動作モニタコンポーネント マニュアル

1 最初に

本稿ではロボット動作モニタコンポ-ネントの運用方法について解説します.
単にロボット動作モニタコンポーネントの動作確認を行う場合は同梱の”GettingStart.pdf”を御覧下さい.

2 ロボットのモデルの構築方法

2.1 概要

動作モニタコンポーネント (以下，単に本コンポーネントと呼びます) はロボットのモデルを表示するものです．よって，ユーザは本コンポーネントに対して表示するロボットの構造 (関節がどこにあって，リンクの長さがどれだけで等) を伝えてあげる必要があります.

本コンポーネントは, それらの情報を伝える手段として外部の設定ファイルと呼ぶものを利用しています. 本コンポーネントを起動する際に，コマンドライン引数に設定ファイルを指定して，そのファイルに書かれた内容を本コンポーネントが解釈することで，ロボットの表示を行います.

2.2 データ構造の概念

ロボットについて学んだ方は図 1 になじみがあることでしょう.

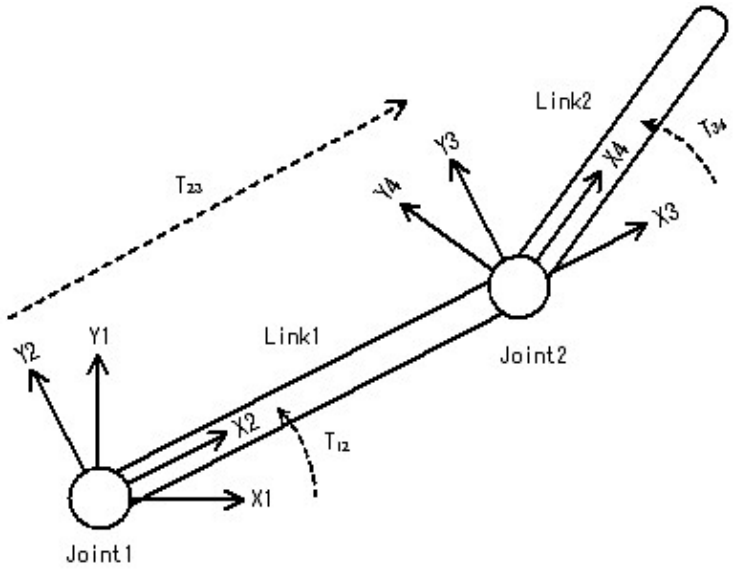

図 1　2 関節マニピュレータの関節の構造と座標系の配置

ロボットの構造を表現する方法として，要所要所に座標系を張り，各々の座標系間の関係を平行移動，回

転移動の同次変換行列の積で表すというのはポピュラかと思います．さて，ここで注意して頂きたい点は，Link2 は Link1 の先に位置しているということです．木構造の用語を用いると，Link1 の子ノードに Link2 が繋がっていると言えます．更に Link1 の親ノードは原点系であると言えます．要するに，シリアルリンク系のロボットの構造は木構造を用いることで表現することが出来ます (図 2)．本コンポーネントはこの性質を利用して複数のパーツからなるロボットを表現しています．

図 2　座標系の主従関係

木構造を構成する要素であるノードの構造は図 3 様になっています．

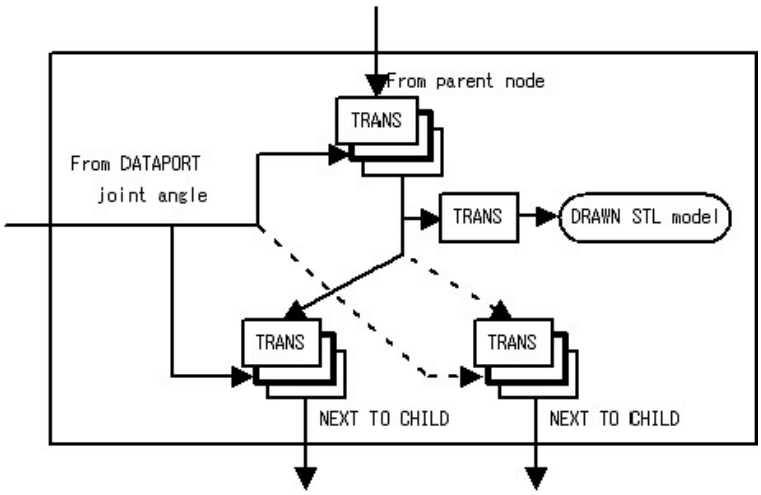

図 3　ノードの構造

ノードは親ノードと接続する部分を一つと，子ノードへ繋がる部分を N 個備えています．そして，ノード内部には座標変換を登録する部分が要所に設けられています．そして座標変換の要素は DATAPORT から取得したデータとリンクさせることが可能となっています．また，ノード内部には描画するモデルデータを登録する部分が設けられています．

3　設定ファイルの書き方

このコンポーネントを実際に使用するには, 設定ファイルの書き方と描画するモデルデータである STL ファイルの出力方法について理解していただく必要があります.

両者について一度に説明するのは難しいので, まずは設定ファイルの書き方について例を踏まえて説明します.

3.1　目標

図 4 の様なマニピュレータのモデルを動作モニタコンポーネントで表示するための設定ファイルの書き方について理解していただくのがここでの目標です. このマニピュレータは三つのパーツから構成されています. そして関節数は 2 つです.

