汎用モーション RTC
インターフェース仕様書

資料番号：RB1400065
改版：第 4 版

YASKAWA

版	年　月	変 更 内 容
<1>	2010.09.17	・列挙型 CommandFrameType の列挙定数の名前を変更 ・腰軸の自由度に予約領域追加
<2>	2011/09/21	・オペレーションを追加　moveUnitAbs、moveUnitRel
<3>	2011/12/16	・グリッパ制御、およびフレームリセット機能は開発中のため将来機能と明記（表 5）
<4>	2011/12/19	・使用するモジュール名を“ 冗長性利用モジュール ”から“ 汎用モーション RTC コア ”に修正。

【 目 次 】

はじめに

本書は、汎用モーション RTC のインターフェース仕様書である。汎用モーション RTC は、SmartPal のような 7 軸アーム、腰軸、全方向移動台車といった冗長自由度を備えたロボットにおいて、アーム先端の制御点の目標位置指令を入力として、関節座標系における補間動作（PTP 動作）や直交座標における直線補間（CP 動作）を計算し、キネマティクス変換を行った上で、各関節への低レベル位置指令を生成するモジュールである。キネマティクス変換処理には、東北大学殿の汎用モーション RTC コア[4]を再利用して実現するものとする。また、制御対象とするロボットは、SmartPalⅤ、PA10、リファレンスハードウェア（RH）の３種類とする（図 1）。

SmartPal（安川電機）

PA10（三菱重工業殿）

RH（前川製作所殿）

図 1 制御対象のロボット

表 1 各ロボットの構成

ロボット	アーム部	腰部	移動台車部
SmartPal	７軸	１軸（２）	３軸（全方向）
PA10	７軸	-	-
RH	６軸	-	２軸（２輪駆動）

図 2 に汎用モーション RTC を用いたシステムの概略構成図を示す。汎用モーション RTC は図の網掛け部分であり、本書に記述するインターフェースは丸で囲ったサービスポートのインターフェースに該当する。図は制御対象として SmartPal を対象とした接続であるが、PA10 や RH を対象とする場合は接続モジュールを切り替えることになる。

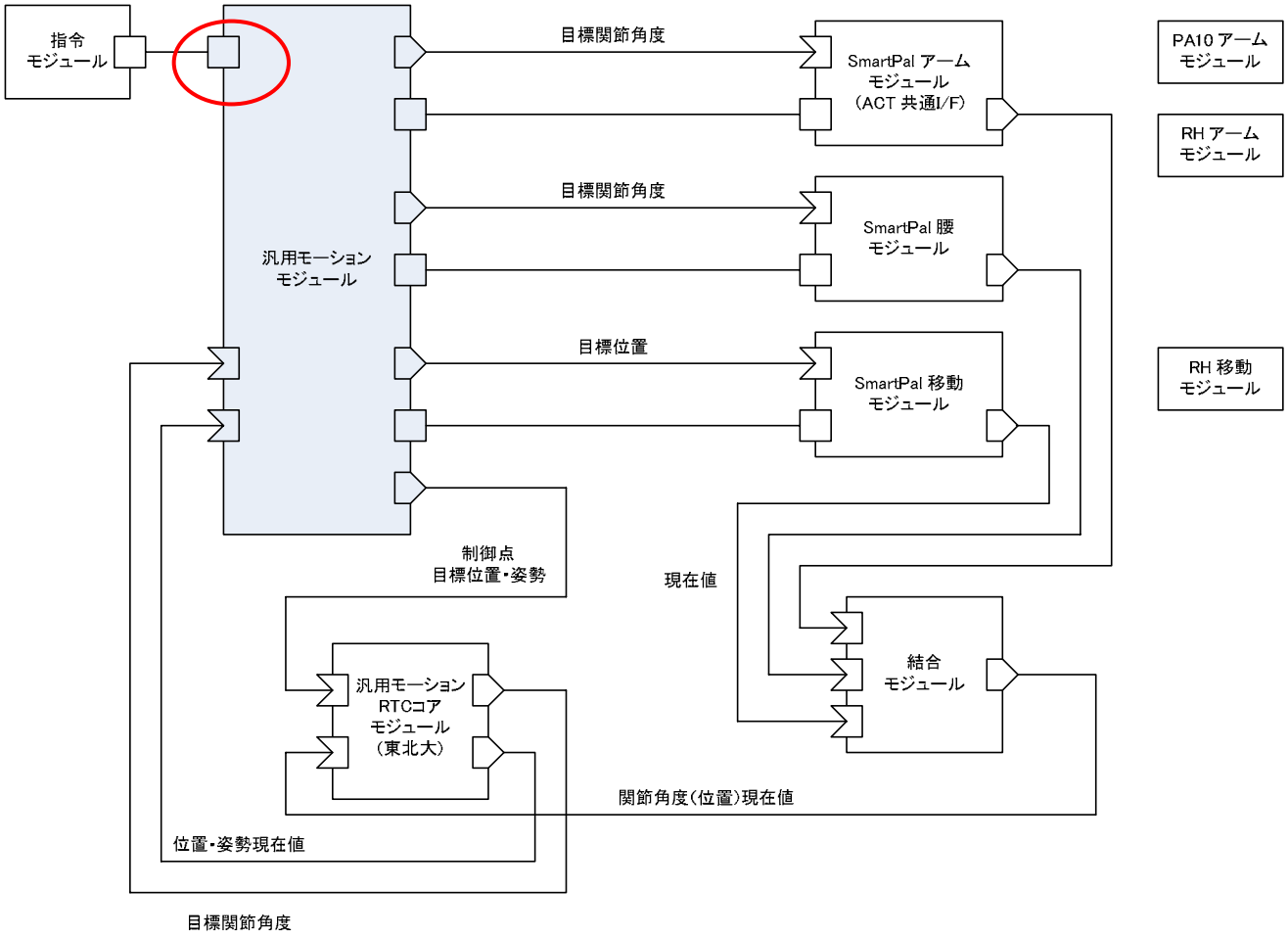

図 2　システム概略構成図

1　座標系

汎用モーション RTC で使用する座標系には、原点フレームと現在フレームの２種類がある。

原点フレームは、モーションの基準となる座標系であり、制御点の指令値は本フレームに基づいて指定する。ここでは、制御対象として SmartPal を用いて説明する。原点フレームの原点は腰軸の第１関節の原点とする。他のロボットを制御対象とする場合は、対象に合わせて原点をとる。原点フレームは、後述の resetOriginalFrame オペレーションで設定される。

