株式会社ワイツー

USB-PC104シリーズ　モーター制御ユニット　ソフトウェアマニュアル

このマニュアルについて

このソフトウェアマニュアルにはソフトウェアに関する情報が記載されています。
取扱説明書（ハードウェアのマニュアル）も併せてお読み下さい。

ソフトウェアについて

本ソフトウェアはUSB-PC104シリーズのモーター制御ユニットを制御する為のソフトウェアです。
モーター制御は、提供されるDLLの関数をコールすることで実現できますので、開発者はUSB接続であるという事を意識せずに使用することができます。

> **オブジェクト指向のライブラリも利用可能になりました**
>
> 詳細は、Y2.UsbIO を参照してください。

用語説明

ユニットID

ユニット識別用スイッチにて設定された値
（設定方法については取扱説明書を参照してください）

軸名

4軸ユニットの場合はX0・Y0・Z0・U0、8軸ユニットの場合はX0・Y0・Z0・U0・X1・Y1・Z1・U1という軸があります。
軸名の詳細については取扱説明書を参照してください。

基本仕様

接続台数

1台のパソコンから制御できるユニットの最大数はUSB-PC104シリーズ全体で16台です。

注意事項

パソコンがスタンバイや休止状態とならないようにOSを設定してください。
スタンバイや休止状態になると、以降ユニットが動作しません。

モーター制御仕様

コントロールLSI

PCD4641A または PCD4541（日本パルスモーター）

> **コントロールLSIについて**
>
> ユニットの生産時期によって異なります。
> 詳細は、モーターコントロールLSIについてを参照してください。

基準クロック

4.9152MHz

位置決めパルス数

-16,777,215 ～ +16,777,215

ドライブ出力方式

2パルス方式（CW/CCW）
1パルス方式（パルス/DIR）

モーターコントロールLSIについて

モーター制御ユニットに搭載されているモーターコントロールLSIは、ユニットの生産時期によって異なります。
（PCD4641A または PCD4541）

使用されているモーターコントロールLSIは、ユニットのシリアルNo.で判断できます。

ユニットのシリアルNo.	モーターコントロールLSI
300000以上	PCD4641A（日本パルスモーター）
299999以下	PCD4541（日本パルスモーター）

互換性について

基本的にPCD4641Aは、PCD4541からのソフト的な上位互換性があります。
したがって、Ver.3.01以下のドライバ・ソフトウェアパックを使用して作成されたソフトウェアは、どちらのユニットでも同じように動作します。

PCD4641Aで拡張された機能や設定範囲を使用する場合は、Ver.3.10.0以上のドライバ・ソフトウェアパックを使用する必要があります。

相違点

モーターコントロールLSI間の主な相違点は以下のとおりです。

相違点	PCD4641A	PCD4541
最高出力周波数	2,457,300[PPS]	400,000[PPS]
スローダウンポイント設定範囲	0～16,777,215	0～65,535
加減速レート設定範囲	2～65,535	2～1,023
現在位置カウンタ	-8,388,608～8,388,607 または 0～16,777,215	なし

動作環境（Windows）

PC

IBM PC/AT互換機（DOS/V機）

OS

Windows 11 x64
Windows 10 x86, x64
Windows 10 IoT Enterprise[1]
Windows 8.1 x86, x64
Windows 8 x86, x64[2]
Windows 7 x86, x64[2]
Windows Vista x86, x64[3]
Windows XP x64[4]
Windows XP[4]
Windows 2000[4]
Windows Me[5][6]
Windows 98, 98SE[5][6]

対応言語

Microsoft Visual C++（6.0, .NET2002以降）
Microsoft Visual C#
Microsoft Visual Basic（6.0, .NET2002以降）
VBA
Python3
その他、Win32API関数をサポートしているプログラミング言語

---

1. Windows 10 IoT Enterprise以外のWindows 10 IoTでは使用できません ↩
2. 2015年10月15日リリースの旧バージョンのドライバを使用します ↩↩
3. 2014年2月28日リリースの旧バージョンのドライバを使用します ↩
4. 2012年6月29日リリースの旧バージョンのドライバを使用します ↩↩↩
5. 2009年10月26日リリースの旧バージョンのドライバを使用します ↩↩
6. YduDioOutputStatusとYduRlyOutputStatusの2つの関数は使用できません ↩↩

エンドリミット信号（+EL, -EL）

概要

動作中に動作方向のEL信号が入力されると、即停止します。
動作開始時に動作方向のEL信号に入力がある場合は動作できません。

設定項目

- センサ極性
  YduPmcsSetSensorConfig関数でセンサ極性を選択できます。

備考

センサ極性の初期値はオフで検知（正論理）です。
EL信号を接続していない場合やオンで検知（負論理）するリミットスイッチを接続した場合は動作しません。
その場合、センサ極性をオンで検知（負論理）に変更してください。

> ⚠ EL信号を接続していない場合は機械系の衝突にご注意ください

原点スイッチ信号（ORG）

概要

原点復帰動作中にORG信号が入力されると、即停止します。
原点復帰動作開始時にORG信号に入力がある場合は動作できません。

設定項目

- センサ極性
  YduPmcsSetSensorConfig関数でセンサ極性を選択できます。

備考

センサ極性の初期値はオフで検知（正論理）です。
オンで検知（負論理）するリミットスイッチを接続した場合は動作しません。
その場合、入力論理をオンで検知（負論理）に変更してください。

減速スイッチ信号（+SD, -SD）

概要

SD信号を有効にすると、加減速動作時（※）に動作方向のSD信号の入力により起動時速度まで減速します。
SD信号の入力がある間、起動時速度で動作し、SD信号の入力が解除されると再度加速します。
※YduPmcsStartMotion関数のstartModeにPMC_ACCを設定した場合

設定項目

- センサ極性
  YduPmcsSetSensorConfig関数でセンサ極性を選択できます。
- 有効・無効
  YduPmcsSetSensorConfig関数で有効・無効を選択できます。

備考

センサ極性の初期値はオフで検知（正論理）です。
オンで検知（負論理）するリミットスイッチを接続した場合は起動速度のまま加速しません。
その場合、センサ極性をオンで検知（負論理）に変更してください。

SD信号は初期設定では無効です。使用する場合は有効に変更してください。

設定例（C言語）

```
// X0軸とY0軸とZ0軸とU0軸のSD信号を有効にする
WORD axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0;
int result = YduPmcsSetSensorConfig(0, axis, PMC_SD_CFG, 1);
```

外部スタート信号（STA）

概要

STA信号を有効にした場合、YduPmcsStartMotion関数で動作を開始させても、STA信号に入力があるまでモーターは動作しません。

設定項目

- センサ極性
  YduPmcsSetSensorConfig関数でセンサ極性を選択できます。
- 有効・無効
  YduPmcsSetSensorConfig関数で有効・無効を選択できます。

備考

STA信号のセンサ極性は軸ごとではなく、サブボードごとに共通です。

- 型番がPMC-S4で始まる4軸モーター制御ユニットの場合

  全ての軸に対応するSTA0またはSTA1の設定は共通です。
  （STA0とSTA1を異なる設定にすることは可能です）
- 型番がPMC-S8で始まる8軸モーター制御ユニットの場合

  X0軸・Y0軸・Z0軸・U0軸に対応するSTA0またはSTA1の設定は共通です。
  （STA0とSTA1を異なる設定にすることは可能です）
  X1軸・Y1軸・Z1軸・U1軸に対応するSTA2またはSTA3の設定は共通です。
  （STA2とSTA3を異なる設定にすることは可能です）

例えばX0軸に対してSTA0のセンサ極性を「オンで検知」に設定し、その後Y0軸に対してSTA0のセンサ極性を「オフで検知」に設定した場合、後から設定した値が有効となりますので、X0軸のSTA0も「オフで検知」となります。

STA信号は初期設定では無効です。
使用する場合は有効に変更してください。

STA信号は有効にする軸を選択することができます。

- 型番がPMC-S4で始まる4軸モーター制御ユニットの場合

  STA0とSTA1を各軸に対して有効にすることができます。
- 型番がPMC-S8で始まる8軸モーター制御ユニットの場合

  STA0とSTA1をX0軸・Y0軸・Z0軸・U0軸に対して、STA2とSTA3をX1軸・Y1軸・Z1軸・U1軸に対して有効にすることができます。

例えば、STA0に入力があった場合にX0軸とZ0軸の動作を開始させ、STA1に入力があった場合にY0軸とU0軸の動作を開始させるという使い方ができます。

設定例（C言語）

```
// X0軸とZ0軸に対してSTA0を有効に、Y0軸とU0軸に対してSTA1を有効に設定する

// X0軸とZ0軸に対してSTA0を有効にする
int result = YduPmcsSetSensorConfig(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_STA_STP, 0x01);
// Y0軸とU0軸に対してSTA1を有効にする
result = YduPmcsSetSensorConfig(0, PMC_AXIS_Y0 + PMC_AXIS_U0, PMC_STA_STP, 0x02);
```

強制停止信号（STP）

概要

STP信号を有効にした場合、STP信号の入力によりモーターは即停止し、その後STP信号の入力が解除されても停止状態を保ちます。
動作開始時に既にSTP信号の入力がある場合は停止状態を保ちます。

設定項目

- センサ極性
  YduPmcsSetSensorConfig関数でセンサ極性を選択できます。
- 有効・無効
  YduPmcsSetSensorConfig関数で有効・無効を選択できます。

備考

STP信号のセンサ極性は軸ごとではなく、サブボードごとに共通です。

- 型番がPMC-S4で始まる4軸モーター制御ユニットの場合

  全ての軸に対応するSTP0またはSTP1の設定は共通です。
  （STP0とSTP1を異なる設定にすることは可能です）

- 型番がPMC-S8で始まる8軸モーター制御ユニットの場合

  X0軸・Y0軸・Z0軸・U0軸に対応するSTP0またはSTP1の設定は共通です。
  （STP0とSTP1を異なる設定にすることは可能です）
  X1軸・Y1軸・Z1軸・U1軸に対応するSTP2またはSTP3の設定は共通です。
  （STP2とSTP3を異なる設定にすることは可能です）

例えばX0軸に対してSTP0のセンサ極性を「オンで検知」に設定し、その後Y0軸に対してSTP0のセンサ極性を「オフで検知」に設定した場合、後から設定した値が有効となりますので、X0軸のSTP0も「オフで検知」となります。

STP信号は初期設定では無効です。
使用する場合は有効に変更してください。

STP信号は有効にする軸を選択することができます。

- 型番がPMC-S4で始まる4軸モーター制御ユニットの場合

  STP0とSTP1を各軸に対して有効にすることができます。

- 型番がPMC-S8で始まる8軸モーター制御ユニットの場合

  STP0とSTP1をX0軸・Y0軸・Z0軸・U0軸に対して、STP2とSTP3をX1軸・Y1軸・Z1軸・U1軸に対して有効にすることができます。

例えば、STP0に入力があった場合にX0軸とZ0軸の動作を停止させ、STP1に入力があった場合にY0軸とU0軸の動作を停止させるという使い方ができます。

設定例（C言語）

```
// X0軸とZ0軸に対してSTP0を有効に、Y0軸とU0軸に対してSTP1を有効に設定する

// X0軸とZ0軸に対してSTP0を有効にする
int result = YduPmcsSetSensorConfig(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_STA_STP, 0x04);

// Y0軸とU0軸に対してSTP1を有効にする
result = YduPmcsSetSensorConfig(0, PMC_AXIS_Y0 + PMC_AXIS_U0, PMC_STA_STP, 0x08);
```

パルス出力

パルス出力論理

パルス出力の負論理・正論理を設定できます。

負論理

移動方向	+PO	-PO
+		H
-	H	

正論理

移動方向	+PO	-PO
+		L
-	L	

設定例（C言語）

```
// X0軸のパルス出力を負論理に設定する
int result = YduPmcsSetPulseConfig(0, PMC_AXIS_X0, PMC_PULSE_OUT, 0);
```

アイドリングパルス

加減速動作時（※1）、通常はスタート直後から加速を開始しますが、起動時速度で数パルス出力してから加速を開始させることができます。
設定範囲は0～7です。
加減速動作時（※1）に有効となります。
一定速起動（※2）の場合はアイドリングパルスは出力されません。

※1 YduPmcsStartMotion関数のstartModeにPMC_ACCを設定した場合
※2 YduPmcsStartMotion関数のstartModeにPMC_CONSTを設定した場合

設定例（C言語）

```
// X0軸のアイドリングパルスを5に設定する
INT result = YduPmcsSetPulseConfig(0, PMC_AXIS_X0, PMC_IDLE, 5);
```

パルス方式

2パルス方式と1パルス方式を切り替えることができます。

- 2パルス（CW/CCW）方式
  正転時は+POからパルスが出力され、逆転時は-POからパルスが出力されます。
- 1パルス（パルス/方向）方式
  +POからパルスが出力され、-POから方向信号が出力されます。

設定例（C言語）

```
// X0軸のパルス方式を1パルス方式に設定する
INT result = YduPmcsSetPulseConfig(0, PMC_AXIS_X0, PMC_METHOD, 1);
```

カウンタ

カウンタには「残パルスカウンタ」と「現在位置カウンタ」があります。

残パルスカウンタ

位置決め動作（PTP動作）時に出力パルスの残パルス数を取得することができます。
YduPmcsGetCounter関数

10000パルス出力のPTP動作時、4000パルス出力時点でカウンタ値を読み出すと6000が取得されます。

設定例（C言語）

```
// X0軸のカウンタ値を取得する
DWORD   dwData;
int result = YduPmcsGetCounter(0, PMC_AXIS_X0, &dwData);
```

現在位置カウンタ

現在位置を取得することができます。
YduPmcsGetPosition関数

> **Note**
>
> 現在位置カウンタは、シリアルNo.299999以下のユニットでは使用できません。
> また、Ver.3.01以下のドライバでは使用できません。

連続動作

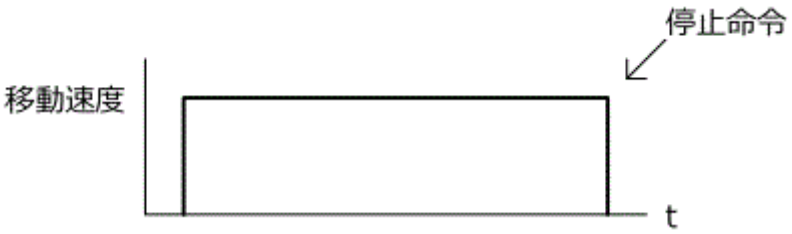
動作概要

停止要求があるまでモーターを動作させます。

動作例

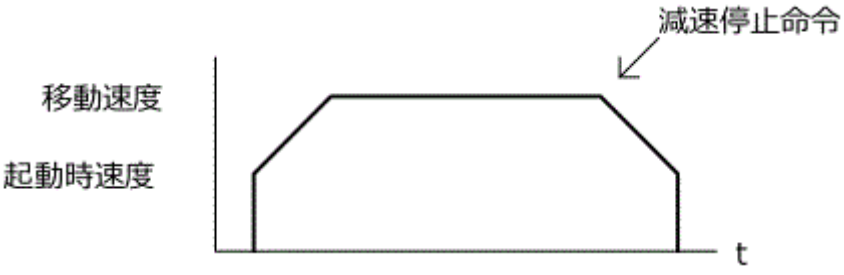
定速起動

動作開始は一定速起動を指定します。
（YduPmcsStartMotion関数のstartModeにPMC_CONSTを設定）
動作停止は即停止を指定します。
（YduPmcsStopMotion関数のstopModeにPMC_IMMEDIATE_STOPを設定）



加減速起動

動作開始は加減速起動を指定します。
（YduPmcsStartMotion関数のstartModeにPMC_ACCを設定）
動作停止は減速停止を指定します。
（YduPmcsStopMotion関数のstopModeにPMC_DEC_STOPを設定）



備考

加減速起動時（※）に動作方向のSD信号の入力により起動時速度まで減速します。
（SD信号有効設定時）
※YduPmcsStartMotion関数のstartModeにPMC_ACCを設定した場合

動作方向のEL信号の入力により即停止します。

設定例

設定項目	X0軸	Y0軸

設定項目	X0軸	Y0軸
加減速モード	直線加減速	S字加減速
起動時速度[PPS]	200	100
移動速度[PPS]	2000	3000
加減速時間[msec]	300	1000
移動方向	+方向	-方向

プログラム例（C言語）

```c
int main()
{
    int     result;
    WORD    axis;
    MOTIONPMCS  motion[4];

    result = YduOpen(0, "PMC-S4/00/00A-U");
    if(result != YDU_RESULT_SUCCESS){
        return -1;
    }

    axis = PMC_AXIS_X0 + PMC_AXIS_Y0;
    motion[0].wAccMode = PMC_ACC_NORMAL;
    motion[0].dwLowSpeed = 200;
    motion[0].dwSpeed = 2000;
    motion[0].wAccTime = 300;
    motion[0].lStep = PMC_DIR_CW;
    motion[1].wAccMode = PMC_ACC_SIN;
    motion[1].dwLowSpeed = 100;
    motion[1].dwSpeed = 3000;
    motion[1].wAccTime = 1000;
    motion[1].lStep = PMC_DIR_CCW;
    result = YduPmcsSetMotion(0, axis, PMC_JOG, motion);
    if(result != YDU_RESULT_SUCCESS){
        YduClose(0);
        return -1;
    }

    result = YduPmcsStartMotion(0, axis, PMC_ACC, PMC_JOG);
    if(result != YDU_RESULT_SUCCESS){
        YduClose(0);
        return -1;
    }

    Sleep(10000);

    result = YduPmcsStopMotion(0, axis, PMC_IMMEDIATE_STOP);
    if(result != YDU_RESULT_SUCCESS){
        YduClose(0);
        return -1;
    }
```

```
    YduClose(0);

    return 0;
}
```

