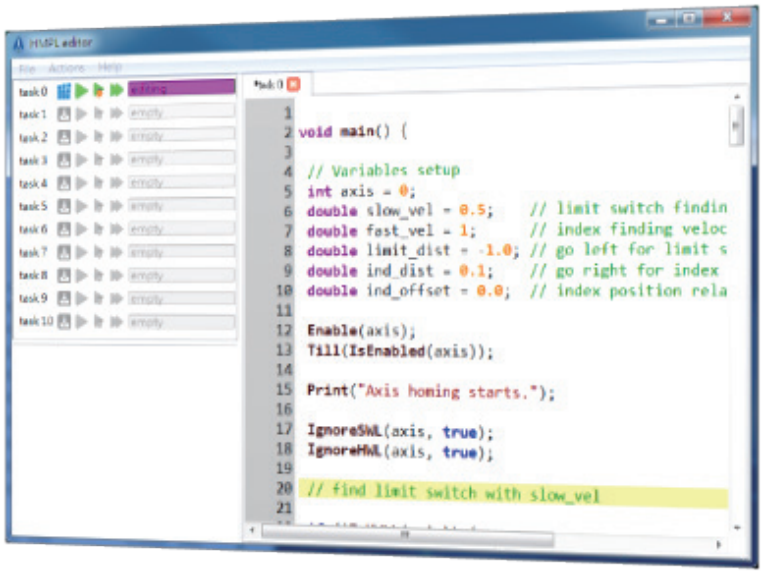
HIWIN® MIKROSYSTEM

Print(“Hello World”);



HIMC

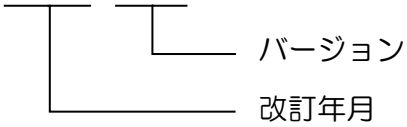
HMPL ユーザーガイド

www.hiwinmikro.tw
MH06UJ01-2503_V1.1

改訂履歴

ガイドのバージョンは、表紙の下部にも記載されています。

MH06UJ01-2503_V1.1



日付	バージョン	適用機種	改訂内容
2025 年 3 月	1.1	iA Studio 3.1.0	1. 機能を追加： ■ 第 2 章 IsSystemPreOp, GetFirmwareDate, GetFirmwareHash, GetModelNum ■ 第 5 章 GetPosFbComp, GetBufferMode, GetCmdNum ■ 第 8 章 SetGrpLookAheadPrm, SetGrpQueueSize ■ 第 12 章 SetPT_PosArray, SetPT_StateArray, SetPT_StartIndex, SetPT_EndIndex ■ 第 20 章 SetHomedStatus 2. 機能の変更： ■ 第 2 章 IsSystemOper ■ 第 5 章 GetCurrFb - the current feedback, SetOpMode ■ 第 10 章 SetSlvAORaw, GetSlvAORaw, GetSlvAIRaw ■ 第 17 章 ReadSDO, ReadSDOEx, WriteSDO, ReadPDO, ReadPDOEx, WritePDO, FroceWritePDO - parameter type of return and setting value 3. 機能説明の変更/追加： ■ 第 1 章 Add data type Data_t. ■ 第 5 章 Add CoE variables in Axis variables. ■ 第 8 章 Modify the description of buffer modes and transition modes. Modify examples. Add the look ahead function. ■ 第 12 章 Add the description, procedure, and examples of Random PT variables. ■ 第 19 章 Add the example of _AUTORUN_.

日付	バージョン	適用機種	改訂内容
			■ 第 20 章 Modify the description of homing methods. 4. エラーメッセージの追加： セクション 18.1.1, セクション 18.1.2, セクション 18.1.3
2023 年 12 月 5 日	1.0	iA Studio 3.0.0	1. HIMC が CoE 通信をサポート ： Read/Write SDO と PDO function を追加 Get/Set Slave Var および Run PDL の関連関数を削除 2. 機能の追加 ■ 第 2 章 IsSystemInit、GetECATSt、GetSlvECATSt、ScanNetwork 追加 ■ 第 5 章 SetOpMode、SetBufferMode 追加 ■ 第 10 章 SetSlvAOHex、GetSlvAOHex ■ 第 13 章 SetTouchProbeFunc 追加 ■ 第 18 章 GetDriveErr 追加 ■ 第 20 章 MoveHome、SetHomeSwitchVel、SetHomeZeroVel、SetHomeAcc、IsHomed, IsHoming 追加 3. 例を追加: セクション 8.1.6 4. 第 12 章、Random PT の関連機能を削除
2022 年 6 月 30 日	0.9	iA Studio 2.0	1. 章・セクションを追加します： ■ 10 章 – AIO 関数 ■ セクション 21.3 – Modbus 通信 2. 機能を追加： ■ 2 章 SendMsgEvent ■ 5 章 MoveTrq, MovePVT, IsAcc ■ 8 章 ArcAngle2D, SetGrpAngMotionProfile, GetGrpCoordTrans, SetGrpCoordTrans, GetGrpPoseCmd, GetGrpPoseFb,

日付	バージョン	適用機種	改訂内容
			CircRel ■ 9 章 SetGPIInvert, SetGPOInvert, BindEMO, GetAllGPIInvertSt, GetAllGPOInvertSt ■ 14 章 SetCompAlgType ■ 20 章 AxisHome, SetHomeType, SetHomeMethod, SetHomeProfile, SetHomeOffset, SetHomeTimeout, SetEndStopPosErr, SetEndStopDist 3. 例を追加：セクション 8.1.6 4. 変数を追加： セクション 5.1.1、セクション 8.1.1 5. エラーメッセージを追加： セクション 18.1.1、セクション 18.1.2、セクション 18.1.3
2021 年 12 月 24 日	0.8	iA Studio 1.4	1. 機能を追加： ■ 第 11 章 SetPT_PosArray, SetPT_StateArray, SetPT_StartIndex, SetPT_EndIndex 2. 例を追加：セクション 11.1.3 3. 削除例：セクション 8.1.6
2021 年 9 月 15 日	0.7	iA Studio 1.4	1. 章・セクションを追加： ■ セクション 9.1.1 – GPIO 変数 ■ 21 章 – 通信機能 2. 機能を追加： ■ 2 章 GetFirmwareVer ■ 5 章 GetVelFb, GetVelErr, GetCurrFb, SetVelScale, GetVelScale, SetRollover, GetVelScale, SetRollover, GetRolloverTurns, IsDriveErr, IsPosErr ■ 8 章 JogGroup, JogGroupAxis, SetGrpVelScale, GetGrpVelScale

日付	バージョン	適用機種	改訂内容
			■ 13 章 SetupComp3D 3. 例を追加： SetHMIScope, セクション 8.1.6、 セクション 9.1.2、セクション 13.1.1、 セクション 19.3.1 4. 変数を追加： セクション 5.1.1、セクション 8.1.1、 セクション 16.1.2 5. エラーメッセージを追加： セクション 17.1.1、セクション 17.1.2、セクション 17.1.3 6. 原点復帰方法を追加： セクション 19.1
2020 年 9 月 16 日	0.6	iA Studio 1.3	1. セクション 13.1.1： 図とコードを変更 2. セクション 15.1： 説明を追加 3. セクション 16.1.2： 表 16.1.2.3 の注記を修正
2020 年 6 月 30 日	0.5	iA Studio 1.3	1. 単位系の変更： meter-radian-second → mm-deg-ms 2. 章を追加： ■ 19 章 マルコの定義と機能 ■ 20 章 原点復帰機能 3. 5 章、7 章、8 章、9 章、10 章、11 章、 12 章、13 章、15 章、16 章、17 章の概 要を追加・修正 4. 機能を追加： ■ 5 章 Halt, Resume, SetAccTime, SetDecTime, IgnorePE ■ 6 章 IsInGear, IsGearMaster, IsGearSlave ■ 7 章 GetGantryPairID, IsGantryPair ■ 8 章 HaltGroup, ResumeGroup, ArcCW2D, ArcCCW2D, GetGrpKin, GetGrpMaxVel, SetGrpVel,

日付	バージョン	適用機種	改訂内容
			GetGrpMaxAcc, SetGrpAcc, SetGrpAccTime, GetGrpMaxDec, SetGrpDec, SetGrpDecTime, GetGrpSMTime, SetGrpSMTime, GetGrpCoordSys, SetGrpCoordSys, GetGrpBufferMode, SetGrpBufferMode, GetGrpTransMode, SetGrpTransMode, SetGrpTransPrm, GetGrpCmdNum ■ 9 章 SetAllGPO, SetSlvAllGPO, GetAllGPI, GetAllGPO, GetSlvAllGPI, GetSlvAllGPO ■ 13 章 GetCompPos ■ 16 章 RunSlvPdlFunc, StopSlvPdlFunc, IsSlvPdlFuncRunning ■ 19 章 AxisHome 5. 削除機能： ■ 8 章 Bezier ■ 12 章 SetPT_Polarity 6. 名前変更機能： ■ 8 章 GroupReset → ResetGroup
2020 年 4 月 28 日	0.4	iA Studio 1.2.4107.1	1. 18 章： Basic、Advanced、および E1 シリーズドライバーガントリーモードの原点復帰手順の例を変更
2019 年 11 月 29 日	0.3	iA Studio 1.2.4032.0	1. セクション 2.10： 関数 Till に注釈を追加 2. 11 章： PT 機能の使用フローを追加 3. 18 章： E1 シリーズドライバーガントリーモードの原点復帰手順例を追加

日付	バージョン	適用機種	改訂内容
2019 年 4 月 2 日	0.2	iA Studio 1.1.3772.0	1. 章の配置を変更 2. 「シンクロモーション」、「フィルター機能」、「エラー機能」の章を追加 3. 名前変更機能： ■ 2 章 IsOperMode → IsSystemOper IsPreOpMode → IsSystemPreOp ■ 9 章 SetGPO_OnOff → SetGPO SetGPO_Toggle → ToggleGPO 4. 機能を追加： ■ 2 章 IsSystemError, GetSlaveNum RescanMoE, SetHMIScope ■ セクション 5.3 GetSWRL, SetSWRL GetSWLL, SetSWLL ■ セクション 8.3 ResetGroup ■ 9 章 SetSlvGPO, ToggleSlvGPO IsSlvGPI_On, IsSlvGPO_On ■ 10 章 SetTableValue, GetTableValue ■ 16 章 GetSlvSt, SetSlvSt GetConfigVar, SetConfigVar
2018 年 4 月 17 日	0.1	iA Studio 1.0.2461.0	初版

目次

1. はじめに

1.1 HMPL の仕組み

HIWIN モーションプログラミング言語 (HMPL) は、ユーザーが自由に使える C のような構文を使用して独立したタスクを構築します。アプリケーションに基づいて、ユーザーは iA Studio の HMPL エディターでモーションコントロールロジックとプログラムを編集でき、プログラムはコンパイルされ、HMPL コンパイラーを介して HIMC にロードされます。HIMC のリアルタイムプロシージャは、通信サイクルごとに固定数の基本コマンドユニットを実行します。

注：
アイコン  は、iA Studio のメッセージウィンドウや、ASCII TCP 通信（第 21 章を参照）を介してインストールされたターミナルアプリケーションで機能が使用できることを示します。

1.2 バージョンの説明

HIMC コントローラーが iA Studio 3.0 (付属) 以降のソフトウェアバージョンで適用されている場合、CoE 通信機能を備えた HIMC コントローラー (製品モデル: MC-XX-XX-01XX) をサポートしますが、HIMC コントローラー (製品モデル: MC-XX-XX-00-XX）MoE 通信機能搭載についてはサポートされません。MoE 通信機能付き HIMC コントローラーを使用する場合、ソフトウェアバージョン は、iA Studio 2.X (付属) 以下を適用する必要があります。

iA Studio 1.3 (同梱) 以降で採用されているモーション変数の単位:
直線運動(mm)、回転運動(deg)、時間(ms)
iA Studio 1.2 (同梱) 以下で採用されているモーション変数の単位:
直線運動(m)、回転運動(rad)、時間(s)

1.3 免責事項

ユーザーは、このガイドで提供されているサンプル コードを特定の用途に採用または変更できます。ただし、さまざまなアプリケーションシナリオでは、正確性、有効性、安全性を保証することはできません。 ユーザーは、ソフトウェア実装の安全性と有効性について全責任を負う必要があります。

1.4 データタイプ

HMPL では、データ型を使用して変数を宣言したり、関数の戻り値を取得したりします。変数の型によって、ストレージ内の領域サイズの占有とその有効な値が決まります。

表 1.3.1

タイプ	説明	サイズ (Byte)	有効な値
char int8_t	8 ビットの符号付き整数	1	-128 ~ 127
unsigned char uint8_t	8 ビット符号なし整数	1	0 ~ 255
short int16_t	16 ビット符号付き整数	2	-32768 ~ 32767
unsigned short uint16_t	16 ビット符号なし整数	2	0 ~ 65535
int int32_t	32 ビット符号付き整数	4	-2147483648 ~ 2147483647
unsigned int uint32_t	32 ビット符号なし整数	4	0 ~ 4294967295
long long int64_t	64 ビット符号付き整数	8	-9223372036854775808 ~ 9223372036854775807
unsigned long long uint64_t	64 ビット符号なし整数	8	0 ~ 18446744073709551615
float	32 ビット浮動小数点型 (小数点以下 6 桁の精度)	4	1.17549e-38 ~ 3.40282e+38
double	64 ビット浮動小数点型 (小数点以下 15 桁の精度)	8	2.225074e-308 ~ 1.797693e+308
int* char* double* …	特定の型の変数の格納場所のアドレスを含むポインター型。	8	N/A
void	戻り値の型が void の関数は値を返しません。	N/A	N/A
void*	任意の型の変数の格納場所のアドレスを含むジェネリックポインター型。	8	N/A
Timer	関数 TON および TOF のタイマーオブジェクトを宣言する型。	8	N/A

1.5 スコープ規則

定義された変数または関数のスコープは、HMPL タスクにおけるその存在領域を示します。領域を超えて、変数と関数にアクセスすることはできません。

表 1.4.1

タイプ	スコープ	宣言の配置	説明
global function	global scope	in **task 0**	どこでも使用できます
task function	task scope	not in **task 0**	そのタスクでのみ使用できます
global variable	global scope	out of all functions but in **task 0**	どこでも使用できます
task variable	task scope	out of all functions and **task 0**	そのタスクでのみ使用できます
local variable	block scope	in a block	そのブロックでのみ使用できます
	function scope	in a function	その機能でのみ使用できます

注：グローバル変数とタスク変数は、コンパイル時にのみ初期化されます。

例 1.

```
//   in task 0
//   Declare a global variable
int global_var = 100;

//   Declare a global function
void GlobalFunction1() {
      Print("%d", global_var);
}
```

例 2.

```
//   in task 1
//   Declare a task variable
int task_var = 0;

//   Declare a task function
void TaskFunction1() {
      //   Declare a local variable
      int local_var = task_var;
      for (int i = 0; i < local_var; ++i) {   //   block start
      global_var += i;   //   i is a local variable with block scope
  }   //   block end
  global_var += local_var;
}
void main() {
      task_var = 10;
      TaskFunction1();
      GlobalFunction1();   //   the output is 155
}
```

(このページはブランクになっています)

2. HIMC システム機能

2.1 IsSystemInit

目的
HIMC システムが「初期化」状態にあるかどうかをクエリします。

構文

```
int IsSystemInit();
```

パラメーター
N/A

戻りの値
HIMC システムが「SystemInit」状態にある場合、int 値 TRUE (1) を返します。 それ以外の場合は、FALSE (0) を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

2.2 IsSystemOper

目的

HIMC システムが「運用」状態にあるかどうかを問い合わせます。その場合、モーション関連の機能を使用できます。

構文

```
int IsSystemOper();
```

パラメーター

N/A

戻りの値

HIMC システムが「SystemOper」状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

2.3 IsSystemError

目的

HIMC システムが「エラー」状態にあるかどうかを問い合わせます。そうである場合、HIMC とスレーブ間の通信でエラーが発生します。

構文

```
int IsSystemError();
```

パラメーター

N/A

戻りの値

HIMC システムが「SystemError」状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

2.4 GetECATSt

目的

コントローラーの EtherCAT ステートマシンを取得します。

構文

```
int GetECATSt();
```

パラメーター

N/A

戻りの値

コントローラーの EtherCAT ステートマシン: 1：Init、2：Pre-Operation、4：Safe-Operation、8：Operation。

条件

サポートされる最小バージョン	iA Studio 3.0

2.5 GetSlvECATSt

目的

スレーブの EtherCAT ステートマシンを取得します。

構文

```
int GetSlvECATSt(
    int slv_id
);
```

パラメーター

N/A

戻りの値

スレーブの EtherCAT ステートマシン: 1：Init、2：Pre-Operation、4：Safe-Operation、8：Operation。

条件

サポートされる最小バージョン	iA Studio 3.0

2.6 DisableAll

目的
すべての軸とすべての軸グループを無効にします。

構文

```
int DisableAll();
```

パラメーター
N/A

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考
軸と軸グループのモーションキューがクリアされます。

条件

サポートされる最小バージョン	iA Studio 0.23

2.7 StopAll

目的

kill 減速で全軸、全軸グループを停止させ、「enable」状態にする

構文

```
int StopAll();
```

パラメーター

N/A

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

軸と軸グループのモーションキューがクリアされます。

条件

サポートされる最小バージョン	iA Studio 0.23

2.8 EStop

目的

コントローラー内のすべての実行プログラム(すべての HMPL タスクを含む)を停止し、すべての軸とすべての軸グループを無効にします。

構文

```
void EStop();
```

パラメーター

N/A

戻りの値

N/A

条件

サポートされる最小バージョン	iA Studio 0.23

2.9 GetSlaveNum

目的

コントローラーに接続されているスレーブの数を取得します。

構文

```
int GetSlaveNum();
```

パラメーター

N/A

戻りの値

コントローラーに接続されているスレーブの数

条件

サポートされる最小バージョン	iA Studio 1.1

2.10 GetFirmwareVer

目的

コントローラーのファームウェアバージョンを取得します。

構文

```
int GetFirmwareVer(
        char *ver_buf
);
```

パラメーター

ver_buf [out]　　返されたファームウェアバージョンの文字列を受け取るバッファーへのポインター

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
void main() {
        char ver_buf[15] = {0};
        GetFirmwareVer(ver_buf);
        Print("%s", ver_buf);
}
```

条件

サポートされる最小バージョン	iA Studio 1.4

2.11 ScanNetwork

目的

コントローラーとスレーブ間の接続を再スキャンします。

構文

```
int ScanNetwork();
```

パラメーター

N/A

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

2.12 SetHMIScope

目的

スコープの実行を開始または停止します。

構文

```
int SetHMIScope(
        int start
);
```

パラメーター

start [in]　　データの記録を開始するには、「1」に設定します。
　　データの記録を停止するには、「0」に設定します。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
void main() {
        //  Enable axis
        int axis_id = 0;
        Enable(axis_id);            Till(IsEnabled(axis_id));
        //  Start executing the scope
        SetHMIScope(1);
        //  P2P motion
        MoveAbs(axis_id, 100); Till(IsInPos(axis_id));
        MoveAbs(axis_id, 0);    Till(IsInPos(axis_id));
        //  Stop executing the scope
        SetHMIScope(0);
}
```

条件

サポートされる最小バージョン	iA Studio 1.1

2.13 Sleep

目的

特定の期間、HMPL タスクの実行を停止します。

構文

```
void Sleep(
    int ms
);
```

パラメーター

ms [in] ミリ秒単位の時間

戻りの値

N/A

条件

サポートされる最小バージョン	iA Studio 0.23

2.14 SendEvent

目的

ホスト PC に ID でイベントを送信します。

構文

```
int SendEvent(
        unsigned short evt_id
);
```

パラメーター

evt_id [in]　　ユーザー定義のイベント ID

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

(1) ホスト PC は、HIMC API リファレンスガイドの関数「HIMC_SetHmplEvtCallback」で、イベント ID をキャプチャするコールバック関数を設定できます。

(2) この関数はあまり頻繁に呼び出すことはできません (通常は 1KHz)。頻繁に呼び出されると、平均呼び出し周波数が 1KHz を下回るまでブロックされます。

条件

サポートされる最小バージョン	iA Studio 0.11

2.15 SendMsgEvent

目的

ホスト PC に文字列メッセージを送信します。

構文

```
int SendMsgEvent(
        char *format,
        …
);
```

パラメーター

format [in] メッセージウィンドウに書き込まれるテキストを含むバッファーへのポインター。オプションで、プロトタイプ「% specifier」に続く埋め込みフォーマット指定子を含めることができます。指定子は、型と対応する引数を定義します。

指定子	出力	例
d or i	符号付き 10 進整数	589
u	符号なし 10 進整数	589
x	符号なし 16 進整数	24d
c	文字	M
s	文字列	Hello world
f	精度が 6 桁の 10 進浮動小数点	589.000000
e	6 桁の精度の科学表記法	5.890000e+02
g	%e または %f の最短表現	589
%	% の後に別の % が続く場合は、1 つの % を表します。	%

… [in] 追加のパラメーター。
各パラメーターには、書式文字列の書式指定子を置き換える値が含まれています。これらのパラメーターは、少なくともフォーマット指定子で指定された値の数と同じ数でなければなりません。余分なパラメーターは関数によって無視されます。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

(1) ホスト PC は、HIMC API リファレンスガイドの関数「HIMC_SetHmplMsgEvtCallback」を使用して、文字列メッセージをキャプチャするコールバック関数を設定できます。
(2) この関数はあまり頻繁に呼び出すことはできません(通常は 1KHz)。頻繁に呼び出されると、平均呼び出し周波数が 1KHz を下回るまでブロックされます。
(3) 文字列メッセージの最大長は 128 文字です。

例

```
void main()
{
    SendMsgEvent("variable: %d", 88);
}
```

条件

サポートされる最小バージョン	iA Studio 2.0

2.16 RunScheduler

目的

実行の準備ができている他のタスクのために、呼び出し元のタスクが CPU リソースを解放するようにします。

構文

```
void RunScheduler();
```

パラメーター

N/A

戻りの値

N/A

条件

サポートされる最小バージョン	iA Studio 0.23

2.17 MutexLock

目的

mutex オブジェクトを ID でロックします。

構文

```
void MutexLock(
    int mutex_id
);
```

パラメーター

mutex_id [in] Mutex オブジェクト ID
使用可能な mutex オブジェクトは 8 つあるため、ID は 0～7 です。

戻りの値

N/A

備考

(1) mutex オブジェクトが現在どのタスクによってもロックされていない場合、この関数は mutex オブジェクトをロックし、すぐに戻ります。mutex オブジェクトは、それをロックするタスクによって所有され、所有者タスクが MutexUnlock 関数を呼び出すか、所有者タスクが停止するまでロックされたままになります。
(2) mutex オブジェクトが現在別のタスクによってロックされている場合、この関数は mutex オブジェクトがロック解除されるまでブロックされます。
(3) この関数を呼び出した同じタスクによって mutex オブジェクトが既にロックされている場合、そのタスクは停止され、実行時エラーメッセージが送信されます。

条件

サポートされる最小バージョン	iA Studio 0.23

2.18 MutexUnlock

目的

ID で metex オブジェクトのロックを解除します。

構文

```
void MutexUnLock(
    int mutex_id
);
```

パラメーター

mutex_id [in] Mutex オブジェクト ID
使用可能な mutex オブジェクトは 8 つあるため、ID は 0～7 です。

戻りの値

N/A

備考

mutex オブジェクトが現在呼び出し元のタスクによってロックされていない場合、何も起こりません。

条件

サポートされる最小バージョン	iA Studio 0.23

2.19 TON

目的

立ち上がりタイマー機能。IN 入力条件が確立されると、この関数の戻り値は、PT ミリ秒後に 0 からゼロ以外の値に変わります。

構文

```
int TON(
        Timer *timer_p,
        int IN,
        int PT
);
```

パラメーター

timer_p [in]	タイマーオブジェクトを格納するバッファーへのポインター。
IN [in]	タイマーコマンド
PT [in]	プログラムされた時間 パラメーター単位：ms

戻りの値

出力信号が低い場合は int 値 0 を返し、出力信号が高い場合はゼロ以外の値を返します。

備考

(1) タイマーは IN 入力の立ち上がりパルスで開始し、経過時間がプログラムされた時間と等しくなるとすぐに停止します。IN 入力の立ち下がりパルスでタイマーを 0 にリセットします。設定した時間が経過すると、出力信号が 1 にセットされます。入力コマンドが立ち下がると、0 にリセットされます。

(2) タイマーを再起動するには、TimerInit (タイマー初期化子)でタイマーオブジェクトを初期化します。

例

```
void main()
{
    //  Initialize counter
    Timer timer1 = TimerInit;
    Timer timer2 = TimerInit;

    int counter = 0;
    int target_cnt = 1000;
    int cnt_reach_delay_1s = 0;
    int cnt_reach_delay_3s = 0;

    for (;;) {
        Sleep(1);
        counter = counter + 1;
        //  After TON condition is satisfied, cnt_reach_delay_1s will change from
            0 to 1 in 1 second.
        cnt_reach_delay_1s = TON(&timer1, counter >= target_cnt, 1000);
        //  After TON condition is satisfied, cnt_reach_delay_3s will change from
            0 to 1 in 2 seconds.
        cnt_reach_delay_3s = TON(&timer2, cnt_reach_delay_1s, 2000);

        //  Save as system parameters to observe changes in value via Scope Manager.
        system_dtest0 = counter;
        system_dtest1 = target_cnt;
        system_dtest2 = cnt_reach_delay_1s;
        system_dtest3 = cnt_reach_delay_3s;

        //  After the condition is satisfied, counter will be reset in 0.5 second.
        if(cnt_reach_delay_3s){
            Sleep(500);
            counter = 0;
        }
    }
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

2.20 TOF

目的

立ち下がりタイマー機能。IN 入力条件が確立されると、この関数の戻り値は PT ミリ秒後にゼロ以外の値から 0 に変わります。

構文

```
int TOF(
    Timer *timer_p,
    int IN,
    int PT
);
```

パラメーター

timer_p [in]	タイマーオブジェクトを格納するバッファーへのポインター。
IN [in]	タイマーコマンド
PT [in]	プログラムされた時間 パラメーター単位：ms

戻りの値

出力信号が低い場合は int 値 0 を返し、出力信号が高い場合はゼロ以外の値を返します。

備考

(1) タイマーは、IN 入力の立ち下がりパルスで開始し、経過時間がプログラムされた時間と等しくなるとすぐに停止します。IN 入力の立ち上がりパルスはタイマーを 0 にリセットします。IN 入力が TRUE に立ち上がると、出力信号は 1 に設定されます。プログラムされた時間が経過すると、0 にリセットされます。

(2) タイマーを再起動するには、TimerInit (タイマー初期化子)でタイマーオブジェクトを初期化します。

例

```
void main()
{
    //  Initialize counter
    Timer timer1 = TimerInit;
    Timer timer2 = TimerInit;

    int counter = 10000;
    int target_cnt = 8000;
    int cnt_reach_delay_1s = 1;
    int cnt_reach_delay_3s = 1;

    for (;;) {
        Sleep(1);
        counter = counter - 1;
        //  After TOF condition is satisfied, cnt_reach_delay_1s will change from
            1 to 0 in 1 second.
        cnt_reach_delay_1s = TOF(&timer1, counter >= target_cnt, 1000);
        //  After TOF condition is satisfied, cnt_reach_delay_3s will change from
            1 to 0 in 2 seconds.
        cnt_reach_delay_3s = TOF(&timer2, cnt_reach_delay_1s, 2000);

        //  Save as system parameters to observe changes in value via Scope Manager.
        system_dtest0 = counter;
        system_dtest1 = target_cnt;
        system_dtest2 = cnt_reach_delay_1s;
        system_dtest3 = cnt_reach_delay_3s;

        //  After the condition is satisfied, counter will be reset in 0.5 second.
        if(!cnt_reach_delay_3s){
        Sleep(500);
            counter = 10000;
        }
    }
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

3. 文字列機能

3.1 概要

HMPL では、文字列は文字の 1 次元配列であり、ヌル文字 \0 で終了します。たとえば、次の宣言と初期化により、文字列「HIMC」が作成されます。

```
char str[5] = {‘H’, ‘I’, ‘M’, ‘C’, ‘\0’};
```

配列の末尾にヌル文字を保持するには、文字列を含む文字配列のサイズは、テキスト内の文字数に 1 を加えたものにする必要があります。したがって、例のサイズは 5 です。

上記のステートメントは、次のように書くこともできます。文字列の末尾にヌル文字を配置する必要はありません。HMPL コンパイラーは、配列を初期化するときに、文字列のサイズを自動的に 5 に設定し、文字列の末尾に「 \0 」を配置します。

```
char str[] = “HIMC”;
```

表 3.1.1 に示すように、上記の 2 つの文字列のメモリレイアウトは同じです。

表 3.1.1

str[0]	str[1]	str[2]	str[3]	str[4]
H	I	M	C	\0

注：HMPL では、文字列の最大長は 512 です。ユーザーは、「HMPL_STR_MAX_LEN」を介してこの値を取得できます。

3.2 Print

目的

フォーマットされた文字列をメッセージウィンドウに書き込みます。

構文

```
int Print(
        char *format,
        …
);
```

パラメーター

format [in] メッセージウィンドウに書き込まれるテキストを含むバッファーへのポインター。オプションで、プロトタイプ「% 指定子」に続く埋め込みフォーマット指定子を含めることができます。指定子は、型と対応する引数を定義します。

指定子	出力	例
d or i	符号付き 10 進整数	589
u	符号なし 10 進整数	589
x	符号なし 16 進整数	24d
c	文字	M
s	文字列	Hello world
f	精度が 6 桁の 10 進浮動小数点	589.000000
e	6 桁の精度の科学表記法	5.890000e+02
g	%e または %f の最短表現	589
%	% の後に別の % が続く場合は、1 つの % を表します	%

… [in] 追加のパラメーター。
各パラメーターには、書式文字列の書式指定子を置き換える値が含まれています。これらのパラメーターは、少なくともフォーマット指定子で指定された値の数と同じ数でなければなりません。余分なパラメーターは関数によって無視されます。

戻りの値

文字の総数。エラーが発生した場合は、-1 を返します。

例

```
void main() {

    char str[] = "hello world";
    int var1 = 321;
    double var2 = 1428.57;

    Print("var1: %d, var2: %f, str: %s", var1, var2, str);
    //   var1: 321, var2: 1428.570000, str: hello world

    Print("var2: %e", var2);
    //   var2: 1.428570e+03

    Print("var2: %g", var2);
    //   var2: 1428.57
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

3.3 StringPrint

目的

書式設定された文字列をバッファーに書き込みます。

構文

```
int StringPrint(
        char *destination,
        char *format,
        …
);
```

パラメーター

destination [out] 文字列の結果を受け取るバッファーへのポインター。

format [in] メッセージウィンドウに書き込まれるテキストを含むバッファーへのポインター
詳細については、セクション 3.2 Print を参照してください。

… [in] 追加のパラメーター
各パラメーターには、書式文字列の書式指定子を置き換える値が含まれています。
これらのパラメーターは、少なくともフォーマット指定子で指定された値の数と同じ数でなければなりません。余分なパラメーターは関数によって無視されます。

戻りの値

文字の総数。エラーが発生した場合は、-1 を返します。

備考

(1) この機能は印刷に似ています。違いは、出力文字列がメッセージウィンドウではなくバッファーに書き込まれることです。
(2) ソース文字列の終端のヌル文字もコピーされます。オーバーフローを避けるために、宛先文字列が指す配列のサイズは、ソース文字列を含むのに十分な長さでなければなりません (終端のヌル文字が含まれます)。

例

```
void main() {

    char dest[80];
    char str[] = "hello world";
    int var1 = 321;
    double var2 = 1428.57;

    StringPrint(dest, "var1: %d, var2: %f, str: %s", var1, var2, str);
    Print("%s", dest);    //    var1: 321, var2: 1428.570000, str: hello world

    StringPrint(dest, "var2: %e", var2);
    Print("%s", dest);    //    var2: 1.428570e+03

    StringPrint(dest, "var2: %g", var2);
    Print("%s", dest);    //    var2: 1428.57
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

3.4 StringLen

目的

文字列の長さを取得します。

構文

```
int StringLen(
        char *str
);
```

パラメーター

str [in] 文字列

戻りの値

文字列の長さ (終端のヌル文字は含まれません)。

例

```
void main() {

        char str[] = "hello world";
        int len = StringLen(str);    //    len = 11
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

3.5 IsStringEqual

目的

2 つの文字列が同じかどうかを確認します。

構文

```
int IsStringEqual(
        char *str1,
        char *str2
);
```

パラメーター

str1 [in]	文字列 1
str2 [in]	文字列 2

戻りの値

2 つの文字列が同じ場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

例

```
void main() {

        char str1[] = "hello world";
        char str2[] = "hello world";
        char str3[] = "hello worldd";

        int is_equal = IsStringEqual(str1, str2);    //    is_equal = 1
        is_equal = IsStringEqual(str1, str3);    //    is_equal = 0
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

3.6 StrFindChar

目的

文字列内で文字が最初に出現する位置を特定します。

構文

```
int StrFindChar(
        char *str,
        char character
);
```

パラメーター

str [in]	文字列
character [in]	文字

戻りの値

文字列内で最初に出現する文字までのオフセット値。
文字が見つからない場合は、-1 が返されます。

例

```
void main() {

        char str[] = "hello world";

        int offset = StrFindChar(str, 'h');    //    offset = 0
        offset = StrFindChar(str, 'l');    //    offset = 2
        offset = StrFindChar(str, 'z');    //    offset = -1
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

3.7 StrFindCharEx

目的

文字セットまたはその補数が文字列内で最初に出現する位置を特定します。

構文

```
int StrFindCharEx(
        char *str,
        char *char_set,
        int     complement_set
);
```

パラメーター

str [in]	文字列
char_set [in]	文字セット
complement_set [in]	オプションを見つけます False (0)：文字セットを見つける True (ゼロ以外)：文字セットの補数を見つけます

戻りの値

文字セット内の文字セットまたはその補数が最初に出現する位置までのオフセット値。
文字が見つからない場合は、-1 を返します。

例

```
void main() {

    char str[] = "hello world";

    int offset = StrFindCharEx(str, "lo ", false);    //    offset = 2
    offset = StrFindCharEx(str, "lo ", true);    //    offset = 0
    offset = StrFindCharEx(str, "zx!c", false);    //    offset = -1
    offset = StrFindCharEx(str, "leh", true);    //    offset = 4
}
```

条件

サポートされる最小バージョン	iA Studio 0.25

3.8 StrFindStr

目的

文字列内で部分文字列が最初に出現する位置を特定します。

構文

```
int StrFindStr(
        char *str1,
        char *str2
);
```

パラメーター

str1 [in]	文字列
str2 [in]	部分文字列

戻りの値

文字列内で部分文字列が最初に出現する位置までのオフセット値。
部分文字列が見つからない場合は、-1 が返されます。

例

```
void main() {

        char str[] = "hello world";

        int offset = StrFindStr(str, "hel");    //    offset = 0
        offset = StrFindStr(str, "wor");    //    offset = 6
        offset = StrFindStr(str, "wol");    //    offset = -1
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

3.9 StringCopy

目的

文字列をコピーします

構文

```
int StringCopy(
        char *destination,
        char *source
);
```

パラメーター

destination [out]　　文字列の結果を受け取るバッファーへのポインター。

source [in]　　コピーする文字列

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

ソース文字列の終端のヌル文字もコピーされます。オーバーフローを避けるために、宛先文字列が指す配列のサイズは、ソース文字列を含むのに十分な長さでなければなりません (終端のヌル文字が含まれます)。

例

```
void main() {

        char source[] = "hello world";
        char destination[80];

        StringCopy(destination, source);
        Print("%s", destination);    //    the output is hello world
}
```

条件

サポートされる最小バージョン	iA Studio 1.3

3.10 StringCopyEx

目的

部分文字列をコピーします。

構文

```
int StringCopyEx(
        char *destination,
        char *source,
        int start_pos,
        int copy_len
);
```

パラメーター

destination [out]	文字列の結果を受け取るバッファーへのポインター。
source [in]	コピーする文字列
start_pos [in]	コピーする部分文字列のオフセット
copy_len [in]	コピーする部分文字列の長さ -1 の場合、文字列の末尾までのすべての文字がコピーされます。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

ソース文字列の終端のヌル文字もコピーされます。オーバーフローを避けるために、宛先文字列が指す配列のサイズは、ソース文字列を含むのに十分な長さでなければなりません (終端のヌル文字が含まれます)。

例

```
void main() {

    char source[] = "hello world";
    char destination[80];

    StringCopyEx(destination, source, 6, 3);
    Print("%s", destination);    //    the output is wor

    StringCopyEx(destination, source, 6, -1);
    Print("%s", destination);    //    the output is world

    StringCopyEx(destination, source, 0, -1);
    Print("%s", destination);    //    the output is hello world
}
```

条件

サポートされる最小バージョン	iA Studio 1.3

3.11 StringCat

目的

2 つの文字列を連結します。

構文

```
int StringCat(
        char *destination,
        char *source
);
```

パラメーター

destination [out]　　文字列の結果を受け取るバッファーへのポインター。

source [in]　　追加する文字列

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

ソース文字列のコピーを宛先文字列に追加します。宛先文字列の終端のヌル文字は、ソース文字列の最初の文字で上書きされます。オーバーフローを避けるために、宛先文字列が指す配列のサイズは、ソース文字列を含むのに十分な長さでなければなりません (終端のヌル文字が含まれます)。

例

```
void main() {

        char str[80] = "hello";
        StringCat(str, " world");
        Print("%s", str);    //    the output is hello world
}
```

条件

サポートされる最小バージョン	iA Studio 1.3

3.12 StringCatEx

目的

部分文字列を連結します

構文

```
int StringCatEx(
        char *destination,
        char *source,
        int start_pos,
        int copy_len
);
```

パラメーター

destination [out]	文字列の結果を受け取るバッファーへのポインター。
source [in]	追加する文字列
start_pos [in]	コピーする部分文字列のオフセット
copy_len [in]	コピーする部分文字列の長さ -1 の場合、文字列の末尾までのすべての文字がコピーされます。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

ソース文字列のコピーを宛先文字列に追加します。宛先文字列の終端のヌル文字は、ソース文字列の最初の文字で上書きされます。オーバーフローを避けるために、宛先文字列が指す配列のサイズは、ソース文字列を含むのに十分な長さでなければなりません (終端のヌル文字が含まれます)。

例

```
void main() {

    char source[] = "friendsmy ";
    char destination[80] = "hello ";

    StringCatEx(destination, source, 7, -1);
    Print("%s", destination);    //    the output is hello my

    //    now the destination is hello my
    StringCatEx(destination, source, 0, 7);
    Print("%s", destination);    //    the output is hello my friends
}
```

条件

サポートされる最小バージョン	iA Studio 1.3

3.13 StringToDouble

目的

文字列を double 型に変換します

構文

```
double StringToDouble(
        char *str
);
```

パラメーター

str [in] 文字列

戻りの値

変換された浮動小数点値

例

```
void main() {
        double v = StringToDouble("1.234");    //    v = 1.234
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

3.14 MemoryCopy

目的

ソースメモリからコピー先メモリにデータのバイトをコピーする。

構文

```
int MemoryCopy(
        void *destination,
        void *source,
        int byte_num
);
```

パラメーター

destination [out]	データの結果を受け取るバッファーへのポインター。
source [in]	コピーするデータ
byte_num [in]	コピーするバイト数

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
void main() {

    int array1[5] = {1, 2, 3, 4, 5};
    int array2[5] = {11, 22, 33, 44, 55};
    int array3[5] = {345, 456, 567, 678, 789};

    MemoryCopy(array1, array2, sizeof(array2));
    //  now values in array1 are 11, 22, 33, 44, 55

    MemoryCopy(array1, array3, sizeof(int)*3);
    //  now values in array1 are 345, 456, 567, 44, 55

    MemoryCopy(&array1[3], &array3[3], sizeof(int)*2);
    //  now values in array1 are 345, 456, 567, 678, 789
}
```

条件

サポートされる最小バージョン	iA Studio 1.3

3.15 MemorySet

目的

宛先メモリの最初のバイト数を特定の値に設定します。

構文

```
int MemorySet(
    void *destination,
    int value,
    int byte_num
);
```

パラメーター

destination [out]	データの結果を受け取るバッファーへのポインター。
value [in]	設定する値。int として渡されますが、関数はこの値の char 変換でメモリを埋めます。
byte_num [in]	設定するバイト数

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
void main() {

    int array1[5] = {1, 2, 3, 4, 5};

    MemorySet(array1, 0, sizeof(array1));
    //  now values in array1 are 0, 0, 0, 0, 0

    MemorySet(array1, 1, sizeof(int));
    //  now values in array1 are 16843009, 0, 0, 0, 0
    //  16843009 = 0x01010101
}
```

条件

サポートされる最小バージョン	iA Studio 1.3

3.16 IsMemoryEqual

目的

2 つのメモリブロックが同じかどうかを確認します。

構文

```
int IsMemoryEqual(
        void *memory_ptr1,
        void *memory_ptr2,
        int byte_num
);
```

パラメーター

memory_ptr1 [in]	メモリブロック 1
memory_ptr2 [in]	メモリブロック 2
byte_num [in]	設定するバイト数

戻りの値

2 つのメモリブロックが同じ場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

例

```
void main() {

        int array1[5] = {1, 2, 3, 4, 5};
        int array2[5] = {1, 2, 3, 44, 55};
        int is_equal = false;
        is_equal = IsMemoryEqual(array1, array2, sizeof(array2));
        //    is_equal = false

        is_equal = IsMemoryEqual(array1, array2, sizeof(int)*3);
        //    is_equal = true
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

(このページはブランクになっています)

4. 演算機能

4.1 sin

目的

x ラジアンのサインを取得します。

構文

```
double sin(
    double x
);
```

パラメーター

x [in]　　ラジアンで角度を表す値
　　1 ラジアンは 180/π度に相当します。

戻りの値

x ラジアンのサイン

例

```
void main() {
    Print("sine of 30.0 degrees is %f.", sin(30.0 * PI / 180));
    //  Sine of 30.0 degrees is 0.5.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.2 cos

目的

x ラジアンのコサインを取得します。

構文

```
double cos(
    double x
);
```

パラメーター

x [in] ラジアンで角度を表す値。
1 ラジアンは 180/π度に相当します。

戻りの値

x ラジアンのコサイン

例

```
void main() {
    Print("cosine of 60.0 degrees is %f.", cos(60.0 * PI / 180));
    //  Cosine of 60.0 degrees is 0.5.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.3 tan

目的

x ラジアンのタンジェントを取得します。

構文

```
double tan(
    double x
);
```

パラメーター

x [in] ラジアンで角度を表す値。
1 ラジアンは 180/π度に相当します。

戻りの値

x ラジアンのタンジェント

例

```
void main() {
    Print("tangent of 45.0 degrees is %f.", tan(45.0 * PI / 180));
    //  Tangent of 45.0 degrees is 1.0.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.4 asin

目的

x の逆正弦を取得します。三角関数では、逆正弦は正弦の逆演算です。

構文

```
double asin(
        double x
);
```

パラメーター

x [in]　　[-1, +1] の間隔の値。

戻りの値

[-π/2, +π/2]ラジアンの間隔での x のアークサイン

1 ラジアンは 180/π度に相当します

備考

x が範囲外の場合、戻り値は定義できません。

例

```
void main() {
        Print("arc sine of 0.5 is %f degrees", asin(0.5) * 180.0 / PI);
        //    Arc sine of 0.5 is 30.0 degrees.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.5 acos

目的

x のアークコサインを取得します。三角関数では、アークコサインはコサインの逆演算です。

構文

```
double acos(
        double x
);
```

パラメーター

x [in]　　[-1, +1] の間隔の値

戻りの値

[0, π]ラジアンの間隔での x のアークコサイン。
1 ラジアンは 180/π度に相当します。

備考

x が範囲外の場合、戻り値は定義できません。

例

```
void main() {
        Print("arc cosine of 0.5 is %f degrees", acos(0.5) * 180.0 / PI);
        //    Arc cosine of 0.5 is 60.0 degrees.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.6 atan

目的

x のアークタンジェントを取得します。三角関数では、アークタンジェントはタンジェントの逆演算です。

構文

```
double atan(
        double x
);
```

パラメーター

x [in]　　[-1, +1]の間隔の値

戻りの値

[0, π]ラジアンの間隔での x のアークタンジェント
1 ラジアンは 180/π度に相当します。

備考

符号があいまいであるため、関数はタンジェント値だけでは角度がどの象限に収まるかを確実に判断できません。代わりに小数パラメーターを取る代替手段については、セクション 4.7 atan2 を参照してください。

例

```
void main() {
        Print("arc tangent of 1.0 is %f degrees", atan(1.0) * 180.0 / PI);
        //    Arc tangent of 1.0 is 45.0 degrees.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.7 atan2

目的

y/x のアークタンジェントを取得します。

構文

```
double atan2(
        double y,
        double x
);
```

パラメーター

y [in]　　Y 座標の比率を示す値

x [in]　　X 座標の比率を示す値

戻りの値

[-π, +π]ラジアンの間隔での y/x のアークタンジェント

1 ラジアンは 180/π度に相当します。

例

```
void main() {
        Print("arc tangent for (x=-10, y=10) is %f degrees",
                atan2(10, -10) * 180.0 / PI);
        //    Arc tangent for (x=-10, y=10) is 135 degrees.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.8 abs

目的

整数 x の絶対値を取得するには：| × |

構文

```
int abs(
        int x
);
```

パラメーター

x [in]　　整数

戻りの値

整数 x の絶対値

例

```
void main() {
        Print("absolute value of -3591 is %d.", abs(-3591));
        //    The absolute value of -3591 is 3591.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.9 fabs

目的

倍精度浮動小数点 x の絶対値を取得するには：| × |

構文

```
double fabs(
    double x
);
```

パラメーター

x [in]　　倍精度浮動小数点

戻りの値

倍精度浮動小数点 x の絶対値

例

```
void main() {
    Print("absolute value of -35.91 is %f.", fabs(-35.91));
    //  The absolute value of -35.91 is 35.91.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.10 ceil

目的

x を整数に切り上げます

構文

```
double ceil(
        double x
);
```

パラメーター

x [in]

戻りの値

x 以上の最小の整数

例

```
void main() {
        Print("ceil of 2.3 is %g", ceil(2.3));    //    Ceil of 2.3 is 3.0.
        Print("ceil of 3.8 is %g", ceil(3.8));    //    Ceil of 3.8 is 4.0.
        Print("ceil of -2.3 is %g", ceil(-2.3));    //    Ceil of -2.3 is -2.0.
        Print("ceil of -3.8 is %g", ceil(-3.8));    //    Ceil of -3.8 is -3.0.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.11 floor

目的

x を切り捨てて整数にします

構文

```
double floor(
        double x
);
```

パラメーター

x [in]

戻りの値

x 以下の最大の整数

例

```
void main() {
        Print("floor of 2.3 is %g", floor(2.3));    //    Floor of 2.3 is 2.0.
        Print("floor of 3.8 is %g", floor(3.8));    //    Floor of 3.8 is 3.0.
        Print("floor of -2.3 is %g", floor(-2.3));    //    Floor of -2.3 is -3.0.
        Print("floor of -3.8 is %g", floor(-3.8));    //    Floor of -3.8 is -4.0.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.12 ldexp

目的

x に 2 を掛けて y 乗した値を取得するには、x ＊ 2^y とします。

構文

```
double ldexp(
    double x,
    int    y
);
```

パラメーター

x [in]

y [in] 整数

戻りの値

x ＊ 2^y

結果の大きさが大きすぎる場合、表現可能な最大の double 値が返されます。

例

```
void main() {
    Print("0.95 * 2^4 = %f", ldexp(0.95, 4 ));    //   0.95 * 2^4 = 15.20
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.13 exp

目的

e の x 乗の値を取得するには：e^x

e は自然対数の底です。これは 2.71828 にほぼ等しいです。

構文

```
double exp(
        double x
);
```

パラメーター

x [in]

戻りの値

e^x

結果の大きさが大きすぎる場合、表現可能な最大の double 値が返されます。

例

```
void main() {
        Print("The exponential value of 5.0 is %f.", exp(5.0));
        //    The exponential value of 5.0 is 148.413159.
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.14 pow

目的

x を y 乗した値を取得するには：x^y

構文

```
double pow(
    double x,
    double y
);
```

パラメーター

x [in]

y [in]

戻りの値

x^y

結果の大きさが大きすぎる場合、表現可能な最大の double 値が返されます。

備考

(1) x が有限の負であり、y が有限だが整数でない場合、戻り値は定義できません。

(2) x と y の両方が 0 の場合、戻り値は定義できません。

(3) x が 0 で y が負の場合、戻り値は定義できません。

例

```
void main() {
    Print("7.0 ^ 3.0 = %f", pow(7.0, 3.0));    //  7.0 ^ 3.0 = 343.0
    Print("4.73 ^ 12.0 = %f", pow(4.73, 12.0));    //  4.73 ^ 12.0 = 125410439.217423
    Print("32.01 ^ 1.54 = %f", pow(32.01, 1.54));    //  32.01 ^ 1.54 = 208.036691
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.15 log

目的

x の自然対数(基数 e)を取得します。

構文

```
double log(
        double x
);
```

パラメーター

x [in]

戻りの値

x の自然対数(基数 e)

備考

(1) x が負またはゼロの場合、戻り値は定義できません。

(2) 常用対数(10 を底とする)については、セクション 4.16 log10 を参照してください。

例

```
void main() {
        Print("log(5.5) = %f", log(5.5));    //    log(5.5) = 1.704748
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.16 log10

目的

x の対数(10 を底とする)を取得します。

構文

```
double log10(
        double x
);
```

パラメーター

x [in]

戻りの値

x の対数(底 10)

備考

x が負またはゼロの場合、戻り値は定義できません。

例

```
void main() {
        Print("log10(1000.0) = %f", log10(1000.0));    //    log10(1000.0) = 3.0
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.17 sqrt