現在フレームは、腰部や移動台車部の姿勢に関係なく、常にロボットについて回る座標系である。現在フレームの原点は原点フレームと同じであり、SmartPal では、腰軸の第１関節の原点となる。

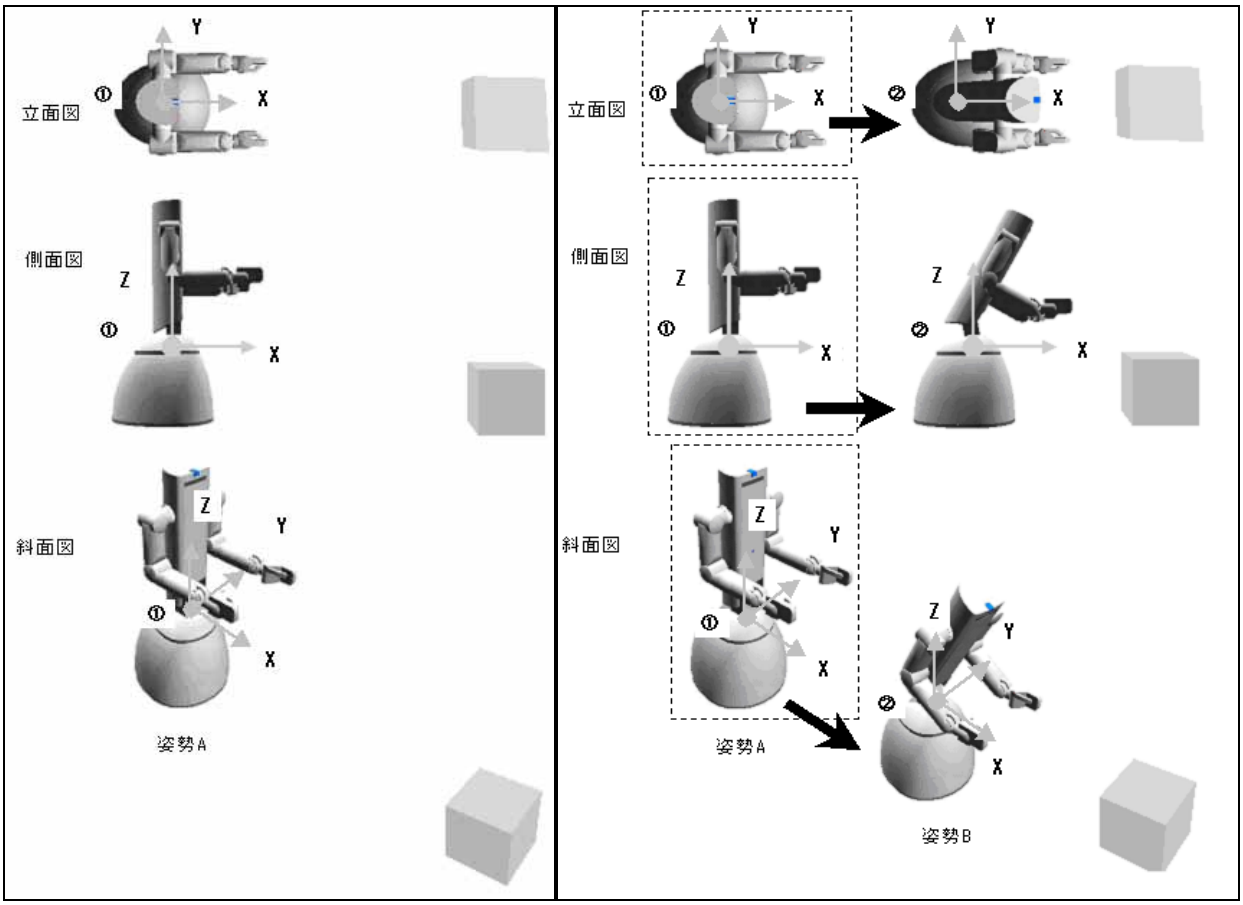

図 3　座標系

図 3 を用いて使用する２つのフレームについて説明する。図中の姿勢Ａ（左側）から移動台車を使用して前進した位置が姿勢Ｂである。この姿勢Ａ，Ｂに対応する各フレームの対応を表 2 に示す。原点フレームは①であり、ロボットが動いてもフレーム自体は元の位置を保持する。現在フレームは①と②であり、ロボットが移動するとそれに応じて変化するフレームである。

表 2　フレーム対応

フレーム	姿勢Ａ	姿勢Ｂ
原点フレーム	①	①
現在フレーム	①	②

2　インターフェース仕様

2.1　オペレーション一覧

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	CarPosWithElbow	位置姿勢（同次変換行列）と肘角を有する構造体
5	CartesianAccel	並進と回転の加速度情報を有する構造体
6	CartesianSpeed	並進と回転の速度情報を有する構造体
7	CommandFrameType	制御点の動作フレームの種別を示す列挙型
8	DoubleSeq<2>	double のシーケンス型
9	FrameType	制御点の原点となるフレームの種別を示す列挙型
10	HgMatrix	同次変換行列の情報を有する構造体
11	LimitValue	上下限の制限値を有する構造体
12	RedundantAccel	冗長軸の加速度情報を有する構造体
13	RedundantAxesMask	冗長軸の指定を有する構造体
14	RedundantLimit	冗長軸のリミット値情報を有する構造体
15	RedundantPos	冗長軸の位置情報を有する構造体
16	RedundantSpeed	冗長軸の速度情報を有する構造体
17	ReturnID	リターン情報を有する構造体
18	RightLeft	アームの左右の種別を示す列挙型
19	UnitType<2>	ユニットの種別を示す列挙型

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
-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	CANNOT_REACH_TARGET_ERR	目標位置へ到達不可能

