



STEPPING & SERVO MOTOR CONTROLLER'S OPTION

MPL-36-02_{V2.00}/USBW32

MPL-37-02_{V2.00}/USBW64

**取扱説明書
(設計者用)**

(USBシリーズ用 Windowsデバイスドライバ)

MCC07ユニット編

USER'S MANUAL

本製品を使用する前に、この取扱説明書を良く読んで十分に理解してください。
この取扱説明書は、いつでも取り出して読めるように保管してください。

はじめに

このデバイスドライバ「取扱説明書」は、USB シリーズのステッピングモータ、サーボモータ、および I/O システム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、Windows における標準的な機能および仕様について説明しています。

各コントローラの「取扱説明書」と同様に、本デバイスドライバ「取扱説明書」を良く読んで十分に理解してください。

このデバイスドライバ「取扱説明書」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を負いません。

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。

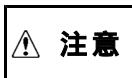
そのような事故の多くは、危険な状況を予め知っていれば回避することができます。

そのため、このデバイスドライバ「取扱説明書」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。



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。



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。

御使用前に

USB シリーズの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-766	X 軸	Y 軸	Z 軸	A 軸
UCD-7610v1/ADB5F30 *1	X 軸	Y 軸		
UCD-7613v1/GDB5F40 *1	X 軸	Y 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。

MCC09 搭載製品については、別冊デバイスドライバ取扱説明書「MCC09 ユニット編」をご覧ください。

応用機能については、別冊デバイスドライバ「取扱説明書」**応用機能編**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

PAGE

1. 概要	
1-1. 特徴	9
2. 関数仕様	
2-1. 一般仕様	10
2-2. 互換性	11
(1) 32ビット版と64ビット版との互換性	11
(2) 従来製品との互換性	11
2-3. ソフト開発に必要なファイル	12
(1) MPL-36-02v2.00/USBW32 (32ビット版)	12
(2) MPL-37-02v2.00/USBW64 (64ビット版)	12
3. 関数の説明	
3-1. システムの構成	13
3-2. システムの制御	14
(1) ユニットの制御	14
(2) ユニットを介した拡張ユニットの制御	14
3-3. 関数体系	15
(1) システム関数	15
(2) ユニット関数	15
(3) デバイス関数	15
(4) I/O PORT 関数	15
3-4. エラー	16
(1) 関数エラー	16
(2) 動作エラー	17
3-5. シーケンス	18
(1) 全体シーケンス	18
(2) ユニット制御シーケンス	19
(3) デバイス制御シーケンス	20
(4) I/O PORT 制御シーケンス	21
(5) MCC07 の実行シーケンス	21
3-6. 並列・並行処理	22
(1) マルチプロセス未対応	22
(2) マルチスレッド対応	23
3-7. 制限事項	24
(1) MCC07 コマンドの制限	24
(2) ORIGIN ドライブに関連する制限	24
(3) READ コマンドに関連する制限	25
4. 関数リファレンス	
4-1. 構造体・関数の見方	26
(1) 構造体	26
(2) 関数	26
(3) 言語固有の仕様	27
4-2. 基本構造体(共通)	29
4-2-1. RESULT 構造体	29
RESULT 構造体	29
4-2-2. コマンドデータ構造体	31
コマンドデータ構造体	31
4-2-3. ステータスデータ構造体	32
ステータスデータ構造体	32

目 次

	PAGE
4-3. システム関数 -----	33
4-3-1. ユニット情報構造体 -----	33
ユニット情報構造体 -----	33
4-3-2. 環境設定 / 接続確認関数 -----	34
環境設定関数 -----	34
ユニット情報読み出し関数 -----	35
4-4. ユニット関数 -----	36
4-4-1. ユニットオープン / クローズ関数 -----	36
ユニットオープン関数 -----	36
ユニットクローズ関数 -----	37
4-4-2. ユニット動作エラークリア関数 -----	38
ユニット動作エラークリア関数 -----	38
4-4-3. 拡張ユニット通信関数 -----	39
拡張ユニット通信設定関数 -----	39
拡張ユニット通信制御関数 -----	40
拡張ユニット通信ステータス読み出し関数 -----	41
拡張ユニット通信設定読み出し関数 -----	42
4-4-4. ユニットアクセス関数 -----	43
ユニットステータス構造体 -----	43
ユニットコマンド構造体 -----	44
入力 PORT 構造体 -----	45
出力 PORT 構造体 -----	46
ユニット DRIVE COMMAND ・ I/O 書き込み関数 -----	47
ユニット DRIVE COMMAND 書き込み / 読み出し関数 -----	48
ユニット STATUS1 ・ I/O 読み出し関数 -----	49
ユニット STATUS1 ・ パルスカウンタ ・ I/O 読み出し関数 -----	51
ユニット I/O PORT 書き込み関数 -----	53
ユニット I/O PORT OR 書き込み関数 -----	54
ユニット I/O PORT AND 書き込み関数 -----	55
ユニット I/O PORT 読み出し関数 -----	56
4-5. デバイス関数 -----	57
4-5-1. デバイスオープン / クローズ関数 -----	57
デバイスオープン関数 -----	57
デバイスクローズ関数 -----	58
4-5-2. 動作エラークリア関数 -----	59
動作エラークリア関数 -----	59
4-5-3. MCC07 PORT アクセス関数 -----	60
DRIVE COMMAND 32 ビット書き込み関数 -----	62
DRIVE COMMAND PORT 書き込み関数 -----	63
DRIVE STATUS1 PORT 読み出し関数 -----	64
DRIVE STATUS2 PORT 読み出し関数 -----	68
DRIVE STATUS3 PORT 読み出し関数 -----	70
DRIVE STATUS4 PORT 読み出し関数 -----	71
DRIVE STATUS5 PORT 読み出し関数 -----	73
STATUS PORT バッファ読み出し関数 -----	76
DRIVE COMMAND 32 ビット書き込み / 読み出し関数 -----	77
4-5-4. WAIT 関数 -----	78
READY WAIT 関数 -----	78
WAIT 状態読み出し関数 -----	79
WAIT 中止関数 -----	80
4-5-5. SPEED ・ RATE 関数 -----	81
SPEED ・ RATE 構造体 -----	81
SPEED ・ RATE セット関数 -----	83
SPEED ・ RATE 読み出し関数 -----	84

目 次

PAGE

4-5-6. 補間ドライブ関数 -----	85
POSITION 構造体 -----	85
2 軸相対アドレス直線補間ドライブ関数 -----	86
2 軸相対アドレス円弧補間ドライブ関数 -----	88
メインチップ 2 軸相対アドレス直線補間ドライブ関数 -----	90
メインチップ 2 軸相対アドレス円弧補間ドライブ関数 -----	92
円の中心点ゲット関数 -----	94
相対アドレス変換関数 -----	95
4-5-7. ORIGIN 関数 -----	96
ORIGIN ドライブパラメータ構造体 -----	96
ORIGIN ドライブステータス読み出し関数 -----	97
ORIGIN SPEC SET 関数 -----	99
ORIGIN MARGIN PULSE SET 関数 -----	102
ORIGIN DELAY SET 関数 -----	103
ORIGIN ERROR PULSE SET 関数 -----	104
ORIGIN OFFSET PULSE SET 関数 -----	105
ORIGIN PRESET PULSE SET 関数 -----	106
ORIGIN ドライブパラメータ読み出し関数 -----	107
ORIGIN FLAG RESET 関数 -----	108
ORIGIN ドライブ関数 -----	109
4-6. I/O PORT 関数 -----	110
4-6-1. I/O オープン/クローズ関数 -----	110
I/O PORT オープン関数 -----	110
I/O PORT クローズ関数 -----	111
4-6-2. I/O アクセス関数 -----	112
I/O PORT 書き込み関数 -----	112
I/O PORT OR 書き込み関数 -----	113
I/O PORT AND 書き込み関数 -----	114
I/O PORT 読み出し関数 -----	115

5. コマンド仕様

5-1. ドライブコマンド -----	116
5-1-1. 入出力仕様の設定 -----	116
(1) SPEC INITIALIZE1 -----	116
(2) SPEC INITIALIZE2 -----	117
(3) SPEC INITIALIZE3 -----	119
5-1-2. ドライブパラメータの設定 -----	122
(1) JSPD SET -----	122
(2) JOG PULSE SET -----	123
5-1-3. ドライブの実行 -----	124
(1) +JOG -----	124
(2) -JOG -----	125
(3) +SCAN -----	126
(4) -SCAN -----	126
(5) INC INDEX -----	127
(6) ABS INDEX -----	128
5-1-4. 停止コマンドの実行 -----	129
(1) SLOW STOP -----	129
(2) FAST STOP -----	129
5-1-5. 出力信号の操作 -----	130
(1) SIGNAL OUT -----	130
(2) DRST OUT -----	131

目 次

PAGE

5-1-6. エラー機能の設定と読み出し -----	132
(1) ERROR STATUS MASK -----	132
(2) ERROR STATUS READ -----	133
5-1-7. 速度・設定データの読み出し -----	135
(1) MCC SPEED READ -----	135
(2) MCC SET DATA READ -----	136
5-1-8. その他 -----	138
(1) NO OPERATION -----	138
5-2. カウンタコマンド -----	139
5-2-1. アドレスカウンタの設定 -----	139
(1) ADDRESS COUNTER INITIALIZE1 -----	139
(2) ADDRESS COUNTER INITIALIZE2 -----	142
(3) ADDRESS COUNTER PRESET -----	144
(4) ADRINT COMPARE REGISTER1,2,3 SET -----	145
(5) ADRINT COMP1 ADD DATA SET -----	146
5-2-2. パルスカウンタの設定 -----	147
(1) PULSE COUNTER INITIALIZE1 -----	147
(2) PULSE COUNTER INITIALIZE2 -----	150
(3) PULSE COUNTER PRESET -----	152
(4) CNTINT COMPARE REGISTER1,2,3 SET -----	153
(5) CNTINT COMP1 ADD DATA SET -----	154
5-2-3. パルス偏差カウンタの設定 -----	155
(1) DFL COUNTER INITIALIZE1 -----	155
(2) DFL COUNTER INITIALIZE2 -----	158
(3) DFL COUNTER INITIALIZE3 -----	161
(4) DFL COUNTER PRESET -----	162
(4) DFLINT COMPARE REGISTER1,2,3 SET -----	163
(5) DFLINT COMP1 ADD DATA SET -----	164
5-2-4. カウントデータの読み出し -----	165
(1) ADDRESS COUNTER READ -----	165
(2) PULSE COUNTER READ -----	165
(2) DFL COUNTER READ -----	165
6. 機能説明	
6-1. ドライブ仕様 -----	166
6-1-1. 入出力仕様 -----	166
(1) パルス出力仕様 -----	166
(2) サーボ対応機能 -----	167
6-1-2. ドライブパラメータ -----	168
(1) 第 1 パルス出力周期 -----	168
(2) 加減速パラメータ -----	169
(3) JOG パラメータ -----	176
6-1-3. 基本ドライブ -----	177
(1) JOG ドライブ -----	177
(2) SCAN ドライブ -----	178
(3) INDEX ドライブ -----	179
6-1-4. ORIGIN ドライブ -----	180
(1) ORIGIN ドライブ仕様 -----	180
(2) ORG-0 ドライブ型式 -----	184
(3) ORG-1 ドライブ型式 -----	186
(4) ORG-2 ドライブ型式 -----	188
(5) ORG-3 ドライブ型式 -----	189
(6) ORG-4, ORG-5 ドライブ型式 -----	190
(7) ORG-10 ドライブ型式 -----	193
(8) ORG-11 ドライブ型式 -----	194
(9) ORG-12 ドライブ型式 -----	195

目 次

PAGE

6-1-5. 補間ドライブ -----	196
(1) 補間ドライブ仕様 -----	196
(2) 直線補間ドライブ仕様 -----	197
(3) 円弧補間ドライブ仕様 -----	198
(4) コマンド予約機能使用時の任意 2 軸補間ドライブの制限 -----	200
(5) 線速一定制御 -----	201
6-1-6. パルス出力停止機能 -----	203
(1) 減速停止機能 -----	203
(2) 即時停止機能 -----	203
(3) LIMIT 停止機能 -----	204
6-1-7. MCC07 エラー出力機能 -----	205
6-1-8. 読み出し機能 -----	207
(1) ステータス読み出し -----	207
(2) 各データの読み出し -----	207
6-2. カウンタ仕様 -----	208
(1) エンコーダパルス入力方式 -----	208
(2) 外部パルス出力機能 -----	209
(3) アドレスカウンタ -----	211
(4) パルスカウンタ -----	212
(5) パルス偏差カウンタ -----	213
(6) コンパレータ機能 -----	216
6-3. I/O 仕様 -----	218
6-3-1. 汎用 I/O PORT -----	218
(1) コントローラの I/O PORT -----	218
(2) コントローラドライバの I/O PORT -----	219
6-3-2. その他の I/O PORT 機能 -----	220
(1) コントローラ本体の入力 PORT -----	220
(2) 出力 PORT -----	221
7. 付録	
7-1. 初期仕様一覧 -----	222
(1) 基本設定 -----	222
(2) 基本ドライブパラメータ -----	223
7-2. 関数一覧 -----	224
7-3. ドライブコマンド一覧 -----	227
(1) 汎用コマンド -----	227
(2) 特殊コマンド -----	228

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- 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 搭載したスレーブコントローラ UC-7660 に加えた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 搭載製品仕様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。

各関数は、パルスジェネレータの MCC07 PORT、または I/O PORT(制御 I/O、汎用 I/O、拡張 I/O およびアナログ入力)のアクセス(読み出し / 書き込み)を行うためのものです。

このように、ポートをアクセスするだけのシンプルな関数構造のため、原則、関数仕様によってモーション仕様が制限されることはありません。

MCC07 のコマンドの与え方によって、簡単な機能から応用的な機能に至るまで、用途に合わせた幅広いモーション制御を行うことができます。

[コントローラ]

MCC07 PORT(各軸)	汎用 I/O(各 2 点/ユニット)	制御 I/O(各軸)	アナログ入力 *1
・DRIVE COMMAND PORT	・汎用 I/O 出力 PORT	・制御 I/O 出力 0 PORT	・AN0 入力
・DRIVE DATA1 PORT	・汎用 I/O 入力 PORT	・制御 I/O 入力 0 PORT	・AN1 入力
・DRIVE DATA2 PORT			・AN2 入力
・DRIVE STATUS1 PORT			・AN3 入力
・DRIVE STATUS2 PORT			
・DRIVE STATUS3 PORT			
・DRIVE STATUS4 PORT			
・DRIVE STATUS5 PORT			

コントローラユニットには、
・MCC07 搭載コントローラ
・MCC07 搭載コントローラドライバ
のユニットがあります。

*1 アナログ入力機能は UC-766 でサポートしています。

[拡張 I/O]

拡張 I/O
・拡張 I/O 出力 0 PORT(16 点)
・拡張 I/O 出力 1 PORT(16 点)
・拡張 I/O 入力 0 PORT(16 点)
・拡張 I/O 入力 1 PORT(16 点)

* 拡張 I/O ユニットには、
・デジタル入出力 16/16 点
・デジタル入出力 32/32 点
のユニットがあります。

デバイスドライバでは PORT アクセス以外の関数として、次の機能を使用することができます。

SPEED・RATE 関数

- ・加減速ドライブに必要な速度パラメータおよび第 1 パルス出力周期を 1Hz 単位で設定できます。
- ・加減速ドライブに必要な加減速時定数(ms/kHz)を RATE テーブル表から選択し設定できます。

ORIGIN ドライブ関数

弊社製チップコントローラ MCC05v2 の ORIGIN ドライブ相当の機械原点検出完了までの一連のシーケンスを実現します。

補間関数

- ・相対アドレスで指定された目的地まで、直線補間ドライブが実行できます。
- ・相対アドレスで指定された中心点と目的地で円弧補間ドライブが実行できます。
- ・絶対アドレスから相対アドレスに変換する関数により、絶対アドレスでの補間ドライブが実現できます。
- ・通過点、目的地から円の中心点を求める関数により、通過点と目的地による円弧補間ドライブが実現できます。

2 . 関数仕様

2-1. 一般仕様

項目	MPL-36-02v2.00/USBW32	MPL-37-02v2.00/USBW64
適用 OS *1	<ul style="list-style-type: none"> ・ Microsoft Windows 10 (x86) *2 ・ Microsoft Windows 8, 8.1 (x86) *2 ・ Microsoft Windows 7 (x86) ・ Microsoft Windows Vista (x86) ・ Microsoft Windows XP(x86) 	<ul style="list-style-type: none"> ・ Microsoft Windows 10 (x64) *2 ・ Microsoft Windows 8, 8.1 (x64) *2 ・ Microsoft Windows 7 (x64) ・ Microsoft Windows Vista (x64) ・ Microsoft Windows XP (x64)
適用言語 *1	<ul style="list-style-type: none"> ・ Visual C++ .NET 2002 ~ 2015 *3 ・ Visual C# .NET 2002 ~ 2015 ・ Visual Basic .NET 2002 ~ 2015 ・ Visual C++ 6.0 ・ Visual Basic 6.0 	<ul style="list-style-type: none"> ・ Visual C++ .NET 2005 ~ 2015 *3 ・ Visual C# .NET 2005 ~ 2015 ・ Visual Basic .NET 2005 ~ 2015
適用製品	(コントローラ) <ul style="list-style-type: none"> ・ UC-766 (MCC07 4 軸コントローラ:エンコーダ入力あり, 拡張ユニット対応) ・ UCD-7610v1/ADB5F30 (コントローラドライバ:MCC07 2 軸 5 相 0.75A/相, 拡張ユニット対応) ・ UCD-7613v1/GDB5F40 (コントローラドライバ:MCC07 2 軸 5 相 1.4A/相, 拡張ユニット対応) (拡張 I/O) <ul style="list-style-type: none"> ・ CB-52/3232 -MIL (拡張 I/O 32/32 点) ・ CB-53/1616 -MIL (拡張 I/O 16/16 点) 	
適用機種	<ul style="list-style-type: none"> ・ IBM PC/AT 互換機 	
使用可能台数	<ul style="list-style-type: none"> ・ 同時使用可能数 : 2 台 ・ 拡張ユニット対応の各スレーブユニットに拡張ユニット× 1 台 	
並列・並行処理	<ul style="list-style-type: none"> ・ マルチプロセス未対応 ・ マルチスレッド対応 	
割り込み	<ul style="list-style-type: none"> ・ 未対応 	

*1 : 各 OS および言語は、日本語版および英語版に対応しています。

*2 : Windows 8, Windows 8.1, Windows 10 環境では、デスクトップアプリのみに対応しています。
ストアアプリには対応していません。

*3 : アンマネージコードのみ対応です。

2-2. 互換性

(1) 32 ビット版と 64 ビット版との互換性

MPL-36-02v2.00/USBW32(x86:32 ビット版)と MPL-37-02v2.00/USBW64(x64:64 ビット版)は、ソースレベルで互換性があります。

- ・ MPL-36-02v2.00/USBW32 は、MPL-37-02v2.00/USBW64 に対してソースレベルで互換ですが、バイナリレベルでは非互換です。
MPL-37-02v2.00/USBW64 の関数定義ファイルで構築された 64 ビットユーザアプリケーションを、MPL-36-02v2.00/USBW32 がインストールされたパソコンで使用する場合、関数定義ファイルを MPL-36-02v2.00/USBW32 の関数定義ファイルに差し替えて、ユーザアプリケーションをビルドアップし直してください。
- ・ MPL-37-02v2.00/USBW64 は、MPL-36-02v2.00/USBW32 に対してバイナリレベルで互換です。
MPL-36-02v2.00/USBW32 の関数定義ファイルで構築された 32 ビットユーザアプリケーションを、MPL-37-02v2.00/USBW64 がインストールされたパソコンで使用する場合、WOW64 (Windows32 on Windows64) 機能により動作します。

(2) 従来製品との互換性

原則ソース互換ですが、バイナリ互換ではありません。
MPL-36-02v2.00/USBW32 または MPL-37-02v2.00/USBW64 に更新する際には、旧バージョンのデバイスドライバの関数定義ファイルを MPL-36-02v2.00/USBW32 または MPL-37-02v2.00/USBW64 の関数定義ファイルに差し替えて、ユーザアプリケーションをビルドアップし直してください。

・ MPL-36-02v2.00/USBW32 の旧バージョン

旧バージョンのデバイスドライバ	MPL-36-02v2.00/USBW32 への互換性
MPL-36v1.0/USBW32	ソース互換 *1 *2
MPL-36-01v1.00/USBW32	ソース互換 *1 *2
MPL-36-01v2.00/USBW32	ソース互換 *2
MPL-36-02v1.00/USBW32	バイナリ互換

・ MPL-37-02v2.00/USBW64 の旧バージョン

旧バージョンのデバイスドライバ	MPL-37-02v2.00/USBW64 への互換性
MPL-37-01v1.00/USBW64	ソース互換 *1 *2
MPL-37-01v2.00/USBW64	ソース互換 *2
MPL-37-02v1.00/USBW64	バイナリ互換

*1 C#で一部ソース互換でない関数

太枠線内の旧バージョンを使用している場合、キャスト処理などユーザアプリケーションの変更が必要です。

構造体名 / 関数名	メンバ / 引数 / 戻り値	MPL-36-02v2.00	MPL-36-01v1.00	MPL-36v1.0
		MPL-37-02v2.00	MPL-37-01v1.00	
データゲット関数	戻り値	uint 型	int 型	int 型

*2 一部ソース互換でない関数

SPEED ・ RATE 構造体には、メンバ SUH, SDH が追加されたため、従来デバイスドライバで SPEED ・ RATE 構造体を宣言時に初期化している場合、ユーザアプリケーションの変更が必要です。

MPL-36v1.0, MPL-36-01v1.00, MPL-36-01v2.00 MPL-37-01v1.00, MPL-37-01v2.00	⇒	MPL-36-02v1.00, MPL-36-02v2.00 MPL-37-02v1.00, MPL-37-02v2.00
MC07_S_SPEED_RATE sr = { 5000, 3000, 300, 0, 0, 0, 7, 7 };		MC07_S_SPEED_RATE sr = { 5000, 3000, 300, 0, 0, 0, <u>0</u> , 7, 7 };

*下線部分のメンバ SUH, SDH にダミー値(例では 0)を代入してください。

2-3. ソフト開発に必要なファイル**(1) MPL-36-02v2.00/USBW32 (32 ビット版)**

項目	ファイル
製品表示	¥Version.txt
Visual Basic.NET 関数定義ファイル	¥Bin¥x86¥Vb.Net 2002¥UsbC.vb ¥Bin¥x86¥Vb.Net 2005¥UsbC.vb (.NET 2002 ~) (.NET 2005 ~)
Visual C++.NET ヘッドファイル	¥Bin¥x86¥Vc¥UsbC.h
Visual C++ ライブラリファイル	¥Bin¥x86¥Vc¥VcMc07UsbC.lib
Visual C#.NET ヘッドファイル	¥Bin¥x86¥C#.Net¥UsbC.cs
Visual Basic 関数定義ファイル	¥Bin¥x86¥Vb¥UsbC.bas

(2) MPL-37-02v2.00/USBW64 (64 ビット版)

項目	ファイル
製品表示	¥Version.txt
Visual Basic.NET 関数定義ファイル	¥Bin¥x64¥Vb.Net¥UsbC.vb
Visual C++.NET ヘッドファイル	¥Bin¥x64¥Vc¥UsbC.h
Visual C++ ライブラリファイル	¥Bin¥x64¥Vc¥VcMc 07UsbC.lib
Visual C#.NET ヘッドファイル	¥Bin¥x64¥C#.Net¥UsbC.cs

3 . 関数の説明

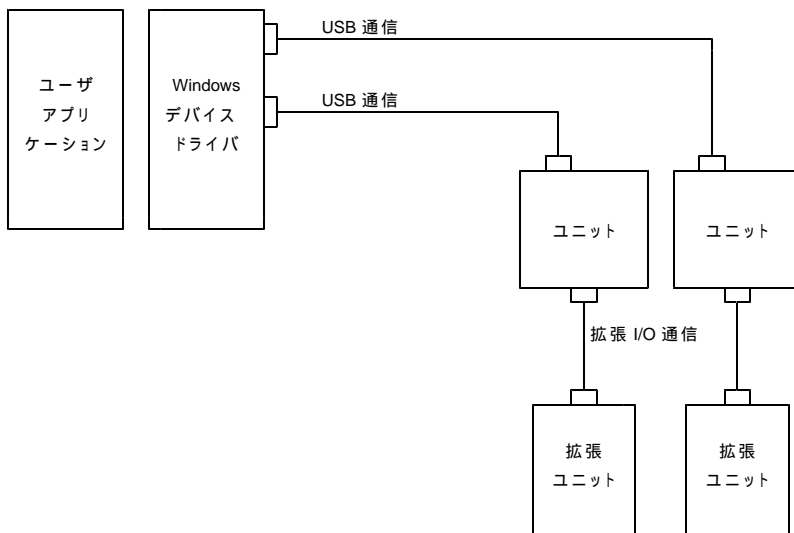
当デバイスドライバ仕様は、MCC07 および MCC09 の搭載製品ならびにデジタル I/O やアナログ入力を含む関数を網羅しています。

ユニットに MCC07 および MCC09 搭載製品が混在した構成も可能です。
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、本書では、これら対応製品の中で、MCC07 ユニット編として解説します。
MCC09 搭載製品仕様については、別冊の USB シリーズ「MCC09 ユニット編」の取扱説明書をご覧ください。

3-1. システムの構成

パソコンには、最大 2 台の USB シリーズユニットを実装することができます。
USB シリーズユニットには、1 個の拡張ユニットを接続することができます。



3-2. システムの制御

ユーザアプリケーションは、Windows 用デバイスドライバを使用し、次の制御を行うことができます。

- ・ USB シリーズユニットの制御
- ・ USB シリーズユニットを介した拡張ユニットの制御

(1) ユニットの制御

環境設定関数で環境設定を行うことにより、USB シリーズユニットの通信設定を行います。

次に、ユニット情報読み出し関数で、接続されている USB シリーズユニットの情報を確認します。

確認後、次の表に示す関数により、ユニットの制御を行います。

使用する関数	必要なハンドル	アクセスの対象
ユニット関数	ユニットハンドル (ユニットオープン関数)	・ 拡張ユニットとの通信設定など ・ 1つのユニットの複数軸、複数の I/O PORT
デバイス関数	デバイスハンドル (デバイスオープン関数)	1 軸
I/O PORT 関数	PORT ハンドル (I/O PORT オープン関数)	1 つの I/O PORT

(2) ユニットの介した拡張ユニットの制御

ユニットオープン関数により USB シリーズユニットをオープン後、拡張ユニット通信設定関数、拡張ユニット通信制御関数により、ユニットと拡張ユニット間の通信の設定および制御を行います。

拡張ユニット通信ステータス読み出し関数で、ユニットと拡張ユニットが接続されている状態であることを確認し、次の表に示す関数により、拡張ユニットの制御を行います。

使用する関数	必要なハンドル	アクセスの対象
ユニット関数	ユニットハンドル (ユニットオープン関数)	1つの拡張ユニットの複数の I/O PORT (軸や他の I/O PORT との同時アクセスも可)
I/O PORT 関数	PORT ハンドル (I/O PORT オープン関数)	1つの拡張ユニットの 1 つの I/O PORT

3-3. 関数体系

デバイスドライバの関数は、主にシステム関数、ユニット関数、デバイス関数、I/O PORT 関数に分類されます。ユニット関数とデバイス関数、I/O PORT 関数は、各オープン後に関数を併用して実行することができます。

(1) システム関数

システム関数は、デバイスドライバで認識されている全ての USB シリーズユニットを対象に、環境設定/接続確認などの設定を行うための関数群です。

システム関数は次のように分類されます。

分類	説明
環境設定/接続確認関数	環境設定の実行とユニットの接続情報の読み出し

(2) ユニット関数

ユニット関数は、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、USB シリーズユニットの制御を行うための関数群です。

ユニット関数は次のように分類されます。

分類	説明
オープン/クローズ関数	ユニットのオープンやクローズ
エラークリア関数	ユニットの動作エラーのクリア
拡張ユニット通信関数	拡張ユニットとの通信設定
A/D 変換関数	A/D 変換されたアナログ入力信号の読み出し
ユニットアクセス関数	MCC07 複数軸の PORT および複数の I/O PORT の読み出しと書き込み 複数の I/O PORT の読み出しと書き込み

(3) デバイス関数

MCC07 の 1 軸をデバイスと呼称します。

デバイス関数は、デバイスオープン関数で取得したデバイスハンドルにより、デバイスの制御を行うための関数群です。デバイス関数は次のように分類されます。

分類	説明
オープン/クローズ関数	デバイスのオープンやクローズ
エラークリア関数	デバイスの動作エラーのクリア
MCC PORT アクセス関数	MCC07 PORT の読み出しと書き込み
WAIT 関数	デバイスの BUSY=0 または COMREG FULL=0 を待機
SPEED・RATE 関数	SPEED パラメータと加減速時定数の設定
補間ドライブ関数	補間ドライブの演算と制御
ORIGIN ドライブ関数	ORIGIN ドライブの制御

(4) I/O PORT 関数

I/O PORT 関数は、I/O PORT オープン関数で取得した PORT ハンドルにより、I/O PORT の制御を行うための関数群です。

I/O PORT 関数は次のように分類されます。

分類	説明
オープン/クローズ関数	I/O PORT のオープンやクローズ
I/O PORT アクセス関数	I/O PORT の読み出しと書き込み

3-4. エラー

エラーには、関数エラーと動作エラーがあります。

(1) 関数エラー

関数実行時に発生するエラーです。関数エラーには通信エラーとその他のエラーがあります。

通信エラー

項目	説明
エラー内容	USB 通信に失敗したときに発生するエラーです。
検出方法	関数がエラー終了し、RESULT 構造体のメンバ MC07_Result [1] に要因を通知します。
インターロック	環境設定関数以外の関数を禁止します。
エラークリア	環境設定関数の実行でクリアします。 *1

*1 RESULT 構造体のメンバ MC07_Result [1] が、100 の関数エラーが発生したときは、動作エラークリア関数でエラーをクリアすることはできません。
このエラーが発生したときは、必ず環境設定関数を実行してください。

その他のエラー

項目	説明
エラー内容	パラメータの異常や、他の様々な要因が発生したエラーです。
検出方法	関数がエラー終了し、RESULT 構造体のメンバ MC07_Result [1] に要因を通知します。
インターロック	- (インターロックはされません)
エラークリア	-

(2) 動作エラー

関数実行により開始された動作の動作中に発生するエラーです。動作エラーには ORIGIN エラーと MCC07 エラーがあります。

ORIGIN ドライブのエラー

項目	説明
エラー内容	ORIGIN ドライブ関数による ORIGIN ドライブ中に発生したエラーです。
検出方法	ORIGIN STATUS に ERROR=1 および ERROR 要因を通知します。
インターロック	次の処理が実行できなくなります。 <ul style="list-style-type: none"> ・ MCC07 ドライブコマンドの汎用コマンドの実行 ・ SPEED・RATE セット関数の実行 ・ メインチップ 2 軸相対アドレス直線補間ドライブ関数の実行 ・ メインチップ 2 軸相対アドレス円弧補間ドライブ関数の実行 ・ 2 軸相対アドレス直線補間ドライブ関数の実行 *2 ・ 2 軸相対アドレス円弧補間ドライブ関数の実行 *2 ・ ORIGIN ドライブ関数の実行
エラークリア	動作エラークリア関数またはユニット動作エラークリア関数の実行でクリアします。 *1

MCC07 のエラー

項目	説明
エラー内容	MCC07 で発生したエラーです。
検出方法	DRIVE STATUS1 PORT に ERROR=1 通知します。 MCC の ERROR STATUS READ COMMAND で ERROR 要因を確認することができます。
インターロック	次の処理が実行できなくなります。 <ul style="list-style-type: none"> ・ MCC07 ドライブコマンドの汎用コマンドの実行 ・ SPEED・RATE セット関数の実行 ・ メインチップ 2 軸相対アドレス直線補間ドライブ関数の実行 ・ メインチップ 2 軸相対アドレス円弧補間ドライブ関数の実行 ・ 2 軸相対アドレス直線補間ドライブ関数の実行 *2 ・ 2 軸相対アドレス円弧補間ドライブ関数の実行 *2 ・ ORIGIN ドライブ関数の実行
エラークリア	動作エラークリア関数またはユニット動作エラークリア関数の実行でクリアします。 *1

*1 動作エラークリア関数を実行したときに、エラー要因が継続しているときはエラーを再検出します。
動作エラークリア関数を実行する前に MCC07 の ERROR STATUS READ コマンドにて ERROR STATUS を読み出し、エラー要因の特定、およびエラーを取り除いてから動作エラークリア関数を実行してください。

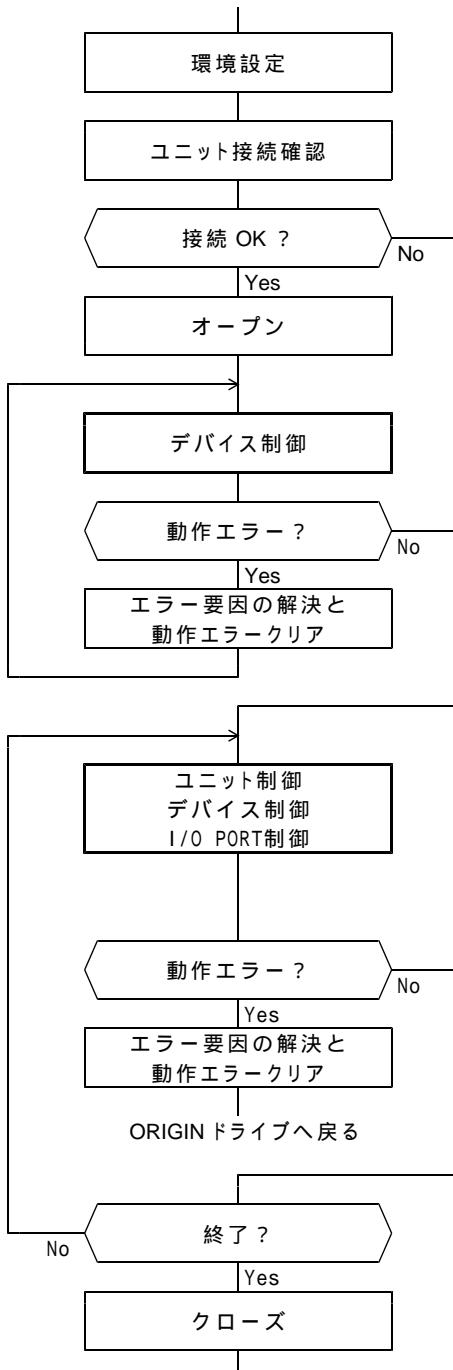
*2 この関数は、MCC07 用として新たに追加されました。
任意 2 軸が可能な補間関数です。

3-5. シーケンス

全体シーケンス、ユニット制御シーケンス、デバイス制御シーケンス、I/O PORT 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) 全体シーケンス

ユーザアプリケーションの開始から終了までの全体のシーケンスを示します。
デバイス制御による ORIGIN ドライブで機械原点を検出後、ユニット制御 / デバイス制御 / I/O PORT 制御によりメインの制御を行います。



初期化処理
環境設定関数で環境設定を行います。

ユニット情報読み出し関数で、ユニットの接続状態を確認します。

ユニット / デバイス / I/O PORT オープン関数で、ユニット / デバイス / I/O PORT をオープンします。

ORIGIN ドライブ
デバイス関数を使用して、ORIGIN ドライブを行います。詳細は、デバイス制御シーケンスを参照して下さい。

動作エラーが発生した場合、エラー要因を確認し、必要な場合はエラー要因を解決します。その後、動作エラークリア関数、又は、ユニット動作エラークリア関数で動作エラーをクリアします。

メインの制御
ユニット関数 / デバイス関数 / I/O PORT 関数を使用して、ユニット / デバイス / I/O PORT を制御します。詳細は、ユニット制御シーケンス、デバイス制御シーケンス、I/O PORT 制御シーケンスを参照して下さい。

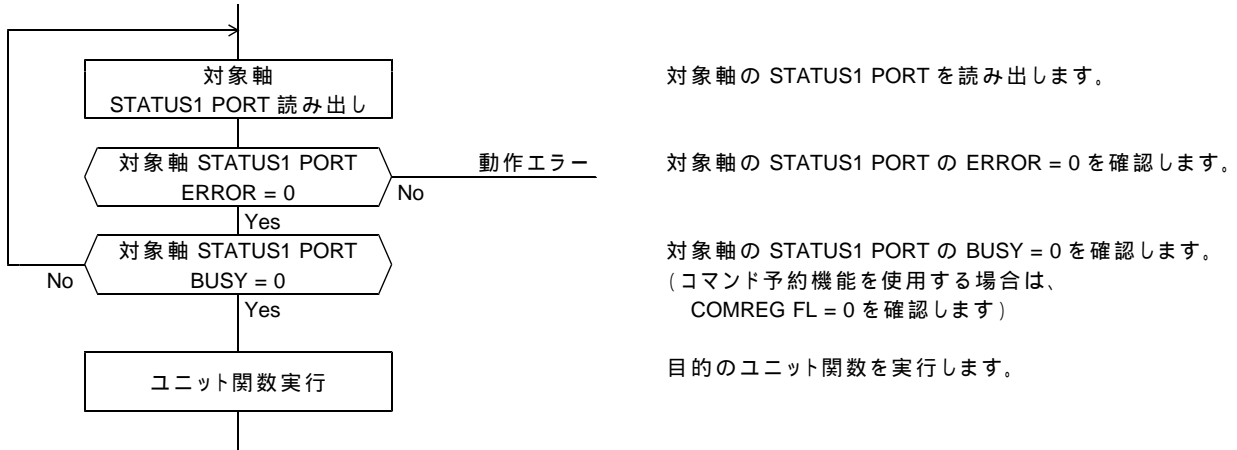
動作エラーが発生した場合、エラー要因を確認し、必要な場合はエラー要因を解決します。その後、動作エラークリア関数、又は、ユニット動作エラークリア関数で動作エラーをクリアします。

終了処理
ユニット / デバイス / I/O PORT をクローズします。

(2) ユニット制御シーケンス

ユニットアクセス関数でユニットを制御するためのシーケンスを示します。
ユニットアクセス関数で次の処理をする場合、該当軸の DRIVE STATUS1 PORT の ERROR = 0 と BUSY = 0 の確認が必要です。

- ・ MCC07 のドライブコマンドの汎用コマンドを実行



次の処理をする場合、対象軸の DRIVE STATUS1 PORT の ERROR = 0 と BUSY = 0 の確認は不要です。

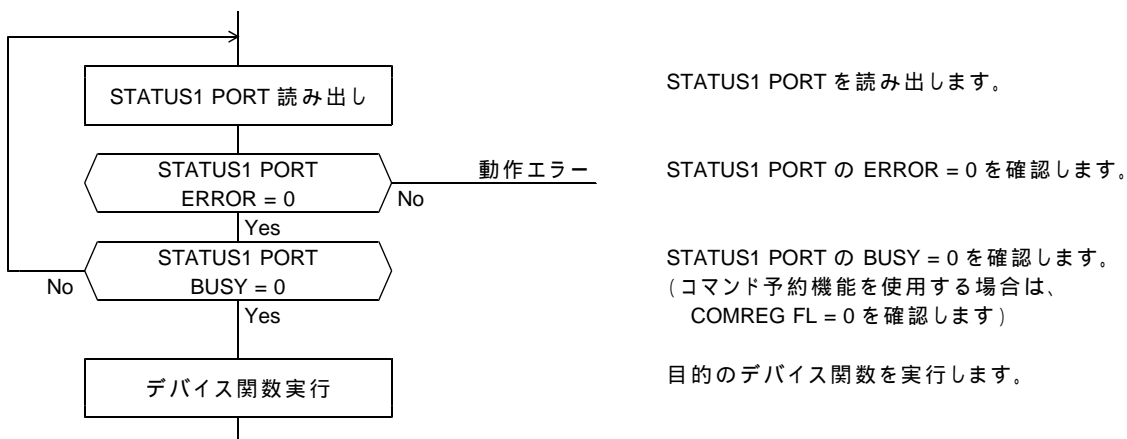
- ・ MCC07 のドライブコマンドの特殊コマンドを実行



(3) デバイス制御シーケンス

デバイス関数でデバイスを制御するためのシーケンスを示します。
デバイス関数で次の処理をする場合、DRIVE STATUS1 PORT の ERROR = 0 と BUSY = 0 の確認が必要です。

- ・ MCC07 のドライブコマンドの汎用コマンドを実行
- ・ SPEED ・ RATE セット関数の実行
- ・ メインチップ 2 軸相対アドレス直線補間ドライブ関数の実行
- ・ メインチップ 2 軸相対アドレス円弧補間ドライブ関数の実行
- ・ 2 軸相対アドレス直線補間ドライブ関数の実行(4 軸製品のみ)
- ・ 2 軸相対アドレス円弧補間ドライブ関数の実行(4 軸製品のみ)
- ・ ORIGIN ドライブ関数の実行



DRIVE STATUS1 PORT の読み出しには、以下のいずれかの関数を使用します。

- ・ DRIVE STATUS1 PORT 読み出し関数
- ・ READY WAIT 関数 (READY WAIT 関数内では STATUS1 PORT を読み出しています)

ORIGIN STATUS の読み出しには、ORIGIN STATUS 読み出し関数を使用します。

次の処理をする場合、対象軸の DRIVE STATUS1 PORT の ERROR = 0 と BUSY = 0 の確認は不要です。

- ・ MCC07 のドライブコマンドの特殊コマンドを実行



(4) I/O PORT 制御シーケンス

I/O PORT 関数で I/O PORT を制御するためのシーケンスを示します。



目的の I/O PORT 関数を実行します

(5) MCC07 の実行シーケンス

MCC07 実行シーケンスの詳細については、5章「コマンド仕様」をご覧ください。

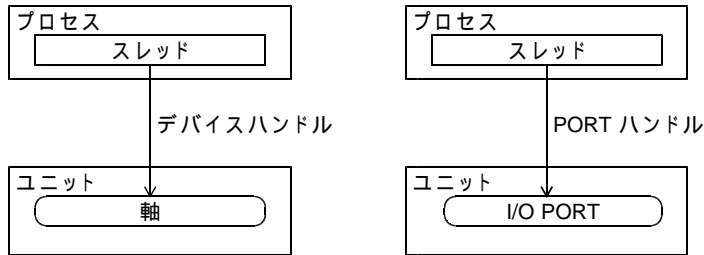
3-6. 並列・並行処理

ユニット上の軸や I/O PORT に複数のスレッドからアクセスする際、デバイスドライバの関数をアトミックに実行するために、ユーザアプリケーションによる排他制御が必要な場合と不要な場合があります。以下に詳細を示します。

(1) マルチプロセス未対応

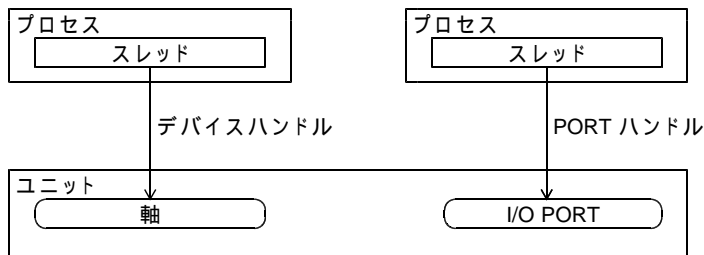
1プロセス / 1ユニット

異なるプロセスのスレッドが、それぞれ異なるユニットに属する軸や I/O PORT にアクセスすることはできません。



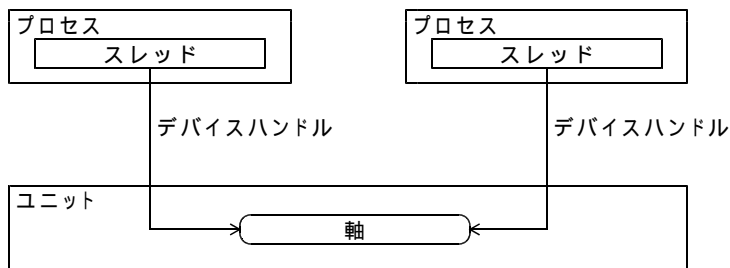
1プロセス / 1軸

異なるプロセスのスレッドが、同一のユニット上のそれぞれ異なる軸や I/O PORT にアクセスすることはできません。



Nプロセス / 1軸

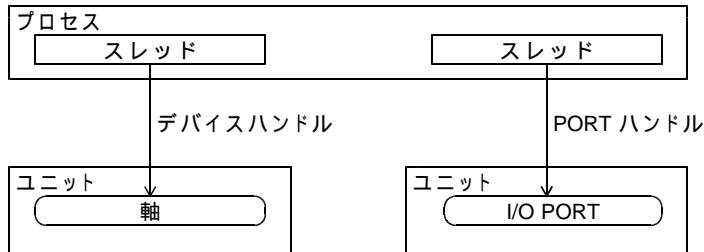
異なるプロセスのスレッドが、同一の軸や I/O PORT にアクセスすることはできません。



(2) マルチスレッド対応

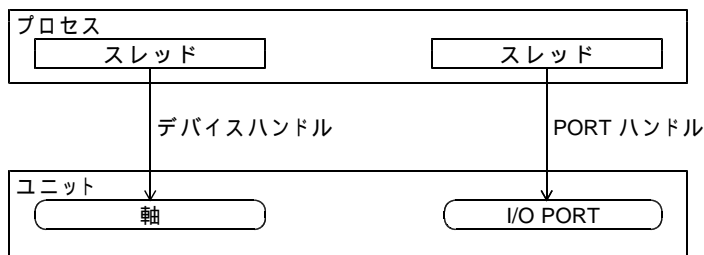
1 スレッド / 1 ユニット

同一のプロセスの異なるスレッドが、それぞれ異なるユニットに属する軸や I/O PORT アクセスする場合、排他制御は不要です。



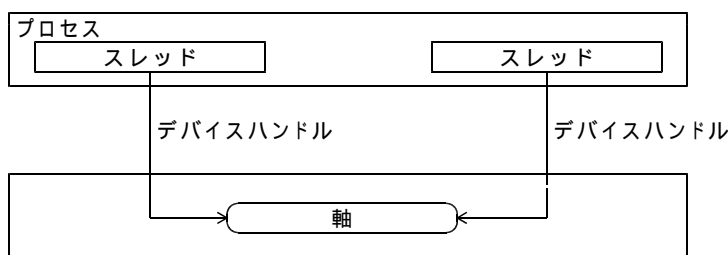
1 スレッド / 1 軸

同一のプロセスの異なるスレッドが、同一のユニット上のそれぞれ異なる軸や I/O PORT にアクセスする場合、排他制御は不要です。



N スレッド / 1 軸

同一のプロセスの異なるスレッドが、同一の軸や I/O PORT にアクセスする場合、排他制御は不要です。ただし、SPEED・RATE 関数、補間関数、ORIGIN ドライブ関数を実行中は、他のスレッドが同じ軸にアクセスしないように排他制御が必要です。



(注 1) システム関数を実行中は、他のスレッドでデバイスドライバの一切の関数を実行することはできません。実行した場合の動作は不定です。

(注 2) READY WAIT 関数、COMREG NOT FULL WAIT 関数を実行中、同じデバイスに対して再度 READY WAIT 関数、COMREG NOT FULL WAIT 関数を実行することはできません。

(注 3) 1つのデバイス / I/O PORT に対しては、オープン アクセス クローズの流れが守られるようにしてください。

3-7. 制限事項

(1) MCC07 コマンドの制限

次のコマンドを実行しても無効です。

- ・ ERRINT STATUS MASK コマンド
- ・ INT FACTOR MASK コマンド

次の設定は禁止されています。

- ・ SPEC INITIALIZE2 コマンドの RDYINT TYPE を「3:出力しない(常時 RDYINT = 0)」以外に設定することはできません。設定した場合、「3:出力しない(常時 RDYINT = 0)」に補正されます。
- ・ HARD INITIALIZE8 コマンドの INT ACTIVE を 0 に設定することはできません。設定した場合、1 に補正されます。

(2) ORIGIN ドライブに関連する制限

次の設定がされている場合、ORIGIN ドライブを実行できません。

- ・ SPEC INITIALIZE2 コマンド
 - ・ CWLM TYPE : 即時停止信号として使用する
 - ・ CCWLM TYPE : 即時停止信号として使用する
 - ・ SS0 TYPE : 減速停止信号として使用する、または、即時停止信号として使用する
- ・ SPEC INITIALIZE3 コマンド
 - ・ DEND/PO TYPE : 減速停止信号として使用する、または、即時停止信号として使用する

ORIGIN ドライブ中は、次のコマンドのみ実行可能です。

- ・ ADDRESS COUNTER READ コマンド
- ・ ADDRESS LATCH DATA READ コマンド
- ・ DFL COUNTER READ コマンド
- ・ DFL LATCH DATA READ コマンド
- ・ MCC SPEED READ コマンド
- ・ SLOW STOP コマンド
- ・ FAST STOP コマンド

ORIGIN ドライブ中、各種カウンタのコンパレータの一致による停止機能は、無効になります。

ORIGIN ドライブ中、ユーザアプリケーションが設定した ERROR STATUS MASK は、無効になります。

ORIGIN ドライブ中、パルスカウンタは、使用できません。

ORIGIN ドライブ中、および ORIGIN ドライブ終了時は、次のステータスのみ有効です。

- ・ STATUS1 PORT の BUSY
- ・ ORIGIN STATUS
- ・ STATUS2 PORT の DEND BUSY, DALM, DEND/PO, DRST, NORG, ZORG, ORG, CWLM, CCWLM, FSSTOP

(3) READ コマンドに関連する制限

MCC07 の READ コマンドを実行する場合、次の書き込み / 読み出し関数を使用すると、1 回の関数実行で済みます。

- ・ DRIVE COMMAND 32 ビット書き込み / 読み出し関数 (MC07_LWRDrive 関数)

READ コマンド

コマンドコード	特殊コマンド名称	機能
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	設定データの読み出し
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

書き込み / 読み出し関数以外の関数で READ コマンドを実行する場合、書き込み関数と読み出し関数を組み合わせる必要があります。

- ・ DRIVE COMMAND 32 ビット書き込み関数 + DRIVE DATA 32 ビット読み出し関数
- ・ DRIVE COMMAND PORT 書き込み関数 + DRIVE DATA 32 ビット読み出し関数
- ・ その他

この場合、書き込み関数を実行してから、読み出し関数を実行するまでの間は、同じユニットに対してデバイスドライバの関数を実行しないようにしてください。

4 . 関数リファレンス

4-1. 構造体・関数の見方

(1) 構造体

構造体	構造体の名称	関数定義ファイル名
<input type="text"/>	構造体に対応するユニットの名称	
説明		
.....	構造体の説明	
書式		
C言語	C言語(Visual C++およびVisual C++.NET)で構造体を使用するときの定義
VB	Visual Basicで構造体を使用するときの定義
VB.NET	Visual Basic.NETで構造体を使用するときの定義
C#.NET	Visual C#.NETで構造体を使用するときの定義
メンバ		
.....	構造体のメンバに格納される値の説明	

(2) 関数

関数	関数の名称	関数定義ファイル名
<input type="text"/>	関数に対応するユニットの名称	
機能		
.....	関数の機能の説明	
書式		
C言語	C言語で、関数を使用するときの定義
VB	Visual Basicで関数を使用するときの定義
VB.NET	Visual Basic.NETで関数を使用するときの定義
C#.NET	C#.NETで構造体を使用するときの定義
引数		
.....	関数の各引数に指定する値の説明	
戻り値		
.....	関数の戻り値の説明	

関数に対応するユニット名称一覧表

名称	品名	定格	備考
UC-766	コントローラ	UC-766	4軸コントローラ(MCC07搭載, ユニット関数対応, エンコーダ入力なし)
UCD-7610v1	コントローラドライバ	UCD-7610v1/ADB5F30	5相2軸コントローラドライバ:0.75A/相(MCC07搭載, ユニット関数対応)
UCD-7613v1	コントローラドライバ	UCD-7613v1/GDB5F40	5相2軸コントローラドライバ:1.4A/相(MCC07搭載, ユニット関数対応)
CB-52	拡張 I/O	CB-52/3232-MIL	拡張タイプの I/O 32/32 点
CB-53	拡張 I/O	CB-53/1616-MIL	拡張タイプの I/O 16/16 点

* 当デバイスドライバは、MCC09 搭載製品にも対応していますが、仕様の差異を明らかにするため、MCC07 搭載製品を代表にして仕様を表記しています。

(3) 言語固有の仕様

RESULT 構造体の NULL ポインタ

Visual C++、Visual C++.NET では、当関数を実行する際、psResult (RESULT 構造体のポインタ) に NULL ポインタを指定することができます。
NULL ポインタを指定すると、実行結果は格納されません。

ブール型の扱い

当デバイスドライバの多くの関数は、戻り値としてブール型の値を返します。
ブール型の戻り値は以下になります。

言語	型	ブール型の戻り値	
		TRUE (真)	FALSE (偽)
Visual C++ Visual C++.NET	BOOL	TRUE	FALSE
Visual Basic	Boolean	1	0
Visual Basic.NET	Boolean	True	False
Visual C#.NET	bool	true	false

詳細は、各関数の書式をご覧ください。

Visual Basic.NET

構造体の初期化

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。

構造体		Initializeメソッドの書式
RESULT構造体	MC07_S_RESULT	Public Sub Initialize()
ユニット情報構造体	MC07_S_UNIT_INFO	
USB情報構造体	MC07_S_USB_INFO	
USB情報リスト構造体	MC07_S_USB_INFO_LIST	
A/Dデータ構造体	MC07_S_AD_DATA	
コマンドバッファ構造体	MC07_S_COMMAND_BUF	
データバッファ構造体	MC07_S_DATA_BUF	
ステータスバッファ構造体	MC07_S_STATUS_BUF	
データ構造体	MC07_S_DATA	

Visual C#.NET

構造体の初期化

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。

構造体		コンストラクタの書式
RESULT構造体	MC07_S_RESULT	Public MC07_S_RESULT(ushort dummy);
ユニット情報構造体	MC07_S_UNIT_INFO	Public MC07_S_UNIT_INFO(ushort dummy);
USB情報構造体	MC07_S_USB_INFO	Public MC07_S_USB_INFO(ushort dummy);
USB情報リスト構造体	MC07_S_USB_INFO_LIST	Public MC07_S_USB_INFO_LIST(ushort dummy);
A/Dデータ構造体	MC07_S_AD_DATA	Public MC07_S_AD_DATA(ushort dummy);
コマンドバッファ構造体	MC07_S_COMMAND_BUF	Public MC07_S_COMMAND_BUF(ushort dummy);
データバッファ構造体	MC07_S_DATA_BUF	Public MC07_S_DATA_BUF(unit dummy);
ステータスバッファ構造体	MC07_S_STATUS_BUF	Public MC07_S_STATUS_BUF(ushort dummy);
データ構造体	MC07_S_DATA	Public MC07_S_DATA(ushort dummy);

*コンストラクタの引数dummyは何を指定しても無効です。

名前空間

利用できる構造体、関数は、名前空間 MELEC にあります。

定数

本リファレンスに示される定数 MC07_XXXX は、全て MC07.MC07_XXXX のように指定します。

(例)

リファレンスに示す定数	Visual C#.NETでの指定
MC07_USB_UNIT_0	MC07.MC07_USB_UNIT_0
MC07_SEL_X	MC07.MC07_SEL_X
MC07_GP_IN	MC07.MC07_GP_IN

4-2. 基本構造体 (共通)

4-2-1. RESULT 構造体

RESULT 構造体

UsbC

UC-766	UCD-7610v1	UCD-7613v1
CB-52	CB-53	

説明

関数を実行した結果が格納されます。

書式

C言語 typedef struct MC07_TAG_S_RESULT {
 WORD *MC07_Result*[4];
 } MC07_S_RESULT;

VB Type MC07_S_RESULT
 MC07_Result(1 To 4) As Integer
 End Type

VB.NET Structure MC07_S_RESULT
 <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> Public MC07_Result() As Short
 Public Sub Initialize()
 ReDim *MC07_Result*(3)
 End Sub
 End Structure

C#.NET struct MC07_S_RESULT
 {
 [MarshalAs(UnmanagedType.ByValArray, SizeConst=4)] public ushort[] *MC07_Result*;
 public MC07_S_RESULT(ushort dummy)
 {
 MC07_Result = new ushort[4];
 }
 }

メンバ

MC07_Result[0] ... 0が読み出されます。

MC07_Result[1] ... 実行結果を示します。(値は10進表記です。)

値	実行結果
0	関数の実行が正常に終了しました。
2	DLL内部でAPIエラーが発生しました。
3	NULLポインタが指定されました。
5	ユニット番号の指定に誤りがあります。
6	軸またはI/O PORTの指定に誤りがあります。
7	指定されたハンドルが不正です。
8	デバイスハンドルで示された軸ではX軸(またはZ軸)座標アドレスの補間パルスを出力できません。
10	デバイスハンドルで示された軸ではY軸(またはA軸)座標アドレスの補間パルスを出力できません。
11	指定されたユニット、デバイスまたはI/O PORTはオープンされていません。
12	2つのデバイスハンドルで示された軸が同一チップ上にありません。
13	指定されたユニット、デバイスまたはI/O PORTはすでにオープンされています。
15	WAIT関数がTIME OVERで終了しています。 ・READY WAIT関数がTIME OVERで終了しています。 ・COMREG NOT FULL WAIT関数がTIME OVERで終了しています。
16	WM_QUITメッセージを受信しました。
17	WAIT関数実行中にWAIT中止関数が実行されました。 ・READY WAIT中にWAIT中止関数が実行されました。 ・COMREG NOT FULL WAIT中にWAIT中止関数が実行されました。

値	実行結果
18	WAIT関数が複数同時に実行されました。 ・同一デバイスのREADY WAIT関数が複数同時に実行されました。 ・同一デバイスのCOMREG NOT FULL WAIT関数が複数同時に実行されました。
21	指定されたユニット番号に該当するユニットがありません。
22	ユニット番号が重複しています。
24	2つのデバイスハンドルで指定された軸が同一軸です。
25	2つのデバイスハンドルで指定された軸が同一ユニット上にありません。
28	2つのデバイスハンドルで指定された軸が同一チップ上にあります。
30	動作エラーが発生しているため、関数をエラー終了しました。 ・DRIVE STATUS1 PORTのERROR = 1のため、関数をエラー終了しました
34	実行しようとした関数は指定されたユニットでは実行できません。
38	軸またはI/O PORTに指定された操作をすることができません。
45	ADDRESS COUNTERがオーバーフローしているため関数が実行できません。
50	中心点、通過点または目的地の座標が指定できる範囲を越えています。
51	ドライブ方向の指定に誤りがあります。
52	ORG TYPEの指定に誤りがあります。
53	RESOL No.が設定範囲を越えています。
54	SPEC INITIALIZE2 COMMANDまたはSPEC INITIALIZE3 COMMANDを使用して、ORIGINドライブで許可されていない設定が行われています。
60	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です。
61	円弧補間で求めた短軸PULSE数が-2,147,483,648~+2,147,483,647の範囲内ではありません。
62	通過点相対アドレスまたは目的地相対アドレスが(0, 0)です。または通過点と目的地が同一です。
63	現在位置、通過点、目的地が直線上にあります。
64	現在位置から中心点までの相対アドレスが-8,388,608~8,388,607の範囲を越えています。
72	指定されたコマンドの個数が設定範囲を越えています。
73	既に他のプロセスで環境設定されています。
74	未だ現在のプロセスで環境設定されていません。
90	拡張ユニットまたは拡張G1/Oユニットとの通信の通信レートの指定に誤りがあります。
91	拡張ユニットまたは拡張G1/Oユニットとの通信のリトライ回数の指定に誤りがあります。
92	拡張ユニットまたは拡張G1/Oユニットとの通信のI/O点数の指定に誤りがあります。
93	拡張ユニットまたは拡張G1/Oユニットとの通信の制御の指定に誤りがあります。
94	信号指定に誤りがあります。
100	USB通信時にエラーが発生しました。

MC07_Result[2] ... 0が読み出されます。

MC07_Result[3] ... 将来の拡張用です。

- ・VBの*MC07_Result(1)~(4)*は、C言語の*MC07_Result[0]~[3]*に対応します。
- ・VB.NETの*MC07_Result(0)~(3)*は、C言語の*MC07_Result[0]~[3]*に対応します。
- ・C#.NETの*MC07_Result[0]~[3]*は、C言語の*MC07_Result[0]~[3]*に対応します。

