アームユニット RTC Ver.3.0 操作マニュアル

資料番号：RB1400157
改版：第２版

版	年　月	変 更 内 容
初版	2011.04.11	アームユニット RTC Ver2.0 操作マニュアルを元に初版作成
２版	2011.05.11	omniORB のバージョンを修正

このマニュアルの使い方

本マニュアルは、（株）安川電機製のアームユニットを、アームユニット RTC Ver3.0 ソフトウェアを介して使用するためのものです。アームユニットを正しくご使用いただくために、本マニュアルをよくお読みください。また、必要なときに参照することができるように本マニュアルを大切に保管してください。

■登録商標など

- Windows、Windows2000/XP は、米国 Microsoft Corporation.の登録商標です。
- Java は、サン・マイクロシステムズの登録商標です。
- Ethernet は、米国 Xerox Corporation の登録商標です。
- 上記の他、本マニュアルに記載した製品名・会社名などの固有名詞は各社の商標・登録商標・商品名です。本文中の各社の商標・登録商標には TM・®マークは表示しておりません。

■ソフトウェアについて

- 本ソフトウェアの逆コンパイル・逆アセンブルなどを行うことは固くお断りいたします。
- 本ソフトウェアの一部または全部を当社の事前の承認なしに譲渡・交換・転貸などによって第三者に使用させることは固くお断りいたします。

安全上の注意

ご使用（運転、保守、点検など）の前に、必ずこのマニュアルとその他の付属書類をすべて熟読し、正しくご使用ください。機器の知識、安全の情報そして注意事項のすべてについても習熟してからご使用ください。お読みになった後は、ご使用になる方がいつでも参照できる場所に保管してください。

目次

1.はじめに

アームユニットの設置および取扱上の注意

アームユニットは、サービスロボット用アームのモーションを研究するために構成したシステムです。７軸構成で冗長軸を有しているため多様な姿勢を生成することが可能ですが、お客様の作成されたプログラムの内容によっては、アームが予想外に動く、または暴走する可能性があります。そのため、下記の注意事項を熟知し、十分な安全対策を講じてください。必要な安全対策を怠ったり、誤った使い方をしたりすると、アームユニットの破損や故障を招くばかりでなく、作業者のけがや、重大な事故につながりかねません。

設置およびご使用になる際には、必ず以下の事項にご留意ください。

1.1.安全上の注意

1.1.1.適切な設置環境の確保

アームユニットは、姿勢により重心が変化しますので、安定した場所に設置してください。また、アームユニット、充電器は、防爆・防塵・防滴などの仕様にはなっていませんので、つぎのような場所に設置することはできません。

- 屋外環境
- 可燃性ガス・引火性液体などの雰囲気
- 金属加工の削りクズなど導電性物質が飛散している雰囲気
- 酸・アルカリなどの腐食性ガスの雰囲気
- 切削液・研削液などのミスト雰囲気
- イオウ含有の切削液・研削液などのミスト雰囲気
- 大型のインバータ，大出力の高周波発信器，大型のコンタクタ，溶接機などの電気ノイズ源の近傍

1.1.2.作業空間の確保

アームユニットが安全に動作できるための作業空間を、十分に確保して設置ください。

1.1.3.コントロール用 PC の設置場所の確保

アームユニットのコントロール用 PC の設置場所は、アームユニットの可動領域外で、かつアームユニットの動作状況が見渡せる場所に設置してください。

1.2.作業上の注意

警告：動作中のアームユニットに接触すると重傷を負う恐れがありますので、必ず以下のことを守って作業を行ってください。

①アームユニット運転中およびモータ電源が入っているときは、絶対にアームユニットの可動領域内に入らないでください。
②やむを得ずアームユニットの可動領域内で運転を伴う作業を行う場合、必ず「１．２．１ 可動領域内で作業を行う作業者の安全確保」に示す措置を講じてください。
③アームユニットを長時間使用しないときは、バッテリの劣化を防ぐため、アームユニットの電源スイッチをすべて OFF にしてください。
④長時間の連続動作時にモータ及びその周辺が高温になることがあります。触れる場合は、ご注意ください。
⑤バッテリの充電は、2.6 節の手順に従って行ってください。

1.2.1.可動領域内で作業を行う作業者の安全確保

アームユニットの可動領域内で作業を行うときは、ヘルメットの着帽など安全防具を着用するとともに、異常時にただちにアームユニットの運転を停止することができるように、必ず以下の措置を講じてください。

（１） アームユニットの可動領域外でかつアームユニットの動作を見渡せる位置に監視人を配置し、監視業務に専念させて次の事項を行わせてください。
① 異常の際にただちに非常停止装置を作動させる。
② 作業従事者以外の者をアームユニットの可動領域内に立ち入らせない。
（２） 可動領域内の作業者自身が、非常停止スイッチをすぐ押せるように携帯させてください。

1.2.2.実験などの作業開始前の点検

実験などの作業を開始する前に次の事項を点検し、異常を認めたときは、ただちに補修その他必要な措置を講じてください。

（１） アームユニットの安全な設置
（２） 外部電線の被覆または外装の損傷の有無
（３） アームユニットの異常の有無（作動時に異常な音、振動がないか）
（４） 非常停止スイッチの機能
（５） アームユニットの可動領域内またはその付近の障害物の有無

1.3.保守時の注意

保守時には、やむを得ずアームユニットの可動領域内で作業を行うことがあります。その際の安全を確保するために、次の事項を必ず守ってください。

1.3.1.作業規定

以下の事項に対する「作業規定」を定めて十分に作業の安全を確保するとともに、作業者への徹底を図ってください。

- 起動方法・スイッチの取扱方法などの作業において必要となるアームユニットの操作手順
- 保守などの作業を行う場合のアームの速度
- 複数の作業者に作業を行わせる場合の合図の方法
- 異常時に作業者がとるべき異常の内容に応じた措置
- 非常停止スイッチなどが作動してアームユニットの運転が停止した後、これを再起動させるために必要な異常事態の解除の確認・安全の確認などの措置
- 上記以外に、アームユニットの不意の作動による危険または、アームユニットの誤操作による危険を防止するために必要な次に掲げる措置
  - アームユニット可動領域内で作業を行う作業者の安全確保
  - アームユニットの動きが常時確認でき、かつ異常時にすぐ退避できる作業位置および姿勢
  - ノイズ防止対策の実施
  - 関連機器の操作者との合図の方法
  - 異常の種類および判別方法

1.3.2.修理時の注意

（1） 確認作業などのために各装置のふたを開くときは、必ず各装置のパワースイッチを切って、電源ケーブルを取り外してください。

（2） アームユニット修理に関しては、独自に行わず、必ず当社にお問い合わせください。

2.システム概要

2.1.システムの構成

本システムのハードウェア構成を図 1 に、ソフトウェア構成を図 2 に示します。本アームユニットは、安川電機が開発したサービスロボット（SmartPal）のアーム部分であり、７軸で構成されています。

アームユニット RTC は、RT ミドルウェアに対応したソフトウェアモジュールであり、他の RT コンポーネントと接続して使用することができます。

実際にアームユニットを動作させるためには、以下の２つの方法があります。

A)　ユーザが開発したアームユニット RTC コンシューマの使用（図 2 の A）

B)　安川電機が開発した安川電機製アームユニット RTC コンシューマの使用（図 2 の B）

本マニュアルでは B のシステム形態について説明します。全体の起動手順を 4 章で、YE 製アームユニット RTC クライアントの操作方法を 5 章で説明します。

A のシステム形態に関しては、別途アームユニット RTC に対応したクライアントソフトウェアをユーザで準備してください。アームユニット RTC のサービスポートの IDL ファイルについては、付録を参照ください。

図 1　ハードウェア構成

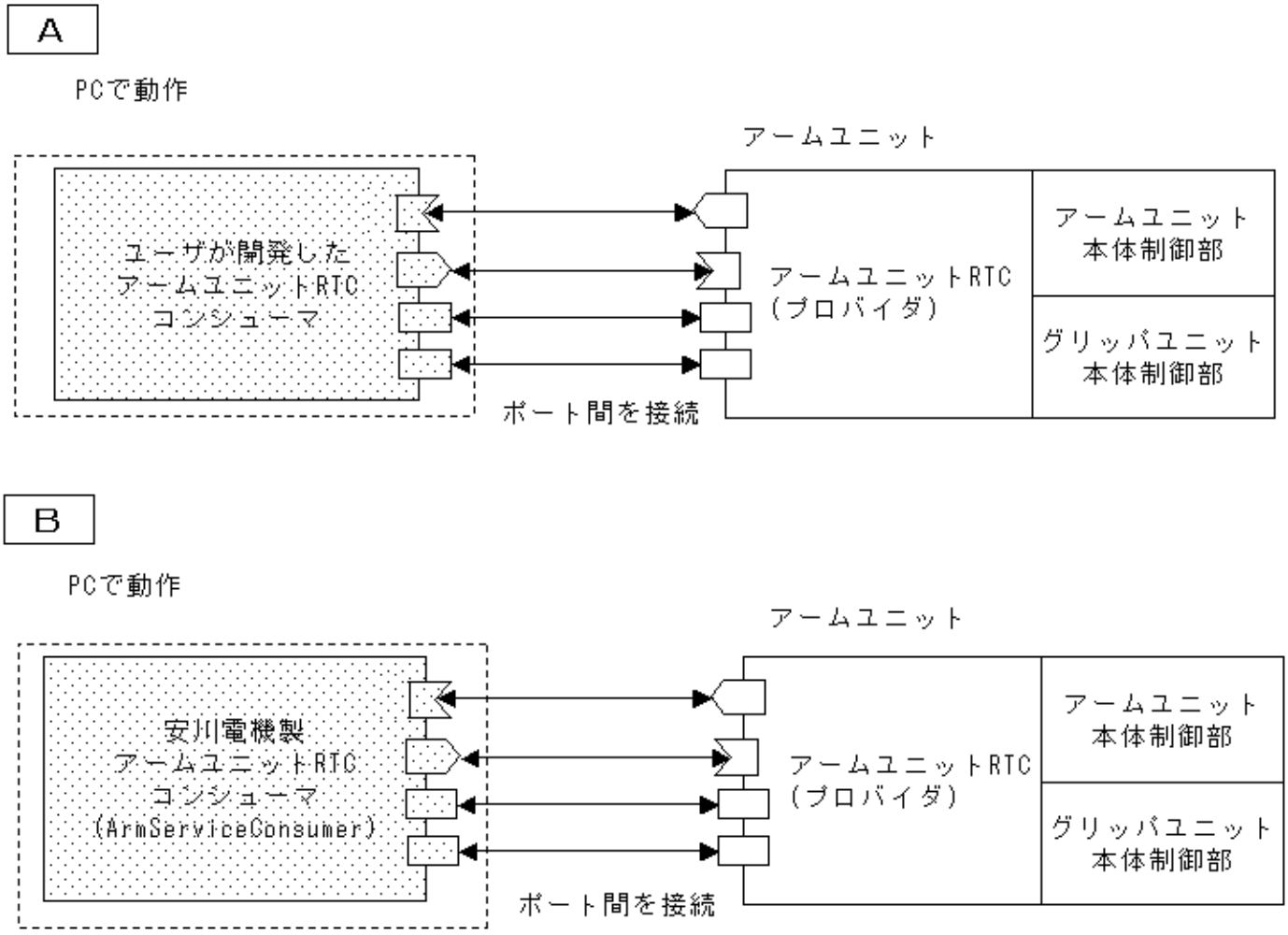

A
PCで動作
アームユニット
ユーザが開発した
アームユニットRTC
コンシューマ
アームユニットRTC
(プロバイダ)
アームユニット
本体制御部
グリッパユニット
本体制御部
ポート間を接続
B
PCで動作
アームユニット
安川電機製
アームユニットRTC
コンシューマ
(ArmServiceConsumer)
アームユニットRTC
(プロバイダ)
アームユニット
本体制御部
グリッパユニット
本体制御部
ポート間を接続

