機能仕様書

リファレンスハードウェア移動制御モジュール

Ｖｅｒ．1.10

2011 年 10 月 13 日

株式会社　東芝

改版履歴

Ver.	改版日	改版内容
1.00	2011/09/26	新規作成
1.10	2011/10/13	関連文書の入手方法を追記 PositionInput および OmniTo2WD にポート追加, RTC の起動順番を不問に変更

目次

１．　はじめに

１．１．　本書の適用範囲

本書では，ノンホロノミック移動台車を位置制御するモジュールの機能について述べる．このモジュールは，図 1-1 に示すリファレンスハード（以下 RH）3 号機を位置制御するために開発されたが，RH3 号機以外のノンホロノミック移動台車もこのモジュールを使用できる．

機能仕様書（本書）はこのモジュールの機能や動作内容について述べ，操作手順書はこのモジュールのビルド方法や起動方法について述べる．

図 1-1　リファレンスハード 3 号機

１．２．　関連文書

本書と関連する文書を表 1-1 に示す．No.1 と No.2 は，本書内で使用する既存モジュールの仕様書である．

表 1-1 関連文書

No.	文書名	作成コンソ名	備考
1	オープンソース移動知能モジュール群　走行系モジュール機能仕様書[*1]	ロボット知能化ソフトウェア再利用性向上技術開発	Odometry，RefHardRh2 の仕様書 (上記は改良版 Odometry，RefHardRh3 の元となるモジュールである)
2	オープンソース移動知能	ロボット知能化	Navigation，PathFollower，

	モジュール群　経路計画・軌道追従モジュール機能仕様書*2	ソフトウェア再利用性向上技術開発	PositionInput の仕様書
3	操作手順書　リファレンスハードウェア移動制御モジュール	作業知能（社会・生活分野）の開発	本モジュールの操作手順書

*1　http://210.154.184.16/pukiwiki/?plugin=attach&refer=SYS_001_V100&openfile=%C1%F6%B9%D4%B7%CF%A5%E2%A5%B8%A5%E5%A1%BC%A5%EB.pdf

*2　http://210.154.184.16/pukiwiki/?plugin=attach&refer=SYS_001_V100&openfile=%B7%D0%CF%A9%B7%D7%B2%E8%A1%A6%B5%B0%C6%BB%C4%C9%BD%BE%A5%E2%A5%B8%A5%E5%A1%BC%A5%EB.pdf

１．３．　本書を読むにあたって

本書を読むにあたり，RT ミドルウェア，RT コンポーネント（以下 RTC）についての知識が必要となる．

２．　機能仕様

RH3 号機は RH2 号機用のモジュールを用いて制御可能だが，このモジュールを使用するために，RH3 号機の使用者は地図データ，外界センサデータ（天井センサデータなど）を用意する必要がある．

本書のモジュールには，地図データや外界センサなしで簡単に RH を制御するための機能を付加した．RH の使用者は，まず本書のモジュールを用いて簡単な動作を実行し，RH の操作法を早く理解できる．その後，RH2 号機用のモジュールを用いて本格的な自律移動を実現できる．

RH2 号機用のモジュールおよび仕様書は，

http://210.154.184.16/pukiwiki/?SYS_001_V100

からダウンロードできる．

２．１．　機能概要

本モジュールは，ノンホロノミック台車を位置制御する機能を持つ．図　2-1 に位置制御の概要を示す．はじめに，台車が P_0 の位置に停止している．P_0 の位置姿勢(x_0, y_0, θ_0)は，オドメトリが管理している座標系を基準とする．本モジュールに P_1 の位置姿勢(x_1, y_1, θ_1)を入力すると，本モジュールは P_1 へ台車を位置制御する．

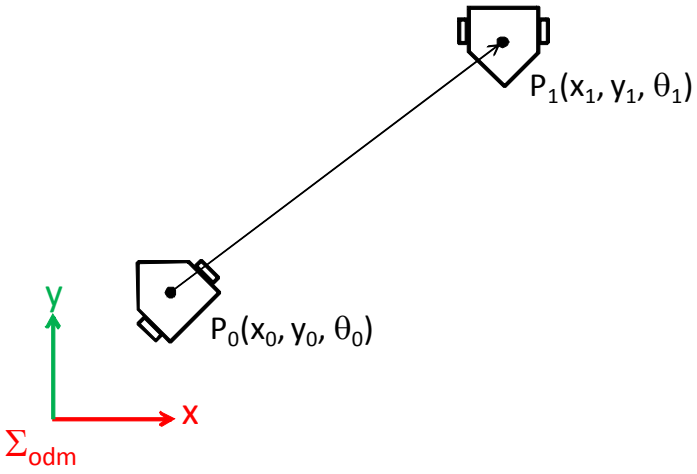

図　2-1　台車の位置制御

２．２．　モジュール構成

本モジュールを構成する RTC を図　2-2 に示す．この構成は，RH3 号機を位置制御するための最小の構成である．この構成では，RH3 号機に搭載されたレーザーレンジファインダ（URG センサ）は使用しない．

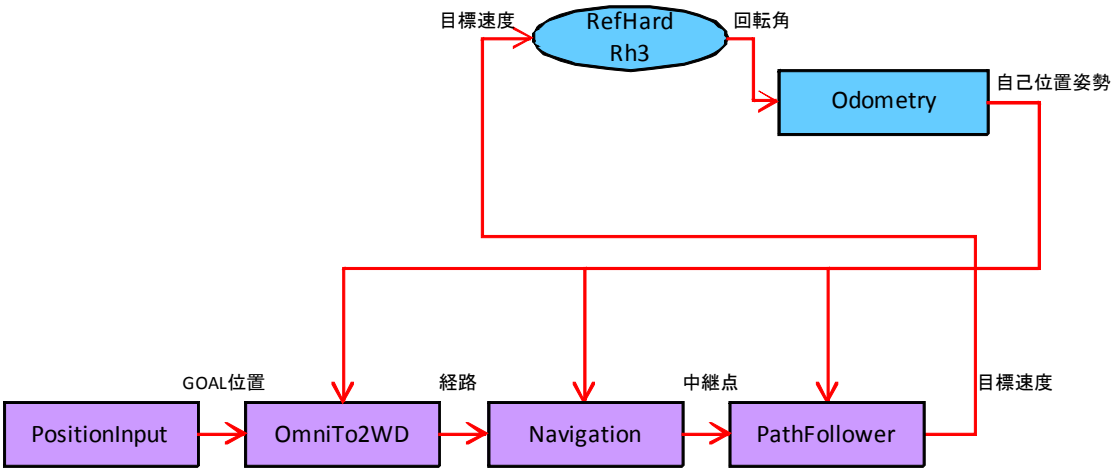

図　2-2　本書で使用する RTC 群

RefHardRh3，Odometry，OmniTo2WD，PositionInput が新規に開発された RTC で，その他の RTC は，RH2 号機用に開発されたものを再利用している．　RT System Editor 上の RTC を図 2-3 に示す．

