FANUC Robot series

R-30*i*B Plus/R-30*i*B Mate Plus/R-30*i*B Compact Plus/

R-30*i*B Mini Plus 制御装置

プラグイン機能

取扱説明書

A-97606-6419JA/12

© FANUC CORPORATION, 2024

［ご案内］

労働安全衛生法では、産業用ロボットの教示などを行う使用者および　検査などを行う使用者に対して、該当業務に関する安全または衛生のための特別教育の実施を義務づけられています。

ファナックアカデミ　ロボット課では、ファナックロボットを
ご使用のお客様に対して各種コースを準備しておりますので受講されることを推奨いたします。

ファナックアカデミ　ロボット課の主な教育コース
- 教示：ロボット教示・操作一般コース
- 保守：ロボット保守コース
- 知能化：*i*RVision 2 次元補正基本コース、ビジュアルトラッキングコース
- オフラインプログラミング：ROBOGUIDE コース

ロボットの機種ごとに行われるコースもありますので詳細については
ロボット課にお問い合わせください。
所在：〒４０１－０５９７　山梨県忍野村
名称：ファナックアカデミ
電話：０５５５－８４－６０３０
FAX：０５５５－８４－５５４０

このたびは、ファナックロボットをご購入いただきまして、誠にありがとうございます。
ロボットをご使用になる前には、「FANUC Robot series 安全マニュアル(B-80687JA)」を必ずご一読いただき、内容を理解された上で、ロボットを使用してください。

・本マニュアルからの無断転載を禁じます。
・本商品の外観および仕様は改良のため予告なく変更することがあります。

本マニュアルに記載された商品は、『外国為替及び外国貿易法』に基づく規制対象です。輸出には日本政府の許可が必要な場合があります。また、商品によっては米国政府の再輸出規制を受ける場合があります。本商品の輸出に当たっては、弊社までお問い合わせください。

本マニュアルでは、できるだけいろいろな事柄を書くように努めています。
しかし、こういうことはやってはいけない、こういうことはできないということは非常に多く説明書が膨大になり、書ききれません。
したがって、本書で特にできると書いていないことは「できない」と解釈してください。

安全にご使用いただくために

本章ではロボットを安全に使用するための注意事項について説明しています。ロボットをご使用になる前に必ず本章をお読みください。

ロボットを操作する上での機能詳細につきましては、説明書によって十分仕様を理解してください。

ロボットと周辺機器、およびそれらの組合せであるロボットシステムを運用するに当たって、使用者およびシステムの安全対策を十分検討してください。
ファナックロボットを安全にご使用していただくために、「FANUC Robot series 安全マニュアル(B-80687JA)」を必ずお読みいただき、指示にしたがってください。

1 使用者

使用者を以下のように定義します。

- **オペレータ**
  ロボットの電源 ON/OFF を行う。
  操作パネルからロボットプログラムを起動する。
- **プログラマ／教示作業者**
  ロボット操作を行う。
  安全防護領域内でロボットの教示などを行う。
- **保守技術者**
  ロボット操作を行う。
  安全防護領域内でロボットの教示などを行う。
  保守（修理、調整、交換）を行う。

“オペレータ”は安全防護領域内で作業はできません。
“プログラマ／教示作業者”、“保守技術者”は、安全防護領域内での作業ができます。
日本の『労働安全衛生規則』や各国の法律に基づき、ロボットが設置されている安全防護領域内で作業するために以下が必要です。
・ロボットシステム全体に対するリスクアセスメントを行ってください。
・リスクアセスメントの結果に応じた適切な安全教育を受けてください。

表 1 (a)に安全防護領域外での作業をまとめます。この表で「○」印がある作業項目は、各ロボット作業者が行っても良いことを示しています。

表 1 (a) 安全防護領域外での作業

	オペレータ	プログラマ /教示作業者	保守技術者
制御装置電源の ON/OFF	○	○	○
運転モードの選択 (AUTO, T1, T2)		○	○
リモート/ローカルモードの選択		○	○
教示操作盤でのプログラム選択		○	○
外部機器でのプログラム選択		○	○
操作盤でのプログラムスタート	○	○	○
教示操作盤でのプログラムスタート		○	○
操作盤でのアラームリセット		○	○
教示操作盤でのアラームリセット		○	○
教示操作盤でのデータ設定		○	○
教示操作盤での教示		○	○
操作盤での非常停止	○	○	○
教示操作盤での非常停止	○	○	○

	オペレータ	プログラマ /教示作業者	保守技術者
操作盤の保守			○
教示操作盤の保守			○

ロボットの操作、プログラミング、保守を行う場合、オペレータ、プログラマ／教示作業者、保守技術者は、安全に気をつけて少なくとも以下の物を身に付けて作業しなければなりません。

－ 作業内容に適した作業服
－ 安全靴
－ ヘルメット

2 安全に関する表記の定義

本マニュアルでは、使用者の安全および機械の破損防止のために、安全に関する注意事項の程度に応じて、本文中に『**警告**』および『**注意**』の表記をしています。また、補足的な説明を記述するために『**注**』の表記をしています。使用する前に、『**警告**』、『**注意**』、『**注**』に記載されている事項をよく読んでください。

シンボル	定義
⚠ **警告**	取扱いを誤った場合に、使用者が死亡または重傷を負う危険の状態が生じることが想定される場合に用いられます。
⚠ **注意**	取扱いを誤った場合に、使用者が軽傷を負うかまたは物的損害のみが発生する危険の状態が生じることが想定される場合に用いられます。
注	警告または注意以外のことで、補足的な説明を記述する場合に用いられます。

目次

安全にご使用いただくために s-1

1 概要 1
1.1 エンドユーザ向け機能 1
1.2 プラグインソフト開発者向け機能 3
1.3 制約事項 4
2 プラグインソフト開発環境 5
2.1 各構成ファイルの作成とトランスレート 5
2.2 プラグインソフト開発SDK 5
3 プラグインソフトのインストールと管理 7
3.1 プラグインソフトのインストール画面 7
3.2 プラグインソフト一覧画面 9
4 インストールパッケージのファイル構成 12
5 インストール定義ファイル 15
5.1 インストール定義ファイルの書式 15
5.2 システム設定のためのコマンド 20
5.3 アンインストール処理の概要 33
6 プラグインソフト開発概要 34
6.1 プラグイン画面で参照するKAREL変数 34
6.2 プラグイン画面の実装 35
6.3 画面データ辞書ファイルの実装 35
6.4 プラグイン画面の実装例 36
7 プラグイン画面作成 37
7.1 基本情報 37
7.2 ビルドツール 38
7.3 iHMIコンポーネントの利用 41
7.4 iHMIコンポーネント部品 45
7.5 ロボット制御装置とのインタフェース 67
7.6 top.frpl関数のデータ型 104
7.7 メニュー追加 108
7.8 ROBOGUIDEでの画面の確認 111
8 プラグイン画面作成 113
8.1 構成ファイル 113
8.2 プラグイン命令定義ファイル 114
8.3 プラグイン命令実行時に呼び出されるプログラム 119
8.4 プラグイン命令アイコンの作成 129
8.5 プラグイン命令の詳細画面作成 129
8.6 プラグイン命令のエラー一覧 132
9 インストール時のエラー表示 133
9.1 エラー表示一覧 133
10 ツールI/F (CRXシリーズのみ) 135
10.1 ツールI/F仕様 135

11 ツールI/F通信機能 (CRXシリーズのみ)…… 139
11.1 ツールI/F通信機能の共通設定……139
11.2 デジタル出力信号の通信設定……140
11.3 アナログ入力信号の通信設定……141
11.4 シリアル通信の通信設定……143
11.5 シリアル通信用KARELビルトイン……144
11.6 シリアル通信サンプルプログラム……146
11.7 アラーム一覧……150
12 ツールチェンジ機能 (CRXシリーズのみ)…… 152
12.1 手動ツールチェンジ画面……152
12.2 切り離し・結合アイコン命令……156
12.3 ツールチェンジ機能の制約事項……157
12.4 ツールチェンジ機能で発生するアラーム……158
12.5 プラグインソフト開発者向け情報……159
13 R-30iB Mini Plus周辺機器インタフェース…… 164
13.1 周辺機器インタフェース(JRM18)……165
13.2 周辺機器インタフェース(JRM18)と周辺機器の接続……167
13.3 周辺機器インタフェース(JRM18)と周辺機器の接続(端子台変換モジュールの場合)……171
13.4 周辺機器のデジタル入出力信号規格……176
13.5 外部への24V供給……178
14 Eclipse開発環境…… 179
14.1 動作環境……179
14.2 Eclipse開発環境準備……180
14.3 KAREL開発環境の利用手順……181
14.4 iHMIコンポーネント開発環境の利用手順……194
15 CRXシリーズ機構部…… 200
15.1 機構部仕様……200

1 概要

プラグイン機能では、弊社ロボットのプラグインとして、サードパーティ各社が開発するソフトウェアをインストールおよび管理できます。弊社ロボット機構部の接続および電気インタフェースの標準化と併せ、本機能はロボットの周辺機器を簡単に接続・実装することを可能とします。本書では、弊社ロボットのプラグインとしてインストールするプラグインソフトウェアの開発に必要な情報を提供します。

プラグイン機能の概要を図 1 (a) に示します。以下の図ではグリッパを例として挙げていますが、センサシステム、ビジョンシステム、その他各種アプリケーションなど、様々な用途のソフトウェアをプラグインとして開発して頂けます。

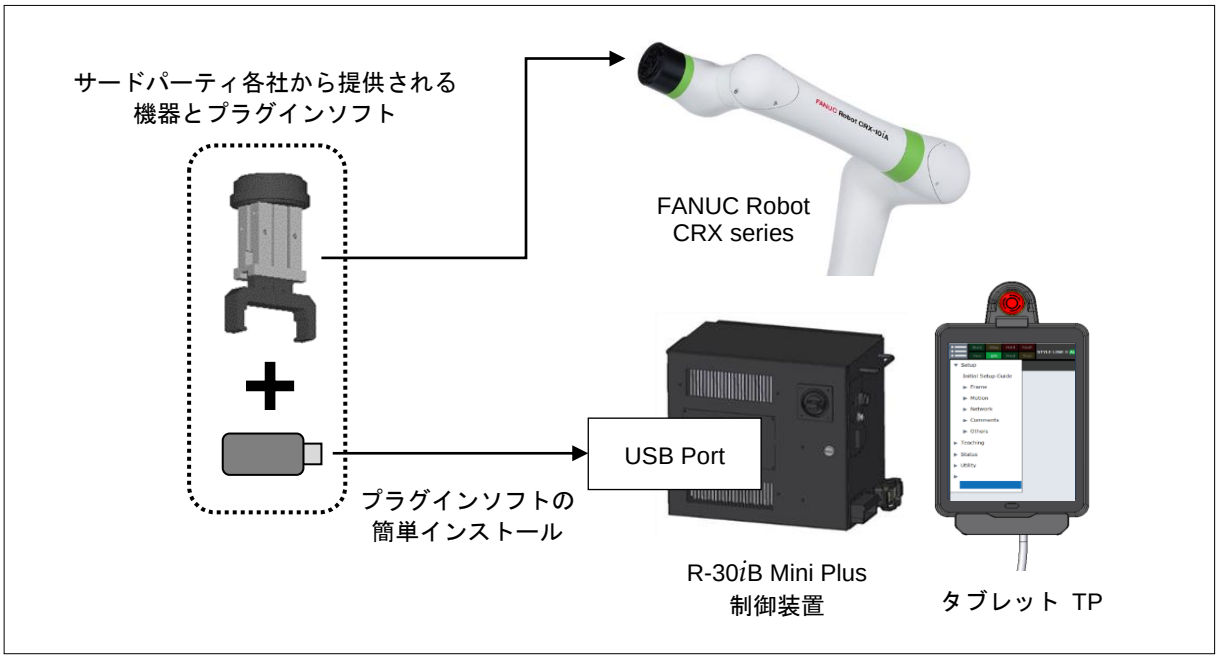


図 1 (a) プラグイン機能の概要

1.1 エンドユーザ向け機能

本機能は、エンドユーザ向けに以下の 2 つの画面を備えています。全てのプラグインソフトのインストール・管理は、この 2 画面から行います。

- **インストール画面**

USB メモリを介して、サードパーティ各社が開発・提供するプラグインソフトを簡単にインストールできます。

- **プラグインソフト一覧画面**

インストール済みのプラグインソフトの一覧・詳細表示、および、アンインストール操作を行えます。

> ⚠ **注意**
> 1 プラグイン対応周辺機器を接続する際は、周辺機器を EE コネクタに接続する前に、プラグインソフトのインストールを完了してください。インストール前に周辺機器を接続した場合、周辺機器が故障するおそれがあります。
> 2 EE コネクタに接続される周辺機器を取り外す、もしくは接続する場合は、必ずロボット制御装置の電源を落としてください。
> 3 プラグインソフトをインストールする時は、必ずプラグインソフト(*.ipl ファイル)を USB メモリまたは FTP クライアントデバイスのルートディレクトリに置いてください。

1.1.1 プラグインソフトのインストール

USB メモリまたは FTP クライアントデバイスを介して、サードパーティ各社が開発・提供するプラグインソフトを簡単にインストールできます。プラグインソフトのインストール操作はインストール画面から行います。インストール画面では、インストールしようとするプラグインソフトのプロバイダや、対応する周辺機器の名称、プラグインソフト版数等の詳細情報が視覚的に分かり易く表示されます。プラグインソフトをインストールすることで、周辺機器専用の設定画面(プラグイン画面)や制御命令(プラグイン命令)の追加、その他各種設定が可能です。

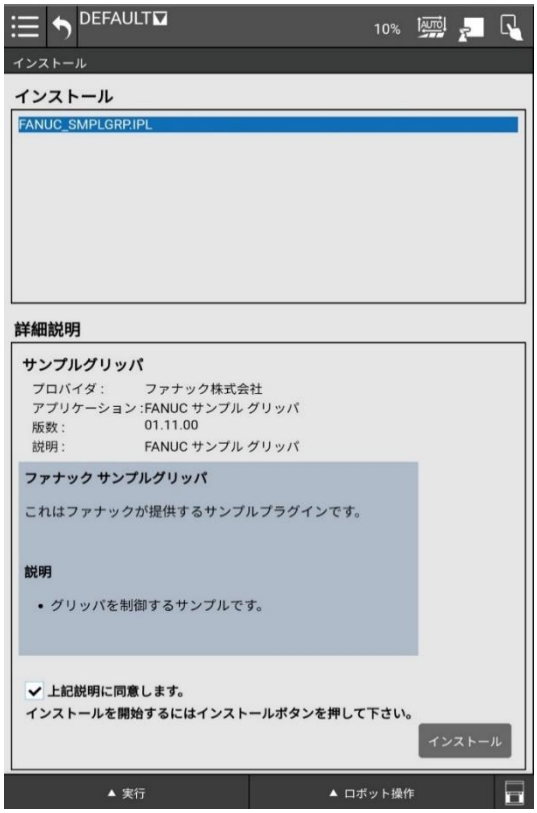


図 1.1.1 (a) プラグインインストール画面

注
1 プラグインインストール画面では UD1、UT1、C1～C8 に置かれたプラグインソフトをインストールできます。ただし、UT1、C1～C8 からインストールするには V9.40P/58 以降のソフトウェアが必要です。
2 C1～C8 を使用する場合は FTP クライアントデバイスを正しく設定し、ファイル画面からデバイスに置かれたファイルを参照できることを確認してください。クライアントデバイスの設定方法についてはイーサネット機能取扱説明書(B-82974JA)の「FTP 操作」の章を参照してください。
3 本機能ではアクティブモードのクライアントデバイスのみサポートしています。
4 UT1 からプラグインソフトをインストールする場合、タブレット TP を使用してください。PC 上で iRProgrammer を使用する場合、UT1 以下のプラグインソフトは画面上に表示されません。

1.1.2 プラグインソフトの管理

インストール済みのプラグインソフトは、プラグインソフト一覧画面で管理できます。この画面では、既にインストールされているプラグインソフトのプロバイダや、対応する周辺機器の名称、プラグインソフト版数等の詳細情報を、エンドユーザに視覚的に分かり易く一覧表示します。さらに、画面右下の「アンインストール」ボタンをタップすると、プラグインソフトのアンインストールも可能です。



図 1.1.2 (a) プラグインソフトー覧画面

1.2 プラグインソフト開発者向け機能

プラグイン機能では、各種周辺機器の設定や状態表示を行うプラグイン画面や、各種周辺機器を制御するプラグイン命令等を開発できます。



図 1.2 (a) プラグインソフト開発例 (左:プラグイン画面 / 右:プラグイン命令)

タブレット TP では HTML ベースの画面を Web ブラウザ上に表示します。そのため、一般的な Web 知識(HTML, JavaScript, CSS)で画面開発を進められます。また、弊社が提供する iHMI コンポーネントを使用すると、画面開発に必要となるコードをすべて HTML, JavaScript, CSS で作成する必要がなく、iHMI デザインの画面開発をスムーズに進められます。

また、グリッパ等の周辺機器を制御するためのプラグイン命令も開発できます。図 1.2 (a) 右図のように、プラグイン命令を独自のアイコンで追加でき、さらに、その制御命令に必要な引数パラメータを視覚的に設定・操作する詳細画面を追加できます。プラグイン命令プログラムは、ロボット TP プログラムで開発でき、複雑な処理が必要な場合には KAREL プログラムで開発することもできます。

開発情報の詳細は 2 章以降をご参照ください。

1.3 制約事項

プラグイン機能には以下の制限があります。

- 本機能はタブレット TP 上でのみ使用可能です。従来の *i*Pendant では使用できません。
- 本機能は V9.40 系列(7DF5 系列)ソフトウェアでのみ使用できます。

2 プラグインソフト開発環境

本章ではプラグインソフト開発環境を説明します。

2.1 各構成ファイルの作成とトランスレート

プラグインソフトに含まれる各構成ファイルの作成、およびトランスレートする環境を説明します。

- TP プログラム

TP プログラムは、ROBOGUIDE もしくは CRX 実機(R-30iB Mini Plus 制御装置)で作成します。TP プログラムについては 8 章「プラグイン命令作成」を参照してください。



図 2.1 (a) TP プログラムの作成

- KAREL プログラム

KAREL プログラムは、ROBOGUIDE もしくは Eclipse 開発環境(KAREL 開発環境)で作成し、トランスレートします。Eclipse 開発環境の利用方法については、13 章「Eclipse 開発環境」を参照してください。



図 2.1 (b) KAREL プログラムの作成・トランスレート

- プラグイン画面やプラグイン命令の詳細画面、メニュー・命令定義ファイル

プラグインソフトに含まれるプラグイン画面やプラグイン命令の詳細画面、メニュー・命令定義ファイル等は、ビルドツール(ihmi_build)、もしくは Eclipse 開発環境(iHMI コンポーネント開発環境)でトランスレートします。ビルドツールについては 7 章「プラグイン画面作成」、Eclipse 開発環境の利用方法については 13 章「Eclipse 開発環境」を参照してください。

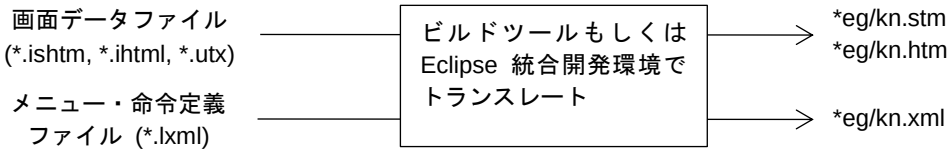


図 2.1 (c) 画面データ・定義ファイルのトランスレート

2.2 プラグインソフト開発 SDK

本書を含むプラグインソフト開発 SDK(Software Development Kit)には、以下の内容物が含まれます。

表 2.2 (a) プラグインソフト開発 SDK に含まれる内容物

内容物	説明	関連章
FANUC_SmplGrp	ビルド前のソースコードを含むプラグインパッケージの実装サンプルです。本フォルダには以下が含まれます。	

内容物		説明	関連章
	output	FANUC_SmplGrp.ipl を解凍した内容です。プラグインソフトは必要なファイル全てを圧縮して、拡張子を IPL に変更することでロボット制御装置にインストールできるようになります。	4 章
	kl_ls	サンプルプラグインソフトに含まれる TP プログラム(アスキー形式)と KAREL プログラムのソースファイルを格納しています。	6 章、8 章
	src	サンプルプラグインソフトをビルドツール(ihmi_build)でビルドする前のソースファイルを格納しています。	7 章
	FANUC_SmplGrp.ipl	サンプルプラグインソフトのインストーラパッケージファイルです。ROBOGUIDE や CRX 実機にそのままインストールしてお試し頂けます(本サンプルは特定の周辺機器を制御するものではありません)	3 章
fanuc_support		プラグインソフトで使用するプログラムファイルの開発を補助するための Eclipse 開発環境です(Eclipse 開発環境の利用は必須ではありません)。本開発環境では KAREL と iHMI コンポーネントの開発環境を利用できます。	13 章
ihmi_build		iHMI コンポーネント画面のビルドツールです。iHMI コンポーネントは、ファナックから提供する画面開発用 UI ライブラリです。iHMI コンポーネントを使用することは必須ではありませんが、画面開発に必要となるコードをすべて HTML, JavaScript, CSS で作成する必要がなく、iHMI デザインの画面開発をスムーズに進めることができます。	7 章
A-97606-6419EN/JA_**.pdf		プラグインソフト開発手順書です(本書)。	-
FANUC_Dev001_Enable_DevMode.ipl		プラグイン開発に利用する ROBOGUIDE もしくは CRX 実機において、開発者モードを有効にするプラグインソフトです。インストールすると、以下の効果があります。 • V9.40P/18 以降で、従来 iPendant 互換モードのプログラム編集画面でプラグイン命令の引数を編集可能にします。アンインストールすると編集禁止に戻ります。	-

3 プラグインソフトのインストールと管理

本機能は、エンドユーザ向けに以下の 2 つの画面を備えています。全てのプラグインソフトのインストール・管理は、この 2 画面から行います。

- **インストール画面**
  USB メモリまたは FTP クライアントデバイスを介して、サードパーティ各社が開発・提供するプラグインソフトを簡単にインストールできます。

- **プラグインソフト一覧画面**
  インストール済みのプラグインソフトの一覧・詳細表示、および、アンインストール操作を行えます。

> ⚠ **注意**
> 1 プラグイン対応周辺機器を接続する際は、周辺機器を EE コネクタに接続する前に、プラグインソフトのインストールを完了してください。インストール前に周辺機器を接続した場合、周辺機器が故障するおそれがあります。
> 2 EE コネクタに接続される周辺機器を取り外す、もしくは接続する場合は、必ずロボット制御装置の電源を落としてください。
> 3 プラグインソフトをインストールする時は、必ずプラグインソフト(*.ipl ファイル)を USB メモリまたは FTP クライアントデバイスのルートディレクトリに置いてください。

3.1 プラグインソフトのインストール画面

USB メモリまたは FTP クライアントデバイスを介して、サードパーティ各社が開発・提供するプラグインソフトを簡単にインストールできます。プラグインソフトのインストール操作はインストール画面から行います。インストール画面では、インストールしようとするプラグインソフトのプロバイダや、対応する周辺機器の名称、プラグインソフト版数等の詳細情報が視覚的に分かり易く表示されます。プラグインソフトをインストールすることで、周辺機器専用の設定画面(プラグイン画面)や制御命令(プラグイン命令)の追加、その他各種設定が可能です。

> **注**
> 1 プラグインインストール画面では UD1、UT1、C1～C8 に置かれたプラグインソフトをインストールできます。ただし、UT1、C1～C8 からインストールするには V9.40P/58 以降のソフトウェアが必要です。
> 2 C1～C8 を使用する場合は FTP クライアントデバイスを正しく設定し、ファイル画面からデバイスに置かれたファイルを参照できることを確認してください。クライアントデバイスの設定方法については、イーサネット機能取扱説明書(B-82974JA)の「FTP 操作」の章を参照してください。
> 3 本機能ではアクティブモードのクライアントデバイスのみサポートしています。
> 4 UT1 からプラグインソフトをインストールする場合、タブレット TP を使用してください。PC 上で iRProgrammer を使用する場合、UT1 以下のプラグインソフトは画面上に表示されません。

手順 3-1 プラグインソフトのインストール

手順

1 ステータスバーのメニューアイコンをタッチすると、以下のようなメニューが表示されます。



図 3.1 (a) メニューアイコン押下時に表示されるメニュー

2 「プラグイン」の下の「インストール」をタップすると、インストール画面が表示されます。インストール画面では、挿入されている USB メモリ(UD1)、タブレット TP アプリの内部メモリ(UT1)およびクライアントデバイス(C1～C8)に置かれているプラグインソフトのインストールパッケージファイルのファイル名、および、プラグインソフトの詳細説明が表示されます。タブレットの内部メモリ(UT1)の使用方法は、タブレット UI 取扱説明書(B-84274JA)の「USB メモリ操作（タブレット TP アプリ V1.12 以降）」の節を参照してください。

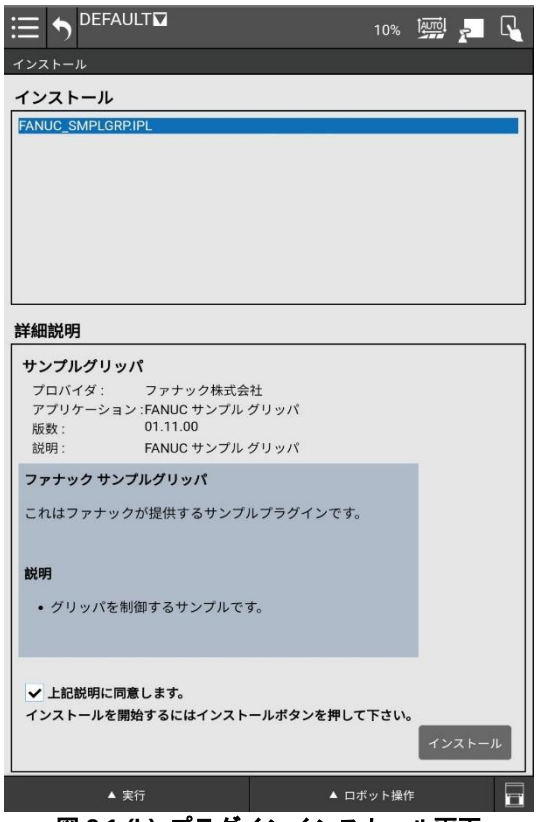


図 3.1 (b) プラグインインストール画面

3 画面右下の「インストール」ボタンをタップすると、プラグインソフトのインストールが始まります。

4 インストール処理が完了すると以下の画面が表示されます。



図 3.1 (c) インストール処理完了画面

5 ロボットの電源を再投入して、インストール操作は完了です。

3.2 プラグインソフトー覧画面

インストール済みのプラグインソフトは、プラグインソフトー覧画面で管理できます。この画面では、既にインストールされているプラグインソフトのプロバイダや、対応する周辺機器の名称、プラグインソフト版数等の詳細情報を、エンドユーザに視覚的に分かり易く一覧表示します。さらに、画面右下の「アンインストール」ボタンをタップすると、プラグインソフトのアンインストールも可能です。

手順 3-2 プラグインソフトー覧画面の操作方法

手順

1 ステータスバーのメニューアイコンをタッチすると、以下のようなメニューが表示されます。



図 3.2 (a) メニューアイコン押下時に表示されるメニュー

2 「プラグイン」の下の「プラグインソフト一覧」をタップすると、プラグインソフト一覧画面が表示されます。



図 3.2 (b) プラグインソフト一覧画面

3 一覧表示されたプラグインソフトのいずれかをタップすると、そのプラグインソフトが選択され、画面の下半分に選択されたプラグインソフトの詳細情報が表示されます。

4 このとき、画面右下の「アンインストール」ボタンをタッチすると選択されたプラグインソフトをアンインストールできます。

5 アンインストール処理が完了すると、以下の画面が表示されます。ロボットの電源を再投入して、アンインストール操作は完了です。



図 3.2 (c) アンインストール完了画面

4 インストールパッケージのファイル構成

プラグインソフトのインストールパッケージは下表に示すファイル群で構成します。下表に記載するファイルを一つのアーカイブファイルにまとめ、".IPL"というファイル拡張子にしたものがインストールパッケージになります。これを USB メモリに入れて提供する、各社ウェブサイトからダウンロードする等の手段でエンドユーザに提供してください。

表 4 (a) インストールパッケージのファイル構成例

ファイル	ファイル名	拡張子	用途	関連章
インストール定義ファイル	Autocfg.csv	*csv	プラグインソフトをロボット制御装置内にインストールするファイルや設定を定義します。また、ロボットの設定変更したい場合、その設定内容を記述します。ファイル名は必ず"autocfg.csv"としてください。	5 章 11 章
多国語対応インストール定義ファイル	plginfo**.dat (ただし**は国を表す 2 文字)	*dat	PLGINFO コマンドを各国語で記述できます。現在選択中の言語に対応するファイルが存在しない場合には Autocfg.csv 内の PLGINFO コマンドがデフォルトとして使用されます。ファイル名は必ず" plginfo**.dat" (ただし**は国を表す 2 文字)としてください。	5 章
詳細情報テキストファイル	任意	*txt *htm	プラグインソフトの詳細情報が記述されたテキストファイルです。プラグイン一覧画面に表示します。標準言語(英語)で作成してください。各国語対応のファイルを追加する事も可能です。	3 章 5 章
ライセンス情報テキストファイル	任意	*txt *htm	ライセンス情報などが記述されたテキストファイルです。プラグインインストール画面に表示します。標準言語(英語)で作成してください。各国語対応のファイルを追加する事も可能です。	5 章
一覧画面用画像ファイル 1	任意	*png *jpg *bmp *gif	プラグインソフトの詳細情報と合わせて表示する画像ファイルです。プラグイン一覧画面に表示します。プラグインソフトが制御する周辺機器や、会社ロゴの画像を貼ることができます。使用は任意です。縦横比が 5:4 になるようにすると表示スペースに適合する大きさになります。表示およびインストール処理時間を適切に保つために画像ファイルの大きさは 100kB 以下になるようにしてください。	3 章 5 章
一覧画面用画像ファイル 2	任意	*png *jpg *bmp *gif	プラグインソフトの詳細情報と合わせて表示する画像ファイルです。プラグイン一覧画面に表示します。プラグインソフトが制御する周辺機器や、会社ロゴの画像を貼ることができます。使用は任意です。縦横比が 5:4 になるようにすると表示スペースに適合する大きさになります。表示およびインストール処理時間を適切に保つために画像ファイルの大きさは 100kB 以下になるようにしてください。	3 章 5 章
コマンドファイル	任意	*cf	インストール・アンインストール時に実行するコマンドファイルです。プラグイン命令プログラム等をロボット制御装置にロードするコマンド等を記述します。	5.2 節
メニュー情報ファイル	plmenu**.xml (ただし**は国を表す 2 文字)	*xml	プルダウンメニューにプラグイン画面名を追加するための定義ファイルです。	7.6 節

ファイル	ファイル名	拡張子	用途	関連章
プラグイン画面データファイル	任意	*stm *htm *js *css	プラグイン画面のデータファイルです。言語毎に生成されます。ファイル名は任意ですが、メニュー情報ファイル(plmenu**.xml) 内の定義と一致する必要があります。 必要に応じて、設定範囲チェック、関連要素の操作、Callback 関数などを記述する*js ファイルや、独自スタイルを適用するための*css ファイルも追加できます。	6 章 7 章 13 章
プラグイン画面用画像ファイル	任意	*png *jpg *bmp *gif	周辺機器メーカが開発するプラグイン画面に表示する画像ファイルです。使用は任意です。表示およびインストール処理時間を適切に保つために画像ファイルの大きさは 100kB 以下になるようにしてください。	6 章 7 章
プラグイン命令定義ファイル	任意	*xml	以下の各項目を関係づけるためのファイルです。言語毎に生成されます。 ・ 各プログラムのファイル名 ・ 表示用アイコン名称 ・ 詳細画面表示名称 ・ 詳細画面表示用 Callback 関数名 ファイル名には半角英数字のみが使用できます。ファイル名は任意です。	8 章
プラグイン命令 TP プログラム	“IPL_(プラグイン ID)_(任意のプログラム名).tp”	*tp	プラグイン命令実行時に呼び出される TP プログラムです。ファイル名は必ず、” IPL_(プラグイン ID)_(任意のプログラム名).tp”としてください。	5 章 8 章
KAREL プログラム	“IPL_(プラグイン ID)_(任意のプログラム名).pc”	*pc	KAREL 変数を定義するファイルです。また、TP プログラムで記述できないような、複雑な制御処理を必要とするプラグイン命令プログラムを作成する場合にも KAREL プログラムを作成します。ファイル名は必ず、”IPL_(プラグイン ID)_(任意のプログラム名).pc”としてください。	5 章 6 章 8 章 10 章 11 章 12 章
プラグイン命令アイコンデータファイル	任意	*png	ihmieditor でプラグイン命令として表示されるアイコン画像ファイルです。ファイル名は任意ですが、プラグイン命令定義ファイル(*.xml) 内の定義と一致する必要があります。表示およびインストール処理時間を適切に保つために画像ファイルの大きさは 100kB 以下になるようにしてください。	8 章
プラグイン命令詳細画面データファイル	任意	*stm *htm *js *css	周辺機器を制御するプラグイン命令の詳細画面のデータファイルです。言語毎に生成されます。ファイル名は任意ですが、プラグイン命令定義ファイル(*.xml) 内の定義と一致する必要があります。 必要に応じて、設定範囲チェック、関連要素の操作、Callback 関数などを記述する*js ファイルや、独自スタイルを適用するための*css ファイルも追加できます。	8 章 13 章
プラグイン定義アラームファイル	plgalm.dat plgalm**.dat (ただし**は国を表す 2 文字)	*dat	プラグインソフトで任意のアラームメッセージを定義するファイルです。標準言語として英語のメッセージテキストを plgalm.dat に記入してください。多国語対応のためには、国名を表す 2 文字を付加したものをファイル名として、IPL ファイル中に追加してください。例えば、日本語のアラームテキストファイルは plgalmkn.dat となります。	8.3.3 節

4. インストールパッケージのファイル構成

注

- インストールパッケージファイル(*.IPL ファイル)内にディレクトリ階層構造を持つことはできません。インストールパッケージの全ての構成ファイルを同じ階層のディレクトリに置いて、アーカイブファイルにまとめてください。
- 表示およびインストール処理時間を適切に保つため、インストールパッケージ内の画像ファイルの大きさは 100kB 以下になるようにしてください。
- インストールパッケージ(*.IPL ファイル)のファイル名はプラグイン ID と同名にしてください。プラグイン ID については 5.1 節「インストール定義ファイルの書式」を参照してください。

以降の章で、インストールパッケージを構成する各ファイルについて説明します。

5 インストール定義ファイル

プラグインソフトをロボット制御装置内に定義するためにインストール定義ファイル(Autocfg.csv)を作成します。ファイル名は必ず"Autocfg.csv"としてください(CSV 形式ファイル: Comma Separated Values、各データをカンマで区切った形式のファイル)

Autocfg.csv は、ロボット設定支援ツール(A05B-2600-J737)で使用できるコマンドと同じコマンドが使用でき、DI/DO 点の定義やツール座標系などの設定が可能です。ロボット設定支援ツールで使用可能なコマンドの詳細については、システム設計ツール取扱説明書(B-83274JA)の 4 章「ロボット設定支援ツール」を参照してください。

本機能では、ロボット設定支援ツールで使用可能なコマンドに加え、プラグインソフトのインストール・アンインストール処理に必要な以下のコマンドを新規追加しています。また、アンインストール処理の際にも、インストール処理同様 Autocfg.csv を使用してアンインストールの処理を行います。

表 5 (a) プラグインソフトで利用できる新規コマンド

コマンド	説明
PLGINFO コマンド	プラグインソフトに関する情報を記述します
PLGID コマンド	プラグインソフトの ID 情報とソフトウェア版数を記述します
OPTION コマンド	プラグインソフトのインストール方法を変更します
REQUIRE コマンド	プラグインソフトが動作するために必要とされるオプション機能を指定します
CHECKVER コマンド	プラグインソフトが動作するために必要なロボット制御装置のソフトウェア版数を指定します
DESCFILE コマンド	プラグインソフトの詳細情報や画像ファイル、インストール時のライセンス情報のファイル名を記述します
RUNCM コマンド	コマンドファイルを実行します
LOAD コマンド	指定したプログラム(TP、KAREL)をロードします
RUNPROG コマンド	指定したプログラム(TP、KAREL)を起動します
VAR コマンド	KAREL 変数を定義・初期設定します
DSET コマンド	KAREL 変数を設定します
PLGMTOP コマンド	プラグインメニューを最上部に移動します

注
PLGINFO, PLGID, OPTION, REQUIRE, CHECKVER, DESCFILE, PLGMTOP コマンドはロボット設定支援ツールオプション(A05B-2600-J737)では使用できません。

本章では、プラグインソフトのインストールのための Autocfg.csv ファイルの内容について説明します。

5.1 インストール定義ファイルの書式

プラグインソフトのインストール･アンインストールを定義する Autocfg.csv の一例を図 5.1 (a) に示します。CSV ファイルの処理のため、各ラインの終わりに終端文字列(!!!!)を記述する必要があります(コメント行 REM を除く)。

```
REM,autocfg.csv,V1.0
PLGINFO,SmplGrp,FANUC Sample Gripper,FANUC Corporation,FANUC Sample Gripper,!!!!
PLGID,FANUC_SmplGrp,01.01.01,!!!!
DESCFILE,DESCRIPTION.HTM,fanuclogo09.png,fanuclogo01.jpg,!!!!

LOAD,IPL_FANUC_SmplGrp_Cfg.PC,!!!!
LOAD,IPL_FANUC_SMPLGRP_OPEN.TP,!!!!
LOAD,IPL_FANUC_SMPLGRP_CLOSE.TP,!!!!

DSET,IPL_FANUC_SMPLGRP_Cfg,OpenDo,INTEGER,0,!!!!
DSET,IPL_FANUC_SMPLGRP_Cfg,ClosDo,INTEGER,0,!!!!
DSET,IPL_FANUC_SMPLGRP_Cfg,StatDi,INTEGER,0,!!!!
END
```

図 5.1 (a) Autocfg.csv の書式

1. Autocfg.csv の 1 行目は、必ず以下の REM 文(コメント行)にする必要があります。

```
REM,autocfg.csv,V1.0
```

この行を追加することにより、この Autocfg.csv ファイルはプラグインソフトのインストール・アンインストールのための定義ファイルとして認識されます。

2. Autocfg.csv の 2 行目には、PLGINFO コマンドによりプラグインソフトの名前や、その他の基本情報を記載します。

```
PLGINFO,Name,Desc,Publisher,Application,!!!!
```

Name	プラグインソフトの名前
Desc	プラグインソフトの簡単な説明
Publisher	プラグインソフトのプロバイダ(メーカ)の会社名
Application	このプラグインソフトが使用される主な目的

各項目は 32 文字以内にしてください。途中に「,」(カンマ)が入る場合には「"」(二重引用子)で囲うようにしてください。

3. Autocfg.csv の 3 行目以降には、PLGID コマンド、OPTION コマンド、REQUIRE コマンド、CHECKVER コマンドを任意の順に指定できます。PLGID コマンドはプラグインソフトの識別子(ID)を定義します。PLGID コマンドは Autocfg.csv ファイル中に一度だけ指定できます。
OPTION コマンドについては、5.1.3 節「インストール方法設定 OPTION コマンド」を参照してください。
REQUIRE コマンドについては、5.1.4 節「REQUIRE コマンド」を参照してください。
CHECKVER コマンドについては、5.1.5 節「CHECKVER コマンド」を参照してください。

```
PLGID,プラグイン ID,Version,!!!!
```

プラグイン ID	プラグインソフトを識別するための識別子を指定します。この識別子は他社のプラグインソフトを含めユニークである必要があります。識別子は"(会社名)_(プラグイン名)"とすることによりユニークな名前としてください。使用できる文字列は英数字およびアンダーバー"_"で 16 文字以内です。 例) FANUC_SmplGrp 1234567890123456
Version	版数を示す「MM.RR.SS」形式の数字列。ここで、MM は大版数、RR は改版数、SS はサブ改版数を示します。

注
- プラグイン ID を"(会社名)_(プラグイン名)"として定義し、TP プログラムと KAREL プログラムのファイル名を"IPL_(プラグイン ID)_(任意のプログラム名)"とする必要があります。例えば、プラグイン ID を "**FANUC_SmplGrp**" と定義した場合、プログラム名は **IPL_FANUC_SmplGrp_*****.TP, **IPL_FANUC_SmplGrp_*****.PC とする必要があります。ここで、"***"は任意のプログラム名を意味します。
- TP プログラムのパラメータ命令でパラメータ名として入力できる文字列は最大 30 文字です。KAREL 変数を指定する場合、KAREL プログラム名と KAREL 変数名をパラメータ命令に指定する必要があります。この制約を踏まえてプラグイン ID、KAREL プログラム名、KAREL 変数名を決定してください。
例）ﾚｼﾞ[10001]=$[IPL_FANUC_SmplGrp_Cfg]OpenDo
123456789012345678901234567890

4. Autocfg.csv ファイルの PLGID コマンド、OPTION コマンド、REQUIRE コマンド、CHECKVER コマンドの後に、DESCFILE コマンドによりプラグイン詳細情報や画像ファイル、ライセンス情報ファイル名を定義します。DESCFILE コマンドの後には PLGID コマンド、OPTION コマンド、REQUIRE コマンド、CHECKVER コマンドは指定できません。

```
DESCFILE,desc_file,pic_file,logo_file,[eula_file,[NOCHKBOX,]]!!!!
```

desc_file	プラグインソフトの詳細情報を記述したテキストファイル名(*.txt または*.htm)を示します。テキストファイルは標準言語として英語で作成してください。ファイル名に国名を表す 2 文字を付加したファイルを言語ごとに追加して多国語対応する事も可能です。
pic_file	画像ファイル 1 のファイル名(例 : プラグインソフトが制御する機器の画像ファイル名)
logo_file	画像ファイル 2 のファイル名(例 : 会社ロゴの画像ファイル名)
eula_file (省略可能)	「インストール画面」でライセンス情報などを表示するためのテキストファイル名(*.txt または*.htm)を示します。テキストファイルは標準言語として英語で作成してください。ファイル名に国名を表す 2 文字を付加したファイルを言語ごとに追加して多国語対応する事も可能です。この項目を省略するとライセンス情報などを表示するテキストボックス自体がインストール画面に表示されません。
NOCHKBOX (省略可能)	上記の eula_file 項目にファイル名を指定した場合、承諾を示すチェックボックスをチェックしないと「インストール」ボタンを押下できません。この項目に「NOCHKBOX」と指定するとチェックボックスが非表示となり、チェックを入れなくてもインストールを開始できるようになります。

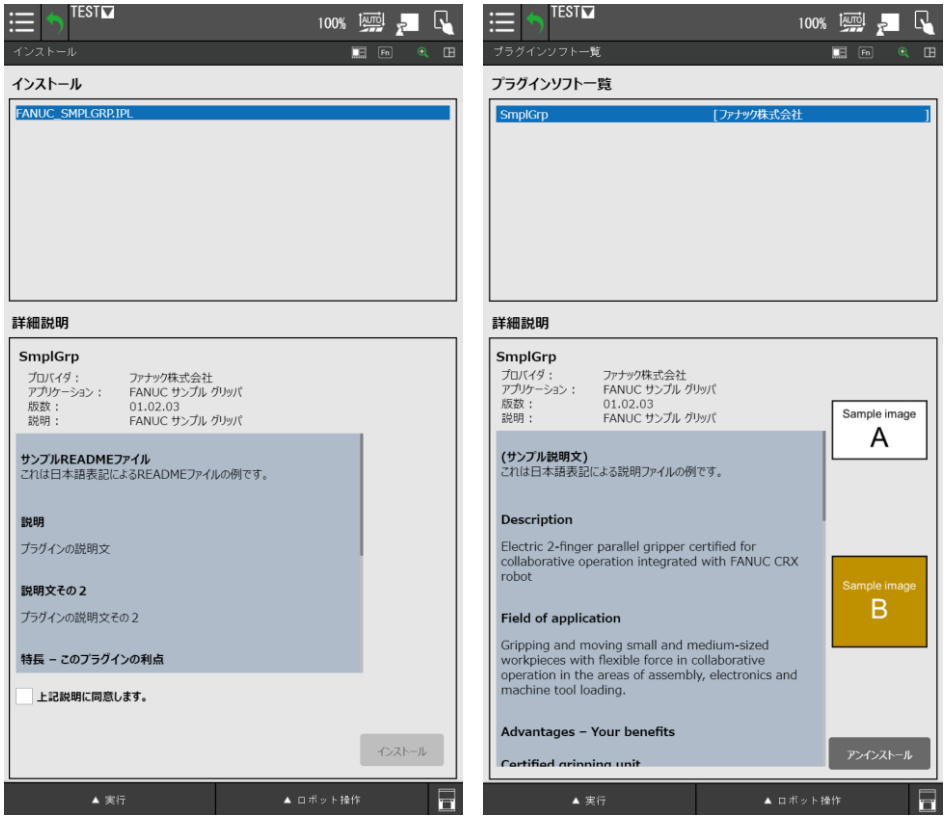


図 5.1 (b) プラグインインストール画面とプラグインソフト一覧画面

注
PLGID コマンド、OPTION コマンド、REQUIRE コマンド、CHECKVER コマンドは、必ず DESCFILE コマンドの前に指定してください。

5. Autocfg.csv ファイルの DESCFILE コマンド以降の部分は、ロボット設定支援ツールのコマンドや、表 5 (a)に記載するコマンドを指定することにより、ロボットの設定を変更できます。

⚠ 注意
プラグインインストール画面のライセンス表示機能は V9.40P/05 版以降で使用できます。プラグインソフトで本機能を使用する場合、周辺機器メーカが提供するプラグインソフト取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/04 版以前の場合は、プラグインソフトをインストールする前に V9.40P/05 版以降の最新版にアップデートしてください」、と明記してください。

注
Autocfg.csv ファイルの改行コードは CR+LF である必要があります。

5.1.1 画像ファイル

DESCFILE コマンドで指定される二つの画像ファイルは「プラグインソフト一覧」画面で表示されます。元の画像の大きさに関わりなく、画面の表示部に収まるように大きさが自動的に調節されます。この時、縦横の辺の長さの比が 5:4 になるようにすると表示スペースにちょうど適合する大きさになります。下図を参照ください。

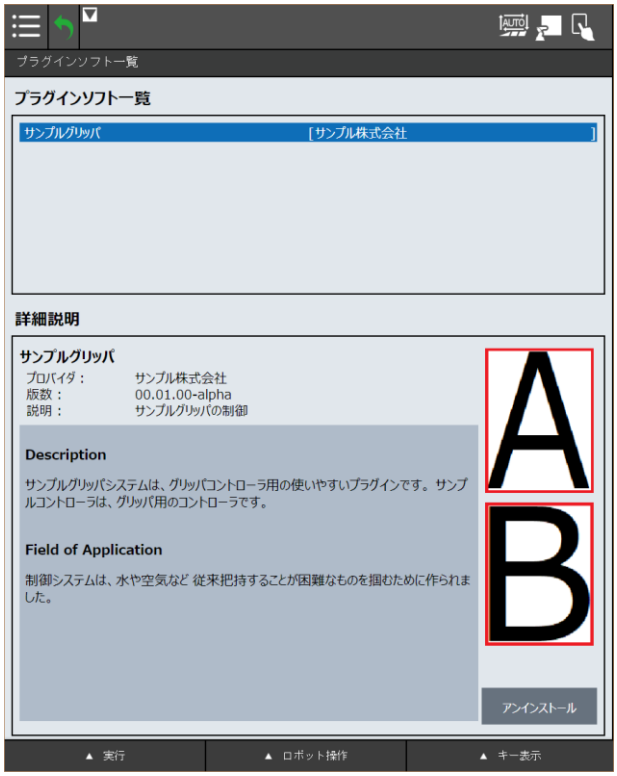


図 5.1.1 (a) プラグインソフト一覧画面での画像の表示

注
表示およびインストール処理時間を適切に保つため、インストールパッケージ内の画像ファイルの大きさは 100kB 以下になるようにしてください。

5.1.2 PLGINFO/PLGSUB コマンドの多国語対応

Autocfg.csv 内の PLGINFO コマンドおよび PLGSUB コマンド(12.5.3 節参照)には標準言語として英語で情報を記述してください。必要に応じ、PLGINFO/PLGSUB コマンドの各国語対応のファイルを追加できます。

ファイル名： plginfo**.dat (ただし**は国名を表す 2 文字を意味します)

ロボット制御装置の言語設定に従って表示言語を切り替えるには、下表に示す正しい接尾辞をファイル名に使用してください。

表 5.1.1 (a) 言語設定とファイル名の接尾辞

言語設定	ファイル名の接尾辞	ファイル名	ファイル内文字コード
英語	eg	(Autocfg.csv の情報が使用されますので plginfo**eg**.dat は必要ありません)	iso-8859-1
日本語	kn	plginfo**kn**.dat	Shift_JIS
中国語	ch	plginfo**ch**.dat	GB2312
台湾語	tw	plginfo**tw**.dat	chinese-big5
ドイツ語	gr	plginfo**gr**.dat	iso-8859-1
フランス語	fr	plginfo**fr**.dat	iso-8859-1
イタリア語	it	plginfo**it**.dat	iso-8859-1
スペイン語	sp	plginfo**sp**.dat	iso-8859-1

plginfo**.dat ファイル内には PLGINFO コマンド一行と必要数分の PLGSUB コマンドだけを指定できます。表 5.1.1 (a) の命名規則に従って、現在選択中の言語に対応する plginfo**.dat ファイルが存在する場合は Autocfg.csv ではなく plginfo**.dat 内の PLGINFO/PLGSUB コマンドが使用されます。現在選択中の言語に対応する plginfo**.dat ファイルが存在しない場合は Autocfg.csv 内の PLGINFO/PLGSUB コマンドが使用されます。

例えば、plginfo**eg**.dat、plginfo**kn**.dat のファイルが存在し、ロボット制御装置の言語が日本語に設定されていた場合、plginfo**kn**.dat 内の PLGINFO/PLGSUB コマンドが使用されます。plginfo**kn**.dat が存在しない場合には、Autocfg.csv 中の PLGINFO/PLGSUB コマンドが使用されます。

plginfo**.dat ファイル内には PLGINFO コマンドおよび PLGSUB コマンド以外は指定しないでください。

注
- プラグインソフトが制御するツール I/F 使用機器が 2 台以上ある場合に限り、plginfo**.dat ファイルに PLGSUB コマンドを指定できます。
- plginfo**.dat ファイルの 1 行目は必ず PLGINFO コマンドを指定してください。plginfo**.dat ファイルに PLGSUB コマンドを指定する場合、PLGINFO コマンドの次の行以降に指定してください。
- PLGINFO/PLGSUB コマンドの行間に空白行を入れないように記述してください。
- PLGSUB コマンドについては、12.5.3 節「ツール I/F 使用機器 2 台目以降のサブ ID 設定」を参照してください。

5.1.3 OPTION コマンド

OPTION コマンドは、プラグインソフトのインストール方法の詳細を設定します。

OPTION, ANYROBOT コマンド

```
OPTION,ANYROBOT,!!!!
```

OPTION,ANYROBOT コマンドを指定しないとプラグインソフトは CRX ロボット以外にインストールできません。OPTION,ANYROBOT コマンドを指定した場合、タブレット TP が接続されている限り、そのプラグインソフトはロボット機種によらずどのロボット機種にもインストールできます。

OPTION,ANYROBOT コマンドを指定しないで CRX ロボット以外にプラグインをインストールしようとすると SDTL-105 (サポートしていないロボット機種です)アラームが表示されます。詳細は 9.1 節「エラー表示一覧」を参照してください。

注
PLGID コマンド、OPTION コマンド、REQUIRE コマンド、CHECKVER コマンドは、必ず DESCFILE コマンドの前に指定してください。

5.1.4 REQUIRE コマンド

REQUIRE コマンドは、プラグインソフトが動作するために必要なオプション機能を指定します。例えば、あるプラグインが動作するために、ユーザソケットメッセージ機能(R648)が必要な場合は、

```
REQUIRE,R648,!!!!
```

と指定してください。指定されたオプション機能がロードされていない場合には SDTL-106 (必要なオプションがロードされていません)アラームが表示され、そのプラグインはインストールできません。アラームについての詳細は 9.1 節「エラー表示一覧」を参照してください。

注
PLGID コマンド、OPTION コマンド、REQUIRE コマンド、CHECKVER コマンドは、必ず DESCFILE コマンドの前に指定してください。

5.1.5 CHECKVER コマンド

CHECKVER コマンドは、プラグインソフトのインストール時に、プラグインソフトが動作するために必要なロボット制御装置ソフトウェア版数をチェックします。

```
CHECKVER,series,version,!!!!
```

series	ロボット制御装置のソフトウェア系列を指定します 例) V9.40
version	ロボット制御装置のソフトウェア版数を指定します 例) 68

例えば、プラグインソフトのインストールを V9.40P/68 版以降でのみ許可する場合、以下のように指定してください。

```
CHECKVER,V9.40,68,!!!!
```

この場合、ロボット制御装置のソフトウェア版数が V9.40P/67 版以前であれば、プラグインソフトのインストール時に「PLUG-031 このプラグインは V9.40P/68 以降が必要です」アラームが表示され、インストールを中断します。

指定した系列の全版数でインストールを禁止する場合は、以下のように version に NA を指定します。

```
CHECKVER,V9.40,NA,!!!!
```

この場合、ロボット制御装置のソフトウェア系列が V9.40 であれば、プラグインソフトのインストール時に「PLUG-033 このプラグインは V9.40 系列では使用できません」アラームが表示され、インストールを中断します。

CHECKVER コマンドは複数行指定できます。複数行指定した場合、1 つでもインストール条件を満たさない行があれば上記アラームが表示され、インストールを中断します。

⚠ 注意
CHECKVER コマンドは V9.40P/58 版以降で使用できます。プラグインソフトで本機能を使用する場合、周辺機器メーカが提供するプラグインソフト取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/57 版以前の場合は、プラグインソフトをインストールする前に V9.40P/58 版以降の最新版にアップデートしてください」、と明記してください。

注
- PLGID コマンド、OPTION コマンド、REQUIRE コマンド、CHECKVER コマンドは、必ず DESCFILE コマンドの前に指定してください。
- CHECKVER コマンドに指定した series とロボット制御装置のソフトウェア系列が一致していない場合は版数チェックがスキップされ、アラームは発生しません。

5.2 システム設定のためのコマンド

以下にプラグインソフトのインストール時に使用するコマンドを説明します。

5.2.1 RUNCM コマンド

RUNCM コマンドは、指定した KCL コマンドファイル(拡張子*.cf)を実行します。

```
RUNCM,cmd_file1,cmd_file2,!!!!
```

cmd_file1	インストール処理の際に実行されます
cmd_file2	アンインストール処理の際に実行されます

コマンドファイル実行の際には、CSV ファイルと同じディレクトリを仮定します。
ディレクトリを変更するためには、CD コマンドを使用できます。

```
CD,path_name,!!!!
```

path_name	ディレクトリのパス名を指定します

5.2.2 LOAD コマンド

LOAD コマンドは、TP プログラム(*.TP)または KAREL プログラム(*.PC)をロードします。

```
LOAD,prog_name.tp,!!!!
LOAD,prog_name.pc,!!!!
```

prog_name.tp	ロードする TP プログラム名を指定します
prog_name.pc	ロードする KAREL プログラム名を指定します

使用例

```
LOAD,IPL_FANUC_SmplGrp_OPEN.TP,!!!!
```

上述の例では"IPL_FANUC_SmplGrp_OPEN"という TP プログラムをロードします。

```
LOAD,prog_name.tp,COLD,!!!!
```

プログラム名の後に「COLD」を付けると、ロードされたプログラムは電源投入の度に起動されます。
アンインストール処理の際には LOAD コマンドでロードされたプログラムは削除されます。

⚠ 注意

- プラグインソフトでロードする KAREL プログラム、TP プログラムでは、WHILE 文や REPEAT 文などで無限ループになり得る処理は実装しないようにしてください。プラグイン画面での特定操作時やプラグイン命令実行時など必要なときのみ呼び出して実行するように作成してください。
- LOAD コマンドのプログラム名の後に COLD オプションを付けると、電源投入の度に起動されます。COLD オプションを指定するプログラムでは、KAREL 変数の設定など即座に終了する処理のみとしてください。

5.2.3 RUNPROG/RUNPRGWT コマンド

RUNPROG/RUNPROGWT コマンドは、プラグインソフトのインストール時、および、アンインストール時に、指定された KAREL プログラムを起動します。

プラグインソフトのインストール時に prog_name1 (KAREL プログラム名) を起動し、アンインストール時に prog_name2 (KAREL プログラム名) を起動する場合は、以下のように記述します。

```
RUNPROG,prog_name1,prog_name2,!!!!
```

ここで、prog_name1 は必須ですが、prog_name2 の指定はオプションです。アンインストール時のプログラム起動が不要な場合は、以下のようにインストール時に起動するプログラム名だけを記述します。

```
RUNPROG,prog_name1,,!!!!
```

prog_name1	インストール処理の際に起動するKAREL プログラム名を指定します。prog_name1 は必ず指定してください。
prog_name2	アンインストール処理の際に起動する KAREL プログラム名を指定します。アンインストール時のプログラム起動が不要な場合は prog_name2 の指定は不要です。

RUNPROG コマンドはプログラムの終了を待たずに続行します。もし、プログラムの実行が終了するまでコマンド実行を待たせたい場合には RUNPRGWT コマンドを使用してください。
RUNPROG/RUNPRGWT コマンドにおいて、prog_name1 の指定は必須、prog_name2 の指定はオプションです。

```
RUNPRGWT,prog_name1,,!!!!
RUNPRGWT,prog_name1,prog_name2,!!!!
```

⚠ 注意
RUNPROG/RUNPRGWT コマンドで呼び出すプログラムは、KAREL 変数の設定など即座に終了する処理のみとし、WHILE 文や REPEAT 文などで無限ループになり得る処理は実装しないでください。

注
- RUNPROG/RUNPRGWT コマンドで呼び出すプログラムは KAREL プログラムを使用してください。RUNPROG/RUNPRGWT コマンドで TP プログラムを呼び出すことはできません。
- RUNPROG/RUNPRGWT コマンドで呼び出す KAREL プログラムには、必ず%NOLOCKGROUP 属性を指定してください。
- RUNPRGWT コマンドは RUNPROG コマンドと同様ですが、CSV コマンドの実行は、このプログラムが終了するまで待たされます。プログラム中で ABORT 文を使用できません。
- RUNPROG/RUNPRGWT コマンドで、インストール時に起動する KAREL プログラム名の指定は必須ですが、アンインストール時に起動する KAREL プログラム名の指定はオプションです。

5.2.4 VAR/DSET コマンド

VAR コマンドは、KAREL 変数を定義します。

```
VAR,prog_name,var_name,type_name,[def_value],!!!!
```

prog_name	KAREL 変数を定義する KAREL プログラム名を指定します
var_name	KAREL 変数名を指定します。KAREL プログラムで使用可能な名前である必要があります。
type_name	KAREL 変数の型名を指定します。INTEGER, REAL, BOOLEAN, STRING[n]が指定可能です(n は文字列の長さを示す数値を指定してください)。本コマンドで利用できるデータ型は表 5.2.4 (a)に示します。
def_value	KAREL 変数の初期値を指定します。変数の型に有効な値を指定してください。

指定されたプログラム名、変数名、変数型の KAREL 変数を CMOS メモリ内に定義します。指定された変数が未初期化の場合には「def_value」で指定された値に初期化されます。別途 KAREL プログラムをロードする必要はありません。

使用例)
```
VAR,IPL_FANUC_SmplGrp_Cfg,OpenDo,INTEGER,12345,!!!!
```

上述の例では、[IPL_FANUC_SmplGrp_Cfg] OpenDo という整数型の変数が CMOS メモリに作成され、初期値 12345 に設定されます。既に同じ名前の変数が存在する場合には新たに作り直さずその変数を使用します。同じ名前の変数に、既に有意の値が設定されていた場合には初期値には設定されません。

注
VAR コマンドでシステム変数を指定しないでください。

DSET コマンドは、KAREL 変数、もしくは、システム変数の値を設定します。

```
DSET,prog_name,var_name,type_name,value,!!!!
```

prog_name	KAREL 変数を定義する KAREL プログラム名を指定します。システム変数の場合は*system*とします。
var_name	KAREL 変数名、もしくは、システム変数名を指定します。KAREL プログラムで使用可能な名前である必要があります。
type_name	KAREL 変数、もしくは、システム変数の型名を指定します。INTEGER, REAL, BOOLEAN, STRING[n]が指定可能です(n は文字列の長さを示す数値を指定してください)。本コマンドで利用できるデータ型は表 5.2.4 (a)に示します。
value	KAREL 変数、もしくは、システム変数の設定値を指定します。変数の型に有効な値を指定してください。

指定された KAREL 変数を指定された値に設定します。VAR コマンドと異なり、変数が存在しない場合には新たに作成しません。

使用例) KAREL 変数(INTEGER)の場合
```
DSET,IPL_FANUC_SmplGrp_Cfg,OpenDo,INTEGER,0,!!!!
```

使用例) KAREL 変数(STRING)の場合
```
DSET,IPL_FANUC_SmplGrp_Cfg,Command,STRING[10],ABCDEF,!!!!
```

使用例) システム変数の場合
```
DSET,*system*,$MNUTOOLNUM[1],INTEGER,1,!!!!
```

VAR/DSET コマンドで利用できるデータ型を表 5.2.4 (a) に示します。

表 5.2.4 (a) VAR/DSET で利用できるデータ型

データ型	説明
BOOLEAN	論理型。TRUE か FALSE の値を持ちます。データ長は 4 バイト。
INTEGER	整数型。データ長は 4 バイト。
REAL	実数型。32 ビットの浮動小数点。データ長は 4 バイト。
STRING[n]	文字列型。データ長は変数の宣言 n で指定。最大 254 文字まで指定可能。 例) tmp_str : STRING[10]

5.2.5 SYSC コマンド

SYSC コマンドは、ロボット制御装置のシステム設定画面の項目を設定します。本節に記載する設定が不要な場合、SSET コマンドを使用する必要はありません。

```
SYSC,item_number,item_name,value1,[value2,]!!!!
```

item_number	設定する項目の項目番号を指定します。使用できる項目番号については、システム設計ツール取扱説明書(B-83274JA)の 4.8.17 節「システム設定」を参照してください。
item_name	項目名を指定します。項目名の部分は単なる説明用のコメントであり、ファイルの読み込み時には無視されます。
value1	設定値を指定します。指定する文字列は、全て大文字で入力して下さい。有効/無効の場合、ENABLE または DISABLE で指定します。
value2	第 2 設定値を指定します。(必要な場合のみ)

5.2.6 SSET コマンド

SSET コマンドはシステム項目の設定を行います。本節に記載する設定が不要な場合、SSET コマンドを使用する必要はありません。

```
SSET,item_name,index,value1,[value2,]!!!!
```

item_name	設定する項目名を指定します。項目名には予め決められた名前を指定してください。使用できる項目名については、次節以降を参照してください。
Index	項目番号を指定します。
value1	設定値を指定します。
value2	第 2 設定値を指定します。(必要な場合のみ)

指定された設定項目に指定された値を設定します。

使用例) Ethernet のポート番号 1 のアドレスを 192.168.1.2 に設定します。

```
SSET,PORT_ADDR,1,”192.168.1.2”,!!!!
```

SSET コマンドで設定可能な項目は次節以降を参照してください。

⚠ 注意

SSET コマンドは V9.40P/18 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで SSET コマンドを使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/17 版以前の場合は、プラグインソフトをインストールする前に V9.40P/18 版以降の最新版にアップデートしてください。V9.40P/17 版以前のロボット制御装置ソフトウェアでプラグインソフトをインストールすると、SDTL-084(データタイプに誤りがあります)アラームが発生してインストールに失敗します。」、と明記してください。

5.2.6.1 ホスト通信(Ethernet)関連項目の設定

以下のホスト通信(Ethernet)関連項目が設定可能です。

1. PORT_ADDR: Ethernet ポートの IP アドレスを設定します。IP アドレスは二重引用符で囲んでください。
   構文
   ```
   SSET,PORT_ADDR,ポート番号,IP アドレス,!!!!
   ```
   使用例
   ```
   SSET,PORT_ADDR,1,”192.168.1.2”,!!!!
   ```

2. PORT_SNMASK: サブネットマスクを設定します。サブネットマスク値は二重引用符で囲んでください。
   構文
   ```
   SSET,PORT_SNMASK,ポート番号,サブネットマスク値,!!!!
   ```
   使用例
   ```
   SSET,PORT_SNMASK,1,”255.255.0.0”,!!!!
   ```

3. PORT_RTADDR: ルータの IP アドレスを設定します。ポート番号には 1 を指定してください。また、IP アドレスは二重引用符で囲んでください。
   構文
   ```
   SSET,PORT_RTADDR,(使用しません),IP アドレス,!!!!
   ```
   使用例
   ```
   SSET,PORT_RTADDR,1,"192.168.1.1",!!!!
   ```

4. PORT_HOSTNAME: ロボット名(ホスト名)を設定します。ポート番号には 1 を指定してください。
   構文
   ```
   SSET,PORT_HOSTNAME,(使用しません),ロボット名,!!!!
   ```
   使用例
   ```
   SSET,PORT_HOSTNAME,1,ROBOT,!!!!
   ```

5. PORT_DHCPENB: DHCP の有効・無効を設定します。
   構文
   ```
   SSET,PORT_DHCPENB,ポート番号,設定値,!!!!
   ```
   使用例
   ```
   SSET,PORT_DHCPENB,1,TRUE,!!!!
   ```

5.2.6.2 Ethernet IP 関連項目の設定

以下の Ethernet IP 関連項目が設定可能です。

1. EIP_ADAPTER: 指定されたスロットをアダプタとして設定します。EIP_ADAPTER はそのスロットの全項目を初期値に戻しますので、他の項目を設定する前に実行してください。
   構文
   ```
   SSET,EIP_ADAPTER,スロット番号,!!!!
   ```
   使用例
   ```
   SSET,EIP_ADAPTER,3,!!!!
   ```

2. EIP_SCANNER: 指定されたスロットをスキャナとして設定します。EIP_SCANNER はそのスロットの全項目を初期値に戻しますので、他の項目を設定する前に実行してください。
   構文
   ```
   SSET,EIP_SCANNER,スロット番号,!!!!
   ```
   使用例
   ```
   SSET,EIP_SCANNER,3,!!!!
   ```

3. EIP_DESC: 指定されたスロットのコメントを設定します。
   構文
   ```
   SSET,EIP_DESC,Slot 番号,コメント文字列,!!!!
   ```
   使用例
   ```
   SSET,EIP_DESC,2,Connection2,!!!!
   ```

