

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
(Windows)

V e r . 1 . 2

2010年4月9日

R T C 再利用技術研究センター

改版履歷

[illegible]

目次

改版履歴	i
目次	ii
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	1
2. ソースディレクトリ構成	2
3. ソフトウェアインストール	3
3. 1. 基本環境	3
3. 2. シミュレータ環境(VPython)	5
3. 3. 実機環境	5
4. 事前準備	6
4. 1. IDL コンパイルの実行	6
4. 2. ツール位置	7
4. 3. アーム先端の速度	7
4. 4. 待機姿勢	7
4. 5. PATH の修正	8
4. 6. RTC のコンパイル	9
5. 実行	10
5. 1. ネームサーバの起動	10
5. 2. RTSystemEditor の起動	10
5. 3. シミュレータ環境	11
5. 4. 実機	21
6. トラブルシューティング	28
7. 特記事項	31

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 (Windows) 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

検証に用いた動作環境は以下のとおりである。

OS	Windows XP Professional SP3
RT ミドルウェア	OpenRTM-aist-0.4.2 RELEASE (C++)
	OpenRTM-aist-Python-0.4.1 RC1
開発言語	C++, Python
コンパイラ	Microsoft Visual C++ 2008 Express Edition
インタプリタ	Python 2.5.1、Python 2.4.4
依存ライブラリ (OpenRTM)	OmniORB-4.1.2
	OmniORBpy-3.1
	ACE-5.6
依存ライブラリ (その他)	VPython-Py2.5-3.2.11
	三菱重工業製 PA ライブラリ (商用)

2. ソースディレクトリ構成

本書では下表のディレクトリ構成のもと話をする。ディレクトリ構成が下表と異なる場合は、適宜その環境に合わせた修正（4.5）が必要になる。

ディレクトリ	言語	内容	備考
C:\PA10RMRC	-	-	-
└ pa10	-	-	-
├└ bin	-	各 RTC のバイナリ、conf ファイル	-
├└ src	-		-
├├└ frm_ctrl	C++	軌跡制御コンポーネント	-
├├└ jinv	C++	逆ヤコビ行列変換コンポーネント	-
├├└ mixer	C++	ヤコビシーケンス生成コンポーネント	-
├├└ move	C++	操作制御コンポーネント	コンソールから操作
├├└ move_py	Python	操作制御コンポーネント	script から制御する場合に使用
├├└ pa10disp	Python	PA10 幾何モデル描画コンポーネント	-
├├└ pa10fk	C++	PA10 順運動学計算コンポーネント	-
├├└ tools	-	-	-
├├├└ geo	C++/Python	幾何演算ライブラリ	pa10fk 等で使用
├├├└ nr	C	行列演算ライブラリ	jinv、move 等で使用
├├├└ v_robot	Python	PA10 シミュレータ描画処理	pa10disp で使用
├├└ vel_7dof	C++	PA10 シミュレータコンポーネント	-
├├└ pa10vel	C++	PA10 実機制御コンポーネント	-
├└ scripts	Python	RT システム起動スクリプト	-
├└ patch	-	RtcHandle 不具合修正 bat ファイル	-
├└ profiles	XML	各 RTC のプロファイル	-
└ package	-	OpenRTM、VPython のインストーラ	-

3. ソフトウェアインストール

動作に必要なソフトウェアを以下に記す。OpenRTM (3.1.2 - 3.1.3) と Vpython (3.2) のインストールは、¥PA10RMRC¥package フォルダ内のインストーラを使用すること。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. Microsoft Visual C++ 2008 Express Edition

C++言語で実装された RTC のコンパイルを行うために必要な Microsoft の統合開発環境 Visual C++ 2008 Express Edition を以下のサイトからダウンロードし、インストールする。

- ・ダウンロード先 URL : <http://www.microsoft.com/japan/msdn/vstudio/express/>

3. 1. 2. 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。以下に示されるのは Visual C++ 2008(VC9)に対応したものである。

インストーラ	インストールされるプログラム
ACE-5.6_vc9.msi	ACE-5.6
omniORB-4.1.2_vc9.msi	omniORB4.1.2
python-2.4.4.msi	Python2.4.4
OpenRTM-aist-0.4.2-jp_vc9.msi	OpenRTM-aist-0.4.2-RELEASE ※

※ OpenRTM-aist-0.4.2-RELEASE は最後にインストールすること

当センターでは本知能モジュールの制御に RtcHandle¹を使用した Python スクリプトを用いることを推奨するが、これを正常に動作させるためには OpenRTM-aist-0.4.2 の既知の不具合を修正する必要がある。修正作業は¥PA10RMRC¥pa10¥patch にある repair.bat ファイルにまとめてあるので、OpenRTM-aist-0.4.2-RELEASE のインストール完了後 repair.bat ファイルを実行するだけで、Python スクリプトを用いたモジュール制御が可能となる。

参考 : OpenRTM-BUG (staff.aist.go.jp)

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

¹ RTC を Python 環境から簡単に扱うことができる Python モジュール。参考 URL : “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

3. 1. 3. OpenRTM-aist-Python-0.4.1 RC1

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。

インストーラ	インストールされるプログラム
python-2.5.1.msi	Python2.5.1
omniORBpy-3.1.msi	omniORBpy-3.1
OpenRTM-aist-Python25-0.4.1-RC1.win32.exe	OpenRTM-aist-Python25-0.4.1-RC1

omniORBpy や OpenRTM-aist-Python は Python2.5 のインストールディレクトリ配下にインストールすること。環境変数に Python2.5 インストールディレクトリを指定すること。また、3.1.2 で Python2.4.4 をインストールしたが、それも削除せずに残しておくこと。

3. 1. 4. 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。

インストーラ	インストールされるプログラム
VPython-Win-Py2.5-3.2.11.exe	VPython-Py2.5-3.2.11

3. 3. 実機環境

PA10 の実機制御については三菱重工業製の PA ライブラリ（商用）を使用する。ライブラリのインストールや運動制御ボードの設置、機器接続については付属のソフトウェアインストールマニュアルに従うこと。尚、本ライブラリは有償製品であるため ¥PA10RMRC¥package フォルダには入っていない。