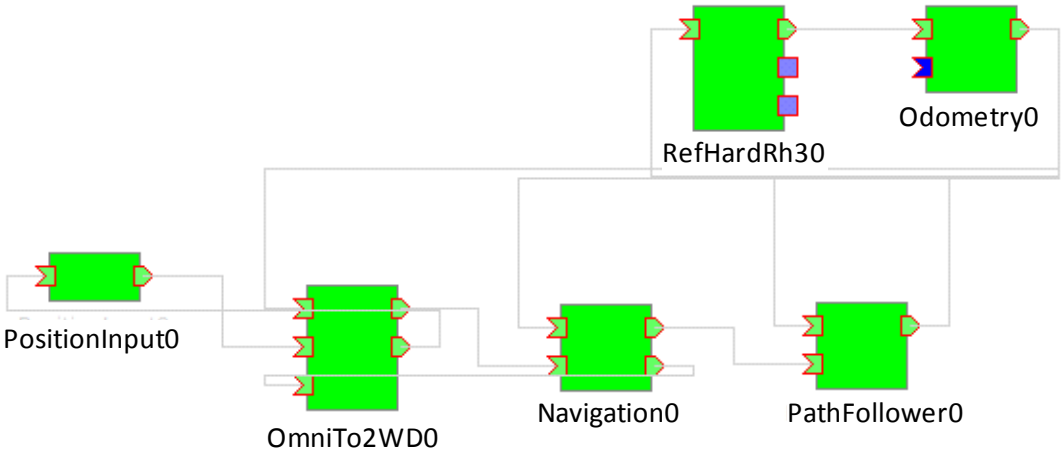

図　2-3　RT System Editor 上の RTC 群

レーザーレンジファインダを用いて障害物を回避する場合や，ゲームパッドを用いて台車の速度をマニュアル制御する場合や，実機の代わりにシミュレータ（OpenHRP）を使用する場合は，図　2-2 に示した構成に，RH2 号機用に開発された RTC を追加する．追加した構成を図　2-4 に示す．

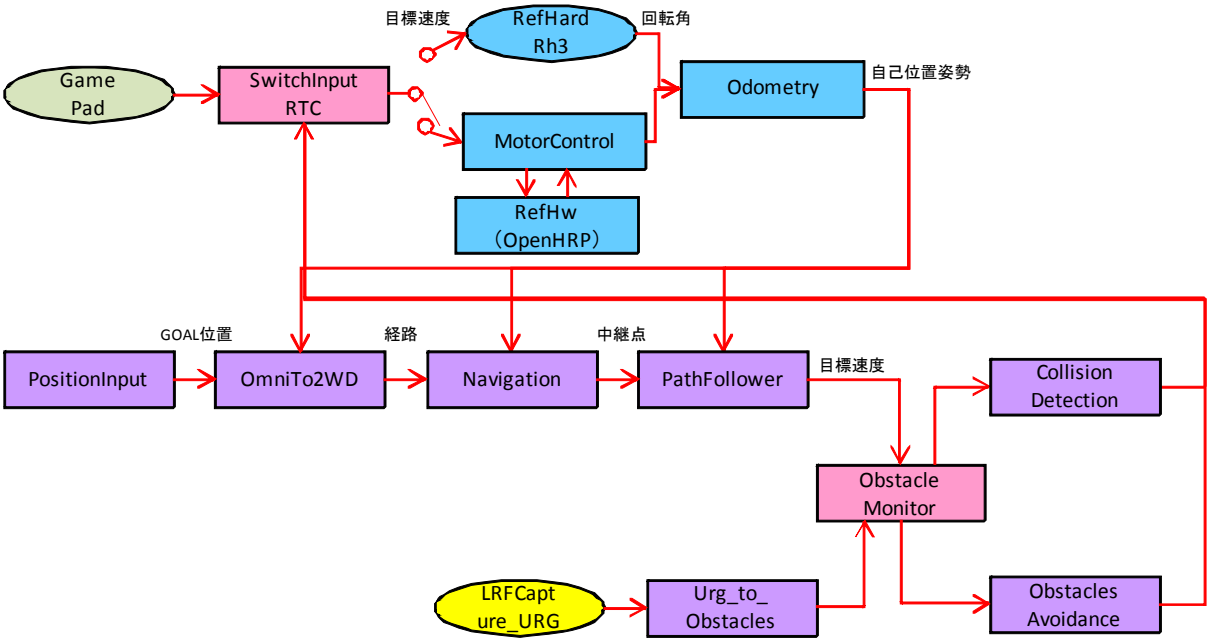

図　2-4　障害物回避機能などを追加した RTC 群

さらに，外界センサによる自己位置推定や，地図を用いたナビゲーションを実施する場合は，RH2 号機用に開発された RTC を用い，RefHardRh2 を RefHardRh3 に変更する．OmniTo2WD や改良版の Odometry は使用しない．このときの RTC 群を図　2-5 に示す．

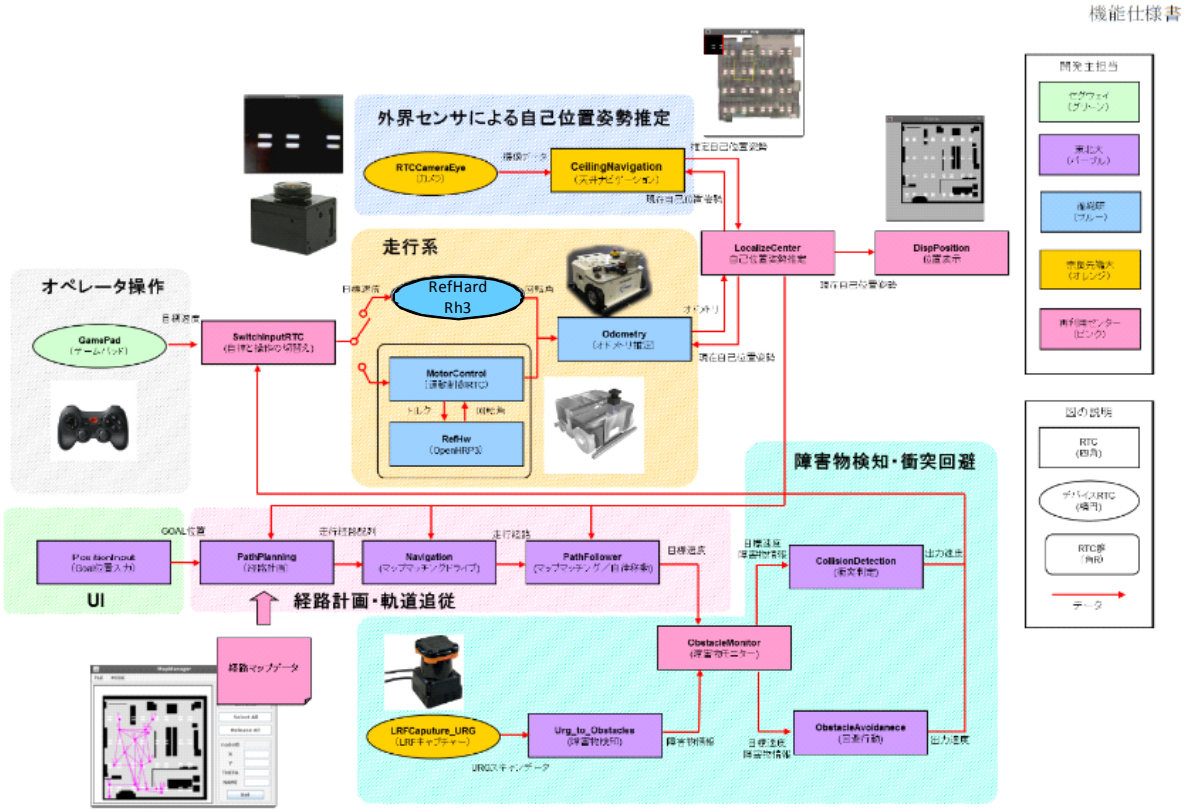

図　2-5　地図を用いたナビゲーションなどを実施する場合の RTC 群

RH2 号機用に開発された RTC については, 仕様書が既に用意されているので, 以下では図 2-2 に示す最小構成について述べる.

