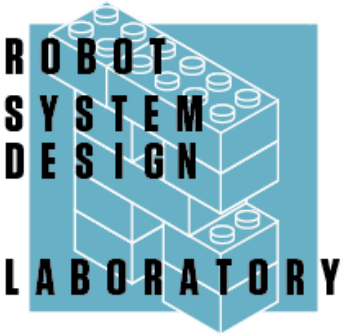
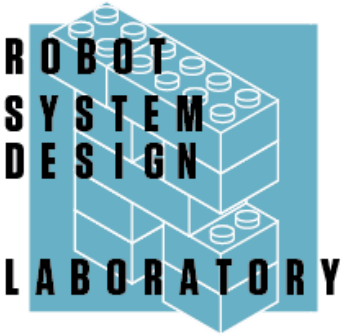


第3部
RTシステム構築演習

第3部の目的

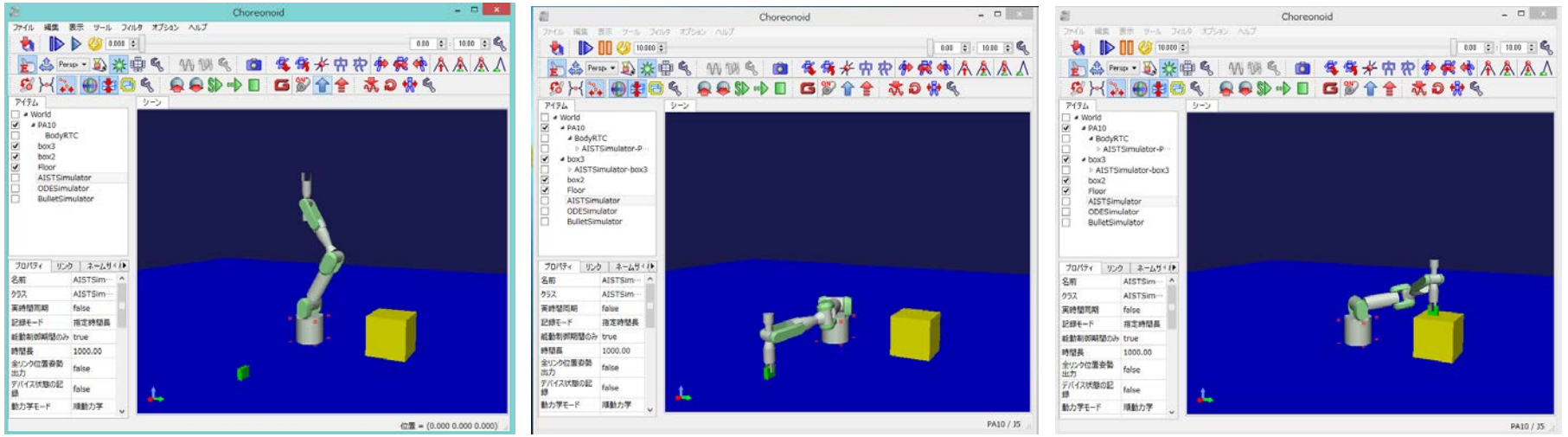
- ロボットアームを用いたRTC群の動作確認と，実際の開発を通じて，RTコンポーネントの開発プロセスを体験する.



ロボットアームRTC群を用いた RTミドルウェアのロボット応用体験

Choreonoidを用いたアームRTC群の動作体験

- ロボットシミュレータChoreonoidのPA10を操作するRTCの動作を体験する.
- ChoreonoidとPA10のモデル，制御用RTCについては，配布したものを利用
- 本システムでは，手先の位置と姿勢を入力してPA10で緑の箱を移動させる.



Choreonoidとは？

- 産業技術総合研究所で開発されているロボット用統合GUIソフトウェア.
- 動力学シミュレーション，動作振り付け機能を標準装備.
- 独自の機能もプラグインとして追加可能.

