次世代ロボット知能化技術開発プロジェクト

ACT（中レベル）共通インターフェース仕様書

Ver. 1.0

2010年6月11日

作業サブ・ワーキンググループ

版	年　月	変 更 内 容
1.0	2010 年 6 月	新規作成

【 目 次 】

1 はじめに

本書は、次世代ロボット知能化技術開発プロジェクトの作業サブ WG において、中レベル ACT RTC（**エラー! 参照元が見つかりません。**参照）の共通インターフェース仕様を規定するものである。

ACT とは、具体的には6自由度あるいは7自由度を有するマニピュレータ及びその先端にエンドエフェクタとして取り付ける1軸グリッパのことを意味する。

本共通 I/F 規定することにより、マニピュレータに指令を出す上位モジュールは、機種が異なっても同一命令で制御することができるため、ハードウェアを差し替えた場合でも、ソフトウェアを再開発する必要がなくなるといったメリットが期待できる。

表 1　ACT インターフェースの 3 レベル

レベル	内　容
低レベル	関節単位の位置を直接指令できるインターフェース。
中レベル	関節座標において直線補間を行う PTP 命令や直交座標における直線補間を行う CP 命令を提供するインターフェース。
高レベル	JOB 実行を行うインターフェース。JOB とは中レベルのモーション命令を複数記述した記述したプログラムのこと。

本仕様と関連のあるドキュメントを以下に示す。

[1] ACT（低レベル）共通インターフェース仕様書

2 全体構成

中レベル ACT の RTC インターフェース構成を図 1 に示す。本 RTC は、2つのサービスポートで構成される。

共通コマンドサービスポートは、サーボ On/Off やステータス取得など、低レベル、中レベルの両方で必要とされるコマンドをまとめたサービスポートである。中レベル・モーションコマンド・サービスポートは、中レベルのモーションを実現するために必要なコマンドをまとめたサービスポートである。

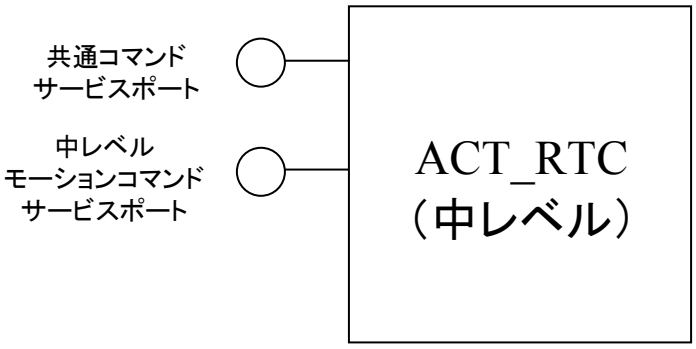

図 1　RTC インターフェース構成（中レベル）

図 2 に中レベル ACT RTC と低レベル ACT RTC を接続した例を示す。共通コマンドサービスポートは、低レベル RTC をラップし、中レベル ACT RTC のサービスポートとして上位モジュールへ提供する。