図 2　ソフトウェア構成

2.2.アームユニット本体

アームユニットの外観を図 3 に、サービスロボット（SmartPal）背面の操作パネル部を図 4 に示します。

図 3　アームユニットの外観

図 4　SmartPal の操作パネル

表 1　アームユニット仕様

項目		内容
軸数		７軸（J1～J7）+Gr(グリッパ)
可搬重量		3kg（但し、グリッパの重量を含む）
動作範囲 （メカニカルリミット） [degree]	J1	-45 ～ 180
	J2	-110 ～ 15
	J3	-120 ～ 120
	J4	-2 ～ 130
	J5	-120 ～ 120
	J6	-16.5 ～ 45
	J7	-90 ～ 60
	G r	-57.9 ～ 1.9
最大動作速度 [degree/s]	J1	178
	J2	247
	J3	247
	J4	247
	J5	247
	J6	180
	J7	180
モータ容量 [W]	J1	130
	J2	70
	J3	70
	J4	70
	J5	70
	J6	22
	J7	22
アーム重量		9kg

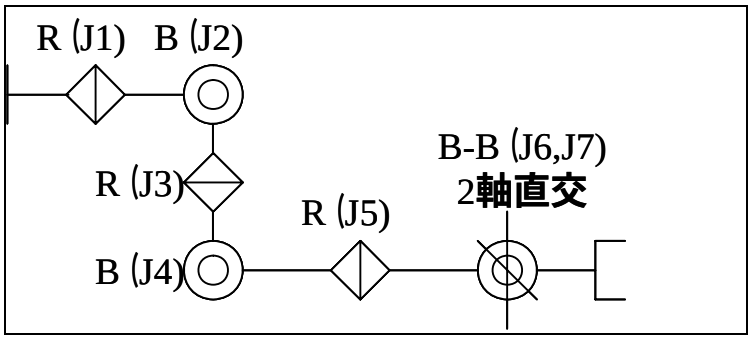


図 5　リンク構成

2.3.座標系

2.3.1.関節座標系

アームユニットは７軸で構成されています。軸の順番は、肩からグリッパに向けてＪ１～Ｊ７と定義します。全ての関節軸を０度とした場合、図に示すようにアームを下方向に垂直に下ろした姿勢（基本姿勢）となります。図において、各関節軸にプラスの位置指令を与えた場合の回転方向を矢印で示しています。Ｊ６およびＪ７は回転中心が交差しているため、軸の順番に注意が必要です。Ｊ６はプラスの位置指令で、前方（ロボット前方）へ回転します。Ｊ７はマイナスの位置指令で、内側（ロボット胴体側）へ回転します。

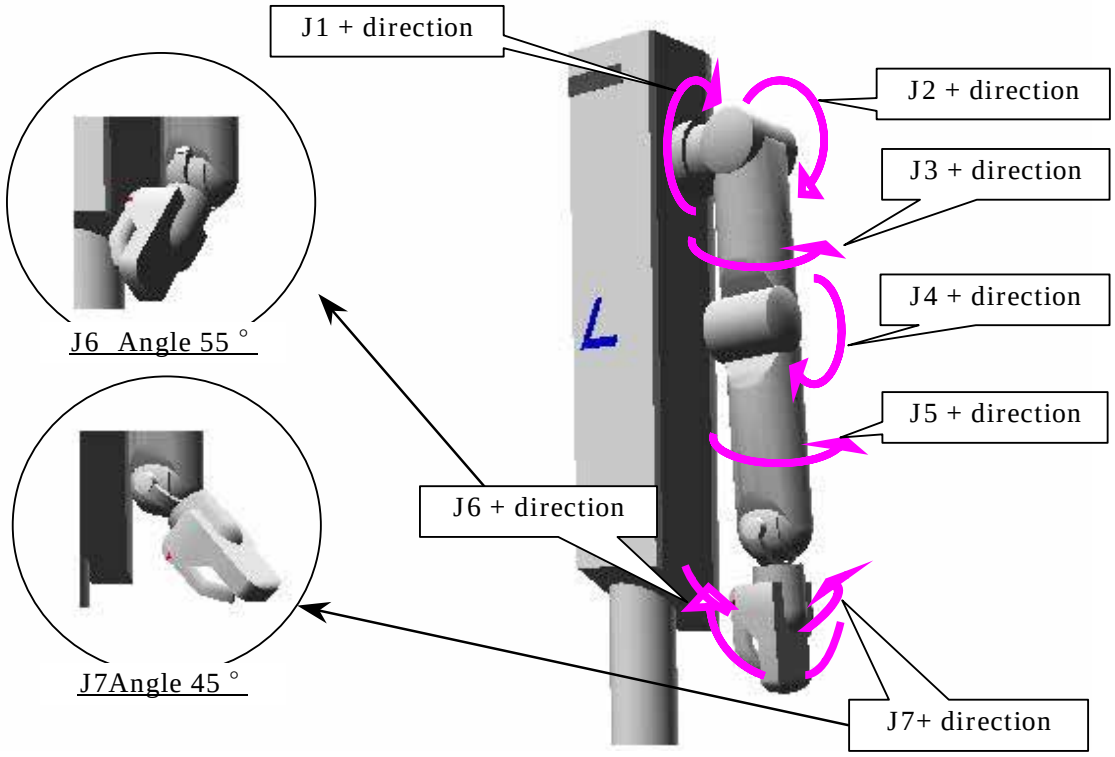


図 6　各関節軸の回転方向

2.3.2.直交座標系

アームユニットの直交座標系の原点位置は、Ｊ１とＪ２の回転軸が交差する位置になります。原点フレームの向きは、図に示す右手座標系となります。（図は左アームの例ですが、右アームについても同様に右手座標系となります。）また、起動時の制御点はフランジ面になっています。制御点を変更する場合は、setControlPointOffset(5.3.17 参照)にて変更可能です。図で示すアーム姿勢の関節座標値を表に、制御点の直交座標値を表に示します。

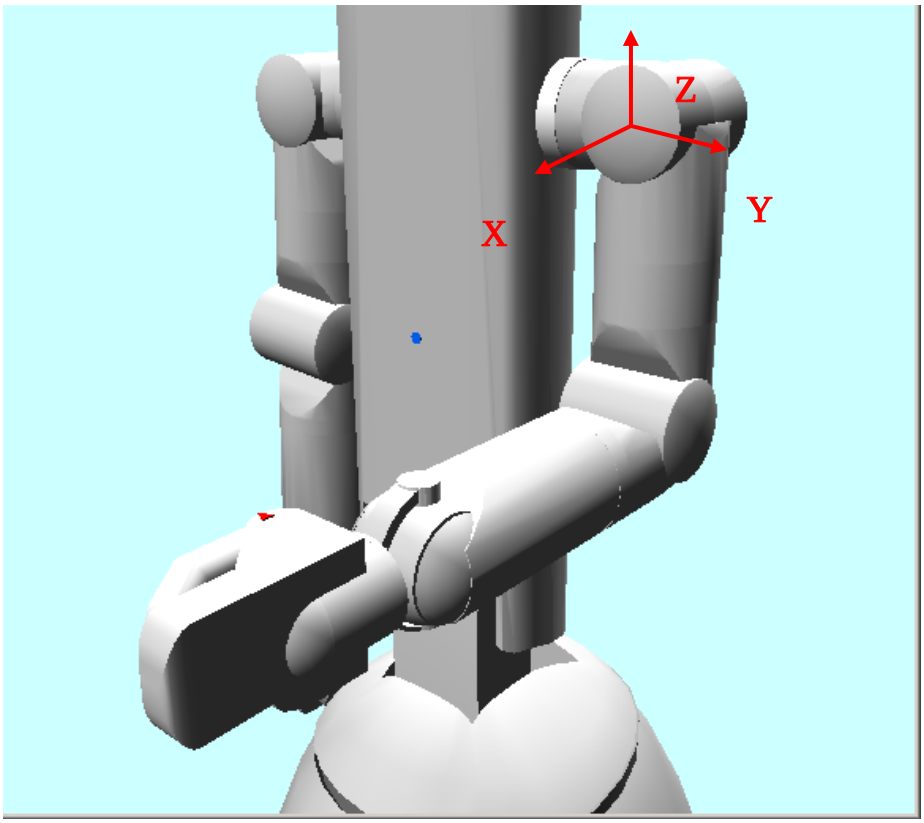


図 7 直交座標系原点と制御点

表 2　関節座標値

関節座標軸	値[degree]
J1	0
J2	0
J3	0
J4	90
J5	0
J6	0
J7	0

表 3　直交座標値

直交座標軸	値（左アーム）	値（右アーム）
x[mm]	355.89	355.89
y[mm]	0	0
z[mm]	-270	-270
rx[degree]	0	-180
ry[degree]	0	0
rz[degree]	0	0
elbow[degree]	-90	90

2.4.別途準備していただく必要があるもの

アームユニットを操作するにあたって、下記のものを別途準備していただく必要があります。

- パーソナルコンピュータ (以下、「PC」と記述)
- Ethernet ケーブル
- アームユニット RTC モジュール
- Java
- Eclipse
- OpenRTM-aist(Java 版)
- OmniORB
- FTP サーバ

(1) PC

アームユニットを操作するために「アームユニット RTC モジュール」に含まれるファイルを PC にコピーする必要があります(「3.1PC のセットアップ」を参照)。HDD の空き容量が 500MB 以上あることを確認してください。

製品名称	形式・仕様
PC	Eclipse が動作すること HDD 空き容量：500MB 以上

(2) Ethernet ケーブル

アームユニットと PC を Ethernet ケーブルで接続します。Ethernet ケーブル（ストレート）を使用する場合は、HUB を中継させてください。

製品名称	形式・仕様
Ethernet ケーブル	10BASE 又は、100BASE

(3) アームユニット RTC モジュール

アームユニット RTC モジュールは、アームユニットの RTC ソフト本体であり、アームユニットを動作させるために必要です。

(4) Java

Java は、アームユニット RTC および RT ミドルウエアを動作させるために必要です。
（バージョン 1.6 推奨）

(5) Eclipse

Eclipse は、RT SystemEditor を動作させるために必要です。
(バージョン 3.4 推奨)

(6) OpenRTM-aist

OpenRTM-aist は、アームユニット RTC を動作させるために必要です。バージョンは 1.0.0 です。

(7) omniORB

omniORB は、NameService 機能を RT ミドルウェアやアームユニットを利用するために必要です。(バージョン 4.1.2 推奨)

(8) FTP サーバ

アームユニット本体が PC から RTC モジュールをダウンロードするために必要です。

2.5.非常停止スイッチについて

非常停止スイッチには、SmartPal 背面の非常停止スイッチ(図 9 の A)とケーブル付き非常停止スイッチ(図 8、図 9 の B)の 2 種類あります。どちらの非常停止スイッチを押してもアームユニットを停止することができます。非常停止スイッチを押すと、アームユニット駆動部の主回路が OFF になり、アームユニットが停止します。非常停止からの復旧方法は、「4.6 非常停止からの復旧方法」を参照ください。