原点復帰動作

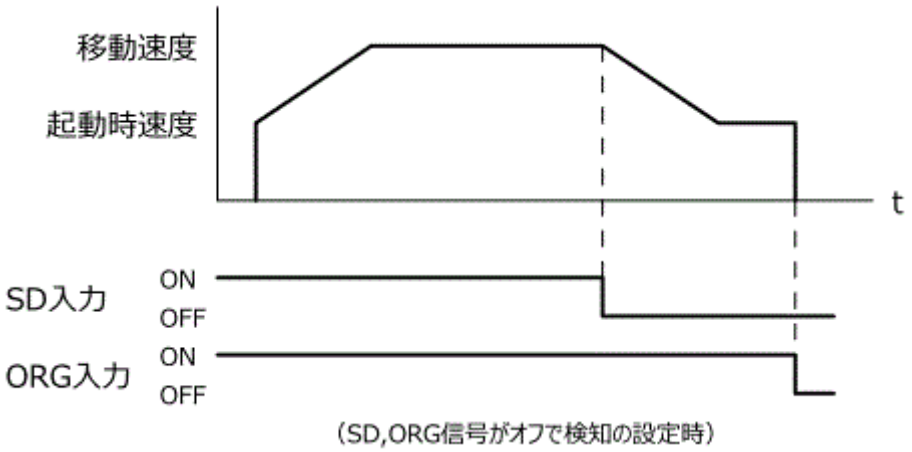
動作概要

ORG信号の入力があるまでモーターを動作させます。

動作例

加減速原点復帰

動作開始は加減速起動を指定します。
（YduPmcsStartMotion関数のstartModeにPMC_ACCを設定）



備考

加減速起動時（※）に動作方向のSD信号の入力により起動時速度まで減速します。
（SD信号有効設定時）
※YduPmcsStartMotion関数のstartModeにPMC_ACCを設定した場合

動作方向のEL信号の入力により即停止します。

設定例

設定項目	X0軸
加減速モード	直線加減速
起動時速度[PPS]	200
移動速度[PPS]	2000
加減速時間[msec]	300
移動方向	+方向

プログラム例（C言語）

```c
int main()
{
    int     result;
    WORD    axis;
    MOTIONPMCS  motion[4];

    result = YduOpen(0, "PMC-S4/00/00A-U");
    if(result != YDU_RESULT_SUCCESS){
        return -1;
    }

    axis = PMC_AXIS_X0;
    motion[0].wAccMode = PMC_ACC_NORMAL;
    motion[0].dwLowSpeed = 200;
    motion[0].dwSpeed = 2000;
    motion[0].wAccTime = 300;
    motion[0].lStep = PMC_DIR_CW;
    result = YduPmcsSetMotion(0, axis, PMC_ORG, motion);
    if(result != YDU_RESULT_SUCCESS){
        YduClose(0);
        return -1;
    }

    result = YduPmcsStartMotion(0, axis, PMC_ACC, PMC_ORG);
    if(result != YDU_RESULT_SUCCESS){
        YduClose(0);
        return -1;
    }

    Sleep(10000);

    result = YduPmcsStopMotion(0, axis, PMC_IMMEDIATE_STOP);
    if(result != YDU_RESULT_SUCCESS){
        YduClose(0);
        return -1;
    }

    YduClose(0);

    return 0;
}
```

位置決め動作（PTP動作）

動作概要

設定されたパルス数だけパルスを出力し、モーターを動作させます。

動作例

加減速位置決め動作

動作開始は加減速起動を指定します。
（YduPmcsStartMotion関数のstartModeにPMC_ACCを設定）

備考

加減速起動時（※）は減速停止します。
※YduPmcsStartMotion関数のstartModeにPMC_ACCを設定した場合

加減速起動時（※）に動作方向のSD信号の入力により起動時速度まで減速します。
（SD信号有効設定時）
※YduPmcsStartMotion関数のstartModeにPMC_ACCを設定した場合

動作方向のEL信号の入力により即停止します。

設定例

設定項目	X0軸
加減速モード	直線加減速
起動時速度[PPS]	200
移動速度[PPS]	2000
加減速時間[msec]	300
移動パルス数	10000

プログラム例（C言語）

```
int main()
{
    int     result;
    WORD    axis;
    MOTIONPMCS  motion[4];

    result = YduOpen(0, "PMC-S4/00/00A-U");
    if(result != YDU_RESULT_SUCCESS){
        return -1;
    }

    axis = PMC_AXIS_X0;
    motion[0].wAccMode = PMC_ACC_NORMAL;
    motion[0].dwLowSpeed = 200;
    motion[0].dwSpeed = 2000;
    motion[0].wAccTime = 300;
    motion[0].lStep = 10000;
    result = YduPmcsSetMotion(0, axis, PMC_PTP, motion);
    if(result != YDU_RESULT_SUCCESS){
        YduClose(0);
        return -1;
    }

    result = YduPmcsStartMotion(0, axis, PMC_ACC, PMC_PTP);
    if(result != YDU_RESULT_SUCCESS){
        YduClose(0);
        return -1;
    }

    Sleep(10000);

    result = YduPmcsStopMotion(0, axis, PMC_IMMEDIATE_STOP);
    if(result != YDU_RESULT_SUCCESS){
        YduClose(0);
        return -1;
    }

    YduClose(0);

    return 0;
}
```

実行手順

1. 準備

   ユニットとパソコンをUSBケーブルで接続し、ユニットへ電源を供給してください。
   （ユニットへ電源供給後、パソコンがユニットを認識するまでに数秒程度必要となる場合があります）

2. ユニットをオープン

   YduOpen関数を使用してユニットをオープンします。
   YduOpen関数にてオープンしたユニットは、アプリケーション終了時に必ずクローズ（YduClose関数）するようにしてください。

3. センサ設定

   YduPmcsSetSensorConfig関数を使用してセンサ設定をおこないます。

4. パルス出力設定

   YduPmcsSetPulseConfig関数を使用してパルス出力設定をおこないます。

5. 動作パラメータ設定

   YduPmcsSetMotion関数を使用して動作パラメータの設定をおこないます。

6. モーター動作

   YduPmcsStartMotion関数を使用してモーター動作を開始します。

7. モーター停止

   YduPmcsStopMotion関数を使用してモーターを停止します。

8. ユニットをクローズ

   YduClose関数を使用してユニットをクローズします。

9. 終了

   ユニットへの電源供給を止めます。

戻り値一覧

エラーコード (16進数値/10進数値)	意味	対処方法
YDU_RESULT_SUCCESS (H'00000000 / 0)	正常終了	
YDU_RESULT_ERROR (H'CE000001 / -838860799)	エラー	エラーが発生しました。 USBケーブルが抜けている事などが考えられます。
YDU_RESULT_NOT_OPEN (H'CE000002 / -838860798)	オープンされていない	オープンされていないユニットに対して操作がおこなわれました。 YduOpen関数にてオープンされたユニットに対して操作をおこなってください。
YDU_RESULT_ALREADY_OPEN (H'CE000003 / -838860797)	オープン済み	既にオープンされているユニットに対してYduOpen関数が実行されました。
YDU_RESULT_INVALID_UNIT_ID (H'CE000004 / -838860796)	ユニットIDが不正	指定されたユニットIDが不正です。 IDは0～15を指定してください。
YDU_RESULT_CANNOT_OPEN (H'CE000006 / -838860794)	デバイスがオープンできない	デバイスがオープンできませんでした。 以下の原因などが考えられます。 1. 供給電源電圧が定格範囲を外れてしまっている （供給電源の電流容量不足による電圧低下など） 2. USBケーブルまたは電源ケーブルの接続不良 3. ユニット識別スイッチの設定が間違っている 4. 型名指定が不正 5. デバイスドライバがインストールできていない
YDU_RESULT_PARAMETER_ERROR (H'CE000008 / -838860792)	引数不正	関数に渡された引数が不正です。 関数仕様やサンプルプログラムを参照し、引数の確認をしてください。
YDU_RESULT_MEM_ALLOC_ERROR (H'CE000009 / -838860791)	メモリ確保エラー	利用可能なメモリが不足しています。 不要なアプリケーションを終了するなどし、利用可能なメモリを増やしてください。
YDU_RESULT_MODELNAME_ERROR (H'CE00000A / -838860790)	型名指定が不正	指定された型名は存在していないか、サポートしていません。 型名を再度確認してください。
YDU_RESULT_HARDWARE_ERROR (H'CE00000B / -838860789)	ハードウェアエラー	以下の原因などが考えられます。 1. 動作中に供給電源電圧が定格範囲を外れてしまっている （供給電源の電流容量不足による電圧低下など） 2. 外部との接続方法に問題がある 3. 指定した型名が間違っている 上記を確認しても問題が見当たらない場合は、ユニットのハードウェアに問題が発生している可能性があります。 現象を弊社サポートまでご連絡ください。

エラーコード (16進数値/10進数値)	意味	対処方法
YDU_RESULT_NOT_SUPPORTED (H'CE00000C / -838860788)	サポートされていない	サポートされていない関数が実行されました。
YDU_RESULT_FATAL_ERROR (H'CEFFFFFF / -822083585)	致命的なエラー	致命的なエラーです。 どのような状況で発生したかを弊社サポートまでご連絡ください。

C#での注意事項

C#で関数を使用する場合、Ydu・YduDio・YduPmcsの後に"."（ドット）を付加して使用します。
（サンプルプログラムも参照してください）

C#以外	C#
YduOpen	Ydu.Open
YduClose	Ydu.Close
YduDioOutput	YduDio.Output
YduDioInput	YduDio.Input
YduPmcsSetMotion	YduPmcs.SetMotion
YduPmcsStartMotion	YduPmcs.StartMotion

関数一覧

関数	機能
YduOpen	ユニットのオープンをおこないます。
YduClose	ユニットのクローズをおこないます。

YduOpen

機能

ユニットのオープンをおこない、ユニットへのアクセスをおこなえるようにします。

書式

```
INT YduOpen(
    WORD unitId,
    LPCSTR modelName,
    WORD mode = YDU_OPEN_NORMAL
);
```

パラメータ

unitId

オープンするユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

modelName

オープンするユニットの型名を指定します。
（備考も参照してください）

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	string	String	String	String^	LPCSTR

mode

オープン時の動作を指定します。

定義	値	オープン時の動作
YDU_OPEN_NORMAL （省略可能）	0	デジタル出力・リレー出力が全てOFFになります。
YDU_OPEN_OUT_NOT_INIT	1	デジタル出力・リレー出力の出力状態は変わりません。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
オープンに失敗した場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

型番末尾に (35V) または (50V) が付加されている型番の場合、modelNameには (35V) または (50V) を除いた型番を指定してください。

例

型番	modelNameに指定する型番
PMC-S4/16/32A-U (35V)	PMC-S4/16/32A-U

> ⚠ YduOpen関数でオープンしたユニットは、アプリケーション終了時に必ずYduClose関数でクローズしてください

使用例

- 例1

  IDが0に設定されているPMC-S4/00/00A-Uをオープンします。
  デジタル出力は全てOFFになります。

- 例2

  IDが0に設定されているPMC-S4/00/00A-Uをオープンします。
  デジタル出力の状態は関数実行前と変わりません。

C#

```
// 例1
var result = Ydu.Open(0, "PMC-S4/00/00A-U");

// 例2
var result = Ydu.Open(0, "PMC-S4/00/00A-U", Ydu.YDU_OPEN_OUT_NOT_INIT);
```

VB（.NET2002以降）

```
' 例1
Dim result As Integer = YduOpen(0, "PMC-S4/00/00A-U")

' 例2
Dim result As Integer = YduOpen(0, "PMC-S4/00/00A-U", YDU_OPEN_OUT_NOT_INIT)
```

VB6.0/VBA

```
' 例1
Dim result As Long
Dim modelName as String
modelName = "PMC-S4/00/00A-U"
result = YduOpen(0, modelName)

' 例2
Dim result As Long
Dim modelName as String
modelName = "PMC-S4/00/00A-U"
result = YduOpen(0, modelName, YDU_OPEN_OUT_NOT_INIT)
```

C++/CLI

```
// 例1
int result = YduOpen(0, "PMC-S4/00/00A-U");

// 例2
int result = YduOpen(0, "PMC-S4/00/00A-U", YDU_OPEN_OUT_NOT_INIT);
```

C/C++

```
// 例1
INT result = YduOpen(0, "PMC-S4/00/00A-U");

// 例2
INT result = YduOpen(0, "PMC-S4/00/00A-U", YDU_OPEN_OUT_NOT_INIT);
```

YduClose

機能

ユニットのクローズをおこない、ユニットへのアクセスを禁止します。

書式

```
BOOL YduClose(
    WORD unitId
);
```

パラメータ

unitId

クローズするユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

戻り値

関数が正常に終了した場合はTRUEが返ります。
クローズに失敗した場合はFALSEが返ります。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	bool	Boolean	Boolean	bool	BOOL

備考

> ⚠ YduOpen関数でオープンしたユニットは、アプリケーション終了時に必ずYduClose関数でクローズしてください

使用例

ユニットIDが0のユニットのクローズ処理をおこないます。

C#

```
var result = Ydu.Close(0);
```

VB（.NET2002以降）

```
Dim result As Boolean = YduClose(0)
```

VB6.0/VBA

```
Dim result As Boolean
result = YduClose(0)
```

C++/CLI

```
bool result = YduClose(0);
```

C/C++

```
BOOL result = YduClose(0);
```

関数一覧

設定

関数	機能
YduPmcsSetSensorConfig	センサの設定をおこないます。
YduPmcsSetPulseConfig	パルス出力の設定をおこないます。
YduPmcsSetMotion	動作パラメータの設定をおこないます。
YduPmcsSetMotionEx	速度倍率を含めた動作パラメータの設定をおこないます。 （動作中に速度変更をおこなう場合はこの関数で動作パラメータの設定をおこなってください）
YduPmcsSetPosition	現在位置を設定します。

取得

関数	機能
YduPmcsGetSensorConfig	センサ設定の取得をおこないます。
YduPmcsGetPulseConfig	パルス出力設定の取得をおこないます。
YduPmcsGetMotion	動作パラメータ設定の取得をおこないます。
YduPmcsGetMotionEx	速度倍率を含めた動作パラメータの取得をおこないます。
YduPmcsGetCounter	カウンタ値の取得をおこないます。
YduPmcsGetPosition	現在位置を取得します。
YduPmcsGetSpeed	速度の取得をおこないます。
YduPmcsGetStatus	ステータスの取得をおこないます。

動作開始・停止

関数	機能
YduPmcsStartMotion	モーターの動作開始をおこないます。

関数	機能
YduPmcsStopMotion	モーターの動作停止をおこないます。
YduPmcsChangeSpeed	動作中にモーターの速度変更をおこないます。

YduPmcsSetSensorConfig

機能

センサの設定をします。

書式

```
INT YduPmcsSetSensorConfig(
    WORD unitId,
    WORD axis,
    WORD mode,
    WORD config
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

センサの設定をする軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

mode

設定する項目を指定します。

定義	値	設定する項目
PMC_LOGIC	0	センサ極性
PMC_STA_STP	1	STA/STP信号の有効・無効
PMC_SD_CFG	2	SD信号の有効・無効

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

config

設定値を指定します。
設定する項目（mode）によって値が異なります。

- wMode : PMC_LOGIC

  各センサの入力論理

bit	15	14	13	12	11	10	9	8
信号	-	-	-	-	-	-	-	STP1 (※1) STP3 (※2)

bit	7	6	5	4	3	2	1	0

bit	7	6	5	4	3	2	1	0
信号	STP0 (※1) STP2 (※2)	STA1 (※1) STA3 (※2)	STA0 (※1) STA2 (※2)	ORG	+SD	-SD	+EL	-EL

各ビットの設定値	内容
0	オフで検知（初期値）
1	オンで検知

> ※1 X0・Y0・Z0・U0指定時
> ※2 X1・Y1・Z1・U1指定時

- wMode : PMC_STA_STP

  STA/STP信号の有効または無効

bit	7	6	5	4	3	2	1	0
信号	-	-	-	-	STP1 (※1) STP3 (※2)	STP0 (※1) STP2 (※2)	STA1 (※1) STA3 (※2)	STA0 (※1) STA2 (※2)

各ビットの設定値	内容
0	無効（初期値）
1	有効

> ※1 X0・Y0・Z0・U0指定時
> ※2 X1・Y1・Z1・U1指定時

- wMode : PMC_SD_CFG

  SD信号の有効または無効

設定値	内容
0	無効（初期値）

設定値	内容
1	有効

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

センサ極性のSTAx・STPxの極性設定はサブボード毎に共通です。
4軸の場合は全ての軸が共通の設定です。
8軸の場合はX0・Y0・Z0・U0が共通の設定、X1・Y1・Z1・U1が共通の設定です。

例

X0軸に対してSTA0のセンサ極性を「オンで検知」に設定し、その後Y0軸に対してSTA0のセンサ極性を「オフで検知」に設定した場合、後から設定した値が有効となりますので、X0軸のSTA0も「オフで検知」となります。

使用例

ユニットIDが0のユニットの、X0・Y0・Z0・U0軸のORG・+SD・-SD・+EL・-EL信号を「オンで検知」に設定します。

C#

```
ushort axis = YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Y0 + YduPmcs.PMC_AXIS_Z0 +
YduPmcs.PMC_AXIS_U0;
var result = YduPmcs.SetSensorConfig(0, axis, YduPmcs.PMC_LOGIC, 0x1F);
```

VB（.NET2002以降）

```
Dim axis As Short = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0
Dim result As Integer = YduPmcsSetSensorConfig(0, axis, PMC_LOGIC, &H1F)
```

VB6.0/VBA

```
Dim result As Long
Dim axis As Integer
axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0
result = YduPmcsSetSensorConfig(0, axis, PMC_LOGIC, &H1F)
```

C++/CLI

```
unsigned short axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0;
int result = YduPmcsSetSensorConfig(0, axis, PMC_LOGIC, 0x1F);
```

C/C++

```
WORD axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0;
INT result = YduPmcsSetSensorConfig(0, axis, PMC_LOGIC, 0x1F);
```

YduPmcsSetPulseConfig

機能

パルス出力の設定をします。

パルス出力について

書式

```
INT YduPmcsSetPulseConfig(
    WORD unitId,
    WORD axis,
    WORD mode,
    WORD config
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