4. EIP_ENABLE: 指定されたスロットの有効・無効を設定します。
   構文
   ```
   SSET,EIP_ENABLE,スロット番号,(TRUE または FALSE),!!!!
   ```
   使用例
   ```
   SSET,EIP_ENABLE,2,TRUE,!!!!
   ```

5. EIP_HOST: ホスト名/IP アドレスを設定します。(スキャナのみ)
   構文
   ```
   SSET,EIP_HOST,スロット番号,ip アドレス,!!!!
   ```
   使用例
   ```
   SSET,EIP_HOST,2,192.168.1.1,!!!!
   ```

6. EIP_VENDOR: ベンダ ID を設定します。(スキャナのみ)
   構文
   ```
   SSET,EIP_VENDOR,スロット番号,整数値,!!!!
   ```
   使用例
   ```
   SSET,EIP_VENDOR,2,0,!!!!
   ```

7. EIP_DEVTYPE: デバイスタイプを設定します。(スキャナのみ)
   構文
   ```
   SSET,EIP_DEVTYPE,スロット番号,整数値,!!!!
   ```
   使用例
   ```
   SSET,EIP_DEVTYPE,2,0,!!!!
   ```

8. EIP_PRDCOD: プロダクトコードを設定します。(スキャナのみ)
   構文
   ```
   SSET,EIP_PRDCOD,スロット番号,整数値,!!!!
   ```

使用例
```
SSET,EIP_PRDCOD,2,0,!!!!
```

9. EIP_INSIZE: 入力サイズを設定します。
構文
```
SSET,EIP_INSIZE,スロット番号,整数値,!!!!
```
使用例
```
SSET,EIP_INSIZE,1,32,!!!!
```

10. EIP_OUTSIZE: 出力サイズを設定します。
構文
```
SSET,EIP_OUTSIZE,スロット番号,整数値,!!!!
```
使用例
```
SSET,EIP_OUTSIZE,1,32,!!!!
```

11. EIP_ALMSEV: アラーム重度を設定します。有効な値は 0: STOP, 1: WARN, 2: PAUSE です。
構文
```
SSET,EIP_ALMSEV,スロット番号,整数値,!!!!
```
使用例
```
SSET,EIP_ALMSEV,1,0,!!!!
```

12. EIP_TORPI: 「ターゲット(T) => オリジネータ(O) RPI」を設定します。(スキャナのみ)
構文
```
SSET,EIP_TORPI,スロット番号,整数値,!!!!
```
使用例
```
SSET,EIP_TORPI,2,32,!!!!
```

13. EIP_OTRPI: 「オリジネータ(O) => ターゲット(T) RPI」を設定します。(スキャナのみ)
構文
```
SSET,EIP_OTRPI,スロット番号,整数値,!!!!
```
使用例
```
SSET,EIP_OTRPI,2,32,!!!!
```

14. EIP_INASSY: 「Assembly instance (入力)」を設定します。(スキャナのみ)
構文
```
SSET,EIP_INASSY,スロット番号,整数値,!!!!
```
使用例
```
SSET,EIP_INASSY,2,200,!!!!
```

15. EIP_OUTASSY: 「Assembly instance (出力)」を設定します。(スキャナのみ)
構文
```
SSET,EIP_OUTASSY,スロット番号,整数値,!!!!
```
使用例
```
SSET,EIP_OUTASSY,2,201,!!!!
```

16. EIP_CONF: 「Configuration instance」を設定します。(スキャナのみ)
構文
```
SSET,EIP_CONF,スロット番号,整数値,!!!!
```
使用例
```
SSET,EIP_CONF,2,0,!!!!
```

17. EIP_DATATYP: 「I/O データタイプ」を設定します(スキャナのみ)。有効な値は TRUE または”8”(8-BIT)もしくは FALSE または”16”(16-BIT)です。
構文
```
SSET,EIP_DATATYP,スロット番号,(TRUE または FALSE),!!!!
```
または
```
SSET,EIP_DATATYP,スロット番号,(8 または 16),!!!!
```
使用例

```
SSET,EIP_DATATYP,2,TRUE,!!!!
True ならば 8-bit
False ならば 16-bit
または
SSET,EIP_DATATYP,2,8,!!!!
8: 8-bit
16: 16-bit
```

18. EIP_TMOMUL: 「タイムアウトマルチプライヤ」を設定します。(スキャナのみ)
構文
```
SSET,EIP_TMOMUL,スロット番号,整数値,!!!!
または
SSET,EIP_TMOMUL,スロット番号,DEFAULT,!!!!
```
使用例
```
SSET,EIP_TMOMUL,2,3,!!!!
または
SSET,EIP_TMOMUL,2,DEFAULT,!!!!
```

以下の値を指定可能です。

タイムアウトマルチプライヤ値
4
8
16
32
64
128
256
512
DEFAULT

19. EIP_RECONN: 「リコネクト」を設定します。(スキャナのみ)
構文
```
SSET,EIP_RECONN,スロット番号,(TRUE または FALSE),!!!!
```
使用例
```
SSET,EIP_RECONN,1,FALSE,!!!!
```

20. EIP_REVISION: 「メジャーリビジョン、マイナーリビジョン」を設定します。(スキャナのみ)
構文
```
SSET,EIP_REVISION,スロット番号,メジャー,マイナー,!!!!
```
使用例
```
SSET,EIP_REVISION,1,1,0,!!!!
```
この例では、メジャーリビジョンを 1 に、マイナーリビジョンを 0 に設定します。

21. EIP_QCONN: 「クイックコネクト」を設定します。(スキャナのみ)
構文
```
SSET,EIP_QCONN,スロット番号,(TRUE または FALSE),!!!!
```
使用例
```
SSET,EIP_QCONN,2,FALSE,!!!!
```

22. EIP_TRPORT: 「トランスポートタイプ」を設定します(スキャナのみ)。指定値が「FALSE」ならばユニキャスト、「TRUE」ならばマルチキャストとなります。
構文
```
SSET,EIP_TRPORT,スロット番号,(TRUE または FALSE),!!!!
```
使用例
```
SSET,EIP_TRPORT,2,FALSE,!!!!
```

23. EIP_FORMAT: 「O=>T フォーマット」と「T=>O フォーマット」を設定します。(スキャナのみ)

構文

```
SSET,EIP_FORMAT,Slot 番号,OT フォーマット(整数値),TO フォーマット(整数値),!!!!
```

使用例

```
SSET,EIP_FORMAT,1,3,0,!!!!
```

「O->T フォーマット」に指定できる整数値は以下の通りです。

0: Run/Idle Header
1: Zero-Length
2: Modeless
3: Heartbeat

「T->O フォーマット」に指定できる整数値は以下の通りです。

0: Run/Idle Header
1: Zero-Length
2: Modeless
3: Heartbeat

5.2.6.3 負荷切替距離、外力制限、無効信号の設定 (CR, CRX シリーズのみ)

SSET コマンドで負荷切替距離、外力制限、無効信号を設定する方法を以下に説明します。

⚠ 注意

- **SSET コマンドによる負荷切替距離、外力制限、無効信号の設定は V9.40P/37 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトの Autocfg.csv でこれらの設定を行う場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/36 版以前の場合は、プラグインソフトをインストールする前に V9.40P/37 版以降の最新版にアップデートしてください。」、と明記してください。**
- **本コマンドは CR シリーズ、CRX シリーズの協働ロボットのみ使用可能です。**

1. CR_PLD_CHNG_X: 負荷切替距離の X 方向制限値を設定します。+/-制限値ともに 0.0～50.0 の正の値を設定可能です。
   構文
   ```
   SSET,CR_PLD_CHNG_X,1,+制限値,-制限値,!!!!
   ```
   使用例
   ```
   SSET,CR_PLD_CHNG_X,1,1.0,2.0,!!!!
   ```

2. CR_PLD_CHNG_Y: 負荷切替距離の Y 方向制限値を設定します。+/-制限値ともに 0.0～50.0 の正の値を設定可能です。
   構文
   ```
   SSET,CR_PLD_CHNG_Y,1,+制限値,-制限値,!!!!
   ```
   使用例
   ```
   SSET,CR_PLD_CHNG_Y,1,1.0,2.0,!!!!
   ```

3. CR_PLD_CHNG_Z: 負荷切替距離の Z 方向制限値を設定します。+/-制限値ともに 0.0～50.0 の正の値を設定可能です。
   構文
   ```
   SSET,CR_PLD_CHNG_Z,1,+制限値,-制限値,!!!!
   ```
   使用例
   ```
   SSET,CR_PLD_CHNG_Z,1,1.0,2.0,!!!!
   ```

4. CR_PLD_CHNG_R: 負荷切替距離の姿勢回転制限値を設定します。制限値は 0.0～50.0 の正の値を設定可能です。
   構文
   ```
   SSET,CR_PLD_CHNG_R,1,制限値,!!!!
   ```
   使用例
   ```
   SSET,CR_PLD_CHNG_R,1,1.0,!!!!
   ```

5. CR_FORCE_LIM1: 協働ロボットの外力制限値 1 を設定します。
   構文
   ```
   SSET,CR_FORCE_LIM1,1,外力制限値,!!!!
   ```
   使用例
   ```
   SSET,CR_FORCE_LIM1,1,100.0,!!!!
   ```

注
- 外力制限値 2, 3, 4 は CR_FORCE_LIM2, 3, 4 コマンドで設定できます。
- 外力制限値を 0.0～150.0 の範囲で設定可能です。ただし、外力制限 1 は 0.0 に設定できません。

6. CR_DISABLE_IO1: 協働ロボットの無効信号 1 を設定します。
   構文
   ```
   SSET,CR_DISABLE_IO1,1,無効信号タイプ,無効信号番号,!!!!
   ```
   使用例
   ```
   SSET,CR_DISABLE_IO1,1,5,1,!!!!
   ```

注
- 無効信号 2, 3, 4 は CR_DISABLE_IO2, 3, 4 コマンドで設定できます。
- 協働ロボットの無効信号タイプと無効信号番号は 0～999 の範囲で設定可能です。不適切な安全 I/O が設定された場合、DCS パラメータへの適用はできません。
- 無効信号タイプはデュアル・チェック・セイフティ機能 取扱説明書(B-83184JA)の 9.1 節「安全 I/O」の安全 I/O の種類一覧表を参照して、信号タイプに対応するスロット番号を入力してください。

5.2.6.4 手動操作禁止信号の設定 (CRX シリーズのみ)

SSET コマンドで手動操作禁止信号を設定する方法を以下に説明します。

⚠ 注意
- SSET コマンドによる手動操作禁止信号の設定は V9.40P/53 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトの Autocfg.csv でこれらの設定を行う場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/52 版以前の場合は、プラグインソフトをインストールする前に V9.40P/53 版以降の最新版にアップデートしてください。」、と明記してください。
- 本コマンドは CRX シリーズの協働ロボットのみ使用可能です。

1. CR_MNL_AUTO_IO_IDX: AUTO モード手動操作の信号のインデックスを設定します。インデックスは 0 から 999 を設定可能です。
   構文
   ```
   SSET,CR_MNL_AUTO_IO_IDX,1,信号インデックス,!!!!
   ```
   使用例
   ```
   SSET,CR_MNL_AUTO_IO_IDX,1,10,!!!!
   ```

5.2.6.5 ダイレクトティーチの設定 (CRX シリーズのみ)

SSET コマンドでダイレクトティーチ(MGT: Manual Guided Teaching)の設定を行う方法を以下に説明します。

⚠ 注意
- SSET コマンドによるダイレクトティーチの設定は V9.40P/51 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトの Autocfg.csv でこれらの設定を行う場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/50 版以前の場合は、プラグインソフトをインストールする前に V9.40P/51 版以降の最新版にアップデートしてください。」、と明記してください。
- 本コマンドは CRX シリーズの協働ロボットのみ使用可能です。

両手ダイレクトティーチの設定

1. MGT_FREE_HAND_IO: 両手ダイレクトティーチの有効化信号を設定します。(信号タイプ: DI, RI, F)
   構文
   ```
   SSET,MGT_FREE_HAND_IO,1,信号タイプ,信号インデックス,!!!!
   ```

使用例

```
SSET,MGT_FREE_HAND_IO,1,RI,2,!!!!
```

設定入力信号の設定

1. MGT_IN_ACTIVE_TYPE: 設定入力信号の有効化方法を指定します。有効化方法には「DISABLE」、「ALWAYS」、「IO」を指定できます。

構文

```
SSET,MGT_IN_ACTIVE_TYPE,1,有効化方法,!!!!
```

使用例

```
SSET,MGT_IN_ACTIVE_TYPE,1,IO,!!!!
```

2. MGT_IN_ACTIVE_IO: 有効化方法が IO の場合の有効化信号を指定します。(信号タイプ: DI, RI, F)

構文

```
SSET,MGT_IN_ACTIVE_IO,1,信号タイプ,信号インデックス,!!!!
```

使用例

```
SSET,MGT_IN_ACTIVE_IO,1,DI,4,!!!!
```

3. MGT_IN_MOTION_TYPE: 動作形式を選択する信号を指定します。(信号タイプ: DI, RI, F, REG, GI)

構文

```
SSET,MGT_IN_MOTION_TYPE,1,信号タイプ,信号インデックス,!!!!
```

使用例

```
SSET,MGT_IN_MOTION_TYPE,1,REG,5,!!!!
```

4. MGT_IN_WEIGHT_FREE: 動作形式がフリーの時の操作の重さを選択する信号を指定します。(信号タイプ: REG, GI)

構文

```
SSET,MGT_IN_WEIGHT_FREE,1,信号タイプ,信号インデックス,!!!!
```

使用例

```
SSET,MGT_IN_WEIGHT_FREE,1,REG,5,!!!!
```

注

FREE を TRANS, ROT, CSTM に入れ替えるとそれぞれ動作形式が平行移動、回転、方向指定の時の操作の重さを選択する信号を指定できます。

5. MGT_IN_CSTM_FRM: 方向指定の座標系を選択する信号を指定します。(信号タイプ: DI, RI, F, REG, GI)

構文

```
SSET,MGT_IN_CSTM_FRM,1,信号タイプ,信号インデックス,!!!!
```

使用例

```
SSET,MGT_IN_CSTM_FRM,1,REG,5,!!!!
```

6. MGT_IN_ CSTM_X: 方向指定で X 軸方向に動作可能かを選択する信号を指定します。(信号タイプ: DI, RI, F)

構文

```
SSET,MGT_IN_CSTM_X,1,信号タイプ,信号インデックス,!!!!
```

使用例

```
SSET,MGT_IN_CSTM_X,1,DI,4,!!!!
```

注

X を Y, Z, RX, RY, RZ に入れ替えるとそれぞれ Y, Z 軸方向と X, Y, Z 軸周り回転方向に動作可能かを選択する信号を指定できます。

状態出力信号の設定

1. MGT_OUT_ACTIVE: ダイレクトティーチ中を示す信号を指定します。(信号タイプ: DO, RO, F)

構文

```
SSET,MGT_OUT_ACTIVE,1,信号タイプ,信号インデックス,!!!!
```

使用例

```
SSET,MGT_OUT_ACTIVE,1,DO,3,!!!!
```

2. MGT_OUT_MOTION_TYPE: 動作形式を示す信号を指定します。(信号タイプ: GO, REG)
   構文
   ```
   SSET,MGT_OUT_MOTION_TYPE,1,信号タイプ,信号インデックス,!!!!
   ```
   使用例
   ```
   SSET,MGT_OUT_MOTION_TYPE,1,REG,3,!!!!
   ```

3. MGT_OUT_FREE: 動作形式がフリーであることを示す信号を指定します。(信号タイプ: DO, RO, F)
   構文
   ```
   SSET,MGT_OUT_FREE,1,信号タイプ,信号インデックス,!!!!
   ```
   使用例
   ```
   SSET,MGT_OUT_FREE,1,DO,3,!!!!
   ```

注
FREE を TRANS, ROT, CSTM に入れ替えるとそれぞれ動作形式が平行移動、回転、方向指定であることを示す信号を指定できます。

4. MGT_OUT_WEIGHT: 現在の操作の重さを出力する信号を指定します。(信号タイプ: GO, REG)
   構文
   ```
   SSET,MGT_OUT_WEIGHT,1,信号タイプ,信号インデックス,!!!!
   ```
   使用例
   ```
   SSET,MGT_OUT_WEIGHT,1,REG,3,!!!!
   ```

5. MGT_OUT_WEIGHT_FREE: フリーの操作の重さを出力する信号を指定します。(信号タイプ: GO, REG)
   構文
   ```
   SSET,MGT_OUT_WEIGHT_FREE,1,信号タイプ,信号インデックス,!!!!
   ```
   使用例
   ```
   SSET,MGT_OUT_WEIGHT_FREE,1,REG,3,!!!!
   ```

注
FREE を TRANS, ROT, CSTM に入れ替えるとそれぞれ平行移動、回転、方向指定の操作の重さを出力する信号を指定できます。

6. MGT_OUT_CSTM_FRM: 方向指定の座標系を出力する信号を指定します。(信号タイプ: GO, REG)
   構文
   ```
   SSET,MGT_OUT_CSTM_FRM,1,信号タイプ,信号インデックス,!!!!
   ```
   使用例
   ```
   SSET,MGT_OUT_CSTM_FRM,1,REG,3,!!!!
   ```

7. MGT_OUT_CSTM_X: 方向指定で X 軸方向に動作可能かを示す信号を指定します。(信号タイプ: DO, RO, F)
   構文
   ```
   SSET,MGT_OUT_CSTM_X,1,信号タイプ,信号インデックス,!!!!
   ```
   使用例
   ```
   SSET,MGT_OUT_CSTM_X,1,DO,3,!!!!
   ```

注
X を Y, Z, RX, RY, RZ に入れ替えるとそれぞれ Y, Z 軸方向と X, Y, Z 軸周り回転方向に動作可能かを出力する信号を指定できます。

TP からの起動無効化の設定

1. MGT_DSBL_TP_IO: ダイレクトティーチの TP からの起動無効化信号を設定します。(信号タイプ: DI, RI, F)
   構文
   ```
   SSET, MGT_DSBL_TP_IO,1,信号タイプ,信号インデックス,!!!!
   ```
   使用例
   ```
   SSET, MGT_DSBL_TP_IO,1,RI,2,!!!!
   ```

5.2.6.6 ツール I/F 通信設定

ツール I/F 通信機能を使用する機器の通信設定は、11 章「ツール I/F 通信機能」を参照してください。
ツール I/F 通信機能を使用する機器を 2 台以上使用する場合の 2 台目以降の通信設定は、12.5 節「プラグインソフト開発者向け情報」を参照してください。

5.2.7 PLGMTOP コマンド

PLGMTOP コマンドを指定すると、メニューの大項目「プラグイン」を最上部に移動します。PLGMTOP コマンドを含むプラグインソフトをアンインストールすると、「プラグイン」メニューは元の位置に戻ります。

```
PLGMTOP,!!!!
```

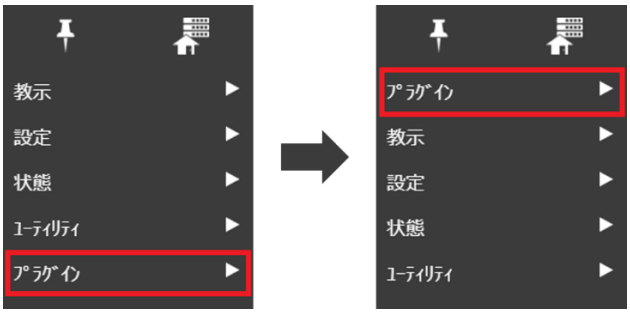


図 5.2.7 (a) プラグインメニューを移動

> ⚠ **注意**
> PLGMTOP コマンドは V9.40P/60 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで PLGMTOP コマンドを使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/59 版以前の場合は、プラグインソフトをインストールする前に V9.40P/60 版以降の最新版にアップデートしてください。V9.40P/59 版以前のロボット制御装置ソフトウェアでプラグインソフトをインストールすると、SDTL-084(データタイプに誤りがあります)アラームが発生してインストールに失敗します。」、と明記してください。

5.2.8 ツール座標系設定

ツール座標系はツール先端点(TCP)の位置とツールの姿勢を定義する直交座標系です。ツール座標系は通常 TCP を原点として、ツール方向を Z 軸に取ります。ツール座標系が未定義の時は、メカニカルインタフェース座標系によって代用されます。ツール座標系は、ツール先端点(TCP)の位置(x, y, z)と、ツールの姿勢(w, p, r)によって構成されます。ツール先端点(TCP)の位置は、メカニカルインタフェース座標系に対するツール先端点の座標値(x, y, z)[mm]で定義します。ツールの姿勢は、メカニカルインタフェース座標系の X, Y, Z 軸回りの回転角(w, p, r)[deg]で定義します。ツール先端点は、位置データの位置を教示するために使われます。ツール姿勢は、ツールの姿勢制御を行うために必要となります。使用するツール座標系番号は$MNUTOOLNUM[group]で設定します。

```
TFRM,GPn,Tool frame number,Comment,!!!!
,x,y,z,!!!!
,w,p,r,!!!!
```

GPn	グループ番号を指定します。最初の 2 文字は GP とし、次の 1 文字でグループ番号を指定します。GP1(グループ 1)を指定してください。
Tool frame number	ツール座標系番号を指定します。ツール座標系番号 1 番に設定することを推奨します。
Comment	ツール座標系にコメントを指定します。半角英数字で最大 20 文字です。

x,y,z,w,p,r	ツール座標系の(x,y,z,w,p,r)を指定します。小数点以下 3 桁で出力します。

設定例

```
TFRM,GP1,1,Gipper TCP,!!!!
,10,20,30,!!!!
,40,50,60,!!!!
DSET,*system*,$MNUTOOLNUM[1],INTEGER,1,!!!!
```

5.3 アンインストール処理の概要

本節では、アンインストール処理時の、Autocfg.csv ファイルで定義されたコマンドの振る舞いを説明します。

アンインストール処理では、以下のコマンドで定義・ロードされたプラグインソフトやプログラムを削除します。

- PLGINFO
- PLGID
- DESCFILE (指定されたファイルを削除します)
- LOAD (指定されたプログラムを削除します)

アンインストール処理では、以下のコマンドは 2 番目に指定された引数を実行します。

- RUNCM
- RUNPROG/RUNPRGWT

アンインストール処理では、以下のコマンドでは指定されたコメントを消去しますが、実際の設定やエントリは削除されず、そのまま残ります。

- レジスタ関連コマンド (REG, PREG, SREG)
- I/O 関連コマンド (DI, DO, AI, AO, GI, GO, etc.)
- 座標系コマンド (UFRM, JFRM, TFRM)
- 基準点設定 (REF)
- 負荷荷重設定 (PLD)
- スペースチェック設定 (RSP)
- プログラムタイマ (PTIM)

アンインストール処理では、以下のコマンドでは指定されたコメントと、実際の設定やエントリを消去します。

- ユーザアラーム (UALM)
- マクロ (MCR)

アンインストール処理では、以下のコマンドでの設定は消去されず、そのまま残ります。

- DI/DO の定義 (DIA, DOA, GIA, GOA, AIA, AOA)
- プログラムコメント (PRG)
- 位置コメント (POS)
- アームロード (ALD)
- 自動バックアップ (ATBK)
- システム設定 (SYSC)
- バッテリアラーム (BLAL)
- オーバライド選択 (OVSL)
- ワーニング履歴削除 (NOHIS)
- TCP/IP (TCPIP, TCPIP_HL)
- RSR, PNS
- iPendant の自動消灯 (BLNK)
- FAULT 信号を ON しないアラーム (ER_NOALM)
- エラーテーブル (ER_TBL)

6 プラグインソフト開発概要

エンドユーザはサードパーティ各社が開発したプラグインソフトを弊社ロボット制御装置にインストールすることで、周辺機器の設定変更や、機器の状態を表示できるようになります。

プラグインソフト開発 SDK(Software Development Kit)には、iHMI コンポーネントを使用したプラグイン画面、および、プラグイン命令の実装サンプル(FANUC_SmplGrp)を同梱しています。この実装サンプルでは、ロボット周辺機器を制御するためのデジタル入出力信号のインデックス番号を設定する画面を例として、KAREL 変数を介してプラグイン命令プログラム(TP プログラム)がプラグイン画面で設定された設定値を参照します。

本章では、プラグインソフト開発 SDK に同梱される実装サンプルに沿って、プラグインソフト開発の概要を説明します。

6.1 プラグイン画面で参照する KAREL 変数

プラグインソフト開発 SDK に同梱される実装サンプルのプラグイン画面では、デジタル入出力信号のインデックス番号を設定します。設定されたデータは不揮発性(電源再投入後に値を保持するメモリ)の KAREL 変数に格納します。一方、プラグイン命令プログラム(TP プログラム)は、KAREL 変数を参照してデジタル入出力信号のインデックス番号を取得し、デジタル入出力信号の状態を読みます。プラグイン命令プログラム(TP プログラム)については後の章で説明します。

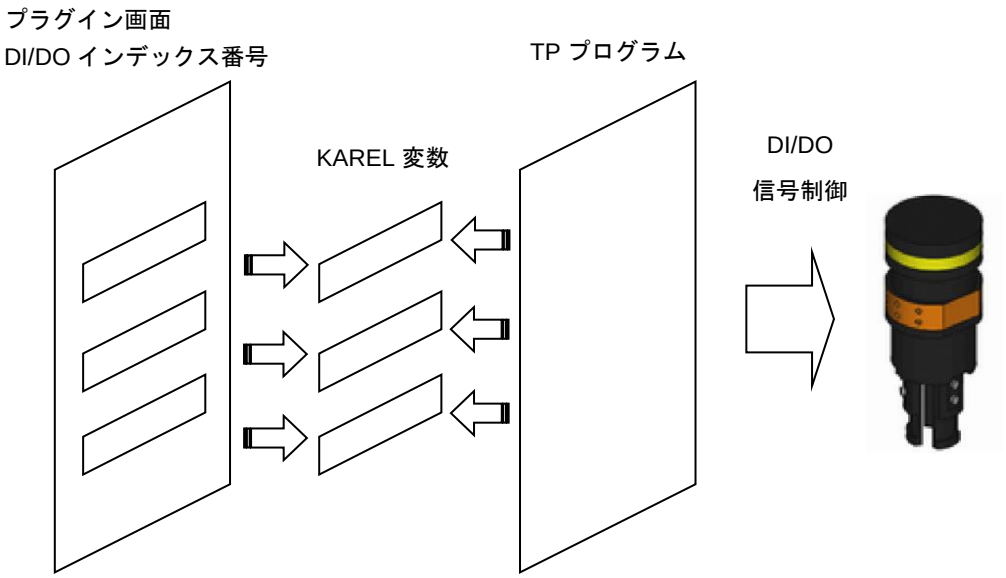


図 6.1 (a) KAREL 変数を介してプラグイン画面の設定を TP プログラムから参照する方法

実装サンプルのプラグイン画面で設定される項目、および、その設定値を格納する KAREL 変数を表 6.1 (a) に示します。

表 6.1 (a) 設定項目および KAREL 変数定義

設定項目	[KAREL プログラム名] KAREL 変数名	データ型
接続インタフェースの選択	[IPL_FANUC_SmplGrp_Cfg] SelSig	INTEGER
グリッパの開指令信号 (DO)	[IPL_FANUC_SmplGrp_Cfg] OpenDo	INTEGER
グリッパの閉指令信号 (DO)	[IPL_FANUC_SmplGrp_Cfg] ClosDo	INTEGER
グリッパの状態入力信号 (DI)	[IPL_FANUC_SmplGrp_Cfg] StatDi	INTEGER

KAREL 変数を定義するための KAREL プログラムの実装例は、下記ファイルを参照してください。

- ¥PluginSDK¥FANUC_SmplGrp¥kl_ls¥IPL_FANUC_SmplGrp_Cfg.kl (KAREL 変数定義ファイル)

注
- プラグイン ID を"(会社名)_(プラグイン名)"として定義し、TP プログラムと KAREL プログラムのファイル名を"IPL_(プラグイン ID)_(任意のプログラム名)"とする必要があります。例えば、プラグイン ID を"**FANUC_SmplGrp**"と定義した場合、プログラム名は **IPL_FANUC_SmplGrp_*****.TP, **IPL_FANUC_SmplGrp_*****.PC とする必要があります。ここで、"***"は任意のプログラム名を意味します。
- TP プログラムのパラメータ命令でパラメータ名として入力できる文字列は最大 30 文字です。KAREL 変数を指定する場合、KAREL プログラム名と KAREL 変数名をパラメータ命令に指定する必要があります。この制約を踏まえてプラグイン ID、KAREL プログラム名、KAREL 変数名を決定してください。
  例）ﾚｼﾞ[10001]=$**[IPL_FANUC_SmplGrp_Cfg]OpenDo**
  123456789012345678901234567890

6.2 プラグイン画面の実装

プラグインソフトでは、各種周辺機器の設定や状態表示を行うプラグイン画面を開発できます。開発した画面はタブレット TP の Web ブラウザで表示され、画面開発は一般的な Web 知識(HTML, JavaScript, CSS)で進めることができます。また、弊社が提供する iHMI コンポーネントを使用すると、画面開発に必要となるコードをすべて HTML, JavaScript, CSS で作成する必要がなく、画面開発をよりスムーズに進められます。画面開発者は、iHMI コンポーネントの独自タグ<ihmi type="部品名">を埋め込んだベースファイル(*.ishtm ファイル)を作成してビルドすることで、*.stm(*.htm)ファイルを生成します。

プラグインソフト開発用 SDK に、iHMI コンポーネントを使用したプラグイン画面、および、プラグイン命令の詳細画面の実装例(FANUC_SmplGrp)を同梱しています。これらの実装例では、ロボット制御装置にある KAREL 変数を読み書きする JavaScript API を使用しています。以下に示すベースファイルを参照してください。

- ¥PluginSDK¥FANUC_SmplGrp¥src¥Sample_Config.ishtm (プラグイン画面ベースファイル)
- ¥PluginSDK¥FANUC_SmplGrp¥src¥Sample_TP_detail.ishtm (プラグイン命令の詳細画面ベースファイル)

プラグイン画面の開発手順やビルドツールの詳細は 7 章「プラグイン画面作成」、ロボット制御装置と通信する JavaScript API については 7.5 節「ロボット制御装置とのインタフェース」を参照してください。

注
iHMI コンポーネントを含むベースファイル(*.ishtm ファイル)のビルド後に生成される*.stm ファイル(*.htm ファイル)のファイル名は任意です。使用できる文字列は英数字およびアンダーバー”_”で、拡張子を含めて 42 文字以内です。

6.3 画面データ辞書ファイルの実装

ロボット制御装置の言語設定に合わせて画面の表示言語を切り替えるには、辞書ファイル(*.utx)の作成が必要です。プラグイン画面では、表 6.3 (a) に示す言語の辞書ファイルが作成可能です。各言語の辞書ファイルとその各辞書要素を参照するベースファイル(*.ishtm)を作成してビルドすることで、各言語の接尾辞が付加された*.stm(*.htm)ファイルが生成されます。表示言語の切り替えについては、7.1.3 節「辞書ファイル」を参照してください。

辞書ファイルを作成する場合、標準言語として英語辞書(***eg**.utx)を必ず作成してください。他の言語は任意です。必要に応じて各国語対応の辞書ファイルを追加してください。ロボット制御装置の言語設定に応じて表示される言語を切り替えるためには、表 6.3 (a) に示す接尾辞を使用してください。

ファイル名： “任意のファイル名”＋**.utx (ただし**は国名を表す 2 文字を意味します)

表 6.3 (a) 言語設定とファイル名の接尾辞(例)

言語設定	ファイル名の接尾辞	ファイル名	ファイル内文字コード
英語	eg	“任意のファイル名”＋**eg**.utx	iso-8859-1
日本語	kn	“任意のファイル名”＋**kn**.utx	Shift_JIS

中国語	ch	“任意のファイル名”＋**ch**.utx	GB2312
台湾語	tw	“任意のファイル名”＋**tw**.utx	chinese-big5
ドイツ語	gr	“任意のファイル名”＋**gr**.utx	iso-8859-1
フランス語	fr	“任意のファイル名”＋**fr**.utx	iso-8859-1
イタリア語	it	“任意のファイル名”＋**it**.utx	iso-8859-1
スペイン語	sp	“任意のファイル名”＋**sp**.utx	iso-8859-1

辞書ファイル(*.utx)の実装例は、プラグインソフト開発 SDK の以下のファイルを参照してください。英語以外の言語の辞書ファイルは任意です。

- ¥PluginSDK¥FANUC_SmplGrp¥src¥Sample_dicteg.utx (英語辞書ファイル)
- ¥PluginSDK¥FANUC_SmplGrp¥src¥Sample_dictkn.utx (日本語辞書ファイル)
- ¥PluginSDK¥FANUC_SmplGrp¥src¥Sample_dictch.utx (中国語辞書ファイル)

6.4 プラグイン画面の実装例

以下にプラグイン画面の実装例と参照する KAREL 変数値を示します。この例では、画面上部でグリッパの接続インタフェースを選択し、ロボット制御装置インタフェース(DI/DO)を選択した場合に画面中部でグリッパ入出力信号のインデックス番号を設定します。また、グリッパの開閉確認を画面下部のボタンで行います。



図 6.4 (a) プラグイン画面と参照する KAREL 変数

7 プラグイン画面作成

本章では、タブレット TP の画面を作成する方法について説明します。タブレット TP では Web ブラウザ上に画面が表示されます。そのため、画面開発は一般的な Web 知識(HTML, JavaScript, CSS)で進めることができます。

iHMI コンポーネントを使用すると、画面開発に必要となるコードをすべて HTML, JavaScript, CSS で作成する必要がなく、iHMI デザインの画面開発をよりスムーズに進められます。ただし、画面開発における iHMI コンポーネントの使用は必須ではありません。

7.1 基本情報

タブレット TP の画面を作成する際に必要な情報について説明します。

7.1.1 対象 Web ブラウザ

タブレット TP では、Web ブラウザ上に画面が表示されます。対象ブラウザは以下です。

- Chrome (Windows タブレット)
- Chrome (iOS)
- Chrome (Android)
- WebView (Android アプリ内 WebView ブラウザ)

7.1.2 ファイルの種類

画面作成に必要となる代表的なファイルは以下です。

表 7.1.2 (a) ファイルの種類

ファイルカテゴリ	ファイルの種類 (拡張子)	ブラウザ
表示や動作	*.STM	Server Side Includes を利用した HTML Web ページです。
	*.ISHTM	STM のベースファイルです。ビルドを行うことで、STM ファイルに変換されます。
	*.HTM	Server Side Includes を含まない HTML Web ページです。
	*.IHTML	HTM のベースファイルです。ビルドを行うことで、HTM ファイルに変換されます。
	*.JS	コンテンツを動的に更新するスクリプトです。
	*.CSS	Web ページのスタイルを指定するファイルです。
画像	*.PNG	画像ファイルです。透過を生かした画像を使用したい場合に有効です。
	*.GIF	画像ファイルです。ボタン、アイコン、バナーのようなサイズが小さいグラフィック(純色の部分が多く、細かい部分の少ない画像)に適しています。
	*.JPG	画像ファイルです。細かい部分と色数の多い画像に適しています。
リソース	*.UTX	辞書ファイルです。ロボット制御装置の言語設定に応じて画面の表示言語を切り替えるために必要です。詳細は 7.3.2 節「辞書ファイルの作成」を参照してください。
KAREL	*.VR	KAREL 変数ファイルです。
	*.PC	KAREL プログラムの実行形式ファイルです。

7.1.3 辞書ファイル

ロボット制御装置の言語設定に応じて言語を切り替える場合には、ファイル名を特定の接尾辞を付けたものにする必要があります。

表 7.1.3 (a) 言語設定とファイル名の接尾辞

言語設定	ファイル名の接尾辞	ファイル名	ファイル内文字コード
英語	eg	“任意のファイル名”＋**eg**.utx	iso-8859-1
日本語	kn	“任意のファイル名”＋**kn**.utx	Shift_JIS
中国語	ch	“任意のファイル名”＋**ch**.utx	GB2312
台湾語	tw	“任意のファイル名”＋**tw**.utx	chinese-big5
ドイツ語	gr	“任意のファイル名”＋**gr**.utx	iso-8859-1
フランス語	fr	“任意のファイル名”＋**fr**.utx	iso-8859-1
イタリア語	it	“任意のファイル名”＋**it**.utx	iso-8859-1
スペイン語	sp	“任意のファイル名”＋**sp**.utx	iso-8859-1

表 7.1.3 (a) での「ファイル名の接尾辞」の命名規則に従うことで、ロボット制御装置の言語設定に応じたファイルが自動的に選択され表示されます。例えば、arc**eg**.stm と arc**kn**.stm のファイルが存在し、ロボット制御装置の言語が英語に設定されていた場合、arc**eg**.stm が選択されます。ただし、接尾辞がないファイル arc.stm が存在した場合は、言語設定にかかわらず arc.stm が選択されます。

注
- 画像イメージファイル(*.png, *.gif, *.jpg)、プラグイン画面データファイル(*.ishtm, *.ihtml, *.stm, *.htm, *.js, *.css)、辞書ファイル(*.utx)のファイル名は任意です。使用できる文字列は英数字およびアンダーバー”_”で、拡張子を含めて 42 文字以内です。
- プラグイン ID を”(会社名)_(プラグイン名)”として定義し、TP プログラムと KAREL プログラムのファイル名を”IPL_(プラグイン ID)_(任意のプログラム名)”とする必要があります。例えば、プラグイン ID を”**FANUC_SmplGrp**”と定義した場合、プログラム名は **IPL_FANUC_SmplGrp_*****.TP, **IPL_FANUC_SmplGrp_*****.PC とする必要があります。ここで、”***”は任意のプログラム名を意味します。
- リンクには接尾辞を含めないでください。
  (正) <link rel="stylesheet" type="text/css" href="arc.css"/>
  (誤) <link rel="stylesheet" type="text/css" href="arc**eg**.css"/>

7.2 ビルドツール

ビルドツール(build_ihmi.bat)による画面作成は、以下の手順で進めます。

1. 環境準備
ビルドツールを使用するために必要な環境を準備します。

2. ベースファイルの作成
ベースファイル(ishtm/ihtml)ファイル)を作成します。ベースファイルには、使用する iHMI コンポーネントを独自タグ<ihmi type="~">で埋め込みます。

3. ビルド
ビルドツールを実行して、ベースファイルに埋め込んだ独自タグを展開します。

注
ロボット制御装置へのロード方法および表示方法については、3 章「プラグインソフトのインストールと管理」を参照してください。

7.2.1 ビルド環境の準備

ビルドツールを使用するために、Windows 上で Perl が動作する環境を準備します。本章での説明は、以下のソフトウェアで動作確認しています。

StrawberryPerl
http://strawberryperl.com/

インストールを終えたら、コマンドプロンプト上に「perl -v」と入力して、perl の version が表示されることを確認します。

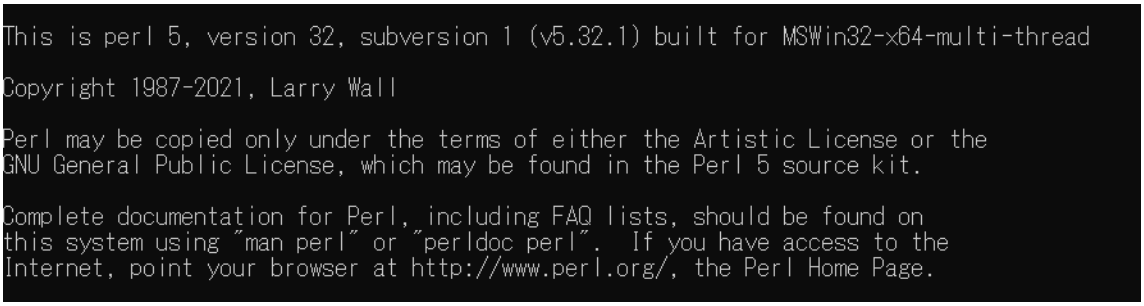


図 7.2.1 (a) perl -v 実行時

> **注**
> ビルドツールは、64 ビット版の Microsoft Windows 7 および Windows 10 が動作する PC で動作確認しています。

> **注**
> 上記画面が表示されない場合、環境変数 Path に以下の記述を追加してください。
> C:¥(インストールフォルダ)¥bin;
> C:¥(インストールフォルダ)¥site¥bin¥
> なお、「C:¥(インストールフォルダ)」部分の指定は Perl をインストールしたディレクトリに合わせてください。

7.2.2 ベースファイルの作成

以下に、ビルドツールのフォルダ構成を示します。画面開発者は、src フォルダ内に必要なファイルを作成します。

表 7.2.2 (a) フォルダ構成

ファイル名・フォルダ名	概要
ihmicomponent	iHMI コンポーネント関連フォルダです。
src	Web 画面に必要なファイルを格納するフォルダです。 iHMI コンポーネントを使った実装方法については、7.3 節「HMI コンポーネントの利用」を参照してください。
build_ihmi.bat	ビルド時に実行するビルドツールです。

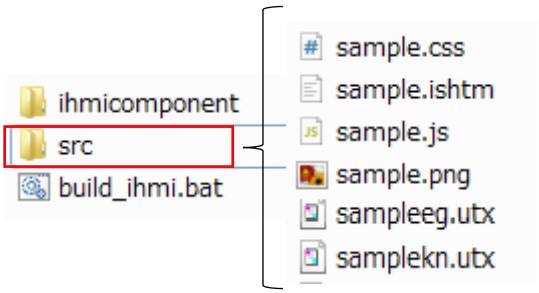


図 7.2.2 (a) フォルダ構成

注

- src フォルダ内には、第 1 階層までに任意の名称のフォルダを作成できます。ただし、src フォルダ内のファイルは、ビルド後にすべて同じフォルダに生成されます。このため src フォルダ内の css や js といった外部ファイルを ishtm(ihtml)にてインクルードする場合、カレントディレクトリでパスを指定してください。
- 手動でキャッシュクリアする場合を除き、タブレット TP はキャッシュを保持する仕様です。JavsScript や CSS ファイルを変更したプラグインソフトをインストールしたときに、キャッシュを使わず変更後の画面データを使用して表示したい場合は、プラグインソフトの JavsScript や CSS ファイルの呼び出し文にクエリパラメータを付与して、この値を変更してください。この値を変更することによって、現在キャッシュしているファイルとは別物として扱われます。ここでクエリパラメータは x=y の形であれば何でも構いません。
  例)
  <script type="text/javascript"
  src="/frh/plugin/FANUC_SMPLGRP/Sample_Config.js**?ver=010000**"></script>
  <link rel=stylesheet type="text/css"
  href="/frh/plugin/FANUC_SMPLGRP/Sample_Config.css**?ver=010000**" />

7.2.3 ビルド

以下の手順で、ベースファイル ishtm(ihtml)と辞書ファイルから、ロボット制御装置に組み込むための stm(htm)ファイルを生成します。

1. build_ihmi.bat をダブルクリックします。
2. output フォルダ直下に stm(htm)ファイルが生成されます。
3. 同時に output フォルダ内のファイルを圧縮した output.ipl ファイルも生成されます。
4. この ipl ファイルを使ってロボット制御装置にインストールします。

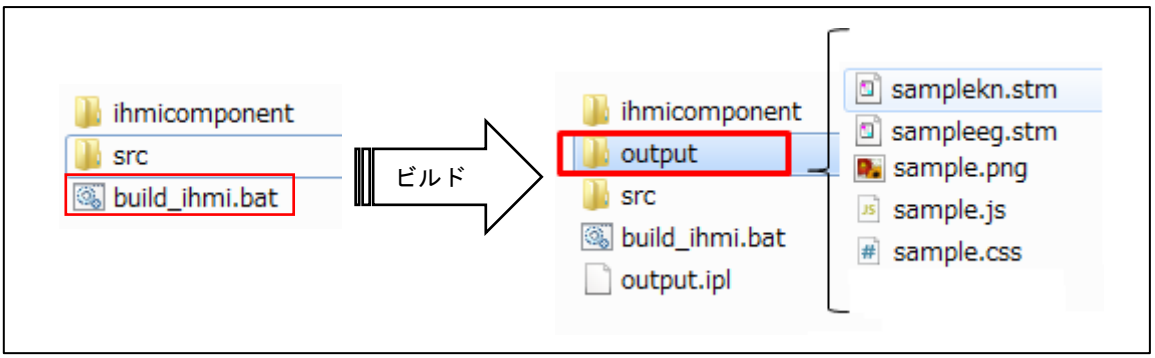


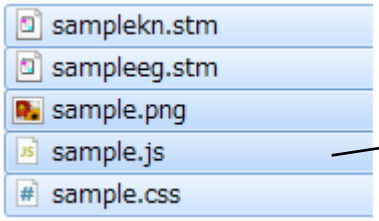
図 7.2.3 (a) ビルド

注

以下のような圧縮エラーが発生した場合は、output フォルダ内のすべてのファイルを手動で圧縮します。

```
convert ssl ihmi tag to html tag [sample.ishtm]
stm file created ( eg, kn, ch, ) from [sample.shtm].
*** ihmi build finished. ***
---COMPRESS ERROR--- Failed to create compressed file.
```

1 output フォルダ内のすべてのファイルを選択したのち、右クリックして圧縮(zip)を実行します。



ファイルを全選択した状態で zip 圧縮

2 拡張子を zip から ipl に変更します。

7.3 iHMI コンポーネントの利用

iHMI コンポーネントを使用することで、iHMI デザインの画面開発をよりスムーズに進めることができます。iHMI コンポーネントは、以下の 3 つで構成されています。

1. **HTML テンプレートエンジン**
   独自タグ<ihmi>で iHMI コンポーネントの部品を指定することで、標準の HTML に展開されます。

2. **CSS フレームワーク**
   iHMI コンポーネント側で CSS が用意されています。iHMI コンポーネントの部品を配置するためのスタイル定義は必要ですが、部品自体のスタイル定義は必要ありません。

3. **JavaScript ライブラリ**
   表示の更新や、ユーザ操作時の Callback 関数の登録などの JavaScript ライブラリが iHMI コンポーネントの部品ごとに提供されています。

iHMI コンポーネントの対応ブラウザは、以下のとおりです。

- Chrome (Windows 7 または Windows 10)
- Chrome (Android OS 標準 Chrome ブラウザ)
- Chrome (iOS)
- WebView (Android アプリ内 WebView ブラウザ)

7.3.1 STM/HTM ファイルの作成

タブレット TP 上に表示される画面は、複数のフレームで構成されています。このうち、画面開発者が iHMI コンポーネントを使って、カスタマイズできる部分は「コンテンツエリア」になります。画面開発者は、コンテンツエリアに表示する stm(htm)ファイルを作成する必要があります。



図 7.3.1 (a) 画面構成

画面開発者は、iHMI コンポーネントの独自タグ<ihmi type="部品名">を埋め込んだベースファイルを作成してビルドすることで、stm(htm)ファイルを生成できます。

ishtm(ihtml)ファイルは、使用する部品に関連する iHMI コンポーネントの css (例: ihmicomponentstyle.css)を読み込ませた上でビルドしてください。

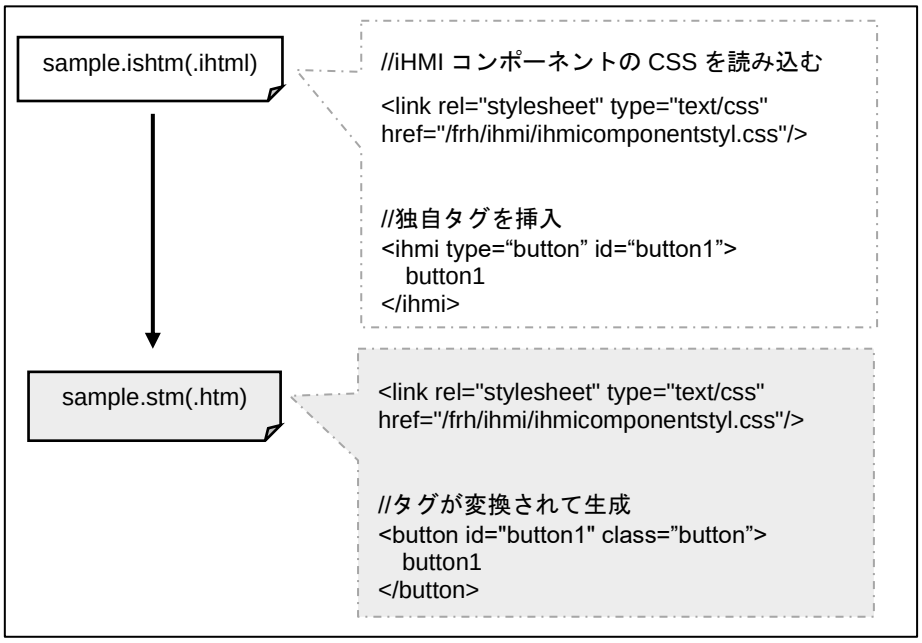


図 7.3.1 (b) button 部品を使った stm(htm)生成の流れ

また、7.3.2 節「辞書ファイルの作成」で作成方法を説明している辞書ファイルを組み込んでビルドすることで、言語別の stm(htm)ファイルを生成できます。

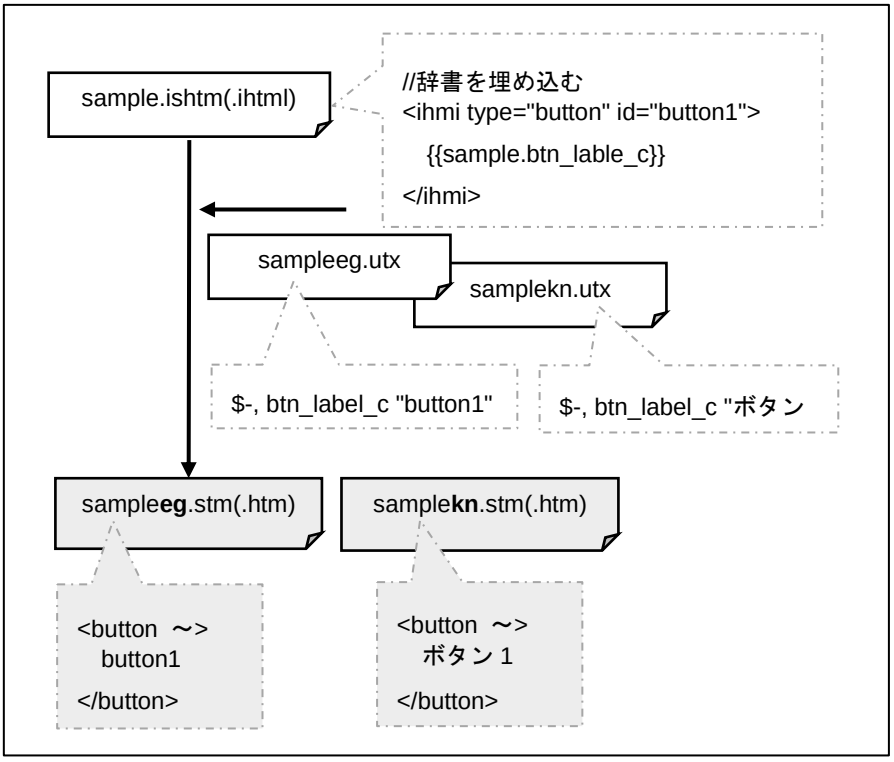


図 7.3.1 (c) 言語別 stm(htm)生成の流れ

7.3.2 辞書ファイルの作成

ロボット制御装置の言語設定に合わせて画面の表示言語を切り替える場合には、辞書ファイル(utx)の作成が必要です。表示言語の切り替えについては、7.1.3 節「辞書ファイル」を参照してください。

辞書ファイル内では、「$-, [辞書要素名] [翻訳テキスト]」の形式でボタン名などを定義します。図 7.3.1 (c)を参照してください。

```
sampleeg.utx
$-, btn_label_c 'button'
```

ishtm(ihtml)ファイルに以下の方法で辞書ファイル名と辞書要素名を指定します。

```
sample.ishtm(ihtml)
<ihmi type="button"~>{{sample.btn_label_c}}</ihmi>
```

注
- 言語ごとに文字コードが異なります。エディタで辞書ファイルを修正の際は注意してください。文字コードについては、7.1.3 節「辞書ファイル」を参照してください。
- 辞書項目の埋め込みは、JavaScript ファイルに対しても可能です。

7.3.3 CSS ファイルの作成

iHMI コンポーネントでは、以下の CSS ファイルが用意されています。ロボット制御装置上の格納先は/frh/ihmi/です。

表 7.3.3 (a) iHMI コンポーネント css

ファイル名	概要	関連部品
ihmicomponentstylkn.css ihmicomponentstyleg.css	色やサイズといったスタイル	全般
ihmicomponentfrm.css	画面の構成関連	textboxInteger

		textboxReal textboxString

<ihmi>タグで iHMI コンポーネントの部品を指定すると、上記の CSS ファイルに定義されている共通デザインのセレクタが、ビルド時に自動的に class 属性に埋め込まれます。

7.3.4 JavaScript ファイルの作成

動的にコンテンツを更新するために JavaScript を使用します。

iHMI コンポーネントは以下を提供します。

- デザインや振る舞いの実現
  ボタン押下時の選択色、フォーカス枠の付け外しといった、どの画面にも共通する振る舞い
- 部品を操作するためのメンバ関数を用意
  例：有効/無効の切り替え、ラベルや色の動的変更

ユーザ操作から画面更新までの基本的な流れは以下のとおりです。

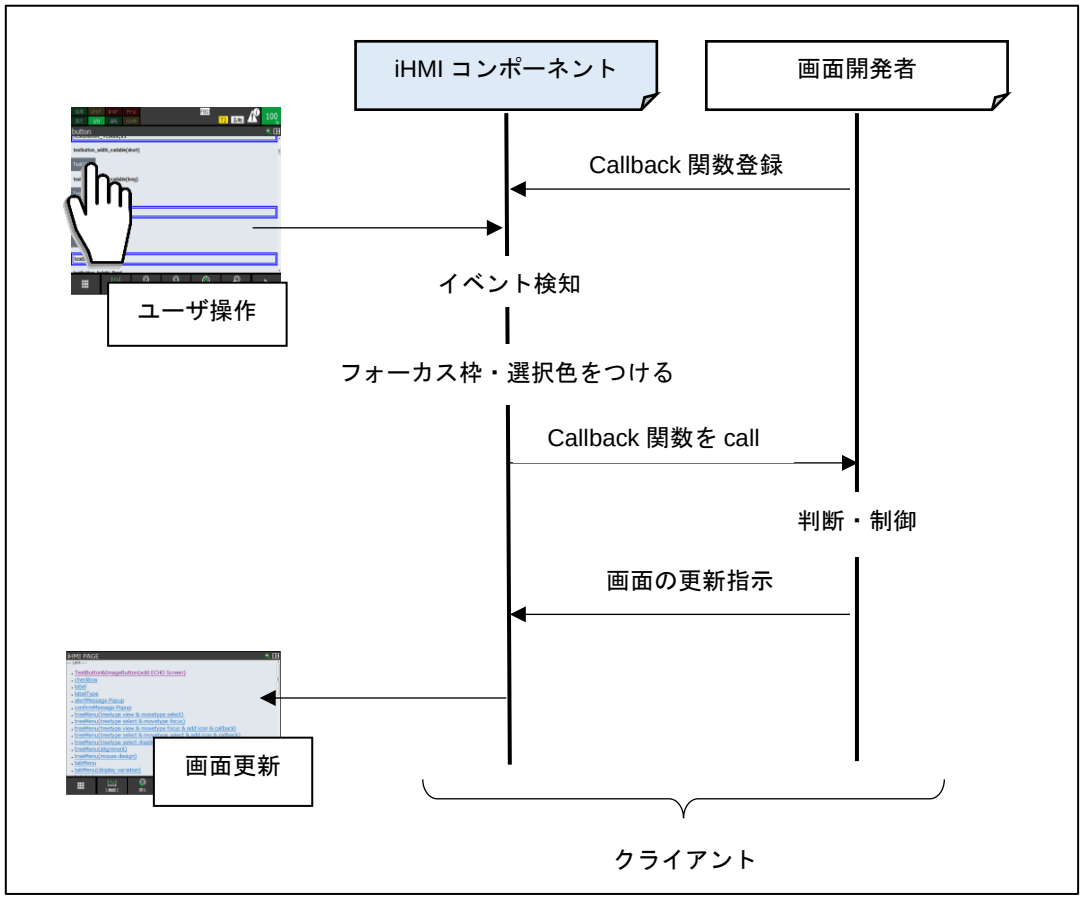


図 7.3.4 (a) 画面更新シーケンス

iHMI コンポーネントの部品が持つメンバ関数を使用する場合は、以下のように呼び出します。

{getElementById()で取得した DOM 要素}.{メンバ関数()}

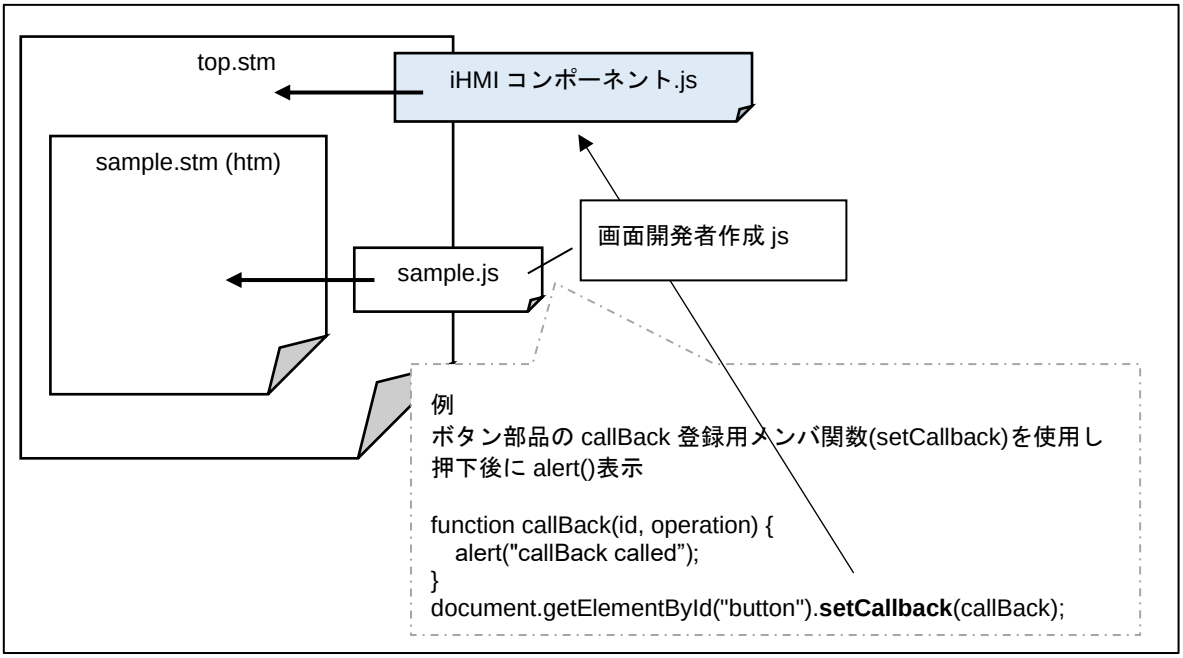


図 7.3.4 (b) 部品メンバ関数の呼び出し方

ビルドでの ishtm(ihtml)から stm(htm)ファイル変換時に、<ihmi>タグで埋め込んだ DOM 要素に、iHMI コンポーネントの部品操作関数をリンク付けするコードが埋め込まれます。その初期化処により、上記の呼び出し方が可能になります。メンバ関数の呼び出しは、HTML ロード後に行ってください。

また、Callback 関数を設定することで、ユーザ操作が発生したイベントを検知できるようになります。図 7.3.4 (a)ではユーザ操作イベントを検知して Callback 関数を呼び出し、画面の更新を指示しています。

7.4 iHMI コンポーネント部品

以下の iHMI コンポーネント部品が用意されています。

表 7.4 (a) 部品の種類

部品名	概要
button	ボタン
textboxInteger	Int 型テキストボックス
textboxReal	Real 型テキストボックス
textboxString	String 型テキストボックス
listSelect & listSelectOption	リスト選択。複数の項目から 1 つを選択する
select	ドロップダウンボタン

本章では、各部品での HTML テンプレートエンジンの設定、部品を操作するための JavaScript 関数について説明します。

注
iHMI コンポーネントを使用して作成した要素は、直接 DOM 操作せずにコンポーネントの I/F を使用してください。解像度の違いにより、表示サイズが異なる場合があります。

例) textbox 入力フォームに値を設定する場合
(HTML)
<ihmi type="textboxInteger"></ihmi>
(NG)
document.getElementById(textboxInteger").val = "123"; // 標準プロパティ
(OK)
document.getElementById(textboxInteger").refresh("123", ~); // iHMI コンポーネント I/F

7.4.1 button

コントローラ概要

デザイン
touch / mouse 
概要
ボタン
その他
テキストや画像を付加可能なボタン

HTML テンプレートエンジン

ihmi タグ type			
button			
引数			
〇：必須 △：オプション	**パラメータ**	**型**	**説明**
△	designpattern	String	デザインパターンの選択 • "touch"： タッチ操作用デザイン • "mouse"： マウス操作用デザイン

△	class	String	ボタン全体に付加する拡張クラス
			例)
			ボックスの縦・横サイズ、テキストのサイズ・位置、ボックスのデザインパターン
			
			拡張クラスに定義する際は、コンポーネントで定義されているデフォルトのサイズを上書きする必要があるため、以下のように詳細度を上げて定義する
			//拡張クラス AAA を定義
			.button.AAA {
			height: ***px;
			}

△	classtext	String	ボタン内のテキスト付加する拡張クラス
△	width	Integer	ボタンの横幅 • 引数 width の値を style 属性の width に反映する • 指定を省略した場合、ボタンの横幅は、設定したテキストの長さによって可変となる
△	height	Integer	ボタンの縦幅 引数 height の値を style 属性の height に反映する

△	image	String	背景画像 背景画像を表示するときに、絶対パスまたは相対パスで指定する
△	imageleft	String	image 画像 テキストの左側に image 画像を表示するときに、絶対パスまたは相対パスで指定する
△	タグ内に設定した値 <ihmi type="button">xxx</ihmi>	String	ボタン内テキスト
△	imageright	String	image 画像 テキストの右側に image 画像を表示するときに、絶対パスまたは相対パスで指定する
△	vertical	Boolean	画像とテキストを縦並びにするときのみ true を指定する
△	onclick	String	Callback 関数の登録 本パラメータを指定した場合、setCallback()で登録した関数は無効となる。

その他

- 背景画像を指定するとき、基本的に img 画像は指定しないものとする。
- spriteleft(spriteright)と imageleft(imageright)は同時には指定できない。
- 引数 image、imageleft、imageright を指定するときには、絶対パスまたは相対パスのどちらかで指定する。
- 引数 vertical を指定して画像とテキストを縦並びにするときには、画像のパス指定に imageleft を使用する。
- ボタン内には左右に 8px の padding が固定で設定される。
- <button>タグを使用しているため、内包する要素は基本的に左右中央寄せ、上下中央寄せになる。
- 画像のみのボタン(ラベルがないボタン)を使用するときは、拡張クラス引数 class に"image-button"を指定する。
- 画像とテキストが横並びのボタンを使用するときは、拡張クラス引数 class に"text-image-button"を指定することで、画像とテキストが上下中央寄せのボタンになる。
- 画像とテキストが縦並びとなるボタンや、画像とテキストが横並びで画像の位置を任意に指定するようなボタンでは、拡張クラス引数 class に"text-image-button"は指定しないこと。

JavaScript ライブラリ

refresh

概要

更新

関数名

refresh

引数

順番	型	説明
1 text	String	ボタン内のテキスト コンポーネントでは、本引数が innerHTML プロパティに設定され、HTML コードとして解釈される
2 color	String	ボタン内のテキスト色 値には 7.4.7 節のカラーテーブルに示すプロパティ名を指定する
3	--	null 固定
4	-	null 固定
5 disabled	Boolean	ボタンの無効(true) / 有効(false)の切り替え
6	-	null 固定

戻り値

変数名	型	説明
-	-	-

その他
refresh をしない項目の引数には null を指定する。ただし、disabled には null は指定できないため、true / false のいずれかを必ず指定する。

setCallback

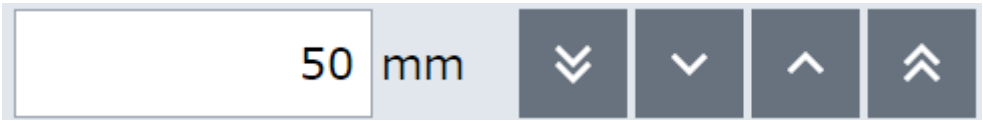
概要		
Callback 関数の登録		
関数名		
setCallback		
引数		
順番	型	説明
1 callback	Function	Callback 関数
引数 (callback)		
順番	型	説明
1 id	String	id 名
2 operation	String	"clicked"の文字列
戻り値		
型	説明	
-	-	
その他		
onclick 属性の指定時は、本関数で登録した Callback 関数は動作しない		

setFocus

概要		
ボタンにフォーカスをあてる		
関数名		
setFocus		
引数		
変数名	型	説明
-	-	-
戻り値		
型	説明	
-	-	
その他		
-		

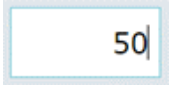
7.4.2 textboxInteger

コントローラ概要

デザイン
touch  mouse 
概要
Integer 型テキストボックス

その他
数値入力用。入力値の最小値および最大値のチェック、増減ボタンによる入力値の変更ができる。 • 入力後のバリデーションチェック 0～9 の半角数字、-(マイナス)以外が入力された場合は、「数値を入力して下さい」というエラーメッセージを表示する。また、最大値および最小値の範囲を超えていた場合は、「(入力値)は範囲外です。(最小値)-(最大値)」というエラーメッセージを表示する。textboxInteger を使用する場合は、refresh 関数で入力値の最小値および最大値を必ず指定してください。

