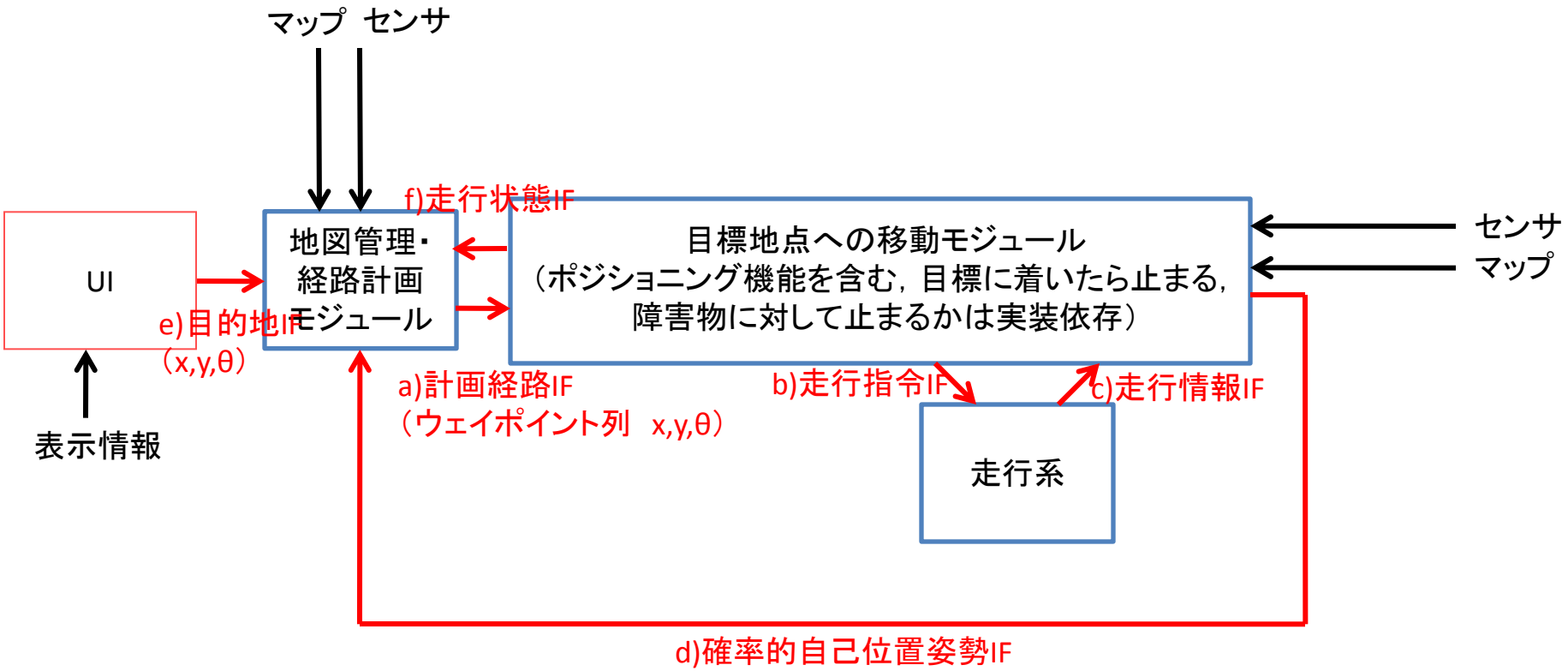
移動SWG　共通IF案

10／10／08

移動サブWG:基本モジュール接続図（10/4/9）

タスク：地図のある室内の既知の場所から指定された任意の場所に自律移動



※赤い部分のIFを揃える

コンポーネントの共通構成及び共通IF型を１０／４／９合意した

a) 計画経路IF

```
#include "BasicDataType.idl"
#include "ExtendedDataTypes.idl"      //RTM1.0において定義
module IIS {//IIS2.idl
  struct TimedPose2D { //推定位置IF
    RTC::Time tm;
    sequence<long> id;
    RTC::Pose2D data;        //RTM1.0において定義
    sequence<double> error;
  };
  struct TimedPath2DSeq { //計画経路IF
    RTC::Time tm;
    sequence<long> id;
    sequence<RTC::Pose2D> pose;
    sequence<RTC::Velocity2D>velocity;        //進行方向
    sequence<double> error;
  };
};
```

```
module RTC {//ExtendedDataTypes.idl
 struct Point2D    {
    /// X coordinate in metres.
    double x;
    /// Y coordinate in metres.
    double y;
  };
  struct Pose2D   {
    /// 2D position.
    Point2D position;
    /// Heading in radians.
    // x軸から反時計回りに0[rad]～ 2π[rad]   負は無し   多回転無し
    double heading;
  };
};
```

