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(PA10)分解運動速度制御モジュール
(Linux)

Ｖｅｒ．３．０

２０１０年4月２１日

ＲＴＣ再利用技術研究センター

改版履歴

Ver.	改版日	改版内容
1.0	2010/03/23	新規作成
2.0	2010/04/19	Debian 用の記載を追加（3.1.1、3.1.2）
3.0	2010/04/21	環境構築における不具合修正（3.1.1 configure 時に prefix を指定）

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
　１．４．　動作環境 2
２．　ソースディレクトリ構成 3
３．　ソフトウェアインストールと環境構築 4
　３．１．　基本環境 4
　３．２．　シミュレータ環境(VPython) 7
　３．３．　実機環境 8
４．　事前準備 9
　４．１．　ツール位置 9
　４．２．　アーム先端の速度 9
　４．３．　PATH（Python ソースコード）の修正 10
　４．４．　ソースファイルの make 10
５．　実行 11
　５．１．　ネームサーバの起動 11
　５．２．　RTSystemEditor の起動 11
　５．３．　シミュレータ環境 12
　５．４．　実機環境 16
６．　トラブルシューティング 19
７．　特記事項 21

１．　はじめに

１．１．　本書の適用範囲

本書は三菱重工業製汎用ロボット PA10 を、産業技術総合研究所が開発した RT ミドルウェア OpenRTM-aist を用いてシミュレータ環境及び実機環境で動作させるための手順を記述したものである。

１．２．　関連文書

本書の関連文書は 下表の通り。

No.	文書名	備考
1	ロボットアーム(PA10)分解運動速度制御モジュール(Linux)　機能仕様書	ロボットアーム(PA10)分解運動速度制御モジュールを構成する各 RT コンポーネントの仕様について記載。

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

URL：http://www.is.aist.go.jp/rt/OpenRTM-aist/

１．４．　動作環境

以下の環境では、実機およびシミュレータともに動作確認が取れている。

OS	Debian4.0r4
RT ミドルウェア	OpenRTM-aist-0.4.2 RELEASE (C++)
	OpenRTM-aist-Python-0.4.1-RELEASE
開発言語	C++、Python
コンパイラ	g++4.1.1-21
インタープリタ	Python2.4.4-2
依存ライブラリ（OpenRTM）	OmniORB-4.0.6-2.1
	OmniORBpy-2.6-3.3
	libace-5.4.7-12
依存ライブラリ（その他）	VPython3.2.1-4+b
	ライフロボティクス社製 PA-10 高速制御用ソフトウェア（商用）

以下の環境では、シミュレータのみ動作確認が取れている。

OS	Ubuntu8.04
RT ミドルウェア	OpenRTM-aist-0.4.2 RELEASE (C++)
	OpenRTM-aist-Python-0.4.1-RELEASE
開発言語	C++、Python
コンパイラ	g++4.2.3-1
インタープリタ	Python2.5.2-0
依存ライブラリ（OpenRTM）	OmniORB-4.1.1.2
	OmniORBpy-3.2.1
	libace-5.4.7-13
依存ライブラリ（その他）	VPython3.2.9-3
	ライフロボティクス社製 PA-10 高速制御用ソフトウェア（商用）

２．　ソースディレクトリ構成

ディレクトリ構成はホームディレクトリ直下に下表の通り作成するとし、構成が異なる場合は、適宜その環境に合わせた修正（4.3）を行うこと。

PA10RMRC_Linux_Debian.tar.gz をホームディレクトリ下に置く

```
$tar xvzf PA10RMRC_Linux_Debian.tar.gz
```