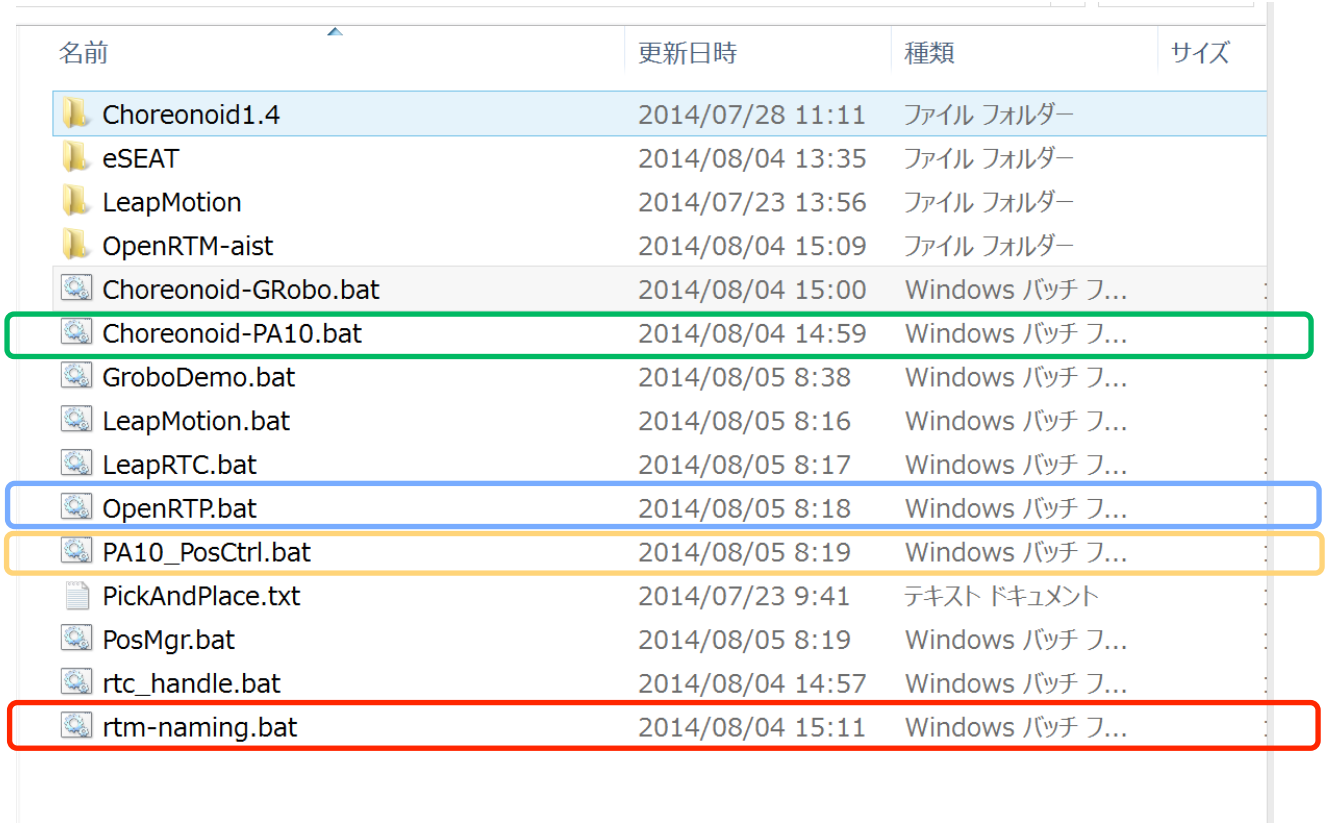
Choreonoid HPより引用

Choreonoid HP

http://choreonoid.org/ja/

Choreonoidを用いたアームRTC群の動作体験

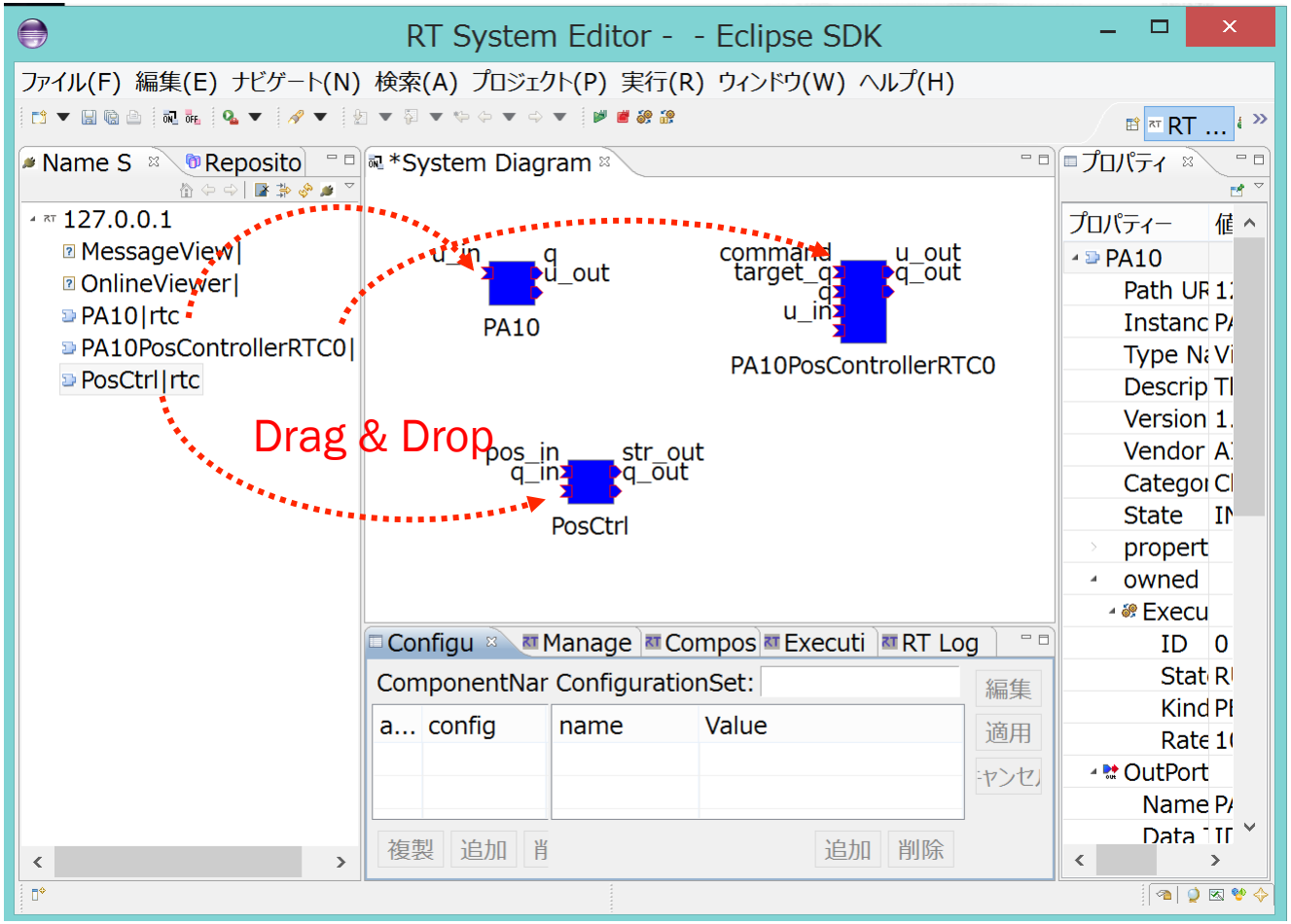
- NameServerの起動：　rtm-naming.bat
- RT SystemEditorの起動：OpenRTP.bat
- Choreonoidの起動：Chorenoid-PA10.bat
- PA10のコントローラの起動：　PA10_PosCtrl.bat