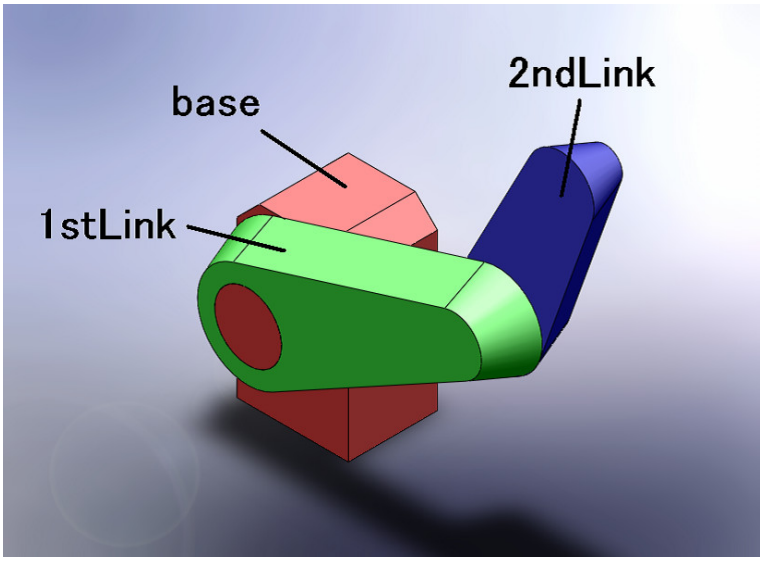

図 4　例に用いるマニピュレータのモデル

説明に用いる三つのモデルデータはこちらで用意しました. モデルデータの寸法は図 5 から 7 のようになっています. 図中の赤字で書かれた XY 座標軸はモデルデータの原点を表しています.

3.2　step1 モデルを一つ表示させてみる

まず, モデルデータを収めた step_stl という名前のディレクトリが RobotMotionMonitor_package/RMM/components の中にあることを確認して下さい.

次に, モニタコンポーネント本体があるディレクトリに”step1.conf”というファイルを作って下さい. そして, ファイルの中身を以下のように書いて下さい.

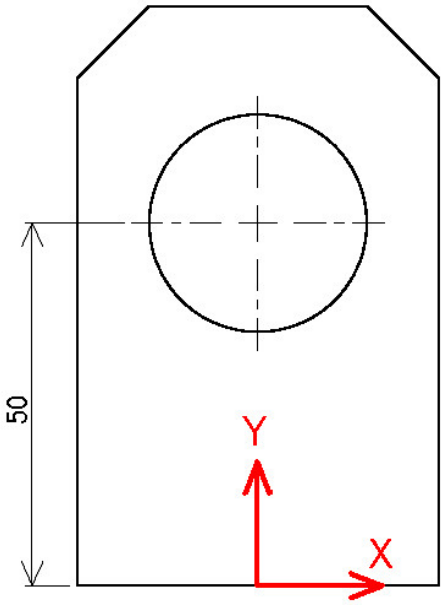

図 5　ベースの図面

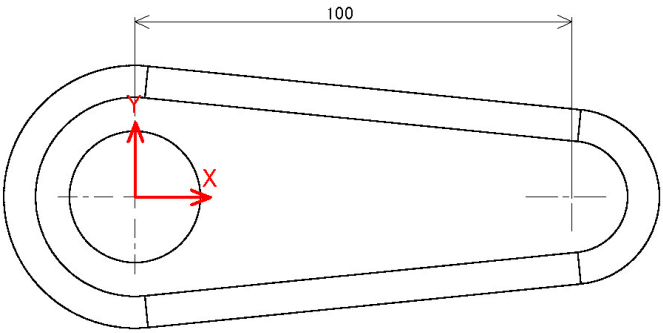

図 6　第一関節の図面

```
#レクチャー用コンフィグファイル
#file name step1.conf
#Step1

BEGIN PARTS = ROOT
    BEGIN CHILD = ToBase
        NEXT = Base
    END CHILD
END PARTS

BEGIN PARTS = Base
    BEGIN DRAWN STL = Base_STL
        COLOR=1.0,0.4,0.4
        PATH = step_stl/base.STL
    END DRAWN STL
END PARTS
```

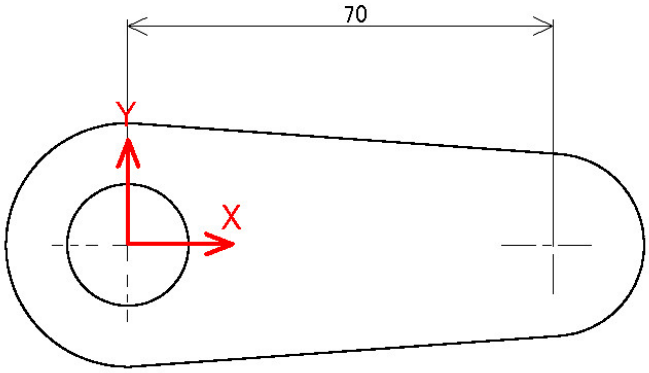

図 7　第二関節の図面

そして,DOS 窓から RobotMotionMonitorComp.exe step1.conf とタイプしてコンポーネントを起動して下さい. すると, 図 8 のように表示されます.

