アームユニット RTC Ver.3.0
インターフェース仕様書

資料番号：RB1400020
改版：第１版

版	年　月	変 更 内 容
初版	2011/04/08	初版作成

【 目 次 】

はじめに

本書は、アームユニットのインターフェース仕様書である。アームユニットのアプリケーションソフトウェアはユニット内に配置された小形MC上に実装され、本書に記述する RTC インターフェースを使用して上位から指令を受けることにより動作する。

本インターフェースが対象とする実ユニットは、SmartPalⅣタイプアームと SmartPalⅤタイプアームの２種類とする。

1　全体構成

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と２つのデータポートから構成される。

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。

位置指令・データポートは、各関節の位置指令データを入力するためのポートである。位置 FB・データポートは、各関節のフィードバック位置データを出力するためのポートである。

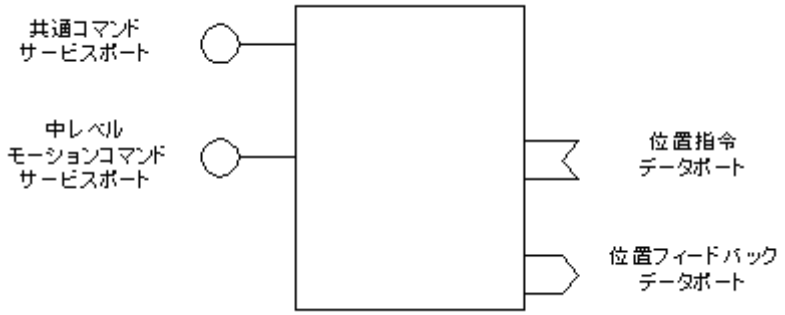

図 1 RTC インターフェース構成

2　インターフェース仕様

本 RTC は、２つのサービスポート(共通コマンド、中レベルモーションコマンドと２つのデータポートから構成される。

2.1　共通データ型

本 RTC で共通に使用するデータ型一覧を表 1 に示す。

表 1　共通データ型一覧

No	データ型	概　要
1	DoubleSeq	double のシーケンス型
2	JointPos	関節座標値を表すシーケンス型
3	LimitValue	上下限の制限値を有する構造体
4	RETURN_ID	リターン情報を有する構造体型
5	TimedJointPos	タイムスタンプ付きの関節座標値を有する構造体
6	ULONG	unsigned long の短縮形

2.1.1 データ型

本 RTC で共通に定義されるデータ型について述べる。

2.1.1.1 DoubleSeq

概要

基本データ型 double のシーケンス型。

定義

```
typedef   sequence<double>      DoubleSeq;
```

備考

なし。

2.1.1.2 JointPos

概要

関節座標値を表すシーケンス型。

定義

```
typedef   sequence<double>      JointPos;
```

備考

なし。

2.1.1.3 LimitValue

概要

上下限の制限値を有する構造体。

定義

```
struct LimitValue {
  double upper;
  double lower;
};
```

備考

なし。

2.1.1.4 RETURN_ID

概要

リターン情報を有する構造体。

定義

```
struct RETURN_ID {
  long id;
  string comment;
};
```

備考

リターンコードの詳細は、表 3 を参照のこと。

2.1.1.5 TimedJointPos

概要

タイムスタンプ付きの関節座標値を有する構造体。

定義

```
struct TimedJointPos {
    Time tm;
    JointPos pos;
};
```

備考

なし。

2.1.1.6 ULONG

概要

基本データ型 unsigned long の短縮形。

定義

```
typedef    unsigned long ULONG;
```

備考

なし。

2.2 共通コマンド

共通コマンド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、戻り値一覧を表 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。

表 2 データ型一覧

No	データ型	概 要
1	AlarmType	アラームの種別を表す列挙型
2	Alarm	アラーム情報を格納する構造体
3	AlarmSeq	Alarm のシーケンス型
4	LimitSeq	LimitValue のシーケンス型
5	ManipInfo	マニピュレータ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

表 4 共通コマンド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
1	clearAlarms	アラームクリア
2	getActiveAlarm	アラーム情報の取得
3	getFeedbackPosJoint	関節座標系の位置フィードバック情報の取得
4	getManipInfo	マニピュレータ情報の取得
5	getSoftLimitJoint	関節座標系のソフトリミット値を取得
6	getState	ユニットの状態取得
7	servoOFF	全軸サーボ OFF
8	servoON	全軸サーボ ON
9	setSoftLimitJoint	関節座標系のソフトリミット値設定