非常停止スイッチの接続は、**必ずアームユニットの電源を切った状態で**行ってください。ケーブル付き非常停止スイッチを使用する際は、ケーブルを SmartPal の車輪などに巻き込まないようご注意ください。

ケーブル付き非常停止スイッチをご使用しない場合は、必ずダミーコネクタを接続してください。接続していない場合、アームユニットは動作しません。

図 8　ケーブル付き非常停止スイッチ

図 9　非常停止スイッチの接続

2.6.充電について

SmartPal のバッテリ電圧が 48.0[V]を下回ったらアームユニットの使用を中止し、バッテリを充電してください。バッテリの充電は専用の充電器で行います。充電器の外観を図 10 に示します。

図 10　充電器の外観

コネクタをバッテリに接続した後、充電器の電源スイッチを ON にすると充電器の電源ランプが点灯し、充電を開始します。充電が完了すると、ステータスランプの色が赤から緑になります。充電の詳細ステップを以下に示します。

① SmartPal のブレーカ、制御スイッチが OFF になっていることを確認してください。
② SmartPal のバッテリ充電口と充電器のコネクタを接続してください。
(充電器のコネクタが２つありますが、どちらでもかまいません)
③ 充電器の電源スイッチを投入してください。
④ 充電器のステータスランプが緑色になったら充電完了です。

2.7.エンコーダバックアップバッテリの交換および故障時の対応

アームユニットは絶対値エンコーダを使用しており、バックアップバッテリを必要とします。バックアップバッテリの電圧が低下すると、エンコーダバックアップエラー（アラームコード 0x33030810）が発生します。

バッテリ電圧 3.0V 以下もしくは 200 日を目安とした交換が必要となります。交換については下記担当者までご連絡ください。

また、アームユニットが起動しない、あるいはトラブルが発生した場合は、まず下記対応をお試しください。それでも復旧しない場合は、状況を下記担当者までご連絡ください。

（想定される故障と対応）

A. **納入時と動きが違う。**

非常停止スイッチを押して、全モータの電源を停止してください。

モータの制御装置にエラーが発生し、モータが動作しなくなったと考えられます。そのまま動作させますと、お客様が負傷したり、アームユニットが破損したりして危険です。その後全ての電源を落としてください。

B. **突然アームユニットの電源が切れて停止してしまった。**

電源のスイッチを OFF にしてください。

⇒バッテリが切れてしまったことが考えられます。

バッテリの充電後、再度電源を入れて操作してみてください。

復旧しない場合、下記担当者までご連絡ください。

安川電機　技術開発本部　開発研究所（ロボット技術開発グループ）

担当者　足立勝　（アダチマサル）

亀井泉寿（カメイモトヒサ）

Tel　:093-571-6026

3.動作環境セットアップ

3.1.PC のセットアップ

アームユニットを操作するために、以下の手順に従って PC をセットアップしてください。それぞれの手順に関する詳しい説明は、右欄に記載した各節をご覧ください。

1. Java のインストール	「3.1.1 Java のインストール」
⇩	
2. omniORB のインストール	「3.1.2 omniORB のインストール」
⇩	
3. Eclipse のインストール	「3.1.3 Eclipse のインストール」
⇩	
4. OpenRTM のインストール	「3.1.4 OpenRTM-aist のインストール」
⇩	
5. ファイルのコピー アームユニット RTC モジュールから必要なファイルを PC へコピーします。	「3.1.5 ファイルの展開」
⇩	
6. FTP サーバのインストール	「3.1.6 FTP サーバのインストール」

3.1.1.Java のインストール

①下記 URL を開き、JDK(バージョン 1.6)ファイルをダウンロードし、インストールを行ってください。

URL: http://java.sun.com/javase/ja/6/download.html

②インストール手順は下記 URL を参照してください。

URL: http://java.sun.com/javase/ja/6/webnotes/install/index.html

3.1.2.omniORB のインストール

①CD-ROM あるいは下記の URL から omniORB のパッケージ(omniORB-4.1.2-x86_win32_vc8.zip)をダウンロードし、適当な場所に解凍してください。

URL: **http://sourceforge.net/projects/omniorb/**

②システム環境変数の設定を行います。「スタート」メニューから、「設定」－「コントロールパネル」を選択し、表示されたコントロールパネル画面から「システム」を選択します。「システムのプロパティ」画面では「詳細設定」タブを選択し、「環境変数」ボタンをクリックします。

③「システム環境変数」の環境変数 PATH に以下の内容を追加してください。

<解凍先ディレクトリ>¥omniORB-4.1.2¥bin¥x86_win32

④ディレクトリ

<解凍先ディレクトリ>¥omniORB-4.1.2¥log

を作成してください。

⑤設定ファイル“ omniORB.cfg ”を CD-ROM から以下の場所にコピーしてください。

<解凍先ディレクトリ>¥omniORB-4.1.2

⑥「システム環境変数」から、「新規」ボタンをクリックし「新しいシステム変数」画面で、以下の内容を入力してください。

変数名	変数値
OMNINAMES_LOGDIR	<解凍先ディレクトリ>¥omniORB-4.1.2¥log
OMNIORB_CONFIG	<解凍先ディレクトリ>¥omniORB-4.1.2¥omniORB.cfg

3.1.3.Eclipse のインストール

①下記 URL の「TOP＞ダウンロード＞ＧＵＩ ツール関係」の,「Eclipse3.4.2+RTSE+RTCB Windows 用全部入り」をダウンロードしてください。

②ダウンロードしたファイルを任意のフォルダ(C:¥Program Files 等)へ解凍してください。

URL:**http://www.openrtm.org/openrtm/ja/content/openrtm-eclipse-tools-10-release**

3.1.4.OpenRTM-aist のインストール

①下記 URL の「OpenRTM-aist-Java-1.0.0.msi」をダウンロードし, インストールを行ってください。

URL: **http://www.openrtm.org/openrtm/ja/content/openrtm-aist-java-100-release**

3.1.5.ファイルの展開

「アームユニット RTC モジュール」に含まれている ZIP ファイルを展開し、展開された
ファイルを PC のフォルダに保存してください。以下の手順に従って、PC に保存してください。

①「アームユニット RTC モジュール」に含まれる" ArmUnitRTC_Binary.zip"をＣドライブの直下に展開してください。
②"ArmUnitRTC_Binary.zip"を展開するとフォルダ(RTC)が生成されます。

3.1.6.FTP サーバのインストール

任意の FTP サーバをインストールしてください。
以下で設定してください。
カレントフォルダ：Ｃドライブ直下
ユーザＩＤ：fmk
パスワード：fmk
※移動ユニット本体からモジュールをダウンロードする為に必要です。
“192.168.11.70”に設定された PC にて FTP サーバをインストールしてください。

3.1.7.ネットワークの設定

PC の IP アドレスとサブネットマスクをアームユニットと接続可能な IP アドレス(たとえば"192.168.11.xx")とサブネットマスク(“255.255.255.0”)に変更してください(図 11)。CORBA のネームサービスを本 PC で起動する場合は、必ず"192.168.11.70"に変更してください。アームユニットが使用するネームサービスの IP アドレスの出荷値は、"192.168.11.70"に設定しているためです。

アームユニット自体の IP アドレスの出荷値は 192.168.11.73 に設定しています。

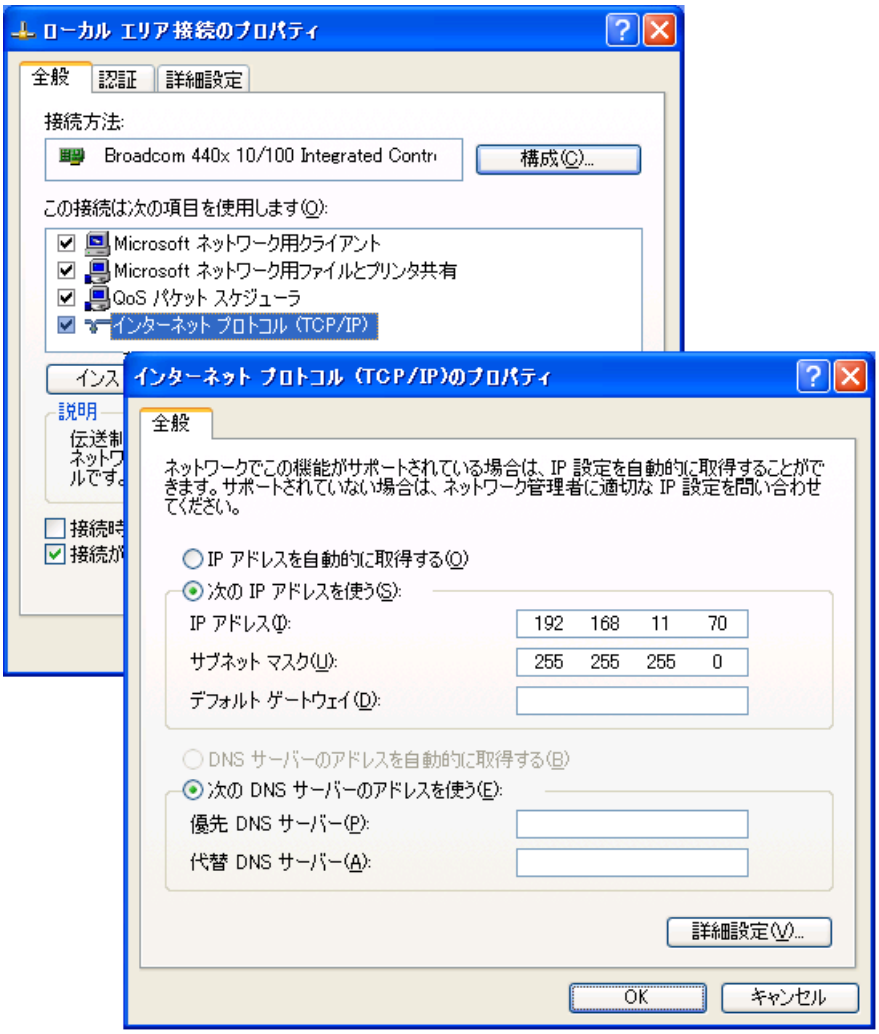


図 11　ネットワークの構成

3.2.omniNamesServer 起動バッチの作成

① [スタート]メニューから[すべてのプログラム] - [アクセサリ] - [メモ帳]を開きます。

② 下記を記述してください。

```
del %OMNINAMES_LOGDIR%¥*.*/Q
omninames -start 5005
```

③ [ファイル] - [名前を付けて保存]でデスクトップに「omniNamesStart.bat」で保存してください。

4.動作準備

4.1.Ethernet ケーブルの接続

Ethernet ケーブルを使用して、アームユニットと PC を Ethernet で接続してください。

4.2.アームユニットの電源投入

① 3.2 で作成した「omniNamesStart.bat」をダブルクリックし実行してください。
② アームユニットの電源を ON にしてください。

4.3.Eclipse の起動

①PC にインストールした Eclipse を起動してください。
②ワークスペースを選択してください。(図 12)

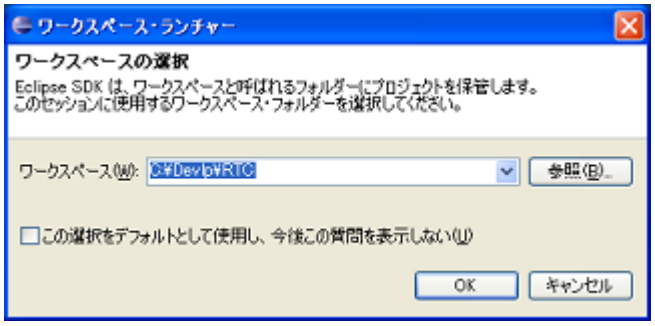


図 12　ワークスペースの選択

4.4. コンポーネントの起動

4.4.1. ArmServiceConsumer の起動

下記手順で ArmServiceConsumer を起動してください。

①"C¥RTC¥RTCWrapArmUnit¥bin¥ArmServiceConsumer_Right.bat を実行してください。左アームの場合は ArmServiceConsumer_Left.bat を実行してください。

