YASKAWA

移動ユニット RTC Ver.2.0
インターフェース仕様書

YASKAWA

資料番号：RB1400031
改版：第１版

版	年　月	変 更 内 容
初版	2010/04/07	新規作成

【 目 次 】

はじめに

本書は、移動ユニットのインターフェース仕様書である。移動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。

1　インターフェース仕様

移動ユニット RTC の IDL（付録参照）において定義されるデータ型一覧を表 1 に、インターフェースのオペレーション一覧を表 2 に示します。

表 1　データ型一覧

No	データ型	概　要
1	Acceleration	Velocity 型を有するデータ型
2	Alarm	アラーム情報を有する構造体
3	AlarmType	アラームかワーニングかを示す列挙型
4	Point	並進位置情報を有する構造体
5	Position	並進位置・姿勢情報を有する構造体
6	SeqAlarm	Alarm 構造体型をデータに持つシーケンス型
7	SeqShort	Short 型の情報を有するシーケンス型
8	Velocity	並進と回転の速度情報を有する構造体

表 2　移動ユニット RTC インターフェースのオペレーション

No	オペレーション名	概　要
1	clearAlarm	アラームクリア
2	getAccelerationLimit	最大加速度の取得
3	getActiveAlarm	アラーム情報の取得
4	getBatteryVoltage	バッテリ電圧の取得
5	getDefaultAcceleration	起動時加速度の取得
6	getDefaultVelocity	起動時速度の取得
7	getIFVersion	RTC のバージョン情報を取得
8	getPosition	現在位置の取得
9	getServoStatus	モータのサーボ制御入切状態を取得
10	getState	ユニット状態の取得
11	getVelocityLimit	最大速度の取得
12	getVersion	ファームウェアのバージョン情報の取得
13	getInput	内蔵 IO モジュールの入力値の取得
14	jogContinuousRel	jog 走行
15	moveCircularRel	円弧運動
16	moveContinuousRel	相対位置として指定された目標位置・姿勢に連続移動
17	moveCruiseAbs	絶対位置として指定された目標位置に向かって滑らかな曲線軌道で移動
18	moveForward	前方向又は、後方へ移動
19	moveLinearAbs	絶対位置として指定された目標位置・姿勢に移動
20	moveLinearRel	相対位置として指定された目標位置・姿勢に移動
21	moveTurn	指定した角度だけ回転
22	setOutput	内蔵 IO モジュールの出力値の設定
23	pause	一時停止
24	resume	一時停止状態から復帰
25	setAcceleration	加速度を設定
26	setJogTimeout	jog 走行時に速度指令を受けてから減速を開始するまでの時間を設定
27	setJoyStick	ジョイスティックの使用／不使用を設定
28	setPosition	現在位置を再定義
29	setPower	主回路電源の入切
30	setServo	サーボ制御の入切
31	setTouchSensor	接触センサの有効／無効を設定

32	setVelocity	速度を設定
33	stop	現在の動作を中止し、減速停止
34	unlock	接触検出停止状態から復帰

1.1 モジュール宣言：

モジュール宣言は使用しない。

1.2 インターフェース宣言：

インターフェース名は VehicleService とする。

1.3 データ型

移動ユニット RTC で定義されるデータ型について述べる。

1.3.1 Acceleration

概要

構造体 Velocity を用いて加速度情報用にしたデータ型。

定義

```
typedef    Velocity Acceleration;
```

備考

単位は並進 mm/sec^2、回転 degree/sec^2

1.3.2 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。

1.3.3 Alarm

概要

アラーム情報を有する構造体。

定義

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

備考

Alarm 構造体型の持つメンバを表 3 に示します。

表 3　Alarm 構造体メンバ

メンバ	内容
code	発生した FAULT/WARNING の数値コードを示します。
type	発生したアラームが FAULT か WARNING か UNKNOW かを示します。 AlarmType 列挙型
description	発生したアラーム(FAULT/WARNING/UNKNOW)の内容を説明する文字列