2.2.1 モジュール宣言

モジュール宣言は使用しない。

2.2.2 インターフェース宣言

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

2.2.3 データ型

共通コマンドに定義されるデータ型について述べる。

2.2.3.1 AlarmType

概要

アラームの種別を表す列挙型。

定義

```
enum AlarmType
{
        FAULT = 0,
        WARNING,
        UNKNOWN
};
```

備考

なし。

2.2.3.2 Alarm

概要

アラーム情報を格納する構造体。

定義

```
struct Alarm
{
        unsigned long code;
        AlarmType type;
        string description;
};
```

備考

アラームコードの一覧を表 5 に示す。

表 5　アラームコード一覧表

アラームコード	説明
0x00000001	非常停止ボタン押下
0x00000002	過負荷
0x00000003	オーバースピード
0x00000004	ソフトリミットオーバ（関節座標）
0x00000005	ソフトリミットオーバ（直交座標）
・・・	（システム予約領域 0x00000006 ～　0x000003FF）
0x000003FF	
0x00000400	

・・・ 0xFFFFFFFF	（機種依存領域　　0x00000400 ～　0xFFFFFFFF）

2.2.3.3 AlarmSeq

概要

Alarm のシーケンス型。

定義

typedef sequence<Alarm> AlarmSeq;

備考

なし。

2.2.3.4 LimitSeq

概要

LimitValue のシーケンス型。

定義

typedef sequence<LimitValue> LimitSeq;

備考

なし。

2.2.3.5 ManipInfo

概要

マニピュレータ情報を有する構造体。

定義

```
struct ManipInfo {
  string manufactur;
  string type;
  ULONG   axisNum;
  ULONG   cmdCycle;
   boolean   isGripper;
};
```

備考

manufactur：　メーカ名
type：　機種名
axisNum：　軸数（グリッパを除く）
cmdCycle：　低レベル位置指令を受ける周期
isGripper：　１軸グリッパの有無（グリッパ未装着時及び多指ハンド装着時は、false とする）

2.2.4 オペレーション

共通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。

2.2.4.1 clearAlarms

機能：

すべてのアラームのクリアを実行する。

宣言：

RTC::RETURN_ID clearAlarms();

引数：

なし。

戻り値：

値	説明
OK	成功。アラームクリア実行の結果、アラームがクリアできなくてもよい。
NG	失敗。アラームクリア実行不可能。

備考：

なし

2.2.4.2 getActiveAlarm

機能：

発生中のアラーム情報を取得する。

宣言：

RTC::RETURN_ID getActiveAlarm(out RTC::AlarmSeq alarms);

引数：

名前	入力・出力	説明
alarms	出力	アラーム情報の配列（シーケンス型）

戻り値：

値	説明
OK	成功。
NG	失敗。アラーム情報が準備できない。

備考：

アラームなしの場合は、引数 alarms はサイズ 0 の double シーケンスとする。
アラームが N 個の場合は、引数 alarms のサイズは N となる。

2.2.4.3 getFeedbackPosJoint

機能：

各軸のフィードバック位置情報を返す。

宣言：

RTC::RETURN_ID getFeedbackPosJoint(out RTC::JointPos 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（シーケンス型）[degree]

戻り値：

値	説明
OK	成功。
NG	失敗。フィードバック位置情報が準備できない。

備考：

引数 pos 配列の値の順番は、J1、J2、J3、J4、J5、J6、J7、gripper　とする。（ただし、グリッパ未装着の場合、gripper は不定値）

2.2.4.4 getManipInfo

機能：

マニピュレータ情報を取得する。

宣言：

RTC::RETURN_ID getManipInfo(out RTC::ManipInfo manipInfo);

引数：

名前	入力・出力	説明
manipInfo	出力	マニピュレータ情報

戻り値：

値	説明
OK	成功。
NG	失敗。マニピュレータ情報が準備できない。

備考：

なし。

2.2.4.5 getSoftLimitJoint

機能：

関節座標系のソフトリミット値を取得する。

宣言：

RTC::RETURN_ID getSoftLimitJoint(out RTC::LimitSeq softLimit);

引数：

名前	入力・出力	説明
softLimit	出力	各軸のソフトリミット値 [degree]

戻り値：

値	説明
OK	成功。
NG	失敗。マニピュレータ情報が準備できない。

備考：

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

格納されるデータ数は７(アームの軸数)+１(グリッパ)とする。
格納の順番は以下とする。

データ位置	格納データ
0	１軸目のソフトリミット値[degree]
1	２軸目の〃
2	３軸目の〃
3	４軸目の〃
4	５軸目の〃
5	６軸目の〃
6	７軸目の〃
7	グリッパの〃（グリッパ未装着時は不定値）

