Melec

STEPPING & SERVO MOTOR CONTROLLER'S OPTION

MPL-36-02V2.00/USBW32
MPL-37-02V2.00/USBW64
取扱説明書 応用機能編 (設計者用)

(USBシリーズ用 Windowsデバイスドライバ)

MCC09ユニット編

USER'S MANUAL

既に本製品の別冊、取扱説明書を読まれていることを
前提に、より多彩な機能･仕様を解説した応用機能編です。
取扱説明書以上の内容についてはこの応用機能編を良く
読んで十分に理解してください。
この応用機能編は、いつでも取り出して読めるように
保管してください。

MN0323

はじめに

このデバイスドライバ「取扱説明書 応用機能編 」は、「USBシリーズのステッピングモータおよびサーボモータ用コントローラ」を正しく安全に使用していただくために、ステッピングモータ、あるいはサーボモータを使った制御装置の設計を担当される方を対象に、応用機能について説明しています。
応用機能を扱うときは各コントローラの「取扱説明書」、ならびにデバイスドライバの「取扱説明書」と同様に、本デバイスドライバ「取扱説明書 応用機能編 」を良く読んで十分に理解してください。
この「取扱説明書 応用機能編 」は、いつでも取り出して読めるように保管してください。

安全設計に関するお願い

■本資料に記載される技術情報は、製品の代表的動作・応用を説明するためのものであり、その使用に際して当社および第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。

■本資料に記載されている回路、ソフトウェア、およびこれらに関連する情報を使用する場合は、お客様の機器およびシステム全体で十分に評価し、お客様の責任において適用可否を判断してください。

■半導体ならびに半導体を使用した製品は、ある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。本製品の故障または誤動作により、人身事故、火災事故、社会的な損害などを生じさせないように、お客様の責任において、お客様の機器またはシステムに必要な安全設計を行うことをお願いします。

■本製品は、一般工業向けの汎用品として設計・製造されていますので、航空機器、航空宇宙機器、海底中継機器、原子力制御システム、輸送機器(車両、船舶等)、交通用信号機器、防災・防犯機器、安全装置、医療機器など、人命や財産に多大な影響が予想される用途には使用しないでください。

■本製品を改造、改変、複製等しないでください。

■輸出に際しては、「外国為替および外国貿易法」など適用される輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続きを行ってください。本製品または本資料に記載されている技術情報を、大量破壊兵器の開発等の目的、軍事利用の目的、その他軍事用途の目的で使用しないでください。
また、本製品を国内外の法令および規制により製造・使用・販売を禁止されている機器に使用することはできません。

■本製品の環境適合性などの詳細につきましては、必ず弊社営業窓口までお問い合わせください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令など適用される環境関連法令を十分調査の上、かかる法令に適合するようにご使用ください。
お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は一切その責任は負いません。

安全に関する事項の記述方法について

本製品は正しい方法で取り扱うことが大切です。
誤った方法で使用された場合、予期しない事故を引き起こし、人身への障害や財産の損壊などの被害を被るおそれがあります。
そのような事故の多くは、危険な状況を予め知っていれば回避することができます。
そのため、このデバイスドライバ「取扱説明書 応用機能編 」では危険な状況が予想できる場合には、注意事項が記述してあります。
それらの記述は、次のようなシンボルマークとシグナルワードで示しています。

⚠ 警告	取り扱いを誤った場合に死亡、または重傷を負うおそれのある警告事項を示します。
⚠ 注意	取り扱いを誤った場合に、軽傷を負うおそれや物的損害が発生するおそれがある注意事項を示します。

御使用の前に

■ USB シリーズコントローラは各軸を独立で制御できるため、各軸を以下のように呼称します。また、本書では、*1 の製品のことをコントローラドライバと呼称します。

製品名		1 軸目	2 軸目	3 軸目	4 軸目
UC-7660		X 軸	Y 軸	Z 軸	A 軸
UCD-7620/A5F31DE	*1	X 軸	Y 軸	ー	ー
UCD-7621/A5F41DE	*1	X 軸	Y 軸	ー	ー
UCD-7630/A5F31Q	*1	X 軸	Y 軸	Z 軸	A 軸

以降、原則として X 軸についてのみ説明します。

■ 入出力仕様ならびに接続に関する取り扱いについては、各コントローラの「取扱説明書」をご覧ください。

■ デバイスドライバを用いた基本的な仕様については、別冊デバイスドライバ「取扱説明書」をご覧ください。

はじめに
安全設計に関するお願い
安全に関する事項の記述方法について
御使用の前に

目　次

目　次

目　次

目　次

本版で改訂された主な箇所

1．概要

1-1. 特徴

● USB シリーズは、パソコンの小規模なシステムに最適な USB 通信によるステッピングモータ、サーボモータ、および I/O をコントロールするシステムです。

・パソコンを選ばずに、モーションおよび I/O コントロールのシステムが容易に構築できます。

・Windows 用デバイスドライバの関数仕様は、弊社製 PCI ボードコントローラ C-VX870 シリーズ(デバイス関数)、および AL-II シリーズ(デバイス関数とユニット関数)間で互いに移行が容易な仕様です。

● MPL-36-02v2.00/USBW32、および MPL-37-02v2.00/USBW64 は、Windows パソコン上の USB ポートから、弊社製 USB シリーズのステッピング＆サーボモータコントローラを動作させるための DLL ベースのドライバ関数です。
当デバイスドライバは、MCC09 搭載した新たなコントローラ UC-7660 に加え新たなスレーブコントローラドライバ UCD-7620/A5F31DE、UCD-7621/A5F41DE、UCD-7630/A5F31Q に対応したバージョンアップ品ですが、MCC07 製品にも対応しています。

・Windows 32 ビット対応版が MPL-36-02v2.00/USBW32 です。

・Windows 64 ビット対応版が MPL-37-02v2.00/USBW64 です。
MPL-37-02v2.00/USBW64 は、画像処理などの高速化を目的とした Windows 64 ビット環境のモーションおよび I/O システムを可能にします。

● 取扱説明書で解説している標準仕様の他に、この応用機能編では、主に以下を解説します。
・コマンド予約機能
・ドライブ CHANGE 機能
・外部トリガ出力機能
・新たなコントローラで対応された Motion Contorl Macro 機能(以降、本書では MCM 機能と呼称します。)
・その他、従来関数を継承できるようにするための関数など

以下についての詳細は、デバイスドライバ取扱説明書をご覧ください。
・サポート OS や言語他
・ソフト開発に必要なファイル
・デバイスドライバの制限事項
・標準的な関数、コマンド、機能の解説
・全体の実行シーケンス等

2．関数リファレンス

当デバイスドライバ仕様は、MCC09 および MCC07 の搭載製品ならびにデジタル I/O やアナログ入力を含む関数を網羅しています。

ユニットに MCC09 および MCC07 搭載製品が混在したシステム構成も可能です。
原則、関数仕様は共通性がありますので、搭載チップ毎のスレーブユニットによって、ユーザアプリケーションを分ける必要はありません。

なお、MCC09 と MCC07 搭載の製品の差異を明らかにできるように、本書では MCC09 搭載製品を対象とした製品の仕様を示します。

2-1. MCC PORT アクセス関数の移行について

本書で説明するデバイス関数は、主に従来より対応している関数群です。
これらの関数は、下記に推奨している標準関数で代用することが可能です。
下記の従来関数から推奨関数に移行することで、特に、マルチスレッドプログラミングにおける、ユーザアプリケーション側での排他処理を不要とすることができます。

従来関数群
・DRIVE COMMAND データ構造体書き込み関数(MC07_IWDrive)
・DRIVE DATA データ構造体書き込み関数(MC07_IWData)
・DRIVE DATA 32 ビット書き込み関数(MC07_LWData)
・DRIVE DATA1 PORT 書き込み関数(MC07_BWDriveData1)
・DRIVE DATA2 PORT 書き込み関数(MC07_BWDriveData2)

⇩

推奨関数
・DRIVE COMMAND 32 ビット書き込み関数(MC07_LWDrive)

従来関数群
・DRIVE DATA データ構造体読み出し関数(MC07_IRDrive)
・DRIVE DATA 32 ビット読み出し関数(MC07_LRDrive)
・DRIVE DATA1 PORT 読み出し関数 (MC07_BRDriveData1)
・DRIVE DATA2 PORT 読み出し関数 (MC07_BRDriveData2)

⇩

推奨関数	
・DRIVE COMMAND 32 ビット書き込み関数／読み出し関数(MC07_LWRDrive)	
・DRIVE COMMAND PORT 書き込み関数／読み出し関数(MC07_BWRDrive)	*1

*1 MCC09 搭載製品のみ対応している関数です。

2-2. ユニット関数

2-2-1. A/D 変換関数

ユニットオープン関数で取得したユニットハンドルにより、ユニットのアナログ入力を制御します。
A/D 変換されたアナログ入力信号の読み出しを行います。

A/Dデータ構造体

UsbC

UC-7660

機　能

全てのアナログ入力信号をA/D変換したデータを格納します。

書　式

C言語
```
typedef struct _MCO7_S_AD_DATA {
    WORD Data[16];
} MCO7_S_AD_DATA;
```

VB
```
Type MCO7_S_AD_DATA
    Data(1 To 16) As Integer
End Type
```

VB.NET
```
Structure MCO7_S_AD_DATA
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=16)> Public Data() As Short
    Public Sub Initialize()
        ReDim Data(15)
    End Sub
End Structure
```

C#.NET
```
struct MCO7_S_AD_DATA
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
    public ushort[] Data;
    public MCO7_S_AD_DATA(ushort dummy)
    {
        Data = new ushort[16];
    }
 }
```

メンバ

Data[0] ･･･ AD0信号のデータが格納されます。
　〜
Data[3] ･･･ AD3信号のデータが格納されます。
Data[4]〜*Data[15]* ･･･ 将来の拡張用です。

・VBの*Data(1)*〜*(16)*は、C言語の*Data [0] 〜 [15]* に対応します。
・VB.NETの*Data(0)*〜*(15)*は、C言語の*Data [0] 〜 [15]* に対応します。
・C#.NETの*Data [0] 〜 [15]* は、C言語の*Data [0] 〜 [15]* に対応します。

ユニット	範囲
UC-766	0〜1023 *1

*1 変換データ

入力電圧値 ＝ 変換データ×入力レンジ÷分解能

入力電圧	AD変換データ(10ビット)
4.995V	1023 (H'3FF)
4.990V	1022 (H'3FE)
〜	〜
0.00488V	1 (H'001)
0V	0 (H'000)

例） 読み出されたAD変換データがH'1FFのとき
入力電圧 ＝ H'1FF×5V÷1024
＝ 511×5V÷1024
＝ 2.441V

A/D DATA読み出し関数

UsbC

UC-7660

機　能

指定されたアナログ入力信号をA/D変換し、データを読み出します。

書　式

C言語　BOOL MCO7_BRADData(DWORD *hUnit*, WORD *Signal*, WORD **pData*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_BRADData(ByVal *hUnit* As Long, ByVal *Signal* As Integer,
ByRef *pData* As Integer, ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_BRADData(ByVal *hUnit* As Integer, ByVal *Signal* As Short,
ByRef *pData* As Short, ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.BRADData(uint *hUnit*, ushort *Signal*, ref ushort *pData*, ref MCO7_S_RESULT *psResult*);

引　数

hUnit　・・・ ユニットハンドルを指定します。
Signal　・・・ アナログ入力信号を指定します。

指定できる値	意味
0	AD0信号
1	AD1信号
2	AD2信号
3	AD3信号

pData　・・・ 読み出した内容が格納される変数のポインタを指定します。

ユニット	範囲
UC-766	0～1023 *1

*1 変換データ

入力電圧値 ＝ 変換データ×入力レンジ÷分解能

psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

A/D DATA A/Dデータ構造体読み出し関数

UsbC

UC-7660

機　能

全てのアナログ入力信号をA/D変換し、データを読み出します。

書　式

C言語　BOOL MCO7_IRADData(DWORD *hUnit*, MCO7_S_AD_DATA **psData*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_IRADData(ByVal *hUnit* As Long, ByRef *psData* As MCO7_S_AD_DATA,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_IRADData(ByVal *hUnit* As Integer, ByRef *psData* As MCO7_S_AD_DATA,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.IRADData(uint *hUnit*, ref MCO7_S_AD_DATA *psData*, ref MCO7_S_RESULT *psResult*);

引　数

hUnit　・・・ ユニットハンドルを指定します。
psData　・・・ A/Dデータ構造体のポインタを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

2-3. デバイス関数

2-3-1. MCC PORT アクセス関数

DRIVE DATA 32ビット書き込み関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORTにデータを書き込みます。

書　式

C言語　BOOL MCO7_LWData(DWORD *hDev*, DWORD **pData*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_LWData(ByVal *hDev* As Long, ByRef *pData* As Long,
　　ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_LWData(ByVal *hDev* As Integer, ByRef *pData* As Integer,
　　ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.LWData(uint *hDev*, ref uint *pData*, ref MCO7_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
pData　・・・ 書き込むデータが格納されている変数のポインタを指定します。
　・変数の上位16ビットは、DRIVE DATA2 PORTに書き込まれます。
　・変数の下位16ビットは、DRIVE DATA1 PORTに書き込まれます。
psResult　・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

DRIVE DATA1 PORT書き込み関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたデバイスのDRIVE DATA1 PORTにデータを書き込みます。

書　式

C言語　BOOL MCO7_BWDriveData1(DWORD *hDev*, WORD **pData*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_BWDriveData1(ByVal *hDev* As Long, ByRef *pData* As Integer,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_BWDriveData1(ByVal *hDev* As Integer, ByRef *pData* As Short,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.BWDriveData1(uint *hDev*, ref ushort *pData*, ref MCO7_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
pData　・・・ 書き込むデータが格納されている変数のポインタを指定します。
psResult　・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

DRIVE DATA2 PORT書き込み関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたデバイスのDRIVE DATA2 PORTにデータを書き込みます。

書　式

C言語　BOOL MCO7_BWDriveData2(DWORD *hDev*, WORD **pData*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_BWDriveData2(ByVal *hDev* As Long, ByRef *pData* As Integer,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_BWDriveData2(ByVal *hDev* As Integer, ByRef *pData* As Short,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.BWDriveData2(uint *hDev*, ref ushort *pData*, ref MCO7_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
pData　・・・ 書き込むデータが格納されている変数のポインタを指定します。
psResult　・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

DRIVE DATA 32ビット読み出し関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORTの内容を読み出します。

書　式

C言語　BOOL MCO7_LRDrive(DWORD *hDev*, DWORD **pData*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_LRDrive(ByVal *hDev* As Long, ByRef *pData* As Long,
ByRef *psResult* As MCO7_S_RESULT)As Boolean

VB.NET　Function MCO7_LRDrive(ByVal *hDev* As Integer, ByRef *pData* As Integer,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.LRDrive(uint *hDev*, ref uint *pData*, ref MCO7_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
pData　・・・ 読み出した内容を格納するための変数のポインタを指定します。
・DRIVE DATA2 PORTの内容が変数の上位16ビットに格納されます。
・DRIVE DATA1 PORTの内容が変数の下位16ビットに格納されます。
psResult　・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

DRIVE DATA1 PORT読み出し関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたデバイスのDRIVE DATA1 PORTの内容を読み出します。

書　式

C言語　BOOL MCO7_BRDriveData1(DWORD *hDev*, WORD **pData*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_BRDriveData1(ByVal *hDev* As Long, ByRef *pData* As Integer,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_BRDriveData1(ByVal *hDev* As Integer, ByRef *pData* As Short,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.BRDriveData1(uint *hDev*, ref ushort *pData*, ref MCO7_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
pData　・・・ 読み出した内容を格納するための変数のポインタを指定します。
psResult　・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

DRIVE DATA2 PORT読み出し関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたデバイスのDRIVE DATA2 PORTの内容を読み出します。

書　式

C言語　BOOL MCO7_BRDriveData2(DWORD *hDev*, WORD **pData*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_BRDriveData2(ByVal *hDev* As Long, ByRef *pData* As Integer,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_BRDriveData2(ByVal *hDev* As Integer, ByRef *pData* As Short,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.BRDriveData2(uint *hDev*, ref ushort *pData*, ref MCO7_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
pData　・・・ 読み出した内容を格納するための変数のポインタを指定します。
psResult　・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

データ構造体

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

DRIVE DATA1 PORT、DRIVE DATA2 PORTに書き込むデータを格納します。

書 式

C言語
```
typedef struct _MCO7_S_DATA{
      WORD MCO7_Data[4];
} MCO7_S_DATA
```

VB
```
Type MCO7_S_DATA
    MCO7_Data(1 To 4) As Integer
End Type
```

VB.NET
```
Structure MCO7_S_DATA
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> Public MCO7_Data() As Short
    Public Sub Initialize()
        ReDim MCO7_Data(3)
    End Sub
End Structure
```

C#.NET
```
struct MCO7_S_DATA
{
    [MarshalAs( UnmanagedType.ByValArray, SizeConst=4 )] public ushort[] MCO7_Data;
    MCO7_S_DATA( ushort dummy )
    {
        MCO7_Data = new ushort[4];
    }
}
```

メンバ

次に示すメンバは、C言語で表記しています。C言語のMCO7_Data[0]～MCO7_Data[3]は、Visual Basicでは
MCO7_Data(1)～MCO7_Data(4)、Visual Basic.NETではMCO7_Data(0)～MCO7_Data(3)、
C#.NETではMCO7_Data[0]～MCO7_Data[3]に対応します。

MCO7_Data[0] ･･･ DRIVE DATA1 PORTの内容を格納します。
MCO7_Data[1] ･･･ DRIVE DATA2 PORTの内容を格納します。
MCO7_Data[2] ･･･ 将来の拡張用です。
MCO7_Data[3] ･･･ 将来の拡張用です。

*VB.NETでは、配列を含む構造体は、Initializeメソッドにより配列を作成します。
*C#.NETでは、配列を含む構造体は、コンストラクタにより配列を作成します。
 コンストラクタの引数dummyは何を指定しても無効です。

DRIVE COMMAND データ構造体書き込み関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORTにデータ構造体の内容を書き込んだ後、
DRIVE COMMAND PORTにコマンドコードを書き込みます。

書 式

C言語 BOOL MCO7_IWDrive(DWORD *hDev*, WORD *Cmd*, MCO7_S_DATA **psData*, MCO7_S_RESULT **psResult*);

VB Function MCO7_IWDrive(ByVal *hDev* As Long, ByVal *Cmd* As Integer, ByRef *psData* As MCO7_S_DATA,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET Function MCO7_IWDrive(ByVal *hDev* As Integer, ByVal *Cmd* As Short, ByRef *psData* As MCO7_S_DATA,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET bool MCO7.IWDrive(uint *hDev*, ushort *Cmd*, ref MCO7_S_DATA *psData*,
ref MCO7_S_RESULT *psResult*);

引 数

hDev ・・・ デバイスハンドルを指定します。
Cmd ・・・ 書き込むコマンドコードを指定します。
psData ・・・ 書き込むデータが格納されているデータ構造体のポインタを指定します。
psResult ・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

DRIVE DATA データ構造体書き込み関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORTにデータを書き込みます。

書　式

C言語　BOOL MCO7_IWData(DWORD *hDev*, MCO7_S_DATA **psData*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_IWData(ByVal *hDev* As Long, ByRef *psData* As MCO7_S_DATA,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_IWData(ByVal *hDev* As Integer, ByRef *psData* As MCO7_S_DATA,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.IWData(uint *hDev*, ref MCO7_S_DATA *psData*,
ref MCO7_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
psData　・・・ 書き込むデータが格納されているデータ構造体のポインタを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

DRIVE DATA データ構造体読み出し関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORTの内容を読み出します。

書　式

C言語　BOOL MCO7_IRDrive(DWORD *hDev*, MCO7_S_DATA **psData*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_IRDrive(ByVal *hDev* As Long, ByRef *psData* As MCO7_S_DATA,
ByRef *psResult* As MCO7_S_RESULT)As Boolean

VB.NET　Function MCO7_IRDrive(ByVal *hDev* As Integer, ByRef *psData* As MCO7_S_DATA,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.IRDrive(uint *hDev*, ref MCO7_S_DATA *psData*, ref MCO7_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
psData　・・・ 読み出した内容を格納するためのデータ構造体のポインタを指定します。
psResult　・・・ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

データセット関数

UsbC

―

機　能

32ビットデータを次の形式でデータ構造体に格納します。

引数psDataで示されるデータ構造体のメンバ*MCO7_Data[0]* (C言語表記)[各種DATA1 PORTに対応]

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D15	←					引数 Data の下位 16 ビット(D15 ～ D0)								→	D0

引数psDataで示されるデータ構造体のメンバ*MCO7_Data[0]* (C言語表記)[各種DATA2 PORTに対応]

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D31	←					引数 Data の上位 16 ビット(D31 ～ D16)								→	D16

書　式

C言語　VOID MCO7_SetData(DWORD *Data*, MCO7_S_DATA **psData*);

VB　Sub MCO7_SetData(ByVal *Data* As Long, ByRef *psData* As MCO7_S_DATA)

VB.NET　Sub MCO7_SetData(ByVal *Data* As Integer, ByRef *psData* As MCO7_S_DATA)

C#.NET　void MCO7.SetData(uint *Data*, ref MCO7_S_DATA *psData*);

引　数

Data　･･･ 32ビットのデータを指定します。
psData　･･･ データ構造体のポインタを指定します。

戻り値

この関数に、戻り値はありません。

データゲット関数

UsbC

―

機　能

データ構造体の内容を、次の形式で32ビットデータに変換し返します。

変換後の32ビットデータ

D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
D15	←	引数psDataで示されるデータ構造体のメンバMCO7_Data[1](C言語表記) [各種DATA2 PORTに対応]												→	D0

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D15	←	引数psDataで示されるデータ構造体のメンバMCO7_Data[0](C言語表記) [各種DATA1 PORTに対応]												→	D0

書　式

C言語　DWORD MCO7_GetData(MCO7_S_DATA *psData);

VB　Function MCO7_GetData(ByRef *psData* As MCO7_S_DATA) As Long

VB.NET　Function MCO7_GetData(ByRef *psData* As MCO7_S_DATA) As Integer

C#.NET　Uint MCO7.GetData(ref MCO7_S_DATA *psData*);

引　数

psData　・・・ データ構造体のポインタを指定します。

戻り値

32ビットに変換されたデータを返します。

2-3-2. コマンド予約機能関数

MCC09 の汎用コマンドは、コマンド予約機能によりコマンドを予約レジスタに書き込むことができます。
コマンド予約機能で汎用コマンドを書き込むときは、DRIVE STATUS1 PORT 内の ERROR=0 および COMREG FL BIT=0 を確認してから汎用コマンドを書き込みます。
また、汎用コマンドをコマンド予約機能で書き込みした後に、予約コマンドの空き待ちを行います。
COMREG NOT FULL WAIT 関数は、この汎用コマンドを予約コマンドとして書き込みした後の終了待ちするときに用いる関数です。

COMREG NOT FULL WAIT関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたデバイスがコマンドレジスタ予約可能な状態(DRIVE STATUS1 PORT COMREG FULL BIT = 0) になるまで待機します。
・最大待ち時間を超えるとエラー終了します。
・待機中にDRIVE STATUS1 PORTのERROR = 1 を検出したときもエラー終了します。

書　式

C言語　BOOL MCO7_BWaitComregNotFull(DWORD *hDev*, WORD *WaitTime*, MCO7_S_RESULT **psResult*);

VB　Function MCO7_BWaitComregNotFull(ByVal *hDev* As Long, ByVal *WaitTime* As Integer,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_BWaitComregNotFull(ByVal *hDev* As Integer, ByVal *WaitTime* As Short,
ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.BWaitComregNotFull(uint *hDev*, ushort *WaitTime*, ref MCO7_S_RESULT *psResult*);

引　数

hDev　・・・ デバイスハンドルを指定します。
WaitTime　・・・ 最大待ち時間を1ms単位で設定します。
0を指定すると無限に待機します。
psResult　・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

DRIVE COMMAND バッファー書き込み関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

コマンドデータ構造体の配列に格納されているコマンド、データを指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE COMMAND PORTに指定された個数書き込みます。

書 式

C言語
```
BOOL MCO7_LWDriveBuf(DWORD hDev, MCO7_S_COMMAND_DATA  CmdDataBuf[], WORD Count,
        MCO7_S_RESULT *psResult);
```

VB
```
Function MCO7_LWDriveBuf(ByVal hDev As Long, ByRef CmdDataBuf As MCO7_S_COMMAND_DATA,
        ByVal Count As Integer, ByRef psResult As MCO7_S_RESULT) As Boolean
```

VB.NET
```
Function MCO7_LWDriveBuf(ByVal hDev As Integer, ByVal CmdDataBuf() As MCO7_S_COMMAND_DATA,
        ByVal Count As Short, ByRef psResult As MCO7_S_RESULT) As Boolean
```

C#.NET
```
bool MCO7.LWDriveBuf(uint hDev, MCO7_S_COMMAND_DATA[] CmdDataBuf, ushort Count,
        ref MCO7_S_RESULT psResult);
```

引 数

hDev ･･･ デバイスハンドルを指定します。
CmdDataBuf ･･･ コマンドデータ構造体の配列に指定します。
CmdDataBuf[0] ･･･ 1番目に書き込むコマンドコード構造体を格納します。
･･･
CmdDataBuf[7] ･･･ 8番目に書き込むコマンドコード構造体を格納します。
CmdDataBuf[8]～CmdDataBuf[15] ･･･ 将来の拡張用です。
Count ･･･ コマンドコード構造体の配列に格納されている要素の個数(最大8個)を指定します。
psResult ･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

2-3-3. 円弧補間の演算関数

円弧補間短軸PULSE数ゲット関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定された円弧の中心点、目的地、回転方向から、目的地までの短軸パルス数を計算します。

座標は、モータの現在位置を座標中心(0,0)とした相対アドレスです。
計算方法については、「4-1-5. 補間ドライブ (4) 円弧補間ドライブ」をご覧ください。
注. 次の場合、関数がエラー終了します。
- 円弧の中心点座標が(0, 0)、または中心点と目的地が同一座標の場合
- 円弧補間で求めた短軸PULSE数が-2,147,483,648～+2,147,483,647の範囲内でない場合
- 円弧補間で指定出来ない目的地座標が指定された場合

書　式

C言語
```
BOOL MCO7_GetCirShortPulse( WORD Dir, MCO7_S_XY_POSITION *psCenterPosition,
        MCO7_S_XY_POSITION *psTargetPosition,
        LONG *pShortPulse, MCO7_S_RESULT *psResult );
```

VB
```
Function MCO7_GetCirShortPulse(ByVal Dir As Integer,
        ByRef psCenterPosition As MCO7_S_XY_POSITION,
        ByRef psTargetPosition As MCO7_S_XY_POSITION, ByRef pShortPulse As Long,
        ByRef psResult As MCO7_S_RESULT) As Boolean
```

VB.NET
```
Function MCO7_GetCirShortPulse(ByVal Dir As Short,
        ByRef psCenterPosition As MCO7_S_XY_POSITION,
        ByRef psTargetPosition As MCO7_S_XY_POSITION, ByRef pShortPulse As Integer,
        ByRef psResult As MCO7_S_RESULT) As Boolean
```

C#.NET
```
bool MCO7.GetCirShortPulse(ushort Dir, ref MCO7_S_XY_POSITION psCenterPosition,
        ref MCO7_S_XY_POSITION psTargetPosition, ref int pShortPulse,
        ref MCO7_S_RESULT psResult);
```

引　数

Dir　　･･･　回転方向を指定します。

指定できる値	意味
MCO7_CCW	-(CCW)方向
MCO7_CW	+(CW)方向

psCenterPosition　･･･ 中心点のX-Y相対座標が格納されているポジション構造体のポインタを指定します。
(指定できる値の範囲はX座標･Y座標ともに、-8,388,607～+8,388,607です。)
中心点のX-Y座標は、現在位置を座標の中心(0,0)とした相対座標です。

psTargetPosition　･･･ 目的地のX-Y相対座標が格納されているポジション構造体のポインタを指定します。
(指定できる値の範囲はX座標･Y座標ともに、-16,777,214～+16,777,214です。)
目的地のX-Y座標は、現在位置を座標の中心(0,0)とした相対座標です。

pShortPulse　　･･･ 目的地の短軸パルス数を格納するための変数のポインタを指定します。

psResult　　･･･ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

2-3-4. その他の関数

16ビット符号なし変換関数

UsbC

－

機　能

指定された16ビット符号付きデータを16ビット符号なしデータに変換します。

書　式

C言語　WORD MCO7_Unsigned16(SHORT *Data*);

VB　－

VB.NET　Function MCO7_Unsigned16(ByVal *Data* As Short) As UShort

C#.NET　ushort MCO7.Unsigned16(short *Data*);

引　数

Data　・・・ 16ビット符号付きデータを指定します。

戻り値

16ビット符号なしデータを返します。

16ビット符号付き変換関数

UsbC

－

機　能

指定された16ビット符号なしデータを16ビット符号つきデータに変換します。

書　式

C言語　SHORT MCO7_Signed16(WORD *Data*);

VB　－

VB.NET　Function MCO7_Signed16(ByVal *Data* As UShort) As Short

C#.NET　short MCO7.Signed16(ushort *Data*);

引　数

Data　･･･ 16ビット符号なしデータを指定します。

戻り値

16ビット符号付きデータを返します。

32ビット符号なし変換関数

UsbC

－

機　能

指定された32ビット符号付きデータを32ビット符号なしデータに変換します。

書　式

C言語　DWORD MCO7_Unsigned32(LONG *Data*);

VB　－

VB.NET　Function MCO7_Unsigned32(ByVal *Data* As Integer) As UInteger

C#.NET　uint MCO7.Unsigned32(int *Data*);

引　数

Data　･･･ 32ビット符号付きデータを指定します。

戻り値

32ビット符号なしデータを返します。

32ビット符号付き変換関数

UsbC

－

機　能

指定された32ビット符号なしデータを32ビット符号つきデータに変換します。

書　式

C言語　LONG MCO7_Signed32(DWORD *Data*);

VB　－

VB.NET　Function MCO7_Signed32(ByVal *Data* As UInteger) As Integer

C#.NET　int MCO7.Signed32(uint *Data*);

引　数

Data　･･･ 32ビット符号なしデータを指定します。

戻り値

32ビット符号付きデータを返します。

2-4. ユニット MCM 関数

MCM の設定・制御を行います。

2-4-1. MCM 情報の表示

MCM の情報を読み出します。

AXIS MCM INFO構造体 UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

軸のMCMの情報が格納されます。

書　式

C言語
```
typedef struct _MCO7_S_AXIS_MCM_INFO {
    DWORD Address;
} MCO7_S_AXIS_MCM_INFO;
```

VB
```
Type MCO7_S_AXIS_MCM_INFO
    Address As Long
End Type
```

VB.NET
```
Structure MCO7_S_AXIS_MCM_INFO
    Public Address As Integer
End Structure
```

C#.NET
```
struct MCO7_S_AXIS_MCM_INFO
{
    public uint Address;
}
```

メンバ

Address　　･･･ RAM領域の次に実行するメモリアドレスが格納されます。

UNIT MCM INFO構造体

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

ユニットのMCMの情報が格納されます。

書　式

C言語
```
typedef struct _MCO7_S_UNIT_MCM_INFO {
    DWORD SheetNo;
    MCO7_S_AXIS_MCM_INFO X;
    MCO7_S_AXIS_MCM_INFO Y;
    MCO7_S_AXIS_MCM_INFO Z;
    MCO7_S_AXIS_MCM_INFO A;
} MCO7_S_UNIT_MCM_INFO;
```

VB
```
Type MCO7_S_UNIT_MCM_INFO
    SheetNo As Integer
    X As MCO7_S_AXIS_MCM_INFO
    Y As MCO7_S_AXIS_MCM_INFO
    Z As MCO7_S_AXIS_MCM_INFO
    A As MCO7_S_AXIS_MCM_INFO
End Type
```

VB.NET
```
Structure MCO7_S_UNIT_MCM_INFO
    Public SheetNo As Short
    Public X As MCO7_S_AXIS_INFO
    Public Y As MCO7_S_AXIS_INFO
    Public Z As MCO7_S_AXIS_INFO
    Public A As MCO7_S_AXIS_INFO
End Structure
```

C#.NET
```
struct MCO7_S_UNIT_MCM_INFO
{
    public ushort SheetNo;
    public MCO7_S_AXIS_MCM_INFO X;
    public MCO7_S_AXIS_MCM_INFO Y;
    public MCO7_S_AXIS_MCM_INFO Z;
    public MCO7_S_AXIS_MCM_INFO A;
}
```

メンバ

SheetNo　　･･･ UNIT MCM SHEET DOWNLOAD関数のダウンロード元のシート番号が格納されます。
X　　･･･ X軸のAXIS MCM INFO構造体の内容が格納されます。
Y　　･･･ Y軸のAXIS MCM INFO構造体の内容が格納されます。
Z　　･･･ Z軸のAXIS MCM INFO構造体の内容が格納されます。
A　　･･･ A軸のAXIS MCM INFO構造体の内容が格納されます。

・ユニットの電源投入後、未だ一度もシートをダウンロードしていない場合、
　メンバ*SheetNo* には、H'00FFが格納されます。

UNIT MCM INFO READ関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

指定されたユニットのMCMの情報を読み出します。

書 式

C言語
```
BOOL MCO7_UReadMcmInfo( DWORD hUnit, DWORD AxisSel,
        MCO7_S_UNIT_MCM_INFO *psUnitMcmInfo,
        MCO7_S_RESULT *psResult );
```

VB
```
Function MCO7_UReadMcmInfo( ByVal hUnit As Long, ByVal AxisSel As Long,
        ByRef psUnitMcmInfo As MCO7_S_UNIT_MCM_INFO,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

VB.NET
```
Function MCO7_UReadMcmInfo( ByVal hUnit As Integer, ByVal AxisSel As Integer,
        ByRef psUnitMcmInfo As MCO7_S_UNIT_MCM_INFO,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

C#.NET
```
bool MCO7.UReadMcmInfo( uint hUnit, uint AxisSel,
        ref MCO7_S_UNIT_MCM_INFO psUnitMcmInfo,
        ref MCO7_S_RESULT psResult );
```

引 数

hUnit ・・・ ユニットハンドルを指定します。
AxisSel ・・・ MCO7_SEL_X_Y_Z_Aを指定します。
psUnitMcmInfo ・・・ MCMの情報を格納するためのUNIT MCM INFO構造体のポインタを指定します。
psResult ・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

2-4-2. MCM SPEC の設定と読み出し

MCM の SPEC 設定と読み出しを行います。
設定する内容については、UNIT MCM SPEC SET1, 2, 3 関数を参照してください。

AXIS MCM SPEC構造体

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

軸の基本設定を格納します。

書 式

C言語
```
typedef struct _MCO7_S_AXIS_MCM_SPEC {
    WORD Spec;
} MCO7_S_AXIS_MCM_SPEC;
```

VB
```
Type MCO7_S_AXIS_MCM_SPEC
    Spec As Integer
End Type
```

VB.NET
```
Structure MCO7_S_AXIS_MCM_SPEC
    Public Spec As Short
End Structure
```

C#.NET
```
struct MCO7_S_AXIS_MCM_SPEC
{
    public ushort Spec;
}
```

メンバ

Spec ･･･ MCM SPEC1，2，3の設定データを格納します。

UNIT MCM SPEC構造体

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

ユニットの基本設定を格納します。

書　式

C言語
```
typedef struct _MCO7_S_UNIT_MCM_SPEC {
    MCO7_S_AXIS_MCM_SPEC X;
    MCO7_S_AXIS_MCM_SPEC Y;
    MCO7_S_AXIS_MCM_SPEC Z;
    MCO7_S_AXIS_MCM_SPEC A;
} MCO7_S_UNIT_MCM_SPEC;
```

VB
```
Type MCO7_S_UNIT_MCM_SPEC
    X As MCO7_S_AXIS_MCM_SPEC
    Y As MCO7_S_AXIS_MCM_SPEC
    Z As MCO7_S_AXIS_MCM_SPEC
    A As MCO7_S_AXIS_MCM_SPEC
End Type
```

VB.NET
```
Structure MCO7_S_UNIT_MCM_SPEC
    Public X As MCO7_S_AXIS_SPEC
    Public Y As MCO7_S_AXIS_SPEC
    Public Z As MCO7_S_AXIS_SPEC
    Public A As MCO7_S_AXIS_SPEC
End Structure
```

C#.NET
```
struct MCO7_S_UNIT_MCM_SPEC
{
    public MCO7_S_AXIS_MCM_SPEC X;
    public MCO7_S_AXIS_MCM_SPEC Y;
    public MCO7_S_AXIS_MCM_SPEC Z;
    public MCO7_S_AXIS_MCM_SPEC A;
}
```

メンバ

X　　･･･ X軸(または$\overline{\text{SOUT0}}$信号)のAXIS MCM SPEC構造体の内容を格納します。
Y　　･･･ Y軸(または$\overline{\text{SOUT1}}$信号)のAXIS MCM SPEC構造体の内容を格納します。
Z　　･･･ Z軸(または$\overline{\text{SOUT2}}$信号)のAXIS MCM SPEC構造体の内容を格納します。
A　　･･･ A軸(または$\overline{\text{SOUT3}}$信号)のAXIS MCM SPEC構造体の内容を格納します。

UNIT MCM SPEC0 SET関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットに対し、MCM SPEC0を設定します。

この関数を実行する前に、MCM STATUS1の全軸のMERR = 0，MBUSY = 0を確認してください。
この関数の実行中は、全軸のMCM STATUS1のMBUSY = 1になります。

UNIT MCM SPEC0 SET関数は、ユニット全体の関数です。
直接ユーザアプリケーションからRAM領域に書き込み／読み出しできるように切り替えます。

書　式

C言語　BOOL MC07_USetMcmSpec0(DWORD *hUnit*, WORD *Spec*, MC07_S_RESULT **psResult*);

```
VB      Function MC07_USetMcmSpec0( ByVal hUnit As Long, ByVal Spec As Integer,
                ByRef psResult As MC07_S_RESULT ) As Boolean

VB.NET  Function MC07_USetMcmSpec0( ByVal hUnit As Integer, ByVal Spec As Short,
                ByRef psResult As MC07_S_RESULT ) As Boolean

C#.NET  bool MC07.USetMcmSpec0( uint hUnit, ushort Spec,
                ref MC07_S_RESULT psResult );
```

引　数

hUnit　　･･･ ユニットハンドルを指定します。
Spec　　･･･ MCM SPEC0の設定データを指定します。
MCMの動作モードを設定します。ユニット単位の設定です。

D15	D14	D13	D12	D11	D10	D9	D8
0	0	0	0	0	0	0	0
D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	RAM ACCESS ENABLE

電源投入後の初期値は H'0000（アンダーライン側）です。

D0　：RAM ACCESS ENABLE
ユーザアプリケーションから、UNIT MCM RAM WRITE関数、UNIT MCM RAM READ関数を用いた
RAM領域へのアクセス機能を「有効にする／無効にする」を設定します。

0：RAM領域へのアクセス機能を無効にする
1：RAM領域へのアクセス機能を有効にする

psResult　　･･･ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM SPEC0 GET関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットに対し、MCM SPEC0の設定を読み出します。

書　式

C言語　BOOL MC07_UGetMcmSpec0(DWORD *hUnit*, WORD **pSpec*, MC07_S_RESULT **psResult*);

VB　Function MC07_UGetMcmSpec0(ByVal *hUnit* As Long, ByRef *pSpec* As Integer,
ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET　Function MC07_UGetMcmSpec0(ByVal *hUnit* As Integer, ByRef *pSpec* As Short,
ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET　bool MC07.UGetMcmSpec0(uint *hUnit*, ref ushort *pSpec*, ref MC07_S_RESULT *psResult*);

引　数

hUnit　　･･･ ユニットハンドルを指定します。
pSpec　　･･･ MCM SPEC0の設定データを格納するための変数のポインタを指定します。
psResult　･･･ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM SPEC1 SET関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

指定されたユニットにMCM SPEC1を設定します。

この関数を実行する前に、MCM STATUS1の全軸のMERR = 0，MBUSY = 0を確認してください。

UNIT MCM SPEC1 SET関数は、ユニット内の１軸に対する設定です。
MCMの同期を行うための信号の割り当てと機能を設定します。
・SS1 SELを「多軸同期の同期信号」に設定し、SS1 OUT1 ASSIGNを設定することにより、
内部トリガによる多軸同期を行います
・SS1 SELを「ユニット外部からの同期信号」に設定することで、
外部トリガによる同期を行います

書 式

C言語
```
BOOL MCO7_USetMcmSpec1( DWORD hUnit, DWORD AxisSel,
        MCO7_S_UNIT_MCM_SPEC *psUnitMcmSpec,
        MCO7_S_RESULT *psResult );
```

VB
```
Function MCO7_USetMcmSpec1( ByVal hUnit As Long, ByVal AxisSel As Long,
        ByRef psUnitMcmSpec As MCO7_S_UNIT_MCM_SPEC,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

VB.NET
```
Function MCO7_USetMcmSpec1( ByVal hUnit As Integer, ByVal AxisSel As Integer,
        ByRef psUnitMcmSpec As MCO7_S_UNIT_MCM_SPEC,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

C#.NET
```
bool MCO7.USetMcmSpec1( uint hUnit, uint AxisSel,
        ref MCO7_S_UNIT_MCM_SPEC psUnitMcmSpec,
        ref MCO7_S_RESULT psResult );
```

引 数

hUnit ・・・ ユニットハンドルを指定します。
AxisSel ・・・ 軸（複数指定可）を指定します。

複数の軸を指定する場合は、論理和を指定します

指定できる値	軸
MCO7_SEL_X	X軸
MCO7_SEL_Y	Y軸
MCO7_SEL_Z	Z軸
MCO7_SEL_A	A軸

次の指定も組み合わせることができます

指定できる値	軸
MCO7_SEL_X_Y	X軸とY軸
MCO7_SEL_X_Z	X軸とZ軸
MCO7_SEL_X_A	X軸とA軸
MCO7_SEL_Y_Z	Y軸とZ軸
MCO7_SEL_Y_A	Y軸とA軸
MCO7_SEL_Z_A	Z軸とA軸

指定できる値	軸
MCO7_SEL_X_Y_Z	X軸とY軸とZ軸
MCO7_SEL_X_Y_A	X軸とY軸とA軸
MCO7_SEL_X_Z_A	X軸とZ軸とA軸
MCO7_SEL_Y_Z_A	Y軸とZ軸とA軸
MCO7_SEL_X_Y_Z_A	X軸とY軸とZ軸とA軸

psUnitMcmSpec ･･･ 設定するデータが格納されているUNIT MCM SPEC構造体のポインタを指定します。

D15	D14	D13	D12	D11	D10	D9	D8
0	0	0	0	0	SS1 SEL2	SS1 SEL1	SS1 SEL0
D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	SS1 AOUT1 ASSIGN	SS1 ZOUT1 ASSIGN	SS1 YOUT1 ASSIGN	SS1 XOUT1 ASSIGN

電源投入後の初期値は H'0000（アンダーライン側）です。

D0 ：SS1 XOUT1 ASSIGN
SS1にXOUT1を「割り当てる／割り当てない」を選択します。
0：割り当てない
1：割り当てる

D1 ：SS1 YOUT1 ASSIGN
SS1にYOUT1を「割り当てる／割り当てない」を選択します。
0：割り当てない
1：割り当てる

D2 ：SS1 ZOUT1 ASSIGN
SS1にZOUT1を「割り当てる／割り当てない」を選択します。
0：割り当てない
1：割り当てる

D3 ：SS1 AOUT1 ASSIGN
SS1にAOUT1を「割り当てる／割り当てない」を選択します。
0：割り当てない
1：割り当てる

D10--8 ：SS1 SEL
SS1の機能を選択します。

SEL2	SEL1	SEL0	SS1の機能	SS1に入力する信号
0	0	0	機能なし	常時ローレベル
0	0	1	多軸同期１のマスター軸からの同期信号	OUT1の切り出し信号
0	1	0	多軸同期２のスレーブ軸からの同期信号	OUT1の合成信号
0	1	1	多軸同期２のマスター軸からの同期信号	OUT1のスルー信号
1	0	0	ユニット外部からの同期信号	$\overline{\text{IN0}}$信号
1	0	1	ユニット外部からの同期信号	$\overline{\text{IN1}}$信号
1	1	0	機能なし	常時ローレベル
1	1	1	機能なし	常時ローレベル

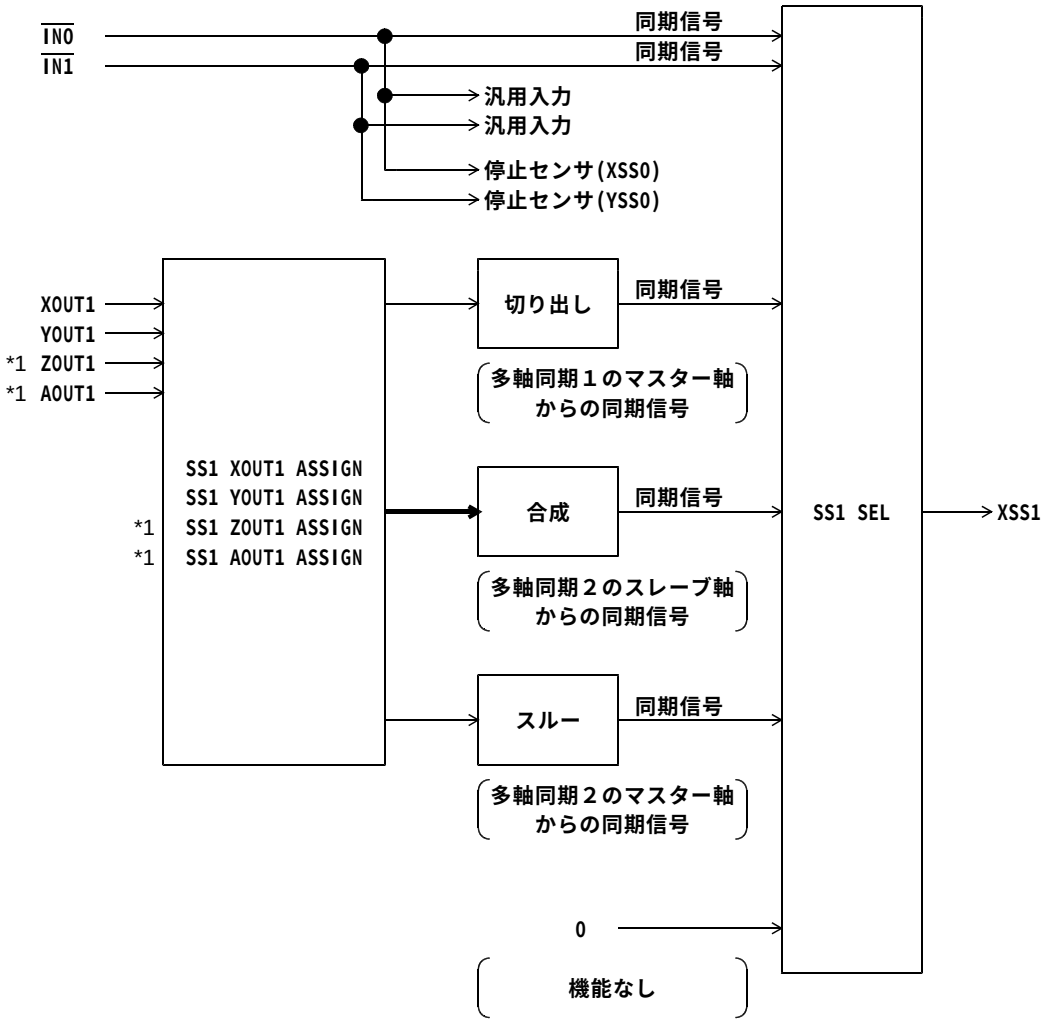
● 内部トリガによる多軸同期
多軸同期１のマスター軸については「機能なし」を選択します。
（多軸同期１のマスター軸とスレーブ軸を兼ねる場合、「多軸同期１のマスター軸からの同期信号」を選択します）

多軸同期１のスレーブ軸は「多軸同期１のマスター軸からの同期信号」を選択します。
多軸同期２のマスター軸は「多軸同期２のスレーブ軸からの同期信号」を選択します。
多軸同期２のスレーブ軸は「多軸同期２のマスター軸からの同期信号」を選択します。

SS1 SELを「マスター軸からの同期信号」にする場合、マスター軸のOUT1のみを
SS1に割り当てるようにSS1 XOUT1 ASSIGN -- SS1 AOUT1 ASSIGNを設定してください。

● 外部トリガによる同期
MCMをIN0信号に同期させる場合、「ユニット外部からの同期信号(IN0信号)」を選択します。
MCMをIN1信号に同期させる場合、「ユニット外部からの同期信号(IN1信号)」を選択します。

【 多軸同期または外部トリガによる同期を行うための信号と軸の機能（X軸の場合）】



*1 ZOUT1,AOUT1,SS1 ZOUT1 ASSIGN,SS1 AOUT1 ASSIGN は、4 軸ユニットのみ指定及び選択が出来ます。

・IN0信号を同期信号として使用する場合、停止センサとして使用しないでください。
・IN1信号を同期信号として使用する場合、停止センサとして使用しないでください。

psResult ・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値
この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM SPEC1 GET関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットのMCM SPEC1の設定を読み出します。

書　式

C言語　BOOL MCO7_UGetMcmSpec1(DWORD *hUnit*, DWORD *AxisSel*,
　　MCO7_S_UNIT_MCM_SPEC **psUnitMcmSpec*,
　　MCO7_S_RESULT **psResult*);

VB　Function MCO7_UGetMcmSpec1(ByVal *hUnit* As Long, ByVal *AxisSel* As Long,
　　ByRef *psUnitMcmSpec* As MCO7_S_UNIT_MCM_SPEC,
　　ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_UGetMcmSpec1(ByVal *hUnit* As Integer, ByVal *AxisSel* As Integer,
　　ByRef *psUnitMcmSpec* As MCO7_S_UNIT_MCM_SPEC,
　　ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.UGetMcmSpec1(uint *hUnit*, uint *AxisSel*,
　　ref MCO7_S_UNIT_MCM_SPEC *psUnitMcmSpec*,
　　ref MCO7_S_RESULT *psResult*);

引　数

hUnit　･･･ ユニットハンドルを指定します。
AxisSel　･･･ MCO7_SEL_X_Y_Z_Aを指定します。
psUnitMcmSpec　･･･ 設定されているデータを格納するためのUNIT MCM SPEC構造体のポインタを指定します。
psResult　･･･ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM SPEC2 SET関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットにMCM SPEC2を設定します。

この関数を実行する前に、MCM STATUS1の全軸のMERR = 0，MBUSY = 0を確認してください。

UNIT MCM SPEC2 SET関数は、ユニット内の１軸に対する設定です。
停止センサSS0に入力する信号を設定します。

書　式

C言語　BOOL MC07_USetMcmSpec2(DWORD *hUnit*, DWORD *AxisSel*,
　　MC07_S_UNIT_MCM_SPEC **psUnitMcmSpec*,
　　MC07_S_RESULT **psResult*);

VB　Function MC07_USetMcmSpec2(ByVal *hUnit* As Long, ByVal *AxisSel* As Long,
　　ByRef *psUnitMcmSpec* As MC07_S_UNIT_MCM_SPEC,
　　ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET　Function MC07_USetMcmSpec2(ByVal *hUnit* As Integer, ByVal *AxisSel* As Integer,
　　ByRef *psUnitMcmSpec* As MC07_S_UNIT_MCM_SPEC,
　　ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET　bool MC07.USetMcmSpec2(uint *hUnit*, uint *AxisSel*,
　　ref MC07_S_UNIT_MCM_SPEC *psUnitMcmSpec*,
　　ref MC07_S_RESULT *psResult*);

引　数

hUnit　　･･･ ユニットハンドルを指定します。
AxisSel　　･･･ 軸（複数指定可）を指定します。

複数の軸を指定する場合は、論理和を指定します

指定できる値	軸
MC07_SEL_X	X軸
MC07_SEL_Y	Y軸
MC07_SEL_Z	Z軸
MC07_SEL_A	A軸

次の指定も組み合わせることができます

指定できる値	軸
MC07_SEL_X_Y	X軸とY軸
MC07_SEL_X_Z	X軸とZ軸
MC07_SEL_X_A	X軸とA軸
MC07_SEL_Y_Z	Y軸とZ軸
MC07_SEL_Y_A	Y軸とA軸
MC07_SEL_Z_A	Z軸とA軸

指定できる値	軸
MC07_SEL_X_Y_Z	X軸とY軸とZ軸
MC07_SEL_X_Y_A	X軸とY軸とA軸
MC07_SEL_X_Z_A	X軸とZ軸とA軸
MC07_SEL_Y_Z_A	Y軸とZ軸とA軸
MC07_SEL_X_Y_Z_A	X軸とY軸とZ軸とA軸

psUnitMcmSpec　･･･ 設定するデータが格納されているUNIT MCM SPEC構造体のポインタを指定します。

D15	D14	D13	D12	D11	D10	D9	D8
0	0	0	0	0	SS0 SEL2	SS0 SEL1	SS0 SEL0
D7	D6	D5	D4	D3	D2	D1	D0
SIGNAL2 SEL				SIGNAL1 SEL			

電源投入後の初期値は次の通りです。
・X軸: H'0000 (SIGNAL1 SEL = 0, SIGNAL2 SEL = 0, SS0 SEL = 0)
・Y軸: H'0001 (SIGNAL1 SEL = 1, SIGNAL2 SEL = 0, SS0 SEL = 0)
・Z軸: H'0400 (SIGNAL1 SEL = 0, SIGNAL2 SEL = 0, SS0 SEL = 4)
・A軸: H'0400 (SIGNAL1 SEL = 0, SIGNAL2 SEL = 0, SS0 SEL = 4)