②ArmService 画面（図 13）が表示されます。

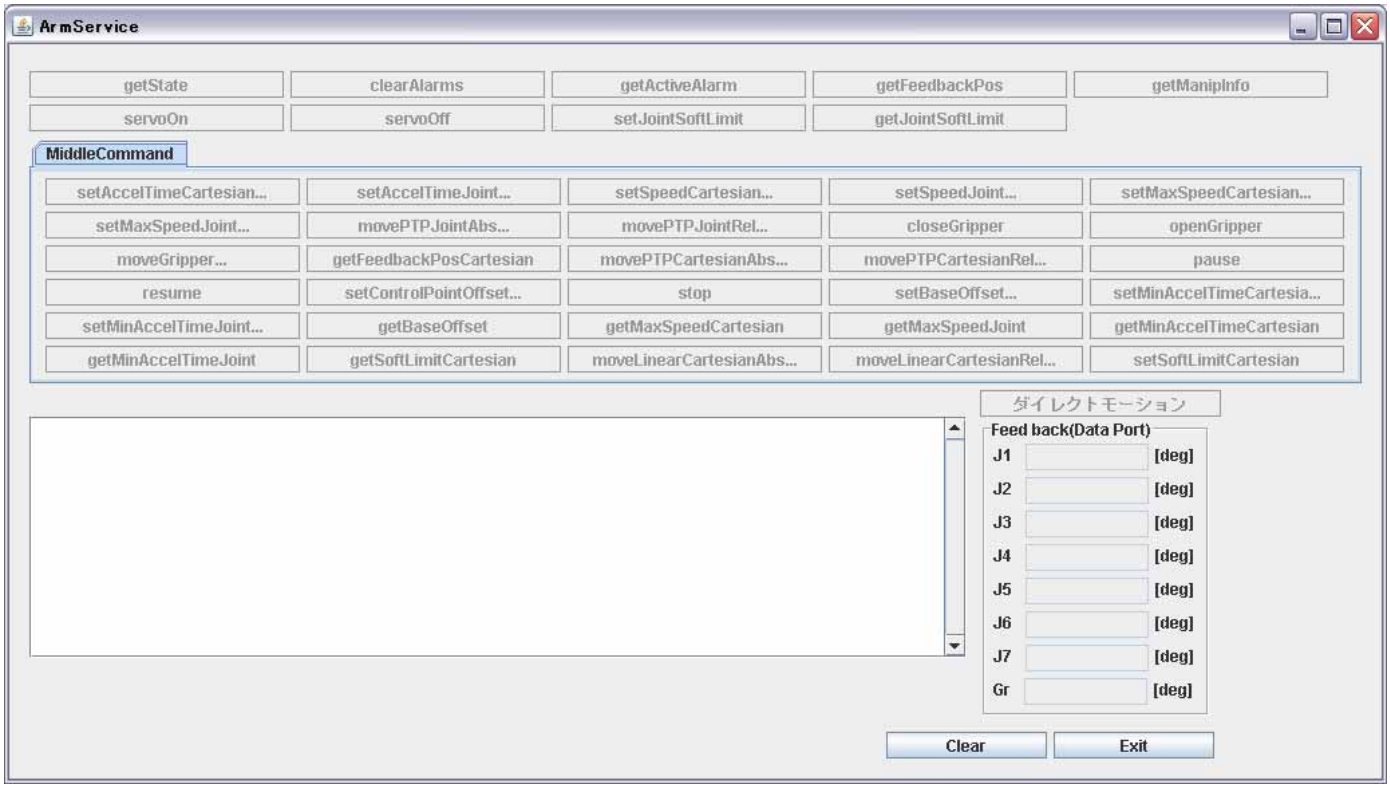


図 13 ArmService 画面

4.5.コンポーネントの接続

①Eclipse 上で RTSystemEditor を開いてください。画面上にない場合は「パースペクティブを開く」メニューから選択してください。

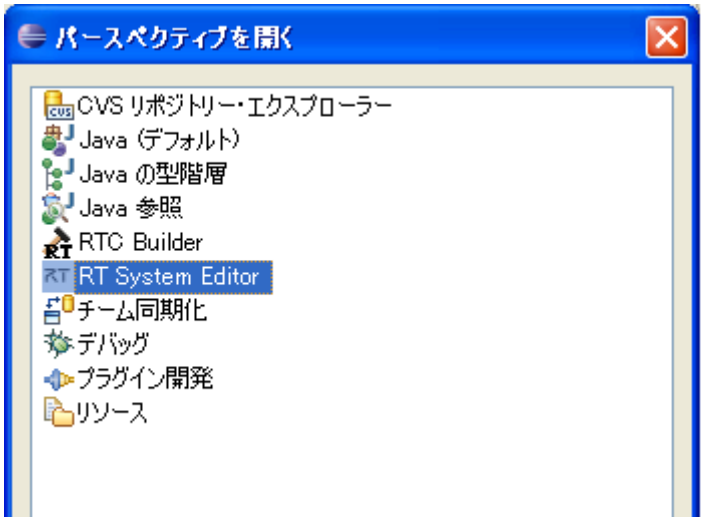


図 14　RTSystemEditor のオープン

②NameServiceView の領域内に「192.168.11.70:5005」（または、localhost:5005）が無い場合は、右クリックを行ってコンテキストメニューを表示し、「Add Name Server」を選択し、「192.168.11.70:5005」を入力してください。

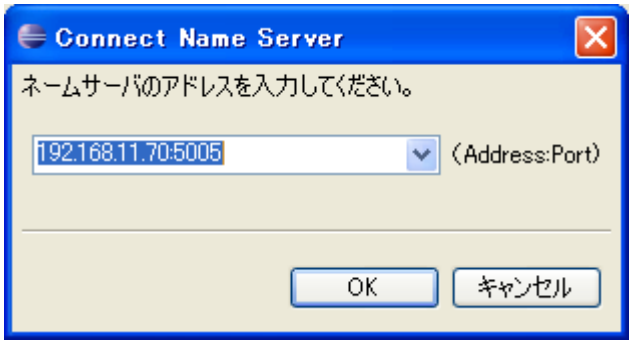


図 15　ネームサーバへの接続

③ NameServiceView 内の「ArmServiceConsumer0_Right/rtc」と「rightarmService0/rtc」を SystemDiagram へドラッグしてください。SystemDiagram 画面はツールメニュー上の ON ボタンで開くことができます。

④SystemDiagram 内の「 」と「 」をマウスでドラッグして結びつけます。
⑤下記画面が表示されますので「OK」ボタンを押下してください。

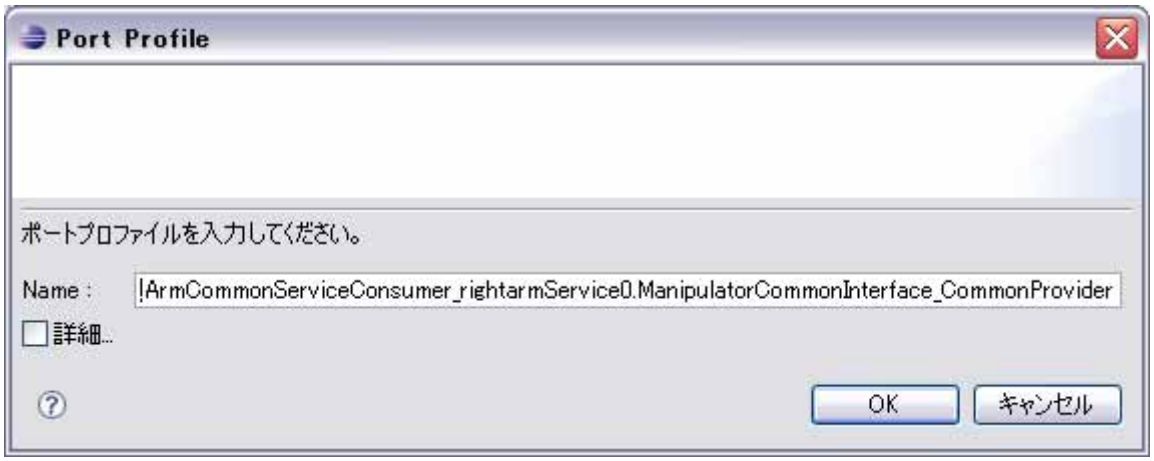
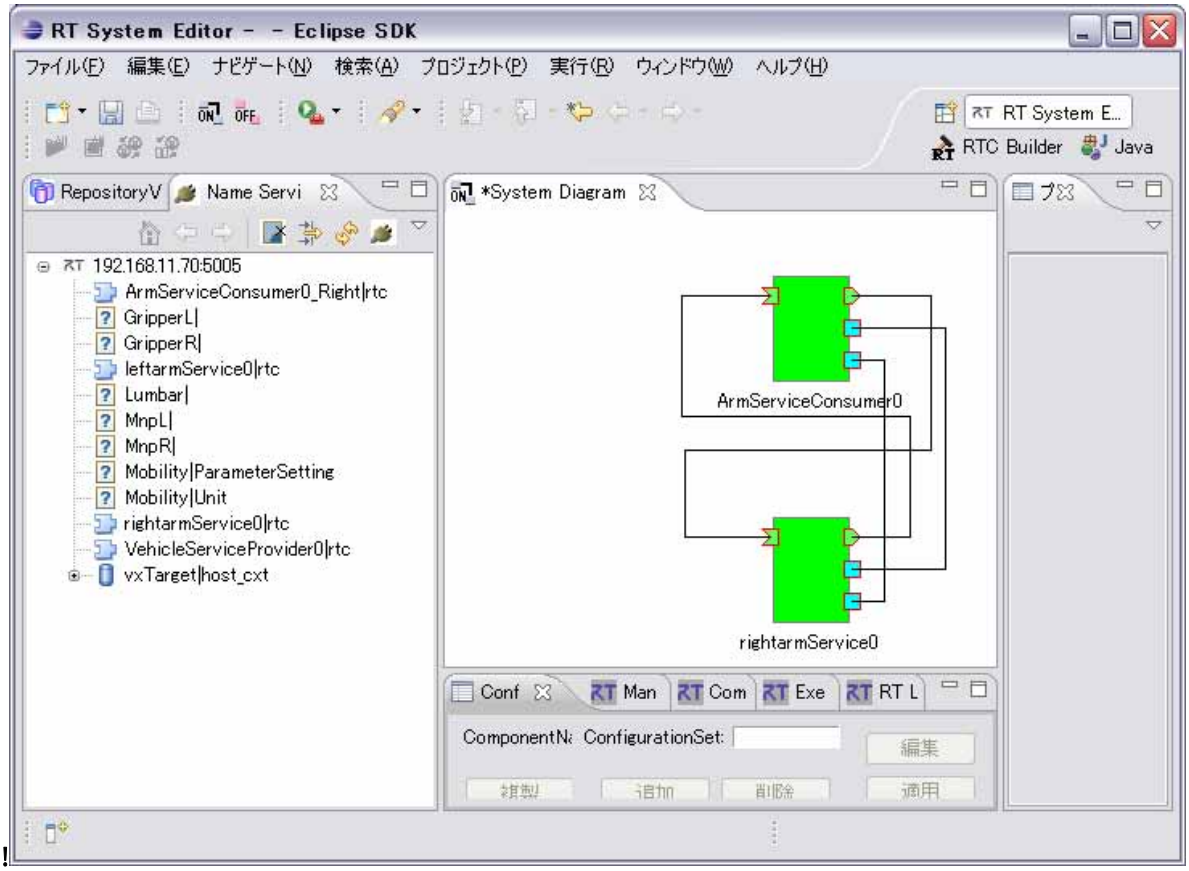


図 16　Port Profile 画面

⑥SystemDiagram 内で右クリックし、「All Activate」を選択してください。確認のメッセージボックスが表示されますので、「OK」ボタンを押下してください。
⑦画面のように、２つのコンポーネントの色が黄緑色になることを確認してください。



エラー!

