STEPPING & SERVO MOTOR CONTROLLER'S OPTION

# MPL-32/PCIW32
# MPL-33/PCIW64
# 取扱説明書
# （設計者用）

（デバイスドライバ PCI MCC06 ボード編）

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。**

MN0096

# 目次

# 1. 概要

MPL-32/PCIW32及びMPL-33/PCIW64は、DOS/VパソコンのWindows上でPCI BUSを介して、弊社製ステッピング＆サーボモータコントローラボードを動作させるためのDLLベースのドライバ関数です。各関数は、次に示すBOARD CONTROLLER上のMCC06 PORT、HENSA PORT、HARD CONFIGURATION PORTのアクセス(読み出し/書き込み)及び、割り込み発生による処理を行う為のものです。BOARD CONTROLLER上のPORT、割り込み要求信号(INT0、INT1、INT2)については、各BOARD CONTROLLERの取扱説明書を御覧ください。

(1)C-V870

| MCC06 PORT | HENSA PORT | HARD CONFIGURATION PORT |
|---|---|---|
| DRIVE COMMAND PORT | HENSA COMMAND PORT | HARD CONFIG COMMAND PORT |
| DRIVE DATA1 PORT | HENSA DATA1 PORT | HARD CONFIG DATA1 PORT |
| DRIVE DATA2 PORT | HENSA DATA2 PORT | HARD CONFIG DATA2 PORT |
| DRIVE DATA3 PORT | HENSA STATUS1 PORT | HARD CONFIG DATA3 PORT |
| STATUS1 PORT | | SIGNAL STATUS PORT |
| STATUS2 PORT | | |
| STATUS3 PORT | | |
| STATUS4 PORT | | |
| STATUS5 PORT | | |
| COUNTER COMMAND PORT | | |
| COUNTER DATA1 PORT | | |
| COUNTER DATA2 PORT | | |
| COUNTER DATA3 PORT | | |

(2)C-V872

| MCC06 PORT | HENSA PORT | HARD CONFIGURATION PORT |
|---|---|---|
| DRIVE COMMAND PORT | HENSA COMMAND PORT | HARD CONFIG COMMAND PORT |
| DRIVE DATA1 PORT | HENSA DATA1 PORT | HARD CONFIG DATA1 PORT |
| DRIVE DATA2 PORT | HENSA DATA2 PORT | HARD CONFIG DATA2 PORT |
| DRIVE DATA3 PORT | HENSA STATUS1 PORT | HARD CONFIG DATA3 PORT |
| STATUS1 PORT | | SIGNAL STATUS1 PORT |
| STATUS2 PORT | | SIGNAL STATUS2 PORT |
| STATUS3 PORT | | |
| STATUS4 PORT | | |
| STATUS5 PORT | | |
| COUNTER COMMAND PORT | | |
| COUNTER DATA1 PORT | | |
| COUNTER DATA2 PORT | | |
| COUNTER DATA3 PORT | | |

(3) MPL とサポート言語の相関関係

| MPL | サポート言語 | | | | | | |
|---|---|---|---|---|---|---|---|
| | C++ Builder | C#.NET | Deliphi | Visual Basic | Visual Basic.NET | Visual C++ | Visual C++.NET |
| MPL-32/PCIW32 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| MPL-33/PCIW64 | × | ○ | × | × | ○ | × | ○ |

# 2. ボードNo.の設定

当デバイスドライバを使用すると、最大10枚のBOARD CONTROLLERを制御することが可能です。複数のBOARD CONTROLLERをご使用の場合、各BOARD CONTROLLER上のボードNo.(S1 ロータリースイッチ)は、必ず異なる設定にしてください。

# 3. デバイスドライバの処理

## 3-1. OS起動時、デバイスドライバ インストール時

デバイスドライバは、OS起動時又は、デバイスドライバ インストール時にデバイスドライバがインストールされたボード上の全MCC06に対して次に示すCOMMANDを実行します。

(1)HARD INITIALIZE1 COMMAND
　OUT0 TYPE3_0　　：『ADRINT出力』
　SIGNAL OUTA TYPE3_0：『CNTINT出力』
　SIGNAL OUTB TYPE3_0：『DFLINT出力』
　OUT3 TYPE3_0　　：『機能なし』　注.ボードの初期値と異なる設定となります。
(2)INT FACTOR MASK COMMAND
　全INT FACTORがマスクされます。

## 3-2. ユーザアプリケーション起動時

デバイスドライバは、ユーザアプリケーション起動時にデバイスドライバがインストールされているボード上の全MCC06に対して次に示すCOMMANDを実行します。尚、ユーザアプリケーションが複数起動された場合、最初に起動されたユーザアプリケーション起動時のみにCOMMANDが実行され、そのあとのユーザアプリケーション起動時には、COMMANDは、実行されません。

(1)HARD INITIALIZE2 COMMAND
　GPIO0 TYPE3_0　　：『汎用入力』
　GPIO1 TYPE3_0　　：『汎用入力』
(2)HARD INITIALIZE1 COMMAND
　OUT0 TYPE3_0　　：『ADRINT出力』
　SIGNAL OUTA TYPE3_0：『CNTINT出力』
　SIGNAL OUTB TYPE3_0：『DFLINT出力』
　OUT3 TYPE3_0　　：『機能なし』　注.ボードの初期値と異なる設定となります。
(3)INT FACTOR MASK COMMAND
　全INT FACTORがマスクされます。
(4)INT FACTOR CLR COMMAND
　全INT FACTORがクリアされます。

## 3-3. ユーザアプリケーション終了時

デバイスドライバは、ユーザアプリケーション終了時にデバイスドライバがインストールされているボード上の全MCC06に対して次に示すCOMMANDを実行します。尚、ユーザアプリケーションが複数起動された場合、最後に終了したユーザアプリケーション終了時のみにCOMMANDが実行され、そのまえのユーザアプリケーション終了時には、COMMANDは、実行されません。

(1)HARD INITIALIZE2 COMMAND
　GPIO0 TYPE3_0　　：『汎用入力』
　GPIO1 TYPE3_0　　：『汎用入力』
(2)HARD INITIALIZE1 COMMAND
　OUT0 TYPE3_0　　：『ADRINT出力』
　SIGNAL OUTA TYPE3_0：『CNTINT出力』
　SIGNAL OUTB TYPE3_0：『DFLINT出力』
　OUT3 TYPE3_0　　：『機能なし』　注.ボードの初期値と異なる設定となります。
(3)INT FACTOR MASK COMMAND
　全INT FACTORがマスクされます。
(4)INT FACTOR CLR COMMAND
　全INT FACTORがクリアされます。

### 3-4.ユーザアプリケーションからのMCC06 COMMANDの書き込み時

ユーザアプリケーションのミスによるトラブルを防止する為にユーザアプリケーションがMCC06に書き込もうとしたCOMMANDに対して次のチェックを行っています。

(1)ユーザアプリケーションがHARD INITIALIZE2 COMMAND又は、HARD INITIALIZE3 COMMANDを書き込もうとした場合、COMMAND無効になり、MCC06に書き込みは行われません。

(2)ユーザアプリケーションがHARD INITIALIZE1 COMMANDを書き込もうとした場合、OUT3 TYPE3_0は、無条件に『機能なし』に補正されMCC06に書き込まれます。尚、HARD INITIALIZE1 COMMANDの他の設定は、ユーザアプリケーションが設定した内容が書き込まれます。

(3)ユーザアプリケーションがSPEC INITIALIZE2 COMMANDを書き込もうとした場合、RDYINT TYPE1_0が『STATUS1 PORTのBUSY=0の立ち下がりエッジ検出でハイレベルにする』に設定されていた時、『出力しない』に補正されて、MCC06に書き込まれます。尚、SPEC INITIALIZE2 COMMANDの他の設定は、ユーザアプリケーションが設定した内容が書き込まれます。

## 4.デバイスのオープンとクローズ

ユーザアプリケーションは、MCC06 PORT、HENSA PORT、HARD CONFIGURATION PORTにアクセス前にボードNo.と軸を指定してデバイスをオープン(デバイスオープン関数)し、デバイスハンドルを受け取ります。以後、各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。クローズが行われていないと、以後正常に動作しません。

# 5.割り込み

割り込みの通知方法は、次の3つが使用できます。

| 通知方法 | 説明 |
|---|---|
| コールバック | 割り込みが発生したときに、設定された関数をコールバックします。 |
| メッセージ | 割り込みが発生したときに、設定されたウィンドウのウィンドウプロシージャにメッセージをポストします。 |
| イベント | 割り込みが発生したときに、設定されたイベントオブジェクトがシグナル状態になります。 |

## 5-1.割り込み処理の手順

ユーザアプリケーションで割り込みを使用する場合、次の手順で行ってください。

(1)DRIVE終了割り込み(RDYINT)使用の場合

①SPEC INITIALIZE2 COMMANDでRDYINT TYPE0_1を『STATUS1 PORTのDRVEND=1の立ち上がりエッジ検出でハイレベルにする』に設定します。

注.RDYINT TYPE0_1を『STATUS1 PORTのBUSY=0の立ち下がりエッジ検出でハイレベルにする』に設定することは、出来ません。設定された場合、デバイスドライバでチェックが行われ、『出力しない』に補正されて、設定されます。

②割り込み通知方法を設定します。

コールバック使用の場合 ･･･ 割り込みコールバック設定関数
メッセージ使用の場合 ･･･ 割り込みメッセージ設定関数
イベント使用の場合 ･･･ 割り込みイベント設定関数

③割り込みオープン関数で割り込みをオープンします。

④割り込みが検出されると通知方法に従い、通知処理が実行されます。

注. 通知が行われた場合であっても、MCC06 STATUS1 PORT BUSY=0が保証されていない為、ユーザアプリケーションでBUSY=0を確認する必要がある場合、STATUS1 PORTを読み出して、BUSY=0を確認してください。

⑤ユーザアプリケーション終了時には、割り込みクローズ関数で割り込みをクローズした後、デバイスをクローズし、ユーザアプリケーションを終了してください。

(2)カウンタ割り込み(ADRINT/CNTINT/DFLINT/SPDINT)使用の場合

①カウンタのCOUNTER COMPARE REGISTERの設定を行います。

②カウンタの機能設定を行います。

ADRINT使用の場合 ･･･ ADDRESS COUNTER INITIALIZE1 COMMAND
注.ADRINT TYPE0_1は、必ず次の設定としてください。
COMP1,2,3の一致出力の出力仕様 ： 『一致出力をエッヂラッチして出力』
クリア条件 ： 『INT FACTOR CLR COMMANDのINT2 ADRINT=1の実行でクリア』
ADDRESS COUNTER INITIALIZE2 COMMAND
ADDRESS COUNTER INITIALIZE3 COMMAND

CNTINT使用の場合 ･･･ PULSE COUNTER INITIALIZE1 COMMAND
注.CNTINT TYPE0_1は、必ず次の設定としてください。
COMP1,2,3の一致出力の出力仕様 ： 『一致出力をエッヂラッチして出力』
クリア条件 ： 『INT FACTOR CLR COMMANDのINT2 CNTINT=1の実行でクリア』
PULSE COUNTER INITIALIZE2 COMMAND
PULSE COUNTER INITIALIZE3 COMMAND

DFLINT使用の場合 ･･･ DFL COUNTER INITIALIZE1 COMMAND
注.DFLINT TYPE0_1は、必ず次の設定としてください。
COMP1,2,3の一致出力の出力仕様 ： 『一致出力をエッヂラッチして出力』
クリア条件 ： 『INT FACTOR CLR COMMANDのINT2 DFLINT=1の実行でクリア』
DFL COUNTER INITIALIZE2 COMMAND
DFL COUNTER INITIALIZE3 COMMAND