D3--0 ：SIGNAL1 SEL
D7--4 ：SIGNAL2 SEL
SIGNAL1, SIGNAL2に入力する信号を選択します。

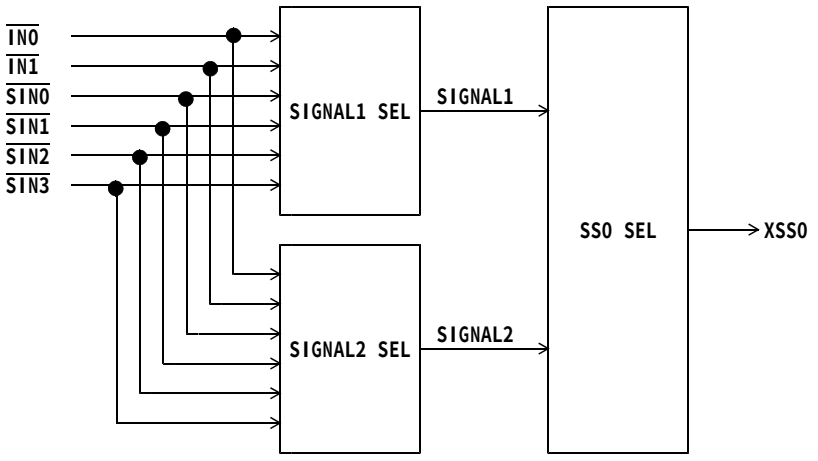
SEL3	SEL2	SEL1	SEL0	SIGNAL1,2に入力する信号
0	0	0	0	$\overline{IN0}$信号
0	0	0	1	$\overline{IN1}$信号
0	0	1	0	設定禁止（常時ローレベル）
0	0	1	1	設定禁止（常時ローレベル）
0	1	0	0	$\overline{SIN0}$信号
0	1	0	1	$\overline{SIN1}$信号
0	1	1	0	$\overline{SIN2}$信号
0	1	1	1	$\overline{SIN3}$信号
1	-	-	-	設定禁止（常時ローレベル）

・SIGNAL1, SIGNAL2に入力する際に負論理信号は正論理信号になります。

D10--8 ：SS0 SEL
SS0に入力する信号を選択します。

SEL2	SEL1	SEL0	SS0に入力する信号
0	0	0	SIGNAL1のスルー出力
0	0	1	SIGNAL1 AND SIGNAL2のスルー出力
0	1	0	SIGNAL1 OR SIGNAL2のスルー出力
0	1	1	設定禁止（常時ローレベル）
1	0	0	設定禁止（常時ローレベル）
1	0	1	設定禁止（常時ローレベル）
1	1	0	設定禁止（常時ローレベル）
1	1	1	設定禁止（常時ローレベル）

【 停止センサ機能（X 軸の場合）】



psResult ・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値
この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM SPEC2 GET関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットのMCM SPEC2の設定を読み出します。

書　式

C言語
```
BOOL MCO7_UGetMcmSpec2( DWORD hUnit, DWORD AxisSel,
        MCO7_S_UNIT_MCM_SPEC *psUnitMcmSpec,
        MCO7_S_RESULT *psResult );
```

VB
```
Function MCO7_UGetMcmSpec2( ByVal hUnit As Long, ByVal AxisSel As Long,
        ByRef psUnitMcmSpec As MCO7_S_UNIT_MCM_SPEC,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

VB.NET
```
Function MCO7_UGetMcmSpec2( ByVal hUnit As Integer, ByVal AxisSel As Integer,
        ByRef psUnitMcmSpec As MCO7_S_UNIT_MCM_SPEC,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

C#.NET
```
bool MCO7.UGetMcmSpec2( uint hUnit, uint AxisSel,
        ref MCO7_S_UNIT_MCM_SPEC psUnitMcmSpec,
        ref MCO7_S_RESULT psResult );
```

引　数

hUnit	･･･ ユニットハンドルを指定します。
AxisSel	･･･ MCO7_SEL_X_Y_Z_Aを指定します。
psUnitMcmSpec	･･･ 設定されているデータを格納するためのUNIT MCM SPEC構造体のポインタを指定します。
psResult	･･･ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM SPEC3 SET関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

指定されたユニットにMCM SPEC3を設定します。
SOUT0信号～SOUT3信号のそれぞれに対する設定です。

この関数を実行する前に、MCM STATUS1の全軸のMERR = 0，MBUSY = 0を確認してください。

UNIT MCM SPEC3 SET関数は、ユニット内の１軸に対する設定です。
ステータス外部出力SOUTx信号に出力する信号を設定します。

書 式

C言語
```
BOOL MC07_USetMcmSpec3( DWORD hUnit, DWORD AxisSel,
        MC07_S_UNIT_MCM_SPEC *psUnitMcmSpec,
        MC07_S_RESULT *psResult );
```

VB
```
Function MC07_USetMcmSpec3( ByVal hUnit As Long, ByVal AxisSel As Long,
        ByRef psUnitMcmSpec As MC07_S_UNIT_MCM_SPEC,
        ByRef psResult As MC07_S_RESULT ) As Boolean
```

VB.NET
```
Function MC07_USetMcmSpec3( ByVal hUnit As Integer, ByVal AxisSel As Integer,
        ByRef psUnitMcmSpec As MC07_S_UNIT_MCM_SPEC,
        ByRef psResult As MC07_S_RESULT ) As Boolean
```

C#.NET
```
bool MC07.USetMcmSpec3( uint hUnit, uint AxisSel,
        ref MC07_S_UNIT_MCM_SPEC psUnitMcmSpec,
        ref MC07_S_RESULT psResult );
```

引 数

hUnit ･･･ ユニットハンドルを指定します。
AxisSel ･･･ 信号（複数指定可）を指定します。

複数の信号を指定する場合は、論理和を指定します

指定できる値	信号
MC07_SEL_X	SOUT0信号
MC07_SEL_Y	SOUT1信号
MC07_SEL_Z	SOUT2信号
MC07_SEL_A	SOUT3信号

次の指定も組み合わせることができます

指定できる値	軸
MC07_SEL_X_Y	SOUT0、SOUT1信号
MC07_SEL_X_Z	SOUT0、SOUT2信号
MC07_SEL_X_A	SOUT0、SOUT3信号
MC07_SEL_Y_Z	SOUT1、SOUT2信号
MC07_SEL_Y_A	SOUT1、SOUT3信号
MC07_SEL_Z_A	SOUT2、SOUT3信号

指定できる値	軸
MC07_SEL_X_Y_Z	SOUT0、SOUT1、SOUT2信号
MC07_SEL_X_Y_A	SOUT0、SOUT1、SOUT3信号
MC07_SEL_X_Z_A	SOUT0、SOUT2、SOUT3信号
MC07_SEL_Y_Z_A	SOUT1、SOUT2、SOUT3信号
MC07_SEL_X_Y_Z_A	SOUT0～SOUT3信号

psUnitMcmSpec ･･･ 設定するデータが格納されているUNIT MCM SPEC構造体のポインタを指定します。

D15	D14	D13	D12	D11	D10	D9	D8
0	0	0	0	0	SOUT SEL2	SOUT SEL1	SOUT SEL0
D7	D6	D5	D4	D3	D2	D1	D0
SIGNAL2 SEL				SIGNAL1 SEL			

電源投入後の初期値は次の通りです。
・$\overline{SOUT0}$: H'0000 (SIGNAL1 SEL = 0, SIGNAL2 SEL = 0, SOUT SEL = 0)
・$\overline{SOUT1}$: H'0001 (SIGNAL1 SEL = 1, SIGNAL2 SEL = 0, SOUT SEL = 0)
・$\overline{SOUT2}$: H'0002 (SIGNAL1 SEL = 2, SIGNAL2 SEL = 0, SOUT SEL = 0)
・$\overline{SOUT3}$: H'0003 (SIGNAL1 SEL = 3, SIGNAL2 SEL = 0, SOUT SEL = 0)

D3--0 ：SIGNAL1 SEL
D7--4 ：SIGNAL2 SEL
SIGNAL1, SIGNAL2に出力する信号を選択します。

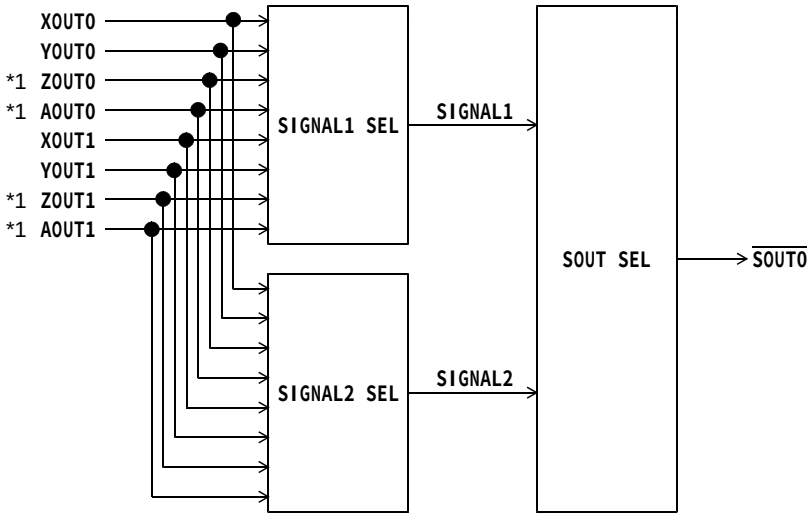
SEL3	SEL2	SEL1	SEL0	SIGNAL1,2に出力する信号
0	0	0	0	XOUT0
0	0	0	1	YOUT0
0	0	1	0	ZOUT0
0	0	1	1	AOUT0
0	1	0	0	XOUT1
0	1	0	1	YOUT1
0	1	1	0	ZOUT1
0	1	1	1	AOUT1
1	-	-	-	設定禁止（常時ローレベル）

D10--8 ：SOUT SEL
$\overline{SOUTx}$信号に出力する信号を選択します。

SEL2	SEL1	SEL0	$\overline{SOUTn}$信号に出力する信号
0	0	0	SIGNAL1のスルー出力
0	0	1	SIGNAL1 AND SIGNAL2のスルー出力
0	1	0	SIGNAL1 OR SIGNAL2のスルー出力
0	1	1	設定禁止（常時ローレベル）
1	0	0	設定禁止（常時ローレベル）
1	0	1	設定禁止（常時ローレベル）
1	1	0	設定禁止（常時ローレベル）
1	1	1	設定禁止（常時ローレベル）

・$\overline{SOUTx}$信号に出力する際に正論理信号は負論理信号になります。

【 ステータス外部出力機能（$\overline{SOUT0}$信号の場合）】



*1 ZOUT0,AOUT0,ZOUT1,AOUT1 は、4 軸ユニットのみ選択が出来ます。

psResult ・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値
この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM SPEC3 GET関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットのMCM SPEC3の設定を読み出します。

書　式

C言語　BOOL MCO7_UGetMcmSpec3(DWORD *hUnit*, DWORD *AxisSel*,
　　MCO7_S_UNIT_MCM_SPEC **psUnitMcmSpec*,
　　MCO7_S_RESULT **psResult*);

VB　Function MCO7_UGetMcmSpec3(ByVal *hUnit* As Long, ByVal *AxisSel* As Long,
　　ByRef *psUnitMcmSpec* As MCO7_S_UNIT_MCM_SPEC,
　　ByRef *psResult* As MCO7_S_RESULT) As Boolean

VB.NET　Function MCO7_UGetMcmSpec3(ByVal *hUnit* As Integer, ByVal *AxisSel* As Integer,
　　ByRef *psUnitMcmSpec* As MCO7_S_UNIT_MCM_SPEC,
　　ByRef *psResult* As MCO7_S_RESULT) As Boolean

C#.NET　bool MCO7.UGetMcmSpec3(uint *hUnit*, uint *AxisSel*,
　　ref MCO7_S_UNIT_MCM_SPEC *psUnitMcmSpec*,
　　ref MCO7_S_RESULT *psResult*);

引　数

hUnit　・・・ ユニットハンドルを指定します。
AxisSel　・・・ MCO7_SEL_X_Y_Z_Aを指定します。
psUnitMcmSpec　・・・ 設定されているデータを格納するためのUNIT MCM SPEC構造体のポインタを指定します。
psResult　・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

2-4-3. シートのダウンロード

MCM のシートから MCC09 の RAM 領域へダウンロードを行います。

UNIT MCM SHEET DOWNLOAD関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットの指定されたシートをRAM領域にダウンロードします。

この関数を実行する前に、MCM STATUS1の全軸のMERR = 0，MBUSY = 0を確認してください。

書　式

C言語
```
BOOL MCO7_UDownloadMcmSheet( DWORD hUnit, WORD SheetNo,
        MCO7_S_RESULT *psResult );
```

VB
```
Function MCO7_UDownloadMcmSheet( ByVal hUnit As Long, ByVal SheetNo As Integer,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

VB.NET
```
Function MCO7_UDownloadMcmSheet( ByVal hUnit As Integer, ByVal SheetNo As Short,
        ByVal psResult As MCO7_S_RESULT ) As Boolean
```

C#.NET
```
bool MCO7.UDownloadMcmSheet( uint hUnit, ushort SheetNo,
        ref MCO7_S_RESULT psResult );
```

引　数

hUnit　　･･･ ユニットハンドルを指定します。
SheetNo　　･･･ シート番号(0～3)を指定します。
psResult　　･･･ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

2-4-4. RAM 領域のアクセス

ユーザアプリケーションから MCC09 の RAM 領域にアクセスを行います。

UNIT MCM RAM WRITE関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットに対し、次の処理を行います。
・RAM領域の指定された軸、指定されたアドレスにコマンドコード、データを書き込みます。

この関数を実行する前に、MCM STATUS1の該当軸のMERR = 0, MBUSY = 0を確認してください。
この関数の実行中は、対象軸のMCM STATUS1のMBUSY = 1になります。

UNIT MCM SPECO SET関数のRAM ACCESS ENABLE =0 の場合、この関数を実行することはできません。
ユーザアプリケーションからRAM領域に書き込み後のRAM動作は、UNIT MCM START関数で実行します。

書　式

C言語
```
BOOL MCO7_UWriteMcmRam( DWORD hUnit, WORD Axis, DWORD Address,
        MCO7_S_COMMAND_DATA *psCmdData,
        MCO7_S_RESULT *psResult );
```

VB
```
Function MCO7_UWriteMcmRam( ByVal hUnit As Long, ByVal Axis As Integer,
        ByVal Address As Long, ByRef psCmdData As MCO7_S_COMMAND_DATA,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

VB.NET
```
Function MCO7_UWriteMcmRam( ByVal hUnit As Integer, ByVal Axis As Short,
        ByVal Address As Integer, ByRef psCmdData As MCO7_S_COMMAND_DATA,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

C#.NET
```
bool MCO7.UWriteMcmRam( uint hUnit, ushort Axis,
        unit Address, ref MCO7_S_COMMAND_DATA psCmdData,
        ref MCO7_S_RESULT psResult );
```

引　数

hUnit　　･･･ ユニットハンドルを指定します。
Axis　　･･･ 軸を指定します。

指定できる値	軸
MCO7_X	X軸
MCO7_Y	Y軸
MCO7_Z	Z軸
MCO7_A	A軸

Address　　･･･ RAM領域のメモリアドレス(0～3,999)を指定します。
psCmdData　　･･･ 書き込むコマンドコード、データが格納されているコマンドデータ構造体のポインタを指定します。
psResult　　･･･ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM RAM READ関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

指定されたユニットに対し、次の処理を行います。
・RAM領域の指定された軸、指定されたアドレスからコマンドコード、データを読み出します

この関数を実行する前に、MCM STATUS1の該当軸のMERR = 0, MBUSY = 0を確認してください。
この関数の実行中は、対象軸のMCM STATUS1のMBUSY = 1になります。

UNIT MCM SPEC0 SET関数のRAM ACCESS ENABLE =0 の場合、この関数は実行することはできません。

書 式

```
C言語BOOL MC07_UReadMcmRam( DWORD hUnit, WORD Axis, DWORD Address,
            MC07_S_COMMAND_DATA *psCmdData,
            MC07_S_RESULT *psResult );

VB      Function MC07_UReadMcmRam( ByVal hUnit As Long, ByVal Axis As Integer,
            ByVal Address As Long, ByRef psCmdData As MC07_S_COMMAND_DATA,
            ByRef psResult As MC07_S_RESULT ) As Boolean

VB.NET  Function MC07_UReadMcmRam( ByVal hUnit As Integer, ByVal Axis As Short,
            ByVal Address As Integer, ByRef psCmdData As MC07_S_COMMAND_DATA,
            ByRef psResult As MC07_S_RESULT ) As Boolean

C#.NET  bool MC07.UReadMcmRam( uint hUnit, ushort Axis,
            unit Address, ref MC07_S_COMMAND_DATA psCmdData,
            ref MC07_S_RESULT psResult );
```

引 数

hUnit ・・・ ユニットハンドルを指定します。
Axis ・・・ 軸を指定します。

指定できる値	軸
MC07_X	X軸
MC07_Y	Y軸
MC07_Z	Z軸
MC07_A	A軸

Address ・・・ RAM領域のメモリアドレス(0～3,999)を指定します。
psCmdData ・・・ 読み出すコマンドコード、データを格納するためのコマンドデータ構造体のポインタを指定します。
psResult ・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

2-4-5. MCM の制御

MCM の制御を行います。

AXIS MCM START DATA構造体

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

MCMを開始するための情報を格納します。

書 式

C言語
```
typedef struct _MCO7_S_AXIS_MCM_START_DATA {
    DWORD Address;
} MCO7_S_AXIS_MCM_START_DATA;
```

VB
```
Type MCO7_S_AXIS_MCM_START_DATA
     Address As Long
End Type
```

VB.NET
```
Structure MCO7_S_AXIS_MCM_START_DATA
     Public Address As Integer
End Structure
```

C#.NET
```
struct MCO7_S_AXIS_MCM_START_DATA
{
     public uint Address;
}
```

メンバ

Address ・・・ 開始アドレスを格納します。

UNIT MCM START DATA構造体

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

MCMを開始するための情報を格納します。

書　式

C言語
```
typedef struct _MCO7_S_UNIT_MCM_START_DATA {
    MCO7_S_AXIS_MCM_START_DATA X;
    MCO7_S_AXIS_MCM_START_DATA Y;
    MCO7_S_AXIS_MCM_START_DATA Z;
    MCO7_S_AXIS_MCM_START_DATA A;
} MCO7_S_UNIT_MCM_START_DATA;
```

VB
```
Type MCO7_S_UNIT_MCM_START_DATA
    X As MCO7_S_AXIS_MCM_START_DATA
    Y As MCO7_S_AXIS_MCM_START_DATA
    Z As MCO7_S_AXIS_MCM_START_DATA
    A As MCO7_S_AXIS_MCM_START_DATA
 End Type
```

VB.NET
```
Structure MCO7_S_UNIT_MCM_START_DATA
    Public X As MCO7_S_AXIS_START_DATA
    Public Y As MCO7_S_AXIS_START_DATA
    Public Z As MCO7_S_AXIS_START_DATA
    Public A As MCO7_S_AXIS_START_DATA
 End Structure
```

C#.NET
```
struct MCO7_S_UNIT_MCM_START_DATA
{
    public MCO7_S_AXIS_MCM_START_DATA X;
    public MCO7_S_AXIS_MCM_START_DATA Y;
    public MCO7_S_AXIS_MCM_START_DATA Z;
    public MCO7_S_AXIS_MCM_START_DATA A;
}
```

メンバ

X	･･･ X軸のAXIS MCM START DATA構造体の内容を格納します。
Y	･･･ Y軸のAXIS MCM START DATA構造体の内容を格納します。
Z	･･･ Z軸のAXIS MCM START DATA構造体の内容を格納します。
A	･･･ A軸のAXIS MCM START DATA構造体の内容を格納します。

UNIT MCM START関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットに対し、次の処理を一括で行います。
・指定された軸（複数指定可）のMCMを開始します。

この関数を実行する前に、MCM STATUS1の該当軸のMERR = 0, MBUSY = 0を確認してください。

書　式

C言語
```
BOOL MC07_UStartMcm( DWORD hUnit, DWORD AxisSel,
        MC07_S_UNIT_MCM_START_DATA *psUnitMcmStartData,
        MC07_S_RESULT *psResult );
```

VB
```
Function MC07_UStartMcm( ByVal hUnit As Long, ByVal AxisSel As Long,
        ByRef psUnitMcmStartData As MC07_S_UNIT_MCM_START_DATA,
        ByRef psResult As MC07_S_RESULT ) As Boolean
```

VB.NET
```
Function MC07_UStartMcm( ByVal hUnit As Integer, ByVal AxisSel As Integer,
        ByRef psUnitMcmStartData As MC07_S_UNIT_MCM_START_DATA,
        ByRef psResult As MC07_S_RESULT ) As Boolean
```

C#.NET
```
bool MC07.UStartMcm( uint hUnit, uint AxisSel,
        ref MC07_S_UNIT_MCM_START_DATA psUnitMcmStartData,
        ref MC07_S_RESULT psResult );
```

引　数

hUnit　　・・・ ユニットハンドルを指定します。
AxisSel　　・・・ 軸（複数指定可）を指定します。

複数の軸を指定する場合は、論理和を指定します

指定できる値	軸
MC07_SEL_X	X軸
MC07_SEL_Y	Y軸
MC07_SEL_Z	Z軸
MC07_SEL_A	A軸

次の指定も組み合わせる事が可能です。

指定できる値	軸
MC07_SEL_X_Y	X軸とY軸
MC07_SEL_X_Z	X軸とZ軸
MC07_SEL_X_A	X軸とA軸
MC07_SEL_Y_Z	Y軸とZ軸
MC07_SEL_Y_A	Y軸とA軸
MC07_SEL_Z_A	Z軸とA軸

指定できる値	軸
MC07_SEL_X_Y_Z	X軸とY軸とZ軸
MC07_SEL_X_Y_A	X軸とY軸とA軸
MC07_SEL_X_Z_A	X軸とZ軸とA軸
MC07_SEL_Y_Z_A	Y軸とZ軸とA軸
MC07_SEL_X_Y_Z_A	X軸とY軸とZ軸とA軸

psUnitMcmStartData　・・・ MCMを開始するための情報が格納されているUNIT MCM START DATA構造体のポインタを指定します。
psResult　・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM FSSTOP関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットに対し、次の処理を一括で行います。
・指定された軸（複数指定可）に即時停止機能を実行します。

書　式

C言語　BOOL MC07_UFsstopMcm(DWORD *hUnit*, DWORD *AxisSel*,
　　MC07_S_RESULT **psResult*);

VB　Function MC07_UFsstopMcm(ByVal *hUnit* As Long,
　　ByVal *AxisSel* As Long,
　　ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET　Function MC07_UFsstopMcm(ByVal *hUnit* As Integer,
　　ByVal *AxisSel* As Integer,
　　ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET　bool MC07.UFsstopMcm(uint *hUnit*, uint *AxisSel*,
　　ref MC07_S_RESULT *psResult*);

引　数

hUnit　　･･･ ユニットハンドルを指定します。
AxisSel　　･･･ 軸（複数指定可）を指定します。

複数の軸を指定する場合は、論理和を指定します

指定できる値	軸
MC07_SEL_X	X軸
MC07_SEL_Y	Y軸
MC07_SEL_Z	Z軸
MC07_SEL_A	A軸

次の指定も組み合わせる事が可能です。

指定できる値	軸
MC07_SEL_X_Y	X軸とY軸
MC07_SEL_X_Z	X軸とZ軸
MC07_SEL_X_A	X軸とA軸
MC07_SEL_Y_Z	Y軸とZ軸
MC07_SEL_Y_A	Y軸とA軸
MC07_SEL_Z_A	Z軸とA軸

指定できる値	軸
MC07_SEL_X_Y_Z	X軸とY軸とZ軸
MC07_SEL_X_Y_A	X軸とY軸とA軸
MC07_SEL_X_Z_A	X軸とZ軸とA軸
MC07_SEL_Y_Z_A	Y軸とZ軸とA軸
MC07_SEL_X_Y_Z_A	X軸とY軸とZ軸とA軸

psResult　　･･･ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM SLSTOP関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットに対し、次の処理を一括で行います。
・指定された軸（複数指定可）に減速停止機能を実行します。

書　式

C言語　BOOL MC07_USlstopMcm(DWORD *hUnit*, DWORD *AxisSel*,
　　MC07_S_RESULT **psResult*);

VB　Function MC07_USlstopMcm(ByVal *hUnit* As Long,
　　ByVal *AxisSel* As Long,
　　ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET　Function MC07_USlstopMcm(ByVal *hUnit* As Integer,
　　ByVal *AxisSel* As Integer,
　　ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET　bool MC07.USlstopMcm(uint *hUnit*, uint *AxisSel*,
　　ref MC07_S_RESULT *psResult*);

引　数

hUnit　　･･･ ユニットハンドルを指定します。
AxisSel　　･･･ 軸（複数指定可）を指定します。

複数の軸を指定する場合は、論理和を指定します

指定できる値	軸
MC07_SEL_X	X軸
MC07_SEL_Y	Y軸
MC07_SEL_Z	Z軸
MC07_SEL_A	A軸

次の指定も組み合わせる事が可能です。

指定できる値	軸
MC07_SEL_X_Y	X軸とY軸
MC07_SEL_X_Z	X軸とZ軸
MC07_SEL_X_A	X軸とA軸
MC07_SEL_Y_Z	Y軸とZ軸
MC07_SEL_Y_A	Y軸とA軸
MC07_SEL_Z_A	Z軸とA軸

指定できる値	軸
MC07_SEL_X_Y_Z	X軸とY軸とZ軸
MC07_SEL_X_Y_A	X軸とY軸とA軸
MC07_SEL_X_Z_A	X軸とZ軸とA軸
MC07_SEL_Y_Z_A	Y軸とZ軸とA軸
MC07_SEL_X_Y_Z_A	X軸とY軸とZ軸とA軸

psResult　　･･･ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

UNIT MCM ERROR CLR関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

指定されたユニットに対し、次の処理を一括で行います。
・指定された軸（複数指定可）のMCMエラーをクリアします。

書 式

C言語
```
BOOL MCO7_UClrMcmError( DWORD hUnit, DWORD AxisSel,
        MCO7_S_RESULT *psResult );
```

VB
```
Function MCO7_UClrMcmError( ByVal hUnit As Long,
        ByVal AxisSel As Long,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

VB.NET
```
Function MCO7_UClrMcmError( ByVal hUnit As Integer,
        ByVal AxisSel As Integer,
        ByRef psResult As MCO7_S_RESULT ) As Boolean
```

C#.NET
```
bool MCO7.UClrMcmError( uint hUnit, uint AxisSel,
        ref MCO7_S_RESULT psResult );
```

引 数

hUnit ・・・ ユニットハンドルを指定します。
AxisSel ・・・ 軸（複数指定可）を指定します。

複数の軸を指定する場合は、論理和を指定します

指定できる値	軸
MCO7_SEL_X	X軸
MCO7_SEL_Y	Y軸
MCO7_SEL_Z	Z軸
MCO7_SEL_A	A軸

次の指定も組み合わせる事が可能です。

指定できる値	軸
MCO7_SEL_X_Y	X軸とY軸
MCO7_SEL_X_Z	X軸とZ軸
MCO7_SEL_X_A	X軸とA軸
MCO7_SEL_Y_Z	Y軸とZ軸
MCO7_SEL_Y_A	Y軸とA軸
MCO7_SEL_Z_A	Z軸とA軸

指定できる値	軸
MCO7_SEL_X_Y_Z	X軸とY軸とZ軸
MCO7_SEL_X_Y_A	X軸とY軸とA軸
MCO7_SEL_X_Z_A	X軸とZ軸とA軸
MCO7_SEL_Y_Z_A	Y軸とZ軸とA軸
MCO7_SEL_X_Y_Z_A	X軸とY軸とZ軸とA軸

psResult ・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値

この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

2-4-6. 状態の表示

ユニットの状態を読み出します。
読み出される内容については、UNIT MCM STATUS READ 関数を参照してください。

AXIS MCM STATUS構造体

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

軸のステータスの内容が格納されます。

書 式

C言語
```
typedef struct _MCO7_S_AXIS_MCM_STATUS {
    WORD  Status;
    WORD NopData;
    DWORD Data;
} MCO7_S_AXIS_MCM_STATUS;
```

VB
```
Type MCO7_S_AXIS_MCM_STATUS
    Status As Integer
    NopData As Integer
    Data As Long
End Type
```

VB.NET
```
Structure MCO7_S_AXIS_MCM_STATUS
    Public Status As Short
    Public NopData As Short
    Public Data As Integer
End Structure
```

C#.NET
```
struct MCO7_S_AXIS_MCM_STATUS
{
   public ushort Status;
   public ushort NopData;
   public uint Data;
}
```

メンバ

Status ･･･ (将来の拡張用です。)
NopData ･･･ 汎用レジスタのデータが格納されます。
Data ･･･ パルスカウンタのカウントデータが格納されます。

UNIT MCM STATUS構造体

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

ユニットのステータスの内容が格納されます。

書　式

C言語

```
typedef struct _MCO7_S_UNIT_MCM_STATUS {
    WORD McmStatus1;
    WORD McmStatus2;
    MCO7_S_AXIS_MCM_STATUS X;
    MCO7_S_AXIS_MCM_STATUS Y;
    MCO7_S_AXIS_MCM_STATUS Z;
    MCO7_S_AXIS_MCM_STATUS A;
    WORD Gpin;
    WORD CtlpOin;
    WORD ExpOin;
    WORD Exp1in;
} MCO7_S_UNIT_MCM_STATUS;
```

VB

```
Type MCO7_S_UNIT_MCM_STATUS
    McmStatus1 As Integer
    McmStatus2 As Integer
    X As MCO7_S_AXIS_MCM_STATUS
    Y As MCO7_S_AXIS_MCM_STATUS
    Z As MCO7_S_AXIS_MCM_STATUS
    A As MCO7_S_AXIS_MCM_STATUS
    Gpin As Integer
    CtlpOin As Integer
    ExpOin As Integer
    Exp1in As Integer
End Type
```

VB.NET

```
Structure MCO7_S_UNIT_MCM_STATUS
    Public McmStatus1 As Short
    Public McmStatus2 As Short
    Public X As MCO7_S_AXIS_STATUS
    Public Y As MCO7_S_AXIS_STATUS
    Public Z As MCO7_S_AXIS_STATUS
    Public A As MCO7_S_AXIS_STATUS
    Public Gpin As Short
    Public CtlpOin As Short
    Public ExpOin As Short
    Public Exp1in As Short
End Structure
```

C#.NET

```
struct MCO7_S_UNIT_MCM_STATUS
{
    public ushort McmStatus1;
    public ushort McmStatus2;
    public MCO7_S_AXIS_MCM_STATUS X;
    public MCO7_S_AXIS_MCM_STATUS Y;
    public MCO7_S_AXIS_MCM_STATUS Z;
    public MCO7_S_AXIS_MCM_STATUS A;
    public ushort Gpin;
    public ushort CtlpOin;
    public ushort ExpOin;
    public ushort Exp1in;
}
```

メンバ

McmStatus1　　･･･ MCM STATUS1の内容が格納されます。
McmStatus2　　･･･ MCM STATUS2の内容が格納されます。
X　　･･･ X軸のAXIS MCM STATUS構造体の内容が格納されます。
Y　　･･･ Y軸のAXIS MCM STATUS構造体の内容が格納されます。
Z　　･･･ Z軸のAXIS MCM STATUS構造体の内容が格納されます。
A　　･･･ A軸のAXIS MCM STATUS構造体の内容が格納されます。
Gpin　　･･･ 汎用I/O入力PORTから読み出された内容が格納されます。
CtlpOin　　･･･ 制御I/O入力0 PORTから読み出された内容が格納されます。
ExpOin　　･･･ 拡張I/O入力0 PORTから読み出された内容が格納されます。
Exp1in　　･･･ 拡張I/O入力1 PORTから読み出された内容が格納されます。

UNIT MCM STATUS READ関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機 能

指定されたユニットの状態を読み出します。

書 式

C言語
```
BOOL MCO7_UReadMcmStatus( DWORD hUnit, DWORD AxisSel,
    MCO7_S_UNIT_MCM_STATUS *psUnitMcmStatus,
    MCO7_S_RESULT *psResult );
```

VB
```
Function MCO7_UReadMcmStatus( ByVal hUnit As Long, ByVal AxisSel As Long,
    ByRef psUnitMcmStatus As MCO7_S_UNIT_MCM_STATUS,
    ByRef psResult As MCO7_S_RESULT ) As Boolean
```

VB.NET
```
Function MCO7_UReadMcmStatus( ByVal hUnit As Integer, ByVal AxisSel As Integer,
    ByRef psUnitMcmStatus As MCO7_S_UNIT_MCM_STATUS,
    ByRef psResult As MCO7_S_RESULT ) As Boolean
```

C#.NET
```
bool MCO7.UReadMcmStatus( uint hUnit, uint AxisSel,
    ref MCO7_S_UNIT_MCM_STATUS psUnitMcmStatus, ref MCO7_S_RESULT psResult );
```

引 数

hUnit ・・・ ユニットハンドルを指定します。
AxisSel ・・・ MCO7_SEL_X_Y_Z_Aを指定します。
psUnitMcmStatus ・・・ 読み出した内容を格納するためのUNIT MCM STATUS構造体のポインタを指定します。
読み出されるユニットの状態は下記です。

ユニットの状態	説明
MCM STATUS1	MCMに関連するユニットの現在の状態
MCM STATUS2	MCMに関連するユニットの現在の状態
パルスカウンタのカウントデータ	パルスカウンタのカウントデータ
汎用レジスタのデータ	NO OPERATIONコマンドで設定された汎用レジスタのデータ
I/O PORT	汎用I/O PORT、制御I/O PORT、拡張I/O PORTの入力状態

●MCM STATUS1
ユニット単位の状態です。
MCMに関連するユニットの現在の状態を表示します。

D15	D14	D13	D12	D11	D10	D9	D8
0	A MERR	A MRUN	A MBUSY	0	Z MERR	Z MRUN	Z MBUSY
D7	D6	D5	D4	D3	D2	D1	D0
0	Y MERR	Y MRUN	Y MBUSY	0	X MERR	X MRUN	X MBUSY

D0 ：X MBUSY
D4 ：Y MBUSY
D8 ：Z MBUSY
D12 ：A MBUSY
MCMの実行中の状態を示します。
1: MCMの開始と同時にセットします
0: MCMの終了でクリアします

UNIT MCM START関数を実行すると、MBUSY = 0→1になります。
STATUS1 PORTのBUSY = 1の場合、MCMの実行状態に関わらずMBUSY = 1になります。
また、次のMCM関数を実行中は、全軸または対象軸のMBUSY = 1になります。
・UNIT MCM SPEC0 SET関数(全軸)
・UNIT MCM SPEC1 SET関数(全軸)
・UNIT MCM SPEC2 SET関数(全軸)
・UNIT MCM SPEC3 SET関数(全軸)
・UNIT MCM SHEET DOWNLOAD関数(全軸)
・UNIT MCM RAM WRITE関数(対象軸)
・UNIT MCM RAM READ関数(対象軸)

D1 ：X MRUN
D5 ：Y MRUN
D9 ：Z MRUN
D13 ：A MRUN
MCMの実行中の状態を示します。
1：MCMの開始と同時にセットします
0：MCMの終了でクリアします

・UNIT MCM START関数を実行すると、MBUSY = 0→1と同時にMRUN = 0→1になります。
・DRIVE STATUS1 PORTのBUSY = 1の場合、MCMの実行状態に関わらずMRUN = 1になります。

D2 ：X MERR
D6 ：Y MERR
D10 ：Z MERR
D12 ：A MERR
MCMエラーが発生したことを示します
1: エラーの発生と同時にセットします
0: UNIT MCM ERROR CLR関数の実行でクリアします

STATUS1 PORTのERROR = 1またはORIGIN STATUSのORIGIN ERROR = 1の場合、
MCMエラーの発生状況に関わらずMERR = 1になります。

●MCM STATUS2
ユニット単位の状態です。
MCMに関連するユニットの現在の状態を表示します。

D15	D14	D13	D12	D11	D10	D9	D8
0	0	A MRSTBY	A MRBUSY	0	0	Z MRSTBY	Z MRBUSY
D7	D6	D5	D4	D3	D2	D1	D0
0	0	Y MRSTBY	Y MRBUSY	0	0	X MRSTBY	X MRBUSY

D0 ：X MRBUSY
D4 ：Y MRBUSY
D8 ：Z MRBUSY
D12 ：A MRBUSY
MCMがRAM領域のコマンドを自動実行中であることを示します。
1: コマンドを自動実行中の状態

D1 : X MRSTBY
D5 : Y MRSTBY
D9 : Z MRSTBY
D13 : A MRSTBY
MCMがコマンドの実行タイミングを待っている状態であることを示します。
1: コマンドの実行タイミングを待っている状態
0: 実行タイミングの検出でクリアします
UNIT MCM FSSTOP関数またはUNIT MCM SLSTOP関数の実行でクリアします
MERR = 1の検出でクリアします

● パルスカウンタのカウントデータ
軸単位の状態です。
パルスカウンタのカウントデータを表示します。

● 汎用レジスタのデータ
軸単位の状態です。
NO OPERATIONコマンドで設定された汎用レジスタのデータを表示します。

● I/O PORT
ユニット単位の状態です。
汎用I/O PORT、制御I/O PORT、拡張I/O PORTの入力状態を表示します。

psResult ・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値
この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

2-4-7. エラー情報の表示

ユニットで発生した動作エラーのエラー情報を読み出します。
読み出される内容については、UNIT MCM ERROR STATUS READ 関数を参照してください。

AXIS MCM ERROR STATUS構造体

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

軸のエラー情報が格納されます。

書　式

C言語
```
typedef struct _MCO7_S_AXIS_MCM_ERROR_STATUS {
    WORD ErrorCode;
    WORD MccErrorStatus;
    WORD MccErrorStatusMask;
    WORD OriginErrorStatus;
} MCO7_S_AXIS_MCM_ERROR_STATUS;
```

VB
```
Type MCO7_S_AXIS_MCM_ERROR_STATUS
    ErrorCode As Integer
    MccErrorStatus As Integer
    MccErrorStatusMask As Integer
    OriginErrorStatus As Integer
} MCO7_S_AXIS_MCM_ERROR_STATUS;
```

VB.NET
```
Structure MCO7_S_AXIS_MCM_ERROR_STATUS
    Public ErrorCode As short
    Public MccErrorStatus As short
    Public MccErrorStatusMask As short
    Public OriginErrorStatus As short
End Structure
```

C#.NET
```
struct MCO7_S_AXIS_MCM_ERROR_STATUS
{
    public ushort ErrorCode;
    public ushort MccErrorStatus;
    public ushort MccErrorStatusMask;
    public ushort OriginErrorStatus;
}
```

メンバ

ErrorCode ･･･ MCM関数のエラーのエラーコードが格納されます
MccErrorStatus ･･･ ERROR STATUSが格納されます
MccErrorStatusMask ･･･ ERROR STATUSのマスクデータが格納されます
OriginErrorStatus ･･･ ORIGIN STATUSのエラーの状態が格納されます

UNIT MCM ERROR STATUS構造体

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

ユニットのエラー情報が格納されます。

書　式

C言語
```
typedef struct _MC07_S_UNIT_MCM_ERROR_STATUS {
    MC07_S_AXIS_MCM_ERROR_STATUS X;
    MC07_S_AXIS_MCM_ERROR_STATUS Y;
    MC07_S_AXIS_MCM_ERROR_STATUS Z;
    MC07_S_AXIS_MCM_ERROR_STATUS A;
} MC07_S_UNIT_MCM_ERROR_STATUS;
```

VB
```
Type MC07_S_UNIT_MCM_ERROR_STATUS
    X As MC07_S_AXIS_MCM_ERROR_STATUS
    Y As MC07_S_AXIS_MCM_ERROR_STATUS
    Z As MC07_S_AXIS_MCM_ERROR_STATUS
    A As MC07_S_AXIS_MCM_ERROR_STATUS
 End Type
```

VB.NET
```
Structure MC07_S_UNIT_MCM_ERROR_STATUS
    Public X As MC07_S_AXIS_ERROR_STATUS
    Public Y As MC07_S_AXIS_ERROR_STATUS
    Public Z As MC07_S_AXIS_ERROR_STATUS
    Public A As MC07_S_AXIS_ERROR_STATUS
 End Structure
```

C#.NET
```
struct MC07_S_UNIT_MCM_ERROR_STATUS
{
   public MC07_S_AXIS_MCM_ERROR_STATUS X;
   public MC07_S_AXIS_MCM_ERROR_STATUS Y;
   public MC07_S_AXIS_MCM_ERROR_STATUS Z;
   public MC07_S_AXIS_MCM_ERROR_STATUS A;
}
```

メンバ

X	･･･ X軸のAXIS MCM ERROR STATUS構造体の内容が格納されます
Y	･･･ Y軸のAXIS MCM ERROR STATUS構造体の内容が格納されます
Z	･･･ Z軸のAXIS MCM ERROR STATUS構造体の内容が格納されます
A	･･･ A軸のAXIS MCM ERROR STATUS構造体の内容が格納されます

UNIT MCM ERROR STATUS READ関数

UsbC

UC-7660 | UCD-7620 | UCD-7621 | UCD-7630

機　能

指定されたユニットのエラー状態を読み出します。

書　式

C言語
```
BOOL MC07_UReadMcmErrorStatus( DWORD hUnit, DWORD AxisSel,
    MC07_S_UNIT_MCM_ERROR_STATUS *psUnitMcmErrorStatus,
    MC07_S_RESULT *psResult );
```

VB
```
Function MC07_UReadMcmErrorStatus( ByVal hUnit As Long, ByVal AxisSel As Long,
    ByRef psUnitMcmErrorStatus As MC07_S_UNIT_MCM_ERROR_STATUS,
    ByRef psResult As MC07_S_RESULT ) As Boolean
```

VB.NET
```
Function MC07_UReadMcmErrorStatus( ByVal hUnit As Integer, ByVal AxisSel As Integer,
    ByRef psUnitMcmErrorStatus As MC07_S_UNIT_MCM_ERROR_STATUS,
    ByRef psResult As MC07_S_RESULT ) As Boolean
```

C#.NET
```
bool MC07.UReadMcmErrorStatus( uint hUnit, uint AxisSel,
    ref MC07_S_UNIT_MCM_ERROR_STATUS psUnitMcmErrorStatus,
    ref MC07_S_RESULT psResult );
```

引　数

hUnit ･･･ ユニットハンドルを指定します。
AxisSel ･･･ MC07_SEL_X_Y_Z_Aを指定します。
psUnitMcmErrorStatus ･･ 読み出した内容を格納するためのUNIT MCM ERROR STATUS構造体のポインタを指定します。
読み出されるユニットのエラー情報は下記です。

動作エラーの分類	エラー情報
MCM関数のエラー	MCM関数のエラーのエラーコード
MCC09のエラー	・ERROR STATUS READコマンドのERROR STATUS ・ERROR STATUS MASKコマンドのERROR STATUSのマスクデータ
ORIGINドライブのエラー	ORIGIN STATUSのエラー情報

● MCMエラー

MCMエラーのエラーコードを表示します。

エラーコード	説明
H'0000	エラーはありません *1
H'0010	MCMの実行中にMCC09でエラーが発生しました
H'0014	ERROR STATUS MASKコマンドでFSSTOP,SLSTOP,LSENDのERROR MASKの何れかが「マスクする」に設定されています
H'0015	RAM領域へのアクセス機能は無効です。
H'001A	メモリアドレス3,999のコマンドを実行後、MCMを強制終了しました
H'001B	MCMの終了時に汎用レジスタのデータがH'FFFFのNO OPERATIONコマンドを確認できませんでした
H'0021	同期信号（SS1）へのOUT1の割り当てが正しくありません ・スレーブ軸のSS1に複数軸のOUT1を割り当てることはできません ・スレーブ軸のSS1にOUT1が割り当てられていません ・多軸同期２のマスター軸のSS1にOUT1が割り当てられていません ・多軸同期２のマスター軸のSS1に自軸のOUT1が割り当てられています

*1 MCC09のエラーやORIGINドライブのエラーにより、MCM STATUS1のMERR = 1になっている場合、
MCMエラーのエラーコードは、H'0000（エラーはありません）になります。

● MCC09のエラー

次の情報を表示します。
・ERROR STATUS READコマンドのERROR STATUS
・ERROR STATUS MASKコマンドのERROR STATUSのマスクデータ

● ORIGINドライブのエラー
ORIGIN STATUSのエラー情報を表示します。

D15	D14	D13	D12	D11	D10	D9	D8
ADDRESS ERROR	ERROR PULSE ERROR	SENSOR ERROR	0	0	0	0	0
D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	ORIGIN ERROR	0	0	0	0

・ORIGIN ERROR、SENSOR ERROR、ERROR PULSE ERROR、ADDRESS ERRORについては、
取扱説明書(標準)の「ORIGINドライブステータス読み出し関数」をご覧ください。

psResult ・・・ この関数を実行した結果を格納するためのRESULT構造体のポインタを指定します。

戻り値
この関数を実行した結果、正常終了したときはTRUE、エラーが発生したときはFALSEを返します。

3．コマンド仕様

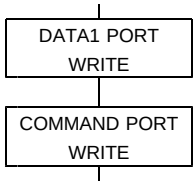
3-1. ドライブコマンド

3-1-1. 入出力仕様の設定

(1) HARD INITIALIZE1

外部ステータス信号(SOUT 信号)に出力する機能、および MCM(RAM)動作の実行タイミング条件を設定します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

<実行シーケンス>



COMMAND H'F1　HARD INITIALIZE1 COMMAND

① DRIVE DATA1 PORT に設定データを書き込みます。

② DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数(DATA2 PORT は、H'0000 を書き込み)

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
OUT3 TYPE3	OUT3 TYPE2	OUT3 TYPE1	OUT3 TYPE0	OUT2 TYPE3	OUT2 TYPE2	OUT2 TYPE1	OUT2 TYPE0

D7	D6	D5	D4	D3	D2	D1	D0
OUT1 TYPE3	OUT1 TYPE2	OUT1 TYPE1	OUT1 TYPE0	OUT0 TYPE3	OUT0 TYPE2	OUT0 TYPE1	OUT0 TYPE0

● 電源投入後の初期値は H'EE31 (アンダーライン側) です。

D3--D0　：OUT0 TYPE3--0　初期値 ＝ CNTINT (外部信号出力可能)
D7--D4　：OUT1 TYPE3--0　初期値 ＝ RDYINT (外部信号出力可能)
D11--D8　：OUT2 TYPE3--0　初期値 ＝ 汎用出力(ステータスを内部フラグに利用可能)
D15--D12　：OUT3 TYPE3--0　初期値 ＝ 汎用出力(ステータスを内部フラグに利用可能)

OUT0, 1 信号および OUT2, 3 信号の出力機能を選択します。

TYPE3	TYPE2	TYPE1	TYPE0	OUT0,1,2,3 の出力機能	
0	0	0	0	ADRINT	カウンタ割り込み要求の ADRINT 出力
0	0	0	1	CNTINT	カウンタ割り込み要求の CNTINT 出力
0	0	1	0	DFLINT	カウンタ割り込み要求の DFLINT 出力
0	0	1	1	RDYINT	コマンド終了割り込み要求の RDYINT 出力
0	1	0	0	STBY	STATUS1 の STBY フラグ出力
0	1	0	1	DRIVE	STATUS1 の DRIVE フラグ出力
0	1	1	0	nSPEED FL	STATUS5 の SPEED FL フラグの反転出力
0	1	1	1	nINDEX FL	STATUS5 の INDEX FL フラグの反転出力
1	0	0	0	UP	STATUS1 の UP フラグ出力
1	0	0	1	DOWN	STATUS1 の DOWN フラグ出力
1	0	1	0	CONST	STATUS1 の CONST フラグ出力
1	0	1	1	EXT PULSE	STATUS1 の EXT PULSE フラグ出力
1	1	0	0	nPULSE MASK	STATUS2 の PULSE MASK フラグの反転出力
1	1	0	1	ORG SIGNAL	STATUS2 の ORG SIGNAL フラグ出力
1	1	1	0	汎用出力	汎用出力として使用する
1	1	1	1	設定禁止	−

「汎用出力」に設定した場合は、SIGNAL OUT コマンドで出力レベルを操作します。

SIGNAL I/O コネクタから外部出力できる信号は、初期設定は下記の割り付けになっています。
UNIT MCM SPEC3 SET 関数で任意軸や OUT1 信号の割り付けも可能です。

外部出力信号	軸と出力(割付信号名)	
	4 軸ユニット	2 軸ユニット
$\overline{\text{SOUT0}}$ 信号	XOUT0(XCNTINT)	XOUT0(XCNTINT)
$\overline{\text{SOUT1}}$ 信号	YOUT0(YCNTINT)	YOUT0(YCNTINT)
$\overline{\text{SOUT2}}$ 信号	ZOUT0(ZCNTINT)	XOUT0(XCNTINT)
$\overline{\text{SOUT3}}$ 信号	AOUT0(ACNTINT)	YOUT0(YCNTINT)

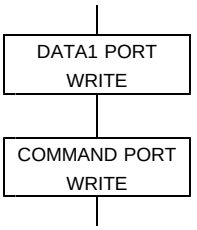
OUT 2,3 信号は、ステータスを利用して、次の機能の条件に応用することができます。

- PAUSE 機能の STBY 解除条件
- SPEED RATE CHANGE の変更動作点
- INDEX CHANGE の変更動作点
- パルス偏差カウンタのパルスカウント開始タイミング
- 各カウンタのラッチタイミング
- RAM 機能の JUMP 条件
- RAM 機能のメモリーデータの NO OPERATION コマンド実行条件

(2) HARD INITIALIZE2

GPIO 2 信号入出力の入出力機能を設定します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

<実行シーケンス>



① DRIVE DATA1 PORT に設定データを書き込みます。

② DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数(DATA2 PORT は、H'0000 を書き込み)

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
–	–	–	–	–	–	–	–

D7	D6	D5	D4	D3	D2	D1	D0
GPIO2 TYPE3	GPIO2 TYPE2	GPIO2 TYPE1	GPIO2 TYPE0	1	1	1	1

● 電源投入後の初期値は H'00FF (アンダーライン側) です。

D3--D0 ：設定禁止(1)
D7--D4 ：GPIO2 TYPE3--0 初期値 ＝ 汎用入力(ステータスを内部フラグに利用可能)

GPIO 2 の入出力機能を選択します。

TYPE3	TYPE2	TYPE1	TYPE0	GPIO2 の入出力機能	
0	0	0	0	ADRINT COMP1	STATUS4 の ADRINT COMP1 フラグ出力
0	0	0	1	ADRINT COMP2	STATUS4 の ADRINT COMP2 フラグ出力
0	0	1	0	ADRINT COMP3	STATUS4 の ADRINT COMP3 フラグ出力
0	0	1	1	XADRINT AND YADRINT	X 軸と Y 軸の ADRINT の AND (論理積) 出力
0	1	0	0	XADRINT OR YADRINT	X 軸と Y 軸の ADRINT の OR (論理和) 出力
0	1	0	1	DFLINT COMP1	STATUS4 の DFLINT COMP1 フラグ出力
0	1	1	0	DFLINT COMP2	STATUS4 の DFLINT COMP2 フラグ出力
0	1	1	1	DFLINT COMP3	STATUS4 の DFLINT COMP3 フラグ出力
1	0	0	0	ERROR	STATUS1 の ERROR フラグ出力
1	0	0	1	LSEND	STATUS1 の LSEND フラグ出力
1	0	1	0	SSEND	STATUS1 の SSEND フラグ出力
1	0	1	1	FSEND	STATUS1 の FSEND フラグ出力
1	1	0	0	SS0	STATUS5 の SS0 フラグ出力
1	1	0	1	DRVEND	STATUS1 の DRVEND フラグ出力
1	1	1	0	汎用出力	汎用出力として使用する
1	1	1	1	汎用入力	汎用入力として使用する

・「汎用出力」に設定した場合は、SIGNAL OUT コマンドで出力レベルが操作できます。

GPIO 2 信号のステータスを利用して、次の機能の条件に応用することができます。
- PAUSE 機能の STBY 解除条件
- SPEED RATE CHANGE の変更動作点
- INDEX CHANGE の変更動作点
- 各カウンタのラッチタイミング
- RAM 機能の JUMP 条件
- RAM 機能のメモリーデータの NO OPERATION コマンド実行条件

(3) HARD INITIALIZE3

GPIO 3 信号の入力機能を設定します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

<実行シーケンス>

COMMAND H'F3　HARD INITIALIZE3 COMMAND

DATA1 PORT WRITE
① DRIVE DATA1 PORT に設定データを書き込みます。

COMMAND PORT WRITE
② DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数(DATA2 PORT は、H'0000 を書き込み)

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
–	–	–	–	–	–	–	–

D7	D6	D5	D4	D3	D2	D1	D0
GPIO3 TYPE3	GPIO3 TYPE2	GPIO3 TYPE1	GPIO3 TYPE0	1	1	1	1

● 電源投入後の初期値は H'FF (アンダーライン側) です。

D3--D0　：設定禁止(1)
D7--D4　：GPIO3 TYPE3--0　初期値 ＝ 汎用入力(ステータスを内部フラグに利用可能)