A) ウェイポイント列(x,y,θ)で表現。

B) ウェイポイントの間隔、個数は任意。

C) 新ウェイポイント列を受信することで、既に受け取った旧ウェイポイント列は破棄し、新ウェイポイント列を採用する。ただし、この処理の間に停止するかどうかは実装依存。

D) C)の特殊ケースとしてウェイポイント列のデータ長（sequence<RTC::Pose2D> data.length()）が0※1は、その場で停止を示しており、旧ウェイポイント列は破棄。

E) ウェイポイント列の1番目データはスタート点とする。つまり少なくとも2ウェイポイント必要。

F) ウェイポイント列スタート点と自己位置にずれがある場合の修正移動については実装依存。対処できない場合は、d)走行状態で停止信号を返すと言った処理が可能。

G) E),F)については4/9において深く議論されておらず、暫定。

※1：データ長0が実装可能なのはRTM0.4.2にて確認済み

b)c) 走行指令・走行情報IF

```
#include "BasicDataType.idl"
#include "ExtendedDataTypes.idl"      //RTM1.0において定義
module IIS {//IIS2.idl
  struct TimedVelocity2D { //走行指令・走行情報IF
    RTC::Time tm;
    sequence<long> id;
    RTC::Velocity2D data;   //RTM1.0において定義
     sequence<double> error;
  };
};
```

```
module RTC {//ExtendedDataTypes.idl
  struct Velocity2D {
    /// Velocity along the x axis in metres per second.
    double vx;
    /// Velocity along the y axis in metres per second.
    double vy;
    /// Yaw velocity in radians per second.
    double va;
  };
};
```

A) ロボット中心座標系における速度(vx,vy,va)で表現。

B) 走行情報IFでは、10[ms]程度以下の間隔で出力することを奨励。

C) 走行情報IFでは、時間が積極的に利用されるのでTime tmの情報附加を奨励。

d)確率的自己位置姿勢IF

```
#include "BasicDataType.idl"
#include "ExtendedDataTypes.idl"      //RTM1.0において定義
module IIS {//IIS2.idl
 struct TimedPose2D { //自己位置姿勢IF
   RTC::Time tm;
   sequence<long> id;
   RTC::Pose2D data;        //RTM1.0において定義
   sequence<double> error;
 };
};
```

```
module RTC {//ExtendedDataTypes.idl
 struct Point2D   {
   /// X coordinate in metres.
   double x;
   /// Y coordinate in metres.
   double y;
 };
 struct Pose2D   {
   /// 2D position.
   Point2D position;
   /// Heading in radians.
   // x軸から反時計回りに0[rad]～ 2π[rad]   負は無し   多回転無し
   double heading;
 };
};
```

A) 位置と姿勢(x,y,θ)で表現。

e)目的地IF

```
#include "BasicDataType.idl"
#include "ExtendedDataTypes.idl"      //RTM1.0において定義
module IIS {//IIS2.idl
  struct TimedPose2D {
    RTC::Time tm;
    sequence<long> id;
    RTC::Pose2D data;        //RTM1.0において定義
    sequence<double> error;
  };
 struct TimedPose2DSeq { //目的地IF
    RTC::Time tm;
    sequence<long> id;
    sequence<RTC::Pose2D> data;
    sequence<double> error;
  };
};
```

```
module RTC {//ExtendedDataTypes.idl
 struct Point2D    {
    /// X coordinate in metres.
    double x;
    /// Y coordinate in metres.
    double y;
  };
  struct Pose2D   {
    /// 2D position.
    Point2D position;
    /// Heading in radians.
    // x軸から反時計回りに0[rad]～ 2π[rad]   負は無し   多回転無し
    double heading;
  };
};
```