パルス出力の設定をする軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸

定義	値	軸
PMC_AXIS_Z1	0x40	Z1軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

mode

設定する項目を指定します。

定義	値	設定する項目
PMC_PULSE_OUT	0	パルス出力論理
PMC_IDLE	1	アイドリングパルス数
PMC_METHOD	2	パルス方式

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

config

設定値を指定します。
設定する項目（mode）によって値が異なります。

- wMode : PMC_PULSE_OUT

設定値	内容
0	負論理（初期値）
1	正論理

- wMode : PMC_IDLE

  0～7（初期値 : 0）

- wMode : PMC_METHOD

設定値	内容
0	2パルス方式（初期値）
1	1パルス方式

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットの、X0・Y0・Z0・U0軸のパルス出力論理を負論理に設定します。

C#

```
ushort axis = YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Y0 + YduPmcs.PMC_AXIS_Z0 +
YduPmcs.PMC_AXIS_U0;
var result = YduPmcs.SetPulseConfig(0, axis, YduPmcs.PMC_PULSE_OUT, 0);
```

VB（.NET2002以降）

```
Dim axis As Short = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0
Dim result As Integer = YduPmcsSetPulseConfig(0, axis, PMC_PULSE_OUT, 0)
```

VB6.0/VBA

```
Dim result As Long
Dim axis As Integer
axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0
result = YduPmcsSetPulseConfig(0, axis, PMC_PULSE_OUT, 0)
```

C++/CLI

```
unsigned short axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0;
int result = YduPmcsSetPulseConfig(0, axis, PMC_PULSE_OUT, 0);
```

C/C++

```
WORD axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0;
INT result = YduPmcsSetPulseConfig(0, axis, PMC_PULSE_OUT, 0);
```

YduPmcsSetMotion

機能

動作パラメータを設定します。

🔗 連続動作について
🔗 原点復帰動作について
🔗 位置決め動作について

書式

```
INT YduPmcsSetMotion(
    WORD unitId,
    WORD axis,
    WORD moveMode,
    MOTIONPMCS* motion
);

// 動作パラメータ構造体
typedef struct {
    WORD wAccMode;
    DWORD dwLowSpeed;
    DWORD dwSpeed;
    WORD wAccTime;
    LONG lStep;
} MOTIONPMCS, *PMOTIONPMCS;
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

動作パラメータを設定する軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸

定義	値	軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

moveMode

パラメータを設定する動作モードを指定します。

定義	値	動作モード
PMC_JOG	0	連続動作
PMC_ORG	1	原点復帰動作
PMC_PTP	2	位置決め動作

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

motion

動作パラメータが格納されているバッファへのポインタを指定します。

🔗 動作パラメータ計算方法

> **Note**
>
> ドライバ内での計算による誤差のため、動作パラメータとして設定した値と実際に設定された値が異なる場合があります。
> YduPmcsGetMotion関数を使用して実際に設定された値を確認することができます。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	MOTIONPMCS	MOTIONPMCS	MOTIONPMCS	MOTIONPMCS*	MOTIONPMCS*

wAccMode

加減速モード

定義	値	加減速モード
PMC_ACC_NORMAL	0	直線加減速
PMC_ACC_SIN	1	S字加減速

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

dwLowSpeed

起動時速度 [PPS]

設定範囲	ユニット および ドライバ
1 ～ 2457300	「ユニットのシリアルNo.が300000以上」 かつ 「ドライバがVer.3.10.0以上」の場合
1 ～ 409550	「ユニットのシリアルNo.が299999以下」 または 「ドライバがVer.3.01以下」の場合

> **Note**
>
> 移動速度（speed）以下の値を設定してください。
> また「ユニットのシリアルNo.」と「ドライバのバージョン」によって設定範囲が異なります。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	uint	Integer	Long	unsigned long	DWORD

dwSpeed

移動速度 [PPS]

設定範囲	ユニット および ドライバ
1 ～ 2457300	「ユニットのシリアルNo.が300000以上」 かつ 「ドライバがVer.3.10.0以上」の場合
1 ～ 409550	「ユニットのシリアルNo.が299999以下」 または 「ドライバがVer.1以下」の場合

> **Note**
>
> 起動時速度（dwLowSpeed）以上の値を設定してください。
> また「ユニットのシリアルNo.」と「ドライバのバージョン」によって設定範囲が異なります。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	uint	Integer	Long	unsigned long	DWORD

wAccTime

加減速時間
設定単位はmsec

速度倍率・起動時速度・移動速度により設定範囲が変わります。
動作パラメータ計算方法を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

lStep

移動方向・移動パルス数
動作モードによって値が異なります。

動作モードが「連続動作（PMC_JOG）」または「原点復帰動作（PMC_ORG）」の場合

定義	値	移動方向
PMC_DIR_CW	1	+方向
PMC_DIR_CCW	-1	-方向

動作モードが「位置決め動作（PMC_PTP）」の場合

値（設定範囲）	設定内容
-16777215 ～ +16777215	移動パルス数

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	long	LONG

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

動作パラメータを複数軸同時に設定することができます。
各軸の動作パラメータは以下のように設定されますので、動作パラメータ格納バッファ（MOTIONPMCS）は軸数分用意し、配列の先頭アドレスを関数へ設定するようにしてください。

```
// 4軸の場合
MOTIONPMCS motion[4];   // 4軸分用意
motion[0]   // X0の動作パラメータ
motion[1]   // Y0の動作パラメータ
motion[2]   // Z0の動作パラメータ
motion[3]   // U0の動作パラメータ

// 8軸の場合
MOTIONPMCS motion[8];   // 8軸分用意
```

```
motion[0]   // X0の動作パラメータ
motion[1]   // Y0の動作パラメータ
motion[2]   // Z0の動作パラメータ
motion[3]   // U0の動作パラメータ
motion[4]   // X1の動作パラメータ
motion[5]   // Y1の動作パラメータ
motion[6]   // Z1の動作パラメータ
motion[7]   // U1の動作パラメータ
```

使用例

ユニットIDが0のユニットへ、X0軸とZ0軸の動作パラメータを設定します。

C#

```
YduPmcs.MOTIONPMCS[] motion = new YduPmcs.MOTIONPMCS[4];
motion[0].wAccMode = YduPmcs.PMC_ACC_SIN;
motion[0].dwLowSpeed = 200;
motion[0].dwSpeed = 2000;
motion[0].wAccTime = 300;
motion[0].lStep = YduPmcs.PMC_DIR_CW;
motion[2].wAccMode = YduPmcs.PMC_ACC_SIN;
motion[2].dwLowSpeed = 100;
motion[2].dwSpeed = 3000;
motion[2].wAccTime = 1000;
motion[2].lStep = YduPmcs.PMC_DIR_CW;
var result = YduPmcs.SetMotion(0, YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Z0,
YduPmcs.PMC_JOG, motion);
```

VB（.NET2002以降）

```
Dim motion(3) As MOTIONPMCS
motion(0).wAccMode = PMC_ACC_SIN
motion(0).dwLowSpeed = 200
motion(0).dwSpeed = 2000
motion(0).wAccTime = 300
motion(0).lStep = PMC_DIR_CW
motion(2).wAccMode = PMC_ACC_SIN
motion(2).dwLowSpeed = 100
motion(2).dwSpeed = 3000
motion(2).wAccTime = 1000
motion(2).lStep = PMC_DIR_CW
Dim result As Integer = YduPmcsSetMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motion)
```

VB6.0/VBA

```
Dim result As Long
Dim motion(3) As MOTIONPMCS
motion(0).wAccMode = PMC_ACC_SIN
motion(0).dwLowSpeed = 200
motion(0).dwSpeed = 2000
motion(0).wAccTime = 300
motion(0).lStep = PMC_DIR_CW
motion(2).wAccMode = PMC_ACC_SIN
motion(2).dwLowSpeed = 100
motion(2).dwSpeed = 3000
motion(2).wAccTime = 1000
```

```
motion(2).lStep = PMC_DIR_CW
result = YduPmcsSetMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motion(0))
```

C++/CLI

```
MOTIONPMCS motion[4];
motion[0].wAccMode = PMC_ACC_SIN;
motion[0].dwLowSpeed = 200;
motion[0].dwSpeed = 2000;
motion[0].wAccTime = 300;
motion[0].lStep = PMC_DIR_CW;
motion[2].wAccMode = PMC_ACC_SIN;
motion[2].dwLowSpeed = 100;
motion[2].dwSpeed = 3000;
motion[2].wAccTime = 1000;
motion[2].lStep = PMC_DIR_CW;
int result = YduPmcsSetMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motion);
```

C/C++

```
INT result;
MOTIONPMCS motion[4];
motion[0].wAccMode = PMC_ACC_SIN;
motion[0].dwLowSpeed = 200;
motion[0].dwSpeed = 2000;
motion[0].wAccTime = 300;
motion[0].lStep = PMC_DIR_CW;
motion[2].wAccMode = PMC_ACC_SIN;
motion[2].dwLowSpeed = 100;
motion[2].dwSpeed = 3000;
motion[2].wAccTime = 1000;
motion[2].lStep = PMC_DIR_CW;
result = YduPmcsSetMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motion);
```

YduPmcsSetMotionEx

機能

速度倍率を含めた動作パラメータを設定します。
動作中に速度変更をおこなう場合はYduPmcsSetMotion関数ではなく、こちらの関数を使用してください。

> **Note**
>
> 速度倍率については、動作パラメータ計算方法を参照してください。

連続動作について
原点復帰動作について
位置決め動作について

書式

```
INT YduPmcsSetMotionEx(
    WORD unitId,
    WORD axis,
    WORD moveMode,
    MOTIONEXPMCS* motionEx
);

// 動作パラメータ構造体
typedef struct {
    WORD wSpeedRate;
    WORD wAccMode;
    DWORD dwLowSpeed;
    DWORD dwSpeed;
    WORD wAccTime;
    LONG lStep;
} MOTIONEXPMCS, *PMOTIONEXPMCS;
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

動作パラメータを設定する軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

moveMode

パラメータを設定する動作モードを指定します。

定義	値	動作モード
PMC_JOG	0	連続動作
PMC_ORG	1	原点復帰動作
PMC_PTP	2	位置決め動作

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

motionEx

動作パラメータが格納されているバッファへのポインタを指定します。

🔗 動作パラメータ計算方法

> **Note**
>
> ドライバ内での計算による誤差のため、動作パラメータとして設定した値と実際に設定された値が異なる場合があります。
> YduPmcsGetMotionEx関数を使用して実際に設定された値を確認することができます。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	MOTIONEXPMCS	MOTIONEXPMCS	MOTIONEXPMCS	MOTIONEXPMCS*	MOTIONEXPMCS*

wSpeedRate

速度倍率

設定範囲	ユニット および ドライバ
1 ～ 300	「ユニットのシリアルNo.が300000以上」 かつ 「ドライバがVer.3.10.0以上」の場合
1 ～ 50	「ユニットのシリアルNo.が299999以下」 または 「ドライバがVer.3.01以下」の場合

> **Note**
>
> 「ユニットのシリアルNo.」と「ドライバのバージョン」によって設定範囲が異なります。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

wAccMode

加減速モード

定義	値	加減速モード
PMC_ACC_NORMAL	0	直線加減速
PMC_ACC_SIN	1	S字加減速

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

dwLowSpeed

起動時速度 [PPS]

設定範囲	ユニット および ドライバ
1 ～ 2457300	「ユニットのシリアルNo.が300000以上」 かつ 「ドライバがVer.3.10.0以上」の場合
1 ～ 409550	「ユニットのシリアルNo.が299999以下」 または 「ドライバがVer.3.01以下」の場合

> **Note**
>
> 移動速度（dwSpeed）以下の値を設定してください。
> また「ユニットのシリアルNo.」と「ドライバのバージョン」によって設定範囲が異なります。

速度倍率によって設定できる範囲が変わります。
速度倍率による設定範囲外の値を指定した場合は設定範囲内の値に変更されます。
（設定範囲については動作パラメータ計算方法を参照してください）

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	uint	Integer	Long	unsigned long	DWORD

dwSpeed

移動速度 [PPS]

設定範囲	ユニット および ドライバ
1 ～ 2457300	「ユニットのシリアルNo.が300000以上」 かつ 「ドライバがVer.3.10.0以上」の場合
1 ～ 409550	「ユニットのシリアルNo.が299999以下」 または 「ドライバがVer.3.01以下」の場合

> **Note**
>
> 起動時速度（dwLowSpeed）以上の値を設定してください。
> また「ユニットのシリアルNo.」と「ドライバのバージョン」によって設定範囲が異なります。

速度倍率によって設定できる範囲が変わります。
速度倍率による設定範囲外の値を指定した場合は設定範囲内の値に変更されます。
設定範囲については動作パラメータ計算方法を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	uint	Integer	Long	unsigned long	DWORD

wAccTime

加減速時間
設定単位はmsec

速度倍率・起動時速度・移動速度により設定範囲が変わります。
動作パラメータ計算方法を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

lStep

移動方向・移動パルス数
動作モードによって値が異なります。

動作モードが「連続動作（PMC_JOG）」または「原点復帰動作（PMC_ORG）」の場合

定義	値	移動方向
PMC_DIR_CW	1	+方向
PMC_DIR_CCW	-1	-方向

動作モードが「位置決め動作（PMC_PTP）」の場合

値（設定範囲）	設定内容
-16777215 ～ +16777215	移動パルス数

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	long	LONG

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

動作パラメータを複数軸同時に設定することができます。
各軸の動作パラメータは以下のように設定されますので、動作パラメータ格納バッファ（MOTIONEXPMCS）は軸数分用意し、配列の先頭アドレスを関数へ設定するようにしてください。

```
// 4軸の場合
MOTIONEXPMCS motionEx\[4\];   // 4軸分用意
motionEx\[0\] // X0の動作パラメータ
motionEx\[1\] // Y0の動作パラメータ
motionEx\[2\] // Z0の動作パラメータ
motionEx\[3\] // U0の動作パラメータ

// 8軸の場合
MOTIONEXPMCS motionEx\[8\];   // 8軸分用意
motionEx\[0\] // X0の動作パラメータ
motionEx\[1\] // Y0の動作パラメータ
motionEx\[2\] // Z0の動作パラメータ
motionEx\[3\] // U0の動作パラメータ
motionEx\[4\] // X1の動作パラメータ
motionEx\[5\] // Y1の動作パラメータ
motionEx\[6\] // Z1の動作パラメータ
motionEx\[7\] // U1の動作パラメータ
```

使用例

ユニットIDが0のユニットへ、X0軸とZ0軸の動作パラメータを設定します。

C#

```
YduPmcs.MOTIONEXPMCS[] motionEx = new YduPmcs.MOTIONEXPMCS[4];
motionEx[0].wSpeedRate = 1;
motionEx[0].wAccMode = YduPmcs.PMC_ACC_SIN;
motionEx[0].dwLowSpeed = 200;
motionEx[0].dwSpeed = 2000;
motionEx[0].wAccTime = 300;
motionEx[0].lStep = YduPmcs.PMC_DIR_CW;
motionEx[2].wSpeedRate = 1;
```

```
motionEx[2].wAccMode = YduPmcs.PMC_ACC_SIN;
motionEx[2].dwLowSpeed = 100;
motionEx[2].dwSpeed = 3000;
motionEx[2].wAccTime = 1000;
motionEx[2].lStep = YduPmcs.PMC_DIR_CW;
var result = YduPmcs.SetMotionEx(0, YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Z0,
YduPmcs.PMC_JOG, motionEx);
```

VB（.NET2002以降）

```
Dim motionEx(3) As MOTIONEXPMCS
motionEx(0).wSpeedRate = 1
motionEx(0).wAccMode = PMC_ACC_SIN
motionEx(0).dwLowSpeed = 200
motionEx(0).dwSpeed = 2000
motionEx(0).wAccTime = 300
motionEx(0).lStep = PMC_DIR_CW
motionEx(2).wSpeedRate = 1
motionEx(2).wAccMode = PMC_ACC_SIN
motionEx(2).dwLowSpeed = 100
motionEx(2).dwSpeed = 3000
motionEx(2).wAccTime = 1000
motionEx(2).lStep = PMC_DIR_CW
Dim result As Integer = YduPmcsSetMotionEx(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motionEx)
```

VB6.0/VBA

```
Dim result As Long
Dim motionEx(3) As MOTIONEXPMCS
motionEx(0).wSpeedRate = 1
motionEx(0).wAccMode = PMC_ACC_SIN
motionEx(0).dwLowSpeed = 200
motionEx(0).dwSpeed = 2000
motionEx(0).wAccTime = 300
motionEx(0).lStep = PMC_DIR_CW
motionEx(2).wSpeedRate = 1
motionEx(2).wAccMode = PMC_ACC_SIN
motionEx(2).dwLowSpeed = 100
motionEx(2).dwSpeed = 3000
motionEx(2).wAccTime = 1000
motionEx(2).lStep = PMC_DIR_CW
result = YduPmcsSetMotionEx(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motionEx(0))
```

C++/CLI

```
MOTIONEXPMCS motionEx[4];
motionEx[0].wSpeedRate = 1;
motionEx[0].wAccMode = PMC_ACC_SIN;
motionEx[0].dwLowSpeed = 200;
motionEx[0].dwSpeed = 2000;
motionEx[0].wAccTime = 300;
motionEx[0].lStep = PMC_DIR_CW;
motionEx[2].wSpeedRate = 1;
motionEx[2].wAccMode = PMC_ACC_SIN;
motionEx[2].dwLowSpeed = 100;
motionEx[2].dwSpeed = 3000;
motionEx[2].wAccTime = 1000;
```

```
motionEx[2].lStep = PMC_DIR_CW;
int result = YduPmcsSetMotionEx(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motionEx);
```

C/C++

```
INT result;
MOTIONEXPMCS motionEx[4];
motionEx[0].wSpeedRate = 1;
motionEx[0].wAccMode = PMC_ACC_SIN;
motionEx[0].dwLowSpeed = 200;
motionEx[0].dwSpeed = 2000;
motionEx[0].wAccTime = 300;
motionEx[0].lStep = PMC_DIR_CW;
motionEx[2].wSpeedRate = 1;
motionEx[2].wAccMode = PMC_ACC_SIN;
motionEx[2].dwLowSpeed = 100;
motionEx[2].dwSpeed = 3000;
motionEx[2].wAccTime = 1000;
motionEx[2].lStep = PMC_DIR_CW;
result = YduPmcsSetMotionEx(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motionEx);
```

