB.NET]　Call Nmc_RpMax(No, IcNo, AXIS_X, 1000)

関数名	機能　及び　内容
Nmc_MR0	多目的レジスタ０を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　　Nmc_MR0(No, IcNo, AXIS_Z, 500000);　// 多目的レジスタ０に 500000 を設定する(Ｚ軸) [VB.NET]　Call Nmc_MR0(No, IcNo, AXIS_Z, 500000) '多目的レジスタ０に 500000 を設定する(Ｚ軸)
Nmc_MR1	多目的レジスタ１を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　　Nmc_MR1(No, IcNo, AXIS_U, 5000);　// 多目的レジスタ１に 5000 を設定する(Ｕ軸) [VB.NET]　Call Nmc_MR1(No, IcNo, AXIS_U, 5000)　'多目的レジスタ１に 5000 を設定する(Ｕ軸)
Nmc_MR2	多目的レジスタ２を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** //多目的レジスタ２に 5000000 を設定する(Ｘ軸) [VC]　　Nmc_MR2(No, IcNo, AXIS_X, 5000000); '多目的レジスタ２に 5000000 を設定する(Ｘ軸) [VB.NET]　Call Nmc_MR2(No, IcNo, AXIS_X, 5000000)

関数名	機能　及び　内容
Nmc_MR3	多目的レジスタ３を設定する。 **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) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata　設定するデータ **戻り値** なし **使用例** //多目的レジスタ３に 30000 を設定する(Ｙ軸) [VC]　Nmc_MR3(No, IcNo, AXIS_Y, 30000); '多目的レジスタ３に 30000 を設定する(Ｙ軸) [VB.NET]　Call Nmc_MR3(No, IcNo, AXIS_Y, 30000)
Nmc_SpeedInc	速度増減値を設定する。 **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) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　Nmc_SpeedInc(No, IcNo, AXIS_Z, 100);　// 速度増減値に 100 を設定する(Ｚ軸) [VB.NET]　Call Nmc_SpeedInc(No, IcNo, AXIS_Z, 100)　'速度増減値に 100 を設定する(Ｚ軸)
Nmc_Timer	タイマー値を設定する。 **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) **入力パラメータ** No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　Nmc_Timer(No, IcNo, AXIS_U, 10000);　// タイマー値に 10000 を設定する(Ｕ軸) [VB.NET]　Call Nmc_Timer(No, IcNo, AXIS_U, 10000)　'タイマー値に 10000 を設定する(Ｕ軸)

関数名	機能　及び　内容
Nmc_TPMax	補間・終点最大値を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　　　Nmc_TPMax(No, IcNo, 10000);　　　// 補間・終点最大値に 10000 を設定する [VB.NET]　Call Nmc_TPMax(No, IcNo, 10000)　　'補間・終点最大値に 10000 を設定する
Nmc_HLNumber	ヘリカル回転数を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　　　Nmc_HLNumber(No, IcNo, 7);　　　// ヘリカル回転数に 7 を設定する [VB.NET]　Call Nmc_HLNumber(No, IcNo, 7)　　'ヘリカル回転数に 7 を設定する
Nmc_HLValue	ヘリカル演算値を設定する。 **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 wdata 設定するデータ **戻り値** なし **使用例** [VC]　　　Nmc_HLValue(No, IcNo, 36799);　　　// ヘリカル演算値に 36799 を設定する [VB.NET]　Call Nmc_HLValue(No, IcNo, 36799)　　'ヘリカル演算値に 36799 を設定する
Nmc_MRmMode	多目的レジスタモードを設定する。 **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) **入力パラメータ** No　　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 axis　　　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 WR6_data　設定するデータ **戻り値** なし **使用例** Ｘ軸多目的レジスタモードにＭＲ１の比較対象(現在速度)、比較条件(>)を設定する [VC]　　　Nmc_MRmMode(No, IcNo, AXIS_X, 0x0060); [VB.NET]　Call Nmc_MRmMode(No, IcNo, AXIS_X, &H60)

関数名	機能　及び　内容
Nmc_PIO1Mode	ＰＩＯ信号設定１(nPIO7～0の機能)を設定する。 **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) **入力パラメータ** No　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 axis　　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 WR6_data　設定するデータ **戻り値** なし **使用例** Ｘ軸ＰＩＯ信号０～３は汎用出力を設定する [VC]　　　Nmc_PIO1Mode(No, IcNo, AXIS_X, 0x0055); [VB.NET]　Call Nmc_PIO1Mode(No, IcNo, AXIS_X, &H55)
Nmc_PIO2Mode	ＰＩＯ信号設定２・その他(同期パルス出力の論理、パルス幅等)を設定する。 **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) **入力パラメータ** No　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 axis　　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 WR6_data　設定するデータ **戻り値** なし **使用例** Ｘ軸パルス信号を正論理、パルス幅を1msecに設定する。 [VC]　　　Nmc_PIO2Mode(No, IcNo, AXIS_X, 0x0070); [VB.NET]　Call Nmc_PIO2Mode(No, IcNo, AXIS_X, &H70)
Nmc_HMSrch1Mode	自動原点出しモード設定１を設定する。 **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) **入力パラメータ** No　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 axis　　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 WR6_data　設定するデータ **戻り値** なし **使用例** Ｘ軸自動原点出しモード設定１に、以下の設定をする。 ・ステップ１,2,3,4：実行 ・ステップ1,2,3の検出方向：－方向 ・ステップ1,2の検出信号：STOP1 ・ステップ2のDCC出力,RPクリア,LPクリア：無効 ・ステップ3のDCC出力,RPクリア,LPクリア：有効 [VC]　　　Nmc_HMSrch1Mode(No, IcNo, AXIS_X, 0xFC37); [VB.NET]　Call Nmc_HMSrch1Mode(No, IcNo, AXIS_X, &HFC37)

関数名	機能　及び　内容
Nmc_HMSrch2Mode	自動原点出しモード設定２を設定する。 **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) **入力パラメータ** No　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 axis　　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 WR6_data　設定するデータ **戻り値** なし **使用例** Ｘ軸自動原点出しモード設定２に、以下の設定をする。 ・ステップ2&3：無効 ・原点出し終了時LPクリア：有効 ・原点出し終了時RPクリア：有効 ・DCCパルス論理：10μsec ・DCCパルス幅：Hiパルス ・ステップ間タイマー：無効 ・タイマー値：0 [VC]　　　Nmc_HMSrch1Mode(No, IcNo, AXIS_X, 0x0006); [VB.NET]　Call Nmc_HMSrch2Mode(No, IcNo, AXIS_X, &H6)
Nmc_FilterMode	入力信号フィルタモードを設定する。 **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) **入力パラメータ** No　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 axis　　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 WR6_data　設定するデータ **戻り値** なし **使用例** Ｘ軸入力信号フィルタモードに、全フィルタ有効、遅延512μsを設定する [VC]　　　Nmc_FilterMode(No, IcNo, AXIS_X, 0xAAFF); [VB.NET]　Call Nmc_FilterMode(No, IcNo, AXIS_X, &HAAFF)

関数名	機能　及び　内容
Nmc_Sync0Mode	同期動作ＳＹＮＣ０を設定する。 **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) **入力パラメータ** No　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 axis　　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 WR6_data　設定するデータ **戻り値** なし **使用例** 「Ｘ軸ドライブ中に加減速ドライブの定速域が開始したら、論理位置カウンタをMR0にセーブする」を設定する [VC]　　Nmc_Sync0Mode(No, IcNo, AXIS_X, 0x0054); [VB.NET]　Call Nmc_Sync0Mode(No, IcNo, AXIS_X, &H54)
Nmc_Sync1Mode	同期動作ＳＹＮＣ１を設定する。 **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) **入力パラメータ** No　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 axis　　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 WR6_data　設定するデータ **戻り値** なし **使用例** Ｘ軸ドライブ中にMR1対象が真に変化したら、現在タイマー値をMR1にセットする」を設定する [VC]　　Nmc_Sync1Mode(No, IcNo, AXIS_X, 0x0071); [VB.NET]　Call Nmc_Sync1Mode(No, IcNo, AXIS_X, &H71)

関数名	機能　及び　内容
Nmc_Sync2Mode	同期動作ＳＹＮＣ２を設定する。 **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) **入力パラメータ** 　No　　　ボード番号（ボード上のロータリースイッチの値(0～15)) 　IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 　axis　　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 　WR6_data　設定するデータ **戻り値** 　なし **使用例** 　「Ｘ軸ドライブ中に内蔵タイマーがタイムアップしたら、論理位置カウンタをMR2セーブし、ＳＹＮＣ３を起動する」を設定する 　[VC]　　Nmc_Sync2Mode(No, IcNo, AXIS_X, 0x0252); 　[VB.NET]　Call Nmc_Sync2Mode(No, IcNo, AXIS_X, &H252)
Nmc_Sync3Mode	同期動作ＳＹＮＣ３を設定する。 **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) **入力パラメータ** 　No　　　ボード番号（ボード上のロータリースイッチの値(0～15)) 　IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 　axis　　データを設定する軸。複数軸指定可能。詳細は補足説明(2)参照。 　WR6_data　設定するデータ **戻り値** 　なし **使用例** 　「Ｘ軸ドライブ中にMR3対象が真に変化したら、論理位置カウンタをMR3にセーブし、Ｙ軸ＳＹＮＣ０を起動する」を設定する 　[VC]　　Nmc_Sync3Mode(No, IcNo, AXIS_X, 0x1051); 　[VB.NET]　Call Nmc_Sync3Mode(No, IcNo, AXIS_X, &H1051)
Nmc_IPMode	補間モードを設定する。 **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) **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)) 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照 　wdata　設定するデータ **戻り値** 　なし **使用例** 　補間軸Ｘ,Ｙ,Ｚを指定する 　[VC]　　Nmc_IPMode(No, IcNo, 0x0007); 　[VB.NET]　Call Nmc_IPMode(No, IcNo, &H7)

関数名	機能　及び　内容
Nmc_ReadLp	論理位置カウンタを読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X，Ｙ軸はAXIS_Y，Ｚ軸はAXIS_Z，Ｕ軸はAXIS_Uを指定する。詳細は補足説明(2)参照。 **戻り値** 現在の論理位置カウンタの値 **使用例** [VC]　　Data = Nmc_ReadLp(No, IcNo, AXIS_X);　// Ｘ軸の論理位置カウンタを読み出す [VB.NET]　Data = Nmc_ReadLp(No, IcNo, AXIS_X)　　'Ｘ軸の論理位置カウンタを読み出す
Nmc_ReadEp	実位置カウンタを読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X，Ｙ軸はAXIS_Y，Ｚ軸はAXIS_Z，Ｕ軸はAXIS_Uを指定する。詳細は補足説明(2)参照。 **戻り値** 現在の実位置カウンタの値 **使用例** [VC]　　Data = Nmc_ReadEp(No, IcNo, AXIS_Y);　// Ｙ軸の実位置カウンタを読み出す [VB.NET]　Data = Nmc_ReadEp(No, IcNo, AXIS_Y)　　'Ｙ軸の実位置カウンタを読み出す
Nmc_ReadSpeed	現在ドライブ速度を読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X，Ｙ軸はAXIS_Y，Ｚ軸はAXIS_Z，Ｕ軸はAXIS_Uを指定する。詳細は補足説明(2)参照。 **戻り値** 現在ドライブ速度の値 **使用例** [VC]　Data = Nmc_ReadSpeed(No, IcNo, AXIS_Z);　　// Ｚ軸の現在ドライブ速度を読み出す [VB.NET]　Data = Nmc_ReadSpeed(No, IcNo, AXIS_Z)　　'Ｚ軸の現在ドライブ速度を読み出す

関数名	機能　及び　内容
Nmc_ReadAccDec	現在加／減速度を読み出す。 ドライブ中の現在加速度、または減速度の値を読み出す。 ドライブ停止時の読み出しデータは不定です。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。詳細は補足説明(2)参照。 **戻り値** 現在加／減速度の値 **使用例** [VC]　　Data = Nmc_ReadAccDec(No, IcNo, AXIS_U);　// Ｕ軸の現在加／減速度を読み出す [VB.NET]　Data = Nmc_ReadAccDec(No, IcNo, AXIS_U)　　'Ｕ軸の現在加／減速度を読み出す
Nmc_ReadMR0	多目的レジスタ０を読み出す。 **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 Integer **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。詳細は補足説明(2)参照。 **戻り値** 多目的レジスタ０の値 **使用例** [VC]　　Data = Nmc_ReadMR0(No, IcNo, AXIS_X);　// Ｘ軸の多目的レジスタ０を読み出す [VB.NET]　Data = Nmc_ReadMR0(No, IcNo, AXIS_X)　　'Ｘ軸の多目的レジスタ０を読み出す
Nmc_ReadMR1	多目的レジスタ１を読み出す。 **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 Integer **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。詳細は補足説明(2)参照。 **戻り値** 多目的レジスタ１の値 **使用例** [VC]　　Data = Nmc_ReadMR1(No, IcNo, AXIS_Y);　// Ｙ軸の多目的レジスタ１を読み出す [VB.NET]　Data = Nmc_ReadMR1(No, IcNo, AXIS_Y)　　'Ｙ軸の多目的レジスタ１を読み出す

関数名	機能　及び　内容
Nmc_ReadMR2	多目的レジスタ２を読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X，Ｙ軸はAXIS_Y，Ｚ軸はAXIS_Z，Ｕ軸はAXIS_Uを指定する。 詳細は補足説明(2)参照。 **戻り値** 多目的レジスタ２の値 **使用例** [VC]　　　Data = Nmc_ReadMR2(No, IcNo, AXIS_Z);　　// Ｚ軸の多目的レジスタ２を読み出す [VB.NET]　Data = Nmc_ReadMR2(No, IcNo, AXIS_Z)　　　'Ｚ軸の多目的レジスタ２を読み出す
Nmc_ReadMR3	多目的レジスタ３を読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X，Ｙ軸はAXIS_Y，Ｚ軸はAXIS_Z，Ｕ軸はAXIS_Uを指定する。 詳細は補足説明(2)参照。 **戻り値** 多目的レジスタ３の値 **使用例** [VC]　　　Data = Nmc_ReadMR3(No, IcNo, AXIS_U);　　// Ｕ軸の多目的レジスタ３を読み出す [VB.NET]　Data = Nmc_ReadMR3(No, IcNo, AXIS_U)　　　'Ｕ軸の多目的レジスタ３を読み出す
Nmc_ReadCT	現在タイマー値を読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0，2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X，Ｙ軸はAXIS_Y，Ｚ軸はAXIS_Z，Ｕ軸はAXIS_Uを指定する。 詳細は補足説明(2)参照。 **戻り値** 現在タイマー値の値 **使用例** [VC]　　　Data = Nmc_ReadCT(No, IcNo, AXIS_X);　　// Ｘ軸の現在タイマー値を読み出す [VB.NET]　Data = Nmc_ReadCT(No, IcNo, AXIS_X)　　　'Ｘ軸の現在タイマー値を読み出す

関数名	機能　及び　内容
Nmc_ReadTX	補間・終点最大値を読み出す。 **VC**　　　long　　　Nmc_ReadTX(int No, int IcNo); **VB.NET**　Function Nmc_ReadTX(ByVal No As Integer, ByVal IcNo As Integer) As Integer **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 **戻り値** 　補間・終点最大値の値 **使用例** 　[VC]　　　Data = Nmc_ReadTX(No, IcNo);　　// 補間・終点最大値を読み出す 　[VB.NET]　Data = Nmc_ReadTX(No, IcNo)　　　'補間・終点最大値を読み出す **注意** 　補間ドライブの実行前と、実行中では読み出される値が異なります。補間ドライブ実行前は、入力中の補間セグメントの最終最大値が読み出されます。補間ドライブ実行中は現在実行中の補間セグメントの終点最大値が読み出されます。
Nmc_ReadCHLN	現在ヘリカル回転数を読み出す **VC**　　　long　　　Nmc_ReadCHLN(int No, int IcNo); **VB.NET**　Function Nmc_ReadCHLN(ByVal No As Integer, ByVal IcNo As Integer) As Integer **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 **戻り値** 　現在ヘリカル回転数の値 **使用例** 　[VC]　　　Data = Nmc_ReadCHLN(No, IcNo);　　// 現在ヘリカル回転数を読み出す 　[VB.NET]　Data = Nmc_ReadCHLN(No, IcNo)　　　'現在ヘリカル回転数を読み出す
Nmc_ReadHLV	ヘリカル演算値を読み出す。 ヘリカル演算命令(6Bh,6Ch)でヘリカル演算を行った結果を読み出すときに使用します。 **VC**　　　long　　　Nmc_ReadHLV(int No, int IcNo); **VB.NET**　Function Nmc_ReadHLV(ByVal No As Integer, ByVal IcNo As Integer) As Integer **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 **戻り値** 　ヘリカル演算値の値 **使用例** 　[VC]　　　Data = Nmc_ReadHLV(No, IcNo);　　// ヘリカル演算値を読み出す 　[VB.NET]　Data = Nmc_ReadHLV(No, IcNo)　　　'ヘリカル演算値を読み出す

関数名	機能　及び　内容
Nmc_ReadWR1	ＷＲ１設定値を読み出す。 **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 **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 　axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。 　　　　詳細は補足説明(2)参照。 **戻り値** 　ＷＲ１の値 **使用例** 　[VC]　　　Data = Nmc_ReadWR1(No, IcNo, AXIS_Y);　　// Ｙ軸のＷＲ１設定値を読み出す [VB.NET]　Data = Nmc_ReadWR1(No, IcNo, AXIS_Y)　　　　’Ｙ軸のＷＲ１設定値を読み出す
Nmc_ReadWR2	ＷＲ２設定値を読み出す。 **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 **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 　axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。 　　　　詳細は補足説明(2)参照。 **戻り値** 　ＷＲ２の値 **使用例** 　[VC]　　　Data = Nmc_ReadWR2(No, IcNo, AXIS_Z);　　// Ｚ軸のＷＲ２設定値を読み出す 　[VB.NET]　Data = Nmc_ReadWR2(No, IcNo, AXIS_Z)　　　’Ｚ軸のＷＲ２設定値を読み出す
Nmc_ReadWR3	ＷＲ３設定値を読み出す。 **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 **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 　axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。 　　　　詳細は補足説明(2)参照。 **戻り値** 　ＷＲ３の値 **使用例** 　[VC]　　　Data = Nmc_ReadWR3(No, IcNo, AXIS_U);　　// Ｕ軸のＷＲ３設定値を読み出す 　[VB.NET]　Data = Nmc_ReadWR3(No, IcNo, AXIS_U)　　　’Ｕ軸のＷＲ３設定値を読み出す

関数名	機能　及び　内容
Nmc_ReadMRM	多目的レジスタモード設定を読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。 　　　詳細は補足説明(2)参照。 **戻り値** 多目的レジスタモードの値 **使用例** [VC]　Data = Nmc_ReadMRM(No, IcNo, AXIS_X);　// Ｘ軸の多目的レジスタモード設定値を読み出す [VB.NET]　Data = Nmc_ReadMRM(No, IcNo, AXIS_X)　'Ｘ軸の多目的レジスタモード設定値を読み出す
Nmc_ReadP1M	ＰＩＯ信号設定１を読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。 　　　詳細は補足説明(2)参照。 **戻り値** ＰＩＯ信号設定１の値 **使用例** [VC]　Data = Nmc_ReadP1M(No, IcNo, AXIS_Y);　　// ＹのＰＩＯ信号設定１設定値を読み出す [VB.NET]　Data = Nmc_ReadP1M(No, IcNo, AXIS_Y)　　'Ｙ軸のＰＩＯ信号設定１設定値を読み出す
Nmc_ReadP2M	ＰＩＯ信号設定２・その他設定を読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。 　　　詳細は補足説明(2)参照。 **戻り値** ＰＩＯ信号設定２・その他設定の値 **使用例** [VC]　　Data = Nmc_ReadP2M(No, IcNo, AXIS_Z); 　　　　　　　　　// Ｚ軸のＰＩＯ信号設定２・その他設定値を読み出す [VB.NET]　Data = Nmc_ReadP2M(No, IcNo, AXIS_Z) 　　　　　　　　　'Ｚ軸のＰＩＯ信号設定２・その他設定値を読み出す