目的

x の平方根を取得します。

構文

```
double sqrt(
        double x
);
```

パラメーター

x [in]

戻りの値

x の平方根

備考

x が負の場合、戻り値は定義できません。

例

```
void main() {
        Print("sqrt(1024.0) = %f", sqrt(1024.0));    //    sqrt(1024.0) = 32.0
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.18 cbrt

目的

x の立方根を取得します。

構文

```
double cbrt(
        double x
);
```

パラメーター

x [in]

戻りの値

x の立方根

例

```
void main() {
        Print("cbrt (27.0) = %f", cbrt(27.0));    //    cbrt (27.0) = 3.0
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

4.19 hypot

目的

x と y を脚とする直角三角形の斜辺を取得します。

構文

```
double hypot(
        double x,
        double y
);
```

パラメーター

x [in]　　直角三角形の 1 つの辺

y [in]　　直角三角形のもう一方の辺

戻りの値

$(x^2 + y^2)$の平方根

結果の大きさが大きすぎる場合、表現可能な最大の double 値が返されます。

例

```
void main() {
        Print("hypot(3.0, 4.0) = %f.", hypot(3.0, 4.0));
        //    hypot(3.0, 4.0) = 5
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

5. 軸機能

5.1 概要

HIMC は、ポイントツーポイント(P2P)、JOG、および同期モーションを含む単軸モーションコマンドを提供します。ユーザーは、アプリケーションと要件に基づいて、関連するモーション機能を使用できます。図 5.1.1 に HIMC 軸モーション制御のフロー図を示します。モーション指令は内蔵のプロファイルジェネレーター(PG)を経由して基準位置を生成し、出力基準位置に誤差補正値を加算してサーボ駆動用の位置出力を生成します。

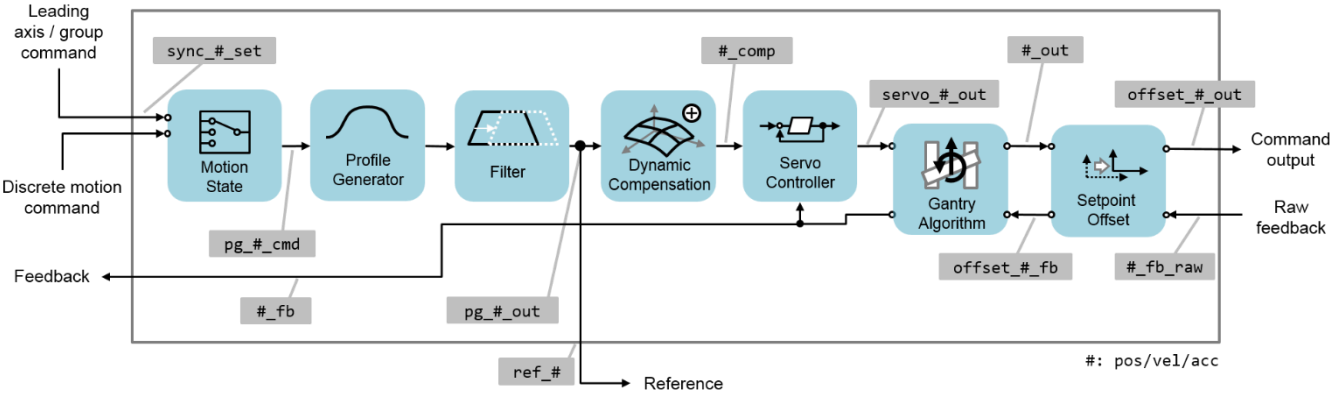


図 5.1.1

HIMC のプロファイルジェネレーターには、図 5.1.2 に示すように、S カーブ速度計画が組み込まれています。ユーザーは、プロファイルジェネレーターの最大速度、最大加速度、最大減速度、および滑らかな時間を設定できます。

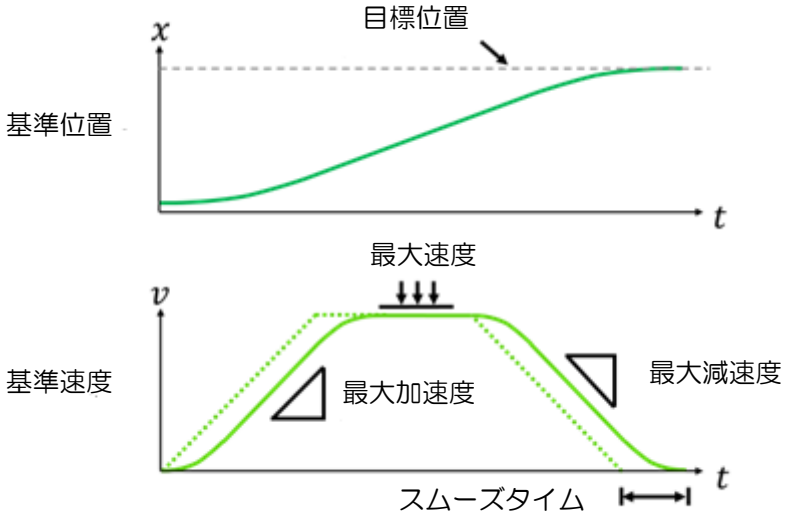


図 5.1.2

図 5.1.3 が示すように、滑らかな時間を追加すると基準速度が遅くなりますが、システムの安定性を高めるために高加速度によって生成されるジャークを効果的に下げることができます。ジャークと最大加速度、滑らかな時間の関係は次のようになります。

ジャーク = 最大加速度 / スムーズ時間 (Ts)

合計加速時間は、次の式で求められます。

合計加速時間 (T) = T カーブ加速時間 (Ta) + スムーズ時間 (Ts)

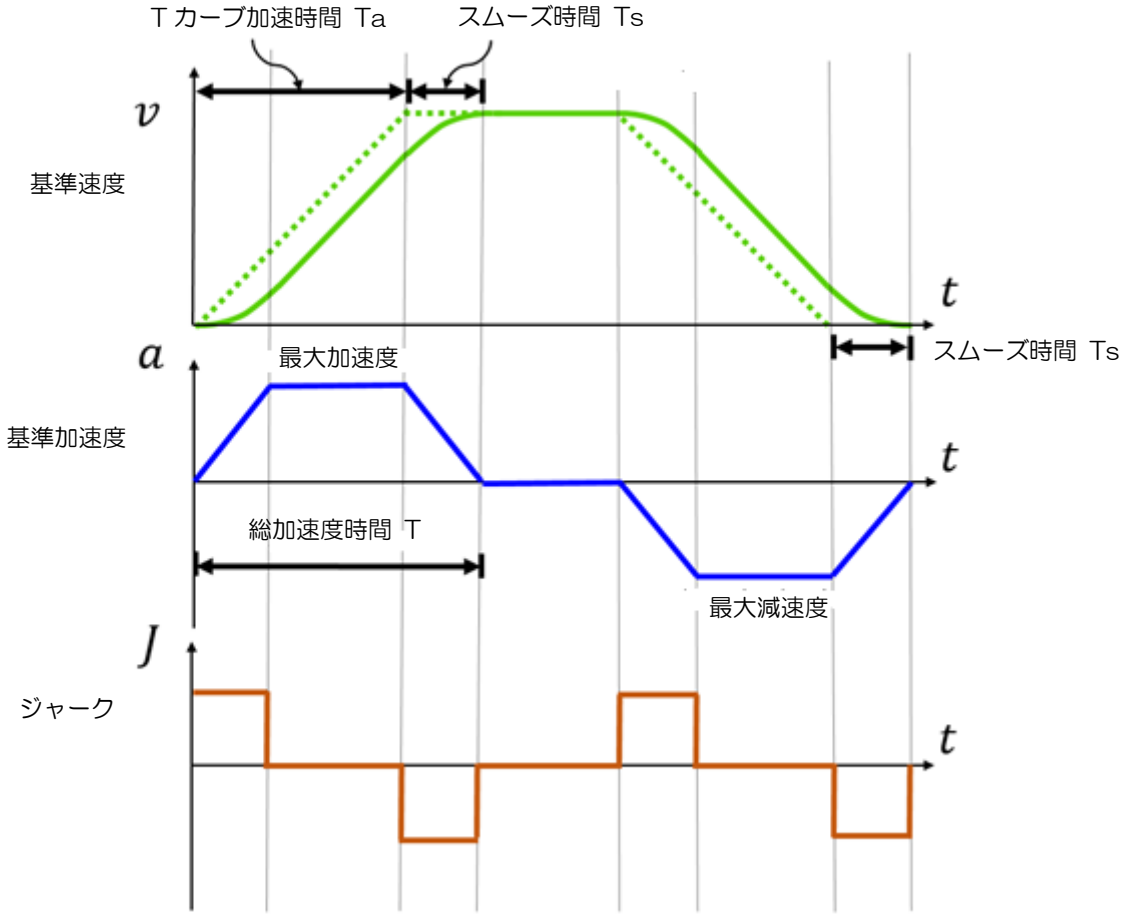


図 5.1.3

さらに、HIMC 軸モーションコマンドは、動的ターゲット位置と速度計画の変更をサポートします(オンザフライ修正)。軸移動中、ユーザーは軸の目標位置、最大速度、最大加減速度を変更できます。HIMC のプロファイルジェネレーターは、新しいコマンドとモーションパラメーターに基づいて、新しいターゲット位置に移動します。

軸の動作状態は、図 5.1.4 に示すように「移動中」と「インポジション」に分けられます。セクション I で軸位置計画コマンドを送信し続け、セクション II に入る前に終了します。コントローラーは、設定されたターゲット半径とデバウンス時間に基づいて、軸が所定の位置にあるかどうかを判断します。

デバウンス時間後に軸位置フィードバックが参照位置のターゲット半径内に残っている場合、軸位置はインポジションと見なされます。このとき、コントローラー内部で「軸インポジション」の状態が確立されます。ただし、デバウンス時間中に軸位置フィードバックが基準位置のターゲット半径を超えると、整定時間の計算がリセットされます。次に軸位置フィードバックがターゲット半径に入るまで、デバウンス時間のインポジション状態はチェックされません。

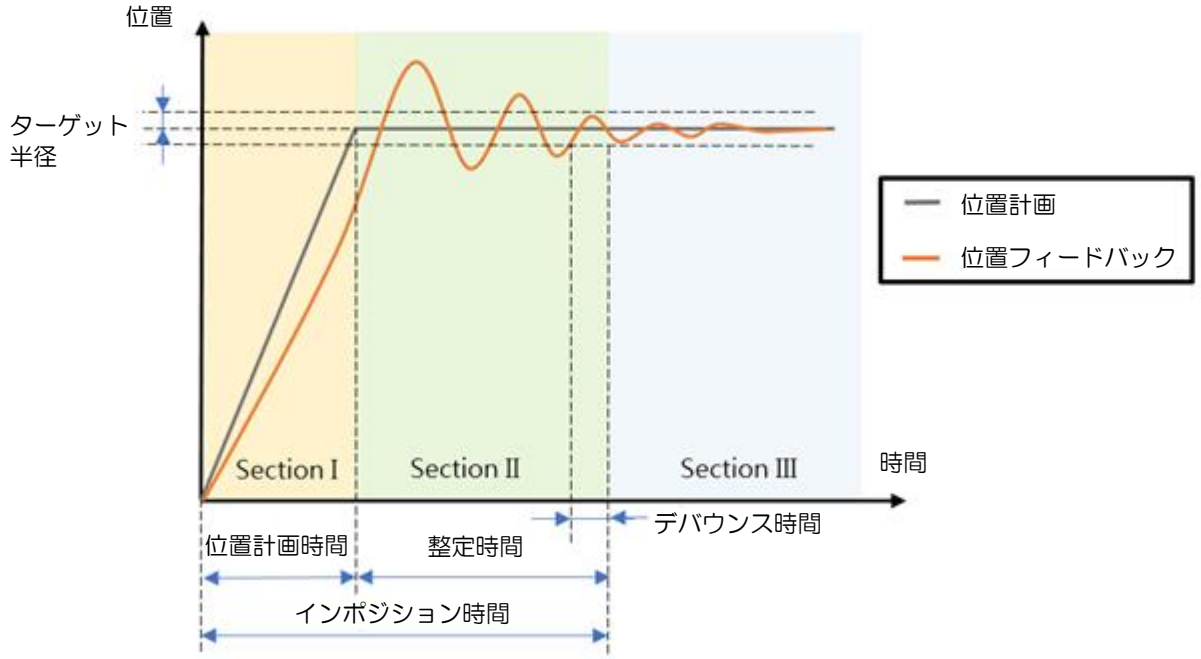


図 5.1.4

図 5.1.4 を参照すると、軸の動作状態は次のように記述されます。

1. セクション I：軸が動いていて、位置が合っていません。
2. セクション II：軸は動いていませんが、位置が合っていません。
3. セクション III：軸は動いておらず、所定の位置にあります。

軸が「同期」状態にある場合、軸グループまたはマスター軸は軸モーションのモーションプロファイルを生成します。軸自体はモーションプロファイルを生成しません。代わりに、軸グループまたはマスター軸によって計画された参照位置にのみ従います。

5.1.1 軸変数

軸変数は、モーションコマンド変数、プロファイルジェネレーター変数、状態変数の 3 つのカテゴリに分類されます。ユーザーは、iA Studio のスコープマネージャーを介して目的の変数を選択できます(「iA Studio ユーザーガイド」のセクション 4.8 を参照)。詳細な説明を表 5.1.1.1 から表 5.1.1.5 に示します。

表 5.1.1.1 軸のモーションコマンド変数

名称	変数	単位	説明
同期位置設定値	sync_pos_set	mm or deg	同期位置セットポイント。これは、軸が同期動作 (軸グループ、カムまたはギア操作など) にある場合に追従する目標位置です。
位置指令	pg_pos_cmd	mm or deg	プロファイルジェネレーターの位置コマンド。これは、軸が個別の動作 (2 点間) にある場合に追従する目標位置です。
基準位置	ref_pos	mm or deg	参考位置。これは、定義済みのモーションプロファイルに従ってプロファイルジェネレーターから生成された位置セットポイントです。
基準速度	ref_vel	mm/s or deg/s	基準速度。これは、定義済みのモーションプロファイルに従ってプロファイルジェネレーターから生成された速度設定値です。
基準加速度	ref_acc	mm/s^2 or deg/s^2	参照加速度。これは、定義済みのモーションプロファイルに従ってプロファイルジェネレーターから生成された加速度設定値です。
位置補正	pos_comp_set	mm or deg	位置補正値。動的誤差補正機能のエラーマップの出力です。機能が無効になっている場合はゼロになります。
補償された位置	pos_comp	mm or deg	補正位置指令値。基準位置と位置補正値の和です。
位置オフセット	pos_offset	mm or deg	位置オフセット。デフォルト値はゼロです。ユーザーがモーターを動かさずに新しい軸位置を設定すると、ゼロ以外になります。
位置出力	pos_out	mm or deg	位置出力。位置オフセットなしの軸位置指令です。
速度出力	vel_out	mm/s or deg/s	速度出力。速度オフセットなしの軸速度指令です。
加速度出力	acc_out	mm/s^2 or deg/s^2	加速出力。加速度オフセットなしの軸加速度指令です。
オフセット位置出力	offset_pos_out	mm or deg	オフセット付き位置出力。これは、位置オフセットを含む最終的に計算された軸位置コマンドです。値は単位「カウント」に変換され、対応するスレーブに送信されます。
生の位置フィードバック	pos_fb_raw	mm or deg	生の位置フィードバック。スレーブから読み取った位置フィードバックです。
オフセット位置フィードバック	offset_pos_fb	mm or deg	オフセット位置フィードバック。位置オフセット付きの位置フィードバックです。
位置フィードバック	pos_fb	mm or deg	位置フィードバック。これは軸座標系です。
速度フィードバック	vel_fb	mm/s or deg/s	速度フィードバック。これは軸座標系です。
位置エラー	pos_err	mm or deg	位置エラー。これは、位置出力と生の位置フィードバックの違いです。

名称	変数	単位	説明
速度エラー	vel_err	mm/s or deg/s	速度エラー。これは、速度出力と生の速度フィードバックの差です。
移動時間	movetime	ms	移動時間
整定時間	settlingtime	ms	整定時間

表 5.1.1.2 軸のプロファイルジェネレーター変数

名称	変数	単位	説明
最大プロファイル速度	max_vel	mm/s or deg/s	最大プロファイル速度。必ずしも達していない。
最大プロファイル加速度	max_acc	mm/s^2 or deg/s^2	最大プロファイル加速度。必ずしも達していない。
プロファイル減速度	max_dec	mm/s^2 or deg/s^2	最大プロファイル減速度。必ずしも達していない。
スムーズ時間	sm_factor	ms	スムーズな時間。入力範囲は 0 ～ 500 です。値を大きくすると、モーション中の機械的振動を減らすことができますが、合計モーション時間は影響を受けます。

表 5.1.1.3 軸の状態変数

名称	変数	単位	説明
障害状態	fault_status	N/A	軸のエラー状態；ビットの定義については、表 5.1.1.4 を参照してください。
モーション状態	motion_status	N/A	軸のモーション状態ビット定義については、表 5.1.1.5 を参照してください。

表 5.1.1.4 軸エラー状態のビット定義

ﾋﾞｯﾄ	名称	説明	デフォルトの応答
0	エラーストップ	「エラー停止」状態の軸	N/A
1	ドライバー障害	スレーブドライバーの障害	コントローラーが軸を無効にします。
2	位置エラー	位置誤差が保護限界を超えています	コントローラーは軸を無効にします
3	ハードウェア右側リミット	軸ハードウェア右リミットがトリガーされました	コントローラーがモーションを停止します。
4	ハードウェア左側リミット	軸ハードウェアの左リミットがトリガーされました	コントローラーがモーションを停止します
5	ソフトウェア右側リミット	軸ソフト右リミット発動	コントローラーがモーションを停止します
6	ソフトウェア左側リミット	軸ソフトウェアの左リミットがトリガーされました	コントローラーがモーションを停止します

表 5.1.1.5 軸モーション状態のビット定義

ビット	名称	説明	備考
0	Enabled	軸が有効になります	N/A
1	Moving	軸が動いています	セクション 5.1 を参照
2	In Position	軸はインポジションです	セクション 5.1 を参照
3	Synchronous	軸は「同期」状態にあります	軸が軸グループ内にあるか、同期モーションのスレーブ軸です。
4	Group	軸は軸グループにあります	セクション 8.1 を参照
5	Gantry	軸はガントリー軸です	セクション 7.1 を参照
6	Input Shape	入力シェイプフィルターを有効にします	セクション 15.1 を参照
7	VSF	VSF フィルターを有効にします	セクション 15.1 を参照
8	Gear	軸は電子ギアスレーブ軸です	セクション 6.1 を参照
9	Cam	軸は電子カムスレーブ軸です	まだサポートされていません。
10	Accelerating	軸は加速しています	セクション 5.1 を参照
11	Homed	軸が原点復帰を完了します	N/A

5.2 軸モーション制御

5.2.1 Enable

目的
軸を有効にします

構文

```
int Enable(
        int axis_id
);
```

パラメーター
axis_id [in]　　Axis index

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.2.2 Disable

目的
軸を無効にします

構文

```
int Disable(
        int axis_id
);
```

パラメーター
axis_id [in]　　Axis index

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考
軸のモーションキューがクリアされます。

条件

サポートされる最小バージョン	iA Studio 0.23

5.2.3 Reset

目的

軸が「エラー停止」状態のときにエラーをクリアします。

構文

```
int Reset(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.2.4 MoveAbs

目的

軸を絶対目標位置に移動します。

構文

```
int MoveAbs(
    int    axis_id,
    double pos
);
```

パラメーター

axis_id [in]	Axis index
pos [in]	絶対目標位置の値 パラメーターの単位: mm または deg

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.2.5 MoveRel

目的

相対距離で軸を移動します。

構文

```
int MoveRel(
    int    axis_id,
    double rel_dist
);
```

パラメーター

axis_id [in]	Axis index
rel_dist [in]	相対距離に対する値 パラメーターの単位: mm または deg

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.2.6 MoveVel

目的

特定の速度で終わりのない動きを開始すること。

構文

```
int MoveVel(
    int    axis_id,
    double vel
);
```

パラメーター

axis_id [in]	Axis index
vel [in]	特定の速度の値 モーションの方向は正負の値で決まります パラメーターの単位：mm/s または deg/s

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.2.7 MoveTrq

目的

特定のトルクで終わりのない運動を開始すること。

構文

```
int MoveTrq(
    int     axis_id,
    double torque_cmd
);
```

パラメーター

axis_id [in]	Axis index
torque_cmd [in]	トルク指令 パラメーター単位：N・m

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

(1) この機能は、「Profile Torque」モードにのみ適用されます。

(2) トルク指令がモーターの連続トルクより大きい場合、モーターは連続トルクの値で動きます。

条件

サポートされる最小バージョン	iA Studio 2.0

5.2.8 MovePVT

目的

指定された位置 (P)、速度 (V)、時間 (T)に基づいて、指定された位置に軸を移動します。

構文

```
int MovePVT(
    int axis_id,
    double *time,
    double *pos,
    double *vel,
    int num_pt
);
```

パラメーター

axis_id [in] Axis index

time [in] ユーザーが指定した時間配列へのポインター。
パラメーター単位：ms

pos [in] ユーザーが指定した位置配列へのポインター。
パラメーターの単位: mm または deg

vel [in] ユーザーが指定した速度配列へのポインター
パラメーターの単位: mm/s または deg/s

num_pt [in] PVT モーションポイントの数。その最大値は 50 です
時間の長さ、位置、および速度の配列は、このパラメーターと同じでなければなりません。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
void main() {
    int axis_id = 0;
    //  Position, velocity and time array of PVT motoin
    double point[6] = {0.0, 153.333, 42.123, 161.21, 177.0, 83.333};
    double velocity[6] = {0.0, 1000.0, 800.0, 1660.0, 450.0, 0.0};
    double time[6] = {0.0, 2000.0, 3000.0, 4000.0, 6000.0, 11000.0};
    Enable(axis_id);
    Till(IsEnabled(axis_id));
    //  Start PVT motion
    MovePVT(axis_id, time, point, velocity, 6);
}
```

条件

サポートされる最小バージョン	iA Studio 2.0

5.2.9 Stop

目的
軸の動きを止めます

構文

```
int Stop(
        int axis_id
);
```

パラメーター
axis_id [in]　　Axis index

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考
軸のモーションキューがクリアされます。

条件

サポートされる最小バージョン	iA Studio 0.23

5.2.10 Halt

目的

軸の動きを停止するには；その速度は 0 に設定されます。

構文

```
int Halt(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

軸が所定の位置にない場合、軸は動き続けます。

条件

サポートされる最小バージョン	iA Studio 1.3

5.2.11 Resume

目的

「停止」状態から軸の動きを再開すること。

構文

```
int Resume(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

5.3 軸設定

5.3.1 GetMaxVel

目的
軸の最大プロファイル速度を取得します。

構文

```
double GetMaxVel(
    int axis_id
);
```

パラメーター
axis_id [in]　　Axis index

戻りの値
軸の最大プロファイル速度
単位：mm/s または deg/s

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.2 SetVel

目的

軸の最大プロファイル速度を設定します。

構文

```
int SetVel(
    int    axis_id,
    double vel
);
```

パラメーター

axis_id [in]	Axis index
vel [in]	軸の新しい最大プロファイル速度。 パラメーターの単位: mm/s または deg/s 入力範囲：ゼロ以外の正の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.3 GetMaxAcc

目的

軸の最大プロファイル加速度を取得します。

構文

```
double GetMaxAcc(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸の最大プロファイル加速度

単位: mm/s^2 または deg/s^2

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.4 SetAcc

目的

軸の最大プロファイル加速度を設定します。

構文

```
int SetAcc(
    int    axis_id,
    double acc
);
```

パラメーター

axis_id [in]	Axis index
acc [in]	軸の新しい最大プロファイル加速度。 パラメーター単位: mm/s² または deg/s² 入力範囲：ゼロ以外の正の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.5 SetAccTime

目的

軸の加速時間を設定します。

構文

```
int SetAccTime(
    int    axis_id,
    double acc_time
);
```

パラメーター

axis_id [in]	Axis index
acc_time [in]	軸の加速時間 パラメーター単位：ms 入力範囲：ゼロ以外の正の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

5.3.6 GetMaxDec

目的

軸の最大プロファイル減速度を取得します。

構文

```
double GetMaxDec(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸の最大プロファイル減速度

単位：mm/s^2 または deg/s^2

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.7 SetDec

目的

軸の最大プロファイル減速度を設定します。

構文

```
int SetDec(
    int    axis_id,
    double dec
);
```

パラメーター

axis_id [in]	Axis index
dec [in]	軸の新しい最大プロファイル減速度。 パラメーター単位：mm/s² または deg/s² 入力範囲：ゼロ以外の正の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.8 SetDecTime

目的

軸の減速時間を設定します。

構文

```
int SetDecTime(
    int    axis_id,
    double dec_time
);
```

パラメーター

axis_id [in]	Axis index
dec_time [in]	軸の減速時間。 パラメーター単位：ms 入力範囲：ゼロ以外の正の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

5.3.9 GetKillDec

目的

軸のキル減速度を取得します。

構文

```
double GetKillDec(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸のキル減速度

単位：mm/s² または deg/s²

条件

サポートされる最小バージョン	iA Studio 1.3

5.3.10 SetKillDec

目的

軸のキル減速度を設定します。

構文

```
int SetKillDec(
    int    axis_id,
    double kill_dec
);
```

パラメーター

axis_id [in]	Axis index
kill_dec [in]	軸の新しいキル減速 パラメーター単位: mm/s² または deg/s² 入力範囲：ゼロ以外の正の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

5.3.11 GetSWRL

目的

軸のソフトリミット位置を取得します。

構文

```
double GetSWRL(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸のソフトウェア右リミット位置。

単位：mm または deg

条件

サポートされる最小バージョン	iA Studio 1.1

5.3.12 SetSWRL

目的

軸のソフトウェア右リミット位置を設定します。

構文

```
int SetSWRL(
    int    axis_id,
    double right_limit_pos
);
```

パラメーター

axis_id [in]	Axis index
right_limit_pos [in]	軸の新しいソフトウェア右リミット位置 パラメーター単位：mm または deg

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

5.3.13 GetSWLL

目的

軸のソフト左リミット位置を取得します

構文

```
double GetSWLL(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸のソフトウェア左リミット位置。
単位：mm または deg

条件

サポートされる最小バージョン	iA Studio 1.1

5.3.14 SetSWLL

目的

軸のソフトウェア左リミット位置を設定します。

構文

```
int SetSWLL(
    int    axis_id,
    double left_limit_pos
);
```

パラメーター

axis_id [in]	Axis index
left_limit_pos [in]	軸の新しいソフトウェア左リミット位置 パラメーター単位：mm または deg

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

5.3.15 GetSMTime

目的

軸のプロファイルの滑らかな時間を取得します。

構文

```
double GetSMTime(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸のプロファイルスムーズ時間。

単位：ms

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.16 SetSMTime

目的

軸のプロファイルスムーズ時間を設定します。

構文

```
int SetSMTime(
    int    axis_id,
    double smooth_time
);
```

パラメーター

axis_id [in]	Axis index
smooth_time [in]	軸の新しいプロファイルの滑らかな時間 パラメーター単位：ms 入力範囲：0 ～ 500

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

軸が移動中の場合、この機能は適用されません。

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.17 GetMoveTime

目的

軸の移動時間を取得します。

構文

```
double GetMoveTime(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の移動時間

単位：ms

条件

サポートされる最小バージョン	iA Studio 0.24

5.3.18 GetSettlingTime

目的

軸の整定時間を取得します。

構文

```
double GetSettlingTime(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の整定時間

単位：ms

条件

サポートされる最小バージョン	iA Studio 0.24

5.3.19 SetPos

目的

軸の位置を設定し、原点オフセットを変更します。

構文

```
int SetPos(
    int    axis_id,
    double pos
);
```

パラメーター

axis_id [in]	Axis index
pos [in]	軸の現在位置の値 パラメーターの単位：mm または deg

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

軸が「同期」状態、軸グループに追加、または「エラー」状態にある場合、この機能は適用されません。

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.20 GetPosFb

目的

軸の位置フィードバックを取得します。

構文

```
double GetPosFb(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸の位置フィードバック

単位：mm または deg

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.21 GetPosOffset

目的

軸の位置オフセットを取得します

構文

```
double GetPosOffset(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の位置オフセット

単位：mm または deg

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.22 GetPosErr

目的

軸の位置誤差を取得します

構文

```
double GetPosErr(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸の位置誤差

単位：mm または deg

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.23 GetVelFb

目的

軸の速度フィードバックを取得します

構文

```
double GetVelFb(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の速度フィードバック

単位：mm/s または deg/s

条件

サポートされる最小バージョン	iA Studio 1.4

5.3.24 GetVelErr

目的

軸の速度誤差を取得します

構文

```
double GetVelErr(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸の速度誤差

単位：mm/s または deg/s

条件

サポートされる最小バージョン	iA Studio 1.4

5.3.25 GetCurrFb

目的
軸の現在のフィードバック(二乗値)を取得します

構文

```
double GetCurrFb(
    int axis_id
);
```

パラメーター
axis_id [in] Axis index

戻りの値
軸の現在のフィードバック(二乗値)
単位：A^2

備考
現在のフィードバックを取得するには、まず「CurABS」を PDO 通信オブジェクトとして構成する必要があります。(「iA Studio ユーザーガイド」のセクション 4.13 を参照)。PDO オブジェクトが構成されていない場合、デフォルト値 0 が返されます。

条件

サポートされる最小バージョン	iA Studio 1.4

5.3.26 GetRefPos

目的

軸の基準位置を取得します

構文

```
double GetRefPos(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の基準位置

単位：mm または deg

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.27 GetRefVel

目的

軸の基準速度を取得します

構文

```
double GetRefVel(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の基準速度

単位：mm/s または deg/s

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.28 GetRefAcc

目的

軸の基準加速度を取得します

構文

```
double GetRefAcc(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の基準加速度

単位：mm/s² または deg/s²

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.29 GetPosOut

目的

コントローラーからスレーブドライブに送信された軸の位置コマンド出力を取得します

構文

```
double GetPosOut(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の位置指令出力

単位：mm または deg

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.30 GetVelOut

目的

コントローラーからスレーブドライブに送信された軸の速度コマンド出力を取得します

構文

```
double GetVelOut(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の速度コマンド出力

単位：mm/s または deg/s

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.31 GetAccOut

目的

コントローラーからスレーブドライブに送信された軸の加速度コマンド出力を取得します

構文

```
double GetAccOut(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の加速度指令出力

単位：mm/s^2 または deg/s^2

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.32 IgnoreHWL

目的

ハードウェア制限保護の警告メッセージを無視する

構文

```
int IgnoreHWL(
        int axis_id,
        int cmd
);
```

パラメーター

axis_id [in]	Axis index
cmd [in]	メッセージを無視するには、「1」に設定します。 メッセージを復元するには、「0」に設定します（初期値）。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.33 IgnoreSWL

目的

ソフトウェア制限保護の警告メッセージを無視する

構文

```
int IgnoreSWL(
    int axis_id,
    int cmd
);
```

パラメーター

axis_id [in]	Axis index
cmd [in]	メッセージを無視するには、「1」に設定します。 メッセージを復元するには、「0」に設定します（初期値）。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.34 IgnorePE

目的

位置偏差制限の警告メッセージを無視する

構文

```
int IgnorePE(
        int axis_id,
        int cmd
);
```

パラメーター

axis_id [in]	Axis index
cmd [in]	メッセージを無視するには、「1」に設定します。 メッセージを復元するには、「0」に設定します（初期値）。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

5.3.35 GetAxisNum

目的
コントローラーに接続されている軸数を取得します。

構文

```
int GetAxisNum();
```

パラメーター
N/A

戻りの値
コントローラーに接続されている軸の数

条件

サポートされる最小バージョン	iA Studio 0.23

5.3.36 SetVelScale

目的

軸運動の速度スケールを設定します

構文

```
int SetVelScale(
    int    axis_id,
    double vel_scale
);
```

パラメーター

axis_id [in]	Axis index
vel_scale [in]	軸運動の新しい速度スケール 入力範囲：0 ～ 100

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.4

5.3.37 GetVelScale

目的

軸運動の速度スケールを取得します

構文

```
double GetVelScale(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸の動きの速度スケール。その範囲は 0 ～ 100 です。

条件

サポートされる最小バージョン	iA Studio 1.4

5.3.38 SetRollover

目的

軸の位置ロールオーバー値を設定します

構文

```
int SetRollover(
    int    axis_id,
    double rollover_val
);
```

パラメーター

axis_id [in]	Axis index
rollover_val [in]	軸の位置ロールオーバー値 パラメーター単位：mm または deg 入力範囲：ゼロ以外の正の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

(1) パラメーター「rollover_val」が 0 に設定されている場合、関数はクローズされます。
(2) この機能は、軸が無効になっている場合にのみ適用されます。
(3) 軸が軸グループに追加されている場合、この機能は適用されません。

条件

サポートされる最小バージョン	iA Studio 1.4

5.3.39 GetRolloverTurns

目的

軸がロールオーバー モードの場合の回転数を取得します。

構文

```
int GetRolloverTurns(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸がロールオーバーモードのときの回転数

条件

サポートされる最小バージョン	iA Studio 1.4

5.3.40 SetOpMode

目的

軸の動作モードを設定します。

構文

```
int SetOpMode(
    int     axis_id,
    int     op_mode
);
```

パラメーター

axis_id [in]	Axis index
op_mode [in]	軸の新しい動作モード 入力範囲： 1(PP), 3(PV), 4(PT), 8(CSP), 9(CSV), 10(CST)

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

備考

この機能を使用する場合、ユーザーはオブジェクト 0x6060 (動作モード) を PDO として設定する必要があります。

条件

サポートされる最小バージョン	iA Studio 3.0

5.3.41 SetBufferMode

目的

軸のバッファモードを設定します。

構文

```
int SetBufferMode(
    int     axis_id,
    int     buf_mode
);
```

パラメーター

axis_id [in]	Axis index
buf_mode [in]	軸の新しいバッファモード. 入力範囲： 0 (即時停止モード), 1(バッファモード), 2(連続動作モード)

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

5.4 軸の状態

5.4.1 IsEnabled

目的

軸の「有効」状態を問い合わせます

構文

```
int IsEnabled(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「有効」状態の場合、int 値 TRUE (1)を返します。 それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.2 IsMoving

目的

軸の「移動」状態を照会します。軸が移動している場合、PG(プロファイルジェネレーター)は新しい位置を出力し続けます。

構文

```
int IsMoving(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「移動」状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.3 IsInPos

目的

軸の「in-position」状態を照会します。軸が所定の位置にある場合、位置エラーは、特定の期間(debounce time)の間、エラーウィンドウ(ターゲット半径)内に保持されます。

構文

```
int IsInPos(
    int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「InPos」状態にある場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.4 IsErrorStop

目的

軸が「エラー停止」状態かどうかを問い合わせます。

構文

```
int IsErrorStop(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「ErrorStop」状態にある場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.5 IsGantry

目的

軸が「gantry」状態にあるかどうかを照会します。

構文

```
int IsGantry(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「Gantry」状態にある場合は、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.6 IsGrouped

目的

軸が軸グループにグループ化されているかどうかを照会します。

構文

```
int IsGrouped(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「グループ化」状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.7 IsSync

目的

軸が「同期」状態にあるかどうかを問い合わせます。軸が「同期」状態にある場合、軸はマスターのコマンドに従います。

構文

```
int IsSync(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「同期」状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.8 IsHWLL

目的

軸がハードウェアの左限界に達しているかどうかを照会します。

構文

```
int IsHWLL(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「HWLL」状態にある場合は、int 値 TRUE (1)が返されます。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.9 IsHWRL

目的

軸がハードウェアの右限界に達しているかどうかを照会します。

構文

```
int IsHWRL(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「HWRL」状態にある場合は、int 値 TRUE (1)が返されます。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.10 IsSWLL

目的

軸がソフトウェアの左リミットに達しているかどうかを問い合わせます。

構文

```
int IsSWLL(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸が「SWLL」状態にある場合は、int 値 TRUE (1)が返されます。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.11 IsSWRL

目的

軸がソフトウェアの右リミットに達しているかどうかを問い合わせます。

構文

```
int IsSWRL(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「SWRL」状態にある場合、TRUE (1)の int 値が返されます。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.12 IsDriveErr

目的

軸がドライバーアラームをトリガーするかどうかを問い合わせます。

構文

```
int IsDriveErr(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「DriveErr」状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 1.4

5.4.13 IsPosErr

目的

軸の位置誤差が保護限界を超えているかどうかを問い合わせます。

構文

```
int IsPosErr(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「PosErr」状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

備考

エラー保護限界は、コントローラー内の軸の位置エラーの許容範囲を示します。

条件

サポートされる最小バージョン	iA Studio 1.4

5.4.14 IsCompActive

目的

補正機能が有効かどうかを問い合わせます。

構文

```
int IsCompActive(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「CompActive」状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

5.4.15 IsAcc

目的

軸が加速しているかどうかを問い合わせます。

構文

```
int IsAcc(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「Acc」状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

6. シンクロモーション機能

6.1 概要

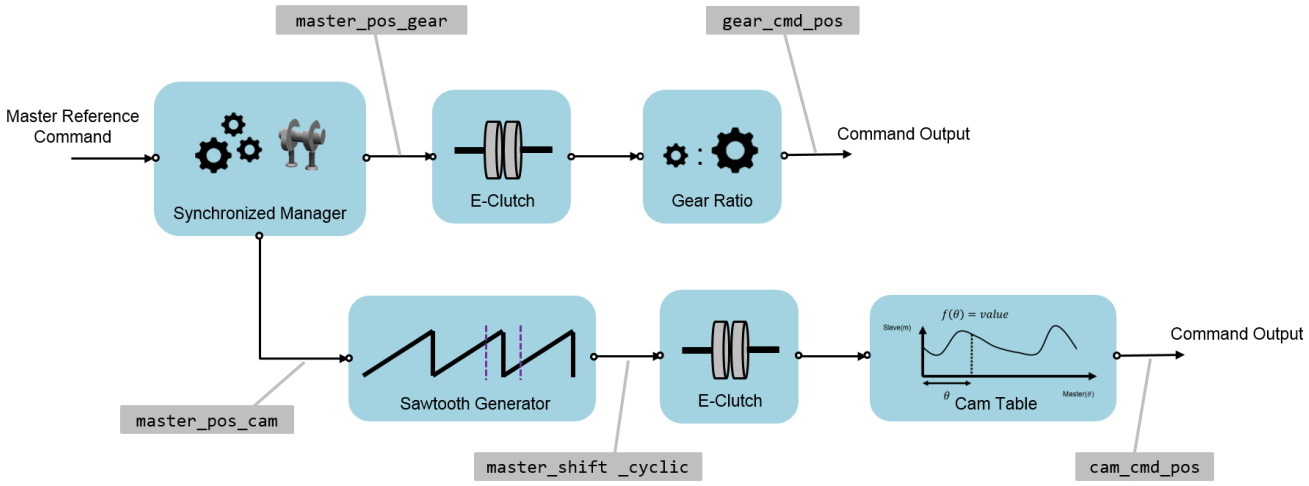


図 6.1.1

ユーザーは、ある軸と別の軸の間の同期モーションを定義できます。先行軸であるマスター軸が最初に位置コマンドを生成し、次にスレーブ軸がモーション構成に基づいてマスター軸を参照します。主従関係が一定であれば、動きは電子ギアです。一方、スレーブ軸がパターンに従う必要がある場合、モーションは電子カムです。図 6.1.2 では、軸 0 がマスター軸として機能し、軸 1、2、3、および 4 が先行します。軸 1、2、および 3 は電子ギアを採用し、軸 4 は電子カムを採用しています。

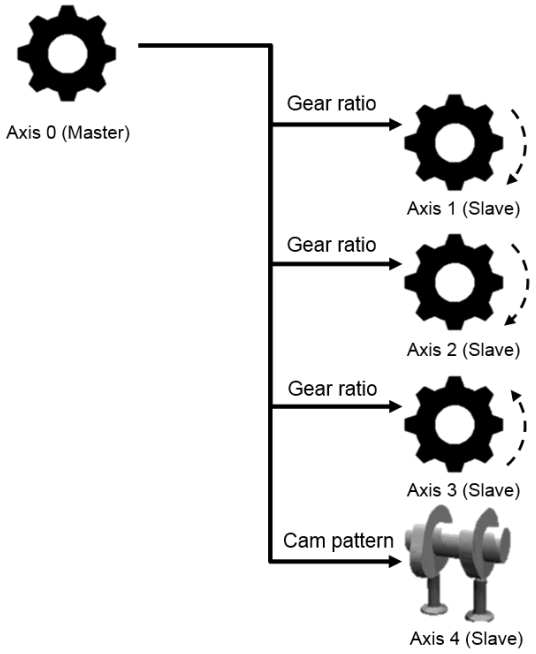


図 6.1.2

6.1.1 シンクロモーション変数

一般的な同期モーション変数を表 6.1.1.1 に示します。ユーザーは、iA Studio のスコープマネージャーを介して目的の変数を選択できます(「iA Studio ユーザーガイド」のセクション 4.8 を参照)。

表 6.1.1.1

名称	変数	単位	説明
元のマスター位置	master_pos_gear	mm または deg	マスターの位置指令
ギア指令位置	gear_cmd_pos	mm または deg	スレーブ軸は位置指令を出力します
ギア比	gear_ratio	mm または deg	ギア比

6.1.2 例

```
void main()
{

    double target = 100;
    double Gear_ratio[4]={1.0, 2.0, 4.0, -1.0};
    int master = 0;
    int slave[4]={1, 2, 3, 4};

    Enable(master);
    Enable(slave[0]);
    Enable(slave[1]);
    Enable(slave[2]);
    Enable(slave[3]);
    Till(IsEnabled(slave[0])&&IsEnabled(slave[1])&&
    IsEnabled(slave[2])&&IsEnabled(slave[3])&&IsEnabled(master))

    //  Couple two axes in a master-slave relationship
    EnableGear(master, slave[0]);
    EnableGear(master, slave[1]);
    EnableGear(master, slave[2]);
    EnableGear(master, slave[3]);
```

```
    //  Change slave axis' state from disengaged to engaged
    GearIn(master, slave[0], Gear_ratio[0]);
    GearIn(master, slave[1], Gear_ratio[1]);
    GearIn(master, slave[2], Gear_ratio[2]);
    GearIn(master, slave[3], Gear_ratio[3]);

    MoveAbs(master, target);
    Till(IsInPos(master));

    //  Change slave axis' state from engaged to disengaged
    GearOut(slave[0]);
    GearOut(slave[1]);
    GearOut(slave[2]);
    GearOut(slave[3]);

}
```

6.2 EnableGear

目的

2 つの軸をマスター、スレーブ関係で結合します。

構文

```
int EnableGear(
        int axis_master_id,
        int axis_slave_id
);
```

パラメーター

axis_master_id [in]　　マスター Axis index

axis_slave_id [in]　　スレーブ Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

この機能は、両方の軸が有効な場合にのみ適用されます。

条件

サポートされる最小バージョン	iA Studio 1.1

6.3 DisableGear

目的

マスタースレーブ関係から 2 つの軸を切り離して、2 つの独立した軸にする

構文

```
int DisableGear(
        int axis_slave_id
);
```

パラメーター

axis_slave_id [in]　　スレーブ Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

6.4 GearIn

目的
スレーブ軸の状態を解放から使用に変更します

構文

```
int GearIn(
        int axis_master_id,
        int axis_slave_id,
        double gear_ratio
);
```

パラメーター

axis_master_id [in]	マスター Axis index
axis_slave_id [in]	スレーブ Axis index
gear_ratio[in]	ギア比の値

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考
この機能は、両方の軸が有効な場合にのみ適用されます。

条件

サポートされる最小バージョン	iA Studio 1.1

6.5 GearOut

目的

スレーブ軸の状態を接続から未接続に変更します。

構文

```
int GearOut(
        int axis_slave_id
);
```

パラメーター

axis_slave_id [in]　　スレーブ Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

6.6 GetGearRatio

目的

スレーブ軸のギア比を取得します

構文

```
double GetGearRatio(
        int axis_slave_id
);
```

パラメーター

axis_slave_id [in]　　スレーブ Axis index

戻りの値

スレーブ軸のギア比

条件

サポートされる最小バージョン	iA Studio 1.3

6.7 IsInGear

目的

スレーブ軸が「engaged」状態にあるかどうかを問い合わせます。

構文

```
int IsInGear(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

スレーブ軸が「InGear」状態にある場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

6.8 IsGearMaster

目的

軸がマスター軸かどうかを問い合わせます

構文

```
int IsGearMaster(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が「GearMaster」状態にある場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

6.9 IsGearSlave

目的

軸がスレーブ軸かどうかを問い合わせます

構文

```
int IsGearSlave(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸が「GearSlave」状態にある場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

7. ガントリー機能

7.1 概要

ガントリー構成は、図 7.1.1 に示すように、右側 (RHS)軸と左側 (LHS)軸のペアを仮想直線軸とヨー軸のペアに変換します。ガントリー構成を確立した後、ユーザーは、RHS 軸に直線軸方向コマンドを与えて、RHS 軸と LHS 軸の両方を同じ方向に駆動し、LHS 軸にヨー軸方向の回転運動コマンドを与えることができます。

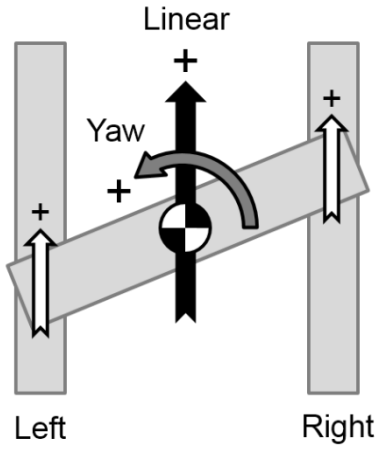


図 7.1.1

ガントリー構成では、直線軸とヨー軸の位置フィードバックは次のように定義されます。

$$Pos_{linear} = \frac{Pos_{RHS} + Pos_{LHS}}{2}; \qquad Pos_{yaw} = \frac{Pos_{RHS} - Pos_{LHS}}{2}$$

Pos_{linear}	：直線軸の位置フィードバック	Pos_{yaw}	：ヨー軸の位置フィードバック
Pos_{RHS}	：右軸の位置フィードバック	Pos_{LHS}	：左軸の位置フィードバック

図 7.1.2 は、直線軸、ヨー軸、RHS 軸、および LHS 軸の位置フィードバック回路図です。

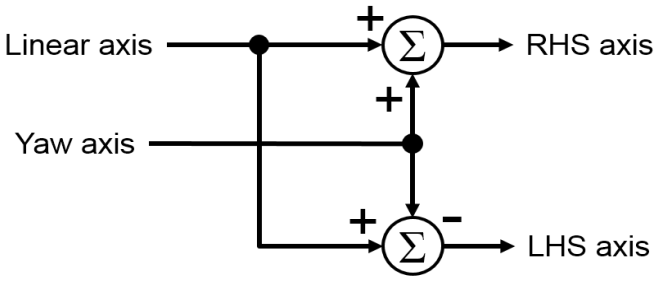


図 7.1.2

7.1.1 例

ガントリーペアをセットアップする方法は、次の HMPL タスクに示されています。

```
void main() {

    int axis_0 = 0;   //   user variable definition
    int axis_1 = 1;

    DisableGantryPair(axis_0);   //   Disable the existing gantry settings
    Till(!IsGantry(axis_0) && !IsGantry(axis_1));

    Enable(axis_0);
    Till(IsEnabled(axis_0));
    Disable(axis_0);
    Till(!IsEnabled(axis_0));

    Enable(axis_1);
    Till(IsEnabled(axis_1));
    Disable(axis_1);
    Till(!IsEnabled(axis_1));

    EnableGantryPair(axis_0, axis_1);
    Enable(axis_0);

    Till(IsEnabled(axis_0) && IsEnabled(axis_1));
    Till(IsGantry(axis_0) && IsGantry(axis_1));
}
```

7.2 EnableGantryPair

目的

ガントリーペアをセットアップします

構文

```
int EnableGantryPair(
        int lhs_axis_id,
        int rhs_axis_id
);
```

パラメーター

lhs_axis_id [in]　　左 Axis index

rhs_axis_id [in]　　右 Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

この機能は、両方の軸が無効になっている場合にのみ適用されます。

条件

サポートされる最小バージョン	iA Studio 0.23

7.3 DisableGantryPair

目的
ガントリーペアを分割します

構文

```
int DisableGantryPair(
        int axis_id
);
```

パラメーター
axis_id [in]　　ガントリーペアのいずれかの Axis index

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考
この機能は、両方の軸が無効になっている場合にのみ適用されます。

条件

サポートされる最小バージョン	iA Studio 0.23

7.4 GetGantryPairID

目的

任意のガントリー軸のガントリーペア ID を取得します

構文

```
int GetGantryPairID(
        int axis_id
);
```

パラメーター

axis_id [in]　　ガントリーペアのいずれかの Axis index

戻りの値

ガントリーペア ID

入力軸がガントリー軸でない場合は、-1 を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

7.5 IsGantryPair

目的

2 つの軸がガントリーペアであるかどうかを照会します

構文

```
int IsGantryPair(
        int axis_id_1
        int axis_id_2
);
```

パラメーター

axis_id_1 [in]　　Axis index 1

axis_id_2 [in]　　Axis index 2

戻りの値

2 つの軸が「GantryPair」状態にある場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

(このページはブランクになっています)

8. グループ機能

8.1 概要

HIMC は、LineAbs2D / 3D、LineRel2D / 3D、Arc2D、Circle2D などを含む、多軸直線および円弧同時補間機能の軸グループモーションコマンドを提供します。軸モーションコマンドと比較して、軸グループモーションコマンドは、グループ内の各軸の同期を保証します。各軸のモーションの開始時間と停止時間は同じであり、コントローラーは、ユーザーが指定した基準速度に基づいて各軸のモーション速度を調整します。