表 5 オペレーション一覧

No	オペレーション名	概　要
1	abort	現在の動作を中止し、即時停止する
2	clearAlarms	全てのアラームを解除する
3	closeGripper	グリッパを完全に閉じる ※将来機能<3>
4	getActiveAlarm	発生中のアラームを取得する
5	getFeedbackPosCartesian	制御点の直交座標のフィードバック値を取得する
6	getFeedbackPosJoint<2>	指定ユニットの各軸の現在位置を取得する
7	getGripperPos<2>	グリッパの現在位置を取得する ※将来機能<3>
8	getState	モジュールの状態を返す
9	getVersion	バージョン情報を取得する
10	isMoving	全軸の動作状態を返す
11	isServoOn	サーボ制御の入切状態を返す
12	moveUnitAbs<2>	ユニットの各軸を指定位置（絶対位置）に移動する
13	moveUnitRel<2>	ユニットの各軸を指定位置（相対位置）に移動する
14	moveCPHoldAbs	制御点を維持したまま冗長軸を絶対指令で動作する
15	moveCPHoldRel	制御点を維持したまま冗長軸を相対指令で動作する
16	moveGripper	グリッパを指定された角度へ移動する ※将来機能<3>
17	moveLinearCartesianAbs	制御点を原点フレームに基づく絶対位置指令(同次変換行列）で直線補間動作する
18	moveLinearCartesianRel	制御点を原点フレームに基づく相対位置指令(同次変換行列）で直線補間動作する
19	movePTPCartesianAbs	関節空間において、目標位置を絶対直交座標指定により、直線補間を動作する
20	movePTPCartesianRel	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
21	openGripper	グリッパを完全に開く ※将来機能<3>
22	pause	全軸の動作を一時停止する
23	resetOriginalFrame	協調制御原点フレームをロボットの現在姿勢でリセットする ※将来機能<3>
24	resume	一時停止した動作を再開する
25	servoOff	全モーションモジュール全軸のサーボ制御を切る
26	servoOn	全モーションモジュール全軸のサーボ制御を入れる
27	setControlPointOffset	制御点のフランジ面からのオフセット量を設定する
28	selectRedundantAxes	動作に使用可能な冗長軸を設定する
29	stop	現在の動作を中止し、減速停止する

2.1.1 モジュール宣言

モジュール宣言は使用しない。

2.1.2 インターフェース宣言

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

2.1.3 データ型

2.1.3.1 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。

2.1.3.2 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;
        string moduleName
};
```

備考

Alarm 構造体型の持つメンバを表 6 に示す。

表 6　Alarm 構造体メンバ

メンバ	内容
code	発生した FAULT/WARNING を示す数値コード
type	発生したアラームが FAULT か WARNING か UNKNOWN かを示す。AlarmType 列挙型
description	発生したアラーム内容を説明する文字列
moduleName	発生元のモジュール名

2.1.3.3 AlarmSeq

概要
Alarm のシーケンス型。
定義
typedef sequence<Alarm> AlarmSeq;
備考
なし。

2.1.3.4 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;
    unsigned long    structFlag;
};
```

備考
CarPosWithElbow 構造体型の持つメンバを表 7 に示す。

表 7　CarPosWithElbow 構造体メンバ

メンバ	内容
carPos	位置姿勢の同次変換行列
elbow	肘角度
structFlag	機種依存データ。詳細は各マニピュレータのドキュメントを参照。

2.1.3.5 CartesianAccel

概要
並進と回転の加速度情報を有する構造体。
定義

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

備考
CartesianAccel 構造体型の持つメンバを表 8 に示す。

表 8　CartesianAccel 構造体メンバ

メンバ	内容
translation	並進加速度[mm/s²]
rotation	回転加速度[deg/s²]

2.1.3.6 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

CartesianSpeed 構造体型の持つメンバを表 9 に示す。

表 9　CartesianSpeed 構造体メンバ

メンバ	内容
translation	並進速度[mm/s]
rotation	回転速度[deg/s]

2.1.3.7 CommandFrameType

概要

制御点の動作フレームの種別を示す列挙型。

定義

```
enum CommandFrameType {
     BASE_FRAME<1>,
     TOOL_FRAME
 };
```

備考

CommandFrameType 列挙型は、要素として *BASE_FRAME*<1> と *TOOL_FRAME* の 2 つを持つ。*BASE_FRAME*<1> は基準直交座標と呼び、ベース座標のフレームに基づいて制御点を動作させる。一方 *TOOL_FRAME* は、制御点について回るツールフレームに基づいて制御点を動作させる。

2.1.3.8 DoubleSeq<2>

概要

double のシーケンス型。

定義

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

備考

なし。

2.1.3.9 FrameType

概要

制御点の原点となるフレームの種別を示す列挙型。

定義

```
enum FrameType {
    ORIGINAL_FRAME,
    CURRENT_FRAME
  };
```

備考

FrameType 列挙型は、要素として *ORIGINAL_FRAME* と *CURRENT_FRAME* の２つを持つ。各フレームの意味は、1 章の座標系を参照。

2.1.3.10 HgMatrix

概要

同次変換行列（Homogeneous Transform Matrix）の情報を有する構造体。

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。

2.1.3.11 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

2.1.3.12 RedundantAccel

概要

冗長軸の加速度情報を有する構造体。

定義

```
struct RedundantAccel {
   double   elbow;
   double   lumbarRx<2>;
   double   lumbarRy<2>;
   double   lumbarRz<2>;
   double   vehicleTranslation;
   double   vehicleRotation;
};
```

備考

RedundantAccel 構造体型の持つメンバを表 10 に示す。

表 10　RedundantAccel 構造体メンバ

メンバ	内容
elbow	肘の加速度[deg/s²]
lumbarRx<2>	腰軸 rx の加速度[deg/s²] 予約
lumbarRy<2>	腰軸 ry の加速度[deg/s²]
lumbarRz<2>	腰軸 rz の加速度[deg/s²] 予約
vehicleTranslation	移動部の並進加速度[mm/s²]
vehicleRotaion	移動部の回転加速度[deg/s²]

2.1.3.13 RedundantAxesMask

概要

冗長軸の指定を有する構造体。

定義

```
struct RedundantAxis {
   boolean elbow;
   double lumbarRx<2>;
   double lumbarRy<2>;
   double lumbarRz<2>;
   boolean vehicleX;
   boolean vehiclesY;
   boolean vehicleTheta;
};
```