1.3.4 Point

概要

並進位置情報を有する構造体。

定義

```
struct Point {
   double x;
   double y;
};
```

備考

Point 構造体型の持つメンバを表 4 に示します。

表 4　Point 構造体メンバ

メンバ	内容
x	並進位置　x[mm]
y	並進位置　y[mm]

1.3.5 Position

概要

並進位置・姿勢情報を有する構造体。

定義

```
struct Position {
   double x;
   double y;
   double theta;
};
```

備考

Position 構造体型の持つメンバを表 5 に示します。

表 5　Position 構造体メンバ

メンバ	内容
x	並進位置　x[mm]
y	並進位置　y[mm]
theta	回転姿勢[deg]

1.3.6 SeqAlarm

概要

Alarm 構造体型をデータに持つシーケンス型。

定義

typedef sequence<Alarm> SeqAlarm;

備考

なし。

1.3.7 SeqShort

概要

基本データ型である short 型の情報を有するシーケンス型。

定義

typedef sequence< short > SeqShort;

備考

なし。

1.3.8 Velocity

概要

並進と回転の速度情報を有する構造体。

定義

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

備考

Velocity 構造体型の持つメンバを表 6 に示します。

表 6 Velocity 構造体メンバ

メンバ	内容
translation	並進速度
rotation	回転速度

1.4　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。

1.4.1 clearAlarm

機能

発生した全てのアラームを解除します。

宣言

boolean getState ();

引数

なし。

戻り値

常に真を返します。

備考

なし。

1.4.2 getAccelerationLimit

機能

移動ユニット許容最大加速度設定値を取得します。

宣言

boolean getAccelerationLimit out Acceleration limitAccel);

引数

名前	入力・出力	説明
limitAccel	出力	許容最大並進加速度[mm/sec^2]と許容最大回転加速度[degree/sec^2]

戻り値

常に真を返します。

備考

なし。

1.4.3 getActiveAlarm

機能

発生中の警報を取得します。

宣言

boolean getActiveAlarm(in short numOfAlarm, out short numOfActiveAlarm, out SeqAlarm activeAlarms);

引数

名前	入力・出力	説明
numOfAlarm	入力	取得する警報の最大個数
numOfActiveAlarm	出力	発生中の警報の個数
activeAlarms	出力	発生中のアラーム列

戻り値

常に真を返します。

備考

出力引数 numOfActive は警報発生数に等しくなります(入力引数 numOfAlarm の影響は受けません)。出力引数 activeAlarms には、入力引数 numOfAlarm で指定された個数を超えない範囲で発生した警報が格納されます。実際に格納された警報の数は sequence 型が提供する length()メソッドで取得できます。

1.4.4 getBatteryVoltage

機能

内蔵蓄電池の出力電圧を取得します。

宣言

boolean getBatteryVoltage (out short voltage);

引数

名前	入力・出力	説明
voltage	出力	蓄電池出力電圧[1][×10 V]

戻り値

常に真を返します。

1.4.5 getDefaultAcceleration

機能

移動ユニット起動時の加速度標準値を取得します。

宣言

boolean getDefaultAcceleration(out Acceleration defaultAccel);

引数

名前	入力・出力	説明

[1] 実際の電圧が５５．０Ｖの場合、550.0 が返ってきます。

defaultAccel	出力	起動時標準並進加速度[mm/sec^2]と起動時標準回転加速度[degree/sec^2]

戻り値

常に真を返します。

備考

なし。

1.4.6 getDefaultVelocity

機能

移動ユニット起動時の速度標準値を取得します。

宣言

boolean getDefaultVelocity(out Velocity defaultVel);

引数

名前	入力・出力	説明
defaultVel	出力	起動時標準並進速度[mm/sec]と起動時標準回転速度[degree/sec]