図 8.1.1 に HIMC 軸グループモーションコマンドのパラメータフロー図を示します。各軸の位置フィードバックは、順運動学の計算を通過するため、機械座標系での軸グループの位置フィードバック(直交位置フィードバック)が取得されます。ユーザーによって与えられたターゲットコマンドに基づいて、コントローラーは、軸グループのモーションプロファイル(図 8.1.2 が示すように)による空間内の補間コマンド(デカルト位置コマンド)を計画し、逆運動学を使用して各軸の対応する位置コマンドを計算します。

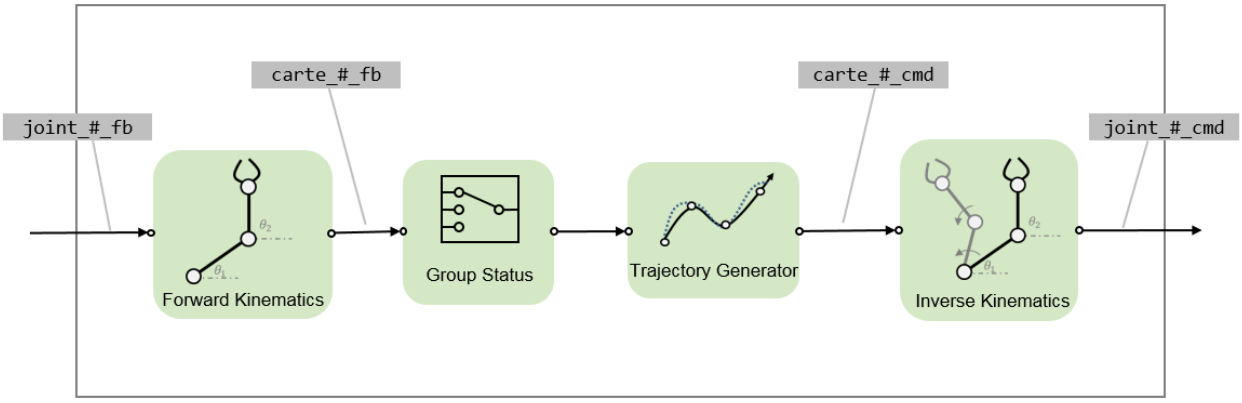


図 8.1.1

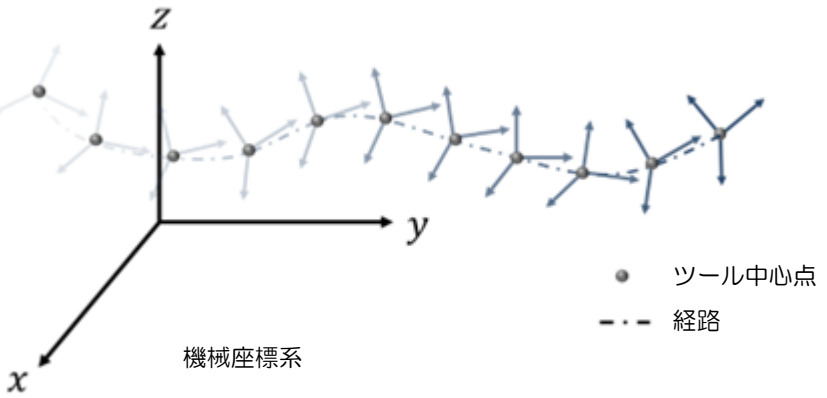


図 8.1.2

軸グループモーションコマンドでは、HIMC は空間内の各セグメントの移動距離を計算します。軸移動コマンドとは異なり、速度計画は空間内の軸グループの移動方向に沿って計画され、移動方向は移動コマンドの方向に基づいて変化します。

軸グループモーションコマンドは、軸モーションコマンドに似ています。また、図 8.1.3 に示すように、S カーブ速度計画も採用しています。空間内の軸群の動きは、並進と回転の 2 つの部分で構成されます。並進指令は XYZ の位置指令で構成され、回転指令は ABC の回転指令で構成されます。軸グループを使用すると、ユーザーは、プロファイルジェネレーターの最大速度、最大加速度、最大減速度、滑らかな時間を含む、平行移動と回転の速度計画パラメーターを設定できます。

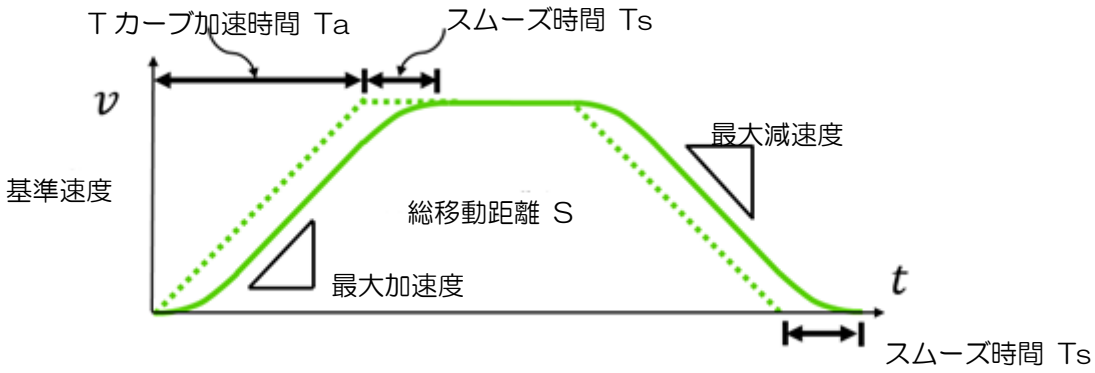


図 8.1.3

図 8.1.4 に示すように、各軸グループのモーションコマンドは 1 つのセグメントとして表示されます。動作中、各セグメントの並進コマンドと回転コマンド、およびユーザーが設定した速度計画パラメーターに基づいて、HIMC は並進コマンドと回転コマンドの移動時間を計算します。長い移動時間の速度計画パラメーターは、軸グループの送り速度として表示されます。移動時間の短い方は、送り速度指令で配分された動作指令に従って移動します。

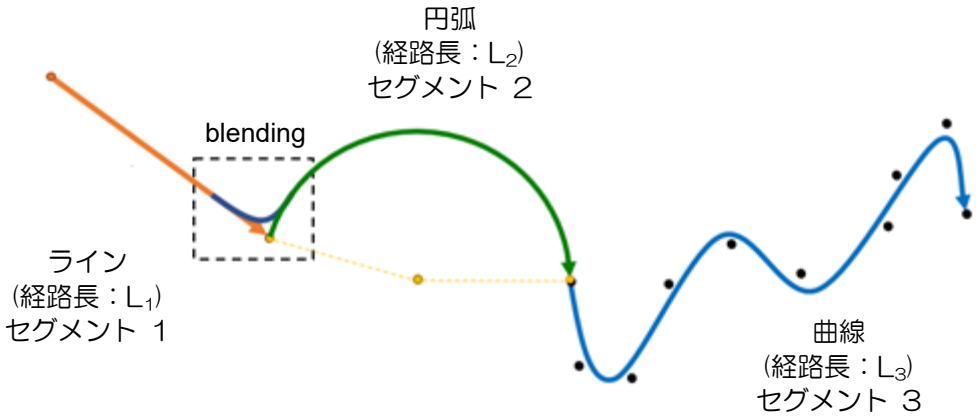


図 8.1.4

HIMC には、軸グループコマンド用の組み込みバッファーがあります。各モーションコマンドのセグメントは、このバッファーに配置されます。同時に最大 512 個のモーションコマンドを受け入れることが

できます。この容量制限を超えるモーションコマンドはコントローラーによって破棄され、エラーメッセージが表示されます。セグメントのモーションコマンド間では、ユーザーが設定したバッファーモードと Transition モードに基づいて、速度とパスが計画されます。選択したモードによって、計画された速度と経路が変更される場合があります。図 8.1.4 を例に取ると、各セグメントの速度ハンドオーバーを設定するためにバッファーモードが使用される場合、この軸モーションの全長は「S=L_1 (直線)+L_2 (円弧)+L_3 (曲線)」になります。詳細については、セクション 8.1.4 および 8.1.5 を参照してください。

軸グループのモーション状態は、軸のモーション状態に似ています。また、「移動」と「定位置」に分けることもできます。動作中は、図 5.1.4 に示すように、以下を含む 3 つのフェーズがあります：

1. 軸グループが動いていて、位置が合っていない。
2. 軸グループは動いていませんが、位置が合っていません。
3. 軸グループは移動しておらず、所定の位置にあります。

ターゲット半径とデバウンス時間を使用して軸がインポジションかどうかをチェックする軸モーションコマンドとは異なり、軸グループモーションコマンドは、グループ内のすべての軸がインポジションかどうかをチェックします。つまり、軸グループがインポジションの場合、グループ内のすべての軸が「インポジション」状態になります。

8.1.1 グループ変数

軸グループ変数は、モーションコマンド変数、プロファイルジェネレーター変数、状態変数の 3 つのカテゴリに分類されます。ユーザーは、iA Studio のスコープマネージャーを介して目的の変数を選択できます(「iA Studio ユーザー ガイド」のセクション 4.8 を参照)。詳細な説明を表 8.1.1.1 から表 8.1.1.5 に示します。

表 8.1.1.1 軸グループのモーションコマンド変数

名称	変数	単位	説明
直交座標コマンド	carte_pose_cmd	mm または deg	機械座標系(MCS)の軸グループの空間位置コマンド。[X Y Z A B C]の値を含む配列です。
直交速度指令	carte_vel_cmd	mm/s または deg/s	機械座標系(MCS)の軸グループの空間速度コマンド。[X Y Z A B C]の値を含む配列です。
直交座標フィードバック	carte_pose_fb	mm または deg	機械座標系(MCS)の軸グループの空間位置フィードバック。[X Y Z A B C]の値を含む配列です。
軸位置指令	joint_pos_cmd	mm または deg	軸座標系(ACS)における軸グループの軸位置指令配列です。
軸速度コマンド	joint_vel_cmd	mm/s または deg/s	軸座標系(ACS)における軸群の軸速度指令の配列です。
軸加速度指令	joint_acc_cmd	mm/s^2 または deg/s^2	軸座標系(ACS)における軸グループの軸加速度指令の配列です。
軸位置フィードバック	joint_pos_fb	mm または deg	軸座標系(ACS)における軸群の軸位置フィードバックの配列です。
直交位置誤差	carte_pose_err	mm または deg	機械座標系(MCS)の軸グループの空間位置エラー。[X Y Z A B C]の値を含む配列です。
参照グループの位置	grp_pg_pos	mm または deg	軸グループの基準位置。これは、軸グループコマンドのモーションプロファイルに従って、プロファイルジェネレーターから生成された位置セットポイントです
基準群速度	grp_pg_vel	mm/s または deg/s	軸グループの基準速度。これは、軸グループコマンドのモーションプロファイルに従って、プロファイルジェネレーターから生成された速度設定値です。
参照グループの加速度	grp_pg_acc	mm/s^2 または deg/s^2	軸グループの基準加速度。これは、軸グループコマンドのモーションプロファイルに従って、プロファイルジェネレーターから生成された加速度設定値です。

表 8.1.1.2 軸グループのプロファイルジェネレーター変数

名称	変数	単位	説明
グループ最大線形プロファイル速度	grp_lin_vel	mm/s	軸グループの最大線形プロファイル速度。 必ずしも達していない。
グループ最大線形プロファイル加速度	grp_lin_acc	mm/s^2	軸グループの最大線形プロファイル加速度。 必ずしも達していない。
グループ最大線形プロファイル減速度	grp_lin_dec	mm/s^2	軸グループの最大線形プロファイル減速度。 必ずしも達していない。
グループ リニア スムーズ タイム	grp_lin_sf	ms	軸グループの線形プロファイルの平滑化時間。入力範囲は 0 ～ 500 です。値を大きくすると、モーション中の機械的振動を減らすことができますが、合計モーション時間は影響を受けます。
グループ最大角プロファイル速度	grp_ang_vel	deg/s	軸グループの最大角プロファイル速度。 必ずしも達していない。
グループの最大角プロファイル加速度	grp_ang_acc	deg/s^2	軸グループの最大角度プロファイル加速度。 必ずしも達していない
グループの最大角度プロファイルの減速度	grp_ang_dec	deg/s^2	軸グループの最大角度プロファイル減速。 必ずしも達していない。
グループ角度スムーズ時間	grp_ang_sf	ms	軸グループの角度プロファイルの平滑化時間。入力範囲は 0 ～ 500 です。値を大きくすると、モーション中の機械的振動を減らすことができますが、合計モーション時間は影響を受けます。

表 8.1.1.3 軸グループの状態変数

名称	変数	単位	説明
グループ障害状態	grp_fault_status	N/A	軸グループのエラー状態; ビット定義については、表 8.1.1.4 を参照してください。
グループモーション状態	grp_motion_status	N/A	軸グループのモーション状態; ビット定義については、表 8.1.1.5 を参照してください。

表 8.1.1.4 軸グループエラー状態のビット定義

Bit	名称	説明	デフォルトの応答
0	Error Stop	「エラー停止」状態の軸グループ	N/A
1	Axis Fault	スレーブドライバーの障害	コントローラーは軸を無効にします。軸グループが同期していません。
2	Software Limit	軸ソフトウェアリミットがトリガーされた	コントローラーがモーションを停止します。軸グループが同期していません。

表 8.1.1.5 軸群モーション状態のビット定義

Bit	名称	説明	備考
0	Enabled	軸グループが有効になります	N/A
1	Moving	軸群が動いています	N/A
2	In Position	軸グループはインポジションです	軸グループ内のすべての軸がインポジションです
3	Input Shape	軸グループの入力形状フィルターを有効にします	セクション 15.1 を参照してください

8.1.2 座標系

表 8.1.2.1 は、軸座標系(ACS)、機械座標系(MCS)、製品座標系(PCS)、ワークピース座標系(WCS)、グローバル座標系、および座標オフセットを含む、HIMC の座標系の定義と説明を示しています。

表 8.1.2.1

HMPL 定義	説明
CS_ACS	軸座標系。 これは、個々のモーターの動きに関連しています。
CS_MCS	機械座標系。 (「ワールド座標系」または「ベース座標系」と呼ばれることもあります) これは機械の原点が固定された座標系であり、キネマティクス変換を介して ACS にリンクされています(セクション 8.1.3 を参照)。空間内の位置と方向を示すために、合計で 6 つの次元があります(3 つの並進、3 つの回転)。
CS_PCS	製品座標系 (CNC プログラムでは「プログラム座標系」)。 製品やワークに取り付けて、座標変換パラメーターを設定できます。
CS_WCS# (#=1~15)	ワーク座標系。 ワーク原点を設定するために使用され、最大 15 の独立したワーク座標系を提供します。デフォルトはオフセットなしです。Product Coordinate System に依存するため、座標変換パラメーターを設定できます。
CS_GLOBAL	グローバル座標系。 グローバルなゼロ点を設定するために使用され、各軸グループのグローバルな空間関係を確立できます。 まだサポートされていません。
CS_OFFSET	座標オフセット。 仮のゼロ点設定に使用します。デフォルトはオフセットなしです。つまり、オフセットの座標原点はマシンの座標原点です。Product Coordinate System に依存するため、座標変換パラメーターを設定できます。

図 8.1.2.1 に回転軸 2 軸のスカラロボットの ACS、MCS、PCS の関係例を示します。ACS と MCS は、順運動学と逆運動学によって変換されます(セクション 8.1.3 を参照)。一方、MCS と PCS には座標変換関係があります。座標系上の位置は、座標の並進と回転によって取得されます。

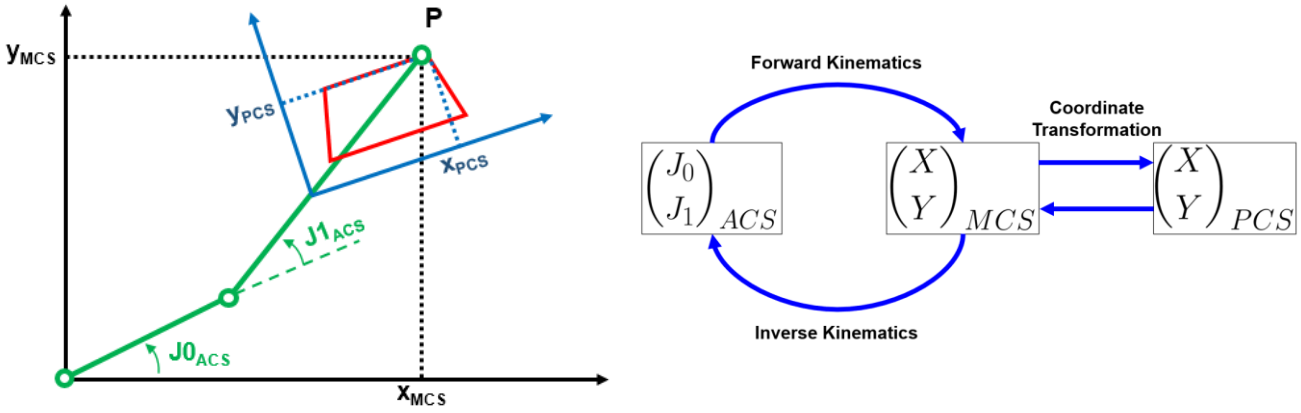


図 8.1.2.1

MCS を PCS に変換する場合、HIMC は必要に応じてマシンのワーク座標(WCS1～15)と座標オフセット(OFFSET)を設定できます。座標系の設定には、並進 3 自由度(X、Y、Z)と回転 3 自由度(A、B、C)を使用して空間での姿勢を示します。

HIMC は、固定角度で「Roll-Pitch-Yaw」回転規則を採用しています。図 8.1.2.2 が示すように、X 軸に沿って回転する自由度はロール、角度 A です。Y 軸に沿って回転する自由度はピッチ、角度 B です。Z 軸に沿って回転する自由度は、Yaw、角度 C です。この回転規則は、図 8.1.2.3 に示すように、スペース内のオブジェクトの向きを示すためにテイトブライアン角度で ZYX のシーケンスを使用するのと同じです。

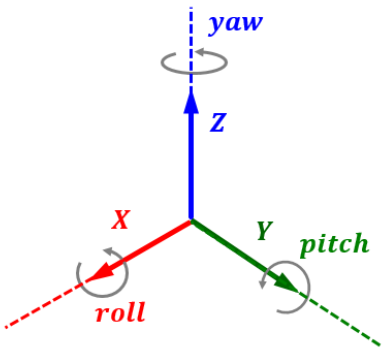


図 8.1.2.2

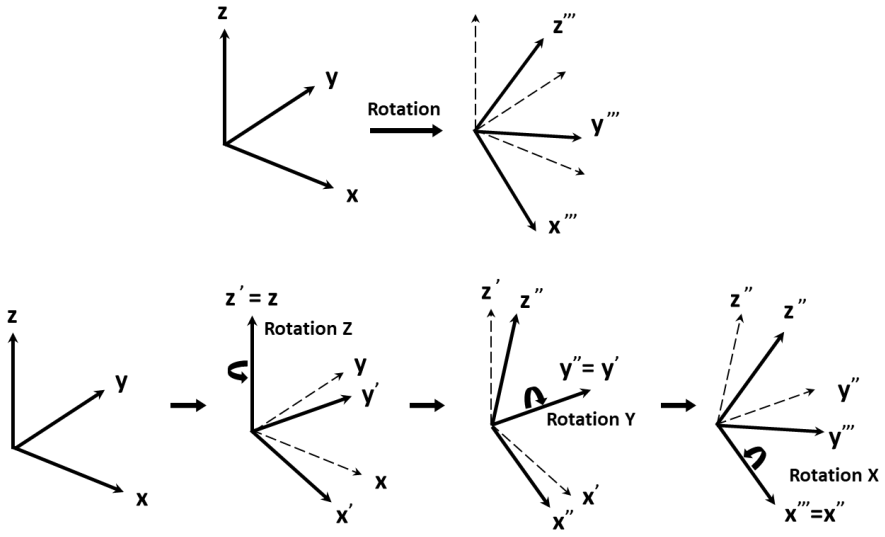


図 8.1.2.3

座標オフセットがない場合の各 WCS と MCS の関係を図 8.1.2.4 に示します。

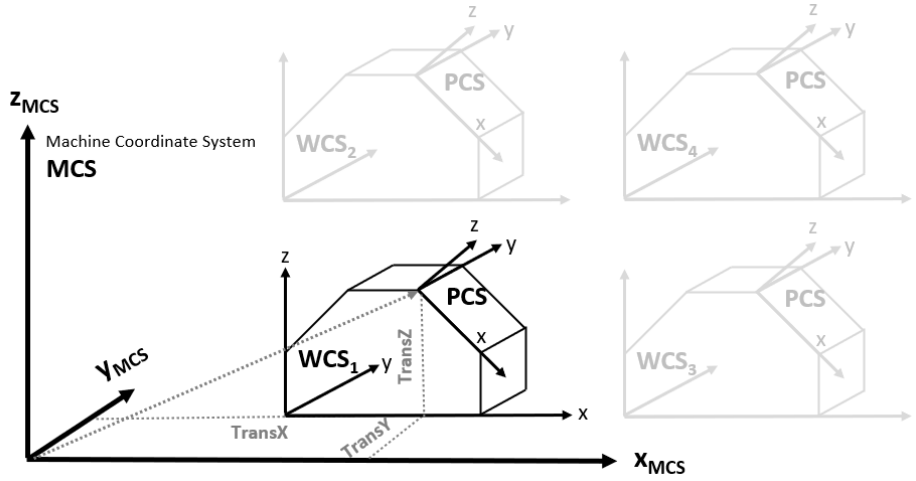


図 8.1.2.4

座標オフセットを加えると、WCS と MCS の関係は図 8.1.2.5 のようになります。座標オフセットの変換が追加されます。

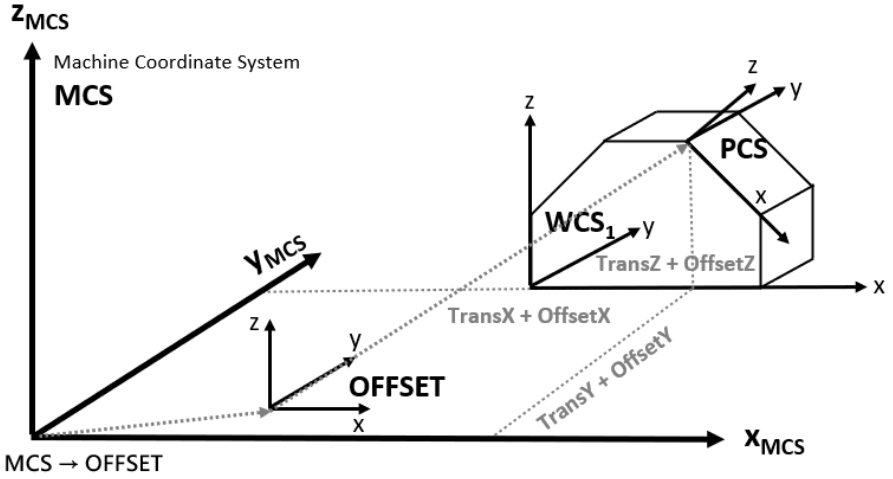


図 8.1.2.5

上記の機能に基づいて、ユーザーは HIMC で座標変換のパラメーターを定義し、座標系間の変換関係を確立できます。図 8.1.2.6 に座標系の関係を示します。わかりやすくするために、図では XY 平面の座標のみを示しています。実際のアプリケーションでは、ユーザーは座標変換に 6 つの自由度(X、Y、Z、A、B、C)を設定できます。

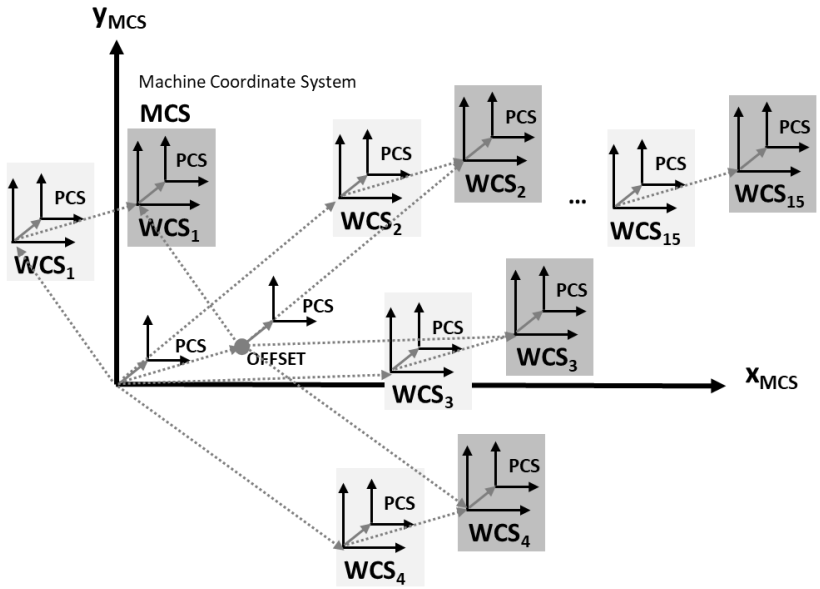


図 8.1.2.6

8.1.3 運動学

キネマティクスは、主に ACS(軸座標系)と MCS(機械座標系)の間の変換を扱います。フォワードキネマティクスは、ACS の各軸の位置フィードバックから MCS の座標位置までの計算です。逆に、逆運動学は、MCS の座標位置から ACS の各軸の位置までの計算です。表 8.1.3.1 に、HIMC が提供するキネマティクスコンフィギュレーションの定義を示します。

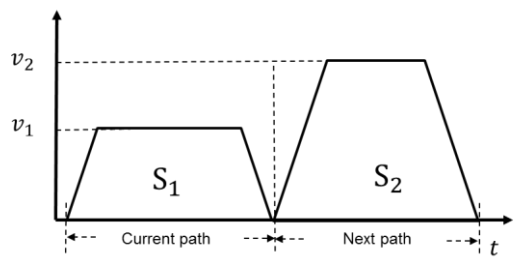
表 8.1.3.1

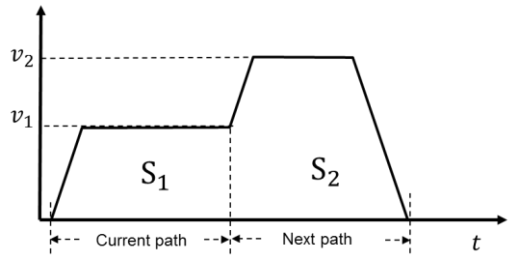
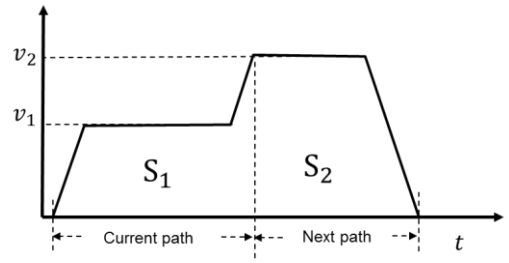
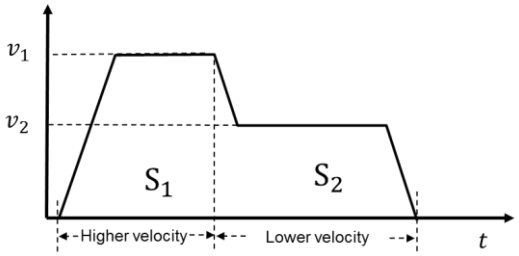
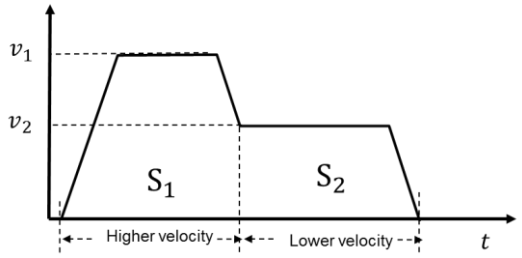
ID	名称	説明
1	直交座標系	連携モーショングループの各軸を直交座標の X、Y、Z、A、B、C 軸にそれぞれマッピングします。関節空間で許容される最大軸数は 6 です。(軸グループのデフォルト)
2	スカラ	(サポートしていません)
3	ウェハ	(サポートしていません)
4	6 軸多関節ロボット	(サポートしていません)

8.1.4 バッファーモード

バッファーモードは、隣接するパスの終点での速度プロファイルを決定します。ユーザーはこの設定を使用して、隣接する 2 つのパスのプロファイル速度を計画できます。表 8.1.4.1 に、HIMC が提供するバッファーモードの定義を示します。

表 8.1.4.1

HMPL 定義	説明
BM_ABORT	進行中のモーションを中止し、すぐに次のモーションを開始します。
BM_BUFF	現在のパスが完了したら、次のパスを開始します。 

BM_PREV	速度は、現在のパスの速度とブレンドされます。(ブレンド) 
BM_NEXT	速度は、次のパスの速度とブレンドされます。(ブレンド) 
BM_HIGH	速度は、2 つのパス間でより高い速度とブレンドされます。(ブレンド) 
BM_LOW	速度は、2 つのパスの間の低い速度とブレンドされます。(ブレンド) 

8.1.5 トランジションモード

トランジションモードは、隣接するパス間のトランジションカーブのタイプを決定します。この設定により、HIMC は、設定されたパラメーターに基づいて 2 つの線形モーションコマンド間のコーナースムージングの計算を行います(表 8.1.5.1 を参照)。より良い計画パスを実現するために、これは元のモーションプロファイルに影響します。

表 8.1.5.1

HMPL 定義	説明	対応パラメーター	単位
TM_NONE	なし：トランジション曲線を挿入しません（初期モード）	N/A	N/A
TM_START_VEL (サポートしていません)	開始速度	TPStartVelocity	mm/s または deg/s
TM_CONST_VEL	等速	TPVelocity	mm/s または deg/s
TM_CORNER_DIST	コーナー距離	TPCornerDistance	mm または deg
TM_MAX_CORNER_DEV (サポートしていません)	最大コーナー偏差	TPCornerDeviation	mm または deg

トランジションモード TM_CONST_VEL で高度なグループモーションコントロールの機能「LinAbs」および「LinRel」を使用して円運動の速度を割り当てる場合、円の開始位置からコーナーまでの距離は、セクション 8.3.25 で設定されたトランジションモードの距離パラメーターによって図.8.1.5.1 のように決定されます。トランジションモード TM_CORNER_DIST を使用して、円の開始位置からコーナーまでの距離を割り当てると、円運動の速度はセクション 8.3.25 で設定されたトランジションモードの速度パラメーターによって決定されます。

トランジションモードで計算されたスムージング円半径がセグメントの長さを超える場合、セグメント間のトランジションモード関数は無視されます。

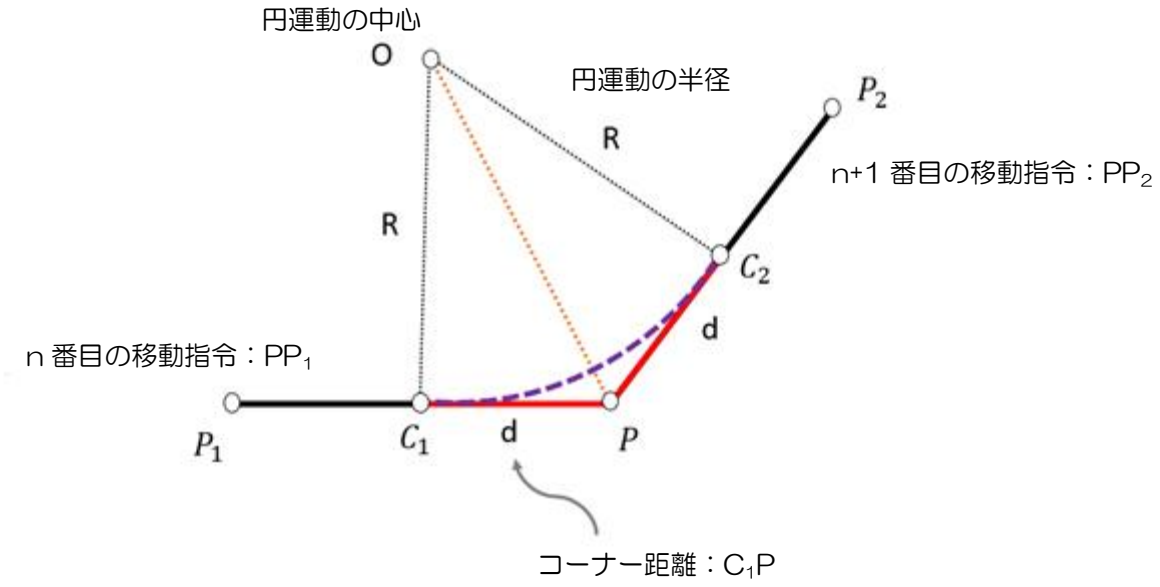

円運動の中心
O
円運動の半径
R
R
P_2
n+1 番目の移動指令：PP_2
C_2
d
n 番目の移動指令：PP_1
P_1
C_1
d
P
コーナー距離：C_1P

図 8.1.5.1

8.1.6 例

例 1: 基本的なグループ設定と直線運動

軸グループを作成して有効にし、連携モーションコマンドを実行する方法を、次の HMPL タスクに示します。

```
void main() {

    int gid = 0;    //    group index

    UngrpAllAxes(gid);
    //    Remove all axes from the existing axis group and disable it (optional)

    Enable(0);
    Enable(1);

    Till(IsEnabled(0) && IsEnabled(1));    //    Wait until all axes are enabled

    SetGrpMotionProfile(gid, 100, 5000, 5000, 200);
    //    Set a motion profile for LineAbs2D

    SetupGroup(gid, 0, 1);
    //    Add axis 0 and axis 1 to the axis group and enable it

    LineAbs2D(gid, 100, 100);    //    absolute linear movement
    Till(IsGrpInPos(gid));

    LineAbs2D(gid, 0.0, 0.0);    //    absolute linear movement
    Till(IsGrpInPos(gid));
}
```

この概念は、PLCopen® Motion Control：Part 4 - Coordinated Motion の概念と似ています。詳細については、PLCopen® 章 4.1「AxisGroup の作成と使用」を参照してください。

注: PLCopen® は、協会 PLCopen によってライセンスされた登録商標です。

例 2: 高度なグループ設定と速度ブレンディング

図 8.1.6.1 の 2 次元パスに沿って工具中心点を移動する方法を、次の HMPL タスクに示します。グループモーションプロファイルは、「BM_PREV」、「BM_NEXT」、および「BM_BUFF」で構成されます。p_1 での調整速度は、「BM_PREV」のパス 1 の最大速度を参照します。一方、p_2 で調整された速度は、「BM_NEXT」により、パス 3 の最大速度を参照します。

注: 「BM_PREV」と「BM_NEXT」の詳細については、セクション 8.1.4 を参照してください。

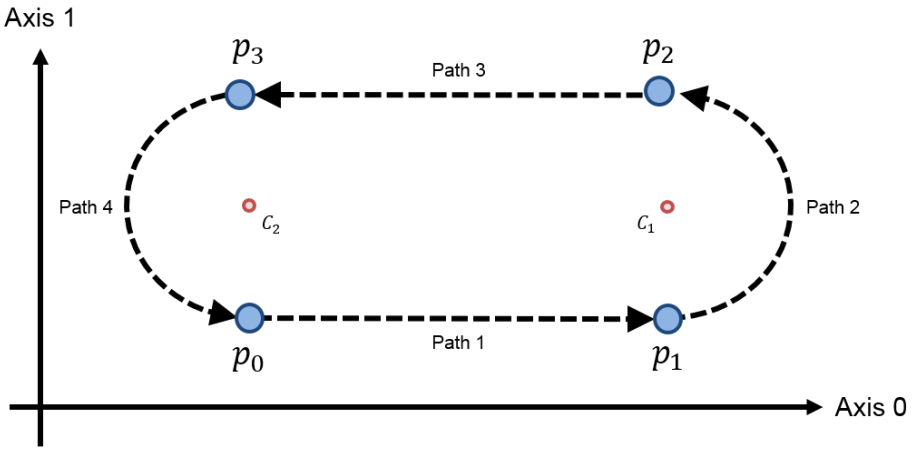


図 8.1.6.1

```
void main() {

    int axis[2] = {0, 1};    //    axis index
    int gid = 0;    //    group index

    UngrpAllAxes(gid);
    //    Remove all axes from the existing axis group and disable it (optional)

    AddAxisToGrp(gid, axis[0]);    //    Add axis to group 0
    AddAxisToGrp(gid, axis[1]);

    Enable(axis[0]);    //    Enable all axes in group 0
    Enable(axis[1]);

    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    //    Wait until all axes are enabled

    EnableGroup(gid);    //    Enable group 0
```

```
    double c1[3] = {100, 50, 0};
    double c2[3] = {0, 50, 0};
    double p0[6] = {0, 0, 0, 0, 0, 0};
    double p1[6] = {100, 0, 0, 0, 0, 0};
    double p2[6] = {100, 100, 0, 0, 0, 0};
    double p3[6] = {0, 100, 0, 0, 0, 0};

    double norm_ccw[3] = {0, 0, 1};
    double vel[4] = {100, 5000, 5000, 50};

    double trans_para[4] = {0, 0, 0, 0};

    LineAbs(gid, p0, vel, CS_MCS, BM_BUFF, TM_NONE, trans_para);
    Till(IsGrpInPos(gid));

    //  Blending Next and Blending Previous
    LineAbs(gid, p1, vel, CS_MCS, BM_PREV, TM_NONE, trans_para);
    //  path 1
    CircleAbs(gid, c1, norm_ccw, 0, p2, vel, CS_MCS, BM_NEXT, TM_NONE, trans_para);
    //  path 2
    LineAbs(gid, p3, vel, CS_MCS, BM_PREV, TM_NONE, trans_para);
    //  path 3
    Till(IsGrpInPos(gid));

    //  Buffered
    CircleAbs(gid, c2, norm_ccw, 0, p0, vel, CS_MCS, BM_BUFF, TM_NONE, trans_para);
    //  path 4
    Till(IsGrpInPos(gid));
}
```

例 3-1: Transition (行間)

図 8.1.6.2 の 2 次元パスに沿って工具中心点を移動する方法を、次の HMPL タスクに示します。グループモーションプロファイルは p_0 から始まります。p_1、p_2、p_3 を経て p_0 に戻ったら、Transition モードを「TM_CORNER_DIST」に設定し、速度と距離を設定します。2 番目のモーションが p_1、p_2、p_3、p_0 を通過すると、パスは図 8.1.6.2 の赤い実線に自動的に変更されます。

注：「TM_CORNER_DIST」の詳細については、セクション 8.1.5 を参照してください。

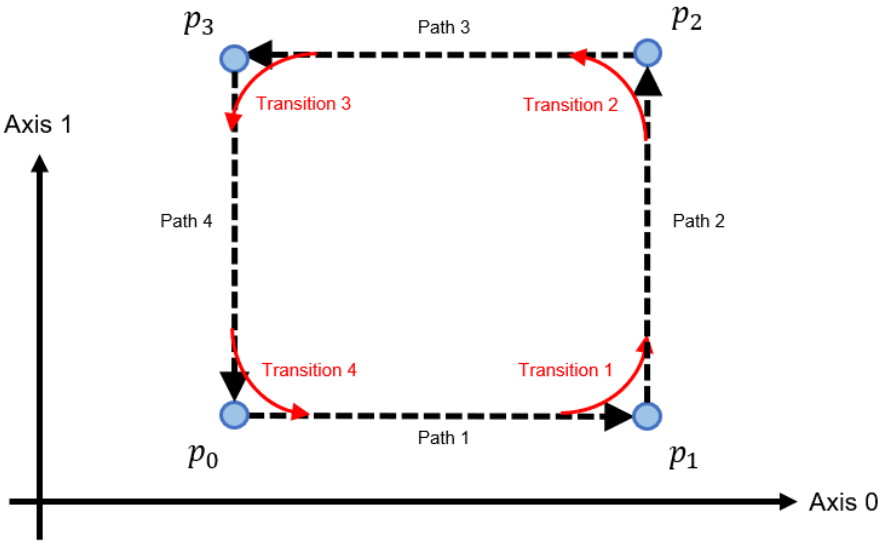


図 8.1.6.2

```
typedef struct {
    double x, y;
} Point2D;

//  Set space points
Point2D p0 = {.x = 0, .y = 0};
Point2D p1 = {.x = 100, .y = 0};
Point2D p2 = {.x = 100, .y = 100};
Point2D p3 = {.x = 0, .y = 100};

void RectangularMotion(int gid) {
    LineAbs2D(gid, p0.x, p0.y);
    LineAbs2D(gid, p1.x, p1.y);
    LineAbs2D(gid, p2.x, p2.y);
    LineAbs2D(gid, p3.x, p3.y);
    LineAbs2D(gid, p0.x, p0.y);
```

```
}

void main() {
    int axis[2] = {0, 1};     //  axis index
    int gid = 0;                //  group index
    double round_vel = 50;   //  transition velocity
    double round_dis = 20;   //  transition distance
    double round_dev = 1;     //  transition maximum deviation
    double round_curv = 5;   //  transition curvature

    //  Create and enable an axis group
    UngrpAllAxes(gid);
    Enable(axis[0]);          Enable(axis[1]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    SetupGroup(gid, axis[0], axis[1]);

    //  rectangular motion profile
    RectangularMotion(gid);
    Till(IsGrpInPos(gid));

    //  Enable transition function and set its velocity and distance
    SetGrpTransMode(gid, TM_CORNER_DIST);
    SetGrpTransPrm(gid, round_vel, round_dis, round_dev, round_curv);

    //  rectangular motion profile
    RectangularMotion(gid);

    //  Set end position
    LineAbs2D(gid, p0.x + round_dis, p0.y);
    Till(IsGrpInPos(gid));

    //  Disable transition function
    SetGrpTransMode(gid, TM_NONE);
}
```

例 3-2: 移行 (円形からラインへ、ラインから円形へ、円形から円形へ)

図 8.1.6.3 の 2 次元パスに沿って工具中心点を移動する方法を、次の HMPL タスクに示します。 グループモーションプロファイルは $p0$ から始まり、 $p1$ 、 $p2$ 、 $p3$ 、 $p4$ を経て、$p0$ に戻り、遷移モードを「TM_CORNER_DIST」に設定し、その速度と距離を設定します。 2 番目のモーションが $p1$ 、 $p2$ 、 $p3$ 、 $p4$ 、 $p0$ を通過すると、パスは図 8.1.6.3 の赤い実線に自動的に変更されます。

注: 「TM_CORNER_DIST」の詳細については、セクション 8.1.5 を参照してください。

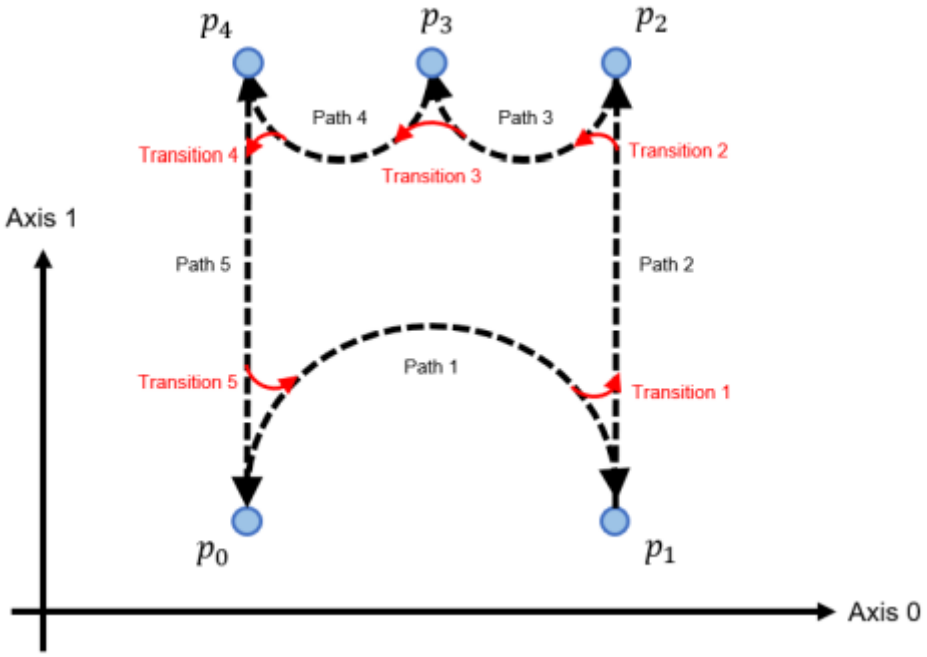


図 8.1.6.3

```
typedef struct {
double x, y;
} Point2D;

//   Set space points
Point2D pt0 = {.x = 0, .y = 0};
Point2D pt1 = {.x = 100, .y = 0};
Point2D pt2 = {.x = 100, .y = 100};
Point2D pt3 = {.x = 50, .y = 100};
Point2D pt4 = {.x = 0, .y = 100};

Point2D c1 = {.x = 50, .y = 0};
Point2D c2 = {.x = 75, .y = 100};
Point2D c3 = {.x = 25, .y = 100};
```

```
void Motion(int gid) {
    LineAbs2D(gid, pt0.x, pt0.y);
    Circle2D(gid, c1.x, c1.y, pt1.x, pt1.y, -1);
    LineAbs2D(gid, pt2.x, pt2.y);
    Circle2D(gid, c2.x, c2.y, pt3.x, pt3.y, -1);
    Circle2D(gid, c3.x, c3.y, pt4.x, pt4.y, -1);
    LineAbs2D(gid, pt0.x, pt0.y);
}

void main() {
    int axis[2] = {0, 1};       //  axis index
    int gid = 0;                //  group index
    double round_vel = 50;      //  transition velocity
    double round_dis = 20;      //  transition distance

    //  Create and enable an axis group
    UngrpAllAxes(gid);
    Enable(axis[0]); Enable(axis[1]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    SetupGroup(gid, 0, 1);

    //  Disable transition function
    SetGrpTransMode(gid, TM_NONE);

    //  Motion profile
    Motion(gid);
    Till(IsGrpInPos(gid))

    //  Enable transition function and set its velocity and distance
    SetGrpTransMode(gid, TM_CORNER_DIST);
    SetGrpTransPrm(gid, round_vel, round_dis, 0, 0);

    //  Motion profile
    Motion(gid);
    Circle2D(gid, c1.x, c1.y, 0.5*(pt0.x + pt1.x), 0.5*(pt0.x + pt1.x), -1);
    Till(IsGrpInPos(gid))
```

Group functions HIMC HMPL User Guide

```
    //  Disable transition function
    SetGrpTransMode(gid, TM_NONE);
}
```

例 4: 3 次元の円運動とらせん運動

```
void main() {

    int axis[3] = {0, 1, 2};    //    axis index
    int gid = 0;    //    group index

    double center1[3] = {0, 50, 0};                //    center of circle 1
    double center2[3] = {0, 0, 50};                //    center of circle 2
    double end_pos[6] = {0, 0, 0, 0, 0, 0};    //    end position
    double norm_x[3] = {1, 0, 0};                  //    normal vector x
    double norm_y[3] = {0, 1, 0};                  //    normal vector y
    double norm_z[3] = {0, 0, 1};                  //    normal vector z
    double vel[4] = {100, 5000, 5000, 50};

    //    Create and enable an axis group
    UngrpAllAxes(gid);
    Enable(axis[0]); Enable(axis[1]); Enable(axis[2]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]) && IsEnabled(axis[2]));
    SetupGroup(gid, axis[0], axis[1], axis[2]);

    //    Move to coordinate (0, 0, 0)
    LineAbs3D(gid, 0, 0, 0);

    //    three-dimensional circular motion
    CircAbs(gid, center1, norm_z, 1, end_pos, vel, CS_MCS, BM_BUFF, TM_NONE, 0);
    CircAbs(gid, center2, norm_y, 1, end_pos, vel, CS_MCS, BM_BUFF, TM_NONE, 0);
    CircAbs(gid, center2, norm_x, 1, end_pos, vel, CS_MCS, BM_BUFF, TM_NONE, 0);
    end_pos[2] = 100;

    //    spiral motion
    CircAbs(gid, center1, norm_z, 5, end_pos, vel, CS_MCS, BM_BUFF, TM_NONE, 0);

    //    Move to coordinate (0, 0, 0)
    LineAbs3D(gid, 0, 0, 0);
    Till(IsGrpInPos(gid));
}
```

例 5-1：座標変換（MCS→PCS）

マシン座標系(MCS)で矢印モーションプロファイルを生成します。元のモーションプロファイルの値を変更する代わりに、参照座標を製品座標系(PCS)とその変換パラメーターとして設定します。そうすることで、図 8.1.6.3 に示すように、MCS を PCS に変換できます。