YduPmcsSetPosition

> **Note**
>
> 本関数は、シリアルNo.299999以下のユニットでは使用できません。
> また、Ver.3.01以下のドライバでは使用できません。

機能

現在位置を設定します。

書式

```
INT YduPmcsSetPosition(
    WORD unitId,
    WORD axis,
    INT position
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

現在位置を取得する軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸

定義	値	軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

position

設定する現在位置を指定します。

値の範囲は、-8,388,608～+8,388,607です。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットの、X0軸とY0軸の現在位置を1,000に設定します。

C#

```
var result = YduPmcs.SetPosition(0, YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Y0, 1000);
```

VB（.NET2002以降）

```
Dim result As Integer = YduPmcsSetPosition(0, PMC_AXIS_X0 + PMC_AXIS_Y0, 1000)
```

VB6.0/VBA

```
Dim result As Long
result = YduPmcsSetPosition(0, PMC_AXIS_X0 + PMC_AXIS_Y0, 1000)
```

C++/CLI

```
int result = YduPmcsSetPosition(0, PMC_AXIS_X0 + PMC_AXIS_Y0, 1000);
```

C/C++

```
INT result = YduPmcsSetPosition(0, PMC_AXIS_X0 + PMC_AXIS_Y0, 1000);
```

YduPmcsGetSensorConfig

機能

センサの設定値を取得します。

書式

```
INT YduPmcsGetSensorConfig(
    WORD unitId,
    WORD axis,
    WORD mode,
    WORD* config
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

設定値を取得する軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

mode

取得する項目を指定します。

定義	値	取得する項目
PMC_LOGIC	0	センサ極性
PMC_STA_STP	1	STA/STP信号有効・無効
PMC_SD_CFG	2	SD信号有効・無効

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

config

設定値を格納するバッファへのポインタを指定します。
取得する項目（mode）によって値が異なります。

- wMode : PMC_LOGIC

  各センサの入力論理

bit	15	14	13	12	11	10	9	8
信号	-	-	-	-	-	-	-	STP1 (※1) STP3 (※2)

bit	7	6	5	4	3	2	1	0

bit	7	6	5	4	3	2	1	0
信号	STP0 (※1) STP2 (※2)	STA1 (※1) STA3 (※2)	STA0 (※1) STA2 (※2)	ORG	+SD	-SD	+EL	-EL

各ビットの値	内容
0	オフで検知
1	オンで検知

> ※1 X0・Y0・Z0・U0指定時
> ※2 X1・Y1・Z1・U1指定時

- wMode : PMC_STA_STP

  STA/STP信号の有効または無効

bit	7	6	5	4	3	2	1	0
信号	-	-	-	-	STP1 (※1) STP3 (※2)	STP0 (※1) STP2 (※2)	STA1 (※1) STA3 (※2)	STA0 (※1) STA2 (※2)

各ビットの値	内容
0	無効
1	有効

> ※1 X0・Y0・Z0・U0指定時
> ※2 X1・Y1・Z1・U1指定時

- wMode : PMC_SD_CFG

  SD信号の有効または無効

値	内容
0	無効

値	内容
1	有効

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out ushort	Short	Integer	unsigned short*	WORD*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

センサ極性のSTAx・STPxの極性設定はサブボード毎に共通です。
4軸の場合は全ての軸が共通の設定です。
8軸の場合はX0・Y0・Z0・U0が共通の設定、X1・Y1・Z1・U1が共通の設定です。

使用例

ユニットIDが0のユニットから、X0軸のセンサ極性設定を取得します。

C#

```
var result = YduPmcs.GetSensorConfig(0, YduPmcs.PMC_AXIS_X0, YduPmcs.PMC_LOGIC, out var
config);
```

VB（.NET2002以降）

```
Dim config As Short
Dim result As Integer = YduPmcsGetSensorConfig(0, PMC_AXIS_X0, PMC_LOGIC, config)
```

VB6.0/VBA

```
Dim result As Long
Dim config As Integer
result = YduPmcsGetSensorConfig(0, PMC_AXIS_X0, PMC_LOGIC, config)
```

C++/CLI

```
unsigned short config;
int result = YduPmcsGetSensorConfig(0, PMC_AXIS_X0, PMC_LOGIC, &config);
```

C/C++

```
WORD config;
INT result = YduPmcsGetSensorConfig(0, PMC_AXIS_X0, PMC_LOGIC, &config);
```

YduPmcsGetPulseConfig

機能

パルス出力の設定値を取得します。

パルス出力について

書式

```
INT YduPmcsGetPulseConfig(
    WORD unitId,
    WORD axis,
    WORD mode,
    WORD* config
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

設定値を取得する軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸

定義	値	軸
PMC_AXIS_Z1	0x40	Z1軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

mode

取得する項目を指定します。

定義	値	取得する項目
PMC_PULSE_OUT	0	パルス出力論理
PMC_IDLE	1	アイドリングパルス数
PMC_METHOD	2	パルス方式

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

config

設定値を格納するバッファへのポインタを指定します。
取得する項目（mode）によって値が異なります。

- wMode : PMC_PULSE_OUT

値	内容
0	負論理
1	正論理

- wMode : PMC_IDLE

  0～7

- wMode : PMC_METHOD

値	内容
0	2パルス方式
1	1パルス方式

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out ushort	Short	Integer	unsigned short*	WORD*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットから、X0軸のパルス出力論理を取得します。

C#

```
var result = YduPmcs.GetPulseConfig(0, YduPmcs.PMC_AXIS_X0, YduPmcs.PMC_PULSE_OUT, out var
config);
```

VB（.NET2002以降）

```
Dim config As Short
Dim result As Integer = YduPmcsGetPulseConfig(0, PMC_AXIS_X0, PMC_PULSE_OUT, config)
```

VB6.0/VBA

```
Dim result As Long
Dim config As Integer
result = YduPmcsGetPulseConfig(0, PMC_AXIS_X0, PMC_PULSE_OUT, config)
```

C++/CLI

```
unsigned short config;
int result = YduPmcsGetPulseConfig(0, PMC_AXIS_X0, PMC_PULSE_OUT, &config);
```

C/C++

```
WORD config;
INT result = YduPmcsGetPulseConfig(0, PMC_AXIS_X0, PMC_PULSE_OUT, &config);
```

YduPmcsGetMotion

機能

動作パラメータを取得します。

書式

```
INT YduPmcsGetMotion(
    WORD unitId,
    WORD axis,
    WORD moveMode,
    MOTIONPMCS* motion
);

// 動作パラメータ構造体
typedef struct {
    WORD wAccMode;
    DWORD dwLowSpeed;
    DWORD dwSpeed;
    WORD wAccTime;
    LONG lStep;
} MOTIONPMCS, *PMOTIONPMCS;
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

動作パラメータを読み込む軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸

定義	値	軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

moveMode

パラメータを取得する動作モードを指定します。

定義	値	動作モード
PMC_JOG	0	連続動作
PMC_ORG	1	原点復帰動作
PMC_PTP	2	位置決め動作

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

motion

動作パラメータを格納するバッファへのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	MOTIONPMCS	MOTIONPMCS	MOTIONPMCS	MOTIONPMCS*	MOTIONPMCS*

wAccMode

加減速モード

定義	値	加速モード
PMC_ACC_NORMAL	0	直線加減速
PMC_ACC_SIN	1	S字加減速

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

dwLowSpeed

起動時速度 [PPS]

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	uint	Integer	Long	unsigned long	DWORD

dwSpeed

移動速度 [PPS]

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	uint	Integer	Long	unsigned long	DWORD

wAccTime

加減速時間[msec]

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

lStep

移動方向・移動パルス数
動作モードによって値が異なります。

動作モードが「連続動作（PMC_JOG）」または「原点復帰動作（PMC_ORG）」の場合

定義	値	移動方向
PMC_DIR_CW	1	+方向
PMC_DIR_CCW	-1	-方向

動作モードが「位置決め動作（PMC_PTP）」の場合

値（範囲）	設定内容
-16,777,215 ～ +16,777,215	移動パルス数

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	long	LONG

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

速度及び加減速時間の値はYduPmcsSetMotion関数で設定した値と異なる場合がありますが、ドライバ内での計算による誤差によるものです。
（動作パラメータ計算方法を参照してください）

動作パラメータを複数軸同時に取得することができます。
各軸の動作パラメータは以下のように格納されますので、動作パラメータ格納バッファ（MOTIONPMCS）は軸数分用意し、配列の先頭アドレスを関数へ設定するようにしてください。

```
// 4軸の場合
MOTIONPMCS motion[4];   // 4軸分用意
motion[0]   // X0の動作パラメータ
motion[1]   // Y0の動作パラメータ
motion[2]   // Z0の動作パラメータ
motion[3]   // U0の動作パラメータ

// 8軸の場合
MOTIONPMCS motion[8];   // 8軸分用意
motion[0]   // X0の動作パラメータ
motion[1]   // Y0の動作パラメータ
motion[2]   // Z0の動作パラメータ
```

```
motion[3]   // U0の動作パラメータ
motion[4]   // X1の動作パラメータ
motion[5]   // Y1の動作パラメータ
motion[6]   // Z1の動作パラメータ
motion[7]   // U1の動作パラメータ
```

使用例

ユニットIDが0のユニットから、X0軸とZ0軸の連続動作の動作パラメータを取得します。
motion[0]にX0軸、motion[2]にZ0軸の動作パラメータが格納されます。

C#

```
YduPmcs.MOTIONPMCS[] motion = new YduPmcs.MOTIONPMCS[4];
var result = YduPmcs.GetMotion(0, YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Z0,
YduPmcs.PMC_JOG, motion);
```

VB（.NET2002以降）

```
Dim motion(3) As MOTIONPMCS
Dim result As Integer = YduPmcsGetMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motion)
```

VB6.0/VBA

```
Dim result As Long
Dim motion(3) As MOTIONPMCS
result = YduPmcsGetMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motion(0))
```

C++/CLI

```
MOTIONPMCS motion[4];
int result = YduPmcsGetMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motion);
```

C/C++

```
MOTIONPMCS motion[4];
INT result = YduPmcsGetMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motion);
```

YduPmcsGetMotionEx

機能

速度倍率を含めた動作パラメータを取得します。

> **Note**
>
> 速度倍率については、動作パラメータ計算方法を参照してください。

書式

```
INT YduPmcsGetMotionEx(
    WORD unitId,
    WORD axis,
    WORD moveMode,
    MOTIONEXPMCS* motionEx
);

// 動作パラメータ構造体
typedef struct {
    WORD wSpeedRate;
    WORD wAccMode;
    DWORD dwLowSpeed;
    DWORD dwSpeed;
    WORD wAccTime;
    LONG lStep;
} MOTIONEXPMCS, *PMOTIONEXPMCS;
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

動作パラメータを読み込む軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

moveMode

パラメータを取得する動作モードを指定します。

定義	値	動作モード
PMC_JOG	0	連続動作
PMC_ORG	1	原点復帰動作
PMC_PTP	2	位置決め動作

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

motionEx

動作パラメータを格納するバッファへのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	MOTIONEXPMCS	MOTIONEXPMCS	MOTIONEXPMCS	MOTIONEXPMCS*	MOTIONEXPMCS*

wSpeedRate

速度倍率

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

wAccMode

加減速モード

定義	値	加速モード
PMC_ACC_NORMAL	0	直線加減速
PMC_ACC_SIN	1	S字加減速

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

dwLowSpeed

起動時速度 [PPS]

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	uint	Integer	Long	unsigned long	DWORD

dwSpeed

移動速度 [PPS]

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	uint	Integer	Long	unsigned long	DWORD

wAccTime

加減速時間 [msec]

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

lStep

移動方向・移動パルス数
動作モードによって値が異なります。

動作モードが「連続動作（PMC_JOG）」または「原点復帰動作（PMC_ORG）」の場合

定義	値	移動方向
PMC_DIR_CW	1	+方向
PMC_DIR_CCW	-1	-方向

動作モードが「位置決め動作（PMC_PTP）」の場合

値（範囲）	設定内容
-16,777,215 ～ +16,777,215	移動パルス数

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	long	LONG

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

速度及び加減速時間の値はYduPmcsSetMotion関数またはYduPmcsSetMotionEx関数で設定した値と異なる場合がありますが、ドライバ内での計算による誤差によるものです。
（動作パラメータ計算方法を参照してください）

動作パラメータを複数軸同時に取得することができます。
各軸の動作パラメータは以下のように格納されますので、動作パラメータ格納バッファ（MOTIONEXPMCS）は軸数分用意し、配列の先頭アドレスを関数へ設定するようにしてください。

```
// 4軸の場合
MOTIONEXPMCS motionEx[4];   // 4軸分用意
motionEx[0] // X0の動作パラメータ
motionEx[1] // Y0の動作パラメータ
motionEx[2] // Z0の動作パラメータ
motionEx[3] // U0の動作パラメータ

// 8軸の場合
MOTIONEXPMCS motionEx[8];   // 8軸分用意
motionEx[0] // X0の動作パラメータ
motionEx[1] // Y0の動作パラメータ
motionEx[2] // Z0の動作パラメータ
motionEx[3] // U0の動作パラメータ
motionEx[4] // X1の動作パラメータ
motionEx[5] // Y1の動作パラメータ
motionEx[6] // Z1の動作パラメータ
motionEx[7] // U1の動作パラメータ
```

使用例

ユニットIDが0のユニットから、X0軸とZ0軸の連続動作の動作パラメータを取得します。
motionEx[0]にX0軸、motionEx[2]にZ0軸の動作パラメータが格納されます。

C#

```
YduPmcs.MOTIONEXPMCS[] motionEx = new YduPmcs.MOTIONEXPMCS[4];
var result = YduPmcs.GetMotionEx(0, YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Z0,
YduPmcs.PMC_JOG, motionEx);
```

VB（.NET2002以降）

```
Dim motionEx(3) As MOTIONEXPMCS
Dim result As Integer = YduPmcsGetMotionEx(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motionEx)
```

VB6.0/VBA

```
Dim result As Long
Dim motionEx(3) As MOTIONEXPMCS
result = YduPmcsGetMotionEx(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motionEx(0))
```

C++/CLI

```
MOTIONEXPMCS motionEx[4];
int result = YduPmcsGetMotionEx(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motionEx);
```

C/C++

```
MOTIONEXPMCS motionEx[4];
INT result = YduPmcsGetMotionEx(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_JOG, motionEx);
```

YduPmcsGetCounter

機能

現在の残パルスカウンタ値を取得します。

カウンタについて

書式

```
INT YduPmcsGetCounter(
    WORD unitId,
    WORD axis,
    DWORD* counter
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

残パルスカウンタ値を読み込む軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸

定義	値	軸
PMC_AXIS_Z1	0x40	Z1軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

counter

残パルスカウンタ値を格納するバッファへのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out uint	Integer	Long	unsigned long*	DWORD*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットから、X0軸の残パルスカウンタ値を読み込みます。

C#

```
var result = YduPmcs.GetCounter(0, YduPmcs.PMC_AXIS_X0, out var counter);
```

VB（.NET2002以降）

```
Dim counter As Integer
Dim result As Integer = YduPmcsGetCounter(0, PMC_AXIS_X0, counter)
```

VB6.0/VBA

```
Dim result As Long
Dim counter As Long
result = YduPmcsGetCounter(0, PMC_AXIS_X0, counter)
```

C++/CLI

```
unsigned long counter;
int result = YduPmcsGetCounter(0, PMC_AXIS_X0, &counter);
```

C/C++

```
DWORD counter;
INT result = YduPmcsGetCounter(0, PMC_AXIS_X0, &counter);
```

YduPmcsGetPosition

> **Note**
>
> 本関数は、シリアルNo.299999以下のユニットでは使用できません。
> また、Ver.3.01以下のドライバでは使用できません。

機能

現在位置を取得します。

書式

```
INT YduPmcsGetPosition(
    WORD unitId,
    WORD axis,
    INT* position
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

現在位置を取得する軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸

定義	値	軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

position

現在位置を格納するバッファへのポインタを指定します。

値の範囲は、-8,388,608～+8,388,607です。
-8,388,608からの-方向への移動で+8,388,607になり、+8,388,607からの+方向への移動で-8,388,608になります。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットから、X0軸の現在位置を読み込みます。

C#

```
var result = YduPmcs.GetPosition(0, YduPmcs.PMC_AXIS_X0, out var position);
```

VB（.NET2002以降）

```
Dim position As Integer
Dim result As Integer = YduPmcsGetPosition(0, PMC_AXIS_X0, position)
```

VB6.0/VBA

```
Dim result As Long
Dim position As Long
result = YduPmcsGetPosition(0, PMC_AXIS_X0, position)
```

C++/CLI

```
int position;
int result = YduPmcsGetPosition(0, PMC_AXIS_X0, &position);
```

C/C++

```
INT position;
INT result = YduPmcsGetPosition(0, PMC_AXIS_X0, &position);
```

YduPmcsGetSpeed

機能

現在の速度データを読み込みます。

書式

```
INT YduPmcsGetSpeed(
    WORD unitId,
    WORD axis,
    WORD* speed
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

速度データを読み込む軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

speed

速度データを格納するバッファへのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out ushort	Short	Integer	unsigned short*	WORD*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットから、X0軸の速度データを読み込みます。

C#

```
var result = YduPmcs.GetSpeed(0, YduPmcs.PMC_AXIS_X0, out var speed);
```

VB（.NET2002以降）

```
Dim speed As Short
Dim result As Integer = YduPmcsGetSpeed(0, PMC_AXIS_X0, speed)
```

VB6.0/VBA

```
Dim result As Long
Dim speed As Integer
result = YduPmcsGetSpeed(0, PMC_AXIS_X0, speed)
```

C++/CLI

```
unsigned short speed;
int result = YduPmcsGetSpeed(0, PMC_AXIS_X0, &speed);
```