ディレクトリ	内容	言語
openrtm	-	-
├ myRTC	-	-
│ ├ coord_trans	座標変換コンポーネント	C++
│ ├ frm_ctrl	軌跡制御コンポーネント	C++
│ ├ j_jacob	ヤコビシーケンス生成コンポーネント	C++
│ ├ jinv	逆ヤコビ行列変換コンポーネント	C++
│ ├ mixer_py	ヤコビシーケンス生成コンポーネント	Python
│ ├ mixer_simple	ヤコビシーケンス生成コンポーネント	C++
│ ├ move_py	操作制御コンポーネント	Python
│ ├ pa10disp_py	PA10 幾何モデル描画コンポーネント	Python
│ ├ pa10fk	PA10 順運動学計算コンポーネント	C++
│ ├ pa10vel	PA10 実機制御コンポーネント	C++
│ ├ script	RT システム起動スクリプト	Python
│ ├ slider	データ送出コンポーネント	Python
│ ├ tools	-	-
│ │ ├ geo	幾何演算ライブラリ	C++/Python
│ │ ├ nr	行列演算ライブラリ	C
│ │ └ v_robot	PA10 シミュレータ描画処理	Python
│ ├ tr_jacob	ヤコビ行列変換コンポーネント	C++
│ └vel_sim	7 軸アームシミュレータコンポーネント	C++
├ tools	-	-
│ └ rtc_handle	RtcHandle	Python
└ patch	RtcHandle に関するバグ修正 patch ファイル	-

３．　ソフトウェアインストールと環境構築

動作に必要なソフトウェアを以下に記す。全てのソフトウェアのインストールが完了したら再起動すること。

３．１．　基本環境

３．１．１．　OpenRTM-aist-0.4.2 RELEASE (C++)

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-0.4.2-RELEASE とその依存パッケージの一括インストールを行い、その後パッチを当てた OpenRTM-aist-0.4.2-RELEASE を再インストールする。このパッチ当ては、当センターが推奨する rtc-handle によるモジュールの Python スクリプト制御が正常に行われるために必要な処理である。

参考：OpenRTM-BUG (staff.aist.go.jp)

URL：http://staff.aist.go.jp/t.suehiro/rtm/OpenRTM-Bug.html

参考：RtcHandle – 使い方とそれを用いた RTC 利用環境の構築

URL：http://staff.aist.go.jp/t.suehiro/rtm/rtc_handle.html

インストール手順

1. 一括インストールスクリプト pkg_install042_debian.sh または pkg_install042_ubuntu.sh を下記 URL から適当なディレクトリにダウンロードし、実行する（ここではホームディレクトリにダウンロードし実行するものとする）。

一括インストールスクリプト

URL：

http://www.openrtm.org/OpenRTM-aist/html/E38380E382A6E383B3E383ADE383BCE383892FC2B2B2F0.4.2-RELEASE.html#jcfc8e89

```
$su
#sh pkg_install042_debian.sh
または、
#sh pkg_install042_ubuntu.sh
#exit
```

2. OpenRTM-aist-0.4.2-RELEASE のソースファイルを適当なディレクトリにダウンロードする（ここではホームディレクトリにダウンロードするものとする）

ソースファイル

URL：

http://www.openrtm.org/OpenRTM-aist/html/E38380E382A6E383B3E383ADE383BCE383892FC2B2B2F0.4.2-RELEASE.html#p8d9179d

3. ソースファイルを展開し、既存の OpenRTM-aist-0.4.2-RELEASE をアンインストールする

```
$tar xvzf OpenRTM-aist-0.4.2-RELEASE.tar.gz
$cd OpenRTM-aist-0.4.2
/OpenRTM-aist-0.4.2$./configure --prefix=/usr
```

4. ソースにパッチを当てる

パッチファイル	PublisherFactory.patch

```
/OpenRTM-aist-0.4.2$cd rtm
/OpenRTM-aist-0.4.2/rtm$
 cp ../../openrtm/patch/PublisherFactory.patch ./
/OpenRTM-aist-0.4.2/rtm$patch < PublisherFactory.patch
/OpenRTM-aist-0.4.2/rtm$cd ../
```

5. ソースの make を行いインストールする

```
/OpenRTM-aist-0.4.2$make clean
/OpenRTM-aist-0.4.2$make
/OpenRTM-aist-0.4.2$sudo make install
```