名前	更新日時	種類	サイズ
Choreonoid1.4	2014/07/28 11:11	ファイル フォルダー	
eSEAT	2014/08/04 13:35	ファイル フォルダー	
LeapMotion	2014/07/23 13:56	ファイル フォルダー	
OpenRTM-aist	2014/08/04 15:09	ファイル フォルダー	
Choreonoid-GRobo.bat	2014/08/04 15:00	Windows バッチ フ...	
Choreonoid-PA10.bat	2014/08/04 14:59	Windows バッチ フ...	
GroboDemo.bat	2014/08/05 8:38	Windows バッチ フ...	
LeapMotion.bat	2014/08/05 8:16	Windows バッチ フ...	
LeapRTC.bat	2014/08/05 8:17	Windows バッチ フ...	
OpenRTP.bat	2014/08/05 8:18	Windows バッチ フ...	
PA10_PosCtrl.bat	2014/08/05 8:19	Windows バッチ フ...	
PickAndPlace.txt	2014/07/23 9:41	テキスト ドキュメント	
PosMgr.bat	2014/08/05 8:19	Windows バッチ フ...	
rtc_handle.bat	2014/08/04 14:57	Windows バッチ フ...	
rtm-naming.bat	2014/08/04 15:11	Windows バッチ フ...	