GPIO 3 の入出力機能を選択します。

TYPE3	TYPE2	TYPE1	TYPE0	GPIO 3 の入出力機能	
0	0	0	0	CNTINT COMP1	STATUS4 の CNTINT COMP1 フラグ出力
0	0	0	1	CNTINT COMP2	STATUS4 の CNTINT COMP2 フラグ出力
0	0	1	0	CNTINT COMP3	STATUS4 の CNTINT COMP3 フラグ出力
0	0	1	1	XCNTINT AND YCNTINT	X 軸と Y 軸の CNTINT の AND (論理積) 出力
0	1	0	0	XCNTINT OR YCNTINT	X 軸と Y 軸の CNTINT の OR (論理和) 出力
0	1	0	1	DFLINT COMP1	STATUS4 の DFLINT COMP1 フラグ出力
0	1	1	0	DFLINT COMP2	STATUS4 の DFLINT COMP2 フラグ出力
0	1	1	1	DFLINT COMP3	STATUS4 の DFLINT COMP3 フラグ出力
1	0	0	0	ERRINT	ERROR STATUS 出力の ERRINT 出力
1	0	0	1	ORGEND	STATUS2 の ORGEND フラグ出力
1	0	1	0	COMREG EP	STATUS1 の COMREG EP フラグ出力
1	0	1	1	COMREG FL	STATUS1 の COMREG FL フラグ出力
1	1	0	0	SS1	STATUS5 の SS1 フラグ出力
1	1	0	1	nBUSY	STATUS1 の BUSY フラグの反転出力
1	1	1	0	汎用出力	汎用出力として使用する
1	1	1	1	汎用入力	汎用入力として使用する

「汎用出力」に設定した場合は、SIGNAL OUT コマンドで出力レベルが操作できます。

GPIO 3 信号のステータスを利用して、次の機能の条件に応用することができます。
- RAM 機能の JUMP 条件
- RAM 機能のメモリーデータの NO OPERATION コマンド実行条件

(4) HARD INITIALIZE7

各入力信号のアクティブ論理を設定します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

> ⚠ 注意
> 設定を誤ると製品が正しく機能しなくなります。
> 入力信号のアクティブ論理を設定するデータ部には、重要な機能(信号)も
> 割り付いています。指定された "1" の論理は間違えないようにしてください。

<実行シーケンス>

COMMAND H'F7　HARD INITIALIZE7 COMMAND

DATA1 PORT WRITE

① DRIVE DATA1 PORT にアクティブデータを書き込みます。

COMMAND PORT WRITE

② DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数(DATA2 PORT は、H'0000 を書き込み)

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
PAUSE ACTIVE(1)	1	1	1	–	NORG ACTIVE(1)	ORG ACTIVE(1)	1

D7	D6	D5	D4	D3	D2	D1	D0
1	DALM ACTIVE(1)	1	1	CCWLM ACTIVE(1)	CWLM ACTIVE(1)	FSSTOP ACTIVE(1)	–

● 電源投入後の初期値は H'F7FE (すべてハイアクティブ) です。
上記の値で、入力アクティブ論理は以下の初期値になっています。

D4, D5, D7, D8, D12--D14 ：設定禁止(1)

D1 ：FSSTOP ACTIVE
FSSTOP 入力信号のアクティブ論理を選択します。
0 ：負論理(ローアクティブ)
1 ：正論理(ハイアクティブ)

D2 ：CWLM ACTIVE
CWLM 入力信号のアクティブ論理を選択します。
0 ：負論理(ローアクティブ)
1 ：正論理(ハイアクティブ)

D3 ：CCWLM ACTIVE
CCWLM 入力信号のアクティブ論理を選択します。
0 ：負論理(ローアクティブ)
1 ：正論理(ハイアクティブ)

D6 ：DALM ACTIVE
DALM 入力信号のアクティブ論理を選択します。
0 ：負論理(ローアクティブ)
1 ：正論理(ハイアクティブ)

上記の信号は、初期値 B 接点入力になっています。

D9 ：ORG ACTIVE
ORG 入力信号のアクティブ論理を選択します。
0 ：正論理(ハイアクティブ)
1 ：負論理(ローアクティブ)

D10 ：NORG ACTIVE
NORG 入力信号のアクティブ論理を選択します。
0 ：正論理(ハイアクティブ)
1 ：負論理(ローアクティブ)

D15 ：PAUSE ACTIVE
PAUSE 信号のアクティブ論理を選択します。
0 ：PAUSE=1 にして STBY にする
1 ：PAUSE=0 にして STBY を解除する

・D15 の PAUSE 信号のアクティブ論理を操作することで、コマンド予約機能による連続ドライブの設定と実行ができます。

・HARD INITIALIZE7 コマンドの実行で、各信号のアクティブ論理を変更します。
アクティブ論理を変更すると、変更した信号の製品内部デジタルフィルタ時定数経過後に、アクティブ論理の変更が確定します。

(5) HARD INITIALIZE8

パルス出力信号のアクティブ論理を設定します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

> ⚠ 注意
> 設定を誤ると製品が正しく機能しなくなります。
> 出力信号のアクティブ論理を設定するデータ部には、重要な機能(信号)も
> 割り付いています。指定された "1" の論理は間違えないようにしてください。

<実行シーケンス>

COMMAND H'F8　HARD INITIALIZE8 COMMAND

DATA1 PORT WRITE

① DRIVE DATA1 PORT にアクティブデータを書き込みます。

COMMAND PORT WRITE

② DRIVE COMMAND PORT に COMMAND を書き込みます。
推奨:MC07_LWDrive 関数(DATA2 PORT は、H'0000 を書き込み)

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
CCWP ACTIVE(0)	CWP ACTIVE(0)	−	−	1	1	1	1

D7	D6	D5	D4	D3	D2	D1	D0
−	−	1	1	1	1	1	1

● 電源投入後の初期値は H'0F3F です。

D0--D5, D8--D11 ：設定禁止(1)

D15 ：CCWP ACTIVE
D14 ：CWP ACTIVE
パルス出力信号のアクティブ論理を選択します。
0 ：ローアクティブ
1 ：ハイアクティブ

CWP, CCWP 信号の電源投入後の初期状態は、「ローアクティブ（負論理出力）」です。
この出力論理の切替は、方向指定型の場合にも有効です。

コントローラドライバ製品では、当コマンドは実行禁止です。

3-1-2. ドライブ パラメータの設定

ドライブのパラメータを設定します。各設定は変更が必要な場合に設定します。

※ SPEED･RATE 関数はこれらのコマンドを使用してドライブパラメータを設定しています。

(1) FSPD SET

ドライブパルス出力の第１パルス目のパルス周期（パルス速度）を設定します。

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D15 ←							FSPD								→ D0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	–	–	–	–	–	–	D22 ←			FSPD			→ D16

●電源投入後の初期値は H'00_1388 （5,000 Hz ：１周期 200 μs） です。

*FSPD は、設定された値の半周期を経過してから第 1 パルス目のパルス出力を開始します。
初期値 200 μ s は、この周期が実際に第 1 パルス目の出力(立ち上がりエッジ)までの遅延時間になります。
よって、起動指令後に速くパルス出力を開始したいときは、ドライバ側の入力応答周波数に注意して、この FSPD の値を設定してください。

FSPD の設定値が "0" の場合は、FSPD を FSPD = RSPD x RESOL に補正します。

・RSPD ：RSPD は、HSPD, LSPD, ELSPD と同様の 15 ビットのパルス速度データです。
DRIVE = 1 → 0 になると、最終出力のパルス速度データを RSPD に記憶します。
ただし、最終出力のパルス速度が FSPD, RFSPD と JSPD の場合は、RSPD を書き換えません。
RSPD のリセット後の初期値は、H'012C（300）です。

・第１パルスのパルス周期の計算式

10,000,000 ／ FSPD ＝ 商 A ＋余り B	→ OFF 周期	＝商 A x 50 ns
(10,000,000 ＋余り B)／ FSPD ＝ 商 C ＋余り D	→ ON 周期	＝商 C x 50 ns

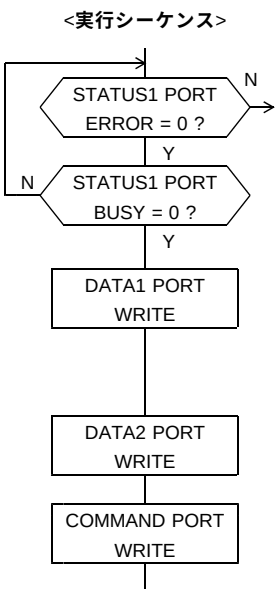
FSPD の設定値と実際に出力する第１パルスのパルス周期

FSPD の設定値	第１パルスのパルス周期（パルス速度）	
8,388,607 ～ 6,666,667 Hz →	OFF 周期 = 50 ns	ON 周期 = 50 ns （10,000,000 Hz）
6,666,666 ～ 5,000,001 Hz →	OFF 周期 = 50 ns	ON 周期 = 100 ns（6,666,666 Hz）
5,000,000 ～ 4,000,001 Hz →	OFF 周期 = 100 ns	ON 周期 = 100 ns（5,000,000 Hz）
4,000,000 ～ 3,333,334 Hz →	OFF 周期 = 100 ns	ON 周期 = 150 ns（4,000,000 Hz）
3,333,333 ～ 2,857,143 Hz →	OFF 周期 = 150 ns	ON 周期 = 150 ns（3,333,333 Hz）

(2) HIGH SPEED SET

加減速ドライブの最高速時のパルス速度データ（HSPD）を設定します。
加減速ドライブの速度データの速度倍率（RESOL）を設定します。

<実行シーケンス>



① DRIVE STATUS1 PORT の ERROR フラグが "0" であることを確認します。

② DRIVE STATUS1 PORT の BUSY フラグが "0" であることを確認します。
推奨:MC07_BRStatus1 関数

③ DRIVE DATA1 PORT に最高速時のパルス速度データを書き込みます。
設定範囲は、0 ～ 32,767（H'0000 ～ H'7FFF）です。
速度倍率を 400 に設定した場合は、
速度データの設定範囲は、0 ～ 25,000（H'0000 ～ H'61A8）です。

④ DRIVE DATA2 PORT に速度倍率の RESOL No. を書き込みます。

⑤ DRIVE COMMAND PORT に COMMAND を書き込みます。
推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	D14 ←						HSPD								→ D0

● 電源投入後の初期値は H'0BB8（3,000：3,000 Hz）です。

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	–	–	–	–	–	–	–	–	–	D3	RESOL No.		D0

● 電源投入後の初期値は H'3（速度倍率＝１）です。

HSPD の設定値が "0" の場合は、HSPD を HSPD = RSPD に補正します。
・最高速時の速度（Hz）＝ HSPD x RESOL

減速ドライブと一定速ドライブの 1 パルス目は、FSPD の第 1 パルスです。
2 パルス目から HSPD x RESOL の速度になります。

RESOL No.を選択して、速度倍率（RESOL）を設定します。

RESOL No.	速度倍率 (RESOL)
H'0	1
H'1	1
H'2	1
H'3	1

RESOL No.	速度倍率 (RESOL)
H'4	2
H'5	5
H'6	10
H'7	20

RESOL No.	速度倍率 (RESOL)
H'8	50
H'9	100
H'A	200
H'B	400

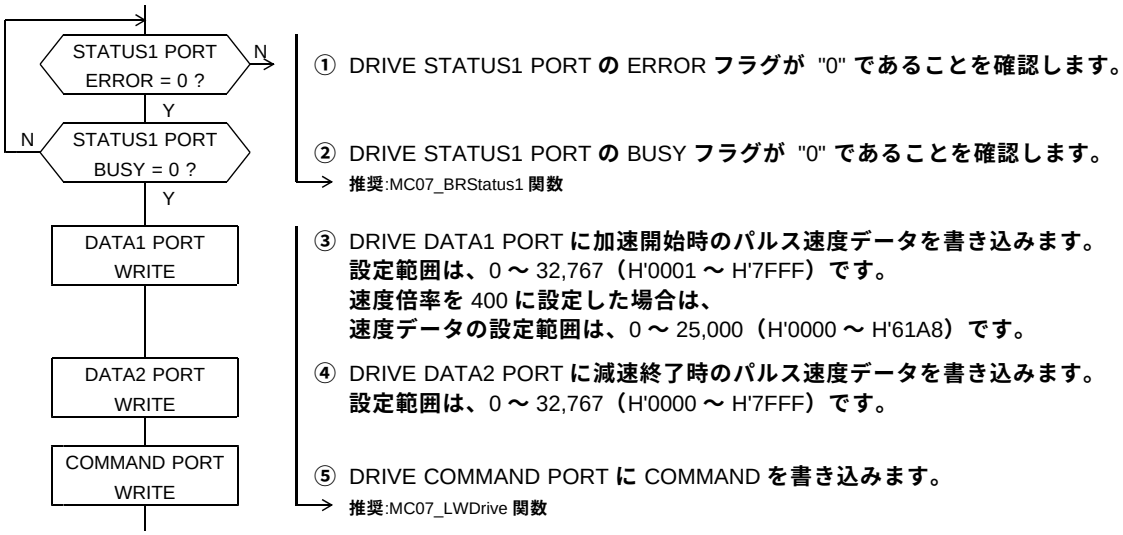
RESOL No.	速度倍率 (RESOL)
H'C	400
H'D	400
H'E	400
H'F	400

・速度設定値　＝ 速度データ x 速度倍率（RESOL）　：1 ～ 10,000,000 Hz
・速度倍率を 400 に設定した場合は、速度データの設定範囲は、0 ～ 25,000（H'0000 ～ H'61A8）です。

(3) LOW SPEED SET

加減速ドライブの加速開始時のパルス速度データ（LSPD**）を設定します。**
加減速ドライブの減速終了時のパルス速度データ（ELSPD**）を設定します。**

<実行シーケンス>　COMMAND H'07　LOW SPEED SET COMMAND



DRIVE DATA1 PORT **の設定データ**

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	D14 ←						LSPD							→	D0

●電源投入後の初期値は H'012C（300：300 Hz） **です。**

DRIVE DATA2 PORT **の設定データ**

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	D14 ←						ELSPD							→	D0

●電源投入後の初期値は H'0000（LSPD **と同じ**） **です。**

LSPD **の設定値が** "0" **の場合は、**LSPD **を** LSPD = RSPD **に補正します。**
・加速開始時の速度（Hz**）＝** LSPD x RESOL

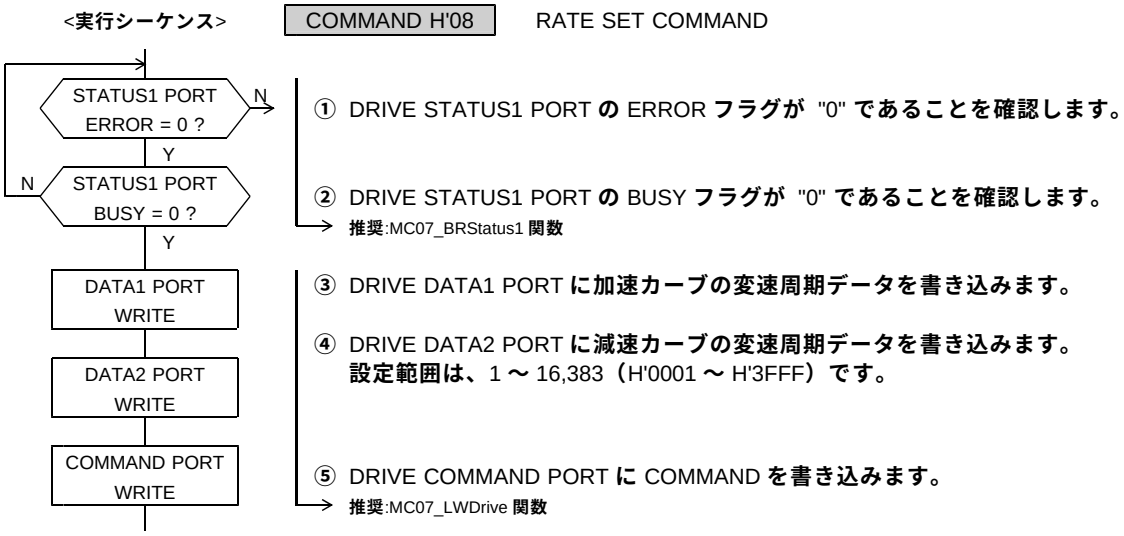
ELSPD **の設定値が** "0" **の場合は、**ELSPD **を** ELSPD = LSPD **に補正します。**
・減速終了時の速度（Hz**）＝** ELSPD x RESOL

加減速ドライブと加速ドライブの１パルス目は、FSPD **の第１パルスです。**
2 **パルス目から** LSPD x RESOL **の速度になります。**

減速ドライブと一定速ドライブの１パルス目は、FSPD **の第１パルスです。**
2 **パルス目から** HSPD x RESOL **の速度になります。**

(4) RATE SET

加速カーブの変速周期データ（UCYCLE**）を設定します。**
減速カーブの変速周期データ（DCYCLE**）を設定します。**



DRIVE DATA1 PORT **の設定データ**

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	–	D13	←				UCYCLE							→	D0

●電源投入後の初期値は H'00C8（200：100 µs **周期）** **です。**

DRIVE DATA2 PORT **の設定データ**

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	–	D13	←				DCYCLE							→	D0

●電源投入後の初期値は H'0000（UCYCLE **と同じ）** **です。**

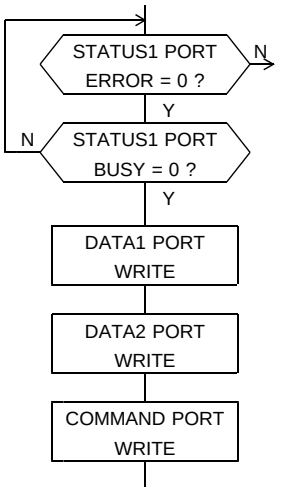
UCYCLE **の設定値が** "0" **の場合は、**"1" **に補正します。**
・変速周期（µs**）＝** UCYCLE x 0.5 µs **：**0.5 µs **～** 8.1915 ms

DCYCLE **の設定値が** "0" **の場合は、**DCYCLE **を** DCYCLE = UCYCLE **に補正します。**
・変速周期（µs**）＝** DCYCLE x 0.5 µs **：**0.5 µs **～** 8.1915 ms

(5) SCAREA SET

加速カーブのＳ字変速領域データ（SUAREA**）を設定します。**
減速カーブのＳ字変速領域データ（SDAREA**）を設定します。**

<実行シーケンス>　COMMAND H'09　SCAREA SET COMMAND



① DRIVE STATUS1 PORT **の** ERROR **フラグが** "0" **であることを確認します。**

② DRIVE STATUS1 PORT **の** BUSY **フラグが** "0" **であることを確認します。**
推奨:MC07_BRStatus1 **関数**

③ DRIVE DATA1 PORT **に加速カーブの** S **字変速領域データを書き込みます。**

④ DRIVE DATA2 PORT **に減速カーブの** S **字変速領域データを書き込みます。**
設定範囲は、0 ～ 32,767（H'0000 ～ H'7FFF）**です。**

⑤ DRIVE COMMAND PORT **に** COMMAND **を書き込みます。**
推奨:MC07_LWDrive **関数**

DRIVE DATA1 PORT **の設定データ**

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	D14	←					SUAREA							→	D0

● **電源投入後の初期値は** H'0000 (0：SUAREA **の変速領域なし**) **です。**

DRIVE DATA2 PORT **の設定データ**

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	D14	←					SDAREA							→	D0

● **電源投入後の初期値は** H'0000 (0：SDAREA **の変速領域なし**) **です。**

SUAREA**（および** SUH**）の設定値が** "0" **の場合は、**
UCYCLE **と** RESOL **による直線加速カーブのみで加速します。**
・Ｓ字加速開始部の変速領域（Hz**）＝** SUAREA x RESOL
・Ｓ字加速終了部の変速領域（Hz**）＝** SUAREA**（または** SUH**）**x RESOL

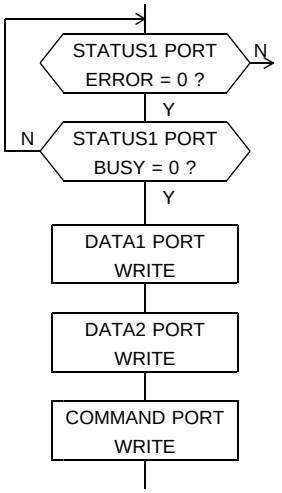
SDAREA**（および** SDH**）の設定値が** "0" **の場合は、**
DCYCLE **と** RESOL **による直線減速カーブのみで減速します。**
・Ｓ字減速開始部の変速領域（Hz**）＝** SDAREA**（または** SDH**）**x RESOL
・Ｓ字減速終了部の変速領域（Hz**）＝** SDAREA x RESOL

● SPEC INITIALIZE3 **コマンドの** SCAREA MODE = 0 **の場合**
変速領域データは以下のように補正します。
・(HSPD - LSPD**) ＜ (**2 * SUAREA**) のときは、重複した変速領域を重ねて加速します。**
・(HSPD - ELSPD**) ＜ (**2 * SDAREA**) のときは、**SDAREA = (HSPD - ELSPD) * 1/2 **に補正します。**

(6) SHAREA SET

SPEC INITIALIZE3 コマンドの SCAREA MODE = 1 に設定している場合に有効です。
加速カーブのＳ字加速終了部の変速領域データ（SUH）を設定します。
減速カーブのＳ字減速開始部の変速領域データ（SDH）を設定します。

<実行シーケンス>　COMMAND H'0B　SHAREA SET COMMAND



① DRIVE STATUS1 PORT の ERROR フラグが "0" であることを確認します。

② DRIVE STATUS1 PORT の BUSY フラグが "0" であることを確認します。
推奨:MC07_BRStatus1 関数

③ DRIVE DATA1 PORT にＳ字加速終了部の変速領域データを書き込みます。

④ DRIVE DATA2 PORT にＳ字減速開始部の変速領域データを書き込みます。
設定範囲は、0 ～ 32,767（H'0000 ～ H'7FFF）です。

⑤ DRIVE COMMAND PORT に COMMAND を書き込みます。
推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
−	D14	←					SUH							→	D0

● 電源投入後の初期値は H'0000（0：SUH の変速領域なし）です。

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
−	D14	←					SDH							→	D0

● 電源投入後の初期値は H'0000（0：SDH の変速領域なし）です。

● SPEC INITIALIZE3 コマンドの SCAREA MODE = 1 の場合
SUAREA および SUH の設定値が "0" の場合は、
UCYCLE と RESOL による直線加速カーブのみで加速します。
・Ｓ字加速開始部の変速領域（Hz）＝ SUAREA x RESOL
・Ｓ字加速終了部の変速領域（Hz）＝ SUH x RESOL

SDAREA および SDH の設定値が "0" の場合は、
DCYCLE と RESOL による直線減速カーブのみで減速します。
・Ｓ字減速開始部の変速領域（Hz）＝ SDH x RESOL
・Ｓ字減速終了部の変速領域（Hz）＝ SDAREA x RESOL

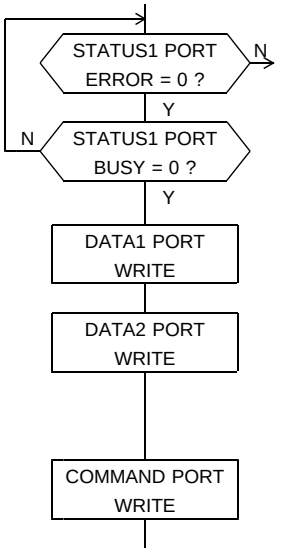
変速領域データは以下のように補正します。
・(HSPD - LSPD) ＜ (SUAREA) のときは、SUAREA = (HSPD - LSPD) に補正します。
・(HSPD - ELSPD) ＜ (SDAREA) のときは、SDAREA = (HSPD - ELSPD) に補正します。
・(HSPD - LSPD) ＜ (SUH + SUAREA) のときは、SUH = (HSPD - LSPD - SUAREA) に補正します。
・(HSPD - ELSPD) ＜ (SDH + SDAREA) のときは、SDH = (HSPD - ELSPD - SDAREA) に補正します。

・HSPD ＜ LSPD or HSPD ＜ ELSPD のときは、SUAREA = SDAREA = SUH = SDH = 0 に補正します。

(7) DOWN PULSE ADJUST

INDEX ドライブの停止位置への減速停止動作時に有効です。
MCC09 が自動検出する減速パルス数に加算するオフセットパルス数を設定します。
または、停止位置への減速停止動作を開始するための減速パルス数をマニュアル設定します。

<実行シーケンス>　COMMAND H'0A　DOWN PULSE ADJUST COMMAND



① DRIVE STATUS1 PORT の ERROR フラグが "0" であることを確認します。

② DRIVE STATUS1 PORT の BUSY フラグが "0" であることを確認します。
推奨:MC07_BRStatus1 関数

③ DRIVE DATA1 PORT にパルス数 D15--D0 を書き込みます。

④ DRIVE DATA2 PORT にパルス数 D23--D16 を書き込みます。

SCAREA MODE = 0 のときの設定範囲は、D15--D0 の
-32,768 ～+32,767（H'8000 ～ H'7FFF）です。
負数の場合は 2 の補数表現にします。
SCAREA MODE = 1 のときの設定範囲は、D23--D0 の
0 ～+16,777,215（H'00_0000 ～ H'FF_FFFF）です。

⑤ DRIVE COMMAND PORT に COMMAND を書き込みます。
推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D15 ←								オフセット／減速パルス数							→ D0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	–	–	–	–	–	D23	←		減速パルス数			→	D16

● 電源投入後の初期値は H'00_0001（＋１パルス）です。

このコマンドは、SPEC INITIALIZE3 コマンドの SCAREA MODE の選択により、
設定データの機能が変わります。

● SPEC INITIALIZE3 コマンドの SCAREA MODE = 0 の場合
DRIVE DATA1 PORT の D15--D0 のみ有効です。
INDEX ドライブのオフセットパルス数を設定します。
設定したオフセットパルス数は、MCC09 が自動検出する減速パルス数に加算します。
・オフセットパルス数を正数にすると、減速パルス数は増加します。
・オフセットパルス数を負数にすると、減速パルス数は減少します。

● SPEC INITIALIZE3 コマンドの SCAREA MODE = 1 の場合
DRIVE DATA1, 2 PORT の D23--D0 が有効です。
INDEX ドライブの減速停止動作を開始するための減速パルス数をマニュアル設定します。
設定した減速パルス数と INDEX ドライブの残アドレス（残パルス数）が同じパルス数になると、
INDEX ドライブの停止位置への減速停止動作を開始します。
・減速パルス数の計算式は、4-1-3.章「S 字加減速ドライブ」に記載しています。

3-1-3. ORIGIN ドライブの設定

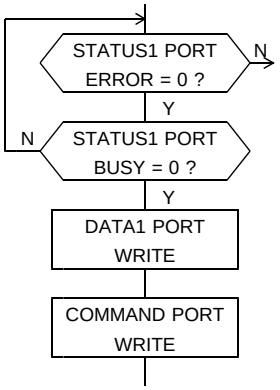
ORIGIN ドライブの動作仕様を設定します。

※ ORIGIN 関数は、これらのコマンドを使用して、従来製品と同様の機械原点検出ドライブを実現しています。

(1) ORIGIN SPEC SET

ORIGIN 停止機能を設定します。

<実行シーケンス>　COMMAND H'0F　ORIGIN SPEC SET COMMAND



① DRIVE STATUS1 PORT の ERROR フラグが "0" であることを確認します。

② DRIVE STATUS1 PORT の BUSY フラグが "0" であることを確認します。
推奨:MC07_BRStatus1 関数

③ DRIVE DATA1 PORT に設定データを書き込みます。

④ DRIVE COMMAND PORT に COMMAND を書き込みます。
推奨:MC07_LWDrive 関数(DATA2 PORT は、H'0000 を書き込み)

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
ORG DIVISION D7	ORG DIVISION D6	ORG DIVISION D5	ORG DIVISION D4	ORG DIVISION D3	ORG DIVISION D2	ORG DIVISION D1	ORG DIVISION D0

D7	D6	D5	D4	D3	D2	D1	D0
AUTO DRST ENABLE	ORG STOP TYPE1	ORG STOP TYPE0	ORG DETECT EDGE	ORG SIGNAL TYPE3	ORG SIGNAL TYPE2	ORG SIGNAL TYPE1	ORG SIGNAL TYPE0

● 電源投入後の初期値は H'0003 (アンダーライン側) です。

D3--0 ：ORG SIGNAL TYPE3--0

ORIGIN 停止機能で検出する信号（ORG 検出信号）を選択します。

TYPE3	TYPE2	TYPE1	TYPE0	ORG 検出信号	
0	0	0	0	ORG 信号	
0	0	0	1	ZPO 信号	
0	0	1	0	ORG 信号と ZPO 信号の AND (論理積)	＊
0	0	1	1	ORG 信号と ZPO 信号の OR (論理和)	＊
0	1	0	0	ORG 信号	
0	1	0	1	DEND 信号	
0	1	1	0	ORG 信号と DEND 信号の AND (論理積)	＊
0	1	1	1	ORG 信号と DEND 信号の OR (論理和)	＊
1	0	0	0	NORG 信号	
1	0	0	1	設定禁止	
1	0	1	0	SS0	
1	0	1	1	設定禁止	
1	1	0	0	設定禁止	
1	1	0	1	設定禁止	
1	1	1	0	設定禁止	
1	1	1	1	設定禁止	

＊：各信号のアクティブレベルを論理合成して、ORG 検出信号にします。

DEND 信号を ORG 検出信号に選択した場合は、
各信号の入力機能と ORIGIN 停止機能の両方が有効になります。

D4 ：ORG DETECT EDGE
ORG 検出信号の検出エッジを選択します。
0 ：ORG 検出信号の 0 → 1（アクティブ）エッジを検出する
1 ：ORG 検出信号の 1 → 0（OFF）エッジを検出する

D5 ：ORG STOP TYPE0
D6 ：ORG STOP TYPE1
検出エッジの検出条件と ORG エッジ信号の停止機能を選択します。

TYPE1	TYPE0	検出エッジの検出条件	ORG エッジ信号の停止機能	備考
0	0	常時検出する	停止しない	−
0	1	DRIVE = 1 のときに検出する	パルス出力を減速停止する	ORGEND = 1 になる SSEND フラグは変化しない
1	0	DRIVE = 1 のときに検出する	パルス出力を即時停止する	ORGEND = 1 になる FSEND フラグは変化しない
1	1	DRIVE = 1 のときに検出する	１パルス停止する	ORGEND = 1 になる FSEND フラグは変化しない

１パルス停止：
・１パルス停止機能は即時停止機能とほぼ同様です。１パルス停止機能を検出すると、
実行中のドライブパルス出力を完全に１周期出力してから、ドライブパルス出力を停止します。
・１パルス停止の後の連続ドライブでは、ドライブパルス周期の連続性を保つことが容易になります。

"01, 10, 11" に設定した場合は、検出と同時に STATUS2 PORT の ORGEND = 1 にします。

D7 ：AUTO DRST ENABLE
SPEC INITIALIZE3 コマンドの DRST TYPE を<サーボ対応>に設定している場合に有効です。
ORG エッジ信号の停止機能が動作（ORGEND = 1）して、ドライブパルス出力を終了した時に、
DRST 信号を「出力する／出力しない」を選択します。
0 ：DRST 信号を出力しない
1 ：DRST 信号を出力する（10 ms 間アクティブレベルにする）

D15--D8 ：ORG DIVISION D7--D0
ORG STOP TYPE の検出条件で検出したエッジ信号の分周数を選択します。
分周した信号が、最終検出の ORG エッジ信号になります。

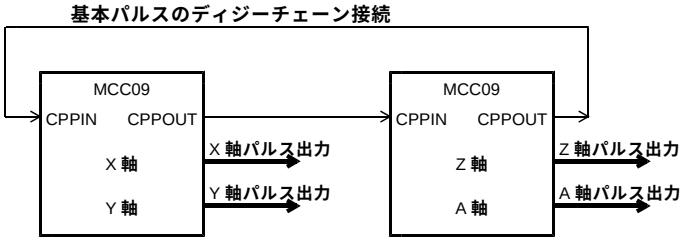
D7--D0	H'FF	H'FE	H'FD	〜	H'03	H'02	H'01	H'00
分周数	256	255	254	〜	4	3	2	1 (分周なし)

ORG DIVISION = H'00 ：１カウント毎のエッジ信号（ORG エッジ信号）を出力する
ORG DIVISION = H'01 ：２カウント毎のエッジ信号（ORG エッジ信号）を出力する
：
ORG DIVISION = H'FF ：256 カウント毎のエッジ信号（ORG エッジ信号）を出力する

以下の場合は、分周カウント値をクリアします。
・ORIGIN SPEC SET コマンドの実行
・ORG STOP TYPE = "01, 10, 11" に設定したときの DRIVE = 0 の状態

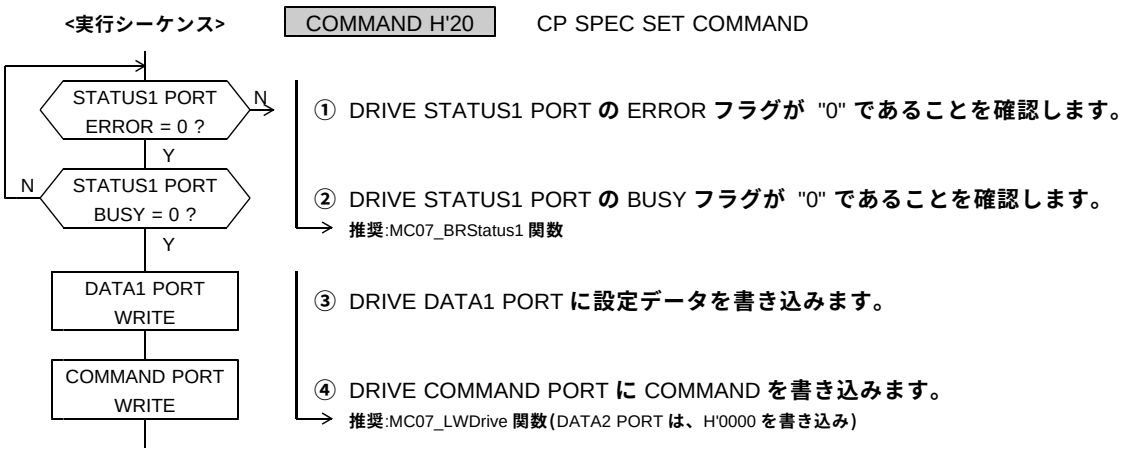
3-1-4. 補間ドライブの設定

各軸 MCC09 の CPPOUT 端子と CPPIN 端子はディジーチェーン接続で繋がっています。
任意軸補間ドライブではメイン軸のチップが CPPOUT 端子に補間ドライブの基本パルスを出力します。
サブ軸のチップは基本パルスを CPPIN 端子から入力して CPPOUT 端子に出力します。



(1) CP SPEC SET

補間ドライブの補間パルスの入出力機能を設定します。
このコマンドの設定は X(Z), Y(A)軸で共有します。X(Z), Y(A)のどの軸に設定しても有効です。



DRIVE DATA1 PORT の設定データ

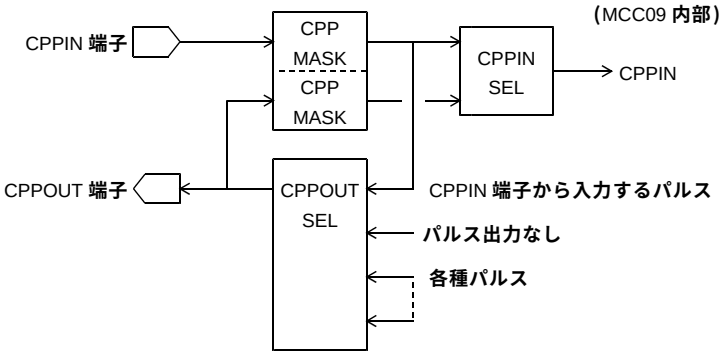
D15	D14	D13	D12	D11	D10	D9	D8
–	–	–	–	–	–	–	–

D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	CPPIN SEL	–	CPPOUT SEL2	CPPOUT SEL1	CPPOUT SEL0

● 電源投入後の初期値は H'01 (アンダーライン側) です。

D0 ：CPPOUT SEL0
D1 ：CPPOUT SEL1
D2 ：CPPOUT SEL2
CPPOUT 端子から出力する補間パルスを選択します。

SEL2	SEL1	SEL0	CPPOUT 端子から出力するパルス	
0	0	0	CPPIN 端子から入力するパルス	
0	0	1	パルス出力なし（ハイレベル出力）	
0	1	0	X(Z)軸の出力パルス（XOP）	
0	1	1	Y(A)軸の出力パルス（YOP）	
1	0	0	EA0, EB0 信号を変換したパルス	*2
1	0	1	EA1, EB1 信号を変換したパルス	*3
1	1	0	X 軸が発生する補間ドライブの基本パルス	*1
1	1	1	Y 軸が発生する補間ドライブの基本パルス	*1



*1：メイン軸直線補間ドライブを実行すると、直線補間ドライブの基本パルスを出力します。
メイン軸円弧補間ドライブを実行すると、円弧補間ドライブの基本パルスを出力します。
その他のドライブを実行する場合は、パルス出力なし（ハイレベル出力）にします。

*2：EA0, EB0 端子から入力する信号を、X 軸の外部パルス出力仕様の設定でパルス出力します。
外部パルス出力仕様は、X 軸の ADDRESS COUNTER INITIALIZE1 コマンドで設定します。
ただし、X 軸の COUNT PULSE SEL を EA0, EB0 または EA1, EB1 に設定している場合は、
X 軸の出力パルス（XOP：EA0, EB0 または EA1, EB1）を CPPOUT 端子から出力します。

*3：EA1, EB1 端子から入力する信号を、Y 軸の外部パルス出力仕様の設定でパルス出力します。
外部パルス出力仕様は、Y 軸の ADDRESS COUNTER INITIALIZE1 コマンドで設定します。
ただし、Y 軸の COUNT PULSE SEL を EA0, EB0 または EA1, EB1 に設定している場合は、
Y 軸の出力パルス（YOP：EA0, EB0 または EA1, EB1）を CPPOUT 端子から出力します。

D4 ：CPPIN SEL
MCC09 内部の補間パルス入力（CPPIN）に入力する補間パルスを選択します。
0 ：CPPIN 端子から入力するパルス
1 ：CPPOUT 端子から出力するパルス

3-1-5. 直線補間ドライブの設定と実行

※デバイスドライバによる補間関数は、これらのコマンドを使用して、座標指定による補間ドライブを実現しています。

メイン軸直線補間ドライブの実行軸には、加減速ドライブのパラメータを設定します。

直線補間ドライブでは、長軸パルスを補間ドライブの基本パルスとし、
短軸側は基本パルスを補間演算して補間パルスを出力します。

現在位置を座標中心（0, 0）とした長軸と短軸の相対アドレスを、座標アドレスとします。
座標アドレスは、正数が ＋(CW)方向、負数が －(CCW)方向です。

長軸と短軸の座標アドレスとドライブ仕様を指定して、直線補間ドライブを実行します。

■ 直線補間ドライブのドライブ仕様

ドライブ仕様は、直線補間ドライブのコマンド実行時に指定します。
・メイン軸は、長軸と短軸のパルス比で基本パルスと補間パルスを発生し、
指定した座標アドレスの補間パルスを出力します。
・サブ軸は、長軸と短軸のパルス比で補間パルスを発生し、
指定した座標アドレスの補間パルスを出力します。

D0 ：DRIVE MODE
直線補間ドライブを「連続ドライブにする／位置決めドライブにする」を選択します。

● 直線補間 SCAN ドライブ
停止指令を検出するまで、連続して補間ドライブを行います。
停止指令を検出すると、メイン軸は、基本パルスと補間パルス出力を停止します。
サブ軸は、補間パルス出力を即時停止します。

● 直線補間 INDEX ドライブ
基本パルスをカウントして、カウント値が長軸の座標アドレスのパルス数と一致すると、
補間ドライブを終了します。

D1 ：CONST CP ENABLE
メイン軸直線補間ドライブで有効です。
線速一定制御を「無効にする／有効にする」を選択します。

● 線速一定制御
直線補間ドライブしている２軸の合成速度を一定にする制御です。

メイン軸が発生する補間ドライブの基本パルスを線速一定制御します。
メイン軸の長軸と短軸の２軸間で、２軸同時にパルス出力したときに、
次の基本パルスの出力周期を 1.414 倍にします。

線速一定で加減速ドライブを行うと、減速後の終了速度でのドライブが長くなります。

D2 ：CPP STOP ENABLE
メイン軸直線補間ドライブで有効です。
メイン軸の CPP STOP 機能を「有効にする／無効にする」を選択します。

● メイン軸の CPP STOP 機能
メイン軸補間ドライブ実行中に機能します。
・メイン軸の CPP STOP 機能を有効にすると、
補間ドライブの基本パルスと CPPIN に入力するパルスを偏差カウントします。
・CPPIN のパルス数が、メイン軸の基本パルス数より２パルス分少なくなると、
補間ドライブの基本パルス出力を停止して、メイン軸のドライブを終了します。

・CPP STOP 機能でドライブを終了した場合は、
メイン軸の ERROR STATUS の CPP STOP ERROR = 1 にします。

・メイン軸は CPP STOP 機能でドライブを終了しますが、サブ軸はドライブを終了しません。
メイン軸が CPP STOP 機能でドライブを終了した場合は、
すべてのサブ軸に停止指令を実行して、ドライブを終了させてください。

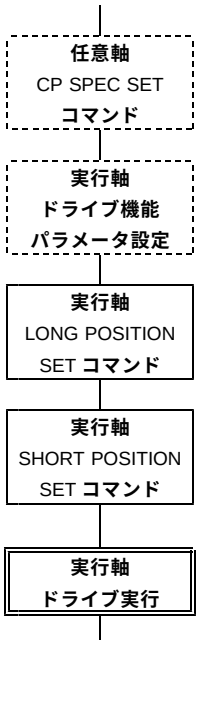
D2 ：CPP MASK ENABLE
サブ軸直線補間ドライブで有効です。
サブ軸の CPPIN **マスク機能を「有効にする／無効にする」を選択します。**

● **サブ軸の** CPPIN **マスク機能**
サブ軸補間ドライブ実行中に機能します。
・**サブ軸の** CPPIN **マスク機能を有効にすると、**
ERROR = 1 **になると、**MCC09 **内部の** CPPIN **に入力するパルスをマスクします。**
・**出力中の補間パルスが** OFF **レベルのときにマスクします。**
出力中の補間パルスのアクティブレベルが続く場合は、
ERROR = 1 **から** 100 µs **後に、補間パルス出力を** OFF **レベルにしてマスクします。**

・CPPOUT SEL **で** CPPOUT **出力を「**CPPIN **端子から入力するパルス」に設定している場合は、**
CPPIN **のマスクにより、**CPPOUT **出力はハイレベル状態になります。**

・CPPIN **マスク機能で** CPPIN **をマスク中は、**STATUS5 PORT **の** CPP MASK = 1 **にします。**
ERROR = 0 **にクリアすると、**CPP MASK = 0 **にします。**

■ メイン軸直線補間ドライブの実行シーケンス



① 任意軸に、CPPOUT 出力と CPPIN 入力を設定します。

② 実行軸に、加減速ドライブに必要なパラメータを設定します。

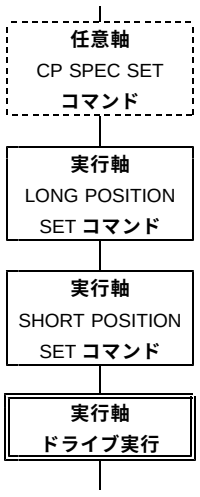
③ 実行軸に、長軸の座標アドレスを設定します。

④ 実行軸に、短軸の座標アドレスを設定します。

⑤ 実行軸に、ドライブ仕様を指定して、ドライブを実行します。

初期値および設定値に対して変更が必要な場合に設定します。

■ サブ軸直線補間ドライブの実行シーケンス



① 任意軸に、CPPOUT 出力と CPPIN 入力を設定します。

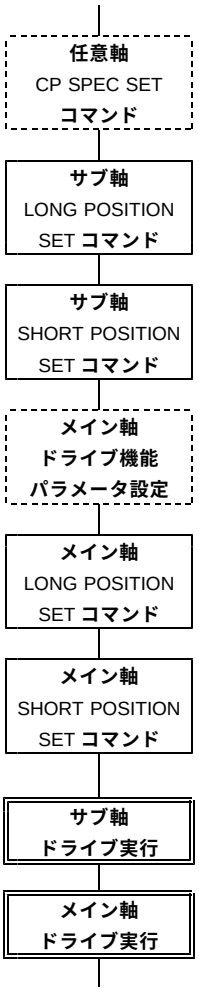
② 実行軸に、長軸の座標アドレスを設定します。

③ 実行軸に、短軸の座標アドレスを設定します。

④ 実行軸に、ドライブ仕様を指定して、ドライブを実行します。

初期値および設定値に対して変更が必要な場合に設定します。

■ １チップ直線補間ドライブの実行シーケンス



① 任意軸に、CPPOUT 出力と CPPIN 入力を設定します。
CPPOUT SEL =「メイン軸が発生する基本パルス」に設定します。
CPPIN SEL =「CPPOUT 端子から出力するパルス」に設定します。

② サブ軸に、長軸の座標アドレスを設定します。

③ サブ軸に、短軸の座標アドレスを設定します。

④ メイン軸に、加減速ドライブに必要なパラメータを設定します。

⑤ メイン軸に、長軸の座標アドレスを設定します。

⑥ メイン軸に、短軸の座標アドレスを設定します。

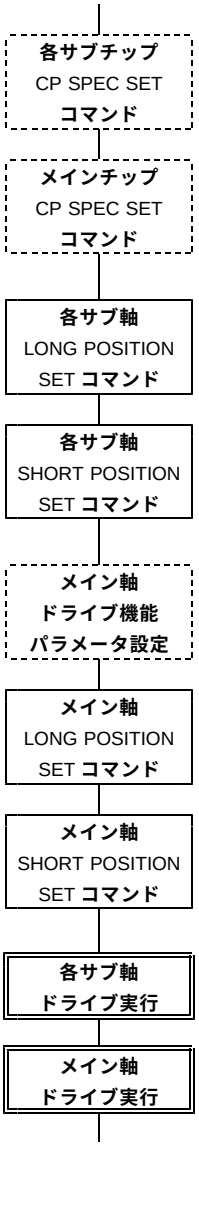
⑦ サブ軸に、サブ軸直線補間ドライブを実行します。

⑧ メイン軸に、メイン軸直線補間ドライブを実行します。

初期値および設定値に対して変更が必要な場合に設定します。

サブ軸直線補間ドライブを実行すると、ドライブがスタンバイ状態になります。
メイン軸直線補間ドライブを実行すると、基本パルスを出力して、補間ドライブを開始します。
サブ軸は、CPPIN に入力するパルスを基本パルスにして、補間ドライブを開始します。

■ マルチチップ直線補間ドライブの実行シーケンス



① 各サブ軸のチップに、CPPOUT 出力と CPPIN 入力を設定します。
CPPOUT SEL =「CPPIN 端子から入力するパルス」に設定します。
CPPIN SEL =「CPPIN 端子から入力するパルス」に設定します。

② メイン軸のチップに、CPPOUT 出力と CPPIN 入力を設定します。
CPPOUT SEL =「メイン軸が発生する基本パルス」に設定します。
CPPIN SEL =「CPPIN 端子から入力するパルス」に設定します。

③ 各サブ軸に、長軸の座標アドレスを設定します。

④ 各サブ軸に、短軸の座標アドレスを設定します。

⑤ メイン軸に、加減速ドライブに必要なパラメータを設定します。

⑥ メイン軸に、長軸の座標アドレスを設定します。

⑦ メイン軸に、短軸の座標アドレスを設定します。

⑧ 各サブ軸に、サブ軸直線補間ドライブを実行します。

⑨ メイン軸に、メイン軸直線補間ドライブを実行します。

初期値および設定値に対して変更が必要な場合に設定します。

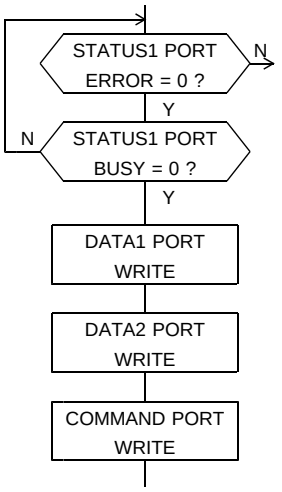
MCC09 の CPPIN 端子と CPPOUT 端子をデイジーチェーン接続します。

サブ軸直線補間ドライブを実行すると、ドライブがスタンバイ状態になります。
メイン軸直線補間ドライブを実行すると、基本パルスを出力して、補間ドライブを開始します。
各サブ軸は、CPPIN 端子から入力するパルスを基本パルスとして、補間ドライブを開始します。

(1) LONG POSITION SET

直線補間ドライブの、長軸の座標アドレスを設定します。

<実行シーケンス>　COMMAND H'22　LONG POSITION SET COMMAND



① DRIVE STATUS1 PORT の ERROR フラグが "0" であることを確認します。

② DRIVE STATUS1 PORT の BUSY フラグが "0" であることを確認します。
推奨:MC07_BRStatus1 関数

③ DRIVE DATA1 PORT に長軸の目的地の座標アドレスを書き込みます。

④ DRIVE DATA2 PORT に長軸の目的地の座標アドレスを書き込みます。
設定範囲は、
-2,147,483,648 ～+2,147,483,647（H'8000_0000 ～ H'7FFF_FFFF）です。
正数が CW、負数が CCW 方向です。負数の場合は 2 の補数表現にします。

⑤ DRIVE COMMAND PORT に COMMAND を書き込みます。
推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A15	←				長軸の目的地の座標アドレス									→	A0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A31	←				長軸の目的地の座標アドレス									→	A16

●電源投入後の初期値は H'0000_0000 です。

指定する座標アドレスは、現在位置を座標中心（0, 0）とした相対アドレスです。

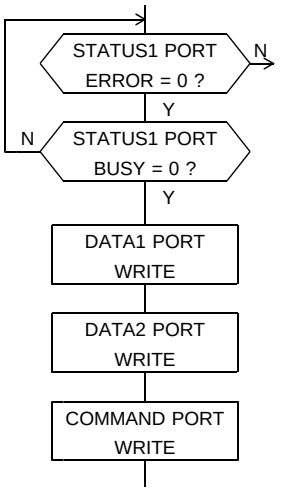
「長軸の目的地の座標アドレス」には、
各補間軸の中で補間パルス数が最も大きい補間軸（長軸）の目的地を設定します。

ドライブ実行コマンドの PULSE SEL で指定した軸（長軸／短軸）の座標アドレスの符号が、
出力する補間パルスの動作方向になります。

(2) SHORT POSITION SET

直線補間ドライブの、短軸の座標アドレスを設定します。

<実行シーケンス>　COMMAND H'23　SHORT POSITION SET COMMAND



① DRIVE STATUS1 PORT の ERROR フラグが "0" であることを確認します。

② DRIVE STATUS1 PORT の BUSY フラグが "0" であることを確認します。
→ 推奨:MC07_BRStatus1 関数

③ DRIVE DATA1 PORT に短軸の目的地の座標アドレスを書き込みます。

④ DRIVE DATA2 PORT に短軸の目的地の座標アドレスを書き込みます。
設定範囲は、
-2,147,483,648 ～+2,147,483,647（H'8000_0000 ～ H'7FFF_FFFF）です。
正数が CW、負数が CCW 方向です。負数の場合は 2 の補数表現にします。

⑤ DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A15	←				短軸の目的地の座標アドレス									→	A0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A31	←				短軸の目的地の座標アドレス									→	A16

●電源投入後の初期値は H'0000_0000 です。

指定する座標アドレスは、現在位置を座標中心（0, 0）とした相対アドレスです。

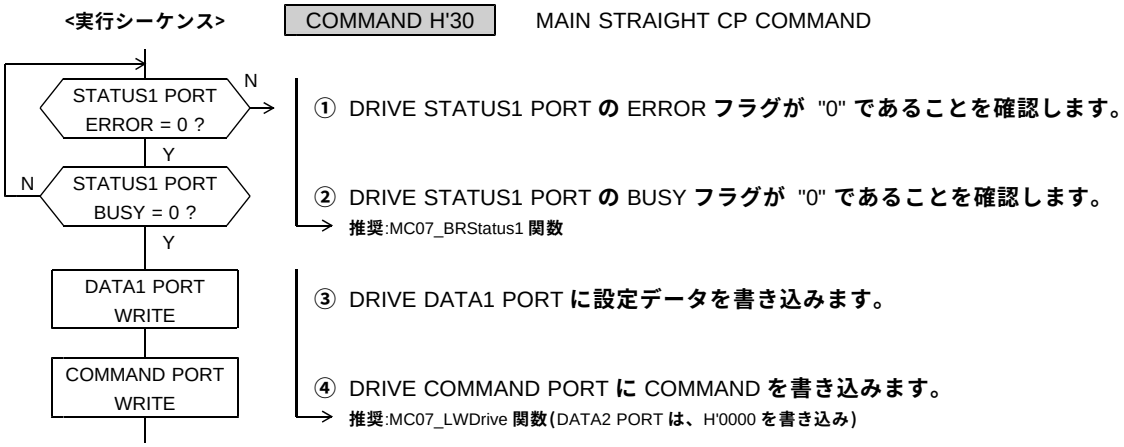
「短軸の目的地の座標アドレス」には、短軸の目的地（符号付きアドレス）を設定します。
・「長軸の座標アドレスの絶対値≧短軸の座標アドレスの絶対値」に設定します。