HTML テンプレートエンジン

ihmi タグ type			
textboxInteger			
引数			
○：必須 △：オプション	**パラメータ**	**型**	**説明**
△	designpattern	String	デザインパターンの選択 • "touch"： タッチ操作用デザイン • "mouse"： マウス操作用デザイン
△	height	String	テキストボックスの高さを指定する • テキスト位置は上下中央寄せに固定される • 本値に上下 2px のボーダー幅は含まれない 例) height="30px"を指定すると、30px+(2px+2px)(ボーダー幅)の高さになる
△	width	String	テキストボックスの横幅を指定する 本値に上下 2px のボーダー幅と、左右の padding は含まれない 例) • width=”100px”を指定すると、デザインパターンが"touch"の場合、100px+2px+16px=118px の幅となる • "mouse"の場合、100px+2px+8px=110px の幅となる
△	unSelect	String	クリック時にテキストボックス内の文字全体の選択を行わないようにする • "true"： クリック時に文字全体を選択しない  • "true"以外または指定なし：クリック時に文字全体を選択する 
△	units	String	ボックス横に表示されるテキスト
△	buttons	String	増減ボタンの表示 • 「step」で増／減ボタンをそれぞれ 1 つずつ • 「step2」で増／減ボタンをそれぞれ 2 つずつ表示する
△	step1	Integer	• 増減ボタンが押下された場合、本値の分だけ入力値が増減する • 本引数の設定を省略した場合、増減幅は 1 になる
△	step2	Integer	• 増減ボタンが押下された場合、本値の分だけ入力値が増減する • buttons の値が「step」のときには、対応するボタンは表示されない • 本引数の設定を省略した場合、増減幅は step1 の 10 倍となる

△	errPopPos	Integer	バリデーションエラー時、textbox の Window から何階層上の Window にポップアップを表示するかを指定する
△	inputType	String	タブレット時のソフトウェアキーボードの type を指定する • "text" : 文字入力の一般的なキーボード(デフォルト値) • "number" : 数値入力用 • "tel" : 数値入力用。小数点やマイナス値のキー入力が可能
その他			
デフォルトでは、テキストボックス内のテキスト位置は右寄せ			

textboxInteger の場合に、inputType パラメータで指定するソフトウェアキーボードは以下の通りです。



“text” “number” “tel”

図 7.4.2 (a) textboxInteger の場合の inputType によるソフトウェアキーボード

注
textboxInteger を使用する場合、負の入力値を許容するか否かに関わらず、inputType=”number”を指定してください。

JavaScript ライブラリ

refresh

概要		
更新		
関数名		
refresh		
引数		
順番	**型**	**説明**
1 cur	String	テキストボックスの入力値
2 min	String	入力値チェック時の最小値
3 max	String	入力値チェック時の最大値
4 disabled	Boolean	無効(true) / 有効(false)の切り替え
戻り値		
型	**説明**	
-	-	
その他		
入力値チェック • cur が min 以下の場合、減ボタンが無効状態になる。 • cur が max 以上の場合、増ボタンが無効状態になる。 • cur が min 未満、または max 超過の場合、文字色が赤くなる。 • cur が数値でなかった場合、入力値が空になって背景色が赤くなり、増減ボタンが無効状態になる。		

setCallback

概要
Callback 関数の登録

関数名		
setCallback		
引数		
変数名	**型**	**説明**
callback	Function	Callback 関数
引数 (callback)		
順番	**型**	**説明**
1 id	String	id 名("部品に指定した id".textbox)
2 operation	String	"changed"の文字列
3 value	String	入力値
戻り値		
型	**説明**	
-	-	
その他		
-		

setFocus

概要		
テキストボックスにフォーカスをあてる		
関数名		
setFocus		
引数		
変数名	**型**	**説明**
-	-	-
戻り値		
型	**説明**	
-	-	
その他		
-		

getValue

概要		
テキストボックスの入力値を取得		
関数名		
getValue		
引数		
変数名	**型**	**説明**
-	-	-
戻り値		
型	**説明**	
-	-	
その他		
-		

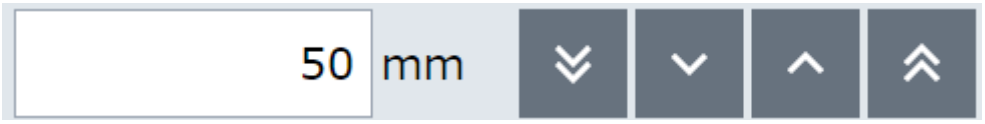
setCursorPos

概要
テキストボックス内カーソル位置の設定
関数名
setCursorPos

引数		
順番	**型**	**説明**
1 targetStart	Number	カーソルをセットする開始位置
2 targetEnd	Number	カーソルをセットする終了位置 範囲を指定しない場合は、開始位置と同じ値を設定する 終了位置には、開始位置以上の値を指定する
戻り値		
型	**説明**	
-	-	
その他		
-		

7.4.3 textboxReal

コントローラ概要

デザイン
touch  mouse 
概要
Real 型テキストボックス
その他
数値入力用。入力値の最小値および最大値のチェック、増減ボタンによる入力値の変更ができる。 入力後のバリデーションチェック • 0～9 の半角数字、ピリオド(.)、-(マイナス)が可能 • 上記以外の値が入力されていた場合は、「数値を入力して下さい」というエラーメッセージを表示する。最大値および最小値の範囲を超えていた場合は、「(入力値)は範囲外です。(最小値)-(最大値)」というエラーメッセージを表示する。

HTML テンプレートエンジン

ihmi タグ type			
textboxReal			
引数			
○：必須 △：オプション	**パラメータ**	**型**	**説明**
△	designpattern	String	デザインパターンの選択 • "touch"： タッチ操作用デザイン • "mouse"： マウス操作用デザイン
○	decimals	Integer	入力可能な小数点以下の桁数。
△	height	String	テキストボックスの高さを指定する • テキスト位置は上下中央寄せに固定される • 本値に上下 2px のボーダー幅は含まれない 例) height="30px"を指定すると、30px+(2px+2px)(ボーダー幅)の高さとなる。

△	width	String	テキストボックスの横幅を指定する 本値に上下 2px のボーダー幅と、左右の padding は含まれない 例) width=”100px”を指定すると、 • デザインパターンが"touch"の場合、100px+2px+16px=118px の幅となる • "mouse"の場合、100px+2px+8px=110px の幅となる
△	unSelect	String	クリック時にテキストボックス内の文字全体の選択を行わないようにする • "true"：クリック時に文字全体を選択しない  • "true"以外または指定なし：クリック時に文字全体を選択する 
△	units	String	ボックス横に表示されるテキスト
△	buttons	String	増減ボタンの表示 「step」または「scale」で増／減ボタンをそれぞれ 1 つずつ、「step2」または「scale2」で増／減ボタンをそれぞれ 2 つずつ表示する
△	step1	Integer	増減ボタンが押下された場合、本値の分だけ入力値が増減する 本引数の設定を省略した場合、増減幅は 1 になる
△	step2	Integer	• 増減ボタンが押下された場合に、本値の分だけ入力値が増減する • buttons の値が「step」のときには、対応するボタンは表示されない • 本引数の設定を省略した場合、増減幅は step1 の 10 倍となる
△	scale1	Integer	• 増減ボタンが押下された場合、入力値に対して本値で乗算または除算して入力値を増減する • 本引数の設定を省略した場合、増減幅は入力値を√2 で乗算または除算した値となる
△	scale2	Integer	• 増減ボタンが押下された場合、入力値に対して本値で乗算または除算して入力値を増減する • 本引数の設定を省略した場合、増減幅は scale1 を 2 乗した値で乗算または除算した値となる • buttons の値が「scale」のときには、対応するボタンは表示されない
△	errPopPos	Integer	バリデーションエラー時、textbox の Window から何階層上の Window にポップアップを表示するかを指定する
△	inputType	String	タブレット時のソフトウェアキーボードの type を指定する • "text"：文字入力の一般的なキーボード(デフォルト値) • "number"：数値入力用 • "tel"：数値入力用。小数点やマイナス値のキー入力が可能
その他			
デフォルトではテキストボックス内のテキスト位置は右寄せ			

textboxReal の場合に、inputType パラメータで指定するソフトウェアキーボードは以下の通りです。



“text” “number” “tel”

図 7.4.3 (a) textboxReal の場合の inputType によるソフトウェアキーボード

注
- 負の入力値を許容する textboxReal の場合は、inputType=”tel”を指定してください。
- 負の入力値を許容しない textboxReal の場合は、inputType=”number”を指定してください。

JavaScript ライブラリ

refresh

概要		
更新		
関数名		
refresh		
引数		
順番	**型**	**説明**
1 cur	String	テキストボックスの入力値
2 min	String	入力値チェック時の最小値
3 max	String	入力値チェック時の最大値
4 disabled	Boolean	無効 (true) / 有効 (false) の切り替え
戻り値		
型	**説明**	
-	-	
その他		
入力値チェック • cur が min 以下の場合、減ボタンが無効状態になる。 • cur が max 以上の場合、増ボタンが無効状態になる。 • cur が min 未満、または max 超過の場合、文字色が赤くなる。 • cur が数値でなかった場合、入力値が空になって背景色が赤くなり、増減ボタンが無効状態になる。		

setCallback

概要		
Callback 関数の登録		
関数名		
setCallback		
引数		
順番	**型**	**説明**
1 callback	Function	Callback 関数
引数 (callback)		
順番	**型**	**説明**
1 id	String	id 名 ("部品に指定した id".textbox)
2 operation	String	"changed"の文字列
3 value	String	入力値
戻り値		
型	**説明**	
-	-	

その他
-

setFocus

概要		
テキストボックスにフォーカスをあてる		
関数名		
setFocus		
引数		
順番	型	説明
-	-	-
戻り値		
型	説明	
-	-	
その他		
-		

getValue

概要	
テキストボックスの入力値を取得	
関数名	
getValue	
引数	
順番	型	説明
-	-	-
戻り値	
型	説明
String	テキストボックスの入力値 decimal 指定時は小数点桁数にあわせた値を返す
その他	
-	
制限事項	
-	

setCursorPos

概要		
テキストボックス内カーソル位置の設定		
関数名		
setCursorPos		
引数		
順番	型	説明
1 targetStart	Number	カーソルをセットする開始位置
2 targetEnd	Number	カーソルをセットする終了位置 範囲を指定しない場合は、開始位置と同じ値を設定する 終了位置には、開始位置以上の値を指定する
戻り値		
型	説明	
-	-	
その他		
-		

7.4.4 textboxString

コントローラ概要

部品名
touch  mouse 
概要
String 型テキストボックス
その他
文字列用。モードを切り替えることで、入力値のチェック内容を切り替えることができる。 デフォルト時のチェック内容は以下の通り • 入力後のバリデーションチェック 「& ' " / ¥」は入力不可 上記以外が入力されていた場合は、「次の文字は使えません」というエラーメッセージを表示する。 • 最小文字数 1 文字 • 半角文字を 1 文字、全角文字を 2 文字とカウントする

HTML テンプレートエンジン

ihmi タグ type			
textboxString			
引数			
〇：必須 △：オプション	**パラメータ**	**型**	**説明**
△	designpattern	String	デザインパターンの選択 • "touch"： タッチ操作用デザイン • "mouse"： マウス操作用デザイン
〇	mode	String	入力値のチェックモード • デフォルトで用意されているチェックモード名は「dataname」「datanamefilter」「comment」「path」「filepath」「allallowed」。詳細は後述する JavaScript ライブラリの「setMode」を参照してください。 • ユーザ任意のモードを用いる場合は、任意のモード名。詳細は後述する JavaScript ライブラリの「setConfig」を参照してください。
△	height	String	テキストボックスの高さを指定する • テキスト位置は上下中央寄せに固定される • テキストボックスの高さを指定すると、増減ボタンも同じ高さに調整される • 本値に上下 2px のボーダー幅は含まれない 例) height="30px"を指定すると、30px+(2px+2px)(ボーダー幅)の高さとなる

△	width	String	テキストボックスの横幅を指定する 本値に上下 2px のボーダー幅と、左右の padding は含まれない 例) • width=”100px”を指定すると、デザインパターンが"touch"の場合、100px+2px+16px=118px の幅となる • "mouse"の場合、100px+2px+8px=110px の幅となる
△	errPopPos	Integer	バリデーションエラー時、textbox の Window から何階層上の Window にポップアップを表示するかを指定する
その他			
デフォルトでは、テキストボックス内のテキスト位置は左寄せ			

JavaScript ライブラリ

refresh

概要		
更新		
関数名		
refresh		
引数		
順番	**型**	**説明**
1 text	String	テキストボックスの入力値
2 maxLen	Number	最大文字数。半角文字は 1 文字、全角文字は 2 文字とカウントする
3 disabled	Boolean	無効(true) / 有効(false)の切り替え
戻り値		
型	**説明**	
-	-	
その他		
最小文字数が 1 以上かつ text の文字数が 0 のときに背景色を変更する。		

setConfig

概要		
ユーザ指定のキー入力チェック条件、入力値チェック条件、最小文字数を設定する		
関数名		
setConfig		
引数		
順番	**型**	**説明**

1 filter	Function	常に true を返す関数を作成し、その関数のアドレスを設定してください。
		
		function dummyFunc(keyCode) {
		  return true;
		}

2 validate	Function	入力値チェックに用いる関数
		
		例 7,8,9 入力時エラーにする場合
		function orgValidate(text) {
		  if ((/[789]/).test(text)) {
		  return new top.Error("validattion error");
		  }
		  return text;

		}
3 minLen	Number	最小文字数
戻り値		
型	**説明**	
-	-	
その他		
最小文字数が 1 以上かつ text の文字数が 0 のときに背景色を変更する。		

setMode

概要		
モードによって、キー入力チェック条件、入力値チェック条件、最小文字数の設定を変える		
関数名		
setMode		
引数		
順番	**型**	**説明**
1 mode	String	入力制限モード名
戻り値		
型	**説明**	
-	-	
その他		

モードによるチェック条件

comment
- 入力後のバリデーションチェック(デフォルト)
  「& ' " / ¥」は入力不可。
  上記以外が入力されていた場合は、「次の文字は使えません」というエラーメッセージを表示する。
- 最小文字数
  0 文字

dataname
- 入力後のバリデーションチェック
  半角の「a～z, A～Z, 0～9, カナカタカナ (･, ﾞ, ﾟ を含む)」以外が入力された場合、「不正な文字が使われています」というエラーメッセージを表示する。
- 最小文字数
  1 文字

datanamefilter
- 入力後のバリデーションチェック
  半角の「a～z, A～Z, 0～9, スペース, カナカタカナ (･, ﾞ, ﾟ を含む)」以外が入力された場合、「不正な文字が使われています」というエラーメッセージを表示する。
- 最小文字数
  0 文字
- その他
  全角スペースが含まれていた場合、半角スペースに変換する。

path (入力可能例: MC:ihmi/)
- 入力後のバリデーションチェック
  半角の「a～z, A～Z, 0～9, : (コロン), ¥ (バックスラッシュ), カナカタカナ (･, ﾞ, ﾟ を含む)」以外が入力された場合、「不正な文字が使われています」というエラーメッセージを表示する。
  半角の「a～z, A～Z, 0～9」2～3 文字の後に: (半角コロン)が続く形式で入力されていない場合、「デバイス名が不適当です」というエラーメッセージを表示する。
- 最小文字数
  1 文字
- その他
  入力値の最後に¥ (半角バックスラッシュ)が付いていない場合¥を付け加える。

filepath(入力可能例: MC:ihmi/ihmipate.stm)
- 入力後のバリデーションチェック

半角の「a～z, A～Z, 0～9, : (コロン), ¥ (バックスラッシュ), . (ピリオド), カナカタカナ (･, ﾞ, ﾟ を含む)」以外が入力された場合、「不正な文字が使われています」というエラーメッセージを表示する。
半角の「a～z, A～Z, 0～9」2～3 文字の後に: (半角コロン)が続く形式で入力されていない場合、「デバイス名が不適当です」というエラーメッセージを表示する。

- 最小文字数
  1 文字

allallowed

- 入力後のバリデーションチェック
  なし
- 最小文字数
  0 文字

setCallback

概要		
Callback 関数の登録		
関数名		
setCallback		
引数		
順番	**型**	**説明**
1 callback	Function	Callback 関数
引数 (callback)		
順番	**型**	**説明**
1 id	String	id 名
2 operation	String	"changed"の文字列
3 value	String	入力値
戻り値		
型	**説明**	
-	-	
その他		
-		

setFocus

概要		
テキストボックスにフォーカスをあてる		
関数名		
setFocus		
引数		
順番	**型**	**説明**
-	-	-
戻り値		
型	**説明**	
-	-	
その他		
-		

getValue

概要
テキストボックスの入力値を取得
関数名
getValue

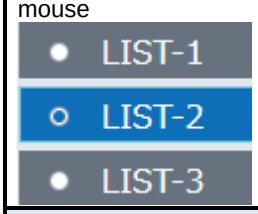
引数		
順番	**型**	**説明**
-	-	-
戻り値		
型	**説明**	
String	テキストボックスの入力値	
その他		
-		

setCursorPos

概要		
テキストボックス内カーソル位置の設定		
関数名		
setCursorPos		
引数		
順番	**型**	**説明**
1 targetStart	Number	カーソルをセットする開始位置
2 targetEnd	Number	カーソルをセットする終了位置 範囲を指定しない場合は、開始位置と同じ値を設定する 終了位置には、開始位置以上の値を指定する
戻り値		
型	**説明**	
-	-	
その他		
-		

7.4.5 listSelect & listSelectOption

コントローラ概要

デザイン
touch  touch  mouse  mouse 
概要
• 複数の項目から 1 つを選択するためのコントローラ • listSelect(リストの外枠)と listSelectOption(リスト内項目)の 2 種類の ihmi タグで作成する
その他
-

HTML テンプレートエンジン

ihmi タグ type			
listSelect			
引数			
〇：必須 △：オプション	**パラメータ**	**型**	**説明**
△	designpattern	String	デザインパターンの選択 • "touch"： タッチ操作用デザイン • "mouse"： マウス操作用デザイン

△	class	String	• リスト全体に付加する拡張クラス
			• class 属性で高さの指定が可能。ただし拡張クラスに定義する際は、iHMI コンポーネントで定義されているデフォルトのサイズを上書きする必要があるため、以下のように詳細度を上げて定義する
			/* 拡張クラス AAA を定義 */
			.list-select.AAA {
			height: ***px;
			}

△	icon	Boolean	• false を指定すると ON/OFF アイコンを表示しない • true(false 以外)を指定すると ON/OFF アイコンを表示する
△	horizontal	Boolean	• リスト項目を横並びにするときのみ true を指定する • 親ブロックの幅に対して、ボタンの幅が均等に設定される
△	タグ内に設定した値 <ihmi type="listSelect"> xxx</ihmi>	String	<ihmi type="listSelectOption">タグ
その他			
-			

ihmi タグ type			
listSelectOption			
引数			
〇：必須 △：オプション	**パラメータ**	**型**	**説明**
○	value	String	リストの項目値

△	class	String	• 項目に付加する拡張クラス
			• 横並び(horizontal="true")の場合、本パラメータに、幅を調整するセレクタ(padding や margin)は指定しないこと
			• class 属性で高さの指定が可能。ただし拡張クラスに定義する際は、iHMI コンポーネントで定義されているデフォルトのサイズを上書きする必要があるため、以下のように詳細度を上げて定義する
			/* 拡張クラス AAA を定義 */
			.list-select .AAA {
			height: ***px;
			}

△	width	Integer	ボタンの横幅
△	タグ内に設定した値 <ihmi type=" listSelectOption"> xxx </ ihmi >	String	項目内テキスト
その他			
-			

JavaScript ライブラリ

refresh

概要		
更新		
関数名		
refresh		
引数		
順番	**型**	**説明**
1 options	object	リスト項目の配列

		• 横並び設定時に(horizontal="true")、本パラメータを指定して refresh()を Call した場合は、setWidth()を合わせて Call して項目の幅を調整すること • 本パラメータを指定すると、元のリスト項目が削除され、本値でリスト項目が更新される
2 selected	String	初期表示で選択状態にするリスト項目を、引数 value の値で指定する
3 disabled	Boolean	無効 (true) / 有効 (false) 切替
引数 (options)		
順番	**型**	**説明**
1 value	String	リストの項目の値 この値がコンポーネントからの Callback 関数に返される
2 text	String	項目に表示するテキスト
3 className	String	項目に対する拡張クラス
戻り値		
型	**説明**	
-	-	
その他		
refresh をしない項目の引数には null を指定する。ただし、disabled には null は指定できないため、true / false のいずれかを必ず指定する。		

setWidth

概要		
リストの横幅の再計算		
関数名		
setWidth		
引数		
順番	**型**	**説明**
-	-	-
戻り値		
型	**説明**	
-	-	
その他		
横並び設定時、かつ refresh()でリストを作成した場合は、本関数を Call すること。リストの横幅とリスト内テキストの幅が本関数にて再計算される。		

setCallback

概要		
Callback 関数の登録		
関数名		
setCallback		
引数		
順番	**型**	**説明**
1 callback	Function	Callback 関数
引数 (callback)		
順番	**型**	**説明**
1 id	String	id 名
2 type	String	"changed"の文字列
3 value	String	選択された項目の値
戻り値		
型	**説明**	
-	-	

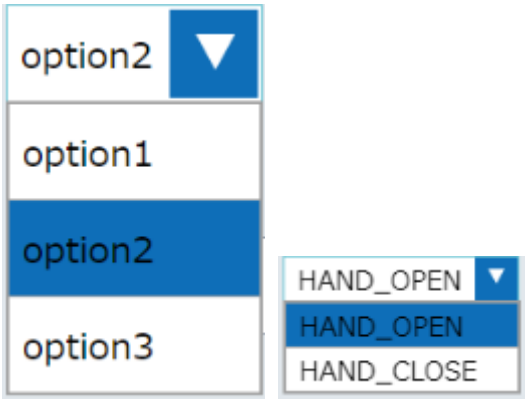
その他
-

getValue

概要		
現在の選択値を取得		
関数名		
getValue		
引数		
順番	型	説明
-	-	-
戻り値		
型	説明	
String	現在の選択値	
その他		
-		

7.4.6 select

コントローラ概要

デザイン
touch / mouse 
概要
• 複数の候補から入力項目を選ぶ際のセレクトボックス • select(外枠・展開ボタン)と selectOption(項目)の 2 種類の ihmi タグで作成する
その他
-

HTML テンプレートエンジン

ihmi タグ type			
select			
引数			
○：必須 △：オプション	パラメータ	型	説明
△	designpattern	String	デザインパターンの選択 • "touch"： タッチ操作用デザイン • "mouse"： マウス操作用デザイン
△	width	String	セレクトボックスの幅 ボックス全体の幅ではない。展開ボタン、padding、margine、border を除いた幅

△	class	String	セレクトボックス内のラベルに付加するクラス
その他			
-			

ihmi タグ type			
selectOption			
引数			
〇：必須 △：オプション	**パラメータ**	**型**	**説明**
〇	value	String	選択項目の値
△	class	String	選択項目に付加するクラス
△	image	String	選択項目の値の左横に表示する画像を相対パスまたは絶対パスで指定する
△	タグ内に設定した値 <ihmi type="selectOption">xxx</ihmi>	String	選択項目のテキスト
その他			
-			

JavaScript ライブラリ

refresh

概要		
更新		
関数名		
refresh		
引数		
順番	**型**	**説明**
1 options	Object	• 選択項目 • すでに HTML で選択項目を生成済みの場合、それに追加する形でリストを生成 例1) テキスト＋選択項目の値＋画像あり ["text1", 0 , "AAA.png"], ["text2", 1 , "BBB.png"], 例2) リスト項目番号を省略する場合は、テキストのみを配列で指定する ["addoption4","addoption5","addoption6"] • 追加の必要がなければ、空の配列を[ ]で設定すること
2 selectedValue	Number String	選択済みの項目値(リスト項目番号またはリストに表示するテキスト) • リスト項目番号: 下記の「引数(options)」の第 2 引数に相当 • リストに表示するテキスト: 下記「引数(options)」の第 1 引数に相当
3 selectedText	String	以前選択されていた値が何らかの操作で選択項目から失われた場合に、以前選択されていた値が削除されたことを赤字で表示する 例) データ名を本コントローラで選択したのちにデータの削除操作が入ると、そのデータ名は選択肢から失われる。そのような場合に、選択していた値を表現する文字列(selectedText)を赤字で表示する
4 disabled	boolean	無効(true) / 有効(false)の切り替え
引数 (options)		
順番	**型**	**説明**
1	String	選択項目のテキスト

2	Number String	選択項目の値 • 本引数を省略した場合、第 1 引数の値が選択項目の値として代用される • 本値は、コンポーネントから Callback 関数で返される
3	String	リストに表示する画像 • 本引数は省略可 • 画像のサイズは、以下のようにコンポーネント CSS で定義される designpattern="touch" : 40px designpattern="mouse" : 26px
戻り値		
型	**説明**	
-	-	
その他		
-		

getValue

概要		
現在値を取得		
関数名		
getValue		
引数		
順番	**型**	**説明**
-	-	-
戻り値		
型	**説明**	
Number String	現在の選択値	
その他		
-		

setCallback

概要		
Callback 関数の登録		
関数名		
setCallback		
引数		
順番	**型**	**説明**
1 callback	Function	Callback 関数
引数 (callback)		
順番	**型**	**説明**
1 id	String	id 名
2 type	String	"changed"の文字列
3 value	String	選択値
戻り値		
型	**説明**	
-	-	
その他		
-		

setFocus

概要
セレクトボタンにフォーカスをあてる

関数名		
setFocus		
引数		
順番	**型**	**説明**
-	-	-
戻り値		
型	**説明**	
-	-	
その他		
-		

7.4.7 コントローラ間共通情報

カラーテーブル

概要		
• 複数のカラーコードをまとめたオブジェクト • 部品間で使用する色をテーブル化 • refresh()時に色を変更する際に本テーブルを使用		
プロパティ名と値		
プロパティ名	カラーコード	用途
red02	#E60012	否定や危険を表す色
green03	#079A3E	安全や OK を表す色
orange01	#EB6100	停止やキャンセルを表す色
yellow02	#FFE109	注意を表す色
blue01	#0E6EB8	選択の色
black01	#000000	明るい背景色上のテキスト 有効状態の文字色
white01	#FFFFFF	暗い背景色上のテキスト
その他		
本テーブル使用部品：button, textboxInteger, textboxReal		

7.5 ロボット制御装置とのインタフェース

ロボット制御装置とのインタフェースとして JavaScript の API を提供します。API を使用すると、ロボット制御装置(サーバ)の KAREL 変数値と I/O 値の取得、設定を行うことができます。本 API は top フレームで読み込まれ、”top.ihmi_setVar”のように使用します。

7.5.1 top.ihmi_setVar

top.ihmi_setVar

概要		
指定された KAREL 変数に値を設定します。		
関数名		
ihmi_setVar		
引数		
変数名	**型**	**説明**
1 prog_name	String	KAREL プログラム名
2 var_name	String	KAREL 変数名
3 value	String	設定値

その他
BOOLEAN 型の KAREL 変数に設定する場合、TRUE は”1”、FALSE は”0”と指定してください。

注
top.ihmi_setVar は可変長配列の KAREL 変数をサポートしていません。

7.5.2 top.ihmi_getVar

top.ihmi_getVar

概要		
指定された KAREL 変数の値を取得します。		
関数名		
ihmi_getVar		
引数		
変数名	**型**	**説明**
1 prog_name	String	KAREL プログラム名
2 var_name	String	KAREL 変数名
3 callback	Function	Callback 関数
4 callbackarg1	String	Callback 関数の第 5 引数値
5 callbackarg2	String	Callback 関数の第 6 引数値
引数 (callback)		
変数名	**型**	**説明**
1 prog_name	String	KAREL プログラム名
2 var_name	String	KAREL 変数名
3 type_code	String	KAREL 変数の型 (通常は使用しません)
4 value	String	取得値
5 callbackarg1	String	ihmi_getVar の第 4 引数値
6 callbackarg2	String	ihmi_getVar の第 5 引数値
その他		
-		

注
top.ihmi_getVar は可変長配列の KAREL 変数をサポートしていません。

7.5.3 top.ihmi_iovalset

top.ihmi_iovalset

概要		
指定された I/O に値を設定します。		
関数名		
ihmi_ iovalset		
引数		
変数名	**型**	**説明**
1 io_type	String	設定する I/O のタイプ。以下の数値を指定します。 DO:2, RO:9, GO:19, AO:4
2 index	String	設定する I/O のインデックス
3 value	String	設定値
その他		
DO, RO に設定する場合、ON は”1”、OFF は”0”と指定してください。		

7.5.4 top.ihmi_iovalrd

top.ihmi_iovalrd

概要		
指定された I/O の値を取得します。		
関数名		
ihmi_iovalrd		
引数		
変数名	**型**	**説明**
1 io_type	String	取得する I/O のタイプ。以下の数値を指定します。 DI:1, DO:2, AI:3, AO:4, RI:8, RO:9, SO:12, GI:18, GO:19
2 index	String	取得する I/O のインデックス
3 callback	Function	Callback 関数
4 callbackarg1	String	Callback 関数の第 4 引数値
5 callbackarg2	String	Callback 関数の第 5 引数値
引数 (callback)		
変数名	**型**	**説明**
1 io_type	String	取得する I/O のタイプ
2 index	String	取得する I/O のインデックス
3 value	String	I/O の取得値
4 callbackarg1	String	ihmi_iovalrd 関数の第 4 引数値
5 callbackarg2	String	ihmi_iovalrd 関数の第 5 引数値
その他		
-		

7.5.5 top.ihmi_setVar2

top.ihmi_setVar2

概要		
指定された KAREL 変数に値を設定します。		
関数名		
ihmi_setVar2		
引数		
変数名	**型**	**説明**
1 prog_name	String	KAREL プログラム名
2 var_name	String	KAREL 変数名
3 value	String	設定値
4 callback	Function	Callback 関数
5 callbackarg1	String	Callback 関数の第 5 引数値
6 callbackarg2	String	Callback 関数の第 6 引数値
引数 (callback)		
変数名	**型**	**説明**
1 prog_name	String	KAREL プログラム名
2 var_name	String	KAREL 変数名
3 value	String	設定値
4 status	String	成功時に 0 を返し、失敗時に非ゼロを返す
5 callbackarg1	String	ihmi_setVar2 関数の第 5 引数値
6 callbackarg2	String	ihmi_setVar2 関数の第 6 引数値
その他		
BOOLEAN 型の KAREL 変数に設定する場合、TRUE は”1”、FALSE は”0”と指定してください。		

> ⚠ **注意**
> top.ihmi_setVar2 は V9.40P/06 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/05 版以前の場合は、プラグインソフトをインストールする前に V9.40P/06 版以降の最新版にアップデートしてください。」、と明記してください。

> **注**
> top.ihmi_setVar2 は可変長配列の KAREL 変数をサポートしていません。

7.5.6 top.ihmi_getVar2

top.ihmi_getVar2

概要		
指定された KAREL 変数の値を取得します。		
関数名		
ihmi_getVar2		
引数		
変数名	**型**	**説明**
1 prog_name	String	KAREL プログラム名
2 var_name	String	KAREL 変数名
3 callback	Function	Callback 関数
4 callbackarg1	String	Callback 関数の第 6 引数値
5 callbackarg2	String	Callback 関数の第 7 引数値
引数 (callback)		
変数名	**型**	**説明**
1 prog_name	String	KAREL プログラム名
2 var_name	String	KAREL 変数名
3 type_code	String	KAREL 変数の型 (通常は使用しません)
4 value	String	取得値
5 status	String	成功時に 0 を返し、失敗時に非ゼロを返す
6 callbackarg1	String	ihmi_getVar2 関数の第 4 引数値
7 callbackarg2	String	ihmi_getVar2 関数の第 5 引数値
その他		
-		

> ⚠ **注意**
> top.ihmi_getVar2 は V9.40P/06 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/05 版以前の場合は、プラグインソフトをインストールする前に V9.40P/06 版以降の最新版にアップデートしてください。」、と明記してください。

> **注**
> top.ihmi_getVar2 は可変長配列の KAREL 変数をサポートしていません。

7.5.7 top.ihmi_iovalset2

top.ihmi_iovalset2

概要
指定された I/O に値を設定します。
関数名
ihmi_ iovalset2

引数		
変数名	**型**	**説明**
1 io_type	String	設定する I/O のタイプ。以下の数値を指定します。 DO:2, RO:9, GO:19, AO:4
2 index	String	設定する I/O のインデックス
3 value	String	設定値
4 callback	Function	Callback 関数
5 callbackarg1	String	Callback 関数の第 5 引数値
6 callbackarg2	String	Callback 関数の第 6 引数値
引数 (callback)		
変数名	**型**	**説明**
1 io_type	String	指定された I/O のタイプ。 DO:2, RO:9, GO:19, AO:4
2 index	String	指定された I/O のインデックス
3 value	String	設定値
4 status	String	成功時に 0 を返し、失敗時に非ゼロを返す
5 callbackarg1	String	ihmi_iovalset2 関数の第 5 引数値
6 callbackarg2	String	ihmi_iovalset2 関数の第 6 引数値
その他		
DO, RO に設定する場合、ON は”1”、OFF は”0”と指定してください。		

⚠ **注意**
top.ihmi_iovalset2 は V9.40P/06 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/05 版以前の場合は、プラグインソフトをインストールする前に V9.40P/06 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.8 top.ihmi_iovalrd2

top.ihmi_iovalrd2

概要		
指定された I/O 値を取得します。		
関数名		
ihmi_iovalrd2		
引数		
変数名	**型**	**説明**
1 io_type	String	取得する I/O のタイプ。以下の数値を指定します。 DI:1, DO:2, AI:3, AO:4, RI:8, RO:9, SO:12, GI:18, GO:19
2 index	String	取得する I/O のインデックス
3 callback	Function	Callback 関数
4 callbackarg1	String	Callback 関数の第 5 引数値
5 callbackarg2	String	Callback 関数の第 6 引数値
引数 (callback)		
変数名	**型**	**説明**
1 io_type	String	指定された I/O のタイプ。 DI:1, DO:2, RI:8, RO:9, GI:18, GO:19, AI:3, AO:4
2 index	String	指定された I/O のインデックス
3 value	String	取得値
4 status	String	成功時に 0 を返し、失敗時に非ゼロを返す
5 callbackarg1	String	ihmi_iovalrd2 関数の第 4 引数値
6 callbackarg2	String	ihmi_iovalrd2 関数の第 5 引数値
その他		
-		

> ⚠ **注意**
> top.ihmi_iovalrd2 は V9.40P/06 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/05 版以前の場合は、プラグインソフトをインストールする前に V9.40P/06 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.9 top.frpl_utl.startIOGroupMonitor

top.frpl_utl.startIOGroupMonitor

概要		
指定した範囲の I/O 値の監視を開始します。指定した範囲の I/O 値に変動があった場合 Callback します。		
関数名		
frpl_utl.startIOGroupMonitor(type, index, count, callback, callFrame, callbackarg)		
引数		
変数名	**型**	**説明**
1 type	Number	監視する I/O のタイプ。以下の数値を指定します。 DI:1, DO:2, RI:8, RO:9, SO:12
2 index	Number	監視する I/O の開始番号
3 count	Number	監視する I/O 数 監視する I/O 数の上限は 500 です。上限数を超えて監視することはできません。
4 callback	Function	監視通知時に呼び出す Callback 関数
5 callFrame	Object	本 API 呼び出し元の Windows オブジェクト
6 callbackarg	String / Number	Callback 関数に渡される引数 (任意)

戻り値		
変数名	**型**	**説明**
1 promise	Promise	PromiseResult として respObj を返します。 PromiseState は下記の通りとなります。 • resolve : I/O 監視開始が正常終了 • reject : I/O 監視開始に失敗などのエラーが発生
詳細: 戻り値 (respObj)		
変数名	**型**	**説明**
1 status	String	結果のステータスを返します。 ステータスの詳細は後述の[詳細: Status 一覧]を参照してください。
2 errorDescription	String	reject 理由のメッセージ resolve の場合は null を返します。 メッセージの詳細は後述の[詳細: Status 一覧]を参照してください。
3 type	Number	監視を開始した I/O の種類 DI:1, DO:2, RI:8, RO:9
4 ioValueArray	Array	引数で指定した監視 I/O の初期値 引数で指定した index から count の個数分のデータが Object 型で格納されます。 詳細は[詳細: 戻り値(respObj.ioValueArray)]を参照してください。
詳細: 戻り値 (respObj.ioValueArray)		
変数名	**型**	**説明**
1 index	String	I/O のインデックス番号
2 val	String	I/O の値。指定した I/O の種類によって以下の値が格納されます。 • I/O の値 (数値) • I/O の状態 (ON: 1, OFF: 0)
備考		
未割当等で取得できない val は空になります。		

使用例) DO[1]~[5]でモニタ開始: frpl_utl.startIOGroupMonitor(2, 1, 5, …);
respObj.ioValueArray[
{ index : “1”, val : “1” },
{ index : “2”, val : “0” },
{ index : “3”, val : “1” },
{ index : “4”, val : “0” },
{ index : “5”, val : “” } // 未割付
]

詳細: Callback		
変数名	**型**	**説明**
1 type	String	値が変動した I/O のタイプ DI:1, DO:2, RI:8, RO:9
2 index	String	値が変動した I/O のインデックス番号
3 value	String	変動した I/O の値。指定した I/O タイプによって以下の値を返します。 • I/O の値 (数値) • I/O の状態 (ON: 1, OFF: 0)
4 callbackarg	String / Number	呼び出し時に指定した引数オブジェクト
備考		
Callback 先が無いなどの理由で Callback 通知に失敗した場合は、コンソールエラーを出します。		

詳細: Status 一覧		
status	**errorDescription**	**説明**
0	Null	成功の場合
1	RequestError	ロボット制御装置へのリクエストに失敗した場合
2	TypeError: Argument (引数番号(パラメータ名)) is not of type (正しい型). 例: Argument 1(type) is not of type Number.	引数不足や型エラー
3	IOTypeError: I/O type (引数に指定した type) does not exist. 例: I/O type 999 does not exist.	存在しない I/O タイプが指定された場合
4	IORangeError: I/O range does not exist.	存在しない I/O 範囲が指定された場合
5	StartMonitorError: The maximum number of monitors that can be configured has been exceeded.	設定可能な監視数の上限を超えている場合
6	StartMonitorError: The same argument has already been set.	既に同じ引数でモニタが開始されている場合
7	StopMonitorError: The designated monitor does not exist.	存在しないモニタの停止を指定された場合 (API を使用してモニタを開始していない)

⚠ **注意**
top.frpl_utl.startIOGroupMonitor および top.frpl_utl.stopIOGroupMonitor は V9.40P/49 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/48 版以前の場合は、プラグインソフトをインストールする前に V9.40P/49 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.10 top.frpl_utl.stopIOGroupMonitor

top.frpl_utl.stopIOGroupMonitor

概要
指定した範囲の I/O 値の監視を停止します。
関数名
frpl_utl.stopIOGroupMonitor(type, index, count, callback, callFrame)

引数		
変数名	**型**	**説明**
1 type	Number	監視する I/O のタイプ。以下の数値を指定します。 DI:1, DO:2, RI:8, RO:9, SO:12
2 index	Number	監視する I/O の開始番号
3 count	Number	監視する I/O 数 監視する I/O 数の上限は 500 です。上限数を超えて監視することはできません。
4 callback	Function	監視通知時に呼び出す Callback 関数
5 callFrame	Object	本 API 呼び出し元の Windows オブジェクト
備考		
• 監視を停止する場合は、必ず開始時と同じパラメータを指定する必要があります。例えば、index を 1、count を 10 に設定して監視開始した場合、停止時に index を 1、count を 9 に設定して 1～9 までのモニタを停止することはできません。 • 呼び出し元が unload された場合は、API 側でモニタを停止します。		

戻り値		
変数名	**型**	**説明**
1 promise	Promise	promiseResult として respObj を返します。 PromiseState は下記の通りとなります。 • resolve : I/O 監視停止が正常終了 • reject : I/O 監視停止に失敗などのエラーが発生
詳細: 戻り値 (respObj)		
変数名	**型**	**説明**
1 status	String	結果のステータスを返します。 ステータスの詳細は、前節の[詳細: Status 一覧]を参照してください。
2 errorDescription	String	reject 理由のメッセージ resolve の場合は null を返します。 メッセージの詳細は、前節の[詳細: Status 一覧]を参照してください。

詳細: Callback		
変数名	**型**	**説明**
1 type	String	値が変動した I/O のタイプ DI:1, DO:2, RI:8, RO:9
2 index	String	値が変動した I/O のインデックス番号
3 value	String	変動した I/O の値。指定した I/O タイプによって以下の値を返します。 • I/O の値 (数値) • I/O の状態 (ON: 1, OFF: 0)
4 callbackarg	String / Number	呼び出し時に指定した引数オブジェクト
備考		
Callback 先が無いなどの理由で Callback 通知に失敗した場合は、コンソールエラーを出します。		

⚠ 注意
top.frpl_utl.startIOGroupMonitor および top.frpl_utl.stopIOGroupMonitor は V9.40P/49 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/48 版以前の場合は、プラグインソフトをインストールする前に V9.40P/49 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.11 top.frpl_utl.jumpPage

top.frpl_utl.jumpPage

概要
任意の FANUC ロボット標準画面に遷移します。本 API には戻り値はありません。

関数名		
frpl_utl.jumpPage		
引数		
変数名	**型**	**説明**
1 spId	String / Number	表示する画面の Softpart ID を指定します。 Softpart ID は$TP_CURSCRN[].$SP_ID を参照してください。
2 scrnId	String / Number	表示する画面の Screen ID を指定します。 Softpart ID は$TP_CURSCRN[].$SCRN_ID を参照してください。
その他		
• タブレット UI エディタを除く任意の画面へ遷移後、ステータスバーの戻るボタンを押下すると、遷移前の画面に戻ります。 • $TP_CURSCRN[].$SP_ID と$TP_CURSCRN[].$SCRN_ID の取得方法については、R-30iB Plus 取扱説明書 (基本操作編) (B-83284JA)の 9.11.6.1 節「ソフトパーツ ID スクリーン ID」を参照してください。		

⚠ 注意
top.frpl_utl.jumpPage は V9.40P/45 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/44 版以前の場合は、プラグインソフトをインストールする前に V9.40P/45 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.12 top.frpl_utl.getMacAddr

top.frpl_utl.getMacAddr

概要		
ロボット制御装置のポート番号 1 の MAC アドレスを文字列で返します。本 API は引数なしで呼び出します。		
関数名		
frpl_utl.getMacAddr		
引数		
変数名	**型**	**説明**
-	-	-

戻り値		
変数名	**型**	**説明**
1 promise	Promise	PromiseResult として respObj を返します。 PromiseState は下記の通りとなります。 • resolve : MAC アドレス取得が正常終了 • reject : 内部エラー等が発生
詳細: resolve 戻り値 (respObj)		
変数名	**型**	**説明**
1 macaddr	String	ロボット制御装置のポート番号 1 の MAC アドレスを文字列で返します。 例) 00:e00:e4:80:8f:2d
詳細: reject 戻り値 (respObj)		
変数名	**型**	**説明**
1 errorCode	String	エラーコードを持つエラーが発生した場合にその値を返します。
2 errorDescription	String	エラーの概要を以下の通り返します。 • getMacAddr is not available. : MAC アドレスを取得する内部プログラムが存在しません。 • Failed to get MAC address with errorCode. : MAC アドレス取得処理でエラーが発生しました。 • Failed to get expected response. : XHR に失敗したか、JSON が正常に取得できませんでした。

備考
-

> ⚠ **注意**
> top.frpl_utl.getMacAddr は V9.40P/48 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/47 版以前の場合は、プラグインソフトをインストールする前に V9.40P/48 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.13 top.frpl_utl.initPayloadIdentData

top.frpl_utl.initPayloadIdentData

概要		
負荷推定 API で測定した負荷推定データを初期化します。		
関数名		
frpl_utl.initPayloadIdentData(grpNum)		
引数		
変数名	**型**	**説明**
grpNum	String	グループ番号
戻り値		
変数名	**型**	**説明**
promise	Promise	PromiseResult として respObj を返します PromiseState は下記の通りです。 • resolve: 成功 • reject: キャンセルやエラー
詳細: 戻り値 (respObj)		
変数名	**型**	**説明**
status	String	結果のステータスを返します。 ステータスの詳細は本節の[詳細: Status 一覧]を参照してください。
errorDescription	String	reject 理由のメッセージ resolve の場合は null を返します。 メッセージの詳細は本節の[詳細: Status 一覧]を参照してください。

その他

- API を用いて負荷推定を始める時に使用してください。
- 本 API を使用しても負荷推定ガイドで測定した負荷推定データは初期化されません。
- API を用いてグループ 1 の負荷推定を行う場合の流れは以下になります。
  1. 測定データの初期化: top.frpl_utl.initPayloadIdentData(“1”)
  2. 位置#1 に移動
  3. 位置#1 で測定: top.frpl_utl.recordPayloadIdentData(“1”)
  4. 位置#2 に移動
  5. 位置#2 で測定: top.frpl_utl.recordPayloadIdentData(“2”)
  6. 位置#3 に移動
  7. 位置#3 で測定: top.frpl_utl.recordPayloadIdentData(“3”)
  8. 位置#4 に移動
  9. 位置#4 で測定: top.frpl_utl.recordPayloadIdentData(“4”)
  10. 測定結果を元に負荷推定
      - 質量は未知として推定: top.frpl_utl.getPayloadIdentResult(“0”, “0”)
      - 質量は既知で 10kg として推定: top.frpl_utl.getPayloadIdentResult(“1”, “10”)
  11. グループ 1 の負荷設定 5 に推定結果の適用: top.frpl_utl.setPayloadData(“1”, “5”, “comment”, m, x, y, z, “0”, “0”, “0”)

詳細: Status 一覧		
status	**errorDescription**	**説明**
0	Null	成功の場合

1	RequestError	制御装置へのリクエストに失敗した場合
2	ArgumentError	引数不足や型エラーの場合

⚠ **注意**
top.frpl_utl.initPayloadIdentData は V9.40P/60 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/59 版以前の場合は、プラグインソフトをインストールする前に V9.40P/60 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.14 top.frpl_utl.recordPayloadIdentData

top.frpl_utl.recordPayloadIdentData

概要		
質量と重心位置を推定するためのデータを現在位置で測定します。		
関数名		
frpl_utl.recordPayloadIdentData(posNum)		
引数		
変数名	**型**	**説明**
posNum	String	測定位置番号 位置#1:1, 位置#2:2, 位置#3:3, 位置#4:4
戻り値		
変数名	**型**	**説明**
promise	Promise	PromiseResult として respObj を返します PromiseState は下記の通りです。 • resolve: 成功 • reject: キャンセルやエラー
詳細: 戻り値 (respObj)		
変数名	**型**	**説明**
status	String	結果のステータスを返します。 ステータスの詳細は本節の[詳細: Status 一覧]を参照してください。
errorDescription	String	reject 理由のメッセージ resolve の場合は null を返します。 メッセージの詳細は本節の[詳細: Status 一覧]を参照してください。
その他		
• 姿勢が大きく異なる 4 か所で本 API を使用してデータを測定してください。		

詳細: Status 一覧		
status	**errorDescription**	**説明**
0	null	成功の場合
1	RequestError	制御装置へのリクエストに失敗した場合
2	TypeError	引数不足や型エラー
3	NonCollaboError	協働モード無効による測定失敗
4	InvDcsPoschError	DCS 位置速度チェック機能無効による測定失敗
5	UninitError	初期化の API 未実行

⚠ **注意**
top.frpl_utl.recordPayloadIdentData は V9.40P/60 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/59 版以前の場合は、プラグインソフトをインストールする前に V9.40P/60 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.15 top.frpl_utl.checkPayloadIdentData

top.frpl_utl.checkPayloadIdentData

概要		
負荷推定 API を用いて測定した結果を確認できます。		
関数名		
frpl_utl.recordPayloadIdentData()		
引数		
変数名	**型**	**説明**
-	-	-
戻り値		
変数名	**型**	**説明**
promise	Promise	PromiseResult として respObj を返します PromiseState は下記の通りです。 • resolve: 成功 • reject: キャンセルやエラー
詳細: 戻り値 (respObj)		
変数名	**型**	**説明**
errorDescription	String	resolve の場合は「null」、reject の場合はエラーメッセージを返します。ステータスの詳細は本節で後述します。
statusInPos1	String	位置#1 での測定結果 測定完了:0, 測定失敗:1, 未測定:2, 確認失敗: 3
statusInPos2	String	位置#2 での測定結果 測定完了:0, 測定失敗:1, 未測定:2, 確認失敗: 3
statusInPos3	String	位置#3 での測定結果 測定完了:0, 測定失敗:1, 未測定:2, 確認失敗: 3
statusInPos4	String	位置#4 での測定結果 測定完了:0, 測定失敗:1, 未測定:2, 確認失敗: 3
その他		
• frpl_utl.recordPayloadIdentData の測定成否を確認したいときに使用してください。 • 測定結果の確認が不要な場合には使用する必要はありません。		

errorDescription 一覧	
errorDescription	**説明**
null	成功の場合
RequestError	制御装置へのリクエストに失敗した場合

> ⚠ **注意**
> top.frpl_utl.checkPayloadIdentData は V9.40P/60 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/59 版以前の場合は、プラグインソフトをインストールする前に V9.40P/60 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.16 top.frpl_utl.getPayloadIdentResult

top.frpl_utl.getPayloadIdentResult

概要		
負荷推定 API での測定結果を元に負荷推定を行います。		
関数名		
frpl_utl.getPayloadIdentResult(massKnown, loadMass)		
引数		
変数名	**型**	**説明**
massKnown	String	質量が既知かどうか 未知:0, 既知:1
loadMass	String	質量[kg]

戻り値		
変数名	**型**	**説明**
promise	Promise	PromiseResult として respObj を返します PromiseState は下記の通りです。 • resolve: 成功 • reject: キャンセルやエラー
詳細: 戻り値 (respObj)		
引数		
変数名	**型**	**説明**
judgement	String	判定
status	String	結果のステータスを返します。 ステータスの詳細は本節の[詳細: Status 一覧]を参照してください。
errorDescription	String	reject 理由のメッセージ resolve の場合は null を返します。 メッセージの詳細は本節の[詳細: Status 一覧]を参照してください。
mass	String	負荷の質量[kg]
momentJ4	String	軸廻りモーメント(J4) [Nm]
momentJ5	String	軸廻りモーメント(J5) [Nm]
momentJ6	String	軸廻りモーメント(J6) [Nm]
centerX	String	重心 (X) [m]
centerY	String	重心 (Y) [m]
centerZ	String	重心 (Z) [m]
allowMass	String	許容値 (負荷の質量 [kg])
allowMomJ4	String	許容値 (軸廻りモーメント(J4) [Nm])
allowMomJ5	String	許容値 (軸廻りモーメント(J5) [Nm])
allowMomJ6	String	許容値 (軸廻りモーメント(J6) [Nm])
その他		
• 4 か所で frpl_utl.recordPayloadIdentData を呼び出した後、本 API を使用してください。 • 第一引数で 1(既知)を設定した場合にのみ、第二引数は有効です。		

詳細: Status 一覧			
judgement	**status**	**errorDescription**	**説明**
0	0	null	成功(OK)の場合
	1	null	成功(許容値に近い)の場合
	2	null	成功(負荷オーバ)の場合
1	3	RequestError	制御装置へのリクエストに失敗した場合
	4	ArgumentError	引数不足や型エラーの場合
	5	MeasurementError	測定未完了の場合
	6	InternalError	内部エラーの場合

⚠ 注意
top.frpl_utl.getPayloadIdentResult は V9.40P/60 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/59 版以前の場合は、プラグインソフトをインストールする前に V9.40P/60 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.17 top.frpl_utl.setPayloadData

top.frpl_utl.setPayloadData

概要
負荷設定を行います。
関数名
frpl_utl.setPayloadData(grpNum, sheduleNum, comment, loadMass, centerX, centerY, centerZ, inertiaX, inertiaY, inertiaZ)

引数		
変数名	**型**	**説明**
grpNum	String	グループ番号
sheduleNum	String	負荷設定番号
comment	String	コメント
loadMass	String	質量[kg]
centerX	String	重心 X [m]
centerY	String	重心 Y [m]
centerZ	String	重心 Z [m]
inertiaX	String	イナーシャ X[kgm^2]
inertiaY	String	イナーシャ Y[kgm^2]
inertiaZ	String	イナーシャ Z[kgm^2]
戻り値		
変数名	**型**	**説明**
promise	Promise	PromiseResult として respObj を返します PromiseState は下記の通りです。 • resolve: 成功 • reject: キャンセルやエラー

詳細: 戻り値 (respObj)		
変数名	**型**	**説明**
status	String	結果のステータスを返します。 ステータスの詳細は本節の[詳細: Status 一覧]を参照してください。
errorDescription	String	reject 理由のメッセージ resolve の場合は null を返します。 メッセージの詳細は本節の[詳細: Status 一覧]を参照してください。
その他		
-		

詳細: Status 一覧		
status	**errorDescription**	**説明**
0	null	成功の場合
1	RequestError	制御装置へのリクエストに失敗した場合
2	TypeError	引数不足や型エラー
3	PayloadSetError	負荷設定失敗

⚠ **注意**
top.frpl_utl.setPayloadData は V9.40P/60 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/59 版以前の場合は、プラグインソフトをインストールする前に V9.40P/60 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.18 top.frpl_utl.applyDcsParam

top.frpl_utl.applyDcsParam

概要
DCS の設定変更を安全パラメータに適用するポップアップを表示します。 • 暗証番号の入力とパラメータ確認のポップアップが表示されます。表示されるポップアップは、図 7.5.18(a)を参照してください。 • 詳細はデュアルチェックセーフティ機能 取扱説明書(B-83184)を参照してください。
関数名
frpl_utl.applyDcsParam
引数

変数名	**型**	**説明**
popupWin	Object	ポップアップ画面表示先ウィンドウ

		指定がない場合は(null の場合は)、top に表示します。
cbArgs	Object	戻り値(respObj)に渡される引数 (任意)
戻り値		
変数名	**型**	**説明**
Promise	Promise	PromiseResult として respObj を返します。respObj の詳細は表[詳細: 戻り値(respObj)]を参照してください。 PromiseState は下記の通りです。 • resolve: 成功の場合 • reject: キャンセルやエラー(詳細は下図を参照)
詳細: 戻り値 (respObj)		
引数		
変数名	**型**	**説明**
cbArgs	Object	呼び出し時に指定した引数オブジェクト
その他		
-		



reject の場合は以下のようなメッセージが popup で表示されます。

- 適用が必要な項目はありません。
- 設定に間違いがあります。
- 設定できる処理時間係数をオーバーしています。処理時間係数がおさまるように、領域やモデルの設定を変更してください。
- 適用処理が既に実行中です。実行中の適用処理を完了してください。
- 暗証番号が違います。

図 7.5.18 (a) 表示 popup

> ⚠ **注意**
> top.frpl_utl.applyDcsParam は V9.40P/60 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/59 版以前の場合は、プラグインソフトをインストールする前に V9.40P/60 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.19 top.frpl_utl.getMultiKarelVars

top.frpl_utl.getMultiKarelVars

概要		
複数の KAREL 変数を一括取得します。		
関数名		
frpl_utl.getMultiKarelVars		
引数		
変数名	**型**	**説明**
jsonData	Object	以下を含む json 形式データ • KAREL プログラム名 • KAREL 変数名
詳細: jsonData		
変数名	**型**	**説明**
data	Object / Array	KAREL プログラム名と KAREL 変数名を含むオブジェクトまたは配列形式
詳細: data[x]		
変数名	**型**	**説明**
prog	String	取得する KAREL 変数名が定義されている KAREL プログラム名
vars	Array	取得する KAREL 変数名の配列 KAREL 構造体や配列の指定方法は後述する top.frpl_utl.setMultiKarelVars の[KAREL 配列と構造体の設定取得例]を参照してください。
例: jsonData		

data = オブジェクト
{
“data”: {
“prog”: ”KAREL プログラム名 1”,
“vars”: [“pos_x”, “pos_y”, “pos_z”]
}
}
		data = 配列
		{
		“data”: [{
		“prog”: ”KAREL プログラム名 1”,
		“vars”: [“pos_x”, “pos_y”, “pos_z”]
		},{
		“prog”: ” KAREL プログラム名 2”,
		“vars”: [“pallet”]
		}]
		}

戻り値		
変数名	**型**	**説明**
promise	Promise	PromiseResult として respObj を返します。respObj の詳細は[詳細: 戻り値(respObj)]を参照してください。 PromiseState は下記の通りです。 • resolve: KAREL 変数値の取得の正常終了 • reject: 取得失敗などのエラー発生
詳細: 戻り値 (respObj)		
変数名	**型**	**説明**
status	String	結果のステータスを返します。 ステータスの詳細は本節の[詳細: Status 一覧]を参照してください。
errorDescription	String	reject 理由のメッセージ resolve の場合は null を返します。 メッセージの詳細は本節の[詳細: Status 一覧]を参照してください。
data	Array	取得結果の内容が入ったオブジェクトの配列

		reject の場合は null を返します。
詳細: respObj.data[x]		
変数名	**型**	**説明**
prog	String	指定した KAREL プログラム名
vars	Object	取得した KAREL 変数名と変数値の組
詳細: respObj.data[x].vars		
変数名	**型**	**説明**
KAREL 変数	-(任意)	KAREL 変数値
例: data		

data = オブジェクト

```
{
  “data”: {
      “prog”: “IPL_FANUC001”,
      “vars”: {
         “pos_x”: 111,
         “pos_y”: 222,
         “pos_z”: 333
      }
   }
}
```

data = 配列

```
{
  “data”: [{
      “prog”: “IPL_FANUC001”,
      “vars”: {
         “pos_x”: 111,
         “pos_y”: 222,
         “pos_z”: 333
      }
   },{
      “prog”: “IPL_FANUC002”,
      “vars”: {
          “pallet”: [“seq1”, “seq4”, “seq2”]
      }
   }]
}
```

詳細: Status 一覧		
status	**errorDescription**	**説明**
0	Null	成功
1	Invalid JSON	不正な JSON
2	Invalid post in data	不正な data キー
3	Invalid post in program	不正な prog キー
4	Invalid post in variable	不正な vars キー
5	Unknown variable name	存在しない KAREL 変数
6	Unknown program name	存在しない KAREL プログラム
7	Bad variable or array index	不正な変数指定あるいは配列インデックス
8	Invalid value	不正な変数値
9	Unknown Error	解析不明な値

⚠ **注意**
top.frpl_utl.getMultiKarelVars および top.frpl_utl.setMultiKarelVars は V9.40P/68 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/67 版以前の場合は、プラグインソフトをインストールする前に V9.40P/68 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.20 top.frpl_utl.setMultiKarelVars

top.frpl_utl.setMultiKarelVars

概要
KAREL 変数を一括設定します。
関数名
frpl_utl.setMultiKarelVars

引数		
変数名	**型**	**説明**
jsonData	Object	以下を含む json 形式データ • KAREL プログラム名 • KAREL 変数名と値の組
詳細: jsonData		
変数名	**型**	**説明**
data	Object / Array	KAREL プログラム名と KAREL 変数名を含むオブジェクトまたは配列形式
詳細: data		
変数名	**型**	**説明**
prog	String	設定する KAREL 変数が定義されている KAREL プログラム名
vars	Object	設定する KAREL 変数名とその値のオブジェクト KAREL 構造体や配列の指定方法は[KAREL 配列と構造体の設定取得例]参照
詳細: vars		
変数名	**型**	**説明**
KAREL 変数	-(任意)	設定する KAREL 変数の値
例: data		

data = オブジェクト
{
“data”: {
“prog”: ”KAREL プログラム名 1”,
“vars”: {“pos_x”: 111,
“pos_y”: 222,
“pos_z”: 333 }
}
}
		data = 配列
		{
		“data”: [{
		“prog”: ”KAREL プログラム名 1”,
		“vars”: {“pos_x”: 111,
		“pos_y”: 222,
		“pos_z”: 333 }
		},{
		“prog”: ”KAREL プログラム名 2”
		“vars”: { “pallet”: [“seq1”, seq2, seq3] }
		}]
		}

戻り値		
変数名	**型**	**説明**
promise	Promise	PromiseResult として respObj を返します。respObj の詳細は後述の[詳細: 戻り値(respObj)]を参照してください。 PromiseState は下記の通りです。 • resolve: KAREL 変数値の設定の正常終了 • reject: 設定失敗などのエラー発生
詳細: 戻り値 (respObj)		
パラメータ	**型**	**説明**
status	String	結果のステータスを返します。 ステータスの詳細は前節の[詳細: Status 一覧]を参照してください。
errorDescription	String	reject 理由のメッセージ resolve の場合は null を返します。 メッセージの詳細は前節の[詳細: Status 一覧]を参照してください。

[KAREL 配列と構造体の設定取得例]

KAREL 配列

```
VAR
  num_ary: ARRAY[2, 3] OF INTEGER
BEGIN
  num_ary[1, 1] = 23
  num_ary[1, 2] = 6
  num_ary[1, 3] = 67
```

```
num_ary[2, 1] = 12
num_ary[2, 2] = 5
num_ary[2 ,3] = 78
```

KAREL 配列を設定する jsonData オブジェクト例

```
{
 “data”: {
  “prog”: ”KAREL プログラム名 1”,
  “vars”: {“num_ary”: [[23,6,7],[12,5,78]] }
 }
}
```

KAREL 配列取得時の jsonData オブジェクト例

```
{
 “data”: {
  “prog”: ”KAREL プログラム名 1”,
  “vars”: { “num_ary[1,1]” }
 }
}
```

KAREL 構造体

```
TYPE
 Point2d_t = STRUCTURE
  x: REAL
  y: REAL
 ENDSTRUCTURE
 rect_t = STRUCTURE
  pt1: point2d_t
  pt2: point2d_t
 ENDSTRUCTURE
VAR
 rectangle: rect_t
```

KAREL 構造体を設定する jsonData オブジェクト例

```
{
 “data”: {
  “prog”: ”KAREL プログラム名 1”,
  “vars”: {“rectangle1” : { “pt1”: {“x”: 0, “y”: 0},“pt2”: {“x”: 0, “y”: 0} }
 }
}
```

KAREL 構造体取得時の jsonData オブジェクト例

```
{
 “data”: {
  “prog”: ”KAREL プログラム名 1”,
  “vars”: [“rectangle1.pt1”]
 }
}
```

⚠ 注意

top.frpl_utl.getMultiKarelVars および top.frpl_utl.setMultiKarelVars は V9.40P/68 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/67 版以前の場合は、プラグインソフトをインストールする前に V9.40P/68 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.21 top.frpl_utl.getRegister

top.frpl_utl.getRegister

概要		
指定したレジスタ番号の値を取得します。		
関数名		
frpl_utl.getRegister		
引数		
変数名	**型**	**説明**
type	String / Number	取得するレジスタの種類 1: 数値レジスタ 2: 位置レジスタ 3: 文字列レジスタ
index	String / Number	取得するレジスタ番号
戻り値		
変数名	**型**	**説明**
promise	Promise	PromiseResult として respObj を返します。respObj の詳細は[詳細: 戻り値(respObj)]を参照してください。 PromiseState は下記の通りです。 • resolve: レジスタ値の取得の正常終了 • reject: 取得失敗などのエラー発生
詳細: 戻り値 (respObj)		
変数名	**型**	**説明**
status	String	結果のステータスを返します。 ステータスの詳細は本節の[詳細: Status 一覧]を参照してください。
errorDescription	String	reject 理由のメッセージ resolve の場合は null を返します。 メッセージの詳細は本節の[詳細: Status 一覧]を参照してください。
regObj	Object	取得結果の内容が入ったオブジェクト reject の場合は null を返します。

詳細: 戻り値 (respObj.regObj)		
変数名	**型**	**説明**
type	Number	取得したレジスタの種類
index	Number	取得したレジスタ番号
comment	String	取得したレジスタ番号のコメント
value	String	取得した数値レジスタの値
	String	取得した文字列レジスタの値
	Object	取得した位置レジスタの値 • 全グループ数分取得します。 • 位置記憶されていない場合は Uninitialized を返します。 • value の詳細な指定方法は本節の[詳細: 位置レジスタデータ (value.gX)]を参照してください。
詳細: 位置レジスタデータ (value)		
変数名	**型**	**説明**
totalgroup	Number	グループの総数
詳細: 位置レジスタデータ (value.gX) (X はグループ番号)		
変数名	**型**	**説明**
group	Number	取得したグループ番号
rep	Number	取得したデータの形式 1: 直交 2: 各軸
axis	Number	ロボットの軸数(直交の場合は 0)
extaxis	Number	付加軸の軸数
config	String	• 直交座標の場合は形態を指定 (例) “N U T, 0, 0, 0” • 各軸座標の場合は指定不要
x,y,z,w,p,r	String	取得した各直交座標値
j1-j6 ext1-ext3	String	• 各軸と付加軸の座標値を設定 • 付加軸の指定がない場合は ext1-ext3 は存在しません

例: 位置レジスタデータ

```
Value :{
  totalgroup:4,
  g1:{
    group:1,axis:0,extaxis:0,
    rep:”1”, config: “N U T,0,0,0”
    x:”1.00”, y :”2.00”, z:”3.00”,   w:”1.01”, p :”2.20”, e:”3.003”
  },
  g2:{
    group:2,axis:0,extaxis:3,
    rep:”1”, config: “N U T,0,0,0”
    x:"1.00", y :"2.00", z:"3.00",   w:"1.01", p :"2.20", r:"3.003,
    ext1: "1.00", ext2: "1.00", ext3: "1.00",
  },
  g3:{
    group:3,axis:6,extaxis:0,
    rep:”2”,
    j1: "1.00", j2: "1.00", j3: "1.00", j4: "1.00", j5: "1.00", j6: "1.00",
  },
  g4:{
    group:4,axis:6,extaxis:3,
    rep:”2”,
    j1: "1.00", j2: "1.00", j3: "1.00", j4: "1.00", j5: "1.00", j6: "1.00",
    ext1: "1.00", ext2: "1.00", ext3: "1.00",
  },
}
```

詳細: Status 一覧		
status	**errorDescription**	**説明**
0	Null	成功の場合
1	ValueError	不正な値が指定してある場合(解析不可)
2	WriteError	ロボット制御装置への書き込みに失敗した場合
3	TypeError: Argument (引数番号(パラメータ名)) is not of type (正しい型).	引数の型エラー
4	TypeError: Argument (引数番号(パラメータ名)) is not 1-3.	存在しないレジスタのタイプが指定された場合
5	IndexError: Argument (引数番号(パラメータ名)) does not exist.	存在しないレジスタ番号が指定された場合(設定時のみ)
6	ValueError: Argument (引数番号(パラメータ名)) cannot use (' &).	位置レジスタのコメントで「'」「&」が入力された場合
7	ValueError: Argument (引数番号(パラメータ名)) is not a number.	位置レジスタや数値レジスタで、値に数字が設定されていない場合
8	ValueError: Argument (引数番号(パラメータ名)) is incorrect configuration.	不正な形態が設定されていた場合
9	ValueError: Argument (引数番号(パラメータ名)) is incorrect rep.	直交/各軸の指定が正しくない場合
10	ValueError: Argument (引数番号(パラメータ名)) is incorrect number of axes.	不正な軸数を設定しようとした場合