A) 位置姿勢(x,y,θ)で表現。

B) 目的地データ型はsequence型だが1番目のデータのみ有効。その他のデータは無視。

C) 新目的地位置姿勢を受信することで、既に受け取った旧目的地位置姿勢は破棄し、新目的地位置姿勢を採用する。

D) C)の特殊ケースとしてデータ長（sequence<RTC::Pose2D> data.length()）が0だった場合は、旧目的地位置姿勢データは破棄。

E) D)ケースにて、経路計画部の動作については実装依存。a)計画経路IFのD)ケースを出力することを奨励。

F) E)については4/9において深く議論されておらず、暫定。

f)走行状態IF（暫定）

```
#include "BasicDataType.idl"
#include "ExtendedDataTypes.idl"      //RTM1.0において定義
module IIS {//IIS2.idl
 struct TimedPose2D { //自己位置姿勢IF
   RTC::Time tm;
   sequence<long> id;
   RTC::Pose2D data;        //RTM1.0において定義
   sequence<double> error;
 };
};
```

```
module RTC {//ExtendedDataTypes.idl
 struct Point2D   {
   /// X coordinate in metres.
   double x;
   /// Y coordinate in metres.
   double y;
 };
 struct Pose2D   {
   /// 2D position.
   Point2D position;
   /// Heading in radians.
   // x軸から反時計回りに0[rad]～ 2π[rad]   負は無し   多回転無し
   double heading;
 };
};
```

A) ウェイポイント番号と走行状態（走行or停止）

B) 与えられたウェイポイント列で通過したウェイポイント番号を通過した度に送信。

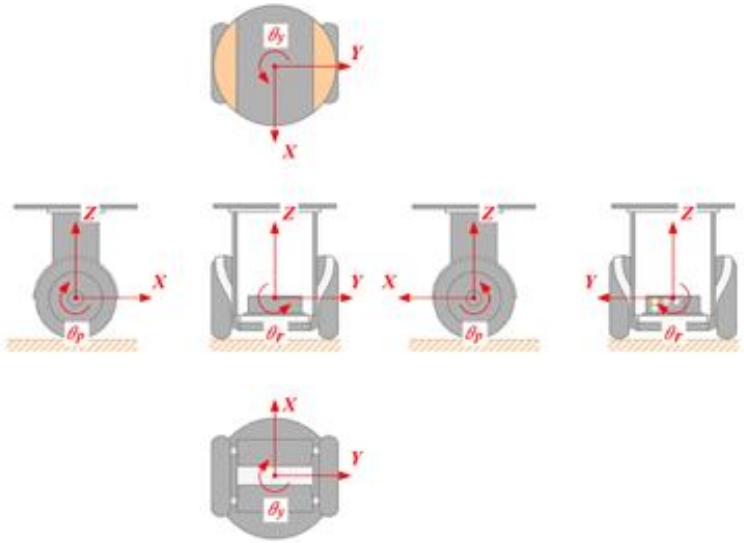
C) 停止し場合は、フラグを立てて送信。その場合も最後に通過したウェイポイントを同時に送信。

論点
下記のような新型を定義しても良いが、通過ウェイポイントの位置姿勢情報が付加されている方が便利ということであれば、TimedPose2D型と情報は同じなので型をむやみに増やさないという面では、TimedPose2D型を採用するという方法もある。

```
struct TimedState {
  RTC::Time tm;
  long id;
  boolean isStop;
};
```

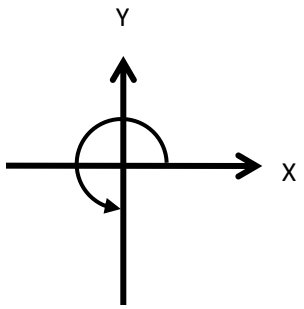
座標系

ロボット中心座標系（右手系）



原点：ロボット中心
X軸方向：ロボット進行方向
単位：m, m/s, radian, radian/s

地図座標系（右手系）



原点：任意
X軸方向：任意
単位：
m, m/s, radian, radian/s

可能であれば
平面直角座標系を奨励