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. IDL コンパイルの実行

RTC に実装されたサービスを使用可能なものにするために IDL ファイルをコンパイルする必要がある。本知能モジュールでは `move` (Python 版)、`frm_ctrl` 及び `pa10fk` にサービスが実装されている。`frm_ctrl` 及び `pa10fk` に関しては、各 RTC の VC++ プロジェクトファイルに従ってコンパイルすれば、統合開発環境がソースコードをコンパイルする前に IDL コンパイルを前もって行ってくれる。しかし、`move` (Python 版) に関しては、各自で IDL コンパイルをする必要がある。以下にその方法を示す。

1. コマンドプロンプトを開く
2. `C:¥PA10RMRC¥pa10¥src¥move_py` ディレクトリへ移動する
3. `omniidl -bpython com_move.idl` と入力し Enter キーを押す

```
C:¥ホームディレクトリ>cd C:¥PA10RMRC¥pa10¥src¥move_py
C:¥PA10RMRC ¥pa10¥src¥move_py>omniidl -bpython com_move.idl
```

図 4.1-1 omniidl コマンドを用いた IDL コンパイル

上記手順で IDL コンパイルを行うと、同ディレクトリ内に `_GloabalIDL`、`_GloabalIDL_POA` フォルダが作成される。

また、Python スクリプトを使用して RTC を操作する場合にも、使用する IDL ファイル全てに対して IDL コンパイルを行う。詳細は下記を参照のこと。

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

(上記サイト内、RTC 利用環境の構築例－事前の利用準備－必要なサービスの idl のコンパイル)

```
C:¥PA10RMRC¥pa10¥script>omniidl -bpython com_move.idl
C:¥PA10RMRC¥pa10¥script>omniidl -bpython com_frm_ctrl.idl
C:¥PA10RMRC¥pa10¥script>omniidl -bpython com_fk.idl
```

図 4.1-2 Python スクリプトで使用する IDL ファイルの IDL コンパイル

4. 2. 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「機能仕様書 (3.4.6)」を参照のこと。

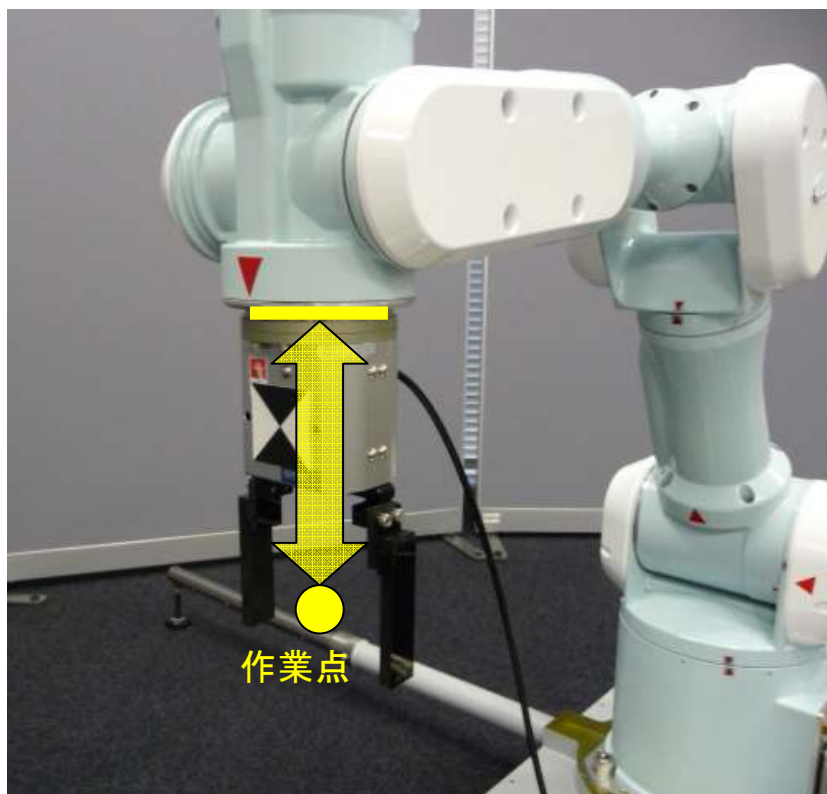


図 4.2-1 ツール長の例

4. 3. 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「機能仕様書 (3.1.6)」を参照のこと。

4. 4. 待機姿勢

本知能モジュールが起動した直後にアームが取る待機姿勢は、`vel_7dof`(PA10 シミュレータコンポーネント)あるいは `pa10vel`(PA10 実機制御コンポーネント)のソースコードに各軸角度を直接記述する。詳細は「機能仕様書 (3.5.6) (3.8.6)」を参照のこと。

4. 5. PATH の修正

以下の二つの場合にはファイル上で指定されたディレクトリパスを修正する必要がある。

1. ソースディレクトリ構成が本書のものと異なる場合

Python ファイル内で指定されているディレクトリパスを修正する必要がある。

- %PA10RMRC%\pa10\script%\pa10sim.py
- %PA10RMRC%\pa10\script%\pa10act.py
- %PA10RMRC%\pa10\src%\pa10disp%\pa10disp.py
- %PA10RMRC%\pa10\src%\tools%\v_robot%\v_pa10.py

下記を参考に修正すること。

※ C:%配下に設定されている場合

```
#!/usr/bin/env python
# -*- Python -*-
# @pa10sim
#
# Robot Arm(MHI-PA10) Resolved Motion Rate Control Components.

## @Library Source Path.
# To Be Edited by User.
RtmToolsDir="C:%PA10RMRC%\pa10\script"
## @Source Path.
# To Be Edited by User.
MyRtcDir="C:%PA10RMRC%\pa10\bin"

<省略>
```

※ ↓C:%mydir%\user 配下に設定する場合

```
RtmToolsDir="C:%mydir%\user%\PA10RMRC%\pa10\script"
MyRtcDir="C:%mydir%\user%\PA10RMRC%\pa10\bin"

<省略>
```

図 4.5-1 pa10_sim.py の修正例

2. Visual C++のを標準のディレクトリ(C:¥Program Files)以外にインストールした場合
バッチファイル上のパスを修正する

※ ↓C:¥Program Files 配下に設定されている場合

```
@echo off
title RTC COMPILE ..
@rem Direcorry Path has VCBUILD.EXE
@set PATH="C:¥Program Files¥Microsoft Visual Studio 9.0¥VC¥vcpackages";%PATH%
<省略>
```