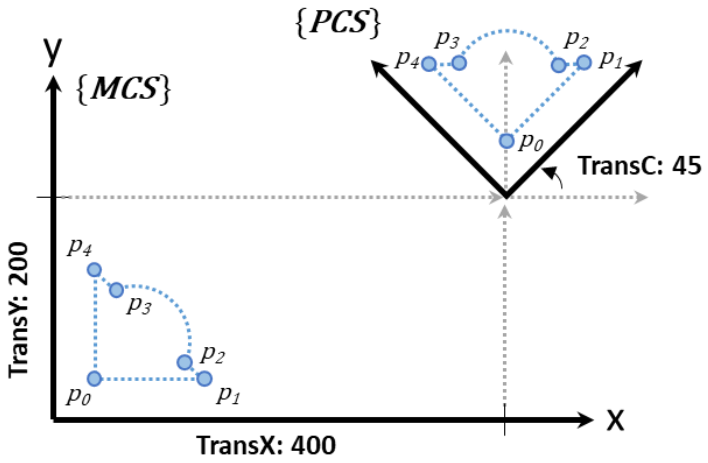


図 8.1.6.3

```
typedef struct {
    double x, y;
} Point2D;

//  Set space points
Point2D p0 = {.x = 10, .y = 10};
Point2D p1 = {.x = 110, .y = 10};
Point2D p2 = {.x = 100, .y = 40};
Point2D p3 = {.x = 40, .y = 100};
Point2D p4 = {.x = 10, .y = 110};
Point2D center = {.x = 60, .y = 60};

void ArrowMotion(int gid) {
    LineAbs2D(gid, p0.x, p0.y);
    LineAbs2D(gid, p1.x, p1.y);
    LineAbs2D(gid, p2.x, p2.y);
    Circle2D(gid, center.x, center.y, p3.x, p3.y, 0);
    LineAbs2D(gid, p4.x, p4.y);
    LineAbs2D(gid, p0.x, p0.y);
}
```

```
void main() {
    int axis[2] = {0, 1};    //  axis index
    int gid = 0;             //  group index

    //  Create and enable an axis group
    UngrpAllAxes(gid);
    Enable(axis[0]);         Enable(axis[1]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    SetupGroup(gid, axis[0], axis[1]);    //  axis 0 is X axis, axis 1 is Y axis

    //  arrow motion profile
    ArrowMotion(gid);
    Till(IsGrpInPos(gid));

    //  Set Product coordinate transformation parameters:
    //  X translates 400 mm, Y translates 200 mm, Z rotates 45 deg
    double transfer[6] = {400, 200, 0, 0, 0, 45};
    SetGrpCoordTrans(gid, CS_PCS, transfer);

    //  Refer to Product Coordinate System
    SetGrpCoordSys(gid, CS_PCS);

    //  arrow motion profile
    ArrowMotion(gid);

    //  Refer to Machine Coordinate System
    SetGrpCoordSys(gid, CS_MCS);

    //  Set end position
    LineAbs2D(gid, 0, 0);
    Till(IsGrpInPos(gid));

    //  Clear coordinate transformation parameters
    double zeros[6] = {0, 0, 0, 0, 0, 0};
    SetGrpCoordTrans(gid, CS_PCS, zeros);
}
```

例 5-2: 座標変換 (MCS→WCSn→PCS)

概念は例 5-1 と同じです。 機械座標系 (MCS) 上で矢印のモーション プロファイルを生成します。 元のモーション プロファイルの値を変更する代わりに、ワークピース座標系 (WCS) を参照座標に追加します。 これにより、図 8.1.6.5 に示すように、元の製品座標系 (PCS) の動作プロファイルを各ワーク座標に転送できます。

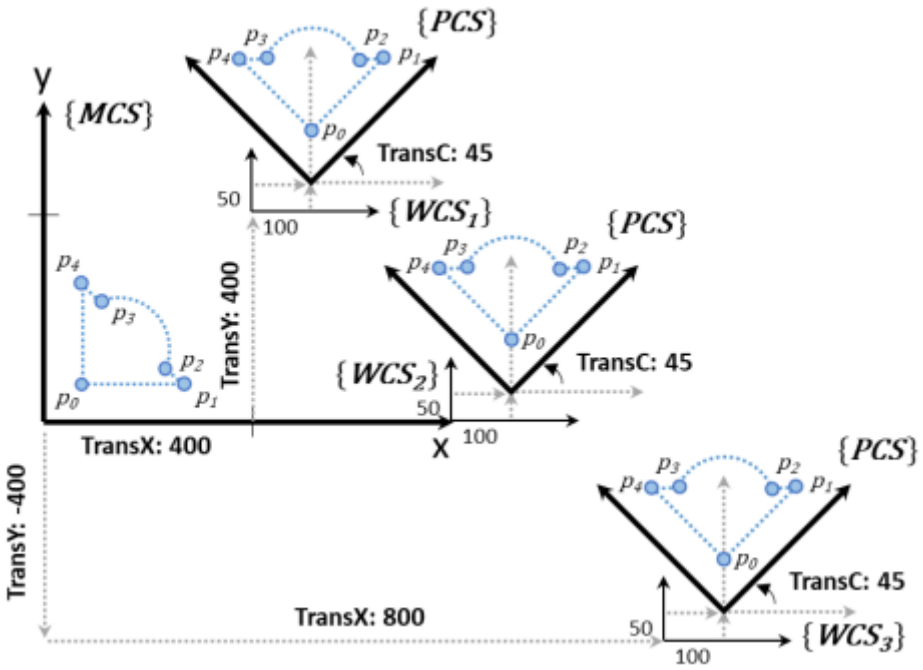


図 8.1.6.5

```
typedef struct {
    double x, y;
} Point2D;

//  Set space points
Point2D p0 = {.x = 10, .y = 10};
Point2D p1 = {.x = 110, .y = 10};
Point2D p2 = {.x = 100, .y = 40};
Point2D p3 = {.x = 40, .y = 100};
Point2D p4 = {.x = 10, .y = 110};
Point2D center = {.x = 60, .y = 60};

void ArrowMotion(int gid) {
    LineAbs2D(gid, p0.x, p0.y);
    LineAbs2D(gid, p1.x, p1.y);
```

```
LineAbs2D(gid, p2.x, p2.y);
    Circle2D(gid, center.x, center.y, p3.x, p3.y, 0);
    LineAbs2D(gid, p4.x, p4.y);
    LineAbs2D(gid, p0.x, p0.y);
}

void main() {
    int axis[2] = {0, 1};    //  axis index
    int gid = 0;             //  group index

    //  Create and enable an axis group
    UngrpAllAxes(gid);
    Enable(axis[0]);         Enable(axis[1]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    SetupGroup(gid, axis[0], axis[1]);    //  axis 0 is X axis, axis 1 is Y axis

    //  arrow motion profile
    ArrowMotion(gid);
    Till(IsGrpInPos(gid));

    //  Set Product coordinate transformation parameters:
    //  X translates 100 mm, Y translates 50 mm, Z rotates 45 deg
    double transfer[6] = {100, 50, 0, 0, 0, 45};

    //  Set transformation parameters of workpiece coordinate 1~3
    double wcs1[6] = {400, 400, 0, 0, 0, 0};
    double wcs2[6] = {600, 0, 0, 0, 0, 0};
    double wcs3[6] = {800, -400, 0, 0, 0, 0};

    SetGrpCoordTrans(gid, CS_PCS, transfer);
    SetGrpCoordTrans(gid, CS_WCS1, wcs1);
    SetGrpCoordTrans(gid, CS_WCS2, wcs2);
    SetGrpCoordTrans(gid, CS_WCS3, wcs3);

    //  Refer to Workpiece Coordinate System 1 & Product Coordinate System
    SetGrpCoordSys(gid, CS_WCS1 | CS_PCS);
    ArrowMotion(gid);
```

```
//   Refer to Workpiece Coordinate System 2 & Product Coordinate System
      SetGrpCoordSys(gid, CS_WCS2 | CS_PCS);
      ArrowMotion(gid);

      //   Refer to Workpiece Coordinate System 3 & Product Coordinate System
      SetGrpCoordSys(gid, CS_WCS3 | CS_PCS);
      ArrowMotion(gid);

      //   Refer to Machine Coordinate System
      SetGrpCoordSys(gid, CS_MCS);

      //   Set end position
      LineAbs2D(gid, 0, 0);
      Till(IsGrpInPos(gid));

      //   Clear coordinate transformation parameters
      double zeros[6] = {0, 0, 0, 0, 0, 0};
      SetGrpCoordTrans(gid, CS_PCS, zeros);
      SetGrpCoordTrans(gid, CS_WCS1, zeros);
      SetGrpCoordTrans(gid, CS_WCS2, zeros);
      SetGrpCoordTrans(gid, CS_WCS3, zeros);
}
```

例 5-3: 座標変換 (MCS→OFFSET→WCSn→PCS)

この例は例 5-2 の拡張です。 HIMC が提供する座標オフセット (OFFSET) を使用すると、図 8.1.6.6 に示すように、例 5-2 の結果をオフセットできます。

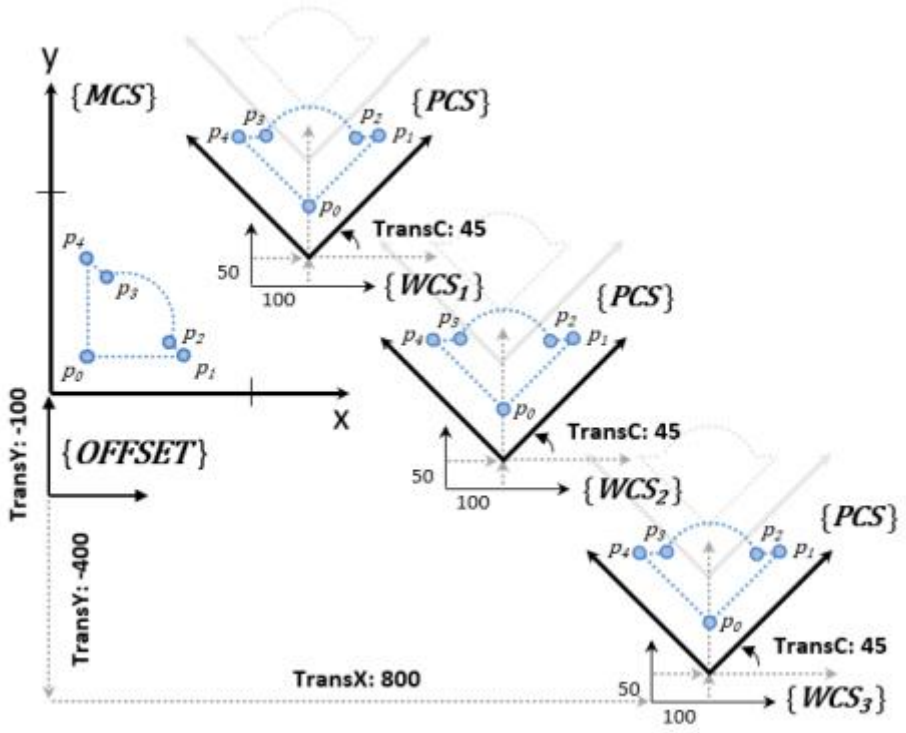


図 8.1.6.6

```
typedef struct {
    double x, y;
} Point2D;

//  Set space points
Point2D p0 = {.x = 10, .y = 10};
Point2D p1 = {.x = 110, .y = 10};
Point2D p2 = {.x = 100, .y = 40};
Point2D p3 = {.x = 40, .y = 100};
Point2D p4 = {.x = 10, .y = 110};
Point2D center = {.x = 60, .y = 60};

void ArrowMotion(int gid) {
    LineAbs2D(gid, p0.x, p0.y);
    LineAbs2D(gid, p1.x, p1.y);
```

```
    LineAbs2D(gid, p2.x, p2.y);
    Circle2D(gid, center.x, center.y, p3.x, p3.y, 0);
    LineAbs2D(gid, p4.x, p4.y);
    LineAbs2D(gid, p0.x, p0.y);
}

void main() {
    int axis[2] = {0, 1};    //  axis index
    int gid = 0;             //  group index

    //  Create and enable an axis group
    UngrpAllAxes(gid);
    Enable(axis[0]);         Enable(axis[1]);
    Till(IsEnabled(axis[0]) && IsEnabled(axis[1]));
    SetupGroup(gid, axis[0], axis[1]);    //  axis 0 is X axis, axis 1 is Y axis

    //  arrow motion profile
    ArrowMotion(gid);
    Till(IsGrpInPos(gid));

    //  Set Product coordinate transformation parameters:
    //  X translates 100 mm, Y translates 50 mm, Z rotates 45 deg
    double transfer[6] = {100, 50, 0, 0, 0, 45};

    //  Set transformation parameters of workpiece coordinate 1~3
    double wcs1[6] = {400, 400, 0, 0, 0, 0};
    double wcs2[6] = {600, 0, 0, 0, 0, 0};
    double wcs3[6] = {800, -400, 0, 0, 0, 0};

    //  Set transformation parameters of Coordinate Offset
    double offset[6] = {0, -100, 0, 0, 0, 0};

    SetGrpCoordTrans(gid, CS_PCS, transfer);
    SetGrpCoordTrans(gid, CS_WCS1, wcs1);
    SetGrpCoordTrans(gid, CS_WCS2, wcs2);
    SetGrpCoordTrans(gid, CS_WCS3, wcs3);
    SetGrpCoordTrans(gid, CS_OFFSET, offset);
    //  Refer to Coordinate Offest & Workpiece Coordinate System 1
```

```
        //  & Product Coordinate System
        SetGrpCoordSys(gid, CS_OFFSET | CS_WCS1 | CS_PCS);
        ArrowMotion(gid);

        //  Refer to Coordinate Offest & Workpiece Coordinate System 2
        //  & Product Coordinate System
        SetGrpCoordSys(gid, CS_OFFSET | CS_WCS2 | CS_PCS);
        ArrowMotion(gid);

        //  Refer to Coordinate Offest & Workpiece Coordinate System 3
        //  & Product Coordinate System
        SetGrpCoordSys(gid, CS_OFFSET | CS_WCS3 | CS_PCS);
        ArrowMotion(gid);

        //  Refer to Machine Coordinate System
        SetGrpCoordSys(gid, CS_MCS);

        //  Set end position
        LineAbs2D(gid, 0, 0);
        Till(IsGrpInPos(gid));

        //  Clear coordinate transformation parameters
        double zeros[6] = {0, 0, 0, 0, 0, 0};
        SetGrpCoordTrans(gid, CS_PCS, zeros);
        SetGrpCoordTrans(gid, CS_WCS1, zeros);
        SetGrpCoordTrans(gid, CS_WCS2, zeros);
        SetGrpCoordTrans(gid, CS_WCS3, zeros);
        SetGrpCoordTrans(gid, CS_OFFSET, zeros);
    }
```

8.2 グループモーションコントロール

8.2.1 EnableGroup

目的
軸グループを有効にします。

構文

```
int EnableGroup(
        int group_id
);
```

パラメーター
group_id [in]　　Axis group index

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考
この関数を実行する前に、グループ内のすべての軸を有効にする必要があります。

条件

サポートされる最小バージョン	iA Studio 0.23

8.2.2 DisableGroup

目的

軸グループを無効にします

構文

```
int DisableGroup(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

8.2.3 ResetGroup

目的

軸グループの状態を「グループエラー停止」から「グループスタンバイ」に変更します。

構文

```
int ResetGroup(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

この機能は、すべてのエラーがクリアされた後にのみ使用できます。

条件

サポートされる最小バージョン	iA Studio 1.3

8.2.4 StopGroup

目的
軸グループの動作を停止します

構文

```
int StopGroup(
        int group_id
);
```

パラメーター
group_id [in]　　Axis group index

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考
軸グループのモーションキューがクリアされます。

条件

サポートされる最小バージョン	iA Studio 0.23

8.2.5 HaltGroup

目的

軸グループの動作を停止するには、その速度は 0 に設定されます。

構文

```
int HaltGroup(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

軸グループが所定の位置にない場合、移動を続けます。

条件

サポートされる最小バージョン	iA Studio 1.3

8.2.6 ResumeGroup

目的

「停止」状態から軸グループの動作を再開します。

構文

```
int ResumeGroup(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.2.7 JogGroup

目的

軸群を機械座標系の指定方向に特定の速度で無限運動を開始させる。

構文

```
int JogGroup(
        int     group_id,
        int     carte_dir,
        double jog_vel
);
```

パラメーター

group_id [in]	Axis group index
carte_dir [in]	機械座標系での運動方向 0 ～ 5 の数字は機械座標系の 6-DOF {X, Y, Z, A, B, C}を順番に表します。
jog_vel [in]	特定の速度の値 その正/負の値は、運動方向の同方向/逆方向の動きを表します。 パラメーター単位：mm/s または deg/s

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.4

8.2.8 JogGroupAxis

目的

軸グループ内の特定の軸に、軸座標系で特定の速度で無限運動を開始させます。

構文

```
int JogGroupAxis(
        int     group_id,
        int     grp_axis,
        double jog_vel
);
```

パラメーター

group_id [in]	Axis group index
grp_axis [in]	軸グループ内の Axis index 番号 0 ～ 8 は、各軸が軸グループに追加される順番を表します。
jog_vel [in]	特定の速度の値 その正/負の値は、運動方向の同方向/逆方向の動きを表します。 パラメーター単位：mm/s または deg/s

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.4

8.2.9 LineAbs2D

目的

機械座標系の絶対目標位置に向けて、軸グループ上で補間された 2 次元の直線移動を指令します。

構文

```
int LineAbs2D(
        int      group_id,
        double end_x,
        double end_y
);
```

パラメーター

group_id [in]	Axis group index
end_x [in]	X 座標での絶対ターゲット位置の値
end_y [in]	Y 座標での絶対ターゲット位置の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
void main() {
        //   Assume that axis group 0 which includes two orthogonal axes is enabled
        double target_x = 100;
        double target_y = 100;
        LineAbs2D (0 /* group_id */, target_x, target_y);
        //   Move to (target_x, target_y)
        //   that is (100, 100)
}
```

条件

サポートされる最小バージョン	iA Studio 0.24

8.2.10 LineAbs3D

目的

機械座標系の絶対目標位置に向かって、軸グループ上で補間された 3 次元の直線移動を命令すること。

構文

```
int LineAbs3D(
    int    group_id,
    double end_x,
    double end_y,
    double end_z
);
```

パラメーター

group_id [in]	Axis group index
end_x [in]	X 座標での絶対ターゲット位置の値
end_y [in]	Y 座標での絶対ターゲット位置の値
end_z [in]	Z 座標での絶対ターゲット位置の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.24

8.2.11 LineRel2D

目的

機械座標系の相対位置に向かって、軸グループ上で補間された 2 次元の直線移動を指令します。

構文

```
int LineRel2D(
        int      group_id,
        double distance_x,
        double distance_y
);
```

パラメーター

group_id [in]	Axis group index
distance_x [in]	X 座標での相対距離の値
distance_y [in]	Y 座標での相対距離の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
void main() {
        //   Assume that axis group 0 which includes two orthogonal axes is enabled
        //   and the starting positions of the two axes are (100, 200)
        double distance_x = 100;
        double distance_y = 100;
        LineRel2D (0 /* group_id */, distance_x, distance_y);
        //   Move to (X starting position + distance_x, Y starting position + distance_y)
        //   that is (200, 300)
}
```

条件

サポートされる最小バージョン	iA Studio 0.24

8.2.12 LineRel3D

目的

機械座標系の相対位置に向かって、軸グループ上で補間された 3 次元の直線移動を指令します。

構文

```
int LineRel3D(
        int      group_id,
        double distance_x,
        double distance_y,
        double distance_z
);
```

パラメーター

group_id [in]	Axis group index
distance_x [in]	X 座標での相対距離の値
distance_y [in]	Y 座標での相対距離の値
distance_z [in]	Z 座標での相対距離の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.24

8.2.13 Arc2D

目的

機械座標系の絶対目標位置に向かって、軸グループ上で補間された 2 次元円運動を指令します。

構文

```
int Arc2D(
        int      group_id,
        double border_x,
        double border_y,
        double end_x,
        double end_y
);
```

パラメーター

group_id [in]	Axis group index
border_x [in]	X 座標での境界の絶対位置の値
border_y [in]	Y 座標での境界の絶対位置の値
end_x [in]	X 座標での絶対終了位置の値
end_y [in]	Y 座標での絶対終了位置の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
void main() {
    //  Assume that axis group 0 which includes two orthogonal axes is enabled
    //  and the starting positions of the two axes are (0, 0)
    double border_x = 100;
    double border_y = 100;
    double end_x = 200;
    double end_y = 0;
    Arc2D(0 /* group_id */, border_x, border_y, end_x, end_y);
    //  Circle to (end_x, end_y) through (border_x , border_y)
    //  that is, circle to (200, 0) through (100, 100)
}
```

条件

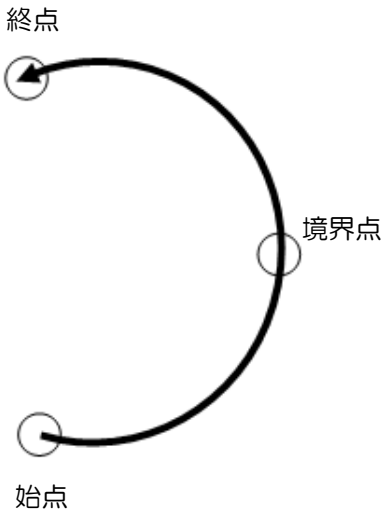
サポートされる最小バージョン	iA Studio 0.24

長所

ユーザーは境界点 (動きの中で最も遠い点) を指定し、マシンがそこに到達できることを確認できます。

短所

1 つのコマンドで角度 < 2π に制限されます。



8.2.14 ArcCW2D

目的

機械座標系の時計回りの絶対目標位置に向かって、軸グループ上で補間された 2 次元円運動を指令します。

構文

```
int ArcCW2D(
    int    group_id,
    double center_x,
    double center_y,
    double end_x,
    double end_y,
);
```

パラメーター

group_id [in]	Axis group index
center_x [in]	X 座標の絶対中心位置の値
center_y [in]	Y 座標の絶対中心位置の値
end_x [in]	X 座標での絶対終了位置の値
end_y [in]	Y 座標での絶対終了位置の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.2.15 ArcCCW2D

目的

機械座標系の反時計回りの絶対目標位置に向かって、軸グループ上で補間された 2 次元円運動を指令します。

構文

```
int ArcCCW2D(
    int    group_id,
    double center_x,
    double center_y,
    double end_x,
    double end_y,
);
```

パラメーター

group_id [in]	Axis group index
center_x [in]	X 座標の絶対中心位置の値
center_y [in]	Y 座標の絶対中心位置の値
end_x [in]	X 座標での絶対終了位置の値
end_y [in]	Y 座標での絶対終了位置の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.2.16 ArcAngle2D

目的

与えられた角度に基づいて、機械座標系の絶対目標位置に向かう軸グループ上の補間された 2 次元円運動を指令します。

構文

```
int ArcAngle2D(
        int      group_id,
        double center_x,
        double center_y,
        double angle
);
```

パラメーター

group_id [in]	Axis group index
center_x [in]	X 座標の絶対中心位置の値
center_y [in]	Y 座標の絶対中心位置の値
angle [in]	絶対中心位置に対する始点と終点の角度。 円運動の方向と回転角度を決定します。 パラメーター単位：度

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

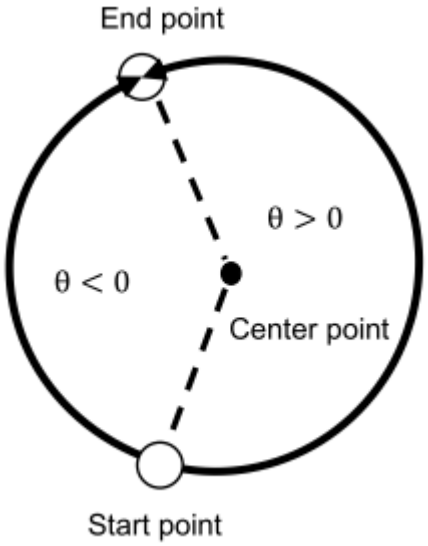
パラメーター「角度」は円軌道の回転方向を表します。
「角度」 > 0 の場合、反時計回りに移動します。 「角度」< 0 の場合、時計回りに移動します。

例

```
void main() {
   //   Assume that axis group 0 which includes two orthogonal axes is enabled
   //   and the starting positions of the two axes are (0, 0)
   double center_x = 10;
   double center_y = 10;
   double angle = 45;
   ArcAngle2D(0 /* group_id */, center_x, center_y, angle);
   //   Take (center_x, center_y) as the center and rotate 45 degrees
   //   that is, take (10, 10) as the center,
   //   and circle to (10, 10-10 √2 ) from (0, 0)
}
```

要件

サポートされる最小バージョン	iA Studio 2.0



θ の値は円運動の方向を決定します。
θ > 0 : 反時計回りに移動 θ < 0 : 時計回りに移動

8.2.17 Circle2D

目的

機械座標系の絶対目標位置に向かって、軸グループ上で補間された 2 次元円運動を指令します。

構文

```
int Circle2D(
    int    group_id,
    double center_x,
    double center_y,
    double end_x,
    double end_y,
    int    turns
);
```

パラメーター

group_id [in]	Axis group index
center_x [in]	X 座標の絶対中心位置の値
center_y [in]	Y 座標の絶対中心位置の値
end_x [in]	X 座標での絶対終了位置の値
end_y [in]	Y 座標での絶対終了位置の値
turns [in]	始点を基準とした円形パスのターン数。 円形パスの方向と合計角度を決定します。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

(1) パラメーター turns は、円軌道の回転方向を表します。
「turns」>= 0 の場合、反時計回りに移動します。「turns」< 0 の場合、時計回りに移動します。

(2) || turns||時 <= 1、円形軌道の合計移動角度は < 360°です。
円軌道の移動角度の合計が >= 360°(つまり、1 回転または複数回転) の場合、|| turns|| >= 2 でなければなりません。

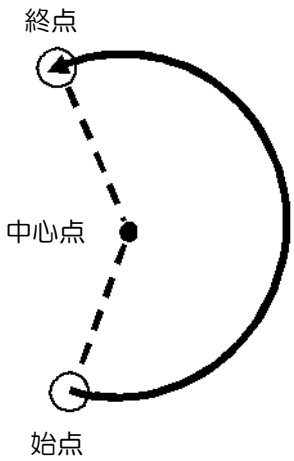
(3) この関数を使用する場合、「turns = 0」と「turns = 1」の動作は同じです。

例

```
void main() {
    //  Assume that axis group 0 which includes two orthogonal axes is enabled
    //  and the starting positions of the two axes are (0, 0)
    double center_x = 100;
    double center_y = 0;
    double end_x = 200;
    double end_y = 0;
    int turns = 1;
    Circle2D(0 /* group_id */, center_x, center_y, end_x, end_y, turns);
    //  Take (center_x, center_y) as the center and circle to (end_x, end_y)
    //  that is, take (100, 0) as the center and circle to (200, 0)
}
```

条件

サポートされる最小バージョン	iA Studio 1.1



長所
角度の制限なし

短所
ユーザーは境界点（動きの最も遠い点）を指定できません。そのため、境界点に到達しない場合があります。

ターンの値は、円運動の方向を決定します。
※ angle=θ+turns×360turns≧0 は C.C.W.方向、turns<0 は C.W.方向を示します。turns =0 の動きは、turns =1 の動きと同じであることに注意してください。次の表では、例として θ=210°を使用しています。

Turns	計算	角度（°）
-2	$210-2\times360°$	$-510°$
-1	$210-1\times360°$	$-150°$
0	$210+0\times360°$	$210°$
1	$210+0\times360°$	$210°$
2	$210+1\times360°$	$570°$

8.3 グループ設定

8.3.1 AddAxisToGrp

目的
軸グループに軸を追加します

構文

```
int AddAxisToGrp(
        int group_id,
        int axis_id
);
```

パラメーター

group_id [in]	軸グループインデックス
axis_id [in]	Axis index

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考
(1) 軸数は最大 9 軸です
(2) 追加する順序は、{X、Y、Z、A、B、C}に対応する必要があります。

条件

サポートされる最小バージョン	iA Studio 0.23

8.3.2 RemoveAxisFromGrp

目的

軸グループから最後の軸を削除します

構文

```
int RemoveAxisFromGrp(
        int group_id
);
```

パラメーター

group_id [in]　　軸グループインデックス

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.24

8.3.3 SetupGroup

目的

特定の順序で軸グループを設定します

構文

```
int SetupGroup(
        int group_id,
        int axis_id,
        int axis_id,
        …
        int axis_id
);
```

パラメーター

group_id [in]　　軸グループインデックス

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

軸の最大数は 9 です

条件

サポートされる最小バージョン	iA Studio 0.25

8.3.4 UngrpAllAxes

目的

軸グループをグループ解除して無効にします。

構文

```
int UngrpAllAxes(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

8.3.5 GetGroupID

目的

軸が属する軸グループ ID を取得します

構文

```
int GetGroupID(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が属する軸グループ ID
軸がどの軸グループにも属していない場合は、-1 を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

8.3.6 SetGrpMotionProfile

目的

軸グループの TCP 線形モーションパラメーターを設定します。

構文

```
int SetGrpMotionProfile(
        int     group_id,
        double max_velocity,
        double max_acceleration,
        double max_deceleration,
        double smooth_time
);
```

パラメーター

group_id [in]	Axis group index
max_velocity [in]	軸グループの最大線形プロファイル速度 パラメーター単位: mm/s 入力範囲: 0 ~ 5000
max_acceleration [in]	軸グループの線形プロファイルの最大加速度 パラメーター単位: mm/s^2 入力範囲: >0 ～ 50000 (加速度を 0 にすることはできません)
max_deceleration [in]	軸グループの最大線形プロファイル減速度 パラメーター単位: mm/s^2 入力範囲: >0 ～ 50000 (減速度を 0 にすることはできません)
smooth_time [in]	軸グループの線形プロファイルの平滑化時間 パラメーター単位: ms 入力範囲: 0 ～ 500

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

線形グループモーションプロファイルの既定値は、速度、加速、減速、およびスムーズ時間に対してそれぞれ [100、500、500、50] です。

条件

サポートされる最小バージョン	iA Studio 0.24

8.3.7 SetGrpAngMotionProfile

目的

軸グループの TCP 角運動パラメーターを設定します。

構文

```
int SetGrpAngMotionProfile(
    int    group_id,
    double max_velocity,
    double max_acceleration,
    double max_deceleration,
    double smooth_time
);
```

パラメーター

group_id [in]	Axis group index
max_velocity [in]	軸グループの最大角プロファイル速度 パラメーター単位: deg/s 入力範囲: 0 ～ 7200
max_acceleration [in]	軸グループの最大角プロファイル加速度 パラメーター単位: deg/s^2 入力範囲: >0 ～ 72000 (加速度を 0 にすることはできません)
max_deceleration [in]	軸グループの最大角度プロファイル減速 パラメーター単位: deg/s^2 入力範囲: >0 ～ 72000 (減速度を 0 にすることはできません)
smooth_time [in]	軸グループの角度プロファイルの平滑化時間 パラメーター単位: ms 入力範囲: 0 ～ 500

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

角度グループモーションプロファイルのデフォルト値は、速度、加速、減速、およびスムーズ時間に対してそれぞれ [360, 1800, 1800, 50]です。

条件

サポートされる最小バージョン	iA Studio 2.0

8.3.8 GetGrpKin

目的

軸グループのキネマティクスタイプを取得します。

構文

```
int GetGrpKin(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

軸グループのキネマティクスタイプ。詳細については、セクション 8.1.3 を参照してください。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.9 SetGrpKin

目的

軸グループのキネマティクスタイプを設定します。

構文

```
int SetGrpKin(
        int group_id,
        int kin_type
);
```

パラメーター

group_id [in]	Axis group index
kin_type [in]	軸グループの新しいキネマティクスタイプ。詳細については、セクション 8.1.3 を参照してください。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

8.3.10 GetGrpMaxVel

目的

軸グループの最大プロファイル速度を取得します。

構文

```
double GetGrpMaxVel(
    int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

軸グループの最大プロファイル速度
単位: mm/s または deg/s

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.11 SetGrpVel

目的

軸グループの最大プロファイル速度を設定します。

構文

```
int SetGrpVel(
    int    group_id,
    double vel
);
```

パラメーター

group_id [in]	Axis group index
vel [in]	軸グループの新しい最大プロファイル速度 パラメーター単位: mm/s または deg/s 入力範囲: 0 ～ 5000

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.12 GetGrpMaxAcc

目的

軸グループの最大プロファイル加速度を取得します。

構文

```
double GetGrpMaxAcc(
    int group_id
);
```

パラメーター

group_id [in] Axis group index

戻りの値

軸グループの最大プロファイル加速度

単位: mm/s^2 または deg/s^2

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.13 SetGrpAcc

目的

軸グループの最大プロファイル加速度を設定します。

構文

```
int SetGrpAcc(
    int    group_id,
    double acc
);
```

パラメーター

group_id [in]	Axis group index
acc [in]	軸グループの新しい最大プロファイル加速度 パラメーター単位 unit: mm/s² または deg/s² 入力範囲: >0 ～ 50000 (加速度を 0 にすることはできません)

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.14 SetGrpAccTime

目的

軸グループの加速時間を設定します

構文

```
int SetGrpAccTime(
    int    group_id,
    double acc_time
);
```

パラメーター

group_id [in]	Axis group index
acc_time [in]	軸グループの加速時間 パラメーター単位: ms 入力範囲: ゼロ以外の正の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.15 GetGrpMaxDec

目的

軸グループの最大プロファイル減速度を取得します。

構文

```
double GetGrpMaxDec(
    int group_id
);
```

パラメーター

group_id [in] Axis group index

戻りの値

軸グループの最大プロファイル減速度

単位: mm/s^2 または deg/s^2

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.16 SetGrpDec

目的

軸グループの最大プロファイル減速度を設定します。

構文

```
int SetGrpDec(
    int    group_id,
    double dec
);
```

パラメーター

group_id [in]	Axis group index
dec [in]	軸グループの新しい最大プロファイル減速度 パラメーター単位: mm/s² または deg/s² 入力範囲: >0 ～ 50000 (減速度を 0 にすることはできません)

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.17 SetGrpDecTime

目的

軸グループの減速時間を設定します

構文

```
int SetGrpDecTime(
    int    group_id,
    double dec_time
);
```

パラメーター

group_id [in]	Axis group index
dec_time [in]	軸群の減速時間です パラメーター単位: ms 入力範囲: ゼロ以外の正の値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.18 GetGrpSMTime

目的

軸グループのプロファイルスムーズタイムを取得します

構文

```
double GetGrpSMTime(
    int group_id
);
```

パラメーター

group_id [in] Axis group index

戻りの値

軸グループのプロファイルスムーズ時間

単位: ms

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.19 SetGrpSMTime

目的

軸グループのプロファイルスムーズ時間を設定します

構文

```
int SetGrpSMTime(
    int    group_id,
    double smooth_time
);
```

パラメーター

group_id [in]	Axis group index
smooth_time [in]	軸グループの新しいプロファイルスムーズ時間 パラメーター単位: ms 入力範囲: 0 ～ 500

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.20 GetGrpCoordSys

目的

軸グループの座標系を取得します

構文

```
double GetGrpCoordSys(
    int group_id
);
```

パラメーター

group_id [in] Axis group index

戻りの値

軸グループの座標系。詳細については、セクション 8.1.2 を参照してください。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.21 SetGrpCoordSys

目的

軸グループの座標系を設定します

構文

```
int SetGrpCoordSys(
    int group_id,
    int coord_sys
);
```

パラメーター

group_id [in]	Axis group index
coord_sys [in]	軸グループの新しい座標系。詳細については、セクション 8.1.2 を参照してください。 例: 1. CS_MCS 2. CS_WCS1 | CS_PCS 3. CS_OFFSET | CS_WCS2 | CS_PCS

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

8.3.22 GetGrpBufferMode

目的

軸グループのバッファーモードを取得します

構文

```
double GetGrpBufferMode(
    int group_id
);
```

パラメーター

group_id [in] Axis group index

戻りの値

軸グループのバッファーモード。詳細については、セクション 8.1.4 を参照してください。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.23 SetGrpBufferMode

目的

軸グループのバッファーモードを設定します

構文

```
int SetGrpBufferMode(
        int group_id,
        int buffer_mode
);
```

パラメーター

group_id [in]	Axis group index
buffer_mode [in]	軸グループの新しいバッファーモード。詳細については、セクション 8.1.4 を参照してください。 入力範囲: 0 ～ 5

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.24 GetGrpTransMode

目的

軸グループの Transition モードを取得します

構文

```
double GetGrpTransMode(
    int group_id
);
```

パラメーター

group_id [in] Axis group index

戻りの値

軸グループの Transition モード。詳細については、セクション 8.1.5 を参照してください。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.25 SetGrpTransMode

目的

軸グループの Transition モードを設定します

構文

```
int SetGrpTransMode(
    int group_id,
    int trans_mode
);
```

パラメーター

group_id [in]	Axis group index
trans_mode [in]	軸グループの新しい Transition モード。詳細については、セクション 8.1.5 を参照してください。 入力範囲: 0 ～ 4

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.26 SetGrpTransPrm

目的

軸グループの Transition モードのパラメーターを設定します。

構文

```
int SetGrpTransPrm(
    int    group_id,
    double trans_vel,
    double trans_dis
);
```

パラメーター

group_id [in]	Axis group index
trans_vel [in]	軸グループの新しい Transition モードの速度パラメーター 詳細については、セクション 8.1.5 を参照してください。
trans_dis [in]	軸グループの新しい Transition モードの距離パラメーター。 詳細については、セクション 8.1.5 を参照してください。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.27 GetGrpCmdNum

目的

コマンドバッファー内の軸グループのコマンド数を取得します。

構文

```
double GetGrpCmdNum(
    int group_id
);
```

パラメーター

group_id [in] Axis group index

戻りの値

コマンドバッファー内の軸グループのコマンド数

条件

サポートされる最小バージョン	iA Studio 1.3

8.3.28 SetGrpVelScale

目的

軸群モーションの速度スケールを設定します

構文

```
int SetGrpVelScale(
    int    group_id,
    double vel_scale
);
```

パラメーター

group_id [in]	Axis group index
vel_scale [in]	軸グループモーションの新しい速度スケール 入力範囲: 0 ～ 100

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.4

8.3.29 GetGrpVelScale

目的

軸群モーションの速度スケールを取得します

構文

```
double GetGrpVelScale(
    int group_id
);
```

パラメーター

group_id [in] Axis group index

戻りの値

軸群モーションの速度スケール。その範囲は 0 から 100 です。

条件

サポートされる最小バージョン	iA Studio 1.4

8.3.30 GetGrpCoordTrans

目的

軸グループの座標系の変換パラメーターを取得します。

構文

```
int GetGrpCoordTrans(
    int group_id,
    int coord_sys,
    double *trans_param
);
```

パラメーター

group_id [in]	Axis group index
coord_sys [in]	座標系。詳細については、セクション 8.1.2 を参照してください。
trans_param [out]	6-DOF {X, Y, Z, A, B, C}の変換パラメーターを含む 6 要素配列へのポインター パラメーター単位: mm (X, Y, Z)、deg (A, B, C)

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

8.3.31 SetGrpCoordTrans

目的

軸グループの座標系の変換パラメーターを設定します

構文

```
int SetGrpCoordTrans(
        int group_id,
        int coord_sys,
        double *trans_param
);
```

パラメーター

group_id [in]	Axis group index
coord_sys [in]	座標系。詳細については、セクション 8.1.2 を参照してください。
trans_param [in]	6-DOF {X, Y, Z, A, B, C}の変換パラメーターを含む 6 要素配列へのポインター パラメーター単位: mm (X,Y,Z)、deg (A,B,C)

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

8.3.32 GetGrpPoseCmd

目的

軸群座標系の姿勢指令を取得します

構文

```
int GetGrpPoseCmd(
        int group_id,
        int coord_sys,
        double *pose_cmd
);
```

パラメーター

group_id [in]	Axis group index
coord_sys [in]	座標系。詳細については、セクション 8.1.2 を参照してください。
pose_cmd [out]	6-DOF {X, Y, Z, A, B, C}のポーズコマンドを含む 6 要素配列へのポインター パラメーター単位: mm (X, Y, Z),deg (A, B, C)

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

8.3.33 GetGrpPoseFb

目的

軸グループの座標系のポーズフィードバックを取得します

構文

```
int GetGrpPoseFb(
    int group_id,
    int coord_sys,
    double *pose_fb
);
```

パラメーター

group_id [in]	Axis group index
coord_sys [in]	座標系。詳細については、セクション 8.1.2 を参照してください。
pose_fb [out]	6-DOF {X, Y, Z, A, B, C}のポーズフィードバックを含む 6 要素配列へのポインター パラメーター単位: mm (X, Y, Z),deg (A, B, C)

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

8.4 グループの状態

8.4.1 IsGrpEnabled

目的
軸グループの Enable 状態を問い合わせます

構文

```
int IsGrpEnabled(
        int group_id
);
```

パラメーター
group_id [in] Axis group index

戻りの値
軸グループが「GrpEnabled」状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

8.4.2 IsGrpMoving

目的

軸グループの Moving 状態を問い合わせます。軸グループが移動している場合、PG (プロファイルジェネレーター)は新しい位置を出力し続けます。

構文

```
int IsGrpMoving(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

軸グループが GrpMoving 状態にある場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

8.4.3 IsGrpInPos

目的

軸グループの in-position 状態を照会します。軸グループがインポジションの場合、グループ内のすべての軸がインポジションです。

構文

```
int IsGrpInPos(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

軸グループが GrpInPos 状態にある場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.24

8.4.4 IsGrpErrorStop

目的

軸グループが error stop 状態かどうかを問い合わせます。

構文

```
int IsGrpErrorStop(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

軸グループが GrpErrorStop 状態にある場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

8.5 高度なグループモーションコントロール

8.5.1 LinAbs

目的

特定の座標系の絶対位置に向かって、軸グループ上で補間された直線移動を命令すること。

構文

```
int LinAbs(
        int group_id,
        double *target_pos,
        double *motion_profile,
        int coord_sys,
        int buff_mode,
        int trans_mode,
        double trans_par
);
```

パラメーター

group_id [in]	Axis group index
target_pos [in]	6-DOF {X、Y、Z、A、B、C}の絶対ターゲット位置と方向を含む 6 要素配列へのポインター パラメーター単位: mm (X, Y, Z),deg (A, B, C)
motion_profile [in]	パス上の TCP の最大接線モーションプロファイルを含む 4 要素配列へのポインター。{max_velocity、max_acceleration、max_deceleration、smooth_time} 詳細については、セクション 8.3.6 SetGrpMotionProfile を参照してください
coord_sys [in]	該当する座標系を指定します。 詳細については、セクション 8.1.2 を参照してください。
buff_mode [in]	パスバッファーモードを指定します。 詳細については、セクション 8.1.4 を参照してください。
trans_mode [in]	パス transition モードを指定します。 詳細については、セクション 8.1.5 を参照してください。

trans_par [in] 特定の transition モードのパラメーターを指定します。
詳細については、セクション 8.1.5 を参照してください。
不要な場合はゼロを記入してください。

戻りの値

関数が成功した場合は、int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

8.5.2 LinRel

目的

特定の座標系の相対位置に向かって、軸グループ上で補間された直線移動を命令すること。

構文

```
int LinRel(
        int group_id,
        double *relative_dist,
        double *motion_profile,
        int coord_sys,
        int buff_mode,
        int trans_mode,
        double trans_par
);
```

パラメーター

group_id [in]	Axis group index
relative_dist [in]	6-DOF {X、Y、Z、A、B、C}の相対距離を含む 6 要素配列へのポインター。 パラメーター単位: mm (X, Y, Z),deg (A, B, C)
motion_profile [in]	パス上の TCP の最大接線モーションプロファイルを含む 4 要素配列へのポインター。{max_velocity、max_acceleration、max_deceleration、smooth_time} 詳細については、セクション 8.3.6 SetGrpMotionProfile を参照してください。
coord_sys [in]	該当する座標系を指定します。 詳細については、セクション 8.1.2 を参照してください。
buff_mode [in]	パスバッファーモードを指定します。 詳細については、セクション 8.1.4 を参照してください。
trans_mode [in]	パス transition モードを指定します。 詳細については、セクション 8.1.5 を参照してください。
trans_par [in]	特定の transition モードのパラメーターを指定します。 詳細については、セクション 8.1.5 を参照してください。 不要な場合はゼロを記入してください。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

8.5.3 CircAbs

目的

特定の座標系の絶対位置に向かって、軸グループ上で補間された円運動を指令します。

構文

```
int CircAbs(
    int group_id,
    double *center_pos,
    double *normal_vector,
    int turns,
    double *target_pos,
    double *motion_profile,
    int coord_sys,
    int buff_mode,
    int trans_mode,
    double trans_par
);
```

パラメーター

group_id [in]	Axis group index
center_pos [in]	円形パスの絶対中心位置 {X, Y, Z}を含む 3 要素配列へのポインター。 パラメーター単位: mm
normal_vector [in]	右手の法則による回転の法線ベクトル {X, Y, Z}を含む 3 要素配列へのポインター。 パラメーター単位: mm
turns [in]	始点を基準とした円形パスのターン数。 円形パスの方向と合計角度を決定します。
target_pos [in]	6-DOF {X、Y、Z、A、B、C}の絶対ターゲット位置と方向を含む 6 要素配列へのポインター。 パラメーターの単位: mm (X,Y,Z)、deg (A,B,C)
motion_profile [in]	パス上の TCP の最大接線モーションプロファイルを含む 4 要素配列へのポインター。

{max_velocity、max_acceleration、max_deceleration、smooth_time}
詳細については、SetGrpMotionProfile セクションを参照してください。

coord_sys [in] 該当する座標系を指定します。
詳細については、セクション 8.1.2 を参照してください。

buff_mode [in] パスバッファーモードを指定します。
詳細については、セクション 8.1.4 を参照してください。

trans_mode [in] パス transition モードを指定します。
詳細については、セクション 8.1.5 を参照してください。

trans_par [in] 特定の transition モードのパラメーターを指定します。
詳細については、セクション 8.1.5 を参照してください。
不要な場合はゼロを記入してください。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

8.5.4 CircRel

目的

特定の座標系の相対位置に向かって、軸グループ上で補間された円運動を指令します。

構文

```
int CircRel(
        int group_id,
        double *center_pos,
        double *normal_vector,
        int turns,
        double *relative_dist,
        double *motion_profile,
        int coord_sys,
        int buff_mode,
        int trans_mode,
        double trans_par
);
```

パラメーター

group_id [in] Axis group index

center_pos [in] 円形パスの絶対中心位置{X, Y, Z}を含む 3 要素配列へのポインター。
パラメーター単位: mm

normal_vector [in] 右手の法則による回転の法線ベクトル {X, Y, Z}を含む 3 要素配列へのポインター。
パラメーター単位: mm

turns [in] 始点を基準とした円形パスのターン数。
円形パスの方向と合計角度を決定します。

relative_dist [in] 6-DOF {X、Y、Z、A、B、C}の相対距離を含む 6 要素配列へのポインター。
パラメーター単位: mm (X, Y, Z),deg (A, B, C)

motion_profile [in] パス上の TCP の最大接線モーションプロファイルを含む 4 要素配列へのポインター。{max_velocity、max_acceleration、max_deceleration、smooth_time}
詳細については、セクション 8.3.6 SetGrpMotionProfile を参照してください。

coord_sys [in] 該当する座標系を指定します。
詳細については、セクション 8.1.2 を参照してください。

buff_mode [in] パスバッファーモードを指定します。
詳細については、セクション 8.1.4 を参照してください。

trans_mode [in] パス transition モードを指定します。
詳細については、セクション 8.1.5 を参照してください。

trans_par [in] 特定の transition モードのパラメーターを指定します。
詳細については、セクション 8.1.5 を参照してください。
不要な場合はゼロを記入してください。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

9. GPIO 機能

9.1 概要

HIMC は、8 セットの汎用入出力(GPIO)ピンを提供します。ハードウェアの遅延時間は 1ms 以内で、電源は 24V です。スレーブデバイスは、MoE 通信を介してコントローラーに接続し、その IO ステータスを更新できます。IO の数は、スレーブデバイスによって異なります。この章で提供される SetGPO や SetSlvGPO などの機能を使用して、ユーザーは HIMC とスレーブの出力ピンの信号をそれぞれ設定できます。また、ユーザーは入力/出力ピンの信号ステータスを照会できます。iA Studio の機能モジュール「デジタル IO」(「iA Studio ユーザーガイド」のセクション 4.4 を参照）を使用すると、HIMC およびスレーブの入出力状態を監視および設定できます。

HIMC のデジタル入力 I8 は非常停止信号(「HIMC インストールガイド」のセクション 3.3 を参照)であり、立ち上がりエッジ信号を受信するとトリガーされます。この時点で、すべての軸が無効になり、すべての HMPL タスクが停止します。

注: 立ち上がりエッジ信号がトリガーされた後、ユーザーは軸を再度有効にするか、HMPL タスクを再実行できます。

9.1.1 GPIO 変数

GPIO 変数は、コントローラーの汎用入出力変数(システム変数)とスレーブの汎用入出力変数(スレーブ変数)の 2 つのカテゴリに分けられます。ユーザーは、iA Studio のスコープマネージャーを介して目的の変数を選択できます(「iA Studio ユーザーガイド」のセクション 4.8 を参照)。詳細な説明を表 9.1.1.1 および表 9.1.1.2 に示します。

表 9.1.1.1 コントローラーの汎用入出力変数

名称	変数	単位	説明
HIMC GPO	himc_gpo_bits	N/A	コントローラーの汎用出力ピンの状態
HIMC GPI	himc_gpi_bits	N/A	コントローラーの汎用入力ピンの状態

表 9.1.1.2 スレーブの汎用入出力変数

名称	変数	単位	説明
Slave GPO	gpoStBits_1	N/A	スレーブの汎用出力ピンの状態
Slave GPI	gpiStBits_1	N/A	スレーブの汎用入力ピンの状態

9.1.2 例

例 1

コントローラーの汎用入力 4 番ピンで、入力の立ち上がりエッジ信号を検出すると、すべての軸を無効にし、コントローラーの汎用出力 3 番ピンの状態を 1 に設定します。

```
void main() {
    int last_state = 0;
    while (1) {
        int in = IsGPI_On(4);
        /*  Detect the rising-edge signal of the 4th general purpose input pin of
    controller  */
        if ( (in ^ last_state) & in) {
            DisableAll();  //  Disable all axes
            SetGPO(3, 1);  /*  Set the state of the 3rd general purpose output pin
                of controller as 1  */
        }
        last_state = in;  /*  Record the state of controller's general purpose
            input  */
    }
}
```