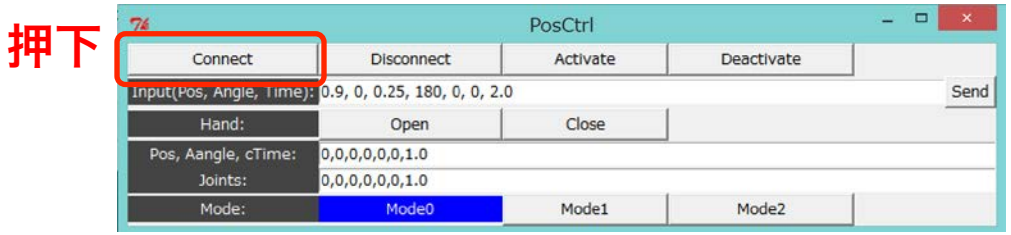
Choreonoidを用いたアームRTC群の動作体験

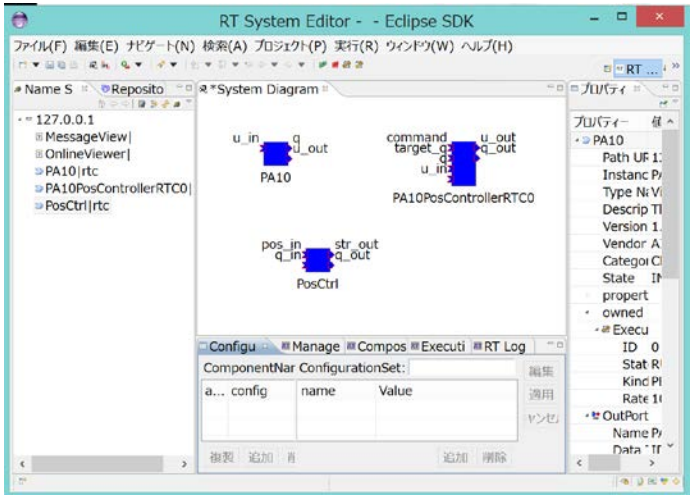
- RT System Editorを操作して，コンポーネントを表示する．



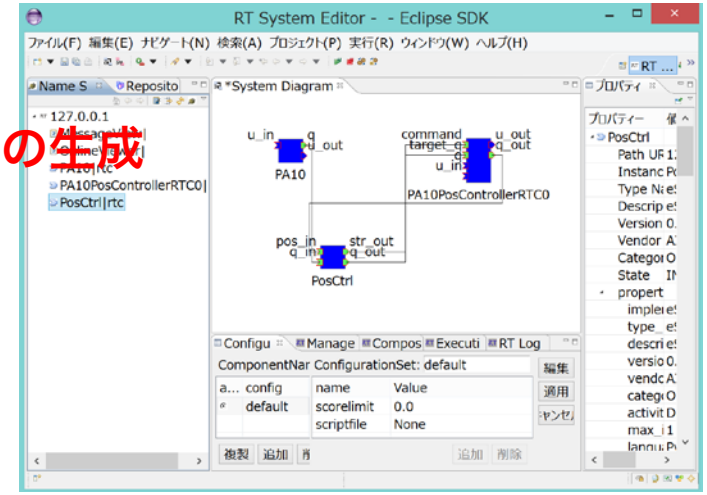
Choreonoidを用いたアームRTC群の動作体験

- PA10_PosCtrlの上部のボタンを操作して、コンポーネント間の接続を生成し、RT　System Editorで確認する



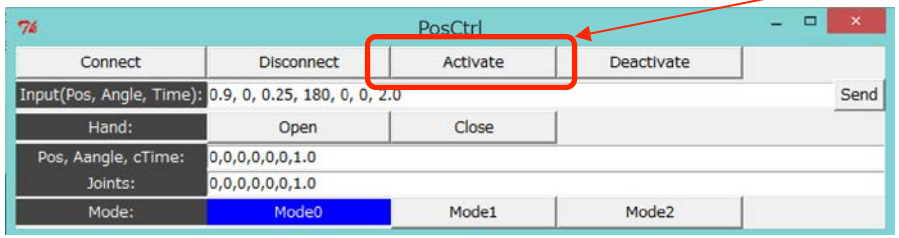


コネクションの生成