※ ↓C:¥mydir¥user 配下に設定する場合

```
@set PATH="C:¥mydir¥user¥Microsoft Visual Studio 9.0¥VC¥vcpackages";%PATH%
<省略>
```

図 4.5-2 バッチファイルの修正例

4. 6. RTC のコンパイル

C++言語で記述された以下の RTC は実行前にビルドしておくこと。

- frm_ctrl
- jinv
- mixer
- move (C++版)
- pa10fk
- pa10vel
- vel_7dof

¥PA10RMRC¥pa10¥src フォルダの buildall.bat を実行することで上記の全 RTC をビルドできる。実行しているコンソールウィンドウのタイトルにビルド中のコンポーネントを表示する。

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

ポート番号を 9876 に指定して CORBA ネームサーバを起動する。

```
C:\Program Files\OpenRTM-aist\0.4\bin>rtm-naming 9876
Starting omniORB omniNames: twatana:9876

Wed Sep 01 14:13:37 2009:

Starting omniNames for the first time.
Wrote initial log file.
Read log file successfully.
Root context is
IOR:010000002b00000049444c3a6f6d672e6f72672f436f734e61
6d696e672f4e616d696e67436f6e746578744578743a312e300000
01000000000000006c000000010102000a00000006c6f63616c686f
73740094260b0000004e616d655365727669636500030000000000
0000080000000100000000545441010000001c0000000100000001
000100010000000100010509010100010000000901010003545441
0800000001ff9d4a01000cc8
Checkpointing Phase 1: Prepare.
Checkpointing Phase 2: Commit.
Checkpointing completed.
```

図 5.1-1 コマンドプロンプトでネームサーバを起動

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

本節ではシミュレータ上で PA10 の動作確認を行うための各 RTC の起動方法から、アームの操作方法までを示す。5.3.1 から 5.3.4 までは一般的な操作方法の説明となる。当センターでは作業の効率化を図るため Python スクリプトを用いた RTC 制御を推奨しており、それについては 5.3.5 に記述してある。

5. 3. 1. RTC の起動

シミュレータ環境において起動する RTC は、

- frm_ctrl
- mixer
- 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
- move

になる。

C++言語で実装された RTC に関しては、コンパイル (4.6) を済ませた後、各 RTC を一つずつ起動する。以下では例として frm_ctrl コンポーネントの場合を示す。

1. コマンドプロンプトを開く
2. C:\¥PA10RMRC¥pa10¥src¥frm_ctrl¥components ディレクトリに移動する
3. カレントディレクトリ内に frm_ctrlComp.exe、rtc.conf があることを確認する
4. コンソールに frm_ctrlComp と入力し Enter キーを押す

```
C:\¥ホームディレクトリ>cd C:\¥PA10RMRC¥pa10¥src¥frm_ctrl¥components
C:\¥PA10RMRC¥pa10¥src¥frm_ctrl¥components>dir
2009/10/23  14:12    <DIR>          .
2009/10/23  14:12    <DIR>          ..
2009/10/20  15:08                163,328 frm_ctrlComp.exe
2009/09/07  11:34                 81 rtc.conf
C:\¥PA10RMRC¥pa10¥src¥frm_ctrl¥components>frm_ctrlComp.exe
```

図 5.3-1 コンソール上での frm_ctrl 起動例

他の RTC (jinv、mixer、move(C++版)、pa10fk、vel_7dof) についても同様に起動する。

pa10disp の起動は、C:\¥PA10RMRC¥pa10¥src¥pa10disp ディレクトリへ移動した後、
python pa10disp.py
と入力して Enter キーを押す。

```
C:\¥ホームディレクトリ>cd C:\¥PA10RMRC¥pa10¥src¥pa10disp
C:\¥PA10RMRC¥pa10¥src¥pa10disp>dir
2009/10/23  14:55    <DIR>          .
2009/10/23  14:55    <DIR>          ..
2009/10/20  15:53                3,563 pa10disp.py
2009/07/28  12:37                2,547 README.pa10disp
2009/08/20  15:24                 81 rtc.conf
2009/07/28  12:37                2,240 RTC.xml
C:\¥PA10RMRC¥pa10¥src¥pa10disp>python pa10disp.py
```

図 5.3-2 コンソール上での pa10disp の起動例

これまでコンソール上で RTC の実行ファイルを指定して起動する方法を述べたが、エクスプローラで frm_ctrl¥components フォルダを開き、frm_ctrlComp.exe のアイコンをダブルクリックしても良い。

5. 3. 2. 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。ポートの詳細は「ロボットアーム(PA10)分解運動速度制御モジュール機能仕様書」を参照のこと。