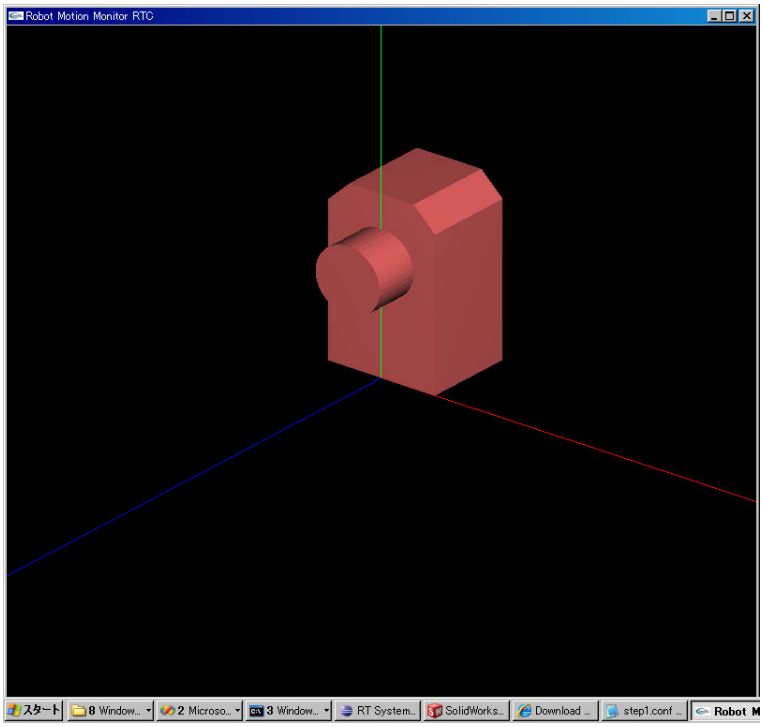

図 8　step1 の起動結果

画面中に赤, 緑, 青の座標軸が表示されていますが, これは描画ウインドウ中のワールド座標系を表しています. 赤, 緑, 青はそれぞれ X 軸,Y 軸,Z 軸に対応しています.

ワールド座標系の原点に base のモデルが表示されていることがわかると思います.

ここで設定ファイルの中身について説明します.

```
#レクチャー用コンフィグファイル　　#が頭にあるとコメント行になります
#Step1

#ここから
BEGIN PARTS = ROOT
    BEGIN CHILD = ToBase
        NEXT = Base
    END CHILD
END PARTS
#ここまでは必ず必要な記述です.
#NEXT=Base となっていることに注意して下さい.


#表示するパーツの情報は BEGIN PARTS から END PARTS の間に記述します. パーツの識別名を=の後に書きます.
#この場合は識別名が Base という PARTS についての設定を行うという意味になります.
BEGIN PARTS = Base

#描画するモデルデータ自体の設定は BEGIN DRAWN STL から END DRAWN STL の間に記述します.
#ここについても識別名をつける必要があります.
    BEGIN DRAWN STL = Base_STL

#描画するモデルの色の指定は RGB により行います.
        COLOR=1.0,0.4,0.4

#描画するモデルデータへのパスはこの様に指定します. モニターコンポーネントがあるディレクトリを起点とし,
相対パスでファイルを指定します.
#ここでは step_stl というディレクトリの中にある base.stl をロードせよという記述を行っています.
        PATH = step_stl/base.STL
    END DRAWN STL
END PARTS
#BEGIN で始めたらそれに対応する END を書く必要があります. 書かないと文法エラーとなります.
```

試しに,COLOR=1.0,0.4,0.4 と書かれている部分を COLOR=1.0,1.0,1.0 と書き換え, 更に,$PATH = step\_stl/base.STL$ と書かれている部分を $PATH = step\_stl/1stLink.STL$ と書き換え, 保存した後, 再度モニターコンポーネントを起動してみて下さい. すると図 9 のように表示されるはずです.

表示されたモデルの色と形状が変化しているので, 設定ファイルを書き換えた内容が反映されたことが分かります.

3.3 Step2 複数のパーツを表示するには

マニピュレータは複数のパーツにより構成されます. ここではモニタコンポーネントで複数のパーツを表示するための方法について説明します.

step2.conf という名前のファイルを作り, 中身を以下のように書いて下さい.

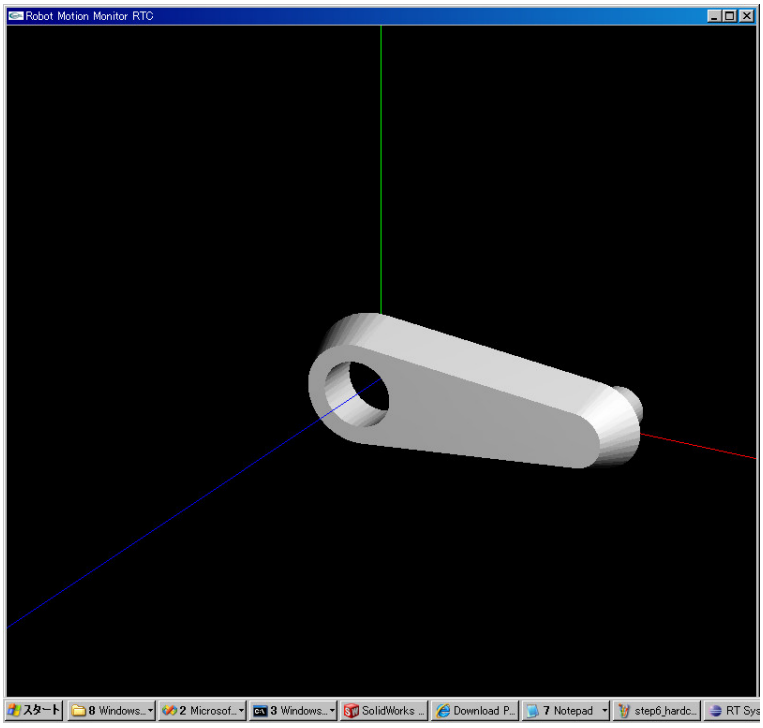

図 9　色を変更した場合の結果

```
#レクチャー用コンフィグファイル
#Step2

BEGIN PARTS = ROOT
    BEGIN CHILD = ToBase
        NEXT = Base
    END CHILD
END PARTS

BEGIN PARTS = Base
    BEGIN DRAWN STL = Base_STL
        COLOR=1.0,0.4,0.4
        PATH = step_stl/base.STL
    END DRAWN STL

    BEGIN CHILD = To1stLink
        NEXT = 1stLink
    END CHILD
END PARTS

BEGIN PARTS = 1stLink
    BEGIN DRAWN STL = 1stLink_STL
        COLOR=0.4,1.0,0.4
        PATH = step_stl/1stLink.STL
    END DRAWN STL
END PARTS
```