戻り値

常に真を返します。

備考

なし。

1.4.7 getIFVersion

機能

移動ユニット RTC のバージョン情報を取得します。

宣言

boolean getIFVersion (out string ver);

引数

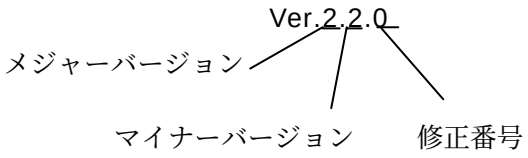
名前	入力・出力	説明
ver	出力	移動ユニット RTC のバージョン情報(RTC VehicleService interface: Ver.###")

戻り値

常に真を返します。

備考

バージョンの数字は以下の意味になります。



1.4.8 getInput

機能

移動ユニット内蔵 IO モジュールの指定した入力端子状態を取得します。

宣言

boolean getInput(in unsigned long ioid, out unsigned long input);

引数

unsigned long ioid：”io.xml”で設定した IO 入力 ID 番号
unsigned long input：上記 ID 番号に対応する入力値

戻り値

真：入力値取得が正常に完了した
偽：指定された ID 番号が存在しないため入力値不定

備考

なし。

1.4.9 getPosition

機能

移動ユニットの現在位置を取得します。

宣言

boolean setPosition (out Position pos);

引数

Position pos 並進位置・姿勢情報を有する構造体

戻り値

常に真を返します。

備考

なし。

1.4.10 getServoStatus

機能

指定した車軸駆動用モータのサーボ制御入切状態を取得します。

宣言

boolean getServoStatus(in unsigned long axisId, out boolean status);

引数

unsigned long axisid： 車軸駆動用モータの ID 番号
boolean status： サーボ制御入切状態(真: サーボ制御入, 偽：サーボ制御切)

戻り値

真：指定 ID 番号に対応したモータのサーボ制御状態の取得が正常に完了した
偽：指定された ID 番号が存在しないため状態取得不可能

備考

なし。

1.4.11 getState

機能

移動ユニットの状態を取得します。

宣言

boolean getState (out short state, out string Msg);

引数

short state ユニットの状態を表す状態コード。
String Msg ユニットの状態を表す状態文字列。

戻り値

常に真を返します。

備考

状態コードと状態文字列、意味を表すユニットの状態一覧を表 7 に示します。

表 7 ユニットの状態一覧

状態コード (state)	状態文字列 (Msg)	意味
0x010	Ready	指令待機状態
0x011	Busy	動作指令実行中
0x012	Warning-###	警告発生(不正な指令を受信)
0x013	Alarm-###	警報発生(解除指令受信まで動作指令拒否)
0x014	Stuck	障害物検知・停止状態
0x015	Paused	動作一時停止状態
0x016	Avoid	障害物回避動作中
0x017	Locked	接触検出停止状態(解除指令受信まで動作指令拒否)
0x018	Powered	主回路入、モータ制御切状態
0x019	Unpowered	主回路切状態

1.4.12 getVelocityLimit

機能

移動ユニット許容最大速度設定値を取得します。

宣言

boolean getVelocityLimit (out double velLimitT, out double velLimitR);

引数

double velLimitT (mm/s2),許容最大並進速度
double velLimitR (deg/s2),許容最大回転速度

戻り値

常に真を返します。

備考

なし。

1.4.13 getVersion

機能

RT ユニット制御ファームウェアのバージョン情報を取得します。

宣言

boolean getVersion (out string ver);

引数

string ver ファームウェアのバージョン情報"Ver.###"を格納する変数

戻り値

常に真を返します。

備考

バージョンの数字は以下の意味になります。

1.4.14 moveCircularRel

機能

移動ユニットからの相対位置として指定された点周りに指定した角度だけ円弧運動します。

宣言

boolean moveCircularRel (in Point centerPoint, in double theta);

引数

double centerPoint.x (mm), 移動ユニット固定座標系における円弧中心 x 座標
double centerPoint.y (mm), 移動ユニット固定座標系における円弧中心 y 座標
double theta (deg), 円弧周りの回転角度