SPDINT使用の場合 ･･･ SPEED COUNTER INITIALIZE1 COMMAND
注.SPDINT TYPE0_1は、必ず次の設定としてください。
COMP1,2,3の一致出力の出力仕様 ： 『一致出力をエッヂラッチして出力』
クリア条件 ： 『INT FACTOR CLR COMMANDのINT2 SPDINT=1の実行でクリア』
SPEED COUNTER INITIALIZE2 COMMAND
SPEED COUNTER INITIALIZE3 COMMAND

③COUNTER COMP MASK COMMANDでコンパレータ出力のマスクを解除します。
④割り込み通知方法を設定します。
　コールバック使用の場合 ･･･ 割り込みコールバック設定関数
　メッセージ使用の場合　 ･･･ 割り込みメッセージ設定関数
　イベント使用の場合　　 ･･･ 割り込みイベント設定関数
⑤割り込みオープン関数で割り込みをオープンします。
⑥割り込みがデバイスドライバ内部で検出されると通知方法に従い、通知処理が実行されます。
⑦ユーザアプリケーション終了時には、割り込みクローズ関数で割り込みをクローズした後、
　デバイスをクローズし、ユーザアプリケーションを終了してください。

(3)SS1/SSO/DALM/MAN割り込み使用の場合
①割り込み通知方法を設定します。
　コールバック使用の場合 ･･･ 割り込みコールバック設定関数
　メッセージ使用の場合　 ･･･ 割り込みメッセージ設定関数
　イベント使用の場合　　 ･･･ 割り込みイベント設定関数
②割り込みオープン関数で割り込みをオープンします。
③割り込みがデバイスドライバ内部で検出されると通知方法に従い、通知処理が実行されます。
④ユーザアプリケーション終了時には、割り込みクローズ関数で割り込みをクローズした後、
　デバイスをクローズし、ユーザアプリケーションを終了してください。

# 6. 関数説明

## 6-1. 説明の見方

### ○○○○構造体 ← 構造体の名称

○○○○ ← 構造体が使用される関数が対応するモータコントローラボードの名称

**説　明**

・・・・・・・・・・・・・・・・・・・・・・・・・・・・ → 構造体の説明

**書　式**

C言語 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ → Ｃ言語で、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 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ → C#.NETで、構造体を使用するときの定義

Delphi ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ → Delphi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

### ○○○○関数 ← 関数の名称

○○○○ ← 関数が対応するモータコントローラボードの名称

**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

Delphi ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ → Delphiで、関数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

## 6-2. 構造体と関数

● 構造体一覧

| 構造体名 | 説明 |
|---|---|
| RESULT構造体 | 関数を実行した結果を格納 |
| データ構造体 | デバイスに一括でアクセスするためのデータを格納 |

● 関数一覧

| 関数名 | 機能 |
|---|---|
| デバイスオープン関数 | デバイスのオープン |
| デバイスクローズ関数 | デバイスのクローズ |
| DRIVE COMMAND一括書き込み関数 | DRIVE COMMAND, DATA1_3 PORTに一括書き込み |
| DRIVE DATA一括書き込み関数 | DRIVE DATA1_3 PORTに一括書き込み |
| DRIVE COMMAND PORT書き込み関数 | DRIVE COMMAND PORTに書き込み |
| DRIVE DATA1 PORT書き込み関数 | DRIVE DATA1 PORTに書き込み |
| DRIVE DATA2 PORT書き込み関数 | DRIVE DATA2 PORTに書き込み |
| DRIVE DATA3 PORT書き込み関数 | DRIVE DATA3 PORTに書き込み |
| STATUS1 PORT読み出し関数 | STATUS1 PORTの読み出し |
| STATUS2 PORT読み出し関数 | STATUS2 PORTの読み出し |
| STATUS3 PORT読み出し関数 | STATUS3 PORTの読み出し |
| STATUS4 PORT読み出し関数 | STATUS4 PORTの読み出し |
| STATUS5 PORT読み出し関数 | STATUS5 PORTの読み出し |
| DRIVE DATA一括読み出し関数 | DRIVE DATA1_3 PORTの一括読み出し |
| DRIVE DATA1 PORT読み出し関数 | DRIVE DATA1 PORTの読み出し |
| DRIVE DATA2 PORT読み出し関数 | DRIVE DATA2 PORTの読み出し |
| DRIVE DATA3 PORT読み出し関数 | DRIVE DATA3 PORTの読み出し |
| READY WAIT関数 | パルスジェネレータがREADYになるまで待機 |
| READY WAIT･HENSA READY WAIT状態読み出し関数 | READY WAIT･HENSA READY WAIT関数の状態の読み出し |
| READY WAIT･HENSA READY WAIT中止関数 | READY WAIT･HENSA READY WAIT関数のREADY待ちを中止 |
| COUNTER COMMAND一括書き込み関数 | COUNTER COMMAND, DATA1_3 PORTに一括書き込み |
| COUNTER COMMAND PORT書き込み関数 | COUNTER COMMAND PORTに書き込み |
| COUNTER DATA1 PORT書き込み関数 | COUNTER DATA1 PORTに書き込み |
| COUNTER DATA2 PORT書き込み関数 | COUNTER DATA2 PORTに書き込み |
| COUNTER DATA3 PORT書き込み関数 | COUNTER DATA3 PORTに書き込み |
| COMREG MODE SET関数 | COMREG MODEのセット |
| COMREG MODE CLR関数 | COMREG MODEのクリア |
| デバイス割り込みオープン関数 | デバイス割り込みのオープン |
| デバイス割り込みクローズ関数 | デバイス割り込みのクローズ |
| 割り込みコールバック設定関数 | 割り込み発生時、コールバックを行うための設定 |
| 割り込みメッセージ設定関数 | 割り込み発生時、メッセージをポストするための設定 |
| 割り込みイベント設定関数 | 割り込み発生時、シグナル状態になるイベントの設定 |
| 割り込みステータス読み出し関数 | 割り込みステータスの読み出し |
| 割り込み設定クリア関数 | 割り込み設定のクリア |
| HENSA COMMAND一括書き込み関数 | HENSA COMMAND, DATA1_2 PORTに一括書き込み |
| HENSA COMMAND PORT書き込み関数 | HENSA COMMAND PORTに書き込み |
| HENSA DATA1 PORT書き込み関数 | HENSA DATA1 PORTに書き込み |
| HENSA DATA2 PORT書き込み関数 | HENSA DATA2 PORTに書き込み |
| HENSA STATUS1 PORT読み出し関数 | HENSA STATUS1 PORTの読み出し |
| HENSA DATA1 PORT読み出し関数 | HENSA DATA1 PORTの読み出し |
| HENSA DATA2 PORT読み出し関数 | HENSA DATA2 PORTの読み出し |
| HENSA READY WAIT関数 | 脱調検出コントロール部がREADYになるまで待機 |
| HARD CONFIG COMMAND一括書き込み関数 | HARD CONFIG COMMAND, DATA1_3 PORTに一括書き込み |
| HARD CONFIG COMMAND PORT書き込み関数 | HARD CONFIG COMMAND PORTに書き込み |
| HARD CONFIG DATA1 PORT書き込み関数 | HARD CONFIG DATA1 PORTに書き込み |
| HARD CONFIG DATA2 PORT書き込み関数 | HARD CONFIG DATA2 PORTに書き込み |
| HARD CONFIG DATA3 PORT書き込み関数 | HARD CONFIG DATA3 PORTに書き込み |

次のページへ続く

前のページからの続き

| | |
|---|---|
| HARD CONFIG SIGNAL STATUS PORT読み出し関数 | SIGNAL STATUS PORTの読み出し |
| HARD CONFIG SIGNAL STATUS1 PORT読み出し関数 | SIGNAL STATUS1 PORTの読み出し |
| HARD CONFIG SIGNAL STATUS2 PORT読み出し関数 | SIGNAL STATUS2 PORTの読み出し |
| HARD CONFIG DATA1 PORT読み出し関数 | HARD CONFIG DATA1 PORTの読み出し |
| HARD CONFIG DATA2 PORT読み出し関数 | HARD CONFIG DATA2 PORTの読み出し |
| HARD CONFIG DATA3 PORT読み出し関数 | HARD CONFIG DATA3 PORTの読み出し |
| データセット1関数 | 32ビットデータをデータ構造体に格納 |
| データセット2関数 | 2つの24ビットデータをデータ構造体に格納 |
| データセット3関数 | 32ビットデータを2つのデータ構造体に格納 |
| データセット4関数 | 16ビットデータをデータ構造体に格納 |
| データゲット関数 | データ構造体の内容を32ビットデータに変換 |