C/C++

```
WORD speed;
INT result = YduPmcsGetSpeed(0, PMC_AXIS_X0, &speed);
```

YduPmcsGetStatus

機能

各ステータスを取得します。

書式

```
INT YduPmcsGetStatus(
    WORD unitId,
    WORD axis,
    WORD mode,
    WORD* status
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

ステータスを読み込む軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

mode

取得する項目を指定します。

値	取得する項目
PMC_BUSY	動作状態
PMC_SENSOR_STATE	センサ入力状態

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

status

ステータスを格納するバッファへのポインタを指定します。
取得する項目（mode）によって値が異なります。

- wMode : PMC_BUSY

値	内容
0	モーター停止中
1	モーター動作中

- wMode : PMC_SENSOR_STATE

bit	7	6	5	4	3	2	1	0
信号	-	STP	STA	ORG	+SD	-SD	+EL	-EL

各ビットの値	内容
0	入力無し
1	入力有り

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out ushort	Short	Integer	unsigned short*	WORD*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットから、X0軸の動作状態を取得します。

C#

```
var result = YduPmcs.GetStatus(0, YduPmcs.PMC_AXIS_X0, YduPmcs.PMC_BUSY, out var status);
```

VB（.NET2002以降）

```
Dim status As Short
Dim result As Integer = YduPmcsGetStatus(0, PMC_AXIS_X0, PMC_BUSY, status)
```

VB6.0/VBA

```
Dim result As Long
Dim status As Integer
result = YduPmcsGetStatus(0, PMC_AXIS_X0, PMC_BUSY, status)
```

C++/CLI

```
unsigned short status;
int result = YduPmcsGetStatus(0, PMC_AXIS_X0, PMC_BUSY, &status);
```

C/C++

```
WORD status;
INT result = YduPmcsGetStatus(0, PMC_AXIS_X0, PMC_BUSY, &status);
```

YduPmcsStartMotion

機能

モーター動作を開始します。

連続動作について
原点復帰動作について
位置決め動作について

書式

```
INT YduPmcsStartMotion(
    WORD unitId,
    WORD axis,
    WORD startMode,
    WORD moveMode
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

動作を開始する軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸

定義	値	軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

startMode

起動モードを指定します。

定義	値	起動モード
PMC_ACC	0	加減速起動
PMC_CONST	1	一定速起動

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

moveMode

動作モードを指定します。

定義	値	動作モード
PMC_JOG	0	連続動作
PMC_ORG	1	原点復帰動作
PMC_PTP	2	位置決め動作

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットの、X0軸とZ0軸を加減速起動で連続動作を開始します。

C#

```
var result = YduPmcs.StartMotion(0, YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Z0,
YduPmcs.PMC_ACC, YduPmcs.PMC_JOG);
```

VB（.NET2002以降）

```
Dim result As Integer = YduPmcsStartMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_ACC, PMC_JOG)
```

VB6.0/VBA

```
Dim result As Long
result = YduPmcsStartMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_ACC, PMC_JOG)
```

C++/CLI

```
int result = YduPmcsStartMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_ACC, PMC_JOG);
```

C/C++

```
INT result = YduPmcsStartMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_ACC, PMC_JOG);
```

YduPmcsStopMotion

機能

モーター動作を停止します。

書式

```
INT YduPmcsStopMotion(
    WORD unitId,
    WORD axis,
    WORD stopMode
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

モーター動作を停止する軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

stopMode

停止モードを指定します。

定義	値	停止モード
PMC_DEC_STOP	0	減速停止
PMC_IMMEDIATE_STOP	1	即停止

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットの、X0軸とZ0軸のモーター動作を即停止します。

C#

```
var result = YduPmcs.StopMotion(0, YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Z0,
YduPmcs.PMC_IMMEDIATE_STOP);
```

VB（.NET2002以降）

```
Dim result As Integer = YduPmcsStopMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_IMMEDIATE_STOP)
```

VB6.0/VBA

```
Dim result As Long
result = YduPmcsStopMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_IMMEDIATE_STOP)
```

C++/CLI

```
int result = YduPmcsStopMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_IMMEDIATE_STOP);
```

C/C++

```
INT result = YduPmcsStopMotion(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_IMMEDIATE_STOP);
```

YduPmcsChangeSpeed

機能

動作中に速度の変更をします。

書式

```
INT YduPmcsChangeSpeed(
    WORD unitId,
    WORD axis,
    DWORD speed
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

速度を変更する軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

speed

速度を指定します。
現在選択されている速度倍率によって設定範囲は異なります。
速度倍率については動作パラメータ計算方法を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	uint	Integer	Long	unsigned long	DWORD

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットの、X0軸の速度を1000PPSに変更します。

C#

```
var result = YduPmcs.ChangeSpeed(0, YduPmcs.PMC_AXIS_X0, 1000);
```

VB（.NET2002以降）

```
Dim result As Integer = YduPmcsChangeSpeed(0, PMC_AXIS_X0, 1000)
```

VB6.0/VBA

```
Dim result As Long
result = YduPmcsChangeSpeed(0, PMC_AXIS_X0, 1000)
```

C++/CLI

```
int result = YduPmcsChangeSpeed(0, PMC_AXIS_X0, 1000);
```

C/C++

```
INT result = YduPmcsChangeSpeed(0, PMC_AXIS_X0, 1000);
```

関数一覧

モーターコントロールLSIのコマンドやレジスタに対して、書き込み・読み込みする場合に使用する関数群です。
（通常はモーター制御関数を使用しますが、モーター制御関数群にて動作を実現できない場合に本関数群を使用してください。）
各コマンド・レジスタ・ステータスの詳細についてはコントロールLSIのマニュアルを参照してください。

関数	機能
YduPmcsWriteCmdBuf	モーターコントロールLSIのコマンドバッファへコマンドを書き込みます。
YduPmcsWriteRegister	モーターコントロールLSIのレジスタへデータを書き込みます。
YduPmcsReadRegister	モーターコントロールLSIのレジスタからデータを読み込みます。
YduPmcsReadStatus0	モーターコントロールLSIのステータス0を読み込みます。
YduPmcsReadStatus1	モーターコントロールLSIのステータス1を読み込みます。
YduPmcsReadStatus2	モーターコントロールLSIのステータス2を読み込みます。
YduPmcsReadStatus3	モーターコントロールLSIのステータス3を読み込みます。

YduPmcsWriteCmdBuf

機能

モーターコントロールLSIへコマンドを書き込みます。

書式

```
INT YduPmcsWriteCmdBuf(
    WORD unitId,
    WORD axis,
    BYTE command
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

コマンドを書き込む軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

command

書き込むコマンド値を指定します。

> **Note**
>
> 設定値の詳細は、モーターコントロールLSIのマニュアルを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	byte	Byte	Byte	unsigned char	BYTE

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットの、X0軸とZ0軸のコマンドバッファへ0xE0を書き込みます。

C#

```
var result = YduPmcs.WriteCmdBuf(0, YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Z0, 0xE0);
```

VB（.NET2002以降）

```
Dim result As Integer = YduPmcsWriteCmdBuf(0, PMC_AXIS_X0 + PMC_AXIS_Z0, &HE0)
```

VB6.0/VBA

```
Dim result As Long
result = YduPmcsWriteCmdBuf(0, PMC_AXIS_X0 + PMC_AXIS_Z0, &HE0)
```

C++/CLI

```
result = YduPmcsWriteCmdBuf(0, PMC_AXIS_X0 + PMC_AXIS_Z0, 0xE0);
```

C/C++

```
INT result = YduPmcsWriteCmdBuf(0, PMC_AXIS_X0 + PMC_AXIS_Z0, 0xE0);
```

YduPmcsWriteRegister

機能

モーターコントロールLSIへレジスタデータを書き込みます。

書式

```
INT YduPmcsWriteRegister(
    WORD unitId,
    WORD axis,
    BYTE byRegNo,
    DWORD dwData
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

レジスタデータを書き込む軸を指定します。
複数の軸を指定することができます。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

byRegNo

データを書き込むレジスタ番号を指定します。

定義	値	書き込むレジスタ
PMC_REG_0	0	R0
PMC_REG_1	1	R1
PMC_REG_2	2	R2
PMC_REG_3	3	R3
PMC_REG_4	4	R4
PMC_REG_5	5	R5
PMC_REG_6	6	R6
PMC_REG_7	7	R7
PMC_REG_RMV（※）	16	RMV
PMC_REG_RFL（※）	17	RFL
PMC_REG_RFH（※）	18	RFH
PMC_REG_RUD（※）	19	RUD
PMC_REG_RMG（※）	20	RMG
PMC_REG_RDP（※）	21	RDP
PMC_REG_RIDL（※）	22	RIDL

定義	値	書き込むレジスタ
PMC_REG_RENV（※）	23	RENV
PMC_REG_RCUN（※）	24	RCUN
PMC_REG_RIOP（※）	26	RIOP

※ RMV以降のレジスタについて

シリアルNo.299999以下のユニットでは使用できません。
また、Ver.3.01以下のドライバでは使用できません。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	byte	Byte	Byte	unsigned char	BYTE

dwData

レジスタへ書き込むデータを指定します。

Note

データについては、モーターコントロールLSIのマニュアルを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	uint	Integer	Long	unsigned long	DWORD

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットの、X0軸とZ0軸のレジスタ1へ0xFFを書き込みます。

C#

```
var result = YduPmcs.WriteRegister(0, YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Z0, 1, 0xFF);
```

VB（.NET2002以降）

```
Dim result As Integer = YduPmcsWriteRegister(0, PMC_AXIS_X0 + PMC_AXIS_Z0, 1, &HFF)
```

VB6.0/VBA

```
Dim result As Long
result = YduPmcsWriteRegister(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_REG_1, &HFF)
```

C++/CLI

```
int result = YduPmcsWriteRegister(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_REG_1, 0xFF);
```

C/C++

```
INT result = YduPmcsWriteRegister(0, PMC_AXIS_X0 + PMC_AXIS_Z0, PMC_REG_1, 0xFF);
```

YduPmcsReadRegister

機能

モーターコントロールLSIのレジスタデータを読み込みます。

書式

```
INT YduPmcsReadRegister(
    WORD unitId,
    WORD axis,
    BYTE byRegNo,
    DWORD* registerData
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

レジスタデータを読み込む軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

byRegNo

データを読み込むレジスタ番号を指定します。

定義	値	読み込むレジスタ
PMC_REG_0	0	R0
PMC_REG_1	1	R1
PMC_REG_2	2	R2
PMC_REG_3	3	R3
PMC_REG_4	4	R4
PMC_REG_5	5	R5
PMC_REG_6	6	R6
PMC_REG_7	7	R7
PMC_REG_RMV（※）	16	RMV
PMC_REG_RFL（※）	17	RFL
PMC_REG_RFH（※）	18	RFH
PMC_REG_RUD（※）	19	RUD
PMC_REG_RMG（※）	20	RMG
PMC_REG_RDP（※）	21	RDP
PMC_REG_RIDL（※）	22	RIDL

定義	値	読み込むレジスタ
PMC_REG_RENV（※）	23	RENV
PMC_REG_RCUN（※）	24	RCUN
PMC_REG_RSTS（※）	25	RSTS
PMC_REG_RIOP（※）	26	RIOP

※ RMV以降のレジスタについて

シリアルNo.299999以下のユニットでは使用できません。
また、Ver.3.01以下のドライバでは使用できません。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	byte	Byte	Byte	unsigned char	BYTE

registerData

レジスタデータを格納するバッファへのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out uint	Integer	Long	unsigned long*	DWORD*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットから、X0軸のレジスタ1のデータを読み込みます。

C#

```
var result = YduPmcs.ReadRegister(0, YduPmcs.PMC_AXIS_X0, 1, out var registerData);
```

VB（.NET2002以降）

```
Dim registerData As Integer
Dim result As Integer = YduPmcsReadRegister(0, PMC_AXIS_X0, 1, registerData)
```

VB6.0/VBA

```
Dim result As Long
Dim registerData As Long
result = YduPmcsReadRegister(0, PMC_AXIS_X0, PMC_REG_1, registerData)
```

C++/CLI

```
unsigned long registerData;
int result = YduPmcsReadRegister(0, PMC_AXIS_X0, PMC_REG_1, &registerData);
```

C/C++

```
DWORD registerData;
INT result = YduPmcsReadRegister(0, PMC_AXIS_X0, PMC_REG_1, &registerData);
```

YduPmcsReadStatus0

機能

モーターコントロールLSIのステータス0を読み込みます。

書式

```
INT YduPmcsReadStatus0(
    WORD unitId,
    WORD axis,
    BYTE* status
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

ステータスを読み込む軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

status

ステータスデータを格納するバッファへのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out byte	Byte	Byte	unsigned char*	BYTE*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットから、X0軸のステータス0を読み込みます。

C#

```
var result = YduPmcs.ReadStatus0(0, YduPmcs.PMC_AXIS_X0, out var status);
```

VB（.NET2002以降）

```
Dim status As Byte
Dim result As Integer = YduPmcsReadStatus0(0, PMC_AXIS_X0, status)
```

VB6.0/VBA

```
Dim result As Long
Dim status As Byte
result = YduPmcsReadStatus0(0, PMC_AXIS_X0, status)
```

C++/CLI

```
unsigned char status;
int result = YduPmcsReadStatus0(0, PMC_AXIS_X0, &status);
```

C/C++

```
BYTE status;
INT result = YduPmcsReadStatus0(0, PMC_AXIS_X0, &status);
```

YduPmcsReadStatus1

機能

モーターコントロールLSIのステータス1を読み込みます。

書式

```
INT YduPmcsReadStatus1(
    WORD unitId,
    WORD axis,
    BYTE* status
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

ステータスを読み込む軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

status

ステータスデータを格納するバッファへのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out byte	Byte	Byte	unsigned char*	BYTE*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットから、X0軸のステータス1を読み込みます。

C#

```
var result = YduPmcs.ReadStatus1(0, YduPmcs.PMC_AXIS_X0, out var status);
```

VB（.NET2002以降）

```
Dim status As Byte
Dim result As Integer = YduPmcsReadStatus1(0, PMC_AXIS_X0, status)
```

VB6.0/VBA

```
Dim result As Long
Dim status As Byte
result = YduPmcsReadStatus1(0, PMC_AXIS_X0, status)
```

C++/CLI

```
unsigned char status;
int result = YduPmcsReadStatus1(0, PMC_AXIS_X0, &status);
```

C/C++

```
BYTE status;
INT result = YduPmcsReadStatus1(0, PMC_AXIS_X0, &status);
```

YduPmcsReadStatus2

機能

モーターコントロールLSIのステータス2を読み込みます。

書式

```
INT YduPmcsReadStatus2(
    WORD unitId,
    WORD axis,
    BYTE* status
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

ステータスを読み込む軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

status

ステータスデータを格納するバッファへのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out byte	Byte	Byte	unsigned char*	BYTE*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットから、X0軸のステータス2を読み込みます。

C#

```
var result = YduPmcs.ReadStatus2(0, YduPmcs.PMC_AXIS_X0, out var status);
```

VB（.NET2002以降）

```
Dim status As Byte
Dim result As Integer = YduPmcsReadStatus2(0, PMC_AXIS_X0, status)
```

VB6.0/VBA

```
Dim result As Long
Dim status As Byte
result = YduPmcsReadStatus2(0, PMC_AXIS_X0, status)
```

C++/CLI

```
unsigned char status;
int result = YduPmcsReadStatus2(0, PMC_AXIS_X0, &status);
```

C/C++

```
BYTE status;
INT result = YduPmcsReadStatus2(0, PMC_AXIS_X0, &status);
```

YduPmcsReadStatus3

機能

モーターコントロールLSIのステータス3を読み込みます。

書式

```
INT YduPmcsReadStatus3(
    WORD unitId,
    WORD axis,
    BYTE* status
);
```

パラメータ

unitId

ユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

axis

ステータスを読み込む軸を指定します。
複数の軸を指定することはできません。

定義	値	軸
PMC_AXIS_X0	0x01	X0軸
PMC_AXIS_Y0	0x02	Y0軸
PMC_AXIS_Z0	0x04	Z0軸
PMC_AXIS_U0	0x08	U0軸
PMC_AXIS_X1	0x10	X1軸
PMC_AXIS_Y1	0x20	Y1軸
PMC_AXIS_Z1	0x40	Z1軸

定義	値	軸
PMC_AXIS_U1	0x80	U1軸

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

status

ステータスデータを格納するバッファへのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	out byte	Byte	Byte	unsigned char*	BYTE*

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットから、X0軸のステータス3を読み込みます。

C#

```
var result = YduPmcs.ReadStatus3(0, YduPmcs.PMC_AXIS_X0, out var status);
```

VB（.NET2002以降）

```
Dim status As Byte
Dim result As Integer = YduPmcsReadStatus3(0, PMC_AXIS_X0, status)
```

VB6.0/VBA

```
Dim result As Long
Dim status As Byte
result = YduPmcsReadStatus3(0, PMC_AXIS_X0, status)
```

C++/CLI

```
unsigned char status;
int result = YduPmcsReadStatus3(0, PMC_AXIS_X0, &status);
```

C/C++

```
BYTE status;
INT result = YduPmcsReadStatus3(0, PMC_AXIS_X0, &status);
```

関数一覧

モーター制御ボードの汎用入出力やデジタル入出力ボード（デジタル入出力ボードと組み合わせたユニットの場合）を制御します。

関数	機能
YduDioInput	入力端子の読み込みをおこないます。
YduDioOutput	出力端子の制御をおこないます。
YduDioOutputStatus	出力の状態を読み込みます。

YduDioInput

機能

任意の点数の入力端子の状態を読み込みます。

書式

```
INT YduDioInput(
    WORD unitId,
    BYTE* inputData,
    WORD start,
    WORD count
);
```

パラメータ

unitId

入力読み込みをおこなうユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

inputData

入力データを格納するバッファへのポインタを指定します。
関数が正常に実行されると、入力データが格納されます。

入力データ	状態
0	OFF
1	ON

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	byte	Byte	Byte	unsigned char*	BYTE*

start

入力開始番号(0～)を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

count

入力の読み込みをおこなう数を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットのIN0からIN7の入力端子の状態を読み込みます。
データはIN0から順にデータバッファへ格納されます。

