次世代ロボット知能化技術開発プロジェクト
ロボット知能ソフトウェア再利用性向上技術の開発

操作手順書
ロボットアーム分解運動速度制御モジュール
（ACT 共通 I/F 対応版）

Ver.1.0

2011 年 6 月 30 日

RTC 再利用技術研究センター

目次

１．はじめに.....1
１．１．本書の適用範囲.....1
１．２．関連文書.....1
１．３．本書を読むにあたって.....1
２．準備.....2
２．１．動作環境.....2
２．２．開発環境構築.....2
２．２．１．モジュールの導入.....2
２．２．２．OpenRTM-aist-1.0.0 RELEASE (C++) Ubuntu 10.04 のインストール.....3
２．２．３．OpenRTM-aist-Python-1.0.0-RELEASE のインストール.....4
２．２．４．Eclipse 関連ツール のインストール.....4
２．２．５．python visual のインストール.....5
２．２．６．RTC のコンパイル.....5
２．２．７．設定ファイル修正.....5
３．使用方法.....6
３．１．シミュレータ環境での操作.....6
３．１．１．PA10.....6
３．１．２．RH.....8
３．２．実機環境での操作.....12
３．２．１．PA10.....12
３．２．２．RH.....13
４．特記事項.....15

1. はじめに

1.1. 本書の適用範囲

本書はロボット向けミドルウェア OpenRTM 上で多軸アームロボットの分解運動速度制御を行う知能モジュールの仕様について記述した文書である。

1.2. 関連文書

本書の関連文書は　下表の通り。

表 1-1 関連文書

No.	文書名	備考
1	ACT（低レベル）共通インターフェース仕様書	-
2	ACT（中レベル）共通インターフェース仕様書	-
3	機能仕様書 ロボットハンド(RH707)制御モジュール	-
4	操作手順書 ロボットハンド(RH707)制御モジュール	-
5	機能仕様書ロボットハンド(PA10)制御モジュール	-
6	操作手順書 ロボットハンド(PA10)制御モジュール	-
7	RefHard2 アーム制御 RTC	-

1.3. 本書を読むにあたって

本書は RT ミドルウェア、RT コンポーネント(以下、RTC)に関する基本知識を備えた利用者を対象としている。RT ミドルウェア、RTC については下記を参照のこと。

OpenRTM-aist Official Website:
http://www.openrtm.org/

2. 準備

2．1． 動作環境

動作 OS	Ubuntu10.04
開発言語	C++
コンパイラ	g++4.4.3-1
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE（C++版）
依存パッケージ(OpenRTM)	OmniORB-4.1.2-1
	libace-5.6.3-6
依存パッケージ(その他)	行列演算ライブラリ

2．2． 開発環境構築

2．2．1． モジュールの導入

表 2-1 ARM_Middlelv.zip

モジュール	説明
ACT_PA10_RH_Middle.zip	中レベル I/F 対応 RTC
patch.zip	OpenRTM 用パッチ

ダウンロードファイル（ARM_Middlelv.zip）を解凍する。（ここでは~/にダウンロードしたものとする）

```
unzip ARM_Middlelv.zip
cp ./ARM_Middlelv/ACT_PA10_RH_Middlel.zip ./
unzip ACT_PA10_RH_Middle.zip
```

２．２．２．　OpenRTM-aist-1.0.0 RELEASE (C++) Ubuntu 10.04　のインストール

C++言語で実装された RTC を実行するための RT ミドルウェアをインストールする。
本書では、まず一括インストールスクリプトを使用し OpenRTM-aist-1.0.0-RELEASE とその依存パッケージの一括インストールを行う。
その後バグ対応を行い（未対応であることを前提としているため）、OpenRTM-aist-1.0.0-RELEASE を再インストールする。

1. 一括インストールスクリプト pkg_install100_ubuntu.sh を下記より適当なディレクトリ（ここでは~/）

   にダウンロードし、実行する。
   ※一括インストールスクリプトで「コンパイラや OmniORB4」をインストールする。

   pkg_install100_ubuntu.sh ダウンロード

```
$su
#sh pkg_install100_ubuntu.sh
#exit
```

対応する問題

- InPort::read()でブロックされる問題　[openrtm-users 01308]参照
- Manager の shutdown に関連したバグ　[openrtm-users 01149]参照
- ipv6 エントリのコメントアウト　[openrtm-users 01270]参照