備考

RedundantAxesMask 構造体型の持つメンバを表 11 に示す。

表 11 RedundantAxesMask 構造体メンバ

メンバ	内容
elbow	アーム肘角を冗長軸指定
lumbarRx<2>	腰軸 rx 予約
lumbarRy<2>	腰軸 ry（腰部２関節の合成角度としての腰曲げ角を冗長軸指定）
lumbarRz<2>	腰軸 rz 予約
vehicleX	移動部 X 軸（前後）方向移動を冗長軸指定
vehicleY	移動部 Y 軸（左右）方向移動を冗長軸指定
vehicleTheta	移動部回転角を冗長軸指定

2.1.3.14 RedundantLimit

概要

冗長軸のリミット値情報を有する構造体。

定義

```
struct RedundantLimit {
   LimitValue   rightElbow;
   LimitValue   leftElbow;
   LimitValue   lumbarRx<2>;
   LimitValue   lumbarRy<2>;
   LimitValue   lumbarRz<2>;
   LimitValue   vehicleX;
   LimitValue   vehicleY;
   LimitValue   vehicleTheta;
};
```

備考

RedundantLimit 構造体型の持つメンバを表 12 に示す。

表 12 RedundantLimit 構造体メンバ

メンバ	内容
rightElbow	アーム部右肘のリミット値
leftElbow	アーム部左肘のリミット値
lumbarRx<2>	腰軸 rx のリミット値 予約
lumbarRy<2>	腰軸 ry のリミット値
lumbarRz<2>	腰軸 rz のリミット値 予約
vehicleX	移動部の X 軸方向のリミット値
vehicleY	移動部の Y 軸方向のリミット値
vehicleTheta	移動部の回転軸のリミット値

2.1.3.15 RedundantPos

概要

冗長軸の位置情報を有する構造体。

定義

```
struct RedundantPos {
    double    elbow;
    double    lumbarRx<2>;
    double    lumbarRy<2>;
    double    lumbarRz<2>;
    double    vehicleX;
    double    vehicleY;
    double    vehicleTheta;
};
```

備考

RedundantPos 構造体型の持つメンバを表 13 に示す。

表 13　RedundantPos 構造体メンバ

メンバ	内容
elbow	肘の指令位置[deg]
lumbarRx<2>	腰軸 rx の指令位置　予約
lumbarRy<2>	腰軸 ry の指令位置
lumbarRz<2>	腰軸 rz の指令位置　予約
vehicleX	移動部のＸ指令位置[mm]
vehicleY	移動部のＹ指令位置[mm]
vehiocleTheta	移動部のθ指令位置[deg]

2.1.3.16 RedundantSpeed

概要

冗長軸の速度情報を有する構造体。

定義

```
struct RedundantSpeed {
    double    elbow;
    double    lumbarRx<2>;
    double    lumbarRy<2>;
    double    lumbarRz<2>;
    double    vehicleTranslation;
    double    vehicleRotation;
};
```

備考

RedundantSpeed 構造体型の持つメンバを表 14 に示す。

表 14 RedundantSpeed 構造体メンバ

メンバ	内容
elbow	肘の速度[deg/s]
lumbarRx<2>	腰軸 rx の速度 予約
lumbarRy<2>	腰軸 ry の速度
lumbarRz<2>	腰軸 rz の速度 予約
vehicleTranslation	移動部の並進速度[mm/s]
vehicleRotaion	移動部の回転速度[deg/s]

2.1.3.17 ReturnID

概要

リターン情報を有する構造体。

定義

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

備考

なし。

2.1.3.18 RightLeft

概要

アームの左右の種別を示す列挙型。

定義

```
enum RightLeft {
    RIGHT,
    LEFT
};
```

備考

なし。

2.1.3.19 UnitType<2>

概要

ユニットの種別を示す列挙型。

定義

```
enum UnitType {,
    RIGHT_ARM
    LEFT_ARM
    LUMBER,
    VEHICLE
  };
```

備考

2.1.4 オペレーション

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。

2.1.4.1 abort

機能：
現在の動作指令払い出しを中止し、即時停止する。蓄積された動作指令がある場合は全て破棄する。モジュールの状態を Ready にする。

宣言：
ReturnID abort();

引数：
なし。

戻り値：
常に OK

備考：
なし

2.1.4.2 clearAlarms

機能：
すべてのアラームのクリアを実行する。

宣言：
ReturnID clearAlarms();

引数：
なし。

戻り値：
常に OK

備考：
なし

2.1.4.3 closeGripper

機能：
グリッパを完全に閉じる。

宣言：
ReturnID closeGripper(
in RightLeft type
);

引数：

名前	入力・出力	説明
type	入力	対象が右アームか左アームかを指定する RightLeft 列挙型の値。

戻り値：

値	説明
OK	成功。
STATUS_ERR	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正
NOT_SV_ON_ERR	全ての軸のサーボが入っていない

備考：
・ 制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。

2.1.4.4 getActiveAlarm

機能：
発生中のアラーム情報を取得する。

宣言：
ReturnID getActiveAlarm(out 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：
常に OK

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。

2.1.4.5 getFeedbackPosCartesian

機能：
制御点の直交座標のフィードバック値を取得する。

宣言：
ReturnID getFeedbackPosCartesian(
in RightLeft type,
in FrameType frameType,
out CarPosWithElbow pos
);

引数：

名前	入力・出力	説明
type	入力	右アームか左アームかを指定する RightLeft 列挙型の値
frameType	入力	制御点の原点となるフレームタイプを指定する FrameType 型の値
pos	出力	制御点のフィードバック位置・姿勢

戻り値：

値	説明
OK	成功。
VALUE_ERR	引数が不正。

備考：
- 制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。
- frameType で指定する FrameType 列挙型は、*OriginalFrame* と *CurrentFrame* の２つが指定可能である。

2.1.4.6 getFeedbackPosJoint<2>

機能：
指定したユニットの各軸の現在位置を取得する。

宣言：
ReturnID getFeedbackPosJoint(
in UnitType type,
out DoubleSeq position
);