関数名	機能　及び　内容
Nmc_ReadAc	加速度設定値を読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。詳細は補足説明(2)参照。 **戻り値** 設定した加速度の値 **使用例** [VC]　　　Data = Nmc_ReadAc(No, IcNo, AXIS_U);　// Ｕ軸の加速度設定値を読み出す [VB.NET]　Data = Nmc_ReadAc(No, IcNo, AXIS_U)　　'Ｕ軸の加速度設定値を読み出す
Nmc_ReadStartSpd	初速度設定値を読み出す。 **VC**　　long　　Nmc_ReadStartSpd(int No, int IcNo, int axis); **VB.NET**　Function Nmc_ReadStartSpd(ByVal No As Integer, ByVal IcNo As Integer, ByVal axis As Integer) As Integer **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。詳細は補足説明(2)参照。 **戻り値** 設定した初速度の値 **使用例** [VC]　　　Data = Nmc_ReadStartSpd(No, IcNo, AXIS_X);　// Ｘ軸の初速度設定値を読み出す [VB.NET]　Data = Nmc_ReadStartSpd(No, IcNo, AXIS_X)　　'Ｘ軸の初速度設定値を読み出す
Nmc_ReadSetSpeed	ドライブ速度設定値を読み出す。 **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 **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。詳細は補足説明(2)参照。 **戻り値** 設定したドライブ速度の値 **使用例** [VC]　　　Data = Nmc_ReadSetSpeed(No, IcNo, AXIS_Y);　// Ｙ軸のドライブ速度設定値を読み出す [VB.NET]　Data = Nmc_ReadSetSpeed(No, IcNo, AXIS_Y)　　'Ｙ軸のドライブ速度設定値を読み出す

関数名	機能　及び　内容
Nmc_ReadPulse	移動パルス数/終点設定値を読み出す。 **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 **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 　axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。 　　　　詳細は補足説明(2)参照。 **戻り値** 　設定した移動パルス数/終点設定値の値 **使用例** 　[VC]　Data = Nmc_ReadPulse(No, IcNo, AXIS_Z);　　// Ｚ軸の移動パルス数/終点設定値を読み出す 　[VB.NET]　Data = Nmc_ReadPulse(No, IcNo, AXIS_Z)　　'Ｚ軸の移動パルス数/終点設定値を読み出す
Nmc_GetDriveStatus	ドライブ状態を取得する。指定軸のドライブが終了したか調べる時に使用する。 **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 **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 　axis　ドライブ状態を取得する軸。複数軸指定可能。 　　　　詳細は補足説明(2)参照。 **戻り値** 　指定した全ての軸のドライブが終了している場合は０を返す。 　指定した軸のうち１つ以上がドライブ中の場合は、０以外を返す。 **使用例** 　[VC]　if(Nmc_GetDriveStatus(No, IcNo, AXIS_X) == 0)　　// Ｘ軸がドライブ終了の場合 　　　　　AfxMessageBox("Ｘ軸ドライブ終了"); 　　　　else 　　　　　AfxMessageBox("Ｘ軸ドライブ中"); 　[VB.NET] 　　If Nmc_GetDriveStatus(No, IcNo, AXIS_X) = 0 Then　　'Ｘ軸がドライブ終了の場合 　　　Call MsgBox("Ｘ軸ドライブ終了") 　　Else 　　　Call MsgBox("Ｘ軸ドライブ中") 　　End If

<table>
<tr><th>関数名</th><th>機能　及び　内容</th></tr>
<tr><td>Nmc_GetCNextStatus</td><td>連続補間次データ書込み可能状態を取得する。
連続補間実行中に次データ書き込み可能になったか調べる時に使用する。

VC　　int　　Nmc_GetCNextStatus(int No, int IcNo);
VB.NET　Function Nmc_GetCNextStatus(ByVal No As Integer, ByVal IcNo As Integer) As Integer

入力パラメータ
　No　　ボード番号（ボード上のロータリースイッチの値(0～15)）
　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。
　詳細は補足説明(7)参照。

戻り値
　連続補間次データ書込み可能の場合は、０以外を返す。
　連続補間次データ書込み可能ではない場合は、０を返す。

使用例
　[VC]　if(Nmc_GetCNextStatus(No, IcNo) != 0)　// 次データ書込み可能の場合
　　　　　AfxMessageBox("連続補間次データ書込み可能である");
　　　　else
　　　　　AfxMessageBox("連続補間次データ書込み可能ではない");

　[VB.NET]
　　If Nmc_GetCNextStatus(No, IcNo) <> 0 Then　'次データ書込み可能の場合
　　　Call MsgBox("連続補間次データ書込み可能である")
　　Else
　　　Call MsgBox("連続補間次データ書込み可能ではない")
　　End If</td></tr>
<tr><td>Nmc_GetSc</td><td>連続補間プリバッファスタックカウンタの値を取得する。

VC　　int　　Nmc_GetSc(int No, int IcNo);
VB.NET　Function Nmc_GetSc(ByVal No As Integer, ByVal IcNo As Integer) As Integer

入力パラメータ
　No　　ボード番号（ボード上のロータリースイッチの値(0～15)）
　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。
　詳細は補足説明(7)参照。

戻り値
　現在のプリバッファスタックカウンタの値

使用例
　[VC]　　Data = Nmc_GetSc(No, IcNo);　　// プリバッファスタックカウンタの値を取得
　[VB.NET]　Data = Nmc_GetSc(No, IcNo)　　'プリバッファスタックカウンタの値を取得</td></tr>
<tr><td>Nmc_WriteRegSetAxis</td><td>指定軸の指定したライトレジスタ(ＷＲ１～ＷＲ３のいずれか)にデータを書き込む。

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)

入力パラメータ
　No　　　　ボード番号（ボード上のロータリースイッチの値(0～15)）
　IcNo　　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。
　　　　　　詳細は補足説明(7)参照
　axis　　　データを書き込む軸。複数軸指定可能。詳細は補足説明(2)参照。
　RegNumber データを書き込むライトレジスタの番号
　　　　　　[VC]　WR1はMCX_WR1, WR2はMCX_WR2, WR3はMCX_WR3 を指定する。
　　　　　　[VB.NET]　WR1はMCX_WR1, WR2はMCX_WR2, WR3はMCX_WR3 を指定する。
　wdata　　書き込むデータ

戻り値
　なし

使用例　　ハードリミット有効を設定する
　[VC]　　Nmc_WriteRegSetAxis(No, IcNo, AXIS_ALL, MCX_WR2, 0x0800);
　[VB.NET]　Call Nmc_WriteRegSetAxis(No, IcNo, AXIS_ALL, MCX_WR2, &H800)</td></tr>
</table>

関数名	機能　及び　内容
Nmc_ReadRegSetAxis	指定軸の指定したリードレジスタ(ＲＲ２、ＲＲ３のどちらか)のデータを読み出す。 **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 **入力パラメータ** No　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。 詳細は補足説明(7)参照 axis　　データを読み出す軸。 Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。 詳細は補足説明(2)参照。 RegNumber データを読み出すリードレジスタの番号 [VC]　RR1はMCX_RR1, RR2はMCX_RR2 を指定する。 [VB.NET]　RR1はMCX_RR1, RR2はMCX_RR2 を指定する。 **戻り値** 指定軸の指定したリードレジスタのデータ **使用例**　Ｘ軸のＲＲ２のデータを読み出す [VC]　　Data = Nmc_ReadRegSetAxis(No, IcNo, AXIS_X, MCX_RR2); [VB.NET]　Data = Nmc_ReadRegSetAxis(No, IcNo, AXIS_X, MCX_RR2) **注意** ＲＲ３(ステータスレジスタ３)は、ページ０およびページ１の２種類存在します。読み出す前にＲＲ３ページ表示命令(7Ah、7Bh)を書き込むことでページの指定をします。リセット時はページ０となります。
Nmc_WriteData	指定軸に指定したパラメータのデータを書き込む。(データ書き込み命令を実行する) **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) **入力パラメータ** No　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 axis　データを書き込む軸。複数軸指定可能。詳細は補足説明(2)参照。 cmd　書き込み命令コード((00)H～(16)H,(19)H～(1B)H)。例：加速度増加率設定は(00)H。 wdata 書き込むデータ **戻り値** なし **使用例**　全軸のドライブ速度に1000を設定する。ドライブ速度の命令コードは(05)H [VC]　　Nmc_WriteData(No, IcNo, AXIS_ALL, 0x05, 1000); [VB.NET]　Call Nmc_WriteData(No, IcNo, AXIS_ALL, &H5, 1000)

関数名	機能　及び　内容
Nmc_ReadData	データ読み出し命令を実行し、データを読み出す。 **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 **入力パラメータ** 　No　　ボード番号（ボード上のロータリースイッチの値(0～15)） 　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。 　axis　データを読み出す軸。Ｘ軸はAXIS_X, Ｙ軸はAXIS_Y, Ｚ軸はAXIS_Z, Ｕ軸はAXIS_Uを指定する。 　cmd　データ読み出し命令コード((30)H～(46)H)。例：論理位置カウンタ読み出しは(30)H。 **戻り値** 　読み出したデータ **使用例**　　Ｘ軸の論理位置カウンタを読み出す。 　[VC]　　　Data = Nmc_ReadData(No, IcNo, AXIS_X, 0x30); 　[VB.NET]　Data = Nmc_ReadData(No, IcNo, AXIS_X, &H30)

関数名	機能　及び　内容

Nmc_2BPExecMC8500P	指定した補間データで２軸ビットパターン補間を実行する。
	この関数は、補間処理が終了した後に制御を返します。補間が終了するまで制御が戻らないのでアプリケーションでスレッドを作成し、そのスレッドからコールする事を推奨します。
	
	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
	
	入力パラメータ
	No　ボード番号（ボード上のロータリースイッチの値(0～15)）
	IcNo　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照
	pData2Bp　２軸ＢＰ補間データの構造体(ユーザー定義型)配列の先頭アドレス(DATA_2BPのアドレス)。DATA_2BPに実行する補間データをセットし、アドレスを指定する。DATA_2BPについては補足説明(3)参照。
	DataCnt　２軸ＢＰ補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。
	IpAxis　補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。補足説明(4)参照。
	
	戻り値
	補間処理が正常終了した場合は BP_END が返ります。
	補間処理でエラーが発生した場合は、下記エラーコードが返ります。
	
	■正常終了
	BP_END　BP補間処理正常終了
	
	■エラーコード
	BP_CNT_ERR　指定されたデータ数が範囲外
	BP_ALREADY_EXEC　既にＢＰ補間、あるいは連続補間が実行中
	BP_MALLOC_ERR　メモリを確保できなかった
	BP_PARAM_ERR　引数の値が正しくない
	BP_NOT_OPEN_ERR　指定したボードがオープンされていない
	BP_OTHER_ERR　その他のエラー
	BP_FUNC_ERR　使用できない関数を使用
	BP_STOP　BP補間が途中で停止した（速度が速く次データのスタックが間に合わなかった場合）
	BP_USER_STOP　BP補間実行中にユーザーが中断した
	BP_DRIVE_ERR　BP補間実行中にボードでエラー発生（RR0にエラー情報がセットされた）
	BP_STOP_CERR　連続補間が途中で停止した（RR2エラーによる停止の場合）
	BP_GETSC_ERR　スタックカウンタ読み出しエラー
	
	補間データに終了コードが書かれていて補間が終了した場合は、BP_ENDまたはBP_STOPのいずれかのエラーコードが返ります。
	
	使用例
	[VC]
	// ２軸ＢＰ補間データ　BP1P,　BP1M,　BP2P,　BP2M
	DATA_2BP Data2Bp[2] = {{0x0000, 0x2BFF, 0xFFD4, 0x0000},
	{0xF6FE, 0x0000, 0x000F, 0x3FC0}};
	
	Nmc_IPMode(No, IcNo,0x0003);　//補間モード設定。Ｘ，Ｙ軸指定。
	
	// 速度関連パラメータ設定(実際の設定記述は省略)
	
	Ret = Nmc_2BPExecMC8500P(No, IcNo, Data2Bp, 2, 0x3); //２軸ＢＰ補間実行。データ数2, X,Y軸
	if(Ret == BP_END)　AfxMessageBox("正常終了");　//戻り値正常

```
[VB.NET]
    '２軸ＢＰ補間データ
    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)                          '補間モード設定。Ｘ，Ｙ軸指定。

    '速度関連パラメータ設定(実際の設定記述は省略)

    Ret = Nmc_2BPExecMC8500P(No, IcNo, Data2Bp, 2, &H3) '２軸ＢＰ補間実行。データ数数2, X,Y軸
    If Ret = BP_END Then    '戻り値正常
        Call MsgBox("正常終了")
    End If
```

注意

この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸などを設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。

この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

関数名	機能　及び　内容

Nmc_3BPExecMC8500P	指定した補間データで３軸ビットパターン補間を実行する。
	この関数は、補間処理が終了した後に制御を返します。補間が終了するまで制御が戻らないのでアプリケーションでスレッドを作成し、そのスレッドからコールする事を推奨します。
	
	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
	
	入力パラメータ
	No　　　　ボード番号（ボード上のロータリースイッチの値(0～15)）
	IcNo　　　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照
	pData3Bp　３軸ＢＰ補間データの構造体(ユーザー定義型)配列の先頭アドレス(DATA_3BPのアドレス)。
	　　　　　DATA_3BPに実行する補間データをセットし、アドレスを指定する。
	　　　　　DATA_3BPについては補足説明(3)参照。
	DataCnt　３軸ＢＰ補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。
	IpAxis　　補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。
	　　　　　補足説明(4)参照。
	
	戻り値
	補間処理が正常終了した場合は BP_END が返ります。
	補間処理でエラーが発生した場合は、下記エラーコードが返ります。
	
	■正常終了
	BP_END　　　　BP補間処理正常終了
	
	■エラーコード
	BP_CNT_ERR　　　指定されたデータ数が範囲外
	BP_ALREADY_EXEC 既にＢＰ補間、あるいは連続補間が実行中
	BP_MALLOC_ERR　メモリを確保できなかった
	BP_PARAM_ERR　　引数の値が正しくない
	BP_NOT_OPEN_ERR 指定したボードがオープンされていない
	BP_OTHER_ERR　　その他のエラー
	BP_FUNC_ERR　　使用できない関数を使用
	BP_STOP　　　　BP補間が途中で停止した（速度が速く次データのスタックが間に合わなかった場合）
	BP_USER_STOP　　BP補間実行中にユーザーが中断した
	BP_DRIVE_ERR　　BP補間実行中にボードでエラー発生（RR0にエラー情報がセットされた）
	BP_STOP_CERR　　連続補間が途中で停止した（RR2エラーによる停止の場合）
	BP_GETSC_ERR　　スタックカウンタ読み出しエラー
	
	補間データに終了コードが書かれていて補間が終了した場合は、BP_ENDまたはBP_STOPのいずれかのエラーコードが返ります。
	
	使用例
	[VC]
	　// ３軸ＢＰ補間データ　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);　　　　　　　　//補間モード設定。Ｘ,Ｙ,Ｚ軸指定。
	
	　// 速度関連パラメータ設定(実際の設定記述は省略)
	
	　Ret = Nmc_3BPExecMC8500P(No, IcNo, Data3Bp, 2, 0x7); //３軸ＢＰ補間実行。データ数2, X,Y,Z軸
	　if(Ret == BP_END)　AfxMessageBox("正常終了");　　//戻り値正常

	[VB.NET] ``` '３軸ＢＰ補間データ 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) '補間モード設定。Ｘ，Ｙ，Ｚ軸指定。 '速度関連パラメータ設定(実際の設定記述は省略) Ret = Nmc_3BPExecMC8500P(No, IcNo, Data3Bp, 2, &H7) '３軸ＢＰ補間実行。データ数2, X,Y,Z軸 If Ret = BP_END Then '戻り値正常 Call MsgBox("正常終了") End If ``` **注意** この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。 この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

関数名	機能　及び　内容
Nmc_4BPExecMC8500P	指定した補間データで４軸ビットパターン補間を実行する。 この関数は、補間処理が終了した後に制御を返します。補間が終了するまで制御が戻らないのでアプリケーションでスレッドを作成し、そのスレッドからコールする事を推奨します。

```
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
```

入力パラメータ

No	ボード番号（ボード上のロータリースイッチの値(0～15)）
IcNo	IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照
pData4Bp	４軸ＢＰ補間データの構造体(ユーザー定義型)配列の先頭アドレス(DATA_4BPのアドレス)。 DATA_4BPに実行する補間データをセットし、アドレスを指定する。 DATA_4BPについては補足説明(3)参照。
DataCnt	４軸ＢＰ補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。
IpAxis	補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。 補足説明(4)参照。

戻り値

補間処理が正常終了した場合は BP_END が返ります。
補間処理でエラーが発生した場合は、下記エラーコードが返ります。

■正常終了

BP_END	BP補間処理正常終了

■エラーコード

BP_CNT_ERR	指定されたデータ数が範囲外
BP_ALREADY_EXEC	既にＢＰ補間、あるいは連続補間が実行中
BP_MALLOC_ERR	メモリを確保できなかった
BP_PARAM_ERR	引数の値が正しくない
BP_NOT_OPEN_ERR	指定したボードがオープンされていない
BP_OTHER_ERR	その他のエラー
BP_FUNC_ERR	使用できない関数を使用(MC8500PでMC8000P用関数を使用したときなど)
BP_STOP	BP補間が途中で停止した（速度が速く次データのスタックが間に合わなかった場合）
BP_USER_STOP	BP補間実行中にユーザーが中断した
BP_DRIVE_ERR	BP補間実行中にボードでエラー発生（RR0にエラー情報がセットされた）
BP_STOP_CERR	連続補間が途中で停止した（RR2エラーによる停止の場合）
BP_GETSC_ERR	スタックカウンタ読み出しエラー

補間データに終了コードが書かれていて補間が終了した場合は、BP_ENDまたはBP_STOPのいずれかのエラーコードが返ります。

使用例

[VC]

```
  // ４軸ＢＰ補間データ    BP1P,   BP1M,   BP2P,   BP2M,   BP3P,   BP3M,   BP4P,   BP4M
 DATA_4BP Data4Bp[2] = {{0xFFE4, 0x0000, 0x03FF, 0x4000, 0x0000, 0xFFFF, 0xFE80, 0x03FF},
                        {0x0000, 0x03FF, 0xFFD0, 0x0000, 0x4AAB, 0x0000, 0x1FFF, 0x0000}};

 Nmc_IPMode(No, IcNo,0x000F);                        //補間モード設定。Ｘ,Ｙ,Ｚ,Ｕ軸指定。

 // 速度関連パラメータ設定(実際の設定記述は省略)

 Ret = Nmc_4BPExecMC8500P(No, IcNo, Data4Bp, 2, 0xF);//４軸ＢＰ補間実行。データ数2, X,Y,Z,U軸
 if(Ret == BP_END)   AfxMessageBox("正常終了");      //戻り値正常
```

	[VB.NET] ``` '４軸ＢＰ補間データ 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 = &H3FFS 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) '補間モード設定。Ｘ，Ｙ，Ｚ，Ｕ軸指定。 '速度関連パラメータ設定(実際の設定記述は省略) Ret = Nmc_4BPExecMC8500P(No, IcNo, Data4Bp, 2, &HF) '４軸ＢＰ補間実行。データ数2, X,Y,Z,U軸 If Ret = BP_END Then Call MsgBox("正常終了") End If ``` **注意** この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。 この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

関数名	機能　及び　内容
Nmc_2BPExecMC8500P_BG	指定した補間データで２軸ビットパターン補間をバックグラウンドで実行する。 この関数は、補間処理を開始した直後に制御を返し、バックグラウンドで補間を実行します。 指定したウィンドウに対して、補間終了時にWM_BP_ENDメッセージを送信し、終了ステータスを渡します。 **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 **入力パラメータ** User_hWnd ユーザーアプリケーションのウィンドウハンドル No　　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。 　　　　　詳細は補足説明(7)参照 pData2Bp　２軸ＢＰ補間データの構造体(ユーザー定義型)配列の先頭アドレス(DATA_2BPのアドレス)。 　　　　　DATA_2BPに実行する補間データをセットし、アドレスを指定する。 　　　　　DATA_2BPについては補足説明(3)参照。 DataCnt　２軸ＢＰ補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。 IpAxis　　補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。 　　　　　補足説明(4)参照。 **戻り値** バックグラウンドで補間処理が正常に開始した場合は BP_START が返ります。 補間処理が開始する前にエラーが発生した場合は、下記の補間開始前のエラーコードが返ります。 ■正常開始 BP_START　　　バックグラウンドでBP補間処理が正常に開始した ■エラーコード(補間開始前のエラー) BP_CNT_ERR　　　指定されたデータ数が範囲外 BP_ALREADY_EXEC　既にＢＰ補間、あるいは連続補間が実行中 BP_THREAD_ERR　　スレッドを起動できなかった BP_MALLOC_ERR　　メモリを確保できなかった BP_PARAM_ERR　　引数の値が正しくない BP_NOT_OPEN_ERR　指定したボードがオープンされていない BP_OTHER_ERR　　その他のエラー BP_FUNC_ERR　　使用できない関数を使用 バックグラウンドで補間処理が正常に開始した後は、補間処理終了時に、指定したウィンドウに対して、WM_BP_ENDメッセージを送信します。WM_BP_ENDのメッセージ受信関数で受け取る第１引数にボード番号を第２引数に終了ステータスを渡します。 その終了ステータスは、補間が正常終了した場合は BP_END です。 補間実行時にエラーが発生した場合は、下記の補間開始後のエラーコードです。 ■正常終了 BP_END　　　BP補間処理正常終了 ■エラーコード(補間開始後のエラー) BP_STOP　　　BP補間が途中で停止した(速度が速く次データのスタックが間に合わなかった) BP_USER_STOP　BP補間実行中にユーザーが中断した BP_DRIVE_ERR　BP補間実行中にボードでエラーが発生した（RR0にエラー情報がセットされた） BP_STOP_CERR　連続補間が途中で停止した（RR2エラーによる停止の場合） BP_GETSC_ERR　スタックカウンタ読み出しエラー 補間データに終了コードが書かれていて補間が終了した場合は、BP_ENDまたはBP_STOPのいずれかのエラーコードが返ります。