4-2-2. コマンドデータ構造体

コマンドデータ構造体

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

DRIVE COMMAND PORT、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_MC07_S_COMMAND_DATA{
 WORD Command;
 DWORD Data;
} MC07_S_COMMAND_DATA

VB Type MC07_S_COMMAND_DATA
 Command As Integer
 Data As Long
End Type

VB.NET Structure MC07_S_COMMAND_DATA
 Public Command As Short
 Public Data As Integer
End Structure

C#.NET struct MC07_S_COMMAND_DATA
{
 public ushort Command;
 public uint Data;
}

メンバ

- Command* ... DRIVE COMMAND PORTに書き込む内容を格納します。
- Data* ... DRIVE DATA1 PORT、DRIVE DATA2 PORTに書き込む内容を格納します。
- ・ 上位16ビットはDRIVE DATA2 PORTに書き込まれます。
 - ・ 下位16ビットはDRIVE DATA1 PORTに書き込まれます。

4-2-3. ステータスデータ構造体

ステータスデータ構造体

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

機能

DRIVE STATUS1 PORT、DRIVE DATA1 PORT、DRIVE DATA2 PORTから読み出した内容を格納します。

書式

C言語 typedef struct _MC07_S_STATUS_DATA {
 WORD *Status1*;
 DWORD *Data*;
 } MC07_S_STATUS_DATA;

VB Type MC07_S_STATUS_DATA
 Status1 As Integer
 Data As Long
 End Type

VB.NET Structure MC07_S_STATUS_DATA
 Public *Status1* As Short
 Public *Data* As Integer
 End Structure

C#.NET struct MC07_S_STATUS_DATA
 {
 public ushort *Status1*;
 public uint *Data*;
 }

メンバ

Status1 ... STATUS1 PORTから読み出した内容を格納します。
Data ... DRIVE DATA1 PORT、DRIVE DATA2 PORTから読み出した内容を格納します。
 ・DRIVE DATA2 PORTの内容が上位16ビットに格納されます。
 ・DRIVE DATA1 PORTの内容が下位16ビットに格納されます。

4-3. システム関数

環境設定関数で環境設定を実行し、ユニット情報読み出し関数でユニットの接続情報を読み出します。USB シリーズのユニットは、環境設定関数を受け付けると、以降の関数に応答ようになります。従って、ユーザアプリケーションの最初で必ず環境設定関数を実行する必要があります。

4-3-1. ユニット情報構造体

ユニット情報構造体

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

機能

パソコンに接続されている全てのユニットのタイプを格納します。

書式

C言語 `typedef struct_MC07_S_UNIT_INFO {
 WORD UnitType[10];
 } MC07_S_SLAVE_INFO;`

VB `Type MC07_S_UNIT_INFO
 UnitType(1 To 10) As Integer
 End Type`

VB.NET `Structure MC07_S_UNIT_INFO
 <MarshalAs(UnmanagedType.ByValArray, SizeConst:=10)> Public UnitType() As Short
 Public Sub Initialize()
 ReDim UnitType(9)
 End Sub
 End Structure`

C#.NET `struct MC07_S_SLAVE_INFO
 {
 [MarshalAs(UnmanagedType.ByValArray, SizeConst = 10)]
 public ushort[] UnitType;
 public MC07_S_SLAVE_INFO(ushort dummy)
 {
 UnitType = new ushort[10];
 }
 }`

メンバ

UnitType[0] … ユニット番号0に接続されるユニットのタイプが格納されます。

UnitType[1] … ユニット番号1に接続されるユニットのタイプが格納されます。

UnitType[2] … 将来の拡張用です。

⋮

UnitType[9] … 将来の拡張用です。

格納される値	意味
H'10	UC-766
H'30	UCD-7610v1
H'31	UCD-7613v1
H'FF	未接続

- VBの *UnitType(1) ~ (10)* は、C言語の *UnitType [0] ~ [9]* に対応します。
- VB.NETの *UnitType(0) ~ (9)* は、C言語の *UnitType [0] ~ [9]* に対応します。
- C#.NETの *UnitType [0] ~ [9]* は、C言語の *UnitType [0] ~ [9]* に対応します。

4-3-2. 環境設定 / 接続確認関数

環境設定関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

機能

USBに接続される全ユニットを対象に環境設定を行います。

書式

C言語 `BOOL MC07_Environment(WORD UnitNo,
 MC07_S_RESULT *psResult);`

VB `Function MC07_Environment(ByVal UnitNo As Integer,
 ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_Environment(ByVal UnitNo As Short,
 ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.Environment(ushort UnitNo,
 ref MC07_S_RESULT psResult);`

引数

UnitNo … 無条件にMC07_USB_UNIT_0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。

ユニット情報読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

USBに接続される全ユニットのタイプを読み出します。

書式

C言語 `BOOL MC07_ReadUnitInfo(WORD UnitNo, MC07_S_UNIT_INFO *psUnitInfo, MC07_S_RESULT *psResult);`

VB `Function MC07_ReadUnitInfo(ByVal UnitNo As Integer, ByRef psUnitInfo As MC07_S_UNIT_INFO, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_ReadUnitInfo(ByVal UnitNo As Short, ByRef psUnitInfo As MC07_S_UNIT_INFO, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.ReadUnitInfo(ushort UnitNo, ref MC07_S_UNIT_INFO psUnitInfo, ref MC07_S_RESULT psResult);`

引数

UnitNo ... 無条件にMC07_USB_UNIT_0を指定します。
psUnitInfo ... ユニットのタイプを格納するためのユニット情報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。

4-4. ユニット関数

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、ユニットの制御を行います。

- ・当関数の実行は、ユニット単位のアクセス、および拡張ユニットの通信設定、通信制御、ステータス読み出しを行うときに使用します。
 - ・拡張I/Oは、ユニットオープンした後、拡張ユニット通信の設定および通信を開始した後は、I/Oオープンして拡張I/Oをコントロールします。
- ユニットオープンしている間は拡張ユニットのステータスを読み出すことで接続状態を監視することができます。

4-4-1. ユニットオープン/ユニットクローズ関数

ユーザアプリケーションは、ユニットをオープンして、ユニットハンドルを受け取ります。

以降、ユニット関数を実行する際にこのユニットハンドルを引数として渡します。

このユニットハンドルは、ユニットをクローズするまで有効です。

ユーザアプリケーション終了時は、必ずユニットをクローズしてください。

ユニットオープン関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたユニットをオープンし、引数 *phUnit* で示される変数にユニットハンドルを格納します。

書式

C言語 `BOOL MC07_UOpen(WORD UnitNo, DWORD *phUnit, MC07_S_RESULT *psResult);`

VB `Function MC07_UOpen(ByVal UnitNo As Integer, ByRef phUnit As Long, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_UOpen(ByVal UnitNo As Short, ByRef phUnit As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.UOpen(ushort UnitNo, ref uint phUnit, ref MC07_S_RESULT psResult);`

引数

UnitNo …… ユニット番号を指定します。

指定できる値	意味
MC07_USB_UNIT_0	ユニット番号0
MC07_USB_UNIT_1	ユニット番号1

phUnit …… ユニットハンドル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。

ユニットクローズ関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

機 能

指定されたユニットをクローズします。

書 式

C言語 `BOOL MC07_UClose(DWORD hUnit, MC07_S_RESULT *psResult);`

VB `Function MC07_UClose(ByVal hUnit As Long,
 ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_UClose(ByVal hUnit As Integer,
 ByRef psResult As MC07_S_RESULT) As Boolean`

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

引 数

hUnit …… ユニットハンドルを指定します。

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。

4-4-2. ユニット動作エラークリア関数

ユニット単位で動作エラーをクリアします。

動作エラーが検出された場合は、エラー要因を確認し、その要因を取り除いてから動作エラークリア関数を実行します。
動作エラークリア関数が実行されるまで、その他の関数実行は正常に行われません。

ユニット動作エラークリア関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

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) の動作エラーをクリアします。

書式

C言語 BOOL MC07_UCIError(DWORD *hUnit*, WORD *AxisSel*, MC07_S_RESULT **psResult*);

VB Function MC07_UCIError(ByVal *hUnit* As Long, ByVal *AxisSel* As Long, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_UCIError(ByVal *hUnit* As Integer, ByVal *AxisSel* As Integer, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.UCIError(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	意味
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	意味
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	意味
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。

4-4-3. 拡張ユニット通信関数

ユニットと拡張ユニット間の通信設定を行います。

最初に拡張ユニットの通信を設定した後、拡張ユニットとのサイクリック通信の開始/停止をコントロールします。

拡張I/Oの読み出し/書き込みは、ユニット関数またはI/O PORT関数で行います。

拡張ユニット通信設定関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機能

指定されたユニットと拡張ユニット間の通信設定を行います。

この関数は、拡張ユニット通信ステータスのPOLLING=0の状態で行ってください。

書式

C言語 BOOL MC07_UWExUnitCommMode(DWORD *hUnit*, WORD *CommRate*, WORD *RetryCount*, WORD *IoBit*,
MC07_S_RESULT **psResult*);

VB Function MC07_UWExUnitCommMode(ByVal *hUnit* As Long, ByVal *CommRate* As Integer,
ByVal *RetryCount* As Integer, ByVal *IoBit* As Integer,
ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_UWExUnitCommMode(ByVal *hUnit* As Integer, ByVal *CommRate* As Short,
ByVal *RetryCount* As Short, ByVal *IoBit* As Short,
ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07_UWExUnitCommMode(uint *hUnit*, ushort *CommRate*, ushort *RetryCount*, ushort *IoBit*,
ref MC07_S_RESULT *psResult*);

引数

hUnit … ユニットハンドルを指定します。

CommRate … 拡張ユニットとの通信の通信レートを指定します。

指定できる値	意味
MC07_EX_UNIT_COMM_RATE_5	5.0Mbps(固定)

RetryCount … 拡張ユニットとの通信リトライ回数(0~3)を指定します。(初期値は0回です。)
ユニットの電源投入後の初期値は0回です。

IoBit … 拡張ユニットとの通信のI/O点数を指定します。
ユニットの電源投入後の初期値はアンダーライン側です。

指定できる値	意味
MC07_EX_UNIT_COMM_32BIT	入力32点/出力32点
MC07_EX_UNIT_COMM_16BIT	入力16点/出力16点

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。

拡張ユニット通信制御関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
CB-52	CB-53	

機能

指定されたユニットと拡張ユニット間の通信を制御します。

書式

C言語 `BOOL MC07_UWExUnitCommControl(DWORD hUnit, WORD ControlSel, MC07_S_RESULT *psResult);`

VB `Function MC07_UWExUnitCommControl(ByVal hUnit As Long, ByVal ControlSel As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_UWExUnitCommControl(ByVal hUnit As Integer, ByVal ControlSel As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

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

引数

hUnit …… ユニットハンドルを指定します。

ControlSel …… 拡張ユニットとの通信の制御を指定します。

指定できる値	意味
MC07_EX_UNIT_COMM_START	通信を開始します。
MC07_EX_UNIT_COMM_STOP	通信を停止します。
MC07_EX_UNIT_COMM_DISC_LATCH_CLR	拡張ユニット通信のステータスのDISCONNECT LATCHをクリアします。

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。

拡張ユニット通信ステータス読み出し関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
CB-52	CB-53	

機能

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します。

書式

C言語 BOOL MC07_URExUnitCommStatus(DWORD *hUnit*, WORD **pStatus*, MC07_S_RESULT **psResult*);

VB Function MC07_URExUnitCommStatus(ByVal *hUnit* As Long, ByRef *pStatus* As Integer, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_URExUnitCommStatus(ByVal *hUnit* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MC07_S_RESULT) As Boolean

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

引数

hUnit … ユニットハンドルを指定します。
pStatus … 拡張ユニット通信ステータスを格納するための変数のポインタを指定します。
拡張ユニット通信ステータスの内容は、次に示す通りです。

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	DIS-CONNECT LATCH	CONNECT	POLLING

ユニット電源投入時の初期値は H'0000 (アンダーライン側) です。

D0 : POLLING

ユニットハンドルで指定されたユニットの通信の状態を示します。

1: 通信中の状態

0: 通信中でない状態

- ・ 拡張ユニット通信制御関数の引数 *ControlSel* に MC07_EX_UNIT_COMM_START を指定すると、POLLING=1 になります。
- ・ 拡張ユニット通信制御関数の引数 *ControlSel* に MC07_EX_UNIT_COMM_STOP を指定すると、POLLING=0 になります。

D1 : CONNECT

ユニットハンドルで指定されたユニットに接続される拡張ユニットの応答の状態を示します。

1: 接続されている状態

0: 接続されていない状態

D2 : DISCONNECT LATCH

ユニットハンドルで指定されたユニットと拡張ユニット間が、通信中に切断された有無を示します。

1: 切断された状態

0: 切断されていない状態

- ・ 拡張ユニット通信制御関数の引数 *ControlSel* に MC07_EX_UNIT_COMM_DISC_LATCH_CLR を指定すると、DISCONNECT LATCH=0 にクリアされます。

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。

拡張ユニット通信設定読み出し関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
CB-52	CB-53	

機能

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します。

書式

C言語 `BOOL MC07_URExUnitCommMode(DWORD hUnit, WORD *pCommRate, WORD *pRetryCount, WORD *ploBit, MC07_S_RESULT *psResult);`

VB `Function MC07_URExUnitCommMode(ByVal hUnit As Long, ByRef pCommRate As Integer, ByRef pRetryCount As Integer, ByRef ploBit As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_URExUnitCommMode(ByVal hUnit As Integer, ByRef pCommRate As Short, ByRef pRetryCount As Short, ByRef ploBit As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.URExUnitCommMode(uint hUnit, ref ushort pCommRate, ref ushort pRetryCount, ref ushort ploBit, ref MC07_S_RESULT psResult);`

引数

- hUnit* … ユニットハンドルを指定します。
pCommRate … 通信レートを格納するための変数のポインタを指定します。
pRetryCount … リトライ回数を格納するための変数のポインタを指定します。
ploBit … I/O点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。

4-4-4. ユニットアクセス関数

複数軸のMCC07 PORTアクセスと、複数のI/O PORTアクセスを一括で行います。

ユニットステータス構造体

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

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 _MC07_S_UNIT_STATUS {
        MC07_S_STATUS_DATA X;
        MC07_S_STATUS_DATA Y;
        MC07_S_STATUS_DATA Z;
        MC07_S_STATUS_DATA A;
        MC07_S_IN_PORT InPort;
    } MC07_S_UNIT_STATUS;
```

```
VB      Type MC07_S_UNIT_STATUS
        X As MC07_S_STATUS_DATA
        Y As MC07_S_STATUS_DATA
        Z As MC07_S_STATUS_DATA
        A As MC07_S_STATUS_DATA
        InPort As MC07_S_IN_PORT
    End Type
```

```
VB.NET  Structure MC07_S_UNIT_STATUS
        Public X As MC07_S_STATUS_DATA
        Public Y As MC07_S_STATUS_DATA
        Public Z As MC07_S_STATUS_DATA
        Public A As MC07_S_STATUS_DATA
        Public InPort As MC07_S_IN_PORT
    End Structure
```

```
C#.NET  struct MC07_S_UNIT_STATUS
    {
        public MC07_S_STATUS_DATA X;
        public MC07_S_STATUS_DATA Y;
        public MC07_S_STATUS_DATA Z;
        public MC07_S_STATUS_DATA A;
        public MC07_S_IN_PORT InPort;
    }
```

メンバ

X …… X軸のステータスデータ構造体の内容が格納されます。

Y …… Y軸のステータスデータ構造体の内容が格納されます。

Z …… Z軸のステータスデータ構造体の内容が格納されます。

A …… A軸のステータスデータ構造体の内容が格納されます。
ステータスデータ構造体は、基本的な構造体をご覧ください。

InPort …… 入力PORT構造体の内容が格納されます。

・ステータスデータ構造体は、4-2.章 基本的な構造体をご覧ください。

ユニットコマンド構造体

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

CB-52	CB-53
-------	-------

機能

ユニットのコマンドを格納します。

書式

```
C言語  typedef struct _MC07_S_UNIT_COMMAND {
        MC07_S_COMMAND_DATA X;
        MC07_S_COMMAND_DATA Y;
        MC07_S_COMMAND_DATA Z;
        MC07_S_COMMAND_DATA A;
        MC07_S_OUT_PORT OutPort;
    } MC07_S_UNIT_COMMAND;
```

```
VB      Type MC07_S_UNIT_COMMAND
        X As MC07_S_COMMAND_DATA
        Y As MC07_S_COMMAND_DATA
        Z As MC07_S_COMMAND_DATA
        A As MC07_S_COMMAND_DATA
        OutPort As MC07_S_OUT_PORT
    End Type
```

```
VB.NET  Structure MC07_S_UNIT_COMMAND
        Public X As MC07_S_COMMAND_DATA
        Public Y As MC07_S_COMMAND_DATA
        Public Z As MC07_S_COMMAND_DATA
        Public A As MC07_S_COMMAND_DATA
        Public OutPort As MC07_S_OUT_PORT
    End Structure
```

```
C#.NET  struct MC07_S_UNIT_COMMAND
    {
        public MC07_S_COMMAND_DATA X;
        public MC07_S_COMMAND_DATA Y;
        public MC07_S_COMMAND_DATA Z;
        public MC07_S_COMMAND_DATA A;
        public MC07_S_OUT_PORT OutPort;
    }
```

メンバ

X … X軸のコマンドデータ構造体の内容を格納します。
Y … Y軸のコマンドデータ構造体の内容を格納します。
Z … Z軸のコマンドデータ構造体の内容を格納します。
A … A軸のコマンドデータ構造体の内容を格納します。
OutPort … 出力PORT構造体の内容を格納します。

・コマンドデータ構造体は、4-2.章 基本的な構造体をご覧ください。

入力PORT構造体

UsbC

UC-766	UCD-7610v1	UCD-7613v1
CB-52	CB-53	

機能

ユニットの入力PORTから読み出された内容が格納されます。
読み出された内容については、6-3-1.章 汎用I/O PORTをご覧ください。

書式

```
C言語  typedef struct _MC07_S_IN_PORT {
        WORD  Gpin;
        WORD  Gp0in;
        WORD  Gp1in;
        WORD  Exp0in;
        WORD  Exp1in;
        WORD  Ctlp0in;
    } MC07_S_IN_PORT;
```

```
VB      Type MC07_S_IN_PORT
        Gpin As Integer
        Gp0in As Integer
        Gp1in As Integer
        Exp0in As Integer
        Exp1in As Integer
        Ctlp0in As Integer
    End Type
```

```
VB.NET  Structure MC07_S_IN_PORT
        Public Gpin As Short
        Public Gp0in As Short
        Public Gp1in As Short
        Public Exp0in As Short
        Public Exp1in As Short
        Public Ctlp0in As Short
    End Structure
```

```
C#.NET  struct MC07_S_IN_PORT
    {
        public ushort Gpin;
        public ushort Gp0in;
        public ushort Gp1in;
        public ushort Exp0in;
        public ushort Exp1in;
        public ushort Ctlp0in;
    }
```

メンバ

<i>Gpin</i>	… 汎用I/O入力PORTから読み出された内容が格納されます。
<i>Gp0in</i>	… 将来の拡張用です。
<i>Gp1in</i>	… 将来の拡張用です。
<i>Exp0in</i>	… 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。
<i>Exp1in</i>	… 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。
<i>Ctlp0in</i>	… 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。

出力PORT構造体

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機能

ユニットの出力PORTに書き込むデータ、OR書き込みするデータ、AND書き込みするデータを格納します。

- ・書き込むデータ ... I/O PORT書き込み関数
- ・OR書き込みデータ ... I/O PORT OR書き込み関数
- ・AND書き込みデータ... I/O PORT AND書き込み関数

書き込むデータについては、6-3-1.章 汎用 I/O PORTをご覧ください。

書式

```
C言語  typedef struct _MC07_S_OUT_PORT {
        WORD  Gpout;
        WORD  Gp0out;
        WORD  Gp1out;
        WORD  Exp0out;
        WORD  Exp1out;
        WORD  Ctlp0out;
    } MC07_S_OUT_PORT;
```

```
VB      Type MC07_S_OUT_PORT
        Gpout As Integer
        Gp0out As Integer
        Gp1out As Integer
        Exp0out As Integer
        Exp1out As Integer
        Ctlp0out As Integer
    End Type
```

```
VB.NET  Structure MC07_S_OUT_PORT
        Public Gpout As Short
        Public Gp0out As Short
        Public Gp1out As Short
        Public Exp0out As Short
        Public Exp1out As Short
        Public Ctlp0out As Short
    End Structure
```

```
C#.NET  struct MC07_S_OUT_PORT
    {
        public ushort Gpout;
        public ushort Gp0out;
        public ushort Gp1out;
        public ushort Exp0out;
        public ushort Exp1out;
        public ushort Ctlp0out;
    }
```

メンバ

<i>Gpout</i>	… 汎用 I/O 出力 PORT に書き込むデータを格納します。
<i>Gp0out</i>	… 将来の拡張用です。
<i>Gp1out</i>	… 将来の拡張用です。
<i>Exp0out</i>	… 拡張 I/O 出力 0 PORT に書き込むデータを格納します。
<i>Exp1out</i>	… 拡張 I/O 出力 1 PORT に書き込むデータを格納します。
<i>Ctlp0out</i>	… 制御 I/O 出力 0 PORT に書き込むデータを格納します。

ユニットDRIVE COMMAND・I/O書き込み関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機能

指定されたユニットに対し、次の書き込みを一括で行います。

- ・指定された軸（複数指定可）のDRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE COMMAND PORTに、軸ごとに個別のコマンドコードとデータを書き込みます。
- ・指定されたI/O PORT（複数指定可）に、I/O PORTごとに個別のデータを書き込みます。

書式

C言語 BOOL MC07_UWDriveIo(DWORD *hUnit*, DWORD *AxisSel*, DWORD *IoPortSel*, MC07_S_UNIT_COMMAND **psUnitCmd*, MC07_S_RESULT **psResult*);

VB Function MC07_UWDriveIo(ByVal *hUnit* As Long, ByVal *AxisSel* As Long, ByVal *IoPortSel* As Long, ByRef *psUnitCmd* As MC07_S_UNIT_COMMAND, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_UWDriveIo(ByVal *hUnit* As Integer, ByVal *AxisSel* As Integer, ByVal *IoPortSel* As Integer, ByRef *psUnitCmd* As MC07_S_UNIT_COMMAND, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.UWDriveIo(uint *hUnit*, uint *AxisSel*, uint *IoPortSel*, ref MC07_S_UNIT_COMMAND *psUnitCmd*, 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。
どの軸も指定しない場合は0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	意味
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	意味	指定できる値	意味
MC07_SEL_X_Y	X軸とY軸	MC07_SEL_X_Y_Z	X軸とY軸とZ軸
MC07_SEL_X_Z	X軸とZ軸	MC07_SEL_X_Y_A	X軸とY軸とA軸
MC07_SEL_X_A	X軸とA軸	MC07_SEL_X_Z_A	X軸とZ軸とA軸
MC07_SEL_Y_Z	Y軸とZ軸	MC07_SEL_Y_Z_A	Y軸とZ軸とA軸
MC07_SEL_Y_A	Y軸とA軸	MC07_SEL_X_Y_Z_A	X軸とY軸とZ軸とA軸
MC07_SEL_Z_A	Z軸とA軸		

- IoPortSel* …… I/O PORT（複数指定可）の組み合わせを指定します。
どのI/O PORTも指定しない場合は0を指定します。

複数のI/O PORTを指定する場合は、論理和を指定します

指定できる値	意味
MC07_SEL_GP_OUT	汎用I/O出力 PORT
MC07_SEL_EXP0_OUT	拡張I/O出力0 PORT
MC07_SEL_EXP1_OUT	拡張I/O出力1 PORT
MC07_SEL_CTLPO_OUT	制御I/O出力0 PORT

次の指定も組み合わせることができます。

指定できる値	意味
MC07_SEL_EXP0_EXP1_OUT	拡張I/O出力0 PORTと拡張I/O出力1 PORT

- psUnitCmd* …… 書き込むデータが格納されているユニットコマンド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。

ユニットDRIVE COMMAND書き込み / 読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたユニットに対し、次の書き込みと読み出しを書き込み 読み出しの順で一括で行います。

- ・指定された軸（複数指定可）のDRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE COMMAND PORTに、軸ごとに個別のコマンドコードとデータを書き込みます。
- ・指定された軸（複数指定可）のDRIVE DATA1 PORT、DRIVE DATA2 PORTの内容を読み出します。

ユニット単位で下記のアクセスを1回の関数実行で処理することができます。

- ・設定データの読み出し
- ・出力中のドライブ速度の読み出し
- ・エラーステータスの読み出し
- ・各カウントデータの読み出し

書式

C言語 BOOL MC07_UWRDrive(DWORD *hUnit*, DWORD *AxisSel*, MC07_S_UNIT_COMMAND **psUnitCmd*, MC07_S_UNIT_STATUS **psUnitStatus*, MC07_S_RESULT **psResult*);

VB Function MC07_UWRDrive(ByVal *hUnit* As Long, ByVal *AxisSel* As Long, ByRef *psUnitCmd* As MC07_S_UNIT_COMMAND, ByRef *psUnitStatus* As MC07_S_UNIT_STATUS, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_UWRDrive(ByVal *hUnit* As Integer, ByVal *AxisSel* As Integer, ByRef *psUnitCmd* As MC07_S_UNIT_COMMAND, ByRef *psUnitStatus* As MC07_S_UNIT_STATUS, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.UWRDrive(uint *hUnit*, uint *AxisSel*, ref MC07_S_UNIT_COMMAND *psUnitCmd*, ref MC07_S_UNIT_STATUS *psUnitStatus*, 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。どの軸も指定しない場合は0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	意味
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	意味	指定できる値	意味
MC07_SEL_X_Y	X軸とY軸	MC07_SEL_X_Y_Z	X軸とY軸とZ軸
MC07_SEL_X_Z	X軸とZ軸	MC07_SEL_X_Y_A	X軸とY軸とA軸
MC07_SEL_X_A	X軸とA軸	MC07_SEL_X_Z_A	X軸とZ軸とA軸
MC07_SEL_Y_Z	Y軸とZ軸	MC07_SEL_Y_Z_A	Y軸とZ軸とA軸
MC07_SEL_Y_A	Y軸とA軸	MC07_SEL_X_Y_Z_A	X軸とY軸とZ軸とA軸
MC07_SEL_Z_A	Z軸とA軸		

psUnitCmd …… 書き込むデータが格納されているユニットコマンド構造体のポインタを指定します。

psUnitStatus …… 読み出した内容を格納するためのユニットステータス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。

ユニット STATUS1・I/O読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機能

指定されたユニットに対し、次の読み出しを一括で行います。
 ・指定された軸（複数指定可）の STATUS1 PORT の内容を読み出します。
 ・指定された I/O PORT（複数指定可）の内容を読み出します。

書式

C言語 BOOL MC07_URStatus1Io(DWORD *hUnit*, DWORD *AxisSel*, DWORD *IoPortSel*,
MC07_S_UNIT_STATUS **psUnitStatus*, MC07_S_RESULT **psResult*);

VB Function MC07_URStatus1Io(ByVal *hUnit* As Long, ByVal *AxisSel* As Long,
ByVal *IoPortSel* As Long, ByRef *psUnitStatus* As MC07_S_UNIT_STATUS,
ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_URStatus1Io(ByVal *hUnit* As Integer, ByVal *AxisSel* As Integer,
ByVal *IoPortSel* As Integer, ByRef *psUnitStatus* As MC07_S_UNIT_STATUS,
ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.URStatus1Io(uint *hUnit*, uint *AxisSel*, uint *IoPortSel*,
ref MC07_S_UNIT_STATUS *psUnitStatus*, 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。
 どの軸も指定しない場合は0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	意味
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	意味	指定できる値	意味
MC07_SEL_X_Y	X軸とY軸	MC07_SEL_X_Y_Z	X軸とY軸とZ軸
MC07_SEL_X_Z	X軸とZ軸	MC07_SEL_X_Y_A	X軸とY軸とA軸
MC07_SEL_X_A	X軸とA軸	MC07_SEL_X_Z_A	X軸とZ軸とA軸
MC07_SEL_Y_Z	Y軸とZ軸	MC07_SEL_Y_Z_A	Y軸とZ軸とA軸
MC07_SEL_Y_A	Y軸とA軸	MC07_SEL_X_Y_Z_A	X軸とY軸とZ軸とA軸
MC07_SEL_Z_A	Z軸とA軸		

IoPortSel …… I/O PORT（複数指定可）の組み合わせを指定します。
 どの I/O PORT も指定しない場合は0を指定します。

複数の I/O PORT を指定する場合は、論理和を指定します

指定できる値	意味
MC07_SEL_GP_IN	汎用 I/O 入力 PORT
MC07_SEL_EXP0_IN	拡張 I/O 入力0 PORT
MC07_SEL_EXP1_IN	拡張 I/O 入力1 PORT
MC07_SEL_CTLPO_IN	制御 I/O 入力0 PORT

次の指定も組み合わせることができます。

指定できる値	意味
MC07_SEL_EXP0_EXP1_IN	拡張 I/O 入力0 PORT と 拡張 I/O 入力1 PORT

psUnitStatus …… 読み出した内容を格納するためのユニットステータス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。

ユニットSTATUS1・パルスカウンタ・I/O読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機能

指定されたユニットに対し、次の読み出しを一括で行います。

- ・指定された軸（複数指定可）のSTATUS1 PORTの内容を読み出します。
- ・指定された軸（複数指定可）のDRIVE DATA1 PORT、DRIVE DATA2 PORTからパルスカウンタの内容を読み出します。
- ・指定されたI/O PORT（複数指定可）の内容を読み出します。

書式

C言語 BOOL MC07_URStatus1PcntIo(DWORD *hUnit*, DWORD *AxisSel*, DWORD *IoPortSel*, MC07_S_UNIT_STATUS **psUnitStatus*, MC07_S_RESULT **psResult*);

VB Function MC07_URStatus1PcntIo(ByVal *hUnit* As Long, ByVal *AxisSel* As Long, ByVal *IoPortSel* As Long, ByRef *psUnitStatus* As MC07_S_UNIT_STATUS, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_URStatus1PcntIo(ByVal *hUnit* As Integer, ByVal *AxisSel* As Integer, ByVal *IoPortSel* As Integer, ByRef *psUnitStatus* As MC07_S_UNIT_STATUS, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.URStatus1PcntIo(uint *hUnit*, uint *AxisSel*, uint *IoPortSel*, ref MC07_S_UNIT_STATUS *psUnitStatus*, 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。
どの軸も指定しない場合は0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	意味
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	意味	指定できる値	意味
MC07_SEL_X_Y	X軸とY軸	MC07_SEL_X_Y_Z	X軸とY軸とZ軸
MC07_SEL_X_Z	X軸とZ軸	MC07_SEL_X_Y_A	X軸とY軸とA軸
MC07_SEL_X_A	X軸とA軸	MC07_SEL_X_Z_A	X軸とZ軸とA軸
MC07_SEL_Y_Z	Y軸とZ軸	MC07_SEL_Y_Z_A	Y軸とZ軸とA軸
MC07_SEL_Y_A	Y軸とA軸	MC07_SEL_X_Y_Z_A	X軸とY軸とZ軸とA軸
MC07_SEL_Z_A	Z軸とA軸		

- IoPortSel* …… I/O PORT（複数指定可）の組み合わせを指定します。
どのI/O PORTも指定しない場合は0を指定します。

複数のI/O PORTを指定する場合は、論理和を指定します

指定できる値	意味
MC07_SEL_GP_IN	汎用I/O入力 PORT
MC07_SEL_EXP0_IN	拡張I/O入力0 PORT
MC07_SEL_EXP1_IN	拡張I/O入力1 PORT
MC07_SEL_CTLPO_IN	制御I/O入力0 PORT

次の指定も組み合わせることができます。

指定できる値	意味
MC07_SEL_EXP0_EXP1_IN	拡張I/O入力0 PORTと拡張I/O入力1 PORT

psUnitStatus …… 読み出した内容を格納するためのユニットステータス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。

ユニット I/O PORT 書き込み関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機能

指定されたユニットに対し、次の書き込みを一括で行います。

- ・指定された I/O PORT (複数指定可) に、I/O PORT ごとに個別のデータを書き込みます。

書式

C言語 `BOOL MC07_UPortOut(DWORD hUnit, DWORD IoPortSel, MC07_S_OUT_PORT *psOutPort, MC07_S_RESULT *psResult);`

VB `Function MC07_UPortOut(ByVal hUnit As Long, ByVal IoPortSel As Long, ByRef psOutPort As MC07_S_OUT_PORT, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_UPortOut(ByVal hUnit As Integer, ByVal IoPortSel As Integer, ByRef psOutPort As MC07_S_OUT_PORT, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.UPortOut(uint hUnit, uint IoPortSel, ref MC07_S_OUT_PORT psOutPort, ref MC07_S_RESULT psResult);`

引数

hUnit …… ユニットハンドルを指定します。

IoPortSel …… I/O PORT (複数指定可) を指定します。

複数の I/O PORT を指定する場合は、論理和を指定します

指定できる値	意味
MC07_SEL_GP_OUT	汎用 I/O 出力 PORT
MC07_SEL_EXPO_OUT	拡張 I/O 出力 0 PORT
MC07_SEL_EXP1_OUT	拡張 I/O 出力 1 PORT
MC07_SEL_CTLPO_OUT	制御 I/O 出力 0 PORT

次の指定も組み合わせることができます

指定できる値	意味
MC07_SEL_EXPO_EXP1_OUT	拡張 I/O 出力 0 PORT と拡張 I/O 出力 1 PORT

psOutPort …… 書き込むデータが格納されている出力 PORT 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。

ユニット I/O PORT OR書き込み関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機能

指定されたユニットに対し、次の書き込みを一括で行います。

- ・指定された I/O PORT (複数指定可) に、I/O PORT ごとに個別のデータを OR書き込みします。

書式

C言語 BOOL MC07_UPortOrOut(DWORD *hUnit*, DWORD *IoPortSel*, MC07_S_OUT_PORT **psOutPort*, MC07_S_RESULT **psResult*);

VB Function MC07_UPortOrOut(ByVal *hUnit* As Long, ByVal *IoPortSel* As Long, ByRef *psOutPort* As MC07_S_OUT_PORT, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_UPortOrOut(ByVal *hUnit* As Integer, ByVal *IoPortSel* As Integer, ByRef *psOutPort* As MC07_S_OUT_PORT, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.UPortOrOut(uint *hUnit*, uint *IoPortSel*, ref MC07_S_OUT_PORT *psOutPort*, ref MC07_S_RESULT *psResult*);

引数

hUnit ... ユニットハンドルを指定します。

IoPortSel ... I/O PORT (複数指定可) を指定します。

複数の I/O PORT を指定する場合は、論理和を指定します

指定できる値	意味
MC07_SEL_GP_OUT	汎用 I/O 出力 PORT
MC07_SEL_EXPO_OUT	拡張 I/O 出力 0 PORT
MC07_SEL_EXP1_OUT	拡張 I/O 出力 1 PORT
MC07_SEL_CTLPO_OUT	制御 I/O 出力 0 PORT

次の指定も組み合わせることができます

指定できる値	意味
MC07_SEL_EXPO_EXP1_OUT	拡張 I/O 出力 0 PORT と拡張 I/O 出力 1 PORT

psOutPort ... OR書き込みするデータが格納されている出力 PORT 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。

ユニット I/O PORT AND書き込み関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機能

指定されたユニットに対し、次の書き込みを一括で行います。

- ・指定された I/O PORT (複数指定可) に、I/O PORT ごとに個別のデータを AND 書き込みします。

書式

C言語 `BOOL MC07_UPortAndOut(DWORD hUnit, DWORD IoPortSel, MC07_S_OUT_PORT *psOutPort, MC07_S_RESULT *psResult);`

VB `Function MC07_UPortAndOut(ByVal hUnit As Long, ByVal IoPortSel As Long, ByRef psOutPort As MC07_S_OUT_PORT, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_UPortAndOut(ByVal hUnit As Integer, ByVal IoPortSel As Integer, ByRef psOutPort As MC07_S_OUT_PORT, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.UPortAndOut(uint hUnit, uint IoPortSel, ref MC07_S_OUT_PORT psOutPort, ref MC07_S_RESULT psResult);`

引数

hUnit ... ユニットハンドルを指定します。

IoPortSel ... I/O PORT (複数指定可) を指定します。

複数の I/O PORT を指定する場合は、論理和を指定します

指定できる値	意味
MC07_SEL_GP_OUT	汎用 I/O 出力 PORT
MC07_SEL_EXP0_OUT	拡張 I/O 出力 0 PORT
MC07_SEL_EXP1_OUT	拡張 I/O 出力 1 PORT
MC07_SEL_CTLPO_OUT	制御 I/O 出力 0 PORT

次の指定も組み合わせることができます

指定できる値	意味
MC07_SEL_EXP0_EXP1_OUT	拡張 I/O 出力 0 PORT と拡張 I/O 出力 1 PORT

psOutPort ... AND 書き込みするデータが格納されている出力 PORT 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。

ユニット I/O PORT 読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機能

指定されたユニットに対し、次の読み出しを一括で行います。
 ・指定された I/O PORT (複数指定可) の内容を読み出します。

書式

C言語 BOOL MC07_UPortIn(DWORD *hUnit*, DWORD *IoPortSel*, MC07_S_IN_PORT **psInPort*,
MC07_S_RESULT **psResult*);

VB Function MC07_UPortIn(ByVal *hUnit* As Long, ByVal *IoPortSel* As Long,
ByRef *psInPort* As MC07_S_IN_PORT, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_UPortIn(ByVal *hUnit* As Integer, ByVal *IoPortSel* As Integer,
ByRef *psInPort* As MC07_S_IN_PORT, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.UPortIn(uint *hUnit*, uint *IoPortSel*, ref MC07_S_IN_PORT *psInPort*,
ref MC07_S_RESULT *psResult*);

引数

hUnit …… ユニットハンドルを指定します。
IoPortSel …… I/O PORT (複数指定可) を指定します。

複数の I/O PORT を指定する場合は、論理和を指定します

指定できる値	意味
MC07_SEL_GP_IN	汎用 I/O 入力 PORT
MC07_SEL_EXPO_IN	拡張 I/O 入力 0 PORT
MC07_SEL_EXP1_IN	拡張 I/O 入力 1 PORT
MC07_SEL_CTLPO_IN	制御 I/O 入力 0 PORT

次の指定も組み合わせることができます

指定できる値	意味
MC07_SEL_EXPO_EXP1_IN	拡張 I/O 入力 0 PORT と 拡張 I/O 入力 1 PORT

psInPort …… 読み出した内容を格納するための入力 PORT 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。

4-5. デバイス関数

MCC07 の 1 軸をデバイスと呼称します。

デバイスオープン関数で取得したデバイスハンドルにより、デバイスを制御します。

4-5-1. デバイスオープン/クローズ関数

ユーザアプリケーションは、デバイスオープンし、デバイスハンドルを受け取ります。

以後、デバイス関数を実行する際に、このデバイスハンドルを引数として渡します。

このデバイスハンドルは、デバイスをクローズするまで有効です。

ユーザアプリケーション終了時は、必ずデバイスをクローズしてください。

デバイスオープン関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

機能

指定されたデバイスをオープンし、引数 *phDev* で示される変数にデバイスハンドルを格納します。

書式

C言語 `BOOL MC07_BOpen(WORD UnitNo, WORD Axis, DWORD *phDev, MC07_S_RESULT *psResult);`

VB `Function MC07_BOpen(ByVal UnitNo As Integer, ByVal Axis As Integer, ByRef phDev As Long, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BOpen(ByVal UnitNo As Short, ByVal Axis As Short, ByRef phDev As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BOpen(ushort UnitNo, ushort Axis, ref uint phDev, ref MC07_S_RESULT psResult);`

引数

UnitNo …… ユニット番号を指定します。

指定できる値	意味
MC07_USB_UNIT_0	ユニット番号0
MC07_USB_UNIT_1	ユニット番号1

Axis …… 軸を指定します。

指定できる値	意味	4軸製品	2軸製品
MC07_X	X軸		
MC07_Y	Y軸		
MC07_Z	Z軸		-
MC07_A	A軸		-

phDev …… デバイスハンドル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。

デバイスクローズ関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスをクローズします。

書式

C言語 `BOOL MC07_BClose(DWORD hDev, MC07_S_RESULT *psResult);`

VB `Function MC07_BClose(ByVal hDev As Long, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BClose(ByVal hDev As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BClose(uint hDev, ref MC07_S_RESULT psResult);`

引数

hDev … デバイスハンドルを指定します。

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。

4-5-2. 動作エラークリア関数

デバイス単位で動作エラーをクリアします。

動作エラーが検出された場合は、エラー要因を確認し、その要因を取り除いてから動作エラークリア関数を実行します。動作エラークリア関数が実行されるまで、その他の関数実行は正常に行われません。

動作エラークリア関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

機能

指定されたデバイスの動作エラーをクリアします。

書式

C言語 `BOOL MC07_ClrError(DWORD hDev, MC07_S_RESULT *psResult);`

VB `Function MC07_ClrError(ByVal hDev As Long, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_ClrError(ByVal hDev As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.ClrError(uint hDev, ref MC07_S_RESULT psResult);`

引数

hDev …… デバイスハンドルを指定します。

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。

4-5-3. MCC07 PORT アクセス関数

MCC07 PORT の読み出しと書き込みを行います。

DRIVE COMMAND PORT

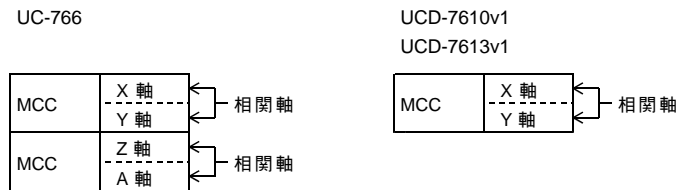
DRIVE COMMAND を書き込む PORT です。この PORT に DRIVE COMMAND を書き込むと、データの設定またはドライブの実行を行います。

書き込む DRIVE COMMAND は下位 8 ビットのみ有効です。上位 8 ビットは無視します。

DRIVE COMMAND には、汎用コマンド (H'00 ~ H'7F) と特殊コマンド (H'80 ~ H'FF) があります。

汎用コマンド (H'00 ~ H'3F)

- 汎用コマンドは DRIVE STATUS1 PORT の BUSY=0、ERROR=0 の時に書き込むことができます。
 - 汎用コマンドの内、補間ドライブの 2 軸相関コマンドについては、相関軸両軸の BUSY=0、ERROR=0 の時に書き込むことができます。
- 当製品での相関軸は以下の通りです。



- 汎用コマンドには、コマンド予約機能があります。(応用機能)
BUSY = 1 でも、STATUS1 PORT の COMREG FL = 0 のときには書き込みができます。
2 軸相関コマンドを予約する場合、相関軸両軸の COMREG FL=0、ERROR=0 の時に書き込むことができます。
書き込んだ汎用コマンド(予約コマンド)は、コマンド予約機能の予約レジスタに格納します。
予約レジスタには、10 命令分の汎用コマンドを格納することができます。
- ERROR = 1 になると、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。

特殊コマンド (H'80 ~ H'FF)

- SPEED CHANGE の設定コマンド (H'C1) は、
DRIVE STATUS5 PORT の SPEED CSET=0 の時に書き込むことができます。
- SPEED CHANGE の実行コマンド (H'C8, H'C9) は、
DRIVE STATUS5 PORT の SPEED CBUSY=0 の時に書き込むことができます。
- INDEX CHANGE の設定コマンド (H'C3) は、
DRIVE STATUS5 PORT の INDEX CSET=0 の時に書き込むことができます。
- INDEX CHANGE の実行コマンド (H'CC, H'CD, H'CE) は
DRIVE STATUS5 PORT の INDEX CBUSY=0 の時に書き込むことができます。
- その他の特殊コマンドの書き込みは、3-7.章の制限事項を除き、常時可能です。
ERROR = 1 でも、コマンドの実行は有効です。

DRIVE DATA PORT(書き込み)

DRIVE COMMAND の設定データ、または指定したドライブの動作データを書き込む PORT です。
この PORT への書き込みは常時可能です。

DRIVE STATUS PORT

これらの STATUS PORT の読み出しは、3-7.章の制限事項を除き、常時可能です。

DRIVE STATUS1 PORT

ドライブコントロールの現在の状態を表示する PORT です。