例 2

スレーブ 1 の 3 番目の汎用入力ピンで、入力の立ち下がりエッジ信号が検出されると、スレーブ 1 の 2 番目の汎用出力ピンの状態をトグルし、コントローラーのすべての汎用出力ピンの状態を 1 に設定します。

```
void main() {
    int last_state = 0;
    while (1) {
        int in = IsSlvGPI_On(1, 3);
        /*  Detect the falling-edge signal of the 3rd general purpose input pin of  slave 1
*/
        if ( (in ^ last_state) && !in) {
            ToggleSlvGPO(1, 2);  /*  Toggle the state of 2nd general purpose
                output pin of slave 1  */
            SetAllGPO(0xff);  /*  Set the state of all general purpose output
                pins of controller as 1  */
        }
        last_state = in;  //  Record the state of slave's general purpose input
```

```
    }
}
```

9.2 コントローラーの IO 設定

9.2.1 SetGPO

目的
コントローラーの汎用出力の状態を設定します。

構文

```
int SetGPO(
    int gpo_idx,
    int on_off
);
```

パラメーター

gpo_idx [in]	汎用出力 index
on_off [in]	設定する状態。1 がオン、0 がオフです。

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合は int 値-1 を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

9.2.2 ToggleGPO

目的
コントローラーの汎用出力の状態を切り替えます。

構文

```
int ToggleGPO(
        int gpo_idx
);
```

パラメーター
gpo_idx [in]　　汎用出力 index

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合は int 値-1 を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

9.2.3 SetAllGPO

目的

コントローラーのすべての汎用出力の状態を切り替えます。

構文

```
int SetAllGPO(
        int all_gpo_state
);
```

パラメーター

all_gpo_state [in]　　すべての汎用出力の状態値。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は int 値-1 を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

9.2.4 SetGPIInvert

目的

コントローラーの汎用入力の反転状態を設定します。

構文

```
int SetGPIInvert(
    int gpi_idx,
    int invert
);
```

パラメーター

gpi_idx [in]	汎用入力 index
invert [in]	設定する反転状態。1 は反転、0 は非反転です。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は int 値-1 を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

9.2.5 SetGPOInvert

目的

コントローラーの汎用出力の反転状態を設定します。

構文

```
int SetGPOInvert(
    int gpo_idx,
    int invert
);
```

パラメーター

gpo_idx [in]	汎用出力 index
invert [in]	設定する反転状態。1 は反転、0 は非反転です。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は int 値-1 を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

9.2.6 BindEMO

目的

汎用入力ピンを E-Stop にバインドするように設定します。

構文

```
int BindEMO(
    int gpi_idx
);
```

パラメーター

gpi_idx [in] 汎用入力 index。デフォルト値は 8 です。
0 に設定すると、すべての汎用入力ピンが E-Stop にバインドされません。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は int 値-1 を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

9.3 スレーブ IO 設定

9.3.1 SetSlvGPO

目的

スレーブの汎用出力の状態を設定します。

構文

```
int SetSlvGPO(
        int slave_id,
        int gpo_idx,
        int on_off
);
```

パラメーター

slave_id [in]	スレーブ ID
gpo_idx [in]	汎用出力 index
on_off [in]	設定する状態。1 がオン、0 がオフです。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は int 値-1 を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

9.3.2 ToggleSlvGPO

目的

スレーブの汎用出力の状態を切り替えます。

構文

```
int ToggleSlvGPO(
    int slave_id,
    int gpo_idx
);
```

パラメーター

slave_id [in]	スレーブ ID
gpo_idx [in]	汎用出力 index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は int 値-1 を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

9.3.3 SetSlvAllGPO

目的

スレーブのすべての汎用出力の状態を切り替えます。

構文

```
int SetSlvAllGPO(
    int slave_id,
    int all_gpo_state,
    int slot_idx = 0
);
```

パラメーター

slave_id [in]	スレーブ ID
all_gpo_state [in]	すべての汎用出力の状態値
slot_idx [in]	デジタル出力モジュールの取り付けスロットのインデックス。 スレーブがサーボドライバーの場合、このパラメーターはオプションです。スレーブが ETA3 の場合は、モジュールの取り付けスロットを埋めます。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は int 値-1 を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

9.4 コントローラーIO 状態

9.4.1 GetGPI

目的
コントローラーの汎用入力の状態をクエリします。

構文

```
int   GetGPI(
   int gpi_idx
);
```

パラメーター
gpi_idx [in]　　汎用入力インデックス。

戻りの値
入力が GPI_On 状態にある場合は、int 値 TRUE (1) を返します。 それ以外の場合は、FALSE (0) を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

9.4.2 GetGPO

目的
コントローラーの汎用出力の状態をクエリします。

構文

```
int GetGPO(
    int gpo_idx
);
```

パラメーター
gpo_idx [in]　　汎用出力 index

戻りの値
出力が GPO_On 状態の場合、int 値 TRUE (1) を返します。 それ以外の場合は FALSE が返されます (0)。

条件

サポートされる最小バージョン	iA Studio 3.0

9.4.3 GetAllGPI

目的

コントローラーのすべての汎用入力の状態を取得します。

構文

```
int GetAllGPI();
```

パラメーター

N/A

戻りの値

コントローラーのすべての汎用入力の状態値
1 番目と 4 番目の GPI ピンが TURE の場合、9 を返します。

条件

サポートされる最小バージョン	iA Studio 1.3

9.4.4 GetAllGPO

目的

コントローラーのすべての汎用出力の状態を取得します。

構文

```
int GetAllGPO();
```

パラメーター

N/A

戻りの値

コントローラーのすべての汎用出力の状態値。
2 番目と 3 番目の GPO ピンが TURE の場合、6 が返されます。

条件

サポートされる最小バージョン	iA Studio 1.3

9.4.5 GetAllGPIInvertSt

目的

コントローラーのすべての汎用入力の反転状態を取得します。

構文

```
int GetAllGPIInvertSt();
```

パラメーター

N/A

戻りの値

コントローラーのすべての汎用入力の状態値を反転します。
1 番目と 4 番目の GPI 反転ピンが TURE の場合、9 が返されます。

条件

サポートされる最小バージョン	iA Studio 2.0

9.4.6 GetAllGPOInvertSt

目的

コントローラーのすべての汎用出力の反転状態を取得します。

構文

```
int GetAllGPOInvertSt();
```

パラメーター

N/A

戻りの値

コントローラーのすべての汎用出力の状態値を反転します。
2 番目と 3 番目の GPO 反転ピンが TURE の場合、6 が返されます。

条件

サポートされる最小バージョン	iA Studio 2.0

9.5 スレーブ IO 状態

9.5.1 GetSlvGPI

目的
スレーブの汎用入力の状態をクエリします。

構文

```
Int   GetSlvGPI(
   int slv_slot_id,
   int gpi_idx
);
```

パラメーター

slv_slot_id [in]	スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。
gpi_idx [in]	汎用入力 index

戻りの値
入力が SlvGPI_On 状態にある場合は、int 値 TRUE (1) を返します。 それ以外の場合は FALSE が返されます

備考
この機能を使用する場合、ユーザーはデジタル入力オブジェクトを PDO として設定する必要があります。 たとえば、ドライバーの 0x60FE(デジタル入力) を PDO として設定します。

条件

サポートされる最小バージョン	iA Studio 3.0

9.5.2 GetSlvGPO

目的
スレーブの汎用出力の状態をクエリします。

構文

```
int GetSlvGPO(
    int slv_slot_id,
    int gpo_idx
);
```

パラメーター

slv_slot_id [in]　スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。

gpo_idx [in]　汎用出力指数

戻りの値
出力が SlvGPO_On 状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

備考
この機能を使用する場合、ユーザーはデジタル出力オブジェクトを PDO として設定する必要があります。 たとえば、ドライバーの 0x60FE(デジタル出力) を PDO として設定します。

条件

サポートされる最小バージョン	iA Studio 3.0

(このページはブランクになっています)

10. AIO 機能

10.1 概要

AIO 機能により、アナログ入力(AI)またはアナログ出力(AO)機能を備えたスレーブは、関連するパラメーターを読み取り、設定することができます。現在、AIO 関数は ETA3 にのみ適用されます。ドライバーのアナログ信号ピンの信号操作はサポートできません。ETA3 の AIO モジュールは、4 組のアナログ入力と 8 組のアナログ出力(4 組の電圧型アナログ出力、4 組の電流型アナログ出力)を提供します。設定に基づいて、アナログ入力は電圧または電流になります。測定・出力できる電圧範囲は±10V、電流範囲は 4～20mA です。詳細な仕様については、「ETA3 インストールガイド」を参照してください。

10.1.1 例

例 1
コントローラー変数は、値を監視するためにアナログ出力にバインドされています。

```
void main() {
    int slv_id = 0;
    int slot_index = 3;          //  index of analog output module installation slot
    int ao_index = 1;            //  analog output channel
    int var_id = (HMPL_VEL_FB | HMPL_AXIS_0);  //  velocity feedback of axis 0
    //  the function of analog output bound to controller variable is ON
    int en_flag = 1;
    double var_scale = 1.0;     //  scale of analog output and controller variable
    double var_offset = 0.0;    //  offset of analog output and controller variable
    SetSlvAOMointor(slv_id, slot_index, ao_index, var_id);
    SetSlvAOParam(slv_id, slot_index, ao_index, var_scale, var_offset);
}
```

例 2

電圧出力と測定。

```
void main() {
    int slv_id = 0;
    int slot_index = 3;          //  index of analog output module installation slot
    int ao_index = 1;            //  analog output channel
    int ai_index = 1;            //  analog input channel
    double ao_volt = 6.0;        //  voltage output: 6V
    SetSlvAO(slv_id, slot_index, ao_index, ao_volt);
    Sleep(100);                  //  Wait for analog output to be effective
    Print("%f", GetSlvAI(slv_id, slot_index, ai_index));
}
```

例 3

現在の出力と測定。

```
void main() {
    int slv_id = 0;
    int slot_index = 3;          //  index of analog output module installation slot
    int ao_index = 5;            //  analog output channel
    int ai_index = 1;            //  analog input channel
    int ai_type = 1;             //  current type
    //  scale of analog input and observed physical quantity
    double ai_scale = 1.0;
    //  offset of analog input and observed physical quantity
    double ai_offset = 0.0;
    int ao_curr = 10;            //  current output: 10mA
    SetSlvAIParam(slv_id, slot_index, ai_index, ai_type, ai_scale, ai_offset);
    SetSlvAO(slv_id, slot_index, ao_index, ao_curr);
    Sleep(100);                  //  Wait for analog output to be effective
    Print("%f", GetSlvAI(slv_id, slot_index, ai_index));
}
```

10.2 スレーブ AIO 設定

10.2.1 SetSlvAIType

目的
スレーブのアナログ入力値の変換タイプを設定します。

構文

```
int SetSlvAIType(
    int slv_slot_id,
    int ai_idx,
    int range_type
);
```

パラメーター

slv_slot_id	[in]	スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。
Ai idx	[in]	アナログ入力チャンネル
range_type	[in]	アナログデータ変換。 詳細については、表 10.1 を参照してください。

戻りの値
関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

10.2.2 SetSlvAOType

目的

スレーブのアナログ出力値の変換タイプを設定します。

構文

```
int   SetSlvAOType(
   int slv_slot_id,
   int ao_idx,
   int range_type
);
```

パラメーター

slv_slot_id [in]	スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。
Ai idx [in]	アナログ出力チャンネル。
range_type [in]	アナログデータ変換。 詳細については、表 10.1 を参照してください。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

10.2.3 SetSlvAOHex

目的

スレーブのアナログ出力の 16 進値を設定します。

構文

```
int   SetSlvAOHex(
   int slv_slot_id,
   int ao_idx,
   long long ao_hex_val
);
```

パラメーター

slv_slot_id [in]　スレーブ ID とそのスロット ID。 スレーブにスロットがない場合、スロット ID は無視される可能性があります。

ao idx [in]　アナログ出力チャンネル

ao_hex_val [in]　アナログ 64 ビット出力値

戻りの値

この機能を使用する場合、ユーザーはアナログ出力オブジェクトを PDO として設定する必要があります。

条件

サポートされる最小バージョン	iA Studio 3.0

10.2.4 SetSlvAO

目的

スレーブのアナログ出力値を設定します。

構文

```
int SetSlvAO(
    int slave_id,
    int slot_idx,
    int ao_idx,
    double ao_value
);
```

パラメーター

slave_id [in]	スレーブ ID
slot_idx [in]	アナログ出力モジュールの取り付けスロットのインデックス。
ao_idx [in]	アナログ出力モジュールのアナログ出力チャンネル。
ao_value [in]	アナログ出力値 パラメーター単位: V または mA

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

10.3 スレーブ AIO 状態

10.3.1 GetSlvAIType

目的

スレーブのアナログ入力タイプを取得します。

構文

```
int SetSlvAIType(
        int slv_slot_id,
    int ai_idx
);
```

パラメーター

slv_slot_id [in]　スレーブ ID とそのスロット ID。 スレーブにスロットがない場合、スロット ID は無視される可能性があります。

ai_idx [in]　アナログ入力チャンネル。

戻りの値

スレーブのアナログ入力タイプ。 詳細については、表 10.1 を参照してください。

条件

サポートされる最小バージョン	iA Studio 3.0

10.3.2 GetSlvAOType

目的

スレーブのアナログ出力タイプを取得します。

構文

```
Int GetSlvAOType(
    Int slv_slot_id,
    Int ao_idx
);
```

パラメーター

slv_slot_id [in] スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。

ao_idx [in] アナログ出力チャンネル

戻りの値

スレーブのアナログ出力タイプ。 詳細については、表 10.1 を参照してください。

条件

サポートされる最小バージョン	iA Studio 3.0

10.3.3 GetSlvAIHex

目的

スレーブのアナログ入力の 16 進値を取得します。

構文

```
int GetSlvAIHex(
    int slv_slot_id,
    int ai_idx
);
```

パラメーター

slv_slot [in] スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。

ao_idx [in] アナログ入力チャンネル

戻りの値

アナログ入力の 16 進値

備考

この機能を使用する場合、ユーザーはアナログ入力オブジェクトを PDO として設定する必要があります。

条件

サポートされる最小バージョン	iA Studio 3.0

10.3.4 GetSlvAI

目的

スレーブのアナログ入力を取得します。

構文

```
double GetSlvAI(
    int slv_slot_id,
    int ai_idx
);
```

パラメーター

slv_slot_id [in]	スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。
ai_idx [in]	アナログ入力チャンネル

戻りの値

アナログ入力値

条件

サポートされる最小バージョン	iA Studio 2.0

10.3.5 GetSlvAOHex

目的

スレーブのアナログ出力の 16 進値を取得します

構文

```
int GetSlvAOHex(
    int slv_slot_id,
    int ao_idx
);
```

パラメーター

slv_slot_id [in]　スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。

ao_idx [in]　アナログ出力チャンネル

戻りの値

アナログ出力の 16 進値

備考

この機能を使用する場合、ユーザーはアナログ出力オブジェクトを PDO として設定する必要があります。

条件

サポートされる最小バージョン	iA Studio 3.0

10.3.6 GetSlvAO

目的

スレーブのアナログ出力値を取得します。

構文

```
double GetSlvAO(
    int slv_slot_id,
    int ao_idx
);
```

パラメーター

slv_slot_id [in]　スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。

ao_idx [in]　アナログ出力チャンネル

戻りの値

アナログ出力値

備考

この機能を使用する場合、ユーザーはアナログ出力オブジェクトを PDO として設定する必要があります。

条件

サポートされる最小バージョン	iA Studio 2.0

10.4 HIMC 内部バッファ変数にバインドされたスレーブ AO

10.4.1 SetSlvAOMonitor

目的

コントローラー変数をアナログ出力にバインドするように設定します。

構文

```
int SetSlvAOMonitor(
    int slv_slot_id,
    int ao_idx,
    int var_id
);
```

パラメーター

slv_slot_id [in] スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。

ao_idx [in] アナログ出力チャンネル

var_id [in] コントローラー変数と軸 OF。 コントローラー変数と軸 ID の定義を以下の表に示します。

コントローラー変数の詳細については、セクション 17.1.2 を参照してください。

例: HMPL_AXIS_0 | HMPL_REF_VE

コントローラー変数	定義	コントローラー変数	定義
HMPL_REF_POS	軸の基準位置	HMPL_POS_FB	軸の位置フィードバック
HMPL_REF_VEL	軸の基準速度	HMPL_VEL_FB	軸の速度フィードバック
HMPL_REF_ACC	軸の基準加速度	HMPL_ACC_FB	軸の加速度フィードバック
		HMPL_CUR_FB	軸の電流フィードバック

Axis ID	定義
HMPL_AXIS_0	Axis 0
HMPL_AXIS_1	Axis 1
・・・	・・・
HMPL_AXIS_15	Axis 15

戻りの値

アナログ出力値

備考

この機能を使用する場合、ユーザーはアナログ出力オブジェクトを PDO として設定する必要があります。

条件

サポートされる最小バージョン	iA Studio 2.0

10.4.2 SetSlvAOParam

目的

スレーブのアナログ出力をコントローラー変数にバインドするように設定します。

構文

```
Int SetSlvAOParam(
    Int slv_slot_id,
    Int ao_idx,
    Int ao_en_bind,
    double ao_scale,
    double ao_offset
);
```

パラメーター

slv_slot_id	[in]	スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。
ao_idx	[in]	アナログ出力チャンネル
ao_en_bind	[in]	0：コントローラー変数にバインドされたアナログ出力機能は OFF（デフォルト） 1：コントローラー変数にバインドされたアナログ出力の機能が ON
ao_scale	[in]	アナログ出力とコントローラー変数のスケール
ao_offset	[in]	アナログ出力とコントローラー変数のオフセット

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

10.4.3 GetSlvAOScale

目的

スレーブのアナログ出力スケールとコントローラー変数を取得します。

構文

```
double SetSlvAOScale(
    int slave_id,
    int slot_idx,
    int ao_idx
);
```

パラメーター

slv_slot_id [in]	スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。
ao_idx [in]	アナログ出力チャンネル

戻りの値

スレーブのアナログ出力とコントローラー変数のスケール。

条件

サポートされる最小バージョン	iA Studio 2.0

10.4.4 GetSlvAOOffset

目的

スレーブのアナログ出力とコントローラー変数のオフセットを取得します。

構文

```
double GetSlvAOOffset(
    int slv_slot_id,
    int ao_idx
);
```

パラメーター

slv_slot_id [in]	スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。
ao_idx [in]	アナログ出力チャンネル

戻りの値

スレーブのアナログ出力とコントローラー変数のオフセット。

条件

サポートされる最小バージョン	iA Studio 2.0

10.4.5 IsSlvAOBound

目的

スレーブのアナログ出力がコントローラー変数にバインドされているかどうかをクエリします。

構文

```
Int IsSlvAOBound(
    Int slv_slot_id,
    Int ao_idx
);
```

パラメーター

slv_slot_id [in]	スレーブ ID とスロット ID。 スレーブにスロットがない場合、スロット ID は無視できます。
ao_idx [in]	アナログ出力チャンネル

戻りの値

スレーブが「SlvAOBound」状態にある場合は、int 値 TRUE (1) を返します。 それ以外の場合は、FALSE (0) を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

（このページはブランクになっています。）

11. ユーザーテーブル機能

11.1 概要

HIMC は、最大 512,000 個の double 型変数データ(500K バイト)を格納できる空きメモリ領域をユーザーに提供します。この章で提供される関数を使用すると、ユーザーはメモリ空間にアクセスできます。書き込まれた値は、コントローラーのランダムアクセスメモリ(RAM)に保存されます。関数 SaveUserTable を使用すると、メモリ空間のユーザーテーブルのデータが HIMC のハードディスク空間に保存されます。HIMC の電源を入れ直した後、関数 LoadUserTable を使用して、保存されたデータがユーザーテーブルのメモリ空間に再度コピーされます。

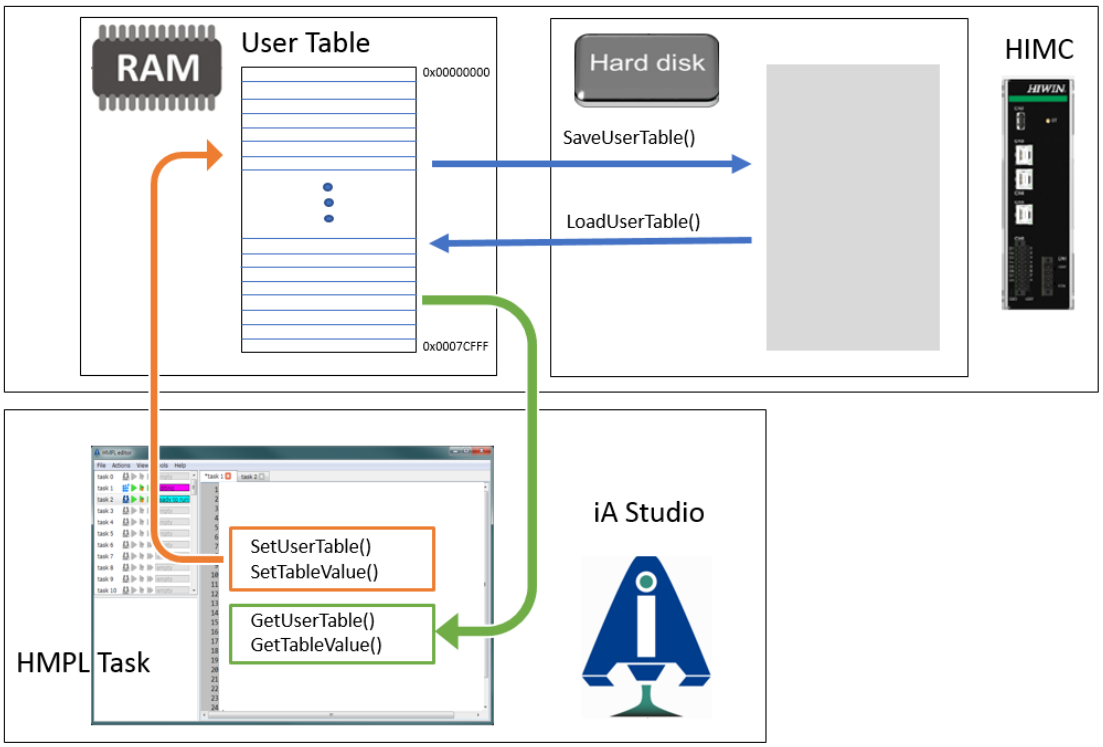


図 11.1.1

注: ユーザーは、iA Studio のテーブルビューアーを介してユーザーテーブルの変数値にアクセスできます(「iA Studio ユーザーガイド」のセクション 4.11 を参照)。これには、HIMC のメモリとハードディスクへの読み込みと保存が含まれます。

注意：
動的エラー補正関数で使用されるエラーマップは、ユーザーテーブルのメモリ空間に保存されます。動的エラー補正を有効にする場合、ユーザーは他のユーザーテーブル値へのアクセスが構築されたエラー補正値に影響を与えないようにする必要があります。

11.2 SetUserTable

目的

ユーザーテーブルにデータを書き込みます。

構文

```
int SetUserTable(
    int     start_idx,
    int     num_data,
    double *input
);
```

パラメーター

start_idx [in] ユーザーテーブルの開始インデックス。

num_data [in] エレメントの数

input [in] 入力データを含む配列へのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
int main() {
    //  Write data to user table
    double data[5] = {-2.0, 0.0, 2.0, 6.0, 4.0};
    SetUserTable(
      1,  //  start index of user table
      5,  //  number of elements
      data  //  pointer to input data array
    );
    //  the above script is the same as below
    system_user_table[1] = -2.0;
    system_user_table[2] = 0.0;
    system_user_table[3] = 2.0;
    system_user_table[4] = 6.0;
    system_user_table[5] = 4.0;
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

11.3 GetUserTable

目的

ユーザーテーブルデータを取得します。

構文

```
int GetUserTable(
    int     start_idx,
    int     num_data,
    double *output
);
```

パラメーター

start_idx [in] ユーザーテーブルの開始インデックス。

num_data [in] 取得するエレメントの数。

output [out] 出力データを含む配列へのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
int main() {
    //  Write data to user table
    double input[5] = {-2.0, 0.0, 2.0, 6.0, 4.0};
    SetUserTable(
      1,  //  start index of user table
      5,  //  number of elements
      input  //  pointer to input data array
    );
    //  now user table has the value "-2.0", "0.0", "2.0", "6.0", "4.0"
    //  Start from index 1

    //  Read user table
    double output[3];
    GetUserTable(
      3,  //  start index of user table
      3,  //  number of elements
      output  //  pointer to output data array
    );
    //  now output[0] = 2.0;
    //  output[1] = 6.0;
    //  output[2] = 4.0;
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

11.4 SetTableValue

目的

ユーザーテーブルの特定のインデックスにデータを書き込みます。

構文

```
int SetTableValue(
    int    index,
    double value
);
```

パラメーター

index [in] ユーザーテーブルのインデックス。

value [in] 入力データ

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

11.5 GetTableValue

目的

ユーザーテーブルの特定のインデックスからデータを取得します。

構文

```
double GetTableValue(
    int index
);
```

パラメーター

index [in] ユーザーテーブルのインデックス。

戻りの値

特定のインデックスのデータ。

条件

サポートされる最小バージョン	iA Studio 1.1

11.6 SaveUserTable

目的

ユーザーテーブルデータを RAM のメモリに保存します。

構文

```
int SaveUserTable(
        int start_idx,
        int num_data
);
```

パラメーター

start_idx [in]　　ユーザーテーブルの開始インデックス

num_data [in]　　格納する要素の数

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
int main() {
    //  Write data to user table
    system_user_table[3] = 2.0;
    system_user_table[4] = 6.0;
    system_user_table[5] = 4.0;

    SaveUserTable(
      3,  //  start index of user table
      3  //  number of elements
    );
    //  Reboot the controller
    //  the value of system_user_table[3] is 2.0
    //  the value of system_user_table[4] is 6.0
    //  the value of system_user_table[5] is 4.0
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

11.7 LoadUserTable

目的

ユーザーテーブルデータをメモリから RAM にロードします。

構文

```
int LoadUserTable(
        int start_idx,
        int num_data
);
```

パラメーター

start_idx [in]　　ユーザーテーブルの開始インデックス。

num_data [in]　　ロードするエレメントの数

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
int main() {
    //  Write data to user table
    system_user_table[3] = 2.0;
    system_user_table[4] = 6.0;
    system_user_table[5] = 4.0;
    SaveUserTable(
      3,   //  start index of user table
      3   //  number of elements
    );
    /* Fill in any value in table[3], table[4] and table[5] */
    system_user_table[3] = 999.0;
    system_user_table[4] = 777.0;
    system_user_table[5] = 888.0;
    LoadUserTable(3, 3);
    //  the value of system_user_table[3] is 2.0
    //  the value of system_user_table[4] is 6.0
    //  the value of system_user_table[5] is 4.0
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

12. ポジショントリガー機能

12.1 概要

HMPL コマンドを使用すると、ユーザーは一部の HIWIN ドライバーで PT(ポジション トリガー)関連の機能を操作できます。PT 関連の機能を操作する前に、互換性のあるドライバーについて HIWIN または地域の代理店にお問い合わせください。

注: HIWIN ドライバーが PT 関連機能を使用するための要件は、(1)デジタルエンコーダー (2)完全な原点復帰手順です。

12.1.1 PT 変数

PT 関連の機能は、表 12.1.1.1 に示す変数に基づいて動作します。

表 12.1.1.1

名称	タイプ	単位	説明	HMPL 関数
状態	int	true / false	PT 機能のステータス。PT がまだ機能しているかどうかを示します。	EnablePT DisablePT IsPTEnabled
開始位置	double	mm または deg	PT 機能の開始位置。PT 出力信号列はここから始まります。	SetPT_StartPos
終了位置	double	mm または deg	PT 機能の終了位置。この時点以降、PT 出力信号は送信されません。	SetPT_EndPos
間隔	double	mm または deg	連続する PT 出力間の位置間隔	SetPT_Interval
パルス幅	int	ns	各 PT 出力信号の幅 1. D1-N シリーズドライバーの場合、範囲は 25 ns から 100,000 ns で、最小増分は 25 ns です。たとえば、25、50、... 100,000 ns です。 2. E1 シリーズドライバーの場合、範囲は 20 ns ～ 80,000 ns で、最小増分は 20 ns です。たとえば、20、40、... 80,000 ns です。	SetPT_PulseWidth
位置配列	double	mm または deg	ランダムインターバル PT 機能のトリガー位置	**エラー! 参照元が見つかりません。**
状態配列	int	high / low	ランダムインターバル PT 機能のステータス出力。	**エラー! 参照元が見つかりません。**
開始インデックス	int	-	ランダム間隔 PT 機能の開始インデックス。	**エラー! 参照元が見つかりません。**
終了インデックス	int	-	ランダム間隔 PT 機能の終了インデックス	**エラー! 参照元が見つかりません。**

図 12.1.1.1 では、極性は「アクティブハイ」に設定されています。

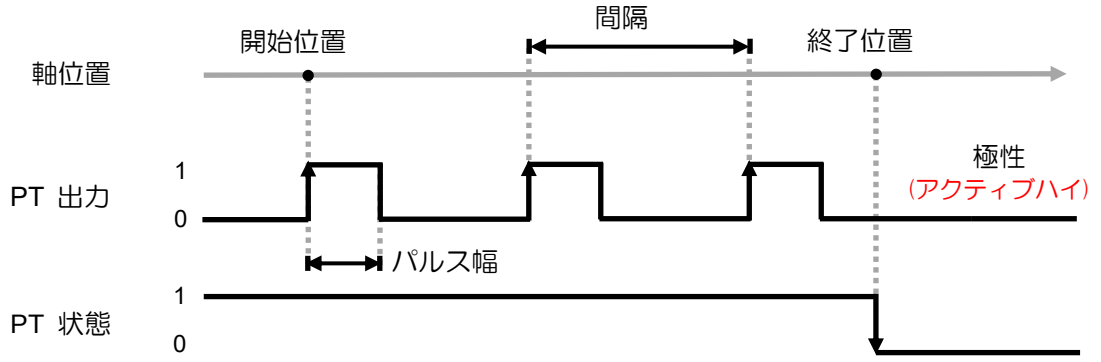


図 12.1.1.1

図 12.1.1.2 では、極性は「アクティブロー」に設定されています。

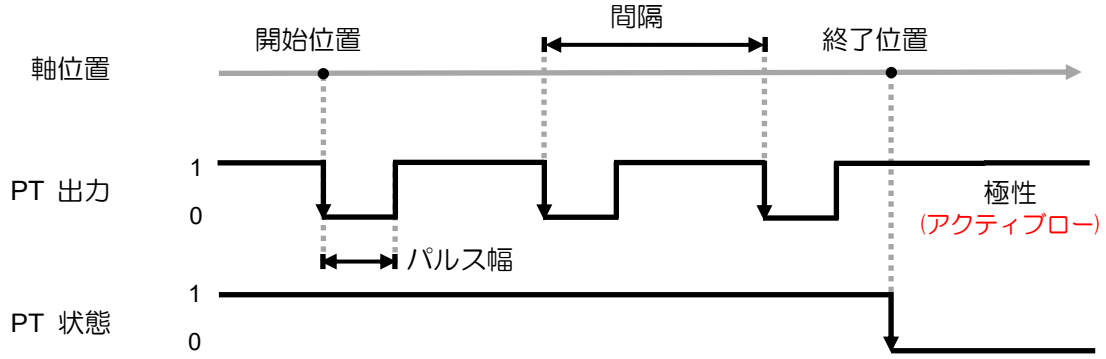


図 12.1.1.2

制限：
PT 機能の間隔と軸の速度は、式「速度 < 間隔 x 位置サンプリングレート」に適合する必要があります。間隔が 100 um に設定され、位置のサンプリングレートが 16 K の場合、速度は 1600 mm/s 未満である必要があります。

注: PT 機能の出力極性(アクティブ ハイ/ロー)を調整するには、ドライバーの HMI に移動して設定します。設定保存後、ドライバーの電源を入れ直して出力極性を有効にします。

12.1.2 PT 機能の利用の流れ

◆ 定間隔 PT 機能

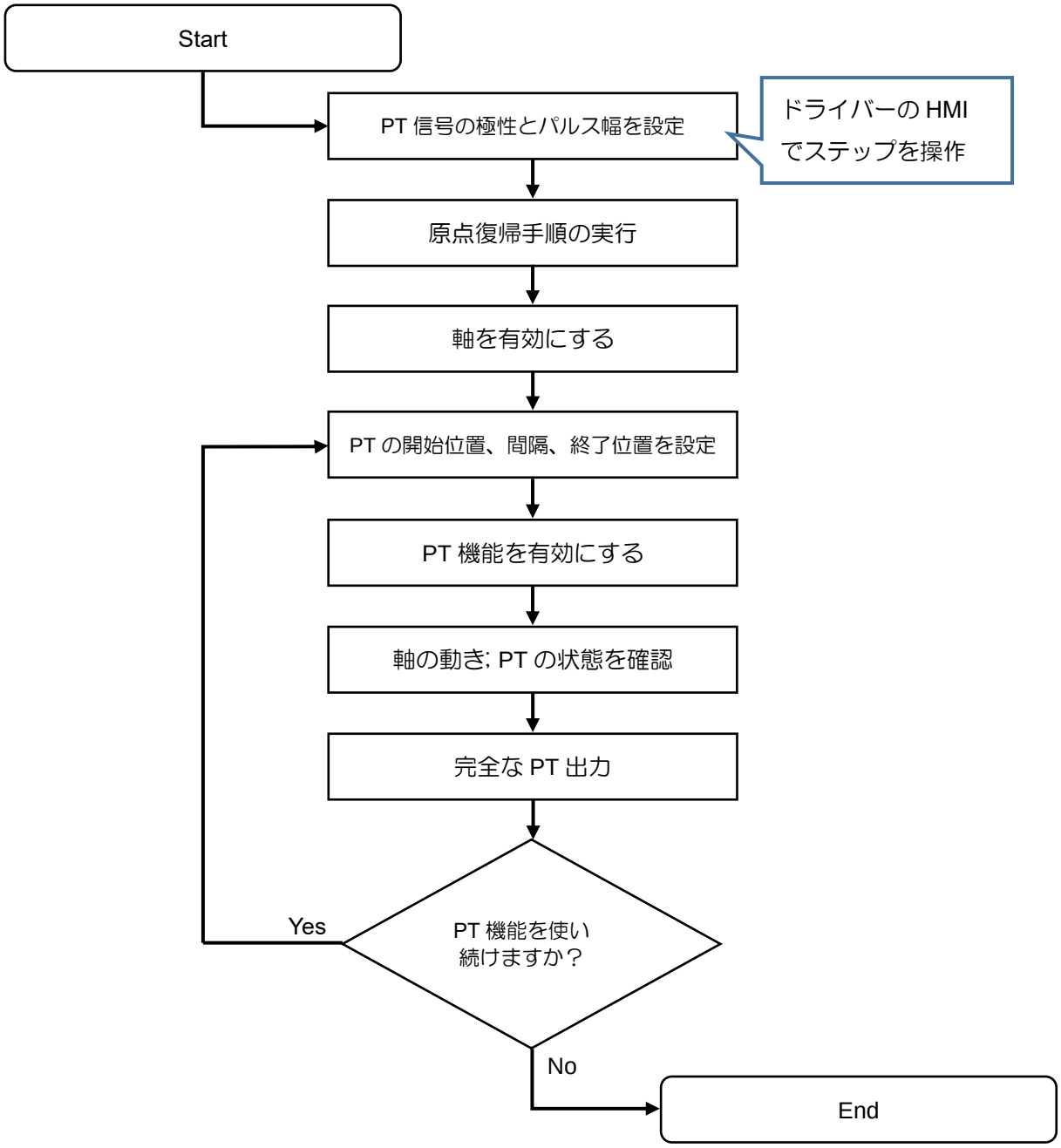


図 12.1.2.1

◆ ランダムインターバル PT 機能

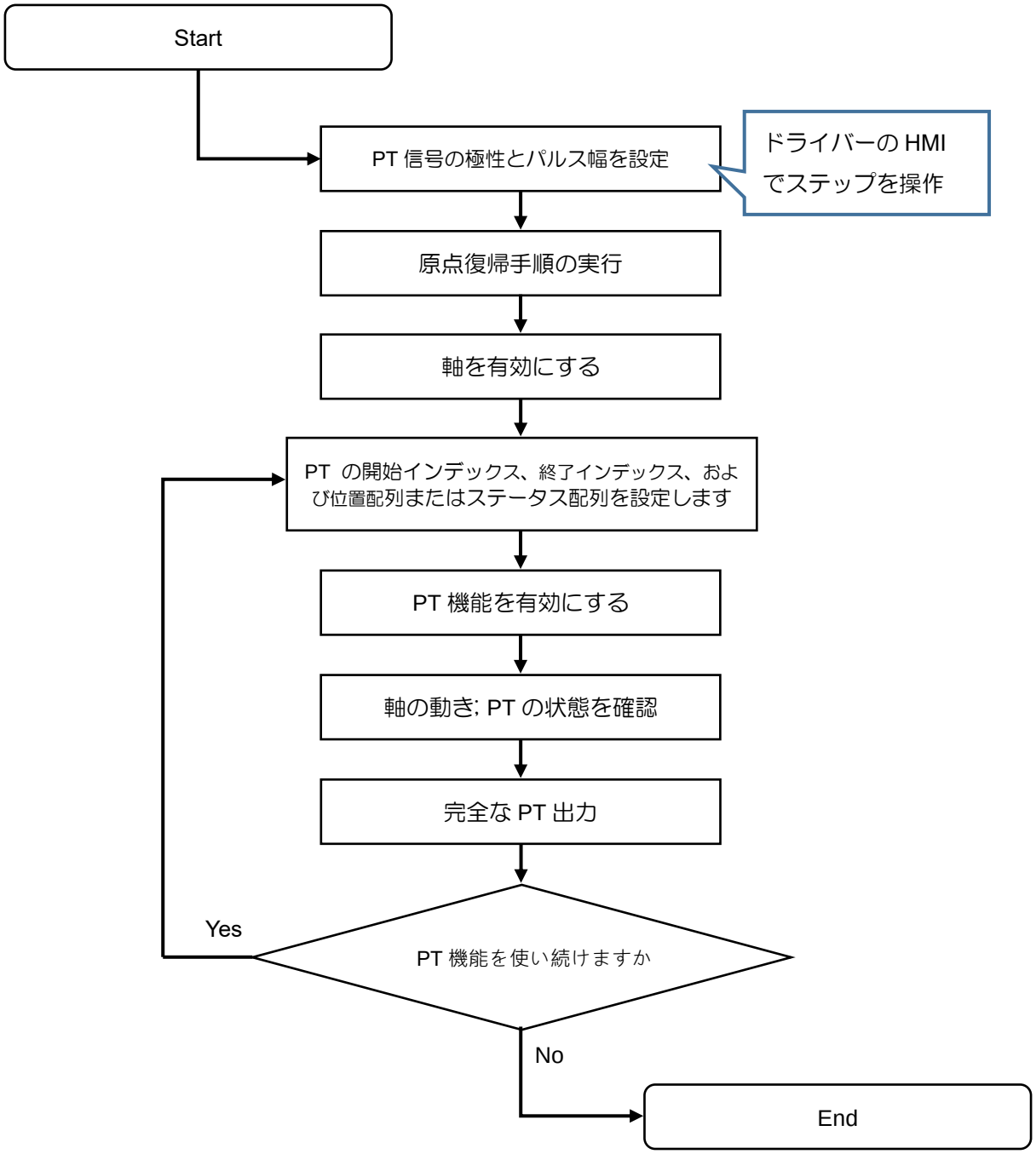


図 12.1.2.2

12.1.3 例

例 1: 定周期 PT 機能

```
void main()
{
    int axis_id = 1;
    double start_pos = 10;      //  start position of PT function: 10 mm
    double end_pos = 20;        //  end position of PT function: 20 mm
    double interval = 1;        //  interval of PT function: 1 mm

    //  pulse width: 1000 ns
    SetPT_PulseWidth(axis_id, 1000);
    //  homing method 33 - homing with index signal from negative direction
    Home(HOME_METHOD_33, axis_id, 0, 20, 0, 10000);
    //  Enable axis
    Enable(axis_id);
    Till(IsEnabled(axis_id));
    //  Move to 0 mm
    MoveAbs(axis_id, 0);
    Till(IsInPos(axis_id));

    // ------ Move toward positive direction ------
    //  Set PT's start position, interval and end position
    SetPT_StartPos(axis_id, start_pos);
    SetPT_Interval(axis_id, interval);
    SetPT_EndPos(axis_id, end_pos);

    //  Enable PT function
    EnablePT(axis_id);

    //  Axis moves from  0 mm to 30 mm
    MoveAbs(axis_id, 30);
    Till(IsInPos(axis_id));

    // ------ Move toward negative direction ------
    //  Set PT's start position and end position
    SetPT_StartPos(axis_id, end_pos);
    SetPT_EndPos(axis_id, start_pos);
```

```
    //  Enable PT function
    EnablePT(axis_id);

    //  Axis moves from 30 mm to 0 mm
    MoveAbs(axis_id, 0.0);
    Till(IsInPos(axis_id));
}
```

例 2: ランダムインターバル PT 機能

```
void main()
{
	int axis_id = 1;
	double pos[11] = {10, 12, 13, 16, 17, 20, 22, 25, 26, 28, 30};

	//	pulse width: 1000 ns
	SetPT_PulseWidth(axis_id, 1000);
	//	homing method 33 - homing with index signal from negative direction
	Home(HOME_METHOD_33, axis_id, 0, 20, 0, 10000);
	//	Enable axis
	Enable(axis_id);
	Till(IsEnabled(axis_id));
	//	Move to 0 mm
	MoveAbs(axis_id, 0);
	Till(IsInPos(axis_id));

	// ------ Move toward positive direction ------
	//	Set PT's trigger positions, start index and end index
	for(int i=0;i<11;i++) {
		SetPT_PosArray(0, i, pos[i]);
	}
	SetPT_StartIndex(0, 0);
	SetPT_EndIndex(0, 10);

	//	Enable PT function
	EnablePT(axis_id);

	//	Axis moves from	0 mm to 40 mm
	MoveAbs(axis_id, 40);
	Till(IsInPos(axis_id));

	// ------ Move toward negative direction ------
	//	Set PT's start index and end index
	SetPT_StartIndex(0, 10);
	SetPT_EndIndex(0, 0);
```

```
    //  Enable PT function
    EnablePT(axis_id);

    //  Axis moves from 40 mm to 0 mm
    MoveAbs(axis_id, 0.0);
    Till(IsInPos(axis_id));
}
```

12.2 EnablePT

目的

軸の位置トリガー機能を有効にします。

構文

```
int EnablePT(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

iA Studio 1.2（同梱）以降で E1 シリーズドライバーの設定に対応。

条件

サポートされる最小バージョン	iA Studio 0.23

12.3 DisablePT

目的

軸の位置トリガー機能を無効にします。

構文

```
int DisablePT(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

iA Studio 1.2（同梱）以降で E1 シリーズドライバーの設定に対応。

条件

サポートされる最小バージョン	iA Studio 0.23

12.4 IsPTEnabled

目的

位置トリガー機能が有効かどうかを問い合わせます。

構文

```
int IsPTEnabled(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸が PTEnabled 状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

備考

iA Studio 1.2（同梱）以降で E1 シリーズドライバーの設定に対応。

条件

サポートされる最小バージョン	iA Studio 0.23

12.5 SetPT_StartPos

目的

位置トリガー機能の開始位置を設定します。

構文

```
int SetPT_StartPos(
    int    axis_id,
    double start_pos
);
```

パラメーター

axis_id [in]	Axis index
start_pos [in]	PT 機能の開始位置 パラメーターの単位: mm または deg

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

iA Studio 1.2（同梱）以降で E1 シリーズドライバーの設定に対応。

条件

サポートされる最小バージョン	iA Studio 0.23

12.6 SetPT_EndPos

目的

ポジショントリガー機能の終了位置を設定します。

構文

```
int SetPT_EndPos(
    int    axis_id,
    double end_pos
);
```

パラメーター

axis_id [in]	Axis index
end_pos [in]	PT 機能の終了位置 パラメーター単位: mm または deg

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

iA Studio 1.2（同梱）以降で E1 シリーズドライバーの設定に対応。

条件

サポートされる最小バージョン	iA Studio 0.23

12.7 SetPT_Interval

目的

位置トリガー機能の位置間隔を設定します。

構文

```
int SetPT_Interval(
    int    axis_id,
    double interval
);
```

パラメーター

axis_id [in]	Axis index
interval [in]	連続する PT 出力間の位置間隔。 パラメーター単位: mm または deg

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

iA Studio 1.2（同梱）以降で E1 シリーズドライバーの設定に対応。

条件

サポートされる最小バージョン	iA Studio 0.23

12.8 SetPT_PulseWidth

目的

位置トリガー機能のパルス幅を設定します。

構文

```
int SetPT_PulseWidth(
    int axis_id,
    int width_ns
);
```

パラメーター

axis_id [in]	Axis index.
width_ns [in]	各 PT 出力信号の幅。 パラメーター単位: ns

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

iA Studio 1.2（同梱）以降で E1 シリーズドライバーの設定に対応。

条件

サポートされる最小バージョン	iA Studio 0.23

13. タッチプローブ機能

13.1 概要

タッチプローブ機能は、原点復帰手順で使用されるラッチ機能です(AC モーター、ダイレクトドライブモーター、リニアモーターに適用可能)。エンコーダー入力信号のエッジトリガでエンコーダーの位置フィードバック値を取り込みます。図 13.1.1 に示すように、ドライバーがエンコーダーインデックス信号を通過すると、タッチプローブ機能がトリガーされ、インデックス信号の位置が記録されます。タッチプローブ機能を使用すると、ユーザーはタッチプローブ機能がトリガーされているかどうかをコントローラーに問い合わせることができ、コントローラーはラッチによって記録されたインデックス信号の位置を取得できます。

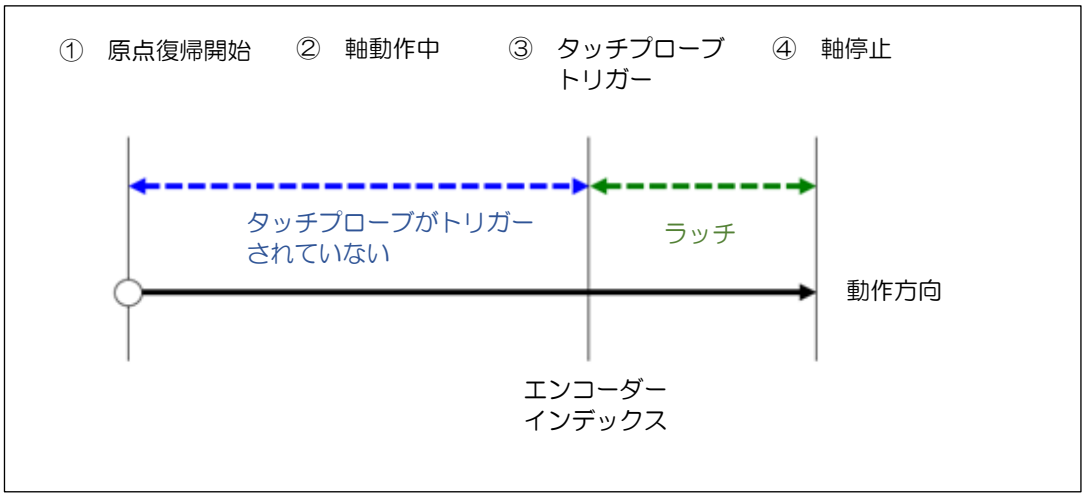


図 13.1.1

13.2 EnableTouchProbe

目的

軸のタッチプローブ機能を有効にします

構文

```
int EnableTouchProbe(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

13.3 DisableTouchProbe

目的

軸のタッチプローブ機能を無効にします。

構文

```
int DisableTouchProbe(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

13.4 IsTouchProbeEnabled

目的

タッチプローブ機能が有効かどうかを問い合わせます。

構文

```
int IsTouchProbeEnabled(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が TouchProbeEnabled 状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

13.5 IsTouchProbeTriggered

目的

タッチプローブ機能がトリガーされているかどうかを問い合わせます。

構文

```
int IsTouchProbeTriggered(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸が TouchProbeTriggered 状態の場合、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

13.6 GetTouchProbePos

目的

軸のタッチプローブ位置を取得します。

構文

```
int GetTouchProbePos(
    int     axis_id,
    double *output
);
```

パラメーター

axis_id [in]	Axis index
output [out]	軸のタッチプローブ位置を受け取るバッファーへのポインター。 パラメーター単位: mm または deg

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

軸がガントリーモードの場合、この機能は適用されません。

条件

サポートされる最小バージョン	iA Studio 0.23

13.7 SetTouchProbeFunc

目的

タッチプローブ機能を設定します。

構文