2. 下記よりソースファイルを（ここでは~/に）ダウンロードし展開する。
   OpenRTM-aist-1.0.0-RELEASE.tar.gz ダウンロード

```
tar -xvzf OpenRTM-aist-1.0.0-RELEASE.tar.gz
```

3. 環境構築 1.で導入した「ARM_Middlelv/patch」フォルダ内にあるパッチファイルを
   ~/OpenRTM-aist-1.0.0/src/lib/rtm 内にコピーする。

```
unzip ~/ARM_Middlelv/patch.zip
cp ~/ARM_Middlelv/patch/RingBuffer.h.diff ~/OpenRTM-aist-1.0.0/src/lib/rtm
cp ~/ARM_Middlelv/patch/Manager.cpp.diff ~/OpenRTM-aist-1.0.0/src/lib/rtm
```

4. パッチを当てる。

```
cd OpenRTM-aist-1.0.0/src/lib/rtm
patch < RingBuffer.h.diff
patch < Manager.cpp.diff
```

5. make する。

```
~/OpenRTM-aist-1.0.0$./configure --prefix=/usr
~/OpenRTM-aist-1.0.0$make clean
~/OpenRTM-aist-1.0.0$make
~/OpenRTM-aist-1.0.0$sudo make install
```

6. /etc/hosts の localhost の ipv6 エントリをコメントアウトする。

```
# ::1         localhost ip6-localhost ip6-loopback
```

2．2．3． OpenRTM-aist-Python-1.0.0-RELEASE のインストール

Python 言語で実装された RTC を実行するための RT ミドルウェアをインストールする。

1. パッケージインストール
   一括インストールスクリプト pkg_install_python_ubuntu.sh を下記から適当なディレクトリ（ここでは~/）にダウンロードし、実行する。

   pkg_install_python_ubuntu.sh ダウンロード

```
su
# sh pkg_install_python_ubuntu.sh
```

   途中、いくつかの質問をたずねられるので、y あるいは Y を入力しながら完了させる。

2．2．4． Eclipse 関連ツール のインストール

RTC を接続したり、状態を監視するためのツール RTSystemEditor をインストールする。
RTSystemEditor は Eclipse のプラグインとして提供されるものであるため Eclipse もインストールする必要がある。
以下の URL から RTSystemEditor をプラグインした Eclipse（Linux 用全部入り）をダウンロードしインストールする。

全部入りパッケージ
URL: http://www.openrtm.org/openrtm/ja/node/941#package

また上記参照先に記載されている Eclipse 動作不具合への対応も必ず行う。

２．２．５． python visual のインストール

シミュレータ環境として、python visual をインストールする。

1. Synaptic マネージャを起動させる。
   「システム」→「システム管理」→「Synaptic パッケージ・マネージャ 」と選択する。
2. python visual をインストールする。
   「python visual」で検索し、チェックを入れ、適用する。
3. 「libgtkglext」で検索し、libgtkglext1 と libgtkglext1-dev にチェックを入れ、適用する。

２．２．６． RTC のコンパイル

```
cd ~/ARM_Middlelv/ACT_PA10_RH_Middle/src
ARM_Middlelv/ACT_PA10_RH_Middle/src$sh makeall.sh
```

２．２．７． 設定ファイル修正

~/ARM_Middlelv/ACT_PA10_RH_Middle/etc/set_env.py にあるネームサーバーを適宜修正する。

```
#!/usr/bin/env python
# -*- Python -*-import subprocess
#
import os,pwd
UserName = pwd.getpwuid(os.getuid())[0]
LibDir="/home/"+UserName+"/ARM_Middlelv/ACT_PA10_RH_Middle/lib"
SrcDir="/home/"+UserName+"/ARM_Middlelv/ACT_PA10_RH_Middle/src/MotionControlle
r"
EtcDir="/home/"+UserName+"/ARM_Middlelv/ACT_PA10_RH_Middle/etc"
IcldDir="/home/"+UserName+"/ARM_Middlelv/ACT_PA10_RH_Middle/include"
LogDir="/home/"+UserName+"/ARM_Middlelv/ACT_PA10_RH_Middle/log"
BinDir="/home/"+UserName+"/ARM_Middlelv/ACT_PA10_RH_Middle/bin"


○○○○に使用する PC の IP アドレスを記述する
```