ドライブ実行コマンドの PULSE SEL で指定した軸（長軸／短軸）の座標アドレスの符号が、
出力する補間パルスの動作方向になります。

(3) MAIN STRAIGHT CP

1 軸単位で直線補間ドライブを行うコマンドです。
任意軸間の直線補間、または複数軸で直線補間ドライブさせるときにメイン軸に実行します。
メイン軸の直線補間ドライブは実行軸の加減速パラメータで動作します。

<実行シーケンス>



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
–	–	–	–	–	–	–	–

D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	PULSE SEL	–	CPP STOP ENABLE	CONST CP ENABLE	DRIVE MODE

D0 ：DRIVE MODE
直線補間ドライブを「連続ドライブにする／位置決めドライブにする」を選択します。
0 ：連続ドライブにする (SCAN ドライブ)
1 ：位置決めドライブにする (INDEX ドライブ)

D1 ：CONST CP ENABLE
線速一定制御を「無効にする／有効にする」を選択します。
0 ：線速一定制御を無効にする
1 ：線速一定制御を有効にする

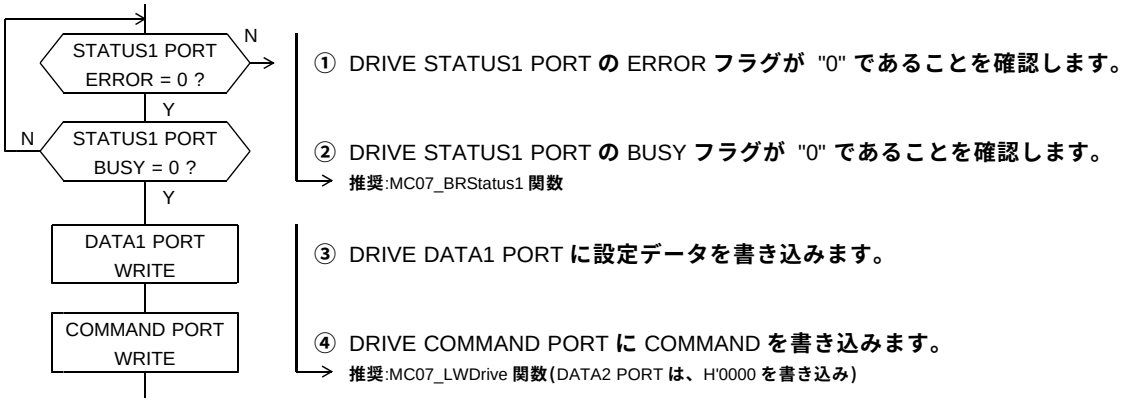
D2 ：CPP STOP ENABLE
CPP STOP 機能を「有効にする／無効にする」を選択します。
0 ：CPP STOP 機能を無効にする
1 ：CPP STOP 機能を有効にする

D4 ：PULSE SEL
出力する補間パルスを選択します。
0 ：LONG POSITION（長軸）の補間パルスを出力する
1 ：SHORT POSITION（短軸）の補間パルスを出力する

(4) SUB STRAIGHT CP

1 軸単位で直線補間ドライブを行うコマンドです。
任意軸間の直線補間、または複数軸で直線補間ドライブさせるときにサブ軸に実行します。
サブ軸の直線補間ドライブは CPPIN 入力パルスで動作します。

<実行シーケンス>　COMMAND H'31　SUB STRAIGHT CP COMMAND



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
–	–	–	–	–	–	–	–

D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	PULSE SEL	–	CPP MASK ENABLE	–	DRIVE MODE

D0 ：DRIVE MODE
直線補間ドライブを「連続ドライブにする／位置決めドライブにする」を選択します。
0 ：連続ドライブにする　(SCAN ドライブ)
1 ：位置決めドライブにする　(INDEX ドライブ)

D2 ：CPP MASK ENABLE
CPPIN マスク機能を「有効にする／無効にする」を選択します。
0 ：CPPIN マスク機能を無効にする
1 ：CPPIN マスク機能を有効にする

D4 ：PULSE SEL
出力する補間パルスを選択します。
0 ：LONG POSITION（長軸）の補間パルスを出力する
1 ：SHORT POSITION（短軸）の補間パルスを出力する

3-1-6. 円弧補間ドライブの設定と実行

メイン軸円弧補間ドライブの実行軸には、加減速ドライブのパラメータを設定します。

円弧補間ドライブでは、
円弧の中心座標からみた短軸側の短軸パルスを補間ドライブの基本パルスとし、
長軸側は基本パルスを補間演算して補間パルスを出力します。

円弧の中心点座標を（0, 0）とした X 軸と Y 軸の相対アドレスを、X-Y 座標アドレスとします。
座標アドレスは、正数が ＋(CW)方向、負数が －(CCW)方向です。

現在位置の X-Y 座標アドレスと、目的地の短軸座標までの短軸パルス数と、ドライブ仕様を
指定して、円弧補間ドライブを実行します。

■ 円弧補間ドライブのドライブ仕様

ドライブ仕様は、円弧補間ドライブのコマンド実行時に指定します。
・メイン軸は、指定の円弧半径と回転方向で基本パルスと補間パルスを発生し、
　指定した座標アドレスの補間パルスを出力します。
・サブ軸は、指定の円弧半径と回転方向で補間パルスを発生し、
　指定した座標アドレスの補間パルスを出力します。

D0 ：DRIVE MODE
円弧補間ドライブを「連続ドライブにする／位置決めドライブにする」を選択します。

● 円弧補間 SCAN ドライブ
停止指令を検出するまで、連続して補間ドライブを行います。
停止指令を検出すると、メイン軸は、基本パルスと補間パルス出力を停止します。
サブ軸は、補間パルス出力を即時停止します。

● 円弧補間 INDEX ドライブ
基本パルスをカウントして、カウント値が短軸パルス数の設定値と一致すると、
補間ドライブを終了します。

D1 ：CONST CP ENABLE
メイン軸円弧補間ドライブで有効です。
線速一定制御を「無効にする／有効にする」を選択します。

● 線速一定制御
補間ドライブしている２軸の合成速度を一定にする制御です。

メイン軸が発生する補間ドライブの基本パルスを線速一定制御します。
X 座標軸と Y 座標軸の２軸間で、２軸同時にパルス出力したときに、
次の基本パルスの出力周期を 1.414 倍にします。

線速一定で加減速ドライブを行うと、減速後の終了速度でのドライブが長くなります。

D2 ：CPP STOP ENABLE
メイン軸円弧補間ドライブで有効です。
メイン軸の CPP STOP 機能を「有効にする／無効にする」を選択します。

● メイン軸の CPP STOP 機能
メイン軸補間ドライブ実行中に機能します。
・メイン軸の CPP STOP 機能を有効にすると、
　補間ドライブの基本パルスと CPPIN に入力するパルスを偏差カウントします。
・CPPIN のパルス数が、メイン軸の基本パルス数より２パルス分少なくなると、
　補間ドライブの基本パルス出力を停止して、メイン軸のドライブを終了します。
・CPP STOP 機能でドライブを終了した場合は、
　メイン軸の ERROR STATUS の CPP STOP ERROR = 1 にします。
・メインは CPP STOP 機能でドライブを終了しますが、サブ軸はドライブを終了しません。
　メイン軸が CPP STOP 機能でドライブを終了した場合は、
　すべてのサブ軸に停止指令を実行して、ドライブを終了させてください。

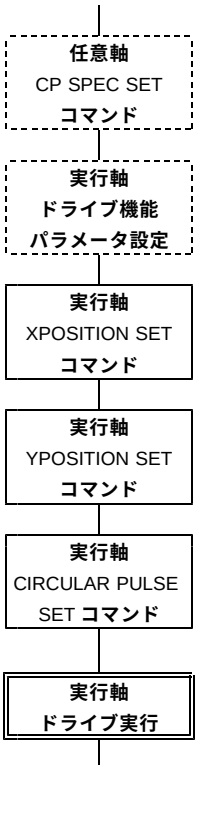
D2 ：CPP MASK ENABLE
サブ軸円弧補間ドライブで有効です。
サブ軸の CPPIN **マスク機能を「有効にする／無効にする」を選択します。**

● **サブ軸の** CPPIN **マスク機能**
サブ軸補間ドライブ実行中に機能します。
・**サブ軸の** CPPIN **マスク機能を有効にすると、**
ERROR = 1 **になると、**MCC09 **内部の** CPPIN **に入力するパルスをマスクします。**
・**出力中の補間パルスが** OFF **レベルのときにマスクします。**
出力中の補間パルスのアクティブレベルが続く場合は、
ERROR = 1 **から** 100 µs **後に、補間パルス出力を** OFF **レベルにしてマスクします。**

・CPPOUT SEL **で** CPPOUT **出力を「**CPPIN **端子から入力するパルス」に設定している場合は、**
CPPIN **のマスクにより、**CPPOUT **出力はハイレベル状態になります。**

・CPPIN **マスク機能で** CPPIN **をマスク中は、**STATUS5 PORT **の** CPP MASK = 1 **にします。**
ERROR = 0 **にクリアすると、**CPP MASK = 0 **にします。**

■ メイン軸円弧補間ドライブの実行シーケンス



① 任意軸に、CPPOUT 出力と CPPIN 入力を設定します。

② 実行軸に、加減速ドライブに必要なパラメータを設定します。

③ 実行軸に、現在位置の X 座標アドレスを設定します。

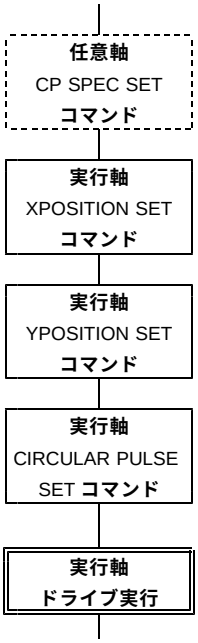
④ 実行軸に、現在位置の Y 座標アドレスを設定します。

⑤ 実行軸に、目的地の短軸座標までのパルス数を設定します。

⑥ 実行軸に、ドライブ仕様を指定して、ドライブを実行します。

初期値および設定値に対して変更が必要な場合に設定します。

■ サブ軸円弧補間ドライブの実行シーケンス



① 任意軸に、CPPOUT 出力と CPPIN 入力を設定します。

② 実行軸に、現在位置の X 座標アドレスを設定します。

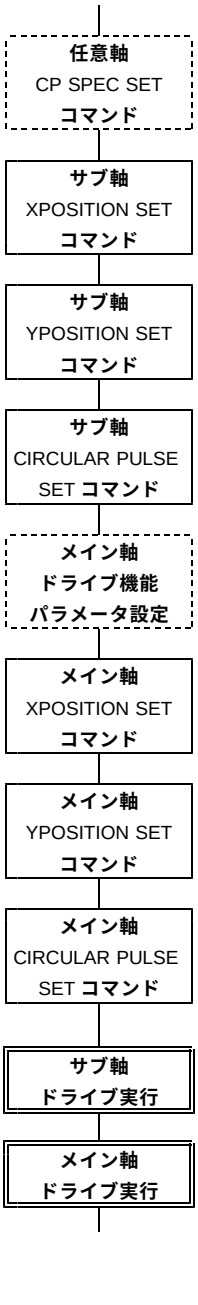
③ 実行軸に、現在位置の Y 座標アドレスを設定します。

④ 実行軸に、目的地の短軸座標までのパルス数を設定します。

⑤ 実行軸に、ドライブ仕様を指定して、ドライブを実行します。

初期値および設定値に対して変更が必要な場合に設定します。

■ １チップ２軸円弧補間ドライブの実行シーケンス



① 任意軸に、CPPOUT 出力と CPPIN 入力を設定します。
CPPOUT SEL =「メイン軸が発生する基本パルス」に設定します。
CPPIN SEL =「CPPOUT 端子から出力するパルス」に設定します。

② サブ軸に、現在位置の X 座標アドレスを設定します。

③ サブ軸に、現在位置の Y 座標アドレスを設定します。

④ サブ軸に、目的地の短軸座標までのパルス数を設定します。

⑤ メイン軸に、加減速ドライブに必要なパラメータを設定します。

⑥ メイン軸に、現在位置の X 座標アドレスを設定します。

⑦ メイン軸に、現在位置の Y 座標アドレスを設定します。

⑧ メイン軸に、目的地の短軸座標までのパルス数を設定します。

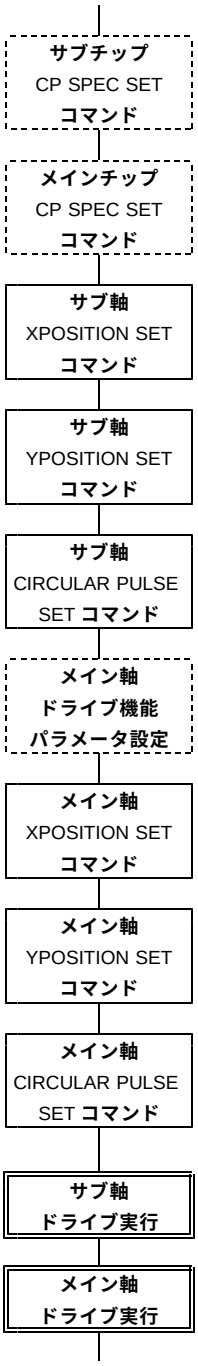
⑨ サブ軸に、サブ軸円弧補間ドライブを実行します。

⑩ メイン軸に、メイン軸円弧補間ドライブを実行します。

初期値および設定値に対して変更が必要な場合に設定します。

サブ軸円弧補間ドライブを実行すると、ドライブがスタンバイ状態になります。
メイン軸円弧補間ドライブを実行すると、基本パルスを出力して、補間ドライブを開始します。
サブ軸は、CPPIN に入力するパルスを基本パルスにして、補間ドライブを開始します。

■ ２チップ２軸円弧補間ドライブの実行シーケンス



① サブ軸のチップに、CPPOUT 出力と CPPIN 入力を設定します。
CPPOUT SEL =「CPPIN 端子から入力するパルス」に設定します。
CPPIN SEL =「CPPIN 端子から入力するパルス」に設定します。

② メイン軸のチップに、CPPOUT 出力と CPPIN 入力を設定します。
CPPOUT SEL =「メイン軸が発生する基本パルス」に設定します。
CPPIN SEL =「CPPIN 端子から入力するパルス」に設定します。

③ サブ軸に、現在位置の X 座標アドレスを設定します。

④ サブ軸に、現在位置の Y 座標アドレスを設定します。

⑤ サブ軸に、目的地の短軸座標までのパルス数を設定します。

⑥ メイン軸に、加減速ドライブに必要なパラメータを設定します。

⑦ メイン軸に、現在位置の X 座標アドレスを設定します。

⑧ メイン軸に、現在位置の Y 座標アドレスを設定します。

⑨ メイン軸に、目的地の短軸座標までのパルス数を設定します。

⑩ サブ軸に、サブ軸円弧補間ドライブを実行します。

⑪ メイン軸に、メイン軸円弧補間ドライブを実行します。

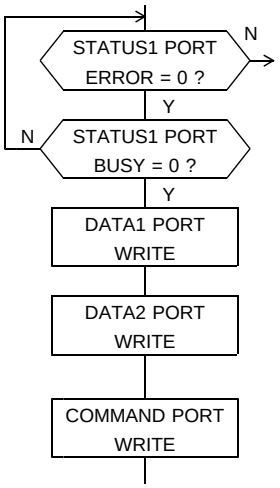
初期値および設定値に対して変更が必要な場合に設定します。

サブ軸円弧補間ドライブを実行すると、ドライブがスタンバイ状態になります。
メイン軸円弧補間ドライブを実行すると、基本パルスを出力して、補間ドライブを開始します。
サブ軸は、CPPIN 端子から入力するパルスを基本パルスとして、補間ドライブを開始します。

(1) CIRCULAR XPOSITION SET

円弧の中心点座標を（0, 0）とした現在位置の X 座標アドレスを設定します。
実行軸のどちらの軸に設定しても有効です。

<実行シーケンス>



COMMAND H'28　CIRCULAR XPOSITION SET COMMAND

① DRIVE STATUS1 PORT の ERROR フラグが "0" であることを確認します。

② DRIVE STATUS1 PORT の BUSY フラグが "0" であることを確認します。
推奨:MC07_BRStatus1 関数

③ DRIVE DATA1 PORT に現在位置の X 座標アドレスを書き込みます。

④ DRIVE DATA2 PORT に現在位置の X 座標アドレスを書き込みます。
設定範囲は、
-8,388,608 ～+8,388,607（H'80_0000 ～ H'7F_FFFF）です。
正数が CW、負数が CCW 方向です。負数の場合は 2 の補数表現にします。

⑤ DRIVE COMMAND PORT に COMMAND を書き込みます。
推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A15	←				現在位置の X 座標アドレス									→	A0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	–	–	–	–	–	A23	←	現在位置の X 座標アドレス				→	A16

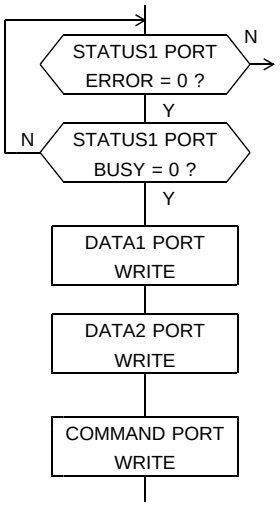
●電源投入後の初期値は H'00_0000 です。

・指定する座標アドレスは、円弧の中心点座標を（0, 0）とした X 軸の相対アドレスです。

(2) CIRCULAR YPOSITION SET

円弧の中心点座標を（0, 0）とした現在位置の Y 座標アドレスを設定します。
実行軸のどちらの軸に設定しても有効です。

<実行シーケンス>　COMMAND H'29　CIRCULAR YPOSITION SET COMMAND



① DRIVE STATUS1 PORT の ERROR フラグが "0" であることを確認します。

② DRIVE STATUS1 PORT の BUSY フラグが "0" であることを確認します。
推奨:MC07_BRStatus1 関数

③ DRIVE DATA1 PORT に現在位置の Y 座標アドレスを書き込みます。

④ DRIVE DATA2 PORT に現在位置の Y 座標アドレスを書き込みます。
設定範囲は、
-8,388,608 ～+8,388,607（H'80_0000 ～ H'7F_FFFF）です。
正数が CW、負数が CCW 方向です。負数の場合は 2 の補数表現にします。

⑤ DRIVE COMMAND PORT に COMMAND を書き込みます。
推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A15	←							現在位置の Y 座標アドレス						→	A0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	–	–	–	–	–	A23	←	現在位置の Y 座標アドレス				→	A16

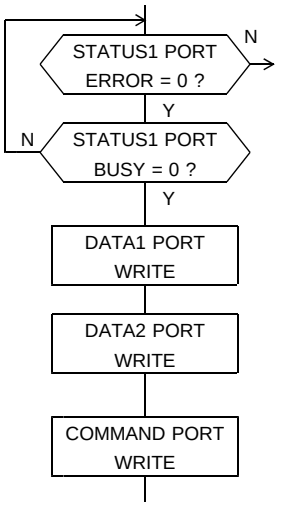
●電源投入後の初期値は H'00_0000 です。

・指定する座標アドレスは、円弧の中心点座標を（0, 0）とした Y 軸の相対アドレスです。

(3) CIRCULAR PULSE SET

現在位置の X-Y 座標アドレスから目的地の短軸座標までの短軸パルス数を設定します。
実行軸のどちらの軸に設定しても有効です。

<実行シーケンス>



① DRIVE STATUS1 PORT の ERROR フラグが "0" であることを確認します。

② DRIVE STATUS1 PORT の BUSY フラグが "0" であることを確認します。
→ 推奨:MC07_BRStatus1 関数

③ DRIVE DATA1 PORT に目的地の短軸座標までのパルス数を書き込みます。

④ DRIVE DATA2 PORT に目的地の短軸座標までのパルス数を書き込みます。
設定範囲は、
-2,147,483,648 ～+2,147,483,647（H'8000_0000 ～ H'7FFF_FFFF）です。
正数が CW、負数が CCW 方向です。負数の場合は 2 の補数表現にします。

⑤ DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D15 ←					目的地の短軸座標までのパルス数									→	D0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D31 ←					目的地の短軸座標までのパルス数									→	D16

●電源投入後の初期値は H'0000_0000 (0 パルス) です。

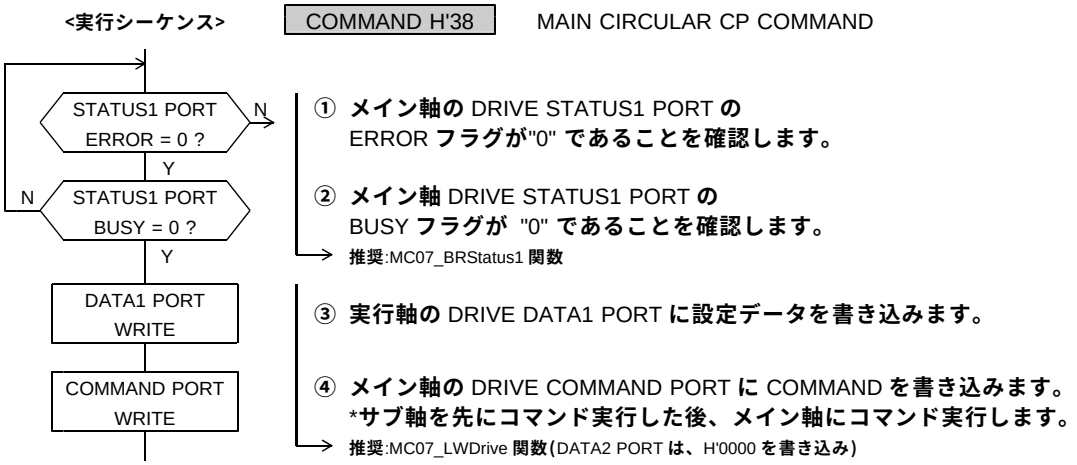
指定するパルス数は、
目的地の短軸座標に到達するまでに経由する、各象限の短軸の合計パルス数です。
・短軸パルス数の計算式は、「4-1-5.(4)円弧補間ドライブ」をご覧ください。
デバイスドライバの円弧補間短軸 PULSE 数ゲット関数を使用すると
指定された円弧の中心点相対アドレス、目的地相対アドレス、回転方向をもとに
目的地の短軸座標までのパルス数を算出します。

円弧を描く回転方向は、パルス数の符号で指定します。
・正数を指定すると CW 方向に回転します。
・負数を指定すると CCW 方向に回転します。

(4) MAIN CIRCULAR CP

1 軸単位で円弧補間ドライブを行うコマンドです。
任意軸間の円弧補間ドライブさせるときにメイン軸に実行します。
メイン軸の加減速パラメータで発生するパルスを基本パルスにして動作します。

<実行シーケンス>



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
–	–	–	–	–	–	–	–

D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	PULSE SEL	–	CPP STOP ENABLE	CONST CP ENABLE	DRIVE MODE

D0 ：DRIVE MODE
円弧補間ドライブを「連続ドライブにする／位置決めドライブにする」を選択します。
0 ：連続ドライブにする (SCAN ドライブ)
1 ：位置決めドライブにする (INDEX ドライブ)

D1 ：CONST CP ENABLE
線速一定制御を「無効にする／有効にする」を選択します。
0 ：線速一定制御を無効にする
1 ：線速一定制御を有効にする

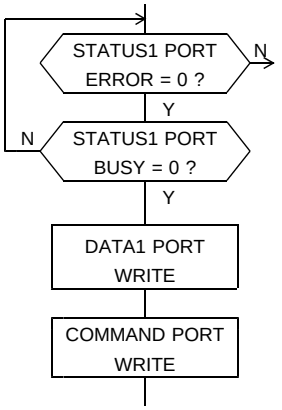
D2 ：CPP STOP ENABLE
CPP STOP 機能を「有効にする／無効にする」を選択します。
0 ：CPP STOP 機能を無効にする
1 ：CPP STOP 機能を有効にする

D4 ：PULSE SEL
出力する補間パルスを選択します。
0 ：円弧補間演算の X 座標アドレスの補間パルス（XCP）を出力する
1 ：円弧補間演算の Y 座標アドレスの補間パルス（YCP）を出力する

(5) SUB CIRCULAR CP

1 軸単位で円弧補間ドライブを行うコマンドです。
任意軸間の円弧補間ドライブさせるときにサブ軸に実行します。
サブ軸は、CPPIN 入力パルスを基本パルスにして動作します。

<実行シーケンス>　COMMAND H'39　SUB CIRCULAR CP COMMAND



① サブ軸の DRIVE STATUS1 PORT の
ERROR フラグが "0" であることを確認します。

② サブ軸の DRIVE STATUS1 PORT の
BUSY フラグが "0" であることを確認します。

推奨:MC07_BRStatus1 関数

③ 実行軸の DRIVE DATA1 PORT に設定データを書き込みます。

④ サブ軸の DRIVE COMMAND PORT に COMMAND を書き込みます。
*サブ軸を先にコマンド実行した後、メイン軸にコマンド実行します。

推奨:MC07_LWDrive 関数(DATA2 PORT は、H'0000 を書き込み)

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
–	–	–	–	–	–	–	–

D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	PULSE SEL	–	CPP MASK ENABLE	–	DRIVE MODE

D0 ：DRIVE MODE
円弧補間ドライブを「連続ドライブにする／位置決めドライブにする」を選択します。
0 ：連続ドライブにする　(SCAN ドライブ)
1 ：位置決めドライブにする　(INDEX ドライブ)

D2 ：CPP MASK ENABLE
CPPIN マスク機能を「有効にする／無効にする」を選択します。
0 ：CPPIN マスク機能を無効にする
1 ：CPPIN マスク機能を有効にする

D4 ：PULSE SEL
出力する補間パルスを選択します。
0 ：円弧補間演算の X 座標アドレスの補間パルス（XCP）を出力する
1 ：円弧補間演算の Y 座標アドレスの補間パルス（YCP）を出力する

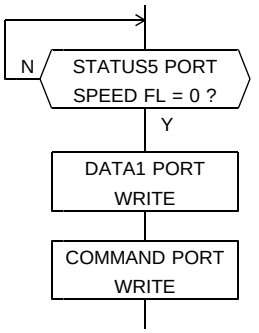
3-1-7. SPEED CHANGE の設定と実行

変更動作点を設定して、SPEED CHANGE を実行します。
変更動作点の設定は、変更動作点の変更が必要な場合に設定します。

(1) SPEED CHANGE SPEC SET

SPEED CHANGE 指令を実行する変更動作点を設定します。

<実行シーケンス>　COMMAND H'C1　SPEED CHANGE SPEC SET COMMAND



① DRIVE STATUS5 PORT の SPEED FL フラグが "0" であることを確認します。
→ MC07_BRStatus5 関数

② DRIVE DATA1 PORT に設定データを書き込みます。

③ DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数(DATA2 PORT は、H'0000 を書き込み)

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
–	–	–	–	–	–	–	–

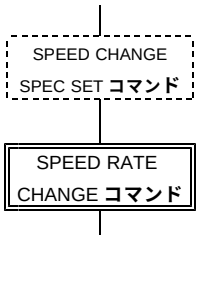
D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	–	–	SPEED CHANGE TYPE2	SPEED CHANGE TYPE1	SPEED CHANGE TYPE0

●電源投入後の初期値は H'0 (アンダーライン側) です。

D2--D0 ：SPEED CHANGE TYPE2--0
SPEED CHANGE 指令を実行する変更動作点を選択します。

D2	D1	D0	SPEED CHANGE を実行する変更動作点	検出仕様
0	0	0	STATUS1 PORT の DRIVE = 1 で実行する	レベル検出
0	0	1	DRIVE = 1 のときに、他軸の STATUS3 PORT の OUT3 = 0 → 1 で実行する	エッジ検出
0	1	0	DRIVE = 1 のときに、STATUS3 PORT の GPIO2 = 0 → 1 で実行する	エッジ検出
0	1	1	DRIVE = 1 のときに、ORIGIN 停止機能の ORG エッジ信号の検出で実行する	エッジ検出
1	0	0	DRIVE = 1 のときに、STATUS3 PORT の OUT2 = 0 → 1 で実行する	エッジ検出
1	0	1	DRIVE = 1 のときに、STATUS3 PORT の OUT3 = 0 → 1 で実行する	エッジ検出
1	1	0	設定禁止	–
1	1	1	設定禁止	–

■ SPEED CHANGE の実行シーケンス



① SPEED CHANGE 指令を実行する変更動作点を設定します。

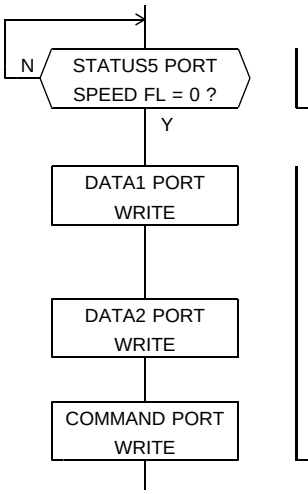
② SPEED RATE CHANGE 指令を実行（設定）します。

初期値および設定値に対して変更が必要な場合に設定します。

(2) SPEED RATE CHANGE

変更動作点の検出で、SPEED RATE CHANGE 指令を実行します。

<実行シーケンス>　COMMAND H'C8　SPEED RATE CHANGE COMMAND



① DRIVE STATUS5 PORT の SPEED FL フラグが "0" であることを確認します。
→ MC07_BRStatus5 関数

② DRIVE DATA1 PORT に変更速度データを書き込みます。
変更速度データの設定範囲は、0 ～ 32,767（H'0000 ～ H'7FFF）です。
速度倍率を 400 に設定した場合は、
変更速度データの設定範囲は、0 ～ 25,000（H'0000 ～ H'61A8）です。

③ DRIVE DATA2 PORT に変速周期データを書き込みます。
変速周期データの設定範囲は、0 ～ 16,383（H'0000 ～ H'3FFF）です。

④ DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	D14	←				SPEED CHANGE データ								→	D0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	–	D13	←			RATE CHANGE データ								→	D0

SPEED CHANGE データの設定値を "0" にすると、CONST DRIVE 指令になります。
・SPEED CHANGE の速度（Hz）＝ SPEED CHANGE データ x RESOL ：1 ～ 10,000,000 Hz

RATE CHANGE データ（変速周期データ）の設定値が "0" の場合は、
「RATE CHANGE データ＝ RATE SET コマンドの UCYCLE または DCYCLE」にします。
・加速カーブの変速周期（μs）＝ RATE CHANGE データ（または UCYCLE）x 0.5 μs
・減速カーブの変速周期（μs）＝ RATE CHANGE データ（または DCYCLE）x 0.5 μs

SPEED RATE CHANGE コマンドを実行しても、予め設定されている速度パラメータの値は変わりません。

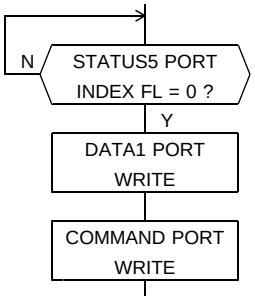
3-1-8. INDEX CHANGE の設定と実行

変更動作点を設定して、INDEX CHANGE を実行します。
変更動作点の設定は、変更動作点の変更が必要な場合に設定します。

(1) INDEX CHANGE SPEC SET

INDEX CHANGE 指令を実行する変更動作点と RFSPD を設定します。

<実行シーケンス>



INDEX CHANGE SPEC SET COMMAND

① STATUS5 PORT の INDEX FL フラグが "0" であることを確認します。
→ MC07_BRStatus5 関数

② DRIVE DATA1 PORT に設定データを書き込みます。

③ DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数(DATA2 PORT は、H'0000 を書き込み)

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
RFSPD D7	RFSPD D6	RFSPD D5	RFSPD D4	RFSPD D3	RFSPD D2	RFSPD D1	RFSPD D0

D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	–	–	INDEX CHANGE TYPE2	INDEX CHANGE TYPE1	INDEX CHANGE TYPE0

● 電源投入後の初期値は H'1400 (アンダーライン側) です。

D2--D0 ：INDEX CHANGE TYPE2--0

INDEX CHANGE 指令を実行する変更動作点を選択します。

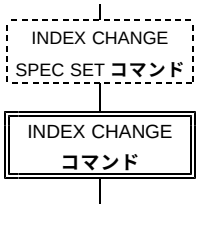
D2	D1	D0	INDEX CHANGE を実行する変更動作点	検出仕様
0	0	0	STATUS1 PORT の DRIVE = 1 で実行する	レベル検出
0	0	1	DRIVE = 1 のときに、他軸の STATUS3 PORT の OUT3 = 0 → 1 で実行する	エッジ検出
0	1	0	DRIVE = 1 のときに、STATUS3 PORT の GPIO2 = 0 → 1 で実行する	エッジ検出
0	1	1	DRIVE = 1 のときに、ORIGIN 停止機能の ORG エッジ信号の検出で実行する	エッジ検出
1	0	0	DRIVE = 1 のときに、STATUS3 PORT の OUT2 = 0 → 1 で実行する	エッジ検出
1	0	1	DRIVE = 1 のときに、STATUS3 PORT の OUT3 = 0 → 1 で実行する	エッジ検出
1	1	0	設定禁止	–
1	1	1	設定禁止	–

D15--D8 ：RFSPD D7--D0

INDEX CHANGE 指令による反転方向の動作も可能です。
この反転ドライブの第１パルス目のパルス周期（パルス速度）を設定します。
・設定範囲は、0 ～ 255（H'00 ～ H'FF）です。設定値は、1 Hz 単位です。
・RFSPD の初期値は、20 Hz（H'14）です。

RFSPD の設定値が "0" の場合は、RFSPD を RFSPD = RSPD x RESOL に補正します。

■ INDEX CHANGE の実行シーケンス



① INDEX CHANGE 指令を実行する変更動作点と
反転ドライブの第１パルス目のパルス周期を設定します。

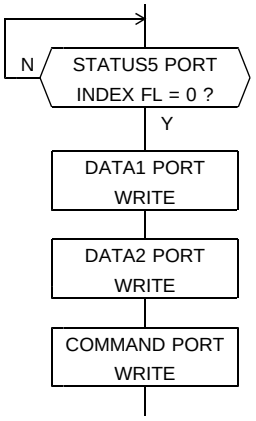
② INDEX CHANGE 指令を実行（設定）します。

初期値および設定値に対して変更が必要な場合に設定します。

(2) INC INDEX CHANGE

変更動作点の検出で、INC INDEX CHANGE 指令を実行します。
指定したデータを、起動位置を原点とする相対アドレスの停止位置に設定して、
INC INDEX ドライブを行います。

<実行シーケンス>　COMMAND H'CC　INC INDEX CHANGE COMMAND



① DRIVE STATUS5 PORT の INDEX FL フラグが "0" であることを確認します。
→ MC07_BRStatus5 関数

② DRIVE DATA1 PORT に目的地の相対アドレス D15--D0 を書き込みます。

③ DRIVE DATA2 PORT に目的地の相対アドレス D31--D16 を書き込みます。
設定範囲は、
-2,147,483,647 ～+2,147,483,647（H'8000_0001 ～ H'7FFF_FFFF）です。
正数が CW、負数が CCW 方向です。負数の場合は 2 の補数表現にします。

④ DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D15	←					INDEX CHANGE データ								→	D0

DRIVE DATA2 PORT の設定データ

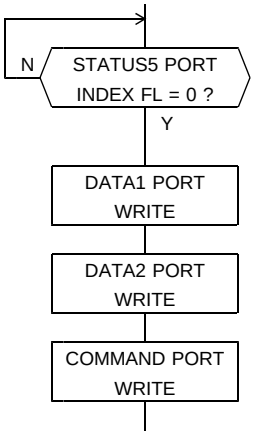
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D31	←					INDEX CHANGE データ								→	D16

指定する相対アドレスは、起動位置から停止位置までのパルス数を、
起動位置を原点として符号付きで表現した値です。

(3) ABS INDEX CHANGE

変更動作点の検出で、ABS INDEX CHANGE 指令を実行します。
指定したデータを、アドレスカウンタで管理している絶対アドレスの停止位置に設定して、
ABS INDEX ドライブを行います。

<実行シーケンス>　COMMAND H'CD　ABS INDEX CHANGE COMMAND



① DRIVE STATUS5 PORT の INDEX FL フラグが "0" であることを確認します。
→ MC07_BRStatus5 関数

② DRIVE DATA1 PORT に目的地の絶対アドレス A15--A0 を書き込みます。

③ DRIVE DATA2 PORT に目的地の絶対アドレス A31--A16 を書き込みます。
設定範囲は、
-2,147,483,647 ～+2,147,483,647（H'8000_0001 ～ H'7FFF_FFFF）です。
負数の場合は 2 の補数表現にします。

④ DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A15	←					INDEX CHANGE データ								→	A0

DRIVE DATA2 PORT の設定データ

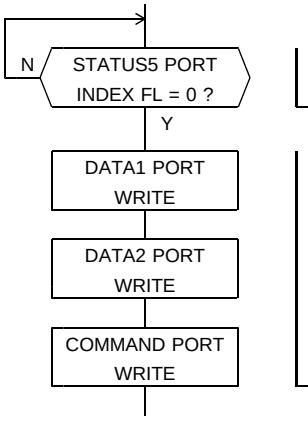
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A31	←					INDEX CHANGE データ								→	A16

指定する絶対アドレスは、アドレスカウンタで管理している絶対アドレスです。

(4) PLS INDEX CHANGE

変更動作点の検出で、PLS INDEX CHANGE 指令を実行します。
指定したデータを、変更動作点の検出位置を原点とする相対アドレスの停止位置に設定して、
INC INDEX ドライブを行います。

<実行シーケンス>　COMMAND H'CE　PLS INDEX CHANGE COMMAND



① DRIVE STATUS5 PORT の INDEX FL フラグが "0" であることを確認します。
→ MC07_BRStatus5 関数

② DRIVE DATA1 PORT に目的地の相対アドレス D15--D0 を書き込みます。

③ DRIVE DATA2 PORT に目的地の相対アドレス D31--D16 を書き込みます。
設定範囲は、
-2,147,483,647 ～+2,147,483,647（H'8000_0001 ～ H'7FFF_FFFF）です。
正数が CW、負数が CCW 方向です。負数の場合は 2 の補数表現にします。

④ DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D15	←					INDEX CHANGE データ								→	D0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D31	←					INDEX CHANGE データ								→	D16

指定する相対アドレスは、変更動作点の検出位置から停止位置までのパルス数を、
変更動作点の検出位置を原点として符号付きで表現した値です。

3-1-9. RAM I/F の設定

ここで説明する RAM I/F 機能は、SSMAP-63 による MCM 編集ツール、または直接ユーザアプリケーションから RAM 領域にアクセスして MCM(RAM 領域)を操作するためのコマンドです。
MCM 実行中に、ユーザアプリケーションから、当コマンドを実行することはできません。

(1) RAM SPEC SET

RAM READ JUMP コマンドの実行タイミングと有効条件を設定します。
NO OPERATION コマンドの実行タイミングを設定します。
このコマンドの実行は常時可能です。

<実行シーケンス>

COMMAND H'B0　RAM SPEC SET COMMAND

↓

DATA1 PORT WRITE　① DRIVE DATA1 PORT に設定データを書き込みます。

↓

COMMAND PORT WRITE　② DRIVE COMMAND PORT に COMMAND を書き込みます。

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
–	–	–	–	–	NOP TYPE2	NOP TYPE1	NOP TYPE0

D7	D6	D5	D4	D3	D2	D1	D0
–	JUMP ENABLE2	JUMP ENABLE1	JUMP ENABLE0	–	JUMP TYPE2	JUMP TYPE1	JUMP TYPE0

● 電源投入後の初期値は H'000 (アンダーライン側) です。

D0 ：JUMP TYPE0
D1 ：JUMP TYPE1
D2 ：JUMP TYPE2

RAM READ JUMP コマンドを内部 RAM のコマンドとして使用する場合に有効です。
内部 RAM にメモリした RAM READ JUMP コマンドの実行タイミングを選択します。

TYPE2	TYPE1	TYPE0	実行タイミング <レベル検出>
0	0	0	即実行する
0	0	1	STATUS3 PORT の OUT2 = 0 のときに実行する
0	1	0	STATUS3 PORT の GPIO2 = 0 のときに実行する
0	1	1	STATUS3 PORT の GPIO3 = 0 のときに実行する
1	0	0	STATUS3 PORT の OUT2 = 1 のときに実行する
1	0	1	設定禁止
1	1	0	STATUS3 PORT の GPIO2 = 1 のときに実行する
1	1	1	STATUS3 PORT の GPIO3 = 1 のときに実行する

実行タイミングの検出で、RAM から読み出した RAM READ JUMP コマンドを実行します。
・RAM READ JUMP コマンドは、JUMP ENABLE の有効条件が一致しているときに実行します。
　不一致のときには、RAM READ JUMP コマンドを無効にして、次のメモリデータを読み出します。

実行タイミングが検出されないときは、実行タイミングの検出を待ちます。
・実行タイミングを待っている間は、内部 RAM のコマンド読み出しを保留します。
・実行タイミングを待っている間は、MCM STATUS2 の MRSTBY = 1 になります。

D4 ：JUMP ENABLE0
D5 ：JUMP ENABLE1
D6 ：JUMP ENABLE2
RAM READ JUMP コマンド実行時の有効条件を選択します。

EN2	EN1	EN0	有効条件 <レベル検出>	無効条件
0	0	0	常時有効にする	無効にしない
0	0	1	STATUS3 PORT の OUT2 = 0 のときは有効にする	OUT2 = 1 のときは無効にする
0	1	0	STATUS3 PORT の GPIO2 = 0 のときは有効にする	GPIO2 = 1 のときは無効にする
0	1	1	STATUS3 PORT の GPIO3 = 0 のときは有効にする	GPIO3 = 1 のときは無効にする
1	0	0	STATUS3 PORT の OUT2 = 1 のときは有効にする	OUT2 = 0 のときは無効にする
1	0	1	設定禁止	―
1	1	0	STATUS3 PORT の GPIO2 = 1 のときは有効にする	GPIO2 = 0 のときは無効にする
1	1	1	STATUS3 PORT の GPIO3 = 1 のときは有効にする	GPIO3 = 0 のときは無効にする

有効条件が一致しているときは、RAM READ JUMP コマンドを実行します。
有効条件が不一致のときは、RAM READ JUMP コマンドを無効にします。

D8 ：NOP TYPE0
D9 ：NOP TYPE1
D10 ：NOP TYPE2
メモリデータの NO OPERATION コマンドの実行タイミングを選択します。

TYPE2	TYPE1	TYPE0	実行タイミング <レベル検出>
0	0	0	COMREG FL = 0 のときに実行する
0	0	1	設定禁止
0	1	0	COMREG FL = 0、STATUS3 PORT の GPIO2 = 0 のときに実行する
0	1	1	COMREG FL = 0、STATUS3 PORT の GPIO3 = 0 のときに実行する
1	0	0	COMREG FL = 0、STATUS3 PORT の OUT2 = 1 のときに実行する
1	0	1	設定禁止
1	1	0	COMREG FL = 0、STATUS3 PORT の GPIO2 = 1 のときに実行する
1	1	1	COMREG FL = 0、STATUS3 PORT の GPIO3 = 1 のときに実行する

実行タイミングの検出で、RAM から読み出した NO OPERATION コマンドを実行します。
・BUSY = 1 のときは、コマンド予約機能の予約レジスタに格納します。

実行タイミングが検出されないときは、実行タイミングの検出を待ちます。
・実行タイミングを待っている間は、内部 RAM のコマンド読み出しを保留します。
・実行タイミングを待っている間は、STATUS3 PORT の RAM READ STBY = 1 になります。

(2) RAM READ JUMP

移動アドレスから、内部 RAM のコマンド読み出しを開始します。

<実行シーケンス>

COMMAND H'B9　RAM READ JUMP COMMAND

DATA1 PORT WRITE

① DRIVE DATA1 PORT に移動アドレス A11--A0 を書き込みます。
設定範囲は、0 ～ 3,999（H'000 ～ H'F9F）です。

COMMAND PORT WRITE

② DRIVE COMMAND PORT に COMMAND を書き込みます。

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
–	–	–	–	A11	←			移動アドレス						→	A0

RAM READ JUMP コマンドは、有効条件が一致しているときに実行します。
・有効条件は、RAM SPEC SET コマンドの JUMP ENABLE で選択します。

有効条件が一致のときに、RAM READ JUMP コマンドを実行すると、
現在実行中の MCM 内のコマンドの読み出しを終了します。
・読み出し中の MCM 内のコマンドは、無効になります。

次に、指定された MCM の移動アドレスから、データとコマンドを読み出して、
MCM 内のコマンドを順次実行します。

・内部 RAM のコマンド実行中に、自動的に移動させる場合、
移動を実行する内部 RAM のメモリアドレスに、RAM READ JUMP コマンドを書き込んでおきます。

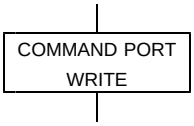
・内部 RAM の RAM READ JUMP コマンドは、実行タイミングの検出で実行します。
実行タイミングは、RAM SPEC SET コマンドの JUMP TYPE で選択します。

・更に、RAM READ JUMP コマンドは、有効条件が一致しているときに実行します。
不一致のときには、RAM READ JUMP コマンドを無効にして、次のコマンドを読み出します。

(3) RAM STOP

内部 RAM I/F を終了します。

<実行シーケンス>



COMMAND H'BF　RAM STOP COMMAND

① DRIVE COMMAND PORT に COMMAND を書き込みます。

MCM 内の RAM STOP コマンドの実行により、MCM(RAM)動作を終了します。

・内部 RAM のコマンド実行中に、自動的に終了させる場合、
最後に実行する内部 RAM のメモリアドレスに、RAM STOP コマンドを書き込んでおきます。

3-1-10. RSPD データの読み出し

(1) RSPD DATA READ

RSPD データを読み出します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

<実行シーケンス>

読み出すデータは、「15 ビットのパルス速度データ」です。

RSPD DATA READ コマンドを実行すると、
RSPD データを DRIVE DATA1 PORT（READ）にセットします。

RSPD データ
・RSPD は、HSPD, LSPD, ELSPD と同様の 15 ビットのパルス速度データです。
・STATUS1 PORT の DRIVE = 1 → 0 になると、最終出力のパルス速度データを RSPD に記憶します。
　ただし、最終出力のパルス速度が FSPD, RFSPD と JSPD の場合は、RSPD を書き換えません。
・RSPD のリセット後の初期値は、H'012C（300）です。

3-2.カウンタコマンド

3-2-1. アドレスカウンタの設定

(1) ADDRESS COUNTER MAX COUNT SET

アドレスカウンタの最大カウント数を設定します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

<実行シーケンス>

COMMAND H'87　ADDRESS COUNTER MAX COUNT SET COMMAND

DATA1 PORT WRITE

DATA2 PORT WRITE

COMMAND PORT WRITE

① DRIVE DATA1 PORT に最大カウント数 A15--A0 を書き込みます。

② DRIVE DATA2 PORT に最大カウント数 A31--A16 を書き込みます。
設定範囲は、1 ～ 4,294,967,295（H'0000_0001 ～ H'FFFF_FFFF）です。

③ DRIVE COMMAND PORT に COMMAND を書き込みます。
推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A15	←					最大カウント数								→	A0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A31	←					最大カウント数								→	A16

●電源投入後の初期値は H'FFFF_FFFF です。

・カウント数が設定値の 1/2 に達すると、STATUS4 PORT の ADDRESS OVF = 1 になります。

・最大カウント数を設定しても、現在のアドレスカウンタの値は変わりません。
アドレスカウンタの値が、最大カウント数の範囲内になったときから、設定が有効になります。

■最大カウント数

設定値をカウンタの最大値として、リングカウントします。
STATUS4 PORT の ADDRESS OVF フラグを無視すれば、回転系のアドレス管理ができます。

・最大カウント数＝ 1,999 の場合（2,000 カウントで１回転）
＋方向のカウント：0 → 1 →…→ 999 → 1000 (ADDRESS OVF = 1)→ 1001 →…→ 1999 → 0
－方向のカウント：0 → 1999 →…→ 1001 → 1000 (ADDRESS OVF = 1)→ 999 →…→ 1 → 0

・最大カウント数＝ 2,000 の場合（2,001 カウントで１回転）
＋方向のカウント：0 → 1 →…→ 1000 → 1001 (1001 になると ADDRESS OVF = 1)→…→ 2000 → 0
－方向のカウント：0 → 2000 →…→ 1001 → 1000 (1000 になると ADDRESS OVF = 1)→…→ 1 → 0

3-2-2. パルスカウンタの設定

(1) PULSE COUNTER MAX COUNT SET

パルスカウンタの最大カウント数を設定します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

<実行シーケンス>

COMMAND H'97　PULSE COUNTER MAX COUNT SET COMMAND

DATA1 PORT WRITE
DATA2 PORT WRITE
COMMAND PORT WRITE

① DRIVE DATA1 PORT に最大カウント数 D15--D0 を書き込みます。

② DRIVE DATA2 PORT に最大カウント数 D31--D16 を書き込みます。
設定範囲は、1 ～ 4,294,967,295（H'0000_0001 ～ H'FFFF_FFFF）です。

③ DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D15	←					最大カウント数								→	D0

DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D31	←					最大カウント数								→	D16

●電源投入後の初期値は H'FFFF_FFFF です。

・カウント数が設定値の 1/2 に達すると、STATUS4 PORT の PULSE OVF = 1 になります。

・最大カウント数を設定しても、現在のパルスカウンタの値は変わりません。
パルスカウンタの値が、最大カウント数の範囲内になったときから、設定が有効になります。

■最大カウント数

設定値をカウンタの最大値として、リングカウントします。
STATUS4 PORT の PULSE OVF フラグを無視すれば、回転系のアドレス管理ができます。

・最大カウント数＝ 1,999 の場合（2,000 カウントで１回転）
＋方向のカウント：0 → 1 →…→ 999 → 1000 (ADDRESS OVF = 1)→ 1001 →…→ 1999 → 0
－方向のカウント：0 → 1999 →…→ 1001 → 1000 (ADDRESS OVF = 1)→ 999 →…→ 1 → 0

・最大カウント数＝ 2,000 の場合（2,001 カウントで１回転）
＋方向のカウント：0 → 1 →…→ 1000 → 1001 (1001 になると ADDRESS OVF = 1)→…→ 2000 → 0
－方向のカウント：0 → 2000 →…→ 1001 → 1000 (1000 になると ADDRESS OVF = 1)→…→ 1 → 0

3-2-3. カウンタのラッチ・クリア機能の設定

設定したラッチタイミングのアクティブエッジで、カウンタのカウントデータをラッチします。
ラッチしたデータは、次のラッチタイミングのアクティブエッジが入力するまで保存します。
ラッチデータは、DRIVE DATA1, 2 PORT（READ）から読み出します。

各カウンタには、ラッチタイミングによるカウンタのクリア機能があります。

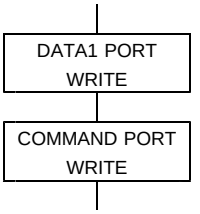
■カウンタのクリア機能

カウントデータのラッチと同時に、カウンタのデータを "0" にクリアします。
カウンタのカウントとクリアのタイミングが同時に発生した場合は、クリアを優先します。

(1) COUNT LATCH SPEC SET

各種カウンタのカウントデータをラッチするタイミングとクリア機能を設定します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

<実行シーケンス>



COMMAND H'E8　COUNT LATCH SPEC SET COMMAND

① DRIVE DATA1 PORT に設定データを書き込みます。

② DRIVE COMMAND PORT に COMMAND を書き込みます。
→ 推奨:MC07_LWDrive 関数(DATA2 PORT は、H'0000 を書き込み)

DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
–	–	–	–	DFL CLR ENABLE	DFL LATCH TYPE2	DFL LATCH TYPE1	DFL LATCH TYPE0

D7	D6	D5	D4	D3	D2	D1	D0
PULSE CLR ENABLE	PULSE LATCH TYPE2	PULSE LATCH TYPE1	PULSE LATCH TYPE0	ADDRESS CLR ENABLE	ADDRESS LATCH TYPE2	ADDRESS LATCH TYPE1	ADDRESS LATCH TYPE0

● 電源投入後の初期値は H'000（アンダーライン側）です。

D0 ：ADDRESS LATCH TYPE0
D1 ：ADDRESS LATCH TYPE1
D2 ：ADDRESS LATCH TYPE2
アドレスカウンタのカウントデータをラッチするタイミングを選択します。

TYPE2	TYPE1	TYPE0	ラッチタイミング <エッジ検出>
0	0	0	ADDRESS LATCH DATA READ コマンドの実行でラッチする
0	0	1	設定禁止
0	1	0	STATUS3 PORT の GPIO2=0 → 1 でラッチする
0	1	1	ORIGIN 停止機能の ORG エッジ信号の検出でラッチする
1	0	0	STATUS3 PORT の OUT2=0 → 1 でラッチする
1	0	1	設定禁止
1	1	0	設定禁止
1	1	1	設定禁止

D3 ：ADDRESS CLR ENABLE
カウンタのクリア機能で、アドレスカウンタを「クリアする／クリアしない」を選択します。
0 ：クリアしない
1 ：クリアする

D4 ：PULSE LATCH TYPE0
D5 ：PULSE LATCH TYPE1
D6 ：PULSE LATCH TYPE2
パルスカウンタのカウントデータをラッチするタイミングを選択します。