BEGIN PARTS と END PARTS のペアが先から一つ増えて三つになっていることに注意してください．加えて二つ目の BEGIN PARTS の中に

```
    BEGIN CHILD = To1stLink
        NEXT = 1stLink
    END CHILD
```

という部分が追加されているのがポイントです．

そして RobotMotionMonitorComp.exe step2.conf とタイプしてコンポーネントを起動して下さい．

すると図 10 のように表示されるはずです．

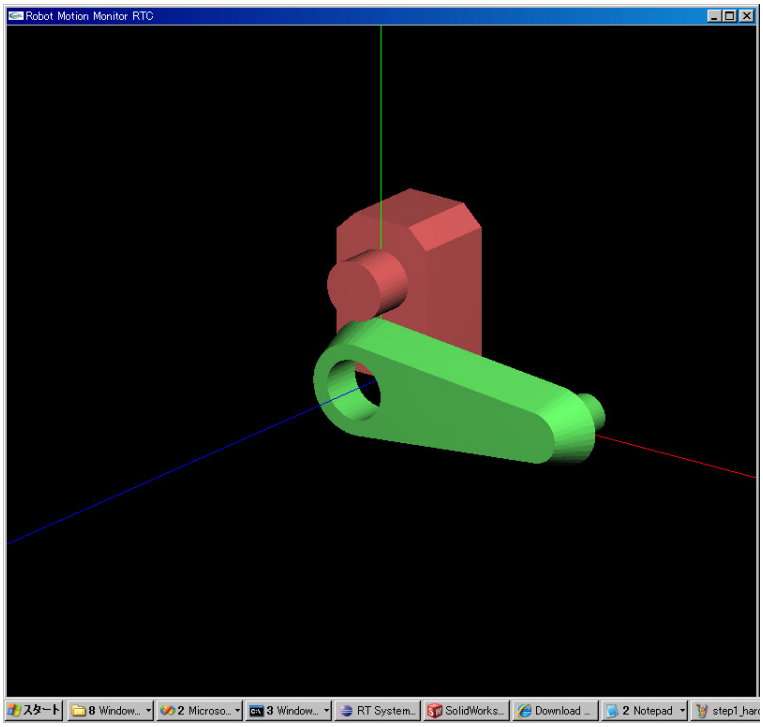

図 10　step.conf の実行結果

これは意図した表示ではないかと思います．base パーツの突起部分に 1stLink パーツの穴が一致するように表示したいものです．これの実現方法は次の Step3 で説明します．

3.4　Step3 パーツの相対位置の指定

二つのパーツの相対位置関係を指定するには座標変換を使います．step3.conf を同様に作成し，モニタコンポーネントを起動してください．

```
#レクチャー用コンフィグファイル
#Step3

BEGIN PARTS = ROOT
    BEGIN CHILD = ToBase
        NEXT = Base
    END CHILD
END PARTS

BEGIN PARTS = Base
    BEGIN DRAWN STL = Base_STL
        COLOR=1.0,0.4,0.4
        PATH = step_stl/base.STL
    END DRAWN STL

    BEGIN CHILD = To1stLink
        NEXT = 1stLink
    END CHILD
END PARTS

BEGIN PARTS = 1stLink
    TRANS = TRANSLATION(0,50,0)
    BEGIN DRAWN STL = 1stLink_STL
        COLOR=0.4,1.0,0.4
        PATH = step_stl/1stLink.STL
    END DRAWN STL
END PARTS
```

すると図 11 のように表示されるはずです.

穴にぴったりとはまった状態で表示することができました. このように表示できた理由は設定ファイル中に”TRANS=TRANSLATION(0,50,0)”という部分を追加したからです.

```
BEGIN PARTS = 1stLink
#ここに追加した TRANS=がポイントです
    TRANS = TRANSLATION(0,50,0)
    BEGIN DRAWN STL = 1stLink_STL
        COLOR=0.4,1.0,0.4
        PATH = step_stl/1stLink.STL
    END DRAWN STL
END PARTS
```

平行移動の座標変換を指定するには TRANS=TRANSLATION(x,y,z) と書きます. 今回は TRANSLATION(0,50,0) となっているので,Y 軸方向にのみ 50 移動させるということになります.

この 50 という数字は base パーツの 2 次元図面 (5) を再度見て頂ければわかるかと思います. base パーツは, 次のパーツがつながる部分までの距離が Y 軸方向に 50 となっています. そのため,TRANSLATION(0,50,0) と書くことによってぴったり一致したわけです.