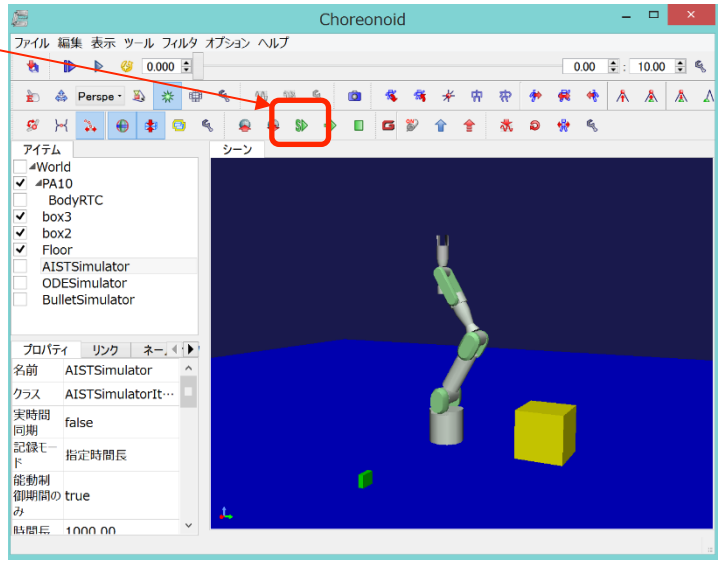


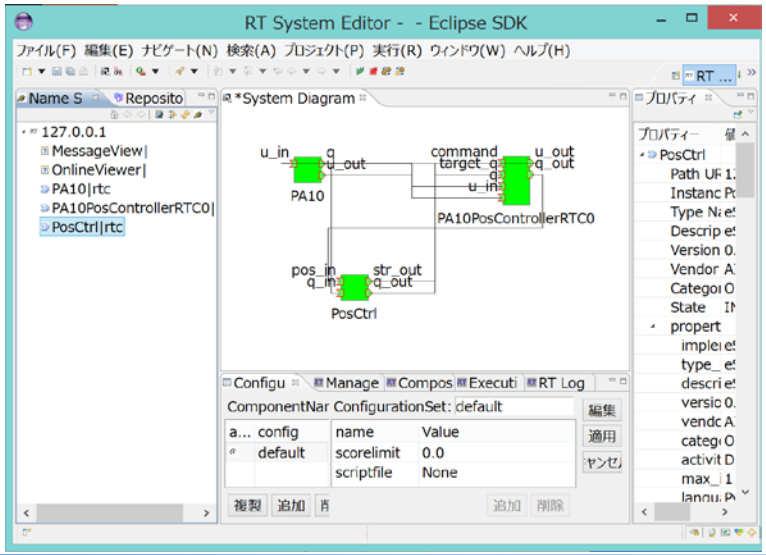
Choreonoidを用いたアームRTC群の動作体験

- PA10_PosCtrlの上部のボタンを操作して、コンポーネントを有効化し、Choreonoidのシミュレーションを開始する



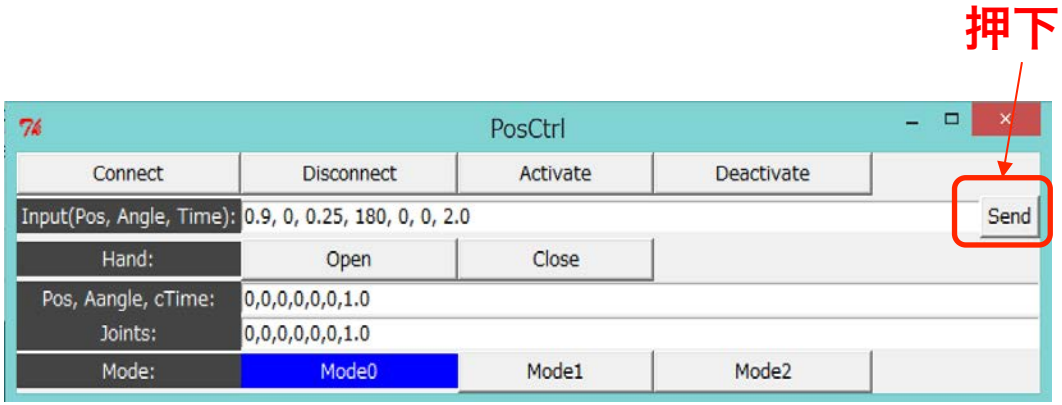
押下

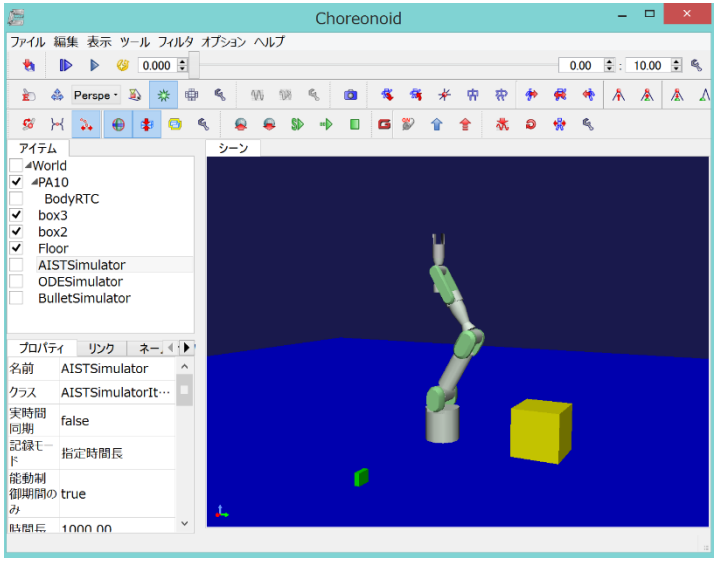




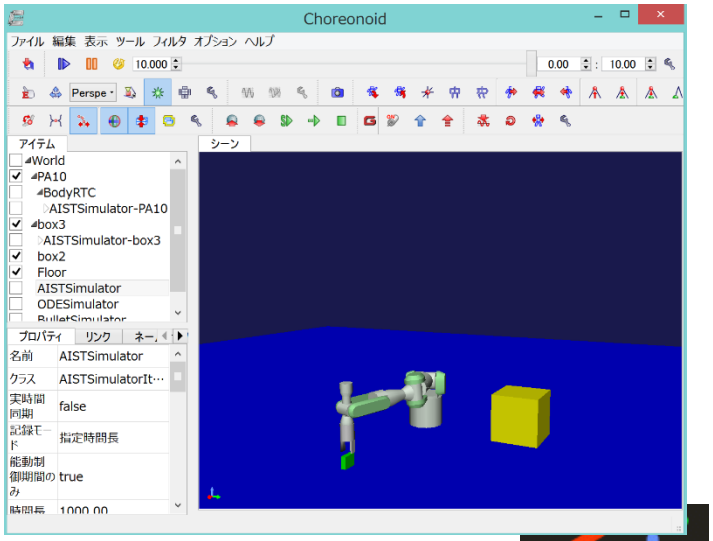
Choreonoidを用いたアームRTC群の動作体験

- PA10_PosCtrlを操作して、Choreonoid内のPA10を操作する.