TYPE2	TYPE1	TYPE0	ラッチタイミング <エッジ検出>
0	0	0	PULSE LATCH DATA READ コマンドの実行でラッチする
0	0	1	設定禁止
0	1	0	STATUS3 PORT の GPIO2=0 → 1 でラッチする
0	1	1	ORIGIN 停止機能の ORG エッジ信号の検出でラッチする
1	0	0	STATUS3 PORT の OUT2=0 → 1 でラッチする
1	0	1	設定禁止
1	1	0	設定禁止
1	1	1	設定禁止

D7 ：PULSE CLR ENABLE
カウンタのクリア機能で、パルスカウンタを「クリアする／クリアしない」を選択します。
0 ：クリアしない
1 ：クリアする

D8 ：DFL LATCH TYPE0
D9 ：DFL LATCH TYPE1
D10 ：DFL LATCH TYPE2
パルス偏差カウンタのカウントデータをラッチするタイミングを選択します。

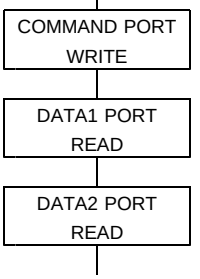
TYPE2	TYPE1	TYPE0	ラッチタイミング <エッジ検出>
0	0	0	DFL LATCH DATA READ コマンドの実行でラッチする
0	0	1	設定禁止
0	1	0	STATUS3 PORT の GPIO2=0 → 1 でラッチする
0	1	1	ORIGIN 停止機能の ORG エッジ信号の検出でラッチする
1	0	0	STATUS3 PORT の OUT2=0 → 1 でラッチする
1	0	1	設定禁止
1	1	0	設定禁止
1	1	1	設定禁止

D11 ：DFL CLR ENABLE
カウンタのクリア機能で、パルス偏差カウンタを「クリアする／クリアしない」を選択します。
0 ：クリアしない
1 ：クリアする

3-2-4. カウントデータのラッチデータの読み出し

■ラッチデータの読み出しシーケンス

<実行シーケンス>



① DRIVE COMMAND PORT に COMMAND を書き込みます。
(DATA1,2 PORT は、H'0000 にして COMMAND を書き込み)

② DRIVE DATA1 PORT からカウントデータ D15--D0 を読み出します。

③ DRIVE DATA2 PORT からカウントデータ D31--D16 を読み出します。

推奨:MC07_LWRDrive 関数

DRIVE DATA1 PORT の読み出しデータ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D15 ←						ラッチデータ									→ D0

DRIVE DATA2 PORT の読み出しデータ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D31 ←						ラッチデータ									→ D16

・ADDRESS LATCH DATA READ コマンドまたは PULSE LATCH DATA READ コマンドを実行すると、カウンタのラッチデータを DRIVE DATA1, 2 PORT（READ）にセットします。

・DFL LATCH DATA READ コマンドを実行すると、カウンタのラッチデータを DRIVE DATA1 PORT（READ）にセットします。

(1) ADDRESS LATCH DATA READ

アドレスカウンタのラッチデータを読み出します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

COMMAND H'DC　ADDRESS LATCH DATA READ COMMAND

(2) PULSE LATCH DATA READ

パルスカウンタのラッチデータを読み出します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

COMMAND H'DD　PULSE LATCH DATA READ COMMAND

(3) DFL LATCH DATA READ

パルス偏差カウンタのラッチデータを読み出します。
このコマンドの実行は DRIVE STATUS1 PORT の BUSY=1 のときも可能です。

COMMAND H'DE　DFL LATCH DATA READ COMMAND

4．機能説明

4-1.ドライブ仕様

4-1-1. コマンド予約機能

MCC09 の各軸には、20 命令分のデータ・コマンドを格納する予約レジスタがあります。
予約レジスタには、DRIVE COMMAND の汎用コマンドを予約することができます。
＊ DRIVE COMMAND の特殊コマンドは予約できません。

予約レジスタの状態は、DRIVE STATUS1 PORT の COMREG EP と COMREG FL フラグで確認します。

BUSY = 1、COMREG FL = 0 のときに、DRIVE COMMAND PORT に汎用コマンドを書き込むと、
DRIVE DATA1, 2 PORT のデータと汎用コマンドの１命令分を、予約レジスタに格納します。

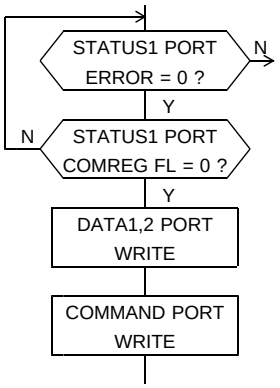
予約レジスタは FIFO 構成になっています。
実行中のコマンド処理が終了すると、予約レジスタに格納したコマンドを順次実行します。

● コマンド予約機能が自動的に無効となる状態
以下の状態を検出すると、予約レジスタに格納している汎用コマンドをすべてクリアします。
同時に COMREG FL = 1、COMREG EP = 1 にして、汎用コマンドの書き込みを無効にします。
・STATUS1 PORT の ERROR = 1 の検出

ERROR = 0 にクリアすると、
COMREG FL = 0、COMREG EP = 1 にして、汎用コマンドの書き込みを有効にします。

■ コマンド予約機能の実行例
汎用コマンドを予約する場合は、BUSY = 0 の代わりに COMREG FL = 0 を確認します。
予約シーケンス実行中に BUSY = 0 になった場合は、通常のコマンド実行になります。

●コマンド予約シーケンス



① DRIVE STATUS1 PORT の ERROR フラグが "0" であることを確認します。

② DRIVE STATUS1 PORT の COMREG FL フラグが "0" であることを確認します。

③ COMMAND に必要なデータを DRIVE DATA1,2 PORT に書き込みます。

④ DRIVE COMMAND PORT に COMMAND を書き込みます。

● INDEX ドライブ → JOG ドライブの連続実行シーケンス

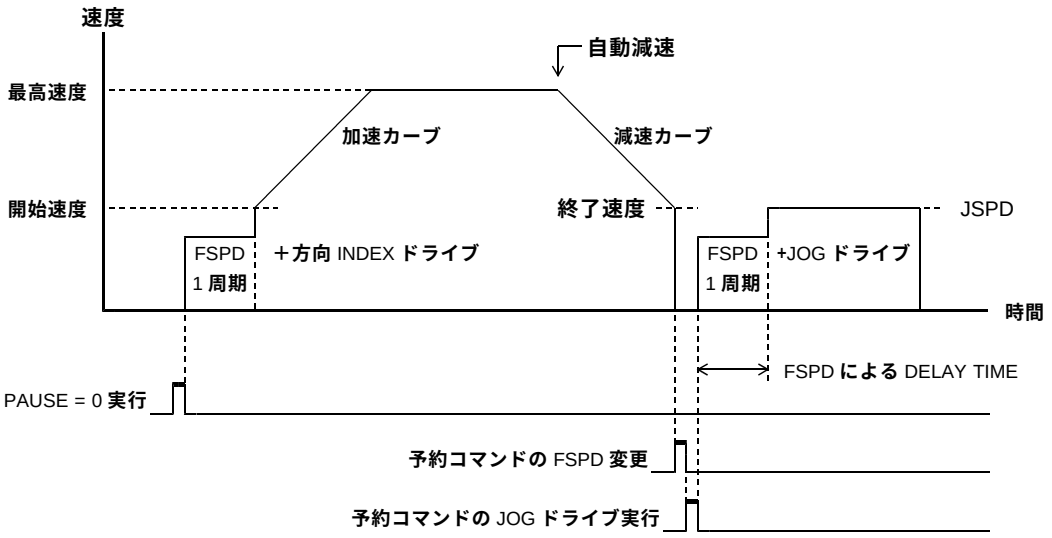
① STATUS1 PORT の PAUSE = 1 にします。

② ＋方向 INDEX ドライブを実行します。
STBY = 1 のままドライブの開始を保留します。

③ FSPD による DELAY TIME の設定を予約します。

④ ＋方向 JOG ドライブの実行を予約します。

⑤ STATUS1 PORT の PAUSE = 0 にします。
STBY = 0 になり、INDEX ドライブ → JOG ドライブを開始します。



4-1-2. 入出力仕様

(1) 入出力信号の論理切り替え機能

下記の入出力信号のアクティブ論理を初期値から切り替えることができます。

入力信号	初期値	対応コマンド
FSSTOP 信号	正論理入力	HARD INITIALIZE7 コマンド
CWLM 信号	正論理入力	
CCWLM 信号	正論理入力	
DALM 信号	正論理入力	
ORG 信号	負論理入力	
NORG 信号	負論理入力	
PAUSE	1 でアクティブ	
CWP 信号	負論理出力	HARD INITIALIZE8 コマンド
CCWP 信号	負論理出力	

・PAUSE 信号は外部とインターフェースする信号ではありません。
コマンド予約機能で操作する MCC09 の入力信号です。

・アクティブ論理を変更すると、変更した信号の内部デジタルフィルタ機能が動作します。
デジタルフィルタ機能の時定数経過後に、アクティブ論理の変更が確定します。

4-1-3. ドライブパラメータ
(1) 加減速パラメータ

加減速ドライブは、加速カーブと減速カーブで加減速を行うドライブです。
・加速カーブは、S字加速の変速領域と直線加速の変速領域で構成します。
・減速カーブは、S字減速の変速領域と直線減速の変速領域で構成します。
・加速カーブと減速カーブを異なる設定にすると、非対称の加減速ドライブになります。

加減速ドライブには、以下のドライブパラメータの設定が必要です。

● 速度のパラメータ
・FSPD ：第１パルスの速度（Hz）
・RFSPD ：INDEX CHANGE による反転ドライブの第１パルスの速度（Hz）

・RESOL ：速度データの速度倍率
・HSPD ：最高速時のパルス速度データ
・LSPD ：加速開始時のパルス速度データ
・ELSPD ：減速終了時のパルス速度データ

・JSPD ：JOG ドライブと JSPD SCAN ドライブの速度（Hz）

・RSPD ：RSPD は、HSPD, LSPD, ELSPD と同様の 15 ビットのパルス速度データです。
DRIVE = 1 → 0 になると、最終出力のパルス速度データを RSPD に記憶します。
ただし、最終出力のパルス速度が FSPD, RFSPD と JSPD の場合は、RSPD を書き換えません。
RSPD のリセット後の初期値は、H'012C（300）です。

● 加減速カーブのパラメータ
・UCYCLE ：加速カーブの変速周期データ
・DCYCLE ：減速カーブの変速周期データ

・SUAREA ：加速カーブのS字変速領域データ
・SDAREA ：減速カーブのS字変速領域データ
・SUH ：加速カーブのS字加速終了部の変速領域データ（SCAREA MODE = 1 のときに有効）
・SDH ：減速カーブのS字減速開始部の変速領域データ（SCAREA MODE = 1 のときに有効）

● 減速パルス数の調整／設定パラメータ
・DOWN PULSE ADJUST ：自動減速パルス数のオフセットパルス数／マニュアル減速パルス数

■ 加減速ドライブの速度とS字領域

加減速ドライブのパルス速度は、速度データと速度倍率で設定します。
・最高速時の速度 （Hz）＝ HSPD x RESOL
・加速開始時の速度（Hz）＝ LSPD x RESOL
・減速終了時の速度（Hz）＝ ELSPD x RESOL

・SPEED RATE CHANGE の指定速度（Hz）＝ SPEED CHANGE データ x RESOL

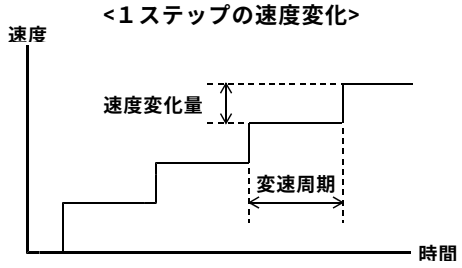
S字領域は、S字変速領域データと速度倍率で設定します。
・S字加速開始部の変速領域（Hz）＝ SUAREA x RESOL
・S字加速終了部の変速領域（Hz）＝ SUAREA（または SUH）x RESOL

・S字減速開始部の変速領域（Hz）＝ SDAREA（または SDH）x RESOL
・S字減速終了部の変速領域（Hz）＝ SDAREA x RESOL

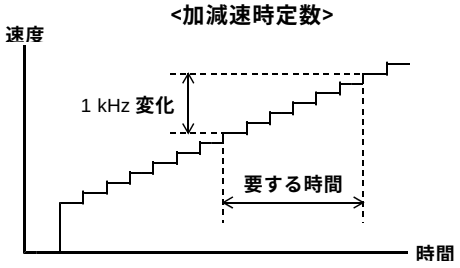
(2) 加減速時定数

加速および減速は、速度変化量を変速周期毎に加算および減算することで行っています。
加減速時定数は、速度を 1 kHz 変化させるのに要する時間（ms/kHz）で表しています。
本資料では、この時定数を RATE と呼称しています。

<1 ステップの速度変化>



<加減速時定数>



● 加減速時定数の設定方法

①最高速度と速度倍率を設定します。設定した速度倍率データで、速度変化量が決定します。
速度倍率を小さくすると、加減速が滑らかになります。

②開始速度と終了速度を設定します。

③加速の変速周期と減速の変速周期を設定します。設定した変速周期で、加減速時間が決定します。
速度変化量と変速周期で、目的に合った加減速時定数（加減速時間）を設定します。

■ 加減速 RATE の計算式

速度倍率データ（RESOL）と変速周期データ（UCYCLE）で、任意の加速 RATE を設定します。
速度倍率データ（RESOL）と変速周期データ（DCYCLE）で、任意の減速 RATE を設定します。

● 加速カーブの直線加速 RATE の計算式

・直線加速 RATE (ms/kHz) ＝ $\frac{\text{加速カーブの変速周期 (ms)}}{\text{直線加速カーブの速度変化量 (kHz)}} = \frac{\text{UCYCLE}}{\text{RESOL} * 2}$

加速カーブの変速周期 (ms) ＝ UCYCLE * 0.5 * 10^-3

直線加速カーブの速度変化量 (kHz) ＝ RESOL * 10^-3

● 減速カーブの直線減速 RATE の計算式

・直線減速 RATE (ms/kHz) ＝ $\frac{\text{減速カーブの変速周期 (ms)}}{\text{直線減速カーブの速度変化量 (kHz)}} = \frac{\text{DCYCLE}}{\text{RESOL} * 2}$

減速カーブの変速周期 (ms) ＝ DCYCLE * 0.5 * 10^-3

直線減速カーブの速度変化量 (kHz) ＝ RESOL * 10^-3

■ RATE DATA TABLE（参考）

RATE	RESOL = 1	RESOL = 5	RESOL = 10	RESOL = 20	RESOL = 50	RESOL = 200	RESOL = 400
(ms/kHz)	U/D CYCLE	U/D CYCLE	U/D CYCLE	U/D CYCLE	U/D CYCLE	U/D CYCLE	U/D CYCLE
5,000	10,000						
3,000	6,000						
2,000	4,000						
1,000	2,000	10,000					
500	1,000	5,000	10,000				
300	600	3,000	6,000	12,000			
200	400	2,000	4,000	8,000			
100	200	1,000	2,000	4,000	10,000		
50	100	500	1,000	2,000	5,000		
30	60	300	600	1,200	3,000	12,000	
20	40	200	400	800	2,000	8,000	16,000
10	20	100	200	400	1,000	4,000	8,000
5	10	50	100	200	500	2,000	4,000
3	6	30	60	120	300	1,200	2,400
2	4	20	40	80	200	800	1,600
1	2	10	20	40	100	400	800
0.5	1	5	10	20	50	200	400
0.3		3	6	12	30	120	240
0.2		2	4	8	20	80	160
0.1		1	2	4	10	40	80
0.05			1	2	5	20	40
0.03					3	12	24
0.02					2	8	16
0.01					1	4	8
0.005						2	4
0.0025						1	2
0.00125							1

(3) 直線加減速ドライブ

直線加減速ドライブは、S字加減速の変速領域を "0" に設定して、加減速を行うドライブです。
・開始速度から最高速度まで、S字変速領域がない直線加速カーブで加速します。
・最高速度から終了速度まで、S字変速領域がない直線減速カーブで減速します。

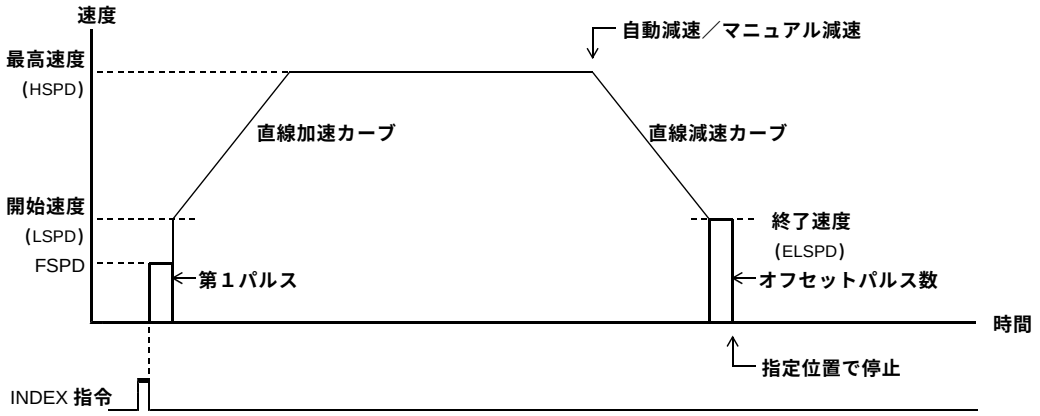
● 直線加速カーブ
SCAREA SET コマンドの SUAREA を "0" に設定します。
SHAREA SET コマンドの SUH を "0" に設定します。
・SPEC INITIALIZE3 コマンドの SCAREA MODE = 0 の場合は、SUH の設定は不要です。

開始速度から最高速度まで、UCYCLE の直線加速カーブで加速します。

● 直線減速カーブ
SCAREA SET コマンドの SDAREA を "0" に設定します。
SHAREA SET コマンドの SDH を "0" に設定します。
・SPEC INITIALIZE3 コマンドの SCAREA MODE = 0 の場合は、SDH の設定は不要です。

最高速度から終了速度まで、DCYCLE の直線減速カーブで減速します。

■ 直線加減速ドライブの動作



オフセットパルス数の設定は、
SPEC INITIALIZE3 コマンドの SCAREA MODE = 0 に設定している場合に有効です。

● 直線加減速ドライブの加速時間と減速時間

直線加速カーブの加速時間（ms）：0 ≦ TU ＜ 最高速度の１周期
= (UCYCLE * 0.5 * 10^-3) * (HSPD - LSPD + 1) + TU + 第１パルスの周期(ms)

直線減速カーブの減速時間（ms）：0 ≦ TD ＜ 終了速度の１周期
= (DCYCLE * 0.5 * 10^-3) * (HSPD - ELSPD + 1) + TD

・オフセットパルス数 = 0 で減速停止する場合の減速時間です。
・INDEX ドライブの自動減速停止時には、オフセットパルス数（初期値：+1）の増減があります。

(4) S 字加減速ドライブ

S字加減速ドライブは、S字加減速の変速領域を設定して、加減速を行うドライブです。
・加速開始部のS字変速領域と加速終了部のS字変速領域を、S字加速カーブで加速します。
・減速開始部のS字変速領域と減速終了部のS字変速領域を、S字減速カーブで減速します。

● S字加速カーブ

SCAREA SET コマンドの SUAREA でS字加速開始部の変速領域を設定します。
SHAREA SET コマンドの SUH でS字加速終了部の変速領域を設定します。
・SPEC INITIALIZE3 コマンドの SCAREA MODE = 0 の場合は、SUH の設定は不要です。
SUH = SUAREA にして、S字加速カーブを形成します。

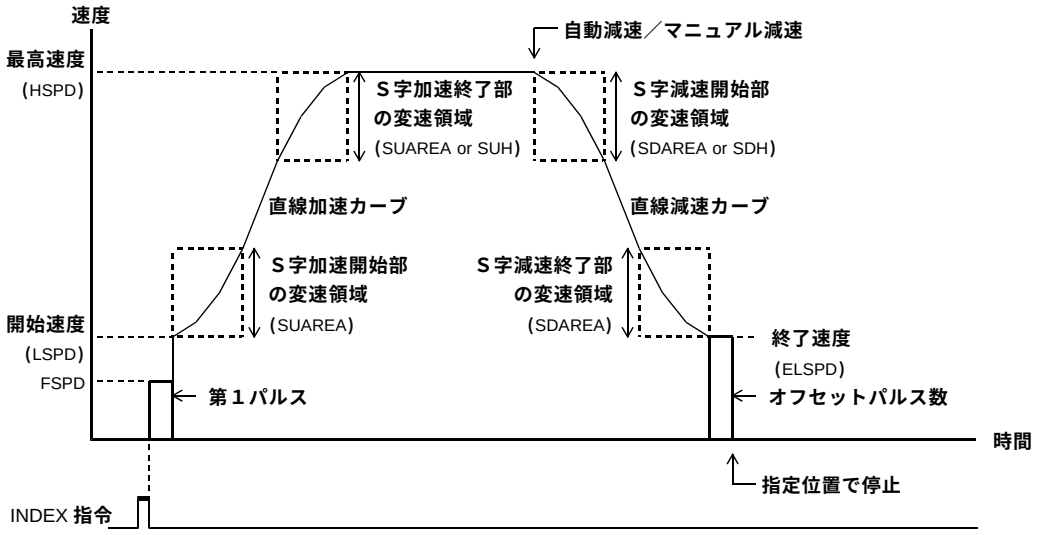
SUAREA と SUH で設定した変速領域が、S字加速カーブを形成します。
残りの速度領域は、UCYCLE の直線加速カーブで加速します。

● S字減速カーブ

SCAREA SET コマンドの SDAREA でS字減速終了部の変速領域を設定します。
SHAREA SET コマンドの SDH でS字減速開始部の変速領域を設定します。
・SPEC INITIALIZE3 コマンドの SCAREA MODE = 0 の場合は、SDH の設定は不要です。
SDH = SDAREA にして、S字減速カーブを形成します。

SDAREA と SDH で設定した変速領域が、S字減速カーブを形成します。
残りの速度領域は、DCYCLE の直線減速カーブで減速します。

■ S字加減速ドライブの動作



オフセットパルス数の設定は、
SPEC INITIALIZE3 コマンドの SCAREA MODE = 0 に設定している場合に有効です。

● S字加減速ドライブの加速時間と減速時間

S字加速カーブの加速時間（ms）：SUL = SUAREA，0 ≦ TU ＜ 最高速度の１周期
= T1 + T2 + T3 + TU + 第１パルスの周期(ms)
= (UCYCLE * 0.5 * 10^-3) * (HSPD - LSPD + 1 + SUH + SUL) + TU + 第１パルスの周期(ms)

・S字加速開始部の変速領域（ms）：T1 = (UCYCLE * 0.5 * 10^-3) * SUL * 2
・直線加速カーブの変速領域（ms）：T2 = (UCYCLE * 0.5 * 10^-3) * (HSPD - LSPD + 1 - SUH - SUL)
・S字加速終了部の変速領域（ms）：T3 = (UCYCLE * 0.5 * 10^-3) * SUH * 2
・(HSPD - LSPD) ≧ (SUH + SUL) で加速する場合の加速時間です。

S字減速カーブの減速時間（ms）：SDL = SDAREA，0 ≦ TD ＜ 終了速度の１周期
= T4 + T5 + T6 + TD
= (DCYCLE * 0.5 * 10^-3) * (HSPD - ELSPD + 1 + SDH + SDL) + TD

・S字減速開始部の変速領域（ms）：T4 = (DCYCLE * 0.5 * 10^-3) * SDH * 2
・直線減速カーブの変速領域（ms）：T5 = (DCYCLE * 0.5 * 10^-3) * (HSPD - ELSPD + 1 - SDH - SDL)
・S字減速終了部の変速領域（ms）：T6 = (DCYCLE * 0.5 * 10^-3) * SDL * 2
・(HSPD - ELSPD) ≧ (SDH + SDL)、オフセットパルス数 = 0 で減速停止する場合の減速時間です。
・INDEX ドライブの自動減速停止時には、オフセットパルス数（初期値：+1）の増減があります。

● S字加減速ドライブの加速パルス数と減速パルス数

S字加速カーブの加速パルス数 ：SUL = SUAREA
= P1 + P2 + P3 + 第１パルス (+1)
= (UCYCLE * 0.5 * 10^-6) * RESOL * (1/2 * (HSPD - LSPD + 1) * (HSPD + LSPD)
+ SUH * HSPD + SUL * LSPD - 1/6 * (SUH - SUL) * (SUH + SUL + 3)) + 1

・S字加速開始部のパルス数 ：P1 = (LSPD + 1/3 * SUL) * RESOL * T1 * 10^-3
・直線加速カーブのパルス数 ：P2 = (HSPD - SUH + LSPD + SUL) * RESOL * 1/2 * T2 * 10^-3
・S字加速終了部のパルス数 ：P3 = (HSPD - 1/3 * SUH) * RESOL * T3 * 10^-3
・(HSPD - LSPD) ≧ (SUH + SUL) で加速する場合の加速パルス数（小数点以下は切り上げ）です。

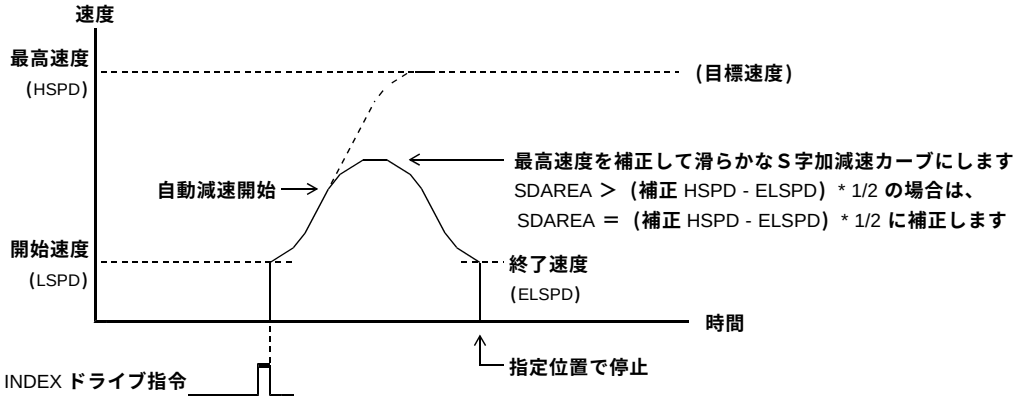
S字減速カーブの減速パルス数 ：SDL = SDAREA
= P4 + P5 + P6 + ADJUST PULSE (+2)
= (DCYCLE * 0.5 * 10^-6) * RESOL * (1/2 * (HSPD - ELSPD + 1) * (HSPD + ELSPD)
+ SDH * HSPD + SDL * ELSPD - 1/6 * (SDH - SDL) * (SDH + SDL + 3)) + 2
= (3 * (HSPD - ELSPD + 1) * (HSPD + ELSPD) + 6 * (SDH * HSPD + SDL * ELSPD)
- (SDH - SDL) * (SDH + SDL + 3)) * RESOL * DCYCLE * 1/12 * 10^-6 + 2

・S字減速開始部のパルス数 ：P4 = (HSPD - 1/3 * SDH) * RESOL * T4 * 10^-3
・直線減速カーブのパルス数 ：P5 = (HSPD - SDH + ELSPD + SDL) * RESOL * 1/2 * T5 * 10^-3
・S字減速終了部のパルス数 ：P6 = (ELSPD + 1/3 * SDL) * RESOL * T6 * 10^-3
・(HSPD - ELSPD) ≧ (SDH + SDL) で減速停止する場合の減速パルス数（小数点以下は切り上げ）です。

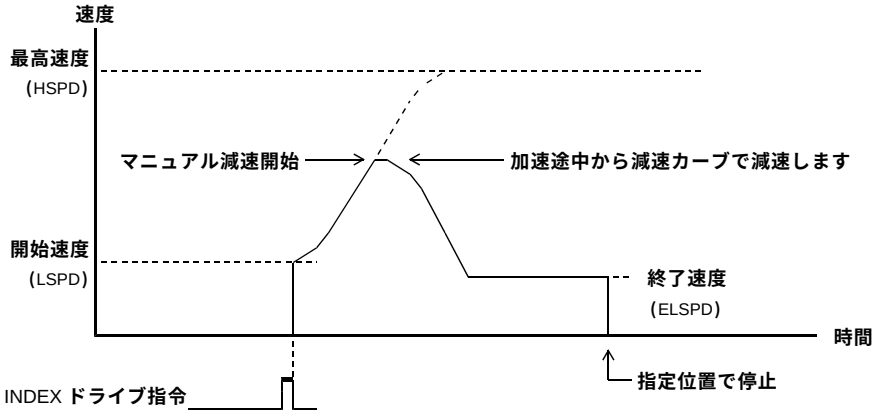
<参考> S字減速カーブの減速パルス数 ：SD = SDL = SDH の場合
= P4 + P5 + P6 + 2
= (DCYCLE * 0.5 * 10^-6) * RESOL * (1/2 * (HSPD - ELSPD + 1) * (HSPD + ELSPD)
+ SD * HSPD + SD * ELSPD - 1/6 * (SD - SD) * (SD + SD + 3)) + 2
= (DCYCLE * 0.5 * 10^-6) * RESOL * 1/2 * (HSPD - ELSPD + 1 + 2 * SD) * (HSPD + ELSPD) + 2
= (HSPD - ELSPD + 1 + 2 * SD) * (HSPD + ELSPD) * RESOL * DCYCLE * 1/4 * 10^-6 + 2

■ S 字加減速 INDEX ドライブの三角駆動回避動作

● SPEC INITIALIZE3 コマンドの SCAREA MODE = 0 の場合
S字加減速の INDEX ドライブで、
停止位置までのパルス数が少なくて、最高速度（目標速度）に達しない場合は、
自動的に最高速度を引き下げて、滑らかなS字加速カーブで加速を行い、
S字加速終了後からS字減速カーブ（または補正したS字減速カーブ）で減速を開始し、
指定位置で INDEX ドライブを停止します。

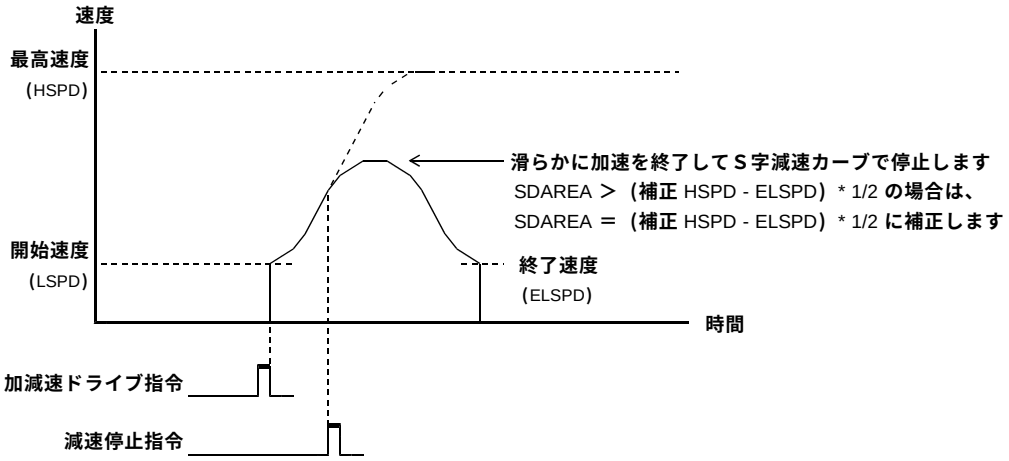


● SPEC INITIALIZE3 コマンドの SCAREA MODE = 1 の場合
このモードの INDEX ドライブでは、マニュアル設定した減速パルス数で減速を開始します。
停止位置までのパルス数が少なくて、最高速度に達しないまま減速を開始した場合は、
加速途中から補正しない減速カーブで終了速度まで減速し、
指定位置で INDEX ドライブを停止します。



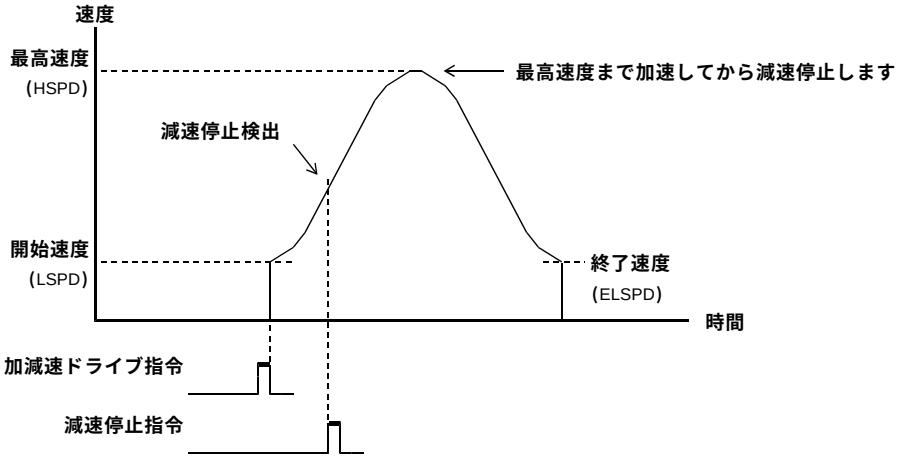
■ 減速停止指令検出時の三角駆動回避動作

● SPEC INITIALIZE3 コマンドの SCAREA MODE = 0 の場合
S字加速中に減速停止指令を検出した場合は、
SUAREA のS字加速終了カーブで滑らかに加速を終了し、
S字減速カーブ（または補正したS字減速カーブ）で減速停止します。



S字加減速の INDEX ドライブでは、減速停止指令を検出すると、自動減速機能をマスクして、
S字減速カーブ（または補正したS字減速カーブ）で減速停止します。
減速中に INDEX ドライブの指定位置を検出した場合は、指定位置で即時停止します。

● SPEC INITIALIZE3 コマンドの SCAREA MODE = 1 の場合
加速中に減速停止指令を検出した場合は、最高速度まで加速してから減速停止します。



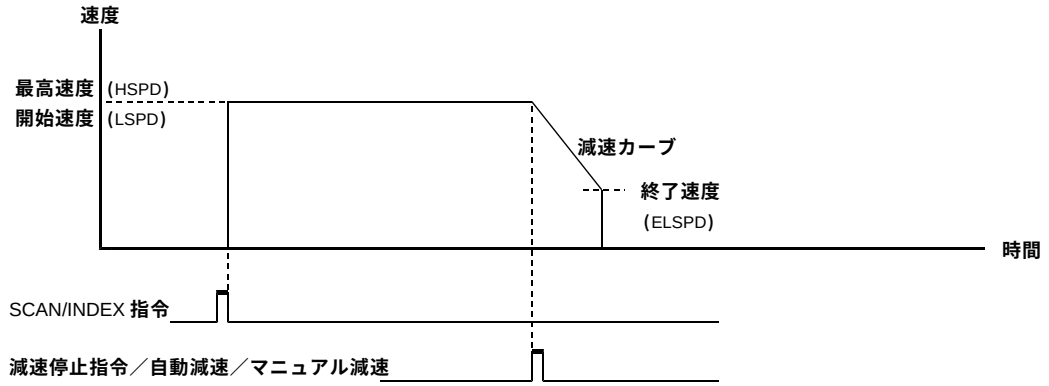
減速停止指令検出後の加速動作中に、
INDEX ドライブのマニュアル設定した減速パルス数で減速を開始した場合は、
加速途中から補正しない減速カーブで減速を開始し、
終了速度または指定位置で INDEX ドライブを停止します。

(5) 加速ドライブ

「開始速度＜最高速度」および「最高速度＝終了速度」に設定すると、
開始速度と最高速度による加速ドライブを行います。

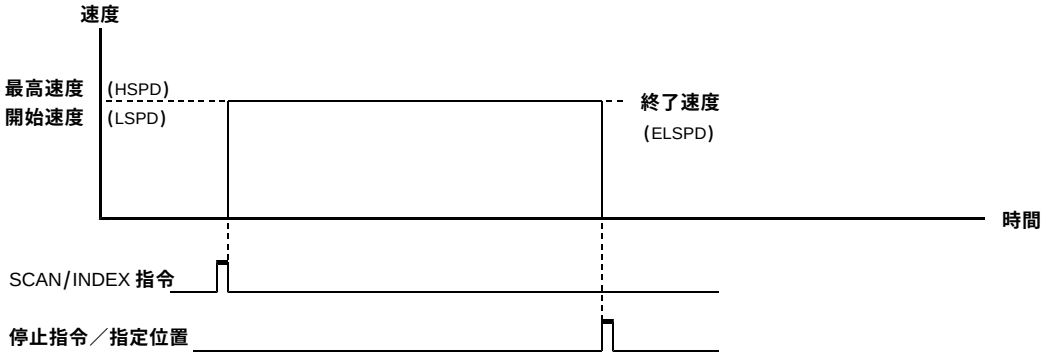
(6) 減速ドライブ

「開始速度＝最高速度」および「最高速度＞終了速度」に設定すると、
最高速度と終了速度による減速ドライブを行います。



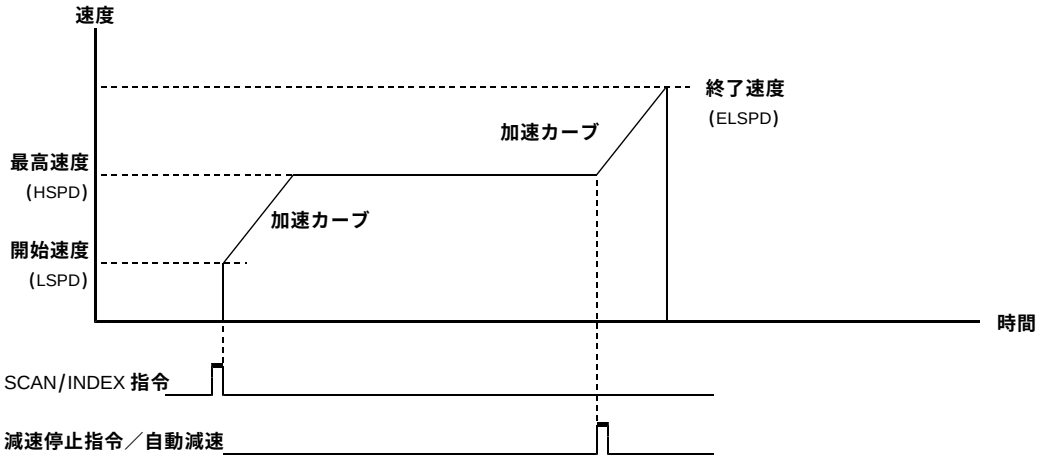
(7) 一定速ドライブ

「開始速度＝最高速度＝終了速度」に設定すると、
開始速度での一定速ドライブを行います。

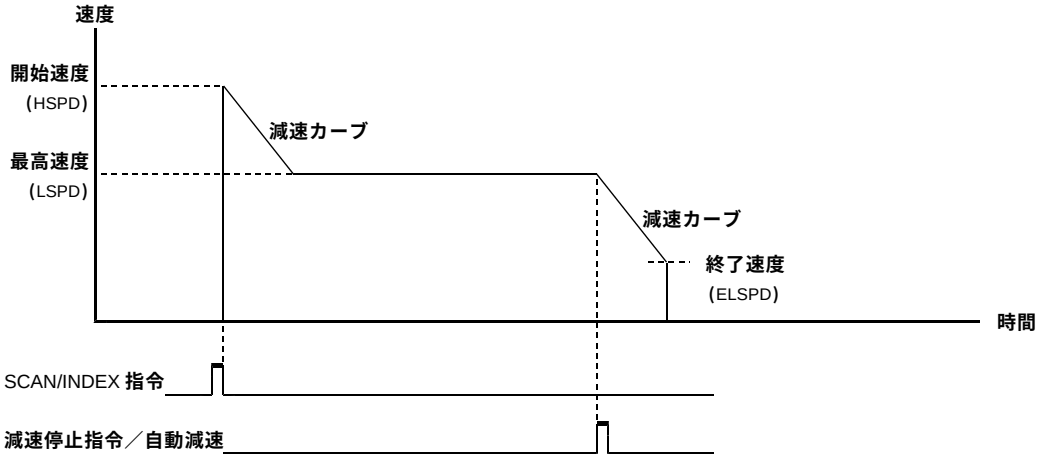


(8) その他のドライブ （SCAREA MODE = 0 の場合に可能）

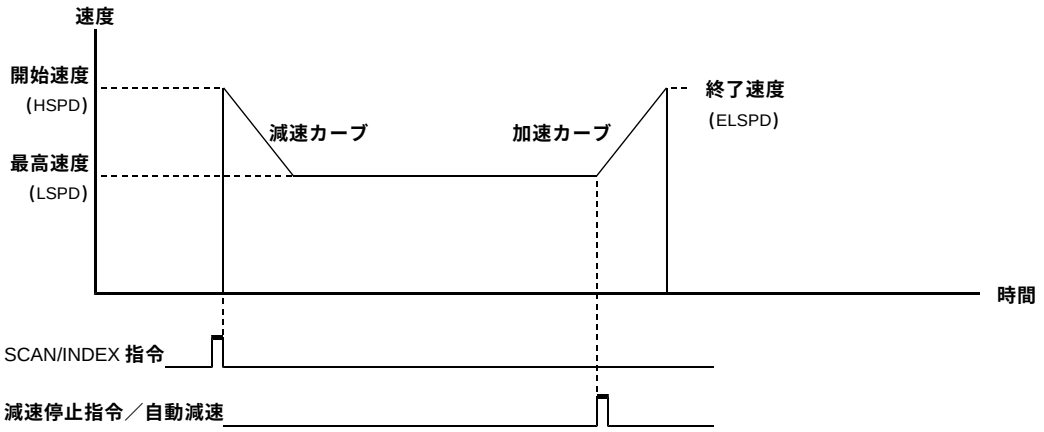
●「開始速度＜最高速度＜終了速度」に設定すると、以下の加速ドライブを行います。



●「開始速度＞最高速度＞終了速度」に設定すると、以下の減速ドライブを行います。



●「開始速度＞最高速度」および「最高速度＜終了速度」に設定すると、
以下の加減速ドライブを行います。

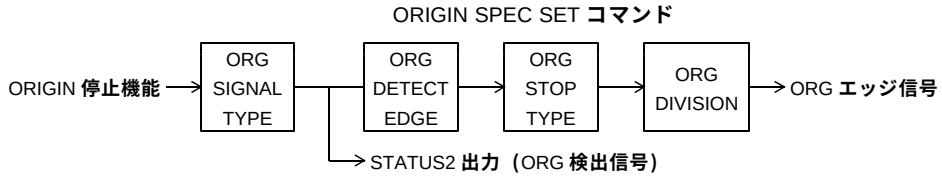


4-1-4. ORIGIN ドライブ （MCC09 チップの機械原点検出機能）

関数による ORIGIN ドライブ機能は、取扱説明書をご覧ください。

ORG エッジ信号を検出して、ドライブパルス出力を減速停止、即時停止、１パルス停止します。
また、検出のみ行うことで、MCC09 の各種機能を実行するトリガ信号として使用できます。
検出する ORG 検出信号は、$\overline{\text{ORG}}$ 信号, ± ZORG 信号, $\overline{\text{DEND}}$/$\overline{\text{PO}}$ 信号, $\overline{\text{NORG}}$ 信号の合成信号から選択します。

ORG エッジ信号の検出条件と停止機能は、ORIGIN SPEC SET コマンドで設定します。
- ORG エッジ信号による停止機能は、STATUS1 PORT の DRIVE = 1 のときに有効です。
- ORG エッジ信号の停止機能が動作すると、STATUS2 PORT の ORGEND = 1 にします。
  STATUS1 PORT の SSEND, FSEND フラグは変化しません。

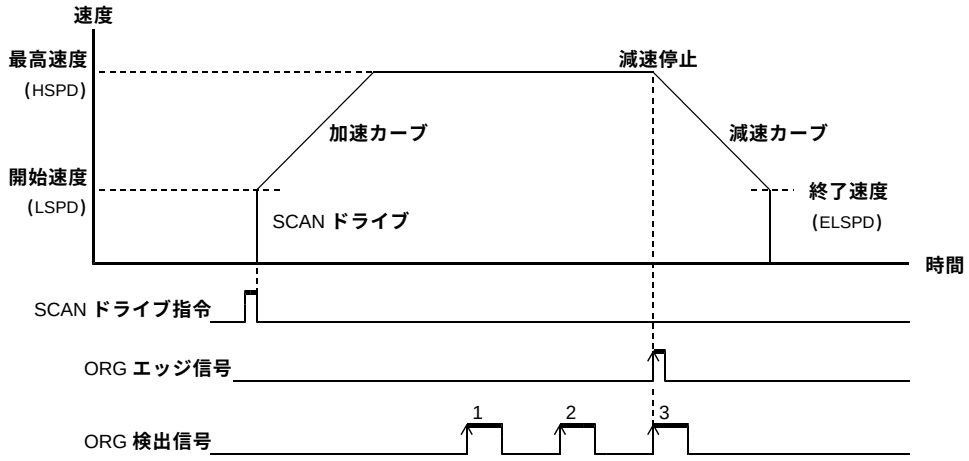


■ ORIGIN 停止機能による原点検出工程

任意のドライブを実行し、ORG エッジ信号の停止機能で原点検出を行います。

<原点検出工程の実行例>
ORG 検出信号の３カウント目のアクティブエッジ検出で、減速停止動作させます。
- ORG DETECT EDGE = 0：ORG 検出信号の 0 → 1（アクティブ）エッジを検出する
- ORG STOP TYPE = "01" ：ORG エッジ信号の停止機能を、減速停止にする
- ORG DIVISION = H'02 ：３カウント毎のエッジ信号（ORG エッジ信号）を出力する

● ORIGIN SCAN ドライブ
加減速ドライブのパラメータで、SCAN（または INDEX）ドライブを実行します。
ORIGIN 停止機能の ORG エッジ信号を検出すると、減速停止します。



● ORIGIN CONSTANT SCAN ドライブ
JSPD SCAN（または JOG）ドライブを実行します。
または、一定速ドライブのパラメータで、SCAN（または INDEX）ドライブを実行します。
ORIGIN 停止機能の ORG エッジ信号を検出すると、減速なしで停止します。

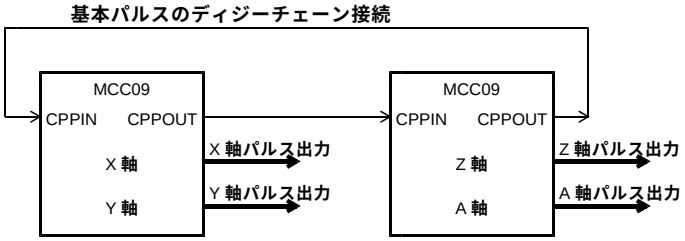
4-1-5. 補間ドライブ

補間ドライブには、１軸単位で直線補間ドライブを実行するコマンドと、
１軸単位で円弧補間ドライブを実行するコマンドがあります。
マルチチップの多軸直線補間ドライブと任意２軸の円弧補間ドライブができます。

(1) 任意軸補間ドライブ

MCC09 の CPPIN 端子と CPPOUT 端子をデイジーチェーン接続しており、
マルチチップ補間ドライブができます。

各軸 MCC09 の CPPOUT 端子と CPPIN 端子はディジーチェーン接続で繋がっています。
任意軸補間ドライブではメイン軸のチップが CPPOUT 端子に補間ドライブの基本パルスを出力します。
サブ軸のチップは基本パルスを CPPIN 端子から入力して CPPOUT 端子に出力します。



任意多軸直線補間ドライブ … MAIN STRAIGHT CP コマンド (メイン軸)
SUB STRAIGHT CP コマンド (サブ軸)
任意 2 軸円弧補間ドライブ … MAIN CIRCULAR CP コマンド (メイン軸)
SUB CIRCULAR CP コマンド (サブ軸)

補間ドライブの基本となる加減速パルスは、メイン軸に設定した加減速パラメータで発生します。
・メイン軸のチップは、基本パルスを CPPOUT 端子から出力する設定にします。
・サブ軸のチップは、基本パルスを CPPIN 端子から入力して CPPOUT 端子に出力する設定にします。

マルチチップ直線補間ドライブは、１つのメイン軸とその他のサブ軸で構成します。
・メイン軸には、メイン軸直線補間ドライブを実行します。
・サブ軸には、サブ軸直線補間ドライブを実行します。

任意２軸円弧補間ドライブは、１つのメイン軸と１つのサブ軸で構成します。
・メイン軸には、メイン軸円弧補間ドライブを実行します。
・サブ軸には、サブ軸円弧補間ドライブを実行します。

● CPPOUT 出力
CP SPEC SET コマンドの CPPOUT SEL で選択したパルスを出力します。

● CPPIN 入力
補間ドライブの基本パルスを入力します。
CPPIN 端子に入力できるパルス速度の最高値は、5 MHz です。

■ 直線補間ドライブの実行と停止機能

直線補間ドライブでは、各補間軸が独立して直線補間パルス（長軸・短軸パルス）を出力します。
・停止指令またはエラーの発生した補間軸は、パルス停止後にドライブを終了しますが、
他の補間軸はドライブを終了しません。
補間軸が停止指令またはエラーにより補間ドライブを終了した場合は、
他の補間軸に停止指令を実行して、ドライブを終了させてください。

● メイン軸直線補間ドライブを実行した場合
マルチチップの直線補間ドライブが実行できます。
・メイン軸直線補間ドライブのコマンドは、
各サブ軸にサブ軸直線補間ドライブを実行した後（または同時）に、メイン軸に実行します。
メイン軸はコマンドの実行でドライブを開始します。

・メイン軸直線補間ドライブを実行すると、コマンド実行軸の加減速パラメータで、
直線補間ドライブの基本パルスと直線補間パルスを発生します。
メイン軸直線補間ドライブでは、線速一定制御した基本パルスを発生することができます。

メイン軸直線補間ドライブ実行中に停止指令が発生した場合
・減速停止指令を検出した場合は、出力中の基本パルスを減速停止して、ドライブを終了します。
・即時停止指令を検出した場合は、出力中の基本パルスが OFF レベルのときに、
基本パルス出力を停止して、ドライブを終了します。

● サブ軸直線補間ドライブを実行した場合
CPPIN 入力によるマルチチップの直線補間ドライブが実行できます。
・サブ軸直線補間ドライブを実行すると、
CPPIN に入力するパルスを直線補間ドライブの基本パルスにして、直線補間パルスを発生します。
サブ軸直線補間ドライブは、コマンドの実行で STBY = 1 になります。
CPPIN のハイレベルを検出すると、STBY = 0、DRIVE = 1 にして、ドライブを開始します。

サブ軸直線補間ドライブ実行中に停止指令が発生した場合
・減速停止指令を検出した場合は、出力中の補間パルスが OFF レベルのときに、ドライブを終了します。
・即時停止指令を検出した場合は、出力中の補間パルスが OFF レベルのときに、ドライブを終了します。

・出力中の補間パルスが OFF レベルの場合は、そのまますぐにドライブを終了します。
・出力中の補間パルスがアクティブレベルの場合は、補間パルス出力が OFF レベルになると、
ドライブを終了します。

● CPP STOP 機能と CPPIN マスク機能によるパルス出力の停止
この機能は、補間ドライブ実行コマンドのドライブ仕様で設定します。
CPPIN 端子と CPPOUT 端子をディジーチェーン接続したマルチチップ補間ドライブで、
メイン軸の CPP STOP 機能とサブ軸の CPPIN マスク機能を有効にしてドライブを実行すると、
サブ軸にエラーが発生した場合に、すべての補間軸のパルス出力を停止させることができます。
・エラーが発生したサブ軸は、CPPIN マスク機能で CPPOUT 出力を停止します。
・エラーを検出すると、出力中の補間パルスが OFF レベルのときにドライブを終了します。
出力中の補間パルスのアクティブレベルが続く場合は、
エラー検出から 100 µs 後に、補間パルス出力を OFF レベルにしてドライブを終了します。

・メイン軸は、CPP STOP 機能でドライブを終了し、CPPOUT 出力を終了します。
・他のサブ軸は、メイン軸の CPPOUT 出力終了でパルス出力を停止します。

■ 円弧補間ドライブの実行と停止機能

円弧補間ドライブでは、各補間軸が独立して円弧補間パルス（X, Y 座標パルス）を出力します。
・停止指令またはエラーの発生した補間軸は、パルス停止後にドライブを終了しますが、
他の補間軸はドライブを終了しません。
補間軸が停止指令またはエラーにより補間ドライブを終了した場合は、
他の補間軸に停止指令を実行して、ドライブを終了させてください。

● メイン軸円弧補間ドライブを実行した場合
マルチチップの円弧補間ドライブが実行できます。
・メイン軸円弧補間ドライブのコマンドは、
サブ軸にサブ軸円弧補間ドライブを実行した後（または同時）に、メイン軸に実行します。
メイン軸はコマンドの実行でドライブを開始します。

・メイン軸円弧補間ドライブを実行すると、コマンド実行軸の加減速パラメータで、
円弧補間ドライブの基本パルスと円弧補間パルスを発生します。
メイン軸円弧補間ドライブでは、線速一定制御した基本パルスを発生することができます。

メイン軸円弧補間ドライブ実行中に停止指令が発生した場合
・減速停止指令を検出した場合は、出力中の基本パルスを減速停止して、ドライブを終了します。
・即時停止指令を検出した場合は、出力中の基本パルスが OFF レベルのときに、
基本パルス出力を停止して、ドライブを終了します。

● サブ軸円弧補間ドライブを実行した場合
CPPIN 入力によるマルチチップの円弧補間ドライブが実行できます。
・サブ軸円弧補間ドライブを実行すると、
CPPIN に入力するパルスを円弧補間ドライブの基本パルスにして、円弧補間パルスを発生します。
サブ軸円弧補間ドライブは、コマンドの実行で STBY = 1 になります。
CPPIN のハイレベルを検出すると、STBY = 0、DRIVE = 1 にして、ドライブを開始します。