使用例

[VC]

```
{
    // ２軸ＢＰ補間データ     BP1P,    BP1M,    BP2P,    BP2M
    DATA_2BP Data2Bp[2] = {{0x0000, 0x2BFF, 0xFFD4, 0x0000},
                           {0xF6FE, 0x0000, 0x000F, 0x3FC0}};

    Nmc_IPMode(No, IcNo,0x0003);                        //補間モード設定。Ｘ，Ｙ軸指定。

    // 速度関連パラメータ設定(実際の設定記述は省略)

    Ret = Nmc_2BPExecMC8500P_BG(hWnd, No, IcNo, Data2Bp, 2, 0x3);
                                                        //2軸ＢＰ補間実行。データ数2, X,Y軸
    if(Ret == BP_START)   AfxMessageBox("補間開始");  //戻り値正常(補間開始)
}

BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)              // WM_BP_ENDメッセージ受信関数設定
    ON_MESSAGE( WM_BP_END, OnMsg_BP )
END_MESSAGE_MAP()


// WM_BP_ENDメッセージ受信関数
afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_BP(WPARAM BoardNo, LPARAM Status)
{
    if(Status == BP_END)   AfxMessageBox("補間正常終了"); // 戻り値正常(補間終了)
    return 0;
}
```

[VB.NET]

```
        '２軸ＢＰ補間データ
        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)             '補間モード設定。Ｘ，Ｙ軸指定。

        '速度関連パラメータ設定(実際の設定記述は省略)

        '2軸ＢＰ補間実行。データ数2, X,Y軸
        Ret = Nmc_2BPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data2Bp, 2, &H3)
        If Ret = BP_START Then    '戻り値正常(補間開始)
            Call MsgBox("補間開始")
        End If
    End Sub

    'WM_BP_ENDメッセージ受信関数
    Protected Overrides Sub WndProc(ByRef m As Message)
        Select Case (m.Msg)
            Case WM_BP_END                            'ＢＰ補間終了メッセージ
                If m.LParam.ToInt32 = BP_END Then   '戻り値正常(補間終了)
                    Call MsgBox("補間正常終了")
                End If
                Exit Select
        End Select
        MyBase.WndProc(m)
    End Sub
```

注意

この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。

この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

<table>
<tr><th>関数名</th><th>機能　及び　内容</th></tr>
<tr><td>Nmc_3BPExecMC8500P
_BG</td><td>指定した補間データで３軸ビットパターン補間をバックグラウンドで実行する。
この関数は、補間処理を開始した直後に制御を返し、バックグラウンドで補間を実行します。
指定したウィンドウに対して、補間終了時にWM_BP_ENDメッセージを送信し、終了ステータスを渡します。

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

入力パラメータ
User_hWnd　ユーザーアプリケーションのウィンドウハンドル
No　ボード番号（ボード上のロータリースイッチの値(0～15)）
IcNo　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。
詳細は補足説明(7)参照
pData3Bp　３軸ＢＰ補間データの構造体(ユーザー定義型)配列の先頭アドレス(DATA_3BPのアドレス)。
DATA_3BPに実行する補間データをセットし、アドレスを指定する。
DATA_3BPについては補足説明(3)参照。
DataCnt　３軸ＢＰ補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。
IpAxis　補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。
補足説明(4)参照。

戻り値
バックグラウンドで補間処理が正常に開始した場合は BP_START が返ります。
補間処理が開始する前にエラーが発生した場合は、下記の補間開始前のエラーコードが返ります。

■正常開始
BP_START　バックグラウンドでBP補間処理が正常に開始した

■エラーコード(補間開始前のエラー)
BP_CNT_ERR　指定されたデータ数が範囲外
BP_ALREADY_EXEC　既にＢＰ補間、あるいは連続補間が実行中
BP_THREAD_ERR　スレッドを起動できなかった
BP_MALLOC_ERR　メモリを確保できなかった
BP_PARAM_ERR　引数の値が正しくない
BP_NOT_OPEN_ERR　指定したボードがオープンされていない
BP_OTHER_ERR　その他のエラー
BP_FUNC_ERR　使用できない関数を使用

バックグラウンドで補間処理が正常に開始した後は、補間処理終了時に、指定したウィンドウに対して、WM_BP_ENDメッセージを送信します。WM_BP_ENDのメッセージ受信関数で受け取る第１引数にボード番号を第２引数に終了ステータスを渡します。
その終了ステータスは、補間が正常終了した場合は BP_END です。
補間実行時にエラーが発生した場合は、下記の補間開始後のエラーコードです。

■正常終了
BP_END　BP補間処理正常終了

■エラーコード(補間開始後のエラー)
BP_STOP　BP補間が途中で停止した(速度が速く次データのスタックが間に合わなかった)
BP_USER_STOP　BP補間実行中にユーザーが中断した
BP_DRIVE_ERR　BP補間実行中にボードでエラーが発生した（RR0にエラー情報がセットされた）
BP_STOP_CERR　連続補間が途中で停止した（RR2エラーによる停止の場合）
BP_GETSC_ERR　スタックカウンタ読み出しエラー

補間データに終了コードが書かれていて補間が終了した場合は、BP_ENDまたはBP_STOPのいずれかのエラーコードが返ります。</td></tr>
</table>

使用例

[VC]

```
{
  // ３軸ＢＰ補間データ    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);                          //補間モード設定。Ｘ，Ｙ，Ｚ軸指定。

  // 速度関連パラメータ設定(実際の設定記述は省略)

  Ret = Nmc_3BPExecMC8500P_BG(hWnd, No, IcNo, Data3Bp, 2, 0x7);
                                                        //3軸ＢＰ補間実行。データ数2, X,Y,Z軸
   if(Ret == BP_START)   AfxMessageBox("補間開始");  //戻り値正常(補間開始)
}

BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)           // WM_BP_ENDメッセージ受信関数設定
    ON_MESSAGE( WM_BP_END, OnMsg_BP )
END_MESSAGE_MAP()

// WM_BP_ENDメッセージ受信関数
afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_BP(WPARAM BoardNo, LPARAM Status)
{
    if(Status == BP_END)   AfxMessageBox("補間正常終了"); // 戻り値正常(補間終了)
    return 0;
}
```

[VB.NET]

```
        '３軸ＢＰ補間データ
        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)              '補間モード設定。Ｘ，Ｙ，Ｚ軸指定。

        '速度関連パラメータ設定(実際の設定記述は省略)

        '3軸ＢＰ補間実行。データ数2, X,Y,Z軸
        Ret = Nmc_3BPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data3Bp, 2, &H7)
        If Ret = BP_START Then    '戻り値正常(補間開始)
            Call MsgBox("補間開始")
        End If
    End Sub

    'WM_BP_ENDメッセージ受信関数
    Protected Overrides Sub WndProc(ByRef m As Message)
        Select Case (m.Msg)
            Case WM_BP_END                          'ＢＰ補間終了メッセージ
                If m.LParam.ToInt32 = BP_END Then    '戻り値正常(補間終了)
                    Call MsgBox("補間正常終了")
                End If
                Exit Select
        End Select
        MyBase.WndProc(m)
    End Sub
```

注意

この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。
この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

関数名	機能　及び　内容
Nmc_4BPExecMC8500P_BG	指定した補間データで４軸ビットパターン補間をバックグラウンドで実行する。 この関数は、補間処理を開始した直後に制御を返し、バックグラウンドで補間を実行します。 指定したウィンドウに対して、補間終了時にWM_BP_ENDメッセージを送信し、終了ステータスを渡します。 **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 **入力パラメータ** User_hWnd　ユーザーアプリケーションのウィンドウハンドル No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。 詳細は補足説明(7)参照 pData4Bp　４軸ＢＰ補間データの構造体(ユーザー定義型)配列の先頭アドレス(DATA_4BPのアドレス)。 DATA_4BPに実行する補間データをセットし、アドレスを指定する。 DATA_4BPについては補足説明(3)参照。 DataCnt　４軸ＢＰ補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。 IpAxis　補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。 補足説明(4)参照。 **戻り値** バックグラウンドで補間処理が正常に開始した場合は BP_START が返ります。 補間処理が開始する前にエラーが発生した場合は、下記の補間開始前のエラーコードが返ります。 ■正常開始 BP_START　バックグラウンドでBP補間処理が正常に開始した ■エラーコード(補間開始前のエラー) BP_CNT_ERR　指定されたデータ数が範囲外 BP_ALREADY_EXEC　既にＢＰ補間、あるいは連続補間が実行中 BP_THREAD_ERR　スレッドを起動できなかった BP_MALLOC_ERR　メモリを確保できなかった BP_PARAM_ERR　引数の値が正しくない BP_NOT_OPEN_ERR　指定したボードがオープンされていない BP_OTHER_ERR　その他のエラー BP_FUNC_ERR　使用できない関数を使用 バックグラウンドで補間処理が正常に開始した後は、補間処理終了時に、指定したウィンドウに対して、WM_BP_ENDメッセージを送信します。WM_BP_ENDのメッセージ受信関数で受け取る第１引数にボード番号を第２引数に終了ステータスを渡します。 その終了ステータスは、補間が正常終了した場合は BP_END です。 補間実行時にエラーが発生した場合は、下記の補間開始後のエラーコードです。 ■正常終了 BP_END　BP補間処理正常終了 ■エラーコード(補間開始後のエラー) BP_STOP　BP補間が途中で停止した(速度が速く次データのスタックが間に合わなかった) BP_USER_STOP　BP補間実行中にユーザーが中断した BP_DRIVE_ERR　BP補間実行中にボードでエラーが発生した（RR0にエラー情報がセットされた） BP_STOP_CERR　連続補間が途中で停止した（RR2エラーによる停止の場合） BP_GETSC_ERR　スタックカウンタ読み出しエラー 補間データに終了コードが書かれていて補間が終了した場合は、BP_ENDまたはBP_STOPのいずれかのエラーコードが返ります。

使用例

[VC]

```
{
 // ４軸ＢＰ補間データ      BP1P,   BP1M,   BP2P,   BP2M,   BP3P,  BP3M,   BP4P,   BP4M
  DATA_4BP Data4Bp[2] = {{0xFFE4, 0x0000, 0x03FF, 0x4000, 0x0000, 0xFFFF, 0xFE80, 0x03FF},
                         {0x0000, 0x03FF, 0xFFD0, 0x0000, 0x4AAB, 0x0000, 0x1FFF, 0x0000}};

  Nmc_IPMode(No, IcNo,0x000F);                        //補間モード設定。Ｘ，Ｙ，Ｚ，Ｕ軸指定。

  // 速度関連パラメータ設定(実際の設定記述は省略)

  Ret = Nmc_4BPExecMC8500P_BG(hWnd, No, IcNo, Data4Bp, 2, 0xF);
                                                      //4軸ＢＰ補間実行。データ数2, X,Y,Z,U軸
   if(Ret == BP_START)   AfxMessageBox("補間開始"); //戻り値正常(補間開始)
}

BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)    // WM_BP_ENDメッセージ受信関数設定
    ON_MESSAGE( WM_BP_END, OnMsg_BP )
END_MESSAGE_MAP()

// WM_BP_ENDメッセージ受信関数
afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_BP(WPARAM BoardNo, LPARAM Status)
{
    if(Status == BP_END)   AfxMessageBox("補間正常終了"); // 戻り値正常(補間終了)
    return 0;
}
```

[VB.NET]

```
        '４軸ＢＰ補間データ
        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 = &H3FFS
        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)            '補間モード設定。Ｘ，Ｙ，Ｚ，Ｕ軸指定。

        '速度関連パラメータ設定(実際の設定記述は省略)

        '4軸ＢＰ補間実行。データ数2, X,Y,Z,U軸
        Ret = Nmc_4BPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data4Bp, 2, &HF)
        If Ret = BP_START Then    '戻り値正常(補間開始)
            Call MsgBox("補間開始")
        End If
    End Sub

    'WM_BP_ENDメッセージ受信関数
    Protected Overrides Sub WndProc(ByRef m As Message)
        Select Case (m.Msg)
            Case WM_BP_END                          'ＢＰ補間終了メッセージ
                If m.LParam.ToInt32 = BP_END Then    '戻り値正常(補間終了)
                    Call MsgBox("補間正常終了")
                End If
                Exit Select
        End Select
        MyBase.WndProc(m)
    End Sub
```

注意

この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。

この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

関数名	機能　及び　内容

Nmc_2CIPExecMC8500P	指定した補間データで２軸連続補間を実行する。
	この関数は、補間処理が終了した後に制御を返します。補間が終了するまで制御が戻らないのでアプリケーションでスレッドを作成し、そのスレッドからコールする事を推奨します。
	
	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
	
	入力パラメータ
	No　ボード番号（ボード上のロータリースイッチの値(0～15)）
	IcNo　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。
	詳細は補足説明(7)参照
	pData2Cip　２軸連続補間データの構造体(ユーザー定義型)配列の先頭アドレス
	(DATA_2CIP_MC8500Pのアドレス)。
	DATA_2CIP_MC8500Pに実行する補間データをセットし、アドレスを指定する。
	DATA_2CIP_MC8500Pについては補足説明(3)参照。
	DataCnt　２軸連続補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。
	IpAxis　補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。
	補足説明(4)参照。
	SpdChgFlg 補間実行中に速度を変更するかどうかを設定する。変更する場合は補足説明(5)参照。
	[VC]　TRUE ：変更する、FALSE ：変更しない。省略可能。省略時FALSE。
	[VB.NET]　True ：変更する、False ：変更しない
	変更する選択時　：DATA_2CIP_MC8500PのSpeedに設定した値を参照します。
	Speedに1～4000000の値設定・・・設定した速度に変更します。
	Speedに0 設定・・・・・・・・・・速度は変更しません。
	変更しない選択時：DATA_2CIP_MC8500PのSpeedに設定した値を参照しません。
	
	戻り値
	補間処理が正常終了した場合は CIP_END が返ります。
	補間処理でエラーが発生した場合は、下記エラーコードが返ります。
	
	■正常終了
	CIP_END　連続補間処理正常終了
	
	■エラーコード
	CIP_CNT_ERR　指定されたデータ数が範囲外
	CIP_ALREADY_EXEC　既にＢＰ補間、あるいは連続補間が実行中
	CIP_MALLOC_ERR　メモリを確保できなかった
	CIP_CMD_ERR　コマンドエラー（ユーザーが指定したコマンドが間違っている）
	CIP_PARAM_ERR　引数の値が正しくない
	CIP_NOT_OPEN_ERR　指定したボードがオープンされていない
	CIP_OTHER_ERR　その他のエラー
	CIP_FUNC_ERR　使用できない関数を使用
	CIP_STOP　連続補間が途中で停止した
	CIP_USER_STOP　連続補間実行中にユーザーが中断した
	CIP_DRIVE_ERR　連続補間実行中にボードでエラー発生（RR0にエラー情報がセットされた）
	CIP_STOP_CERR　連続補間が途中で停止した（RR2エラーによる停止の場合）
	CIP_GETSC_ERR　スタックカウンタ読み出しエラー
	
	使用例
	[VC]
	// ２軸連続補間データ　コマンド　速度　終点１ 終点２ 中心点１ 中心点２
	DATA_2CIP_MC8500P Data2Cip[2]={{MC8500P_CMD_2CIP, 0, 4500, 0, 0, 0}, //２軸直線補間
	{MC8500P_CMD_CIPCCW, 0, 1500, 1500, 0, 1500}}; //CCW 円弧補間
	
	Nmc_IPMode(No, IcNo,0x0003);　//補間モード設定。Ｘ,Ｙ軸指定。
	// 速度関連パラメータ設定
	Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);　// 定速ドライブにするため4M
	// 他の設定記述は省略
	Ret = Nmc_2CIPExecMC8500P(No, IcNo, Data2Cip, 2, 0x03); //２軸連続補間実行。データ数2, X,Y軸
	if(Ret == CIP_END)　AfxMessageBox("正常終了");　//戻り値正常

	[VB.NET] ``` '２軸補間データ Dim Data2Cip(1) As DATA_2CIP_MC8500P Data2Cip(0).Command = MC8500P_CMD_2CIP '２軸直線補間 Data2Cip(0).Speed = 0 '速度 Data2Cip(0).EndP1 = 4500 '終点（第１軸） Data2Cip(0).EndP2 = 0 '終点（第２軸） Data2Cip(0).Center1 = 0 '中心点１ Data2Cip(0).Center2 = 0 '中心点２ Data2Cip(1).Command = MC8500P_CMD_CIPCCW 'CCW 円弧補間 Data2Cip(1).Speed = 0 '速度 Data2Cip(1).EndP1 = 1500 '終点（第１軸） Data2Cip(1).EndP2 = 1500 '終点（第２軸） Data2Cip(1).Center1 = 0 '中心点１ Data2Cip(1).Center2 = 1500 '中心点２ Call Nmc_IPMode(No, IcNo, &H3) '補間モード設定。Ｘ，Ｙ軸指定。 '速度関連パラメータ設定 Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000) '定速ドライブにするため4M '他の設定記述は省略 '２軸連続補間実行。データ数2, X,Y軸 Ret = Nmc_2CIPExecMC8500P(No, IcNo, Data2Cip, 2, &H3, False) If Ret = CIP_END Then '戻り値正常 Call MsgBox("正常終了") End If ``` **注意** この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。 この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

<table>
<tr><th>関数名</th><th>機能　及び　内容</th></tr>
<tr><td>Nmc_3CIPExecMC8500P</td><td>指定した補間データで３軸連続補間を実行する。この関数は、補間処理が終了した後に制御を返します。補間が終了するまで制御が戻らないのでアプリケーションでスレッドを作成し、そのスレッドからコールする事を推奨します。

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

入力パラメータ
No　ボード番号（ボード上のロータリースイッチの値(0～15)）
IcNo　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。
　詳細は補足説明(7)参照
pData3Cip　３軸連続補間データの構造体(ユーザー定義型)配列の先頭アドレス
　(DATA_3CIP_MC8500Pのアドレス)。
　DATA_3CIP_MC8500Pに実行する補間データをセットし、アドレスを指定する。
　DATA_3CIP_MC8500Pについては補足説明(3)参照。
DataCnt　３軸連続補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。
IpAxis　補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。
　補足説明(4)参照。
SpdChgFlg　補間実行中に速度を変更するかどうかを設定する。変更する場合は補足説明(5)参照。
　[VC]　TRUE ：変更する、FALSE ：変更しない。省略可能。省略時FALSE。
　[VB.NET]　True ：変更する、False ：変更しない
　変更する選択時　：DATA_3CIP_MC8500PのSpeedに設定した値を参照します。
　　Speedに1～4000000の値設定・・・設定した速度に変更します。
　　Speedに0 設定・・・・・・・・・・速度は変更しません。
　変更しない選択時：DATA_3CIP_MC8500PのSpeedに設定した値を参照しません。

戻り値
補間処理が正常終了した場合は CIP_END が返ります。
補間処理でエラーが発生した場合は、下記エラーコードが返ります。

■正常終了
CIP_END　連続補間処理正常終了

■エラーコード
CIP_CNT_ERR　指定されたデータ数が範囲外
CIP_ALREADY_EXEC　既にＢＰ補間、あるいは連続補間が実行中
CIP_MALLOC_ERR　メモリを確保できなかった
CIP_CMD_ERR　コマンドエラー（ユーザーが指定したコマンドが間違っている）
CIP_PARAM_ERR　引数の値が正しくない
CIP_NOT_OPEN_ERR　指定したボードがオープンされていない
CIP_OTHER_ERR　その他のエラー
CIP_FUNC_ERR　使用できない関数を使用
CIP_STOP　連続補間が途中で停止した
CIP_USER_STOP　連続補間実行中にユーザーが中断した
CIP_DRIVE_ERR　連続補間実行中にボードでエラー発生（RR0にエラー情報がセットされた）
CIP_STOP_CERR　連続補間が途中で停止した（RR2エラーによる停止の場合）
CIP_GETSC_ERR　スタックカウンタ読み出しエラー</td></tr>
</table>