## RESULT構造体

C-V870　C-V872

**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 MCO6_TAG_S_RESULT {
    WORD  MCO6_Result[4];
} MCO6_S_RESULT;
```

VB
```
Type MCO6_S_RESULT
    MCO6_Result(1 To 4) As Interger
End Type
```

VB.NET
```
Structure MCO6_S_RESULT
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> Public MCO6_Result() As Short
    Public Sub Initialize()
        ReDim MCO6_Result(4)
    End Sub
End Structure
```

C#.NET
```
struct MCO6_S_RESULT
{
     [MarshalAs( UnmanagedType.ByValArray, SizeConst=4 )] public ushort[] MCO6_Result;
     public MCO6_S_RESULT( ushort dummy )
     {
            MCO6_Result = new ushort[4];
     }
}
```

Delphi
```
MCO6_S_RESULT = record
    MCO6_Result: array[1..4] of WORD;
end;
```

**メンバ**

次に示すメンバは、C言語で表記しています。C言語の*MCO6_Result[0]*～*MCO6_Result[3]*は、Visual Basicでは*MCO6_Result(1)*～*MCO6_Result(4)*、Visual Basic.NETでは*MCO6_Result(0)*～*MCO6_Result(3)*、C#.NETでは*MCO6_Result[0]*～*MCO6_Result[3]*、Delphiでは*MCO6_Result[1]*～*MCO6_Result[4]*に対応します。

*MCO6_Result[0]*　･･･ 実行された関数を示します。(値は10進表記です。)

| 値 | 実行された関数 |
|---|---|
| 10 | デバイスオープン関数 |
| 11 | デバイスクローズ関数 |
| 12 | 割り込みコールバック設定関数 |
| 13 | 割り込みメッセージ設定関数 |
| 14 | 割り込みイベント設定関数 |
| 15 | 割り込みクリア関数 |
| 16 | デバイス割り込みオープン関数 |
| 17 | デバイス割り込みクローズ関数 |
| 18 | 割り込みステータス読み出し関数 |
| 20 | DRIVE COMMAND PORT書き込み関数 |
| 21 | DRIVE DATA1 PORT書き込み関数 |
| 22 | DRIVE DATA2 PORT書き込み関数 |
| 23 | DRIVE DATA3 PORT書き込み関数 |
| 30 | COUNTER COMMAND PORT書き込み関数 |
| 31 | COUNTER DATA1 PORT書き込み関数 |
| 32 | COUNTER DATA2 PORT書き込み関数 |
| 33 | COUNTER DATA3 PORT書き込み関数 |
| 41 | STATUS1 PORT読み出し関数 |
| 42 | STATUS2 PORT読み出し関数 |

| 値 | 実行された関数 |
|---|---|
| 81 | READY WAIT関数 |
| 82 | READY WAIT･HENSA READY WAIT状態読み出し関数 |
| 83 | READY WAIT･HENSA READY WAIT中止関数 |
| 90 | DRIVE COMMAND一括書き込み関数 |
| 91 | DRIVE DATA一括読み出し関数 |
| 92 | COUNTER COMMAND一括書き込み関数 |
| 93 | DRIVE DATA一括書き込み関数 |
| 94 | COMREG MODE SET関数 |
| 95 | COMREG MODE CLR関数 |
| 100 | HENSA COMMAND一括書き込み関数 |
| 101 | HENSA COMMAND PORT書き込み関数 |
| 102 | HENSA DATA1 PORT書き込み関数 |
| 103 | HENSA DATA2 PORT書き込み関数 |
| 104 | HENSA STATUS1 PORT読み出し関数 |
| 105 | HENSA DATA1 PORT読み出し関数 |
| 106 | HENSA DATA2 PORT読み出し関数 |
| 108 | HENSA READY WAIT関数 |
| 130 | HARD CONFIG COMMAND一括書き込み関数 |
| 131 | HARD CONFIG COMMAND PORT書き込み関数 |

| | |
|---|---|
| 43 | STATUS3 PORT読み出し関数 |
| 44 | STATUS4 PORT読み出し関数 |
| 45 | STATUS5 PORT読み出し関数 |
| 51 | DRIVE DATA1 PORT読み出し関数 |
| 52 | DRIVE DATA2 PORT読み出し関数 |
| 53 | DRIVE DATA3 PORT読み出し関数 |
| 132 | HARD CONFIG DATA1 PORT書き込み関数 |
| 133 | HARD CONFIG DATA2 PORT書き込み関数 |
| 134 | HARD CONFIG DATA3 PORT書き込み関数 |
| 135 | HARD CONFIG SIGNAL STATUS PORT読み出し関数 |
| 136 | HARD CONFIG DATA1 PORT読み出し関数 |
| 137 | HARD CONFIG DATA2 PORT読み出し関数 |
| 138 | HARD CONFIG DATA3 PORT読み出し関数 |
| 139 | HARD CONFIG SIGNAL STATUS1 PORT読み出し関数 |
| 140 | HARD CONFIG SIGNAL STATUS2 PORT読み出し関数 |

*MC06_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。 |
| 4 | C-V870.sys／C-V872.sysがロードされていません。 |
| 5 | 指定されたボード番号に誤りがあります。 |
| 6 | 軸の指定に誤りがあります。 |
| 7 | デバイスハンドルの内容が異常です。 |
| 9 | デバイスに空きがないため、オープンできません。 |
| 11 | 指定されたデバイスは、オープンされていません。 |
| 13 | 指定されたデバイスは、すでにオープンされています。 |
| 15 | READY 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 | READY WAIT中にREADY WAIT中止関数が実行されました。 |
| 18 | 同一デバイスのREADY WAIT関数が複数同時に実行されました。 |
| 19 | ボードが１枚も検出できません。 |
| 20 | 検出したボード枚数が10枚を超えました。 |
| 21 | 指定されたボード番号に該当するボードがありません。 |
| 22 | ボード番号が重複しています。 |
| 23 | 原因不明のエラーが発生しました。 |
| 24 | 指定された割り込みはすでに設定されています。 |
| 30 | INTの指定に誤りがあります。 |
| 31 | すでに指定したINTは、設定されています。 |
| 32 | 割り込みがクローズされていません。 |
| 33 | INT FACTORの指定に誤りがあります。 |
| 35 | 割り込みがオープンされていません。 |
| 36 | INTの設定が行われていません。 |
| 37 | 指定したMCC06のSTATUS1 PORT BUSY=1である為、関数の実行が出来ません。 |
| 38 | C-V870.sys／C-V872.sys内部でMCC06にCOMMANDを書き込んだ後、STATUS1 PORT BUSY=0の確認を行っていましたが、100ms待っても BUSY=0にならない為、関数の実行を中止しました。 |

*MC06_Result[2]* ･･･ 将来の拡張用です。
*MC06_Result[3]* ･･･ 将来の拡張用です。

## データ構造体

C-V870　C-V872

### 説　明

データを一括で読み書きするときに使用します。

● データを一括で読み書きするとき
・DRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE DATA3 PORTのデータを一括で書き込むとき
・DRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE DATA3 PORTのデータを一括で読み出すとき
・COUNTER DATA1 PORT、COUNTER DATA2 PORT、COUNTER DATA3 PORTのデータを一括で書き込むとき
・HENSA DATA1 PORT、HENSA DATA2 PORTのデータを一括で書き込むとき
・HARD CONFIG DATA1 PORT、HARD CONFIG DATA2 PORT、HARD CONFIG DATA3 PORTのデータを一括で書き込むとき

### 書　式

C言語
```
typedef struct MCO6_TAG_S_DATA {
    WORD MCO6_Data[4];
} MCO6_S_DATA;
```

VB
```
Type MCO6_S_DATA
    MCO6_Data(1 To 4) As Integer
End Type
```

VB.NET
```
Structure MCO6_S_DATA
    <MarshalAs(UnmanagedType.ByValArray, SizeConst:=4)> Public MCO6_Data() As Short
    Public Sub Initialize()
        ReDim MCO6_Data(4)
    End Sub
End Structure
```

C#.NET
```
struct MCO6_S_DATA
{
    [MarshalAs( UnmanagedType.ByValArray, SizeConst=4 )] public ushort[] MCO6_Data;
        MCO6_S_DATA( ushort dummy )
    {
        MCO6_Data = new ushort[4];
    }
}
```

Delphi
```
MCO6_S_DATA = record
    MCO6_Data: array[1..4] of WORD;
end;
```

### メンバ

次に示すメンバは、C言語で表記しています。C言語の*MCO6_Data[0]*～*MCO6_Data[3]*は、Visual Basicでは*MCO6_Data(1)*～*MCO6_Data(4)*、Visual Basic.NETでは*MCO6_Data(0)*～*MCO6_Data(3)*、C#.NETでは*MCO6_Data[0]*～*MCO6_Data[3]*、Delphiでは*MCO6_Data[1]*～*MCO6_Data[4]*に対応します。

*MCO6_Data[0]*　･･･ DRIVE DATA1 PORT、COUNTER DATA1 PORT、HENSA DATA1 PORT、HARD CONFIG DATA1 PORTのいずれかの内容を格納します。
*MCO6_Data[1]*　･･･ DRIVE DATA2 PORT、COUNTER DATA2 PORT、HENSA DATA2 PORT、HARD CONFIG DATA2 PORTのいずれかの内容を格納します。
*MCO6_Data[2]*　･･･ DRIVE DATA3 PORT、COUNTER DATA3 PORT、HARD COFIG DATA3 PORTのいずれかの内容を格納します。
*MCO6_Data[3]*　･･･ 将来の拡張用です。

## デバイスオープン関数

C-V870　C-V872

**機　能**

指定されたボード番号、軸で、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 MC06_BOpen(WORD *BoardNo*, WORD *Axis*, DWORD FAR **phDev*, MC06_S_RESULT FAR **psResult*);

VB　Function MC06_BOpen(ByVal *BoardNo* As Integer, ByVal *Axis* As Integer, *phDev* As Long, *psResult* As MC06_S_RESULT) As Boolean

VB.NET　Function MC06_BOpen(ByVal *BoardNo* As Short, ByVal *Axis* As Short, ByRef *phDev* As Integer, ByRef *psResult* As MC06_S_RESULT) As Boolean

C#.NET　bool MC06.BOpen(ushort *BoardNo*, ushort *Axis*, ref uint *phDev*, ref MC06_S_RESULT *psResult*);

Delphi　function MC06_BOpen(*BoardNo*:WORD; *Axis*:WORD; var *phDev*:DWORD; var *psResult*:MC06_S_RESULT): Boolean;

**引　数**

*BoardNo*　･･･ ボード番号を指定します。0～9のいずれかになります。

*Axis*　･･･ ボード上の軸を指定します。

C-V870の場合

| 引数*Axis*の値 | 軸 |
|---|---|
| MC06_X | X軸 |
| MC06_Y | Y軸 |
| MC06_Z | Z軸 |
| MC06_A | A軸 |

C-V872の場合

| 引数*Axis*の値 | 軸 | 引数*Axis*の値 | 軸 |
|---|---|---|---|
| MC06_X1 | X1軸 | MC06_X2 | X2軸 |
| MC06_Y1 | Y1軸 | MC06_Y2 | Y2軸 |
| MC06_Z1 | Z1軸 | MC06_Z2 | Z2軸 |
| MC06_A1 | A1軸 | MC06_A2 | A2軸 |

*phDev*　･･･ デバイスハンドルが格納され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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## デバイスクローズ関数

C-V870　C-V872

**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 MC06_BClose(DWORD *hDev*, MC06_S_RESULT FAR **psResult*);

VB　Function MC06_BClose(ByVal *hDev* As Long, *psResult* As MC06_S_RESULT) As Boolean

VB.NET　Function MC06_BClose(ByVal *hDev* As Integer, ByRef *psResult* As MC06_S_RESULT) As Boolean

C#.NET　bool MC06.BClose(uint *hDev*, ref MC06_S_RESULT *psResult*);

Delphi　function MC06_BClose(*hDev*:DWORD; var *psResult*:MC06_S_RESULT): Boolean;

**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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## DRIVE COMMAND一括書き込み関数

C-V870　C-V872

**機　能**

指定されたデバイスのDRIVE COMMAND PORT、DRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE DATA3 PORTにコマンドコード、データを一括書き込みします。

**書　式**

C言語　BOOL MCO6_IWDrive(DWORD *hDev*, WORD *Cmd*, MCO6_S_DATA FAR **psData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_IWDrive(ByVal *hDev* As Long, ByVal *Cmd* As Integer, *psData* As MCO6_S_DATA, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_IWDrive(ByVal *hDev* As Integer, ByVal *Cmd* As Short, ByRef *psData* As MCO6_S_DATA, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.IWDrive(uint *hDev*, ushort *Cmd*, ref MCO6_S_DATA *psData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_IWDrive(*hDev*:DWORD; *Cmd*:WORD; var *psData*:MCO6_S_DATA; var *psResult*: MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*Cmd*　･･･ 書き込むコマンドコードを指定します。
*psData*　･･･ 書き込むデータが格納されているデータ構造体のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## DRIVE DATA一括書き込み関数

C-V870　C-V872

**機　能**

指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE DATA3 PORTにデータを一括書き込みします。

**書　式**

C言語　BOOL MCO6_IWData(DWORD *hDev*, MCO6_S_DATA FAR **psData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_IWData(ByVal *hDev* As Long, *psData* As MCO6_S_DATA, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_IWData(ByVal *hDev* As Integer, ByRef *psData* As MCO6_S_DATA, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.IWData(uint *hDev*, ref MCO6_S_DATA *psData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_IWData(*hDev*:DWORD; var *psData*:MCO6_S_DATA; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*psData*　･･･ 書き込むデータが格納されているデータ構造体のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## DRIVE COMMAND PORT書き込み関数

C-V870　C-V872

**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。

**書　式**

C言語　BOOL MCO6_BWDriveCommand(DWORD *hDev*, WORD FAR **pCmd*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWDriveCommand(ByVal *hDev* As Long, *pCmd* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWDriveCommand(ByVal *hDev* As Integer, ByRef *pCmd* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWDriveCommand(uint *hDev*, ref ushort *pCmd*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWDriveCommand(*hDev*:DWORD; var *pCmd*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## DRIVE DATA1 PORT書き込み関数

C-V870　C-V872