引数：

名前	入力・出力	説明
type	入力	対象となるユニットを指定する UnitType 列挙型の値
position	出力	指定ユニットの現在位置情報 引数 unit が RIGHT_ARM の場合： 関節各軸の現在位置　単位[deg] 引数 unit が LEFT_ARM の場合： 関節各軸の現在位置　単位[deg] 引数 unit が LUMBER の場合： 腰軸の現在位置　単位[deg] 引数 unit が VEHICLE の場合： 移動台車の並進位置　x,y 単位[mm] 回転位置　θ 単位[deg]

戻り値：

値	説明
OK	成功。
VALUE_ERR	引数が不正。

備考：
・指定したユニットがない場合はエラー（VALUE_ERR）となる。

2.1.4.7 getGripperPos<2>

機能：
グリッパの現在位置を取得する。

宣言：

ReturnID getGripperPos(
in RightLeft type,
out unsigned long angleRatio
);

引数：

名前	入力・出力	説明
type	入力	対象が右アームか左アームかを指定する RightLeft 列挙型の値
angleRatio	出力	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。
VALUE_ERR	引数が不正

備考：
・ 制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。

2.1.4.8 getState

機能：
汎用モーション RTC の状態を返す。

宣言：

ReturnID getState(
out unsigned long statusId,
out string message
);

引数：

名前	入力・出力	説明
statusId	出力	状態を表す状態コード
message	出力	状態を表す状態文字列

戻り値：
常に OK

備考：
状態コードと状態文字列、意味を表 15 に示す。

表 15 状態一覧

状態コード(statusId)	状態文字列(message)	意味
0x010	Unpowered	主回路電源オフ状態 （主回路オフ状態のモジュールがある）
0x012	Ready	指令待機状態 主回路電源オン、サーボ制御オン （すべてのモジュールがサーボ制御オン状態）
0x013	Busy	指令実行状態 （指令実行中のモジュールがある）
0x014	Pause	動作一時停止状態 （一時停止中のモジュールがある）
0x015	Alarm	アラーム発生状態

2.1.4.9 getVersion

機能：
汎用モーション RTC のバージョン情報を返す。

宣言：

ReturnID getVersion(
out string versionMessage
);

引数：

名前	入力・出力	説明
versionMessage	出力	バージョンを示す文字列

戻り値：
常に OK

備考：

2.1.4.10 isMoving

機能：
全モーションモジュールの軸の動作状態を返す。

宣言：

boolean isMoving();

引数：
なし

戻り値：
動作中の軸が存在し、指令実行状態の場合 true。
全モーションモジュールの全軸が動作完了し、指令待機状態の場合 false。

備考：

2.1.4.11 isServoOn

機能：
全モーションモジュールの軸のサーボ制御状態を返す。

宣言：
boolean isServoOn();

引数：
なし

戻り値：
全モーションモジュール全軸のサーボ制御状態がオン状態の場合 true。サーボ制御状態がオフ状態の軸が１つでも存在する場合は false。

備考：

2.1.4.12 moveUnitAbs<2>

機能：
指定したユニットの各軸を指定された位置（絶対位置）に移動する。

宣言：

```
ReturnID moveUnitAbs (
        in UniType type,
        in DoubleSeq absPos,
        in DoubleSeq speed,
        in DoubleSeq accel
        );
```

引数：

名前	入力・出力	説明
type	入力	対象となるユニットを指定する UnitType 列挙型の値
absPos	入力	指定ユニットの目標位置情報 引数 unit が RIGHT_ARM or LEFT_ARM の場合： 関節各軸の指令位置　単位[deg] 引数 unit が LUMBER の場合： 腰軸の指令位置　単位[deg] 引数 unit が VEHICLE の場合： 移動台車の並進位置　x,y 単位[mm]、回転位置　θ 単位[deg]
speed	入力	指定ユニットの目標速度情報 引数 unit が RIGHT_ARM or LEFT_ARM の場合： 指令速度　単位[deg/s] 引数 unit が LUMBER の場合： 指令速度　単位[deg/s] 引数 unit が VEHICLE の場合： 移動台車の並進速度　単位[mm/s]、回転速度　単位[deg/s]
accel	入力	指定ユニットの目標加速度情報 引数 unit が RIGHT_ARM or LEFT_ARM の場合： 指令加速度　単位[deg/s^2] 引数 unit が LUMBER の場合： 指令加速度　単位[deg/s^2] 引数 unit が VEHICLE の場合： 移動台車の並進加速度　単位[mm/s^2]、回転加速度 単位[deg/s^2]

戻り値：

値	説明
OK	成功。
STATUS_ERR	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正。

備考：
- Busy 状態または Pause 状態で新たなモーション命令を受けた場合は、その命令は破棄して実行しない（スタックしない）。エラー（STATUS_ERR）となる。
- 本オペレーション指令に対する応答はブロックせず、モーション処理を開始したらすぐにリターンする。
- type は制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。

2.1.4.13 moveUnitRel<2>

機能：
指定したユニットの各軸を指定された位置（相対位置）に移動する。

宣言：

ReturnID moveUnitRel (
in UniType type,
in DoubleSeq relPos,
in DoubleSeq speed,
in DoubleSeq accel
);

引数：

名前	入力・出力	説明
type	入力	対象となるユニットを指定する UnitType 列挙型の値
relPos	入力	指定ユニットの目標位置情報 引数 unit が RIGHT_ARM or LEFT_ARM の場合： 関節各軸の指令位置　単位[deg] 引数 unit が LUMBER の場合： 腰軸の指令位置　単位[deg] 引数 unit が VEHICLE の場合： 移動台車の並進位置　x,y 単位[mm]、回転位置　θ 単位[deg]
speed	入力	指定ユニットの目標速度情報 引数 unit が RIGHT_ARM or LEFT_ARM の場合： 指令速度　単位[deg/s] 引数 unit が LUMBER の場合： 指令速度　単位[deg/s] 引数 unit が VEHICLE の場合： 移動台車の並進速度　単位[mm/s]、回転速度　単位[deg/s]
accel	入力	指定ユニットの目標加速度情報 引数 unit が RIGHT_ARM or LEFT_ARM の場合： 指令加速度　単位[deg/s^2] 引数 unit が LUMBER の場合： 指令加速度　単位[deg/s^2] 引数 unit が VEHICLE の場合： 移動台車の並進加速度　単位[mm/s^2]、回転加速度 単位[deg/s^2]