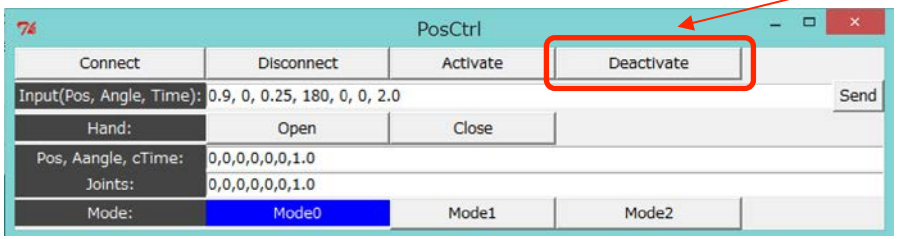


PA10の手先を
目標位置へ移動

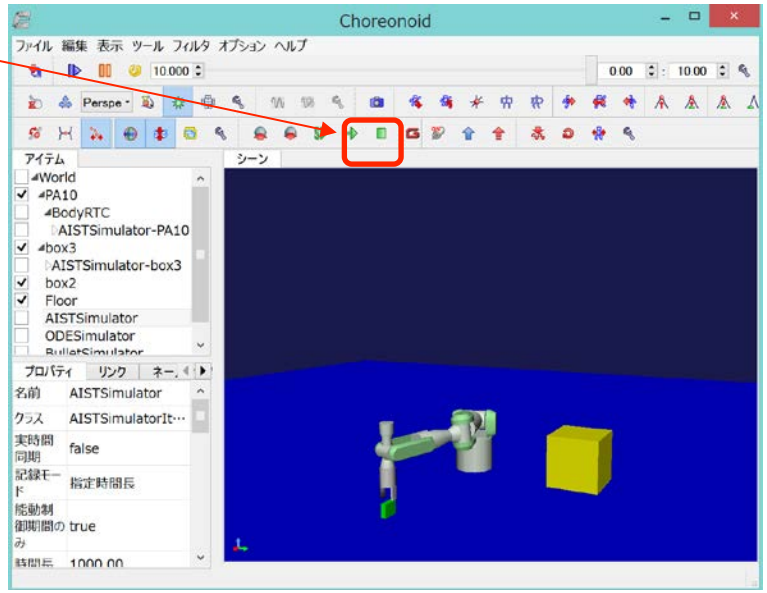


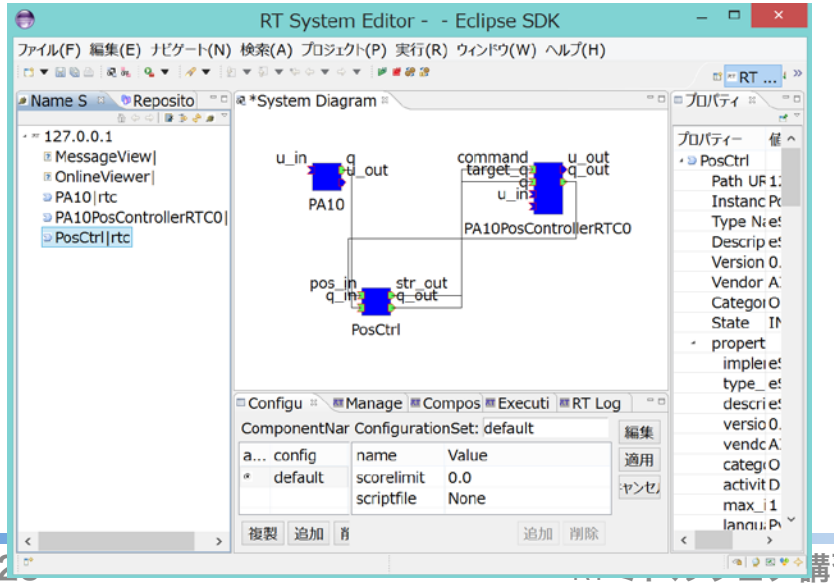
Choreonoidを用いたアームRTC群の動作体験

- PA10_PosCtrlの上部のボタンを操作して、コンポーネントを無効化し、Choreonoidのシミュレーションを終了する



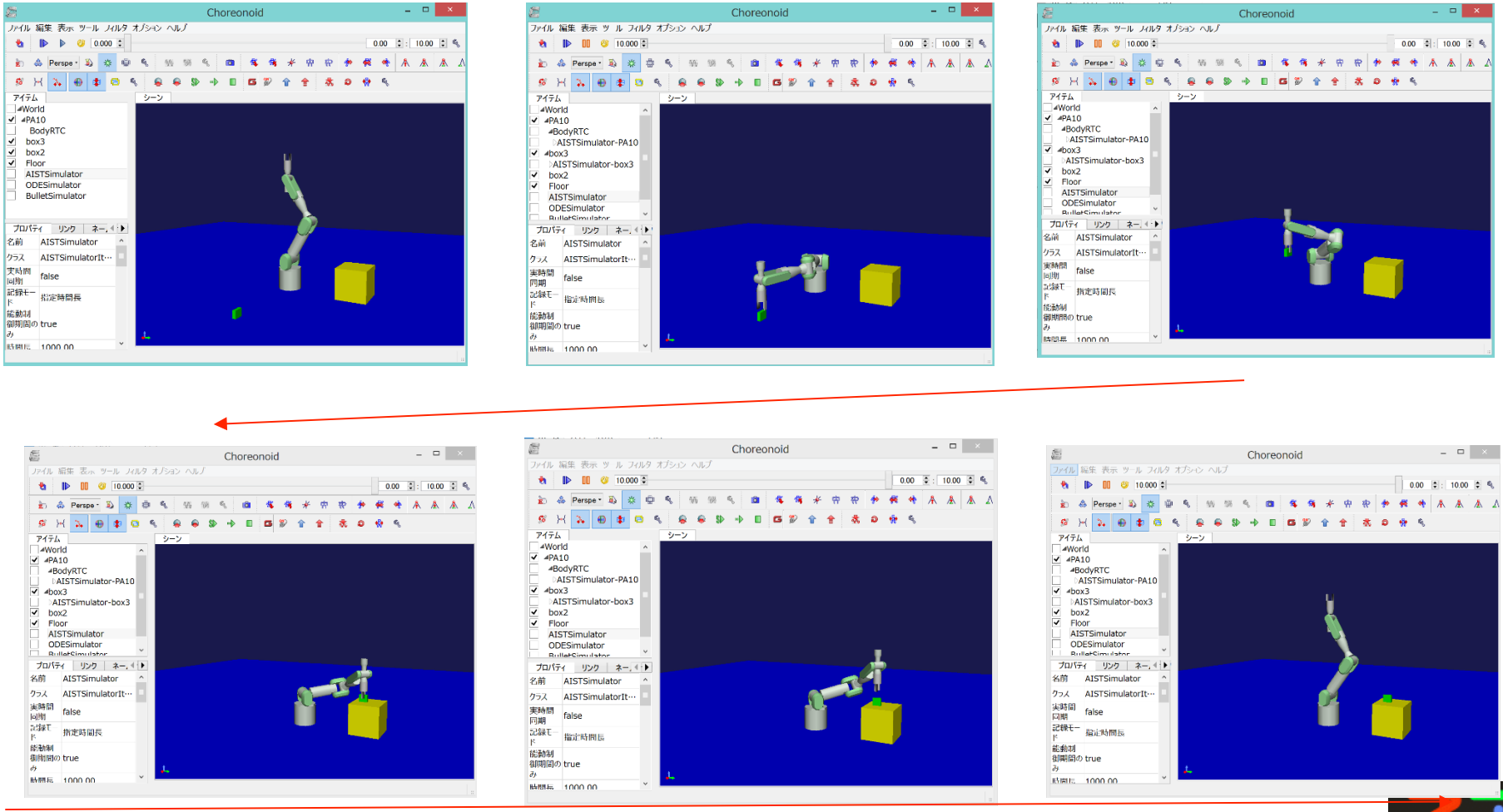
押下





RTコンポーネント開発演習

- PA10_PosCtrlと同じようにChoreonoid内のPA10を制御して，緑色の箱を移動させる



RTコンポーネント開発演習

- PA10_PosCtrlと同じようにChoreonoid内のPA10を制御して、緑色の箱を移動させるRTCをVC++で実装する。
- 各動作の目標位置は、 (X, Y, Z, roll, pitch, yaw)で与える