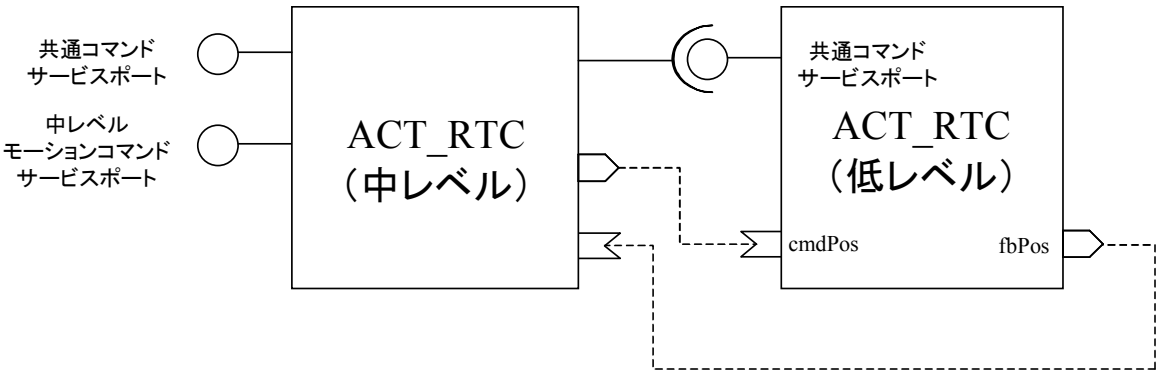

図 2　低レベル RTC との接続例

3 共通データ型

関連ドキュメント[1]の3章を参照のこと。

4 共通コマンド・サービスポートの仕様

関連ドキュメント[1]の4章を参照のこと。

5 中レベル・モーションコマンド・サービスポートの仕様

中レベル・モーションコマンド用サービスポートにおいて定義されるデータ型一覧を表 2 に、戻り値を表 3 に、インターフェースのオペレーション一覧を表 4 に示す。

本インターフェースを備えた RTC を使用する上位アプリケーションは、異なるマニピュレータであっても同一のアプリケーションで操作できることを期待する。しかし、表 4 の非互換オペレーションの使用には注意が必要である。マニピュレータを関節指令で動作させた場合、アーム先端の軌跡は機種ごとに異なり保証されないからである。

表 2 データ型一覧

No	データ型	概 要
1	HgMatrix	同次変換行列型
2	CarPosWithElobow	位置姿勢(同次変換行列)と肘角を有する構造体
3	CartesianSpeed	並進と回転の速度情報を有する構造体

表 3 戻り値一覧

値	戻り値名	概 要
0	OK	オペレーションを正常に受け付け
-1	NG	オペレーション拒否
-2	STATUS_ERR	オペレーションを受け付け可能な状態でない
-3	VALUE_ERR	引数が不正
-4	NOT_SV_ON_ERR	全ての軸のサーボが入っていない
-5	FULL_MOTION_QUEUE_ERR	バッファが一杯
・・・	(システム予約領域 0 ～ -9999)	-
・・・	(機種依存領域 -10000 以降)	-

表 4 中レベル・モーションコマンド用サービスポートのオペレーション

No	オペレーション名	概 要	非互換
1	closeGripper	グリッパを閉じる	
2	getBaseOffset	マニピュレータの設置位置を取得	
3	getFeedbackPosCartesian	直交座標系の位置フィードバック情報の取得	
4	getMaxSpeedCartesian	直交動作時の最大動作速度を取得	
5	getMaxSpeedJoint	関節動作時の最大動作速度を取得	
6	getMinAccelTimeCartesian	直交動作時の最小動作加速時間を取得	
7	getMinAccelTimeJoint	関節動作時の最小動作加速時間を取得	
8	getSoftLimitCartesian	直交座標系のソフトリミット値を取得	
9	moveGripper	グリッパの開閉動作	
10	moveLinearCartesianAbs	直交座標の直線補間(絶対指令)	
11	moveLinearCartesianRel	直交座標の直線補間(相対指令)	
12	movePTPCartesianAbs	関節座標の直線補間(直交・絶対指令)	
13	movePTPCartesianRel	関節座標の直線補間(直交・相対指令)	
14	movePTPJointAbs	関節座標の直線補間(関節・絶対指令)	×
15	movePTPJointRel	関節座標の直線補間(関節・相対指令)	×
16	openGripper	グリッパを開く	
17	pause	動作の一時停止	
18	resume	動作の再開	
19	stop	動作の停止	
20	setAccelTimeCartesian	直交動作時の加速時間を設定	
21	setAccelTimeJoint	関節動作時の加速時間を設定	

22	setBaseOffset	マニピュレータの設置位置を設定	
23	setControlPointOffset	制御点のフランジ面からのオフセット量を設定	
24	setMaxSpeedCartesian	直交動作時の最大動作速度を設定	
25	setMaxSpeedJoint	関節動作時の最大動作速度を設定	
26	setMinAccelTimeCartesian	直交動作時の最小動作加速時間を設定	
27	setMinAccelTimeJoint	関節動作時の最小動作加速時間を設定	
28	setSoftLimitCartesian	直交座標系のソフトリミット値を設定	
29	setSpeedCartesian	直交動作時の速度を設定	
30	SetSpeedJoint	関節動作時の速度を設定	

