

ロボットアーム制御機能共通インタフェース仕様書

(第 1.1 版 草案)

NEDO 次世代ロボット智能化技術開発プロジェクト

埼玉大学工学部機械工学科 設計工学研究室

2013 年 10 月 28 日



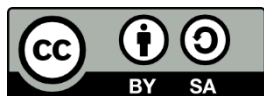
【改版履歴】

日付	版番号	改版ページ	改版内容
2012.2.24	1.0	全ページ	新規作成 (NEDO)
2013.10.28	1.1		改版草案作成

【本書の利用にあたって】

本書は、クリエイティブ・コモンズ表示 2.1 ライセンスの下に提供される。

(<http://creativecommons.org/licenses/by-sa/2.1/jp/>)



【本書の策定メンバー】

(敬称略、五十音順)

足立勝	(株式会社安川電機)
小笠原哲也	(東京大学大学院 情報理工学系研究科 知能機械情報学専攻)
河井良浩	(独立行政法人産業技術総合研究所 知能システム研究部門 タスクビジョン研究グループ)
中本啓之	(株式会社セック 開発本部 第四開発部)
二宮恒樹	(富士ソフト株式会社 ロボット事業グループ 商品開発ユニット)
野田哲男	(三菱電機株式会社)
原田研介	(独立行政法人産業技術総合研究所)
米澤浩	(I D E C株式会社)

(所属は 2012 年 2 月 24 日現在)

【目次】

1	はじめに	7
1.1	対象機能の概要	7
1.2	標準システム構成	8
2	本書を読む上での注意	9
2.1	基本方針	9
2.2	フォーマットと表現方法	9
2.2.1	列挙型定義	9
2.2.2	型定義	9
2.2.3	インタフェース定義	9
2.3	本仕様書における前提条件	10
2.3.1	座標系定義について	10
3	名前空間定義	11
4	データ型定義	11
4.1	標準型	11
4.1.1	RTC::Time	11
4.2	型宣言	11
4.2.1	DoubleSeq	11
4.2.2	JointPos	11
4.2.3	ULONG	11
4.2.4	AlarmSeq	11
4.2.5	LimitSeq	11
4.2.6	HgMatrix	11
4.3	ロボットアーム制御機能用	12
4.3.1	LimitValue	12
4.3.2	RETURN_ID	12
4.3.3	TimedJointPos	12
4.3.4	AlarmType	13
4.3.5	Alarm	13
4.3.6	Manipinfo	14
4.3.7	CarPosWithElbow	14
4.3.8	CartesianSpeed	14
5	共通インタフェース定義	15
5.1	データポート	15
5.1.1	位置指令インタフェース	15
5.1.2	位置フィードバック指令インタフェース	15

5.2 サービスポート	16
5.2.1 ManipulatorCommonInterface_Common	16
5.2.2 ManipulatorCommonInterface_Middle	17
6 CORBA IDL	24
6.1 ManipulatorCommonInterface_DataTypes.idl	24
6.2 ManipulatorCommonInterface_Common.idl	25
6.3 ManipulatorCommonInterface_Middle.idl	26

1 はじめに

近年、ロボットの開発を効率化するためにコンポーネントベースのミドルウェア開発が盛んになっている。コンポーネントベースのミドルウェア開発において、インタフェースの共通化は、コンポーネントの相互接続性や相互運用性を確保するうえで非常に重要である。このような背景に基づき、本書では、ロボットアーム制御機能に関わるインタフェースの共通仕様を定義する。

本共通インタフェースを規定することにより、使用するマニピュレータの機種が異なっても、ロボットアームに指示を出す上位モジュールは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に、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。

1.1 対象機能の概要

本仕様書では、1～3 自由度の直交座標型、4 自由度の水平多関節型、5～7 自由度の垂直多関節型の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1 軸グリップを取り付けたロボットアームを制御するための共通インタフェース(ACT インタフェース)を規定している。

ロボットアーム制御機能共通インタフェースを実装した RT コンポーネントの使用シーンの一例を以下に示す。

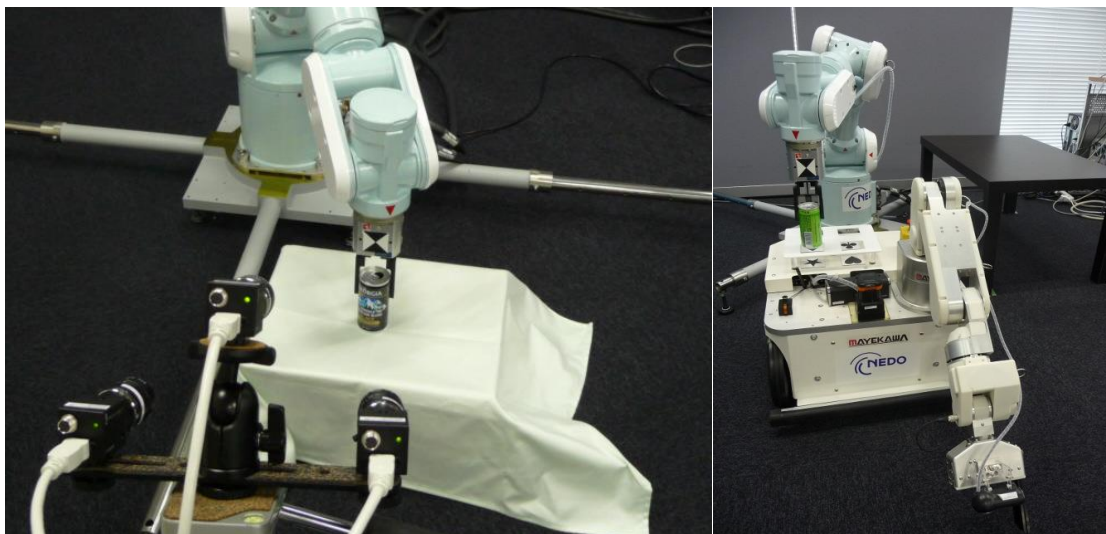
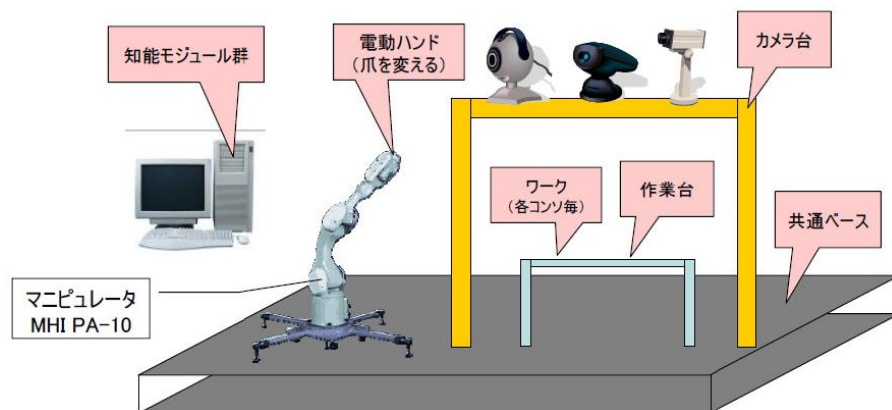