C#

```
var inputData = new byte[8];
var result = YduDio.Input(0, inputData, 0, 8);
```

VB（.NET2002以降）

```
Dim inputData(7) As Byte
Dim result As Integer = YduDioInput(0, inputData, 0, 8)
```

VB6.0/VBA

```
Dim result As Long
Dim inputData(7) As Byte
result = YduDioInput(0, inputData(0), 0, 8)
```

C++/CLI

```
unsigned char inputData[8];
int result = YduDioInput(0, inputData, 0, 8);
```

C/C++

```
BYTE inputData[8];
INT result = YduDioInput(0, inputData, 0, 8);
```

YduDioOutput

機能

任意の点数の出力端子を制御します。

書式

```
INT YduDioOutput(
    WORD unitId,
    BYTE* outputData,
    WORD start,
    WORD count
);
```

パラメータ

unitId

出力をおこなうユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

outputData

ユニットへ出力するデータを格納したバッファへのポインタを指定します。

値	出力するデータ
0	OFF
1	ON
2	現在の状態を保持

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	byte	Byte	Byte	unsigned char*	BYTE*

start

出力開始番号（0～）を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

count

出力するデータ数を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

ユニットIDが0のユニットへ、OUT0に0、OUT1に1、OUT2に0、OUT3に1、OUT4は現在の出力状態、OUT5は現在の出力状態、OUT6に0、OUT7に1を出力する。

C#

```
var outputData = new byte[] { 0, 1, 0, 1, 2, 2, 0, 1 };
var result = YduDio.Output(0, outputData, 0, 8);
```

VB（.NET2002以降）

```
Dim outputData() As Byte = {0, 1, 0, 1, 2, 2, 0, 1}
Dim result As Integer = YduDioOutput(0, outputData, 0, 8)
```

VB6.0/VBA

```
Dim result As Long
Dim outputData(7) As Byte
outputData(0) = 0
outputData(1) = 1
outputData(2) = 0
outputData(3) = 1
outputData(4) = 2
outputData(5) = 2
```

```
outputData(6) = 0
outputData(7) = 1
result = YduDioOutput(0, outputData(0), 0, 8)
```

C++/CLI

```
unsigned char outputData[8] = { 0, 1, 0, 1, 2, 2, 0, 1 };
int result = YduDioOutput(0, outputData, 0, 8);
```

C/C++

```
BYTE outputData[8] = { 0, 1, 0, 1, 2, 2, 0, 1 };
INT result = YduDioOutput(0, outputData, 0, 8);
```

YduDioOutputStatus

機能

任意の点数のデジタル出力状態を読み込みます。
現在のデジタル出力の状態を知りたい場合に使用します。

書式

```
INT YduDioOutputStatus(
    WORD unitId,
    BYTE* status,
    WORD start,
    WORD count
);
```

パラメータ

unitId

出力読み込みを行うユニットのID番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

status

出力データを格納するバッファへのポインタを指定します。
関数が正常に実行されると、出力データが格納されます。

値	出力データ
0	OFF
1	ON

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	byte	Byte	Byte	unsigned char*	BYTE*

start

出力の読み込みを開始する出力番号（0～）を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

count

出力の読み込みを行う出力点数を指定します。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	ushort	Short	Integer	unsigned short	WORD

戻り値

関数が正常に終了した場合は0（YDU_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C#	VB（.NET2002以降）	VB6.0/VBA	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

- 例1
  ユニットIDが0のユニットのOUT0からOUT7の出力状態を読み込みます。
  データはOUT0から順にデータバッファへ格納されます。
- 例2
  ユニットIDが0のユニットのOUT5からOUT7の出力状態を読み込みます。
  データはOUT5から順にデータバッファへ格納されます。

C#

```
// 例1
var outputStatus = new byte[8];
var result = YduDio.OutputStatus(0, outputStatus, 0, 8);

// 例2
var outputStatus = new byte[3];
var result = YduDio.OutputStatus(0, outputStatus, 5, 3);
```

VB（.NET2002以降）

```
' 例1
Dim outputStatus(7) As Byte
Dim result As Integer = YduDioOutputStatus(0, outputStatus, 0, 8)
```

```
' 例2
Dim outputStatus(2) As Byte
Dim result As Integer = YduDioOutputStatus(0, outputStatus, 5, 3)
```

VB6.0/VBA

```
' 例1
Dim result As Long
Dim outputStatus(7) As Byte
result = YduDioOutputStatus(0, outputStatus(0), 0, 8)

' 例2
Dim result As Long
Dim outputStatus(2) As Byte
result = YduDioOutputStatus(0, outputStatus(0), 5, 3)
```

C++/CLI

```
// 例1
unsigned char outputStatus[8];
int result = YduDioOutputStatus(0, outputStatus, 0, 8);

// 例2
unsigned char outputStatus[3];
int result = YduDioOutputStatus(0, outputStatus, 5, 3);
```

C/C++

```
// 例1
BYTE outputStatus[8];
INT result = YduDioOutputStatus(0, outputStatus, 0, 8);

// 例2
BYTE outputStatus[3];
INT result = YduDioOutputStatus(0, outputStatus, 5, 3);
```

動作パラメータ計算方法

速度倍率

コントロールLSI内部で各軸ごとに速度倍率という値を持ち、それを元に出力パルスが作成されます。
速度倍率の設定値により、出力できる速度範囲及び速度設定間隔が決まります。

速度設定範囲最小値 = 速度倍率
速度設定範囲最大値 = 速度倍率 × 8191

高倍率になるほど設定できる速度間隔が粗くなりますので、できるだけ小さな倍率を設定してください。

- 動作パラメータの移動速度（dwSpeed）が8191の倍数（8191, 16382...）の場合

  速度倍率 = dwSpeed / 8191
- 動作パラメータの移動速度（dwSpeed）が8191の倍数でない（8191で割り切れない）場合

  速度倍率 = dwSpeed / 8191 + 1

> **Note**
>
> 動作パラメータに設定された値は速度倍率の値を元に再計算されて設定されます。
> その際、近似値に設定されることがあります。
> （速度設定値は速度倍率の倍数のみとなります）

速度倍率と速度設定範囲の例

速度倍率	速度設定範囲	速度設定間隔
1	1〜8,191 （1, 2, 3 ..... 8190, 8191）	1
2	2〜16,382 （2, 4, 6 ..... 16380, 16382）	2
5	5〜40,955 （5, 10, 15 ..... 40950, 40955）	5
10	10〜81,910 （10, 20, 30 ..... 81900, 81910）	10
20	20〜163,820 （20, 40, 60 ..... 163800, 163820）	20
50	50〜409,550 （50, 100, 150 ..... 409500, 409550）	50

速度倍率	速度設定範囲	速度設定間隔
100	100～819,100 （100, 200, 300 ..... 819000, 819100）	100
200	200～1,638,200 （200, 400, 600 ..... 1638000, 1638200）	200
300	300～2,457,300 （300, 600, 900 ..... 2457000, 2457300）	300

速度設定範囲最小値 = 速度倍率
速度設定範囲最大値 = 速度倍率 × 8,191

> **Note**
>
> 「ユニットのシリアルNo.が299999以下」または「ドライバがVer.3.01以下」の場合、最高速度は409,550[PPS]です。
> したがって、速度倍率は50以下を使用してください。

起動時速度（dwLowSpeed）

1. 動作パラメータに設定された起動時速度からコントロールLSIの設定値を計算します。

   起動時速度設定レジスタ = dwLowSpeed / 速度倍率

2. コントロールLSIに設定された値から再計算された値が実際の起動時速度になります。

   起動時速度 = 起動時速度設定レジスタ × 速度倍率

計算例

設定項目	値
速度倍率	3
起動時速度	100

コントロールLSIの設定値
100 / 3 = 33.333 ≒ 33

実際の起動時速度
33 × 3 = 99[PPS]

移動速度（dwSpeed）

1. 動作パラメータに設定された移動速度からコントロールLSIの設定値を計算します。

移動速度設定レジスタ = dwSpeed / 速度倍率

2. コントロールLSIに設定された値から再計算された値が実際の移動速度になります。

   移動速度 = 移動速度設定レジスタ × 速度倍率

計算例

設定項目	値
速度倍率	3
移動速度	20000

コントロールLSIの設定値
20000 / 3 = 6666.666 ≒ 6666

実際の移動速度
6666 × 3 = 19998[PPS]

加減速時間（wAccTime）

直線加減速（accModeがPMC_ACC_NORMAL）

1. 動作パラメータに設定された加減速時間からコントロールLSIの設定値を計算します。

   加減速時間設定レジスタ（※） = 加減速時間 / 1,000 × 4,915,200 / (移動速度設定レジスタ - 起動時速度設定レジスタ)

   > ※「ユニットのシリアルNo.が300000以上」かつ「ドライバがVer.3.10.0以上」の場合
   >
   > 加減速時間設定レジスタの設定範囲は2～65,535です。
   > 2未満になる場合は2に、65,535を超える場合は65,535に丸められます。

   > ※「ユニットのシリアルNo.が299999以下」または「ドライバがVer.3.01以下」の場合
   >
   > 加減速時間設定レジスタの設定範囲は2～1,023です。
   > 2未満になる場合は2に、1,023を超える場合は1,023に丸められます。

2. コントロールLSIに設定された値から再計算された値が実際の加減速時間になります。

   加減速時間 = (移動速度設定レジスタ - 起動時速度設定レジスタ) × 加減速時間設定レジスタ / 4,915,200 × 1,000

計算例

設定項目	値
起動時速度レジスタ	33
移動速度レジスタ	6,666
加減速時間[msec]	500

コントロールLSIの設定値
500 / 1,000 × 4,915,200 / (6,666 - 33) = 370.511 ≒ 370

実際の加減速時間
(6,666 - 33) × 370 / 4,915,200 × 1,000 ≒ 499[msec]

S字加減速（accModeがPMC_ACC_SIN）

1. 動作パラメータに設定された加減速時間からコントロールLSIの設定値を計算します。

   加減速時間設定レジスタ（※） = 加減速時間 / 1,000 × 4,915,200 / ((移動速度設定レジスタ - 起動時速度設定レジスタ) × 2)

   > ※「ユニットのシリアルNo.が300000以上」かつ「ドライバがVer.3.10.0以上」の場合
   >
   > 加減速時間設定レジスタの設定範囲は2～65,535です。
   > 2未満になる場合は2に、65,535を超える場合は65,535に丸められます。

   > ※「ユニットのシリアルNo.が299999以下」または「ドライバがVer.3.01以下」の場合
   >
   > 加減速時間設定レジスタの設定範囲は2～1,023です。
   > 2未満になる場合は2に、1,023を超える場合は1,023に丸められます。

2. コントロールLSIに設定された値から再計算された値が実際の加減速時間になります。

   加減速時間 = (移動速度設定レジスタ - 起動時速度設定レジスタ) × 加減速時間設定レジスタ × 2 / 4,915,200 × 1,000

計算例

設定項目	値
起動時速度レジスタ	33
移動速度レジスタ	6,666

設定項目	値
加減速時間[msec]	500

コントロールLSIの設定値
500 / 1,000 × 4,915,200 / ((6,666 - 33) × 2) = 185.255 ≒ 185

実際の加減速時間
(6,666 - 33) × 185 × 2 / 4,915,200 × 1,000 ≒ 499[msec]

アクセス時間

デジタル入出力に要する時間は以下の通りです。
（コンピュータの性能やCPUの負荷状況によって変わります）

デジタル入出力

入力関数（YduDioInput）及び出力関数（YduDioOutput）を10000回実行した場合の1回当たりの平均所要時間は以下の通りです。

USB2.0

入力

入力点数	10,000回実行時間	1回当たりの実行時間	使用コンピュータ
1	1010msec	101μsec	A
1	2510～2530msec	251～253μsec	B
1	5000msec	500μsec	C
192	7000msec	700μsec	A
192	7500～7510msec	750～751μsec	B
192	8700msec	870μsec	C

出力

出力点数	10,000回実行時間	1回当たりの実行時間	使用コンピュータ
1	1010msec	101μsec	A
1	2510～2530msec	251～253μsec	B
1	5000msec	500μsec	C
192	7200msec	720μsec	A
192	7500～7510msec	750～751μsec	B
192	8800msec	880μsec	C

USB1.1（USB1.1ハブ使用）

> ℹ 入力関数と出力関数の平均所要時間は同じです

入力または出力点数	10,000回実行時間	1回当たりの実行時間	使用コンピュータ
1	30000msec	3msec	C
192	30000msec	3msec	C

使用コンピュータ

使用コンピュータ	OS	CPU
A	Windows 10	Intel Core i7-9700 3.00GHz
B	Windows 7	Intel Core 2 Quad Q8400 2.66GHz
C	Windows XP	Intel CeleronM 1.5GHz

C/C++

開発環境の設定

Visual C++ .NET2002以降

1. 以下のファイルをプロジェクトフォルダにコピーします。

   YduApi.h
   YduPmcSApi.h
   YduResult.h
   Ydu.lib

2. YduApi.h, YduPmcSApi.h, YduResult.hをプロジェクトに追加します。
3. Ydu.libを以下の手順でプロジェクトに追加します
   メニューの[プロジェクト]-[プロパティ]を選択し、プロパティページのダイアログを開きます。
   ダイアログの左ペインで[構成プロパティ]-[リンカ]-[入力]を選択します。
   右ペインの[追加の依存ファイル]にYdu.libと入力します。
4. ソースファイルにYduApi.h, YduPmcSApi.h, YduResult.hをインクルードします
   （下記プログラム例を参照して下さい）

Visual C++ 6.0

1. 以下のファイルをプロジェクトフォルダにコピーします。

   YduApi.h
   YduPmcSApi.h
   YduResult.h
   Ydu.lib

2. YduApi.h, YduPmcSApi.h, YduResult.h, Ydu.libをプロジェクトに追加します
3. ソースファイルにYduApi.h, YduPmcSApi.h, YduResult.hをインクルードします
   （下記プログラム例を参照して下さい）

プログラム例

```
#include <windows.h>
#include <stdio.h>
#include "YduApi.h"
#include "YduPmcSApi.h"
#include "YduResult.h"

void main()
{
    int result;
    WORD axis;
    MOTIONPMCS motion[4];
    BOOL resultClose;

    // IDが0に設定されているPMC-S4/00/00A-Uをオープンします
    result = YduOpen(0, "PMC-S4/00/00A-U");
    if(result != YDU_RESULT_SUCCESS) {
```

```
        printf("オープンできません\n");
        return;
    }

    // オンで検知するセンサを接続している場合や、リミットスイッチを接続していない場合はモーターが動作しません
    // その場合は以下の関数を実行してセンサ設定を"オンで検知"に変更してください
    axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0;
    result = YduPmcsSetSensorConfig(0, axis, PMC_LOGIC, 0x1F);

    // X0軸の動作パラメータを設定します
    motion[0].wAccMode = PMC_ACC_NORMAL;
    motion[0].dwLowSpeed = 200;
    motion[0].dwSpeed = 2000;
    motion[0].wAccTime = 300;
    motion[0].lStep = PMC_DIR_CW;
    result = YduPmcsSetMotion(0, PMC_AXIS_X0, PMC_JOG, motion);

    // モーター動作を開始します
    result = YduPmcsStartMotion(0, PMC_AXIS_X0, PMC_ACC, PMC_JOG);

    // モーター動作を停止します
    result = YduPmcsStopMotion(0, PMC_AXIS_X0, PMC_IMMEDIATE_STOP);

    // ユニットをクローズします
    resultClose = YduClose(0);
}
```

C++/CLI

開発環境の設定

1. 以下のファイルをプロジェクトフォルダにコピーします。

   YduApiCLI.h
   YduPmcSApiCLI.h

2. YduApiCLI.h, YduPmcSApiCLI.hをプロジェクトに追加します。

3. ソースファイルにYduApiCLI.h, YduPmcSApiCLI.hをインクルードします。
   （下記プログラム例を参照してください）

4. usingディレクティブを使ってYduCLIを宣言します。

```
using namespace YduCLI;
```

プログラム例

```
#include "stdafx.h"
#include "YduApiCLI.h"
#include "YduPmcSApiCLI.h"

using namespace System;
using namespace YduCLI;

int main(array<System::String ^> ^args)
{
    // IDが0に設定されているPMC-S4/00/00A-Uをオープンします
    const unsigned short UnitId = 0;
    int result = YduOpen(UnitId, "PMC-S4/00/00A-U");
    if (result != YDU_RESULT_SUCCESS) {
        Console::WriteLine("オープンできません");
        return -1;
    }

    // オンで検知するセンサを接続している場合や、リミットスイッチを接続していない場合はモーターが動作しません
    // その場合は以下の関数を実行してセンサ設定を"オンで検知"に変更してください
    unsigned short axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0;
    result = YduPmcsSetSensorConfig(UnitId, axis, PMC_LOGIC, 0x1F);

    // X0軸の動作パラメータを設定します
    MOTIONPMCS motion[4];
    motion[0].wAccMode = PMC_ACC_NORMAL;
    motion[0].dwLowSpeed = 200;
    motion[0].dwSpeed = 2000;
    motion[0].wAccTime = 300;
    motion[0].lStep = PMC_DIR_CW;
    result = YduPmcsSetMotion(UnitId, PMC_AXIS_X0, PMC_JOG, motion);

    // モーター動作を開始します
    result = YduPmcsStartMotion(UnitId, PMC_AXIS_X0, PMC_ACC, PMC_JOG);

    // モーター動作を停止します
    result = YduPmcsStopMotion(UnitId, PMC_AXIS_X0, PMC_IMMEDIATE_STOP);

    // ユニットをクローズします
```

```
    YduClose(UnitId);

    return 0;
}
```

開発環境の設定

プロジェクトにドライバへの参照を追加します。

- Nugetからインストールする場合
  - Y2.UsbPc104.YduCs をインストールします。
- ソフトウェアパックから追加する場合
  - 以下のファイルをプロジェクトフォルダにコピーします。
    - Ydu.cs
    - YduPmcS.cs
  - Ydu.cs, YduPmcS.csをプロジェクトに追加します。