**機　能**

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

**書　式**

C言語　BOOL MCO6_BWDriveData1(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWDriveData1(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWDriveData1(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWDriveData1(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWDriveData1(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## DRIVE DATA2 PORT書き込み関数

C-V870　C-V872

**機　能**

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

**書　式**

C言語　BOOL MCO6_BWDriveData2(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWDriveData2(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWDriveData2(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWDriveData2(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWDriveData2(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## DRIVE DATA3 PORT書き込み関数

C-V870　C-V872

**機　能**

指定されたデバイスのDRIVE DATA3 PORTにデータを書き込みます。

**書　式**

C言語　BOOL MCO6_BWDriveData3(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWDriveData3(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWDriveData3(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWDriveData3(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWDriveData3(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## STATUS1 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたデバイスのSTATUS1 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRStatus1(DWORD *hDev*, WORD FAR **pStatus*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRStatus1(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRStatus1(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRStatus1(uint *hDev*, ref ushort *pStatus*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRStatus1(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pStatus*　･･･ 読み出した内容が格納され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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラー(0)が発生したときはFALSEを返します。

## STATUS2 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたデバイスのSTATUS2 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRStatus2(DWORD *hDev*, WORD FAR **pStatus*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRStatus2(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRStatus2(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRStatus2(uint *hDev*, ref ushort *pStatus*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRStatus2(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pStatus*　･･･ 読み出した内容が格納され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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラー(0)が発生したときはFALSEを返します。

## STATUS3 PORT読み出し関数

C-V870　C-V872

### 機　能

指定されたデバイスのSTATUS3 PORTを読み出します。

### 書　式

C言語　BOOL MCO6_BRStatus3(DWORD *hDev*, WORD FAR **pStatus*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRStatus3(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRStatus3(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRStatus3(uint *hDev*, ref ushort *pStatus*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRStatus3(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

### 引　数

*hDev*　･･･ デバイスハンドルを指定します。
*pStatus*　･･･ 読み出した内容が格納され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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラー(0)が発生したときはFALSEを返します。

## STATUS4 PORT読み出し関数

C-V870　C-V872

### 機　能

指定されたデバイスのSTATUS4 PORTを読み出します。

### 書　式

C言語　BOOL MCO6_BRStatus4(DWORD *hDev*, WORD FAR **pStatus*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRStatus4(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRStatus4(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRStatus4(uint *hDev*, ref ushort *pStatus*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRStatus4(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

### 引　数

*hDev*　･･･ デバイスハンドルを指定します。
*pStatus*　･･･ 読み出した内容が格納され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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## STATUS5 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたデバイスのSTATUS5 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRStatus5(DWORD *hDev*, WORD FAR **psStatus*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRStatus5(ByVal *hDev* As Long, *psStatus* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRStatus5(ByVal *hDev* As Integer, ByRef *psStatus* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRStatus5(uint *hDev*, ref ushort *psStatus*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRStatus5(*hDev*:DWORD; var *psStatus*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*psStatus*　･･･ 読み出した内容が格納される変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## DRIVE DATA一括読み出し関数

C-V870　C-V872

**機　能**

指定されたデバイスのDRIVE DATA1 PORT、DRIVE DATA2 PORT、DRIVE DATA3 PORTを一括読み出しします。

**書　式**

C言語　BOOL MCO6_IRDrive(DWORD *hDev*, MCO6_S_DATA FAR **psData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_IRDrive(ByVal *hDev* As Long, *psData* As MCO6_S_DATA, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_IRDrive(ByVal *hDev* As Integer, ByRef *psData* As MCO6_S_DATA, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.IRDrive(uint *hDev*, ref MCO6_S_DATA *psData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_IRDrive(*hDev*:DWORD; var *psData*:MCO6_S_DATA; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*psData*　･･･ 読み出した内容が格納されるデータ構造体のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## DRIVE DATA1 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたデバイスのDRIVE DATA1 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRDriveData1(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRDriveData1(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRDriveData1(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRDriveData1(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRDriveData1(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 読み出した内容が格納され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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## DRIVE DATA2 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたデバイスのDRIVE DATA2 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRDriveData2(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRDriveData2(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRDriveData2(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRDriveData2(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRDriveData2(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 読み出した内容が格納され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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## DRIVE DATA3 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたデバイスのDRIVE DATA3 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRDriveData3(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRDriveData3(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRDriveData3(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRDriveData3(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRDriveData3(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 読み出した内容が格納され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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## READY WAIT関数

C-V870　C-V872

**機　能**

指定されたデバイス(MCCO6)がREADY（STATUS1 PORTのBUSY = 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 MCO6_BWaitDriveCommand(DWORD *hDev*, WORD *WaitTime*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWaitDriveCommand(ByVal *hDev* As Long, ByVal *WaitTime* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWaitDriveCommand(ByVal *hDev* As Integer, ByVal *WaitTime* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWaitDriveCommand(uint *hDev*, ushort *WaitTime*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWaitDriveCommand(*hDev*:DWORD; *WaitTime*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## READY WAIT・HENSA READY WAIT状態読み出し関数

C-V870　C-V872

### 機　能

指定されたデバイスのREADY WAIT関数・HENSA READ WAIT関数の状態を返します。

### 書　式

C言語　BOOL MCO6_BIsWait(DWORD *hDev*, WORD FAR **pWaitSts*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BIsWait(ByVal *hDev* As Long, *pWaitSts* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BIsWait(ByVal *hDev* As Integer, ByRef *pWaitSts* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BIsWait(uint *hDev*, ref ushort *pWaitSts*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BIsWait(*hDev*:DWORD; var *pWaitSts*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

### 引　数

*hDev*　･･･ デバイスハンドルを指定します。

*pWaitSts*　･･･ READY WAIT関数・HENSA READ WAIT関数の状態が格納される変数のポインタを指定します。

| 格納される値 | 状態 |
|---|---|
| 0 | READY WAIT関数、HENSA READ WAIT関数共に実行中でない。 |
| 1 | READY WAIT関数、HENSA READ WAIT関数のいずれかが実行中である。 |

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

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## READY WAIT・HENSA READY WAIT中止関数

C-V870　C-V872

### 機　能

指定されたデバイスのREADY WAIT関数・HENSA READ WAIT関数のREADY待ちを中止します。
READY WAIT関数のREADY待ち中は、READY WAIT関数READY待ちを中止します。
HENSA READY WAIT関数のREADY待ち中は、HENSA READY WAIT関数READY待ちを中止します。

### 書　式

C言語　BOOL MCO6_BBreakWait(DWORD *hDev*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BBreakWait(ByVal *hDev* As Long, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BBreakWait(ByVal *hDev* As Integer, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BBreakWait(uint *hDev*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BBreakWait(*hDev*:DWORD; var *psResult*:MCO6_S_RESULT): Boolean;

### 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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## COUNTER COMMAND一括書き込み関数

C-V870　C-V872

**機　能**

指定されたデバイスのCOUNTER COMMAND PORT、COUNTER DATA1 PORT、COUNTER DATA2 PORT、COUNTER DATA3 PORTにコマンドコード、データを一括書き込みします。