３．１．２．　OpenRTM-aist-Python-0.4.1-RELEASE

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。(3.1.1)と同様、こちらも Python スクリプトによるモジュール制御が正常に行われるようパッチを当てる。

インストール手順

1. 依存パッケージのインストール

   Debian、Ubuntu 環境共に下記サイトの Ubuntu7.10 の場合を参考にして行う

   URL：

   http://www.openrtm.org/OpenRTM-aist/html/E3839EE3838BE383A5E382A2E383AB2FE382A4E383B3E382B9E38388E383BCE383ABPythonUNIX.html#k01d460e

2. OpenRTM-aist-0.4.1-Python-RELEASE のソースファイルと、パッチファイル patch-OpenRTM-aist-Python-0.4.1-20081030 を適当なディレクトリにダウンロードする（ここではホームディレクトリにコピーするものとする）

3.

   ソースファイル・パッチファイル

   URL：

   http://www.openrtm.org/OpenRTM-aist/html/E38380E382A6E383B3E383ADE383BCE383892FPython2F0.4.1-RELEASE.html#k3a68189

4. ソースファイルを展開し、パッチを当てる

```
$tar xvzf OpenRTM-aist-Python-0.4.1-RELEASE.tar.gz
$cd OpenRTM-aist-Python-0.4.1/OpenRTM
/OpenRTM-aist-Python-0.4.1/OpenRTM$
 cp ../../patch-OpenRTM-aist-Python-0.4.1-20081030 ./
/ OpenRTM-aist-Python-0.4.1/OpenRTM$
 patch < patch-OpenRTM-aist-Python-0.4.1-20081030
```

5. ソースのインストール

```
/OpenRTM-aist-Python-0.4.1/OpenRTM$cd ../
/OpenRTM-aist-Python-0.4.1$python setup.py build
/OpenRTM-aist-Python-0.4.1$sudo python setup.py install
```

6. IDL コンパイルをしなおす

Debian 環境

```
/OpenRTM-aist-Python-0.4.1$cd
$cd /usr/lib/python2.4/site-packages/OpenRTM/RTM_IDL
usr/lib/python2.4/site-packages/OpenRTM/RTM_IDL$
 sudo rm -rf RTC RTC__POA SDOPackage SDOPackage__POA
usr/lib/python2.4/site-packages/OpenRTM/RTM_IDL$
 sudo omniidl -bpython *.idl
```

Ubuntu 環境

```
/OpenRTM-aist-Python-0.4.1$cd
$cd /usr/lib/python2.5/site-packages/OpenRTM/RTM_IDL
usr/lib/python2.5/site-packages/OpenRTM/RTM_IDL$
 sudo rm -rf RTC RTC__POA SDOPackage SDOPackage__POA
usr/lib/python2.5/site-packages/OpenRTM/RTM_IDL$
 sudo omniidl -bpython *.idl
```

３．１．３．Eclipse 関連ツール

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 をダウンロードしインストールする。

ダウンロード先

URL：

http://www.is.aist.go.jp/rt/OpenRTM-aist/html/E3839EE3838BE383A5E382A2E383AB2FRTSystemEditorE383BBRTCBuilderE381AEE382A4E383B3E382B9E38388E383BCE383AB.html

３．２．　シミュレータ環境(VPython)

Python の三次元グラフィクスモジュールである VPython をインストールする。

インストールするプログラム	python–visual3.2.9-3

・Synpatic パッケージマネージャを使用する場合

１．Synpatic パッケージマネージャを開く

２．キーワード「python-visual」で検索する

３．python-visual にインストール指定をし、適用ボタンをクリックする

参考 URL：http://vpython.org/index.html

３．３．　実機環境

PA10 の実機制御についてはライフロボティクス社製 PA-10 高速制御用ソフトウェア（商用）を使用する。ライブラリのインストールや ARCNET ボードの設置、機器接続については付属の PA-10 用制御ライブラリ取扱説明書（PA-10、1ms 制御用マニュアル）ver.1.1 に従うこと。尚、本ライブラリは有償製品であるため本モジュールには同封されていない。