図 17　コンポーネントの Activate

⑧ArmService 画面の各ボタンが押せるようになります。
ArmService 画面の操作については、「5 アームユニットの操作」をご参照ください。

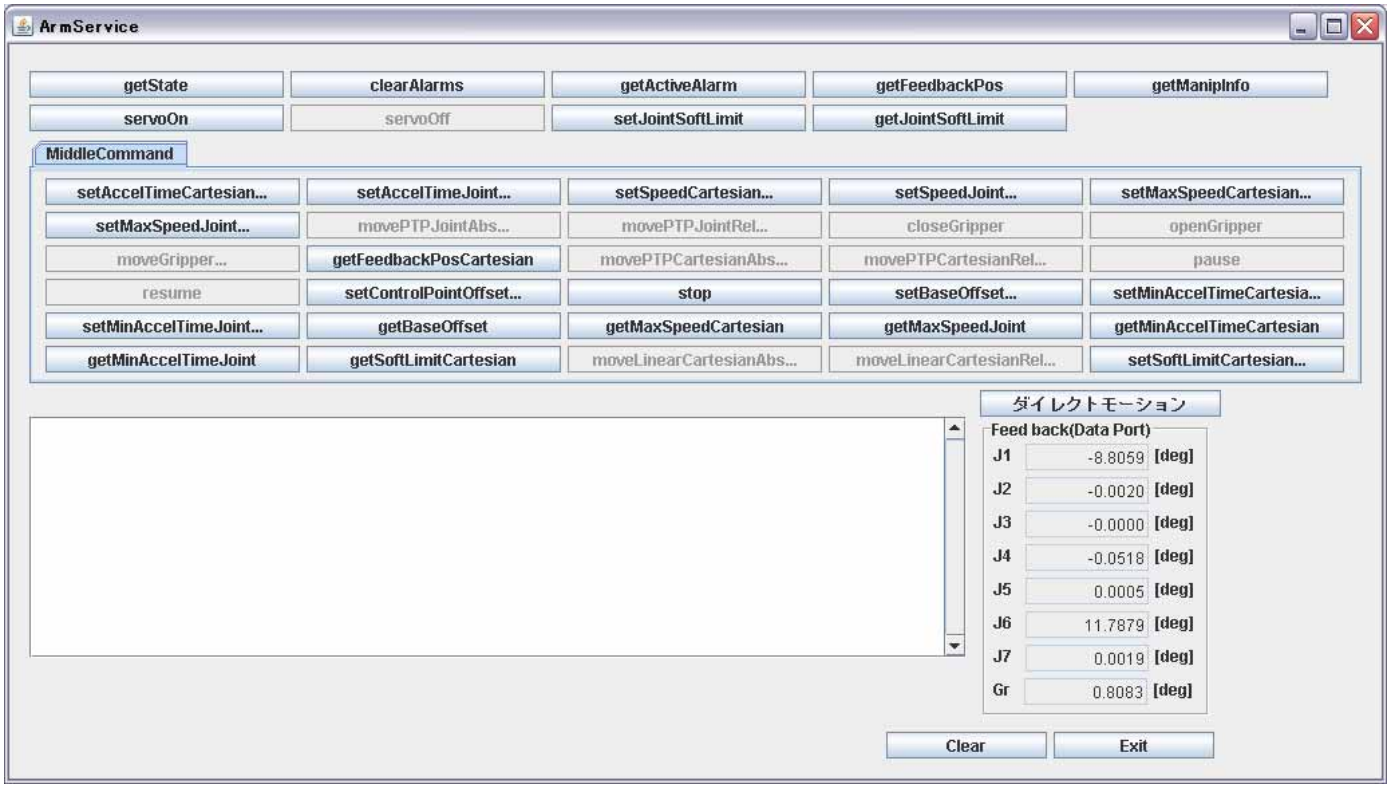


図 18　ArmService 画面

4.6.非常停止からの復旧方法

非常停止スイッチを押すことで、アームユニット駆動部の主回路が OFF になり、アームユニットを停止させることができます。非常停止した状態から動作復旧させるには次の手順で操作します。

＜1＞ 電源再投入

① SmartPal の電源を OFF にします。
② 「4.2 アームユニットの電源投入」の手順に従って、アームユニットの電源投入を行います。
③ Eclipse を終了させます。
④ 「4.3Eclipse の起動」「4.4 コンポーネントの起動」「4.5 コンポーネントの接続」の手順に従って復旧させます。

＜2＞ArmService 画面による動作復旧：

アームユニットの電源を遮断することなく非常停止状態から普及させる方法です。

① ArmService 画面の ClearAlarm ボタンを押下します。
② ArmService 画面の ServoOn ボタンを押下します。

5.アームユニットの操作

5.1.ArmService 画面の説明

ArmService の画面は図 19 に示すように、各コマンドを実行するためのボタン群と、実行結果を表示する実行結果表示エリア、各コマンドの実行によって取得した情報を表示する取得情報表示エリアから構成されます。

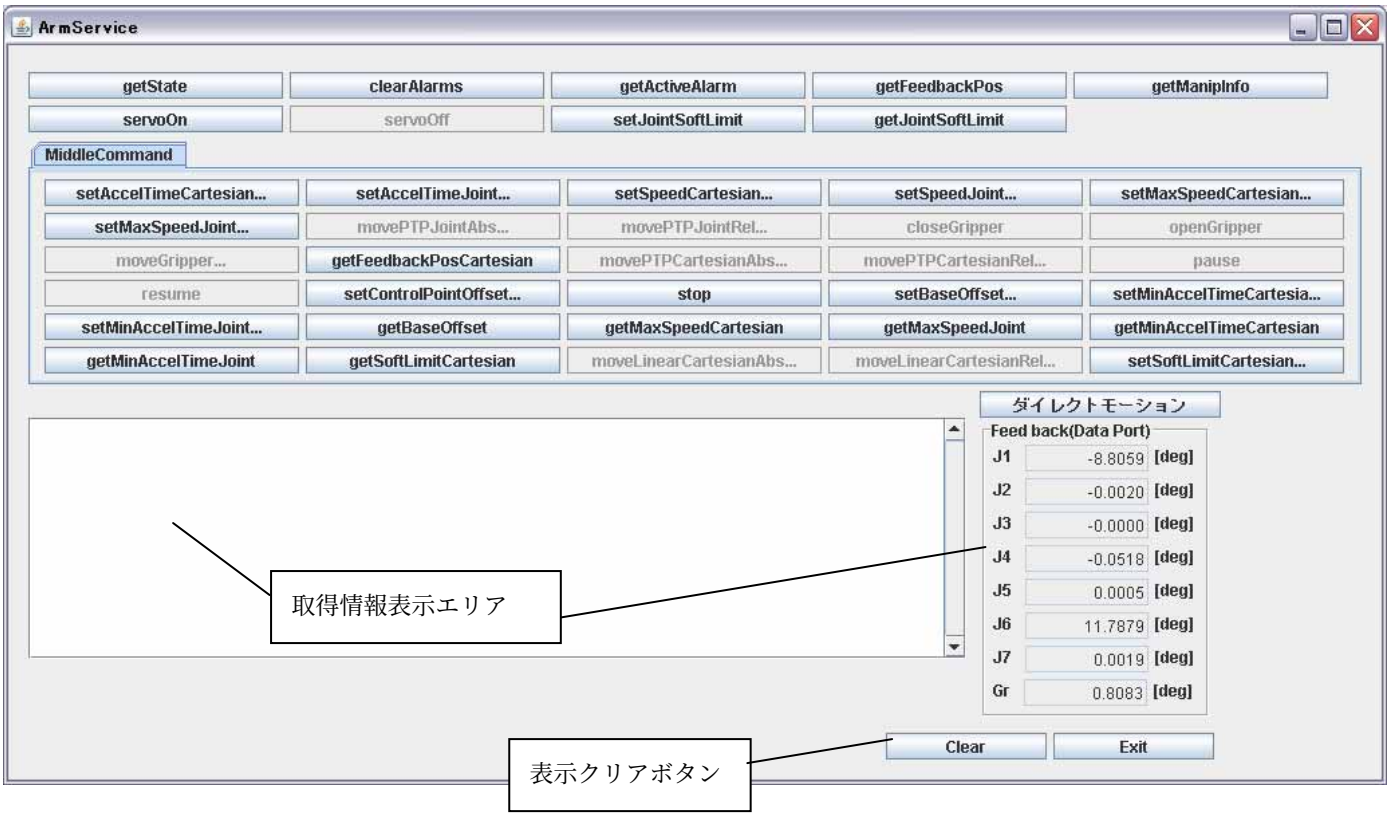


図 19　ArmService 画面

5.2.共通コマンド

5.2.1.servoOn

アームユニットの主回路電源、駆動用モータのサーボ制御を入れます。

5.2.2.servoOff

アームユニットの主回路電源、駆動用モータのサーボ制御を切ります。

5.2.3.getFeedbackPosJoint

各軸のフィードバック位置情報[degree]を取得します。

5.2.4.clearAlarms

サーボアンプとコントローラに発生した全てのアラームのクリアを実行します。

5.2.5.getActiveAlarm

発生中のアラーム情報を取得します。

5.2.6.getManipInfo

マニピュレータ情報を取得します。

manufactur：メーカ名

type：機種名

axisNum：軸数（グリッパを除く）

cmdCycle：低レベル位置指令を受ける周期[ms]

isGripper：１軸グリッパの有無（グリッパ未装着時及び多指ハンド装着時は、false とする）

5.2.7.getSoftLimitJoint

関節座標系のソフトリミット値[degree]を取得します。

5.2.8.getState

アームユニットの状態を取得します。

表 4 にユニットの状態ビット一覧を示します。

表 4　アームユニットの状態一覧

状態コード	説明
0x01	サーボ On 中
0x02	動作中
0x04	アラーム発生中
0x08	Move 命令のバッファがフル
0x10	一時停止中

5.2.9. setSoftLimitJoint

関節座標系のソフトリミット値を設定します。（※将来機能です）

5.3. 中レベルモーションコマンド

5.3.1. setAccelTimeCartesian

直交動作時の加速時間[s]を設定します。デフォルト値は 1[s]です。

5.3.2. setAccelTimeJoint

関節動作時の加速時間[s]を設定します。デフォルト値は 1[s]です。

5.3.3. setSpeedCartesian

直交動作時の速度を[%]設定します。

5.3.4. setSpeedJoint

関節動作時の速度を[%]設定します。

5.3.5. setMaxSpeedCartesian

直交動作時の最大動作速度を設定します。

5.3.6. setMaxSpeedJoint

関節動作時の最大動作速度を設定します。

5.3.7. movePTPJointAbs

関節座標系における直線補間を実行します。

各軸を現在位置から指定された終点位置（絶対座標）まで直線的に移動させます。速度は、setMaxSpeedJoint で設定された中で、動作角度が一番大きい軸の値(現在の角度からではなく、最後に指令した角度)×setSpeedJoint で設定された値[%]となります。