**書　式**

C言語　BOOL MCO6_IWCounter(DWORD *hDev*, WORD *Cmd*, MCO6_S_DATA FAR **psData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_IWCounter(ByVal *hDev* As Long, ByVal *Cmd* As Integer, *psData* As MCO6_S_DATA, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_IWCounter(ByVal *hDev* As Integer, ByVal *Cmd* As Short, ByRef *psData* As MCO6_S_DATA, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.IWCounter(uint *hDev*, ushort *Cmd*, ref MCO6_S_DATA *psData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_IWCounter(*hDev*:DWORD; *Cmd*:WORD; var *psData*:MCO6_S_DATA; var *psResult*: MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*Cmd*　･･･ 書き込むコマンドコードを指定します。
*psData*　･･･ 書き込むデータが格納されているデータ構造体のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## COUNTER COMMAND PORT書き込み関数

C-V870　C-V872

**機　能**

指定されたデバイスのCOUNTER COMMAND PORTにコマンドコードを書き込みます。

**書　式**

C言語　BOOL MCO6_BWCounterCommand(DWORD *hDev*, WORD FAR **pCmd*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWCounterCommand(ByVal *hDev* As Long, *pCmd* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWCounterCommand(ByVal *hDev* As Integer, ByRef *pCmd* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWCounterCommand(uint *hDev*, ref ushort *pCmd*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWCounterCommand(*hDev*:DWORD; var *pCmd*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## COUNTER DATA1 PORT書き込み関数

C-V870　C-V872

**機　能**

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

**書　式**

C言語　BOOL MCO6_BWCounterData1(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWCounterData1(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWCounterData1(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWCounterData1(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWCounterData1(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## COUNTER DATA2 PORT書き込み関数

C-V870　C-V872

**機　能**

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

**書　式**

C言語　BOOL MCO6_BWCounterData2(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWCounterData2(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWCounterData2(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWCounterData2(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWCounterData2(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## COUNTER DATA3 PORT書き込み関数

C-V870　C-V872

### 機　能

指定されたデバイスのCOUNTER DATA3 PORTにデータを書き込みます。

### 書　式

C言語　BOOL MCO6_BWCounterData3(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWCounterData3(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWCounterData3(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWCounterData3(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWCounterData3(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

### 引　数

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## COMREG MODE SET関数

C-V870　C-V872

### 機　能

MCC06のCOMREG機能を使用出来る状態にします。
当関数が実行されると次に示すCOMMANDがMCC06に書き込まれます。
①当関数の引数のSpdIntFlag=1の場合、HARD INITIALIZE1 COMMANDでOUT3 TYPE3_0に『SPDINT』を設定します。
当関数が実行される前にHARD INITIALIZE1 COMMANDで設定された内容は、デバイスドライバ内部で記憶されており、当関数実行時に再設定される為、OUT3 TYPE3_0、SIGNAL OUTA TYPE3_0、SIGNAL OUTB TYPE3_0は、ユーザアプリケーションで設定した内容が有効となります。尚、SpdIntFlag=0の場合は、この処理は行われません。
②HARD INITIALIZE2 COMMANDでGPIO0に『ERROR』をGPIO1に『FSEND』を設定します。
③INT FACTOR MASK COMMANDでGPIO0に『マスクしない』をGPIO1に『マスクしない』を設定します。
当関数が実行される前にINT FACTOR MASK COMMANDで設定された内容は、デバイスドライバ内部で記憶されており、当関数実行時に再設定される為、他のINT FACTOR MASKは、ユーザアプリケーションで設定した内容が有効となります。
④INT FACTOR CLR COMMANDで全INT FACTORをクリアします。
⑤SPEC INITIALIZE3 COMMANDでCOMREG ENABLEに『コマンド予約機能を有効にする』をCOMREG STOP MODEに『減速停止後にはコマンド予約を実行する』を設定します。
当関数が実行される前にSPEC INITIALIZE3 COMMANDで設定された内容は、デバイスドライバ内部で記憶されており、当関数実行時に再設定される為、SRATE STOP MODE、SRATE INDEX MODE、SRATE DRIVE TYPE、SOFT LIMIT ENABLE、END LSPD ENABLE、END PULSE STOP MODE、END PULSEは、ユーザアプリケーションで設定した内容が有効となります。
注.当関数を呼ぶ前にMCC06 STATUS1 PORT BUSY=0を確認してください。
BUSY=1で当関数が呼ばれるとエラーとなります。

### 書　式

C言語　BOOL MC06_SetComregMode(DWORD *hDev*, WORD *SpdIntFlag*, MC06_S_RESULT FAR **psResult*);

VB　Function MC06_SetComregMode(ByVal *hDev* As Long, ByVal *SpdIntFlag* As Integer, *psResult* As MC06_S_RESULT) As Boolean

VB.NET　Function MC06_SetComregMode(ByVal *hDev* As Integer, ByVal *SpdIntFlag* As Short, ByRef *psResult* As MC06_S_RESULT) As Boolean

C#.NET　bool MC06.SetComregMode(uint *hDev*, ushort *SpdIntFlag*, ref MC06_S_RESULT *psResult*);

Delphi　function MC06_SetComregMode(*hDev*:DWORD; *SpdIntFlag*:WORD; var *psResult*:MC06_S_RESULT): Boolean;

### 引　数

*hDev*　･･･ デバイスハンドルを指定します。
*SpdIntFlag*　･･･ 0:HARD INITIALIZE1 COMMANDを実行しません。
1:HARD INITIALIZE1 COMMANDでOUT3 TYPE3_0に『SPDINT』を設定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## COMREG MODE CLR関数

C-V870　C-V872

### 機　能

COMREG MODE SET関数で設定した状態をクリアします。
当関数が実行されると次に示すCOMMANDがMCC06に書き込まれます。
①SPEC INITIALIZE3 COMMANDでCOMREG ENABLEに『コマンド予約機能を無効にする』を、COMREG STOP MODEに『減速停止後にはコマンド予約を実行しない』を設定します。
当関数が実行される前にSPEC INITIALIZE3 COMMANDで設定された内容は、デバイスドライバ内部で記憶されており、当関数実行時に再設定される為、SRATE STOP MODE、SRATE INDEX MODE、SRATE DRIVE TYPE、SOFT LIMIT ENABLE、END LSPD ENABLE、END PULSE STOP MODE、END PULSEは、ユーザアプリケーションで設定した内容が有効となります。
②HARD INITIALIZE2 COMMANDでGPIO0に『汎用入力』を、GPIO1に『汎用入力』を設定します。
③HARD INITIALIZE1 COMMANDでOUT3 TYPE3_0に『常時NOT ACTIVE』を設定します。
④COMREG SET関数が実行される前にINT FACTOR MASK COMMANDで設定された内容は、デバイスドライバ内部で記憶されています。これをINT MASK COMMANDで設定します。
⑤INT FACTOR CLR COMMANDで全INT FACTORをクリアします。
注.当関数を呼ぶ前にMCC06 STATUS1 PORT BUSY=0を確認してください。
BUSY=1で当関数が呼ばれるとエラーとなります。

### 書　式

C言語　BOOL MC06_ClrComregMode(DWORD *hDev*, MC06_S_RESULT FAR **psResult*);

VB　Function MC06_ClrComregMode(ByVal *hDev* As Long, *psResult* As MC06_S_RESULT) As Boolean

VB.NET　Function MC06_ClrComregMode(ByVal *hDev* As Integer, ByRef *psResult* As MC06_S_RESULT) As Boolean

C#.NET　bool MC06.ClrComregMode(uint *hDev*, ref MC06_S_RESULT *psResult*);

Delphi　function MC06_ClrComregMode(*hDev*:DWORD; var *psResult*:MC06_S_RESULT): Boolean;

### 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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## デバイス割り込みオープン関数

C-V870　C-V872

### 機　能

指定されたデバイスの割り込みをオープンします。
当関数が実行されると次に示すCOMMANDがMCC06に書き込まれます。
①割り込みコールバック設定関数、割り込みメッセージ設定関数、割り込みイベント設定関数で設定されている INT FACTORをINT FACTOR CLR COMMANDでクリアします。
②割り込みコールバック設定関数、割り込みメッセージ設定関数、割り込みイベント設定関数で設定されている INT FACTORをINT FACTOR MASK COMMANDで『マスクしない』 に設定し、他のINT FACTORは『マスクする』に 設定します。

### 書　式

C言語　BOOL MC06_OpenDevInt(DWORD *hDev*, MC06_S_RESULT FAR **psResult*);

VB.NET　Function MC06_OpenDevInt(ByVal *hDev* As Integer, ByRef *psResult* As MC06_S_RESULT) As Boolean

C#.NET　bool MC06.OpenDevInt(uint *hDev*, ref MC06_S_RESULT *psResult*);

Delphi　function MC06_OpenDevInt(*hDev*:DWORD; var *psResult*:MC06_S_RESULT):Boolean;

### 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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## デバイス割り込みクローズ関数

C-V870　C-V872

### 機　能

指定されたデバイスの割り込みをクローズします。
当関数が実行されると次に示すCOMMANDがMCC06に書き込まれます。
①INT0、INT1、INT2のINT FACTORをINT FACTOR MASK COMMANDで『マスクする』に設定します。

### 書　式

C言語　BOOL MC06_CloseDevInt(DWORD *hDev*, MC06_S_RESULT FAR **psResult*);

VB.NET　Function MC06_CloseDevInt(ByVal *hDev* As Integer, ByRef *psResult* As MC06_S_RESULT) As Boolean

C#.NET　bool MC06.CloseDevInt(uint *hDev*, ref MC06_S_RESULT *psResult*);

Delphi　function MC06_CloseDevInt(*hDev*:DWORD; var *psResult*:MC06_S_RESULT):Boolean;

### 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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## 割り込みコールバック設定関数

C-V870　C-V872

**機　能**

指定されたデバイスで、指定された割り込みが発生した場合の、コールバック関数を設定します。この関数を複数回実行することにより、INT0、INT1、INT2のそれぞれに、異なるコールバック関数を設定することもできます。

**書　式**

C言語　BOOL MCO6_SetIntCallBack(DWORD *hDev*, WORD *IntSelect*, WORD *IntFactor*, PLPMCO6CALLBACK *Func*, DWORD *UserData*, MCO6_S_RESULT FAR **psResult*);

VB.NET　Function MCO6_SetIntCallBack(ByVal *hDev* As Integer, ByVal *IntSelect* As Short, ByVal *IntFactor* As Short, ByVal *Func* As PLPMCO6CALLBACK, ByVal *UserData* As Integer, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.SetIntCallBack(uint *hDev*, ushort *IntSelect*, ushort *Intfactor*, PLPMCO6CALLBACK *Func*, uint *UserData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_SetIntCallBack(*hDev*:DWORD; *IntSelect*:WORD; *IntFactor*:WORD; *Func*:Pointer; *UserData*:DWORD; var *psResult*:MCO6_S_RESULT):Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。

*IntSelect*　･･･ 割り込みを指定します。

| 引数*IntSelect*の値 | 割り込み |
|---|---|
| MCO6_INT0 | INT0 |
| MCO6_INT1 | INT1 |
| MCO6_INT2 | INT2 |

*IntFactor*　･･･ 割り込み要因を指定します。

| 引数*IntFactor*の値 | 割り込み要因 |
|---|---|
| MCO6_INT0_FACTOR_RDYINT | RDYINT |
| MCO6_INT0_FACTOR_COMREG_EP | COMREG EP |
| MCO6_INT0_FACTOR_nCOMREG_FL | nCOMREG FL |
| MCO6_INT1_FACTOR_SS1 | SS1 |
| MCO6_INT1_FACTOR_SS0 | SS0 |
| MCO6_INT1_FACTOR_DALM | DALM |
| MCO6_INT2_FACTOR_ADRINT | ADRINT |
| MCO6_INT2_FACTOR_CNTINT | CNTINT |
| MCO6_INT2_FACTOR_DFLINT | DFLINT |
| MCO6_INT2_FACTOR_SPDINT | SPDINT |

*Func*　･･･ コールバック関数を指定します。

*UserData*　･･･ ユーザーデータを指定します。

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

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

**コールバック関数**

**書　式**

C言語　VOID WINAPI Func(DWORD *Status*, DWORD *UserData*);

C#.NET　void Func( int *Status*, int UserData );

VB.NET　Sub Func(ByVal *Status* As Integer, ByVal UserData As Integer)

Delphi　procedure Func( *Status*:DWORD; UserData:DWORD );

**引　数**

*Status*　･･･ 割り込みステータスを格納します。

| 割り込み | 引数*Status*の値 |
|---|---|
| INT0 | 下位16ビットにSTATUS1 PORTの内容、上位16ビットは0 |
| INT1 | 0 |
| INT2 | 下位16ビットにSTATUS4 PORTの内容、上位16ビットは0 |

*UserData*　･･･ ユーザーデータを格納します。

## 割り込みコールバック設定拡張関数

C-V870　C-V872

**機　能**

指定されたデバイスで、指定された割り込みが発生した場合の、コールバック関数を設定します。この関数を複数回実行することにより、INT0、INT1、INT2のそれぞれに、異なるコールバック関数を設定することもできます。

**書　式**

C言語　BOOL MCO6_SetIntCallBackEx(DWORD *hDev*, WORD *IntSelect*, WORD *IntFactor*, PLPMCO6CALLBACKEX *Func*, DWORD64 *UserData*, MCO6_S_RESULT FAR **psResult*);