図 1.1 ロボットアーム制御機能共通インタフェースの使用シーン例

1.2 標準システム構成

ACT インタフェースでは、指令の抽象度に応じて以下の 3 レベルを想定している。

表 1.1 ACT インタフェースの 3 レベル

レベル	内容
低レベル	関節単位的位置を直接指令できるインタフェース
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フェース
高レベル	JOB 実行を行うインタフェース。JOB とは中レベルのモーション命令を複数記述したプログラムのこと。

ロボットアーム制御機能共通インタフェースを利用した標準的なシステム構成例を以下に示す。

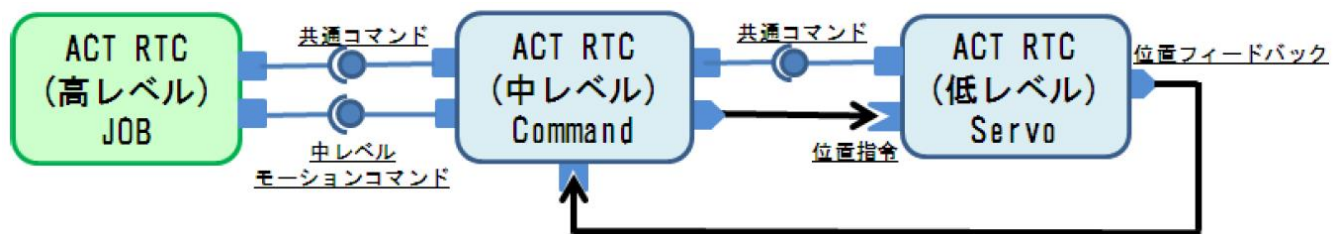


図 1.2 ロボットアーム制御機能共通インタフェースを使用したシステム例

本仕様書中では、上図中の「ACT RTC(中レベル)Command」と「ACT RTC(低レベル) Servo」を対象としており、これらのモジュールがやり取りするデータの形式および動作指示を与えるためのインタフェースを規定している。ただし、低レベルにおける指令モードとしては、位置指令のみを対象としており、速度指令やトルク指令は対象外としている。また、「ACT RTC(高レベル) JOB」のインタフェースも対象外である。

上図において、共通コマンドは、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ものである。また、位置指令は、各関節の位置指令データをやり取りするための情報であり、位置フィードバックは、各関節の角度情報をフィードバックするためのデータである。

2 本書を読む上での注意

2.1 基本方針

インタフェース仕様の共通化は、仕様に合致しないコンポーネントを排除するため、時に開発内容を制限してしまうこともある。本仕様では、そのような制限を低減するために、以下のような方針で共通インタフェース仕様を定義する。

- 最低限のインタフェース仕様の定義：コンポーネントを相互接続・相互運用するために必要な最低限のインタフェース仕様のみを定義する。開発の制約となる仕様は最低限にとどめ、その他の部分は開発者が自由に拡張することができるようにする。
- 任意の機能の定義：いくつかの機能については実装を任意とする。実装された場合は、本書に書かれた仕様に準拠することを要求するが、実装をするかどうかは任意であり、それを実装していなかったからといって共通インタフェース仕様から外れるものとはしない。

2.2 フォーマットと表現方法

2.2.1 列挙型定義

本仕様書では、列挙型定義を次の表形式を用いて記述する。

表 XX <列挙型名>

<定数名>	<内容>

2.2.2 型定義

本仕様書では、型定義を次の表形式を用いて記述する。

表 XX <型名>

属性		
<要素名>	<要素型>	<内容>
...