使用例

```
[VC] DATA_3CIP_MC8500P Data3Cip[2];                  // ３軸連続補間データ用

    // ３軸連続補間データ設定
    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);                    //補間モード設定。Ｘ，Ｙ，Ｚ軸指定。
    // 速度関連パラメータ設定
    Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);        // 定速ドライブにするため4M
                                                    // 他の設定記述は省略

    Ret = Nmc_3CIPExecMC8500P(No, IcNo, Data3Cip, 2, 0x007);
                                                    //３軸連続補間実行。データ数2, X,Y,Z軸
    if(Ret == CIP_END)   AfxMessageBox("正常終了"); //戻り値正常

[VB.NET]
    '３軸連続補間データ設定
    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)                      '補間モード設定。Ｘ，Ｙ，Ｚ軸指定。
    '速度関連パラメータ設定
    Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)        '定速ドライブにするため4M
                                                        '他の設定記述は省略
    '３軸連続補間実行。データ数2, X,Y,Z軸
    Ret = Nmc_3CIPExecMC8500P(No, IcNo, Data3Cip, 2, &H7, False)
    If Ret = CIP_END Then    '戻り値正常
        Call MsgBox("正常終了")
    End If
```

注意

この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。
この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

関数名	機能　及び　内容
Nmc_4CIPExecMC8500P	指定した補間データで４軸連続補間を実行する。この関数は、補間処理が終了した後に制御を返します。補間が終了するまで制御が戻らないのでアプリケーションでスレッドを作成し、そのスレッドからコールする事を推奨します。 **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 **入力パラメータ** No　　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。 　　　　　詳細は補足説明(7)参照 pData4Cip　４軸連続補間データの構造体(ユーザー定義型)配列の先頭アドレス。 　　　　　(DATA_4CIP_MC8500Pのアドレス)。 　　　　　DATA_4CIP_MC8500Pに実行する補間データをセットし、アドレスを指定する。 　　　　　DATA_4CIP_MC8500Pについては補足説明(3)参照。 DataCnt　　４軸連続補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。 IpAxis　　補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。 　　　　　補足説明(4)参照。 SpdChgFlg　補間実行中に速度を変更するかどうかを設定する。変更する場合は補足説明(5)参照。 　　　　　[VC]　　　TRUE　：変更する、FALSE　：変更しない。省略可能。省略時FALSE。 　　　　　[VB.NET]　True　：変更する、False　：変更しない 　　　　　**変更する選択時**　：DATA_4CIP_MC8500PのSpeedに設定した値を参照します。 　　　　　　　　　　　Speedに1～4000000の値設定・・・設定した速度に変更します。 　　　　　　　　　　　Speedに0 設定・・・・・・・・・速度は変更しません。 　　　　　**変更しない選択時**：DATA_4CIP_MC8500PのSpeedに設定した値を参照しません。 **戻り値** 補間処理が正常終了した場合は CIP_END が返ります。 補間処理でエラーが発生した場合は、下記エラーコードが返ります。 ■正常終了 CIP_END　　　　　　連続補間処理正常終了 ■エラーコード CIP_CNT_ERR　　　　指定されたデータ数が範囲外 CIP_ALREADY_EXEC　既にＢＰ補間、あるいは連続補間が実行中 CIP_MALLOC_ERR　　メモリを確保できなかった CIP_CMD_ERR　　　　コマンドエラー（ユーザーが指定したコマンドが間違っている） CIP_PARAM_ERR　　　引数の値が正しくない CIP_NOT_OPEN_ERR　指定したボードがオープンされていない CIP_OTHER_ERR　　　その他のエラー CIP_FUNC_ERR　　　使用できない関数を使用 CIP_STOP　　　　　連続補間が途中で停止した CIP_USER_STOP　　　連続補間実行中にユーザーが中断した CIP_DRIVE_ERR　　　連続補間実行中にボードでエラー発生（RR0にエラー情報がセットされた） CIP_STOP_CERR　　　連続補間が途中で停止した（RR2エラーによる停止の場合） CIP_GETSC_ERR　　　スタックカウンタ読み出しエラー

	使用例

```
[VC] DATA_4CIP_MC8500P Data4Cip[2];               // ４軸連続補間データ用

    // ４軸連続補間データ設定
    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);                  //補間モード設定。Ｘ,Ｙ,Ｚ,Ｕ軸指定。
    // 速度関連パラメータ設定
    Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);      // 定速ドライブにするため4M
                                                  // 他の設定記述は省略
    Ret = Nmc_4CIPExecMC8500P(No, IcNo, Data4Cip, 2, 0x00F);
                                                  //４軸連続補間実行。データ数2, X,Y,Z,U軸
    if(Ret == CIP_END)   AfxMessageBox("正常終了");  //戻り値正常

[VB.NET]
    '４軸連続補間データ設定
    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)                 '補間モード設定。Ｘ,Ｙ,Ｚ,Ｕ軸指定。
    '速度関連パラメータ設定
    Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)   '定速ドライブにするため4M
                                                   '他の設定記述は省略
    '４軸連続補間実行。データ数2, X,Y,Z,U軸
    Ret = Nmc_4CIPExecMC8500P(No, IcNo, Data4Cip, 2, &HF, False)
    If Ret = CIP_END Then    '戻り値正常
        Call MsgBox("正常終了")
    End If
```

注意

この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。
この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

関数名	機能　及び　内容
Nmc_2CIPExecMC8500P _BG	指定した補間データで２軸連続補間をバックグラウンドで実行する。 この関数は、補間処理を開始した直後に制御を返し、バックグラウンドで補間を実行します。 指定したウィンドウに対して、補間終了時にWM_CIP_ENDメッセージを送信し、終了ステータスを渡します。 **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 **入力パラメータ** User_hWnd ユーザーアプリケーションのウィンドウハンドル No　　　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。 　　　　詳細は補足説明(7)参照 pData2Cip ２軸連続補間データの構造体(ユーザー定義型)配列の先頭アドレス 　　　　(DATA_2CIP_MC8500Pのアドレス)。 　　　　DATA_2CIP_MC8500Pに実行する補間データをセットし、アドレスを指定する。 　　　　DATA_2CIP_MC8500Pについては補足説明(3)参照。 DataCnt　２軸連続補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。 IpAxis　補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。 　　　　補足説明(4)参照。 SpdChgFlg 補間実行中に速度を変更するかどうかを設定する。変更する場合は補足説明(5)参照。 　　　　[VC]　　TRUE ：変更する、FALSE ：変更しない。省略可能。省略時FALSE。 　　　　[VB.NET]　True ：変更する、False ：変更しない 　　　　**変更する選択時**　：DATA_2CIP_MC8500PのSpeedに設定した値を参照します。 　　　　　　Speedに1～4000000の値設定・・・設定した速度に変更します。 　　　　　　Speedに0 設定・・・・・・・・・速度は変更しません。 　　　　**変更しない選択時**：DATA_2CIP_MC8500PのSpeedに設定した値を参照しません。 **戻り値** バックグラウンドで補間処理が正常に開始した場合は CIP_START が返ります。 補間処理が開始する前にエラーが発生した場合は、下記の補間開始前のエラーコードが返ります。 ■正常開始 CIP_START　バックグラウンドで連続補間処理が正常に開始した ■エラーコード(補間開始前のエラー) CIP_CNT_ERR　指定されたデータ数が範囲外 CIP_ALREADY_EXEC　既にＢＰ補間、あるいは連続補間が実行中 CIP_THREAD_ERR　スレッドを起動できなかった CIP_MALLOC_ERR　メモリを確保できなかった CIP_CMD_ERR　コマンドエラー（ユーザーが指定したコマンドが間違っている） CIP_PARAM_ERR　引数の値が正しくない CIP_NOT_OPEN_ERR　指定したボードがオープンされていない CIP_OTHER_ERR　その他のエラー CIP_FUNC_ERR　使用できない関数を使用 バックグラウンドで補間処理が正常に開始した後は、補間処理終了時に、指定したウィンドウに対して、WM_CIP_ENDメッセージを送信します。WM_CIP_ENDのメッセージ受信関数で受け取る第１引数にボード番号を、第２引数に終了ステータスを渡します。その終了ステータスは、補間が正常終了した場合はCIP_ENDです。補間実行時にエラーが発生した場合は、下記の補間開始後のエラーコードです。 ■正常終了 CIP_END　連続補間処理正常終了 ■エラーコード(補間開始後のエラー) CIP_USER_STOP　連続補間実行中にユーザーが中断した CIP_DRIVE_ERR　連続補間実行中にボードでエラー発生（RR0にエラー情報がセットされた） CIP_STOP_CERR　連続補間が途中で停止した（RR2エラーによる停止の場合） CIP_GETSC_ERR　スタックカウンタ読み出しエラー

使用例

[VC]
```
  {
   // ２軸連続補間データ           コマンド          速度  終点１ 終点２ 中心点１ 中心点２
   DATA_2CIP_MC8500P Data2Cip[2]={{MC8500P_CMD_2CIP,  0,   4500,    0,    0,    0},  //２軸直線補間
                                 {MC8500P_CMD_CIPCCW, 0,   1500,  1500,    0,  1500}}; //CCW 円弧補間

     Nmc_IPMode(No, IcNo,0x0003);                          //補間モード設定。Ｘ，Ｙ軸指定。
      // 速度関連パラメータ設定
      Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);            // 定速ドライブにするため4M
                                                          // 他の設定記述は省略
      Ret = Nmc_2CIPExecMC8500P_BG(hWnd, No, IcNo, Data2Cip, 2, 0x03);
                                                         //2軸連続補間実行。データ数2, X,Y軸
      if(Ret == CIP_START)   AfxMessageBox("補間開始");  //戻り値正常(補間開始)
  }

  BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)    // WM_CIP_ENDメッセージ受信関数設定
     ON_MESSAGE( WM_CIP_END, OnMsg_CIP )
  END_MESSAGE_MAP()

  // WM_CIP_ENDメッセージ受信関数
  afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_CIP(WPARAM BoardNo, LPARAM Status)
  {
     if(Status == CIP_END)   AfxMessageBox("補間正常終了"); // 戻り値正常(補間終了)
     return 0;
  }
```

[VB.NET]
```
        '２軸連続補間データ
        Dim Data2Cip(1) As DATA_2CIP_MC8500P
        Data2Cip(0).Command = MC8500P_CMD_2CIP   '２軸直線補間
        Data2Cip(0).Speed = 0                    '速度変更する
        Data2Cip(0).EndP1 = 4500                 '終点１
        Data2Cip(0).EndP2 = 0                    '終点２
        Data2Cip(0).Center1 = 0                  '中心点１
        Data2Cip(0).Center2 = 0                  '中心点２
        Data2Cip(1).Command = MC8500P_CMD_CIPCCW 'CCW円弧補間
        Data2Cip(1).Speed = 0                    '速度変更する
        Data2Cip(1).EndP1 = 1500                 '終点１
        Data2Cip(1).EndP2 = 1500                 '終点２
        Data2Cip(1).Center1 = 0                  '中心点１
        Data2Cip(1).Center2 = 1500               '中心点２

        Call Nmc_IPMode(No, IcNo, &H3)           '補間モード設定。Ｘ，Ｙ軸指定。
        '速度関連パラメータ設定
        Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)  '定速ドライブにするため4M
                                                      '他の設定記述は省略
        '2軸連続補間実行。データ数2, X,Y軸
        Ret = Nmc_2CIPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data2Cip, 2, &H3, False)
        If Ret = CIP_START Then    '戻り値正常(補間開始)
            Call MsgBox("補間開始")
        End If
    End Sub

    'WM_CIP_ENDメッセージ受信関数
    Protected Overrides Sub WndProc(ByRef m As Message)
        Select Case (m.Msg)
            Case WM_CIP_END                           '連続補間終了メッセージ
                If m.LParam.ToInt32 = CIP_END Then   '戻り値正常(補間終了)
                    Call MsgBox("補間正常終了")
                End If
                Exit Select
        End Select
        MyBase.WndProc(m)
    End Sub
```

注意

この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。
この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

関数名	機能　及び　内容
Nmc_3CIPExecMC8500P_BG	指定した補間データで３軸連続補間をバックグラウンドで実行する。

この関数は、補間処理を開始した直後に制御を返し、バックグラウンドで補間を実行します。
指定したウィンドウに対して、補間終了時にWM_CIP_ENDメッセージを送信し、終了ステータスを渡します。

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
```

入力パラメータ

User_hWnd ユーザーアプリケーションのウィンドウハンドル
No ボード番号（ボード上のロータリースイッチの値(0～15)）
IcNo IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。
詳細は補足説明(7)参照
pData3Cip ３軸連続補間データの構造体(ユーザー定義型)配列の先頭アドレス
(DATA_3CIP_MC8500Pのアドレス)。
DATA_3CIP_MC8500Pに実行する補間データをセットし、アドレスを指定する。
DATA_3CIP_MC8500Pについては補足説明(3)参照。
DataCnt ３軸連続補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。
IpAxis 補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。
補足説明(4)参照。
SpdChgFlg 補間実行中に速度を変更するかどうかを設定する。変更する場合は補足説明(5)参照。
[VC] TRUE ：変更する、FALSE ：変更しない。省略可能。省略時FALSE。
[VB.NET] True ：変更する、False ：変更しない
変更する選択時 ：DATA_3CIP_MC8500PのSpeedに設定した値を参照します。
Speedに1～4000000の値設定・・・設定した速度に変更します。
Speedに0 設定・・・・・・・・・速度は変更しません。
変更しない選択時：DATA_3CIP_MC8500PのSpeedに設定した値を参照しません。

戻り値

バックグラウンドで補間処理が正常に開始した場合は CIP_START が返ります。
補間処理が開始する前にエラーが発生した場合は、下記の補間開始前のエラーコードが返ります。

■正常開始

CIP_START	バックグラウンドで連続補間処理が正常に開始した

■エラーコード(補間開始前のエラー)

CIP_CNT_ERR	指定されたデータ数が範囲外
CIP_ALREADY_EXEC	既にＢＰ補間、あるいは連続補間が実行中
CIP_THREAD_ERR	スレッドを起動できなかった
CIP_MALLOC_ERR	メモリを確保できなかった
CIP_CMD_ERR	コマンドエラー（ユーザーが指定したコマンドが間違っている）
CIP_PARAM_ERR	引数の値が正しくない
CIP_NOT_OPEN_ERR	指定したボードがオープンされていない
CIP_OTHER_ERR	その他のエラー
CIP_FUNC_ERR	使用できない関数を使用

バックグラウンドで補間処理が正常に開始した後は、補間処理終了時に、指定したウィンドウに対して、WM_CIP_ENDメッセージを送信します。WM_CIP_ENDのメッセージ受信関数で受け取る第１引数にボード番号を、第２引数に終了ステータスを渡します。その終了ステータスは、補間が正常終了した場合はCIP_ENDです。補間実行時にエラーが発生した場合は、下記の補間開始後のエラーコードです。

■正常終了

CIP_END	連続補間処理正常終了

■エラーコード(補間開始後のエラー)

CIP_USER_STOP	連続補間実行中にユーザーが中断した
CIP_DRIVE_ERR	連続補間実行中にボードでエラー発生（RR0にエラー情報がセットされた）
CIP_STOP_CERR	連続補間が途中で停止した（RR2エラーによる停止の場合）
CIP_GETSC_ERR	スタックカウンタ読み出しエラー

使用例

[VC]

```
{
   DATA_3CIP_MC8500P Data3Cip[2];                    // ３軸連続補間データ用

   // ３軸連続補間データ設定
   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);                      //補間モード設定。Ｘ，Ｙ，Ｚ軸指定。
   // 速度関連パラメータ設定
   Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);          // 定速ドライブにするため4M
                                                     // 他の設定記述は省略
   Ret = Nmc_3CIPExecMC8500P_BG(hWnd,No, IcNo, Data3Cip, 2, 0x7);
                                                     //3軸連続補間実行。データ数2, X,Y,Z軸
   if(Ret == CIP_START)   AfxMessageBox("補間開始");//戻り値正常(補間開始)
}

BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)  // WM_CIP_ENDメッセージ受信関数設定
   ON_MESSAGE( WM_CIP_END, OnMsg_CIP )
END_MESSAGE_MAP()

// WM_CIP_ENDメッセージ受信関数
afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_CIP(WPARAM BoardNo, LPARAM Status)
{
   if(Status == CIP_END)   AfxMessageBox("補間正常終了"); // 戻り値正常(補間終了)
   return 0;
}
```

[VB.NET]

```
        '３軸連続補間データ設定
        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)               '補間モード設定。Ｘ，Ｙ，Ｚ軸指定。
        '速度関連パラメータ設定
        Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)  '定速ドライブにするため4M
                                                     '他の設定記述は省略
        '3軸連続補間実行。データ数2, X,Y,Z軸
        Ret = Nmc_3CIPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data3Cip, 2, &H7, False)
        If Ret = CIP_START Then                      '戻り値正常(補間開始)
            Call MsgBox("補間開始")
        End If
    End Sub

    'WM_CIP_ENDメッセージ受信関数
    Protected Overrides Sub WndProc(ByRef m As Message)
        Select Case (m.Msg)
            Case WM_CIP_END                          '連続補間終了メッセージ
                If m.LParam.ToInt32 = CIP_END Then   '戻り値正常(補間終了)
                    Call MsgBox("補間正常終了")
                End If
                Exit Select
        End Select
        MyBase.WndProc(m)
    End Sub
```

注意

この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。

この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

関数名	機能　及び　内容
Nmc_4CIPExecMC8500P _BG	指定した補間データで４軸連続補間をバックグラウンドで実行する。 この関数は、補間処理を開始した直後に制御を返し、バックグラウンドで補間を実行します。 指定したウィンドウに対して、補間終了時にWM_CIP_ENDメッセージを送信し、終了ステータスを渡します。 **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 **入力パラメータ** User_hWnd ユーザーアプリケーションのウィンドウハンドル No　ボード番号（ボード上のロータリースイッチの値(0～15)） IcNo　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。 　詳細は補足説明(7)参照 pData4Cip ４軸連続補間データの構造体(ユーザー定義型)配列の先頭アドレス 　(DATA_4CIP_MC8500Pのアドレス)。 　DATA_4CIP_MC8500Pに実行する補間データをセットし、アドレスを指定する。 　DATA_4CIP_MC8500Pについては補足説明(3)参照。 DataCnt　４軸連続補間データの数。構造体(ユーザー定義型)配列の配列数を指定する。 IpAxis　補間を実行する軸。補間モード設定(2A)HのD0～D3(軸指定)の設定値と同じ値を指定する。 　補足説明(4)参照。 SpdChgFlg 補間実行中に速度を変更するかどうかを設定する。変更する場合は補足説明(5)参照。 　[VC]　TRUE ：変更する、FALSE ：変更しない。省略可能。省略時FALSE。 　[VB.NET]　True ：変更する、False ：変更しない 　**変更する選択時**　：DATA_4CIP_MC8500PのSpeedに設定した値を参照します。 　　Speedに1～4000000の値設定・・・設定した速度に変更します。 　　Speedに0 設定・・・・・・・・・速度は変更しません。 　**変更しない選択時**：DATA_4CIP_MC8500PのSpeedに設定した値を参照しません。 **戻り値** バックグラウンドで補間処理が正常に開始した場合は CIP_START が返ります。 補間処理が開始する前にエラーが発生した場合は、下記の補間開始前のエラーコードが返ります。 ■正常開始 CIP_START　バックグラウンドで連続補間処理が正常に開始した ■エラーコード(補間開始前のエラー) CIP_CNT_ERR　指定されたデータ数が範囲外 CIP_ALREADY_EXEC　既にＢＰ補間、あるいは連続補間が実行中 CIP_THREAD_ERR　スレッドを起動できなかった CIP_MALLOC_ERR　メモリを確保できなかった CIP_CMD_ERR　コマンドエラー（ユーザーが指定したコマンドが間違っている） CIP_PARAM_ERR　引数の値が正しくない CIP_NOT_OPEN_ERR　指定したボードがオープンされていない CIP_OTHER_ERR　その他のエラー CIP_FUNC_ERR　使用できない関数を使用 バックグラウンドで補間処理が正常に開始した後は、補間処理終了時に、指定したウィンドウに対して、WM_CIP_ENDメッセージを送信します。WM_CIP_ENDのメッセージ受信関数で受け取る第１引数にボード番号を、第２引数に終了ステータスを渡します。その終了ステータスは、補間が正常終了した場合はCIP_ENDです。補間実行時にエラーが発生した場合は、下記の補間開始後のエラーコードです。 ■正常終了 CIP_END　連続補間処理正常終了 ■エラーコード(補間開始後のエラー) CIP_USER_STOP　連続補間実行中にユーザーが中断した CIP_DRIVE_ERR　連続補間実行中にボードでエラー発生（RR0にエラー情報がセットされた） CIP_STOP_CERR　連続補間が途中で停止した（RR2エラーによる停止の場合） CIP_GETSC_ERR　スタックカウンタ読み出しエラー

使用例

[VC]

```
	{
		DATA_4CIP_MC8500P Data4Cip[2];                    // ４軸連続補間データ用
		// ４軸連続補間データ設定
		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);                       //補間モード設定。Ｘ，Ｙ，Ｚ，Ｕ軸指定。
		// 速度関連パラメータ設定
		Nmc_StartSpd(No, IcNo, AXIS_X, 4000000);          // 定速ドライブにするため4M
		                                                  // 他の設定記述は省略
		Ret = Nmc_4CIPExecMC8500P_BG(hWnd,No, IcNo, Data4Cip, 2, 0xF);
		                                                  //4軸連続補間実行。データ数2, X,Y,Z,U軸
		if(Ret == CIP_START)   AfxMessageBox("補間開始");//戻り値正常(補間開始)
	}