⚠ 注意
top.frpl_utl.getRegister および top.frpl_utl.setRegister は V9.40P/61 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/60 版以前の場合は、プラグインソフトをインストールする前に V9.40P/61 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.22 top.frpl_utl.setRegister

top.frpl_utl.setRegister

概要		
指定したレジスタ番号の値を設定します。		
関数名		
frpl_utl.setRegister		
引数		
変数名	**型**	**説明**
regObj	String / Number	設定したいレジスタ情報をまとめたオブジェクト regObj の詳細は[詳細: 引数(regObj)]を参照してください。

戻り値		
変数名	**型**	**説明**
promise	Promise	PromiseResult として respObj を返します。respObj の詳細は[詳細: 戻り値(respObj)]を参照してください。 PromiseState は下記の通りです。 • resolve: レジスタ値の設定の正常終了 • reject: 設定失敗などのエラー発生
詳細: 引数 (regObj)		
変数名	**型**	**説明**
type	Number	設定したいレジスタの種類
index	Number	設定したいレジスタ番号
comment	String	設定したいレジスタ番号のコメント
value	String	設定したい数値レジスタの値
	String	設定したい文字列レジスタの値
	Object	設定したい位置レジスタの値 • 設定したいグループ数分指定可能 • value の詳細な指定方法は本節の[詳細: 位置レジスタデータ (value.gX)]を参照してください。
詳細: 戻り値 (respObj)		
変数名	**型**	**説明**
status	String	結果のステータスを返します。 status の詳細は前節の[詳細: Status 一覧]を参照してください。
errorDescription	String	reject 理由のメッセージ resolve の場合は null を返します。 メッセージの詳細は前節の[詳細: Status 一覧]を参照してください。
詳細: 位置レジスタデータ (value.gX) (X はグループ番号)		
変数名	**型**	**説明**
rep	Number	設定する位置データ形式 1: 直交 2: 各軸
config	String	• 直交座標の場合は形態を指定 (例) “N U T, 0, 0, 0” • 各軸座標の場合は指定不要
x,y,z,w,p,r	String/Number	各直交座標値を設定
j1-j6 ext1-ext3	String/Number	各軸と付加軸の座標値を設定 • 付加軸の指定がない場合は ext1-ext3 は不要です。 • unit(単位)の指定は不要 • ロボットの設定と異なる軸数を設定した場合は書き込みに失敗するため、top.frpl_utl.getRegister()にて取得可能なロボットの軸数(regObj.value.gX.axes)、付加軸数(regObj.value.gX.extaxes)を参照し、データを作成してください。

例: 位置レジスタデータ

```
Value :{
  g1:{
    rep:”1”, config: “N U T,0,0,0”
    x:”1.00”, y :”2.00”, z:”3.00”,   w:”1.01”, p :”2.20”, e:”3.003”
  },
  g3:{
    rep:”1”, config: “N U T,0,0,0”
    x:"1.00", y :"2.00", z:"3.00",   w:"1.01", p :"2.20", r:"3.003,
    ext1: "1.00", ext2: "1.00", ext3: "1.00",
  },
  g4:{
    rep:”2”,
    j1: "1.00", j2: "1.00", j3: "1.00", j4: "1.00", j5: "1.00", j6: "1.00",
  },
  g5:{
```

```
      rep:”2”,
      j1: "1.00", j2: "1.00", j3: "1.00", j4: "1.00", j5: "1.00", j6: "1.00",
      ext1: "1.00", ext2: "1.00", ext3: "1.00",
   },
}
```

⚠ 注意
top.frpl_utl.getRegister および top.frpl_utl.setRegister は V9.40P/61 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/60 版以前の場合は、プラグインソフトをインストールする前に V9.40P/61 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.23 top.frpl.getIO

top.frpl.getIO

概要		
指定された I/O 値を取得します。		
関数名		
frpl.getIO(ioType, point)		
引数		
変数名	**型**	**説明**
ioType	IOType	取得する I/O のタイプ
point	Number	取得する I/O のインデックス番号
戻り値		
変数名	**型**	**説明**
promise	IOPoint	I/O のデータを持つ IOPoint 型の promise を返します。
例: API の呼び出し		

DO[5]の値の取得

- let a1 = await top.frpl.getIO(IOType.DO, 5);

“IOType.DO”の詳細については後述の 7.6.2 節を参照してください。

例: データの返り値

```
a1 = {
   point: 5,
   tag: "DO[5]",
   type: 2,
   value: 0
}
// a1.value = 0
```

⚠ 注意
top.frpl.getIO は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.24 top.frpl.setIO

top.frpl.setIO

概要
指定した I/O タイプに値を設定します。
関数名
frpl.setIO(ioType, point, value)

引数		
変数名	**変数名**	**変数名**
ioType	IOType	設定する I/O のタイプ
point	Number	設定する I/O のインデックス番号
Value	Number	指定した I/O に設定する値
戻り値		
変数名	**変数名**	**変数名**
N/A	Status	リクエストのステータス情報を持つ Status 型の promise を返します。

例: API の呼び出し

DO[5]を ON に設定

- let a1 = await top.frpl.setIO(IOType.DO, 5, 1);

AO[7]の値を 33 に設定

- let a2 = top.frpl.setIO(IOType.AO, 7, 33);

“IOType.DO”と“IOType.AO”の詳細については後述の 7.6.2 節を参照してください。

例: データの返り値

```
a1 = {
   errorCode: "",
   errorDescription: "",
   state: true
}
// a1.state = true
a2 = {
      errorCode: "",
      errorDescription: "",
      state: true
   }
}
// a2.state = true
```

> ⚠ **注意**
> top.frpl.setIO は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.25 top.frpl.startIOMonitor

top.frpl.startIOMonitor

概要
指定された I/O について I/O モニタを開始します。 ロボット制御装置が I/O 値の変化を検出した場合、Callback 関数を起動してその値を通知します。
関数名
frpl.startIOMonitor(ioType, point, clientData, callback)

引数		
変数名	**型**	**説明**
ioType	IOType	監視する I/O のタイプ
point	Number	変更を監視する I/O のインデックス番号
clientData	Any	監視中の I/O が変化した際にクライアントに渡すデータ
callback	IIOHandler	監視通知時に呼び出す Callback 関数。 Callback 関数は OnIOArgs 型のパラメータを受け取ります。
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータスを持つ Status 型の promise を返します。

引数 (callback)

変数名	型	説明
args	OnIOArgs	Callback 関数は OnIOArgs 型のパラメータを受け取ります。

例: API の呼び出し

```
// * コールバック関数の振る舞いについては、この表の最終行を参照してください
function callback(args) {
   let myOutput = args.ioPoint.value;
   if (myOutput == 1) {
      if(args.clientData.isIOEnabled !== undefined) {
         args.clientData.isIOEnabled = true;
      }
   }
   else {
      if(args.clientData.isIOEnabled !== undefined) {
         args.clientData.isIOEnabled = false;
      }
   }
   var foo = args.clientData;
}
let simpleStrVar = "Hello there";
let a1 = await top.frpl.startIOMonitor(IOType.DO, 3, simpleStrVar, callback);
let tempObject = {
     myStr = “this is complexVar”,
     isIOEnabled: null
}
let a2 = await top.frpl.startIOMonitor(IOType.DO, 5, tempObject, callback);
```

“IOType.DO”の詳細については後述の 7.6.2 節を参照してください。
注: モニタの開始と終了には同じ Callback 関数を指定する必要があります。

例: データの返り値

```
a1 = {
   errorCode: "",
   errorDescription: "",
   state: true
}
a2 = {
   errorCode: "",
   errorDescription: "",
   state: true
}
```

例: Callback 関数

I/O モニタを開始した後に DO[3]=ON に変化すると、”a1”について以下の出力が生成されます。

```
      foo = “Hello there”
```

さらに DO[3]=OFF に変化すると、”a1”について以下の出力が生成されます。

```
      foo = “Hello there”
```

I/O モニタを開始した後に DO[5]=ON に変化すると、”a2”について以下の出力が生成されます。

```
   foo = {
      myStr = “this is complexVar”,
      isIOEnabled: true
   }
```

さらに DO[5]=OFF に変化すると、”a2”について以下の出力が生成されます。

```
   foo = {
      myStr = “this is complexVar”,
      isIOEnabled: false
```

```
}
```

⚠ **注意**
top.frpl.startIOMonitor は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.26 top.frpl.stopIOMonitor

top.frpl.stopIOMonitor

概要

既に開始されている所定の I/O についての I/O モニタを停止します。
モニタは既に登録されている Callback 関数を削除することによって停止されます。

関数名

frpl.stopIOMonitor(ioType, point, callback)

引数

変数名	型	説明
ioType	IOType	監視する I/O のタイプ
point	Number	変更を監視する I/O のインデックス番号
callback	IIOHandler	モニタ開始時に登録した Callback 関数。 Callback 関数は OnIOArgs 型のパラメータを受け取ります。

戻り値

変数名	型	説明
promise	Status	リクエストのステータスを持つ Status 型の promise を返します。

例: API の呼び出し

```
let a1 = await top.frpl.stopIOMonitor(IOType.DO, 3, callback);
let a2 = await top.frpl.stopIOMonitor(IOType.DO, 5, callback);
```

“IOType.DO”の詳細については後述の 7.6.2 節を参照してください。
モニタの開始と終了には同じ Callback 関数を指定する必要があります。

例: データの返り値

```
a1 = {
   errorCode: "",
   errorDescription: "",
   state: true
}
a2 = {
   errorCode: "",
   errorDescription: "",
   state: true
}
```

⚠ **注意**
top.frpl.stopIOMonitor は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.27 top.frpl.getJointPos

top.frpl.getJointPos

概要

指定したグループの各軸位置を取得します。

関数名		
frpl.getJointPos(groupNum)		
引数		
変数名	**型**	**説明**
groupNum	Number	各軸データを取得する動作グループ番号
戻り値		
変数名	**型**	**説明**
promise	RobotJointAngles	各軸データを持つ RobotJointAngles 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.getJointPos(2); // position of group-2		
例: データの返り値		

a1 = {
extended: [5.092478, 7.092478],
groupNum:2,
joints: [25.965131, 5.092478, -0.081807, -1.589457, -92.658141, 8.724838]
}
//a1.groupNum = 2		

> ⚠ **注意**
> top.frpl.getJointPos は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.28 top.frpl.startJointPosMonitor

top.frpl.startJointPosMonitor

概要		
ロボット軸の各軸位置のモニタを開始します。 ロボット制御装置が各軸位置の変化を検出した場合、Callback 関数を起動してその値を通知します。		
関数名		
frpl.startJointPosMonitor(clientData, callback)		
引数		
変数名	**型**	**説明**
clientData	Any	Callback 関数が呼ばれた際にクライアントに渡すデータ
callback	IJointPosHandler	ロボット各軸位置が変動した際に呼び出す Callback 関数
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータスを持つ Status 型の promise を返します。
引数 (callback)		
変数名	**型**	**説明**
args	OnJointPosChange	Callback 関数は OnJointPosChange 型のパラメータを受け取ります。
例: API の呼び出し		

function callback(args) {
let groupNum = 1;
let items = args.get(groupNum);
let posData = items;
let foo = args.clientData;
}
let a1 = await top.frpl.startJointPosMonitor("foo_data_exchange", callback);
 	
注: モニタの開始と終了には同じ Callback 関数を指定する必要があります。		

例: データの返り値		

```
a1 = {
   errorCode: "",
   errorDescription: "",
   state: true
}
```

例: Callback 関数

```
posData = [
   {'J1': '40.782', 'J1_units': 'deg'},
   {'J2': '1.978', 'J2_units': 'deg'},
   {'J3': '1.014', 'J3_units': 'deg'},
   {'J4': '-2.581', 'J4_units': 'deg'},
   {'J5': '-93.803', 'J5_units': 'deg'},
   {'J6': '-6.188', 'J6_units': 'deg'}
]
foo = "foo_data_exchange";
```

⚠ **注意**
top.frpl.startJointPosMonitor は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.29 top.frpl.stopJointPosMonitor

top.frpl.stopJointPosMonitor

概要

登録済みの各軸位置モニタを停止します。
モニタは既に登録されている Callback 関数を削除することによって停止されます。

関数名

frpl.stopJointPosMonitor(callback)

引数 (callback)

変数名	型	説明
callback	IJointPosHandler	モニタ開始時に登録した Callback 関数

戻り値

変数名	型	説明
promise	Status	リクエストのステータスを持つ Status 型の promise を返します。

例: API の呼び出し

```
let a1 = await top.frpl.stopJointPosMonitor(callback);
```

注: モニタの開始と終了には同じ Callback 関数を指定する必要があります。

例: データの返り値

```
a1 = {
   errorCode: "",
   errorDescription: "",
   state: true
}
```

⚠ **注意**
top.frpl.stopJointPosMonitor は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.30 top.frpl.getWorldPos

top.frpl.getWorldPos

概要		
指定した動作グループのワールド座標上の直交位置を取得します。		
関数名		
frpl.getWorldPos(groupNum)		
引数		
変数名	**型**	**説明**
groupNum	Number	ワールド座標上の直交位置を取得する動作グループ番号
戻り値		
変数名	**型**	**説明**
promise	RobotXyzWprPos	ワールド座標位置データを持つ RobotXyzWprPos 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.getWorldPos(1); //動作グループ 1 の位置データ		
例: データの返り値		

a1 = {
config: 'N U T, 0, 0, 0',
groupNum: 1,
x: 585.411987, y: 123.238007, z: 373.181213, p: 2.465889, r: 34.768513, w: -178.011581,
E1: 0, E2: 0, E3: 0,
}
// a1.config = 'N U T, 0, 0, 0';		

⚠ **注意**
top.frpl.getWorldPos は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.31 top.frpl.getWorldPosWithUF

top.frpl.getWorldPosWithUF

概要		
指定したグループの現在選択されているユーザ座標系の直交位置を取得します。		
関数名		
frpl.getWorldPosWithUF(groupNum)		
引数		
変数名	**型**	**説明**
groupNum	Number	ユーザ座標系位置を取得する動作グループ番号
戻り値		
変数名	**型**	**説明**
promise	RobotXyzWprPos	ユーザ座標位置データを持つ RobotXyzWprPos 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.getWorldPosWithUF(1); // 動作グループ 1 の位置データ		
例: データの返り値		
a1 = { config: "N U T, 0, 0, 0", groupNum: 1, x: 585.411987, y: 123.238007, z: 373.181213, p: 2.465889, r: 34.768513, w: -178.011581, E1: 0, E2: 0, E3: 0		

```
}
// a1.groupNum = 1
```

⚠ **注意**
top.frpl.getWorldPosWithUF は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.32 top.frpl.startWorldPosMonitor

top.frpl.startWorldPosMonitor

概要

ロボットのワールド座標系から見たツール先端点位置(TCP)のモニタを開始します。
ロボット制御装置がワールド座標位置の変化を検出した場合、Callback 関数を起動してその値を通知します。

関数名

frpl.startWorldPosMonitor(clientData, callback)

引数

変数名	型	説明
clientData	Any	Callback 関数が呼ばれた際にクライアントに渡すデータ
callback	IWorldPosHandler	ロボットの TCP 位置が変動した際に呼び出す Callback 関数

戻り値

変数名	型	説明
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。

引数 (callback)

変数名	型	説明
args	OnWorldPosChange	Callback 関数は OnWorldPosChange 型のパラメータを受け取ります。

例: API の呼び出し

```
function worldPosCallback(args) {
      let groupNum = 1;
      let items = args.get(groupNum);
      let posData = items;
      let foo = args.clientData;
}
let a1 = await top.frpl.startWorldPosMonitor("foo_data_exchange", callback);
```

注: モニタの開始と終了には同じ Callback 関数を指定する必要があります。

例: データの返り値

```
a1 = {state: true, errorCode: '', errorDescription: ''}
```

例: Callback 関数

```
posData = [
   {'C' : 'N U T, 0, 0, 0'},
   {'X' : '510.314', 'X_units' : 'mm'},
   {'Y' : '251.804', 'Y_units' : 'mm'},
   {'Z' : '382.835', 'Z_units' : 'mm'},
   {'W' : '-177.732', 'W_units' : 'deg'},
   {'P' : '3.048', 'P_units' : 'deg'},
   {'R' : '34.763', 'R_units' : 'deg'}
]
foo = "foo_data_exchange"
```

> ⚠ **注意**
> top.frpl.startWorldPosMonitor は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.33 top.frpl.stopWorldPosMonitor

top.frpl.stopWorldPosMonitor

概要		
ロボットのワールド座標系から見たツール先端点位置(TCP)のモニタを停止します。 モニタは既に登録されている Callback 関数を削除することによって停止されます。		
関数名		
frpl.stopWorldPosMonitor(callback)		
引数		
変数名	**型**	**説明**
callback	IWorldPosHandler	モニタ開始時に登録した Callback 関数
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータスを持つ Status 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.stopWorldPosMonitor(callback); **注**: モニタの開始と終了には同じ Callback 関数を指定する必要があります。		
例: データの返り値		
a1 = {state: true, errorCode: '', errorDescription: ''}		

> ⚠ **注意**
> top.frpl.stopWorldPosMonitor は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.34 top.frpl.getVariable

top.frpl.getVariable

概要		
KAREL 変数を取得します。		
関数名		
frpl.getVariable(programName, varName)		
引数		
変数名	**型**	**説明**
programName	String	取得する変数を持つ KAREL プログラム名
varName	String	取得する KAREL 変数名
戻り値		
変数名	**型**	**説明**
promise	RobotVarInfo	変数データを持つ RobotVarInfo 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.getVariable(“prog_1”, “my_var”);		
例: データの返り値		
a1 = {variableName: 'my_var', programName: 'prog_1', value: 'hello', tag: 'prog_1+my_var'}		

> ⚠ **注意**
> top.frpl.getVariable は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.35 top.frpl.setVariable

top.frpl.setVariable

概要		
ロボット制御装置の KAREL 変数を設定します。		
関数名		
frpl.setVariable(programName, varName, value)		
引数		
変数名	**型**	**説明**
programName	String	設定する変数を持つ KAREL プログラム名
varName	String	設定する KAREL 変数名
value	Any	設定する値
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータスを持つ Status 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.setVariable(“prog_1”, “my_var”, “test_string”);		
例: データの返り値		
a1 = {state: true, errorCode: ' ', errorDescription: ' '}		

> ⚠ **注意**
> top.frpl.setVariable は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.36 top.frpl.startVarMonitor

top.frpl.startVarMonitor

概要		
KAREL 変数のモニタを開始します。 監視する KAREL 変数の値の変化を検出した場合、Callback 関数を起動してその値を通知します。		
関数名		
frpl.startVarMonitor(programName, varName, clientData, callback)		
引数		
変数名	**型**	**説明**
programName	String	監視する変数を持つ KAREL プログラム
varName	String	監視する KAREL 変数名
clientData	Any	Callback 関数が呼ばれた際にクライアントに渡すデータ
callback	IVarHandler<RobotVarInfo>	変数データが変動した際に呼び出す Callback 関数
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。
引数 (callback)		
変数名	**型**	**説明**

args	OnVarArgs<RobotVarInfo>	Callback 関数は OnVarArgs<RobotVarInfo>型のパラメータを受け取ります。

例: API の呼び出し

```
function callback(args) {
      let foo_1 = args;
      let foo_2 = args.varInfo.value;
}
let a1 = await top.frpl.startVarMonitor("myprog", "my_var", "foo_data_exchange", varCallback);
```

注: モニタの開始と終了には同じ Callback 関数を指定する必要があります。

例: データの返り値

```
a1 = {state: true, errorCode: '', errorDescription: ''}
```

例: Callback 関数

```
foo_1 = {
   clientData: "foo_data_exchange",
   varInfo: {
      programName: "myprog",
      variableName: "my_var"
      value: "10",
      tag: "myprog +my_var",
   }
}
foo_2 = 10;
```

⚠ **注意**
top.frpl.startVarMonitor は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.37 top.frpl.stopVarMonitor

top.frpl.stopVarMonitor

概要

登録済みの KAREL 変数モニタを停止します。
モニタは既に登録されている Callback 関数を削除することによって停止されます。

関数名

frpl.stopVarMonitor(programName, varName, callback)

引数

変数名	型	説明
programName	String	監視を停止する変数を持つ KAREL プログラム名
varName	String	監視を停止する KAREL 変数名
callback	IVarHandler<RobotVarInfo>	モニタ開始時に登録した Callback 関数

戻り値

変数名	型	説明
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。

例: API の呼び出し

```
let a1 = await top.frpl.stopVarMonitor("myprog", "my_var", callback);
```

注: モニタの開始と終了には同じ Callback 関数を指定する必要があります。

例: データの返り値

```
a1 = {state: true, errorCode: '', errorDescription: ''}
```

> ⚠ **注意**
> top.frpl.stopVarMonitor は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.38 top.frpl.runProgram

top.frpl.runProgram

概要		
TP プログラムを実行します。		
関数名		
frpl.runProgram(programName, startAtLine)		
引数		
変数名	**型**	**説明**
programName	String	実行する TP プログラム名
startAtLine	Number	プログラム実行を開始する行番号(標準: 1)
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.runProgram(“myprog”); let failCase = await top.frpl.runProgram(“myprog”);		
例: データの返り値		
a1 = {state: true, errorCode: '', errorDescription: ''} failCase = {state: false, errorCode: '0x30007', errorDescription: 'PROG-007 Program is already running'}		

> ⚠ **注意**
> top.frpl.runProgram は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.39 top.frpl.pauseProgram

top.frpl.pauseProgram

概要		
プログラムを一時停止します。		
関数名		
frpl.pauseProgram(programName)		
引数		
変数名	**型**	**説明**
programName	String	一時停止する TP プログラム名
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.pauseProgram("myprog");		
例: データの返り値		
a1 = {state: true, errorCode: '', errorDescription: ''}		

> ⚠ **注意**
> top.frpl.pauseProgram は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.40 top.frpl.resumeProgram

top.frpl.resumeProgram

概要		
TP プログラムを再開します。		
関数名		
frpl.resumeProgram(programName)		
引数		
変数名	**型**	**説明**
programName	String	再開する TP プログラム名
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.resumeProgram("myprog");		
例: データの返り値		
a1 = {state: true, errorCode: '', errorDescription: ''}		

> ⚠ **注意**
> top.frpl.resumeProgram は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.41 top.frpl.abortProgram

top.frpl.abortProgram

概要		
TP プログラムを中断します。		
関数名		
frpl.abortProgram(programName)		
引数		
変数名	**型**	**説明**
programName	String	中断する TP プログラム名
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.abortProgram("myprog");		
例: データの返り値		
a1 = {state: true, errorCode: ' ', errorDescription: ' '}		

> ⚠ **注意**
> この API は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.42 top.frpl.abortAllPrograms

top.frpl.abortAllPrograms

概要		
全ての TP プログラムを中断します。		
関数名		
frpl.abortAllPrograms()		
引数		
変数名	**型**	**説明**
N/A	N/A	N/A
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.abortAllPrograms();		
例: データの返り値		
a1 = {state: true, errorCode: ' ', errorDescription: ' '}		

> ⚠ **注意**
> top.frpl.abortAllPrograms は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.43 top.frpl.readProgramToApi

top.frpl.readProgramToApi

概要		
TP プログラムを API に読み込みます。		
関数名		
frpl.readProgramToApi(programName)		
引数		
変数名	**型**	**説明**
programName	String	読み込む TP プログラムの名前
戻り値		
変数名	**型**	**説明**

promise	{
	status:Status,
	data:String
	}
		'status'と'data'のキーを持つ JSON 型オブジェクトの promise を返します。'status'のキーは成功/失敗のステータスと、あればエラーの説明の情報が格納されます。'data'のキーには、リクエストされたプログラムの全てのテキストを含む文字列が格納されます。

例: API の呼び出し		
let a1 = await top.frpl.readProgramToAPI("myprog");		
例: データの返り値		
a1 = { data: "/PROG MYPROG /ATTR OWNER = MNEDITOR;		

```
        COMMENT             = "";
        PROG_SIZE           = 244;
        CREATE              = DATE 25-02-11   TIME 18:16:46;
        MODIFIED  = DATE 25-02-11   TIME 18:17:24;
        FILE_NAME           = ;
        VERSION             = 0;
        LINE_COUNT          = 4;
        MEMORY_SIZE         = 732;
        PROTECT             = READ_WRITE;
        TCD:   STACK_SIZE   = 0,
               TASK_PRIORITY          = 50,
               TIME_SLICE   = 0,
               BUSY_LAMP_OFF          = 0,
               ABORT_REQUEST          = 0,
               PAUSE_REQUEST          = 0;
        DEFAULT_GROUP    = 1,*,*,*,*;
        CONTROL_CODE     = 00000000 00000000;
        LOCAL_REGISTERS  = 0,0,0;
        /APPL

        AUTO_SINGULARITY_HEADER;
          ENABLE_SINGULARITY_AVOIDANCE    : FALSE;

        PLTZ_MODE_HEADER;
         PLTZ_MODE_ENABLE    : FALSE;
         J4TURN   : ZERORAD;
         ORIENT   : DOWNWARDS;
        /MN
           1:   LBL[1] ;
           2:   R[10]=45      ;
           3:   WAIT     2.00(sec) ;
           4:   JMP LBL[1] ;
        /POS
        /END
        ",
  status: {state: true, errorCode: '', errorDescription: ''}
}
```

> ⚠ **注意**
> top.frpl.readProgramToAPI は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.44 top.frpl.listTpPrograms

top.frpl.listTpPrograms

概要		
ロボット制御装置内の全ての TP プログラムの名前の一覧を取得します。		
関数名		
frpl.listTpPrograms()		
引数		
変数名	**型**	**説明**
N/A	N/A	N/A

戻り値		
変数名	**型**	**説明**

promise	{
	status:Status,
	data:String[ ]
	}
		'status'と'data'のキーを持つ JSON 型オブジェクトの promise を返します。'status'のキーは成功/失敗のステータスと、あればエラーの説明の情報が格納されます。'data'のキーには、ロボット制御装置内の全ての TP プログラムの名前の一覧/配列が格納されます。

例: API の呼び出し		
let a1 = await top.frpl.listTpPrograms();		
例: データの返り値		
a1 = { data: ['DEFAULT', 'MOTN_BKWD', 'MOTN_FWD', 'SRI', 'TEST', 'TEST_1', 'TRANSPORT'], status: {state: true, errorCode: '', errorDescription: ''} }		

⚠ **注意**
top.frpl.listTpPrograms は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.45 top.frpl.deleteProgram

top.frpl.deleteProgram

概要		
TP プログラムをロボット制御装置から削除します。		
関数名		
frpl.deleteProgram(programName)		
引数		
変数名	**型**	**説明**
programName	String	削除する TP プログラム名
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.deleteProgram();		
例: データの返り値		
a1 = {state: true, errorCode: '', errorDescription: ''};		

⚠ **注意**
top.frpl.deleteProgram は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.46 top.frpl.uploadProgram

top.frpl.uploadProgram

概要
TP プログラムをロボット制御装置にアップロードします。任意のパラメータを渡す場合は'programData'に空文字(' ')を指定します。
関数名
frpl.uploadProgram(programName, programData, programCommands, positionData, clientHeaders)

引数

変数名	型	説明
programName	String	アップロードする TP プログラムの名前
programData	String	アップロードする TP プログラムのデータ
programCommands	String	(任意のパラメータ)プログラムに渡されるコマンド/コード
positionData	String	(任意のパラメータ)プログラムの位置データ
clientHeaders	{ key:String, value:String } の型を持つ JSON オブジェクト	(任意のパラメータ){ key:String, value:String }のペアを持つオブジェクトとして渡されるヘッダのデータ

戻り値

変数名	型	説明
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。

例: API の呼び出し

以下に示す LS プログラム例で API を説明します。

```
/PROG  test_1
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "";
PROG_SIZE     = 572;
CREATE                = DATE 25-04-29  TIME 16:08:54;
MODIFIED      = DATE 25-04-29  TIME 16:12:44;
FILE_NAME     = ;
VERSION               = 0;
LINE_COUNT    = 2;
MEMORY_SIZE  = 936;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE      = 0,
      TASK_PRIORITY   = 50,
      TIME_SLICE      = 0,
      BUSY_LAMP_OFF   = 0,
      ABORT_REQUEST   = 0,
      PAUSE_REQUEST   = 0;
DEFAULT_GROUP         = 1,*,*,*,*;
CONTROL_CODE          = 00000000 00000000;
LOCAL_REGISTERS       = 0,0,0;
/APPL

AUTO_SINGULARITY_HEADER;
  ENABLE_SINGULARITY_AVOIDANCE    : FALSE;

PLTZ_MODE_HEADER;
 PLTZ_MODE_ENABLE    : FALSE;
 J4TURN   : ZERORAD;
 ORIENT   : DOWNWARDS;
/MN
   1:J P[2] 100% FINE    ;
   2:J P[1] 100% FINE    ;
/POS
P[1]{
   GP1:
   UF : 0, UT : 1,            CONFIG : 'N U T, 0, 0, 0',
   X =    549.423584  mm,          Y =    115.284813  mm,       Z =    392.809387  mm,
   W =   -178.404541 deg, P =       1.901237 deg,          R =      34.723373 deg
};
P[2]{
```

```
    GP1:
    UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
    X =    718.013184  mm,          Y =    237.137070  mm,       Z =    237.881500  mm,
    W =   -178.404510 deg, P =       1.901244 deg,          R =      34.723370 deg
};
/END
```

以下に示す 2 つの使用例では、API を使用して上記 LS プログラムのロボット制御装置へのアップロードを行います。

例 1: 全てのデータを渡してプログラムをアップロードする

以下の例では TP プログラムをアップロードする基本的な方法を示します。この方法では 2 つの変数を使用します。1 つ目の変数(prog_name)は STRING データでプログラム名を保持します。2 つ目の変数(prog_data)は STRING データで保存された LS ファイルの内容全てを保持します。

```
let prog_name = `test_1`
let prog_data = `
/PROG  test_1
/ATTR
OWNER                   = MNEDITOR;
COMMENT                 = "";
PROG_SIZE       = 572;
CREATE                  = DATE 25-04-29  TIME 16:08:54;
MODIFIED        = DATE 25-04-29  TIME 16:12:44;
FILE_NAME       = ;
VERSION                 = 0;
LINE_COUNT      = 2;
MEMORY_SIZE  = 936;
PROTECT                 = READ_WRITE;
TCD:   STACK_SIZE       = 0,
       TASK_PRIORITY    = 50,
       TIME_SLICE       = 0,
       BUSY_LAMP_OFF    = 0,
       ABORT_REQUEST    = 0,
       PAUSE_REQUEST    = 0;
DEFAULT_GROUP           = 1,*,*,*,*;
CONTROL_CODE            = 00000000 00000000;
LOCAL_REGISTERS         = 0,0,0;
/APPL

AUTO_SINGULARITY_HEADER;
  ENABLE_SINGULARITY_AVOIDANCE    : FALSE;

PLTZ_MODE_HEADER;
 PLTZ_MODE_ENABLE    : FALSE;
 J4TURN   : ZERORAD;
 ORIENT   : DOWNWARDS;
/MN
    1:J P[2] 100% FINE     ;
    2:J P[1] 100% FINE     ;
/POS
P[1]{
    GP1:
    UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
    X =    549.423584  mm,          Y =    115.284813  mm,       Z =    392.809387  mm,
    W =   -178.404541 deg, P =       1.901237 deg,          R =      34.723373 deg
};
P[2]{
```

```
    GP1:
    UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
    X =    718.013184  mm,          Y =    237.137070  mm,      Z =    237.881500  mm,
    W =   -178.404510 deg, P =      1.901244 deg,        R =     34.723370 deg
};
/END
`; // need this new line here
let a1 = await top.frpl.uploadProgram(prog_name, prog_data);
```

例 2: 位置データと TP プログラム命令を含むプログラムをアップロードする

以下の例ではユーザがプログラムの内容を選択できる TP プログラムをアップロードする方法を示します。この方法では、TP プログラム命令と位置データのみ使用されます。全てのデータは各変数に保存されています。

注: 任意のパラメータを使用する場合、API の第二引数(programData)に空文字を指定する必要があります。

```
let prog_name = `test_2`;
let prog_data = '';
let prog_commands = `1:J P[2] 100% FINE ;
    2:J P[1] 100% FINE   ;`
let pos_data = ` P[1]{
    GP1:
    UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
    X =    549.423584  mm,          Y =    115.284813  mm,      Z =    392.809387  mm,
    W =   -178.404541 deg, P =      1.901237 deg,        R =     34.723373 deg
};
P[2]{
    GP1:
    UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
    X =    718.013184  mm,          Y =    237.137070  mm,      Z =    237.881500  mm,
    W =   -178.404510 deg, P =      1.901244 deg,        R =     34.723370 deg
};`
let client_headers = "";
let a2 = await top.frpl.uploadProgram(prog_name, prog_data, prog_commands, pos_data, client_headers);
```

例: データの返り値

```
a1 = {state: true, errorCode: '', errorDescription: ''}
a2 = {state: true, errorCode: '', errorDescription: ''}
```

⚠ **注意**

top.frpl.uploadProgram は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.47 top.frpl.getCurrentProgramName

top.frpl.getCurrentProgramName

概要		
現在選択中または実行中のプログラムの名前を取得します。		
関数名		
frpl.getCurrentProgramName()		
引数		
変数名	**型**	**説明**
N/A	N/A	N/A

戻り値		
変数名	**型**	**説明**
promise	String	文字列データを持つ promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.getCurrentProgramName();		
例: データの返り値		
a1 = "myProgram1";		

> ⚠ **注意**
> top.frpl.getCurrentProgramName は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.48 top.frpl.getJointLimits

top.frpl.getJointLimits

概要		
ロボット軸の動作範囲を取得します。		
関数名		
frpl.getJointLimits()		
引数		
変数名	**型**	**説明**
N/A	N/A	N/A
戻り値		
変数名	**型**	**説明**
promise	RobotJointLimits	ロボット軸の動作範囲データを持つ RobotJointLimits 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.getJointLimits();		
例: データの返り値		

a1 = {
axisTypes: [1, 1, 1, 1, 1, 1],
lowerLimits: [-189.999989, -179.998994, -194.999988, -189.999989, -179.998994, -224.999996],
upperLimits: [189.999989, 179.998994, 374.99998, 189.999989, 179.998994, 224.999996]
}		

> ⚠ **注意**
> top.frpl.getJointLimits は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.49 top.frpl.getActiveUserFrameNumber

top.frpl.getActiveUserFrameNumber

概要		
指定した動作グループで現在選択されているユーザ座標(UF)番号を取得します。		
関数名		
frpl.getActiveUserFrameNumber(groupNumber)		
引数		
変数名	**型**	**説明**
groupNumber	Number	ユーザ座標番号を取得する動作グループ番号

戻り値		
変数名	**型**	**説明**
promise	Number	数値を持つ promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.getActiveUserFrameNumber(1);		
例: データの返り値		
a1 = 2		

> ⚠ **注意**
> top.frpl.getActiveUserFrameNumber は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.50 top.frpl.getActiveUserToolNumber

top.frpl.getActiveUserToolNumber

概要		
指定した動作グループで現在選択されているツール座標(UT)番号を取得します。		
関数名		
frpl.getActiveUserToolNumber(groupNumber)		
引数		
変数名	**型**	**説明**
groupNumber	Number	ツール座標番号を取得する動作グループ番号
戻り値		
変数名	**型**	**説明**
promise	Number	数値を持つ promise を返します。
例: API の呼び出し		
let a1 = top.frpl.getActiveUserToolNumber(1);		
例: データの返り値		
a1 = 2		

> ⚠ **注意**
> top.frpl.getActiveUserToolNumber は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.51 top.frpl.startActiveProgramMonitor

top.frpl.startActiveProgramMonitor

概要		
アクティブなプログラムの状態をモニタします。		
関数名		
frpl.startActiveProgramMonitor(clientData, callback)		
引数		
変数名	**型**	**説明**
clientData	Any	Callback 関数が呼ばれた際にクライアントに渡すデータ
callback	IActiveProgMonHandler	アクティブなプログラムの状態が変動した際に呼び出す Callback 関数
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。

引数 (callback)		
変数名	**型**	**説明**
args	ProgramState	Callback 関数は ProgramState 型のパラメータを受け取ります。
例: API の呼び出し		

function callback(args) {
let foo = args;
}
let a1 = top.frpl.startActiveProgramMonitor(“prog_mon_data_exchange”, callback);		

例: データの返り値		
a1 = {state: true, errorCode: '', errorDescription: ''}		
例: 出力データ		
foo = {clientData: 'prog_mon_data_exchange', programName: 'MOTN_FWD', taskName: '', executionState: 3, activeLineNumber: 0} **注:** executionState = 3 はプログラム終了状態を意味する ENUM 宣言です。詳細は 7.6.17 節を参照してください。		

⚠ 注意
top.frpl.startActiveProgramMonitor は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

注
top.frpl.startActiveProgramMonitor はデータの返り値が早く、再帰的に行われます(250msec 程度)。そのため、この API に渡されるコールバック関数も 250msec 毎に起動され、それに伴ってコールバック関数内に定義された全ての処理も 250msec 毎に実行されます。

7.5.52 top.frpl.stopActiveProgramMonitor

top.frpl.stopActiveProgramMonitor

概要		
プログラム状態モニタを停止します。		
関数名		
frpl.stopActiveProgramMonitor(callback)		
引数		
変数名	**型**	**説明**
callback	IActiveProgMonHandler	モニタ開始時に登録した Callback 関数
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。
例: API の呼び出し		

function callback(args) {
let foo = args;
}
let a1 = top.frpl.stopActiveProgramMonitor(callback);		

例: データの返り値		
a1 = {state: true, errorCode: '', errorDescription: ''}		

> ⚠ **注意**
> top.frpl.stopActiveProgramMonitor は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.53 top.frpl.stopAllMonitors

top.frpl.stopAllMonitors

概要		
全てのモニタをクリアします。		
関数名		
frpl.stopAllMonitors()		
引数		
変数名	**型**	**説明**
N/A	N/A	N/A
戻り値		
変数名	**型**	**説明**
promise	Status	リクエストのステータス情報を持つ Status 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.stopAllMonitors();		
例: データの返り値		
a1 = {state: true, errorCode: '', errorDescription: ''}		

> ⚠ **注意**
> top.frpl.stopAllMonitors は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.5.54 top.frpl.isEnabled

top.frpl.isEnabled

概要		
top.frpl の API が使用可能か確認するための関数です。		
関数名		
frpl.isEnabled ()		
引数		
変数名	**型**	**説明**
N/A	N/A	N/A
戻り値		
変数名	**型**	**説明**
promise	Boolean	リクエストのステータス情報を持つ Boolean 型の promise を返します。
例: API の呼び出し		
let a1 = await top.frpl.isEnabled();		
例: データの返り値		
a1 = true		

> ⚠ **注意**
> top.frpl.isEnabled は V9.40P/65 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 API を使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/64 版以前の場合は、プラグインソフトをインストールする前に V9.40P/65 版以降の最新版にアップデートしてください。」、と明記してください。

7.6 top.frpl 関数のデータ型

7.6.1 Status

データ型名		
Status		
プロパティ		
変数名	**型**	**説明**
state	Boolean	ロボット制御装置から返されるリクエストのステータス
errorCode	String	ロボット制御装置から返されるリクエストのエラーコード
errorDescription	String	ロボット制御装置から返されるエラーコードの説明

7.6.2 IOType

データ型名			
IOType (Enum)			
プロパティ			
変数名	**型**	**値**	**説明**
DI	Enum	1	デジタル入力
DO	Enum	2	デジタル出力
AI	Enum	3	アナログ入力
AO	Enum	4	アナログ出力
RI	Enum	8	ロボット入力
RO	Enum	9	ロボット出力
WI	Enum	16	溶接入力
WO	Enum	17	溶接出力
GI	Enum	18	グループ入力
GO	Enum	19	グループ出力

7.6.3 IOPoint

データ型名		
IOPoint		
プロパティ		
変数名	**型**	**説明**
type	IOType	I/O タイプ
point	Number	I/O のインデックス番号
value	Number	更新後の最新の I/O の値

7.6.4 IIOHandler

データ型名
IIOHandler (Interface Type)

プロパティ		
変数名	**型**	**説明**
args	OnIOArgs	I/O の情報

7.6.5 OnIOArgs

データ型名		
OnIOArgs		
プロパティ		
変数名	**型**	**説明**
ioPoint	IOPoint	変動があった IOPoint
clientData	Any	モニタ開始時に渡されたクライアントデータを返します。

7.6.6 RobotXyzWprPos

データ型名		
RobotXyzWprPos		
プロパティ		
変数名	**型**	**説明**
config	String	指定したロボットワールド座標系の形態
groupNum	Number	ワールド座標位置のグループ
x	Number	ワールド座標原点からの x 方向の距離
y	Number	ワールド座標原点からの y 方向の距離
z	Number	ワールド座標原点からの z 方向の距離
w	Number	指定グループの座標系の x 方向まわりの回転角度
p	Number	指定グループの座標系の y 方向まわりの回転角度
r	Number	指定グループの座標系の z 方向まわりの回転角度

7.6.7 AxisTypes

データ型名			
AxisTypes (Enumeration)			
プロパティ			
変数名	**型**	**値**	**説明**
linear	Enum	1	直動軸
rotary	Enum	2	回転軸

7.6.8 RobotJointLimits

データ型名		
RobotJointLimits		
プロパティ		
変数名	**型**	**説明**
axisTypes	AxisTypes[ ] [ ]	Enum の AxisTypes 型の 2 次元配列です。1 番目の配列はグループ 1, 2 番目の配列はグループ 2 のデータが格納されています。それ以降の配列番号も同様です。
lowerLimits	Number[ ] [ ]	数値の 2 次元配列です。1 番目の配列はグループ 1, 2 番目の配列はグループ 2 の軸の下限値が格納されています。それ以降の配列番号も同様です。
upperLimits	Number[ ] [ ]	数値の 2 次元配列です。1 番目の配列はグループ 1, 2 番目の配列はグループ 2 の軸の上限値が格納されています。それ以降の配列番号も同様です。

7.6.9 RobotJointAngles

データ型名		
RobotJointAngles		
プロパティ		
変数名	**型**	**説明**
groupNum	Number	リクエストされた動作グループ番号です。
joints	Number [ ]	リクエストされた動作グループの各軸データを持つ配列です。

7.6.10 OnJointPosChange

データ型名		
OnJointPosChange		
プロパティ		
変数名	**型**	**説明**
clientData	Any	モニタ開始時に設定したクライアントデータがそのまま渡されます。
関数		
定義	**戻り値**	**説明**
get (groupNum: Number)	Array (Map<string, string>)	この関数ではインプットとしてグループ番号を指定し、そのグループの各軸位置を取得できます。戻り値は以下のようなマップを含む配列となります。 [{ "J1": "24.015", "J1_units": "deg" }, { "J2": "7.101", "J2_units": "deg" }, : : { "J6": "-4.101", "J6_units": "deg" },]

7.6.11 IJointPosHandler

データ型名		
IJointPosHandler		
プロパティ		
変数名	**型**	**説明**
args	OnJointPosChange	軸位置の変動イベントのラッパークラス

7.6.12 OnWorldPosChange

データ型名		
OnWorldPosChange		
プロパティ		
変数名	**型**	**説明**
clientData	Any	モニタ開始時に設定したクライアントデータがそのまま渡されます。
関数		
定義	**戻り値**	**説明**
get (groupNum: Number)	Array (Map<string, string>)	動作グループ番号を指定し、そのグループのワールド座標位置を取得できます。戻り値は以下のようなマップを含む配列となります。 [{ "C": "N U T, 0, 0, 0" }, { "X": "1574.230", "X_units": "mm" },

		{ "Y": "701.383", "Y_units": "mm" }, : : { "R": "1.500", "R_units": "deg" }]

7.6.13 IWorldPosHandler

データ型名		
IWorldPosHandler		
プロパティ		
変数名	型	説明
args	OnWorldPosChange	座標位置の変動イベントのラッパークラス

7.6.14 RobotVarInfo

データ型名		
RobotVarInfo		
プロパティ		
変数名	型	説明
programName	String	プログラム名
variableName	String	プログラム内の変数名
value	Any	更新後の変数の最新の値

7.6.15 onVarArgs

データ型名		
onVarArgs		
プロパティ		
変数名	型	説明
varInfo	RobotVarInfo	変数の情報
clientData	Any	モニタ開始時に設定したクライアントデータがそのまま渡されます。

7.6.16 IVarHandler

データ型名		
IVarHandler		
プロパティ		
変数名	型	説明
args	onVarArgs	KAREL 変数変動イベントのラッパークラス

7.6.17 ExecutionState

データ型名		
ExecutionState		
プロパティ		
変数名	型	説明
na	0	アクティブではありません。
running	1	プログラムが実行中です。
paused	2	プログラムが一時停止中です。

aborted	3	プログラムは終了しています。

7.6.18 ProgramState

データ型名		
ProgramState		
プロパティ		
変数名	**型**	**説明**
clientData	Any	クライアントから渡されたデータ
programName	String	メインのプログラム名
taskName	String	タスク(サブプログラム)名
executionState	ExecutionState	プログラムの状態を示す Enum の値
activeLineNumber	Number	実行中の行番号

7.6.19 IActiveProgMonHandler

データ型名		
IActiveProgMonHandler		
プロパティ		
変数名	**型**	**説明**
args	ProgramState	アクティブなプログラムの変動イベントのラッパークラス

7.7 メニュー追加

作成したプラグイン画面をメニューに追加する方法について説明します。タブレット TP では、画面左上のメニューボタンからメニューを開きます。画面開発者は、表示されるプラグインメニュー内に、作成したプラグイン画面のメニューを追加できます。



図 7.7 (a) プラグイン画面のメニュー

画面開発者は、作成したプラグイン画面をメニューに追加するために、メニュー構成を記述した XML ファイルを作成する必要があります。

7.7.1 XML ファイルの作成

画面開発者は、メニュー構成を記述した LXML ファイルをビルドすることで、言語別の XML ファイルを生成できます。LXML ファイルをビルドすると、各言語分の XML ファイルが生成されます。LXML ファイルのビルド方法については、7.2 節「ビルドツール」を参照してください。

メニュー構成を記述する LXML ファイルは、必ずファイル名を PLMENU.LXML としてください。PLMENU.LXML は、プラグインソフト名を記述する<FanucMenu>タグと、メニュー項目を記述する<FanucMenuItem>タグで構成されます。

以下は、図 7.7 (a)のように、「Sample」という項目をメニューに追加する場合の記述例です。

PLMENU.LXML の記述例

```
<FanucMenu Name="company1_plugin1">
  <FanucMenuItem Header="Sample" Screen="company1_plugin1/sample.stm" />
</FanucMenu>
```

注
- 作成する LXML ファイル名は、必ず PLMENU.LXML としてください。別名で作成した場合、メニューを追加できません。
- PLMENU.LXML ファイルに記載できる<FanucMenu>タグは 1 つまでです。

各タグに必要なパラメータは以下の通りです。

<FanucMenu>タグ
・Name : 会社名_プラグイン名

<FanucMenuItem>タグ
・Header : メニュー項目名
・Screen : 作成したプラグイン画面の URL(会社名_プラグイン名/HTML ファイル名)

注
Screen パラメータに記述する画面 URL は、必ず 58 文字以内としてください(記号を含む)。58 文字を超える場合、画面を正しく表示できない場合があります。

Name パラメータおよび Header パラメータは、7.2 節「ビルドツール」で作成した辞書ファイル(utx)の要素を使用できます。辞書要素を使用して LXML ファイルを作成した場合、辞書要素で記載した箇所が、ビルド時に対応する言語に置き換わります。

以下は、Header パラメータを辞書要素で記述した場合の例です。

辞書要素の例
英語辞書： sampleeg.utx
辞書要素： $-, menu_item_c ‘Sample’

辞書要素を使用した場合の PLMENU.LXML の記述例

```
<FanucMenu Name="company1_plugin1 ">
  <FanucMenuItem Header=”{{sample.menu_item_c}}” Screen="company1_plugin1/sample.stm" />
</FanucMenu>
```

生成される XML ファイル(英語)の例

```
<FanucMenu Name="company1_plugin1 ">
  <FanucMenuItem Header=”Sample” Screen="company1_plugin1/sample.stm" />
</FanucMenu>
```

ビルドして生成された XML ファイルは、自動的にパッケージに組み込まれます。プラグインソフトをインストールすることで、記述したメニュー構成がプラグインメニューに追加されます。

注
辞書要素を使用しない場合、LXML ファイルに記述した文字列がそのまま XML ファイルに記載されます。ロボット制御装置の言語設定に連動してメニュー表示を切り替える場合は、辞書要素を用いて LXML ファイルを作成してください。

7.7.2 複数の画面をメニューに追加する方法

複数の画面をメニューに追加したい場合、<FanucMenuItem>タグを複数記載する必要があります。

複数画面を追加する場合の PLMENU.LXML の記述例

```
<FanucMenu Name="company1_plugin1 ">
    <FanucMenuItem Header="Sample1" Screen="company1_plugin1/sample1.stm" />
    <FanucMenuItem Header="Sample2" Screen="company1_plugin1/sample2.stm" />
</FanucMenu>
```



図 7.7.2 (a) 複数メニューの追加

また、FanucMenuItem タグを入れ子にすることで、セクションを分けることも可能です。

セクションを分ける場合の PLMENU.LXML の記述例

```
<FanucMenu Name="company1_plugin1 ">
  <FanucMenuItem Header=”MENU1”>
    <FanucMenuItem Header="Sample1" Screen="company1_plugin1/sample1.stm"/>
  </FanucMenuItem>
  <FanucMenuItem Header=”Menu2”>
    <FanucMenuItem Header="Sample2" Screen="company1_plugin1/sample2.stm" />
  </FanucMenuItem>
</FanucMenu>
```



図 7.7.2 (b) セクションを分けた場合のメニュー追加

7.8 ROBOGUIDE での画面の確認

作成した周辺機器専用のプラグイン画面、および、後述するプラグイン命令の詳細画面は、ROBOGUIDE 上のタブレット TP 画面で確認できます。

7.8.1 事前準備

ROBOGUIDE は ROBOGUIDE Version 9 Rev.L 以降の最新バージョンを使用してください。
作成するワークセルのロボットソフトウェアバージョンは V9.40 を選択してください。

7.8.2 タブレット TP ウインドウの使用方法

ROBOGUIDE ワークセルを開き、[タブレット TP]ボタンをクリックしてください。タブレット TP ウインドウが開き、タブレット TP と同様の操作が可能です。作成したプラグインパッケージファイル(*.ipl ファイル)をインストールして、作成した周辺機器専用のプラグイン画面、および、後述するプラグイン命令の詳細画面を確認してください。プラグインパッケージファイルのインストール方法は本資料の 3 章を参照してください。

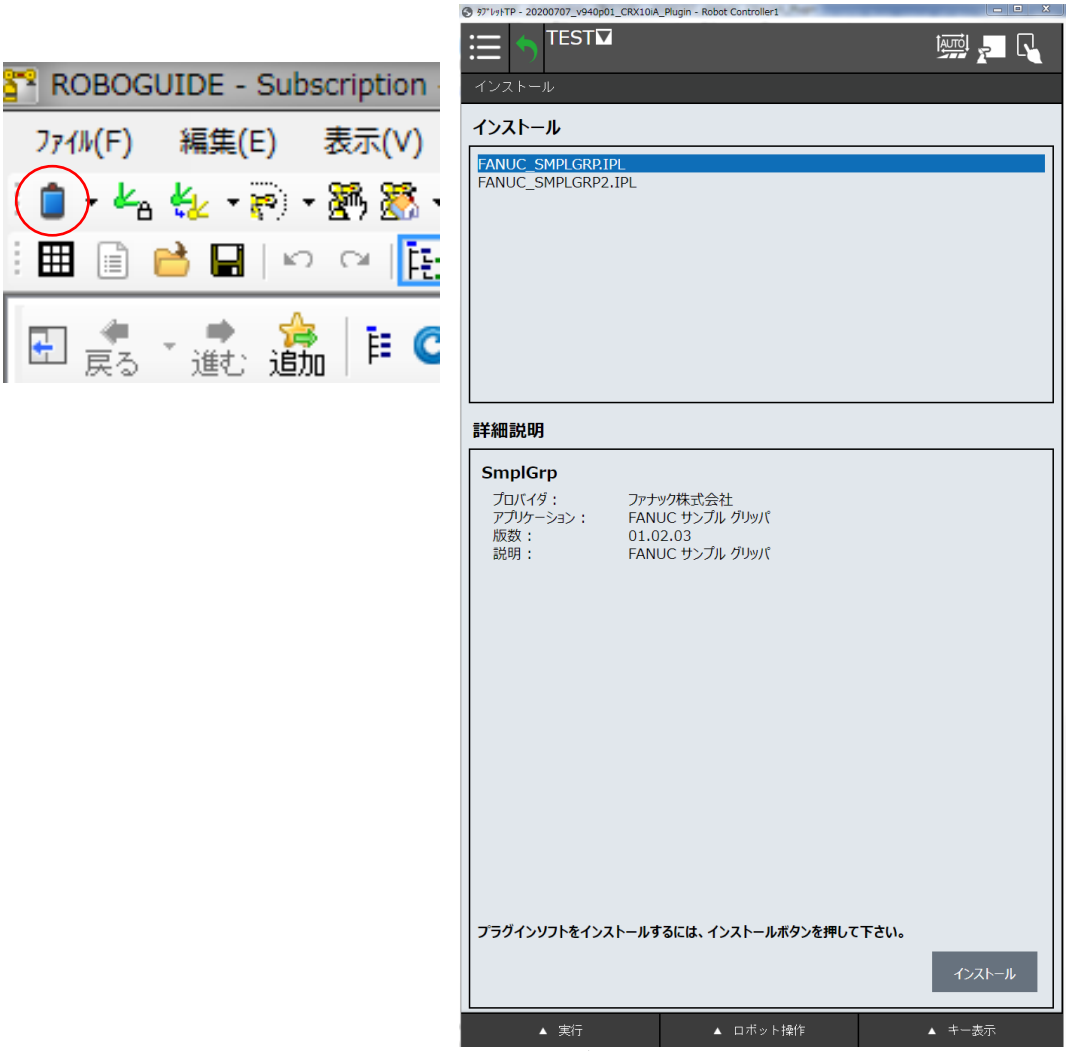


図 7.8.2 (a) ROBOGUIDE のタブレット TP ウインドウ

注

ROBOGUIDE 上で開くタブレット TP ウインドウでは、ブラウザの開発者モードを使用できません。ブラウザの開発者モードを使用する場合は、PC ブラウザでロボットホームページを開き、iRProgrammer を起動してください。iRProgrammer の起動方法については、R-30iB Plus 制御装置 取扱説明書 (基本操作編) (B-83284JA) の 13 章「教示操作盤無しの操作」を参照してください。

8 プラグイン命令作成

プラグインソフトでは、周辺機器を制御するプラグイン命令を追加できます。本章では、プラグイン命令の作成方法について説明します。



ロボットグラフィック表示部

プログラムライン

アイコンパレット
命令詳細

図 8 (a) TP プログラムエディタ画面

8.1 構成ファイル

プラグイン命令を追加するためには、以下のファイルを作成します。

表 8.1 (a) 構成ファイル

内容	ファイル名	ファイルの内容
プラグイン命令定義ファイル	(任意のファイル名).lxml	以下の項目を定義するファイルです。 • プラグイン命令アイコンのタイプ • プラグイン命令アイコンの命令カテゴリ • プラグイン命令アイコンの画像データ • アイコンパレット上に表示するプラグイン命令アイコン名のテキスト • プラグイン命令実行時に呼び出されるプログラム • プラグイン命令の詳細画面データ • プラグイン命令教示時の初期値の設定 • タイムライン上のプラグイン命令選択時の Callback 関数

		ビルドすることで、(任意のファイル名)+(各言語のサフィックス).xml のファイルが生成されます。 ファイル名には半角英数字のみが使用できます。ファイル名は任意です。
プラグイン命令実行時に呼び出されるプログラム	IPL_(プラグイン ID)_(任意のプログラム名).tp “IPL_(プラグイン ID)_(任意のプログラム名).pc	プログラム実行時に呼び出されるプログラムです。プラグイン命令実行時の挙動を記述します。TP プログラム(*.tp)もしくは KAREL プログラム(*.pc)のいずれかを指定できます。
プラグイン命令アイコン	(任意のファイル名).png	プラグイン命令アイコンの画像ファイルです。
プラグイン命令詳細画面	(任意のファイル名).ishtm	プラグイン命令の詳細画面データです(HTML)。
	(任意のファイル名).js	プラグイン命令の詳細画面で使用する JavsScript ファイルです。設定範囲チェック、関連要素の操作、Callback 関数などを記述します。
	(任意のファイル名).css	プラグイン命令の詳細画面で使用する CSS ファイルです。プラグイン命令の詳細画面に独自スタイルを適用する場合に作成します。iHMI コンポーネントのスタイルを使用する場合は作成不要です。

注
- プラグイン命令定義ファイル(*.lxml, *.xml)、プラグイン命令アイコンデータファイル(*.png)、プラグイン命令詳細画面データファイル(*.ishtm, *.stm, *.js, *.css)のファイル名は任意です。使用できる文字列は英数字およびアンダーバー”_”で、拡張子を含めて 42 文字以内です。
- プラグイン ID を”(会社名)_(プラグイン名)”として定義し、TP プログラムと KAREL プログラムのファイル名を”IPL_(プラグイン ID)_(任意のプログラム名)”とする必要があります。例えば、プラグイン ID を”**FANUC_SmplGrp**”と定義した場合、プログラム名は **IPL_FANUC_SmplGrp_*****.TP, **IPL_FANUC_SmplGrp_*****.PC とする必要があります。ここで、”***”は任意のプログラム名を意味します。

8.2 プラグイン命令定義ファイル

ここでは、プラグイン命令の作成手順を例として示します。以下に、プラグイン命令の完成イメージを示します。



図 8.2 (a) プラグイン命令のイメージ

プラグイン命令を作成するために、XML ファイルでプラグイン命令を定義する必要があります。

多言語に対応するために、プラグイン命令定義ファイル(*.lxml)では辞書ファイルを参照することも可能です。辞書を参照する場合は拡張子を lxml としてください。辞書ファイルは言語毎に用意する必要があります。

ビルドツールを実行してビルドをすると、各辞書の内容が XML ファイルに埋め込まれ、言語毎の XML ファイルが出力されます。ビルド方法は 7.2 節「ビルドツール」を参照ください。ビルドツールを実行すると、生成された言語毎の XML ファイルを含めた状態で、インストールパッケージファイル(*.ipl)が生成されます。

sampleapp.lxml の内容 (ビルド前のベースファイル)

```
<?xml version="1.0" encoding="{{ienc.ienc_xml_encoding}}"?>
<editor>
  <instruction type="normal" section="cat_plugin">
    <iconfile>sampleapp.png</iconfile>
    <icontext>{{sampleapp.icon_text}}</icontext>
    <program>IPL_FANUC_SmplGrp_sample.tp</program>
    <detail>
      <title>{{sampleapp.title_text}}</title>
      <src>sampleapp.stm</src>
       <init param="1,2"></init>
      <disp>dispAdvInstData</disp>
    </detail>
  </instruction>
</editor>
```

encoding 属性を encoding="{{ienc.ienc_xml_encoding}}" と記載することで、build_ihmi.bat の実行後に生成される各言語の XML ファイルの 1 行目の encoding 属性が各言語の文字コードに自動的に置き換わります

辞書変数名を記載

属性に初期パラメータを指定

sampleappeg.utx の内容 (英語用辞書ファイル)

```
$-, icon_text     "Sample icon"
$-, title_text    "Sample instruction"
```

sampleappkn.utx の内容 (日本語用辞書ファイル)

```
$-, icon_text     'サンプルアイコン'
$-, title_text    'サンプル命令'
```

sampleappeg.xml の内容 (ビルド後の英語ファイル)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<editor>
  <instruction type="normal" section="cat_plugin">
    <iconfile>sampleapp.png</iconfile>
    <icontext>Sample icon</icontext>
    <program>IPL_FANUC_SmplGrp_sample.tp</program>
    <detail>
      <title>Sample instruction</title>
      <src>sampleapp.stm</src>
       <init param="1,2"></init>
      <disp>dispAdvInstData</disp>
    </detail>
  </instruction>
</editor>
```

sampleappkn.xml の内容 (ビルド後の日本語ファイル)