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

・移動速度および加速度はメソッド setVelocity(), setAcceleration()でそれぞれ設定される並進・回転の値を何れも上回らないように制御されます。
・Ready 状態でのみ実行可能です。それ以外の場合は設定値を無視し，偽を返します。
・二輪差動型移動ユニットの場合，引数 ”centerX_rel ” は常に零とします。二輪差動型移動ユニットに引数 ”centerX_rel ” に非零の値が入力された場合は設定値を無視し偽を返します。

1.4.15 moveContinuousRel

機能

現在位置・姿勢から移動ユニットからの相対位置として指定された目標位置・姿勢に向かう直線経路に沿って移動します。連続実行した場合, 最後に設定された目標点以外は消去されます。移動中に実行すると，速度が連続的に変化するように加減速しつつ，新たな経路へと移行します。

宣言

boolean moveContinuousRel (in Position pos);

引数

double pos.x (mm), 移動ユニット固定座標系における目標位置 x 座標
double pos.y (mm), 移動ユニット固定座標系における目標位置 y 座標
double pos.theta (deg), 移動ユニット固定座標系における目標姿勢角度

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

移動ユニットのサーボ制御が入状態でのみ実行可能です。サーボ制御が切状態では設定値を無視し，偽を返します。

1.4.16 moveCruiseAbs

機能

環境に固定された絶対座標系で指定された目標位置に向かって滑らかな曲線軌道を描きながら移動します。連続実行した場合，実行回数に等しい目標点列が蓄積され，順に再生されます。途中の目標点では停止せず，滑らかに加減速して最終目標点に向かいます。

宣言

boolean moveCruiseAbs (in Point cruisePoint);

引数

double cruisePoint .x (mm), 絶対座標系における目標位置 x 座標
double cruisePoint .y (mm), 絶対座標系における目標位置 y 座標

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

移動ユニットのサーボ制御が入状態でのみ実行可能です。サーボ制御が切状態では設定値を無視し，偽を返します。

1.4.17 moveForward

機能

前方向に引数で指定した距離だけ移動します。引数が負の値の場合，後方向に動作します。

宣言

boolean moveForward(in double x_);

引数

double x_ (mm), 前方向移動量(負の場合，後方向)

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

Ready 状態以外で実行すると設定値を無視し，偽を返します。

1.4.18 moveJog

機能

指定した速度で走行します。速度は移動ユニット固定座標系(図 1)で定義します。本メソッドが最後に呼ばれてからメソッド setJogTimeout()で設定した時間が経過すると，自動的に減速・停止します。

宣言

boolean moveJog (in double vx_, in double vy_, in double vtheta_);

引数

double vx_ : (mm/s), 移動ユニット固定座標系における x 方向並進速度
double vy_ : (mm/s), 移動ユニット固定座標系における y 方向並進速度
double vtheta_ : (deg/s), 回転角速度

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

- 移動ユニットの最大速度を超えた値は設定できません。また, 零以下の値も設定できません。引数の値が移動ユニット最大速度を超えていた場合，または零以下だった場合は，速度設定を実行せずに偽を返します。
  ・移動距離が短い場合，加減速のため設定速度に達しない場合があります。
  ・移動ユニットのサーボ制御が入状態でのみ実行可能です。サーボ制御が切状態では設定値を無視し，偽を返します。
  ・移動ユニットの種類によって実現不可能な指令が与えられた場合(二輪差動型移動体に対して y 方向の指令が非零であった場合)，設定値を無視し。偽を返します。