VB.NET　Function MCO6_SetIntCallBackEx(ByVal *hDev* As Integer, ByVal *IntSelect* As Short, ByVal *IntFactor* As Short, ByVal *Func* As PLPMCO6CALLBACKEX, ByVal *UserData* As Long, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.SetIntCallBackEx(uint *hDev*, ushort *IntSelect*, ushort *Intfactor*, PLPMCO6CALLBACKEX *Func*, long *UserData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_SetIntCallBackEx(*hDev*:DWORD; *IntSelect*:WORD; *IntFactor*:WORD; *Func*:Pointer; *UserData*:INT64; var *psResult*:MCO6_S_RESULT):Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。

*IntSelect*　･･･ 割り込みを指定します。

| 引数*IntSelect*の値 | 割り込み |
|---|---|
| MCO6_INT0 | INT0 |
| MCO6_INT1 | INT1 |
| MCO6_INT2 | INT2 |

*IntFactor*　･･･ 割り込み要因を指定します。

| 引数*IntFactor*の値 | 割り込み要因 |
|---|---|
| MCO6_INT0_FACTOR_RDYINT | RDYINT |
| MCO6_INT0_FACTOR_COMREG_EP | COMREG EP |
| MCO6_INT0_FACTOR_nCOMREG_FL | nCOMREG FL |
| MCO6_INT1_FACTOR_SS1 | SS1 |
| MCO6_INT1_FACTOR_SS0 | SS0 |
| MCO6_INT1_FACTOR_DALM | DALM |
| MCO6_INT2_FACTOR_ADRINT | ADRINT |
| MCO6_INT2_FACTOR_CNTINT | CNTINT |
| MCO6_INT2_FACTOR_DFLINT | DFLINT |
| MCO6_INT2_FACTOR_SPDINT | SPDINT |

*Func*　･･･ コールバック関数を指定します。

*UserData*　･･･ ユーザーデータを指定します。

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

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

**コールバック関数**

**書　式**

C言語　VOID WINAPI Func(DWORD *Status*, DWORD64 *UserData*);

C#.NET　void Func( int *Status*, long UserData );

VB.NET　Sub Func(ByVal *Status* As Integer, ByVal UserData As Long)

Delphi　procedure Func( *Status*:DWORD; UserData:INT64 );

**引　数**

*Status*　･･･ 割り込みステータスを格納します。

| 割り込み | 引数*Status*の値 |
|---|---|
| INT0 | 下位16ビットにSTATUS1 PORTの内容、上位16ビットは0 |
| INT1 | 0 |
| INT2 | 下位16ビットにSTATUS4 PORTの内容、上位16ビットは0 |

*UserData*　･･･ ユーザーデータを格納します。

## 割り込みメッセージ設定関数

C-V870　C-V872

### 機　能

指定されたデバイスで、指定された割り込みが発生した場合にポストするメッセージを設定します。この関数を複数回実行することにより、INT0、INT1、INT2のそれぞれに、異なるメッセージを設定することもできます。

### 書　式

C言語　BOOL MC06_SetIntMessage(DWORD *hDev*, WORD *IntSelect*, WORD *IntFactor*, HWND *hWnd*, UINT *Message*, DWORD *UserData*, MC06_S_RESULT FAR **psResult*);

### 引　数

*hDev*　　･･･ デバイスハンドルを指定します。

*IntSelect*　　･･･ 割り込みを指定します。

| 引数*IntSelect*の値 | 割り込み |
|---|---|
| MC06_INT0 | INT0 |
| MC06_INT1 | INT1 |
| MC06_INT2 | INT2 |

*IntFactor*　　･･･ 割り込み要因を指定します。

| 引数*IntFactor*の値 | 割り込み要因 |
|---|---|
| MC06_INT0_FACTOR_RDYINT | RDYINT |
| MC06_INT0_FACTOR_COMREG_EP | COMREG EP |
| MC06_INT0_FACTOR_nCOMREG_FL | nCOMREG FL |
| MC06_INT1_FACTOR_SS1 | SS1 |
| MC06_INT1_FACTOR_SS0 | SS0 |
| MC06_INT1_FACTOR_DALM | DALM |
| MC06_INT2_FACTOR_ADRINT | ADRINT |
| MC06_INT2_FACTOR_CNTINT | CNTINT |
| MC06_INT2_FACTOR_DFLINT | DFLINT |
| MC06_INT2_FACTOR_SPDINT | SPDINT |

*hWnd*　　･･･ メッセージのポスト先のウィンドウハンドルを指定します。

*Message*　　･･･ メッセージコードを指定します。

*UserData*　　･･･ ユーザーデータを指定します。(LPARAMに格納されます。)

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

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

● 割り込みステータス

メッセージのポスト先で、割り込みステータスを確認することができます。割り込みステータスは、第一メッセージパラメータ（WPARAM）に格納されます。

| 割り込み | 格納される値 |
|---|---|
| INT0 | 下位16ビットにSTATUS1 PORTの内容、上位16ビットは0 |
| INT1 | 0 |
| INT2 | 下位16ビットにSTATUS4 PORTの内容、上位16ビットは0 |

## 割り込みイベント設定関数

C-V870　C-V872

### 機　能

指定されたデバイスで、指定された割り込みが発生した場合にシグナル状態になるイベントを設定します。この関数を複数回実行することにより、INT0、INT1、INT2のそれぞれに、異なるイベントを設定することもできます。
イベントがシグナル状態になった後は割り込みステータス読み出し関数により、割り込みステータスを確認します。

### 書　式

C言語　BOOL MCO6_SetIntEvent(DWORD *hDev*, WORD *IntSelect*, WORD *IntFactor*, HANDLE *hEvent*, MCO6_S_RESULT FAR **psResult*);

VB.NET　Function MCO6_SetIntEvent(ByVal *hDev* As Integer, ByVal *IntSelect* As Short, ByVal *IntFactor* As Short, ByVal *hEvent* As Integer, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.SetIntEvent(uint *hDev*, ushort *IntSelect*, ushort *Intfactor*, uint *hEvent*, ref MCO6_S_RESULT *psResult*);

Delphi　MCO6_SetIntEvent(*hDev*:DWORD; *IntSelect*:WORD; *IntFactor*:WORD; *hEvent*:DWORD;
var *psResult*:MCO6_S_RESULT ):Boolean;

### 引　数

*hDev*　･･･ デバイスハンドルを指定します。

*IntSelect*　･･･ 割り込みを指定します。

| 引数*IntSelect*の値 | 割り込み |
|---|---|
| MCO6_INT0 | INT0 |
| MCO6_INT1 | INT1 |
| MCO6_INT2 | INT2 |

*IntFactor*　･･･ 割り込み要因を指定します。

| 引数*IntFactor*の値 | 割り込み要因 |
|---|---|
| MCO6_INT0_FACTOR_RDYINT | RDYINT |
| MCO6_INT0_FACTOR_COMREG_EP | COMREG EP |
| MCO6_INT0_FACTOR_nCOMREG_FL | nCOMREG FL |
| MCO6_INT1_FACTOR_SS1 | SS1 |
| MCO6_INT1_FACTOR_SS0 | SS0 |
| MCO6_INT1_FACTOR_DALM | DALM |
| MCO6_INT2_FACTOR_ADRINT | ADRINT |
| MCO6_INT2_FACTOR_CNTINT | CNTINT |
| MCO6_INT2_FACTOR_DFLINT | DFLINT |
| MCO6_INT2_FACTOR_SPDINT | SPDINT |

*hEvent*　･･･ イベントオブジェクトのハンドルを指定します。

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

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## 割り込みステータス読み出し関数

C-V870　C-V872

### 機　能

割り込みステータスを読み出します。
割り込みイベント設定関数により割り込みの設定をした場合に、この関数により割り込みステータスを確認します。

### 書　式

C言語　BOOL MC06_ReadIntStatus(DWORD *hDev*, WORD *IntSelect*, WORD FAR **pStatus*,
MC06_S_RESULT FAR **psResult*);

VB.NET　Function MC06_ReadIntStatus(ByVal *hDev* As Integer, ByVal *IntSelect* As Short,
ByRef *pStatus* As Short, ByRef *psResult* As MC06_S_RESULT) As Boolean

C#.NET　bool MC06.ReadIntStatus(uint *hDev*, ushort *IntSelect*, ref ushort *pStatus*,
ref MC06_S_RESULT *psResult*);

Delphi　function　MC06_ReadIntStatus(*hDev*:DWORD; *IntSelect*:WORD; var *pStatus*:WORD;
var *psResult*:MC06_S_RESULT):Boolean;

### 引　数

*hDev*　　･･･ デバイスハンドルを指定します。
*IntSelect*　　･･･ 割り込みを指定します。

| 引数*IntSelect*の値 | 割り込み |
|---|---|
| MC06_INT0 | INT0 |
| MC06_INT1 | INT1 |
| MC06_INT2 | INT2 |

*pStatus*　　･･･ 割り込みステータスが格納される変数のポインタを指定します。

| 割り込み | 格納される値 |
|---|---|
| INT0 | 下位16ビットにSTATUS1 PORTの内容、上位16ビットは0 |
| INT1 | 0 |
| INT2 | 下位16ビットにSTATUS4 PORTの内容、上位16ビットは0 |

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

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## 割り込み設定クリア関数

C-V870　C-V872

**機　能**

割り込みコールバック設定関数、割り込みメッセージ設定関数、割り込みイベント設定関数の設定をクリアします。
各設定関数で設定内容を変更したい場合、割り込み設定クリア関数で設定クリアしないと設定の変更を行うことができません。

**書　式**

C言語　VOID MCO6_ClearIntSet(DWORD *hDev,* MCO6_S_RESULT FAR **psResult*);

VB.NET　Function MCO6_ClearIntSet(ByVal *hDev* As Integer, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.ClearIntSet(uint *hDev*, ref MCO6_S_RESULT *psResult*);

Delphi　function　MCO6_ClearIntSet(*hDev*:DWORD; var *psResult*:MCO6_S_RESULT):Boolean;

**引　数**

*hDev*　　･･･ デバイスハンドルを指定します。

**戻り値**

この関数に、戻り値はありません。

## HENSA COMMAND一括書き込み関数

C-V870　C-V872

**機　能**

指定されたデバイスのHENSA COMMAND PORT、HENSA DATA1 PORT、HENSA DATA2 PORTにコマンドコード、データを一括書き込みします。

**書　式**

C言語　BOOL MCO6_IWHensa(DWORD *hDev*, WORD *Cmd*, MCO6_S_DATA FAR **psData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_IWHensa(ByVal *hDev* As Long, ByVal *Cmd* As Integer, *psData* As MCO6_S_DATA, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_IWHensa(ByVal *hDev* As Integer, ByVal *Cmd* As Short, ByRef *psData* As MCO6_S_DATA, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.IWHensa(uint *hDev*, ushort *Cmd*, ref MCO6_S_DATA *psData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_IWHensa(*hDev*:DWORD; *Cmd*:WORD; var *psData*:MCO6_S_DATA; var *psResult*: MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　　･･･ デバイスハンドルを指定します。
*Cmd*　　･･･ 書き込むコマンドコードを指定します。
*psData*　　･･･ 書き込むデータが格納されているデータ構造体のポインタを指定します。
　　データ構造体の各メンバは、下位8ビットが有効です。
*psResult*　　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
　　NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HENSA COMMAND PORT書き込み関数

C-V870　C-V872

**機　能**

指定されたデバイスのHENSA COMMAND PORTにコマンドコードを書き込みます。

**書　式**

C言語　BOOL MCO6_BWHensaCommand(DWORD *hDev*, WORD FAR **pCmd*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWHensaCommand(ByVal *hDev* As Long, *pCmd* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWHensaCommand(ByVal *hDev* As Integer, ByRef *pCmd* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWHensaCommand(uint *hDev*, ref ushort *pCmd*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWHensaCommand(*hDev*:DWORD; var *pCmd*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**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。
　書き込むコマンドコードが格納されている変数は、下位8ビットが有効です。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
　NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HENSA DATA1 PORT書き込み関数

C-V870　C-V872

**機　能**

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

**書　式**

C言語　BOOL MCO6_BWHensaData1(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWHensaData1(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWHensaData1(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWHensaData1(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWHensaData1(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
　書き込むデータが格納されている変数は、下位8ビットが有効です。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
　NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HENSA DATA2 PORT書き込み関数

C-V870　C-V872

**機　能**

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

**書　式**

C言語　BOOL MCO6_BWHensaData2(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWHensaData2(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWHensaData2(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWHensaData2(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWHensaData2(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
書き込むデータが格納されている変数は、下位8ビットが有効です。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HENSA STATUS1 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたデバイスのHENSA STATUS1 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRHensaStatus1(DWORD *hDev*, WORD FAR **pStatus*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRHensaStatus1(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRHensaStatus1(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRHensaStatus1(uint hDev, ref ushort pStatus, ref MCO6_S_RESULT psResult);