戻り値：

値	説明
OK	成功。
STATUS_ERR	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正。

備考：
- Busy 状態または Pause 状態で新たなモーション命令を受けた場合は、その命令は破棄して実行しない（スタックしない）。エラー（STATUS_ERR）となる。
- 本オペレーション指令に対する応答はブロックせず、モーション処理を開始したらすぐにリターンする。
- type は制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。

2.1.4.14 moveCPHoldAbs

機能：

制御点の位置を保持したまま冗長軸を原点フレームに基づく絶対指令で動作する。

宣言：

ReturnID moveCPHoldAbs (
in RightLeft type,
in RedundantPos absPos,
in RedundantSpeed speed,
in RedundantAccel accel
);

引数：

名前	入力・出力	説明
type	入力	対象が右アームか左アームかを指定する RightLeft 列挙型の値
absPos	入力	動作目標位置を絶対値で指定する RedundantPos 構造体型の値
speed	入力	動作速度を指定する RedunadantSpeed 構造体型の値
accel	入力	動作加速度を指定する RedundantAccel 構造体型の値

戻り値：

値	説明
OK	成功。
STATUS_ERR	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正。

備考：

- Busy 状態または Pause 状態で新たなモーション命令を受けた場合は、その命令は破棄して実行しない（スタックしない）。エラー（STATUS_ERR）となる。
- 本オペレーション指令に対する応答はブロックせず、モーション処理を開始したらすぐにリターンする。
- type は制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。
- 本動作は、selectRedundantAxes オペレーションによる冗長軸設定の影響は受けない（全冗長軸が有効）。引数で与えられた指令に基づいて動作する。

2.1.4.15 moveCPHoldRel

機能：

制御点の位置を保持したまま冗長軸を原点フレームに基づく相対指令で動作する。

宣言：

```
ReturnID moveCPHoldRel (
        in RightLeft type,
        in RedundantPos relPos,
        in RedundantSpeed speed,
        in RedundantAccel accel
        );
```

引数：

名前	入力・出力	説明
type	入力	対象が右アームか左アームかを指定する RightLeft 列挙型の値
relPos	入力	動作目標位置を相対値で指定する RedundantPos 構造体型の値
speed	入力	動作速度を指定する RedunadantSpeed 構造体型の値
accel	入力	動作加速度を指定する RedundantAccel 構造体型の値

戻り値：

値	説明
OK	成功。
STATUS_ERR	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正。

備考：

- Busy 状態または Pause 状態で新たなモーション命令を受けた場合は、その命令は破棄して実行しない（スタックしない）。エラー（STATUS_ERR）となる。
- 本オペレーション指令に対する応答はブロックせず、モーション処理を開始したらすぐにリターンする。
- type は制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。
- 本動作は、selectRedundantAxes オペレーションによる冗長軸設定の影響は受けない（全冗長軸が有効）。引数で与えられた指令に基づいて動作する。

2.1.4.16 moveGripper

機能：

グリッパを指定された角度へ移動する。

宣言：

ReturnID moveGripper(
in RightLeft type,
in unsigned long angleRatio
);

引数：

名前	入力・出力	説明
type	入力	対象が右アームか左アームかを指定する RightLeft 列挙型の値
angleRatio	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	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正
NOT_SV_ON_ERR	全ての軸のサーボが入っていない

備考：

・ 制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。

2.1.4.17 moveLinearCartesianAbs

機能：

制御点を原点フレームに基づく絶対指令（同次変換行列）で直線補間動作する。

宣言：

```
ReturnID moveLinearCartesianAbs (
        in RightLeft    type,
        in HgMatrix    absHgMat,
        in:CartesianSpeed    speed,
        in CartesianAccel    accel
        in boolean    isHoldOtherArm,
        );
```

引数：

名前	入力・出力	説明
type	入力	対象が右アームか左アームかを指定する RightLeft 列挙型の値
absHgMat	入力	動作目標位置を絶対値で指定する HgMatrix 構造体型の値 単位は並進[mm]、回転要素は無次元
speed	入力	動作速度を指定する CartesianSpeed 構造体型の値
accel	入力	動作加速度を指定する CartesianAccel 構造体型の値
isHoldOtherArm	入力	引数 *type* で指定したアームと反対のアームの動作の種類を指定する boolean 型の値。t rue の場合、反対のアーム制御点の位置・姿勢を維持するように動作する。false の場合、反対のアームはモーション制御しない。

戻り値：

値	説明
OK	成功。
STATUS_ERR	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正。
CANNOT_REACH_TARGET_ERR	目標位置へ到達不可能

備考：

- Busy 状態または Pause 状態で新たなモーション命令を受けた場合は、その命令は破棄して実行しない（スタックしない）。エラー（STATUS_ERR）となる。
- 本オペレーション指令に対する応答はブロックせず、モーション処理を開始したらすぐにリターンする。
- type は制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。
- 本補間動作は、selectRedundantAxes オペレーションによる冗長軸設定に基づいて動作する。

2.1.4.18 moveLinearCartesianRel

機能：

制御点を原点フレームに基づく相対指令（同次変換行列）で直線補間動作する。

宣言：

```
ReturnID moveLinearCartesianRel (
        in RightLeft    type,
        in CommandFrameType    frameType,
        in HgMatrix    relHgMat,
        in CartesianSpeed    speed,
        in CartesianAccel    accel
        in boolean    isHoldOtherArm,
);
```

引数：

名前	入力・出力	説明
type	入力	対象が右アームか左アームかを指定する RightLeft 列挙型の値
frameType	入力	モーションの制御フレームを指定する CommandFrameType 列挙型の値
relHgMat	入力	動作目標位置を相対値で指定する HgMatrix 構造体型の値 単位は並進[mm]、回転要素は無次元。
speed	入力	動作速度を指定する CartesianSpeed 構造体型の値
accel	入力	動作加速度を指定する CartesianAccel 構造体型の値
isHoldOtherArm	入力	引数 *type* で指定したアームと反対のアームの動作の種類を指定する boolean 型の値。t rue の場合、反対のアーム制御点の位置・姿勢を維持するように動作する。false の場合、反対のアームはモーション制御しない。