2.2.3 インタフェース定義

本仕様書では、インタフェース定義を次の表形式を用いて記述する。

表 XX <インタフェース名>

メソッド				
<メソッド名>		<戻り値型>	<内容>	
	<方向>	<パラメータ名>	<パラメータ型>	<内容>

<備考>				

2.3 本仕様書における前提条件

2.2.1 座標系定義について

本仕様書で使用している座標系定義を以下に示す。

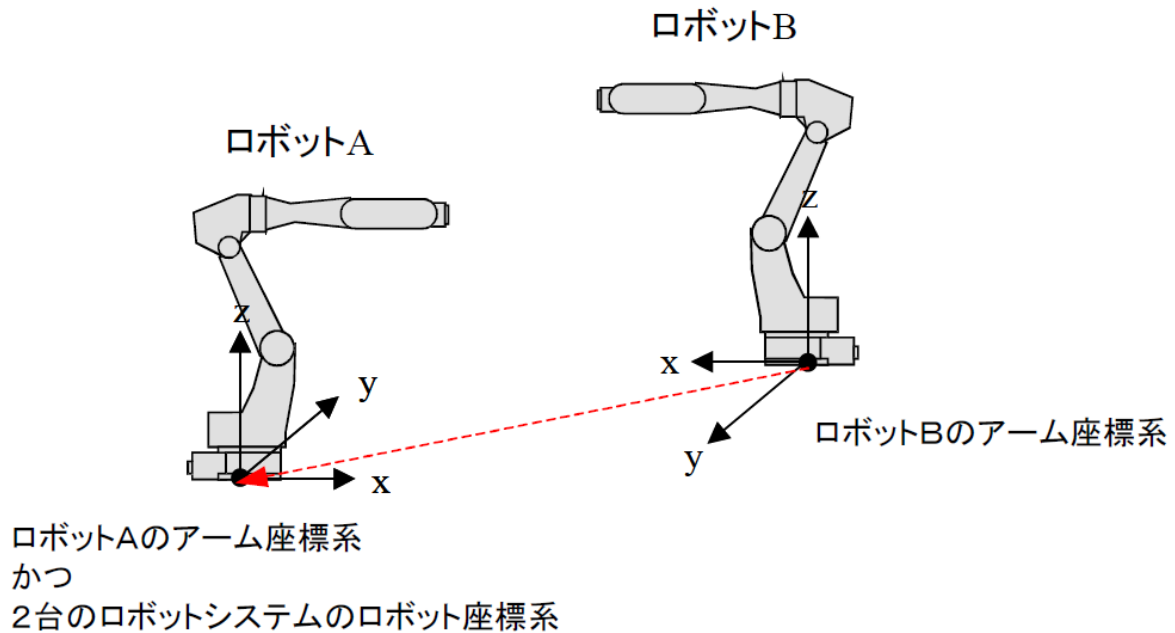


図 2.1 座標系定義

それぞれのロボットには、ベース部を原点とした右手系でアーム座標系が設定されている。また、複数ロボットが存在している場合で、ロボット間の座標系を合わせる必要がある場合には、ベースオフセットを設定することで座標系間の整合を取る。例えば上図の例において、ユーザがロボット B の座標系をロボット A の座標系に合わせて運転を行いたい(ロボット A のアーム座標系を全体システムの座標系として扱いたい)場合を考える。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、ロボット B のベースオフセットとして設定することで座標系の変換を行い、座標系間の整合を取る。

3 名前空間定義

ロボットアーム制御機能共通インタフェースでは、固有の名前空間は定義していない。

4 データ型定義

ロボットアーム制御機能共通インタフェースで使用するデータ型を以下に示す。

4.1 標準型

4.1.1 RTC::Time

時刻情報を格納するための型。OpenRTM-aist の標準型として BasicDataType.idl 内で定義されている。

表 4.1 RTC::Time

属性		
sec	unsigned long	秒単位の時刻情報
nsec	unsigned long	ナノ秒単位の時刻情報

4.2 型宣言

4.2.1 DoubleSeq

基本データ型 double のシーケンス型

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

4.2.2 JointPos

関節座標値を表現するための型。Double の配列として定義されている。

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

4.2.3 ULONG

基本データ型 unsigned long の短縮形。

```
typedef unsigned long ULONG;
```

4.2.4 AlarmSeq

アラーム情報のシーケンス型。

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

4.2.5 LimitSeq

上下制限値情報のシーケンス型。

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

4.2.6 HgMatrix

同次変換行列 4×4 の第 4 行を省略した 3×4 の行列。座標系は右手系。

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

4.3 ロボットアーム制御機能用

4.3.1 LimitValue

上下限の制限値を保持するための型。

表 4.2 LimitValue

属性		
upper	double	上限値
lower	double	下限値

4.3.2 RETURN_ID

リターン情報を保持するための型。

表 4.3 RETURN_ID

属性		
id	long	リターンコード
comment	string	戻りを説明するための詳細コメント

※本仕様書ではidに格納するリターンコードとして、使用頻度が高いと思われる以下の値を事前定義する。

表 4.4 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	オペレーションを拒否
-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
-6～ -9999	システム予約領域	
-10000 ～	機種依存領域	

4.3.3 TimedJointPos

アームロボットの関節座標値をタイムスタンプ付きで格納するための型。

表 4.5 TimedJointPos

属性		
tm	RTC::Time	時刻情報。関節座標値を設定した時刻などを格納するために利用。
pos	JointPos	関節座標値

※RTC::Time 型は OpenRTM-aist で用意している標準型。

4.3.4 AlarmType

アラームの種別を表現するための列挙型。

表 4.6 AlarmType

アラーム型	内容
FAULT	回復不能な致命的なエラー
WARNING	回復可能な軽微なエラー
UNKNOWN	重篤度が不明なエラー

4.3.5 Alarm

アラーム情報を格納するための型。

表 4.7 Alarm

属性		
code	unsigned long	アラームコード
type	AlarmType	アラームの種別
description	string	アラームに関する詳細説明

※本仕様書では code に格納するアラームコードとして、使用頻度が高いと思われる以下の値を事前定義する。

表 4.8 アラームコード一覧

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)
0x00000006～ 0x000003FF	システム予約領域
0x00000400～ 0xFFFFFFFF	機種依存領域