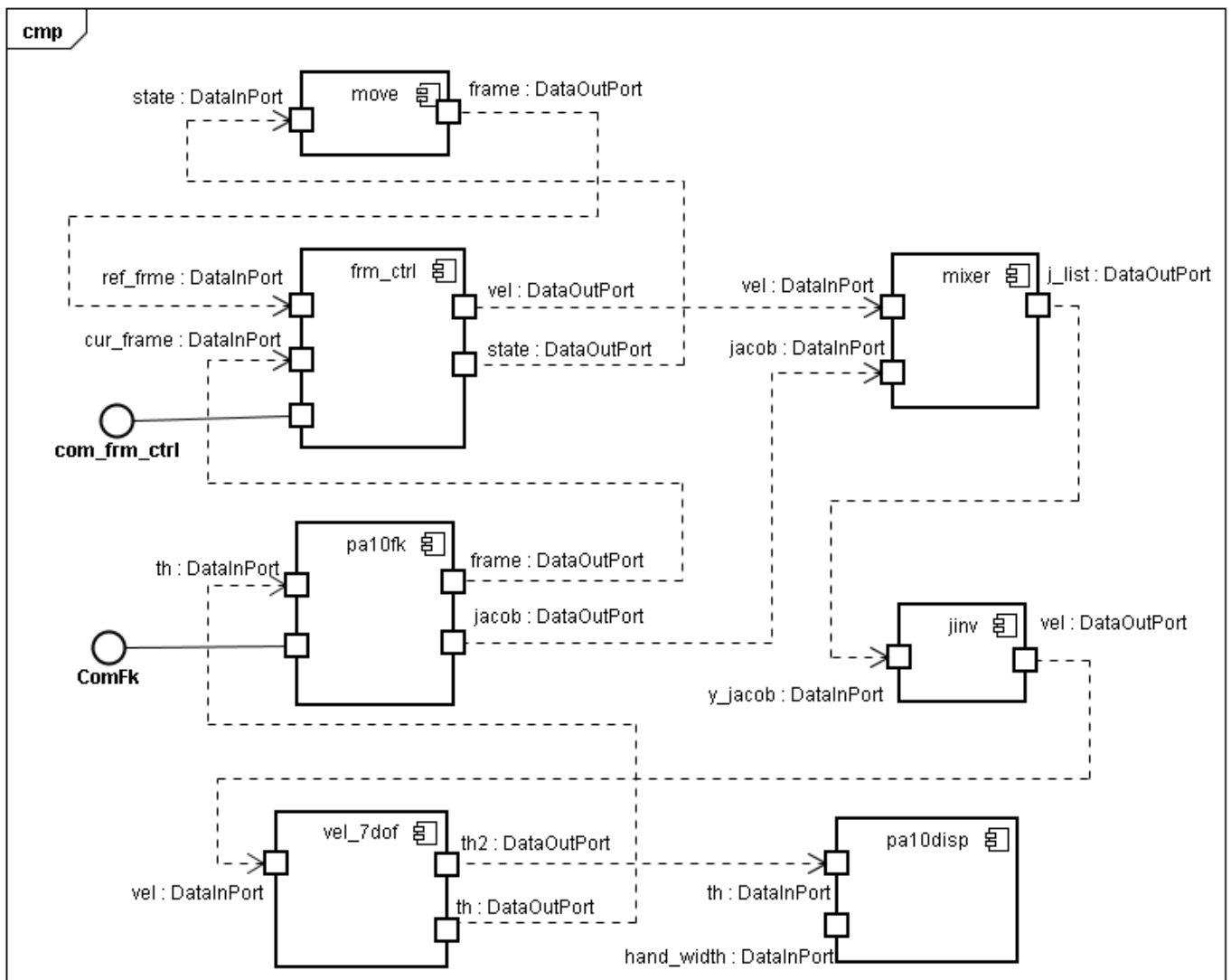


図 5.3-3 RTC とポート名

5. 3. 3. 活性化

接続が完了したら、下記の順に各 RTC の活性化を行う。RTSystemEditor 上で RTC を選択し、サブメニューから”activate”を選択することで活性化する。全 RTC が活性化された状態を図 5.3-4 に示す。

1. vel_7dof (シミュレータ環境用 RTC)
 2. pa10disp (シミュレータ環境用 RTC)
 3. pa10fk
 4. frm_ctrl
 5. mixer
 6. jinv
 7. move (C++版)
- ※ move はコンソール入力を受けた後に活性化される

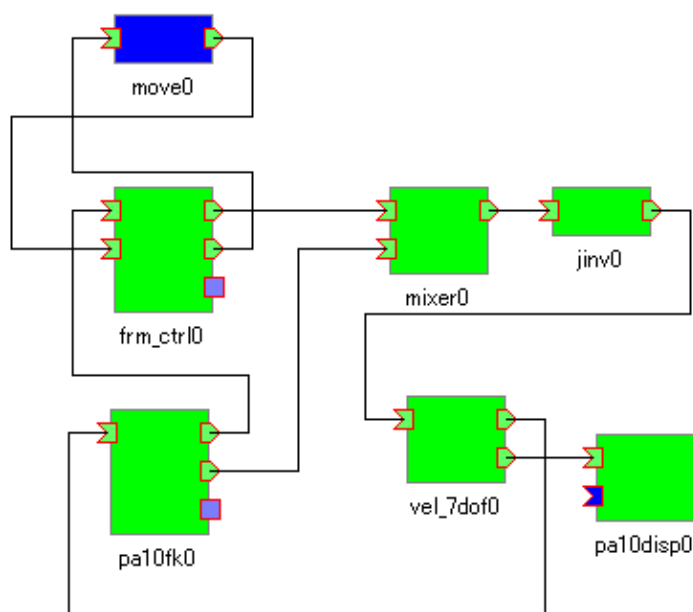


図 5.3-4 接続された RTC(活性)の例

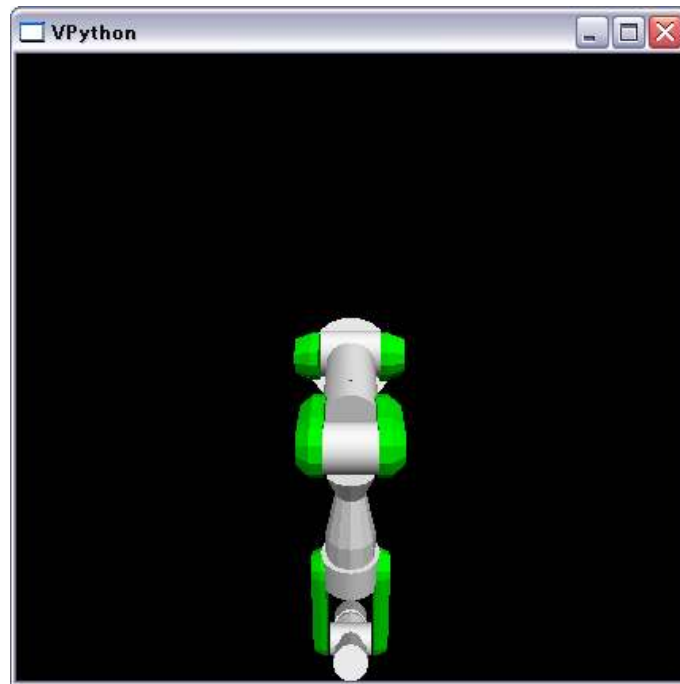


図 5.3-5 活性化された後の VPython シミュレータ画面

図 5.3-5 は、待機姿勢を取った PA10 のモデルを真上（Y 軸正の方向）から見たものである。シミュレータ画面に対しては以下の機能を使用できる。

- ・ 画面上でマウスを右クリックしドラッグすることで視点を変えることができる
- ・ 画面上でマウスの左右ボタンを同時にクリックしドラッグすることで視野スケールを変えることができる