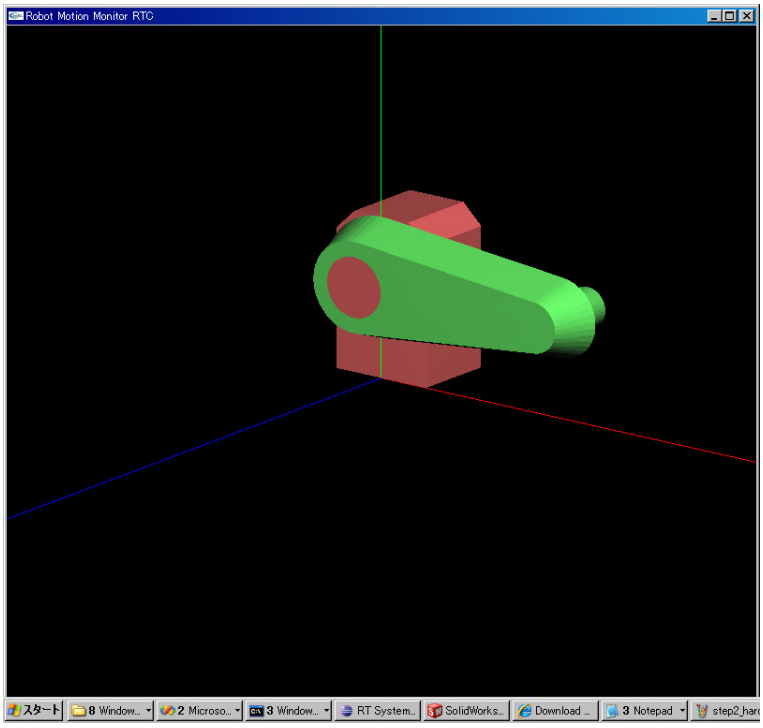

図 11　step3.conf を実行した結果

3.5　Step4 パーツを回転させる方法

step4.conf を作成し, これを引数に与えて実行してみて下さい.

```
#レクチャー用コンフィグファイル
#Step4

BEGIN PARTS = ROOT
    BEGIN CHILD = ToBase
        NEXT = Base
    END CHILD
END PARTS

BEGIN PARTS = Base
    BEGIN DRAWN STL = Base_STL
        COLOR=1.0,0.4,0.4
        PATH = step_stl/base.STL
    END DRAWN STL

    BEGIN CHILD = To1stLink
        NEXT = 1stLink
    END CHILD
END PARTS

BEGIN PARTS = 1stLink
    TRANS = TRANSLATION(0,50,0)
    TRANS = ROTATION(Z,30)
    BEGIN DRAWN STL = 1stLink_STL
        COLOR=0.4,1.0,0.4
        PATH = step_stl/1stLink.STL
    END DRAWN STL
END PARTS
```

すると図 12 の様に表示されるはずです.

先程は水平だった 1stLink が傾いていることに注意して下さい. 傾き, つまり回転の座標変換を追加するには以下の様に書きます.

```
BEGIN PARTS = 1stLink
    TRANS = TRANSLATION(0,50,0)
#ここの行がミソです.
    TRANS = ROTATION(Z,30)
    BEGIN DRAWN STL = 1stLink_STL
        COLOR=0.4,1.0,0.4
        PATH = step_stl/1stLink.STL
    END DRAWN STL
END PARTS
```

TRANS=ROTATION(axis,degree) と書くことで axis 周りに degree だけ回転させるという指定が可能です.axis には X,Y,Z を指定することが出来ます. 今回の場合は TRANS = ROTATION(Z,30) と書いたので Z 軸回りに 30 度だけ回転させるということになります.

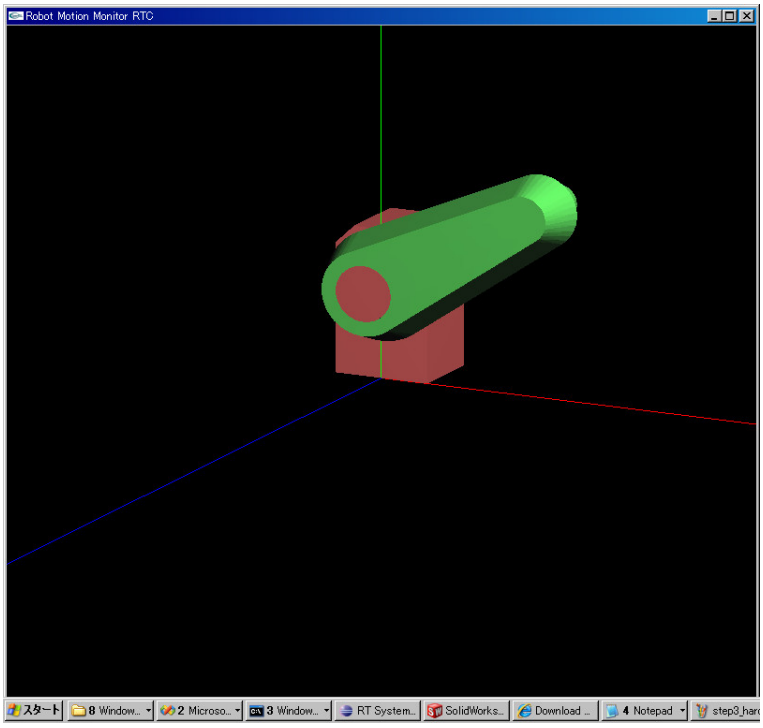

図 12　step4.conf の実行結果

3.6　Step5 更にパーツを追加するには

これまでの Step を踏まえて,2ndLink を追加してみましょう.

```
#レクチャー用コンフィグファイル
#Step5

BEGIN PARTS = ROOT
    BEGIN CHILD = ToBase
        NEXT = Base
    END CHILD
END PARTS

BEGIN PARTS = Base
    BEGIN DRAWN STL = Base_STL
        COLOR=1.0,0.4,0.4
        PATH = step_stl/base.STL
    END DRAWN STL

    BEGIN CHILD = To1stLink
        NEXT = 1stLink
    END CHILD
END PARTS

BEGIN PARTS = 1stLink
    TRANS = TRANSLATION(0,50,0)
    TRANS = ROTATION(Z,30)
    BEGIN DRAWN STL = 1stLink_STL
        COLOR=0.4,1.0,0.4
        PATH = step_stl/1stLink.STL
    END DRAWN STL

    BEGIN CHILD = To2ndLink
        NEXT = 2ndLink
    END CHILD
END PARTS

BEGIN PARTS = 2ndLink
    TRANS = TRANSLATION(100,0,0)
    TRANS = ROTATION(Z,30)
    BEGIN DRAWN STL = 1stLink_STL
        COLOR=0.4,0.4,1.0
        PATH = step_stl/2ndLink.STL
    END DRAWN STL
END PARTS
```

起動すると図 13 のように表示されるはずです.
設定ファイルの書き方には慣れてきたでしょうか?