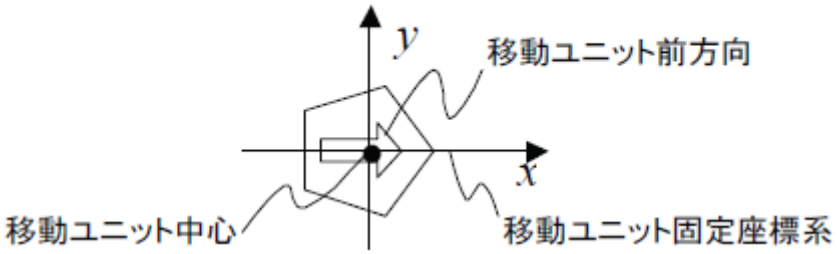


図 1 移動ユニット固定座標系

1.4.19 moveLinearAbs

機能

現在位置・姿勢から環境に固定された座標系において指定された目標位置・姿勢に向かって直線的に移動します。連続実行した場合，実行回数に等しい目標点列が蓄積され，順に再生されます。
直線軌道を保証するため，各目標点では一旦停止します。

宣言

boolean moveLinearAbs (in Position pos);

引数

double pos.x (mm), 絶対座標系における目標位置 x 座標
double pos.y (mm), 絶対座標系における目標位置 y 座標
double pos.theta (deg), 絶対座標系における目標姿勢角度

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

・移動ユニットのサーボ制御が入状態でのみ実行可能です。サーボ制御が切状態では設定値を無視し，偽を返します。
・移動ユニットの種類によって目標位置・姿勢までの途中の動作が異なります。

1.4.20 moveLinearRel

機能

移動ユニットからの相対位置として指定された目標位置・姿勢に向かう直線経路に沿って移動します。

宣言

boolean moveLinearRel (in Position pos);

引数

double pos.x (mm), 移動ユニット固定座標系における目標位置 x 座標
double pos.y (mm), 移動ユニット固定座標系における目標位置 y 座標
double pos.theta (deg), 移動ユニット固定座標系における目標姿勢角度

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

・Ready 状態でのみ実行可能です。それ以外の場合は設定値を無視し，偽を返します。
・移動ユニットの種類によって目標位置・姿勢までの途中の動作が異なります。

1.4.21 moveTurn

機能

引数で指定した角度だけ回転します。

宣言

boolean moveTurn(in double theta_);

引数

double theta_ (deg), 回転角度(ユニット上面からみて反時計方向が正)

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

Ready 状態以外で実行すると設定値を無視し，偽を返します。

1.4.22 pause

機能

一時停止状態に移行します。動作中の場合は，まず設定された最大加速度で停止し，完全停止後に一時停止状態となります。待機中の場合は，即一時停止状態に移行します。

宣言

boolean pause ();

引数

なし。

戻り値

常に真を返します。

備考

stop()とは異なり，動作指令は廃棄されません。また，一時停止状態で入力された動作指令は内部に蓄積され，一時停止状態から復帰した後(resume()実行後)に実行されます。

1.4.23 resume

機能

一時停止状態から復帰します。

宣言

boolean resume ();

引数

なし。

戻り値

常に真を返します。

備考

なし。

1.4.24 setAcceleration

機能

移動時の加速度を設定します。

宣言

boolean setAcceleration (in Velocity accel);

引数

double accel. translation (mm/s2), 並進加速度
double accel. rotation (deg/s2), 回転加速度

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

- 移動ユニット(機種依存)の最大加速度を超えた値は設定できません。また，零以下の値も設定できません。引数の値が移動ユニット最大加速度を超えていた場合，または零以下だった場合は，加速度設定を実行せずに偽を返します。
- 移動中は実行禁止です(対応状態：Busy, Avoid)。移動中に実行すると，設定値を無視して偽を返します。

1.4.25 setDigitalSensorLock

機能

移動ユニット搭載接触センサの有効／無効を設定します。有効時に接触を検出すると，最大加速度で停止し，警報発生状態に移行します。

宣言

boolean setDigitalSensorLock(in boolean OnOff);