5.1 モジュール宣言

モジュール宣言は使用しない。

5.2 インターフェース宣言

インターフェース名は ManipulatorCommonInterface_Middle とする。

5.3 データ型

中レベル・モーションコマンド・サービスポートで定義されるデータ型について述べる。

5.3.1 HgMatrix

概要

同次変換行列型。

定義

```
typedef double HgMatrix [3][4];
```

備考

同次変換行列 4 x 4 の第4行を省略した 3 x 4 の行列。 座標系は、右手系とする。

5.3.2 CarPosWithElbow

概要

位置姿勢（同次変換行列）と肘角を有する構造体。

定義

```
struct CarPosWithElbow {
    HgMatrix carPos;
    double   elbow;
    ULONG    structFlag;
};
```

備考

structFlag は機種依存データである。詳細は各マニピュレータのドキュメントを参照のこと。

5.3.3 CartesianSpeed

概要

並進と回転の速度情報を有する構造体。

定義

```
struct CartesianSpeed {
    double translation;
    double rotation;
};
```

備考

なし。

5.4 オペレーション

中レベル・モーションコマンド・サービスポートで定義されるオペレーションについて述べる。

5.4.1 closeGripper

機能:
グリッパを完全に閉じる。

宣言:
RTC::RETURN_ID closeGripper();

引数:
なし

戻り値:

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
FULL_MOTION_QUEUE_ERR	失敗。バッファが一杯のためキューイング不可である。
NOT_SV_ON_ERR	失敗。すべての軸がサーボオン状態でない。
NG	失敗。上記以外の失敗。

備考:
グリッパの閉じた姿勢は、機種依存である。

5.4.2 getBaseOffset

機能:

アーム座標系からロボット座標系までのオフセット量を取得する。

宣言:

RTC::RETURN_ID getBaseOffset(out RTC::HgMatrix offset);

引数:

名前	入力・出力	説明
offset	出力	オフセット量

戻り値:

値	説明
OK	成功。
NG	失敗。オフセット量が準備できない。

備考:

図 3 を用いて、ベースオフセットの設定例を説明する。本例のロボットシステムは、右を向いたロボットAと左を向いたロボットBの2台から構成される。各ロボットのベース部には、右手系のアーム座標系が設定されている。ユーザは、ロボットBの座標系をロボットAの座標系に合わせて運転させたい。すなわちロボットAのアーム座標系を本ロボットシステムのロボット座標系としたい。この場合、ロボットBのアーム座標系から見たロボットAのアーム座標系までの位置・姿勢のオフセット量を、setBaseOffset オペレーションを使って設定する。