三菱重工業製汎用ロボット PA10 については以下のサイトを参照のこと。

汎用ロボット PA10 シリーズ

URL：http://www.mhi.co.jp/products/detail/portable_general_purpose_intelligent_arm.html

４．　事前準備

実際に動作させる前に行う設定を以下に記す。

４．１．　ツール位置

アームの先端にツールを取り付ける場合 pa10fk(PA10 順運動学コンポーネント)が提供するサービス set_tool を用いてロボットアーム先端を基準としたツールの作業点までの距離(ツール長)を設定する必要がある。詳細は「関連文書１（3.6.5）」を参照のこと。

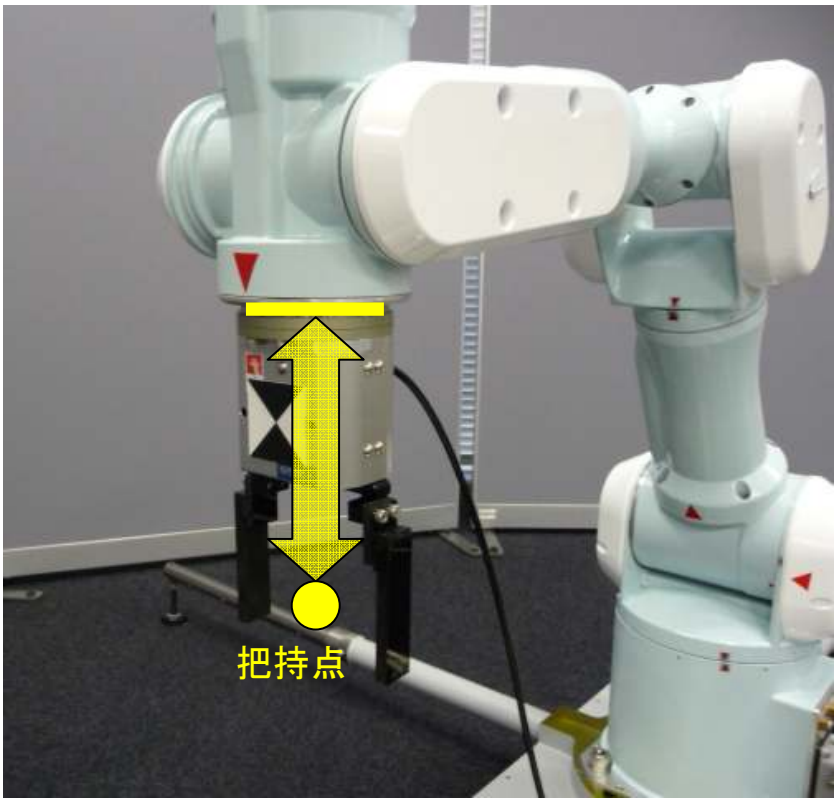

図 4.2-1　ツール長の例

４．２．　アーム先端の速度

アームを動作させる際の移動速度の上限値を設定する場合は、frm_ctrl(軌跡制御コンポーネント)が提供するサービス set_param を用いる。詳細は「関連文書１（3.1.5）」を参照のこと。

４．３．　PATH（Python ソースコード）の修正

Python ソースコード上のディレクトリパスを修正する。ディレクトリパスは、

openrtm/myRTC/script/set_env.py ファイルで指定されており、それを各個人の環境に合わせて変更する。

<ユーザ名：**User** の場合**>**

```
<変更前>
RtmToolsDir="/home/YourName/openrtm/tools"
MyRtcDir="/home/YourName/openrtm/myRTC"
<変更後>
RtmToolsDir="/home/User/openrtm/tools"
MyRtcDir="/home/User/openrtm/myRTC"
```