3. 使用方法

本章では、シミュレータ環境および実機環境における PA10、RH アームの操作方法について記述する。本モジュールに同封されている RTC はシミュレータ環境用のもののみであり、実機環境における操作を行う場合には、PA10 または RH アームの実機制御モジュールを別途用意する必要がある。

3．1． シミュレータ環境での操作

3．1．1． PA10

1. RTC の起動

ネームサーバ（rtm-naming 9876）を立ち上げる。

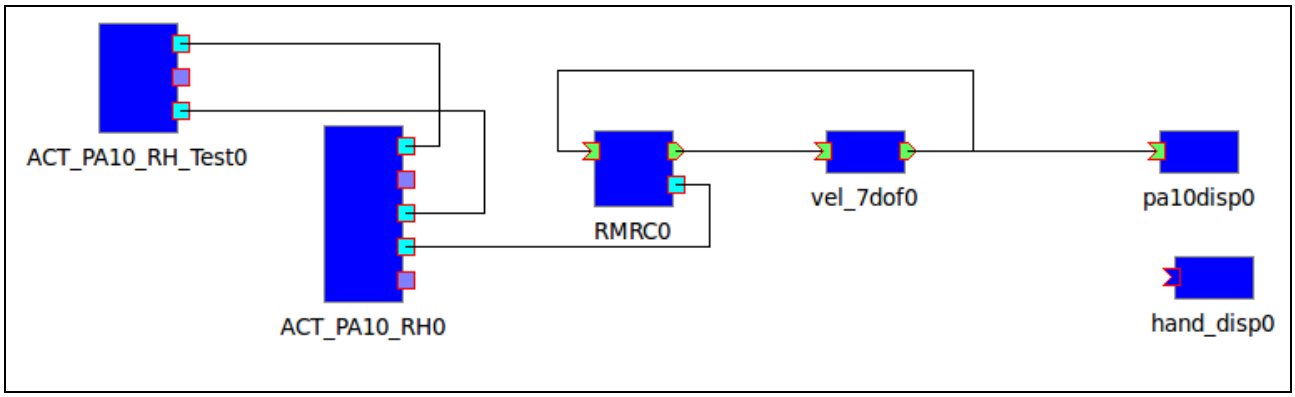
```
rtm-naming 9876
```

分解運動速度制御 RTC を起動させる。

```
cd ~/ARM_Middlelv/ACT_PA10_RH_Middle/bin/script
ARM_Middlelv/ACT_PA10_RH_Middle/bin/script $sh simPA10SysRun.sh
```

2. RTC の接続

RTSystemEditor を用いて各 RTC を接続する。下図を参考にコンポーネントのポートを接続する。



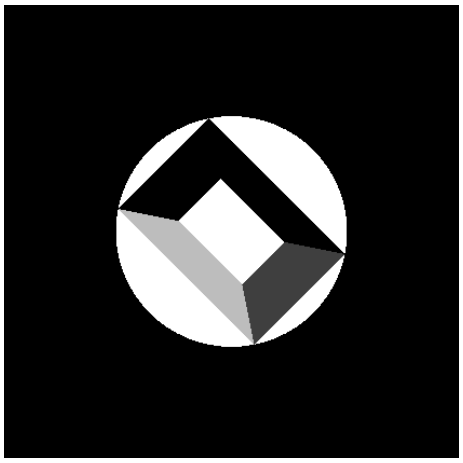
3. ACT_PA10_RH_Test コンポーネントによるモジュール操作

1. 接続が完了したら各 RTC の活性化を行う（活性化の順序は問わない）。
2. 全ての RTC を活性化し終えたら、ACT_PA10_RH_TestComp コンソール、VPython コンソールが立ち上がっていることを確認する。