	BEGIN_MESSAGE_MAP(CMC_SAMPLEDlg, CDialog)          // WM_CIP_ENDメッセージ受信関数設定
		ON_MESSAGE( WM_CIP_END, OnMsg_CIP )
	END_MESSAGE_MAP()

	// WM_CIP_ENDメッセージ受信関数
	afx_msg LRESULT CMC_SAMPLEDlg::OnMsg_CIP(WPARAM BoardNo, LPARAM Status)
	{
		if(Status == CIP_END)   AfxMessageBox("補間正常終了"); // 戻り値正常(補間終了)
		return 0;
	}
```

[VB.NET]

```
		'４軸補間データ
		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)             '補間モード設定。Ｘ，Ｙ，Ｚ，Ｕ軸指定。
		'速度関連パラメータ設定
		Call Nmc_StartSpd(No, IcNo, AXIS_X, 4000000)  '定速ドライブにするため4M
		                                              '他の設定記述は省略
		'4軸連続補間実行。データ数2, X,Y,Z,U軸
		Ret = Nmc_4CIPExecMC8500P_BG(Handle.ToInt32, No, IcNo, Data4Cip, 2, &HF, False)
		If Ret = CIP_START Then    '戻り値正常(補間開始)
			Call MsgBox("補間開始")
		End If
	End Sub

	'WM_CIP_ENDメッセージ受信関数
	Protected Overrides Sub WndProc(ByRef m As Message)
		Select Case (m.Msg)
			Case WM_CIP_END                        '連続補間終了メッセージ
				If m.LParam.ToInt32 = CIP_END Then   '戻り値正常(補間終了)
					Call MsgBox("補間正常終了")
				End If
				Exit Select
		End Select
		MyBase.WndProc(m)
	End Sub
```

注意

この関数を実行する前に、必ず補間モード設定(2A)Hで補間軸を設定して下さい。補間モード設定で設定した軸と、引数IpAxisで指定する軸は必ず同じ値にして下さい。
この関数は定速ドライブで動作します。定速ドライブになるように速度パラメータを設定してください。

<table>
<tr><th>関数名</th><th>機能　及び　内容</th></tr>
<tr><td>Nmc_IPStop</td><td>補間処理を中断する。
補間ドライブを即停止し、Nmc_xxxの補間関数で実行中の補間処理を終了します。

Nmc_IPStopを使用して補間処理を中断した場合、各補間関数の戻り値は下記のエラーコードです。
　◆ＢＰ補間　：　BP_USER_STOP
　◆連続補間　：　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

入力パラメータ
　No　　ボード番号（ボード上のロータリースイッチの値(0～15)）
　IcNo　IC番号(0～1)。搭載ICが1つの時は0, 2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。

戻り値
　[VC]　　　補間処理の中断に成功するとTRUE、失敗するとFALSE
　[VB.NET]　補間処理の中断に成功すると０以外、失敗すると０

使用例
　[VC]　　　Nmc_IPStop(No, IcNo);　　　　// 補間処理を中断する
　[VB.NET]　Call Nmc_IPStop(No, IcNo)　　　'補間処理を中断する</td></tr>
<tr><td>Nmc_IPGetMsgNo</td><td>補間終了時に受信した下記メッセージの引数wParamからボード番号とＩＣ番号を取得する。
　◆ＢＰ補間　：　WM_BP_END
　◆連続補間　：　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)

入力パラメータ
　wParam　補間終了時に受信したメッセージの引数wParam
　No　　　[VC]ボード番号を格納する変数のアドレス。
　　　　　[VB.NET]ボード番号を格納する変数。
　IcNo　　[VC]ＩＣ番号を格納する変数のアドレス。
　　　　　[VB.NET]ＩＣ番号を格納する変数。

戻り値
　なし

使用例
　[VC]　　　int　BdNo;　　　　　　// ボード番号
　　　　　　int　IcNo;　　　　　　//　ＩＣ番号
　　　　　　Nmc_IPGetMsgNo(wParam, &BdNo, &IcNo);

　[VB.NET]
　　Dim BdNo As Integer　　　　　'ボード番号
　　Dim IcNo As Integer　　　　　'ＩＣ番号（0～1）
　　Call Nmc_IPGetMsgNo(m.WParam.ToInt32(), BdNo, IcNo)</td></tr>
</table>

<table>
<tr><th>関数名</th><th>機能　及び　内容</th></tr>
<tr><td>Nmc_HLValueExec</td><td>
指定した補間データでヘリカル演算を実行する。

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

入力パラメータ
No　　　　ボード番号（ボード上のロータリースイッチの値(0～15)）
IcNo　　　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。
　　　　　詳細は補足説明(7)参照。
pDataHl　　ヘリカル補間データの構造体(ユーザー定義型)配列の先頭アドレス(DATA_HLのアドレス)。
　　　　　DATA_HLに実行するヘリカル演算のデータをセットし、アドレスに指定する。
　　　　　詳細は補足説明(3)参照。
　　　　　DirectionにはＣＷ円弧補間(HL_DIR_CW)、ＣＣＷ円弧補間(HL_DIR_CCW)のどちらかを指定する。

戻り値
演算処理が正常終了した場合は HL_VALUE_END が返ります。
演算処理でエラーが発生した場合は、下記エラーコードが返ります。

■正常終了
HL_VALUE_END　　　ヘリカル演算正常終了

■エラーコード
HL_PARAM_ERR　　　引数の値が正しくない
HL_NOT_OPEN_ERR　　指定したボードがオープンされていない
HL_FUNC_ERR　　　　使用できない関数を使用
HL_OTHER_ERR　　　その他のエラー

使用例
[VC]
<pre>
{
  DATA_IPMODE IpMode;   // 補間モード

  // 補間モード設定
  IpMode.Cxiv = FALSE;   // 補間軸入れ替え無し
  IpMode.Lmdf = FALSE;   // 短軸パルス均一化無効
  IpMode.Vcnst = 3;   // 2軸高精度線速一定

  DATA_HL DataHL;   // ヘリカル補間データ

  // ヘリカル補間データ設定
  DataHL.Direction = HL_DIR_CCW;   // 回転方向(CCW円弧補間)
  DataHL.Speed = 1000;   // 速度
  DataHL.CenterX = 0;   // Ｘ軸円弧中心点
  DataHL.CenterY = 10000;   // Ｙ軸円弧中心点
  DataHL.EndPX = 0;   // Ｘ軸終点
  DataHL.EndPY = 0;   // Ｙ軸終点
  DataHL.MoveZ = 3000;   // Ｚ軸移動量
  DataHL.MoveU = 400;   // Ｕ軸移動量
  DataHL.HlNum = 1;   // ヘリカル回転数
  DataHL.HlValue = 0;   // ヘリカル演算値
  DataHL.IpMode = IpMode;   // 補間モード設定

  // ヘリカル演算実行
  Ret = Nmc_HLValueExec(0, 0, &DataHL);
  if(Ret == HL_VALUE_END)   AfxMessageBox("演算終了");   // ヘリカル演算正常終了

  // ヘリカル演算値取得
  Value = DataHL.HlValue;   // 演算終了後、演算結果がHlValueに格納される
}
</pre>
</td></tr>
</table>

	[VB.NET] ``` Dim IpMode As DATA_IPMODE '補間モード ``` ...

```
[VB.NET]
    Dim IpMode As DATA_IPMODE         '補間モード
    Dim DataHL As DATA_HL             'ヘリカル補間データ

    '補間モード設定
    IpMode.Cxiv = False               '円弧補間のときの、補間軸の入れ替えない
    IpMode.Lmdf = False               '短軸パルス均一化(無効)
    IpMode.Vcnst = 3                  '線速一定(２軸高精度)

    'ヘリカル補間データ設定
    DataHL.Direction = HL_DIR_CCW    '回転方向(CCW円弧補間)
    DataHL.Speed = 1000               '速度
    DataHL.CenterX = 0                'Ｘ軸円弧中心点
    DataHL.CenterY = 10000            'Ｙ軸円弧中心点
    DataHL.EndPX = 0                  'Ｘ軸終点
    DataHL.EndPY = 0                  'Ｙ軸終点
    DataHL.MoveZ = 3000               'Ｚ軸移動量
    DataHL.MoveU = 400                'Ｕ軸移動量
    DataHL.HlNum = 1                  'ヘリカル回転数
    DataHL.HlValue = 0                'ヘリカル演算値
    DataHL.IpMode = IpMode            '補間モード設定

    'ヘリカル演算実行
    Res = Nmc_HLValueExec(0, 0, DataHL)
    If Res = HL_VALUE_END Then        'ヘリカル演算正常終了
        'ヘリカル演算値取得
        Value = DataHL.HlValue        '演算終了後、演算結果がHlValueに格納される
        Call MsgBox("演算終了")
    End If
```

注意

この関数を実行すると、ドライバ内で、引数のヘリカル補間データの構造体の値で補間モード設定(2A)Hを行います。関数実行前に設定した補間モード設定(2A)Hは無効となりますのでご注意下さい。

関数名	機能　及び　内容

Nmc_HLExec	指定した補間データでヘリカル補間を実行する。
	
	VC　DWORD　Nmc_HLExec(int No, int IcNo, DATA_HL* pDataHl);
	VB.NET　Function Nmc_HLExec Lib "MC8000P.DLL" (ByVal No As Integer, ByVal IcNo As Integer, ByRef pDataHl As DATA_HL) As Integer
	
	入力パラメータ
	No　ボード番号（ボード上のロータリースイッチの値(0～15)）
	IcNo　IC番号(0～1)。搭載ICが1つの時は0,2つ以上の時はIC-Aが0,IC-Bが1。詳細は補足説明(7)参照。
	pDataHl　ヘリカル補間データの構造体(ユーザー定義型)配列の先頭アドレス(DATA_HLのアドレス)。DATA_HLに実行するヘリカル演算のデータをセットし、アドレスに指定する。詳細は補足説明(3)参照。
	
	ヘリカル補間データの構造体の変数 ヘリカル演算値HLValue の値を0に設定すると、ドライバ内でヘリカル演算を行い、演算で算出された値を使用してヘリカル補間を行います。ヘリカル演算値HLValueは演算により算出された演算値に書き換えられます。
	ヘリカル演算値を0以上に設定するとドライバ内ではヘリカル演算を行わず、設定した値を使用してヘリカル補間を行います。
	
	戻り値
	ヘリカル補間コマンドが正常に書き込まれた場合は HL_START が返ります。
	ヘリカル補間が開始する前にエラーが発生した場合は、下記のヘリカル補間開始前のエラーコードが返ります。
	
	■正常開始
	HL_START　ヘリカル補間開始
	
	■エラーコード(ヘリカル補間開始前のエラー)
	HL_CNT_ERR　指定されたデータ数が範囲外
	HL_PARAM_ERR　引数の値が正しくない
	HL_NOT_OPEN_ERR　指定したボードがオープンされていない
	HL_FUNC_ERR　使用できない関数を使用
	HL_OTHER_ERR　その他のエラー
	
	使用例
	[VC]
	DATA_IPMODE IpMode;　// 補間モード
	
	// 補間モード設定
	IpMode.Cxiv = FALSE;　// 補間軸入れ替え無し
	IpMode.Lmdf = FALSE;　// 短軸パルス均一化無効
	IpMode.Vcnst = 3;　// 2軸高精度線速一定
	
	DATA_HL DataHL;　// ヘリカル補間データ
	
	// ヘリカル補間データ設定
	DataHL.Direction = HL_DIR_CCW;　// 回転方向(CCW円弧補間)
	DataHL.Speed = 1000;　// 速度
	DataHL.CenterX = 0;　// Ｘ軸円弧中心点
	DataHL.CenterY = 10000;　// Ｙ軸円弧中心点
	DataHL.EndPX = 0;　// Ｘ軸終点
	DataHL.EndPY = 0;　// Ｙ軸終点
	DataHL.MoveZ = 3000;　// Ｚ軸移動量
	DataHL.MoveU = 400;　// Ｕ軸移動量
	DataHL.HlNum = 1;　// ヘリカル回転数
	DataHL.HlValue = 0;　// ヘリカル演算値(ヘリカル演算値をドライバ内で計算する)
	DataHL.IpMode = IpMode;　// 補間モード設定
	
	// ヘリカル補間実行
	Ret = Nmc_HLExec(0, 0, &DataHL);
	if(Ret == HL_START)　AfxMessageBox("ヘリカル補間開始");　// ヘリカル補間開始
	
	// ヘリカル演算値取得
	Value = DataHL.HlValue;　// 演算結果がHlValueに格納される

	[VB.NET]
	`Dim IpMode As DATA_IPMODE      '補間モード`
	`Dim DataHL As DATA_HL          'ヘリカル補間データ`
	
	`'補間モード設定`
	`IpMode.Cxiv = False            '円弧補間のときの、補間軸の入れ替えない`
	`IpMode.Lmdf = False            '短軸パルス均一化(無効)`
	`IpMode.Vcnst = 3               '線速一定(２軸高精度)`
	
	`'ヘリカル補間データ設定`
	`DataHL.Direction = HL_DIR_CCW  '回転方向(CCW円弧補間)`
	`DataHL.Speed = 1000            '速度`
	`DataHL.CenterX = 0             'Ｘ軸円弧中心点`
	`DataHL.CenterY = 10000         'Ｙ軸円弧中心点`
	`DataHL.EndPX = 0               '終点 (X軸)`
	`DataHL.EndPY = 0               '終点 (Y軸)`
	`DataHL.MoveZ = 3000            'Ｚ軸移動量`
	`DataHL.MoveU = 400             'Ｕ軸移動量`
	`DataHL.HlNum = 1               'ヘリカル回転数`
	`DataHL.HlValue = 0             'ヘリカル演算値(ヘリカル演算値をドライバ内で計算する)`
	`DataHL.IpMode = IpMode         '補間モード設定`
	
	`'ヘリカルドライブ実行`
	`Res = Nmc_HLExec(0, 0, DataHL)`
	`If Res = HL_START Then          'ヘリカル補間開始`
	`    Call MsgBox("ヘリカル補間開始")`
	`End If`
	`'ヘリカル演算値取得`
	`Value = DataHL.HlValue          '演算結果がHlValueに格納される`
	
	注意
	この関数を実行すると、ドライバ内で、引数のヘリカル補間データの構造体の値で補間モード設定(2A)Hを行います。関数実行前に設定した補間モード設定(2A)Hは無効となりますのでご注意下さい。
	この関数は定速ドライブで動作します。そのため、関数内でＸ軸の初速度はドライブ速度の値に設定されます。

■補足説明

(1) 各定義は、下記のファイルで行っています。

```
VC++   ・・・MC8000P_DLL.h
VB.NET・・・MC8000P_DLL.vb
```

ＶＣ＋＋、ＶＢ．ＮＥＴの定義内容を以下に示します。

①レジスタ番号
```
#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
```

②軸定義
```
#define AXIS_ALL        0xF    // 全軸
#define AXIS_X          0x1    // Ｘ軸
#define AXIS_Y          0x2    // Ｙ軸
#define AXIS_Z          0x4    // Ｚ軸
#define AXIS_U          0x8    // Ｕ軸
#define AXIS_NONE       0      // 軸指定無し
```

③デバイスＩＤ
```
#define ID_MC8541P      0xA808 // MC8541P
#define ID_MC8581P      0xA809 // MC8581P
```

（使用例）MC8541PはID_MC8541P、MC8581PはID_8581P

④コマンド定義　　　※使用できるコマンドは各ＩＣの取扱説明書参照。
```
// ドライブ命令
#define MC8500P_CMD_DRVRL       0x0050      // 相対位置ドライブ
#define MC8500P_CMD_DRVNR       0x0051      // 反相対位置ドライブ
#define MC8500P_CMD_DRVVP       0x0052      // ＋方向連続パルスドライブ
#define MC8500P_CMD_DRVVM       0x0053      // －方向連続パルスドライブ
#define MC8500P_CMD_DRVAB       0x0054      // 絶対位置ドライブ
#define MC8500P_CMD_DRVSBRK     0x0056      // ドライブ減速停止
#define MC8500P_CMD_DRVFBRK     0x0057      // ドライブ即停止
#define MC8500P_CMD_DIRCP       0x0058      // 方向信号＋設定
#define MC8500P_CMD_DIRCM       0x0059      // 方向信号－設定
#define MC8500P_CMD_HMSRC       0x005A      // 自動原点出し実行

// 補間命令
#define MC8500P_CMD_1CIP        0x0060      // 1軸直線補間ドライブ（マルチチップ用）※MC8581P専用
#define MC8500P_CMD_2CIP        0x0061      // 2軸直線補間ドライブ
#define MC8500P_CMD_3CIP        0x0062      // 3軸直線補間ドライブ
#define MC8500P_CMD_4CIP        0x0063      // 4軸直線補間ドライブ
#define MC8500P_CMD_CIPCW       0x0064      // ＣW円弧補間ドライブ
#define MC8500P_CMD_CIPCCW      0x0065      // ＣＣW円弧補間ドライブ
#define MC8500P_CMD_BPIP2       0x0066      // 2軸ビットパターン補間ドライブ
#define MC8500P_CMD_BPIP3       0x0067      // 3軸ビットパターン補間ドライブ
```

```
#define MC8500P_CMD_BPIP4        0x0068      // 4軸ビットパターン補間ドライブ
#define MC8500P_CMD_HLCW         0x0069      // CWヘリカル補間ドライブ
#define MC8500P_CMD_HLCCW        0x006A      // CCWヘリカル補間ドライブ
#define MC8500P_CMD_HLPCW        0x006B      // CWヘリカル演算
#define MC8500P_CMD_HLPCCW       0x006C      // CCWヘリカル演算
#define MC8500P_CMD_DECEN        0x006D      // 減速有効
#define MC8500P_CMD_DECDIS       0x006E      // 減速無効
#define MC8500P_CMD_CLRINTRPT    0x006F      // 補間割り込みクリア
#define MC8500P_CMD_IPSTEP       0x006F      // 補間ステップ

// その他の命令
#define MC8500P_CMD_VINC         0x0070      // 速度増加
#define MC8500P_CMD_VDEC         0x0071      // 速度減少
#define MC8500P_CMD_TMSTA        0x0073      // タイマー始動
#define MC8500P_CMD_TMSTP        0x0074      // タイマー停止
#define MC8500P_CMD_DHOLD        0x0077      // ドライブ開始ホールド
#define MC8500P_CMD_DFREE        0x0078      // ドライブ開始フリー
#define MC8500P_CMD_R2CLR        0x0079      // エラー･終了ステータスクリア
#define MC8500P_CMD_RR3P0        0x007A      // RR3 ページ0表示
#define MC8500P_CMD_RR3P1        0x007B      // RR3 ページ1表示
#define MC8500P_CMD_TXCLR        0x007C      // 終点最大値クリア
#define MC8500P_CMD_NOP          0x001F      // NOP
#define MC8500P_CMD_RST          0x00FF      // コマンドリセット
```

⑤補間終了メッセージ、終了ステータス

```
// 補間終了メッセージ
#define WM_BP_END               (WM_USER + 1) // ＢＰ補間終了メッセージ
#define WM_CIP_END              (WM_USER + 2) // 連続補間終了メッセージ

//***** BP補間　終了ステータス *****
// ■正常
#define BP_START                0x101  // バックグラウンドでBP補間を開始した
#define BP_END                  0x102  // BP補間正常終了

// ■補間開始前のエラー
#define BP_CNT_ERR              0x111  // 指定されたデータ数が範囲外
#define BP_ALREADY_EXEC         0x112  // 既にＢＰ補間、あるいは連続補間が実行中
#define BP_THREAD_ERR           0x113  // スレッドを起動できなかった
#define BP_MALLOC_ERR           0x114  // メモリを確保できなかった
#define BP_PARAM_ERR            0x116  // 引数の値が正しくない
#define BP_NOT_OPEN_ERR         0x117  // 指定したボードがオープンされていない
#define BP_OTHER_ERR            0x118  // その他のエラー
#define BP_FUNC_ERR             0x119  // 使用できない関数を使用

// ■補間実行中のエラー
#define BP_STOP                 0x121  // BP補間が途中で停止した(速度が速く次ﾃﾞｰﾀのｽﾀｯｸが間に合わなかった場合)
#define BP_USER_STOP            0x122  // BP補間実行中にユーザーが中断した
#define BP_DRIVE_ERR            0x123  // BP補間実行中にボードでエラー発生 (RR0にエラー情報がセットされた
#define BP_STOP_CERR            0x124  // 連続補間が途中で停止した (RR2エラーによる停止の場合)
#define BP_GETSC_ERR            0x125  // スタックカウンタ読み出しエラー