図 2-2 に示す RTC の，ビルド方法および起動方法は，操作手順書に記載した.

２．３． ターゲットハードウェア

RefHardRh3 は RH3 号機の車輪速度を制御する RTC で, ゲイン等が RH3 号機用にチューニングされているので，RH3 号機のみで使用できる．RH2 号機には RefHardRh2 を使用する.

その他の RTC は，RH2 号機やノンホロノミック台車（独立 2 輪駆動タイプ）に使用できる.

３． RTC 仕様

３．１． PositionInput

３．１．１． 機能概要

PositionInput は，目標位置姿勢または経路を指令するためのインタフェースである．

この RTC は RH2 号機用に開発された PositionInput（表 1-1 の No.2）を改変して作成した．

３．１．２． 動作環境

PositionInput の動作環境を表 3-1 に示す．

表 3-1 PositionInput の動作環境

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C++
コンパイラ	GCC ver.4.4.3
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	なし

３．１．３． ポート情報

A) データポート（InPort）

表 3-2 PositionInput のデータポート（InPort）

名称	型	データ長	説明
status	TimedState	1	現在の状態

B) データポート（OutPort）

表 3-3 PositionInput のデータポート（OutPort）

名称	型	データ長	説明
.positon	IIS::TimedPose2D	1	目標位置姿勢または中継点 [m][rad]

C) サービスポート（Provider）

なし

D) サービスポート（Consumer）

なし

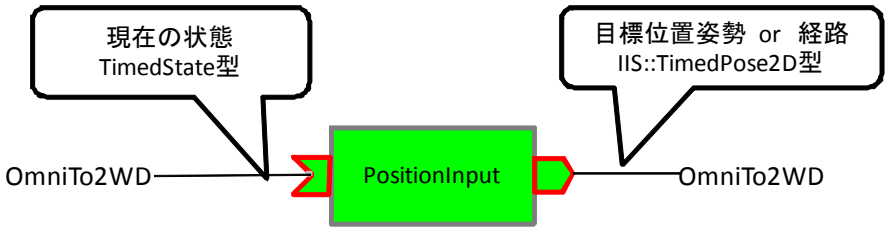

図 3-1 PositionInput のポート

３．１．４． コンフィグレーション

目標位置姿勢を変更するためのコンフィグレーションパラメータが定義されているが，今回はコマンドライン上から目標位置姿勢を入力するため，コンフィグレーションパラメータの値は使用されない．コンフィグレーションパラメータ経由で目標位置姿勢を入力したい場合は，onActivated（）のコメントアウトを外し，onExecute()をコメントアウトすること．

表 3-4 PositionInput のコンフィグレーション

名称	型	デフォルト値	説明
01_x	double	3.4	x の目標値[m]
01_y	double	7.7	y の目標値[m]
01_th	double	-1.57	theta の目標値[rad]

３．１．５． 入出力データフォーマット

TimedState 型の定義を以下に示す．この型は OpenRTM-aist の拡張型である．

```
struct TimedState {
  RTC::Time tm;
  long data;//状態
};
```

data に入力される状態を，以下に示す．

```
enum{
  FINISH = 0,   //ゴールに到達
  ANGLE_CHECK,  //目標姿勢に達したかチェック
  TURNING,  //超信地回転中（中継点での回転時に遷移する）
  TURNING2,  //超信地回転中（ゴール位置での回転時に遷移する）
  GOAL_CHECK,  //目標位置到達のチェック
  SPEED_DOWN,  //並進移動中
  STOPPING    //徐行で並進移動中
};
```

最終目的地に到着すると，FINISH が入力される．なお FINISH は，中継点通過時には入力されない．

IIS::TimedPose2D の定義を以下に示す．この定義は，intellirobot.idl に記載されている．

```
struct TimedPose2D {
RTC::Time tm;
sequence<long> id;
RTC::Pose2D data;
sequence<double> error;
};
```

data の座標系は，Odometry が管理している座標系とする．

目標位置姿勢を指令する場合，図　3-2 に示すように data へ目標位置姿勢を入力する．data 以外の変数には何も入力しない．

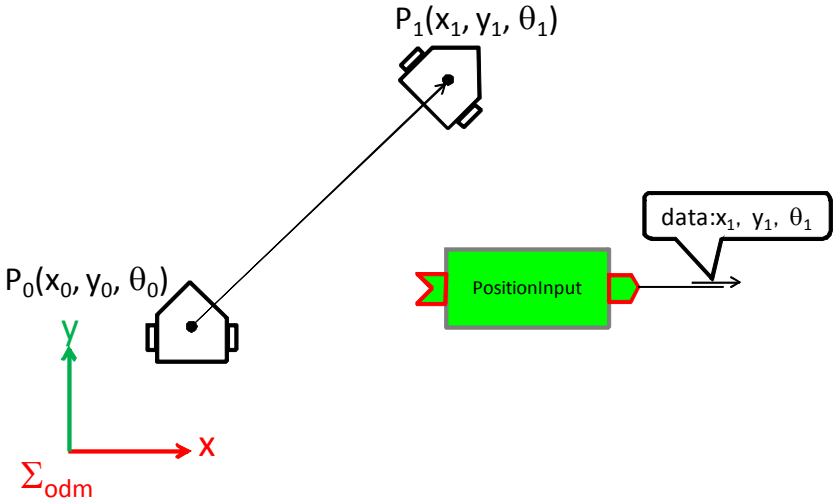

図　3-2　目標位置姿勢のデータフォーマット

経路を指令する場合，図　3-3 に示すように data へ経路を入力し，id の 0 番目に識別番号（開始：100，中間地点：101，終了：102）を入力する．そして，入力データをシーケンシャルに送信する．OmniTo2WD からの ACK は返信されない．OmniTo2WD は，102 の id を受信した時点で動作生成を開始する．

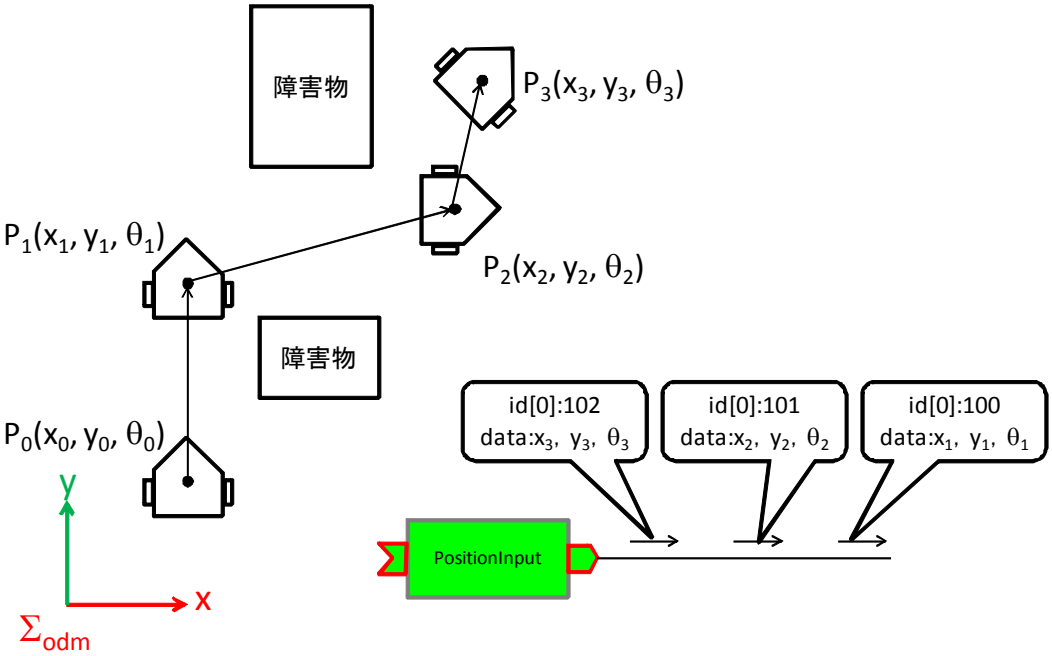

図 3-3 経路のデータフォーマット

３．１．６． フローチャート

onExecute()における動作のフローチャートを図 3-4 に示す.