5. 3. 4. アーム操作

`move` (C++版) のコンソールから目標位置座標と姿勢²を入力することでシミュレータ上のアームを動作させることができる。本コンポーネントではあらかじめ可動範囲が定義されており、入力した座標が可動範囲外だった場合はエラーが表示される(機能仕様書(3.7.1.6))。以下が `move` のコンソールに入力する移動先位置座標と姿勢の書式である。座標・姿勢を入力して `Enter` を押すとアームが移動する。

[X 軸座標] [Y 軸座標] [Z 軸座標] [ロール] [ピッチ] [ヨー]

- ※ 座標はベース座標系で単位は[mm]で指定する。
- ※ ロール、ピッチ、ヨーの単位は[度]で指定する。
- ※ 数値の間に半角スペースをはさむ。

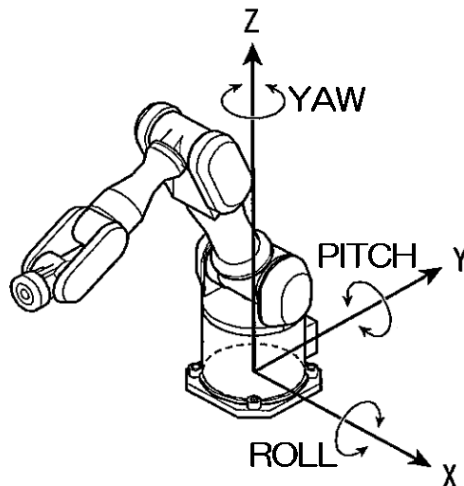


図 5.3-6 コンソール操作の際のベース座標系

```
Please input x,y,z [mm] and a,b,c [deg]:500 400 300 0 180 0
vtr 500 400 300
abc 0 3.14159 0
write
end
state = 1
Please input x,y,z [mm] and a,b,c [deg]:
```

図 5.3-7 `move` のコンソール入力例

² 姿勢は X,Y,Z 軸のまわりの回転角度で決める。

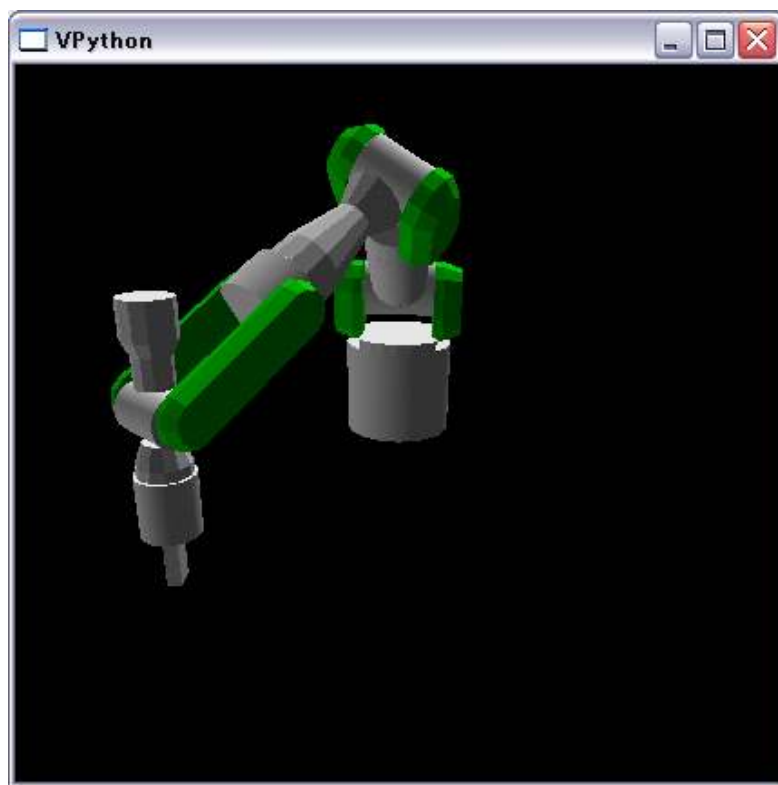


図 5.3-8 シミュレータ出力結果の例

ロボットアームの操作を終了する際は、まず RTSystemEditor 上で各 RTC の非活性化を行う。その後、非活性化した RTC と同じ名前のコンソールを閉じていく。もしくは ¥PA10RMRC¥pa10¥script にある KillRTCall.bat ファイルを実行しても良い。

5. 3. 5. スクリプト操作

5. 3. 5. 1. RTC 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 RTC の接続、活性化を自動で行う。

コマンドプロンプトを開き、C:\¥PA10RMRC¥pa10¥script ディレクトリに移動して Python を起動する。pa10sim.py を読み込むことでシミュレータ環境における PA10 分解運動速度制御モジュールの RTC 群の起動、接続、活性化が行われる。

pa10sim.py では以下の RTC を起動する。

- frm_ctrl
- mixer
- jinv
- move (Python 版)
- pa10disp (シミュレータ環境用 RTC)
- pa10fk
- vel_7dof (シミュレータ環境用 RTC)

pa10sim.py では move (Python 版) を起動していることに注意すること。

```
C:\¥PA10RMRC¥pa10¥script>python
Python 2.5.1 (r251:54863, Apr 18 2007, 08:51:08) [MSC
v.1310 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more
information.
>>>from pal0sim import *
    Wait for start up.. 5 4 3 2 1 0 OK!
objcet jinv0.rtc was listed.
handle for jinv0.rtc was created.
objcet mixer0.rtc was listed.
handle for mixer0.rtc was created.
objcet vel_7dof0.rtc was listed.
handle for vel_7dof0.rtc was created.
objcet frm_ctrl0.rtc was listed.
handle for frm_ctrl0.rtc was created.
objcet pal0fk0.rtc was listed.
handle for pal0fk0.rtc was created.
objcet move0.rtc was listed.
handle for move0.rtc was created.
objcet pal0disp0.rtc was listed.
handle for pal0disp0.rtc was created.
    Wait for activate.. 14 13 12 11 10 9 8 7 6 5 4 3 2 1 OK!
```