図 3-1 ACT_PA10_RH_TestComp コンソール

```
********** ACT_PA10_RH_Test **********
 A:closeGripper
 B:openGripper
 C:getBaseOffset
 D:getFeedbackPosCartesian
 E:getSoftLimitCartesian
 F:moveLinearCartesianAbs
 G:movePTPJointAbs (ready)
 H:movePTPJointAbs (clear)
 I:setControlPointOffset
 J:setMaxSpeedJoint
 K:setSoftLimitCartesian
 LcreateCarPosWithElbow
****************************************
 =>
```

図 3-2 VPython コンソール



3. このコンソールから中レベル I/F メソッドを呼び出す。

例）getBaseOffset メソッドを実行する場合

コンソールへ C と入力し Enter キーを押す。

```
********** ACT_PA10_RH_Test **********
 A:closeGripper
 B:openGripper
 C:getBaseOffset
 D:getFeedbackPosCartesian
 E:getSoftLimitCartesian
 F:moveLinearCartesianAbs
 G:movePTPJointAbs (ready)
 H:movePTPJointAbs (clear)
 I:setControlPointOffset
 J:setMaxSpeedJoint
 K:setSoftLimitCartesian
 LcreateCarPosWithElbow
****************************************
 =>C
********** ACT_PA10_RH_Test **********
 A:closeGripper
 B:openGripper
 C:getBaseOffset
 D:getFeedbackPosCartesian
 E:getSoftLimitCartesian
 F:moveLinearCartesianAbs
 G:movePTPJointAbs (ready)
 H:movePTPJointAbs (clear)
 I:setControlPointOffset
 J:setMaxSpeedJoint
 K:setSoftLimitCartesian
 LcreateCarPosWithElbow
****************************************
 =>
```

getBaseOffset メソッドが実行され、
ARM_Middlelv/ACT_PA10_RH_Middle/log/ACT_PA10_RH.log 内に、

```
Jun 20 15:46:19 INFO: ManipulatorCommonInterface_MiddleSVC_impl: getBaseOffset() was
completed.
      offset :[0.000000, -0.000000, -0.000000, 0.000000
               -0.000000, 0.000000, 0.000000, 0.000000
               -0.000000, 0.000000, 0.000000, 0.000000]
```

と出力される。

※PA10 のシミュレータ環境においてはアーム操作のみが行える。hand_disp はハンドモデルの描画を行っているのみであり closeGripper、openGripper によるハンド開閉は行わない。

4. モジュール操作終了

RTSystemEditor を用いて全ての RTC を非活性化させた後、以下のシェルスクリプトを実行する。

```
ARM_Middlelv/ACT_PA10_RH_Middle/bin/script $sh killPA10rtc.sh
```