//***** 連続補間　終了ステータス *****
// ■正常
#define CIP_START               0x201  // バックグラウンドで連続補間を開始した
#define CIP_END                 0x202  // 連続補間正常終了

// ■補間開始前のエラー
#define CIP_CNT_ERR             0x211  // 指定されたデータ数が範囲外
#define CIP_ALREADY_EXEC        0x212  // 既にＢＰ補間、あるいは連続補間が実行中
#define CIP_THREAD_ERR          0x213  // スレッドを起動できなかった
#define CIP_MALLOC_ERR          0x214  // メモリを確保できなかった
#define CIP_CMD_ERR             0x215  // コマンドエラー (ユーザーが指定したコマンドが間違っている)
#define CIP_PARAM_ERR           0x216  // 引数の値が正しくない
#define CIP_NOT_OPEN_ERR        0x217  // 指定したボードがオープンされていない
#define CIP_OTHER_ERR           0x218  // その他のエラー
```

```
#define CIP_FUNC_ERR            0x219  // 使用できない関数を使用

// ■補間実行中のエラー
#define CIP_STOP                0x221  // 連続補間が途中で停止した(速度が速く次ﾃﾞｰﾀのｾｯﾄが間に合わなかった場合)
#define CIP_USER_STOP           0x222  // 連続補間実行中にユーザーが中断した
#define CIP_DRIVE_ERR           0x223  // 連続補間実行中にボードでエラー発生 (RR0にｴﾗｰ情報がセットされた)
#define CIP_STOP_CERR           0x224  // 連続補間が途中で停止した (RR2エラーによる停止の場合)
#define CIP_GETSC_ERR           0x225  // スタックカウンタ読み出しエラー

//***** ヘリカル補間  終了ステータス *****
// ■正常
#define HL_START                0x301       // ヘリカル補間開始
#define HL_VALUE_END            0x302       // ヘリカル演算終了

// ■ヘリカル補間開始前のエラー
#define HL_CNT_ERR              0x311       // 指定されたデータ数が範囲外
#define HL_PARAM_ERR            0x312       // 引数の値が正しくない
#define HL_NOT_OPEN_ERR         0x313       // 指定したボードがオープンされていない
#define HL_FUNC_ERR             0x314       // 使用できない関数を使用
#define HL_OTHER_ERR            0x315       // その他のエラー
```

⑥ヘリカル補間回転方向

```
#define HL_DIR_CW               0           // ＣＷヘリカル補間
#define HL_DIR_CCW              1           // ＣＣＷヘリカル補間
```

(2) 軸指定の方法は、下記の通りです。

軸	VC、VB.NET
X	AXIS_X
Y	AXIS_Y
Z	AXIS_Z
U	AXIS_U
全軸	AXIS_ALL

①１軸指定の場合

AXIS_X, AXIS_Y, AXIS_Z, AXIS_U のいずれかを指定します。

(使用例) Ｘ軸にドライブ速度 1000 を設定する

```
[VC]       Nmc_Speed(No, IcNo, AXIS_X, 1000);
[VB.NET]   Call Nmc_Speed(No, IcNo, AXIS_X, 1000)
```

②２軸指定の場合

ビットＯＲ演算子を使用します。
例えばＸ軸とＹ軸を一度に指定する場合、
[VC]・・・AXIS_X | AXIS_Y を指定します。
[VB.NET]・・・AXIS_X Or AXIS_Y を指定します。

(使用例) Ｘ,Ｙ軸にドライブ速度 1000 を設定する

```
[VC]       Nmc_Speed(No, IcNo, AXIS_X | AXIS_Y, 1000);
[VB.NET]   Call Nmc_Speed(No, IcNo, AXIS_X Or AXIS_Y, 1000)
```

③３軸指定の場合

ビットＯＲ演算子を使用します。
例えばＸ軸とＹ軸とＺ軸を一度に指定する場合、
[VC]・・・AXIS_X | AXIS_Y | AXIS_Z を指定します。
[VB.NET]・・・AXIS_X Or AXIS_Y Or AXIS_Z を指定します。

(使用例) Ｘ,Ｙ,Ｚ軸にドライブ速度 1000 を設定する

```
[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)
```

④全軸指定の場合
AXIS_ALL を指定します。

（使用例） 全軸にドライブ速度 1000 を設定する
［VC］ Nmc_Speed(No, IcNo, AXIS_ALL, 1000);
［VB.NET］ Call Nmc_Speed(No, IcNo, AXIS_ALL, 1000)

(3) 補間関数で使用する構造体の定義は、下記の通りです。
①ＶＣ＋＋

```
// ２軸ＢＰ補間
typedef struct _DATA_2BP
{
    USHORT Bp1p;        // BP1Pデータ
    USHORT Bp1m;        // BP1Mデータ
    USHORT Bp2p;        // BP2Pデータ
    USHORT Bp2m;        // BP2Mデータ
} DATA_2BP;

// ３軸ＢＰ補間
typedef struct _DATA_3BP
{
    USHORT Bp1p;        // BP1Pデータ
    USHORT Bp1m;        // BP1Mデータ
    USHORT Bp2p;        // BP2Pデータ
    USHORT Bp2m;        // BP2Mデータ
    USHORT Bp3p;        // BP3Pデータ
    USHORT Bp3m;        // BP3Mデータ
} DATA_3BP;

// ４軸ＢＰ補間
typedef struct _DATA_4BP
{
    USHORT Bp1p;        // BP1Pデータ
    USHORT Bp1m;        // BP1Mデータ
    USHORT Bp2p;        // BP2Pデータ
    USHORT Bp2m;        // BP2Mデータ
    USHORT Bp3p;        // BP3Pデータ
    USHORT Bp3m;        // BP3Mデータ
    USHORT Bp4p;        // BP4Pデータ
    USHORT Bp4m;        // BP4Mデータ
} DATA_4BP;

// ２軸連続補間
typedef struct _DATA_2CIP_MC8500P
{
    USHORT Command;     // 命令番号
                        // （MC8500P_CMD_2CIP, MC8500P_CMD_CIPCW, MC8500P_CMD_CIPCCWのいずれかをセットする）
    long   Speed;       // 速度（速度を変更する場合は１～４，０００，０００、変更しない場合は０をセットする）
    long   EndP1;       // 終点（第１軸）
    long   EndP2;       // 終点（第２軸）
    long   Center1;     // 円弧中心点（第１軸）
    long   Center2;     // 円弧中心点（第２軸）
} DATA_2CIP_MC8500P;    // 注：補間軸は補間モード設定(2A)Hで指定する。

// ３軸連続補間
typedef struct _DATA_3CIP_MC8500P
{
    long   EndP1;       // 終点（第１軸）
    long   EndP2;       // 終点（第２軸）
    long   EndP3;       // 終点（第３軸）
    long   Speed;       // 速度（速度を変更する場合は１～４，０００，０００、変更しない場合は０をセットする）
} DATA_3CIP_MC8500P;    // 注：補間軸は補間モード設定(2A)Hで指定する。
```

```
// ４軸連続補間
typedef struct _DATA_4CIP_MC8500P
{
    long   EndP1;        // 終点（第１軸）
    long   EndP2;        // 終点（第２軸）
    long   EndP3;        // 終点（第３軸）
    long   EndP4;        // 終点（第４軸）
    long   Speed;        // 速度（速度を変更する場合は１～４，０００，０００、変更しない場合は０をセットする）
} DATA_4CIP_MC8500P;     // 注：補間軸は補間モード設定(2A)Hで指定する。

// 補間モード設定
typedef struct _DATA_IPMODE
{
    BOOL   Cxiv;         // 円弧補間のときの、補間軸の入れ替え（TURE:入れ替える、FALSE:入れ替えない）
    BOOL   Lmdf;         // 短軸パルス均一化 （TRUE:有効、FALSE:無効）
    USHORT Vcnst;        // 線速一定（0:線速一定なし、1:２軸簡易、2:３軸簡易、3:２軸高精度）
}
DATA_IPMODE;

// ヘリカル補間
typedef struct _DATA_HL
{
    USHORT Direction;    // 回転方向（HL_DIR_CW, HL_DIR_CCWのいずれかをセットする）
    long   Speed;        // 速度（１～２，０００，０００）
    long   CenterX;      // 円弧中心点（X軸）
    long   CenterY;      // 円弧中心点（Y軸）
    long   EndPX;        // 終点（X軸）
    long   EndPY;        // 終点（Y軸）
    long   MoveZ;        // 移動量（Z軸）(0:Z軸の使用なし、0以外:Z軸の移動量)
    long   MoveU;        // 移動量（U軸）(0:U軸の使用なし、0以外:U軸の移動量)
    USHORT HlNum;        // ヘリカル回転数（０～６５，５３５）
    long   HlValue;      // ヘリカル演算値
                          (0:ヘリカル演算値を求めてから補間を実行する, 0以上:セットした演算値で補間を実行する)
    DATA_IPMODE   IpMode;// 補間モード設定
}
DATA_HL;
```

②ＶＢ．ＮＥＴ

```
’ ２軸ＢＰ補間
Structure DATA_2BP
     Dim Bp1p As Short      ’ BP1Pデータ
     Dim Bp1m As Short      ’ BP1Mデータ
     Dim Bp2p As Short      ’ BP2Pデータ
     Dim Bp2m As Short      ’ BP2Mデータ
End Structure

’ ３軸ＢＰ補間
Structure DATA_3BP
     Dim Bp1p As Short      ’ BP1Pデータ
     Dim Bp1m As Short      ’ BP1Mデータ
     Dim Bp2p As Short      ’ BP2Pデータ
     Dim Bp2m As Short      ’ BP2Mデータ
     Dim Bp3p As Short      ’ BP3Pデータ
     Dim Bp3m As Short      ’ BP3Mデータ
End Structure
```

```
’ ４軸ＢＰ補間
Structure DATA_4BP
    Dim Bp1p As Short       ’ BP1Pデータ
    Dim Bp1m As Short       ’ BP1Mデータ
    Dim Bp2p As Short       ’ BP2Pデータ
    Dim Bp2m As Short       ’ BP2Mデータ
    Dim Bp3p As Short       ’ BP3Pデータ
    Dim Bp3m As Short       ’ BP3Mデータ
    Dim Bp4p As Short       ’ BP4Pデータ
    Dim Bp4m As Short       ’ BP4Mデータ
End Structure

’ ２軸連続補間
Structure DATA_2CIP_MC8500P
    Dim Cmd As Short         ’ 命令番号（CMD_IP_2ST, CMD_IP_CW, CMD_IP_CCWのいずれかをセットする）
    Dim Speed As Integer     ’ 速度（速度を変更する場合は１～４，０００，０００、変更しない場合は０をセットする）
    Dim EndP1 As Integer     ’ 終点（第１軸）
    Dim EndP2 As Integer     ’ 終点（第２軸）
    Dim Center1 As Integer   ’ 円弧中心点（第１軸）
    Dim Center2 As Integer   ’ 円弧中心点（第２軸）
 End Structure               ’ 注：補間軸は補間モード設定(2A)Hで指定する

’３軸連続補間
 Structure DATA_3CIP_MC8500P
    Dim EndP1 As Integer     ’ 終点（第１軸）
    Dim EndP2 As Integer     ’ 終点（第２軸）
    Dim EndP3 As Integer     ’ 終点（第３軸）
    Dim Speed As Integer     ’ 速度（速度を変更する場合は１～４，０００，０００、変更しない場合は０をセットする）
 End Structure               ’ 注：補間軸は補間モード設定(2A)Hで指定する

’４軸連続補間
 Structure DATA_4CIP_MC8500P
    Dim EndP1 As Integer     ’ 終点（第１軸）
    Dim EndP2 As Integer     ’ 終点（第２軸）
    Dim EndP3 As Integer     ’ 終点（第３軸）
    Dim EndP4 As Integer     ’ 終点（第４軸）
    Dim Speed As Integer     ’ 速度（速度を変更する場合は１～４，０００，０００、変更しない場合は０をセットする）
 End Structure               ’ 注：補間軸は補間モード設定(2A)Hで指定する

’ 補間モード設定
 Structure DATA_IPMODE
    Dim Cxiv As Boolean       ’ 円弧補間のときの、補間軸の入れ替え(True:入れ替える、False:入れ替えない)
    Dim Lmdf As Boolean       ’ 短軸パルス均一化（TRUE:有効、FALSE:無効）
    Dim Vcnst As UShort       ’ 線速一定（0:線速一定なし、1:２軸簡易、2:３軸簡易、3:２軸高精度）
 End Structure

’ ヘリカル補間
 Structure DATA_HL
    Dim Direction As UShort    ’回転方向（HL_DIR_CW:ＣＷ円弧補間, HL_DIR_CCW：CCW円弧補間のいずれかをセットする）
    Dim Speed       As Integer ’速度（１～２０００００００）
    Dim CenterX     As Integer ’円弧中心点（X軸）
    Dim CenterY     As Integer ’円弧中心点（Y軸）
    Dim EndPX       As Integer ’終点（X軸）
    Dim EndPY       As Integer ’終点（Y軸）
    Dim MoveZ       As Integer ’移動量（Z軸）(0:Z軸の使用なし、0以外:Z軸の移動量)
    Dim MoveU       As Integer ’移動量（U軸）(0:U軸の使用なし、0以外:U軸の移動量)
    Dim HlNum       As UShort  ’ヘリカル回転数 （０～６５，５３５）
    Dim HlValue     As Integer ’ヘリカル演算値 （0:ヘリカル演算値を求めてから補間を実行する,
                               ’0以上:セットした演算値で補間を実行する）
    Dim IpMode      As DATA_IPMODE  ’補間モード設定
 End Structure
```

構造体の使用例を以下に示します。（ＶＣの場合

例１．定義、初期化が別の場合

```
DATA_2BP [] Data2Bp = new DATA_2BP[4];   // ２軸ＢＰ補間データ

// 補間データ設定
Data2Bp[0].Bp1p = 0xFF30;  // 1111 1111 0011 0000  BP1＋方向        10パルス
Data2Bp[0].Bp1m = 0;       // 0000 0000 0000 0000  BP1－方向         0パルス
Data2Bp[0].Bp2p = 0;       // 0000 0000 0000 0000  BP2＋方向         0パルス
Data2Bp[0].Bp2m = 0x84FF;  // 1000 0100 1111 1111  BP2－方向        10パルス

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

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

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

例２．定義と初期化を同時に行う場合

```
// ２軸ＢＰ補間データ      BP1P,    BP1M,    BP2P,    BP2M  (MCX514取説3.4.8 ビットパターン補間の実例のデータ)
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) 補間関数で指定する補間軸(IpAxis)の指定は下記の通りです。
４軸Ｘ、Ｙ、Ｚ、Ｕの中から補間する軸を指定します。

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

◆各ビットの説明

D3～0　U-EN～X-EN 補間ドライブを行う軸を指定します。下表にビットに対応した補間軸を示します。
　　　　　　０：補間軸として不使用、1：補間軸として使用

軸	指定ビット
X	D0
Y	D1
Z	D2
U	D3

主軸はX-EN＞Y-EN＞Z-EN＞U-ENの優先順で、1が設定されているビットの軸が選ばれます。

(5) 連続補間関数実行時の速度変更について

連続補間関数は、補間実行中に速度変更を行う事ができます。各セグメント毎に速度を設定できます。
補間実行中に速度変更を行う場合は、関数のパラメータSpdChgFlgにTRUE(True)を設定します。

◆各セグメント毎の速度の設定方法

DATA_2CIP_MC8500PのSpeed、DATA_3CIP_MC8500PのSpeed、DATA_4CIP_MC8500PのSpeed に、
各セグメントの速度を設定します。
・前のセグメントと異なる速度にする場合は、1～4,000,000 の速度を設定します。
・前のセグメントと同じ速度のままにする場合は、0 を設定します。

(6) アドレスの指定

Nmc_OutPort,Nmc_InPort関数のアドレスには、各ボードの取扱説明書に記載しているＩ／Ｏアドレスを指定します。

ライトレジスタ、リードレジスタなどのＩ／Ｏアドレスは下記の通りです。詳細は各ボードの取扱説明書を参照。

ボード名	ライトレジスタのアドレス	リードレジスタのアドレス
MC8541P	０～７	０～７
MC8581P	０～７	０～７

注１：アドレスは１６進数で記載しています。
注２：ライトレジスタ、リードレジスタにアクセスする場合、Nmc_OutPort,Nmc_InPort関数ではなく、専用の関数を使用した方が便利です。

(7) ＩＣ番号の指定は下記の通りです。

①ＩＣを１つ搭載しているボードの場合は０を指定します。

②ＩＣを２つ搭載しているボードの場合は次の値を指定します。
・ＩＣ－Ａの場合は０
・ＩＣ－Ｂの場合は１

ボードの搭載ＩＣ数と指定するＩＣ番号は次の通りです。

ボード名	搭載ＩＣ数	ＩＣ番号
MC8541P	1	0
MC8581P	2	0,1

注： ＩＣとはMCX514などのＩＣの事を指します。

3.4.3　使用方法

■API関数宣言

API関数宣言は、下記の場所で行っています。

VC++　　　　　　　　　　　　: MC8000P_DLL.h
VB.NET　　　　　　　　　　　: MC8000P_DLL.vb

■使用方法

(1) 開始処理・・・各関数を使用する前にNmc_Openを一度実行して下さい。
(2) 終了処理・・・プログラム終了時にNmc_Close、又はNmc_CloseAllを実行して下さい。

■ボード番号について

アプリケーションから指定するボード番号は下記の通りです。

	ボードのロータリースイッチの値	アプリケーションから指定するボード番号（１０進数）
1	０～９	０～９
2	Ａ～Ｆ	１０～１５

■関数使用時の注意

(1) VC++,VB.NET（全ての言語）について

①Nmc_Open関数実行前に各関数を実行した場合の動作保証はできません。

②接続していないボードの番号を誤って指定した場合も､各関数の動作の保証はできません。

③１枚のボードに対して、２つ以上のアプリケーションから同時にアクセス（オープンなど）を行わないで下さい。

④Nmc_Close(Nmc_CloseAll)関数実行後にNmc_Open関数以外の関数を実行した場合の動作保証はできません。

(2) VC++のみ

①割り込み処理関数を使用する場合はWindowsの性格上、割り込み発生からユーザー定義ルーチンへ制御が移行するまでの時間を保証することは出来ません。

②割込みを行う場合は、割込みユーザー関数（Nmc_SetEventで指定した関数）を実行中にクローズ処理（Nmc_Close又はNmc_CloseAll）を実行しないで下さい。
クローズ処理を行う場合は、必ず、割込みユーザー関数が終了している状態で行って下さい。

■ＶＣ＋＋にて割込みを処理する方法

①Nmc_Open関数にて割込みを使用する設定にします。

```
Nmc_Open(No, TRUE);     // 第２引数  TRUE : 割込みを使用する  FALSE : 割込みを使用しない
```

②Nmc_SetEvent関数にて割込みを処理するユーザー関数を設定します。また、許可する割込みを設定します。

```
Nmc_SetEvent(No, MC_EventProc, lpParam);   // ユーザー関数のアドレスと引数を設定
Nmc_WriteReg1(No, IcNo, AXIS_ALL, 0x0080); // 指定したＩＣの停止時割込み発生（全軸）
```

③割込みが発生すると、Nmc_SetEvent関数で設定した割込みユーザー関数が呼び出されます。
割込みユーザー関数では、割込み要因を確認します。RR3の割込み要因は、Nmc_ReadEvent関数にて取得します。

■割込みユーザー関数例

```
DWORD WINAPI MC_EventProc(LPVOID lpParam)
{
      ・・・・
    long RrIrX, RrIrY, RrIrZ, RrIrU;
    Nmc_ReadEvent(No, IcNo, &RrIrX,&RrIrY,&RrIrZ,&RrIrU);
      ・・・・
    return 0;
}
```

④割込み処理するユーザー関数の設定を解除する場合は Nmc_ResetEvent を実行して下さい。
この関数を実行すると、ボードで割込みが発生してもユーザー関数は呼び出されません。