4.3.6 ManipInfo

制御対象のマニピュレータの情報を格納するための型。

表 4.9 ManipInfo

属性		
manufactur	string	メーカー名
type	string	機種名
axisNum	ULONG	グリップを除いた軸数
cmdCycle	ULONG	低レベル位置指令を受け取る周期
isGripper	boolean	1 軸グリップの有無。グリップ未装着時及び多指ハンド装着時は false を設定する。

4.3.7 CarPosWithElbow

位置姿勢(同次変換行列)と肘角を格納するための型。

表 4.10 CarPosWithElbow

属性		
carPos	HgMatrix	位置姿勢を表現する同次変換行列
elbow	elbow	肘の角度
structFlag	ULONG	付加情報を格納するためのフラグ

※structFlag に格納する情報は、機種依存データとなる。詳細は各マニピュレータのドキュメントを参照のこと。

4.3.8 CartesianSpeed

並進と回転の速度情報を格納するための型。

表 4.11 CartesianSpeed

属性		
translation	double	並進速度
rotation	double	回転角速度

5 共通インタフェース定義

以下にロボットアーム制御機能共通インタフェースで使用する共通インタフェースの定義を示す。

5.1 データポート

5.1.1 位置指令インタフェース

低レベル ACT RTC が、中レベル ACT RTC からマニピュレータの各関節への角度指令を受け取るためのインタフェースである。

位置指令は、TimedJointPos 型を用いて受け渡され、データ長はアーム軸数+グリッパ軸数(1 軸)となる (アーム軸数およびグリッパ有無の情報は、getManipInfo オペレーションにて取得可能)。

本インタフェースでは、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を推奨する。また、上位モジュールは、設定されたデータ受信周期に応じた位置指令データを準備する必要がある (データ受信周期は、機種依存であり getManipInfo オペレーションにて取得可能)。

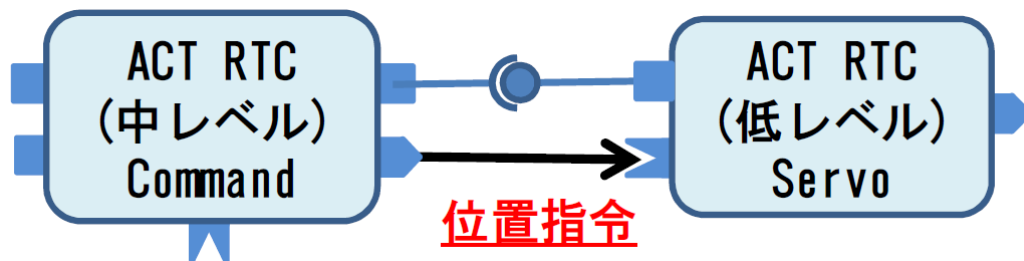


図 5.1 位置指令インタフェース

5.1.2 位置フィードバック指令インタフェース

低レベル ACT RTC が、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中レベル ACT RTC に伝達するためのインタフェースである。

位置フィードバック指令は、TimedJointPos 型を用いて出力され、データ長はアーム軸数+グリッパ軸数(1 軸)となる (アーム軸数およびグリッパ有無の情報は、getManipInfo オペレーションにて取得可能)。

本インタフェースでは、最新の位置フィードバック値を出力する必要があるため、NullBuffer 型のバッファ指定を使用することを推奨する。また、データの出力周期は、機種依存であり getManipInfo オペレーションにて取得可能である。