```
Int SetTouchProbeFunc(
    Int     axis_id,
    int     tp_source,
    int     cont_tigger,
    int     detect_edge
);
```

パラメーター

axis_id [in]	Axis index
tp_source [out]	タッチプローブソース 入力範囲: 0 (タッチ プローブ 1、デフォルト)、1 (タッチ プローブ 2)。
cont_trigger [out]	トリガーモード 入力範囲: 0 (単一トリガー、デフォルト)、1 (連続トリガー)
detect_edge [out]	エッジ検出モード 入力範囲: 0 (立ち上がりエッジ検出、デフォルト)、1 (立ち下がりエッジ検出)

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

14. ダイナミックエラー補償機能

14.1 概要

HIMC は、動的な 1D / 2D / 3D エラー補正機能を提供します。関連するエラー測定と計算結果に基づいて、ユーザーはエラーマップを作成し、HIMC で設定を行うことができます。設定パラメーターには、補正軸、参照軸、マップ点の間隔、マップ点のベース、マップ点の数、および各マップ点の補正値が含まれます。補正値の設定に関しては、HIMC ユーザーテーブルのメモリ空間が使用され、エラーマップの最初の ID 位置がユーザーテーブルに提供されます。

注 1: ユーザーテーブルの使用法と説明については、11 章を参照してください。

注 2: 動的エラー補正を有効にする前に、軸は原点復帰を完了して補正軸と基準軸の座標位置を固定する必要があります。

ユーザーは、同じ軸を補正軸と基準軸の両方として選択することも、複数の異なる軸を補正軸の基準軸として選択することもできます。たとえば、補正軸は Z 軸で、基準軸は X 軸と Y 軸です。補正軸の補正値は、基準軸の移動位置に応じて変化します。軸の移動中、確立されたエラーマップは、線形補間を使用してマップポイント間の補正コマンド値を計算するため、補正値が連続的に維持されます。軸の位置がエラーマップによって確立された範囲を超えると、HIMC は、図 14.1.1 に示すように、最も近いマップポイントの補正値を補正コマンドとして使用します。

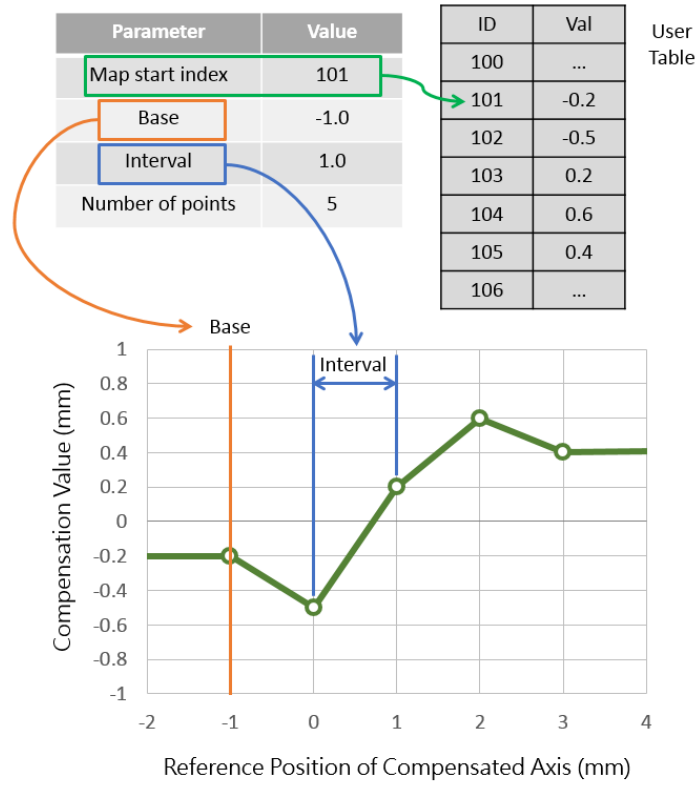


図 14.1.1

動的誤差補正を有効にした後、コントローラーの軸制御コマンドで、出力基準位置は、既知の測定誤差を排除するために補正される変位を追加します。図 5.1.1 に示すように、関係は次のように記述できます：

Reference position + Position compensation = Position output
基準位置 ＋ 位置補正 ＝ 位置出力

動的エラー補正を有効にすると、ユーザーは iA Studio のスコープマネージャーを介して変数を観察できます。

- 補正値：軸 → モーション変数 → 位置補正
- 基準位置 (補正なし)：軸 → モーション変数 → 基準位置
- 基準位置（補正あり）：軸 → モーション変数 → 位置出力

制限:
HIMC では、補正コマンドはプロファイルジェネレーターを経由せず、コントローラーによってプリセットされる最大補正値は 1 mm です。補正値が 1 mm より大きい場合、システムはエラーメッセージを表示してユーザーに通知します。

動的エラー補正を有効にする場合、補正する基準座標を固定する必要があります。したがって、ユーザーは軸の原点オフセットを変更できません。

14.1.1 例

例 1: 一次元の動的エラー補償

この例では、軸 0 が補正軸で、軸 1 が基準軸です。補正軸の補正値は、基準軸の位置によって変化します。それらの関係を図 14.1.1.1 に示します。

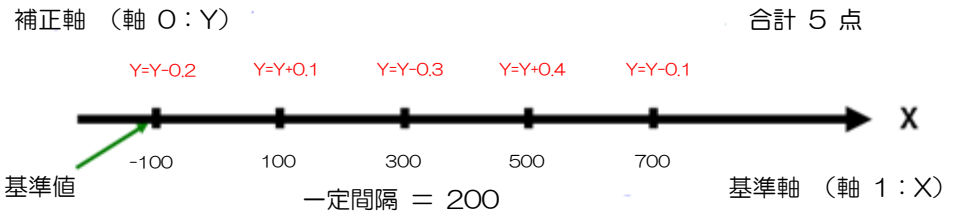


図 14.1.1.1

次の HMPL を使用して補正を設定し、有効にします。その後、補正軸(軸 0)を有効にし、基準軸(軸 1)を移動して補正結果を観察します。

```
void main() {
    //  Set up user map table
    double data[5] = {-0.2, 0.1, -0.3, 0.4, -0.1};
    SetUserTable(1, 5, data);

    SetupComp(
      0,     //  axis to be compensated
      1,     //  start index in user table
      -100, //  base value
      200,   //  interval
      5,     //  number of points (base data included)
      1      //  reference axis (input)
    );
    EnableComp(0);   //  Enable compensation on axis 0
    Enable(0);   //  Enable axis 0 to activate compensation
}
```

ユーザーが補正を無効にすると、軸の基準位置が現在のフィードバックとしてリセットされます。

```
void main() {
    Disable(0);
    Till(!IsEnabled(0));
    DisableComp(0);   //  Disable compensation on axis 0
```

```
}
```

例 2: 二次元の動的エラー補償

この例は、補正軸と基準軸が異なるアプリケーションで、主に XYZ ステージの Z 軸に使用されます。Z 軸は XY 位置が異なると数ミクロンの高さ誤差が生じるためです。この例では、軸 2 が補正軸で、軸 0 と軸 1 が参照軸です。軸 2 の補正値は、軸 0 と軸 1 の位置に応じて変化します。それらの関係を図 14.1.1.2 に示します。

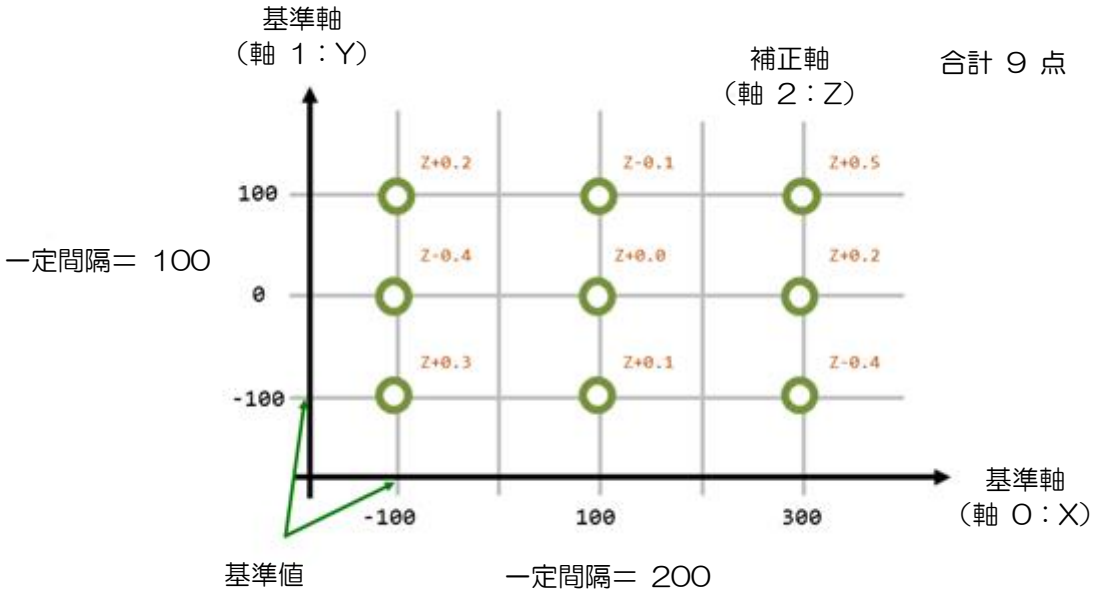


図 14.1.1.2

次の HMPL を使用して補正を設定し、有効にします。その後、補正された軸(軸 2)を有効にし、参照軸(軸 0 と軸 1)を移動して補正結果を観察します。

```
void main() {
    //  Set up user map table
    double data[9] = {0.3, 0.1, -0.4, -0.4, 0.0, 0.2, 0.2, -0.1, 0.5};
    SetUserTable(3, 9, data);

    double base[2] = {-100, -100};
    double interval[2] = {200, 100};
    int num_pt[2] = {3, 3};
    int ref_axis[2] = {0, 1};

    SetupComp2D(
      2,      //  axis to be compensated
      3,      //  start index in user table
```

```
        base,   //  base values
        interval,     //  intervals
        num_pt,       //  number of points (base data included)
        ref_axis      //  reference axes (input)
    );
    EnableComp(2);  //  Enable compensation on axis 2
    Enable(2);  //  Enable axis 2 to activate compensation
}
```

ユーザーが補正を無効にすると、軸の基準位置が現在のフィードバックとしてリセットされます。

```
void main() {
    Disable(2);
    Till(!IsEnabled(2));
    DisableComp(2);  //  Disable compensation on axis 2
}
```

例 3: 三次元動的エラー補償

この例は、精密 XYZ ステージに適用される 3 次元動的エラー補正です。この例では、軸 2 が補正軸で、軸 0、軸 1、軸 2 が参照軸です。軸 2 の補正値は、軸 0、軸 1、軸 2 の位置に応じて変化します。図 14.1.1.3 にパラメーター入力のシーケンスとその補正値を示し、ユーザーマップテーブルの詳細な説明を 図 14.1.1.4～図 14.1.1.6 に示します。

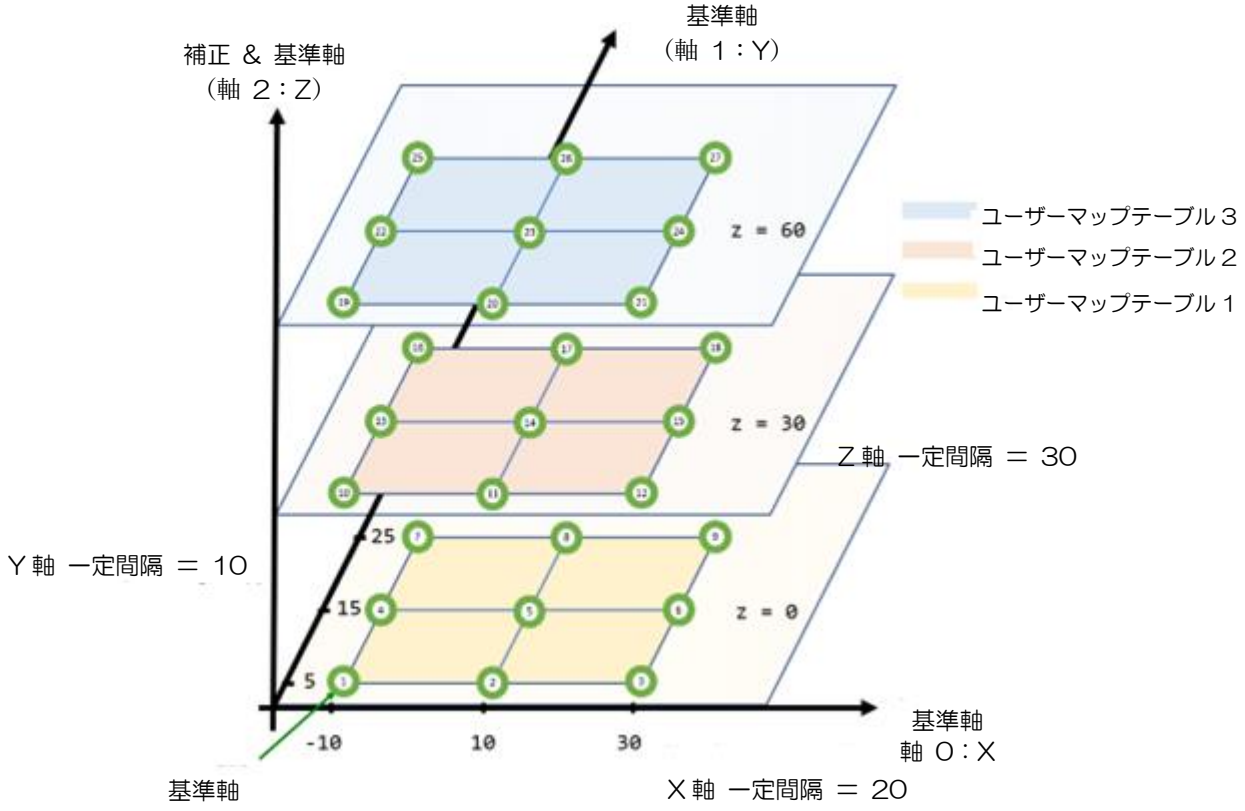


図 14.1.1.3

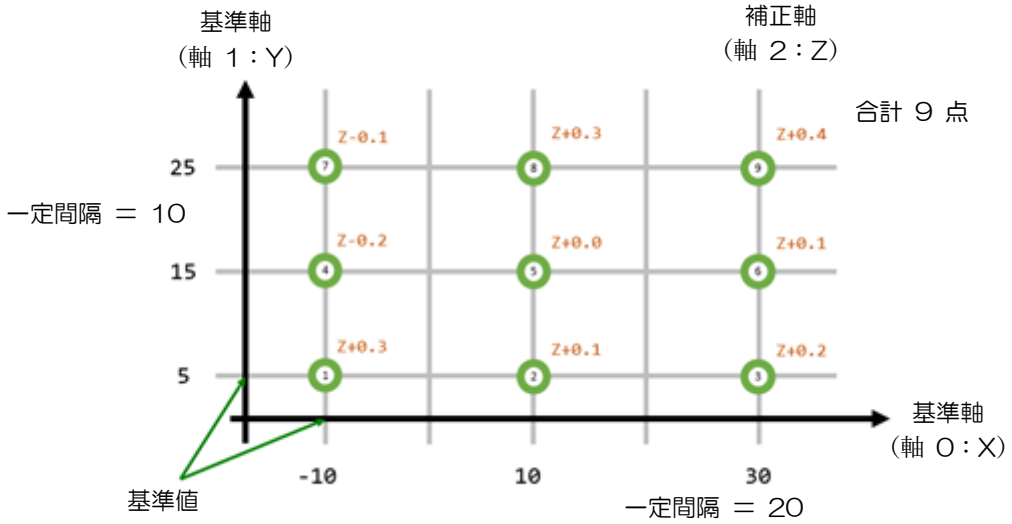


図 14.1.1.4 ユーザーマップテーブル 1

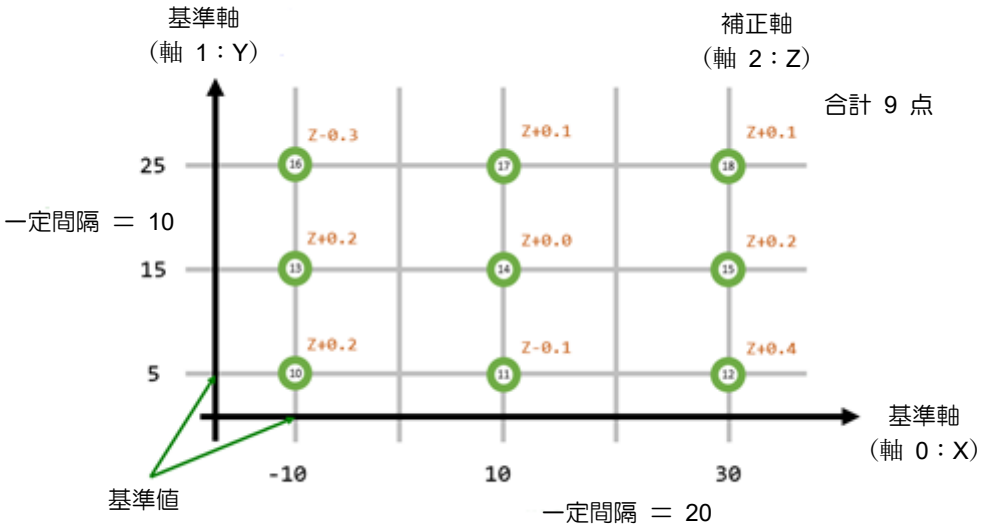


図 14.1.1.5 ユーザーマップテーブル 2

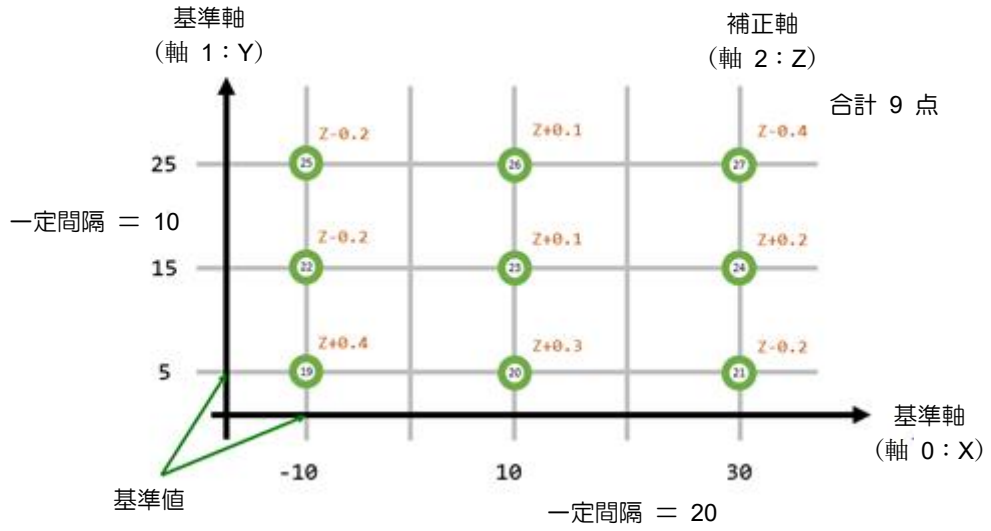


図 14.1.1.6 ユーザーマップテーブル 3

次の HMPL を使用して補正を設定し、有効にします。その後、補正された軸 (軸 2) を有効にし、参照軸(軸 0、軸 1、軸 2)を移動して補正結果を観察します。

```
void main() {
    //  Set up user map table
    double data[27] = {0.3, 0.1, 0.2, -0.2, 0.0, 0.1, -0.1, 0.3, 0.4,   //  1
                       0.2, -0.1, 0.4, 0.2, 0.0, 0.2, 0.3, 0.1, 0.1,   //  2
                       0.4, 0.3, -0.2, -0.2, 0.1, 0.2, -0.2, 0.1, -0.4};   //  3
    SetUserTable(3, 27, data);

    double base[3] = {-10, 5, 0};
```

```
    double interval[3] = {20, 10, 30};
    int num_pt[3] = {3, 3, 3};
    int ref_axis[3] = {0, 1, 2};

    SetupComp3D(
      2,        //  axis to be compensated
      3,        //  start index in user table
      base,     //  base values
      interval,       //  intervals
      num_pt,         //  number of points (base data included)
      ref_axis        //  reference axes (input)
    );
    EnableComp(2);   //  Enable compensation on axis 2
    Enable(2);   //  Enable axis 2 to activate compensation
}
```

ユーザーが補正を無効にすると、軸の基準位置が現在のフィードバックとしてリセットされます。

```
void main() {
    Disable(2);
    Till(!IsEnabled(2));
    DisableComp(2);   //  Disable compensation on axis 2
}
```

14.2 EnableComp

目的

軸の動的エラー補正を有効にします。

構文

```
int EnableComp(
        int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

軸が有効な場合、この機能は適用されません。

条件

サポートされる最小バージョン	iA Studio 0.23

14.3 DisableComp

目的

軸の動的エラー補正を無効にします。

構文

```
int DisableComp(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

(1) 電流フィードバックとして、軸の基準位置がリセットされます。
(2) 軸が有効な場合、この機能は適用されません。
(3) 動的誤差補正の設定がクリアされます。
　　動的エラー補正を再度有効にするには、設定をリセットする必要があります。

条件

サポートされる最小バージョン	iA Studio 0.23

14.4 SetupComp

目的

軸の 1 次元の動的誤差補正を設定します。

構文

```
int SetupComp(
        int     axis_id,
        int     start_idx,
        double  base_val,
        double  interval,
        int     num_pt,
        int     ref_axis_id
);
```

パラメーター

axis_id [in] Axis index

start_idx [in] ユーザーテーブル内のマップポイントの開始インデックス。

base_val [in] ベース値（マップ入力の最小値）
パラメーター単位: mm または deg

interval [in] 隣接するマップポイント間の一定の間隔
パラメーター単位: mm または deg

num_pt [in] マップポイントの数

ref_axis_id [in] 基準軸のインデックス

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

14.5 SetupComp2D

目的

軸の 2 次元動的エラー補正を設定します。

構文

```
int SetupComp2D(
    int     axis_id,
    int     start_idx,
    double *base_val,
    double *interval,
    int     *num_pt,
    int     *ref_axis_id
);
```

パラメーター

axis_id [in] Axis index

start_idx [in] ユーザーテーブル内のマップポイントの開始インデックス。

base_val [in] 各次元のベース値 (マップ入力の最小値) を含む 2 要素配列へのポインター。
パラメーター単位: mm または deg

interval [in] 隣接するマップポイント間の各次元の一定間隔を含む 2 要素配列へのポインター。
パラメーター単位: mm また deg

num_pt [in] 各次元のマップポイント数を含む 2 要素配列へのポインター。

ref_axis_id [in] 各次元の参照軸のインデックスを含む 2 要素配列へのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

14.6 SetupComp3D

目的

軸の 3 次元動的誤差補正を設定します。

構文

```
int SetupComp3D(
    int     axis_id,
    int     start_idx,
    double *base_val,
    double *interval,
    int     *num_pt,
    int     *ref_axis_id
);
```

パラメーター

axis_id [in] Axis index

start_idx [in] ユーザーテーブル内のマップポイントの開始インデックス

base_val [in] 各次元のベース値(マップ入力の最小値) を含む 3 要素配列へのポインター。
パラメーター単位: mm または deg

interval [in] 隣接するマップポイント間の各次元の一定間隔を含む 3 要素配列へのポインター。
パラメーター単位: mm または deg

num_pt [in] 各次元のマップポイント数を含む 3 要素配列へのポインター。

ref_axis_id [in] 各次元の参照軸のインデックスを含む 3 要素配列へのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.4

14.7 GetCompPos

目的

コントローラーからドライバーに送信された軸の誤差補正値を取得します。

構文

```
double GetCompPos(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸の誤差補正値

単位: mm または deg

条件

サポートされる最小バージョン	iA Studio 1.3

14.8 SetCompAlgType

目的

軸の動的誤差補正の補間方法を設定します。

構文

```
int SetCompAlgType(
    int axis_id,
    int alg_type
);
```

パラメーター

axis_id [in]	Axis index
alg_type [in]	動的エラー補正の補間方法 0: 一次線形補間(初期値) 1: 三次スプライン補間

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

3 次元動的誤差補正は、3 次スプライン補間をサポートしていません。

条件

サポートされる最小バージョン	iA Studio 2.0

15. フィルター機能

15.1 概要

フィルター関数は、プロファイルジェネレーターの位置コマンドを修正するために使用されます。現在、HMPL は、平滑時間、振動抑制フィルター(VSF)、入力整形フィルター (InShape) の 3 種類のフィルターを提供しています。

Smooth time はモーターをスムーズに加速させて滑らかな動きを実現し、VSF と InShape は移動中のモーターの振動(特に機構の負荷が片持ちの場合)を抑制します。「周波数」と「減衰比」をチューニングすることで、制振効果が得られます。

VSF と InShape は同時に使用できませんが、どちらかをスムーズに使用できます。

その上、協調運動になると、Axis InShape 関数は役に立ちません。ユーザーは、振動を抑えるために Group InShape 機能を採用する必要があります。

注: フィルターを使用すると、移動時間が長くなり、デバウンス時間が短くなります。

15.1.1 例

入力整形フィルター(InShape)を設定する方法は、次の HMPL タスクに示されています。

例 1: 単軸

```
void main()
{
    SetAxisInShape(0, 5.5, 0.03, SHAPER_NORMAL);
    //  axis_id, frequency, damping_ratio, shaper_type
    EnableAxisInShape(0);   //  Enable InShape filter of axis 0
}
```

例 2: 軸グループ

```
void main()
{
    int gid = 0;            //  Set gid as group id 0
    int axis1 = 0;          //  Set axis1 as axis 0
    int axis2 = 1;          //  Set axis2 as axis 1
```

```
    Enable(axis1);          //  Enable axis 0
    Enable(axis2);          //  Enable axis 1
    Till(IsEnabled(axis1) && IsEnabled(axis2));

    AddAxisToGrp(gid, axis1);          //  Add axis1 to axis group gid
    AddAxisToGrp(gid, axis2);          //  Add axis2 to axis group gid
    EnableGroup(gid);                  //  Enable axis group gid

    //  Set InShape filter's parameters of axis group gid
    SetGrpInShape(gid, 10, 0.15, 1);
    //  Enable InShape filter of axis group gid
    EnableGrpInShape(gid);
}
```

15.2 EnableAxisVsf

目的

軸の VSF フィルターを有効にします。

構文

```
int EnableAxisVsf(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

モーターが動いているときは、この機能は適用されません。モーターが予期せぬ振動を発生させます。

条件

サポートされる最小バージョン	iA Studio 1.1

15.3 DisableAxisVsf

目的

軸の VSF フィルターを無効にします。

構文

```
int DisableAxisVsf(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

15.4 SetAxisVsf

目的

軸の VSF フィルターのパラメーターを設定します。

構文

```
int SetAxisVsf(
    int axis_id,
    double frequency,
    double damping_ratio
);
```

パラメーター

axis_id [in]	Axis index
frequency [in]	システム周波数 パラメーター単位: Hz 入力範囲: 0.1 ～ 200
damping_ratio [in]	減衰比 入力範囲: 0.7 ～ 1.5 (**1.0** を推奨)

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

モーターが動いているときは、この機能は適用されません。モーターが予期せぬ振動を発生させます。

条件

サポートされる最小バージョン	iA Studio 1.1

15.5 EnableAxisInShape

目的

軸の InShape フィルターを有効にします。

構文

```
int EnableAxisInShape(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

モーターが動いているときは、この機能は適用されません。モーターが予期せぬ振動を発生させます。

条件

サポートされる最小バージョン	iA Studio 1.1

15.6 DisableAxisInShape

目的

軸の InShape フィルターを無効にします。

構文

```
int DisableAxisInShape(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

15.7 SetAxisInShape

目的

軸の InShape フィルターのパラメーターを設定します。

構文

```
int SetAxisInShape(
    int axis_id,
    double frequency,
    double damping_ratio,
    int shaper_type
);
```

パラメーター

axis_id [in]	Axis index
frequency [in]	システム周波数 パラメーター単位: Hz 入力範囲: 1.5 ～ 300
damping_ratio [in]	減衰比 入力範囲: 0.0 ～ 0.3
shaper_type [in]	シェイパータイプ “1”は SHAPER_NORMAL の場合、”0”は SHAPER_ROBUST の場合です。 SHAPER_ROBUST は SHAPER_NORMAL よりも堅牢ですが、 SHAPER_NORMAL は振動を抑えるのに十分な強度があります。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

(1) この機能は、モーターが動いているときは適用できません。モーターが予期せぬ振動を発生させます。

(2) 周波数と減衰比の初期値は、それぞれ 5.5Hz と 0.03 です。

条件

サポートされる最小バージョン	iA Studio 1.1

15.8 EnableGrpInShape

目的

軸グループの InShape フィルターを有効にします。

構文

```
int EnableGrpInShape(
        int group_id
);
```

パラメーター

group_id [in] Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

モーターが動いているときは、この機能は適用されません。モーターが予期せぬ振動を発生させます。

条件

サポートされる最小バージョン	iA Studio 1.1

15.9 DisableGrpInShape

目的

軸グループの InShape フィルターを無効にします。

構文

```
int DisableGrpInShape(
        int group_id
);
```

パラメーター

group_id [in] Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

15.10 SetGrpInShape

目的

軸グループの InShape フィルターのパラメーターを設定します。

構文

```
int SetGrpInShape(
    int group_id,
    double frequency,
    double damping_ratio,
    int shaper_type
);
```

パラメーター

group_id [in] Axis group index

frequency [in] システム周波数
パラメーター単位: Hz
入力範囲: 3.0 ～ 300

damping_ratio [in] 減衰比
入力範囲: 0.0 ～ 0.3

shaper_type [in] シェイパータイプ
“1”は SHAPER_NORMAL の場合、”0”は SHAPER_ROBUST の場合です。
SHAPER_ROBUST は SHAPER_NORMAL よりも堅牢ですが、
SHAPER_NORMAL は振動を抑えるのに十分な強度があります。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

(1) この機能は、モーターが動いているときは適用できません。モーターが予期せぬ振動を発生させます。

(2) この機能は、モーターが動いているときは適用できません。モーターが予期せぬ振動を発生させます。

条件

サポートされる最小バージョン	iA Studio 1.1

16. HMPL タスク機能

16.1 概要

HIMC には、ユーザーがアプリケーションに基づいてモーションプロファイルコマンドを練習するための 64 のビルトイン HMPL タスクがあります。どの HMPL タスクでも、ユーザーは HMPL タスク機能を使用して他の HMPL タスクを開始または停止できます。HMPL タスクが実行されている場合、ユーザーは再実行を要求できません。代わりに、ユーザーはタスクの実行が完了し、タスクが「停止」ステータスになるまで待つ必要があります。ただし、ユーザーは HMPL タスクが現在実行されているかどうかを照会し、それに応じてアプリケーションの複数の HMPL タスクの順序を制御できます。

16.2 StartTask

目的

HMPL タスクの実行を開始します。

構文

```
int StartTask(
        int task_id
);
```

パラメーター

task_id [in]　　HMPL タスク ID

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.22

16.3 StartTaskFunc

目的

HMPL タスクで関数の実行を開始します。

構文

```
int StartTaskFunc(
    int     task_id,
    char *func_name
);
```

パラメーター

task_id [in]　　HMPL タスク ID

func_name [in]　　HMPL タスクに関数名を格納するためのバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 0.22

16.4 StopTask

目的

HMPL タスクの実行を停止します。

構文

```
int StopTask(
        int task_id
);
```

パラメーター

task_id [in]　　HMPL タスク ID

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します

条件

サポートされる最小バージョン	iA Studio 0.22

16.5 StopAllTask

目的

すべての HMPL タスク(呼び出し元を含む)の実行を停止します。

構文

```
void StopAllTask();
```

パラメーター

N/A

戻りの値

N/A

条件

サポートされる最小バージョン	iA Studio 0.23

16.6 IsTaskStop

目的

HMPL タスクの実行が停止されているかどうかを照会します。

構文

```
int IsTaskStop(
        int task_id
);
```

パラメーター

task_id [in]　　HMPL タスク ID

戻りの値

HMPL タスクが TaskStop 状態にある場合は、int 値 TRUE (1)を返します。それ以外の場合は、FALSE (0)を返します。

条件

サポートされる最小バージョン	iA Studio 0.23

(このページはブランクになっています)

17. 変数と関数演算機能

17.1 概要

HIMC は、ドライバーとコントローラーの変数操作機能をユーザーに提供します。ドライバーの変数操作では、ドライバー変数の文字列を指定してその値にアクセスし、MoE 通信を介してデータ交換を行う必要があります。コントローラーの変数操作に関しては、コントローラーの変数 ID リストに基づいて、特定の変数のアドレッシング ID を指定してアクセスする必要があります。ドライバーとコントローラーのパラメーターの定義は、それぞれセクション 17.1.1 と 17.1.2 に記載されています。

注意：
特定の目的に対する要件がない場合は、ユーザーが関連する HMI および機能を使用して関連するシステム変数にアクセスすることをお勧めします。変数操作機能を使用する場合、ユーザーは変数へのアクセスと値の入力の安全性を確保する必要があります。

17.1.1 ドライバー変数一覧

E1 シリーズドライバーの Pt パラメーターの説明については、「E1 シリーズドライバーユーザーマニュアル」の 15 章を参照してください。表 17.1.1.1 に示す Pt パラメーターには、関連する機能のビット定義があるため、パラメーター名は PtXXX.all にする必要があります。その他のパラメーターについては、PtXXX を使用します。

表 17.1.1.1

基本機能設定用パラメーター (Pt0XX)	チューニング用パラメーター (Pt1XX)	位置関連パラメーター(Pt2XX)	トルク関連パラメーター (Pt4XX)	I/O 設定用パラメーター (Pt5XX)	回生抵抗設定パラメーター (Pt6XX)
Pt000.all	Pt10B.all	Pt200.all	Pt408.all	Pt50A.all	Pt70A.all
Pt001.all	Pt139.all	Pt207.all	Pt416.all	Pt50B.all	Pt70B.all
Pt002.all	Pt140.all	Pt22A.all	Pt423.all	Pt50C.all	Pt710.all
Pt006.all	Pt170.all			Pt50D.all	
Pt007.all				Pt50E.all	
Pt008.all				Pt50F.all	
Pt009.all				Pt511.all	
Pt010.all				Pt512.all	
Pt00A.all				Pt513.all	
Pt00B.all				Pt514.all	
Pt00D.all				Pt515.all	
Pt00E.all				Pt516.all	
Pt00F.all				Pt517.all	
				Pt519.all	
				Pt51A.all	

変数操作関数を使用して変数にアクセスする場合、ユーザーは、変数の変更によって生成される安全性の問題を含め、設定された変数の正確性を確認する必要があります。関連する操作要件を除き、可変書き込みエラーによるドライバーの機能への影響を避けるために、Thunder HMI を使用することをお勧めします。

注: 現在、この章で提供されている関数は、D シリーズドライバーのパラメーターをサポートしていません。関連する HMI および機能からパラメーターにアクセスしてください。

17.1.2 コントローラー変数一覧

HIMC は、コントローラー変数のアドレス指定 ID として 32 ビットを取ります。そのタイプは 0x□□□□□□□□で、「0x」は値が 16 進数であることを示します。変数演算機能により、HIMC が提供するシステム変数、軸変数、軸グループ変数にアクセスできます。ID のアドレッシングの規則は、次のように説明されます：

1. アドレッシング ID の 1 番目と 2 番目の値は、「コントローラー変数のカテゴリ」を示します。0x00□□□□□□はシステム変数に属します。0x83□□□□□□は軸変数に属します。0x82□□□□□□は軸グループ変数に属します。

2. アドレッシング ID の 5 ～ 8 番目の値は、「コントローラーのシステム、軸または軸グループ変数のアドレッシング場所」を示します。パラメーターのリストと説明については、表 17.1.2.1 から表 17.1.2.3 を参照してください。

表 17.1.2.1

システム変数		
宛先 ID	変数名	説明
0x0000012c	HCV_ID_fclk	システム実行クロック（250us あたり 1 カウントアップ）
0x0000012e	HCV_ID_timeInMs	システム実行時間(ms)
0x000007d0	HCV_ID_user_table	ユーザーが自由に使える double[512000]配列変数
0x00002328	HCV_ID_ltest0	ユーザーが自由に使える int 変数
0x00002329	HCV_ID_ltest1	ユーザーが自由に使える int 変数
0x0000232a	HCV_ID_ltest2	ユーザーが自由に使える int 変数
0x0000232b	HCV_ID_ltest3	ユーザーが自由に使える int 変数
0x0000232c	HCV_ID_ltest4	ユーザーが自由に使える int 変数
0x0000232d	HCV_ID_ltest5	ユーザーが自由に使える int 変数
0x0000232e	HCV_ID_ltest6	ユーザーが自由に使える int 変数
0x0000232f	HCV_ID_ltest7	ユーザーが自由に使える int 変数
0x00002330	HCV_ID_ltest8	ユーザーが自由に使える int 変数
0x00002331	HCV_ID_ltest9	ユーザーが自由に使える int 変数
0x0000235a	HCV_ID_dtest0	ユーザーが自由に使える double 変数
0x0000235b	HCV_ID_dtest1	ユーザーが自由に使える double 変数
0x0000235c	HCV_ID_dtest2	ユーザーが自由に使える double 変数
0x0000235d	HCV_ID_dtest3	ユーザーが自由に使える double 変数
0x0000235e	HCV_ID_dtest4	ユーザーが自由に使える double 変数
0x0000235f	HCV_ID_dtest5	ユーザーが自由に使える double 変数
0x00002360	HCV_ID_dtest6	ユーザーが自由に使える double 変数
0x00002361	HCV_ID_dtest7	ユーザーが自由に使える double 変数

システム変数		
宛先 ID	変数名	説明
0x00002362	HCV_ID_dtest8	ユーザーが自由に使える double 変数
0x00002363	HCV_ID_dtest9	ユーザーが自由に使える double 変数
0x0000238c	HCV_ID_mtest	ユーザーが自由に使える double[10] 配列変数

表 17.1.2.2

軸変数		
宛先 ID	変数名	説明
0x83□□0015	HCV_ID_motion_type	モーションタイプ
0x83□□0033	HCV_ID_pos_tr	インポジション収束半径
0x83□□0034	HCV_ID_pos_tr_t	インポジション整定時間
0x83□□0065	HCV_ID_sw_RL	ソフトウェア権利制限
0x83□□0066	HCV_ID_sw_LL	ソフトウェア左制限
0x83□□0067	HCV_ID_vel_lim	最大速度制限
0x83□□0068	HCV_ID_acc_lim	最大加速度制限
0x83□□0069	HCV_ID_dec_lim	最大減速度制限
0x83□□0079	HCV_ID_max_pos_err	位置エラー限界
0x83□□007a	HCV_ID_max_comp_lim	位置補正限界
0x83□□00d3	HCV_ID_max_vel	目標速度
0x83□□00d4	HCV_ID_max_acc	目標加速度
0x83□□00d5	HCV_ID_max_dec	目標減速度
0x83□□00d7	HCV_ID_sm_factor	スムーズ時間
0x83□□00db	HCV_ID_vel_scale	速度スケール(0～100)
0x83□□00dd	HCV_ID_p2p_del	P2P モーション待ち時間
0x83□□00de	HCV_ID_p2p_pos1	P2P ポジション 1
0x83□□00df	HCV_ID_p2p_pos2	P2P ポジション 2
0x83□□00e0	HCV_ID_p2p_repeat	P2P モーションを繰り返す
0x83□□00e1	HCV_ID_rlt_dist	相対移動距離
0x83□□00e2	HCV_ID_en_motionManager	Motion Manager の動作軸選択
0x83□□00e3	HCV_ID_acc_time	加速時間
0x83□□00e4	HCV_ID_dec_time	減速時間
0x83□□00e9	HCV_ID_map_io_type	エラー補正タイプ
0x83□□0117	HCV_ID_rollover_turns	ロールオーバーターン
0x83□□0119	HCV_ID_rollover_val	ロールオーバー値
0x83□□012d	HCV_ID_servo_type	サーボ制御モード（CSP：0、PP：5）
0x83□□0193	HCV_ID_gant_pair	ガントリー構成のガントリーペア ID
0x83□□01f7	HCV_ID_en_delay	軸有効化のタイムアウト

軸変数		
宛先 ID	変数名	説明
0x83□□01ff	HCV_ID_fb_ratio_pos	ドライバーの位置分解能; 長さの単位（分母）
0x83□□0200	HCV_ID_fb_ratio_cnt	ドライバーの位置分解能; 単位: カウント（分子）
0x83□□0263	HCV_ID_last_err	軸エラーコード
0x83□□03c2	HCV_ID_gear_ratio	ギア比

注：記号□□は 16 進形式の軸 ID になります。たとえば、01 は Axis index.01 を表します。0f は Axis index.15 を表します。

表 17.1.2.3

軸グループ変数		
宛先 ID	変数名	説明
0x82□□0002	HCV_ID_grp_num_axis	軸グループの軸数
0x82□□00c9	HCV_ID_grp_lin_vel_lim	軸グループの速度制限
0x82□□00ca	HCV_ID_grp_lin_acc_lim	軸グループの加速度制限
0x82□□00cb	HCV_ID_grp_lin_dec_lim	軸グループの減速度制限
0x82□□00d0	HCV_ID_grp_lin_vel	軸グループの目標速度
0x82□□00d1	HCV_ID_grp_lin_acc	軸グループの目標加速度
0x82□□00d2	HCV_ID_grp_lin_dec	軸グループの目標減速度
0x82□□00d3	HCV_ID_grp_lin_sf	軸グループの平滑化時間
0x82□□00f0	HCV_ID_grp_lin_acc_time	軸グループの目標加速時間
0x82□□00f1	HCV_ID_grp_lin_dec_time	軸グループの目標減速時間
0x82□□00e4	HCV_ID_grp_coord_sys	軸グループの座標系
0x82□□00e5	HCV_ID_grp_buffer_mode	軸グループのバッファーモード
0x82□□00e6	HCV_ID_grp_trans_mode	軸グループの遷移モード
0x82□□00e7	HCV_ID_grp_trans_vel	軸グループの移行速度
0x82□□00e8	HCV_ID_grp_trans_dis	軸グループの遷移距離
0x82□□00eb	HCV_ID_grp_vel_scale	軸グループの速度スケール(0~100)
0x82□□00da	HCV_ID_grp_shaper_fr	軸グループの InShape フィルターの頻度
0x82□□00db	HCV_ID_grp_shaper_xi	軸グループの InShape フィルターの減衰比
0x82□□038f	HCV_ID_grp_last_err	軸群エラーコード

注：記号□□は 16 進形式の軸グループ ID になります。たとえば、01 は軸グループインデックス 01 を表します。0f は、軸グループインデックス 15 を表します。

17.2 ドライバー可変運転

17.2.1 ReadSDO

目的

SDO を介してスレーブのオブジェクト値を読み取ります。

構文

```
double ReadSDO(
    int slv_id,
    int obj_index,
    int obj_subindex,
    int obj_length
);
```

パラメーター

slv_id	[in]	スレーブ index
obj_index	[in]	スレーブ オブジェクトのインデックス
obj_subindex	[in]	スレーブ オブジェクトのサブインデックス
obj_length	[in]	スレーブオブジェクトのバイト長

戻りの値

関数が成功した場合は double 型の SDO オブジェクト値を返し、関数が失敗した場合は double 型の -1 値を返します。

例

```
void main()
{
    double value= ReadSDO(0, 0x6041, 0 , 2);
    Print("value = %f", value);
}
```

条件

サポートされる最小バージョン	iA Studio 3.0

17.2.2 ReadSDOEx

目的

SDO を介してスレーブのオブジェクト値を読み取り、バッファへのポインタにデータを格納します。

構文

```
int ReadSDOEx(
    int slv_id,
    int obj_index,
    int obj_subindex,
    int obj_length,
    double* obj_value
);
```

パラメーター

slv_id [in]	スレーブ index
obj_index [in]	スレーブ オブジェクトの index
obj_subindex [out]	スレーブ オブジェクトのサブインデックス
obj_length [out]	スレーブオブジェクトのバイト長
obj_value [out]	返された SDO オブジェクト値を格納するバッファへのポインター

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は -1 値を返します。

例

```
void main()
{
    double value;
    int rtn;
    rtn = ReadSDOEx(0, 0x6041, 0 , 2, &value);
    Print("return = %d, value = %f", rtn, value);
}
```

条件

サポートされる最小バージョン	iA Studio 3.0

17.2.3 WriteSDO

目的

SDO を介してスレーブのオブジェクト値を書き込みます。

構文

```
Int WriteSDO(
    int slv_id,
    int obj_index,
    int obj_subindex,
    int obj_length,
    double obj_value
);
```

パラメーター

slv_id [in]	スレーブ index
obj_index [in]	スレーブ オブジェクトの index
obj_subindex [in]	スレーブ オブジェクトのサブインデックス
obj_length [in]	スレーブオブジェクトのバイト長
obj_value [in]	書き込まれる SDO オブジェクト値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は -1 値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

17.2.4 ReadPDO

目的

PDO を介してスレーブの PDO オブジェクト値を読み取ります。

構文

```
double ReadPDO(
    int slv_id,
    int obj_index,
    int obj_subindex,
);
```

パラメーター

slv_id [in]	スレーブ index
obj_index [in]	スレーブ オブジェクトの index
obj_subindex [in]	スレーブ オブジェクトのサブインデックス
obj_value [in]	返された PDO オブジェクト値を格納するバッファへのポインター

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は -1 値を返します。

例

```
void main()
{
    double value;
    int rtn;
    rtn = ReadPDO(0, 0x6041, 0, &value);
    Print("value = %f", value);
}
```

条件

サポートされる最小バージョン	iA Studio 3.0

17.2.5 ReadPDOEx

目的

PDO を介してスレーブの PDO オブジェクト値を読み取ります。

構文

```
int ReadPDOEx(
    int slv_id,
    int obj_index,
    int obj_subindex,
    double* obj_value
);
```

パラメーター

slv_id [in]	スレーブ index
obj_index [in]	スレーブ オブジェクトの index
obj_subindex [in]	スレーブ オブジェクトのサブインデックス
obj_value [in]	返された PDO オブジェクト値を格納するバッファへのポインター

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は -1 値を返します。

例

```
void main()
{
    double value;
    value = ReadPDOEx(0, 0x6041, 0, &value);
    Print("value = %f", value);
}
```

条件

サポートされる最小バージョン	iA Studio 3.0

17.2.6 WritePDO

目的

PDO を介してスレーブの PDO オブジェクトを書き込みます。

構文

```
int WritePDO(
    int slv_id,
    int obj_index,
    int obj_subindex,
    double obj_value
);
```

パラメーター

slv_id [in]	スレーブ index
obj_index [in]	スレーブ オブジェクトの index
obj_subindex [in]	スレーブ オブジェクトのサブインデックス
obj_value [in]	書き込まれる PDO オブジェクト値

戻りの値

他のソースが同時にオブジェクトに書き込むと、上書きされる危険性があります。

条件

サポートされる最小バージョン	iA Studio 3.0

17.2.7 ForceWritePDO

目的

PDO を介してスレーブの PDO オブジェクトを強制的に書き込みます。

構文

```
int ForceWritePDO(
    int slv_id,
    int obj_index,
    int obj_subindex,
    double obj_value
);
```

パラメーター

slv_id [in]	スレーブ index
obj_index [in]	スレーブ オブジェクトの index
obj_subindex [in]	スレーブ オブジェクトのサブインデックス
obj_value [in]	書き込まれる PDO オブジェクト値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は -1 値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

17.2.8 ReleasePDO

目的

強制的に書き込まれた PDO オブジェクトを解放し、ForceWritePDO で使用します。

構文

```
int ReleasePDO(
    int slv_id,
    int obj_index,
    int obj_subindex
);
```

パラメーター

slv_id [in]	スレーブ index
obj_index [in]	スレーブ オブジェクトの index
obj_subindex [in]	スレーブ オブジェクトのサブインデックス

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は -1 値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

17.3 コントローラー変数操作

17.3.1 GetConfigVar

目的

コントローラーの変数値を取得します。

構文

```
double GetConfigVar(
        int hcv_id,
        int *result
);
```

パラメーター

hcv_id [in]	HIMC コントローラー変数 ID 定義については、セクション 17.1.2 を参照してください。
result [out]	関数が失敗した場合は-1 を返します。

戻りの値

変数の値

例

```
void main()
{
        int result = 0;
        double SW_RL = GetConfigVar(0x83000065, &result);
        Print("SW_RL = %f", SW_RL);
}
```

条件

サポートされる最小バージョン	iA Studio 1.1

17.3.2 SetConfigVar

目的

コントローラーの変数値を設定します

構文

```
int SetConfigVar(
    int hcv_id,
    double value
);
```

パラメーター

hcv_id [in]	HIMC コントローラー変数 ID 定義については、セクション 17.1.2 を参照してください。
value [in]	変数の新しい値

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

例

```
void main()
{
    int result = 0;
    SetConfigVar(0x83000065, 0.0);    //    0x83000065 is axis 0 software right limit
    Print("SW_RL = %f", GetConfigVar(0x83000065, &result));
}
```

条件

サポートされる最小バージョン	iA Studio 1.1

18. エラー機能

18.1 概要

HIMC は、関連するエラーメッセージを表す 32 ビットエラーコードを提供します。この章で提供される機能を使用して、ユーザーはシステム、軸、および軸グループのエラーコードを取得またはクリアできます。(各カテゴリのエラーコード、名前、および説明は、セクション 18.1.1 ～ 18.1.3 に記載されています。) エラーコードのタイプは 0x□□□□□□□□で、「0x」は値が 16 進数であることを示します。その規則は、次のように説明されるコントローラー変数のアドレス指定 ID の規則と同じです：

3. エラーコードの 1 番目と 2 番目の値は、「コントローラー変数のカテゴリ」を示します。0x00□□□□□□はシステム変数に属します。0x83□□□□□□は軸変数に属します。0x82□□□□□□は軸グループ変数に属します。

4. エラーコードの 3 番目と 4 番目の値は、「軸 ID または軸グループ ID」を示します。たとえば、軸変数 0x8302□□□□は Axis index.02 を格納する変数であり、軸グループ変数 0x8201□□□□は軸グループインデックス 01 を格納する変数です。