４．４．　ソースファイルの make

openrtm フォルダ（PA10RMRC_Linux_Debian.tar.gz を解凍したもの）には各 RTC の make 済みファイルが用意されているが、各自で再度 make したい場合は以下の様にする。

make 用のシェルスクリプト

openrtm/myRTC/build.sh

が用意されているのでこれを実行する。

```
$cd openrtm/myRTC
/openrtm/myRTC$sh build.sh
```

実機制御コンポーネント（pa10vel）に関しては、ライフロボティクス社製ライブラリをパスの通ったディレクト下に用意した後、make を実行する。

５．　実行

実行環境にはシミュレータ上で PA10 の動作確認を行うためのシミュレータ環境と、実際に PA10 実機を動かすための実機環境の二つがある。ネームサーバの起動（5.1）、Eclipse の起動（5.2）については両環境において共通の作業である。本章ではまず、シミュレータ環境における実行手順を示し（5.3）、次に実機環境における実行手順を示す（5.4）。

５．１．　ネームサーバの起動

新規にターミナルを開き、ポート番号を 9876 に指定して CORBA ネームサーバを起動する。

```
fujisoft@fujisoft:~$ rtm-naming 9876
Starting omniORB omniNames: fujisoft:9876
fujisoft@fujisoft:~$
Mon Apr 19 15:09:23 2010:

Starting omniNames for the first time.
Wrote initial log file.
Read log file successfully.
Root context is
IOR:010000002b00000049444c3a6f6d672e6f72672f436f734e616d696
e672f4e616d696e67436f6e746578744578743a312e3000000100000000
0000006400000001010200 0f0000003139322e3136382e33302e3134330
000942600000b0000004e616d6553657276696365000200000000000000
0800000001000000005454410100000001c000000010000000100010001 0
0000001000105090101000100000009010100
Checkpointing Phase 1: Prepare.
Checkpointing Phase 2: Commit.
Checkpointing completed.
```

５．２．　RTSystemEditor の起動

Eclipse を立ち上げその上で RTSystemEditor を起動する。RTSystemEditor 上で NameServiceView にネームサーバが起動していることを確認する。RTSystemEditor の使用法は以下のサイトを参照のこと。

RTSystemEditor マニュアル

URL：

http://www.is.aist.go.jp/rt/OpenRTM-aist/html/E3839EE3838BE383A5E382A2E383AB2FRTSystemEditor.html#a943c1e6

５．３． シミュレータ環境

５．３．１． モジュールの起動

ネームサーバ、RTSystemEditor を起動した後、Python ライブラリ RtcHandle を用いて各 RTC の起動、接続、活性化を行う。RtcHandle を用いた具体的な RTC の操作は Python スクリプトに記述してある。

新規のターミナル上で Python を起動し Python スクリプト pa10_test_sim.py を読み込む

```
$cd openrtm/myRTC/script
/openrtm/myRTC/script/$python
Python 2.4.4 (#2, Oct 22 2008, 19:52:44)
[GCC 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>from pa10_test_sim import *
are you sure that every rtc needed are running?
```

Enter キーを押す

```
are you sure that every rtc needed are running?
objcet coord_trans0.rtc was listed.
handle for coord_trans0.rtc was created.
objcet j_jacob0.rtc was listed.
handle for j_jacob0.rtc was created.
objcet tr_jacob0.rtc was listed.
handle for tr_jacob0.rtc was created.
objcet mixer0_py.rtc was listed.
handle for mixer0_py.rtc was created.
objcet jinv0.rtc was listed.
handle for jinv0.rtc was created.
objcet frm_ctrl0.rtc was listed.
handle for frm_ctrl0.rtc was created.
objcet move0.rtc was listed.
handle for move0.rtc was created.
objcet vel_7dof0.rtc was listed.
handle for vel_7dof0.rtc was created.
objcet pa10fk0.rtc was listed.
handle for pa10fk0.rtc was created.
objcet sliderComp0.rtc was listed.
handle for sliderComp0.rtc was created.
objcet pa10disp0.rtc was listed.
handle for pa10disp0.rtc was created.
```