詳細は「DRIVE STATUS1 PORT 読み出し関数」をご覧ください。

DRIVE STATUS2 PORT

外部入出力信号の状態を表示する PORT です。
詳細は「DRIVE STATUS2 PORT 読み出し関数」をご覧ください。

DRIVE STATUS3 PORT

割り込み要求出力とステータス信号の状態を表示する PORT です。
詳細は「DRIVE STATUS3 PORT 読み出し関数」をご覧ください。

DRIVE STATUS4 PORT

カウンタのコンパレータ出力状態とオーバーフローを表示する PORT です。
詳細は「DRIVE STATUS4 PORT 読み出し関数」をご覧ください。

DRIVE STATUS5 PORT

入力信号とドライブ CHANGE 指令の現在の状態を表示する PORT です。
詳細は「DRIVE STATUS5 PORT 読み出し関数」をご覧ください。

DRIVE STATUS バッファ

上記 DRIVE STATUS1 PORT から DRIVE STATUS5 PORT までと ORIGIN STATUS を一括で読み出す関数を用意しています。

詳細は「DRIVE STATUS バッファ読み出し関数」をご覧ください。
ORIGIN ドライブ STATUS の内容については、「ORIGIN STATUS 読み出し関数」をご覧ください。

DRIVE DATA PORT(読み出し)

各種データを読み出す PORT です。
READ コマンドを DRIVE COMMAND PORT に書き込むと、該当データを DRIVE DATA1,2 PORT にセットします。
DRIVE DATA1,2 PORT にセットしたデータは次の READ コマンドの書き込みまで保持します。
新しいデータを読み出す場合は、都度 READ コマンドを実行してから読み出します。

DRIVE COMMAND 32ビット書き込み関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

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 MC07_LWDrive(DWORD hDev, WORD Cmd, DWORD *pData, MC07_S_RESULT *psResult);`

VB `Function MC07_LWDrive(ByVal hDev As Long, ByVal Cmd As Integer, ByRef pData As Long, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_LWDrive(ByVal hDev As Integer, ByVal Cmd As Short, ByRef pData As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.LWDrive(uint hDev, ushort Cmd, ref uint pData, ref MC07_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。
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 COMMAND PORT書き込み関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスのDRIVE COMMAND PORTにコマンドコードを書き込みます。
データを持たないコマンドコードの書き込みで使用します。

- ・ JOG コマンド
- ・ SCAN コマンド
- ・ FAST STOP コマンド
- ・ SLOW STOP コマンドなど

書式

C言語 BOOL MC07_BWDriveCommand(DWORD *hDev*, WORD **pCmd*, MC07_S_RESULT **psResult*);

VB Function MC07_BWDriveCommand(ByVal *hDev* As Long, ByRef *pCmd* As Integer, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_BWDriveCommand(ByVal *hDev* As Integer, ByRef *pCmd* As Short, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.BWDriveCommand(uint *hDev*, ref ushort *pCmd*, ref MC07_S_RESULT *psResult*);

引数

hDev … デバイスハンドルを指定します。
pCmd … 書き込むコマンドコード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 STATUS1 PORT読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスのDRIVE STATUS1 PORTを読み出します。
ドライブコントロールの現在の状態を表示するPORTです。

書式

C言語 BOOL MC07_BRStatus1(DWORD *hDev*, WORD **pStatus*, MC07_S_RESULT **psResult*);

VB Function MC07_BRStatus1(ByVal *hDev* As Long, ByRef *pStatus* As Integer, ByRef *psResult* As MC07_S_RESULT)As Boolean

VB.NET Function MC07_BRStatus1(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.BRStatus1(uint *hDev*, ref ushort *pStatus*, ref MC07_S_RESULT *psResult*);

引数

hDev … デバイスハンドルを指定します。
pStatus … 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 STATUS1 PORTの内容は、次に示す通りです。

D15	D14	D13	D12	D11	D10	D9	D8
COMREG FL	COMREG EP	PAUSE	0	EXT PULSE	CONST	DOWN	UP
D7	D6	D5	D4	D3	D2	D1	D0
FSEND	SSEND	LSEND	ERROR	DRVEND	DRIVE	STBY	BUSY

D0 : BUSY

汎用コマンド処理中、ドライブ実行中、または<サーボ対応>実行中の状態を示します。
1 : 汎用コマンドの実行と同時にセットします
またはEXT PULSE = 0 1と同時にセットします
0 : 汎用コマンドの終了およびDEND, DRST信号の<サーボ対応>の終了でクリアします
またはEXT PULSE = 1 0でクリアします
・2軸相関コマンド実行中は相関軸2軸ともBUSY=1となります。

D1 : STBY

ドライブパルス出力の準備(パラメータ処理)が完了した状態を示します。
1 : パルス出力の準備が完了した状態
0 : SPEC INITIALIZE3コマンドのSTBY TYPE (STBY解除条件)の検出でクリアします
停止指令を検出した場合は、強制終了と同時にクリアします

D2 : DRIVE

ドライブパルス出力中の状態を示します。
1 : パルス出力中の状態
0 : パルス出力停止中の状態

DRIVE STATUS2 PORTのPULSE MASK = 1のときは、ドライブパルス出力はOFFレベル固定です。

D3 : DRVEND

ドライブの実行または<サーボ対応>の実行を終了したことを示します。

- 1 : 以下の汎用コマンドまたはドライブが実行された後のBUSY = 1 0と同時にセットします
 - ・パルス出力を伴う汎用コマンド
 - ・DRSTコマンド
 - ・外部パルス出力 (EXT PULSE = 1)
- 0 : 次のBUSY = 0 1と同時にクリアします

停止指令またはエラーの発生により、ドライブの実行をパルス出力なしで終了した場合も、BUSY = 0と同時に、DRVEND = 1にします。

コマンド予約機能の予約コマンドによる連続ドライブ中 (BUSY = 1) に、DRVENDフラグをセットする汎用コマンドを実行した場合は、予約コマンド終了後のBUSY = 0と同時に、DRVEND = 1にします。

2軸相関補間ドライブ終了時は、各軸ともにDRVEND = 1になります。

D4 : ERROR

エラーが発生したことを示します。

- 1 : マスクしていないERROR STATUSが、"1" になった状態
 - 0 : マスクしていないERROR STATUSが、すべて "0" の状態
- ERROR STATUSは、動作エラークリア関数の実行でクリアします
- ・ ERROR STATUSは、検出条件が一致している間はクリアできません

ERRORフラグは、D15を除く15個のERROR STATUSのOR (論理和) 出力です。

- ・ ERROR STATUSは、ERROR STATUS MASKコマンドで個別にマスクできます。
- ・ ERROR STATUSは、ERROR STATUS READコマンドで読み出しできます。
- ・ ERROR=1の間はCOMREG FL=1、COMREG=1となり汎用コマンドの書き込みが無効となります。
- ・ 補間ドライブでエラーが発生した場合、エラー該当軸がERROR=1となります。

D5 : LSEND

LIMIT減速停止指令またはLIMIT即時停止指令を検出したことを示します。

- 1 : LIMIT減速停止指令またはLIMIT即時停止指令を検出した状態
 - 0 : 次のBUSY = 0 1と同時にクリアします
- またはNO OPERATIONコマンドの実行でクリアします
- EXT PULSE = 1の場合は、次のパルス出力開始でクリアします

2軸相関補間ドライブでLIMIT停止指令を検出した場合、相関軸両軸がLSEND=1となります。

LIMIT減速停止指令

- ・ 入力機能をLIMIT減速停止に設定したCWLM, CCWLM信号
- ・ 停止機能をLIMIT減速停止に設定した各種カウンタのコンパレータ出力

LIMIT即時停止指令

- ・ 入力機能をLIMIT即時停止に設定したCWLM, CCWLM信号
- ・ 停止機能をLIMIT即時停止に設定した各種カウンタのコンパレータ出力

D6 : SSEND

減速停止指令を検出したことを示します。

- 1 : 減速停止指令を検出した状態
 - 0 : 次のBUSY = 0 1と同時にクリアします
- またはNO OPERATIONコマンドの実行でクリアします

2軸補間ドライブで減速停止指令を検出した場合、相関軸両軸がSSEND=1となります。

減速停止指令

- ・ SLOW STOPコマンド
- ・ 入力機能を減速停止に設定したSS0, DEND, DALM信号
- ・ 停止機能を減速停止に設定した各種カウンタのコンパレータ出力

D7 : FSEND

即時停止指令を検出したことを示します。

- 1 : 即時停止指令を検出した状態
- 0 : 次のBUSY = 0 1と同時にクリアします

即時停止指令がアクティブでも、データ設定コマンドの処理は正常に実行します。

- ・ 即時停止指令の検出でFSEND = 1にし、コマンド処理終了後にBUSY = 0にします。

2軸補間ドライブで即時停止指令を検出した場合、相関軸両軸がFSEND=1となります。

即時停止指令

- ・ FAST STOPコマンド
- ・ FSSTOP信号
- ・ 入力機能を即時停止に設定したCWLM, CCWLM信号
- ・ 入力機能を即時停止に設定したSS0, DEND, DALM信号
- ・ 停止機能を即時停止に設定した各種カウンタのコンパレータ出力

D8 : UP

出力中のドライブパルス速度が、加速中の状態を示します。

- 1 : 加速中の状態
- 0 : 減速中または一定速中または停止中の状態

D9 : DOWN

出力中のドライブパルス速度が、減速中の状態を示します。

- 1 : 減速中の状態
- 0 : 加速中または一定速中または停止中の状態

D10 : CONST

出力中のドライブパルス速度が、一定速中の状態を示します。

- 1 : 一定速中の状態
- 0 : 加速中または減速中または停止中の状態

補間ドライブ実行中は、基本パルス出力軸のUP, DOWN, CONSTフラグのみが有効です。

D11 : EXT PULSE(応用機能)

ADDRESS COUNTER INITIALIZE1コマンドのCOUNT PULSE SELを

「外部パルス出力」に設定している状態を示します。

- 1 : COUNT PULSE SELを「01: 他軸のパルス」、「10, 11: 外部パルス」に設定している状態
- 0 : COUNT PULSE SELを「00: 自軸の発生パルス」に設定している状態

COUNT PULSE SELの設定は、CWP, CCWP端子から出力するドライブパルスになります。

- ・ EXT PULSE = 1のときは、他軸のパルスまたは外部パルス信号を出力します。
- ・ EXT PULSE = 0のときは、自軸の発生パルスを出力します。

なお、コントローラドライバは、外部パルス出力機能はありません。

D13 : PAUSE(応用機能)

PAUSE信号によるSTBY = 1の状態を保持する機能が有効な状態を示します。

- 1 : STBY = 1の状態を保持する機能が有効な状態

PAUSE信号のアクティブレベルを検出すると、PAUSE = 1にします。

PAUSE信号のOFFレベルを検出すると、PAUSE = 0にします。

- ・ PAUSE = 1のときは、STBY = 1の状態を保持して、ドライブパルス出力の開始を保留します。

PAUSE信号は、以下のドライブ実行時のSTBY = 1で有効です。

- ・ パルス出力を伴うコマンド実行時のSTBY = 1
- ・ 予約コマンドによる連続ドライブ中の、パルス出力を伴うコマンド実行時のSTBY = 1
- ・ 外部パルス出力実行時のSTBY = 1

D14 : COMREG EP (応用機能)

コマンド予約機能の予約レジスタの格納状態を示します。

- 1 : 予約コマンドを格納していない状態 (EMPTY)
またはDRIVE STATUS1 PORTのERROR = 1の状態
- 0 : 1 命令以上の予約コマンドを格納している状態

D15 : COMREG FL (応用機能)

コマンド予約機能の予約レジスタの格納状態を示します。

- 1 : 10命令の予約コマンドを格納している状態 (FULL)
またはDRIVE STATUS1 PORTのERROR = 1の状態
- 0 : 9命令以下の予約コマンドを格納している状態

・ COMREG EPとCOMREG FLの関係は以下の状態を表します。

COMREG FL	COMREG EP	状態
0	0	予約レジスタに1~9命令のコマンドを格納している状態
0	1	予約レジスタが空の状態 (EMPTY)
1	0	予約レジスタに10命令分のコマンドを格納している状態 (FULL)
1	1	ERROR = 1の状態 / (リセット中の状態)

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 STATUS2 PORT読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスのDRIVE STATUS2 PORTを読み出します。
外部入出力信号の状態を表示するPORTです。

書式

C言語 `BOOL MC07_BRStatus2(DWORD hDev, WORD *pStatus, MC07_S_RESULT *psResult);`

VB `Function MC07_BRStatus2(ByVal hDev As Long, ByRef pStatus As Integer, ByRef psResult As MC07_S_RESULT)As Boolean`

VB.NET `Function MC07_BRStatus2(ByVal hDev As Integer, ByRef pStatus As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BRStatus2(uint hDev, ref ushort pStatus, ref MC07_S_RESULT psResult);`

引数

hDev … デバイスハンドルを指定します。
pStatus … 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 STATUS2 PORTの内容は、次に示す通りです。

D15	D14	D13	D12	D11	D10	D9	D8
DEND BUSY	DALM	DEND/PO	DRST/MF	0	NORG	ZORG	ORG
D7	D6	D5	D4	D3	D2	D1	D0
0	0	ORG SIGNAL	PULSE MASK	CCWLM	CWLM	FSSTOP	0

D1 : FSSTOP
FSSTOP信号の現在の入力状態を示します。
1 : アクティブレベル入力中の状態

D2 : CWLM
CWLM信号の現在の入力状態を示します。
1 : アクティブレベル入力中の状態

D3 : CCWLM
CCWLM信号の現在の入力状態を示します。
1 : アクティブレベル入力中の状態

D4 : PULSE MASK
SPEC INITIALIZE1コマンドのPULSE OUTPUT MASK = 1に設定している状態を示します。
1 : PULSE OUTPUT MASK = 1に設定している状態

D5 : ORG SIGNAL
ORIGIN SPEC SETコマンドのORG SIGNAL TYPEで設定しているORG検出信号です。
ORG検出信号の現在の入力状態を示します。
1 : アクティブレベル入力中の状態

D8 : ORG

ORG信号の現在の入力状態を示します。

1 : アクティブレベル入力中の状態

D9 : ZORG

±Z相信号の現在の入力状態を示します。

1 : アクティブレベル入力中の状態

D10 : NORG

NORG信号の現在の入力状態を示します。

1 : アクティブレベル入力中の状態

D12 : DRST/MF

DRST/MF信号の現在の出力状態を示します。

1 : アクティブレベル出力中の状態

D13 : DEND/PO

DEND/PO信号の現在の入力状態を示します。

1 : アクティブレベル入力中の状態

D14 : DALM

DALM信号の現在の入力状態を示します。

1 : アクティブレベル入力中の状態

- ・ SPEC INITIALIZE3コマンドでDALM入力信号を停止機能に設定することができます。
- ・ DALM入力信号のアクティブ論理を切り替えることができます。(応用機能)

D15 : DEND BUSY

SPEC INITIALIZE3コマンドのDEND信号を<サーボ対応>に設定している場合に有効です。

DEND信号のアクティブレベル検出待ちの状態を示します。

1 : パルス出力を完了 (DRIVE = 1 0) して、DEND信号のアクティブレベル検出待ちの状態

0 : DEND信号のアクティブレベルの検出でクリアします

即時停止指令を検出した場合は、強制終了と同時にクリアします

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 STATUS3 PORT読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスのDRIVE STATUS3 PORTを読み出します。
ステータス信号の状態を表示するPORTです。

書式

C言語 `BOOL MC07_BRStatus3(DWORD hDev, WORD *pStatus, MC07_S_RESULT *psResult);`

VB `Function MC07_BRStatus3(ByVal hDev As Long, ByRef pStatus As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BRStatus3(ByVal hDev As Integer, ByRef pStatus As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BRStatus3(uint hDev, ref ushort pStatus, ref MC07_S_RESULT psResult);`

引数

hDev …… デバイスハンドルを指定します。
pStatus …… 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 STATUS3 PORTの内容は、次に示す通りです。

D15	D14	D13	D12	D11	D10	D9	D8
(不定)	(不定)	(不定)	(不定)	(不定)	0	0	FSSTOP
D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	SOUT	0	0	0	(不定)

D4 : SOUT(応用機能)

1 : 選択したステータスがアクティブレベルを出力中であることを示します。

- ・ SOUT信号の対応有無については、各ユニットの取扱説明書をご覧ください。
- ・ SOUT信号にはHARD INITIALIZE1コマンドによりそれぞれカウンタ一致検出やドライブコントロールの状態など15のステータスから1つを選択して出力することができます。
- ・ SOUT信号はX軸とY軸のみステータス外部出力できます。
X軸のSOUTはSOUT0信号、Y軸のSOUTはSOUT1信号に割り付いています。

D8 : FSSTOP

1 : FSSTOP入力信号がアクティブレベルであることを示します。

- ・ DRIVE STATUS2 PORTのD1ビットと同じです。

D15--D11, D0 : (不定)

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 STATUS4 PORT読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスのDRIVE STATUS4 PORTを読み出します。
カウンタのコンパレータ出力状態とオーバーフローを表示するPORTです。

書式

C言語 `BOOL MC07_BRStatus4(DWORD hDev, WORD *pStatus, MC07_S_RESULT *psResult);`

VB `Function MC07_BRStatus4(ByVal hDev As Long, ByRef pStatus As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BRStatus4(ByVal hDev As Integer, ByRef pStatus As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BRStatus4(uint hDev, ref ushort pStatus, ref MC07_S_RESULT psResult);`

引数

hDev …… デバイスハンドルを指定します。
pStatus …… 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 STATUS4 PORTの内容は、次に示す通りです。

D15	D14	D13	D12	D11	D10	D9	D8
0	0	0	0	DFL OVF	DFLINT COMP3	DFLINT COMP2	DFLINT COMP1
D7	D6	D5	D4	D3	D2	D1	D0
PULSE OVF	CNTINT COMP3	CNTINT COMP2	CNTINT COMP1	ADDRESS OVF	ADRINT COMP3	ADRINT COMP2	ADRINT COMP1

D0 : ADRINT COMP1

D1 : ADRINT COMP2

D2 : ADRINT COMP3

アドレスカウンタの値がCOMPARE REGISTER1の検出条件と一致したことを示します。
アドレスカウンタの値がCOMPARE REGISTER2の検出条件と一致したことを示します。
アドレスカウンタの値がCOMPARE REGISTER3の検出条件と一致したことを示します。

1 : 検出条件が一致した状態

0 : クリア条件の入力でクリアします

検出条件およびクリア条件は、ADDRESS COUNTER INITIALIZE1, 2コマンドで設定します。

D3 : ADDRESS OVF

アドレスカウンタの値がオーバーフローしたことを示します。

1 : オーバフローした状態

0 : ADDRESS COUNTER PRESETコマンドまたはカウンタのクリア機能の実行でクリアします

D4 : CNTINT COMP1
D5 : CNTINT COMP2
D6 : CNTINT COMP3

パルスカウンタの値がCOMPARE REGISTER1の検出条件と一致したことを示します。
パルスカウンタの値がCOMPARE REGISTER2の検出条件と一致したことを示します。
パルスカウンタの値がCOMPARE REGISTER3の検出条件と一致したことを示します。

- 1 : 検出条件が一致した状態
- 0 : クリア条件の入力でクリアします

検出条件およびクリア条件は、PULSE COUNTER INITIALIZE1, 2コマンドで設定します。

D7 : PULSE OVF

パルスカウンタの値がオーバーフローしたことを示します。

- 1 : オーバフローした状態
- 0 : PULSE COUNTER PRESETコマンドまたはカウンタのクリア機能の実行でクリアします

D8 : DFLINT COMP1
D9 : DFLINT COMP2
D10 : DFLINT COMP3

パルス偏差カウンタの値または指定のデータ値がCOMPARE REGISTER1の検出条件と一致したことを示します。
パルス偏差カウンタの値または指定のデータ値がCOMPARE REGISTER2の検出条件と一致したことを示します。
パルス偏差カウンタの値または指定のデータ値がCOMPARE REGISTER3の検出条件と一致したことを示します。

- 1 : 検出条件が一致した状態
- 0 : クリア条件の入力でクリアします

検出条件(指定のデータ値)およびクリア条件は、DFL COUNTER INITIALIZE1, 2, 3コマンドで設定します。

D11 : DFL OVF

パルス偏差カウンタの値がオーバーフローしたことを示します。

- 1 : オーバフローした状態
- 0 : DFL COUNTER PRESETコマンドまたはカウンタのクリア機能の実行でクリアします

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 STATUS5 PORT読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスのDRIVE STATUS5 PORTを読み出します。
入力信号とドライブCHANGE指令の現在の状態を表示するPORTです。

書式

C言語 `BOOL MC07_BRStatus5(DWORD hDev, WORD *pStatus, MC07_S_RESULT *psResult);`

VB `Function MC07_BRStatus5(ByVal hDev As Long, ByRef pStatus As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BRStatus5(ByVal hDev As Integer, ByRef pStatus As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BRStatus5(uint hDev, ref ushort pStatus, ref MC07_S_RESULT psResult);`

引数

hDev … デバイスハンドルを指定します。
psResult … 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 STATUS5 PORTの内容は、次に示す通りです。

D15	D14	D13	D12	D11	D10	D9	D8
INDEX CSET	INDEX CBUSY	SPEED CSET	SPEED CBUSY	RATE CSET	CPP MASK	CPPOUT	CPPIN
D7	D6	D5	D4	D3	D2	D1	D0
EB1	EA1	EB0	EA0	0	0	(不定)	SS0

D0 : SS0(応用機能)

SS0信号の現在の入力状態を示します。

1 : アクティブレベル入力中の状態

- ・SS0信号による多用途センサ機能は、X軸とY軸のみ使用できます。
X軸のSS0は汎用入力IN0信号、Y軸のSS0は汎用入力IN1信号から操作することができます。

D4 : EA0

D5 : EB0

X軸とY軸の表示内容は同じです。

EA0, EB0信号の現在の入力状態を示します。

1 : ハイレベル入力中の状態

X軸,Z軸の場合、EA0,EB0に各軸±EA,±EB入力状態を示します。

なお、UC-766以外の製品は、±EA,±EB入力はありません。

D6 : EA1

D7 : EB1

X軸とY軸の表示内容は同じです。

EA1, EB1信号の現在の入力状態を示します。

1 : ハイレベル入力中の状態

Y軸,A軸の場合、EA1,EB1に各軸±EA,±EB入力状態を示します。

なお、UC-766以外の製品は、±EA,±EB入力はありません。

D8 : CPPIN

CPPIN信号の現在の入力状態を示します。

1 : ハイレベル入力中の状態

D9 : CPPOUT

CPPOUT信号の現在の出力状態を示します。

1 : ハイレベル出力中の状態

D10 : CPP MASK

CPPIN入力のマスク状態を示します。

1 : CPPIN入力のマスク状態がマスクしている状態

0 : DRIVE STATUS1 PORTのERROR = 1 0でクリアします

サブ軸補間ドライブのCPPINマスク機能が動作すると、CPP MASK = 1にします。

CPPIN入力は、X軸とY軸のCPP MASKのOR (論理和) でマスクします。

D11 : RATE CSET(応用機能)

1 : RATE CHANGE指令が待機中の状態を示します。

0 : RATE CHANGE指令なしの状態を示します。

待機中のCHANGE指令はスピード系ドライブCHANGE機能の変更動作点の検出で実行します。

RATE CHANGEコマンドはSPEED CBUSY=0を確認してから実行してください。

D12 : SPEED CBUSY(応用機能)

1 : スピード系ドライブCHANGEの実行処理中を示します。

0 : スピード系ドライブCHANGEの実行可能な状態を示します。

ドライブCHANGEコマンドはSPEED CBUSY=0を確認してから実行してください。

D13 : SPEED CSET(応用機能)

1 : スピード系ドライブCHANGE指令が待機中の状態を示します。

0 : スピード系ドライブCHANGE指令なしの状態を示します。

待機中のCHANGE指令は各CHANGE機能の変更動作点の検出で実行します。

ドライブCHANGE設定コマンドはSPEED CSET=0を確認してから実行してください。

D14 : INDEX CBUSY(応用機能)

1 : INDEX CHANGEコマンドの実行処理中を示します。

0 : INDEX CHANGEコマンドの実行可能な状態を示します。

INDEX CHANGEコマンドはINDEX CBUSY=0を確認してから実行してください。

D15 : INDEX CSET(応用機能)

1 : INDEX CHANGE指令が待機中の状態を示します。

0 : INDEX CHANGE指令なしの状態を示します。

待機中のCHANGE指令はINDEX CHANGE機能の変更動作点の検出で実行します。

INDEX CHANGE設定コマンドはINDEX CSET=0を確認してから実行してください。

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。

STATUS PORT バッファ読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスのDRIVE STATUS1 PORT、DRIVE STATUS2 PORT、DRIVE STATUS3 PORT、DRIVE STATUS4 PORT、DRIVE STATUS5 PORT、ORG STATUSの内容を読み出します。

書式

C言語 `BOOL MC07_BRStatusBuf(DWORD hDev, WORD StatusBuf[], MC07_S_RESULT *psResult);`

VB `Function MC07_BRStatusBuf(ByVal hDev As Long, ByRef StatusBuf As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BRStatusBuf(ByVal hDev As Integer, ByVal StatusBuf() As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BRStatusBuf(uint hDev, ushort[] StatusBuf, ref MC07_S_RESULT psResult);`

引数

hDev ... デバイスハンドルを指定します。

StatusBuf ... ステータスの配列(16要素、32バイトで領域確保されたもの)を指定します。
StatusBuf[*MC07_STATUS1*]... DRIVE STATUS1 PORTの内容が格納されます。
StatusBuf[*MC07_STATUS2*]... DRIVE STATUS2 PORTの内容が格納されます。
StatusBuf[*MC07_STATUS3*]... DRIVE STATUS3 PORTの内容が格納されます。
StatusBuf[*MC07_STATUS4*]... DRIVE STATUS4 PORTの内容が格納されます。
StatusBuf[*MC07_STATUS5*]... DRIVE STATUS5 PORTの内容が格納されます。
StatusBuf[*MC07_ORG_STATUS*]... ORG STATUSの内容が格納されます。

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。

- ・ステータスの詳細は、各STATUS PORTの読み出し関数をご覧ください。

DRIVE COMMAND 32ビット書き込み関数 / 読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE COMMAND PORTにデータ、コマンドを書き込み、DRIVE DATA1 PORT、DRIVE DATA2 PORTの内容を読み出します。

デバイス毎のアクセスを1回の関数実行で処理することができます。

- ・ 設定データの読み出し
- ・ 出力中のドライブ速度の読み出し
- ・ エラーステータスの読み出し
- ・ 各カウントデータの読み出し

書式

C言語 BOOL MC07_LWRDrive(DWORD *hDev*, WORD *Cmd*, DWORD **pWriteData*, DWORD **pReadData*, MC07_S_RESULT **psResult*);

VB Function MC07_LWRDrive(*hDev* As Long, *Cmd* As Integer, *pWriteData* As Long, *pReadData* As Long, *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_LWRDrive(*hDev* As Integer, *Cmd* As Short, *pWriteData* As Integer, *pReadData* As Integer, *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.LWRDrive(uint *hDev*, ushort *Cmd*, ref uint *pWriteData*, ref uint *pReadData*, ref MC07_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。
- pWriteData* ... 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。
- pReadData* ... 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。

4-5-4. WAIT 関数

MCC07 はコマンドの処理中またはドライブ実行中のとき、DRIVE STATUS1 PORT BUSY=1 になります。

また、MCC07 上でエラーがあるときは、DRIVE STATUS1 PORT ERROR=1 になります。

MCC07 に汎用コマンドを書き込むときは、DRIVE STATUS1 PORT 内の ERROR=0 および BUSY BIT=0 を DRIVE STATUS1 PORT 読み出し関数で確認してからコマンドを書き込みます。

また、汎用コマンドを書き込みした後に、終了待ちを行います。

WAIT 関数は、この汎用コマンドを書き込みした後の終了待ちするときに用いる関数です。

READY WAIT関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

機能

指定されたデバイスがREADY(DRIVE STATUS1 PORT BUSY 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。

書式

C言語 `BOOL MC07_BWaitDriveCommand(DWORD hDev, WORD WaitTime, MC07_S_RESULT *psResult);`

VB `Function MC07_BWaitDriveCommand(ByVal hDev As Long, ByVal WaitTime As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BWaitDriveCommand(ByVal hDev As Integer, ByVal WaitTime As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BWaitDriveCommand(uint hDev, ushort WaitTime, ref MC07_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READYになるまで無限に待機します。

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。

WAIT状態読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスで、次のWAIT関数のWAIT状態を返します。

- ・ REDY WAIT関数
- ・ COMREG NOT FULL WAIT関数

書式

C言語 `BOOL MC07_BIsWait(DWORD hDev, WORD *pWaitSts, MC07_S_RESULT *psResult);`

VB `Function MC07_BIsWait(ByVal hDev As Long, ByRef pWaitSts As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BIsWait(ByVal hDev As Integer, ByRef pWaitSts As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BIsWait(uint hDev, ref ushort pWaitSts, ref MC07_S_RESULT psResult);`

引数

hDev … デバイスハンドルを指定します。

pWaitSts … WAIT状態を格納するための変数のポインタを指定します。

格納される値	意味
0	WAIT関数の実行中ではありません。
1	WAIT関数の実行中です。

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。

WAIT中止関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

機 能

指定されたデバイスで、次のWAIT関数の実行を中止します。

- ・ REDY WAIT関数
- ・ COMREG NOT FULL WAIT関数

書 式

C言語 `BOOL MC07_BBreakWait(DWORD hDev, MC07_S_RESULT *psResult);`

VB `Function MC07_BBreakWait(ByVal hDev As Long, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BBreakWait(ByVal hDev As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BBreakWait(uint hDev, ref MC07_S_RESULT psResult);`

引 数

hDev …… デバイスハンドルを指定します。

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。

4-5-5. SPEED・RATE 関数

SPEED パラメータ、加減速時定数(RATE)を一括で設定します。

- ・ SPEED パラメータは、加減速ドライブに必要な速度パラメータおよび第1パルスの速度を 1Hz 単位で設定します。
- ・ 加減速時定数(RATE)は、加減速ドライブに必要な加減速時定数(ms/kHz)を RATE TABLE 表から選択し設定します。

SPEED・RATE構造体

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

説明

SPEEDパラメータ、RATEを格納します。

書式

<p>C言語</p> <pre>typedef struct _MC07_S_SPEED_RATE { DWORD FSpd; DWORD HighSpeed; DWORD LowSpeed; DWORD EndLowSpeed; DWORD SUAarea; DWORD SDArea; DWORD SUH; DWORD SDH; DWORD URateNo; DWORD DRateNo; } MC07_S_SPEED_RATE;</pre>	<p>VB</p>	<p>Type MC07_S_SPEED_RATE</p> <pre>FSpd As Long HighSpeed As Long LowSpeed As Long EndLowSpeed As Long SUAarea As Long SDArea As Long SUH As Long SDH As Long URateNo As Long DRateNo As Long</pre> <p>End Type</p>
<p>VB.NET</p> <pre>Structure MC07_S_SPEED_RATE { Public FSpd As Integer Public HighSpeed As Integer Public LowSpeed As Integer Public EndLowSpeed As Integer Public SUAarea As Integer Public SDArea As Integer Public SUH As Integer Public SDH As Integer Public URateNo As Integer Public DRateNo As Integer End Structure</pre>	<p>C#.NET</p>	<pre>struct MC07_S_SPEED_RATE { public int FSpd; public int HighSpeed; public int LowSpeed; public int EndLowSpeed; public int SUAarea; public int SDArea; public int SUH; public int SDH; public int URateNo; public int DRateNo; }</pre>

メンバ

<i>FSpd</i>	… 第一パルスのパルス速度 (× 1Hz) 電源投入時の初期値は、5,000(5000Hz)です。
<i>HighSpeed</i>	… 最高速度 (× 1Hz) 電源投入時の初期値は、3,000(3000Hz)です。
<i>LowSpeed</i>	… 開始速度 (× 1Hz) 電源投入時の初期値は、300(300Hz)です。
<i>EndLowSpeed</i>	… 終了速度 (× 1Hz) 電源投入時の初期値は、0(LowSpeedと同じ)です。
<i>SUAarea</i>	… SUAREA (× 1Hz) 電源投入時の初期値は、0(SUAREAの変速領域なし)です。
<i>SDArea</i>	… SDAREA (× 1Hz) 電源投入時の初期値は、0(SDAREAの変速領域なし)です。

- SUH* …… 0に設定してください。
- SDH* …… 0に設定してください。
- URateNo* …… URATE No. (6-1-2. 章ドライブパラメータ RATEテーブル表のRATE No.を参照してください。) 電源投入時の初期値は、7(No.7)です。
- DRateNo* …… DRATE No. (6-1-2. 章ドライブパラメータ RATEテーブル表のRATE No.を参照してください。) 電源投入時の初期値は、7(No.7)です。

SPEED・RATE読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスからSPEEDパラメータ設定と加減速時定数(RATE)設定の値を読み出します。

書式

C言語 BOOL MC07_ReadSpeedRate(DWORD *hDev*, WORD **pResolNo*, MC07_S_SPEED_RATE **psSpeedRate*, MC07_S_RESULT **psResult*);

VB Function MC07_ReadSpeedRate(ByVal *hDev* As Long, ByRef *pResolNo* As Integer, ByRef *psSpeedRate* As MC07_S_SPEED_RATE, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_ReadSpeedRate(ByVal *hDev* As Integer, ByRef *pResolNo* As Short, ByRef *psSpeedRate* As MC07_S_SPEED_RATE, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.ReadSpeedRate(uint *hDev*, ref ushort *pResolNo*, ref MC07_S_SPEED_RATE *psSpeedRate*, ref MC07_S_RESULT *psResult*);

引数

hDev … デバイスハンドルを指定します。
pResolNo … 読み出したRESOL No. を格納するための変数のポインタを指定します。
psSpeedRate … 読み出したSPEEDパラメータと加減速時定数(RATE)を格納するためのSPEED・RATE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。

- ・SPEED・RATE構造体のメンバFspd, HighSpeed, LowSpeed, EndLowSpeed, SUArea, SDAreaは、1Hz未満の速度を切り捨てて格納します。
- ・U/D CYCLEがRATEテーブル表に存在しない場合、SPEED・RATE構造体のメンバURATE No., DRATE No. は隣接する2つのRATE No.のうち、大きい側に補正されます。

4-5-6. 補間ドライブ関数

補間ドライブの演算と制御を行います。

次の補間ドライブ関数を用意しています。

- ・ 2 軸相対アドレス直線補間ドライブ関数
- ・ 2 軸相対アドレス円弧補間ドライブ関数
- ・ メインチップ 2 軸相対アドレス直線補間ドライブ関数
- ・ メインチップ 2 軸相対アドレス円弧補間ドライブ関数

また、次の補間演算関数を用意しています。

- ・ 円の中心点ゲット関数
- ・ 相対アドレス変換関数

POSITION構造体

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

説明

補間ドライブのX・Y座標を格納します。

書式

C言語 `typedef struct _MC07_S_XY_POSITION {
 LONG X;
 LONG Y;
 } MC07_S_XY_POSITION;`

VB `Type MC07_S_XY_POSITION
 X As Long
 Y As Long
 End Type`

VB.NET `Structure MC07_S_XY_POSITION
 Public X As Integer
 Public Y As Integer
 End Structure`

C#.NET `struct MC07_S_XY_POSITION
 {
 public int X;
 public int Y;
 }`

メンバ

X …… X座標
Y …… Y座標

2軸相対アドレス直線補間ドライブ関数

UsbC

UC-766

機能

指定された2つのデバイスで、指定された目的地までの任意2軸直線補間ドライブを行います。
この関数を実行する前に、以下を確認してください。

- 両デバイスを含む全てのチップ上の全ての軸でSTATUS1 PORTのERROR = 0, BUSY = 0
(コマンド予約機能を使用する場合は、STATUS1 PORTのERROR = 0, COMREG FL = 0)

書式

C言語 BOOL MC07_IncStrCp(DWORD *hDevX*, DWORD *hDevY*, WORD *DrvSpec*,
MC07_S_XY_POSITION **psTargetPosition*, MC07_S_RESULT **psResult*);

VB Function MC07_IncStrCp(ByVal *hDevX* As Long, ByVal *hDevY* As Long,
ByVal *DrvSpec* As Integer, ByRef *psTargetPosition* As MC07_S_XY_POSITION,
ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_IncStrCp(ByVal *hDevX* As Integer, ByVal *hDevY* As Integer,
ByVal *DrvSpec* As Short, ByRef *psTargetPosition* As MC07_S_XY_POSITION,
ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.IncStrCp(uint *hDevX*, uint *hDevY*, ushort *DrvSpec*,
ref MC07_S_XY_POSITION *psTargetPosition*, ref MC07_S_RESULT *psResult*);

引数

- hDevX* ... X座標アドレスの補間パルスを出力する軸のデバイスハンドルを指定します。
hDevY ... Y座標アドレスの補間パルスを出力する軸のデバイスハンドルを指定します。
*hDevX*と*hDevY*には、同じコントローラボード上の異なる軸を指定します。
DrvSpec ... ドライブ仕様を指定します。

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	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。

psTargetPosition ... 目的地のX・Y座標が格納されているPOSITION構造体のポイントを指定します。
(指定できる値の範囲はX座標・Y座標共に-2,147,483,648~2,147,483,647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。

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

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。

補間ドライブ中、メイン軸のCPP STOP機能、サブ軸のCPPINマスク機能は有効になります。メイン軸がCPP STOP機能でドライブを停止した場合は、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。

コマンド予約機能を使用し、任意2軸直線補間を行う場合、使用方法に制限があります。制限については、5-3. 補間ドライブを参照して下さい。

MCC07搭載製品では、ユニット内の補間を行わないチップについては、CP SPEC SETコマンドのCPPOUT SELは「CPPIN端子から入力するパルス」に設定されている必要があります。

MCC07搭載製品では、2軸相対アドレス直線補間ドライブ関数または2軸相対アドレス円弧補間ドライブ関数による補間ドライブを、同じユニット内で複数同時に実行することはできません。

この関数を実行することにより、MCC07の次のコマンドの設定値が更新されます。

- ・ CP SPEC SETコマンド (指定された両軸が属するそれぞれのMCC07)
- ・ LONG POSITION SETコマンド (指定された両軸)
- ・ SHORT POSITION SETコマンド (指定された両軸)

2軸相対アドレス円弧補間ドライブ関数

UsbC

UC-766

機能

指定された2つのデバイスで、指定された目的地までの中心点指定の任意2軸円弧補間ドライブを行います。この関数を実行する前に、以下を確認してください。

- 両デバイスを含む全てのチップ上の全ての軸でSTATUS1 PORTのERROR = 0, BUSY = 0 (コマンド予約機能を使用する場合は、STATUS1 PORTのERROR = 0, COMREG FL = 0)

書式

C言語 BOOL MC07_IncCirCp(DWORD *hDevX*, DWORD *hDevY*, WORD *DrvSpec*, WORD *Dir*, MC07_S_XY_POSITION **psCenterPosition*, MC07_S_XY_POSITION **psTargetPosition*, MC07_S_RESULT **psResult*) As Boolean

VB Function MC07_IncCirCp(ByVal *hDevX* As Long, ByVal *hDevY* As Long, ByVal *DrvSpec* As Integer, ByVal *Dir* As Integer, ByRef *psCenterPosition* As MC07_S_XY_POSITION, ByRef *psTargetPosition* As MC07_S_XY_POSITION, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_IncCirCp(ByVal *hDevX* As Integer, ByVal *hDevY* As Integer, ByVal *DrvSpec* As Short, ByVal *Dir* As Short, ByRef *psCenterPosition* As MC07_S_XY_POSITION, ByRef *psTargetPosition* As MC07_S_XY_POSITION, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.IncCirCp(uint *hDevX*, uint *hDevY*, ushort *DrvSpec*, ushort *Dir*, ref MC07_S_XY_POSITION *psCenterPosition*, ref MC07_S_XY_POSITION *psTargetPosition*, ref MC07_S_RESULT *psResult*);

引数

- hDevX* ... X座標アドレスの補間パルスを出力する軸のデバイスハンドルを指定します。
hDevY ... Y座標アドレスの補間パルスを出力する軸のデバイスハンドルを指定します。
*hDevX*と*hDevY*には、同じコントローラボード上の異なる軸を指定します。
DrvSpec ... ドライブ仕様を指定します。

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	CONST CP ENABLE	DRIVE MODE

- Dir* ... 回転方向を指定します。

指定できる値	意味
MC07_CCW	-(CCW)方向
MC07_CW	+(CW)方向

- psCenterPosition* ... 中心点のX・Y相対座標が格納されているPOSITION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座標共に-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POSITION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座標共に-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。
- 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。

補間ドライブ中、メイン軸のCPP STOP機能、サブ軸のCPPINマスク機能は有効になります。
メイン軸がCPP STOP機能でドライブを停止した場合は、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。

コマンド予約機能を使用し、任意2軸円弧補間を行う場合、使用方法に制限があります。
制限については、5-3. 補間ドライブを参照して下さい。

MCC07搭載製品では、2軸相対アドレス円弧補間ドライブ関数による補間ドライブを、同じユニット内で
複数同時に実行することはできません。

この関数を実行することにより、MCC07の次のコマンドの設定値が更新されます。

- ・ CP SPEC SETコマンド (指定された両軸が属するそれぞれのMCC07)
- ・ CIRCULAR XPOSITION SETコマンド (指定された両軸)
- ・ CIRCULAR YPOSITION SETコマンド (指定された両軸)
- ・ CIRCULAR PULSE SETコマンド (指定された両軸)

メインチップ2軸相対アドレス直線補間ドライブ関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定された相関 2 軸で、指定された相対アドレスの目的地まで2軸直線補間ドライブを実行します。
この関数を実行する前に、以下を確認して下さい。

- ・両デバイスのSTATUS1 PORTのERROR = 0, BUSY = 0
(コマンド予約機能を使用する場合は、STATUS1 PORTのERROR = 0, COMREG FL = 0)

書式

C言語 BOOL MC07_McIncStrCp(DWORD *hDevX*, DWORD *hDevY*, WORD *DrvSpec*,
MC07_S_XY_POSITION **psTargetPosition*, MC07_S_RESULT **psResult*);

VB Function MC07_McIncStrCp(ByVal *hDevX* As Long, ByVal *hDevY* As Long,
ByVal *DrvSpec* As Integer, ByRef *psTargetPosition* As MC07_S_XY_POSITION,
ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_McIncStrCp(ByVal *hDevX* As Integer, ByVal *hDevY* As Integer,
ByVal *DrvSpec* As Short, ByRef *psTargetPosition* As MC07_S_XY_POSITION,
ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.McIncStrCp(uint *hDevX*, uint *hDevY*, ushort *DrvSpec*,
ref MC07_S_XY_POSITION *psTargetPosition*, ref MC07_S_RESULT *psResult*);

引数

hDevX ... X/Z軸のデバイスハンドルを指定します。
hDevY ... Y/A軸のデバイスハンドルを指定します。

UC-766		UCD-7610v1 / UCD-7613v1	
<i>hDevX</i>	<i>hDevY</i>	<i>hDevX</i>	<i>hDevY</i>
X軸	Y軸	X軸	Y軸
Z軸	A軸		

DrvSpec ... ドライブ仕様を指定します。

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	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。

psTargetPosition ... 目的地のX・Y座標 (-2,147,483,648 ~ +2,147,483,647) が格納されている POSITION構造体のポインタを指定します。
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。

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。

この関数を実行することにより、MCC07の次のコマンドの設定値が更新されます。

- ・ LONG POSITION SETコマンド (指定された両軸)
- ・ SHORT POSITION SETコマンド (指定された両軸)

メインチップ2軸相対アドレス円弧補間ドライブ関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定された相関2軸で、指定された相対アドレスの目的地まで2軸円弧補間ドライブを実行します。
この関数を実行する前に、以下を確認して下さい。

- 両デバイスのSTATUS1 PORTのERROR = 0, BUSY = 0
(コマンド予約機能を使用する場合は、STATUS1 PORTのERROR = 0, COMREG FL = 0)

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 MC07_McIncCirCp(DWORD *hDevX*, DWORD *hDevY*, WORD *DrvSpec*, WORD *Dir*,
MC07_S_XY_POSITION **psCenterPosition*,
MC07_S_XY_POSITION **psTargetPosition*, MC07_S_RESULT **psResult*) As Boolean

VB Function MC07_McIncCirCp(ByVal *hDevX* As Long, ByVal *hDevY* As Long, ByVal *DrvSpec* As Integer,
ByVal *Dir* As Integer, ByRef *psCenterPosition* As MC07_S_XY_POSITION,
ByRef *psTargetPosition* As MC07_S_XY_POSITION,
ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_McIncCirCp(ByVal *hDevX* As Integer, ByVal *hDevY* As Integer,
ByVal *DrvSpec* As Short, ByVal *Dir* As Short,
ByRef *psCenterPosition* As MC07_S_XY_POSITION,
ByRef *psTargetPosition* As MC07_S_XY_POSITION,
ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.McIncCirCp(uint *hDevX*, uint *hDevY*, ushort *DrvSpec*, ushort *Dir*,
ref MC07_S_XY_POSITION *psCenterPosition*, ref MC07_S_XY_POSITION *psTargetPosition*,
ref MC07_S_RESULT *psResult*);

引数

- hDevX* ... X/Z軸のデバイスハンドルを指定します。
hDevY ... Y/A軸のデバイスハンドルを指定します。

UC-766		UCD-7610v1 / UCD-7613v1	
<i>hDevX</i>	<i>hDevY</i>	<i>hDevX</i>	<i>hDevY</i>
X軸	Y軸	X軸	Y軸
Z軸	A軸		

- DrvSpec* ... ドライブ仕様を指定します。

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	1	0	0	0	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。

D4 : 0

D5 : 1

D4とD5には必ず0と1を指定してください。
他の指定をした場合は、正しい円弧になりません。

Dir ... 回転方向を指定します。

指定できる値	意味
MC07_CCW	-(CCW)方向
MC07_CW	+(CW)方向

psCenterPosition ... 中心点のX・Y相対座標 (-8,388,607 ~ +8,388,607)が格納されているPOSITION構造体のポイントを指定します。

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 (-16,777,214 ~ +16,777,214)が格納されているPOSITION構造体のポイントを指定します。

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。

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

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。

【注意】

線速一定制御を「有効」にして、円弧補間ドライブを実行するときは、

下記のように円弧補間ドライブの終了処理を実行してください。

円弧補間ドライブの終了処理が行われないと、円弧補間ドライブの後に直線補間ドライブを実行したとき、設定速度が出力されないことがあります。

- ・ CIRCULAR XPOSITION SETコマンド (H'28) : H'00_0000に設定
 - ・ CIRCULAR YPOSITION SETコマンド (H'29) : H'00_0000に設定
 - ・ CIRCULAR PULSE SETコマンド (H'2A) : H'0000_0000に設定
 - ・ メイン軸円弧補間ドライブ (H'3A) : DATA1=H'0001で実行
- } 円弧補間ドライブ
(0パルス、終了位置0°)

*上記の各コマンドは応用機能編をご覧ください。

*円弧補間ドライブの終了処理で動作することはありません。

円の中心点ゲット関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

円弧の通過点と目的地から中心点と回転方向を求めます。

注：次の場合、関数がエラー終了します。

- ・通過点相対アドレスまたは目的地相対アドレスが(0, 0)の場合
- ・通過点相対アドレスと目的地相対アドレスが同一の場合

書式