初期姿勢：0.0, 0.025, 1.2, 0.0, 0, 0, 2.0
アプローチ点：0.9, 0.0, 0.25, 180, 0, 0, 2.0
把持点：0.9, 0.0, 0.20, 180, 0, 0, 2.0
移動中間点：0.6, 0.0, 0.60, 180, 0, 0, 2.0
移動先アプローチ点：0.0, 0.65, 0.6, 180, 0, 90, 2.0
移動先：0.0, 0.65, 0.5, 180, 0, 90, 2.0
退避点：0.0, 0.65, 0.6, 180, 0, 90, 2.0
最終姿勢：0.0, 0.025, 1.2, 0.0, 0.0, 0.0, 2.0

- 把持点でハンドを閉じ、移送先でハンドを開けるようにする
- 各移動終了がわかるように、現在のロボットの位置姿勢を読み込むようにする.

RTコンポーネントの動作フロー

- RTCの生成時　→　onInitialize　を実行し待機状態に移行
- RTCの有効化　→　onActivated を実行し実行状態に移行

  （RT SystemEditorまたはrtshellなどから操作）
- RTCの実行時　→　onExecuteを周期実行
  - 実行周期は、rtc.confで指定可能
- RTCの無効化　→　onDeactivatedを実行し待機状態に移行

  （RT SystemEditorまたはrtshellなどから操作）

コアロジックの実装

0.　PA10の目標位置姿勢のデータ、ローカル変数を追記

1. PA10Sampleがアクティベートされたときに、初期姿勢になるように目標位置姿勢をPA10PosContollerRTCに送信

2. ロボットハンドが動作し始めると現在の位置姿勢が送信されてきますので、与えた目標位置姿勢と現在の位置姿勢の差分dを計算

3. dのノルムが十分に小さい場合、目標位置姿勢に到達したと考えられるので、次の目標位置姿勢をPA10PosControllerRTCに送信

4. 以降2,3を繰り返す

コアロジックの実装

0. PA10の目標位置姿勢のデータ、ローカル変数を追記

```
static const double path[]={
        0.0, 0.025, 1.2, 0, 0, 0, 2.0,
        0.9, 0.0, 0.25, 180, 0, 0,2.0,
        0.9, 0.0, 0.20, 180, 0, 0, 2.0,
        0.6, 0.0, 0.60, 180, 0, 0, 2.0,
        0.0, 0.65, 0.6, 180, 0, 90, 2.0,
        0.0, 0.65, 0.5, 180, 0, 90, 2.0,
        0.0, 0.65, 0.6, 180, 0, 90, 2.0,
        0.0, 0.025, 1.2, 0.0, 0.0, 0.0, 2.0
};
```

X=0.9m, Y=0.0m, Z=0.25m
r=180°, p=0°, y=0°
Time（移動時間） =2.0秒

PA10Sample.cpp

```
int nPath, count;
double *targetPos;
```

目標位置姿勢のインデックス

目標位置姿勢の状態のカウント

目標位置姿勢

PA10Sample.h

コアロジックの実装

0. PA10の目標位置姿勢のデータ、ローカル変数を追記

```
PA10Sample::PA10Sample(RTC::Manager* manager)
    // <rtc-template block="initializer">
  : RTC::DataFlowComponentBase(manager),
    m_Pos_inIn("Pos_in", m_Pos_in),
    m_CommandOut("Command", m_Command),
    m_Pos_outOut("Pos_out", m_Pos_out)

    // </rtc-template>
{
  nPath = 0;
  count = 0;
  targetPos = new double[7];
}
```

入出力ポートの内部変数の初期化

追記した内部変数の初期化

コアロジックの実装

1. PA10Sampleがアクティベートされたときに、初期姿勢になるように目標位置姿勢をPA10PosContollerRTCに送信

```
RTC::ReturnCode_t PA10Sample::onActivated(RTC::UniqueId ec_id)
{
  nPath=count=0;
  m_Pos_out.data.length(7);
```

出力データポートのためのバッファを確保

```
  for(int i=0; i<7;i++){
    m_Pos_out.data[i] = targetPos[i] = path[nPath*7+i];
  }
```

初期姿勢を出力データに代入

```
  m_Pos_outOut.write();
```

データを出力ポートへ書き込み。
送信は、自動的に行う。

```
  return RTC::RTC_OK;
}
```

コアロジックの実装

2. ロボットハンドが動作し始めると現在の位置姿勢が送信されてきますので、与えた目標位置姿勢と現在の位置姿勢の差分dを計算

```
RTC::ReturnCode_t PA10Sample::onExecute(RTC::UniqueId ec_id)
{
  double diffVal=0;

  if(m_Pos_inIn.isNew()){
    m_Pos_inIn.read();

  for(int i=0; i < 6;i++){
    diffVal += fabs(targetPos[i] - m_Pos_in.data[i]);
  }

 • • •
  }
 return RTC::RTC_OK;
}
```