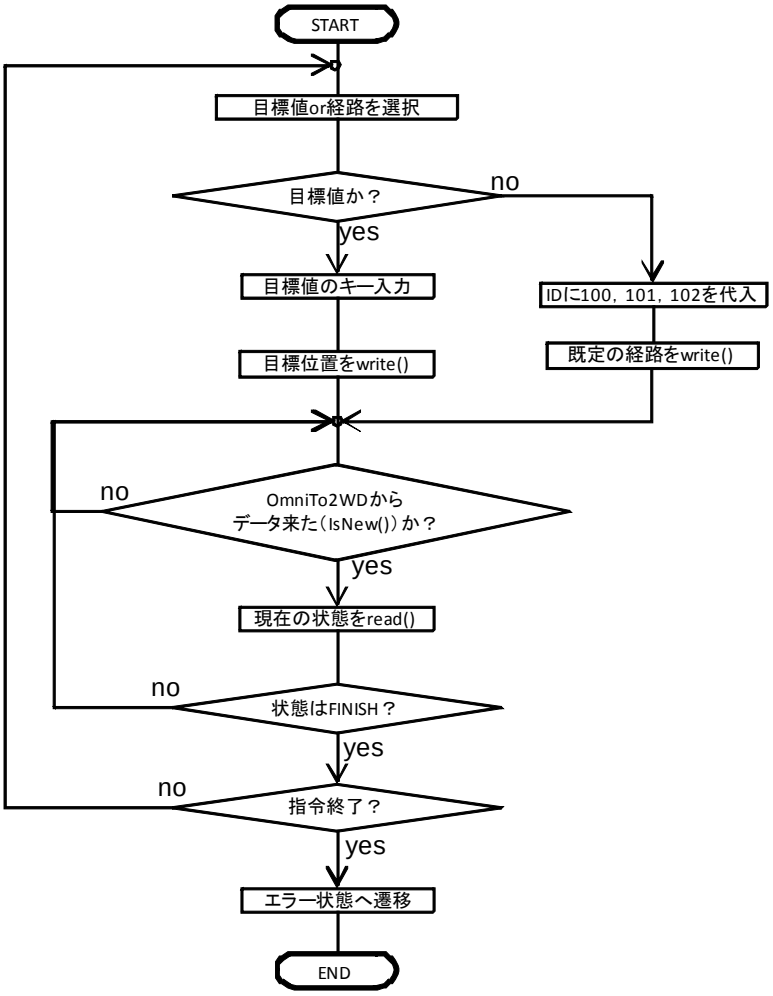

図 3-4 PositionInput のフローチャート

３．１．７． 使用方法

１． PositionInputComp（PositionInput の実行ファイル）を起動する.

２． PositionInput のデータポートを，OmniTo2WD と接続する.

３． PositionInput をアクティベートする.

４． コンソール画面の表示に従い，目標位置姿勢または経路を指令する.

現在固定値の経路を任意の値に変更するには，PositionInput クラスのメンバ変数 defined_path の値を変更する.

３．１．８． 設定ファイル

設定ファイルは不要.

３．２． OmniTo2WD

３．２．１． 機能概要

OmniTo2WD は，任意の位置姿勢へノンホロノミック台車を移動させるために，回転，前

進，回転の３動作からなる経路を計算する．

３．２．２．　経路の計算方法

OmniTo2WD が算出する経路を図　3-5 に示す．この図は，台車が P_0 から P_1 へ移動する際の動作を表す．

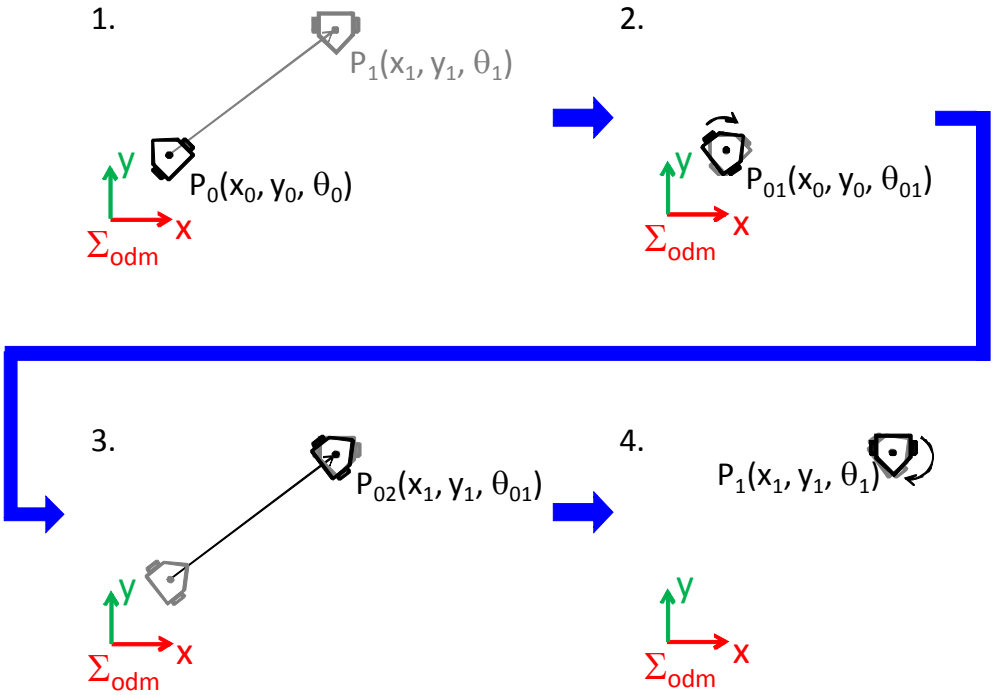

図　3-5　経路

図中 1. が初期状態を表す．2. では，P_0 と P_1 を結んだ直線上に台車を方向転換するため，P_{01} へ超信地回転する．超信地回転時の角度θ_{01} は以下の式で求める．

$$\theta_{01} = \mathrm{atan2}\left(\frac{y_1 - y_0}{x_1 - x_0}\right)$$

atan2 の解の範囲は(-π，π)，atan の解の範囲は(-π/2，π/2)である．台車は(-π，π)の範囲で回転するので，atan2 を使用する．

atan2 の解の範囲について，図　3-6 を用いて補足する．図中では，説明のしやすさのため，P_0 の位置を原点に移動している．atan2 は各象限で異なる値をとり，(-π，π)の範囲を記述できる．一方 atan は，点対称の象限，例えば第一象限と第三象限では，tan の値が同じことから atan の値も同じになる．

