NetwoRC プロバイダ

デンソー ロボット

Version 1.2.19

ユーザーズ ガイド

Obtober 13, 2015

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0	2006-05-11	初版
1.0.1	2006-07-17	@n, ROTATE pose オプション,ST_xxx コマンド対応
1.0.2	2006-08-31	SYSSTATE, 型変換, コンベアトラッキングのコマンドに対応
1.0.3	2006-09-13	POSEDATA 型対応
1.0.4	2006-10-16	トラブル時のチェックリスト追加
1.0.5	2006-12-09	IO 変数で長さ指定オプション追加
1.1.0	2007-07-12	RC5 対応,単軸制御ログコマンド追加
1.1.1	2007-08-10	MyIP オプション追加
1.1.2	2007-11-21	TORetry オプション追加
1.2.0	2008-01-14	付加軸対応
1.2.1	2008-01-21	エラーコード表追加
1.2.2	2008-06-18	高軌跡制御機能モード対応
1.2.3	2008-07-02	マイナーバージョンアップ
1.2.4	2008-08-22	FIGAPRP, FIGAPRL コマンド対応
1.2.5	2008-09-30	一部改訂
1.2.6	2008-11-21	変数番号の動的変更に対応, ローカルポート指定機能追加
1.2.7	2009-06-12	衝突検出機能対応
1.2.8	2009-07-07	Tips 追加, CCLink リモートレジスタ(WDin, WDOut)対応
1.2.9	2009-09-07	変数のタイムスタンプ取得に対応(Microsecond プロパティ)
1.2.10	2010-02-11	ミニ I/O の全汎用化オプション対応,エラーコード追加
1.2.11	2010-09-22	1 章に概要追記
1.2.12	2010-11-17	Execute のコマンド文字列修正(大文字・小文字)
1.2.13	2011-01-11	PAC プログラムの転送 改訂
1.2.14	2011-02-01	RobSlave 必須コマンド明記
1.2.15	2011-04-05	ロボットコントローラーの対応機種修正, セットアップ修正
1.2.16	2011-04-21	無停止教示点補正機能追加
1.2.17	2011-05-16	@ERROR_CODE_HEX 変数追加
1.2.18	2011-05-26 2012-07-17	Bundle 版の同時接続数＝3 を廃止 ドキュメントのバージョンルールを変更
1.2.19	2015-10-13	@ERROR_LEVEL 変数追加

【対応機器】

機種	バージョン	注意事項
RC7	V2.330 以上	要 ORiN オプション(1214)
RC5	V1.998 以上	要 ORiN オプション(1213), ※付録 D を参照

【注意】

デンソーロボットへの同時接続数は最大 79 接続[1]まで可能です.

[1] 接続数＝NetwoRC プロバイダのインスタンス数＝CaoWorkspace::AddController で作成されたオブジェクト数

目次

1. はじめに 8
1.1. 概要 9
2. セットアップ 11
2.1. 非常停止スイッチの設置 11
2.2. コントローラの初期設定 11
2.2.1. ティーチングペンダントを用いた初期設定 11
2.2.2. ミニペンダントを用いた初期設定 14
2.3. 専用 I/O ポートの処置 17
2.3.1. 標準（I/O 増設ボードがない）構成の場合 17
2.3.1.1. ミニ I/O 全汎用化オプション 18
2.3.2. オプション（I/O 増設ボードがある）構成の場合 18
2.4. ロボットコントローラの起動権 19
2.4.1. ロボットコントローラの起動権に関する基礎知識 19
2.4.2. ANSI 仕様ロボットコントローラにおける注意点 20
2.5. PAC プログラムの転送 22
2.6. RobMaster の機能 22
3. NetwoRC プロバイダを使用したプログラミング 24
3.1. 接続 24
3.2. 変数の読み書き 25
3.2.1. 接続 25
3.2.2. 変数リード・ライト 26
3.2.3. 切断 26
3.2.4. サンプルプログラム 27
3.3. PAC プログラムの開始・停止 27
3.3.1. 接続 28
3.3.2. PAC プログラムの開始・停止 28
3.3.3. サンプルプログラム 28
3.4. ロボットの動作 29
3.4.1. 接続 30
3.4.2. ロボットの移動と停止 30
3.5. その他サンプル 31

4. コマンドリファレンス 32
4.1. コマンド一覧 32
4.2. メソッド・プロパティ 33
4.2.1. CaoWorkspace::AddController メソッド 33
4.2.1.1. Conn オプション 34
4.2.2. CaoController::AddCommand メソッド 35
4.2.3. CaoController::AddFile メソッド 35
4.2.4. CaoController::AddRobot メソッド 36
4.2.5. CaoController::AddTask メソッド 37
4.2.6. CaoController::AddVariable メソッド 37
4.2.7. CaoController::get_TaskNames プロパティ 38
4.2.8. CaoController::get_VariableNames プロパティ 38
4.2.9. CaoController::Execute メソッド 38
4.2.10. CaoController::OnMessage イベント 41
4.2.11. CaoCommand::Execute メソッド 42
4.2.12. CaoCommand::get_Parameters プロパティ 42
4.2.13. CaoCommand::put_Parameters プロパティ 42
4.2.14. CaoFile::AddFile メソッド 42
4.2.15. CaoFile::AddVariable メソッド 43
4.2.16. CaoFile::get_VariableNames プロパティ 43
4.2.17. CaoFile::Copy メソッド 43
4.2.18. CaoFile::Delete メソッド 43
4.2.19. CaoFile::Move メソッド 43
4.2.20. CaoFile::get_DateCreated プロパティ 43
4.2.21. CaoFile::get_DateLastAccessed プロパティ 43
4.2.22. CaoFile::get_DateLastModified プロパティ 43
4.2.23. CaoFile::get_FileNames プロパティ 43
4.2.24. CaoFile::get_Attribute プロパティ 43
4.2.25. CaoFile::get_Path プロパティ 44
4.2.26. CaoFile::get_Size プロパティ 44
4.2.27. CaoFile::get_Type プロパティ 44
4.2.28. CaoFile::get_Value プロパティ 44
4.2.29. CaoFile::put_Value プロパティ 44
4.2.30. CaoRobot::Accelerate メソッド 44
4.2.31. CaoRobot::AddVariable メソッド 44
4.2.32. CaoRobot::get_VariableNames プロパティ 44

4.2.33. CaoRobot::Halt メソッド 44
4.2.34. CaoRobot::Change メソッド 45
4.2.35. CaoRobot::Drive メソッド 45
4.2.36. CaoRobot::Move メソッド 45
4.2.37. CaoRobot::Rotate メソッド 49
4.2.38. CaoRobot::Speed メソッド 50
4.2.39. CaoRobot::Execute メソッド 50
4.2.40. CaoTask::AddVariable メソッド 67
4.2.41. CaoTask::get_VariableNames プロパティ 67
4.2.42. CaoTask::Start メソッド 67
4.2.43. CaoTask::Stop メソッド 67
4.2.44. CaoVariable::get_Value プロパティ 68
4.2.45. CaoVariable::put_Value プロパティ 68
4.2.46. CaoVariable::put_ID プロパティ 68
4.2.47. CaoVariable::get_ID プロパティ 68
4.2.48. CaoVariable::get_Microsecond プロパティ 68
4.2.49. CaoMessage::Clear メソッド 68
4.3. 変数一覧 69
4.3.1. コントローラクラス 69
4.3.2. ロボットクラス 71
4.3.3. タスククラス 73
4.3.4. ファイルクラス 74
5. ロボット動作命令の実行概要 75
6. Tips 77
6.1. コントローラエラー発生時の書き込み 77
6.1.1. 特権タスク機能を有効にする 78
6.1.2. 特権タスクプログラムを作成する 79
6.1.3. PC から特権タスクに対して通知する 80
付録 A. POSEDATA 型定義 81
付録 A.1. 表記例 82
付録 B. PAC 言語対応状況 86
付録 C. トラブル時の確認事項 89
付録 C.2. ロボットコントローラと通信できない 89

付録 C.3. ロボットコントローラの変数を読み書きできない 89
付録 C.4. ロボットの制御ができない 90

付録 D. コントローラ別対応状況 91

付録 E. エラーコード一覧 96

付録 F. 無停止教示点補正機能(外観検査軌道生成) 98

付録 F.1. パラメータ 98
付録 F.2. エラーコード 98
付録 F.3. 制限事項 99
付録 F.4. サンプルプログラム 100
付録 F.5. 軌道補正失敗時(エラーコード:0x800123xx)の回避方法 101

1. はじめに

本仕様書は，デンソーロボットの NetwoRC コントローラ(RC5,RC7)用の CAO プロバイダの外部仕様を規定するものです．本書で扱う CAO プロバイダ(CaoProvNetwoRC.dll)を NetwoRC プロバイダと呼びます．NetwoRC プロバイダは，CAO プロバイダ仕様で規定されている全てのインタフェースを実装しています．

本仕様書では，NetwoRC プロバイダの接続パラメータ，システム変数，ユーザ変数，ファイルおよび独自拡張に関する仕様を記載しています．

また，本仕様書で記載される機能は NetwoRC コントローラの機種およびバージョンにより依存するため，次表の通りに記号として本書に表記しています．

表 1-1 NetwoRC コントローラ別対応記号

コントローラ		記号	意味
機種	バージョン		
RC7M	2.330 以上	2.330	コントローラバージョンが 2.330 以上の場合，使用可能です．
-	-	RobSlave	コマンドを実行するには RobSlave の実行が必要です．

全グローバル変数(I,F,D,V,P,J,T,S)の[0]から[9]まではシステム予約です．
これらの変数へのアクセスは行わないでください．

1.1. 概要

NetwoRC プロバイダは，ロボットコントローラ(RC5，RC7)に依存する部分を吸収し CAO プロバイダ・インタフェース仕様で規定された機能を提供する CAO プロバイダです．そのファイル形式は DLL(Dynamic Link Library)であり，CAO エンジンから使用時に動的にロードされます．NetwoRC プロバイダを使用するにあたっては下表を参照して手作業でレジストリ登録を行う必要があります．

表 1-2 NetwoRC プロバイダ

ファイル名	CaoProvNetwoRC.dll
ProgID	CaoProv.DENSO.NetwoRC
レジストリ登録[2]	regsvr32 CaoProvNetwoRC.dll
レジストリ登録の抹消	regsvr32 /u CaoProvNetwoRC.dll

CAO エンジンが動作するには予め，PC 毎に正規の ORiN2 SDK ライセンスが 1 つ登録されていなくてはなりません．これに関しては ORiN2 SDK ユーザーズガイドの「ライセンスの追加と削除」を参照してください．

NetwoRC プロバイダは RS232C，Ethernet デバイスで接続されているロボットコントローラに接続し，ロボットコントローラの変数の読み書き，PAC プログラムの起動，ロボットの動作等を行うプロバイダです．ロボットコントローラに接続する手段としては，図 1-1 にある通信手段が用意されています．本プロバイダは赤枠で囲っている通信手段を提供しています．

[2] ORiN SDK でインストールした場合は手動で登録／抹消する必要はありません．

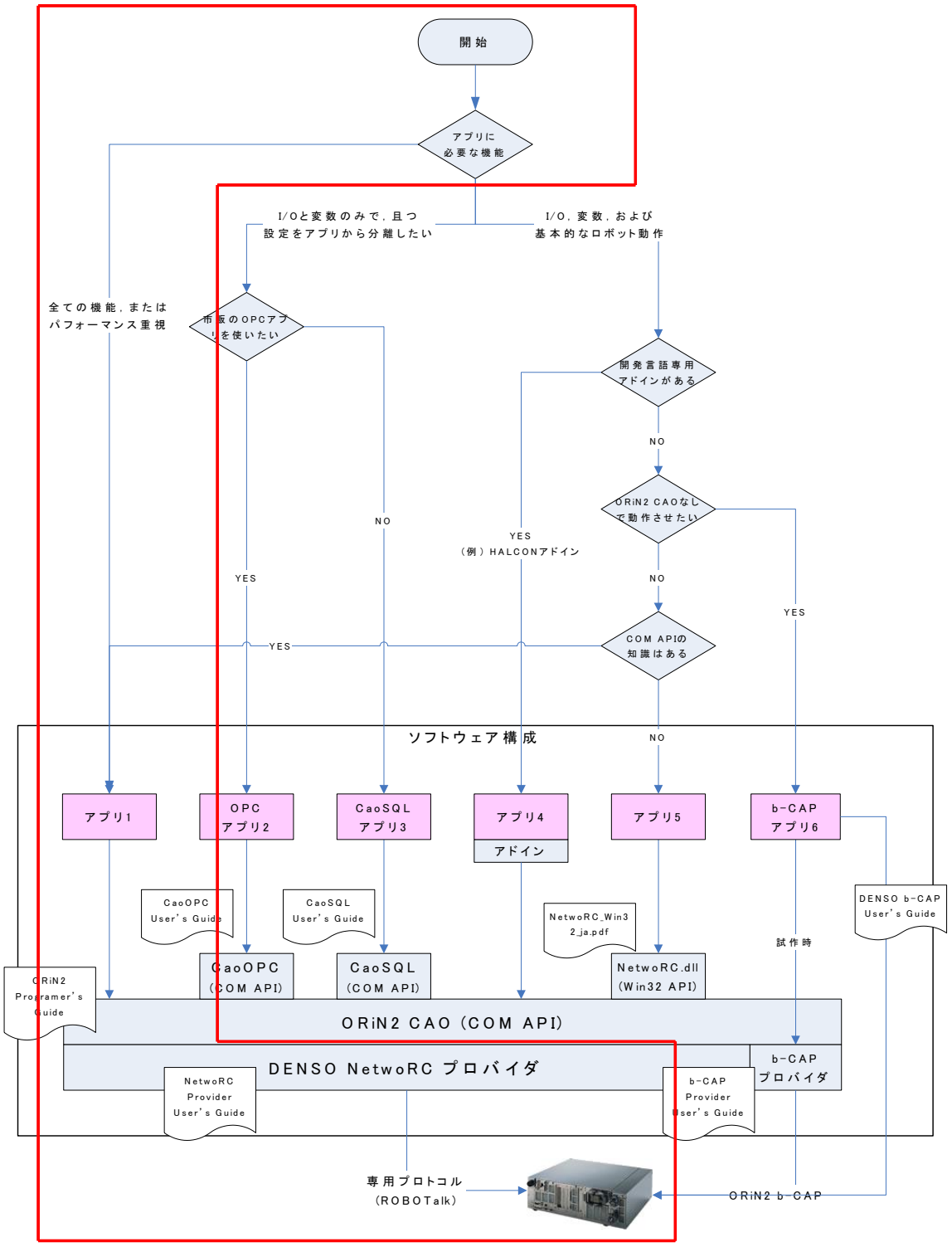


図 1-1 デンソーロボット用 ORiN アプリの分類

2. セットアップ

2.1. 非常停止スイッチの設置

ロボットコントローラを使用になる前に，非常の際にただちにロボットの運転を停止できるよう，作業者が容易に操作できる位置に非常停止スイッチを設置してください．

(1) 非常停止スイッチは，赤色にしてください．

(2) 非常停止の機能は，作動させたあと自動的に復帰せず，また他の作業者が不用意に復帰させることができないようにしてください．

(3) 非常停止スイッチは，電源スイッチとは別個に設けてください．

2.2. コントローラの初期設定

NetwoRC プロバイダを用いる前に，制御対象となるロボットコントローラの設定を行う必要があります．ロボットコントローラの初期設定にはティーチングペンダントもしくはミニペンダントのどちらかが必要となります．以降，それぞれを使った初期設定の方法について記述します．

2.2.1. ティーチングペンダントを用いた初期設定

ティーチングペンダントを用いたロボットコントローラの初期設定は，以下の手順で行います．

(1) ロボットコントローラを手動モードに設定します．

(2) コントローラの通信権限を設定します．接続方法として Ethernet を用いる場合は，ティーチングペンダントの通信設定メニュー => 通信権で[Ethernet]に読み込み・書き込み権限を設定します．
RS232C を用いる場合はそれぞれの COM ポートに読み込み・書き込み権限を設定します．
[F6:設定] =>[F5:通信設定．]=>[F1:通信権．] で設定を行ってください．

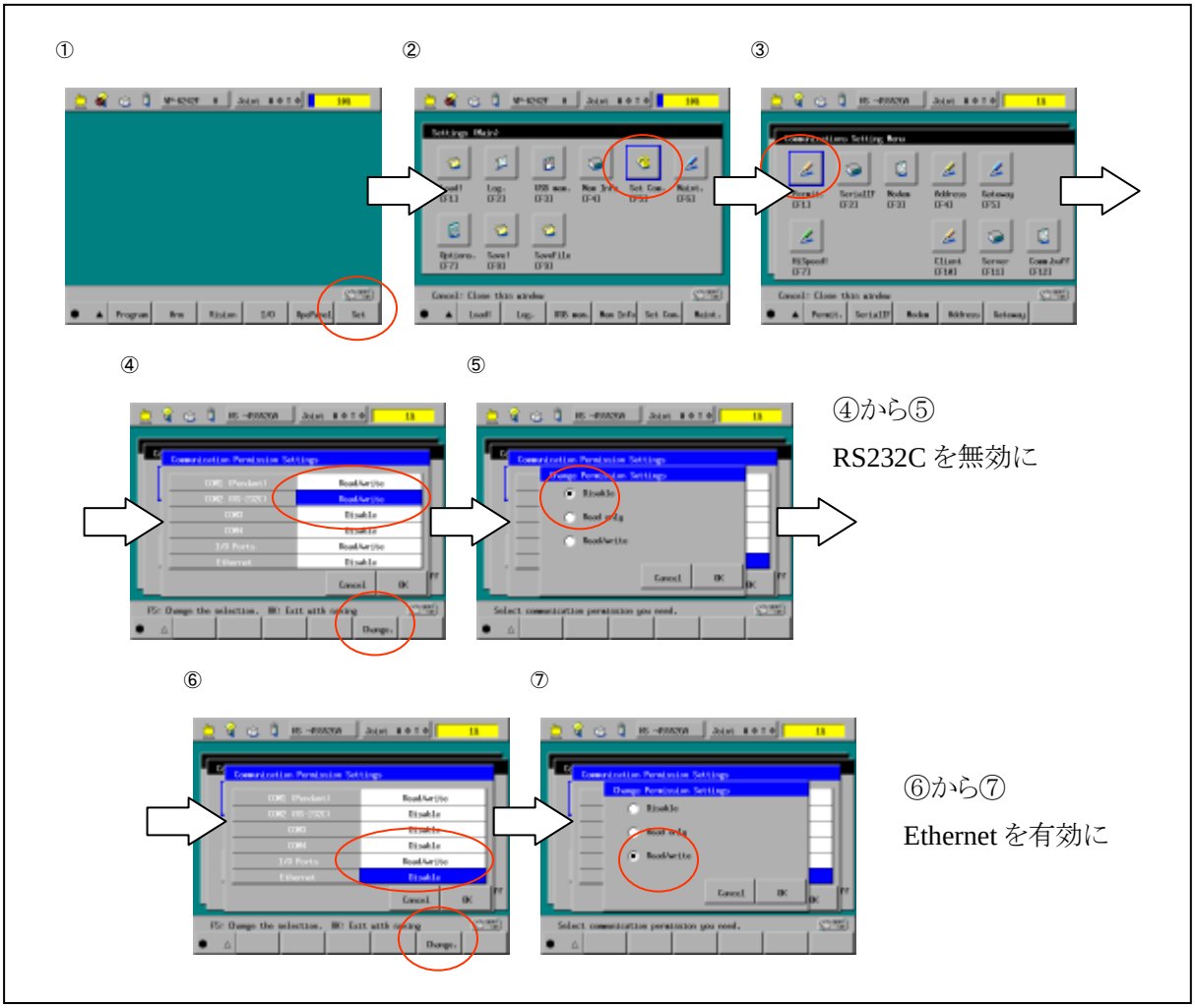


図 2-1 通信権限の設定

(3) ORiN アプリケーションからモータの ON/OFF やプログラムの起動を行う場合には，コントローラの起動権を設定する必要があります．起動権の設定を行うためには，“ORiN“オプションを有効にする必要があります．

[F6:設定] =>[F7:オプション．]=>[F8:機能拡張] =>[F5:機能追加] で“1214“を入力し，“ORiN“オプションを有効にします．ロボットコントローラが RC5 の場合，“1213”を入力して下さい．[3]

[3] この”ORiN”オプションを設定することで，ORiN アプリケーションから変数・I/O へのアクセスが自由に可能となります．変数や I/O へのアクセスはロボット・コントローラ・プログラムなどの状態や内容をよく把握した上で行ってください．特に，変数・I/O への書き込み処理はロボットやプログラムの動作に重大な影響を与える場合があります
また，”ORiN”オプションが有効な状態では，内部自動モードにおいてはエラーレベル 3 以上のエラーが発生するとプログラムが停止しますが，外部自動モードにおいては，エラーレベル 2 以上のエラーが発生するとプログラムが停止します．外部自動モードでの誤操作や誤ったコマンド送信などに注意ください．

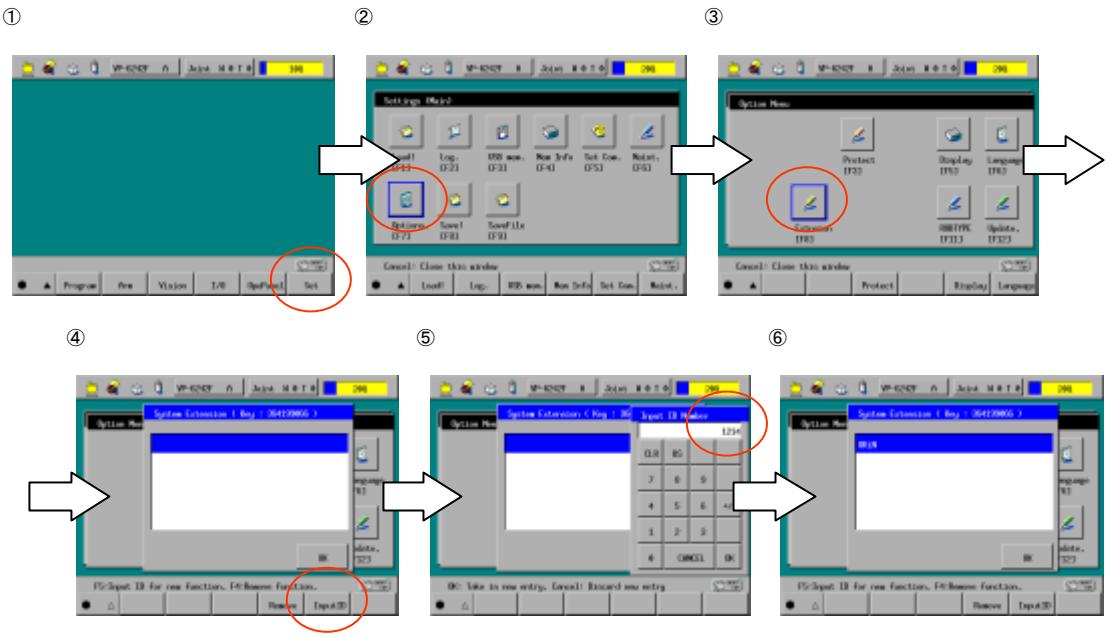


図 2-2 ORiN オプションの入力

(4) 接続方法として Ethernet を用いる場合は，ティーチングペンダントの通信設定メニュー => 起動権で[Ethernet]に起動権を設定し，その後「F4:IP 設定」メニューでクライアントPCの IP アドレスを設定します．RS232C を用いる場合はそれぞれのポートに起動権を設定します．
[F6:設定] =>[F5:通信設定．]=>[F6:起動権]で設定を行ってください．

2.330

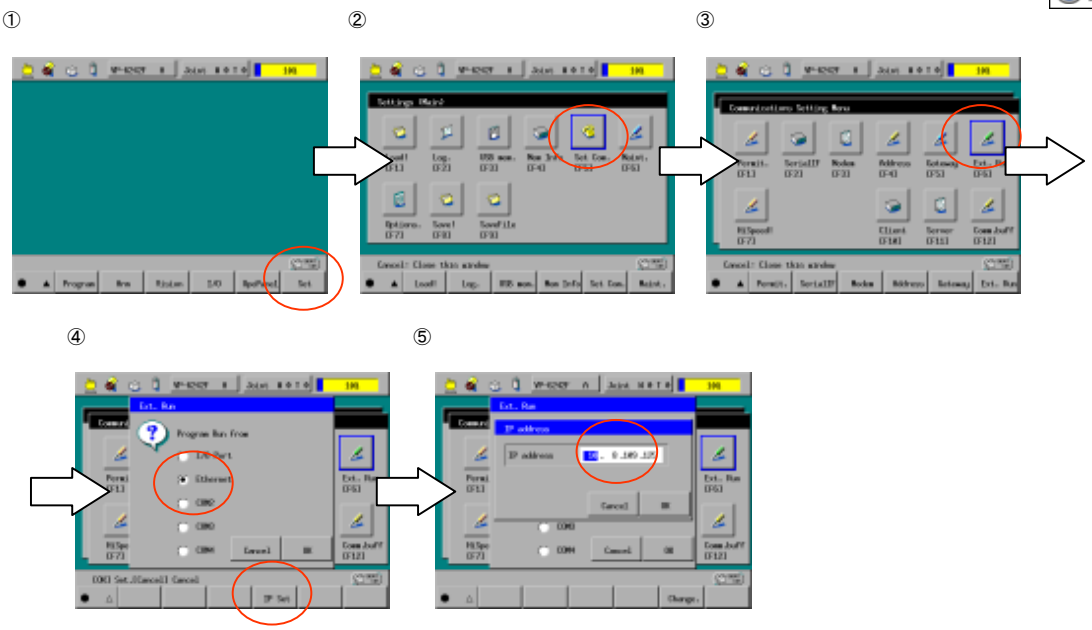


図 2-3 起動権の設定

2.2.2. ミニペンダントを用いた初期設定

2.330

ミニペンダントを用いたコントローラの初期設定は，以下の手順で行います．

(1) ロボットコントローラを手動モードに設定します．

(2) ロボットコントローラの通信権限を設定します．接続方法として Ethernet を用いる場合は，ミニペンダントの[COM Setting] => [Permit]で[Ethernet]に[R/W]を設定します．RS232C を用いる場合はそれぞれのポートに[R/W]を設定します．