引数

boolean OnOff (true/false), 接触センサの有効／無効
真：接触センサ有効
偽：接触センサ無効

戻り値

常に真を返します。

備考

移動ユニットに接触センサが組み込まれている必要があります。

1.4.26 setJogTimeout

機能

速度指定 jog 走行時に速度指令を受けてから減速を開始するまでの時間を設定します。

宣言

boolean setJogTimeout (in short timeout);

引数

short timeout : (msec), 速度指令受信から減速開始までの時間

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

・零以下の値を設定することはできません。引数の値が零以下だった場合，設定値は無視して偽を返します。

1.4.27 setJoyStick

機能

ジョイスティックの使用／不使用を設定します。

宣言

boolean setJoystick (in boolean OnOff);

引数

boolean OnOff (true/false), 真：ジョイスティック使用，偽：ジョイスティック不使用

戻り値

常に真を返します。

備考

正常実行のためには移動ユニットにジョイスティックが接続されている必要があります。
ジョイスティック使用時は，ジョイスティックの入力を目標移動速度として動作します。不使用時は，ジョイスティックの入力は無視されます。
ジョイスティックで入力される目標速度の大きさは，モジュール vehicle のインターフェィス basicCommand に含まれるメソッド setVelocity()で設定された値に制限されます。また，目標加速度は同インターフェィスの setAcceleration()で設定された値に制限されます。
本メソッドは移動ユニットの状態に関係なく実行可能です。移動中にジョイスティック不使用を設定(引数 OnOff=偽で実行)すると，減速停止後に Ready 状態に移行します。減速時の加速度の大きさには，インターフェィス parameterSetting のメソッド setAccelerationLimit()で設定された値が適用されます。

1.4.28 setOutput

機能

移動ユニット内蔵 IO モジュールの指定した出力端子状態を設定します。正常完了時，設定した値は 4ms 以内に指定した出力端子から出力されます。出力値は別の値を上書きするまで維持されます。

宣言

boolean setOutput(in unsigned long ioid, in unsigned long output, in unsigned long mask);

引数

unsigned long ioid：”io.xml”で設定した IO 出力 ID 番号
unsigned long output：上記 ID 番号に対応する出力設定値
unsigned long mask：出力マスク

戻り値

真：出力端子状態の設定が正常に完了した
偽：指定された ID 番号が存在しないため出力値変更なし

備考

IO 出力 ID 番号の 0 は移動ユニット機能のため予約されており，出力値設定はできません。ID 番号 0 の設定値は無視し，偽を返します。

1.4.29 setPosition

機能

移動ユニットの現在位置を再定義します。

宣言

boolean setPosition (in Position pos);

引数

Position pos 並進位置・姿勢情報を有する構造体

戻り値

常に真を返します。

備考

なし。

1.4.30 setPower

機能

モータ駆動用の主回路電源を入切します。

宣言

boolean setPower (in boolean OnOff);

引数：

boolean OnOff : 駆動用電源の入切(真:入，偽: 切)

戻り値

真：引数で指定した処理を実行完了
偽：主回路電源投入に失敗

備考

なし。

1.4.31 setServo

機能

車輪駆動用モータのサーボ制御を入切します。

宣言

boolean setServo (in boolean OnOff);

引数

boolean OnOff : サーボ制御の入切(真:入，偽: 切)

戻り値

真：引数で指定した処理を実行完了

偽：駆動電源が入っていないためサーボ制御入り切り不可能

備考

移動ユニットの状態と setServo()の実行結果の関係を表 8 に示します。

表 8 移動ユニット状態と setServo()の実行結果の関係

		移動ユニットの状態			
		駆動電源入			駆動電源切
		サーボ制御入	サーボ制御切(アラームなし)	サーボ制御切(アラームあり)	サーボ制御切
OnOff の値	真	真, 入状態を維持	真, 制御入	偽, 切状態を維持	偽
	偽	真, 制御切	真, 切状態を維持	真, 切状態を維持	偽, 切状態を維持

1.4.32 setVelocity

機能

移動速度を設定します。

宣言

boolean setVelocity (in Velocity vel);