図 5.3-9 Python スクリプトによる自動起動を行ったコンソール画面

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 経由から行うことはできない。

5. 3. 5. 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。
pa10sim.py で使用するアーム操作関数は以下のとおり。

関数名	引数 1	引数 2	内容
go_to	move	pos	指定位置に把持点を移動させる。
auto	move	pos_list	go_to の動作を連続して行う。
hand_open	なし	なし	PA10 に取り付けられたハンドを開く
hand_close	なし	なし	PA10 に取り付けられたハンドを閉じる

・アームを動かす

移動先となる把持点を決め、アームを動かす。

```
>>>pa = FRAME(xyzabc=[700.0, 300.0, 200.0, 0.0, pi, pi])
>>>pb = FRAME(xyzabc=[700.0, -300.0, 320.0, 0.0, pi, pi])
>>>pc = FRAME(xyzabc=[700.0, 0.0, 350.0, 0.0, pi, pi])
>>>go_to(move, pa)
>>>pos_list = [pc, pb, pa]
>>>auto(move, pos_list)
>>>hand_open(pa10disp)
>>>hand_close(pa10disp)
```

図 5.3-10 関数を用いた制御の例

※デモンストレーションとして一連のアーム動作を行うスクリプト pa10_with_hand.py も参照のこと。

```
>>>python
:
>>>from pa10_with_hand import *
:
>>>pickup_place()
```

ロボットアームの操作を終了する際は、Python スクリプトに記述された deactivate() 、end_comp()関数を用いて RTC を終了させる。

```
>>>deactivate()
>>>end_comp()
>>>
```

図 5.3-11 関数を用いた RTC 終了の例

5. 4. 実機

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 から 5.4.5 までは一般的な操作方法の説明となる。当センターでは作業の効率化を図るため Python スクリプトを用いた RTC 制御を推奨しており、それについては 5.4.6 に記述してある。

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. 機器が環境構成仕様書通りに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. RTC の起動

5. 3. 1 と同様にして各 RTC を起動する。起動する RTC は以下の通りである。

実機環境において起動する RTC は、

- frm_ctrl
- mixer
- jinv
- pa10vel （実機環境用 RTC）
- pa10fk
- move （C++版）

になる。

5. 4. 3. 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。ポートの詳細は機能仕様書を、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>

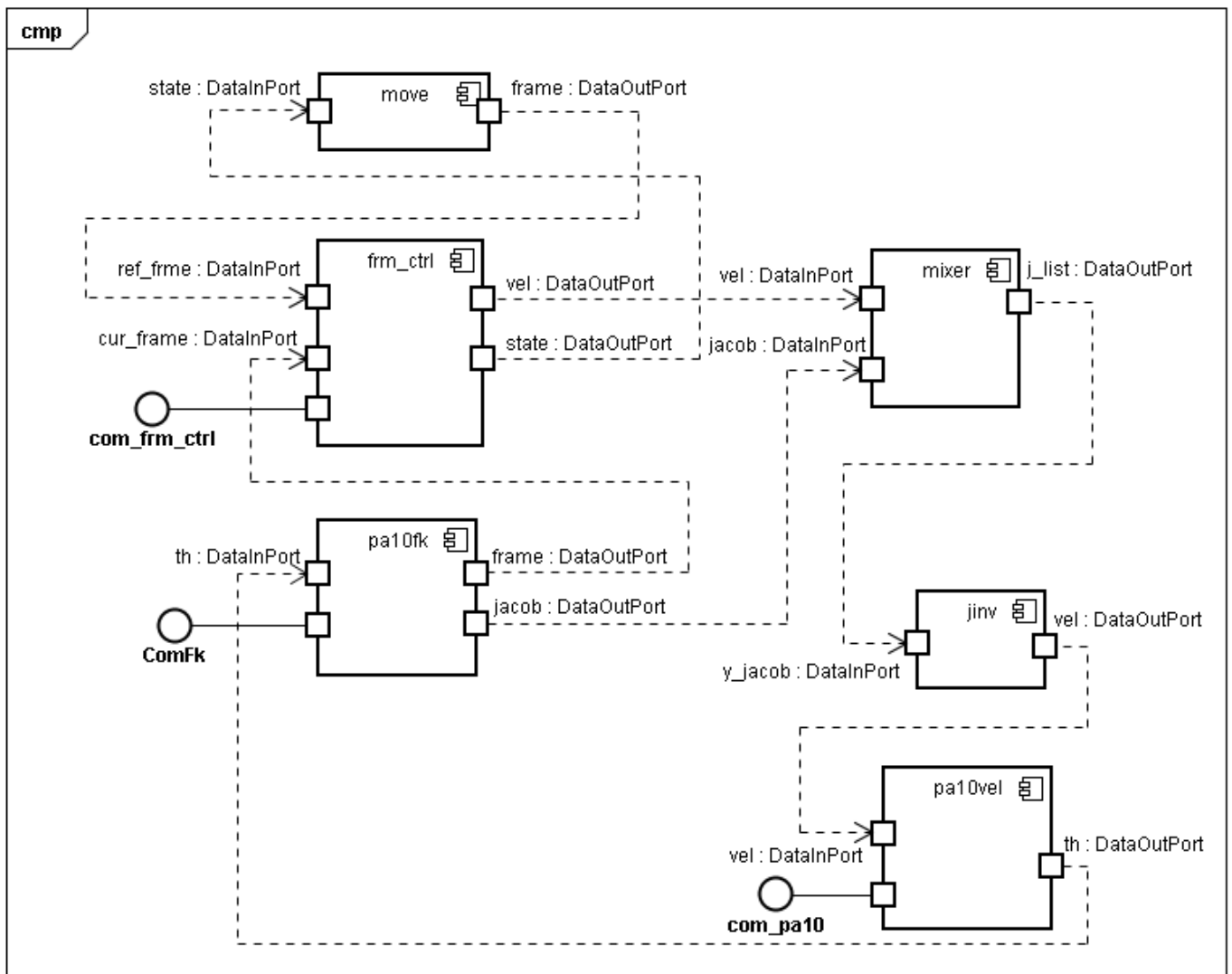


図 5.4-1 RTC とポート名

5. 4. 4. 活性化

接続が完了したら、下記の順に RTC を活性化する。RTSystemEditor 上で RTC を選択し、サブメニューから”activate”を選択することで活性化する。全 RTC を活性化すると下図のようになる。