```
C言語  BOOL MC07_GetCirCenterPosition(MC07_S_XY_POSITION *psPassPosition,
                                     MC07_S_XY_POSITION *psTargetPosition, WORD *pDir,
                                     MC07_S_XY_POSITION *psCenterPosition, MC07_S_RESULT *psResult );
```

```
VB      Function MC07_GetCirCenterPosition(ByRef psPassPosition As MC07_S_XY_POSITION,
                                           ByRef psTargetPosition As MC07_S_XY_POSITION, ByRef pDir As Integer,
                                           ByRef psCenterPosition As MC07_S_XY_POSITION,
                                           ByRef psResult As MC07_S_RESULT) As Boolean
```

```
VB.NET  Function MC07_GetCirCenterPosition(ByRef psPassPosition As MC07_S_XY_POSITION,
                                           ByRef psTargetPosition As MC07_S_XY_POSITION, ByRef pDir As Short,
                                           ByRef psCenterPosition As MC07_S_XY_POSITION,
                                           ByRef psResult As MC07_S_RESULT) As Boolean
```

```
C#.NET  bool MC07.GetCirCenterPosition(ref MC07_S_XY_POSITION psPassPosition,
                                       ref MC07_S_XY_POSITION psTargetPosition, ref ushort pDir,
                                       ref MC07_S_XY_POSITION psCenterPosition,
                                       ref MC07_S_RESULT psResult);
```

引数

- psPassPosition* ... 通過点のX・Y相対座標が格納されているPOSITION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。)
通過点の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POSITION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。
- pDir* ... 円弧の回転方向が格納される変数のポインタです。

格納される値	意味
MC07_CCW	-(CCW)方向
MC07_CW	+(CW)方向

- psCenterPosition* ... 中心点のX・Y相対座標を格納するためのPOSITION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。

- 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。

4-5-7. ORIGIN 関数

ORIGIN ドライブは、MCC07 が持っている ORIGIN ドライブのセンサー検出機能を組み合わせ、コントローラが自動的にセンサー検出工程を順次行って、機械原点検出を完了させるドライブです。

ORIGIN ドライブには、ORG-0 ~ 5, 10, 11, 12 の 9 種類のドライブ型式があります。

ORIGIN ドライブ機能の詳細は、6-1-4. 章「ORIGIN ドライブ」をご覧ください。

ORIGIN ドライブ パラメータ構造体

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

説明

ORIGIN ドライブ パラメータを格納します。

ORIGIN ドライブパラメータについては、ORIGIN ドライブパラメータ読み出し関数をご覧ください。

書式

<p>C言語</p> <pre>typedef struct _MC07_S_ORG_PARAM { DWORD Spec; DWORD MarginPulse; DWORD LimitDelay; DWORD ScanDelay; DWORD PulseDelay; DWORD CScanErrorPulse; DWORD PulseErrorPulse; DWORD OffsetPulse; LONG PresetPulse; } MC07_S_ORG_PARAM;</pre>	<p>VB</p> <pre>Type MC07_S_ORG_PARAM Spec As Long MarginPulse As Long LimitDelay As Long ScanDelay As Long PulseDelay As Long CScanErrorPulse As Long PulseErrorPulse As Long OffsetPulse As Long PresetPulse As Long End Type</pre>
<p>VB.NET</p> <pre>Structure MC07_S_ORG_PARAM Public Spec As Integer Public MarginPulse As Integer Public LimitDelay As Integer Public ScanDelay As Integer Public PulseDelay As Integer Public CScanErrorPulse As Integer Public PulseErrorPulse As Integer Public OffsetPulse As Integer Public PresetPulse As Integer End Structure</pre>	<p>C#.NET</p> <pre>struct MC07_S_ORG_PARAM { public uint Spec; public uint MarginPulse; public uint LimitDelay; public uint ScanDelay; public uint PulseDelay; public uint CScanErrorPulse; public uint PulseErrorPulse; public uint OffsetPulse; public int PresetPulse; }</pre>

メンバ

<p><i>Spec</i></p> <p><i>MarginPulse</i></p> <p><i>LimitDelay</i></p> <p><i>ScanDelay</i></p> <p><i>PulseDelay</i></p> <p><i>CScanErrorPulse</i></p> <p><i>PulseErrorPulse</i></p> <p><i>OffsetPulse</i></p> <p><i>PresetPulse</i></p>	<p>… ORIGIN ドライブの動作仕様を格納します。</p> <p>… MARGIN PULSE数 (0 ~ 65,535) を格納します。</p> <p>… LIMIT DELAY TIME (× 5ms) を格納します。(0 ~ 255: 0 ~ 1,275ms)</p> <p>… SCAN DELAY TIME (× 5ms) を格納します。(0 ~ 255: 0 ~ 1,275ms)</p> <p>… PULSE DELAY TIME (× 5ms) を格納します。(0 ~ 255: 0 ~ 1,275ms)</p> <p>… CONSTANT SCAN工程時にエラー判定する最大PULSE数 (0 ~ 2,147,483,647) を格納します。</p> <p>… 1PULSE送り工程時にエラー判定する最大PULSE数 (0 ~ 2,147,483,647) を格納します。</p> <p>… 機械原点近傍アドレスのOFFSET PULSE数 (0 ~ 2,147,483,647) を格納します。</p> <p>… 機械原点位置からのPRESET PULSE数 (-2,147,483,648 ~ 2,147,483,647) を格納します。</p>
--	---

ORIGINドライブステータス読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスのORIGIN STATUSの内容を読み出します。
STATUS PORT ALL読み出し関数では、各DRIVE STATUSと一緒にORIGIN STATUSを読み出しすることもできます。

書式

C言語 `BOOL MC07_ReadOrgStatus(DWORD hDev, WORD *pStatus, MC07_S_RESULT *psResult);`

VB `Function MC07_ReadOrgStatus(ByVal hDev As Long, ByRef pStatus As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_ReadOrgStatus(ByVal hDev As Integer, ByRef pStatus As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.ReadOrgStatus(uint hDev, ref ushort pStatus, ref MC07_S_RESULT psResult);`

引数

hDev … デバイスハンドルを指定します。
pStatus … 読み出したORIGIN STATUSの内容が格納される変数のポインタを指定します。
ORIGIN STATUSの内容は、次に示す通りです。

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
FSEND	SSEND	LSEND	ORIGIN ERROR	0	0	0	ORIGIN FLAG

- D0 : ORIGIN FLAG
ORIGINドライブの機械原点アドレスの記憶状態を示します。
1: 機械原点の絶対アドレスを記憶している状態
0: 機械原点の絶対アドレスを記憶していない状態
- D4 : ORIGIN ERROR
SENSOR ERROR、ERROR PULSE ERROR、ADDRESS ERRORのいずれかを検出したことを示します。
1: エラーが発生した状態
0: 動作エラークリア関数を実行してクリアします。
- D5 : LSEND
ORIGINドライブ中にLIMIT減速停止指令またはLIMIT即時停止指令を検出したことを示します。
1: LIMIT減速停止指令またはLIMIT即時停止指令を検出した状態
0: 次のORIGINドライブの実行でクリアされます。
- ・LIMIT減速停止指令
入力機能をLIMIT減速停止に設定したCWLM、CCWLM信号
 - ・LIMIT即時停止指令
入力機能をLIMIT即時停止に設定したCWLM、CCWLM信号

- D6 : SSEND
ORIGINドライブ中に減速停止指令を検出したことを示します。
1:減速停止指令を検出した状態
0:次のORIGINドライブの実行でクリアされます。
- ・減速停止指令
SLOW STOPコマンド
入力機能を減速停止に設定したDALM信号
- D7 : FSEND
ORIGINドライブ中に即時停止指令を検出したことを示します。
1:即時停止指令を検出した状態
0:次のORIGINドライブの実行でクリアされます。
- ・即時停止指令
FAST STOPコマンド
入力機能を即時停止に設定したDALM信号
FSSTOP信号
- D13 : SENSOR ERROR
ORIGINドライブ中にSENSOR ERRORを検出したことを示します。
1:SENSOR ERRORを検出した状態
0:動作エラークリア関数を実行してクリアします。注1.
- D14 : ERROR PULSE ERROR
ORIGINドライブ中にERROR PULSE ERROR検出機能でERROR PULSE ERRORを検出したことを示します。
1:ERROR PULSE ERRORを検出した状態
0:動作エラークリア関数を実行してクリアします。注1.
- D15 : ADDRESS ERROR
ORIGINドライブの機械原点近傍ADDRESSの計算結果が-2,147,483,647~2,147,483,647の範囲を越えていることを示します。
1:ADDRESS ERRORを検出した状態
0:動作エラークリア関数を実行してクリアします。注1.

注1.エラーを検出した場合は、必ず動作エラークリア関数を実行してエラーをクリアしてください。
エラーがクリアされていない場合、ORIGINドライブ関数の実行、およびMCC07に汎用コマンドを書き込むことができません。

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。

ORIGIN SPEC SET関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

ORIGINドライブの動作仕様を設定します。
この関数を実行する前にDRIVE STATUS1 PORT ERROR=0、DRIVE STATUS1 PORT BUSY=0を確認してください。

書式

C言語 BOOL MC07_SetOrgSpec(DWORD *hDev*, WORD *Spec*, MC07_S_RESULT **psResult*);

VB Function MC07_SetOrgSpec(ByVal *hDev* As Long, ByVal *Spec* As Integer, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_SetOrgSpec(ByVal *hDev* As Integer, ByVal *Spec* As Short, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.SetOrgSpec(uint *hDev*, ushort *Spec*, ref MC07_S_RESULT *psResult*);

引数

- hDev* ... デバイスハンドルを指定します。
- Spec* ... ORIGINドライブの動作仕様を指定します。

D15	D14	D13	D12	D11	D10	D9	D8
NORG SIGNAL TYPE3	NORG SIGNAL TYPE2	NORG SIGNAL TYPE1	NORG SIGNAL TYPE0	ORG SIGNAL TYPE3	ORG SIGNAL TYPE2	ORG SIGNAL TYPE1	ORG SIGNAL TYPE0
D7	D6	D5	D4	D3	D2	D1	D0
SCAN MARGIN ENABLE	AUTO DRST ENABLE	ERROR PULSE ERROR ENABLE	0	SENSOR ERROR TYPE1	SENSOR ERROR TYPE0	PULSE SENSOR TYPE	ORIGIN START DIR

電源投入時の初期値は、H'8000アンダーライン側です。

D0 : ORIGIN START DIR
ORIGINドライブの起動方向を選択します。

- 0 : -(CCW)方向に起動する。
- 1 : +(CW)方向に起動する。

D1 : PULSE SENSOR TYPE
最終工程となる1PULSE送り工程での機械原点信号の検出方法を選択します。

- 0 : 機械原点信号のエッジを検出して工程を終了する。
- 1 : 機械原点信号のレベルを検出して工程を終了する。

D2 : SENSOR ERROR TYPE0

D3 : SENSOR ERROR TYPE1

機械原点信号のレベルエラー発生時の動作仕様を選択します。
CONSTANT SCAN工程終了後のDELAY TIME経過後に機械原点信号のレベルをチェックします。
信号のレベルが検出時のレベルと異なる場合には、選択した動作仕様を実行します。

TYPE0	TYPE1	機械原点信号のレベルエラー発生時の動作仕様
0	0	ORIGINドライブをエラー終了する。
0	1	現在位置からCONSTANT SCAN工程を開始する。
1	0	現在位置からSCAN工程を開始する。
1	1	レベルエラーを無視して次の工程に進む。*1

*1 原点センサに検出幅が狭い Z 相を用いる場合、レベルエラーになる場合があります。
このようなときは、「レベルエラーを無視して次工程に進む」の設定にしてください。

ORIGINドライブ型式により、レベルをチェックする機械原点信号が異なります。

・ORG0～5では、ORG SIGNAL TYPEで選択したORG検出信号

・ORG-11,12では、CWLMまたはCCWLM信号

最終工程終了時とORG-10では、レベルチェックによるエラー判定を行いません。

D4 : 0

D5 : ERROR PULSE ERROR ENABLE

ERROR PULSE ERROR検出機能を『有効にする/無効にする』を選択します。

0 : ERROR PULSE ERROR検出機能を無効にする。

1 : ERROR PULSE ERROR検出機能を有効にする。

ERROR PULSE ERROR検出機能

CONSTANT SCAN工程および、1PULSE送り工程実行中に検出信号を検出できずに出力パルス数がエラー判定するパルス数に達したらORIGINドライブを強制終了します。

エラー判定するパルス数、ORIGIN ERROR PULSE SET関数で設定します。

コントローラ内でERROR PULSE ERROR検出のためにMCC07のPULSE COUNTERを使用します。

このため、ユーザアプリケーションでMCC07のPULSE COUNTERは使用できなくなりますので御注意ください。

D6 : AUTO DRST ENABLE

SPEC INITIALIZE3 COMMANDでDRST信号を<サーボ対応>に設定している場合に有効です。
機械原点信号の検出完了時に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。(10ms間アクティブレベルにする)

AUTO DRST ENABLE=1のときは、SPEC INITIALIZE3 COMMANDでDEND信号を<サーボ対応>に設定している場合でも、最終工程となるCONSTANT SCAN工程、または1PULSE送り工程ではDEND信号の確認を行いません。

D7 : SCAN MARGIN ENABLE

SCAN工程時にMARGIN PULSEを入れる/入れないを選択します。

0 : SCAN工程時にMARGIN PULSEをいれない。

1 : SCAN工程時にMARGIN PULSEを入れる。

D8 : ORG SIGNAL TYPE0

D9 : ORG SIGNAL TYPE1

D10: ORG SIGNAL TYPE2

D11: ORG SIGNAL TYPE3

ORG SIGNAL TYPE				
TYPE3	TYPE2	TYPE1	TYPE0	ORG検出信号
0	0	0	0	ORG信号(出荷時設定)
0	0	0	1	± ZORG信号
0	0	1	0	ORG信号と± ZORG信号のAND
0	0	1	1	ORG信号と± ZORG信号のOR
0	1	0	1	PO信号
0	1	1	0	ORG信号とPO信号のAND
0	1	1	1	ORG信号とPO信号のOR
1	X	X	X	設定禁止

D12: NORG SIGNAL TYPE0
 D13: NORG SIGNAL TYPE1
 D14: NORG SIGNAL TYPE2
 D15: NORG SIGNAL TYPE3

NORG SIGNAL TYPE				NORG検出信号
TYPE3	TYPE2	TYPE1	TYPE0	
0	X	X	X	設定禁止
1	0	0	0	NORG信号(出荷時設定)
1	0	0	1	± ZORG信号
1	0	1	0	NORG信号と± ZORG信号のAND
1	0	1	1	NORG信号と± ZORG信号のOR
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

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。

ORIGIN MARGIN PULSE SET関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスに、MARGINパルス数を設定します。
SCAN工程およびCSCAN工程時に挿入するMARGINパルスのパルス数です。

SCAN工程では、ORIGINドライブの動作仕様のSCAN MARGIN ENABLE=1のときに有効です。
ORIGIN SPEC SET関数で設定します。

NORG検出工程およびORIGINドライブの最終工程では、MARGIN PULSEを挿入しません。
CONSTANT SCAN工程では、機械原点信号を検出すると進行方向へMARGINパルス分の進入を行ってから停止します。
SCAN工程では機械原点信号を検出し、ドライブを減速停止した後、MARGINパルス数分の進入を行います。
電源投入時の初期値は、5パルスです。

この関数を実行する前にDRIVE STATUS1 PORT ERROR=0、DRIVE STATUS1 PORT BUSY=0を確認してください。

書式

C言語 BOOL MC07_SetOrgMarginPulse(DWORD *hDev*, DWORD *MarginPulse*, MC07_S_RESULT **psResult*);

VB Function MC07_SetOrgMarginPulse(ByVal *hDev* As Long, ByVal *MarginPulse* As Long,
ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_SetOrgMarginPulse(ByVal *hDev* As Integer, ByVal *MarginPulse* As Integer,
ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.SetOrgMarginPulse(uint *hDev*, uint *MarginPulse*, ref MC07_S_RESULT *psResult*);

引数

hDev ... デバイスハンドルを指定します。
MarginPulse ... MARGINパルス (0 ~ 65,535パルス)を設定します。
65,535パルスより大きい値が設定された場合は、65,535パルスに補正されます。
また、MARGINパルスは、必ずCONSTANT SCAN工程時にエラー判定する最大パルス数
(ORIGIN ERROR PULSE SET関数を参照)より、2以上小さいパルスに設定してください。
小さくない場合は、ORIGINドライブ実行時にエラー判定する最大パルス -1がMARGINパルス
となります。
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。

ORIGIN DELAY SET関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスに、LIMIT DELAY TIME, SCAN DELAY TIME, PULSE DELAY TIMEを設定します。
SPEC INITIALIZE3 COMMANDでDEND信号を<サーボ対応>に設定している場合には、各工程で工程終了後、DEND信号の<サーボ対応>完了後にDELAY TIMEを挿入します。

この関数を実行する前にDRIVE STATUS1 PORT ERROR=0、DRIVE STATUS1 PORT BUSY=0を確認してください。

LIMIT DELAY TIME

LIMIT停止信号を検出して停止したときに挿入するディレイです。

SPEC INITIALIZE3 COMMANDでDRST信号を<サーボ対応>に設定している場合には、DRST信号出力完了後、LIMIT DELAY TIMEを挿入します。

電源投入後の初期値は、300msです。

SCAN DELAY TIME

SCAN工程およびCSCAN工程の動作反転時に挿入するディレイです。

- ・SCAN工程で検出信号を検出して停止したときに挿入します。
 - ・CONSTANT SCAN工程で検出信号を検出して停止したときに挿入します。
 - ・機械原点近傍アドレスまでのINDEXドライブ終了後に挿入します。
 - ・機械原点検出終了後、PRESETドライブ開始前に挿入します。
- 電源投入後の初期値は、50msです。

PULSE DELAY TIME

1PULSE送り工程で、1パルスずつパルス出力する時間間隔です。

検出信号は、PULSE DELAY TIME経過後にチェックします。

電源投入後の初期値は、20msです。

書式

C言語 BOOL MC07_SetOrgDelay(DWORD *hDev*, WORD *LimitDelay*, WORD *ScanDelay*,
WORD *PulseDelay*, MC07_S_RESULT **psResult*);

VB Function MC07_SetOrgDelay(ByVal *hDev* As Long, ByVal *LimitDelay* As Integer,
ByVal *ScanDelay* As Integer, ByVal *PulseDelay* As Integer,
ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_SetOrgDelay(ByVal *hDev* As Integer, ByVal *LimitDelay* As Short,
ByVal *ScanDelay* As Short, ByVal *PulseDelay* As Short,
ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.SetOrgDelay(uint *hDev*, ushort *LimitDelay*, ushort *ScanDelay*, ushort *PulseDelay*,
ref MC07_S_RESULT *psResult*);

引数

- hDev* ... デバイスハンドルを指定します。
- LimitDelay* ... LIMIT DELAY TIME(×5ms)を設定します。(0~1,275ms)
1,275msより大きい値が設定された場合は、1,275msに補正されます。
- ScanDelay* ... SCAN DELAY TIME(×5ms)を設定します。(0~1,275ms)
1,275msより大きい値が設定された場合は、1,275msに補正されます。
- PulseDelay* ... PULSE DELAY TIME(×5ms)を設定します。(0~1,275ms)
1,275msより大きい値が設定された場合は、1,275msに補正されます。
- 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。

ORIGIN ERROR PULSE SET関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスに、CONSTANT SCAN工程時にエラー判定する最大パルス数、および1PULSE送り工程時にエラー判定する最大パルス数を設定します。

ORIGIN SPEC SET関数でERROR PULSE ERROR ENABLE=1に設定している場合に有効です。
電源投入後の初期値は、CONSTANT SCAN工程時にエラー判定する最大パルス数、1PULSE送り工程時にエラー判定する最大パルス数ともに2,147,483,647パルスです。

この関数を実行する前にDRIVE STATUS1 PORT ERROR=0、DRIVE STATUS1 PORT BUSY=0を確認してください。

書式

C言語 BOOL MC07_SetOrgErrorPulse(DWORD *hDev*, DWORD *CScanErrorPulse*, DWORD *PulseErrorPulse*,
MC07_S_RESULT **psResult*);

VB Function MC07_SetOrgErrorPulse(ByVal *hDev* As Long, ByVal *CScanErrorPulse* As Long,
ByVal *PulseErrorPulse* As Long, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_SetOrgErrorPulse(ByVal *hDev* As Integer, ByVal *CScanErrorPulse* As Integer,
ByVal *PulseErrorPulse* As Integer, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.SetOrgErrorPulse(uint *hDev*, uint *CScanErrorPulse*, uint *PulseErrorPulse*,
ref MC07_S_RESULT *psResult*);

引数

hDev … デバイスハンドルを指定します。
CScanErrorPulse … CONSTANT SCAN工程時にエラー判定するパルス数(2~2,147,483,647パルス)を設定します。
1~2,147,483,647パルス以外が設定された場合は、1パルスに補正されます。
PulseErrorPulse … 1PULSE送り工程時にエラー判定する最大パルス数(2~2,147,483,647パルス)を設定します。
1~2,147,483,647パルス以外が設定された場合は、1パルスに補正されます。
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。

ORIGIN OFFSET PULSE SET関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスに、OFFSETパルス数を設定します。
機械原点アドレスに加算するパルス数です。
機械原点アドレスにOFFSETパルス数を加算したアドレスが機械原点近傍アドレスになります。

電源投入後の初期値は、100パルスです。

この関数を実行する前にDRIVE STATUS1 PORT ERROR=0、DRIVE STATUS1 PORT BUSY=0を確認してください。

書式

C言語 `BOOL MC07_SetOrgOffsetPulse(DWORD hDev, DWORD OffsetPulse, MC07_S_RESULT *psResult);`

VB `Function MC07_SetOrgOffsetPulse(ByVal hDev As Long, ByVal OffsetPulse As Long,
ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_SetOrgOffsetPulse(ByVal hDev As Integer, ByVal OffsetPulse As Integer,
ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.SetOrgOffsetPulse(uint hDev, uint OffsetPulse, ref MC07_S_RESULT psResult);`

引数

hDev … デバイスハンドルを指定します。
OffsetPulse … 機械原点近傍アドレスのOFFSETパルス数(0 ~ 2,147,483,647パルス)を設定します。
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。

ORIGIN PRESET PULSE SET関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機 能

指定されたデバイスに、PRESETパルス数を設定します。
機械原点検出後に実行するPRESETドライブで出力するパルス数です。

電源投入後の初期値は、0パルスです。

この関数を実行する前にDRIVE STATUS1 PORT ERROR=0、DRIVE STATUS1 PORT BUSY=0を確認してください。

書 式

C言語 BOOL MC07_SetOrgPresetPulse(DWORD *hDev*, LONG *PresetPulse*, MC07_S_RESULT **psResult*);

VB Function MC07_SetOrgPresetPulse(ByVal *hDev* As Long, ByVal *PresetPulse* As Long,
ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_SetOrgPresetPulse(ByVal *hDev* As Integer, ByVal *PresetPulse* As Integer,
ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.SetOrgPresetPulse(uint *hDev*, int *PresetPulse*, ref MC07_S_RESULT *psResult*);

引 数

hDev … デバイスハンドルを指定します。
PresetPulse … PRESETパルス数 (-2,147,483,648 ~ +2,147,483,647パルス)を設定します。
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。

ORIGIN FLAG RESET関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機 能

指定されたデバイスのORIGIN FLAGを0にRESETします。

以下の場合は、ORIGIN FLAGをRESETした後にORIGINドライブを行ってください。

- ・回転系で絶対アドレスに意味がないとき
- ・機械原点の高速化機能を無効にし、毎回センサ基準で機械原点検出を行いたいとき
- ・1回目の機械原点検出完了後に、強制的に装置上の実位置をずらしたとき

この関数を実行する前にDRIVE STATUS1 PORT ERROR=0、DRIVE STATUS1 PORT BUSY=0を確認してください。

書 式

C言語 BOOL MC07_ResetOrgFlag(DWORD *hDev*, MC07_S_RESULT **psResult*);

VB Function MC07_ResetOrgFlag(ByVal *hDev* As Long, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_ResetOrgFlag(ByVal *hDev* As Integer, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.ResetOrgFlag(uint *hDev*, ref MC07_S_RESULT *psResult*);

引 数

hDev …… デバイスハンドルを指定します。

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。

ORIGINドライブ関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

機能

指定されたデバイスで、ORIGINドライブを行います。

この関数を実行する前にDRIVE STATUS1 PORT ERROR=0、DRIVE STATUS1 PORT BUSY=0を確認してください。

書式

C言語 `BOOL MC07_Org(DWORD hDev, WORD OrgType, MC07_S_RESULT *psResult);`

VB `Function MC07_Org(ByVal hDev As Long, ByVal OrgType As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_Org(ByVal hDev As Integer, ByVal OrgType As Short, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.Org(uint hDev, ushort OrgType, ref MC07_S_RESULT psResult);`

引数

hDev … デバイスハンドルを指定します。
OrgType … ORIGINドライブ型式を指定します。

指定できる値	意味	指定できる値	意味
MC07_ORG0	ORG-0	MC07_ORG5	ORG-5
MC07_ORG1	ORG-1	MC07_ORG10	ORG-10
MC07_ORG2	ORG-2	MC07_ORG11	ORG-11
MC07_ORG3	ORG-3	MC07_ORG12	ORG-12
MC07_ORG4	ORG-4		

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。

4-6. I/O PORT 関数

I/O PORT オープン関数で取得した I/O PORT ハンドルにより、I/O PORT を制御します。

4-6-1. I/O オープン/クローズ関数

ユーザアプリケーションは、I/O PORT をオープンし、PORT ハンドルを受け取ります。以後、I/O PORT の関数を実行する際に、この PORT ハンドルを引数として指定します。この PORT ハンドルは、I/O PORT をクローズするまで有効です。ユーザアプリケーション終了時は、必ず I/O PORT をクローズしてください。

I/O PORT オープン関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
CB-52	CB-53	

機能

I/O PORT をオープンし、引数 *phPort* で示される変数に PORT ハンドルを格納します。

書式

C言語 `BOOL MC07_BPortOpen(WORD UnitNo, WORD IoPort, DWORD *phPort, MC07_S_RESULT *psResult);`

VB `Function MC07_BPortOpen(ByVal UnitNo As Integer, ByVal IoPort As Integer, ByRef phPort As Long, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BPortOpen(ByVal UnitNo As Integer, ByVal IoPort As Short, ByRef phPort As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BPortOpen(ushort UnitNo, ushort IoPort, ref uint phPort, ref MC07_S_RESULT psResult);`

引数

UnitNo …… ユニット番号を指定します。

指定できる値	意味
MC07_USB_UNIT_0	ユニット番号0
MC07_USB_UNIT_1	ユニット番号1

IoPort …… I/O PORT を指定します。

指定できる値	意味	指定できる値	意味
MC07_GP_IN	汎用 I/O 入力 PORT	MC07_GP_OUT	汎用 I/O 出力 PORT
MC07_EXPO_IN	拡張 I/O 入力0 PORT	MC07_EXPO_OUT	拡張 I/O 出力0 PORT
MC07_EXP1_IN	拡張 I/O 入力1 PORT	MC07_EXP1_OUT	拡張 I/O 出力1 PORT
MC07_CTLPO_IN	制御 I/O 入力0 PORT	MC07_CTLPO_OUT	制御 I/O 出力0 PORT

phPort …… PORT ハンドル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。

I/O PORTクローズ関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

CB-52	CB-53
-------	-------

機 能

指定された I/O PORT をクローズします。

書 式

C言語 `BOOL MC07_BPortClose(DWORD hPort, MC07_S_RESULT *psResult);`

VB `Function MC07_BPortClose(ByVal hPort As Long, ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BPortClose(ByVal hPort As Integer, ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BPortClose(uint hPort, ref MC07_S_RESULT psResult);`

引 数

hPort …… PORTハンドルを指定します。

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。

4-6-2. I/O アクセス関数

I/O PORT の書き込みと読み出しを行います。

I/O PORT書き込み関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

CB-52	CB-53
-------	-------

機 能

指定された I/O PORT にデータを書き込みます。

書 式

C言語 `BOOL MC07_BPortOut(DWORD hPort, WORD *pData, MC07_S_RESULT *psResult);`

VB `Function MC07_BPortOut(ByVal hPort As Long, ByRef pData As Integer,
ByRef psResult As MC07_S_RESULT) As Boolean`

VB.NET `Function MC07_BPortOut(ByVal hPort As Integer, ByRef pData As Short,
ByRef psResult As MC07_S_RESULT) As Boolean`

C#.NET `bool MC07.BPortOut(uint hPort, ref ushort pData, ref MC07_S_RESULT psResult);`

引 数

hPort ... PORT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。
書き込むデータについては、6-3-1.章をご覧ください。
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。

I/O PORT OR 書き込み関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機能

指定された I/O PORT にデータを OR 書き込みます。

書式

C言語 BOOL MC07_BPortOrOut(DWORD *hPort*, WORD **pData*, MC07_S_RESULT **psResult*);

VB Function MC07_BPortOrOut(ByVal *hPort* As Long, ByRef *pData* As Integer, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_BPortOrOut(ByVal *hPort* As Integer, ByRef *pData* As Short, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.BPortOrOut(uint *hPort*, ref ushort *pData*, ref MC07_S_RESULT *psResult*);

引数

hPort ... PORTハンドルを指定します。
pData ... OR書き込みするデータが格納されている変数のポインタを指定します。
OR書き込むデータについては、6-3-1.章をご覧ください。
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。

I/O PORT AND 書き込み関数

UsbC

UC-766	UCD-7610v1	UCD-7613v1
--------	------------	------------

CB-52	CB-53
-------	-------

機 能

指定された I/O PORT にデータを AND 書き込みます。

書 式

C言語 BOOL MC07_BPortAndOut(DWORD *hPort*, WORD **pData*, MC07_S_RESULT **psResult*);

VB Function MC07_BPortAndOut(ByVal *hPort* As Long, ByRef *pData* As Integer, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_BPortAndOut(ByVal *hPort* As Integer, ByRef *pData* As Short, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.BPortAndOut(uint *hPort*, ref ushort *pData*, ref MC07_S_RESULT *psResult*);

引 数

hPort ... PORTハンドルを指定します。
pData ... AND書き込みするデータが格納されている変数のポインタを指定します。
AND書き込むデータについては、6-3-1.章をご覧ください。
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。

I/O PORT読み出し関数

UsbC

UC-766 UCD-7610v1 UCD-7613v1

CB-52 CB-53

機 能

指定された I/O PORT のデータを読み出します。

書 式

C言語 BOOL MC07_BPortIn(DWORD *hPort*, WORD **pData*, MC07_S_RESULT **psResult*);

VB Function MC07_BPortIn(ByVal *hPort* As Long, ByRef *pData* As Integer, ByRef *psResult* As MC07_S_RESULT) As Boolean

VB.NET Function MC07_BPortIn(ByVal *hPort* As Integer, ByRef *pData* As Short, ByRef *psResult* As MC07_S_RESULT) As Boolean

C#.NET bool MC07.BPortIn(uint *hPort*, ref ushort *pData*, ref MC07_S_RESULT *psResult*);

引 数

hPort ... PORT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。
読み出したデータについては、6-3-1.章をご覧ください。
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。

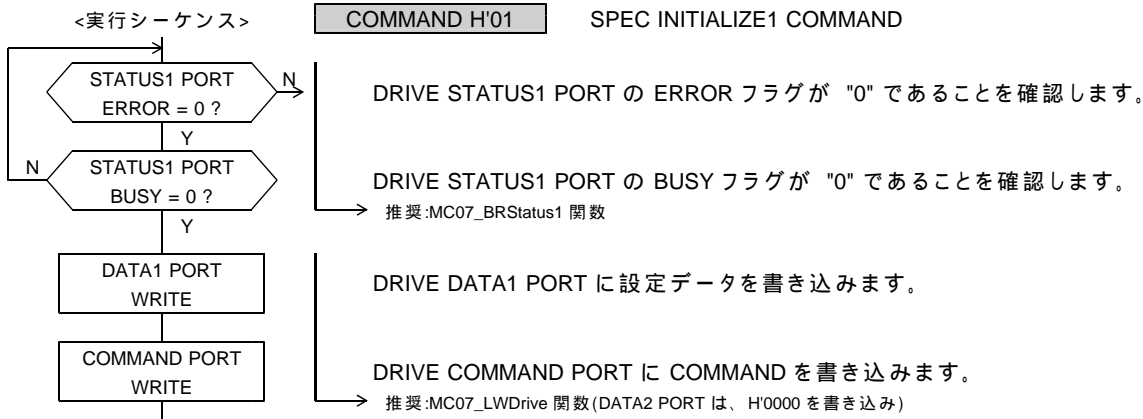
5. コマンド仕様

5-1. ドライブコマンド

5-1-1. 入出力仕様の設定

(1) SPEC INITIALIZE1

ドライブパルスの出力仕様を設定します。



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
-	-	-	-	-	PULSE OUTPUT MASK	PULSE OUTPUT TYPE1	PULSE OUTPUT TYPE0

電源投入後の初期値は H'0000 (アンダーライン側) です。

D0 : PULSE OUTPUT TYPE0

D1 : PULSE OUTPUT TYPE1

CWP, CCWP 信号出力のドライブパルス出力方式を選択します。

TYPE1	TYPE0	パルス出力方式	CWP 信号出力	CCWP 信号出力
0	0	独立方向出力	+ 方向のパルス出力	- 方向のパルス出力
0	1	方向指定出力	パルス出力	方向出力
1	0	2 逓倍の位相差信号	A 相出力	B 相出力
1	1	4 逓倍の位相差信号	A 相出力	B 相出力

- ・コントローラドライバ製品のドライブパルス出力方式の設定は禁止です。
- ・CWP, CCWP のアクティブ論理が「ローアクティブ」のときの出力仕様です。
- ・方向出力の場合、 $\overline{\text{CCWP}}$ 信号 = HIGH で+(CW)方向, $\overline{\text{CCWP}}$ 信号 = LOW で-(CCW)方向を示します。

D2 : PULSE OUTPUT MASK

CWP, CCWP 信号出力のドライブパルス出力を「マスクする / マスクしない」を選択します。

0 : ドライブパルス出力をマスクしない (パルスを出力する)

1 : ドライブパルス出力をマスクする (パルスを出力しない)

"1"「マスクする」に設定した場合は、CWP, CCWP 信号の出力を OFF レベルに固定します。

- ・アドレスカウンタのカウンパルスと発生パルス(OP)の出力もマスクします。
- ・アドレスカウンタが停止するため、ABS INDEX ドライブを実行すると自動停止できません。
- ・パルスカウンタとパルス偏差カウンタも発生パルス(OP)をカウントすることができません。
- ・その他の機能は「マスクしない」を選択した場合と同様です。

"1"「マスクする」に設定すると、DRIVE STATUS2 PORT の PULSE MASK = 1 にします。

- ・パルス出力をマスクしたドライブの実行時間は、タイマとして使用できます。

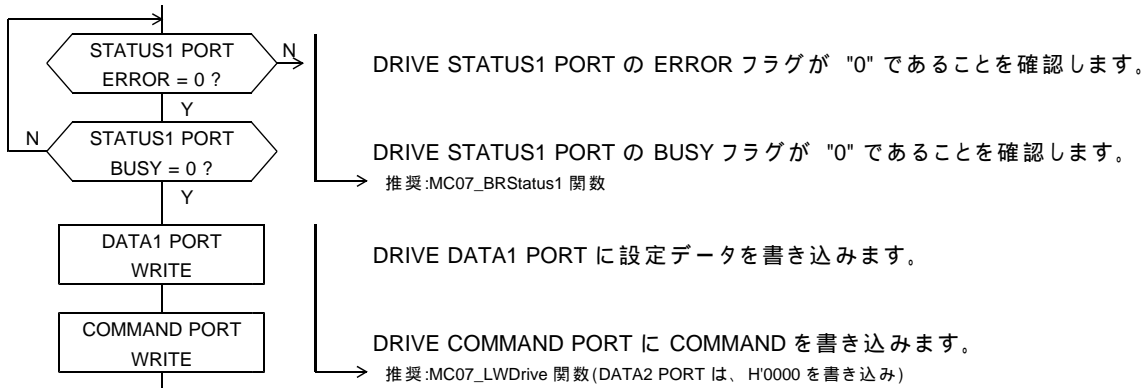
(2) SPEC INITIALIZE2

CWLM, CCWLM 信号の入力機能を設定します。SS0 信号の入力機能を設定します。

<実行シーケンス>

COMMAND H'02

SPEC INITIALIZE2 COMMAND



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
-	-	-	-	1	1	SS0 TYPE1	SS0 TYPE0

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	CCWLM TYPE1	CCWLM TYPE0	CWLM TYPE1	CWLM TYPE0

電源投入後の初期値は H'0F30 (アンダーライン側) です。

D0 : CWLM TYPE0

D1 : CWLM TYPE1

CWLM 信号入力のアクティブレベル検出時の入力機能を選択します。

TYPE1	TYPE0	CWLM 信号の入力機能
0	0	+ 方向の LIMIT 即時停止信号として使用する
0	1	+ 方向の LIMIT 減速停止信号として使用する
1	0	即時停止信号として使用する
1	1	汎用入力として使用する

D2 : CCWLM TYPE0

D3 : CCWLM TYPE1

CCWLM 信号入力のアクティブレベル検出時の入力機能を選択します。

TYPE1	TYPE0	CCWLM 信号の入力機能
0	0	- 方向の LIMIT 即時停止信号として使用する
0	1	- 方向の LIMIT 減速停止信号として使用する
1	0	即時停止信号として使用する
1	1	汎用入力として使用する

CWLM TYPE および CCWLM TYPE が "1,0" の "即時停止信号として使用する" に設定されている場合は、ORIGIN ドライブを実行できません。

上記信号を "即時停止信号として使用する" にするときは、ORIGIN ドライブ実行後に設定してください。

D4 : 1

D5 : 1

D8 : SS0 TYPE0

D9 : SS0 TYPE1

SS0 信号入力のアクティブレベル検出時の入力機能を選択します。

TYPE1	TYPE0	SS0 信号の入力機能
0	0	設定禁止
0	1	減速停止信号として使用する
1	0	即時停止信号として使用する
1	1	汎用入力として使用する