注）Python スクリプトで起動したときに RtcHandle がネームサーバにアクセスしてコンポーネントの情報を取得するが、インスタンス名が取得できない場合、「object ○○.rtc was listed」が表示されず、RtcHandle がコンポーネントをハンドルすることに失敗する。また「object ○○.rtc was created」ではなく「object ○○.rtc was not alived.」と表示された場合も直接コンポーネントにアクセスしようとしてハンドルに失敗している。ハンドルに失敗すると接続や活性化を RtcHandle を用いて行うことはできない。

<使用する RTC>

- frm_ctrl
- tr_jacob
- jinv
- vel_7dof　（シミュレータ環境用 RTC）
- pa10disp　（シミュレータ環境用 RTC）
- pa10fk
- coord_trans
- j_jacob
- mixer　（mixer_py/mixer.py）
- move
- slider

５．３．２．アーム操作

Python スクリプト上で定義された関数を用いてロボットアームの操作を行う。pa10_test_sim.py でアーム操作に使用する関数は以下のとおり。

関数名	引数 1	引数 2	内容
mode_joint	なし	なし	関節角度制御モードにする。
mode_rmrc	なし	なし	分解運動速度制御モードにする。
move_joint	vel_7dof	j_ready, j_park	アーム固定姿勢（待機 or 基本姿勢）へ移行させる。
go_to	move	pos	指定位置に把持点を移動させる。
demo	move	pos_list	go_to の動作を連続して行う。

・待機姿勢へ移行させる

起動時点で基本姿勢（アーム取り付け面に対して直立した姿勢）を取っているアームを関節角度制御モードで待機姿勢へと移行させる。

```
>>>mode_joint()
>>>move_joint(vel_7dof, j_ready)
```