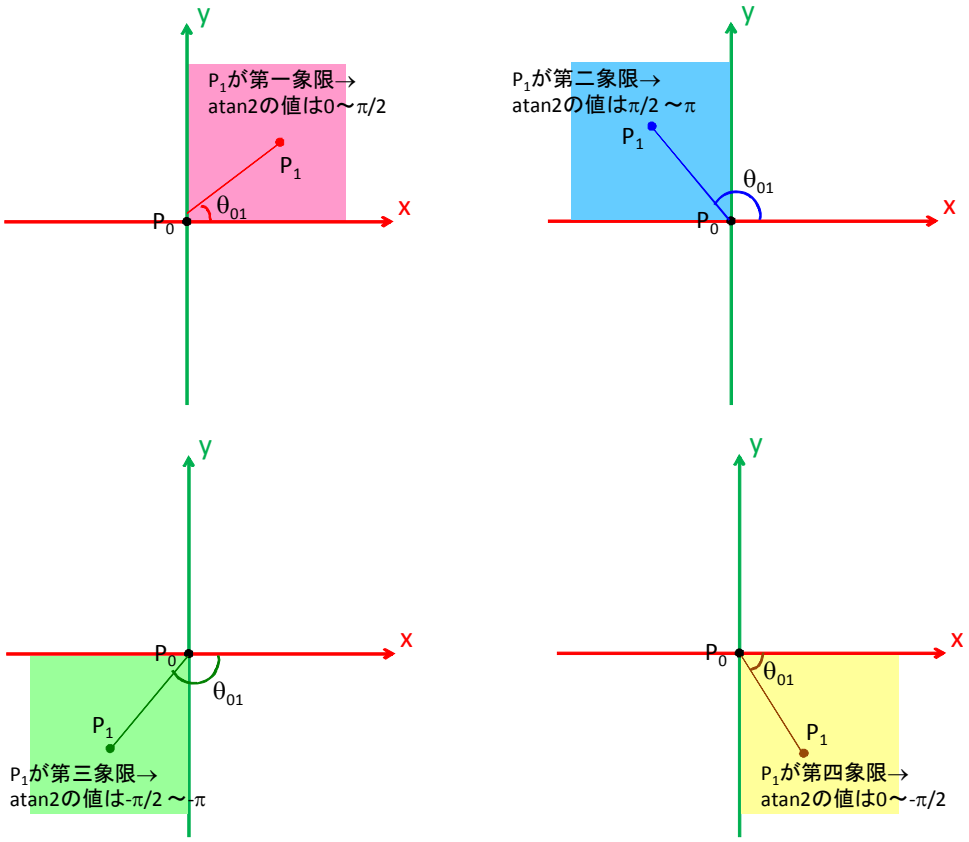

図　3-6　atan2 の値の範囲

図　3-5 の 3. では，台車を前進させ，P_{02} へ移動する．往復移動の復路などの場合，後進したほうがロボットの動作量が少なくなるが，後進はしない．

４．で，目標角度θ_1まで台車を超信地回転させる．

OmniTo2WD は，目標位置姿勢または経路の入力を受け付けている．経路の各中継点について図　3-5 のような経路を計算する場合，回転，前進，回転，回転，前進，回転・・・が算出され，回転動作が冗長になる．そこで，中継点の経路を計算する際は，回転，前進の 2 段階経路を算出する．

３．２．３．　動作環境

OmniTo2WD の動作環境を表　3-5 に示す．

表　3-5　OmniTo2WD の動作環境

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C++
コンパイラ	GCC ver.4.4.3
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	なし

３．２．４．　ポート情報

A)　データポート（InPort）

Inport の一覧を表　3-6 に示す.

表　3-6　OmniTo2WD のデータポート(InPort)

名称	型	データ長	説明
current_position	IIS::TimedPose2D	1	現在位置姿勢[m][rad]
goal	IIS::TimedPose2D	1	目標位置姿勢または中継点[m][rad]
status	TimedState	1	現在の状態

B)　データポート（OutPort）

OutPort の一覧を表　3-7 に示す．min_path の経路長は，目標位置姿勢が入力された場合は図　3-5 の P_0，P_{01}，P_{02}，P_1 の 4 点，経路が入力された場合は中継点数に依存する．

表　3-7　OmniTo2WD のデータポート(OutPort)

名称	型	データ長	説明
min_path	IIS::TimedPath2DSeq	可変	経路
status	TimedState	1	現在の状態

C)　サービスポート（Provider）

なし

D)　サービスポート（Consumer）

なし

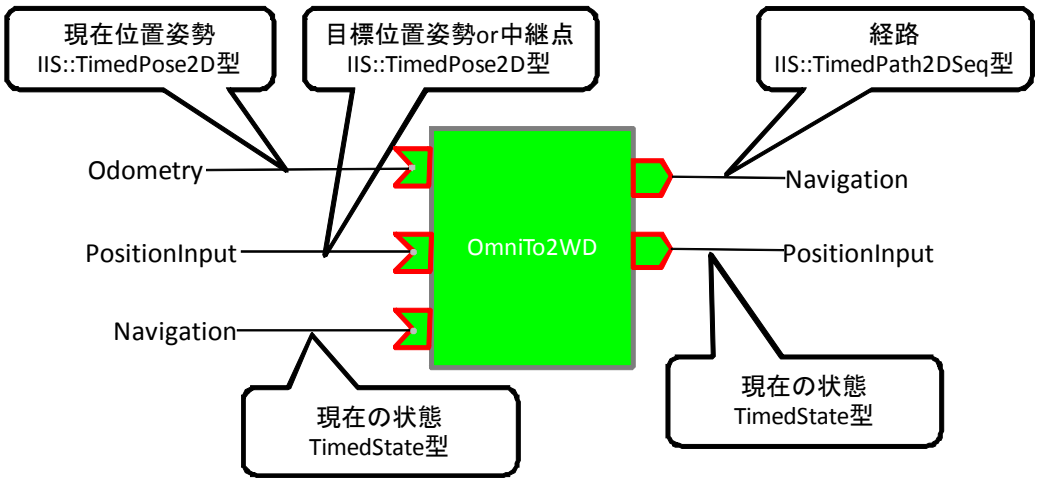

図　3-7　OmniTo2WD のポート

３．２．５．　コンフィグレーション

なし

３．２．６．　入出力データフォーマット

TimedPath2DSeq の定義を以下に示す.

```
struct TimedPath2DSeq {
    RTC::Time tm;
    sequence<long> id;
    sequence<RTC::Pose2D> pose;
    sequence<RTC::Velocity2D> velocity;
    sequence<double> error;
  };
```

TimedPose2D 型，TimedState 型の定義は，３．１．５．を参照のこと.
また，位置姿勢や経路の座標系は，Odometry が管理している座標系に従う.

３．２．７．　フローチャート

onExecute()のフローチャートを図 3-8 に示す.

図 3-8 OmniTo2WD のフローチャート

３．２．８． 使用方法

１． OmniTo2WDComp（OmniTo2WD の実行ファイル）を起動する.

２． OmniTo2WD のデータポートを，PositionInput，Odometry，Navigation と接続する.

３． OmniTo2WD をアクティベートする.

３．２．９． 設定ファイル

独自の設定ファイルは不要.