戻り値：

値	説明
OK	成功。
STATUS_ERR	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正。
CANNOT_REACH_TARGET_ERR	目標位置へ到達不可能

備考：

- Busy 状態または Pause 状態で新たなモーション命令を受けた場合は、その命令は破棄して実行しない（スタックしない）。エラー（STATUS_ERR）となる。
- 本オペレーション指令に対する応答はブロックせず、モーション処理を開始したらすぐにリターンする。
- type は制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。
- 本補間動作は、selectRedundantAxes オペレーションによる冗長軸設定に基づいて動作する。

2.1.4.19 movePTPCartesianAbs

機能：

関節空間において、目標位置を絶対直交座標指定により、直線補間を動作する。

宣言：

ReturnID movePTPCartesianAbs (
in RightLeft type,
in HgMatrix absHgMat,
in double time,
in boolean isHoldOtherArm,
);

引数：

名前	入力・出力	説明
type	入力	対象が右アームか左アームかを指定する RightLeft 列挙型の値
absHgMat	入力	動作目標位置を絶対値で指定する HgMatrix 構造体型の値 単位は並進[mm]、回転要素は無次元
time	入力	移動時間[s]を指定する
isHoldOtherArm	入力	引数 *type* で指定したアームと反対のアームの動作の種類を指定する boolean 型の値。 t rue の場合、反対のアーム制御点の位置・姿勢を維持するように動作する。false の場合、反対のアームはモーション制御しない。 ※将来機能　現状は常に false<3>

戻り値：

値	説明
OK	成功。
STATUS_ERR	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正
NOT_SV_ON_ERR	全ての軸のサーボが入っていない

備考：

- Busy 状態または Pause 状態で新たなモーション命令を受けた場合は、その命令は破棄して実行しない（スタックしない）。エラー（STATUS_ERR）となる。
- 本オペレーション指令に対する応答はブロックせず、モーション処理を開始したらすぐにリターンする。
- type は制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。
- 本補間動作は、selectRedundantAxes オペレーションによる冗長軸設定に基づいて動作する。
- 指定した移動時間で同時起動・同時停止するように動作する。加速度はシステム固定。

2.1.4.20 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：

ReturnID movePTPCartesianRel (
in RightLeft type,
in HgMatrix relHgMat,
in double time,
in boolean isHoldOtherArm,
);

引数：

名前	入力・出力	説明
type	入力	対象が右アームか左アームかを指定する RightLeft 列挙型の値
relHgMat	入力	動作目標位置を相対値で指定する HgMatrix 構造体型の値 単位は並進[mm]、回転要素は無次元。
time	入力	移動時間[s]を指定する
isHoldOtherArm	入力	引数 *type* で指定したアームと反対のアームの動作の種類を指定する boolean 型の値。 t rue の場合、反対のアーム制御点の位置・姿勢を維持するように動作する。false の場合、反対のアームはモーション制御しない。 ※将来機能　現状は常に false<3>

戻り値：

値	説明
OK	成功。
STATUS_ERR	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正
NOT_SV_ON_ERR	全ての軸のサーボが入っていない

備考：

- Busy 状態または Pause 状態で新たなモーション命令を受けた場合は、その命令は破棄して実行しない（スタックしない）。エラー（STATUS_ERR）となる。
- 本オペレーション指令に対する応答はブロックせず、モーション処理を開始したらすぐにリターンする。
- type は制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。
- 本補間動作は、selectRedundantAxes オペレーションによる冗長軸設定に基づいて動作する。
- 指定した移動時間で同時起動・同時停止するように動作する。加速度はシステム固定。

2.1.4.21 openGripper

機能：
グリッパを完全に開く。

宣言：
ReturnID openGripper(
in RightLeft type
);

引数：

名前	入力・出力	説明
type	入力	対象が右アームか左アームかを指定する RightLeft 列挙型の値

戻り値：

値	説明
OK	成功。
STATUS_ERR	オペレーションを受け付け可能な状態でない。
VALUE_ERR	引数が不正
NOT_SV_ON_ERR	全ての軸のサーボが入っていない

備考：
・ 制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。

2.1.4.22 pause

機能：
全モーションモジュールの軸を一時停止し、状態を動作一時停止状態(Pause)にする。

宣言：
ReturnID pause();

引数：
なし

戻り値：
常に OK

備考：
- pause オペレーションを実行した場合、動作中の全モーションモジュール全軸を減速停止し、状態を一時停止状態(Pause)とする。軸が動作していない場合は、即時一時停止状態となる。
- 一時停止状態中に新たな動作指令が与えられた場合、内部に蓄積せず新動作命令は破棄し、resume オペレーションにより一時停止させたモーションのみ復帰動作する。
- 一時停止状態中に pause オペレーションが実行された場合、現在の状態を維持する。
- 一時停止中に abort または stop オペレーションが実行された場合は、全モーションモジュール全軸の動作指令を破棄し、状態を指令待機状態(Ready)とする。

2.1.4.23 resetOriginalFrame

機能：
原点フレームを移動部の現在位置でリセットする。

宣言：
ReturnID resetOriginalFrame ();

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。

備考：

2.1.4.24 resume

機能：
一時停止した動作を再開し、状態を指令実行状態(Busy)にする。

宣言：
ReturnID resume();

引数：
なし

戻り値：
常に OK

備考：
- 動作一時停止状態中に resume オペレーションを実行した場合、一時停止中のモーションモジュールの動作を再開し、モーションモジュールの状態を動作指令実行状態(Busy)とする。
- 動作一時停止状態以外の場合に resume オペレーションを実行した場合、現在の状態を維持する。

2.1.4.25 selectRedundantAxes

機能：
モーション動作時に使用可能な冗長軸を設定する。

宣言：
```
ReturnID selectRedundantAxes(
        in RedundantAxesMask redundancy
        in RedundantLimit limit
);
```

引数：

名前	入力・出力	説明
redundancy	入力	使用可能な冗長軸を指定する RedundantAxesMask 構造体型の値
limit	入力	冗長軸の動作可能範囲を指定する RedundantLimit 構造体型の値