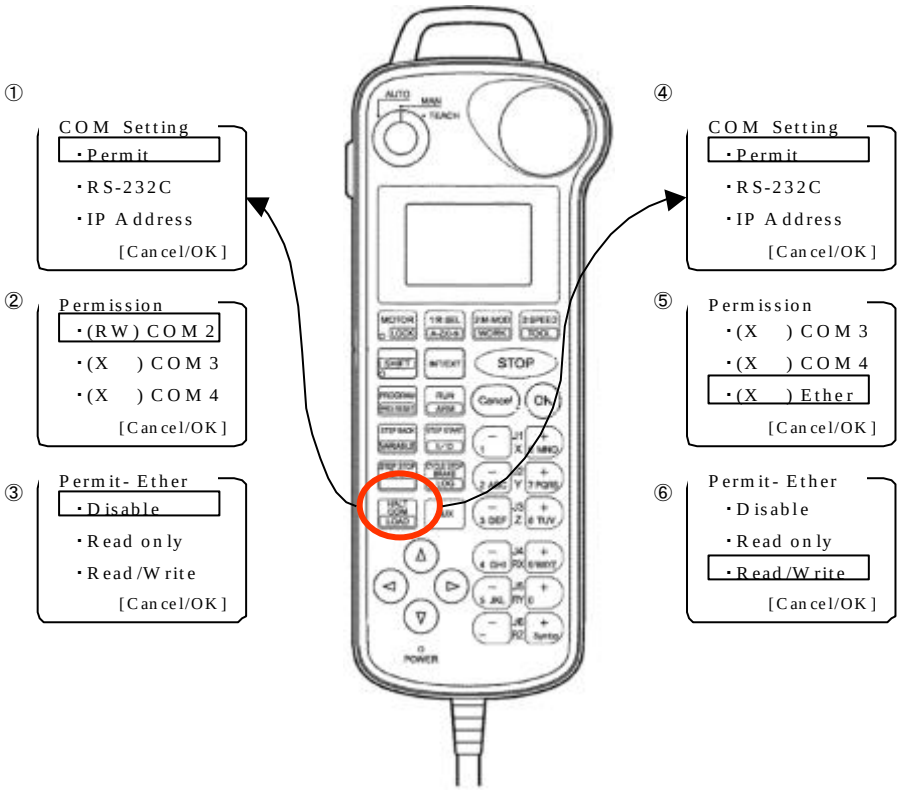


図 2-4 通信権限の設定

(3) ORiN アプリケーションからモータの ON/OFF やプログラムの起動を行う場合には，コントローラの起動権を設定する必要があります．コントローラの起動権を設定するために，ORiN オプションを有効にします．[Aux Function] => [Extension] => [Extension] => [Add]で”1214”を入力し”ORiN”を有効にします．[3]

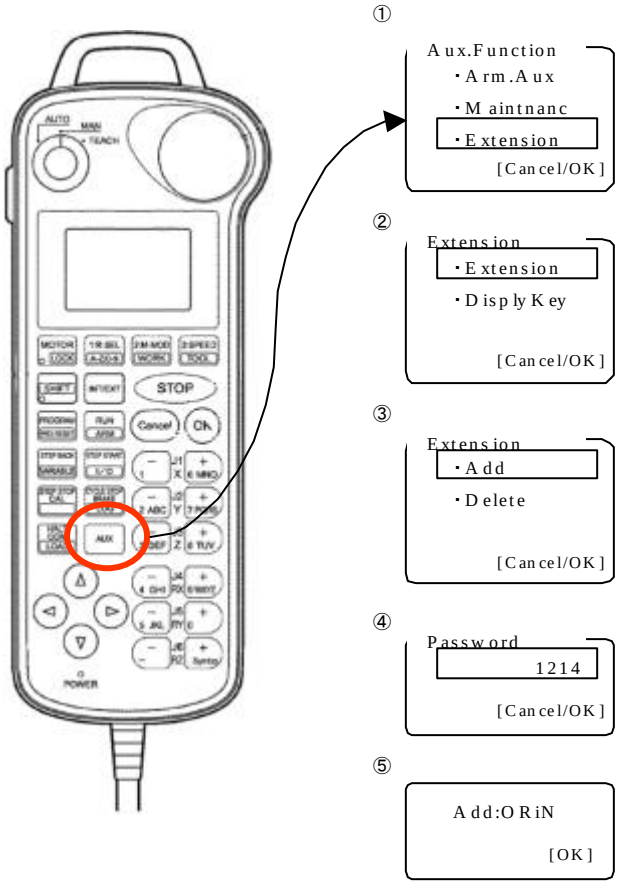


図 2-5 ORiN オプションの入力

(4) 接続方法として Ethernet を用いる場合は, ミニペンダントの[COM Setting] => [Ext Run] => [Ext Run]で[Ethernet]に起動権を設定し, [Client IP]でクライアントPCの IP アドレスを設定します. RS232C を用いる場合はそれぞれの COM ポートに起動権を設定します.

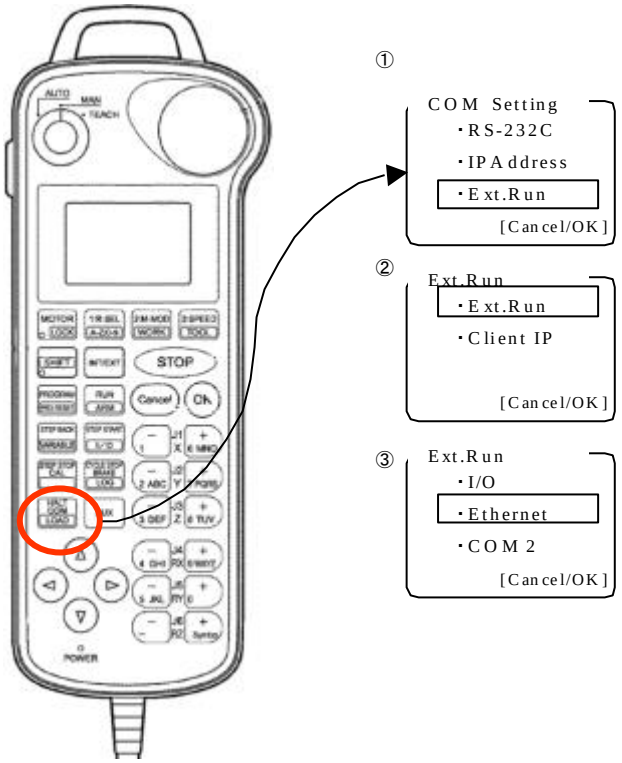


図 2-6 起動権の設定

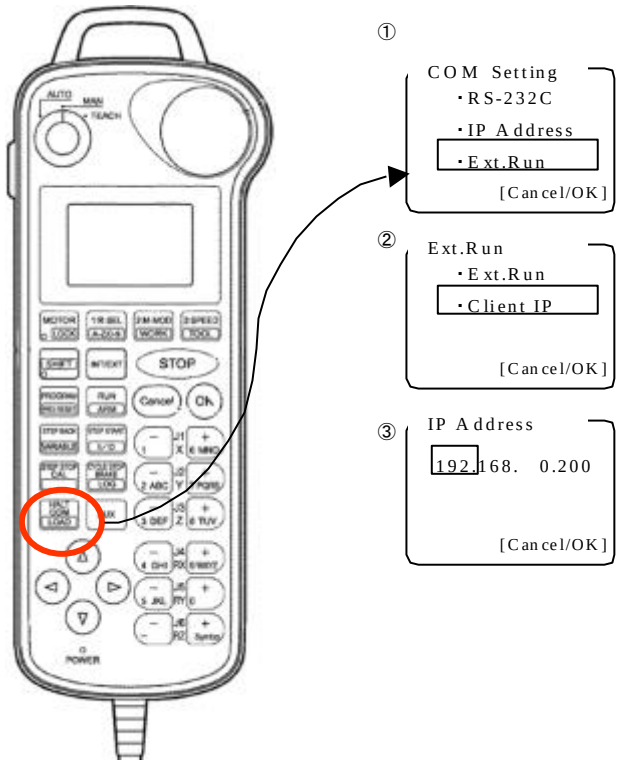


図 2-7 クライアント PC IP アドレスの設定

2.3. 専用 I/O ポートの処置

2.330

ロボットコントローラは多くの専用入力信号を持っています.

外部のPCから ORiN を用いてロボットやプログラム(PAC)を動作させるには,「ステップ停止(全タスク)信号」及び「瞬時停止信号」をプログラムが動作可能な状態にしておく必要があります.

「ステップ停止(全タスク)信号」及び「瞬時停止信号」はロボットコントローラの構成や I/O の割り付けにより配置されるコネクタや I/O 番号が異なりますのでご使用の構成にあわせて適切な処置を行ってください.

※RC5 ではロボット動作が使用出来ないためこの処置は不要です.

※ロボットやプログラム(PAC)の制御を行わない(変数やファイルアクセスのみの)場合, この処置は不要です.

2.3.1. 標準(I/O 増設ボードがない)構成の場合

I/O の標準構成は MiniI/O コネクタ CN5 のみを使用する場合の構成を指します. この場合, 出荷時の設定では入力点数16点の内8点が専用入力, 出力点数16点の内8点が専用出力として割り付けられています.

専用入力の内, Mini I/O:汎用・専用入出力コネクタ CN5 の端子 No.11 にステップ停止(全タスク)信号(ポート番号 0 番)が割り当てられていますので,

Mini I/O:汎用・専用入出力コネクタ CN5 の端子 No.11 をクローズしてください.

⇒ ステップ停止(全タスク)信号(ポート番号 0 番)が ON し,ロボット及びプログラムが動作に可能になります.

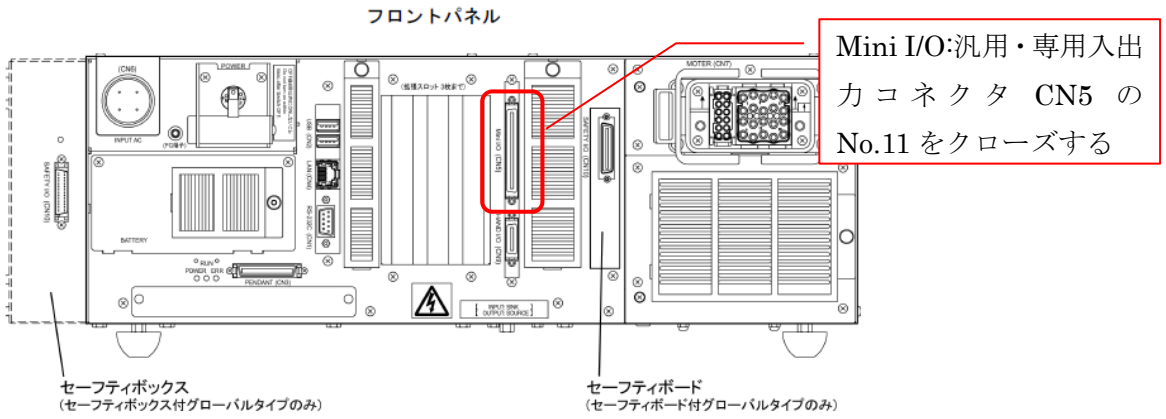


図 2-8 ステップ停止(全タスク)の処置例

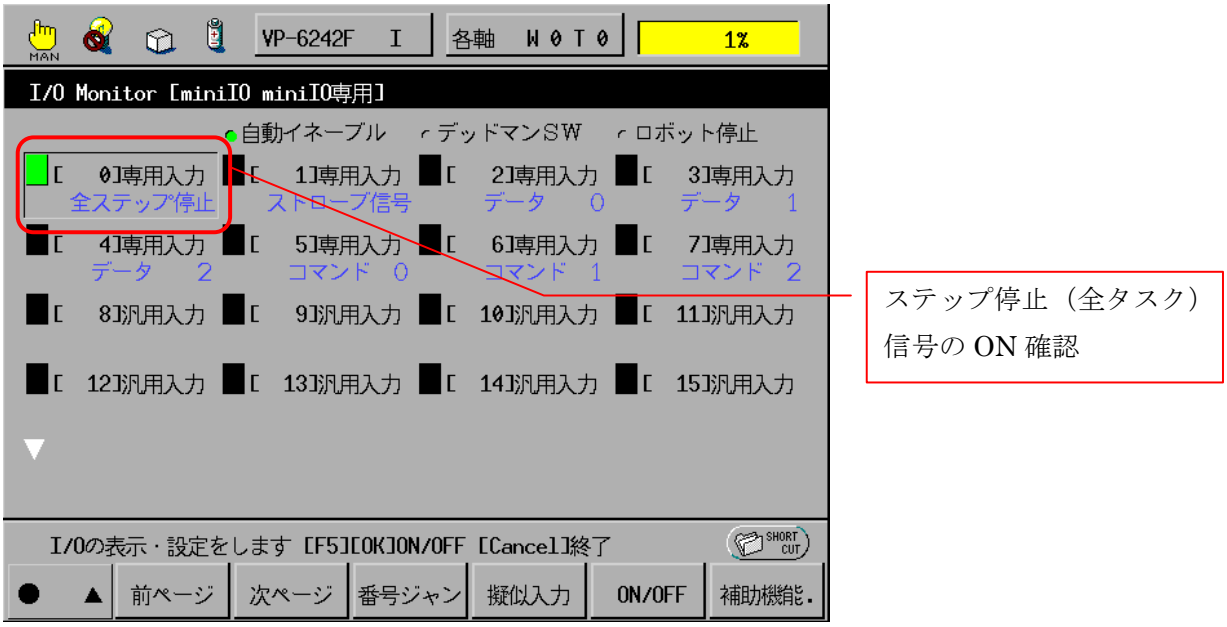


図 2-9 処置後の状態確認

瞬時停止信号はないため処置は不要です.

2.3.1.1. ミニ I/O 全汎用化オプション

ミニ IO に割り付けられている専用入力・出力を必要としない場合でかつロボットコントローラがバージョン 2.90 以上の場合，ミニ I/O の割付を全汎用化して，従来のステップ停止専用信号に対する処置を簡略することが出来ます.

ティーチングペンダントを用いたロボットコントローラの設定は，以下の手順で行います.

(1) ロボットコントローラを手動モードに設定します.
(2) ミニ IO 全汎用化オプションを有効にします. [オプション] => [機能拡張]で"**6319**"を入力します.
(3) ロボットコントローラの電源を切った後，再起動します.

※【注意事項1】ミニ IO 全汎用化オプションを有効にした場合，専用信号であるステップ停止信号の割付は無くなるためステップ停止を行うことが出来なくなります.

※【注意事項 2】ミニ IO 全汎用化オプションを有効にすることで，ステップ停止信号に対する処置は不要になりますが，自動イネーブル信号と非常停止信号に対する処置は必要で，省略することは出来ません.

2.3.2. オプション（I/O 増設ボードがある）構成の場合

I/O 増設ボード（パラレル I/O,DeviceNet,CC-Link,PROFIBUS 等）がある構成の場合はロボットコントローラの「設置・保守ガイド」および「オプション機器説明書」の「第2部 RC7M 用 I/O 増設ボード」を参照の上，「ステップ停止（全タスク）信号」及び「瞬時停止信号」を ON してください.

2.4. ロボットコントローラの起動権

2.330

2.4.1. ロボットコントローラの起動権に関する基礎知識

ORiN アプリケーションからモータの ON/OFF やプログラムの起動を行うには起動権の設定が必要です. (2.2.1 と 2.2.2 を参照)また, 安全のため, ロボットコントローラは選択された一つの機器からのみ外部から制御可能となります(Single point of control). また, ORiN アプリケーションからのモータの ON/OFF やタスクの起動はロボットコントローラが外部自動モードの状態でのみ実行可能となります.

コントローラの起動権は以下の図のように遷移します.

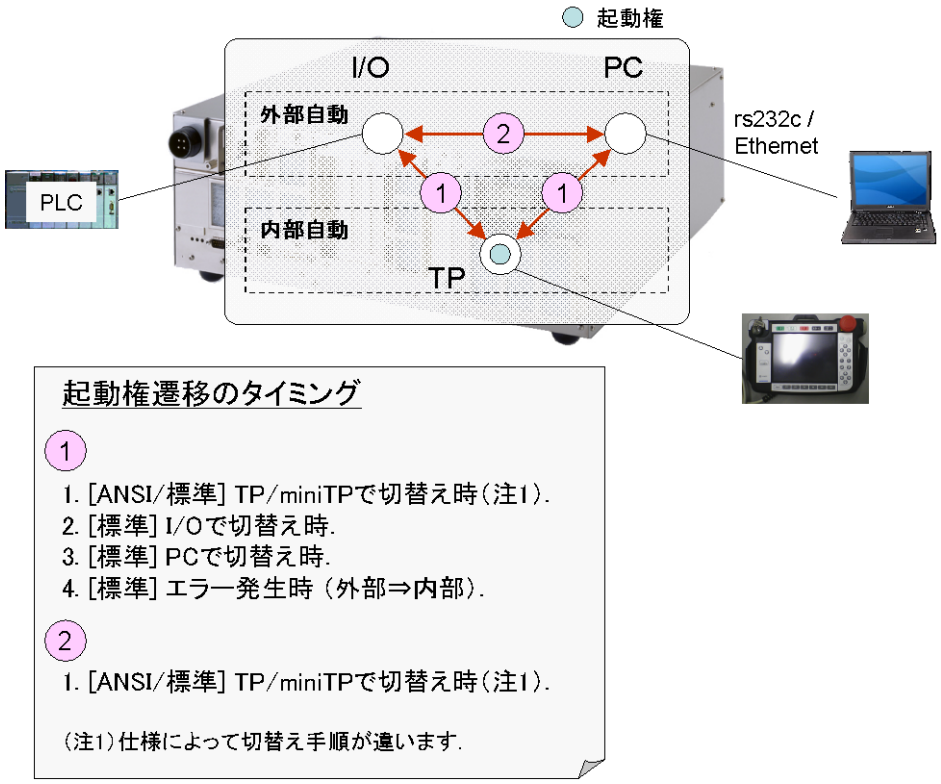


図 2-10 起動権の遷移

2.4.2. ANSI 仕様ロボットコントローラにおける注意点

前述のように，ORiN アプリケーションからのモータの ON/OFF やタスクの起動はロボットコントローラが外部自動モードの状態でのみ実行可能となります．

ANSI 仕様のロボットコントローラでは，I/O 補助機能メニューの単一位置制御設定にて「外部自動」を選択しておかないと，外部自動モードの状態にならないことに注意してください．

以下に ANSI 仕様ロボットコントローラでの外部自動モードの選択手順を示します．

(1) ティーチングペンダントを用いた選択手順

この選択処理が完了した後に自動モードへ切り替えを行うと，選択したモード（内部/外部）に切り替わります．

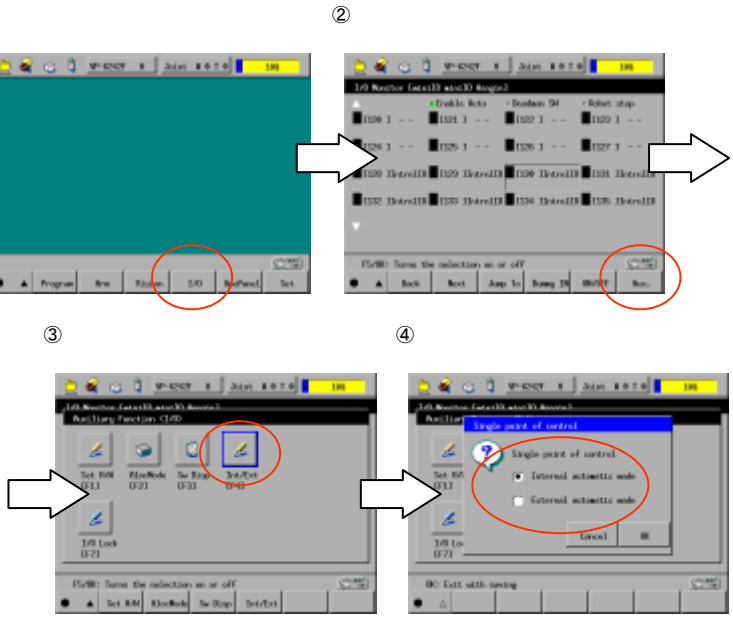


図 2-11 ペンダントを用いた内部/外部自動モード選択

(2) ミニペンダントを用いた選択手順

この選択処理が完了した後に自動モードへ切り替えを行うと，選択したモード（内部/外部）に切り替わります．

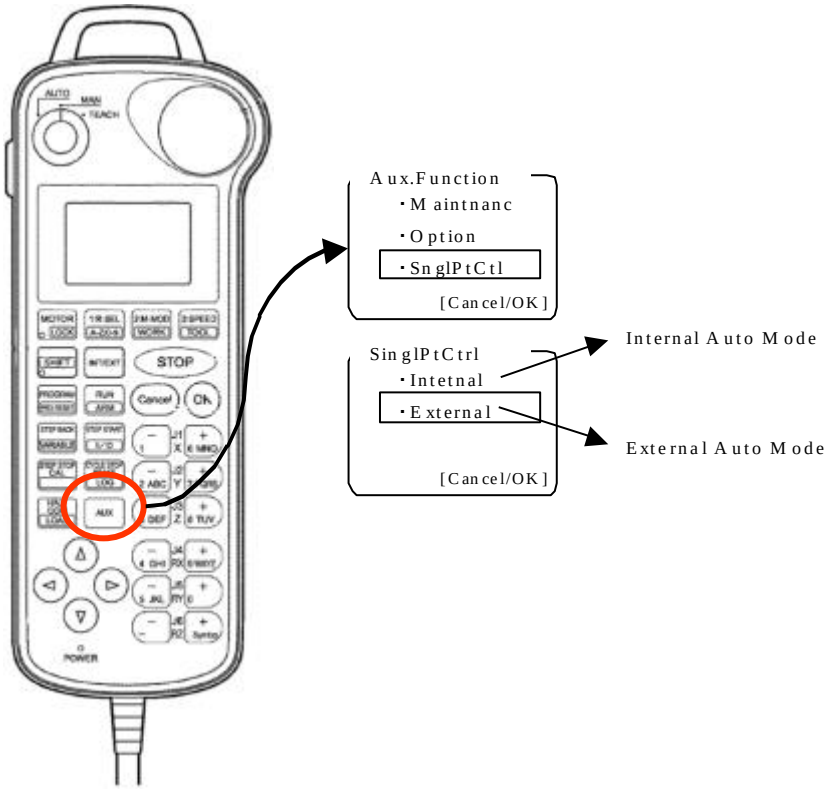


図 2-12 ミニペンダントを用いた内部/外部自動モード選択

2.5. PAC プログラムの転送

2.330

NetwoRC プロバイダでロボット動作命令(表 4-8 参照)を実行するには，予めロボットコントローラに必要な PAC プログラム（RobSlave.pac, RobSlave.h, UserExtention.pac）を転送して実行しなくてはなりません．

PAC プログラムをコントローラへ転送するには，付属ツールの RobMaster.exe を使用します．

これらの方法ではミニペンダント(miniTP)またはティーチングペンダント(TP)が必要です．

RobMaster.exe[4]はロボットコントローラの状態表示と RobSlave タスクの制御を PC から直接と行うためのツールです．RobMaster.exe は以下のフォルダにインストールされています．

<ORiN2 インストールフォルダ>¥CAO¥ProviderLib¥DENSO¥NetwoRC¥Bin

下記にセットアップ手順を示します．

1. コントローラを起動し，[手動]モードにする
2. RobMaster を起動する
   スタートメニュー→すべてのプログラム→ORiN2→CAO→ProviderLib→RobMaster
3. コントローラーの IP アドレスを入力し，［接続］ボタンでコントローラと接続する
4. RobMaster の[セットアップ]ボタンを押して指示に従って必要な PAC プログラムを転送する