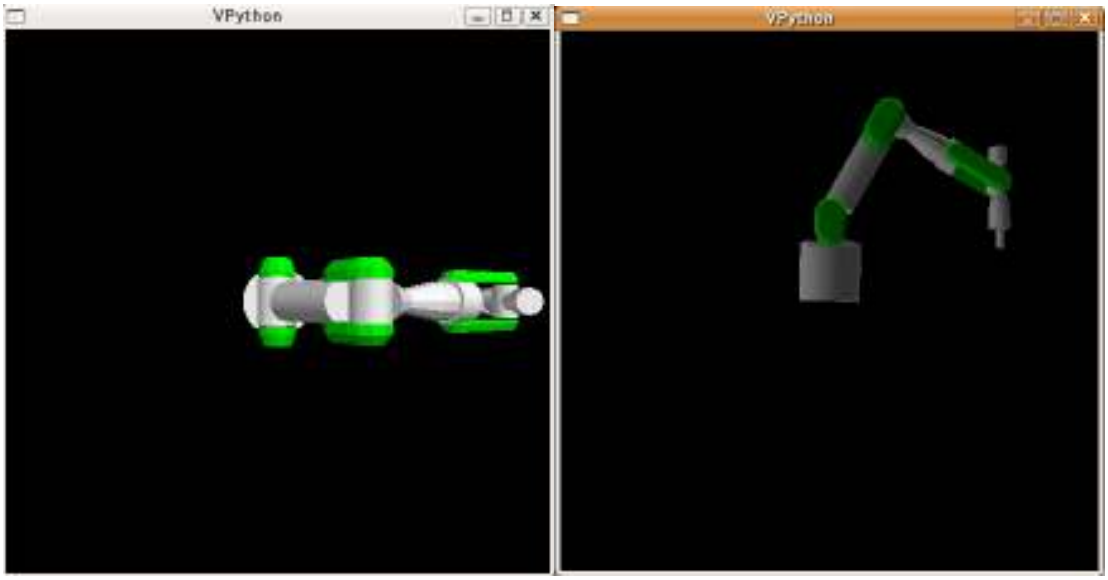

図 5-1 待機姿勢　左：Z 軸正の方向から見たもの　右：Y 軸負の方向から見たもの

シミュレータ画面に対しては以下の機能が使用できる。

- 視点の変更：画面上でマウスの右ドラッグ
- 表示スケールの変更：画面上でマウスの左右同時押しドラッグ

・アームを動かす

アームが待機姿勢を取ったら、分解運動速度制御モードに切り替え、移動先となる把持点を決めアームを動かす。

```
>>>mode_rmrc()
>>>pa = FRAME(xyzabc=[600.0, 0.0, 220.0, 0.0, pi, pi/4])
>>>pb = FRAME(xyzabc=[600.0, 0.0, 500.0, 0.0, pi, pi/4])
>>>pc = FRAME(xyzabc=[600.0, 300.0, 300.0, 0.0, pi, pi/4])
>>>go_to(move, pa)
>>>pos_list = [pc, pb, pa]
>>>demo(move, pos_list)
```

・アーム操作を終了する

アームの操作を止める際は基本姿勢に戻しておく。

```
>>>mode_joint()
>>>move joint(vel 7dof, j park)
```

アームが基本姿勢に戻ったら、起動している全てのコンポーネントを非活性化させる。

```
>>>deactivate_components()
```

Ctrl+C キーを押して各 RTC を終了させる

```
>>>KeyboardInterrupt
>>>
```

本内容より詳細な動作を行いたい場合は以下を参照のこと。

URL：

http://openrtm.sakura.ne.jp/cgi-bin/wiki/wiki.cgi/2009/1A32?page=%CD%F8%CD%D1%A5%DE%A5%CB%A5%E5%A5%A2%A5%EB

５．４．　実機環境

本節では実機 PA10 を操作するための各 RTC の起動方法からアームの操作方法までを示す。

５．４．１．　機器の起動

以下の手順に従って機器を起動する。

1. 機器が正しく接続されていることを確認し、コントローラ電源ケーブルを電源に挿す（電源ケーブルは通常電源から抜いておくものとする）
2. 非常停止ボタンを手元に置く
3. コントローラの電源を入れる

５．４．２．　モジュールの起動

５．３．１と同様にしてモジュールを起動する。使用する RTC は以下の通りである。

<使用する RTC>

- frm_ctrl
- tr_jacob
- jinv
- **pa10vel**　（実機環境用 RTC）
- pa10fk
- coord_trans
- j_jacob
- mixer　（mixer_py/mixer.py）
- move
- slider

新規のターミナル上で Python を起動し Python スクリプト pa10_test.py を読み込む

```
$cd openrtm/myRTC/script
/openrtm/myRTC/script/$python
Python 2.4.4 (#2, Oct 22 2008, 19:52:44)
[GCC 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>from pa10_test import *
are you sure that every rtc needed are running?
```

５．４．３． アーム操作

(5.3.2)の場合と同様、実機操作用の Python スクリプト pa10_test.py に記述された関数（pa10_test_sim.py のものと同じ）を使用してロボットアームの操作を行う。

関数名	引数 1	引数 2	内容
mode_joint	なし	なし	関節角度制御モードにする。
mode_rmrc	なし	なし	分解運動速度制御モードにする。
move_joint	**pa10vel**	j_ready, j_park	アーム固定姿勢（待機 or 基本姿勢）へ移行させる。
go_to	move	pos	指定位置に把持点を移動させる。
demo	move	pos_list	go_to の動作を連続して行う。