戻り値：

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
VALUE_ERR	引数値が不正。

備考：
なし。

2.1.4.26 servoOff

機能：
全モーションモジュール全軸のサーボ制御を切り、状態を主回路電源オン、サーボ制御オフ状態にする。

宣言：
ReturnID 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：

2.1.4.27 servoOn

機能：
全モーションモジュール全軸のサーボ制御を入れ、状態を指令待機状態にする。

宣言：
ReturnID 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：

2.1.4.28 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：

```
ReturnID setControlPointOffset(
        in RightLeft type,
        in:HgMatrix offset
);
```

引数：

名前	入力・出力	説明
type	入力	右アームか左アームかを指定する RightLeft 列挙型の値
offset	入力	動作目標位置を相対値で指定する HgMatrix 構造体型の値 単位は並進[mm]、回転要素は無次元

戻り値：

値	説明
OK	成功。
STATUS_ERR	失敗。指令可能状態でない。
VALUE_ERR	引数値が不正。
NG	失敗。上記以外の要因で失敗。

備考：
・ 制御対象のアームが１つしかない場合は RIGHT を指定する。LEFT を指定した場合はエラー（VALUE_ERR）となる。

2.1.4.29 stop

機能：
現在の動作を中止し減速停止する。蓄積された動作指令がある場合は全て破棄する。状態を Ready にする。

宣言：

```
ReturnID stop();
```

引数：
なし

戻り値：
常に OK

備考：
なし

3 IDL

本 RTC の IDL を以下に示す。

```
/*
 Integrated Motion component interface
 Yaskawa Electric Corporation 2010
 rev. 2011
*/

#ifndef INTEGRATEDMOTION_INTERFACE_
#define INTEGRATEDMOTION_INTERFACE_

enum AlarmType {
	FAULT,
	WARNING,
	UNKNOWN
};

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

typedef sequence<Alarm> AlarmSeq;

typedef double HgMatrix[3][4];

struct CarPosWithElbow {
	HgMatrix carPos;
	double elbow;
	unsigned long structFlag;
};

struct CartesianAccel {
	double translation;
	double rotation;
};

struct CartesianSpeed {
	double translation;
	double rotation;
};

enum CommandFrameType {
	BASE_FRAME,
	TOOL_FRAME
};
```

```
typedef sequence<double> DoubleSeq;

enum FrameType {
        ORIGINAL_FRAME,
        CURRENT_FRAME
};

struct LimitValue {
        double upper;
        double lower;
};

struct RedundantAccel {
        double elbow;
        double lumbarRx;
        double lumbarRy;
        double lumbarRz;
        double vehicleTranslation;
        double vehicleRotation;
}

struct RedundantAxesMask {
        boolean elbow;
        boolean lumbarRx;
        boolean lumbarRy;
        boolean lumbarRz;
        boolean vehicleX;
        boolean vehicleY;
        boolean vehicleTheta;
}

struct RedundantLimit {
        LimitValue rightElbow;
        LimitValue leftElbow;
        LimitValue lumbarRx;
        LimitValue lumbarRy;
        LimitValue lumbarRz;
        LimitValue vehicleX;
        LimitValue vehicleY;
        LimitValue vehicleTheta;
}

struct RedundantPos {
        double elbow;
        double lumbarRx;
        double lumbarRy;
        double lumbarRz;
        double vehicleX;
        double vehicleY;
        double vehicleTheta;
}
```

```
struct RedundantSpeed {
        double elbow;
        double lumbarRx;
        double lumbarRy;
        double lumbarRz;
        double vehicleTranslation;
        double vehicleRotation;
}

struct ReturnID {
        long id;
        string comment;
}

enum RightLeft {
        RIGHT,
        LEFT
}

enum UnitType {
        RIGHT_ARM,
        LEFT_ARM,
        LUMBER,
        VEHICLE
};
```

```
interface IntegratedMotionInterface {

    ReturnID abort();
    ReturnID clearAlarms();
    ReturnID closeGripper(in RightLeft type);
    ReturnID getActiveAlarm(out AlarmSeq alarms);
    ReturnID getFeedbackPosCartesian(in RightLeft type, in FrameType frameType,
                                                    out CarPosWithElbow pos);
    ReturnID getFeedbackPosJoint(in UnitType type, out DoubleSeq position);
    ReturnID getGripperPos(in RightLeft type, out unsigned long angleRatio);
    ReturnID getState(out unsigned long statusId, out string message);
    ReturnID getVersion(out string version);
    boolean isMoving();
    boolean isServoOn();
    ReturnID moveUnitAbs(in UnitType unit, in DoubleSeq absPos, in DoubleSeq speed,
                                in DoubleSeq accel);
    ReturnID moveUnitRel(in UnitType unit, in DoubleSeq relPos, in DoubleSeq speed,
                                in DoubleSeq accel);
    ReturnID moveCPHoldAbs(in RightLeft type, in RedundantPos absPos,
                                    in RedundantSpeed speed, in RedundantAccel accel);
    ReturnID moveCPHoldRel(in RightLeft type, in RedundantPos relPos,
                                    in RedundantSpeed speed, in RedundantAccel accel);
    ReturnID moveGripper(in RightLeft type, in unsigned long angleRatio);
    ReturnID moveLinearCartesianAbs(in RightLeft type, in HgMatrix absHgMat,
                            in CartesianSpeed speed, in CartesianAccel accel, in boolean isHoldOtherArm);
    ReturnID moveLinearCartesianRel(in RightLeft type, in CommandFrameType frameType,
in HgMatrix relHgMat, in CartesianSpeed speed, in CartesianAccel accel, in boolean isHoldOtherArm);
    ReturnID movePTPCartesianAbs(in RightLeft type, in HgMatrix absHgMat,
                                    in double time, in boolean isHoldOtherArm);
    ReturnID movePTPCartesianRel(in RightLeft type, in HgMatrix relHgMat,
                                    in double time, in boolean isHoldOtherArm);
    ReturnID openGripper(in RightLeft type);
    ReturnID pause();
    ReturnID resetOriginalFrame();
    ReturnID resume();
    ReturnID selectRedundantAxes(in RedundantAxesMask redundancy, in RedundantLimit limit);
    ReturnID servoOff();
    ReturnID servoOn();
    ReturnID setControlPointOffset(in RightLeft type, in HgMatrix offset)
    ReturnID stop();

};

#endif
```