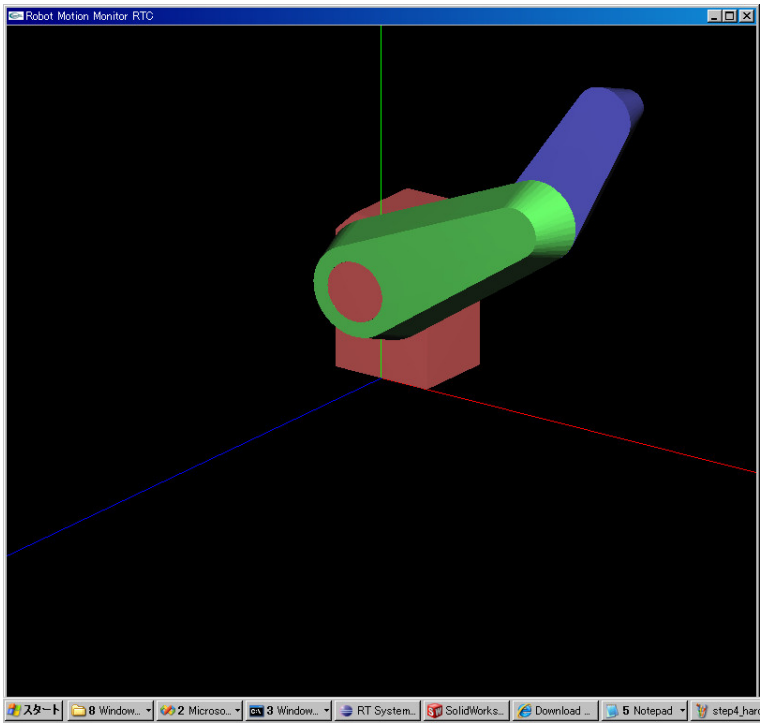

図 13　step5.conf の実行結果

3.7　Step6 関節角度をデータポートから変更するには

モニタコンポーネントはデータポートから入力された情報を描画に反映させることができます．この機能によって, データポートから関節角度を受け取り, その通りにモデルを動かすということが実現出来ます．

”step6.conf”を作成し, 以下のように記述して下さい．

```
#レクチャー用コンフィグファイル
#Step6

BEGIN PARTS = ROOT
    BEGIN CHILD = ToBase
        NEXT = Base
    END CHILD
END PARTS

BEGIN PARTS = Base
    BEGIN DRAWN STL = Base_STL
        COLOR=1.0,0.4,0.4
        PATH = step_stl/base.STL
    END DRAWN STL

    BEGIN CHILD = To1stLink
        NEXT = 1stLink
    END CHILD
END PARTS

BEGIN PARTS = 1stLink
    TRANS = TRANSLATION(0,50,0)
    TRANS = ROTATION(Joint1,Z,30)
    BEGIN DRAWN STL = 1stLink_STL
        COLOR=0.4,1.0,0.4
        PATH = step_stl/1stLink.STL
    END DRAWN STL

    BEGIN CHILD = To2ndLink
        NEXT = 2ndLink
    END CHILD
END PARTS

BEGIN PARTS = 2ndLink
    TRANS = TRANSLATION(100,0,0)
    TRANS = ROTATION(Joint2,Z,30)
    BEGIN DRAWN STL = 1stLink_STL
        COLOR=0.4,0.4,1.0
        PATH = step_stl/2ndLink.STL
    END DRAWN STL
END PARTS

BEGIN DATAPORT
    1stLink.Joint1
    2ndLink.Joint2
END DATAPORT
```

step6.conf の下側に以下の内容が追加されているのがミソです.

```
BEGIN DATAPORT
    1stLink.Joint1
    2ndLink.Joint2
END DATAPORT
```

そして更に,TRANS=ROTATION() の部分が変更されていることにも注意して下さい.

```
BEGIN PARTS = 1stLink
    TRANS = TRANSLATION(0,50,0)
    TRANS = ROTATION(Joint1,Z,30)       #回転軸と角度の指定の前に Joint1 という文字列が追加されて
いる
    BEGIN DRAWN STL = 1stLink_STL
            :
            :
            :
BEGIN PARTS = 2ndLink
    TRANS = TRANSLATION(100,0,0)
    TRANS = ROTATION(Joint2,Z,30)       #同様に Joint2 が追加されている
    BEGIN DRAWN STL = 1stLink_STL
            :
            :
            :
```

今回はモニタコンポーネントに加えて ArrayOutput というコンポーネントを一緒に使います.

ArrayOutput コンポーネントは引数を与えずに起動すると, コンソールから入力したデータをデータポートから出力するという動作をします. モニタコンポーネントも起動して SystemEditor で図 14 のように配置して下さい.

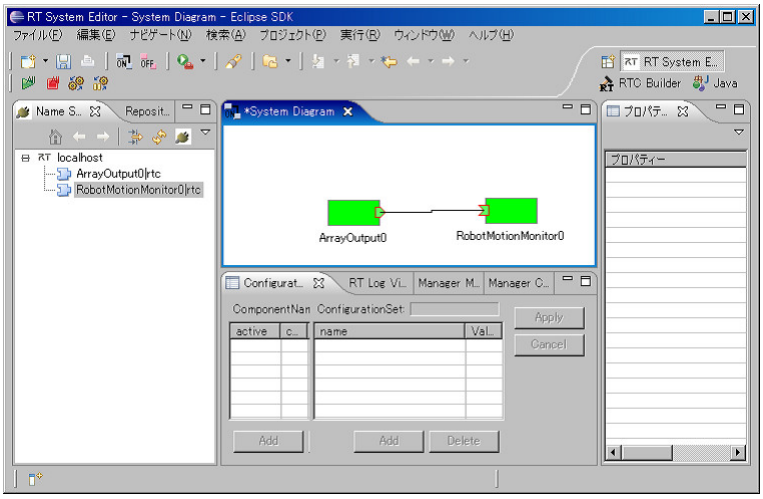

図 14　二つのコンポーネントの接続

両コンポーネントを Activate した後に,ArrayOutput コンポーネントを起動したプロンプトで”0,0”と入力して下さい. するとモニタコンポーネント上のモデルがまっすぐな姿勢になるはずです (図 15). 更に, 色々な角度を ArrayOutput コンポーネントに入力してみて下さい (図 16,17).