Delphi　function MCO6_BRHensaStatus1(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pStatus*　･･･ 読み出した内容が格納される変数のポインタを指定します。
読み出した内容は、下位8ビットが有効です。上位8ビットには、0が読み出されます。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HENSA DATA1 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたデバイスのHENSA DATA1 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRHensaData1(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRHensaData1(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRHensaData1(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRHensaData1(uint hDev, ref ushort pData, ref MCO6_S_RESULT psResult);

Delphi　function MCO6_BRHensaData1(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 読み出した内容が格納される変数のポインタを指定します。
読み出した内容は、下位8ビットが有効です。上位8ビットには、0が読み出されます。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HENSA DATA2 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたデバイスのHENSA DATA2 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRHensaData2(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRHensaData2(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRHensaData2(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRHensaData2(uint hDev, ref ushort pData, ref MCO6_S_RESULT psResult);

Delphi　function MCO6_BRHensaData2(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ デバイスハンドルを指定します。
*pData*　･･･ 読み出した内容が格納される変数のポインタを指定します。
読み出した内容は、下位8ビットが有効です。上位8ビットには、0が読み出されます。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HENSA READY WAIT関数

C-V870　C-V872

**機　能**

指定されたデバイス上の脱調検出コントロール部がREADY（HENSA STATUS1 PORTのH.RDY = 1）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 MCO6_BWaitHensaCommand(DWORD *hDev*, WORD *WaitTime*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWaitHensaCommand(ByVal *hDev* As Long, ByVal *WaitTime* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWaitHensaCommand(ByVal *hDev* As Integer, ByVal *WaitTime* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWaitHensaCommand(uint *hDev*, ushort *WaitTime*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWaitHensaCommand(*hDev*:DWORD; *WaitTime*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG COMMAND一括書き込み関数

C-V870　C-V872

**機　能**

指定されたモータコントローラボードのHARD CONFIG COMMAND PORT、HARD CONFIG DATA1 PORT、HARD CONFIG DATA 2 PORT、HARD CONFIG DATA3 PORTにコマンドコード、データを一括書き込みします。

**書　式**

C言語　BOOL MCO6_IWHardConfig(DWORD *hDev*, WORD *Cmd*, MCO6_S_DATA FAR **psData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_IWHardConfig(ByVal *hDev* As Long, ByVal *Cmd* As Integer, *psData* As MCO6_S_DATA, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_IWHardConfig(ByVal *hDev* As Integer, ByVal *Cmd* As Short, ByRef *psData* As MCO6_S_DATA, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.IWHardConfig(uint *hDev*, ushort *Cmd*, ref MCO6_S_DATA *psData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_IWHardConfig(*hDev*:DWORD; *Cmd*: WORD; var *psData*:MCO6_S_DATA; var *psResult*: MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸のデバイスハンドルを指定します。
*Cmd*　･･･ 書き込むコマンドコードを指定します。
*psData*　･･･ 書き込むデータが格納されているデータ構造体のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG COMMAND PORT書き込み関数

C-V870　C-V872

**機　能**

指定されたモータコントローラボードのHARD CONFIG COMMAND PORTにコマンドコードを書き込みます。

**書　式**

C言語　BOOL MC06_BWHardConfigCommand(DWORD *hDev*, WORD FAR **pCmd*, MC06_S_RESULT FAR **psResult*);

VB　Function MC06_BWHardConfigCommand(ByVal *hDev* As Long, *pCmd* As Integer, *psResult* As MC06_S_RESULT) As Boolean

VB.NET　Function MC06_BWHardConfigCommand(ByVal *hDev* As Integer, ByRef *pCmd* As Short, ByRef *psResult* As MC06_S_RESULT) As Boolean

C#.NET　bool MC06.BWHardConfigCommand(uint *hDev*, ref ushort *pCmd*, ref MC06_S_RESULT *psResult*);

Delphi　function MC06_BWHardConfigCommand(*hDev*:DWORD; var *pCmd*:WORD; var *psResult*:MC06_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸の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が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG DATA1 PORT書き込み関数

C-V870　C-V872

**機　能**

指定されたモータコントローラボードのHARD CONFIG DATA1 PORTにデータを書き込みます。

**書　式**

C言語　BOOL MC06_BWHardConfigData1(DWORD *hDev*, WORD FAR **pData*, MC06_S_RESULT FAR **psResult*);

VB　Function MC06_BWHardConfigData1(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MC06_S_RESULT) As Boolean

VB.NET　Function MC06_BWHardConfigData1(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MC06_S_RESULT) As Boolean

C#.NET　bool MC06.BWHardConfigData1(uint *hDev*, ref ushort *pData*, ref MC06_S_RESULT *psResult*);

Delphi　function MC06_BWHardConfigData1(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MC06_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸のデバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG DATA2 PORT書き込み関数

C-V870　C-V872

**機　能**

指定されたモータコントローラボードのHARD CONFIG DATA2 PORTにデータを書き込みます。

**書　式**

C言語　BOOL MCO6_BWHardConfigData2(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWHardConfigData2(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWHardConfigData2(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWHardConfigData2(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWHardConfigData2(*hDev*:DWORD; var *pData*:WORD; var *psResult*: MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸のデバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG DATA3 PORT書き込み関数

C-V870　C-V872

**機　能**

指定されたモータコントローラボードのHARD CONFIG DATA3 PORTにデータを書き込みます。

**書　式**

C言語　BOOL MCO6_BWHardConfigData3(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BWHardConfigData3(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BWHardConfigData3(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BWHardConfigData3(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BWHardConfigData3(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸のデバイスハンドルを指定します。
*pData*　･･･ 書き込むデータが格納されている変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG SIGNAL STATUS PORT読み出し関数

C-V870

### 機　能

指定されたモータコントローラボードのSIGNAL STATUS PORTを読み出します。

### 書　式

C言語　BOOL MCO6_BRHardConfigSigStatus(DWORD *hDev*, WORD FAR **pStatus*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRHardConfigSigStatus(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRHardConfigSigStatus(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRHardConfigSigStatus(uint *hDev*, ref ushort *pStatus*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRHardConfigSigStatus(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*: MCO6_S_RESULT): Boolean;

### 引　数

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸のデバイスハンドルを指定します。
*pStatus*　･･･ 読み出した内容が格納される変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG SIGNAL STATUS1 PORT読み出し関数

C-V872

### 機　能

指定されたモータコントローラボードのSIGNAL STATUS1 PORTを読み出します。

### 書　式

C言語　BOOL MCO6_BRHardConfigSigStatus1(DWORD *hDev*, WORD FAR **pStatus*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRHardConfigSigStatus1(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRHardConfigSigStatus1(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRHardConfigSigStatus1(uint *hDev*, ref ushort *pStatus*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRHardConfigSigStatus1(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*: MCO6_S_RESULT): Boolean;

### 引　数

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸のデバイスハンドルを指定します。
*pStatus*　･･･ 読み出した内容が格納される変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

### 戻り値

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG SIGNAL STATUS2 PORT読み出し関数

C-V872

**機　能**

指定されたモータコントローラボードのSIGNAL STATUS2 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRHardConfigSigStatus2(DWORD *hDev*, WORD FAR **pStatus*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRHardConfigSigStatus2(ByVal *hDev* As Long, *pStatus* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRHardConfigSigStatus2(ByVal *hDev* As Integer, ByRef *pStatus* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRHardConfigSigStatus2(uint *hDev*, ref ushort *pStatus*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRHardConfigSigStatus2(*hDev*:DWORD; var *pStatus*:WORD; var *psResult*: MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸のデバイスハンドルを指定します。
*pStatus*　･･･ 読み出した内容が格納される変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG DATA1 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたモータコントローラボードのHARD CONFIG DATA1 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRHardConfigData1(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRHardConfigData1(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRHardConfigData1(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRHardConfigData1(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRHardConfigData1(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸のデバイスハンドルを指定します。
*pData*　･･･ 読み出した内容が格納される変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG DATA2 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたモータコントローラボードのHARD CONFIG DATA2 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRHardConfigData2(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRHardConfigData2(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRHardConfigData2(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRHardConfigData2(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRHardConfigData2(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸のデバイスハンドルを指定します。
*pData*　･･･ 読み出した内容が格納される変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## HARD CONFIG DATA3 PORT読み出し関数

C-V870　C-V872

**機　能**

指定されたモータコントローラボードのHARD CONFIG DATA3 PORTを読み出します。

**書　式**

C言語　BOOL MCO6_BRHardConfigData3(DWORD *hDev*, WORD FAR **pData*, MCO6_S_RESULT FAR **psResult*);