```
<?xml version="1.0" encoding="shift_jis"?>
<editor>
  <instruction type="normal" section="cat_plugin">
    <iconfile>sampleapp.png</iconfile>
    <icontext>サンプルアイコン</icontext>
    <program>IPL_FANUC_SmplGrp_sample.tp</program>
    <detail>
      <title>サンプル命令</title>
      <src>sampleapp.stm</src>
       <init param="1,2"></init>
      <disp>dispAdvInstData</disp>
    </detail>
  </instruction>
</editor>
```

注

プラグイン命令定義ファイル(*.lxml) の 1 行目の encoding 属性は、必ず「encoding="{{ienc.ienc_xml_encoding}}"」と指定してください。このように指定することで、build_ihmi.bat の実行後に生成される各言語の XML ファイルの 1 行目の encoding 属性が各言語の文字コードに自動的に置き換わります。

XML には以下の内容を記述可能です。

- プラグイン命令アイコンのタイプ
- プラグイン命令アイコンの命令カテゴリ
- プラグイン命令アイコンの画像データ

- アイコンパレット上に表示するプラグイン命令アイコンのテキスト
- プラグイン命令実行時に呼び出されるプログラム
- プラグイン命令の詳細画面データ
- プラグイン命令教示時の初期値の設定
- タイムライン上のプラグイン命令選択時の Callback 関数

XML ファイルのフォーマットについて説明します。

タグ名		
instruction		
属性		
type	string	アイコンのタイプを選択します。type="normal"を指定してください。 "normal"：通常のアイコン 
section	string	命令カテゴリタブのどこに表示するかを選択します。section="cat_plugin"を指定してください。指定を省略した場合は cat_plugin になります。 • cat_plugin： プラグイン • cat_motion： 動作 • at_ ctrl： 制御 • cat_io： 入出力 • cat_other： その他
必須/任意		
必須		

⚠ 注意
- プラグインソフトの命令カテゴリは「cat_plugin」を指定してください。
- 「cat_plugin」の命令カテゴリは V9.40P/09 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで「cat_plugin」を指定する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/08 版以前の場合は、プラグインソフトをインストールする前に V9.40P/09 版以降の最新版にアップデートしてください。」、と明記してください。

タグ名
Iconfile
コンテンツ
アイコンのイメージデータファイル名を相対パスで記述します。 ロボット制御装置上で同じフォルダ内に配置することを想定しているため、ファイル名のみになります。
必須/任意
必須

タグ名
icontext
コンテンツ
アイコン下部に表示する命令名を 16 バイト以内で記述します。
必須/任意
必須

タグ名
program
コンテンツ
TP プログラム(*.tp)もしくは KAREL プログラム(*.pc)の名前を記述します。
必須/任意

必須

タグ名
detail
コンテンツ
プラグイン命令の詳細画面処理に必要となる詳細データを子ノードに持ちます。 詳細画面がない場合、<detail>タグは省略してください。
必須/任意
任意

タグ名
title
コンテンツ
タイトル行に表示する文字列を記述します。
必須/任意
任意

タグ名
src
コンテンツ
詳細画面の stm 名または htm 名を記述する
必須/任意
init タグの属性が func の場合、必ず<src>タグを指定してください。

タグ名
init
コンテンツ
• プラグイン命令アイコンをプログラムラインにドラッグアンドドロップしたときの初期値となる、プラグイン命令アイコンの引数文字列を指定します。 • プラグイン命令アイコンの引数文字列の初期化処理は、(1) param 属性(引数文字列の指定)と、(2) func 属性(初期化関数名の指定)のいずれかの属性で指定してください。 (1) param 属性は以下のように使用します。 - param 属性を指定する場合、param 属性で記述する引数文字列の値が、プラグイン命令アイコンの引数文字列として保存される - param 属性を指定する場合、func 属性の指定は不要 - ユーザがプログラムライン上にプラグイン命令アイコンをドラッグしたときに、タブレット TP のエディタ UI がその行を保存し実行する際のプラグイン命令の引数として使用する - ユーザがプログラムライン上にプラグイン命令アイコンをドラッグし、アイコンをクリックしたときに、プラグイン命令アイコンの詳細画面入力フォームのデフォルト値として使用する (2) func 属性は以下のように使用します。 - func 属性は、プラグイン命令アイコンをプログラムラインにドラッグアンドドロップしたときに実行する初期化関数の関数名を指定し、その関数の return 値がプラグイン命令の引数文字列として保存される - func 属性を指定する場合、その init タグ内に param 属性の指定も必要 - func 属性を指定して、param 属性を指定しなかった場合、プラグイン命令アイコンをプログラムラインにドラッグアンドドロップしたときに、エラーI117 が発生する - ただし、func 属性、param 属性の両方を指定した場合は、func 属性で指定した初期化関数の return の値が優先される - func 属性を指定する場合、その初期化関数(JavaScript 関数)では、必ず setInstructionParam を使用して、プラグイン命令アイコンの引数文字列の初期化を行う必要がある
属性

param	string	引数文字列
func	string	関数名 (関数コールでの初期値設定)
必須/任意		
任意		

タグ名
Disp
コンテンツ
プログラムライン上のプラグイン命令を選択したときに実行する Callback 関数を記述します。
必須/任意
詳細画面に設定値がある場合は必ず指定してください。

8.3 プラグイン命令実行時に呼び出されるプログラム

プラグイン命令実行時の挙動を TP プログラム、もしくは KAREL プログラムで記述します。プラグイン命令の挙動を変更するパラメータはプログラムの引数として設定します。パラメータはプラグイン命令の詳細画面で設定します。詳細画面の作成については、8.5 節「命令詳細画面作成」を参照してください。パラメータ値は、TP プログラム内でヒキスウ[..] (AR[..])として読み込むことができます。TP プログラムの詳細は R-30iB Plus 取扱説明書(基本操作編) (B-83284JA)の 4 章「プログラムの構成」を参照してください。

注
- プラグイン命令で定義する TP プログラムは、ロボットの動作命令や負荷設定命令を含む場合を除き、モーショングループは［*,*,*,*,*,*,*,*］を設定して、モーショングループを持たない TP プログラムとしてください。
- ロボットの動作命令や負荷設定命令を含む TP プログラムに限り、モーショングループは［1,*,*,*,*,*,*,*］を設定して、グループ 1 のみモーショングループを設定してください。
- プラグイン命令で定義する TP プログラムには、グループ 2 以降のモーショングループを設定しないでください。
- プラグインソフトで使用する KAREL プログラムは全て、必ず「%NOLOCKGROUP」属性、もしくは「%LOCKGROUP=1」属性を指定してください。

8.3.1 TP プログラム新規作成

TP プログラムは、ROBOGUIDE もしくはロボット制御装置の環境で作成します。

本プラグインソフトで使用する TP プログラムのファイル名には、アルファベットで始まる英数字を使用してください。Autocfg.csv で定義するプラグイン ID を”(会社名)_(プラグイン名)”として定義し、TP プログラムと KAREL プログラムのファイル名を”IPL_(プラグイン ID)_(任意のプログラム名)”とする必要があります。例えば、プラグイン ID を”FANUC_SmplGrp”と定義した場合、プログラム名は IPL_FANUC_SmplGrp_***.TP, IPL_FANUC_SmplGrp_***.PC とする必要があります。ここで、”***”は任意のプログラム名を意味します。詳細は 5.1 節「インストール定義ファイルの書式」を参照してください。任意のプログラム名は ”IPL_FANUC_SmplGrp_OPEN.TP” や ”IPL_FANUC_SmplGrp_CLOSE.TP” のように、命令の内容が分かる名前で生成することを推奨します。”IPL_(プラグイン ID)_(任意のプログラム名).TP” というファイル名の合計が 36 文字以下でなければなりません。

プログラム中で制御する動作グループが無い場合は、TP プログラム生成時の詳細画面で、以下のようにモーショングループの項目にカーソルを合わせて「*」を設定します。例えば、ハンド開閉などのロジック命令のみでロボットの動作命令を含まない TP プログラムの場合は、モーショングループの項目で全グループ「*」を設定してください。詳細は R-30iB Plus 取扱説明書(基本操作編) (B-83284JA)の 4.1.4 節「モーショングループ」を参照してください。



図 8.3.1 (a) TP プログラム詳細画面のグループマスク設定

> **注**
> 負荷設定命令を含む TP プログラムの場合、ロボットの動作命令の有無に関係なく、モーショングループの設定が必要です。その場合は、モーショングループの項目で［1,*,*,*,*,*,*,*］を設定してください。

8.3.2 TP プログラムのローカルレジスタ

TP プログラム内のローカル変数として、ローカルレジスタ(レジスタ、文字列レジスタ、位置レジスタ)が使用可能です。ローカルレジスタを使用する場合は、TP プログラム生成時に、使用するローカルレジスタの数を予め設定する必要があります。

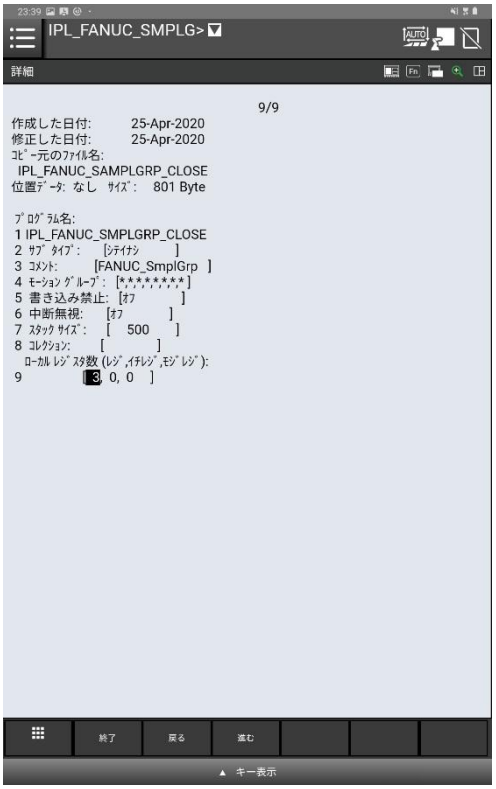


図 8.3.2 (a) TP プログラム詳細画面のローカルレジスタ数の設定

> **注**
> ローカルレジスタの数を設定すると、TP プログラムの実行時に設定した数だけ内部メモリが確保されます。設定するローカルレジスタ数を必要最小限とし、必要以上にローカルレジスタ数を設定しないようにして下さい。

各ローカルレジスタの書式は以下の通りです。レジスタ番号 1～9999 はグローバルレジスタ用のレジスタ番号として確保されており、グローバルレジスタはエンドユーザが使用するレジスタ領域ですので、プラグイン命令の TP プログラムではローカルレジスタを使用してください。ローカルレジスタのレジスタ番号は、必ず 10001～(10000＋設定したレジスタ数)の範囲のレジスタ番号を指定します。レジスタの詳細は B-83284JA R-30iB Plus 取扱説明書(基本操作編)の 4.5 節「レジスタ命令」を参照してください。

表 8.3.2 (a) ローカルレジスタ

ローカルレジスタ名称	TP プログラムでの書式	用途
ローカルレジスタ	レジ[10001～(10000＋設定したレジスタ数)]	ローカルレジスタはある整数値または小数値を格納します。レジスタの算述演算や間接指定に使用できます。
ローカル文字列レジスタ	モジレジ[10001～(10000＋設定したレジスタ数)]	ローカル文字列レジスタは英数字、カタカナ、漢字の文字列を格納します。それぞれのレジスタには最大で 254 文字を格納できます。
ローカル位置レジスタ	イチレジ[10001～(10000＋設定したレジスタ数)] イチレジ[10001～(10000＋設定したレジスタ数),位置レジスタ要素番号]	位置レジスタは、位置データ(x, y, z, w, p, r)を格納する変数です。イチレジ[a, b]のように記載する場合、a は位置レジスタ番号を表し、b は位置レジスタの要素番号を表します。

ローカルレジスタが使用できる TP 命令と、ローカルレジスタが使用できない TP 命令については、以下の表を参照してください。

表 8.3.2 (b) ローカルレジスタが使用できる TP 命令

命令	ローカルレジスタが使用できる命令一覧
制御命令	ﾚｼﾞｽﾀ, I/O, ﾓｼ/ｾﾝﾀｸ, 待機, ｼﾞｬﾝﾌﾟ/ﾗﾍﾞﾙ, 呼出し/終了, その他の命令, ｽｷｯﾌﾟ, 文字列, ﾌｶｾｯﾃｲ, FOR/ENDFOR, ﾕｰｻﾞｻﾞﾋｮｳﾊﾞﾝｺﾞｳ, ﾂｰﾙｻﾞﾋｮｳﾊﾞﾝｺﾞｳ, ﾕｰｻﾞｻﾞﾋｮｳ, ﾂｰﾙｻﾞﾋｮｳ
動作命令	ﾁｮｸｾﾝ ｲﾁﾚｼﾞ[], ﾚｼﾞ[] mm/sec
動作付加命令	ｶｿｸﾄﾞ, ｲﾁ補正,ｲﾁﾚｼﾞ[], ｲﾝｸﾘﾒﾝﾀﾙ, ﾂｰﾙ補正, ｲﾁﾚｼﾞ[], DO[ ]=…, RO[ ]=…, GO[ ]=…, AO[ ]=…, ｻｷｼﾞｯｺｳ, ｱﾄｼﾞｯｺｳ, ｻｷｼﾞｯｺｳ ｷｮﾘ

表 8.3.2 (c) ローカルレジスタが使用できない TP 命令

命令	ローカルレジスタが使用できない命令一覧
制御命令	ｲﾁ補正, ﾂｰﾙ補正
動作命令	ﾅﾒﾗｶ ﾚｼﾞ[]
動作付加命令	ｲﾁ補正, ﾂｰﾙ補正

注
上述の表は V9.40P/01 版の時点での仕様です。将来、ローカルレジスタが使用できる TP 命令が追加される可能性があります。

8.3.3 プラグイン定義アラーム

プラグインソフトで任意のアラームメッセージを定義しておき、TP プログラムおよび KAREL プログラムからアラームとして表示できます。多国語にも対応しており、アラームメッセージテキストを各言語で定義しておくことで、ロボット制御装置で選択されている言語でアラームテキストが表示されます。また、プラグインソフト毎に定義できますので、他社プラグインソフトとの衝突を考慮する必要もありません。

プラグイン定義のアラームは「PLUG-001」として以下の書式で表示されます。

```
PLUG-001 "プラグインソフトで定義されたメッセージテキスト"
```

任意の数のアラームテキストを定義可能で、アラームごとに重度(表示のみ、一時停止、強制終了)を定義できます。表示されたアラームメッセージはアラーム履歴に残ります。

使用方法の詳細は次節以降をご覧ください。

⚠ 注意
- プラグイン定義アラームは V9.40P/06 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトでプラグイン定義アラームを使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/05 版以前の場合は、プラグインソフトをインストールする前に V9.40P/06 版以降の最新版にアップデートしてください。」、と明記してください。
- 個別のプラグイン定義アラームはファナックの取扱説明書には記述されません。プラグインソフトで定義する各アラームの説明は周辺機器メーカが提供するプラグインソフト取扱説明書に記述してください。

8.3.3.1 アラームテキストの定義ファイル

アラームテキストを定義するファイルは ASCII ファイルとして作成し、plgalm.dat というファイル名でプラグインソフトのインストールパッケージファイル(*.IPL)に同梱してください。

plgalm.dat には標準言語として英語のメッセージテキストを記入してください。多国語対応のためには、国名を表す 2 文字(例: 日本語は kn)を付加したものをファイル名として、IPL ファイル中に追加してください。例えば、日本語のアラームテキストファイルは plgalmkn.dat となります。

plgalm.dat ファイルのフォーマットは以下の通りです。

```
NAME,"アラーム表示用プラグイン名"
$1,PAUSE,"メッセージテキスト 1"
$2,WARN,"メッセージテキスト 2"
$3,ABORT,"メッセージテキスト 3"
...
END
```

1 行目は必ず「NAME」から始まるようにしてください。1 行目の「NAME」の後に指定される文字列は、後述する Format 引数の値が 1 または 2 の時にプラグイン名として表示されます。

2 行目以降に各アラーム番号に対応するアラーム重度とアラームテキストを定義します。

- "$"(ドル)マークの後の番号がアラーム番号です。
- ","(コンマ)で区切られた次の文字列によりアラームの重度を定義します。
  - WARN: ワーニング。アラームテキストを表示した後もプログラムの実行は続行します。
  - PAUSE: 一時停止。アラームテキストを表示した後、実行中のプログラムは一時停止されます。
  - ABORT: 強制終了。アラームテキストを表示した後、実行中のプログラムは強制終了されます。
- ","(コンマ)区切りの後、次に続くアラームメッセージテキストは"(ダブルクォート)で囲んでください。大体 30 文字(30 バイト)を目安に長さを調節してください。メッセージテキストの途中に「%s」を挿入すると、次節に後述する ipl_fanuc_sys_erpost ルーチン、もしくは、ipl_erpost ルーチンの第 4 引数で指定された文字列で置き換えられて表示されます。

ファイルの最終行には「END」を必ず指定してください。

8.3.3.2 TP プログラムからプラグイン定義アラームを表示する方法

TP プログラムの呼び出し命令で以下の KAREL プログラムを呼び出して、プラグイン定義アラームを表示します。

```
ﾖﾋﾞﾀﾞｼ ipl_fanuc_sys_erpost(プラグイン ID, Alarm_Number, Cause_code, Param, Format)
```

ここで、各引数の意味は以下のとおりです。

- プラグイン ID (文字列): IPL ファイル内の Autocfg.csv で指定されたプラグイン ID 名
- Alarm_Number (整数): アラームテキスト定義ファイル(plgalm.dat)中のアラーム番号。アラームテキストファイルについては前節を参照ください。
- Cause_code (整数): 通常ゼロ(0)を指定してください。
- Param (文字列): メッセージテキスト内にある%s を置き換える文字列
- Format (整数): この値によりメッセージの表示方法を以下の各例のように指定できます。
  0: PLUG-001 "メッセージテキスト 3"
  1: PLUG-001 "[プラグイン名-003] メッセージテキスト 3"
  2: PLUG-001 "メッセージテキスト 3 [プラグイン名-003]"

なお、アラーム番号に対応するメッセージテキストが定義されていない場合には、 "[プラグイン名-003]"のように表示されます。この場合、アラーム重度はワーニングとなります。

8.3.3.3 KAREL プログラムからプラグイン定義アラームを表示する方法

KAREL プログラムからプラグイン定義アラームを表示することもできます。以下の KAREL ルーチン定義による IPL_ERPOST ルーチンを KAREL プログラムから呼び出してください。

目的:

指定したアラーム番号のプラグイン定義アラームを表示します。

構文:

IPL_ERPOST (*Plugin_id, Alarm_number, Cause_code, Param, Format*) from IPL_FANUC_SYS_ERPOST
入出力パラメータ:
[in] *Plugin_id* : STRING

[in] *Alarm_number* : INTEGER
[in] *Cause_code* : INTEGER
[in] *Param* : STRING
[in] *Format* : INTEGER

詳細:

- *Plugin_id* に、IPL ファイル内の Autocfg.csv に記述したプラグイン ID を指定します。
- *Alarm_number* に、アラームテキスト定義ファイル(plgalm.dat)中のアラーム番号を指定します。アラームテキスト定義ファイルについては前節を参照してください。
- *Cause_code* には、通常ゼロ(0)を指定してください。
- *Param* に、メッセージテキスト内にある%s を置き換える文字列を指定します。
- *Format* に、アラームテキストを表示するフォーマットを指定します。
  0: PLUG-001 “メッセージテキスト 3”
  1: PLUG-001 “[プラグイン名-003] メッセージテキスト 3”
  2: PLUG-001 “メッセージテキスト 3 [プラグイン名-003]”

なお、アラーム番号に対応するメッセージテキストが定義されていない場合には、“[プラグイン名-003]”のように表示されます。

8.3.4 TP プログラム作成例

グリッパ制御ユニットとロボット制御装置間で、DI/DO 信号(デジタル入出力信号)によりグリッパ制御のための信号通信を行う機器構成、および、プラグイン命令の TP プログラムの実装例を示します。エンドユーザはこのプラグイン命令の TP プログラム(IPL_FANUC_SmplGrp_Ctrl.TP)を呼び出すことで、グリッパの開閉を制御します。

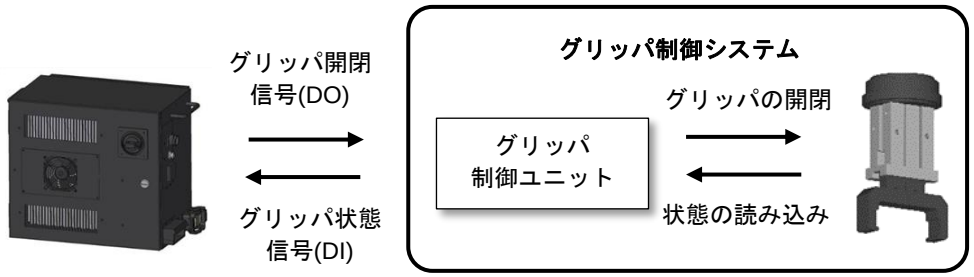


図 8.3.4 (a) プラグイン命令プログラムの概要

入出力信号とアラーム表示
本節の例では、引数パラメータで DO 信号(グリッパ開閉信号)を設定して、グリッパ開閉を制御することを想定します。

- 引数パラメータ値が 1 : DO[101]=OFF / DO[102]=ON を設定することで、グリッパが閉じます。
- 引数パラメータ値が 0 : DO[101]=ON / DO[102]=OFF を設定することで、グリッパが開きます。

また、DI 信号(グリッパ状態信号)を読むことにより、グリッパの状態を確認できます。

- グリッパ状態 DI[101] = ON : グリッパがエラー状態にあることを示します
- グリッパ状態 DI[101] = OFF : グリッパが正常に作動できる状態を示します

上記 DI/DO 信号のインデックス番号は、6 章「HTML 画面と TP プログラムで KAREL 変数を共有する」で実装例として説明するプラグイン画面で設定し、その設定値を 6.1 節「設定項目および KAREL 変数定義」で設定する KAREL 変数に格納することを想定しています。

TP プログラムの挙動
グリッパの開閉操作の前にグリッパの状態を確認し、エラー状態である場合には次のユーザアラームを表示します。

表示されるアラーム：
‘PLUG-001 グリッパがエラー状態です’

```
/PROG   IPL_FANUC_SmplGrp_Ctrl
DEFAULT_GROUP  = *,*,*,*,*,*,*,*,*;
/MN
    1:   !local register ;
    2:   !R[10001-10003] are usable ;
    3:   !---------- ;
    4:   !Copy to local variable ;
    5:   R[10001]=$[IPL_FANUC_SmplGrp_Cfg]OpenDo ;
    6:   R[10002]=$[IPL_FANUC_SmplGrp_Cfg]ClosDo ;
    7:   R[10003]=$[IPL_FANUC_SmplGrp_Cfg]StatDi ;
    8:    ;
    9:   !Check Index Number ;
   10:   IF R[10001]<=0,JMP LBL[1] ;
   11:   IF R[10002]<=0,JMP LBL[1] ;
   12:   IF R[10003]<=0,JMP LBL[1] ;
   13:    ;
   14:   !Check Gripper Status ;
   15:   IF (DI[R[10003]]=ON) THEN ;
   16:   CALL IPL_FANUC_SYS_ERPOST('FANUC_SmplGrp', 1, 0, '', 0) ;
   17:   ELSE ;
   18:   IF (AR[1] = 0) THEN
   19:   !Open Hand ;
   20:   DO[R[10001]]=ON ;
   21:   DO[R[10002]]=OFF ;
   22:   ELSE ;
   23:   !Close Hand ;
   24:   DO[R[10001]]=OFF ;
   25:   DO[R[10002]]=ON ;
   26:   ENDIF ;
   27:   ENDIF ;
   28:    ;
   29:   LBL[1] ;
/POS
/END
```

図 8.3.4 (b) TP プログラム実装例

注

- プラグイン ID を”(会社名)_(プラグイン名)”として定義し、TP プログラムと KAREL プログラムのファイル名を”IPL_(プラグイン ID)_(任意のプログラム名)”とする必要があります。例えば、プラグイン ID を”**FANUC_SmplGrp**”と定義した場合、プログラム名は **IPL_FANUC_SmplGrp_*****.TP, **IPL_FANUC_SmplGrp_*****.PC とする必要があります。ここで、”***”は任意のプログラム名を意味します。
- TP プログラムのパラメータ命令でパラメータ名として入力できる文字列は最大 30 文字です。KAREL 変数を指定する場合、KAREL プログラム名と KAREL 変数名をパラメータ命令に指定する必要があります。この制約を踏まえてプラグイン ID、KAREL プログラム名、KAREL 変数名を決定してください。

  例）ﾚｼﾞ[10001]=$**[IPL_FANUC_SmplGrp_Cfg]OpenDo**
  123456789012345678901234567890

8.3.5 サードパーティビジョンインタフェースの使用

サードパーティビジョンインタフェース(A05B-2670-R897)を使用することで、ファナック以外のビジョンシステムを TP プログラムから実行することができます。ビジョンシステムはロボット制御装置とイーサネットで接続でき、またビジョンシステムがサードパーティビジョンインタフェースに対応している必要があります。詳細はサードパーティビジョンインタフェース 取扱説明書(B-84244JA)を参照してください。

8.3.6 KAREL プログラムによるユーザソケットメッセージ機能の通信設定

KAREL プログラムで、ユーザソケットメッセージ機能(A05B-2600-R648)を使用する際に必要な通信設定を行う KAREL ルーチンを使用できます。ユーザソケットメッセージ機能の詳細はイーサネット機能 取扱説明書(B-82974JA)の 10 章「ソケットメッセージ」を参照してください。

> ⚠ **注意**
> ソケットメッセージ通信設定を行う IPL_SETUPSM ルーチンは V9.40P/31 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 KAREL ルーチンを使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/30 版以前の場合は、プラグインソフトをインストールする前に V9.40P/31 版以降の最新版にアップデートしてください。」、と明記してください。

以下の KAREL ルーチン定義による IPL_SETUPSM ルーチンを KAREL プログラムから呼び出してください。

目的:

ユーザソケットメッセージ機能を使用する際に必要な通信設定を行います。

構文:

IPL_SETUPSM (*Mode, Tag_num, Port_num, Ip_addr, Comment, Use_udp, Status*) from IPL_FANUC_SYS_SETUPSM
入出力パラメータ:
[in] *Mode* : INTEGER
[in] *Tag_num* : INTEGER
[in] *Port_num* : INTEGER
[in] *Ip_addr* : STRING
[in] *Comment* : STRING
[in] *Use_udp* : BOOLEAN
[out] *Status* : INTEGER

詳細:

- *Mode* にクライアントタグ、サーバタグのどちらを設定するか選択します。
  1: クライアントタグ
  2: サーバタグ
- *Tag_num* に使用するタグの番号を指定します。1 から 8 を使用できます。
- *Port_num* にソケット通信に使用するサーバポートの番号を指定します。0 から 32767 の範囲で指定してください。
- *Ip_addr* にサーバの IP アドレス、もしくはホスト名を指定します。この項目は *Mode* に「1: クライアントタグ」を選択したときのみ設定が必要です。*Mode* に「2: サーバタグ」を選択した場合は空文字を設定してください。
- *Comment* にタグにコメントを設定します。コメントが不要の場合は空文字を設定してください。
- *Use_udp* (真理値)に UDP で通信を行うか指定します。
  FALSE: UDP を使用しない
  TRUE: UDP を使用する
- *Status* に処理のステータスが返ります。0 は正常、それ以外はエラーが発生したことを示します。各 *Status* 値に対応するアラームメッセージは以下の通りです。

表 8.3.6 (a) ステータス値に対応するアラームメッセージ

ステータス値	アラーム	メッセージ	対策
169006	PLUG-006	引数が不正です(IPL_SETUPSM,1)	第 1 引数を確認してください。
169007	PLUG-007	引数が不正です(IPL_SETUPSM,2)	第 2 引数を確認してください。
169008	PLUG-008	引数が不正です(IPL_SETUPSM,3)	第 3 引数を確認してください。
169009	PLUG-009	引数が不正です(IPL_SETUPSM,4)	第 4 引数を確認してください。
169010	PLUG-010	引数が不正です(IPL_SETUPSM,5)	第 5 引数を確認してください。
169011	PLUG-011	引数が不正です(IPL_SETUPSM,6)	第 6 引数を確認してください。
169012	PLUG-012	R648 が必要です	ユーザソケットメッセージ機能 (A05B-2600-R648)をインストールしてください。

8.3.7 KAREL プログラムによる負荷情報の設定

KAREL プログラムで、負荷設定のコメント、負荷重量、重心位置、イナーシャ値を設定できます。
以下の KAREL ルーチン定義による IPL_SETPLD ルーチンを KAREL プログラムから呼び出してください。

> ⚠ **注意**
> 負荷情報を設定する IPL_SETPLD ルーチンは V9.40P/41 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 KAREL ルーチンを使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/40 版以前の場合は、プラグインソフトをインストールする前に V9.40P/41 版以降の最新版にアップデートしてください。」、と明記してください。

目的:

指定した動作グループの負荷設定スケジュールに、引数で指定した負荷情報を設定します。

構文:

IPL_SETPLD (*Gnum, Shc_num, Comment, Mass, Cog, Inertia, Unit_type, Flg_lim, Status*) from IPL_FANUC_SYS_PLGUTIL
入出力パラメータ:
[in] *Gnum* : INTEGER
[in] *Sch_num* : INTEGER
[in] *Comment* : STRING
[in] *Mass* : REAL
[in] *Cog* : ARRAY[3] OF REAL
[in] *Inertia* : ARRAY[3] OF REAL
[in] *Unit_type* : INTEGER
[out] *Flg_lim* : BOOLEAN
[out] *Status* : INTEGER

詳細:

- *Gnum* に、負荷情報を設定するグループ番号を指定します。
- *Sch_num* に、負荷情報を設定するスケジュール番号を指定します。
- *Comment* に、負荷スケジュールに設定するコメントを指定します。未初期化の STRING 変数が指定された場合、コメントは設定されません。
- *Mass* に負荷の重量を指定します。単位は[kg]です。
- *Cog* に負荷の重心位置を指定します。*Cog[1]*には X 方向、*Cog[2]*には Y 方向、*Cog[3]*には Z 方向の距離を指定します。単位は *Unit_type* に指定した値によって異なります。
- *Inertia* に負荷のイナーシャ値を指定します。*Inertia[1]*には X 軸まわり、*Inertia[2]*には Y 軸まわり、*Inertia[3]*には Z 軸まわりのイナーシャ値を指定します。単位は *Unit_type* に指定した値によって異なります。
- *Unit_type* に指定した値に応じて、以下のように *Cog*, *Inertia* に指定する値の単位が変わります。
  - *Unit_type*=1 : *Cog* [cm], *Inertia* [kgfcms^2]
  - *Unit_type*=2 : *Cog* [m], *Inertia* [kgm^2]
  - *Unit_type*=3 : *Cog* [mm], *Inertia* [kgm^2]
- *Flg_lim* には過負荷制限チェックの結果が返ります。過負荷制限を超えた場合に TRUE が返り、それ以外は FALSE が返ります。
- *Status* には処理の以下のステータスが返ります。*Status* が非ゼロの場合は、POST_ERR ビルトインでアラームポストしてください。
  - 0 : 設定に成功しました。
  - 15170 : 設定に成功しました。ただし、許容値に近いです。
  - 15171 : 設定に成功しました。ただし、許容負荷を超えています。
  - 15671 : 負荷設定に失敗しました。
  - 169025 : 負荷設定に失敗しました。引数が不正です。
  - その他 : 内部エラーです。

8.3.8 KAREL プログラムによる負荷情報の取得

KAREL プログラムで、負荷設定のコメント、負荷重量、重心位置、イナーシャ値を取得できます。
以下の KAREL ルーチン定義による IPL_GETPLD ルーチンを KAREL プログラムから呼び出してください。

> ⚠ **注意**
> 負荷情報を取得する IPL_GETPLD ルーチンは V9.40P/41 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 KAREL ルーチンを使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/40 版以前の場合は、プラグインソフトをインストールする前に V9.40P/41 版以降の最新版にアップデートしてください。」、と明記してください。

目的:

指定した動作グループの負荷設定スケジュールの負荷情報を取得します。

構文:

IPL_GETPLD (*Gnum, Shc_num, Comment, Mass, Cog, Inertia, Unit_type, Status*) from IPL_FANUC_SYS_PLGUTIL
入出力パラメータ:
[in] *Gnum* : INTEGER
[in] *Sch_num* : INTEGER
[out] *Comment* : STRING
[out] *Mass* : REAL
[out] *Cog* : ARRAY[3] OF REAL
[out] *Inertia* : ARRAY[3] OF REAL
[in] *Unit_type* : INTEGER
[out] *Status* : INTEGER

詳細:

- *Gnum* に、負荷情報を取得するグループ番号を指定します。
- *Sch_num* に、負荷情報を取得するスケジュール番号を指定します。
- *Comment* に、指定した負荷スケジュールに設定されているコメントが返ります。
- *Mass* に、指定した負荷スケジュールに設定されている重量が返ります。単位は[kg]です。
- *Cog* に、指定した負荷スケジュールに設定されている重心位置が返ります。*Cog[1]*には X 方向、*Cog[2]*には Y 方向、*Cog[3]*には Z 方向の距離が返ります。取得した値の単位は *Unit_type* に指定した値によって異なります。
- *Inertia* に、指定した負荷スケジュールに設定されている負荷のイナーシャ値が返ります。*Inertia[1]*には X 軸まわり、*Inertia[2]*には Y 軸まわり、*Inertia[3]*には Z 軸まわりのイナーシャ値が返ります。取得した値の単位は *Unit_type* に指定した値によって異なります。
- *Unit_type* に指定した値に応じて、以下のように *Cog*, *Inertia* に返る値の単位が変わります。
  - *Unit_type*=1 : *Cog* [cm], *Inertia* [kgfcms^2]
  - *Unit_type*=2 : *Cog* [m], *Inertia* [kgm^2]
  - *Unit_type*=3 : *Cog* [mm], *Inertia* [kgm^2]
- *Status* には処理の以下のステータスが返ります。*Status* が非ゼロの場合は、POST_ERR ビルトインでアラームポストしてください。
  - 0 : 取得に成功しました。
  - 169026 : 負荷設定値の取得に失敗しました。引数が不正です。
  - その他 : 内部エラーです。

8.3.9 KAREL プログラムによる MAC アドレスの取得

KAREL プログラムで、ロボット制御装置のポート 1 の MAC アドレスを取得できます。MAC アドレスはロボット制御装置ごとに固有の値となるため、ロボット制御装置の個体判別等に使用できます。
本機能を使用するには、以下の KAREL ルーチン定義による IPL_GETMACADDR ルーチンを KAREL プログラムから呼び出してください。

> ⚠ **注意**
> 負荷情報を設定する IPL_GETMACADDR ルーチンは V9.40P/44 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本 KAREL ルーチンを使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/43 版以前の場合は、プラグインソフトをインストールする前に V9.40P/44 版以降の最新版にアップデートしてください。」、と明記してください。

目的:

ロボット制御装置のポート 1 の MAC アドレスを文字列で返します。

構文:

IPL_GETMACADDR (*Mac_addr*, *Status*) from IPL_FANUC_SYS_PLGUTIL
入出力パラメータ:
[out] *Mac_addr* : STRING
[out] *Status* : INTEGER

詳細:

- *Mac_addr* にロボット制御装置のポート 1 の MAC アドレスの文字列が返ります。(例 : ”00:e0:e4:80:8f:2d”)
- MAC アドレスは 17 文字のため、*Mac_addr* には 17 文字以上の STRING を指定してください。
- *Status* には処理の以下のステータスが返ります。*Status* が非ゼロの場合は、POST_ERR ビルトインでアラームポストしてください。
  - 0 : 取得に成功しました。
  - 12337 : ROBOGUIDE 環境で実行されました。IPL_GETMACADDR ルーチンは、ROBOGUIDE 環境で MAC アドレスを取得できません。
  - その他 : 内部エラーです。

8.4 プラグイン命令アイコンの作成

エディタ画面に表示するプラグイン命令アイコンは、透過 PNG 形式で作成する必要があります。以下にアイコンに必要な要素を示します。



- 画像形式 : 透過 PNG 形式
- サイズ : 256px × 256px (正方形)
- 色 : 1 色～2 色利用
- 背景色 : 透明
- ファイル名 : 英文字

8.5 プラグイン命令の詳細画面作成

プラグイン命令ごとのパラメータ設定のための画面を表示します。プラグイン命令の詳細画面は、iHMI エディタでプラグイン命令アイコンを選択時に、詳細領域に表示されます。
HTML で詳細画面を作成します。拡張子を ishtm としてください。ファイルの作成法方法は 6.3.2 節「iHMI コンポーネントの利用」を参照ください。
多言語に対応するために、プラグイン命令詳細画面ファイルでは辞書ファイル(*.utx)を参照することが可能です。辞書を参照する場合は拡張子を*.ishtml (*.ihtml)としてください。辞書ファイルは言語毎に用意する必要があります。
ビルドツールによってビルドをすると、各辞書の内容が*.stm (*.htm)ファイルに埋め込まれ、言語毎の*.stm (*.htm) ファイルが出力されます。ビルド方法は 7.2 節「ビルドツール」を参照ください。生成された言語毎の*.stm (*.htm) ファイルをインストールパッケージファイル(*.ipl)に含め、インストールファイルとして使用します。

sampleapp.ishtm の内容 (ビルド前オリジナルファイル)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="<!-- #echo var={_CHARSET} -->">
    <link rel="stylesheet" href="/frh/ihmi/ihmicomponentstyl.css"/>
    <link rel="stylesheet" href="/frh/inst/sampleapp/sampleapp.css"/>
    <script src="/frh/plugin/FANUC_SmplGrp/sampleapp.js"></script>
  </head>
  <body>
    <div class="content">
      <div id="labelTitle" class="labelTitle titlename">
        <label id="label1">{{sampleapp.title_text}}</label>
      </div>
      <div class="formLine">
        <div class="mouse text headerTextbox">text1:</div>
        <ihmi type="textboxString" id="text1" mode="allallowed" designpattern="mouse"
width="120px" errPopPos="1" unSelect="true"></ihmi>
      </div>
      <div class="formLine">
        <div class="mouse text headerTextbox">text2:</div>
        <ihmi type="textboxString" id="text2" mode="allallowed" designpattern="mouse"
width="120px" errPopPos="1" unSelect="true"></ihmi>
      </div>
    </div>
  </body>
</html>
```

文字コードは SSI で取得

iHMI コンポーネントの CSS 読み込み

通常のタグで用いる CSS 読み込み

設定範囲チェック、関連要素の操作、Callback 関数などのコード (JavaScript) 読み込み

詳細画面で設定されるパラメータは、TP プログラムの引数として設定します。
設定のため、初期値設定用の Callback 関数及び、詳細画面にて設定値を表示する Callback 関数の記載例を下記に示します。下記の「dispAdvInstData」関数が詳細画面にて設定値を表示する Callback 関数です。本関数の argStr には、設定されているサブプログラムの引数が入ります。
「callbackInputText1」関数と「callbackInputText2」関数は詳細画面のテキストボックスの値を変更した場合の例となります。詳細画面のテキストボックスの値を変更したときには、「setInstructionParam」API をコールし、サブプログラム引数の更新を行います。

```
// 入力フォームの値の設定
function dispAdvInstData(argStr) {//argStr="1,2"
      //表示時の処理

      if( argStr.length === 0 ){
          return false; //引数が不正の場合は、false で返す
      }
       var params = argStr.split(',');
       argText1 = params[0];
       argText2 = params[1];

      //入力フォームに設定
       document.getElementById('text1').refresh(argText1, '1', '200', false);
       document.getElementById('text2').refresh(argText2, '1', '200', false);

      //Callback 関数の登録
       document.getElementById('text1').setCallback(callbackInputText1);
       document.getElementById('text2').setCallback(callbackInputText2);

       return true; // 処理の成功
}
// フォーム入力時 Callback
function callbackInputText1(id, type, value) {
      var iptText1Num = value;
      var iptText2Num = document.getElementById('text2').getValue();
      parent. setInstructionParam (iptText1Num + ',' + iptText2Num);
}
function callbackInputText2(id, type, value) {
      var iptText2Num = value;
      var iptText1Num = document.getElementById('text1').getValue();
      parent. setInstructionParam (iptText1Num + ',' + iptText2Num);
}
```

8.5.1 詳細画面作成用 API

プラグイン命令詳細画面で、プラグイン命令で呼び出す TP プログラムもしくは KAREL プログラムの引数を指定する API について説明します。

top.setInstructionParam

概要		
プラグイン命令アイコンのドラッグアンドドロップ(XML 定義の初期化関数の呼び出し)、タイムライン上のアイコンの選択(XML 定義の描画関数の呼び出し)で、詳細画面の描画を完了したときに、その完了を ihmieditor に伝えるための関数です。		
関数名		
setInstructionParam (argStr)		
引数		
変数名	**型**	**説明**
argStr	String	プラグイン命令で呼び出す TP プログラムもしくは KAREL プログラムの引数を指定します。
戻り値		
変数名	**型**	**説明**
N/A	N/A	N/A

注

setInstructionParam を使用して、プラグイン命令の引数に REAL 値を指定するとき、その REAL 値と整数値の差分が 0 の場合は、整数値として引数に渡されます。その場合、プラグイン命令の KAREL プログラムで引数の型をチェックして、その変数型に応じて格納先を分岐するように実装してください。

8.6 プラグイン命令のエラー一覧

プラグイン命令作成者が入力ミスやファイル配置ミスなどの誤りが検出できるように、エラーコードが表示されます。エラーは複数のプラグイン命令で同時に検出され、最大 3 つのプラグイン命令エラーが表示されます。

表 8.6 (a) エラーコード一覧

No	エラーコード	エラー内容
I0xx : プラグイン命令　ファイル存在		
1	I002	02 : PNG が存在しない
2	I003	03 : src タグがあり、STM が存在しない
I1xx : プラグイン命令　XML 記載ミス		
3	I102	02 : iconfile タグがない
4	I103	03 : icontext タグがない
5	I104	04 : program タグがない
6	I110	10 : instruction の type がない
7	I111	11 : src タグがなく、init が func
8	I113	13 : type に不正な値が入っている
9	I114	14 : section に不正な値が入っている
10	I117	17 : init func があるのに init param の設定が無い
11	I119	19 : instruction タグがない
12	I120	20 :同一の TP プログラム、KAREL プログラム名がある
13	I123	23 : TP プログラム名が 36 文字以上
14	I124	24 : TP プログラム名が数字から始まる、または全角文字が使用されている
I2xx : プラグイン命令　XML 取得通信エラー		
15	I201	01 : XML 取得時のパースエラー
16	I202	02 : XML 取得時のタイムアウトエラー
17	I203	03 : XML 取得時のその他のエラー
I5xx : プラグイン命令　Js エラー		
18	I501	01 : disp 用の Callback 関数の return が false
19	I502	02 : init 用の Callback 関数の return が false
20	I503	03 : init の func にある関数が存在しない

9 インストール時のエラー表示

本章では、プラグインソフトインストール時に発生するエラー表示を説明します。

9.1 エラー表示一覧

9.1.1 SDTL (ID = 123)

SDTL-084 データタイプに誤りがあります(行%s)
［原因］ インストール定義ファイル(Autocfg.csv)に不正なコマンド、もしくは、現在インストールされているロボット制御装置のソフトウェア版数でサポートされていないコマンドが含まれています。
［対策］ インストール定義ファイル(Autocfg.csv)の内容を確認してください。ロボット制御装置のソフトウェア版数が古い場合は、ロボット制御装置のソフトウェアを最新版にアップデートしてください。

SDTL-101 プラグインの数が多すぎます
［原因］ インストールされたプラグインの数が限度を超えました。
［対策］ 不必要なプラグインをアンインストールしてからもう一度インストール操作をしてください。

SDTL-102 操作がキャンセルされました
［原因］ インストールが操作者によりキャンセルされました。
［対策］ もう一度操作を行なってください。

SDTL-103 適合しないプログラム名です
［原因］ プラグインソフトの内容が不正です。TP プログラムまたは KAREL プログラムのファイル名が”IPL_(プラグイン ID)_(任意のプログラム名)”のルールに適合していません。
［対策］ プラグインソフトの作成者に連絡してください。

SDTL-104 無効なプラグインソフトです(%s)
［原因］ プラグインソフトの内容が不正です。
［対策］ プラグインソフトの作成者に連絡してください。

SDTL-105 サポートしていないロボット機種です
［原因］ サポートしていないロボット機種にプラグインをインストールしようとしています。
［対策］ サポートしていないロボットのため、このプラグインはインストールできません。プラグインソフトの作成者に連絡して、ロボット機種に適合するプラグインを入手してください。

SDTL-106 必要なオプション(%s)がロードされていません
［原因］ プラグインが機能するために必要としているオプション機能がロードされていません。
［対策］ このプラグインが必要としているオプション機能がないとこのプラグインは機能しません。必要なオプションをロードしてから再度プラグインをインストールしてください。

SDTL-111 メニュー情報取得エラー
［原因］ プラグインソフトの内容が不正です。メニュー情報ファイルが見つかりません。
［対策］ プラグインソフトの作成者にご連絡ください。

SDTL-112 メニュー情報構文エラー
［原因］ プラグインソフトの内容が不正です。メニュー情報ファイルの構文エラーです。
［対策］ プラグインソフトの作成者にご連絡ください。

SDTL-113 システムエラー
［原因］ システムエラーです。システム変数の読み込みに失敗しました。
［対策］ ファナックにご連絡ください。

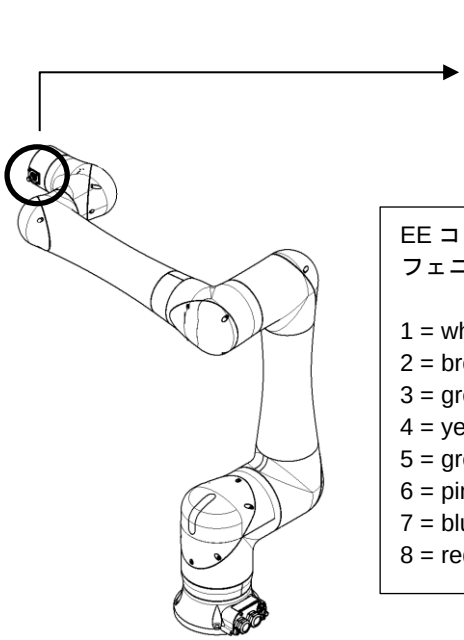
SDTL-114 システムエラー

［原因］　システムエラーです。システム変数の設定に失敗しました。

［対策］　ファナックにご連絡ください。

10 ツール I/F (CRX シリーズのみ)

本章では、FANUC Robot CRX シリーズ手首部の EE コネクタに搭載するツール I/F 仕様を説明します。



EE コネクタ
フェニックス・コンタクト 1424230

1 = white = アナログ入力 1 / RS485+
2 = brown = アナログ入力 2 / RS485-
3 = green = デジタル入力 2 (RI[2])
4 = yellow = デジタル入力 1 (RI[1])
5 = grey = Power1
6 = pink = デジタル出力 2 (RO[2]) / Ground2 / OPEN
7 = blue = デジタル出力 1 (RO[1]) / Power2 / OPEN
8 = red = Ground1

図 10 (a) EE コネクタ上のツール I/F

⚠ 注意

- EE コネクタに接続するツールの場合、ツールを EE コネクタに接続する前に、ツール I/F 通信機能の設定を完了しておく必要があります。ツール I/F 通信機能の設定を完了する前にツールを接続した場合、例えば不正な供給電圧設定等によりツールが故障する恐れがあります。ツール I/F 通信機能の各種設定は、Autocfg.csv 内に記述し、プラグインソフトをインストールした後に電源再投入することで設定内容が反映されます。
- FANUC Robot CRX シリーズは EE (EE1)コネクタ接続の活線挿抜に対応していません。EE コネクタに接続するツールの場合、ツールメーカが提供する取扱説明書で下記の注意事項を明記してください。
  [注意事項]
  - ツールを EE コネクタに接続する前に、プラグインソフトのインストールを完了してください。
  - EE コネクタに接続するツールを着脱する場合は、以下のいずれかの方法で実施してください。以下に示す方法以外の手段でツールを着脱した場合、ツールが故障するおそれがあります。
    1. 手動ツールチェンジ画面のガイダンスに従い、ツールを着脱する
    2. ツール切り離し・結合アイコン命令を教示したプログラムを実行して、ツールを着脱する
    3. ロボット制御装置の電源を落とした状態で、ツールを着脱する

10.1 ツール I/F 仕様

EE コネクタの仕様は以下の通りです。ツール側は一般的な M8 コネクタ(ソケット)がご利用いただけます。表 10.1 (a) にコネクタ仕様例を記載しておりますが、この表から選択する必要はありません。ツールに合わせて適切な長さを選択してください。

表 10.1 (a) EE コネクタ仕様

コネクタ	仕様	製造・販売元
ツール側	1404187 M8 ソケット ストレート (1.5m) (片側ばら線) 1404196 M8 ソケット ストレート (0.3m) (片側ストレートプラグ) 1404198 M8 ソケット ストレート (0.6m) (片側ストレートプラグ) 1404199 M8 ソケット ストレート (1.5m) (片側ストレートプラグ) 1404191 M8 ソケット アングル (1.5m) (片側ばら線) 1404201 M8 ソケット アングル (0.3m) (片側ストレートプラグ) 1404202 M8 ソケット アングル (0.6m) (片側ストレートプラグ) 1404203 M8 ソケット アングル (1.5m) (片側ストレートプラグ)	フェニックス・コンタクト (株)
ロボット機構部側	1424230	フェニックス・コンタクト (株)

表 10.1 (b) ロボット機構部側 EE コネクタのピン配置

ピン番号	信号または電源名
1	アナログ入力 1 / RS485+
2	アナログ入力 2 / RS485-
3	デジタル入力 2 (RI[2])
4	デジタル入力 1 (RI[1])
5	Power1 (24V / 12V / Power off)
6	デジタル出力 2 (RO[2]) / Ground2 (*1) / Open (*2)
7	デジタル出力 1 (RO[1]) / Power2 (*1) (24V / 12V / Power off) / Open (*2)
8	Ground1

(*1) ピン番号 7 を電源供給ピン(Power2)として使用することも可能です。ピン番号 7 を Power2 として使用する場合、ピン番号 6 は Ground2 になります。
(*2) 2A 供給モードの場合、ピン番号 6 とピン番号 7 は Open になり、ピン番号 5 (Power1) とピン番号 8 (Ground1) から電源供給が行われます。

注
Power1 と Power2 は独立した電源になっています。Power1 と Power2 を接続してはいけません。

10.1.1 ツール I/F 出力電源規格

ツール I/F からツールへの出力電源の規格は以下の通りです。

出力電源 (Power1、Power2) は 24V 供給、12V 供給、電源供給しない、の 3 つのモードに対応しています。
Power1 と Power2 は独立した電源であり、異なるモードの設定が可能です。
出力電源の電気的仕様は以下の通りです。

表 10.1.1 (a) 出力電源電気的仕様

項目	Min	Typ	Max	Unit
24V 供給モード	21.6	24	26.4	V
12V 供給モード	10.8	12	13.2	V
定格電流 (*2)	--	1000	2000 (*1)	mA

項目	Min	Typ	Max	Unit
定格電流 (2A 供給モード) (*2)	--	2000	2000	mA

(*1) 2A 供給モードでない場合の 2000mA は max 1s、Duty cycle max 10%であり、平均出力電流は 1000mA を超えないようにしてください。また Power2 を使用する場合は、Power1 と Power2 の合計電流が 2000mA を越えないようにしてください。
(*2) 誘導性負荷を接続する場合は、保護ダイオードを使用してください。

注
2A 供給モードを使用する場合、デジタル出力 1 (RO[1]) (Power2) とデジタル出力 2 (RO[2]) (Ground2) は使用できません。
Power1 と Power2 は独立した電源になっています。Power1 と Power2 を接続してはいけません。

10.1.2 デジタル出力信号規格

ツール I/F からツールへのデジタル出力信号の規格は以下の通りです。

デジタル出力信号は以下の 3 つのモードに対応しています。

表 10.1.2 (a) デジタル出力信号モード仕様

モード	Active	Inactive
シンク	Low	Open
ソース	High	Open
Push-Pull	High	Low

デジタル出力信号の電気的特性は以下の通りです。

表 10.1.2 (b) デジタル出力信号電気的仕様

項目	Min	Typ	Max	Unit
オープン時電圧	-0.5	-	26	V
シンク電流 / ピン	0	600	3000 (*1)	mA
ソース電流 / ピン	-	-	500 (*2)	mA

(*1) 3000mA は max 1s、Duty cycle max 10%であり、平均出力電流は 600mA を超えないようにしてください。
(*2) Power1 とデジタル出力信号のソース電流の合計電流が 2000mA を超えないようにしてください。

注
2A 供給モードを使用する場合、デジタル入力 1 (RI[1]) (Power2) とデジタル出力 2 (RO[2]) (Ground2) は使用できません。

デジタル出力信号は 4ms 周期で更新されます。

10.1.3 デジタル入力信号規格

ツールからツール I/F へのデジタル入力信号の規格は以下の通りです。

表 10.1.3 (a) デジタル入力信号電気的仕様

項目	Min	Typ	Max	Unit
入力電圧	-0.5	-	26	V
L レベルロジック電圧	-	-	2.0	V

項目	Min	Typ	Max	Unit
H レベルロジック電圧	5.5	-	-	V
入力抵抗	-	48k		Ω

デジタル入力信号は 4ms 周期で更新されます。

10.1.4 アナログ入力信号規格

ツールからツール I/F へのアナログ入力信号の規格は以下の通りです。

アナログ入力は電圧入力(0-10V) と電流入力(4-20mA) に対応しており、アナログ入力 1 とアナログ入力 2 で異なる入力モードの設定が可能です。

アナログ入力信号の電気的特性は以下の通りです。

表 10.1.4 (a) アナログ入力信号電気的仕様

項目	Min	Typ	Max	Unit
電圧モード 入力電圧	-0.5	-	12.5	V
入力抵抗 (0V to 10V)	-	10.6k		Ω
分解能	-	12	-	bit
電流モード 入力電圧	-0.45	-	4.5	V
電流モード 入力電流	-2.5	-	24	mA
入力抵抗 (4mA to 20mA)	-	183	187	Ω
分解能	-	12	-	bit

注
アナログ入力 1 とアナログ入力 2 を使用する場合、RS485 通信は使用できません。

アナログ入力信号は 4ms 周期で更新されます。

10.1.5 RS485 通信信号規格

ツール I/F とツールの RS485 通信信号規格を以下に示します。
ボーレート、ストップビット、パリティについて以下から選択可能です。

ボーレート	9.6k, 19.2k, 38.4k, 57.6k, 115.2k, 1M, 2M, 5M
ストップビット	1, 2
パリティ	None, Odd, Even
終端抵抗	120Ω (*1)

(*1) ツールの RS485 通信ラインに 120Ω の終端抵抗を挿入してください。

注
RS485 通信を使用する場合、アナログ入力 1 とアナログ入力 2 は使用できません。

11 ツール I/F 通信機能 (CRX シリーズのみ)

本章では、FANUC Robot CRX シリーズの EE(EE1)コネクタに接続するツールとロボット制御装置との通信機能を提供するツール I/F 通信機能を説明します。本機能では、EE コネクタに接続するツールとロボット制御装置間の以下の通信が可能です。

- デジタル入力信号 : 2 点 (RI[1], RI[2])
- デジタル出力信号 : 2 点 (RO[1], RO[2])
- アナログ入力信号 : 2 点 (AI)
- RS485 シリアル通信

ツール I/F 通信機能を使用する機器が 1 台のみ、もしくは 2 台以上使用する場合の 1 台目の通信設定は、本章を参照してください。
ツール I/F 通信機能を使用する機器を 2 台以上実装する場合の 2 台目以降の通信設定は、12.5 節「プラグインソフト開発者向け情報」を参照してください。

注
RS485 シリアル通信を使用する場合、アナログ入力 1 とアナログ入力 2 は使用できません。

⚠ 注意
- EE コネクタに接続するツールの場合、ツールを EE コネクタに接続する前に、ツール I/F 通信機能の設定を完了しておく必要があります。ツール I/F 通信機能の設定を完了する前にツールを接続した場合、例えば不正な供給電圧設定等によりツールが故障する恐れがあります。ツール I/F 通信機能の各種設定は、Autocfg.csv 内に記述し、プラグインソフトをインストールした後に電源再投入することで設定内容が反映されます。
- FANUC Robot CRX シリーズは EE (EE1)コネクタ接続の活線挿抜に対応していません。EE コネクタに接続するツールの場合、ツールメーカが提供する取扱説明書で下記の注意事項を明記してください。
  [注意事項]
  - ツールを EE コネクタに接続する前に、プラグインソフトのインストールを完了してください。
  - EE コネクタに接続するツールを着脱する場合は、以下のいずれかの方法で実施してください。以下に示す方法以外の手段でツールを着脱した場合、ツールが故障するおそれがあります。
    1. 手動ツールチェンジ画面のガイダンスに従い、ツールを着脱する
    2. ツール切り離し・結合アイコン命令を教示したプログラムを実行して、ツールを着脱する
    3. ロボット制御装置の電源を落とした状態で、ツールを着脱する

11.1 ツール I/F 通信機能の共通設定

ツール I/F 通信機能を使用する場合に必要な共通設定を行います。Autocfg.csv に記述する SSET,TLIF_COMMON コマンドで設定します。Autocfg.csv については 5 章「インストール定義ファイル」、SSET コマンドについては 5.2.6 節「SSET コマンド」を参照してください。設定後は電源再投入が必要です。ツール I/F 通信機能を使用しない場合は設定不要です。

⚠ 注意
本節に記載する SSET,TLIF_COMMON コマンドは V9.40P/73 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本コマンドを使用する場合、ツールメーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/72 版以前の場合は、プラグインソフトをインストールする前に V9.40P/73 版以降の最新版にアップデートしてください。」、と明記してください。

構文

SSET,TLIF_COMMON,ツール I/F 有効/無効,電源供給モード,Power1 の電圧設定,Power2 の電圧設定,2A 供給モード時の電圧設定,!!!!

使用例

```
SSET,TLIF_COMMON,1,1,2,0,0,!!!!
```

表 11.1 (a) ツール I/F 通信機能の共通設定

設定項目	設定値
ツール I/F 有効/無効	0: ツール I/F 通信機能を無効にします 1: ツール I/F 通信機能を有効にします(標準)
電源供給モード	電源供給モードの設定 1: デジタル出力モード(標準) 2: 2A 供給モード 3: 2 ピン供給モード (2 つのピンで電源供給するモード)
Power1 の電圧設定	Power1 (5 番ピン) の供給電圧の設定 0: OFF(電源供給しない) (標準) 1: 12V 2: 24V のいずれかを指定してください。 それ以外の値に設定された場合は 0[V]になります。
Power2 の電圧設定	Power2 (7 番ピン) の供給電圧の設定 0: OFF(電源供給しない) (標準) 1: 12V 2: 24V のいずれかを指定してください。 それ以外の値に設定された場合は 0[V]になります。
2A 供給モード時の電圧設定	2A 供給モード時 (電源供給モードに 2A 供給モードを選択した場合) の供給電圧の設定 0: OFF(電源供給しない) (標準) 1: 12V 2: 24V のいずれかを指定してください。 それ以外の値に設定された場合は 0[V]になります。

注
2A 供給モード、もしくは、2 ピン供給モードを使用する場合、デジタル出力 1,2 (RO[1], RO[2]) は使用できません。

11.2 デジタル出力信号の通信設定

ツール I/F 通信機能によるデジタル出力 1,2 (RO[1], RO[2])を使用する場合に必要な通信設定を行います。Autocfg.csv に記述する SSET,TLIF_DOCFG コマンドで設定します。Autocfg.csv については 5 章「インストール定義ファイル」、SSET コマンドについては 5.2.6 節「SSET コマンド」を参照してください。設定後は電源再投入が必要です。デジタル出力信号を使用しない場合は設定不要です。

⚠ 注意
本節に記載する SSET,TLIF_DOCFG コマンドは V9.40P/73 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本コマンドを使用する場合、ツールメーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/72 版以前の場合は、プラグインソフトをインストールする前に V9.40P/73 版以降の最新版にアップデートしてください。」、と明記してください。

構文

```
SSET,TLIF_DOCFG,デジタル出力 1 モード設定,デジタル出力 2 モード設定,デジタル出力電圧設定,!!!!
```

使用例

```
SSET,TLIF_DOCFG,2,2,2,!!!!
```

表 11.2 (a) ツール I/F 通信機能のデジタル出力信号の設定

名前	設定値
デジタル出力 1 モード設定	電源供給モードにデジタル出力モードを選択した場合の、デジタル出力 1 (RO[1]) (7 番ピン) の出力モード設定 0: 使用しない (標準) 1: シンク 2: ソース 3: Push-Pull
デジタル出力 2 モード設定	電源供給モードにデジタル出力モードを選択した場合の、デジタル出力 2 (RO[2]) (6 番ピン) の出力モード設定 0: 使用しない (標準) 1: シンク 2: ソース 3: Push-Pull
デジタル出力電圧設定	デジタル出力 1 (RO[1]) (7 番ピン) とデジタル出力 2 (RO[2]) (6 番ピン) の電圧設定 0: OFF (標準) 1: 12V 2: 24V

注
- 電源供給モードに「2 (2A 供給モード)」、もしくは「3 (2 ピン供給モード)」を指定する場合、デジタル出力 1 (RO[1]) とデジタル出力 2 (RO[2]) は使用できません。
- デジタル出力 1, 2 モードのうち、少なくとも一方が「1 (シンク)」もしくは「3 (Push-Pull)」の場合、デジタル出力電圧設定を「1 (12V)」に設定できません。
- デジタル出力電圧設定の設定可能な設定値は、デジタル出力 1, 2 モードの組み合わせによって変わります。表 11.2 (c)を参照してください。

表 11.2 (b) 各電源供給モードにおける必要な設定項目

ピン番号	デジタル出力モード	2A 供給モード	2 ピン供給モード
Power1 (5 番ピン)	Power1 の電圧設定	2A 供給モード時の電圧設定	Power1 の電圧設定
Power2 (7 番ピン)	-	-	Power2 の電圧設定
デジタル出力 1 (RO[1]) (7 番ピン)	デジタル出力 1 モード デジタル出力電圧設定	-	-
デジタル出力 2 (RO[2]) (6 番ピン)	デジタル出力 2 モード デジタル出力電圧設定	-	-

表 11.2 (c) デジタル出力電圧設定の設定可能な設定値

		デジタル出力 2 モード			
		0 (使用しない)	**1 (シンク)**	**2 (ソース)**	**3 (Push-Pull)**
デジタル出力 1 モード	**0 (使用しない)**	0 (OFF)	0 (OFF)	1 (12V), 2 (24V)	2 (24V)
	1 (シンク)	0 (OFF)	0 (OFF)	2 (24V)	2 (24V)
	2 (ソース)	1 (12V), 2 (24V)	2 (24V)	1 (12V), 2 (24V)	2 (24V)
	3 (Push-Pull)	2 (24V)	2 (24V)	2 (24V)	2 (24V)

11.3 アナログ入力信号の通信設定

ツール I/F 通信機能によるアナログ入力信号 1,2 を使用する場合に必要な通信設定を行います。Autocfg.csv に記述する SSET,TLIF_AICFG コマンドで設定します。Autocfg.csv については 5 章「インストール定義ファイル」、SSET コマンドについては 5.2.6 節「SSET コマンド」を参照してください。設定後は電源再投入が必要です。アナログ入力信号を使用しない場合は設定不要です。

⚠ 注意
本節に記載する SSET,TLIF_AICFG コマンドは V9.40P/73 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本コマンドを使用する場合、ツールメーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/72 版以前の場合は、プラグインソフトをインストールする前に V9.40P/73 版以降の最新版にアップデートしてください。」、と明記してください。

構文

```
SSET,TLIF_AICFG,1/2 番ピン使用方法の設定,アナログ入力 1 モード設定,アナログ入力 2 モード設定,!!!!
```

使用例

```
SSET,TLIF_AICFG,2,1,1,!!!!
```

表 11.3 (a) ツール I/F 通信機能のアナログ入力信号の設定

名前	設定値
1/2 番ピン使用方法の設定	EE コネクタの 1/2 番ピンの使用方法の設定 (アナログ入力と RS485 シリアル通信の切り替え) 0: OFF 1: RS485 (標準) 2: アナログ入力
アナログ入力 1 モード設定	アナログ入力 1 (1 番ピン)のモード設定 0: OFF (標準) 1: 0-10V (電圧) 2: 4-20mA (電流)
アナログ入力 2 モード設定	アナログ入力 2 (2 番ピン)のモード設定 0: OFF (標準) 1: 0-10V (電圧) 2: 4-20mA (電流)

RS485 シリアル通信とアナログ入力は、どちらか 1 つのみ使用できます。アナログ入力を使用する設定にした場合、アナログ入力信号 2 点は AI にラック 107, スロット 1 として自動的に割り付けられます。自動割り付けされた AI 信号のインデックス番号を特定するには GET_PORT_ASG ビルトインを使用します。また、SET_PORT_ASG ビルトインを使用して割り付け位置を変更することも可能です。なお、アナログ入力のハードウェアの解像度は 12 ビットです。

注
- RS485 シリアル通信を使用する場合、アナログ入力信号は使用できません。
- アナログ入力を使用する場合は、1/2 番ピン使用方法の設定値を「2:アナログ入力」に設定し、使用するアナログ入力のアナログ入力モードを必ず「1: 0-10V (電圧)」または「2: 4-20mA (電流)」に設定してください。
- ツール I/F 通信機能のアナログ入力信号は電源投入時に AI に自動割り付けされますが、割り付けられる AI のインデックス番号はシステムによって異なります。ツール I/F 通信機能のアナログ入力信号を使用する場合は、本節で後述する GET_PORT_ASG ビルトインを使用して電源投入時に AI 信号のインデックス番号を特定してください。Autocfg.csv の LOAD コマンドに COLD オプションを付与すると、電源投入の度に KAREL プログラムを起動できます。詳細は 5.2.2 節「LOAD コマンド」を参照してください。

11.3.1 GET_PORT_ASG ビルトインプロシージャ

目的:

I/O の割り付け情報を取得します。指定の論理 I/O に割り当てられている物理 I/O を突き止めます。

構文:

GET_PORT_ASG(*Log_port_type, Log_port_no, Rack_no, Slot_no, Phy_port_type, Phy_port_no, N_ports, Status*)
入出力パラメータ:
[in] *Log_port_type* :INTEGER
[in] *Log_port_no* :INTEGER

[out] *Rack_no* :INTEGER
[out] *Slot_no* :INTEGER
[out] *Phy_port_type* :INTEGER
[out] *Phy_port_no* :INTEGER
[out] *N_ports* :INTEGER
[out] *Status* :INTEGER
%ENVIRONMENT Group :IOSETUP

詳細:

- *Log_port_type* は割付にアクセスする I/O の種類を指定します。AI (アナログ入力) の場合 3 を指定します。
- *Log_port_no* に割付にアクセスする I/O の番号を指定します。割り付け情報を取得する AI 信号のインデックス番号を指定します。
- *Rack_no* にはモジュールを含んでいるラックが返ります。ツール I/F のアナログ入力信号の場合は 107 です。
- *Slot_no* には I/O モジュールを含んでいるスロットが返ります。ツール I/F のアナログ入力信号の場合は 1 です。
- *Phy_port_type* には割り付けられている I/O の種類が返されます。これは多くの場合 *Log_port_type* と同じです。例外は *Log_port_type* がグループタイプ(io_gpin 又は io_gpout)である場合です。又は I/O がメモリイメージやダミーポートに割り付けられている場合です。
- *Phy_port_no* には割り付けられている I/O 番号が返ります。ツール I/F のアナログ入力信号の 1 番ピンの場合は 1, 2 番ピンの場合は 2 です。
- *N_ports* には論理 I/O に割り付けられている物理 I/O の数が返ります。
- *Status* はパラメータが有効で指定のポートが割り付けられているなら 0 が返ります。さもなくば、エラーコードが返ります。

11.3.2 SET_PORT_ASG ビルトインプロシージャ

目的:

I/O の割り付けを設定します。1 つまたはそれ以上の論理 I/O ポートを指定の物理 I/O ポートに割り当てます。

構文:

SET_PORT_ASG(*Log_port_type, Log_port_no, Rack_no, Slot_no, Phy_port_type, Phy_port_no, N_ports, Status*)
入出力パラメータ:
[in] *Log_port_type* : INTEGER
[in] *Log_port_no* : INTEGER
[in] *Rack_no* : INTEGER
[in] *Slot_no* : INTEGER
[in] *Phy_port_type* : INTEGER
[in] *Phy_port_no* : INTEGER
[in] *N_ports* : INTEGER
[out] *Status* : INTEGER
%ENVIRONMENT Group :IOSETUP

詳細:

- *Log_port_type* に割付を行う論理 I/O ポートの種類を指定します。AI (アナログ入力)の場合 3 を指定します。
- *Log_port_no* に割付を行う論理 I/O の番号を指定します。
- *Rack_no* に I/O モジュールのラックを指定します。ツール I/F のアナログ入力信号の場合は 107 です。
- *Slot_no* に I/O モジュールを含んでいるスロットを指定します。ツール I/F のアナログ入力信号の場合は 1 です。
- *Phy_port_type* に割付先となる物理 I/O ポートの種類を指定します。これは多くの場合 *Log_port_type* と同じです。例外は *Log_port_type* がグループタイプ(io_gpin 又は io_gpout)である場合です。又は I/O がメモリメージやダミーポートに割り付けられる場合です。AI (アナログ入力) の場合 3 を指定します。
- *Phy_port_no* に割付先となる物理 I/O ポートの番号を指定します。ツール I/F のアナログ入力信号の 1 番ピンの場合は 1, 2 番ピンの場合は 2 です。
- *N_ports* に論理 I/O ポートが割り付けられる物理 I/O ポートの数を指定します。
- *Status* はパラメータが有効なら 0 が返ります。さもなくば、エラーコードが返ります。もし指定されたポートが存在しないか、*Log_port_type* の *Phy_port_type* への割付が認められていない場合、割付は不正です。例えば、GIN は DOUT に割付できません。*Log_port_type* と *Phy_port_type* のどちらもシステム I/O ポート(例えば SOPIN)にできません。

注
信号割り付けを変更した場合は、ロボット制御装置の電源再投入が必要です。

11.4 シリアル通信の通信設定

ツール I/F 通信機能によるシリアル通信(RS485)を使用する場合に必要な通信設定を行います。Autocfg.csv に記述する SSET コマンドで設定します。Autocfg.csv については 5 章「インストール定義ファイル」、SSET コマンドについては 5.2.6 節「SSET コマンド」を参照してください。設定後は電源再投入が必要です。シリアル通信を使用しない場合は設定不要です。

⚠ 注意
本節に記載する SSET,TLIF_RS485 コマンドは V9.40P/73 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本コマンドを使用する場合、ツールメーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/72 版以前の場合は、プラグインソフトをインストールする前に V9.40P/73 版以降の最新版にアップデートしてください。」、と明記してください。

構文

```
SSET,TLIF_RS485,1/2 番ピン使用方法の設定,パリティ設定,ストップビット設定,ボーレート設定,!!!!
```

使用例

```
SSET,TLIF_RS485,1,0,0,0,!!!!
```

注
- シリアル通信を使用する場合、使用するプロトコルは、ロボット制御装置からの 1 つ(492 バイト以内)の要求に対しツールが 1 つ(500 バイト以内)応答するプロトコルである必要があります。
- ツールは要求の開始から 10ms 以内に応答してください。
- シリアル通信のタイミングでは I/O の更新間隔が 12ms になります。

表 11.4 (a) ツール I/F 通信機能のシリアル通信設定

名前	説明
1/2 番ピン使用方法の設定	EE コネクタの 1/2 番ピンの使用方法の設定 (アナログ入力と RS485 シリアル通信の切り替え) 0: OFF 1: RS485 (標準) 2: アナログ入力
パリティ設定	シリアル通信のパリティ設定 0: なし(標準) 1: 奇数 2: 偶数
ストップビット設定	シリアル通信のストップビット設定 0: 1-bit (標準) 1: 2-bit
ボーレート設定	シリアル通信のボーレート設定 0: 5M (標準), 1: 2M, 2: 1M, 3: 115.2k, 4: 57.6k, 5: 38.4k, 6: 19.2k, 7: 9.6k [bps]

11.5 シリアル通信用 KAREL ビルトイン

シリアル通信用に以下の KAREL ビルトイン関数があります。KAREL プログラムで使用可能です。

ROBOGUIDE で TLIF_WRITE ビルトイン、TLIF_READ ビルトイン、TLIF_CLEAR_BUFFER ビルトインを実行した場合、status に「HOST-427 ROBOGUIDE はﾂｰﾙｲﾝﾀﾌｪｰｽ通信機能のｼﾘｱﾙ通信用 KAREL ﾋﾞﾙﾄｲﾝに対応していません」を表す 67427 を返します。
ROBOGUIDE で TLIF_BYTES_AHEAD を実行すると、*N_bytes* は常に 0 を返します。

以下のカレルビルトインはマルチタスクに対応しています。

11.5.1 TLIF_WRITE

目的:

シリアルデータを送信します。

構文:

TLIF_WRITE(*Length, Send_data, Status*)
入出力パラメータ:
[in] *Length* : INTEGER
[in] *Send_data* : ARRAY OF BYTE
[out] *Status* : INTEGER
%ENVIRONMENT Group :TLIF

詳細:

- *Length* で送信バイト数を指定します。最大 492 バイトまで送信できます。
- *Send_data* の先頭から *length* バイト送信されます。
- *Status* は処理のステータスです。0 以外の値はエラーが発生したことを意味します。
- 本関数を一度呼んだ後は、11.5.3 節に記載する TLIF_READ で応答データを全て読み込んでから、次のデータを送信してください。本関数を複数回呼んだ後に応答が来るような使い方はできません。そのような使い方をした場合、通信がタイムアウトするまで停止しワーニングが発生することがあります。
- 本関数は実際に送信する前に返ります。
- アラームが発生するかエラーが返らない限り必ず送信されるので、送信バッファの中身が送信されたことを確認する関数は用意されません。

11.5.2 TLIF_BYTES_AHEAD

目的:

EE コネクタに接続されたツールから受信したシリアルデータのバイト数を調べます。

構文:

TLIF_BYTES_AHEAD(*N_bytes*)
入出力パラメータ:
[out] *N_bytes* : INTEGER
%ENVIRONMENT Group :TLIF

詳細:

- *N_bytes* が 0 の場合データを受信していません。1 以上の場合は受信したデータのバイト長です。

11.5.3 TLIF_READ

目的:

EE コネクタに接続されたツールから受信したシリアルデータを読みます。

構文:

TLIF_READ(*Timeout, Size, Read_data, Status*)
入出力パラメータ:
[in] *Timeout* : INTEGER
[in,out] *Size* : INTEGER
[out] *Read_data* : ARRAY OF BYTE
[out] *Status* : INTEGER
%ENVIRONMENT Group :TLIF

詳細:

- *Timeout* [ms]でタイムアウトします。0 の場合はすでに受信しているサイズを読みます(ただし *Size* バイト以下)。(-1)の場合は *Size* バイト受信するまで無限待ちします。タイムアウト時は、タイムアウトまでに受け取ったバイト数を *Size* に設定し、データを *Read_data* に設定します。
- *Size* に設定されたバイト数読みます。読んだバイト数が *Size* に設定されます。*Size* に設定されたバイト数以上受信すると本関数が返ります。最大 500 バイトまで受信できます。

- *Read_data* に読み込まれます。
- *Status* は処理のステータスです。0 以外の値はエラーが発生したことを意味します。

11.5.4 TLIF_CLEAR_BUFFER

目的:

送信バッファ、受信バッファをクリアします。

構文:

TLIF_CLEAR_BUFFER(*Status*)
入出力パラメータ:
[out] *Status* : INTEGER
%ENVIRONMENT Group :TLIF

詳細:

- 特にプログラム開発時などに前回の動作が不具合や途中で止まるなどして、受信バッファに溜まった状態でプログラムを動作させた際に、前回受信したデータを受信することを防ぐために、最初に本関数を呼ぶことを推奨します。
- クリア後に受信してしまうなどして全てクリアできない場合、ツールが送信後しばらく待ってから本関数を呼ぶとクリアできます。
- *Status* は処理のステータスです。0 以外の値はエラーが発生したことを意味します。

11.5.5 TLIF_CRC16

目的:

Modbus などで使用されている CRC を計算します。

構文:

TLIF_CRC16(*PuchMsg, UsDataLen, Crc, Status*)
入出力パラメータ:
[in] *PuchMsg* : ARRAY OF BYTE
[in] *UsDataLen* : INTEGER
[out] *Crc* : ARRAY OF BYTE
[out] *Status* : INTEGER
%ENVIRONMENT Group :TLIF

詳細:

- *PuchMsg* の先頭から *UsDataLen* 番目までのデータを使って、*Crc*[1], *Crc*[2]に Modbus などで使用されている CRC を設定します。
- *Status* は処理のステータスです。0 以外の値はエラーが発生したことを意味します。

11.6 シリアル通信サンプルプログラム

シリアル通信のサンプルプログラムを以下に示します。
Modbus の例を示していますが、Modbus である必要はありません。

```
PROGRAM IPL_FANUC_SmplGrp_SrlComm

%NOLOCKGROUP
%ENVIRONMENT TLIF
%ENVIRONMENT PC
%INCLUDE        etdccopt
%INCLUDE        klevccdf

VAR
   stat             : INTEGER
   send_data        : ARRAY[255] OF BYTE --  最大 492
   receive_data : ARRAY[255] OF BYTE --  最大 500
   crc              : ARRAY[2]     OF BYTE
   start_add        : INTEGER
   register_num : INTEGER
   read_size        : INTEGER

ROUTINE add_intpc2(out_buffer : ARRAY OF BYTE; dat_index : INTEGER; number : INTEGER) FROM
IPL_FANUC_SmplGrp_SrlComm
ROUTINE add_realpc2(out_buffer : ARRAY OF BYTE; dat_index : INTEGER; number : REAL) FROM
IPL_FANUC_SmplGrp_SrlComm


------------------------------------------------------------
----------------- メイン関数 ----------------------
------------------------------------------------------------
BEGIN
   -- TLIF_CLEAR_BUFFER は、前回の動作が途中で止まるなどして、受信バッファに溜まった状態で
   --プログラムを動作させた際に、前回受信したデータを受信することを防ぐ
   TLIF_CLEAR_BUFFER(stat)
   IF stat <> 0 THEN
   POST_ERR(stat, '', 0, 2)
      ABORT
   ENDIF

   send_data[1] =   1 -- ｽﾚｰﾌﾞ ｱﾄﾞﾚｽ
   send_data[2] = 16 -- ﾌｧﾝｸｼｮﾝ ｺｰﾄﾞ
   start_add = 10000
   send_data[3] = start_add DIV 256
   send_data[4] = start_add MOD 256
   register_num = 4
   send_data[5] = register_num DIV 256
   send_data[6] = register_num MOD 256
   send_data[7] = register_num * 2
   add_intpc2(send_data, 8, 2147483647)
   add_realpc2(send_data, 12, 1.5)

   TLIF_CRC16(send_data, 15, crc, stat)
   IF stat <> 0 THEN
      POST_ERR(stat, '', 0, 2)
      ABORT
   ENDIF
   send_data[16] = crc[1]
   send_data[17] = crc[2]
```

```
TLIF_WRITE(17, send_data, stat)
IF stat <> 0 THEN
   POST_ERR(stat, '', 0, 2)
   ABORT
ENDIF

DELAY 15
TLIF_BYTES_AHEAD(read_size)
WHILE read_size = 0 DO
   DELAY 1
   TLIF_BYTES_AHEAD(read_size)
ENDWHILE

TLIF_READ(1000, read_size, receive_data, stat)
IF stat <> 0 THEN
   POST_ERR(stat, '', 0, 2)
   ABORT
ENDIF

TLIF_CRC16(receive_data, read_size - 2, crc, stat)
IF stat <> 0 THEN
   POST_ERR(stat, '', 0, 2)
   ABORT
ENDIF

IF (crc[1] <> receive_data[read_size - 1]) OR (crc[2] <> receive_data[read_size]) THEN
   WRITE TPERROR (CHR(cc_clear_win)) -- Clear TP error window
   WRITE TPERROR('MODBUS ｼﾞｭｼﾝ CRC ｴﾗｰ', CR)
   --              12345678901234567892123456789312345
   ABORT
ENDIF

IF receive_data[1] <> 1 THEN
   WRITE TPERROR (CHR(cc_clear_win)) -- TP エラーウィンドウのクリア
   WRITE TPERROR('MODBUS ｼﾞｭｼﾝ ｱﾄﾞﾚｽ ｴﾗｰ', CR)
   --              12345678901234567892123456789312345
   ABORT
ENDIF

IF receive_data[2] <> 16 THEN
   WRITE TPERROR (CHR(cc_clear_win)) -- TP エラーウィンドウのクリア
   WRITE TPERROR('MODBUS ｼﾞｭｼﾝ ﾌｧﾝｸｼｮﾝｺｰﾄﾞ ｴﾗｰ', CR)
   --              12345678901234567892123456789312345
   ABORT
ENDIF

start_add = receive_data[3] * 256 + receive_data[4]
IF start_add <> 10000 THEN
   WRITE TPERROR (CHR(cc_clear_win)) -- TP エラーウィンドウのクリア
   WRITE TPERROR('MODBUS ｼﾞｭｼﾝ ｶｲｼｱﾄﾞﾚｽ ｴﾗｰ', CR)
   --              12345678901234567892123456789312345
   ABORT
ENDIF
```

TLIF_BYTES_AHEAD を使用しないのであれば、この部分は必要ありません。

```
   register_num = receive_data[5] * 256 + receive_data[6]
   IF register_num <> 4 THEN
      WRITE TPERROR (CHR(cc_clear_win)) -- TP エラーウィンドウのクリア
      WRITE TPERROR('MODBUS ｼﾞｭｼﾝ ﾚｼﾞｽﾀｽｳ ｴﾗｰ', CR)
      ABORT
   ENDIF

   WRITE TPERROR (CHR(cc_clear_win)) -- TP エラーウィンドウのクリア
   WRITE TPERROR('ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ ｾｲｺｳ', CR)
END IPL_FANUC_SmplGrp_SrlComm

-----------------------------------------------------------
ROUTINE add_intpc2
VAR
   dat_buffer     : ARRAY[8] OF BYTE
   local_stat     : INTEGER
BEGIN
   ADD_INTPC(dat_buffer, 1, number, local_stat)
   IF local_stat = 0 THEN
      out_buffer[dat_index] = dat_buffer[3]
      out_buffer[dat_index + 1] = dat_buffer[4]
      out_buffer[dat_index + 2] = dat_buffer[5]
      out_buffer[dat_index + 3] = dat_buffer[6]
   ELSE
      IF local_stat = er_uninitv THEN
         out_buffer[dat_index] = 127
         out_buffer[dat_index + 1] = 255
         out_buffer[dat_index + 2] = 255
         out_buffer[dat_index + 3] = 255
      ELSE
         POST_ERR(local_stat, '', 0, 2)
         ABORT
      ENDIF
   ENDIF
END add_intpc2

-----------------------------------------------------------
ROUTINE add_realpc2
VAR
   dat_buffer     : ARRAY[8] OF BYTE
   local_stat     : INTEGER
BEGIN
   ADD_REALPC(dat_buffer, 1, number, local_stat)
   IF local_stat = 0 THEN
      out_buffer[dat_index] = dat_buffer[3]
      out_buffer[dat_index + 1] = dat_buffer[4]
      out_buffer[dat_index + 2] = dat_buffer[5]
      out_buffer[dat_index + 3] = dat_buffer[6]
   ELSE
      POST_ERR(local_stat, '', 0, 2)
      ABORT
   ENDIF
END add_realpc2
```

11.7 アラーム一覧

ホスト-410 TLIF_READ ﾀｲﾑｱｳﾄ

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_READ がタイムアウトしました。

［対策］ 本ワーニングは通知のためのものです。タイムアウトまでに受信されたデータは読まれ、そのサイズが TLIF_READ の第 2 引数に返されます。TLIF_READ の第 1 引数であるタイムアウト値を大きくする必要がある可能性があります。

ホスト-411 TLIF_READ 要求ｻｲｽﾞｴﾗｰ

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_READ に 500 バイトより大きいサイズが要求されました。

［対策］ TLIF_READ の第 2 引数である要求サイズを小さくしてください。

ホスト-412 ﾂｰﾙ I/F 機能内部ｴﾗｰ %s

［原因］ %s=1 の場合、「ホスト-425」によって発生します。

［対策］ %s=1 の場合、「ホスト-425」と同じ対策を実施してください。

ホスト-413 TLIF_READ 受信ﾊﾞｯﾌｧ ｵｰﾊﾞﾌﾛｰ

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_READ の受信バッファがオーバフローしました。

［対策］ 受信データのサイズを 500 バイト以下にしてください。

ホスト-414 TLIF_WRITE の第 1 引数が不正

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_WRITE の第 1 引数が未初期化か 1 より小さいです。

［対策］ 送信サイズを表す TLIF_WRITE の第 1 引数を修正してください。

ホスト-415 TLIF_WRITE の第 1 引数が第 2 引数のｻｲｽﾞより大きい

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_WRITE の第 1 引数が第 2 引数のサイズよりも大きいです。

［対策］ TLIF_WRITE 内の、送信サイズを表す第 1 引数を修正するか、送信データ用 BYTE 配列である第 2 引数の配列長を修正してください。

ホスト-416 TLIF_READ の第 2 引数が未初期化か 1 より小さい

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_READ の第 2 引数が未初期化か 1 より小さいです。

［対策］ 受信データのサイズを表す TLIF_READ の第 2 引数を修正してください。

ホスト-417 TLIF_READ の第 2 引数が第 3 引数のｻｲｽﾞより大きい

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_READ の第 2 引数が第 3 引数のサイズより大きいです。

［対策］ TLIF_READ で、読み取るデータサイズを表す第 2 引数を修正するか、読み取るデータを格納する BYTE 配列である第 3 引数の配列長を修正してください。

ホスト-418 TLIF_READ の第 1 引数が未初期化かｰ 1 より小さい

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_READ の第 1 引数が未初期化か-1 より小さいです。

［対策］ タイムアウトを表す TLIF_READ の第 1 引数を修正してください。

ホスト-419 TLIF_CRC16 の第 2 引数が不正

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_CRC16 の第 2 引数が未初期化か 1 より小さいか 32767 より大きいです。

［対策］ データ長を表す TLIF_CRC16 の第 2 引数を修正してください。

ホスト-420 TLIF_CRC16 の第 2 引数が第 1 引数のｻｲｽﾞより大きい

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_CRC16 の第 2 引数が第 1 引数の配列長より大きいです。

［対策］ データ長を表す TLIF_CRC16 の第 2 引数を修正するか、第 1 引数の BYTE の配列長を修正してください。

ホスト-421 TLIF_CRC16 の第 3 引数のｻｲｽﾞが 2 より小さい

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_CRC16 の第 3 引数のサイズが 2 より小さいです。

［対策］ TLIF_CRC16 の第 3 引数の BYTE の配列長を 1 より大きくしてください。

ホスト-422 TLIF_READ の第 2 引数が 1 より小さい

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_READ の第 2 引数が 1 より小さいです。

［対策］ TLIF_READ の読み取るデータサイズを表す第 2 引数を修正してください。

ホスト-423 TLIF_WRITE 送信ﾊﾞｯﾌｧｵｰﾊﾞﾌﾛｰ

［原因］ ツール I/F 通信機能用 KAREL ビルトイン関数 TLIF_WRITE の送信バッファがオーバフローしました。

［対策］ 492 バイト以下を送信するようにしてください。すでに 492 バイト以下を送信するようにしている場合、本関数を一度呼んだ後は、11.5.3 節に記載する TLIF_READ で応答データを全て読み込んでから、次のデータを送信してください。本関数を複数回呼んだ後に応答が来るような使い方はできません。そのような使い方をした場合、通信がタイムアウトするまで停止しワーニングが発生することがあります。

ホスト-424 ﾂｰﾙ I/F 通信ｴﾗｰ

［原因］ ツール I/F ボードとロボット制御装置間の通信エラーが発生しました。

［対策］ 1. ツール I/F に接続したツールの定格電流が 2A を超える場合、ツールを交換するか、外部から電源を取得するようにしてください。
2. R-30iB Mini Plus 制御装置 保守説明書(B-84175)の「3.6 ヒューズによるトラブルシューティング」に記載されている、メインボードの FUSE11 が溶断している場合の対策を行ってください。

ホスト-425 ﾂｰﾙ I/F 通信ｴﾗｰ (%d)

［原因］ 1. ツール I/F ボードとロボット制御装置間の通信エラーが発生しました。
2. ツールからのシリアル通信の応答を 10ms 以内に受信できませんでした。

［対策］ アラームメッセージ内に表示される番号(%d)が 91 の場合
- ツールとシリアル通信を行っていない場合は、ツール I/F ボードとロボット制御装置間の機構部内ケーブル、ロボット接続ケーブル、それぞれのコネクタを確認してください。
- ツールとシリアル通信を行っている場合は、上記に加えて、ツールが 10ms 以内に応答していることと、ツールーEE コネクタ間の配線に問題が無いことを確認してください。また、「CRX-10iA, CRC-10iA/L, CRX-20iA/L 機構部 保守説明書」(B-84195) の 3.2 節「ヒューズによるトラブルシューティング」を確認してください。

アラームメッセージ内に表示される番号(%d)が 91 以外の場合、FANUC のサービス員に連絡し、アラームメッセージと、本アラームメッセージ内に表示される番号を連絡してください。

ホスト-426 ﾂｰﾙ I/F ﾎﾞｰﾄﾞﾒｼﾞｬｰﾊﾞｰｼﾞｮﾝｴﾗｰ.ﾎﾞｰﾄﾞ = %d, ｺﾝﾄﾛｰﾗ = %d

［原因］ ツール I/F ボードの版数が古いです。

［対策］ FANUC のサービス員に連絡し、アラームメッセージと本アラーム内のバージョン番号を連絡してください。

ホスト-427 ROBOGUIDE はﾂｰﾙ I/F 通信機能のｼﾘｱﾙ通信用 KAREL ﾋﾞﾙﾄｲﾝに未対応

［原因］ ROBOGUIDE はツール I/F 通信機能のシリアル通信用 KAREL ビルトインに対応していません。

［対策］ 本ワーニングは通知用です。

ホスト-428 ﾃﾞｼﾞﾀﾙ出力電圧設定・ﾓｰﾄﾞ設定ｴﾗｰ

［原因］ ツール I/F 通信機能のデジタル出力電圧設定が「1 (12V)」の場合、デジタル出力 1,2 モード設定を「1 (シンク)」もしくは「3 (Push-Pull)」に設定できません。

［対策］ 該当するデジタル出力 1,2 モード設定は「0 (使用しない)」に自動的に変更されました。正しい値を設定してください。

12 ツールチェンジ機能 (CRX シリーズのみ)

本章では、ロボット制御装置の電源を投入した状態で、CRX シリーズ手首部の EE (EE1)コネクタに接続するツールを着脱するツールチェンジ機能と、プラグインソフト側で必要な対応を説明します。

EE コネクタに接続する機器をツールチェンジ機能に対応するには、本書 11 章「ツール I/F 通信機能」に記載するツール I/F 通信機能の通信設定の他、ツール切り離し時の待機時間の設定が必要です。必要に応じてツール結合時の待機時間も設定可能です。さらに、ツール I/F 通信機能を使用する機器の設定を 1 つの Autocfg.csv に複数登録できます。

ツール I/F 通信機能を使用する機器が 1 台のみ、もしくは 2 台以上使用する場合の 1 台目の通信設定とツール切り離し・結合時の待機時間の設定は以下を参照してください。
- 11 章「ツール I/F 通信機能」
- 12.5.2 節「ツール I/F 使用機器 1 台目の切り離し・結合時の待機時間の設定」

ツール I/F 通信機能を使用する機器を 2 台以上実装する場合の 2 台目以降の通信設定とツール切り離し・結合時の待機時間の設定は以下を参照してください。
- 12.5.3 節「ツール I/F 使用機器 2 台目以降のサブ ID 設定」
- 12.5.4 節「ツール I/F 使用機器 2 台目以降のツール I/F 通信設定」
- 12.5.5 節「ツール I/F 使用機器 2 台目の切り離し・結合時の待機時間の設定」

⚠ 注意
- EE コネクタに接続するツールの場合、ツールを EE コネクタに接続する前に、ツール I/F 通信機能の設定を完了しておく必要があります。ツール I/F 通信機能の設定を完了する前にツールを接続した場合、例えば不正な供給電圧設定等によりツールが故障する恐れがあります。ツール I/F 通信機能の各種設定は、Autocfg.csv 内に記述し、プラグインソフトをインストールした後に電源再投入することで設定内容が反映されます。
- FANUC Robot CRX シリーズは EE (EE1)コネクタ接続の活線挿抜に対応していません。EE コネクタに接続するツールの場合、ツールメーカが提供する取扱説明書で下記の注意事項を明記してください。
  [注意事項]
  - ツールを EE コネクタに接続する前に、プラグインソフトのインストールを完了してください。
  - EE コネクタに接続するツールを着脱する場合は、以下のいずれかの方法で実施してください。以下に示す方法以外の手段でツールを着脱した場合、ツールが故障するおそれがあります。
    1. 手動ツールチェンジ画面のガイダンスに従い、ツールを着脱する
    2. ツール切り離し・結合アイコン命令を教示したプログラムを実行して、ツールを着脱する
    3. ロボット制御装置の電源を落とした状態で、ツールを着脱する

12.1 手動ツールチェンジ画面

手動ツールチェンジ画面は、EE コネクタに接続するツールを手動で着脱する手順をガイダンス表示して、ロボット制御装置の電源を投入した状態で、簡単にツールを着脱できます。本画面は、FANUC Robot CRX シリーズをタブレット TP で使用する場合に、標準機能として使用できます。

手動ツールチェンジ画面を表示するには、タブレット TP 画面左上のメニューアイコンをタッチしてメニューを表示し、「設定」メニューの中から「ツール I/F」を選択し、「ツール I/F」画面上部のタブで「手動ツールチェンジ」を選択します。

⚠ 注意

- 手動ツールチェンジ画面で切り離し・結合するには、ロボット制御装置ソフトウェア版数が V9.40P/38 以降、手動ツールチェンジ画面右上に表示される「Tool I/F Version」の値が V1.4 以降である必要があります。「Tool I/F Version」が V1.3 以前の場合は、ファナックにお問合わせください。
- 「Tool I/F Version」が V1.3 以前の状態で、本画面の「切り離し」もしくは「結合」ボタンを押すと、「HOST-443 ﾂｰﾙ I/F ﾎﾞｰﾄﾞﾊﾞｰｼﾞｮﾝがﾂｰﾙﾁｪﾝｼﾞに未対応です」アラームが発生します。
- ツール I/F 設定画面でツール I/F が無効に設定されている場合、手動ツールチェンジ画面は使用できません。その場合は、ツール I/F を有効にして電源再投入してください。

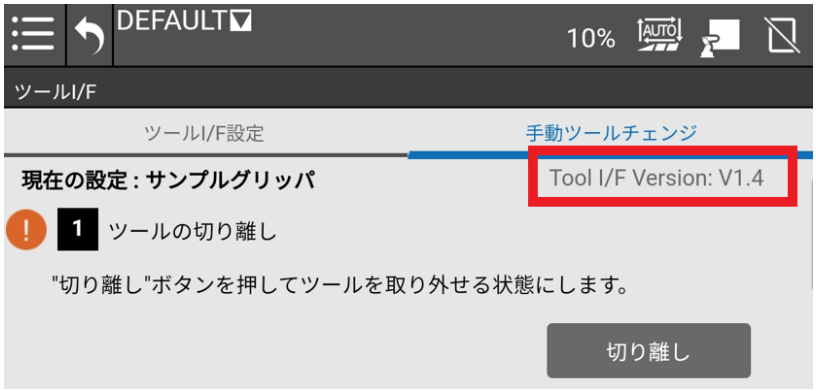


図 12.1 (a) Tool I/F Version

12.1.1 ツールの切り離し

EE コネクタからツールに対する電源供給を遮断し、ツールを安全に取り外せる状態にするには、手動ツールチェンジ画面上部の「切り離し」ボタンを押します。電源供給を遮断してからプラグインソフトであらかじめ設定された待機時間が経過した後、ボタン表示が「切り離し完了」に切り替わります。ボタン表示が「切り離し完了」に切り替わったことを確認して、ツールを取り外してください。



図 12.1.1 (a) 手動ツールチェンジ画面 (切り離し)

ただし、プラグインソフトに待機時間が設定されていない場合、「切り離し」ボタンを押したときに以下のポップアップが表示されます。ここで必要な待機時間が不明の場合は、一旦「キャンセル」を押して切り離し操作を中断するとともに、ツールへの電源供給を遮断してから安全に取り外せる状態になるまでに必要な待機時間について、ご使用のツールの仕様を確認してください。必要な待機時間が既知の場合は、「切り離し」ボタンを押して、表示される以下のポップアップで「OK」を押した後、ツールの仕様に従って、必要な待機時間が経過した後に取り外してください。

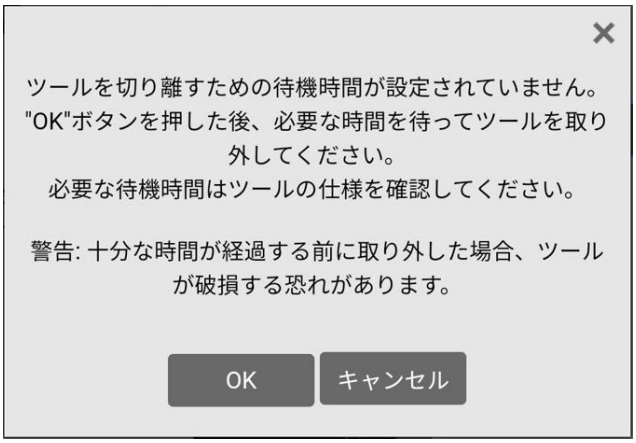


図 12.1.1 (b) 手動ツールチェンジ画面で切り離し時に表示されるポップアップ

> ⚠ **注意**
> **手動ツールチェンジ画面でツールを切り離す場合、ツールへの電源供給を遮断してから安全に取り外せる状態になるまでに必要な待機時間について、ツールの仕様を必ず確認して、その時間が経過した後に取り外してください。十分な時間を待たずに取り外した場合、ツールが破損する恐れがあります。**

12.1.2 ツールの結合

EE コネクタに接続されたツールへの電源供給を開始し、ツールとの通信を開始するには、手動ツールチェンジ画面下部のリストから結合するツールを選択し、「結合」ボタンを押します。電源供給を開始してからプラグインソフトであらかじめ設定された待機時間が経過した後、ボタン表示が「結合完了」に切り替わり、ツールが使用できる状態になります。

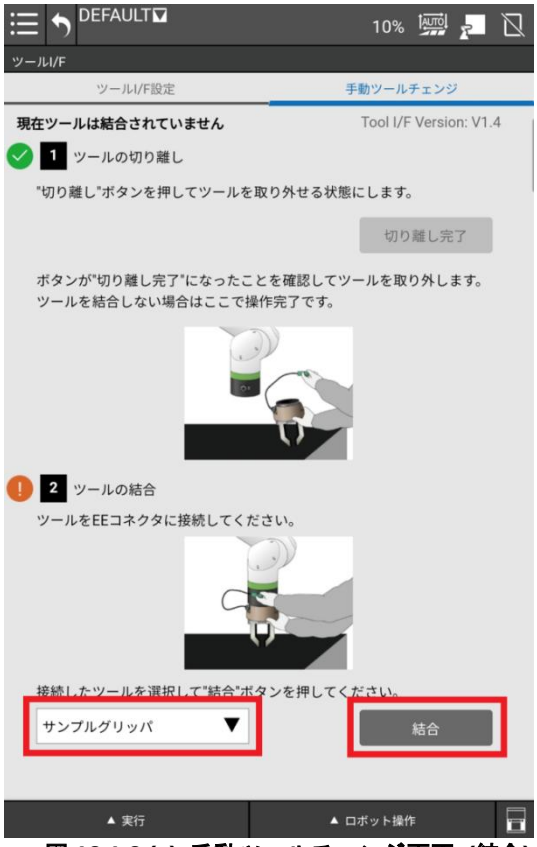


図 12.1.2 (a) 手動ツールチェンジ画面 (結合)

⚠ 注意

- 手動ツールチェンジ画面の結合ボタンで選択できるツールは、プラグインソフトがインストールされているものに限ります。
- 手動ツールチェンジ画面でツール結合時に選択するツールは、実際に接続されているツールと必ず一致するものを選択してください。選択されたツールが、実際に接続されているツールと一致しない状態で「結合」ボタンを押すと、ツールが破損する恐れがあります。
- ツール I/F 供給電源の電圧設定(24V, 12V)が異なるツールのツールチェンジはできません。供給電源の電圧設定が異なる場合は、ロボット制御装置の電源再投入が必要です。
- ツール I/F デジタル出力(RO[1], RO[2])の電圧設定が異なるツールのツールチェンジはできません。デジタル出力の電圧設定が異なる場合は、ロボット制御装置の電源再投入が必要です。

12.2 切り離し・結合アイコン命令

EE コネクタに接続するツールを切り離し、結合するためのアイコン命令を説明します。

- ツール切り離しアイコン命令
- ツール結合アイコン命令

⚠ 注意

- ツール切り離し・結合アイコン命令を使用するには、ロボット制御装置ソフトウェア版数が V9.40P/38 以降、手動ツールチェンジ画面右上に表示される「Tool I/F Version」の値が V1.4 以降である必要があります。「Tool I/F Version」の値が V1.3 以前の場合は、ファナックにお問合わせください。
- 「Tool I/F Version」の値が V1.3 以前の状態で本節のアイコン命令を実行すると、「HOST-443 ﾂｰﾙ I/F ﾎﾞｰﾄﾞﾊﾞｰｼﾞｮﾝがﾂｰﾙﾁｪﾝｼﾞに未対応です」アラームが発生します。
- ツール I/F 設定画面で、ツール I/F が無効に設定されている場合、本節のアイコン命令は使用できません。その場合は、ツール I/F を有効にして電源再投入してください。

12.2.1 ツール切り離し

ツール切り離しアイコン命令は、EE コネクタからツールに対する電源供給を遮断し、ツールを安全に取り外すことができる状態にするためのアイコン命令です。命令カテゴリは「その他」です。



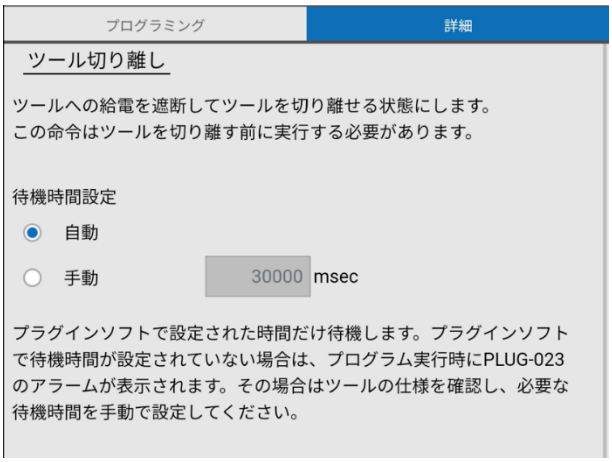


図 12.2.1 (a) ツール切り離しアイコン命令

ツール切り離しアイコン命令の詳細画面では、電源供給を遮断してから、ツールを安全に取り外せる状態になるまでの待機時間を設定できます。

「自動」を選択した場合、電源供給を遮断した後、プラグインソフトであらかじめ設定された待機時間の経過を待って、命令の実行を完了します。ただし、プラグインソフトに待機時間が設定されていないツールが接続された状態で、本命令を実行すると「PLUG-023 プラグインソフトに切り離し待機時間が設定されていません」アラームが発生します。その場合は、必要な待機時間についてご使用のツールの仕様を確認の上、必要な待機時間を手動で設定してください。

> **⚠ 注意**
> **ツール切り離しアイコン命令の詳細画面で「手動」を選択して待機時間を手動で設定する場合、ツールへの電源供給を遮断した後、安全に取り外せる状態になるまでに必要な待機時間について、ご使用のツールの仕様を確認して、その時間を手動で設定してください。十分な時間を待たずに取り外した場合、ツールが破損する恐れがあります。**

12.2.2 ツール結合

ツール結合アイコン命令は、EE コネクタに接続されたツールへの電源供給と通信を開始する命令です。命令カテゴリは「その他」です。



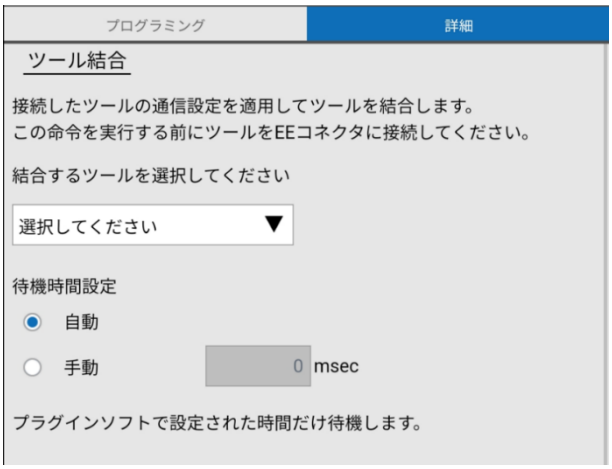


図 12.2.2 (a) ツール結合アイコン命令

ツール結合アイコン命令の詳細画面では、プラグインソフトがインストール済みのツールの一覧が表示され、結合するツールを選択します。

また、ツールによっては、電源投入直後の初期化処理のため、電源供給開始から一定時間後に使用可能になる場合があります。ツール結合アイコン命令の詳細画面では、電源供給を開始してから、ツールが使用可能になるまでの待機時間を設定できます。

「自動」を選択した場合、電源供給を開始してから、プラグインソフトであらかじめ設定された待機時間の経過を待って、命令の実行を完了します。「手動」を選択した場合、適切な待機時間を手動で設定します。

⚠ 注意
- **ツール結合アイコン命令の詳細画面で選択できるツールは、プラグインソフトがインストールされているものに限ります。**
- **ツール結合アイコン命令の詳細画面で選択するツールは、実際に接続されているツールと必ず一致するものを選択してください。選択されたツールが、実際に接続されているツールと一致しない状態でツール結合アイコン命令を実行すると、ツールが破損する恐れがあります。**
- **ツール I/F 供給電源の電圧設定(24V, 12V)が異なるツールのツールチェンジはできません。供給電源の電圧設定が異なる場合は、ロボット制御装置の電源再投入が必要です。**
- **ツール I/F デジタル出力(RO[1], RO[2])の電圧設定が異なるツールのツールチェンジはできません。デジタル出力の電圧設定が異なる場合は、ロボット制御装置の電源再投入が必要です。**

12.3 ツールチェンジ機能の制約事項

本章に説明するツールチェンジ機能の制約は以下の通りです。

- ツールチェンジ機能は、ロボット制御装置のソフトウェア版数が V9.40P/38 以降、なおかつ、手動ツールチェンジ画面右上に表示される「Tool I/F Version」の値が V1.4 以降である必要があります。
- ツールチェンジ機能は、タブレット TP が選択されている CRX シリーズでのみ利用可能です。
- ツール結合アイコン命令、および、手動ツールチェンジ画面の結合ボタンで選択できるツールは、プラグインソフトがインストールされているものに限ります。
- ツール I/F 設定画面でツール I/F が無効の場合、本機能は使用できません。その場合は、ツール I/F を有効にして電源再投入してください。
- ツール I/F 供給電源の電圧設定(24V, 12V)が異なるツールのツールチェンジはできません。供給電源の電圧設定が異なる場合は、ロボット制御装置の電源再投入が必要です。
- ツール I/F デジタル出力(RO[1], RO[2])の電圧設定(24V, 12V)が異なるツールのツールチェンジはできません。デジタル出力の電圧設定が異なる場合は、ロボット制御装置の電源再投入が必要です。

12.4 ツールチェンジ機能で発生するアラーム

PLUG-013 EE ｲﾝﾀﾌｪｰｽを有効にして電源再投入してください

［原因］ 手動ツールチェンジ画面の切り離し・結合操作時、もしくは、ツール切り離し・結合アイコン命令実行時に、ツール I/F 通信機能が無効です。

［対策］ プラグインソフトのインストール定義ファイル(Autocfg.csv)、もしくはツール I/F 設定画面でツール I/F 通信機能を有効にしてください。

PLUG-014 ｼﾝｸまたは Push-Pull では 12V を設定できません

［原因］ プラグインソフトインストール時に、インストール定義ファイル(Autocfg.csv)で定義する電圧設定が不正です。具体的には、デジタル出力 1,2 モード設定のうち少なくとも一方が「1: シンク」もしくは「3: Push-Pull」で、なおかつ、デジタル出力電圧設定が「1: 12V」に設定されています。この組み合わせは設定できません。

［対策］ プラグインソフトのインストール定義ファイル(Autocfg.csv)で定義するデジタル出力 1,2 モード設定、もしくは、デジタル出力電圧設定を見直してください。

PLUG-015 ﾂｰﾙ I/F の電圧設定を変更するには電源再投入が必要です

［原因］ 手動ツールチェンジ画面の結合操作時、もしくは、ツール結合アイコン命令実行時に、ツール結合前後の電圧設定が異なっています。ツール結合時の結合前後で電圧設定が異なる場合は、ロボット制御装置の電源再投入が必要です。

［対策］ 結合しようとしているツールの選択を見直すか、ロボット制御装置の電源を再投入して下さい。

PLUG-016 第 1 引数が不正です(ﾂｰﾙﾁｪﾝｼﾞ)

PLUG-017 第 2 引数が不正です(ﾂｰﾙﾁｪﾝｼﾞ)

PLUG-018 第 3 引数が不正です(ﾂｰﾙﾁｪﾝｼﾞ)

［原因］ ツール切り離し・結合アイコン命令の引数が不正です。

［対策］ ツール切り離し・結合アイコン命令の引数を見直してください。

PLUG-019 切り離しまたは結合処理中です

［原因］ 手動ツールチェンジ画面で前回の切り離し・結合処理が完了していない状態で、次の切り離し・結合操作が行われました。

［対策］ 切り離し・結合処理の完了を待って、次の切り離し・結合操作を行ってください。

PLUG-020 先にﾂｰﾙ切り離し命令を実行してください

［原因］ ツールが結合されている状態で、ツール結合命令が実行されました。

［対策］ ツールが結合されているときは、先にツール切り離し命令を実行して、その後にツール結合命令を実行してください。

PLUG-021 ﾂｰﾙ I/F がありません

［原因］ ツール I/F を使用できない機種で IPL_START_SERIALCOMM 関数が実行されました。

［対策］ ツール I/F を使用できない機種では IPL_START_SERIALCOMM 関数を使用できません。
IPL_START_SERIALCOMM 関数を呼び出す KAREL プログラムを見直してください。

PLUG-022 結合するﾂｰﾙを選択してください

［原因］ ツール結合命令で適切なツールが選択されていません。

［対策］ ツール結合命令で適切なツールを選択してください。

PLUG-023 ﾌﾟﾗｸﾞｲﾝｿﾌﾄに切り離し待機時間が設定されていません

［原因］ プラグインソフトに待機時間が設定されていないツールが接続された状態で、ツール切り離し命令が実行されました。

［対策］ プラグインソフトのインストール定義ファイル(Autocfg.csv)に必要な待機時間を設定してください。もしくは、ツール切り離し命令の詳細画面の待機時間設定で「手動」を選択し、ご使用のツールの仕様を確認の上、必要な待機時間を設定してください。

PLUG-024 ﾂｰﾙﾁｪﾝｼﾞ機能の初期化に失敗しました

［原因］ ツールチェンジ機能の初期化に失敗しました。ツール I/F 通信機能を使用する機器を 2 台以上使用する場合において、プラグインソフトのインストール定義ファイル(Autocfg.csv)で、2 台目以降のツールに PLGSUB コマンドが指定されていない可能性があります。

［対策］ プラグインソフトのインストール定義ファイル(Autocfg.csv)を見直し、2 台目以降のツールに PLGSUB コマンドの指定が抜けている箇所がないか確認してください。

12.5 プラグインソフト開発者向け情報

ツール I/F 通信機能を使用するツールの場合、ツール I/F 通信設定とツール切り離し・結合時の待機時間を Autocfg.csv の SSET コマンドで設定します。Autocfg.csv については 5 章「インストール定義ファイル」、SSET コマンドについては 5.2.6 節「SSET コマンド」を参照してください。Autocfg.csv に記述された設定を有効にするには電源再投入が必要です。

⚠ 注意
1 つのプラグインソフトでツール I/F 通信機能を使用する機器を 2 台以上登録し、なおかつ、そのうち少なくとも 1 台がツール I/F のアナログ入力信号を使用する場合、アナログ入力信号を使用する機器をツール I/F 通信機能を使用する機器の 1 台目に設定する必要があります。

12.5.1 ツール I/F 使用機器 1 台目のツール I/F 通信設定

ツール I/F 通信機能を使用する機器が 1 台のみ、もしくは 2 台以上使用する場合の 1 台目の通信設定は、11 章「ツール I/F 通信機能」に記載する SSET コマンドで設定してください。

12.5.2 ツール I/F 使用機器 1 台目の切り離し・結合時の待機時間の設定

ツール I/F 通信機能を使用する機器が 1 台のみ、もしくは 2 台以上使用する場合の 1 台目のツール切り離し・結合時の待機時間は、本節で説明する SSET,TLIF_DTCHTM と SSET,TLIF_ATCHTM コマンドで設定します。Autocfg.csv については 5 章「インストール定義ファイル」、SSET コマンドについては 5.2.6 節「SSET コマンド」を参照してください。設定後は電源再投入が必要です。ツール I/F 通信機能を使用しない場合は設定不要です。

⚠ 注意
本節に記載する SSET,TLIF_DTCHTM コマンドと SSET,TLIF_ATCHTM コマンドは V9.40P/38 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本コマンドを使用する場合、ツールメーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/37 版以前の場合は、プラグインソフトをインストールする前に V9.40P/38 版以降の最新版にアップデートしてください。」、と明記してください。

1. TLIF_DTCHTM: ツール I/F 通信機能を使用する機器(1 台目)について、EE コネクタからツールに対する電源供給を遮断し、ツールを安全に取り外せる状態になるまでの待機時間[ms]を設定します。

   構文
   ```
   SSET,TLIF_DTCHTM,ツール切り離し時の待機時間[ms],!!!!
   ```

   使用例
   ```
   SSET,TLIF_DTCHTM,1500,!!!!
   ```

⚠ 注意
- SSET,TLIF_DTCHTM コマンドは、EE コネクタからツールに対する電源供給を遮断した後、ツールに搭載されるコンデンサが放電完了して、安全に取り外せる状態になるまでの待機時間を設定します。ツールのコンデンサが放電完了する前に取り外した場合、ツールが故障するおそれがあります。
- ツール切り離し時の待機時間が設定されていない場合、手動ツールチェンジ画面で切り離しボタン押下時に警告のポップアップ (12.1.1 節参照)、ツール切り離しアイコン命令実行時に PLUG-023 アラーム (12.2.1 節参照) が表示されます。使用者がツールを安全に着脱できるよう、ツール切り離し時の待機時間の設定をお願いします。

2. TLIF_ATCHTM: ツール I/F 通信機能を使用する機器(1 台目)について、EE コネクタからツールに対する電源供給を開始してから、ツールが使用可能になるまでの待機時間[ms]を設定します。

   構文
   ```
   SSET,TLIF_ATCHTM,ツール結合時の待機時間[ms],!!!!
   ```

   使用例
   ```
   SSET,TLIF_ATCHTM,800,!!!!
   ```

   ツール結合時の待機時間の設定は必須ではありませんが、ツールによっては、電源投入直後の初期化処理のため、電源供給開始から一定時間後に使用可能になる場合があります。そのような場合に、SSET,TLIF_ATCHTM コマンドを使用してください。

12.5.3 ツール I/F 使用機器 2 台目以降のサブ ID 設定

ツール I/F 通信機能を使用する機器を 2 台以上使用する場合、2 台目以降のそれぞれのツールに対して、他の機器と重複しないサブ ID を設定する必要があります。2 台目以降のサブ ID は、本節で説明する PLGSUB コマンドで設定します。

⚠ 注意
本節に記載する PLGSUB コマンドは V9.40P/38 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本コマンドを使用する場合、ツールメーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/37 版以前の場合は、プラグインソフトをインストールする前に V9.40P/38 版以降の最新版にアップデートしてください。」、と明記してください。

構文

```
PLGSUB,サブ ID,サブ機器の名称,!!!!
```

使用例

```
PLGSUB,FANUC_SmplGrp2,FANUC Device,!!!!
```

サブ ID	2 台目以降のツール I/F 使用機器を識別するための識別子(ID)を指定します。この識別子は他社のツールを含めユニークである必要があります。識別子は"(会社名)_(機器名)"とすることによりユニークな名前としてください。使用できる文字列は英数字およびアンダーバー"_"で 16 文字以内です。 例) FANUC_SmplGrp2 **1234567890123456**
サブ機器の名称	2 台目以降のツール I/F 使用機器の名称を指定します。ここで設定した名称は、手動ツールチェンジ画面の結合ボタン横のツールリスト(12.1.2 節参照)、ツール結合アイコン命令の詳細画面のツールリスト(12.2.2 節参照)の表示に使用されます。

注
- ツール I/F 通信機能を使用する機器を 2 台以上使用する場合、本節で説明する PLGSUB コマンドを使用して、必ず 2 台目以降のそれぞれのツールのサブ ID と名称を設定してください。PLGSUB コマンドでサブ ID を定義せずに、次節以降の SSET コマンドで通信設定を行った場合、ロボット制御装置の電源投入時にツールチェンジ機能の初期化に失敗して、「PLUG-024 ﾂｰﾙﾁｪﾝｼﾞ機能の初期化に失敗しました」が発生します。
- PLGSUB コマンドで定義するサブ ID は、プラグイン ID と同様に"(会社名)_(機器名)"として定義する必要があります。他の機器と重複しないサブ ID を設定してください。

また、ツール I/F 使用機器 2 台目以降の機器名称を多国語対応する場合、PLGINFO コマンド同様に、plginfo**.dat ファイルに PLGSUB コマンドを指定できます。plginfo**.dat ファイルおよび PLGINFO コマンドについては、本書 5.1.2 節「PLGINFO/PLGSUB コマンドの多国語対応」を参照してください。

注
- プラグインソフトが制御するツール I/F 使用機器が 2 台以上ある場合に限り、plginfo**.dat ファイルに PLGSUB コマンドを指定できます。
- plginfo**.dat ファイルの 1 行目は必ず PLGINFO コマンドを指定してください。plginfo**.dat ファイルに PLGSUB コマンドを指定する場合、PLGINFO コマンドの次の行以降に指定してください。
- PLGINFO/PLGSUB コマンドの行間に空白行を入れないように記述してください。

12.5.4 ツール I/F 使用機器 2 台目以降のツール I/F 通信設定

ツール I/F 通信機能を使用する機器を 2 台以上使用する場合の 2 台目以降の通信設定は、本節で説明する SSET コマンドで設定します。基本的には 11 章「ツール I/F 通信機能」に記載するコマンドと同じ構文ですが、通信設定を行う機器を指定するため、先頭のパラメータに必ずサブ ID を指定します。

⚠ 注意
本節に記載する SSET コマンドは V9.40P/73 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本コマンドを使用する場合、ツールメーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/72 版以前の場合は、プラグインソフトをインストールする前に V9.40P/73 版以降の最新版にアップデートしてください。」、と明記してください。

注
- 本節で説明する SSET コマンドのサブ ID は、プラグイン ID と同様に"(会社名)_(プラグイン名)"として定義する必要があります。他の機器と重複しないサブ ID を設定してください。
- 本節で説明する SSET コマンドのサブ ID は、前節で説明する PLGSUB コマンドで指定したサブ ID と同じ ID を指定してください。

1. TLIF_SUB_COMMON: ツール I/F 通信機能を使用する機器 (2 台目以降) の共通設定を行います。SSET,TLIF_SUB_COMMON コマンド内の各設定項目の説明は 11.1 節「ツール I/F 通信機能の共通設定」を参照してください。

   構文
   ```
   SSET,TLIF_SUB_COMMON,サブ ID,EE 有効/無効,電源供給モード,Power1 の電圧設定,Power2 の電圧設定,2A 供給モード時の電圧設定,!!!!
   ```

   使用例
   ```
   SSET,TLIF_SUB_COMMON,FANUC_SmplGrp2,1,1,2,0,0,!!!!
   ```

2. TLIF_SUB_DOCFG: ツール I/F 通信機能を使用する機器 (2 台目以降) でデジタル出力 1 (RO[1])、デジタル出力 2 (RO[2])を使用する場合に必要な通信設定を行います。SSET,TLIF_SUB_DOCFG コマンド内の各設定項目の説明は 11.2 節「デジタル出力信号の通信設定」を参照してください。

   構文
   ```
   SSET,TLIF_SUB_DOCFG,サブ ID,デジタル出力 1 モード設定,デジタル出力 2 モード設定,デジタル出力電圧設定,!!!!
   ```

   使用例
   ```
   SSET,TLIF_SUB_DOCFG,FANUC_SmplGrp2,2,2,2,!!!!
   ```

3. TLIF_SUB_AICFG: ツール I/F 通信機能を使用する機器 (2 台目以降) でアナログ入力 1、アナログ入力 2 を使用する場合に必要な通信設定を行います。SSET,TLIF_SUB_AICFG コマンド内の各設定項目の説明は 11.3 節「アナログ入力信号の通信設定」を参照してください。

   構文
   ```
   SSET,TLIF_SUB_AICFG,サブ ID,1/2 番ピン使用方法の設定,アナログ入力 1 モード設定,アナログ入力 2 モード設定,!!!!
   ```

   使用例
   ```
   SSET,TLIF_SUB_AICFG,FANUC_SmplGrp2,2,1,1,!!!!
   ```

4. TLIF_SUB_RS485: ツール I/F 通信機能を使用する機器 (2 台目以降) でシリアル通信(RS485)を使用する場合に必要な通信設定を行います。SSET,TLIF_SUB_RS485 コマンド内の各設定項目の説明は 11.4 節「シリアル通信の通信設定」を参照してください。

   構文
   ```
   SSET,TLIF_SUB_RS485,サブ ID,1/2 番ピン使用方法の設定,パリティ設定,ストップビット設定,ボーレート設定,!!!!
   ```

   使用例
   ```
   SSET,TLIF_SUB_RS485,FANUC_SmplGrp2,1,0,0,0,!!!!
   ```

12.5.5 ツール I/F 使用機器 2 台目の切り離し・結合時の待機時間の設定

ツール I/F 通信機能を使用する機器を 2 台以上使用する場合の 2 台目以降のツール切り離し・結合時の待機時間は、本節で説明する SSET コマンドで設定します。基本的には 12.5.2 節「ツール I/F 使用機器 1 台目の切り離し・結合時の待機時間の設定」に記載するコマンドと同じ構文ですが、設定を行う機器を指定するため、先頭のパラメータに必ずサブ ID を指定します。

⚠ 注意
本節に記載する SSET,TLIF_SUB_DTCHTM コマンドと SSET,TLIF_SUB_ATCHTM コマンドは V9.40P/38 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本コマンドを使用する場合、ツールメーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/37 版以前の場合は、プラグインソフトをインストールする前に V9.40P/38 版以降の最新版にアップデートしてください。」、と明記してください。

1. TLIF_SUB_DTCHTM: ツール I/F 通信機能を使用する機器 (2 台目以降) について、EE コネクタからツールに対する電源供給を遮断し、ツールを安全に取り外せる状態になるまでの待機時間[ms]を設定します。

   構文
   ```
   SSET,TLIF_SUB_DTCHTM,サブ ID,電源供給を遮断した後の待機時間[ms],!!!!
   ```

   使用例
   ```
   SSET,TLIF_SUB_DTCHTM,FANUC_SmplGrp2,1500,!!!!
   ```

⚠ 注意
- **SSET,TLIF_SUB_DTCHTM コマンドは、EE コネクタからツールに対する電源供給を遮断した後、ツールに搭載されるコンデンサが放電完了して、安全に取り外せる状態になるまでの待機時間を設定します。ツールのコンデンサが放電完了する前に取り外した場合、ツールが故障するおそれがあります。**
- **ツール切り離し時の待機時間が設定されていない場合、手動ツールチェンジ画面で切り離しボタン押下時に警告のポップアップ (12.1.1 節参照)、ツール切り離しアイコン命令実行時に PLUG-023 アラーム (12.2.1 節参照) が表示されます。使用者がツールを安全に着脱できるよう、ツール切り離し時の待機時間の設定をお願いします。**

2. TLIF_SUB_ATCHTM: ツール I/F 通信機能を使用する機器 (2 台目以降) について、EE コネクタからツールに対する電源供給を開始してから、ツールが使用可能になるまでの待機時間[ms]を設定します。

   構文
   ```
   SSET,TLIF_SUB_ATCHTM,サブ ID,電源供給を開始してから使用可能になるまでの待機時間[ms],!!!!
   ```

   使用例
   ```
   SSET,TLIF_SUB_ATCHTM,FANUC_SmplGrp2,800,!!!!
   ```

   ツール結合時の待機時間の設定は必須ではありませんが、ツールによっては、電源投入直後の初期化処理のため、電源供給開始から一定時間後に使用可能になる場合があります。そのような場合に、SSET,TLIF_SUB_ATCHTM コマンドを使用してください。

12.5.6 ツールの結合状態を確認する KAREL 関数

本節に説明する KAREL 関数 IPL_START_SERIALCOMM と IPL_END_SERIALCOMM を使用して、KAREL プログラムでツールの結合状態を確認できます。

目的:

指定したプラグイン ID のツールが電気的に結合されていて通信が可能な状態か否かを確認します。指定したツールが結合されていてシリアルデータの送受信が可能な状態の場合、そのツールを通信中の状態にしてツール切り離し命令によるツールの切り離しが行えない状態に遷移します。次に説明する IPL_END_SERIALCOMM 関数を呼ぶまでこの状態が続きます。

構文:

IPL_START_SERIALCOMM(*Plugin_id, Ok_to_start, Status*) from IPL_FANUC_SYS_PLGUTIL
入出力パラメータ:
[in] *Plugin_id* : STRING
[out] *Ok_to_start* : INTEGER
[out] *Status* : INTEGER

詳細:

- *Plugin_id* に結合状態を確認するツールのプラグイン ID (Autocfg.csv で PLGID コマンドで指定されたプラグイン ID)を指定します。ここで指定されたプラグイン ID を持つツールの結合状態を返します。
- *Ok_to_start* に指定されたプラグイン ID を持つツールが結合されていて通信可能か否かを返します。
  0: 通信が不可能 (切り離し状態)
  1: 通信が可能 (結合状態)
- *Status* に処理ステータスを返します。0 は正常、それ以外はエラーが発生したことを示します。

目的:

指定したプラグイン ID のツールが結合されているとき、シリアル通信が完了してツール切り離し命令によるツール切り離しができる状態に遷移します。

構文:

IPL_END_SERIALCOMM(*Plugin_id, Status*) from IPL_FANUC_SYS_PLGUTIL
入出力パラメータ:
[in] *Plugin_id* : STRING
[out] *Status* : INTEGER

詳細:

- *Plugin_id* に、シリアル通信が完了してツール切り離し命令によるツール切り離しができる状態に遷移させたいツールのプラグイン ID を指定します。
- *Status* に処理ステータスを返します。0 は正常、それ以外はエラーが発生したことを示します。

⚠ 注意

- 本節に記載する IPL_START_SERIALCOMM 関数と IPL_END_SERIALCOMM 関数は V9.40P/71 版以降のロボット制御装置ソフトウェアで使用できます。プラグインソフトで本コマンドを使用する場合、周辺機器メーカが提供する取扱説明書で、「ロボット制御装置ソフトウェアが V9.40P/70 版以前の場合は、プラグインソフトをインストールする前に V9.40P/71 版以降の最新版にアップデートしてください。」、と明記してください。
- 一度 IPL_START_SERIALCOMM 関数を呼んで通信中の状態にした場合、次に IPL_END_SERIALCOMM 関数を呼ぶまでツール切り離しができません。IPL_START_SERIALCOMM 関数を呼んだ後は、必ず IPL_END_SERIALCOMM 関数を呼ぶように実装して下さい。
- IPL_START_SERIALCOMM 関数を呼び出しした後に、IPL_END_SERIALCOMM 関数を呼ぶことなく、TLIF_WRITE/TLIF_READ ビルトインを繰り返し実行するような設計はしないで下さい。
  IPL_START_SERIALCOMM 関数を呼び出して IPL_END_SERIALCOMM 関数を呼ぶまでの間、TLIF_WRITE/TLIF_READ ビルトインを複数回実行すると、ツール切り離し命令実行時の待機時間が長くなり、TP プログラムの実行を妨げる要因になります。

12.5.7 インストール定義ファイルの実装例

サブ ID を含めた Autocfg.csv の実装例を以下に示します。

```
REM,autocfg.csv,V1.0
PLGINFO,SmplGrp,FANUC Sample Gripper,FANUC Corporation,FANUC Sample Gripper,!!!!
PLGID,FANUC_SmplGrp,01.01.01,!!!!
DESCFILE,DESCRIPTION.HTM,sample_img_a.png,sample_img_b.png,!!!!

SSET,TLIF_COMMON,1,1,2,0,0,!!!!
SSET,TLIF_DOCFG,2,2,2,!!!!
SSET,TLIF_RS485,1,2,1,5,!!!!
SSET,TLIF_DTCHTM,1300,!!!!
SSET,TLIF_ATCHTM,500,!!!!

PLGSUB,FANUC_SUB2,FANUC EQ2,!!!!
SSET,TLIF_SUB_COMMON,FANUC_SUB2,1,1,2,0,0,!!!!
SSET,TLIF_SUB_DOCFG,FANUC_SUB2,1,2,2,!!!!
SSET,TLIF_SUB_AICFG,FANUC_SUB2,0,0,0,!!!!
SSET,TLIF_SUB_DTCHTM,FANUC_SUB2,1000,!!!!
SSET,TLIF_SUB_ATCHTM,FANUC_SUB2,0,!!!!

PLGSUB,FANUC_SUB3,FANUC EQ3,!!!!
SSET,TLIF_SUB_COMMON,FANUC_SUB3,1,2,0,0,2,!!!!
SSET,TLIF_SUB_DOCFG,FANUC_SUB3,0,0,0,!!!!
SSET,TLIF_SUB_AICFG,FANUC_SUB3,2,1,1,!!!!
SSET,TLIF_SUB_DTCHTM,FANUC_SUB3,500,!!!!
SSET,TLIF_SUB_ATCHTM,FANUC_SUB3,800,!!!!

PLGSUB,FANUC_SUB4,FANUC EQ4,!!!!
SSET,TLIF_SUB_COMMON,FANUC_SUB4,1,3,2,2,0,!!!!
SSET,TLIF_SUB_DOCFG,FANUC_SUB4,0,0,0,!!!!
SSET,TLIF_SUB_RS485,FANUC_SUB4,1,2,1,7,!!!!
SSET,TLIF_SUB_DTCHTM,FANUC_SUB4,1200,!!!!
SSET,TLIF_SUB_ATCHTM,FANUC_SUB4,500,!!!!

END
```

図 12.5.7 (a) Autocfg.csv の実装例

13 R-30iB Mini Plus 周辺機器インタフェース

本章では、ロボット制御装置 R-30*i*B Mini Plus の周辺機器インタフェースを説明します。
下表に R-30*i*B Mini Plus の周辺機器インタフェースを示します。

表 13 (a) 周辺機器インタフェースの種類

名称	手配仕様	I/O 点数		備考
		JRM18		
		DI	DO	
メインボード A	A05B-2695-H001	20	16 (ソース型)	Vision I/F、力センサ I/F、 ライントラッキング I/F、HDI 付

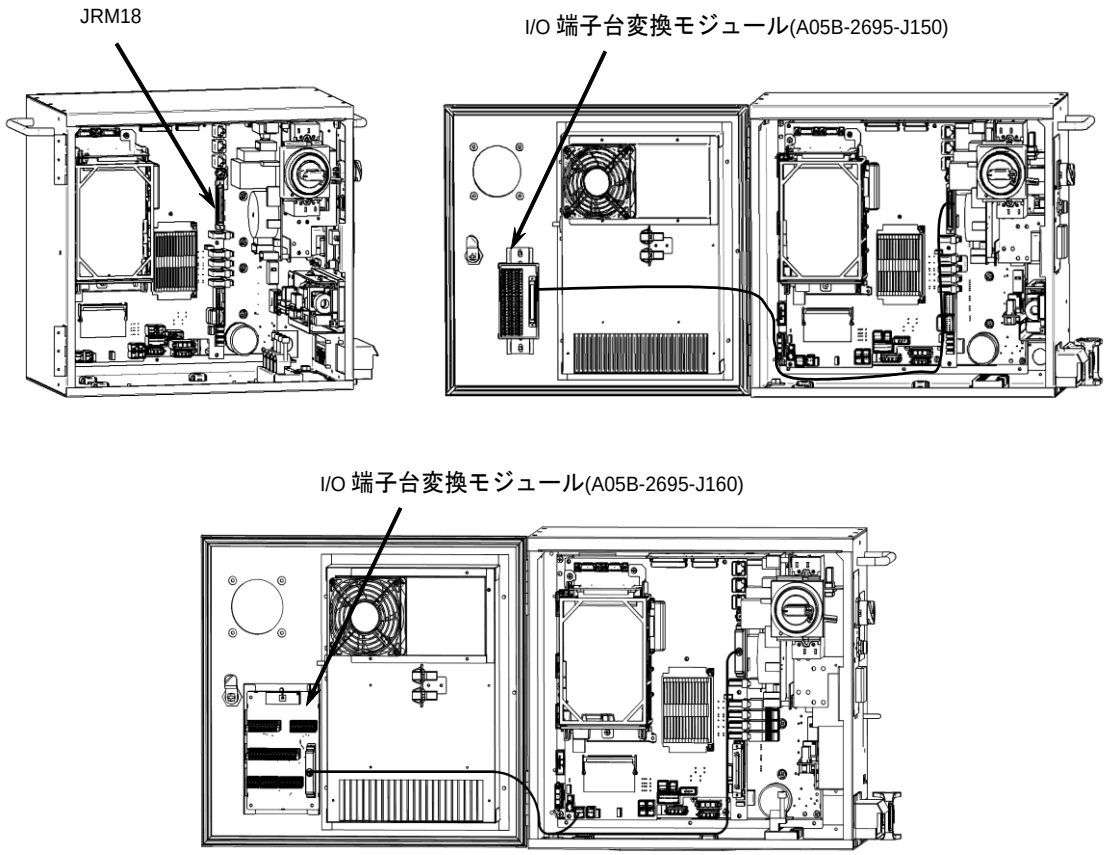


図 13 (a) 周辺機器インタフェース(JRM18)
左上：I/O 端子台変換モジュールなしの場合
右上：I/O 端子台変換モジュール(A05B-0695-J150)内蔵ありの場合
下：I/O 端子台変換モジュール(A05B-0695-J160)内蔵ありの場合

> ⚠ **注意**
> R-30*i*B Mini Plus では、DI/DO, Safety DI/DO と合わせて 0.7A 以下であれば制御装置から外部周辺機器への 24V 電源供給が可能です。接続先に関する情報は 13.5 節「外部への 24V 供給」を参照ください。0.7A 以上になる場合には、外部電源の準備をお願いします。

13.1 周辺機器インタフェース(JRM18)

メインボードの周辺機器インタフェース(JRM18)の I/O 信号には DI: 20 点、DO: 16 点の入出力信号があります。
下表に周辺機器インタフェースの DI/DO 信号の意味を示します。

表 13.1 (a) 周辺機器インタフェースの DI 信号の意味

コネクタ番号	信号名称	標準 I/O 割付		信号の意味	備考
		UOP 自動割付: 簡略(JRM18)	UOP 自動割付: なし 全て 全て(スレーブ) 簡略 簡略(スレーブ)		
JRM18-8	DI01	DI[101]	DI[101]	周辺機器状態	汎用信号
JRM18-42	DI02	DI[102]	DI[102]		
JRM18-9	DI03	DI[103]	DI[103]		
JRM18-43	DI04	DI[104]	DI[104]		
JRM18-10	DI05	DI[105]	DI[105]		
JRM18-44	DI06	DI[106]	DI[106]		
JRM18-11	DI07	DI[107]	DI[107]		
JRM18-45	DI08	DI[108]	DI[108]		
JRM18-12	DI09	DI[109]	DI[109]		
JRM18-46	DI10	DI[110]	DI[110]		
JRM18-13	DI11	DI[111]	DI[111]		
JRM18-47	DI12	DI[112]	DI[112]		
JRM18-14	DI13	UI[2] *HOLD	DI[81]	一時停止	
JRM18-48	DI14	UI[5] RESET	DI[82]	リセット	
JRM18-15	DI15	UI[6] START	DI[83]	起動	
JRM18-49	DI16	UI[8] ENBL	DI[84]	動作許可	
JRM18-16	DI17	UI[9] PNS1	DI[85]	プログラム番号選択	
JRM18-50	DI18	UI[10] PNS2	DI[86]	プログラム番号選択	
JRM18-17	DI19	UI[11] PNS3	DI[87]	プログラム番号選択	
JRM18-51	DI20	UI[12] PNS4	DI[88]	プログラム番号選択	

表 13.1 (b) 周辺機器インタフェースの DO 信号の意味

コネクタ番号	信号名称	標準 I/O 割付		信号の意味	備考
		UOP 自動割付： 簡略(JRM18)	UOP 自動割付： なし 全て 全て(スレーブ) 簡略 簡略(スレーブ)		
JRM18-18	DO01	DO[101]	DO[101]	周辺機器制御信号	汎用信号
JRM18-52	DO02	DO[102]	DO[102]		
JRM18-19	DO03	DO[103]	DO[103]		
JRM18-53	DO04	DO[104]	DO[104]		
JRM18-20	DO05	DO[105]	DO[105]		
JRM18-54	DO06	DO[106]	DO[106]		
JRM18-21	DO07	DO[107]	DO[107]		
JRM18-55	DO08	DO[108]	DO[108]		
JRM18-22	DO09	DO[109]	DO[109]		
JRM18-56	DO10	DO[110]	DO[110]		
JRM18-23	DO11	DO[111]	DO[111]		
JRM18-57	DO12	DO[112]	DO[112]		
JRM18-24	DO13	UO[1] CMDENBL	DO[81]	自動運転中	
JRM18-58	DO14	UO[6] FAULT	DO[82]	アラーム	
JRM18-25	DO15	UO[9] BATALM	DO[83]	バッテリ電圧低下	
JRM18-59	DO16	UO[10] BUSY	DO[84]	稼働中	

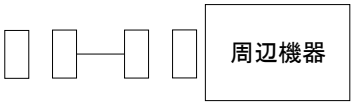
13.2 周辺機器インタフェース(JRM18)と周辺機器の接続

制御装置

周辺機器制御インタフェース (ソース型 DO)