３．３．　Navigation

３．３．１．　機能概要

Navigation は，入力された経路に沿って台車を動作させるために，各中継点について，走行モードや台車の速度（以後は台車速度と呼ぶ）を付加する RTC である．また，ノンホロノミック台車で実現できない中継点が含まれている場合，台車の速度を変更する場合に，中継点を新規に追加する．

さらに，台車の動作状態（３．１．５．の data 参照）を決定する．

この RTC は，RH2 号機用に開発されたもので，RH3 号機もこの RTC を再利用している．詳細は表　1-1 の No.2 を参照のこと．

３．３．２．　走行モードの定義

走行モードおよび台車速度の定義を以下に示す．この定義は，PathType.h に記載されている．

```
enum{
RUN_FREE = 0,
RUN_STOP,
RUN_LINEFOLLOW,
RUN_TO_POINT,
RUN_CIRCLEFOLLOW,
RUN_SPIN,
RUN_VEL,
RUN_WHEEL_VEL
};
```

RUN_FREE から RUN_SPIN までが走行モード，RUN_VEL と RUN_WHEEL_VEL が台車速度である．OmniTo2WD は，直進と超信地回転のみ指令するので，実行される走行モードは RUN_LINEFOLLOW，RUN_SPIN，RUN_STOP の３つである．台車速度について，RUN_VEL が並進速度，RUN_WHEEL_VEL が回転速度である．

３．３．３．　動作環境

Navigation の動作環境を表　3-8 に示す．

表　3-8　Navigation の動作環境

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C++
コンパイラ	GCC ver.4.4.3
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	なし

３．３．４． ポート情報

A) データポート（InPort）

Inport の一覧を表 3-9 に示す. position には, Odometry から現在位置姿勢が入力される. min_path には，OmniTo2WD から経路が入力される.

表 3-9 Navigation のデータポート(InPort)

名称	型	データ長	説明
position	IIS::TimedPose2D	1	現在位置姿勢[m][rad]
min_path	IIS::TimedPath2DSeq	4	経路[m][rad]

B) データポート（OutPort）

OutPort の一覧を表 3-10 に示す．path は中継点１つを PathFollower へ出力する．その中継点に台車が到達したら，次の中継点を出力する．status は現在の状態を OmniTo2WD へ出力する.

表 3-10 Navigation のデータポート(OutPort)

名称	型	データ長	説明
path	Path	1	中継点
status	TimedState	1	現在の状態

C) サービスポート（Provider）

なし

D) サービスポート（Consumer）

なし

３．３．５． コンフィグレーション

コンフィグレーションの一覧を表 3-11 に示す.

表 3-11 Navigation のコンフィグレーション

名称	型	デフォルト値	説明
judge_radius	double	0.5	目標到達を判断する円の半径[m]
max_vel	double	0.3	並進時の最大速度[m/s]

３．３．６． 入出力データフォーマット

Path 型の定義を以下に示す．Path 型は，OpenRTM-aist の拡張型である.

```
struct Path{
  short type;//走行モード（３．３．２．参照）
  short coordinate;//未使用
```

```
    double x;
    double y;
    double theta;
    double v;
    double w;
    double r;//未使用
    Time tm;
  };
```

IIS::TimedPose2D, TimedState の定義は，３．１．５．を参照のこと．IIS::TimedPath2DSeq の定義は，３．２．６．を参照のこと．

現在位置姿勢や中継点の座標系は，Odometry が管理している座標系とする．

３．３．７．　使用方法

１．　NavigationComp（Navigation の実行ファイル）を起動する．

２．　Navigation のデータポートを，OmniTo2WD ，Odometry，PathFollower と接続する．

３．　Navigation をアクティベートする．

３．３．８．　設定ファイル

独自の設定ファイルは不要．

３．４．　PathFollower

３．４．１．　機能概要

PathFollower は，入力された中継点に沿って台車を動作させるために，台車の制御周期毎の並進速度および回転速度を生成する RTC である．

この RTC は RH2 号機用に開発されたもので，RH3 号機もこの RTC を再利用している．詳細は表　1 -1 の No.2 を参照のこと．

３．４．２．　動作環境

PathFollower の動作環境を表　3 -12 に示す．

表 3-12 PathFollower の動作環境

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C++
コンパイラ	GCC ver.4.4.3
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	なし

３．４．３． ポート情報

A) データポート（InPort）

Inport の一覧を表 3-13 に示す．position には，Odometry から現在位置姿勢が入力される．target_path には，Navigation から中継点が入力される．

表 3-13 PathFollower のデータポート(InPort)

名称	型	データ長	説明
position	IIS::TimedPose2D	1	現在位置姿勢[m][rad]
target_path	Path	1	中継点[m][rad]

B) データポート（OutPort）

OutPort の一覧を表 3-14 に示す．velocity は台車の並進速度および回転速度を出力する．

表 3-14 PathFollower のデータポート(OutPort)

名称	型	データ長	説明
velocity	IIS::TimedVelocity2D	1	台車速度[m/s][rad/s]

C) サービスポート（Provider）

なし

D) サービスポート（Consumer）

なし

３．４．４． コンフィグレーション

コンフィグレーションの一覧を表 3-15 に示す．

表 3-15 PathFollower のコンフィグレーション

名称	型	デフォルト値	説明
control_cycle	double	0.02	制御周期[s]
line_C1	double	0.01	経路追従パラメータ
line_Dist	double	0.05	経路追従パラメータ： 経路からの距離の最大値

名称	型	デフォルト値	説明
line_K1	double	800	経路追従パラメータ： 経路からの距離に関する係数
line_K2	double	300	経路追従パラメータ： 経路からの角度に関する係数
line_K3	double	400	経路追従パラメータ： 角速度に関する係数
max_acc_v	double	0.0003	最大加速度[m/制御周期]
max_acc_w	double	0.1	最大角加速度[rad/制御周期]
max_v	double	1.0	最大速度[m/s]
max_w	double	0.3	最大角速度[rad/s]

台車の加減速度を変更したい場合は，max_acc_v，max_acc_w を変更する．max_acc_v，max_acc_w のデフォルト値は，PathFollower.cpp 内で変更する．表 3-15 の max_acc_v，max_acc_w の値は，RH3 号機をビニール系床材の上で動作させることを想定して設定した．

３．４．５． 入出力データフォーマット

IIS::TimedVelocity2D 型の定義を以下に示す．

```
struct TimedVelocity2D {
  RTC::Time tm;
  sequence<long> id;  //不使用
  RTC::Velocity2D data;  //台車速度
  sequence<double> error;  //不使用
};
```

台車速度は，data に入力される．Velocity2D 型の定義を以下に示す．Velocity2D 型は，OpenRTM-aist の拡張型である．

```
struct Velocity2D{
  double vx;//並進速度[m/s]
  double vy; //不使用
  double va;//回転速度[rad/s]
};
```

ノンホロノミック台車は左右に移動できないので，vy は使用しない．

IIS::TimedPose2D 型の定義は，３．１．５．を参照のこと．Path 型 の定義は，３．３．６．を参照のこと．

現在位置姿勢や中継点の座標系は，Odometry が管理している座標系に従う．

３．４．６．　使用方法

１．　PathFollowerComp（PathFollwer の実行ファイル）を起動する．

２．　PathFollower のデータポートを，Navigation，Odometry，RefHardRh3 と接続する．

３．　PathFollower をアクティベートする．

３．４．７．　設定ファイル

独自の設定ファイルは不要．

３．５．　RefHardRh3

３．５．１．　機能概要

RefHardRh3 は，入力された台車速度を実現するための車輪速度を生成する RTC である．この RTC は，RH の開発元である前川製作所殿よりご提供いただいた．

RefHardRh3 の挙動，インタフェースは RH2 号機制御用 RTC（RefHardRh2，表 1-1 の No.1 参照）と同じである．ただし，RefHardRh2 や RefHardRh3 内で呼ばれている RH 制御ライブラリ(libsvm)のバージョンが RH2 号機と 3 号機とで異なり，ギア比も 2 号機と 3 号機とで異なることから，RefHardRh3 と RefHardRh2 のソースコードは異なる．このため，RH3 号機には RefHardRh3 を使用する．

３．５．２．　動作環境

RefHardRh3 の動作環境を表 3-16 に示す．

表 3-16　RefHardRh3 の動作環境

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C++
コンパイラ	GCC ver.4.4.3
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	なし