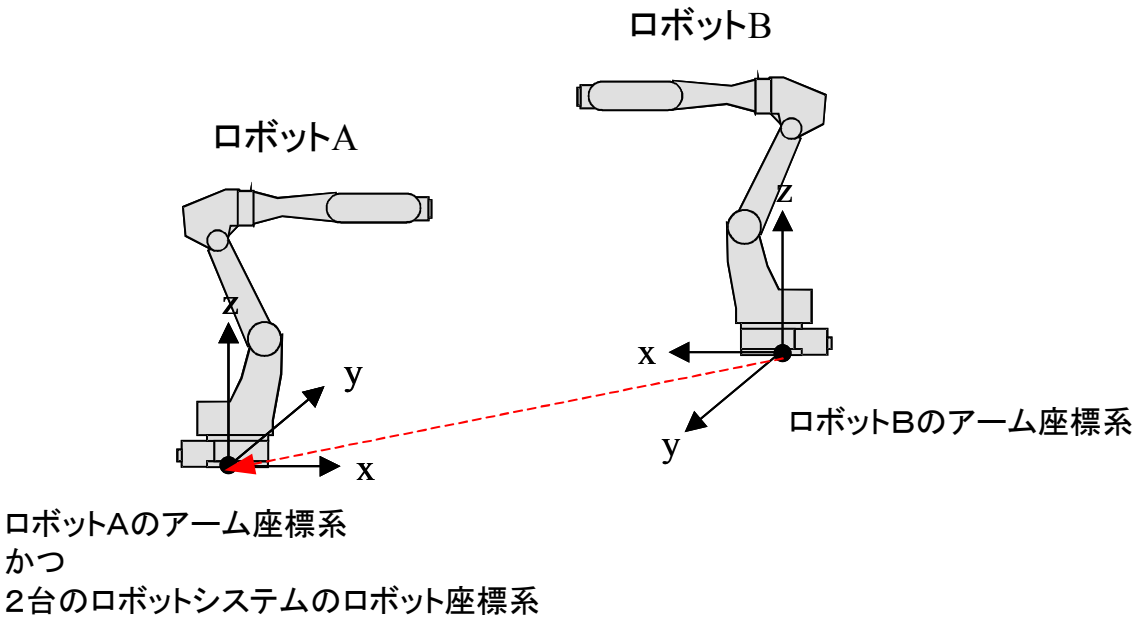

図 3 ベースオフセット

5.4.3 getFeedbackPosCartesian

機能:

ロボット座標系でのフィードバック位置情報を返す。

宣言:

RTC::RETURN_ID getFeedbackPosCartesian(out RTC::CarPosWithElbow pos);

引数:

名前	入力・出力	説明
pos	出力	位置フィードバック情報[mm, degree]

戻り値:

値	説明
OK	成功。
NG	失敗。フィードバック情報が準備できない。

備考:

6軸アームの場合は elbow は省略。

5.4.4 getMaxSpeedCartesian

機能:

直交動作時の最大動作速度を取得する。

宣言:

RTC::RETURN_ID getMaxSpeedCartesian(out RTC::CartesianSpeed speed);

引数:

名前	入力・出力	説明
Speed	出力	最大速度 最大並進速度 [mm/s] 最大回転速度 [degree/s]

戻り値:

値	説明
OK	成功。
NG	失敗。最大速度情報が準備できない。

備考:

setMaxSpeedCartesian オペレーションで設定した値を取得する。

5.4.5 getMaxSpeedJoint

機能:

関節動作時の最大動作速度を取得する。

宣言:

RTC::RETURN_ID getMaxSpeedJoint(out RTC::DoubleSeq speed);

引数:

名前	入力・出力	説明
Speed	出力	各軸の最大動作速度 [degree/s]

戻り値:

値	説明
OK	成功。
NG	失敗。最大動作速度が準備できない。

備考:

本値はモータ容量、ギア比、負荷といった条件から算出するものであり、機種依存である。

5.4.6 getMinAccelTimeCartesian

機能:

直交動作時の最大速度までの最小動作加速時間を取得する。

宣言:

RTC::RETURN_ID getMinAccelTimeCartesian(out double aclTime);

引数:

名前	入力・出力	説明
aclTime	出力	最小加速度時間 [s]

戻り値:

値	説明
OK	成功。
NG	失敗。最小動作加速時間が準備できない。

備考:

setMinAccelTimeCartesian オペレーションで設定した値を取得する。

5.4.7 getMinAccelTimeJoint

機能:

関節動作時の最大速度までの最小動作加速時間を取得する。

宣言:

RTC::RETURN_ID getMinAccelTimeJoint(out double aclTime);

引数:

名前	入力・出力	説明
aclTime	出力	最小加速度時間 [s]

戻り値:

値	説明
OK	成功。
NG	失敗。最小動作加速時間が準備できない。

備考:

本値はモータ容量、ギア比、負荷といった条件から算出するものであり、機種依存である。

5.4.8 getSoftLimitCartesian

機能:

ロボット座標系でのソフトリミット値を取得する。

宣言:

RTC::RETURN_ID getSoftLimitCartesian(out RTC::LimitValue xLimit, out RTC::LimitValue yLimit, out RTC::LimitValue zLimit);

引数:

名前	入力・出力	説明
xLimit	出力	x 軸のソフトリミット値 [mm]
yLimit	出力	y 軸のソフトリミット値 [mm]
zLimit	出力	z 軸のソフトリミット値 [mm]

戻り値:

値	説明
OK	成功。
NG	失敗。ソフトリミット値が準備できない。

備考:

本機能は、各 move 命令の動作において、アーム先端の制御点が本オペレーションで設定した範囲を超える場合は、動作を停止してアラームとする。

オペレーション setSoftLimitCartesian で設定した値を取得する。本 RTC 起動後、オペレーション setSoftLimitCartesian を1回も実行していない場合の値は、実装依存とする。

5.4.9 moveGripper

機能:

グリッパを指定された角度へ動作する。

宣言:

RTC::RETURN_ID moveGripper(in ULONG angleRatio);

引数:

名前	入力・出力	説明
Angle	入力	グリッパの開閉角度割合 [%] 0% : 完全に閉じた状態 100% : 完全に開いた状態

戻り値:

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
VALUE_ERR	失敗。設定値が不正である。
NOT_SV_ON_ERR	失敗。すべての軸がサーボオン状態でない。
NG	失敗。上記以外の要因で失敗。

備考:

なし

5.4.10 moveLinearCartesianAbs

機能:

直交空間において、目標位置をロボット座標系絶対値指定により、直線補間を動作する。

宣言:

RTC::RETURN_ID moveLinearCartesianAbs(in RTC::CarPosWithElbow carPoint);

引数:

名前	入力・出力	説明
carPoint	入力	絶対目標位置・姿勢 [mm, degree]

戻り値:

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
VALUE_ERR	失敗。設定値が不正である。
FULL_MOTION_QUEUE_ERR	失敗。バッファが一杯のためキューイング不可である。
NOT_SV_ON_ERR	失敗。すべての軸がサーボオン状態でない。
NG	失敗。上記以外の要因で失敗。

備考:

直線補間とは、直交空間における並進、回転の各動作が、全て同時起動同時停止に、また全ての加速時間と減速時間が同じになるよう軌跡生成する動作のことである。

6軸マニピュレータの場合、引数 carPoint のメンバ変数 elbow は無視される。

5.4.11 moveLinearCartesianRel

機能:

直交空間において、目標位置を相対直交座標指定により、直線補間を動作する。

宣言:

RTC::RETURN_ID moveLinearCartesianRel(in RTC::CarPosWithElbow carPoint);

引数:

名前	入力・出力	説明
carPoint	入力	相対目標位置・姿勢 [mm, degree]

戻り値:

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
VALUE_ERR	失敗。設定値が不正である。
FULL_MOTION_QUEUE_ERR	失敗。バッファが一杯のためキューイング不可である。
NOT_SV_ON_ERR	失敗。すべての軸がサーボオン状態でない。
NG	失敗。上記以外の要因で失敗。

備考:

直線補間とは、直交空間における並進、回転の各動作が、全て同時起動同時停止に、また全ての加速時間と減速時間が同じになるよう軌跡生成する動作のことである。

6軸マニピュレータの場合、引数 carPoint のメンバ変数 elbow は無視される。

5.4.12 movePTPCartesianAbs

機能:

関節空間において、目標位置をロボット座標系絶対値指定により、直線補間を動作する。

宣言:

RTC::RETURN_ID movePTPCartesianAbs(in RTC::CarPosWithElbow carPoint);

引数:

名前	入力・出力	説明
carPoint	入力	絶対目標位置・姿勢 [mm, degree]

戻り値:

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
VALUE_ERR	失敗。設定値が不正である。
FULL_MOTION_QUEUE_ERR	失敗。バッファが一杯のためキューイング不可である。
NOT_SV_ON_ERR	失敗。すべての軸がサーボオン状態でない。
NG	失敗。上記以外の要因で失敗。

備考:

直線補間とは、全軸同時起動同時停止に、また全軸の加速時間と減速時間が同じになるよう軌跡生成する動作のことである。

6軸マニピュレータの場合、引数 carPoint のメンバ変数 elbow は無視される。

5.4.13 movePTPCartesianRel

機能:

関節空間において、目標位置を相対直交座標指定により、直線補間を動作する。

宣言:

RTC::RETURN_ID movePTPCartesianRel(in RTC::CarPosWithElbow carPoint);

引数:

名前	入力・出力	説明
carPoint	入力	相対目標位置・姿勢 [mm, degree]

戻り値:

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
VALUE_ERR	失敗。設定値が不正である。
FULL_MOTION_QUEUE_ERR	失敗。バッファが一杯のためキューイング不可である。
NOT_SV_ON_ERR	失敗。すべての軸がサーボオン状態でない。
NG	失敗。上記以外の要因で失敗。

備考:

直線補間とは、全軸同時起動同時停止に、また全軸の加速時間と減速時間が同じになるよう軌跡生成する動作のことである。

6軸マニピュレータの場合、引数 carPoint のメンバ変数 elbow は無視される。

5.4.14 movePTPJointAbs

機能:

関節空間において、目標位置を絶対関節座標指定により、直線補間を動作する。

宣言:

RTC::RETURN_ID movePTPJointAbs(in RTC::JointPos jointPoints);

引数:

名前	入力・出力	説明
jointPoints	入力	絶対目標位置 [degree]

戻り値:

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
VALUE_ERR	失敗。設定値が不正である。
FULL_MOTION_QUEUE_ERR	失敗。バッファが一杯のためキューイング不可である。
NOT_SV_ON_ERR	失敗。すべての軸がサーボオン状態でない。
NG	失敗。上記以外の要因で失敗。

備考:

直線補間とは、全軸同時起動同時停止に、また全軸の加速時間と減速時間が同じになるよう軌跡生成する動作のことである。

引数 jointPoints 配列の値の順番は、J1、J2、J3、・・・ とする。

5.4.15 movePTPJointRel

機能:

関節空間において、目標位置を相対関節座標指定により、直線補間を動作する。

宣言:

RTC::RETURN_ID movePTPJointRel(in RTC:: JointPos jointPoints);

引数:

名前	入力・出力	説明
jointPoints	入力	相対目標位置 [degree]

戻り値:

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
VALUE_ERR	失敗。設定値が不正である。
FULL_MOTION_QUEUE_ERR	失敗。バッファが一杯のためキューイング不可である。
NOT_SV_ON_ERR	失敗。すべての軸がサーボオン状態でない。
NG	失敗。上記以外の要因で失敗。

備考:

直線補間とは、全軸同時起動同時停止に、また全軸の加速時間と減速時間が同じになるよう軌跡生成する動作のことである。

引数 jointPoints 配列の値の順番は、J1、J2、J3、・・・ とする。

5.4.16 openGripper

機能:

グリッパを完全に開く。

宣言:

RTC::RETURN_ID openGripper();

引数:

なし

戻り値:

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
VALUE_ERR	失敗。設定値が不正である。
FULL_MOTION_QUEUE_ERR	失敗。バッファが一杯のためキューイング不可である。
NOT_SV_ON_ERR	失敗。グリッパ軸がサーボオン状態でない。
NG	失敗。上記以外の要因で失敗。

備考:

グリッパの開いた姿勢は、機種依存である。

5.4.17 pause

機能:

マニピュレータのすべての軸を一時停止にする。

宣言:

RTC::RETURN_ID pause();

引数:

なし

戻り値:

値	説明
OK	成功。
NG	失敗。

備考:

軸が動作中の場合、減速停止する。一時停止状態である場合に他のモーション指令を実行しても、一時停止状態が解除されるまで動作しない。一時停止状態を解除する場合は resume オペレーションを使用する。
サーボ Off 中、アラーム中、一時停止中、停止中の本オペレーション要求は無視される。

5.4.18 resume

機能

動作を再開する。

宣言:

RTC::RETURN_ID resume();

引数:

なし

戻り値:

値	説明
OK	成功。
NG	失敗。

備考:

一時停止中のみ有効とし、他の状態の場合は全て無視される。

5.4.19 stop

機能:

すべての軸の動作を減速停止して、蓄積している他の命令も全て破棄する。

宣言:

RTC::RETURN_ID stop();

引数:

なし

戻り値:

値	説明
OK	成功。
NG	失敗。

備考:

動作中に、本オペレーションを受け付けた場合は、減速停止後に、蓄積している全てのモーション命令を破棄する。

一時停止状態中に、本オペレーションを受け付けた場合は、蓄積している全てのモーション命令を破棄する。一時停止状態も解除する。

サーボ Off 中、アラーム中の本オペレーション要求は無視される。

5.4.20 setAccelTimeCartesian

機能:

直交動作時の加速時間を設定する。

宣言:

RTC::RETURN_ID setAccelTimeCartesian(in double aclTime);

引数:

名前	入力・出力	説明
aclTime	入力	加速時間 [s]

戻り値:

値	説明
OK	成功。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記以外の要因で失敗。

備考:

setMinAccelTimeCartesian オペレーションで設定した値を下回る値を指定した場合、エラーとする。

5.4.21 setAccelTimeJoint

機能:

関節動作時の加速時間を設定する。

宣言:

RTC::RETURN_ID setAccelTimeJoint(in double aclTime);

引数:

名前	入力・出力	説明
aclTime	入力	加速時間 [s]

戻り値:

値	説明
OK	成功。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記以外の要因で失敗。

備考:

getMinAccelTimeJoint オペレーションで取得する値を下回る値を指定した場合、エラーとする。

5.4.22 setBaseOffset

機能:

本マニピュレータのアーム座標系からロボット座標系（基準）までのオフセット量を設定する。

宣言:

RTC::RETURN_ID setBaseOffset(in RTC::HgMatrix offset);

引数:

名前	入力・出力	説明
offset	入力	オフセット量

戻り値:

値	説明
OK	成功。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記以外の要因で失敗。

備考:

本機能の詳細は、5.4.2 項を参照のこと。 本 RTC 起動後、本オペレーションを1回も実行していない場合の値は、0 とする。

5.4.23 setControlPointOffset

機能:

制御点のフランジ面からのオフセット量を設定する。

宣言:

RTC::RETURN_ID setControlPointOffset(in RTC::HgMatrix offset);

引数:

名前	入力・出力	説明
offset	入力	オフセット量

戻り値:

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。動作中やアラーム発生中は設定できない。
NG	失敗。上記以外の要因で失敗。

備考:

なし。

5.4.24 setMaxSpeedCartesian

機能:

直交動作時の最大動作速度を設定する。

宣言:

RTC::RETURN_ID setMaxSpeedCartesian(in RTC::CartesianSpeed speed);

引数:

名前	入力・出力	説明
speed	入力	最大速度 最大並進速度 [mm/s] 最大回転速度 [degree/s]

戻り値:

値	説明
OK	成功。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記以外の要因で失敗。

備考:

本 RTC 起動後、本オペレーションを1回も実行していない場合の値は、実装依存とする。

5.4.25 setMaxSpeedJoint

機能:

関節動作時の最大動作速度を設定する。

宣言:

RTC::RETURN_ID setMaxSpeedJoint(in RTC::DoubleSeq speed);

引数:

名前	入力・出力	説明
Speed	入力	各軸の最大動作速度 [degree/s]

戻り値:

値	説明
OK	成功。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記以外の要因で失敗。

備考:

本 RTC 起動後、本オペレーションを1回も実行していない場合の値は、実装依存とする。

5.4.26 setMinAccelTimeCartesian

機能:

直交動作時の最大速度までの最小動作加速時間を設定する。

宣言:

RTC::RETURN_ID setMinAccelTimeCartesian(in double aclTime);

引数:

名前	入力・出力	説明
aclTime	入力	最小加速度時間 [s]

戻り値:

値	説明
OK	成功。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記以外の要因で失敗。

備考:

本 RTC 起動後、本オペレーションを1回も実行していない場合の値は、実装依存とする。

5.4.27 setMinAccelTimeJoint

機能:

関節動作時の最大速度までの最小動作加速時間を取得する。

宣言:

RTC::RETURN_ID setMinAccelTimeJoint(in double aclTime);

引数:

名前	入力・出力	説明
aclTime	入力	最小加速度時間 [s]

戻り値:

値	説明
OK	成功。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記以外の要因で失敗。

備考:

本 RTC 起動後、本オペレーションを1回も実行していない場合の値は、実装依存とする。

5.4.28 setSoftLimitCartesian

機能:

直交座標系のソフトリミット値を設定する。

宣言:

RTC::RETURN_ID setSoftLimitCartesian(in RTC::LimitValue xLimit, in RTC::LimitValue yLimit,
in RTC::LimitValue zLimit);

引数:

名前	入力・出力	説明
xLimit	入力	x 軸のソフトリミット値 [mm]
yLimit	入力	y 軸のソフトリミット値 [mm]
zLimit	入力	z 軸のソフトリミット値 [mm]

戻り値:

値	説明
OK	成功。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記以外の要因で失敗。

備考:

本 RTC 起動後、本オペレーションを1回も実行していない場合の値は、実装依存とする。 本機能(直交のリミット)と関節座標系のリミット(低レベルRTC)は同時に機能する。

5.4.29 setSpeedCartesian

機能:

直交動作時の速度を%指定する。

宣言:

RTC::RETURN_ID setSpeedCartesian(in RTC::ULONG spdRatio);

引数:

名前	入力・出力	説明
spdRatio	入力	最大速度に対する割合指定 [%] 注）100%を上限とする 初期値:0%

戻り値:

値	説明
OK	成功。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記以外の要因で失敗。

備考:

なし。

5.4.30 setSpeedJoint

機能:

関節動作時の速度を%指定する。

宣言:

RTC::RETURN_ID setSpeedJoint(in RTC::ULONG spdRatio);

引数:

名前	入力・出力	説明
spdRatio	入力	最大速度に対する割合指定 [%] 注）100%を上限とする 初期値は0とする。

戻り値:

値	説明
OK	成功。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記以外の要因で失敗。

備考:

なし

6 付録 IDL

本RTCの IDL を以下に示す。

```
/*
Manipulator Common Interface (Middle Level Commands)
          - This IDL is used as service port on RTC
      - This command specification is provided by Intelligent RT Software
                  Project of NEDO.
rev. 20100318
*/

#ifndef MANIPULATORCOMMONINTERFACE_MIDDLE_IDL
#define MANIPULATORCOMMONINTERFACE_MIDDLE_IDL

#include "ManipulatorCommonInterface_DataTypes.idl"

module RTC
{

   typedef double HgMatrix [3][4];

   struct CarPosWithElbow {
      HgMatrix carPos;
      double     elbow;
      ULONG    structFlag;
   };

   struct CartesianSpeed {
      double translation;
      double rotation;
   };

};

interface ManipulatorCommonInterface_Middle
{

   RTC::RETURN_ID closeGripper();

   RTC::RETURN_ID getBaseOffset(out RTC::HgMatrix offset);

   RTC::RETURN_ID getFeedbackPosCartesian(out RTC::CarPosWithElbow pos);

   RTC::RETURN_ID getMaxSpeedCartesian(out RTC::CartesianSpeed speed);

   RTC::RETURN_ID getMaxSpeedJoint(out RTC::DoubleSeq speed);
```

```
RTC::RETURN_ID getMinAccelTimeJoint(out double aclTime);

RTC::RETURN_ID getSoftLimitCartesian(out RTC::LimitValue xLimit,

                                     out RTC::LimitValue yLimit,
                                         out RTC::LimitValue zLimit );

RTC::RETURN_ID moveGripper(in RTC::ULONG angleRatio);

RTC::RETURN_ID moveLinearCartesianAbs(in RTC::CarPosWithElbow carPoint);

RTC::RETURN_ID moveLinearCartesianRel(in RTC::CarPosWithElbow carPoint);

RTC::RETURN_ID movePTPCartesianAbs(in RTC::CarPosWithElbow carPoint);

RTC::RETURN_ID movePTPCartesianRel(in RTC::CarPosWithElbow carPoint);

RTC::RETURN_ID movePTPJointAbs(in RTC::JointPos jointPoints);

RTC::RETURN_ID movePTPJointRel(in RTC::JointPos jointPoints);

RTC::RETURN_ID openGripper();

RTC::RETURN_ID pause();

RTC::RETURN_ID resume();

RTC::RETURN_ID stop();

RTC::RETURN_ID setAccelTimeCartesian(in double aclTime);

RTC::RETURN_ID setAccelTimeJoint(in double aclTime);

RTC::RETURN_ID setBaseOffset(in RTC::HgMatrix offset);

RTC::RETURN_ID setControlPointOffset(in RTC::HgMatrix offset);

RTC::RETURN_ID setMaxSpeedCartesian(in RTC::CartesianSpeed speed);

RTC::RETURN_ID setMaxSpeedJoint(in RTC::DoubleSeq speed);

RTC::RETURN_ID setMinAccelTimeCartesian(in double aclTime);

RTC::RETURN_ID setMinAccelTimeJoint(in double aclTime);

RTC::RETURN_ID setSoftLimitCartesian(in RTC::LimitValue xLimit,
                                     in RTC::LimitValue yLimit,
                                     in RTC::LimitValue zLimit);

RTC::RETURN_ID setSpeedCartesian(in RTC::ULONG spdRatio);
```