2.2.4.6 getState

機能：
ユニットの状態を取得する。

宣言：
RTC::RETURN_ID getState(out RTC::ULONG state);

引数：

名前	入力・出力	説明
state	出力	ユニットの状態を表すビットコード

戻り値：

値	説明
OK	成功。
NG	失敗。ユニットの状態情報が準備できない。

備考：
状態ビットの一覧を表 6 に示す。

表 6　状態ビットの一覧

状態ビット	説明
0x01	サーボ On 中
0x02	動作中
0x04	アラーム発生中
0x08	Move 命令のバッファがフル
0x10	一時停止中

2.2.4.7 servoOFF

機能：
マニピュレータ及びグリッパすべての軸をサーボオフする。

宣言：
RTC::RETURN_ID servoOFF();

引数：
なし

戻り値：

値	説明
OK	成功。すべての軸がサーボオフ状態。
NG	失敗。

備考：
処理が正常に終了し、全ての軸のサーボ制御がオフ状態になった場合、状態ビット 0x02 が 0 となる。

2.2.4.8 servoON

機能：
マニピュレータ及びグリッパすべての軸をサーボオンする。

宣言：
RTC::RETURN_ID servoON();

引数：
なし

戻り値：

値	説明
OK	成功。すべての軸がサーボオン状態。
NG	失敗。

備考：
処理が正常に終了し、全ての軸のサーボ制御がオン状態になった場合、状態ビット 0x02 が 1 となる。

2.2.4.9 setSoftLimitJoint

機能：
関節座標系のソフトリミット値を設定する。

宣言：
RTC::RETURN_ID setSoftLimitJoint(in RTC::LimitSeq softLimit);

引数：

名前	入力・出力	説明
softLimit	入力	各軸のソフトリミット値 [degree]

戻り値：

値	説明
OK	成功。
STATUS_ERR	失敗。設定可能状態でない。
VALUE_ERR	失敗。設定値が不正である。
NG	失敗。上記要因以外で失敗。

備考：
引数 softLimit のサイズは、マニピュレータの軸数に対応する。
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。
本オペレーションの実行は、動作中、アラーム発生中は拒否される。
格納されるデータ数は接続される軸数（7(アームの軸数)+１(グリッパ)の場合８）とする。軸数よりデータ数が多い場合、過多の情報を無視し、正常終了する。軸数より少ない場合は、戻り値に VALUE_ERR を返す。格納の順番は以下とする。

データ位置	格納データ
0	1軸目のソフトリミット値[degree]
1	2軸目の〃
2	3軸目の〃
3	4軸目の〃
4	5軸目の〃
5	6軸目の〃
6	7軸目の〃
7	グリッパの〃（グリッパ未装着時は不定値）

2.3　中レベルモーションコマンド

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 7 に、戻り値を表 8 に、インターフェースのオペレーション一覧を表 9 に示す。

表 7　データ型一覧

No	データ型	概　要
1	HgMatrix	同次変換行列型
2	CarPosWithElbow	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 8　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

表 9　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
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

2.3.1 モジュール宣言

モジュール宣言は使用しない。

2.3.2 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。

2.3.3 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。

2.3.3.1 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。

2.3.3.2 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。本 RTC では不定値とする。

位置姿勢（同次変換行列）とは、x、y、z、rx、ry、rz を同次変換行列に変換したものである。単位は、x、y、z は[mm]、rx、ry、rz は[degree]とする。