加速時間は、setAccelTimeJoint で設定された時間となります。

その他の軸の速度は、すべての軸が同時起動、同時停止するように決定されます。

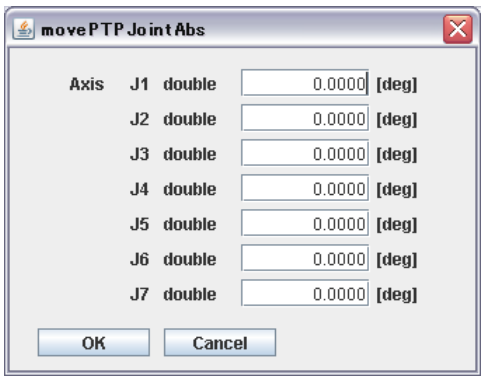


図 20 movePTPJointAbs

5.3.8. movePTPJointRel

関節座標系における直線補間を実行します。

各軸を現在位置から指定された終点位置（相対座標）まで直線的に移動させます。速度は、setMaxSpeedJoint で設定された中で、動作角度が一番大きい軸の値×setSpeedJoint で設定された値[%]となります。

加速時間は、setAccelTimeJoint で設定された時間となります。

その他の軸の速度は、すべての軸が同時起動、同時停止するように決定されます。

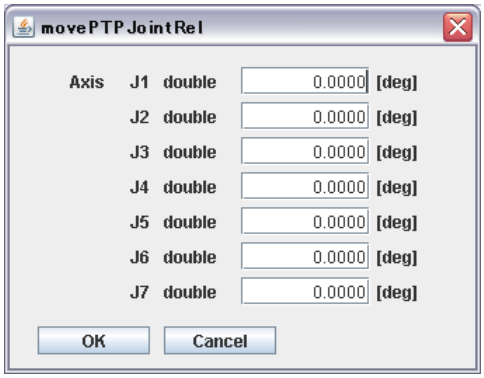


図 21 movePTPJointRel

5.3.9. CloseGripper

グリッパを完全に閉じます。

5.3.10. OpenGripper

グリッパを完全に開きます。

5.3.11. moveGripper

グリッパを指定された角度[%](0%～100%)へ動作します。

0%：完全に閉じた状態

100%：完全に開いた状

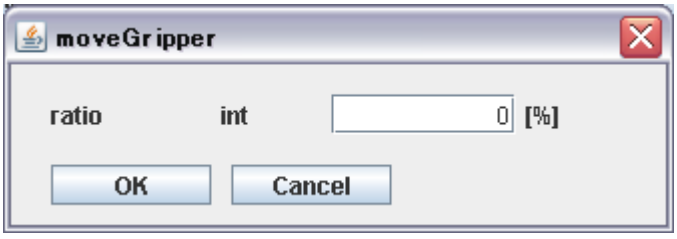


図 22 moveGripper

5.3.12. getFeedbackPosCartesian

ロボット座標系でのフィードバック位置情報を取得します。

5.3.13. movePTPCartesianAbs

関節空間において、目標位置をロボット座標系絶対直交座標指定により、直線補間を動作します。（※将来機能です）

5.3.14. movePTPCartesianRel

関節空間において、目標位置を相対直交座標指定により、直線補間を動作します。（※将来機能です）

5.3.15. pause

一時停止状態になります。動作中の場合は減速停止します。

一時停止状態中に指示された動作指令は内部に蓄積され、一時停止状態から復帰した後に実行されます。低レベル位置指令動作中の一時停止は行えません。

5.3.16. resume

一時停止状態から復帰します。

5.3.17. setControlPointOffset

アームの制御点の位置を設定します。設定はフランジ面からのオフセット量で行います。

制御点の位置[mm]・姿勢[degree]

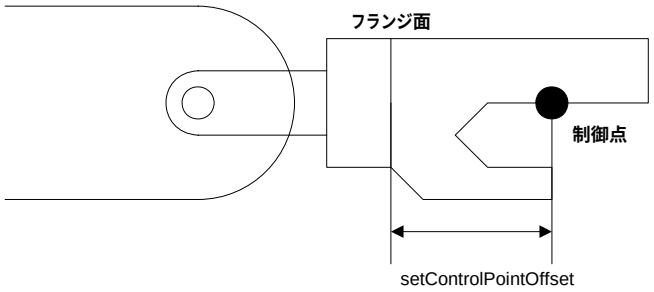


図 23　オフセット量

5.3.18. stop

現在の動作を減速停止します。

5.3.19. setBaseOffset

本マニピュレータのアーム座標系からロボット座標系（基準）までのオフセット量を設定します。

図 24 ベースオフセットを用いて、ベースオフセットの設定例を説明します。本例のロボットシステムは、右を向いたロボットＡと左を向いたロボットＢの２台から構成されます。各ロボットのベース部には、右手系のアーム座標系が設定されています。ユーザは、ロボットＢの座標系をロボットＡの座標系に合わせて運転させたい。すなわちロボットＡのアーム座標系を本ロボットシステムのロボット座標系としたい。この場合、ロボットＢのアーム座標系から見たロボットＡのアーム座標系までの位置・姿勢のオフセット量を、setBaseOffset オペレーションを使って設定します。

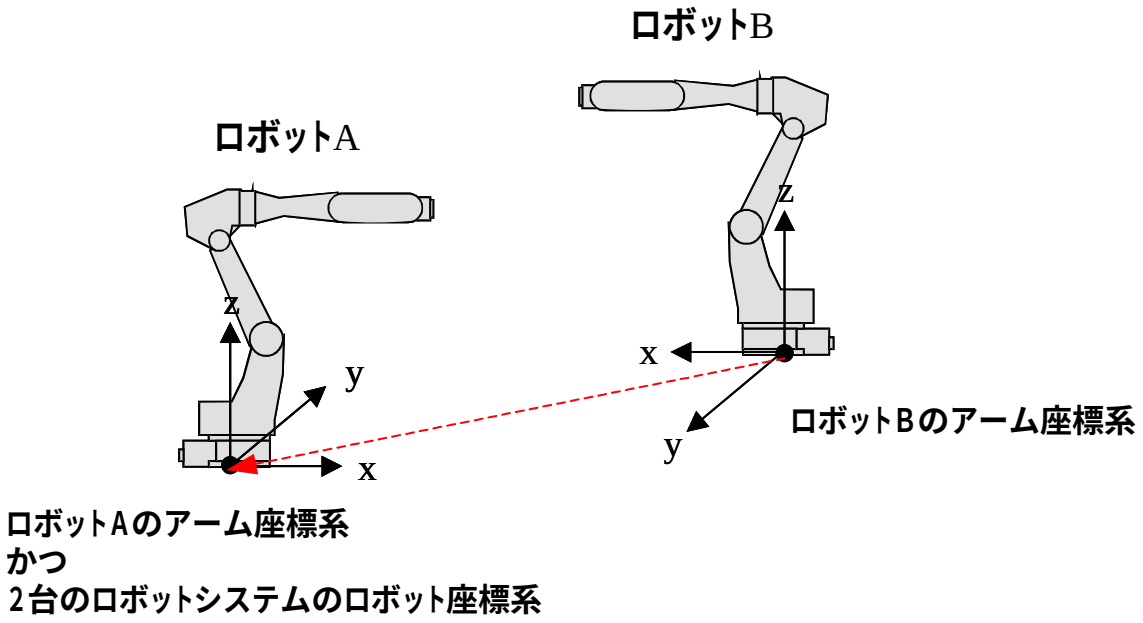


図 24 ベースオフセット

5.3.20. setMinAccelTimeCartesian

直交動作時の最大速度までの最小動作加速時間[s]を設定します。

5.3.21. setMinAccelTimeJoint

関節動作時の最大速度までの最小動作加速時間[s]を取得します。

5.3.22. getBaseOffset

アーム座標系からロボット座標系までのオフセット量を取得します。

5.3.23. getMaxSpeedCartesian

直交動作時の最大動作速度を取得します。

5.3.24. getMaxSpeedJoint

関節動作時の最大動作速度を取得します。

5.3.25. getMinAccelTimeCartesian

直交動作時の最大速度までの最小動作加速時間を取得します。

5.3.26. getMinAccelTimeJoint

関節動作時の最大速度までの最小動作加速時間を取得します。

5.3.27. getSoftLimitCartesian

ロボット座標系でのソフトリミット値を取得します。（※将来機能です）

5.3.28. moveLinearCartesianAbs

直交座標系における直線補間を実行します。

終点位置は絶対座標で指定します。elbow は冗長軸（肘）の姿勢です。速度は、setMaxSpeedCartesian で設定された最大速度×setSpeedCartesian で設定された値[%]となります。
加速時間は、setAccelTimeCartesian で設定された時間となります。その他の軸の速度は、すべての軸が同時起動、同時停止するように決定されます。

制御点の位置[mm]・姿勢[degree]

5.3.29. moveLinearCartesianRel

直交座標系における直線補間を実行します。
終点位置は相対座標で指定します。elbow は冗長軸（肘）の姿勢です。速度は、setMaxSpeedCartesian で設定された最大速度×setSpeedCartesian で設定された値[%]となります。
加速時間は、setAccelTimeCartesian で設定された時間となります。その他の軸の速度は、すべての軸が同時起動、同時停止するように決定されます。

制御点の位置[mm]・姿勢[degree]

5.3.30. setSoftLimitCartesian

直交座標系のソフトリミット値を設定します。（※将来機能です）

5.4.位置指令データポート（InPort）

ダイレクトモーションボタンを押下してください。
ファイルの選択画面が表示されますので、位置指令データ(カンマ区切り)を選択してください。
データの順番は、J1、J2、J3、J4、J5、J6、J7、Gr(グリッパ)です。
※行毎のデータは、getManipInfo オペレーションで取得することができる cmdCycle の周期に対応した位置指令データを準備してください。