３．１．２．　RH

1. RTC の起動

ネームサーバ（rtm-naming 9876）を立ち上げる。

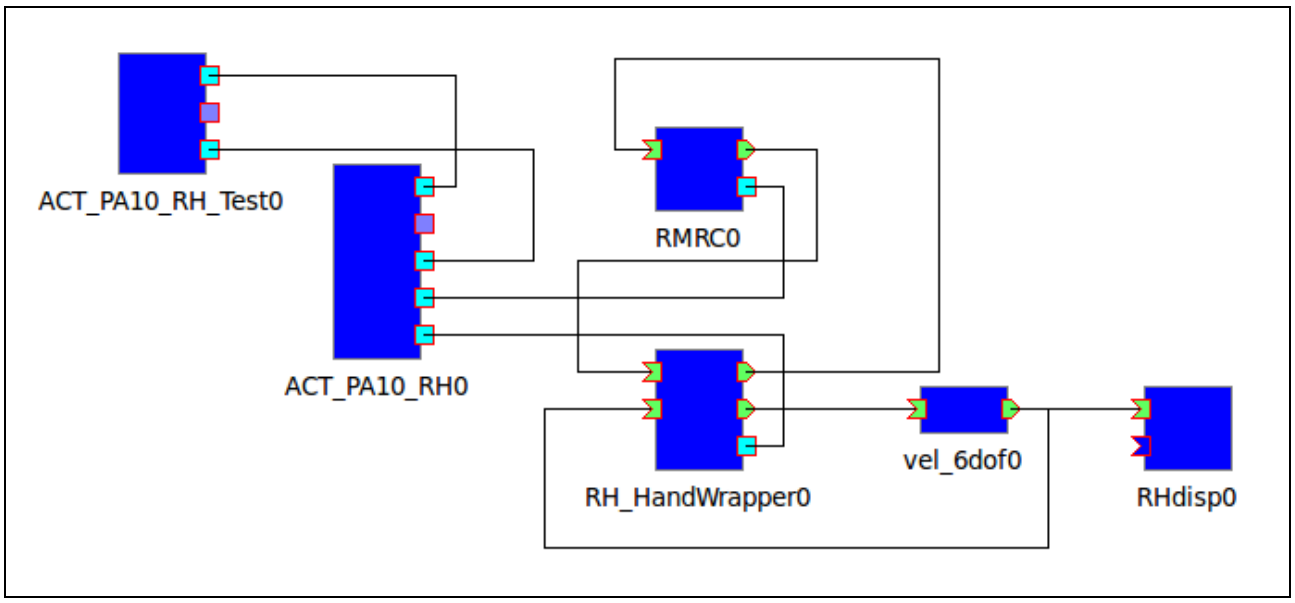
```
rtm-naming 9876
```

分解運動速度制御 RTC を起動させる。

```
cd ~/ARM_Middlelv/ACT_PA10_RH_Middle/bin/script
ARM_Middlelv/ACT_PA10_RH_Middle/bin/script $sh simRHSysRun.sh
```

2. RTC の接続

RTSystemEditor を用いて各 RTC を接続する。下図を参考にコンポーネントのポートを接続する。



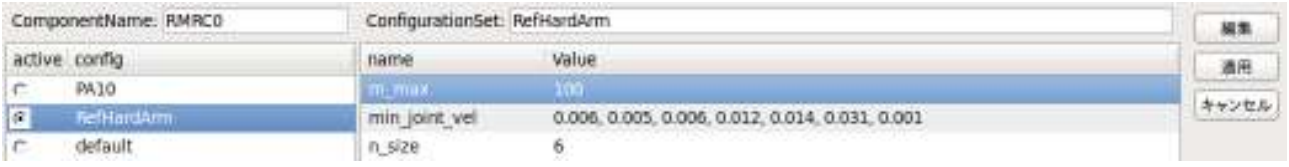
3. コンフィギュレーションの設定

- ACT_PA10_RH_Test

コンフィギュレーションセット“RH”を選択し、適用させる。

- RMRC

コンフィギュレーションセット“RefHardArm”を選択し、適用させる。



4. ACT_PA10_RH_Test コンポーネントによるモジュール操作

PA10 の場合と同様に、接続が完了後、各 RTC の活性化を行い（活性化の順序は問わない）、コンソール入力によりモジュールを操作する。

例）openGripper メソッドを実行する場合

コンソールへ B と入力し Enter キーを押す。

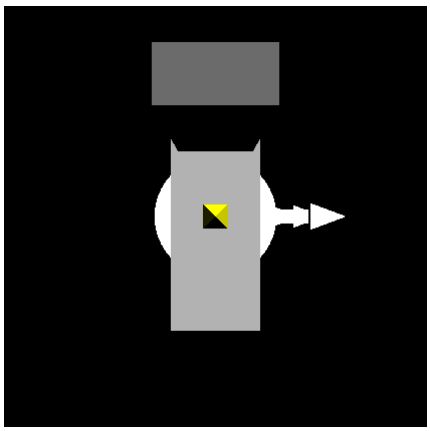
```
********** ACT_PA10_RH_Test **********
 A:closeGripper
 B:openGripper
 C:getBaseOffset
 D:getFeedbackPosCartesian
 E:getSoftLimitCartesian
 F:moveLinearCartesianAbs
 G:movePTPJointAbs
 H:setControlPointOffset
 I:setMaxSpeedJoint
 J:setSoftLimitCartesian
 K:createCarPosWithElbow
****************************************
 =>B
********** ACT_PA10_RH_Test **********
 A:closeGripper
 B:openGripper
 C:getBaseOffset
 D:getFeedbackPosCartesian
 E:getSoftLimitCartesian
 F:moveLinearCartesianAbs
 G:movePTPJointAbs
 H:setControlPointOffset
 I:setMaxSpeedJoint
 J:setSoftLimitCartesian
 K:createCarPosWithElbow
****************************************
 =>
```