2.3.3.3 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。

2.3.4 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。

2.3.4.1 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。

2.3.4.2 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。上記以外の失敗。

備考：

図 2 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。

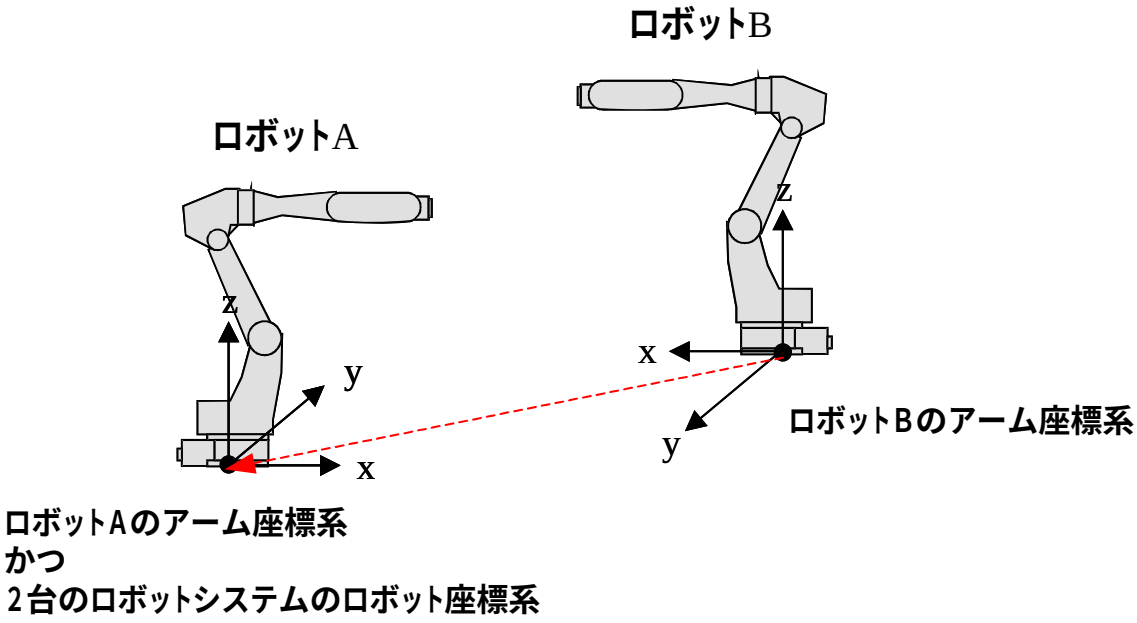

図 2　ベースオフセット

2.3.4.3 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。

備考：

なし

2.3.4.4 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。

2.3.4.5 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。

2.3.4.6 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。

2.3.4.7 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。

2.3.4.8 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。

2.3.4.9 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

2.3.4.10 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直交座標指定により、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。

速度は、setMaxSpeedCartesian で設定された最大速度×setSpeedCartesian で設定された値[%]となる。

加速時間は、setAccelTimeCartesian で設定された時間となる。

2.3.4.11 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。

速度は、setMaxSpeedCartesian で設定された最大速度×setSpeedCartesian で設定された値[%]となる。

加速時間は、setAccelTimeCartesian で設定された時間となる。

2.3.4.12 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直交座標指定により、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。

速度は、setMaxSpeedJoint で設定された中で、動作角度が一番大きい軸の値(現在の角度からではなく、最後に指令した角度)×setSpeedJoint で設定された値[%]となる。

加速時間は、setAccelTimeJoint で設定された時間となる。

2.3.4.13 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。

速度は、setMaxSpeedJoint で設定された中で、動作角度が一番大きい軸の値(現在の角度からではなく、最後に指令した角度)×setSpeedJoint で設定された値[%]となる。

加速時間は、setAccelTimeJoint で設定された時間となる。

2.3.4.14 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。グリッパは動作の対象外の為、引数に

含めない。
　アームの軸数と比較しデータ数が多い場合、過多の情報は無視して処理される。少ない場合は、戻り値にVALUE_ERR を返す。
　速度は、setMaxSpeedJoint で設定された中で、動作角度が一番大きい軸の値(現在の角度からではなく、最後に指令した角度)×setSpeedJoint で設定された値[%]となる。
　加速時間は、setAccelTimeJoint で設定された時間となる。

2.3.4.15　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。グリッパは動作の対象外の為、引数に含めない。
　アームの軸数と比較しデータ数が多い場合、過多の情報は無視して処理される。少ない場合は、戻り値にVALUE_ERR を返す。
　速度は、setMaxSpeedJoint で設定された中で、動作角度が一番大きい軸の値(現在の角度からではなく、最後に指令した角度)×setSpeedJoint で設定された値[%]となる。
　加速時間は、setAccelTimeJoint で設定された時間となる。

2.3.4.16　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。

2.3.4.17 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、一時停止中の本オペレーション要求は無視される。

2.3.4.18 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。

2.3.4.19 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。

2.3.4.20 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。

2.3.4.21 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。

2.3.4.22 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、2.3.4.2 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。

2.3.4.23 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。

2.3.4.24 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。

2.3.4.25 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。
軸数と比較してデータ数が多い場合は、過多のデータを無視して処理する。少ない場合は、戻り値に VALUE_ERR を返す。

2.3.4.26 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。

2.3.4.27 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。

2.3.4.28 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。

2.3.4.29 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。

2.3.4.30 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：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

2.4 位置指令データポート（InPort）

機能：
マニピュレータの各関節へ角度指令を入力する。

ポート情報：

ポート名	データ型	データ長	説　明
cmdPos	TimedJointPos	可変	各軸の関節角度指令値 [degree]