SS0 信号が"減速停止信号として使用する"、または"即時停止信号として使用する"に設定されている場合は、ORIGIN ドライブを実行できません。

上記信号を"停止信号として使用する"にするときは、ORIGIN ドライブ実行後に設定してください。

- ・ 汎用入力 $\overline{IN0}$ 信号から X 軸 SS0、 $\overline{IN1}$ 信号から Y 軸 SS0 が操作できます。
SS0 信号の各軸の割り当て(Z 軸と A 軸の SS0 は操作できません。)

SS0	信号
X 軸	$\overline{IN0}$ 信号
Y 軸	$\overline{IN1}$ 信号
Z 軸	-
A 軸	-

$\overline{IN0}$ 信号および $\overline{IN1}$ 信号に接続されている SS0 信号の汎用入力機能は、各種のトリガ信号としての応用が可能です。

- ・ DFL カウンタをハードタイマとしたカウント開始のタイミング (DFL COUNTER INITIALIZE1 コマンドで設定)
- ・ UP / DOWN / CONST CHANGE のタイミング * (UDC SPEC SET コマンドで設定)
- ・ SPEED CHANGE および RATE CHANGE のタイミング * (SPEED CHANGE SPEC SET コマンドで設定)
- ・ INDEX CHANGE のタイミング * (INDEX CHANGE SPEC SET コマンドで設定)
- ・ カウンタのカウントデータラッチ / クリアのタイミング * (COUNT LATCH SPEC SET コマンドで設定)

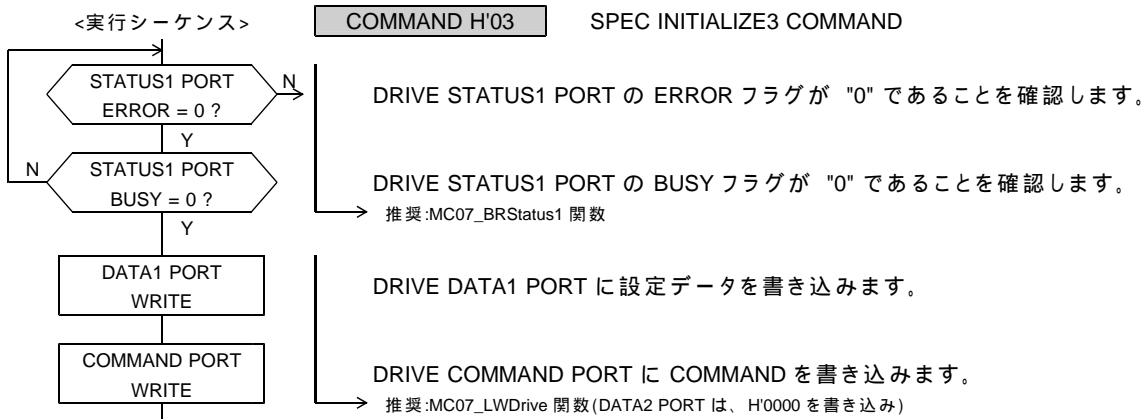
*は、応用機能です。

D10 : 1

D11 : 1

(3) SPEC INITIALIZE3

DRST/MF 信号の出力機能、 $\overline{\text{DEND/PO}}$, DALM 信号の入力機能を設定します。



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
-	0	-	DOWN PULSE MASK	-	0	0	0

D7	D6	D5	D4	D3	D2	D1	D0
-	-	DALM TYPE1	DALM TYPE0	DEND/PO TYPE1	DEND/PO TYPE0	DRST TYPE1	DRST TYPE0

電源投入後の初期値は H'003F (アンダーライン側) です。

D0 : DRST TYPE0

D1 : DRST TYPE1

DRST/MF 信号の出力機能を選択します。

TYPE1	TYPE0	DRST 信号の出力機能	サーボ対応
0	0	サーボ対応の停止時に 10 ms 間アクティブレベルを出力する	<サーボ対応>
0	1	設定禁止	-
1	0	設定禁止	-
1	1	汎用出力として使用する	-

"00"に設定した場合は、DRST 信号の サーボ対応 機能を実行します。

"11"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。

コントローラドライバ製品は、汎用出力(初期値)の設定とし、SIGNAL OUT コマンドによってモータの励磁電流を ON/OFF することができます。

DRST 信号のサーボ対応

- ・ STBY = 1、DRIVE = 1 または DEND BUSY = 1 のとき、以下の停止指令を検出すると、ドライブパルス出力終了後に、DRST 信号に 10 ms 間アクティブレベルを出力します。
 - ・ 即時停止指令
 - ・ LIMIT 即時停止指令
- ・ ORIGIN SPEC SET 関数の AUTO DRST ENABLE = 1 のときには、ORG エッジ信号の停止機能が動作 (ORGEND = 1) すると、ドライブパルス出力終了後に、DRST 信号に 10 ms 間アクティブレベルを出力します。
- ・ DRST コマンドを実行すると、DRST 信号に 10 ms 間アクティブレベルを出力します。

DRST 信号の<サーボ対応>実行中は、DRIVE STATUS1 PORT の BUSY = 1 のままです。
DRST 信号および DEND 信号の<サーボ対応>終了後に、ドライブを終了します。

即時停止指令

- ・ FAST STOP コマンド
- ・ 入力機能を即時停止に設定した SS0 信号
- ・ 入力機能を即時停止に設定した DEND, DALM 信号
- ・ 入力機能を即時停止に設定した CWLM, CCWLM 信号
- ・ 停止機能を即時停止に設定した各種カウンタのコンパレータ出力

LIMIT 即時停止指令

- ・ 入力機能を LIMIT 即時停止に設定した CWLM, CCWLM 信号
- ・ 停止機能を LIMIT 即時停止に設定した各種カウンタのコンパレータ出力

D2 : DEND/PO TYPE0

D3 : DEND/PO TYPE1

DEND/PO 信号の入力機能を選択します。

TYPE1	TYPE0	DEND/PO 信号の入力機能	サーボ対応
0	0	DEND/PO のアクティブを検出するまでドライブを終了しない	<サーボ対応>
0	1	減速停止信号として使用する	—
1	0	即時停止信号として使用する	—
1	1	汎用入力として使用する	—

"00"を設定した場合は、DEND 信号のサーボ対応を実行します。
コントローラドライブ製品の DEND/PO 入力機能は設定禁止です。(初期値の汎用入力としてください。)

DEND 信号のサーボ対応

- ・ ドライブパルス出力が終了しても、
DEND 信号のアクティブレベルを検出するまで、DRIVE STATUS2 PORT の DEND BUSY = 1 にします。
DEND BUSY = 1 の間は、DRIVE STATUS1 PORT の BUSY = 1 のままです。
DEND 信号のアクティブレベルを検出すると、DEND BUSY = 0 にします。
- ・ 即時停止指令を検出すると、DEND 信号の<サーボ対応>を中止して、DEND BUSY = 0 にします。

DEND 信号が"減速停止信号として使用する"、または"即時停止信号として使用する"に設定されている場合は、ORIGIN ドライブを実行できません。
上記信号を"停止信号として使用する"にするとときは、ORIGIN ドライブ実行後に設定してください。

D4 : DALM TYPE0

D5 : DALM TYPE1

DALM 信号のアクティブレベル検出時の入力機能を選択します。

TYPE1	TYPE0	DALM 信号の入力機能	サーボ対応
0	0	機能はありません (汎用入力)	—
0	1	減速停止信号として使用する	—
1	0	即時停止信号として使用する	—
1	1	汎用入力として使用する	—

- ・ DALM 信号の入力状態は DRIVE STATUS2 PORT の DALM フラグから読み出すことができます。
DALM 信号の検出(サーボ異常やステッピングモータドライバの過熱警告信号など)により、即時停止または減速停止させることができます。

D8 : 0
D9 : 0
D10 : 0

D12 : DOWN PULSE MASK

INDEX ドライブの停止位置への減速停止動作時に有効です。(応用機能)

MCC07 が自動検出する減速パルス数、およびマニュアル設定の減速パルス数を「マスクする / マスクしない」を選択します。

- 0 : マスクしない (停止位置へ自動減速 / マニュアル減速の停止動作で停止する)
- 1 : マスクする (停止位置で即時停止する: 減速パルス数 "0")

"0"「マスクしない」に設定した場合は、
「加減速ドライブ」および「減速ドライブ」の INDEX ドライブ中に、
停止位置への減速停止動作を開始して、指定アドレスで停止します。

"1"「マスクする」に設定した場合は、減速パルス数による減速開始地点の検出を行いません。
INDEX ドライブの指定アドレスを検出すると、減速パルス数なしで即時停止します。

- ・ドライブ形状を「加減速ドライブ」に設定している場合は、「加速ドライブ」の形状でドライブを終了します。最高速度に到達した場合の終了速度は、HSPD x RESOL です。
- ・ドライブ形状を「減速ドライブ」に設定している場合は、「一定速ドライブ」の形状でドライブを終了します。この場合の終了速度は、HSPD x RESOL です。
- ・ドライブ中に減速停止指令または ERROR = 1 を検出した場合は、終了速度まで減速してドライブを終了します。
ただし、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。
- ・S 字加速中の減速停止指令検出時の三角駆動回避動作は有効です。
加速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ドライブを終了します。
ただし、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。
- ・S 字加減速 INDEX ドライブの三角駆動回避動作は無効になります。
最高速度へ加速したまま、指定アドレスで即時停止します。

"1"「マスクする」に設定した場合は、INDEX CHANGE 機能は無効になります。

D14 : 0

5-1-2. 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。

各ドライブのパラメータ設定は、変更が必要な場合に設定します。

ドライブパラメータには、デバイスドライバの SPEED・RATE 関数で設定するパラメータとドライブコマンドで直接設定するパラメータがあります。

デバイスドライバの関数で設定するパラメータ

- ・第 1 パルス出力周期
- ・加減速パラメータ

ドライブコマンドで直接設定するパラメータ

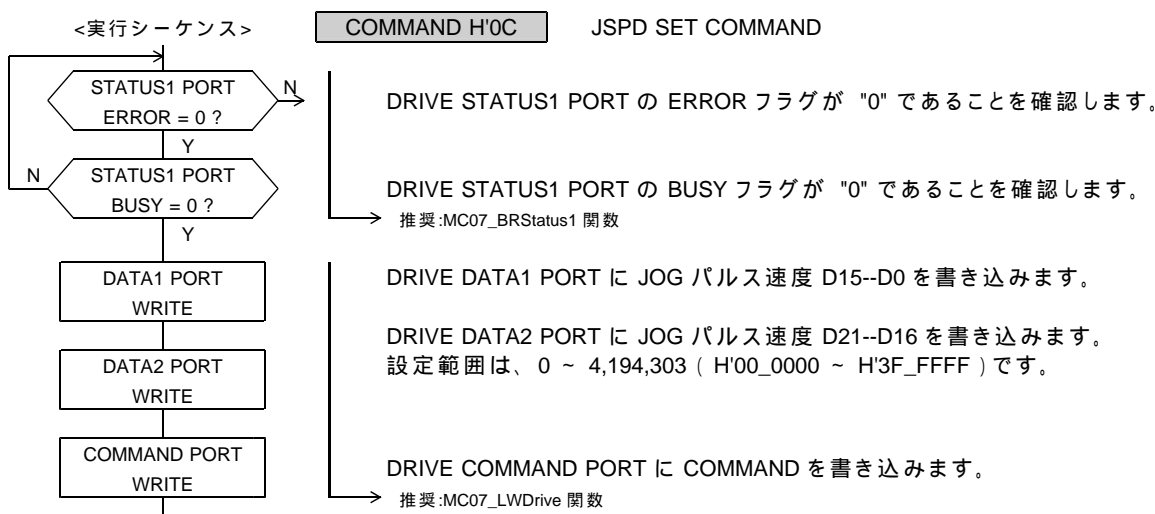
- ・JOG パラメータ

ここではドライブコマンドで直接設定するパラメータのコマンド仕様を示します。

デバイスドライバの関数で実行するパラメータについては SPEED・RATE 関数仕様をご覧ください。

(1) JSPD SET

JOG ドライブの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
← JSPD →															

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
-	-	-	-	-	-	-	-	-	-	← JSPD →					-

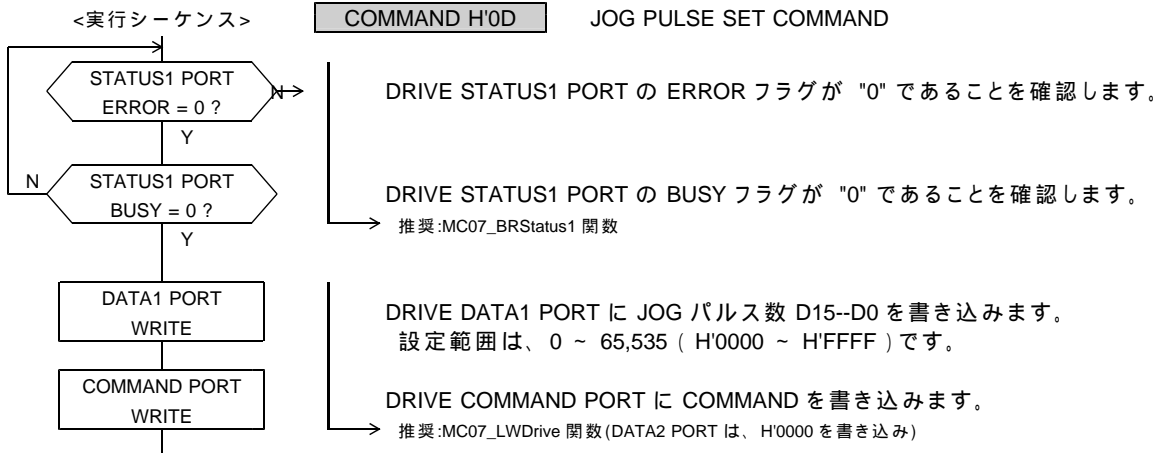
電源投入後の初期値は H'00_012C (300 Hz) です。

- ・ JSPD の設定値が "0" の場合は、"1" に補正します。
- ・ JOG ドライブと CONSTANT SCAN 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 JSPD になります。

(2) JOG PULSE SET

JOG コマンドによる複数パルスの JOG 動作を行うときに設定します。

JOG パルス数を設定します。



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
← JOG PULSE →															

電源投入後の初期値は H'0001 (1 パルス) です。

- ・ JOG PULSE が "0" の場合は、パルス出力なしで、JOG ドライブを終了します。

5-1-3. ドライブの実行

ドライブの実行には、ドライブコマンドで直接実行するドライブとデバイスドライバの関数で実行するドライブがあります。

ドライブコマンドで直接実行するドライブ

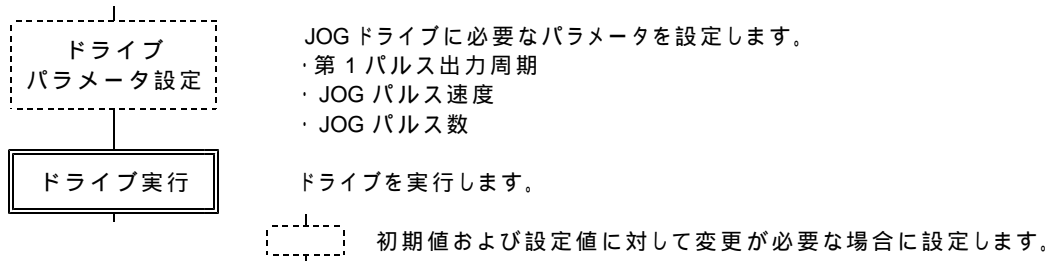
- ・ JOG ドライブ
- ・ SCAN ドライブ
- ・ INC INDEX ドライブ
- ・ ABS INDEX ドライブ

関数で実行するドライブ

- ・ ORIGIN ドライブ(機械原点検出はコントローラ側で各工程を自動的に実行します。)
- ・ 2 軸相対アドレス直線補間ドライブ関数
- ・ 2 軸相対アドレス円弧補間ドライブ関数
- ・ メインチップ 2 軸相対アドレス直線補間ドライブ関数
- ・ メインチップ 2 軸相対アドレス円弧補間ドライブ関数

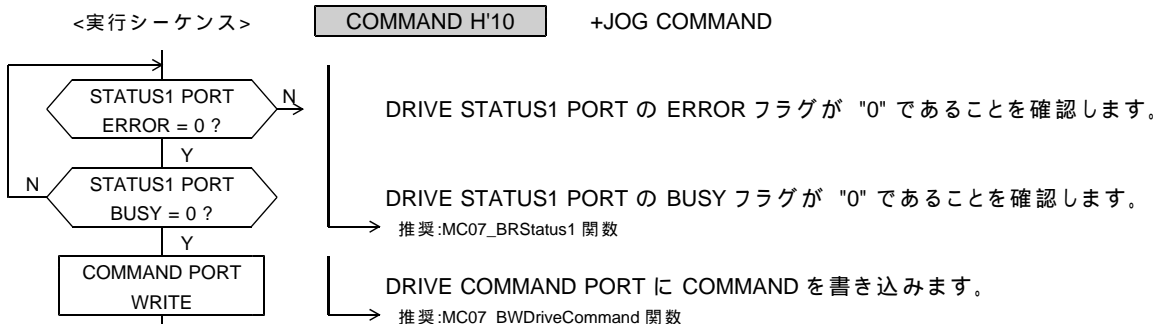
ここではドライブコマンドで直接実行するドライブのコマンド仕様を示します。
デバイスドライバの関数で実行するドライブについては各ドライブの関数仕様をご覧ください。

JOG ドライブの実行シーケンス



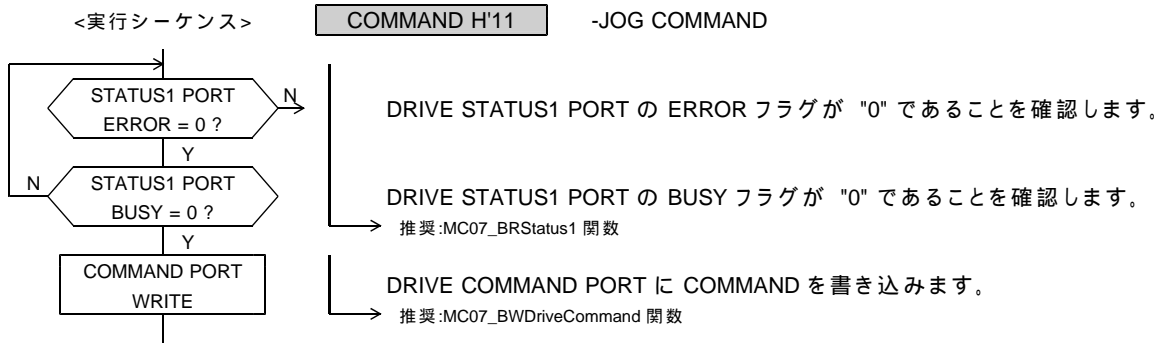
(1) +JOG

+ (CW)方向の JOG ドライブを実行します。

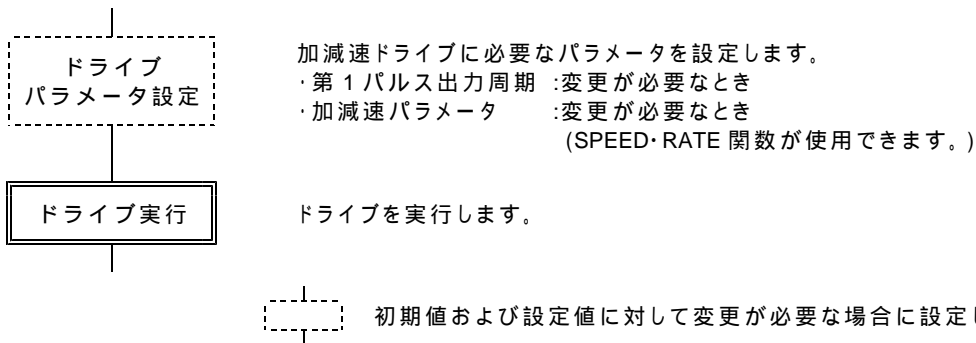


(2) -JOG

- (CCW)方向の JOG ドライブを実行します。

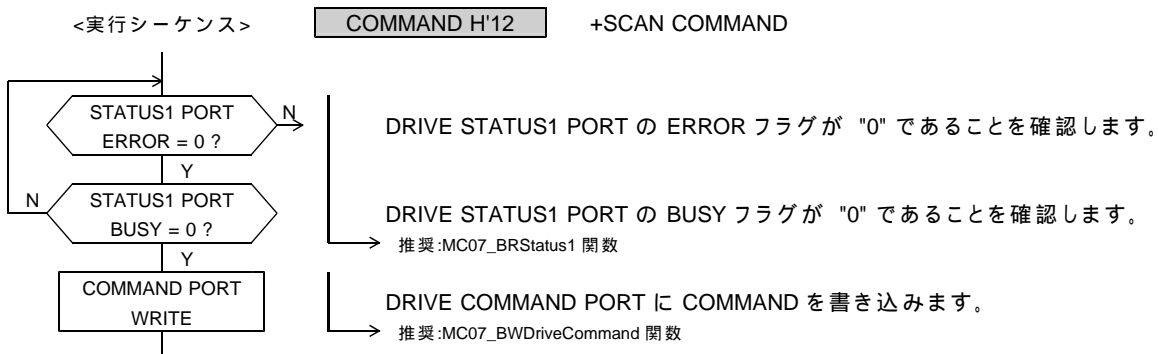


加減速ドライブの実行シーケンス



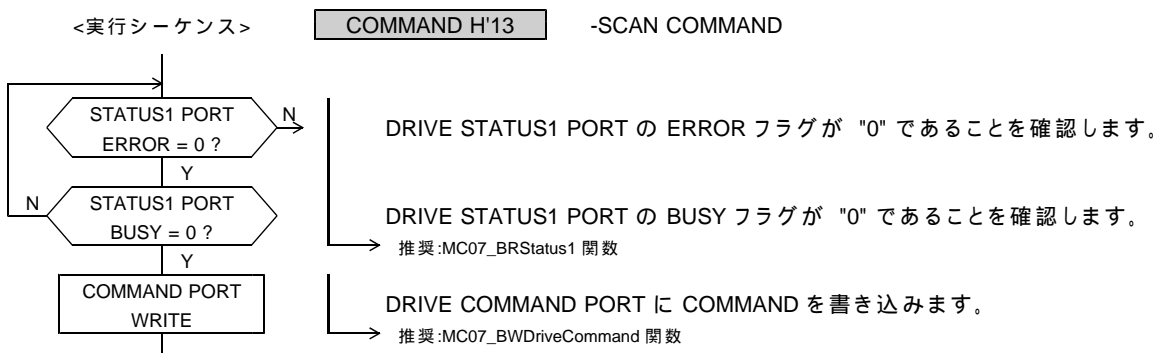
(3) +SCAN

停止指令を検出するまで、+ (CW)方向のパルスを連続して出力します。



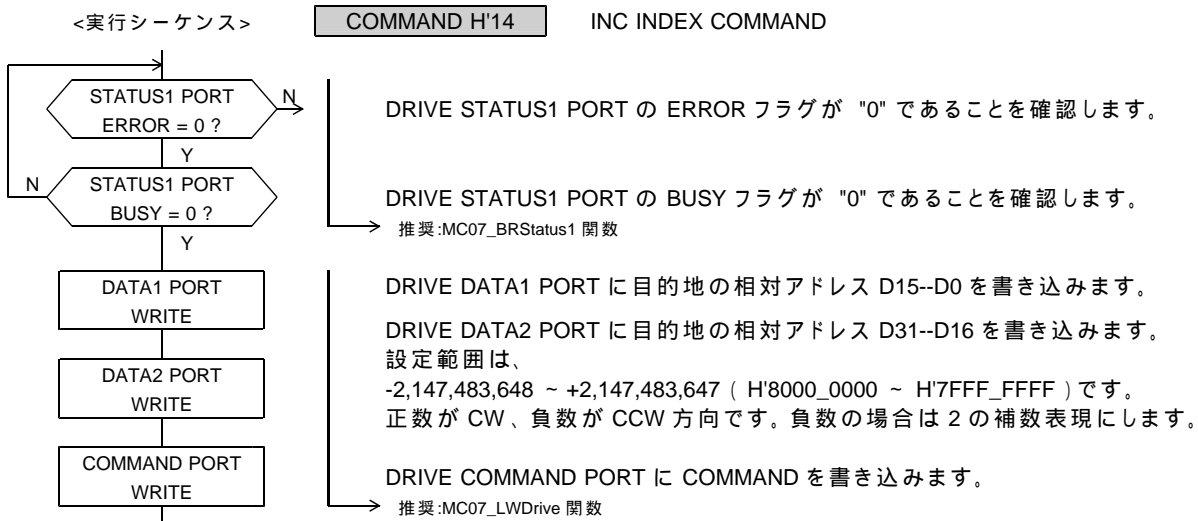
(4) -SCAN

停止指令を検出するまで、- (CCW)方向のパルスを連続して出力します。



(5) INC INDEX

指定の相対アドレスに達するまで、+ (CW)方向、または - (CCW)方向のパルスを出力します。



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 ← 目的地の相対アドレス → 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 ← 目的地の相対アドレス → D16															

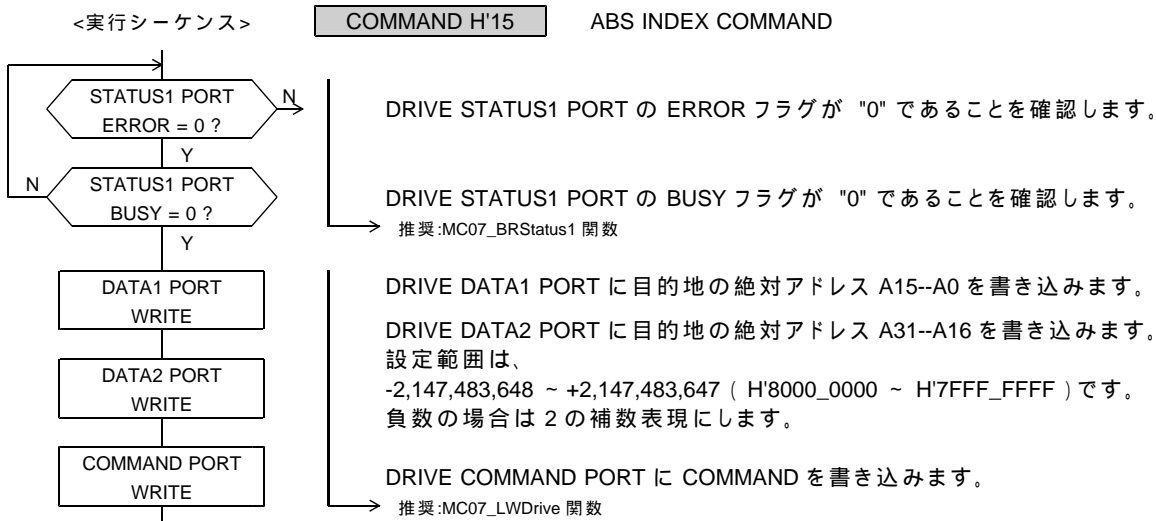
指定する相対アドレスは、
起動位置から停止位置までのパルス数を、起動位置を原点として符号付きで表現した値です。

相対アドレスが "H'0000_0000" の場合は、パルス出力なしでドライブを終了します。

以下の場合、エラーになり、ERROR STATUS の INC INDEX ERROR = 1 にします。
・相対アドレスがオーバーフローしているときに、INC INDEX CHANGE 指令を検出した。

(6) ABS INDEX

指定の絶対アドレスに達するまで、+ (CW)方向、または - (CCW)方向のパルスを出力します。



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
A15 ← 目的地の絶対アドレス → 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 ← 目的地の絶対アドレス → 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。

指定アドレスがアドレスカウンタの値と同じ場合は、パルス出力なしでドライブを終了します。

以下の場合、エラーになり、ERROR STATUS の ABS INDEX ERROR = 1 にします。

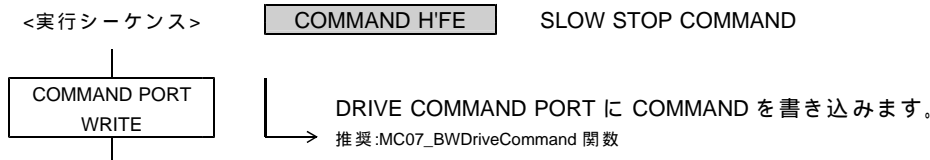
- ・ ABS INDEX ドライブ実行中に、アドレスカウンタのオーバーフローを検出したとき。
- ・ アドレスカウンタがオーバーフローしているときに、ABS INDEX CHANGE 指令を検出したとき。

5-1-4. 停止コマンドの実行

パルス出力停止機能を実行して、ドライブを終了します。
停止コマンドには、減速停止コマンドと即時停止コマンドがあります。

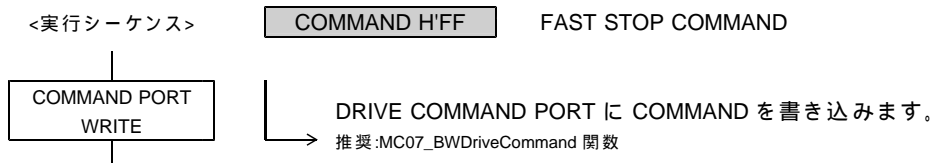
(1) SLOW STOP

コマンドによる減速停止機能を実行します。このコマンドの実行は常時可能です。
DRIVE STATUS1 PORT の STBY = 1 または DRIVE = 1 のときに有効です。



(2) FAST STOP

コマンドによる即時停止機能を実行します。このコマンドの実行は常時可能です。
DRIVE STATUS1 PORT の BUSY = 1 のときに有効です。



- ・ FAST STOP コマンドを検出すると、BUSY = 0 になるまで、即時停止機能が有効状態になります。

コマンド予約機能(応用機能)時の停止コマンド

SLOW STOP コマンドを受け付けると、DRIVE STATUS1 PORT の SSEND=1 になります。
FAST STOP コマンドを受け付けると、DRIVE STATUS1 PORT の FSEND=1 になります。
この停止コマンドにより停止した各 DRIVE STATUS1 PORT のフラグを ERROR STATUS MASK コマンドの設定によって、DRIVE STATUS1 PORT の ERROR=1 にする/しないの設定ができます。
DRIVE STATUS1 PORT の ERROR=1 になると、予約されているコマンドをキャンセルすることができます。

SLOW STOP コマンド

ERROR STATUS MASK コマンドで SSEND=1 で ERROR にしないとき(初期値)

- ・ 予約されている汎用コマンドの内、速度パラメータの設定は実行されます。
- ・ 実行しているドライブは減速停止した後に、次に予約されているドライブを実行します。

ERROR STATUS MASK コマンドで SSEND=1 で ERROR にしたとき

- ・ 予約されている汎用コマンドを全てキャンセルします。
- ・ 動作エラークリア関数が実行されるまで、ERROR=1 となります。

FAST STOP コマンド

ERROR STATUS MASK コマンドで FSEND=1 で ERROR にするとき(初期値)

- ・ 予約されている全ての汎用コマンドはキャンセルされます。
- ・ 動作エラークリア関数が実行されるまで、ERROR=1 となります。

ERROR STATUS MASK コマンドで FSEND=1 で ERROR にしないとき

- ・ 予約されている汎用コマンドの内、速度パラメータの設定は実行されます。
- ・ 実行しているドライブは即時停止した後に、次に予約されているドライブを実行します。

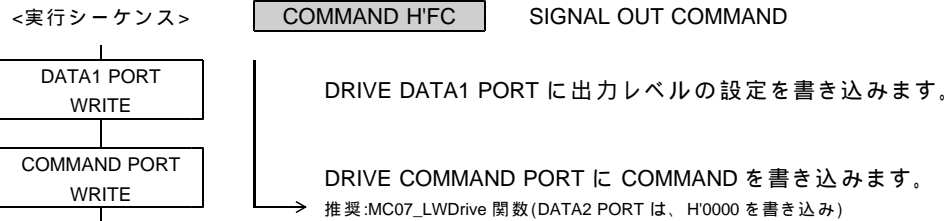
5-1-5. 出力信号の操作

(1) SIGNAL OUT

DRST は、 $\overline{\text{DRST/MF}}$ 信号から、設定された出力レベルを出力できます。

UC-766, UCD-7613v1/GDB5F40 は、SOUT0 信号から X 軸の SOUT を、SOUT1 信号から Y 軸の SOUT を汎用出力信号として設定された出力レベルを出力することができます。

コントローラドライバは、DRST/MF OUT をコマンド操作することで、モータの励磁電流を ON/OFF することができます。このコマンドの実行は常時可能です。



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
0	0	0	0	0	0	0	0

D7	D6	D5	D4	D3	D2	D1	D0
DRST/MF OUT	-	-	SOUT OUT	0	0	0	0

電源投入後の初期値は H'0000 (すべて OFF レベル出力) です。

D4 : SOUT OUT

SOUT 信号が出力するレベルを選択します。(X 軸と Y 軸のみ出力可能です。)

- 0 : OFF レベル出力
- 1 : アクティブレベル出力

D7 : DRST/MF OUT

出力信号が出力するレベルを選択します。

- 0 : OFF レベル出力 (HIGH レベル)
- 1 : アクティブレベル出力 (LOW レベル)

SIGNAL OUT コマンドの実行で、汎用出力信号の出力レベルが変化します。

各信号は、出力機能を「汎用出力」に設定している場合に有効です。

- ・ SOUT : HARD INITIALIZE1 コマンドで設定します。(初期値: CNTINT)
- ・ DRST/MF OUT : SPEC INITIALIZE3 コマンドで設定します。(初期値: 汎用出力)

(2) DRST OUT

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。
 $\overline{\text{DRST}}/\overline{\text{MF}}$ 信号から、サーボの偏差クリア信号として 10 ms 間アクティブレベルを出力します。
 このコマンドの実行は常時可能です。



- ・ $\overline{\text{DRST}}/\overline{\text{MF}}$ 信号の出力機能を「停止時に 10ms 間アクティブレベル出力」に設定している場合に有効です。
 $\overline{\text{DRST}}/\overline{\text{MF}}$ 信号の出力機能は SPEC INITIALIZE3 コマンドで設定します。
- ・ 10 ms 以内に連続してコマンドを実行すると、 $\overline{\text{DRST}}/\overline{\text{MF}}$ 信号のアクティブレベル出力を保持します。
 最後にコマンドを実行した時点から、10 ms 間アクティブレベルを出力して終了します。

5-1-6. エラー機能の設定と読み出し

(1) ERROR STATUS MASK

ERROR に出力する ERROR STATUS を個別にマスクします。

マスクする設定にすると、マスクされた要因による DRIVE STATUS1 PORT の ERROR=1 をマスクします。
このコマンドの実行は常時可能です。



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
1	FSSTOP ERROR MASK	DALM ERROR MASK	PULSE OVF ERROR MASK	ADDRESS OVF ERROR MASK	SSEND ERROR MASK	LSEND ERROR MASK	FSEND ERROR MASK
D7	D6	D5	D4	D3	D2	D1	D0
EXT PULSE ERROR MASK	CPP STOP ERROR MASK	CHANGE CLR ERROR MASK	0	0	0	COMREG CLR ERROR MASK	COMMAND ERROR MASK

電源投入後の初期値は H'FE00 です。

D14-D0 : マスクデータ

ERROR に出力する ERROR STATUS のマスクデータを選択します。

- 0 : マスクしない
- 1 : マスクする

- ・ ERROR 出力は、ERROR に出力する ERROR STATUS の OR (論理和) 出力です。
マスクした ERROR STATUS の出力は、OR (論理和) の入力で "0" 出力にします。
- ・ マスクしても、ERROR STATUS はクリアされません。
ERROR STATUS をクリアするときは、動作エラークリア関数を実行してください。
- ・ D4,D3,D2 の ERROR STATUS は、マスクできません。
D14-D9 の ERROR STATUS は、電源投入後の初期状態では「マスクする」です。
D15 の ERROR STATUS は、「1(マスクする)」にしてください。

ORIGIN ドライブ中は、ユーザが設定した ERROR STATUS MASK は無効となります。

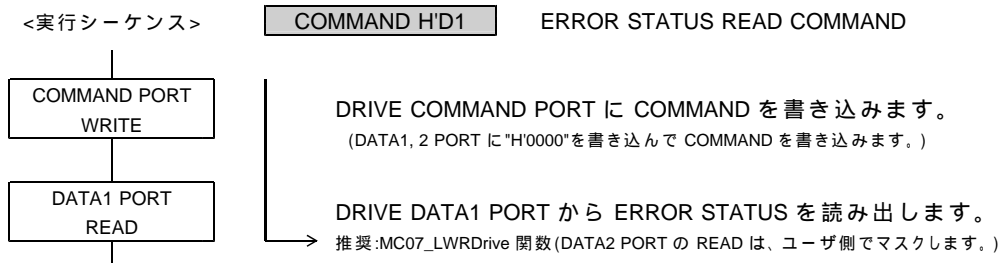
ORIGIN ドライブが終了すると ERROR STATUS MASK は、ユーザアプリケーションが設定した状態に戻ります。

(2) ERROR STATUS READ

ERROR STATUS を読み出します。

ERROR STATUS MASK されていても ERROR STATUS はクリアされずに ERROR 要因を読み出しできます。

このコマンドの実行は常時可能です。



DRIVE DATA1 PORT の読み出しデータ (ERROR STATUS)

D15	D14	D13	D12	D11	D10	D9	D8
0	FSSTOP ERROR	DALM ERROR	PULSE OVF ERROR	ADDRESS OVF ERROR	SSEND ERROR	LEND ERROR	FSEND ERROR

D7	D6	D5	D4	D3	D2	D1	D0
EXT PULSE ERROR	CPP STOP ERROR	CHANGE CLR ERROR	INDEX CHANGE ERROR	ABS INDEX ERROR	INC INDEX ERROR	COMREG CLR ERROR	COMMAND ERROR

各 ERROR STATUS は、"1" でエラーが発生したことを示します。

ERROR STATUS READ コマンドを実行すると、

ERROR STATUS を、DRIVE DATA1 PORT (READ) にセットします。

D10--D8 は、STATUS フラグが "1" でも、次の BUSY = 0 1 ではエラーになりません。

・ BUSY = 0 1 と同時に、FSEND, LSEND, SSEND = 1 0 にします。

D0 : COMMAND ERROR

未定義の汎用コマンドを実行したことを示します。

以下の場合、エラーになりません。コマンドおよび書き込みは無効にします。

- ・未定義の特殊コマンドを実行した
- ・ COMREG FL = 1 のときに、汎用コマンドを書き込んだ
- ・ SPEED CSET = 1 のときに、スピード系のドライブ CHANGE 設定コマンドを書き込んだ
- ・ SPEED CBUSY = 1 のときに、スピード系のドライブ CHANGE 実行コマンドを書き込んだ
- ・ INDEX CSET = 1 のときに、INDEX CHANGE 設定コマンドを書き込んだ
- ・ INDEX CBUSY = 1 のときに、INDEX CHANGE 実行コマンドを書き込んだ

D1 : COMREG CLR ERROR(応用機能)

コマンド予約機能で格納している実行待ちの予約コマンドをクリアしたことを示します。

D2 : INC INDEX ERROR

相対アドレスのオーバーフローで、INC INDEX ドライブを終了したことを示します。

・相対アドレスがオーバーフローしているときに、INC INDEX CHANGE 指令を検出した

D3 : ABS INDEX ERROR

アドレスカウンタのオーバーフローで、ABS INDEX ドライブを終了したことを示します。

- ・ ABS INDEX ドライブ実行中に、アドレスカウンタのオーバーフローを検出した
- ・ アドレスカウンタがオーバーフローしているときに、ABS INDEX CHANGE 指令を検出した

- D4 : INDEX CHANGE ERROR(応用機能)
反転動作が必要な INDEX CHANGE 指令を検出したことを示します。
・反転動作が必要な INDEX CHANGE 指令を検出した
・ ABS INDEX ドライブ中に、アドレスカウンタの現在位置が変更され、反転動作が必要になった
- D5 : CHANGE CLR ERROR(応用機能)
実行待ちの INDEX CHANGE 指令を無効にしたことを示します。
- D6 : CPP STOP ERROR(応用機能)
補間ドライブのメイン軸の CPP STOP 機能でドライブを終了したことを示します。
- D7 : EXT PULSE ERROR
外部パルス出力機能を実行中に、正常な外部パルス出力ができなかったことを示します。
なお、コントローラドライバには外部パルス出力機能はありません。
・アクティブ幅の2倍の時間内に、次のカウントタイミングが入力した
- D8 : FSEND ERROR
BUSY = 1 のときに、DRIVE STATUS1 PORT の FSEND = 1 を検出したことを示します。
- D9 : LSEND ERROR
BUSY = 1 のときに、DRIVE STATUS1 PORT の LSEND = 1 を検出したことを示します。
- D10 : SSEND ERROR
BUSY = 1 のときに、DRIVE STATUS1 PORT の SSEND = 1 を検出したことを示します。
- D11 : ADDRESS OVF ERROR
BUSY = 1 のときに、DRIVE STATUS4 PORT の ADDRESS OVF = 1 を検出したことを示します。
- D12 : PULSE OVF ERROR
DRIVE STATUS4 PORT の PULSE OVF = 1 を検出したことを示します。
- D13 : DALM ERROR
DRIVE STATUS2 PORT の DALM = 1 を検出したことを示します。
- D14 : FSSTOP ERROR
DRIVE STATUS2 PORT の FSSTOP = 1 を検出したことを示します。
- ・ ERROR STATUS READ コマンドを実行すると、ERROR STATUS D15--D0 を DRIVE DATA1 PORT (READ)にセットします。
・ FSEND, LSEND, SSEND フラグが "1" でも、次の BUSY = 0 1 ではエラー検出されません。
BUSY = 0 1 と同時に、FSEND, LSEND, SSEND = 1 0 になります。

5-1-7. 速度・設定データの読み出し

(1) MCC SPEED READ

MCC07 が現在出力しているドライブパルス速度を読み出します。
このコマンドの実行は常時可能です。



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
← パルス速度データ →															

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
← パルス速度データ →															

・読み出すデータは、「ドライブパルス速度 (Hz) の 10 倍」のパルス速度データです。
ドライブパルス速度 (Hz) = パルス速度データ / 10

MCC SPEED READ コマンドを実行すると、MCC07 が現在出力しているドライブパルス速度の 10 倍のデータを DRIVE DATA1, 2 PORT (READ) にセットします。

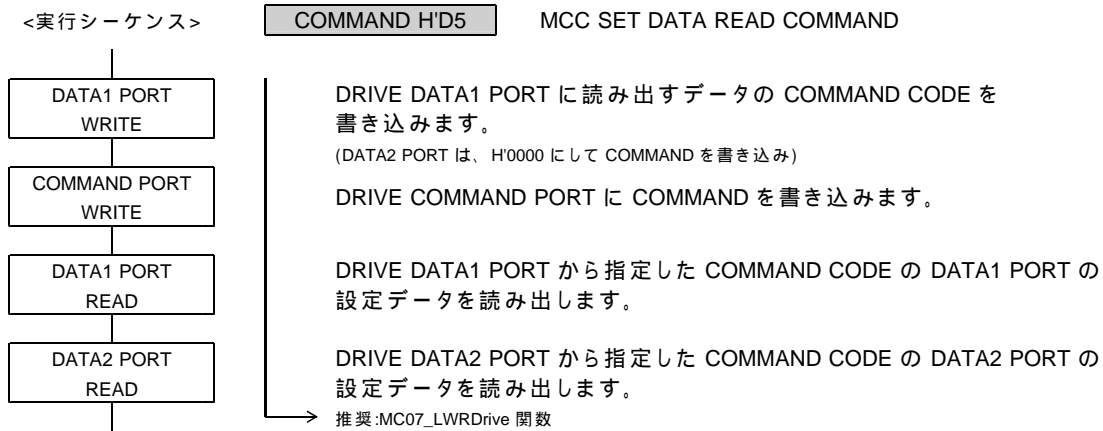
補間ドライブ実行中は、メイン軸のパルス速度の読み出しのみ有効です。
メイン軸から読み出すデータは、補間ドライブの基本となる加減速パルスの速度です。

以下の場合には、パルス速度の読み出しは無効です。

- ・ DRIVE STATUS1 PORT の DRIVE = 0 のとき
- ・ DRIVE STATUS1 PORT の EXT PULSE = 1 のとき (外部パルス出力機能の実行中)

(2) MCC SET DATA READ

MCC07 に設定した設定データを読み出します。
このコマンドの実行は常時可能です。



読み出すデータは、MCC07 内部で範囲補正していない設定データです。
電源投入後は、各機能の設定データの初期値が読み出されます。

SET DATA READ コマンドを実行すると、
指定したコマンドの設定データを DRIVE DATA1, 2 PORT (READ) にセットします。
設定データがないコマンドの読み出しデータは、不定になります。

読み出しできるドライブパラメータと各機能の設定データ

COMMAND CODE	汎用コマンド名称	機能
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, SS0 の設定
H'03	SPEC INITIALIZE3	DRST, DEND, DALM, 自動減速の設定
H'05	FSPD SET *	第1パルスのパルス周期の設定
H'06	HIGH SPEED SET *	加減速ドライブの速度倍率と最高速度の設定
H'07	LOW SPEED SET *	加減速ドライブの開始速度と終了速度の設定
H'08	RATE SET *	加減速カーブの変速周期の設定
H'09	SCAREA SET *	加減速カーブのS字変速領域の設定
H'0A	DOWN PULSE ADJUST *	減速パルス数のオフセット設定
H'0C	JSPD SET *	JOGドライブのパルス速度の設定
H'0D	JOG PULSE SET *	JOGドライブのパルス数の設定
H'0F	ORIGIN SPEC SET *	ORIGINドライブの動作仕様の設定
H'20	CP SPEC SET *	補間パルスの入出力機能の設定
H'22	LONG POSITION SET *	直線補間ドライブの長軸アドレスの設定
H'23	SHORT POSITION SET *	直線補間ドライブの短軸アドレスの設定
H'28	CIRCULAR XPOSITION SET *	円弧補間ドライブのX座標アドレスの設定
H'29	CIRCULAR YPOSITION SET *	円弧補間ドライブのY座標アドレスの設定
H'2A	CIRCULAR PULSE SET *	円弧補間ドライブの短軸パルス数の設定

* 応用機能です。

読み出しできる各機能の設定データ

COMMAND CODE	特殊コマンド名称	機能
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'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'98	CNTINT COMPARE REGISTER1 SET	CNTINT のコンペアレジスタ1の設定
H'99	CNTINT COMPARE REGISTER2 SET	CNTINT のコンペアレジスタ2の設定
H'9A	CNTINT COMPARE REGISTER3 SET	CNTINT のコンペアレジスタ3の設定
H'9C	CNTINT COMP1 ADD DATA SET	CNTINT の COMP1 ADD データの設定
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'C0	UDC SPEC SET *	UP/DOWN/CONST の変更動作点の設定
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'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 *	SOUT の設定
H'F4	HARD INITIALIZE4 *	軸制御部のデジタルフィルタの設定
H'F5	HARD INITIALIZE5 *	軸制御部のデジタルフィルタの設定
H'F6	HARD INITIALIZE6 *	外部パルスのデジタルフィルタの設定
H'F7	HARD INITIALIZE7 *	入力信号のアクティブ論理の選択
H'FC	SIGNAL OUT	汎用出力信号の操作

* 応用機能です。

COMMAND CODE H'88, H'98 HA8 の COMPARE REGISTER1 SET コマンドのデータは、自動加算機能で加算された現在値が読み出されます。

5-1-8. その他

(1) NO OPERATION

機能はありません。



このコマンドの実行により、以下の STATUS フラグがクリアされます。

- ・ DRIVE STATUS1 PORT の DRVEND フラグ
- ・ DRIVE STATUS1 PORT の LSEND フラグ
- ・ DRIVE STATUS1 PORT の SSEND フラグ
- ・ DRIVE STATUS1 PORT の FSEND フラグ

5-2. カウンタコマンド

5-2-1. アドレスカウンタの設定

(1) ADDRESS COUNTER INITIALIZE1

アドレスカウンタの各機能を設定します。
このコマンドの実行は常時可能です。



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
AUTO	AUTO	COMP	COMP	ADRINT	ADRINT	ADRINT	ADRINT
ADD	CLEAR	GATE	GATE	PULSE	PULSE	ADRINT	ADRINT
ENABLE	ENABLE	TYPE1	TYPE0	TYPE1	TYPE0	TYPE1	TYPE0

D7	D6	D5	D4	D3	D2	D1	D0
EXT	EXT	EXT	EXT	EXT	EXT	COUNT	COUNT
COUNT	PULSE	PULSE	PULSE	COUNT	COUNT	PULSE	PULSE
DIRECTION	TYPE2	TYPE1	TYPE0	TYPE1	TYPE0	SEL1	SEL0

電源投入後の初期値は H'0030 (アンダーライン側) です。

D0 : COUNT PULSE SEL0

D1 : COUNT PULSE SEL1

カウンタのカウントパルスを選択します。

選択したカウントパルスは、CWP, CCWP 端子から出力するドライブパルスになります。

< X 軸, Z 軸に設定する場合 >

SEL1	SEL0	カウントパルス	カウント方向
<u>0</u>	<u>0</u>	自軸(X,Z 軸)の発生パルスをカウントする	+ 方向出力でカウントアップ - 方向出力でカウントダウン
0	1	他軸(Y,A 軸)の発生パルスをカウントする	
1	0	自軸(X,Z 軸)の外部パルス信号をカウントする	EXT COUNT DIRECTION で選択
1	1	他軸(Y,A 軸)の外部パルス信号をカウントする	

< Y 軸, A 軸に設定する場合 >

SEL1	SEL0	カウントパルス	カウント方向
<u>0</u>	<u>0</u>	自軸(Y,A 軸)の発生パルスでカウントする	+ 方向出力でカウントアップ - 方向出力でカウントダウン
0	1	他軸(X,Z 軸)の発生パルスをカウントする	
1	0	他軸(X,Z 軸)の外部パルス信号をカウントする	EXT COUNT DIRECTION で選択
1	1	自軸(Y,A 軸)の外部パルス信号でカウントする	

- ・ UC-766 以外の製品は、外部パルスをカウントすることはできません。
- ・ DRIVE STATUS1 PORT の EXT PULSE = 0、BUSY = 1 のときに「10」、「11」を選択した場合は、実行中の処理を終了した後 (BUSY = 0) に、EXT PULSE = 1、BUSY = 1 になります。

D2 : EXT COUNT TYPE0

D3 : EXT COUNT TYPE1

外部パルス信号入力のカウント方法を選択します。

TYPE1	TYPE0	カウント方法	パルス入力方式
0	0	EA, EB を1 逓倍でカウントする	位相差信号入力
0	1	EA, EB を2 逓倍でカウントする	
1	0	EA, EB を4 逓倍でカウントする	
1	1	EA で + 方向のカウント、EB で - 方向のカウント	独立方向パルス入力

D4 : EXT PULSE TYPE0

D5 : EXT PULSE TYPE1

D6 : EXT PULSE TYPE2

外部パルス信号のカウントタイミングのアクティブ幅を選択します。

TYPE2	TYPE1	TYPE0	アクティブ幅	TYPE2	TYPE1	TYPE0	アクティブ幅
0	0	0	100 ns	1	0	0	2.0 μs
0	0	1	200 ns	1	0	1	5.0 μs
0	1	0	500 ns	1	1	0	10 μs
0	1	1	1.0 μs	1	1	1	20 μs

EXT COUNT TYPE で選択した外部パルス信号のカウントタイミングを、
EXT PULSE TYPE で選択したアクティブ幅のパルスに変換して、
アドレスカウンタの COUNT PULSE SEL ブロックに入力します。

カウンタのカウントパルスを「外部パルス信号」に設定した場合は、
選択したアクティブ幅のパルスが、
カウンタのカウントパルスおよび CWP, CCWP 端子の出力パルスになります。

D7 : EXT COUNT DIRECTION

外部パルス入力 EA, EB のカウント方向を選択します。

0 : 外部パルス信号の入力方向と同じ方向にカウントする

1 : 外部パルス信号の入力方向と逆の方向にカウントする

- ・「0 : 同じ方向」の場合は、 + 方向入力で、 + 方向カウント (+ 方向パルス出力)、
- 方向入力で、 - 方向カウント (- 方向パルス出力) になります。
- ・「1 : 逆の方向」の場合は、 + 方向入力で、 - 方向カウント (- 方向パルス出力)、
- 方向入力で、 + 方向カウント (+ 方向パルス出力) になります。
- ・カウンタのカウントパルスを「外部パルス信号」に設定した場合は、選択したカウント方向がカウンタの
カウント方向、およびドライブパルスの出力方向になります。

D8 : ADRINT TYPE0

D9 : ADRINT TYPE1

COMP1, 2, 3 の一致出力の出力仕様を選択します。

TYPE1	TYPE0	COMP1, 2, 3 の一致出力の出力仕様	クリア条件
0	0	一致出力をレベルラッチして出力する	検出条件が不一致のときに DRIVE STATUS4 PORT のリード終了でクリア
0	1	一致出力をエッジラッチして出力する	DRIVE STATUS4 PORT の リード終了でクリア
1	0	一致出力をそのままスルーで出力する	検出条件の不一致でクリア
1	1	設定禁止	-

レベルラッチの場合は、検出条件が一致している間はクリアできません。
"10" のスルー出力の場合は、ADRINT PULSE TYPE で最小出力幅を選択します。

D10 : ADRINT PULSE TYPE0

D11 : ADRINT PULSE TYPE1

ADRINT TYPE = "10" (スルー出力) に設定している場合に有効です。

COMP1, 2, 3 の一致出力の最小出力幅を選択します。

TYPE1	TYPE0	一致出力の最小出力幅
0	0	200 ns
0	1	10 μs
1	0	100 μs
1	1	1,000 μs

スルー出力にオートクリア機能または自動加算機能を併用した場合も、この最小出力幅を出力します。この最小出力幅はリトリガ出力です。

D12 : COMP GATE TYPE0

D13 : COMP GATE TYPE1

COMP1, 2, 3 の一致出力の、合成出力を選択します。

TYPE1	TYPE0	一致出力の合成出力				
0	0	COMP1	OR	(COMP2	OR	COMP3)
0	1	COMP1	OR	(COMP2	AND	COMP3)
1	0	COMP1	AND	(COMP2	OR	COMP3)
1	1	COMP1	AND	(COMP2	AND	COMP3)

OR : 論理和、AND : 論理積

D14 : AUTO CLEAR ENABLE

COMP1 のオートクリア機能で、カウンタを「クリアする / クリアしない」を選択します。

0 : COMP1 の一致出力でカウンタをクリアしない

1 : COMP1 の一致出力でカウンタをクリアする

オートクリア機能

COMP1 の一致検出と同時に、アドレスカウンタのデータを "0" にクリアします。

COMP1 の一致出力がスルー出力のときは、一致出力の最小出力幅を出力します。

D15 : AUTO ADD ENABLE

COMP1 の自動加算機能で、検出データを「再設定する / 再設定しない」を選択します。

0 : COMP1 の一致出力でデータを再設定しない

1 : COMP1 の一致出力でデータを再設定する

自動加算機能

COMP1 の一致検出と同時に、COMP1 ADD データに設定されているデータを、

COMPARE REGISTER1 のデータに加算して、COMPARE REGISTER1 を再設定します。

$$\text{COMPARE REGISTER1} \leq \text{COMPARE REGISTER1} + \text{COMP1 ADD データ}$$

COMP1 の一致出力がスルー出力のときは、一致出力の最小出力幅を出力します。

(2) ADDRESS COUNTER INITIALIZE2

アドレスカウンタの各機能を設定します。
このコマンドの実行は常時可能です。



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
COMP3 TYPE1	COMP3 TYPE0	COMP2 TYPE1	COMP2 TYPE0	COMP3 STOP TYPE1	COMP3 STOP TYPE0	COMP3 STOP ENABLE	COMP3 INT ENABLE
D7	D6	D5	D4	D3	D2	D1	D0
COMP2 STOP TYPE1	COMP2 STOP TYPE0	COMP2 STOP ENABLE	COMP2 INT ENABLE	-	COMP1 STOP TYPE	COMP1 STOP ENABLE	COMP1 INT ENABLE

電源投入後の初期値は H'0000 (アンダーライン側) です。

D0 : COMP1 INT ENABLE

COMP1 の一致出力を、ADRINT に「出力する / 出力しない」を選択します。

- 0 : COMP1 の一致出力を ADRINT に出力しない
- 1 : COMP1 の一致出力を ADRINT に出力する

D1 : COMP1 STOP ENABLE

COMP1 の一致出力による停止機能を「実行する / 実行しない」を選択します。

- 0 : COMP1 の一致出力の停止機能を実行しない
- 1 : COMP1 の一致出力の停止機能を実行する

D2 : COMP1 STOP TYPE

COMP1 の一致出力による停止機能を選択します。

- 0 : 一致出力でパルス出力を即時停止する
- 1 : 一致出力でパルス出力を減速停止する

・COMP1 の検出条件は、「カウンタの値 = COMPARE REGISTER1 の値」です。

D4 : COMP2 INT ENABLE

COMP2 の一致出力を、ADRINT に「出力する / 出力しない」を選択します。

- 0 : COMP2 の一致出力を ADRINT に出力しない
- 1 : COMP2 の一致出力を ADRINT に出力する

D5 : COMP2 STOP ENABLE

COMP2 の一致出力による停止機能を「実行する / 実行しない」を選択します。

- 0 : COMP2 の一致出力の停止機能を実行しない
- 1 : COMP2 の一致出力の停止機能を実行する

D6 : COMP2 STOP TYPE0

D7 : COMP2 STOP TYPE1

COMP2 の一致出力による停止機能を選択します。

TYPE1	TYPE0	COMP2 の停止機能
0	0	一致出力でパルス出力を即時停止する
0	1	一致出力でパルス出力を減速停止する
1	0	一致出力で、+ (CW)方向のパルス出力を即時停止する
1	1	一致出力で、+ (CW)方向のパルス出力を減速停止する

D8 : COMP3 INT ENABLE

COMP3 の一致出力を、ADRINT に「出力する / 出力しない」を選択します。

0 : COMP3 の一致出力を ADRINT に出力しない

1 : COMP3 の一致出力を ADRINT に出力する

D9 : COMP3 STOP ENABLE

COMP3 の一致出力による停止機能を「実行する / 実行しない」を選択します。

0 : COMP3 の一致出力の停止機能を実行しない

1 : COMP3 の一致出力の停止機能を実行する

D10 : COMP3 STOP TYPE0

D11 : COMP3 STOP TYPE1

COMP3 の一致出力による停止機能を選択します。

TYPE1	TYPE0	COMP3 の停止機能
0	0	一致出力でパルス出力を即時停止する
0	1	一致出力でパルス出力を減速停止する
1	0	一致出力で、- (CCW)方向のパルス出力を即時停止する
1	1	一致出力で、- (CCW)方向のパルス出力を減速停止する

D12 : COMP2 TYPE0

D13 : COMP2 TYPE1

COMP2 の検出条件を選択します。

TYPE1	TYPE0	COMP2 の検出条件
0	0	カウンタの値 = COMPARE REGISTER2 の値
0	1	カウンタの値 < COMPARE REGISTER2 の値
1	0	カウンタの値 > COMPARE REGISTER2 の値
1	1	設定禁止

D14 : COMP3 TYPE0

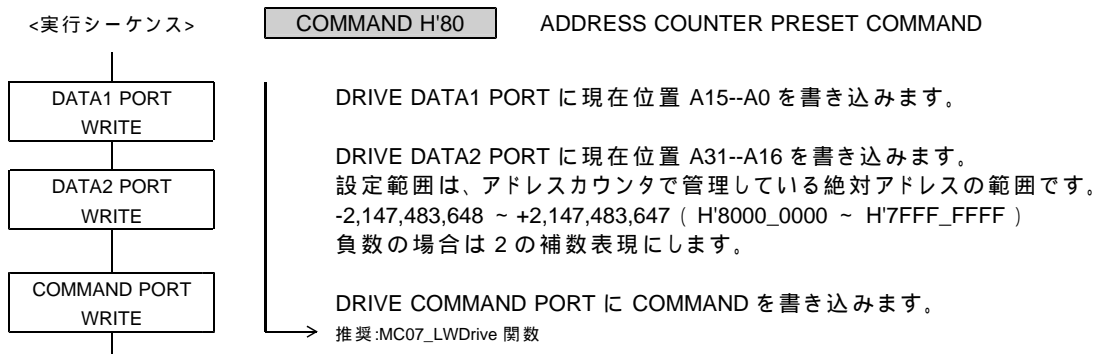
D15 : COMP3 TYPE1

COMP3 の検出条件を選択します。

TYPE1	TYPE0	COMP3 の検出条件
0	0	カウンタの値 = COMPARE REGISTER3 の値
0	1	カウンタの値 < COMPARE REGISTER3 の値
1	0	カウンタの値 > COMPARE REGISTER3 の値
1	1	設定禁止

(3) ADDRESS COUNTER PRESET

アドレスカウンタの現在位置を設定します。
このコマンドは常時実行可能です。



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
A15	← 現在位置 →														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	← 現在位置 →														A16

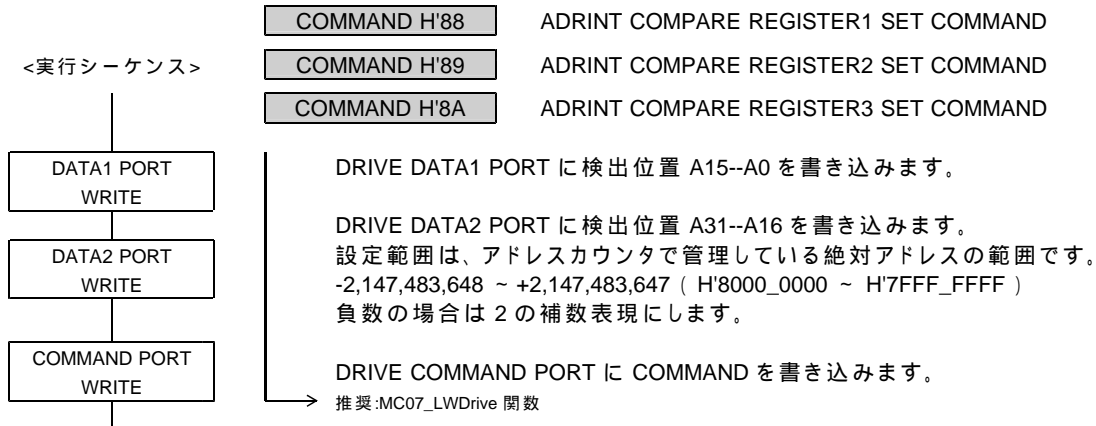
電源投入後の初期値は H'0000_0000 です。

現在位置には、H'8000_0000 を設定することもできます。
ただし、H'8000_0000 を設定すると、DRIVE STATUS4 PORT の ADDRESS OVF = 1 になります。

以下の場合、エラーになり、ERROR STATUS の INDEX CHANGE ERROR = 1 にします。
ABS INDEX ドライブ中に、ADDRESS COUNTER PRESET コマンドの実行で現在位置が変更され、
反転動作が必要な状態になったとき。

(4) ADRINT COMPARE REGISTER1,2,3 SET

アドレスカウンタの COMPARE REGISTER1, 2, 3 に検出位置を設定します。
このコマンドの実行は常時可能です。



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
A15	← 検出位置 →														A0

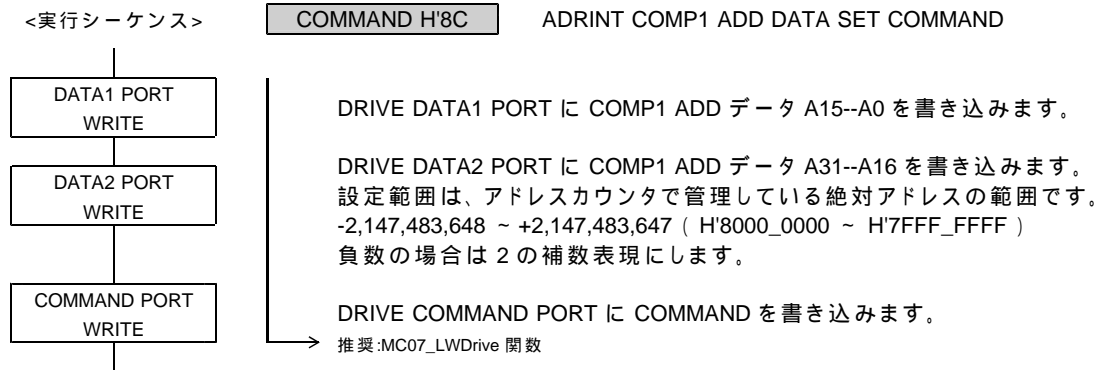
DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A31	← 検出位置 →														A16

電源投入後の初期値は H'8000_0000 です。

(5) ADRINT COMP1 ADD DATA SET

アドレスカウンタの COMP1 の加算データを設定します。
このコマンドは常時実行可能です。



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
A15 ← COMP1 ADD 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 ← COMP1 ADD データ → A16															

電源投入後の初期値は H'0000_0000 です。

5-2-2. パルスカウンタの設定

(1) PULSE COUNTER INITIALIZE1

パルスカウンタの各機能を設定します。このコマンドの実行は常時可能です。

ORIGIN ドライブ中は、パルスカウンタを使用することはできません。



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
AUTO	AUTO	COMP	COMP	CNTINT	CNTINT	CNTINT	CNTINT
ADD	CLEAR	GATE	GATE	PULSE	PULSE	CNTINT	CNTINT
ENABLE	ENABLE	TYPE1	TYPE0	TYPE1	TYPE0	TYPE1	TYPE0

D7	D6	D5	D4	D3	D2	D1	D0
EXT				EXT	EXT	COUNT	COUNT
COUNT	-	-	-	COUNT	COUNT	PULSE	PULSE
DIRECTION				TYPE1	TYPE0	SEL1	SEL0

電源投入後の初期値は H'0000 (アンダーライン側) です。

D0 : COUNT PULSE SEL0

D1 : COUNT PULSE SEL1

カウンタのカウントパルスを選択します。

< X 軸, Z 軸に設定する場合 >

SEL1	SEL0	カウントパルス	カウント方向
<u>0</u>	<u>0</u>	<u>自軸(X,Z軸)の出力パルスをカウントする</u>	+ 方向出力でカウントアップ - 方向出力でカウントダウン
0	1	他軸(Y,A軸)の出力パルスをカウントする	
1	0	自軸(X,Z軸)の外部パルス信号をカウントする	EXT COUNT DIRECTION で選択
1	1	他軸(Y,A軸)の外部パルス信号をカウントする	

< Y 軸, A 軸に設定する場合 >