JRM18

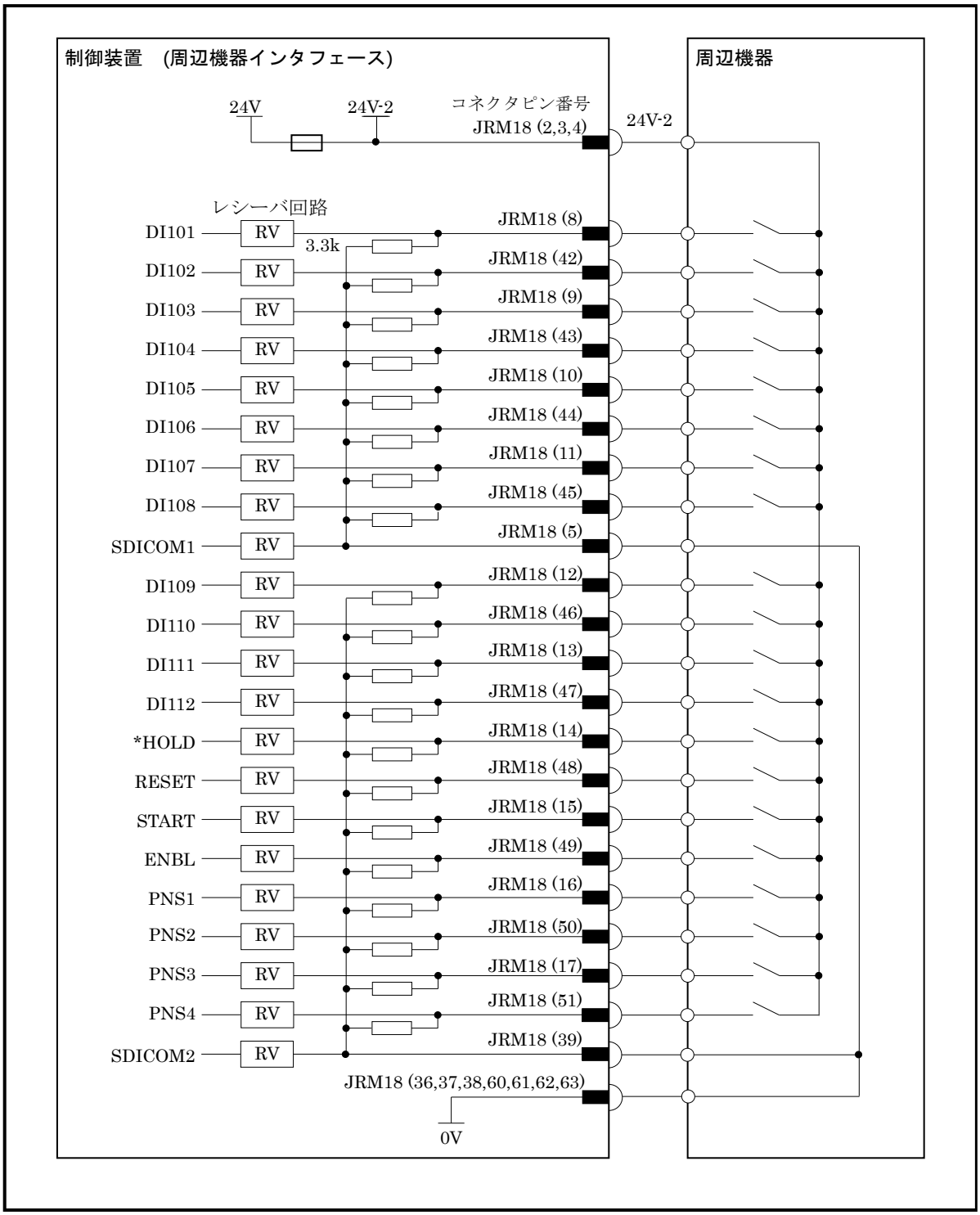
1	24VEXT_IN	35	0VEXT
2	24V-2	36	0V
3	24V-2	37	0V
4	24V-2	38	0V
5	SDICOM1	39	SDICOM2
6	EES1	40	EES2
7	EAS1	41	EAS2
8	DI101	42	DI102
9	DI103	43	DI104
10	DI105	44	DI106
11	DI107	45	DI108
12	DI109	46	DI110
13	DI111	47	DI112
14	*HOLD	48	RESET
15	START	49	ENBL
16	PNS1	50	PNS2
17	PNS3	51	PNS4
18	DO101	52	DO102
19	DO103	53	DO104
20	DO105	54	DO106
21	DO107	55	DO108
22	DO109	56	DO110
23	DO111	57	DO112
24	CMDENBL	58	FAULT
25	BATALM	59	BUSY
26	DOSRC1	60	0V
27	DOSRC1	61	0V
28	DOSRC2	62	0V
29	DOSRC2	63	0V
30	ESPB1	64	ESPB11
31	ESPB2	65	ESPB21
32	ESPB1	66	ESPB11
33	ESPB2	67	ESPB21
34		68	



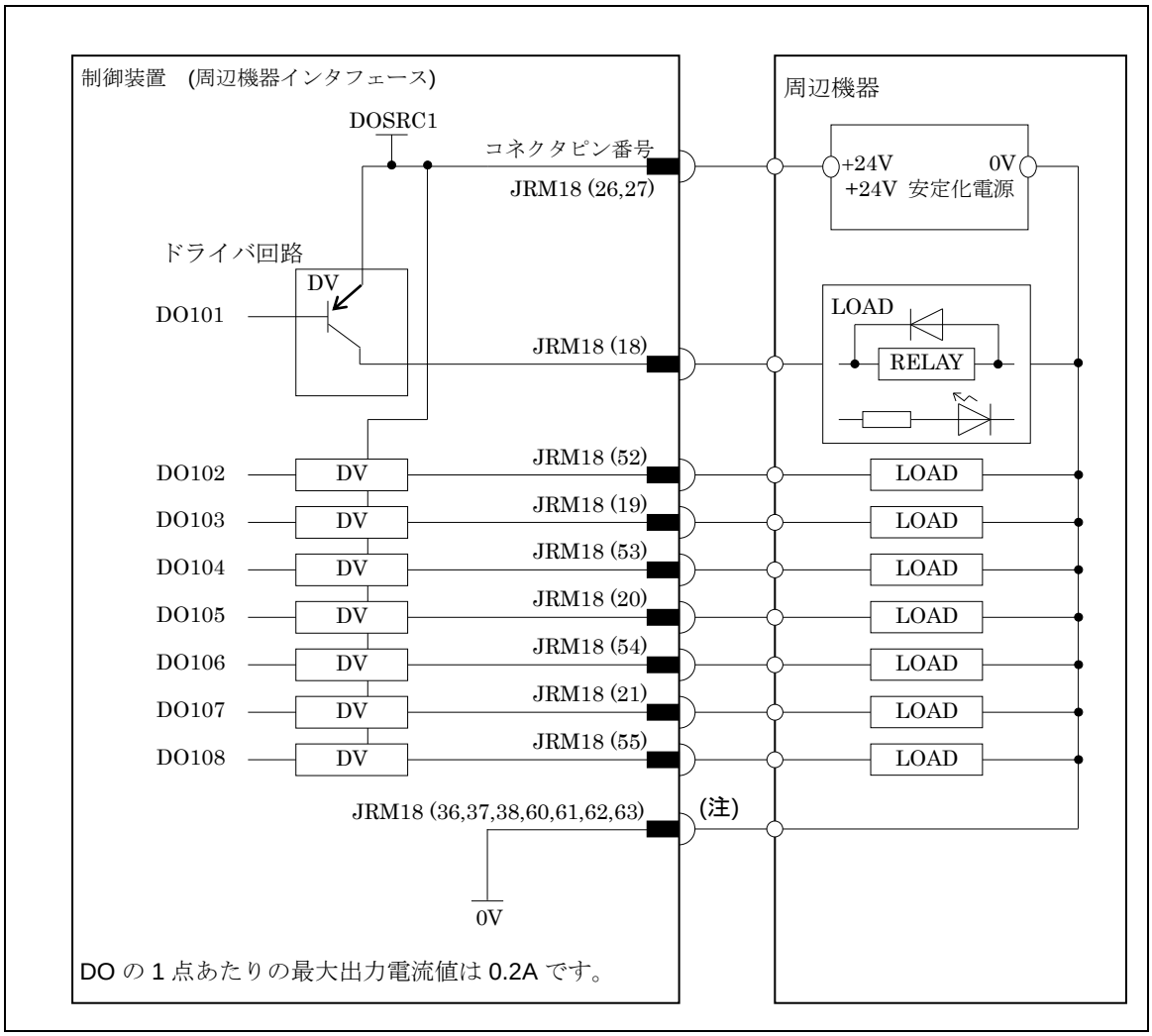
SDICOM1～2 は SDI のコモン切替用信号です。
24V-2 コモン時 ― 0V に接続
0V コモン時 ― 24V-2 に接続

SDICOM1 : DI101～DI108 のコモンを切り替えます。
SDICOM2 : DI109～DI112, *HOLD, RESET, START, ENBL, PNS1～PNS4 のコモンを切り替えます。

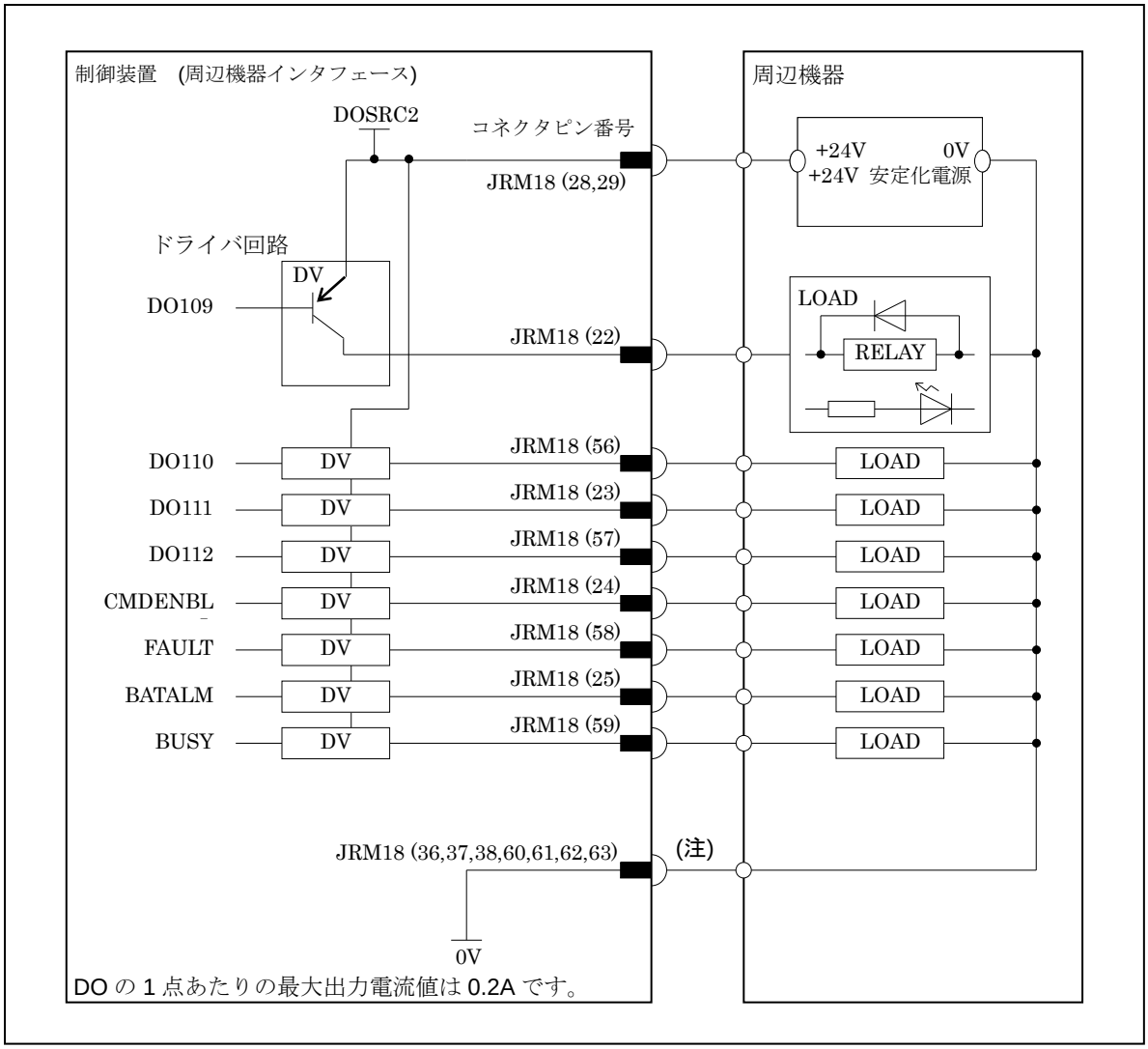
注
1 周辺機器接続ケーブルはオプションです。
2 JRM18 のピン「DOSRC1」「DOSRC2」はドライバへの電源供給端子です。
(全てのピンを接続してください)



注
本図は+24V コモンの場合の接続です。

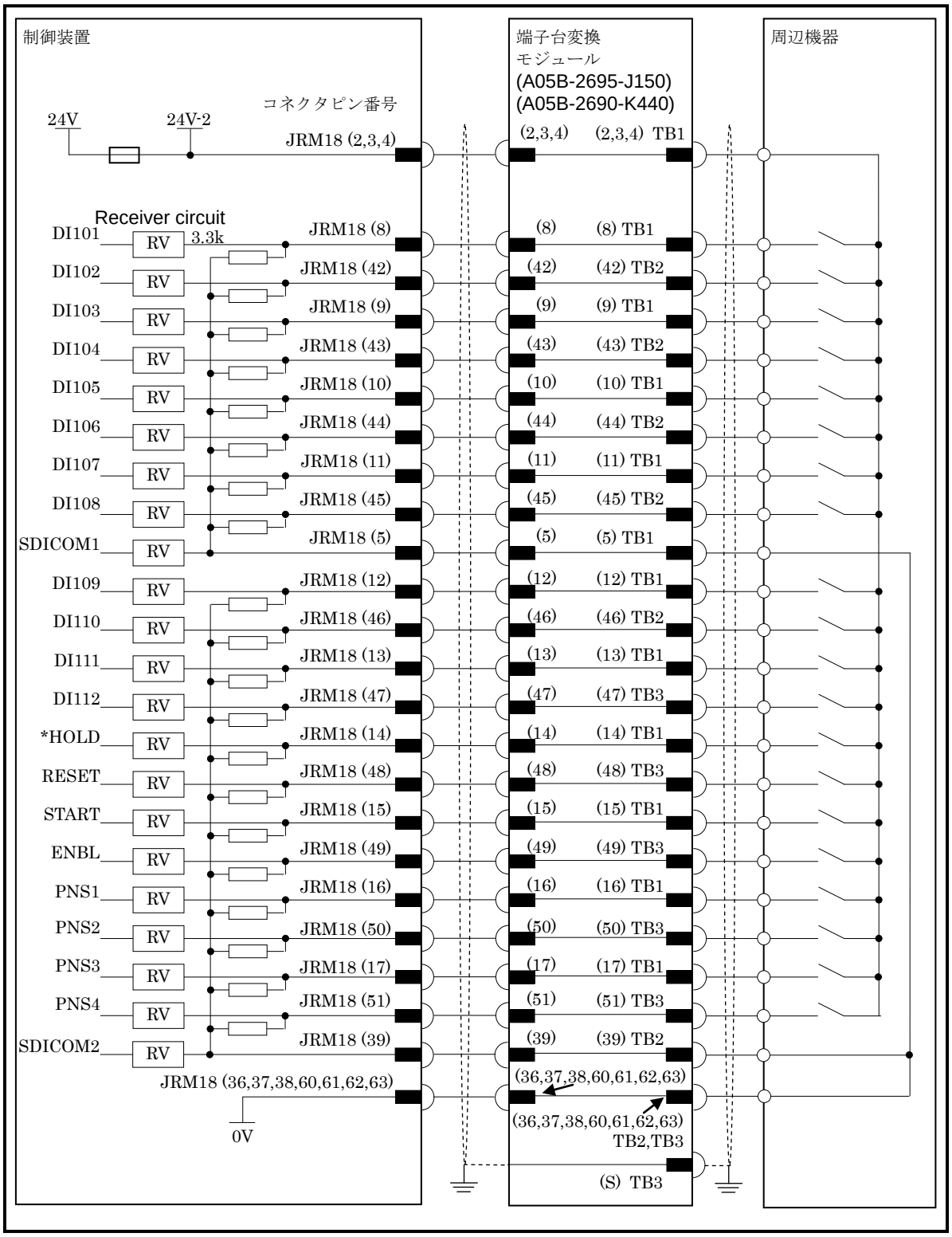


注
JRM18 のピン番号 36～38、60～63 は 0V 用です。全てのピンを接続してください。

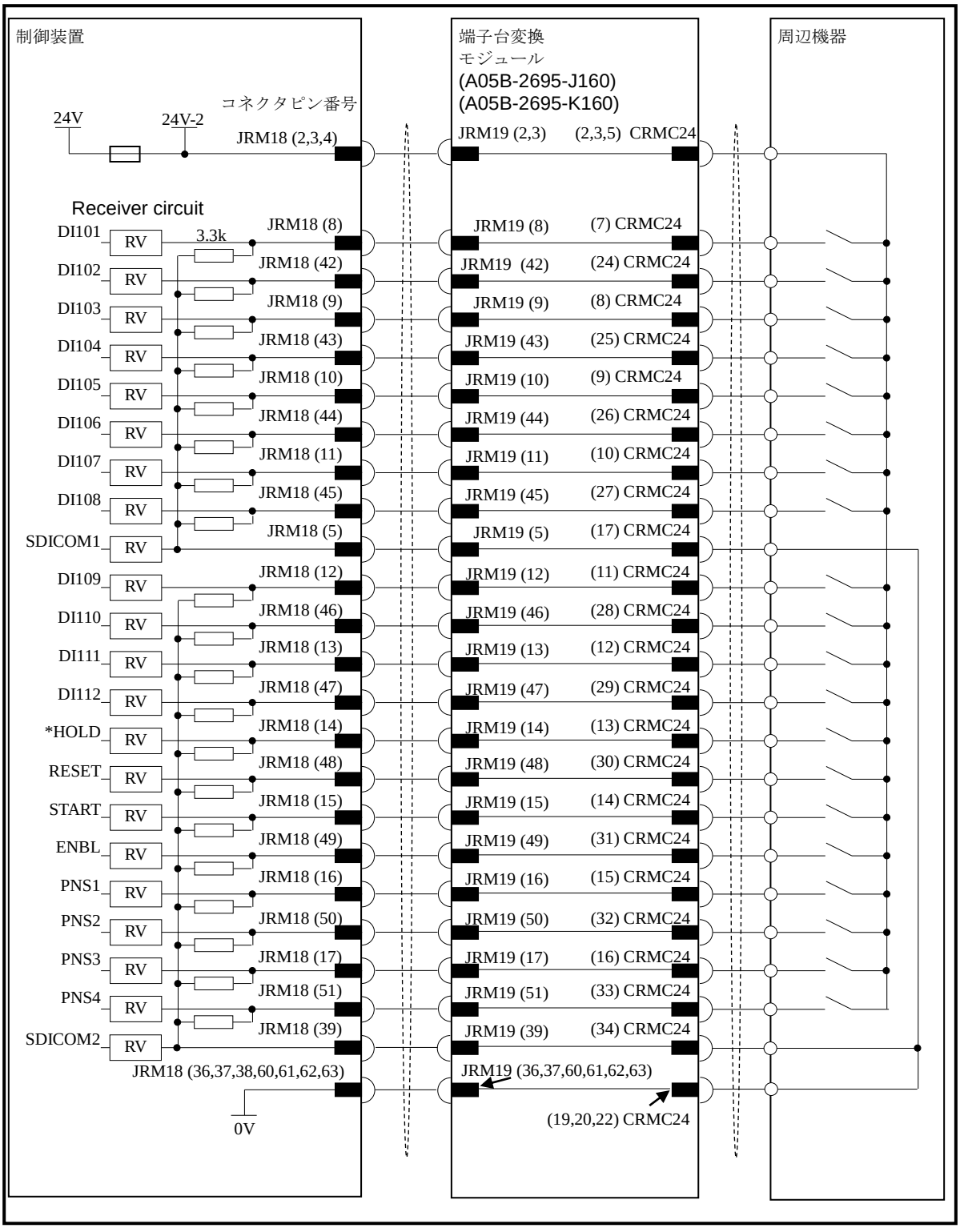


注
JRM18 のピン番号 36～38、60～63 は 0V 用です。全てのピンを接続してください。

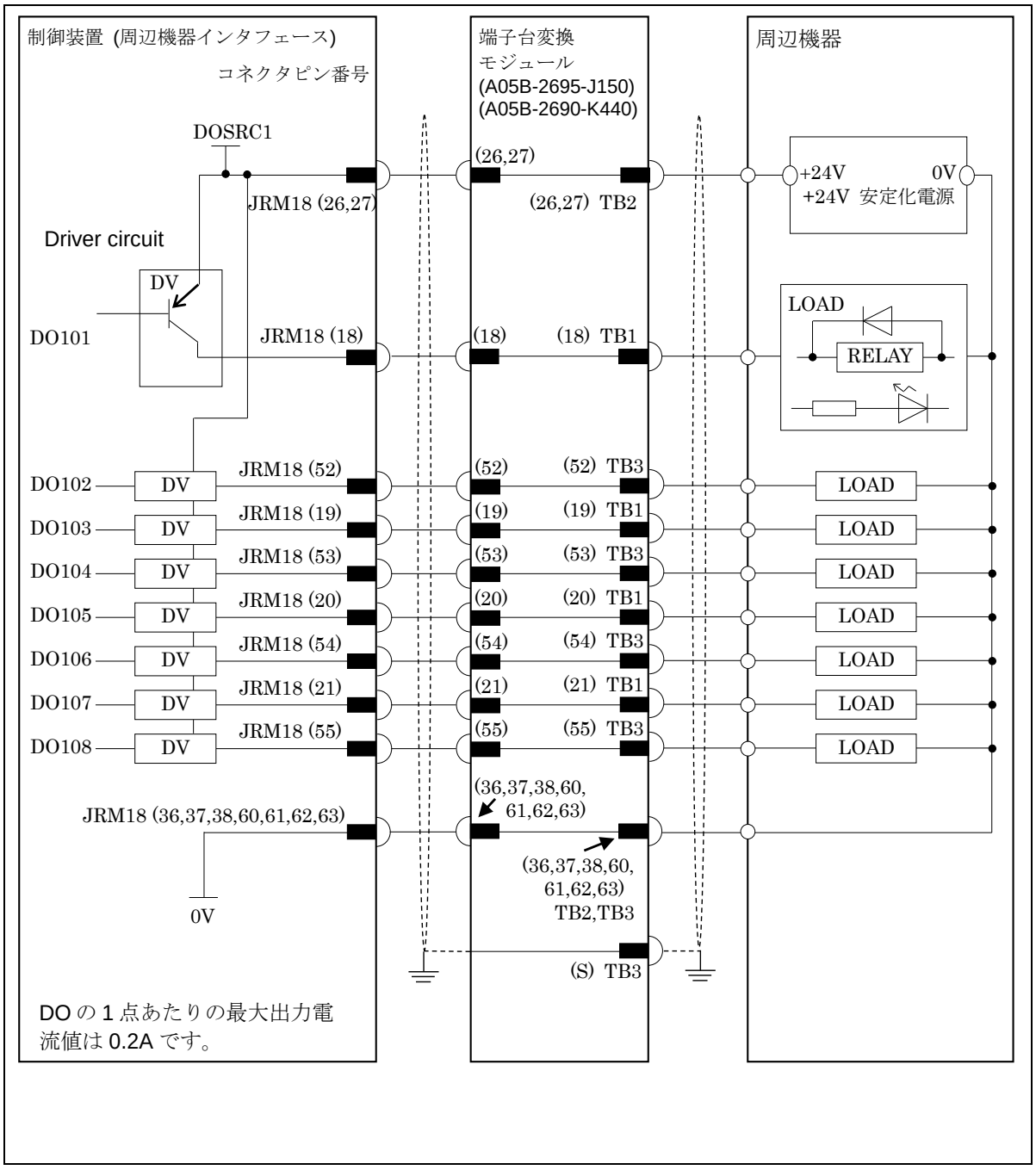
13.3 周辺機器インタフェース(JRM18)と周辺機器の接続(端子台変換モジュールの場合)



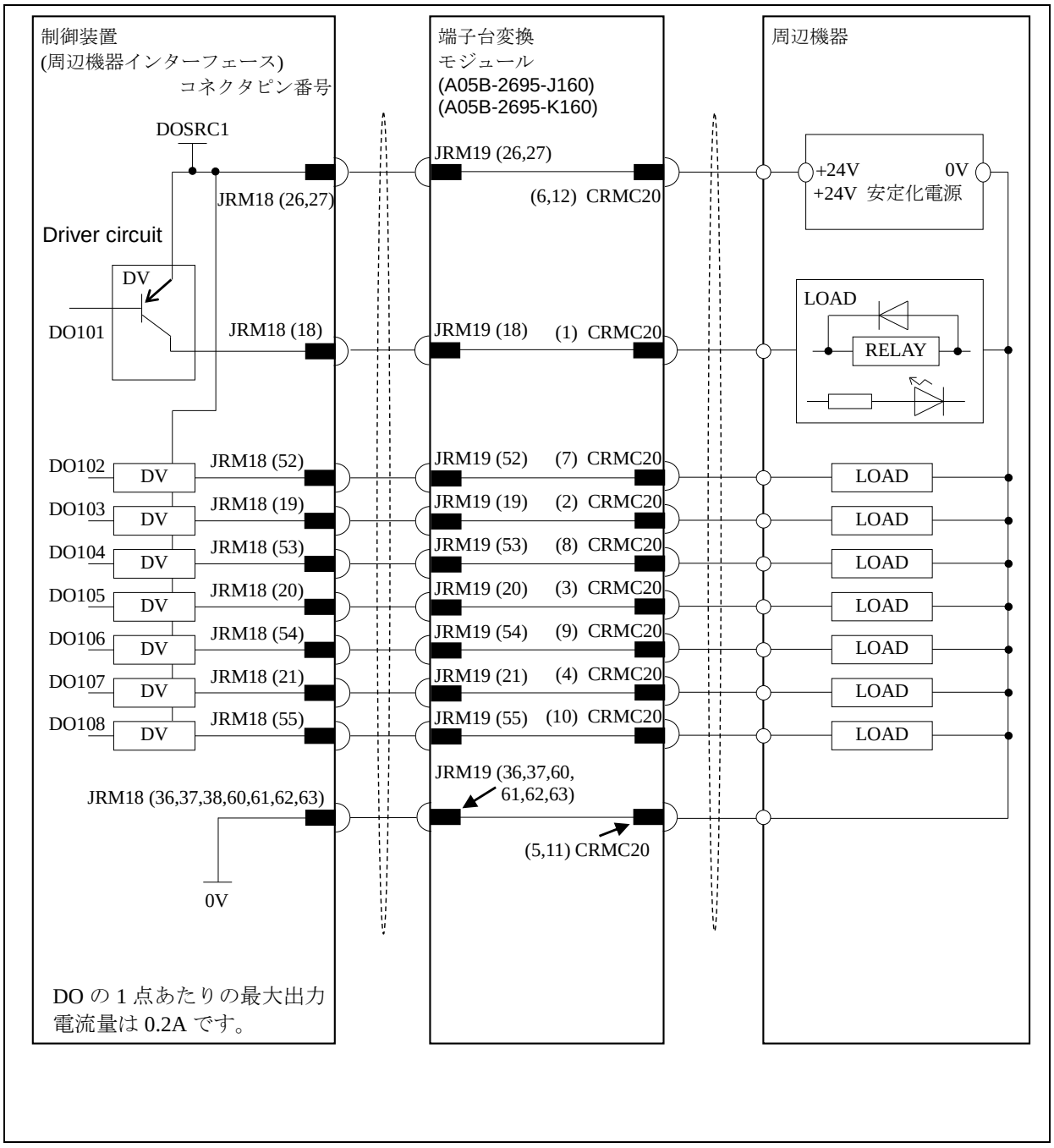
注
本図は+24V コモンの場合の接続です。



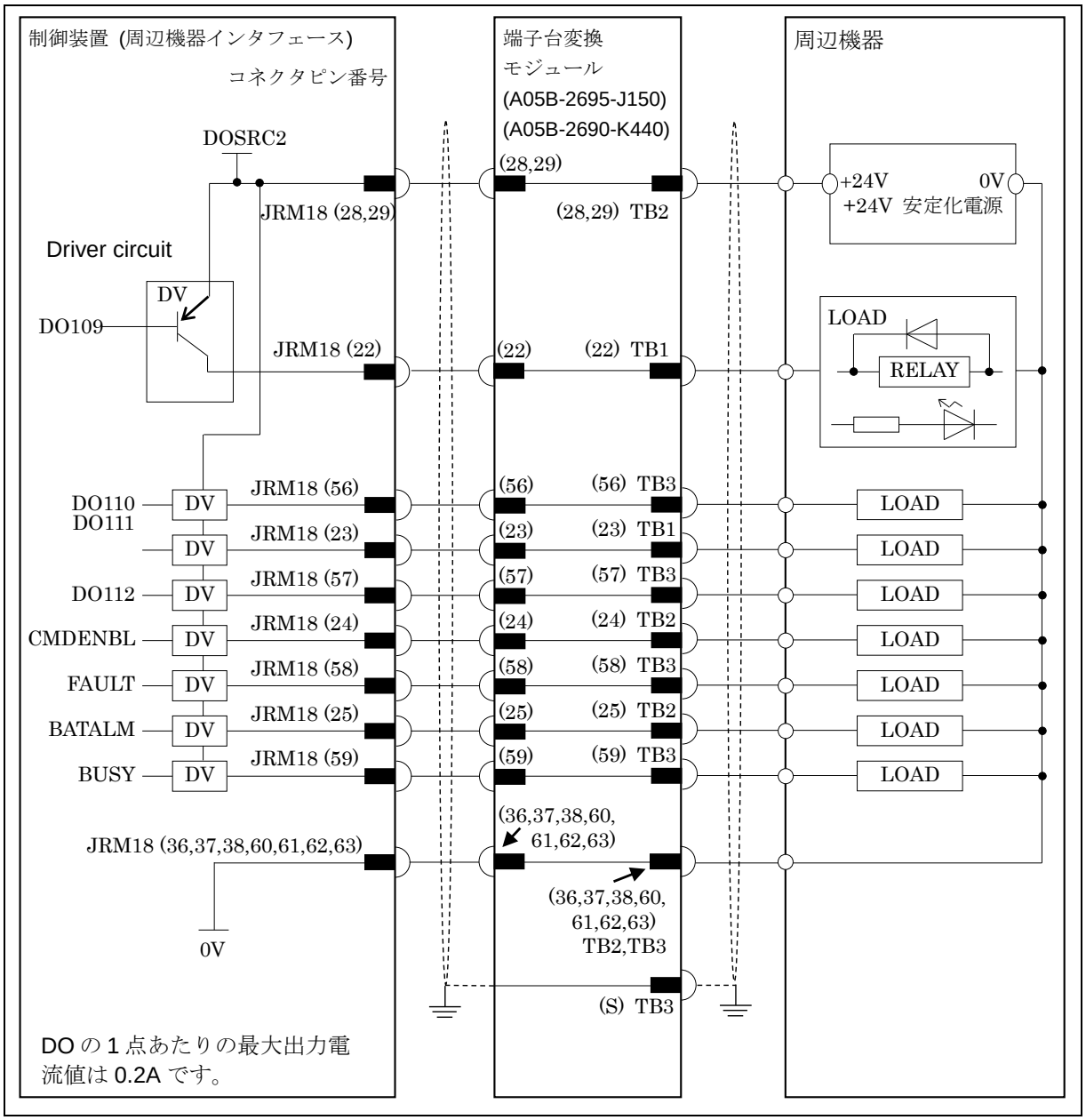
注
本図は+24V コモンの場合の接続です。



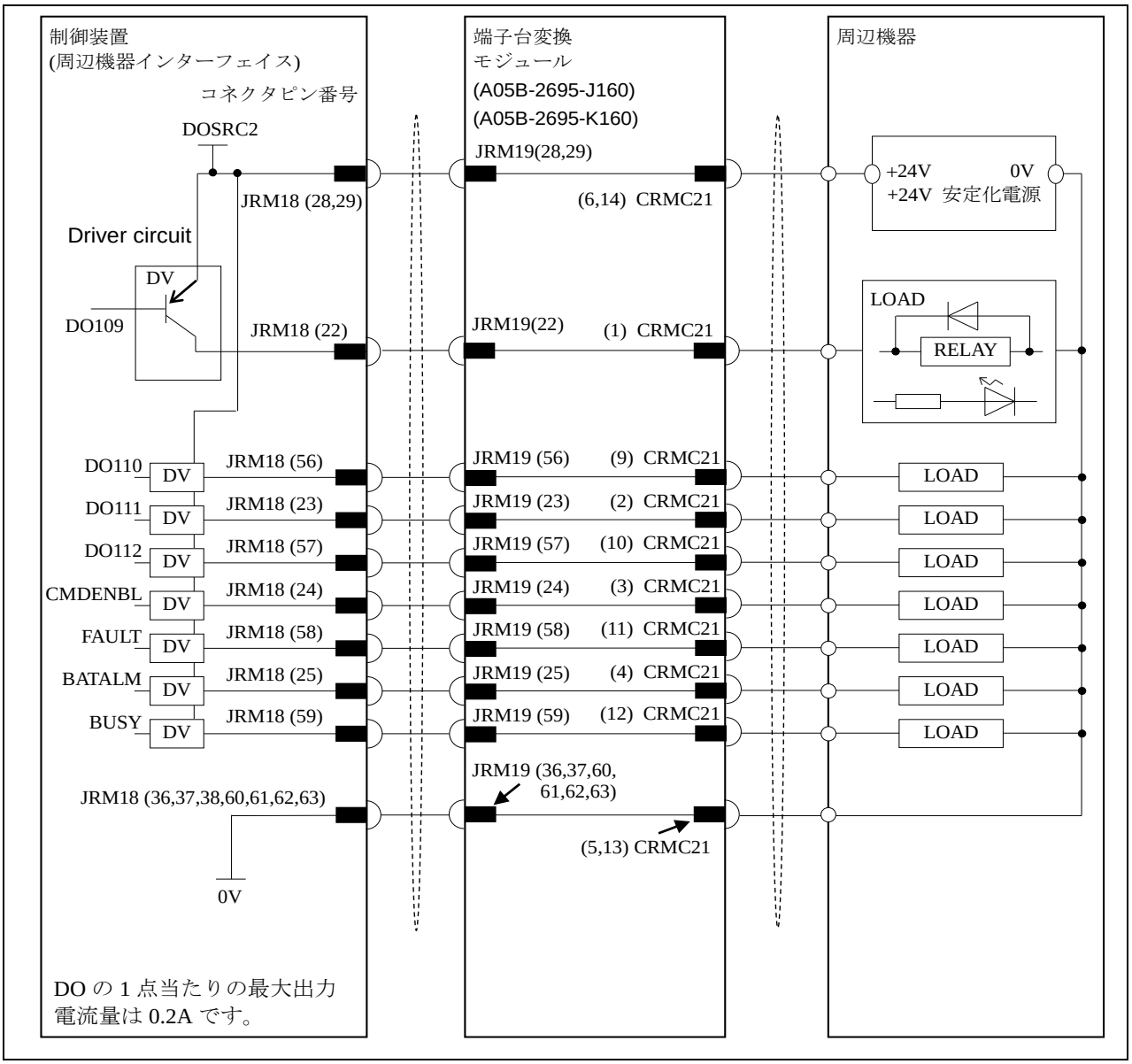
注
JRM18 のピン番号 36～38、60～63 は 0V 用です。全てのピンを接続してください。



注
JRM18 のピン番号 36～38、60～63 は 0V 用です。全てのピンを接続してください。



注
JRM18 のピン番号 36～38、60～63 は 0V 用です。全てのピンを接続してください。

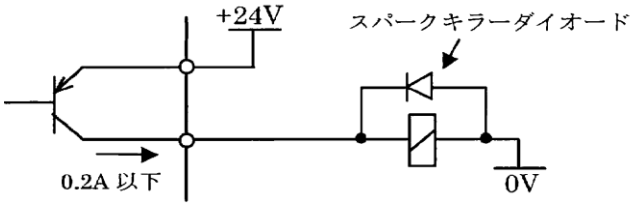


注
JRM18 のピン番号 36～38、60～63 は 0V 用です。全てのピンを接続してください。

13.4 周辺機器のデジタル入出力信号規格

周辺機器のデジタル入出力信号の規格を以下に示します。

(1) 周辺機器インタフェースの出力信号規格 (ソース型信号出力)
(a) 接続例



(b) 電気的規格
ドライバ ON 時最大負荷電流: 200mA (瞬時を含む)

ドライバ ON 時飽和電圧: 1.0Vmax
耐電圧: 24V±20% (瞬時を含む)
ドライバ OFF 時流出漏れ電流: 100μA

(c) 出力信号の外部電源として、以下のような電源を用意してください。
電源電圧: +24V±10%
電源電流: 本プリント板 1 枚あたり
(瞬時を含む最大負荷電流の総和+100mA 以上)
電源投入タイミング: ロボット制御装置と同時、あるいはそれ以前
電源切断タイミング: ロボット制御装置と同時、あるいはそれ以降

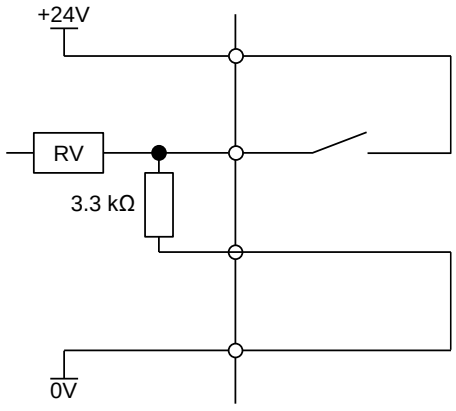
(d) スパークキラーダイオード
定格せん頭逆耐圧: 100V 以上
定格実効順電流: 1A 以上

(e) 出力信号用のドライバについて
ドライバ素子内では 1 点の出力信号ごとにその電流を監視し、過電流を検出するとその出力を OFF させます。過電流による出力 OFF は、OFF したことにより過電流状態でなくなるため ON 状態に復帰します。このため、地絡や過負荷状態においては、その出力は ON/OFF を繰り返すことになります。このような状態は突入電流の大きな負荷を接続した場合にも発生します。
さらに、ドライバ素子内には過熱検出回路があり、出力の地絡などで過電流状態が続き、素子内部の温度が上昇した場合には素子の出力全てを OFF させます。この OFF 状態は保持されますが、素子内部の温度が下がった後でロボット制御装置の電源を ON/OFF することで復帰します。

(f) 使用上の注意
DI/DO, SFDO/SFDO, 周辺機器への 24V 供給併せて 0.7A 以下であれば、制御装置の 24V 電源を使用可能です。0.7A を超える場合は外部電源を使用ください。詳細は 13.5 節「外部への 24V 供給」を参照ください。
リレー、ソレノイド類を直接接続する場合には、逆起電圧防止用ダイオードを負荷と並列に接続してください。
ランプなど ON 時に突入電流が発生する負荷を接続する場合は、保護用抵抗を入れてください。

(g) 適用信号
メインボード JRM18 の出力信号
[CMDENBL、FAULT、BATALM、BUSY、DO101-112]

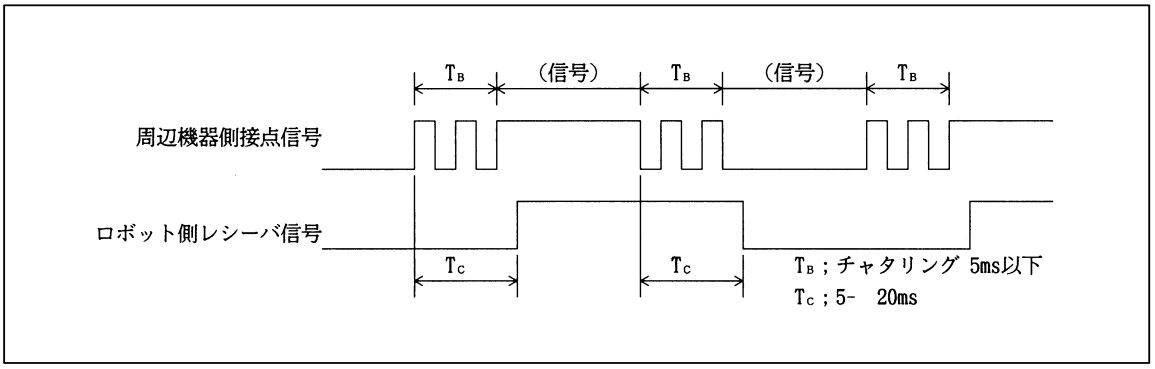
(2) 周辺機器インタフェースの入力信号規格
(a) 接続例 (+24V コモンの場合)



(b) レシーバの電気的規格
形式: 接地形電圧レシーバ
定格入力電圧: 接点 “閉” +20V～+28V
接点 ”開” 0V～+4V
最大入力印加電圧: DC +28V
入力インピーダンス: 約 3.3kΩ
応答時間: 5～20ms

(c) 周辺機器側接点規格

電圧と電流:	DC24V、0.1A (最小負荷が 5mA 以下の接点を使用してください)
入力信号幅:	ON/OFF いずれも 200ms 以上
チャタリング時間:	5ms 以下
閉回路抵抗:	100Ω 以下
開回路抵抗:	100kΩ 以上



(d) 使用上の注意

レシーバに供給する電圧はロボット側の+24V 電源を使用してください。
但し、ロボット側のレシーバ部で上記信号規格を満足する必要があります。

(e) 適用信号

メインボード JRM18 の入力信号
[XHOLD、RESET、START、ENBL、DNS1-DNS4、DI101-DI112]

13.5 外部への 24V 供給

DI/DO, SFDO/SFDO, 周辺機器への 24V 供給併せて 0.7A 以下であれば、制御装置の 24V 電源を使用可能です。
接続先は、I/O 端子台変換モジュールの有無や、端子台変換モジュールタイプに依存します。

表 13.5 (a) 外部への 24V 供給と接続先

条件	接続先
I/O 端子台変換モジュール無しの場合	メインボード上 JRM18 コネクタの”24V-2”と”0V”
I/O 端子台変換モジュール有 (A05B-2695-J150 もしくは A05B-2690-K440) の場合	端子台変換モジュール上 TB1 の”24V-2”と”0V”
I/O 端子台変換モジュール有 (A05B-2695-J160 もしくは A05B-2690-K160) の場合	端子台変換モジュール上 CRMC24 の”24V-2”と”0V”

14 Eclipse 開発環境

本章では、プラグインインストールパッケージに含まれる各種プログラムファイルの開発を補助するための Eclipse 開発環境の設定および利用手順を説明します。プラグインソフト開発 SDK で提供される Eclipse 開発環境は、キーワードやコメントの色指定、各種プログラムファイルのビルドおよびデバッグをサポートします。本開発環境のサポート内容は以下の通りです。

表 14 (a) Eclipse 開発環境のサポート内容

名称	拡張子	色指定	ビルド	デバッグ
KAREL	*.kl	可	可	可
iHMI コンポーネント	*.ishtm	不可	可	不可

⚠ 注意
Eclipse 開発環境は ROBOGUIDE V9 のみサポートしています。

注
本章で説明する Eclipse 開発環境は、プラグインソフトで使用するプログラムファイルの開発を補助するために提供される開発ツールであり、本開発環境の使用は必須ではありません。

14.1 動作環境

14.1.1 Eclipse 開発環境

下記バージョン以上での Eclipse 開発環境を保証します。

- Eclipse IDE for Eclipse Committers (Version: 2019-03 (4.11.0))



図 14.1.1 (a) Eclipse IDE for Eclipse Committers (Version: 2019-03 (4.11.0))

14.1.2 Eclipse 動作環境

下記バージョン以上での Eclipse 動作環境を保証します。

- Eclipse IDE for Enterprise Java Developers (Version: 2019-03 (4.11.0))



図 14.1.2 (a) Eclipse IDE for Enterprise Java Developers (Version: 2019-03 (4.11.0))

14.2 Eclipse 開発環境準備

Eclipse 開発環境を使用するには、以下のインストールが必要です。

- Eclipse
- ROBOGUIDE

14.2.1 Eclipse のインストール

以下の手順で Eclipse と Fanuc support プラグインをインストールします。

手順 14-1 Eclipse のインストール方法

手順

1. 以下のいずれかのソフトをインストールしてください。

   Eclipse (英語版)
   https://www.eclipse.org/downloads/

   Eclipse (日本語プラグイン All in One パッケージ)
   http://mergedoc.osdn.jp/

Eclipse 開発環境を日本語で利用したい場合は、All in One パッケージの Eclipse のインストールをお勧めします。

注
- 英語版の Eclipse を利用する場合は、事前に JDK (Java Development Kit)をインストールしてください。
- Eclipse 自体の使用方法については、本マニュアルには記載しません。

2. インストール後、Eclipse の dropins フォルダ直下に、Fanuc Support プラグイン用ファイル(fanuc_support フォルダ)をフォルダごと格納し、Eclipse をクリーン実行します。

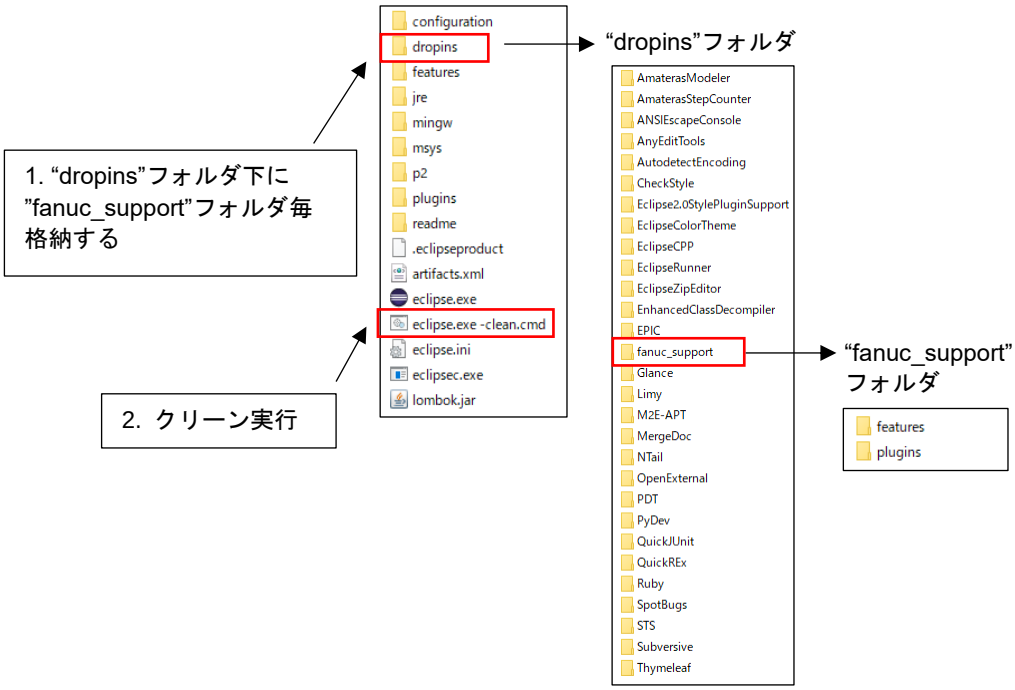


図 14.2.1 (a) Eclipse の dropins フォルダとクリーン実行ファイル

14.2.2 ROBOGUIDE のインストール

Fanuc Support プラグインがインストールされた Eclipse を起動する前に、必ず ROBOGUIDE のインストールを完了してください。ROBOGUIDE がインストールされていない場合、Fanuc Support プラグインがインストールされた Eclipse を起動できません。なお、ROBOGUIDE をインストールしていない環境でも、Fanuc Support プラグインを削除することで Eclipse を起動できます。

14.2.3 Eclipse の起動

インストールした Eclipse フォルダ内にある eclipse.exe を実行して Eclipse を起動します。開発環境を準備する手順については、次節以降を参照してください。

14.3 KAREL 開発環境の利用手順

14.3.1 KAREL 開発環境の準備

以下の手順で KAREL 開発環境を準備します。

手順 14-2 KAREL 開発環境の準備

手順

1. Eclipse のメインメニューから「ファイル(F)」－「新規(N)」－「その他(O)」の順に選択します。



図 14.3.1 (a) Eclipse の起動方法

2. ウィザードより「FANUC」内の「KAREL プロジェクト」を選択します。

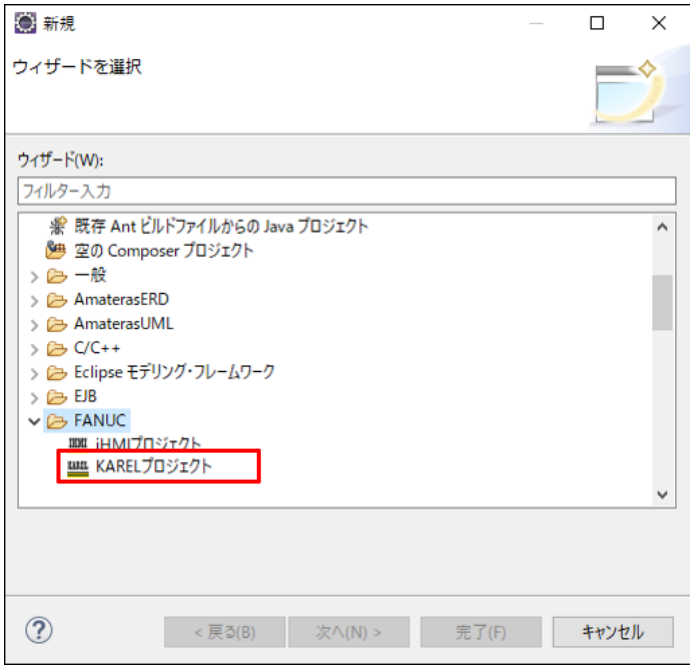


図 14.3.1 (b) KAREL プロジェクト

3. プロジェクト名欄に任意のプロジェクト名を入力します。

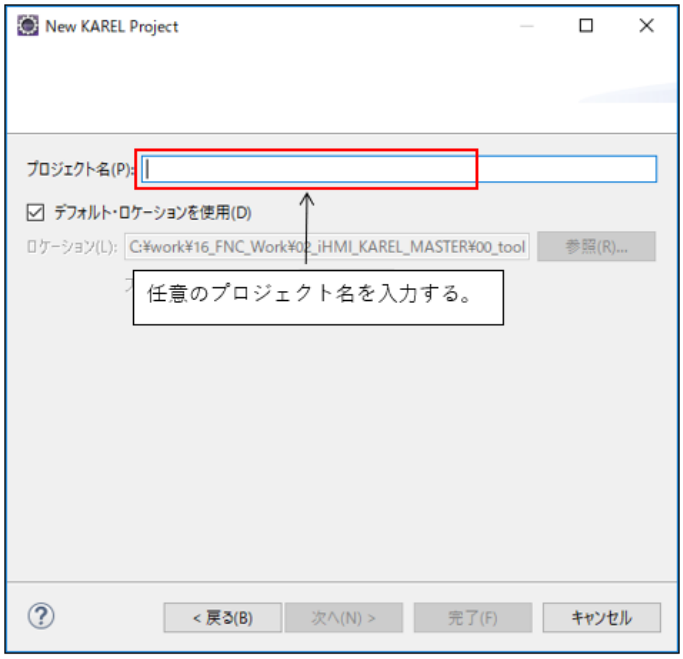


図 14.3.1 (c) KAREL プロジェクト名の入力

4. プロジェクト作成時、自動ビルドが有効になっていると確認メッセージが表示されます。「はい(Y)」をクリックします。自動ビルドの設定は「プロジェクト」－「自動的にビルド」で確認できます。



図 14.3.1 (d) 自動ビルド設定の確認メッセージ

5. 以下のようなフォルダ構成の WorkSpace が作成されます。

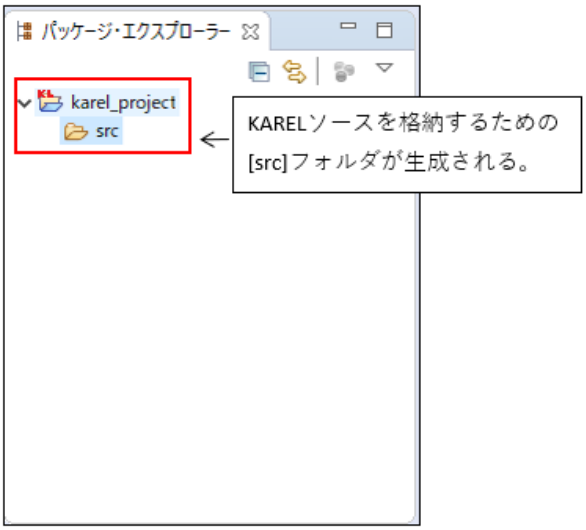


図 14.3.1 (e) KAREL ソースの格納フォルダ

6. ROBOGUIDE / ロボット制御装置を使用してデバッグをするため、タブレット TP 画面上から以下の設定を行ってください。

(1) 「テスト実行」画面で、グループ動作を「無効」、シングルステップ状態を「プログラム毎」に設定します。

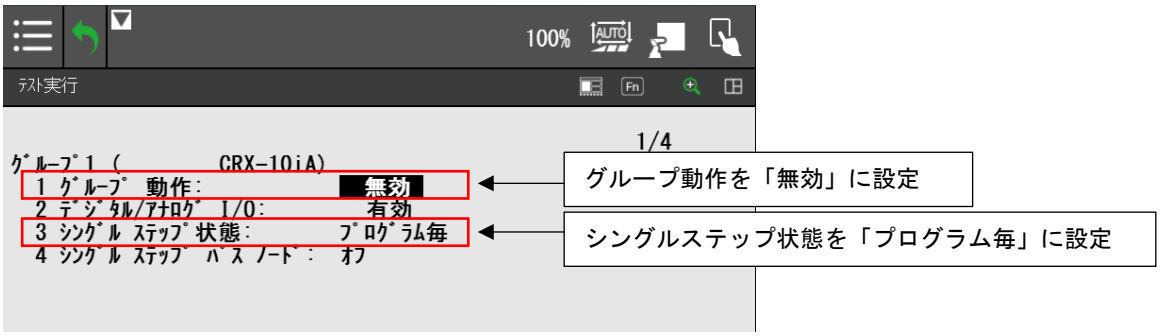


図 14.3.1 (f) グループ動作とシングルステップ状態の設定

(2) 「システム」－「システム設定」画面からリモート/ローカル設定を「リモート」にします。

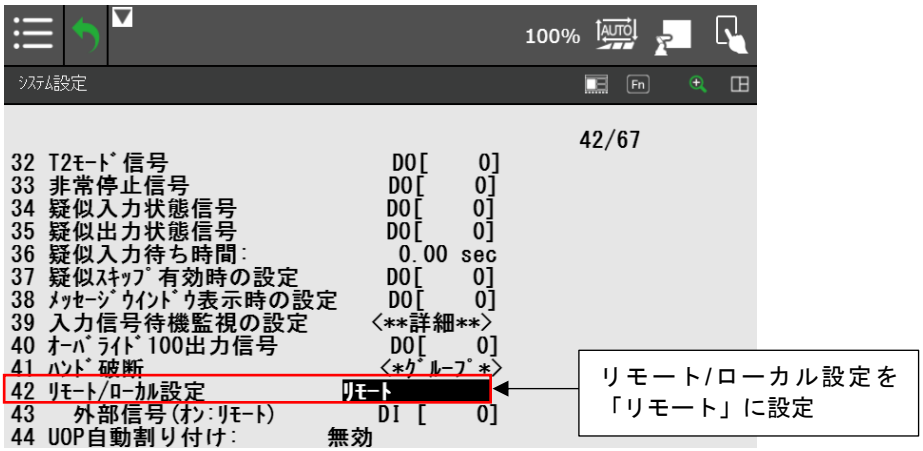


図 14.3.1 (g) リモート / ローカル設定

(3) 「システム」－「システム変数」画面から、システム変数 $RMT_MASTER の値を 1 にします。

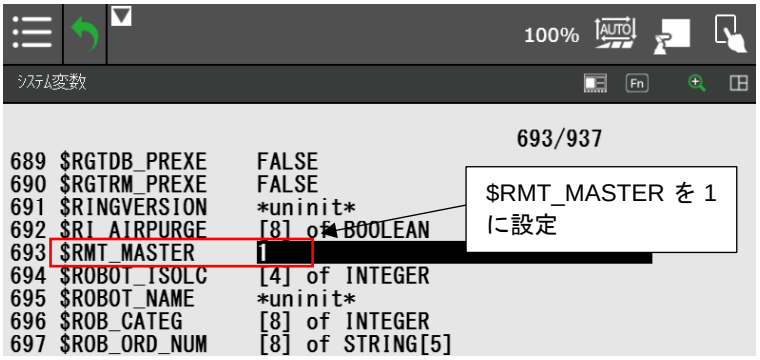


図 14.3.1 (h) $RMT_MASTER 設定

(4) TP スイッチを無効にします。

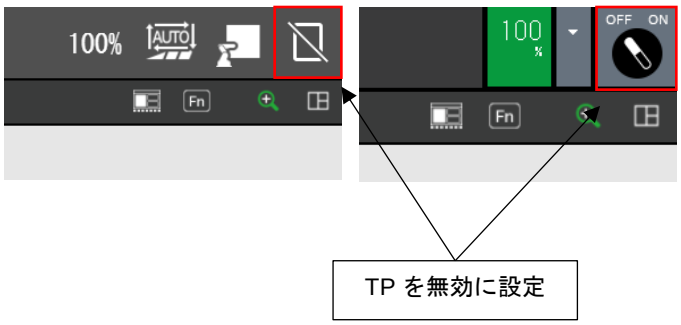


図 14.3.1 (i) TP 無効

14.3.2 KAREL プログラムのビルド

作成した KAREL プログラム(.kl) をビルドし、バイナリファイル(.pc) とルーチンスタックファイル(.rs) を生成するための手順を以下に示します。

手順 14-3 KAREL プログラムのビルド

手順

1. プロジェクトを選択して右クリックし、「プロパティ(R)」を選択します。ビルド対象のバージョンを「V9.40-1」に変更します。

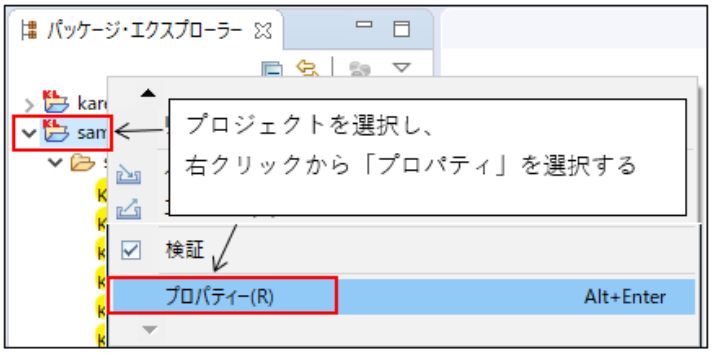


図 14.3.2 (a) プロジェクトのプロパティ



図 14.3.2 (b) ビルド対象のバージョン変更

2. 対象のプロジェクトを選択して、Eclipse のメインメニューから「プロジェクト(P)」－「プロジェクトのビルド(B)」の順に選択します。ビルドに成功すると、output フォルダにバイナリファイル(.pc) とルーチンスタックファイル(.rs) が生成されます。

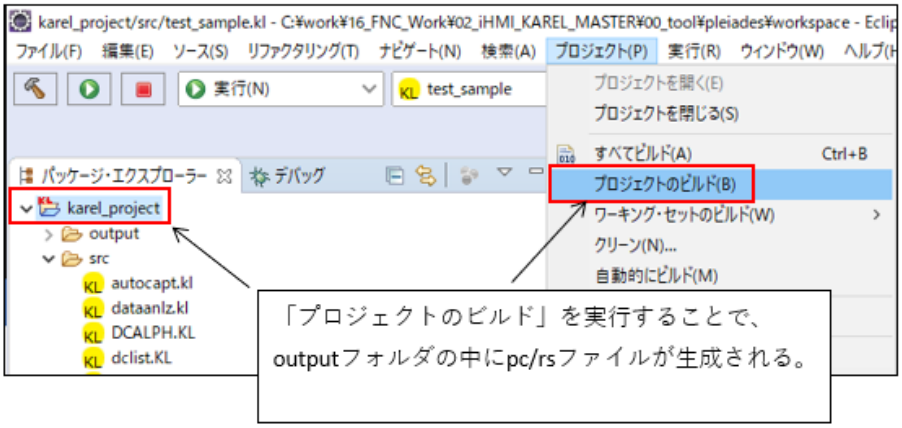


図 14.3.2 (c) KAREL プロジェクトのビルド

3. ビルド時にエラーが発生した場合は、コンソールに出力されたエラー内容を元に下記対応を行います。

表 14.3.2 (a) エラーの原因と対策

No.	エラー内容	原因	対策
1	'ktrans.exe' は、内部コマンドまたは外部コマンド、操作可能なプログラムまたはバッチファイルとして認識されていません。	ROBOGUIDE に内蔵されるビルド用ツールへのパスが環境変数に登録されていません。	環境変数に ktrans.exe が格納されているパスを登録します。64bitOS Windows 10 の場合は'C:Program Files (x86) ¥FANUC¥WinOLPC¥bin'です。

14.3.3 KAREL プログラムのデバッグ実行

ビルドで生成したバイナリファイル(.pc)を ROBOGUIDE / ロボット制御装置にロードし、デバッグを行います。デバッグ実行手順を以下に示します。

手順 14-4 KAREL プログラムのデバッグ実行

手順

1. 事前に対象の KAREL ファイルをビルドし、output フォルダにバイナリファイル(.pc) とルーチンスタックファイル(.rs) を生成しておきます。

2. Eclipse のメインメニューから「ウィンドウ(W)」－「パースペクティブ(R)」－「パースペクティブを開く(O)」－「その他(O)」を選択します。

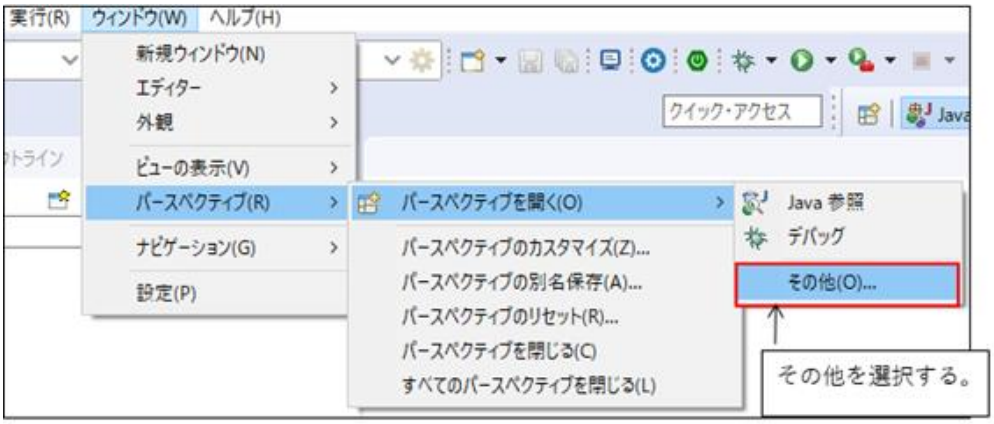


図 14.3.3 (a) Eclipse のパースペクティブ

3. パースペクティブダイアログで「デバッグ」を選択し、「開く(O)」をクリックします。Eclipse のメニューバーにデバッグメニューが追加されます。

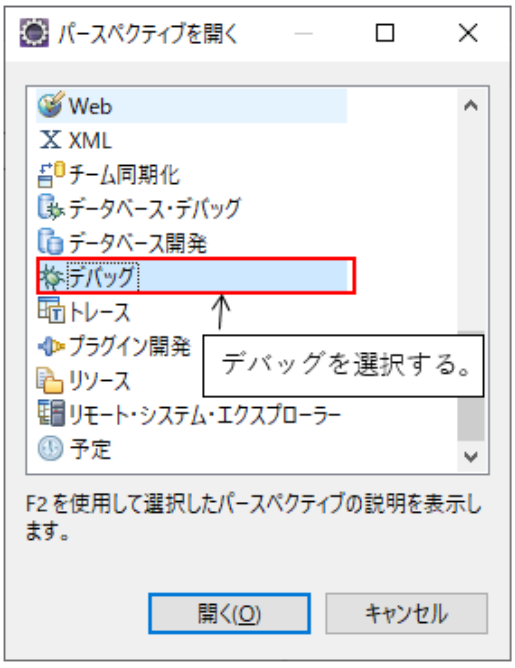


図 14.3.3 (b) パースペクティブダイアログ

4. KCL コマンドエラーが発生しないよう、ROBOGUIDE／ロボット制御装置上のタブレット TP 画面から「設定」－「ホスト通信」－「HTTP 認証」の画面で KCL をアンロックにします。

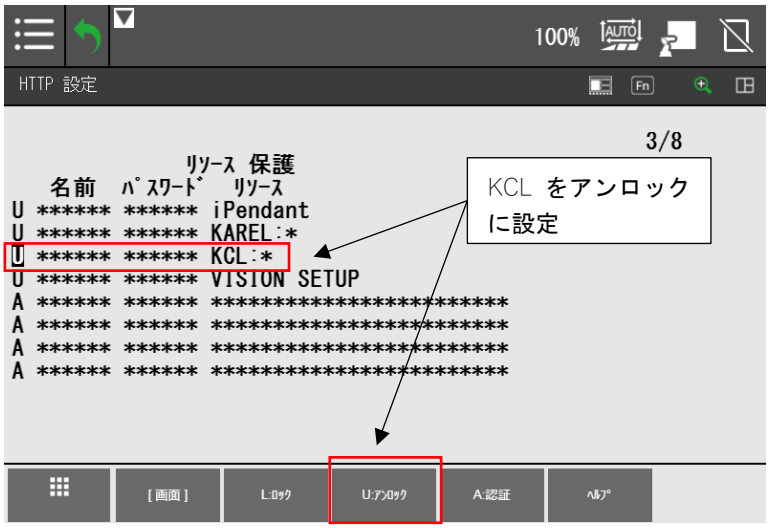


図 14.3.3 (c) KCL のアンロック

5. 「実行」アイコンのプルダウンメニューから「実行の構成(N)」をクリックし、デバッグ用のランチャーを作成します。

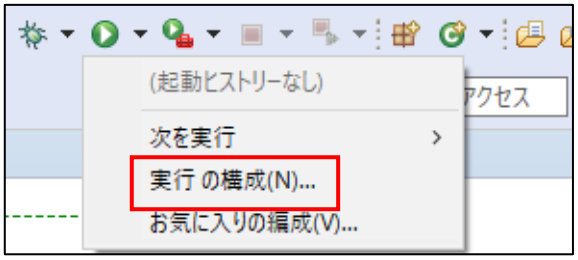


図 14.3.3 (d) 実行の構成

6. 「FANUC KAREL」を右クリックして「新規作成(W)」を選択し、デバッグ用の実行プログラムを新規作成します。

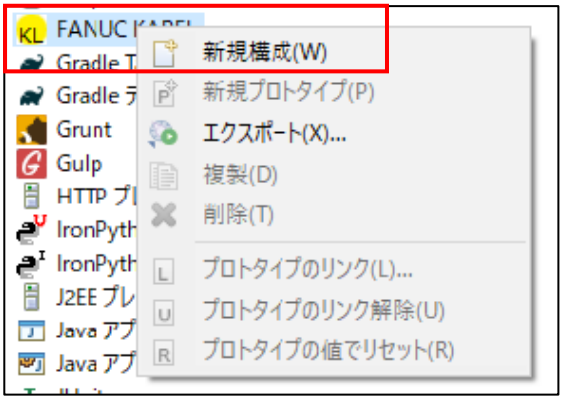


図 14.3.3 (e) デバッグ用実行プログラムの新規作成

7. 任意の名前を入力し、ランチャーを作成します。

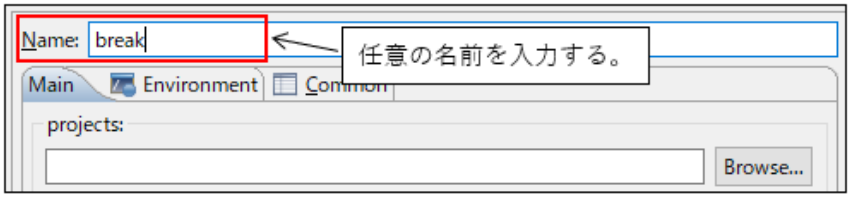


図 14.3.3 (f) デバッグ用ランチャー作成

8. 「projects」の「Browse」ボタンをクリックし、プロジェクトを選択します。

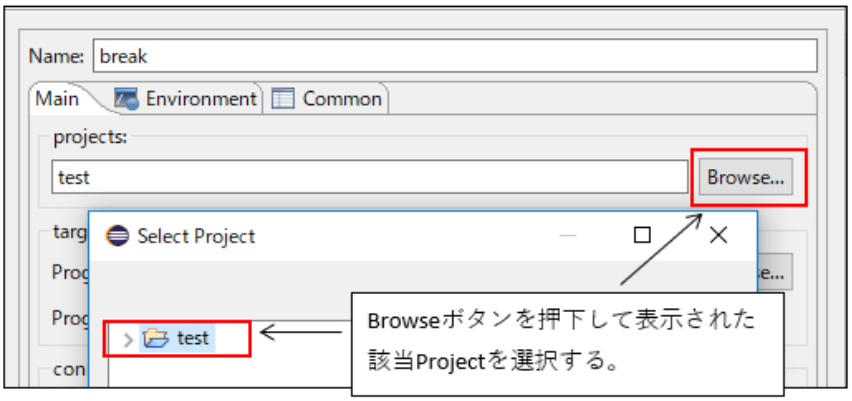


図 14.3.3 (g) プロジェクト選択

9. 「target programs」の「Browse」ボタンをクリックし、「Select File」からファイル名を選択します。ファイルを選択すると、ファイル名とプログラム名が自動で入力されます。