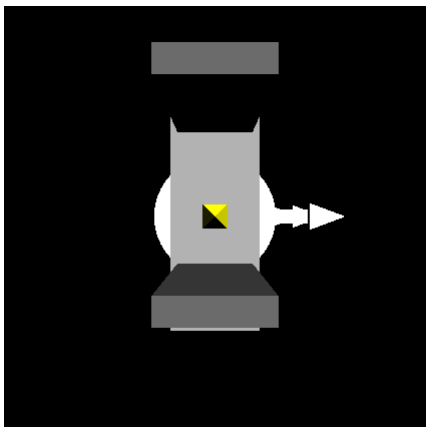
openGripper メソッドが実行され、RH アームモデルのハンドが開口し（**図 3-3 ハンド開口前**　　**図 3-4 ハンド開口後**）、
ARM_Middlelv/ACT_PA10_RH_Middle/log/ACT_PA10_RH.log 内に、

```
Jun 20 16:41:58 INFO: ManipulatorCommonInterface_MiddleSVC_impl: openGripper() was
```

と出力される。

図 3-3 ハンド開口前　　**図 3-4 ハンド開口後**





5. モジュール操作終了

RTSystemEditor を用いて全ての RTC を非活性化させた後、以下のシェルスクリプトを実行する。

```
ARM_Middlelv/ACT_PA10_RH_Middle/bin/script $sh killRHrtc.sh
```

３．２．　実機環境での操作

３．２．１．　PA10

1. RTC の起動

ネームサーバ（rtm-naming 9876）を立ち上げる。

```
rtm-naming 9876
```

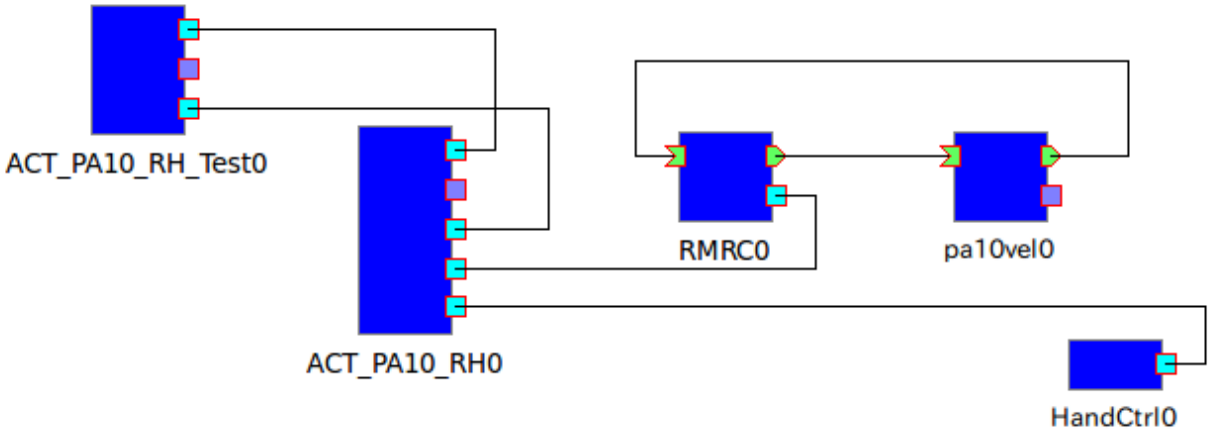
分解運動速度制御 RTC を起動させる。

```
cd ~/ARM_Middlelv/ACT_PA10_RH_Middle/bin/script
ARM_Middlelv/ACT_PA10_RH_Middle/bin/script $sh PA10SysRun.sh
```