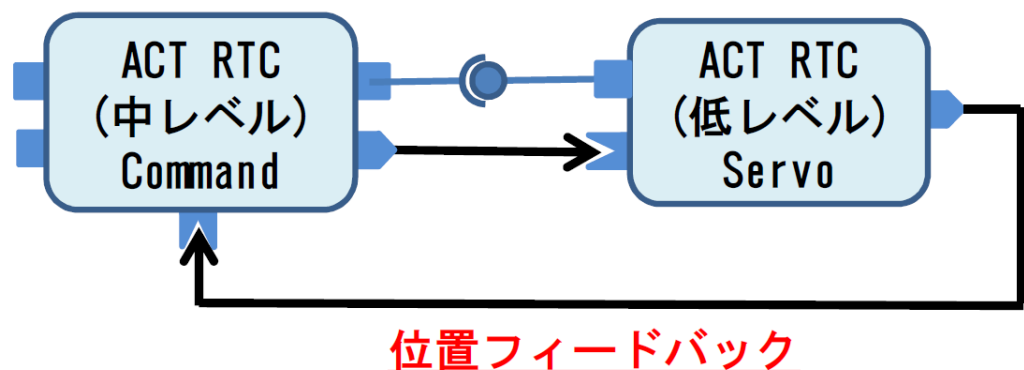


図 5.2 位置フィードバック指令インタフェース

5.2 サービスポート

5.2.1 ManipulatorCommonInterface_Common

サーボ On/Off やステータス取得など、低・中レベル両方で使用するコマンドをまとめた共通インタフェース。

表 5.1 ManipulatorCommonInterface_Common

メソッド				
clearAlarms		RETURN_ID	アラームクリア	
getActivAlarm		RETURN_ID	アラーム情報の取得	
	out	alarms	AlarmSeq	アラーム情報の配列
アラームなしの場合は、サイズ 0 の double シーケンスを返す。 アラームが N 個の場合は、サイズ N の double シーケンスを返す。				
getFeedbackPosJoint		RETURN_ID	関節座標系の位置フィードバック情報の取得	
	out	pos	JointPos	位置フィードバック情報(シーケンス型)
配列の値の順番は、アーム(J1、J2、J3 …)+グリッパ(1 軸)とする。 アーム軸数およびグリッパの有無は、getManipInfo オペレーションにて取得可能。				
getManipInfo		RETURN_ID	マニピュレータ情報の取得	
	out	mInfo	ManipInfo	マニピュレータ情報
getSoftLimitJoint		RETURN_ID	関節座標系のソフトリミット値を取得	
	out	softLimit	LimitSeq	各軸のソフトリミット値[単位:degree or mm]
RTC を起動後、オペレーション setSoftLimitJoint を 1 回も実行していない場合の値は、実装依存となる。				
getState		RETURN_ID	ユニットの状態取得	
	out	state	ULONG	ユニットの状態を表すビットコード
各ビットコードの詳細については、表 5.2 状態ビット一覧を参照。				
servoOFF		RETURN_ID	全軸サーボ OFF	
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 0x01 が 0 となる。				
servoON		RETURN_ID	全軸サーボ ON	
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 0x01 が 1 となる。				
setSoftLimitJoint		RETURN_ID	関節座標系のソフトリミット値設定	
	in	softLimit	LimitSeq	各軸のソフトリミット値[単位:degree or mm] サイズはマニピュレータの軸数に対応。
動作中、アラーム発生中は、本オペレーションの実行は拒否される。				

表 5.2 状態ビット一覧

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(中レベル RTC のみ)
0x10	一時停止中

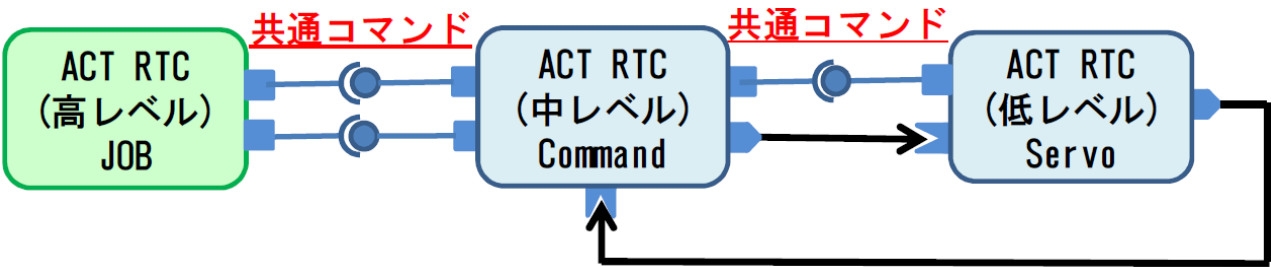


図 5.3 低・中レベル共通インタフェース

5.2.2 ManipulatorCommonInterface_Middle

中レベル・モーションコマンドで使用するコマンドをまとめた共通インタフェース

表 5.3 ManipulatorCommonInterface_Middle

メソッド				
closeGripper		RETURN_ID	グリッパを完全に閉じる。	
グリッパを閉じた際の姿勢については、機種依存。				
getBaseOffset		RETURN_ID	アーム座標系からロボット座標系までのベースオフセットを取得する。	
	out	offset	HgMatrix	オフセット量
getFeedbackPosCartesian		RETURN_ID	ロボット座標系の位置フィードバック情報を取得する。	
	out	pos	CarPosWithElbow	位置フィードバック情報[単位:mm、degree]
1 ～ 6 軸アームの場合は、elbow は省略。				
getMaxSpeedCartesian		RETURN_ID	直交空間における動作時の最大速度を取得する。	
	out	speed	CartesianSpeed	最大並進速度[単位:mm/s]、最大回転速度[単位:degree/s]からなる最大速度情報。
setMaxSpeedCartesian オペレーションで設定した値を取得する。				
getMaxSpeedJoint		RETURN_ID	関節空間における動作時の最大速度を取得する。	
	out	speed	DoubleSeq	各軸の最大動作速度[単位:degree/s or mm/s]
本値は、モータ容量、ギヤ比、負荷といった条件から算出するものであるため、機種依存となる。				
getMinAccelTimeCartesian		RETURN_ID	直交動作時の最大速度までの最小加速時間を取得する。	
	out	aclTime	double	最小加速時間[単位:s]
setMinAccelTimeCartesian オペレーションで設定した値を取得する。				
getMinAccelTimeJoint		RETURN_ID	関節動作時の最大速度までの最小加速時間を取得する。	
	out	aclTime	double	最小加速時間[単位:s]
本値は、モータ容量、ギヤ比、負荷といった条件から算出するものであるため、機種依存となる。				

メソッド				
getSoftLimitCartesian			RETURN_ID	ロボット座標系でのソフトリミット値を取得する。
	out	xLimit	double	X 軸のソフトリミット値[単位:mm]
	out	yLimit	double	Y 軸のソフトリミット値[単位:mm]
	out	zLimit	double	Z 軸のソフトリミット値[単位:mm]
<p>各 move 命令による動作時に、アーム先端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を出力する。</p> <p>オペレーション setSoftLimitCartesian で設定した値を取得する。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。</p>				
moveGripper			RETURN_ID	グリップスを指定した開閉角度とする。
	in	angleRatio	ULONG	グリップスの開閉角度割合[%] 0%:完全に閉じた状態 100%:完全に開いた状態
moveLinearCartesianAbs			RETURN_ID	ロボット座標系の絶対値で指定された目標位置に対し、直交空間における直線補間で動作する。
	in	carPoint	CarPosWithElbow	絶対目標位置・姿勢[単位:mm、degree]
<p>「直交空間における直線補間」とは、直交空間中の各方向の並進および回転動作が、同時に開始・終了するとともに、全ての加速時間と減速時間が同じになるように軌跡生成する動作のことである。</p> <p>1～3 軸の直交座標型マニピュレータの場合は、戻り値 RETURN_ID のメンバ変数 id には NG(-1)が格納され、オペレーションは拒否される。</p> <p>4 軸の水平多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における Z 軸以外の軸回転による目標姿勢は無視される。</p> <p>5 軸の垂直多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における X 軸の回転による目標姿勢は無視される。</p> <p>また、4～6 軸のマニピュレータの場合は、引数 carPoint のメンバ変数 elbow は無視される。</p>				
moveLinearCartesianRel			RETURN_ID	ロボット座標系の相対値で指定された目標位置に対し、直交空間における直線補間で動作する。
	in	carPoint	CarPosWithElbow	相対目標位置・姿勢[単位:mm、degree]
<p>「直交空間における直線補間」とは、直交空間中の各方向の並進および回転動作が、同時に開始・終了するとともに、全ての加速時間と減速時間が同じになるように軌跡生成する動作のことである。</p> <p>1～3 軸の直交座標型マニピュレータの場合は、戻り値 RETURN_ID のメンバ変数 id には NG(-1)が格納され、オペレーションは拒否される。</p> <p>4 軸の水平多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における Z 軸以外の軸回転による目標姿勢は無視される。</p> <p>5 軸の垂直多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における X 軸の回転による目標姿勢は無視される。</p> <p>また、4～6 軸のマニピュレータの場合は、引数 carPoint のメンバ変数 elbow は無視される。</p>				