・move_joint の第一引数を **pa10vel とする**ことに注意する

待機姿勢へ移行させる

```
>>>mode_joint()
>>>move_joint(pa10vel, j_ready)
```

基本姿勢へ戻す

```
>>>mode_joint()
>>>move_joint(pa10vel, j_park)
```

５．４．４． 実機の終了手順

アームが基本姿勢へ戻っていることを確認し、以下の手順で実機を終了する。

1. 起動している RTC を全て終了させる

各 RTC の非活性化

```
>>>deactivate_components()
```

Ctrl+C キーを押して各 RTC を終了させる

```
>>>KeyboardInterrupt
>>>
```

2. 非常停止ボタンを押す
3. コントローラの電源を切る
4. コントローラ電源ケーブルを電源から引き抜く

５．４．５．　実機異常動作時の対応

実機操作を行った後は以下の手順で終了する。

1. 非常停止ボタンを押す
2. 各 RTC を終了させる
3. コントローラの電源を切る
4. コントローラ電源が落ちてから 5 秒間経ったことを確認し、再びコントローラの電源を入れる
5. 非常停止ボタンを回し、ブレーキを解除する
6. モジュールを起動しなおし、実機を基本姿勢に戻す

```
>>>mode_joint()
>>>move_joint(pa10vel, j_park)
```

６．　トラブルシューティング

Python スクリプトで RTC が立ち上がらない

・ネームサーバが正しく起動していない場合

[対策] ネームサーバが立ち上がっているか、また、ポートは 9876 に指定しているかどうか確認する（5.1）。ネームサーバを立ち上げず Rtc_Handle を使用し RTC をハンドルしようとすると下記のエラーがでる。

```
Python 2.4.4 (#2, Oct 22 2008, 19:52:44)
[GCC 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from pa10_test import *
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "pa10_test.py", line 215, in ?
    env = RtmEnv(sys.argv,["localhost:9876"])
  File "/home/fujisoft/openrtm/tools/rtc_handle/rtc_handle.py", line
31, in __init__
    self.name_space[ns]=NameSpace(orb, server_name=ns)
  File "/home/fujisoft/openrtm/tools/rtc_handle/rtc_handle.py", line
47, in __init__
    self.naming = CorbaNaming(self.orb, server_name)
  File "/usr/lib/python2.4/site-packages/OpenRTM/CorbaNaming.py", line
89, in __init__
    self._rootContext = obj._narrow(CosNaming.NamingContext)
  File "/usr/lib/python2.4/site-packages/omniORB/CORBA.py", line 667, in
_narrow    return _omnipy.narrow(self, dest._NP_RepositoryId)
omniORB.CORBA.TRANSIENT: Minor: TRANSIENT_ConnectFailed, COMPLETED_NO.
>>>
```

図 6-1 RtcHandle の動作に対する omniORB のエラー

・スクリプトに記述されたパスが正しくない場合

[対策] スクリプト内のパス記述が正しいかどうか確認する（4.3）。

・正しいグローバル IDL ファイルが作成されていない

[対策] サービスが実装された RTC に対して IDL コンパイルをし直す（4.4）。

Python で記述された RTC を RtcHandle からハンドルできない

「5.3.1.モジュールの起動」の注)（P.12 下部）にある内容が起こり、RTC のハンドルができない。

・OpenRTM-aist-0.4.2-RELEASE、OpenRTM-aist-Python-0.4.1-RELEASE の既知の不具合に未対応の場合
[対策] 修正パッチをあてる（3.1.1)、(3.1.2)。

シミュレーション時にコンソールから入力できなくなる

本モジュールをシミュレータ環境で動作させている際に、一見、コンソール入力ができなくなってしまったかの様な状態になる場合がある。これはアーム先端の位置・姿勢が指定した目標値に未到達であるために起こる現象で、十分時間が経ちアーム先端の位置・姿勢が目標値に到達すれば再びコンソールから次の入力ができるようになる。即座に状況を復旧したい場合には以下の処理をすれば良い。

・ move（Python 版）を使用してモジュールを動作させていた場合
[対策]全ての RTC 群を終了させ、再び RTC 群を起動しなおす。

7. 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。

- 本モジュールは独立行政法人 新エネルギー･産業技術総合開発機構の「次世代ロボット知能化技術開発プロジェクト」内実施者向けに評価を目的として提供するものであり、商用利用など他の目的で使用することを禁じます。
- 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。
- PA-10 高速制御用ソフトウェアは、ライフロボティクス株式会社の製品であり、権利はライフロボティクス株式会社に帰属します。

【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