サブ軸円弧補間ドライブ実行中に停止指令が発生した場合
・減速停止指令を検出した場合は、出力中の補間パルスが OFF レベルのときに、ドライブを終了します。
・即時停止指令を検出した場合は、出力中の補間パルスが OFF レベルのときに、ドライブを終了します。

・出力中の補間パルスが OFF レベルの場合は、そのまますぐにドライブを終了します。
・出力中の補間パルスがアクティブレベルの場合は、補間パルス出力が OFF レベルになると、
ドライブを終了します。

● CPP STOP 機能と CPPIN マスク機能によるパルス出力の停止
この機能は、補間ドライブ実行コマンドのドライブ仕様で設定します。
CPPIN 端子と CPPOUT 端子をディジーチェーン接続したマルチチップ補間ドライブで、
メイン軸の CPP STOP 機能とサブ軸の CPPIN マスク機能を有効にしてドライブを実行すると、
サブ軸にエラーが発生した場合に、メイン軸のパルス出力を停止させることができます。
・エラーが発生したサブ軸は、CPPIN マスク機能で CPPOUT 出力を停止します。
・エラーを検出すると、出力中の補間パルスが OFF レベルのときにドライブを終了します。
出力中の補間パルスのアクティブレベルが続く場合は、
エラー検出から 100 µs 後に、補間パルス出力を OFF レベルにしてドライブを終了します。

・メイン軸は、CPP STOP 機能でドライブを終了し、CPPOUT 出力を終了します。

(2) 直線補間ドライブ

マルチチップの多軸直線補間ドライブができます。
各補間軸は任意の長軸と短軸で座標を構成し、指定軸のパルスを出力して直線補間します。
補間ドライブの最高速度は、5 MHz です。指定直線に対する位置誤差は、± 0.5 LSB です。
座標指定できる相対アドレス範囲は、-2,147,483,647 ～ +2,147,483,647（32 ビット）です。

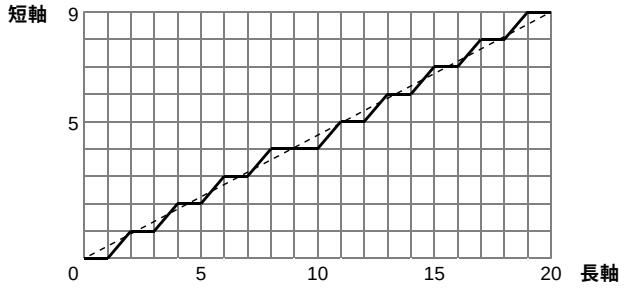
● マルチチップ用の１軸単位の補間ドライブ

H'30	MAIN STRAIGHT CP	メイン軸直線補間ドライブの実行
H'31	SUB STRAIGHT CP	サブ軸直線補間ドライブの実行

メイン軸直線補間ドライブには、以下のドライブパラメータの設定が必要です。
・実行軸の加減速ドライブのパラメータ
・CP SPEC ：CPPOUT 出力、CPPIN 入力
・LONG POSITION ：補間軸の長軸の座標アドレス
・SHORT POSITION ：補間軸の短軸の座標アドレス

サブ軸直線補間ドライブには、以下のドライブパラメータの設定が必要です。
・CP SPEC ：CPPOUT 出力、CPPIN 入力
・LONG POSITION ：補間軸の長軸の座標アドレス
・SHORT POSITION ：補間軸の短軸の座標アドレス

■ 直線補間ドライブの軌跡（長軸 20：短軸 9 の例）



直線補間ドライブの軌跡は、現在位置と目的地を結ぶ直線に沿います。
・直線補間 SCAN ドライブの場合は、停止指令を検出するまで目的地の指定方向にパルス出力を続けます。
・直線補間 INDEX ドライブの場合は、長軸のパルス数が目的地のパルス数になるとドライブを終了します。

● 直線補間の長軸と短軸
補間パルス数が大きい方の軸が長軸、小さい方の軸が短軸になります。

(3) 円弧補間ドライブ

マルチチップの任意２軸円弧補間ドライブができます。
現在の座標と中心点で形成する円弧曲線上を、指定の短軸パルス数に達するまで円弧補間します。
補間ドライブの最高速度は、5 MHz です。指定円弧曲線に対する位置誤差は、± 1 LSB です。
座標指定できる相対アドレス範囲は、-8,388,608 ～ +8,388,607（24 ビット）です。
短軸パルス数の設定範囲は、-2,147,483,647 ～ +2,147,483,647（32 ビット）です。

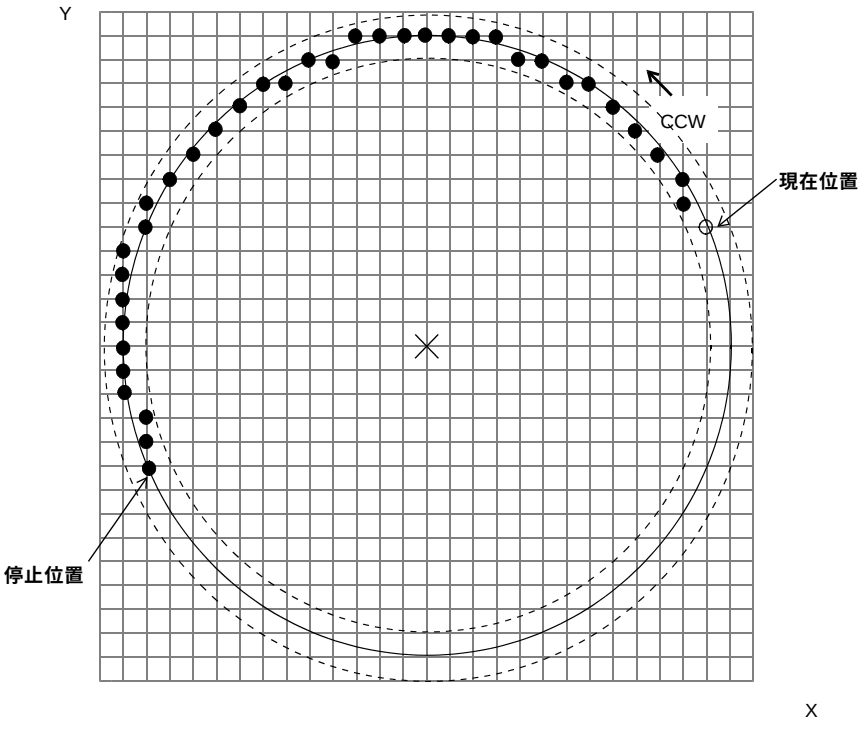
● 任意２軸用の１軸単位の円弧補間ドライブ

H'38	MAIN CIRCULAR CP	メイン軸円弧補間ドライブの実行
H'39	SUB CIRCULAR CP	サブ軸円弧補間ドライブの実行

メイン軸円弧補間ドライブには、以下のドライブパラメータの設定が必要です。
・実行軸の加減速ドライブのパラメータ
・CP SPEC ：CPPOUT 出力、CPPIN 入力
・CIRCULAR XPOSITION ：現在位置の X 座標アドレス
・CIRCULAR YPOSITION ：現在位置の Y 座標アドレス
・CIRCULAR PULSE ：目的地の短軸座標までの短軸パルス数

サブ軸円弧補間ドライブには、以下のドライブパラメータの設定が必要です。
・CP SPEC ：CPPOUT 出力、CPPIN 入力
・CIRCULAR XPOSITION ：現在位置の X 座標アドレス
・CIRCULAR YPOSITION ：現在位置の Y 座標アドレス
・CIRCULAR PULSE ：目的地の短軸座標までの短軸パルス数

■ 円弧補間ドライブの軌跡（CCW 回転の例）



円弧補間ドライブの軌跡は、現在位置と円弧の中心点の距離を半径とした円周に沿います。
・円弧補間 SCAN ドライブの場合は、停止指令を検出するまで指定の円弧半径と回転方向で
パルス出力を続けます。
・円弧補間 INDEX ドライブの場合は、短軸パルス数が指定の短軸パルス数になるとドライブを終了します。

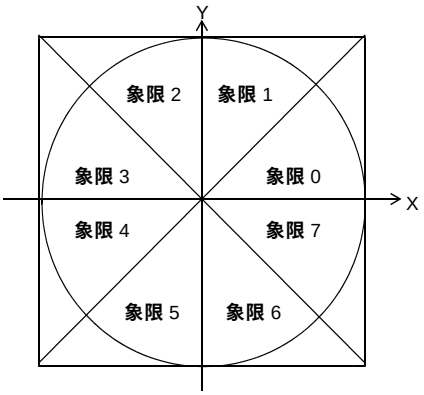
●円弧補間の短軸

円弧補間の中心点(0, 0)としたときに

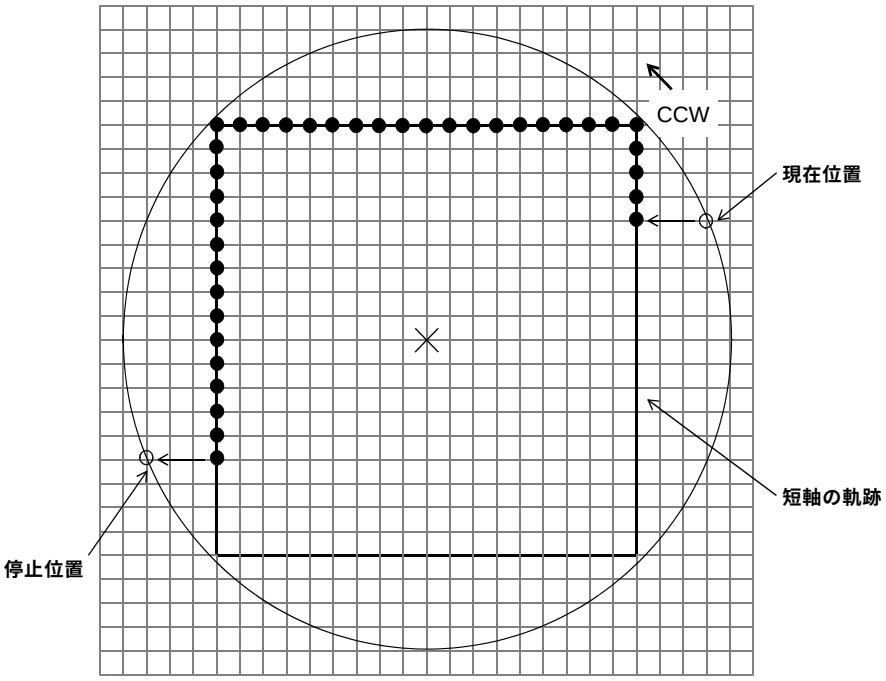
補間座標(X, Y)の絶対値が小さい方の軸が短軸になります。

右図の 1, 2, 5, 6 象限は X 軸が短軸です。

0, 3, 4, 7 象限は Y 軸が短軸です。



■円弧補間ドライブ短軸パルスの軌跡(CCW 回転の例)



円弧補間ドライブは、円弧の中心座標からみた短軸側が補間ドライブの基本パルス（短軸パルス）を
常に出力し、長軸側は基本パルス（短軸パルス）を補間演算して補間パルスを出力します。

●短軸パルス数の計算式
半径 R の円における１象限当たりの短軸パルス数 Ps は、以下の条件式で算出します。

K ＝ int(R ／√ 2)　：int() は小数点以下を切り捨てた整数

(1)$R^2 \leqq K^2 + (K+1)^2$ のとき

条件式： $|K^2 + (K+1)^2 - R^2| > |K^2 + K^2 - R^2|$ のときは、Ps ＝ K
$|K^2 + (K+1)^2 - R^2| \leqq |K^2 + K^2 - R^2|$ のときは、Ps ＝ K ＋ 1/2

(2)$R^2 > K^2 + (K+1)^2$ のとき

条件式： $|K^2 + (K+1)^2 - R^2| > |(K+1)^2 + (K+1)^2 - R^2|$ のときは、Ps ＝ K ＋ 1
$|K^2 + (K+1)^2 - R^2| \leqq |(K+1)^2 + (K+1)^2 - R^2|$ のときは、Ps ＝ K ＋ 1/2

短軸パルス数 P ＝ int(各象限の短軸パルス数の合計)

1 象限当たりの短軸パルス数の 8 倍（Ps x 8）が、1 回転のパルス数になります。

●短軸パルス数の計算例 1（CCW 回転）
中心点（0, 0）に対して、現在位置を（10, 5）、目的地を（-10, -5）として CCW 回転させる場合の目的地の短軸座標までの短軸パルス数 P は、以下のようになります。

R ＝ $\sqrt{(10^2 + 5^2)}$ ＝ √ 125,　K ＝ int(R ／√ 2) ＝ int(7.9) ＝ 7

R^2 ＝ 125,　$K^2 + (K+1)^2$ ＝ 113,　なので　$R^2 > K^2 + (K+1)^2$

$|K^2 + (K+1)^2 - R^2|$ ＝ 12,　$|(K+1)^2 + (K+1)^2 - R^2|$ ＝ 3, なので

$|K^2 + (K+1)^2 - R^2| > |(K+1)^2 + (K+1)^2 - R^2|$ のときは、Ps ＝ K ＋ 1 ＝ 8

短軸パルス数 P ＝ int(象限 0 のパルス数 ＋ 象限 1, 2, 3 のパルス数 ＋ 象限 4 のパルス数)
＝ (8 － 5) ＋ (8 ＋ 8 ＋ 8) ＋ 5 ＝ 32

CCW 回転は負数で指定するので、CIRCULAR PULSE ＝ -32 ＝ H'FFFF_FFE0

CCW 回転時の現在位置の象限の短軸パルス数は、以下のようになります。
・現在位置が象限 0, 2, 4, 6 の短軸パルス数： Ps －(現在位置の短軸座標の絶対値)
・現在位置が象限 1, 3, 5, 7 の短軸パルス数： 現在位置の短軸座標の絶対値

CCW 回転時の目的位置の象限の短軸パルス数は、以下のようになります。
・目的位置が象限 0, 2, 4, 6 の短軸パルス数： 目的位置の短軸座標の絶対値
・目的位置が象限 1, 3, 5, 7 の短軸パルス数： Ps －(目的位置の短軸座標の絶対値)

●短軸パルス数の計算例２（CW 回転）
中心点（0, 0）に対して、現在位置を（20, 5）、目的地を（-20, -5）として CW 回転させる場合の目的地の短軸座標までの短軸パルス数 P は、以下のようになります。

R ＝ $\sqrt{(20^2 + 5^2)}$ ＝ √ 425,　K ＝ int(R ／√ 2) ＝ int(14.6) ＝ 14

R^2 ＝ 425,　$K^2 + (K+1)^2$ ＝ 421,　なので　$R^2 > K^2 + (K+1)^2$

$|K^2 + (K+1)^2 - R^2|$ ＝ 4,　$|(K+1)^2 + (K+1)^2 - R^2|$ ＝ 25,　なので

$|K^2 + (K+1)^2 - R^2| \leqq |(K+1)^2 + (K+1)^2 - R^2|$ のときは、Ps ＝ K ＋ 1/2 ＝ 14.5

短軸パルス数 P ＝ int(象限 0 のパルス数 ＋ 象限 7, 6, 5 のパルス数 ＋ 象限 4 のパルス数)
＝ 5 ＋ (14.5 ＋ 14.5 ＋ 14.5) ＋ (14.5 － 5) ＝ 58

CW 回転は正数で指定するので、CIRCULAR PULSE ＝ 58 ＝ H'0000_003A

CW 回転時の現在位置の象限の短軸パルス数は、以下のようになります。
・現在位置が象限 0, 2, 4, 6 の短軸パルス数： 現在位置の短軸座標の絶対値
・現在位置が象限 1, 3, 5, 7 の短軸パルス数： Ps －(現在位置の短軸座標の絶対値)

CW 回転時の目的位置の象限の短軸パルス数は、以下のようになります。
・目的位置が象限 0, 2, 4, 6 の短軸パルス数： Ps －(目的位置の短軸座標の絶対値)
・目的位置が象限 1, 3, 5, 7 の短軸パルス数： 目的位置の短軸座標の絶対値

(4) 線速一定制御

メイン軸直線補間ドライブで有効です。
メイン軸円弧補間ドライブで有効です。

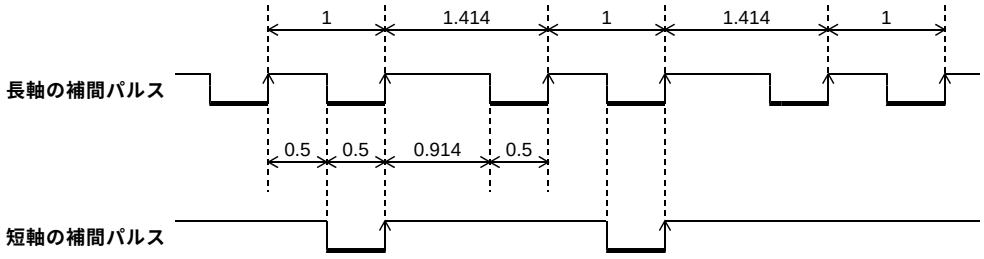
補間ドライブしている２軸の合成速度を一定にする制御です。

メイン軸直線／円弧補間ドライブの基本パルスを線速一定制御します。
２軸同時にパルス出力したときに、次の基本パルスの出力周期を 1.414 倍にします。
・直線補間ドライブでは、メイン軸の長軸と短軸の２軸間で、線速一定制御します。
・円弧補間ドライブでは、X 座標軸と Y 座標軸の２軸間で、線速一定制御します。

線速一定で加減速ドライブを行うと、減速パルス数が増加します。
INDEX ドライブの自動減速では、減速後の終了速度でのドライブが長くなります。

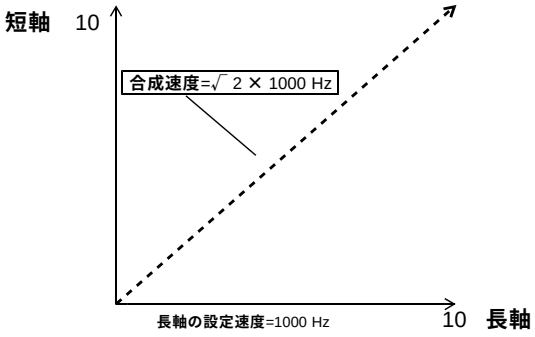
■ 線速一定の補間パルス出力（２軸直線補間ドライブの例）

アクティブレベルの幅はそのままで、OFF レベルの幅を長くします。



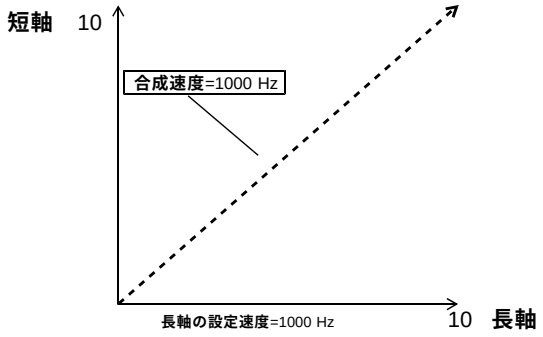
直線補間ドライブの軌跡(長軸 10:短軸 10 の例)

<線速一定制御なしのとき>



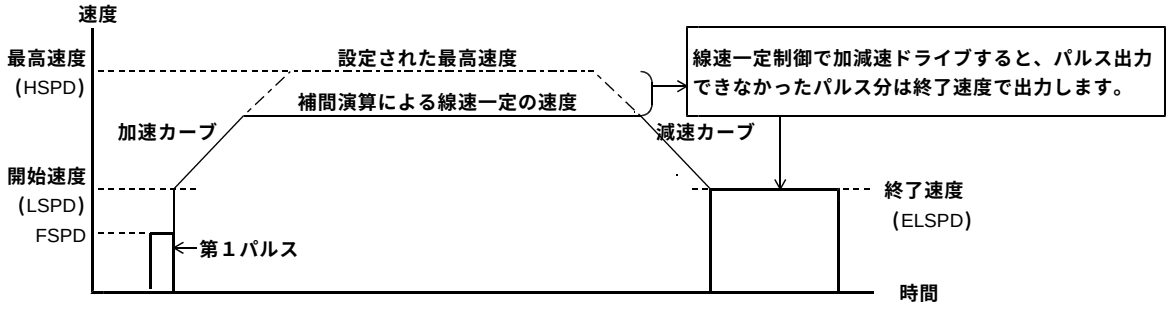
長軸の速度を一定速の 1000Hz とすると、
2 軸直線補間で描かれる軌跡の合成速度は
√ 2 × 1000Hz でドライブします。

<線速一定制御ありのとき>



コマンド実行軸の速度を一定速の 1000Hz として
線速一定制御を設定すると、2 軸直線補間で
描かれる軌跡の合成速度が 1000Hz となるように
ドライブします。

※線速一定で加減速ドライブを行うと、減速後の終了速度でのドライブが長くなります。



4-1-6. ドライブ CHANGE 機能

ドライブ CHANGE 機能には、SPEED CHANGE 機能と INDEX CHANGE 機能があります。
ドライブ CHANGE 指令は、予約して実行することができます。

(1) SPEED CHANGE 機能

SPEED CHANGE 指令には、以下のコマンドがあります。

・設定コマンド
SPEED CHANGE SPEC SET コマンド：SPEED CHANGE 指令を実行する変更動作点
・実行コマンド
SPEED RATE CHANGE コマンド　　：SPEED & RATE CHANGE 指令

SPEED CHANGE の状態は、STATUS5 PORT の SPEED EP と SPEED FL フラグで確認します。
・SPEED FL = 0 のときに、変更動作点を設定して、SPEED RATE CHANGE コマンドを書き込みます。
SPEED CHANGE SPEC SET コマンドを実行しても、SPEED FL = 0 のままです。
SPEED RATE CHANGE コマンドを書き込むと、SPEED CHANGE 専用のレジスタに格納します。
同時に SPEED FL = 1、SPEED EP = 0 にします。
・DRIVE = 1 のときに、変更動作点を検出して、SPEED RATE CHANGE コマンドを実行します。
同時に SPEED FL = 0、SPEED EP = 1 にします。

変更動作点を検出して、SPEED RATE CHANGE コマンドを実行すると、
指定したドライブパルス速度まで（SPEED CHANGE）、
指定した変速周期データで（RATE CHANGE）、加速または減速します。
・指定する速度は、最高速度以上および開始速度／終了速度以下にできます。

・直線加減速ドライブ（SUAREA = 0, SDAREA = 0）の加減速中に CHANGE 指令を実行した場合は、
現在の速度から、指定した速度まで指定した変速周期で加速または減速します。

・S 字加減速ドライブの加減速中に CHANGE 指令を実行した場合は、以下のようになります。
S 字加減速ドライブでは、現在の加減速カーブを終了するまでは SPEED FL = 1 のままです。

現在の加速または減速状態で、指定した速度に到達できる場合は、
現在の変速周期のまま加速または減速します。この場合は、指定した変速周期は無効になります。

現在の加速または減速状態で、指定した速度に到達できない場合は、
現在の加速または減速状態を、現在の変速周期の S 字カーブで滑らかに終了させてから、
指定した速度まで指定した変速周期で加速または減速します。

・減速停止動作時および自動減速動作中の INDEX CHANGE による再加速時の変速周期は、
RATE SET コマンドで設定した UCYCLE または DCYCLE の変速周期で動作します。

指定するドライブパルス速度を "0" に設定すると、CONST DRIVE 指令になります。
CONST DRIVE 指令を実行すると、加速または減速を終了して、一定速にします。
・直線加減速ドライブの加減速中の場合は、CONST DRIVE 指令の実行で加速または減速を終了します。

・S 字加減速ドライブの加減速中の場合は、CONST DRIVE 指令を実行すると、
現在の変速周期の S 字カーブで滑らかに加速または減速を終了して、一定速にします。

SPEED RATE CHANGE コマンドを実行しても、ドライブパラメータの設定は変わりません。

■ SPEED CHANGE 指令が有効となるドライブ（コマンド）

補間ドライブでは、基本パルス発生軸の加減速パラメータに対して CHANGE 指令を実行します。

H'12	+SCAN	*P	＋方向 SCAN ドライブの実行
H'13	-SCAN	*P	－方向 SCAN ドライブの実行
H'14	INC INDEX	*P	相対アドレス INDEX ドライブの実行
H'15	ABS INDEX	*P	絶対アドレス INDEX ドライブの実行
H'30	MAIN STRAIGHT CP	*P	メイン軸直線補間ドライブの実行
H'38	MAIN CIRCULAR CP	*P	メイン軸円弧補間ドライブの実行
－	－	*P	MANUAL ドライブの SCAN ドライブの実行

・上記のドライブを実行すると DRIVE = 1 から、変更動作点の検出が有効になります。
　変更動作点を検出すると、格納している SPEED RATE CHANGE コマンドを実行します。

● SPEED CHANGE 指令が自動的に無効となる状態
以下の状態を検出すると、格納している SPEED RATE CHANGE コマンドをクリアします。
SPEED FL = 1、SPEED EP = 1 にして、SPEED CHANGE コマンドの書き込みを無効にします。
・SPEC INITIALIZE3 コマンドの SCAREA MODE = 1 の検出
・STATUS1 PORT の ERROR = 1 の検出
・DRIVE = 1 のときに、LSEND = 1、SSEND = 1、FSEND = 1、ORGEND = 1 の検出
・INDEX ドライブの自動減速地点の検出
・反転動作が必要な INDEX CHANGE 指令の検出
・DRIVE = 1 → 0 の検出（格納している実行コマンドのクリアのみ行います）
・SERVO RESET コマンドの実行

SCAREA MODE = 0、ERROR = 0、DRIVE = 0 になると、
SPEED FL = 0、SPEED EP = 1 にして、SPEED CHANGE コマンドの書き込みを有効にします。

(2) INDEX CHANGE 機能

INDEX CHANGE 指令には、以下のコマンドがあります。

・設定コマンド
INDEX CHANGE SPEC SET コマンド ：INDEX CHANGE 指令を実行する変更動作点
反転ドライブの第１パルス周期（RFSPD）

・実行コマンド
INC INDEX CHANGE コマンド ：INC INDEX CHANGE 指令
ABS INDEX CHANGE コマンド ：ABS INDEX CHANGE 指令
PLS INDEX CHANGE コマンド ：PLS INDEX CHANGE 指令

INDEX CHANGE の状態は、STATUS5 PORT の INDEX EP と INDEX FL フラグで確認します。
・INDEX FL = 0 のときに、変更動作点を設定して、INDEX CHANGE 実行コマンドを書き込みます。
INDEX CHANGE SPEC SET コマンドを実行しても、INDEX FL = 0 のままです。
INDEX CHANGE 実行コマンドを書き込むと、INDEX CHANGE 専用のレジスタに格納します。
同時に INDEX FL = 1、INDEX EP = 0 にします。

・DRIVE = 1 のときに、変更動作点を検出して、INDEX CHANGE 実行コマンドを実行します。
同時に INDEX FL = 0、INDEX EP = 1 にします。

任意の変更動作点の検出で、INDEX CHANGE コマンドを実行します。
・変更動作点を検出して、INC INDEX CHANGE コマンドを実行すると、
指定したデータを、起動位置を原点とする相対アドレスの停止位置に設定して、
INC INDEX ドライブを行います。

・変更動作点を検出して、ABS INDEX CHANGE コマンドを実行すると、
指定したデータを、アドレスカウンタで管理している絶対アドレスの停止位置に設定して、
ABS INDEX ドライブを行います。

・変更動作点を検出して、PLS INDEX CHANGE コマンドを実行すると、
指定したデータを、変更動作点の検出位置を原点とする相対アドレスの停止位置に設定して、
INC INDEX ドライブを行います。

PLS INDEX CHANGE コマンドを実行すると、INC INDEX ドライブの原点を変更します。
・PLS INDEX CHANGE コマンド実行後に、INC INDEX CHANGE コマンドを実行すると、
PLS INDEX CHANGE コマンドの変更動作点の検出位置を原点として INC INDEX ドライブを行います。

反転動作が必要な INDEX CHANGE 指令を検出した場合は、
終了速度まで減速停止後に反転ドライブを行い、指定の停止位置まで移動します。

■ INDEX CHANGE 指令が有効となるドライブ（コマンド）

H'12	+SCAN	*P	＋方向 SCAN ドライブの実行
H'13	-SCAN	*P	－方向 SCAN ドライブの実行
H'14	INC INDEX	*P	相対アドレス INDEX ドライブの実行
H'15	ABS INDEX	*P	絶対アドレス INDEX ドライブの実行

・上記のドライブを実行すると DRIVE = 1 から、変更動作点の検出が有効になります。
変更動作点を検出すると、格納している INDEX CHANGE 実行コマンドを実行します。

● INDEX CHANGE 指令が自動的に無効となる状態
以下の状態を検出すると、格納している INDEX CHANGE 実行コマンドをクリアします。
INDEX FL = 1、INDEX EP = 1 にして、INDEX CHANGE コマンドの書き込みを無効にします。
・SPEC INITIALIZE3 コマンドの DOWN PULSE MASK = 1 の検出
・SPEC INITIALIZE3 コマンドの SCAREA MODE = 1 の検出
・STATUS1 PORT の ERROR = 1 の検出
・DRIVE = 1 のときに、LSEND = 1、SSEND = 1、FSEND = 1、ORGEND = 1 の検出
・DRIVE = 1 → 0 の検出（格納している実行コマンドのクリアのみ行います）
・SERVO RESET コマンドの実行

DOWN PULSE MASK = 0、SCAREA MODE = 0、ERROR = 0、DRIVE = 0 になると、
INDEX FL = 0、INDEX EP = 1 にして、INDEX CHANGE コマンドの書き込みを有効にします。

● INDEX CHANGE 指令によるエラー
内部カウンタの相対アドレス範囲は、-2,147,483,647 ～ +2,147,483,647（32 ビット）です。
出力パルス数範囲は、0 ～ 2,147,483,647（31 ビット）です。

反転動作が必要な INDEX CHANGE 指令を検出して、反転ドライブを行う場合に、
反転ドライブの出力パルス数が 2,147,483,647 を超えて、オーバフローしてしまう場合があります。
この場合は、ERROR STATUS の INDEX ERROR = 1 にします。
STATUS1 PORT の ERROR = 1 となり、実行中のドライブを終了速度まで減速して停止します。
・INDEX CHANGE 指令検出後の反転ドライブで、出力パルス数がオーバフローした

SCAN ドライブの実行または反転動作が必要な INC INDEX CHANGE 指令の実行により、
内部カウンタの相対アドレスがオーバフローしてしまう場合があります。
この場合は、ERROR STATUS の INDEX ERROR = 1 にします。
STATUS1 PORT の ERROR = 1 となり、実行中のドライブを終了速度まで減速して停止します。
・INC INDEX CHANGE 指令検出後のドライブで、内部の相対アドレスがオーバフローした

SCAN ドライブの実行または反転動作が必要な ABS INDEX CHANGE 指令の実行により、
アドレスカウンタがオーバフローしてしまう場合があります。
この場合は、ERROR STATUS の INDEX ERROR = 1 にします。
STATUS1 PORT の ERROR = 1 となり、実行中のドライブを終了速度まで減速して停止します。
・ABS INDEX CHANGE 指令検出後のドライブで、アドレスカウンタがオーバフローした

■ INDEX CHANGE コマンド実行前と実行後のドライブの停止

INDEX CHANGE コマンド実行前と実行後では、INDEX ドライブの減速停止動作が異なります。

● INDEX CHANGE コマンド実行前の INDEX ドライブの停止
・自動減速停止動作開始後に指定アドレスを検出すると即時停止します。

・ドライブ中に減速停止指令を検出した場合は、終了速度まで減速して指定アドレス内で停止します。
減速中に指定アドレスを検出した場合は、指定アドレスで即時停止します。

・ドライブ中に ERROR = 1 を検出した場合は、終了速度まで減速して指定アドレス内で停止します。
減速中に指定アドレスを検出した場合は、指定アドレスで即時停止します。

● INDEX CHANGE コマンド実行後の INDEX ドライブの停止
・INDEX CHANGE コマンドの実行で "指定アドレスの検出で停止" をマスクします。
終了速度になるとマスクを一時解除します。DRIVE = 0 の検出でマスクを完全に解除します。

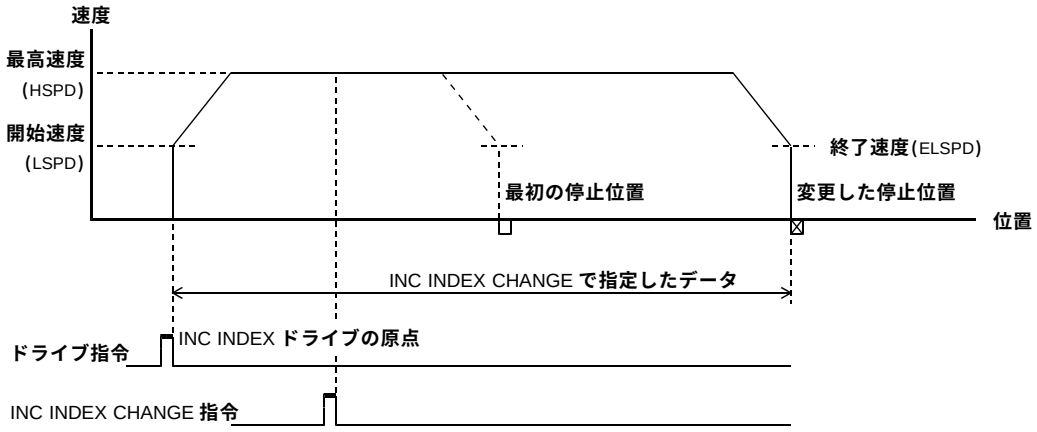
・自動減速停止動作開始後は、終了速度まで減速すると指定アドレスを検出して停止します。
ドライブ中に指定アドレスを超えた場合は、終了速度まで減速停止後に反転ドライブを行います。
反転ドライブでは、自動減速停止動作開始後に指定アドレスを検出すると即時停止します。

・ドライブ中に減速停止指令を検出した場合は、終了速度まで減速して停止します。
減速中に指定アドレスを検出しても停止しません。終了速度で停止します。

・ドライブ中に ERROR = 1 を検出した場合は、終了速度まで減速して停止します。
減速中に指定アドレスを検出しても停止しません。終了速度で停止します。

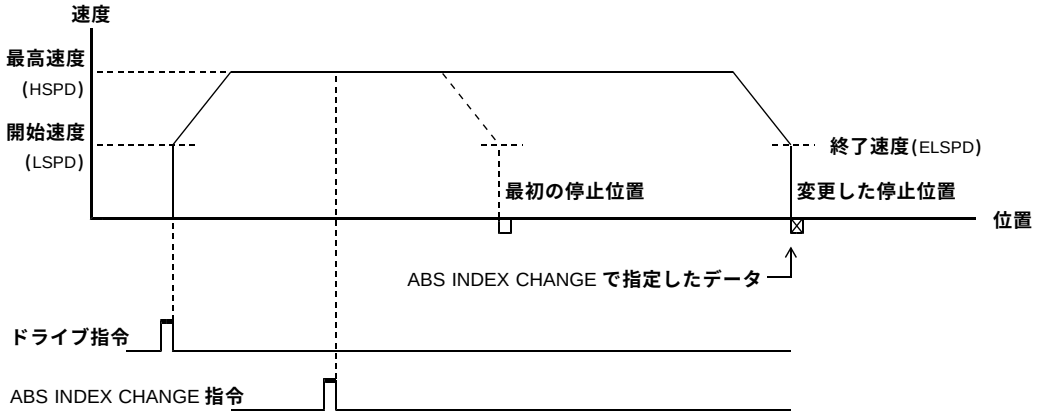
■ INC INDEX CHANGE の動作

指定したデータを、起動位置を原点とする相対アドレスの停止位置にします。



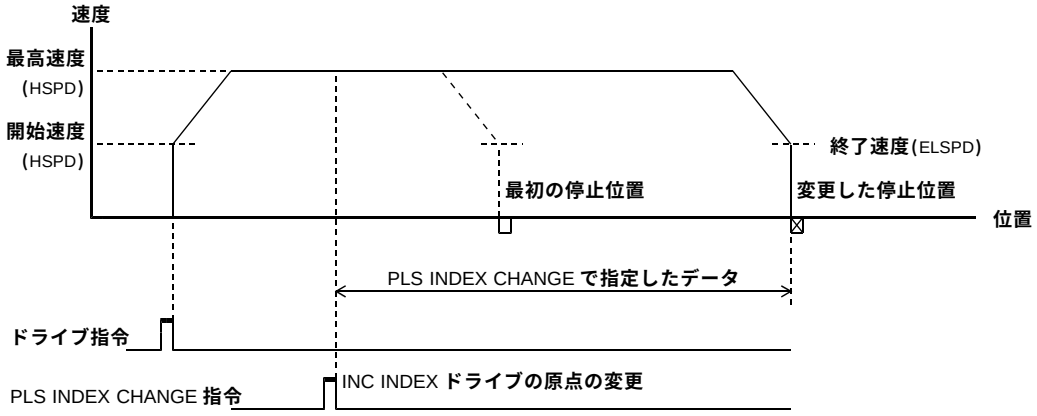
■ ABS INDEX CHANGE の動作

指定したデータを、アドレスカウンタが管理している絶対アドレスの停止位置にします。



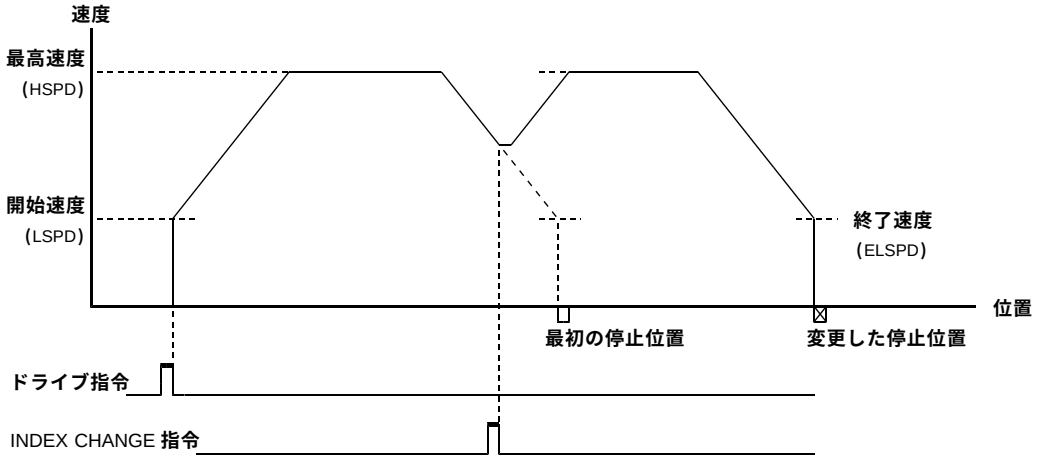
■ PLS INDEX CHANGE の動作

指定したデータを、変更動作点の検出位置を原点とする相対アドレスの停止位置にします。

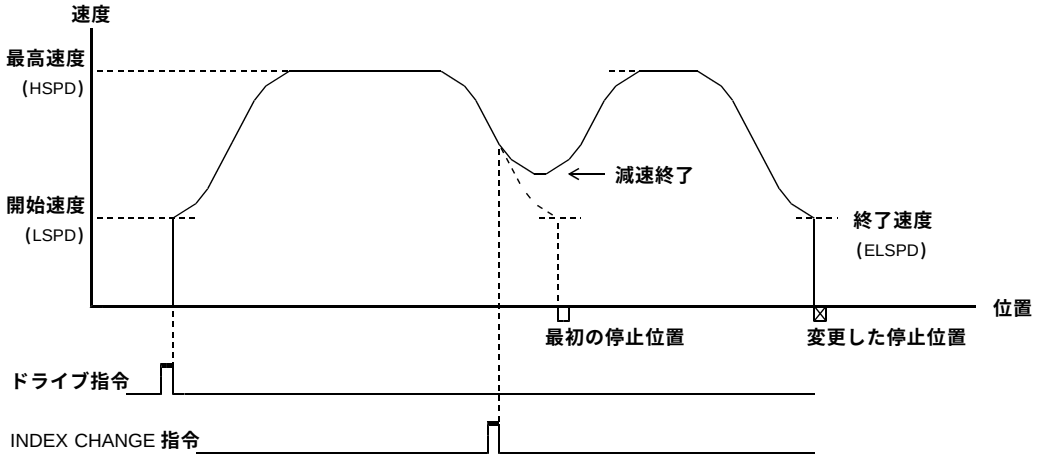


■ 減速中の INDEX CHANGE 動作

● 直線加減速ドライブの減速中の INDEX CHANGE
停止位置への減速中に、加速が必要な INDEX CHANGE 指令を検出した場合は、
減速の途中から再加速して、変更した停止位置までドライブします。

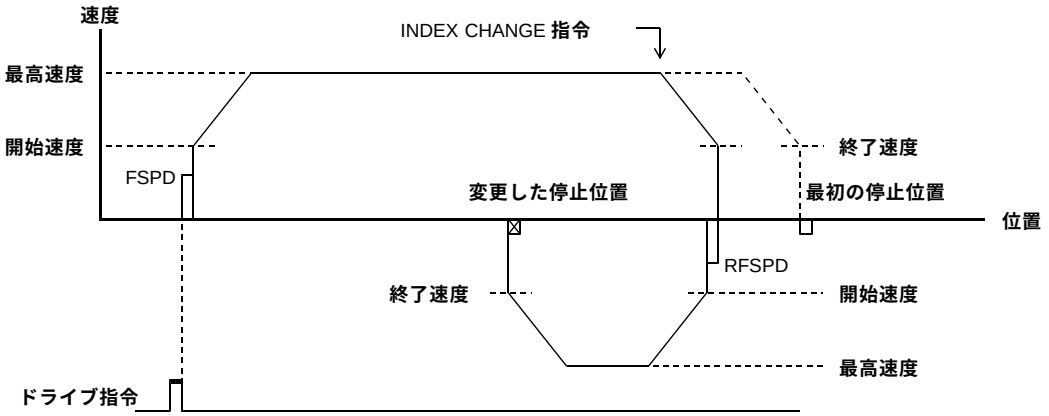


● S字加減速ドライブの減速中の INDEX CHANGE
停止位置への減速中に、加速が必要な INDEX CHANGE 指令を検出した場合は、
S字減速カーブで滑らかに減速を終了させてから、S字加速カーブで再加速します。



■ INDEX CHANGE の反転ドライブ

終了速度まで減速停止後に反転ドライブを行い、指定の停止位置まで移動します。



● INDEX CHANGE 指令による反転ドライブ

反転動作が必要な INDEX CHANGE 指令を検出した場合は、以下のようになります。

・INDEX CHANGE コマンドの実行で "指定アドレスの検出で停止" をマスクします。
終了速度になるとマスクを一時解除します。DRIVE = 0 の検出でマスクを完全に解除します。

・実行中のドライブを終了速度まで減速して停止します。
  ・STATUS1 PORT の DRIVE = 1 → 0 になります。
  ・DRIVE = 0 の検出で "指定アドレスの検出で停止" のマスクを完全に解除します。
  ・DEND 信号の<サーボ対応>も実行します。
  ・この間は、BUSY = 1 のままです。

・LSEND = 0、SSEND = 0、FSEND = 0、ORGEND = 0、ERROR = 0 のときに、
指定アドレスを超えて停止した場合は、反転ドライブを実行し、指定の停止位置まで移動します。
  ・反転ドライブの第１パルス周期は、FSPD ではなく RFSPD になります。
  RFSPD は、INDEX CHANGE SPEC SET コマンドで設定します。
  その他のドライブパラメータは、INC/ABS INDEX ドライブの設定値で動作します。
  ・反転ドライブの自動減速停止動作開始後に指定アドレスを検出すると即時停止します。
  ・停止後には、DEND 信号の<サーボ対応>も実行します。

反転ドライブ中に、反転動作が必要な INDEX CHANGE 指令を検出した場合は、
上記の動作を繰り返します。

以下の状態を検出した場合は、指定アドレスを超えて停止しても、反転ドライブを実行しません。
・STATUS1 PORT の ERROR = 1、LSEND = 1、SSEND = 1、FSEND = 1 の検出
・STATUS2 PORT の ORGEND = 1 の検出

4-1-7. 読み出し機能

下記データの読み出しは、取扱説明書(標準)の 3-7.章の制限事項を除き、常時可能です。

下記の関数にて、コマンドの書き込みと、データの読み出しを一括で処理することができます。
これらの関数は、コマンドの書き込みからデータの読み出しまで、関数内で排他処理されています。
マルチスレッドプログラミングのように、複数のスレッドで処理される場合は、この関数によって
読み出しすることを推奨します。

・デバイス単位 … DRIVE COMMAND 32 ビット書き込み／読み出し関数
DRIVE COMMAND PORT 書き込み／読み出し関数

(1) カウントデータのラッチデータ読み出し

MCCC09 に各カウンタ LATCH DATA READ コマンドを実行するとラッチしたカウントデータを
読み出すことができます。

4-1-8. RSPD 機能

RSPD とは、ドライブ終了時に MCC09 が記憶している終了速度です。

RSPD 設定方法

・JOG パルス速度(JSPD)を "0" に設定すると、JSPD ＝ RSPD x RESOL に補正して JSPD が設定されます。
・第 1 パルス周期(FSPD)を "0" に設定すると、FSPD ＝ RSPD x RESOL に補正して FSPD が設定されます。
・INDEX CHANGE の反転開始速度(RFSPD)を "0" に設定すると、RFSPD ＝ RSPD x RESOL に補正して RFSPD が設定されます。
・最高速度(HSPD)を "0" に設定すると、HSPD ＝ RSPD x RESOL に補正して HSPD が設定されます。
・最低速度(LSPD)を "0" に設定すると、LSPD ＝ RSPD x RESOL に補正して LSPD が設定されます。
終了速度(ELSPD)を "0" に設定した場合は、ELSPD は、ELSPD = LSPD に補正されます。

但し、以下の場合は、RSPD は書き換えません。
・最終出力のパルス速度が FSPD のとき
・最終出力のパルス速度が JSPD のとき
RSPD の電源投入後の初期値は、H'012C（300）です。

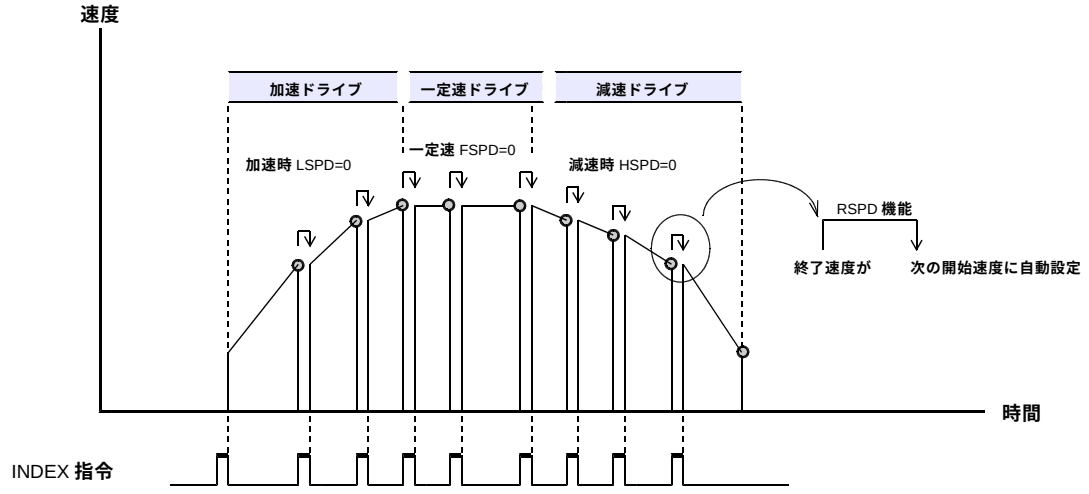
RSPD の読み出し

RSPD DATA READ コマンドにより、FPSD、HSPD、LSPD と同様のパルス速度データが読み出しできます。

●応用例

連続したドライブなど、2 回目の開始速度を MCC09 が記憶している速度でドライブを再開して、変速のないドライブパターンのつなぎ方が可能です。
一回の位置決め量が少ないときは、加速ドライブパターンで設定した最高速度、減速ドライブパターンで設定した終了速度まで到達せずに、ドライブを終了する可能性があります。
このときの終了した速度は、通常アプリケーション側から再設定することはできません。
これを RSPD 機能により、前の区間のドライブの終了速度を、次のドライブ開始速度として、自動的に再設定することを可能にしています。
RSPD 機能と加減速時定数(RATE)の変更や任意なドライブパターンを組み合わせてコマンド予約機能を使用すると、任意なドライブ形状を作ることが可能です。

下記の例で示す○印が MCC09 が記憶する終了速度(RSPD)です。



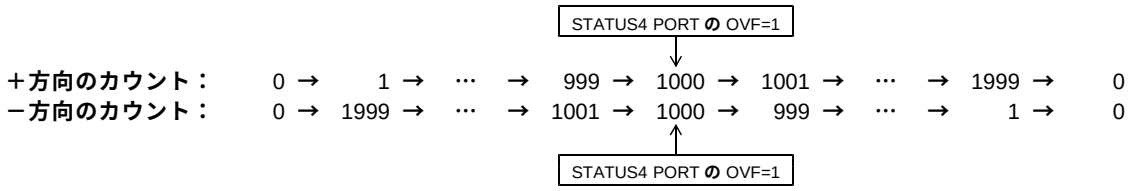
(SPEED パラメータや移動量をコマンド予約で変更)

4-2. カウンタ仕様

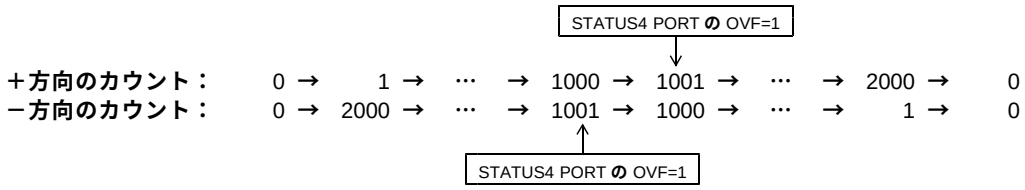
4-2-1. リングカウンタ機能

アドレスカウンタ、およびパルスカウンタは、カウント最大値を設定すると、設定値をカウンタの最大値としてリングカウントします。
回転系のアドレス管理に便利です。

●最大カウント数＝ 1,999 の場合(2,000 カウントで 1 回転)



●最大カウント数＝ 2,000 の場合(2,001 カウントで 1 回転)



・パルス偏差カウンタにはリングカウンタ機能はありません。

・カウント数が設定値の 1/2 に達すると、STATUS4 PORT の各カウンタの OVF=1 になります。

・カウント最大値は各カウンタの COUNTER MAX COUNT SET コマンドで設定します。

・アドレスカウンタにカウント最大値を設定して、ABS INDEX ドライブでカウント最大値以上のアドレスを指定した場合、アドレスカウンタが OVF した時点でドライブは停止し、エラーとなります。

4-2-2. カウントデータのラッチ・クリア機能

■カウンタのラッチ機能

設定したラッチタイミングのアクティブエッジで、カウンタのカウントデータをラッチします。
ラッチしたデータは次のラッチタイミングのアクティブエッジを検出するまで保存します。
ラッチしたデータは各カウンタ LATCH DATA READ コマンドで読み出すことができます。

●設定できるラッチタイミング

ラッチタイミング <エッジ検出>
・各カウンタ LATCH DATA READ コマンドの実行でラッチする
・他軸の STATUS3 PORT の OUT3 = 0 → 1 でラッチする
・STATUS3 PORT の GPIO2 = 0 → 1 でラッチする
・ORIGIN 停止機能の ORG エッジ信号の検出でラッチする
・STATUS3 PORT の OUT2 = 0 → 1 でラッチする
・STATUS3 PORT の OUT3 = 0 → 1 でラッチする

・各カウンタのラッチタイミングは COUNT LATCH SPEC SET コマンドで設定します。

(注)GPIO2(SS0)信号は SPEC INITIALIZE2 コマンドで SS0 機能が「汎用入力」に設定している場合に有効です。

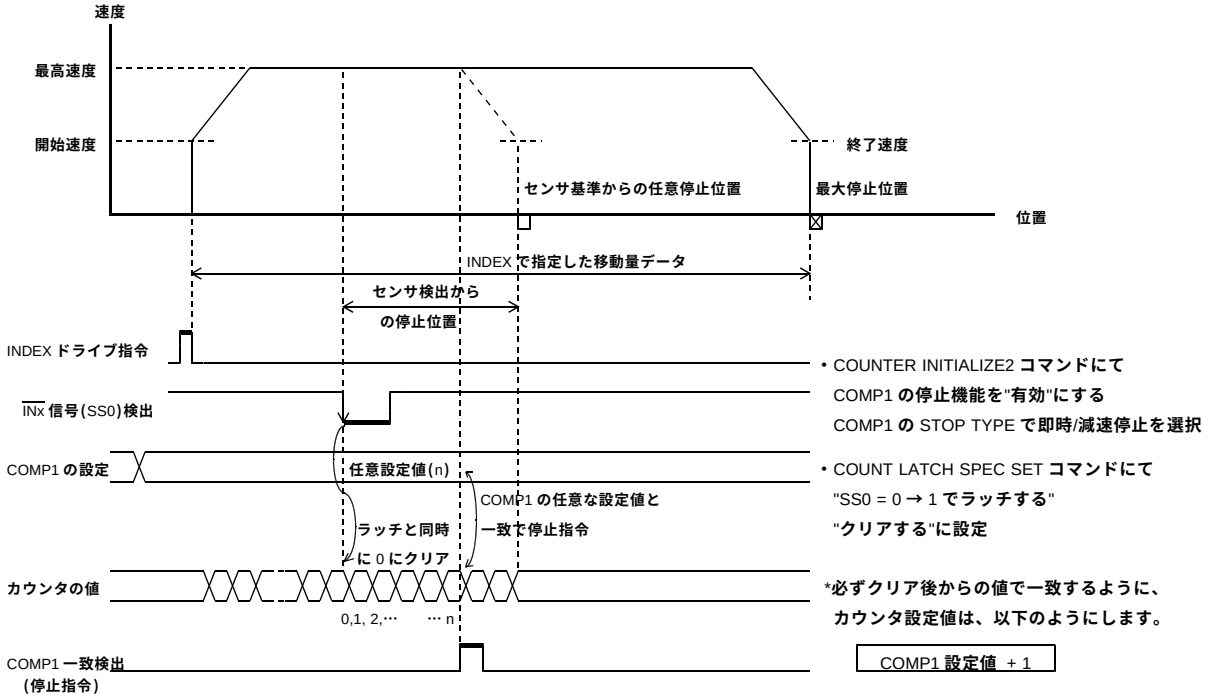
・SS0 信号は下記の軸に対応しています。(初期値)