メソッド				
movePTPCartesianAbs			RETURN_ID	ロボット座標系の絶対値で指定された目標位置に対し、関節空間における直線補間で動作する。
	in	carPoint	CarPosWithElbow	絶対目標位置・姿勢[単位:mm、degree]
<p>「関節空間における直線補間」とは、全軸の動作が、同時に開始・終了するとともに、全ての加速時間と減速時間が同じになるように軌跡生成する動作のことである。</p> <p>1～3軸の直交座標型マニピュレータの場合は、戻り値 RETURN_ID のメンバ変数 id には NG(-1)が格納され、オペレーションは拒否される。</p> <p>4軸の水平多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における Z 軸以外の軸回転による目標姿勢は無視される。</p> <p>5軸の垂直多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における X 軸の回転による目標姿勢は無視される。</p> <p>また、4～6軸のマニピュレータの場合は、引数 carPoint のメンバ変数 elbow は無視される。</p>				
movePTPCartesianRel			RETURN_ID	ロボット座標系の相対値で指定された目標位置に対し、関節空間における直線補間で動作する。
	in	carPoint	CarPosWithElbow	相対目標位置・姿勢[単位:mm、degree]
<p>「関節空間における直線補間」とは、全軸の動作が、同時に開始・終了するとともに、全ての加速時間と減速時間が同じになるように軌跡生成する動作のことである。</p> <p>1～3軸の直交座標型マニピュレータの場合は、戻り値 RETURN_ID のメンバ変数 id には NG(-1)が格納され、オペレーションは拒否される。</p> <p>4軸の水平多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における Z 軸以外の軸回転による目標姿勢は無視される。</p> <p>5軸の垂直多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における X 軸の回転による目標姿勢は無視される。</p> <p>また、4～6軸のマニピュレータの場合は、引数 carPoint のメンバ変数 elbow は無視される。</p>				
movePTPJntAbs			RETURN_ID	絶対関節座標で指定された目標位置に対し、関節空間における直線補間で動作する。
	in	jointPoint	JointPos	絶対目標位置[単位:degree or mm]
<p>「関節空間における直線補間」とは、全軸の動作が、同時に開始・終了するとともに、全ての加速時間と減速時間が同じになるように軌跡生成する動作のことである。</p> <p>引数 jointPoint 配列の値の順番は、J1、J2、J3、・・・とする。</p>				
movePTPJntRel			RETURN_ID	相対関節座標で指定された目標位置に対し、関節空間における直線補間で動作する。
	in	jointPoint	jointPos	相対目標位置[単位:degree or mm]
<p>「関節空間における直線補間」とは、全軸の動作が、同時に開始・終了するとともに、全ての加速時間と減速時間が同じになるように軌跡生成する動作のことである。</p> <p>引数 jointPoint 配列の値の順番は、J1、J2、J3、・・・とする。</p>				

メソッド				
openGripper		RETURN_ID		グリッパを完全に開く。
グリッパを開いた際の姿勢については、機種依存。				
pause		RETURN_ID		マニピュレータの全ての軸を一時停止する。
マニピュレータが動作中の場合、減速停止する。 一時停止状態において、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は行わない。 一時停止状態の解除は、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、停止中に本オペレーションが呼ばれた場合には無視する。				
resume		RETURN_ID		マニピュレータの動作を再開する。
一時停止中以外に本オペレーションが呼ばれた場合には全て無視する。				
stop		RETURN_ID		マニピュレータの動作を停止する。
マニピュレータが動作中の場合は、減速停止し、蓄積されている全てのモーション命令を破棄する。 マニピュレータが一時停止中の場合は、蓄積されている全てのモーション命令を破棄し、一時停止状態も解除する。 サーボ Off 中、アラーム中に本オペレーションが呼ばれた場合には無視する。				
setAccelTimeCartesian		RETURN_ID		直交空間における動作時の加速時間を設定する。
	in	aclTime	double	加速時間[単位:s]
setMinAccelTimeCartesian オペレーションで設定された値未満の値が指定された場合には、エラーとする。				
setAccelTimeJoint		RETURN_ID		関節空間における動作時の加速時間を設定する。
	in	aclTime	double	加速時間[単位:s]
setMinAccelTimeJoint オペレーションで設定された値未満の値が指定された場合には、エラーとする。				
setBaseOffset		RETURN_ID		オフセット量を設定する。
	in	offset	Hgmatrix	オフセット量
対象マニピュレータのアーム座標系から、基準となるロボット座標系までのオフセット量を設定する。 本オペレーションを 1 回も実行していない場合のオフセット量は 0 とする。				
setControlPointOffset		RETURN_ID		制御点のフランジ面からのオフセット量を設定する。
	in	offset	HgMatrix	オフセット量
setMaxSpeedCartesian		RETURN_ID		直交空間における動作時の最大動作速度を設定する。
	in	speed	CartesianSpeed	最大並進速度[単位:mm/s]、最大回転速度[単位:degree/s]からなる最大速度情報
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。				
setMaxSpeedJoint		RETURN_ID		関節空間における動作時の最大動作速度を設定する。
	in	speed	DoubleSeq	各軸の最大動作速度[単位:degree/s or mm/s]
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。				