```
Nmc_ResetEvent(No);
```

■連続補間について

連続補間を行う場合、MCX514取扱説明書の「3.7 連続補間」などを必ず参照し、その章に記載している処理をアプリケーションで行ってください。また、連続補間関数（注１）ではこれらの一部の処理をDLLで行っていますので、この連続補間関数を使用して連続補間の処理を行うこともできます。但し、連続補間関数を使用する場合はいくつか注意事項がありますので、ご注意ください。

注１：Nmc_2CIPExecMC8500P, Nmc_3CIPExecMC8500P, Nmc_4CIPExecMC8500P
Nmc_2CIPExecMC8500P_BG, Nmc_3CIPExecMc8500P_BG, Nmc_4CIPExecMC8500P_BG

連続補間関数を使用する場合の注意事項：

引数の短軸パルス均一化モードは必ず無効に設定してください。有効に設定した場合、予期せぬ動作を起こす可能性があります。
連続補間関数では、MCX514のプリバッファに、あらかじめ8セグメント目までのデータをセットしてから実行を開始します。
エラーチェックを行い、エラー発生の場合は関数を終了します。エラーが発生していない場合は、RR0の連続補間のプリバッファスタックカウンタ（SC）の確認を行い、可能になった場合は次のセグメントデータや補間命令を書き込みます。このとき、本関数では書き込み可能とするスタックカウンタ(SC)の値を６としています。（本関数は、連続補間開始後のプリバッファは６段までの使用となっています。）本関数は、連続補間が終了するまでこの処理を繰り返します。エラーチェックと次のセグメントデータ書き込み可能チェック処理などがDLL内部で常にループしているため、本関数実行中にアプリケーションで他の処理も行いたい場合は、本関数の使用が適さない場合があります。この場合は、連続補間関数は使用せず、MCX514取扱説明書を参考にしてアプリケーションで連続補間の処理を作成してください。
連続補間の加減速ドライブについてはMCX514取扱説明書の「3.8 加減速ドライブでの補間」を参照してください。
また、連続補間関数を使用する場合は、関数を実行する前に初速度を４０００００００に設定してください。（本関数実行中に初速度を変更しないでください。）この場合、各セグメント内は定速ドライブになります。

■ＢＰ補間（ビットパターン補間）について

ＢＰ補間を行う場合、MCX514取扱説明書の「3.4 ビットパターン補間」などを必ず参照し、その章に記載している処理をアプリケーションで行ってください。また、ＢＰ補間関数（注２）ではこれらの一部の処理をDLLで行っていますので、このＢＰ補間関数を使用してＢＰ補間の処理を行うこともできます。但し、ＢＰ補間関数を使用する場合はいくつか注意事項がありますので、ご注意ください。

注２：Nmc_2BPExecMC8500P, Nmc_3BPExecMC8500P, Nmc_4BPExecMC8500P
Nmc_2BPExecMC8500P_BG, Nmc_3BPExecMC8500P_BG, Nmc_4BPExecMC8500P_BG

ＢＰ補間関数を使用する場合の注意事項：

ＢＰ補間関数では、MCX514のプリバッファに、あらかじめ8セグメント目までのデータをセットしてから実行を開始します。
エラーチェックを行い、エラー発生の場合は関数を終了します。エラーが発生していない場合は、RR0の連続補間のプリバッファスタックカウンタ（SC）の確認を行い、可能になった場合は次のセグメントデータや補間命令を書き込みます。このとき、本関数では書き込み可能とするスタックカウンタ(SC)の値を６としています。（本関数は、連続補間開始後のプリバッファは６段までの使用となっています。）本関数は、連続補間が終了するまでこの処理を繰り返します。エラーチェックと次のセグメントデータ書き込み可能チェック処理などがDLL内部で常にループしているため、本関数実行中にアプリケーションで他の処理も行いたい場合は、本関数の使用が適さない場合があります。この場合は、ＢＰ補間関数は使用せず、MCX514取扱説明書を参考にしてアプリケーションでＢＰ補間の処理を作成してください。また、ＢＰ補間の加減速ドライブについては、MCX514取扱説明書の「3.8 加減速ドライブでの補間」を参照してください。

■補間関数使用時の注意

(1) 下記の補間関数について、一度に実行できるのは１つの補間関数のみです。
補間関数実行中に、別の補間関数は実行できません。実行した場合、エラーが返ります。

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) 上記の補間関数実行中は、下記の処理を実行しないで下さい。

①補間命令（60h～6Fh）の実行
②補間モード設定（2Ah）の変更

(3) 下記の補間関数はバックグラウンドで実行する為、補間関数開始時に補間データ用のメモリを確保し、ユーザーが指定した補間データをコピーします。そして、バックグラウンドで実行していた補間処理が終了する時にそのメモリを解放し、ユーザーウィンドウにメッセージを送信します。
この為、下記補間関数がバックグラウンドで実行している最中にクローズ処理（Nmc_Close 又は Nmc_CloseAll）を実行しないで下さい。
また、下記補間関数がバックグラウンドで実行している最中にアプリケーションを終了しないで下さい。
補間関数の実行を途中で止めたい時は、補間中断関数(Nmc_IPStop)を実行し、必ず中断メッセージを受け取って下さい。

Nmc_2BPExecMC8500P_BG	Nmc_3BPExecMC8500P_BG	Nmc_4BPExecMC8500P_BG
Nmc_2CIPExecMC8500P_BG	Nmc_3CIPExecMC8500P_BG	Nmc_4CIPExecMC8500P_BG

■ヘリカル補間連関数使用時の注意

(1) 下記のヘリカル補間関連関数について、一度に実行できるのは１つの関数のみです。
ヘリカル補間関数(Nmc_HLExec)実行後のドライブ中に、ヘリカル演算関数(Nmc_HLValueExe)は実行しないでください。
この場合、ヘリカル補間ドライブの停止を確認してから、ヘリカル演算関数(Nmc_HLValueExe)を実行してください。
逆も同様です。動作中に実行した場合の動作保証はできません。

Nmc_HLValueExe　　Nmc_HLExec

(2) 上記のヘリカル補間関連関数実行時、補間モード設定(2Ah)が引数で指定した内容で設定されます。

(3) ヘリカル補間関数(Nmc_HLExec)は定速ドライブで実行します。
そのため本関数実行時、X軸の初速度が指定したドライブ速度の値で設定されます。

(4) 上記のヘリカル補間関連関数実行後の動作中は、下記の処理を実行しないでください。
動作中に実行した場合の動作保証はできません。

①補間命令（60h～6Fh）の実行
②補間モード設定（2Ah）の変更

(5) 上記のヘリカル補間関連関数実行後の動作中は、下記の連続補間関数は実行しないでください。
ヘリカル演算、ヘリカル補間の動作終了を確認してから実行してください。
動作中に実行した場合の動作保証はできません。

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

■マルチスレッドアプリケーション開発時の注意

ここでは、マルチスレッドで動作するアプリケーションを開発する際の注意事項を説明します。

Nmc_xxx の関数では、軸切り替え処理、ＷＲ６，ＷＲ７にデータを書き込む処理、ＲＲ６，ＲＲ７にデータを読み出す処理を実行している関数があります。それぞれの Nmc_xxx 関数は次の通りです。

◆軸切り替え処理を実行している関数

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_DecP	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_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		

◆ＷＲ６，ＷＲ７にデータを書き込む処理を実行している関数

Nmc_Jerk	Nmc_DJerk	Nmc_Acc	Nmc_Dec	Nmc_StartSpd	Nmc_Speed
Nmc_Pulse	Nmc_DecP	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_TPMax
Nmc_HLNumber	Nmc_HLValue				

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

Nmc_OutPortまたはNmc_WriteRegでWR6,WR7に書いた場合

◆ＲＲ６，ＲＲ７にデータを読み出す処理を実行している関数

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

ＷＲ１～ＷＲ３書き込み、ＲＲ２～ＲＲ３読み出し、データ書き込み命令、データ読み出し命令を実行する場合、基本的には次の Nmc_xxx 関数を使用して下さい。

◆ＷＲ１～ＷＲ３書き込み

Nmc_WriteReg1	Nmc_WriteReg2	Nmc_WriteReg3	Nmc_WriteRegSetAxis

◆ＲＲ２～ＲＲ３読み出し

Nmc_ReadReg2	Nmc_ReadReg3P	Nmc_ReadReg3	Nmc_ReadRegSetAxis

◆データ書き込み命令

Nmc_Jerk	Nmc_DJerk	Nmc_Acc	Nmc_Dec	Nmc_StartSpd
Nmc_Speed	Nmc_Pulse	Nmc_DecP	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_TPMax	Nmc_HLNumber
Nmc_HLValue				

Nmc_MRmMode	Nmc_PIO1Mode	Nmc_PIO2Mode	Nmc_HMSrch1Mode	Nmc_HMSrch2Mode
Nmc_FilterMode	Nmc_Sync0Mode	Nmc_Sync1Mode	Nmc_Sync2Mode	Nmc_Sync3Mode

Nmc_WriteData

◆データ読み出し命令

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

ＷＲ１～ＷＲ３書き込み、ＲＲ２～ＲＲ３読み出し、データ書き込み命令、データ読み出し命令を実行する際、これらの関数を使用せずに同じ処理を行った場合、マルチスレッド環境では注意が必要です。

（１）例えば、ＷＲ１書き込み時には Nmc_WriteReg1 を使用しますが、それ以外の方法としては下記の方法などがあります。

```
① Nmc_OutPort(No, MCX_WR0,  0x011F);          // Ｘ軸に切り替える（ＩＣ－Ａ）
② Nmc_OutPort(No, MCX_WR1,  Data);            // ＷＲ１書き込み（ＩＣ－Ａ）
```

また、次のような関数でも同じ処理を実行できます。

```
③ Nmc_WriteReg(No, IcNo, MCX_WR0,  0x011F);   // Ｘ軸に切り替える
④ Nmc_WriteReg(No, IcNo, MCX_WR1,  Data);     // ＷＲ１書き込み
```

この場合、①と②の間、③と④の間に、軸を切り替える Nmc_xxx 関数が実行された場合、別の軸のＷＲ１にデータが書かれてしまいます。

（２）例えば、速度設定時には Nmc_Speed を使用しますが、それ以外の方法としては下記の方法などがあります。

```
① Nmc_OutPort( No, MCX_WR6, Data );           // ＷＲ６書き込み（ＩＣ－Ａ）
② Nmc_OutPort( No, MCX_WR0, 0x0105 );         // ＷＲ６データをＸ軸の速度に設定する（ＩＣ－Ａ）
```

また、次のような関数でも同じ処理を実行できます。

```
③ Nmc_WriteReg6(No, IcNo, Data);              // ＷＲ６書き込み
④ Nmc_Command(No, IcNo, AXIS_X, 0x05);        // ＷＲ６データをＸ軸の速度に設定する
```

この場合、①と②の間、③と④の間に、ＷＲ６，ＷＲ７にデータを書き込む Nmc_xxx 関数が実行された場合、別のデータが速度に書かれてしまいます。

（３）例えば、論理位置カウンタ読み出し時には Nmc_ReadLp を使用しますが、それ以外の方法としては下記の方法などがあります。