ロボットハンド（RH707）制御モジュール、PA10 制御モジュールを起動させる（関連文書4、6を参照）。

2. RTC の接続

RTSystemEditor を用いて各 RTC を接続する。下図を参考にコンポーネントのポートを接続する。



3. ACT_PA10_RH_Test コンポーネントによるモジュール操作

操作方法はシミュレータ環境と同様となる。

3．2．2． RH

1. RTC の起動

ネームサーバ（rtm-naming 9876）を立ち上げる。

```
rtm-naming 9876
```

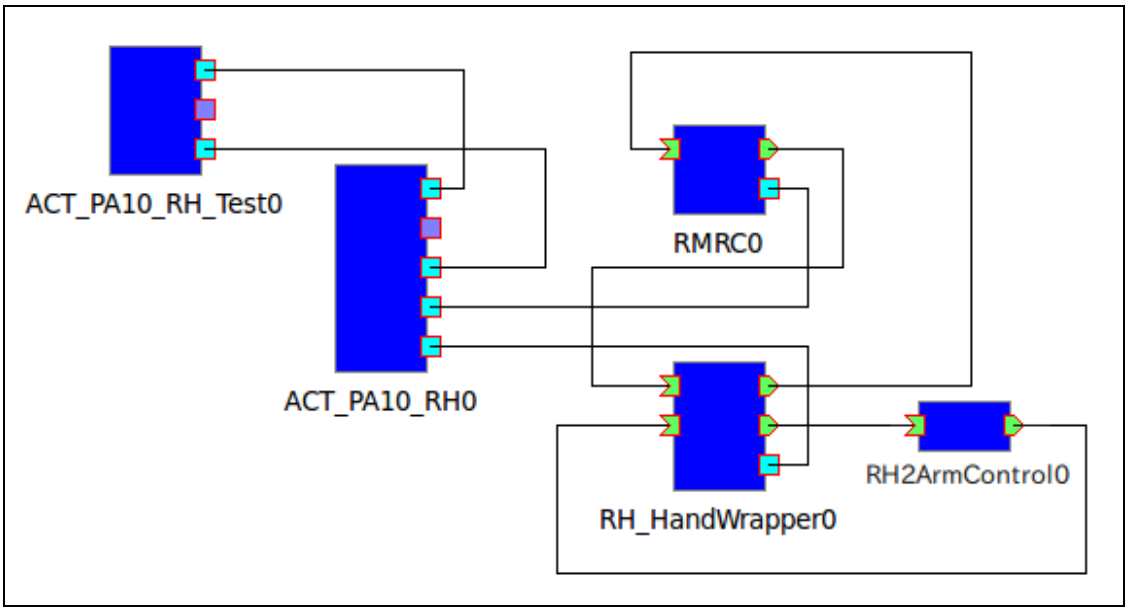
分解運動速度制御 RTC を起動させる。

```
cd ~/ARM_Middlelv/ACT_PA10_RH_Middle/bin/script
ARM_Middlelv/ACT_PA10_RH_Middle/bin/script $sh RHSysRun.sh
```

RefHard2 アーム制御 RTC を起動させる（関連文書 7 を参照）。

2. RTC の接続

RTSystemEditor を用いて各 RTC を接続する。下図を参考にコンポーネントのポートを接続する。



3. コンフィギュレーションの設定

設定方法はシミュレータ環境と同様になる。

4. ACT_PA10_RH_Test コンポーネントによるモジュール操作

操作方法はシミュレータ環境と同様になる。

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。

- ドキュメントに情報を掲載する際には万全を期していますが、それらの情報の正確性またはお客様にとっての有用性等については一切保証いたしません。
- 利用者が本モジュールを利用することにより生じたいかなる損害についても一切責任を負いません。
- 本モジュールの変更、削除等は、原則として利用者への予告なしに行います。また、止むを得ない事由により公開を中断あるいは中止させていただくことがあります。
- 本モジュールの情報の変更、削除、公開の中断、中止により、利用者に生じたいかなる損害についても一切責任を負いません。
- 本モジュールは独立行政法人 新エネルギー・産業技術総合開発機構（NEDO 技術開発機構）の「次世代ロボット知能化技術開発プロジェクト」（平成 19 年～平成 23 年度）において、評価を目的として開発されたものであり、商用以外の利用の場合、BSD ライセンスが適用されます。詳しくは同封の LICENSE-BSD.TXT を参照ください。
- 商用利用の際には連絡を要し、使用条件は個別に検討するものとします。

【連絡先】
RTC 再利用技術研究センター
〒101-0021 東京都千代田区外神田 1-18-13 秋葉原ダイビル 1303 号室
Tel/Fax：03-3256-6353 E-Mail：contact@rtc-center.jp