/IN0 信号	X 軸の SS0 信号
/IN1 信号	Y 軸の SS0 信号

■カウンタのクリア機能

カウントデータのラッチと同時にカウントデータを "0" にクリアします。
カウンタのカウントタイミングとクリアタイミングが同時に発生した場合はクリアを優先します。
・各カウンタのラッチクリア機能は COUNT LATCH SPEC SET コマンドで設定します。

●カウンタラッチとクリア機能を応用したドライブ例
センサ(IN0 信号または IN1 信号)の検出を基準とした位置から、カウンタのコンパレーター一致による停止機能でドライブを停止させることが可能です。
・ドライブ起動前に必要な設定を済ませることができ、ドライブ中の Windows OS や AL-II 通信などの影響を受けない繰り返し精度の高いセンサ検出によるドライブが可能です。
・予め INDEX ドライブを起動することで、センサ信号検出しない場合でもメカの最大移動量は保証できます。
・センサ検出から減速開始または即時停止するまでの移動量をコンパレータの設定値により調整できます。減速停止時の減速開始から出力されるパルス数は保証されます。
・ドライブ中の他の停止信号や停止コマンドも有効です。

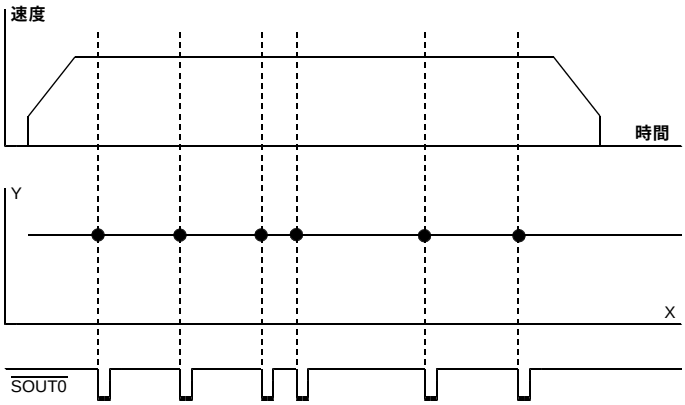


4-3. MCM 機能

4-3-1. 特徴

MCM(Motion Control Macro)とは、ユーザが独自に複数の MCC09 コマンドを連続実行するプログラムをシート単位のメモリーにマクロ化することができる機能です。
ユーザアプリケーションから、MCM を開始することにより、ユーザマクロプログラムはユニット上のハードウェア処理で自動実行させることができます。その結果、リアルタイムなモーション制御が可能になり、モーションコントロールの精度および信頼性の向上を図ることができます。
また、モーションコントロールがユニットに分散されるため、アプリケーション処理に要する負荷を軽減させることができます。

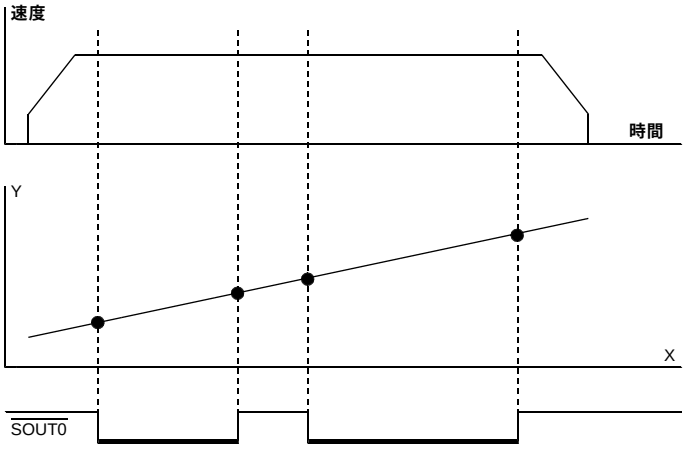
●応用例 1
INDEX ドライブ中に任意の位置で $\overline{\text{SOUT0}}$ 信号にトリガ出力します



X 軸のパルスカウンタのコンパレーター致と同時に、予め MCM に登録されている任意な検出値を順次コンパレータに自動的に書き換えます。
このコンパレーター致検出タイミングを $\overline{\text{SOUT0}}$ 信号に出力させます。

OS に依存するコンパレータの書き換え時間の不安定性さや、且つその処理の負荷も省くことができます。
パルスカウンタの実位置を基準としたタイミングでリアルタイムな外部機器との同期制御が可能です。

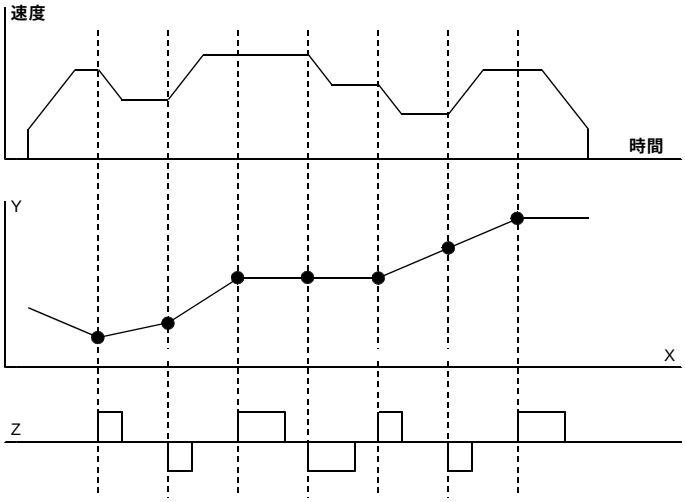
●応用例 2
直線補間ドライブ中に任意の区間で $\overline{\text{SOUT0}}$ 信号にアクティブレベルを出力します



X 軸のパルスカウンタのコンパレーター致と同時にコンパレータの書き換えを行います。
コンパレーター致直後に、予め MCM に登録されている SIGNAL OUT コマンドを実行して、$\overline{\text{SOUT0}}$ 信号の ON/OFF 出力を自動実行します。

カウンタの読み出しから I/O 信号出力までの制御時間の遅れや繰り返し誤差をなくすことができます。
パルスカウンタの実位置を基準としたタイミングでリアルタイムな外部機器同期が可能です。

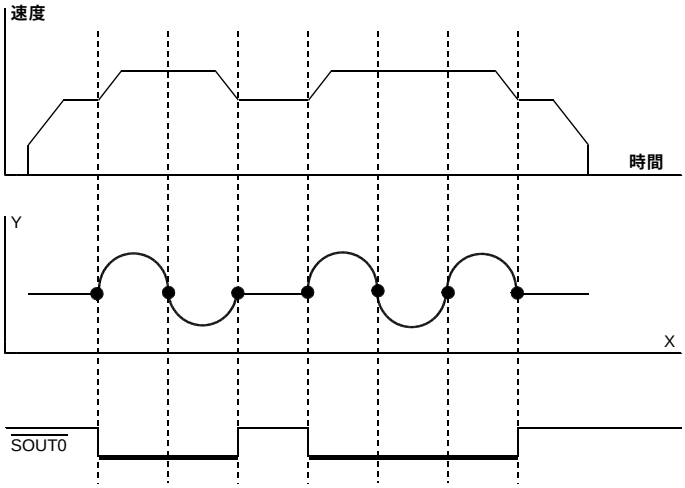
●応用例 3
直線補間ドライブ中、区間の始まりに同期して Z 軸の INDEX ドライブを行います



予め MCM に登録されている X-Y 軸の各ポジションへ直線補間ドライブを実行します。
各ポジションの通過時に、次の X-Y 軸の直線補間ドライブの実行タイミングと同期を合わせて、Z 軸の INDEX ドライブを実行します。

アプリケーションから X-Y 軸ポジションの通過点となる終了タイミングを監視(読み出す)する必要はなく、X-Y 座標の任意な位置で Z 軸のドライブ開始タイミングをリアルタイムに制御できます。ヘッド制御のアーチモーション等にも応用できます。

●応用例 4
円弧補間ドライブ中に任意の区間で $\overline{\text{SOUT0}}$ 信号にアクティブレベルを出力します



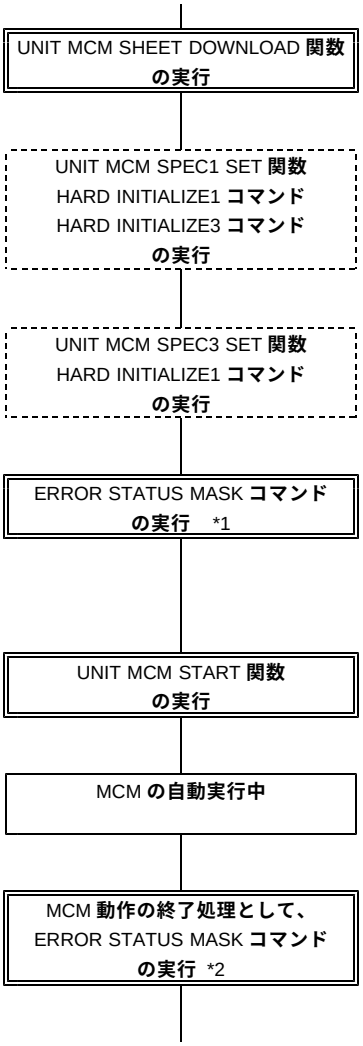
予め MCM に登録されている X 軸の INDEX ドライブまたは X-Y 軸の各ポジションへ円弧補間ドライブを実行します。
各ポジションの通過時に、次の X 軸の INDEX ドライブまたは X-Y 軸の円弧補間ドライブの実行タイミングと同期を合わせて、SIGNAL OUT コマンドを実行して、$\overline{\text{SOUT0}}$ 信号の ON/OFF 出力を自動実行します。

アプリケーションから X-Y 軸ポジションの通過点となる終了タイミングを監視(読み出す)する必要はなく、リアルタイムな外部機器同期が可能です。

4-3-2. MCM 関数シーケンス

MCM による制御シーケンスを示します。
このシーケンスには、関数エラーが発生した場合のフローは含まれていません。

(1) MCM の全体実行シーケンス



① RAM 領域に MCM シートをダウンロードします。

② MCM の同期動作を行うときの信号の割り当てと機能を設定します。
- UNIT MCM SPEC1 :SS1 の機能を選択します。
- HARD INITIALIZE1 :OUT1 の同期タイミングを取る信号を選択します。
- HARD INITIALIZE3 :GPIO3 機能を SS1 にします。

③ MCM の自動実行中に内部タイミングを外部出力するときの信号の割り当てと機能を設定します。
- UNIT MCM SPEC3 :SOUT 信号に出力する軸と信号を設定します。
- HARD INITIALIZE1 :OUT0 信号の出力機能を選択します。

④ MCM 動作を次の要因で停止できるように ERROR STATUS MASK を「マスクしない」に設定します。
- **即時停止要因** :FSSTOP ERROR MASK
- **減速停止要因** :SLSTOP ERROR MASK
- **リミット停止** :LSEND ERROR MASK

⑤ 開始アドレスを指定して、MCM を開始します。

⑥ 開始アドレスから、内部 RAM のデータ・コマンドを自動実行します。
- RAM STOP コマンドを MCM に組み込んでおくと自動終了します

⑦ MCM 終了処理として、MCM 起動前の ERROR STATUS MASK に設定し直します。
- **即時停止要因** :FSSTOP ERROR MASK
- **減速停止要因** :SLSTOP ERROR MASK
- **リミット停止** :LSEND ERROR MASK

初期値および設定値に対して変更が必要な場合に設定します。

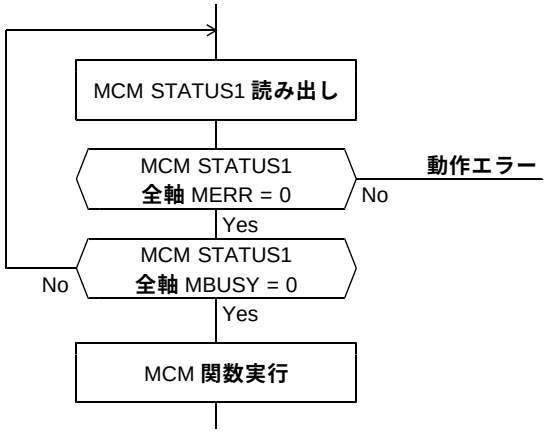
*1 ユーザアプリケーションから MCM 動作の開始前に、ERROR STATUS MASK を設定してください。
MCM 動作中の即時停止は、UNIT MCM FSSTOP 関数で実行します。
MCM 動作中の減速停止は、UNIT MCM SLSTOP 関数で実行します。
ERROR STATUS MASK 設定および上記の関数の実行により、RAM 領域の停止と予約コマンドに格納されている汎用コマンドがクリアされ、MCM 動作が停止します。

*2 MCM 動作終了後に、ERROR STATUS MASK 設定を MCM 起動前の元の設定に戻してください。

(2) 全軸の確認が必要な MCM 関数

MCM 関数で次の処理をする場合、MCM STATUS1 の全軸の MERR = 0 と MBUSY = 0 の確認が必要です。

- UNIT MCM SPEC0 SET 関数
- UNIT MCM SPEC1 SET 関数
- UNIT MCM SPEC2 SET 関数
- UNIT MCM SPEC3 SET 関数
- UNIT MCM SHEET DOWNLOAD 関数



MCM STATUS1 を読み出します。

MCM STATUS1 の全軸の MERR = 0 を確認します。

MCM STATUS1 の全軸の MBUSY = 0 を確認します。

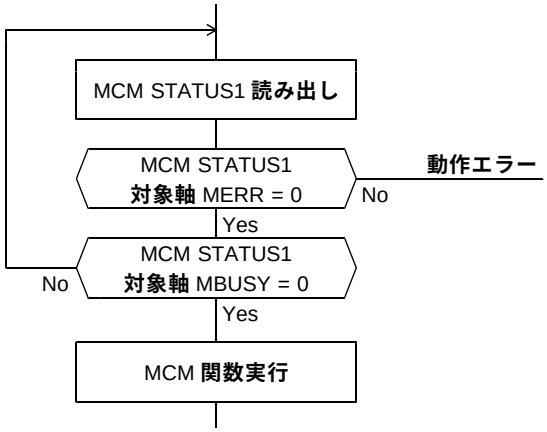
目的の MCM 関数を実行します。

・ユニットの一部の軸で MERR = 1 または MBUSY = 1 の場合、上記の MERR = 0 と MBUSY = 0 の確認が必要な関数を実行すると、関数はエラー終了します。

(3) 対象軸の確認が必要な MCM 関数

MCM 関数で次の処理をする場合、MCM STATUS1 の対象軸の MERR = 0 と MBUSY = 0 の確認が必要です。

- UNIT MCM START 関数の実行
- UNIT MCM WRITE 関数の実行
- UNIT MCM READ 関数の実行



MCM STATUS1 を読み出します。

MCM STATUS1 の対象軸の MERR = 0 を確認します。

MCM STATUS1 の対象軸の MBUSY = 0 を確認します。

目的の MCM 関数を実行します。

- 対象軸の一部または全部で MERR = 1 または MBUSY = 1 の場合、上記の MERR = 0 と MBUSY = 0 の確認が必要な関数を実行すると、関数はエラー終了します。

次の MCM 関数を実行する場合、MCM STATUS1 の対象軸の MERR = 0 と MBUSY = 0 の確認は不要です。

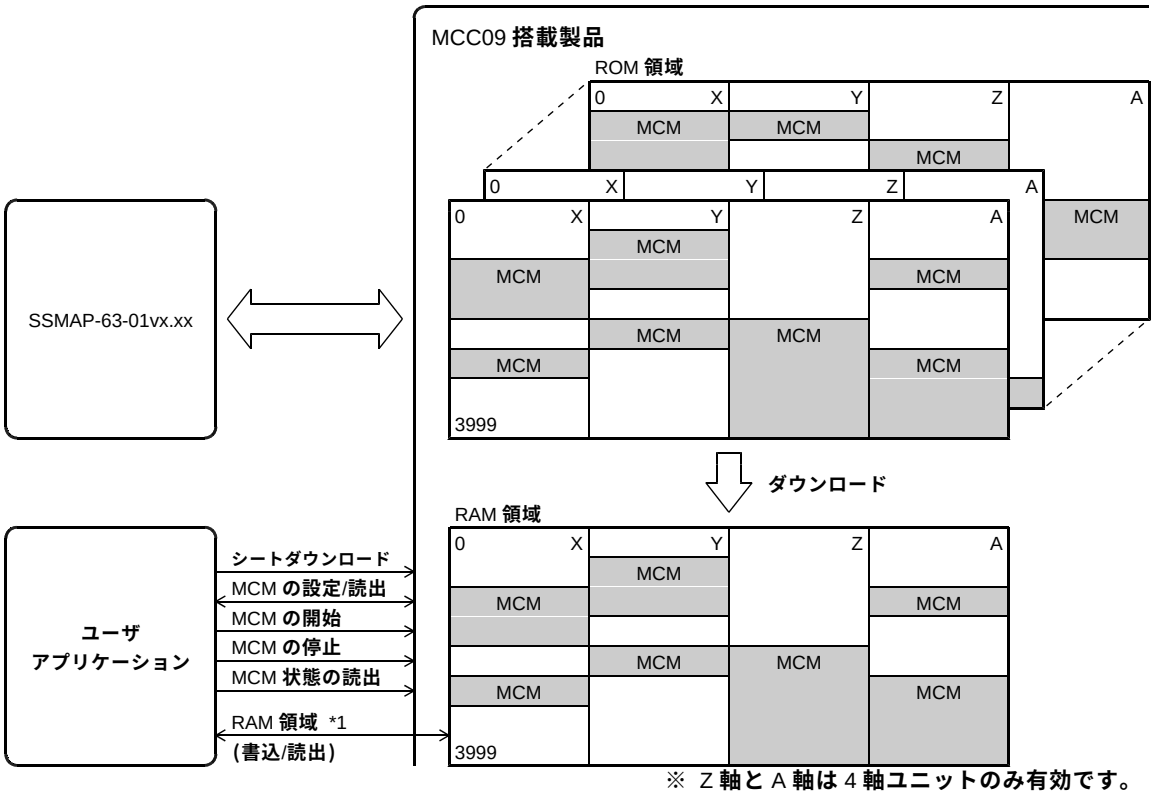
- UNIT MCM FSSTOP 関数
- UNIT MCM SLSTOP 関数
- UNIT MCM ERROR CLR 関数

目的の MCM 関数を実行します。

4-3-3. 構成

MCC09 搭載製品には、MCM を保存するための ROM 領域と、MCM を実行するための RAM 領域があります。
ROM 領域には 4 枚のシートが存在し、各シートには MCC09 のコマンドを 4000 個× 4 軸分だけ保存することができます。RAM 領域には 1 枚のシートと同じ容量が確保されています。

ROM 領域に格納されているコマンドは、ユニットの電源を切っても失われることはありません。
RAM 領域に格納されているコマンドは、ユニットの電源を切ると、全て失われます。



※ Z 軸と A 軸は 4 軸ユニットのみ有効です。

ユニットの出荷時、ROM 領域の全エリアは H'FFFF_FFFF_FF に初期化されています。
ユニットの電源投入時、RAM 領域の全エリアは H'FFFF_FFFF_FF に初期化されます。

*1 ユニットの電源投入時、ユーザアプリケーションからの RAM 領域アクセスは無効になっています。
MCM SPEC0 SET 関数により、ユーザアプリケーションからの RAM 領域アクセスを、無効から有効に切り替えることができます。

ユーザアプリケーションから実行できる MCM 関数は、以下の通りです。

項目	関数名	機能	初期値
MCM の設定	UNIT MCM SPEC1 SET ／ GET 関数	MCM 同期制御を行うときの設定と読み出し	−
	UNIT MCM SPEC2 SET ／ GET 関数	外部入力信号の割り付け設定と読み出し	−
	UNIT MCM SPEC3 SET ／ GET 関数	外部出力信号の割り付け設定と読み出し	−
MCM シートの設定	UNIT MCM SHEET DOWNLOAD 関数	MCM シートのダウンロード	−
MCM の開始	UNIT MCM START 関数	MCM の起動	−
MCM の停止	UNIT MCM FSSTOP 関数	MCM の即時停止	−
	UNIT MCM SLSTOP 関数	MCM の減速停止	−
MCM 状態読み出し	UNIT MCM STATUS READ 関数	MCM ステータスの読み出し	−
MCM 情報読み出し	UNIT MCM INFO READ 関数	MCM の実行シートと実行アドレスの読み出し	−
MCM エラー読み出し	UNIT MCM ERROR STATUS READ 関数	MCM のエラーの読み出し	−
MCM エラー解除	UNIT MCM ERROR CLR 関数	MCM エラーのクリア	−
RAM アクセス設定	UNIT MCM SPEC0 SET ／ GET 関数	RAM 領域アクセスの有効／無効の切替	無効
RAM 領域書き込み	UNIT MCM RAM WRITE 関数	RAM 領域へ 1 アドレス分の書き込み	無効
RAM 領域読み出し	UNIT MCM RAM READ 関数	RAM 領域から 1 アドレス分の読み出し	無効

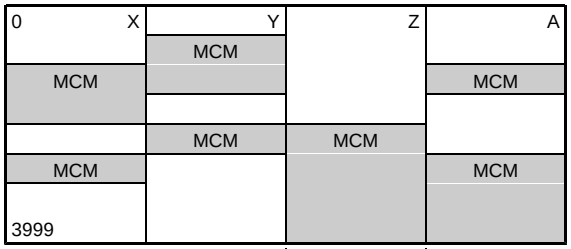
1 枚のシートには複数の MCM を保存することができ、専用エディッタツールソフト SSMAP-63-01 を用いることで、シート上の任意の位置にプログラミングした MCM を配置することができます。

* SSMAP-63-01 は、USB 通信を介して、MCC09 搭載スレーブユニット上の MCM シート編集、書き込み／読み出しを行うためのエディッタツール兼簡易動作アプリケーションソフトです。
SSMAP-63-01 の取り扱い方法については、HELP ファイルにてご確認ください。

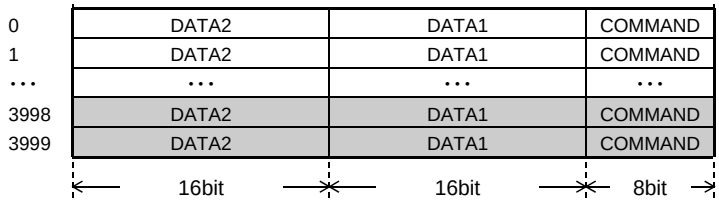
シート上に MCM を配置後、UNIT MCM SHEET DOWNLOAD 関数を実行すると、ROM 領域の中の 1 枚のシートのイメージが RAM 領域に転送され、MCM が実行可能になります。
また、ユーザアプリケーションからは、UNIT MCM SPEC0 SET 関数により、直接 RAM 領域にアクセスすることも可能です。

ROM 領域および RAM 領域上の MCM は MCC09 のコマンドから構成され、各コマンドは
16 ビットの DATA2、16 ビットの DATA1、8 ビットの COMMAND で構成されます。
・DATA1 ・・・ MCC09 の DRIVE DATA1 PORT に書き込むデータ
・DATA2 ・・・ MCC09 の DRIVE DATA2 PORT に書き込むデータ
・COMMAND ・・・ MCC09 の DRIVE COMMAND PORT に書き込むデータ

ROM 領域（シート）／ RAM 領域

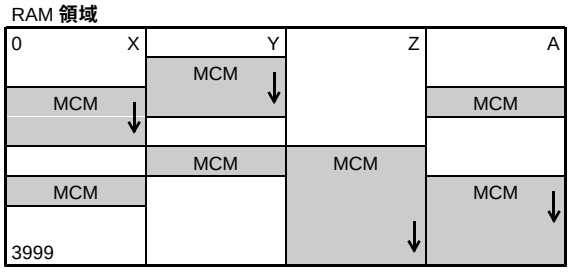


※ Z 軸と A 軸は 4 軸ユニットのみ有効です。



MCM は、RAM 領域にメモリされたデータ・コマンドを自動実行します。
・1 つの軸につき、複数の MCM を同時に実行させることはできません
・異なる軸であれば、複数の MCM を同時に実行させることができます

RAM 領域



↓ 実行中のコマンド

※ Z 軸と A 軸は 4 軸ユニットのみ有効です。

メモリされたコマンド・データは、実行タイミングの検出で順次実行します。
・汎用コマンドおよび特殊コマンドを自動実行して、任意形状ドライブのシーケンス制御ができます。
・SPEED CHANGE コマンドを自動実行して、任意形状ドライブができます。
・パルスカウンタの COMP1 設定コマンドを自動実行して、任意時間連続検出ができます。
・パルス偏差カウンタの COMP1 設定コマンドを自動実行して、任意時間連続検出ができます。

4-3-4. MCM の設定

MCM SPEC を設定すると、MCM で次のような制御が可能になります。

● **MCM SPEC0 SET 関数**

直接ユーザアプリケーションから MCC09 **の** RAM **領域に書き込み／読み出しができるように切り替えることができます。**

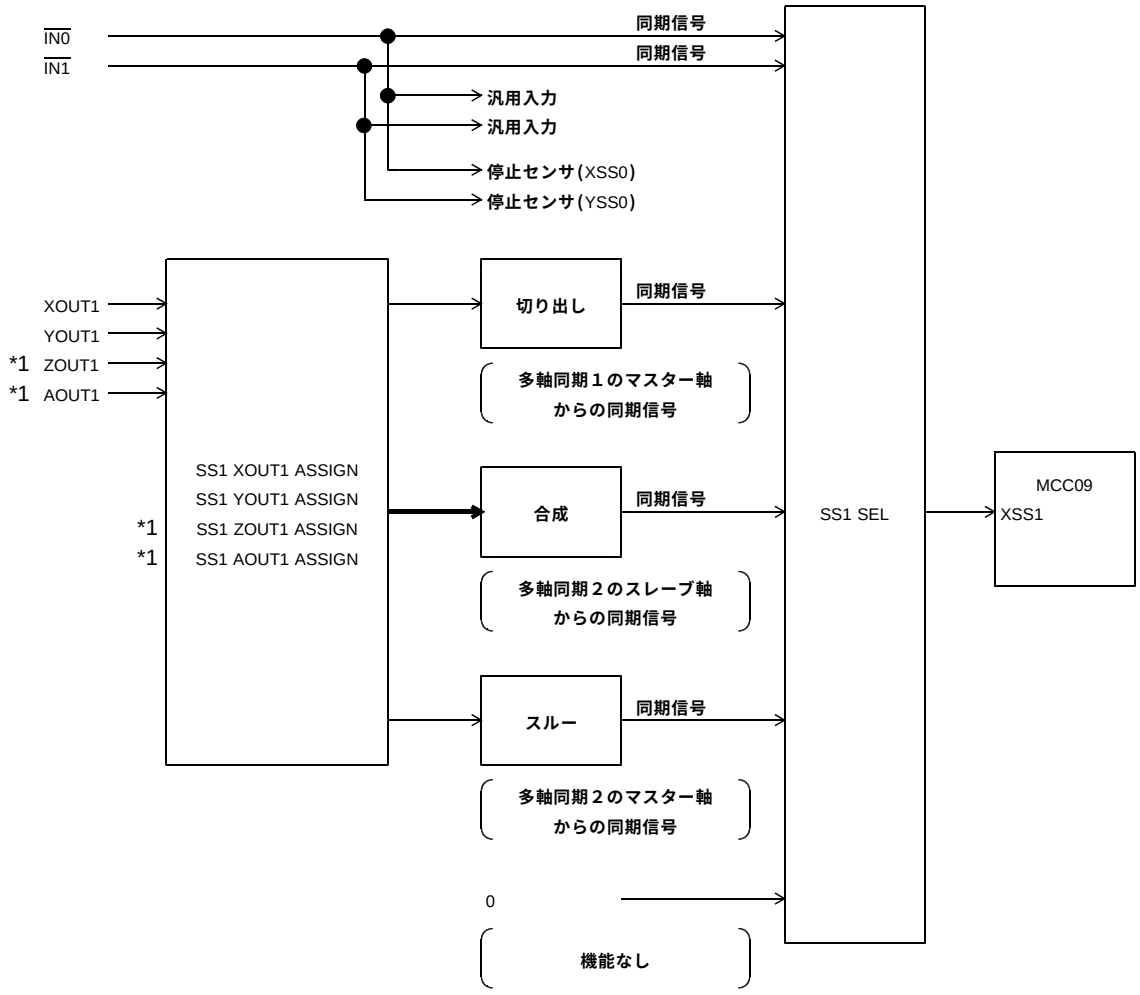
・UNIT MCM WRITE **関数により、指定された** RAM **領域のアドレスにコマンドとデータを書き込みます。**
・UNIT MCM READ **関数により、指定された** RAM **領域のアドレスからコマンドとデータを読み出します。**

● **MCM SPEC1 SET 関数**

MCM **での同期制御を行うときの設定です。**
MCM **の同期を行うための信号の割り当てと機能を設定することができます。**
・SS1 SEL **により、外部トリガによる同期を行うための設定をします。**
・SS1 SEL、SS1 OUT1 ASSIGN **により、内部トリガによる多軸同期を行うための設定をします。**

電源投入時の初期値は、「SS1 **に割り当てない」、「**SS1 **機能なし」です。**

【 多軸同期または外部トリガによる同期を行うための信号と軸の機能（X **軸の場合）】**



*1 ZOUT1,AOUT1,SS1 ZOUT1 ASSIGN,SS1 AOUT1 ASSIGN **は、**4 **軸ユニットのみ指定及び選択が出来ます。**

・$\overline{\text{IN0}}$ **信号を同期信号として使用する場合、停止センサ(**SS0**)として使用しないでください。**
・$\overline{\text{IN1}}$ **信号を同期信号として使用する場合、停止センサ(**SS0**)として使用しないでください。**

UNIT MCM SPEC1 SET **関数により、**MCM SPEC1 **を設定します**
HARD INITIALIZE3 **コマンドで** GPIO3 **信号を** SS1 **に設定します。**

MCM SPEC1 **で** MCC09 **の** SS1 **に信号を割り付けると、**GPIO3 **の** SS1 **フラグを用いた、**
MCM **同期の制御が可能になります。**

● **MCM SPEC2 SET**

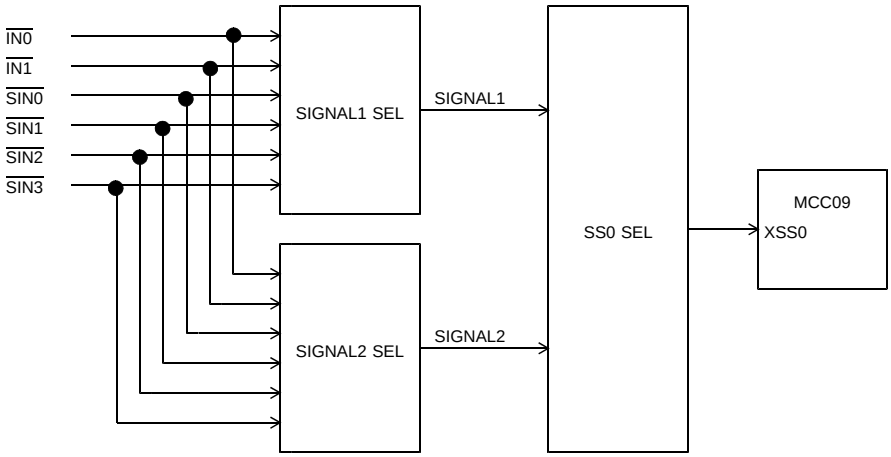
外部信号から制御を行うときの設定です。
外部信号から MCC09 **の** SS0 **に入力する信号を設定することができます。**
MCC09 **の** SS0 **は、停止機能または外部トリガとして使用ことができます。**

SIGNAL1 **と** SIGNAL2 **には、それぞれ次の信号を割り付けることができます。**
・$\overline{\text{IN0}}$ **信号** : +24V I/F **フォトカプラ入力**
・$\overline{\text{IN1}}$ **信号** : +24V I/F **フォトカプラ入力**
・$\overline{\text{SIN0}}$ **信号** :TTL **レベル入力**
・$\overline{\text{SIN1}}$ **信号** :TTL **レベル入力**
・$\overline{\text{SIN2}}$ **信号** :TTL **レベル入力**
・$\overline{\text{SIN3}}$ **信号** :TTL **レベル入力**

SS0 SEL **により、上記** SIGNAL1 **または** SIGNAL2 **に割り付けされた外部入力信号を**
SS0 **信号に入力する条件を加えて接続することができます。**
・SIGNAL1 **側の信号をスルー出力で** SS0 **に接続する**
・SIGNAL1 **側の信号と** SIGNAL2 **側の信号を論理積(**AND**)を取って** SS0 **に接続する**
・SIGNAL1 **側の信号と** SIGNAL2 **側の信号を論理和(**OR**)を取って** SS0 **に接続する**

外部信号と SS0 **信号の初期値は次のようになっています。**
・X **軸** SS0 : $\overline{\text{IN0}}$ **信号**
・Y **軸** SS0 : $\overline{\text{IN1}}$ **信号**
・Z **軸** SS0 : **外部信号割り当てなし**
・A **軸** SS0 : **外部信号割り当てなし**

【 停止センサ機能（X **軸の場合）】**



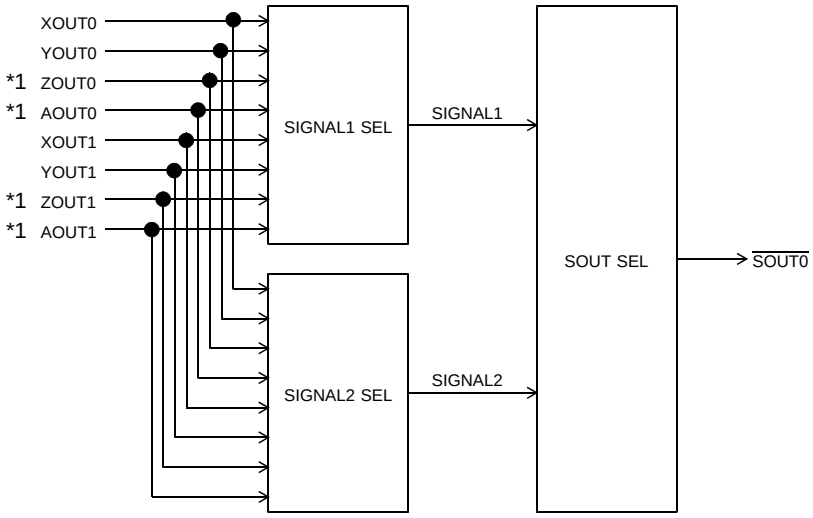
UNIT MCM SPEC2 SET **関数により、**MCM SPEC2 **を設定します**
・SS0 **を停止機能として使用する場合、**SPEC INITIALIZE2 **コマンドで** SS0 **信号を停止機能に設定します。**

・SS0 **をトリガ信号として使用する場合、**SPEC INITIALIZE2 **コマンドで** SS0 **信号を汎用入力に設定します。**
MCM SPEC2 **で** MCC09 **の** SS0 **に信号を割り付けると、外部信号を** GPIO2 **の** SS0 **フラグにより、**
次のタイミングを取ることができます。
PAUSE **機能の** STBY **解除条件**
SPEED RATE CHANGE **の変更動作点**
INDEX CHANGE **の変更動作点**
各カウンタのラッチタイミング
パルス偏差カウンタのカウント信号
RAM **機能の** JUMP **条件**
RAM **機能のメモリーデータの** NO OPERATION **コマンド実行条件**

● **MCM SPEC3 SET**

MCC09 **の内部ステータス信号** OUT0 **信号**, OUT1 **信号を** SOUT **信号に出力する信号として設定できます。**
UNIT MCM SPEC3 SET **関数により、**MCM SPEC3 **を設定します**

【 ステータス外部出力機能（$\overline{\text{SOUT0}}$ **信号の場合）】**



*1 ZOUT0,AOUT0,ZOUT1,AOUT1 **は、**4 **軸ユニットのみ選択が出来ます。**

OUT0, OUT1 **信号の選択は、**HARD INITIALIZE1 **コマンドで設定します。**

OUT0,1 **信号の出力機能**

TYPE3	TYPE2	TYPE1	TYPE0	OUT0,1 **の出力機能**	
0	0	0	0	ADRINT	**カウンタ割り込み要求の** ADRINT **出力**
0	0	0	1	CNTINT	**カウンタ割り込み要求の** CNTINT **出力**
0	0	1	0	DFLINT	**カウンタ割り込み要求の** DFLINT **出力**
0	0	1	1	RDYINT	**コマンド終了割り込み要求の** RDYINT **出力**
0	1	0	0	STBY	STATUS1 **の** STBY **フラグ出力**
0	1	0	1	DRIVE	STATUS1 **の** DRIVE **フラグ出力**
0	1	1	0	nSPEED FL	STATUS5 **の** SPEED FL **フラグの反転出力**
0	1	1	1	nINDEX FL	STATUS5 **の** INDEX FL **フラグの反転出力**
1	0	0	0	UP	STATUS1 **の** UP **フラグ出力**
1	0	0	1	DOWN	STATUS1 **の** DOWN **フラグ出力**
1	0	1	0	CONST	STATUS1 **の** CONST **フラグ出力**
1	0	1	1	EXT PULSE	STATUS1 **の** EXT PULSE **フラグ出力**
1	1	0	0	nPULSE MASK	STATUS2 **の** PULSE MASK **フラグの反転出力**
1	1	0	1	ORG SIGNAL	STATUS2 **の** ORG SIGNAL **フラグ出力**
1	1	1	0	**汎用出力**	**汎用出力として使用する**
1	1	1	1	**設定禁止**	**－**

OUT0 **初期値** ＝ CNTINT
OUT1 **初期値** ＝ RDYINT

SOUT 信号には、割り当てる軸、ステータス信号の種、および任意な２つの信号の論理和・論理積を加えて出力させることができます。
SOUT **信号の初期値は次のようになっています。**

外部出力信号	**軸と出力(割付信号名)**		SIGNAL1 SEL
	4 **軸ユニット**	2 **軸ユニット**	
$\overline{\text{SOUT0}}$	XOUT0(XCNTINT)	XOUT0(XCNTINT)	**信号をスルー出力**
$\overline{\text{SOUT1}}$	YOUT0(YCNTINT)	YOUT0(YCNTINT)	**信号をスルー出力**
$\overline{\text{SOUT2}}$	ZOUT0(ZCNTINT)	XOUT0(XCNTINT)	**信号をスルー出力**
$\overline{\text{SOUT3}}$	AOUT0(ACNTINT)	YOUT0(YCNTINT)	**信号をスルー出力**

MCM SPEC3 **で** MCC09 **内部ステータス信号** SOUT **信号を割り付けると、**OUT0, OUT1 **の内部ステータスのタイミングで外部機器に信号出力することができます。**

4-3-5. MCM の自動実行

MCM を開始すると、MCM を構成する MCC09 のデータ・コマンドは自動実行します。
・MCC09 のコマンドは、実行タイミングの検出で順次実行します。

(1) MCM の開始

以下の関数で、MCM の実行を開始します。
・UNIT MCM START 関数 ：開始アドレスから、RAM 領域のコマンド自動実行の開始

MCM STATUS1 の MBUSY = 0 のときに、MCM START 関数を実行すると、指定された開始アドレスから、データ・コマンドを読み出して、RAM 領域のコマンドを順次実行します。
・UNIT MCM START 関数を実行すると、MCM STATUS1 の MBUSY = 1 になります。
・同時に MCM STATUS1 の MRUN = 0 → 1、MCM STATUS2 の MRBUSY = 0 → 1 になります。

・40 ビットのデータ・コマンドを読み出して、１命令のコマンドとして実行します。
RAM 領域のコマンドは、実行タイミングを検出すると実行します。
実行タイミングを待っている間は、RAM 領域のコマンド読み出しを保留します。
実行タイミングを待っている間は、MCM STATUS2 の MRSTBY = 1 になります。

・RAM 領域の開始アドレスから読み出したコマンドを実行すると、RAM 領域の開始アドレス＋１のアドレスのデータ・コマンドを読み出します。
・RAM 領域から読み出したコマンドを実行すると、前回の読み出しアドレス＋１のアドレスのデータ・コマンドを読み出します。以降は、これを繰り返します。

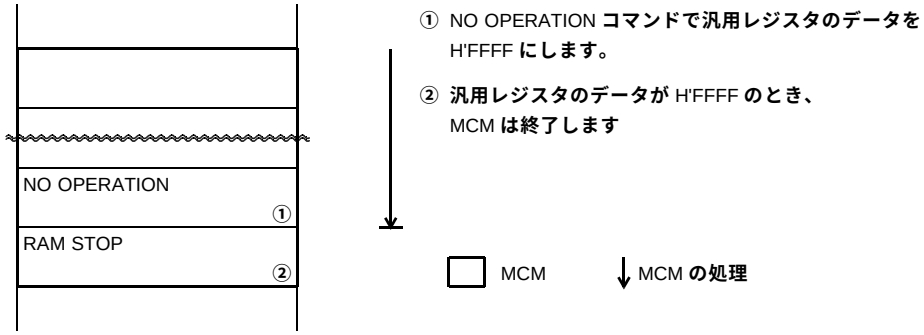
<RAM 領域のコマンドの実行タイミング>
・汎用コマンド（H'01 ～ H'3F）は、COMREG FL = 0 のときに実行します。
BUSY = 1 のときは、コマンド予約機能の予約レジスタに格納します。
・SPEED CHANGE コマンド（H'C1, H'C8）は、SPEED FL = 0 のときに実行します。
・INDEX CHANGE コマンド（H'C3, H'CC, H'CD, H'CE）は、INDEX FL = 0 のときに実行します。
・ADRINT COMPARE REGISTER1 SET コマンド（H'88）は、STATUS4 PORT の ADRINT COMP1 = 0 → 1 のアクティブエッジの検出で実行します。
・CNTINT COMPARE REGISTER1 SET コマンド（H'98）は、STATUS4 PORT の CNTINT COMP1 = 0 → 1 のアクティブエッジの検出で実行します。
・DFLINT COMPARE REGISTER1 SET コマンド（H'A8）は、STATUS4 PORT の DFLINT COMP1 = 0 → 1 のアクティブエッジの検出で実行します。
・RAM READ JUMP コマンド（H'BC）の実行タイミングは、RAM SPEC SET コマンドの JUMP TYPE で選択します。
・その他のコマンドは、即実行します。

(2) MCM の終了

以下のコマンドで MCM はその実行を終了します。
・RAM STOP コマンド ：RAM 領域のコマンド自動実行の終了

MCM に埋め込まれた RAM STOP コマンドを実行すると、MCM はコマンド自動実行を終了します。
・コマンド自動実行を終了すると、MCM STATUS2 の MRBUSY = 1 → 0 になります。
・MCM が自動実行を終了した後、実行中または予約中の汎用コマンドが終了すると、MCM は終了します。
・MCM が終了すると、MCM STATUS1 の MBUSY = 1 → 0、MRUN = 1 → 0 になります。

RAM STOP コマンドで MCM を終了させるときは、RAM STOP コマンドの一つ手前のアドレスに
NO OPERATION コマンドを入れます。
・NO OPERATION コマンドで設定する汎用レジスタのデータは H'FFFF にしてください。



RAM 領域の読み出しアドレスが 3,999 になると、読み出したコマンド実行後に、コマンド読み出しを終了します。
・アドレス 3,999 が RAM STOP コマンド以外の場合、MCM はエラー終了します

※ RAM STOP コマンドは、MCM の中で自動的に実行するコマンドです。
ユーザアプリケーションから実行することはできません。

MCM STATUS1 の MBUSY = 1 のときに、UNIT MCM FSSTOP 関数または UNIT MCM SLSTOP 関数、
即時停止機能または減速停止機能を実行すると、MCM はコマンド自動実行を終了します。
・コマンド自動実行を終了すると、MCM STATUS2 の MRBUSY = 1 → 0 になります。
実行タイミングを待っているコマンドは無効にします。
・MCM が自動実行を終了した後、実行中の汎用コマンドが終了すると、MCM は終了します。
・MCM が終了すると、MCM STATUS1 の MBUSY = 1 → 0、MRUN = 1 → 0 になり、同時に
MCM エラーが発生します。

MCM STATUS1 の MBUSY = 1 のときに、STATUS1 PORT の ERROR = 1 を検出すると、MCM は
コマンド自動実行を終了します。
・コマンド自動実行を終了すると、MCM STATUS2 の MRBUSY = 1 → 0 になります。
実行タイミングを待っているコマンドは無効にします。
・MCM が自動実行を終了した後、実行中の汎用コマンドが終了すると、MCM は終了します。
・MCM が終了すると、MCM STATUS1 の MBUSY = 1 → 0、MRUN = 1 → 0 になり、同時に
MCM エラーが発生します。

(3) MCM の同期

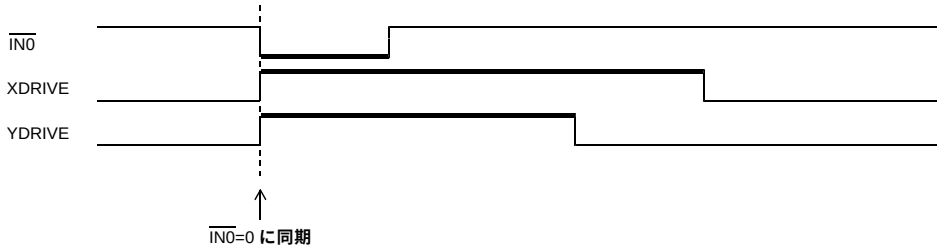
MCM の中に同期ポイントを設けることで、MCM の実行を目的のタイミングに同期させることができます。
・同期ポイントとは、MCM を作成する際に MCM に埋め込んでおく必要がある一連のコマンド群です。
・MCM の実行中、同期ポイントに達するまでは通常に動作し、同期ポイントに達したときに、
同期信号の出力や同期信号の待機を行います。
・同期ポイントで待機中の MCM は、目的のタイミングを検出すると、MCM の実行を再開します。

MCM の同期には、外部トリガによる同期と、内部トリガによる同期があります。
1 つの MCM の中で、外部トリガによる同期と、内部トリガによる多軸同期の両方を使用することはできません。

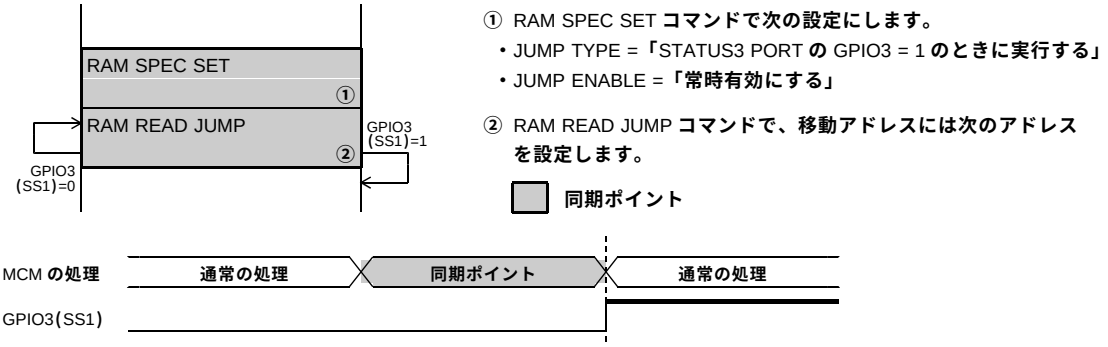
■ 外部トリガによる同期

$\overline{IN0}$ または $\overline{IN1}$ を外部トリガとして、MCM を同期させることができます。
外部トリガによる同期を利用すると、次の例の様な動作が可能です。

例. $\overline{IN0}$ のアクティブと同時にドライブを開始する



外部トリガによる同期を行う場合、$\overline{IN0}$ または $\overline{IN1}$ がアクティブになるまで、MCM の実行を中断します。
・$\overline{IN0}$ または $\overline{IN1}$ がアクティブになると、GPIO3(SS1) = 1 になります。
・GPIO3(SS1) = 0 の間待機し続ける MCM は、GPIO3(SS1) = 1 の検出で実行を再開します。



● 同期信号
GPIO3 に出力する同期信号として、$\overline{IN0}$ または $\overline{IN1}$ の選択ができます。

● MCM 開始前の設定
外部トリガによる同期を用いた MCM の開始前には、ユーザアプリケーションで次の設定をし、
MCM の終了後に元に戻します。
・UNIT MCM SPEC1 SET 関数で次の設定をする
SS1 SEL を「ユニット外部からの同期信号」にする
・HARD INITIALIZE3 コマンドの GPIO3 TYPE を「SS1」に設定する

※ 外部トリガによる同期を用いた MCM の中では、次の機能に SS1 または GPIO3 を使用しないでください。
・ORIGIN SPEC SET コマンドの ORG SIGNAL TYPE (SS1)
・RAM SPEC SET コマンドの NOP TYPE (GPIO3)

■ 内部トリガによる同期

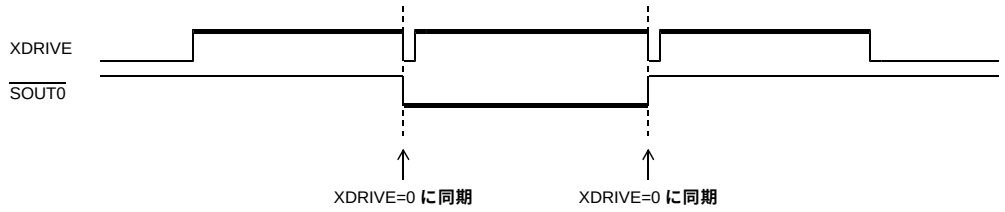
MCC09 の内部信号をトリガとして、MCM を同期させることができます。
内部トリガによる同期には、次の３つのタイプがあります。
・１軸同期
・多軸同期１
・多軸同期２

1 つの MCM の中で、１軸同期と多軸同期１の両方または１軸同期と多軸同期２の両方を使用することができます。
1 つの MCM の中で、多軸同期１と多軸同期２の両方を使用することはできません。

● １軸同期

自軸が特定の状態になるまで、自軸の MCM の実行を中断します。
１軸同期を利用すると、次の例の様な動作が可能です。

例. ドライブを連続で 3 回行い、2 回目の区間のみ $\overline{\text{SOUT0}}$ をアクティブレベルにする



MCM の各コマンドは、実行タイミングの検出により実行されます。（MCM のコマンド自動実行機能）
このような実行タイミングの検出を待つ動作も、自軸が特定の状態になるまで自軸の MCM の実行を
中断しますが、ここでは実行タイミングの検出を待つ動作自体を１軸同期とは定義していません。

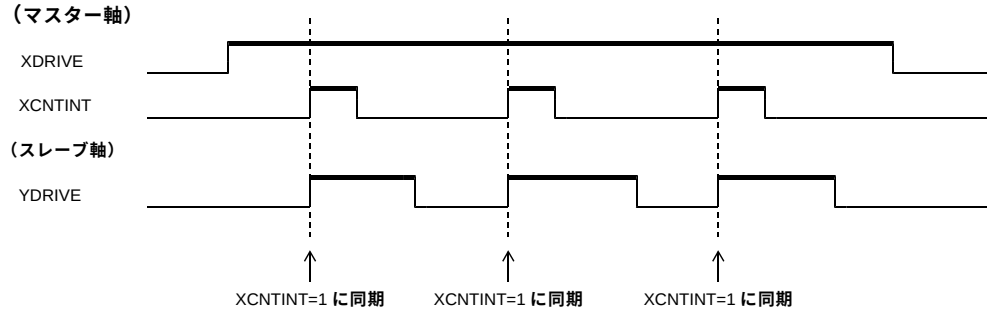
● 多軸同期

ユニット内の複数軸を使用した多軸同期ができます。
多軸同期を行う場合、多軸同期を行うための信号の割り当てと機能を MCM SPEC1 SET 関数で定義します。

【 多軸同期１】

マスター軸が特定の状態になるまで、スレーブ軸の MCM の実行を中断します。
・同期させる軸がマスター軸、同期する軸がスレーブ軸です。
・多軸同期１は、ユニット内の 1 つのマスター軸と、1 ～ 4 軸のスレーブ軸で構成します。
　このとき、マスター軸がスレーブ軸を兼ねることも可能です。