2.6. RobMaster の機能

2.330

付属ツール RobMaster はロボットコントローラと接続して，下記の機能を提供しています．

1. ORiN 用にロボットコントローラをセットアップする
2. ロボットコントローラの RobSlave タスクの起動および停止を行う
3. ロボットコントローラのモータ ON/OFF を行う
4. ロボットコントローラのエラー表示とエラークリアを行う
5. ロボットコントローラの状態表示を行う

以下に RobMaster の機能紹介を示します．

[4] <ORiN2 SDK インストールフォルダ>¥CAO¥ProviderLib¥DENSO¥NetwoRC¥Bin¥RobMaster.exe

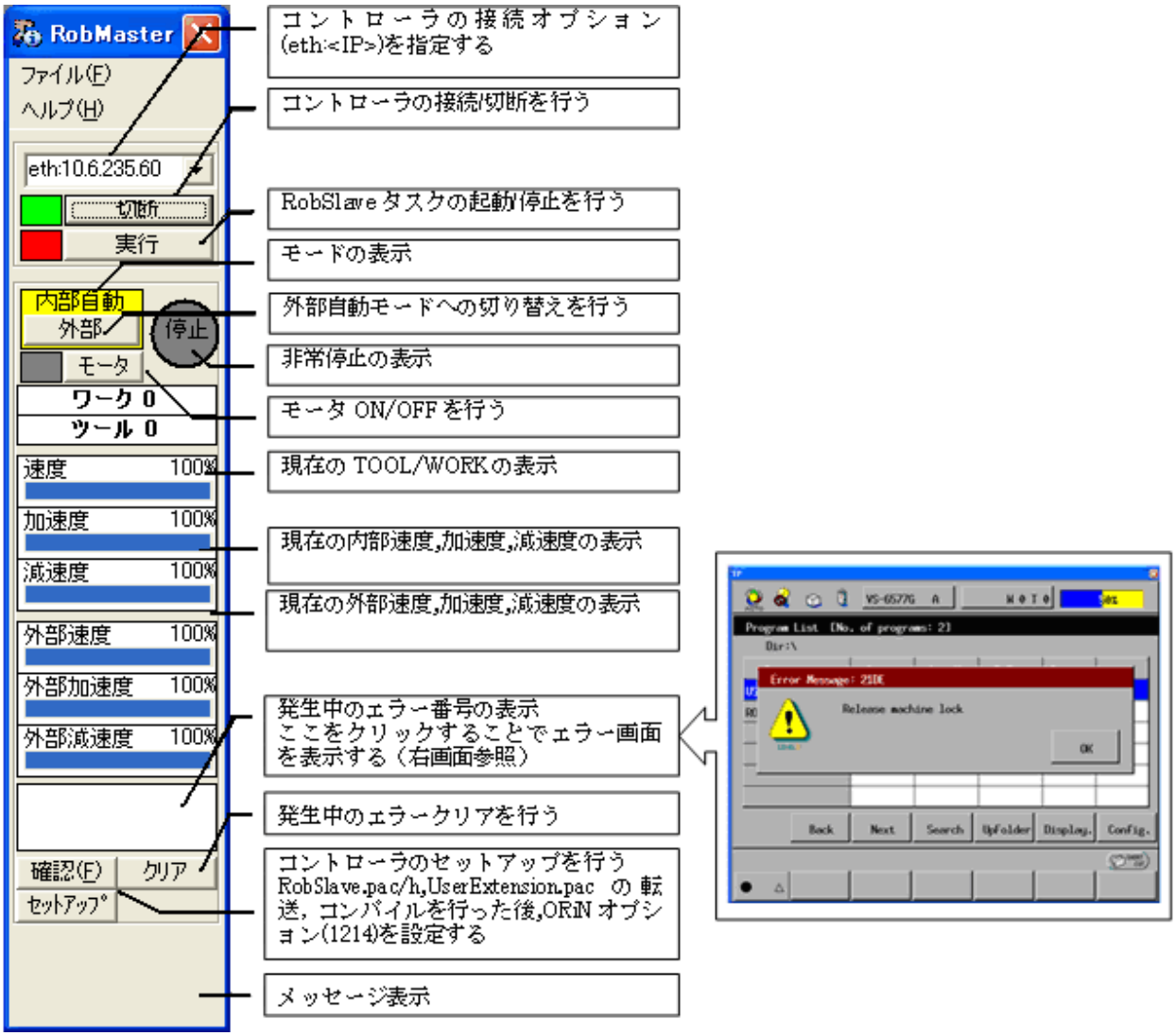


図 2-13 RobMaster の機能紹介

3. NetwoRC プロバイダを使用したプログラミング

3.1. 接続

NetwoRC プロバイダを使用してロボットを制御するには，ORiN がインストールされた PC とロボットコントローラを RC232C，Ethernet デバイスで接続する必要があります．また一部コマンドはロボットコントローラの設定が必要なものもあります．設定の方法は 2 章，各種コマンドについては 4 章を参照してください．

図 3-1 ロボットとの接続

作成するアプリケーションがロボットコントローラと通信するためには，NetwoRC プロバイダを使用し，専用パケット(RobTalk)を生成して，ロボットコントローラと通信を行います．またコントローラの専用プログラム(RobSlave)を介してハンドシェイクを行う事で，ロボットを動作させます．詳しくは 5 章を参照してください．

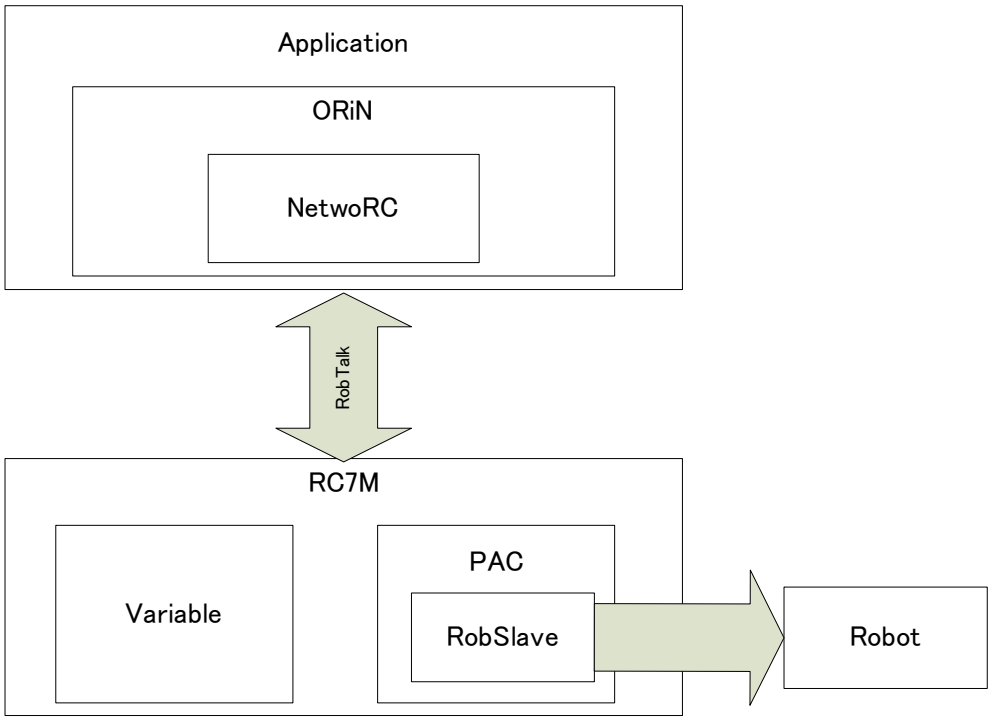


図 3-2 プログラミング概要

NetwoRC プロバイダでは，CaoEngine の作成→CaoWorkspace の作成→CaoController の作成といった手順で Cao オブジェクトを作成する事により，PC とロボットコントローラを接続する事ができます．ロボットコントローラに接続した後は，CaoVariable オブジェクトを生成することで，コントローラ内の変数にアクセスすることや，CaoRobot オブジェクトを生成することでロボットを動作させることができます．以下に具体例を挙げながら，ロボットプログラムの手順を示します．

3.2. 変数の読み書き

変数の読み書きを行うには，図 3-3 に示す処理を行います．

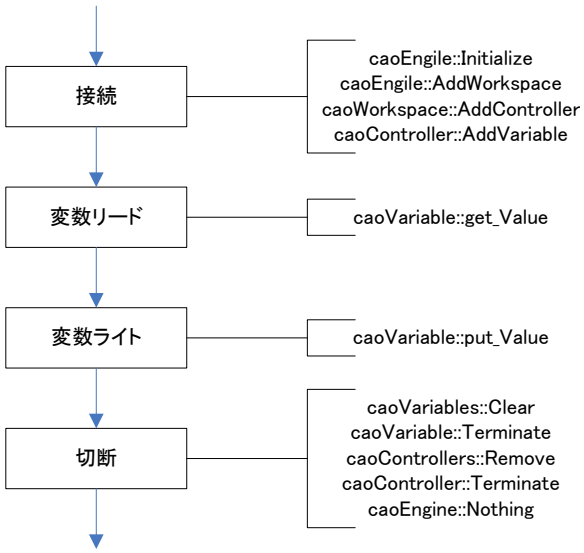


図 3-3 変数の読み書きの流れ

3.2.1. 接続

コントローラとの接続を行うには，以下の手順を取ります．

(1) オブジェクトを保持するための変数を用意します．コントローラ接続に必要なオブジェクトは，CaoEngine オブジェクトと CaoWorkspace オブジェクトと CaoController オブジェクトです．CaoWorkSpace オブジェクトは，CaoWorkspaces から取得する場合には変数を用意する必要はありません．また変数にアクセスするための CaoVariable オブジェクトも必要になります．以下に VB6 でのコード例を示します．

```
Dim g_eng as CaoEngine          ‘ CaoEngine オブジェクト用の変数
Dim g_wrks as caoWorkspace      ‘ CaoWorkspace オブジェクト用の変数
Dim g_ctrl as CaoController     ‘ CaoController オブジェクト用の変数
Dim g_val as CaoVariable        ‘ CaoVariable オブジェクト用の変数
```

(2) CaoEngine オブジェクトを生成します．CaoEngine オブジェクトは New キーワードを使って生成しま

す.

```
Set g_eng = New CaoEngine    ‘ CaoEngine オブジェクトの生成
```

(3) CaoWorkspaceオブジェクトを取得もしくは生成します. CaoEngineオブジェクトを生成すると, デフォルトで Caoworkspaces オブジェクトと Caoworkspace オブジェクトを 1 つずつ生成しています. サンプルプログラムでは, デフォルトワークスペースを利用しています. 以下に CaoWorkspace オブジェクトを新しく生成するコード例を示します.

```
g_wrks = g_ctrl.Addworkspace(“NewWrks”, “”)
```

(4) CaoControllerオブジェクトを生成します. CaoControllerオブジェクトを生成するには, 使用するプロバイダ名と使用するためのパラメータを設定します. NetwoRC プロバイダでは, 接続先のコントローラの IP アドレスをオプションで指定します. 以下にコード例を示します.

```
g_ctrl = g_wrks.AddController(“RC”, “CaoProv.DENSO.NetwoRC”, “”, “conn=eth:192.168.0.1”)
```

(5) CaoVariable オブジェクトを生成します. 接続したい変数の CaoVariable オブジェクトを生成します. 以下に P 型の 10 番目の変数オブジェクトを生成するコード例を示します.

```
g_val = g_ctrl.AddVariable(“P10”, “”)
```

3.2.2. 変数リード・ライト

接続した変数の値を取得・設定するには, CaoVariable オブジェクトの Value プロパティを参照します. 変数に格納, 変数からの設定を行う場合は, 接続した変数型に合わせた変数を用意する必要があります. 以下にコード例を示します.

```
Dim vntPotision as Variant
vntPotision = g_val.Value                    ‘ 値の取得
g_val.Value = Array(50, 50, 50, 0, 0, 0, -1)    ‘ 値の設定
```

3.2.3. 切断

コントローラと切断する場合には, 生成したオブジェクトを消去すると共に, オブジェクトを管理するコレクションクラスから消去するオブジェクトを削除します. 以下にコード例を示します.

```
g_ctrl.Variables.Clear          ‘ CaoVariables から全てのオブジェクトの削除
Set g_val = Nothing             ‘ CaoVariable の消去
g_wrks.Controllers.Remove g_ctrl.Index  ‘ CaoControllers から CaoController の削除
Set g_ctrl = Nothing            ‘ CaoCtonroller の消去
g_eng.Workspaces.Remove g_wrks.Index ‘ CaoWorkspaces からの CaoWorkspace の削除
Set g_wrks = Nothing            ‘ CaoWorkspace の消去
Set g_eng = Nothing             ‘ CaoEngine の消去
```

3.2.4. サンプルプログラム

以下に VB6 で作成したサンプルプログラムを示します. IP や Port は各コントローラで設定した値を用いてください. サンプルプログラムでは以下の設定値を用いて接続しています.

IP:192.168.0.1

Port:4112

List 3-1 Variable.frm

```
Dim g_eng As CaoEngine
Dim g_ctrl As CaoController
Dim g_val As CaoVariable

Private Sub Form_Load()
    Set g_eng = New CaoEngine

    ‘ 接続処理　IP/Port は各コントローラの設定にしてください
    Set g_ctrl = g_eng.Workspaces(0).AddController(“RC7”, “caoProv.DENSO.NetwoRC”, “”,
“conn=eth:192.168.0.1:4112”)

    ‘ 変数名”IO150”
    Set g_val = g_ctrl.AddVariable(“IO150”, “”)

End Sub

Private Sub Form_Unload(Cancel As Integer)
    ‘ 変数オブジェクトの破棄
    g_ctrl.Variables.Clear
    Set g_val = Nothing

    ‘ コントローラオブジェクトの破棄
    g_eng.Workspaces(0).Controllers.Remove g_ctrl.Index
    Set g_ctrl = Nothing

    ‘ CaoEngine の破棄
    Set g_eng = Nothing
End Sub

Private Sub Command1_Click()
    ‘ 変数の読み込み
    Text1.Text = g_val.Value
End Sub

Private Sub Command2_Click()
    ‘ 変数の書き込み
    g_val.Value = Cbool(Text2.Text)
End Sub
```

3.3. PAC プログラムの開始・停止

PAC プログラムの開始・停止を行うには， 図 3-4 示す処理を行います. PAC を起動するには, コントローラが外部自動モードになっている必要があります. またコントローラの起動権の設定を ORiN がインストールされている PC の IP に設定してください. 詳しくは 2.4 章を参照してください.

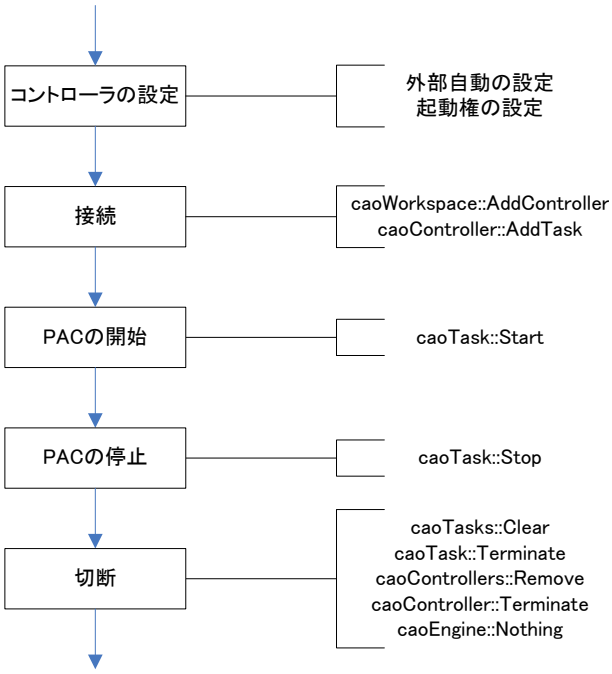


図 3-4 PAC の開始・停止の流れ

3.3.1. 接続

CaoControllerオブジェクトを生成するまでの手順は3.2.1を参照してください. PACプログラムを開始・停止させるにはCaoTaskオブジェクトを生成します. タスクオブジェクトを以下にコード例を示します.

```
Dim g_task as CaoTask          ‘ CaoTask オブジェクトを格納する変数
Set g_task = g_ctrl.AddTask(“PRO01”, “”)
```

3.3.2. PAC プログラムの開始・停止

PACプログラムの開始と停止を行うには, CaoTaskオブジェクトのStartメソッドとStopメソッドを使用します. 以下にPACプログラムの連続実行とサイクル停止の例を示します.

```
g_task.Start 2          ‘ 連続実行
g_task.Stop  3          ‘ サイクル停止
```

3.3.3. サンプルプログラム

List 3-2 Task.frm

```
Dim g_eng As CaoEngine
Dim g_ctrl As CaoController
Dim g_task As CaoTask

Private Sub Command1_Click()
g_task.Start 2
End Sub

Private Sub Command2_Click()
```

```
    g_task.Stop 3
End Sub

Private Sub Form_Load()
    Set g_eng = New CaoEngine

    ‘ 接続処理　IP/Port は各コントローラの設定にしてください
    Set g_ctrl = g_eng.Workspaces(0).AddController(“RC7”, “caoProv.DENSO.NetwoRC”, “”,
“conn=eth:192.168.0.1”)

    ‘ タスク名”PRO1”
    Set g_task = g_ctrl.AddTask(“PRO1”, “”)
End Sub

Private Sub Form_Unload(Cancel As Integer)
    g_ctrl.Tasks.Clear
    Set g_task = Nothing

    g_eng.Workspaces(0).Controllers.Remove g_ctrl.Index
    Set g_ctrl = Nothing

    Set g_eng = Nothing
End Sub
```

3.4. ロボットの動作

ロボットの動作させるには，コントローラで RobSlave を起動させておき，自動モードにしておく必要があります．RobSlave については 2.5 章を参照してください．モータ起動も NetwoRC プロバイダを通して行う事もできます．詳しくは 4.2.39 の CaoTask::Execute motor コマンドを参照して下さい．

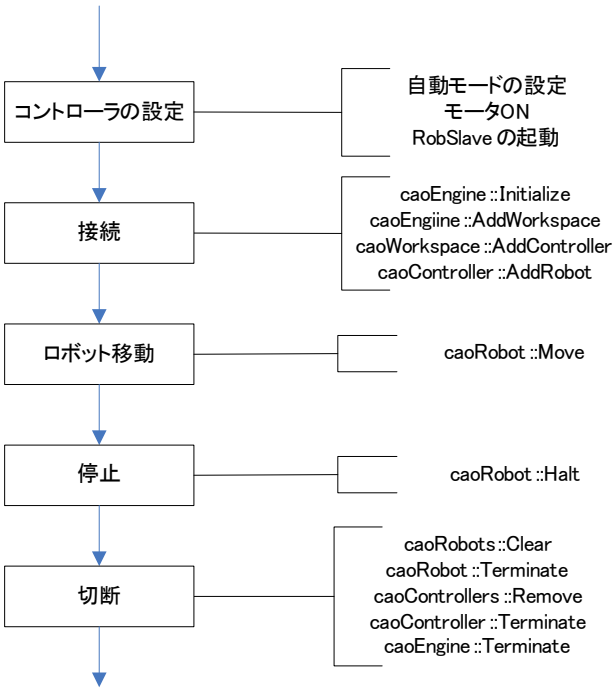


図 3-5 ロボット動作の流れ

3.4.1. 接続

CaoController オブジェクトを生成するまでの手順は 3.2.1 を参照してください. ロボットを動作させるにはCaoRobot オブジェクトを生成します. 以下にコード例を示します.

```
Dim g_robot as CaoRobot        ‘  CaoRobot オブジェクトを格納する変数
Set g_ robot = g_ctrl.AddRobot(“Arm”, “”)
```

3.4.2. ロボットの移動と停止

ロボットを動作させるには CaoRobot::Move メソッドを用います. Move の詳細は 4.2.36 章を参照してください. Move 時に NEXT オプションをつけることによって, CaoRobot::Halt メソッドでロボットの動作を停止させることができます.

List 3-3 Robot.frm

```
Dim g_eng As CaoEngine
Dim g_ctrl As CaoController
Dim g_robot As CaoRobot


‘ ロボット停止
Private Sub Command2_Click()
    g_robot.Halt
End Sub

‘ ロボット動作  IP/Port は各コントローラの設定にしてください
Private Sub Command1_Click()
    g_robot.Move 1, “@P P10”, “NEXT”
End Sub

Private Sub Form_Load()
    Set g_eng = New CaoEngine
    Set g_ctrl = g_eng.Workspaces(0).AddController(“RC7M”, “caoProv.DENSO.NetwoRC”, “”,
“conn=eth:10.6.235.60”)
    Set g_robot = g_ctrl.AddRobot(“Arm”)
End Sub

Private Sub Form_Unload(Cancel As Integer)
    g_ctrl.Robots.Clear
    Set g_robot = Nothing
    g_eng.Workspaces(0).Controllers.Remove g_ctrl.Index
    Set g_ctrl = Nothing
    Set g_eng = Nothing
End Sub
```

3.5. その他サンプル

その他のサンプルはORiN2 SDKのORiN2¥CAO¥ProviderLib¥DENSO¥NetwoRC¥Sample以下を参照して下さい.

表 3-1 サンプルプログラム一覧

サンプル名	区分	内容
Variable	CaoVariable	コントローラの変数,I/O,CNFの読み書き.
File	CaoFile	コントローラのファイルの読み書き.
Tree	CaoFile	コントローラ内のフォルダ一覧表示とファイル取得.
Log	CaoFile	コントローラのエラーログ,操作ログの取得.
Task	CaoTask	コントローラのタスクの情報表示と操作(起動,停止) . ※ RC5ではタスク起動不可です. 2.330
Robot	CaoRobot	ロボットの動作コマンド実行, 現在位置の取得, ユーザ拡張コマンドの呼び出し. ※ ORiN2¥CAO¥ProviderLib¥DENSO¥NetwoRC¥Bin の RobSlave.pac, UserExtention.pac, RobSlave.h がコントローラで必要です. ※ RC5では使用出来ません. 2.330
Trans	CaoController CaoVariable CaoFile	コントローラの全データバックアップとリストア.
Info	CaoController CaoVariable CaoRobot	コントローラの状態表示.
Tracking	CaoRobot	コンベアトラッキング. ※ RC5では使用出来ません. 2.330
SrvLog	CaoRobot	単軸制御ログ取得. ※ RC5では[Run],[Motor]コマンドが使用出来ません. 2.330

4. コマンドリファレンス

4.1. コマンド一覧

表 4-1 コマンド一覧

カテゴリ	メソッド/プロパティ	機能	
caoWorkspace			
	Addcontroller	コントローラに接続	P. 33
caoController			
	AddCommand		P. 35
	AddFile	PAC・システムファイル，フォルダに接続	P. 35
	AddRobot	ロボットに接続	P. 36
	AddTask	タスク(PAC)に接続	P. 37
	AddVariable	ユーザ・システム変数に接続	P. 37
	get_TaskNames	タスク(PAC)名の一覧の取得	P. 38
	get_VariableNames	ユーザ・システム変数の一覧の取得	P. 38
	Execute	コントローラクラスに実装されているコマンドの実行	P. 38
	OnMessage	OnMessage イベント	P. 41
CaoCommand			
	Execute	コマンドの実行	P. 42
	get_Parameters	実行パラメータの取得	P. 42
	put_parammeters	実行パラメータの設定	P. 42
CaoFile			
	AddFile	PAC ファイルに接続	P. 42
	AddVariable	ファイルのシステム変数に接続	P. 43
	get_VariableNames	システム変数の一覧取得	P. 43
	Copy	ファイルのコピー	P. 43
	Delete	ファイルの削除	P. 43
	Move	ファイルの移動	P. 43
	get_DateCreated	作成日時の取得	P. 43
	get_DateLastAccessed	最終アクセス日時の取得	P. 43
	get_DateLastModified	最終更新日の取得	P. 43
	get_FileNames	ファイル名一覧の取得	P. 43
	get_Attribute	ファイルの属性値の取得	P. 43
	get_Path	ファイルのパスの取得	P. 44
	get_Size	ファイルのサイズの取得	P. 44
	get_Type	ファイルの拡張子の取得	P. 44
	get_Value	ファイルの内容の取得	P. 44
	put_Value	ファイルの内容の書き換え	P. 44

CaoRobot			
	Accelerate	内部加速度, 内部減速度の設定	P. 44
	AddVariable	システム変数の接続	P. 44
	get_VariableNames	システム変数一覧の取得	P. 44
	Halt	ロボット非同期動作時の停止	P. 44
	Change	ロボットのツール座標系/ユーザ座標系の変更	P. 45
	Drive	サポートされていません.	P. 45
	Move	ロボットの動作	P. 45
	Rotate	軸回りの回転動作	P. 49
	Speed	内部移動速度の設定	P. 50
	Execute	ロボット特有の動作の実行	P. 50
CaoTask			
	AddVariable	システム変数の接続	P. 67
	get_VariableNames	システム変数の一覧取得	P. 67
	Start	PAC プログラムの実行	P. 67
	Stop	PAC プログラムの停止	P. 67
CaoVariable			
	get_Value	値の取得	P. 68
	put_Value	値の設定	P. 68
	put_ID	変数の番号の設定	P. 68
	get_ID	変数の番号の取得	P. 68
	get_Microsecond	タイムスタンプの取得	P. 68
CaoMessage			
	Clear	発生中のエラーのクリア	P. 68

4.2. メソッド・プロパティ

4.2.1. CaoWorkspace::AddController メソッド

NetwoRC プロバイダでは AddController 時に, 通信用の接続パラメータを参照し, 通信の接続を行います.