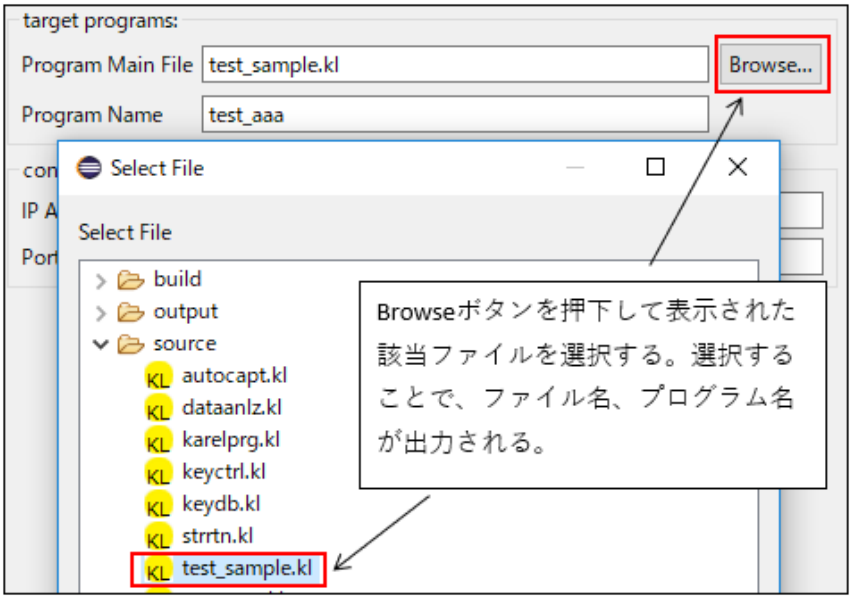


図 14.3.3 (h) ファイル名選択

10. 接続先サーバを設定します。
    - IP アドレス: ROBOGUIDE の場合は「localhost」、ロボット制御装置の場合はロボット制御装置の IP アドレスを設定します。
    - ポート番号: 標準では「80」を設定します。ポート番号を変更した場合はその番号を設定します。
    - FTP ポート番号: 標準では「21」を設定します。FTP ポート番号を変更した場合はその番号を設定します。

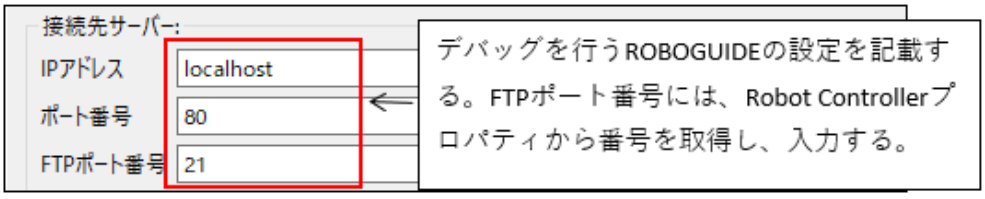


図 14.3.3 (i) 接続先サーバの設定

11. ユーザが ROBOGUIDE / ロボット制御装置のパスワード保護を行っている場合、FTP ユーザ名と FTP パスワードを入力してください。

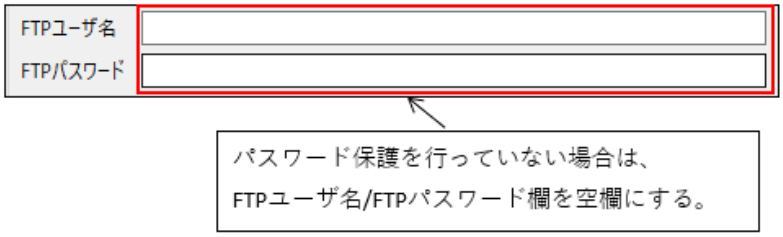


図 14.3.3 (j) FTP ユーザ名と FTP パスワードの入力

12. デバッグを行う KAREL ファイルを Eclipse 上で開き、ruler 上の任意の場所にダブルクリックし、ブレークポイントを設定します。

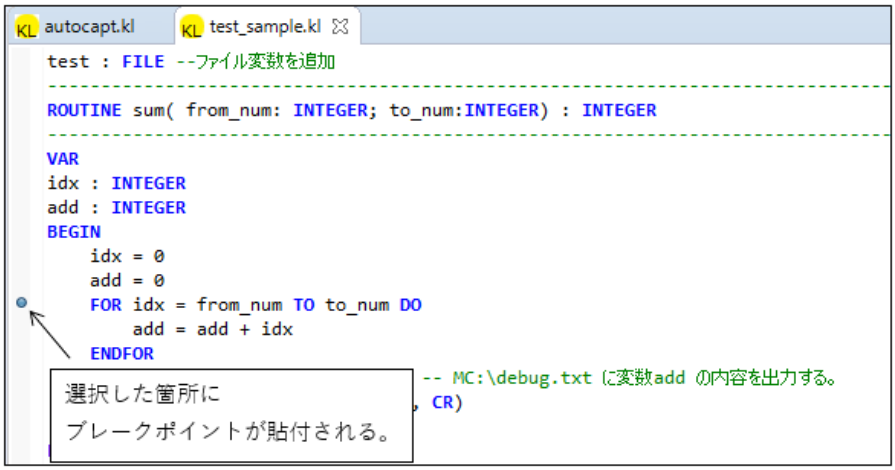


図 14.3.3 (k) ブレークポイントの設定

13. 作成したランチャーを実行します。ランチャーを実行することで、対象の ROBOGUIDE／ロボット制御装置にバイナリファイル(.pc)がロードされ、対象のプログラムが実行されます。

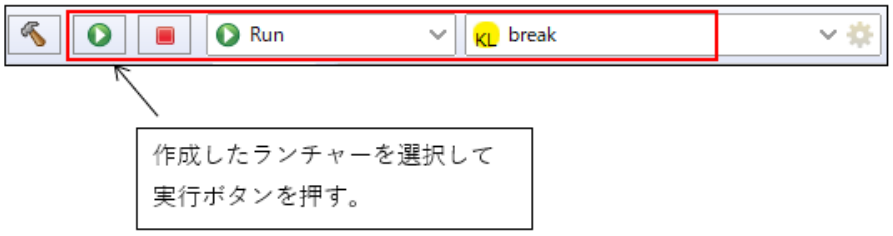


図 14.3.3 (l) デバッグ用ランチャーの実行

14. ビルドして生成されたバイナリファイルを ROBOGUIDE／ロボット制御装置にロードすると、以下のダイアログが表示されます。その場合、OK ボタンを押下した後、ROBOGUIDE／ロボット制御装置をコールドスタートし、再度デバッグを実行してください。

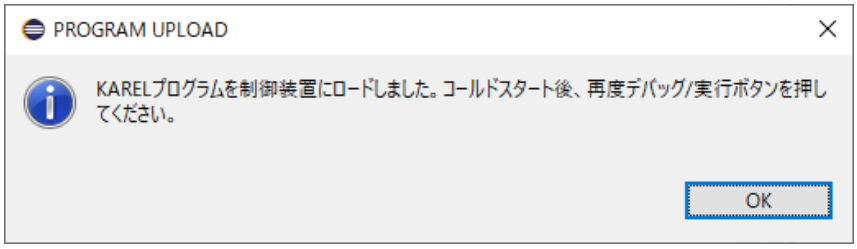


図 14.3.3 (m) KAREL プログラムロードの確認画面

15. プログラムが実行されると、Eclipse の View 上に実行中の KAREL ファイルが開き、ブレークポイントで停止している行が表示されます。

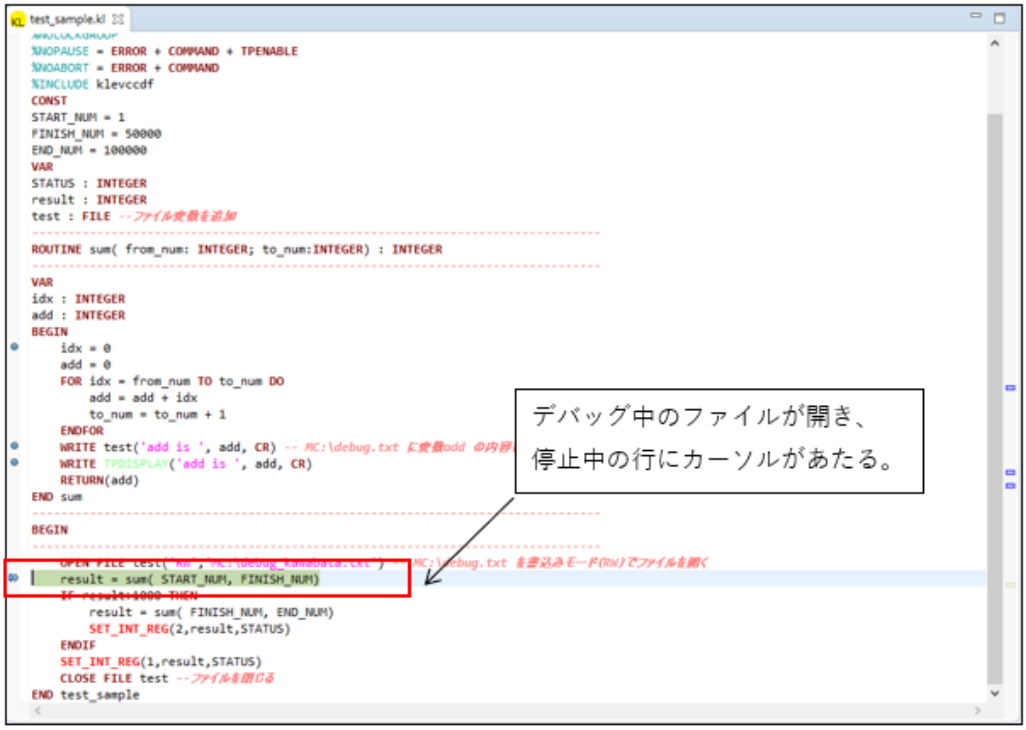


図 14.3.3 (n) ブレークポイントで停止した行

16. デバッグウィンドウに実行中のプログラム名と停止行が表示されます。停止位置がルーチン内の場合、ルーチン呼び出し行とルーチン内の停止行が表示されます。



図 14.3.3 (o) 実行中のプログラム名と停止行

17. 「KAREL Debugger」を選択すると「再開」ボタンと「終了」ボタンが有効になり、「Thread」を選択すると「再開」ボタン、「終了」ボタン、「Step Over」ボタンが有効になります。

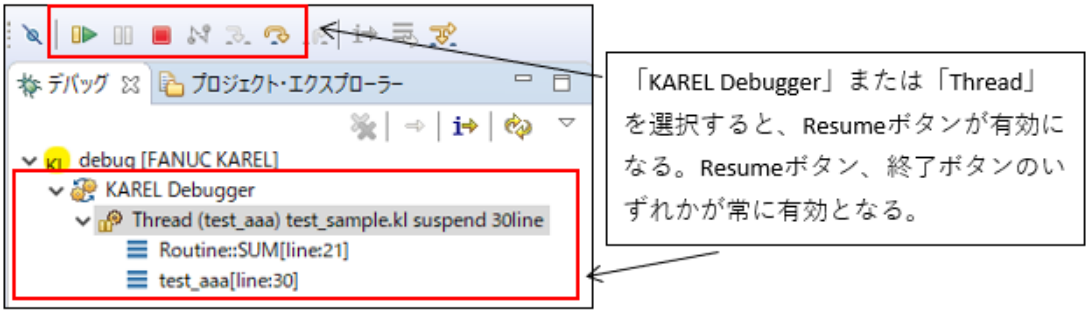


図 14.3.3 (p) 再開ボタンと終了ボタン

18. 「再開」ボタンをクリックし、プログラムを再開すると次のブレークポイントで停止します。

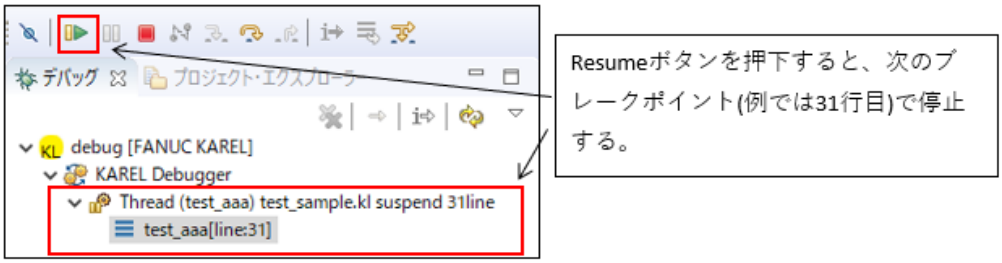


図 14.3.3 (q) 再開ボタン

19. 「ステップオーバー」ボタンをクリックすると、次の行で停止します。



図 14.3.3 (r) ステップオーバボタン

20. 「終了」ボタンをクリックするか、プログラムが終了した場合は、Thread が終了します。



図 14.3.3 (s) Thread 終了

14.3.4 KAREL デバッグ中の変数値を確認する方法

デバッグ実行中の KAREL プログラムの変数の値を確認したい場合、プログラムがブレークポイントで停止中にプログラム名やルーチン名を選択します。実行中プログラムのグローバル変数値やルーチン内のローカル変数値が表示されます。

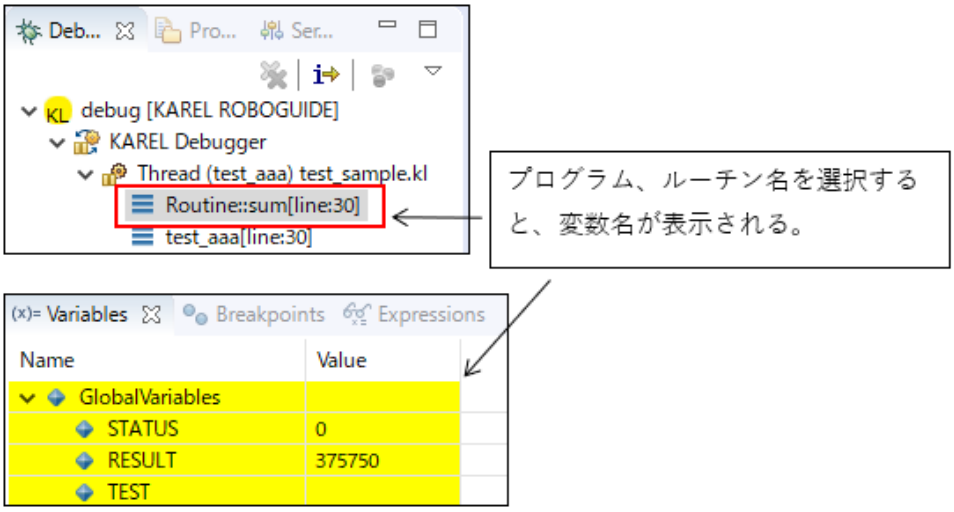


図 14.3.4 (a) KAREL デバッグ中の変数値を確認する方法

14.3.5 KAREL デバッグ中の変数値を変更する方法

デバッグ中のプログラムの変数の値を変更したい場合、プログラムがブレークポイントで停止中に、表示されている変数名の Value 欄の値を変更できます。

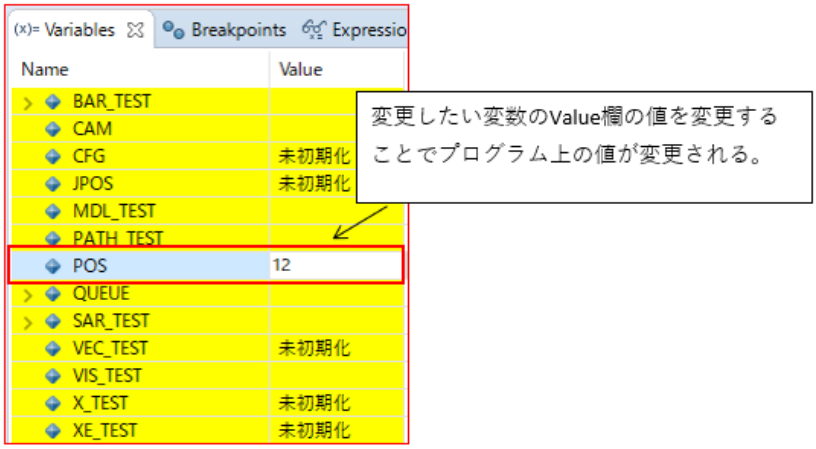


図 14.3.5 (a) KAREL デバッグ中の変数値を変更する方法

14.3.6 KAREL エディタ変更

View に表示している KAREL プログラムの文字色をお好みの文字色に変更できます。お好みの文字色に変更する方法を以下に示します。

手順 14-5 KAREL エディタ変更

手順

1. Eclipse のメインメニューから「ウィンドウ(W)」－「設定(P)」を選択し、設定のダイアログボックスを開きます。

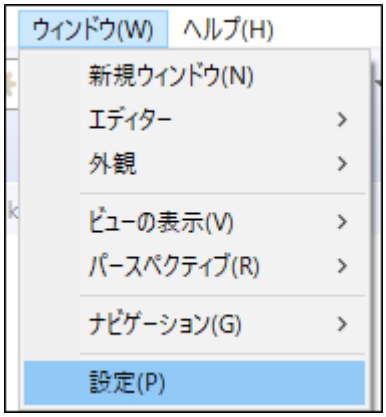


図 14.3.6 (a) 設定ダイアログボックス

2. フィルタ入力欄に「KAREL」と入力します。



図 14.3.6 (b) フィルタ入力

3. 表示された KAREL エディタ色付け設定項目で自由に色変更が行えます。



図 14.3.6 (c) KAREL エディタ色付け設定

14.4 iHMI コンポーネント開発環境の利用手順

14.4.1 iHMI コンポーネント開発環境準備

以下の手順で iHMI コンポーネント開発環境を準備します。

手順 14-6 iHMI コンポーネント開発環境の準備

手順

1. Eclipse のメインメニューから「ファイル(F)」－「新規(N)」－「その他(O)」の順に選択します。



図 14.4.1 (a) Eclipse の起動方法

2. ウィザードより「FANUC」内の「iHMI プロジェクト」を選択します。

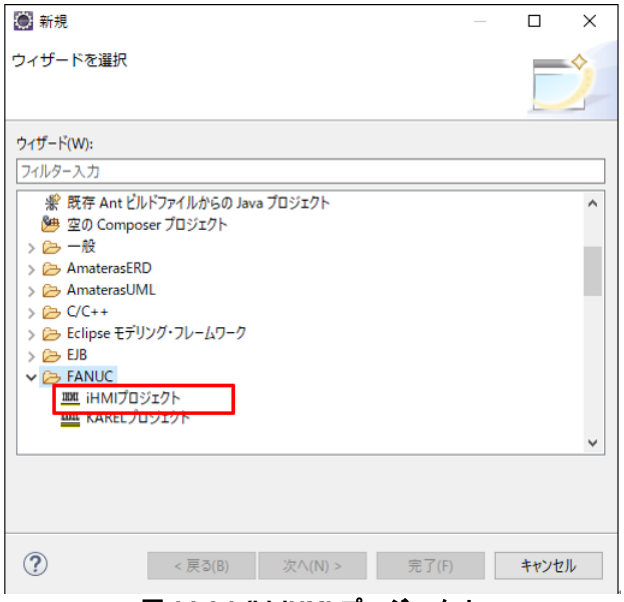


図 14.4.1 (b) iHMI プロジェクト

3. プロジェクト名欄に任意のプロジェクト名を入力します。

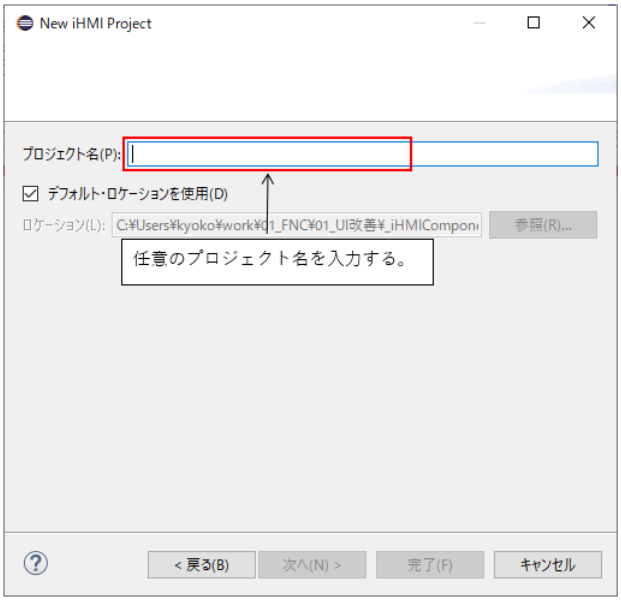


図 14.4.1 (c) iHMI プロジェクト名の入力

4. 以下のようなフォルダ構成の WorkSpace が作成されます。

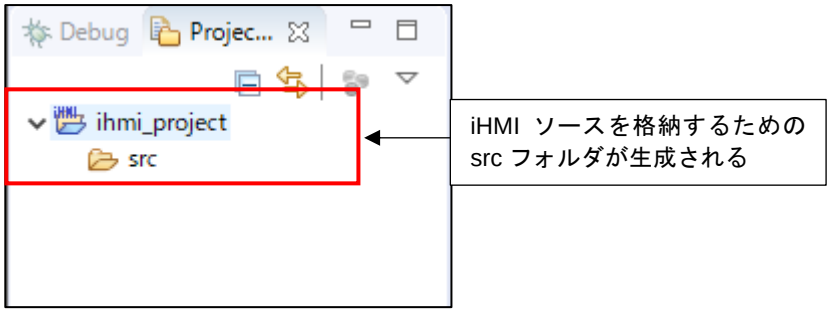


図 14.4.1 (d) iHMI ソースの格納フォルダ

14.4.2 iHMI コンポーネントプログラムをビルドする方法

iHMI コンポーネント開発環境は、ベースファイル*.ishtm(*.ihtml)ファイルと辞書ファイルから、ロボット制御装置に組み込むための*.html ファイルまたは*.stm(*.htm)ファイルを生成します。ベースファイル*.ishtm(*.ihtml)ファイルの作成方法は本書 7 章「プラグイン画面作成」、ビルドで生成された*.stm(*.htm)ファイルの利用方法は本書 7 章「プラグイン画面作成」、および 8 章「プラグイン命令作成」を参照してください。

作業プロジェクトのフォルダを右クリックして「プロジェクトのビルド」を選択するとビルドが実行され、ロボット制御装置に組み込むための*.stm(*.htm)ファイルが output フォルダ直下に生成されます。また、生成された output フォルダ内のファイルを 1 つにまとめた*.ipl ファイル(プラグインパッケージ)も生成されます。*.ipl ファイルはプラグインソフトをロボット制御装置にインストールする際に使用します。

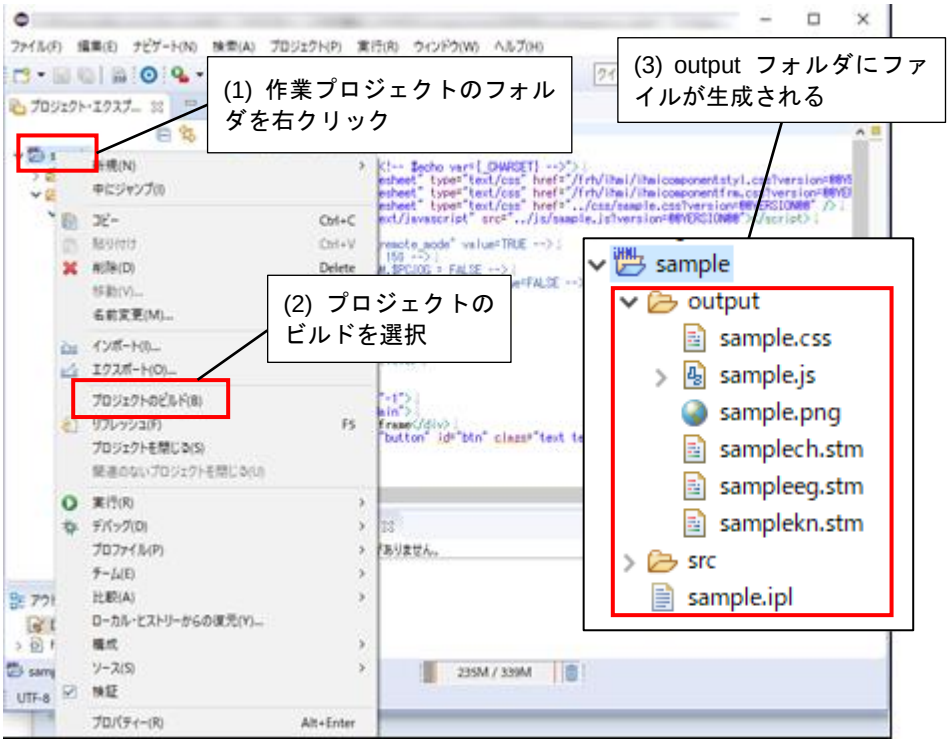


図 14.4.2 (a) iHMI コンポーネントプログラムをビルドする方法

14.4.3 Eclipse クリーン実行

ファイルが更新されていない場合、ビルドは実行されません。強制的にビルドを実行する場合は、以下の手順で Eclipse 機能のクリーンを実行してください。

手順 14-7 Eclipse クリーン実行

手順

1. Eclipse のメインメニューから「プロジェクト(P)」－「クリーン(N)」を選択し、クリーンダイアログボックスを表示します。

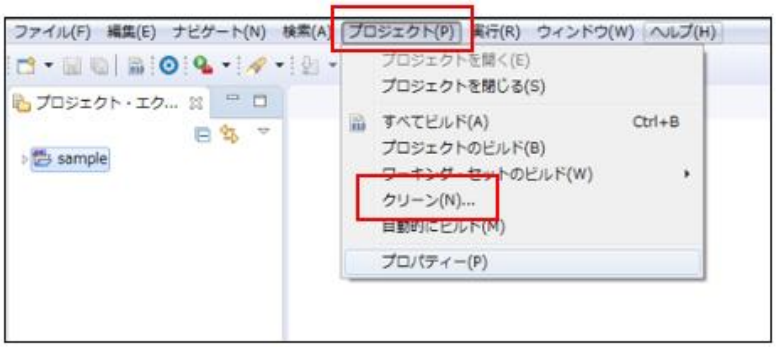


図 14.4.3 (a) クリーンダイアログボックスの表示

2. クリーンダイアログボックスで「クリーン(C)」ボタンをクリックします。

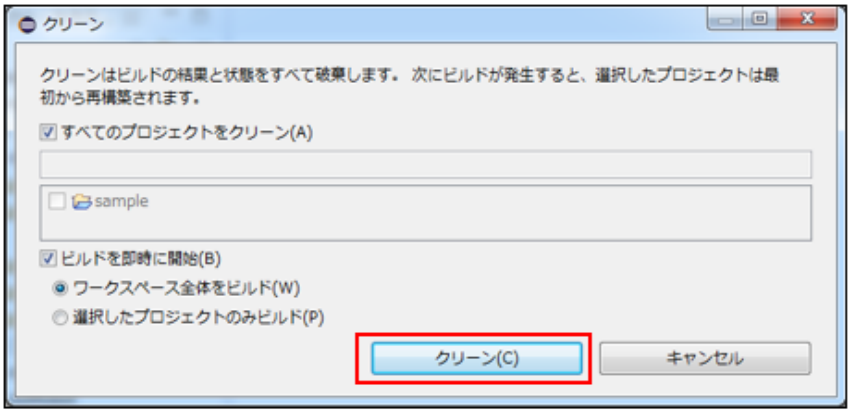


図 14.4.3 (b) クリーン実行

14.4.4 文字コードを変更する手順

Eclipse のソースファイルで文字化けする場合は、次の手順で文字コードを変更してください。

手順 14-8 文字コードの変更

手順

1. 対象のソースファイルを右クリックして、「プロパティー」を選択し、ダイアログボックスを表示します。

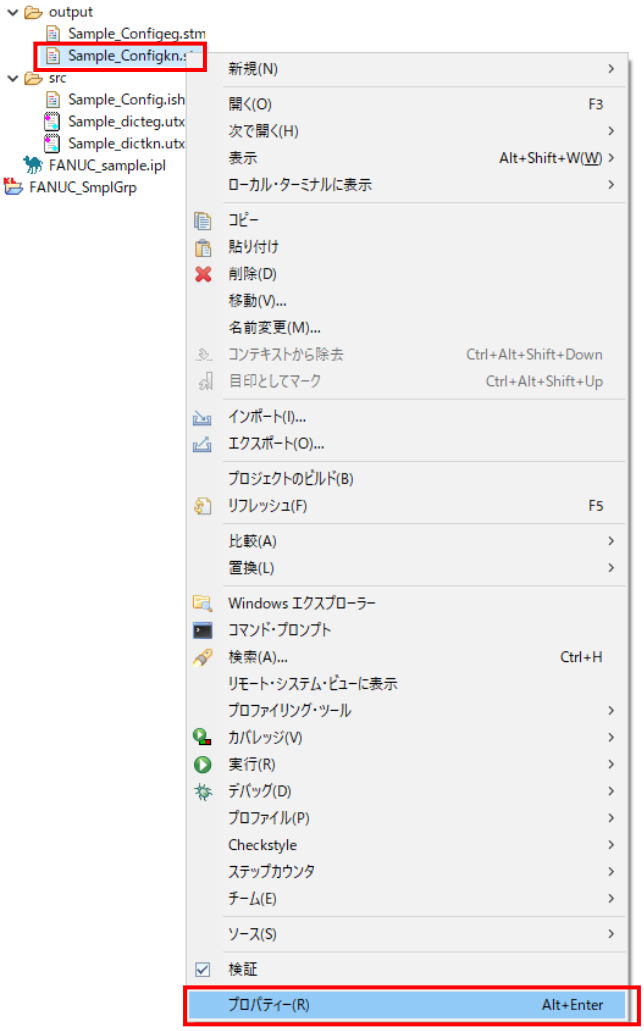


図 14.4.4 (a) プロパティーのダイアログボックスの表示

2. プロパティのダイアログボックスで「リソース」を選択し、「テキスト・ファイル・エンコード」で適切な文字コードを指定します。初期状態では「Shift-JIS」が表示されませんが、直接「Shift-JIS」と入力することで「Shift-JIS」を指定できます。一度直接入力すると、次回からはリストに表示されます。

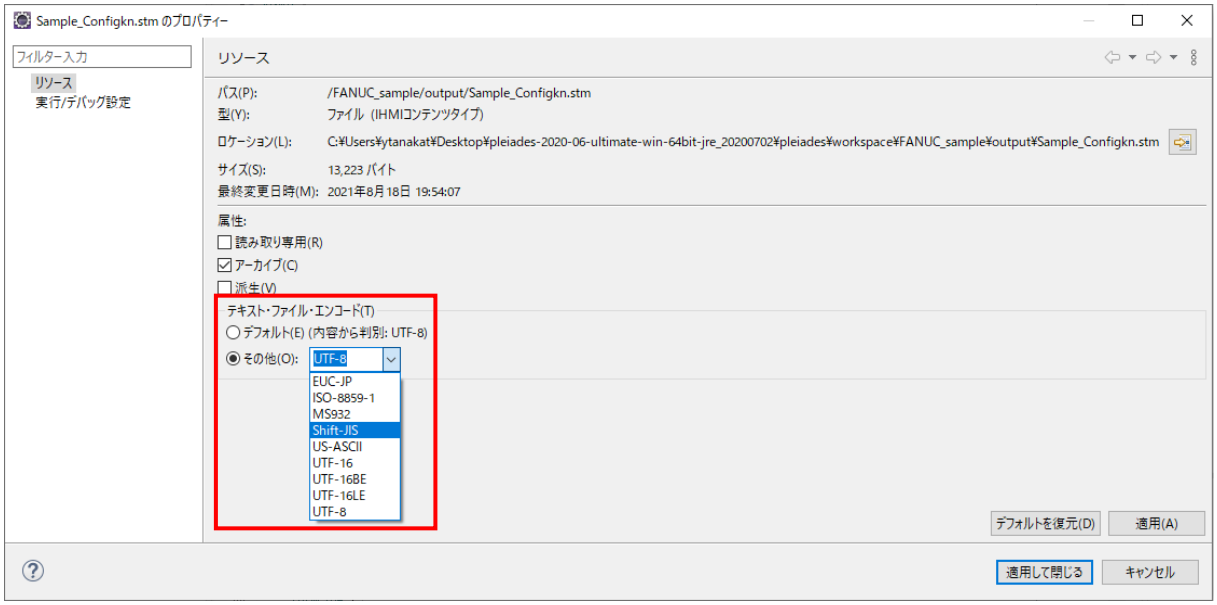


図 14.4.4 (b) テキスト・ファイル・エンコードの選択

3. 文字コードの指定が完了したら、「適用して閉じる」を押して設定を適用します。

15 CRX シリーズ機構部

本章では、FANUC Robot CRX-10iA、CRX-10iA/L 機構部の仕様を説明します。

15.1 機構部仕様

ロボット機構部仕様を下表に示します。

表 15.1 (a) 仕様一覧 (注 1)

項目		仕様一覧	
機種名		CRX-10*i*A	CRX-10*i*A/L
動作形態		多関節形ロボット	
制御軸数		6 軸 (J1, J2, J3, J4, J5, J6)	
リーチ		1249 mm	1418 mm
設置形式		床置、天吊、傾斜角 (注 2)	
動作範囲	J1 軸	380° (6.63 rad)	360° (6.28 rad)
	J2 軸	360° (6.28 rad)	
	J3 軸	570° (9.95 rad)	540° (9.95 rad)
	J4 軸	380° (6.63 rad)	
	J5 軸	360° (6.28 rad)	
	J6 軸	380° (6.63 rad)	
最大動作速度 (注 3)		1000mm/s (注 4) (最大 2000mm/s (注 5))	
手首可搬質量		10kg	
手首部許容負荷モーメント	J4 軸	34.8Nm	
	J5 軸	26.0Nm	
	J6 軸	11.0Nm	
手首部許容負荷イナーシャ	J4 軸	1.28kg.m^2	
	J5 軸	0.90kg.m^2	
	J6 軸	0.30kg.m^2	
位置繰り返し精度 (注 6)		±0.04mm	
ロボット機構部質量		40kg	40kg
防塵防滴構造 (注 7)		IP67 準拠	
騒音		70dB 以下 (注 8)	
設置条件		周囲温度： 運転時 0～45℃(注 9) 輸送、貯蔵時 -10～+60℃ 周囲湿度： 通常: 75%RH 以下 (結露しないこと) 短期(1 ヶ月以内): 95%RH 以下 許容高度： 海抜 1000m 以下 腐食性ガスのないこと(注 10) 振動加速度：4.9m/s^2 (0.5G) 以下 (注 11) 火気がない環境であること	

注

1 使用範囲内でご使用される場合でも、動作プログラムによっては減速機寿命が短くなる場合や、オーバヒートアラームが発生する場合があります。ロボット導入前の詳細検討には、ROBOGUIDE を活用してください。
2 全ての設置形式において動作領域に制限はありません。
3 短い動作距離では各軸の最高速度に到達しないことがあります。
4 周辺との挟み込みも考慮し、システムのリスクアセスメントに応じて、動作速度を決める必要があります。
5 安全センサ(別置)による領域監視時
6 ISO9283 に準拠しています。
7 IP(保護構造等級)の定義について
IP67 の定義
6 = 耐塵型：粉塵が内部に侵入しない。
7 = 浸漬に対する保護：規定の圧力、時間で水中に浸漬しても有害な影響を受けない。
8 この値は ISO11201 (EN31201)に従って計測した A 荷重等価騒音レベルです。計測は下記の条件で行っております。
- 最大荷重
- 自動運転(AUTO モード)
9 0℃に近い低温環境でロボットを使用する場合、もしくは休日や夜間に 0℃を下回るような環境でロボットを長時間休止させた場合、運転開始直後は駆動部の抵抗が大きいために、衝突検出アラーム(サーボ-050)などが発生することがあります。このような場合は、数分間の暖機運転を行うことを推奨します。
10 高温、低温環境、振動、塵埃、切削油などの濃度が高い環境で使用する場合には、当社まで相談してください。
11 床の振動やハンドの振動などがロボットに加わると、ロボットが停止する場合があります。

15.1.1 ロボットベース寸法

下図にロボットベースの寸法を示します。

⚠ 注意

ロボット設置面の平面度は 0.5mm 以下、傾きは 0.5°以下にしてください。ロボットベース設置面の平面度が悪いと、ロボットベースの破損やロボットの性能が十分に発揮できない場合があります。

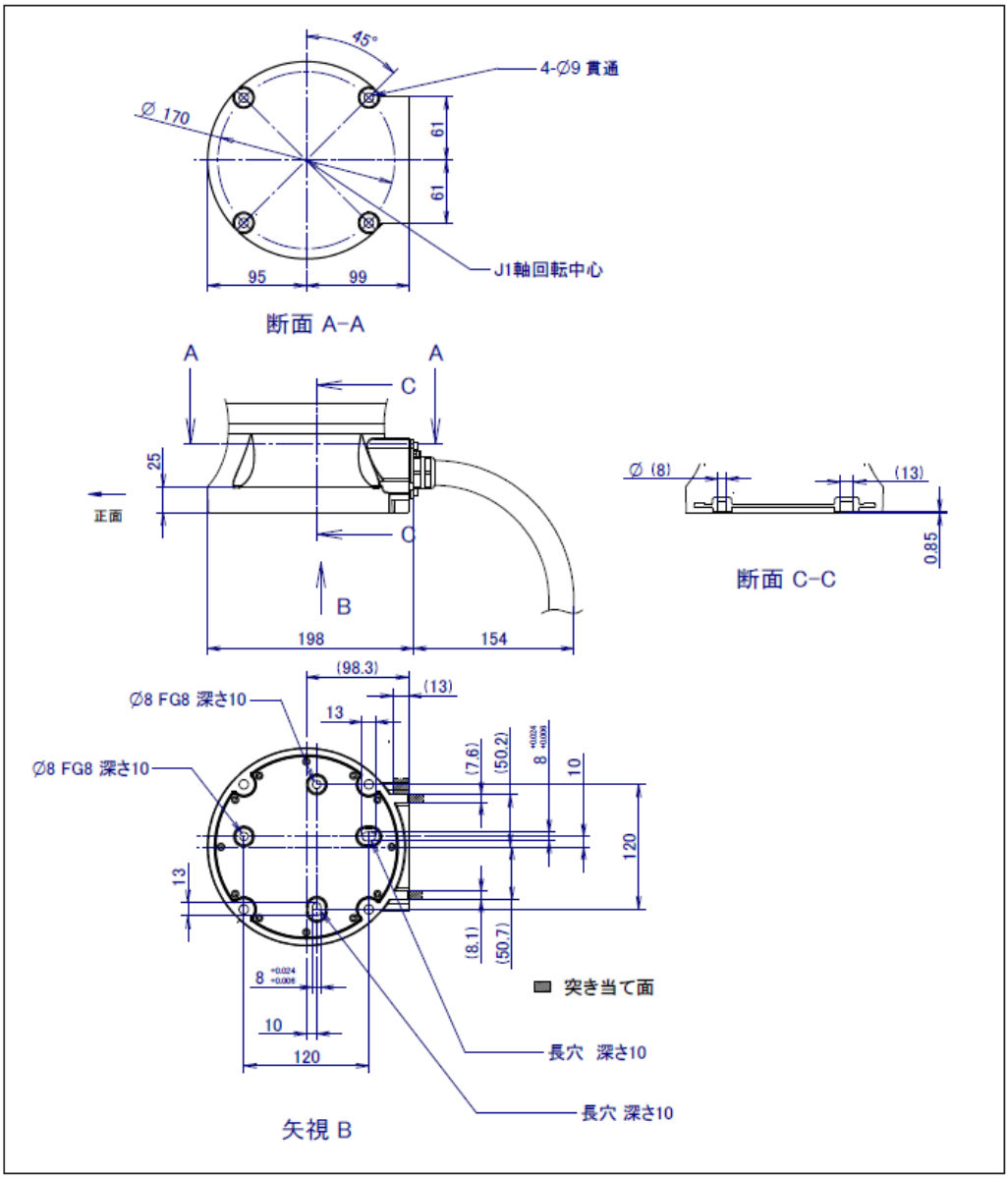


図 15.1.1 (a) ロボットベース寸法 (背面分線盤の場合)

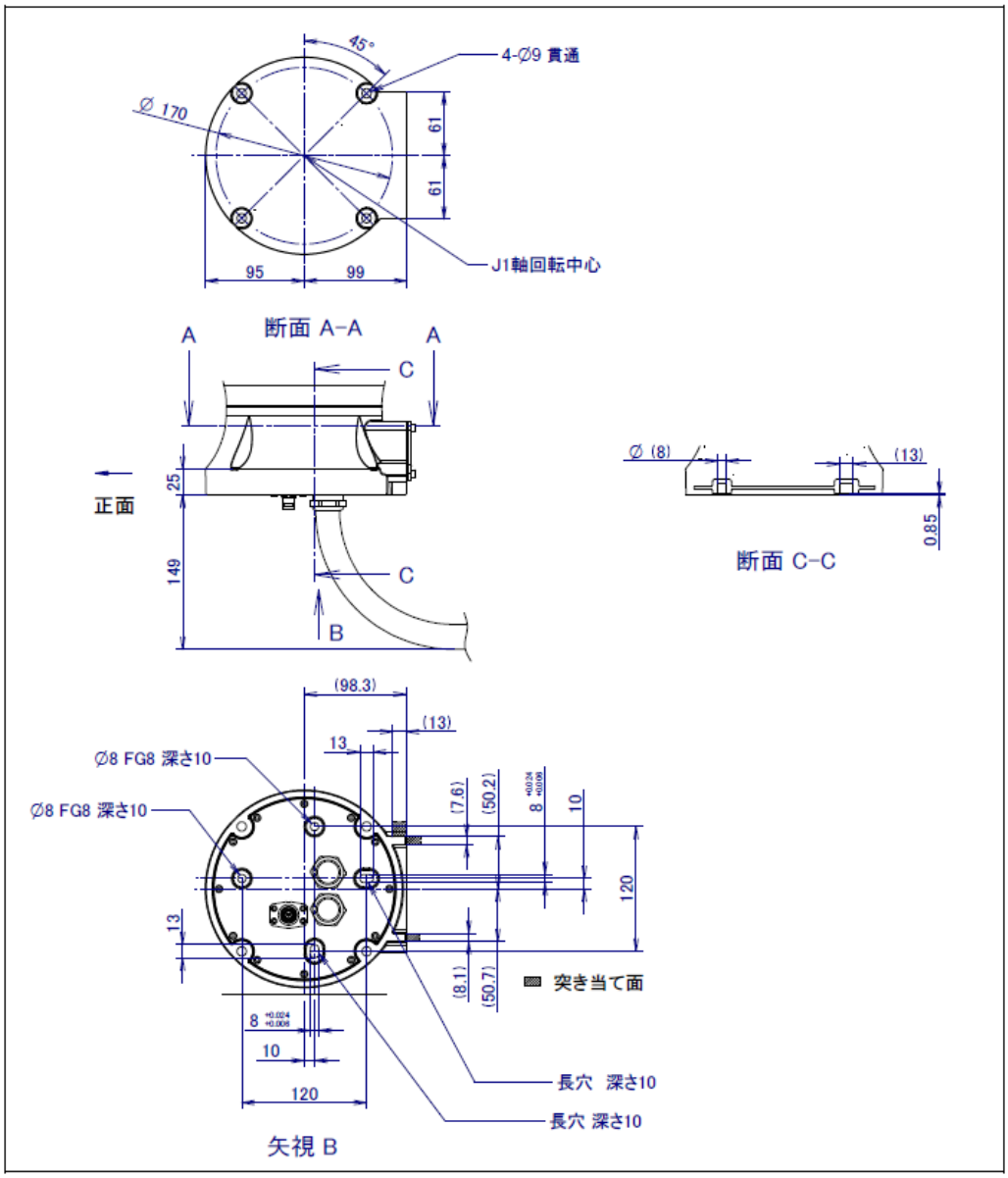


図 15.1.1 (b) ロボットベース寸法 (底面分線盤の場合)

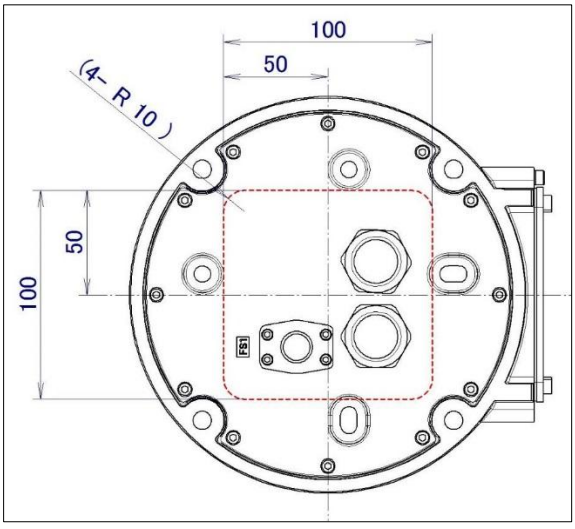


図 15.1.1 (c) 架台穴寸法 (底面分線盤)

15.1.2 ロボットベースに作用する力およびモーメント

図 15.1.2 (a)、表 15.1.2 (a)にロボットベースに作用するモーメントを示します。

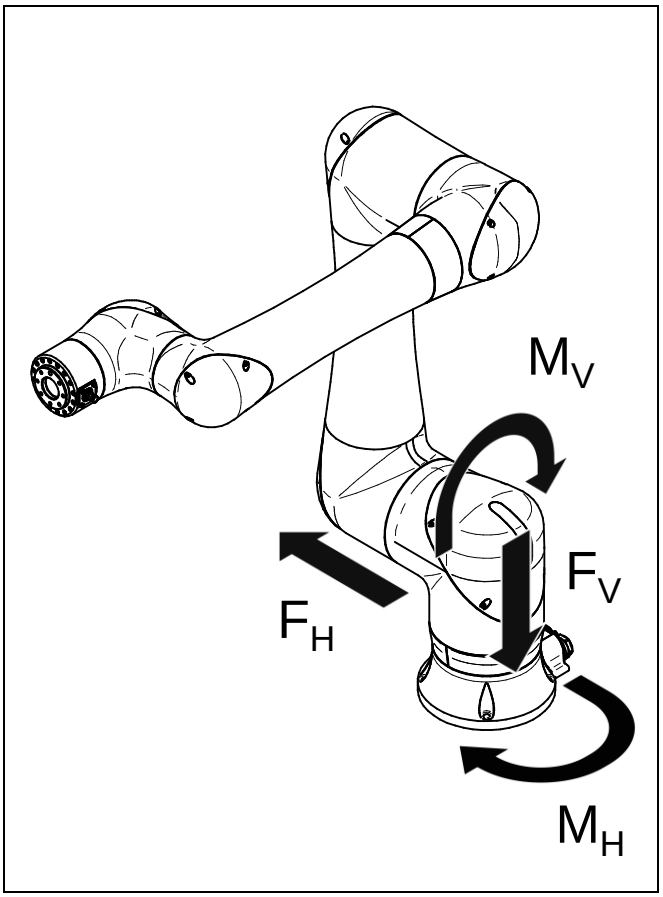


図 15.1.2 (a) ロボットベースに作用する力およびモーメント

表 15.1.2 (a) ロボットベースに作用する力およびモーメント

	垂直面モーメント M_V (Nm)	垂直方向の力 F_V (N)	水平面モーメント M_H (Nm)	水平方向の力 F_H (N)
静止時	310	490	0	0
加減速時	380	550	120	180
スムースストップ時	610	800	250	220

15.1.3 手首先端へのツールの取り付け

手首先端へのツール取付面を下図に示します。使用するボルトおよび位置決めピンは、タップおよびピン穴深さを十分考慮して長さを選択してください。

⚠ 注意
- ツール取付面に機器を取り付ける場合、インロー長さ以上の嵌め合いを行わないようにしてください。
- ツールを取り付けたとき、ツールはロボット自身に衝突するおそれや、ツールとロボットの間に、体の一部が挟まり、重傷を負うおそれがあります。

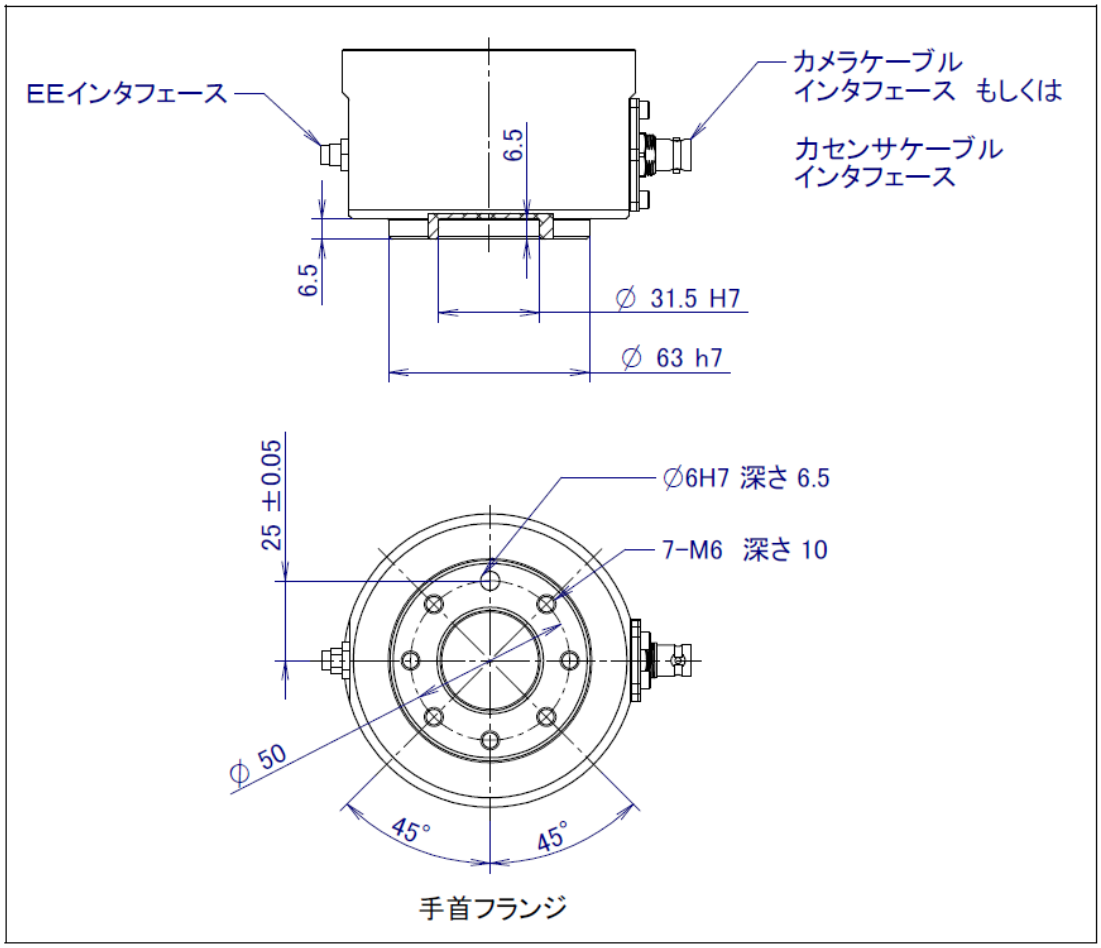


図 15.1.3 (a) ツール取付面 (CRX-10*i*A, CRX-10*i*A/L)

15.1.4 手首負荷条件

下図に手首部許容負荷線図を示します。

- 負荷条件がグラフの範囲内に入るようにしてください。
- 手首許容モーメント、手首許容イナーシャの条件を満たすよう使用してください。手首許容モーメント、手首許容イナーシャについては表 15.1 (a)を参照してください。

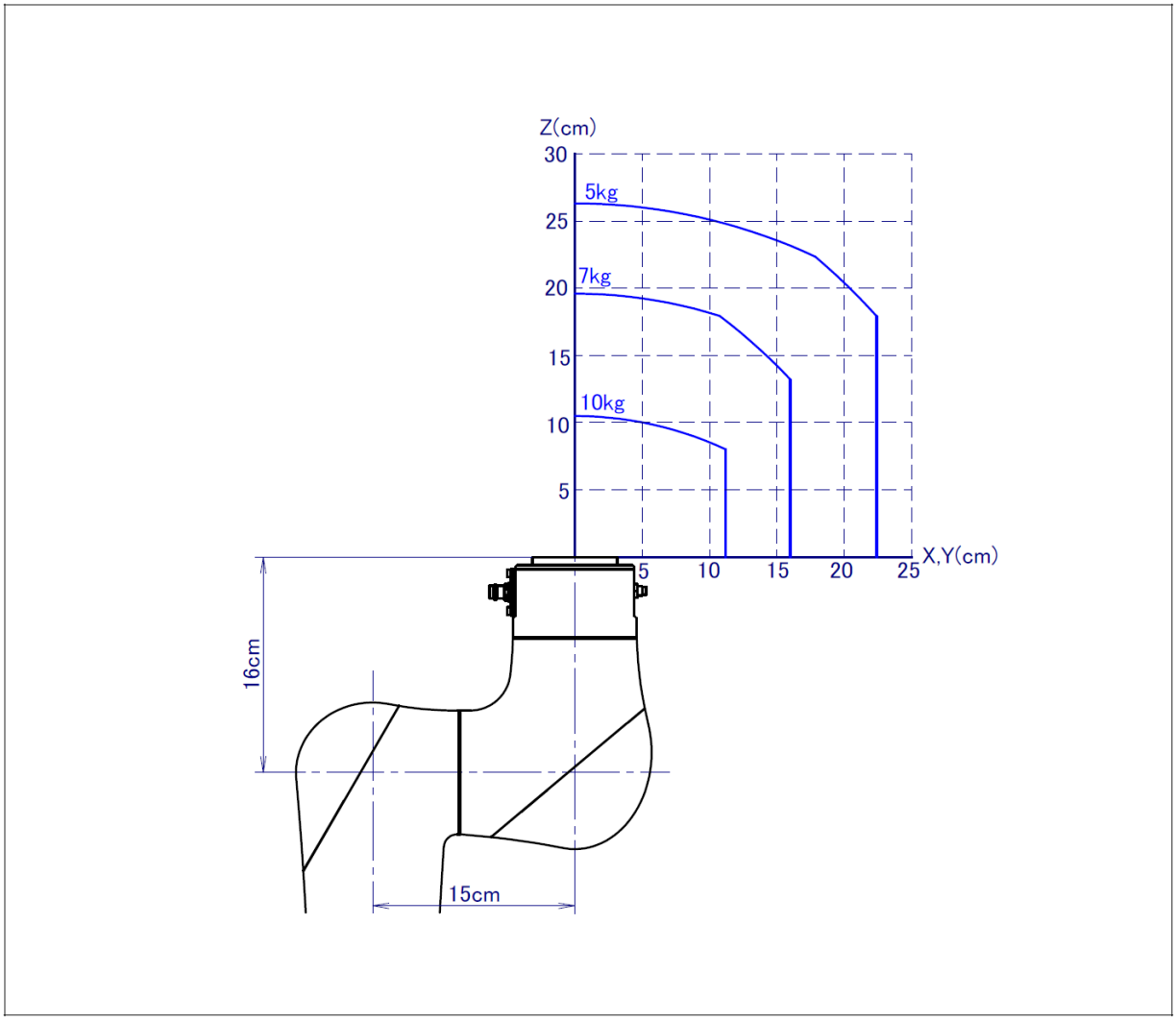


図 15.1.4 (a) 手首許容負荷線図 (CRX-10*i*A, CRX-10*i*A/L)

説明書改版履歴

版数	年月	変更内容
12	2025 年 10 月	・5.1.2 節, 台湾語、ドイツ語、フランス語、イタリア語、スペイン語の文字コードを追記 ・6.3 節, 台湾語、ドイツ語、フランス語、イタリア語、スペイン語の文字コードを追記 ・7.1.3 節, 台湾語、ドイツ語、フランス語、イタリア語、スペイン語の文字コードを追記 ・7.5 節, JavaScript API の記述修正、および、使用例を追記 ・11 章, SSET コマンドによるツール I/F 通信設定方法を追記
11	2025 年 1 月 13 日	・5.1.2 節, 中国語の文字コードを追記 ・5.2.6.3 節, SSET コマンドによる負荷切替距離、外力制限、無効信号の設定方法を追記 ・5.2.7 節, PLGMTOP コマンドの説明を追記 ・6.3 節, 中国語の文字コードを追記 ・7.1.3 節, 中国語の文字コードを追記 ・7.4.2 節, textboxInteger の inputType の注意事項を追記 ・7.4.3 節, textboxReal の inputType の注意事項を追記 ・7.5 節, JavaScript API の追加と記述更新 ・8.3 節, プラグイン命令の TP プログラムと KAREL プログラムの注意事項を追記 ・8.3.7 節, 負荷情報を設定する IPL_SETPLD の説明を追記 ・8.3.8 節, 負荷情報を取得する IPL_GETPLD の説明を追記 ・8.3.9 節, MAC アドレスを取得する IPL_GETMACADDR の説明を追記 ・8.5.1 節, プラグイン命令の引数に REAL 値を指定する場合の注意事項を追記 ・10.1.2 節, デジタル出力データの更新周期の記述更新 ・10.1.3 節, デジタル入力データの更新周期の記述更新 ・11.4 節, デジタル入出力データの更新周期の記述更新 ・11.7 節, ホスト-412 とホスト-428 の記述更新 ・12 章, CRX ツールチェンジ機能の説明を追加
10	2022 年 6 月 10 日	・5.2.3 節, RUNPROG/RUNPRGWT コマンドの記述修正 ・7.2.1 節, Perl 実行環境の記述更新 ・7.2.2 節, プラグインソフトの JavaScript と CSS ファイル更新時の注意事項を追記 ・7.4.5 節, listSelect & listSelectOption の注意事項追記 ・8.2 節, 命令定義 XML ファイルの init タグで指定する param/func 属性の記述更新 ・8.3.6 節, KAREL プログラムによるユーザソケットメッセージ機能の通信設定を追加 ・11.2 節, デジタル出力 1, 2 の注意事項と表 11.2 (a)を追記 ・11.8 節, ホスト-412, 425 の記述を更新 ・[ihmi_build] build_ihmi.bat 実行時に eg,kn,ch が常に出力されるように修正 ・[ihmi_build] スペースを含むディレクトリパスに build_ihmi.bat を置くとビルドエラーになる不具合修正
09	2021 年 12 月 24 日	・2 章, プラグインソフト開発環境の説明を追加 ・4 章, プラグイン定義アラームファイルの記述追加 ・5.1 節, ライセンス表示機能と改行コードについて注意事項を追記 ・5.1 節, PLGID, OPTION, REQUIRE コマンドについて注意事項を追記 ・5.1.3 節, PLGID, OPTION, REQUIRE コマンドについて注意事項を追記 ・5.1.4 節, PLGID, OPTION, REQUIRE コマンドについて注意事項を追記 ・5.2.2 節, LOAD コマンドに複数プログラム選択の注意事項を追記 ・5.2.3 節, RUNPROG/RUNPRGWT コマンドの注意事項を追記 ・5.2.5 節, SYSC コマンドの説明を追加 ・5.2.6 節, SSET コマンドの説明を追加 ・7.4.2 節, inputType で指定するソフトウェアキーボードの図を追加 ・8.2 節, 命令カテゴリに cat_plugin(プラグイン)を追加 ・8.3.5 節, サードパーティビジョンインタフェースの説明を追記 ・9.1 節, SDTL-084 アラームの記述を追記 ・10.1.5 節, RS485 通信ラインの終端抵抗について説明を追記 ・10 章, EE コネクタ接続の注意事項追記 ・11 章, EE コネクタ接続の注意事項追記 ・11.3 節, アナログ入力信号の自動割り付けについて説明を追記 ・11.4 節, シリアル通信用システム変数について記述追記

版数	年月	変更内容
		・11.6.1 節, TLIF_WRITE について記述更新 ・11.8 節, ホスト-412 の記述更新 ・12 章, I/O 端子台変換モジュールの記述追加 ・12 章, 周辺機器への電源供給には外部電源が必要な旨の注意事項を追記 ・13 章, Eclipse 開発環境の立ち上げ手順説明を修正 ・14 章, FANUC Robot CRX シリーズ機構部の説明を追加
08	2020 年 11 月 26 日	・4 章, ライセンス情報ファイルの記述追加 ・5.1 節, OPTION/REQUIRE コマンドの記述追加 ・6 章, 設定画面サンプルの更新 ・8.2 節, <program>タグの記述追加 ・8.3.3 節, プラグイン定義アラームの記述追加 ・8.5 節, sampleapp.ishtm が参照する js ファイルのパス修正 ・8.6.1 節, エラーコード一覧表を更新 ・9.1 節, インストール時のエラー表示の記述追加 ・10 章, ツールインタフェースの図を更新 ・10.1 節, EE コネクタ仕様情報を更新 ・11.1 節, 電源供給モードの注意事項の記述変更、表 11.1 (b)を追加 ・11.2 節, 電源供給モードの注意事項の記述変更 ・13 章, Eclipse プラグイン開発環境の章を追加
07	2020 年 7 月 30 日	・4 章, インストールパッケージ内のディレクトリ階層について注意事項追記、イメージファイルについての説明追記 ・5.1.1 節, 画像ファイルについて記述追加 ・5.2.4 節, DSET コマンドでシステム変数を設定する方法を追記 ・5.3 節, ツール座標系設定に関する記述追記 ・7.4.2 節, textboxInteger の最小値と最大値に関する記述追記 ・7.5 節, GI/GO, AI/AO の io_type 追記 ・7.6.1 節, <FanucMenu>タグの注意事項を追記 ・7.7 節, ROBOGUIDE での画面の確認方法を追記 ・8.2 節, <disp>タグの記述追加 ・8.3.1 節, 負荷設定命令を含む TP プログラムの注意事項追記 ・8.3.2 節, ローカルレジスタ対応の TP 命令更新 ・9.1 節, ツール側 EE コネクタ仕様追記 ・9.1.3 節, 表 9.1.3(a)の誤記訂正 ・10 章, 注意事項を追記 ・10.1 節, システム変数の仕様変更および新規追加 ・10.3 節, システム変数の仕様変更、注意事項を追記 ・10.4 節, システム変数の仕様変更および新規追加・削除 ・10.5 節, ツールインタフェース通信機能のシステム変数の設定方法を追記 ・10.6.1 節, 詳細説明の記述変更 ・10.6.3 節, 記述の追加 ・10.6.4 節, 記述の追加 ・10.7 節, 説明の記述追加・サンプルプログラムの変更 ・10.8 節, ツールインタフェース通信機能のアラーム追記 ・11 章, ロボット制御装置の周辺機器インタフェースを追記
06	2020 年 4 月 26 日	・4 章, 表 4 (a) に 9 章と 10 章を追加 ・5-8 章, TP/KAREL プログラムの命名規則変更 ・8 章, 8.3.2 節(TP プログラムのローカルレジスタ)追加 ・9 章, 図 9 (a) 追加 ・9.1.1 節, 保護ダイオードを使用してもらう場合の記載を追記 ・9.1.4 節, 電気的仕様の訂正 ・9.1.5 節, ボーレート仕様の追記 ・10 章, 新規追加
05	2020 年 3 月 27 日	・4-5 章, プラグイン情報の多国語対応機能追加。PLGID の最大文字数訂正 ・4 章, インストールパッケージの構成ファイル追加 ・6 章, 実装サンプル更新 ・4-8 章, ファイル命名規則追記、誤記訂正

版数	年月	変更内容
		・7.5 節, ihmi_iovalrd 記述追記 ・8.3.2 節, RI/RO(ツールインタフェース上の DI/DO)記述追記
04	2020 年 2 月 23 日	・中国語とアイコンタイプの記述を削除
03	2020 年 1 月 23 日	・CSV コマンド書式および 5 章注訂正、5.2.3 節追記、誤記訂正
02	2020 年 1 月	・誤記訂正
01	2019 年 12 月	・初版作成