前後しますが,BEGIN DATAPORT と END DATAPORT の間に書いた座標変換の対象がデータポートと

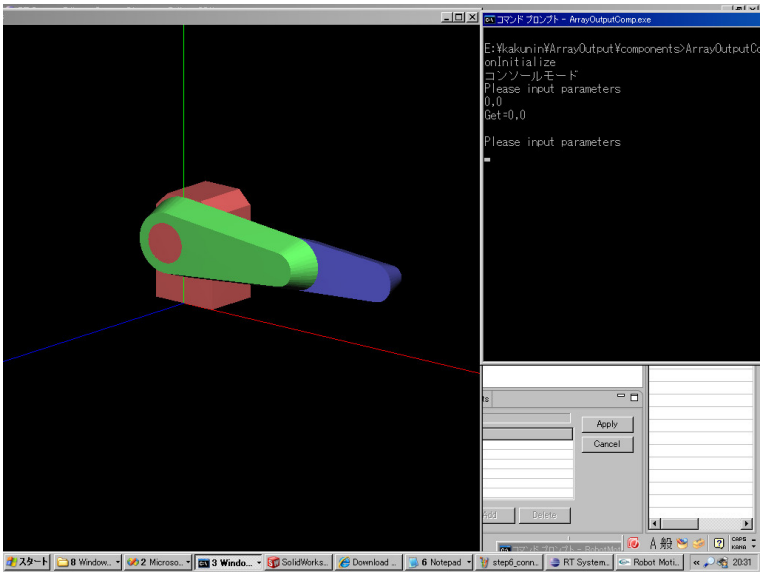

図 15　0,0 と入力した場合の姿勢

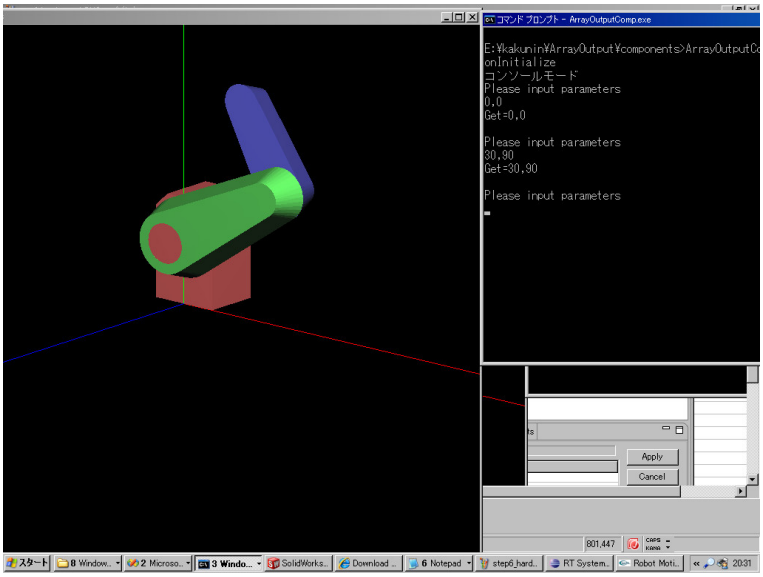

図 16　30,90 と入力した場合の姿勢

リンクされることがわかるかと思います．座標変換の対象の指定は BEGIN PARTS=で指定した識別名の後に (ドット) を付け, その後に座標変換の識別名を書くという方法で行います．