このときオプションで通信形態, 接続パラメータ, タイムアウトの設定を指定します. オプションとオプションの区切りは“,”で行います.

書式 AddController(<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPcName:BSTR > [,<bstrOption:BSTR>])

bstrCtrlName	:	[in]	コントローラ名
bstrProvName	:	[in]	プロバイダ名. 固定値 =” CaoProv.DENSO.NetwoRC”.
bstrPcName	:	[in]	プロバイダの実行マシン名
bstrOption	:	[in]	オプション文字列＝“<オプション 1>,<オプション 2>,…”

以下にオプション文字列に指定するリストを示します.

表 4-2 CaoWorkspace::AddController のオプション文字列

オプション	説明
Conn=<接続パラメータ>	必須. 通信形態とその接続パラメータを設定します. 詳細は 4.2.1.1 に示します.
MyIP=[<ローカル IP アドレス>][:ローカル UDP ポート番号]	通信に使用するローカルマシンの IP アドレスと UDP ポート番号を指定します. 複数の NIC を使う場合にこのオプションで IP アドレスを指定して NIC を選択することができます. 省略した場合は, 自動的に選択されます. ローカルマシンに割り当てられていない IP アドレスを指定したときはエラーを返します. UDP ポート番号を省略した場合は, 自動的に選択されます. RS232C 接続の場合, このオプションは無視されます.
Timeout=<タイムアウト時間>	送受信時のタイムアウト時間. (デフォルト:400) msec
TORetry=<リトライ回数>	送受信時のリトライ回数. 1～7(デフォルト:5) 1 以下の場合, 1 として扱われます. 7 以上の場合, 7 として扱われます.

RS232C 接続の場合は, 1 つのポートにつき 1 つのコントローラオブジェクトしか作成できません.

4.2.1.1. Conn オプション

以下に Conn オプションの接続パラメータ文字列を示します. ここで角括弧(“[ ]”)内は省略可能を示します. また, 各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値を示します.

- **RS232C デバイス**

“com:[<COM Port>[:<BaudRate>[:<Parity>:<DataBits>:<StopBits>]]]”

<COM Port> : COM ポート番号. ‘1’-COM1, ‘2’-COM2, …
<BaudRate> : 通信速度. 4800, 9600, 19200, 38400, 57600, 115200.
<Parity> : パリティ. ‘N’-NONE, ‘E’-EVEN, ‘O’-ODD.
<DataBits> : データビット数. ‘7’-7bit, ‘8’-8bit.
<StopBits> : ストップビット数. ‘1’-1bit, ‘2’-2bit.

- **Ethernet デバイス**

“eth:<IP Address>[:<Port No>]”

<IP Address> : : IP アドレス.
例:“127.0.0.1”, “192.168.0.1”
<Port No> : : UDP 接続ポート番号. 4112, 4113, …任意指定可能

4.2.2. CaoController::AddCommand メソッド

CaoController クラスの AddCommand メソッドの引数は, コマンド名(BSTR 型)を指定します.
現状では仕様公開されているコマンド名はありません.

4.2.3. CaoController::AddFile メソッド

CaoController クラスの AddFile メソッドの引数は, ファイル名(BSTR 型)を指定します. ここで指定する“ファイル名”は PAC プログラム名あるいはシステム予約されているファイル名を指定します.
この引数にファイルパスのみを指定してディレクトリを指定することもできます. また, パスを省略した場合は, デフォルトフォルダである”/rom/prj”内のファイルであるとします.
以下に AddFile の仕様を示します.

書式 AddFile(<bstrName:BSTR > [,<bstrOption:BSTR>])

bstrName : [in] ファイル名
bstrOption : [in] オプション文字列

オプションは以下の文字列を使用します.

表 4-3 CaoController::AddFile のオプション文字列

オプション	意味
@Create[=<0～2>]	指定したファイルがないとき, このオプション値に従ってファイルを作成します. 0:ファイルを作成しません. (デフォルト) 1:ファイルを作成します. 2:フォルダを作成します. 指定したファイルが存在するときはこのオプションは無視されます.

ファイルの一覧を下表に示します. ファイルの詳細フォーマットに関してはファイル仕様書を参照して下さい.

表 4-4: ファイルの実装状況一覧

	ORiN2ファイル名	形式	説明
1	*.PAC	text	PAC ソース
2	*.H	text	PAC ヘッダ

3	*.NIC	bin	PAC 実行形式
4	*.MAP	text	PAC 相互参照
5	@VAR_INT	bin	I 型変数
6	@VAR_SNG	bin	F 型変数
7	@VAR_DBL	bin	D 型変数
8	@VAR_VEC	bin	V 型変数
9	@VAR_POS	bin	P 型変数
10	@VAR_JNT	bin	J 型変数
11	@VAR_TRN	bin	T 型変数
12	@VAR_STR	bin	S 型変数
13	@VAR_TOOL	bin	ツール座標定義
14	@VAR_WORK	bin	作業座標定義
15	@VAR_AREA	bin	エリア定義
16	@LOG_ERROR	bin	エラーログ
17	@LOG_OPERATION	bin	操作ログ
18	@LOG_CONTROL	bin	制御ログ
19	@CNF_ITP	bin	インタプリタ環境設定
20	@CNF_PAC	bin	プログラム環境設定
21	@CNF_DIO	bin	I/O 環境設定
22	@CNF_ARM	bin	軌道生成環境設定
23	@CNF_SRV	bin	サーボ環境設定
24	@CNF_SPD	bin	使用条件設定
25	@CNF_VIS	bin	視覚環境設定
26	@CNF_COM	bin	通信環境設定

【注意】

CaoFile オブジェクトはファイルへの同時アクセスに対応していません.

必ずアプリケーション側でファイルへの排他制御を行うようにしてください.

4.2.4. CaoController::AddRobot メソッド

CaoController クラスの AddRobot メソッドの引数は, ロボット名(BSTR 型)を指定します. ここで指定する“ロボット名”は任意の文字列で特に何の制限もありません. 例えば, AddRobot (“Arm1”)のように指定することができます. AddRobot メソッドを呼び出すと CaoRobot オブジェクトが取得できます.

書式 AddRobot(<bstrName:BSTR > [,<bstrOption:BSTR>])

bstrName : [in] ロボット名

bstrOption : [in] オプション文字列(未使用)

4.2.5. CaoController::AddTask メソッド

CaoController クラスの AddTask メソッドの引数は，タスク名(BSTR 型)を指定します．ここで指定する“タスク名”は PAC プログラム名を指定します．例えば，AddTask (“pro1”)といった表現で CaoTask オブジェクトが取得できます．

書式 AddTask(<bstrName:BSTR > [,<bstrOption:BSTR>])

bstrName	:	[in]	タスク名
bstrOption	:	[in]	オプション文字列(未使用)

タスク名に“@ALL”を指定した場合は，作成される CaoTask オブジェクトは NetwoRC コントローラの全タスクを対象とする CaoTask::Start メソッド及び CaoTask::Stop メソッドの機能を提供します．

4.2.6. CaoController::AddVariable メソッド

この CaoController クラスの AddVariable メソッドは，CAO の一般的意味では，変数にアクセスするためのメソッドです．NetwoRC プロバイダでは，変数名にはユーザ変数，システム変数のどちらでも指定することができます．

ユーザ変数の場合は，そのまま NetwoRC コントローラの変数(I,F,V,P,J,D,T,S)，ツール，ワーク，エリア，I/O およびシステムパラメータ(CNF)に対応します．

以下に，AddVariable の仕様を示します．

書式 AddVariable(<bstrName:BSTR > [,<bstrOption:BSTR>])

bstrName	:	[in]	変数名 “<変数名>[<番号>]”
bstrOption	:	[in]	オプション文字列 “<オプション>”

<変数名>	:	I, F, V, P, J, D, T, S または IO, TOOL, WORK, AREA または _ITP, _PAC, _DIO, _ARM, _SRV, _SPD, _VIS, _COM または Wdin, WDOut 文字はすべて半角指定(全角は無効)で大小の区別はありません． システム変数(CNF)は“_”(アンダースコア)で開始します．
<番号>	:	変数名で指す変数の番号 または ＊ または ＊_<数値> 番号は 10 進数で指定します． ＊を指定した場合，初期値 0 として扱われます．変数の番号は変数オブジェクトの ID プロパティで参照，変更することができます． すべて半角指定(全角は無効) ． ＊_<数値>の数値は 10 進数で指定します．同種変数のワイルドカード(＊)指定を複数定義可能にするための識別番号です．
<オプション>	:	“LEN=<ビット長>”（IO 変数の場合のみ有効） <ビット長>は 1, 8, 16 が指定できます．デフォルトは 1 です． すべて半角指定(全角は無効)

“[”および“] ”は省略できます.

(例 1)	“i0”, “I[0]”	・・・	I 型変数の 0 番を指定
(例 2)	“IO128”, “io[128]”	・・・	I/O 変数の 128 番を指定
(例 3)	“_ARM0”, “_arm[0]”	・・・	ARMCNF の 0 番目を指定 (???CNF[0]には???CNF の要素数が格納されています)
(例 4)	“_itp19”, “_itp[19]”	・・・	ITPCNF の 19 番目を指定
(例 5)	“tool10”, “Tool[10]”	・・・	ツールの 10 番目を指定
(例 6)	“I*”, “_PAC[*]”	・・・	ID プロパティで番号を指定する
(例 7)	“I*_1”, “I*_2” , “I*_3”	・・・	複数ワールドカード指定

CNF 変数(_ITP, _PAC, _DIO, _ARM, _SRV, _SPD, _VIS, _COM)の番号はペンダントで表示されている値とイコールではないので注意して下さい. ペンダントでは 0 番目の要素数は表示されないので値がずれています. (多くの場合 CNF 変数=ペンダント表示+1)

また, システム変数を指定するときは, 変数名の最初に“@”をつけます. 変数名の最初に“@”がないものは, すべてユーザ変数として扱われます.

NetwoRC プロバイダで実装されているシステム変数は 4.3 を参照して下さい.

4.2.7. CaoController::get_TaskNames プロパティ[5]

AddTask メソッドで指定できる PAC プログラム名の一覧を取得します.

4.2.8. CaoController::get_VariableNames プロパティ

AddVariable メソッドで指定できる変数名とシステム変数名の一覧を取得します.

4.2.9. CaoController::Execute メソッド

コマンドを実行します.

Execute メソッドの引数は, コマンドを BSTR, パラメータを VARIANT 配列で指定します.

書式 [<vntRet:VARIANT> =] Execute(<bstrCmd:BSTR > [,<vntParam:VARIANT>])

bstrCmd	:	[in]	コマンド
vntParam	:	[in]	パラメータ
vntRet		[out]	返り値

コマンドの実行に必要なパラメータおよび取得する結果は表 4-5 を参照して下さい.

[5] VT_ARRAY|VT_VARIANT (Ver1.1.0 未満は VT_ARRAY|VT_BSTR)を返す.

例)

```
Dim vRes as Variant
vRes = caoCtrl.Execute("PutMode", 1 )  'モードの設定(auto)
```

表 4-5: CaoController::Execute メソッドのコマンド実装一覧

コマンド	パラメータ	返り値	動作
GetAutoMode	なし	<Mode:VT_I2> = 0:Unkown 1:内部自動 2:外部自動	自動モードの状態を取得します.
PutAutoMode <Mode>	<Mode:VT_I2> = 1:内部自動 2:外部自動	なし	自動モードの設定を行います.
StartLog	なし	なし	サーボ制御ログの記録を開始します.
StopLog	なし	なし	サーボ制御ログの記録を停止します.
ClearLog	なし	なし	サーボ制御ログの記録を初期化します.
SaveFile	なし	なし	ファイルの保存を行います.
GetFileTransMode	なし	<Mode:VT_I4> = ファイル転送モード 0＝通常転送 0bit:旧手順 1bit: ROM 操作 2bit:進行状況通知 (OnMessage イベントあり)	ファイル転送モードの状態を取得します.

PutFileTransMode <Mode>	<Mode:VT_I4> = ファイル転送モード 0＝通常転送 0bit:旧手順 1bit: ROM 操作 2bit:進行状況通知 (OnMessage イベントあり)	なし	ファイル転送モードの設定を行います.
ChangeConfig <CnfID>	<CnfID:VT_I2> = 1 : COMCNF 2 : ARMCNF 3 : VISCNF 4 : PACCNF 5 : SRVCNF 6 : SPDCNF 7 : ITPCNF 8 : DIOCNF 9 : SYSCNF	なし	CNF の変更を反映します.
SetDummyIO <IO>,<Value>[,<Range>]	<IO:VT_I2> = I/O 番号 <Value:VT_I2> = 状態 0:OFF 1:ON [<Range:VT_I2>] = 範囲(省略時:1) ex. 1 -> <IO>のみ, 8 -> <IO>から <IO>+7 まで	なし	擬似入力の設定を行います. I/O 番号に-1 を指定すると, 全クリアを行います. (<Value> は無視されます. Range は省略するか 1 を指定してください.)
GetDummyIO <IO>	<IO:VT_I2> =I/O 番号	<Value:VT_I2> = 状態 0:OFF, 1:ON	擬似入力の設定を取得します.
LoadNIC <WaitForCompletion>	< WaitForCompletion:VT_BOOL > = ロードの完了待ち True,False	なし	NIC ファイルをロードします.
DoSignal <Mode>,<Action>	<Mode:VT_I4>=モード <Action:VT_I4>=アクション	なし	コントローラへ処理開始のタイミングを通知します.
GetVarSize <Type>	<Type:VT_BSTR>= 変数型 “I”,”F”,”D”,”V”,”P”,”J”,”T”,”S”	<Num:VT_I4>=変数個数	変数個数を取得します.
Compile	なし	なし	コンパイルを実行します. 2.330

GetCompileState	なし	<Mode:VT_I4> = 1:コンパイル中(ロード中) 0:正常終了 -1:異常終了(コンパイルエラー) -2:異常終了(コンパイルエラー以外の要因)	コンパイルの進捗状況を取得します. 2.330
SetExtension <Mode>, <Key >	<Mode:VT_I2>= 1:Add , 2:Remove <Key:VT_I4>=キー	なし	機能拡張の追加/削除を行います.
ClearError <ErrNo>	<ErrNo:VT_I4>=エラー番号	<Value:VT_I2> = 状態 0:OFF , 1:ON	エラークリアを実行します.
InitNonStopPathLib[6]	なし	なし	軌道生成処理の初期化を行います.
GenerateNonStopPath[6]	<教示点:<座標情報> | VT_ARRAY>, <エリア:<エリア情報> | VT_ARRAY>, <教示点数:VT_I4>, <総速度比:VT_R4(0.0～1.0)>, <補正係数:VT_R4(0.0～1.0)>, [<補正方法:VT_I4> = 0:付加軸非同期 1:付加軸同期(デフォルト値)]	<動作点:<座標情報> | VT_ARRAY>	軌道生成処理を実行します. <教示点>の1点目が開始位置, 最終点が終了位置となります. 詳細は「付録 F 無停止教示点補正機能(外観検査軌道生成)」を参照してください.

4.2.10. CaoController::OnMessage イベント

CaoController クラスの OnMessage イベントの実装状況を下表に示します.

表 4-6: OnMessage イベントの実装状況一覧

番号	データ型	通知内容	意味
1	VT_BSTR	エラーメッセージ	エラーが発生.
2	VT_BSTR	エラーメッセージ	応答なし.
3	VT_I4	<LONG:RangeMax>	GetText:開始, 進捗状況の通知開始, 最大レンジは<RangeMax>です

[6] 【注意】本コマンドを使用するには「無停止教示点補正機能」オプションのライセンスを別途ご購入ください.

			※Execute(“Put_FileTransMode <Mode>”)で 2bit:進行状況 = 1 必須
4	VT_I4	<LONG:Range>	GetText:進行中,現在の進捗を返します ※Execute(“Put_FileTransMode <Mode>”)で 2bit:進行状況 = 1 必須
5	VT_I4	<LONG:RangeMax> または-1(エラー)	GetText:完了 処理完了, 最大レンジの場合は正常,-1 の場合は NG ※Execute(“Put_FileTransMode <Mode>”)で 2bit:進行状況 = 1 必須
6	VT_I4	<LONG:RangeMax>	PutText:開始 進捗状況の通知開始, 最大レンジは<RangeMax>です ※Execute(“Put_FileTransMode <Mode>”)で 2bit:進行状況 = 1 必須
7	VT_I4	<LONG:Range>	PutText:進行中,現在の進捗を返します ※Execute(“Put_FileTransMode <Mode>”)で 2bit:進行状況 = 1 必須
8	VT_I4	<LONG:RangeMax> または-1(エラー)	PutText:完了 処理完了, 最大レンジの場合は正常,-1 の場合は NG ※Execute(“Put_FileTransMode <Mode>”)で 2bit:進行状況 = 1 必須

4.2.11. CaoCommand::Execute メソッド

このメソッドは, コマンドの実行を行います. コマンド実行の詳細な動作については CaoController クラスの AddCommand メソッドを参照してください.

4.2.12. CaoCommand::get_Parameters プロパティ

現在設定されている実行パラメータを取得します.

4.2.13. CaoCommand::put_Parameters プロパティ

4.2.11 の実行パラメータを設定します.

パラメータの内容が不正な場合でも, このプロパティではエラーを返しません. このような場合はコマンドの実行時にエラーを返します.

4.2.14. CaoFile::AddFile メソッド

前述 4.2.3 と同様にファイルオブジェクトを作成します. 作成した CaoFile オブジェクトに対応しているファイルのパスは“<親オブジェクトのパス>/<AddFile で指定したファイル名>”となります.

4.2.15. CaoFile::AddVariable メソッド

CaoFile クラスの AddVariable メソッドの引数は，システム変数名を指定します．

実装されているシステム変数の一覧は表 4-14 を参照して下さい．

4.2.16. CaoFile::get_VariableNames プロパティ

AddVariable メソッドで指定できる変数名とシステム変数名の一覧を取得します．

4.2.17. CaoFile::Copy メソッド

オブジェクトに対応しているファイルを指定した場所にコピーします．

4.2.18. CaoFile::Delete メソッド

オブジェクトに対応しているファイルを削除します．ファイルを削除した後もオブジェクトは消去されないので，不必要なときはクライアントでオブジェクトを消す必要があります．

4.2.19. CaoFile::Move メソッド

オブジェクトに対応しているファイルを指定した場所に移動します．対応するファイルの場所は変更されますが，オブジェクト名は変更されません．

4.2.20. CaoFile::get_DateCreated プロパティ

オブジェクトに対応しているファイルの作成日時を取得します．

4.2.21. CaoFile::get_DateLastAccessed プロパティ

オブジェクトに対応しているファイルの最終アクセス日時を取得します．

4.2.22. CaoFile::get_DateLastModified プロパティ

オブジェクトに対応しているファイルの最終更新日時を取得します．

4.2.23. CaoFile::get_FileNames プロパティ[7]

オブジェクトに対応しているパスがディレクトリのときのみ実行できます．

実行するとディレクトリ内のファイル名リストを取得します．

4.2.24. CaoFile::get_Attribute プロパティ

オブジェクトに対応しているファイルの属性値を取得します．

[7] VT_ARRAY|VT_VARIANT (Ver1.1.0 未満は VT_ARRAY|VT_BSTR)を返す．

4.2.25. CaoFile::get_Path プロパティ

オブジェクトに対応しているファイルのパスを取得します. 取得した値にファイル名は含まれません.

4.2.26. CaoFile::get_Size プロパティ

オブジェクトに対応しているファイルのファイルサイズを取得します.

4.2.27. CaoFile::get_Type プロパティ

オブジェクトに対応しているファイルの拡張子を取得します.

4.2.28. CaoFile::get_Value プロパティ

オブジェクトに対応しているファイルの内容を取得します.

4.2.29. CaoFile::put_Value プロパティ

オブジェクトに対応しているファイルの内容を書き換えます.

4.2.30. CaoRobot::Accelerate メソッド

ロボットの内部加速度, 内部減速度を指定します.

このメソッドは PAC 言語の ACCEL,JACCEL 命令に対応します.

以下に, Accelerate の仕様を示します.

書式 Accelerate <lAxis:LONG >, <fAccel:FLOAT> [,<fDecel:FLOAT>]

lAxis	: [in]	軸番号	-1:手先加速度(ACCEL), 0:全軸加速度(JACCEL)
fAccel	: [in]	加速度	(-1:現在の設定のままに,変更なし)
fDecel	[in]	減速度	(-1:現在の設定のままに,変更なし)

(例 1) Accelerate 0, 50.0, -1 ‘ 加速度 = 50% , 減速度 = 変更なし

(例 2) Accelerate 0, -1, 60.0 ’ 加速度 =変更なし , 減速度 = 60%

4.2.31. CaoRobot::AddVariable メソッド

CaoRobot クラスの AddVariable メソッドの引数は, システム変数名を指定します.

実装されているシステム変数の一覧は表 4-12 を参照してください.

4.2.32. CaoRobot::get_VariableNames プロパティ

AddVariable メソッドで指定できる変数名とシステム変数名の一覧を取得します.

4.2.33. CaoRobot::Halt メソッド

CaoRobot クラスの Move, Drive, Rotate 等のロボット動作命令実行メソッドに“NEXT”オプションを指定して非同期実行した場合は Halt メソッドでロボット動作を途中で停止させることが出来ます. ただし, 非同期実行

であってもロボット動作命令を二つ以上連続して実行した場合，一つ前の動作命令が完了していない内は次の動作命令は“待ち”の状態（同期実行）になり，一定周期で CaoController クラスの OnMessage イベントが発行されます．この状況下では Halt メソッドは動作しませんので，ロボット動作を停止させるには下記の何れかの処理を実行するようにしてください．

(1) CaoTask クラスの Stop メソッドを使用して“ROBSLAVE”タスクの実行を停止させる
(2) I/O を介してロボット停止信号を入力する

動作命令完了待ちでない状態の時に Halt メソッドを呼んだ場合は何も行いません．

4.2.34. CaoRobot::Change メソッド

RobSlave 2.330

ロボットのツール座標系/ユーザ座標系を変更します．
このメソッドは PAC 言語の CHANGETOOL 及び CHANGEWORK 命令に対応します．

以下に，Change の仕様を示します．

書式 Change <bstrName:BSTR>

bstrName	:	[in]	CHANGETOOL 動作の場合= “Tool <番号>”
			CHANGEWORK 動作の場合= “Work <番号>”
<番号>	:		10 進数で表現される数値（デフォルト：0）

4.2.35. CaoRobot::Drive メソッド

2.330

このメソッドは本プロバイダで直接サポートされていません．
代わりに複数軸同時動作可能な CaoRobot::Execute の“DriveEx”, “DriveAEx”を使用してください．

4.2.36. CaoRobot::Move メソッド

RobSlave 2.330

ロボットを指定座標へ移動します．このメソッドは PAC 言語の MOVE 命令に対応します．以下に，Move の仕様を示します．

書式 Move <lComp:LONG >, <vntPose:POSEDATA> [,<vntPose:POSEDATA>…] [, < bstrOpt:BSTR>]

lComp	:	[in]	補間指定 1:MOVE P,… , 2:MOVE L,… , 3:MOVE C,… , 4:MOVE S,…
vntPose	:	[in]	ポーズ列(POSEDATA 型)
bstrOpt	:	[in]	動作オプション “NEXT” = 非同期実行

ポーズ列の POSEDATA 型に関しては「付録 APOSEDATA 型定義」を参照してください．
POSEDATA 型データで文字列指定する場合の表記形式と意味は下記の通りです．

VT_BSTR 型(文字列)指定 :

・ 補間指定 = 1, 2 の場合

“[<@パス開始変位>]<ポーズ> [<付加軸オプション>]”

ex. “P1”, “@P T100”, “@E J520”

・ 補間指定 = 3 の場合

“<ポーズ 1> [<付加軸オプション>], [<@パス開始変位>] <ポーズ 2> [<付加軸オプション>]”

※ *** ポーズ 1,2 は必ず同じ変数型でなくてはなりません! ***

ex. „**P**1,@E **P**2“, „**T**100,@P **T**101“

・ 補間指定 = 4 の場合

“[<@パス開始変位>]<自由曲線軌道番号> [<付加軸オプション>]”

ex. “1”, “@P 20”, “@E 5”

<自由曲線軌道番号> : 10 進数で表現される数値(スプライン曲線番号 1～20)

<ポーズ> : “<変数型><番号>” または “[<変数型>](<要素 1>,<要素 2>,…)”

: <変数型> : ’P’,’T’,’J’のいずれか一文字
※要素(即値)指定で変数型が省略された場合は’P’指定扱いになります.