ソースファイルにusing ディレクティブを使ってYduCsを宣言します。

```
using YduCs;
```

プログラム例

```
using YduCs;

static void Main()
{
    // IDが0に設定されているPMC-S4/00/00A-Uをオープンします
    const ushort UnitId = 0;
    var result = Ydu.Open(UnitId, "PMC-S4/00/00A-U");
    if (result != Ydu.YDU_RESULT_SUCCESS)
    {
        Console.WriteLine("オープンできません");
        return;
    }

    // オンで検知するセンサを接続している場合や、リミットスイッチを接続していない場合はモーターが動作しません
    // その場合は以下の関数を実行してセンサ設定を"オンで検知"に変更してください
    ushort axis = YduPmcs.PMC_AXIS_X0 + YduPmcs.PMC_AXIS_Y0 + YduPmcs.PMC_AXIS_Z0 +
YduPmcs.PMC_AXIS_U0;
    result = YduPmcs.SetSensorConfig(UnitId, axis, YduPmcs.PMC_LOGIC, 0x1F);

    // X0軸の動作パラメータを設定します
    var motion = new YduPmcs.MOTIONPMCS[4];
    motion[0].wAccMode = YduPmcs.PMC_ACC_NORMAL;
    motion[0].dwLowSpeed = 200;
    motion[0].dwSpeed = 2000;
    motion[0].wAccTime = 300;
    motion[0].lStep = YduPmcs.PMC_DIR_CW;
    result = YduPmcs.SetMotion(UnitId, YduPmcs.PMC_AXIS_X0, YduPmcs.PMC_JOG, motion);

    // モーター動作を開始します
    result = YduPmcs.StartMotion(UnitId, YduPmcs.PMC_AXIS_X0, YduPmcs.PMC_ACC,
YduPmcs.PMC_JOG);

    // モーター動作を停止します
```

```
    result = YduPmcs.StopMotion(UnitId, YduPmcs.PMC_AXIS_X0, YduPmcs.PMC_IMMEDIATE_STOP);

    // ユニットをクローズします
    Ydu.Close(UnitId);
}
```

VB（.NET2002以降）

開発環境の設定

1. 以下のファイルをプロジェクトフォルダにコピーします。

   YduApi.vb
   YduPmcSApi.vb
   YduResult.vb

2. YduApi.vb, YduPmcSApi.vb, YduResult.vbをプロジェクトに追加します。

プログラム例

```
Dim result As Integer
Dim axis As Short
Dim motion(3) As MOTIONPMCS

'IDが0に設定されているPMC-S4/00/00A-Uをオープンします
result = YduOpen(0, "PMC-S4/00/00A-U")
If result <> YDU_RESULT_SUCCESS Then
    MessageBox.Show("オープンできません", "", MessageBoxButtons.OK, MessageBoxIcon.Stop)
    Exit Sub
End If

'オンで検知するセンサを接続している場合や、リミットスイッチを接続していない場合はモーターが動作しません
'その場合は以下の関数を実行してセンサ設定を"オンで検知"に変更してください
axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0
result = YduPmcsSetSensorConfig(0, axis, PMC_LOGIC, &H1F)

'X0軸の動作パラメータを設定します
motion(0).wAccMode = PMC_ACC_NORMAL
motion(0).dwLowSpeed = 200
motion(0).dwSpeed = 2000
motion(0).wAccTime = 300
motion(0).lStep = PMC_DIR_CW
result = YduPmcsSetMotion(0, PMC_AXIS_X0 + PMC_AXIS_Y0, PMC_JOG, motion)

'モーター動作を開始します
result = YduPmcsStartMotion(0, PMC_AXIS_X0, PMC_ACC, PMC_JOG)

'モーター動作を停止します
result = YduPmcsStopMotion(0, PMC_AXIS_X0, PMC_IMMEDIATE_STOP)

'ユニットをクローズします
YduClose(0)
```

Visual Basic 6.0/VBA

開発環境の設定

VB6.0

1. 以下のファイルをプロジェクトフォルダにコピーします
   YduBaseApi.bas
   YduPmcSApi.bas
2. YduBaseApi.bas, YduPmcSApi.basをプロジェクトに追加します

VBA

1. 以下のファイルをインポートします
   YduBaseApi.bas
   YduPmcSApi.bas

プログラム例

```
Dim result As Long
Dim modelName As String
Dim axis As Integer
Dim motion(3) As MOTIONPMCS
Dim resultClose As Boolean

'IDが0に設定されているPMC-S4/00/00A-Uをオープンします
modelName = "PMC-S4/00/00A-U"
result = YduOpen(0, modelName)
If result <> YDU_RESULT_SUCCESS Then
    MsgBox "オープンできません", vbInformation
    Exit Sub
End If

'オンで検知するセンサを接続している場合や、リミットスイッチを接続していない場合はモーターが動作しません
'その場合は以下の関数を実行してセンサ設定を"オンで検知"に変更してください
axis = PMC_AXIS_X0 + PMC_AXIS_Y0 + PMC_AXIS_Z0 + PMC_AXIS_U0
result = YduPmcsSetSensorConfig(0, axis, PMC_LOGIC, &H1F)

'X0軸の動作パラメータを設定します
motion(0).wAccMode = PMC_ACC_NORMAL
motion(0).dwLowSpeed = 200
motion(0).dwSpeed = 2000
motion(0).wAccTime = 300
motion(0).lStep = PMC_DIR_CW
result = YduPmcsSetMotion(0, PMC_AXIS_X0, PMC_JOG, motion(0))

'モーター動作を開始します
result = YduPmcsStartMotion(0, PMC_AXIS_X0, PMC_ACC, PMC_JOG)

'モーター動作を停止します
result = YduPmcsStopMotion(0, PMC_AXIS_X0, PMC_IMMEDIATE_STOP)

'ユニットをクローズします
resultClose = YduClose(0)
```

C/C++

開発環境の設定

Visual C++ .NET2002以降

1. 以下のファイルをプロジェクトフォルダにコピーします
   YduApi.h
   YduDioApi.h
   YduResult.h
   Ydu.lib
2. YduApi.h, YduDioApi.h, YduResult.hをプロジェクトに追加します
3. Ydu.libを以下の手順でプロジェクトに追加します
   メニューの[プロジェクト]-[プロパティ]を選択し、プロパティページのダイアログを開きます。
   ダイアログの左ペインで[構成プロパティ]-[リンカ]-[入力]を選択します。
   右ペインの[追加の依存ファイル]にYdu.libと入力します。
4. ソースファイルにYduApi.h, YduDioApi.h, YduResult.hをインクルードします
   （下記プログラム例を参照して下さい）

Visual C++ 6.0

1. 以下のファイルをプロジェクトフォルダにコピーします
   YduApi.h
   YduDioApi.h
   YduResult.h
   Ydu.lib
2. YduApi.h, YduDioApi.h, YduResult.h, Ydu.libをプロジェクトに追加します
3. ソースファイルにYduApi.h, YduDioApi.h, YduResult.hをインクルードします
   （下記プログラム例を参照して下さい）

プログラム例

```
#include <windows.h>
#include <stdio.h>
#include "YduApi.h"
#include "YduDioApi.h"
#include "YduResult.h"

void main()
{
    int     result;
    BYTE    inputData[16];
    BYTE    outputData[32];
    BOOL    resultClose;
    int     i;

    // IDが0に設定されているPMC-S4/16/32A-Uをオープンします
    result = YduOpen(0, "PMC-S4/16/32A-U");
    if(result != YDU_RESULT_SUCCESS){
```

```
        printf("オープンできません\n");
        return;
    }

    // IN0～15の入力をおこないます
    result = YduDioInput(0, inputData, 0, 16);
    // 入力データの表示
    for(i = 0; i < 16; i++){
        printf("IN%u : %u\n", i, inputData[i]);
    }

    // OUT0～31の出力をONします
    memset(outputData, 1, 32);
    result = YduDioOutput(0, outputData, 0, 32);

    // ユニットをクローズします
    resultClose = YduClose(0);
}
```

C++/CLI

開発環境の設定

1. 以下のファイルをプロジェクトフォルダにコピーします。

   YduApiCLI.h
   YduDioApiCLI.h

2. YduApiCLI.h, YduDioApiCLI.hをプロジェクトに追加します。
3. ソースファイルにYduApiCLI.h, YduDioApiCLI.hをインクルードします。
   （下記プログラム例を参照してください）
4. usingディレクティブを使ってYduCLIを宣言します。

```
using namespace YduCLI;
```

プログラム例

```
#include "stdafx.h"
#include "YduApiCLI.h"
#include "YduDioApiCLI.h"

using namespace System;
using namespace YduCLI;

int main(array<System::String ^> ^args)
{
    // IDが0に設定されているPMC-S4/16/32A-Uをオープンします
    unsigned short unitId = 0;
    int result = YduOpen(unitId, "PMC-S4/16/32A-U");
    if (result != YDU_RESULT_SUCCESS) {
        Console::WriteLine("オープンできません");
        return -1;
    }

    // IN0～15の入力をおこないます
    const int inputCount = 16;
    unsigned char inputData[inputCount];
    result = YduDioInput(unitId, inputData, 0, inputCount);
    for (int i = 0; i < inputCount; i++) {
        Console::WriteLine("IN{0:D} : {1:D}", i, inputData[i]);
    }

    // OUT0～31の出力をONします
    const int outputCount = 32;
    unsigned char outputData[outputCount];
    for (int i = 0; i < outputCount; i++) {
        outputData[i] = 1;
    }
    result = YduDioOutput(unitId, outputData, 0, outputCount);

    // ユニットをクローズします
    YduClose(0);

    return 0;
}
```

開発環境の設定

プロジェクトにドライバへの参照を追加します。

- Nugetからインストールする場合
  - Y2.UsbPc104.YduCs をインストールします。
- ソフトウェアパックから追加する場合
  - 以下のファイルをプロジェクトフォルダにコピーします
    - Ydu.cs
    - YduDio.cs
  - Ydu.cs, YduDio.csをプロジェクトに追加します

ソースファイルにusing ディレクティブを使ってYduCsを宣言します

```
using YduCs;
```

プログラム例

```
using YduCs;

static void Main()
{
    // IDが0に設定されているPMC-S4/00/00A-Uをオープンします
    const ushort UnitId = 0;
    var result = Ydu.Open(UnitId, "PMC-S4/00/00A-U");
    if (result != Ydu.YDU_RESULT_SUCCESS)
    {
        Console.WriteLine("オープンできません");
        return;
    }

    // デジタル入力（IN0～7）の状態を読み込みます
    const int InputCount = 8;
    var inputData = new byte[InputCount];
    result = YduDio.Input(UnitId, inputData, 0, InputCount);
    for (var i = 0; i < InputCount; i++)
    {
        Console.WriteLine($"IN{i} : {inputData[i]}");
    }

    // デジタル出力（OUT0～11）を全てONします
    const int OutputCount = 12;
    var outputData = Enumerable.Repeat(1, OutputCount).Select(x => (byte)x).ToArray();
    result = YduDio.Output(UnitId, outputData, 0, OutputCount);

    // ボードをクローズします
    Ydu.Close(UnitId);
}
```

VB（.NET2002以降）

開発環境の設定

1. 以下のファイルをプロジェクトフォルダにコピーします。

   YduApi.vb
   YduDioApi.vb
   YduResult.vb

2. YduApi.vb, YduDioApi.vb, YduResult.vbをプロジェクトに追加します。

プログラム例

```
Dim result As Integer
Dim inputData(15) As Byte
Dim outputData(31) As Byte
Dim msgText As String
Dim i As Integer

'IDが0に設定されているPMC-S4/16/32A-Uをオープンします
result = YduOpen(0, "PMC-S4/16/32A-U")
If result <> YDU_RESULT_SUCCESS Then
    MessageBox.Show("オープンできません", "", MessageBoxButtons.OK, MessageBoxIcon.Stop)
    Exit Sub
End If

msgText = ""
'IN0～15の入力をおこないます
result = YduDioInput(0, inputData, 0, 16)
For i = 0 To 15
    msgText = msgText & "IN" & i & " : " & inputData(i) & ControlChars.CrLf
Next
MessageBox.Show(msgText)

'OUT0～31の出力をONします
For i = 0 To 31
    outputData(i) = 1
Next
result = YduDioOutput(0, outputData, 0, 32)

'ユニットをクローズします
YduClose(0)
```

Visual Basic 6.0/VBA

開発環境の設定

VB6.0

1. 以下のファイルをプロジェクトフォルダにコピーします
   YduBaseApi.bas
   YduDioApi.bas
2. YduBaseApi.bas, YduDioApi.basをプロジェクトに追加します

VBA

1. 以下のファイルをインポートします
   YduBaseApi.bas
   YduDioApi.bas

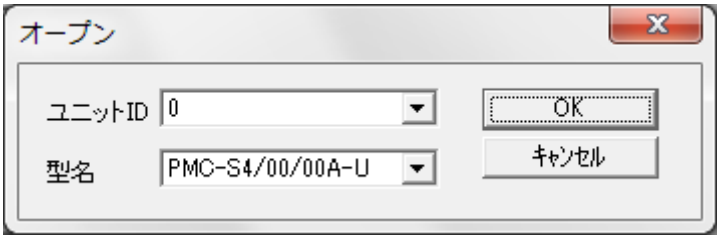
プログラム例

```
Dim result As Long
Dim modelName As String
Dim inputData(0 To 15) As Byte
Dim outputData(0 To 31) As Byte
Dim resultClose As Boolean
Dim strInData As String
Dim i As Integer

'IDが0に設定されているPMC-S4/16/32A-Uをオープンします
modelName = "PMC-S4/16/32A-U"
result = YduOpen(0, modelName)
If result <> YDU_RESULT_SUCCESS Then
    MsgBox "オープンできません", vbInformation
    Exit Sub
End If

'IN0～15の入力をいます
result = YduDioInput(0, inputData(0), 0, 16)
'入力データの表示
For i = 0 To 15
    strInData = strInData & "IN" & i & " : " & inputData(i) & vbCrLf
Next
MsgBox strInData, vbInformation

'OUT0～31の出力をONします
For i = 0 To 31
    outputData(i) = 1
Next
result = YduDioOutput(0, outputData(0), 0, 32)

'ユニットをクローズします
resultClose = YduClose(0)
```