SEL1	SEL0	カウントパルス	カウント方向
<u>0</u>	<u>0</u>	<u>自軸(Y,A軸)の出力パルスをカウントする</u>	+ 方向出力でカウントアップ - 方向出力でカウントダウン
0	1	他軸(X,Z軸)の出力パルスをカウントする	
1	0	他軸(X,Z軸)の外部パルス信号をカウントする	EXT COUNT DIRECTION で選択
1	1	自軸(Y,A軸)の外部パルス信号をカウントする	

・ UC-766 以外の製品は、外部パルスをカウントすることはできません。

D2 : EXT COUNT TYPE0

D3 : EXT COUNT TYPE1

外部パルス信号入力のカウント方法を選択します。

TYPE1	TYPE0	カウント方法	パルス入力方式
0	0	EA, EB を 1 通倍でカウントする	位相差信号入力
0	1	EA, EB を 2 通倍でカウントする	
1	0	EA, EB を 4 通倍でカウントする	
1	1	EA で + 方向のカウント、EB で - 方向のカウント	独立方向パルス入力

D4 : 0

D5 : 0

D7 : EXT COUNT DIRECTION

外部パルス信号入力 EA, EB のカウント方向を選択します。

0 : 外部パルス信号の入力方向と同じ方向にカウントする

1 : 外部パルス信号の入力方向と逆の方向にカウントする

- ・「0 : 同じ方向」の場合は、 + 方向入力で、 + 方向カウント(+ 方向パルス出力)、
- 方向入力で、 - 方向カウント(- 方向パルス出力)になります。
- ・「1 : 逆の方向」の場合は、 + 方向入力で、 - 方向カウント(- 方向パルス出力)、
- 方向入力で、 + 方向カウント(+ 方向パルス出力)になります。
- ・カウンタのカウントパルスを「外部パルス信号」に設定した場合は、選択したカウント方向が
カウンタのカウント方向になります。

D8 : CNTINT TYPE0

D9 : CNTINT TYPE1

DRIVE STATUS4 PORT と CNTINT に出力する COMP1, 2, 3 の一致出力の、出力仕様を選択します。

TYPE1	TYPE0	COMP1, 2, 3 の一致出力の出力仕様	クリア条件
0	0	一致出力をレベルラッチして出力する	検出条件が不一致のときに DRIVE STATUS4 PORT のリード終了でクリア
0	1	一致出力をエッジラッチして出力する	DRIVE STATUS4 PORT のリード終了でクリア
1	0	一致出力をそのままスルーで出力する	検出条件の不一致でクリア

- レベルラッチの場合は、検出条件が一致している間はクリアできません。
- "10" スルー出力の場合は、CNTINT PULSE TYPE で最小出力幅を選択します。

D10 : CNTINT PULSE TYPE0

D11 : CNTINT PULSE TYPE1

CNTINT TYPE = "10" (スルー出力) に設定している場合に有効です。

COMP1, 2, 3 の一致出力の最小出力幅を選択します。

TYPE1	TYPE0	一致出力の最小出力幅
0	0	200 ns
0	1	10 μs
1	0	100 μs
1	1	1,000 μs

スルー出力にオートクリア機能または自動加算機能を併用した場合も、この最小出力幅を出力します。この最小出力幅はリトリガ出力です。

D12 : COMP GATE TYPE0

D13 : COMP GATE TYPE1

COMP1, 2, 3 の一致出力の合成出力を選択します。

TYPE1	TYPE0	一致出力の合成出力				
0	0	COMP1	OR	(COMP2	OR	COMP3)
0	1	COMP1	OR	(COMP2	AND	COMP3)
1	0	COMP1	AND	(COMP2	OR	COMP3)
1	1	COMP1	AND	(COMP2	AND	COMP3)

OR : 論理和、AND : 論理積

D14 : AUTO CLEAR ENABLE

COMP1 のオートクリア機能で、カウンタを「クリアする / クリアしない」を選択します。

0 : COMP1 の一致出力でカウンタをクリアしない

1 : COMP1 の一致出力でカウンタをクリアする

オートクリア機能

COMP1 の一致検出と同時に、パルスカウンタのデータを "0" にクリアします。

COMP1 の一致出力がスルー出力のときは、一致出力の最小出力幅を出力します。

D15 : AUTO ADD ENABLE

COMP1 の自動加算機能で、検出データを「再設定する / 再設定しない」を選択します。

0 : COMP1 の一致出力でデータを再設定しない

1 : COMP1 の一致出力でデータを再設定する

自動加算機能

COMP1 の一致検出と同時に、COMP1 ADD データに設定されているデータを、

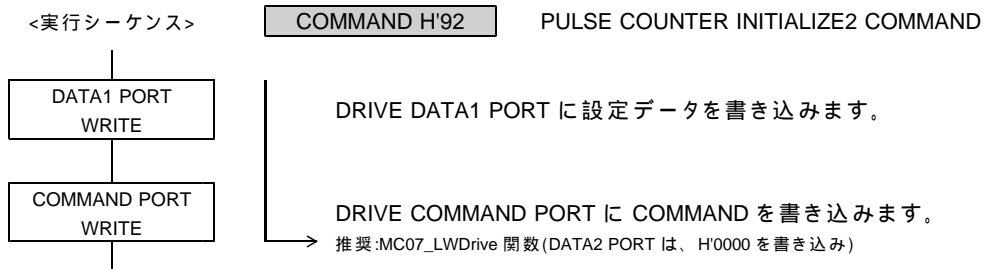
COMPARE REGISTER1 のデータに加算して、COMPARE REGISTER1 を再設定します。

$$\text{COMPARE REGISTER1} \leq \text{COMPARE REGISTER1} + \text{COMP1 ADD データ}$$

COMP1 の一致出力がスルー出力のときは、一致出力の最小出力幅を出力します。

(2) PULSE COUNTER INITIALIZE2

パルスカウンタの各機能を設定します。
このコマンドの実行は常時可能です。



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
COMP3 TYPE1	COMP3 TYPE0	COMP2 TYPE1	COMP2 TYPE0	COMP3 STOP TYPE1	COMP3 STOP TYPE0	COMP3 STOP ENABLE	COMP3 INT ENABLE
D7	D6	D5	D4	D3	D2	D1	D0
COMP2 STOP TYPE1	COMP2 STOP TYPE0	COMP2 STOP ENABLE	COMP2 INT ENABLE	-	COMP1 STOP TYPE	COMP1 STOP ENABLE	COMP1 INT ENABLE

電源投入後の初期値は H'0000 (アンダーライン側) です。

D0 : COMP1 INT ENABLE

COMP1 の一致出力を、CNTINT に「出力する / 出力しない」を選択します。
0 : COMP1 の一致出力を CNTINT に出力しない
 1 : COMP1 の一致出力を CNTINT に出力する

D1 : COMP1 STOP ENABLE

COMP1 の一致出力による停止機能を「実行する / 実行しない」を選択します。
0 : COMP1 の一致出力の停止機能を実行しない
 1 : COMP1 の一致出力の停止機能を実行する

D2 : COMP1 STOP TYPE

COMP1 の一致出力による停止機能を選択します。
0 : 一致出力でパルス出力を即時停止する
 1 : 一致出力でパルス出力を減速停止する

・COMP1 の検出条件は、「カウンタの値 = COMPARE REGISTER1 の値」です。

D4 : COMP2 INT ENABLE

COMP2 の一致出力を、CNTINT に「出力する / 出力しない」を選択します。
0 : COMP2 の一致出力を CNTINT に出力しない
 1 : COMP2 の一致出力を CNTINT に出力する

D5 : COMP2 STOP ENABLE

COMP2 の一致出力による停止機能を「実行する / 実行しない」を選択します。
0 : COMP2 の一致出力の停止機能を実行しない
 1 : COMP2 の一致出力の停止機能を実行する

D6 : COMP2 STOP TYPE0

D7 : COMP2 STOP TYPE1

COMP2 の一致出力による停止機能を選択します。

TYPE1	TYPE0	COMP2 の停止機能
0	0	一致出力でパルス出力を即時停止する
0	1	一致出力でパルス出力を減速停止する
1	0	一致出力で、+ (CW)方向のパルス出力を即時停止する
1	1	一致出力で、+ (CW)方向のパルス出力を減速停止する

D8 : COMP3 INT ENABLE

COMP3 の一致出力を、CNTINT に「出力する / 出力しない」を選択します。

0 : COMP3 の一致出力を CNTINT に出力しない

1 : COMP3 の一致出力を CNTINT に出力する

D9 : COMP3 STOP ENABLE

COMP3 の一致出力による停止機能を「実行する / 実行しない」を選択します。

0 : COMP3 の一致出力の停止機能を実行しない

1 : COMP3 の一致出力の停止機能を実行する

D10 : COMP3 STOP TYPE0

D11 : COMP3 STOP TYPE1

COMP3 の一致出力による停止機能を選択します。

TYPE1	TYPE0	COMP3 の停止機能
0	0	一致出力でパルス出力を即時停止する
0	1	一致出力でパルス出力を減速停止する
1	0	一致出力で、- (CCW)方向のパルス出力を即時停止する
1	1	一致出力で、- (CCW)方向のパルス出力を減速停止する

D12 : COMP2 TYPE0

D13 : COMP2 TYPE1

COMP2 の検出条件を選択します。

TYPE1	TYPE0	COMP2 の検出条件
0	0	カウンタの値 = COMPARE REGISTER2 の値
0	1	カウンタの値 < COMPARE REGISTER2 の値
1	0	カウンタの値 > COMPARE REGISTER2 の値
1	1	設定禁止

D14 : COMP3 TYPE0

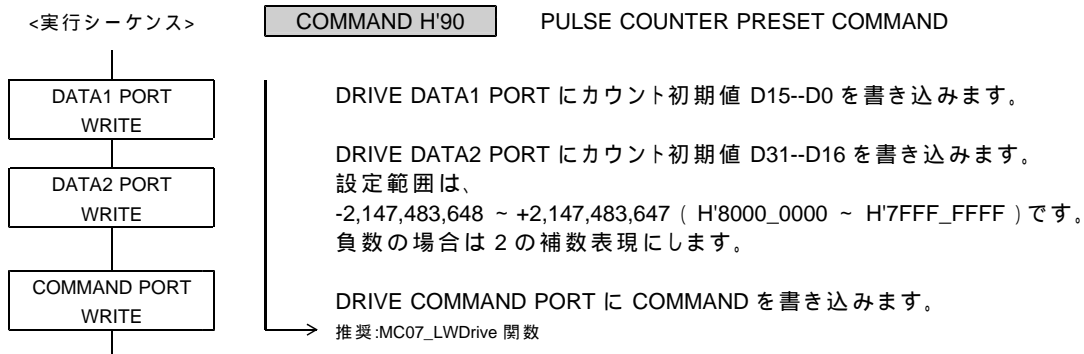
D15 : COMP3 TYPE1

COMP3 の検出条件を選択します。

TYPE1	TYPE0	COMP3 の検出条件
0	0	カウンタの値 = COMPARE REGISTER3 の値
0	1	カウンタの値 < COMPARE REGISTER3 の値
1	0	カウンタの値 > COMPARE REGISTER3 の値
1	1	設定禁止

(3) PULSE COUNTER PRESET

パルスカウンタのカウンタ初期値を設定します。
このコマンドは常時実行可能です。



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
D15 ← カウンタ初期値 → D0															

DRIVE DATA2 PORT の設定データ

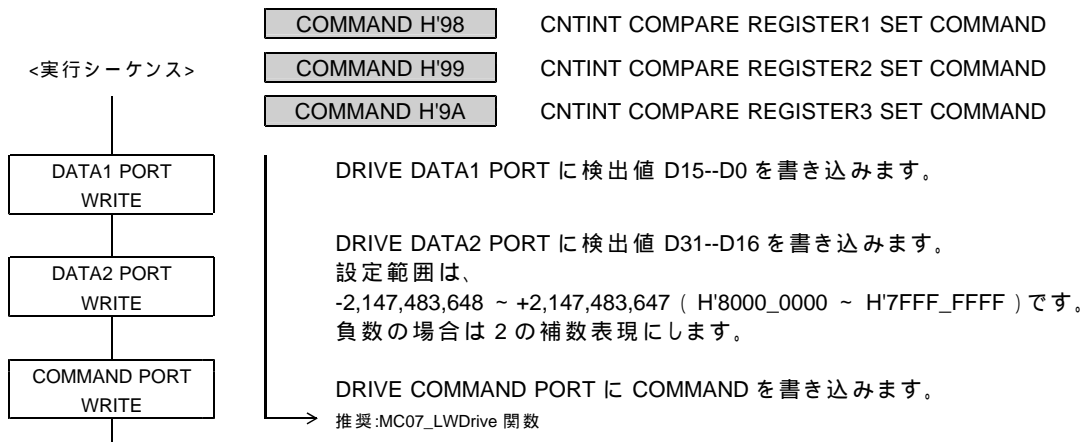
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D31 ← カウンタ初期値 → D16															

電源投入後の初期値は H'0000_0000 です。

カウンタに設定する初期値には、H'8000_0000 を設定することもできます。
ただし、H'8000_0000 を設定すると、DRIVE STATUS4 PORT の PULSE OVF = 1 になります。

(4) CNTINT COMPARE REGISTER1,2,3 SET

パルスカウンタの COMPARE REGISTER1, 2, 3 に検出値を設定します。
このコマンドの実行は常時可能です。



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
D15	← 検出値 →														D0

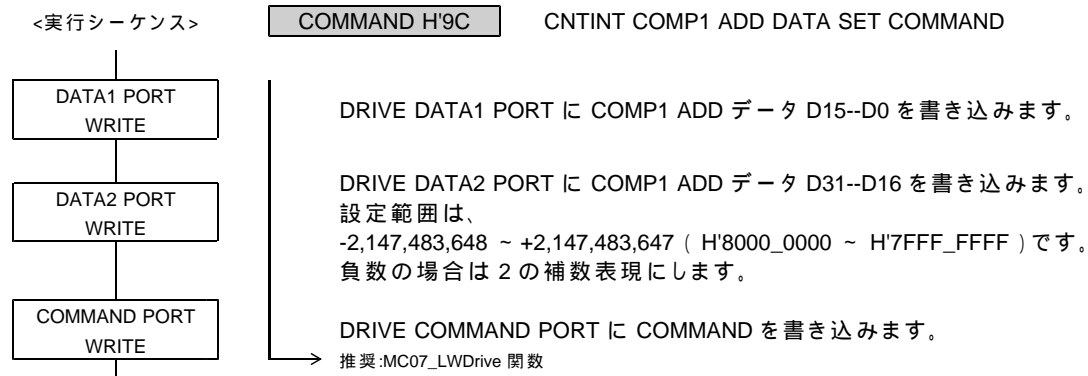
DRIVE DATA2 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D31	← 検出値 →														D16

電源投入後の初期値は H'8000_0000 です。

(5) CNTINT COMP1 ADD DATA SET

パルスカウンタの COMP1 の加算データを設定します。
このコマンドは常時実行可能です。



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
← COMP1 ADD データ →															

DRIVE DATA2 PORT の設定データ

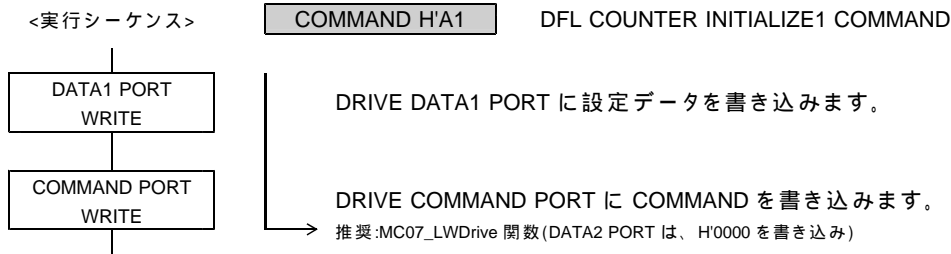
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
← COMP1 ADD データ →															

電源投入後の初期値は H'0000_0000 です。

5-2-3. パルス偏差カウンタの設定

(1) DFL COUNTER INITIALIZE1

パルス偏差カウンタの各機能を設定します。
このコマンドの実行は常時可能です。



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
AUTO ADD ENABLE	AUTO CLEAR ENABLE	COMP GATE TYPE1	COMP GATE TYPE0	DFLINT PULSE TYPE1	DFLINT PULSE TYPE0	DFLINT TYPE1	DFLINT TYPE0
D7	D6	D5	D4	D3	D2	D1	D0
DIVISION TYPE	TIMER START TYPE2	TIMER START TYPE1	TIMER START TYPE0	EXT COUNT TYPE1	EXT COUNT TYPE0	COUNT PULSE SEL1	COUNT PULSE SEL0

電源投入後の初期値は H'0000 (アンダーライン側) です。

D0 : COUNT PULSE SEL0

D1 : COUNT PULSE SEL1

カウンタのカウントパルスを選択します。

< X 軸, Z 軸に設定する場合 >

SEL1	SEL0	カウントパルス 1	カウントパルス 2
0	0	自軸(X,Z 軸)の外部パルス信号	自軸(X,Z 軸)の出力パルス
0	1	他軸(Y,A 軸)の外部パルス信号	自軸(X,Z 軸)の出力パルス
1	0	他軸(Y,A 軸)の外部パルス信号	自軸(X,Z 軸)の外部パルス信号
1	1	20 MHz クロック	- (なし)

< Y 軸, A 軸に設定する場合 >

SEL1	SEL0	カウントパルス 1	カウントパルス 2
0	0	自軸(Y,A 軸)の外部パルス信号	自軸(Y,A 軸)の出力パルス
0	1	他軸(X,Z 軸)の外部パルス信号	自軸(Y,A 軸)の出力パルス
1	0	他軸(X,Z 軸)の外部パルス信号	自軸(Y,A 軸)の外部パルス信号
1	1	20 MHz クロック	- (なし)

・ UC-766 以外の製品は、外部パルスをカウントすることはできません。

カウント方向

・ カウントパルス 1 : + 方向入力でカウントアップ、- 方向入力でカウントダウン

・ カウントパルス 2 : - 方向入力でカウントアップ、+ 方向入力でカウントダウン

タイマ機能

"11" に設定すると、カウントパルス 1 を、+ 方向にカウントアップします。

カウントパルス 1 の 20 MHz クロックは、1/1 ~ 1/256 に分周してカウントできます。

D2 : EXT COUNT TYPE0

D3 : EXT COUNT TYPE1

外部パルス信号入力のカウント方法を選択します。

TYPE1	TYPE0	カウント方法	パルス入力方式
0	0	EA, EB を 1 通倍でカウントする	位相差信号入力
0	1	EA, EB を 2 通倍でカウントする	
1	0	EA, EB を 4 通倍でカウントする	
1	1	EA で + 方向のカウント、EB で - 方向のカウント	独立方向パルス入力

D4 : TIMER START TYPE0

D5 : TIMER START TYPE1

D6 : TIMER START TYPE2

COUNT PULSE SEL を "11" に設定している場合に有効です。

タイマ機能のカウントパルス1のカウントを開始するタイミングを選択します。

TYPE2	TYPE1	TYPE0	カウント開始タイミング <レベル検出>
0	0	0	カウントしない (カウントを終了する)
0	0	1	設定禁止
0	1	0	DRIVE STATUS5 PORT の SS0 = 1 でカウントを開始する
0	1	1	DFL COUNTER INITIALIZE1 コマンドの実行でカウントを開始する
1	0	0	設定禁止
1	0	1	設定禁止
1	1	0	設定禁止
1	1	1	設定禁止

・汎用入力 $\overline{IN0}$ 信号から X 軸 SS0、 $\overline{IN1}$ 信号から Y 軸 SS0 が操作できます。
Z 軸と A 軸の SS0 は操作できません。

D7 : DIVISION TYPE

分周するカウントパルスを選択します。

0 : カウントパルス1を分周する

1 : カウントパルス2を分周する

D8 : DFLINT TYPE0

D9 : DFLINT TYPE1

DRIVE STATUS4 PORT と DFLINT に出力する COMP1, 2, 3 の一致出力の、出力仕様を選択します。

TYPE1	TYPE0	COMP1, 2, 3 の一致出力の出力仕様	クリア条件
0	0	一致出力をレベルラッチして出力する	検出条件が不一致のときに DRIVE STATUS4 PORT のリード終了でクリア
0	1	一致出力をエッジラッチして出力する	DRIVE STATUS4 PORT のリード終了でクリア
1	0	一致出力をそのままスルーで出力する	検出条件の不一致でクリア

レベルラッチの場合は、検出条件が一致している間はクリアできません。

"10" スルー出力の場合は、DFLINT PULSE TYPE で最小出力幅を選択します。

D10 : DFLINT PULSE TYPE0

D11 : DFLINT PULSE TYPE1

DFLINT TYPE = "10" (スルー出力) に設定している場合に有効です。
COMP1, 2, 3 の一致出力の最小出力幅を選択します。

TYPE1	TYPE0	一致出力の最小出力幅
0	0	200 ns
0	1	10 μs
1	0	100 μs
1	1	1,000 μs

スルー出力にオートクリア機能または自動加算機能を併用した場合も、この最小出力幅を出力します。この最小出力幅はリトリガ出力です。

D12 : COMP GATE TYPE0

D13 : COMP GATE TYPE1

COMP1, 2, 3 の一致出力の合成出力を選択します。

TYPE1	TYPE0	一致出力の合成出力				
0	0	COMP1	OR	(COMP2	OR	COMP3)
0	1	COMP1	OR	(COMP2	AND	COMP3)
1	0	COMP1	AND	(COMP2	OR	COMP3)
1	1	COMP1	AND	(COMP2	AND	COMP3)

OR : 論理和、AND : 論理積

D14 : AUTO CLEAR ENABLE

COMP1 のオートクリア機能で、カウンタを「クリアする / クリアしない」を選択します。

0 : COMP1 の一致出力でカウンタをクリアしない

1 : COMP1 の一致出力でカウンタをクリアする

オートクリア機能

COMP1 の一致検出と同時に、パルス偏差カウンタのデータを "0" にクリアします。
COMP1 の一致出力がスルー出力のときは、一致出力の最小出力幅を出力します。

D15 : AUTO ADD ENABLE

COMP1 の自動加算機能で、検出データを「再設定する / 再設定しない」を選択します。

0 : COMP1 の一致出力でデータを再設定しない

1 : COMP1 の一致出力でデータを再設定する

自動加算機能

COMP1 の一致検出と同時に、COMP1 ADD データに設定されているデータを、
COMPARE REGISTER1 のデータに加算して、COMPARE REGISTER1 を再設定します。

$$\text{COMPARE REGISTER1} \leq \text{COMPARE REGISTER1} + \text{COMP1 ADD データ}$$

COMP1 の一致出力がスルー出力のときは、一致出力の最小出力幅を出力します。

(2) DFL COUNTER INITIALIZE2

パルス偏差カウンタの各機能を設定します。
このコマンドの実行は常時可能です。



DRIVE DATA1 PORT の設定データ

D15	D14	D13	D12	D11	D10	D9	D8
COMP3 TYPE1	COMP3 TYPE0	COMP2 TYPE1	COMP2 TYPE0	COMP3 DETECT TYPE	COMP3 STOP TYPE	COMP3 STOP ENABLE	COMP3 INT ENABLE
D7	D6	D5	D4	D3	D2	D1	D0
COMP2 DETECT TYPE	COMP2 STOP TYPE	COMP2 STOP ENABLE	COMP2 INT ENABLE	COMP1 DETECT TYPE	COMP1 STOP TYPE	COMP1 STOP ENABLE	COMP1 INT ENABLE

電源投入後の初期値は H'9000 (アンダーライン側) です。

D0 : COMP1 INT ENABLE

COMP1 の一致出力を、DFLINT に「出力する / 出力しない」を選択します。

- 0 : COMP1 の一致出力を DFLINT に出力しない
- 1 : COMP1 の一致出力を DFLINT に出力する

D1 : COMP1 STOP ENABLE

COMP1 の一致出力による停止機能を「実行する / 実行しない」を選択します。

- 0 : COMP1 の一致出力の停止機能を実行しない
- 1 : COMP1 の一致出力の停止機能を実行する

D2 : COMP1 STOP TYPE

COMP1 の一致出力による停止機能を選択します。

- 0 : 一致出力でパルス出力を即時停止する
- 1 : 一致出力でパルス出力を減速停止する

D3 : COMP1 DETECT TYPE

COMP1 が比較するカウンタ値の、検出方法を選択します。

- 0 : カウンタ値を絶対値に変換して比較する
- 1 : カウンタ値を符号付きのまま比較する

・COMP1 の検出条件は、「カウンタの値 = COMPARE REGISTER1 の値」です。

D4 : COMP2 INT ENABLE

COMP2 の一致出力を、DFLINT に「出力する / 出力しない」を選択します。

- 0 : COMP2 の一致出力を DFLINT に出力しない
- 1 : COMP2 の一致出力を DFLINT に出力する

D5 : COMP2 STOP ENABLE

COMP2 の一致出力による停止機能を「実行する / 実行しない」を選択します。

- 0 : COMP2 の一致出力の停止機能を実行しない
- 1 : COMP2 の一致出力の停止機能を実行する

D6 : COMP2 STOP TYPE

COMP2 の一致出力による停止機能を選択します。

- 0 : 一致出力でパルス出力を即時停止する
- 1 : 一致出力でパルス出力を減速停止する

D7 : COMP2 DETECT TYPE

COMP2 が比較するカウンタ値の、検出方法を選択します。

- 0 : カウンタ値を絶対値に変換して比較する
- 1 : カウンタ値を符号付きのまま比較する

D8 : COMP3 INT ENABLE

COMP3 の一致出力を、DFLINT に「出力する / 出力しない」を選択します。

- 0 : COMP3 の一致出力を DFLINT に出力しない
- 1 : COMP3 の一致出力を DFLINT に出力する

D9 : COMP3 STOP ENABLE

COMP3 の一致出力による停止機能を「実行する / 実行しない」を選択します。

- 0 : COMP3 の一致出力の停止機能を実行しない
- 1 : COMP3 の一致出力の停止機能を実行する

D10 : COMP3 STOP TYPE

COMP3 の一致出力による停止機能を選択します。

- 0 : 一致出力でパルス出力を即時停止する
- 1 : 一致出力でパルス出力を減速停止する

D11 : COMP3 DETECT TYPE

COMP3 が比較するカウンタ値の、検出方法を選択します。

- 0 : カウンタ値を絶対値に変換して比較する
- 1 : カウンタ値を符号付きのまま比較する

D12 : COMP2 TYPE0

D13 : COMP2 TYPE1

COMP2 の検出条件を選択します。

TYPE1	TYPE0	COMP2 の検出条件
0	0	カウンタの値 = COMPARE REGISTER2 の値
0	1	カウンタの値 < COMPARE REGISTER2 の値
1	0	カウンタの値 > COMPARE REGISTER2 の値
1	1	設定禁止

取扱説明書

(USB シリーズ MCC07 ユニット編)

D14 : COMP3 TYPE0

D15 : COMP3 TYPE1

COMP3 の検出条件を選択します。

TYPE1	TYPE0	COMP3 の検出条件
0	0	カウンタの値 = COMPARE REGISTER3 の値
0	1	カウンタの値 < COMPARE REGISTER3 の値
1	0	カウンタの値 > COMPARE REGISTER3 の値
1	1	設定禁止

(3) DFL COUNTER INITIALIZE3

パルス偏差カウンタの各機能を設定します。
このコマンドの実行は常時可能です。



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
DIVISION D7	DIVISION D6	DIVISION D5	DIVISION D4	DIVISION D3	DIVISION D2	DIVISION D1	DIVISION D0

電源投入後の初期値は H'0000 です。

D7--D0 : DIVISION D7--D0

DIVISION TYPE で選択したカウントパルスのカウントタイミングの分周数を選択します。

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)

分周したカウントタイミングが、カウンタのカウントパルスになります。

外部パルス信号の分周機能は、COUNT TYPE の逡倍機能と組み合わせて使用できます。

・ UC-766 以外の製品は、外部パルスをカウントすることはできません。

D15--D8 : 0

分周機能(分周数4の場合)

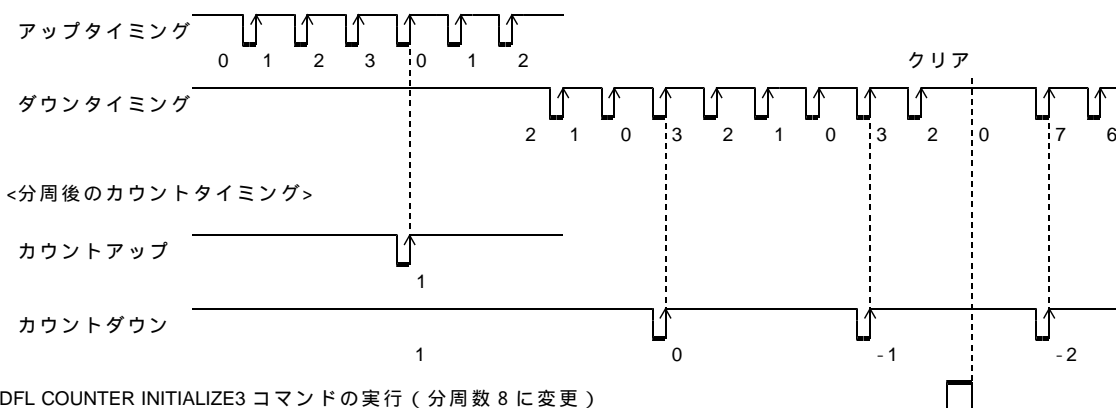
COUNT PULSE SEL と DIVISION TYPE で選択したカウントパルスのカウントタイミングを分周します。

外部パルス信号の場合は、COUNT TYPE で逡倍したカウントタイミングを分周します。

分周したカウントタイミングで、カウンタをアップダウンカウントします。

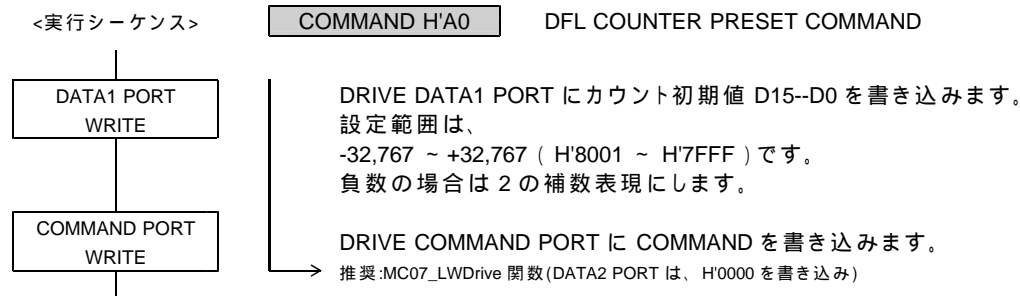
DFL COUNTER INITIALIZE3 コマンドを実行すると、分周中の分周カウント値をクリアします。

<カウントパルスの入力>



(4) DFL COUNTER PRESET

パルス偏差カウンタのカウンタ初期値を設定します。このコマンドは常時実行可能です。



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
← カウンタ初期値 →															

電源投入後の初期値は H'0000 です。

カウンタ初期値には、H'8000 を設定することもできます。
ただし、H'8000 を設定すると、DRIVE STATUS4 PORT の DFL OVF = 1 になります。

カウンタ値は、DFL COUNTER INITIALIZE2 コマンドの各 COMP DETECT TYPE の設定により、絶対値検出または符号付き検出の比較データになります。

COMP DETECT TYPE = 0 の場合 (絶対値検出)

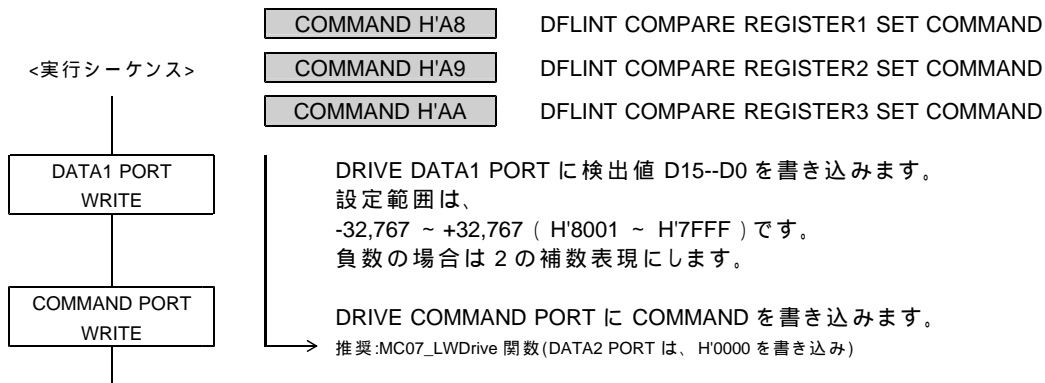
- ・カウンタ値を絶対値に変換して、絶対値に変換した検出値と比較します。
|H'8001 ~ H'FFFF| = +32,767 ~ +1 にします。
|H'0000 ~ H'7FFF| = 0 ~ +32,767 にします。

COMP DETECT TYPE = 1 の場合 (符号付き検出)

- ・カウンタ値はそのまま符号付きの値で、符号付きの検出値と比較します。
H'8001 ~ H'7FFF = -32,767 ~ +32,767 です。

(5) DFLINT COMPARE REGISTER1,2,3 SET

パルス偏差カウンタの COMPARE REGISTER1, 2, 3 に検出値を設定します。
このコマンドの実行は常時可能です。



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
← 検出値 →															

電源投入後の初期値は H'8000 です。

検出値は、DFL COUNTER INITIALIZE2 コマンドの COMP1, 2, 3 の各 COMP DETECT TYPE の設定により、絶対値検出または符号付き検出の比較データになります。

COMP DETECT TYPE = 0 の場合 (絶対値検出)

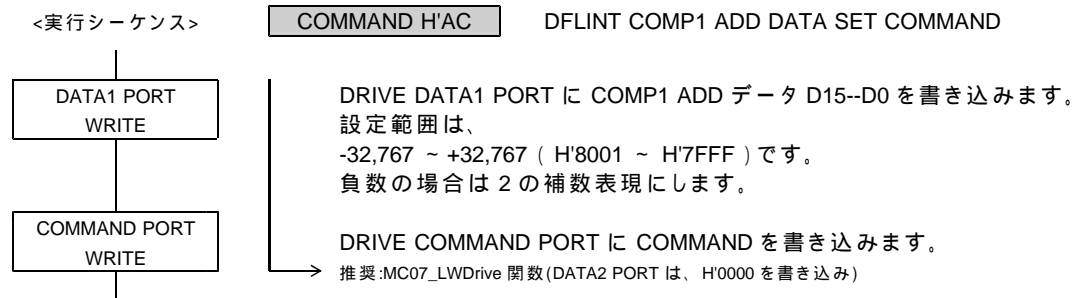
- 検出値を絶対値に変換して、絶対値に変換したカウンタ値と比較します。
|H'8001 ~ H'FFFF| = +32,767 ~ +1 にします。
|H'0000 ~ H'7FFF| = 0 ~ +32,767 にします。

COMP DETECT TYPE = 1 の場合 (符号付き検出)

- 検出値はそのまま符号付きの値で、符号付きのカウンタ値と比較します。
H'8001 ~ H'7FFF = -32,767 ~ +32,767 です。

(6) DFLINT COMP1 ADD DATA SET

パルス偏差カウンタの COMP1 の加算データを設定します。
このコマンドは常時実行可能です。



DRIVE DATA1 PORT の設定データ

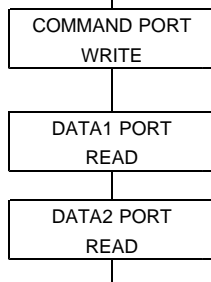
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D15	← COMP1 ADD データ →														D0

電源投入後の初期値は H'0000 です。

5-2-4. カウンタデータの読み出し

カウンタデータの読み出しシーケンス

<実行シーケンス>



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)

DRIVE DATA1 PORT からカウンタデータ D15--D0 を読み出します。

DRIVE DATA2 PORT からカウンタデータ 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
← カウンタデータ →															

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
← カウンタデータ →															

各 COUNTER READ コマンドを実行すると、
カウンタのカウンタデータを DRIVE DATA1, 2 PORT (READ) にセットします。

(1) ADDRESS COUNTER READ

アドレスカウンタのカウンタデータを読み出します。
このコマンドの実行は常時可能です。

COMMAND H'D8 ADDRESS COUNTER READ COMMAND

(2) PULSE COUNTER READ

パルスカウンタのカウンタデータを読み出します。
このコマンドの実行は常時可能です。

COMMAND H'D9 PULSE COUNTER READ COMMAND

(3) DFL COUNTER READ

パルス偏差カウンタのカウンタデータを読み出します。
このコマンドの実行は常時可能です。

COMMAND H'DA DFL COUNTER READ COMMAND

6 . 機能説明

6-1. ドライブ仕様

6-1-1. 入出力仕様

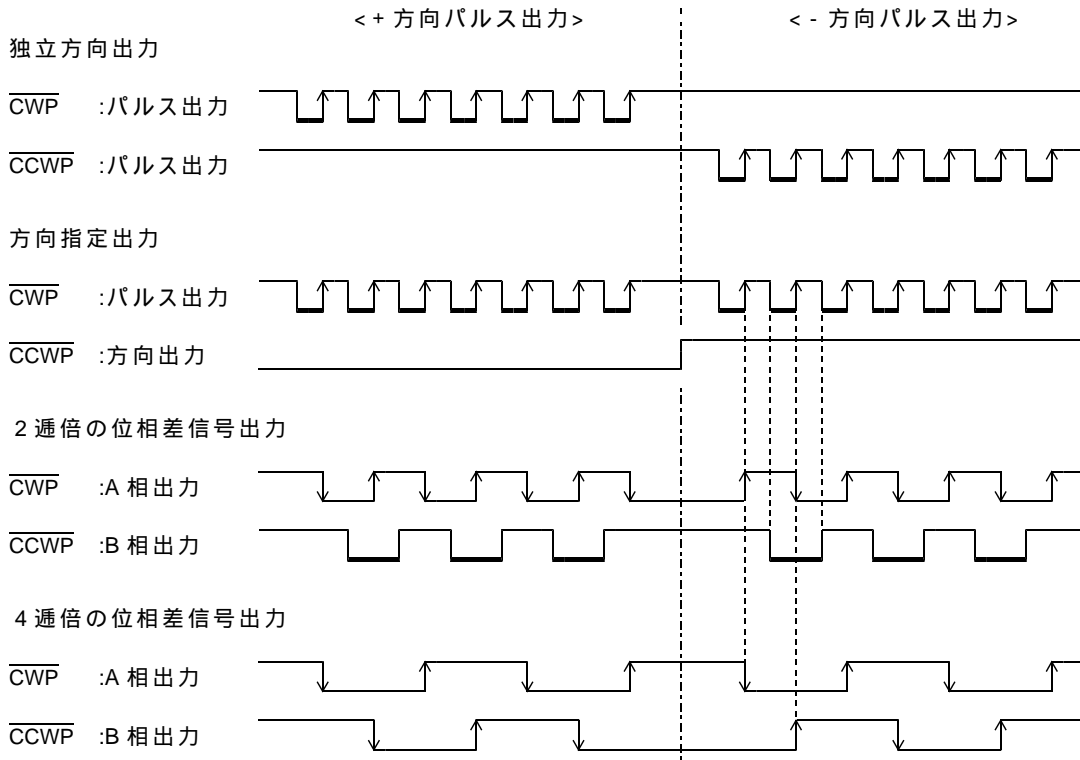
(1) パルス出力仕様

CWP,CCWP 信号から出力するパルスの出力方式を以下の 4 種類の中から選択できます。

(初期値は独立方向出力/各軸で動作します。)

各軸のパルス出力方式は、対象の軸に SPEC INITIALIZE1 コマンドで設定します。

コントローラドライバは、パルス出力仕様を設定することはできません。



CWP, CCWP のアクティブ論理が「ローアクティブ」のときの出力仕様です。

矢印はパルス出力の終了エッジ(アドレスカウンタのカウントエッジ)です。

方向指定出力の方向出力は、出力するパルスの方向が確定すると変化します。

- ・ JOG, SCAN, INDEX, ORIGIN, 直線補間ドライブでは、STBY = 1 で方向が確定します。
- ・ 円弧補間ドライブでは、STBY = 1 で方向確定し、パルス出力直後に次のパルスの方向が確定します。
- ・ 外部パルス出力では、出力する外部パルスの検出で方向が確定します。

位相差信号出力は、独立方向出力のパルス終了エッジのタイミングで変化します。

(2) サーボ対応機能

各軸には、サーボドライバに対応する信号として以下の信号があります。

信号名	機能	読み書きするポート	関数(コマンド)
$\overline{\text{DRST}}$ 信号出力	サーボリセット出力	DRIVE COMMAND PORT	DRIVE COMMAND 書き込み関数 (DRST OUT または SIGNAL OUT)
$\overline{\text{DEND}}$ 信号入力	サーボ位置決め完了入力	DRIVE STATUS2 PORT	DRIVE STATUS2 読み出し関数
DALM 信号入力	ドライバアラーム入力	DRIVE STATUS2 PORT	DRIVE STATUS2 読み出し関数
$\overline{\text{S.ON}}$ 信号出力	サーボ ON	制御 I/O 出力 PORT	I/O PORT 書き込み関数
$\overline{\text{A.CLR}}$ 信号出力	サーボアラームクリア	制御 I/O 出力 PORT	I/O PORT 書き込み関数
$\overline{\text{S.RDY}}$ 信号入力	サーボレディー	制御 I/O 入力 PORT	I/O PORT 読み出し関数

下記のサーボ対応機能は、対象の軸に SPEC INITIALIZE3 コマンドで設定します。

$\overline{\text{DRST}}/\overline{\text{MF}}$ 信号

サーボ対応無効時は、汎用出力としてステッピングモータドライバの M.F 信号(モータ励磁電流の ON/OFF)などに使用できます。

サーボ対応有効時は、ドライブ中に即時停止指令、または LIMIT 即時停止指令を検出すると、 $\overline{\text{DRST}}$ 信号が 10 ms 間アクティブレベルを出力します。

また、ORIGIN SPEC SET 関数の AUTO DRST ENABLE=1 の時は、ORIGIN ドライブ終了時に 10ms 間アクティブレベルを出力します。初期設定はサーボ対応無効です。

- ・ $\overline{\text{DRST}}$ 信号がサーボ対応でアクティブレベルを出力中は DRIVE STATUS1 PORT の BUSY=1 となります。
DRST 信号および DEND 信号の<サーボ対応>終了後に、ドライブを終了します。
- ・ SIGNAL OUT コマンドで ON/OFF レベルを出力することができます。

$\overline{\text{DEND}}/\overline{\text{PO}}$ 信号

サーボ対応無効時は、ステッピングモータドライバの PO 信号入力、または汎用入力として使用できます。

サーボ対応有効時は、ドライブ実行時にパルス出力が終了しても、 $\overline{\text{DEND}}/\overline{\text{PO}}$ 信号のアクティブレベルを検出するまでドライブを終了しません。初期設定はサーボ対応無効です。

DEND 信号の状態は、MCC07 の DRIVE STATUS2 PORT から確認することができます。

- ・ $\overline{\text{DEND}}/\overline{\text{PO}}$ 信号がサーボ対応でアクティブレベルの検出待ちの間は、DRIVE STATUS1 PORT の BUSY = 1、DRIVE STATUS2 PORT の DEND BUSY = 1 になります。
- ・ 即時停止指令を検出した場合は、サーボ対応を中止してドライブを終了します。
即時停止指令の検出で、BUSY = 0、DEND BUSY = 0 になります。

DALM 信号

MCC07 の入力機能を選択すると、ドライバからのアラーム信号入力によって、減速停止、または即時停止させることができます。初期設定は汎用入力です。

DALM 信号の状態は、MCC07 の DRIVE STATUS2 PORT から確認することができます。

また、DALM 信号によって、ERROR とすることができます。

- ・ ERROR の設定は ERROR STATUS MASK コマンドで設定します。

$\overline{\text{S.ON}}$, $\overline{\text{A.CLR}}$, $\overline{\text{S.RDY}}$ 信号

制御 I/O PORT からサーボオン信号、アラームクリア信号の操作、サーボレディー信号の確認ができます。

また、UCD-7613v1/GDB5F40 では、サーボオン(S.ON)信号は各軸 C.S 信号になります。

- ・ 制御 I/O PORT は、ユニットアクセス関数または I/O 関数で操作できます。

6-1-2. ドライブパラメータ

(1) 第 1 パルス出力周期

ドライブ開始時の第 1 パルス目は FSPD で設定したパルス周期を必ず出力します。

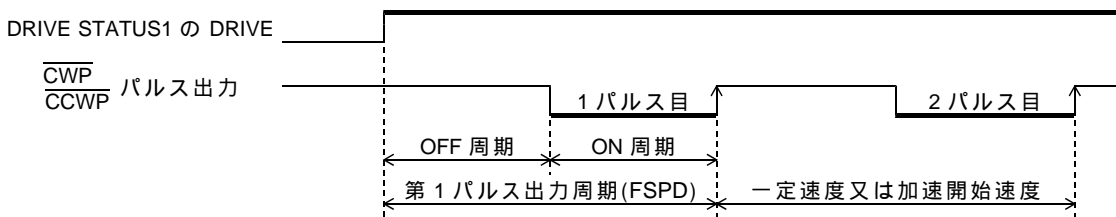
FSPD を調整することにより、パルス出力の指令を与えてからパルス出力開始までの時間を速くすることができます。

- ・初期値は 5kHz (1 周期 200 μ s) です。
- ・駆動するドライブ側の入力応答周波数の範囲内で調整してください。

コマンド予約機能と第 1 パルス出力周期を組み合わせることで、連続したドライブを作ることができます。

FSPD は SPEED・RATE 関数で設定します。

FSPD …… ドライブ開始時の第 1 パルスの出力周期を 1Hz 単位で設定します。
(0 ~ 8,388,607Hz)



- ・ FSPD の設定値と実際に出力する周期は以下の通りとなります。

設定値	実際に出力する周期 (周波数)		
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)

FSPD による DELAY TIME の挿入

FSPD の第 1 パルスは、各ドライブの起動時に必ず出力します。

コマンド予約機能(応用機能)で連続ドライブを行う場合には、次のドライブの FSPD の周期を調整することにより、FSPD を連続ドライブ時の DELAY TIME として利用できます。

FSPD で停止しない連続ドライブを行う

現在のドライブ 次の連続ドライブ間を、開始速度のパルス周期でつなげます。

- ・最初のドライブ実行中に、予約コマンドで「次の連続ドライブ」を設定します。
- ・「次の連続ドライブ」の FSPD を、「次の連続ドライブ」の開始速度に設定します。
- ・MCC07 は、現在のドライブ終了後に予約コマンドの処理を行います。
- ・「次の連続ドライブ」の 1 パルス目 (FSPD) に「次の連続ドライブ」の開始速度を 1 周期出力します。
- ・2 パルス目以降は、「次の連続ドライブ」の開始速度からパルス出力します。

FSPD で反転ドライブの停止時間を挿入する

現在のドライブ 次の反転ドライブ間に、DELAY TIME (例: 50 ms (20 Hz)) を挿入します。

- ・最初のドライブ実行中に、予約コマンドで「次の反転ドライブ」を設定します。
- ・「次の反転ドライブ」の FSPD を、20 Hz に設定します。
- ・MCC07 は、現在のドライブ終了後に予約コマンドの処理を行います。
- ・「次の反転ドライブ」の 1 パルス目 (FSPD) に 20 Hz を 1 周期出力します。
- ・2 パルス目以降は、「次の反転ドライブ」の開始速度からパルス出力します。

DELAY TIME の挿入としては、

SPEC INITIALIZE1 コマンドの PULSE OUTPUT MASK の機能を使用して、

「パルス出力をマスクしたドライブの実行時間」を DELAY TIME として利用することもできます。

(2) 加減速パラメータ

RESOL No.により、RESOL (速度倍率)を設定します。
RESOL No.は、SPEED・RATE セット関数で設定します。

RESOL No.と RESOL (速度倍率) の対応

RESOL No.	RESOL [Hz]	RESOL No.	RESOL [Hz]
0	0.1	6	10
1	0.2	7	20
2	0.5	8	50
3	1	9	100
4	2	10	200
5	5	11	-

SPEED パラメータ設定機能

SPEED・RATE セット関数で加減速ドライブに必要な第1パルス目の速度および速度パラメータを1Hz単位で設定します。
第1パルスの速度(FSPD)は1Hz単位のまま、MCC07に設定されます
加減速ドライブに必要な速度パラメータは、速度データに変換後、MCC07に設定されます

- ・ FSPD … ドライブ開始時の第1パルス目の速度を1Hz単位で設定します。
- ・ HSPD … 加減速ドライブの最高速時のPULSE速度(最高速度)を1Hz単位で設定します。
- ・ LSPD … 加減速ドライブの加速開始時のPULSE速度(開始速度)を1Hz単位で設定します。
- ・ ELSPD … 加減速ドライブの減速終了時のPULSE速度(終了速度)を1Hz単位で設定します。
- ・ SUAREA … 加速カーブのS字変速領域を1Hz単位で設定します。
設定が0のとき、直線加速カーブになります。
- ・ SDAREA … 減速カーブのS字変速領域を1Hz単位で設定します。
設定が0のとき、直線減速カーブになります。
- ・ SUH … 機能はありません。0に設定してください。
- ・ SDH … 機能はありません。0に設定してください。
- ・ URATE … 加速時定数を設定します。(RATEテーブル表のRATE No.で指定してください。)
- ・ DRATE … 減速時定数を設定します。(RATEテーブル表のRATE No.で指定してください。)

- ・設定したSPEEDの値が設定範囲を越えていた場合は、最大値に補正します。
- ・設定したRESOL No.、URATE No.、DRATE No.が設定範囲を越えていた場合、関数エラーとなります。

最高速度、開始速度、終了速度、SUAREA、SDAREAの設定範囲

RES0 No.	RESOL (速度倍率)	HSPD,LSPD,ELSPD 設定範囲	SUAREA,SDAREA 設定範囲
0	0.1	1 ~ 3,276Hz	1 ~ 1,638Hz
1	0.2	1 ~ 6,553Hz	1 ~ 3,276Hz
2	0.5	1 ~ 16,383Hz	1 ~ 8,191Hz
3	1	1 ~ 32,767Hz	1 ~ 16,383Hz
4	2	2 ~ 65,534Hz	2 ~ 32,767Hz
5	5	5 ~ 163,835Hz	5 ~ 81,950Hz
6	10	10 ~ 327,670Hz	10 ~ 163,830Hz
7	20	20 ~ 655,340Hz	20 ~ 327,660Hz
8	50	50 ~ 1,638,350Hz	50 ~ 819,150Hz
9	100	100 ~ 3,276,700Hz	100 ~ 1,638,300Hz
10	200	200 ~ 6,553,400Hz	200 ~ 3,276,600Hz
-	-		

取扱説明書

(USB シリーズ MCC07 ユニット編)

RATEテーブル表

RATE No.	RATE (ms/kHz)	RESOL No.0	RESOL No.1	RESOL No.2	RESOL No.3	RESOL No.4	RESOL No.5
		RESOL=0.1 U/D CYCLE	RESOL=0.2 U/D CYCLE	RESOL=0.5 U/D CYCLE	RESOL=1 U/D CYCLE	RESOL=2 U/D CYCLE	RESOL=5 U/D CYCLE
0	5,000	1,000	2,000	5,000	10,000		
1	3,000	600	1,200	3,000	6,000	12,000	
2	2,000	400	800	2,000	4,000	8,000	
3	1,000	200	400	1,000	2,000	4,000	10,000
4	500	100	200	500	1,000	2,000	5,000
5	300	60	120	300	600	1,200	3,000
6	200	40	80	200	400	800	2,000
7	100	20	40	100	200	400	1,000
8	50	10	20	50	100	200	500
9	30	6	12	30	60	120	300
10	20	4	8	20	40	80	200
11	10	2	4	10	20	40	100
12	5	1	2	5	10	20	50
13	3		1	3	6	12	30
14	2			2	4	8	20
15	1			1	2	4	10
16	0.5				1	2	5
17	0.3					1	3
18	0.2						2
19	0.1						1
20	0.05						
21	0.03						
22	0.02						
23	0.01						
24	0.005						
25	0.0025						

数値は U/D CYCLE (単位 0.5 μ s)です。

RATE テーブル表(続き)

RATE No.	RATE (ms/kHz)	RESOL No.6	RESOL No.7	RESOL No.8	RESOL No.9	RESOL No.10
		RESOL=10	RESOL=20	RESOL=50	RESOL=100	RESOL=200
		U/D CYCLE	U/D CYCLE	U/D CYCLE	U/D CYCLE	U/D CYCLE
0	5,000					
1	3,000					
2	2,000					
3	1,000					
4	500	10,000				
5	300	6,000	12,000			
6	200	4,000	8,000			
7	100	2,000	4,000	10,000		
8	50	1,000	2,000	5,000	10,000	
9	30	600	1,200	3,000	6,000	12,000
10	20	400	800	2,000	4,000	8,000
11	10	200	400	1,000	2,000	4,000
12	5	100	200	500	1,000	2,000
13	3	60	120	300	600	1,200
14	2	40	80	200	400	800
15	1	20	40	100	200	400
16	0.5	10	20	50	100	200
17	0.3	6	12	30	60	120
18	0.2	4	8	20	40	80
19	0.1	2	4	10	20	40
20	0.05	1	2	5	10	20
21	0.03		1	3	6	12
22	0.02			2	4	8
23	0.01			1	2	4
24	0.005				1	2
25	0.0025					1

数値は U/D CYCLE (単位 0.5 μ s) です。

直線加減速ドライブの動作

直線加減速ドライブは、S字加速の変速領域を "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

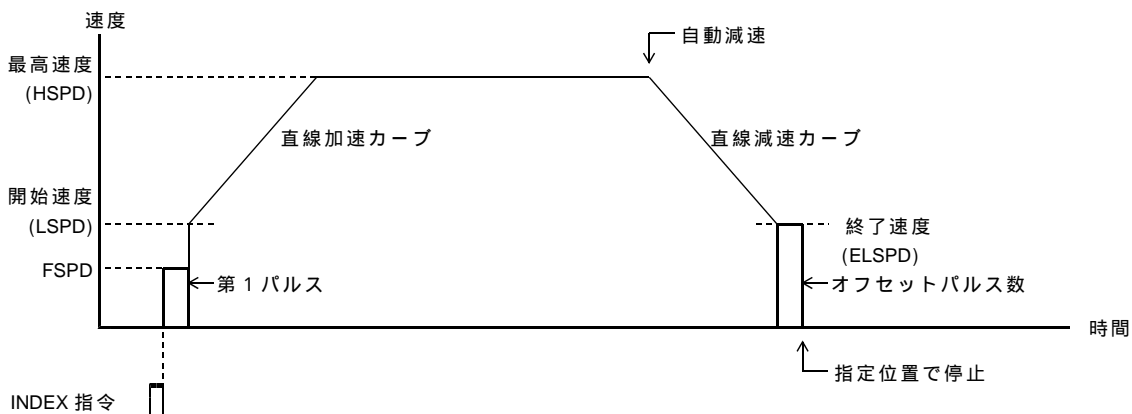
SCAREA SET コマンドの SUAREA を "0" に設定します。

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。

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。



S 字加減速ドライブの動作

S 字加減速ドライブは、S 字加速の変速領域を設定して、加減速を行うドライブです。

- ・加速開始時の 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。
- ・減速開始時の 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

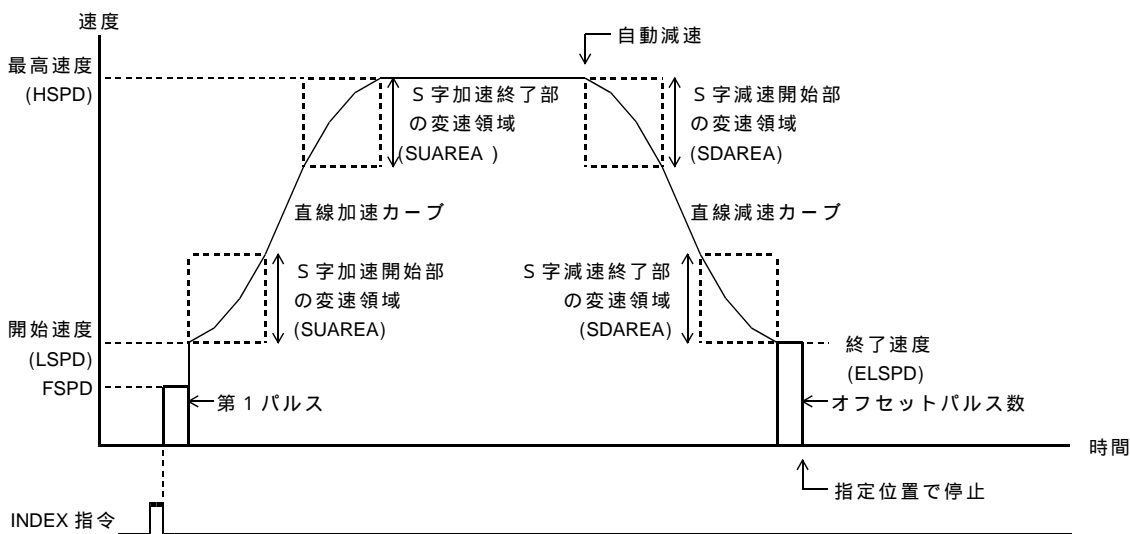
SCAREA SET コマンドの SUAREA で S 字加速開始部の変速領域を設定します。

SUAREA 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

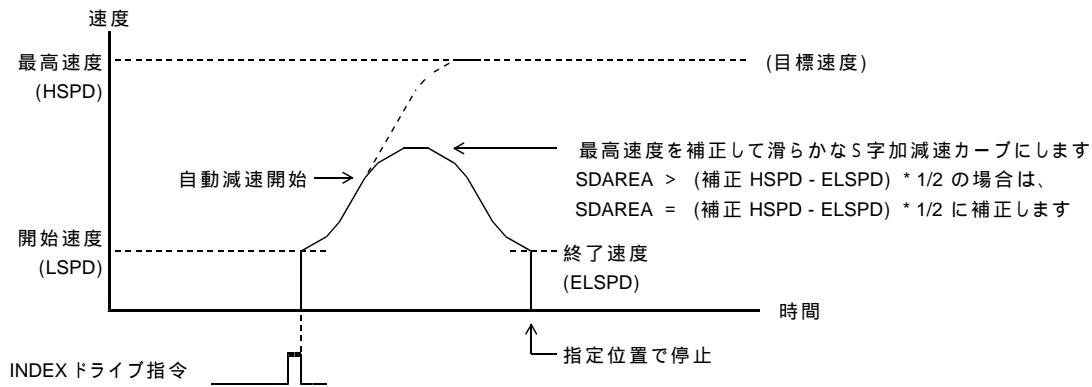
SDAREA SET コマンドの SDAREA で S 字減速終了部の変速領域を設定します。

SDAREA 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。



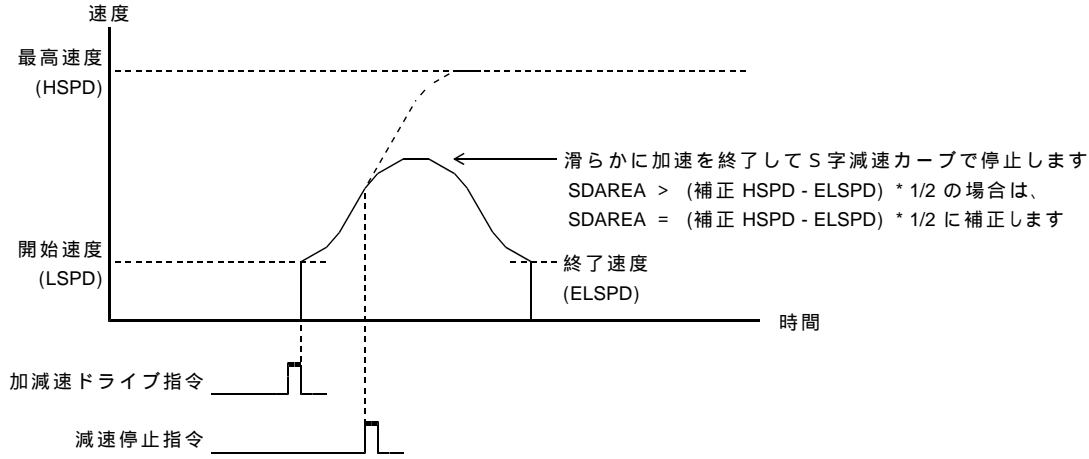
S 字加減速 INDEX ドライブの三角駆動回避動作

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。



減速停止指令検出時の三角駆動回避動作

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。この機能は常時有効です。



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。

(3) JOG パラメータ

JOG パルス速度

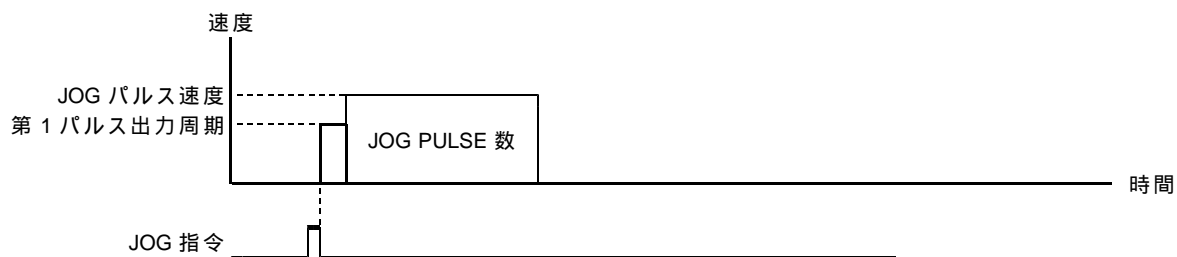
JOG ドライブを実行すると設定した JOG パルス速度の一定速でドライブを行います。
JOG パルス速度の設定範囲は 1 ~ 4,194,303Hz(1Hz 単位)です。

- ・ JOG パルス速度は JSPD SET コマンドで設定します。

JOG パルス数

JOG ドライブを実行すると設定した JOG パルス数のパルスを出力します。
JOG パルス数設定範囲は、0 ~ 65,535 (H'0000 ~ H'FFFF) です。

- ・ JOG ドライブパルス数は JOG PULSE SET コマンドで設定します。
- ・ JOG パルス速度は ORIGIN ドライブ工程の CONSTANT SCAN ドライブ工程の
パルス速度にも適用します。



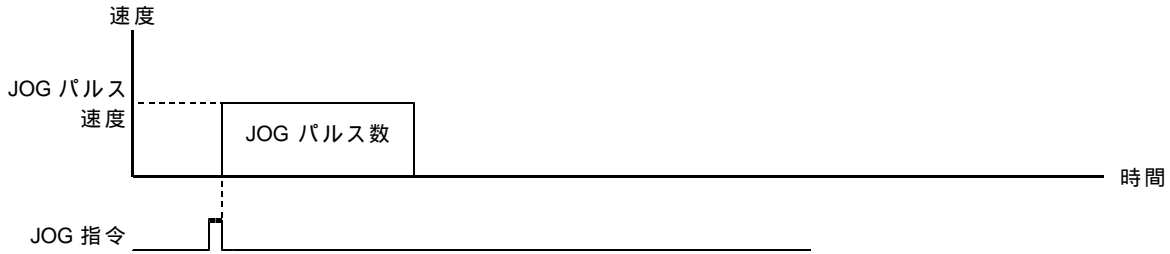
6-1-3. 基本ドライブ

(1) JOG ドライブ

+/- JOG コマンドを実行すると、JOG パルス速度の一定速で JOG パルス数のパルスを出力します。

減速停止指令を検出すると、パルス出力を即時停止してドライブを終了します。

即時停止指令を検出すると、パルス出力を即時停止してドライブを終了します。

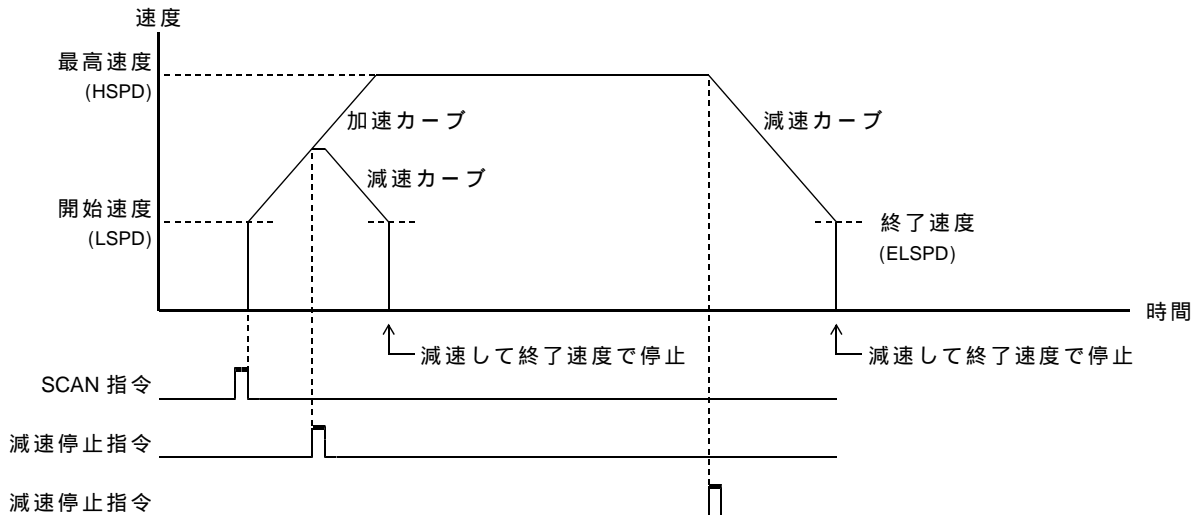
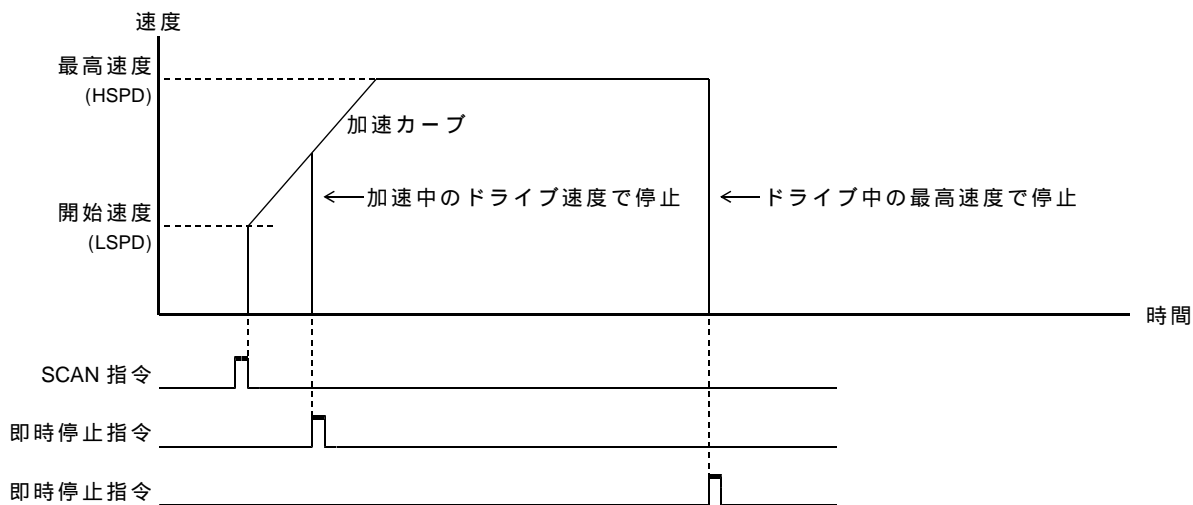


(2) SCAN ドライブ

+/- SCAN コマンドを実行すると、停止指令を検出するまで連続してパルスを出します。

減速停止指令を検出すると、パルス出力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すると、パルス出力を即時停止してドライブを終了します。

減速停止指令による停止動作**即時停止指令による停止動作**

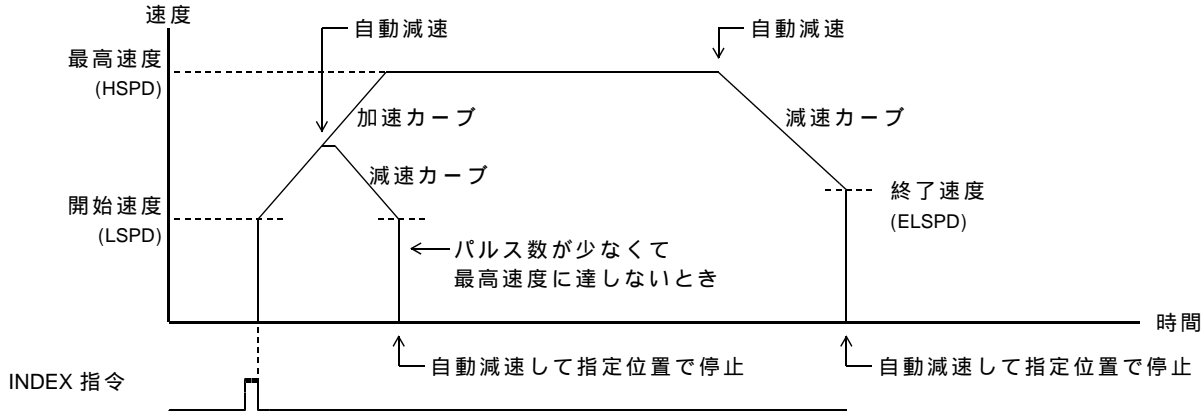
(3) INDEX ドライブ

INC INDEX コマンドを実行すると、指定した相対アドレスに達するまでパルスを出力します。

ABS INDEX コマンドを実行すると、指定した絶対アドレスに達するまでパルスを出力します。

加減速ドライブ中には、パルス速度を自動減速して指定位置で停止します。

- ・減速停止指令を検出すると、パルス出力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すると、パルス出力を即時停止してドライブを終了します。

自動減速機能による停止動作

- ・現在速度が終了速度以下の場合、減速停止指令を検出すると終了速度に向かって加速します。
- 自動減速地点を検出すると終了速度に向かって加速し、指定位置でパルス出力を停止します。

6-1-4. ORIGIN ドライブ

(1) ORIGIN ドライブ仕様

ORIGIN ドライブは、センサを検出する各種ドライブ工程を順次行い、機械原点信号を検出してドライブを終了します。ORIGIN ドライブは、MCC07 が持っている ORIGIN ドライブを組み合わせて、コントローラが ORIGIN ドライブを実現する機能です。