5. エラーコードの 5 ～ 8 番目の値は「変数 ID」を示します。詳細については、セクション 18.1.1 ～ 18.1.3 を参照してください。

注: 関数の戻り値は 10 進数であるため、正しいエラーコードを取得するには、ユーザーが自分でそれを 16 進数に変換する必要があります。

18.1.1 システムエラーメッセージ

表 18.1.1.1

システムエラーコード		
エラーコード	エラー名	説明
0x00000001	eERR_HCV_ID_NOT_FOUND	変数 ID が見つかりませんでした
0x00000002	eERR_DATA_EXCEEDED	要求されたデータは範囲外です
0x00000003	eERR_HCV_IS_READ_ONLY	読み取り専用パラメーター
0x00000004	eERR_HCV_VALUE_OUT_OF_RANGE	入力値が範囲外です
0x00000064	eERR_EMERGENCY_STOP	非常停止が作動しました。すべての軸を無効にし、すべてのタスクを停止します。
0x000000ff	eERR_MOE_NOT_READY	MoE は準備ができていません。
0x00000100	eERR_MAIL_BOX_BUSY	コントローラーとスレーブ間のメールボックスがビジーです
0x00000101	eERR_VAR_NOT_IN_SLV_DB	スレーブ変数が見つかりませんでした
0x00000102	eERR_VAR_NOT_REGYET	スレーブ変数を読み取ることができません
0x00000103	eERR_READ_VAR_NO_RECV	スレーブからの応答はありませんでした
0x00000104	eERR_PREV_SLV_CMD_NOT_FIN	スレーブへの前のコマンドが終了していません
0x00000105	eERR_SLV_ID_INVALID	スレーブ ID が無効です
0x00000106	eERR_PDO_NUM_EXCEED	PDO の数が範囲外です
0x00000107	eERR_NOT_VALID_TASKID	タスク ID が無効です
0x00000108	eERR_TASK_IS_RUNNING	タスクはすでに実行中です
0x00000109	eERR_FUNC_NOT_IN_TASK	関数がタスクに見つかりませんでした
0x0000010a	eERR_TASK_EMPTY	タスクは空です
0x0000010b	eERR_TASK_NOT_RUNNING	タスクは実行されていません
0x0000012c	eERR_NIC_INIT_TOUT	mega-ulink のネットワークポートの準備ができていません。
0x0000012d	eERR_HARDWARE_MISMATCH	ハードウェアが認識されていません
0x0000012e	eERR_SLAVE_NUM_MISMATCH	スレーブの数は構成とは異なります
0x0000012f	eERR_INVALID_PDO	PDO が無効です
0x00000130	eERR_INVALID_MCK_CNFG	モーションカーネルの構成が無効です
0x00000136	eERR_MOE_SEND_FAIL	mega-ulink パケットの送信に失敗しました
0x00000137	eERR_MOE_RECV_FAIL	mega-ulink パケットの受信に失敗しました
0x00000138	eERR_HIMC_LOAD_CONFIG_FAIL	SSD からの構成のロードに失敗しました。もう一度保存してください
0x00000139	eERR_HIMC_SAVE_CONFIG_FAIL	HIMC への構成の保存に失敗しました。 もう一度保存してください。
0x0000013a	eERR_HIMC_SAVE_CONFIG_COPY_FAIL	HIMC への構成の保存に失敗しました。ファイルを SAVE フォルダに保存できません。
0x0000013b	eERR_HIMC_SAVE_UPDATE_PRM_TIMEOUT	HIMC への構成の保存に失敗しました。Prm 値の更新タイムアウト。
0x000001f4	eERR_ISR_NOT_STABLE	割り込みの周期は安定していません
0x000001f5	eERR_MCK_OVERLOAD	モーションカーネルが過負荷になっています。
0x000001f6	eERR_ISR_OVERLOAD	CPU が過負荷になっています。
0x000001f7	eERR_MOE_ISR_NOT_STABLE	MoE では、割り込みの周期が安定しません。

システムエラーコード		
エラーコード	エラー名	説明
0x000003e8	eERR_PP_MODE_NOT_SUPPORTED	この機能は PP モードではサポートされていません。
0x00001388	eERR_HMPL_INVALID_ARG	引数は HMPL では無効です
0x00001389	eERR_HMPL_INVALID_PTR	ポインターは HMPL では無効です
0x0000138a	eERR_HMPL_STACK_OVERFLOW	HMPL のスタックオーバーフロー
0x0000138b	eERR_HMPL_ILLEGAL_MEM_OP	HMPL ではメモリの操作は違法です
0x0000138c	eERR_HMPL_MOTION_NOT_READY	モーション関数は同期状態で呼び出す必要があります。
0x0000138d	eERR_HMPL_STR_TOO_LONG	文字列の長さが範囲外です
0x0000138e	eERR_HMPL_INVALID_STR_FORMAT	文字列形式が無効です。
0x0000138f	eERR_HMPL_ARG_OUT_OF_RANGE	引数が範囲外です
0x00001392	eERR_HMPL_ASCII_AGENT_RUNNING	ASCII エージェントはすでに実行中です
0x0000139c	eERR_HMPL_CANNOT_RUN_IN_DEBUG	関数はデバッグモードでは実行できません
0x000013a6	eERR_HMPL_TOO_MANY_BRK_POINT	タスクにブレークポイントが多すぎます。
0x000013ec	eERR_HMPL_MUTEX_LOCK_TWICE	同じタスクで同じミューテックスを 2 回ロックすることはできません。
0x00001450	eERR_HMPL_INVALID_SYS_TIME_MEMORY	バッファーが小さすぎます。最小サイズは 30 バイトである必要があります。
0x00001451	eERR_HMPL_NOT_SUPPORTED	この HMPL 関数は、このプラットフォームではサポートされていません。
0x00001452	eERR_HMPL_CLIENT_NOT_CONNECTED	クライアントが切断されているため送信できません。
0x0000176f	eERR_HMPL_INTERNAL_ERROR	HMPL 内部エラー
0x00001770	eERR_HMPL_EXEC_FAILED	HMPL 関数の実行に失敗しました。
0x00001771	eERR_HMPL_ASM_LOAD_FAILED	HMPL のコンパイルに失敗しました。アセンブリファイルが空であるか、生成されていません
0x00001772	eERR_HMPL_STARTTASK_TIMEOUT	HMPL StartTask 関数のタイムアウト
0x00001773	eERR_HMPL_STOPTASK_TIMEOUT	HMPL StopTask 関数のタイムアウト
0x000017d4	eERR_ASCII_CONNECT_TIMEOUT	ASCII クライアント接続タイムアウト
0x000017d5	eERR_ASCII_CONNECT_FAILED	ASCII クライアント接続に失敗しました。 IP とポートを確認してください。
0x000017d6	eERR_ASCII_MULTI_CONNECTING	同時に接続する複数の ASCII クライアント
0x000017d7	eERR_ASCII_MULTI_DISCONNECTING	複数の ASCII クライアントが同時に切断します
0x000017d8	eERR_ASCII_DISCONNECT_TIMEOUT	ASCII クライアント切断タイムアウト
0x000017de	eERR_ASCII_RECV_TIMEOUT	ASCII クライアント受信タイムアウト。後でもう一度やり直してください。
0x000017df	eERR_ASCII_RECV_FAIL	ASCII クライアントの受信に失敗しました。接続がまだ生きているかどうかを確認してください。
0x000017e0	eERR_ASCII_MULTI_RECVING	複数の ASCII クライアントが同時に受信しています。
0x000017e8	eERR_ASCII_SEND_TIMEOUT	ASCII クライアント送信タイムアウト。後でもう一度やり直してください。
0x000017e9	eERR_ASCII_SEND_FAIL	ASCII クライアントの送信に失敗しました。接続がまだ生きているかどうかを確認してください。

システムエラーコード		
エラーコード	エラー名	説明
0x000017ea	eERR_ASCII_MULTI_SENDING	同時に送信する複数の ASCII クライアント。
0x00001838	eERR_MODBUS_CONNECT_TIMEOUT	Modbus クライアント接続タイムアウト。
0x00001839	eERR_MODBUS_CONNECT_FAILED	Modbus クライアント接続に失敗しました。ip を確認してください。
0x0000183a	eERR_MODBUS_MULTI_CONNECTING	同時に接続する複数の Modbus クライアント
0x0000183b	eERR_MODBUS_MULTI_DISCONNECTING	複数の Modbus クライアントが同時に切断されます。
0x0000183c	eERR_MODBUS_DISCONNECT_TIMEOUT	同時に送信する複数の ASCII クライアント。
0x0000183d	eERR_MODBUS_DATALENGTH_ERR	Modbus クライアントの読み書きデータ数が制限を超えています。
0x0000183e	eERR_MODBUS_SOCKET_BUSY	Modbus クライアントは、同時に 2 つ以上のコマンドを処理します。
0x0000183f	eERR_MODBUS_JOB_TIMEOUT	Modbus クライアントジョブの実行タイムアウト。後でもう一度やり直してください
0x00001840	eERR_MODBUS_JOB_FAIL	Modbus クライアントジョブの実行に失敗しました。接続がまだ生きているかどうかを確認してください。

18.1.2 軸エラーメッセージ

次のエラーコードは、軸のエラーまたは無効な操作が原因で表示されます。記号□□は 16 進形式の軸 ID になります。たとえば、01 は Axis index.01 を表します。0f は Axis index.15 を表します。

表 18.1.2.1

軸エラーコード		
エラーコード	エラー名	説明
0x83□□000a	eERR_AXIS_CMD_UNKOWN	コマンド名は不明です
0x83□□001e	eERR_AXIS_CMD_QUEUE_FULL	軸コマンドキューがいっぱいです
0x83□□0064	eERR_AXIS_CMD_INVALID_STATE	軸は、現在のモーション状態ではコマンドを実行できません。
0x83□□006e	eERR_AXIS_CMD_INVALID_ENABLED	このコマンドは、有効になっている間は使用できません。
0x83□□0078	eERR_AXIS_CMD_INVALID_DISABLED	このコマンドは、無効になっている間は使用できません
0x83□□0082	eERR_AXIS_CMD_INVALID_MOVING	軸は移動中にコマンドを実行できません
0x83□□008c	eERR_AXIS_CMD_INVALID_STOPPING	軸が停止している場合、コマンドは無効です
0x83□□0096	eERR_AXIS_CMD_INVALID_ERROR_STATE	軸が ErrorStop 状態の場合、コマンドは無効です
0x83□□00a0	eERR_AXIS_CMD_INVALID_IN_SYNC	軸が同期動作状態の場合、コマンドは無効です
0x83□□00aa	eERR_AXIS_CMD_INVALID_GEAR_MASTER	軸がギアマスタ軸の場合、コマンドは無効です。
0x83□□00b4	eERR_AXIS_CMD_INVALID_PP_MODE	軸が PP モードの場合、コマンドは無効です。
0x83□□00c8	eERR_AXIS_CMD_INVALID_INPUTSHAPING_ENABLED	位置指令整形機能が有効な場合、軸は指令を実行できません。
0x83□□00d2	eERR_AXIS_CMD_INVALID_COMP_ENABLED	動的補正が有効になっている場合、軸はコマンドを実行できません。
0x83□□00dc	eERR_AXIS_CMD_INVALID_GANTRY_MODE	軸は、ガントリーモードでコマンドを実行できません。
0x83□□00e6	eERR_AXIS_CMD_INVALID_GROUPED	軸が軸グループにある場合、このコマンドは使用できません。
0x83□□012c	eERR_AXIS_CMD_INVALID_parameter	軸コマンドのパラメーターが不正です
0x83□□0136	eERR_AXIS_CMD_INVALID_POS	軸目標位置が許容範囲外です
0x83□□0140	eERR_AXIS_CMD_INVALID_VEL	軸速度設定が許容範囲外です。
0x83□□014a	eERR_AXIS_CMD_INVALID_ACC	軸加速度の設定が許容範囲外です。
0x83□□0154	eERR_AXIS_CMD_INVALID_DEC	軸減速度の設定が許容範囲外です。
0x83□□015e	eERR_AXIS_CMD_INVALID_JERK	軸加々速度の設定が許容範囲外です。
0x83□□0168	eERR_AXIS_CMD_INVALID_SM_TIME	軸スムーズ時間設定が許容範囲外です。
0x83□□0172	eERR_AXIS_CMD_INVALID_KILL_DEC	軸キル減速度の設定が許容範囲外です。
0x83□□017c	eERR_AXIS_CMD_INVALID_VEL_SCALE	軸速度スケールの設定が許容範囲外です
0x83□□0190	eERR_AXIS_COMP_NOT_CNFG	軸の動的補正設定が正しく構成されていません。
0x83□□01c2	eERR_AXIS_CMD_INVALID_MASTER_SLAVE_CONNECTION	主従関係の設定が無効です
0x83□□01cc	eERR_AXIS_CMD_INVALID_SLAVE_ID	スレーブ ID の設定が無効です
0x83□□01d6	eERR_AXIS_CMD_INVALID_GEAR_RATIO	スレーブ軸のギア比設定が許容範囲外です

軸エラーコード		
エラーコード	エラー名	説明
0x83□□01f4	eERR_AXIS_CMD_INVALID_ROLLOVER_POS	軸のロールオーバー位置が無効です。正の値にする必要があります。
0x83□□03f2	eERR_AXIS_DRIVE_FAULT	ドライブが障害を報告しました。ドライバー内の対応するエラーメッセージを確認してください
0x83□□03fc	eERR_AXIS_DRIVE_ABNORMAL_DISABLE	ドライバーが異常に無効になっています
0x83□□0406	eERR_AXIS_DRIVE_ENABLE_TOUT	ドライバーを有効にするのに時間がかかりすぎました。
0x83□□0410	eERR_AXIS_DRIVE_CLEAR_ERROR_TOUT	ドライバーエラーのクリアに時間がかかりすぎました。
0x83□□041a	eERR_AXIS_DRIVE_DISABLE_TOUT	ドライバーを無効にするのに時間がかかりすぎました。
0x83□□0456	eERR_AXIS_VEL_LIMIT	基準速度が速度制限を超えました
0x83□□0460	eERR_AXIS_ACC_LIMIT	基準加速度が加速度制限を超えています
0x83□□046a	eERR_AXIS_CURR_LIMIT	現在のコマンドが現在の制限を超えました
0x83□□0474	eERR_AXIS_DAMPINGRATIO_LIMIT	軸の減衰比設定が許容範囲外です
0x83□□047e	eERR_AXIS_FREQUENCY_LIMIT	軸の周波数設定が許容範囲外です
0x83□□07da	eERR_AXIS_SWRL	軸基準位置が右ソフトリミットに達しました
0x83□□07e4	eERR_AXIS_SWLL	軸基準位置が左ソフトリミットに達しました
0x83□□07ee	eERR_AXIS_HWRL	軸右ハードウェアリミット信号がトリガーされました。
0x83□□07f8	eERR_AXIS_HWLL	軸左ハードウェアリミット信号がトリガーされました。
0x83□□0802	eERR_AXIS_COMP_LIMIT	軸補正位置が最大補正限界を超えました。
0x83□□083e	eERR_AXIS_PERR	軸位置偏差が保護限界を超えました。 モーターの動きに機械的な干渉がないかどうかを最初に確認してください。
0x83□□0848	eERR_AXIS_VERR	軸速度エラーが保護限界を超えました。 モーターの動きに機械的な干渉がないかどうかを最初に確認してください。
0x83□□08a2	eERR_AXIS_PVT_MOTION_VEL_LIMIT	軸 PVT モーションの速度が保護限界を超えました。指定されたパラメーターが有効かどうかを最初に確認してください。
0x83□□08ac	eERR_AXIS_PVT_MOTION_ACC_LIMIT	軸 PVT モーションの加速度が保護限界を超えました。指定されたパラメーターが有効かどうかを最初に確認してください。
0x83□□08b6	eERR_AXIS_PVT_MOTION_INVALID_TIME	軸 PVT モーションのタイムシーケンスが無効です。指定されたパラメーターが有効かどうかを最初に確認してください。
0x83□□0bb8	eERR_AXIS_CTRL_ERR	軸内部制御エラー
0x83□□0fa0	eERR_AXIS_CMD_GEAR_DISABLED	ギアが無効になっている間は、ギアコマンドを使用できません。

18.1.3 グループのエラーメッセージ

以下のエラーコードは、軸グループでのエラーまたは無効な操作により表示されます。記号□□は 16 進形式の軸グループ ID になります。たとえば、01 は軸グループインデックス 01 を表します。0f は、軸グループインデックス 15 を表します。

表 18.1.3.1

軸グループのエラーコード		
エラーコード	エラー名	説明
0x82□□000a	eERR_CRD_CMD_UNKNOWN	軸グループコマンドが不明です
0x82□□0028	eERR_CRD_CMD_AXIS_DUPLICATED	軸は既にグループにあるため、追加できませんでした
0x82□□0032	eERR_CRD_CMD_GRP_SIZE_EMPTY	軸グループが空です
0x82□□003c	eERR_CRD_CMD_GRP_SIZE_FULL	軸グループがいっぱいで、これ以上軸を保持できません
0x82□□0046	eERR_CRD_CMD_INVALID_MOVING	軸群移動中はコマンド無効です
0x82□□0050	eERR_CRD_CMD_INVALID_DISABLED	軸グループが無効の場合、コマンドは無効です
0x82□□005a	eERR_CRD_CMD_INVALID_INPUTSHAPING_parameter_INCOMPLETE	軸グループインシェイプ機能のパラメーターが不完全です。
0x82□□001e	eERR_CRD_CMD_INVALID_KIN_SETTING	キネマティクスタイプの設定が無効です
0x82□□001f	eERR_CRD_CMD_INVALID_SPECIFIC_KIN	軸グループが特定のキネマティクスタイプにある場合、このコマンドは無効です。
0x82□□006e	eERR_CRD_CMD_INVALID_STATE	軸グループは、現在のモーション状態ではコマンドを実行できません。
0x82□□0078	eERR_CRD_CMD_QUEUE_FULL	最後のコマンドが完了するまでお待ちください。
0x82□□00d2	eERR_CRD_CMD_INVALID_POS	軸グループの目標位置または姿勢が許容範囲外です。
0x82□□00dc	eERR_CRD_CMD_INVALID_LIN_VEL	軸群の線速度設定が許容範囲外です
0x82□□00e6	eERR_CRD_CMD_INVALID_LIN_ACC	軸グループの直線加速度設定が許容範囲外です
0x82□□00f0	eERR_CRD_CMD_INVALID_LIN_DEC	軸グループの直線減速度の設定が許容範囲外です。
0x82□□00fa	eERR_CRD_CMD_INVALID_LIN_JERK	軸グループの直線加減速設定が許容範囲外です
0x82□□0104	eERR_CRD_CMD_INVALID_LIN_SM_TIME	軸グループの直線平滑時間設定が許容範囲外です。
0x82□□010e	eERR_CRD_CMD_INVALID_DAMPINGRATIO	軸群の減衰比設定が許容範囲外です
0x82□□0118	eERR_CRD_CMD_INVALID_FREQUENCY	軸グループの周波数設定が許容範囲外です
0x82□□0140	eERR_CRD_CMD_INVALID_ANG_VEL	軸グループの角速度設定が許容範囲外です
0x82□□014a	eERR_CRD_CMD_INVALID_ANG_ACC	軸グループの角加速度設定が許容範囲外です
0x82□□0154	eERR_CRD_CMD_INVALID_ANG_DEC	軸群の減速度設定が許容範囲外です
0x82□□015e	eERR_CRD_CMD_INVALID_ANG_JERK	軸群の加々速度設定が許容範囲外です
0x82□□0168	eERR_CRD_CMD_INVALID_ANG_SM_TIME	軸グループの角度平滑時間設定が許容範囲外です
0x82□□0190	eERR_CRD_CMD_INVALID_VEL_SCALE	軸グループの速度スケールが許容範囲外です
0x82□□019a	eERR_CRD_CMD_INVALID_TRANS_VEL	軸グループの移行速度が無効です
0x82□□01a4	eERR_CRD_CMD_INVALID_TRANS_DIS	軸グループの遷移距離が無効です

軸グループのエラーコード		
エラーコード	エラー名	説明
0x82□□01b8	eERR_CRD_CMD_TRANS_MODE_UNKNOWN	パス遷移モード名が不明です
0x82□□01c2	eERR_CRD_CMD_COORD_SYS_UNKNOWN	座標系が不明です
0x82□□01cc	eERR_CRD_CMD_BLEND_MODE_UNKNOWN	パスブレンドモード名が不明です
0x82□□01fe	eERR_CRD_CMD_LIN_INVALID_PARAM	パラメーターは、直線経路計画では無効です
0x82□□0262	eERR_CRD_CMD_CIRC_INVALID_PARAM	パラメーターは円形経路計画では無効です
0x82□□026c	eERR_CRD_CMD_CIRC_INVALID_CENTER	円形パスの中心位置が始点/終点に近すぎます
0x82□□0276	eERR_CRD_CMD_CIRC_ANGLE_SMALL	円形パスの中心角が小さすぎます
0x82□□0280	eERR_CRD_CMD_CIRC_INVALID_RADIUS	円形パスの半径が無効です
0x82□□028a	eERR_CRD_CMD_CIRC_INVALID_COORD	円形パスの座標系が無効です
0x82□□02c6	eERR_CRD_CMD_BEZIER_INVALID_PARAM	パラメーターは、ベジエ曲線パス計画では無効です。
0x82□□02d0	eERR_CRD_CMD_BSPLINE_INVALID_PARAM	パラメーターは BSpline 曲線パス計画では無効です。
0x82□□02da	eERR_CRD_CMD_CURVE_INVALID_STARTPOS	開始位置は曲線パス計画では無効です
0x82□□03f2	eERR_CRD_AXIS_ABNORMALLY_DISABLED	軸グループ内の 1 つまたは複数の軸が異常に無効になっています。
0x82□□03fc	eERR_CRD_AXIS_SWL	軸グループの軸の 1 つがソフトウェアリミットに達しています。

18.2 GetSystemLastErr

目的

コントローラーの最新のエラーコードを取得します。

構文

```
int GetSystemLastErr();
```

パラメーター

N/A

戻りの値

コントローラーの最新のエラーコード
定義については、セクション 18.1.1 を参照してください。

条件

サポートされる最小バージョン	iA Studio 1.1

18.3 GetAxisLastErr

目的

軸の最新のエラーコードを取得します

構文

```
int GetAxisLastErr(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

軸の最新のエラーコード

定義については、セクション 18.1.2 を参照してください。

条件

サポートされる最小バージョン	iA Studio 1.1

18.4 ClearAxisLastErr

目的

軸の最新のエラーコードをクリアします

構文

```
int ClearAxisLastErr(
        int axis_id
);
```

パラメーター

axis_id [in]　　Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

18.5 GetGrpLastErr

目的

軸グループの最新のエラーコードを取得します

構文

```
int GetGrpLastErr(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

軸グループの最新のエラーコード

定義については、セクション 18.1.3 を参照してください。

条件

サポートされる最小バージョン	iA Studio 1.1

18.6 ClearGrpLastErr

目的

軸グループの最新のエラーコードをクリアします。

構文

```
int ClearGrpLastErr(
        int group_id
);
```

パラメーター

group_id [in]　　Axis group index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 1.1

18.7 GetDriveErr

目的
ドライバーのエラーコードを取得します。

構文

```
int GetDriveErr(
    int axis_id
);
```

パラメーター
axis_id [in] Axis index

戻りの値
ドライバーのエラーコードを返します。

備考
この機能を使用する場合、ユーザーはオブジェクト 0x603F(エラー コード)を PDO として設定する必要があります。

条件

サポートされる最小バージョン	iA Studio 3.0

（このページはブランクになっています）

19. マルコの定義と機能

19.1 _TASKID_

目的

現在の HMPL タスク ID を照会します

例

```
#if _TASKID_ == 0
int global_var = 0;   //   only be compiled if current task ID is 0
#endif

void test(){

    for (;;) {
      if (HIMC_GPI(1)) {
        StopTask( _TASKID_ );   //   Stop current task
      }
    }
}

void main() {
    test();
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

19.2 _AUTORUN_

目的

起動時のタスクを自動化する

例

```
_AUTORUN_ void main() {
    Till(IsSystemOper());
    //  Do something
}
```

条件

サポートされる最小バージョン	iA Studio 0.22

19.3 Till

目的

特定の条件が満たされるまで HMPL タスクの実行を停止します

構文

```
Till(
        condition
);
```

パラメーター

condition [in] タイプ: **int**
条件評価の結果 → true (非ゼロ)または false (0)

備考

この関数を呼び出す際、HMPL アプリケーションの無効化(特定の条件が満たされない場合)に起因する HIMC の異常な動作は、ユーザーの責任で行ってください。

例

```
void main() {
        Till(IsEnabled(0) && IsEnabled(1));

        //    Do something
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

19.4 HIMC_GPI

目的

コントローラーの汎用入力の状態を照会します

構文

```
HIMC_GPI(
    int gpi_idx
);
```

パラメーター

gpi_idx [in]　　汎用入力インデックス

例

```
void main() {
    //  Get the state of the specific general input
    if (HIMC_GPI(4) && HIMC_GPI(6)) {
    //  if both HIMC_GPI(4) and HIMC_GPI(6) are at the “on” state
      //  Do something
    }
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

19.5 HIMC_GPO

目的

コントローラーの汎用出力の状態を問い合わせます。

構文

```
HIMC_GPO(
    int gpo_idx
);
```

パラメーター

gpo_idx [in]　　汎用出力指数

例

```
void main() {
    //  Get the state of the specific general output
    if (HIMC_GPO(5)) {  //  if HIMC_GPO(5) is at the “on” state
      //  Do something
    }
    HIMC_GPO(1) = HIMC_GPI(4) && HIMC_GPI(6);  //  Set specific general output
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

20. 原点復帰機能

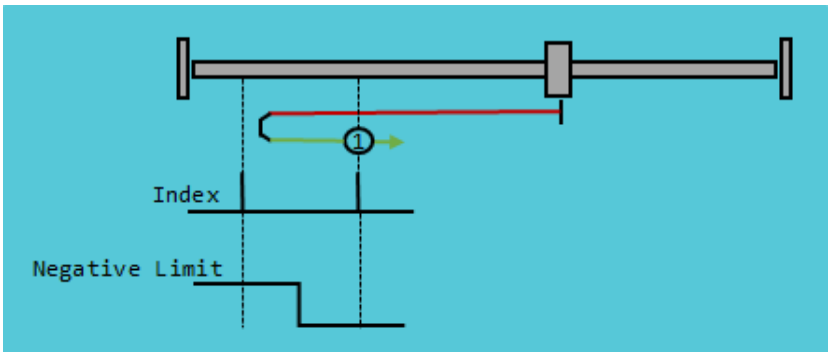
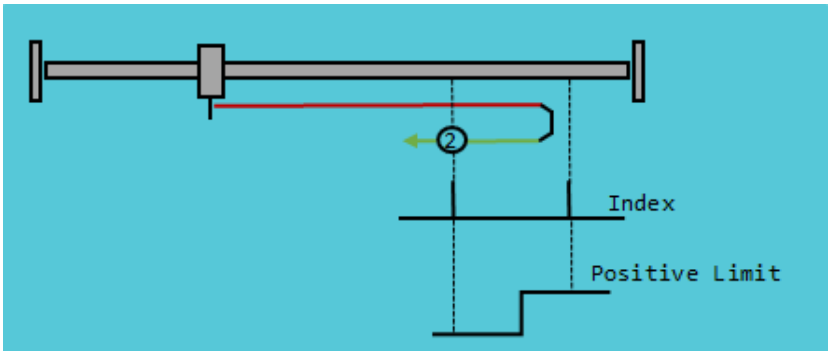
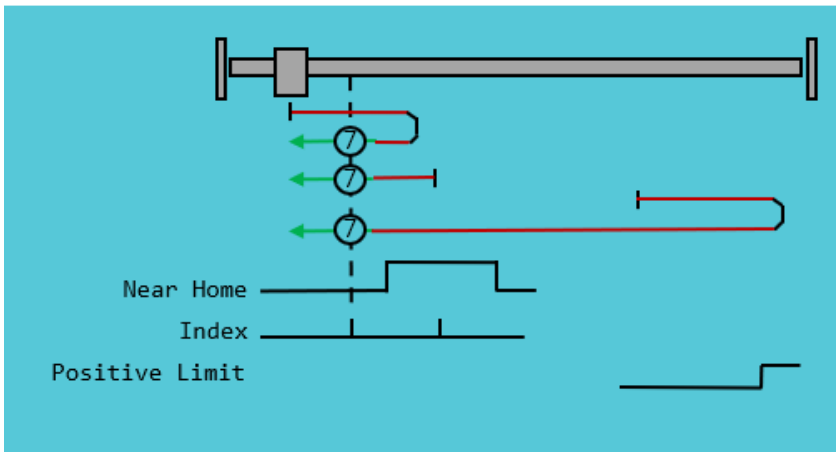
20.1 概要

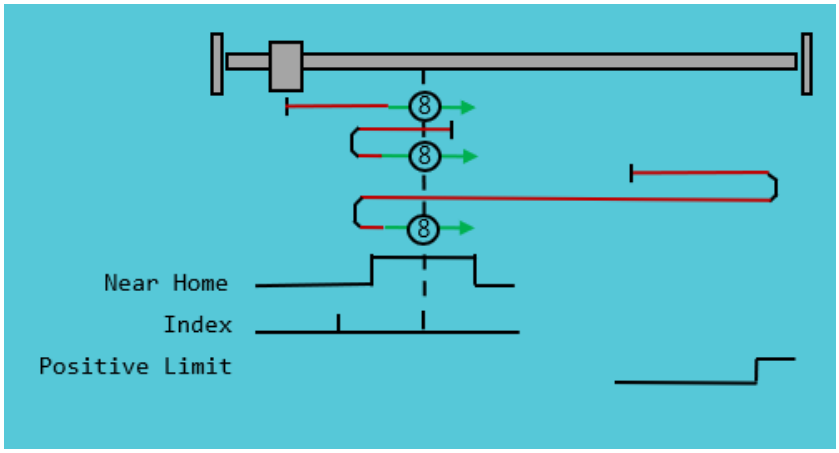
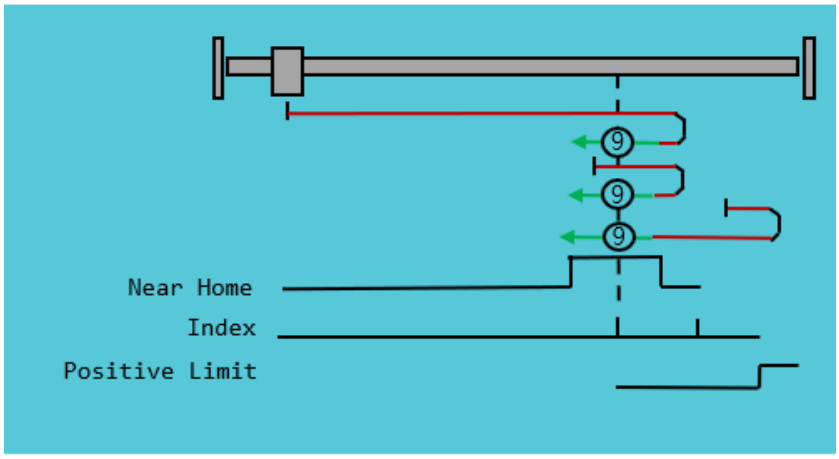
HIMC は、「リミットスイッチ/オーバートレード」の検索、「ニアホーム/DOG」の検索、「エンドストップ」原点復帰などの内部原点復帰方法を提供します。HIMC は、ユーザーがステージ構成に基づいて原点復帰方法を設定した後、原点復帰手順を実行します。すべての原点復帰方法を表 20.1.1 に示し、詳細な図と説明を表 20.1.2 に示します。

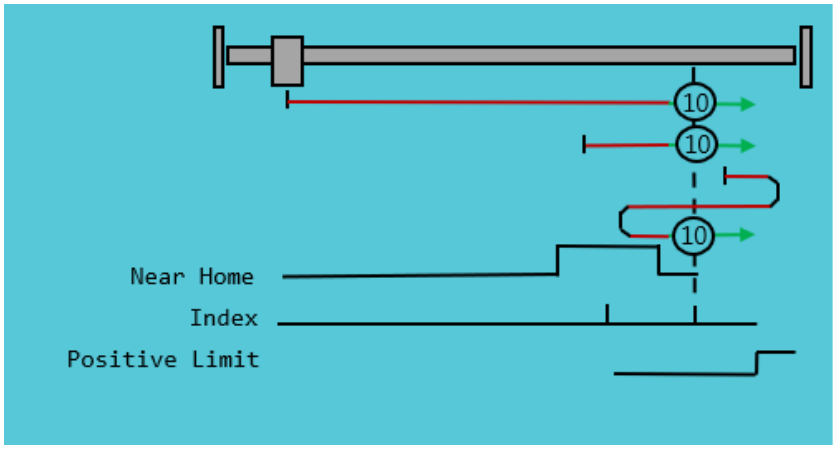
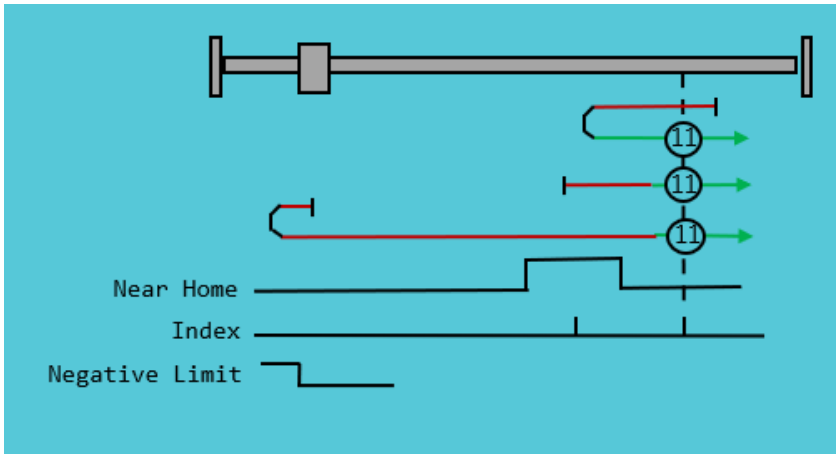
表 20.1.1

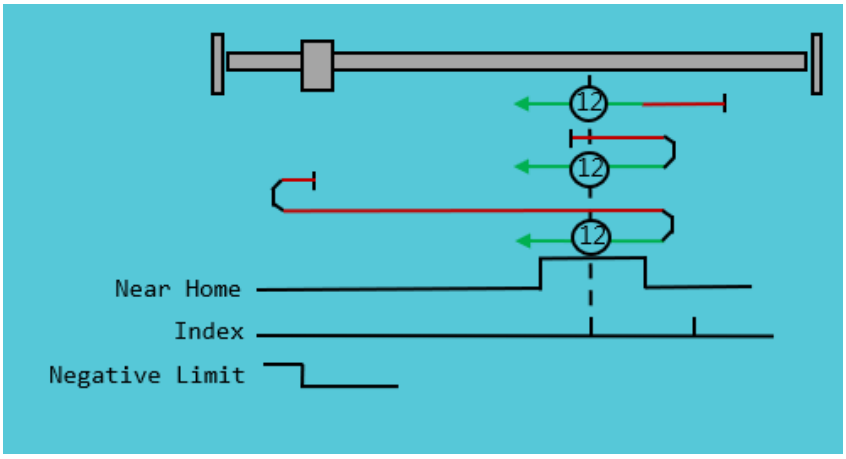
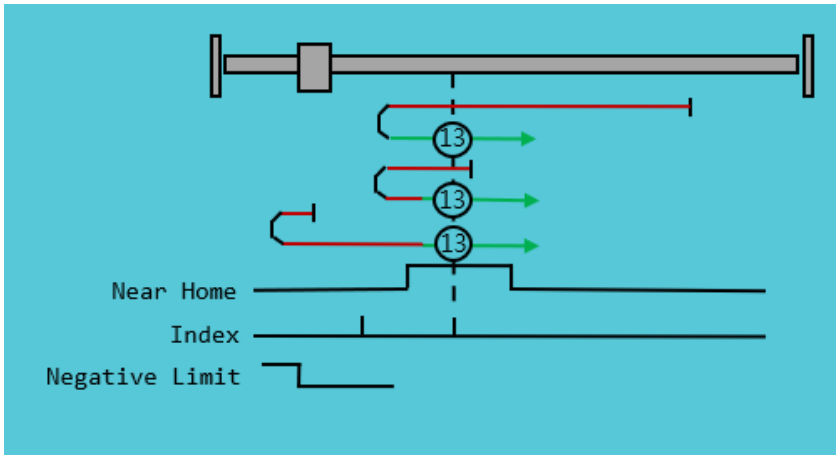
HMPL 定義	説明
HOME_METHOD_1	最初に負の方向で負の制限を検索し、正の方向でインデックスを検索します。
HOME_METHOD_2	最初に正の方向で正の制限を検索し、負の方向でインデックスを検索します。
HOME_METHOD_7	ホーム立ち上がりエッジ付近の左側のインデックスを正方向に検索します。
HOME_METHOD_8	ホーム立ち上がりエッジ付近の右側のインデックスを正方向に検索します。
HOME_METHOD_9	ホーム立ち下がりエッジ付近の左側のインデックスを正方向に検索します。
HOME_METHOD_10	ホーム立ち下がりエッジ付近の右側のインデックスを正方向に検索します。
HOME_METHOD_11	ホーム立ち上がりエッジ付近の右側のインデックスを負の方向に検索します。
HOME_METHOD_12	負方向のホーム立ち上がりエッジ付近の左側のインデックスを検索します。
HOME_METHOD_13	ホーム立ち下がりエッジ付近の右側のインデックスを負の方向に検索します。
HOME_METHOD_14	負方向のホーム立ち下がりエッジ付近の左側のインデックスを検索します。
HOME_METHOD_17	まずマイナス方向のマイナスリミットをサーチし、原点オフセットの位置に移動します。
HOME_METHOD_18	まず正方向の正リミットをサーチし、原点オフセットの位置に移動します。
HOME_METHOD_19	最初にマイナス方向のマイナスリミットを検索し、プラス方向のプラスリミットを検索します。最後に、正と負の制限の中間に移動します。
HOME_METHOD_33	最初にマイナス方向にインデックスをサーチし、原点オフセットの位置に移動します。
HOME_METHOD_34	最初に正方向にインデックスを検索し、原点オフセットの位置に移動します。
HOME_METHOD_N1	最初に負の方向で機構限界を検索し、正の方向でインデックスを検索します。
HOME_METHOD_N2	最初に正の方向で機構限界を検索し、負の方向でインデックスを検索します
HOME_METHOD_N4	まず負方向にメカリミットをサーチし、メカリミットオフセットの位置に移動します。
HOME_METHOD_N5	まず正方向にメカリミットをサーチし、メカリミットオフセットの位置に移動します。

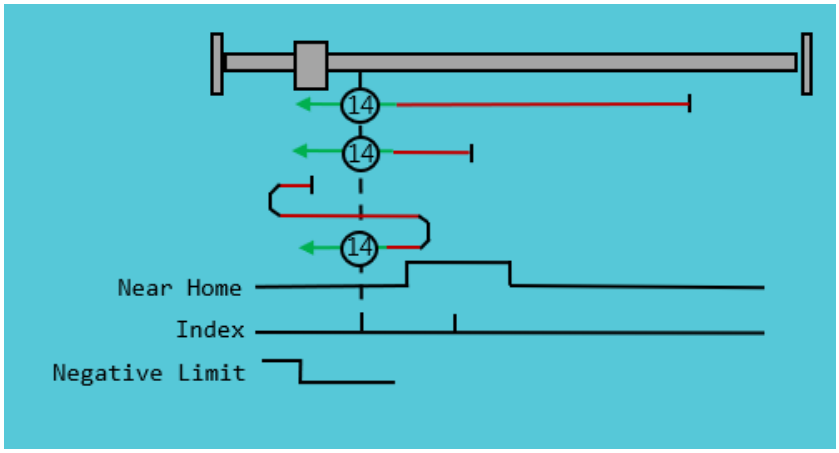
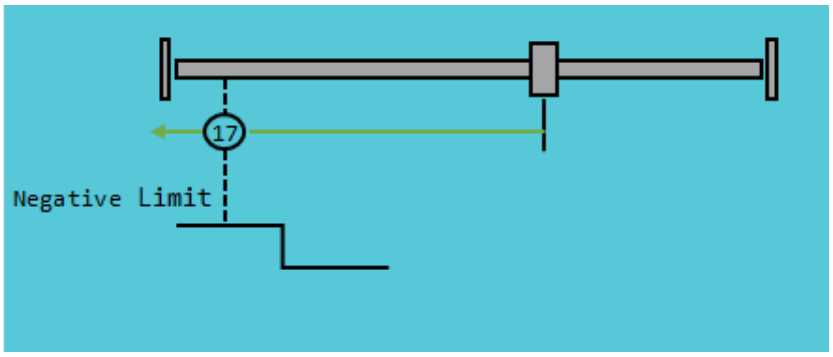
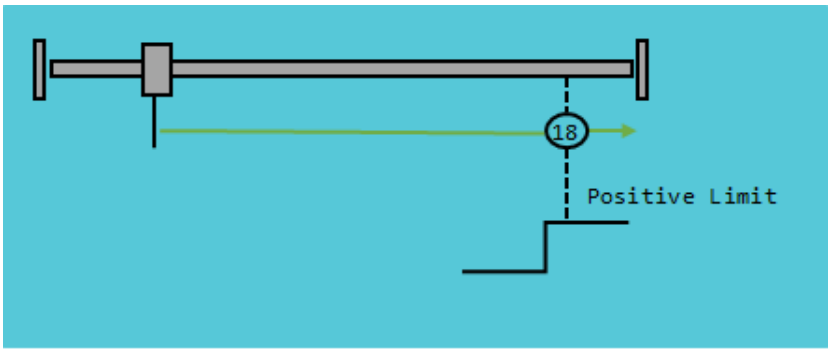
表 20.1.2

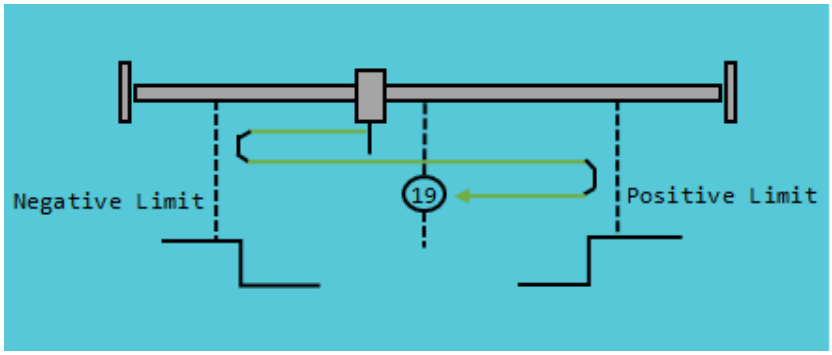
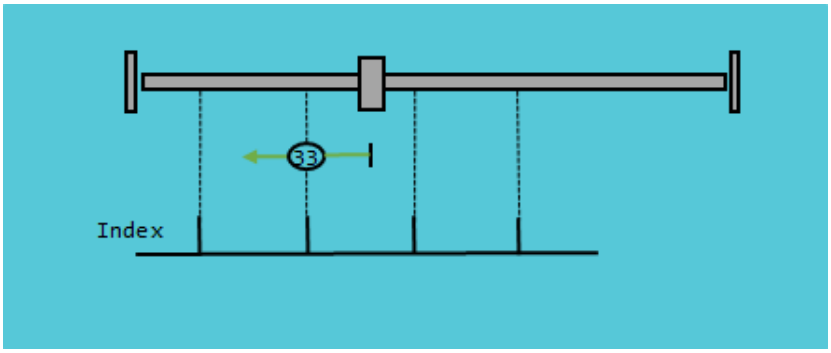
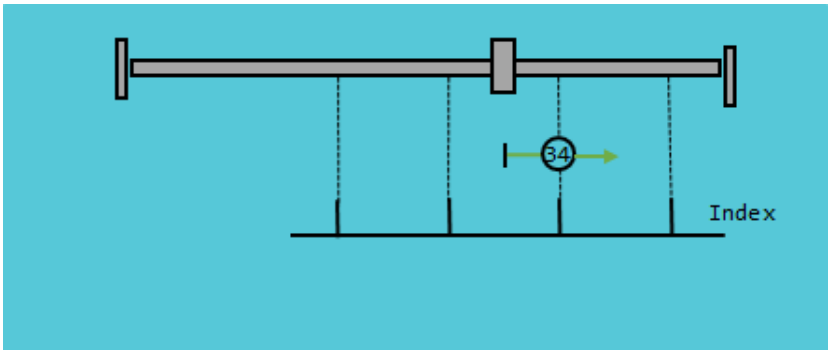
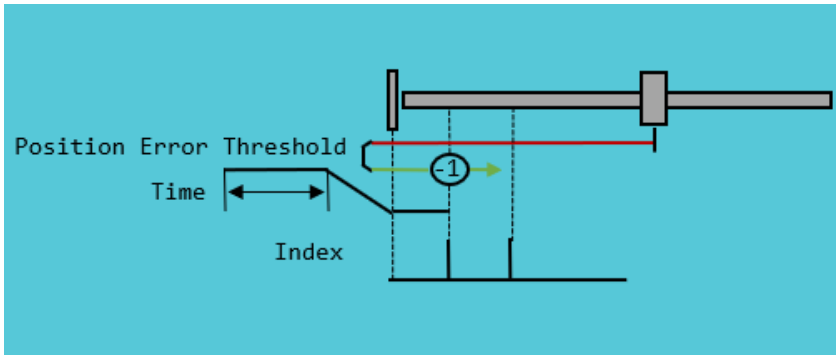
HMPL 定義	原点復帰手順
HOME_METHOD_1	
	最初に速い速度で負の方向に負のリミットを検索し、遅い速度で正の方向にインデックスを検索します。インデックスの位置を原点とし、原点オフセットまで低速で移動します。
HOME_METHOD_2	
	最初に速い速度で正の方向の正のリミットを検索し、遅い速度で負の方向のインデックスを検索します。インデックスの位置を原点とし、原点オフセットまで低速で移動します。
HOME_METHOD_7	
	1. (ニアホームではない)最初にプラス方向でニアホーム立ち上がりエッジを検索し、マイナス方向でインデックスを検索します。 2. (ニアホームで)ニアホームの立ち下がりエッジを最初に負の方向で検索し、負の方向でインデックスを検索します。 3. (ニアホームではない)最初に正方向で正のリミットを検索し、負の方向でホーム付近の立ち下がりエッジを検索します。最後に、負の方向のインデックスを検索します。

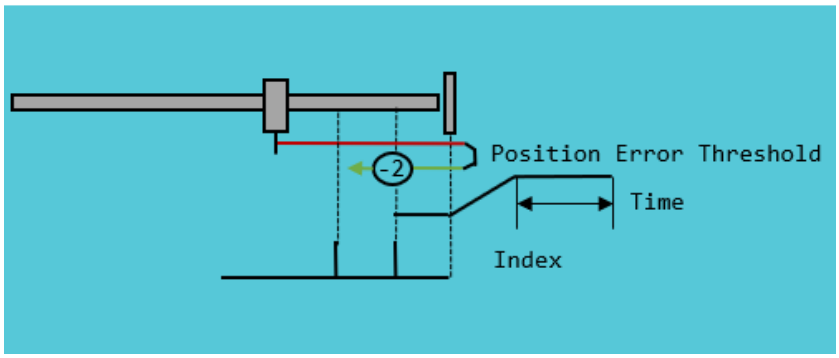
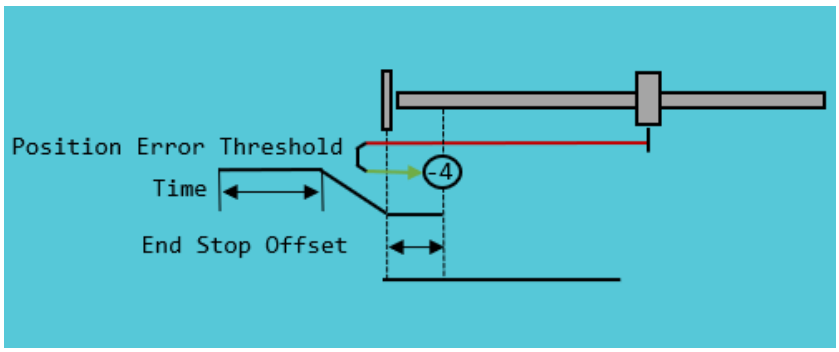
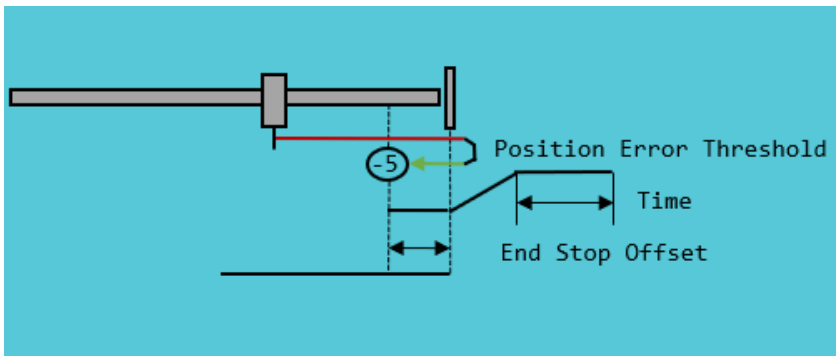
HMPL 定義	原点復帰手順
HOME_METHOD_8	
	1. (ニアホームではない)ホーム付近の立ち上がりエッジを最初に正方向に検索し、インデックスを正方向に検索します。 2. (ニアホームで)ニアホームの立ち下がりエッジを最初に負の方向で検索し、正の方向でインデックスを検索します。 3. (ニアホームではない)最初に正方向で正のリミットを検索し、負の方向でホーム付近の立ち下がりエッジを検索します。最後に、正方向のインデックスを検索します。
HOME_METHOD_9	
	1. (ニアホームではない)ニアホーム立ち下がりエッジを最初に正方向で検索し、負方向でインデックスを検索します。 2. (ニアホーム上) ニアホーム立ち下がりエッジを最初に正方向に検索し、負方向にインデックスを検索します。 3. (ホーム付近ではない)最初に正方向で正のリミットを検索し、負の方向でホームの立ち上がりエッジ付近を検索します。 最後に、負の方向のインデックスを検索します。