メソッド				
setMinAccelTimeCartesian			RETURN_ID	直交空間における動作時の最大速度までの最小加速時間を設定する。
	in	aclTime	double	最小加速時間[単位:s]
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。				
setMinAccelTimeJoint			RETURN_ID	関節空間における動作時の最大速度までの最小加速時間を設定する。
	in	aclTime	double	最小加速時間[単位:s]
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。				
setSoftLimitCartesian			RETURN_ID	ロボット座標系でのソフトリミット値を設定する。
	in	xLimit	double	X 軸ソフトリミット値[単位:mm]
	in	yLimit	double	Y 軸ソフトリミット値[単位:mm]
	in	zLimit	double	Z 軸ソフトリミット値[単位:mm]
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。 ロボット座標系でのリミットと関節座標系でのリミットは同時に機能する。				
setSpeedCartesian			RETURN_ID	直交空間における動作時の速度を%指定する。
	in	spdRation	ULONG	最大速度に対する割合指定[単位:%]
上限は 100%、初期値は 0%				
setSpeedJoint			RETURN_ID	関節空間における動作時の速度を%指定する。
	in	spdRation	ULONG	最大速度に対する割合指定[単位:%]
上限は 100%、初期値は 0%				

メソッド				
moveCircularCartesianAbs			RETURN_ID	ロボット座標系の絶対値で指定された中継位置・目標位置に対し、直交空間における円弧補間で動作する。
	in	carPointR	CarPosWithElbow	絶対中継位置・姿勢[単位:mm、degree]
	in	carPointT	CarPosWithElbow	絶対目標位置・姿勢[単位:mm、degree]
<p>「直交空間における円弧補間」とは、直交空間中の現在位置と 2 円周点を通る円弧となるように軌跡生成する動作のことである。</p> <p>1 軸の直交座標型マニピュレータの場合は、戻り値 RETURN_ID のメンバ変数 id には NG(-1)が格納され、オペレーションは拒否される。</p> <p>2 軸の直交座標型マニピュレータの場合は、中継位置・目標位置をアーム座標系の絶対値で指定する。このとき、J1 軸を X 軸、J2 軸を Y 軸とみなし、引数 carPoint のメンバ変数 HgMatrix における Z 軸の値、及び全ての軸回転による目標姿勢は無視される。</p> <p>3 軸の直交座標型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における全ての軸の回転による目標姿勢は無視される。</p> <p>4 軸の水平多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における Z 軸以外の軸回転による目標姿勢は無視される。</p> <p>5 軸の垂直多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における X 軸の回転による目標姿勢は無視される。</p> <p>また、2～6 軸のマニピュレータの場合は、引数 carPoint のメンバ変数 elbow は無視される。</p>				
moveCircularCartesianRel			RETURN_ID	ロボット座標系の相対値で指定された中継位置・目標位置に対し、直交空間における円弧補間で動作する。
	in	carPointR	CarPosWithElbow	相対中継位置・姿勢[単位:mm、degree]
	in	carPointT	CarPosWithElbow	相対目標位置・姿勢[単位:mm、degree]
<p>「直交空間における円弧補間」とは、直交空間中の現在位置と 2 円周点を通る円弧となるように軌跡生成する動作のことである。</p> <p>1 軸の直交座標型マニピュレータの場合は、戻り値 RETURN_ID のメンバ変数 id には NG(-1)が格納され、オペレーションは拒否される。</p> <p>2 軸の直交座標型マニピュレータの場合は、中継位置・目標位置をアーム座標系の相対値で指定する。このとき、J1 軸を X 軸、J2 軸を Y 軸とみなし、引数 carPoint のメンバ変数 HgMatrix における Z 軸の値、及び全ての軸回転による目標姿勢は無視される。</p> <p>3 軸の直交座標型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における全ての軸の回転による目標姿勢は無視される。</p> <p>4 軸の水平多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における Z 軸以外の軸回転による目標姿勢は無視される。</p> <p>5 軸の垂直多関節型マニピュレータの場合は、引数 carPoint のメンバ変数 HgMatrix における X 軸の回転による目標姿勢は無視される。</p> <p>また、2～6 軸のマニピュレータの場合は、引数 carPoint のメンバ変数 elbow は無視される。</p>				

メソッド				
setHome			RETURN_ID	原点復帰時の位置を関節座標系の絶対値で設定する。
	in	jointPoint	JointPos	絶対位置[単位:degree or mm]
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。				
引数 jointPoint 配列の値の順番は、J1、J2、J3、・・・とする。				
getHome			RETURN_ID	関節座標系の絶対値で定義された原点復帰位置を取得する。
	out	jointPoint	JointPos	絶対位置[単位:degree or mm]
setHome オペレーションで設定した値を取得する。				
goHome			RETURN_ID	関節座標系の絶対値で指定された原点復帰位置に対し、関節空間における直線補間で動作する。
「関節空間における直線補間」とは、全軸の動作が、同時に開始・終了するとともに、全ての加速時間と減速時間が同じになるように軌跡生成する動作のことである。				