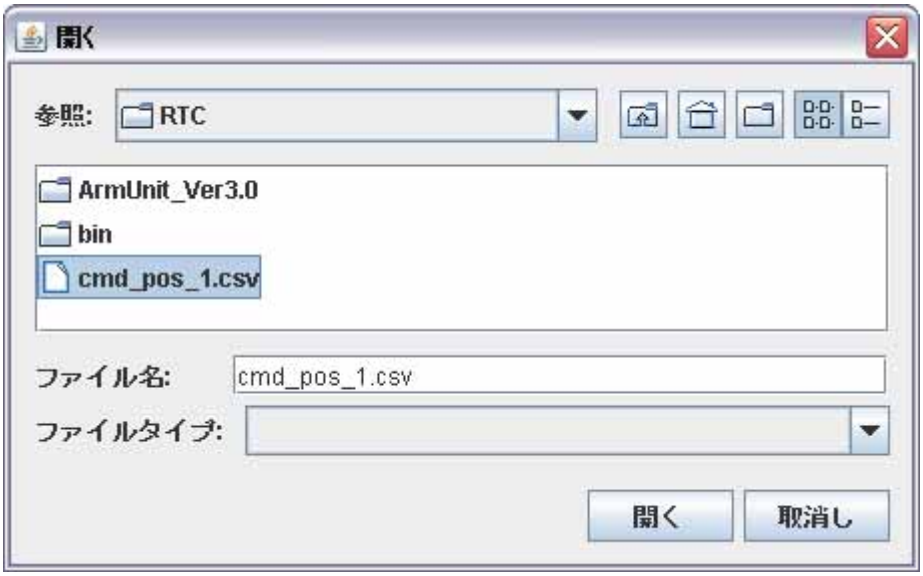


図 25 ファイル選択画面

位置指令データ例

```
0,-20,0,0,0,0,0,1.9
5.00E-04,-20.00044444,3.33E-04,0.001,3.33E-04,2.22E-04,2.22E-04,1.851
0.0015,-20.00133333,0.001,0.003,0.001,6.67E-04,6.67E-04,1.802
0.003,-20.00266667,0.002,0.006,0.002,0.001333333,0.001333333,1.753
0.005,-20.00444444,0.003333333,0.01,0.003333333,0.002222222,0.002222222,1.704
0.0075,-20.00666667,0.005,0.015,0.005,0.003333333,0.003333333,1.655
0.0105,-20.00933333,0.007,0.021,0.007,0.004666667,0.004666667,1.606
0.014,-20.01244444,0.009333333,0.028,0.009333333,0.006222222,0.006222222,1.557
0.018,-20.016,0.012,0.036,0.012,0.008,0.008,1.508
0.0225,-20.02,0.015,0.045,0.015,0.01,0.01,1.459
0.0275,-20.02444444,0.018333333,0.055,0.018333333,0.012222222,0.012222222,1.41
0.033,-20.02933333,0.022,0.066,0.022,0.014666667,0.014666667,1.361
0.039,-20.03466667,0.026,0.078,0.026,0.017333333,0.017333333,1.312
0.0455,-20.04044444,0.030333333,0.091,0.030333333,0.020222222,0.020222222,1.263
                                        .
                                        .
```

5.5.位置 FB 指令データポート（OutPort）

マニピュレータの各関節のフィードバック角度データを取得できます。
取得した値は、取得情報表示エリアに表示されます。(図 26)
データの順番は、J1、J2、J3、J4、J5、J6、J7、Gr(グリッパ)です。

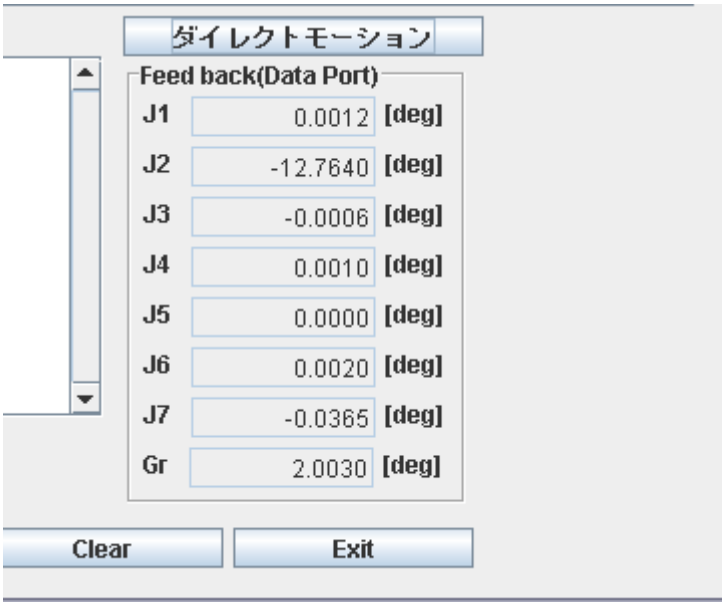


図 26 位置 FB 指令データ

5.6.操作手順

アームユニット RTC の簡単な操作手順を記します。

①アーム各軸の主回路電源、駆動用モータのサーボ制御を入れます。

servoOn ボタンを押下してください。

②関節座標系における直線補間時の各軸の加速時間を設定します。

setAccelTimeJoint ボタンを押下してください。

下記画面が表示されますので、任意の値を入力し、OK ボタンを押下してください。

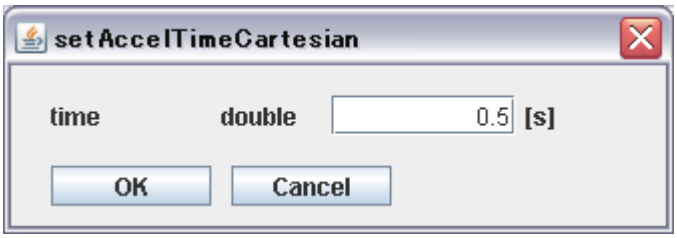
※小さい値を設定する場合は、アームユニットの動作速度が速くなりますので、ご注意ください。

③直交座標系における直線補間時の各軸の加速時間を設定します。

setAccelTimeCartesian ボタンを押下してください。

下記画面が表示されますので、任意の値を入力し、OK ボタンを押下してください。

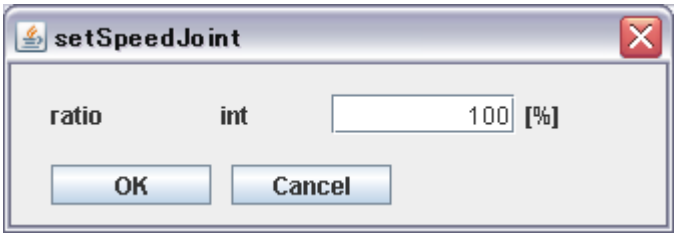
※小さい値を設定する場合は、アームユニットの動作速度が速くなりますので、ご注意ください。