３．５．３．　ポート情報

A)　データポート（InPort）

Inport の一覧を表 3-17 に示す．InTargetVelocity には，PathFollower から台車速度が入力される．

表 3-17　RefHardRh3 のデータポート(InPort)

名称	型	データ長	説明
InTargetVelocity	IIS::TimedVelocity2D	1	台車速度[m/s][rad/s]

B) データポート（OutPort）

OutPort の一覧を表 3-18 に示す．OutWheelAngle はエンコーダカウンタが計算した左右車輪の回転角である．

表 3-18 RefHardRh3 のデータポート(OutPort)

名称	型	データ長	説明
OutWheelAngle	TimedDoubleSeq	2	車輪回転角度[rad]

C) サービスポート（Provider）

Provider の一覧を表 3-19 に示す．InventGUIRefHard は，台車の走行開始/走行停止のトリガである．ただし，このトリガを与えずに台車速度を与えても台車が走行を開始し，速度を 0 にすると走行を停止するため，本書では使用していない．BumpRefHard は，台車の走行停止のトリガである．

表 3-19 RefHardRh3 のサービスポート（Provider）

サービス名	インタフェース名	説明
InventGUIRefHard	RefHard	走行開始／走行停止
BumpRefHard	RefHard	走行停止

D) サービスポート（Consumer）

なし

3．5．4． コンフィグレーション

コンフィグレーションの一覧を表 3-20 に示す．「デフォルト値」列内で下線が引かれた値が，RH2 号機の設定値と異なる．

表 3-20 RefHardRh3 のコンフィグレーション

名称	型	デフォルト値	説明
DeviceFileName	string	/dev/ttyUSB0	RS485シリアル通信のためのデバイスファイル名
gain_P	int	10	駆動モーターの位置制御モードでの PID 制御パラメータ
gain_I	int	0	
gain_D	int	0	
gain_VP	int	<u>10</u>	駆動モーターの速度モードでの PID 制御パラメータ
gain_VI	int	0	
gain_VD	int	<u>1</u>	
Velocity_limit	double	10.0	台車の移動速度制限値
Acceleration_limit	double	50.0	台車の移動加速度制限
WheelTread	double	0.441	車輪間の距離

名称	型	デフォルト値	説明
WheelRadiusLeft	double	0.1	左右車輪の半径
WheelRadiusRight	double	0.1	
GearRatio	double	74.91	駆動モーターから車輪までのギア比
EncoderCount	int	2000	駆動モーター1 回転のエンコーダカウント値
EncoderConstant_L	double	1.0	左右車輪のエンコーダ
EncoderConstant_R	double	1.0	

３．５．５．　入出力データフォーマット

IIS::TimedVelocity2D 型の定義は，３．４．５．を参照のこと．

３．５．６．　サービスポート I/F 仕様

インタフェース RefHard の I/F 仕様を表 3-21 に示す．RefHard は，RefHardRh3Pro.idl 内で定義されている．

表 3-21　RefHard の I/F 仕様

関数名	Stop（）
引数	なし
戻り値	なし
説明	走行中の移動台車を停止させる

関数名	Go（）
引数	なし
戻り値	なし
説明	停止中の移動台車を走行させる

３．５．７．　使用方法

１．　RefHardRh3 を起動する．
２．　RefHardRh3 のデータポートを，PathFollower，Odometry と接続する．
３．　RefHardRh3 をアクティベートする．

３．５．８．　設定ファイル

独自の設定ファイルは不要．

３．６．　Odometry

３．６．１．　機能概要

Odometry は，台車の自己位置姿勢を推定する RTC である．推定には，車輪の回転角度を用いる．また，推定する場所は，駆動軸の中心とする．図　3-9 に示すように，RH3 号機の駆動軸の中心は台車中心とは異なる．

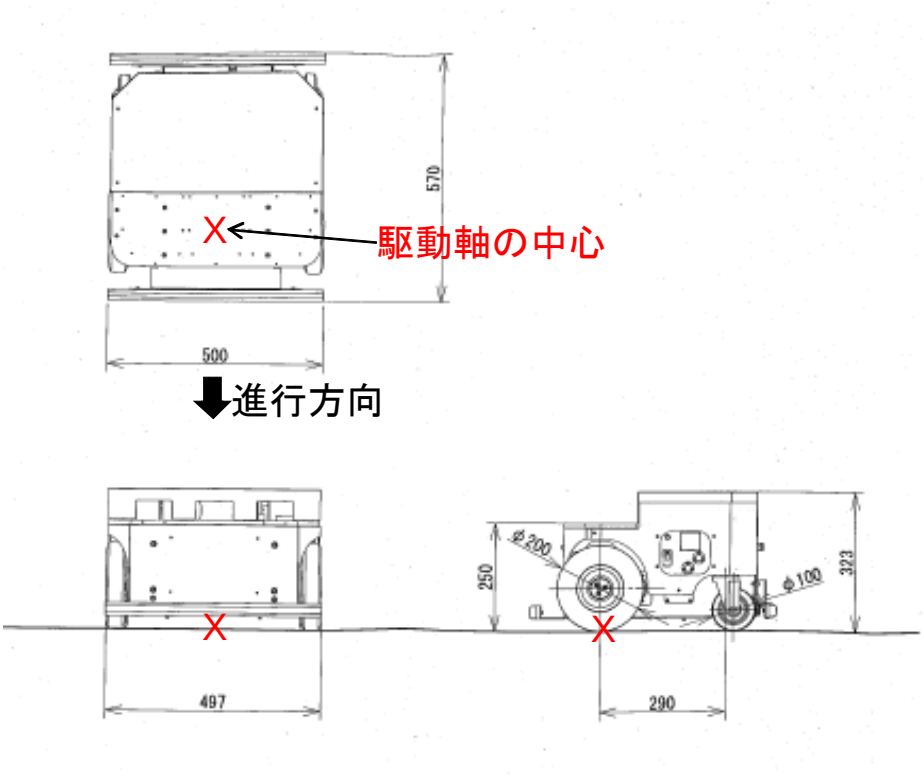

図　3-9　駆動軸の中心

Odometry は，RH2 号機用に開発された Odometry（表　1-1 の No.1）の機能を一部変更したものである．具体的には，外界センサを用いて台車の自己位置姿勢を修正する機能を削除し，車輪回転角のみを用いて自己位置を推定するようにした．I/F は RH2 号機用 Odometry と同じとした．

３．６．２．　動作環境

Odometry の動作環境を表　3-22 に示す．

表　3-22　Odometry の動作環境

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C++
コンパイラ	GCC ver.4.4.3
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	なし

３．６．３． ポート情報

A) データポート（InPort）

Inport の一覧を表 3-23 に示す．CurrentWheelAngle には，RefHardRh3 から車輪回転角度が入力される．LocalizedPosition は，本書では使用していない．

表 3-23 Odometry のデータポート(InPort)

名称	型	データ長	説明
CurrentWheelAngle	TimedDoubleSeq	2	車輪回転角度(絶対値) data[0]：左車輪回転角度[rad] data[1]：右車輪回転角度[rad]
LocalizedPosition	IIS::TimedPose2D	1	外界センサ値より推定された自己位置姿勢（不使用）

B) データポート（OutPort）

OutPort の一覧を表 3-24 に示す.