1. pa10vel (実機環境用 RTC)
2. pa10fk
3. move (C++版)
4. frm_ctrl
5. mixer
6. jinv

※ 注意：pa10vel が活性化されると実機が待機姿勢へと移行するので注意すること。また本コンポーネントは速度制御モードになり、データポート vel の入力データを待ち受ける。不適切なデータが入力されると意図しない動作が実行される可能性があるため、順序どおりに活性化を行うこと。

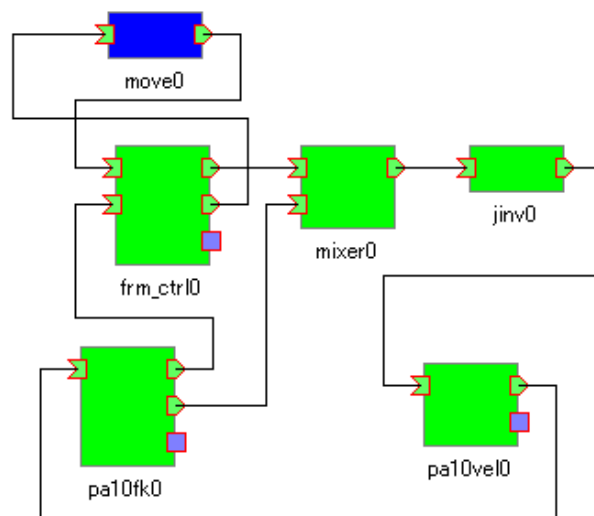


図 5.4-2 接続された RTC(活性)の例

5. 4. 5. アーム操作

move (C++版) のコンソールから目標位置座標と姿勢を入力することで実機を動作させることができる。入力方法はシミュレーションのアーム操作(5.3.4 参照)と同様である。

実機の操作を終了する際は、実機制御を行う pa10vel を最初に終了させ（**コンソールをアクティブにして Ctrl+C を押す**）、その後残りの RTC を 5.3.4 のときと同様にして終了させる。pa10vel を終了させる際、PA10 は基本姿勢（アーム取り付け面に対して直立した姿勢）へ移行するので注意すること。

5. 4. 6. スクリプト操作

5. 4. 6. 1. RTC の起動

コマンドプロンプトを開き、`C:\PA10RMRC\pa10\script` ディレクトリに移動して Python を起動する。`pa10act.py` を読み込むことで PA10 分解運動速度制御モジュールの RTC 群の起動、接続、活性化が行われる。

`pa10act.py` では以下の RTC を起動する。

- `frm_ctrl`
- `mixer`
- `jinv`
- `move` (Python 版)
- `pa10vel` (実機環境用 RTC)
- `pa10fk`

`pa10act.py` では `move` (Python 版) を起動していることに注意すること。

```
C:\¥PA10RMRC¥pa10¥script>python
Python 2.5.1 (r251:54863, Apr 18 2007, 08:51:08) [MSC v.1310 32 bit
(Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>>from pa10act import *
pa10 moves. ok?
are you sure that every rtc needed are running?
objcet vel_7dof0.rtc was listed.
vel_7dof0.rtc is not alive.
objcet move0.rtc was listed.
move0.rtc is not alive.
objcet pa10disp0.rtc was listed.
pa10disp0.rtc is not alive.
objcet mixer0.rtc was listed.
handle for mixer0.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 pa10fk0.rtc was listed.
handle for pa10fk0.rtc was created.
objcet pa10vel0.rtc was listed.
handle for pa10vel0.rtc was created.
>>>
```

図 5.4-3 Python スクリプトによる自動起動

活性化されると下図のようになる。

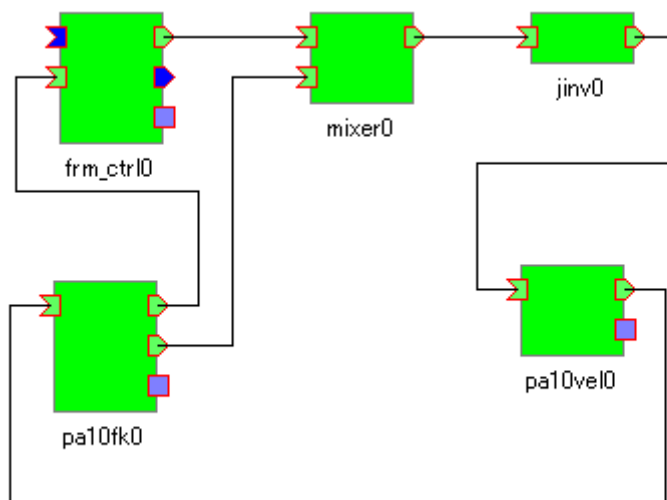


図 5.4-4 接続された RTC(活性)の例

5. 4. 6. 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。
pal0test.py で使用するアーム操作関数は以下のとおり。

関数名	引数 1	引数 2	内容
go_to	frm_ctrl	pos	指定位置に把持点を移動させる。
auto	frm_ctrl	pos_list	go_to の動作を連続して行う。

・ **go_to, auto** の第一引数に **move** ではなく **frm_ctrl** を与える仕様になっていることに注意する

ロボットアームの動作終了操作は、`deactivate0`関数を使用し各 RTC の非活性化を行った後、`pal0vel` を終了させ（コンソールをアクティブにして **Ctrl+C** を押す）、その後 (5.3.5.2) と同様にして残りの RTC の終了を行う。

