次世代ロボット知能化技術開発プロジェクト

ACT（低レベル）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、低レベル ACT RTC（表 1 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。指令モードは、位置指令のみとし、速度指令やトルク指令は本インターフェースには含めないこととする。

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（中レベル）共通インターフェース仕様書

2 全体構成

低レベル ACT の RTC インターフェース構成を図 1 に示す。本 RTC は、1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。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。

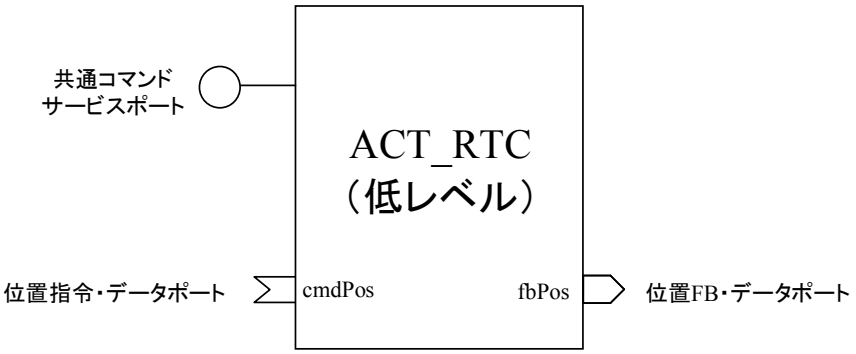

図 1　RTC インターフェース構成（低レベル）

3 共通データ型

ACT RTCで共通に使用するデータ型一覧を表 2 に示す。

表 2　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

3.1 データ型

ACT RTCで共通に定義されるデータ型について述べる。

3.1.1 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。

3.1.2 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。

3.1.3 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。

3.1.4 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 4 と関連文書[1]のサービスポートの説明を参照のこと。

3.1.5 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。

3.1.6 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。

4 共通コマンドサービスポートの仕様

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 3 に、戻り値一覧を表 4 に、インターフェースのオペレーション一覧を
表 5 に示す。

表 3 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 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
・・・	（システム予約領域 0 ～ -9999）	-
・・・	（機種依存領域 -10000 以降）	-

表 5 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

4.1 モジュール宣言

モジュール宣言は使用しない。

4.2 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。

4.3 データ型

共通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。

4.3.1 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。

4.3.2 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 6 に示す。

表 6　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	
・・・	（機種依存領域　0x00000400 ～ 0xFFFFFFFF）

0xFFFFFFFF	

4.3.3 AlarmSeq

概要

Alarm のシーケンス型。

定義

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

備考

なし。

4.3.4 LimitSeq

概要

LimitValue のシーケンス型。

定義

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

備考

なし。

4.3.5 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）

4.4 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。

4.4.1 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

4.4.2 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。

4.4.3 getFeedbackPosJoint

機能:

指定された座標系における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)

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、アーム(J1、J2、J3・・・)＋gripper(1軸) とする。(ただし、アーム軸数および gripper 有無の情報は、getManipInfo オペレーションで取得可能)

4.4.4 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。

4.4.5 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。

4.4.6 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 7 に示す。

表 7 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 （ただし、中レベル RTC のみ）
0x10	一時停止中

4.4.7 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。

4.4.8 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。

4.4.9 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。

5 低レベル・データポートの仕様

低レベルデータポート仕様について述べる。

5.1 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 gripper 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:

7軸アーム+グリッパの場合の例を以下に示す。

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	Gripper の〃

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。

5.2　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 gripper 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：

7軸アーム＋グリッパの場合の例を以下に示す。

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	Gripper の〃

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。

6 付録 IDL

本RTCの IDL を以下に示す。

```
/*
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. 20100318
*/

#ifndef MANIPULATORCOMMONINTERFACE_DATATYPES_IDL
#define MANIPULATORCOMMONINTERFACE_DATATYPES_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;                           /**< エラーコト                          */
   };

   struct TimedJointPos {
            Time tm;
            JointPos pos;
   };

   typedef unsigned long ULONG;

};
```

```
/*
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. 20100318
*/

#ifndef MANIPULATORCOMMONINTERFACE_COMMON_IDL
#define MANIPULATORCOMMONINTERFACE_COMMON_IDL

#include "ManipulatorCommonInterface_DataTypes.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);

};

#endif // MANIPULATORCOMMONINTERFACE_COMMON_IDL
```