ORIGIN ドライブには、ORG-0 ~ 5, 10,11,12 の 9 種類のドライブ型式があります。

ドライブ型式の特徴

ドライブ型式	検出するセンサ数	検出完了時のセンサの状態	ドライブ工程数	所要時間	精度	CWLM信号の入力機能	CCWLM信号の入力機能
ORG-0	1	OFF	2	短	低	+ 方向のLIMIT	- 方向のLIMIT
ORG-1	1	ON	2	短	低	+ 方向のLIMIT	- 方向のLIMIT
ORG-2	1	OFF	4	長	中	+ 方向のLIMIT	- 方向のLIMIT
ORG-3	1	ON	4	長	中	+ 方向のLIMIT	- 方向のLIMIT
ORG-4	2	OFF	4/5	最長	高	+ 方向のLIMIT	- 方向のLIMIT
ORG-5	2	ON	4/5	最長	高	+ 方向のLIMIT	- 方向のLIMIT
ORG-10	2	ON	2	最短	低	+ 方向のLIMIT	- 方向のLIMIT
ORG-11	1	OFF	2	短	低	+ 方向のLIMIT 検出信号	- 方向のLIMIT
ORG-12	1	OFF	4	長	中	+ 方向のLIMIT 検出信号	- 方向のLIMIT

ORG-0 ~ 5,10 で検出するセンサ信号は、 $\overline{\text{ORG}}$, $\overline{\text{NORG}}$, $\pm \text{ZORG}$ 信号入力を合成した ORG, NORG 検出信号です。

- ・ NORG 検出信号は、ORIGIN SPEC SET 関数の NORG SIGNAL TYPE で選択します。
- ・ ORG 検出信号は、ORIGIN SPEC SET 関数の ORG SIGNAL TYPE で選択します。

機械原点近傍アドレスまでの INDEX ドライブ

ORG-0 ~ 5,11,12 の各工程では 1 度検出された機械原点アドレスを記憶し、以後の機械原点検出を短時間で実行機能が付加されています。

このため内部に検出 FLAG を用意しており、この FLAG が ON の場合は、機械原点近傍 (原点+OFFSET PULSE) まで INDEX ドライブで移動し、その後に原点検出の工程を行います。FLAG が OFF の場合は INDEX ドライブを行わず各原点検出工程の DRIVE から行います。

- 1 度機械原点検出した後に、機械側だけを手動で動かした場合は、実際の機械の位置とコントローラ側で管理しているアドレスが異なる可能性があります。
- このような状態で 2 回目の ORIGIN ドライブを起動すると、正しく機械原点が検出できない場合があります。
- 一度 ORIGIN FLAG RESET 関数により ORIGIN FLAG をリセットしてください。
- これにより、電源投入時の最初の 1 回目と同じように、ORIGIN センサを検出する ORIGIN ドライブとなります。

- ・ ORIGIN ドライブで正常に機械原点が検出されたとき、ORIGIN STATUS の ORIGIN FLAG = 1 になります
- ・ ORIGIN FLAG = 1 の状態で ORIGIN ドライブを実行すると、機械原点近傍アドレスまで INDEX ドライブで移動し、その後に原点検出の工程を行います。
- ・ ORIGIN FLAG = 0 の状態で ORIGIN ドライブを実行すると、機械原点近傍アドレスまで INDEX ドライブを行わずに、原点検出の工程を行います。
- ・ 機械原点近傍アドレスは、機械原点アドレスに OFFSET パルス数を加算したアドレスです。

記憶される機械原点アドレスは、ORIGIN ドライブのドライブ型式により異なります。

- ・ ORG-0 ~ 3,11,12 の場合は、機械原点検出位置が機械原点アドレスとして記憶されます
- ・ ORG-4,5 の場合は、NORG 信号検出位置が機械原点アドレスとして記憶されます

ADDRESS COUNTER PRESET コマンドを実行すると、記憶された機械原点アドレスも更新されます。

このため、物理的な位置はユーザは何も考慮する必要はありません。

ORIGIN FLAG ON 条件

ORIGIN ドライブで正常に機械原点が検出されたとき。

ただし、以下の場合は、正常に機械原点が検出されても ORIGIN FLAG = 1 になりません。

- ・アドレスカウンタの最大カウント数を FFFFFFFF 以外に設定している場合
- ・アドレスカウンタがオーバーフローしている場合
- ・ORIGIN ドライブの動作仕様の AUTO DRST ENABLE = 1 の場合

ORIGIN FLAG OFF 条件

- ・コントローラの電源投入時
- ・ORIGIN FLAG RESET 関数実行時
- ・ORIGIN SPEC SET 関数実行時
- ・ORIGIN MARGIN PULSE SET 関数実行時
- ・ORIGIN DELAY SET 関数実行時
- ・ORIGIN ERROR PULSE SET 関数実行時
- ・ORIGIN OFFSET PULSE SET 関数実行時
- ・ORIGIN PRESET PULSE SET 関数実行時
- ・即時停止指令によるドライブ終了時
 - ORIGIN ドライブ中は、ORIGIN STATUS の FSEND = 1
 - ORIGIN ドライブ中以外は、DRIVE STATUS1 PORT の FSEND = 1
- ・LIMIT 減速停止指令または LIMIT 即時停止指令によるドライブ終了時
 - ORIGIN ドライブ中は、ORIGIN STATUS の LSEND = 1
 - ORIGIN ドライブ中以外は、STATUS1 PORT の LSEND = 1
- ・CPP STOP 機能によるドライブ終了時
- ・アドレスカウンタがオーバーフローした時
- ・ORIGIN STATUS の ORIGIN ERROR = 1 で ORIGIN ドライブを終了
- ・前回の ORIGIN ドライブと異なる型式の ORIGIN ドライブを起動した時
- ・ADDRESS COUNTER MAX COUNT SET コマンドを実行した時
- ・機械原点アドレスが ± 2,147,483,647 の範囲を超えた時
- ・DRST OUT コマンドを実行したとき

ORIGIN ドライブの各種ドライブ工程

ORIGIN ドライブには、SCAN 工程、CONSTANT SCAN 工程、1PULSE 送り工程の3つの工程があります。

SCAN 工程

加減速ドライブのパラメータで ORIGIN SCAN ドライブを行います。センサ信号を検出すると減速停止します。

CONSTANT SCAN 工程

ORIGIN CSCAN ドライブ(一定速ドライブ)を行います。センサ信号を検出すると停止します。

1PULSE 送り工程

PULSE DELAY TIME の間隔で 1 パルスずつ出力します。センサ信号を検出すると停止します。

- ・ORIGIN SPEC SET 関数の PULSE SENSOR TYPE = 0、または AUTO DRST ENABLE = 1 の場合、PULSE DELAY TIME を SPEED 換算して、ORIGIN CSCAN ドライブ(一定速ドライブ)を行います。

例) PULSE DELAY TIME=20ms のとき、50Hz でドライブします。

AUTO DRST ENABLE=1 のとき、PULSE SENSOR TYPE の設定にかかわらず、機械原点検出のエッジを検出して工程を終了します。

- ・ORIGIN SPEC SET 関数の PULSE SENSOR TYPE = 1、且つ AUTO DRST ENABLE = 0 の場合、PULSE DELAY TIME の時間間隔で 1PULSE ドライブを繰り返し行います。

PULSE DELAY TIME は、ORIGIN DELAY SET 関数で設定します。

ORIGIN ドライブの LIMIT 信号について

ORIGIN ドライブでは、CWLM, CCWLM 信号を LIMIT 信号として使用します。
CWLM, CCWLM 信号にはシステムの LIMIT センサ信号を入力してください。

ORIGIN ドライブ(SCAN 工程、CONSTANT SCAN 工程、1PULSE 送り工程)では、
CWLM 信号を + 方向、CCWLM 信号を - 方向の LIMIT 停止信号として検出します。

ORG-11,ORG-12 では、CWLM,CCWLM 信号の一方が機械原点信号になります。
・ ORIGIN ドライブの起動方向が CCW 方向の場合は、CCWLM 信号が機械原点になり、
CWLM 信号は LIMIT 停止信号になります。
・ ORIGIN ドライブの起動方向が CW 方向の場合は、CWLM 信号が機械原点になり、
CCWLM 信号は LIMIT 停止信号になります。

機械原点近傍アドレスまでの INDEX ドライブ、および PRESET ドライブの実行中、CWLM, CCWLM 信号は
以下のように機能します。

- ・ SPEC INITIALIZE2 コマンドで設定されている「CWLM TYPE」と、「CCWLM TYPE」で機能します。
- ・ 入力機能が LIMIT 停止機能の場合は、LIMIT 停止後に ORIGIN ドライブを終了します。

プリセットドライブ

機械原点検出ドライブが正常終了後、PRESET PULSE 数で設定された位置までドライブを行います。
PRESET PULSE 数は、ORIGIN PRESET PULSE SET 関数で設定します。

ERROR PULSE ERROR 検出機能

CONSTANT SCAN 工程、1PULSE 送り工程実行中に検出信号を検出できずに出力したパルス数がエラー判定
する最大パルス数に達した場合、ORIGIN STATUS の ERROR PULSE ERROR = 1 で ORIGIN ドライブが終了します。
ORIGIN SPEC SET 関数の ERROR PULSE ERROR ENABLE = 1 の場合に有効です。
エラー判定 PULSE 数は、ORIGIN ERROR PULSE SET 関数で設定します。

MARGIN パルス機能

SCAN 工程および CSCAN 工程のときに MARGIN パルスを挿入します。
SCAN 工程では、ORIGIN SPEC SET 関数の SCAN MARGIN ENABLE=1 のときに有効です。

NORG 検出工程および ORIGIN ドライブの最終工程では、MARGIN PULSE を挿入しません。
・ CONSTANT SCAN 工程では、機械原点信号を検出すると進行方向へ MARGIN パルス分の進入を
行ってから停止します。
・ SCAN 工程では機械原点信号を検出し、ドライブを減速停止した後、MARGIN パルス数分の進入を行います。
MARGIN パルスは、MARGIN PULSE SET 関数で設定します。

DELAY TIME

- ・ SCAN 工程、および CSCAN 工程の動作反転時に SCAN DELAY TIME を挿入します。
ORIGIN DELAY SET 関数の SCAN DELAY TIME で設定します。
- ・ SCAN 工程に LIMIT 停止し、反転する場合に LIMIT DELAY TIME を挿入します。
ORIGIN DELAY SET 関数の LIMIT DELAY TIME で設定します。
- ・ 1 パルス送り工程時、PULSE DELAY TIME の時間間隔で 1 パルスずつパルス出力します。
ORIGIN DELAY SET 関数の PULSE DELAY SET 関数で設定します。

ORIGIN ドライブの実行シーケンス

直線加減速ドライブ、または S 字加減速ドライブに必要なパラメータ、CONSTANT SCAN 工程のパルス速度、ORIGIN ドライブパラメータを設定し、ORIGIN ドライブを実行します。

加減速ドライブに必要なパラメータは、以下の工程に適用されます。

- ・機械原点近傍アドレスまでの INDEX ドライブ
- ・PRESET ドライブ
- ・SCAN 工程

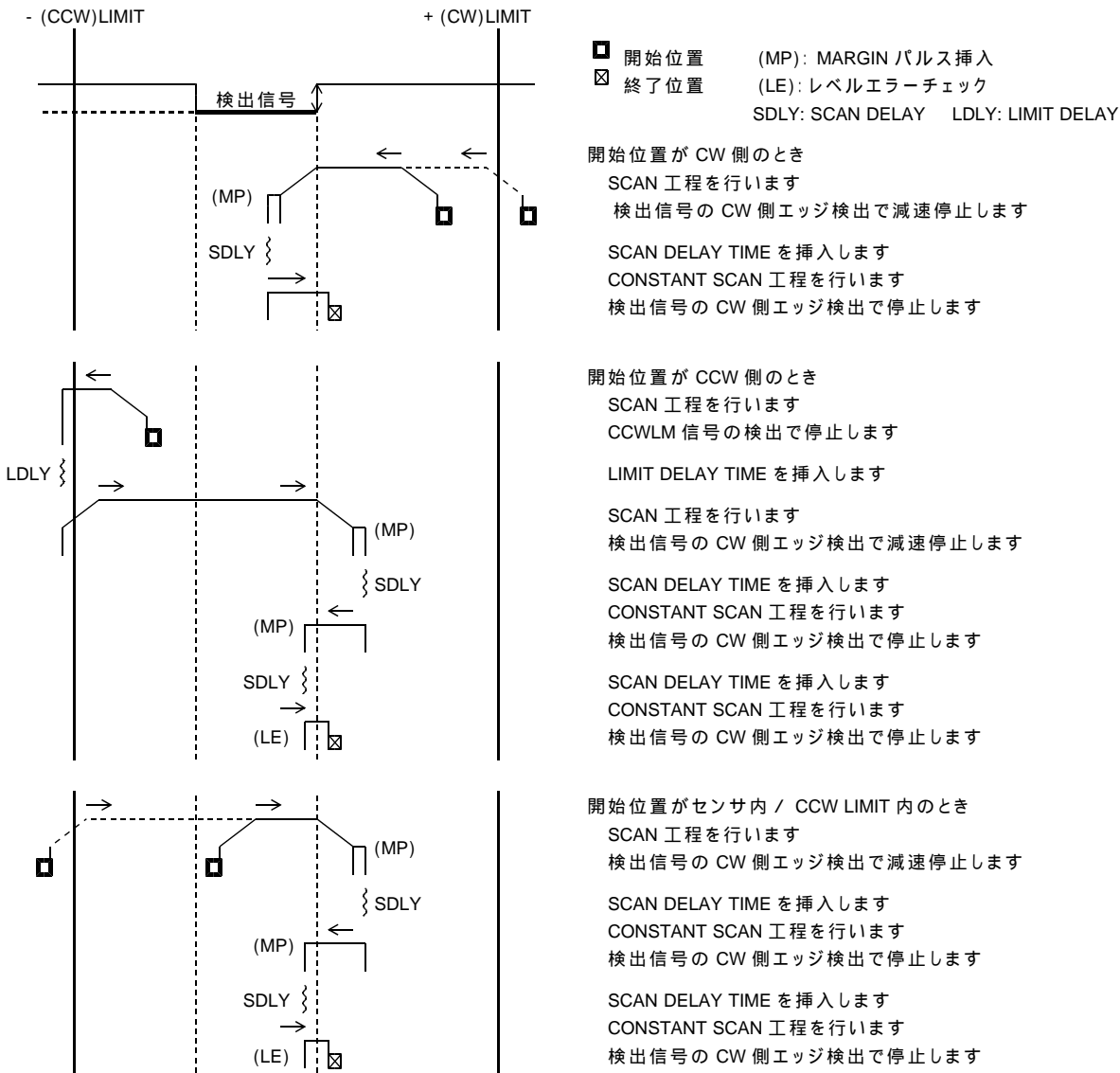
ORIGIN ドライブの実行シーケンス



(2) ORG-0 ドライブ型式

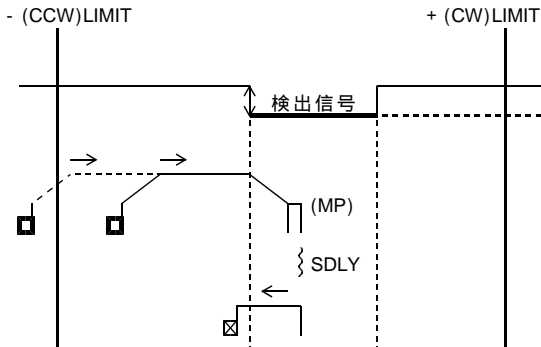
ORIGIN ドライブの起動方向が - (CCW)方向の場合

CCW 方向の ORG-0 型式は、ORG 検出信号の CW 側エッジ検出で機械原点を検出します。
ORG 検出信号には、1つのパルス、または - (CCW)側レベル保持のセンサ信号を入力します。
最高速度でセンサを通過したときに、1ms 以上の幅の信号が検出されるようにします。



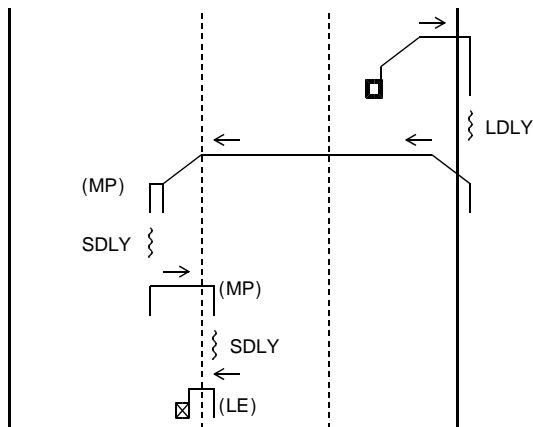
ORIGIN ドライブの起動方向が + (CW)方向の場合

起動方向が CW 方向の場合は、CCW 方向と対称の動作で、対称方向のエッジを検出します。
CW 方向の ORG-0 型式は、ORG 検出信号の CCW 側エッジ検出で機械原点を検出します。
ORG 検出信号には、1つのパルス、または + (CW)側レベル保持のセンサ信号を入力します。
最高速度でセンサを通過したときに、1ms 以上の幅の信号が検出されるようにします。

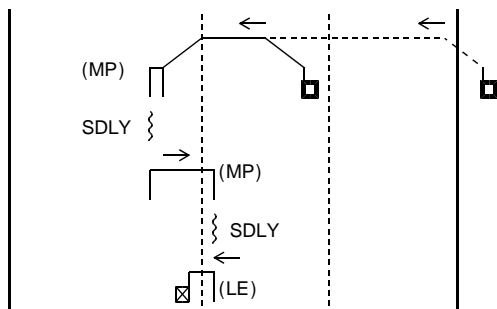


- 開始位置 (MP): MARGIN パルス挿入
- ☒ 終了位置 (LE): レベルエラーチェック
- SDLY: SCAN DELAY LDLY: LIMIT DELAY

開始位置が CCW 側のとき
SCAN 工程を行います
検出信号の CCW 側エッジ検出で減速停止します
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CCW 側エッジ検出で停止します



開始位置が CW 側のとき
SCAN 工程を行います
CWLM 信号の検出で停止します
LIMIT DELAY TIME を挿入します
SCAN 工程を行います
検出信号の CCW 側エッジ検出で減速停止します
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CCW 側エッジ検出で停止します
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CCW 側エッジ検出で停止します

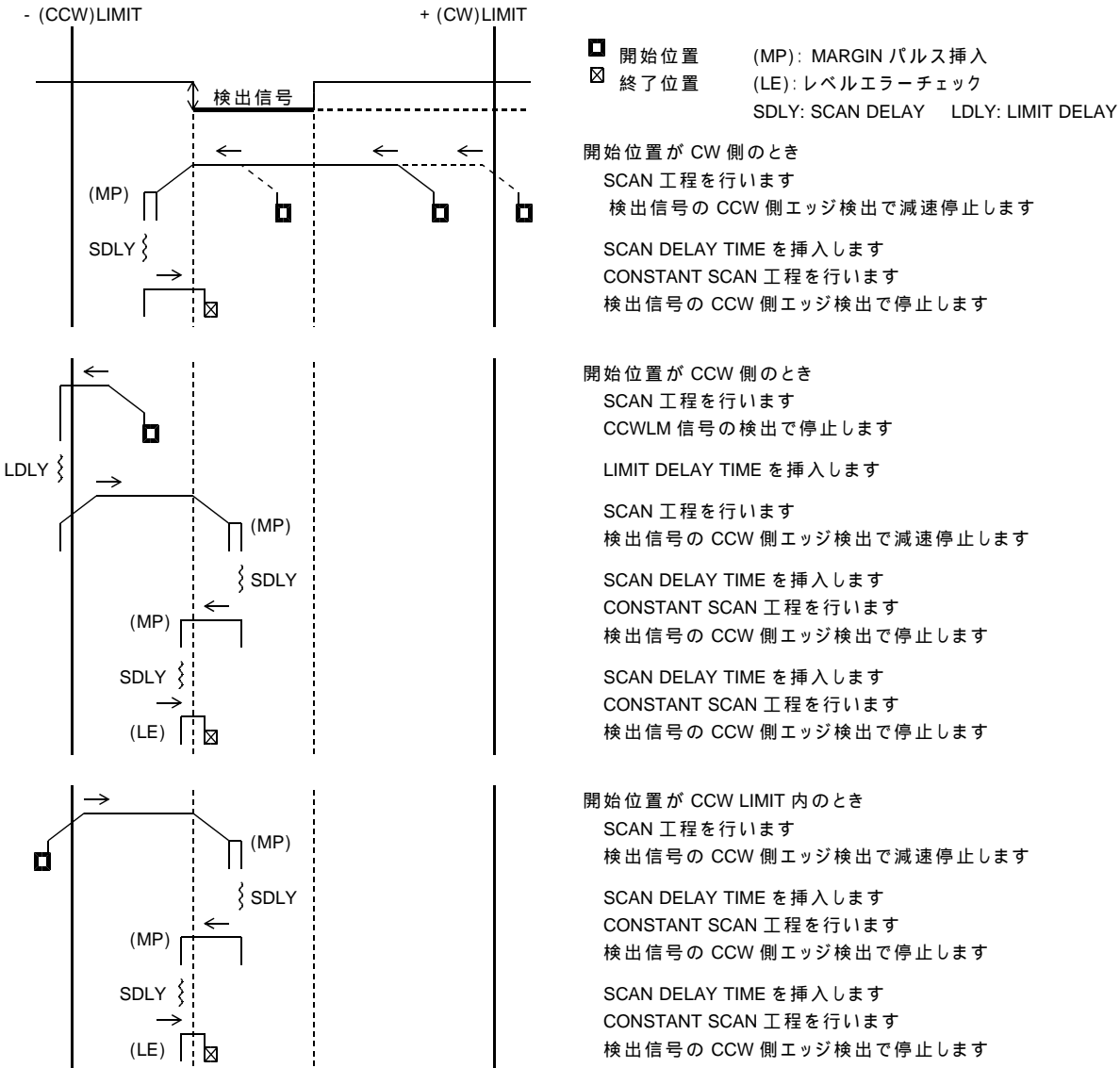


開始位置がセンサ内 / CW LIMIT 内のとき
SCAN 工程を行います
検出信号の CCW 側エッジ検出で減速停止します
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CCW 側エッジ検出で停止します
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CCW 側エッジ検出で停止します

(3) ORG-1 ドライブ型式

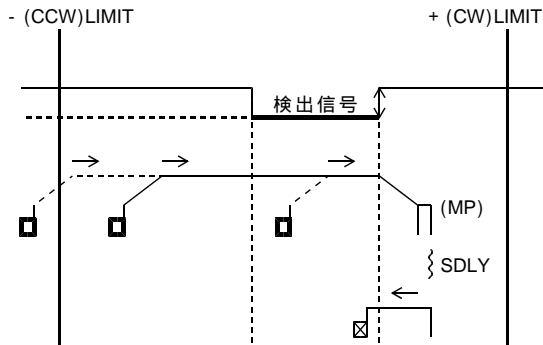
ORIGIN ドライブの起動方向が - (CCW)方向の場合

CCW 方向の ORG-1 型式は、ORG 検出信号の CCW 側エッジ検出で機械原点を検出します。
ORG 検出信号には、1つのパルス、または + (CW)側レベル保持のセンサ信号を入力します。
最高速度でセンサを通過したときに、1ms 以上の幅の信号が検出されるようにします。



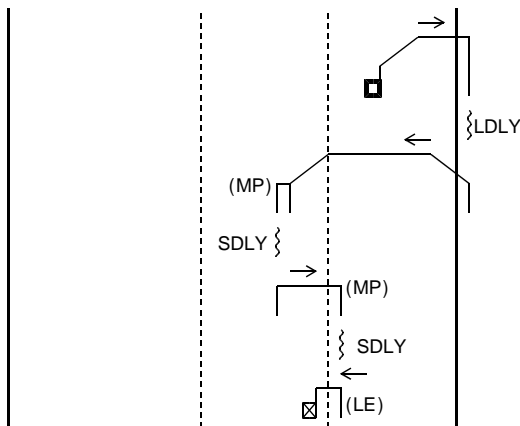
ORIGIN ドライブの起動方向が + (CW)方向の場合

起動方向が CW 方向の場合は、CCW 方向と対称の動作で、対称方向のエッジを検出します。
CW 方向の ORG-1 型式は、ORG 検出信号の CW 側エッジ検出で機械原点を検出します。
ORG 検出信号には、1つのパルス、または - (CCW)側レベル保持のセンサ信号を入力します。
最高速度でセンサを通過したときに、1ms 以上の幅の信号が検出されるようにします。

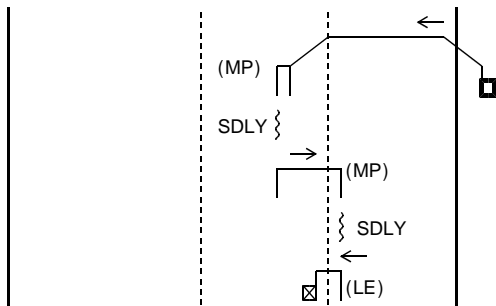


- 開始位置 (MP): MARGIN パルス挿入
- ☒ 終了位置 (LE): レベルエラーチェック
- SDLY: SCAN DELAY LDLY: LIMIT DELAY

開始位置が CCW 側のとき
SCAN 工程を行います
検出信号の CW 側エッジ検出で減速停止します
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します



開始位置が CW 側のとき
SCAN 工程を行います
CWLM 信号の検出で停止します
LIMIT DELAY TIME を挿入します
SCAN 工程を行います
検出信号の CW 側エッジ検出で減速停止します
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します

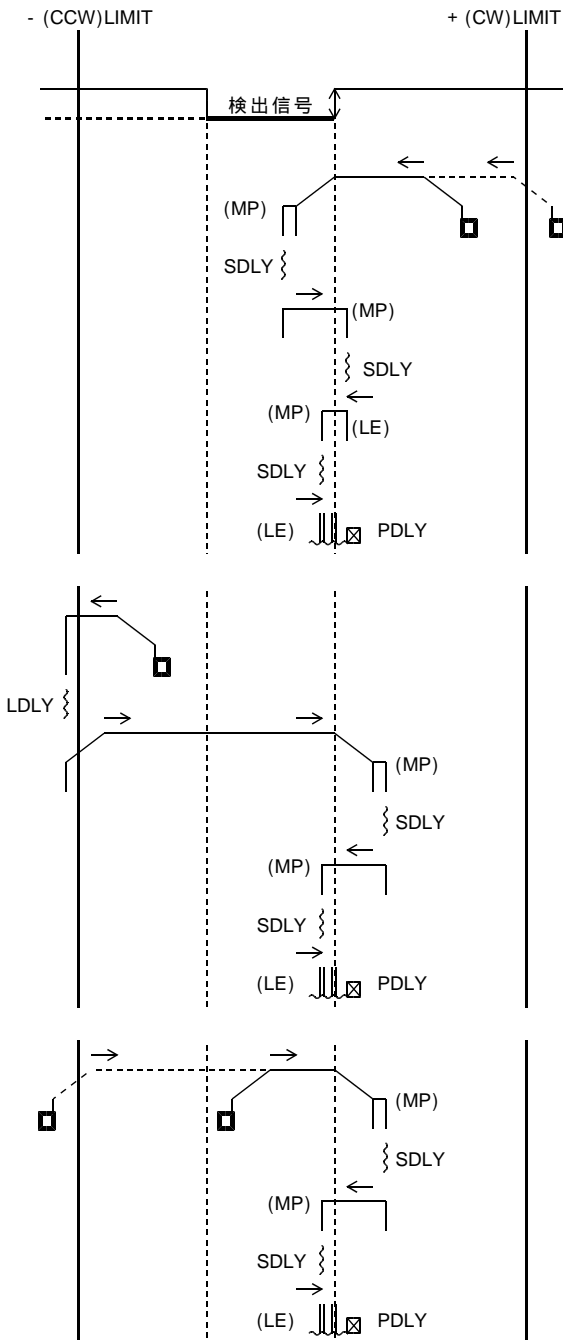


開始位置が CW LIMIT 内のとき
SCAN 工程を行います
検出信号の CW 側エッジ検出で減速停止します
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します

(4) ORG-2 ドライブ型式

ORIGIN ドライブの起動方向を、- (CCW)方向として説明します。

ORG-2 型式は、ORG-0 型式に 1PULSE 送り工程を付加して精度を高めた型式です。



- 開始位置 (MP): MARGIN パルス挿入
- ☒ 終了位置 (LE): レベルエラーチェック

SDLY: SCAN DELAY LDLY: LIMIT DELAY

PDLY: PULSE DELAY

開始位置が CW 側のとき

SCAN 工程を行います
検出信号の CW 側エッジ検出で減速停止します

SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します

SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します

SCAN DELAY TIME を挿入します
1PULSE 送り工程を行います
検出信号の CW 側エッジ検出で停止します

開始位置が CCW 側のとき

SCAN 工程を行います
CCWLM 信号の検出で停止します

LIMIT DELAY TIME を挿入します

SCAN 工程を行います
検出信号の CW 側エッジ検出で減速停止します

SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します

SCAN DELAY TIME を挿入します
1PULSE 送り工程を行います
検出信号の CW 側エッジ検出で停止します

開始位置がセンサ内 / CCW LIMIT 内のとき

SCAN 工程を行います
検出信号の CW 側エッジ検出で減速停止します

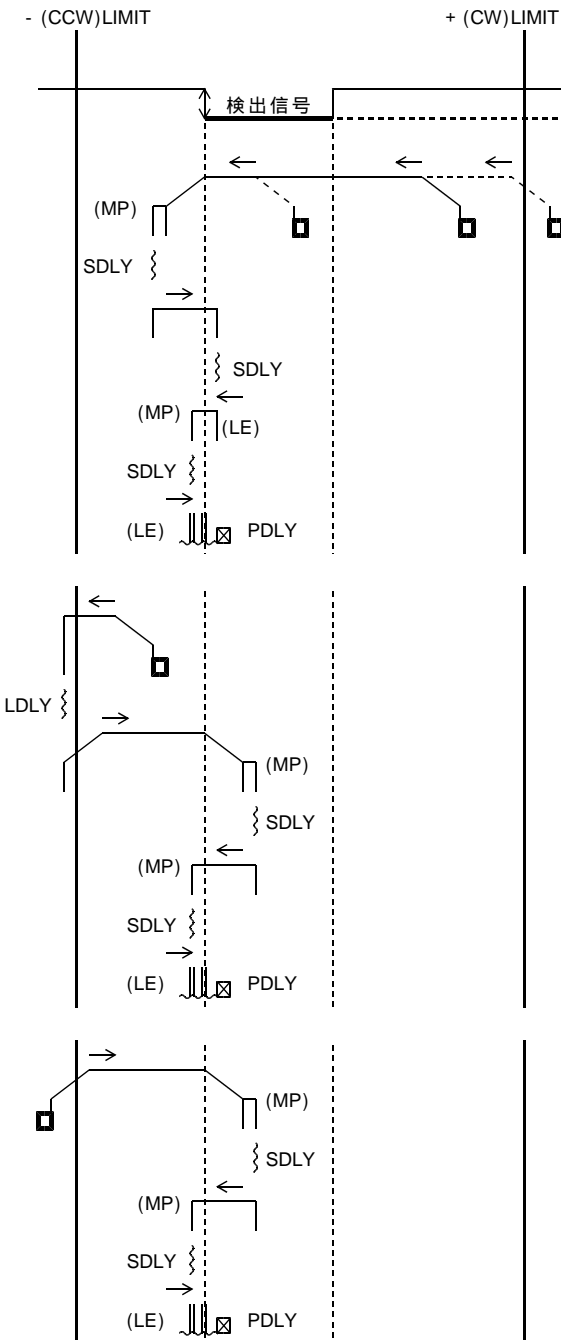
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します

SCAN DELAY TIME を挿入します
1PULSE 送り工程を行います
検出信号の CW 側エッジ検出で停止します

(5) ORG-3 ドライブ型式

ORIGIN ドライブの起動方向を、- (CCW)方向として説明します。

ORG-3 型式は、ORG-1 型式に JOG 工程を付加して精度を高めた型式です。



- 開始位置 (MP): MARGIN パルス挿入
- ☒ 終了位置 (LE): レベルエラーチェック
- SDLY: SCAN DELAY LDLY: LIMIT DELAY
- PDLY: PULSE DELAY

開始位置が CW 側するとき
SCAN 工程を行います
検出信号の CCW 側エッジ検出で停止します

SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CCW 側エッジ検出で停止します

SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CCW 側エッジ検出で停止します

SCAN DELAY TIME を挿入します
1PULSE 送り工程を行います
検出信号の CCW 側エッジ検出で停止します

開始位置が CCW 側するとき
SCAN 工程を行います
CCWLM 信号の検出で停止します

LIMIT DELAY TIME を挿入します

SCAN 工程を行います
検出信号の CCW 側エッジ検出で減速停止します

SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CCW 側エッジ検出で停止します

SCAN DELAY TIME を挿入します
1PULSE 送り工程を行います
検出信号の CCW 側エッジ検出で停止します

開始位置が CCW LIMIT 内するとき
SCAN 工程を行います
検出信号の CCW 側エッジ検出で減速停止します

SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CCW 側エッジ検出で停止します

SCAN DELAY TIME を挿入します
1PULSE 送り工程を行います
検出信号の CCW 側エッジ検出で停止します

(6) ORG-4, ORG-5 ドライブ型式

ORG-4, ORG-5 型式は、NORG 検出信号と ORG 検出信号で機械原点を検出します。

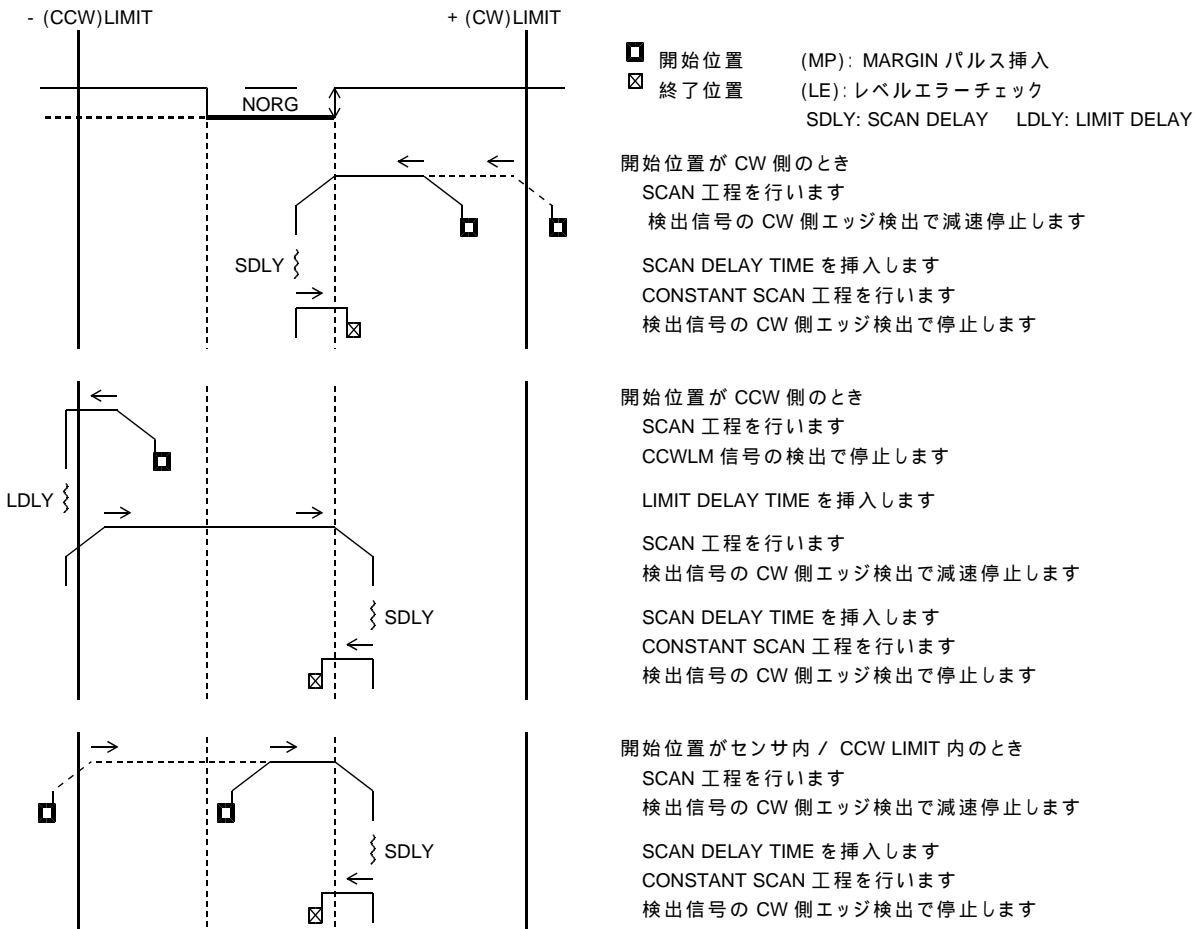
ORG-4, ORG-5 型式は、最初に NEAR ORIGIN 工程を実行します。次に ORIGIN 工程を実行します。

ORG-4、ORG-5 型式の NEAR ORIGIN 工程

ORIGIN ドライブの起動方向を、- (CCW) 方向として説明します。

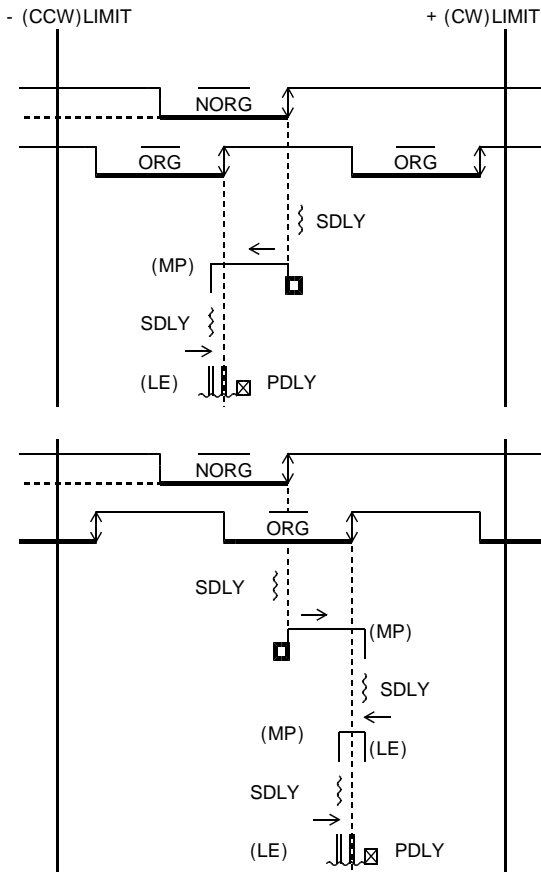
起動方向が CW 方向の場合は、対称の動作で、対称方向のエッジを検出します。

NORG 検出信号には、1つのパルス、または - (CCW) 側レベル保持のセンサ信号を入力します。
最高速度でセンサを通過したときに、1ms 以上の幅の信号が検出されるようにします。



ORG-4 型式の ORIGIN 工程

ORIGIN ドライブの起動方向を、- (CCW)方向として説明します。
 起動方向が CW 方向の場合は、対称の動作で対称方向のエッジを検出します。
 ORG 検出信号には、回転軸のスリットなど周期的に信号を発生するセンサ信号を入力します。
 CONSTANT SCAN 工程の速度 (CSPD) でセンサを通過したときに、1ms 以上の幅の信号が検出されるようにします。



- 開始位置 (MP): MARGIN パルス挿入
- ☒ 終了位置 (LE): レベルエラーチェック
- SDLY: SCAN DELAY LDLY: LIMIT DELAY
- PDLY: PULSE DELAY

NORG 検出時に ORG がハイレベルのとき
 SCAN DELAY TIME を挿入します
 CONSTANT SCAN 工程を行います
 検出信号の CW 側エッジ検出で停止します

 SCAN DELAY TIME を挿入します
 1PULSE 送り工程を行います
 検出信号の CW 側エッジ検出で停止します

NORG 検出時に ORG がローレベルのとき
 SCAN DELAY TIME を挿入します
 CONSTANT SCAN 工程を行います
 検出信号の CW 側エッジ検出で停止します

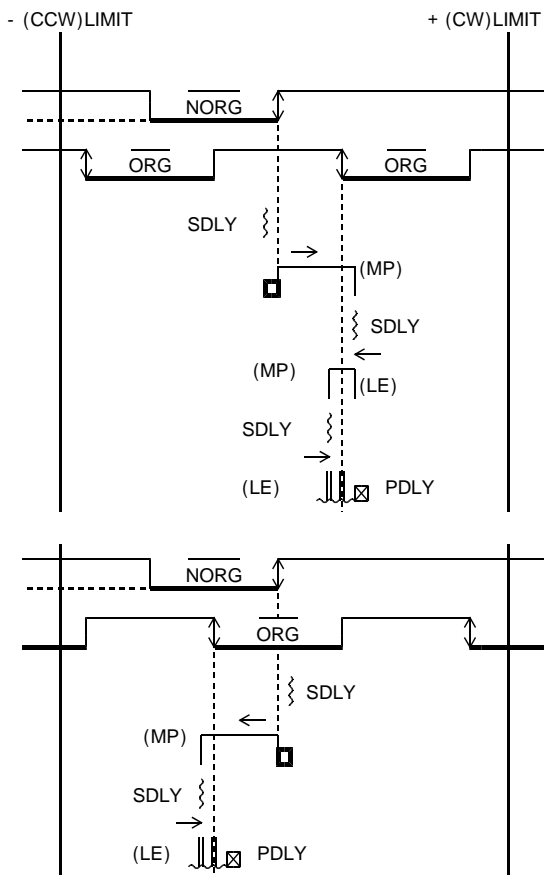
 SCAN DELAY TIME を挿入します
 CONSTANT SCAN 工程を行います
 検出信号の CW 側エッジ検出で停止します

 SCAN DELAY TIME を挿入します
 1PULSE 送り工程を行います
 検出信号の CW 側エッジ検出で停止します

* 原点センサに検出幅が狭い Z 相を用いる場合、レベルエラーになる場合があります。
 このようなときは、ORIGIN SPEC SET 関数の SENSOR ERROR TYPE を
 「レベルエラーを無視して次工程に進む」の設定にしてください。

ORG-5 型式の ORIGIN 工程

ORIGIN ドライブの起動方向を、- (CCW)方向として説明します。
 起動方向が CW 方向の場合は、対称の動作で、対称方向のエッジを検出します。
 ORG 検出信号には、回転軸のスリットなど周期的に信号を発生するセンサ信号を入力します。
 CONSTANT SCAN 工程の速度でセンサを通過したときに、1ms 以上の幅の信号が検出されるように
 します。



- 開始位置 (MP): MARGIN パルス挿入
- ☒ 終了位置 (LE): レベルエラーチェック
- SDLY: SCAN DELAY LDLY: LIMIT DELAY
- PDLY: PULSE DELAY

NORG 検出時に ORG がハイレベルのとき
 SCAN DELAY TIME を挿入します
 CONSTANT SCAN 工程を行います
 検出信号の CCW 側エッジ検出で停止します

 SCAN DELAY TIME を挿入します
 CONSTANT SCAN 工程を行います
 検出信号の CCW 側エッジ検出で停止します

 SCAN DELAY TIME を挿入します
 1PULSE 送り工程を行います
 検出信号の CCW 側エッジ検出で停止します

NORG 検出時に ORG がローレベルのとき
 SCAN DELAY TIME を挿入します
 CONSTANT SCAN 工程を行います
 検出信号の CCW 側エッジ検出で停止します

 SCAN DELAY TIME を挿入します
 1PULSE 送り工程を行います
 検出信号の CCW 側エッジ検出で停止します

* 原点センサに検出幅が狭い Z 相を用いる場合、レベルエラーになる場合があります。
 このようなときは、ORIGIN SPEC SET 関数の SENSOR ERROR TYPE を
 「レベルエラーを無視して次工程に進む」の設定にしてください。

(7) ORG-10 ドライブ型式

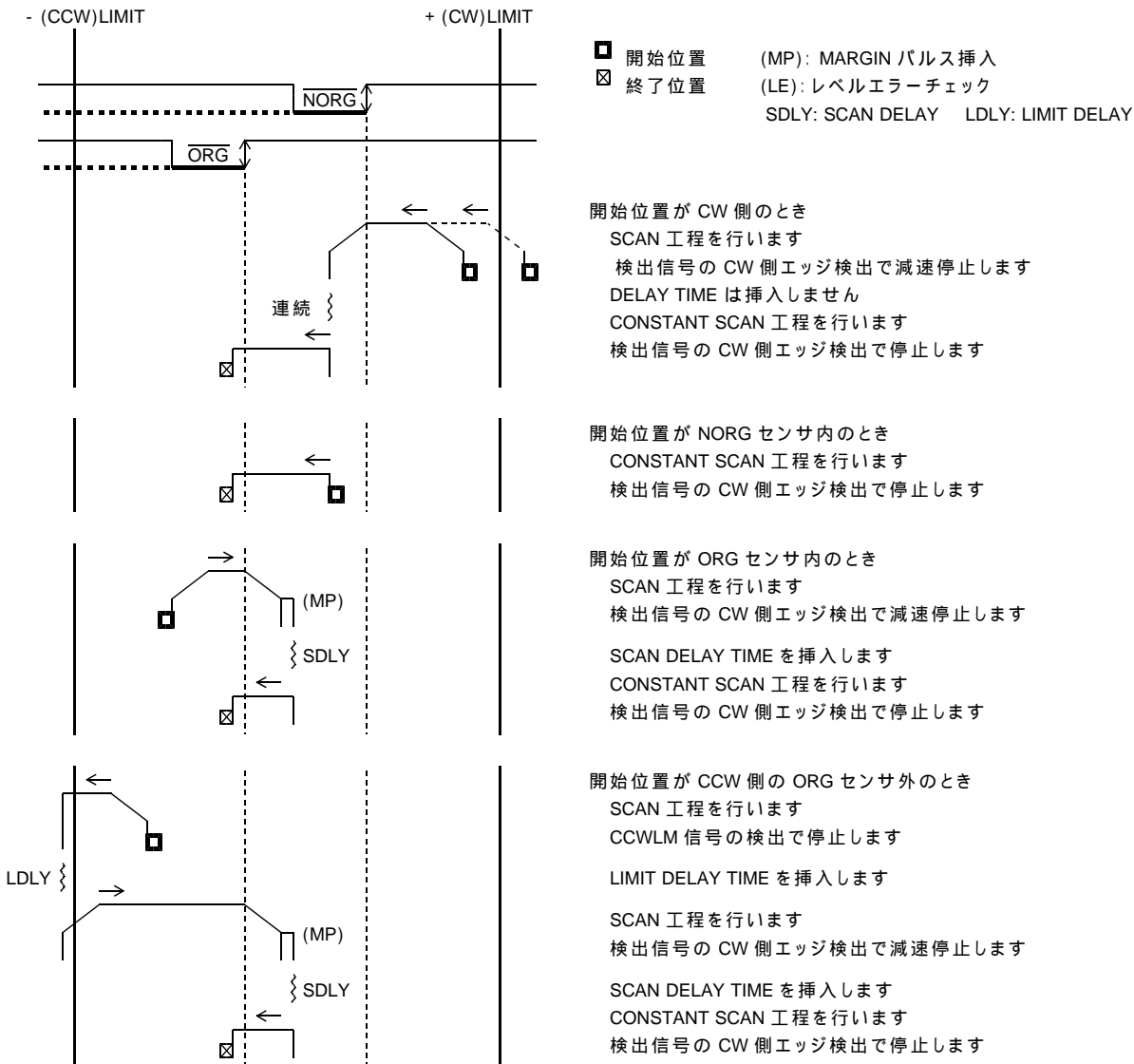
ORIGINドライブの起動方向を、- (CCW)方向として説明します。

起動方向が + (CW)方向の場合は、対称の動作で、対称方向のエッジを検出します。

ORG-10 型式は、NORG 検出信号と ORG 検出信号で機械原点を検出します。

検出信号には、1つのパルス、または - (CCW)側レベル保持のセンサ信号を入力します。

最高速度でセンサを通過したときに、1ms 以上の幅の信号が検出されるようにします。

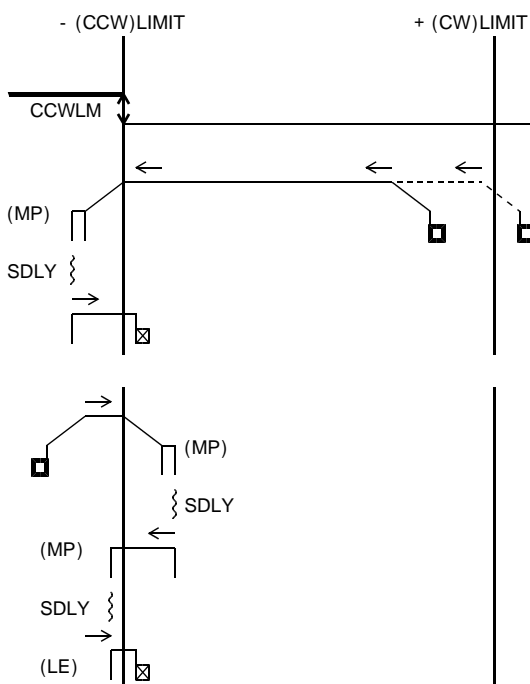


(8) ORG-11 ドライブ型式

注意

メカ限界点へぶつかり、メカや加工品などを破損させるおそれがあります。
RATE,HSPD などを変更した場合、停止点が変わるのでメカ限界点までの距離を
確認し直してください。
ORG-11 型式では ORG 検出中の LIMIT 停止は減速停止になります。

起動方向が CCW 方向の場合は、CCWLM 信号の CW 側エッジ検出で機械原点を検出します。
起動方向が CW 方向の場合は、CWLM 信号の CCW 側エッジ検出で機械原点を検出します。
ORIGIN ドライブの起動方向を、-(CCW)方向として説明します。
起動方向が+(CW)方向の場合は、対称の動作で、機械原点を検出します。
CCWLM 信号には、1つのパルス、または -(CCW)側レベル保持のセンサ信号を入力します。
最高速度でセンサを通過したときに、1ms 以上の幅の信号が検出されるようにします。
SCAN 工程では、CCWLM 信号検出後の停止機能は減速停止になります。
CCWLM 信号からシステムの -(CCW)方向の限界までの距離は、減速停止するのに十分な距離にします。



- 開始位置 (MP): MARGIN パルス挿入
- ☒ 終了位置 (LE): レベルエラーチェック
- SDLY: SCAN DELAY

開始位置が CW 側のとき
SCAN 工程を行います
検出信号の CW 側エッジ検出で減速停止します

SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します

開始位置が CCW LIMIT 内のとき
SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します

SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します

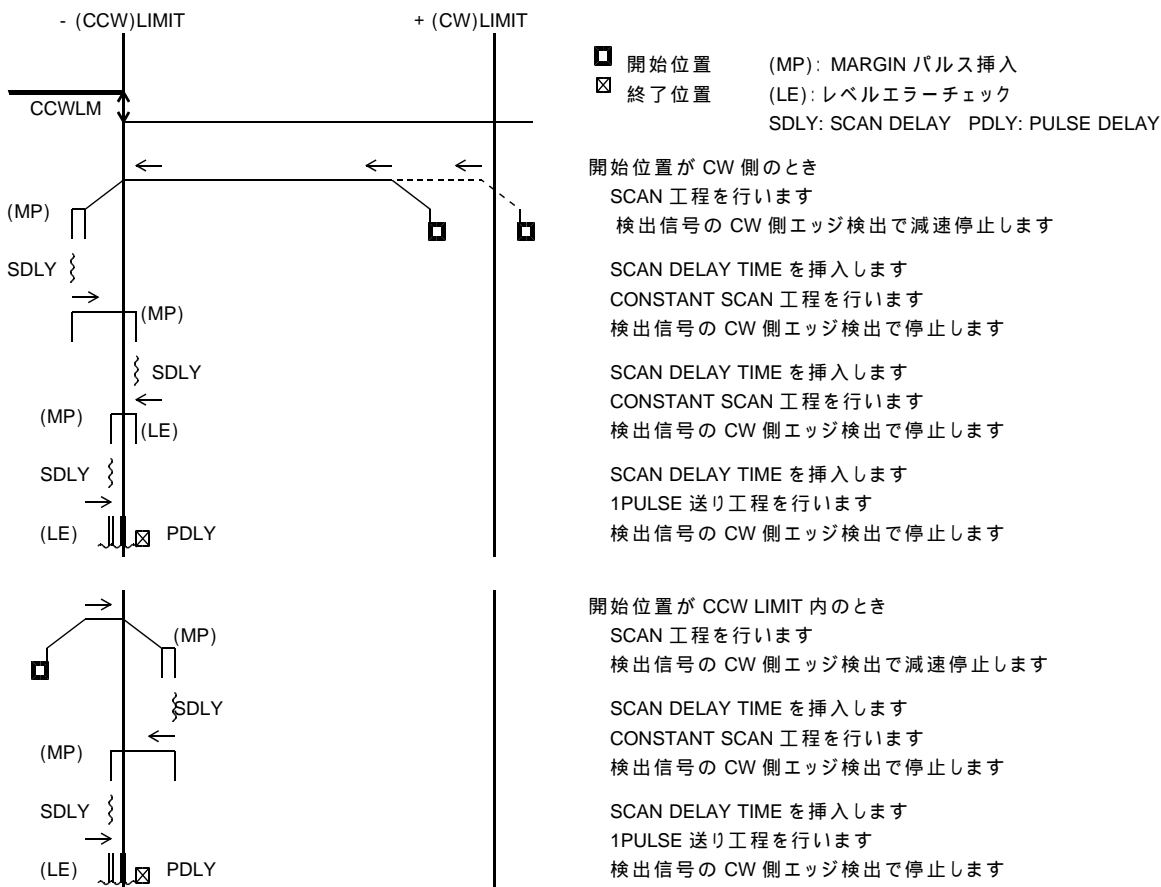
SCAN DELAY TIME を挿入します
CONSTANT SCAN 工程を行います
検出信号の CW 側エッジ検出で停止します

(9) ORG-12 ドライブ型式

注意

メカ限界点へぶつかり、メカや加工品などを破損させるおそれがあります。
速度パラメータを変更した場合、停止点が変わるのでメカ限界点までの距離を確認し直してください。
ORG-12 型式では ORG 検出中での LIMIT 停止は、「減速停止」になります。

起動方向が CCW 方向の場合は、CCWLM 信号の CW 側エッジ検出で機械原点を検出します。
起動方向が CW 方向の場合は、CWLM 信号の CCW 側エッジ検出で機械原点を検出します。
ORIGIN ドライブの起動方向を、-(CCW)方向として説明します。
ORG-12 型式は、ORG-11 型式に JOG 工程を付加して精度を高めた型式です。

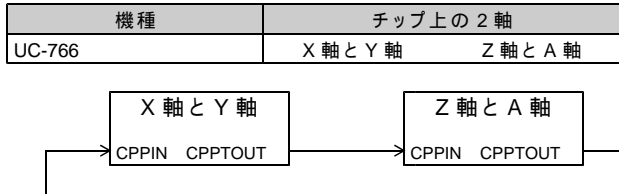


6-1-5. 補間ドライブ

(1) 補間ドライブ仕様

2 軸製品では、メインチップ 2 軸相対アドレス直線補間ドライブ関数またはメインチップ 2 軸相対アドレス円弧補間ドライブ関数を使用してください。

4 軸製品では、2 軸/1 チップの MCC07 を 2 チップ搭載してデジチェーン接続されています。



- ・同一チップで 2 軸補間を行う場合は、メインチップ 2 軸相対アドレス直線補間ドライブ関数または、メインチップ 2 軸相対アドレス円弧補間ドライブ関数を使用してください。
- ・チップ間の任意の 2 軸補間を行う場合は、2 軸相対アドレス直線補間ドライブ関数または、2 軸相対アドレス円弧補間ドライブ関数を使用してください
- ・同一チップ内の 2 軸(X 軸-Y 軸, Z 軸-A 軸)の 2 軸補間ドライブは同時実行が可能です。
- ・チップをまたいだ任意の 2 軸(X 軸-Z 軸, Y 軸-A 軸等)の 2 軸補間ドライブでは同時実行できません。
- ・1 台のコントローラ上で任意多軸(3 軸 ~ 4 軸)の直線補間ドライブを行う場合は、応用機能編をご覧ください。

デバイスドライバでは、次の補間ドライブ関数を用意しています。

- ・メインチップ 2 軸相対アドレス直線補間ドライブ関数(2 軸相関直線補間ドライブ)
- ・メインチップ 2 軸相対アドレス円弧補間ドライブ関数 (2 軸相関円弧補間ドライブ)
- ・2 軸相対アドレス直線補間ドライブ関数
- ・2 軸相対アドレス円弧補間ドライブ関数
- ・円の中心点ゲット関数
- ・相対アドレス変換関数

メインチップ 2 軸相対アドレス直線補間ドライブ

MCC07 同一チップ上の X-Y 軸、または Z-A 軸による 2 軸補間ドライブです。
相対アドレスで指定された目的地まで、2 軸直線補間ドライブを行います。

メインチップ 2 軸相対アドレス円弧補間ドライブ

MCC07 同一チップ上の X-Y 軸、または Z-A 軸による 2 軸補間ドライブです。
相対アドレスで指定された中心点と目的地で、2 軸円弧補間ドライブを行います。

- ・メインチップ 2 軸補間ドライブは、X-Y 軸または Z-A 軸の内、X 軸または Z 軸の加減速パラメータで補間ドライバの基本 PULSE を発生し、発生した基本 PULSE を補間演算して補間 PULSE を出力します。

2 軸相対アドレス直線補間ドライブ関数

相対アドレスで指定された目的地まで、任意な 2 軸間で直線補間ドライブを行います。
・4 軸コントローラのみ対応しています。(2 軸コントローラドライバではサポートしていません。)

2 軸相対アドレス円弧補間ドライブ関数

相対アドレスで指定された中心点と目的地で、任意な 2 軸間で円弧補間ドライブを行います。
・4 軸コントローラのみ対応しています。(2 軸コントローラドライバではサポートしていません。)

- ・メイン軸の加減速パラメータで補間ドライブの基本パルスが発生し、発生した基本パルスを補間演算して補間パルスを出力します。

円の中心点ゲット関数

通過点、目的地から中心点を演算します。
この関数により、通過点と目的地による円弧補間ドライブが容易に実現できます。

相対アドレス変換関数

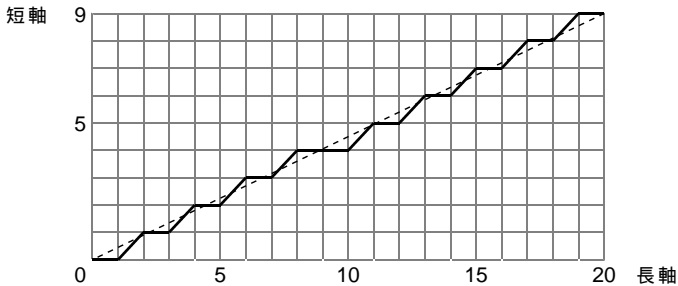
絶対アドレスから相対アドレスへの変換を行います。
この関数により、絶対アドレス指定での補間ドライブが容易に実現できます。

(2) 直線補間ドライブ仕様

相対アドレスで指定された目的地まで、2軸直線補間ドライブを行います。

- ・指定直線に対する位置誤差は、 $\pm 0.5\text{LSB}$ です。
- ・座標指定できる相対アドレス範囲は、 $-2,147,483,648 \sim +2,147,483,647$ (32ビット)です。

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。

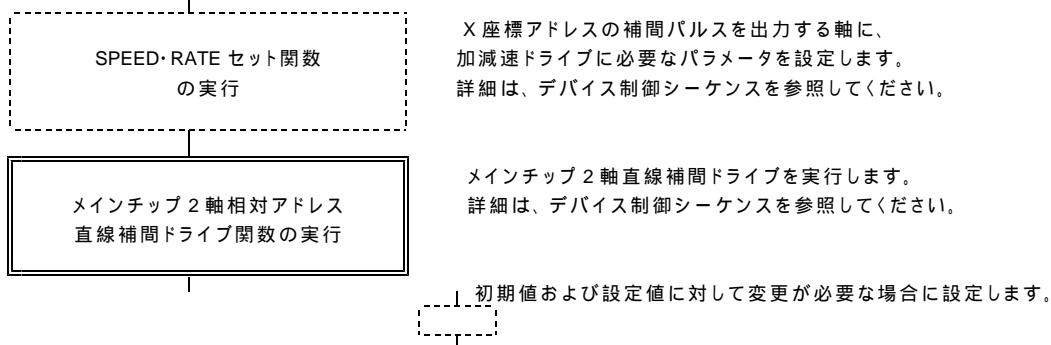
直線補間 SCAN ドライブの場合は、停止指令を検出するまで目的地の指定方向にパルス出力を続けます。

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。

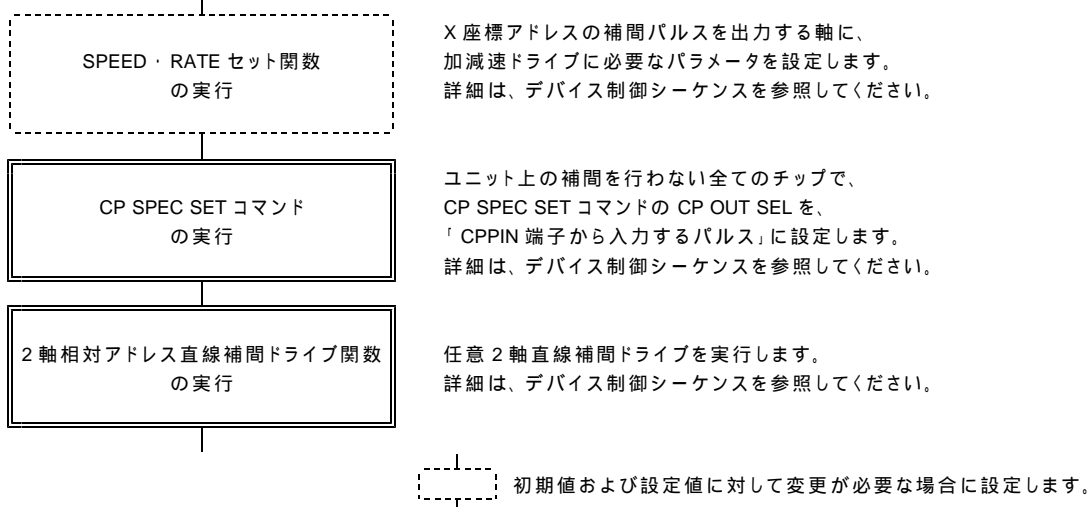
直線補間の長軸と短軸

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。

メインチップ 2 軸相対アドレス直線補間ドライブの実行シーケンス



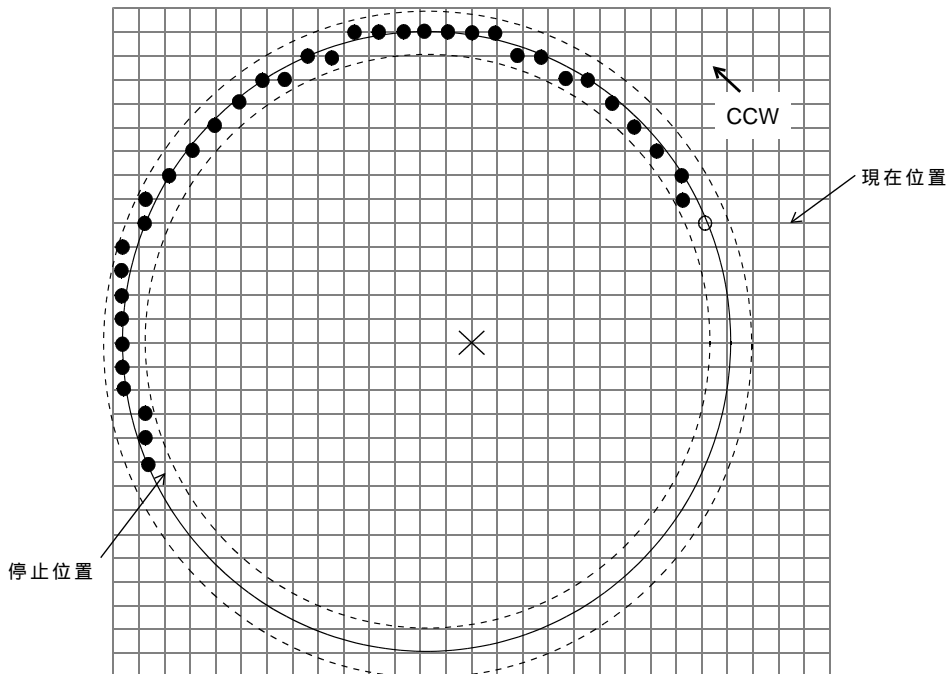
任意 2 軸直線補間ドライブの実行シーケンス



(3) 円弧補間ドライブ仕様

相対アドレスで指定された中心点と目的地で、2軸円弧補間ドライブを行います。

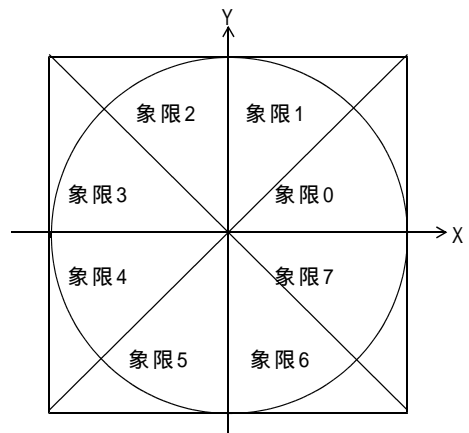
- ・円弧曲線に対する位置誤差は、 ± 1 LSB です。
- ・座標指定できる相対アドレス範囲は、 $-8,388,607 \sim +8,388,607$ です。

円弧補間ドライブの軌跡(CCW回転の例)

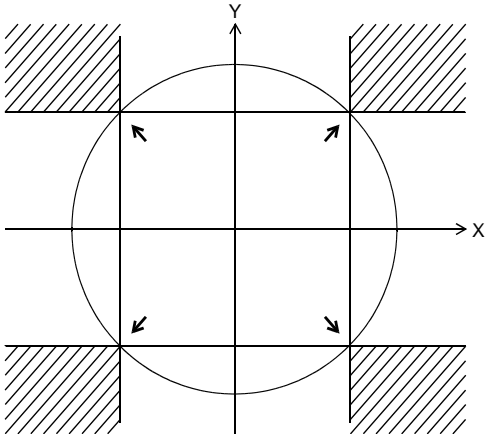
円弧補間ドライブの軌跡は、現在位置と円弧の中心点の距離を半径とした円周に沿います。
目的地が円周上に存在しない場合は、目的地と同じ象限内の短軸が一致した位置でドライブを終了します。

円弧補間の短軸

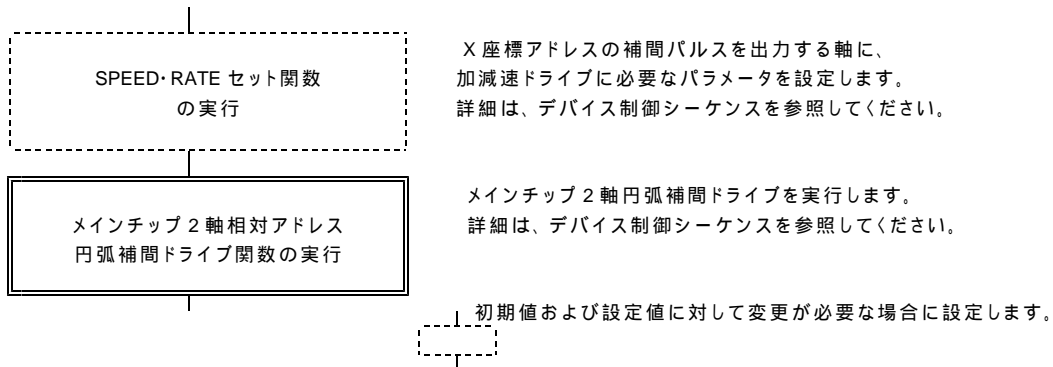
円弧補間の中心点(0,0)としたときに補間座標(X,Y)の絶対値が小さい方の軸が短軸になります。



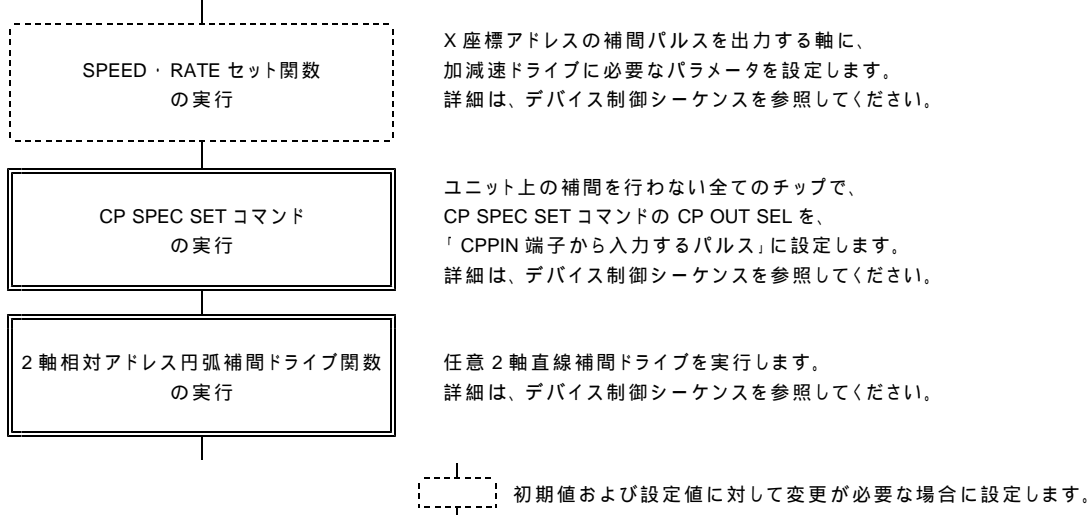
目的地が円周上に存在しない場合は、目的地と同じ象限内の短軸が一致した位置でドライブが終了しますが、目的地を下図の斜線部分に指定した場合は、各斜線部分と円周が接した点(下図の矢印)でドライブが終了します。



メインチップ 2 軸相対アドレス円弧補間ドライブの実行シーケンス



任意 2 軸円弧補間ドライブの実行シーケンス



(4) コマンド予約機能使用時の任意 2 軸補間ドライブの制限

2 軸相対アドレス直線補間ドライブ関数、および 2 軸相対アドレス円弧補間ドライブ関数を実行し、コマンド予約機能を使用した任意 2 軸直線補間・任意 2 軸円弧補間ドライブを行う場合、以下の使用方法に制限があります。

- ・コマンド予約機能の使用を開始するときは、両軸の DRIVE STATUS1 PORT の BUSY=0 および ERROR=0 を確認してください。
- ・補間ドライブ前に、予約コマンドを挿入する場合、補間の基本パルスを出力する軸(速度パラメータを設定する軸)にのみ挿入できます。
- ・コマンド予約機能を使用中は、補間の基本パルスを出力する軸を固定にしてください。

コマンド予約機能を使用した任意 2 軸補間ドライブの実行シーケンス

X 座標の補間パルスを出力する軸を Y 軸、Y 座標の補間パルスを出力する軸を Z 軸とした 2 軸相対アドレス直線補間ドライブの例で、コマンド予約機能を使用した任意 2 軸補間ドライブのシーケンスを示します。



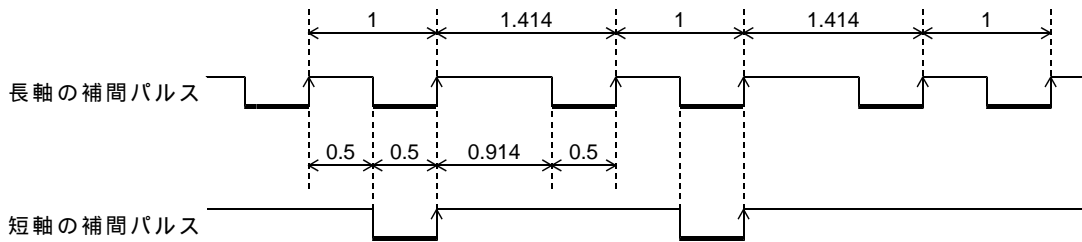
(5) 線速一定制御

補間ドライブする2軸の合成速度を一定にする制御です。

- ・補間ドライブの基本パルスを線速一定制御します。
- ・設定された速度が合成速度に反映されます。
- ・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。

線速一定の補間パルス出力(2軸直線補間ドライブの例)

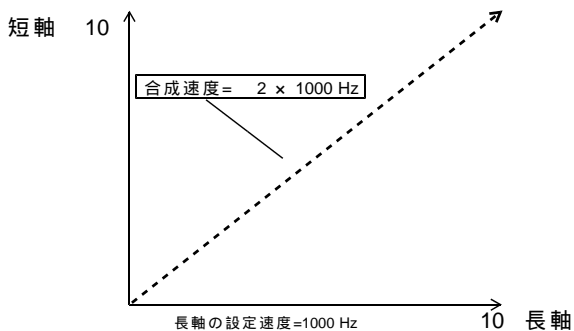
ON周期の幅はそのまま、OFF周期の幅が長くなります。



- ・直線補間ドライブでは、コマンド実行軸の長軸と短軸の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。
- ・線速一定制御は各補間ドライブの実行コマンドで設定します。

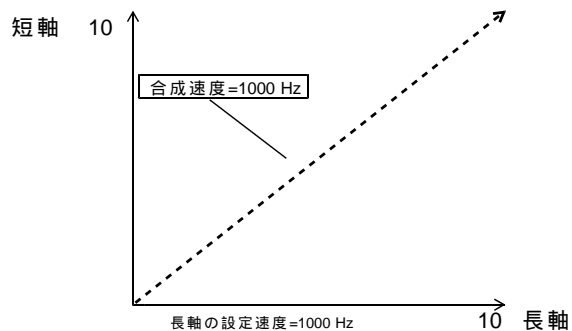
直線補間ドライブの軌跡(長軸10:短軸10の例)

<線速一定制御なしのとき>



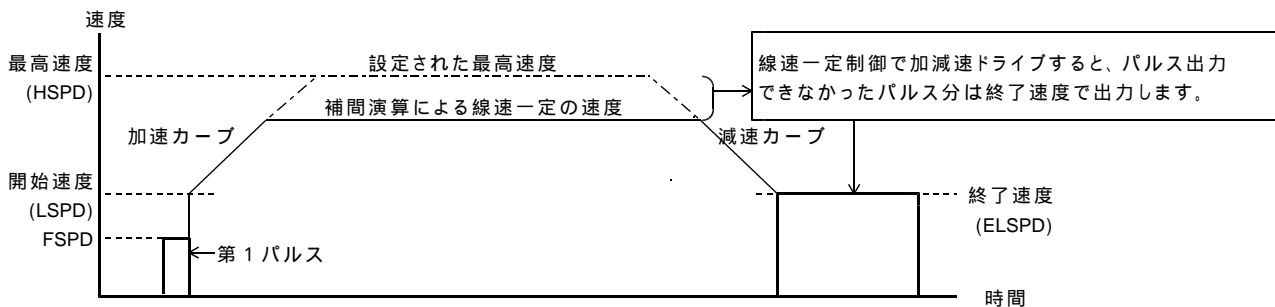
長軸の速度を一定速の1000Hzとすると、
2軸直線補間で描かれる軌跡の合成速度は
2 x 1000Hzでドライブします。

<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。



【注意事項】

線速一定制御には注意事項があります。

線速一定制御有効の円弧補間ドライブが、2 軸同時にパルス出力した位置で終了した場合に、以降に実行する線速一定制御有効の直線補間ドライブのパルス出力が、常に設定値の 1.414 倍の周期 (常に線速一定制御される) になります。

- ・ 1 軸のみパルス出力する位置 (例: 0 °, 90 °, 180 °, 270 °) で終了した場合は正常です。
- ・ 2 軸同時にパルス出力する位置 (例: 45 °, 135 °, 225 °, 315 °) で終了した場合に不具合が発生します。

線速一定制御有効の円弧補間ドライブ終了後は、

以下の円弧補間ドライブ (0 パルス、終了位置 0 °) を実行して、正常終了にしてください。

- ・ CIRCULAR XPOSITION SET コマンド (H'28) : H'00_0000 に設定
- ・ CIRCULAR YPOSITION SET コマンド (H'29) : H'00_0000 に設定
- ・ CIRCULAR PULSE SET コマンド (H'2A) : H'0000_0000 に設定
- ・ MAIN CIRCULAR CP コマンド (H'38) : DATA1=H'0001 で実行

6-1-6. パルス出力停止機能

パルス出力停止機能は、実行中のドライブを終了させる機能です。

パルス出力停止機能には、減速停止機能、即時停止機能、LIMIT 減速停止機能、LIMIT 即時停止機能があります。

ORIGIN ドライブ中の停止機能には、以下の制限があります。

- ・ LIMIT 信号 (CWLM, CCWLM) による即時停止
- ・ センサ信号 (SS0) による減速停止、および即時停止
- ・ DEND 信号による減速停止、および即時停止
- ・ 各種カウンタのコンパレータの一致による減速停止、および即時停止

(1) 減速停止機能

減速停止機能には、以下の減速停止指令があります。

- ・ SLOW STOP コマンド
- ・ 入力機能を減速停止に設定した SS0, DEND, DALM 信号
- ・ 停止機能を減速停止に設定した各種カウンタのコンパレータ出力

DRIVE STATUS1 PORT の STBY = 1 または DRIVE = 1 のときに有効になる停止機能です。

減速停止指令のアクティブを検出すると、実行中のドライブパルス出力を終了速度まで減速してから、パルス出力を停止後にドライブを終了します。

- ・ STBY = 1 のときに検出した場合は、パルス出力なしでドライブを終了します。
- ・ 基本パルスを発生しない軸の補間ドライブ実行中に検出した場合は、減速なしで停止します。

減速停止指令の検出と同時に、DRIVE STATUS1 PORT の SSEND = 1 にします。

(2) 即時停止機能

即時停止機能には、以下の即時停止指令があります。

- ・ FAST STOP コマンド
- ・ FSSTOP 信号入力による即時停止
- ・ 入力機能を即時停止に設定した SS0, DEND, DALM 信号
- ・ 入力機能を即時停止に設定した CWLM, CCWLM 信号
- ・ 停止機能を即時停止に設定した各種カウンタのコンパレータ出力

DRIVE STATUS1 PORT の BUSY = 1 のときに有効になる停止機能です。

即時停止指令を検出すると、BUSY = 0 になるまで、即時停止機能のアクティブを維持します。

- ・ 即時停止機能がアクティブでも、データ設定コマンドの処理は正常に実行します。

即時停止指令を検出すると、実行中のドライブを強制終了します。

- ・ DEND BUSY = 1 で検出した場合は、DEND 信号の<サーボ対応>を中止して、DEND BUSY = 0 にします。
- ・ DRST 信号の<サーボ対応>実行中は BUSY = 1 です。<サーボ対応>終了後に BUSY = 0 にします。
- ・ ORIGIN 停止機能の AUTO DRST 出力中に検出した場合は、DRST 信号の<サーボ対応>も実行します。DRST 出力終了後に BUSY = 0 にします。この場合の DRST 出力はリトリガ出力になります。
- ・ EXT PULSE = 1 で検出した場合は、ドライブの強制終了後も BUSY = 1 のままです。EXT PULSE = 1 の場合は、EXT PULSE = 0 にすると、BUSY = 0 になります。

即時停止指令の検出と同時に、DRIVE STATUS1 PORT の FSEND = 1 にします。

(3) LIMIT 停止機能

LIMIT 停止機能は、方向別のドライブパルス出力停止機能です。
LIMIT 停止機能には、LIMIT 減速停止機能と LIMIT 即時停止機能があります。
LIMIT 停止機能は、SPEC INITIALIZE2 コマンドで設定します。

- + 方向の LIMIT 停止機能 (CWLM 信号、各種カウンタの COMP2)
- + 方向の LIMIT 停止指令を検出すると、+ 方向のドライブを減速停止または即時停止します。
- 方向のドライブでは、+ 方向の LIMIT 停止指令は無効です。

- 方向の LIMIT 停止機能 (CCWLM 信号、各種カウンタの COMP3)
- 方向の LIMIT 停止指令を検出すると、- 方向のドライブを減速停止または即時停止します。
- + 方向のドライブでは、- 方向の LIMIT 停止指令は無効です。

LIMIT 減速停止機能

LIMIT 減速停止機能には、以下の LIMIT 減速停止指令があります。

- ・入力機能を LIMIT 減速停止に設定した CWLM, CCWLM 信号
- ・停止機能を LIMIT 減速停止に設定した各種カウンタのコンパレータ出力

DRIVE STATUS1 PORT の STBY = 1 または DRIVE = 1 のときに有効になる停止機能です。
DRIVE STATUS1 PORT の EXTPULSE = 1 の場合は、DRIVE = 1 のときに有効になります。
DRIVE STATUS2 PORT の DEND BUSY = 1 のときには、LIMIT 停止指令の検出のみ行います。

LIMIT 減速停止指令を検出すると、実行中のドライブパルス出力を終了速度まで減速してから、ドライブパルス出力を停止します。パルス出力停止後にドライブを終了します。

- ・ STBY = 1 のときに検出した場合は、パルス出力なしでドライブを終了します。
- ・ DEND BUSY = 1 で検出した場合は、LSEND = 1 になりますが、DEND BUSY = 1 は継続します。
- ・基本パルスを発生しない軸の補間ドライブ実行中に検出した場合は、減速なしで停止します。

LIMIT 減速停止指令の検出と同時に、DRIVE STATUS1 PORT の LSEND = 1 にします。

LIMIT 即時停止機能

LIMIT 即時停止機能には、以下の LIMIT 即時停止指令があります。

- ・入力機能を LIMIT 即時停止に設定した CWLM, CCWLM 信号
- ・停止機能を LIMIT 即時停止に設定した各種カウンタのコンパレータ出力

DRIVE STATUS1 PORT の STBY = 1 または DRIVE = 1 のときに有効になる停止機能です。
DRIVE STATUS1 PORT の EXTPULSE = 1 の場合は、DRIVE = 1 のときに有効になります。
DRIVE STATUS2 PORT の DEND BUSY = 1 のときには、LIMIT 停止指令の検出のみ行います。

LIMIT 即時停止指令を検出すると、実行中のドライブを強制終了します。

- ・ STBY = 1 のときに検出した場合は、パルス出力なしでドライブを終了します。
- ・ DEND BUSY = 1 で検出した場合は、LSEND = 1 になりますが、DEND BUSY = 1 は継続します。
- ・ DEND 信号または DRST 信号の<サーボ対応>実行中は BUSY = 1 にします。

- ・ EXTPULSE = 1 で検出した場合は、ドライブの強制終了後も BUSY = 1 のままです。
DEND BUSY = 0 の場合は、LIMIT 停止方向と逆方向のパルス出力ができます。

LIMIT 即時停止指令の検出と同時に、DRIVE STATUS1 PORT の LSEND = 1 にします。

6-1-7. MCC07 エラー機能

DRIVE STATUS1 PORT の ERROR フラグには、15 個の ERROR STATUS を OR (論理和) で出力します。

- ・ ERROR STATUS は、エラーの発生を検出して、“1” になります。
- ・ ERROR STATUS は、ERROR STATUS READ コマンドで読み出しできます。
- ・ ERROR STATUS は、動作エラークリア関数でクリアします。
- ・ ERROR に出力する ERROR STATUS は、ERROR STATUS MASK コマンドで個別にマスクできます。

DRIVE STATUS1 PORT の ERROR = 1 になると、以下のコマンドの書き込みを無効にします。

DRIVE STATUS1 PORT の ERROR = 0 にクリアすると、コマンドの書き込みを有効にします。

- ・ 汎用コマンド
- ・ SPEED CHANGE コマンド(応用機能)
- ・ INDEX CHANGE コマンド(応用機能)

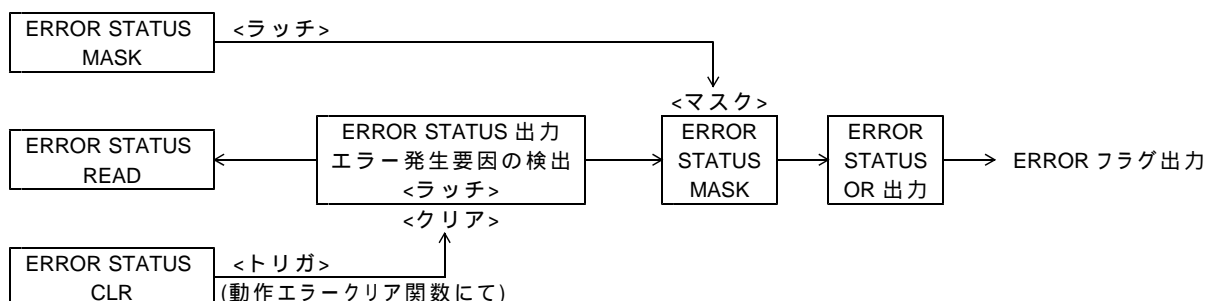
ドライブ中に DRIVE STATUS1 PORT の ERROR = 1 を検出した場合は、以下のようになります。

- ・ STBY = 1 のときに ERROR = 1 を検出した場合は、パルス出力なしでドライブを終了します。
- ・ ドライブ実行中に停止要因の ERROR = 1 を検出した場合は、停止要因の停止機能で停止します。
- ・ ドライブ実行中に停止要因以外の ERROR = 1 を検出した場合は、減速停止します。
- ・ コマンド予約機能(応用機能)によりコマンドを予約している状態で ERROR フラグ=1 になると、予約コマンドを全てクリアしてインターロック状態になります。
- ・ ERROR フラグ=1 の間は COMREG FL=1, COMREG EP=1 になります。

ERROR STATUS

ERROR STATUS	エラー内容
COMMAND ERROR	未定義の汎用コマンドを実行した。
COMREG CLR ERROR	コマンド予約機能で格納している実行待ちの予約コマンドをクリアした。
INC INDEX ERROR	相対アドレスのオーバーフローで、INC INDEX ドライブを終了した。
ABS INDEX ERROR	アドレスカウンタのオーバーフローで、ABS INDEX ドライブを終了した。
INDEX CHANGE ERROR	反転動作が必要な INDEX CHANGE 指令を検出した。
CHANGE CLR ERROR	実行待ちの INDEX CHANGE 指令を無効にした。
CPP STOP ERROR	補間ドライブのメイン軸の CPP STOP 機能でドライブを終了した。
EXT PULSE ERROR	外部パルス出力機能を実行中に正常な外部パルス出力ができなかった。
FSEND ERROR	BUSY = 1 のときに、DRIVE STATUS1 PORT の FSEND = 1 を検出した。
LSEND ERROR	BUSY = 1 のときに、DRIVE STATUS1 PORT の LSEND = 1 を検出した。
SSEND ERROR	BUSY = 1 のときに、DRIVE STATUS1 PORT の SSEND = 1 を検出した。
ADDRESS OVFL ERROR	BUSY = 1 のときに、DRIVE STATUS4 PORT の ADDRESS OVFL = 1 を検出した。
PULSE OVFL ERROR	DRIVE STATUS4 PORT の PULSE OVFL = 1 を検出した。
DALM ERROR	DALM 信号のアクティブ入力を検出した。
FSSTOP ERROR	FSSTOP 信号のアクティブ入力を検出した。
-	-

エラー発生要因と ERROR 出力の構成



【注意】

停止機能を ERROR=1 の発生要因に設定している場合で、
予約コマンドを格納したドライブを実行して、ERROR=1 が発生した場合は、
STATUS1 PORT の DRVEND, LSEND, SSEND フラグが "1" にならない場合があります。

予約コマンドを格納したドライブを実行して、ERROR=1 が発生した場合は、
以下のフラグで停止・終了を確認してください。

- ・停止要因は、ERROR STATUS の FSEND ERROR, LSEND ERROR, SSEND ERROR で確認する。
- ・ドライブの終了は、STATUS1 PORT の BUSY=0 で確認する。

6-1-8. 読み出し機能

(1) ステータス読み出し

これらの STATUS PORT の読み出しは、3-7.章の制限事項を除き、常時可能です。

DRIVE STATUS1 PORT

ドライブコントロールの現在の状態を表示する PORT です。
DRIVE STATUS1 PORT 読み出し関数で読み出します。

DRIVE STATUS2 PORT

外部入出力信号の状態を表示する PORT です。
DRIVE STATUS2 PORT 読み出し関数で読み出します。

DRIVE STATUS3 PORT

割り込み要求出力とステータス信号の状態を表示する PORT です。
DRIVE STATUS3 PORT 読み出し関数で読み出します。

DRIVE STATUS4 PORT

カウンタのコンパレータ出力状態とオーバーフローを表示する PORT です。
DRIVE STATUS4 PORT 読み出し関数で読み出します。

DRIVE STATUS5 PORT

入力信号とドライブ CHANGE 指令の現在の状態を表示する PORT です。
DRIVE STATUS5 PORT 読み出し関数で読み出します。

DRIVE STATUS バッファ

上記 DRIVE STATUS1 PORT から DRIVE STATUS5 PORT までと ORIGIN STATUS を一括で読み出すことができます。
DRIVE STATUS バッファ読み出し関数で読み出します。

(2) 各データの読み出し

これらデータの読み出しは、3-7.章の制限事項を除き、常時可能です。

以降の読み出しは、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。

- ・ユニット単位 ... ユニット DRIVE COMMAND 書き込み / 読み出し関数
- ・デバイス単位 ... DRIVE COMMAND 32 ビット書き込み / 読み出し関数

設定データの読み出し

SET DATA READ コマンドを実行すると、MCC07 に設定したデータが読み出せます。

出力中のドライブ速度の読み出し

MCC SPEED READ コマンドを実行すると、MCC07 に現在出力中のドライブパルス速度が読み出せます。読み出されたデータは、「ドライブパルス速度 (Hz) の 10 倍」のパルス速度データです。