注）データ長は、アーム軸数＋グリッパ軸数（１軸）　とする。（ただし、アーム軸数およびグリッパ有無の情報は、getManipInfo オペレーションで取得可能）

データフォーマット：

データ位置	格納データ
0	１軸目の関節角度指令データ
1	２軸目の〃
2	３軸目の〃
3	４軸目の〃
4	５軸目の〃
5	６軸目の〃
6	７軸目の〃
7	グリッパの〃（グリッパ未装着時は不定値）

制約：
なし

備考：
本ポートは連続した位置指令を上位モジュールから受信するため、SyncFIFO 型のバッファ指定を使用すること。
データを受信する周期（機種依存）は、getManipInfo オペレーションで取得することが可能である。上位モジュールはこの周期に対応した位置指令データを準備すること。

2.5　位置 FB 指令データポート（OutPort）

機能：

マニピュレータの各関節のフィードバック角度データを出力する。

ポート情報：

ポート名	データ型	データ長	説　明
fbPos	TimedJointPos	可変	各軸の関節角度フィードバック値 [degree]

注）データ長は、アーム軸数＋グリッパ軸数（１軸）　とする。（ただし、アーム軸数およびグリッパ有無の情報は、getManipInfo オペレーションで取得可能）

データフォーマット：

データ位置	格納データ
0	１軸目の関節角度フィードバックデータ
1	２軸目の〃
2	３軸目の〃
3	４軸目の〃
4	５軸目の〃
5	６軸目の〃
6	７軸目の〃
7	グリッパの〃（グリッパ未装着時は不定値）

制約：

なし

備考：

本ポートは最新の位置フィードバック値を出力するため、NullBuffer 型のバッファ指定を使用すること。
データを出力する周期（機種依存）は、getManipInfo オペレーションで取得することが可能である。

付録　IDL
IDL を以下に示す。

```
        #ifndef ARMUNITRTC_IDL_
        #define ARMUNITRTC_IDL_

        module RTC
        {
           typedef sequence<double> DoubleSeq;
           typedef sequence<double> JointPos;
           struct LimitValue {
              double upper;
              double lower;
           };
           struct RETURN_ID
           {
              long id;
        /**< エラーID                                    */
              string comment;
        /**< エラーコード                                */
           };
           struct TimedJointPos {
            Time tm;
            JointPos pos;
           };
           typedef unsigned long ULONG;
        };
#endif

        #ifndef MANIPULATORCOMMONINTERFACE_COMMON_IDL
        #define MANIPULATORCOMMONINTERFACE_COMMON_IDL

        module RTC
        {
           enum AlarmType {
              FAULT,
              WARNING,
              UNKNOWN
           };

           struct Alarm {
              unsigned long code;
              AlarmType type;
              string description;
           };
```

```
typedef sequence<Alarm> AlarmSeq;
  typedef sequence<LimitValue> LimitSeq;

  struct ManipInfo {
    string manufactur;
    string type;
    ULONG   axisNum;
    ULONG   cmdCycle;
    boolean isGripper;
  };

  const ULONG CONST_BINARY_00000001 = 0x01;              /* isServoOn      */
  const ULONG CONST_BINARY_00000010 = 0x02;              /* isMoving       */
  const ULONG CONST_BINARY_00000100 = 0x04;              /* isAlarmed      */
  const ULONG CONST_BINARY_00001000 = 0x08;              /* isBufferFull */
};

interface ManipulatorCommonInterface_Common
{
  RTC::RETURN_ID clearAlarms();
  RTC::RETURN_ID getActiveAlarm(out RTC::AlarmSeq alarms);
  RTC::RETURN_ID getFeedbackPosJoint(out RTC::JointPos pos);
  RTC::RETURN_ID getManipInfo(out RTC::ManipInfo manipInfo);
  RTC::RETURN_ID getSoftLimitJoint(out RTC::LimitSeq softLimit);
  RTC::RETURN_ID getState(out RTC::ULONG state);
  RTC::RETURN_ID servoOFF();
  RTC::RETURN_ID servoON();
  RTC::RETURN_ID setSoftLimitJoint(in RTC::LimitSeq softLimit);
};

#ifndef MANIPULATORCOMMONINTERFACE_MIDDLE_IDL
#define MANIPULATORCOMMONINTERFACE_MIDDLE_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 getMinAccelTimeCartesian(out double aclTime);
      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);
      RTC::RETURN_ID setSpeedJoint(in RTC::ULONG spdRatio);

    };

#endif // MANIPULATORCOMMONINTERFACE_MIDDLE_IDL
```