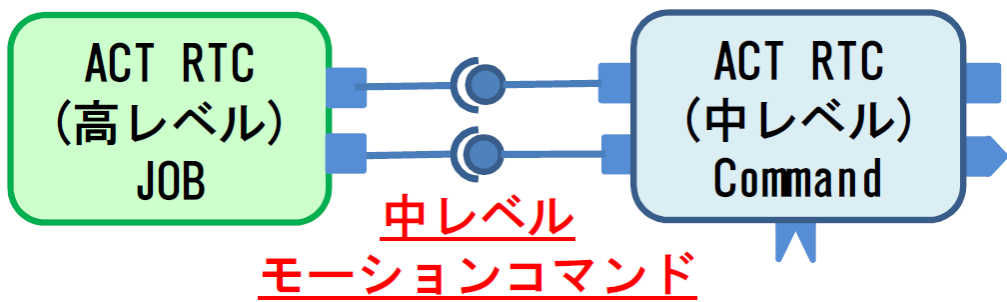


図 5.4 中レベル・モーションコマンドインタフェース

6 CORBA IDL

ロボットアーム制御機能共通インタフェースの IDL 定義を以下に示す。

6.1 ManipulatorCommonInterface_DataTypes.idl

```
/*
    Manipulator Common Interface (Data type defenition)
    - This IDL is used as service port on RTC
    - This command specification is provided by Intelligent RT Software
    Project of NEDO.
    rev. 20100502
*/
#ifndef MANIPULATORCOMMONINTERFACE_DATATYPES_IDL
#define MANIPULATORCOMMONINTERFACE_DATATYPES_IDL

#include "BasicDataType.idl"

typedef sequence<double> DoubleSeq;
typedef sequence<double> JointPos;

struct LimitValue {
    double upper;
    double lower;
};

struct RETURN_ID{
    long id;
    string comment;
};

struct TimedJointPos {
    Time tm;
    JointPos pos;
};

typedef unsigned long ULONG;

#endif // MANIPULATORCOMMONINTERFACE_DATATYPES_IDL
```


6.2 ManipulatorCommonInterface_Common.idl

```
/*
    Manipulator Common Interface (Common Commands)
        - This IDL is used as service port on RTC
        - This command specification is provided by Intelligent RT Software
    Project of NEDO.
    rev. 20100502
*/
#ifndef MANIPULATORCOMMONINTERFACE_COMMON_IDL
#define MANIPULATORCOMMONINTERFACE_COMMON_IDL

#include "ManipulatorCommonInterface_DataTypes.idl"

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

#endif // MANIPULATORCOMMONINTERFACE_COMMON_IDL
```

6.3 ManipulatorCommonInterface_Middle.idl

```
/*
    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. 20131028
*/
#ifndef MANIPULATORCOMMONINTERFACE_MIDDLE_IDL
#define MANIPULATORCOMMONINTERFACE_MIDDLE_IDL

#include "ManipulatorCommonInterface_DataTypes.idl"

typedef double HgMatrix [3][4];

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

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

interface ManipulatorCommonInterface_Middle {
    RETURN_ID closeGripper();
    RETURN_ID getBaseOffset(out HgMatrix offset);
    RETURN_ID getFeedbackPosCartesian(out CarPosWithElbow pos);
    RETURN_ID getMaxSpeedCartesian(out CartesianSpeed speed);
    RETURN_ID getMaxSpeedJoint(out DoubleSeq speed);
    RETURN_ID getMinAccelTimeCartesian(out double aclTime);
    RETURN_ID getMinAccelTimeJoint(out double aclTime);
    RETURN_ID getSoftLimitCartesian(out LimitValue xLimit,
                                     out LimitValue yLimit, out LimitValue zLimit );
    RETURN_ID moveGripper(in ULONG angleRatio);
    RETURN_ID moveLinearCartesianAbs(in CarPosWithElbow carPoint);
    RETURN_ID moveLinearCartesianRel(in CarPosWithElbow carPoint);
    RETURN_ID movePTPCartesianAbs(in CarPosWithElbow carPoint);
    RETURN_ID movePTPCartesianRel(in CarPosWithElbow carPoint);
    RETURN_ID movePTPJointAbs(in JointPos jointPoints);
    RETURN_ID movePTPJointRel(in JointPos jointPoints);
    RETURN_ID openGripper();
    RETURN_ID pause();
    RETURN_ID resume();
    RETURN_ID stop();
    RETURN_ID setAccelTimeCartesian(in double aclTime);
    RETURN_ID setAccelTimeJoint(in double aclTime);
    RETURN_ID setBaseOffset(in HgMatrix offset);
    RETURN_ID setControlPointOffset(in HgMatrix offset);
    RETURN_ID setMaxSpeedCartesian(in CartesianSpeed speed);
    RETURN_ID setMaxSpeedJoint(in DoubleSeq speed);
    RETURN_ID setMinAccelTimeCartesian(in double aclTime);
    RETURN_ID setMinAccelTimeJoint(in double aclTime);
    RETURN_ID setSoftLimitCartesian(in LimitValue xLimit,
                                     in LimitValue yLimit, in LimitValue zLimit);
    RETURN_ID setSpeedCartesian(in ULONG spdRatio);
    RETURN_ID setSpeedJoint(in ULONG spdRatio);
};
```

```
RETURN_ID moveCircularCartesianAbs(in CarPosWithElbow carPointR,  
                                     in CarPosWithElbow carPointT);  
RETURN_ID moveCircularCartesianRel(in CarPosWithElbow carPointR,  
                                     in CarPosWithElbow carPointT);  
RETURN_ID setHome(in JointPos jointPoint);  
RETURN_ID getHome(out JointPos jointPoint);  
RETURN_ID goHome();  
};  
#endif // MANIPULATORCOMMONINTERFACE_MIDDLE_IDL
```