引数

並進と回転の速度情報を有する構造体。

戻り値

真：引数で指定した値が正常に設定された
偽：引数で指定した値が不正のため設定が拒否された

備考

- 移動ユニット(機種依存)の最大速度を超えた値は設定できません。また，零以下の値も設定できません。引数の値が移動ユニット最大速度を超えていた場合，または零以下だった場合は，速度設定を実行せずに偽を返します。
- 移動距離が短い場合，加減速のため設定速度に達しない場合があります。
- 移動中に実行すると，移動速度を連続かつ動的に変化させて設定値に一致させます。
- 障害物警戒時の速度は，本メソッドで設定した値とメソッド setCautionVelocity()で設定した値の最小値に制限されます。

1.4.33 stop

機能

現在の動作を中止し，設定された最大加速度で減速停止します。停止後は全ての動作指令を廃棄し，待機状態に移行します。

宣言

boolean stop ();

引数

なし。

戻り値

常に真を返します。

備考

なし。

1.4.34 unlock

機能

接触検出停止状態から復帰します。

宣言

boolean unlock ();

引数

なし。

戻り値

常に真を返します。

備考

なし。

付録　IDL

本ＲＴＣの IDL を以下に示す。

```
#ifndef MOBILITY_IDL
#define MOBILITY_IDL

typedef sequence<short> SeqShort;
typedef sequence<long> SeqLong;
typedef sequence<float> SeqFloat;
typedef sequence<double> SeqDouble;

enum AlarmType
{
        unknown
        ,fault
        ,warning
};

struct Point {
  double x;
  double y;
};

struct Position {
  double x;
  double y;
  double theta;
};

typedef struct structAlarm
{
   unsigned long code;
   AlarmType       type;
   string            source;
}Alarm;

typedef sequence<Alarm> SeqAlarm;

struct Velocity {
        double translation;
        double rotation;
};

typedef   Velocity Acceleration;
```

```
interface VehicleService
{
	const string ifVersion = "RTC VehicleService interface: ver.2.0.0";

	boolean clearAlarm();
	boolean getAccelerationLimit( out double accLimitT, out double accLimitR );
	boolean getActiveAlarm( in short numOfAlarm, out short numOfActiveAlarm, out SeqAlarm activeAlarms );
	boolean getBatteryVoltage( out double voltage );
	boolean getDefaultAcceleration( out Acceleration defaultAccel );
	boolean getDefaultVelocity( out Velocity defaultVel );
	boolean getIFVersion(out string version);
	boolean getInput(in unsigned long ioid, out unsigned long input);
	boolean getPosition(out Position pos);
	boolean getServoStatus(in unsigned long axisid, out boolean status);
	boolean getState( out short state, out string Msg );
	boolean getVelocityLimit( out double velLimitT, out double velLimitR );
	boolean getVersion( out string ver );
	boolean moveCircularRel(in Point centerPoint, in double theta);
	boolean moveContinuousRel(in Position pos);
	boolean moveCruiseAbs(in Point cruisePoint);
	boolean moveForward( in double x_ );
	boolean moveJog(in double xVel, in double yVel, in double thetaVel);
	boolean moveLinearAbs(in Position pos);
	boolean moveLinearRel(in Position pos);
	boolean moveTurn( in double theta_ );
	boolean pause();
	boolean resume();
	boolean setAcceleration(in Velocity accel);
	boolean setDigitalSensorLock ( in boolean OnOff);
	boolean setJogTimeout( in short timeout );
	boolean setJoystick( in boolean OnOff );
	boolean setOutput(in unsigned long ioid, in unsigned long output, in unsigned long mask);
	boolean setPosition( in Position pos );
	boolean setPower( in boolean OnOff);
	boolean setServo( in boolean OnOff);
	boolean setVelocity( in Velocity vel );
	boolean stop();
	boolean unlock();

};

#endif
```