多軸同期１を利用すると、次の例の様な動作が可能です。
例. X 軸のパルスカウンタが所定のカウント値に達する度に Y 軸でドライブを起動する

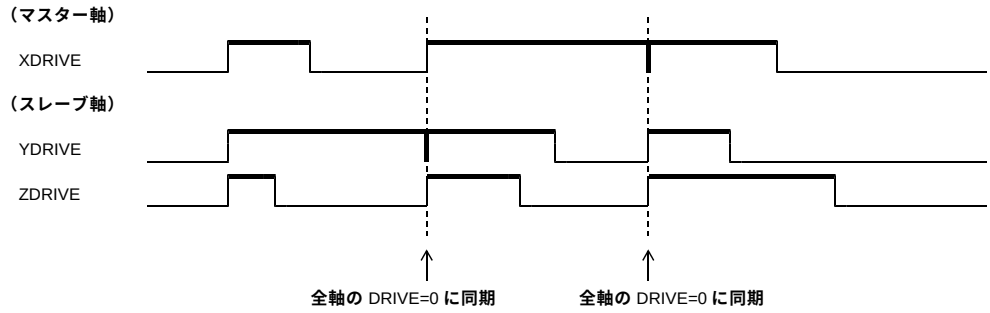


【 多軸同期２】

マスター軸および全てのスレーブ軸が特定の状態になるまで、マスター軸・スレーブ軸の MCM を中断します。
・マスター軸とスレーブ軸が相互に同期させる軸と同期する軸になります。
・多軸同期２は、ユニット内の 1 つのマスター軸と、それ以外の 1 ～ 3 軸のスレーブ軸で構成します。

多軸同期２を利用すると、次の例の様な動作が可能です。

例. 全軸がドライブを終了してから、一斉に次のドライブを開始する

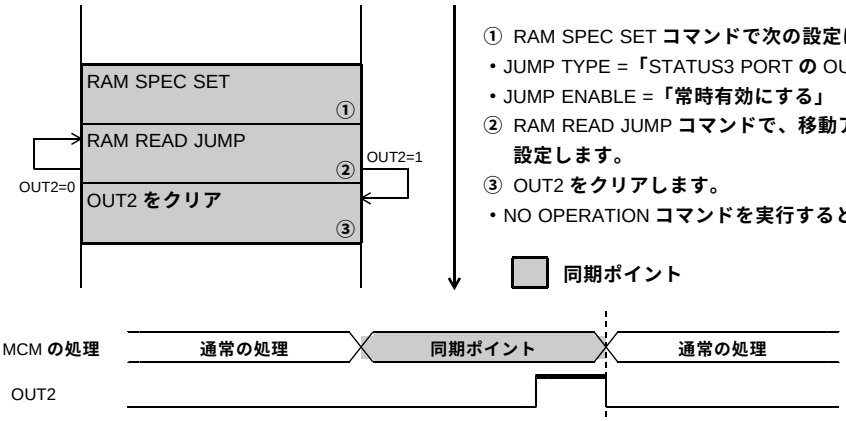


【１軸同期の設定例】

自軸が特定の状態になるまで、自軸の MCM **の実行を中断します。**

・同期信号を OUT2 **に割り当てます。**

・OUT2 = 0 **の間待機し続ける** MCM **は、**OUT2 = 1 **の検出で実行を再開し、**OUT2 = 0 **にクリアします。**



● **同期信号**

OUT2 **に出力する同期信号として、次の信号を選択します。**

・RDYINT **：コマンド終了割り込み要求の** RDYINT **出力**

次の同期信号で１軸同期を行う必要がある場合、その 1 **軸をマスター軸兼スレーブ軸として、多軸同期１を使用して下さい。**

・DRIVE **：**STATUS1 **の** DRIVE **フラグ出力**
・UP **：**STATUS1 **の** UP **フラグ出力**
・DOWN **：**STATUS1 **の** DOWN **フラグ出力**
・CONST **：**STATUS1 **の** CONST **フラグ出力**

● MCM **開始前と終了後の設定**

１軸同期を用いた MCM **の開始前には、ユーザアプリケーションで次の設定をし、**MCM **の終了後に元に戻します。**

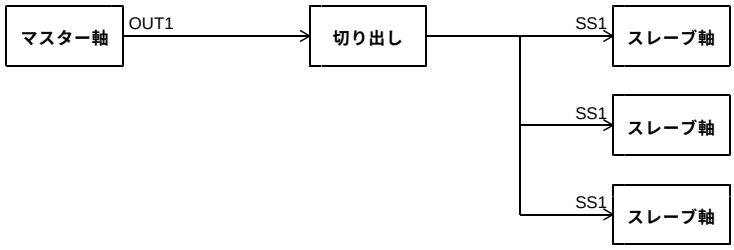
・HARD INITIALIZE1 **コマンドの** OUT2 TYPE **で** OUT2 **の出力機能を設定します。**

※ **１軸同期を用いた** MCM **の中では、次の機能に** OUT2 **を使用しないでください。**

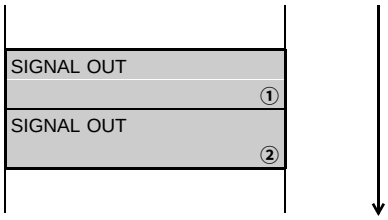
・SPEC INITIALIZE3 **コマンドの** STBY TYPE
・SPEED CHANGE SPEC SET **コマンドの** SPEED CHANGE TYPE
・INDEX CHANGE SPEC SET **コマンドの** INDEX CHANGE TYPE
・DFL COUNTER INITIALIZE1 **コマンドの** COUNT START TYPE
・COUNT LATCH SPEC SET **コマンドの各カウンタの** LATCH TYPE
・RAM SPEC SET **コマンドの** NOP TYPE

【多軸同期１の設定例】
マスター軸が特定の状態になるまで、スレーブ軸の MCM **の実行を中断します。**

・マスター軸の OUT1 **がアクティブになると、全てのスレーブ軸の** GPIO3(SS1) = 0 → 1 → 0 **になります。**
・GPIO3(SS1) = 0 **の間、待機し続ける全てのスレーブ軸の** MCM **は、**GPIO3(SS1) = 0 → 1 → 0 **の検出で実行を再開します。**



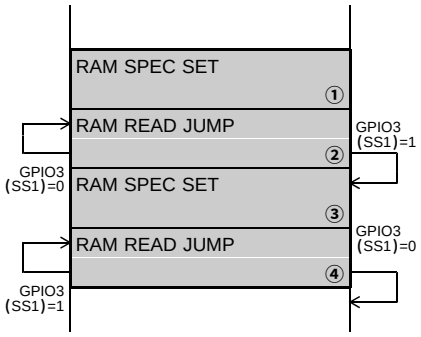
マスター軸
同期信号を汎用出力にすると、MCM **の処理と同期して同期信号がアクティブになります。**
（同期信号を汎用出力以外にすると、MCM **の処理とは非同期に同期信号がアクティブになります。）**



① SIGNAL OUT **コマンドで** OUT1 = 1 **にします**
② SIGNAL OUT **コマンドで** OUT1 = 0 **にします**

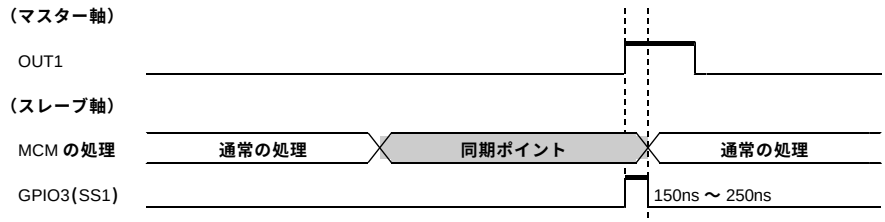
同期ポイント

スレーブ軸



① RAM SPEC SET **コマンドで次の設定にします。**
・JUMP TYPE =**「**STATUS3 PORT **の** GPIO3 = 1 **のときに実行する」**
・JUMP ENABLE =**「常時有効にする」**
② RAM READ JUMP **コマンドで、移動アドレスには次のアドレスを設定します**
③ RAM SPEC SET **コマンドで次の設定にします。**
・JUMP TYPE =**「**STATUS3 PORT **の** GPIO3 = 0 **のときに実行する」**
・JUMP ENABLE =**「常時有効にする」**
④ RAM READ JUMP **コマンドで、移動アドレスには次のアドレスを設定します**

同期ポイント



● 同期信号

マスター軸は、OUT1 に出力する同期信号として、次のいずれかの選択ができます。

・CNTINT ：カウンタ割り込み要求の CNTINT 出力
・RDYINT ：コマンド終了割り込み要求の RDYINT 出力
・DRIVE ：STATUS1 の DRIVE フラグ出力
・UP ：STATUS1 の UP フラグ出力
・DOWN ：STATUS1 の DOWN フラグ出力
・CONST ：STATUS1 の CONST フラグ出力
・汎用出力：汎用出力として使用する

● MCM 開始前と終了後の設定

多軸同期１を用いた MCM の開始前には、ユーザアプリケーションで次の設定をし、
MCM の終了後に元に戻します。

軸	設定
マスター軸	UNIT MCM SPEC1 SET 関数で次の設定をする ・SS1 SEL を「機能なし」にする HARD INITIALIZE1 コマンドの OUT1 TYPE で OUT1 の出力機能を設定する
スレーブ軸	UNIT MCM SPEC1 SET 関数で次の設定をする ・SS1 OUT1 ASSIGN で自軸の SS1 にマスター軸の OUT1 を割り当てる ・SS1 SEL を「多軸同期１のマスター軸からの同期信号」にする HARD INITIALIZE3 コマンドの GPIO3 TYPE を「SS1」に設定する
マスター軸 兼 スレーブ軸	UNIT MCM SPEC1 SET 関数で次の設定をする ・SS1 OUT1 ASSIGN で自軸の SS1 に自軸の OUT1 を割り当てる ・SS1 SEL を「多軸同期１のマスター軸からの同期信号」にする HARD INITIALIZE1 コマンドの OUT1 TYPE で OUT1 の出力機能を設定する HARD INITIALIZE3 コマンドの GPIO3 TYPE を「SS1」に設定する

多軸同期１を実行する場合、マスター軸が同期信号をアクティブにする前に、全てのスレーブ軸が同期ポイントに達する必要があります。
スレーブ軸が同期ポイントに達する前にマスター軸が同期信号をアクティブにすると、スレーブ軸は同期信号を検出できません。

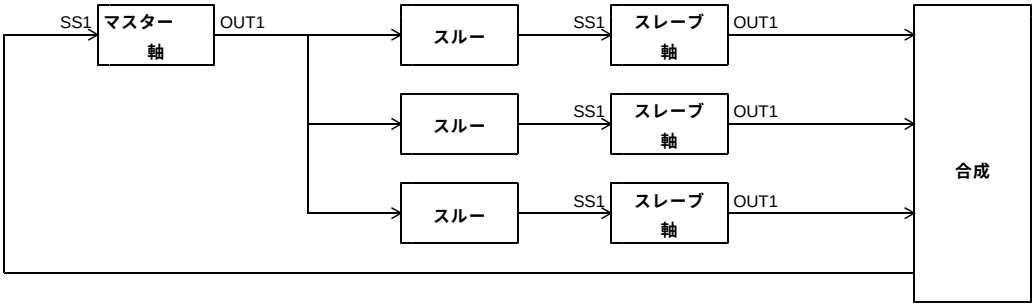
※ 多軸同期１を用いた MCM の中では、次の機能に SS1 または GPIO3 を使用しないでください。

・ORIGIN SPEC SET コマンドの ORG SIGNAL TYPE(SS1)
・RAM SPEC SET コマンドの NOP TYPE(GPIO3)

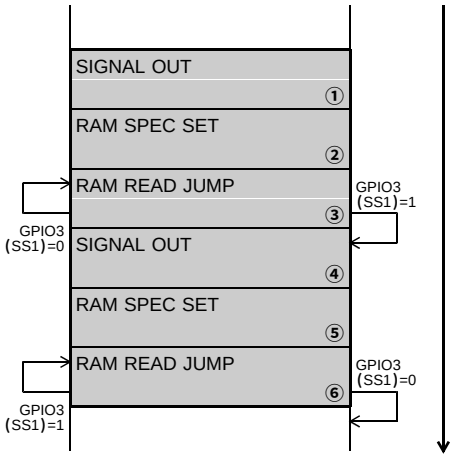
【多軸同期２の設定例】

マスター軸が特定の状態になるまでスレーブ軸の MCM **の実行を中断し、全てのスレーブ軸が特定の状態になるまでマスター軸の** MCM **の実行を中断します。**

- **マスター軸の** OUT1 = 1 **になると、全てのスレーブ軸の** GPIO3(SS1) = 1 **に、全てのスレーブ軸の** OUT1 = 1 **になると、マスター軸の** GPIO3(SS1) = 1 **になります**
- **マスター軸・スレーブ軸ともに、**GPIO3(SS1) = 0 **の間、待機し続ける** MCM **は、**GPIO3(SS1) = 1 **の検出で** OUT1 = 0 **にします**
- **マスター軸の** OUT1 = 0 **になると、全てのスレーブ軸の** SS1(GPIO3) = 1 **に、全てのスレーブ軸の** OUT1 = 0 **になると、マスター軸の** SS1(GPIO3) = 0 **になります。**
- **マスター軸・スレーブ軸ともに、**GPIO3 = 1 **の間待機し続ける** MCM **は、**GPIO3(SS1) = 0 **の検出で実行を再開します。**



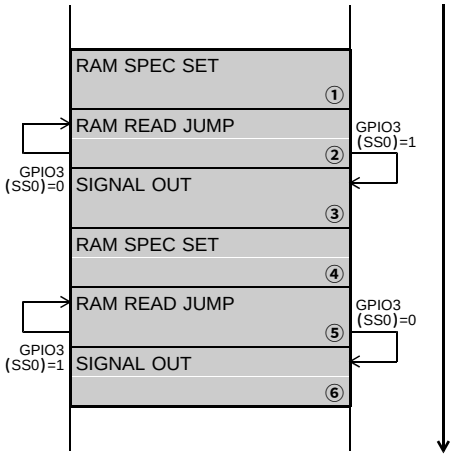
マスター軸



① SIGNAL OUT **コマンドで** OUT1 = 1 **にします。**
② RAM SPEC SET **コマンドで次の設定にします。**
・JUMP TYPE =**「**STATUS3 PORT **の** GPIO3 = 1 **のときに実行する」**
・JUMP ENABLE =**「常時有効にする」**
③ RAM READ JUMP **コマンドで、移動アドレスには次のアドレスを設定します**
④ SIGNAL OUT **コマンドで** OUT1 = 0 **にします。**
⑤ RAM SPEC SET **コマンドで次の設定にします。**
・JUMP TYPE =**「**STATUS3 PORT **の** GPIO3 = 0 **のときに実行する」**
・JUMP ENABLE =**「常時有効にする」**
⑥ RAM READ JUMP **コマンドで、移動アドレスには次のアドレスを**

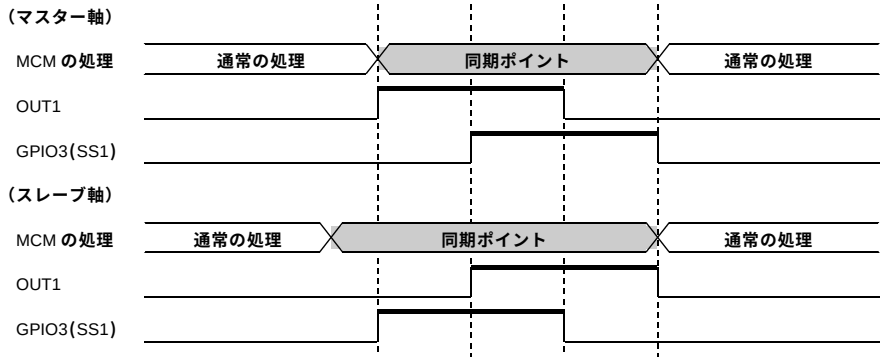
同期ポイント

スレーブ軸



① RAM SPEC SET **コマンドで次の設定にします。**
・JUMP TYPE =**「**STATUS3 PORT **の** GPIO3 = 1 **のときに実行する」**
・JUMP ENABLE =**「常時有効にする」**
② RAM READ JUMP **コマンドで、移動アドレスには次のアドレスを設定します**
③ SIGNAL OUT **コマンドで** OUT1 = 1 **にします。**
④ RAM SPEC SET **コマンドで次の設定にします。**
・JUMP TYPE =**「**STATUS3 PORT **の** GPIO3 = 0 **のときに実行する」**
・JUMP ENABLE =**「常時有効にする」**
⑤ RAM READ JUMP **コマンドで、移動アドレスには次のアドレスを**
⑥ SIGNAL OUT **コマンドで** OUT1 = 0 **にします。**

同期ポイント



● 同期信号
マスター軸・スレーブ軸とも、OUT1 に出力する同期信号として、汎用出力信号を選択します。

● MCM 開始前と終了後の設定
多軸同期２を用いた MCM の開始前には、ユーザアプリケーションで次の設定をし、
MCM の終了後に元に戻します。

軸	設定
マスター軸	・UNIT MCM SPEC1 SET 関数で次の設定をする ・SS1 OUT ASSIGN で、自軸の SS1 に全てのスレーブ軸の OUT1 を割り当てる ・SS1 SEL で「多軸同期２のスレーブ軸からの同期信号」を選択 ・HARD INITIALIZE1 コマンドの OUT1 TYPE で OUT1 の出力機能を「汎用出力」にする ・HARD INITIALIZE3 コマンドの GPIO3 TYPE を「SS1」に設定する
スレーブ軸	・UNIT MCM SPEC1 SET 関数で次の設定をする ・SS1 OUT ASSIGN で、自軸の SS1 にマスター軸の OUT1 を割り当てる ・SS1 SEL で「多軸同期２のマスター軸からの同期信号」を選択 ・HARD INITIALIZE1 コマンドの OUT1 TYPE で OUT1 の出力機能を「汎用出力」にする ・HARD INITIALIZE3 コマンドの GPIO3 TYPE を「SS1」に設定する

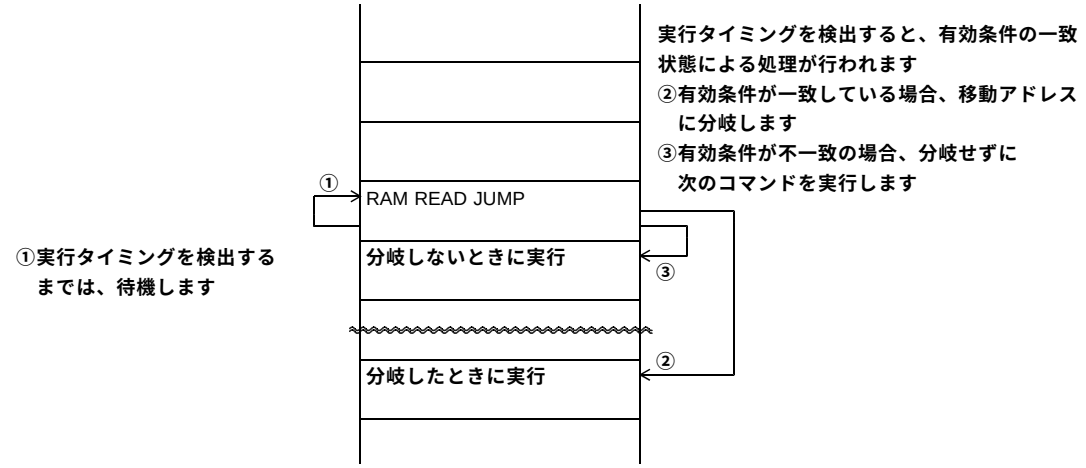
※ 多軸同期２を用いた MCM の中では、次の機能に SS1 または GPIO3 を使用しないでください。
・ORIGIN SPEC SET コマンドの ORG SIGNAL TYPE(SS1)
・RAM SPEC SET コマンドの NOP TYPE(GPIO3)

(4) MCM の分岐

以下のコマンドで、MCM は任意のアドレスに分岐します。
・RAM READ JUMP コマンド ：分岐先のアドレスから、コマンド自動実行の開始

RAM 領域の RAM READ JUMP コマンドは、実行タイミングの検出で実行します。
実行タイミングは、RAM SPEC SET コマンドの JUMP TYPE で選択します。

更に、RAM READ JUMP コマンドは、有効条件が一致しているときに実行します。
不一致のときには、RAM READ JUMP コマンドを無効にして、次のコマンドを読み出します。



※ RAM SPEC SET コマンド、RAM READ JUMP コマンドは、MCM の中で自動実行するコマンドです。
ユーザアプリケーションから実行することはできません。

(5) MCM を組む上での注意

● **MCC09 コマンド**

MCM は、リアルタイムなモーションと I/O 制御を提供できるように考慮されたものです。
USB シリーズコントローラで仕様公開されている MCC09 コマンドは、MCM のプログラムを組む上で特に制限を設けていません。
しかし、確実なリアルタイム性あるモーション制御を行わせるためには、ユーザシステム上の基本的なパラメータはユーザアプリケーションから予め設定した後に、下記のような MCM による制御を行うことを推奨します。

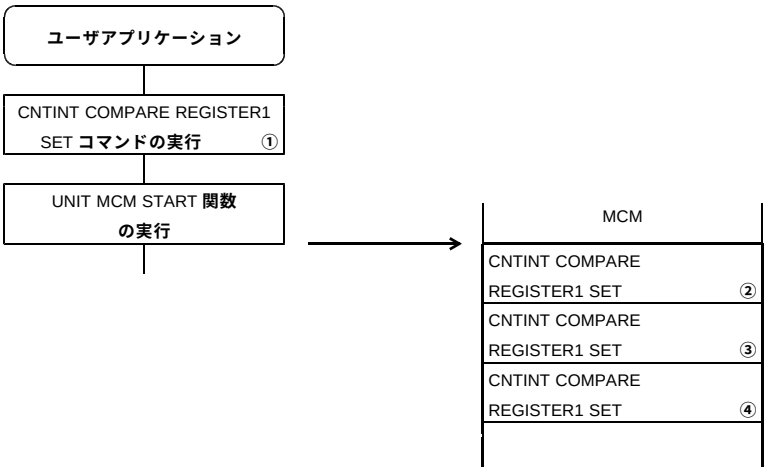
・ドライブ機能の設定と実行を行うコマンド
・パルスカウンタのコンパレータの設定
・汎用出力信号の操作

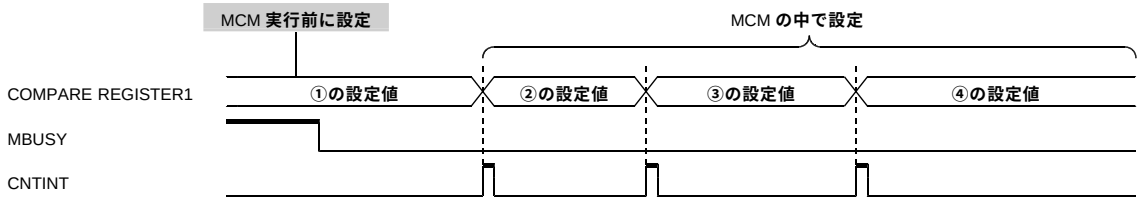
MCM 内で推奨しない MCC09 コマンド例

コマンドコード	コマンド名	備考
H'01	SPEC INITIALIZE1	ドライブパルスの出力仕様の設定
H'02	SPEC INITIALIZE2	CWLM, CCWLM, RDYINT, SS0, SS1 の設定
H'80	ADDRESS COUNTER PRESET	アドレスカウンタの現在位置の設定
H'81	ADDRESS COUNTER INITIALIZE1	アドレスカウンタの各機能の設定
H'82	ADDRESS COUNTER INITIALIZE2	アドレスカウンタの各機能の設定
H'87	ADDRESS COUNTER MAX COUNT SET	アドレスカウンタの最大カウント数の設定
H'88	ADRINT COMPARE REGISTER1 SET	ADRINT のコンペアレジスタ１の設定
H'89	ADRINT COMPARE REGISTER2 SET	ADRINT のコンペアレジスタ２の設定
H'8A	ADRINT COMPARE REGISTER3 SET	ADRINT のコンペアレジスタ３の設定
H'8C	ADRINT COMP1 ADD DATA SET	ADRINT の COMP1 ADD データの設定
H'90	PULSE COUNTER PRESET	パルスカウンタのカウント初期値の設定
H'91	PULSE COUNTER INITIALIZE1	パルスカウンタの各機能の設定
H'92	PULSE COUNTER INITIALIZE2	パルスカウンタの各機能の設定
H'97	PULSE COUNTER MAX COUNT SET	パルスカウンタの最大カウント数の設定
H'9C	CNTINT COMP1 ADD DATA SET	CNTINT の COMP1 ADD データの設定
H'A0	DFL COUNTER PRESET	パルス偏差カウンタのカウント初期値の設定
H'A1	DFL COUNTER INITIALIZE1	パルス偏差カウンタの各機能の設定
H'A2	DFL COUNTER INITIALIZE2	パルス偏差カウンタの各機能の設定
H'A3	DFL COUNTER INITIALIZE3	パルス偏差カウンタの各機能の設定
H'A8	DFLINT COMPARE REGISTER1 SET	DFLINT のコンペアレジスタ１の設定
H'A9	DFLINT COMPARE REGISTER2 SET	DFLINT のコンペアレジスタ２の設定
H'AA	DFLINT COMPARE REGISTER3 SET	DFLINT のコンペアレジスタ３の設定
H'AC	DFLINT COMP1 ADD DATA SET	DFLINT の COMP1 ADD データの設定
H'C1	SPEED CHANGE SPEC SET	SPEED CHANGE の変更動作点の設定
H'C3	INDEX CHANGE SPEC SET	INDEX CHANGE の変更動作点の設定
H'D1	ERROR STATUS READ	ERROR STATUS の読み出し
H'D4	MCC SPEED READ	ドライブパルス速度の読み出し
H'D5	SET DATA READ	MCC09 設定データの読み出し
H'D6	RSPD DATA READ	RSPD データの読み出し
H'D8	ADDRESS COUNTER READ	アドレスカウンタの読み出し
H'D9	PULSE COUNTER READ	パルスカウンタの読み出し
H'DA	DFL COUNTER READ	パルス偏差カウンタの読み出し
H'DC	ADDRESS LATCH DATA READ	アドレスカウンタのラッチデータの読み出し
H'DD	PULSE LATCH DATA READ	パルスカウンタのラッチデータの読み出し
H'DE	DFL LATCH DATA READ	パルス偏差カウンタのラッチデータの読み出し
H'E5	ERROR STATUS MASK	ERROR に出力する ERROR STATUS のマスク
H'E8	COUNT LATCH SPEC SET	カウントデータのラッチタイミングの設定
H'F1	HARD INITIALIZE1	OUT 0,1 出力機能、OUT2,3 ステータスの設定
H'F7	HARD INITIALIZE7	入力信号のアクティブ論理の選択
H'F8	HARD INITIALIZE8	出力信号のアクティブ論理の選択

● パルスカウンタ COMP1 設定コマンドの自動実行

パルスカウンタの COMP1 設定コマンドの自動実行を利用して MCM を構築する場合、
最初の検出値については、ユーザアプリケーションで設定してください。





4-3-6. 状態の表示

●各種 MCM 情報
UNIT MCM STATUS READ 関数を実行すると、ユニットの状態を読み出すことができます。

ユニットの状態	説明
MCM STATUS1	MCM に関連するユニットの現在の状態
MCM STATUS2	MCM に関連するユニットの現在の状態
パルスカウンタのカウントデータ	パルスカウンタのカウントデータ
汎用レジスタのデータ	NO OPERATION コマンドで設定された汎用レジスタのデータ
I/O PORT	汎用 I/O PORT、制御 I/O PORT、拡張 I/O PORT の入力状態

● MCM ERROR 情報
UNIT MCM ERROR STATUS READ 関数を実行すると、ユニット上で発生した動作エラーの
エラー情報を読み出すことができます。

動作エラーの分類	エラー情報
MCM 関数のエラー	MCM 関数のエラーのエラーコード
MCC のエラー	・ERROR STATUS READ コマンドの ERROR STATUS ・ERROR STATUS MASK コマンドの ERROR STATUS のマスクデータ
ORIGIN ドライブのエラー	ORIGIN STATUS のエラー情報

4-4. I/O 仕様

4-4-1. その他の I/O PORT

(1) SIGNAL I/O1 信号

UC-7660、UCD-7620/A5F31DE、UCD-7621/A5F41DE、UCD-7630/A5F31Q は、各軸の任意ステータス信号を、SIGNAL I/O1 コネクタの SOUT 信号から、外部出力することができます。

SOUT0、SOUT1、SOUT2、SOUT3 信号は接続する信号によって、オープンコレクタまたは
ラインドライバ出力が選択できます。初期値は下記の割付になっています。

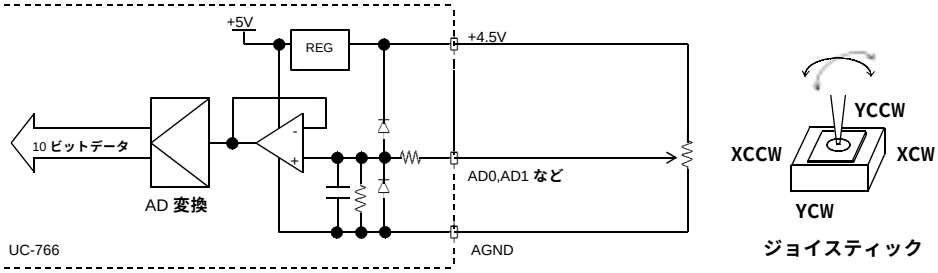
外部出力信号	軸と出力(割付信号名)	
	4 軸ユニット	2 軸ユニット
$\overline{\text{SOUT0}}$ 信号	XOUT0(XCNTINT)	XOUT0(XCNTINT)
$\overline{\text{SOUT1}}$ 信号	YOUT0(YCNTINT)	YOUT0(YCNTINT)
$\overline{\text{SOUT2}}$ 信号	ZOUT0(ZCNTINT)	XOUT0(XCNTINT)
$\overline{\text{SOUT3}}$ 信号	AOUT0(ACNTINT)	YOUT0(YCNTINT)

SOUT 信号の状態は、各軸の DRIVE STATUS3 PORT から読み出すことができます。

SOUT 信号の軸割り当てや、出力できる信号の詳細については、4-3-4.章 MCM の設定を
ご覧ください。UNIT MCM SPEC3 SET 関数により設定します

(2) SIGNAL I/O2 信号

UC-7660 は SIGNAL I/O2 コネクタに、0V ～+5V までのアナログ電圧を入力することができます。
ジョイスティックレバーの倒角度をアナログ電圧入力し、読み出したデジタル変換値によって、
UC-7660 の出力パルス速度を変化させるジョイスティック運転アプリケーションが可能です。



●変換データ

入力電圧値 ＝ 変換データ×入力レンジ÷分解能

入力電圧	AD変換データ(10ビット)
4.995V	1023 (H'3FF)
4.990V	1022 (H'3FE)
≀	≀
0.00488V	1 (H'001)
0V	0 (H'000)

例) 読み出されたAD変換データがH'1FFのとき
入力電圧 ＝ H'1FF×5V÷1024
＝ 511×5V÷1024
＝ 2.441V

●制御例

UC-7660 でサポートしている+4.5V 電源を使用した例を示します。
これらジョイスティックアプリケーションは、ADx の入力電圧を A/D DATA 読み出し関数または A/D DATA
A/D データ構造体読み出し関数で読み出し、その電圧情報に基づいて各軸をコマンド制御します。

・レバーの方向判別 … 4.5V/2 のセンター値 2.25V に対して、
(方向は任意) アナログ電圧が高いとき:CW 方向の SCAN DRIVE を起動します。
アナログ電圧が低いとき:CCW 方向の SCAN DRIVE を起動します。

・倒角と速度の制御 …読み出したアナログ電圧が 2.25V に対して、
(速度データは任意) アナログ電圧が高くなる、または低くなるに連れ速度を速く制御します。
2.25V ～ 4.5V の範囲(または 2.25V ～ 0V の範囲)をある任意な電圧区間毎に
SPEED CHANGE 指令で速度を変更します。

・停止処理 …レバーがニュートラルのポジションになったとき、STOP COMMAND でドライブを
(即時/減速停止は任意) 停止させます。

・ティーチング …停止したときのカウンタの値を読み出します。
(ティーチングは任意) その読み出した値を指定したパソコン内のメモリーに保存します。

*ジョイスティックを操作するサンプルプログラムについては、別途お問い合わせください。

5．付録

5-1.初期仕様一覧

(1) 応用設定

項　目	初期仕様	対応関数/コマンド
■入力信号のデジタルフィルタ機能	–	
■入出力信号のアクティブ論理		
FSSTOP 信号	正論理入力	HARD INITIALIZE7 コマンド
CWLM 信号	正論理入力	
CCWLM 信号	正論理入力	
DALM 信号	正論理入力	
$\overline{\text{ORG}}$ 信号	負論理入力	
$\overline{\text{NORG}}$ 信号	負論理入力	
PAUSE 信号	1 でアクティブ	
CWP 信号	負論理出力	HARD INITIALIZE8 コマンド
CCWP 信号	負論理出力	
■アドレスカウンタ		
最大カウント数	H'FFFF_FFFF	ADDRESS COUNTER MAX COUNT SET コマンド
■パルスカウンタ		
最大カウント数	H'FFFF_FFFF	PULSE COUNTER MAX COUNT SET コマンド
■カウンタのラッチ・クリア機能		
アドレスカウンタラッチタイミング	ADDRESS LATCH DATA READ コマンドの実行	COUNT LATCH SPEC SET コマンド
ADDRESS CLR ENABLE	クリアしない	
パルスカウンタラッチタイミング	PULSE LATCH DATA READ コマンドの実行	
PULSE CLR ENALBE	クリアしない	
パルス偏差カウンタラッチタイミング	DFL LATCH DATA READ コマンドの実行	
DFL CLR ENABLE	クリアしない	
■ RAM アクセス機能		
直接 RAM アクセス機能	無効	MCM SPEC0 SET 関数
■多用途センサ機能		
各軸 SS0 入力機能	汎用入力	SPEC INITIALIZE2 コマンド
X 軸 SS1 接続信号	割り当てなし(LOW 固定)	MCM SPEC1 SET 関数
Y 軸 SS1 接続信号	割り当てなし(LOW 固定)	
Z 軸 SS1 接続信号	割り当てなし(LOW 固定)	
A 軸 SS1 接続信号	割り当てなし(LOW 固定)	
X 軸 SS0 接続信号	/IN0 信号のスルー	MCM SPEC2 SET 関数
Y 軸 SS0 接続信号	/IN1 信号のスルー	
Z 軸 SS0 接続信号	割り当てなし(LOW 固定)	
A 軸 SS0 接続信号	割り当てなし(LOW 固定)	
■ステータス外部出力機能		
各軸 OUT0 出力ステータス	CNTINT	HARD INITIALIZE1 コマンド
各軸 OUT1 出力ステータス	RDYINT	
各軸 OUT2 ステータス	汎用出力	
各軸 OUT3 ステータス	汎用出力	
各軸 GPIO2 ステータス	汎用入力	HARD INITIALIZE2 コマンド
各軸 GPIO3 ステータス	汎用入力	HARD INITIALIZE3 コマンド
$\overline{\text{SOUT0}}$ 出力信号	X 軸 OUT0 のスルー	MCM SPEC3 SET 関数
$\overline{\text{SOUT1}}$ 出力信号	Y 軸 OUT0 のスルー	
$\overline{\text{SOUT2}}$ 出力信号	Z 軸 OUT0 のスルー	
$\overline{\text{SOUT3}}$ 出力信号	A 軸 OUT0 のスルー	
■同期スタート機能		
各軸 STBY 解除条件	PAUSE=0 で解除	SPEC INITIALIZE3 コマンド
各軸 SS1(同期)機能	機能なし	MCM SPEC1 SET 関数

(2) 応用ドライブパラメータ

項　目	初期仕様	対応関数/コマンド
■第 1 パルス出力周期　(FSPD)	5,000Hz	FSPD SET コマンド
■加減速パラメータ		
速度倍率　(RESOL)	1 (No.3)	HIGH SPEED SET コマンド
最高速時の速度データ　(HSPD)	3,000	
加速開始時の速度データ　(LSPD)	300	LOW SPEED SET コマンド
減速終了時の速度データ　(ELSPD)	300	
加速カーブ変速周期データ　(UCYCLE)	200	RATE SET コマンド
減速カーブ変速周期データ　(DCYCLE)	200	
加速カーブ S 字変速領域データ　(SUAREA)	0(変速領域なし)	SCAREA SET コマンド
減速カーブ S 字変速領域データ　(SDAREA)	0(変速領域なし)	
加速カーブ S 字変速領域データ　(SUH)	0(変速領域なし)	SHAREA SET コマンド
減速カーブ S 字変速領域データ　(SDH)	0(変速領域なし)	
減速パルス数のオフセットパルス数	1 パルス	DOWN PULSE ADJUST コマンド
■ ORIGIN ドライブパラメータ		
ORG 検出信号	$\overline{\text{ORG}}$ 信号と ± ZORG 信号の OR(論理和)	ORIGIN SPEC SET コマンド
ORG 検出信号の検出エッジ	ORG 検出信号の 0 → 1(アクティブ)エッジ	
ORG STOP 型式	常時検出/エッジ検出で停止しない	
停止時 DRST 出力	出力しない	
検出エッジのカウント数	1 カウント目のエッジ検出で停止	
■補間ドライブパラメータ		
CPPOUT 端子から出力するパルス	パルス出力なし	CP SPEC SET コマンド
直線補間 長軸の目的地の座標アドレス	H'0000_0000	LONG POSITION SET コマンド
直線補間 短軸の目的地の座標アドレス	H'0000_0000	SHORT POSITION SET コマンド
円弧補間 現在位置の X 座標アドレス	H'00_0000	CIRCULAR XPOSITION SET コマンド
円弧補間 現在位置の Y 座標アドレス	H'00_0000	CIRCULAR YPOSITION SET コマンド
円弧補間 目的地までの短軸パルス数	H'0000_0000	CIRCULAR PULSE SET コマンド
■ドライブ CHANGE パラメータ		
SPEED RATE CHANGE 変更動作点	STATUS1 PORT の DRIVE = 1 で実行	SPEED CHANGE SPEC SET コマンド
INDEX CHANGE 変更動作点	STATUS1 PORT の DRIVE = 1 で実行	INDEX CHANGE SPEC SET コマンド

5-2. 関数一覧

種別	名称 / 説明	記号	ページ 標準編	ページ 応用編
基本構造体	RESULT構造体 関数を実行した結果を格納する	MC07_S_RESULT	30	
	コマンドデータ構造体 DRIVE COMMAND PORT、DRIVE DATA1 PORT。DRIVE DATA2 PORTに書き込むデータを格納する	MC07_S_COMMAND_DATA	32	
	ステータスデータ構造体 DRIVE STATUS1 PORT、DRIVE DATA1 PORT、DRIVE DATA2 PORTから読み出した内容を格納する	MC07_S_STATUS_DATA	33	
システム	ユニット情報構造体 パソコンに接続される全ユニットのタイプを格納する	MC07_S_UNIT_INFO	34	
	環境設定関数 パソコンに接続されている全ユニットを対象に環境設定を行う	MC07_Environment	35	
	ユニット情報読み出し関数 環境設定されている全ユニットのタイプを読み出す	MC07_ReadUnitInfo	36	
ユニット	ユニットオープン関数 指定ユニット番号でユニットオープンし、引数*phUnit*の変数にユニットハンドルを格納する	MC07_UOpen	37	
	ユニットクローズ関数 指定されたユニットをクローズする	MC07_UClose	38	
	ユニット動作エラークリア関数 指定ユニットに対し、指定された軸の動作エラークリア処理を一括で行う	MC07_UClrError	39	
	拡張ユニット通信設定関数 指定されたユニットと拡張ユニット間の通信設定を行う	MC07_UWExUnitCommMode	40	
	拡張ユニット通信制御関数 指定されたユニットと拡張ユニット間の通信を制御する	MC07_UWExUnitCommControl	41	
	拡張ユニット通信ステータス読み出し関数 指定されたユニットと拡張ユニット間の通信の状態を読み出す	MC07_URExUnitCommStatus	42	
	拡張ユニット通信設定読み出し関数 指定されたユニットと拡張ユニット間の通信設定を読み出す	MC07_URExUnitCommMode	43	
ユニットアクセス	ユニットステータス構造体 ユニットのステータスの内容を格納する	MC07_S_UNIT_STATUS	44	
	ユニットコマンド構造体 ユニットのコマンドを格納する	MC07_S_UNIT_COMMAND	45	
	入力PORT構造体 ユニットの入力PORTから読み出された内容を格納する	MC07_S_IN_PORT	46	
	出力PORT構造体 ユニットの出力PORTに書き込むデータ,OR書き込みデータ,AND書き込みデータを格納する	MC07_S_OUT_PORT	47	
	ユニットDRIVE COMMAND･I/O書き込み関数 指定されたユニットに対し、各軸のDATA,COMMAND,各I/O PORTデータの書き込みを一括で行う	MC07_UWDriveIo	48	
	ユニットDRIVE COMMAND書き込み/読み出し関数 指定されたユニットに対し、各軸のDATA,COMMANDを書き込み、読み出しする順の処理を一括で行う	MC07_UWRDrive	49	
	ユニットSTATUS1･I/O読み出し関数 指定されたユニットに対し、各軸のSTATUS1 PORT、I/O PORTの読み出しを一括で行う	MC07_URStatus1Io	50	
	ユニットSTATUS1･パルスカウンタ･I/O読み出し関数 指定されたユニットに対し、各軸のSTATUS1 PORT、パルスカウンタ値、I/O PORTの読み出しを一括で行う	MC07_URStatus1PcntIo	52	
	ユニットI/O PORT書き込み関数 指定されたユニットに対し、各I/O PORT毎の個別データを書き込む	MC07_UPortOUT	54	
	ユニットI/O PORT OR書き込み関数 指定されたユニットに対し、各I/O PORT毎の個別データをOR書き込む	MC07_UPortOrOUT	55	
	ユニットI/O PORT AND書き込み関数 指定されたユニットに対し、各I/O PORT毎の個別データをAND書き込む	MC07_UPortAndOUT	56	
	ユニットI/O PORT読み出し関数 指定されたユニットに対し、各I/O PORTの内容を一括で読み出す	MC07_UPortIn	57	
	A/D DATA構造体 全てのアナログ入力信号をA/D変換したデータを格納する	MC07_S_AD_DATA		10
	A/D DATA読み出し関数 指定されたアナログ入力信号をA/D変換し、データを読み出す	MC07_BRADData		11
	A/D DATA A/Dデータ構造体読み出し関数 全てのアナログ入力信号をA/D変換し、データを読み出す	MC07_IRADData		12

5-3.ドライブコマンド一覧

(1) 汎用コマンド

種別	コマンド名 / 説明	コマンドコード	ページ 標準編	ページ 応用編
ドライバ固有	NO OPERATION 機能なし	H'00	138	
	SPEC INITIALIZE1 ドライブパルスの出力仕様の設定	H'01	115	
	SPEC INITIALIZE2 CWLM,CCWLM,SSOの設定	H'02	116	
	SPEC INITIALIZE3 DRST,DEND/PO,DALMの設定	H'03	118	
	FSPD SET ドライブパルス出力の第１パルス目のパルス周期（パルス速度)を設定	H'05		74
	HIGH SPEED SET 加減速ドライブの最高速時のパルス速度データ(HSPD)、速度データの速度倍率(RESOL)を設定	H'06		75
	LOW SPEED SET 加減速ドライブの加速開始時パルス速度データ(LSPD)、減速終了時パルス速度データ(ELSPD)を設定	H'07		76
	RATE SET 加速カーブの変速周期データ(UCYCLE)、減速カーブの変速周期データ(DCYCLE)を設定	H'08		77
	SCAREA SET 加速カーブのS字変速領域データ(SUAREA)、減速カーブのS字変速領域データ(SDAREA)を設定	H'09		78
	DOWN PULSE ADJUST INDEXドライブの自動減速停止動作を開始する減速パルス数のオフセットパルス数を設定	H'0A		80
	SHAREA SET 加速カーブのS字変速領域データ(SUH)、減速カーブのS字変速領域データ(SDH)を設定	H'0B		79
	JSPD SET JOGドライブのパルス速度の設定	H'0C	122	
	JOG PULSE SET JOGドライブのパルス数の設定	H'0D	123	
	ORIGIN SPEC SET ORIGINドライブの動作仕様を設定	H'0F		81
	+JOG +方向JOGドライブの実行	H'10	124	
	-JOG -方向JOGドライブの実行	H'11	125	
	+SCAN +方向SCANドライブの実行	H'12	126	
	-SCAN -方向SCANドライブの実行	H'13	126	
	INC INDEX 相対アドレスINDEXドライブの実行	H'14	127	
	ABS INDEX 絶対アドレスINDEXドライブの実行	H'15	128	
	+JSPD SCAN +方向JSPD SCANドライブの実行	H'16	125	
	-JSPD SCAN -方向JSPD SCANドライブの実行	H'17	125	
	SERVO RESET DRST信号出力に10ms間アクティブを出力	H'1F	131	

種別	コマンド名	コマンドコード	ページ	
	説明		標準編	応用編
ドライバ固有	CP SPEC SET	H'20		83
	相関2軸1チップのCPPOUT端子から出力するパルスを設定			
	LONG POSITION SET	H'22		90
	直線補間ドライブの、長軸の座標アドレスを設定			
	SHORT POSITION SET	H'23		91
	直線補間ドライブの、短軸の座標アドレスを設定			
	CIRCULAR XPOSITION SET	H'28		99
	円弧の中心点座標を(0,0)とした現在位置のX座標アドレスを設定			
	CIRCULAR YPOSITION SET	H'29		100
	円弧の中心点座標を(0,0)とした現在位置のY座標アドレスを設定			
	CIRCULAR PULSE SET	H'2A		101
	現在位置のX-Y座標アドレスから目的地の短軸座標までの短軸パルス数を設定			
	MAIN STRAIGHT CP	H'30		92
	任意軸、複数軸の直線補間ドライブのときのメイン軸に実行			
	SUB STRAIGHT CP	H'31		93
	任意軸、複数軸の直線補間ドライブのときのサブ軸に実行			
	MAIN CIRCULART CP	H'38		102
	任意軸の円弧補間ドライブのときのメイン軸に実行			
	SUB CIRCULART CP	H'39		103
	任意軸の円弧補間ドライブのときのサブ軸に実行			

(2) 特殊コマンド

種別	コマンド名 / 説明	コマンドコード	ページ 標準編	ページ 応用編
エンコーダ系	ADDRESS COUNTER PRESET アドレスカウンタの現在位置設定	H'80	144	
	ADDRESS COUNTER INITIALIZE1 アドレスカウンタの各機能の設定	H'81	139	
	ADDRESS COUNTER INITIALIZE2 アドレスカウンタの各機能の設定	H'82	142	
	ADDRESS COUNTER MAX COUNT SET アドレスカウンタの最大カウント数の設定	H'87		115
	ADRINT COMPARE REGISTER1 SET ADRINTのコンペアレジスタ1の設定	H'88	145	
	ADRINT COMPARE REGISTER2 SET ADRINTのコンペアレジスタ2の設定	H'89	145	
	ADRINT COMPARE REGISTER3 SET ADRINTのコンペアレジスタ3の設定	H'8A	145	
	ADRINT COMP1 ADD DATA SET ADRINTのCOMP1 ADDデータの設定	H'8C	146	
	PULSE COUNTER PRESET パルスカウンタのカウント初期値の設定	H'90	152	
	PULSE COUNTER INITIALIZE1 パルスカウンタの各機能の設定	H'91	147	
	PULSE COUNTER INITIALIZE2 パルスカウンタの各機能の設定	H'92	150	
	PULSE COUNTER MAX COUNT SET パルスカウンタの最大カウント数の設定	H'97		116
	CNTINT COMPARE REGISTER1 SET CNTINTのコンペアレジスタ1の設定	H'98	153	
	CNTINT COMPARE REGISTER2 SET CNTINTのコンペアレジスタ2の設定	H'99	153	
	CNTINT COMPARE REGISTER3 SET CNTINTのコンペアレジスタ3の設定	H'9A	153	
	CNTINT COMP1 ADD DATA SET CNTINTのCOMP1 ADDデータの設定	H'9C	154	
	DFL COUNTER PRESET パルス偏差カウンタのカウント初期値の設定	H'A0	163	
	DFL COUNTER INITIALIZE1 パルス偏差カウンタの各機能の設定	H'A1	155	
	DFL COUNTER INITIALIZE2 パルス偏差カウンタの各機能の設定	H'A2	158	
	DFL COUNTER INITIALIZE3 パルス偏差カウンタの各機能の設定	H'A3	161	
	DFLINT COMPARE REGISTER1 SET DFLINTのコンペアレジスタ1の設定	H'A8	164	
	DFLINT COMPARE REGISTER2 SET DFLINTのコンペアレジスタ2の設定	H'A9	164	
	DFLINT COMPARE REGISTER3 SET DFLINTのコンペアレジスタ3の設定	H'AA	164	
	DFLINT COMP1 ADD DATA SET DFLINTのCOMP1 ADDデータの設定	H'AC	165	

種別	コマンド名	コマンドコード	ページ	
	説明		標準編	応用編
モーション系	RAM SPEC SET	H'B0		110
	RAM READ JUMP, NO OPERATIONの実行タイミングと有効条件を設定			
	RAM READ JUMP	H'B9		112
	移動アドレスからRAM領域のコマンド読み出しを開始			
	RAM STOP	H'BF		113
	RAM領域のコマンド読み出しを終了			
	SPEED CHANGE SPEC SET	H'C1		104
	SPEED CHANGE指令を実行する変更動作点を設定			
	INDEX CHANGE SPEC SET	H'C3		106
	INDEX CHANGE指令を実行する変更動作点を設定する			
	SPEED RATE CHANGE	H'C8		105
	実行中のパルス出力を指定ドライブパルス速度とRATEで加速または減速させる			
	INC INDEX CHANGE	H'CC		107
	指定したデータを起動位置を原点とする相対アドレスの停止位置に設定してINC INDEXを行う			
	ABS INDEX CHANGE	H'CD		108
	指定データをアドレスカウンタで管理している絶対アドレスの停止位置に設定してABS INDEXを行う			
	PLS INDEX CHANGE	H'CE		109
	指定データを変更点の検出位置を原点とする相対アドレスの停止位置に設定してINC INDEXを行う			
	ERROR STATUS READ	H'D1	133	
	ERROR STATUSの読み出し			
	MCC SPEED READ	H'D4	135	
	ドライブパルス速度の読み出し			
	MCC SET DATA READ	H'D5	136	
	設定データの読み出し			
	RSPD DATA READ	H'D6		114
	RSPDデータの読み出し			
	ADDRESS COUNTER READ	H'D8	166	
	アドレスカウンタの読み出し			
	PULSE COUNTER READ	H'D9	166	
	パルスカウンタの読み出し			
	DFL COUNTER READ	H'DA	166	
	パルス偏差カウンタの読み出し			
	ADDRESS LATCH DATA READ	H'DC		119
	アドレスカウンタのラッチデータを読み出す			
	PULSE LATCH DATA READ	H'DD		119
	パルスカウンタのラッチデータを読み出す			
	DFL LATCH DATA READ	H'DE		119
	パルス偏差カウンタのラッチデータを読み出す			
	ERROR STATUS MASK	H'E5	132	
	ERRORに出力するERROR STATUSのマスク			
	COUNT LATCH SPEC SET	H'E8		117
	各種カウンタのカウントデータをラッチするタイミングとクリア機能を設定			

種別	コマンド名	コマンドコード	ページ	
	説明		標準編	応用編
	HARD INITIALIZE1	H'F1		67
	ステータス信号(SOUT信号)に出力する機能を設定			
	HARD INITIALIZE2	H'F2		69
	GPIO2信号の入出力機能を設定			
	HARD INITIALIZE3	H'F3		70
	GPIO3信号の入出力機能を設定			
	HARD INITIALIZE7	H'F7		71
	CWLM信号、CCWLM信号、FSSTOP信号、DALM信号の入力アクティブ論理を設定			
	HARD INITIALIZE8	H'F8		73
	CWP信号、CCWP信号の出力アクティブ論理を設定			
	SIGNAL OUT	H'FC	130	
	汎用出力信号の操作			
	SLOW STOP	H'FE	129	
	減速停止の実行			
	FAST STOP	H'FF	129	
	即時停止の実行			

本版で改訂された主な箇所

箇　所	内　容
	なし

■ 製品保証

保証期間と保証範囲について

● 納入品の保証期間は、納入後１ヶ年と致します。
● 上記保証期間中に当社の責により故障を生じた場合は、その修理を当社の責任において行います。
(日本国内のみ)
ただし、次に該当する場合は、この保証対象範囲から除外させて頂きます。
(1) お客様の不適当な取り扱い、ならびに使用による場合。
(2) 故障の原因が、当製品以外からの事由による場合。
(3) お客さまの改造、修理による場合。
(4) 製品出荷当時の科学・技術水準では予見が不可能だった事由による場合。
(5) その他、天災、災害等、当社の責にない場合。
(注1)ここでいう保証は､納入品単体の保証を意味するもので納入品の故障により誘発される損害はご容赦頂きます。
(注2)当社において修理済みの製品に関しましては、保証外とさせて頂きます。

技術相談のお問い合わせ

TEL.(042)664-5382 FAX.(042)666-5664
E-mail s-support@melec-inc.com

販売に関するお問い合わせ

TEL.(042)664-5384 FAX.(042)666-2031

株式会社メレック 制御機器営業部
〒193-0834 東京都八王子市東浅川町516-10 URL:http://www.melec-inc.com