入力データポートをチェック

入力データポートからデータを読み込み

与えた目標位置姿勢と現在の位置姿勢の差分を計算

コアロジックの実装

3. dのノルムが十分に小さい場合、目標位置姿勢に到達したと考えられるので、次の目標位置姿勢をPA10PosControllerRTCに送信

```
RTC::ReturnCode_t PA10Sample::onExecute(RTC::UniqueId ec_id)
{

 ・・・
   if (diffVal < 0.0001){ count++; }
  else { count = 0; }

    if (count > 0 && nPath < 8){
     nPath++;
     for(int i=0; i<7;i++){
       m_Pos_out.data[i] = targetPos[i] = path[nPath*7+i];
     }
     m_Pos_outOut.write();
    }
  }
  return RTC::RTC_OK;
}
```

与えた目標位置姿勢と現在の位置姿勢の差分が十分小さいことを検証

次の目標位置姿勢を出力

コアロジックの実装

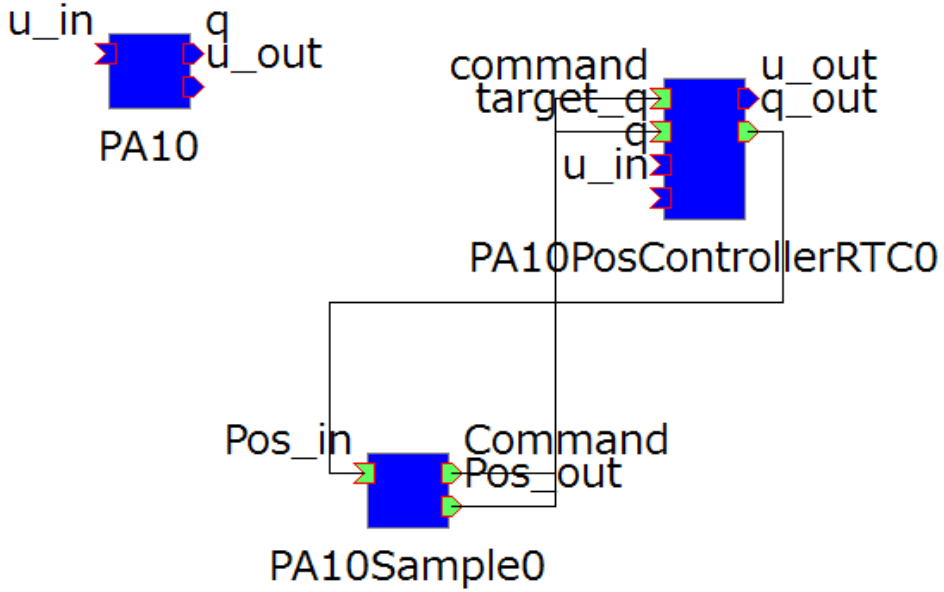
```
RTC::ReturnCode_t PA10Sample::onExecute(RTC::UniqueId ec_id)
{
  double diffVal=0;

  if(m_Pos_inIn.isNew()) {
        m_Pos_inIn.read();
        for(int i=0; i < 6;i++){    diffVal += fabs(targetPos[i] - m_Pos_in.data[i]);    }
        if (diffVal < 0.0001){ count++; }    else { count = 0; }
      if (count > 0 && nPath < 8){
          if (nPath == 2 && count < 1000){
              if (count == 1){
                  m_Command.data="Close";
                  m_CommandOut.write();
              }
          }else if(nPath == 5 &&  count < 1000){
              if (count == 1){
                  m_Command.data="Open";
          m_CommandOut.write();
              }
          }else{
                nPath++;
                for(int i=0; i<7;i++){    m_Pos_out.data[i] = targetPos[i] = path[nPath*7+i];    }
                m_Pos_outOut.write();
          }
        }
 }
 return RTC::RTC_OK;
}
```

ハンドを閉じる命令を送信し、1000ループ待つ

ハンドを開く命令を送信し、1000ループ待つ

コンパイルと実行

- VC++2010のソリューションを開きビルド、エラーがなければ実行
- Choreonoidを起動　(Choreonoid-PA10.bat)
- RT SystemEditorで下記のように接続



- Choreonoidのシミュレーションを開始し、PA10Sample0をRT SystemEditorで実行状態にする
- PA10の動作を確認

まとめ

- ロボットアーム動作RTCの動作検証，実装を通じて，RTC開発の基礎を体験．

RTミドルウェアを体験用モジュールセットのご紹介

下記のサイトから，RTCの体験モジュールセットがダウンロード可能．

https://github.com/tork-a/openrtm_tutorial

上記のものをUSBメモリにコピーしていただけば，手軽にRTCで作られたシステムの体験ができる．
（ただし，Windows+IE環境のみで動作．）

NEDOプロジェクトを核とした人材育成、産学連携等の総合的展開
RTミドルウェアの実践的展開

RTミドルウェアについて，開発方法など知りたい．
RTミドルウェアの導入を検討したいが自社での立ち上げは困難．

ご相談いただければ，講習会など開催いたします．