5. 4. 7. 実機の終了手順

実機操作を行った後は以下の手順で終了する。

1. 各 RTC が終了していることを確認する
2. コントローラの電源を切る
3. コントローラ電源ケーブルを電源から引き抜く

5. 4. 8. 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。

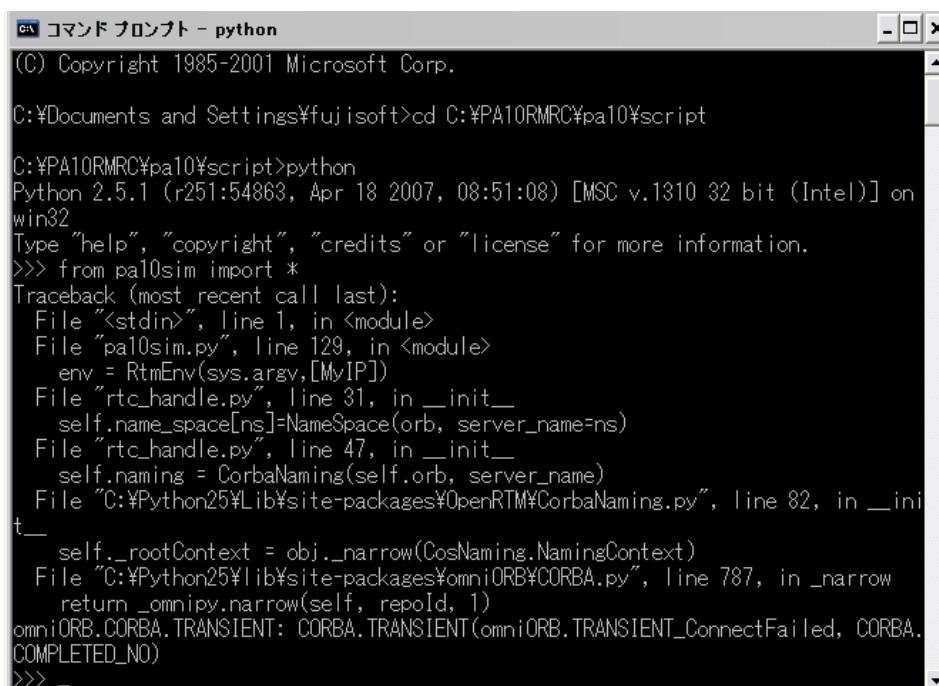
1. 非常停止ボタンを押す
2. 各 RTC を終了させる
3. コントローラの電源を切る
4. 非常停止ボタンを回し、ブレーキを解除する
5. コントローラ電源が落ちてから 5 秒間経ったことを確認し、再びコントローラの電源を入れる
6. モジュールを起動しなおす

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を確認する (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すると下記のエラーがでる。



```

(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\fujisoft>cd C:\PA10RMRC\pa10\script

C:\PA10RMRC\pa10\script>python
Python 2.5.1 (r251:54863, Apr 18 2007, 08:51:08) [MSC v.1310 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from pa10sim import *
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "pa10sim.py", line 129, in <module>
    env = RtmEnv(sys.argv,[MyIP])
  File "rtc_handle.py", line 31, in __init__
    self.name_space[ns]=Namespace(orb, server_name=ns)
  File "rtc_handle.py", line 47, in __init__
    self.naming = CorbaNaming(self.orb, server_name)
  File "C:\Python25\Lib\site-packages\OpenRTM\CorbaNaming.py", line 82, in __ini
t__
    self._rootContext = obj._narrow(CosNaming.NamingContext)
  File "C:\Python25\Lib\site-packages\omniORB\CORBA.py", line 787, in _narrow
    return _omnipy.narrow(self, repoId, 1)
omniORB.CORBA.TRANSIENT: CORBA.TRANSIENT(omniORB.TRANSIENT_ConnectFailed, CORBA.
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を確認する (4.5)。

- ・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 (4.1)。

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

- OpenRTM-aist-0.4.2 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 (3.1.2)。

- Python で書かれたサービスを実装する RTC がハンドルできない場合

[対策] 正しく OpenRTM-aist-Python-0.4.1-**RC1** をインストールしているかどうか確認する。OpenRTM-aist-Python-0.4.1-**RELEASE** をインストールしていると正常に動作しない場合がある。以下を参考にインストールされているバージョンを確認すること。

※ Windows の「コントロールパネル」「プログラムの追加と削除」

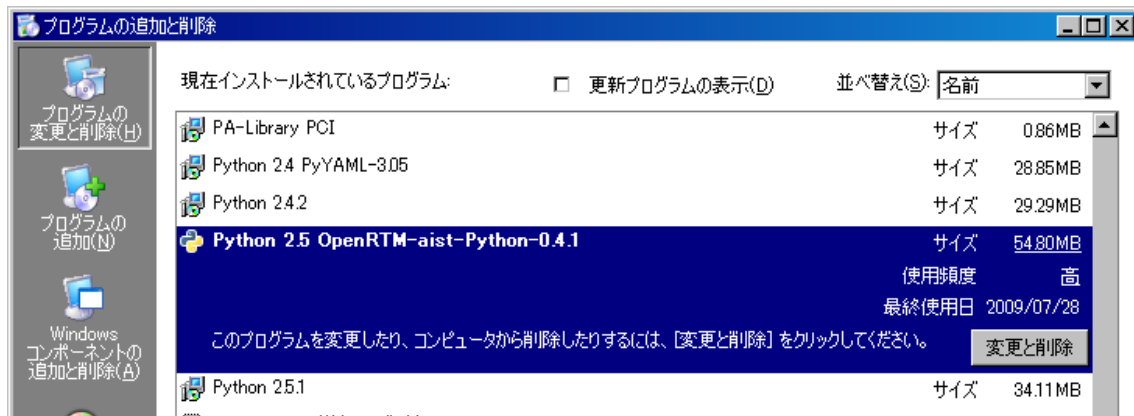


図 6-2 OpenRTM-aist-Python-0.4.1-RC1 の場合



図 6-3 OpenRTM-aist-Python-0.4.1-RELEASE の場合

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う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 (C++版) を使用してモジュールを動作させていた場合
[対策] 一旦 move コンポーネントを終了させ、再びその起動、ポート接続、活性化を行い新しい目標値を入力する。
- ・ 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] バッチファイル KillRTCall.bat を実行し全ての RTC 群を強制的に終了させ、再び RTC 群を起動しなおす。

「MSVCP71.dll が見つからなかったため、...」というエラーで終了する

[対策] 下記 URL より、WINDOWS¥system32 フォルダへダウンロードする。
<http://www.vector.co.jp/soft/win95/util/se435079.html>

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 ライブラリは、三菱重工業株式会社の製品であり、権利は三菱重工業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