④関節動作時の速度を[%]設定します。

SetSpeedJoint ボタンを押下してください。

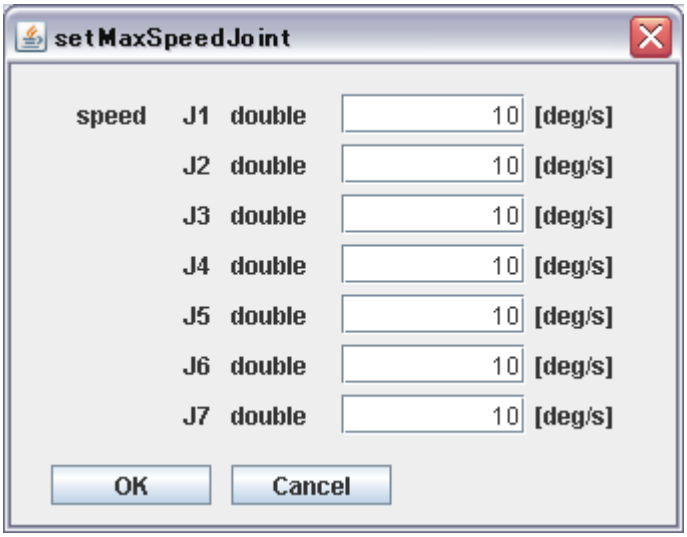
下記画面が表示されますので、任意の値を入力し、OK ボタンを押下してください。



⑤関節動作時の最大動作速度を設定します。

SetMaxSpeedJoint ボタンを押下してください。

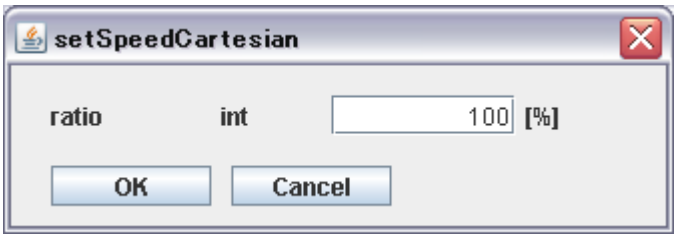
下記画面が表示されますので、任意の値を入力し、OK ボタンを押下してください。



⑥直交動作時の速度を[%]設定します。

SetSpeedCartesian ボタンを押下してください。

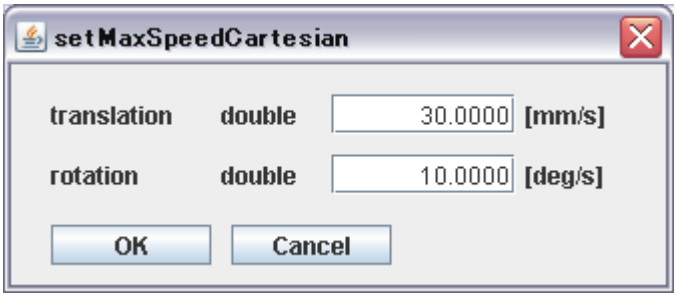
下記画面が表示されますので、任意の値を入力し、OK ボタンを押下してください。



⑦直交動作時の最大動作速度を設定します。

SetMaxSpeedCartesian ボタンを押下してください。

下記画面が表示されますので、任意の値を入力し、OK ボタンを押下してください。

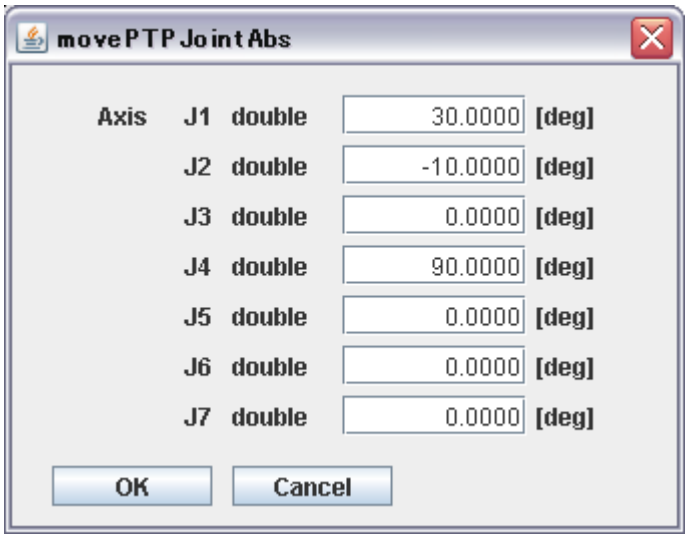


⑧アームユニットを関節座標系の直線補間で動かします。

movePTPJointAbs ボタンを押下してください。

下記画面が表示されますので、任意の値を入力し、OK ボタンを押下してください。下記の例では肘を曲げ、前方に振り上げた姿勢をとります。

※可動範囲を考慮して指定してください。



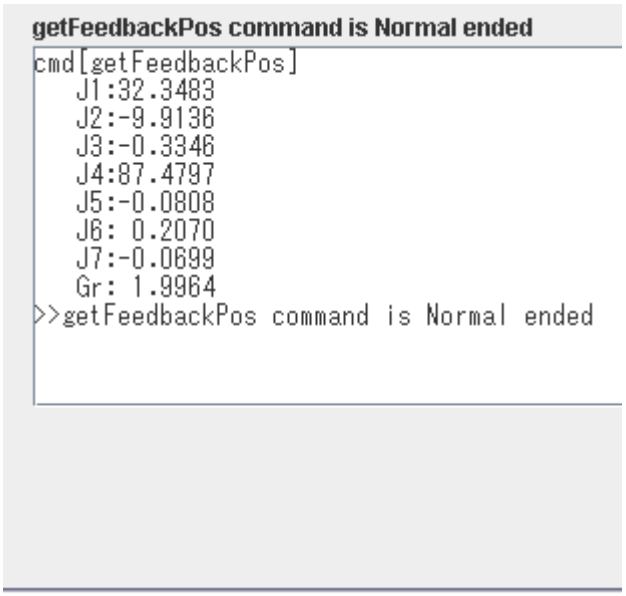
⑨アームユニットを直交座標系の直線補間で動かします。

moveLinearCartesianRel ボタンを押下します。

任意の値を入力し、OK ボタンを押下してください。

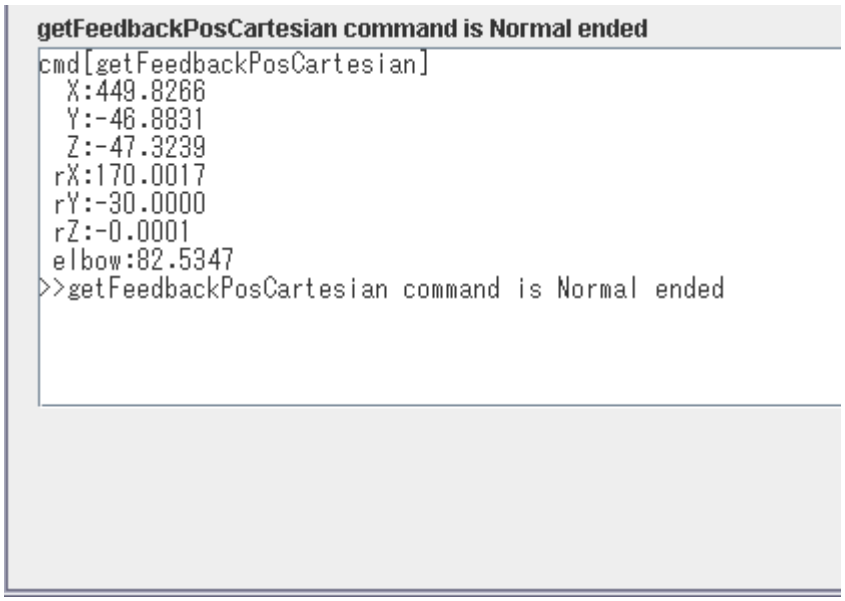
⑩アームユニットの各軸の現在位置を取得します。

getFeedbackPos ボタンを押下します。取得情報表示エリアに各関節の現在位置[degree]が表示されます。



⑪アームユニットの直交座標系での現在位置を取得します。

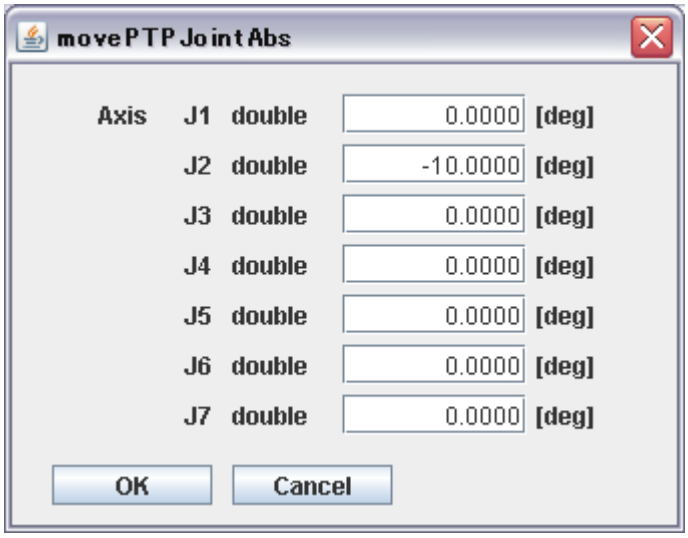
getFeedbackPosCartesian ボタンを押下します。取得情報表示エリアに各直交軸の現在位置[mm、degree]が表示されます。



⑫アームユニットを関節座標系の直線補間で動かします。

movePTPJointAbs ボタンを押下してください。

下記画面が表示されますので、J2 に-10[degree]、他は 0[degree]を入力し、OK ボタンを押下してください。アームを下ろした姿勢をとります。



⑬アーム各軸の主回路電源、駆動用モータのサーボ制御を切ります。

servoOff ボタンを押下してください。

6.付録

アームユニット RTC のサービスポート定義を表 5 に、データポート定義を表 6 に示します。

表 5 アームユニット RTC のサービスポート定義

ポートタイプ	ポート名	インタフェース名	タイプ名
サービスポート(共通コマンド)	rightarmService0.ManipulatorCommonInterface_CommonProvider	rightarmService_Common	ManipulatorCommonInterface_Common
サービスポート(中レベルモーションコマンド)	rightarmService0.ManipulatorCommonInterface_MiddleProvider	rightarmService_Middle	ManipulatorCommonInterface_Middle

表 6 アームユニット RTC のデータポート定義

ポートタイプ	ポート名	型
データポート(InPort)	arm.cmdPos	TimedJointPos
データポート(OutPort)	arm.fbPos	TimedJointPos

アームユニット RTC の IDL（ArmUnit.idl）

```
#ifndef ARMUNITRTC_IDL_
#define ARMUNITRTC_IDL_

module RTC
{
  typedef sequence<double> DoubleSeq;
  typedef sequence<double> JointPos;
  struct LimitValue {
    double upper;
    double lower;
  };
  struct RETURN_ID
  {
    long id;                                  /**< エラーID                                */
    string comment;                           /**< エラーコト                              */
  };
  struct TimedJointPos {
    Time tm;
    JointPos pos;
  };
  typedef unsigned long ULONG;
};
#endif

// -*- IDL -*-
/*!
 * @file DataType.idl
 * @brief Basic Data Type definition
 * @date $Date: 2007-01-09 15:36:29 $
 * @author Noriaki Ando <n-ando@aist.go.jp>
 *
 * Copyright (C) 2003-2006
 *     Task-intelligence Research Group,
 *     Intelligent Systems Research Institute,
 *     National Institute of
 *         Advanced Industrial Science and Technology (AIST), Japan
 *     All rights reserved.
 *
 * $Id: BasicDataType.idl 775 2008-07-28 16:14:45Z n-ando $
```

```
 *
 */

#ifndef BasicDataType_idl
#define BasicDataType_idl

#endif // end of BasicDataType_idl

/*
Manipulator Common Interface (Common Commands)
  - This IDL is used as service port on RTC
    - This command specification is provided by Intelligent RT Software
          Project of NEDO.
rev. 20100502
*/

#ifndef MANIPULATORCOMMONINTERFACE_COMMON_IDL
#define MANIPULATORCOMMONINTERFACE_COMMON_IDL

module RTC
{
  enum AlarmType {
    FAULT,
    WARNING,
    UNKNOWN
  };

  struct Alarm {
    unsigned long code;
    AlarmType type;
    string description;
  };

  typedef sequence<Alarm> AlarmSeq;
  typedef sequence<LimitValue> LimitSeq;

  struct ManipInfo {
    string manufactur;
    string type;
    ULONG  axisNum;
    ULONG  cmdCycle;
    boolean isGripper;
```

```
  };

  const ULONG CONST_BINARY_00000001 = 0x01;          /* isServoOn      */
  const ULONG CONST_BINARY_00000010 = 0x02;          /* isMoving       */
  const ULONG CONST_BINARY_00000100 = 0x04;          /* isAlarmed      */
  const ULONG CONST_BINARY_00001000 = 0x08;          /* isBufferFull */
};

interface ManipulatorCommonInterface_Common
{
  RTC::RETURN_ID clearAlarms();
  RTC::RETURN_ID getActiveAlarm(out RTC::AlarmSeq alarms);
  RTC::RETURN_ID getFeedbackPosJoint(out RTC::JointPos pos);
  RTC::RETURN_ID getManipInfo(out RTC::ManipInfo manipInfo);
  RTC::RETURN_ID getSoftLimitJoint(out RTC::LimitSeq softLimit);
  RTC::RETURN_ID getState(out RTC::ULONG state);
  RTC::RETURN_ID servoOFF();
  RTC::RETURN_ID servoON();
  RTC::RETURN_ID setSoftLimitJoint(in RTC::LimitSeq softLimit);
};

#endif // MANIPULATORCOMMONINTERFACE_COMMON_IDL

/*
Manipulator Common Interface (Data type defenition)
  - This IDL is used as service port on RTC
    - This command specification is provided by Intelligent RT Software
        Project of NEDO.
rev. 20100502
*/

#ifndef MANIPULATORCOMMONINTERFACE_DATATYPES_IDL
#define MANIPULATORCOMMONINTERFACE_DATATYPES_IDL

#endif // MANIPULATORCOMMONINTERFACE_DATATYPES_IDL

/*
Manipulator Common Interface (Middle Level Commands)
  - This IDL is used as service port on RTC
    - This command specification is provided by Intelligent RT Software
        Project of NEDO.
rev. 20100502
```

```
*/

#ifndef MANIPULATORCOMMONINTERFACE_MIDDLE_IDL
#define MANIPULATORCOMMONINTERFACE_MIDDLE_IDL


module RTC
{
  typedef double HgMatrix [3][4];
  struct CarPosWithElbow {
    HgMatrix carPos;
    double   elbow;
    ULONG    structFlag;
  };
  struct CartesianSpeed {
    double translation;
    double rotation;
  };
};

interface ManipulatorCommonInterface_Middle
{
  RTC::RETURN_ID closeGripper();
  RTC::RETURN_ID getBaseOffset(out RTC::HgMatrix offset);
  RTC::RETURN_ID getFeedbackPosCartesian(out RTC::CarPosWithElbow pos);
  RTC::RETURN_ID getMaxSpeedCartesian(out RTC::CartesianSpeed speed);
  RTC::RETURN_ID getMaxSpeedJoint(out RTC::DoubleSeq speed);
  RTC::RETURN_ID getMinAccelTimeCartesian(out double aclTime);
  RTC::RETURN_ID getMinAccelTimeJoint(out double aclTime);
  RTC::RETURN_ID getSoftLimitCartesian(out RTC::LimitValue xLimit,
                                       out RTC::LimitValue yLimit,
                                       out RTC::LimitValue zLimit );
  RTC::RETURN_ID moveGripper(in RTC::ULONG angleRatio);
  RTC::RETURN_ID moveLinearCartesianAbs(in RTC::CarPosWithElbow carPoint);
  RTC::RETURN_ID moveLinearCartesianRel(in RTC::CarPosWithElbow carPoint);
  RTC::RETURN_ID movePTPCartesianAbs(in RTC::CarPosWithElbow carPoint);
  RTC::RETURN_ID movePTPCartesianRel(in RTC::CarPosWithElbow carPoint);
  RTC::RETURN_ID movePTPJointAbs(in RTC::JointPos jointPoints);
  RTC::RETURN_ID movePTPJointRel(in RTC::JointPos jointPoints);
  RTC::RETURN_ID openGripper();
  RTC::RETURN_ID pause();
  RTC::RETURN_ID resume();
```

```
    RTC::RETURN_ID stop();
    RTC::RETURN_ID setAccelTimeCartesian(in double aclTime);
    RTC::RETURN_ID setAccelTimeJoint(in double aclTime);
    RTC::RETURN_ID setBaseOffset(in RTC::HgMatrix offset);
    RTC::RETURN_ID setControlPointOffset(in RTC::HgMatrix offset);
    RTC::RETURN_ID setMaxSpeedCartesian(in RTC::CartesianSpeed speed);
    RTC::RETURN_ID setMaxSpeedJoint(in RTC::DoubleSeq speed);
    RTC::RETURN_ID setMinAccelTimeCartesian(in double aclTime);
    RTC::RETURN_ID setMinAccelTimeJoint(in double aclTime);
    RTC::RETURN_ID setSoftLimitCartesian(in RTC::LimitValue xLimit,
                                         in RTC::LimitValue yLimit,
                                         in RTC::LimitValue zLimit);
    RTC::RETURN_ID setSpeedCartesian(in RTC::ULONG spdRatio);
    RTC::RETURN_ID setSpeedJoint(in RTC::ULONG spdRatio);

};

#endif // MANIPULATORCOMMONINTERFACE_MIDDLE_IDL
```