<番号> : 10 進数で表現される数値

<要素 n> : 各変数型(P,J,T)の要素

P 型＝P(<x>,<y>,<z>,<rx>,<ry>,<rz>,<fig>)

J 型＝J(<j1>,<j2>,<j3>,<j4>,<j5>,<j6>,<j7>,<j8>)

T 型＝T(<x>,<y>,<z>,<ox>,<oy>,<oz>,<ax>,<ay>,<az>,<fig>)

※4 軸ﾛﾎﾞｯﾄの場合 P 型の T 要素は<rz>に対応します. <rx>,<ry>は未使用

<@パス開始変位> : “@0” , “@P” , “@E” , “@<数値>”のいずれか

<付加軸オプション> : 付加軸を扱う場合は下記書式で付加軸オプションを指定可能です[8].
(ポーズデータ後にブランクを挿入し, 続いて付加軸オプションを指定します)

“EX((<軸番号 1>,<相対移動量 1>)[,(<軸番号 2>,<相対移動量 2>)…])”

あるいは

“EXA((<軸番号 1>,<軸座標 1>)[,(<軸番号 2>,<軸座標 2>)…])”

[8] 付加軸オプションを使用する場合は必ずコントローラ側で付加軸に対応した設定（アームグループの定義等）を行った上で, TakeArm コマンドで付加軸が制御できるアームグループを予め選択してください.

(例 1) Move 1, “@P P1” ,”NEXT” ‘ MOVE P, @P P1, NEXT

(例 2) Move 3, “P1,@E P2” ‘ MOVE C, P1,@E P2

(例 3) Move 2, “@0 P(307.1856,-157.8244,107.0714,160,0,0,1)” ‘ MOVE L,@0 P(307.1856,-157.8244,107.0714,160,0,0,1)

(例 4) Move 4, “@E 2” ‘ MOVE S, @E 2

(例 5) Move 1, “@P P10 EX((7, 30.5))” ,”NEXT” ‘ MOVE P, @P P10 EX((7,30.5)), NEXT

(例 6) Move 2, “@E P20 EXA((7, 30.8), (8, 90.5)) ‘MOVE L, @E P20 EXA((7, 30.8), (8, 90.5))”

Moveメソッドを二つ以上連続して実行した場合，一つ前の動作命令が完了していない内は次の動作命令は“待ち”の状態になり，アプリケーションが応答なしの状態と同じように見えます．この間は周期的にCaoControllerクラスのOnMessageイベント9番号が発行されますので必要に応じてこのイベントを捕らえてアプリケーションに制御権を渡すようにして下さい．

Move メソッドで対応している PAC 言語の MOVE コマンドの一覧を示します．

表 4-7 Move コマンド一覧

区分	PAC コマンド	Move メソッド
MOVE P,…	MOVE P, P<*n1*>	Move 1, “P<*n1*>”
	MOVE P, @P P<*n1*>	Move 1, “@P P<*n1*>”
	MOVE P, @E P<*n1*>	Move 1, “@E P<*n1*>”
	MOVE P, T<*n1*>	Move 1, “T<*n1*>”
	MOVE P, @P T<*n1*>	Move 1, “@P T<*n1*>”
	MOVE P, @E T<*n1*>	Move 1, “@E T<*n1*>”
	MOVE P, J<*n1*>	Move 1, “J<*n1*>”
	MOVE P, @P J<*n1*>	Move 1, “@P J<*n1*>”
	MOVE P, @E J<*n1*>	Move 1, “@E J<*n1*>”
MOVE L,…	MOVE L, P<*n1*>	Move 2, “P<*n1*>”
	MOVE L, @P P<*n1*>	Move 2, “@P P<*n1*>”
	MOVE L, @E P<*n1*>	Move 2, “@E P<*n1*>”
	MOVE L, T<*n1*>	Move 2, “T<*n1*>”
	MOVE L, @P T<*n1*>	Move 2, “@P T<*n1*>”
	MOVE L, @E T<*n1*>	Move 2, “@E T<*n1*>”
	MOVE L, J<*n1*>	Move 2, “J<*n1*>”
	MOVE L, @P J<*n1*>	Move 2, “@P J<*n1*>”
	MOVE L, @E J<*n1*>	Move 2, “@E J<*n1*>”

MOVE C,…	MOVE C, P<*n1*>, P<*n2*>	Move 3, “P<*n1*>, P<*n2*>”
	MOVE C, P<*n1*>, @P P<*n2*>	Move 3, “P<*n1*>, @P P<*n2*>”
	MOVE C, P<*n1*>, @E P<*n2*>	Move 3, “P<*n1*>, @E P<*n2*>”
	MOVE C, T<*n1*>, T<*n2*>	Move 3, “T<*n1*>, T<*n2*>”
	MOVE C, T<*n1*>, @P T<*n2*>	Move 3, “T<*n1*>, @P T<*n2*>”
	MOVE C, T<*n1*>, @E T<*n2*>	Move 3, “T<*n1*>, @E T<*n2*>”
	MOVE C, J<*n1*>, J<*n2*>	Move 3, “J<*n1*>, J<*n2*>”
	MOVE C, J<*n1*>, @P J<*n2*>	Move 3, “J<*n1*>, @P J<*n2*>”
	MOVE C, J<*n1*>, @E J<*n2*>	Move 3, “J<*n1*>, @E J<*n2*>”
MOVE S,…	MOVE S, *n1*	Move 4, “*n1*”
	MOVE S, @P *n1*	Move 4, “@P *n1*”
	MOVE S, @E *n1*	Move 4, “@E *n1*”
付加軸対応	MOVE P, P<*n1*> EX((*j1*, *v1*))	Move 1, “P<*n1*> EX((*j1*,*v1*))”
	MOVE P, P<*n1*> EX((*j1*, *v1*),(*j2*, *v2*))	Move 1, “P<*n1*> EX((*j1*,*v1*),(*j2*, *v2*))”
	MOVE P, P<*n1*> EXA((*j1*, *v1*))	Move 1, “P<*n1*> EXA((*j1*,*v1*))”
	MOVE P, P<*n1*> EXA((*j1*, *v1*),(*j2*, *v2*))	Move 1, “P<*n1*> EXA((*j1*,*v1*),(*j2*, *v2*))”
その他	MOVE P, P<*n1*> +(*x*,*y*,*z*,*rx*,*ry*,*rz*)	Move 1, DEV(“P<*n1*>”, “P(*x*,*y*,*z*,*rx*,*ry*,*rz*)”)
	MOVE P, P<*n1*> +(*x*,*y*,*z*,*rx*,*ry*,*rz*)H	Move 1, DEVH(“P<*n1*>”, “P(*x*,*y*,*z*,*rx*,*ry*,*rz*)”)
		※DEV, DEVH は CaoRobot::Execute を参照

<*n1*>, <*n2*>：整数 0～65535 または”(<要素 1>,<要素 2>,…)”

CaoRobot クラスのロボット動作命令に関するコーディングは ORiN2 SDK の ORiN2¥CAO¥ProviderLib¥DENSO¥NetwoRC¥Sample¥Robot のサンプルを参照ください.

【必要条件】

現在の仕様では CaoRobot クラスのロボット動作命令(表 4-8)を実行するには予めロボットコントローラ RC7 で”RobSlave.pac”プログラムが実行されていなくてはなりません. RobSlave.pac は ORiN2¥CAO¥ProviderLib¥DENSO¥NetwoRC¥Bin にあります.

ロボットコントローラの T[0], T[1]及び I[0]変数が通信用に使用されますのでこれらの変数の使用を避けてください！

表 4-8 RoboSlave.pac を必要とするメソッド一覧

メソッド	コマンド
CcaoProvRobot::Accelerate	
CcaoProvRobot::Change	
CcaoProvRobot::Halt	
CcaoProvRobot::Move	
CcaoProvRobot::Rotate	

CcaoProvRobot::Speed	
CcaoProvRobot::Execute	Approach, Depart, Draw, Motor , ClrSplinePoint , SetSplinePoint , GetSplinePoint , WaitSplinePoint , WaitMotionEnd , MotionSkip , MotionComp , DefTool , DefWork , DefArea , Interrupt , PosClr , Arrive , RotateH , DriveEx , DriveAEx , Delay , SYSSTATE , J2P , J2T , P2J , P2T, T2J, T2P, TINV, NORMTRN, TMUL, DEV, DEVH, FigAprp, FigAprl, TrackDataInitialize , TrackDataSet , TrackDataGet , TrackDataInfo , TrackDataNum , CurTrackPos , CurTrackSpd , WaitTrackMove , CalcWorkPos , CurTrackPosEx , WaitTrackMoveEx , SetTrackMove , ResetTrackMove , SetTrackStartArea , UserExt , ST_*, TakeArm, GiveArm,

【注意】

全グローバル変数(I,F,D,V,P,J,T,S)のインデックス[0]から[9]まで変数はシステム予約しています. これらの変数へのアクセスはユーザープログラムでは行わないでください.

4.2.37. CaoRobot::Rotate メソッド

RobSlave 2.330

指定した軸回りの回転動作を行います.

このメソッドは PAC 言語の ROTATE 命令に対応します.

以下に, Rotate の仕様を示します.

書式 Rotate <vntRotSuf:POSEDATA>, <fDeg:FLOAT>, <vntPivot:POSEDATA>, <bstrOpt:BSTR>

vntRotSuf	:	[in]	回転面
fDeg	:	[in]	角度(deg)
vntPivot	:	[in]	回転中心
bstrOpt		[in]	動作オプション “@0”,”@P”,”@E”,“pose=<n>”,“NEXT” , “@<数値>”

ここで, POSEDATA 型に関しては「付録 A POSEDATA 型定義」を参照してください. POSEDATA 型データで文字列指定する場合の表記形式と意味は下記の通りです.

VT_BSTR 型(文字列)指定 :

・ vntRotSuf（回転面）の場合

“V<n1>,V<n2>,V<n3>” または ”XY”,”YZ”,”ZX”,”XYH”,”YZH”,”ZXH”
または “V(<*x*>,<*y*>,<*z*>),V(…),V(…)”
ex. “V100,V101,V102”
但し, ”XY”,”YZ”,”ZX”,”XYH”,”YZH”,”ZXH”は VT_BSTR 型でのみ対応.

・ vntPivot(回転中心)の場合
“V<n4>” または ”V(<*x*>,<*y*>,<*z*>)”
ex. “V103”

(例 1) Rotate “V1,V2,V3”, 45.8, “V4”, “@E” ‘ ROTATE V1,V2,V3, @E 45.8, V4
(例 2) Rotate “V(0,0,1),V(0,1,0),V(0,0,0)”, 30.0, “V(0,0,0)”, “@E,pose=1,NEXT”
(例 3) Rotate “XY”, 90.0, “V(0,0,0)”, “@P”
(例 4) Rotate “XYH”, -45.0, “V(250,0,0)”, “@150”

回転面は 3 つの V 型変数の番号を指定して, その 3 点の XYZ 座標からベース座標基準の平面を作ります. 引数 vntRotSuf には BSTR(文字列)型でカンマ (またはスペース, タブ) 区切りの 3 つの V 型変数を指定します.

回転中心 vntPivot には BSTR(文字列)型で 1 つのベクトル型の変数を指定します.

4.2.38. CaoRobot::Speed メソッド

RobSlave 2.330

ロボットの内部移動速度を指定します.

このメソッドは PAC 言語の SPEED,JSPEED 命令に対応します.

外部移動速度に関しては CaoRobot::Execute の“ExtSpeed”を参照してください.

以下に, Speed の仕様を示します.

書式 Speed <lAxis:LONG >, <fSpeed:FLOAT>

lAxis	:	[in]	軸番号 -1:手先速度(SPEED), 0:全軸速度(JSPEED)
fSpeed	:	[in]	速度

4.2.39. CaoRobot::Execute メソッド

2.330

Execute メソッドは CaoRobot クラスで対応していないロボット特有の動作コマンドをユーザ自身で定義し実装できるようにする機能を提供します.

書式 [<vntRet:VARIANT> =] Execute(<bstrCmd:BSTR > [,<vntParam:VARIANT>])

bstrCmd	:	[in]	コマンド
vntParam	:	[in]	パラメータ
vntRet		[out]	返り値

現在指定可能なコマンドの一覧を示します.

表 4-9: CaoRobot クラスの Execute コマンド実装一覧[9]

コマンド	パラメータ	返り値	PAC 言語/備考
UserExt <コマンド番号>[, <パラメータ 1 >[, <パラメータ 2>…]]	<コマンド番号:VT_I4>, <パラメータ n:VT_R4>	VT_R4 | VT_ARRAY	拡張コマンドを実行します.
GetSrvData	<データ番号:VT_I4>	VT_R4 | VT_ARRAY	GetSrvData ロボット軸のサーボ内部データを取得します.
GetJntData	<データ番号:VT_I4>, <軸番号:VT_I2>	VT_R4	GetJntData 指定軸のサーボ内部データを取得します.
Approach	<補間指定:VT_I2(=1:P, 2:L)>, <基準位置:POSEDATA-C0>, <[パス] アプローチ長:POSEDATA-C2> [,オプション:VT_BSTR “NEXT”]	なし	APPROACH <補間指定>, <基準位置変数型><基準位置変数番号>, <パス> <アプローチ長>[,NEXT] ツール座標系指定の絶対動作を行ないます. RobSlave
Depart	<補間指定:VT_I2(=1:P, 2:L)>, <[パス] デパート長:POSEDATA-C2> [,オプション:VT_BSTR “NEXT”]	なし	DEPART <補間指定>, <パス> <デパート長>[,NEXT] ツール座標系指定で相対動作を行ないます. RobSlave
ExtSpeed	<外部速度:VT_R4 (=0.1 ～ 100.0)> , <外部加速度:VT_R4 (=0.0001～100.0)>, <外部減速度:VT_R4 (=0.0001～100.0)>	なし	外部速度,加速度, 減速度設定 外部速度設定を行います.

[9] POSEDATA 型に関しては「付録 A.POSEDATA 型定義」を参照してください.

SetSplinePoint	<自由曲線軌道番号:VT_I2 (1～20)>, <通過点:POSEDATA-C0(P,Jのみ)>	なし	SETSPLINEPOINT <自由曲線軌道番号(1～20)>, <通過点> 自由曲線動作の通過点を登録します. RobSlave
GetSplinePoint	<自由曲線軌道番号:VT_I2 (1～20)>, <通過点番号:VT_I4>	<P 型 :VT_R4 | VT_ARRAY >	< 出 力 P 変 数 >=GETSPLINEPOINT <自由曲線軌道番号(1～20)>,<通過点番号> 登録済の自由曲線動作の通過点を取り出します. RobSlave
ClrSplinePoint	<自由曲線軌道番号:VT_I2 (1～20)>	なし	SETSPLINEPOINT <自由曲線軌道番号(1～20)> 自由曲線動作の通過点をすべて消去します. RobSlave
WaitSplinePoint	<通過点番号:VT_I4>, <待ち条件:VT_I2 (0:指令値,0 以外:ｴﾝｺｰﾀﾞ値)>	なし	xdWAITSPLINE <通過点番号>,<待ち条件(0:指令値,0 以外:ｴﾝｺｰﾀﾞ値)> 自由曲線が指定した通過点を通過するのを待ちます. RobSlave
WaitMotionEnd	なし	なし	ロボット動作完了待ちを行います. RobSlave
MotionSkip	なし	なし	MotionSkip 実行中の動作命令を中断します. RobSlave
MotionComp	なし	VT_I2 0:動作中 1:動作完了	MotionComp 動作命令実行が完了したかどうかを判断します. RobSlave

Motor	<状態:VT_I2(1:ON,0:OFF)>	なし	Motor ON/OFF モータON/OFF モータ電源 ON/OFF を行ないます. RobSlave
DefTool	<no:VT_I2>, <ユーザー座標:POSEDATA-C0(P型のみ)>	なし	TOOL 変数を設定します. RobSlave
DefWork	<no:VT_I2>, <ユーザー座標:POSEDATA-C0(P型のみ)> [,<0:標準/1:固定>]	なし	WORK 変数を設定します. RobSlave
DefArea	<no:VT_I2>, <中心位置と角度:POSEDATA-C0(P 型のみ)>, <範囲:POSEDATA-C1(V 型のみ)>, <io:VT_I4>,<pos:VT_I4>,<err:VT_I4>,<enable:VT_I2(0,1)>,	なし	AREA 変数を設定します. RobSlave
Draw	<補間指定:VT_I2(=1:P, 2:L)>, <[パス] 移動量:POSEDATA-C1>, [,オプション:VT_BSTR “NEXT”]	なし	DRAW <補間指定>, <パス>(<x>,<y>,<z>) [,NEXT] ワーク作業座標系指定で相対動作を行ないます. RobSlave
DriveAEx	<[パス] (<軸番号1>,<軸座標1>):POSEDATA-C3>, [<(<軸番号 2>,<軸座標 2>):POSEDATA-C3>, : [<(<軸番号 8>,<軸座標 8>):POSEDATA-C3>]] [,オプション:VT_BSTR “NEXT”]	なし	DRIVEA <@パス開始変位>(<軸番号1>, <軸座標1>),(<軸番号 2>, <軸座標 2>),..,(<軸番号 8>, <軸座標 8>)[,NEXT] 各軸の絶対動作を行ないます. RobSlave

DriveEx	<[パス] (<軸番号1>,<軸座標1>):POSEDATA-C3>, [<(<軸番号 2>,<軸座標 2>):POSEDATA-C3>, : [<(<軸番号 8>,<軸座標 8>):POSEDATA-C3>]] [,オプション:VT_BSTR “NEXT”]	なし	DRIVE <@パス開始変位>(<軸番号1>, <軸座標1>), (<軸番号 2>, <軸座標 2>),.., (<軸番号 8>, <軸座標 8>)[, NEXT] 各軸の相対動作を行ないます. RobSlave
RotateH	<[パス] 相対回転角度:POSEDATA-C2>, [,オプション:VT_BSTR “NEXT”]	なし	ROTATEH [@<パス開始変位>] <アプローチベクトル回りの相対回転角>[,NEXT] アプローチベクトルを軸とした, 回転動作を行ないます. RobSlave
Arrive	<動作割合:VT_R4>	なし	ARRIVE <動作割合> 動作命令の全移動距離に対する動作割合を設定する事によって, ロボットが設定した動作割合に到達するまでプログラムを待機させます. RobSlave
PosClr	<軸番号:VT_I2>	なし	POSCLR <軸番号> 軸の現在位置を強制的に 0mm または 0 度にします. RobSlave
Interrupt	<動作:VT_I2(0:OFF 1:ON)>	なし	INTERRUPT ON/OFF ロボットの動作を中断します. RobSlave
ST_aspACLD	<先端負荷質量:VT_R4>, <先端負荷重心位置 X 座標:VT_R4>, <先端負荷重心位置Y 座標:VT_R4>, <先端負荷重心位置 Z 座標:VT_R4>	なし	ST_aspACLD 内部負荷条件値を変更します. RobSlave

ST_aspChange	<モード:VT_I2> 0： 無効 1 : PTP のみ有効 2 : CP のみ有効 3 : PTP, CP ともに有効	なし	ST_aspChange 最適可搬質量設定モードの内部モードを選択します. RobSlave
ST_SetGravity	なし	なし	ST_SetGravity 各関節の静荷重(重力トルク)を補正し，重力バランスを設定します. RobSlave
ST_ResetGravity	なし	なし	ST_ResetGravity 重力バランスを無効にします. RobSlave
ST_SetGrvOffset	なし	なし	ST_SetGrvOffset 各関節の重力トルクより重力補償値を補正します. RobSlave
ST_ResetGrvOffset	なし	なし	ST_ResetGrvOffset 重力補償値の補正を無効にします. RobSlave
ST_SetCurLmt	< 軸 番 号 :VT_I2> ， < 設 定 値:VT_R4>	なし	ST_SetCurLmt 指定した軸のモータ電流値を制限します. RobSlave
ST_ResetCurLmt	<軸番号:VT_I2>	なし	ST_ResetCurLmt 指定した軸のモータ電流制限を解除します. RobSlave
ST_SetEralw	< 軸 番 号 :VT_I2> ， < 設 定 値:VT_R4>	なし	ST_SetEralw 指定した軸の偏差許容値を変更します. RobSlave

ST_ResetEralw	<軸番号:VT_I2>	なし	ST_ResetEralw 指定した軸の偏差許容値をデフォルト値に戻す. RobSlave
ST_OnSrvLock	<指定軸:VT_I2>	なし	ST_OnSrvLock 指定した軸をサーボロック状態にします.（4軸ロボット専用） RobSlave
ST_OffSrvLock	<指定軸:VT_I2>	なし	ST_OffSrvLock 指定した軸のサーボロックを解除します.（4軸ロボット専用） RobSlave
ST_SetCompControl	なし	なし	ST_SetCompControl 力制限機能を有効にします.(6軸専用命令) RobSlave
ST_SetCompFControl	なし	なし	ST_SetCompFControl 力制限機能を有効にします.(6軸専用命令) RobSlave
ST_ResetCompControl	なし	なし	ST_ResetCompControl 力制限機能を無効にします.(6軸専用命令) RobSlave
ST_SetFrcCoord	<設定値:VT_R4>	なし	ST_SetFrcCoord 力制限設定座標系を選択します. (6軸専用命令応) RobSlave
ST_SetFrcLimit	<X 方向制限割合:VT_R4>,<Y 方向制限割合:VT_R4>,<Z 方向制限割合:VT_R4>,<X 回り制限割合:VT_R4>,<Y 回り制限割合:VT_R4>,<Z 回り制限割合:VT_R4>	なし	ST_SetFrcLimit 力制限割合を設定します. (6軸専用命令) RobSlave

ST_ResetFrcLimit	なし	なし	ST_ResetFrcLimit 力制限割合を初期化します. (6軸専用命令) RobSlave
ST_SetCompRate	<X 方向柔らかさ:VT_R4>,<Y 方向柔らかさ:VT_R4>,<Z 方向柔らかさ:VT_R4>,<X 回り柔らかさ:VT_R4>,<Y 回り柔らかさ:VT_R4>,<Z 回り柔らかさ:VT_R4>	なし	ST_SetCompRate 力制限時の柔らかさの割合を設定します. (6軸専用命令) RobSlave
ST_ResetCompRate	なし	なし	ST_ResetCompRate 力制限時の柔らかさの割合を初期化します. (6軸専用命令) RobSlave
ST_SetFrcAssist	<X 方向オフセット力:VT_R4>,<Y 方向オフセット力:VT_R4>,<Z 方向オフセット力:VT_R4>,<X 回りオフセットモーメント:VT_R4>,<Y 回りオフセットモーメント:VT_R4>,<Z 回りオフセットモーメント:VT_R4>	なし	ST_SetFrcAssist 力制限時のオフセット力を設定します. (力制限特殊機能ステートメント)(6軸専用命令) RobSlave
ST_ResetFrcAssist	なし	なし	ST_ResetFrcAssist 力制限時のオフセット力を初期化します. (力制限特殊機能) (6軸専用命令) RobSlave
ST_SetCompJLimit	<J1 電流制限値:VT_R4>,< J2 電流制限値:VT_R4>,< J3 電流制限値 :VT_R4>,<J4 電流制限値 :VT_R4>,< J5 電流制限値 :VT_R4>,< J6 電流制限値:VT_R4>	なし	ST_SetCompJLimit 力制限時の電流制限値を設定します. (力制限特殊機能)(6軸専用命令) RobSlave

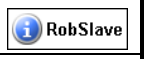
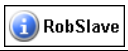
ST_ResetCompJLimit	なし	なし	ST_ResetCompJLimit 力制限時の電流制限値を初期化します. (力制限特殊機能)(6軸専用命令) RobSlave
ST_SetCompVMode	なし	なし	ST_SetCompVMode 力制限時の速度制御モードを設定します. (力制限特殊機能)(6軸専用命令) RobSlave
ST_ResetCompVMode	なし	なし	ST_ResetCompVMode 力制限時の速度制御モードを無効にします. (力制限特殊機能)(6軸専用命令) RobSlave
ST_SetCompEralw	<X 方向偏差許容値:VT_R4>,<Y 方向偏差許容値:VT_R4>,<Z 方向偏差許容値:VT_R4>,<X 回り偏差許容値:VT_R4>,<Y 回り偏差許容値 :VT_R4>,<Z 回り偏差許容値:VT_R4>	なし	ST_SetCompEralw 力制限時のツール端の位置, 姿勢偏差許容値を設定します. (6軸専用命令) RobSlave
ST_ResetCompEralw	なし	なし	ST_ResetCompEralw 力制限時のツール端の位置, 姿勢偏差許容値を初期化します. (6軸専用命令) RobSlave
ST_SetDampRate	<X 方向粘性割合:VT_R4>,<Y 方向粘性割合:VT_R4>,<Z 方向粘性割合 :VT_R4>,<X 回り粘性割合 :VT_R4>,<Y 回り粘性割合 :VT_R4>,<Z 回り粘性割合:VT_R4>	なし	ST_SetDampRate 力制限時の粘性割合を設定します. (6軸専用命令) RobSlave
ST_ResetDampRate	なし	なし	ST_ResetDampRate 力制限時の粘性割合を初期化します. (6軸専用命令) RobSlave