エラーステータスの読み出し

ERROR STATUS READ コマンドを実行すると、MCC07 で発生しているエラーの状態が読み出せます。

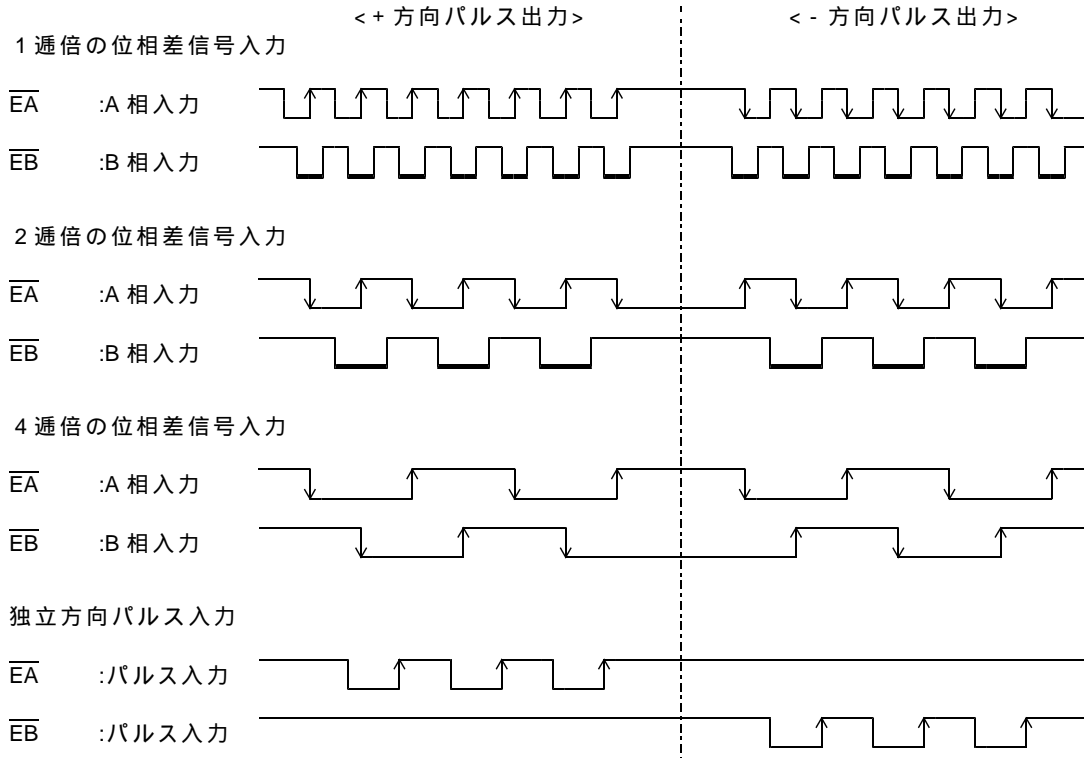
カウンタデータの読み出し

各カウンタ READ コマンドを実行すると、MCC07 の現在のカウンタデータが読み出せます。ADDRESS COUNTER READ コマンドを実行すると、アドレスカウンタのカウンタデータが読み出せます。PULSE COUNTER READ コマンドを実行すると、パルスカウンタのカウンタデータが読み出せます。DFL COUNTER READ コマンドを実行すると、パルス偏差カウンタのカウンタデータが読み出せます。

6-2.カウンタ仕様

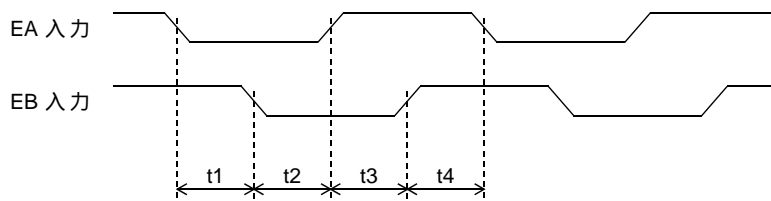
(1) エンコーダパルス入力方式

EA, EB 信号に外部パルス信号を入力して各カウンタでカウントできます。
カウント方法はカウンタ毎に以下の 4 種類の中から選択できます。



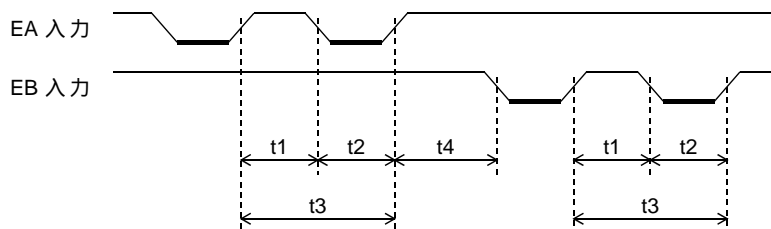
- ・矢印は入力パルスのカウントエッジです。
- ・各カウンタのパルスカウント方法は各 COUNTER INITIALIZE1 コマンドで行います。

【位相差信号入力タイミング】



- ・アドレスカウンタの場合
 - 2 通倍のとき $t_1, t_2, t_3, t_4 > 50 \text{ ns}$
 - $t_1 + t_2 \quad 200 \text{ ns}$
 - $t_3 + t_4 \quad 200 \text{ ns}$
- 4 通倍のとき $t_1, t_2, t_3, t_4 \quad 200 \text{ ns}$
- ・その他のカウンタの場合
 - $t_1, t_2, t_3, t_4 > 50 \text{ ns}$

【独立方向パルス入力タイミング】



- ・アドレスカウンタの場合
 - $t_1, t_2, t_4 > 50 \text{ ns}$
 - $t_3 \quad 200 \text{ ns}$
- ・その他のカウンタの場合
 - $t_1, t_2, t_4 > 50 \text{ ns}$
 - $t_3 > 100 \text{ ns}$

COUNTER INITIALIZE1 コマンドの EXT COUNT DIRECTION = 0 のときのカウント方向です。

(2) 外部パルス出力機能

アドレスカウンタのカウンタパルスを「外部パルス信号」に設定すると、CWP, CCWP 端子から、外部パルス信号のカウンタタイミングをパルス出力します。カウンタパルスを「自軸のパルス」に設定すると、外部パルス信号の出力を終了します。

外部パルス出力は、ADDRESS COUNTER INITIALIZE1 コマンドで設定します。

- ・ COUNT PULSE SEL で、出力する外部パルスを選択します。
- ・ EXT COUNT TYPE で、出力する外部パルスのカウント方法を選択します。
- ・ EXT COUNT DIRECTION で、出力する外部パルスの出力方向を選択します。
- ・ EXT PULSE TYPE で、出力する外部パルスのアクティブ幅を選択します。

EXT PULSE TYPE で選択したアクティブ幅の2倍の時間内に、次の外部パルスのカウンタタイミングが入力した場合は、正常なパルス出力ができません。この場合は、ERROR STATUS の EXT PULSE ERROR = 1 にします。

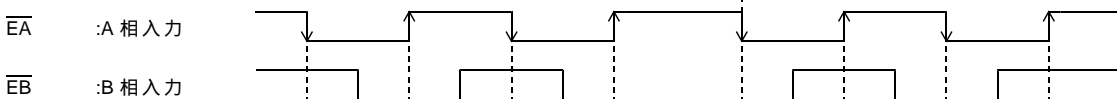
LIMIT 停止指令を検出すると、検出方向の外部パルス出力を停止して、STBY 状態にします。減速停止指令、即時停止指令、ORGEND = 1 または ERROR = 1 を検出すると、外部パルス出力を停止して、外部パルス出力機能を無効状態にします。

DRIVE STATUS1 PORT の EXT PULSE = 1 でも、コマンド予約機能、同期スタート機能、DEND, DRST 信号のサーボ対応機能が有効です。また、DRIVE STATUS1, 2 PORT の以下のフラグも有効です。

- ・ BUSY、STBY、DRIVE、DRVEND、ERROR、LSEND、SSEND、FSEND、PAUSE、COMREG EP、COMREG FL、ORGEND、DEND BUSY

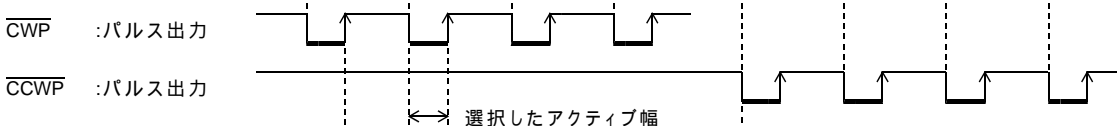
<入力パルス>

2 通倍の位相差信号入力

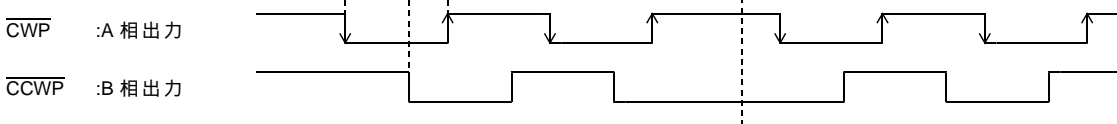


<出力パルス>

独立方向出力



2 通倍の位相差信号出力



・ 方向指定出力の場合は、カウンタタイミングの入力でパルスの出力方向が確定するため、方向出力信号の変化とアクティブ幅の立ち下がりエッジ出力が同時になります。

・ 2 通倍の位相差信号出力の場合は、EXT PULSE TYPE で選択したアクティブ幅が、出力信号の位相差になります。

外部パルス出力中のステータスと停止機能

外部パルス出力がアクティブレベルを出力中に、外部パルス出力の停止要因を検出した場合は、出力中のパルスのアクティブ幅を確保した後にパルス出力を停止します。

外部パルス出力中のステータスフラグは、以下のように変化します。

外部パルス出力の開始と終了

- ・ EXT PULSE = 0、BUSY = 0、ERROR = 0 のときに、COUNT PULSE SEL の「01, 10, 11」(他軸の発生パルス、外部パルス信号)設定を検出すると、EXT PULSE = 1、BUSY = 1、STBY = 1、DRIVE = 0 になります。
- ・ EXT PULSE = 0、BUSY = 1 のときに、COUNT PULSE SEL を「01, 10, 11」に設定すると、現在の BUSY = 1 状態終了後に、EXT PULSE = 1、BUSY = 1 になります。
- ・ EXT PULSE = 1、STBY = 1 の状態は、出力する外部パルス信号の入力待ちの状態です。
- ・ 出力する外部パルス信号を検出すると、外部パルス出力を開始して、EXT PULSE = 1、BUSY = 1、STBY = 0、DRIVE = 1 になります。EXT PULSE = 1、DRIVE = 1 の状態は、外部パルス出力中の状態です。
- ・ EXT PULSE = 1 のときに、COUNT PULSE SEL の「00」(自軸の発生パルス)設定を検出すると、EXT PULSE = 0、BUSY = 0 になります。EXT PULSE = 0、BUSY = 0 の状態は、外部パルス出力を終了した状態です。
- ・ STBY = 1 または DRIVE = 1 のときに COUNT PULSE SEL の「00」を検出した場合は、DEND 信号の<サーボ対応>も実行します。<サーボ対応>実行中は、EXT PULSE = 1 です。

LIMIT 停止機能による外部パルス出力の停止

- ・ EXT PULSE = 1 のときに、LIMIT 停止指令を検出すると、外部パルス出力を停止して、EXT PULSE = 1、BUSY = 1、STBY = 1、DRIVE = 0 にします。EXT PULSE = 1、STBY = 1 の状態は、出力する外部パルス信号の入力待ちの状態です。LIMIT 停止指令がアクティブ状態でも、LIMIT 停止指令と反対方向の外部パルスが出力できます。
- ・ LIMIT 減速停止指令は、DRIVE = 1、DEND BUSY = 1 のときに検出します。LIMIT 即時停止指令は、DRIVE = 1、DEND BUSY = 1 のときに検出します。
- ・ LSEND フラグも変化します。DEND 信号または DRST 信号の<サーボ対応>も実行します。<サーボ対応>実行中は、STBY = 0、DRIVE = 0 です。

その他の停止機能による外部パルス出力機能の無効

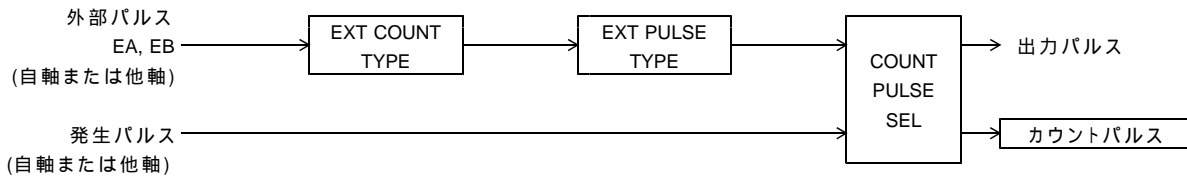
- ・ EXT PULSE = 1 のときに、減速停止指令、即時停止指令、ORGEND = 1 または ERROR = 1 を検出すると、外部パルス出力を停止して、EXT PULSE = 1、BUSY = 1、STBY = 0、DRIVE = 0 にします。
- ・ 減速停止指令は、STBY = 1 または DRIVE = 1 のときに検出します。即時停止指令、ORGEND = 1 および ERROR = 1 は、BUSY = 1 のときに検出します。
- ・ SSEND、FSEND、ORGEND フラグも変化します。DEND 信号または DRST 信号の<サーボ対応>も実行します。
- ・ SSEND = 1、FSEND = 1、ORGEND = 1 または ERROR = 1 で外部パルス出力を停止した状態は、外部パルス出力機能が無効の状態です。COUNT PULSE SEL を「00」に設定して、外部パルス出力を終了させてください。

(3) アドレスカウンタ

ドライブパルス出力をカウントして、絶対アドレスを管理する 32 ビットのカウンタです。カウントパルスを外部パルス信号に設定すると、「外部パルス出力」になります。3 個の専用コンパレータは、任意のカウント値を検出して ADRINT を出力します。コンパレータの一致検出で、ドライブパルス出力を停止させることができます。

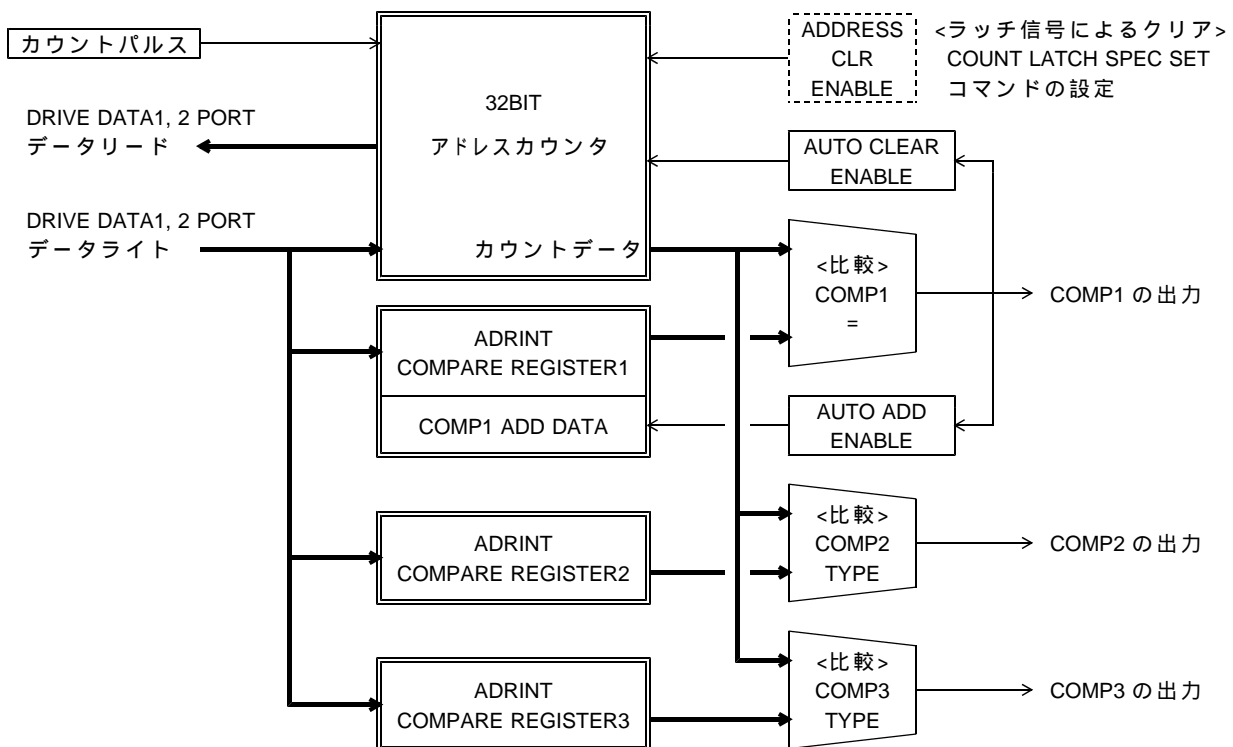
- ・+(CW)方向のパルスでカウントアップ、-(CCW)方向のパルスでカウントダウンします。
- ・カウンタの有効範囲は、-2,147,483,647 ~ +2,147,483,647 (H'8000_0001 ~ H'7FFF_FFFF) です。負数の場合は、2 の補数表現になります。

アドレスカウンタのパルス選択部



- ・アドレスカウンタのパルス選択機能は ADDRESS COUNTER INITIALIZE1 コマンドで設定します。
- ・アドレスカウンタのパルス選択機能で外部パルスを選択した場合、CWP, CCWP から出力するパルスは外部パルスのタイミングで発生します。詳細は 5-2-2.章「外部パルス出力機能」を参照してください。
- ・UC-766 以外の製品は、外部パルスの選択はできません。

アドレスカウンタとコンパレータの構成



- ・アドレスカウンタとコンパレータの機能は ADDRESS COUNTER INITIALIZE1 コマンドで設定します。
- ・アドレスカウンタの現在値は ADDRESS COUNTER PRESET コマンドで設定します。
- ・アドレスカウンタの現在値は ADDRESS COUNTER READ コマンドで読み出せます。

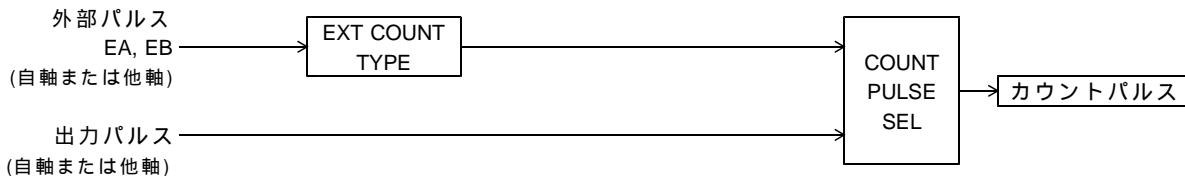
(4) パルスカウンタ

外部パルス信号をカウントして、実位置を管理する 32 ビットのカウンタです。
ドライブ出力パルスのカウントもできます。

3 個の専用コンパレータは、任意のカウント値を検出して CNTINT を出力します。
コンパレータの一致検出で、ドライブパルス出力を停止させることができます。

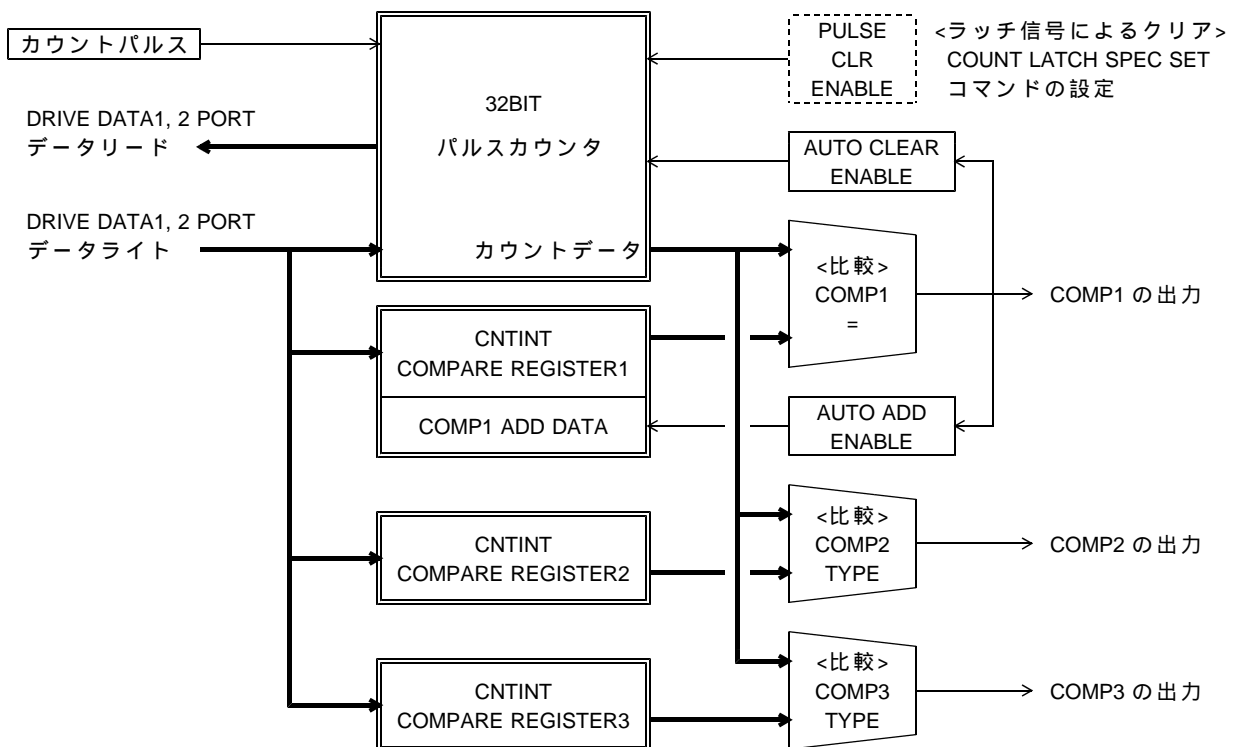
- ・+(CW)方向のパルスでカウントアップ、-(CCW)方向のパルスでカウントダウンします。
- ・カウンタの有効範囲は、-2,147,483,647 ~ +2,147,483,647 (H'8000_0001 ~ H'7FFF_FFFF) です。
負数の場合は、2 の補数表現になります。

パルスカウンタのパルス選択部



- ・パルスカウンタのパルス選択機能は PULSE COUNTER INITIALIZE1 コマンドで設定します。
- ・UC-766 以外の製品は、外部パルスの選択はできません。

パルスカウンタとコンパレータの構成



- ・パルスカウンタとコンパレータの機能は PULSE COUNTER INITIALIZE1 コマンドで設定します。
- ・パルスカウンタの現在値は PULSE COUNTER PRESET コマンドで設定します。
- ・パルスカウンタの現在値は PULSE COUNTER READ コマンドで読み出せます。

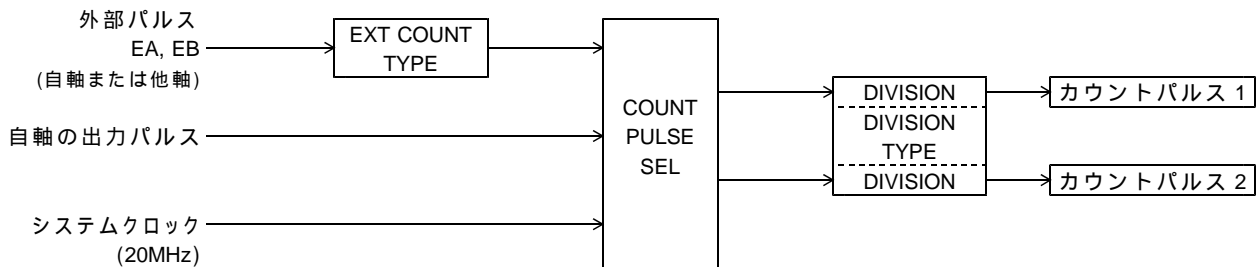
(5) パルス偏差カウンタ

外部パルス信号とドライブ出力パルスの2種のパルスをカウントして、パルス数の偏差を検出する16ビットのカウンタです。

システムクロック(20MHz)のみをカウントしてタイマとして使用することもできます。カウントパルスには、分周機能があります。

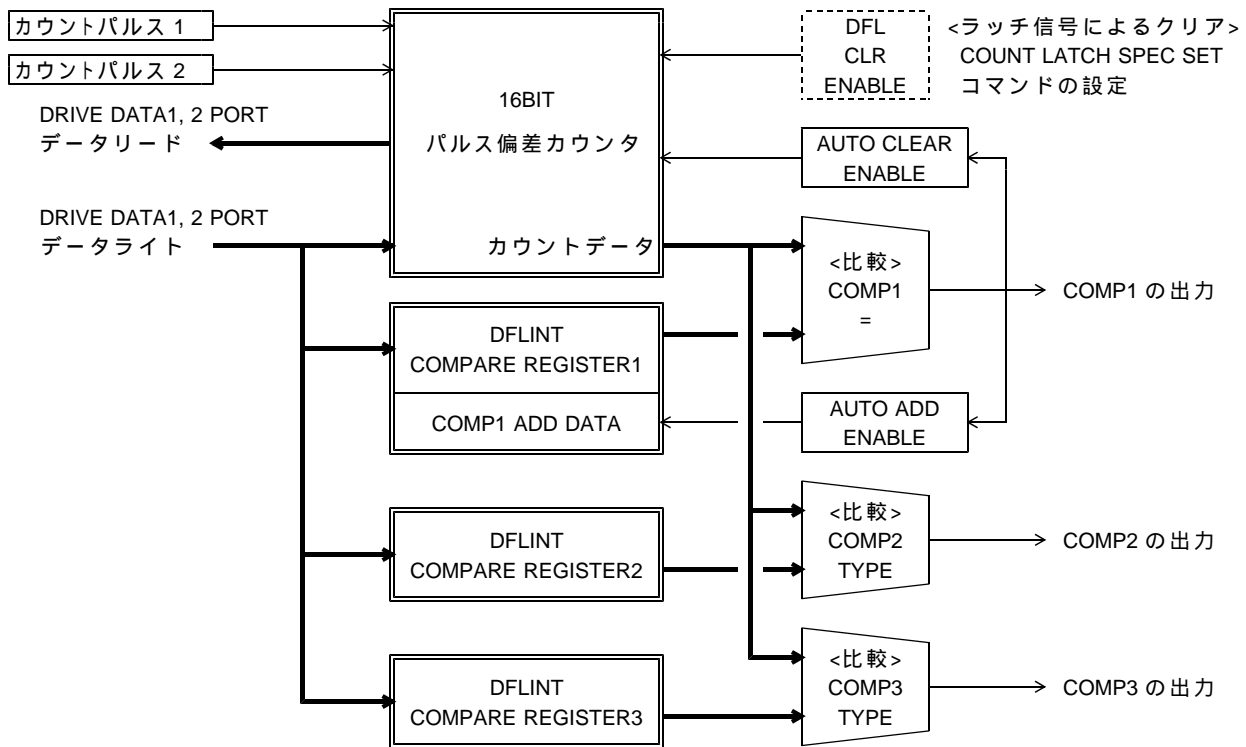
3個の専用コンパレータは、指定のデータ値を検出してDFLINTを出力します。コンパレータの一致検出で、ドライブパルス出力を停止させることができます。

- ・外部入力パルスは+(CW)方向のパルスでカウントアップ、-(CCW)方向のパルスでカウントダウンします。
- ・ドライブ出力パルスは-(CCW)方向のパルスでカウントアップ、+(CW)方向のパルスでカウントダウンします。
- ・カウンタの有効範囲は、-32,767 ~ +32,767 (H'8001 ~ H'7FFF) です。
負数の場合は、2の補数表現になります。

パルス偏差カウンタのパルス選択部

- ・パルス偏差カウンタのパルス選択機能はDFL COUNTER INITIALIZE1 コマンドで設定します。
- ・UC-766 以外の製品は、外部パルスの選択はできません。

パルス偏差カウンタとコンパレータの構成



- ・パルス偏差カウンタとコンパレータの機能は DFL COUNTER INITIALIZE1 コマンドで設定します。
- ・パルス偏差カウンタの現在値は DFL COUNTER PRESET コマンドで設定します。
- ・パルス偏差カウンタの現在値は DFL COUNTER READ コマンドで読み出せます。

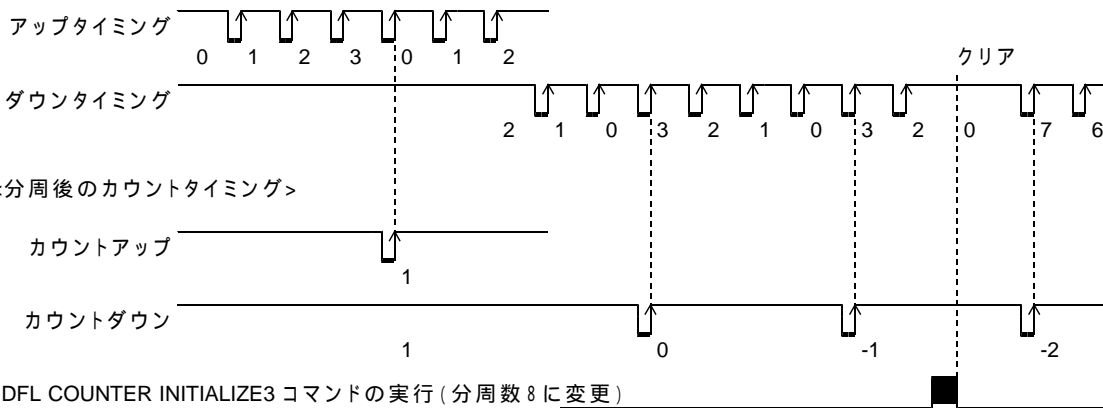
分周機能

パルス偏差カウンタでは COUNT PULSE SEL で選択したカウントパルスのカウントタイミングを分周することができます。

カウンタは分周したカウントタイミングでカウントアップ、またはカウントダウンします。

カウントタイミングを 4 分周する場合

<カウントパルスの入力>

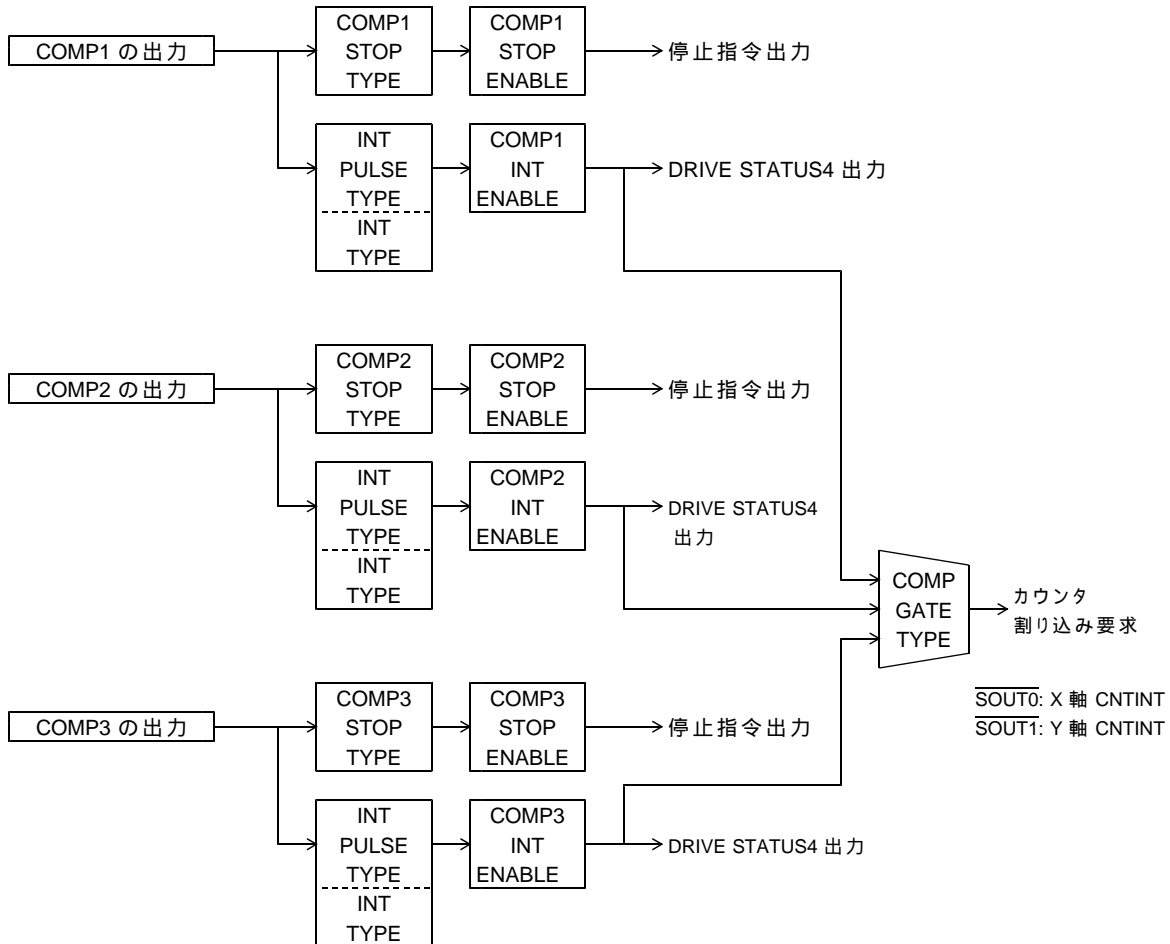


- ・パルス偏差カウンタの分周機能は DFL COUNTER INITIALIZE3 コマンドで設定します。
- ・DFL COUNTER INITIALIZE3 コマンドを実行すると分周中の分周カウント値をクリアします。

(6) コンパレータ機能

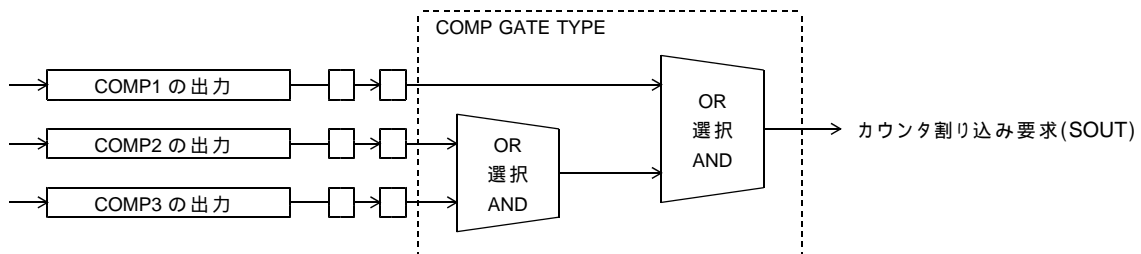
各カウンタには 3 個の専用コンパレータが付いており、カウンタ値と COMPARE REGISTER1, 2, 3 の値を比較して、検出条件が一致すると停止指令またはカウンタ割り込み要求(応用機能)を出力します。各カウンタ割り込み要求の出力状態は DRIVE STATUS4 PORT で確認できます。

コンパレータ出力の構成



- ・コンパレータ出力機能は各カウンタ COUNTER INITIALIZE1, 2 コマンドで設定します。
- ・UC-766 以外の製品は、カウンタ割り込み要求(SOUT)機能はありません。

COMP GATE TYPE の構成



- ・COMP GATE TYPE は各カウンタ COUNTER INITIALIZE1 コマンドで設定します。

コンパレータ出力仕様とクリア方法(INT TYPE)

コンパレータの出力仕様を以下の内から選択できます。

COMP1, 2, 3 の一致出力の出力仕様	クリア条件
一致出力をレベルラッチして出力する	検出条件が不一致のときに DRIVE STATUS4 PORT のリード終了でクリア
一致出力をエッジラッチして出力する	DRIVE STATUS4 PORT のリード終了でクリア
一致出力をそのままスルーで出力する	検出条件の不一致でクリア

- ・コンパレータ出力仕様とクリア方法(INT TYPE)は各カウンタ COUNTER INITIALIZE1 コマンドで設定します。
- ・レベルラッチ出力の場合は、検出条件が一致している間はクリアできません。
- ・スルー出力の場合は、最小出力幅が選択できます。

オートクリア機能

COMP1 の一致検出と同時に各カウンタの値を "0" にクリアします。

- ・オートクリア機能は各カウンタ COUNTER INITIALIZE1 コマンドで設定します。

自動加算機能

COMP1 の一致検出と同時に、COMP1 ADD データに設定されている値を COMPARE REGISTER1 に加算して、COMPARE REGISTER1 を再設定します。

$$\text{COMPARE REGISTER1} \leq \text{COMPARE REGISTER1} + \text{COMP1 ADD データ}$$

- ・自動加算機能は各カウンタ COUNTER INITIALIZE1 コマンドで設定します。

6-3. I/O 仕様

6-3-1. 汎用 I/O PORT

USB シリーズは、以下の汎用入出力を使用することができます。

別な USB ポートを占有したり、他の USB I/O 機器を用いずに、シンプルなモーションと I/O のシステムが構築できます。

- ・各ユニットに標準装備している汎用入出力各 2 点
- ・各ユニットに標準装備している拡張ポートから 1 台接続できる拡張 I/O ユニット(16/16 点または 32/32 点)
- ・UC-766 のドライバインターフェース用に装備している制御 I/O(汎用 I/O としても使えます。)

(1)コントローラの I/O PORT

次の関数により I/O PORT の書き込みと読み出しができます。

関数	書き込み	読み出し
ユニット関数	<ul style="list-style-type: none"> ・ユニット DRIVE COMMAND ・ I/O 書き込み関数 ・ユニット I/O PORT 書き込み関数 ・ユニット I/O PORT OR 書き込み関数 ・ユニット I/O PORT AND 書き込み関数 	<ul style="list-style-type: none"> ・ユニット STATUS1 ・ I/O 読み出し関数 ・ユニット STATUS1 ・ バルスカウンタ ・ I/O 読み出し関数 ・ユニット I/O PORT 読み出し関数
I/O PORT 関数	<ul style="list-style-type: none"> ・ I/O PORT 書き込み関数 ・ I/O PORT OR 書き込み関数 ・ I/O PORT AND 書き込み関数 	<ul style="list-style-type: none"> ・ I/O PORT 読み出し関数

・ユニット関数により、出力 PORT への書き込み、入力 PORT からの読み出しができます。

・ I/O PORT 関数により、出力 PORT への書き込み、出力・入力 PORT からの読み出しができます。

UC-766

I/O PORT		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
出力 PORT	汎用 I/O 出力 PORT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OUT1	OUT0
	制御 I/O 出力 0 PORT	0	0	A軸 ACLR	A軸 SON	0	0	Z軸 ACLR	Z軸 SON	0	0	Y軸 ACLR	Y軸 SON	0	0	X軸 ACLR	X軸 SON
	拡張 I/O 出力 0 PORT	OUT0F	OUT0E	OUT0D	OUT0C	OUT0B	OUT0A	OUT09	OUT08	OUT07	OUT06	OUT05	OUT04	OUT03	OUT02	OUT01	OUT00
	拡張 I/O 出力 1 PORT *1	OUT1F	OUT1E	OUT1D	OUT1C	OUT1B	OUT1A	OUT19	OUT18	OUT17	OUT16	OUT15	OUT14	OUT13	OUT12	OUT11	OUT10
入力 PORT	汎用 I/O 入力 PORT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IN1	IN0
	制御 I/O 入力 0 PORT	0	0	0	A軸 SRDY	0	0	0	Z軸 SRDY	0	0	0	Y軸 SRDY	0	0	0	X軸 SRDY
	拡張 I/O 入力 0 PORT	IN0F	IN0E	IN0D	IN0C	IN0B	IN0A	IN09	IN08	IN07	IN06	IN05	IN04	IN03	IN02	IN01	IN00
	拡張 I/O 入力 1 PORT *1	IN1F	IN1E	IN1D	IN1C	IN1B	IN1A	IN19	IN18	IN17	IN16	IN15	IN14	IN13	IN12	IN11	IN10

*1 CB-52/3232/MIL のとき

・書き込み/読み出し (0: ノットアクティブ、1: アクティブ)

・OR書き込み (0: 現在の出力状態を維持、1: アクティブ)

・AND書き込み (0: ノットアクティブ、1: 現在の出力状態を維持)

・拡張 I/O PORT のアクセスには、本体から CB-52/3232-MIL または CB-53/1616-MIL の接続が必要です。

(2)コントローラドライバの I/O PORT

次の関数により I/O PORT の書き込みと読み出しができます。

関数	書き込み	読み出し
ユニット関数	<ul style="list-style-type: none"> ・ユニット DRIVE COMMAND ・ I/O 書き込み関数 ・ユニット I/O PORT 書き込み関数 ・ユニット I/O PORT OR 書き込み関数 ・ユニット I/O PORT AND 書き込み関数 	<ul style="list-style-type: none"> ・ユニット STATUS1 ・ I/O 読み出し関数 ・ユニット STATUS1 ・ パルスカウンタ ・ I/O 読み出し関数 ・ユニット I/O PORT 読み出し関数
I/O PORT 関数	<ul style="list-style-type: none"> ・ I/O PORT 書き込み関数 ・ I/O PORT OR 書き込み関数 ・ I/O PORT AND 書き込み関数 	<ul style="list-style-type: none"> ・ I/O PORT 読み出し関数

・ユニット関数により、出力 PORT への書き込み、入力 PORT からの読み出しができます。

・ I/O PORT 関数により、出力 PORT への書き込み、出力 PORT ・入力 PORT からの読み出しができます。

UCD-7610v1/ADB5F30

I/O PORT		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
出力 PORT	汎用 I/O 出力 PORT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	制御 I/O 出力 PORT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	拡張 I/O 出力 0 PORT	OUT0F	OUT0E	OUT0D	OUT0C	OUT0B	OUT0A	OUT09	OUT08	OUT07	OUT06	OUT05	OUT04	OUT03	OUT02	OUT01	OUT00	
	拡張 I/O 出力 1 PORT *1	OUT1F	OUT1E	OUT1D	OUT1C	OUT1B	OUT1A	OUT19	OUT18	OUT17	OUT16	OUT15	OUT14	OUT13	OUT12	OUT11	OUT10	
入力 PORT	汎用 I/O 入力 PORT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	制御 I/O 入力 PORT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	拡張 I/O 入力 0 PORT	IN0F	IN0E	IN0D	IN0C	IN0B	IN0A	IN09	IN08	IN07	IN06	IN05	IN04	IN03	IN02	IN01	IN00	
	拡張 I/O 入力 1 PORT *1	IN1F	IN1E	IN1D	IN1C	IN1B	IN1A	IN19	IN18	IN17	IN16	IN15	IN14	IN13	IN12	IN11	IN10	

*1 CB-52/3232/MIL のとき

- ・書き込み/読み出し (0: ノットアクティブ、1: アクティブ)
- ・OR書き込み (0: 現在の出力状態を維持、1: アクティブ)
- ・AND書き込み (0: ノットアクティブ、1: 現在の出力状態を維持)

・拡張 I/O PORT のアクセスには、本体から CB-52/3232-MIL または CB-53/1616-MIL の接続が必要です。

UCD-7613v1/GDB5F40

I/O PORT		D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
出力 PORT	汎用 I/O 出力 PORT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	制御 I/O 出力 PORT	0	0	0	0	0	0	0	0	0	0	0	0	Y軸 CS	0	0	0
	拡張 I/O 出力 0 PORT	OUT0F	OUT0E	OUT0D	OUT0C	OUT0B	OUT0A	OUT09	OUT08	OUT07	OUT06	OUT05	OUT04	OUT03	OUT02	OUT01	OUT00
	拡張 I/O 出力 1 PORT *1	OUT1F	OUT1E	OUT1D	OUT1C	OUT1B	OUT1A	OUT19	OUT18	OUT17	OUT16	OUT15	OUT14	OUT13	OUT12	OUT11	OUT10
入力 PORT	汎用 I/O 入力 PORT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	制御 I/O 入力 PORT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	拡張 I/O 入力 0 PORT	IN0F	IN0E	IN0D	IN0C	IN0B	IN0A	IN09	IN08	IN07	IN06	IN05	IN04	IN03	IN02	IN01	IN00
	拡張 I/O 入力 1 PORT *1	IN1F	IN1E	IN1D	IN1C	IN1B	IN1A	IN19	IN18	IN17	IN16	IN15	IN14	IN13	IN12	IN11	IN10

*1 CB-52/3232/MIL のとき

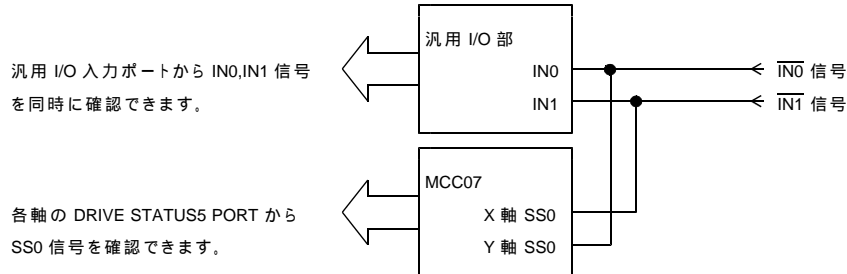
- ・書き込み/読み出し (0: ノットアクティブ、1: アクティブ)
- ・OR書き込み (0: 現在の出力状態を維持、1: アクティブ)
- ・AND書き込み (0: ノットアクティブ、1: 現在の出力状態を維持)

・拡張 I/O PORT のアクセスには、本体から CB-52/3232-MIL または CB-53/1616-MIL の接続が必要です。

6-3-2. その他の I/O PORT 機能

(1) コントローラ本体の入力 PORT

各ユニットに標準装備している汎用入力端子の $\overline{IN0}$ 信号および $\overline{IN1}$ 信号は、汎用 I/O 機能の他に、他の機能を割り付けることができます。



- ・ X 軸と Y 軸の SS0 信号は、SPEC INITIALIZE2 コマンドにより、各軸に次の機能を設定することができます。
 - 外部トリガ信号として使用する
 - 減速停止信号として使用する
 - 即時停止信号として使用する
- ・ COUNT LATCH SPEC SET コマンドにより、外部トリガ信号 (SS0 信号) を次の機能として使うことができます。
 - アドレスカウンタのラッチ信号
 - パルスカウンタのラッチ信号
 - パルス偏差カウンタのラッチ信号
 - パルス偏差カウンタをハードタイマとしたときのカウント開始信号
 各カウンタには、ラッチタイミングによるカウンタクリア機能があります。
 この SS0 信号のラッチ・クリア機能を使うことで、パソコンの OS や USB 通信などの遅れに影響を受けない、センサ入力 (INx 信号) を起点とした位置決め (自動停止) の応用が可能です。
- ・ SPEED CHANGE SPEC SET コマンド (応用機能)、INDEX CHANGE SPEC SET コマンド (応用機能) により、SS0 信号を外部トリガ信号として次の機能が使用できます。
 - UP/DOWN/CONST ドライブのドライブ CHANGE 変更動作点
 - SPEED CHANGE の変更動作点
 - INDEX CHANGE の変更動作点

(2) 出力 PORT

データ書き込みは、現在出力しているビットのアクティブレベルに影響を与えないような条件をアプリケーション側で管理しなければならない場合があります。

このような場合、AND 書き込みまたは OR 書き込みで設定すると、アプリケーション側では現在の出力状態を気にせず、変化したいビットだけ指定しながらデータを設定することができます。

使い分けについては、下記のアンダーラインをご覧ください。

出力信号の AND 書き込み機能

現在出力ポートで出力しているデータを AND データで書き込み、出力を ON/OFF することができます。

- ・あるビットの出力を OFF にする場合、AND で 0 を書き込みます。
- ・あるビットの出力を変化させない場合、AND で 1 を書き込みます。

前データ	AND データ	出力データ	備考
0	0	0	AND データ 0 は、前回の出力データ に関係なく、出力を OFF にします。
1	0	0	
0	1	0	AND データ 1 は、前回の出力データ に関係なく、は変化しません。
1	1	1	

出力信号の OR 書き込み機能

現在出力ポートで出力しているデータを OR データで書き込み、出力を ON/OFF することができます。

- ・あるビットの出力を変化させない場合、OR で 0 を書き込みます。
- ・あるビットの出力を ON にする場合、OR で 1 を書き込みます。

前データ	OR データ	出力データ	備考
0	0	0	OR データ 0 は、前回の出力データ に関係なく、は変化しません。
1	0	1	
0	1	1	OR データ 1 は、前回の出力データ に関係なく、出力を ON にします。
1	1	1	

7. 付録

7-1. 初期仕様一覧

(1) 基本設定

項目	初期仕様	対応関数/コマンド
パルス出力機能		
パルス出力方式	独立方向出力	SPEC INITIALIZE1 コマンド
パルス出力マスク	マスクしない	
LIMIT 停止機能		
CWLM 信号入力機能	+方向の LIMIT 即時停止入力	SPEC INITIALIZE2 コマンド
CCWLM 信号入力機能	-方向の LIMIT 即時停止入力	
多用途センサ機能(応用)		
多用途センサ(SS0)機能	汎用入力、各種機能のトリガ入力	SPEC INITIALIZE2 コマンド
サーボ対応機能		
DRST 信号出力機能	汎用出力	SPEC INITIALIZE3 コマンド
DEND/PO 信号入力機能	汎用入力	
DALM 信号入力機能	汎用入力	
自動減速停止機能(応用)		
DOWN PULSE マスク	マスクしない	SPEC INITIALIZE3 コマンド
エラー出力要因		
COMMAND ERROR	マスクしない	ERROR STATUS MASK コマンド
COMREG CLR ERROR	マスクしない	
INC INDEX ERROR	マスクしない(変更できません)	
ABS INDEX ERROR	マスクしない(変更できません)	
INDEX CHANGE ERROR	マスクしない(変更できません)	
CHANGE CLR ERROR	マスクしない	
CPP ST ERROR	マスクしない	
EXT PULSE ERROR	マスクしない	
FSEND ERROR	マスクしない	
LSEND ERROR	マスクする	
SSEND ERROR	マスクする	
ADDRESS OVF ERROR	マスクする	
PULSE OVF ERROR	マスクする	
DALM ERROR	マスクする	
FSSTOP ERROR	マスクする	
アドレスカウンタ		
カウントパルス	発生パルス	ADDRESS COUNTER INITIALIZE1 コマンド
エンコーダ入力パルスカウント方法	1 週倍	
外部パルス出力のアクティブ幅	1 μ s	
ADRINT 出力仕様	レベルラッチ	
ADRINT スルー時最小出力幅	200ns	
COMP 合成出力選択	論理和(OR)	
COMP1 クリア機能	クリアしない	
COMP1 自動加算機能	加算再設定しない	
COMP1,2,3 INT ENABLE	出力しない	
COMP1,2,3 STOP ENABLE	停止しない	
COMP1,2,3 STOP TYPE	即時停止	ADDRESS COUNTER INITIALIZE2 コマンド
COMP2,3 検出条件	= (一致)	
カウンタ値	H'0000_0000	
COMP1 自動加算値	H'0000_0000	ADDRESS COUNTER PRESET コマンド ADRINT COMPARE REGISTER1,2,3 SET ADRINT COMP ADD DATA SET コマンド
COMP1,2,3 COMPARE REGISTER 値(1,2,3)	H'8000_0000	
COMP1 自動加算値	H'0000_0000	
パルスカウンタ		
カウントパルス	出力パルス	PULSE COUNTER INITIALIZE1 コマンド
エンコーダ入力パルスカウント方法	1 週倍	
CNTINT 出力仕様	レベルラッチ	
CNTINT スルー時最小出力幅	200ns	
COMP 合成出力選択	論理和(OR)	
COMP1 クリア機能	クリアしない	
COMP1 自動加算機能	加算再設定しない	

(2) 基本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(200 μs)	SPEED・RATE 関数
加減速パラメータ		
最高速度	3,000Hz	SPEED・RATE 関数
開始速度	300Hz	
終了速度	300Hz	
加速カーブ S 字変速領域 (SUAREA)	変速領域なし	
減速カーブ S 字変速領域 (SDAREA)	変速領域なし	
加速時定数 (URATE)	100ms/kHz	
減速時定数 (DRATE)	100ms/kHz	
速度倍率 (RESOL)	1 (No.3)	
JOG パラメータ		
JOG パルス速度	300Hz	JSPD SET コマンド
JOG パルス数	1PULSE	JOG PULSE SET コマンド
ORIGIN パラメータ		
ORG.START DIR	-(CCW)方向、ORG 信号と、ZORG 信号の論理和(OR)	ORIGIN SPEC SET 関数
PULSE SENSOR TYPE	機械原点信号のエッジを検出して工程を終了する	
SENSOR ERROR TYPE	エラー終了する	
ERROR PULSE ERROR ENABLE	ERROR PULSE ERROR 検出機能 無効	
AUTO DRST ENABLE	DRST 信号を出力しない	
SCAN MARGIN ENABLE	SCAN 工程時に MARGIN PULSE を入れない	
ORG SIGNAL TYPE	ORG 信号	
NORG SIGNAL TYPE	NORG 信号	
MARGIN PULSE	5 パルス	ORIGIN MARGIN PULSE SET 関数
LIMIT DELAY TIME	300ms	ORIGIN DELAY SET 関数
SCAN DELAY TIME	50ms	
PULSE DELAY TIME	20ms	
CSCAN ERROR PULSE	2,147,483,647 パルス	ORIGIN ERROR PULSE SET 関数
PULSE ERROR PULSE	2,147,483,647 パルス	
OFFSET PULSE	100 パルス	ORIGIN OFFSET PULSE SET 関数
PRESET PULSE	0 パルス	ORIGIN PRESET PULSE SET 関数

7-2.関数一覧

種別	名称	記号	ページ
	説明		
体 型 構 造 基 礎	RESULT構造体	MC07_S_RESULT	29
	関数を実行した結果を格納する。		
	コマンドデータ構造体	MC07_S_COMMAND_DATA	31
	DRIVE COMMAND PORT、DRIVE DATA1 PORT、DRIVE DATA2 PORTに書き込むデータを格納する。		
ステータスデータ構造体	MC07_S_STATUS_DATA	32	
	DRIVE STATUS1 PORT、DRIVE DATA1 PORT、DRIVE DATA2 PORTから読み出した内容を格納する。		
ユ ニ ツ ク ロ ー ズ	ユニット情報構造体	MC07_S_UNIT_INFO	33
	パソコンに接続されている全ユニットのタイプを格納する。		
	環境設定関数	MC07_Environment	34
	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。		
ユ ニ ツ ク ロ ー ズ	ユニット情報読み出し関数	MC07_ReadUnitInfo	35
	環境設定されている全ユニットのタイプを読み出す。		
ト ク ロ ー ズ	ユニットオープン関数	MC07_UOpen	36
	ユニットをオープンし、引数 $phUnit$ の変数にユニットハンドルを格納する。		
	ユニットクローズ関数	MC07_UClose	37
	指定されたユニットをクローズする。		
	ユニット動作エラークリア関数	MC07_UClrError	38
	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。		
	拡張ユニット通信設定関数	MC07_UWExUnitCommMode	39
	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。		
	拡張ユニット通信制御関数	MC07_UWExUnitCommControl	40
	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。		
ユ ニ ツ ク ロ ー ズ	拡張ユニット通信ステータス読み出し関数	MC07_URExUnitCommStatus	41
	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。		
ユ ニ ツ ク ロ ー ズ	拡張ユニット通信設定読み出し関数	MC07_URExUnitCommMode	42
	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。		
ス テ タ ス	ユニットステータス構造体	MC07_S_UNIT_STATUS	43
	ユニットのステータスの内容を格納する。		
ユ ニ ツ ク ロ ー ズ	ユニットコマンド構造体	MC07_S_UNIT_COMMAND	44
	ユニットのコマンドを格納する。		
ユ ニ ツ ク ロ ー ズ	入力PORT構造体	MC07_S_IN_PORT	45
	ユニットの入力PORTから読み出された内容を格納する。		
ユ ニ ツ ク ロ ー ズ	出力PORT構造体	MC07_S_OUT_PORT	46
	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。		
ユ ニ ツ ク ロ ー ズ	ユニットDRIVE COMMAND・I/O書き込み関数	MC07_UWDriveIo	47
	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。		
ユ ニ ツ ク ロ ー ズ	ユニットDRIVE COMMAND書き込み/読み出し関数	MC07_UWRDrive	48
	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。		
ユ ニ ツ ク ロ ー ズ	ユニットSTATUS1・I/O読み出し関数	MC07_URStatus1Io	49
	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。		
ユ ニ ツ ク ロ ー ズ	ユニットSTATUS1・パルスカウンタ・I/O読み出し関数	MC07_URStatus1PcntIo	51
	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。		
ユ ニ ツ ク ロ ー ズ	ユニットI/O PORT書き込み関数	MC07_UPortOUT	53
	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。		
ユ ニ ツ ク ロ ー ズ	ユニットI/O PORT OR書き込み関数	MC07_UPortOrOUT	54
	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。		
ユ ニ ツ ク ロ ー ズ	ユニットI/O PORT AND書き込み関数	MC07_UPortAndOUT	55
	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。		
ユ ニ ツ ク ロ ー ズ	ユニットI/O PORT読み出し関数	MC07_UPortIn	56
	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。		

種別	名称	記号	ページ	
K I P ト	デバイスオープン関数 指定ユニット番号、軸でデバイスオープンし、引数 <i>phDev</i> の変数にデバイスハンドルを格納する。	MC07_BOpen	57	
	デバイスクローズ関数 指定されたデバイスをクローズする。	MC07_BClose	58	
	動作エラークリア関数 指定されたデバイスの動作エラーをクリアする。	MC07_ClrError	59	
	DRIVE COMMAND 32ビット書き込み関数 指定デバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORTにデータを書き込んだ後コマンドを書き込む。	MC07_LWDrive	62	
	DRIVE COMMAND PORT書き込み関数 指定デバイスのDRIVE COMMAND PORTにコマンドコードを書き込む。	MC07_BWDriveCommand	63	
	DRIVE STATUS1 PORT読み出し関数 指定デバイスのDRIVE STATUS1 PORTを読み出す。	MC07_BRStatus1	64	
	DRIVE STATUS2 PORT読み出し関数 指定デバイスのDRIVE STATUS2 PORTを読み出す。	MC07_BRStatus2	68	
	DRIVE STATUS3 PORT読み出し関数 指定デバイスのDRIVE STATUS3 PORTを読み出す。	MC07_BRStatus3	70	
	DRIVE STATUS4 PORT読み出し関数 指定デバイスのDRIVE STATUS4 PORTを読み出す。	MC07_BRStatus4	71	
	DRIVE STATUS5 PORT読み出し関数 指定デバイスのDRIVE STATUS5 PORTを読み出す。	MC07_BRStatus5	73	
	DRIVE STATUSバッファ読み出し関数 指定デバイスのDRIVE STATUS1、STATUS2、STATUS3、STATUS4、STATUS5 PORT、ORIGIN STATUSを読み出す	MC07_BRStatusBuf	76	
	DRIVE COMMAND 32ビット書き込み/読み出し関数 指定デバイスのDRIVE DATA、COMMAND PORTにデータ、コマンドを書き込み、DRIVE DATA PORTを読み出す。	MC07_LWRDrive	77	
	W A I T	READY WAIT関数 指定デバイスがREADY (STATUS1 BUSY BIT=0)まで待機し、最大待ち時間を超えるとエラー終了する。	MC07_BWaitDriveCommand	78
		WAIT状態読み出し関数 指定デバイスのWAIT状態を返す。	MC07_BIsWait	79
		WAIT中止関数 指定デバイスのREADY WAIT関数またはCOMREG NOT FULL WAIT関数の実行を中止する。	MC07_BBreakWait	80
	S P E E D	SPEED・RATE構造体 SPEED・RATEセット関数で使用する。	MC07_S_SPEED_RATE	81
SPEED・RATEセット関数 指定のRESOL No.とSPEED・RATE構造体を元にSPEEDパラメータ設定、加減速時定数(RATE)設定を実行する。		MC07_SetSpeedRate	83	
SPEED・RATE読み出し関数 指定デバイスからSPEEDパラメータ、加減速時定数設定を読み出し、SPEED・RATE構造体に格納する。		MC07_ReadSpeedRate	84	
P O S I T I O N 関 数	POSITION構造体 X・Y座標を指定するときに使用する。	MC07_S_XY_POSITION	85	
	2軸相対アドレス直線補間ドライブ関数 相対アドレスで指定された目的地まで任意2軸直線補間ドライブを行う。	MC07_IncStrCp	86	
	2軸相対アドレス円弧補間ドライブ関数 相対アドレスで指定された目的地まで任意2軸円弧補間ドライブを行う。	MC07_IncCirCp	88	
	メインチップ2軸相対アドレス直線補間ドライブ関数 相対アドレスで指定された目的地までメインチップ2軸直線補間ドライブを行う。	MC07_McIncStrCp	90	
	メインチップ2軸相対アドレス円弧補間ドライブ関数 相対アドレスで指定された目的地までメインチップ2軸円弧補間ドライブを行う。	MC07_McIncCirCp	92	
	円の中心点ゲット関数 円弧の通過点相対アドレス、目的地相対アドレスを元に中心点相対アドレス、回転方向を求める。	MC07_GetCirCenterPosition	94	
	相対アドレス変換関数 指定された絶対アドレスを相対アドレス(絶対アドレス - 現在位置)に変換する。	MC07_IncFromAbs	95	

種別	名称	記号	ページ
O R I G I N	ORIGINドライブパラメータ構造体	MC07_S_ORG_PARAM	96
	ORIGINドライブパラメータ読み出し関数で読み出した内容を格納する。		
	ORIGINドライブステータス読み出し関数	MC07_ReadOrgStatus	97
	ORIGIN STATUSの内容を読み出す。		
	ORIGIN SPEC SET関数	MC07_SetOrgSpec	99
	ORIGINドライブの動作仕様を設定する。		
	ORIGIN MARGIN PULSE SET関数	MC07_SetOrgMarginPulse	102
	ORIGINドライブの機械原点信号検出後のMARGINパルス数を設定する。		
	ORIGIN DELAY SET関数	MC07_SetOrgDelay	103
	ORIGINドライブの各工程後で挿入するDELAYを設定する。		
	ORIGIN ERROR PULSE SET関数	MC07_SetOrgErrorPulse	104
	CONSTANT SCAN工程時および1PULSE送り工程時にエラー判定する各最大パルス数を設定する。		
	ORIGIN OFFSET PULSE SET関数	MC07_SetOrgOffsetPulse	105
	機械原点近傍アドレスのOFFSETパルス数を設定する。		
	ORIGIN PRESET PULSE SET関数	MC07_SetOrgPresetPulse	106
機械原点検出終了後に実行するORIGINドライブのPRESETパルスを設定する。			
ORIGINドライブパラメータ読み出し関数	MC07_ReadOrgParam	107	
設定されたORIGINドライブパラメータを読み出し、ORIGINドライブパラメータ構造体に格納する。			
ORIGIN FLAG RESET関数	MC07_ResetOrgFlag	108	
ORIGIN FLAGをRESETする。			
ORIGINドライブ関数	MC07_Org	109	
指定された機械原点の型式に従いORIGINドライブを行う。			
O P E N	I/O PORTオープン関数	MC07_BPortOpen	110
	拡張・汎用・制御I/O PORTをオープンし、引数 $phPort$ の変数にPORTハンドルを格納する。		
	I/O PORTクローズ関数	MC07_BPortClose	111
	指定された拡張・汎用・制御I/O PORTをクローズする。		
	I/O PORT書き込み関数	MC07_BPortOut	112
	指定された拡張・汎用・制御・GユニットのI/O PORTにデータを書き込む。		
	I/O PORT OR書き込み関数	MC07_BPortOrOut	113
指定された拡張・汎用・制御・GユニットのI/O PORTにORデータを書き込む。			
I/O PORT AND書き込み関数	MC07_BPortAndOut	114	
指定された拡張・汎用・制御・GユニットのI/O PORTにANDデータを書き込む。			
I/O PORT読み出し関数	MC07_BPortIn	115	
指定された拡張・汎用・制御・GユニットのI/O PORTのデータを読み出す。			

7-3. ドライブコマンド一覧

(1) 汎用コマンド

種別	コマンド名	コマンドコード	ページ
汎用 コマンド	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	116
	SPEC_INITIALIZE2 CWLM, CCWLM, SSOの設定	H'02	117
	SPEC_INITIALIZE3 DRST, DEND/PO, DALMの設定絶対アドレスINDEXドライブの実行	H'03	119
	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
	+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	H'15	128

(2) 特殊コマンド

種別	コマンド名	コマンドコード	ページ
エ ハ ト 口 差 埠	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
	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タ初期値の設定	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	
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タ初期値の設定	H'A0	162	
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	163	
DFLINT COMPARE REGISTER2 SET DFLINTのコンペアレジスタ2の設定	H'A9	163	
DFLINT COMPARE REGISTER3 SET DFLINTのコンペアレジスタ3の設定	H'AA	163	
DFLINT COMP1 ADD DATA SET DFLINTのCOMP1 ADDデータの設定	H'AC	164	
ERROR STATUS READ ERROR STATUSの読み出し	H'D1	133	
MCC SPEED READ ドライブパルス速度の読み出し	H'D4	135	
MCC SET DATA READ 設定データの読み出し	H'D5	136	

取扱説明書

(USB シリーズ MCC07 ユニット編)

種別	コマンド名	コマンドコード	ページ
	説明		
レ ハ ト 口 添 装	ADDRESS COUNTER READ アドレスカウンタの読み出し	H'D8	165
	PULSE COUNTER READ パルスカウンタの読み出し	H'D9	165
	DFL COUNTER READ パルス偏差カウンタの読み出し	H'DA	165
	ERROR STATUS MASK ERRORに出力するERROR STATUSのマスク	H'E5	132
	SIGNAL OUT 汎用出力信号の操作	H'FC	130
	DRST OUT DRSTに10ms間のアクティブレベルを出力	H'FD	131
	SLOW STOP 減速停止の実行	H'FE	129
	FAST STOP 即時停止の実行	H'FF	129

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いただきます。

- (1) お客様の不適當な取り扱い、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) お客様の改造、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、保証外とさせていただきます。

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>