基本動作

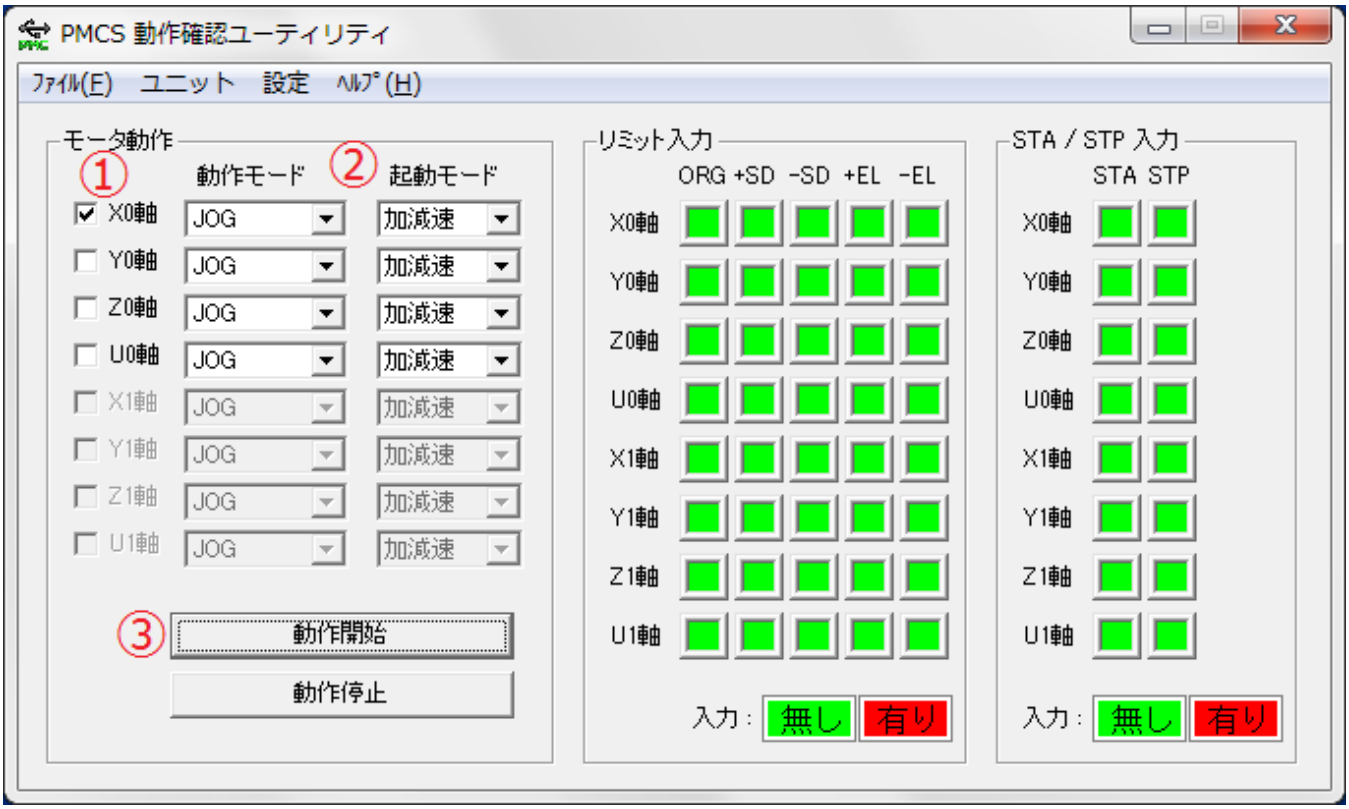
各軸のモーター動作の確認ができます。

ユニットのオープン



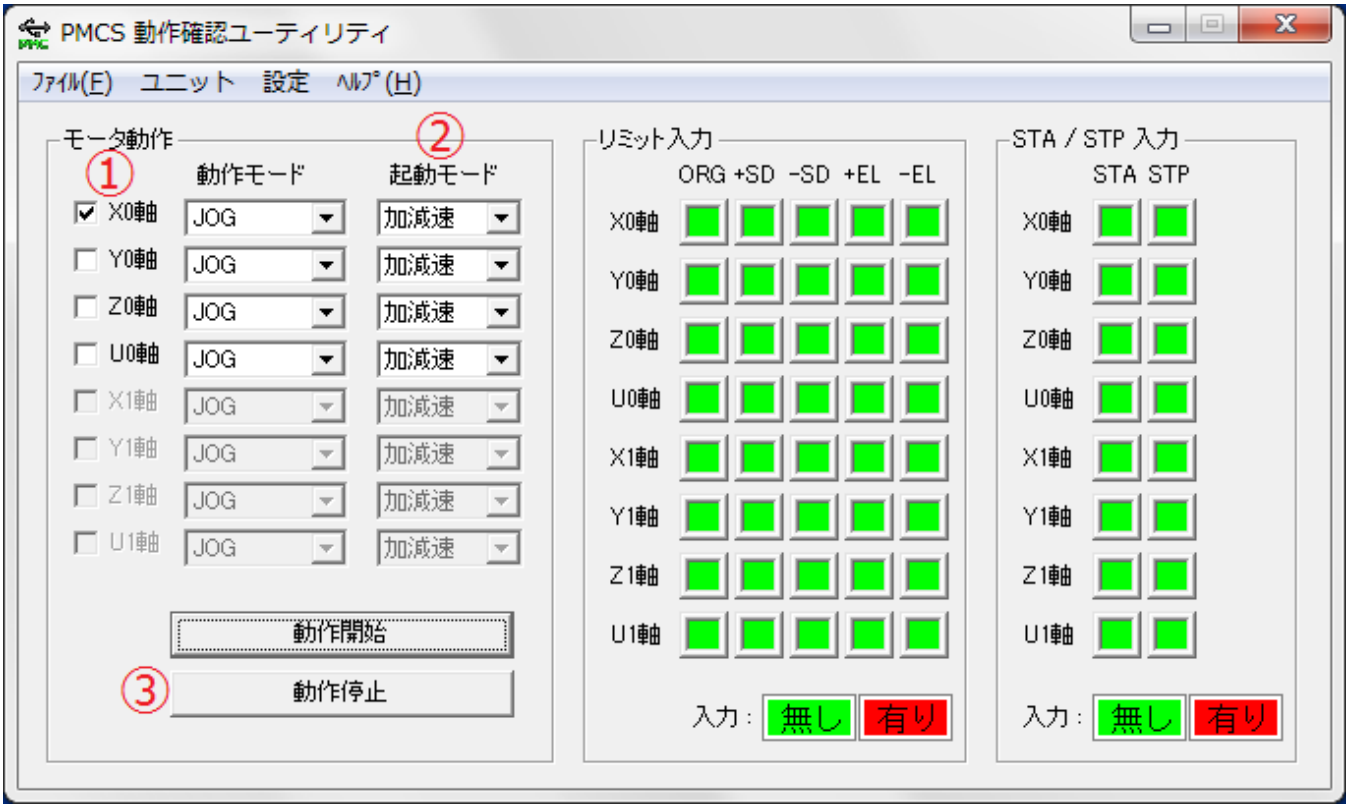
1. メニューの[ユニット]-[オープン]をクリックしてください
2. 表示された画面にてユニットID及び型名を選択して[OK]をクリックしてください

各軸の動作



1. 動作させる軸のチェックボックスをクリックし、チェックされた状態にしてください
2. 動作させる軸の動作モード及び起動モードを選択してください
3. 動作開始ボタンをクリックしてください

各軸の停止



1. 停止させる軸のチェックボックスをクリックし、チェックされた状態にしてください
2. 起動モードが[加減速]を選択している場合は減速停止、[一定速]を選択している場合は即停止となります
3. 動作停止ボタンをクリックしてください

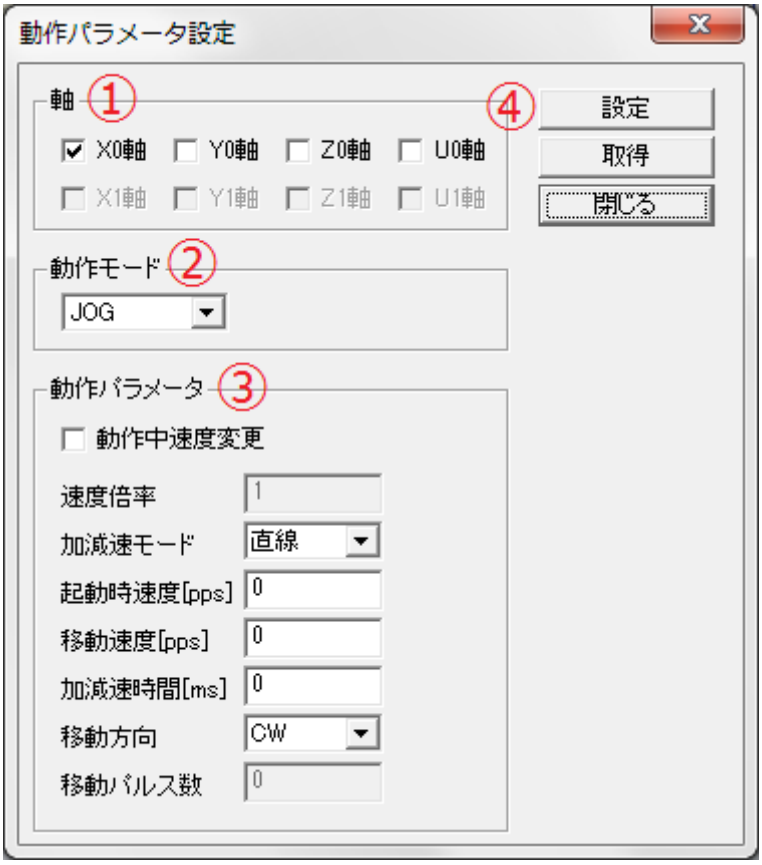
各センサ表示

100msecごとに各センサを監視しています。
入力が無い場合は緑、入力がある場合は赤で表示されます。

動作パラメータ設定

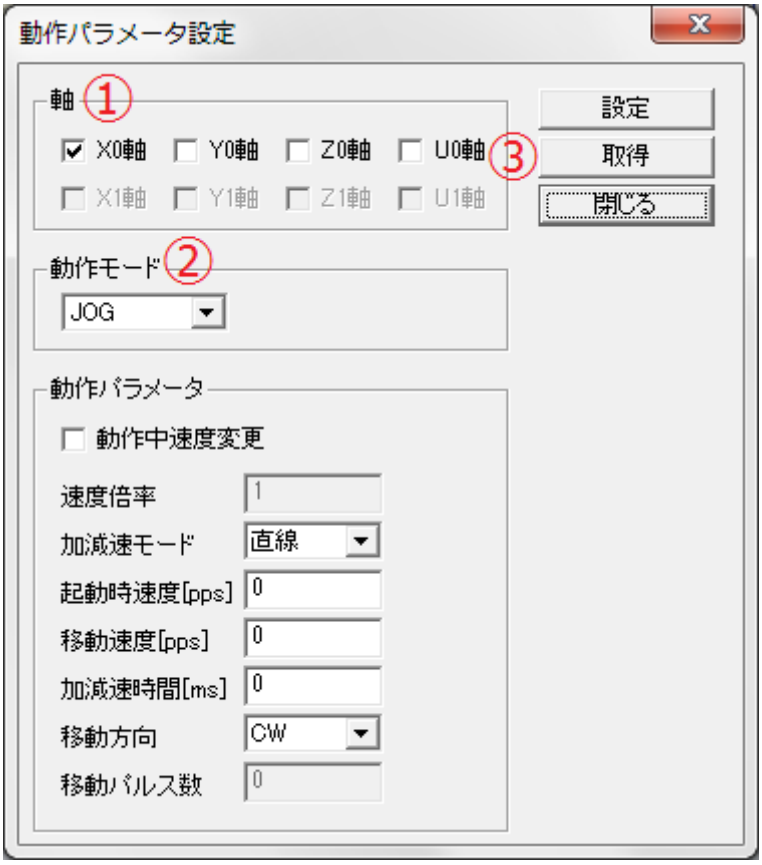
メニューの[設定]-[動作パラメータ]をクリックすると動作パラメータ設定の画面が表示されます。

設定



1. 動作パラメータの設定をおこなう軸のチェックボックスをクリックし、チェックされた状態にしてください（複数軸の指定が可能です）
2. 動作モードを選択してください
3. 各パラメータを設定してください
4. [設定]ボタンをクリックしてください

取得



1. 動作パラメータ設定の取得をおこなう軸のチェックボックスをクリックし、チェックされた状態にしてください（複数軸の指定はできません）
2. 動作モードを取得してください
3. [取得]ボタンをクリックしてください

動作中速度変更について

モーターを動作させたまま速度を変更したい場合は以下の方法でおこなってください。

1. 動作パラメータ設定画面にて[動作中速度変更]にチェックを入れ、適切な速度倍率及び各パラメータを設定し、[設定]ボタンをクリックしてください（速度倍率については動作パラメータ計算方法を参照してください）
2. 動作パラメータ設定画面を閉じ、モーターを動作させてください
3. モーターを動作させた状態で再度動作パラメータ設定画面を開いてください
4. 動作パラメータの取得をおこなってください
5. [動作中速度変更]にチェックを入れ、[移動速度]のみを変更し、[設定]ボタンをクリックしてください
6. 動作パラメータ設定画面を閉じ、メイン画面にて[動作開始]ボタンをクリックしてください

パルス出力設定

メニューの[設定]-[パルス出力]をクリックするとパルス出力設定の画面が表示されます。

設定

1. パルス出力の設定をおこなう軸のチェックボックスをクリックし、チェックされた状態にしてください（複数軸の指定が可能です）
2. 設定をおこなう項目（パルス出力論理、アイドリングパルス、パルス方式）の設定をおこなってください
3. 設定をおこなう項目の[設定]ボタンをクリックしてください

取得

1. パルス出力設定の取得をおこなう軸のチェックボックスをクリックし、チェックされた状態にしてください（複数軸の指定はできません）
2. 取得をおこなう項目の[取得]ボタンをクリックしてください

センサ設定

メニューの[設定]-[センサ]をクリックするとセンサ設定の画面が表示されます。

設定

1. センサの設定をおこなう軸のチェックボックスをクリックし、チェックされた状態にしてください（複数軸の指定が可能です）
2. 設定をおこなう項目（センサ極性、STA/STP有効・無効、SD有効・無効）の設定をおこなってください
3. 設定をおこなう項目の[設定]ボタンをクリックしてください

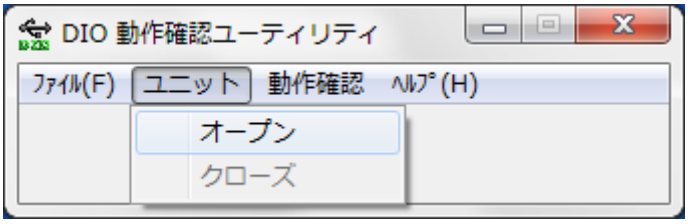
取得

1. センサ設定の取得をおこなう軸のチェックボックスをクリックし、チェックされた状態にしてください（複数軸の指定はできません）
2. 取得をおこなう項目（センサ極性、STA/STP有効・無効、SD有効・無効）の[取得]ボタンをクリックしてください

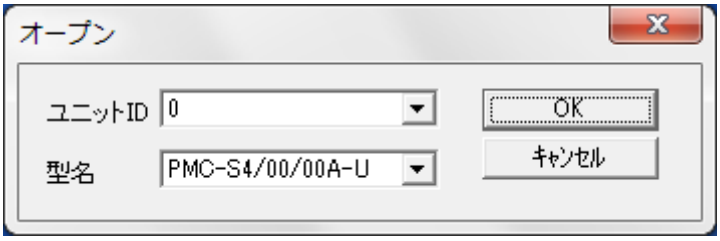
デジタル入出力確認ユーティリティ

全入出力の確認ができます。
ユーティリティを起動後、以下の手順で入出力確認ができます。

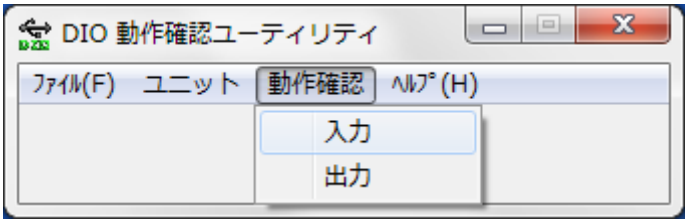
1. ユニットのオープン
   [ユニット]メニューの[オープン]をクリックしてください。



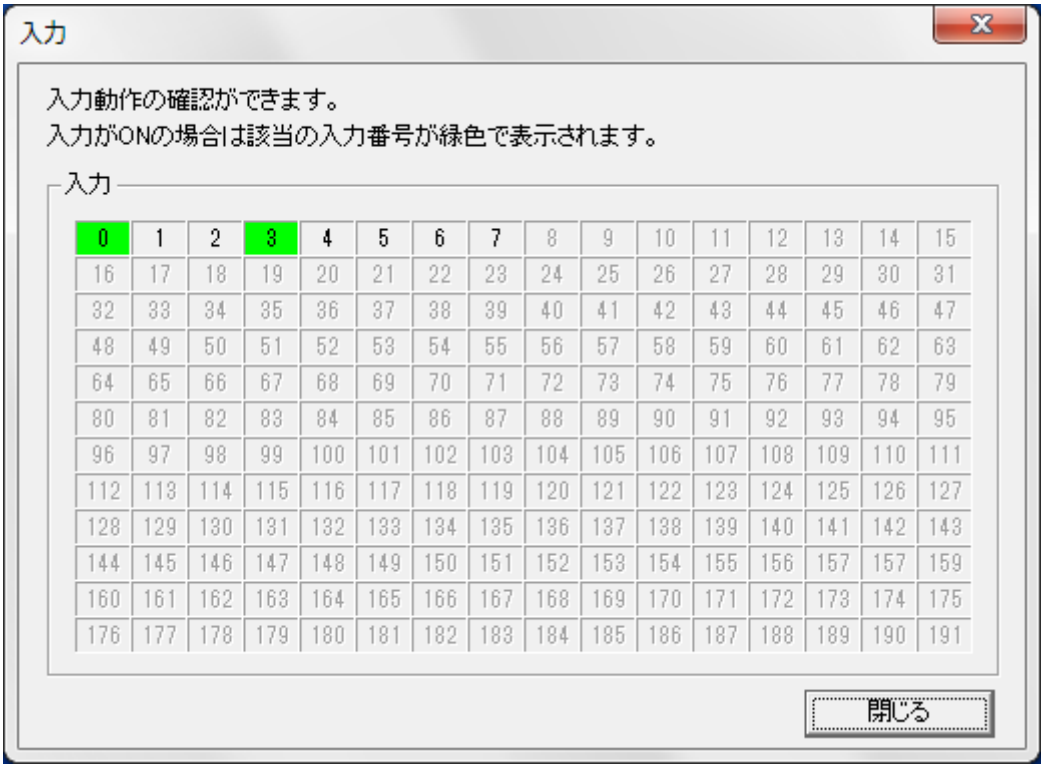
   表示された画面にてユニットID及び型名を選択してユニットのオープンをおこなってください。



2. 入力の確認
   [動作確認]メニューの[入力]をクリックしてください。

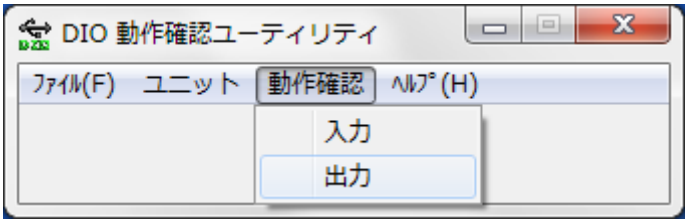


   入力番号の一覧が表示されており、該当の入力がONの場合は緑、OFFの場合は灰色で表示されます。
   （入力の状態を100msec毎に監視しています）



3. 出力の確認

[動作確認]メニューの[出力]をクリックしてください。

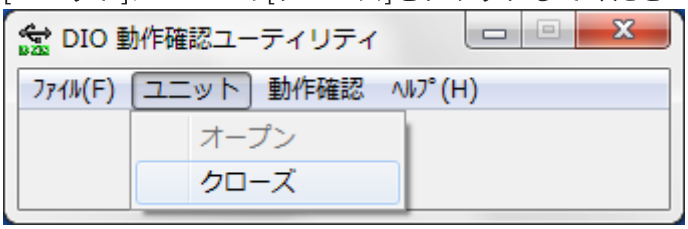


出力番号の一覧が表示されていますので、出力ON/OFFを切り替えたい出力番号をクリックしてください。ONの時には緑、OFFの時は灰色で表示されます。

（出力画面を閉じた際に全出力OFFになります）

4. ユニットのクローズ

[ユニット]メニューの[クローズ]をクリックしてください。



続けて他のユニットを確認したい場合は、1から実行してください。

（ユーティリティ終了時にクローズ処理をおこなうようにしていますので、クローズを実行せずにユーティリティを終了させても問題ありません）

注意事項

本製品及び本書の内容については、改良のために予告なく変更することがあります。

本製品を運用した結果の他への影響については、上記にかかわらず責任を負いかねますのでご了承ください。

本製品は人命にかかわる設備や機器、及び高度な信頼性を必要とする設備や機器としての使用またはこれらに組み込んでの使用は意図されておりません。
これら、設備や機器、制御システムなどに本製品を使用され、本製品の故障により人身事故、火災事故、損害などが生じても、弊社ではいかなる責任も負いかねます。
設備や機器、制御システムなどにおいて、安全設計に万全を期されるようご注意願います。

本製品は「外国為替及び外国貿易法」の規定により戦略物資等輸出規制製品に該当する場合があります。
国外に持ち出す際には、日本国政府の輸出許可申請などの手続きが必要になる場合があります。

本製品は日本国内仕様です。本製品を日本国外で使用された場合、弊社は一切の責任を負いかねます。
また、弊社は本製品に関し、日本国外への技術サポート等をおこなっておりませんので、予めご了承ください。