ST_SetZBalance	なし	なし	ST_SetZBalance Z 軸, T 軸の重力補償値を設定します. (4軸専用) RobSlave
ST_ResetZBalance	なし	なし	ST_ResetZBalance 重力補償値の補正を無効にします.（4軸専用） RobSlave
DELAY	<時間:VT_I2>	なし	単位 msec 指定した時間の間, プログラムの進行を停止します. RobSlave
SYSSTATE	なし	なし	SYSSTATE コントローラのステータスを取得します. RobSlave
J2P	<J 型:POSEDATA-C0>	<P 型 :VT_R4 | VT_ARRAY >	J2P ジョイント型からポジション型に変換します. RobSlave
J2T	<J 型: POSEDATA-C0>	<T 型 :VT_R4 | VT_ARRAY >	J2T ジョイント型から同次変換型に変換します. RobSlave
P2J	<P 型: POSEDATA-C0>	<J 型 :VT_R4 | VT_ARRAY >	P2J ポジション型からジョイント型に変換します. RobSlave
P2T	<P 型: POSEDATA-C0>	<T 型 :VT_R4 | VT_ARRAY >	P2T ポジション型から同次変換型に変換します. RobSlave
T2J	<T 型: POSEDATA-C0>	<J 型 :VT_R4 | VT_ARRAY >	T2J 同次変換型からジョイント型に変換します. RobSlave

T2P	<T 型: POSEDATA-C0>	<P 型 :VT_R4 | VT_ARRAY >	T2P 同次変換型からポジション型に変換します. RobSlave
TINV	<T 型: POSEDATA-C0>	<T 型 :VT_R4 | VT_ARRAY >	TINV 同次変換型の逆行列を計算します. RobSlave
NORMTRN	<T 型: POSEDATA-C0>	<T 型 :VT_R4 | VT_ARRAY >	NORMTRN 同次変換型の正規化計算を行います. RobSlave
TMUL	<T 型 n1 : POSEDATA-C0>, <T 型 n2 : POSEDATA-C0>	<T 型 :VT_R4 | VT_ARRAY >	=T<n1> * T<n2> 行列の計算をします. RobSlave
DEVH	<P 型 n1: POSEDATA-C0>, <P 型 n2: POSEDATA-C0>	<P 型 :VT_R4 | VT_ARRAY >	**ツール座標系を基準とした移動先の座標算出します**. n1>1の場合:**指定座標** =P<n1> + (P<n2>.x, P<n2>.y, P<n2>.z, P<n2>.rx, P<n2>.ry, P<n2>.rz)H n1=0の場合: **目標位置** =DESTPOS + (P<n2>.x, P<n2>.y, P<n2>.z, P<n2>.rx, P<n2>.ry, P<n2>.rz)H n1=-1の場合: **現在位置** =CURPOS + (P<n2>.x, P<n2>.y, P<n2>.z, P<n2>.rx, P<n2>.ry, P<n2>.rz)H RobSlave

DEV	<P 型 n1: POSEDATA-C0>, <P 型 n2: POSEDATA-C0>	<P 型 :VT_R4 | VT_ARRAY >	**ベース座標系を基準とした移動先の座標算出します.** n1>1の場合: **指定座標** =P<n1> + (P<n2>.x, P<n2>.y, P<n2>.z, P<n2>.rx, P<n2>.ry, P<n2>.rz) n1=0の場合: **目標位置** =DESTPOS + (P<n2>.x, P<n2>.y, P<n2>.z, P<n2>.rx, P<n2>.ry, P<n2>.rz) n1=-1の場合: **現在位置** =CURPOS + (P<n2>.x, P<n2>.y, P<n2>.z, P<n2>.rx, P<n2>.ry, P<n2>.rz) RobSlave
TrackDataInitialize	<初期化モード:VT_I2>	なし	TRACKDATAINITIALIZE コンベアトラッキングデータバッファ内のデータを初期化します. RobSlave
TrackDataSet	<コンベア番号:VT_I2>,<認識ワーク個数:VT_I2>,<認識ワーク位置 P 型: POSEDATA-C0 >	なし	TRACKDATASET コンベアトラッキングバッファにデータを保存します. RobSlave
TrackDataGet	<コンベア番号:VT_I2>,<データ番号:VT_I2>	<データ残り個数:VT_I2>,<認識ワーク位置 P 型 :VT_R4|VT_ARRAY >	TRACKDATAGET コンベアトラッキングバッファよりデータを取得します. RobSlave

TrackDataInfo	<コンベア番号:VT_I2>,<データ番号:VT_I2>	<認識時のエンコーダ値:VT_I4>,<取得成否:VT_I2>,<認識時のワーク位置 P 型 :VT_R4|VT_ARRAY>	TRACKDATAINFO コンベアトラッキングバッファ内の情報を取得します. RobSlave
TrackDataNum	<コンベア番号:VT_I2>	<保存データ数:VT_I2>	TRACKDATANUM TRACKDATASET にて保持されたデータ個数を取得します. RobSlave
CurTrackPos	<コンベア番号:VT_I2>,<認識ワーク位置 P 型: POSEDATA-C0>,<モード:VT_I2>	<トラッキング対象のワーク位置 P 型 :VT_R4|VT_ARRAY >	CURTRACKPOS トラッキング対象のワーク位置を P 型にて取得します. RobSlave
CurTrackPosEx	<コンベア番号:VT_I2>,<認識ワーク位置 P 型: POSEDATA-C0>,<モード:VT_I2>	<エラー情報:VT_I2>,<トラッキング対象のワーク位置 P 型 :VT_R4|VT_ARRAY>	CURTRACKPOSEX トラッキング対象のワーク位置を P 型にて取得します. RobSlave
WaitTrackMove	<コンベア番号:VT_I2>,<認識ワーク位置 P 型: POSEDATA-C0>,<タイムアウト時間:VT_I2>	なし	WAITTRACKMOVE トラッキング対象ワーク位置がトラッキング開始範囲内に入るのを待ちます. RobSlave
WaitTrackMoveEx	<コンベア番号:VT_I2>,<認識ワーク位置 P 型: POSEDATA-C0>,<タイムアウト時間:VT_I2>	<エラー情報 : VT_I2>	AITTRACKMOVEEX トラッキング対象ワーク位置がトラッキング開始範囲内に入るのを待ちます. RobSlave
CurTrackSpd	<コンベア番号:VT_I2>	<コンベア速度 : VT_I4>	CURTRACKSPD <コンベア番号>で指定したコンベアの速度を取得します. RobSlave

CalcWorkPos	<コンベア番号:VT_I2>,<認識ワーク位置 P 型: POSEDATA-C0>,<認識エンコーダ値:VT_I4>	<現在のワーク位置 P 型 : VT_R4 | VT_ARRAY>	CALCWORKPOS 指定したワークの現在位置を取得します. 
SetTrackMove	<コンベア番号:VT_I2>	なし	SETTRACKMOVE 指定したコンベアに対してトラッキング動作を開始します. 
ResetTrackMove	なし	なし	RESETTRACKMOVE トラッキング動作モードから通常動作モードに切り替えます. 
SetTrackStartArea	<コンベア番号:VT_I2>,<コンベア上流(-)側トラッキング開始位置:VT_I2>,<コンベア下流(+)側トラッキング開始位置:VT_I2>	なし	SETTRACKSTARTAREA WAITTRACKMOVE 時のトラッキング開始範囲を設定します. 
ClearSrvLog	なし	なし	ClearSrvMonitor 単軸制御ログをクリアします.
StartSrvLog	なし	なし	StartSrvMonitor 単軸制御ログの取得を開始します.
StopSrvLog	なし	なし	StopSrvMonitor 単軸制御ログの取得をストップします.
SetSrvLogCond	<軸番号:VT_I2>,<データ番号 1:VT_I2>,<データ番号 2:VT_I2>,<サンプリングタイム:VT_I2>	なし	単軸制御ログ取得の条件を設定します. SetMonitorCond
GetSrvLogCond	なし	<軸番号:VT_I2>,<データ番号 1:VT_I2>,<データ番号 2:VT_I2>,<サンプリングタイム:VT_I2>>,<サンプリング数:VT_I2>	単軸制御ログ取得の条件を取得します.

GetSrvLog	なし	<サーボデータ (2 次元配列) : VT_R4 | VT_ARRAY>	単軸制御ログを取得します.
TakeArm	<アームグループ:VT_I4> [, <Keep オプション:VT_I4>]	なし	TakeArm <Keep オプション>:= 0(省略時): 内部速度を 100 に設定する, 1: 内部速度を現在のまま保持する アームグループを取得します. RobSlave
GiveArm	なし	なし	GiveArm 現在取得しているアームグループを解放します. RobSlave
SetHighPathAccuracy	なし	なし	高軌跡制御機能モードを ON します.
ResetHighPathAccuracy	なし	なし	高軌跡制御機能モードを OFF します.
SetSingularAvoid	<モード:VT_I2 >	なし	特異点回避機能の ON/OFF をします. <モード> = 0:OFF, 1:ON
FigAprp	<基準位置:POSEDATA-C0(P 型のみ)>, <アプローチ長:VT_R4>	<形態(FIG):VT_I2>	FIGAPRP PTP 動作可能なアプローチ位置と基準位置の形態を算出します. RobSlave
FigAprl	<基準位置:POSEDATA-C0(P 型のみ)>, <アプローチ長:VT_R4>	<形態(FIG):VT_I2>	FIGAPRL CP 動作可能なアプローチ位置と基準位置の形態を算出します. RobSlave
GetCollisionForce	なし	< 各軸値 : VT_R4|VT_ARRAY>	衝突検出 外力最大値を取得します.
ClearCollisionForce	なし	なし	衝突検出 外力最大値をクリアします.

ResetCollisionJnt	<軸番号:VT_I2>	なし	衝突検出 指定軸の衝突判定を無効化します.
SetCollisionJnt	<軸番号:VT_I2>	なし	衝突検出 指定軸の衝突判定を有効化します.
SetCollisionLevel	<軸番号:VT_I2>, <検出レベル:VT_I4(1～500)>	なし	衝突検出 検出レベルを設定します.
SetExtForeceDetect	<値:VT_I2(0/1)>	なし	衝突検出 有効無効を切替ます.
RPM	<軸番号:VT_I2>, <RPM の値:VT_R4>	<SPEED 値(%):VT_R4>	指定した軸の RPM(回転数))からCP 動作時の最大内部速度に対する割合(%)に変換します.
MPS	＜MPS の値:VT_R4＞	<SPEED 値(%):VT_R4>	指定した速度 MPS(mm/sec)からCP 動作時の最大内部速度に対する割合(%)に変換します.

CaoRobot クラスの Execute メソッドの引数は, コマンド番号+パラメータを VARIANT 配列で指定します.

例:

```
Dim vRes as Variant
vRes = caoRobot.Execute("GetJntData",Array(1, 6 ) )  ' 6 軸のモータ速度現在値[rpm]
caoRobot.Execute "ExtSpeed",Array(50.0, 25.0, 25.0 )  ' 外部速度=50%,加速度=25%,減速度=25%
caoRobot.Execute "APPROACH", Array(1, "P11", "@P 100", "NEXT") 'APPROACH P,P11,@P 100, NEXT
```

ユーザによる独自コマンドの拡張は UserExtension.pac ファイルの UserExtention プログラムに追加コマンドを定義し, それに対応する実行コードを記述することで行うことができます. 拡張コマンドは, コマンド名"UserExt"を指定して呼び出します.

下記に GETSRVDATA ,GETJNTDATA コマンドを追加した際の具体的の例を示します.

1. UserExtension.pac, RobSlave.h を取得する.
   <ORiN2 ﾌｫﾙﾀﾞ>¥CAO¥ProviderLib¥DENSO¥NetwoRC¥Bin¥ 以下にあります.
2. 追加するコマンド番号を定義する.
   RobSlave.h に新たに追加するコマンド番号を定義してください.

```
'User Extention Commands Def.
'-------------------------------------------------------
#define RBS_CMDEX_APPROACH_L                    (RBS_CMD_EXTENTION +1)
#define RBS_CMDEX_APPROACH_P                    (RBS_CMD_EXTENTION +2)
#define RBS_CMDEX_GETSRVDATA                    (RBS_CMD_EXTENTION +3)
#define RBS_CMDEX_GETJNTDATA                    (RBS_CMD_EXTENTION +4)
```

RBS_CMD_EXTENTION の値は 10000 です. RobSlave.h で定義されています.

3. 追加コマンドに対する実行コードを記述する.

Pv.x (= T[RBS_IDX_COMMAND].X) に実行コマンドの ID が格納されていますのでこの値を参照して実行コードを SELECT-CASE 文で分岐します. コマンドの引き値は pv.Y,pv.Z,ov.X,ov.Y,ov.Z,av.X,av.Y,av.Z (= T[RBS_IDX_COMMAND].Y,Z,…)に順に格納されていますので必要に応じて参照します.

```
' User Extention Commands Impl.
'-------------------------------------------------------------------------------
PROGRAM UserExtention(pv as VECTOR, ov as VECTOR, av as VECTOR)
  DEFSNG ret
  DEFINT index, path
  DEFINT var 型, varindex
  DEFSNG length
  DEFJNT jv

  select case POSX(pv)
  '-----------------------------------------------------------
  case RBS_CMDEX_GETJNTDATA          'GetJntData(<Index>, <JontNo>)
  '-----------------------------------------------------------
    LETX T[RBS_IDX_RESULT] = GetJntData ( POSY(pv), POSZ(pv) )
    I[RBS_IDX_STATE] = RBS_STA_RETVAL ' Return value
  case RBS_CMDEX_GETSRVDATA          'GetSrvData(<Index>)
    '-----------------------------------------------------------
    jv = GetSrvData ( POSY(pv) )
#ifdef  __VERTICAL_ROBOT__
    T[RBS_IDX_RESULT] = ( JOINT(1,jv), JOINT(2,jv), JOINT(3,jv), JOINT(4,jv), JOINT(5,jv),
JOINT(6,jv), 0,0,0, -1 )
#else
    T[RBS_IDX_RESULT] = ( JOINT(1,jv), JOINT(2,jv), JOINT(3,jv), JOINT(4,jv), 0,0,0,0,0, -1 )
#endif
    I[RBS_IDX_STATE] = RBS_STA_RETVAL ' Return value
    '-----------------------------------------------------------
  case else
    I[RBS_IDX_STATE] = RBS_STA_ERR
  end select
```

値を返さない場合は I[RBS_IDX_STATE] = RBS_STA_DONE として下さい.

値を返す場合は例のように T[RBS_IDX_RESULT] に返したい値を代入して I[RBS_IDX_STATE] = RBS_STA_RETVAL とします. この場合, CaoRobot クラスの Execute メソッドの返り値として T[RBS_IDX_RESULT]の各要素を VT_R4(単精度実数)として順に VARIANT 配列に格納した値が返ります.

4. UserExtention.pac ファイルの CRC32 情報を更新する.

RobSlave.h に UserExtension.pac の CRC32 情報が記録されています.

```
'CRC code
#define RBS_SLVCRC_CODE[10]          &H62cb2dc4
#define RBS_EXTCRC_CODE              &H1e5d8368
```

この情報を更新してしないと動作命令実行時にエラーが発生しますので UserExtension.pac の CRC32 を計算して RobSlave.h の値を更新してください.

CRC32 の値は CaoFile クラスの@CRC システム変数で取得することが出来ます.

5. CaoRobot クラスの Execute メソッドの Command に“UserExt”を指定して実行する.

[10] RobSlave.pac の CRC32 の値

Command＝”UserExt“

Parameter＝<追加コマンド番号>,<引数 1>,<引数 2>,… (VARIANT 配列)

例: vRes = CaoRobot.Execute(“UserExt”, Array(10004, 1, 6)) ‘= GetJntData(1,6)

【必要条件】

現在の仕様では CaoRobot クラスの Execute メソッドを実行するには予めロボットコントローラ RC7 で”RobSlave.pac”及び, ”UserExtention.pac”プログラムが実行されていなくてはなりません. RobSlave.pac, UserExtention.pac は ORiN2¥CAO¥ProviderLib¥DENSO¥NetwoRC¥Bin にあります.

4.2.40. CaoTask::AddVariable メソッド

CaoTask クラスの AddVariable メソッドの引数は, システム変数名を指定します.

実装されているシステム変数の一覧は表 4-13 を参照して下さい.

4.2.41. CaoTask::get_VariableNames プロパティ

AddVariable メソッドで指定できる変数名とシステム変数名の一覧を取得します.

4.2.42. CaoTask::Start メソッド

2.330

オブジェクトに対応している PAC プログラムを実行します.

以下に, Start の仕様を示します.

書式 Start <lMode:LONG>, <bstrOpt:BSTR>

lMode	:	[in]	開始モード 1:1 サイクル実行, 2:連続実行, 5:コンティ起動
bstrOpt	:	[in]	オプション(未使用)

開始モードで“5:コンティ起動”を指定したときは, ロボットコントローラ内のコンティ停止中のタスクに対してコンティ起動を行います.

4.2.43. CaoTask::Stop メソッド

オブジェクトに対応している PAC プログラムを停止します.

以下に, Stop の引数仕様を示します.

書式 Stop <lMode:LONG>, <bstrOpt:BSTR>

lMode	:	[in]	停止モード 0:デフォルト停止, 1:瞬時停止, 2:ステップ停止, 3:サイクル停止, 4:初期化停止, 5:コンティ停止
bstrOpt	:	[in]	オプション(未使用)

停止モードで“0:デフォルト停止”を指定したときは, “1:瞬時停止”として扱われます.

停止モードで“5:コンティ停止”を指定したときは, ロボットコントローラ内の起動中の全タスクをコンティ停止します.

4.2.44. CaoVariable::get_Value プロパティ

オブジェクトに対応している変数の値を取得します.

変数の実装状況およびデータ型は 4.3 を参照して下さい.

4.2.45. CaoVariable::put_Value プロパティ

オブジェクトに対応している変数に値を設定します.

変数の実装状況およびデータ型は 4.3 を参照して下さい.

4.2.46. CaoVariable::put_ID プロパティ

オブジェクトに対応している変数の番号を設定します.

オブジェクト作成時に＊を指定した場合のみ設定できます.

使用例 (VB):

```
Set objIO = caoCtrl.AddVariable( “IO[*]” )      ‘IO のワイルドカード指定
objIO.ID = 128                                  ‘128 番を指定
boolValue = objIO.Value                         ‘IO[128]の値を取得
```

4.2.47. CaoVariable::get_ID プロパティ

オブジェクトに対応している変数の番号を取得します.

4.2.48. CaoVariable::get_Microsecond プロパティ

オブジェクトに対応している変数のタイムスタンプを取得します.

タイムスタンプはコントローラ立上げ時からの 500 μ sec 単位でのカウンタ値を意味し， 0から順に500 μ sec 単位でカウントアップします.

0,1,2→2147483647(0x7fffffff) ,-2147483648(0x80000000) ,-2147483647(0x80000001) →-2,-1,0 と順に巡回します.

※オブジェクトがタイムスタンプに対応していない場合は 0 を返します.

4.2.49. CaoMessage::Clear メソッド

CaoController クラスの Clear メソッドを使って，発生中のエラーをクリアすることができます.

4.3. 変数一覧

4.3.1. コントローラクラス

表 4-10 コントローラクラス ユーザ変数一覧

変数名	データ型	説明	属性	
			get	put
I	VT_I4	I 型変数. 変数名の後ろに変数番号(0～)を指定します.	○	○
F	VT_R4	F 型変数. 変数名の後ろに変数番号(0～)を指定します.	○	○
D	VT_R8	D 型変数. 変数名の後ろに変数番号(0～)を指定します.	○	○
V	VT_ARRAY | VT_R4	V 型変数. 変数名の後ろに変数番号(0～)を指定します. データの要素数は 3	○	○
P	VT_ARRAY | VT_R4	P 型変数. 変数名の後ろに変数番号(0～)を指定します. データの要素数は 7	○	○
J	VT_ARRAY | VT_R4	J 型変数. 変数名の後ろに変数番号(0～)を指定します. データの要素数は 6	○	○
T	VT_ARRAY | VT_R4	T 型変数. 変数名の後ろに変数番号(0～)を指定します. データの要素数は 10	○	○
S	VT_BSTR	S 型変数. 変数名の後ろに変数番号(0～)を指定します.	○	○
IO	VT_BOOL	IO 型変数. 変数名の後ろに変数番号(0～)を指定します.	○	○
TOOL	VT_ARRAY | VT_R4	TOOL. 変数名の後ろに変数番号(0～)を指定します.	○	○
WORK	VT_ARRAY | VT_R4	WORK. 変数名の後ろに変数番号(0～)を指定します.	○	○
AREA	VT_ARRAY | VT_R4	AREA. 変数名の後ろに変数番号(0～)を指定します. (V[0],..,V[8],IO,Pos,Err,Enable) V[0]～V[8]: エリア IO : ポート番号 Pos : ポジション番号 Err : エラーフラグ Enable : 機能の有効無効	○	○

_ITP	VT_I4	ITPCNF. 変数名の後ろに変数番号(0～)を指定します.	○	○
_PAC	VT_I4	PACCNF. 変数名の後ろに変数番号(0～)を指定します.	○	○
_DIO	VT_I4	DIOCNF. 変数名の後ろに変数番号(0～)を指定します.	○	○
_ARM	VT_I4	ARMCNF. 変数名の後ろに変数番号(0～)を指定します.	○	○
_SRV	VT_I4	SRVCNF. 変数名の後ろに変数番号(0～)を指定します.	○	○
_SPD	VT_I4	SPDCNF. 変数名の後ろに変数番号(0～)を指定します.	○	○
_VIS	VT_I4	VISCNF. 変数名の後ろに変数番号を指定します.	○	○
_COM	VT_I4	COMCNF. 変数名の後ろに変数番号を指定します.	○	○

表 4-11 コントローラクラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@CURRENT_TIME	VT_DATE	コントローラ保有の現在時間	○	○
@FREE_USER_MEM	VT_I4	空きユーザメモリのバイト数	○	-
@NORMAL_STATUS	VT_BOOL	true=正常, false=異常（エラー発生中）	○	-
@AUTO_MODE	VT_BOOL	true=自動モード, false=自動モード以外	○	-
@MODE	VT_I2	1: 手動, 2: ティーチチェック, 3: 内部自動, 4: 外部自動	○	-
@BUSY_STATUS	VT_BOOL	true=プログラム動作中, false=プログラム停止中	○	
@EMERGENCY_STOP	VT_BOOL	true=非常停止中 false=非常停止中ではありません	○	-
@ERROR_CODE	VT_I4	発生中のエラーの番号を 10 進数の値で取得します. エラーが発生していないときは, 0 を返します.	○	-

@ERROR_CODE_HEX	VT_BSTR	発生中のエラーの番号を4桁の16進数の大文字で取得します.	○	-
@ERROR_LEVEL	VT_I4	発生中のエラーのレベル	○	-
@ERROR_DESCRIPTION	VT_BSTR	発生中のエラーの内容	○	-
@MAKER_NAME	VT_BSTR	“DENSO CORPORATION”	○	-
@TYPE	VT_BSTR	“NetwoRC Controller”	○	-
@VERSION	VT_BSTR	コントローラのバージョン	○	-
@SERIAL_NO	VT_BSTR	コントローラのシリアル番号	○	-

4.3.2. ロボットクラス

表 4-12 ロボットクラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@CURRENT_POSITION	VT_ARRAY | VT_R4	ロボットの現在位置. 単位は任意 P 型変数.	○	-
@CURRENT_ANGLE	VT_ARRAY | VT_R4	ロボットの現在位置(各軸値). 単位は任意. J 型変数	○	-
@SERVO_ON	VT_BOOL	true=サーボ ON, false=サーボ OFF	○	-
@ZERO_RETURN_REQUIRED	VT_BOOL	true=原点復帰が必要, false=原点復帰は必要でない	○	-
@BUSY_STATUS	VT_BOOL	true=アーム動作中, false=アーム停止中	○	-
@TYPE_NAME	VT_BSTR	ロボットの形式 “VM-D”,”VS-D”,”VSS-D”,”DM-D”,”Unknown”	○	-
@TYPE	VT_I4	ロボットタイプデータ	○	-
@CURRENT_TRANS	VT_ARRAY | VT_R4	ロボットの現在位置. T 型	○	-
@CURRENT_TOOL	VT_I2	現在使用中のツール番号	○	-
@CURRENT_WORK	VT_I2	現在使用中のワーク番号	○	-