```
① Nmc_OutPort( No, MCX_WR0, 0x0130 );  // Ｘ軸の論理位置カウンタをＲＲ６，ＲＲ７に読み出す（ＩＣ－Ａ）
② d6 = Nmc_InPort( No, MCX_RR6 );      // ＲＲ６を読み出す（ＩＣ－Ａ）
③ d7 = Nmc_InPort( No, MCX_RR7 );      // ＲＲ７を読み出す（ＩＣ－Ａ）
```

この場合、①と②の間、②と③の間に、ＲＲ６，ＲＲ７にデータを読み出す Nmc_xxx 関数が実行された場合、ここでは別のデータが読み出されてしまいます。

このように、マルチスレッド環境では、目的の処理を実行するのに２回以上 API 関数をコールする場合は、そのような処理を行わないようにするか、あるいは、排他制御を行う必要があります。

Nmc_xxx の関数を１回コールして処理が完結する場合は、マルチスレッド環境でも問題なく動作します。
Nmc_xxx の各関数同士は排他制御を行っています。

また、マルチスレッドと割り込みを併用するアプリケーションを開発する場合は、割込み発生のタイミングとスレッドからのボードへのアクセスタイミングが重ならないようにしてください。

3.5　プログラミング上の注意点

（１）入力信号フィルタの初期設定

ボードのリミット信号などの各入力信号は、MCX514内蔵の積分フィルタを使用します。ノヴァ電子より供給されるデバイスドライバでは、パソコン起動時の初期設定において、MCX514に入力信号フィルタモード設定コマンド（25h）を発行して各入力信号に対して、以下のようにフィルタを設定しています。

フィルタ遅延時間：512μsec

各信号のフィルタの有効/無効：

信号名	有効 / 無効
EMG	有効
nLMTP, nLMTM	有効
nSTOP0, nSTOP1	有効
nINPOS, nALARM	有効
nPIO6	有効
nSTOP2	有効
nECA, nECB	無効

アプリケーション上で、これらの入力信号フィルタの有効/無効を変更する場合には、MCX514の取扱説明書7.3.6節を参照してください。拡張モード設定コマンド（25h）によって変更することが出来ます。次の記述例では、ボード番号０の全ＩＣのX,Y,Z,U全軸に対して、上記の初期設定と同じ内容を設定しています。Nmc_FilterModeは拡張モード設定コマンド（25h）を実行しています。

（例１）
```
  Nmc_FilterMode(0, 0, AXIS_ALL, 0xAA7F);  // ＩＣ－Ａの設定
```

（例２）
```
  // ＩＣ－Ａの設定
  Nmc_WriteReg6(0, 0, 0xAA7F);
  Nmc_WriteReg0(0, 0, 0x0F25);
```

注意：
①コマンドリセット(00FF)を実行した場合にも、各入力信号のフィルタは上記のパソコン起動時初期設定と同じ設定がデバイスドライバ内で行なわれます。

（２）パソコンのスタンバイ、休止動作

本デバイスドライバは、パソコンのスタンバイ動作、あるいは休止動作を行った後のドライバの動作を保証していません。
パソコンのスタンバイ動作、あるいは休止動作を行った後に、ボードにアクセスしたい場合は、一度パソコンを再起動させてから行って下さい。

（３）割り込みサポートについて

割り込みは、ＶＣ＋＋で開発したアプリケーションでのみサポートしています。
本デバイスドライバでサポートしている割り込みの種類は下記の通りです。

・ＲＲ１レジスタで報告される割り込み全て
・連続補間ドライブ、ビットパターン補間でプリバッファのスタックカウンタが８→７、４→３に変わった時の割り込み

（４）割り込みクリアについて

①ＲＲ１レジスタで報告される割り込みについて

ボードでこの割込みが発生した直後、ドライバ内でＲＲ１を読み出し、割り込みをクリアしています。
その後、アプリケーションの割り込み用ユーザー関数が呼び出されます。（ユーザー関数を設定した場合のみ）

②連続補間ドライブ、ビットパターン補間でプリバッファのスタックカウンタが８→７、４→３に変わった時の割り込み

ボードでこの割込みが発生した直後、ドライバ内で補間割り込みをクリアしています。
その後、アプリケーションの割り込み用ユーザー関数が呼び出されます。（ユーザー関数を設定した場合のみ）

４．評価ツール

(1) MCX514評価ツール・・・・MCX514搭載ボード(MC8541P, MC8581P)を評価するツールです。(4.1参照)

4.1 MCX514評価ツール

MCX514 評価ツールプログラムは、MCX514を搭載しているボード(MC8541P, MC8581P)を評価するツールです。
弊社ホームページよりダウンロードすることができます。（MC8541P_8581Pデバイスドライバソフトに添付)
評価ツールを実行する前に、MC8541P_8581Pデバイスドライバをインストールして下さい。

注：この章では、MCX514 評価ツールの概要について説明します。詳細については、Tool¥MCX514 Boardフォルダの ReadMe.txt(操作説明書)を参照して下さい。

4.1.1 実行プログラムについて

実行プログラムはMCX514_Tool.exeです。
32bitOS用はTool¥MCX514 Board¥32フォルダに、64bitOS用はTool¥MCX514 Board¥64フォルダにあります。

■評価するMCX514について

各評価ツールが評価するMCX514は下記の通りです。

●MCX514_Tool.exe

①MCX514を１つ搭載しているボード(MC8541P)のMCX514
②MCX514を２つ搭載しているボード(MC8581P)のMCX514

■実行について

異なるボードに対して複数のMCX514_Tool.exeを同時に実行する事ができます。exeの名前を異なる名前に変更してから実行して下さい。(例：MCX514_Tool_A.exe、MCX514_Tool_B.exe など)
同じボードのICに対して、複数のMCX514_Tool.exeを同時に実行することはできません。

■起動時にエラーになる場合

起動時にエラーが発生して実行できない場合は、「3.2.2 サンプルプログラムおよび評価ツールの実行時にエラーが発生する場合の対応」を参照して対応を行ってください。

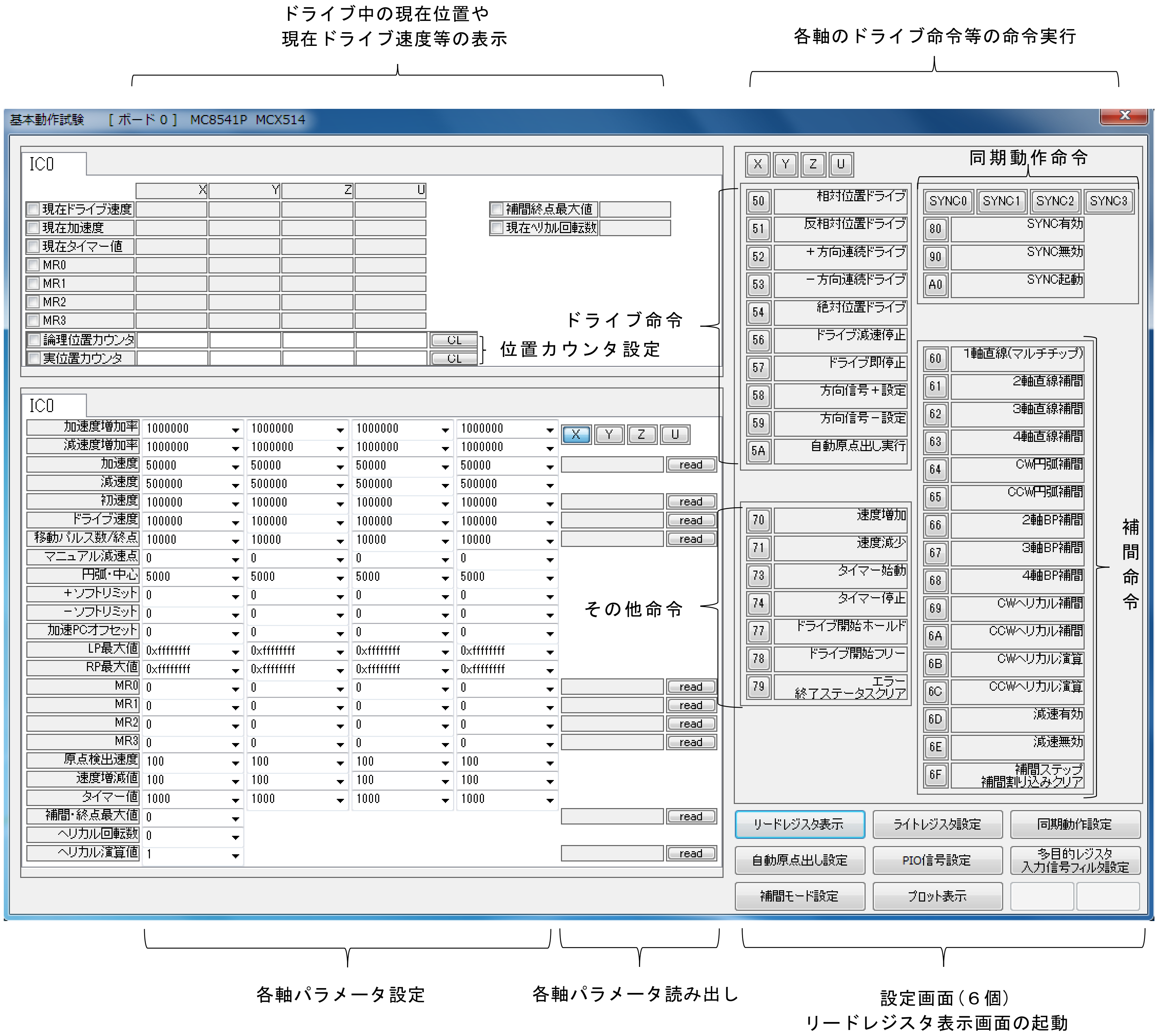
4.1.2 機能概要

評価ツールを起動すると、ボード番号選択画面が表示され、ボード番号（ボードのロータリースイッチ番号(0～F)）を選択することができます。ボード名表示ボタンを押すと、ボード番号の横にボード名(本ドライバを使用しているボードのみ)を表示することができます。ボード番号を選択後、基本動作試験ボタンを押すと、基本動作試験画面が表示されます。

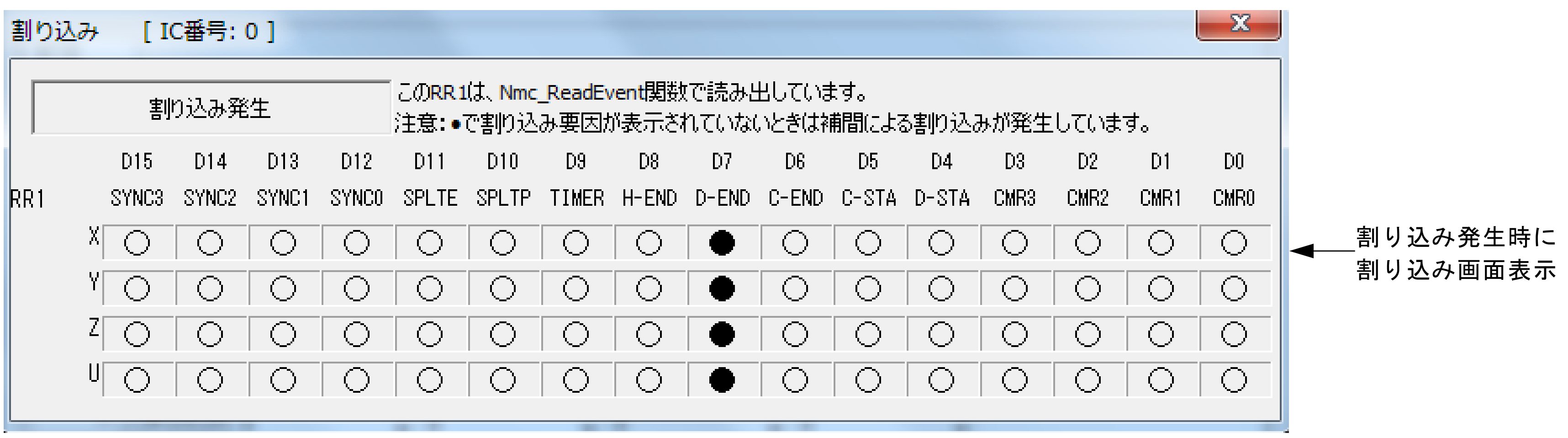
メイン画面では、各軸パラメータ設定および読み出し、ドライブ命令等の命令実行、現在位置・現在速度の表示、割り込み画面表示等を行います。また、モード設定画面(ライトレジスタ・同期動作・自動原点出し・ＰＩＯ・多目的レジスタ・入力信号フィルタ・補間モード設定画面)やリードレジスタ画面を開き、モード設定、リードレジスタ参照等を行う事ができます。

4.1.3 メイン画面

メイン画面では、下図のような操作を行います。

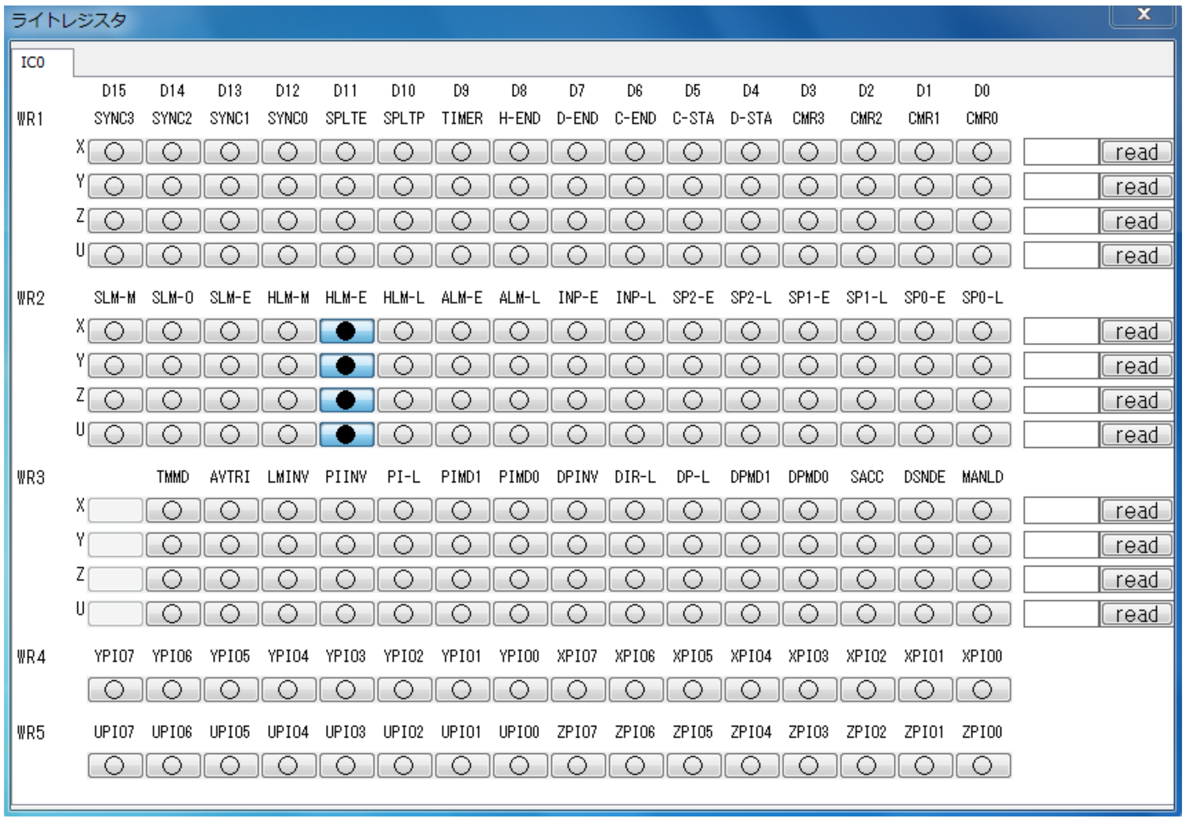


ドライブ中の現在位置や現在ドライブ速度などの表示は、
項目の左隣の□にチェックを入れると周期的に値を読み出して表示します。



4.1.4　ライトレジスタ設定画面

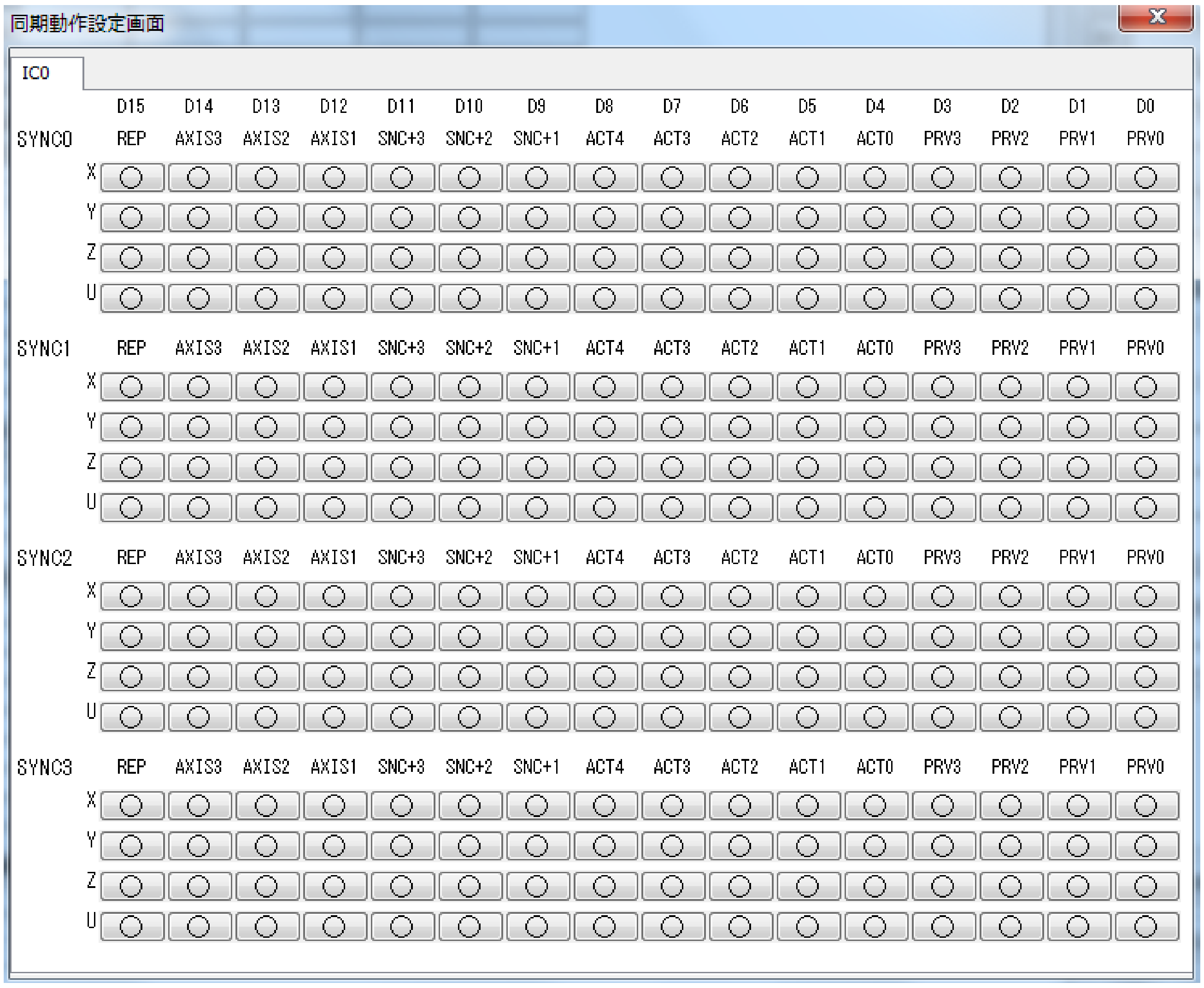
ライトレジスタ設定画面では、MCX514のWR1～WR5（モードレジスタ、アウトプットレジスタ）を設定、WR1～WR3（モードレジスタ）の読み出しをします。



WR4およびWR5は全てのボタンがチェックできるようになっていますが、
MC8541P/MC8581Pでは使用できるPIO信号が限定されています。ハードウエア取扱説明書を参考に正しく設定を行ってください。

4.1.5　同期動作設定画面

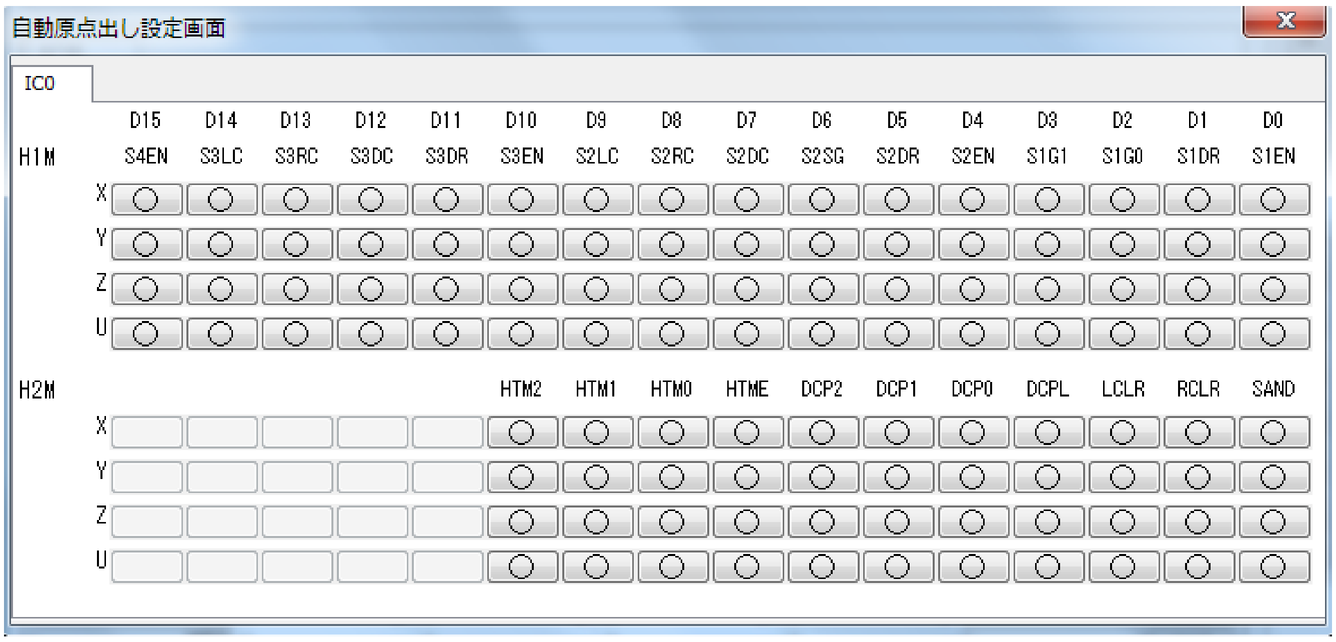
同期動作設定画面では、MCX514の同期動作モードレジスタ（SYNC0～SYNC3）を設定します。



MC8541P/MC8581Pでは使用できる同期動作の種類が限定されています。
ハードウエア取扱説明書を参考に正しく設定を行ってください。

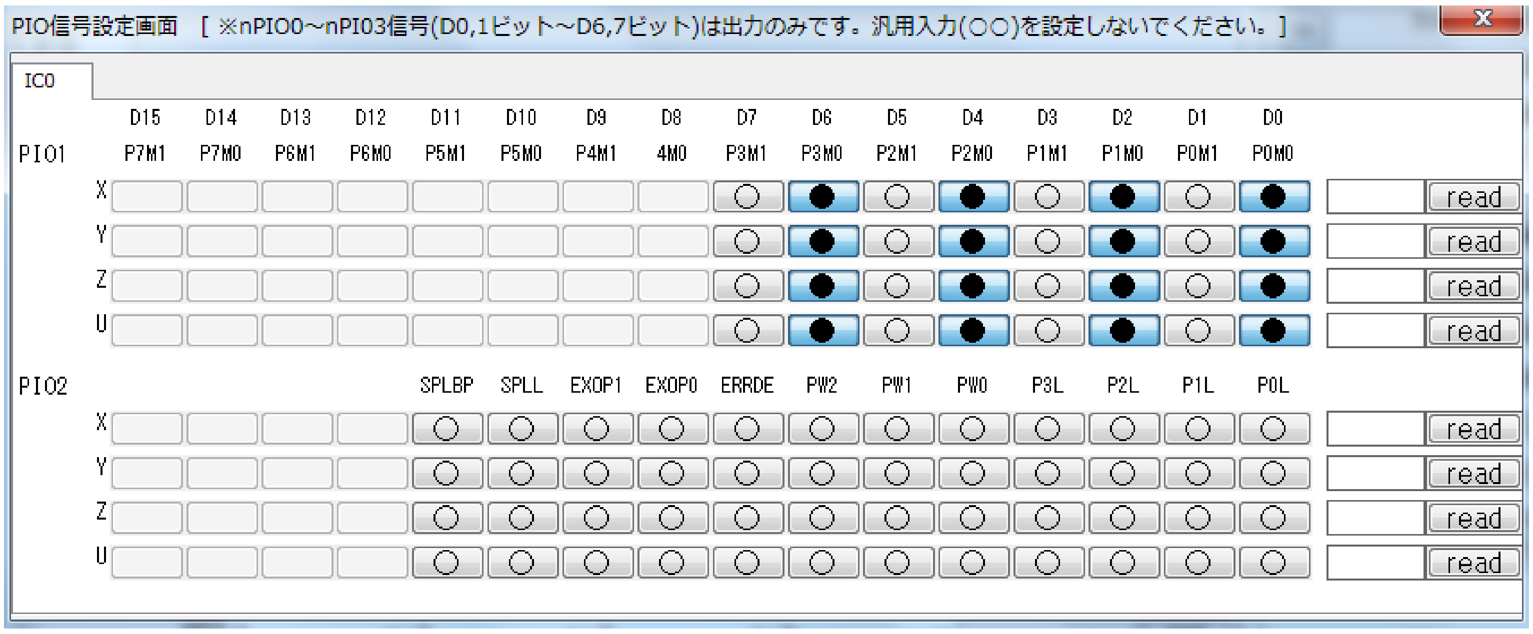
4.1.6　自動原点出し設定画面

自動原点出し設定画面では、MCX514の自動原点出しレジスタ（H1M,H2M)を設定します。



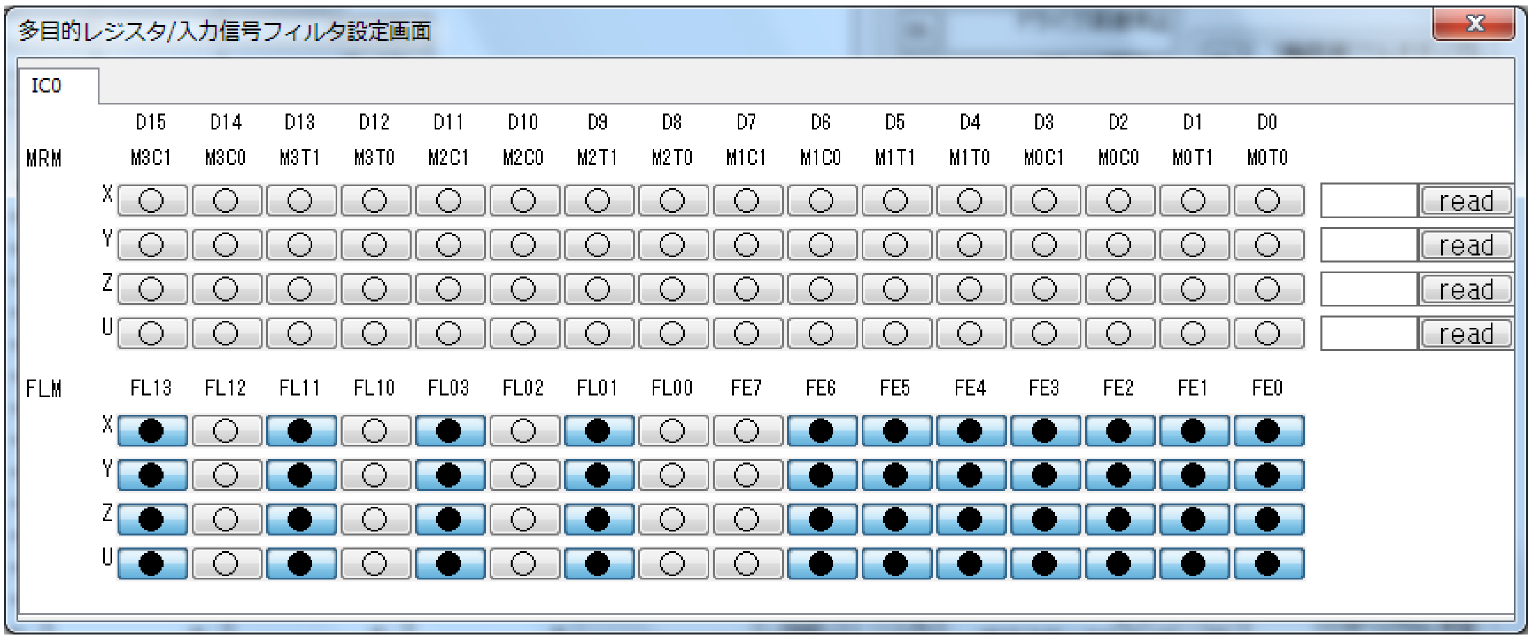
4.1.7　PIO信号設定画面

PIO信号設定画面では、MCX514のPIO（PIO1,PIO2）を設定、読み出しをします。



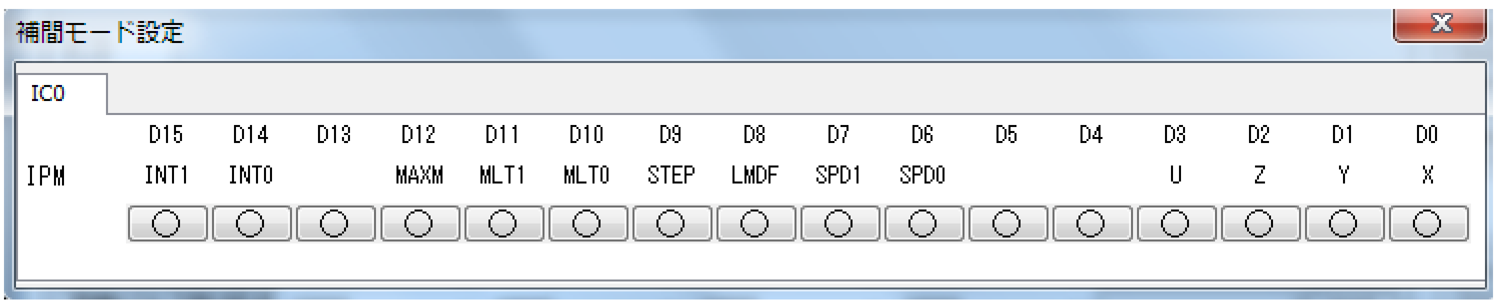
4.1.8　多目的レジスタ/入力信号フィルタ設定画面

多目的レジスタ/入力信号フィルタ設定画面では、MCX514の多目的レジスタ（MRM）の設定および読み出し、入力信号フィルタ設定（FLM）の設定をします。



4.1.9　補間モード設定画面

補間モード設定画面では、MCX514の補間モード（IPM）を設定します。



補間ドライブの評価が終了し、他の評価をおこなう際には、必ず全設定ボタンをクリアしてください。

4.1.10 リードレジスタ画面

リードレジスタ画面では、RR0,RR2,RR3,RR4,RR5（ステータスレジスタ、ＰＩＯリードレジスタ）の現在の値を一定間隔でMCX514から読み出し、画面に表示します。データの読み出し間隔は、50ミリ秒間隔です。
RR1 については、割り込み発生時、割り込み画面に表示します。