表 3-24 Odometry のデータポート(OutPort)

名称	型	データ長	説明
OdometryPosition	IIS::TimedPose2D	1	オドメトリによる自己位置姿勢 x：X 位置[m] y：Y 位置[m] theta：姿勢角[rad] id：ID 用配列（未使用） error:エラーコード用配列（未使用） tm：タイムスタンプ[sec][nsec]

C) サービスポート（Provider）

なし

D) サービスポート（Consumer）

なし

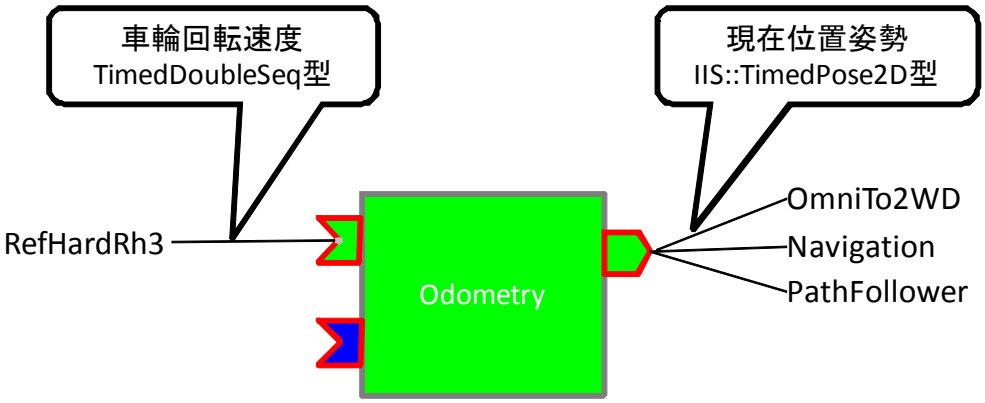

図 3-10 Odometry のポート

3．6．4． コンフィグレーション

コンフィグレーションの一覧を表 3-25 に示す.

表 3-25 Odometry のコンフィグレーション

名称	型	デフォルト値	説明
leftWheelID	int	0	左駆動輪のID 番号
rightWheelID	int	1	右駆動輪の ID 番号
radiusOfLeftWheel	double	0.0995	左駆動輪の車輪半径[m]
radiusOfRightWheel	double	0.0995	右駆動輪の車輪半径[m]
lengthOfAxle	double	0.464	左右駆動輪間の長さ[m]
radiusOfBodyArea	double	0.45	車体全体を円で囲んだ場合の半径[m] 安全を考慮したエリア定義用
initialPose_x	double	0.0	台車のx座標の初期値
initialPose_y	double	0.0	台車のy座標の初期値
initialPose_theta	double	0.0	台車の角度の初期値

自己位置同定の精度を上げるために，車輪半径や駆動輪間の長さを調整する．並進の同定精度を上げる場合は，車輪半径を調整する．回転の同定精度を上げる場合は，駆動輪間長を調整する．表 3-25 の値は，RH3 号機がビニール系の床で動作する場合を想定している．その他の場合の調整例が Odometry.conf に記述されている．

Odometry.conf 内の値を変更すると，コンフィグレーションパラメータのデフォルト値を変更できる．なお，Odometry.cpp 内で設定されている値は Odometry.conf の値に上書きされ，無効となる．

initialPose_x, initialPose_y, initialPose_theta が自己位置姿勢の初期値となる．RH3 号機に次の作業を実行させる時などに，初期値を RTSystemEditor から書き変える様子を図 3-11 に示す.

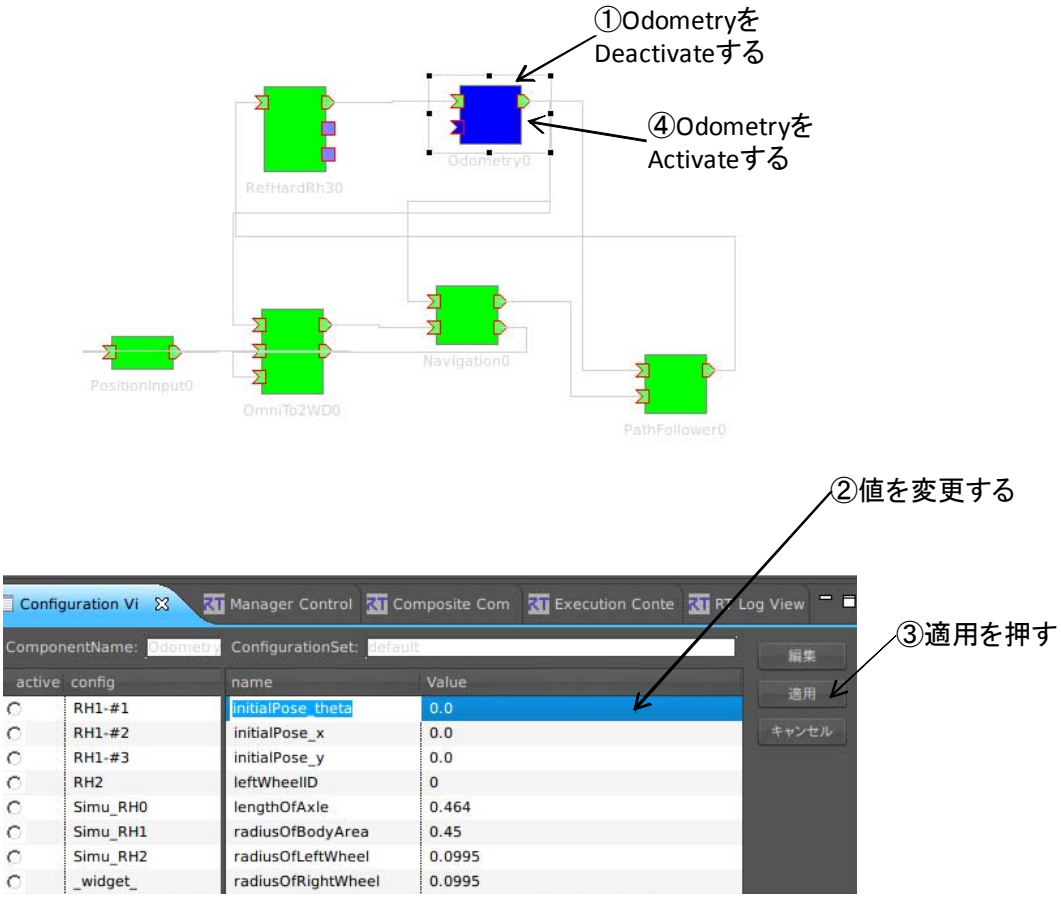

図　3-11　初期位置姿勢の書き換え

Odometry は自己位置同定の履歴を onDeactivated()でクリアし，onActivated()で初期位置姿勢を読み込むため，初期値の変更前に Odometry を Deactivate し，変更後に Activate すること．

３．６．５．　入出力データフォーマット

IIS:: TimedPose2D 型の定義は，３．１．５．を参照のこと．

３．６．６．　使用方法

１．　Odometry を起動する．

２．　Odometry のデータポートを，RefHardRh3，PathFollower，Navigation，OmniTo2WD と接続する．ポートは一対多接続できる．

３．　Odometry をアクティベートする．

Odometry の OnActivate()内で，コンフィグレーションパラメータに記載された台車の初期位置姿勢を読み込む．もし台車の位置姿勢を初期値に戻したい場合は，Odometry を Deactivate し，再度 Activate する．

３．６．７．　設定ファイル

独自の設定ファイルは不要.

４．　特記事項

本モジュールをご利用される場合には，以下の記載事項・条件にご同意いただいたものとします．

本モジュールのライセンスは Eclipse Public License(EPL)に従います．利用条件の詳細については，下記サイトを参照ください．なお，本モジュールは利用条件に同意した場合にのみ利用可能となっており，本モジュールを利用した時点でライセンス条項に同意したものとみなします．

Eclipse Public License http://www.eclipse.org/legal/epl-v10.html