@SPEED	VT_R4	内部移動速度 2.330	○	-
@ACCEL	VT_R4	内部移動加速度 2.330	○	-
@DECEL	VT_R4	内部移動減速度 2.330	○	-
@JSPEED	VT_R4	内部軸速度 2.330	○	-
@JACCEL	VT_R4	内部軸加速度 2.330	○	-
@JDECEL	VT_R4	内部軸減速度 2.330	○	-
@EXTSPEED	VT_R4	外部移動速度	○	-
@EXTACCEL	VT_R4	外部移動加速度	○	-
@EXTDECEL	VT_R4	外部移動減速度	○	-
@HIGH_CURRENT_POSITION	VT_ARRAY | VT_R4	ロボットの現在位置. P 型変数. 2.330 動作仕様: マシンロック時以外はエンコーダ値を返します. (分解能:500 μ sec) マシンロック時はコントローラ内部の指令値を返します. (分解能:8msec) 現在位置取得時のタイムスタンプは CaoVariable クラスの Microsecond プロパティで取得できます. ※Ver2.90 未満のコントローラではタイムスタンプに対して未対応です. また, マシンロック状態では現在位置が不定になります.	○	-
@HIGH_CURRENT_ANGLE	VT_ARRAY | VT_R4	ロボットの現在位置(各軸値). J 型変数. 2.330 動作仕様に関しては@HIGH_CURRENT_POSITION を参照してください.	○	-
@HIGH_CURRENT_TRANS	VT_ARRAY | VT_R4	ロボットの現在位置. T 型. 2.330 動作仕様に関しては@HIGH_CURRENT_POSITION を参照してください.	○	-

4.3.3. タスククラス

表 4-13 タスククラス システム変数一覧

変数名	データ型	説明	属性	
			get	Put
@STATUS	VT_I4	タスクの状態. 1: DORMANT 2: READY 3: RUN 4: WAIT 6: SUSPEND 0: NON_EXISTENT	○	-
@PRIORITY	VT_I4	タスクの優先度.	○	-
@LINE_NO	VT_I4	現在実行中のメインプログラムの行番号.	○	-
@CYCLE_TIME	VT_I4	タスクの 1 サイクルの実行時間.	○	-
@START	VT_I4	タスクの開始. 値の意味は CaoTask::Start メソッドの Mode 引数と同じ. モードは1:1 サイクル実行, 2:連続実行, 3:1ステップ送り, ~~4:1ステップ 戻し~~, 5:コンティ起動 Start メソッドのようにオプションを指定することはできません. (注 2)“5:コンティ起動”を指定したときは, ロボットコントローラ内のコンティ停止中のタスクに対してコンティ起動を行います. 2.330	-	○
@STOP	VT_I4	タスクの停止. 値の意味は CaoTask::Stop メソッドの Mode 引数と同じ. モードは 0:デフォルト停止, 1:瞬時停止, 2:ステップ停止, 3:サイクル停止, 4:初期化停止, 5:コンティ停止 Stop メソッドのようにオプションを指定することはできません. デフォルト停止(0)は瞬時停止(1)に対応します. (注 2)“5:コンティ停止”を指定したときは, ロボットコントローラ内の起動中の全タスクをコンティ停止します.	-	○

4.3.4. ファイルクラス

表 4-14 ファイルクラス システム変数一覧

変数名	データ型	説明	属性	
			get	Put
@ACTIVE	VT_I4	使用/未使用の状態(コンパイル対象). 0: 未使用 1: 使用 2.330	○	○
@CRC	VT_I4	CRC32 2.330	○	-

5. ロボット動作命令の実行概要

2.330

CaoRobot クラスのロボット動作命令(Move, GoHome, Accelerate, Change, Speed, Execute メソッド)の実行はコントローラ内で動作する PAC プログラムである RobSlave(RobSlave.pac)と通信を行って，指示された PAC コマンドが実行されることで実現されています． NetwoRC プロバイダと RobSlave 間の通信にコントローラのグローバル変数 I[0],T[0],T[1]の変数が使用されます．I[0]は実行中のコマンドの状態（実行中，完了，エラー等）を表し，T[0]は NetwoRC から RobSlave へのコマンドと引数を渡すために使用されます．T[1]は RobSlave から NetwoRC へ値を返すために使用されます．

以下にロボット動作命令の実行手順を示します．

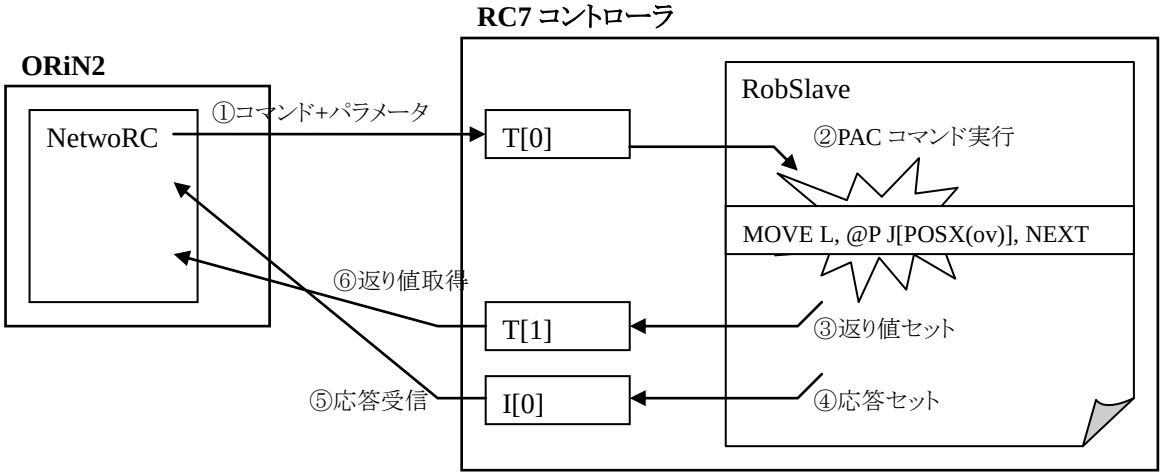


図 5-1 ロボット動作命令の変数指定実行手順

ロボット動作命令を定数(即値)指定で実行した場合は，P[2],J[2],T[2]， P[3],J[3],T[3]のグローバル変数は間接的に使用されます．

以下にロボット動作命令の実行手順を示します．

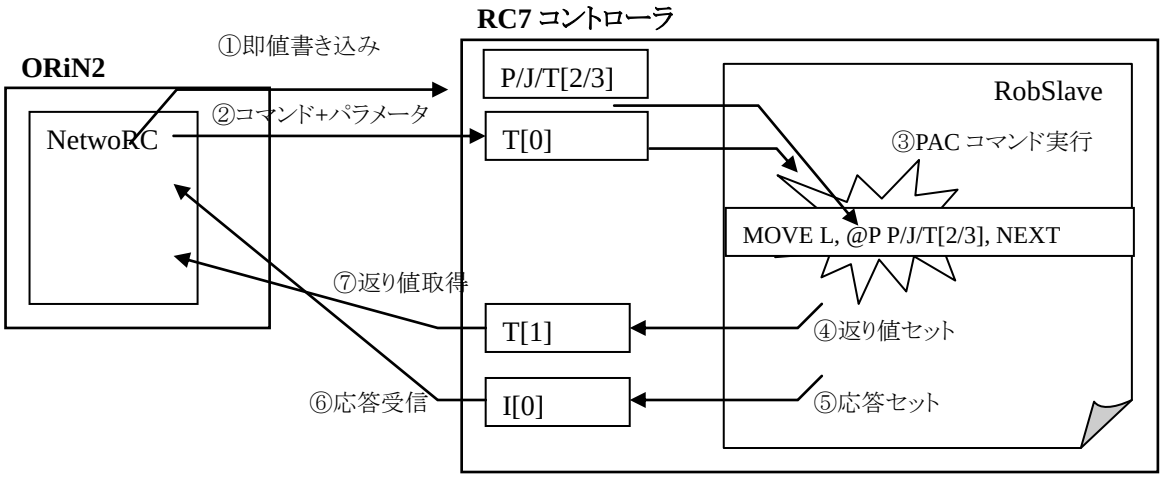


図 5-2 ロボット動作命令の定数指定実行手順

【注意】

全グローバル変数(I,F,D,V,P,J,T,S)のインデックス[0]から[9]まで変数はシステム予約しています. これらの変数へのアクセスはユーザープログラムでは行わないでください.

6. Tips

6.1. コントローラエラー発生時の書き込み

ロボットコントローラ内でエラーが発生している場合，変数や I/O への書き込みは安全上の理由により行えない仕様となっています．しかし，コントローラが持っている特権タスクの機能を使用することでこの問題に対処することができます．

特権タスク（TSR1,TSR2,...）はシステムにより自動起動されたり，エラー発生時でも書き込みを行うことができる特別な権限を持っています．次に示す手順で実現できます．

(1) 特権タスク機能を有効にする．
(2) 外部からの通知で書き込みを行う特権タスクプログラムを作成し，起動する．
(3) PC から特権タスクに対して通知する．

特権タスクと NetwoRC プロバイダを同時に使う場合のシステム構成は下図の通りです．

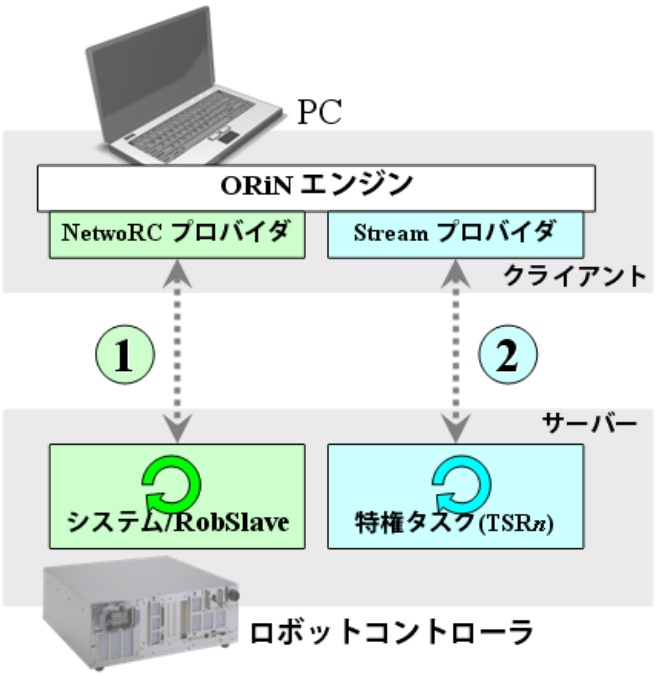


図 6-1 特権タスク(PAC)によるシステム拡張

Stream プロバイダ[11]経由で PC からコントローラ内の特権タスクと通信を行うことでエラー発生時の書き込みを可能にします．

通常は NetwoRC プロバイダを用いた方が高速で簡単ですが，エラー発生時に特別な処理を実行したい場合などにこの方法が有効です(表 6-1).

[11] Stream プロバイダに関しては ORiN2 SDK にある「Stream プロバイダユーザーズ ガイド」を参照ください．

表 6-1 通信方法比較

通信方法	処理速度	特権タスク	エラー発生時の書込み
1. NetwoRC プロバイダ (Robotalk/UDP/IP)	○ 高速	○ 不要 (システムプログラム＋RobSlave)	× 不可
2. Stream プロバイダ (TCP/IP)	× 低速	× 必要 (PAC プログラム)	○ 可能

特権タスク機能の詳細に関しては下記マニュアルを参照してください.

「デンソーロボット取扱説明書 - 操作ガイド」

第 3 章 動作モードと付加機能 - 特権タスク(簡易 PLC 機能)

6.1.1. 特権タスク機能を有効にする

コントローラで特権タスクを有効にする設定は, 以下の手順で行います.

(1) [1111]及び[1112]のオプションを追加する.

トップ画面 → 設定[F6] → オプション[F7] → 機能拡張[F8]

(2) コントローラを再起動する.

(3) 特権タスク機能を有効にする.

トップ画面 → SHIFT → 特権タスク[F2] → 起動設定 [F2]

(4) コントローラを再起動する.

6.1.2. 特権タスクプログラムを作成する

外部 PC から通知を受けて IO への書き込みを行う特権タスクプログラム(PAC)の例を以下に示します. コントローラはサーバーモードで動作し, 待ち受けは#4 番, TCP/IP ソケット通信でデフォルトの 5001 番ポートを使用する設定とします.

List 6-1 TSR1.pac

```
PROGRAM TSR1
        Defint lval
        Defstr sval

        Do '通信可能な状態まで待つ
                com_state #4, lval
                If lval <> -1 Then Exit Do
                Delay 100
        Loop

        Do 'コマンド待ち
                Input #4, sval      '#4 = TCP 5001 port
                lval = Val(sval)    '文字列から数値へ変換
                If lval>0 Then
                        Set IO[lval]        '"128" -> IO[128] = ON
                        Print #4, sval; " ON"
                Else
                        Reset IO[-lval]     '"-128" -> IO[128] = OFF
                        Print #4, sval; " OFF"
                End If
        Loop
END
```

[コントローラ設定]

トップ画面 → 設定[F6] → 通信設定[F5] → サーバー[F11]

6.1.3. PC から特権タスクに対して通知する

PC から特権タスクに対して書き込み通知を行うクライアントプログラム(Visual Basic 6.0)の例を以下に示します. コントローラの IP アドレス(192.168.0.1)と待ち受けポート番号(5001)は, 上で設定した値に合わせてください.

List 6-2 Client.frm

```
Private eng As CaoEngine
Private WithEvents ctrl As CaoController

Private Sub Form_Load()

    Set eng = New CaoEngine

    ' クライアントモードで通信開始
    Set ctrl = eng.Workspaces(0).AddController("Client", _
                                               "CaoProv.DNWA.STREAM", _
                                               "", _
                                               "Conn=eth:192.168.0.1:5001")
End Sub

Private Sub Command1_Click()

    ' 送信処理
    ctrl.Execute "Send", text1.Text   '入力例 : "128" → IO128=ON

End Sub

' 受信イベント
Private Sub ctrl_OnMessage(ppCaoMess As CAOLib.ICaoMessage)

    ' データ受信
    text2.Text = ppCaoMess.Value

End Sub
```

付録A. POSEDATA 型定義

NetwoRC プロバイダではデンソーロボットのポーズデータ型およびベクトル型を VARIANT 型変数で扱えるよう“POSEDATA 型”と称して次に示す型定義を行っています.

POSEDATA 型(VARIANT)

- VT_BSTR[12] "[<パス>] [<変数型>]<インデックス> [<付加軸>]"
  または
  "[<パス>] [<変数型>](<要素 1>,<要素 2>,...) [<付加軸>]"
- VT_R4|VT_ARRAY[13] <即値> = (<要素 1:VT_R4>,<要素 2:VT_R4>,...)[14]
- VT_VARIANT|VT_ARRAY (<値>[,<変数型>[,<パス>[, <付加軸>]]])
  - <値> <インデックス:VT_R4>
    または
    <即値:VT_R4|VT_ARRAY>
  - <変数型> P, T, J, V 型の VT_I4 または VT_BSTR 指定 (省略時＝P 扱い)
  - <パス> @P, @E, @0, @<数値>の VT_I4 または VT_BSTR 指定
    (省略時＝@0)
  - <付加軸> <付加軸オプション:VT_VARIANT|VT_ARRAY>
    (省略時＝付加軸指定なし)

<パス> : @P, @E, @0, @<数値>

表記	@P	@E	@0	@<数値:n>	なし
VT_BSTR	"@P"	"@E"	"@0"	"@n"	""
VT_I4	-1	-2	0	n	0

<変数型> : P 型, T 型, J 型, V 型

表記	P	T	J	V	なし
VT_BSTR	"P"	"T"	"J"	"V"	""
VT_I4	0	1	2	3	-1

<インデックス> : <数値:VT_R4>

<要素 n> : <数値:VT_R4>

<付加軸オプション> : (<EX または EXA>, (<軸 1:VT_I4>,<値 1:VT_R8>)[,(<軸 2>,<値 2>)...])

[12] VT_BSTR のみ複数 POSEDATA 型の”, ”カンマ区切りでの同時指定も可能です.
[13] <変数型>,<パス>は指定できないため, それぞれデフォルトで P 型, @0 と同じ扱いになります.
[14] <変数型>,<パス>は指定できないため, それぞれデフォルトで P 型, @0 と同じ扱いになります.

表記	EX	EXA	なし
VT_BSTR	"EX"	"EXA"	""
VT_I4	1	2	0

POSEDATA 型を使用して次の PAC 言語の書式を表現することが出来ます.

[<パス開始変位>] <ポーズ:P,T,J 型> [<付加軸>] (C0 書式)

[<パス開始変位>] <移動量:V 型> (C1 書式)

[<パス開始変位>] <値> [<付加軸>] (C2 書式)

[<パス開始変位>] (<要素 1>,<要素 2>,...) [<付加軸>] (C3 書式)

付録A.1. 表記例

[<パス開始変位>] <ポーズ> [<付加軸>] (C0)

ex1. T200

文字列指定	"T200"
VARIANT 型配列指定 (変数型は文字列指定)	Array(200,"T")[15]
VARIANT 型配列指定 (変数型は数値指定)	Array(200,1)

ex2. @P J100

文字列指定	"@P J100"
VARIANT 型配列指定 (変数型, パスは文字列指定)	Array(100,"J","@P")
VARIANT 型配列指定 (変数型, パスは数値指定)	Array(100,2,-1)

ex3. @E P(10.0, 10.5, 34.6, 0.0, 90.0, 0.0, -1.0)

文字列指定	"@E P(10.0, 10.5, 34.6, 0.0, 90.0, 0.0, -1.0)"
VARIANT 型配列指定 (即値指定. 変数型, パスは文字列指定)	Dim p(6) as Single Dim vP as Variant p(0) = 10.0 : p(1) = 10.5 : p(2) = 34.6 : p(3) = 0.0 p(4) = 90.0 : p(5) = 0.0 : p(6) = -1.0 vP = p() Array(vP, "P", "@E")
VARIANT 型配列指定 (即値指定. 変数型, パスは数値指定)	Dim p(6) as Single Dim vP as Variant p(0) = 10.0 : p(1) = 10.5 : p(2) = 34.6 : p(3) = 0.0 p(4) = 90.0 : p(5) = 0.0 : p(6) = -1.0 vP = p() Array(vP, 0, -2)

[15] Array(…)は渡された要素を配列に代入してその配列を返す関数を表しています.(VB6 の Array 関数)

ex4. @P J100 EXA((7, 30.5), (8, 90.5))

文字列指定	"@P J100 EXA((7, 30.5), (8, 90.5))"
VARIANT 型配列指定 (変数型, パス, 付加軸は文字列指定)	Array(100,"J","@P", Array("EXA",Array(7,30.5), Array(8,90.5)))
VARIANT 型配列指定 (変数型, パス, 付加軸は数値指定)	Array(100,2,-1, Array(2, Array(7,30.5), Array(8,90.5)))

[<パス開始変位>] <移動量> (C1)

ex1. @P V20

文字列指定	"@P V20"
VARIANT 型配列指定 (変数型, パスは文字列指定)	Array(20,"V","@P")
VARIANT 型配列指定 (変数型, パスは数値指定)	Array(20,3,-1)

ex2. @E V(0.0, 125.5, 50.0)

文字列指定	"@E V(0.0, 125.5, 50.0)"
VARIANT 型配列指定 (即値指定. 変数型, パスは文字列指定)	Dim v(2) as Single Dim vV as Variant v(0) = 0.0 : v(1) = 125.5 : v(2) = 50.0 vV = v() Array(vV, "V", "@E")
VARIANT 型配列指定 (即値指定. 変数型, パスは数値指定)	Dim v(2) as Single Dim vV as Variant v(0) = 0.0 : v(1) = 125.5 : v(2) = 50.0 vV = v() ' = VT_R4 | VT_ARRAY Array(vV, 3, -2)

[<パス開始変位>] <値> [<付加軸>] (C2)

ex1. @P 1

文字列指定	"@P 1"
VARIANT 型配列指定 (変数型, パスは文字列指定)	Array(1, "", "@P")
VARIANT 型配列指定 (変数型, パスは数値指定)	Array(1,-1,-1)

ex2. @P 1.56

文字列指定	"@P 1.56"
VARIANT 型配列指定 (変数型, パスは文字列指定)	Array(1.56,"","@P")
VARIANT 型配列指定 (変数型, パスは数値指定)	Array(1.56,-1,-1)

[<パス開始変位>] (<要素 1>,<要素 2>,..) [<付加軸>] (C3)

ex1. @P (1, 30.0)

文字列指定	"@P (1, 30.0)"
VARIANT 型配列指定 (変数型, パスは文字列指定)	Dim v(1) as Single v(0) = 1 : v(1) = 30.0 Dim vV as Variant vV = v() Array(vV, "", "@P")
VARIANT 型配列指定 (変数型, パスは数値指定)	Dim v(1) as Single v(0) = 1 : v(1) = 30.0 Dim vV as Variant vV = v() Array(vV, -1, -1)

その他の表記方法

ex1. V1,V2,V3

(CaoRobot::Rotate()の回転面)

文字列指定	"V1,V2,V3"
文字列型配列指定	Array("V1","V2","V3")
VARIANT 型配列指定 (変数型は文字列指定)	Array(Array(1,"V"),Array(2,"V"),Array(3,"V"))
VARIANT 型配列指定 (変数型は数値指定)	Array(Array(1,3),Array(2,3),Array(3,3))

ex2. APPROACH P,P70, 60, NEXT

(CaoRobot::Execute()の Approach コマンド アプローチ長パス指定なし)

第2引数:文字列指定 第3引数:文字列指定	.Execute "APPROACH", Array(1, "P70", "60", "NEXT")
第2引数:VARIANT 配列指定 第3引数:VARIANT 配列指定	.Execute "APPROACH", Array(1, Array(70, "P"),_ Array(60, "", ""), "NEXT")

ex3. APPROACH L,J(60.5,30.3,400,90),@100 70, NEXT

(CaoRobot::Execute()の Approach コマンド アプローチ長パス指定なし)

第2引数:文字列指定	.Execute "APPROACH", Array(2, "J(60.5,30.3,400,90)", "@100 70", "NEXT")

第3引数:文字列指定	
第2引数:VARIANT 配列指定 (即値指定. 変数型は文字列指定) 第3引数:VARIANT 配列指定 (変数型, パスは文字列指定)	Dim j(3) as Single Dim vJ as Variant j(0) = 60.5 : j(1) = 30.3 : j(2) = 400 : j(3) = 90 vJ = j() ‘ = VT_R4 | VT_ARRAY .Execute "APPROACH", Array(2, Array(vJ, "J"), _ Array(70,"","@100"), "NEXT")
第2引数:VARIANT 配列指定 (即値指定. 変数型は文字列指定) 第3引数:VARIANT 配列指定 (変数型, パスは数値指定)	Dim j(3) as Single Dim vJ as Variant j(0) = 60.5 : j(1) = 30.3 : j(2) = 400 : j(3) = 90 vJ = j() ‘ = VT_R4 | VT_ARRAY .Execute "APPROACH", Array(2, Array(vJ, "J"), _ Array(70, -1, 100), "NEXT")

[注意事項]

即値を POSEDATA 型 VT_R4|VT_ARRAY 形式で直接指定した場合はデフォルトで P 型, @0 扱いになるため P 型以外のデータを VT_R4|VT_ARRAY 形式で直接扱うことは出来ません. このような場合は VT_VARIANT|VT_ARRAY 形式または VT_BSTR 形式を使用して明示的に扱うデータの変数型を指定するようにしてください.