VB　Function MCO6_BRHardConfigData3(ByVal *hDev* As Long, *pData* As Integer, *psResult* As MCO6_S_RESULT) As Boolean

VB.NET　Function MCO6_BRHardConfigData3(ByVal *hDev* As Integer, ByRef *pData* As Short, ByRef *psResult* As MCO6_S_RESULT) As Boolean

C#.NET　bool MCO6.BRHardConfigData3(uint *hDev*, ref ushort *pData*, ref MCO6_S_RESULT *psResult*);

Delphi　function MCO6_BRHardConfigData3(*hDev*:DWORD; var *pData*:WORD; var *psResult*:MCO6_S_RESULT): Boolean;

**引　数**

*hDev*　･･･ 対象とするモータコントローラボードのいずれかの軸のデバイスハンドルを指定します。
*pData*　･･･ 読み出した内容が格納される変数のポインタを指定します。
*psResult*　･･･ この関数を実行した結果が格納されるRESULT構造体のポインタを指定します。
NULLポインタまたは0が指定されると、実行結果が格納されません。

**戻り値**

この関数を実行した結果、正常終了したときはTRUE(1)、エラーが発生したときはFALSE(0)を返します。

## データセット1関数

C-V870　C-V872

### 機　能

32ビットデータを、次の形式で、データ構造体に格納します。

引数*psData*で示されるデータ構造体のメンバ*MC06_Data[1]*（C言語表記）［各種DATA2 PORTに対応］

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^{31}$ ← | | | | | 引数*Data*の上位16ビット（$2^{31}$～$2^{16}$） | | | | | | | | | | → $2^{16}$ |

引数*psData*で示されるデータ構造体のメンバ*MC06_Data[2]*（C言語表記）［各種DATA3 PORTに対応］

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^{15}$ ← | | | | | 引数*Data*の下位16ビット（$2^{15}$～$2^{0}$） | | | | | | | | | | → $2^{0}$ |

### 書　式

C言語　VOID MC06_SetData1(DWORD *Data*, MC06_S_DATA FAR **psData*);

VB　Sub MC06_SetData1(ByVal *Data* As Long, *psData* As MC06_S_DATA)

VB.NET　Sub MC06_SetData1(ByVal *Data* As Integer, ByRef *psData* As MC06_S_DATA)

C#.NET　void MC06.SetData1(int *Data*, ref MC06_S_DATA *psData*);

Delphi　procedure MC06_SetData1(*Data*:DWORD; var *psData*:MC06_S_DATA);

### 引　数

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

### 戻り値

この関数に、戻り値はありません。

## データセット2関数

C-V870　C-V872

### 機　能

2つの32ビットデータ(下位24ビット)を、次の形式で、データ構造体に格納します。

引数*psData*で示されるデータ構造体のメンバ*MC06_Data[0]*（C言語表記）［各種DATA1 PORTに対応］

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^{23}$ | ← | | | | | 引数*Data1*の$2^{23}$～$2^{8}$ | | | | | | | | → | $2^{8}$ |

引数*psData*で示されるデータ構造体のメンバ*MC06_Data[1]*（C言語表記）［各種DATA2 PORTに対応］

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^{7}$ | ← | | 引数*Data1*の$2^{7}$～$2^{0}$ | | | | → | $2^{0}$ | $2^{23}$ | ← | | 引数*Data2*の$2^{23}$～$2^{16}$ | | → | $2^{16}$ |

引数*psData*で示されるデータ構造体のメンバ*MC06_Data[2]*（C言語表記）［各種DATA3 PORTに対応］

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^{15}$ | ← | | | | | | 引数*Data2*の$2^{15}$～$2^{0}$ | | | | | | | → | $2^{0}$ |

### 書　式

C言語　VOID MC06_SetData2(DWORD *Data1*, DWORD *Data2*, MC06_S_DATA FAR **psData*);

VB　Sub MC06_SetData2(ByVal *Data1* As Long, ByVal *Data2* As Long, *psData* As MC06_S_DATA)

VB.NET　Sub MC06_SetData2(ByVal *Data1* As Integer, ByVal *Data2* As Integer, ByRef *psData* As MC06_S_DATA)

C#.NET　void MC06.SetData2(int *Data1*, int *Data2*, ref MC06_S_DATA *psData*);

Delphi　procedure MC06_SetData2(*Data1*:DWORD; *Data2*:DWORD; var *psData*: MC06_S_DATA);

### 引　数

*Data1*　･･･ 32ビットのデータを指定します。
*Data2*　･･･ 32ビットのデータを指定します。
*psData*　･･･ データ構造体のポインタを指定します。

### 戻り値

この関数に、戻り値はありません。

## データセット3関数

C-V870　C-V872

### 機　能

32ビットデータを、次の形式で、2つのデータ構造体に格納します。

引数*psData1*で示されるデータ構造体のメンバ*MCO6_Data[0]*（C言語表記）［各種DATA1 PORTに対応］

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^{31}$ ← | | | | | 引数*Data*の上位16ビット（$2^{31}$～$2^{16}$） | | | | | | | | | | → $2^{16}$ |

引数*psData2*で示されるデータ構造体のメンバ*MCO6_Data[0]*（C言語表記）［各種DATA1 PORTに対応］

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^{15}$ ← | | | | | 引数*Data*の下位16ビット（$2^{15}$～$2^{0}$） | | | | | | | | | | → $2^{0}$ |

### 書　式

C言語　VOID MCO6_SetData3(DWORD *Data*, MCO6_S_DATA FAR **psData1*, MCO6_S_DATA FAR **psData2*);

VB　Sub MCO6_SetData3(ByVal *Data* As Long, *psData1* As MCO6_S_DATA, *psData2* As MCO6_S_DATA)

VB.NET　Sub MCO6_SetData3(ByVal *Data* As Integer, ByRef *psData1* As MCO6_S_DATA, ByRef *psData2* As MCO6_S_DATA)

C#.NET　void MCO6.SetData3(int Data, ref MCO6_S_DATA psData1, ref MCO6_S_DATA psData2);

Delphi　procedure MCO6_SetData3(*Data*:DWORD; var *psData1*:MCO6_S_DATA; var *psData2*:MCO6_S_DATA);

### 引　数

*Data*　･･･ 32ビットのデータを指定します。
*psData1* ･･･ データ構造体のポインタを指定します。
*psData2* ･･･ データ構造体のポインタを指定します。

### 戻り値

この関数に、戻り値はありません。

## データセット4関数

C-V870　C-V872

### 機　能

16ビットデータを、次の形式で、データ構造体に格納します。

引数*psData*で示されるデータ構造体のメンバ*MCO6_Data[0]*（C言語表記）［各種DATA1 PORTに対応］

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $2^{15}$ | ← | 引数*Data*の$2^{15}$～$2^{8}$ | | | | → | $2^{8}$ |

引数*psData*で示されるデータ構造体のメンバ*MCO6_Data[1]*（C言語表記）［各種DATA2 PORTに対応］

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $2^{7}$ | ← | 引数*Data*の$2^{7}$～$2^{0}$ | | | | → | $2^{0}$ |

### 書　式

C言語　VOID MCO6_SetData4(WORD *Data*, MCO6_S_DATA FAR **psData*);

VB　Sub MCO6_SetData4(ByVal *Data* As Integer, *psData* As MCO6_S_DATA)

VB.NET　Sub MCO6_SetData4(ByVal *Data* As Short, ByRef *psData* As MCO6_S_DATA)

C#.NET　void MCO6.SetData4(short Data, ref MCO6_S_DATA psData);

Delphi　procedure MCO6_SetData4(*Data*:WORD; var *psData*:MCO6_S_DATA);

### 引　数

*Data*　･･･ 16ビットのデータを指定します。
*psData*　･･･ データ構造体のポインタを指定します。

### 戻り値

この関数に、戻り値はありません。

# データゲット関数

C-V870　C-V872

## 機　能

データ構造体の内容を、次の形式で、32ビットデータに変換し、返します。

変換後の32ビットデータ

| $2^{31}$ | $2^{30}$ | $2^{29}$ | $2^{28}$ | $2^{27}$ | $2^{26}$ | $2^{25}$ | $2^{24}$ | $2^{23}$ | $2^{22}$ | $2^{21}$ | $2^{20}$ | $2^{19}$ | $2^{18}$ | $2^{17}$ | $2^{16}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^{15}$ ← | 引数*psData*で示されるデータ構造体のメンバ*MCO6_Data[1]*（C言語表記）<br>[各種DATA2 PORTに対応] | | | | | | | | | | | | → | | $2^{0}$ |

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^{15}$ ← | 引数*psData*で示されるデータ構造体のメンバ*MCO6_Data[2]*（C言語表記）<br>[各種DATA3 PORTに対応] | | | | | | | | | | | | → | | $2^{0}$ |

## 書　式

C言語　DWORD MCO6_GetData(MCO6_S_DATA FAR **psData*);

VB　Function MCO6_GetData(*psData* As MCO6_S_DATA) As Long

VB.NET　Function MCO6_GetData(ByRef *psData* As MCO6_S_DATA) As Integer

C#.NET　int MCO6.GetData(ref MCO6_S_DATA psData);

Delphi　function MCO6_GetData(var *psData*:MCO6_S_DATA): DWORD;

## 引　数

*psData*　･･･ データ構造体のポインタを指定します。

## 戻り値

32ビットに変換されたデータを返します。

# 7. ソフト開発に必要なファイル

ユーザアプリケーション開発に必要なファイルは、インストール時に指定する次のフォルダに格納されています。
(インストール時にパスを¥Program Files指定した場合)

| | MPL-32/PCIW32 | MPL-33/PCIW64 |
|---|---|---|
| C++ Builder<br>ヘッダファイル<br>ライブラリファイル | ¥Program Files¥Mpl32¥Bin¥x86¥Builder¥Mc06A.h<br>¥Program Files¥Mpl32¥Bin¥x86¥Builder¥BcMc06A.lib | |
| C#.NET<br>ヘッダファイル | ¥Program Files¥Mpl32¥Bin¥x86¥Vc¥C#.Net¥Mc06A.cs | ¥Program Files¥Mpl33¥Bin¥x64¥Vc¥C#.Net¥Mc06A.cs |
| Deliphi<br>関数定義ファイル | ¥Program Files¥Mpl32¥Bin¥x86¥Delphi¥Mc06A.pas | |
| Visual Basic<br>関数定義ファイル | ¥Program Files¥Mpl32¥Bin¥x86¥Vb¥Mc06A.bas | |
| Visual Basic.NET<br>関数定義ファイル | ¥Program Files¥Mpl32¥Bin¥x86¥Vb.NET¥Mc06A.vb | ¥Program Files¥Mpl33¥Bin¥x64¥Vb.NET¥Mc06A.vb |
| Visual C++.NET, Visual C++<br>ヘッダファイル<br>ライブラリファイル | ¥Program Files¥Mpl32¥Bin¥x86¥Vc¥Mc06A.h<br>¥Program Files¥Mpl32¥Bin¥x86¥Vc¥VcMc06A.lib | ¥Program Files¥Mpl33¥Bin¥x64¥Vc¥Mc06A.h<br>¥Program Files¥Mpl33¥Bin¥x64¥Vc¥VcMc06A.lib |

# 8. サンプルプログラムについて

Visual Basic.NET、Visual C++.NET、Visual C++、C#.NET、C++ Builder、Visual Basic、Delphiのサンプルプログラムが用意されています。

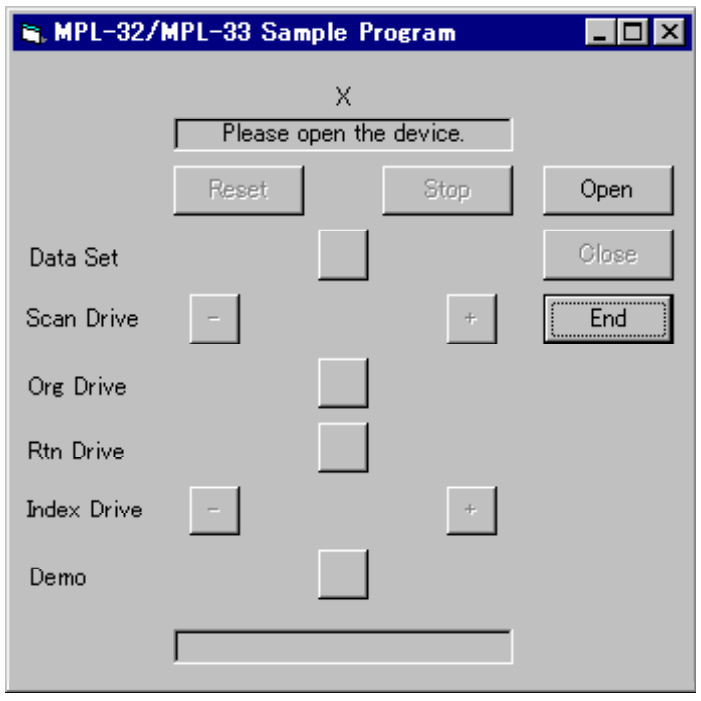
サンプルのファイルは、インストール時に指定する次のフォルダに格納されています。

（インストール時にパスを¥Program Files指定した場合）

| | MPL-32/PCIW32 | MPL-33/PCIW64 |
|---|---|---|
| C++ Builder | ¥Program Files¥Mpl32¥Sample¥x86¥Builder | |
| C#.NET | ¥Program Files¥Mpl32¥Sample¥x86¥C#.NET | ¥Program Files¥Mpl33¥Sample¥x64¥C#.NET |
| Deliphi | ¥Program Files¥Mpl32¥Sample¥x86¥Delphi | |
| Visual Basic | ¥Program Files¥Mpl32¥Sample¥x86¥Vb | |
| Visual Basic.NET | ¥Program Files¥Mpl32¥Sample¥x86¥Vb.NET | ¥Program Files¥Mpl33¥Sample¥x64¥Vb.NET |
| Visual C++.NET, Visual C++ | ¥Program Files¥Mpl32¥Sample¥x86¥Vc | ¥Program Files¥Mpl33¥Sample¥x64¥Vc |

## 8-1. 仕様

使用するボードのボード番号を0に設定してください。
本サンプルプログラムはボード番号0のＸ軸でのみ動作します。
サンプルプログラムには、Visual Basic.NET 2003、C#.NET 2003、Visual C++6.0、C++ Builder5、Visual Basic6.0、Delphi5で作成したものを用意してあります。これらは、同じ仕様で作られています。サンプルプログラムを参照する場合には、 それぞれの言語の開発環境からプロジェクトファイルを開いてください。




Openボタン　　･･･　デバイスをオープンします。
Closeボタン　　･･･　デバイスをクローズします。
Endボタン　　･･･　サンプルプログラムを終了します。
Resetボタン　　･･･　ADDRESS COUNTERを0にPRESETします。
Stopボタン　　･･･　DRIVEを即時停止します。
Data Setボタン　　･･･　次の設定にします。
　　RATE TYPE : L1-TYPE　　LSPD　　: 1000Hz
　　URATE　　: 10ms/1000Hz　　HSPD　　: 5000Hz
　　DRATE　　: 10ms/1000Hz
Scan Drive +ボタン　･･･　+(CW)方向へSCAN DRIVEします。
Scan Drive -ボタン　･･･　-(CCW)方向へSCAN DRIVEします。
Org Drive -ボタン　･･･　機械原点検出形式ORG-3でORIGIN DRIVEを行います。
Rtn Driveボタン　　･･･　絶対ADDRESS 0へ移動するABSOLUTE INDEX DRIVEを行います。
Index Drive +ボタン ･･･　+(CW)方向へ3000パルス移動するINCREMENTAL INDEX DRIVEを行います。
Index Drive -ボタン ･･･　-(CCW)方向へ3000パルス移動するINCREMENTAL INDEX DRIVEを行います。
Demoボタン　　･･･　次の動作を連続して行います。
　　①機械原点の検出(ORG DRIVE)
　　②電気原点の設定(ADDRESS COUNTERを0にPRESET)
　　③+(CW)方向へ4000パルス移動を4回繰り返す(INCREMENTAL INDEX DRIVE)
　　④絶対ADDRESS 30000へ移動(ABSOLUTE INDEX DRIVE)
　　⑤電気原点へ移動(ABSOLUTE INDEX DRIVE)

# 9.トラブルシューティング

作成したアプリケーション プログラムが正常に動作しない場合、次のことを行って下さい。

## 9-1.MCC06のCOMMANDが実行されない。

(1)RESULT構造体の確認

各関数は、アプリケーションプログラムによって与えられた引数の内容をチェックし、エラーがある場合は、FALSE(0)を返し、正常である場合は、TRUE(1)を返します。関数が正常に動作していないと思われるステップの後にブレークポイントを設定し、関数が返した値がTRUE(1)であることを確認してください。
TRUEでない場合は、エラー原因を特定する為にRESULT構造体の内容を参照してください。

## 9-2.割り込み処理が実行されない。

5-1.の割り込み処理の手順をもう一度確認してください。

■ 製品保証

保証期間と保証範囲について

● 納入品の保証期間は、納入後１ヶ年と致します。
● 上記保証期間中に当社の責により故障を生じた場合は、その修理を当社の責任において行います。
(日本国内のみ)
ただし、次に該当する場合は、この保証対象範囲から除外させて頂きます。
(1) お客様の不適当な取り扱い、ならびに使用による場合。
(2) 故障の原因が、当製品以外からの事由による場合。
(3) お客さまの改造、修理による場合。
(4) 製品出荷当時の科学・技術水準では予見が不可能だった事由による場合。
(5) その他、天災、災害等、当社の責にない場合。
(注1)ここでいう保証は、納入品単体の保証を意味するもので、納入品の故障により誘発される損害はご容赦頂きます。
(注2)当社において修理済みの製品に関しましては、保証外とさせて頂きます。

**技術相談のお問い合わせ**

TEL.(042)664-5382 FAX.(042)666-5664
E-mail s-support@melec-inc.com

**販売に関するお問い合わせ**

TEL.(042)664-5384 FAX.(042)666-2031

株式会社 メレック 制御機器営業部
〒193-0834 東京都八王子市東浅川町516-10 **URL:http://www.melec-inc.com**

 c0811