C 言語での構造体のメンバアクセスと同じ要領です．

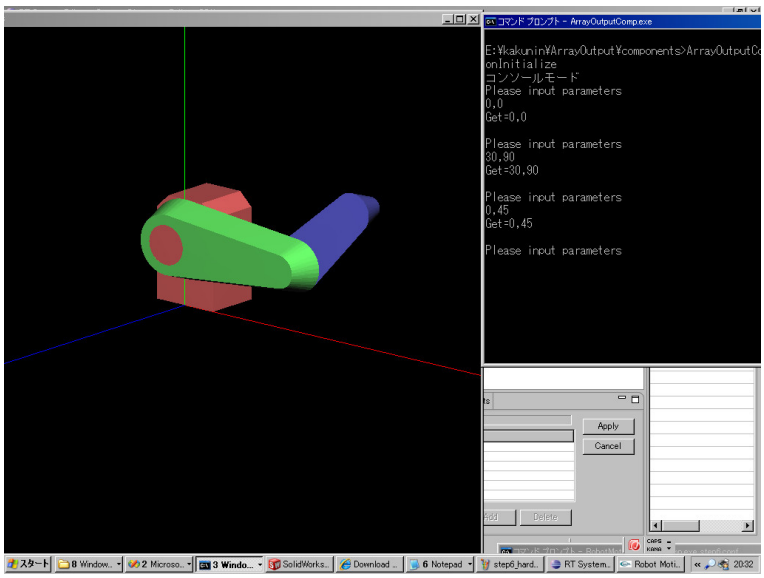

図 17　0,45 と入力した場合

4　STL ファイルの出力方法

Solidworks の場合について説明します．図 18 のモデルを STL ファイル形式に変換するための方法を順を追って説明します．

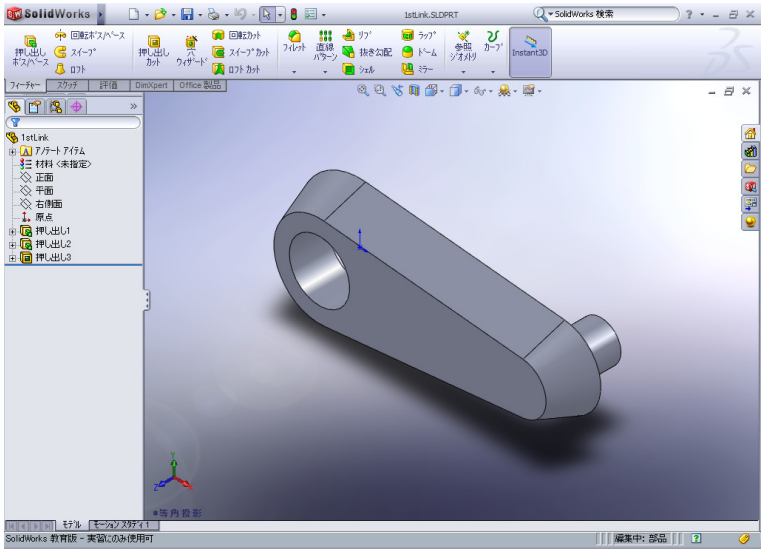

図 18　Solidworks で作成したモデル

まずは, メニューバーのファイル→指定保存...(s) をクリックして, ファイル保存画面 (図 19) を出して下さい．ファイルの種類のプルダウンメニューから STL を選択して下さい．

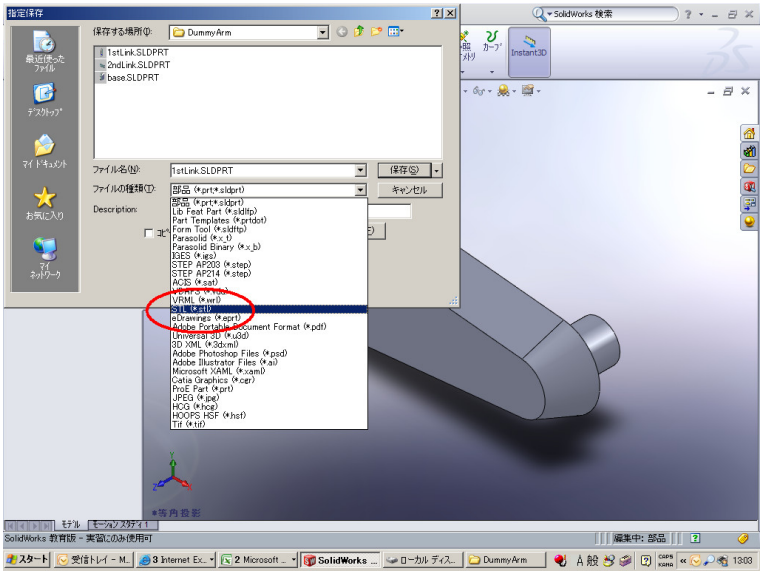

図 19　ファイル保存画面

STL 形式で保存する画面になると, 画面右下にオプションというボタンが現れるので, クリックして下さい (図 20).

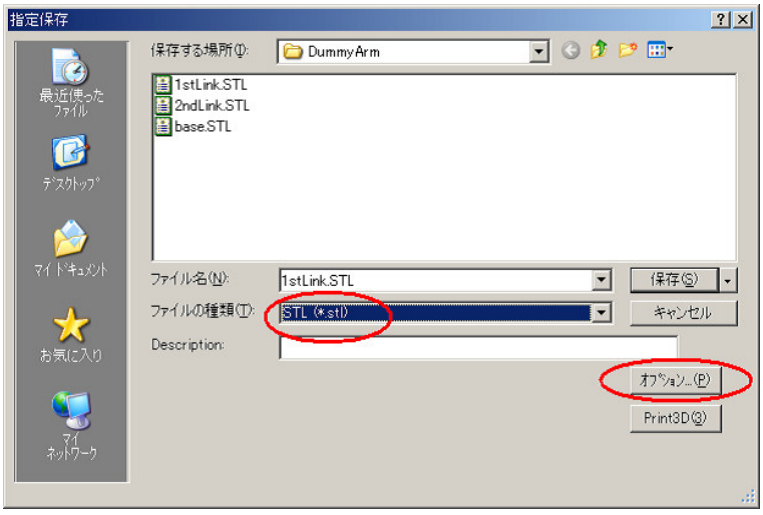

図 20　オプションボタンの押下

エクスポートオプション画面画面が表示されます (図 21). ここでは二つの操作が必要です.

一つ目の操作は出力フォーマットの部分でアスキーにチェックを入れることです. 二つ目の操作はモデルの座標系で出力という部分にチェックを入れることです.

チェックを入れたことを確認したら OK ボタンを押して,STL ファイル形式で保存を行って下さい. お疲れ様でした.

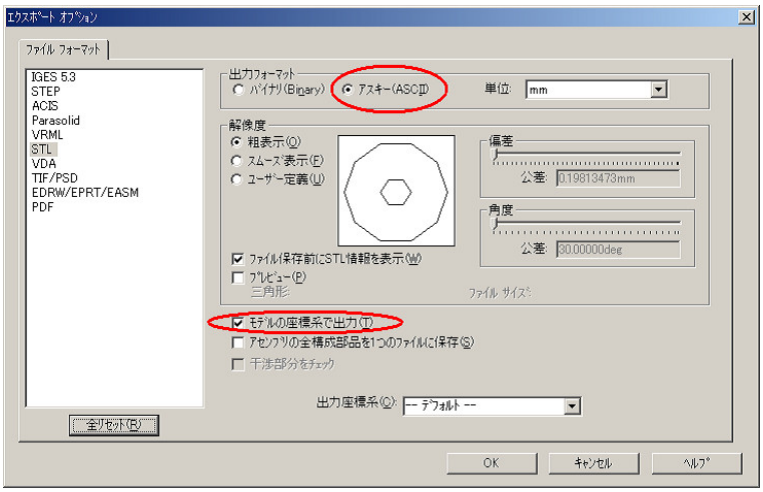

図 21　エクスポートオプション画面での操作