次のようなコードは期待する動作にはならないので注意してください.

```
‘[PAC] MOVE P, J100
        Dim vJ as Variant
        vJ=CaoCtrl.Variables(“J100”).Value   ‘VT_R4|VT_ARRAY
        Robot.Move 1, vJ                      ‘間違い!! = MOVE P, P(<j1>,<j2>,<j3>,…)
```

正しくは次のようなコードになります.

```
        Robot.Move 1, Array(vJ,”J”)           ‘変数型指定= MOVE P, J(<j1>,<j2>,<j3>,…)
```

付録B. PAC 言語対応状況

2.330

表 6-2 PAC 言語 動作コマンド対応状況

分類	PAC コマンド名	対応
動作制御		
	APPROACH	○
	DEPART	○
	DRAW	○
	DRIVE	○
	DRIVEA	○
	HOME	×
	GOHOME	×
	MOVE	○
	ROTATE	○
	ROTATEH	○
	CURJNT	○
	CURPOS	○
	CURTRN	○
	CUREXJ	×
	DESTJNT	×
	DESTPOS	×
	DESTTRN	×
	DESTEXJ	×
	ARRIVE	○
	POSCLR	○
	SETSPLINEPOINT	○
	CLRSPLINEPOINT	○
	GETSPLINEPOINT	○
	FIGAPRL	○
	FIGAPRP	○
停止制御		
	HOLD	×
	HALT	×
	INTERRUPT ON/OFF	○
速度制御		
	SPEED	○
	JSPEED	○
	ACCEL	○
	JACCEL	○
	DECEL	○
	JDECEL	○
	CURACC	○
	CURJACC	○
	CURDEC	○
	CURJDEC	○
	CURJSPD	○
	CURSPD	○
	CUREXTACC	○
	CUREXTDEC	○
	CUREXTSPD	○
	EXTSPEED	○
	CHANGETOOL	○
	CHANGEWORK	○
	CURTOOL	○
	CURWORK	○
干渉チェック		
	SETAREA	○
	RESETAREA	○
サーボ内部データ		
	GetSrvData	○
	GetJntData	○
	GetSrvState	○
サーボ ON		
	MOTOR {ON|OFF}	○
特殊制御		

	ST_aspACLD	○
	ST_aspChange	○
	ST_SetGravity	○
	ST_ResetGravity	○
	ST_SetGrvOffset	○
	ST_ResetGrvOffset	○
	ST_SetCurLmt	○
	ST_ResetCurLmt	○
	ST_SetEralw	○
	ST_ResetEralw	○
	ST_OnSrvLock	○
	ST_OffSrvLoc	○
	ST_SetCompControl	○
	ST_SetCompFControl	○
	ST_ResetCompControl	○
	ST_SetFrcCoord	○
	ST_SetFrcLimit	○
	ST_ResetFrcLimit	○
	ST_SetCompRate	○
	ST_ResetCompRate	○
	ST_SetFrcAssist	○
	ST_ResetFrcAssist	○
	ST_SetCompJLimit	○
	ST_ResetCompJLimit	○
	ST_SetCompVMode	○
	ST_ResetCompVMode	○
	ST_SetCompEralw	○
	ST_ResetCompEralw	○
	ST_SetDampRate	○
	ST_ResetDampRate	○
	ST_SetZBalance	○
	ST_ResetZBalance	○
ポーズデータ型変換		
	J2P	○
	J2T	○

	P2J	○
	P2T	○
	T2J	○
	T2P	○
	TINV	○
	NORMTRN	○
コンベアトラッキング		
	TRACKDATAINITIALIZE	○
	TRACKDATASET	○
	TRACKDATAGET	○
	TRACKDATAINFO	○
	TRACKDATANUM	○
	CURTRACKPOS	○
	CURTRACKSPD	○
	WAITTRACKMOVE	○
	CALCWORKPOS	○
	CURTRACKPOSEX	○
	WAITTRACKMOVEEX	○
	SETTRACKMOVE	○
	RESETTRACKMOVE	○
	CONVCAL	×
	CALCCAMCALPOS	×
	CALCIOTEACHPOS	×
	SetTrackStartArea	○
	CalcConvPos	×
	SetConvLowVelErr	×
	CalcConvVec	×
	SortTrackData	×
	SortTrackAllData	×
その他		
	MotionSkip	○
	MotionComp	○
	XdWaitSpline	○
	DELAY	○
	SYSSTATE	○

	ClearSrvMonitor	○
	StartSrvMonitor	○
	StopSrvMonitor	○
	SetMonitorCond	○

付録C. トラブル時の確認事項

付録C.2. ロボットコントローラと通信できない...

確認事項	対処
■ コントローラ側	
□ コントローラの ROM バージョンは ORiN 対応バージョン（V2.330 以上）になっていますか？	バージョンを確認してください.（コントローラマニュアル参照）
□ RS232C や Ethernet ケーブルは正しく接続されていますか？	ケーブルを確認してください.（コントローラマニュアル参照）
□ ケーブルの種類（ストレート/クロス）は正しいですか？	ケーブルを確認してください.（コントローラマニュアル参照）
□ Ethernet接続の場合，アドレス設定は正しくされていますか？	通信設定を確認してください.（コントローラマニュアル参照）
□ セグメントをまたいだ Ethernet 接続の場合，デフォルトゲートウェイの設定は正しくされていますか？	通信設定を見直してください.（コントローラマニュアル参照）
□ RS232C 接続の場合，通信パラメータの設定は正しくされていますか？	通信設定を見直してください.（コントローラマニュアル参照）
□ 通信権は正しく設定されていますか？	通信設定を見直してください.（2.2.1 参照）
□ ORiN オプション機能は追加されていますか？	ORiN オプションを追加してください.（2.2.1 参照）
■ パソコン側	
□ Windows のバージョンは ORiN 対応バージョン（Windows 2000 SP4 以上，Windows XP SP1 以降）になっていますか？	バージョンを確認してください.
□ RS232C 接続の場合，他のアプリケーション（例：ハイパーターミナル）がポートを占有していませんか？	占有しているアプリケーションを終了してください.
□ RobMaster.exe は正常に起動しますか？	動作確認してください.
□ 作成プログラムの AddController の引数は正しく指定されていますか？	AddController の引数を確認してください.（4.2.1 参照）

付録C.3. ロボットコントローラの変数を読み書きできない...

確認事項	対処
■ コントローラ側	
□ 通信権が使用不可または読込みのみの設定になっていませんか？	通信設定を見直してください.（2.2.1 参照）
□ ティーチングペンダントで編集中画面が表示中ではあり	編集中画面を閉じてください.

ませんか？	
□ ティーチングペンダントでエラーメッセージが表示中ではありませんか？	エラーをクリアしてください.
■ パソコン側	
□ 変数名は正しいですか？	変数の指定を見直してください.
□ 書き込みに失敗する場合, その変数は読み込み専用の変数ではありませんか？	変数の指定を見直してください.
□ アクセス可能な範囲の変数指定を正しくしていますか？	I/O 変数の場合は割付表で確認してください.
□ 書き込む値及び型は仕様を満たしていますか？	変数仕様を確認してください.

付録C.4. ロボットの制御ができない...

確認事項	対処
■ コントローラ側	
□ 単一位置制御の起動権を正しく設定していますか？	通信設定を見直してください. (2.4.1 参照)
□ RobSlave タスクは動作していますか？	RobMaster 等で RobSlave タスクを起動してください. 起動できない場合はステップ停止信号 OFF 等を確認してください.
□ ロボットは起動できる状態ですか？	外部自動モード, モータ ON, 非常停止 OFF 等を確認してください.
□ RobSlave.pac と NetwoRC プロバイダの整合性は取れていますか？	RobSlave.pac と NetwoRC プロバイダは同一 ORiN2 SDK のものをお使いください.
■ パソコン側	
□ コマンドの引数は正しく指定していますか？	コマンド仕様を確認してください. 引数の変数型に注意してください.

付録D. コントローラ別対応状況

表 6-3 コントローラ別対応状況

クラス	プロパティ メソッド イベント		R/W	RC5		RC7		備考
				1.998 未満	1.998 以上	2.330 未満	2.330 以上	
CaoController	Attribute	P	R	-	○	-	○	
	CommandNames	P	R	-	○	-	○	
	Commands	P	R	-	○	-	○	
	ExtensionNames	P	R	-	○	-	○	
	Extensions	P	R	-	○	-	○	
	FileNames	P	R	-	○	-	○	
	Files	P	R	-	○	-	○	
	Help	P	R	-	○	-	○	
	ID	P	R/W	-	○	-	○	
	Index	P	R	-	○	-	○	
	Name	P	R	-	○	-	○	
	RobotNames	P	R	-	○	-	○	
	Robots	P	R	-	○	-	○	
	Tag	P	R/W	-	○	-	○	
	TaskNames	P	R	-	○	-	○	
	Tasks	P	R	-	○	-	○	
	VariableNames	P	R	-	○	-	○	
	Variables	P	R	-	○	-	○	
	AddCommand	M	S	-	○	-	○	
	AddExtension	M	S	-	○	-	○	
	AddFile	M	S	-	○	-	○	
	AddRobot	M	S	-	○	-	○	
	AddTask	M	S	-	○	-	○	
	AddVariable	M	S	-	○	-	○	
	Execute	M	S	-	○	-	○	
	GetMessage	M	R	-	○	-	○	
	OnMessage	E	R	-	○	-	○	

CaoVariable	Attribute	P	R	-	○	-	○	
	DateTime	P	R	-	○	-	○	
	Help	P	R	-	○	-	○	
	ID	P	R/W	-	○	-	○	
	Index	P	R	-	○	-	○	
	Microsecond	P	R	-	○	-	○	
	Name	P	R	-	○	-	○	
	Tag	P	R/W	-	○	-	○	
	Value	P	R/W	-	○	-	○	
CaoFile	Attribute	P	R	-	○	-	○	
	DateCreated	P	R	-	○	-	○	
	DateLastAccessed	P	R	-	○	-	○	
	DateLastModified	P	R	-	○	-	○	
	FileNames	P	R	-	○	-	○	
	Files	P	R	-	○	-	○	
	Help	P	R	-	○	-	○	
	ID	P	R/W	-	○	-	○	
	Index	P	R	-	○	-	○	
	Name	P	R	-	○	-	○	
	Path	P	R	-	○	-	○	
	Size	P	R	-	○	-	○	
	Tag	P	R/W	-	○	-	○	
	Type	P	R	-	○	-	○	
	Value	P	R/W	-	○	-	○	
	VariableNames	P	R	-	○	-	○	
	Variables	P	R	-	○	-	○	
	AddFile	M	S	-	○	-	○	
	AddVariable	M	S	-	○	-	○	
	Copy	M	W	-	○	-	○	
	Delete	M	W	-	○	-	○	
	Execute	M	S	-	○	-	○	
	Move	M	W	-	○	-	○	
	Run	M	W	-	-	-	-	

CaoTask	Attribute	P	R	-	○	-	○	
	FileName	P	R	-	○	-	○	
	Help	P	R	-	○	-	○	
	ID	P	R/W	-	○	-	○	
	Index	P	R	-	○	-	○	
	Name	P	R	-	○	-	○	
	Tag	P	R/W	-	○	-	○	
	VariableNames	P	R	-	○	-	○	
	Variables	P	R	-	○	-	○	
	AddVariable	M	S	-	○	-	○	
	Delete	M	W	-	○	-	○	
	Execute	M	S	-	○	-	○	
	Start	M	W	-	-	-	○	
	Stop	M	W	-	○	-	○	
						-		
CaoRobot	Attribute	P	R	-	○	-	○	
	Help	P	R	-	○	-	○	
	ID	P	R/W	-	○	-	○	
	Index	P	R	-	○	-	○	
	Name	P	R	-	○	-	○	
	Tag	P	R/W	-	○	-	○	
	VariableNames	P	R	-	○	-	○	
	Variables	P	R	-	○	-	○	
	Accelerate	M	W	-	-	-	○	
	AddVariable	M	S	-	○	-	○	
	Change	M	W	-	-	-	○	
	Chuck	M	W	-	-	-	-	
	Drive	M	W	-	-	-	-	Execute 参照
	Execute	M	S	-	-	-	○	
	GoHome	M	W	-	-	-	-	
	Hold	M	W	-	-	-	○	
	Halt	M	W	-	-	-	○	
	Move	M	W	-	-	-	○	
	Rotate	M	W	-	-	-	○	
	Speed	M	W	-	-	-	○	

	Unchuck	M	W	-	-	-	-	
	Unhold	M	W	-	-	-	-	
CaoCommand	Attribute	P	R	-	○	-	○	
	Help	P	R	-	○	-	○	
	ID	P	R/W	-	○	-	○	
	Index	P	R	-	○	-	○	
	Name	P	R	-	○	-	○	
	Parameters	P	R/W	-	○	-	○	
	Result	P	R	-	○	-	○	
	State	P	R	-	○	-	○	
	Tag	P	R/W	-	○	-	○	
	Timeout	P	R/W	-	○	-	○	
	Cancel	M	S	-	○	-	○	
	Execute	M	S	-	○	-	○	
CaoExtension	Attribute	P	R	-	○	-	○	
	Help	P	R	-	○	-	○	
	ID	P	R/W	-	○	-	○	
	Index	P	R	-	○	-	○	
	Name	P	R	-	○	-	○	
	Tag	P	R/W	-	○	-	○	
	VariableNames	P	R	-	○	-	○	
	Variables	P	R	-	○	-	○	
	AddVariable	M	S	-	○	-	○	
	Execute	M	S	-	○	-	○	
CaoMessage	DateTime	P	R	-	○	-	○	
	Description	P	R	-	○	-	○	
	Destination	P	R	-	○	-	○	
	Number	P	R	-	○	-	○	
	SerialNumber	P	R	-	○	-	○	
	Source	P	R	-	○	-	○	
	Value	P	R	-	○	-	○	
	Clear	M	W	-	○	-	○	

	Reply M	W	-	○	-	○	
					-		
記号の意味	M:メソッド P:プロパティ E:イベント	R:読込 W:書込 S:仕様依存					

付録E. エラーコード一覧

プロバイダのエラーコードは HRESULT 型の構造になっています. HRESULT 型の詳細については, 以下の URL を参照してください.

<http://msdn2.microsoft.com/en-us/library/bb401631.aspx>

プロバイダ内で使用しているエラーコードには, Microsoft Windows で標準的に定義されている「標準エラー」とプロバイダ独自に定義した「カスタムエラー」の 2 種類のがあります. 標準エラーは, Winerror.h で定義されている共通のエラーコードです. カスタムエラーはプロバイダで定義されているローカルなエラーコードです. 以下にエラーコードの一覧を示します.

表 6-4 NetwoRCプロバイダのエラーコード

番号	マクロ名	内容
標準エラー(抜粋)		
0x00000000	S_OK	No error occurred
0x00000001	S_FALSE	No error occurred, but the command was not finished properly.
0x8000FFFF	E_UNEXPECTED	Catastrophic failure
0x80004001	E_NOTIMPL	Not implemented
0x8007000E	E_OUTOFMEMORY	Ran out of memory
0x80070057	E_INVALIDARG	One or more arguments are invalid
0x80004005	E_FAIL	Unspecified error
カスタムエラー		
0x80000801	E_CAOP_NO_ROBSLAVE	RobSlave program does not exist in the robot controller.
0x80000802	E_CAOP_ROBSLAVE_NOT_READY	RobSlave program in the robot controller is not running.
0x80000803	E_CAOP_ROBSLAVE_CRC_ERROR	RobSlave program CRC error.
0x80000804	E_CAOP_ILLEGAL_CTRLVER	Illegal robot controller version.
0x80000805	E_CAOP_NO_EXECTOKEN	No executable token.
0x80000806	E_CAOP_ILLEGAL_ROBSLAVE	Illegal RobSlave version.
0x80000807	E_CAOP_NO_LICENSE	The count of connections is over the possible number. Please purchase an additional license.
0x80000900	E_TIMEOUT	Timeout occurred
0x80010900	E_SEND_NAK	NAK occurred

0x80010902	E_REJECTED	Reject occurred
0x80010903	E_ABNORMAL_R_PACKET	Receive packet broken
0x80010904	E_ABNORMAL_S_PACKET	Send packet broken

付録F. 無停止教示点補正機能(外観検査軌道生成)

付録F.1. パラメータ

GenerateNonStopPath コマンドの<座標情報>パラメータと, <エリア情報>パラメータについて, 詳細を以下に示します.

<座標情報:VT_VARIANT | VT_ARRAY> =
<X:VT_R4>,
<Y:VT_R4>,
< Z:VT_R4>,
< RX:VT_R4>,
< RY:VT_R4>,
< RZ:VT_R4>,
< Fig:VT_I4>,
< J7:VT_R4>,
< J8:VT_R4>,
<動作速度:VT_R4>=(0.0～1.0),
<動作パターン:VT_I4> = (0:@P, 1:@0, 2:@E),
< Tool 番号 :VT_I4>

<エリア情報:VT_R4 | VT_ARRAY > =
<エリアサイズ X:VT_R4>,
<エリアサイズ Y:VT_R4>,
< エリアサイズ Z:VT_R4>,
< エリア Angle:VT_R4>,
< エリアサイズ J7:VT_R4>,
< エリアサイズ J8:VT_R4>

付録F.2. エラーコード

プロバイダ内で定義している GenerateNonStopPath コマンド独自のカスタムエラーコードの一覧を以下に示します.

番号	マクロ名	内容
0x800120**	ERR_ORG_P2J_CONV	P2J 変換エラー(元データ)
0x800121**	ERR_ORG_SOFT_LIMIT	ソフトリミット範囲エラー(元データ)
0x800122**	ERR_ECH_SPD_RATE	個別速度比の範囲エラー

0x800123**	ERR_GEN_P2J_CONV	P2J 変換エラー(補正中データ)
0x800124**	ERR_GEN_SOFT_LIMIT	ソフトリミット範囲エラー (補正中データ)
0x800125**	ERR_GEN_IMPOSSIBLE	補正演算収束不能
0x800126**	ERR_TOO_NEAR_POINT	隣接する教示転換の距離(+姿勢)が近すぎる
0x800127**	ERR_SPEED_ZERO	速度設定が 0
0x800128**	ERR_INVALID_AREA_INFO	エリア情報不正(元データ)
0x80012A0*	ERR_INVALID_JOINTFLG	ジョイントフラグ値が無限回転, 有限回転のみに対応
0x80012F00	ERR_SPEED_RATE	総速度比が範囲エラー
0x80012F01	ERR_GEN_COEF	補正係数の範囲エラー
0x80012F02	ERR_PASS_GEN	PassGenerate 失敗
0x80012F03	ERR_MEMORY_TOO_SMALL	メモリ不足
0x80012F04	ERR_ARMDNTLL_LOAD_FAIL	ArmNT.dll ロード失敗
0x80012F05	ERR_DIVISION_BY_ZERO	姿勢補正, ゼロ演算発生
0x80012F06	ERR_UNESTB_ARMCNF	ArmCnf 未設定
0x80012F07	ERR_UNESTB_SPDCNF	SpdCnf 未設定
0x80012F08	ERR_UNESTB_SRVCNF	SrvCnf 未設定
0x80012F09	ERR_MEMORY_LEAK	メモリリーク発生
0x80012F0A	ERR_OUT_OF_DATA_NUM	データサイズオーバー
0x80012F10	ERR_IVALID_TOOL_DATA	Tool 番号のデータが範囲外
0x80012F50	ERR_OUT_OF_TOOL_NUM	Tool 番号の範囲外

0x80012000～0x80012800 のエラーコード

** に, エラーが発生した教示点番号＋1の値が入ります.

0x80012A0*のエラーコード

* に, 無限回転が設定された軸番号が入ります.

付録F.3. 制限事項

GenerateNonStopPath コマンドの実行には以下の制限があります.

- 教示点数の上限 = 200 点
- 6 軸ロボットのみ有効
- エリアサイズの付加軸の値は, 設置する付加軸によって, 回転(degree), 直動(mm)の値をそれぞ

れ指定します

- 付加軸の無限回転は対応不可
- 最適可搬質量設定には未対応
- 先端負荷質量の設定は，1000g 単位

付録F.4. サンプルプログラム

以下にCaoScriptで作成したサンプルプログラムを示します．教示座標は，VS-6577G-BAのロボットタイプを用いています．IPは各コントローラで設定した値を用いてください．サンプルプログラムでは以下の設定値を用いて接続しています．

IP: 10.6.235.72

Sample NonStopPath.vbs

```
' Generate NonStopPath
Const RC_ADDRESS = "10.6.235.72"

Sub Main
  Dim rc
  Dim vntTeachPos()
  Dim vntAreaInfo()
  Dim vntMovePos
  Dim vntParam
  Dim lIndex

  dbg.ClearLog

  set rc = cao.AddController("RC", "CaoProv.DENSO.NetwoRC", "", "conn=eth:" & RC_ADDRESS)
  ' ----------------------------------------
  ' Initialize NonStopPath Library
  ' ----------------------------------------
  call rc.Execute("InitNonStopPathLib")

  ' GenerateNonStopPath Command Parameter (Pos, Area, Size, SpdRate, Coef)
  ' Pos : TeachPoint Data (x, y, z, rx, ry, rz, fig, J7, J8, SpdRate, attr, ToolNum)
  redim vntTeachPos(7)

  vntTeachPos(0) = Array(300.0, 100.0, 600.0, 180.0, 0.0, 180.0, 5, 0.0, 0.0, 100 * 0.01,
1, 0)
  vntTeachPos(1) = Array(300.0, 91.0, 600.0, 180.0, 0.0, -180.0, 5, 0.0, 0.0, 100 * 0.01,
0, 0)
  vntTeachPos(2) = Array(310.0, 30.0, 600.0, 180.0, 0.0, -180.0, 5, 0.0, 0.0, 100 * 0.01,
1, 0)
  vntTeachPos(3) = Array(315.5, 24.5, 600.0, 180.0, 0.0, -180.0, 5, 0.0, 0.0, 100 * 0.01,
0, 0)
  vntTeachPos(4) = Array(300.0, 10.0, 600.0, 180.0, 0.0, 173.0, 5, 0.0, 0.0, 100 * 0.01,
1, 0)
  vntTeachPos(5) = Array(300.0, 10.0, 600.0, 180.0, 0.0, 176.0, 5, 0.0, 0.0, 100 * 0.01,
0, 0)
  vntTeachPos(6) = Array(300.0, 10.0, 600.0, 180.0, 0.0, 171.0, 5, 0.0, 0.0, 100 * 0.01,
0, 0)
  vntTeachPos(7) = Array(300.0, 10.0, 600.0, 180.0, 0.0, -180.0, 5, 0.0, 0.0, 100 * 0.01,
1, 0)

  ' Area : Area Info (x, y, z, angle, J7, J8)
  redim vntAreaInfo(7)

  vntAreaInfo(0) = Array(4,4,4,4,0,0)
```

```
    vntAreaInfo(1) = Array(4,4,4,4,0,0)
    vntAreaInfo(2) = Array(4,4,4,4,0,0)
    vntAreaInfo(3) = Array(4,4,4,4,0,0)
    vntAreaInfo(4) = Array(4,4,4,4,0,0)
    vntAreaInfo(5) = Array(4,4,4,4,0,0)
    vntAreaInfo(6) = Array(4,4,4,4,0,0)
    vntAreaInfo(7) = Array(4,4,4,4,0,0)

    dbg.Output "Teach Points"
    for lIndex = 0 to Ubound(vntTeachPos)
            dbg.Output lIndex & ":" & dat.BstrFromVariant(vntTeachPos(lIndex))
    next

    ' ----------------------------------------
    ' Generate NonStopPath
    ' ----------------------------------------
    vntMovePos = rc.Execute("GenerateNonStopPath", Array(vntTeachPos, vntAreaInfo,
Ubound(vntTeachPos) + 1, 100.0 * 0.01, 0.7, 1))

    dbg.Output "Move Points"
    for lIndex = 0 to Ubound(vntMovePos)
            dbg.Output lIndex & ":" & dat.BstrFromVariant(vntMovePos(lIndex))
    next
End Sub
```

付録F.5. 軌道補正失敗時(エラーコード:0x800123xx)の回避方法

GenerateNonStopPath コマンドに失敗する場合，補正中の教示点座標が，ロボットの各軸に対する最大補正角度の値を超過している可能性があります(エラーコード:0x800123xx(xx は教示点番号))．

上記のエラーが発生した場合は，エラーの発生した教示点の座標を変更するか，選択アルゴリズムによって以下のパラメータを調整してください．

- アルゴリズムが各軸補正(デフォルト値)の場合

NetwoRC プロバイダの Bin フォルダ内にある cOrbitGenSync.ini を開き，以下のパラメータを調整してください．

パラメータ	概要	入力範囲
FIGCHECK.MAX_DISPLACEMENT_dJ1	1 軸最大補正角度のオフセット[deg]	-180.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_dJ2	2 軸最大補正角度のオフセット[deg]	-180.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_dJ3	3 軸最大補正角度のオフセット[deg]	-180.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_dJ4	4 軸最大補正角度のオフセット[deg]	-180.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_dJ5	5 軸最大補正角度のオフセット[deg]	-180.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_dJ6	6 軸最大補正角度のオフセット[deg]	-180.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_J7	7 軸最大補正角度のオフセット[deg]	0.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_J8	8 軸最大補正角度のオフセット[deg]	0.0 ～ 180.0
FIGCHECK.ERROR_DISTANCE	許容移動距離[mm]	

各軸補正のアルゴリズムの場合，ini ファイル内に，直接各軸の最大補正角度を設定出来ません．

そのため，DEBUG.FILEOUT パラメータを 1 に変更すると，DEBUG.FILEOUT で指定したフォルダ内に，内部で自動計算した各軸の最大補正角度のパラメータを出力することが出来ます．
（MaxDispJoint.csv）
軌道生成で使用する各軸の最大補正角度は，出力され値に，各軸の最大補正角度のオフセット（MAX_DISPLACEMENT_dJ*）のパラメータをオフセットした値となります．

DEBUG.FILEOUT	1
DEBUG.FILEPATH	任意の出力フォルダ

（注意）DEBUG.FILEOUT パラメータが 1 の場合，InitNonStopPathLib，GenerateNonStopPath コマンドを実行する毎にファイルが生成されます．
教示点座標の調整の完了後，DEBUG.FILEOUT パラメータを 0 に変更してください．

・ アルゴリズムが先端座標補正の場合

NetwoRC プロバイダの Bin フォルダ内にある cOrbitGen.ini を開き，以下のパラメータを調整してください．

パラメータ	概要	入力範囲
FIGCHECK.MAX_DISPLACEMENT_dJ1	1 軸最大補正角度[deg]	0.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_dJ2	2 軸最大補正角度[deg]	0.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_dJ3	3 軸最大補正角度[deg]	0.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_dJ4	4 軸最大補正角度[deg]	0.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_dJ5	5 軸最大補正角度[deg]	0.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_dJ6	6 軸最大補正角度[deg]	0.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_J7	7 軸最大補正角度[deg]	0.0 ～ 180.0
FIGCHECK.MAX_DISPLACEMENT_J8	8 軸最大補正角度[deg]	0.0 ～ 180.0