HMPL 定義	原点復帰手順
HOME_METHOD_10	
	1. (ニアホームではない)ニアホーム立ち下がりエッジを最初に正方向に検索し、正方向にインデックスを検索します。 2. (ニアホーム上) ニアホーム立ち下がりエッジを最初に正方向に検索し、インデックスを正方向に検索します。 3. (ホーム付近ではない) 最初に正方向で正のリミットを検索し、負の方向でホームの立ち上がりエッジ付近を検索します。最後に、正方向のインデックスを検索します。
HOME_METHOD_11	
	1. (ニアホームではない) ホーム付近の立ち上がりエッジを最初に負の方向で検索し、インデックスを正の方向で検索します。 2. (ニアホーム上)ニアホーム立ち下がりエッジを最初に正方向に検索し、インデックスを正方向に検索します。 3. (ニアホームではない)最初に負の方向で負のリミットを検索し、正の方向でホームの近くの立ち下がりエッジを検索します。最後に、正方向のインデックスを検索します。

HMPL 定義	原点復帰手順
HOME_METHOD_12	
	1. (ニアホームではない)ホーム付近の立ち上がりエッジを最初に負の方向で検索し、インデックスを負の方向で検索します。 2. (ニアホーム上)ニアホーム立ち下がりエッジを最初に正方向に検索し、負方向にインデックスを検索します。 3. (ニアホームではない)最初に負の方向で負のリミットを検索し、正の方向でホームの近くの立ち下がりエッジを検索します。最後に、負の方向のインデックスを検索します。
HOME_METHOD_13	
	1. (ニアホームではない)ニアホーム立ち下がりエッジを最初に負の方向で検索し、インデックスを正の方向で検索します。 2. (ニアホームで)ニアホームの立ち下がりエッジを最初に負の方向で検索し、正の方向でインデックスを検索します。 3. (ホーム付近ではない)最初にマイナス方向でマイナスリミットを検索し、プラス方向でホーム付近の立ち上がりエッジを検索します。最後に、正方向のインデックスを検索します。

HMPL 定義	原点復帰手順
HOME_METHOD_14	
	1. (ニアホームではない)まず負方向のニアホーム立ち下がりエッジを検索し、負方向のインデックスを検索します。 2. (ニアホームで)ニアホームの立ち下がりエッジを最初に負の方向で検索し、負の方向でインデックスを検索します。 3. (ホーム付近ではない)最初にマイナス方向でマイナスリミットを検索し、プラス方向でホーム付近の立ち上がりエッジを検索します。最後に、負の方向のインデックスを検索します。
HOME_METHOD_17	
	低速で負の方向の負のリミットを検索します。マイナスリミットの位置を原点とし、原点オフセットまで低速で移動します。
HOME_METHOD_18	
	低速で正の方向の正の限界を検索します。プラスリミットの位置を原点とし、原点オフセットまで低速で移動します。

HMPL 定義	原点復帰手順
HOME_METHOD_19	
	最初に低速で負の方向の負のリミットを検索し、低速で正の方向の正のリミットを検索します。ゆっくりとした速度で中央に移動し、中央の位置をホームポジションとします。
HOME_METHOD_33	
	低速で負方向のインデックスを検索します。インデックスをホームポジションとして、低速でホームオフセットに移動します。
HOME_METHOD_34	
	低速で正方向のインデックスを検索します。インデックスをホームポジションとして、低速でホームオフセットに移動します。
HOME_METHOD_N1	

HMPL 定義	原点復帰手順
	最初に高速で負の方向にエンドストップを検索し、低速で正の方向にインデックスを検索します。インデックスをホームポジションにします。
HOME_METHOD_N2	
	最初に速い速度で正の方向にエンドストップを検索し、遅い速度で負の方向にインデックスを検索します。インデックスをホームポジションにします。
HOME_METHOD_N4	
	速い速度で負方向のエンドストップを検索します。エンドストップをホームポジションとして、低速で正方向にエンドストップオフセットに移動します。
HOME_METHOD_N5	
	速い速度で正方向のエンドストップを検索します。エンドストップをホームポジションとして、低速で負の方向にエンドストップオフセットに移動します。

これらの原点復帰方法は、単軸原点復帰、HIMC ガントリーモード原点復帰、および E1 シリーズドライバーガントリーモード原点復帰に適用できます。各モードでサポートされている原点復帰方法を表 20.1.3 に示します。

表 20.1.3

原点復帰方法	単軸	HIMC ガントリーモード	E1 シリーズドライバー ガントリーモード
原点復帰方法 1	√	√	√
原点復帰方法 2	√	√	√
原点復帰方法 7	√	--	--
原点復帰方法 8	√	--	--
原点復帰方法 9	√	--	--
原点復帰方法 10	√	--	--
原点復帰方法 11	√	--	--
原点復帰方法 12	√	--	--
原点復帰方法 13	√	--	--
原点復帰方法 14	√	--	--
原点復帰方法 17	√	--	--
原点復帰方法 18	√	--	--
原点復帰方法 19	√	--	--
原点復帰方法 33	√	√	√
原点復帰方法 34	√	√	√
原点復帰方法 N1	√	--	--
原点復帰方法 N2	√	--	--
原点復帰方法 N4	√	--	--
原点復帰方法 N5	√	--	--

注意 1：HOME_METHOD_7 ～ HOME_METHOD_14 は非同期通信方式でドライバーのニアホームの入力信号を読み取ります。より速い速度で原点復帰する場合は、非同期通信方式による時間遅延を考慮する必要があります。

注意 2：HOME_METHOD_N1 ～ HOME_METHOD_N5 は、コントローラーの位置偏差を検出してエンドストップを検索します。アプリケーションに基づいて、SetEndStopPosErr と SetEndStopDist を使用して、ユーザーは位置エラーのしきい値とエンドストップのオフセット距離を設定できます。

注意 3：原点復帰方法はシミュレータをサポートしていません。

20.1.1 例

例 1: HOME_METHOD_33 による 1 軸原点復帰 (コントローラーで実行)

```
void main()
{
	int axis_id = 0;	//	axis index (Axis Mode)
	int home_type = 0;	//	homing type (Controller executes homing procedure)
	int home_method = HOME_METHOD_33;	//	homing method
	double fast_vel = 20;	//	homing velocity (Search for Limit Switch)
	double slow_vel = 2;	//	homing velocity (Search for Index)
	int acc_time = 0;	// acceleration time
	double home_offsets = 0;	//	home offset
	int time_out = 10000;	//	time out

	SetHomeType(axis_id, home_type);
	SetHomeMethod(axis_id, home_method);
	SetHomeProfile(axis_id, fast_vel, slow_vel, acc_time);
	SetHomeOffset(axis_id, home_offsets);
	SetHomeTimeout(axis_id, time_out);
	int result = AxisHome(axis_id);

	if ( result == 0 ) {
		Print("Home success.");
	} else {
		Print("Home fail:%d.", result);
	}
}
```

例 2: HOME_METHOD_33 による 1 軸原点復帰(E1 シリーズドライバーで実行)

```
Void main()
{
	int axis_id = 0;	//	axis index (Axis Mode)
	int home_type = 1;	//	homing type (Servo drive executes homing procedure)
	int home_method = HOME_METHOD_33;	//	homing method
	double fast_vel = 10;	//	homing velocity (Search for Limit Switch)
	double slow_vel = 5;	//	homing velocity (Search for Index)
	int acc_time = 10;	// acceleration time
	double home_offsets = 0;	//	home offset
	int time_out = 10000;	//	time out

	SetHomeType(axis_id, home_type);
	SetHomeMethod(axis_id, home_method);
	SetHomeProfile(axis_id, fast_vel, slow_vel, acc_time);
	SetHomeOffset(axis_id, home_offsets);
	SetHomeTimeout(axis_id, time_out);
	int result = AxisHome(axis_id);

	if ( result == 0 ) {
		Print("Home success.");
	} else {
		Print("Home fail:%d.", result);
	}
}
```

例 3: HOME_METHOD_1 による E1 シリーズドライバーガントリーモード原点復帰

```
void main()
{
	int axis_id = 0;	//	axis index (Axis Mode)
	int home_type = 0;	//	homing type (Controller executes homing procedure)
	int home_method = HOME_METHOD_1;	//	homing method
	double fast_vel = 20;	//	homing velocity (Search for Limit Switch)
	double slow_vel = 2;	//	homing velocity (Search for Index)
	int acc_time = 0;	// acceleration time
	double home_offsets = 0;	//	home offset
	int time_out = 10000;	//	time out

	SetHomeType(axis_id, home_type);
	SetHomeMethod(axis_id, home_method);
	SetHomeProfile(axis_id, fast_vel, slow_vel, acc_time);
	SetHomeOffset(axis_id, home_offsets);
	SetHomeTimeout(axis_id, time_out);
	int result = AxisHome(axis_id);

	if ( result == 0 ) {
		Print("Home success.");
	} else {
		Print("Home fail:%d.", result);
	}
}
```

例 4: HOME_METHOD_33 による HIMC ガントリーモード原点復帰

```
void main()
{
	//	Set GantryPair
	int axis_1 = 0;	//	Set master axis' index
	int axis_2 = 1;	//	Set slave axis' index

	int axis_id = axis_1;	//	axis index (Axis Mode)
	int home_type = 0;	//	homing type (Controller executes homing procedure)
	int home_method = HOME_METHOD_33;	//	homing method
	double fast_vel = 20;	//	homing velocity (Search for Limit Switch)
	double slow_vel = 2;	//	homing velocity (Search for Index)
	int acc_time = 0;	// acceleration time
	double home_offsets = 0;	//	home offset
	int time_out = 10000;	//	time out

	DisableGantryPair(axis_1);	//	Disable the existing gantry settings
	Till(!IsGantry(axis_1) && !IsGantry(axis_2));

	Enable(axis_1);	Till(IsEnabled(axis_1));
	Disable(axis_1);	Till(!IsEnabled(axis_1));
	Enable(axis_2);	Till(IsEnabled(axis_2));

	SetHomeType(axis_id, home_type);
	SetHomeMethod(axis_id, home_method);
	SetHomeProfile(axis_id, fast_vel, slow_vel, acc_time);
	SetHomeOffset(axis_id, home_offsets);
	SetHomeTimeout(axis_id, time_out);
	int result = AxisHome(axis_id);

	if ( result == 0 ) {
		Print("Home success.");
	} else {
		Print("Home fail:%d.", result);
	}
}
```

例 4: HOME_METHOD_N1 による単軸原点復帰

```
void main()
{
	int axis_id = 0;	//	axis index (Axis Mode)
	int home_method = HOME_METHOD_N1;	//	homing method
	double fast_vel = 10;	//	homing velocity (Search for End Stop)
	double slow_vel = 10;	//	homing velocity (Search for Index)
	int acc_time = 0;	// acceleration time
	int time_out = 10000;	//	time out
	double pos_err = 5;	//	position error threshold for End Stop
	double distance = 0.0;	//	offset distance for End Stop

	SetHomeMethod(axis_id, home_method);
	SetHomeProfile(axis_id, fast_vel, slow_vel, acc_time);
	SetHomeTimeout(axis_id, time_out);
	SetEndStopPosErr(axis_id, pos_err);
	SetEndStopDist(axis_id, distance);
	int result = AxisHome(axis_id);

	if ( result == 0 ) {
		Print("Home success.");
	} else {
		Print("Home fail:%d.", result);
	}
}
```

20.1.2 ユーザー定義の原点復帰手順

例 1: 正/負方向のインデックス信号を検索します

```
//   standard procedure for axis homing --- basic version
//   single axis (including E1 gantry mode with Thunder 1.4.8 or above) homing
//   (touch probe only)
//   HIMC version 1.2

/* Set parameters */
int axis = 0;
double Home_vel = -20;   //   the direction depends on + or -
double ind_offset = 0.0;   //   index position relative to 0 after homing

void main()
{
        DisableTouchProbe(axis);
        Till(!IsTouchProbeEnabled(axis));
        EnableTouchProbe(axis);
        Till(IsTouchProbeEnabled(axis));

        Enable(axis);
        Till(IsEnabled(axis));
        IgnoreSWL(axis, true);

        Print("Search index...");

        MoveVel(axis,-Home_vel);

        Till(IsTouchProbeTriggered(axis) || !IsMoving(axis) || IsHWLL(axis));
        Stop(axis);
        Till(!IsMoving(axis));

        if (!IsEnabled(axis)) {goto Error;}
        else if (IsHWLL(axis)) {goto Error;}
        else if (IsTouchProbeTriggered(axis)){
                Print("Index found.");
                double pos1, pos2;
```

```
                GetTouchProbePos(axis, &pos1);
                pos2 = GetPosFb(axis) - pos1 + ind_offset;
                SetPos(axis, pos2);
                Print("Go to zero...");
                MoveAbs(axis, 0.0);    //    go to zero
                Till(!IsMoving(axis));
                //    If touch probe homing is used in E1 gantry mode, remove /* and */.
                /*    if (GetSlvSt(axis,"X_gantry_active")) {
                            SetSlvVar(axis, "gantry.apply_mode", 11);
                       Till (GetSlvSt(axis,"X_yaw_align"));
                } */
                if (GetRefPos(axis)==0.0 && IsEnabled(axis)) {goto HomeSuccess;}
                else {goto Error;}
                }

        Error:
        Print("Axis homing fails.");
        goto RestoreFaultResponse;

        HomeSuccess:
        Print("Axis homing succeeds.");
        goto RestoreFaultResponse;

        RestoreFaultResponse:
        IgnoreSWL(axis, false);
}
```

例 2: 最初に正/負方向のリミットを検索し、インデックス信号を検索します

```
//   standard procedure for axis homing --- advanced version
//   single axis (including E1 gantry mode with Thunder 1.4.8 or above)
//   and HIMC gantry mode homing
//   (touch probe and limit switch)
//   HIMC version 1.2

/* Set parameters */
int Homing_Type=0;   //   homing method
int axis = 1;   //   Configure the axis
int axis_1 = 2;   //   Configure the axes to a HIMC gantry pair
double Home_vel = -20;   //   the direction depends on + or -

//   Homing_Type=0 : Find the index directly without searching limit switch
//   Homing_Type=1 : Find the left limit switch and index
//   Homing_Type=2 : Set the middle point of switch(L,R) as the home position

//   Type 3 & 4 is for HIMC gantry mode
//   Homing_Type=3 : the motion for homing is the same as Homing_Type=0
//   Homing_Type=4 : the motion for homing is the same as Homing_Type=1

double ind_offset = 0.0;   //   index position relative to 0 after homing

int Homing_Type_0(void);
int Homing_Type_1(void);
int Homing_Type_2(double *);
int Homing_Type_3(void);
int Homing_Type_4(void);

void main()
{
    int err=0;
    if (Homing_Type==0 || Homing_Type==1)
    {
        Print("Start single axis homing...");
        if (Homing_Type==0){
            err=Homing_Type_0();
            if (err==1) {goto Error;}
```

```
            }
        if (Homing_Type==1){
            err=Homing_Type_1();
            if (err==1) {goto Error;}
        }
        if (!IsEnabled(axis)) {goto Error;}
        else if (IsHWLL(axis)) {goto Error;}
        else if (IsTouchProbeTriggered(axis)) {

            Print("Index found.");
            double pos1, pos2;
            GetTouchProbePos(axis, &pos1);
            pos2 = GetPosFb(axis) - pos1 + ind_offset;
            SetPos(axis, pos2);

            Print("Go to zero...");
            MoveAbs(axis, 0.0);    //    go to zero

            Till(!IsMoving(axis));
            //    If E1 gantry mode is used, remove /* and */.
            /*    if (GetSlvSt(axis,"X_gantry_active")) {
                        SetSlvVar(axis, "gantry.apply_mode", 11);
                  Till (GetSlvSt(axis,"X_yaw_align"));
            } */
            if (GetRefPos(axis)==0.0 && IsEnabled(axis)) {goto HomeSuccess;}
            else {goto Error;}
        }
        else {goto Error;}
    }

    if (Homing_Type==2)
    {
        double home_pos;
        Print("Start single axis homing...");
        if (Homing_Type==2){
            err=Homing_Type_2(&home_pos);
            if (err==1) {goto Error;}
        }
```

```
        if (!IsEnabled(axis)) {goto Error;}
        else {
            Print("Hardware limit found.");
            Print("Go to home position...");
            MoveAbs(axis, home_pos);    //    go to zero
            Till(!IsMoving(axis));
            SetPos(axis, 0.0);
            if (GetRefPos(axis)==0.0 && IsEnabled(axis)) {goto HomeSuccess;}
            else {goto Error;}
        }
    }

    if (Homing_Type==3 || Homing_Type==4)
    {
        Print("Start gantry pair homing...");
        if (Homing_Type==3){
            err=Homing_Type_3();
            if (err==1) {goto Error;}
        }
        if (Homing_Type==4){
            err=Homing_Type_4();
            if (err==1) {goto Error;}
        }

        if (!IsEnabled(axis)) {goto Error;}
        else if (IsHWLL(axis)) {goto Error;}
        else if (IsTouchProbeTriggered(axis))
        {
            Print("Index found.");
            double pos1, pos2, pos3, pos4;
            GetTouchProbePos(axis, &pos1);
            GetTouchProbePos(axis_1, &pos3);

            pos2 = GetPosFb(axis) - pos1 + ind_offset;
            pos4 = GetPosFb(axis_1) - pos3 + ind_offset;

            SetPos(axis, pos2);
```

```
                SetPos(axis_1, pos4);

                //   Enable HIMC gantry pair
                Disable(axis); Disable(axis_1);
                Till(!IsEnabled(axis)&& !IsEnabled(axis_1));
                EnableGantryPair(axis, axis_1);
                Till(IsGantry(axis) && IsGantry(axis_1));

                Enable(axis);
                Till(IsEnabled(axis) && IsEnabled(axis_1));

                Print("Go to zero...");
                MoveAbs(axis, 0.0);

                Till(!IsMoving(axis));
                if (0.0 == GetRefPos(axis) && IsEnabled(axis)) {goto HomeSuccess;}
                else {goto Error;}
            }
            else {goto Error;}
        }

        Error:
        Print("Axis homing fails.");
        goto RestoreFaultResponse;

        HomeSuccess:
        Print("Axis homing succeeds.");
        goto RestoreFaultResponse;

        RestoreFaultResponse:
        IgnoreSWL(axis, false);
        IgnoreSWL(axis_1, false);
}

int Homing_Type_0()
{
        DisableTouchProbe(axis);
        Till(!IsTouchProbeEnabled(axis));
```

```
    EnableTouchProbe(axis);
    Till(IsTouchProbeEnabled(axis));

    Enable(axis);
    Till(IsEnabled(axis));
    IgnoreSWL(axis, true);

    Print("Search index...");

    MoveVel(axis, -Home_vel);

    Till(IsTouchProbeTriggered(axis) || !IsMoving(axis) || IsHWLL(axis));

    Stop(axis);
    Till(!IsMoving(axis));
    return 0;
}

int Homing_Type_1()
{
    Enable(axis);
    Till(IsEnabled(axis));
    IgnoreSWL(axis, true);
    IgnoreHWL(axis, true);

    //   Find limit switch
    if (!IsHWLL(axis)) {
        Print("Search hardware left limit...");
        MoveVel(axis, Home_vel);
    }
    Till(!IsMoving(axis) || IsHWLL(axis));
    Stop(axis);
    Till(!IsMoving(axis));
    if(IsHWLL(axis)==false) {return 1;}
    Print("Hardware left limit found.");

    DisableTouchProbe(axis);
    Till(!IsTouchProbeEnabled(axis));
```

```
        EnableTouchProbe(axis);
        Till(IsTouchProbeEnabled(axis));

        Print("Search Index...");

        MoveVel(axis, -Home_vel);

        Till(IsTouchProbeTriggered(axis) || !IsMoving(axis) || IsHWRL(axis));
        Stop(axis);
        Till(!IsMoving(axis));
        return 0;
}

int Homing_Type_2(double *home_pos)
{
        Enable(axis);
        Till(IsEnabled(axis));
        IgnoreSWL(axis, true);
        IgnoreHWL(axis, true);
        double pos1, pos2;

        //    Find left limit switch
        if (!IsHWLL(axis)) {
                Print("Search hardware left limit...");
                MoveVel(axis, Home_vel);
        }
        Till(IsHWLL(axis)) {
           pos1 = GetPosFb(axis);
        };

        Stop(axis);
        Till(!IsMoving(axis));
        if (IsHWLL(axis)==false) {return 1;}
        Print("Hardware left limit found.");

        //    Find right limit switch
        if (!IsHWRL(axis)) {
                Print("Search hardware right limit...");
```

```
            MoveVel(axis, -Home_vel);
        }
        Till(IsHWRL(axis)) {
            pos2 = GetPosFb(axis);
        };

        Stop(axis);
        Till(!IsMoving(axis));
        if (IsHWRL(axis)==false) {return 1;}
        Print("Hardware right limit found.");

        *home_pos = (pos2+pos1)/2.0;    //    pos3 is home position
        return 0;
}

int Homing_Type_3()
{
        DisableGantryPair(axis);
        Till(!IsGantry(axis) && !IsGantry(axis_1));

        DisableTouchProbe(axis); DisableTouchProbe(axis_1);
        Till(!IsTouchProbeEnabled(axis) && !IsTouchProbeEnabled(axis_1));
        EnableTouchProbe(axis); EnableTouchProbe(axis_1);
        Till(IsTouchProbeEnabled(axis) && IsTouchProbeEnabled(axis_1));

        Enable(axis); Enable(axis_1);
        Till(IsEnabled(axis) && IsEnabled(axis_1));

        IgnoreSWL(axis, true); IgnoreSWL(axis_1, true);

        MoveVel(axis, -Home_vel); MoveVel(axis_1, -Home_vel);

        Till((IsTouchProbeTriggered(axis) && IsTouchProbeTriggered(axis_1)) ||
            (!IsMoving(axis) || !IsMoving(axis_1) || IsHWLL(axis) || IsHWLL(axis_1)));

        Stop(axis); Stop(axis_1);
        Till(!IsMoving(axis) && !IsMoving(axis_1));
        return 0;
```

```
}

int Homing_Type_4()
{
	DisableGantryPair(axis);
	Till(!IsGantry(axis) && !IsGantry(axis_1));

	DisableTouchProbe(axis); DisableTouchProbe(axis_1);
	Till(!IsTouchProbeEnabled(axis) && !IsTouchProbeEnabled(axis_1));
	EnableTouchProbe(axis); EnableTouchProbe(axis_1);
	Till(IsTouchProbeEnabled(axis) && IsTouchProbeEnabled(axis_1));

	Enable(axis); Enable(axis_1);
	Till(IsEnabled(axis) && IsEnabled(axis_1));

	IgnoreHWL(axis, true); IgnoreHWL(axis_1, true);
	IgnoreSWL(axis, true); IgnoreSWL(axis_1, true);

	//   Find limit switch
	if (!IsHWLL(axis) || !IsHWLL(axis_1)) {
		Print("Search hardware left limit...");
		MoveVel(axis, Home_vel);
		MoveVel(axis_1, Home_vel);
	}

	Till(!IsMoving(axis) || !IsMoving(axis_1) || IsHWLL(axis) || IsHWLL(axis_1));

	Stop(axis); Stop(axis_1);

	Till(!IsMoving(axis) && !IsMoving(axis_1));
	if (IsHWLL(axis)==false && IsHWLL(axis_1)==false) {return 1;}
	Print("Hardware left limit found.");

	MoveVel(axis, -Home_vel); MoveVel(axis_1, -Home_vel);

	Till((IsTouchProbeTriggered(axis) && IsTouchProbeTriggered(axis_1)) ||
		(!IsMoving(axis) || !IsMoving(axis_1) || IsHWLL(axis) || IsHWLL(axis_1)));
```

```
    Stop(axis); Stop(axis_1);
    Till(!IsMoving(axis) && !IsMoving(axis_1));
    return 0;
}
```

20.2 MoveHome

目的

軸の原点復帰手順を実行します。

構文

```
Int MoveHome(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

備考

この機能を使用する場合、ユーザーはオブジェクト 0x6060 (動作モード) とオブジェクト 0x6061 (動作モード表示) を PDO として設定する必要があります。

条件

サポートされる最小バージョン	iA Studio 3.0

20.3 SetHomeMethod

目的

原点復帰手順の原点復帰方法を設定します

構文

```
int SetHomeMethod(
        int axis_id,
        int method
);
```

パラメーター

axis_id [in] Axis index

method [in] 原点復帰方式インデックスの HMPL 定義の説明。
詳細については、表 20.1.1 および表 20.1.2 を参照してください。
初期値は HOME_METHOD_33 です。

原点復帰方法インデックス	HMPL 定義の説明	原点復帰方法インデックス	HMPL 定義の説明
1	原点復帰方法 1	17	原点復帰方法 17
2	原点復帰方法 2	18	原点復帰方法 18
7	原点復帰方法 7	19	原点復帰方法 19
8	原点復帰方法 8	33	原点復帰方法 33
9	原点復帰方法 9	34	原点復帰方法 34
10	原点復帰方法 10	-1	原点復帰方法 N1
11	原点復帰方法 11	-2	原点復帰方法 N2
12	原点復帰方法 12	-4	原点復帰方法 N4
13	原点復帰方法 13	-5	原点復帰方法 N5
14	原点復帰方法 14		

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はゼロ以外の値を返します。

備考

原点復帰タイプが「コントローラーが原点復帰手順を実行する」の場合、上記の原点復帰方法が利用可能です。「ドライバーが原点復帰手順を実行する」の場合、原点復帰方法については、E シリーズドライバーユーザーマニュアルを参照してください。

条件

サポートされる最小バージョン	iA Studio 2.0

20.4 SetHomeSwitchVel

目的

原点復帰手順の原点復帰速度を速く設定します。

構文

```
Int SetHomeSwitchVel(
    int axis_id,
    double fast_vel
);
```

パラメーター

axis_id [in]	Axis index
fast_vel [in]	高速ホーミング速度。デフォルト値は 20 です。 パラメーター単位: mm/s または deg/s

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

20.5 SetHomeZeroVel

目的

原点復帰手順の原点復帰速度を遅く設定します。

構文

```
Int SetHomeZeroVel(
    int axis_id,
    double slow_vel
);
```

パラメーター

axis_id [in]	Axis index
slow_vel [in]	遅いホーミング速度。デフォルト値は 5 です。 パラメーターの単位: mm/s または deg/s

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

20.6 SetHomeAcc

目的

原点復帰手順の原点復帰加速度を設定します。

構文

```
Int SetHomeAcc(
    int axis_id,
    doubleacc
);
```

パラメーター

axis_id [in]	Axis index
acc [in]	ホーミング加速、デフォルト値は 2000 です。 パラメーターの単位: mm/s または deg/s

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

20.7 SetHomeOffset

目的

原点復帰手順の原点オフセットを設定します。

構文

```
Int SetHomeOffset(
    int axis_id,
    double offset
);
```

パラメーター

axis_id [in]	Axis index
offset [in]	ホームオフセット。 デフォルト値は 0 です。 パラメーターの単位: mm または deg

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

20.8 SetHomeTimeout

目的

原点復帰手順のタイムアウトを設定します。

構文

```
Int SetHomeTimeout(
    int axis_id,
    int timeout
);
```

パラメーター

axis_id [in]	Axis index
timeout [in]	タイムアウト。デフォルト値は 120,000 です。 パラメーター単位：ms

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合は 0 以外の値を返します。

条件

サポートされる最小バージョン	iA Studio 2.0

20.9 IsHomed

目的

軸が原点復帰手順を完了したかどうかを問い合わせます。

構文

```
Int IsHomed(
   int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸が原点復帰手順を完了した場合は int 値 TRUE (1) を返し、それ以外の場合は FALSE (0) を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

20.10 IsHoming

目的

軸が原点復帰手順を実行しているかどうかを問い合わせます。

構文

```
Int IsHoming(
    int axis_id
);
```

パラメーター

axis_id [in] Axis index

戻りの値

軸が原点復帰手順を実行している場合は int 値 TRUE (1) を返し、それ以外の場合は FALSE (0) を返します。

条件

サポートされる最小バージョン	iA Studio 3.0

（このページはブランクになっています）

21. 通信機能

21.1 概要

HIMC は、TCP/IP プロトコルとネットワークインターフェイスを使用して、API、Modbus、および ASCII の 3 つの通信方法を提供します。これらの方法により、HIMC は関連するデバイスまたはユーザーが開発したアプリケーションに接続し、通信を行うことができます。ネットワークソケット通信の仕組みにより、HIMC をサーバーやクライアントとして利用し、関連機器やアプリケーションを受け付け、HIMC API(注 1)、Modbus(注 2)、ACSII 経由で通信を行うことができます。

HIMC をサーバーとして使用する場合、HMPL は、Native ASCII および User ASCII を含む ASCII 通信に関連する機能を提供します。Native ASCII のデフォルト接続ポートは 3999、User ASCII のデフォルト接続ポートは 4000 です (注 3)。ネイティブ ASCII は、このマニュアルの各関数の説明の上部に  としてマークされている HMPL 関数のほとんどをサポートできます。一方、ユーザーASCII は、ユーザー定義のパーサー(セクション 21.2.1 START_ASCII_AGENT を参照)を介してユーザー定義の文字列インターフェイスを使用して、より柔軟にすることができます。

HIMC をクライアントとして使用すると、他のデバイスに接続できます。HMPL は ASCII および Modbus 通信機能を提供します。ASCII クライアント機能は、通信デバイスとの間で ASCII コードデータを送受信できます。一方、Modbus クライアント機能は、保持レジスタ、入力レジスタ、コイル、ディスクリート入力を含む Modbus のメモリブロックで読み取り/書き込み操作を実行できます。

注 1: 「HIMC API リファレンスガイド」を参照してください

注 2: 「Modbus TCP ユーザーガイド」を参照してください。

注 3: ユーザーは、iA Studio の IP 設定を介して接続ポートを変更できます。「iA Studio ユーザーガイド」のセクション 4.12 を参照してください。

21.2 アスキー通信

21.2.1 START_ASCII_AGENT

目的

コントローラーをサーバーとして使用して、ユーザー定義の ASCII コマンドパーサーエージェントを開始します。

構文

```
START_ASCII_AGENT(
    parser_function
);
```

パラメーター

parser_function [in] パーサー関数の名前。
パーサー関数は、ASCII コマンドが応答を入出力できるようにするバイナリ関数である必要があります。
つまり、そのプロトタイプは次のとおりです。
void (*ParserFunctionPrototype)(char *command, char *response)

戻りの値

N/A

例 1.

```
void AsciiAgent(char *cmd, char *res) {

  for (int i = 0; ; ++i){

    if (cmd[i] != '\0') {
      res[i] = cmd[i] + 1;
    } else {
      res[i] = '\0';
      break;
    }
  }
}
```

```
void main() {

    START_ASCII_AGENT(AsciiAgent);
    //  Run it in any task and key some words in Message Window to get return ASCII
    //  If the ASCII command is “hello”, the response is “ifmmp”.
    //  If the ASCII command is “asdf”, the response is “bteg”.
}
```

例 2.

```
void AsciiAgent(char *cmd, char *res) {

    char token_str[3][40];
    int token_start = 0;
    int token_num = 0;
    for (int i = 0; i < 3; ++i){
      int token_len = StrFindChar(&cmd[token_start], ' ');
      StringCopyEx(token_str[i], &cmd[token_start], 0, token_len);
      ++token_num;
      Print("%s", token_str[i]);

      if (token_len > 0) {
        int space_len = StrFindCharEx(&cmd[token_start + token_len], " ", true);
        token_start += token_len + space_len;
      } else {
        token_start = -1;
      }
      if (token_start < 0){
        break;
      }
    }
    Print("token number: %d", token_num);

    double token2_value = 0;
    double token3_value = 0;
    if (token_num >= 2){
      token2_value = StringToDouble(token_str[1]);
      if (token_num >= 3){
        token3_value = StringToDouble(token_str[2]);
```

```
        }
    }

    if (IsStringEqual(token_str[0], "ENABLE")){
        if (token_num == 2){
            Enable(token2_value);
        }
    }
    else if (IsStringEqual(token_str[0], "MOVEABS")){
        if (token_num == 3){
            MoveAbs(token2_value, token3_value);
        }
    } else if (IsStringEqual(token_str[0], "MOVEREL")){
        if (token_num == 3){
            MoveRel(token2_value, token3_value);
        }
    } else if (IsStringEqual(token_str[0], "STOP")){
        if (token_num == 2){
            Stop(token2_value);
        }
    }
}
void main() {
    Till(IsOperMode());
    START_ASCII_AGENT(AsciiAgent);
    //  the valid command is:
    //  ENABLE 0
    //  MOVEABS 0   0.05
    //  MOVEREL 0   0.01
    //  STOP 0
}
```

条件

サポートされる最小バージョン	iA Studio 0.23

21.2.2 ASCII_ServerBroadcast

目的

コントローラーをサーバーとして、接続されているすべてのクライアントにメッセージを送信します。

構文

```
void ASCII_ServerBroadcast(
    char *message,
    int  length
);
```

パラメーター

message [in] ブロードキャストメッセージの文字列

length [in] ブロードキャストメッセージの文字列の長さ。その最大値は 128 です。

戻りの値

N/A

備考

この機能を実行する前に、User ASCII の接続を完了してください。

例

```
void main() {
    char buf[128] = {0};
    //  Write the string to be sent “test” by function StringCopy
    //  \n stands for newline, and \r stands for carriage return.
    StringCopy(buf, "test\n\r");
    int len = StringLen(buf);
    ASCII_ServerBroadcast(buf, len);  //  Send the string command
}
```

条件

サポートされる最小バージョン	iA Studio 1.4

21.2.3 ASCII_ClientConnect

目的

コントローラーをクライアントとして、サーバーとの ASCII 通信を構築します。

構文

```
int ASCII_ClientConnect(
        char *ip,
        char *port,
        int    *socket_id
);
```

パラメーター

ip [in]	サーバーに接続するための IP アドレス
port [in]	サーバーに接続するための通信ポート
socket_id [out]	サーバーに正常に接続されたソケット ID を受け取るバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char ip[15] = "192.168.0.2";
        char port[5] = "1234";
        int    socket_id;
        int    err = ASCII_ClientConnect(ip, port, &socket_id);
       switch (err) {
            case 0:
                Print("Connect Sucess: Sock id %d", socket_id);
                break;
            case 0x17D5:
                Print("Connect Fail: Please check ip & port or already reach limit");
                break;
            case 0x17D4:
                Print("Connect Fail: Connect timeout");
                break;
```

```
            default:
                Print("Connect Fail");
                break;
        }
}
```

条件

サポートされる最小バージョン	iA Studio 1.4

21.2.4 ASCII_ClientRecv

目的

コントローラーをクライアントとして、サーバーから送信された ASCII データを受信します。

構文

```
int ASCII_ClientRecv(
        int    socket_id,
        int    length,
        char *buffer
);
```

パラメーター

socket_id [in]	ASCII データを受け取るソケット ID
length [in]	ASCII データを受け取る文字列の長さ。その最大値は 512 です。
buffer [out]	受信する ASCII データを受け取るバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char buff[200] = {0};
        int    len = 100;
        int    socket_id = 10;
        int    err = ASCII_ClientRecv(socket_id, len, buff);
        switch (err) {
                case 0:
                        Print("Recv = %s", buff);
                        break;
                case 0x17DF:
                        Print("Recv Fail: Can't Recv from this client");
                        break;
                case 0x17DE:
                        Print("Recv Fail: Timeout");
                        break;
```

```
            default:
                Print("Recv Fail");
                break;
        }
}
```

条件

サポートされる最小バージョン	iA Studio 1.4

21.2.5 ASCII_ClientSend

目的

コントローラーをクライアントとして、ASCII データをサーバーに送信します。

構文

```
int ASCII_ClientSend(
        int    socket_id,
        int    length,
        char *buffer
);
```

パラメーター

socket_id [in]	ASCII データを送信するためのソケット ID
length [in]	ASCII データを送信する文字列の長さ。その最大値は 512 です。
buffer [in]	送信する ASCII データを格納するバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char msg[20] = "Test String";
        int    len = 100;
        int    socket_id = 10;
        int    err = ASCII_ClientSend(socket_id, len, msg);
       switch (err) {
            case 0:
                Print("Send OK = %s", msg);
                break;
            case 0x17E9:
                Print("Send Fail: Can't Send from this client");
                break;
            case 0x17E8:
                Print("Send Fail: Timeout");
                break;
```

```
            default:
                Print("Send Fail");
                break;
        }
}
```

条件

サポートされる最小バージョン	iA Studio 1.4

21.2.6 ASCII_ClientDisconnect

目的

コントローラーをクライアントにして、サーバーとの ASCII 通信を終了します。

構文

```
int ASCII_ClientDisconnect(
        int socket_id
);
```

パラメーター

socket_id [in]　　通信を終了するソケット ID

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char ip[15] = "192.168.0.2";
        char port[5] = "1234";
        int socket_id;
        int err = ASCII_ClientConnect(ip, port, &socket_id);
        if (err) {
                Print("Connect Fail");
                return;
        }
        //   Do something: Read/Write by function ASCII_ClientRecv and ASCII_ClientSend
        err = ASCII_ClientDisconnect(socket_id);
        if (err == 0x17D8)
                Print("Disconnect Timeout");
}
```

条件

サポートされる最小バージョン	iA Studio 1.4

21.3 Modbus 通信

21.3.1 Modbus_ClientConnect

目的

サーバーとの Modbus 通信を構築するために、コントローラーをクライアントとして使用します。

構文

```
int Modbus_ClientConnect(
        char *ip,
        int    *socket_id
);
```

パラメーター

ip [in]	サーバーに接続するための IP アドレス
socket_id [out]	サーバーに正常に接続されたソケット ID を受け取るバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char ip[15] = "169.254.188.17";
        int    socket_id;
        int    err = Modbus_ClientConnect(ip, &socket_id);
       switch (err) {
            case 0:
                Print("Connect Sucess: Sock id %d", socket_id);
                break;
            case 0x1839:
                Print("Connect Fail: Please check ip or already reach limit");
                break;
            case 0x1838:
                Print("Connect Fail: Connect timeout");
                break;
```

```
            default:
                Print("Connect Fail");
                break;
        }
}
```

条件

サポートされる最小バージョン	iA Studio 2.0

21.3.2 Modbus_ClientDisconnect

目的

サーバーとの Modbus 通信を終了するには、コントローラーをクライアントとして使用します。

構文

```
int Modbus_ClientDisconnect(
        int socket_id
);
```

パラメーター

socket_id [in]　　通信を終了するソケット ID

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char ip[15] = "169.254.188.17";
        int socket_id;
        int err = Modbus_ClientConnect(ip, &socket_id);
        if (err) {
                Print("Connect Fail");
                return;
        }
        //    Do something: Perform Read/Write operation with Modbus related functions
        err = Modbus_ClientDisconnect(socket_id);
        if (err == 0x183C)
                Print("Disconnect Timeout");
}
```

条件

サポートされる最小バージョン	iA Studio 2.0

21.3.3 Modbus_ClientRead_HoldReg

目的

コントローラーをクライアントにして、サーバーの Modbus 保持レジスタデータを読み取ります。

構文

```
int Modbus_ClientRead_HoldReg(
        int socket_id,
        uint16_t start_addr,
        uint16_t num_regs,
        uint8_t *output_buf,
        int *use_length
);
```

パラメーター

socket_id [in]	保持レジスタデータを読み取るためのソケット ID
start_addr [in]	読み取る保持レジスタデータの開始アドレス
num_regs [in]	読み取る保持レジスタデータの数。その最大値は 125 です。
output_buf [out]	読み取る保持レジスタデータを格納するバッファーへのポインター。
use_length [out]	「output_buf」の使用長を格納するバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char ip[15] = "169.254.188.17";
        int    socket_id;
        int    err = Modbus_ClientConnect(ip, &socket_id);
        //    Wait for client to connect to server
        Sleep(5000);
        if (!err) {
                uint8_t bufs[512];
                int len = 0;
                int addr = 30;
                int regs = 1;
```

```
            err = Modbus_ClientRead_HoldReg(socket_id, addr, regs, bufs, &len);
            if(!err){
                for(int i = 0; i < len; i++){
                    Print("%x", bufs[i]);
                }
            }
        }
}
```

条件

サポートされる最小バージョン	iA Studio 2.0

21.3.4 Modbus_ClientRead_InputReg

目的

サーバーの Modbus 入力レジスタデータを読み取るために、コントローラーをクライアントとして使用します。

構文

```
int Modbus_ClientRead_InputReg(
        int socket_id,
        uint16_t start_addr,
        uint16_t num_regs,
        uint8_t *output_buf,
        int *use_length
);
```

パラメーター

socket_id [in]	入力レジスタデータを読み取るためのソケット ID
start_addr [in]	読み取る入力レジスタデータの開始アドレス
num_regs [in]	読み取る入力レジスタ データの数。その最大値は 125 です。
output_buf [out]	読み取る入力レジスタデータを格納するバッファーへのポインター。
use_length [out]	「output_buf」の使用長を格納するバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char ip[15] = "169.254.188.17";
        int    socket_id;
        int    err = Modbus_ClientConnect(ip, &socket_id);
        //   Wait for client to connect to server
        Sleep(5000);
        if (!err) {
                uint8_t bufs[512];
                int len = 0;
                int addr = 30;
```

```
            int regs = 1;
            err = Modbus_ClientRead_InputReg(socket_id, addr, regs, bufs, &len);
            if(!err){
                for(int i = 0; i < len; i++){
                    Print("%x", bufs[i]);
                }
            }
        }
}
```

条件

サポートされる最小バージョン	iA Studio 2.0

21.3.5 Modbus_ClientRead_Coils

目的

サーバーの Modbus コイルデータを読み取るために、コントローラーをクライアントとして使用します。

構文

```
int Modbus_ClientRead_Coils(
        int socket_id,
        uint16_t start_addr,
        uint16_t num_coils,
        uint8_t *output_buf,
        int *use_length
);
```

パラメーター

socket_id [in]	Coils データを読み取るためのソケット ID
start_addr [in]	読み取るコイルデータの開始アドレス
num_coils [in]	読み取る Coils データの数。その最大値は 2000 です。
output_buf [out]	読み取る Coils データを格納するバッファーへのポインター。
use_length [out]	「output_buf」の使用長を格納するバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char ip[15] = "169.254.188.17";
        int    socket_id;
        int    err = Modbus_ClientConnect(ip, &socket_id);
        //   Wait for client to connect to server
        Sleep(5000);
        if (!err) {
                uint8_t bufs[512];
                int len = 0;
                int addr = 30;
```

```
            int coils = 40;
            err = Modbus_ClientRead_Coils(socket_id, addr, coils, bufs, &len);
            if(!err){
                for(int i = 0; i < len; i++){
                    Print("%x", bufs[i]);
                }
            }
        }
}
```

条件

サポートされる最小バージョン	iA Studio 2.0

21.3.6 Modbus_ClientRead_Inputs

目的

サーバーの Modbus ディスクリート入力データを読み取るために、コントローラーをクライアントとして使用します。

構文

```
int Modbus_ClientRead_Inputs(
        int socket_id,
        uint16_t start_addr,
        uint16_t num_inputs,
        uint8_t *output_buf,
        int *use_length
);
```

パラメーター

socket_id [in]	ディスクリート入力データを読み取るためのソケット ID
start_addr [in]	読み取るディスクリート入力データの開始アドレス。
num_inputs [in]	読み取るディスクリート入力データの数。その最大値は 2000 です。
output_buf [out]	読み取る離散入力データを格納するバッファーへのポインター。
use_length [out]	「output_buf」の使用長を格納するバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char ip[15] = "169.254.188.17";
        int    socket_id;
        int    err = Modbus_ClientConnect(ip, &socket_id);
        //   Wait for client to connect to server
        Sleep(5000);
        if (!err) {
                uint8_t bufs[512];
                int len = 0;
                int addr = 30;
```

```
            int inputs = 40;
            err = Modbus_ClientRead_Inputs(socket_id, addr, inputs, bufs, &len);
            if(!err){
                for(int i = 0; i < len; i++){
                    Print("%x", bufs[i]);
                }
            }
        }
}
```

条件

サポートされる最小バージョン	iA Studio 2.0

21.3.7 Modbus_ClientWrite_HoldReg

目的

コントローラーをクライアントとして使用し、Modbus 保持レジスタデータをサーバーに書き込みます。

構文

```
int Modbus_ClientWrite_HoldReg(
        int socket_id,
        uint16_t start_addr,
        uint16_t num_regs,
        uint16_t *write_data,
        uint8_t *output_buf,
        int *use_length
);
```

パラメーター

socket_id [in]	保持レジスタデータを書き込むソケット ID
start_addr [in]	書き込み対象の保持レジスタデータの開始アドレス。
num_regs [in]	書き込み対象の保持レジスタデータの数。その最大値は 123 です。
write_data [in]	書き込み対象の保持レジスタデータを格納するバッファーへのポインター。
output_buf [out]	サーバーから返信された保持レジスタデータを格納するバッファーへのポインター。
use_length [out]	「output_buf」の使用長を格納するバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char ip[15] = "169.254.188.17";
        int    socket_id;
        int    err = Modbus_ClientConnect(ip, &socket_id);
        //   Wait for client to connect to server
        Sleep(5000);
        if (!err) {
                uint8_t bufs[512];
```

```
            uint8_t data[1];
            //    Enable axis 0 in HIMC
            data[0] = 1;
            int len = 0;
            int addr = 30;
            int regs = 1;
            err = Modbus_ClientWrite_HoldReg(socket_id, addr, regs, data, bufs, &len);
            if(!err){
                for(int i = 0; i < len; i++){
                    Print("%x", bufs[i]);
                }
            }
        }
}
```

条件

サポートされる最小バージョン	iA Studio 2.0

21.3.8 Modbus_ClientWrite_Coils

目的

コントローラーをクライアントにして、Modbus Coils データをサーバーに書き込みます。

構文

```
int Modbus_ClientWrite_Coils(
        int socket_id,
        uint16_t start_addr,
        uint16_t num_coils,
        uint16_t *write_data,
        uint8_t *output_buf,
        int *use_length
);
```

パラメーター

socket_id [in]	Coils データを書き込むためのソケット ID
start_addr [in]	書き込むコイルデータの先頭アドレス。
num_coils [in]	書き込む Coils データの数。その最大値は 1968 です。
write_data [in]	書き込む Coils データを格納するバッファーへのポインター。
output_buf [out]	サーバーから返信された Coils データを格納するバッファーへのポインター。
use_length [out]	「output_buf」の使用長を格納するバッファーへのポインター。

戻りの値

関数が成功した場合は int 値 0 を返し、関数が失敗した場合はコントローラーのエラーコード(定義についてはセクション 18.1.1 を参照)を返します。

例

```
void main() {
        char ip[15] = "169.254.188.17";
        int   socket_id;
        int   err = Modbus_ClientConnect(ip, &socket_id);
        //   Wait for client to connect to server
        Sleep(5000);
        if (!err) {
                uint8_t bufs[512];
                uint8_t data[1];
```

```
            data[0] = 15;
            int len = 0;
            int addr = 30;
            int coils = 4;
            err = Modbus_ClientWrite_Coils(socket_id, addr, coils, data, bufs, &len);
            if(!err){
                for(int i = 0; i < len; i++){
                    Print("%x", bufs[i]);
                }
            }
        }
}
```

条件

サポートされる最小バージョン	iA Studio 2.0

22. 付録

22.1 数学定数

表 22.1.1

名称	説明	定義値
PI	円周の直径に対する比率	3.14159265358979323846
SQRT2	2 の平方根	1.41421356237309504880
SQRT1_2	2 の平方根の逆数、つまり 1/2 の平方根	0.70710678118654752440 1

22.2 システム変数

表 22.2.1

名称	タイプ	説明
system_timeInMs	int	HIMC のシステム時刻をミリ秒単位で格納する変数。
system_fclk	int	コントローラーサイクルごとに 1 ずつ増加する変数。
system_user_table[512000]	double	ユーザーの配列。永久メモリに保存できます。 セクション 11.6 SaveUserTable を参照してください。
system_ltest0 system_ltest1 ... system_ltest9	int	ユーザーの変数
system_dtest0 system_dtest1 ... system_dtest9	double	ユーザーの変数
system_mtest[10]	double	ユーザーの配列

22.3 ビット操作

変数内のビットを設定、クリア、反転、またはチェックする組み込み関数はありません。ただし、ユーザーはビット操作のために次のコードを HMPL タスクにコピーできます。

```
#define BIT_SET(a, idx)          ((a) |= (1<<(idx)))
#define BIT_CLEAR(a, idx)        ((a) &= ~(1<<(idx)))
#define BIT_FLIP(a, idx)         ((a) ^= (1<<(idx)))
#define BIT_CHECK(a, idx)        ((a) & (1<<(idx)))
```

例

```
#define BIT_SET(a, idx)          ((a) |= (1<<(idx)))
#define BIT_CLEAR(a, idx)        ((a) &= ~(1<<(idx)))
#define BIT_FLIP(a, idx)         ((a) ^= (1<<(idx)))
#define BIT_CHECK(a, idx)        ((a) & (1<<(idx)))

void main() {
    int bits_value = 0;

    BIT_SET(bits_value, 0);   //   now the value of bits_value is 1
    BIT_SET(bits_value, 3);   //   now the value of bits_value is 9
    BIT_CLEAR(bits_value, 0);   //   now the value of bits_value is 8
    BIT_FLIP(bits_value, 4);   //   now the value of bits_value is 24

    bits_value = 684;
    if (BIT_CHECK(bits_value, 5)) {
       Print("bit 5 is 1");
    } else {
       Print("bit 5 is 0");
    }
    //   the output is: bit 5 is 1
}
```

（このページはブランクになっています）

HIMC HMPL ユーザーガイド
バージョン：V1.1　　2025 年 3 月改訂

1．HIWIN は HIWIN Mikrosystem Corp., HIWIN Technologies Corp.，ハイウィン株式会社の登録商標です。ご自身の権利を保護するため、模倣品を購入することは避けてください。

2．実際の製品は、製品改良等に対応するため、このカタログの仕様や写真と異なる場合があります。

3．HIWIN は「貿易法」および関連規制の下で制限された技術や製品を販売・輸出しません。制限された HIWIN 製品を輸出する際には、関連する法律に従って、所管当局によって承認を受けます。また、核・生物・化学兵器やミサイルの製造または開発に使用することは禁じます。
